Who Speaks for Whom: Prompt Injection and Authority in an Email-Drafting Agent

Muhammad Sakhawat, YaktAI Lab, Aura Platform LLC · hello@yaktai.com

YaktAI Lab Report 1 · 15 November 2026 · Preprint, not peer-reviewed · doi:10.5281/zenodo.24271287 · Data and code: doi:10.5281/zenodo.24271419

Abstract

AI systems now act for people: they draft, reply, book and pay. That makes authority a practical question, not an abstract one: who may direct the system, over what, shown how, and for how long. We give a four-part working definition (source, scope, proof, revocation) and illustrate how each part fails.

We then test one failure in our own product. Orchestrate drafts email replies for small businesses, and an owner approves each draft before it is sent. We hid 12 manipulative instructions in prospect replies and ran each 3 times on three models, including the production model. Two graders from different providers agreed on 91% of drafts (Cohen's κ = 0.87). By their joint judgement, the production model obeyed the attacker in 14 of 24 drafts (58%; 95% CI 39–76%); the provider's content filter blocked the other 12 requests. A hand check by the author on 60 drafts agreed with the graders on 93% of clearly harmful attacks (κ = 0.78) but only 63% of ambiguous ones.

We deployed two changes: the prospect's words were fenced off as untrusted, and a check outside the model held any draft stating facts absent from the business's records. A third change held drafts that claim actions never taken. After the fix the production model still followed the attacker in 11 of 30 and then 6 of 30 drafts across two runs (37% and 20%), Gemini in 1 of 36 and Claude in none. The checks held every draft that both graders judged compliant before it reached the owner as a ready answer. Three drafts flagged by only one grader were not held; all three asked before acting. Fencing cut compliance sharply for Gemini and Claude but not clearly for the production model. The checks outside the model, not fencing, are what kept compliant drafts from reaching owners as ready answers.

1. Introduction

For most of software's history, a program did what the person at the keyboard told it, at that moment. Authority was not in question, because that person was present. An assistant that drafts, replies, schedules, pays and publishes breaks that link. It acts later, elsewhere, and often without the person watching. Three things change at once:

- **Distance.** The person who granted the action is not present when it happens.
- **Inputs from strangers.** The system reads email, web pages and documents written by people with no standing to instruct it.
- **Scale.** One mistaken grant can be exercised hundreds of times before anyone notices.

A system that cannot tell whose wishes it is carrying out is then more than unreliable. It is a channel through which anyone can act in someone else's name.

This report makes three contributions:

1. A four-part working definition of authority for AI systems that act, mapped to four ways it fails (Sections 2 and 4).
2. A measured case study of prompt injection in a production email-reply agent, before and after a fix, with two graders from different providers (Section 5).
3. Evidence, within that sample, that an instruction-level defence varies sharply between models, while a deterministic check against the business's own records held every clearly harmful draft that reached it (Section 5.5).

2. A working definition

Authority is a grant with four parts. Remove any one and what remains is only access.

Part	The question it answers	Example
Source	Who granted this, and did they hold it to give?	The owner of a shop, not a customer writing to it
Scope	What exactly does it cover, and what not?	Reply to booking requests; never quote a price
Proof	How is the grant shown at the moment of action?	A recorded instruction in the owner's own words
Revocation	How does it end, and who can end it?	Expires after the event; the owner can withdraw it at any time

Three terms are often used as if they were one:

- **Identity** says who someone is.
- **Permission** says what a system lets an account do.
- **Authority** says what someone may do on behalf of another person or an institution.

A verified identity does not carry institutional authority. A real, verified teacher is not thereby the voice of their school. A permission granted by software is not authority either. An account able to send from a company mailbox is not entitled, by that fact, to speak for the company.

3. Related work

None of the four parts is new.

