

Message-Level Detection of Support-Team and Recruiter Impersonation on Consumer Platforms: A Layered Detector and Adversarial Benchmark

Sochima Ezema
Independent researcher
sochimaezema00@gmail.com

October 2026 (draft v0.1)

Abstract

Enterprise email security protects employees behind a corporate mail gateway. The people most exposed to employment and support-impersonation scams—students and early-career job seekers—are contacted *inside* consumer platforms, where no gateway sees the message. We present a dependency-free, explainable detector that works at the level of the individual message and its sender, and a public benchmark for comparing such detectors under cheap adversarial transformation. The detector has four layers: (1) deterministic lookalike checks built on Unicode UTS #39 confusable skeletons, mixed-script detection, and bounded edit distance; (2) a logistic-regression sender scorer over 21 named features trained on a synthetic generator spanning eight brands (holdout F_1 0.993); (3) a message-intent classifier with a rule baseline and a prompt-injection-hardened LLM wrapper; and (4) a 76-message, brand-parametric benchmark with ten label-preserving adversarial variants scored on per-layer recall, false-positive rate, injection hijack rate, and latency. With the rule backend, the full pipeline reaches recall 1.00 at FPR 0.00 on the clean corpus and recall ≥ 0.94 on every variant, while any single layer alone fails on at least one; on an out-of-sample corpus paraphrased by a language model it holds FPR 0.00 at recall 0.96. A Llama 3.1 8B backend reaches recall 1.00 on every variant and on the paraphrases, but at FPR 0.30–0.43 that no threshold on its confidence repairs, and it obeyed an appended injection on 5 of 46 attacks, each caught only by the deterministic overrule; a Gemma 3 27B backend lowers the FPR to 0.13 (0.17 on paraphrases) at recall 0.98 and obeyed no injection, and its remaining false positives are all genuine notices from trusted institutions that only the sender layers can vouch for. Three findings fall out: UTS #39 skeletons do not fold Cyrillic U+0442 or Greek U+03BA to ASCII, so skeleton equality alone misses a mixed-script brand lookalike; regex intent cues lose 37 points of recall to a single confusable glyph unless text is confusable-folded first (2 points after); and clause-level negation handling keeps a real employer’s “never send bank details by email” from reading as a payment request.

Code, weights, corpus, and per-message verdict dumps are released under the MIT license.

1 Introduction

A student receives a message from “Handshake Security Team” saying that unusual login activity was detected and that the account will be suspended unless they verify their identity at a link within 24 hours. A job seeker is told they have been selected, with no interview, for a \$45-per-hour remote data-entry role and should continue the conversation on Telegram. Both arrive through the platform’s own inbox or an address it exposed, not through a corporate mail server with a secure email gateway in front of it. The platform operator is the only party positioned to inspect the message before the recipient does, and usually has no mail-security stack to call.

The U.S. Federal Bureau of Investigation’s Internet Crime Complaint Center recorded 20,044 employment-scam complaints and \$264,223,271 in employment-scam losses in 2024, up from \$70,234,079 in 2023 and \$52,204,269 in 2022, a more-than-threelfold rise in one year [4]. Existing open-source defences, however, operate at the wrong level for this setting. Domain-permutation tools such as `dnstwist` [13] and confusable detectors such as `confusable_homoglyphs` [5] judge a registered domain, not a message. Fake-job-posting classifiers [14] judge a public listing, not a direct message to one recipient. LLM phishing benchmarks [12, 15] are email-centric and assume a mail-gateway vantage point with headers and attachments.

Threat model. The victim is a student or early-career job seeker on a consumer career platform. The attacker can send a message with a chosen display name, a chosen sending address or handle, a subject, a body, and links. The attacker may (a) imitate the platform’s own support, security, admin, or trust team; (b) imitate a recruiter or employer; (c) ask for credentials or one-time codes; or (d) redirect a payment. The attacker can cheaply obfuscate the message with confusable glyphs, zero-width characters,

digit substitutions, letter spacing, synonyms, lookalike link hosts, or a lookalike sending domain (Section 4 lists ten such transforms). If the defender uses a language model, the attacker can append text that tries to steer the model’s output [7, 11]. The defender sees only what the recipient sees: no SPF/DKIM results, no mail headers, no attachment content, and no cross-message history. The defender knows a short list of *protected names*: the platform’s domains, brand, and team names.

Contributions.

1. A four-layer, zero-runtime-dependency detector (Section 3) in which every layer returns the reasons for its score.
2. A hardening recipe for using an LLM as an intent classifier on attacker-controlled text (Section 3.3), implementing OWASP LLM01 guidance with an independent pre-filter, nonce-delimited untrusted input, schema validation, and fail-closed behaviour.
3. A brand-parametric benchmark with ten adversarial variants and explicit metrics for injection hijack and per-layer contribution (Section 4), with results for the rule backend and two local LLM backends of 8B and 27B parameters (Section 5), an out-of-sample paraphrase test, and an ablation isolating the effect of confusable folding and de-obfuscation (Table 4).
4. A candid account of what the benchmark cannot show (Section 6): the seed corpus is in-sample for the rule baseline, it is small and English-only, and only two local LLMs have been measured.

