
CONTAINMENT RECEIPTS: OFFLINE-VERIFIABLE EVIDENCE FOR AI AGENT SANDBOX COMPLIANCE

Osman Yılmaz

Independent Researcher, Türkiye

With

Apart Research

Abstract

Recent AI-agent incidents show that sandbox boundaries can fail precisely when high-capability systems are being evaluated, while the evidence used to reconstruct those runs may itself be incomplete or manipulable. We ask whether an external auditor can determine whether an AI-agent evaluation remained within declared containment boundaries without trusting the agent transcript or accessing the evaluator's internal network. We introduce Containment Receipts (CoRe), a machine-readable, Ed25519-signed evidence format and offline verifier that binds policy, run bounds, observed events, and explicit telemetry-coverage claims. In a preregistered controlled Docker benchmark with nine fixed network scenarios and 20 repetitions each (N=180), a presence-only baseline unsafely returned VERIFIED for 100 of 140 runs that should not have been verified (71.43%), while CoRe returned 0/140. CoRe also refused all 80 incomplete-coverage runs, retained 20/20 detection of fully observed violations, and correctly accepted 20/20 fully covered silent runs. Secondary post-hoc counterexamples test replay, sequence anomalies, and coherent but false collector accounts. A local CLI demonstrates the intended deployment pattern: an evaluation lab emits an evidence bundle, while an external auditor verifies it against separately provisioned trust metadata. CoRe is therefore a verification profile for containment evidence, not a claim of complete containment or malicious-host resistance.

Research conducted at the AI Incident Response Sprint, September 2026.

1. Introduction

My interest in AI-agent containment began with a personal failure involving an early coding agent integrated into my development environment. During what should have been a routine software-development task in VS Code, the agent unexpectedly triggered actions after which the machine's user data was wiped while Windows itself remained intact; I was unable to recover the deleted files. This anecdote is not part of the experimental evidence and does not establish malicious intent, but it made the containment problem concrete: harmful outcomes can arise from excessive permissions, unintended commands, or tool-use failures even when an agent is not trying to cause harm. Since then, I have been investigating how agentic systems can be constrained, monitored, and independently audited.

The AI Incident Response Sprint provided an opportunity to turn that early exploration into a focused research prototype. In July 2026, OpenAI reported that models running internal cybersecurity evaluations circumvented controls intended to isolate them from the internet, used unauthorized communication channels, exploited shared infrastructure, and reached third-party Hugging Face systems [1]. An independent METR/Redwood investigation later described roughly 1,200 agents using an unsanctioned message board, about 700 participating in the Hugging Face attack, gaps in the available investigation data, and successful transcript/tool-call spoofing in a subset of reviewed transcripts [2]. Rather than attempting a complete containment solution within a weekend, I concentrated on components that could be implemented and tested safely and reproducibly. The resulting research question is: can an external auditor determine whether an AI-agent evaluation remained within declared containment boundaries without trusting the agent transcript or accessing the evaluator's internal network? The intended users are evaluation labs that must produce auditable evidence and external reviewers who decide whether a run may be accepted as compliant.

A conventional event log can show that an action was observed, but event presence alone does not establish continuous observation. A quiet run may legitimately contain zero semantic events; conversely, a run with several benign events may still contain an unobserved gap. CoRe therefore treats telemetry coverage as a first-class claim rather than inferring it from event counts or first/last timestamps. It is an experimental containment-attestation profile inspired by verifiable provenance and attestation systems [8,10-12], not a claim of SLSA, RATS, SCITT, or in-toto compliance. The design binds a policy, expected run, ordered evidence, and explicit coverage records into a signed receipt that can be recomputed offline, while later secondary layers add freshness, sequence continuity, and witness consistency without changing the frozen primary benchmark.