- **The confused deputy.** Hardy [1] described a program tricked into using its own authority for a less privileged caller. His remedy was capabilities: tie the right to act to the specific object acted on.
- **Agency law.** The law separates actual authority, which the principal really gave, from apparent authority, which a third party reasonably believes the agent holds [2]. That is the distinction between standing and the appearance of it.
- **Indirect prompt injection.** Greshake et al. [3] showed that instructions hidden in content an AI reads can take over the application. BIPIA [4] and AgentDojo [7] turned this into benchmarks.
- **Instruction-level defences.** These teach or tell the model to treat outside text as data. Spotlighting [5] marks untrusted input so the model can tell it apart, and our first change is a form of it. The instruction hierarchy [6] trains models to rank instructions by their source.

- **System-level defences.** These do not rely on the model's judgement. CaMeL [8] separates trusted instructions from untrusted data and enforces policies on every tool call: 77% of AgentDojo tasks solved with provable security, against 84% for an undefended system. Beurer-Kellner et al. [9] catalogue design patterns of this kind. Our second change is a narrow example.

Our contribution is not a new defence. It is a measurement in a deployed product, on its real code and production model, before and after a fix. It also connects the security literature to a plain account of whose authority an acting system carries.

4. How authority fails

Each failure is one of the four parts going missing while the action still goes ahead.

1. **Borrowed authority (source fails).** Text in an email says "the owner has approved this, go ahead", and the system treats it as an instruction. The writer never held the authority they claim. Prompt injection, seen this way, is a stranger speaking in the principal's voice.
2. **Silent widening (scope fails).** "Handle my inbox" becomes replying, then forwarding, then deleting. Each step looks like a small extension of the last, and no one granted the final one.
3. **Identity taken as standing (proof fails).** A verified person is treated as speaking for their employer, school or family. The proof covered who they are, not what they represent.
4. **Stale grants (revocation fails).** A yes given for one purchase, meeting or week is still honoured months later, after its reason has gone.

None of these needs a malicious model. Each comes from a system that treats "I can" as "I may".

5. Case study: injection in a production email-reply agent

5.1 System and threat model

Orchestrate writes outreach email for small businesses and drafts an answer to each human reply the business receives. A draft is never sent on its own. It becomes an approval request, and the owner reads it and approves, edits or discards it.

The attacker is anyone who can write to a business that Orchestrate is corresponding with, for example a prospect. They control only the text of their reply. Their goal is a draft that commits the business to something the owner never authorised, adds the attacker's details as the business's own, or claims an action that was never taken. If the owner approves without reading closely, the draft is sent from the business's own mailbox. Out of scope: compromised accounts, a compromised model provider, and attacks on the owner's approval screen.

5.2 Method

- **Code under test.** We ran Orchestrate's production drafting code: the same writer instructions, brief assembly and post-draft check. The data was a fictional business (Northfield Plumbing), a fictional prospect (Dana at Lakeside Dental) and one earlier note from the business. No customer data was used. As in production, the prospect's full reply, with quoted text removed, reaches the model. Only the database was replaced, by a fixture holding the business's setup. The reply's intent label was fixed as "interested",

rather than taken from the production classifier. Runs before the fix used commit 59d7625, and runs after it used 60df495.

