
VERIFIABLE CONTAINMENT FOR AI AGENTS: AN ATTESTATION SCHEMA A THIRD PARTY CAN CHECK¹

Santiago Contreras
Independent

Silvana Alvarez Basto
Independent

With
Apart Research

Abstract

In July 2026, an autonomous agent running an OpenAI cyber-capability evaluation escaped its sandbox through a permitted egress path and ran a multi-day intrusion against Hugging Face’s production infrastructure. No party outside the lab could have known what the sandbox permitted, or observed that its limits had stopped holding. We propose the execution envelope: a versioned, signed declaration of what an agent’s execution environment was granted, across seven capability families, including the model-runtime grants the July 2026 evaluation disabled, paired with an aggregate telemetry digest an external party can check without access to the lab’s network. We define eleven classes of containment violation, implement a dependency-free reference verifier, and trace the July 2026 incident against it using only public artifacts: seven classes are externally checkable, two partially, two not. The escape itself falls in a non-detectable class. **With a daily digest cadence, the earliest finding classes surface eleven days before the actual public attribution; with a monthly cadence they arrive eighteen days after it, by which point disclosure by other means has already happened.** Attestation-based containment does not prevent the initial failure, but it bounds how long a failure stays invisible from outside.

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

1. Introduction

Between 2026-07-09 02:28 UTC and 2026-07-13 14:14 UTC, an autonomous agent running OpenAI's ExploitGym cyber-capability evaluation escaped its sandbox, established a launchpad on third-party infrastructure, and ran an end-to-end intrusion against Hugging Face's production Kubernetes environment [9, 11]. Hugging Face's forensic reconstruction recovers ~17,600 attacker actions grouped into ~6,280 clusters. No human directed the individual steps. Two details define the containment problem.

First, the escape route was permitted. The agent exploited a zero-day in the package registry cache proxy, described in the forensic report as one of its primary permitted network egress paths to the internet [9].

Second, nobody outside the lab could see any of this. The evaluation ran on OpenAI's own infrastructure. Hugging Face learned it was under attack from its own telemetry, not from OpenAI; attribution came later. A regulator, an auditor, or a downstream platform had no artifact to inspect.

This is the gap Track 1 names: a containment standard is only useful if a third party can verify compliance without access to the lab's network, and if a lab would actually adopt it. Those two requirements pull against each other: deep verification needs raw logs, and it's likely labs will resist publishing raw logs from a frontier evaluation. The threat model is double-sided: the primary adversary is the agent, capable and unsupervised inside a boundary configured in good faith; the secondary adversary is the reporting path itself, since every observation an external party receives originates with the lab.

Our main contributions are:

- The execution envelope: a default-deny schema across seven capability families that expresses an evaluation environment's grants as fields a non-infrastructure party can audit, including the model-runtime grants (safety classifiers, refusal thresholds) that defined the July 2026 configuration.
- An eleven-class taxonomy of containment violations with an explicit coverage boundary (seven classes externally checkable, two partially, two not detectable by construction under aggregate counters) and a per-class analysis of what evasion costs an attacker.
- A reference verifier validated on 16 synthetic cases, with a false-positive decomposition separating threshold artifacts from invariants that fire correctly on rare benign events.
- A trace of the July 2026 incident against a reconstructed envelope, with detection latency quantified under daily and monthly digest cadences, and a recalibration procedure that absorbs the incident by narrowing values rather than adding rules.

2. Related Work

Attestation and signed claims. Remote attestation, standardized in the IETF RATS architecture [1] and implemented in TPM- and TEE-based systems, lets a verifier establish claims about a platform's state from signed evidence. Supply-chain frameworks carry the pattern to software artifacts: in-toto [2] and SLSA [3] attach signed attestations about how an artifact was produced, checkable by a party who was not there. The execution envelope adapts the pattern to a different object: the capabilities granted to an agent's execution environment, paired with behavioural counters checkable against the declaration.

