
SpecShield: A Structural Trusted Monitor for Tool-Using AI Agents

Nhi Dinh · University of Illinois Urbana - Champaign

With Apart Research

Abstract

Most LLM safety systems moderate what a model says, but tool-using agents must also be evaluated by what they do. Agents that read untrusted emails, documents, or spreadsheets are vulnerable to prompt injection: a malicious instruction can cause the agent to leak private data, expose credentials, or perform unsafe financial actions through a tool call that looks benign.

We study this risk in the AgentDojo email/spreadsheet setting and build a structural trusted monitor for agent actions. Instead of asking another LLM to judge action text, our monitor checks typed tool fields, dataflow taint, external recipients, private-data flow, and whether any approval came from a trusted user. It enforces one invariant: untrusted content may not trigger money-moving, credential-exposing, or PRIVATE-to-external actions without trusted user approval.

On AgentDojo, the structural monitor blocks all 17 relevant attacks while allowing all 49 benign actions, assuming correct provenance. Because provenance is the key assumption, we corrupt it and measure safety under noise: as provenance accuracy falls from 100% to 50%, safety decreases from 100% to about 70% but remains useful. We also find that structural and LLM monitors fail differently, making a gated hybrid more robust.

The main takeaway is that structural authorization does not solve prompt injection, but makes agent safety measurable as a function of provenance quality.

1. Introduction

Most deployed “AI safety” is content moderation: a check on what a model *says*. But as LLMs become agents that call tools, a less-guarded layer opens up: what they *do*. A courteously-worded email can still forward customer records; a benign-looking tool call can still wire money. The danger is in the **action**, not the words, so it sails through every content filter. This paper is about that layer: *action authorization*.

The defining risk is not whether a model can *detect* malicious text, but whether the system can **prevent a dangerous action even when detection fails**. This is the **AI-control** problem, distinct from the content-alignment problem most LLM safety addresses. The standard recipe places a *trusted monitor* between agent and tools. Most work uses an **LLM monitor** that reads and scores the action; that design is jailbreakable (the attacker controls the text), costs a model call per action, and degrades on untuned languages.

We ask whether a **deterministic, structural** monitor can do the job in the email/spreadsheet setting. Instead of reading the payload, it inspects the typed action: *did PRIVATE data flow into this call (taint)? where did the “approval” come from (provenance)? is the destination on an allow-list derived from the user’s own tasks?* These checks are language-agnostic, auditable, and model-cost free, but only as good as the provenance signal they consume. We make that dependence the object of study. We motivate the setting with a regionally grounded threat model (BEC/phishing fraud in Vietnam, where edge agents are common [7, 9]), but the quantitative claims rest on AgentDojo [3].

Contributions: (1) a structural trusted monitor (taint + provenance + allow-list, deny-overrides) framed as *invariant enforcement*, a falsifiable claim, not a detector; (2) the **provenance-sensitivity curve**: structural safety is a graceful function of provenance accuracy (100%→~70% as accuracy falls 100%→50%), quantifying that structural authorization *relocates* prompt injection to provenance rather than solving it; (3) a **disjoint-failure analysis** under an optimizing attacker showing the structural and LLM monitors have non-overlapping blind spots, so a gated hybrid holds across the attacker-effort frontier; the attacker also found and closed a real web-sink hole.

2. Related Work

Information-flow control. *Fides* is the closest prior art: a deterministic taint-tracking planner that stops all AgentDojo prompt injection *given good labels* [2]. **CaMeL** constrains what injected content can trigger via capabilities/provenance [4]. **SPECSHIELD** differs in three ways: it is **hybrid** (structural + optional LLM joined by a gated controller, not pure IFC); it **measures the provenance dependence** that pure-IFC results take as given; and it studies an **optimizing attacker** against that dependence. It is complementary to *Fides*: *Fides* shows IFC suffices with good labels; we ask what the safety/usefulness curve looks like as label quality degrades.

Monitors. *AgentWatcher* applies rules via a monitor LLM over causal context [1]. The **OWASP Top 10 for LLM Applications** lists prompt injection and excessive agency as primary risks [6]. **OpenAI (2026)** argues defenses should constrain the *impact* of manipulation, not only detect malicious inputs; we adopt that stance [5]. **AgentDojo** is our substrate [3].

