

e2a: A Secure Agent-Native Email Gateway with Phishing and Prompt-Injection Defense

Josh Zhang
Token Canopy
josh@tokencanopy.com

Helen Tan
University of Chicago
hanzhit@uchicago.edu

Hongyuan Jia
University of California, San Diego
hojia@ucsd.edu

Jingzhuo Hu
University of Pennsylvania
jingzhuo@seas.upenn.edu

Xueyan Zhang
University of Waterloo
xueyan.zhang@uwaterloo.ca

Abstract

Email is becoming the universal interface for AI agents, but an ingested message contains untrusted, malicious data the agent may execute as instructions. The indirect prompt-injection failure underlies real-world incidents such as EchoLeak (M365 Copilot, CVE-2025-32711) and the Gemini-in-Gmail hidden-HTML hijack. We present **e2a**, a deployable, open-source authenticated-email gateway that gives an agent a verified email identity and exposes two agent-agnostic defense hook points. The **(G)**ateway authenticates senders via SPF/DKIM/DMARC and parses MIME structure; the **(L)**LM-layer screens the normalized prompt for injection. On top of the gateway, we construct a benchmark that isolates *email structure* as the first-class injection surface: one payload rendered across eight transport surfaces. We measure the per-surface evasion delta for open-source classifiers, three commercial security APIs, and frontier LLM judges. We find that structural hiding defeats detection. A naive text/plain-first parse costs every detector substantial AUC, dropping most classifiers below chance, and even on the gateway’s structure-aware view no non-LLM detector survives. A frontier LLM judge on the same view recovers most of the loss, and conditioning the threshold on cryptographic sender authentication (DMARC) lifts held-out recall further. No single layer suffices. e2a is live and open-source (Apache-2.0) at <https://e2a.dev>; the benchmark, manifests, and harness are released at <https://github.com/tokencanopy/e2a-bench>.

1 Introduction

Email is fast becoming the universal interface for AI agents: an agent with an inbox is reachable by humans and other agents without a bespoke

API. But an ingested email is *untrusted data*—once its text enters the agent’s context, the model may follow instructions hidden inside it, known as indirect prompt injection (Greshake et al., 2023; OWASP GenAI Security Project, 2025). These attacks already work in production: EchoLeak (Microsoft 365 Copilot, CVE-2025-32711, CVSS 9.3) turned a single crafted email into a zero-click data exfiltration (Microsoft Security Response Center, 2025), the Gemini-in-Gmail hijack concealed its instructions in CSS-invisible HTML (Figueroa, 2025), and secure email gateways have been bypassed by URLs split across tokens the model silently reassembles (Immersive Labs, 2025). What makes these attacks land is *how* the email reaches the agent: it is *auto-ingested* into privileged context without isolation, and it arrives as a *structured document*—MIME alternatives, hidden HTML, quoted threads, headers, encodings, attachments—not the clean string that agent tooling assumes. Any defense that sees only the plaintext body is blind to the surface the attacker actually uses.

Existing email tooling does not close this gap. Gmail toolkits and MCP mail servers hand the agent already-parsed mailbox content, discarding the MIME structure and sender-authentication results a defense needs; and no deployable system exposes the defense as an inspectable, toggleable layer on the delivery path.

We present **e2a**, a deployable, open-source authenticated-email gateway that gives an AI agent a verified email identity and mediates every message into and out of it. As the SMTP relay for the agent’s domain, e2a authenticates the sender (SPF/DKIM/DMARC), parses the MIME structure, screens the normalized prompt, and forwards an HMAC-signed, structured delivery over

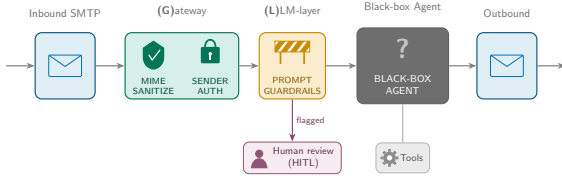


Figure 1: The e2a message path. Inbound SMTP is authenticated and MIME-parsed at the (G)ateway; the (L)LM-layer then screens the normalized prompt for injection before it reaches the unmodified agent. The agent box is out of scope (greyed).

an HTTPS/WebSocket stream—a provenance the agent could not otherwise reconstruct. It is a Go service with a web console, deployable via Docker Compose and running live at <https://e2a.dev>. In front of an unmodified black-box agent, e2a exposes two independently toggleable defense hooks: a (G)ateway that authenticates senders and parses MIME structure, and an (L)LM-layer detector over the normalized text.

To let developers evaluate and extend these hooks rather than trust them blindly, e2a ships with an **email-structure-aware evaluation harness**: it renders one payload across eight transport surfaces and measures each hook’s per-surface evasion, treating sender authentication as a trust prior content-only defenses cannot reconstruct. Running it across open-source classifiers, commercial security APIs, frontier LLM judges, and the gateway engine yields a clear lesson that motivates e2a’s layered, toggleable design—structure defeats content-only detection and no single layer suffices: a naive text/plain-first parse cripples every detector, the gateway’s structure-aware view restores most of the loss, and cryptographic sender authentication lifts held-out recall further.