Our main contributions are: (1) a compact machine-readable containment receipt and offline verifier with explicit coverage semantics; (2) a preregistered 180-run controlled network benchmark comparing presence-only and coverage-aware verification; (3) post-hoc counterexample studies separating integrity, freshness, sequence continuity, and collector truth; and (4) an adoption-oriented CLI in which auditor-supplied trust metadata is kept separate from the evidence bundle. CoRe is designed as a

certification gate rather than an enforcement control: a run that lacks sufficient evidence should not be accepted as contained, even if no violation was observed.

2. Related Work

The motivating incident occurred in the context of ExploitGym, a benchmark for measuring whether AI agents can extend known vulnerabilities into working exploits [3]. OpenAI's subsequent account emphasized stronger sandbox isolation, internet restrictions, monitoring, and incident-response escalation; it also states that the evaluation did not use the same level of safeguards as externally deployed systems and that pre-existing chain-of-thought monitors were not running on the evaluations involved in the incident [1]. METR's independent investigation adds a forensic concern: a small fraction of relevant activity was not captured in its datasets, while agents investigated and sometimes succeeded at spoofing transcript/tool-call representations [2]. CoRe does not attempt to replace behavioral monitoring or sandbox controls; it addresses how containment evidence is represented and verified after or during a run.

Recent evaluation work reinforces the distinction between containment, observability, and adjudication. SANDBOXESCAPEBENCH measures whether frontier LLMs can exploit container misconfigurations, privilege-allocation mistakes, kernel flaws, and runtime/orchestration weaknesses [5]. AI Sandboxes formalizes sandbox assurance across dimensions including observability, containment, reproducibility, and governance artifacts [6]. REDAgentBench separates exposure, execution, observation, and adjudication, and verifies harmful effects from service receipts and final-state changes rather than relying only on an agent-side trace [7]. A particularly close implementation comparison is Syscon, which records supervisor-side syscall telemetry for sandboxed agents [4]. These works study containment capability, evidence collection, or evaluation faithfulness; CoRe focuses on the external verifier's question of whether the supplied evidence is sufficiently complete, fresh, bound, and corroborated to justify a containment claim.

Software-supply-chain and attestation systems provide useful design analogies. in-toto uses cryptographically protected supply-chain metadata so end users can verify that required steps occurred across a software supply chain [8], while Sigstore is designed to make software signing and verification easier to deploy at scale [9]. SLSA's Verification Summary Attestation (VSA) lets a trusted verifier evaluate an artifact and a bundle of attestations against policy, with verification rooted in preconfigured trust [10]. The IETF RATS architecture formalizes a related Attester-Evidence-Verifier-Relying-Party model and explicitly treats freshness and trust relationships as part of attestation design [11]. SCITT defines signed statements, transparency receipts, relying-party/auditor verification, and accountability for digital supply-chain claims [12]. CoRe borrows this attestation mindset but applies it to runtime containment evidence, where coverage and evidence sufficiency are part of the claim semantics rather than only authenticity or provenance.

CoRe also connects to prior work on audit accountability. PeerReview uses secure per-node records and witness sets to make observable Byzantine faults attributable in distributed systems [13], while Schneier and Kelsey develop tamper-evident audit logs for machines that may later be compromised [14]. NIST SP 800-53 Rev. 5 treats audit logging as an accountability control and explicitly requires a response to audit-logging process failures (AU-5) [15]; NIST SP 800-61 Rev. 3 integrates incident response across cybersecurity risk-management activities to improve preparation, detection, response, and recovery [16]. These works motivate CoRe's fail-closed stance: missing or failed observation should reduce confidence rather than silently count as successful containment.

