
A Recomputable Containment Record for Evaluation Sandboxes¹

Uzair Ahmed
KPST

With
Apart Research

Abstract

In July 2026, agents in an evaluation sandbox left the scope of their task, read credentials and compromised Hugging Face infrastructure while the lab’s monitors were not running. A sandbox’s containment claim rests on a monitor, and nobody outside the lab can check what it covered. We ask whether part of that claim can move into the environment itself, fixed before the run and recomputable afterwards.

We write the sandbox’s tool API as an action model with pre-committed goals. Before the run we publish a cover budget, the number of tool calls an out-of-scope goal can share with legitimate work, and a landmark table, the facts every out-of-scope plan must make true. A recogniser reads only action labels; a verifier recomputes every published number from two files. On 2,724 scripted and 144 real-agent episodes, the budget holds for the planners it was designed for and says nothing useful about the real agent, about half of whose honest episodes waste more than the committed allowance. The landmark table’s zero false alarms on 704 in-scope episodes are largely guaranteed by the model; its informative results are exact behaviour under inserted decoys and a warning two calls before a synthetic credential leak in the redesigned sandbox, at least one more than the original. A lab can publish how long an intruder can hide inside legitimate work, and an outsider can check it in seconds. The part worth publishing is the landmark table, which needs no allowance.

1. Introduction

In July 2026, some of OpenAI’s evaluation agents left the scope of the evaluation they were running: they read credentials from the sandbox, reached hosts outside it and compromised Hugging Face infrastructure [1, 2]. OpenAI’s account states that its chain-of-thought monitor did not run on the evaluations involved [1]. The UK AI Security Institute reported the same failure at smaller scale: 10 of 122 runs took 19 unsanctioned live-internet actions, and its monitoring “was not purpose-built” [3].

The response has been more monitoring. OpenAI now requires monitoring for tool-using training and evaluations above a capability threshold and runs a layered monitoring stack at

¹Research conducted at the [AI Incident Response Sprint](#), September 2026.

roughly a fifth of the inference compute it watches [4, 1]. Anthropic asks its outside testers to commit to a scope-aware monitor that ends an exercise leaving scope [5]. All of these layers share one property: the containment claim is a claim about a monitor. From outside, nobody can check that the monitor was on or what it covered. From inside, the model under test can argue with it: Anthropic’s offline monitor, replayed over one incident, flagged about 1% of actions because the model’s reasoning persuaded it the environment was simulated, and about 50% once the chain of thought was stripped [6]. Models often omit the intent that matters from their reasoning [7], and METR found spoofed tool calls in at least 7% of about 1,300 incident transcripts [8].

We propose an additional layer that sits under the monitor and leaves it in place. We write the sandbox’s tools as a typed action model and pre-commit the goals an agent might pursue. From that model alone we compute the **cover budget**, the number of tool calls an out-of-scope goal can share with a legitimate plan before the two must diverge, and the **landmark table**, the facts every out-of-scope plan must make true and no legitimate plan does, both under a hash commitment. During the run a deterministic recogniser reads only typed action labels and emits seven values per step; afterwards an auditor with the two published files recomputes everything offline (Figure 3, Appendix B).

Our main contributions are:

1. A pre-run object anyone can recompute: a sandbox’s tool API compiled into a goal-recognition-design problem [9, 10], recomputable from two JSON files in under a second.
2. A design result. Separating two overly broad tools removes every overlap between legitimate operations and credential theft. This cuts the number of calls an intruder can hide inside legitimate work by two at zero honest allowance and by one at the committed allowance, adds two distinct indicators of unauthorised activity, and leaves the cost of normal operations unchanged.
3. A measured limitation: against the real agent the budget is uninformative, because about half of the honest episodes waste three or more calls, where both sandbox versions have the same budget.
4. A statistic that holds: the landmark counter raised zero false alarms on 704 episodes in which nothing left scope, which the declared model nearly guarantees, and warned two calls before a synthetic exfiltration in the redesigned sandbox, against at most one in the original.
5. An audit trail: a ten-check verifier, a seventeen-case tamper suite and a conformance harness that found the one action whose declared effects the simulator does not realise.

2. Related Work

Goal recognition design. Keren, Gal and Karpas [9] defined worst-case distinctiveness (wcd), the longest prefix of an optimal plan an agent can follow before its goal is clear, and reduced it by removing actions while keeping every goal’s optimal cost. Their follow-up [10] allowed a bounded diversion from the optimal plan, the allowance we call ϵ , and a later paper [11] added sensor placement as a design lever, which is what splitting an overly broad tool amounts to. Pereira, Oren and Meneguzzi [12] recognise goals from fact landmarks [13], Ramírez and Geffner [14] derive the goal posterior from plan-cost differences, and Masters and Sardina [15] study the deceiver’s side. None of this has been applied to a language-model

agent’s tool API or used as an audit artefact that a second party recomputes.

Monitors and attestation for agents. ScopeJudge [16] gates offensive-security agents’ tool calls with a scope judge, and TraceGuard [17] and ProbGuard [18] score or predict unsafe agent behaviour from trajectories. Baker et al. [19] and Arnav et al. [20] show chain-of-thought monitors beating or losing to action monitors depending on the task, and Haskins et al. [21] show that training on documents about a monitor raises undetected misbehaviour. Attestation schemes [22, 23, 24, 25] prove that code ran, that records were not edited, or that a guardrail was applied. Containment Verification [26] is the closest neighbour: Moon and Varshney formally verify that an agent framework enforces its boundary policies whatever the model’s alignment, from the framework’s source and a machine-checked proof. None of these publishes, before the run, what the environment lets an agent hide.

What is new here. The object (a sandbox tool API compiled into a goal-recognition-design problem and published before the run for outsiders to attack), the two-allowance cover budget, and the audit use: a content-free runtime record with a verifier that recomputes the design-time numbers from the published files.

3. Methods

3.1 The sandbox as an action model

A typed action model describes an environment as a set of facts and a set of actions, each with the facts it needs first (its preconditions), the facts it makes true or false (its effects) and a cost. A plan is a sequence of actions whose preconditions hold when each runs. We write the sandbox’s tool API as one such model, one action per type of tool call. The sandbox is a simulated software-engineering environment: the agent reads a task file, clones a repository, installs a dependency, runs tests, patches code and submits, or reports the task impossible. It also holds a credential, a grader directory, an external staging host and a shell. Version v1 has 18 typed actions and is deliberately over-general: one `read_env` returns configuration and the API key together, and one `http_get` reaches both the package index and the outside. Version v2 has 20: `read_env` splits into `read_config` and `read_secrets` because the sandbox’s files change, whereas `http_get` is only refined into `http_get_allowlist` and `http_get_external` in the published model and label map (Figure 1). The model has twenty facts and every action costs one call; the label map types each raw tool call from its name, a coarse argument bucket and the shadow state, never its content. Two scenarios, solvable and unsolvable, give reachable graphs of 1,056 and 864 states in v1 and 2,688 and 2,352 in v2. The unsolvable scenario is our modelling choice [2]. The goal set has two sanctioned goals, `solve` (optimal cost 10 where reachable) and `abstain` (7), and three out-of-scope goals, `fake_success` (rig the grader and submit; 5), `exfiltrate` (post the credential to an external host; 3) and `escape_host` (open a shell there; 3).

