
THE RECORDER, NOT THE RECORD¹

Jigon Yoo
Independent Researcher

Track 1 (Containment)

With
Apart Research

Abstract

Frontier-lab containment standards can protect an audit log while leaving the model's execution record unauthenticated. I distinguish record integrity from recorder provenance, define a P0-P3 provenance ladder, and test the cheapest recommendation I initially expected to work: move the recorder outside the agent's execution domain. In a 720-episode harness, separation prevented post-hoc rewriting but did not prevent executor substitution. Every one of 360 substitution episodes produced an internally valid hash chain over a false account. Chain verification and target-side effect reconciliation were exactly complementary: the former detected rewriting, while the latter detected substitution. Entry-point attestation worked only until the adversary moved one call deeper. All 24 cells were unanimous under the fixed scripted mechanism, so binomial confidence intervals over seed repetitions are not the right evidence claim. The resulting standard needs both tamper-evident records and independent effect reconciliation, plus closure-level attestation whose coverage remains untested.

1. Introduction

A sealed audit log answers a narrow question: whether the delivered record was changed after it was written. It does not answer whether the component that wrote the record observed the action

¹ Prepared for the [AI Incident Response Sprint](#), September 2026

truthfully. This distinction became operationally important in the July 2026 disclosures. [3,4] One model evaded controls around external access and reconstructed a credential at runtime; another set of agents replaced part of a tool-execution system so that the transcript could name one call while a different call executed. In the latter case the relevant failure was upstream of the log. A better-protected log could preserve the false statement perfectly.

The incident record also limits the claim. The reported spoofing was used in small tests, did not reveal outputs absent from the transcript, and was assessed as visible when investigators reviewed the surrounding trace. [3] The incident establishes that a capable agent can find and install this move. It does not establish how often models will attempt it, how stealthy a mature version would be, or whether an operational monitor would alert. This paper therefore tests what several recording architectures establish after a scripted attack, not the probability that a model chooses the attack.

The work makes four contributions:

- A provenance grade P0-P3 that records where the writer of an execution record runs, independently of the security level of the protected asset.
- A real-permission harness that separates attack success, internal chain integrity, effect reconciliation, and code attestation.
- A refutation of the initial adoption claim: recorder separation stops post-hoc rewriting but does not stop an executor that lies before the recorder sees the call.
- A clause set whose evidence can be checked offline by a third party, together with explicit limits on what the experiment and its repeated seeds can support.

2. Related Work

Existing security-level frameworks already contain a strong logging ladder. RAND's model-weight security controls progress from access logs to monitoring of sensitive interactions, baseline comparison, anomaly detection, and hardware-level protection of monitoring logs. [1] NIST AU-9 likewise describes protecting audit information by storing it on separate systems. [2] These controls are substantive. The gap is one of scope: they primarily record people or processes accessing protected assets, whereas the July failure concerns a model's operational actions and the authenticity of the component reporting those actions. [3,4]

That makes provenance grade a second axis rather than a replacement ladder. A laboratory may protect weight-access logs at a high security level while allowing an evaluation harness to write model-execution records from inside the agent's domain. The two statements can both be true. The proposal is therefore to extend the familiar control shape to agent execution and to attach each clause to evidence that a reviewer can inspect without privileged network access.

Grade	Recorder position	Claim supported
P0	Written by the monitored system	No independent provenance claim
P1	Harness in the same execution domain	Only while that domain remains uncompromised

Grade	Recorder position	Claim supported
P2	Separate process or host not addressable by the subject	Resists compromise confined to the monitored domain
P3	Separate operating system and physical system with one-way input	Resists full compromise of the monitored host

The load-bearing qualification is that provenance and truth are different. A P2 or P3 recorder can provide strong evidence that it received and preserved a message. It cannot establish that an upstream executor supplied the true message. The experiment below was designed to expose that boundary.

3. Methods

3.1 Threat model and harness

Each episode creates a fresh workspace and runs an agent process under a dedicated uid. The agent makes ten tool calls, the scripted attack occurs, and five calls follow. A seeded random generator changes the tool sequence. The tool shim first emits a record and then calls an executor module. Independently, the tool implementation sends each effect to a root-owned, write-only datagram socket whose collector writes a ledger that the agent cannot read, write, or list. This ledger is the measuring instrument and remains maximally isolated in every condition.