When to use it. Prefer a structural monitor when you can produce even a noisy provenance signal, need an auditable zero-cost language-agnostic guarantee, or deploy small/edge models. Add an LLM monitor when provenance is unavailable or the attack is purely semantic. The gated hybrid pays for the LLM only on cases the structural layer cannot settle.

3. Methods

Data model. Each tool call is a typed `Action(kind, args, tainted_labels, claims, text)`. `tainted_labels` records dataflow (did PRIVATE data flow in?); `claims["approval"]` records *provenance* (USER_TURN = trusted user, INJECTED_DOC = untrusted content); `text` is the payload only fragile baselines read. An `AgentState` carries an allow-list of recipients/IBANs and the action kinds the user confirmed.

Predicates and policy. Five language-agnostic predicates inspect the typed fields: recipient-on-allow-list, no PRIVATE→external dataflow, trusted approval provenance, non-destructive shell, and (added during red-teaming, §4) a destination backstop for web sinks. Each returns True/False/ABSTAIN; a deny-overrides policy aggregates them into ALLOW / ASK_USER / REVIEW / BLOCK (severity-max). **The ground-truth attack/benign label is never read by the verdict;** only structural signals are.

The provenance signal (the load-bearing assumption). On AgentDojo we derive each action from a task’s ground-truth tool calls: benign actions are tagged user-approved, injection actions are not, and PRIVATE taint is set when a read precedes the sink. This assumes a system that can distinguish user-requested from injected actions (e.g. a CaMeL/Fides-style tracker [2, 4]). §4 measures sensitivity to it; §5 states how every claim changes if it fails.

Monitors compared. `prompt_only` (no enforcement, floor); `rule_based` (English keyword + recipient rules); `llm_judge` (a model scores action text); `structural` (the policy engine); `hybrid` (`max(structural, llm)`); `gated` (trust the structural verdict when decisive, call the LLM only on the uncertain middle). The LLM scorer is averaged over $k = 5$ and cached by payload text, so a covert variant stripping only taint scores identically to its overt twin.

Attacker. One fixed attacker, run identically against every monitor, with declared affordances: it knows the rules, may **launder taint** and **sanitize cover text**, but **cannot** forge the user-confirmation record or send to an allow-listed recipient. We evaluate a fixed-effort escalation (L0 overt → L1 launder → L2 launder+sanitize) and an **optimizing white-box** attacker that searches the full evasion powerset.

Benchmark. AgentDojo live (v1.2.1, four suites) covering egress and consequential non-egress sinks (delete, credential change, payment modify, reservation, access change, calendar): **75 actions: 26 attack, 49 benign** (57 pure-read tasks skipped). A 6-scenario Vietnam set (EN+VI) and a bilingual BEC set (EN/VI/code-switched) probe the language axis; the Vietnamese/code-switched text was authored and vetted by a native speaker (the author) for fluency and semantic equivalence. We report **95% Wilson CIs** throughout. Everything runs offline on the bundled sample (stdlib only); live/real-model rows require `pip install agentdojo` (Python ≥ 3.10) and `ANTHROPIC_API_KEY`.

What did not work. (i) A block-list of destructive shell commands, defeated by `find -delete`;

allow-list commands instead. (ii) The “LLM degrades on Vietnamese” hypothesis, false at the frontier (§4). (iii) The first design lacked a recipient backstop on web sinks; the optimizing attacker exploited it, and we added rule `no-untrusted-destination`.

4. Results

4.1 The invariant holds, conditional on provenance

Metric	Value (95% Wilson CI)
Enforcement (relevant attacks prevented)	17/17 = 100% [82–100%]
Usefulness cost (benign blocked/escalated)	0/49 = 0% [0–7%]

A hard guarantee at $\sim$ zero cost, but provenance-conditional: with the user-vs-injected signal removed, structural usefulness collapses to $\sim$ 5%. Read it as “given provenance of quality q , safety is $f(q)$,” not as an unconditional number.