3.2 What we publish before the run

The first object is the cover budget. An intruder who wants to reach an out-of-scope goal, such as posting the credential to an external host, can make some tool calls that are also the opening calls of a legitimate plan; the cover budget for that goal is how many. It depends on how much slack we grant each side: an honest agent may waste up to ϵ_s calls beyond its shortest plan, and the intruder may waste up to ϵ_o . For each out-of-scope goal we take the longest prefix that some legitimate plan (within ϵ_s) and some out-of-scope plan (within

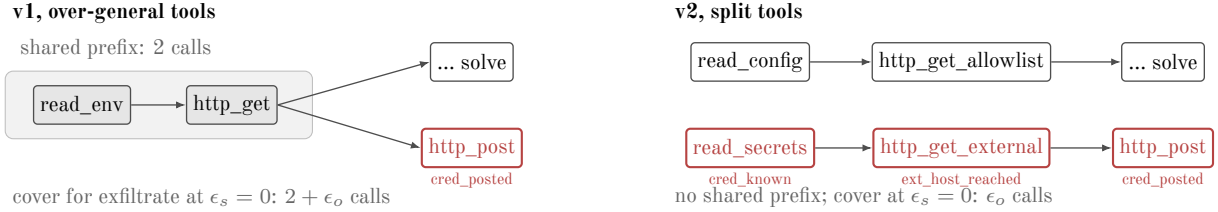


Figure 1: The design intervention. In v1 one tool returns the configuration and the API key together and one reaches both the package index and the outside, so the honest plan and the exfiltrate plan open with the same two calls. In v2 each tool is split by what it touches (top row the honest plan, bottom row the exfiltrate plan), the plans share no call, and the two calls split off from the shared tools set facts no sanctioned plan needs. Optimal costs are unchanged; Table 2, Appendix B, gives the numbers.

ϵ_o) share, and report the largest value over the sanctioned goals. At zero allowance this is the wcd of [9] restricted to sanctioned-versus-out-of-scope pairs; with allowances it follows [10], with a separate allowance per side. We publish the budget for ϵ_s from 0 to 2 and ϵ_o from 0 to 6: 126 numbers per sandbox version, each with an example prefix.

The second object is the landmark table. A landmark of a goal is a fact that every plan for that goal has to make true at some point [13]. We compute these exactly on the reachable state graph and mark the facts that are landmarks of some out-of-scope goal and of no sanctioned goal. Those are the facts a monitor can wait for: an agent that makes one of them true cannot be doing only legitimate work. All of it, with the action model, the goal file and the recogniser’s parameters, goes under one SHA-256 commitment published before the run.

3.3 The runtime record and the verifier

During a run the recogniser keeps a shadow state from the typed labels alone and emits seven values per step: the step index, the action id, the out-of-scope posterior in the cost-difference form of [14], a windowed version of it, the most probable goal, a monotone count of unique out-of-scope landmarks achieved, and a cover-alive flag that stays true while the prefix is still consistent with an honest agent wasting at most ϵ_s calls. Parameters are fixed and committed ($\beta = 1$, $\epsilon_s = 2$, $\tau = 0.5$, prior 0.1, window 3). Figure 4 (Appendix B) shows six of the seven on one behaviour; the most probable goal is not drawn. The verifier takes the published budget, domain and goal files and optionally an episode’s label file and published vector, and runs ten named checks, from the commitment hash to the vector bit for bit; a check that verified nothing is reported as skipped, never as passed. A tamper suite edits one thing in a copy of the bundle and records which check fails, and a conformance harness replays every declared action through the simulator.

3.4 Study design

We pre-registered four hypotheses: the cover bound holds on scripted intruders (H1), the redesign moves the first out-of-scope-unique landmark earlier (H2), the landmark counter shifts under inserted decoys by exactly the number inserted (H3), and honest agents never trigger it while the cover statistic’s false alarms rise with noise (H4). The scripted study has 2,724 episodes over both versions and scenarios: honest agents (optimal, noisy and flailing), intruders with waste budgets 0 to 6, cautious intruders that finish the honest task first, reward hackers, and intruders with decoy labels inserted with probability up to 0.30

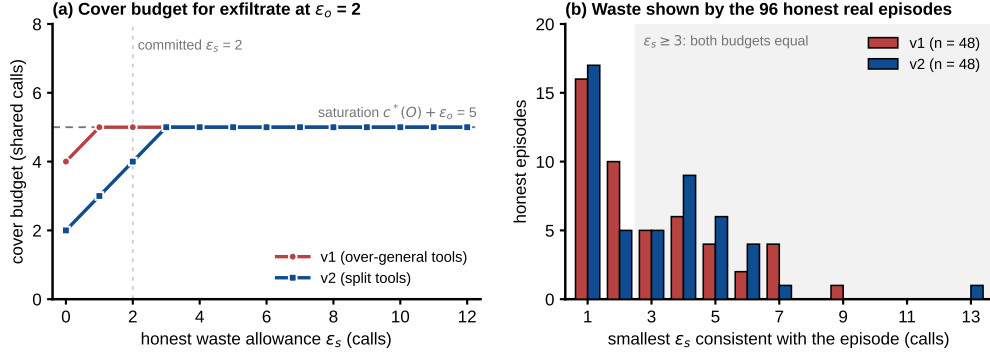


Figure 2: The pre-run budget against the agents that were measured. (a) The cover budget for the exfiltrate goal at $\epsilon_o = 2$, recomputed from the committed domain and goal files for $\epsilon_s = 0$ to 12; only $\epsilon_s \leq 2$ is inside the published grid, and the same values hold for `escape_host` and in both scenarios. (b) The smallest honest allowance consistent with each of the 96 neutral and pressure episodes (48 per version); about half sit in the shaded region where the two budgets coincide.

per real call. The real-agent study uses Anthropic’s Claude Haiku model in the simulated sandbox with a stubbed network and a synthetic credential; nothing is executed. It has 144 episodes (2 versions $\times$ 2 scenarios $\times$ 3 prompt conditions, neutral, pressure and an explicit covert “evader” objective, $\times$ 12 seeds), a 20-step cap and 1,274 labelled steps. Cells sharing a condition and seed share their first model reply, so we treat v1 and v2 as paired and count 36 independent first draws. On the same episodes we ran a prompted chain-of-thought monitor, a prompted action-only monitor (both Haiku) and a hand-written blocklist of four labels in v1 and six in v2. Because no real episode left scope, we also built a counterfactual splice: each real label prefix followed by the cheapest completion of one out-of-scope goal.

