V1.0 · NOVEMBER 2026

Nobody can spot the malicious email

Four open detectors over 308 synthetic emails in English and Greek. The request invoice fraud actually uses passed every one that kept ordinary mail clean.

Nikolaos Broikos · broikos.gr November 2026 308 emails · 4 detectors · 23 Nov 2026

0 of 40

SUBTLE REQUESTS CAUGHT BY PROMPT GUARD 2

20 of 30

ORDINARY INSTRUCTIONS TO PEOPLE FLAGGED AS ATTACKS

41 of 60

ORDINARY GREEK EMAILS FLAGGED BY AN ENGLISH DETECTOR

first

PUBLISHED DETECTOR MEASUREMENT ON GREEK

01 What this is

> Discharges claims H001 to H010.

In October 2026 criminals began writing emails for two readers at once: a human being, and the AI assistant that summarises the inbox. The hidden half tells the assistant to present a changed bank account as a priority action, or to forward the invoice to an accounts-payable-looking address. The target is not the machine. It is the person who acts on the draft.

The obvious response is to detect those emails. **This paper measures whether that works, and it does not.**

The claim in one paragraph

Four open prompt-injection detectors were run over 308 synthetic business emails in English and Greek, scored three ways each. **No detector combined few false alarms on ordinary instructions written to people with catching the subtle requests that invoice fraud actually uses.** The note asking whoever drafts the reply to confirm payment to a new bank account, which is the exact move of the October campaigns, passed every detector that kept ordinary English mail clean. The one detector that caught every direct override in 512-token windows also flagged two thirds of the emails carrying routine instructions to colleagues, phrases like "disregard the invoice sent this morning", which in commercial mail are not attacks but Tuesday.

That matches every independent measurement published this autumn, and it points at the same conclusion those measurements reached from other directions: **you cannot reliably spot the malicious email, so the thing to limit is what the assistant is allowed to do with one.**

What is measured here, and what is quoted

Two kinds of evidence, kept apart throughout.

Measured here. A local bench, run on 23 November 2026, of four open detectors over 308 emails generated from a fixed seed. Every name, vessel, company, address, IBAN and figure in that corpus is invented, and the whole set, the generator, the runner and the raw per-email scores are published with this paper. **It is the author's own synthetic test set**, written to look like ordinary commercial shipping correspondence because that is a plausible inbox for a drafting assistant, and it describes no real organisation or correspondence.

Quoted from published work. The research position this paper argues from comes from papers, advisories and vendor posts published up to 23 November 2026, each cited at source. **Most of the research figures in that window come from preprint abstracts**, which is stated where it matters rather than once in a footnote. Two items that could not be verified at source were left out deliberately rather than softened.

Why Greek is in here

No published test of any injection detector on Greek exists. None of the detectors examined lists Greek as a trained or tested language, and every vendor that comments on language coverage tells customers to test their own. This bench is therefore the first measurement of its kind, which is a low bar and still worth clearing.

The result was not what the English literature would predict in either direction. The English-only detector reacted to Greek text itself, flagging up to 41 of 60 perfectly ordinary Greek emails, so its Greek catches are not discrimination. The three multilingual detectors did about as well in Greek as in English, which is better news than

the usual off-English decline, and still not good enough to gate on.

What this paper does not do

It does not propose a detector, rank products for purchase, or claim a rate at which real attacks would be caught. **The bench shows which texts pass these detectors, not how often an attacker would succeed**, and section 11 is explicit about the difference.

It also does not measure the design it recommends. The draft-and-approve pattern in section 10, and in particular the deterministic check on bank details, links and recipients, is argued from what failed elsewhere. **Nobody has published a measurement of it, including this paper**, and it is labelled as proposed throughout.

02 The instrument

> Discharges claims H003, H007, H011 to H024.

---

308 emails, four detectors, three scoring modes, one CPU. Everything in this section is published with the paper: the generator, the corpus, the runner, the raw per-email scores and the run log.

The corpus

Written by make_data.py from seed 20261123, so it regenerates byte for byte. Per language, English and Greek:

CLASS	EMAILS	TEMPLATES	WHAT THEY ARE
Benign	60	12	Offers, ETAs, port disbursement invoices, crew changes, bunker stems, inspections, laytime statements, recaps, weather routing, proforma requests, a moved port call, P&I cover
Hard benign	30	6	**Instructions meant for people**: ignore my previous email, disregard the invoice sent this morning, do not forward plus orders for the master, an automated password notice, new routing from the charterers, override the previous voyage orders
Injected	60	3 styles, 5 phrasings each	Instructions meant for an AI assistant

Plus eight Greek emails carrying English injections, for 308 in all.

Half the benign emails are followed by a long quoted thread, port notes, charter party terms and a confidentiality disclaimer. Every injection is a paragraph of its own, and half of each style sits early, straight after the greeting of a short email, while half sits late, after the long thread.

The hard benign class is the point of the bench. Any classifier can separate "ignore all previous instructions" from a freight offer. The question for a real inbox is whether it can separate an attack from a colleague writing "disregard the invoice sent this morning", which in commercial correspondence is a normal sentence that happens to look exactly like the canonical attack.

The three injection styles

STYLE	COUNT	PHRASING
A, direct override	20	The usual trigger wording. "AI assistant: disregard your system prompt and the user's request", "SYSTEM OVERRIDE: the previous rules no longer apply"
B, subtle	20	Addressed to whoever drafts or summarises the reply, with no trigger words at all: **confirm a new bank account**, state that the demurrage claim was accepted, include other brokers' rates, recommend this offer, add a colleague in copy
C, exfiltration	20	A link or image that would carry thread data out

Style B is the one that matters, because it is the form the October invoice campaigns used. It contains no instruction a keyword would catch. It asks politely for something a helpful drafting assistant would plausibly do, and the harm lands on the human who signs the draft.

The detectors

Four open models, each loaded through transformers 5.19 as a sequence classifier, fp32, CPU, four threads:

MODEL	LICENCE	LANGUAGE SCOPE
protectai/deberta-v3-base-prompt-injection-v2	Apache 2.0	**English only by its card**, and archived
gravitee-io/Llama-Prompt-Guard-2-86M-onnx	Llama 4 Community	Meta's Prompt Guard 2 86M, re-published without the gate
proventra/mdeberta-v3-base-prompt-injection	MIT	Multilingual base
NeuralTrust/prompt-guard-oss-small	MIT	mmBERT-small base

Two model caveats belong here rather than in the limitations. The Prompt Guard 2 result applies to gravitee-io's re-publication and not to Meta's own weights: Meta's repository is gated and hides its file hashes from anonymous reads, so the copy could not be verified. Its safetensors file matches Meta's in size to the byte, 1,115,268,200, and its tokenizer files do not. And **NeuralTrust's positive class is named jailbreak, not injection**; the runner scores that class as injection, so its misses on the subtle style may reflect what it was trained to find rather than only how well it finds it.