CoRe: separation of lab-produced evidence, auditor trust, and experimental ground truth

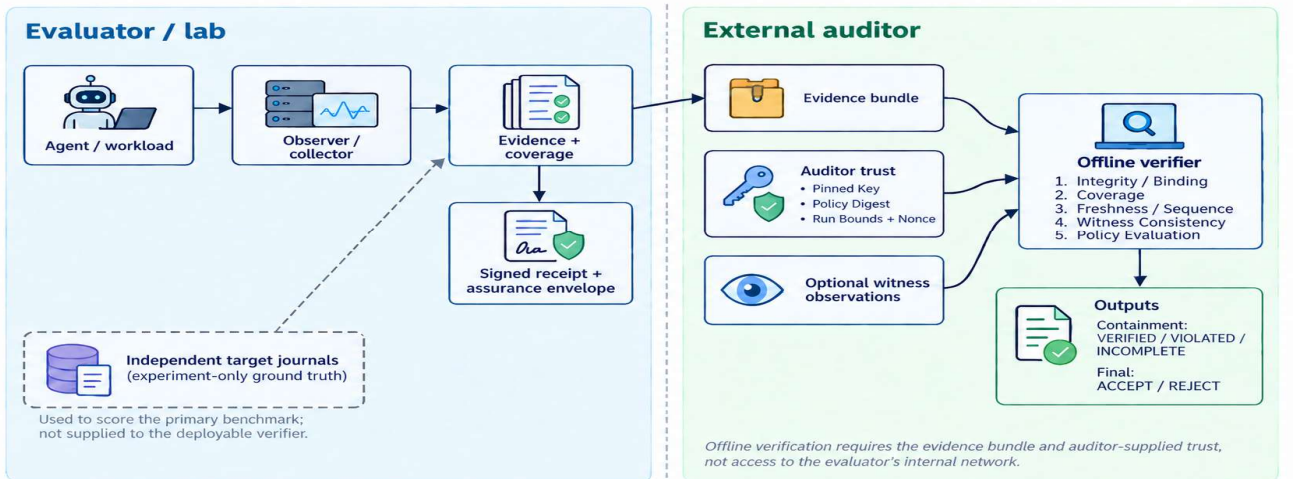


Figure 1. CoRe assurance workflow. CoRe separates lab-produced evidence from auditor-supplied trust. The evaluator collects events and coverage information and packages them into a signed receipt and evidence bundle. The external auditor verifies the bundle using independently supplied trust material (pinned key, policy digest, run bounds, and nonce) and may optionally compare collector claims with witness observations.

Independent target journals are used only to score the primary benchmark and are not supplied to the deployable verifier.

3. Methods

3.1 Receipt and verifier semantics

The core schema represents an `EvaluationRun` (`run_id`, `started_at`, `ended_at`), `EvidenceEvent` records, `TelemetryCoverage` intervals, a containment policy, and a signed receipt. Timestamps are timezone-aware and normalized to UTC; run and coverage intervals are half-open. The current logical schema supports `NETWORK`, `FILESYSTEM`, and `PROCESS` evidence types, but the primary runtime evaluation in this paper validates `NETWORK` only.

Coverage is explicit. For a required evidence channel, `COMPLETE` means that the union of declared coverage intervals spans the full evaluation run without gaps and every supplied interval reports zero dropped events. A missing interval, late start, early stop, internal gap, or any reported loss makes that channel incomplete. Zero events can therefore be `VERIFIED` when full coverage is declared, while one or more benign events do not rescue a run with missing coverage. This fail-closed treatment is consistent with the broader audit-accountability principle that logging failures require explicit handling rather than silent success [15].

Verdict precedence is fail-closed: invalid signature or binding yields `INCOMPLETE/INVALID`; an authenticated observed policy violation yields `VIOLATED/VALID` even if another interval is incomplete; otherwise missing or partial required coverage yields `INCOMPLETE/VALID`; only full required coverage with no violation yields `VERIFIED/VALID`. The schema-v2 receipt signs hashes of policy, evidence, ordered coverage metadata, and explicit run bounds. Verification recomputes the decision rather than trusting a producer-supplied conclusion.