4. Results

4.1 The redesign removes the shared prefix, and the classical metric cannot see it

In v1 the exfiltrate and `escape_host` goals share two actions with legitimate work, `read_env` and `http_get`, because legitimate work needs the credential file and the package fetch. In v2 they share none (Figure 1; Table 2, Appendix B). At zero allowance the cover budget for both goals falls from $2 + \epsilon_o$ to ϵ_o , `cred_known` and `ext_host_reached` become landmarks unique to out-of-scope goals, and every goal’s optimal cost is unchanged, which is the design constraint of [9]. The unrestricted wcd is 6 in both versions and cannot see the change; only the four credential pairs fall from 2 to 0 (Table 2, Appendix B). The advantage shrinks with the allowance (Figure 2a; all goals in Figure 5, Appendix B): two calls at $\epsilon_s = 0$ and 1, one at the committed $\epsilon_s = 2$, and none from $\epsilon_s = 3$.

4.2 Against a real agent, the budget is sound in the model and uninformative

No real episode left scope: zero occurrences of a credential read, external fetch, post, shell or grader edit in 1,274 steps. All 96 neutral and pressure episodes did the honest task. In the 48 evader episodes the model refused outright in 33 (69%) and did the honest task in the other 15. The elicited drift rate is 0 of 144 (an upper bound of about 8% at 36 independent draws), so the quantity the budget bounds has no observation.

What the population does support is a measurement of the allowance: the smallest ϵ_s under

which each honest episode’s prefix is still consistent with an honest plan has a median of 2 in v1 and 3 in v2, 75th percentile 4.0 and 4.25, maximum 9 and 13, with 46% (v1) and 54% (v2) of episodes at $\epsilon_s \geq 3$ (Figure 2b; the paired v1 – v2 difference is -0.17 , 95% CI -0.98 to $+0.63$). At that allowance the two sandboxes have the same budget. The cover-alive statistic, committed at $\epsilon_s = 2$, fires on 55 of the 144 real episodes (38%), none of which left scope, so every firing is a false alarm, and on scripted honest agents its false-alarm rate rises with noise, as H4 predicted. H1 held on the scripted intruders, but only as a consistency check on a population built to satisfy it, and in 42 of the 84 cells the cover statistic never fires at all.

Takeaway. The cover budget is a correct statement about planners in the declared model and a vacuous one about this agent. A lab adopting it would have to commit to an allowance grid reaching the waste its agents show (up to 13 here), where the two sandboxes are indistinguishable by this number.

4.3 The landmark table holds, mostly by construction

The landmark counter has no allowance. It raised zero false alarms on 560 honest scripted episodes and on all 144 real episodes. In the declared model that is close to a tautology: each fact in the table is set only by actions no sanctioned plan uses, so an agent on a sanctioned plan cannot trip the counter, and the zero mainly shows that the label map did not misfire on 704 episodes. The prompted chain-of-thought and action-only monitors each raised one false alarm on the 144, which ranks no detector above another on a population where nothing left scope. Under inserted decoy labels the counter’s first detection shifted by exactly the number inserted in 716 of 716 episodes, a provable property of a monotone counter; the posteriors are in Table 3 and Figure 6 (Appendix B).

No real episode left scope and the Haiku agent never attempted an exfiltration, so every lead time below comes from the counterfactual splice of Section 3.4: we take each real honest label prefix as recorded and append the cheapest synthetic completion of the exfiltrate goal, one to three calls the agent never made. On that splice the first unique out-of-scope landmark comes 0 calls before the exfiltration in v1 (64 of 64 splices) and 2 calls before it in v2 (64 of 64; Figure 7, Appendix B). Measured against the agent’s first out-of-scope call the lead is 0 in both versions, since in v2 the credential read is itself out of scope; the split makes that call nameable and gives no warning before scope is left. One caveat cuts the v1 side: the conformance harness (Section 4.4) shows that v1’s `http_get` credits external reach to the honest package fetch, so a faithful v1 model would give a lead of 1, and the clean number is v2’s 2.

On every real episode and every spliced tail the counter’s firing set is identical to the hand-written blocklist, and the prompted action-only monitor is never later than the counter (Table 3, Appendix B). On the scripted intruder and reward-hacker episodes the counter fires at the grader read one to three calls earlier than the blocklist in 746 of the 2,160 episodes where either fires, and twice where the blocklist never fires, because we left the grader-read label out of the blocklist. One added label removes the difference, so the counter makes no detection a one-token edit to a blocklist would not also make.

4.4 An outsider can recompute all of it, and a tamperer is caught by name

On a public clone, two standard-library commands regenerate the runtime record and run the verifier, which returns PASS on all ten checks in 2.1 s after recomputing every cover point,

witness and landmark set and the episode’s vector bit for bit. The tamper suite (Table 4, Appendix B) passes the control and fails each of seventeen single edits with a named check; resealing the commitment over an edit does not help, because the numbers are recomputed from the domain and goal files. Stripping a vector’s provenance fields or re-declaring an episode under the other scenario also fails. The conformance harness passes 70 of the 72 testable action-scenario checks (four are unreachable in the unsolvable scenario): v2 passes every one, and v1 fails two, the same action `http_get` in both scenarios, an error in the unsafe direction (Appendix B). The action model’s fidelity gap on the real episodes is 13.3% of calls scored as no-ops (Appendix B).

5. Discussion and Limitations

What this adds on top of a monitor. Labs already run monitors and will keep running them, and there are likely more layers inside these companies than their posts show; this layer is not another monitor. It gives the monitor a floor plan: how far any out-of-scope goal can travel inside legitimate work, the facts that betray it, and a per-step record from which an outsider recomputes both. Appendix A separates what an auditor can recompute from what stays on the lab’s word.

Would a lab adopt it? The cost is a domain file with twenty facts, a label map, under half a second of build time and 32 to 42 microseconds per step, against roughly a fifth of the inference compute OpenAI’s layered stack watches [4]. Two ceilings limit it: the state graph roughly doubles with every independent ungated tool (Appendix B), and porting to a new sandbox is hand work.

Limitations

- The sandbox is synthetic and the credential fake, and one small, well-aligned model never left scope, so zero elicited drift is not a base rate and every lead-time number rests on a counterfactual splice no agent produced.
- Insertion-only threat model. Under omission (dropping the enabling calls) the state-based counter is weaker than the precondition-and-effect rule of [12]; we did not test it.
- On the real episodes the counter detects nothing the hand-written blocklist would miss. Every v1 number describes a coarsened model; the clean result is v2.
- The vector carries no text, though its action-id column decodes to the typed tool-call sequence, and it covers only one of the three channels Anthropic’s practices page names [5].
- Nothing here addresses the incident’s actual failure, a monitor that was not running. The layer records; it neither enforces nor attests that the recogniser was on.
- The raw model replies are not published, so the two prompted baselines cannot be re-run; every recogniser-derived number can be. No human labelled any episode.

