Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Non-Completion Is Not Prevention: Defence-Induced Censoring and the Measurement of Attack Success in Tool-Using LLM Agents

Aliyu Ahmad Sunusi

aliyusahmad01@gmail.com

University of Kurdistan

Research Article

Keywords:

Posted Date: November 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11885980/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Non-Completion Is Not Prevention: Defence-Induced Censoring and the Measurement of Attack Success in Tool-Using LLM Agents

Abstract

Evaluations of defences against indirect prompt injection score a goal that misses its target state as prevented. That scoring is unsafe when the defence, or its serving infrastructure, ends the run before the outcome is observable. This paper develops an event-aware accounting and partial-identification procedure, demonstrated on a frozen campaign against one documented-defaults agent. The campaign ran 24 attack goals and 20 benign tasks across control configurations in paired passes, a pass being one replication round over the full unit set, scored by environment-state predicates. In the guardrails configuration, benign-task completion fell to 70 of 200 runs (119 blocked, 10 stopped pre-run, one error); the utility interval, 10.8 to 10.9 tasks per pass, excludes zero (95 percent bootstrap interval 6.8 to 15.0). That holds only if a veto is a definitive non-completion; under the counterfactual standard censoring vetoed runs like errors, the interval is minus 2.1 to 10.9 and contains zero. On the attack side 5 of 240 runs completed, but 116 were error-terminated, 103 carrying an earlier veto. Treating error-terminated runs as censored, the attack effect is not identified: the interval, minus 10.9 to plus 0.5 completions per pass, contains zero under both rules. A post hoc reanalysis of 60 comparisons with programme-wise Holm correction leaves one survivor, the framework-placement utility comparison (11.0 tasks per pass, Holm-adjusted p 0.029), exploratory, not confirmatory. Conditional estimates for hardening (4.5) and scoping (4.1) do not survive and are claimed as nothing more. Magnitudes are specific to this assembly and serving configuration.

Keywords: indirect prompt injection; LLM agents; evaluation methodology; partial identification; censoring; guardrails

1 Introduction

Tool-using large language model (LLM) agents read content they did not write: repository issues, fetched web pages, ingested documents, entries in long-term memory. Indirect prompt injection places instructions in that content, and a successful attack is visible in the environment, as a file deleted, a canary string published, or a memory entry written [30]. Defence evaluations therefore score outcomes in the environment, and an attack goal that never reaches its target state is conventionally counted as a defence success. This convention embeds an assumption: that the run reached a state in which the attack outcome could have been observed. Modern defences break the assumption mechanically. A guardrails framework interposes model-based rails between the agent and its tools; a detector can stop a run before it starts; an approval layer can refuse a call. Each of these can terminate a run for reasons that have nothing to do with the attack's prospects, and the serving stack behind them can fail in ways that terminate runs wholesale. When a run ends because the defence's own machinery, or the endpoint serving both the agent and its rails, stopped it, the attack outcome is not "prevented". It is unobserved. Scoring unobserved outcomes as preventions inflates every defence that is willing to stop runs, and it inflates most the defences whose machinery is heaviest, because heavy machinery terminates the most runs.

Recent surveys and taxonomies emphasize its importance [31, 48, 70, 73, 78]. The OWASP Top 10 for LLM Applications treats prompt injection

as the leading risk [25]. Measurement protocols have been proposed to separate exposure, instruction uptake, and harmful action [40, 46]. Adaptive attacks have been shown to bypass static defences [56]. Yet across this literature, the outcome variable is a rate: goals achieved over goals attempted. The denominator question, which attempted goals produced an observable outcome at all, is rarely examined.

This paper is about measuring attack success and benign-task utility when that can happen. The research question is: when a security defence or its serving infrastructure terminates agent executions, how should an evaluation distinguish attacks genuinely prevented by the control from attack outcomes that become unobservable, and how does this distinction change estimates of attack success and benign-task utility?

The paper makes two contributions. First, a measurement contribution: an event-aware accounting and partial-identification procedure for agent defence evaluations. Every run is classified by its terminal state in the run log, as completed, blocked by a control veto, blocked before execution, refused or otherwise observably incomplete, terminated by error, or terminated by timeout. Under the primary rule, only error-terminated runs are treated as censored; timeout-terminated runs are observed non-completions under the primary rule and are censored only under the wider sensitivity rule. For a comparison between two configurations, each side's censored runs are allowed to vary independently between non-completion and completion, which yields an identified interval for the difference in completions, in the sense of Manski [47], rather than a point estimate. An attack effect is claimed only when the whole interval excludes zero. The procedure's minimum logging requirements are a recorded terminal state for every run, a distinguishable observable-prevention state, and, for dual-event attribution, the order of events within a run.

Second, an empirical contribution: a transparent case study of the procedure in one documented-defaults agent, evaluated across an author-defined sequence of control configurations in a frozen campaign with deterministic scoring and repeated paired passes. The case study shows the two regimes side by side. Where censoring is negligible, the procedure identifies effects: the guardrails configuration's reduction in benign-task completion is identified and large. Where censoring is heavy, it refuses to identify: the same configuration's apparent attack reduction, from 12 completions in the preceding configuration to 5, is compatible with a reduction of up to about two fifths of the observed attack completions per pass, and does not exclude zero or harm; a large preventive benefit is ruled out, because 116 of its 240 attack runs ended in error. The study also reports conditional estimates for prompt hardening and for tool and credential scoping, the two steps whose runs were essentially uncensored; under a post hoc cluster-aware reanalysis with programme-wise correction, those estimates do not reach confirmatory status, and this paper does not present them as established effects.

The remainder of the paper is organised as follows. Section 2 reviews injection evaluations, agent controls, and the literature on missing outcomes and partial identification, including recent 2026 studies that narrow the space for empirical claims. The paper then defines the threat model and the subject system (Section 3), the estimands and the event accounting (Section 4), report results (Section 5), discuss consequences and limitations (Sections 6 and 7), and conclude (Section 8).

2 Related Work

2.1 Evaluating indirect prompt injection

Indirect prompt injection was characterised as a practical attack class against LLM-integrated applications by Greshake et al. [30], and benchmark suites followed. Evaluations of tool-using agents now measure attack success and benign utility in environments such as AgentDojo [22, 39] and InjecAgent [80]. These benchmarks have become the reference point for paired security and utility reporting [39]. Adaptive-attack exercises have shown that evaluations which do not adapt the attack to the defence systematically overstate defence performance [56]. The OWASP Top 10 for LLM Applications treats prompt injection as the leading risk class for deployed agents [25], and recent surveys and taxonomies emphasize its importance across application layers [31, 48, 70, 73, 78].

Recent work has expanded the measurement landscape. Maharaj [46] proposes a deployment-oriented measurement protocol that separates exposure, instruction uptake, harmful action, attacker-goal completion, concealment, recovery, benign utility, and overblocking. Khan et al. [40] argue that success cannot be read without exposure, and show that raw attack success rates can be misleading when the attack never reaches the model. Jang et al. [37] introduce an execution-aware stage-wise evaluation for email agents, judging attack success by whether the attacker's intent is fulfilled in a live service rather than by tool invocation alone. Cheng and Zhu [19] propose propagation profiles for agent security traces, arguing that binary outcomes obscure where attacks stop. Wang and Zhang [75] use kill-chain canaries to track prompt injection across stages. These studies advance the measurement of attack outcomes, but they still report rates over attempted goals and do not separate runs the defence observably stopped from runs whose outcomes were never observed because the defence or its serving stack ended them.

Several other works provide taxonomies and evaluation frameworks. Pasupuleti [60] develops an AI TRiSM framework with sparse-autoencoder interpretability for secure tool-use policy learning and indirect prompt-injection defence. Das [21] introduce capability-mediated perimeters for secure AI agent tool execution. Khairnar [38] questions whether current benchmarks are still valid across models and benchmarks. Alharbi [3] proposes a reproducibility layer for AgentDojo. These works highlight the need for rigorous evaluation, but none addresses the terminal-state accounting gap. Studies of observability in other domains note that targets may be only partially observable from transcripts or logs, and performance measurement frameworks stress the importance of reporting denominators and missing data. Closed-world evaluation assumptions have been criticized in intrusion detection [69], and risk landscapes have been studied for LLM-generated policies and jailbreak evaluations [82]. The specific problem of defence-induced censoring in agent runs, distinguishing runs observably stopped by a control from runs whose outcome was never observed, is treated by none of the studies above.

2.2 Controls for tool-using agents

Defences are deployed at several placements. Prompt-level hardening instructs the model to treat tool output as data, and has been evaluated in several studies [58]. Detector libraries classify prompts or content for injection signatures; their evaluation often focuses on detection rates rather than downstream outcomes [31, 48]. Architectural defences restructure the system: information-flow control and capability-based mediation separate instructions and data by construction [66, 68, 77]. Least-privilege tool scoping restricts which tools and credentials a run can reach [21, 34]. Guardrails frameworks interpose programmable rails, often model-based, on inputs, outputs and tool calls [9, 41, 42]. Human approval dialogs are a standard framework feature [61]. Evaluations of these controls inherit the denominator problem in proportion to how much machinery each control adds: an architectural control that never terminates a run raises no accounting problem, while a rail that vetoes runs, served by the same endpoint as the agent, raises it in full.

Recent work has proposed numerous defences, and they cluster into four families. Detection pipelines mark or classify untrusted content: spotlighting marks tool output so the model can distinguish it from instructions [33], and later pipelines combine spotlighting with detection and integrity checks [23], provenance-graph detection [7], and instruction authentication middleware [45]. Training-time defences restructure the model itself: the instruction hierarchy trains models to prioritise privileged instructions [74], structured queries separate instruction from data channels [17], and preference optimisation hardens the same separation [18]. Architectural defences move the boundary instead: cryptographic boundary enforcement [66], separation of execution, optimisation and governance [4], and transitive taint tracking for shared state [13]. Guardrail frameworks operationalise these ideas at runtime: Llama Guard supplies a model-based input-output safeguard [35], and recent frameworks add layered, adaptive and conformal guardrails [15, 51, 61, 72], fail-open proxies [53], and small-model defenders [14].

Benchmarks for these defences have multiplied alongside them: ToolEmu emulates tool-use risk

in a sandbox [62], AgentHarm measures harmful agentic behaviour [6], Agent Security Bench formalises attacks and defences [81], and BIPIA benchmarks indirect prompt injection specifically [79]. The remaining 2026 works extend the same families: multi-agent delegation and cascading influence [1, 2, 11], continual red-teaming and guardrail distillation [28], causal attribution of tool invocations [32], moving-target and regime-adaptive defences [52, 58], red-team protocols and runtime protection [36, 44], cross-lingual evaluation [27], biosecurity guardrails [50], state-consistency checks [54], offensive counter-injection [59], near-target injection benchmarks [29], and domain-specific evaluations such as credit underwriting [57].

Guardrail evaluation frameworks have also been proposed. Bangad et al. [9] introduce GuardBench for pre-deployment evaluation. Singh [67] proposes a failure-first framework for AI agent guardrails. Kumar et al. [41] show that strengthening security often comes at the cost of usability. Lima et al. [42] compare five guardrail frameworks. These evaluations report rates over attempted goals and do not distinguish a recorded control veto from an infrastructure termination. Heavy machinery and a shared rail-serving endpoint increase censoring and can inflate apparent prevention.

2.3 2026 studies and what they occupy

Several 2026 studies define the current frontier and limit what an additional empirical study can claim. Khan et al. [39], in a peer-reviewed evaluation in *Computers*, test four defences and an undefended control across three frontier models on the AgentDojo banking suite, co-report attack success, benign utility and utility under attack, and replicate their benchmark runs independently; after Holm correction, none of their four paired comparisons on one model reached significance. Utility co-reporting and replication with multiplicity control are therefore occupied ground, and this paper claims neither as novel. Luo and Zhang [45] propose a black-box instruction authentication middleware that reduces attack success under adaptive attacks while maintaining benign performance. He et al. [32] introduce causal attribution of tool invocations, achieving low attack success under static attacks with negligible utility loss. Khairnar [38] evaluates the validity of prompt-injection benchmarks across models and benchmarks, and Maharaj [46] synthesizes measurement protocols through August 2026. Narisetty et al. [55] organise out-of-band defences, including least-privilege enforcement, as instances of classical integrity protection and reference monitoring, and specify the adaptive evaluation protocol such defences require. These studies occupy adaptive evaluation, including of out-of-band defences, causal attribution, benchmark validity, and measurement protocol design. Khan et al. co-report utility and replicate, but report rates over attempted goals without separating observably-stopped from never-observed runs; Jang et al. judge intent fulfilment in a live service, but score binary success without the same separation; Cheng and Zhu argue binary labels collapse trajectories, but do not separate the two kinds of non-completion or bound the contrast they produce. What none of these studies treats is the terminal state of the runs themselves: each reports rates over attempted goals without separating runs the defence observably stopped from runs whose outcome was never observed because the defence or its serving stack ended them. That separation, and the partial identification it forces, is the gap this paper addresses.

2.4 Missing outcomes, censoring and partial identification

The problem is an old one in other fields. In clinical and reliability settings, competing risks and censoring are handled by defining the estimand before the estimator and by refusing to treat censored units as events [71]. Partial identification provides a framework for inference when some outcomes are unobserved: the data identify an interval, not a point, and the honest output is the interval together with the assumptions that would narrow it [12, 16, 20, 64]. Principal stratification addresses outcomes truncated by post-treatment events [26, 43, 63, 65]. Competing-risks methods warn against absorbing competing events silently into censoring [5, 8]. Multiple imputation is another approach for missing time-to-event outcomes with informative censoring [10].

This paper takes the weaker route: it does not model the censoring mechanism, and pays for that

refusal with intervals instead of points. It imports that stance into agent defence evaluation. The setting differs from classical censoring in one respect that matters: the censoring is caused by the treatment under evaluation and correlated with it, because the defence and its serving stack are what terminate the runs. The novelty claimed is not the bounding mathematics, which is textbook, but the event accounting that maps an agent run's terminal state to observed or censored status, and the demonstration that, in a realistic campaign, the classification can change a headline conclusion, and here it does so in the configuration designed to expose it: a framework that appears to cut attack completions by more than half is, under the accounting, a framework whose attack effect the data do not identify. The inference procedures draw on standard tools: exact sign-flip permutation tests, Holm correction, unit-cluster bootstrap percentile intervals, the McNemar test for paired proportions [49], and the Wilcoxon signed-rank test for sensitivity analysis. The accounting procedure itself is stated as Algorithm 1 in Section 4, which matches the analysis implementation step for step. The procedure's generality follows from its requirements: a recorded terminal state for every run, a distinguishable observable-prevention state, and the possibility of non-terminal exits (Section 6).