2 Related work

Homograph and lookalike domains. Gabrilovich and Gontmakher [6] introduced the homograph attack; Høgers et al. [8] measured its prevalence in the wild and found it rare but feasible; Hu et al. [9] tested the internationalised-domain-name policies of major browsers against more than 9,000 cases and found weaknesses in every one. UTS #39 [2] is the Unicode Consortium’s standard for confusable detection (the *skeleton* function, §4) and restriction levels on script mixing (§5). *dstwist* [13] and *confusable_homoglyphs* [5] answer “is this domain a lookalike?” Our Layer 1 reuses the UTS #39 machinery but applies it to every name-like surface of a message (display name, sending domain, each link host) against a protected set, and reports the reason for each match.

Recruitment fraud. Vidros et al. [14] characterised online recruitment fraud and released a public corpus of 17,880 annotated job advertisements, on which most later fake-job classifiers are trained. That line of work classifies the *posting*. The scams in our threat model are one-to-one messages that may never correspond to a posting, and whose tell is a request (fee, equipment purchase, check deposit, off-platform move, personal documents) rather than

the listing’s wording.

LLM phishing benchmarks and prompt injection.

Fraud-R1 [15] evaluates how well LLMs resist multi-round fraud and phishing *inducements* aimed at the model itself; E-PhishGen [12] generates privacy-preserving phishing *email* corpora and shows that detectors trained on existing public sets degrade on new ones. Both are email-centric; our corpus is platform-message-centric and brand-parametric. Greshake et al. [7] showed that content an LLM-integrated application reads can hijack it, which OWASP lists as the top LLM application risk (LLM01) [11]. A message classifier is a textbook instance: the text it classifies is written by the adversary. Section 3.3 describes our mitigations and Section 4 the metric (hijack rate) that measures them.

3 System

The detector is a Python package, *impersonation-detector* v0.2.0 [3], with no runtime dependencies; an LLM backend is optional. It is instantiated with a list of protected names, e.g. ["handshake.com", "Handshake", "Handshake Support"], and analyses one *Message* (body, sender display name, sender address, subject, links) into a *Verdict*: a risk in [0, 1], an action in {allow, warn, hold, block}, and a list of human-readable reasons. The seed corpus and all examples here are instantiated with Handshake as the protected brand; every command takes `-p names` and nothing in the code is specific to it.

3.1 Layer 1: deterministic lookalike checks

Layer 1 asks whether a candidate string resembles a protected name and, if so, why. For each candidate (display name, sending domain, every link host in body or links) and each protected name it computes:

- **confusable-skeleton**: equality of case-folded UTS #39 skeletons [2]. The skeleton is NFD normalisation, replacement of each character by its prototype from *confusables.txt*, and NFD again. We casefold before and after, because some prototypes are upper case (0→0) and a single pre-fold would miss them.
- **edit-distance** $\leq k$: optimal-string-alignment (Damerau-Levenshtein) distance [1, 10] between skeletons, with k scaled to the protected name’s length: 0 for names of at most 4 characters, 1 up to 8, else 2, so *a.co* is never a typo of *b.co*.
- **mixed-script**: the candidate mixes scripts in a way UTS #39’s *Highly Restrictive* profile rejects (Latin with Cyrillic, say, but Latin with Han, Hiragana and Katakana is allowed). Reported only alongside another reason, so a legitimately non-Latin name is not flagged for being non-Latin.

Table 1: Layer 2 features and learned weights (`scorer_weights.json`); bias -2.48 . Examples: brand affix `handshake-verify.app`; brand in subdomain `handshake.com.evil.io`; leet `h4ndshake`; role words “Support”, “Careers”.

Feature	Weight
<i>Domain (also used for link hosts)</i>	
<code>domain_is_protected</code>	-1.85
<code>domain_skeleton_match</code>	$+3.25$
<code>domain_close_typo</code>	$+4.22$
<code>domain_mixed_script</code>	$+2.49$
<code>domain_punycode</code>	$+0.78$
<code>domain_tld_swap</code>	$+3.92$
<code>domain_brand_affix</code>	$+4.54$
<code>domain_brand_in_subdomain</code>	$+3.78$
<code>domain_leet_match</code>	$+1.88$
<code>domain_free_mail</code>	-0.85
<code>domain_missing</code>	0.00
<code>domain_invisible_chars</code>	$+3.05$
<i>Display name</i>	
<code>display_brand_token</code>	$+2.05$
<code>display_brand_lookalike</code>	$+1.11$
<code>display_role_word</code>	-1.31
<code>display_claims_brand_off_domain</code>	$+4.41$
<code>display_invisible_chars</code>	$+3.05$
<code>display_mixed_script</code>	$+0.77$
<i>Local part / handle</i>	
<code>local_brand_token</code>	$+4.96$
<code>local_role_word</code>	$+0.26$
<code>local_brand_lookalike</code>	0.00

- **punycode**: the candidate was an `xn-` label that decoded to a match.