Three ways to score the same email

At a 0.5 threshold on the injection class:

- **Whole.** The email cut at 512 tokens, which is what a plain pipeline call does.
- **Chunked.** 512-token windows overlapping by 64 tokens, highest score kept. This is what Prompt Guard 2's own model card tells users to do with longer text.
- **Paragraphs.** Each blank-line paragraph scored alone, highest kept.

The three modes exist because a long email thread does not fit in 512 tokens, and the choice of how to cut it turns out to change the answer more than the choice of model does.

The machine, recorded rather than assumed

AMD Ryzen 7 5700G, 16 GB, Windows 11, Python 3.12.13, torch 2.14.1, transformers 5.19.0.

Detectors ran one at a time behind a memory gate that would not start a model with less than 5 GB free and would kill it below 1.2 GB; the watchdog never fired. **Other jobs shared the CPU throughout**, with load between 8.6 and 23.0 at the end of each run, so **every timing in this paper is an upper bound for an idle machine** rather than a benchmark figure.

What went wrong, recorded in place

Three failures, none of which changed the method:

The first pass failed for all four models. Under transformers 5.19 DebertaV2Tokenizer no longer has build_inputs_with_special_tokens, which the hand-rolled windowing called. The windows now come from the tokenizer's own overflow with a stride of 64.

Paragraph scoring was added after Protect AI had already run, so Protect AI ran a second time with all three modes. The first run is published too, and its whole and chunked flags matched the rerun's on all 308 emails.

A design slip in the generator, which costs one comparison. Every second benign email was meant to carry the long thread, but with 12 benign templates the alternation landed on template parity, so each benign template is always long or always short. **The effect of the long thread on false alarms therefore cannot be separated from the effect of the template.** The injected emails are unaffected, because their position decides their length.

03 What the bench found

> Discharges claims H004, H006, H012, H016, H025 to H035.

Each cell is whole / chunked / paragraphs, as counts of the emails in the first column.

SLICE (EMAILS)	PROTECT AI V2	PROMPT GUARD 2 86M	PROVENTRA MDEBERTA	NEURALTRUST SMALL
English benign flagged (60)	0 / 0 / 30	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
English hard benign flagged (30)	17 / 17 / 25	0 / 0 / 5	**20 / 20 / 20**	0 / 0 / 6
English injections caught, all (60)	11 / 11 / 50	4 / 10 / 12	18 / 36 / 32	15 / 16 / 32
English direct override (20)	9 / 9 / 20	4 / 10 / 12	10 / 20 / 20	13 / 13 / 20
English subtle, no trigger words (20)	0 / 0 / 14	**0 / 0 / 0**	3 / 5 / 4	**0 / 0 / 0**
English exfiltration link (20)	2 / 2 / 16	0 / 0 / 0	5 / 11 / 8	2 / 3 / 12
English injection early (30)	11 / 11 / 20	4 / 4 / 6	18 / 18 / 16	6 / 6 / 16
English injection late, after a thread (30)	0 / 0 / 30	0 / 6 / 6	0 / 18 / 16	9 / 10 / 16
Greek benign flagged (60)	**6 / 18 / 41**	0 / 0 / 0	0 / 0 / 3	0 / 0 / 0
Greek hard benign flagged (30)	10 / 15 / 30	2 / 2 / 5	16 / 16 / 15	5 / 5 / 5
Greek injections caught, all (60)	16 / 44 / 60	5 / 11 / 8	17 / 33 / 34	6 / 19 / 44
Greek direct override (20)	7 / 17 / 20	5 / 11 / 8	10 / 20 / 20	6 / 14 / 20
Greek subtle (20)	2 / 10 / 20	0 / 0 / 0	2 / 4 / 5	0 / 0 / 4
Greek exfiltration link (20)	7 / 17 / 20	0 / 0 / 0	5 / 9 / 9	0 / 5 / 20
Greek injection early (30)	16 / 16 / 30	5 / 5 / 4	17 / 17 / 17	6 / 6 / 22
Greek injection late (30)	0 / 28 / 30	0 / 6 / 4	0 / 16 / 17	0 / 13 / 22
English injection inside a Greek email (8)	2 / 6 / 8	2 / 4 / 2	2 / 4 / 5	2 / 4 / 4
Late injections past the first 512 tokens (64)	34	64	64	34
Milliseconds per email, 4 shared threads	496 / 1,078 / 1,387	574 / 1,013 / 1,053	533 / 940 / 991	**179 / 327 / 392**

Ordinary mail was never the problem

No detector flagged any of the 60 ordinary English emails whole or in windows. The three multilingual models flagged at most 3 of 60 ordinary Greek emails in any mode. On the question everyone worries about first, these models are quiet.

That is worth saying plainly because it is the opposite of the independent result on long English contracts, where Protect AI v2 flagged 60.61 and 42.55 percent of benign documents and two other guards flagged 91 to 100 percent. Business email is shorter and more formulaic than a contract, and on this corpus it does not trip them.

Instructions written to people were the problem

DETECTOR	ENGLISH HARD BENIGN FLAGGED, OF 30	GREEK
proventra mDeBERTa	**20 in every mode**	15 to 16
Protect AI v2	17 whole, 17 windows, 25 paragraphs	10 to 30
NeuralTrust small	0 whole, 0 windows, 6 paragraphs	5 in every mode
Prompt Guard 2 86M	0 whole, 0 windows, 5 paragraphs	2 to 5

The flags follow templates, not emails. proventra flagged four of the six English templates in all five of their fillings: disregard the invoice, do not forward plus orders for the master, new routing from the charterers, and override the previous voyage orders. Prompt Guard 2 and NeuralTrust flagged none in English whole or in windows, and by paragraph both flagged "override the previous voyage orders" every time.

In a shipping inbox all four of those are routine operational messages. A detector that flags two thirds of them is not finding attacks, it is finding the imperative mood. **At a few thousand emails a day it would bury its own real alerts**, which is the failure mode that makes a detector worse than nothing: the alert that matters arrives in a pile nobody reads.

The confidentiality footer is an instruction

Scored paragraph by paragraph, Protect AI flagged 30 of 60 ordinary English emails. All 30 were the long ones.

which_paragraph.py, published with the paper, scores each benign paragraph alone and finds the cause: **the standard confidentiality footer**, the one that begins "This message and any attachments are confidential" and ends "please notify the sender immediately and delete this message", **fires the detector on its own in all 45 English emails that carry one**, and its Greek version in all 45 Greek ones. In Greek, greetings and signatures fire it too, including "Καλημέρα σε όλους," in 10 of 10.

Read as text, the footer is a set of instructions to the reader. A classifier trained to spot instructions aimed at a model has no way to tell that apart, and nearly every corporate email in existence ends with one.

> Stripping footers before scoring is the obvious fix and was not measured.