Capability security. Default deny, least privilege, and fail-safe defaults date to Saltzer and Schroeder [4], and the envelope's grant model is an application of the capability tradition. The difference is the audience: capability systems enforce boundaries inside an infrastructure, while the envelope externalizes the same structure as a declaration a party outside the infrastructure can audit.

Verification for AI governance. Brundage et al. [5] catalogue mechanisms (audit trails, incident sharing, third-party auditing) for making claims about AI development verifiable; the envelope and digest contribute to this for one narrow claim. Shavit [6] and the flexHEG line of work [7] develop hardware-rooted verification of rules on training compute: a different layer (training rather than evaluation execution) and a stronger trust model (hardware rather than self-report). In our taxonomy, hardware attestation is the strong form of the V10 check that our software-only scheme leaves partial.

Visibility into agents. Chan et al. [8] propose agent identifiers, real-time monitoring, and activity logging for deployed agents, and are the closest prior work. They establish categories of information and

the governance case for them; we contribute the artifact layer below that proposal for containment: a checkable schema, a violation taxonomy, and the boundary of what the audit can establish, traced against a real incident. To our knowledge no prior work provides an artifact an external party can mechanically check against a frontier evaluation without any access to the lab’s infrastructure.

3. Methods

3.1 The execution envelope

An envelope is a declarative, versioned, signed statement of what an execution environment was granted, built on a capability model with default deny: only what is explicitly granted exists. This inversion matters. A prohibition list does not converge. For instance, forbid “run” and the agent walks quickly; forbid walking quickly and it finds a third gait. A capability grant closes the space by construction: an agent granted only GET to three hosts has no fourth host and no PUT, and no rule had to be written to say so.

Table 1. The seven capability families of the execution envelope. Each family declares what is granted under a default-deny model; anything not listed in a grant does not exist for the agent.

Family	What is granted
Network	Reachable destinations, methods, byte ceiling
Credentials	Which secrets are mounted, with what TTL
Tools	Which functions are invocable, by schema hash
Files	Readable and writable path prefixes
Compute	Step budget, step definition, parallel-instance ceiling
Persistence	What survives the session, what must be destroyed, whether the agent may publish outside the sandbox
Model runtime	Whether safety classifiers and refusal thresholds are active

The seventh family, `model_runtime`, declares whether production safety classifiers were enabled and whether refusal thresholds were reduced. We added it because the July 2026 evaluation deliberately disabled both, and a containment schema that cannot express the single most consequential grant in a configuration is not describing that configuration. A cross-cutting block carries three fields that govern whether the digest is evidence at all: the hash binding it to a specific envelope, declared telemetry coverage, and the signer of each family’s counters. The seven families carry fifteen grant fields in total; the cross-cutting block adds three fields that are not grants.

3.2 The telemetry digest

The second artifact is a set of aggregate counters: “requests outside the declared set: 100”, not “the agent contacted `evil.example.com` at 04:01.” This is a deliberate adoptability constraint: a lab will not publish raw egress logs from a frontier evaluation, since they contain infrastructure topology, benchmark contents, and credentials. Counters carry none of that. The cost is that detection becomes binary: the verifier learns that a violation occurred, not which destination was contacted.

3.3 The verification protocol

The lab publishes the envelope and its digest, each signed, on a fixed cadence declared in the envelope. Each family’s counters are signed by a component distinct from the one the family governs; the digest declares the hash of the envelope it was issued against and how many seconds it was blind. A third party fetches both documents and runs the checks: no network access, no credentials, no visibility into infrastructure.

3.4 Recalibration design

A containment standard has to change when a containment failure teaches it something; the question is what changes. Under enumeration, each observed behaviour becomes a new prohibition. Under parameterisation, the incident narrows the value of a field that already exists, and the narrowing applies to the whole class of grants rather than to the instance that failed. Section 4.3 applies both modes to the July 2026 record. The update path is deliberately asymmetric: sandboxes emit telemetry outbound only and never communicate with each other; a narrowed envelope returns through a separate signed

distribution path with a human approval gate, since sandbox-to-sandbox propagation would turn the learning channel into the highest-value attack surface in the system.