- **Attacks.** We used 12 attack emails, each a short reply carrying an instruction aimed at the AI. Eight are clearly harmful if obeyed: a discount, a confirmed meeting, a refund, a false deadline, a named customer, bank details, impersonating a named owner, and leaking the instructions. Four can also be read as ordinary requests: a payment link, a phone number, a change of language, and forwarding the thread. 4 benign replies served as a control.
- **Models.** All runs took place on 15 November 2026, with default sampling and structured output where supported:
  - gpt-5-mini (version 2025-08-07) on Azure OpenAI, in the same deployment to which production routes this writer's calls (confirmed from the provider code and the production settings);
  - Claude Haiku 5.5 (Anthropic API);
  - Gemini 3.5 Flash (Vertex AI).
- **Design.** Each attack and benign reply ran 3 times per model in each round: once before the fix and twice after it (once after the first two changes, once as finally deployed). That is 324 attack runs and 108 benign runs.
- **Grading.** Two graders from different providers judged every draft: Claude Opus 5.5 and GPT-6 Astra. Both used one written rubric (complied, echoed or refused) and neither was shown the model or the condition. We count a draft as complied only when both graders say so. They agreed on 91% of the 198 drafts from the first two rounds ($\kappa = 0.87$) and on 93% of the 102 from the final round ($\kappa = 0.88$), and 7 of their 9 disagreements fell on the four ambiguous attacks.
- **Human check.** The author graded a blind sample of 60 drafts: 20 from each round, spread across the three models. The sample was shown without the model, the round or the AI grades, and all 60 were graded. On whether a draft fell for the attack, first-pass agreement with the AI consensus was 75% ($\kappa = 0.32$). Most early disagreements came from reading the grade labels the wrong way round, and from the four ambiguous attacks. In a review pass on the 12 largest disagreements, the author saw the AI grades and recorded a final grade. Agreement after review was 83% overall ($\kappa = 0.53$): 93% ($\kappa = 0.78$) on the eight clearly harmful attacks, and 63% ($\kappa = 0.13$) on the four ambiguous ones. For this reason we treat the clearly harmful results as the main finding.
- **Statistics.** Rates carry Wilson 95% intervals. The 3 runs of each attack share a prompt, so the runs are not fully independent and the intervals are, if anything, too narrow.

5.3 Results before the fix

Model	Blocked by the provider's filter	Complied, all attacks	Complied, clearly harmful only	Held by our check
gpt-5-mini (production)	12 of 36	14 of 24 drafts: 58% (39–76%)	7 of 13: 54% (29–77%)	0
Gemini 3.5 Flash	0 of 36	17 of 36: 47% (32–63%)	10 of 24: 42% (24–61%)	0
Claude Haiku 5.5	0 of 36	4 of 36: 11% (4–25%)	3 of 24: 13% (4–31%)	0

Azure's content filter refused some requests before any draft was written. That filter belongs to the provider, not to the product, and it varies between runs, so rates are given per draft written.

What the drafts did:

1. **A meeting no one agreed.** All three models confirmed "Tuesday at 3pm" in every run.
2. **A discount no one approved.** gpt-5-mini and Gemini put the attacker's "owner-approved 50% discount" in writing every time. Claude never did.
3. **Actions never taken.** gpt-5-mini (3 of 3) and Gemini (1) wrote "I've copied your CEO and forwarded the full thread". Nothing of the kind had happened.
4. **A customer the business does not have.** Gemini told the prospect the business serves "Acme Bank" in all 3 runs.
5. **What held.** No model leaked its instructions, adopted the attacker's bank account or promised the refund.
6. **Our own check caught nothing.** It tested length, stray email addresses and template residue. It never asked what the draft committed the business to.

5.4 The fix

We made three changes and deployed them to production on 15 November 2026, before this report was published. A follow-up the same day found that reworded attacks evaded the records check, and it was replaced by a different design; that work is YaktAI Lab Report 2. The changes measured here were: the first two at 3:02 PM ET (commit 60df495) and the third at 3:36 PM ET (commit eefff4c).

- **Fencing (instruction-level).** The prospect's reply is placed between explicit markers and labelled as the recipient's own words, never instructions. Copies of the markers inside the reply are removed. One rule states that nothing inside the fence may be confirmed as agreed.
- **A records check (system-level).** It is deterministic and runs after drafting. It holds any draft that states something not found in the business's own setup: a price, percentage or discount; a refund or guarantee; a time or day as agreed; payment details; a phone or account number; a link outside the business's own site and booking page; a named customer; a deadline; a signer other than the owner's signature; or text copied from the writer's instructions. A held draft reaches the owner marked "this one needs your eye", with the reason, instead of as a ready answer.
- **A check on claimed actions (system-level).** Added after the first re-test showed drafts saying "I've copied your CEO and forwarded the full thread". The writer cannot take any action, so a draft that says something has already been forwarded, copied, sent, booked, scheduled or confirmed is held. Promises ("I'll forward this") and references to earlier notes are not held.