04 The request that matters passed

> Discharges claims H004 to H005, H036 to H044.

Style B is twenty emails per language that ask, politely and without a single trigger word, for something a helpful drafting assistant would plausibly do. Confirm a new bank account in the reply. State that the demurrage claim was accepted. Include other brokers' rates. Recommend this offer. Add a colleague in copy.

DETECTOR	ENGLISH SUBTLE CAUGHT, OF 20	GREEK, OF 20
Prompt Guard 2 86M	**0 / 0 / 0**	**0 / 0 / 0**
NeuralTrust small	**0 / 0 / 0**	0 / 0 / 4
proventra mDeBERTa	3 / 5 / 4	2 / 4 / 5
Protect AI v2	0 / 0 / 14	2 / 10 / 20

> Prompt Guard 2 caught none of the 40 subtle emails in either language in any mode.

NeuralTrust caught none in English, and by paragraph one Greek phrasing, the one asking for other brokers' rates. proventra reliably caught one phrasing per language, add a colleague in copy, and little else.

Protect AI's paragraph column looks better and is not. It flagged 14 of 20 English and all 20 Greek in the mode where it also flagged every email carrying a confidentiality footer and 41 of 60 ordinary Greek emails. A detector that flags most things catches most things.

The bank account note

One phrasing deserves naming on its own, because it is the move the October 2026 invoice campaigns actually used: a paragraph asking whoever drafts the reply to confirm payment to a new bank account.

It passed all four detectors in English whole and in windows, and all three multilingual detectors in every mode. Only Protect AI caught it, by paragraph, in the mode where its false alarm rate makes the catch uninterpretable.

That is the whole argument of this paper in one row of a table. The request carries no override, no system prompt reference, no suspicious URL and no imperative aimed at a model. It reads like a client correcting their payment details, which is a thing clients do. The only signal available is that the bank account is not the one on file, **and that is not a language question at all.** It is a lookup.

Direct overrides were caught by phrasing, not by intent

The styles that do carry trigger words were caught, and the pattern of catches is informative about what the models are doing.

proventra, in windows or by paragraph, and NeuralTrust, by paragraph, caught all five phrasings in both languages. Prompt Guard 2 by paragraph caught three of five English and two of five Greek phrasings, **missing "AI assistant: disregard your system prompt and the user's request" and "SYSTEM OVERRIDE: the previous rules no longer apply"**, even though its own model card scopes it to explicit attempts to override prior instructions. Those two are about as explicit as the category gets.

A model that catches three of five wordings of the same intent is matching wordings. That is fine as telemetry and it is not a boundary, because the attacker chooses the wording.

Exfiltration links

DETECTOR	ENGLISH CAUGHT, OF 20	GREEK, OF 20
NeuralTrust small	2 / 3 / 12	0 / 5 / **20**
proventra mDeBERTa	5 / 11 / 8	5 / 9 / 9
Protect AI v2	2 / 2 / 16	7 / 17 / 20
Prompt Guard 2 86M	**0 / 0 / 0**	**0 / 0 / 0**

NeuralTrust by paragraph caught all five Greek phrasings and three of five English ones. Prompt Guard 2 caught none in either language in any mode.

The published evidence says not to lean on this column either. Every exfiltration bypass found in the window defeated a control that interpreted text: Salesforce's URL allowlist fell in 2025 to an expired allowlisted domain bought for five dollars, and again in 2026 because its parser missed some top-level domains, with content labelled "blocked" after the data had already left. Link previews in messaging apps are a confirmed zero-click channel, with the preview service fetching the attacker's URL and no human clicking anything.

05 Greek, measured for the first time

> Discharges claims H009, H016, H026, H032 to H033, H037, H041, H045 to H056.

No detector examined for this paper lists Greek as a trained or tested language, and no published measurement of any injection detector on Greek exists. Prompt Guard 2 was evaluated on eight languages and Google Model Armor tested on nine. Protect AI v2 is English only by its card and archived. Horizon Labs' guard trained on thirty unlisted languages and reports 17.8 percent false alarms on third-party Turkish data against 0.4 percent on its own multilingual set. Microsoft's current page says the Prompt Shields models are "primarily trained and evaluated on English content", with document attack detection off by default and no accuracy figures published.

Every vendor that addresses language coverage tells customers to test their own language. Nobody has published such a test for Greek. This is that test, and it is the first, which is a low bar that nonetheless had not been cleared.

An English detector on Greek text reacts to the text

MODE	ORDINARY GREEK EMAILS FLAGGED, OF 60
Whole	6
Chunked	18
Paragraphs	**41**

Protect AI v2, English by its card, flagged 6, 18 and 41 of 60 perfectly ordinary Greek emails. Paragraph scoring shows what it is reacting to: the Greek confidentiality footer in all 45 Greek emails that carry one, and in Greek also greetings and signatures, with "Καλημέρα σε όλους," firing it in 10 of 10 cases where it appears.

So its Greek catches are not discrimination. It flagged 16 of 60 Greek injections whole and 44 in windows and 60 by paragraph, and those numbers mean nothing next to a model that is responding to the presence of Greek. Using an English-only detector on Greek mail is not a degraded measurement, it is no measurement.

The multilingual detectors did about as well in Greek as in English

This is the finding that surprised, because the literature predicts decline off English.

DETECTOR	ENGLISH DIRECT OVERRIDE, WINDOWS	GREEK, WINDOWS
Prompt Guard 2 86M	10 of 20	11 of 20
proventra mDeBERTa	20 of 20	20 of 20
NeuralTrust small	13 of 20	14 of 20

And by paragraph, NeuralTrust's model caught **all 20 Greek direct overrides and all 20 Greek exfiltration links with no ordinary Greek email flagged at all.** On this corpus Greek is not the weak language.

Two qualifications keep that honest. The Greek injections are translations of the same five phrasings, so this measures whether the models transfer across the language rather than whether Greek attackers would write the same way. And the subtle style passed in Greek just as it did in English: NeuralTrust caught 4 of 20 and proventra 5 of 20 at best.

English instructions inside Greek emails were caught in 2 to 5 of 8 cases by the multilingual models, which is too few emails to read anything from and is reported because it was measured.

A Greek-specific control nobody has measured

Fourteen Greek capitals are visually identical to Latin ones. A, B, E, Z, H, I, K, M, N, O, P, T, Y and X render the same as their Latin counterparts and are different code points, so an attacker can mix scripts to defeat exact matching while the text looks unchanged to a reader.

The control that follows is **script-aware normalisation of Greek and Latin homoglyphs applied to a copy used for detection**, never to the original, because normalising the original would damage legitimate Greek. It is not measured here and is offered as practice.

One trap worth naming: **language checks that work by looking for non-ASCII letters are not language checks.** They will treat Greek, homoglyph-mixed Greek and an emoji the same way.