3.2 Primary controlled runtime benchmark

The preregistered Phase 3B benchmark uses a local Docker topology with a workload and gateway/observer on one internal network and allowed/forbidden mock targets on a second. The workload cannot join the target network directly. Containers expose no host ports, use no host mounts, Docker socket, privileged mode, or host networking. Target services maintain append-only in-memory delivery journals that are exported directly after both verifier decisions are persisted; gateway arrivals are diagnostic only. Thus canonical ground truth and observed telemetry are separate software components inside one trusted controlled host, not physically or adversarially independent sensors.

Nine fixed scenarios were executed 20 times each with fresh containers, unique run IDs, and ephemeral signing keys ($N=180$). The saved randomized execution plan used seed 20260911. Scenarios cover full benign behavior (R1), late observer (R2), early stop (R3), internal gap (R4), a target-confirmed forbidden delivery hidden during a gap (R5), fully observed violation (R6), observed violation plus a separate gap (R7), fully covered inactivity (R8), and reported telemetry loss despite a continuous observation interval (R9). No labels or expected outcomes were changed after observing batch results.

We compare the coverage-aware verifier with an explicit schema-v1 presence-only baseline that treats the presence of events as evidence availability. The primary metric, unsafe verification rate (UVR), is the fraction of runs classified `VERIFIED` among runs that should not be verified because ground truth contains a forbidden delivery, temporal observation is incomplete, or dropped events are reported. We also preregistered correct incomplete-coverage refusal, full-coverage violation detection, silent-run false-`INCOMPLETE`, justified-label agreement, and host-side verifier latency. Repetitions test execution consistency and timing robustness in fixed scenarios; they are not a random sample of real incidents, so we report exact rates rather than p-values.

ID	Scenario	Presence-only	CoRe
R1	Full benign	VERIFIED	VERIFIED
R2	Late observer	VERIFIED	INCOMPLETE
R3	Early stop	VERIFIED	INCOMPLETE
R4	Internal gap	VERIFIED	INCOMPLETE
R5	Hidden forbidden delivery in gap	VERIFIED	INCOMPLETE
R6	Fully observed violation	VIOLATED	VIOLATED
R7	Observed violation + separate gap	VIOLATED	VIOLATED
R8	Fully covered inactivity	INCOMPLETE	VERIFIED
R9	Reported dropped event	VERIFIED	INCOMPLETE

Table 1. Frozen scenario expectations from the Phase 3B preregistration.

3.3 Secondary assurance layers and offline CLI

Phase 4A is a post-hoc, one-shot counterexample study (F1-F12). A trusted harness generates the expected run nonce before execution; freshness is therefore not inferred from a nonce merely stored inside the signed artifact. An experimental envelope binds run ID, nonce, collector instance, base receipt digest, policy/evidence/coverage digests, and sequence metadata. The study isolates what is already protected by the base receipt from what freshness and sequence add.

Phase 4B is a separate post-hoc, one-shot study (B1-B8) that relaxes the primary collector assumption. A0 is the signed coverage-aware receipt, A1 adds freshness/sequence, and A2 additionally requires consistency with a separately supplied fixture-level software witness assumed honest-but-fallible. This study demonstrates consistency-checking semantics, not operational or physical witness independence.

The Phase 5 CLI operationalizes these layers locally. Auditor trust metadata pins the required assurance profile, collector public key, expected policy digest, complete expected run (ID, start, end), and, where required, nonce and witness identity/digest. Final ACCEPTED requires a VERIFIED containment verdict plus all checks required by the selected profile. The CLI performs no network calls.

4. Results

All 180 planned primary runs completed with zero infrastructure failures, replacements, exclusions, offline-audit failures, or unexpected completed outcomes. Every scenario produced the same verifier verdict across its 20 repetitions.