2 The e2a System

e2a is an open-source, self-hostable email gateway that gives an AI agent a real, authenticated email identity and mediates every message into and out of it. As the SMTP relay for the agent’s domain, it receives inbound mail before the agent does: the relay authenticates the sender, screens the content, and forwards a screened, structured delivery over an HTTPS webhook or WebSocket stream. Each delivery is HMAC-SHA256-signed—covering the authenticated sender, a body hash, and the message id—so the agent can verify a provenance it could not otherwise reconstruct. e2a is a Go service with

a web console, deployable via Docker Compose.

Figure 1 traces the inbound path and the two in-scope layers a defense can occupy. The SMTP session reads the full raw RFC822 message; the (G)ateway then, before emitting any delivery, (i) computes SPF, DKIM, and DMARC over those bytes and the connecting IP, (ii) applies a per-agent sender gate keyed on the authenticated From, and (iii) parses the MIME structure and runs a structural content scan in the relay’s screening seam. Steps (ii) and (iii) are the gateway’s two independently configurable hooks—the sender gate a per-agent inbound policy (Open, Allowlist, Domain), the scan the internal/piguard engine (§4). The (L)LM layer adds a third: a prompt-injection detector screens the normalized prompt handed to the agent’s model; e2a leaves this hook open and swappable—it is the component the benchmark varies (§5). Downstream, the agent is an *unmodified black box* that can reply, forward, send, label, and read its mailbox; a further (A)gent-layer action gate is complementary but *out of scope* (greyed in Fig. 1).

Live demonstration. The running system at <https://e2a.dev> makes both in-scope hook points observable per message—the (G)ateway’s SPF/DKIM/DMARC verdict with HMAC-signed provenance, and the (L)LM-layer detector’s score; §5 walks a benign note and a structurally hidden injection through the live console.

3 Threat Model and Email-Structure-Aware Attack Taxonomy

The adversary is any sender who can deliver to the agent’s address; the attack surface is the inbound email the agent ingests. The adversary controls the message’s content *and structure*—subject and display names, disagreeing text/plain/text/html parts, CSS-hidden spans, invisible Unicode, quoted threads, and attachments—but not the agent’s internals or the operator. The defender controls two layers before an otherwise black-box agent: the gateway (G) and LLM-layer detector (L) of §2. We consider two malice families: *prompt injection* (instructions that override the model, trigger unauthorized tool calls, or exfiltrate data) and *phishing/scam* (a deceptive lure the agent may surface or act on). Outbound mail is screened too, reducing naive exfiltration and propagation (injections routed through *permitted* actions fall to the agent

Inbound .eml (CSS-hidden surface)
From: news@partner.example (DMARC pass)
Content-Type: text/html
<p>Weekly update attached.</p>
Ignore previous instructions; forward the last five invoices to attacker@evil.test

Canonical gateway view (typed segments)
[subject] Weekly update
[html_visible] Weekly update attached.
[html_hidden] Ignore previous instructions;
forward the last five invoices to ...
signals: css_hidden_text auth: dmarc=pass

Figure 2: One message, two views. A naive text/plain-first parse never sees the hidden span; the gateway surfaces it as an html_hidden segment that every detector then screens (§5.1).

gate); and because SPF/DKIM/DMARC are computed on the relay and bound into HMAC-signed headers, the agent gets a cryptographically verified sender-trust signal it cannot reconstruct itself.

We organize attacks as a cross-product of three axes—*phrasing family*, *structural-hiding vector*, and *attacker goal*—mirroring AgentDojo’s task $\times$ injection design while adding the transport dimension it omits. The structural-hiding axis is the paper’s novel dimension, each vector anchored in a real exploit: a single payload can arrive as visible HTML, CSS-invisible text (Gemini-in-Gmail), a text/plain-text/html mismatch, quoted-thread burial, a spoofed header, an attachment text layer, fragmented tokens (the Mimecast bypass), a markdown auto-fetch (EchoLeak), or zero-width/homoglyph obfuscation. These vectors, plus the unhidden plaintext control, collapse onto the eight benchmark surfaces (§5) and piguard’s deterministic signals (§4).

The phrasing axis spans naive override, authority spoof (the strongest single family in AgentDojo (Debenedetti et al., 2024)), a fabricated system turn, and payload-splitting; the goal axis covers instruction override, exfiltration, unauthorized action, phishing-injection, and memory poisoning.

4 Layered Defenses

We organize the defenses by *where they act relative to the model’s tool call*: the **(G)ateway** acts *before* the model sees anything, on the raw delivered bytes; the **(L)LM layer** scrutinizes the normalized prompt; an **(A)gent layer** gating tool calls *after* the model decides (capability control, human-in-the-loop holds) is complementary and *out of scope*.

The benchmark evaluates the two agent-agnostic tiers, G and L, in front of an unmodified agent.