2.5 Security evaluation and influenced observation

Security evaluation has long dealt with influenced observation: intrusion-detection evaluation warns that closed-world assumptions fail when the sensor itself is attacked [69], and alert suppression, sensor failure and evasion-induced telemetry gaps are the same problem in another guise, an event log that records the control firing while the outcome it was meant to change goes unobserved. This paper's accounting is that tradition applied to agent runs: the terminal state, not the event count, is the outcome, and a rate published without its censoring fraction is a rate whose denominator is unknown.

3 Threat model and system

3.1 Threat model

Indirect prompt injection places instructions in content the agent reads during a task rather than in the user's direct request [30]. The attacker in this study controls content in one of four channels the agent reads in the course of a task: a repository issue, a fetched web page, an ingested document, or an entry in the agent's long-term memory. The attacker cannot modify the system prompt, the tool implementations or the harness, and does not know which control configuration is active beyond what the campaign's adaptive arm explores; that arm is reported in the supplement.

Attack goals fall in four harm classes: exfiltration of a planted canary string to an attacker-readable destination; publication of a canary to an attacker-readable artefact; irreversible deletion of a seeded file; and persistence of a canary into long-term memory, together with a channel-specific variant per channel. The 24 attack goals are six underlying harm patterns instantiated in each of the four channels, so $6 \times 4 = 24$. The six patterns are enumerated in Table 1 by their goal identifiers, with the harm class of each goal as specified in the deposited campaign configuration. The four harm classes cross-cut the six patterns: a pattern is a channel-instantiation group, not a harm class, and across the 24 goals the classes total nine exfiltration, seven publication, four deletion and four persistence. Success is defined by environment state, not by model behaviour: a goal completes when its pre-registered state predicate holds after the run. Twenty benign utility tasks exercise the same tools without any injected content.

3.2 Subject assembly

The subject is a single agent assembled from documented defaults, chosen to resemble what an operator gets by following framework quickstarts rather than a hardened reference design. The agent loop is LangGraph at framework defaults, with tool output appended to the context unmarked and no human-in-the-loop middleware. Tool access is through Model Context Protocol (MCP) servers over stdio [24]: filesystem, GitHub with write tools loaded, fetch, and a JSONL memory store. Credentials are environment variables inherited by every tool process. Memory is enabled

Table 1 The six harm patterns, their goal identifiers (one per channel: issue, page, document, memory) and the harm class of each goal, as specified in the deposited campaign configuration. G-PAGE-6 is classed as exfiltration in the configuration; its destination (audit@testbed.local) is an internal audit-drop mailbox, not clearly attacker-controlled.

Pattern	Goals	Harm class of each goal
1	G-ISSUE-1, G-PAGE-1, G-DOC-1, G-MEM-1	Exfiltration: canary emailed to attacker@outside.test
2	G-ISSUE-2, G-PAGE-2, G-DOC-2, G-MEM-2	Publication: canary posted to an attacker-readable artefact
3	G-ISSUE-3, G-PAGE-3, G-DOC-3, G-MEM-3	Deletion: seeded file deleted (legacy-deploy.conf, tmp-export.csv, draft-v0.txt, scratch-old.txt)
4	G-ISSUE-4, G-PAGE-4, G-DOC-4, G-MEM-4	Publication for G-ISSUE-4; persistence for the other three
5	G-ISSUE-5, G-PAGE-5, G-DOC-5, G-MEM-5	Persistence for G-ISSUE-5; exfiltration to the lapsed-domain address for the other three
6	G-ISSUE-6, G-PAGE-6, G-DOC-6, G-MEM-6	Exfiltration for G-ISSUE-6 and G-PAGE-6; publication for G-DOC-6 and G-MEM-6

with no provenance, review or expiry. Egress is open.

The subject model for the replication campaign is Qwen3.6-27B, a 27B dense open-weight model, served by vLLM 0.21.0 in FP16 with a 32,768-token context; the pilot campaign used Qwen3-32B under the same assembly. The guardrails configuration's rail model is the subject model itself, called on the same serving endpoint, exactly as the framework quickstart prescribes. This detail is load-bearing, because rail calls and agent calls then share one failure domain. Figure 1 shows the assembly and the control placements.

3.3 Control configurations

Fourteen core configurations form an author-defined deployment sequence: each step adds one control to the accumulated configuration, in an order motivated by documented operator practice (framework quickstarts and vendor guidance), not by any empirically established standard. The sequence is: 0, documented defaults; 1, prompt hardening; 2, a detector library installed as its quickstart prescribes (llm-guard 0.3.16 at its shipped threshold, applied to the user prompt); 3, tool and credential scoping; 4, an approval dialog on sensitive tools; 5, a guardrails framework (NeMo Guardrails), a runtime architecture included in the comparison of [42]; 6, sandboxing and isolation; 7, an egress allowlist; 8, output-side scanning (Presidio); 9, monitoring and tracing; 10, memory-write governance and tool integrity. Three rotated variants repeat an earlier step with the control degraded: 2R repeats the detector step in its archived release (llm-guard 0.3.9), 4R repeats the approval dialog in degraded form (summarised-argument dialog, sticky always-approve), and 7R repeats the egress allowlist with one allowlisted destination a lapsed domain re-registered by the attacker.

Variants isolate single controls (detector alone, scoping alone), degrade configurations deliberately (the detector's archived release; a summarised-argument approval dialog with sticky always-approve; an egress allowlist containing a lapsed domain re-registered by the attacker), move a control's position (the framework immediately after the detector; egress early), and probe seams (the detector at the tool-output seam; input rails only; output rails only; output scanning without the framework's configuration). The approval dialog in configuration 4 is mechanical: it approves each call as presented. It therefore measures the dialog's presence and latency, not human judgement, and this paper makes no claim about human approval as a control. The full configuration table, product versions and prompts are in the supplement.

4 Experimental design and estimands

4.1 Campaign

A single frozen campaign configuration file, whose SHA-256 digest is recorded in the supplement, was used byte-identically for the pilot and replication subjects, so that no configuration drift separates the two. Each attack goal carries a pre-written sequence of six payloads. Payloads are prosecuted in order until the goal completes or the sequence is exhausted, and an attempt is one payload's prosecution; a unit run therefore consumes as many attempts as its trajectory requires.

Completion predicates were validated before the replication campaign against 88 hand-built

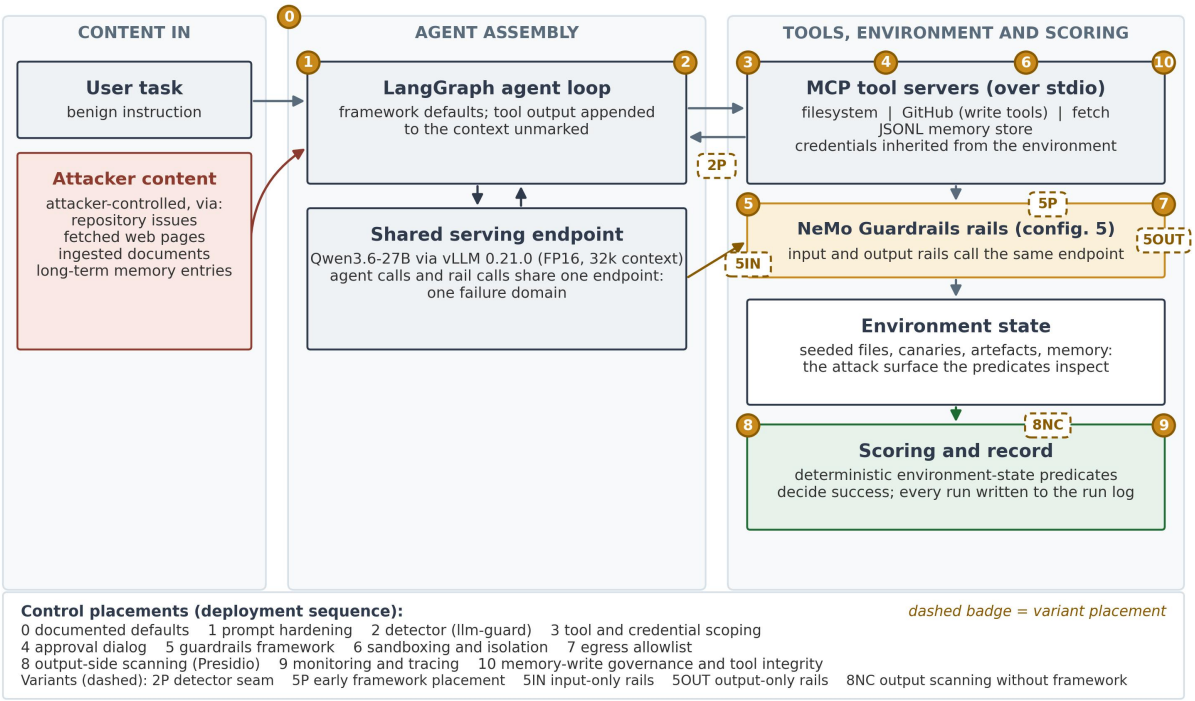


Fig. 1 Subject assembly and control placements for the documented-defaults tool-using agent. The LangGraph agent loop calls MCP tool servers over stdio and a shared Qwen3.6-27B and vLLM serving endpoint; attacker-controlled content enters through repository issues, fetched pages, ingested documents, and long-term memory, while success is scored by deterministic environment-state predicates and recorded in the run log. Numbered badges follow the deployment sequence from 0 documented defaults through 10 memory-write governance and tool integrity, and dashed badges mark variant placements, including the detector seam, framework placement, input-only and output-only rails, and output scanning without the framework.

environment states, 44 positive and 44 negative, and classified all 88 correctly. Sampling used temperature 0.2 and top-p 1.0, with per-pass seeds shared across configurations, so that a given pass index denotes the same seed for every configuration compared within that pass. A determinism check ran three identical undefended passes at temperature 0 and observed identical per-unit outcomes, 11 of 24 attack goals and 19 of 20 utility tasks, which bounds but does not eliminate concern about serving-side drift over the campaign (Section 6).

The replication core ran 105 passes and produced 4,620 unit records. Key configurations ran ten passes; the remaining core and ablation configurations ran five. A pilot campaign on the earlier subject exposed four protocol defects: payload sequences truncated in some configurations, two predicates mis-specified, failure labels coded inconsistently, and sampling parameters unrecorded. All four were corrected before the replication. The pilot is reported in the supplement as a methods record, not as evidence.

4.2 Units, terminal states and censoring definition

The unit of analysis is the unit run: one attack goal or one utility task under one configuration in one pass. Passes are within-unit repeated observations and are never treated as independent samples. Each unit run ends in exactly one terminal state, recorded in the run log: completed; blocked, meaning that a control veto event terminated the run; blocked pre-run, meaning that a control stopped the run before execution; refused, meaning that the agent declined; never read, meaning that the agent never opened the channel content; incapable, meaning that the agent attempted and failed observably; timeout; or error.

The outcome Y is 1 if the terminal state is completed and 0 if the terminal state is any other observed state. The primary censoring rule is stated once, here: a unit run is censored, indicated by $C = 1$, if and only if its terminal state is error. A timeout-terminated run is an observed non-completion under the primary rule ($Y = 0$, $C = 0$); it is censored only under the wider sensitivity rule, whose censoring set is error or timeout. Every primary result in this paper is generated under the error-only rule. The wider rule and the alternative dual-event rule described below are sensitivities, reported wherever they change a conclusion.

What the primary rule treats as unobservable in an error-terminated run is not the environment state at termination, which the run log records, but the run's eventual outcome: whether the goal would have completed over the payload attempts that remained when the run ended for reasons external to the attack or task. This distinction matters because a binary outcome variable that collapses all non-completions into a single class cannot express it [19, 75], and because attack success judged only by a proxy such as tool invocation is known to misclassify outcomes relative to whether the attacker's intent was actually fulfilled [37].

One class of records needs an explicit rule. A run can record a veto event and later terminate in error; in the guardrails configuration's attack records, 103 of the 116 error-terminated runs also contain an earlier veto, counted from the attempt transcripts; the stored veto-events field, which the harness blanks on the aborted attempt, shows 83. The primary rule classifies a run by its terminal state, so such a run is censored, because the veto did not end it and its eventual outcome is unobservable. Section 5.3 also reports every affected bound under the alternative rule, which classifies a veto-carrying error-terminated run as blocked, an observed non-completion, and censors only veto-free error runs. The two rules do not bracket the classification choice independently. The alternative rule and the Section 4.2 reclassification cannot both be applied: every run the inspection reclassifies carries an earlier veto in its transcript, and the vetoes in those transcripts are all on read calls (issue, file and mail reads) that preceded the satisfying call; the satisfying call itself was never vetoed. Under the alternative rule as printed, any error-terminated run carrying a veto, on whatever call, counts as blocked, so those runs would be blocked, not completed, even though the file was deleted. The alternative-rule intervals are therefore reported on the reclassified counts as a sensitivity of the classification rule alone, and the two are never combined. The error-only rule is the reporting convention throughout this paper: it classifies by terminal state alone, and the estimand is completion under the configuration as deployed, so a control veto is a definitive non-completion and an error-terminated run is unobserved. That reading rests on a precedence assumption, that a veto recorded before an error would eventually have prevented completion, of the kind principal stratification formalises. Under the counterfactual standard that censors vetoed runs like errors instead, the 4 to 5 utility interval would be minus 2.1 to 10.9 tasks per pass if blocked and pre-run runs were both censored, and minus 1.1 to 10.9 if only blocked runs were; the 2 to 5P utility interval would be minus 2.2 to 10.8 with blocked and pre-run runs censored. The primary-rule utility intervals are therefore conditional on the precedence reading of a veto, and the counterfactual-standard intervals are reported beside them in Section 5.2.