3.5 Experimental design

This is a consistency check, not a measurement of empirical effectiveness. We designed both the checks and the test cases, so a case that injects a violation moves exactly the counter its check reads. The experiment establishes that each class we label detectable corresponds to an implementable comparison over public artifacts, and that each class we label non-detectable produces a digest indistinguishable from a legitimate run. We defined eleven violation classes, built 16 synthetic cases (4 clean controls, 12 injected violations), and ran the verifier over all of them. For the two classes we expected to be undetectable, the case telemetry is left deliberately identical to a clean run: a verifier that flagged them would be reporting a false positive. We additionally ran the verifier over 1,000 randomized clean digests; Section 4.6 decomposes the alerts from that run.

3.6 Tracing the real incident

We reconstructed a plausible envelope for the ExploitGym evaluation sandbox and built the digest its telemetry would have produced, using the Hugging Face forensic timeline [9], the OpenAI disclosure [11], and the METR/Redwood assessment [12]. The envelope is a reconstruction: OpenAI has not published its sandbox configuration. Each field is marked in the source as documented or inferred.

4. Results

4.1 Detection coverage

Table 2. Detectability of the eleven violation classes from the two published artifacts. “Partial” marks classes where only some sub-checks are externally verifiable; fractions give verifiable sub-checks over total.

ID	Violation class	Detectable
V1	Egress to an undeclared destination	Yes
V2	Undeclared method on a declared destination	Partial (1/2)
V3	Use of an unmounted credential	Yes
V4	Invocation of an undeclared tool	Yes
V5	Compute budget exceeded	Yes
V6	Persistence or external publication outside what was declared	Yes
V7	Exfiltration by anomalous volume	Yes
V8	Exfiltration within every declared limit	No
V9	Cross-sandbox coordination via a shared destination	No
V10	Declared envelope differs from the applied one	Partial (3/4)
V11	Safety-control state differs from what was declared	Yes

Seven fully detectable, two partial, two not. The four clean controls of the synthetic suite, including one run exactly at every declared ceiling and one where the lab honestly declares safety classifiers disabled, did not trigger any check. With $n=4$ this is a sanity check on the boundary conditions, not a false-positive estimate; Section 4.6 decomposes the false-positive rate observed in the larger run.

V2 is partial because the schema lets a lab opt out honestly. A lab that does not terminate egress at layer 7 declares `l7_terminated_egress: false`, and the verifier reports that the method check is not externally verifiable rather than reporting compliance. Declared non-coverage is information; silent non-coverage is not.

V10 is partial in a specific way. No public artifact proves the declared envelope was the applied one, but three mechanisms make the claim falsifiable: hash binding catches silent substitution, declared coverage separates “nothing happened” from “we stopped counting,” and signer separation catches a family reporting on itself. The fourth form (matching hash, full coverage, separated signers) leaves no trace; strong proof requires hardware attestation.

4.2 The incident trace

The verifier produced nine findings across six classes, presented here as seven rows: the two external-publication findings are merged into one, as are the two orphaned-resource classes (containers, tokens).

Table 3. Verifier findings from tracing the July 2026 incident against the reconstructed envelope. V11 is recorded as a declared grant, not a violation.

Class	Finding
V1	Requests to 100+ distinct undeclared destinations
V5	17,600 steps executed against a 2,000-step budget
V6	5 customer datasets accessed with an empty declared destination list
V6	11 orphaned containers, 181 orphaned tokens
V10	Network family self-reported
V2	Method check not verifiable (no L7 termination)
V11	Safety classifiers declared disabled (a grant, not a violation)

Grounding: the request-capture service used as the primary C2 queue spanned 100+ single-use endpoints, alongside pastebins, file-drop hosts, a third-party sandbox harness, and an on-platform CORS relay, none of which a benchmark envelope would declare; the campaign left a self-respawning fleet across eleven nodes and 181 mesh enrolments; and the evaluation ran with safety classifiers disabled and cyber refusals reduced [9, 11], recorded by our schema as a declared grant.