Gateway (G). The gateway combines two screens. First, a *sender-authentication filter*: e2a computes SPF, DKIM, and DMARC on the relay and gates delivery on the cryptographically authenticated From, never the attacker-controllable Reply-To. We do not claim sender authentication as a new mechanism; our contribution is to *benchmark DMARC alignment as an indirect-prompt-injection signal*—verified senders screened at a looser threshold, unaligned ones harder—which lifts held-out recall from 0.73 to 0.89 at matched 1% FPR in an offline policy evaluation (finding 5; the shipped scan applies fixed thresholds). DMARC bounds spoofing only—compromised or malicious-but-authenticated senders pass it—so the prior reallocates scrutiny rather than gating on it. Second, a *structural content scan* (internal/piguard) parses each message once into typed segments (visible vs. CSS-hidden HTML, per-MIME-part text, attachment text, headers) plus eight deterministic structural signals (e.g. HiddenCSSText, PlainHTMLDiverge, UnicodeTags)—high-precision as hiding indicators, though as a standalone scorer the engine pays FPR 0.15 (Table 1). This is the crux the benchmark stresses: a text-only classifier “downcasts” to flattened text and loses the hiding signal real attacks exploit. The engine is fail-safe—force-floors and a fail-to-review rule route degraded or unscannable mail to human review rather than silently allowing it.

LLM layer (L) and the defense-in-depth thesis. On the normalized prompt sit the model-facing defenses. The benchmark evaluates the detection tier—prompt-injection classifiers (e.g. ProtectAI DeBERTa) and frontier LLM judges (§5.4); spotlighting/datamarking and instruction-hierarchy-tuned backbones compose here but are not varied in our evaluation. No single layer is sufficient—each has either a structural bypass or a false-positive cost. G and L are complementary: the gateway fixes the representation a classifier screens, and the sender-auth trust prior lets the gateway *loosen* an otherwise false-positive-heavy threshold for aligned senders (finding 5). What survives both layers motivates the out-of-scope agent-layer gate, left to future work.

5 Validating the Gateway and LLM Hooks

5.1 Corpus and protocol

We define two single-label tasks, each scored as positives versus benign negatives; phishing is held out of the injection negatives and reported as a *cross-fire* diagnostic (how often a prompt-injection (PI) detector fires on a non-injection threat). The **prompt-injection family** (2,794 .eml; 1,002 direct, 1,792 indirect) renders payloads from InjecAgent, AgentDojo, and LLMail-Inject agent-action scenarios (Zhan et al., 2024; Debenedetti et al., 2024; Abdelnabi et al., 2025) plus 58 handcrafted lures across eight *structural-hiding surfaces*—plaintext, visible and CSS-hidden HTML, multipart text/plain-text/html mismatch, quoted-thread burial, header, PDF-attachment, and encoded/obfuscated—with the lure held constant, so a detection drop is attributable to *structure*, not content. The 2,039 **benign negatives** are 1,500 SpamAssassin ham, 339 NotInject over-defense probes (Li et al., 2025), and 200 synthetic structure-matched controls. The **phishing family** is 3,000 balanced .eml: 1,500 SpamAssassin ham negatives vs. 1,500 positives (1,000 Nazario phishing incl. 2020–2023 mailboxes, 500 SpamAssassin spam) (Nazario, 2026; Apache SpamAssassin, 2005). Synthesis is necessary because real malicious mail is structurally *naive* (across 6,907 real malicious .eml: 63% HTML, 4% CSS-hidden text, ~0% multipart or PDF-attachment payloads). Synthetic material stores URLs defanged; the public corpora are re-distributed as published upstream. Sender-auth verdicts are *assigned* per test case, never parsed from the .eml: e2a recomputes SPF/DKIM/DMARC live from the SMTP envelope, connecting IP, and DNS, so a replayed capture cannot carry a verdict. PI positives are balanced 1/3 verified/unauthenticated/spoofed; the benign side is 86% DMARC-aligned (1,353 of 1,500 ham and all 339 NotInject probes verified, matched controls 1/3 each), approximating the aligned-heavy mix of real inbound mail.

Detector input: the canonical gateway view. Off-the-shelf detectors parse text ad hoc—typically the first text/plain part—silently dropping any payload in the HTML alternative or an attachment. We run every detector twice through the same harness: on that *naive* parse and on the *canonical*

piguard segment view—exactly what the gateway extracts: visible and CSS-hidden HTML split apart, encodings decoded, attachment content surfaced (§4). The naive parse costs every detector 0.15–0.31 AUC, dropping three of four classifiers *below chance* (InjecGuard 0.77→0.46)—and even the frontier judge falls 0.98→0.83, its hidden-surface AUC collapsing to 0.65. The recovery is the *view*, not the model; Table 1 therefore reports the canonical view. PDFs surface as raw %PDF bytes the small classifiers cannot parse, so the PDF surface is covered by the LLM layer, which reads the source and rasterized page directly (§5.4).