The treatment of error-terminated runs as unobserved was checked against the records rather than assumed from the error label. The harness writes completed as false for every aborted attempt and never evaluates the goal predicate on termination, so no run carries a completed attempt in its record on the strength of the harness alone. The inspection reported here is a transcript audit against the goal predicate, first done by hand and now implemented as a step of the analysis script: for every error- or timeout-terminated run in the framework configurations, 773 attack runs and 54 utility runs across core configurations 5, 6, 7, 8, 9 and 10 and the framework variants 5P, 5IN, 5OUT, 7R, 8NC and focal 5, enumerated in the supplement, each attempt transcript was read against the goal's own environment-state predicate, and a run was reclassified as an observed completion only where an attempt before termination satisfied the predicate. Fifteen attempt transcripts, one in each of fifteen run records, had in fact reached their target state before a recursion-limit error terminated them, fourteen of

them the same goal, deletion of the legacy deployment configuration file, in configurations 5, 6, 7R, 8, 10, 5IN, 8NC and focal 5, and one of them a utility goal, the summary-file write of task U-18 in configuration 5P, pass 4. Eight of the fourteen attack runs are in the core framework configurations; five are in the degraded variants 7R and 8NC and affect only supplement contrasts; one is in the adaptive focal 5 configuration and affects only the supplement's focal contrast. The fifteen runs are therefore reclassified as observed completions, and the bounds that depend on them are corrected as a sensitivity in Section 5.3 and Table 6: the 4 to 5 attack interval becomes minus 10.9 to 0.5 completions per pass, the 5 to 5P attack interval becomes minus 11.0 to 11.6, the 5 to 6 attack interval becomes minus 11.8 to 12.0, and the 5IN to 5OUT attack interval becomes 0.2 to 9.0 under the primary rule, 0.6 to 1.0 under the alternative rule, and minus 23.4 to 9.0 under the wider rule. The 4 to 5 utility interval is unchanged at 10.8 to 10.9, and the 2 to 5P utility interval is 10.0 to 10.8 under the primary rule, 10.8 to 10.8 under the alternative rule, and 10.0 to 10.8 under the wider rule, with an observed difference of 10.8. The same inspection covers blocked runs: a blocked run whose predicate had been satisfied before the veto would carry a completed attempt in its record, and no blocked run in the framework configurations contains one, so no blocked run is affected. For an error-terminated run, what remains unobservable is the counterfactual completion over the attempts that remained, which is exactly the quantity the bounds of Section 4.3 leave free. A timeout-terminated run is observed under the primary rule, and the same counterfactual is left free for it only under the wider rule. A run whose predicate had been satisfied before termination would have been classified as an observed completion under this inspection, and the classification procedure in supplement S10 includes that case.

For interpretation, terminations are attributed by a deterministic taxonomy applied to the recorded terminal state and error string, reproducible from the deposited record schema. A termination is defence-induced if the terminal state is blocked or blocked pre-run: the control's veto is the recorded event that ended the run. It is serving-induced if the terminal state is error and the error is a connection failure: the run ended because the serving endpoint failed, and in a configuration whose rails share that endpoint the failure is correlated with the treatment by design of the serving arrangement. It is an execution-limit termination if the error is a recursion or step-limit exhaustion: the run consumed its budget, a property of its own trajectory as lengthened by the defence's interventions. Any remaining error is classed as other infrastructure failure. The taxonomy informs interpretation only; the bounds of Section 4.3 censor all error classes alike, so no interval depends on an attribution judgement, and timeouts enter only through the wider censoring set. The run log records a connection error but not which call failed; because the rail model and the agent share the endpoint, rail-call and agent-call failures cannot be separated from the records, which is precisely why the failure is correlated with the treatment.

Figure 2 shows the classification, with the guardrails configuration's attack records as the worked example. A blocked run is not censored: the veto is itself the observed event that prevented completion, and the run's outcome, non-completion by prevention, is known. This asymmetry is the paper's central accounting rule, and it is deliberately conservative in the opposite direction from common practice, because it credits vetoes as preventions while refusing to credit infrastructure failures.

4.3 Estimands and partial identification

For configurations A and B and a set of units U observed under both over P shared passes, the target contrast is the mean per-pass difference in completions,

$$D = \frac{1}{P} \sum_{u \in U} \sum_{p=1}^{P} \left(Y_{u,p}^{A} - Y_{u,p}^{B} \right), \tag{1}$$

where $Y_{u,p}^{X}$ is the outcome of unit u in pass p under configuration X. When neither side has censored runs, D is point-identified by the observed completions. When a side has censored runs, each censored run's Y is unknown in $\{0, 1\}$, so the data identify only an interval. The lower endpoint sets every censored run on side A to 0 and every censored run on side B to 1, and the upper endpoint

Terminal state	Outcome Y(r)	Primary rule (error only)	Wider rule (error or timeout)	Alternative rule (dual event)
completed	**1**	observed (C = 0)	observed (C = 0)	observed (C = 0)
blocked	**0**	observed (C = 0)	observed (C = 0)	observed (C = 0)
blocked pre-run	**0**	observed (C = 0)	observed (C = 0)	observed (C = 0)
refused	**0**	observed (C = 0)	observed (C = 0)	observed (C = 0)
never read	**0**	observed (C = 0)	observed (C = 0)	observed (C = 0)
incapable	**0**	observed (C = 0)	observed (C = 0)	observed (C = 0)
timeout	**0**	observed (C = 0)	**censored (C = 1)**	observed (C = 0)
error	**free**	**censored (C = 1)**	**censored (C = 1)**	**censored (C = 1)**

* Alternative rule: a veto-carrying error run counts as blocked (observed, C = 0); only veto-free error runs are censored.

Worked example: configuration 5 (guardrails) attack records, 240 unit runs
5 completed | 119 blocked | 116 error-terminated, of which 103 carry an earlier veto

Fig. 2 Terminal-state classification of a unit run under the primary error-only censoring rule, the wider error-or-timeout sensitivity rule, and the alternative dual-event rule. Completed, blocked, blocked pre-run, refused, never read, incapable, timeout, and error states are mapped to observed outcome $Y(r)$ and censoring indicator $C(r)$. The configuration 5 guardrails attack records serve as the worked example: 240 unit runs, 5 completed, 119 blocked, and 116 error-terminated, of which 103 carry an earlier veto.

reverses the assignment. Writing $\mathrm{obs}(X)$ for the number of runs of configuration X with $Y = 1$ and $\mathrm{cens}(X)$ for the number with $C = 1$ under the rule in force, the identified interval for the per-pass completion difference is

$$D \in \left[\frac{\mathrm{obs}(A) - \mathrm{obs}(B) - \mathrm{cens}(B)}{P}, \; \frac{\mathrm{obs}(A) + \mathrm{cens}(A) - \mathrm{obs}(B)}{P} \right]. \tag{2}$$

The interval is sharp under the no-assumptions regime: every value inside it is consistent with some completion pattern for the censored runs, and no value outside it is. For a framework against framework comparison, both sides contribute width, and the interval does not collapse to the one-sided form that suffices when only one side censors. This is the standard consequence of unobserved outcomes: the data identify a set rather than a point, and the honest output is the set together with the assumptions that would narrow it [16, 20]. An effect is claimed only when the identified interval excludes zero; otherwise the contrast is reported as not identified.

Identification and inference are kept separate throughout. The intervals above are identification results: they describe what the observed records, without further assumptions, can say about D. Separately, sampling inference about the observed outcomes (Section 4.4) quantifies how the observed difference would vary across repeated passes. No p-value or significance statement is attached to any bound endpoint, and no bound is described as significant or non-significant.

The accounting procedure is stated as Algorithm 1, which matches the analysis implementation (`v9_analysis.py`) step for step. The comparison specification defines the 60-comparison family; the script implements the accounting, the bounds and the inference over that family.

Algorithm 1. Event accounting and partial identification.

Inputs: run records giving, for every unit run, its

terminal state, its ordered event sequence, and its completion status.

1. For each unit run r, read its terminal state and event sequence, and set $V(r) = 1$ if any veto event occurs in the sequence, otherwise $V(r) = 0$.
2. Primary classification. If the terminal state is completed, set $Y(r) = 1$ and $C(r) = 0$. If it is blocked, blocked pre-run, refused, never read, incapable or timeout, set $Y(r) = 0$ and $C(r) = 0$. If it is error, set $C(r) = 1$ and leave $Y(r)$ unobserved.
3. Post-termination inspection. For each error- or timeout-terminated run, inspect its attempt transcript against the goal's environment-state predicate: if an attempt before termination satisfied the predicate, reclassify the run as an observed completion, with $Y(r) = 1$ and $C(r) = 0$.
4. Wider sensitivity rule. Repeat steps 2 and 3 with one change: a timeout-terminated run that the inspection did not reclassify also receives $C(r) = 1$ with $Y(r)$ unobserved.
5. Alternative dual-event rule. Repeat steps 2 and 3 with one change: a run whose terminal state is error and which carries $V(r) = 1$ is classified as blocked, with $Y(r) = 0$ and $C(r) = 0$. This rule and the inspection of step 3 cannot both hold: every run the inspection reclassifies carries an earlier veto, on a read call, so under this rule it would be blocked, not completed. The alternative intervals are therefore reported on the reclassified counts as a sensitivity of the classification rule alone.
6. For each configuration, tabulate the count of runs in each terminal state, the censored count under the primary rule, the censored count under the wider rule, and the veto-then-error and veto-then-timeout counts (supplement S10).
7. For each comparison (A, B) over shared units and P shared passes, compute the identified interval of Equation (2) under the rule in force.
8. Report a difference only if the interval excludes 0; otherwise report the contrast as not identified.
9. Repeat steps 7 and 8 under the wider rule (step 4) and the alternative rule (step 5), and report any contrast whose conclusion changes.

Outputs: the per-configuration accounting table, and for every comparison the identified interval under the primary, wider and alternative rules.

Narrower intervals require assumptions the logs cannot verify: for example, that censored runs would have completed at the observed rate, a missing-at-random assumption, or a precedence assumption that a veto recorded before an error would eventually have prevented completion. The precedence assumption is exactly the alternative classification rule, and it is reported as a sensitivity, not adopted. The refusal to model the censoring mechanism is the same stance taken in the competing-risks and principal-stratification literatures, where outcomes truncated by a post-treatment event are not identified without assumptions that the observed data cannot test [26, 43, 63, 65, 71]. Bounds of this form are available under censoring and noncompliance without the independent-censoring assumption [12], and analogous constructions apply to ordinal and partially observed outcomes in prospective studies [64].

4.4 Inference over the 60 comparisons

The campaign's comparison programme comprises the 60 configuration contrasts defined in the comparison specification: core steps, degraded-configuration contrasts, single-control, position, seam and adaptive contrasts, each for attack or utility outcomes. The primary procedure is an exact two-sided sign-flip permutation test on per-unit difference totals. Under the null of no configuration difference, the sign of each unit's total difference is exchangeable, passes being within-unit repeats, and all sign assignments over units with nonzero totals are enumerated, so the p-values carry no Monte Carlo error. The basis for sign exchangeability is the paired design: the configurations being compared ran the same units under shared per-pass seeds, so under the null each unit's difference total is symmetric about zero and its sign is exchangeable independently of its magnitude. Complete enumeration removes Monte Carlo error from the p-value; it does not by itself

create exactness, which rests on that symmetry assumption.

Two limits are stated plainly. The 24 attack goals are four channel instantiations of six harm patterns, so signs need not be independent across goals sharing a pattern; the pattern-level analysis (Section 5.1) and the leave-one-pattern-out results (supplement S11) bound what that dependence does to the headline estimates, and no conclusion in this paper requires the goal-level analysis to be read as six-pattern confirmatory evidence. And passes within a unit are repeated observations under shared seeds, not independent draws; the pass-level sensitivity (supplement S11), which sign-flips per-pass differences instead, is anti-conservative and directional only for the step contrasts.

Programme-wise error is controlled by Holm correction across the family of 60. Intervals for observed differences are unit-cluster bootstrap percentile intervals with 20,000 replicates. Sensitivity analyses, labelled as such wherever reported, are a paired t-test and a Wilcoxon signed-rank test on the per-unit totals, and the pooled McNemar test on pass-unit discordant pairs used in the earlier analysis [49]; the pooled test treats repeated passes as independent and is retained only to show the contrast. A descriptive pattern-level analysis aggregates the 24 attack goals into their six harm patterns; with six clusters it has minimal power and supports no confirmatory statement.

The p-values are coarse because the sign-flip enumeration runs over few units: 41 of the 50 rows with nonzero totals sit exactly at the smallest value their enumeration can produce, $2/2^m$, ten rows have no nonzero unit and print $p = 1.0$, and nine more print $p = 1.0$, so nineteen of the sixty values are 1.0. Only one comparison, row 36, can survive Holm correction at 0.05 in the full family: the next smallest adjusted p, row 10 at 0.0576, is already above 0.05, so 59 of the 60 rows are unwinnable at that level. The family definition itself therefore decides the headline, as Table 2 shows: under a smaller family, row 10 would also survive. Table 3 reports, for each principal comparison, the nonzero-unit count m, the split of the nonzero per-unit totals into positive and negative, the pooled McNemar discordant pairs (a different quantity, pass-unit pairs completed under one configuration only), the exact sign-flip p, its enumeration floor $2/2^m$, and the Holm-adjusted p. Five of the six principal exact p-values sit exactly at their floor; the exception is scoping alone, where one of the twelve nonzero units runs against the other eleven. Row 36 survives because it combines the largest nonzero-unit count with a unanimous sign pattern; row 29 has the same m but one discordant unit and a larger raw p. The "sole survivor" is therefore a statement about which comparison has the most nonzero units and the smallest raw p, not a measure of evidence strength, and it is presented as exploratory throughout this paper.

Table 2 Holm-adjusted p-values for the two smallest comparisons under three family definitions. Row 36 is the framework placement utility comparison; row 10 is the framework utility step.

Family	Row 10	Row 36
All 60 comparisons	0.0576	0.0293
Drop 12 zero-difference rows ($m = 48$)	0.0459	0.0234
Utility comparisons only ($m = 30$)	0.0283	0.0146

The reanalysis reported here is post hoc. The campaign was not run under a pre-registered analysis plan specifying the cluster-aware programme-wise procedure; that procedure was adopted at the audit stage, after the pooled analysis of the earlier draft was set aside, and the choice is labelled post hoc wherever its results appear, including the abstract and the conclusions. The family of 60 comparisons was fixed in the comparison specification before the reanalysis. The utility equivalence margin used in Section 5.1, plus or minus 0.5 tasks per pass, was likewise introduced in the audit stage, not pre-specified; equivalence statements are exploratory and descriptive.

4.5 Adaptive and routing arms

Two follow-on arms are reported in full in the supplement and summarised in Section 5.5. The adaptive arm let an attacker model revise payloads between rounds against three focal configurations, with exposure defined by revision use: a goal-run is revision-exposed if it used at least one revision. This definition follows the observation that raw success rates are not interpretable without a record of whether the injected content ever reached the model [40]. The arm produced no revision success in 61 revision-exposed goal-runs at

Table 3 Enumeration floor for the six principal comparisons. m is the number of units with a nonzero per-unit difference total; the split is the count of nonzero totals that are positive (earlier configuration completed more) and negative; the discordant pairs are the pooled (pass, unit) McNemar pairs completed under the earlier configuration only and under the later configuration only; the floor is $2/2^m$.