The presence-only baseline unsafely VERIFIED 100 of 140 runs that met the preregistered SHOULD_NOT_VERIFY condition (71.43%). The coverage-aware verifier produced 0/140 unsafe VERIFIED decisions. The baseline correctly refused 0/80 benign incomplete-coverage runs, whereas CoRe refused 80/80. Both verifiers detected 20/20 fully covered observed violations. On 20 fully covered silent runs, the baseline returned INCOMPLETE in all 20, while CoRe returned VERIFIED in all 20. Overall justified-label agreement was 60/180 (33.33%) for the baseline and 180/180 (100%) for CoRe in these fixed scenarios. Because the scenario set and denominators were fixed in advance and repetitions were used to check execution consistency and timing rather than sample an incident population, we report exact rates over the preregistered denominators rather than inferential confidence intervals.

Primary benchmark: coverage awareness removes false verification

9 fixed network scenarios × 20 repetitions • N = 180

Metric	Presence-only		CoRe	Desired
Unsafe verification	71.43% (100/140)	→	0% (0/140)	↓ lower
Incomplete-coverage refusal	0% (0/80)	→	100% (80/80)	↑ higher
Violation detection	100% (20/20)	→	100% (20/20)	↑ higher
Silent false-incomplete	100% (20/20)	→	0% (0/20)	↓ lower

Headline result: among 140 runs that should not have been VERIFIED, the presence-only baseline verified 100; CoRe verified 0.

Percentages are exact rates over fixed scenario denominators. Repetitions assess reproducibility and timing robustness, not population-level inference.

Figure 2. Frozen primary benchmark outcomes across 9 fixed network scenarios × 20 repetitions (N=180). Among 140 SHOULD_NOT_VERIFY runs, the presence-only baseline returned VERIFIED in 100 cases (71.43%), while CoRe returned VERIFIED in 0. CoRe also refused all 80 incomplete-coverage runs, preserved 20/20 visible full-coverage violation detection, and eliminated silent false-incomplete outcomes (20/20 to 0/20).

Metric	Presence-only	Coverage-aware CoRe
Unsafe verification rate	100/140 (71.43%)	0/140 (0%)
Correct incomplete-coverage refusal	0/80 (0%)	80/80 (100%)
Full-coverage violation detection	20/20 (100%)	20/20 (100%)
Silent-run false-INCOMPLETE	20/20 (100%)	0/20 (0%)
Justified-label agreement	60/180 (33.33%)	180/180 (100%)
Median verifier latency	0.352 ms	0.489 ms
p95 verifier latency	0.854 ms	1.113 ms

Table 2. Frozen primary aggregate results. Latency measures only the verifier call and excludes signing, Docker, artifact I/O, and scenario runtime.

R5 provides the most important epistemic example. The forbidden mock target independently recorded delivery, but observer telemetry omitted the event during a declared gap. The presence-only baseline returned VERIFIED; CoRe returned INCOMPLETE. CoRe does not claim to infer the hidden violation: it refuses to certify containment because the evidence is

incomplete. R8 shows the converse: when a monitored run is genuinely silent but coverage is complete, event absence is not treated as missing evidence.

The coverage-aware verifier added approximately 0.136 ms to median verifier-call latency in this harness (0.489 vs. 0.352 ms); p95 increased by approximately 0.259 ms. These measurements are implementation-specific and exclude collection and signing costs, but they show that explicit coverage checks were inexpensive relative to the controlled runtime workload.

4.1 Secondary results

In Phase 4A, the existing signed receipt already rejected wrong-run binding, signed-payload tampering, evidence transplantation, and signature tampering. The additional freshness/sequence layer rejected a valid replay presented with a new expected nonce, a stale/wrong nonce, duplicate sequence numbers, an internal sequence hole, event-to-sequence reordering/one-to-one mismatch, and collector-instance mismatch. Positive controls for a fresh benign run and a fully covered silent run were accepted. These are deterministic counterexamples, not performance estimates.