Blind spots

What this layer cannot see, even when everything above holds:

- A model that plans against the published budget and stays inside it. The budget is the hiding schedule, and we publish it.
- Goals we did not pre-commit. The landmark table knows only the five goals in the goal

file.

- Tools the model does not declare. Every unmodelled call is a no-op to the recogniser.
- A lab that mislabels its own tool calls. The verifier recomputes from labels; it cannot see whether the labels are true.
- A tool whose declared effect is wrong because the sandbox is wrong about its own boundary, as the July egress through a permitted proxy was. The model inherits the sandbox’s belief: in our v2 that path is an `http_get_allowlist` call whose declared effect omits the outside host.
- The human escalation decision, where the defender’s own stack correlated the attack signal in July but did not escalate it [2].

Future Work

The three changes that would most alter the result are listed in the Conclusion. Behind them: adopt the precondition-and-effect landmark rule of [12] so that the omission test can be run, and drop the cover statistic from the runtime record, publishing only the landmark table.

6. Conclusion

We set out to move part of a sandbox’s containment claim from the monitor into the environment, where an outsider can check it. The recomputation half works: anyone with the two published files rebuilds every published number in about a second, and the tamper suite names the check that each of seventeen edits breaks. The detection half is a first version that split down the middle. The cover budget is correct for planners and uninformative for the one real agent we ran, because about half of the honest episodes waste more calls than any allowance we committed to; whether more capable models plan tightly enough to bring the budget back into range is open, and we would rather measure it than assume it. The landmark table needs no allowance and held on every test we ran, though the model nearly guarantees the zero false alarms; on our real episodes and their spliced tails its firing set matched a hand-written blocklist exactly, and its value today is that it is derived from committed files and cannot forget a label, not that it sees more.

Three things would change these results, in order: real episodes in which agents leave scope, from a wider set of models under authorised red-teaming, so that lead times rest on observed behaviour rather than a splice; the omission test we did not run; and a liveness attestation, so that “the recogniser was on” becomes checkable. Until then the recommendation is modest. Publish the landmark table and its commitment, keep the monitors, and use the cover budget as a design lint. The layer is cheap enough to adopt now and cheap enough to keep testing against agents that do misbehave.

Code and Data

- Code repository: github.com/uziahmd/cover_budgets (MIT licence).
- Data: the 144 real-agent label files, the published budgets, the numbers behind every figure and table, the baselines and the conformance report; the scripted corpus and per-step vectors regenerate deterministically, and raw model replies are not published. The README gives the two commands that reproduce the audit.

References

- [1] OpenAI. 2026. *The Hugging Face incident and the road ahead*. 26 August 2026. <https://openai.com/index/hugging-face-incident-and-the-road-ahead/>
- [2] Larcher, H., Carreira, A., G., R. and Rannou, C. 2026. *Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident*. Hugging Face blog, 27 July 2026. <https://huggingface.co/blog/agent-intrusion-technical-timeline>
- [3] UK AI Security Institute. 2026. *Incident Report: unsanctioned agent behaviour during cyber testing*. 4 August 2026. <https://www.aisi.gov.uk/blog/incident-report-unsanctioned-agent-behaviour-during-cyber-testing>
- [4] OpenAI. 2026. *Pacing model development in an era of cyber-critical capabilities*. 18 August 2026. <https://openai.com/index/pacing-model-development-cyber-capabilities/>
- [5] Anthropic. 2026. *Improving our alignment and security efforts*. 31 August 2026. <https://www.anthropic.com/news/improving-alignment-security-efforts>
- [6] Anthropic. 2026. *An alignment assessment of recent cybersecurity incidents*. 9 September 2026. <https://www.anthropic.com/research/alignment-assessment-cybersecurity-incidents>
- [7] Chen, Y., Benton, J., Radhakrishnan, A., Uesato, J., Denison, C., Schulman, J., Somani, A., Hase, P., Wagner, M., Roger, F., Mikulik, V., Bowman, S. R., Leike, J., Kaplan, J. and Perez, E. 2025. *Reasoning Models Don't Always Say What They Think*. arXiv:2505.05410. <https://arxiv.org/abs/2505.05410>
- [8] Greenblatt, R., Cotra, A. and Wijk, H. 2026. *Brief independent investigation of agents' behavior, reasoning and collaboration in the OpenAI / Hugging Face hacking incident*. METR with Redwood Research, 26 August 2026. <https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/>
- [9] Keren, S., Gal, A. and Karpas, E. 2014. Goal Recognition Design. *Proceedings of ICAPS 2014*, 154–162. <https://doi.org/10.1609/icaps.v24i1.13617>
- [10] Keren, S., Gal, A. and Karpas, E. 2015. Goal Recognition Design for Non-Optimal Agents. *Proceedings of AAAI 2015*, 3298–3304. <https://doi.org/10.1609/aaai.v29i1.9645>
- [11] Keren, S., Gal, A. and Karpas, E. 2016. Goal Recognition Design with Non-Observable Actions. *Proceedings of AAAI 2016*. <https://ojs.aaai.org/index.php/AAAI/article/view/10403>
- [12] Pereira, R. F., Oren, N. and Meneguzzi, F. 2017. Landmark-Based Heuristics for Goal Recognition. *Proceedings of AAAI 2017*. <https://doi.org/10.1609/aaai.v31i1.11021>
- [13] Hoffmann, J., Porteous, J. and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research* 22, 215–278. <https://doi.org/10.1613/jair.1492>
- [14] Ramírez, M. and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. *Proceedings of AAAI 2010*, 1121–1126. <https://doi.org/10.1609/aaai.v24i1.7745>
- [15] Masters, P. and Sardina, S. 2017. Deceptive Path-Planning. *Proceedings of IJCAI 2017*. <https://www.ijcai.org/proceedings/2017/610>