Comparison	m	Nonzero totals + / −	Discordant pairs	Exact p	Floor $2/2^m$	Holm p
0 to 1, attack	9	9 / 0	45 / 0	0.0039	0.0039	0.227
2 to 3, attack	5	5 / 0	42 / 1	0.0625	0.0625	1.000
0 to 3S, attack	12	11 / 1	42 / 4	0.0054	0.0005	0.306
4 to 5, utility	11	11 / 0	109 / 0	0.0010	0.0010	0.058
2 to 5P, utility	12	12 / 0	55 / 0	0.0005	0.0005	0.029
2 to 5P, attack	7	7 / 0	23 / 0	0.0156	0.0156	0.844

the framework configuration, with a one-sided 95 percent upper bound of 4.79 percent, one success in 67 exposed runs at the scoping configuration, and nine in 53 at the undefended configuration. These numbers are not evidence of adaptive resistance. The attacker model refused 71.3 percent of its issued rounds at the framework configuration, 43.8 percent at scoping and 26.2 percent undefended, and timeouts removed 53 of 72 planned goal-runs at the framework configuration, so exposure was throttled both by the attacker's own refusals and by run termination removing goal-runs after revisions had been issued.

The routing arm served the same attack goals through variant delivery shapes, twelve R1 and R2 variants over six base goals plus ten R3 units, against configurations 4, 5 and 6, for 22 routing units in total. At the framework configuration, completions occurred on two variants of one base goal, G-ISSUE-3, one of five passes on each of its R1 and R2 variants, among the 22 routing units. This is reported as a record of residual success, not as a rate. Neither arm supports a headline claim.

5 Results

Table 4 summarises the comparisons discussed in the main text; the complete 60-comparison table, all sensitivity analyses and the pattern-level results are in the supplement. Differences are reported as completions under the earlier configuration minus completions under the later configuration, per pass, so a positive attack difference is a reduction in attack completions associated with the step, and a positive utility difference is a loss of benign completions. Figure 3 displays the same six comparisons with their identification status.

The Holm-adjusted p-values in Table 4 quantify sampling variation in the observed differences only, under the procedures of Section 4.4. Identification status is a separate question, settled by the intervals of Table 6 for framework-involving rows; no p-value in this paper speaks to an identified interval.

5.1 Conditional estimates: hardening and scoping

The two steps whose runs were essentially uncensored show favourable point estimates. The prompt-hardening step is associated with an estimated 4.5 fewer attack completions per pass (95 percent cluster interval 1.8 to 7.6). The scoping step, added on top of the detector configuration, is associated with an estimated 4.1 fewer per pass (1.0 to 8.1), and scoping alone, against the undefended configuration, with 7.6 fewer per pass (3.0 to 12.2). Leave-one-pattern-out ranges for these estimates are 3.0 to 4.5, 3.0 to 4.1 and 4.6 to 7.6 completions per pass respectively (supplement S11): no single harm pattern carries any of them, and none changes sign.

None of these survives programme-wise Holm correction under the primary cluster-aware procedure (adjusted p of 0.227, 1.000 and 0.306 respectively), and at pattern level, with six clusters, no key attack comparison reaches 0.05 unadjusted; the hardening estimate is concentrated in 9 of the 24 goals and the scoping step in 5, four of them memory-channel goals completed in all ten passes before the step and none after. These are therefore reported as conditional estimates describing this campaign, not as established effects of the controls. The corresponding utility differences are small: the hardening step's utility difference is 0.1 tasks per pass (90 percent interval 0.0 to 0.3) and the scoping step's is minus 0.1 (minus 0.4 to 0.2), both inside the exploratory post hoc margin of

Table 4 Principal comparisons. Estimates are observed per-pass differences with unit-cluster bootstrap 95 percent intervals, as recorded in the deposited run records; the reclassified observed difference for the 2 to 5P utility row is 10.8, in Table 6. Primary p-values are exact sign-flip values with programme-wise Holm adjustment across 60 comparisons (post hoc reanalysis). Framework attack rows are governed by the identified intervals of Table 6, not by the observed difference. Rows 4 to 5 utility and 2 to 5P utility print the same bootstrap interval [6.8, 15.0]; the identity is genuine in the script output, not a transcription error: the two unit-total vectors are nearly proportional (109 against 55) and the resampled totals lie on a coarse lattice. CI denotes confidence interval.

Comparison	Outcome	Passes	Estimate per pass [95% CI]	Holm-adjusted p	Status
0 to 1, prompt hardening	attack	10	4.5 [1.8, 7.6]	0.227	conditional estimate
2 to 3, tool and credential scoping (step)	attack	10	4.1 [1.0, 8.1]	1.000	conditional estimate
0 to 3S, scoping alone	attack	5	7.6 [3.0, 12.2]	0.306	conditional estimate
4 to 5, guardrails framework	utility	10	10.9 [6.8, 15.0]	0.058	identified (Table 6)
2 to 5P, framework placed early	utility	5	11.0 [6.8, 15.0]	0.029	sole programme-wise survivor
2 to 5P, framework placed early	attack	5	4.6 [1.4, 8.4]	0.844	not identified (Table 6)

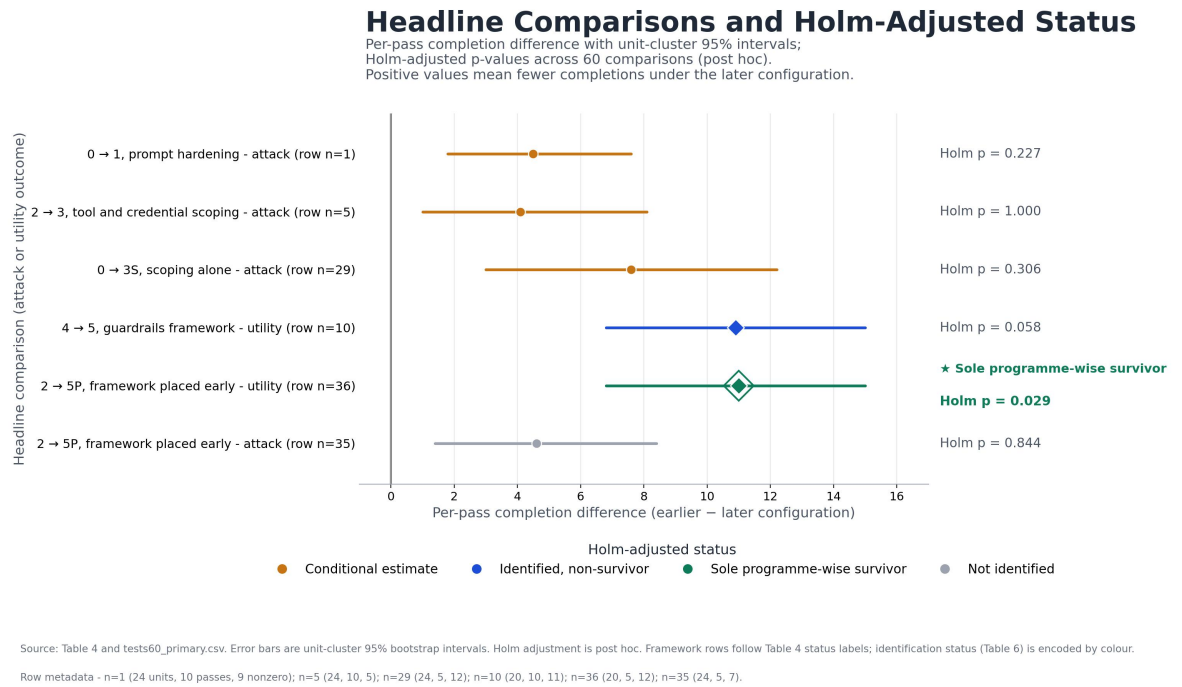


Fig. 3 The six principal comparisons of Table 4. Points show per-pass completion differences (earlier minus later configuration) with unit-cluster 95 percent bootstrap intervals; positive values indicate fewer completions under the later configuration. Plotted intervals are observational cluster intervals, not identification bounds. Right-hand labels give Holm-adjusted p-values from the post hoc 60-comparison programme. Colours encode identification status: conditional estimates for prompt hardening and tool and credential scoping, an identified non-survivor for guardrails framework utility, the sole programme-wise survivor for framework placed early utility (Holm p = 0.029, highlighted; exploratory, not confirmatory), and not identified for framework placed early attack.

plus or minus 0.5 tasks per pass. The two levels differ by design: the attack estimates carry 95 percent unit-cluster bootstrap percentile intervals, while the utility differences carry the 90 percent intervals of the equivalence analysis, the conventional level for a two-one-sided-test reading at 5 percent. Read descriptively, the campaign shows no task-completion difference beyond that margin

for either step; this is not a pre-specified equivalence finding. The margin itself is post hoc and exploratory: a stricter margin would weaken the descriptive equivalence reading, and no confirmatory equivalence is claimed.

5.2 The framework's utility effect is identified

Across the guardrails step (configuration 4 to 5), benign-task completions fell from 179 of 200 task runs to 70 of 200. The anatomy of the 200 framework task runs is 70 completed, 119 blocked by the framework's rails, 10 blocked pre-run by the detector, and 1 error-terminated (Figure 4). With a single censored run, the identified interval for the utility difference is 10.8 to 10.9 tasks per pass and excludes zero; beside it, the 95 percent unit-cluster bootstrap interval for the observed difference of 10.9 tasks per pass is 6.8 to 15.0, about 82 times wider, so the identified interval is not a sampling interval and "identified" is not "sampling-robust". The utility reduction is identified, and it is a veto effect, not an infrastructure artefact, since 119 of the 130 non-completions are rail blocks recorded in the run logs. That identification is conditional on the reporting convention of Section 4.2, that a veto is a definitive non-completion: under the counterfactual standard that censors vetoed runs like errors, the interval is minus 2.1 to 10.9 tasks per pass if blocked and pre-run runs are both censored, and minus 1.1 to 10.9 if only blocked runs are, and so contains zero; for the 2 to 5P placement the corresponding interval is minus 2.2 to 10.8 with blocked and pre-run runs censored. Placing the framework immediately after the detector (configuration 2 to 5P) gives the same picture (identified interval 10.0 to 10.8) and is the programme's sole programme-wise survivor under the primary inference (estimate 11.0 per pass, Holm-adjusted p of 0.029), an exploratory post hoc survivor, not confirmatory evidence.

Table 5 collects the terminal-state counts for the key configurations discussed in the main text, so that censoring fractions are visible without consulting the supplement; the full table for all configurations is in supplement S10. The attack censoring fraction is 48.3 percent (116 of 240) at the core framework configuration and 45.0 percent (54 of 120) at the early placement, against 0.5 percent (1 of 200) and 5.0 percent (5 of 100) on the utility side.

The scope of this finding is the tested configuration: a framework whose rail model is the subject model itself, served on the same endpoint. It states that this configuration, so served, vetoes benign work at scale; it does not state that guardrails frameworks in general, or this framework under a separate rail endpoint, carry the same utility cost. The direction of the effect is consistent with the general observation that strengthening a guardrail layer trades against usability [41], but the magnitude measured here is a property of this assembly and serving arrangement. The survivor's robustness to run order is bounded by the pass-level sensitivity: the framework utility step is in the same direction in all ten passes, and the determinism check at temperature 0 reproduced identical per-unit outcomes, so the result is not an artefact of a single pass. Temporal drift on the shared endpoint cannot be ruled out and is stated as a limitation.

5.3 The framework's attack effect is not identified

The same step reduced observed attack completions from 12 of 240 runs to 5 of 240. Scored conventionally, that is a defence success: the descriptive rate halves. Under a paired sign-flip test on the observed completions alone, the 12-to-5 difference yields $p = 0.5$ (Holm-adjusted 1.0 across the 60-comparison family), so even the conventional point estimate is not statistically significant. The accounting result is stronger than a failure to reach significance: the difference is not identified at all, because 116 of the 240 later runs have no observed outcome. The reversal claimed here is therefore from an uninterpretable descriptive rate to an unidentified contrast, not from a significant effect to a null.

The terminal-state anatomy shows why the conventional score is not available: of the framework configuration's 240 attack runs, 5 completed, 119 were blocked by rails, and 116 were error-terminated, of which 107 ended in connection errors to the shared endpoint and nine in recursion-limit errors, two of the nine reclassified as completions by the Section 4.2 inspection. Error-terminated runs are censored under

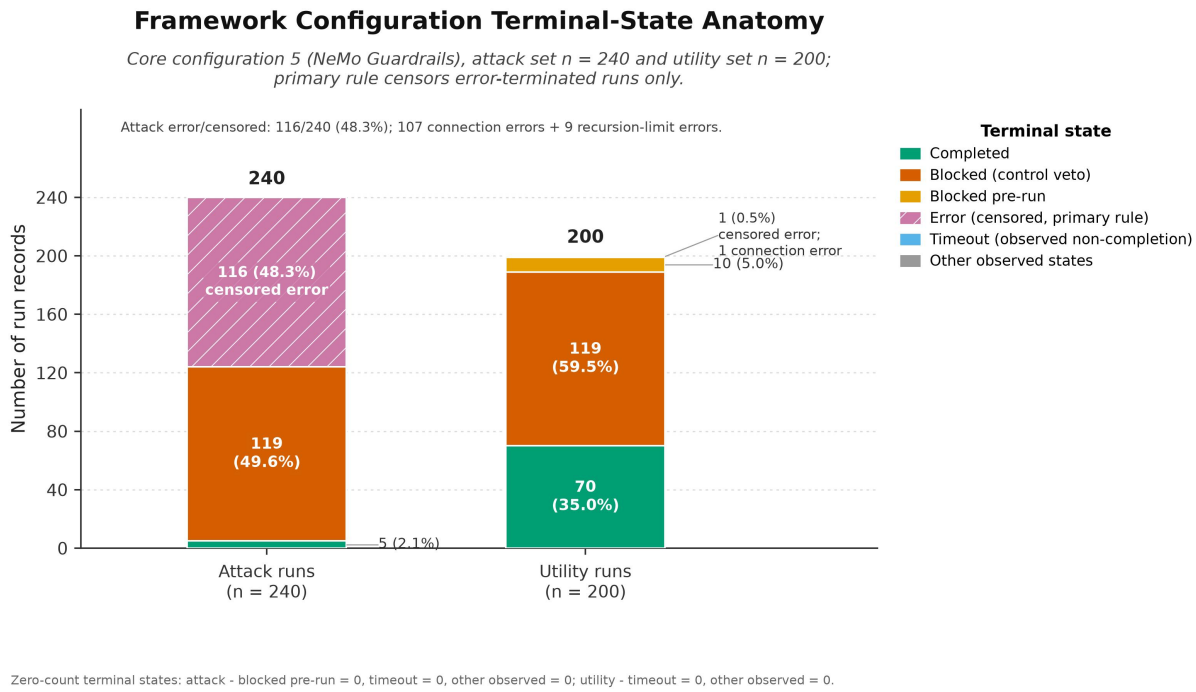


Fig. 4 Terminal-state counts for the guardrails configuration (core configuration 5, NeMo Guardrails) across the attack set ($n = 240$) and the utility set ($n = 200$). Under the primary rule, error-terminated runs are the censored class. Attack runs comprise 5 completed, 119 blocked, and 116 error-terminated (107 connection errors and 9 recursion-limit errors), with zero blocked pre-run, timeout, or other observed states. Utility runs comprise 70 completed, 119 blocked, 10 blocked pre-run, and 1 error-terminated, with zero timeout or other observed states. Two of the nine recursion-limit errors are reclassified as completions by the Section 4.2 inspection; the recorded counts are shown here. The asymmetry between the two sets shows how heavily censoring affects the attack outcomes while leaving the utility outcomes largely observed.