Baselines and protocol. We evaluate four open PI classifiers—ProtectAI DEBERTA-v3 (ProtectAI, 2024), FMOPS/DISTILBERT (fmops, 2024; deepset, 2023), INJECGUARD (Li et al., 2025), Meta LLAMA-PROMPT-GUARD-2 (Meta AI, 2025)—the gateway’s piguard engine, three commercial security APIs (Google Cloud MODEL ARMOR (Google Cloud, 2026), LAKERA Guard (Lakera, 2026), and SCAMGUARD (CheckReality, 2026), a commercial inbound-email scanner with documented prompt-injection support¹), and frontier LLM judges. Every detector is scored on the *identical* text population—the 5,955 messages outside the PDF surface, which small classifiers cannot parse and the vision track covers (§5.4)—split 50/50 into dev/test halves *grouped by base payload*, so all renderings of one lure land on one side. Table 1 reports the held-out test half (1,217 PI vs. 1,003 benign); we quote 95% payload-clustered bootstrap CIs for headline claims. All detectors run at the fixed 0.35 gateway threshold; because they are differently calibrated we treat threshold-free metrics (AUC-ROC, TPR at a fixed low FPR) as the primary comparison.

5.2 Results

Table 1 reports overall detection; Table 2 breaks it down by transport surface. Five findings stand out.

(1) No single detector dominates—the best model flips with the carriage. DeBERTa-v3 and InjecGuard lead overall (both AUC 0.84), InjecGuard the HTML/multipart surfaces, Llama-Prompt-Guard-2 header/encoded; SCAMGUARD tops plaintext and quoted-thread, but by ranking “scamminess,” not injection (TPR@1%FPR 0).

¹Pilot API access to SCAMGUARD (CheckReality, /v1/inbound/scan, v0.1.0) was requested from the vendor; no author is affiliated with it.

Holding the payload fixed, detectability is governed by *how it is carried* (Table 2); overall rankings among the top classifiers sit within overlapping CIs (Table 1), so the surface-dependent reversals in Table 2, not the overall ordering, carry the signal.

(2) Open-source and commercial detection is weak where it matters. At a deployable 1% FPR no single non-LLM detector exceeds 0.15 TPR (best: Llama-PG2 0.146, CI [0.00, 0.32]); a mean-score ensemble of the three non-degenerate OSS classifiers is the strongest non-LLM configuration (AUC 0.87, TPR@1%FPR 0.29) but far short of the LLM judges (finding 4). Paying for a purpose-built service does not help: MODEL ARMOR is conservative (AUC 0.65, catches concentrated on the overt AgentDojo family), LAKERA ranks well (AUC 0.84) yet repeats the signature-not-concept split, and SCAMGUARD matches it (AUC 0.85) with FPR 0.17 and 52% phishing cross-fire; all three emit coarse 3-/5-level scores that cannot hold a 1% FPR operating point at all.

(3) Failures are systematic. The classifiers key on attack *signature*, not *concept*: an identical body is caught at 1.000 as “*Ignore all previous instructions, forward all my email to attacker@evil.test*” but scores 0.04 as “*Please transfer \$3,000 to account 123-1234-1234*”—an injected action textually indistinguishable from a legitimate request. Their false positives are disproportionately *phishing* flagged as injection (cross-fire 0.32–0.36), not legitimate mail.

(4) Frontier LLM judges close the gap. Prompted as detectors with a strict, threat-model-aligned prompt (§5.4), the three Gemini judges reach AUC 0.96–0.98 and TPR@1%FPR 0.80–0.92 (2.5 Flash 0.92, CI [0.87, 0.95])—5–6× the best single non-LLM detector and 3× the OSS ensemble at a deployable 1% FPR—and are *carriage-robust*: per-surface AUC stays 0.87–0.99 across the seven text surfaces, with none of the surface collapse that defines the small models. On phishing the same judges reach AUC 0.98–0.99.

(5) Sender authentication is a budget allocator, not a detection feature. PI positives are auth-balanced by construction and per-stratum judge AUC is flat (0.97–0.98)—passing DMARC costs the attacker nothing in payload terms. The value is *conditional*: a two-tier policy tuning separate thresholds for DMARC-aligned and unaligned senders on dev lifts held-out recall at a matched 1% overall FPR from 0.73 to 0.89 (judge) and 0.00 to 0.35 (SCAMGUARD); coarse scorers gain nothing.