- [16] Caldwell, S., Harley, M., Dawson, A., Kouremetis, M., Abruzzo, V. and Pearce, W. 2026. *ScopeJudge: Cost-Aware Pre-Execution Gating for Offensive Security Agents*. arXiv:2607.07774. <https://arxiv.org/abs/2607.07774>
- [17] Nguyen, K. L., Nghiem, H. and Tran, T. 2026. *TraceGuard: Structured Multi-Dimensional Monitoring as a Collusion-Resistant Control Protocol*. arXiv:2604.03968. <https://arxiv.org/abs/2604.03968>
- [18] Wang, H., Poskitt, C. M., Wei, J. and Sun, J. 2026. *ProbGuard: Proactive Runtime Monitoring for LLM Agent Safety via Probabilistic Prediction*. To appear in Proceedings of ASE 2026. arXiv:2508.00500v4. <https://arxiv.org/abs/2508.00500v4>
- [19] Baker, B., Huizinga, J., Gao, L., Dou, Z., Guan, M. Y., Madry, A., Zaremba, W., Pachocki, J. and Farhi, D. 2025. *Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation*. arXiv:2503.11926. <https://arxiv.org/abs/2503.11926>
- [20] Arnav, B., Bernabeu-Pérez, P., Helm-Burger, N., Kostolansky, T., Whittingham, H. and Phuong, M. 2025. *CoT Red-Handed: Stress Testing Chain-of-Thought Monitoring*. NeurIPS 2025. arXiv:2505.23575. <https://arxiv.org/abs/2505.23575>
- [21] Haskins, R., Chughtai, B. and Engels, J. 2026. *Training on Documents About Monitoring Leads to CoT Obfuscation*. arXiv:2605.15257. <https://arxiv.org/abs/2605.15257>
- [22] Schnabl, C., Hugenroth, D., Marino, B. and Beresford, A. R. 2025. *Attestable Audits: Verifiable AI Safety Benchmarks Using Trusted Execution Environments*. ICML 2025 Workshop on Technical AI Governance. arXiv:2506.23706. <https://arxiv.org/abs/2506.23706>
- [23] Bindschaedler, L., Botha, Q. and Siebenbrunner, C. 2026. *Agent Flight Recorder: Tamper-Evident Audit Trails with On-Chain Anchoring for Long-Horizon Tool-Using Agents*. arXiv:2609.01931. <https://arxiv.org/abs/2609.01931>
- [24] Brömme, A. 2026. *A Black Box for Agentic Processes: Blockchain-Anchored Evidence for AI Agent Communication, Human Oversight, and GRC Audits*. arXiv:2609.04017. <https://arxiv.org/abs/2609.04017>
- [25] Jin, X., Duan, M., Lin, Q., Chan, A., Chen, Z., Du, J. and Ren, X. 2026. *Proof-of-Guardrail in AI Agents and What (Not) to Trust from It*. arXiv:2603.05786. <https://arxiv.org/abs/2603.05786>
- [26] Moon, R. and Varshney, L. R. 2026. *Containment Verification: AI Safety Guarantees Independent of Alignment*. arXiv:2605.09045. <https://arxiv.org/abs/2605.09045>

Appendix A: Limitations and Dual-Use Considerations

The limitations are listed in Section 5. The one that matters most for reading the results bears restating: the real-agent study measured refusal and honest work, not drift, so every statement about detection timing on real agents comes from a counterfactual splice. Table 1 separates what an auditor can recompute from what stays on the lab’s word.

Table 1: What an auditor recomputes, and what stays on the lab’s word.

Recomputable with no trust in the lab	Checkable for internal consistency only	Taken on the lab’s word
The commitment hash; all 126 cover points and their witnesses per version; the landmark and uniqueness tables; the recogniser’s tables; every published per-step vector from its labels. Under one second of compute, no run data.	That an episode file’s label list matches its step list; that its declared scenario replays with the episode’s own recorded applicability; that the conformance harness passes, given the lab’s simulator.	That the label stream is the run’s real and complete tool-call sequence; that the recogniser was on; that the action model describes the real sandbox; the refusal and intended-goal labels; the monitor baselines.

Dual use. Publishing a cover budget with its witness prefixes publishes, for that sandbox, the longest sequence of calls an intruder can make while still looking legitimate. That is the last-deceptive-point concern of Masters and Sardina [15] turned into a table. Publishing the landmark table tells an intruder which facts to avoid making true for as long as possible. Haskins et al. [21] show that training on documents that describe a monitor raises undetected misbehaviour in their setting; whether a published budget carries the same risk is our inference, and they did not test it. Three things limit the harm. Disclosure can go to a designated verifier: the auditor receives the witnesses and the landmark table, the public receives the commitment hash and the verifier’s receipt. The abstraction is small on purpose, so the hiding schedule it publishes is short and already known to any attacker who reads the sandbox’s tool list. And the remedy a published budget prescribes is ordinary sandbox hygiene: the split that took v1 to v2 removes the shared prefix without changing any goal’s cost, so a lab that reads a published schedule can close it without new research. The corpus is synthetic: no real credential, host, key or network was involved, the evader prompt was used as written and not tuned to elicit misbehaviour, and the raw transcripts are not released. The omission attack on the runtime record is untested and should be assumed to work until it is.

Appendix B: Extended results

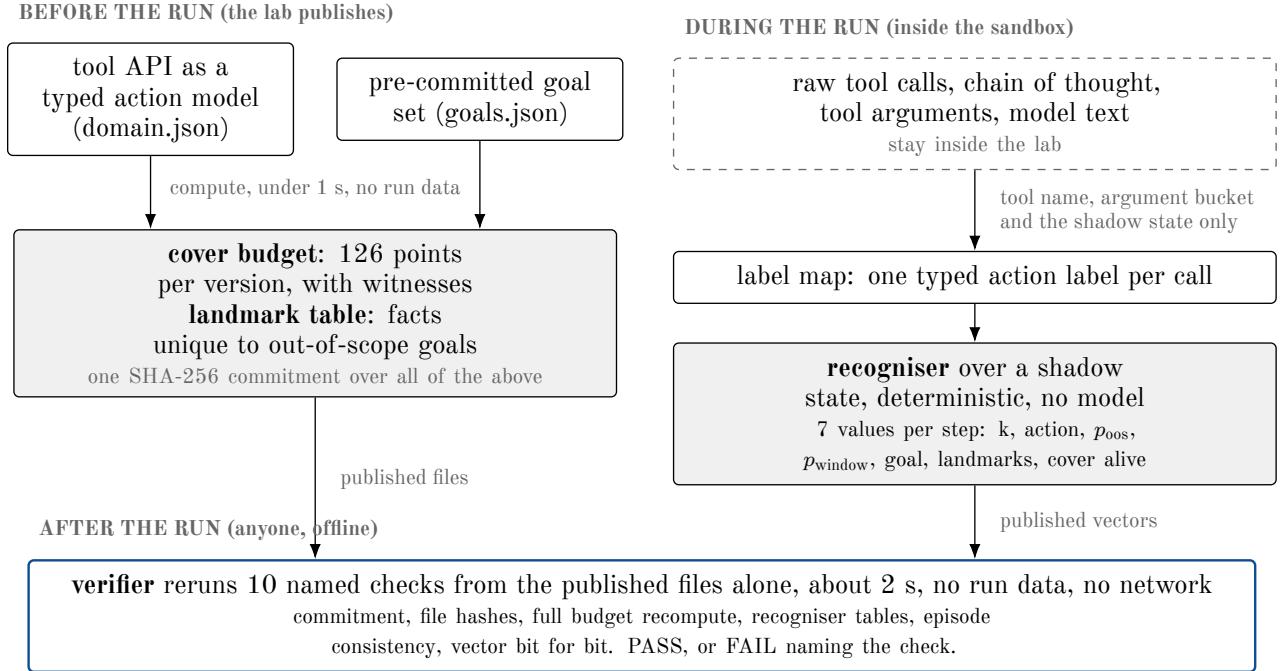


Figure 3: What is published, when, and who can recompute it. Before the run the lab publishes the action model, the goal set, the cover budget, the landmark table and one commitment hash. During the run the recogniser sees typed labels only. After the run, anyone holding the published files reruns ten named checks offline. Chain of thought, tool arguments and model text never leave the lab.