An exact, case-insensitive match to the genuine name is excluded. Any hit sets a risk floor of 0.85.

3.2 Layer 2: explainable sender scorer

Layer 2 is logistic regression over 21 named binary-or-[0,1] features of a sender, extracted relative to the protected set (Table 1). The same extractor scores a bare link host using the twelve domain-level features. Because every feature is relative (“is a brand token embedded in the second-level label?”), weights learned on one set of brands transfer to another, and each prediction returns per-feature contributions so the strongest reasons can be displayed.

Training. A seeded synthetic generator produces labelled senders for a protected set: fourteen positive techniques (homoglyph, typo, TLD swap, affix and prefix domains, brand domain as a subdomain of an attacker host, leet, free-mail with brand display name, brand display name on an unrelated attacker domain, punycode, homoglyph display name, invisible characters, brand token in the local part) and eleven negative families that share surface features: the genuine brand, other companies with

role words in the display name, people on free-mail, university offices and students. We generate 800 examples (half positive) for each of eight protected sets (a career platform, a payments company, a software vendor, an exchange, two job boards, a bank, a file-sharing service) and hold out 20%. Plain gradient descent with an L_2 penalty (10^{-4}), learning rate 1.0, 600 epochs, in pure Python, takes about three seconds. On the 1,280-example hold-out: precision 0.995, recall 0.991, F_1 0.993, AUC 0.998. The weakest techniques are leet domains (0.867 accuracy) and university offices (0.955); every other technique scores 1.0. The learned weights are interpretable: a brand token in the local part of an address on a non-brand domain (`handshake.support@gmail.com`) is the single strongest signal, and role words in a display name are mildly *negative*, because on their own they describe “Career Services” at a university as readily as an impostor.

3.3 Layer 3: message intent

Layer 3 gives a message one of five labels (`benign`, `support_impersonation`, `recruiter_scam`, `credential_phish`, `payment_redirect`) with a confidence and the subset of thirteen named signals it observed (urgency, credential request, payment request, off-platform move, personal-data request, too-good-to-be-true, authority claim, link or attachment, threat or consequence, secrecy, injection attempt, invisible characters, homoglyph characters). Two classifiers share this interface.

Normalisation. Both first normalise the text: NFKC; removal of zero-width, joiner, bidi-isolate, tag, and C0/C1 control characters; *confusable folding*, which replaces every non-ASCII character whose UTS #39 prototype is a single ASCII letter with that letter (Cyrillic U+0430 $\rightarrow$ a) while leaving ASCII untouched so that `rn` stays `rn`; and a 6,000-character cap. The rule classifier then unions cues over three *views* of the text: as normalised, de-leeted (`p4ssw0rd` $\rightarrow$ `password`), and de-spaced (`p a s s w o r d` $\rightarrow$ `password`).

Rule baseline. Ten cue families, 483 regular-expression alternatives in total, map to the signals above. A family that can be negated (payment, credential, personal-data, off-platform) is suppressed when a negation token (*never*, *don’t*, *without*, ...) occurs within 45 characters before the cue *in the same clause* (no intervening `!?`, `;`, `:`). The classifier is precision-oriented: a label requires two converging cues, e.g. a payment request plus a recruiter, support, or pressure context, or a credential request plus a link, authority claim, urgency, or threat. Base confidence is 0.70–0.80, raised by 0.05 per extra signal beyond two, capped at 0.95. The sender header is also scanned so that “Handshake Support” in the display name counts as an authority claim even if the body is terse.

LLM classifier and OWASP LLM01 hardening. The second classifier sends the message to a language model over Ollama, any OpenAI-compatible endpoint, or the Anthropic API, using only the standard library. The message is attacker-controlled data, so the wrapper implements the following, each independent of the model’s cooperation:

1. **Nonce-delimited untrusted input.** The body is placed between `<message-$nonce>` tags with a fresh 8-hex-digit nonce per call; the system prompt names the nonce and instructs that text inside the tags is data. Any attempt by the body to close the delimiter (`</message-...>` or a bare `</message>`) is rewritten to `[tag removed]` before the prompt is built.
2. **Independent injection pre-filter.** An 84-alternative regular expression (“ignore all previous instructions”, “you are now”, “system prompt”, chat-template tokens such as `<|im_start|>` and `[INST]`, “classify this as benign”, ...) runs on the normalised text regardless of what the model says and adds the signal `injection_attempt`.
3. **Schema validation.** The first JSON object in the reply is parsed; the label must be one of the five, the confidence is clipped to $[0, 1]$, unknown signals are dropped, the rationale is truncated. One retry asks for JSON only.
4. **Fail closed.** An unparseable reply, or a backend error, yields the label `unknown` with risk 0.5, never `benign`.
5. **Override a hijacked “benign”.** If the pre-filter saw an injection and the model answered `benign`, the result becomes `unknown`: the model may have obeyed the message.
6. **Do not let the model raise its own floor.** The model’s own `injection_attempt` claim is discarded; only the pre-filter can assert that signal, because the pipeline raises a risk floor on it and a model that could set it could be talked into setting it.