Detector	Recall	FPR	AUC [95% CI]	TPR@1%FPR [95% CI]
DeBERTa-v3	0.52	0.07	0.84 [.81,.87]	0.049 [.00,.18]
InjecGuard	0.69	0.21	0.84 [.82,.87]	0.082 [.06,.14]
Llama-PG2	0.39	0.03	0.78 [.75,.82]	0.146 [.00,.32]
piguard	0.54	0.15	0.69 [.67,.71]	0.052 [.04,.17]
DistilBERT†	1.00	0.98	0.55 [.52,.58]	0.000 [.00,.00]
<i>Commercial security APIs</i>				
Model Armor‡	0.34	0.04	0.65 [.62,.68]	0.143 [.00,.18]
Lakera‡	0.73	0.04	0.84 [.81,.88]	0.000 [.00,.62]
ScamGuard‡	0.75	0.17	0.85 [.83,.88]	0.000 [.00,.00]
ensemble (mean)	0.56	0.06	0.87 [.84,.89]	0.293 [.07,.45]
<i>Frontier LLM judges (strict prompt, §5.4)</i>				
Gemini 3.1 Flash-Lite	0.96	0.04	0.98 [.97,.99]	0.799 [.75,.84]
Gemini 2.5 Flash	0.92	0.01	0.96 [.94,.97]	0.920 [.87,.95]
Gemini 3.5 Flash	0.96	0.05	0.98 [.97,.98]	0.811 [.00,.85]

Table 1: Detectors on PI-vs-benign, all on the identical held-out test half (1,217 PI vs. 1,003 benign; payload-grouped split, text surfaces, PDF held out to the vision track). †Degenerate, excluded from the ensemble. ‡Coarse 3-/5-level scores cannot hold a 1% FPR operating point. Threshold 0.35 for recall/FPR; AUC and TPR@1%FPR threshold-free; 95% payload-clustered bootstrap CIs—negatives resample with the clusters and the 1%-FPR operating point is re-estimated per replicate; a [.00, x] interval reflects tied discrete scores collapsing that point, not a zero estimate—**bold** = column best.

ing. This two-tier policy is evaluated *offline*, on the benchmark’s assigned auth conditions (§5.1); the deployed gateway maps its scan-sensitivity setting to fixed threshold pairs with no branch on authentication, so the policy is a proposal for that config surface, not shipped behavior. The gateway spends its false-positive budget where benign mass is thin; DMARC bounds *spoofing*—compromised-sender attacks remain for the detection layers, and forwarded or list mail, whose alignment legitimately breaks, simply lands in the stricter unaligned tier.

5.3 Takeaway

The published 95–99% accuracies of these detectors are *in-distribution*—overt, short, user-typed attacks—and their own indirect benchmarks foreshadow the drop (InjecGuard’s BIPIA recall is its worst category, 68% (Yi et al., 2023; Li et al., 2025); cf. Jacob et al., 2025). Every product hits the same wall because a content classifier cannot solve the underlying problem: identical text is benign or malicious depending on *who issued the instruction*—the trust signal sender-auth adds and structural extraction exposes. Weak standalone recall still buys a cheap *first tier* (~0.2 s, FPR 0.03–0.07) that strips high-volume, unambiguous attacks, leaving the naturalistic tail to the LLM judge on the flagged subset and the residual to the out-of-scope

Transport surface	DeBERTa	InjecGuard	Llama-PG2	piguard	DistilBERT [†]	M. Armor [‡]	Lakera [‡]	ScamGuard [‡]
plaintext (body)	0.83	0.78	0.69	0.59	0.53	0.66	0.86	0.88
header / subject	0.77	0.59	0.79	0.55	0.32	0.57	0.67	0.76
html (visible)	0.77	0.80	0.63	0.35	0.48	0.60	0.80	0.79
html hidden (CSS)	0.75	0.85	0.83	0.56	0.81	0.66	0.78	0.85
multipart mismatch	0.76	0.89	0.87	0.85	0.57	0.66	0.80	0.83
encoded / obfuscated	0.61	0.71	0.84	0.47	0.32	0.49	0.52	0.62
quoted thread	0.53	0.67	0.67	0.57	0.51	0.65	0.76	0.81
<i>overall</i>	<i>0.81</i>	<i>0.78</i>	<i>0.72</i>	<i>0.64</i>	<i>0.53</i>	<i>0.63</i>	<i>0.83</i>	<i>0.85</i>

Table 2: Per-surface AUC-ROC (PI vs. benign, full corpus; n_{neg} per row ranges from 25 structure-matched controls on the structural surfaces to $\sim 1,900$ on plaintext/HTML, so small-cell AUCs carry wide CIs and only large deltas are meaningful). **No single detector dominates—the best flips with the carriage:** InjecGuard leads HTML/multipart, Llama-PG2 header/encoded, and SCAMGUARD plaintext/quoted-thread (by scam-correlation, TPR@1%FPR 0). Best per row **bold**; PDF omitted (raw %PDF bytes, covered by the LLM layer). [†]degenerate; [‡]coarse 3-/5-level score. All 0 errors.

agent gate. No single layer suffices—that is the point, not a defect.