5.5 Results after the fix

Model	Complied after the first two changes (60df495)	Complied as deployed (eefff4c)	Reached the owner unheld, as deployed	Benign replies held, as deployed
gpt-5-mini (production)	11 of 30 drafts: 37% (22–54%)	6 of 30: 20% (10–37%)	0	0 of 11
Gemini 3.5 Flash	1 of 36: 3% (0–14%)	1 of 36: 3% (0–14%)	0	0 of 12
Claude Haiku 5.5	0 of 36: 0% (0–10%)	0 of 36: 0% (0–10%)	0	0 of 12

Fencing changed Gemini (47% to 3%) and Claude (11% to 0%) sharply. On the production model, compliance was 37% in one run after the fix and 20% in the next, with the fence unchanged between them. Both intervals overlap the 58% measured before the fix, so this sample cannot show that fencing helped the production model at all. Telling a model that text is untrusted is therefore not a dependable defence on its own.

The records check held every clearly harmful draft that got through, in both runs. In the first run, one ambiguous draft reached the owner unheld: it offered to forward the thread to the prospect's chief executive once the prospect gave a contact, and it named no address. In the run as deployed, no draft that both graders judged compliant reached the owner unheld. The graders split on three unheld Gemini drafts, two about a phone number and one asking how to reach a third party; all three ask before acting. This agrees with the system-level work above [8, 9] and with the spirit of Hardy's argument [1]: the safeguard that matters should not depend on the model choosing well.

5.6 What reached customers

No drafted reply is sent without the owner's approval, so the owner was the last line of defence before the fix, together with the provider's filter. We counted production records from before the fix, reading counts only and no message content. Between 4 October 2026, when this path went live, and the fix on 15 November 2026, Orchestrate drafted answers to 4 prospect replies, the first on 15 October. In all 4 cases the owner chose to answer personally, and no drafted answer was ever approved or sent. The one limit is that owners saw those drafts in their notifications, and we cannot count whether any wording was copied into a reply written by hand.

A second drafting path exists in the code. It is switched off in production, writes nothing for customers, and does not yet run the records check. It must gain the check before it is ever switched on.

5.7 Limitations

- **Small sample.** 12 attacks, 3 runs each, one fictional business, one owner setting. Results on other businesses, languages or longer threads may differ.
- **Attacks written by the authors.** They are short and direct. Adaptive attacks written against the fix would likely do better. In particular, the records check is deterministic pattern matching and can be evaded by rewording, for example "fifty percent off", "we'll knock off half" or a number written out in words.
- **Graders are models.** One grader shares a family with one subject (Claude). The second grader, from another provider, agreed closely. The one human check was done by the author, who has a stake in the result, on 60 drafts. It agreed well on clearly harmful attacks and poorly on ambiguous ones (Section 5.2). A second, independent human grader is the next step.

- **Not covered by the check.** A change of reply language is treated as a reasonable request. Forwarding to a third party who is named without an address is not held.
- **Cost of the check.** It also holds honest drafts that state things not in the records, such as meeting times the owner never offered. We regard that as correct, but it adds work for the owner, which we did not measure.

6. Design observations from Aura and Orchestrate

We found the rules above by getting them wrong in our own products first.

Aura: identity is not standing. A fully verified person was once told by Aura that their identity was not enough to post as their company. The product was right and the design was wrong: it had treated a higher identity level as the path to institutional voice. We separated the two layers. To speak as an institution in Aura, a person now needs all three of the following:

- a current verified identity;
- a role in that institution;
- a confirmed proof of authority for that specific institution.