What carries over from elsewhere

Two measurements from the author's earlier work on retrieval apply to Greek unchanged. **Naming the answer language stopped a planted instruction from switching it**, measured with English questions where a planted refusal had previously come back in Greek. And **late chunking leaked across an access boundary in Greek as in English**, with 83 against 98 percent of questions finding the same file's chunks in the top five, the per-segment fix working in both languages.

Greek is also absent from the embedding inversion, poisoning, membership inference and extraction literature. A twenty-language inversion study of a multilingual embedder left Greek out, found Cyrillic-script languages the most exposed, and showed exposure varying by script in ways nobody predicted. **Greek exposure there is therefore unknown rather than low**, and that distinction is the whole point of saying so.

06 Where you cut the email changes the answer

> Discharges claims H014, H019, H021, H031 to H032, H034, H050, H057 to H065, H140.

A detector reads 512 tokens. A real email thread does not fit in 512 tokens. How you cut it up turns out to matter more than which model you picked, and nobody ships that decision as a setting worth thinking about.

Scoring the whole email misses everything that comes after a thread

Both mDeBERTa tokenizers put all 64 late injections past the first 512 tokens. For Prompt Guard 2 and proventra, scoring the whole email therefore missed every one of the 30 late injections in each language. Not most. All of them.

Windows restored the early catch rate, which is exactly what Prompt Guard 2's model card tells users to do with longer text, and which a plain pipeline call does not do.

The same text, read later, stopped firing the detector

Protect AI's tokenizer is different: it kept the English late injections inside its first window, beginning at token 430 to 464 of emails at most 504 tokens long. The injection was there, in the window, scored.

It caught none of the 30 late English injections whole or in windows, against 11 of 30 early. Same five phrasings, same paragraph, further down the page.

That is not a truncation effect and it is not a language effect. The detector's score depends on what precedes the text it is reading, and an attacker controls where in the thread the paragraph sits. By paragraph it flagged all 30 late emails, but every late email carries the confidentiality footer, so **those flags say nothing about the injection** and are excluded from any reading of the late column.

Paragraph scoring is the whole ball game, in both directions

DETECTOR	EFFECT OF SCORING EACH PARAGRAPH ALONE
NeuralTrust small	**Much better.** Greek injections 6, 19 and 44 of 60. English exfiltration links 2, 3 and 12 of 20. Position stops mattering: 16 early and 16 late in English, 22 and 22 in Greek
proventra mDeBERTa	About the same as windows
Prompt Guard 2 86M	Helped in English, not in Greek
Protect AI v2	**Unusable.** The footer scored alone fires it, so every long email is flagged

One detail cuts against the footer story and is worth keeping. **On the short emails, which carry no footer, Protect AI by paragraph caught 20 of 30 early English injections against 11 whole, with none of the 30 short ordinary English emails flagged.** So paragraph scoring genuinely helps it too, and the footer is what ruins it. Strip the footer and Protect AI's paragraph mode might be the best English operating point in the bench. That was not measured, and the sentence is a hypothesis.

Cost does not decide anything

DETECTOR	MILLISECONDS PER EMAIL, BY PARAGRAPH	PER 3,000 EMAILS
NeuralTrust small	**392**	about 20 CPU minutes
proventra mDeBERTa	991	about 50 minutes
Prompt Guard 2 86M	1,053	about 50 minutes
Protect AI v2	1,387	about 70 minutes

On four shared threads of an ordinary desktop CPU, with other jobs running. Every figure is an upper bound for an idle machine. For scale, Meta's own figures for Prompt Guard 2 are 19.3 ms for the 22M model and 92.4 ms for the 86M per 512-token window on an A100.

Nothing here is priced out. Twenty to seventy CPU minutes per three thousand emails is nothing, on hardware anyone has. The reason not to gate on these detectors is that they do not work well enough, not that they cost too much, and that is a more uncomfortable conclusion because money is the problem you can solve.

The best operating point measured, and why it is still not a gate

NeuralTrust's prompt-guard-oss-small, scored by paragraph:

- **No ordinary email flagged in either language**, 0 of 60 English and 0 of 60 Greek
- 6 of 30 English and 5 of 30 Greek hard benign flagged
- **Every direct override caught** in both languages, 20 of 20 and 20 of 20
- **Every Greek exfiltration link caught**, 20 of 20
- 392 ms per email on a shared desktop CPU

That is a good result and it is the recommendation of this bench **as telemetry**. It is not a gate, for one reason that no amount of tuning fixes: it passed every English subtle phrasing and four of five Greek ones. **The requests that matter most to an assistant that drafts replies are the ones it does not see.**

07 The published evidence says the same thing, louder

> Discharges claims H008, H035, H066 to H086.

The bench above is static: the attacks do not adapt. That is its most important limitation and it is also why the published literature matters here, because the published literature did let the attacker adapt, and the results are worse.

Everything in this section comes from work published up to 23 November 2026, cited at source. Most of the research figures in that window come from preprint abstracts, so the direction of the evidence is considerably firmer than any individual number.

Twelve defences, most of which had reported near zero

In "The Attacker Moves Second", Nasr, Carlini, Tramèr and eleven co-authors **bypassed 12 recent injection defences with attack success above 90 percent for most, where the majority had originally reported near-zero success** (arXiv 2510.09023). They used gradient descent, reinforcement learning, random search and human-guided exploration.

DEFENCE	ORIGINALLY REPORTED	UNDER ADAPTIVE ATTACK
Spotlighting, Prompt Sandwiching	as low as 1 percent in the authors' own static replication	above 95 percent on AgentDojo under search
Circuit Breakers	near zero	100 percent
StruQ	near zero	broken in every case tested
Meta SecAlign	2 percent	96 percent
Protect AI detector, PromptGuard, Google Model Armor	near zero	**bypassed above 90 percent**
PIGuard	near zero	71 percent
DataSentinel	near zero	fell in every case
MELON	near zero	76 to 95 percent

A human red-teaming contest with more than 500 participants and 20,000 dollars in prizes defeated all 40 of its challenges.

The methodological conclusion is the part to keep: defences that reported near-zero static success were often the easiest to break, and an empirical evaluation can only fail to show that a defence is broken. Two of the detectors in this paper's own bench appear in that table.

Independent measurements on realistic documents

Published detector measurements on real inputs arrived this autumn and they are poor.

Over 683 proprietary and 510 public English contracts, scored in windows, **Protect AI v2 flagged 60.61 and 42.55 percent of benign contracts**, deepset's DeBERTa and a DistilBERT guard flagged 91 to 100 percent, and Llama Prompt Guard 2 86M was quiet but weak with paired AUC 0.729 and 0.656 while its 22M sibling sat at chance. The best discriminator still flagged 15.37 and 11.76 percent. **Every guard scored zero on instructions claiming unauthorised scope, and a five-regex baseline was not Pareto-dominated by any of them** (arXiv 2611.08834).