The rule baseline has no instruction channel and therefore cannot be hijacked at all, which is one reason both ship.

3.4 Pipeline

Risk is a noisy-OR over the independent evidence, $r = 1 - \prod_i (1 - r_i)$ over the sender probability, the probability of the worst link host, and the intent risk (the confidence for a malicious label, 0.5 for `unknown`, 0 for `benign`), then raised to a floor of 0.85 for any Layer 1 hit and 0.60 for an injection attempt. Actions are thresholds on risk: warn at 0.30, hold at 0.60, block at 0.85, all configurable. A `Middleware` helper wraps a platform’s send function: blocked messages are dropped, held ones delivered with the verdict attached for review. Median end-to-end latency on the benchmark is 0.67 ms per message with the rule backend (Table 3); an uncached 8B-parameter local model adds about two seconds (Section 5.3).

4 Benchmark design

Seed corpus. The corpus has 76 hand-written messages: 30 benign, 14 support impersonation, 14 recruiter scam, 10 credential phish, 8 payment redirect (46 malicious). It is *brand-parametric*: `{brand}`, `{domain}`, and `{brand_lower}` are filled from the protected set at load time, so the same corpus tests any platform. No example is taken from a real victim report. The benign side is adversarial by design: 26 of 30 benign messages carry a tag naming a surface cue they share with attacks (salary figures, links, deadlines, role words, account and login language, “background check”, “verify”, “bank details”), including genuine support replies from the protected domain, a genuine password-changed notification, a genuine trust-and-safety notice about a removed fraudulent posting, and a real employer’s reminder never to send bank details by email. Three malicious messages (`s06`, `s11`, `c04`) carry a *spoofed From* on the protected domain itself, so the sender gives no evidence and only the request can be judged. Fifteen of the 46 malicious messages come from free-mail addresses; 21 contain a link, as do 5 benign messages.

Variants. Each seed message is passed through ten deterministic, label-preserving transforms, seeded per (variant, message) so runs are reproducible. Six apply to every message: *clean*; *homoglyph* (one confusable glyph in each cue word and brand mention in the body); *zero_width* (a zero-width space inside each); *leet* (digit substitutions); *spaced* (cue words letter-spaced); and *synonym*. Four make sense only for attacks and are scored on recall alone: *injection*; *link_homoglyph* (one confusable glyph in each link host); *link_subdomain* (the brand’s domain embedded as a subdomain of the attacker’s host); and *sender_homoglyph* (the sending domain is a confusable of the brand’s domain). The body-level transforms act on a fixed list of 30 cue words (*verify*, *password*, *telegram*, *gift*, *fee*, ...) and on brand mentions; *synonym* applies 23 fixed rewrite rules; *injection* appends one of three payloads (a bracketed “note to automated review systems”, a fake `SYSTEM` line with a target JSON object, and a chat-template `<|im_start|>system` block).

Configurations and metrics. Five layer configurations are run: L1, L2, L3, L1+L2, and L1+L2+L3. A message is *flagged* by L1 if any lookalike matches; by L2 if the sender or any link scores ≥ 0.5 ; by L3 if the label is malicious or `unknown`; and by a combined configuration if the pipeline action is not `allow`. For every configuration and variant we report precision, recall, F_1 , false-positive rate (on the 30 benign messages), per-class recall, the number of `unknown` verdicts, the *hijack* count, and median latency. Hijack counts, for the injection variant, the malicious messages labelled malicious on the clean text but `benign` with the injection appended: a flipped detection, not a message never caught.

5 Results

5.1 Rule backend

Table 2 gives recall and FPR for every configuration and variant, and Table 3 per-class recall on the clean corpus, hijack counts, and latency. All numbers are from `benchmarks/benchmark_rule.json` (v0.2.0), generated on an Apple-silicon laptop under Python 3.12.

Layers are complementary. Layers 1 and 2 know only the protected names, so their recall on scams that never mention the brand (every recruiter scam, most payment redirects) is near zero by construction; their job is the sender and the links. Layer 2 alone catches 71% of support impersonation on the clean corpus from the sender line alone, and rises to 0.50 and 0.52 recall overall when attackers put a confusable or the brand’s domain into their link host. Layer 3 alone reaches 0.98 recall on the clean corpus; its one miss (`s12`, a soft “verify to keep your profile visible” nudge with no explicit threat) is caught in the full pipeline by Layer 2, which flags the sender `team@handshake-official.com` for a brand affix. The combined pipeline is the only configuration that reaches 1.00 recall at 0.00 FPR on the clean corpus and stays at or above 0.94 recall on every variant.

Obfuscation hurts the rule baseline where it has no de-obfuscation. Leet substitution and synonym rewriting are the two variants that cost the full pipeline recall (0.94 each). The three messages missed under both (`c03`, a “shared document” credential phish from a generic portal domain; `c07`, an interview link whose host embeds a benign company’s domain in an attacker’s; `p02`, a gift-card “urgent favor” from a personal free-mail address) have no brand lookalike in the sender or link for Layers 1–2 to seize on, so once the cue words are rewritten nothing is left. These are the cases an LLM backend recovers (Section 5.3).