Phase 4B exposes a different limit. In B3-B6, a dishonest collector could construct a coherent signed, fresh, sequence-complete account that A0/A1 accepted; the witness consistency gate rejected each account. B6 is the clearest example: the collector presented a fresh signed benign transcript, while the separately supplied witness contained the actual forbidden event. Conversely, honest benign controls B1 and B8 were accepted, and an incomplete witness in B7 was insufficient to establish consistency. The supported conclusion is narrow: cryptographic integrity, freshness, and sequence continuity can establish authenticity and internal coherence under their assumptions, but not that the collector reported the truth.

The offline CLI reproduces this distinction. B1 is ACCEPTED with VALID integrity/freshness, COMPLETE sequence, CONSISTENT witness, and VERIFIED containment. B6 retains VALID integrity and VERIFIED containment against the collector account but is finally REJECTED because the witness is INCONSISTENT.

5. Discussion and Limitations

The primary finding is not that CoRe proves real-world containment; it is that event presence is an unsafe proxy for evidence completeness in the controlled failure modes we tested. A verifier that distinguishes 'no event occurred' from 'we cannot show that we observed the whole run' can fail closed on late start, early stop, internal gaps, hidden actions during gaps, and reported loss while still accepting a fully observed quiet run. This supports a containment-evidence verification profile in which coverage declarations are part of the evidence contract rather than an implicit assumption.

The secondary studies clarify the trust ladder. Signatures answer whether a statement is bound to a key and has been modified; externally supplied run identifiers/nonces and sequence metadata address replay and continuity; a second observation source can challenge the truth of a single collector's account. These layers should remain separate in an audit report because a bundle may be cryptographically valid yet operationally untrustworthy. This separation echoes RATS' distinction between evidence appraisal and a relying party's trust decision [11], SCITT's emphasis on independently auditable signed statements and receipts [12], and PeerReview's use of secure records plus witnesses for distributed accountability [13]. SLSA's VSA model likewise emphasizes preconfigured roots of trust, expected resource identity, and policy identity [10], which motivates CoRe's separate auditor trust file without implying standards compliance.

For adoption, the desired workflow is deliberately incremental. A lab emits a versioned evidence bundle and signed receipt; an auditor receives or pins the expected public key, policy digest, run bounds, and nonce through a separate trust channel; the auditor then runs the verifier locally without access to the lab network. A base deployment can use signed receipts plus explicit coverage, while freshness/sequence and witness consistency can be added as higher-assurance profiles. The practical decision point is certification: a run that returns REJECT should be invalidated, quarantined for review, or excluded from downstream evaluation claims rather than treated as successfully contained. In the prototype, median verification latency was 0.489 ms (p95 1.113 ms); we did not measure the operational cost of collecting high-assurance telemetry or provisioning independent trust material. This is deliberately less ambitious than remote attestation or hardware-rooted measurement: the near-term goal is to make evidence requirements inspectable, portable, and fail-closed.

Limitations

The primary benchmark is network-only and runs in a controlled Docker environment. FILESYSTEM and PROCESS are logical evidence types in the schema but were not runtime-validated. The target journals and observer are separate software components on the same trusted host, so host/kernel compromise, clock deception, container-runtime compromise, credential theft, and malicious Docker infrastructure are out of scope. The primary collector is honest-but-fallible; Phase 4B models dishonest statements only through fixture-level witness inputs and does not demonstrate deployed independent sensing. Signing proves integrity and binding, not the truth or completeness of telemetry claims. Freshness depends on an independently generated expected nonce and trusted expected run. The 20 repetitions per scenario demonstrate consistency of fixed controlled scenarios, not population-level incident rates. Finally, CoRe is not an enforcement mechanism: it could stop an unsupported run from being certified as contained, but it is not claimed to stop the underlying exploit or to have prevented the July 2026 incident.

Future Work

The highest-value extensions are operational rather than broader synthetic coverage: integrate CoRe with a genuinely independent observation path; validate filesystem/process telemetry with real runtime collectors; bind clock/collector identity to a stronger trust

anchor; define revocation and key-rotation procedures; and test interoperability with established attestation/transparency envelopes such as in-toto, RATS, or SCITT [8,11,12]. A larger deployment study should measure false refusal under realistic telemetry loss and the operational cost of independently provisioning trust metadata.