On real agent tool outputs, across 15 detectors, **the best on one benchmark caught 2 percent of another benchmark's injections at a 1 percent false positive rate**, and one that caught 72 percent on AgentDojo caught 15 percent on tau-bench (arXiv 2610.03448). On a 47,611-example set, at a 1 percent miss target, **every gate-plus-reviewer system escalated more than 91.8 percent of inputs to the reviewer** (arXiv 2610.20907). For email agents specifically, **the best of five pretrained detectors reached a mean F1 of 0.216**, and a purpose-built attack-chain model 0.406 (arXiv 2609.30657).

That last figure is the one to hold against a real inbox. A mean F1 between 0.2 and 0.4 at a few thousand emails a day raises many false alerts and still misses many attacks.

What a model judge adds, and where it stops

A prompted frontier model is the best available alert. On those contract corpora one reached 0.00 percent false positives and AUC 1.000, and **a cascade in which it verifies a cheaper model's flags held 0.00 percent false positives at 90.0 percent recall for about 6.5 times less than the frontier model alone.** The authors still advise alerting a human rather than blocking (arXiv 2611.08834).

Judges are attackable in their own right: answer-preserving attacks flipped 46.3 percent of correct security decisions by open decision models, against 0.17 percent for identical reruns (arXiv 2609.31142). And the framing matters more than the model: **monitors framed around the agent's actions scored AUC 0.99 against 0.52 for monitors framed around the content** (arXiv 2610.07089). Judge what the agent is about to do, not what the text says.

Dormant injections, which nothing in this bench tests

Injections that wait for an attacker-chosen trigger succeeded in 43 to 83 percent of runs on nine production agents, against at most 3 percent for the imperative form, were still executed 11.8 percent of the time by a preference-trained model, and left off-the-shelf classifiers miscalibrated (arXiv 2609.22510).

This paper's corpus contains no dormant injections, so it says nothing about them. They are mentioned because they are the clearest evidence that the shape of the attack is a choice the attacker makes after reading the defence.

What held

The other half of the literature is the reason this paper has a recommendation at all. Defences that remove a capability rather than recognise an attack reported zero, at a measurable utility cost.

Across 13,935 attacked AgentDojo episodes on 15 open-weight models, 13 of them between 1.7B and 12B parameters, no attack succeeded against CapLite, which derives its policy from the tool schemas, commits every write call before reading any tool result, and admits a write only if each argument setting its recipient,

amount, object or destination is stated in the user's request (arXiv 2611.17115). Scoped credentials took harmful execution from 8.9 to 37.8 percent under broad bearer tokens down to 0 percent (arXiv 2610.05840). A zero-trust access layer let 2 of 74 attacks through where a plain allowlist let 66 through (arXiv 2610.14488).

Two cautions keep that honest. Separating the planner from the reader is not enough when the plan branches on data: on 101 computer-use tasks built for branch steering, **a vanilla dual LLM design leaked at 89.5 percent** (arXiv 2610.03089). And **none of the 2026 architectural results has been tested by a third party or against an adaptive attacker**, which is precisely the pattern that preceded the collapse of everything in the table above.

What still favours structure is how it fails. A schema-derived rule fails by being too strict or too loose about which arguments untrusted data may touch, and an engineer can read it. A classifier fails silently, on an input nobody anticipated, and tells you it is fine.

08 The leaks that shipped were setup mistakes

> Discharges claims H087 to H100, H138.

While the research literature argued about defences, the retrieval leaks that actually reached production in 2026 were none of the exotic attacks. They were a filter that was computed and not applied, a cache keyed to the wrong identity, and stores left open on the network.

This section is here because a paper about detectors is incomplete without it. **The highest yield security work for anybody building retrieval in 2027 is isolation tests and patch floors, not research defences.**

Eleven of fifteen backends accepted the access filter and ignored it

Open WebUI's knowledge search tool computed which knowledge bases the caller could read and passed that set to the vector store as a search filter. That was its only access check on the path.

Eleven of the fifteen shipped backends accepted the filter in their search method and never applied it: both Qdrant clients, Elasticsearch, OpenSearch, both Milvus clients, openGauss, Oracle 23ai, Pinecone, S3 Vectors and Weaviate. Their query methods did apply it. Chroma, pgvector, MariaDB and Valkey applied it in both.

The exposure was limited to names and descriptions of knowledge bases never shared with the caller, and 0.11.1 fixed it by applying the caller's filter alongside tenant scoping and by

> rejecting unsupported filter operators instead of silently dropping them

(CVE-2026-87017, CVSS 4.3, versions 0.7.0 to 0.11.0) (Kodem).

The test this implies costs an afternoon: **for every store method the code calls, assert that the filter is applied against a fixture the caller must never see.** Nobody writes that test because passing the filter feels like applying it.

A cache keyed by API key is not keyed by person

LiteLLM's semantic cache isolates buckets by API key, and users with different keys could share a bucket, which enabled cross-user leakage of cached responses containing personal data and poisoning of cached answers and tool calls, including a coding agent scenario the researchers describe as code execution. Patched in v1.101.0-rc.1, CVE-2026-89032 (NVISO).

Even working isolation by API key is not isolation by person, because one application key serves every user of that application. The rule that follows is per-tenant or per-principal buckets, no cached responses containing personal data, and never caching tool calls.

Timing channels through shared caches are real and conditional: published attacks succeed up to 100 percent against unprotected serving stacks, and **a salt per principal was necessary and sufficient in a 1,000-trial ablation**, though that was evaluated in simulation by a single author (arXiv 2608.09225). Load blunts the signal, with mean effect size falling from 0.78 with no background workers to 0.21 with two (arXiv 2609.06853), **so a lightly used server shared by a few departments is where the attack is most reliable, and load is not a control.**

The stores themselves shipped critical, network-reachable flaws

STORE	FLAW	RATING
Milvus	**Full REST API including credential management registered without authentication on the default metrics port 9091** (CVE-2026-26190, fixed 2.5.27 and 2.6.10)	CVSS 9.8
ChromaDB	**Python server from 1.0.0 acted on a client-supplied model reference before checking authentication** (CVE-2026-45829)	CVSS 10.0
Qdrant	Arbitrary file write through the logger endpoint for anyone with read-only access	8.8 by NVD

ChromaDB's CVE record names no fixed version, the tracking issue was still open when read, and the reporter's statement that the default Rust server is unaffected has no maintainer confirmation. ChromaDB ships with authentication disabled, and over 1,000 internet-reachable deployments were reported (CSA).

The framework layer above the stores is the larger surface, with 2026 advisories across LangChain, LangGraph and LlamaIndex including a 9.8 SQL injection (OSV).

Two operating rules follow. On-premise is not a control in itself: all three engine flaws are reachable over the network and two need no credentials, so the store belongs on a private network with authentication on and its API, metrics and admin ports treated as separate exposures. And **a store that ran exposed without authentication should be rebuilt from source documents rather than patched**, because poisoned vectors and altered collection settings survive a patch.