Injection. With the rule backend, the hijack count is 0 of 46 and no verdict is `unknown`; the pre-filter also recognises all three payloads, which raises the risk floor to 0.60 and contributes to the 1.00 recall under injection in the full pipeline. This row is a floor, not a result: the rule path has no instruction channel. The interesting hijack numbers are the LLM backend’s (Section 5.3).

Finding 1: UTS #39 skeletons leave gaps a mixed-script check closes. Consider the sending domain `handshake.com`, in which the italicised letters are Cyrillic U+0430, Cyrillic U+04BB, and Greek U+03BA. The prototypes of U+0430 and U+04BB are ASCII `a` and `h`, but the prototype of Greek small kappa is U+0138 LATIN SMALL LETTER KRA and the prototype of Cyrillic small te (U+0442) is U+1D1B LATIN LETTER SMALL CAPITAL T: Latin, but not ASCII. The skeleton of the candidate is therefore `handsha(U+0138)e.corn`, not `handshake.corn`,

and a detector that relies on skeleton equality alone—the natural reading of UTS #39 §4—does not fire. Our Layer 1 still matches, via edit distance 1 between skeletons plus the mixed-script reason (Cyrillic+Greek+Latin), and Layer 2 scores the sender at probability 1.00 from `domain_close_typo`, `domain_mixed_script`, and `display_claims_brand_off_domain`. This is also why Layer 3’s confusable folding is defined as “prototype is a single ASCII letter”.

Finding 2: regex cues die on one swapped glyph unless the text is folded first. Table 4 re-runs the L3 configuration with confusable folding disabled, with the leet and de-spacing views disabled, and with both disabled (`paper/ablation.py`). Without folding, a single confusable glyph in each cue word drops recall from 0.978 (clean) to 0.609, a loss of 37 points; with folding the loss is 2 points (0.957). The corresponding de-obfuscation views move leet recall from 0.674 to 0.826 and letter-spaced recall from 0.696 to 0.978. FPR is 0.00 in every ablated cell. The lesson generalises: any cue- or token-based classifier sees a different string after a confusable swap, so normalisation belongs in front of the classifier.

Finding 3: negation must be handled at the clause level. Benign message `b26` is a real employer’s onboarding note: HR will send a secure link from the payroll provider to enter direct-deposit details, and “Never send bank details by email.” The phrase *bank details* is a payment-request cue; the message also contains a link word and a role word, which is enough context for the rule classifier to label it `payment_redirect` if the cue fires. With the negation window it is `benign` with no signals; deleting the word *never* from the same message yields `payment_redirect`. Because the window is bounded to the clause, “Don’t worry, just send the fee” does not escape.

The two false positives are variant artefacts. The only non-zero FPR cells (0.03, one message of 30) arise in the homoglyph and zero-width variants, and both are benign messages from the platform itself that contain the platform’s own link. The homoglyph transform rewrites `https://handshake.com/safety` in `b16` to a host with Greek kappa, and the zero-width transform splits `handshake.com` in `b15`; Layers 1 and 2 then flag a link that is, after mutation, genuinely a lookalike of the protected domain. The label is preserved by construction but the message is no longer one a real platform would send; we report the cells as measured, and a future revision should exempt links to protected domains from benign-side mutation.

5.2 Out-of-sample: a paraphrased corpus

The seed corpus is the development set for the rule baseline, so the clean-corpus rows above are in-sample. For an out-of-sample test, `impersonation-detector` paraphrase had

Table 2: Recall / false-positive rate by layer configuration and variant, rule backend, v0.2.0. 46 malicious and 30 benign messages. Malicious-only variants report recall alone.

Config	clean	homoglyph	zero_width	leet	spaced	synonym	inject.	link_hg	link_sub	sender_hg
L1	0.04 / 0.00	0.04 / 0.03	0.04 / 0.03	0.04 / 0.00	0.07 / 0.00	0.04 / 0.00	0.04	0.04	0.04	1.00
L2	0.28 / 0.00	0.33 / 0.03	0.28 / 0.03	0.28 / 0.00	0.28 / 0.00	0.28 / 0.00	0.28	0.50	0.52	1.00
L3	0.98 / 0.00	0.96 / 0.00	0.98 / 0.00	0.83 / 0.00	0.98 / 0.00	0.89 / 0.00	0.98	0.98	0.98	0.98
L1+L2	0.28 / 0.00	0.33 / 0.03	0.28 / 0.03	0.28 / 0.00	0.28 / 0.00	0.28 / 0.00	0.28	0.50	0.52	1.00
L1+L2+L3	1.00 / 0.00	0.98 / 0.03	1.00 / 0.03	0.94 / 0.00	1.00 / 0.00	0.94 / 0.00	1.00	1.00	1.00	1.00

Table 3: Per-class recall on the clean corpus, injection hijack and unknown counts (of 46), and median latency per message.