4.1.1 The provenance-sensitivity curve: the central result

Flipping the provenance signal on a fraction of actions (30 seeded trials/level, structural-only, fully offline):

Provenance accuracy	Safety	Usefulness
100%	100% [100–100]	100% [100–100]
90%	93% [91–95]	92% [91–94]
80%	89% [87–91]	86% [84–87]
70%	83% [80–85]	76% [73–78]
60%	74% [71–77]	70% [68–72]
50%	70% [67–73]	63% [60–65]

Degradation is **graceful, not catastrophic** (safety falls only to $\sim$ 70% at coin-flip provenance), and safety stays **above usefulness** throughout because the monitor is multi-signal: taint and recipient-allow-list rules backstop safety when the approval signal is wrong. The headline is a function, $safety = f(provenance\ accuracy)$, with tight CIs (pooled 30×75 decisions), reproducible with no API key.

4.2 No monitor dominates

Four-way comparison at $T = 0.5$ (real model, Anthropic, $k = 5$, 75 actions):

Monitor	Safety	Usefulness	Adaptive (covert)	LLM cost
structural	100%	100%	92%	0%
llm	69%	82%	69%	100%
hybrid (max)	100%	82%	100%	100%
gated (defer-to-trusted)	85%	100%	77%	12%

Adding the LLM is not free: the hybrid’s usefulness drops to 82% from LLM false positives. The gated controller calls the LLM on only **9/75 (12%)** of actions. **Recommendation:** *gated* as the practical default; *hybrid* when adaptive attackers are expected; *structural* alone as the auditable zero-cost baseline, given a provenance signal.

4.3 Disjoint failures → the hybrid holds across the frontier

Figure 1. Safety vs. attacker effort (one fixed attacker, all monitors, $T = 0.5$), *after* the web-sink fix. Structural holds across the whole frontier; the LLM collapses as cover text is sanitized (L2). Blind spots are disjoint, so the hybrid stays at 100%.

	L0 overt	L1 launder	L2 +sanitize
structural	100	100	100
llm	62	62	42
hybrid	100	100	100
gated	81	81	69

Pre-fix the structural row read $100 \rightarrow 92 \rightarrow 92$; the optimizing attacker exposed that residual and the fix closed it.

Optimizing (white-box) attacker (safety, 95% CI). The fix targets the structural layer; llm/gated rows are $\sim$ unchanged ($k = 5$ sampling variance):

Monitor	Before web-sink fix	After fix
structural	92% [76–98%]	100% [87–100%]
llm	35% [19–54%]	35% [19–54%]
hybrid	92% [76–98%]	100% [87–100%]
gated	65% [46–81%]	65% [46–81%]

The optimizer found that $\sim 8\%$ of attacks hit taint-only sinks (e.g. `web_request`) that lacked the recipient backstop used for email, defeating both monitors at once. The added destination backstop (`no-untrusted-destination`) recovers structural and hybrid to 100% at **zero usefulness cost** (benign hosts are allow-listed). This is the intended use of an optimizing attacker: it exposed what the fixed-transform test hid and motivated a verified defense.

4.4 The Vietnamese/code-switched language gap is a confirmed null

Bilingual BEC, **16 attacks** $\times$ **EN/VI/code-switched**, smaller model (Claude Haiku), $T = 0.5$:

Language	Structural	LLM (Haiku, 95% CI)
English	100%	88% [64–97]
Vietnamese	100%	94% [72–99]
code-switched	100%	100% [81–100]

Both a frontier model (Opus, prior run) and a smaller model (Haiku) show **no Vietnamese/code-switched gap**; the trend runs *opposite* the hypothesis ($EN \leq VI \leq CS$). So “text-based LLM monitoring degrades on Vietnamese” is **not supported**; the LLM’s real failure axis is adversarial text-sanitization (§4.3, L2), not language. Structural is language-agnostic *by construction* (100% across all three, since it reads typed fields). A 3B edge model is the one untested regime, but the prior is strongly null, so we rest no headline on the language gap. Vietnam grounds the threat model; it carries no quantitative claim.

4.5 Red-team: honest limits

Hand-crafted attacks leak **2 of 3**: taint-laundering to an allow-listed alias (taint must propagate across steps, not per-call), `find -delete` obfuscation (allow-list commands), and mislabeled-PUBLIC sensitive data (data labels need provenance too). These are the monitor’s real boundaries, not hidden failures.