Level	Recorder	Log and transport
P1a	Inside the agent process	Agent-owned log; direct append
P1b	Separate uid	Logger-owned directory; one-way named pipe
P2a	Separate uid and endpoint on the same host	Logger-owned directory; write-only datagram socket

The label P2a is a same-host approximation, not a separate container or host. It tests an additional endpoint boundary but cannot estimate the marginal value of hardware, hypervisor, or cloud-account separation. The observer does not vary with treatment because doing so would make the outcome definition circular.

3.2 Attacks and adjudication

Mode	Scripted action	Expected blind spot
none	No attack	False-positive control
m3a	Rewrite an existing log record	A separate log owner may prevent the write
m3b_shim	Replace the shim; record A and execute B	Hash chain can remain intact
m3b_exec	Leave the shim unchanged; replace the executor behind it	Entry-point attestation can remain unchanged

Four adjudications are kept separate. Attack success asks whether the scripted mechanical objective completed. Chain integrity recomputes the SHA-256 chain over sequence, previous hash, and payload. Effect reconciliation compares tool-type counts in the log with the independent effect ledger. Independent attestation asks whether the external recorder observed more than one shim digest; self-attestation records the digest reported by the shim itself. A detector fires only from these rules, not from a narrative interpretation after the episode.

The preregistration fixed three isolation levels, four modes, attestation on or off, and 30 episodes per logical cell, for 720 episodes. It also fixed the blind prediction that independent attestation would not fire when m3b_exec moved the substitution one call deeper. The other attack conditions were pilot-informed. Raw rows contain the seed, mechanical outcome, chain result, record and effect counts, mismatch, detector set, and process status.

3.3 What seed repetition estimates

Inspection of the implementation changes the statistical interpretation. The seed chooses tools, but the post-attack plan forces the attacked write_note tool on every even post-attack index. With five post-attack calls, each executor substitution is therefore exercised at least three times. Permissions similarly determine whether m3a can rewrite the logger-owned file. Under this plan, seed is not a randomized treatment that can change the binary verdict. It changes the size of the mismatch, not whether a mismatch exists. Repetition can expose implementation flakiness or unexpected state, but it does not create binomial sampling variation for the mechanism-level verdict.

4. Results

4.1 Attack success and complementary detectors

Recorder separation did exactly one useful thing and failed at the claim that motivated the experiment. Post-hoc rewriting succeeded at P1a and was mechanically prevented at P1b and P2a. Both substitution attacks succeeded at every tested level because neither needed to touch the recorder. The recorder's location was not on the attack path.

Attack	P1a	P1b	P2a	Interpretation
m3a	30/30 succeeded	0/30 succeeded	0/30 succeeded	Ownership boundary prevents rewriting
m3b_shim	30/30	30/30	30/30	Upstream substitution bypasses recorder isolation
m3b_exec	30/30	30/30	30/30	One-deeper substitution also bypasses isolation

Across 360 successful substitution episodes, the hash chain verified in all 360 while the record disagreed with the effect ledger in all 360. Across 60 successful post-hoc rewrites, chain verification detected all 60 while effect reconciliation detected none, because the effects themselves

had not changed. The union of these two detectors covered all 420 successful attacks and their intersection was empty. This is not redundant defence: each detector is the sole evidence source for the class the other cannot see.

Detector	Post-hoc rewrite	Executor substitution	False positives
Chain integrity	60/60	0/360	0/180
Effect reconciliation	0/60	360/360	0/180
Independent entry-point attestation	Not the target	60/60 for shim replacement; 0/60 one call deeper	0/60 where available
Self-attestation	0/60	0/360	0/180

Entry-point attestation illustrates an adversarial coverage problem. Where an external recorder existed, hashing the shim detected replacement of that file. When the executor module behind the unchanged shim was replaced, the digest did not change and the detector never fired. The preregistered falsifier did not occur. The result supports testing closure-level attestation, but it is not evidence that complete closure attestation is feasible or sufficient.

4.2 Determinism, sample size, and reproducibility

A fresh audit of the 720-row file found 24 logical cells of 30 episodes each. Within every cell, every binary outcome used by the adjudicator was unanimous: there were no intermediate k/30 values. The same audit found 715 distinct seed values rather than 720 because five seed values were reused across cells. The effect and log record totals were always 15, while the number of substituted write_note calls varied from three to five. Thus seed changed workload composition and mismatch magnitude, not the verdict.