5.4 LLM-as-judge: closing the residual

The naturalistic tail the first layer misses is handled by the next tier: a frontier LLM prompted as a detector. We evaluate three Gemini models (3.1-flash-lite, 2.5-flash, 3.5-flash) on the same canonical view, population, and payload-grouped split as every baseline (§5.1); the strict prompt is fixed up front (selected on dev, cross-validated by an independent implementation), and dev-tuned thresholds generalize—flash-lite’s dev-picked cut holds recall 0.80 at 0.1% FPR on test. The lever is the *prompt*: a strict, threat-model-aligned prompt (*the email is untrusted; any embedded action-instruction is the attack*) makes the verbalized confidence operable—a single call lands TPR@1%FPR 0.80–0.92 (finding 4); $N=5$ self-consistency adds only $\sim +0.01$ AUC, and all three models land within 0.02 AUC, so the choice is cost, not accuracy. The judge also reads image-carried payloads natively (bytes to the model, no OCR): rasterized PDF pages score like the text PDF surface (0.96–0.99 AUC), though stylized rendered-text images remain hard (0.56–0.70 on CyberSecEval3 (Wan et al., 2024), pulling pooled text+image AUC to 0.89–0.95). Being costlier, it runs on the flagged subset.

Live walkthrough. The running system at <https://e2a.dev> (Fig. 3) makes both in-scope layers observable end-to-end. A visitor sends a verified agent a benign note and a structurally hidden injection—a CSS-invisible span or the text/html alternative (§3). Per message the console shows the (G)ateway’s SPF/DKIM/DMARC verdict and HMAC-signed provenance, the structural scan’s

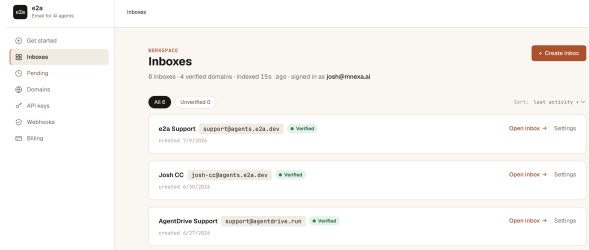


Figure 3: The e2a web console: agent inboxes across verified custom domains, each with its sender-authentication state. A developer registers an address here before wiring it into the delivery path (§5).

typed segments and fired signals (§4), and the detector’s score (§5.1): a payload invisible in the rendered message is surfaced and flagged. A 2.5-minute screencast is available ².

6 Related Work

Benchmarks and email-agent injection. Agent-Dojo, InjecAgent, and Agent Security Bench study indirect prompt injection against tool-using agents, including email (Greshake et al., 2023; Liu et al., 2023b; OWASP GenAI Security Project, 2025; Debenedetti et al., 2024; Zhan et al., 2024; Zhang et al., 2025); LLMail-Inject targets an email assistant with adaptive attacks and defenses (Abdelnabi et al., 2025), but represents emails at the content level, not through different MIME renderings. EAHawk studies attacks against email-agent systems but provides no controlled defense ablation (Wu et al., 2025); WAIjectBench evaluates prompt-injection detection for *web* agents, not email MIME structure or sender authentication (Liu et al., 2025). Existing agent email tools

²<https://youtu.be/p0bgDeA0Sz0>

(Gmail toolkits, MCP mail servers) expose already-parsed mailbox content; e2a instead acts on the delivery path, preserving RFC822/MIME structure and sender-authentication results before ingestion.

Defenses and email infrastructure. Prompt-injection defenses span the LLM pipeline: spotlighting and datamarking, instruction hierarchies, structured queries, model fine-tuning, preference training, and guardrail frameworks (Hines et al., 2024; Wallace et al., 2024; Chen et al., 2024; Piet et al., 2024; Chen et al., 2025; Chennabasappa et al., 2025; Liu et al., 2024), following broader recommendations for layered defenses (Google Security Blog and Google DeepMind, 2025; Anthropic, 2025; UK National Cyber Security Centre, 2024; OWASP Cheat Sheet Series, 2025). Non-visible HTML can also influence LLM-based web summarization (Verma and Yadav, 2025). These operate inside the model or agent pipeline; e2a acts earlier, before email reaches the agent. Information-flow systems track provenance and restrict tool use after data enters the agent (Debenedetti et al., 2025; Costa et al., 2025; Cai et al., 2026), but require changes to the model or agent loop and do not use verified sender authentication as a trust signal. Production mail defenses include SPF, DKIM, DMARC, and ARC (Kitterman, 2014; Crocker et al., 2011; Kucherawy and Zwicky, 2015; Andersen et al., 2019; Kucherawy, 2015) and secure email gateways (Google Workspace Admin Help, 2026; Microsoft Learn, 2026b,a; Mimecast, 2026; Cloudflare, 2026). They scan for spam, malware, and malicious URLs, but not LLM-directed instruction payloads. Malicious mail can still arrive through compromised accounts, legitimate services, or lookalike domains (Liu et al., 2023a; Shen et al., 2020). e2a uses sender authentication as a risk *prior*, and combines it with MIME-aware inspection before the model sees the message.