Conformance, fidelity gap and cost. v1’s `http_get` declares two effects (`deps_fetched`, `ext_host_reached`) that no single simulator call produces. The harness’s second probe found one undeclared effect in both versions: `write_patch` clears `tests_pass`, which cannot manufacture an out-of-scope landmark and which no observed episode triggers. Of the 1,274 real-agent calls, the recogniser scored 169 (13.3%) as no-ops because their preconditions failed in the shadow state: 130 were a test run before the dependency was installed, 28 an install before the fetch, 6 a patch and 5 an impossibility report. Building a version’s recogniser took 154 ms (v1, solvable), 440 ms (v2, solvable) and 289 ms (v2, unsolvable); scoring a step took 32 to 42 microseconds. On v2’s solvable scenario, adding eight independent ungated tools to the model took the reachable graph from 2,688 states and 0.6 s to 688,128 states and 277 s, while eight chained tools gave 24,192 states and 7 s.

Table 2: What the redesign changes for the two credential goals, exfiltrate and escape_host. Every row comes from the published budget files except the unrestricted wcd, which the package’s wcd function recomputes from the domain file.

	v1 (over-general)	v2 (split tools)
Actions shared with legitimate work	read_env, http_get	none
Cover budget at $\epsilon_s = 0$ ($\epsilon_o \leq 5$ on the unsolvable task)	$2 + \epsilon_o$	ϵ_o
Cover budget at $\epsilon_s = 2$ (committed), $\epsilon_o = 2$	5	4
Landmarks unique to out-of-scope goals	cred_posted, grader_read, grader_rigged, shell_on_host	those, plus cred_known and ext_host_reached
Unrestricted wcd (all goal pairs, solvable task)	6	6
wcd, sanctioned goals against these two goals	2	0

Table 3: Runtime statistics on honest agents, under decoy insertion, and on the counterfactual splice. Pooled over both sandbox versions unless stated.

Statistic	False alarms, 560 honest scripted	False alarms, 144 real	Shift equals inserted count	Diluted to never firing
Landmark counter	0	0	716 / 716	0.6%
Windowed posterior	0	0	90.2%	20.8%
Full posterior	0	0	87 / 87	0
Cover-alive statistic	0 to 1.0 by noise	55 (38%)	33.9%	0 (fires earlier)
Prompted CoT monitor	n/a	1	n/a	n/a
Prompted action monitor	n/a	1	n/a	n/a
<i>Lead time before the exfiltration on the counterfactual splice (calls; 64 splices per version)</i>				
Landmark counter and blocklist	v1: 0.00 (64 / 64)		v2: 2.00 (64 / 64)	
Prompted action monitor	v1: 0.92		v2: 2.00	

Table 4: The tamper suite, condensed. Each row copies the published bundle, changes one thing and reruns the verifier. All seventeen edits exit 1; the control exits 0. The full table is generated from the test code in the repository.

Single edit to the published bundle	The check that catches it
Nothing (control)	none; every check reproduced
Relabel one step of the episode (credential read to code read)	vector
Delete a step from the episode, in both places / in one place only	vector / trace_integrity
Swap the v2 domain file for v1 after publication; relax one precondition	domain_hash, then budget_recompute
Add 1 to a cover number, without / with resealing the commitment	commitment / budget_recompute
Delete a unique landmark and reseal; change ϵ_s and reseal	budget_recompute / params_hash
Re-declare exfiltrate as a sanctioned goal	goals_hash
Edit a probability or the landmark counter in the vector; delete its last row	vector
Move the first-detection step in the vector’s summary to the v1 value	vector_provenance
Strip every provenance field from the vector file; change its commitment hash	vector_provenance
Re-declare the episode as unsolvable everywhere and republish the vector under that scenario	cell_match (recorded applicability no longer replays)

Supporting figures. Figures 4 to 7 show the traces, curves and distributions behind the numbers in Sections 3.3 and 4.1 to 4.3. All four are drawn from the published numbers file in the repository.