5. Discussion and Limitations

Implications. Structural authorization converts agent safety into a measurable function of provenance quality, making the guarantee falsifiable and the residual explicit: you can state exactly what an attacker must do (launder taint *and* sanitize text *and* reach a backstop-free sink) and exactly where an LLM layer pays for itself. The disjoint-failure result is the practical payoff: a cheap gated hybrid inherits the structural monitor’s auditability and the LLM’s semantic coverage at $\sim 12\%$ LLM calls.

Limitations.

- *Provenance is assumed, not built.* If the system cannot distinguish user-requested from injected actions, every structural number degrades (usefulness toward $\sim 5\%$). Building/measuring a real noisy provenance tracker is the most important open problem.
- *Small N.* 75 AgentDojo actions, 6 bilingual attacks; CIs are wide. Results are directional.
- *Live/real-model rows require external dependencies.* AgentDojo + API key; the offline artifact verifies the engine, policy math, red-team, and a 10-action sample.
- *Scope.* Single-step authorization in email/spreadsheet/banking/slack/travel suites; multi-step taint propagation, command allow-listing, and document/Zalo/Teams agents are out of scope. Deepfake *detection* is out of scope because claimed authority is insufficient without trusted

approval.

Future work. (1) Derive the provenance signal from trajectory structure and measure its in-the-wild accuracy. (2) Run a 3B open model (Qwen/Llama via Ollama) to locate any edge-model language gap. (3) Propagate taint across multi-step trajectories; allow-list commands; add aggregate monitoring for sub-threshold fraud. (4) A Lean/SMT backend for the decidable predicates.

6. Conclusion

We built a structural trusted monitor that authorizes tool actions from typed fields, taint, and approval provenance rather than payload text, evaluated as invariant enforcement on AgentDojo. It enforces the invariant on every relevant attack at near-zero cost, but only given a provenance signal. That dependence is our main finding and one prior IFC results take for granted. Under an optimizing attacker the structural and LLM monitors fail in disjoint ways, so a cheap gated hybrid holds where each alone breaks. Structural authorization does not solve prompt injection so much as *relocate* it to a measurable provenance problem, and quantifying that relocation is the more honest and useful contribution.

Code and Data

- **Code repository:** <https://github.com/nhidinh2/specshield>.

References

- [1] AgentWatcher Authors. AgentWatcher: A Rule-based Prompt Injection Monitor, 2026. URL <https://github.com/wang-yanting/AgentWatcher>.
- [2] Manuel Costa, Boris Köpf, et al. Securing AI Agents with Information-Flow Control, 2025. URL <https://github.com/microsoft/fides>. Fides.
- [3] Edoardo Debenedetti et al. AgentDojo: A Dynamic Environment to Evaluate Attacks and Defenses for LLM Agents, 2025.
- [4] Edoardo Debenedetti et al. Defeating Prompt Injections by Design, 2025. URL <https://github.com/google-research/camel-prompt-injection>. CaMeL.
- [5] OpenAI. Designing AI Agents to Resist Prompt Injection, 2026. URL to be verified before submission.
- [6] OWASP Foundation. OWASP Top 10 for LLM Applications, 2026. URL <https://owasp.org/www-project-top-10-for-large-language-model-applications/>.
- [7] Prime Minister of Vietnam. National Strategy on R&D and Application of AI to 2030, Decision No. 127/QD-TTg, 2021.
- [8] Vietnam National Assembly. Law on Cybersecurity No. 116/2025/QH15, 2025. Effective 1 July 2026.

[9] VNETWORK. Vietnam Cybersecurity Threat Report 2026, 2026. See CITATIONS.md for figures and URLs.

LLM Usage Statement

I used an LLM assistant to consolidate documentation, draft prose, and review structure. All quantitative claims were produced by the project's own deterministic/CLI code and independently re-run; the report distinguishes offline-verifiable results from those requiring the live AgentDojo harness and an API key. The final manuscript was reviewed and edited by the author. Citation IDs and external figures flagged in References/Appendix require verification against primary sources prior to submission.