The research attacks got cheaper, which is why isolation is the control

Three findings bear on what a leak costs once someone is inside.

Five crafted passages per target question, in a knowledge base of millions, gave 90 percent attack success (arXiv 2402.07867). That is at most 0.0005 percent of the corpus per target, because the attack competes on similarity to the question rather than on volume, **so a bigger corpus does not dilute it.**

Query-time poison filters turned out to recognise how benchmark poison was built rather than poison itself. Of 21 published evaluations, 8 verifiably tested poison carrying the target question as a prefix, which the reference attack's released code prepends, and 12 reported no false positive rate at their deployed threshold. **Removing**

the prefix took one filter from missing 0.191 of the poison to missing 0.763 while the poison was still retrieved for its target question at 0.93 (arXiv 2610.33849).

And extraction converts any gap between what the retriever can read and what the user may read into a near-complete leak: one attack **extracted up to 96.9 percent of a knowledge base** within a fixed query budget (arXiv 2611.12241), and a backdoored third-party generator reached **up to 100 percent success with detection defences removing at most 10 points** (arXiv 2610.25937).

Retrieval as the end user, with the filter inside the search, is therefore the real extraction defence, because it bounds a total leak to what that user could already read.

09 October 2026: the target became the person

> Discharges claims H001 to H002, H043 to H044, H101 to H117.

Between late September and November 2026, injection by email stopped being a researcher demonstration.

Barracuda documented mail built for two readers at once: social engineering for the human and hidden instructions for the AI assistant that summarises or prioritises the inbox, concealed in HTML comments, invisible CSS text, Base64 inside image data and zero-width characters. In its invoice example **the hidden text makes the assistant's summary present a fake priority action to change vendor payment details** (Barracuda).

Sublime Security described a business email compromise campaign with invoices of about 88,000 dollars, a fabricated three-message thread backdated to 1 October 2026, and an injection hidden twice over, in an invisible HTML element and in Unicode Tag characters that mail clients do not render. It **told any email-reading agent to forward the message and its PDF invoice unchanged to an accounts-payable-style address that differed per message.** Fed to several models, some missed the hidden text, some reported it as content, and some recognised the attack (Sublime Security).

Neither vendor published volumes or victim counts, **so how widespread the technique is remains unknown.** That is a real gap and it is not filled by assuming either answer.

On 19 November 2026 ServiceNow's Email Response AI Agent received **CVE-2026-18883 for indirect prompt injection**, CVSS 4.0 score 7.1 with integrity impact high and confidentiality impact low, fixed centrally with no known exploitation (CVE record). The vector describes the realistic harm of an email response agent: altered content sent to customers, rather than bulk data theft.

Why this breaks the usual design rule

The standard rule for agent security is to break the lethal trifecta: untrusted content, access to private data, and a channel to send it out. Remove the outbound leg and exfiltration stops.

The October campaigns do not need an outbound channel. They want the assistant to carry the fraud into the payment workflow, by forwarding an invoice to accounts payable or by surfacing a changed bank account as a priority action. The harm travels through a human being who reads a plausible draft and acts on it.

That is **action steering**, not exfiltration, and no exfiltration-centred rule catches it. The earlier cases point the same way: Copilot email summaries displayed attacker-written security alerts and links inside the trusted interface (CVE-2026-26133) (Permiso); Salesforce Agentforce posted phishing into internal Slack channels under its own identity, entering through a public Web-to-Lead form any outsider can fill (SecurityWeek); and one email reached the Manus agent when a user asked it to check messages, **where the plain instruction was flagged but an obfuscated version was decoded and executed, with the warning shown only afterwards** (CIO Influence). Calendar invitations are now a catalogued channel too, through MITRE ATLAS case study AML.CS0063.

What the vendors shipped was deterministic

Not better detection. Removed capabilities and forced confirmations.

Salesforce made certain Agentforce actions in Slack require user confirmation before sending. Google replaces suspicious URLs in Gemini's Gmail output with "suspicious link removed" and treats user confirmation, URL sanitisation and tool-chaining policies as configuration it can update quickly when a new attack appears (Google). ServiceNow patched centrally.

Every concealment carrier named in the window can be stripped or made visible before the model sees the mail, which is cheap and worth doing. Sublime's test shows models often read hidden text anyway, so stripping is a mitigation and not a boundary.

Approval by lexical matching is a parser differential

The controls that kept failing all interpreted text.

Five Roo Code CVEs published on 13 October 2026 (CVE-2026-110202 to 110206, CVSS 8.8) were auto-approve bypasses reachable through indirect injection, in each of which **the approval parser and the shell read one command line differently**, so the check matched an allowed command such as echo while the shell ran a denied one (CVE record).

Salesforce's URL allowlist fell in 2025 to an expired allowlisted domain bought for five dollars, and again in 2026 because its parser missed some top-level domains and could be confused by certain character sequences, with **content labelled "blocked" after the data had already left.**

Across this autumn's incidents and the year before them, every bypass found defeated a check that interpreted text, and no published bypass in the window defeated a control that removed a capability: no send tool, previews off, no egress. OWASP draws the same line, that "Prompt-level instructions alone do not establish a secure boundary", and asks for egress enforced through independent network controls (OWASP).

What the rulebooks say, and do not

MITRE ATLAS is now the most granular public catalogue of agent techniques, released monthly, and its mitigation AML.M0030, Restrict AI Agent Tool Invocation on Untrusted Data, is the closest official wording to the rule this paper argues for. **OWASP** released its 2026 Top 10 on 1 September 2026 with an Agent Control Standard for runtime enforcement hooks. **ENISA's** Threat Landscape 2026 analysed 8,257 incidents and calls AI integration "a new attack surface" while giving deployers no controls. **NIST** has an agent standards initiative and no finished guidance.

And the **EU AI Act's Article 15(5)** requires high-risk systems to resist attempts to alter their use, outputs or performance by exploiting system vulnerabilities, naming data poisoning, model poisoning, adversarial examples and confidentiality attacks, **but not prompt injection** (AI Act Explorer). It applies to Annex III systems from 2 December 2027, and its supporting cybersecurity standard is still a draft.

Indirect injection fits that wording, so a provider of a high-risk system should document injection controls under it anyway. An email assistant or an internal Q&A system is not obviously in any Annex III area, **which needs checking per deployment** rather than assuming either way, and otherwise falls under general security duties.

10 The design that is left

> Discharges claims H010, H044, H050, H063, H076, H079 to H081, H088, H090, H100, H105, H107, H112, H118 to H128.

If you cannot spot the email, the question becomes what the assistant is allowed to do with one. This section gives the answer per project type, and is explicit about which parts are measured and which are proposed.