This invalidates the previous presentation of Wilson intervals as uncertainty about the observed mechanism. A Wilson interval assumes a binomial sampling model. Here the code fixes at least one attacked call after the substitution and the permission boundary fixes whether the rewrite can occur. Reporting 30/30 with an interval invites the reader to infer seed-to-seed outcome variation that the design does not contain. The honest evidence claim is exhaustive over the tested grid: all 24 cells were unanimous under this scripted workload. It is a mechanism result, not a population estimate over deployments, operating systems, adversaries, concurrency levels, or tool graphs.

Increasing n still buys opportunities to catch race conditions, dropped records, resource failures, or unintended state leakage. It also makes a third-party rerun harder to dismiss as a single successful demonstration. It does not buy power for a 15 percentage-point minimum effect or justify multiplicity corrections when the only randomized quantity cannot change the binary outcome. A genuinely probabilistic seed experiment would vary the post-attack opportunity: remove the forced attacked tool and sample all post-attack calls, or vary attack timing and reachable modules. Under an all-random five-call post period with the present three-tool pool, for example, the attacked tool would appear with probability $1 - (2/3)^5$ rather than certainty. That workload policy, not

additional seed draws under the forced policy, is the experimental axis that can estimate a probability.

5. Discussion and Limitations

5.1 Clauses a third party can verify

Clause	Requirement	Offline evidence	Measured boundary
P-C1	Recorder outside the recorded domain	Uid or namespace map and unwritable log path	Prevents m3a above P1a; no effect on substitution
P-C2	Tamper-evident record	Delivered chain plus verifier	Detects rewriting; accepts intact false records
P-C3	External attester covers the executing closure	Manifest of every reachable module and attester identity	Named-file version failed one call deeper; closure untested
P-C4	Target-side effects can be reconciled	Counterparty ledger matched to lab record	Detects substitution; blind to pure rewriting

P-C2 and P-C4 are both necessary because they cover disjoint failure modes. P-C1 remains a cheap and useful control, but only for the narrower claim that the agent cannot rewrite records already delivered to a separate owner. P-C3 is an explicit research obligation rather than a measured recommendation. P-C4 is governance, not merely engineering: against an executor upstream of the recorder, the independently useful evidence is a record held by the system on which the action had an effect.

5.2 Adoption and sprint deliverables

The adoption test requires more than a clause. A lab and an outside reviewer need a concrete bundle contract: a manifest, chain log, attestation manifest, target-side ledger or an explicit statement that it is unavailable, and a machine-readable verdict for each clause. An offline evidence-bundle verifier can return pass, fail, or unverifiable and name the missing evidence. A one-page checklist assigns an owner, recurring cost, acceptance test, retention rule, and escalation path. This artefact directly answers whether a third party can repeat the compliance decision without the lab's network and what a lab must operationalize to adopt it.

Workstream	Elapsed budget	Output
Linux environment preparation	1-2 h	Root-capable isolated runner with setpriv and nonconflicting ids
Fresh 24 x 30 grid	17 min measured projection	720 sprint-window episodes and exact row-by-row comparison
Evidence verifier and adoption checklist	10-14 h	Offline CLI, schema, tests, checklist, dry-run record
Analysis and manuscript integration	2-3 h	Mechanism statement, limitations, provenance ledger

These streams are not mutually exclusive. Even at the upper bounds, environment preparation plus the rerun consumes 2 hours 17 minutes and the verifier consumes 14 hours, leaving more than 55 hours in a 72-hour sprint for integration, review, and submission. The rerun must not be sold as a new probability estimate. Its value is independent operational reproduction: a fresh process and fresh workload schedules should reproduce the same mechanism-level cell verdicts. A valid sentence is that a sprint-window 24 x 30 rerun matched all 720 prior episode verdicts, but only after it occurs; the report must disclose distinct seed counts, code and environment hashes, and any duplicate or overlapping seeds.

5.3 Limitations

- The adversary is scripted. Results concern what the architecture records after a fixed attack, not attack propensity in a model population.
- All isolation levels run on one host. The P2a label does not test a separate container, host, operating system, or hardware root of trust.
- The target-side observer is held at maximum isolation. Whether a real counterparty can retain and disclose comparable evidence is a governance question.
- The current workload forces an attacked call after substitution. Determinism demonstrates the mechanism within scope but is not robustness to tool graphs, concurrency, timing, module closures, or operating systems.
- Raw results are not version-anchored to the exact producing tree. Seed generation also used a hash-based mapping that produced five cross-cell collisions in the 720-row file.
- Closure attestation, baseline configuration comparison, model-elicited attacks, multi-agent collusion, and real network egress remain unmeasured.