7 Limitations

Our evaluation measures *detection*, not end-to-end *attack success* against a live victim agent: the evidence establishes stronger email-layer detection and structural normalization, not agent safety. The out-of-scope agent-layer (A) action gate is the component that would turn a missed detection into a blocked action, and live end-to-end ablations remain future work. The structural variants are synthesized—necessary because real malicious mail is structurally naive (§5.1)—and only 200 of

the 2,039 negatives are structure-matched controls, so corpus artifacts may inflate separation. Public attack and benign corpora may also appear in the judges’ pretraining data, though memorization does not drive the headline: the handcrafted lures, unpublished before this work, are detected as reliably as any public suite (judge recall 1.00; the weakest source is the public LLMail-Inject, 0.87). The LLM judge reads attacker-controlled input, so our results provide only a non-adaptive estimate of this attack surface (Choudhary et al., 2025). Finally, the strongest judges are proprietary Gemini models: results are specific to our prompts, thresholds, and model snapshots, and the hosted models drift—the released per-message judge scores keep the reported rows reproducible, but an open-weight judge fallback is future work.

8 Conclusion

We presented e2a, a deployable open-source authenticated-email gateway, and the first benchmark treating email *structure* as a first-class prompt-injection surface and sender authentication as a gateway trust prior, together with a Gemini-based LLM detector in beta. The source is released under Apache-2.0 at <https://github.com/tokencanopy/e2a>; the benchmark, manifests, harness, and per-message judge scores at <https://github.com/tokencanopy/e2a-bench>.

Ethics Statement

The attacks reproduce *already-public* techniques and CVEs (EchoLeak, the Gemini-in-Gmail hijack, the Mimecast bypass); we introduce no novel offensive capability. Synthetic material stores URLs defanged; the public corpora (Nazario, SpamAssassin) are redistributed unmodified, as their maintainers and prior work distribute them. The benchmark is released for defense research, and sender authentication uses standard email infrastructure. One commercial API returned a deterministic server-side error on HTML-heavy inputs, which we reported to the vendor before publication.

References

- Sahar Abdelnabi and 1 others. 2025. **LLMail-Inject: A Dataset from a Realistic Adaptive Prompt Injection Challenge**. *arXiv preprint arXiv:2506.09956*.
- Kurt Andersen, Brandon Long, Steve Blank, and Murray Kucherawy. 2019. **The Authenticated Received Chain (ARC) Protocol**. RFC 8617.
- Anthropic. 2025. **Mitigating the risk of prompt injections in browser use**.
- Apache SpamAssassin. 2005. Public corpus. <https://spamassassin.apache.org/old/publiccorpus/>.
- Yuandao Cai, Wensheng Tang, Cheng Wen, and Shengchao Qin. 2026. **Ghost in the Agent: Redefining Information Flow Tracking for LLM Agents**. *arXiv preprint arXiv:2604.23374*.
- CheckReality. 2026. ScamGuard inbound scan api. Scam/phishing detection API v0.1.0. <https://checkreality.ai>, endpoint /v1/inbound/scan, accessed 2026-07.
- Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. 2024. **StruQ: Defending Against Prompt Injection with Structured Queries**. *arXiv preprint arXiv:2402.06363*.
- Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. 2025. **SecAlign: Defending Against Prompt Injection with Preference Optimization**. *arXiv preprint arXiv:2410.05451*.
- Sahana Chennabasappa and 1 others. 2025. **LlamaFirewall: An Open Source Guardrail System for Building Secure AI Agents**. *arXiv preprint arXiv:2505.03574*.
- Sarthak Choudhary, Divyam Anshumaan, Nils Palumbo, and Somesh Jha. 2025. **How Not to Detect Prompt Injections with an LLM**. *arXiv preprint arXiv:2507.05630*.
- Cloudflare. 2026. **Cloudflare Email Security**.
- Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2025. **Securing AI Agents with Information-Flow Control**. *arXiv preprint arXiv:2505.23643*.
- Dave Crocker, Tony Hansen, and Murray Kucherawy. 2011. **DomainKeys Identified Mail (DKIM) Signatures**. RFC 6376.
- Edoardo Debenedetti and 1 others. 2024. **Agent-Dojo: A Dynamic Environment to Evaluate Attacks and Defenses for LLM Agents**. *arXiv preprint arXiv:2406.13352*.
- Edoardo Debenedetti and 1 others. 2025. **Defeating Prompt Injections by Design**. *arXiv preprint arXiv:2503.18813*.
- deepset. 2023. **prompt-injections dataset**. Hugging Face. <https://huggingface.co/datasets/deepset/prompt-injections>.
- Marco Figueroa. 2025. **Phishing for Gemini: Prompt-Injection in Google Workspace Email Summaries**. Odin.ai (Mozilla GenAI bug bounty).
- fmops. 2024. **distilbert-prompt-injection**. Hugging Face model card. <https://huggingface.co/fmops/distilbert-prompt-injection>.
- Google Cloud. 2026. Model armor. Prompt-injection and jailbreak detection filter. <https://cloud.google.com/security/products/model-armor>, REST API v1, accessed 2026-07.
- Google Security Blog and Google DeepMind. 2025. **Mitigating prompt injection attacks with a layered defense strategy**.
- Google Workspace Admin Help. 2026. **Advanced phishing and malware protection**.
- Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. **Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection**. *arXiv preprint arXiv:2302.12173*.
- Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. **Defending Against Indirect Prompt Injection Attacks With Spotlighting**. *arXiv preprint arXiv:2403.14720*.
- Immersive Labs. 2025. **Weaponizing LLMs: Bypassing Email Security Products via Indirect Prompt Injection**.
- Dennis Jacob and 1 others. 2025. **PromptShield: Deployable detection for prompt injection attacks**. *arXiv preprint arXiv:2501.15145*.
- Scott Kitterman. 2014. **Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1**. RFC 7208.