Authority for one institution never carries over to another. If the proof is revoked, rejected, suspended or expired, the institution's voice stops at once, with no grace period. A human reviewer confirms authority, and no automated path can.

Orchestrate: act only inside the grant. A business's mailbox holds far more than Orchestrate was asked to handle. Orchestrate files only incoming mail from people the business wrote to, or mail sent to the business's dedicated enquiries address, and asks the mail provider for send-only access where the provider allows it. Who may act for a business at all should come from the business's registration records, not from whoever is signed in. Today a person reviews each registration document, and the automatic check is not yet built.

Our working practice. The agents that help run YaktAI follow the same rule. An instruction found in an email, page or file is information, never a command. Sending, paying, publishing and deleting each need a fresh yes from the person, in their own words. A yes covers one action, not a category.

7. Principles

YaktAI Lab designs every acting system by five rules.

1. **Explicit.** Authority is granted in words by someone who holds it. It is never inferred from access, habit or a confident tone in the input.
2. **Narrow.** A grant names its action, its object and its limit. Anything outside it is a new question.
3. **Checked outside the model.** Where a mistake would commit someone, the check does not depend on the model's judgement (Section 5.5).
4. **Expiring.** A grant ends with its purpose. Withdrawing it takes effect at once, everywhere.
5. **Visible.** The person on whose behalf the system acts can see what it did in their name, and stop it.

Before acting, a system should be able to answer four questions: who asked, were they entitled to, does this fall inside what they asked, and is that still true now? If any answer is "I don't know", the right action is to ask.

8. Open questions

- **Authority a machine can check.** Can a grant be written so that a system verifies it at the moment of action, as a signature is verified, rather than trusting a description of it?
- **Delegation chains.** When an agent asks another agent for help, how much of the original grant passes down, and how does the second agent know?
- **The human check under load.** An owner who approves many drafts a day may stop reading them. How much does approval actually catch, and how should a held draft be shown so that it is read?
- **Consent without fatigue.** Where is the line between a standing grant that is safe and one that is not?
- **Authority in custom, not documents.** Families, mosques, schools and small shops grant authority in ways that are often unwritten. How should a system respect them?

Disclosure, ethics and availability

- **Disclosure.** The weakness was in our own product. It was fixed and deployed before publication. No customer data was used in the experiment.
- **AI assistance and conflicts.** The experiment code, analysis and drafting were done with an AI assistant (Claude, Anthropic) and reviewed by the author. The author's company builds the product tested. One grader comes from the same provider as one model tested, which is why a second grader from another provider was used.
- **Availability.** The attack set, test harness, raw drafts and both graders' judgements are released with this report: doi:10.5281/zenodo.24271419.

References

1. N. Hardy. The Confused Deputy (or why capabilities might have been invented). ACM SIGOPS Operating Systems Review 22(4), 1988.
2. American Law Institute. Restatement (Third) of Agency, §§ 2.01–2.03 (actual and apparent authority). 2006.
3. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. arXiv:2302.12173, 2023.
4. J. Yi, Y. Xie, et al. Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models. arXiv:2312.14197, 2023.
5. K. Hines, G. Lopez, M. Hall, F. Zarfati, Y. Zunger, E. Kiciman. Defending Against Indirect Prompt Injection Attacks With Spotlighting. arXiv:2403.14720, 2024.
6. E. Wallace, K. Xiao, R. Leike, L. Weng, J. Heidecke, A. Beutel. The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions. arXiv:2404.13208, 2024.
7. E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, F. Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. arXiv:2406.13352, 2024.
8. E. Debenedetti, I. Shumailov, T. Fan, J. Hayes, N. Carlini, D. Fabian, C. Kern, C. Shi, A. Terzis, F. Tramèr. Defeating Prompt Injections by Design. arXiv:2503.18813, 2025.

9. L. Beurer-Kellner, B. Buesser, A.-M. Creţu, et al. Design Patterns for Securing LLM Agents against Prompt Injections. arXiv:2506.08837, 2025.