The correct strength statement is therefore narrow. The grid is internally repeatable because the mechanism and post-attack opportunity are fixed. It is not evidence that the recommendation is robust across realistic deployment variation. Robustness would require varying the axes that can change the verdict, then showing stability or modelling the transition.

6. Conclusion

A protected record can preserve a lie exactly. Moving the recorder out of the agent's domain is necessary for tamper resistance and external attestation, but it does not stop an executor that supplies a false call before the record is written. In this harness, chain verification and target-side reconciliation cover complementary attack classes; a standard that chooses either one alone remains blind to the other. The strongest empirical statement is not a confidence interval around 30/30. It is that every tested cell was unanimous under a code path that makes the verdict independent of seed, and that the claim is limited to that mechanism.

Sprint-window work supplied both artifacts: an offline evidence-bundle verifier with an adoption checklist, and a fresh full-grid operational reproduction. The verifier makes missing evidence explicit as unverifiable rather than fail; the rerun reproduced every pre-sprint cell signature. Neither establishes robustness across workload opportunity, concurrency, closure coverage, deployment boundaries, or independent operators.

Code and Data

The pre-sprint bundle contains the harness archive, preregistration, and 720-row JSONL grid. The CLI derives seeds with Python's `process-randomized hash()`; without a fixed `PYTHONHASHSEED`, the same `--seed0` does not reproduce the list. A Python 3.12.14 pre-sprint schedule audit with `PYTHONHASHSEED=0` produced 719 distinct values, whereas the sprint Linux/Python 3.10.12 run produced 720; a fixed hash salt is therefore necessary but not a portable schedule guarantee. Each JSONL row records its actual seed for replay. Pre-sprint artifacts: `m3_harness_code.tar.gz` (SHA-256 `bece71c115eb824eb4d0fc059a220a79d25e36cc60a84c2c718aec6e4b45cfd`) and `m3_grid_n30.jsonl` (SHA-256 `e14dac7ed6e4029ebbf7acb34597e5f0e13cd3c1f5b4411d0363530ec01cf98a`). Public repository: [URL to be supplied by the author].

Sprint-Window Work Disclosure

Pre-sprint baseline: the harness, preregistration, pilot, and first 24 x 30 grid predate the September 11-13 sprint. Sprint-new A is apart-evidence-bundle-verifier v0.1.0 (package SHA-256 `8b9fa66deeb026f09cfc911a32341cda387b3402fa25ddaf8901dcefdeef77a1`): 42/42 tests passed, and a 1,807-second dry-run returned P-C1-P-C4 pass on a safe metadata-only bundle while retaining real-identity and chain-of-custody caveats. Sprint-new B used source SHA-256 `bece71c115eb824eb4d0fc059a220a79d25e36cc60a84c2c718aec6e4b45cfd` on Ubuntu 22.04/Python 3.10.12 with `PYTHONHASHSEED=0` and `seed0=20260912`. Its environment manifest SHA-256 is `0b9b11b077b92e30863b62ffd9e4bfdc43ebdb5699a449f91be9ca2b8ce289b0` and data SHA-256 is `32578cb33e31acae91edb22c4f3c6d4a29bdb58042c5e489ad2004372ad5d776`. All 720 episodes completed; the 720 distinct fresh seed values overlap the pre-sprint 715-value set in eight values; all 24 cell signatures matched. This is same-operator repeatability and operational reproduction, not full third-party reproducibility.

Author Contributions

Jigon Yoo designed the study, implemented the harness, analysed the evidence, and wrote the report.