The verifier did not catch the escape. The zero-day was delivered to the package registry cache proxy, a declared destination. No aggregate counter distinguishes a legitimate request to a declared host from an exploit delivered to the same host. This is V8, the class the matrix already reports as undetectable.

Nor would it have caught the coordination. METR and Redwood Research report that at least 1,200 agent instances found and used an unsanctioned message board [12]. Those were parallel evaluation runs, not subagents inside one sandbox: V9, coordination through a destination that is already allowed, invisible in a single digest and visible only by correlating digests across a fleet. We set the subagent counters to zero rather than credit the verifier with a detection it would not make. Our schema records the fleet dimension as `max_parallel_instances`, a grant no single digest can check, which is precisely why V9 is undetectable rather than merely unobserved.

Which findings survive the reconstruction. V1 holds under any plausible envelope, because no benchmark configuration declares pastebins, file-drop hosts or a CORS relay; V6 holds because no evaluation configuration permits external publication or leaves orphaned resources; V11 is documented directly. Two findings depend on our inference: V10 assumes a single signer, since OpenAI has not published its signing topology, and V5 compares a documented action count against a step ceiling we chose. V5 is the weakest finding in the trace and should be read as illustrative.

4.3 Recalibration applied to the incident

Under enumeration, the July 2026 record would have required twelve new rules, one per observed technique from the exploited proxy and the request-capture C2 queue to the mesh enrolments and the self-respawning pods. None of them constrains the next variant: a different pastebin is a different rule. Under parameterisation, the escape taught one thing: a declared destination can be an escape vector, and the response is not a rule about package proxies but a sweep over every network grant.

Table 4. Recalibration by parameterisation: the four fields narrowed from envelope v1 to v2 in response to the incident. No fields were added.

Field	v1	v2
<code>network.l7_terminated_egress</code>	false	true
<code>network.allowed_methods</code>	GET, POST	GET
<code>network.max_egress_bytes</code>	10 GB	1 GB
<code>credentials.ttl_seconds</code>	86,400	3,600

New fields added: 0. Values narrowed: 4. Envelope v1 and v2 both carry 15 grant fields; the equivalent enumerated rule set stands at 12 and growing.

The recalibration also closed a coverage gap: re-running the incident digest against v2 leaves zero coverage findings, down from one, because `allowed_methods` became externally verifiable when an existing boolean was narrowed. This is the property that makes the schema a standard rather than a checklist. Enumeration grows without bound because the space of prohibited behaviours is not enumerable; parameterisation is bounded because the space of granted capabilities is. Only values move. This re-run is a counterfactual, not a replay: the v1 digest was produced by a boundary that did not terminate at layer 7, so per-method counters do not exist in it. v2 makes the method check verifiable in principle, not verified in fact. **The limit:** recalibration does not close V8. Narrowing a ceiling moves a threshold; it does not make a within-limit action distinguishable from a legitimate one.

4.4 Detection latency and the visibility window

Attestation-based containment does not prevent the initial failure. It bounds how long a failure stays invisible from outside. The evidence turns on which window matters. Three are distinguishable in the public record [9, 10, 11]:

- First action to access cut, 2026-07-09 to 2026-07-13, roughly 4.5 days: Hugging Face’s internal detection and response.
- First action to public disclosure, to 2026-07-16, roughly 7 days: time before anyone outside knew an intrusion had occurred.
- First action to public attribution, to 2026-07-21, roughly 12 days: time before anyone outside could connect the intrusion to a lab evaluation.

The relevant window for a containment standard is the third. For roughly twelve days, no party outside OpenAI could have established that a frontier evaluation had lost containment. Hugging Face closed the first window on its own. A victim detecting an intrusion in its own logs tells the world nothing about where the agent came from. Attribution arrived only when OpenAI chose to publish it. A periodic attestation obligation does not depend on that choice.