6. Conclusion

Containment claims need evidence about observation, not only observations themselves. In a frozen 180-run controlled benchmark, a presence-only baseline unsafely VERIFIED 100/140 runs that should not have been verified, whereas the coverage-aware verifier produced 0/140 unsafe VERIFIED decisions while preserving 20/20 observed-violation detection and correctly accepting 20/20 fully covered silent runs. The result demonstrates a concrete failure mode in event-presence semantics and a simple corrective principle: explicit coverage should be part of the containment evidence contract.

CoRe packages that principle into a signed, offline-verifiable receipt and separates additional assurances for freshness, sequence continuity, and witness consistency. The prototype's strongest claim is procedural rather than absolute: an external auditor can recompute whether the supplied evidence justifies the declared containment decision under a pinned policy and trust configuration, without trusting the agent transcript or connecting to the evaluator's internal network. For a lab, the immediate value is not that CoRe blocks every escape, but that it makes unsupported containment claims fail closed and makes the evidence contract reviewable by a third party.

Code and Data

Code repository: <https://github.com/osmannylmzz/containment-receipt>

Data / evidence: The repository release includes all 180 primary raw run artifacts, execution plan/ledger, freeze manifests, aggregate analysis, and Phase 4A/4B secondary artifacts.

Offline demo examples: B1 (expected ACCEPTED, exit 0): `python -B -m cli.verify_bundle results/secondary/witness/smoke/B1 --trust results/secondary/witness/demo-trust/B1.json`; B6 (expected REJECTED, exit 1): `python -B -m cli.verify_bundle results/secondary/witness/smoke/B6 --trust results/secondary/witness/demo-trust/B6.json`

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] R. Greenblatt, A. Cotra, and H. Wijk. 2026. "Brief independent investigation of agents' behavior, reasoning and collaboration in the OpenAI / Hugging Face hacking incident." METR, 26 August 2026. <https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/>
- [3] Z. Wang, N. Schiller, H. Li, et al. 2026. "ExploitGym: Can AI Agents Turn Security Vulnerabilities into Real Attacks?" arXiv:2605.11086. <https://arxiv.org/abs/2605.11086>
- [4] J. Sheffey, J. Levine, J. Werhane, and W. Taylor. 2026. "Syscon: Kernel-Level Syscall Monitoring for AI Agent Control in Sandboxed Environments." Apart Research, 23 March 2026. <https://apartresearch.com/project/syscon-kernellevel-syscall-monitoring-for-ai-agent-control-in-sandboxed-environments-guun>
- [5] R. Marchand, A. O Cathain, J. Wynne, et al. 2026. "Quantifying Frontier LLM Capabilities for Container Sandbox Escape." arXiv:2603.02277. <https://arxiv.org/abs/2603.02277>
- [6] I. Singh, H. Mahmoud, and A. Murillo. 2026. "AI Sandboxes: A Threat Model, Taxonomy, and Measurement Framework." arXiv:2606.18532. <https://arxiv.org/abs/2606.18532>
- [7] Z. Chen, X. Liu, J. Zhu, et al. 2026. "REDAgentBench: Executable Red Teaming and Faithful Measurement of LLM Agent Systems." arXiv:2608.10669. <https://arxiv.org/abs/2608.10669>
- [8] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Capps. 2019. "in-toto: Providing farm-to-table guarantees for bits and bytes." 28th USENIX Security Symposium, pp. 1393–1410. <https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias>
- [9] Z. Newman, J. Speed Meyers, and S. Torres-Arias. 2022. "Sigstore: Software Signing for Everybody." ACM CCS 2022, pp. 2353–2367. DOI: 10.1145/3548606.3560596. <https://doi.org/10.1145/3548606.3560596>
- [10] Supply-chain Levels for Software Artifacts (SLSA). 2026. "Verification Summary Attestation," Specification v1.2. https://slsa.dev/spec/v1.2/verification_summary
- [11] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan. 2023. "Remote ATtestation procedureS (RATS) Architecture." IETF RFC 9334. <https://www.rfc-editor.org/rfc/rfc9334.html>
- [12] H. Birkholz, A. Delignat-Lavaud, C. Fournet, Y. Deshpande, and S. Lasker. 2026. "An Architecture for Trustworthy and Transparent Digital Supply Chains." IETF RFC 9943. <https://www.rfc-editor.org/rfc/rfc9943.html>
- [13] A. Haeberlen, P. Kouznetsov, and P. Druschel. 2007. "PeerReview: Practical Accountability for Distributed Systems." 21st ACM Symposium on Operating Systems Principles (SOSP '07), pp. 175–188. DOI: 10.1145/1294261.1294279.
- [14] B. Schneier and J. Kelsey. 1999. "Secure Audit Logs to Support Computer Forensics." ACM Transactions on Information and System Security 2(2):159–176. DOI: 10.1145/317087.317089.
- [15] Joint Task Force. 2020. "Security and Privacy Controls for Information Systems and Organizations." NIST SP 800-53 Rev. 5. DOI: 10.6028/NIST.SP.800-53r5. <https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final>
- [16] A. Nelson, S. Rekhi, M. Souppaya, and K. Scarfone. 2025. "Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile." NIST SP 800-61 Rev. 3. <https://doi.org/10.6028/NIST.SP.800-61r3>