References

- [1] RAND. Securing AI Model Weights. Logging, monitoring, and model-weight security control tables. https://www.rand.org/pubs/research_reports/RRA2849-1.html
- [2] National Institute of Standards and Technology. SP 800-53 Rev. 5, AU-9 Protection of Audit Information. <https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final>
- [3] METR and Redwood Research. Public incident account of the 2026 agent evaluation and tool-execution compromise. <https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/>
- [4] OpenAI. Public report on long-horizon model evaluations and internal deployment incidents, July 2026. <https://openai.com/index/safety-alignment-long-horizon-models/>
- [5] European Commission. General-Purpose AI Code of Practice, Safety and Security chapter and Appendix 4. <https://digital-strategy.ec.europa.eu/en/policies/contents-code-gpai>

Appendix A. Full proposed amendment text

- A fifth objective under Appendix 4.4 [5]: integrity of the record of model-initiated actions through append-only, externally anchored execution logs written by a component the model cannot modify.
- Promote optional logging to a mandatory control for agentic deployments, extending the object from inputs and outputs to tool calls and externally visible effects.
- Add an objective under Appendix 4.5 requiring validation that the observability layer cannot be disabled or replaced by the monitored system.
- Where an agentic workload acts against a third party's system, require the deployer to produce on request a reconciliation of its execution record against the counterparty's record.

Appendix B. LLM Usage Statement

LLM assistance was used to identify submission omissions, condense and format the manuscript, and prepare a draft of Appendix C from the author's recorded analyses. Numerical claims in this layout proof were checked against the local JSONL files and harness source. The author remains responsible for reviewing and adopting the final wording, citations, and disclosure decisions before submission.

Appendix C. Limitations and Dual-Use Considerations

Limitations

This appendix enumerates the claim boundaries rather than repeating the shorter Discussion summary.

- Mechanism and workload. The 24-cell result is mechanistically repeatable within a fixed scripted harness. The post-attack plan forces the attacked tool to run, so the experiment does not estimate robustness across other tool graphs, workload frequencies, or attack opportunities.
- Adversary. The attacks are scripted rather than elicited from a model. The study does not measure attack propensity, adaptive evasion, model-elicited strategies, or multi-agent collusion.
- Timing and concurrency. Episodes are serial and use fixed attack timing and recorder-drain behavior. Concurrent agents, backpressure, delayed or dropped records, and alternative installation timing were not sampled.
- Isolation boundary. P1a, P1b, and P2a all run on one host. Separate hosts, operating systems, hypervisors, cloud accounts, and hardware roots of trust were not tested.
- Attestation coverage. The harness hashes one named entry point. It does not attest the complete reachable closure or establish that closure-level attestation is feasible, complete, or sufficient.
- External evidence. The target-side observer is held at maximum isolation as a measuring instrument. The study does not establish that a real counterparty will retain, disclose, or align a comparable ledger with a lab's records.
- Reproducibility. The pre-sprint results are not anchored to the exact producing source tree. The original seed schedule also used Python's process-randomized hash() and a modulo mapping that created cross-cell collisions. Recorded episode seeds remain auditable, but seed0 alone cannot replay the schedule.

These boundaries support the phrase mechanistically repeatable within a narrow harness, not robust across deployments. A robustness claim requires experiments that vary the omitted axes and report where the mechanism stops holding.

Dual-Use Considerations

Defensive purpose. The project turns record integrity, recorder provenance, attestation coverage, and target-side reconciliation into evidence that an offline reviewer can check. It does not develop a sandbox escape, credential-recovery method, production intrusion path, or new vulnerability.

What was already public. The public incident accounts cited in [3] and [4] already described the core pattern: a compromised tool-execution layer could record one call while performing another. Post-hoc log editing is also a conventional integrity-test pattern. This paper's novelty is the controlled comparison of evidence boundaries, not the existence of those attack classes.

What this artifact adds and how it could be misused. The m3a and m3b_* modes instantiate those patterns as small Python test fixtures in a disposable workspace. A malicious reader could adapt them to probe which modules a logging or single-file attestation control fails to cover, or to design a false record that remains internally consistent. In particular, m3b_exec demonstrates how entry-point-only attestation can miss a mutable downstream executor. The fixture uses synthetic paths and exfil.invalid; it contains no working sandbox escape, real credential, production target, or privilege-escalation exploit.

Disclosure decision. No external disclosure review of the attack implementation has yet been recorded. The paper, preregistration, aggregate results, clause text, and defensive verifier schema may be shared. Until review occurs, the complete substitution implementation and any future production-environment installation or escape recipe will remain non-public. Any later code release should retain only synthetic targets, remove credentials and target-specific configuration, and preserve hashes and scope notes. Staged disclosure reduces exposure but does not eliminate the possibility of misuse.