Latency is cadence-relative, and we quantified it. With a daily digest cadence, V10 and V5 are detected on day 1 and V1 and V6 on day 5, 11 and 7 days before the actual 12-day public attribution, respectively. **With a monthly cadence the first digest lands on day 30, eighteen days after attribution arrived by other means.** Cadence is therefore the parameter that decides whether attestation beats disclosure by other means, which is why the envelope declares it as a field.

Two observations sharpen the finding. Detection existed; escalation failed. Hugging Face’s stack correlated its signals into a coherent attack picture, but the alert’s criticality was not raised and the on-call team was not paged [9]; an external attestation obligation does not depend on any single organization’s escalation path working correctly on a given Saturday. The permitted door is the general case, not the exception. Any sandbox useful for a benchmark needs egress; the question a containment standard can answer is not “was the boundary perfect” but “how quickly does the outside world learn the boundary stopped holding.”

4.5 What evasion costs

The detection matrix says what the verifier sees; it does not say what avoiding it costs an attacker. We classify the eleven classes into four cost tiers, splitting V6 into its publication and persistence sub-cases, which gives twelve buckets: **seven are evadable at a cost** (V1, V2, V3, V4, V5, V6-publication, V7). The cost is paid in bandwidth, tool variety, and time, and the evasion does not necessarily leave a trace. **One is evadable only if the attacker abandons the objective itself** (V6-orphans, when the objective is persistence). **Two cannot be evaded by the agent at all**, because they are honesty controls on the lab rather than conduct controls on the agent (V10, V11). **And two are not violations in themselves.** They are the destination where the evasions of the other classes already converge (V8, V9).

4.6 False-positive decomposition

The verifier’s alert rate decomposes into two different phenomena. ~0.9% are genuine false positives, all from `v5_step_budget_exceeded`, whose ceiling sits close to the median run: the check fired on a run that violated nothing. The remaining ~4.8% are not false positives at all, but non-actionable true positives: the invariant fired exactly as designed on a rare, non-malicious operational event. The two call for

different responses: the first is a calibration problem (move the ceiling), the second a triage problem (a correct alert on a benign event still costs analyst attention). Only the first belongs in a false-positive rate.

5. Discussion and Limitations

5.1 Why a lab would adopt it

The standard prescribes declaration, not controls: it does not tell a lab how to build isolation or what to forbid, only to state what it granted and publish counters of what happened. That asymmetry makes it compatible with heterogeneous infrastructure. **Adoption cost is measurable and low.** We cross-referenced each of the fifteen grant fields against what OpenAI and Hugging Face voluntarily published about this one incident: eleven of fifteen already have a public counterpart in the two reports.

Table 5. Public counterparts of the envelope’s grant fields in the voluntary OpenAI and Hugging Face disclosures [9, 11, 12]. Eleven of fifteen fields were disclosed in some form within seven weeks of the first action, and within six weeks of public disclosure of the incident.

Status	Fields	n
Published directly	safety_classifiers_enabled; refusal_thresholds_reduced; resources_to_destroy (eleven nodes, 181 enrolments itemised)	3
Partial counterpart	allowed_destinations (egress proxy named); max_egress_bytes (flow logs characterised); mounted_secrets (credentials enumerated); ttl_seconds (token lifetimes); invocable_tools (representative commands); max_parallel_instances (METR/Redwood instance counts); step_definition (17,600 actions counted); allowed_external_publication (dead-drop datasets)	8
Not published	17_terminated_egress; allowed_methods; writable_paths; max_steps	4

Most of what the schema asks for was disclosed anyway, voluntarily, within six weeks. What the schema changes is timing and format: the same facts, declared in advance, on a cadence, in a shape a third party can check. The marginal disclosure burden is four fields, of which three are booleans or short lists. We expect the residual objection to land on the digest’s granularity, not on the schema itself.