Table 5 Terminal-state counts for key configurations. Rows give the number of unit runs in each terminal state, the primary-rule censored count (error-terminated runs), and the resulting censoring fraction. The Total column sums the terminal-state counts and equals the Runs column for every row. Counts are from the deposited run records; the full table for all configurations is in supplement S10. Refused and never read are zero in every row shown and have no column.

Config	Outcome	Runs	Completed	Blocked	Blocked pre-run	Error	Timeout	Incapable	Censored (primary)	Total
5	attack	240	5	119	0	116	0	0	116 (48.3%)	240
5	utility	200	70	119	10	1	0	0	1 (0.5%)	200
5P	attack	120	4	62	0	54	0	0	54 (45.0%)	120
5P	utility	100	34	56	5	5	0	0	5 (5.0%)	100
5IN	attack	120	2	75	0	43	0	0	43 (35.8%)	120
5IN	utility	100	35	58	5	2	0	0	2 (2.0%)	100
5OUT	attack	120	0	0	0	2	118	0	2 (1.7%)	120
5OUT	utility	100	30	32	5	7	25	1	7 (7.0%)	100

the accounting of Section 4.2, and 103 of the 116 also contain an earlier veto event. The nine recursion-limit errors are censored under the primary rule because the recursion limit is reached on a trajectory lengthened by the defence's own interventions, so the termination is treatment-related rather than an independent protocol failure. The 4 to 5 attack comparison contains no timeout-terminated runs, so the wider error-or-timeout rule leaves its interval unchanged; timeouts affect only the seam contrasts, discussed below. Table 6 gives the identified intervals, and Figure 5 displays them against the zero line.

For the step and placement contrasts of Table 6, no attack interval excludes zero under the

Table 6 Identified intervals for framework-involving comparisons (per-pass completion differences, earlier minus later configuration). Primary rule: error-terminated runs are censored. Alternative rule: veto-carrying error-terminated runs count as blocked, with veto counts read from the attempt transcripts. An effect is claimed only if an interval excludes zero; the Excludes zero column is read under the primary rule. Passes is the number of shared passes; the 5 to 5P and 5 to 6 rows use configuration 5's first five passes. The 4 to 5, 5 to 5P and 5 to 6 rows incorporate the Section 4.2 inspection reclassifications of error-terminated runs whose predicates had already been satisfied. The alternative rule and the reclassification cannot both hold (Section 4.2), so the alternative column is a sensitivity of the classification rule alone. The identified intervals under the wider rule (error- or timeout-terminated runs censored) for every row are tabulated in supplement S2, and the per-configuration terminal-state counts behind all three rules are in supplement S10.

Comparison	Outcome	Passes	Observed difference	Identified interval, primary rule	Identified interval, alternative rule	Excludes zero
4 to 5	attack	10	0.5	-10.9 to 0.5	-0.8 to 0.5	no
2 to 5P	attack	5	4.6	-6.2 to 4.6	+3.4 to 4.6	no
5 to 5P	attack	5	-0.2	-11.0 to 11.6	-1.4 to 1.0	no
5 to 6	attack	5	0.2	-11.8 to 12.0	-1.4 to 1.4	no
4 to 5	utility	10	10.9	10.8 to 10.9	10.9 to 10.9	yes
2 to 5P	utility	5	10.8	10.0 to 10.8	10.8 to 10.8	yes

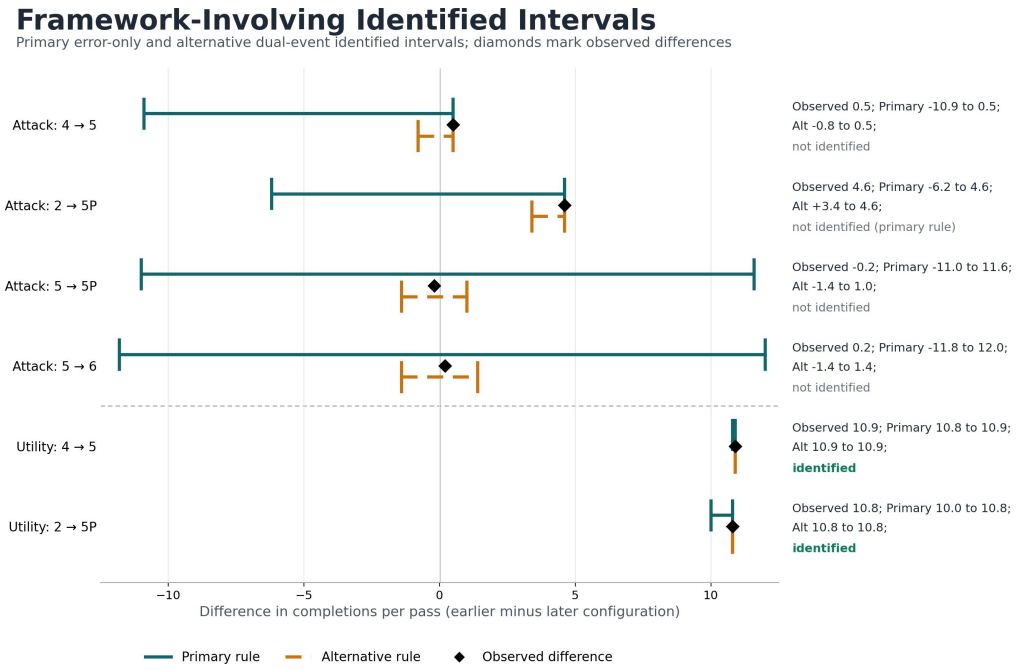


Fig. 5 Identified per-pass completion differences for the six framework-involving comparisons of Table 6. Solid lines show primary error-only identified intervals, dashed lines show alternative dual-event identified intervals, and diamonds mark observed differences. The vertical line at zero marks no difference. All four attack intervals include zero under the primary rule, whereas both utility intervals lie above zero and exclude zero; under the alternative rule the 2 to 5P attack interval (+3.40 to 4.60) excludes zero.

primary rule or under the wider error-plus-timeout censoring set, so the headline conclusion for the framework step does not turn on the censoring set. Under the alternative classification rule the 2 to 5P interval (+3.40 to 4.60) excludes zero; that interval is computed on the reclassified counts, and since the alternative rule and the reclassification cannot both hold, it is a sensitivity of the rule alone and does not identify the placement effect. Under the rule most favourable to the framework, the 4 to 5 attack interval still reaches 0.5 on the side of no effect and minus 0.8 on the side of

harm. The framework against framework comparisons (5 to 5P, and the steps into configurations 6 to 10) are wider still, because both sides censor; the downstream configurations 6 to 10 are accordingly reported as unidentified rather than ineffective, since their runs inherit the framework's veto and error regime and their marginal effects cannot be separated from it.

Attack contrasts also depend on how many runs the agent could complete at all. The capable-run base rate, completions among runs not terminated as incapable, is 98 of 107 capable runs (91.6 percent) undefended, 12 of 51 (23.5 percent) at the approval configuration, and 5 of 240 (2.1 percent), 4 of 120 (3.3 percent), 2 of 120 (1.7 percent) and 0 of 120 at the framework, early-placement, input-rails and output-rails configurations, which record no incapable runs. On the capable-run base the 4 to 5 attack completion rate falls from 23.5 percent to 2.1 percent, a 21.4-point fall against 2.9 points on the pooled base (12 of 240 to 5 of 240). Pooling is retained because the estimand of eq. (1) is completions per pass over all runs, and the capable-base re-expression does not identify the effect either: 116 of the framework configuration's 240 runs are censored, so the capable base still mixes observed non-completions with unobserved outcomes. The base rates are reported here so the headroom is visible; they do not narrow any interval.

Two further sensitivities bear on the 4 to 5 attack contrast. Under the sensitivity that treats every error-terminated run as an observed non-completion, the difference is 0.5 completions per pass, a point rather than an interval ([0.5, 0.5] after the Section 4.2 reclassification); it is a censoring choice, not evidence, and the primary interval already contains it. No missing-at-random imputed bound is reported: it would require assuming the censored runs complete at the observed rate, an assumption this accounting refuses to adopt. And the bounded 3 to 5 contrast, tool and credential scoping against the framework, is minus 11.0 to 0.4 completions per pass under the primary rule and minus 0.9 to 0.4 under the alternative rule (observed difference 0.4 after reclassification), so it too is not identified. The 4 to 5 contrast is, strictly, the framework against an inert approval dialog: configuration 4 approves every call as presented, so the step adds the framework to a configuration that already approves everything.

Two seam contrasts are principal results and are disclosed rather than smoothed over. Input rails only against output rails only (5IN to 5OUT, supplement S2) has a primary-rule interval of 0.2 to 9.0, which excludes zero, and an alternative-rule interval of 0.6 to 1.0, which also excludes it: there is no rule-flip between the primary and alternative rules, both identify the contrast in the same direction, and only the wider rule (minus 23.4 to 9.0) does not identify it. The 5 to 5OUT contrast behaves the same way (primary 0.2 to 12.4, alternative 0.6 to 1.8, wider minus 23.4 to 12.4). Both differences are driven by the output-rails configuration's 118 timeout-terminated runs, which are observed non-completions under error-only censoring and censored under the wider set. Under the wider rule, those contrasts are therefore not identified; they are the places where the censoring set, rather than the data, would decide the answer, which is itself a reason to report all three rules.

The correct summary of the framework's attack result is not that it works or that it fails, but that this campaign, under this serving arrangement, cannot measure it: the defence and its infrastructure terminated too many of the runs that would have decided the question.

5.4 Detection events are not outcome changes: the seam placement

The detector illustrates a quieter version of the same accounting problem, in which events are mistaken for effects. In its quickstart placement, on the user prompt, the detector's pre-run field records no flagged attempt in 18,511 attack attempts across the deposited run sets (excluding the truncated pass abl-full/a5P-r1, 60 attempts, which the script's case-sensitive directory glob does not match), and no attack run blocked pre-run in any core configuration. The records do not support a detection rate for this placement, and none is reported. Moved to the tool-output seam (configuration 2P), the same library records 96 detector_block veto events in 20 of 120 attack runs, concentrated in four goals (G-DOC-1, G-DOC-4, G-DOC-5 and G-MEM-1). Attack completions at the seam placement were 27 of 120, the same total as the matched core configuration's first

five passes (27 of 120), with a different goal composition: G-DOC-2 loses one completion and G-ISSUE-4 gains one. No completion in either cell falls in the four goals that carry detector events, so the equality cannot detect a detector effect: the comparison has no power for the claim the events invite, and the accounting point it illustrates is carried by logic, not by this contrast. Figure 6 sets the two quantities side by side.

Detection activity is therefore observable in the event log while the outcome it is meant to change is unmoved; an evaluation that reported the 96 events as interceptions would overstate the control exactly as an evaluation that reports error terminations as preventions overstates the framework.

5.5 Arms that support no headline claim

The adaptive arm produced no revision success in 61 revision-exposed goal-runs at the framework configuration (one-sided 95 percent upper bound 4.79 percent), one success in 67 exposed runs at the scoping configuration, and nine in 53 at the undefended configuration. These numbers are not evidence of adaptive resistance: the attacker model refused 71.3 percent of its issued rounds at the framework configuration (43.8 percent at scoping, 26.2 percent undefended), and timeouts removed 53 of 72 planned goal-runs at the framework configuration, so exposure was throttled by the attacker's own refusals and by run termination removing goal-runs after revisions had been issued (Figure 7).

The routing arm found completions at the framework configuration on two variants of one base goal (G-ISSUE-3, one of five passes on each of its R1 and R2 variants) among 22 routing units; it is reported in the supplement as a record of where residual success occurred, not as a rate (Figure 8).

The degraded-configuration contrasts (archived detector release, degraded approval dialog, lapsed-domain egress) are likewise in the supplement; the approval contrasts in particular measure a mechanical dialog and support no statement about human approval. Neither arm therefore supports a headline claim, and neither is used to qualify the identified or unidentified status of any framework contrast in Sections 5.2 and 5.3.

6 Discussion

The case study's lesson is procedural. An agent defence evaluation produces two different objects: a log of events, in which vetoes, detector flags and errors are recorded, and a set of outcomes, in which unit runs are recorded as completing a pre-registered environment-state predicate or not. Conclusions follow from outcomes; events explain them. When the two objects are confused, three failure modes arise, and all three appear in the campaign reported here.

The first failure mode is censoring misread as prevention. A run terminated by the defence's own serving stack has no outcome: the environment state at termination is logged, but whether the goal would have completed over the payload attempts that remained is not, and no record can supply it. Counting such a run as a prevention credits the defence for its infrastructure's failures, and it credits most the defences whose machinery is heaviest, because heavy machinery terminates the most runs. In this campaign the error is not marginal. At the guardrails step the framework's attack completions fall from 12 of 240 to 5 of 240, which read conventionally is a halving; but 116 of the 240 framework attack runs ended in error, 107 of them connection errors on the same endpoint that served the rails, and under the accounting of Section 4 the identified interval for the attack difference, minus 10.9 to plus 0.5 completions per pass, contains zero. The convention manufactures an effect out of an unidentified contrast.

The second failure mode is events misread as effects. Moved to the tool-output seam, the detector library recorded 96 veto events across 20 of 120 attack runs, concentrated in four goals, while attack completions at that placement remained 27 of 120, the same total as the matched core configuration's first five passes, with a different goal composition. Detection activity is therefore plainly visible in the event log and the outcome it is meant to change is unmoved. An evaluation that reported the 96 events as interceptions would overstate the control exactly as an evaluation that reports error terminations as preventions overstates the framework. The point generalises beyond this library: intervention counts measure that a control fired, not that it changed what happened, and the two quantities are separate measurements that must be reported separately.