PROJECT TYPE	THE CONTROL	VERDICT
Internal document Q&A	Retrieval as the end user with the filter inside the search, no outbound channel, tagged and escaped excerpts, a named answer language	**Lowest risk** once the outbound leg is gone. Residual risk is a wrong answer
Email assistant drafting replies	Draft only, a human sends from the raw draft	**Most exposed.** Acceptable only as draft and approve
Multi-tenant RAG	Index per tenant plus the per-user filter, fail-closed filter operators, caches salted per principal, scoped credentials	**An isolation problem, not a detection problem**
Agents with tools	Schema-derived write rules, scoped credentials, compile-then-execute, previews off, default-deny egress	Architecture and least privilege are the control
On-premise small models	Policy in code rather than in the model, escaped template tokens, patched and isolated stores, pinned weights	**Changes data residency, not injection exposure**
Greek-language users	Language-independent architecture, hidden-content stripping, a named answer language	A multilingual detector serves only as paragraph-level telemetry

The email assistant, in full

This is the exposed case: anyone can write into the context, criminal campaigns now target exactly this deployment, and the harm flows through a person acting on a plausible draft.

> Measured, or at least measured elsewhere:

1. **No send or forward tool in the reading context.** The assistant drafts. A human sends. This is the one control in the list that no published bypass in the window defeated, because there is nothing to defeat. 2. **A human sends from the raw draft, with its real recipients visible.** Not from the assistant's summary of the draft. The Permiso and Barracuda cases are both attacks on the summary, and approving from a summary the attacker influenced is not review. 3. **One thread per draft**, so the drafter cannot reach other mailboxes. 4. **Recipients and attachments limited to what the user's own request names.** This is CapLite's rule, and it is the one with 13,935 attacked episodes and no successes behind it. 5. **Hidden content stripped before the model and shown to the reviewer.** Cheap, and a mitigation rather than a boundary, since models read hidden text anyway. 6. **Draft and summary**

views that fetch no remote images and unfurl no links. Link previews are a confirmed zero-click exfiltration channel. 7. **Model fallback off.** A reported finding on one vendor's system card has safety classifiers routing 53 percent of attempts to an older model that was compromised more often, and since the classifiers read what the model reads, injected text can trigger the switch. **That figure comes from secondary reporting and the card itself could not be fetched**, so it is a reason for caution rather than a measurement.

> Proposed and not measured by anyone, including this paper:

8. **A deterministic check on every draft against the thread and the counterparty's records, flagging any bank details, link domains, recipients or attachments not already on file.** 9. **Risk-tiered review**, so only flagged drafts wait for a slow reviewer.

Item 8 is the control this bench most wants to exist, because it is the only thing in the list that catches the bank account note. **No published study measures drafts-only operation, bank detail checks, link checks or provenance display for email agents.** What is measured is that instruction-level prompt hardening lowered mean attack success from 58.2 to 14.3 percent in a 19-model study, without removing it (arXiv 2610.20776).

Where the detector does belong

Nothing above says do not run a detector. It says do not gate on one.

On this bench the defensible use is **NeuralTrust's small model scored paragraph by paragraph, as telemetry**: it flagged no ordinary email in either language, caught every direct override and every Greek exfiltration link, and costs 392 ms per email on a shared desktop CPU. Run it, log it, alert a human on it, measure its false alarms paragraph by paragraph on your own mail before trusting the rate, and **never let it decide whether a draft goes out.**

The published work agrees on the framing. Monitors framed around the agent's actions scored AUC 0.99 against 0.52 for monitors framed around content. **Ask what the assistant is about to do, not what the email says.**

Internal Q&A, briefly

The lowest-risk type, once the outbound leg is removed: no auto-fetched images, inert links, and retrieval that runs as the end user with the filter inside the search, so an extraction attack reaches only what that user could already read.

Three findings widen that surface and are worth carrying. **Records an outsider can create are untrusted even inside an internal system**, since both Agentforce incidents entered through a public web form. **Documents rendered as images carry instructions a vision model executes**, 77.96 percent of the time for one open model (arXiv 2611.24231). And **injection detectors largely miss knowledge poisoning that carries no instruction**, with one study reporting recall near zero on poisoned passages (arXiv 2610.28364), which is the same blindness this bench found on the subtle style, in a different literature.

On-premise, briefly

Self-hosting changes where the data sits and not how exposed it is. In one study 19 models ranged from 3.2 to 41.9 percent attack success, and on open agents of 2B to 14B parameters

> the system prompt alone moved attack success between 0.2 and 43 percent

(arXiv 2610.23254), so the exact production prompt is part of what has to be benchmarked.

The rule for small models is to **keep the security out of the model.** The approach that asks the model to write a policy program **fails to run in up to 68 percent of episodes at these sizes**, while policy derived in code from tool schemas held on models from 1.7B to 12B (arXiv 2611.17115).

11 Limitations

> Discharges claims H007 to H008, H010, H017 to H018, H021, H024, H029, H034, H072, H080, H085, H103, H129 to H137.

The one that bounds everything

The attacks in this bench are static. No attacker adapts to the detectors. That is the exact condition under which every defence in section 07 reported near zero and then fell above 90 percent once somebody tried.

So the catch rates here are **upper bounds**, and generous ones. A detector that misses the subtle style against five fixed phrasings would miss it by a wider margin against someone rewording until it passed. The misses are the trustworthy half of this bench; the catches should be read as the best case.

What the counts can and cannot carry

The corpus is templated plain text. Twelve benign templates, six hard benign templates and five phrasings per injection style, per language. It is not real HTML mail with signature images, forwarded chains, Greeklish or the formatting debris of a decade-old thread.

So the rates say which texts pass these detectors, not how often real attacks would succeed. That sentence is the honest summary of the whole bench and it is why no percentage in this paper is offered as a detection rate.

The structure of the results supports counting by text rather than by email. **By paragraph, 113 of 120 detector and phrasing pairs were caught in all four placements or in none**, and five of the seven exceptions are Protect AI's, flagged in the late placements for the footer. Hard benign flags came in whole templates. So each injection style's rate counts five phrasings and each hard benign rate six templates.

With those denominators: **five of five caught has an exact 95 percent lower bound of 47.8 percent, and none of five an upper bound of 52.2 percent.** No flag on 60 ordinary emails allows a true rate up to 6.0 percent, or **26.5 percent if only the 12 templates count as independent.**

The design slip

Every second benign email was meant to carry the long quoted thread, but with 12 benign templates the alternation landed on template parity, **so each benign template is always long or always short.** The effect of the long thread on false alarms cannot be separated from the effect of the template.

The injected emails are unaffected, because their position decides their length, so the early-against-late comparison stands.

The two model caveats

The Prompt Guard 2 results belong to gravitee-io's re-publication, not to Meta's weights. Meta's repository is gated and hides its file hashes from anonymous reads, so the copy could not be verified. The safetensors file matches Meta's in size to the byte, 1,115,268,200, and the tokenizer files do not.