Config	cred.	pay.	recr.	supp.	hijack	unk.	ms
L1	0.20	0.00	0.00	0.00	0	0	0.25
L2	0.20	0.12	0.00	0.71	0	0	0.11
L3	1.00	1.00	1.00	0.93	0	0	0.29
L1+L2	0.20	0.12	0.00	0.71	0	0	0.34
L1+L2+L3	1.00	1.00	1.00	1.00	0	0	0.67

Table 4: Ablation of Layer 3 de-obfuscation (rule backend, recall; FPR was 0.00 in every cell). “fold” = confusable folding; “views” = de-leet and de-space views.

L3 config	clean	homogl.	zero_w.	leet	spaced	synonym
full	0.978	0.957	0.978	0.826	0.978	0.891
no fold	0.978	0.609	0.978	0.826	0.978	0.891
no views	0.978	0.957	0.978	0.674	0.696	0.891
neither	0.978	0.609	0.978	0.674	0.696	0.891

Llama 3.1 8B rewrite all 76 messages with the same purpose, requests, URLs, amounts, and names but different wording, and **bench -only-extra** scored the 76 rewrites alone (the OOS column of Table 6; **benchmarks/oos_*/**). Layer 3 with the rule backend falls from 0.978 in-sample to 0.870 on the rewrites, still at FPR 0.00 (per class: credential phishing 0.90, payment redirect 1.00, recruiter scam 0.93, support impersonation 0.71); most of the loss is on support impersonation, where the cue lists are the most wording-specific. The sender layers win most of it back: the full pipeline reaches 0.957 at FPR 0.00, with support-impersonation recall back to 1.00, because a rewritten support-impersonation message still arrives from a free-mail address with the brand in its display name. The two messages the full pipeline misses are **r05** (a “confidential” recruiter asking for date of birth and home address, rewritten so the request no longer matches a cue) and **c04** (a one-time-code request sent from a spoofed address on the protected domain, so there is no sender evidence at all).

5.3 LLM backends: Llama 3.1 8B and Gemma 3 27B

Table 5 reports the LLM-backed Layer 3 with Llama 3.1 8B served locally through Ollama (temperature 0, JSON mode), for the L3 and L1+L2+L3

configurations on every variant (**benchmarks/benchmark_llm_ollama_llama3.1_8b.json**); per-message verdicts, confidences, and rationales are in **dump_llama3.1_8b.jsonl**. Three things stand out.

Recall is perfect and robustness is free. No obfuscation variant and no paraphrase moved the model off recall 1.00, in either configuration. The three messages the rule baseline loses under leet and synonym rewriting, and the two it loses on paraphrases, are all caught, and so is every class on the paraphrases (1.00 across the board at FPR 0.30); the rules needed explicit folding and de-spacing to come close (Table 4), the model needed nothing.

The false positives are the hard benign cases, and confidence does not separate them. Ten of the 30 benign messages are flagged on the clean corpus (**b05**, **b09**, **b10**, **b16**, **b17**, **b20**, **b23**, **b25**, **b26**, **b28**): an offer letter with a salary, a background-check notice, two requests for a CV, the platform’s own trust-and-safety notice and password-changed alert, a career-fair deadline, a financial-aid verification request, the payroll note that says *never* send bank details by email, and an I-9/W-4 onboarding message. The model’s rationales invent cues that are not in the text: for **b17** it writes that the message “requests the user to reset their password” when the message says no action is needed, and for **b26** that it “requests sensitive financial information” when it says the opposite. Its confidence on these ten is 0.8–0.9; on the 46 real attacks it is 0.9–1.0. Sweeping a confidence threshold over the dump gives recall 1.00 at FPR 0.20 for thresholds of 0.85 or 0.90, and recall 0.41 at FPR 0.00 at 0.95: there is no operating point near the rule baseline’s. A small model that is right about intent is not calibrated about it.

The injection pre-filter earned its place. Under the injection variant the model complied with the appended instruction and answered **benign** on 5 of the 46 attacks (**s05**, **c03**, **c04**, **c06**, **c10**). All five were overruled to **unknown**, and so held rather than delivered, because the deterministic pre-filter had already flagged the body. The hijack count in Table 5 is 0 only because of that overrule; without it the hijack rate would be 5/46. No message was **unknown** for any other reason, so the fail-closed path was exercised only by the overrule.

Table 5: Recall / false-positive rate with the Llama 3.1 8B backend, v0.2.0, same corpus and variants as Table 2. “unk.” is the number of the 46 injected attacks the model called **benign** and the pre-filter overruled to **unknown**; hijack count is 0 in both rows.