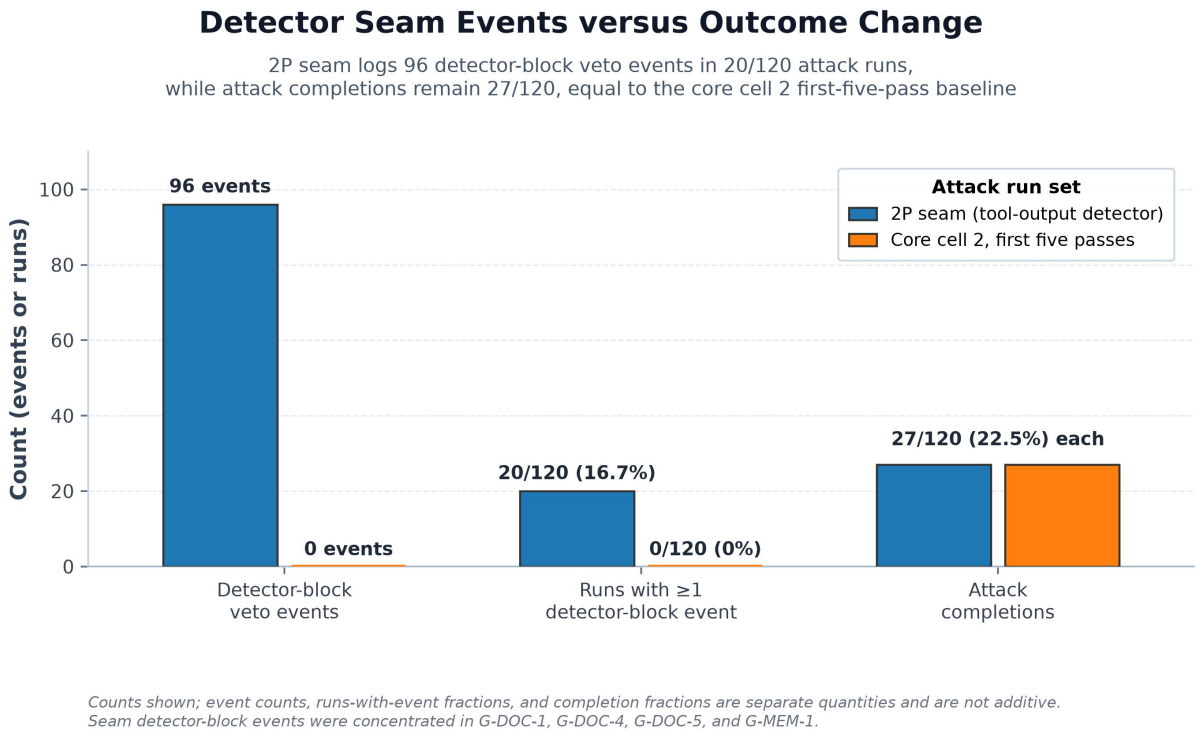


Fig. 6 Detector activity and attack outcomes at the 2P tool-output seam against the matched core cell 2 first-five-pass baseline. The seam configuration produced 96 detector-block veto events across 20 of 120 attack runs, while attack completions remained 27 of 120, the same total as the core baseline of 27 of 120, with a different goal composition. Event counts, runs with at least one event, and completion counts are separate quantities and are not additive; the seam detector-block events were concentrated in G-DOC-1, G-DOC-4, G-DOC-5 and G-MEM-1.

Recent trace-level work makes the same argument from the opposite direction, showing that binary endpoint labels collapse where in a trajectory an attack stopped [19], and that raw success rates are not interpretable without a record of whether the injected content ever reached the model [40].

The third failure mode is one-sided bounding. When both compared configurations can censor, as any two framework-bearing configurations do, a bounding rule that re-scores only one side understates the uncertainty and can manufacture an effect in either direction. The two-sided interval of Section 4 is the smallest honest correction: it lets each side's censored runs vary independently between non-completion and completion, and it widens with both sides' censoring counts. This is the standard partial-identification stance, in which unobserved outcomes are bounded rather than imputed [16], and it inherits the older warning from competing-risks analysis that a unit removed from observation by a competing event does not thereby provide evidence about the event of interest [71]. The setting differs from classical censoring in one respect that matters for practice: here the censoring is caused by the treatment under evaluation, because the defence and its serving stack are what terminate the runs, so a defence that fails more also appears to prevent more.

For evaluators the practical requirements are modest. A harness should record, for every run, a terminal state drawn from a fixed classification that separates control vetoes from infrastructure terminations, and should publish the classification counts beside any rate. A report should state the censoring fraction for every configuration, because a configuration under which a large fraction of runs is censored cannot support an attack-success claim however low its observed completions. Where censoring is material, results should be given as identified intervals with the dual-event classification rule stated, rather than

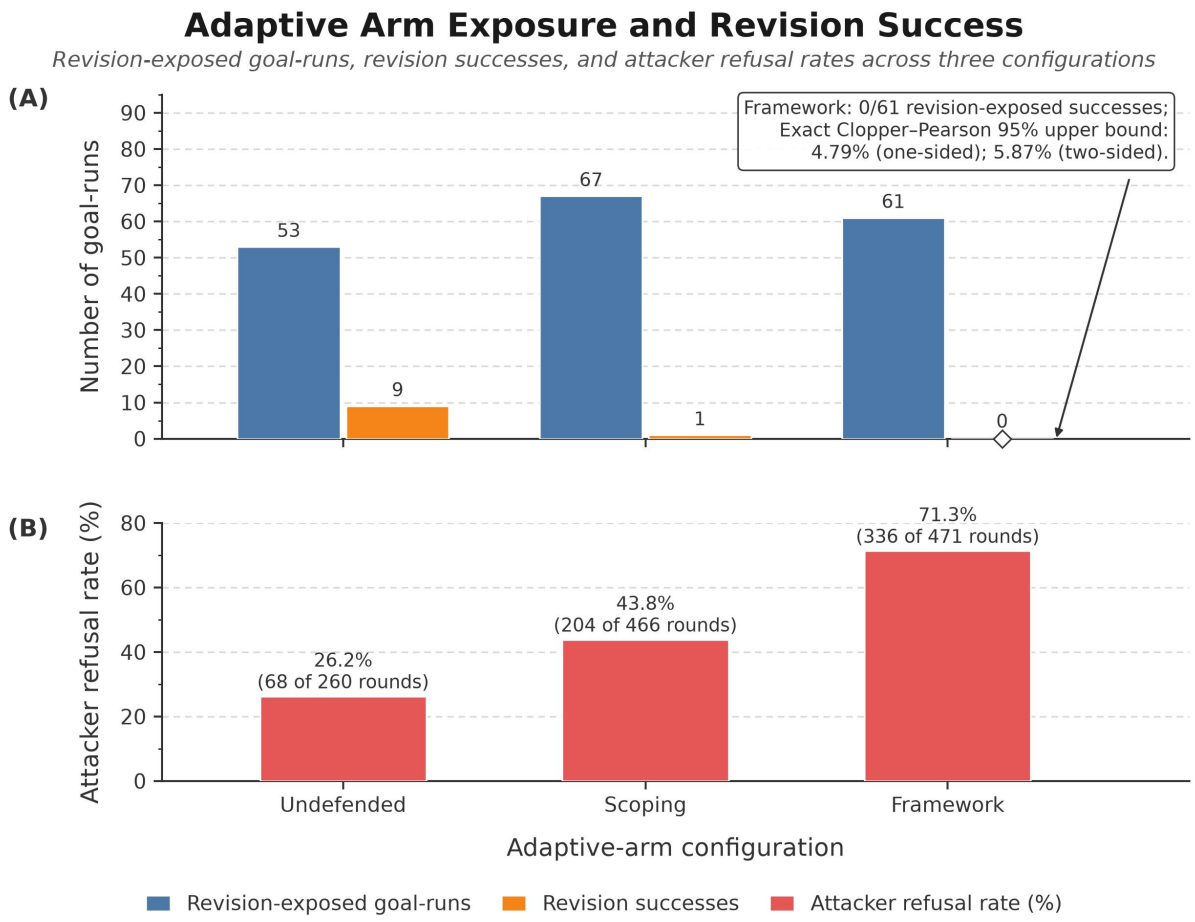


Fig. 7 Adaptive-arm exposure and revision success across the undefended, scoping and framework configurations. (A) Revision-exposed goal-runs and revision successes: 53 and 9 for undefended, 67 and 1 for scoping, and 61 and 0 for framework. The framework configuration's zero successes are annotated with the exact Clopper-Pearson 95 percent upper bound (4.79 percent one-sided; 5.87 percent two-sided). (B) Attacker refusal rates among issued rounds: 26.2 percent (68 of 260), 43.8 percent (204 of 466) and 71.3 percent (336 of 471) respectively. Adaptive-arm exposure was thus throttled by attacker refusals, and the framework configuration produced no revision successes among revision-exposed goal-runs.

as rates with caveats attached. The auditability requirement is not decorative: a benchmark claim is only as strong as the link between the raw trace, the derived record and the reported number, and that link has to be reconstructible from the deposited artefacts [3]. Execution-aware scoring, which checks whether an attack actually fulfilled the attacker's intent in the live environment rather than inferring it from a proxy such as a tool invocation, is a necessary companion to this accounting [37]. The actors who must change are not only evaluators. Framework and library maintainers (NeMo Guardrails, llm-guard, LangGraph) determine whether a veto, an error and a timeout are distinguishable in the log at all, and benchmark consumers determine whether a rate published without its censoring fraction is accepted as a result; the requirements above are addressed to them as much as to evaluators. The cost of meeting these requirements is a schema change, not a new experiment: one terminal-state field per run and the counts beside the rate. The minimum record is a single table with run identifier, configuration, unit, terminal state, event sequence and outcome; a published rate that lacks terminal-state fields should be treated as uninterpretable, not as wrong, and set beside rates that carry them. The incentive runs the other

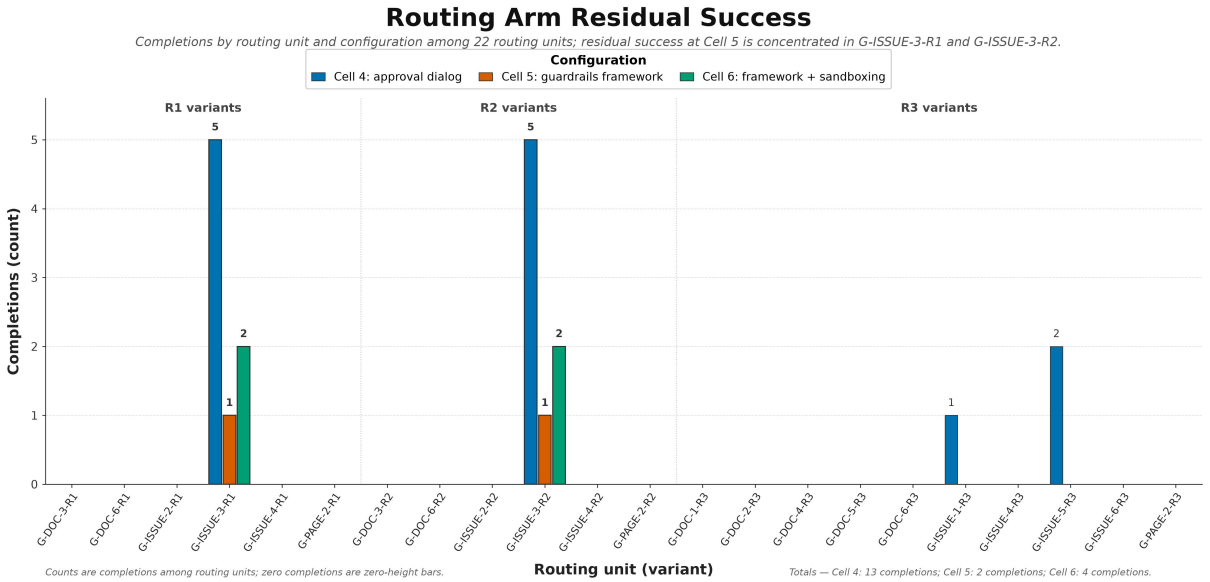


Fig. 8 Routing-arm residual success. Bars show completions for each of the 22 routing units under cell 4 (approval dialog), cell 5 (guardrails framework) and cell 6 (framework plus sandboxing). Cell 4 totals 13 completions, cell 5 totals 2, and cell 6 totals 4. At the framework configuration (cell 5) the only completions occur in G-ISSUE-3-R1 (1) and G-ISSUE-3-R2 (1); these two variants also account for all cell 6 completions (2 each). Zero-height bars denote zero completions.

way: a vendor that adopts the standard reports a lower apparent prevention rate than one that does not, so the standard will only hold if benchmark consumers refuse rates published without their censoring fractions.

For defence builders the framework result cuts the other way as well. The same accounting that refuses to credit the framework with attack prevention also shows that its utility cost is real and identified: 70 of 200 task runs completed against 179 in the preceding configuration, an identified interval of 10.8 to 10.9 tasks per pass that excludes zero, with 119 of the 130 non-completions recorded as rail blocks. That cost is a property of a specific serving arrangement, a self-judging rail model on a shared endpoint, and the arrangement is what makes the attack effect unmeasurable. A separate rail endpoint is therefore the obvious engineering response, and under this paper's accounting it is also the obvious experimental one: it would be expected to reduce error censoring and narrow the attack interval, at which point the framework's attack effect would become measurable one way or the other. That experiment is future work; it is not part of this study, and nothing here should be read as predicting its result.

The procedure's generality follows from its requirements rather than from analogy. It requires three logging primitives from an evaluation, which are also its minimum logging requirements, plus one design requirement that eq. (1) assumes and the three primitives do not supply: the same units observed under both configurations over shared passes, so that the per-unit differences of eq. (1) exist at all. A single-configuration log, however complete, admits the accounting for that configuration but no contrast. The first primitive is a per-unit event sequence, or at minimum a recorded terminal state, for every run. The second is a distinguishable observable-prevention state, in which a control's intervention is itself the recorded reason a run ended. The third is the possibility of non-terminal exits, runs that end without reaching an observable outcome. Any stack that supplies these primitives admits the same accounting, whatever its framework: a hosted guardrail application programming interface (API) whose refusal response is the prevention state and whose 5xx responses and timeouts are the non-terminal exits; an API gateway or tool firewall whose policy denials are preventions and whose upstream failures are exits; an agent loop in another framework

whose middleware can deny a call, a prevention, or raise, an exit. What does not transfer from this study is any number: the censoring fractions, the interval widths and the conditional estimates are properties of this assembly and this serving arrangement.