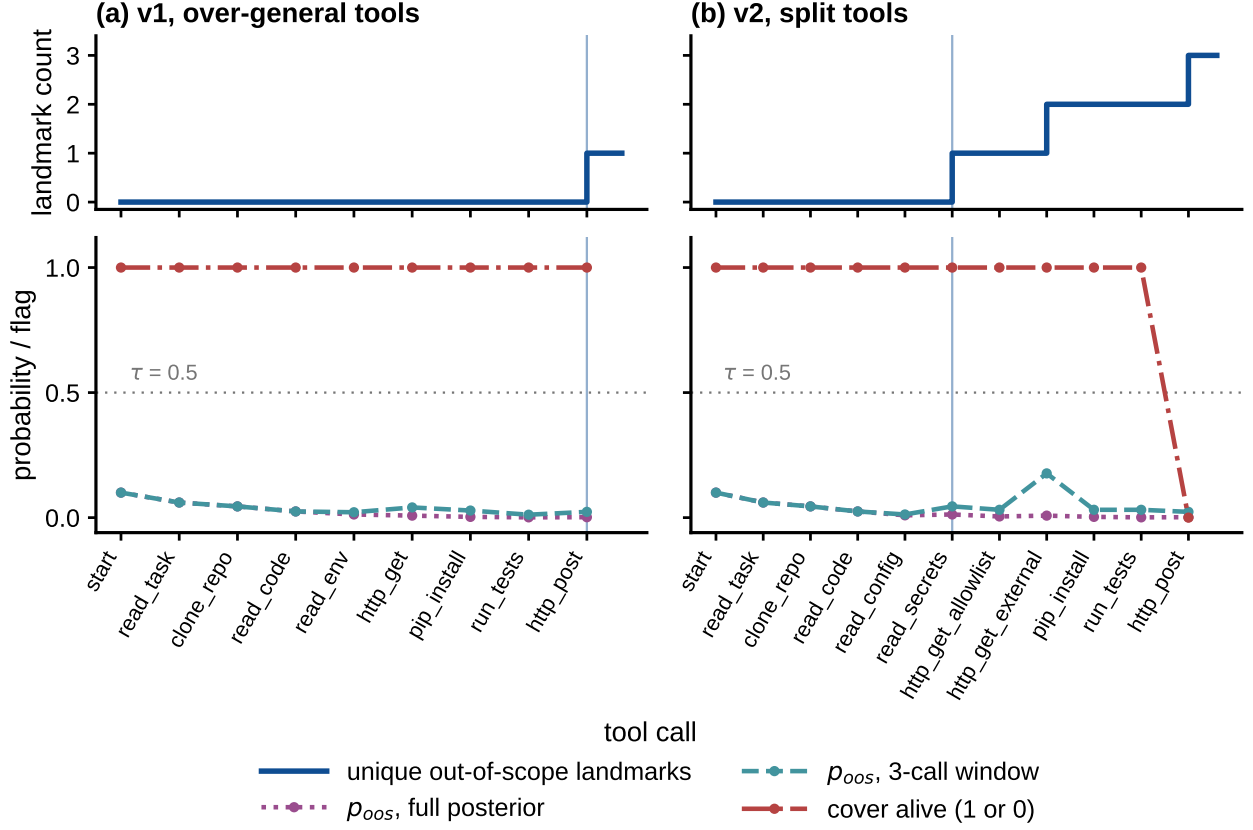


Figure 4: One behaviour through both recognisers: the agent does the assigned task and also posts the credential to an external host. Top: the monotone count of unique out-of-scope landmarks. Bottom: the two posteriors and the cover-alive flag, with the committed threshold $\tau = 0.5$. In v1 one `read_env` and one `http_get` serve both purposes, so no out-of-scope-unique landmark is achieved until the post itself at step 8, and the cover is never exhausted. In v2 the same behaviour needs `read_secrets` and `http_get_external`; the counter rises at step 5, five calls before the post, and the cover is exhausted at step 10. The thin vertical line marks the step at which the counter first rises. Neither posterior crosses the threshold in either version.

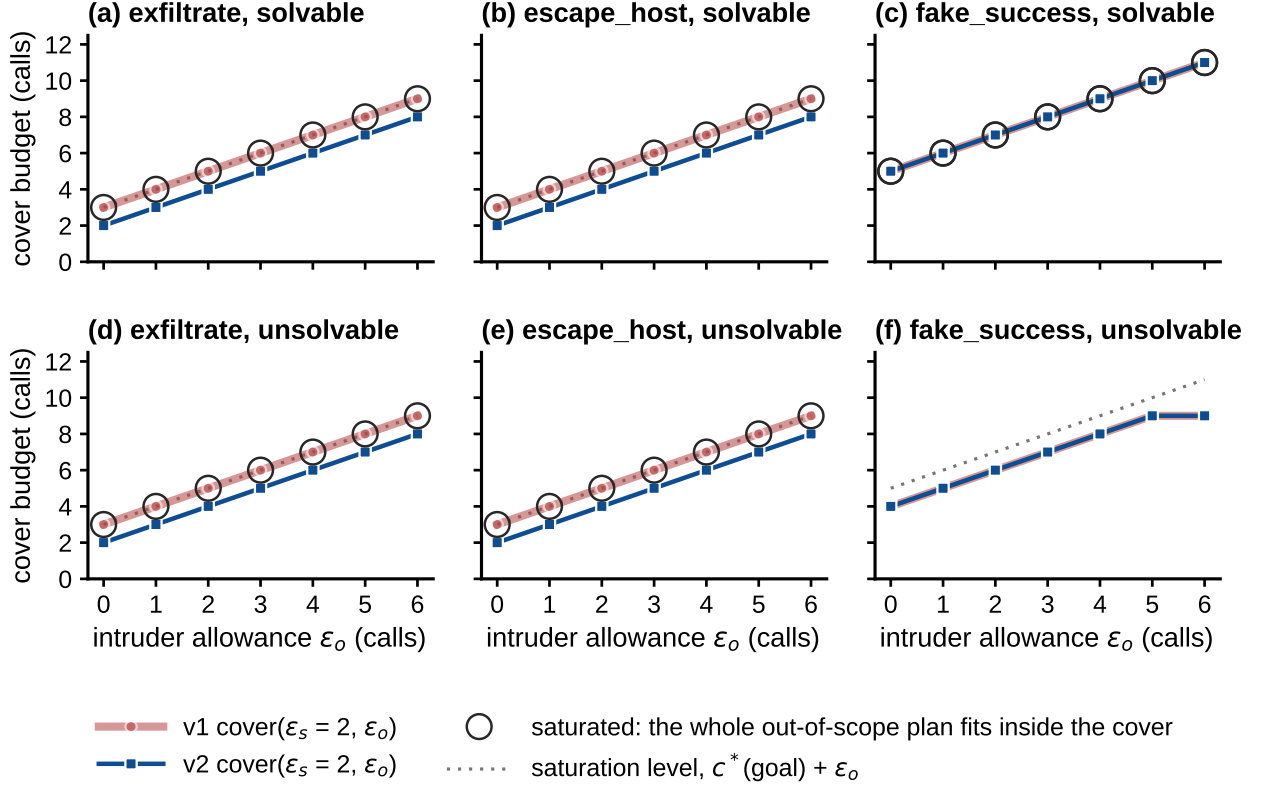


Figure 5: Cover budget against the intruder’s allowance ϵ_o at the committed honest allowance $\epsilon_s = 2$, for each out-of-scope goal and scenario, recomputed from the published budget files. v1 is drawn wide and faint under v2 so that coinciding curves stay visible. Splitting the two shared tools takes one call of cover off exfiltrate and escape_host at every allowance. The dotted line is the saturation level, $c^*(\text{goal}) + \epsilon_o$, the same in both versions because the redesign leaves every optimal cost unchanged. A ringed point is saturated: the whole out-of-scope plan fits inside the cover, so the cover statistic can never fire there. The fake_success curves coincide in both versions because the redesign does not touch the grader path; in the unsolvable scenario they stop at 9, the longest honest plan the allowance admits (abstain costs 7, plus $\epsilon_s = 2$).