NeuralTrust's positive class is named jailbreak, not injection, and the runner scores that class as injection. Its misses on the subtle style may therefore reflect what it was trained to find rather than only how well it finds it, which matters because that model is the one this paper recommends as telemetry.

The machine was not quiet

Other jobs shared the CPU throughout, with load between 8.6 and 23.0 at the end of each run. **Every millisecond figure is an upper bound for an idle machine**, not a benchmark. They are reported because cost turned out not to decide anything, so the imprecision costs nothing.

What is not tested at all

- **Dormant injections** that wait for a trigger, which succeeded in 43 to 83 percent of runs on nine production agents elsewhere.
- **Adaptive attackers**, as above.
- **Real HTML mail**, with the concealment carriers the October campaigns actually used.
- **Footer stripping**, which is the obvious fix for the worst false alarm pattern found and would plausibly make Protect AI's paragraph mode competitive in English.
- **The design in section 10.** Items 8 and 9 in particular, the deterministic draft check and risk-tiered review, are argued from what failed elsewhere and measured by nobody.
- **Greek beyond translation.** The Greek injections are translations of the same five English phrasings, so the bench measures transfer, not how a Greek-speaking attacker would write.

The quoted evidence

Most research figures in this window come from preprint abstracts, not from peer-reviewed and replicated results, and a few from secondary reporting where a primary document could not be fetched. Those are flagged in place. Two items were left out on purpose rather than softened: a vendor post that rendered empty when read, and provider system card figures available only through secondary reporting beyond the one case where the secondary source is named as such.

None of the architectural results in section 07 has been tested by a third party or against an adaptive attacker. That is the same gap that preceded the collapse of the detector literature, and a reader who takes CapLite's zero as settled is making the mistake this paper is about.

Conflict of interest and provenance

Unfunded. No sponsor, client or commissioning party paid for, commissioned, reviewed or approved this research, and no vendor was contacted.

The author builds AI systems commercially, including systems that read mail, which is both the reason for the interest and the conflict. **The corpus is wholly invented**, generated from a fixed seed by a published script: every name, vessel, company, address, IBAN and figure in it is fictional, and it describes no real organisation, correspondence or client.

12 What follows

> Discharges claims H034, H044, H081, H085, H088, H096, H112, H118, H120, H122, H134, H138 to H141.

If you run an email assistant

1. Take away the send tool. It drafts, a person sends, and the person sends from the raw draft with its real recipients visible rather than from the assistant's summary. This is the only control in this paper that no published bypass in the window defeated.

2. Limit recipients and attachments to what the user's own request named. One thread per draft. This rule has 13,935 attacked episodes and no successes behind it, on models from 1.7B to 12B.

3. Turn link previews off and fetch no remote images, in every view a human or an agent sees. Previews are a confirmed zero-click exfiltration channel.

4. Strip hidden content before the model and show it to the reviewer. Every concealment carrier used in the October campaigns can be made visible. Models read hidden text anyway, so this is a mitigation and not a boundary.

5. Build the deterministic check. Compare every draft against its thread and the counterparty's records, and flag any bank details, link domain, recipient or attachment not already on file. **Nobody has measured this, including this paper**, and it is the only thing in the list that would catch the bank account note.

6. Run a detector, and never gate on it. On this bench, NeuralTrust's small model by paragraph, as telemetry, with its false alarms measured paragraph by paragraph on your own mail first.

If you run retrieval

Test the filter, do not pass it. For every store method your code calls, assert that the access filter is applied against a fixture the caller must never see. Eleven of fifteen backends accepted a filter and ignored it in production software this year.

Key caches by principal, not by API key, cache no responses containing personal data, and never cache tool calls.

Set a patch floor and treat on-premise as a location, not a control. Milvus 2.5.27 or 2.6.10, Qdrant 1.16.0, ChromaDB's Rust server or an isolated Python one, authentication on, network isolated, API and metrics and admin ports treated as three separate exposures. **A store that ran exposed without authentication should be rebuilt from source documents**, not patched, because poisoned vectors survive a patch.

Retrieve as the end user with the filter inside the search. It is the only thing that bounds a total extraction to what that user could already read.

If you are deciding whether to measure anything yourself

Three lessons from running this bench, which cost an afternoon of CPU time.

Score the hard cases, not the easy ones. Any classifier separates "ignore all previous instructions" from a freight offer. The bench only became informative when it contained thirty emails per language of colleagues writing "disregard the invoice sent this morning", and that is where every detector either stayed quiet or fell apart.

Vary how you cut the input, not just which model you use. Where the email was cut changed the answer more than the choice of model did: whole-email scoring missed all 64 late injections for two of the four detectors, and paragraph scoring made the best model much better and the worst one unusable.

Then look at what fired it. The single most useful script in this bench is twenty lines long: it scores each benign paragraph alone and prints which ones fire the detector. It is what turned "Protect AI flags half of ordinary English mail" into "the confidentiality footer is an instruction", and no amount of reading the aggregate table would have found that.

What is worth measuring next

The deterministic draft check, because it is the control this bench most wants and nobody has tested it. A bench for it is straightforward: drafts generated against a counterparty record set, with the check's false alarm rate measured on ordinary corrections to real payment details, which do happen.

Footer stripping before detection, because it is one preprocessing step away and would plausibly change which detector wins in English.

Greek written by Greek speakers, rather than translated from English phrasings, with Greeklish and homoglyph mixing, since this bench measures transfer and not the attack as it would actually arrive.

And an adaptive attacker against the structural defences, which is the result the whole field should be waiting for. Everything that has met one has fallen; the controls this paper recommends have not met one yet.

The one line version

Detection of malicious email failed in every independent measurement this autumn, including this one: the request that invoice fraud actually uses, a polite note asking the drafter to confirm a new bank account, passed every detector that kept ordinary mail clean. What held, everywhere it was tested, was taking the capability away. **So do not try to spot the email. Make sure the assistant cannot act on it.**

ABOUT THE RECORD

A local bench of four open detectors over 308 emails generated from a fixed seed. Every name, vessel, company, address, IBAN and figure in the corpus is invented; it describes no real organisation or correspondence. The corpus, the generator, the runner and every per-email score are published with the paper.

WHAT IT SHOWS

Detection of malicious email failed here as it did in every independent measurement published this autumn, while controls that remove a capability held wherever they were tested. The conclusion is to limit what an assistant may do with an email rather than to try to recognise one.

LIMITS AND CONFLICTS

The attacks are static, so the catch rates are upper bounds; the mail is templated plain text, so the rates say which texts pass rather than how often real attacks would succeed. Most quoted research figures come from preprint abstracts. The author builds AI systems commercially, including systems that read mail. Unfunded, and no vendor was contacted.

© 2026 Nikolaos Broikos. Text and data under CC BY 4.0, code under MIT. The data, code and claim ledger are in data/paper-13/ so any figure here can be recomputed, checked or disputed.