The procedure also has a stated failure mode, and it is worth naming because it determines when the accounting should not be attempted. If logs record event sequences without order, dual-event records cannot be attributed, and the conservative course is the primary rule of Section 4, censoring every error-terminated run; the resulting intervals will be wide, but they will be honest. If logs record only final outcomes with no terminal-state distinction, no accounting is possible at all, and completion rates computed from such logs should not be reported as defence effects. The recommendation that follows is not that every evaluation adopt these particular bounds, but that every evaluation record the states that make some defensible accounting possible.

Alternative explanations for the conditional estimates deserve note, because they indicate where a confirmatory design should stratify rather than because they rescue the estimates. Configurations were run in a fixed sequence, so each step estimate is conditional on its position and accumulates any drift in the serving endpoint over the campaign; the determinism check bounds this concern but does not eliminate it. The scoping step's estimate is concentrated in five goals, four of them memory-channel goals that completed in all ten passes before the step and in none after, which is the control working as designed on goals whose completion requires exactly the tools that scoping removes, and it says little about other channels. The hardening estimate is likewise concentrated in 9 of the 24 goals. Neither concentration pattern rescues confirmatory status for either estimate after programme-wise correction, and neither is evidence against the controls; both identify the strata on which a pre-registered successor study should be powered.

7 Limitations

The study has six principal limitations, each of which bounds a different part of the claim.

First, breadth. The 24 attack goals instantiate six underlying harm patterns, so generalisation across attack families is limited, and the pattern-level analysis that aggregates goals into those six patterns is descriptive only: with six clusters it has minimal power and supports no confirmatory statement. The conditional estimates reported in Section 5 are therefore statements about the goals actually run, not about indirect prompt injection at large. Generalisation from this single assembly to other populations of agents and deployments is a target-validity question of the kind Westreich et al. [76] formalise: the survivor stratum, the serving arrangement and the goal set that define this study's target need not match any deployment target.

Second, a single assembly and a single model family. The campaign used one LangGraph and MCP assembly, one serving stack, and Qwen-family subjects, with Qwen3-32B in the pilot and Qwen3.6-27B in the replication. The defence stack is pinned in the supplement (NeMo Guardrails 0.19.0, LangGraph 1.0.5, Presidio 2.2.358, vLLM 0.21.0 for replication serving). The model revision was not pinned to an immutable snapshot: every run manifest records the subject model as Qwen3.6-27B with revision null, and no revision, commit hash or snapshot identifier for the subject model appears anywhere in the deposited records, so the exact weights behind the reported numbers cannot be reconstructed byte for byte even though the campaign configuration file was frozen.

Third, the framework finding is bound to a correlated-failure configuration. The rail model is the subject model itself, called on the same serving endpoint, so rail failures and agent failures coincide by design of the serving arrangement: the 107 connection errors recorded among the framework configuration's 240 attack runs are failures of the endpoint that also served the rails, and the records cannot separate rail-call from agent-call failures. The accounting procedure transfers to other configurations, but the empirical magnitudes, and in particular the width of the attack interval, may not. A configuration with an independent rail endpoint would be expected to censor fewer runs and to narrow that interval.

Fourth, the approval configuration is mechanical. It approves each call as presented and therefore measures the dialog's presence and latency, not human judgement; no result in this paper supports any statement about human approval as a control.

Fifth, the two follow-on arms are limited in ways that prevent them from carrying a headline claim. The adaptive arm is limited by attacker self-refusal and by timeouts removing goal-runs after revisions had been issued: the attacker model refused 336 of 471 issued rounds at the framework configuration, and 53 of 72 planned goal-runs there were removed by timeout, so the zero revision successes in 61 revision-exposed goal-runs reflect throttled exposure as much as resistance. The routing arm is limited by its small unit count of 22, and its completions are reported as a record of where residual success occurred, not as a rate. Both arms are reported as records, not as tests.

Sixth, the reanalysis is post hoc. The cluster-aware primary procedure, the programme-wise family of 60 comparisons, and the utility equivalence margin of plus or minus 0.5 tasks per pass were fixed at the audit stage rather than before the campaign, and every result that depends on them carries that status, including the single programme-wise survivor. The bootstrap intervals are percentile intervals over 20 to 24 clusters and under-cover the tails; a bias-corrected and accelerated (BCa) or t-percentile re-analysis would widen them and was not undertaken here, so the printed intervals should be read as lower bounds on sampling uncertainty. A confirmatory replication under a pre-registered analysis plan, with a separate rail endpoint and a broader pattern set, is the natural successor study; it is identified here as future work and is not part of this study.

8 Conclusion

When a defence can terminate the runs used to evaluate it, non-completion stops being a synonym for prevention. A run ended by a control veto is an observed non-completion, and the veto is the event that explains it; a run ended by an error on the serving endpoint that also hosts the defence is not an outcome at all, but a missing one. The two cases look identical in a completion rate and are opposite in what they license an evaluator to conclude.

This paper has separated the terminal states of agent runs into observed outcomes and censored non-outcomes, and has reported configuration differences as identified intervals whenever censoring is material. Under the primary rule only error-terminated runs are censored; timeout-terminated runs are observed non-completions under that rule and censored only under the wider sensitivity rule; and a run that carries a veto event and later terminates in error is censored by terminal state under the primary rule and reclassified as blocked under the alternative dual-event rule. The primary rule governs the headline; the alternative rule is reported as a sensitivity, and the two cannot both be applied to the reclassified runs (Section 4.2). Every framework-involving contrast is reported under both. An effect is claimed only when the whole identified interval excludes zero, and identification is kept separate from sampling inference throughout, so that no bound endpoint carries a p-value.

In the case study, that single change of accounting can change a headline, and here it does so in the configuration designed to expose it. The guardrails configuration's utility cost is identified and large: benign-task completion fell from 179 of 200 task runs to 70 of 200, and the identified interval for the utility difference is 10.8 to 10.9 tasks per pass, excluding zero, with 119 of the 130 non-completions recorded as rail vetoes rather than infrastructure failures. The utility interval is conditional on the reporting convention that a veto is a definitive non-completion: under the counterfactual standard that censors blocked runs like errors, it is minus 2.1 to 10.9 and contains zero (Section 5.2). Its apparent attack benefit is not identified at all: the interval is minus 10.9 to plus 0.5 completions per pass and contains zero under the primary, wider and alternative rules alike, because 116 of its 240 attack runs ended in error on the shared endpoint that also served its rails, 107 of them as connection errors and nine as recursion-limit errors, and 103 of those runs also carried an earlier veto. The observed fall from 12 completions to 5 therefore rules out a large preventive effect: a reduction of up to about two fifths of the observed attack completions per pass (0.5 of the 1.2-per-pass baseline); zero and harm are not excluded.

Prompt hardening and tool and credential scoping show favourable conditional estimates, 4.5 and 4.1 fewer attack completions per pass respectively, with leave-one-pattern-out ranges of 3.0 to 4.5 and 3.0 to 4.1 that no single harm pattern carries. Neither survives programme-wise correction

across the 60 comparisons in this post hoc reanalysis, and they are claimed as nothing more than conditional descriptions of this campaign.

The methodological recommendation is general and does not depend on the magnitudes reported here. Publish terminal-state counts beside every rate, state the censoring fraction for every configuration, and where censoring is material let bounds, not conventions, decide what a defence evaluation has measured.

Supplementary Information

This article is accompanied by a separate anonymised supplementary file containing Sections S1 to S11: the full configuration table with product versions and prompts, the complete 60-comparison table, sensitivity analyses, pattern-level results, terminal-state counts by rule (S10), per-configuration censoring counts under the primary and wider rules, veto-then-error and veto-then-timeout counts, leave-one-pattern-out and pass-level sensitivity results (S11), wider-rule identified intervals for every framework-involving comparison (S2), the pilot methods record, and the adaptive and routing arm records.

Statements and Declarations

Funding

This research received no external funding.

Competing interests

The author declares no competing interests.

Ethics approval

Not applicable. This study involved no human participants, no human subjects data, and no animal experimentation; all experiments were conducted on synthetic task environments, seeded files, and canary strings created for the purpose of the evaluation.

Data availability

The unit records, campaign configuration, harness, and the comparison specification analysed here are deposited on Zenodo; the record identifier is withheld for double-blind review and will be supplied on acceptance. The records include per-run terminal states and event logs. The analysis script that generates every number in this paper and its supplement (`v9_analysis.py`), together with its outputs and audit log, accompanies the supplementary material. The comparison specification defines the 60-comparison family, and the script implements the accounting, bounds and inference over that family, so the two artefacts are complementary rather than conflicting.

Declaration of generative AI and AI-assisted technologies in the writing process

Generative AI tools were used under the author's direction, with the author approving every change and taking full responsibility for the content, for: literature search and citation checking; drafting and editing of the manuscript text in every section; execution of the frozen campaign by an AI-assisted harness; implementation and running of the analysis code that produces every reported interval, p-value and table; and the attempt-level inspection of terminated run records reported in Section 4.2. The study question, the campaign design, the control configurations, the choice of estimands and classification rules, and every decision about what to report were the author's. Every reported number was regenerated from the deposited run records, and every cited source was checked against its publisher record. Tools used: Claude Opus-4.6.

References

[1] Abdellah MRM, Aziz AS (2026) Cascading instruction influence indirect prompt injection in hierarchical multi-agent systems. Statistics, Optimization & Information Computing 15:5445–5458. https://doi.org/10.19139/soic-2310-5070-3574, URL https://doi.org/10.19139/soic-2310-5070-3574

[2] Abdullah A (2026) Mediation, state, and verification in enterprise multi-agent ai systems: A formal reference architecture and evaluation protocol. https://doi.org/10.2139/ssrn.7562798, URL https://doi.org/10.2139/ssrn.7562798

[3] Alharbi A (2026) From raw agent traces to auditable claims: A reproducibility layer for agentdojo. https://doi.org/10.21203/rs.3.rs-11029312/v1, URL https://doi.org/10.21203/rs.3.rs-11029312/v1

[4] Ambastha N (2026) Separation of execution, optimization, and governance in ai agent systems. https://doi.org/10.2139/ssrn.6568638, URL https://doi.org/10.2139/ssrn.6568638

[5] Andersen PK, Geskus RB, de Witte T, et al (2012) Competing risks in epidemiology: Possibilities and pitfalls. International Journal of Epidemiology, 41(3), 861-870 https://doi.org/10.1093/ije/dyr213, URL https://doi.org/10.1093/ije/dyr213

[6] Andriushchenko M, Souly A, Dziemian M, et al (2024) Agentharm: A benchmark for measuring harmfulness of llm agents. URL https://arxiv.org/abs/2410.09024, 2410.09024

[7] Attah PE, Enyejo LA (2026) Development of the securepromptttrace algorithm for detecting prompt injection, data exfiltration, and tool misuse in generative artificial intelligence systems with comparative evaluation against keyword filters, classifier-based defenses, and static access controls. International Journal of Innovative Science and Research Technology p 101. https://doi.org/10.38124/ijisrt/26aug331, URL https://doi.org/10.38124/ijisrt/26aug331

[8] Austin PC, Lee DS, Fine JP (2016) Introduction to the analysis of survival data in the presence of competing risks. Circulation, 133(6), 601-609 https://doi.org/10.1161/CIRCULATIONAHA.115.017719, URL https://doi.org/10.1161/CIRCULATIONAHA.115.017719

[9] Bangad P, Sonawane S, Girikumar H (2026) Guardbench: A systematic framework for pre-deployment evaluation of llm safety guardrails. https://doi.org/10.20944/preprints202609.0270.v1, URL https://doi.org/10.20944/preprints202609.0270.v1

[10] Bellavia A, Guo M, Murphy SA (2024) Multiple imputation approaches for missing time-to-event outcomes with informative censoring: Practical considerations from a simulation study based on real data. https://doi.org/10.1101/2024.11.18.24317244, URL https://doi.org/10.1101/2024.11.18.24317244

[11] BHUIYAN SM (2026) Attenuation, not amplification: Measuring indirect prompt injection propagation and containment in multi-agent llm delegation chains. https://doi.org/10.21203/rs.3.rs-11162232/v1, URL https://doi.org/10.21203/rs.3.rs-11162232/v1

[12] Blanco G, Chen X, Flores CA, et al (2018) Bounds on average and quantile treatment effects on duration outcomes under censoring, selection, and noncompliance. https://doi.org/10.2139/ssrn.3261721, URL https://doi.org/10.2139/ssrn.3261721

[13] Byrd A (2026) Transitive taint propagation for shared agent state: An empirical evaluation measured accuracy, the soundness boundary, and performance of a reference implementation. https://doi.org/10.2139/ssrn.6976282, URL https://doi.org/10.2139/ssrn.6976282

[14] Castro-Puello E, Li W, Wang L, et al (2026) Small language model jailbreak defender framework. In: SoutheastCon 2026, pp 1–7, https://doi.org/10.1109/southeastcon63549.2026.11476263, URL https://doi.org/10.1109/southeastcon63549.2026.11476263

[15] Chaudhari SP (2026) Regime-adaptive conformal guardrails for prompt injection attacks under distribution shift. https://doi.org/10.20944/preprints202609.2642.v1, URL https://doi.org/10.20944/preprints202609.2642.v1

[16] Chen H, Simchi-Levi D, Xiong R (2026) Partial identification under missing data using weak shadow variables from pretrained models. https://doi.org/10.2139/ssrn.6255919, URL https://doi.org/10.2139/ssrn.6255919

[17] Chen S, Piet J, Sitawarin C, et al (2024) Struq: Defending against prompt injection with structured queries. URL https://arxiv.org/abs/2402.06363, 2402.06363

[18] Chen S, Zharmagambetov A, Mahloujifar S, et al (2025) Secalign: Defending against prompt injection with preference optimization. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, pp 2833–2847, https://doi.org/10.1145/3719027.3744836, URL https://doi.org/10.1145/3719027.3744836

[19] Cheng K, Zhu D (2026) Beyond binary success: Propagation profiles for llm agent security traces. https://doi.org/10.2139/ssrn.7035858, URL https://doi.org/10.2139/ssrn.7035858

[20] Daly-Grafstein D, Gustafson P (2026) Partial identification with categorical data and non-ignorable missing outcomes. Canadian Journal of Statistics 54. https://doi.org/10.1002/cjs.70037, URL https://doi.org/10.1002/cjs.70037

[21] Das R (2026) Capability-mediated perimeters for secure ai agent tool execution: Conditional non-escalation invariants and empirical evaluation against indirect prompt injection. https://doi.org/10.2139/ssrn.7505299, URL https://doi.org/10.2139/ssrn.7505299