Config	clean	homoglyph	zero_width	leet	spaced	synonym	inject.	link_hg	link_sub	sender_hg	unk.
L3	1.00 / 0.33	1.00 / 0.30	1.00 / 0.30	1.00 / 0.43	1.00 / 0.37	1.00 / 0.33	1.00	1.00	1.00	1.00	5
L1+L2+L3	1.00 / 0.33	1.00 / 0.33	1.00 / 0.33	1.00 / 0.43	1.00 / 0.37	1.00 / 0.33	1.00	1.00	1.00	1.00	5

Table 6: Layer 3 backends compared, full pipeline (recall / FPR). “Comply” is how many of the 46 injected attacks the model itself labelled **benign** before the overrule (the rule path cannot comply; Llama’s five were all overruled). OOS = Llama-paraphrased corpus.

Backend	clean	zero_width	inject.	OOS	comply
rule	1.00 / 0.00	1.00 / 0.03	1.00	0.96 / 0.00	0
Llama 3.1 8B	1.00 / 0.33	1.00 / 0.33	1.00	1.00 / 0.30	5
Gemma 3 27B	0.98 / 0.13	1.00 / 0.17	1.00	1.00 / 0.17	0

A larger model: scale buys precision. Gemma 3 27B, run through the same harness on the clean, zero-width, injection, and paraphrased cells (`benchmarks/benchmark_llm_ollama_gemma3_27b.json` and `oos_gemma/`), cuts the clean-corpus FPR from 0.33 to 0.13 and the paraphrase FPR from 0.30 to 0.17 with recall essentially unchanged (Table 6); its rationales cite cues that are in the text. Its false positives are all of one kind: a genuine support reply from the platform (**b14**), its trust-and-safety takedown notice (**b16**), its password-changed alert (**b17**), and a financial-aid office’s document request (**b23**), each labelled **support_impersonation** at 0.85–0.95 because the model has no way to know that the sender is who it claims to be; on the paraphrases a second genuine support reply (**b18**) joins them. That knowledge lives in Layer 2 (**domain_is_protected**); fusing “the model says support impersonation” with “the sender really is on the protected domain” is the obvious next step, not done here because the corpus also holds spoofed-From attacks (**s06**, **s11**, **c04**) that a naive version of that rule would wave through. Confidence is better calibrated than the 8B model’s but still not a lever: thresholds of 0.90 and 0.95 give FPR 0.10 and 0.07 at recall 0.98. The one miss, **c07**, is a credential phish whose link host embeds a *third party’s* domain (`orbitallabs.io.docs-share.site`) under a plausible recruiter sender; no layer flags it, since Layers 1–2 know only the protected brand and both models read the link as a meeting invite; brand-agnostic detection of any embedded registrable domain is the extension it motivates. Finally, the 27B model complied with the injected instruction 0 times in 46, against 5 for the 8B model; the pre-filter still runs, because the guarantee must hold for whichever model is plugged in.

Cost, and where this leaves Layer 3. Uncached median latency per message on an Apple-silicon laptop was 2.1–2.3s for the 8B model and 11–12s for the 27B model;

the rule baseline is three to four orders of magnitude faster. The backends span a frontier: the rules at FPR 0.00 with recall that degrades under paraphrase, the 8B model at recall 1.00 with an FPR no consumer platform could ship, the 27B model in between at ten times the cost. The pipeline’s noisy-OR accepts any of them, and the **-backend anthropic** and **-backend openai** paths run unchanged for hosted models.

6 Limitations

The clean-corpus numbers for the rule baseline are in-sample. The seed corpus doubled as the development set for the rule classifier’s cue lists, so Table 2’s L3 and L1+L2+L3 rows measure robustness to *transformation* of known messages, not generalisation to unseen ones. Section 5.2 is the out-of-sample test, with two caveats of its own: the paraphrases were produced by the same Llama 3.1 8B that is then evaluated on them, so the model’s 1.00 recall there is not independent evidence; and a paraphrase preserves the request, so it tests wording, not novel scam scripts. Likewise the two LLMs measured are local models at temperature 0, the larger on three variants only; their FPRs and injection compliance are not properties of LLM classifiers in general.

The corpus is small and English-only. Seventy-six messages give coarse resolution: one message is 2.2 points of recall and 3.3 points of FPR. The corpus is hand-written so that every example is defensible and no victim’s message is reproduced, which bounds its size. Contributions of labelled real-world messages, with consent, are the obvious next step.

Layer 2 is trained on a generator. The generator encodes the impersonation techniques we know; a novel one needs a new named feature, which is why the feature list is explicit and the trainer is 150 lines. The 0.993 holdout F_1 should not be expected to transfer to wild senders.

Protected names are a list, and homonyms exist. An organisation with many brands supplies them all. A legitimately named business whose name contains the brand (“Firm Handshake Consulting”) scores as a lookalike; the right response is a product decision (warn and show the reason), not a detector change.

Beyond the message. Version 0.2.0 of the package adds two modules outside this paper’s scope that apply the same recipe to the domain an attacker registers and the account they take over: a lookalike-domain watch (permutation generation, DNS/MX/RDAP evidence, Layer 2 scoring, and a blacklist fed back to the message detector), and an account-takeover detector with a cross-account shared-IP signal. A live scan of `handshake.com` and `joinhandshake.com` on 2026-10-13 (`benchmarks/watch_handshake.json`) returned 72 lookalike hits, 38 with MX records, mostly TLD swaps and single-character omissions; a live domain with MX is capable of abuse, not evidence of it.