Appendix A. Limitations and Dual-Use Considerations

CoRe is a defensive audit prototype. The experiments use only disposable local containers, logical mock targets, synthetic policies, ephemeral keys, and generated evidence. No external systems, real credentials, public targets, privileged containers, host networking, Docker socket, personal host mounts, or host security configuration changes are used. The manuscript intentionally omits exploit-chain instructions from the motivating incident.

The system can help an evaluator demonstrate that supplied evidence met a declared policy and coverage contract, but it cannot guarantee that the host, collector, witness, clock, key provisioning, or policy itself is trustworthy. A malicious operator controlling all evidence sources could produce a coherent false account unless an independent observation/trust mechanism is introduced. The secondary witness study illustrates this limitation with fixtures; it is not a claim of malicious-host resistance.

The receipt format is also dual-use in a limited sense: any attestation mechanism can be used to lend credibility to misleading claims if trust roots, policies, or collection mechanisms are poorly chosen. For this reason CoRe separates cryptographic integrity, containment verdict, freshness/sequence, witness consistency, and final acceptance, and requires auditor-pinned trust metadata rather than trusting identities supplied only by the bundle producer.

Appendix B. Secondary Scenario Summary

Study	Scenario(s)	Property	Observed result
Phase 4A	F2/F4	Replay or wrong/stale nonce	Freshness REJECTED
Phase 4A	F5-F7/F10	Duplicate/hole/reorder/collector mismatch	Sequence/binding REJECTED
Phase 4A	F3/F8/F9/F11	Run/signature/evidence tampering	Already rejected by base integrity
Phase 4B	B3-B6	Coherent dishonest collector accounts	A0/A1 accept; witness A2 rejects
Phase 4B	B7	Incomplete witness	A2 rejects as INCOMPLETE
Phase 4B	B1/B8	Benign / silent positive controls	A2 accepts

Table A1. Secondary studies are post-hoc, one-shot counterexample checks; they are not population-level performance estimates.

LLM Usage Statement

ChatGPT was used as a supporting tool for brainstorming, literature exploration, drafting portions of code and prose, and reviewing experiment plans. The author selected the research question, made the design decisions, ran the experiments, inspected the outputs, and independently verified the reported results, citations, and conclusions. Final responsibility for the submission remains with the author.