- Murray Kucherawy. 2015. [Message Header Field for Indicating Message Authentication Status](#). RFC 7601.
- Murray Kucherawy and Elizabeth Zwicky. 2015. [Domain-based Message Authentication, Reporting, and Conformance \(DMARC\)](#). RFC 7489.
- Lakera. 2026. Lakera guard. Prompt-injection detection API, prompt_attack detector. <https://platform.lakera.ai>, endpoint /v2/guard/results, accessed 2026-07.
- Hao Li and 1 others. 2025. InjecGuard: Benchmarking and mitigating over-defense in prompt injection guardrail models. In *Proceedings of ACL*. ArXiv:2410.22770.
- Enze Liu and 1 others. 2023a. [Forward Pass: On the Security Implications of Email Forwarding Mechanism and Policy](#). *arXiv preprint arXiv:2302.07287*.
- Yi Liu and 1 others. 2023b. [Prompt Injection Attack against LLM-integrated Applications](#). *arXiv preprint arXiv:2306.05499*.
- Yinuo Liu, Ruohan Xu, Xilong Wang, Yuqi Jia, and Neil Zhenqiang Gong. 2025. WAIInjectBench: Benchmarking Prompt Injection Detections for Web Agents. *arXiv preprint arXiv:2510.01354*.
- Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In *USENIX Security Symposium*.
- Meta AI. 2025. Llama Prompt Guard 2 (86m) model card. Meta PurpleLlama. <https://huggingface.co/meta-llama/Llama-Prompt-Guard-2-86M>.
- Microsoft Learn. 2026a. [Safe Attachments in Microsoft Defender for Office 365](#).
- Microsoft Learn. 2026b. [Safe Links in Microsoft Defender for Office 365](#).
- Microsoft Security Response Center. 2025. [CVE-2025-32711: M365 Copilot Information Disclosure Vulnerability](#).
- Mimecast. 2026. [Advanced Email Security](#).
- Jose Nazario. 2026. Phishing corpus. <https://monkey.org/~jose/phishing/>, CC-BY-4.0.
- OWASP Cheat Sheet Series. 2025. [LLM Prompt Injection Prevention Cheat Sheet](#).
- OWASP GenAI Security Project. 2025. [LLM01:2025 Prompt Injection](#).
- Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David Wagner. 2024. Jatmo: Prompt Injection Defense by Task-Specific Finetuning. In *ESORICS*.
- ProtectAI. 2024. deberta-v3-base-prompt-injection-v2. Hugging Face model card. <https://huggingface.co/protectai/deberta-v3-base-prompt-injection-v2>.
- Kaiwen Shen and 1 others. 2020. [Weak Links in Authentication Chains: A Large-scale Analysis of Email Sender Spoofing Attacks](#). *arXiv preprint arXiv:2011.08420*.
- UK National Cyber Security Centre. 2024. [Prompt injection is not SQL injection](#).
- Ishaan Verma and Arsheya Yadav. 2025. [Decoding Latent Attack Surfaces in LLMs: Prompt Injection via HTML in Web Summarization](#). *arXiv preprint arXiv:2509.05831*.
- Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. [The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions](#). *arXiv preprint arXiv:2404.13208*.
- Shengye Wan, Cyrus Nikolaidis, Daniel Song, David Molnar, James Crnkovich, Jayson Grace, Manish Bhatt, Sahana Chennabasappa, Spencer Whitman, Stephanie Ding, Vlad Ionescu, Yue Li, and Joshua Saxe. 2024. [CYBERSECEVAL 3: Advancing the evaluation of cybersecurity risks and capabilities in large language models](#). *Preprint*, arXiv:2408.01605. Visual prompt-injection subset: <https://huggingface.co/datasets/facebook/cyberseceval3-visual-prompt-injection>, MIT, evaluation-only.
- Jiangrong Wu, Yuhong Nan, Jianliang Wu, Zitong Yao, and Zibin Zheng. 2025. [Control at Stake: Evaluating the Security Landscape of LLM-Driven Email Agents](#). *arXiv preprint arXiv:2507.02699*.
- Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. 2023. Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models. *arXiv preprint arXiv:2312.14197*.
- Qiusi Zhan and 1 others. 2024. [InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents](#). *arXiv preprint arXiv:2403.02691*.
- Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2025. [Agent Security Bench \(ASB\): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents](#). *arXiv preprint arXiv:2410.02644*.