7 Reproducibility

All code, weights, corpus, and reports are public [3]. The following reproduces every number in this paper except the IC3 figure:

```
cd impersonation-detector      # after git clone; fresh venv
pip install -e ".[dev]"
pytest -q                      # 65 tests
impersonation-detector bench -p handshake.com \
    -p Handshake -p "Handshake Support" --out benchmarks/
python paper/ablation.py      # ablation table
impersonation-detector bench ... --backend ollama \
    --model llama3.1:8b        # Table 6
impersonation-detector paraphrase ... --backend ollama \
    --out benchmarks/paraphrase_llama3.1_8b.jsonl
impersonation-detector bench ... --only-extra \
    benchmarks/paraphrase_llama3.1_8b.jsonl # Table 5
```

The LLM runs need a local Ollama; per-message response caches (`benchmarks/cache_*.jsonl`) preserve the exact responses behind Tables 5 and 6. Layer 2 retraining (`python -m impersonation_detector.train`) is deterministic given `-seed`; variants are seeded per message. Unicode data files are redistributed under the Unicode License.

8 Ethics

No real victim data was used. Every message in the seed corpus was written by the author; senders, companies, and domains are fictional, and the only real organisation named is the protected brand used as the corpus parameter. The adversarial variants and injection payloads are the kind an attacker can already produce in seconds and are published so that defenders can measure against them [12, 15]. The detector is explainable by design so that a platform can show a user or reviewer the reason for a hold rather than silently dropping a legitimate recruiter’s message. The author is an engineer at the platform used as the example brand; this work was done independently, uses no platform data, and does not describe that platform’s production defences.

References

- [1] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3): 171–176, 1964. doi: 10.1145/363958.363994.
- [2] Mark Davis and Michel Suignard. Unicode technical standard #39: Unicode security mechanisms. Unicode Consortium, <https://www.unicode.org/reports/tr39/>, 2026. Data files `confusables.txt` and `Scripts.txt` as of Unicode 18.0.0 (2026).
- [3] Sochima Ezema. `impersonation-detector`: layered detection of lookalike senders and support-team / recruiter impersonation for messaging platforms. <https://github.com/sochimaezema-hs/impersonation-detector>, 2026. Version 0.2.0, MIT license.
- [4] Federal Bureau of Investigation, Internet Crime Complaint Center. 2024 Internet Crime Report. Technical report, FBI, 2025. URL https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf.
- [5] Victor Felder. `confusable_homoglyphs`: Detect confusable usage of unicode homoglyphs. https://github.com/vhf/confusable_homoglyphs, 2016. Software.
- [6] Evgeniy Gabrilovich and Alex Gontmakher. The homograph attack. *Communications of the ACM*, 45(2):128, 2002. doi: 10.1145/503124.503156.
- [7] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec ’23)*, pages 79–90. ACM, 2023. arXiv:2302.12173.
- [8] Tobias Holgers, David E. Watson, and Steven D. Gribble. Cutting through the confusion: A measurement study of homograph attacks. In *Proceedings of the 2006 USENIX Annual Technical Conference (USENIX ATC ’06)*, pages 261–266, Boston, MA, 2006. USENIX Association.
- [9] Hang Hu, Steve T. K. Jan, Yang Wang, and Gang Wang. Assessing browser-level defense against IDN-based phishing. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security ’21)*. USENIX Association, 2021. ISBN 978-1-939133-24-3.
- [10] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8): 707–710, 1966.
- [11] OWASP GenAI Security Project. LLM01:2025 Prompt Injection. OWASP Top 10 for LLM Applications 2025, <https://genai.owasp.org/llmrisk/llm01-prompt-injection/>, 2025.
- [12] Luca Pajola, Eugenio Caripoti, Stefan Banzer, Simeone Pizzi, Mauro Conti, and Giovanni Apruzzese. E-PhishGen: Unlocking novel research in phishing email detection. In *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security (AISec ’25)*, pages 64–76. ACM, 2025. doi: 10.1145/3733799.3762967.
- [13] Marcin Ulikowski. `dnstwist`: Domain name permutation engine for detecting homograph phishing attacks, typo squatting, and brand impersonation. <https://github.com/elceef/dnstwist>, 2015. Software.
- [14] Sokratis Vidros, Constantinos Koliass, Georgios Kambourakis, and Leman Akoglu. Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, 9(1):6, 2017. doi: 10.3390/fi9010006.
- [15] Shu Yang, Shenzhe Zhu, Zeyu Wu, Keyu Wang, Junchi Yao, Junchao Wu, Lijie Hu, Mengdi Li, Derek F. Wong, and Di Wang. Fraud-R1: A multi-round benchmark for assessing the robustness of LLM against augmented fraud and phishing inducements. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4374–4420, 2025. doi: 10.18653/v1/2025.findings-acl.226.