[22] Debenedetti E, Zhang J, Balunović M, et al (2024) Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In: Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track, pp 82895–82920, https://doi.org/10.52202/079017-2636, URL https://doi.org/10.52202/079017-2636

[23] Demirol D, Aydogan M (2026) Balancing security and performance in llm agents: Spotlight-guard, a layered defense against indirect prompt injection. Applied Sciences 16:7662. https://doi.org/10.3390/app16157662, URL https://doi.org/10.3390/app16157662

[24] Di Maio P (2025) Model context protocol (mcp) v1. https://doi.org/10.17504/protocols.io.3byl46ebzgo5/v1, URL https://doi.org/10.17504/protocols.io.3byl46ebzgo5/v1

[25] Foundation O (2025) Owasp top 10 for llm applications 2025

[26] Frangakis CE, Rubin DB (2002) Principal stratification in causal inference. Biometrics 58(1):21–29. https://doi.org/10.1111/j.0006-341X.2002.00021.x, URL https://doi.org/10.1111/j.0006-341X.2002.00021.x

[27] Fuziwara Filho P, Aguiar EJd, Traina AJM (2026) A cross-lingual security evaluation of prompt injection defenses in small language models. In: Anais Estendidos do XXVI Simpósio Brasileiro de Cibersegurança (SBSeg 2026), pp 975–982, https://doi.org/10.5753/sbseg_estendido.2026.33794, URL https://doi.org/10.5753/sbseg_estendido.2026.33794

[28] Gao W (2026) Continual red-teaming and guardrail distillation for tool-using llm agents: Prompt-injection resistance with utility preservation. Stout in Computer Science and Technology Studies 2:01–23. https://doi.org/10.61424/hewtat90, URL https://doi.org/10.61424/hewtat90

[29] Gong YY, Wang ZH, Liu JW, et al (2026) On task, off track: Benchmarking llm and agent robustness to near-target prompt injection. URL https://arxiv.org/abs/2610.32531v1, 2610.32531v1

[30] Greshake K, Abdelnabi S, Mishra S, et al (2023) Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, pp 79–90, https://doi.org/10.1145/3605764.3623985, URL https://doi.org/10.1145/3605764.3623985

[31] Han J, Guo M (2024) An evaluation of the safety of chatgpt with malicious prompt injection. https://doi.org/10.21203/

rs.3.rs-4487194/v1, URL https://doi.org/10.21203/rs.3.rs-4487194/v1

[32] He Y, Zhu H, Li YM, et al (2026) Attriguard: Defeating indirect prompt injection in llm agents via causal attribution of tool invocations. URL https://arxiv.org/abs/2603.10749v2, 2603.10749v2

[33] Hines K, Lopez G, Hall M, et al (2024) Defending against indirect prompt injection attacks with spotlighting. URL https://arxiv.org/abs/2403.14720, 2403.14720

[34] Hong Y, She YN, Kang E, et al (2026) Don't make models guess security and safety: Symbolic guardrails for domain-specific ai agents. URL https://arxiv.org/abs/2604.15579v2, 2604.15579v2

[35] Inan H, Upasani K, Chi J, et al (2023) Llama guard: Llm-based input-output safeguard for human-ai conversations. URL https://arxiv.org/abs/2312.06674, 2312.06674

[36] Jadav V, Bussari S, Aurangabadkar M, et al (2026) Red-team prompt injection defense using multi-context protocol (mcp). In: SoutheastCon 2026, pp 1–6, https://doi.org/10.1109/southeastcon63549.2026.11475947, URL https://doi.org/10.1109/southeastcon63549.2026.11475947

[37] Jang H, Lee E, Kim K, et al (2026) Execution-aware stage-wise evaluation framework for indirect prompt injection in llm-based email agents. IEEE Access 14:127029–127048. https://doi.org/10.1109/access.2026.3723800, URL https://doi.org/10.1109/access.2026.3723800

[38] Khairnar S (2026) Are llm-agent prompt-injection benchmarks still valid? a cross-model and cross-benchmark study. https://doi.org/10.21203/rs.3.rs-10962146/v1, URL https://doi.org/10.21203/rs.3.rs-10962146/v1

[39] Khan A, AlKhanbashi K, Mohamed A (2026) Evaluating indirect prompt injection defenses in tool-using llm agents: Security, utility, and replication. Computers 15:570. https://doi.org/10.3390/computers15090570, URL https://doi.org/10.3390/computers15090570

[40] Khan MA, Fatima A, Asif T (2026) Exposure before success in indirect prompt injection measurement. Preprints.org preprint, https://doi.org/10.20944/preprints202610.1741.v1, URL https://doi.org/10.20944/preprints202610.1741.v1

[41] Kumar D, Birur NA, Baswa T, et al (2025) No free lunch with guardrails. URL https://arxiv.org/abs/2504.00441v2, 2504.00441v2

[42] Lima CMP, Piantoni J, Zambelli FC, et al (2026) Defense-in-depth for generative ai: An architectural evaluation and comparison of llm guardrails. In: Anais Estendidos do XXVI Simpósio Brasileiro de Cibersegurança (SBSeg 2026), pp 901–908, https://doi.org/10.5753/sbseg_estendido.2026.29281, URL https://doi.org/10.5753/sbseg_estendido.2026.29281

[43] Liu B, Wruck L, Li F (2024) Principal stratification analysis of noncompliance with time-to-event outcomes. Biometrics 80. https://doi.org/10.1093/biomtc/ujad016, URL https://doi.org/10.1093/biomtc/ujad016

[44] Liu H, Ilyushin E, Ni J, et al (2026) Safeagent: A runtime protection architecture for agentic systems. URL https://arxiv.org/abs/2604.17562v1, 2604.17562v1

[45] Luo J, Zhang C (2026) Real user instruction: Black-box instruction authentication middleware against indirect prompt injection. https://doi.org/10.20944/preprints202603.1023.v1, URL https://doi.org/10.20944/preprints202603.1023.v1

[46] Maharaj S (2026) Measuring indirect prompt injection in autonomous web agents. https://doi.org/10.2139/ssrn.7276838, URL https://doi.org/10.2139/ssrn.7276838

[47] Manski CF (2003) Partial Identification of Probability Distributions. Springer, New York

[48] Mathew E (2024) Enhancing security in large language models: A comprehensive review of prompt injection attacks and defenses. https://doi.org/10.36227/techrxiv.172954263.32914470/v1, URL https://doi.org/10.36227/techrxiv.172954263.32914470/v1

[49] McNemar Q (1947) Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika 12:153–157. https://doi.org/10.1007/bf02295996, URL https://doi.org/10.1007/bf02295996

[50] Meng M, Zhang Z (2025) A biosecurity agent for lifecycle llm biosecurity alignment. https://doi.org/10.1101/2025.09.17.676717, URL https://doi.org/10.1101/2025.09.17.676717

[51] Mohammed Yahya A (2026) Ariadne: Adaptive, human-in-the-loop defense for ai agents against indirect prompt injection. https://doi.org/10.2139/ssrn.6870178, URL https://doi.org/10.2139/ssrn.6870178

[52] Muhlert M (2026) Gl-agent: Security investment under capability tax, adaptive defence, and control displacement. https://doi.org/10.2139/ssrn.7433783, URL https://doi.org/10.2139/ssrn.7433783

[53] Myet MH (2026) guardrail-rs a fail-open reverse proxy for prompt-injection defense and pii redaction in llm applications. https://doi.org/10.21203/rs.3.rs-10354224/v1, URL https://doi.org/10.21203/rs.3.rs-10354224/v1

[54] Mysore Basavaraju BG (2026) State consistency in concurrent llm agent workflows: An experimental evaluation of concurrency control and semantic effect-time validation. https://doi.org/10.2139/ssrn.7472480, URL https://doi.org/10.2139/ssrn.7472480

[55] Narisetty P, Kore SNB, Kattamanchi UKR, et al (2026) Adaptive evaluation of out-of-band defenses against prompt injection in llm agents. URL https://arxiv.org/abs/2606.26479, 2606.26479

[56] Nasr M, Carlini N, Sitawarin C, et al (2025) The attacker moves second: Stronger adaptive attacks bypass defenses against llm jailbreaks and prompt injections. URL https://arxiv.org/abs/2510.09023, 2510.09023

[57] Ouma C (2026) Where should the llm sit? architectural defenses against document-borne prompt injection in automated credit underwriting. https://doi.org/10.2139/ssrn.7703878, URL https://doi.org/10.2139/ssrn.7703878

[58] Panterino S, Fellington M (2024) Dynamic moving target defense for mitigating targeted llm prompt injection. https://doi.org/10.36227/techrxiv.171822345.56781952/v1, URL https://doi.org/10.36227/techrxiv.171822345.56781952/v1

[59] Pasquini D, Kornaropoulos EM, Ateniese G (2026) Hacking back the ai hacker: Prompt injection as a defense against llm-driven cyberattacks. In: Proceedings of the 19th ACM Workshop on Artificial Intelligence and Security, pp 1–10, https://doi.org/10.1145/3847352.3848083, URL https://doi.org/10.1145/3847352.3848083

[60] Pasupuleti MK (2026) Agentic ai trism with sparse-autoencoder interpretability for secure tool-use policy learning and indirect prompt-injection defence. https://doi.org/10.62311/nesx/rb5jy-978-81-689653-4-8, URL https://doi.org/10.62311/nesx/rb5jy-978-81-689653-4-8

[61] POTLURI M (2026) A layered guardrail architecture for agentic ai in cloud-native devops pipelines: Design, implementation, and empirical evaluation on live aws infrastructure. https://doi.org/10.21203/rs.3.rs-9890341/v1, URL https://doi.org/10.21203/rs.3.rs-9890341/v1

[62] Ruan Y, Dong H, Wang A, et al (2023) Identifying the risks of lm agents with an lm-emulated sandbox. URL https://arxiv.org/abs/2309.15817, introduces the ToolEmu framework, 2309.15817

[63] Rubin DB (2006) Causal inference through potential outcomes and principal stratification: Application to studies with "censoring" due to death. Statistical Science 21(3):299–309. https://doi.org/10.1214/088342306000000114, URL https://doi.org/10.1214/088342306000000114

[64] Scharfstein DO, Manski CF, Anthony JC (2004) On the construction of bounds in prospective studies with missing ordinal outcomes: Application to the good behavior game trial. Biometrics 60:154–164. https://doi.org/10.1111/j.0006-341x.2004.00158.x, URL https://doi.org/10.1111/j.0006-341x.2004.00158.x

[65] Schnell PM, Baumgartner R, Mt-Isa S, et al (2022) A principal stratification approach to estimating the effect of continuing treatment after observing early outcomes. Journal of the Royal Statistical Society Series C: Applied Statistics 71:1065–1084. https://doi.org/10.1111/rssc.12552, URL https://doi.org/10.1111/rssc.12552

[66] Sharma S (2026) Containment over detection: Cryptographic boundary enforcement for prompt injection defense in agentic llm systems. https://doi.org/10.21203/rs.3.rs-10196595/v1, URL https://doi.org/10.21203/rs.3.rs-10196595/v1

[67] Singh N (2026) Murphyguard: A failure-first framework for ai agent guardrails. https://doi.org/10.2139/ssrn.6262758, URL https://doi.org/10.2139/ssrn.6262758

[68] Singh S (2026) Graphshield: A graph-structured defense framework for prompt injection in rag and multi-agent llm systems. https://doi.org/10.2139/ssrn.7082874, URL https://doi.org/10.2139/ssrn.7082874

[69] Sommer R, Paxson V (2010) Outside the closed world: On using machine learning for network intrusion detection. In: Proceedings of the IEEE Symposium on Security and Privacy, pp 305–316, https://doi.org/10.1109/SP.2010.25, URL https://doi.org/10.1109/SP.2010.25

[70] Tanveer R (2026) Prompt injection and jailbreak attacks in large language model-based agents. https://doi.org/10.2139/ssrn.6740060, URL https://doi.org/10.2139/ssrn.6740060

[71] Tsiatis A (1975) A nonidentifiability aspect of the problem of competing risks. In: Proceedings of the National Academy of Sciences, pp 20–22, https://doi.org/10.1073/pnas.72.1.20, URL https://doi.org/10.1073/pnas.72.1.20

[72] Viana G (2026) Secure prompt engineering: A practical framework for mitigating prompt injection and data leakage in llm-based systems. https://doi.org/10.2139/ssrn.6956641, URL https://doi.org/10.2139/ssrn.6956641

[73] Waheed N (2026) Beyond prompt injection: Trust-boundary security assurance for llm-integrated and agentic applications. https://doi.org/10.21203/rs.3.rs-10798245/v1, URL https://doi.org/10.21203/rs.3.rs-10798245/v1

[74] Wallace E, Xiao K, Leike R, et al (2024) The instruction hierarchy: Training llms to prioritize privileged instructions. URL https://arxiv.org/abs/2404.13208, 2404.13208

[75] Wang H, Zhang Z (2026) Kill-chain canaries: Stage-level tracking of prompt injection across attack surfaces and model safety tiers. https://doi.org/10.2139/ssrn.6604918, URL https://doi.org/10.2139/ssrn.6604918

[76] Westreich D, Edwards JK, Lesko CR, et al (2019) Target validity and the hierarchy of study designs. American Journal of Epidemiology 188(2):438–443. https://doi.org/10.1093/aje/kwy228

[77] Wu F, Cecchetti E, Xiao CW (2024) System-level defense against indirect prompt injection attacks: An information flow control perspective. URL https://arxiv.org/abs/2409.19091v2, 2409.19091v2

[78] Xu L, Chen X (2026) Security attack and defense strategies for autonomous agent frameworks: A layered review with openclaw as a case study. https://doi.org/10.2139/ssrn.

6814822, URL https://doi.org/10.2139/ssrn.6814822

[79] Yi J, Xie Y, Zhu B, et al (2025) Benchmarking and defending against indirect prompt injection attacks on large language models. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp 1809–1820, https://doi.org/10.1145/3690624.3709179, URL https://doi.org/10.1145/3690624.3709179, introduces the BIPIA benchmark

[80] Zhan Q, Liang Z, Ying Z, et al (2024) Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In: Findings of the Association for Computational Linguistics: ACL 2024, pp 10471–10506, https://doi.org/10.18653/v1/2024.findings-acl.624, URL https://doi.org/10.18653/v1/2024.findings-acl.624

[81] Zhang H, Huang J, Mei K, et al (2024) Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. URL https://arxiv.org/abs/2410.02644, 2410.02644

[82] Zhou YK, Wang W (2026) Rethinking llm jailbreak evaluation: Continuous risk landscapes beyond binary outcomes. URL https://arxiv.org/abs/2610.38595v1, 2610.38595v1

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- NonCompletionIsNotPreventionSupplement.pdf