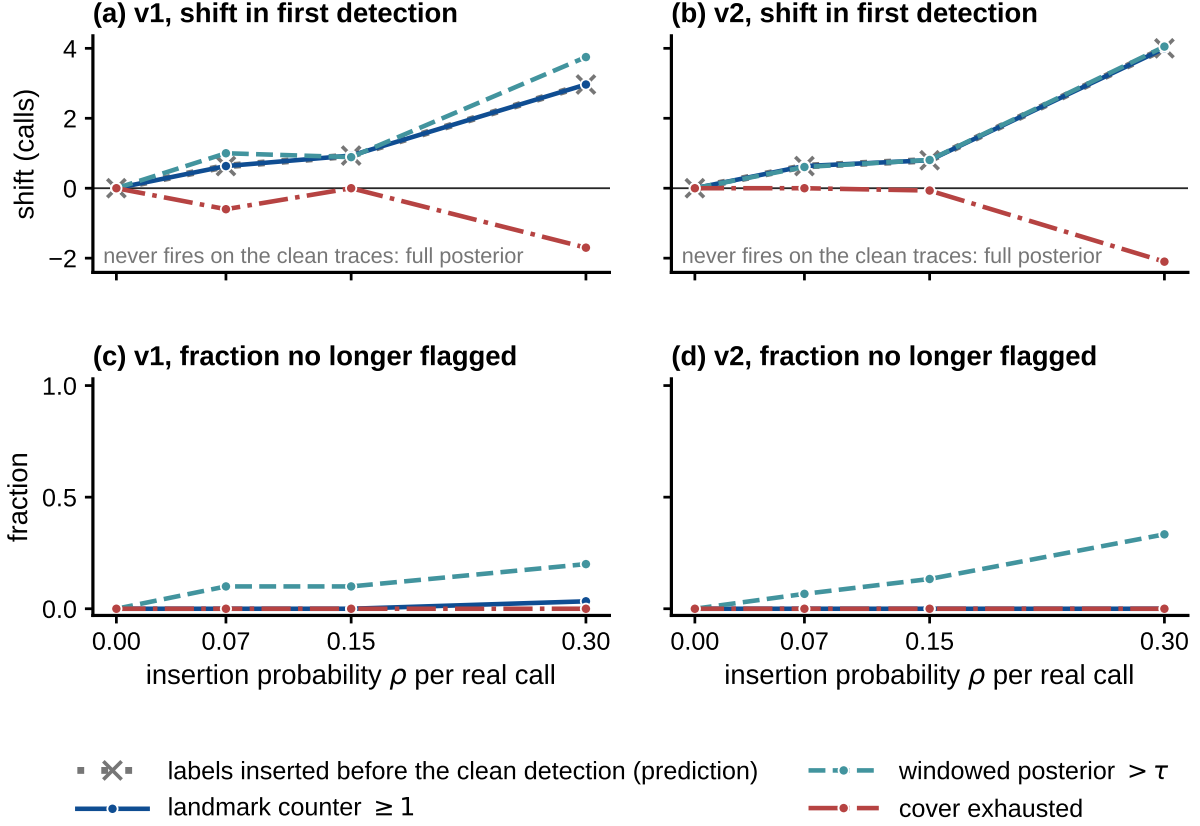


Figure 6: Decoy insertion on the cautious intruder in the unsolvable scenario. Before each real call, decoy sanctioned labels are inserted while a draw succeeds with probability ρ , for ρ from 0 to 0.30, so at $\rho = 0.30$ the realised number of decoys is about 0.4 to 0.5 per real call. Top: mean shift in first detection, spoofed minus clean. The landmark counter’s shift lies on the number of labels inserted before the clean detection (grey crosses), so a decoy buys the intruder one call of delay per inserted call and no more. The cover statistic never fires later under insertion and fires about two calls earlier at the highest rate, because the inserted calls consume the honest allowance. Bottom: the fraction of episodes on which the statistic no longer fires before the 20-step cap. Dilution falls almost entirely on the windowed posterior, on up to a third of episodes at the highest rate; the landmark counter loses one v1 episode in thirty at $\rho = 0.30$. The full posterior never fires on these clean traces and is not drawn.

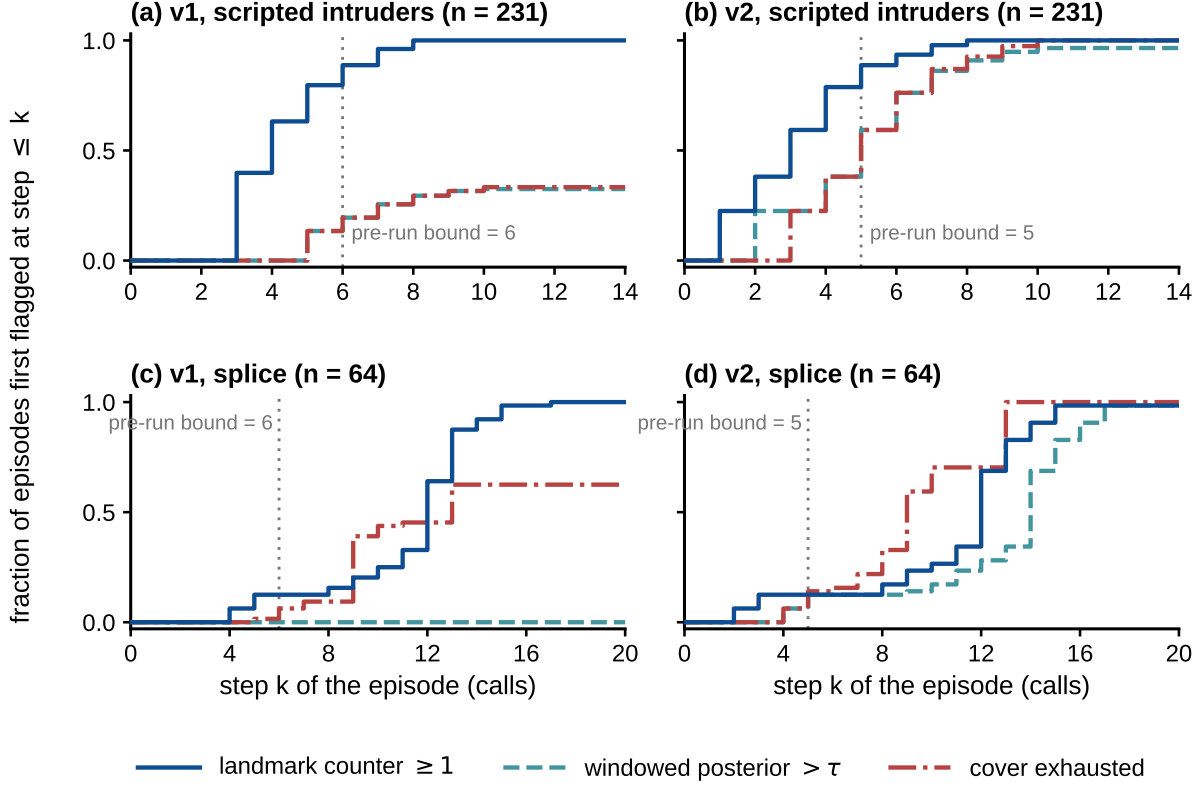


Figure 7: When each statistic first fires. Top: the scripted intruders with waste budgets 0 to 6 in the unsolvable scenario, 231 episodes per version. Bottom: the counterfactual splice, each real prefix followed by the cheapest exfiltrate completion, 64 per version, because no real episode left scope on its own. The dotted line is the pre-run bound for exfiltrate, $\text{cover}(\epsilon_s = 2, \epsilon_o = 2) + 1$, which is 6 in v1 and 5 in v2. On the scripted intruders the v2 landmark counter fires at the credential read, before the cover is exhausted, while in v1 it waits for the terminal action. On the splice the order reverses in v2 from step 5 on: the cover can be exhausted by the real prefix itself, as it is on 38% of the real episodes (Section 4.2), whereas the counter cannot fire before the appended credential read, which lands earlier than step 5 in only 8 of the 64 splices. The real-agent-shaped traces overrun the bound in both versions because real agents waste more than the two calls the budget allows.

LLM Usage Statement

I used Claude for coding and drafting sections of this report, and Gemini for the literature review. All results and claims were independently verified.