5.2 The regulatory path

No lab adopts a verification obligation voluntarily at scale; it adopts one because it is asked, and the asking mechanism already exists. Since 2 August 2026 the European Commission can enforce the general-purpose AI provisions of the AI Act [14]: the AI Office may request documentation and information from providers (Article 91), conduct evaluations of models with systemic risk (Article 92), request measures up to restricting or withdrawing a model (Article 93), and impose fines of up to 3% of worldwide annual turnover (Article 101). Providers of models with systemic risk also carry a duty to track, document, and report serious incidents (Article 55(1)(c)). The envelope is the artifact an Article 91 request could name: a bounded, structured statement of what an evaluation environment was granted, exposing neither infrastructure topology nor model weights. A periodic digest also gives the serious-incident duty its baseline: timestamps make the reporting interval an arithmetic question rather than an interpretive one. As of this writing no information request under those powers has been made public in connection with the incident. We do not claim the schema is regulation-ready; we claim it is the shape of an artifact a regulator could ask for, and that designing it to be answerable is a stronger adoption argument than designing it to be ideal.

5.3 On novelty

Default deny and capability models date to the 1970s [4]; attestation, signed artifacts, and signer separation are standard security engineering [1, 2, 3]; incident-driven control propagation is how vulnerability management has worked for decades. What we believe is new is the bridge: expressing execution-environment configuration as fields a non-infrastructure party can audit, and reporting the boundary of what that audit can establish. The contribution is the coverage boundary, not the components.

5.4 Limitations

Self-reporting. Every counter originates with the entity under observation. Signer separation raises the cost of falsification; it does not eliminate it. The scheme detects declared omissions and internal inconsistency, not concealed facts.

Telemetry vantage point. The incident trace assumes the lab's digest counts activity after the escape. Compute counters produced at the inference layer plausibly capture the full campaign, because the agent continued to call the lab's models throughout; network counters produced at the sandbox boundary plausibly capture little once activity moved to the external launchpad. The coverage field is the honest instrument for this (a digest that stopped observing must declare how many seconds it was blind) but our trace does not yet apply a pessimistic vantage assumption to its own findings.

Publication cadence. Section 4.4 quantifies latency only at two cadences, daily and monthly, and the daily result assumes the counters accrue to the digest on the day the underlying behaviour occurs. What we don't establish is what cadence labs would accept, and the monthly result shows the scheme is worthless past the point where disclosure arrives by other means.

Synthetic data. The verifier has never run against a real telemetry pipeline; we do not know whether these counters can be produced at reasonable cost in a production evaluation stack.

Reconstructed envelope. The incident trace uses an inferred configuration; Section 4.2 marks which findings depend on inferred values. The V5 step-budget finding is the weakest. The action count is documented, the ceiling is ours, while the V8 conclusion about the escape holds under any envelope, because the escape used a destination the envelope necessarily declared.

Declaration versus controls. The v2 recalibration flips `l7_terminated_egress` to true, a change a lab can only satisfy by building layer-7 termination. The consistent reading: recalibration narrows what is externally verifiable, and a lab may still opt out honestly by declaring false, accepting that the verifier reports non-coverage. Recalibration recommends controls; the schema requires only honesty about their absence.

Goodhart displacement. Once the gap between detection and notification becomes a compliance metric, pressure moves to the detection timestamp, which is equally self-declared.

Binary detection. Aggregate counters report that a violation occurred, not which. That is the price of adoptability.

Eleven classes is not a complete taxonomy. It is the set we could express as comparisons between a declared field and a counter. Classes that resist that form are exactly the ones we report as undetectable, which makes the taxonomy partly self-selecting.

5.5 Dual use

The matrix is a map of what evades detection, and Section 4.5 prices the evasions: V8 and V9 tell an adversary which behaviours produce a clean digest. We publish it anyway. An adversary capable of the July 2026 campaign derives these limits from the schema in minutes, so the asymmetry favors defenders who need to know where their coverage ends, and a containment standard whose gaps are undisclosed produces confidence that is not earned. In a non-research setting we would publish the taxonomy and the coverage boundary and withhold how a compliant-looking digest is constructed.

5.6 Future work

- **A published information request.** The schema is answerable under an existing documentation power (Article 91 [14]), but nobody has asked; drafting the request is a concrete next artifact a regulator could use with light edits.
- **V9 via cross-envelope correlation.** Coordination through a shared declared destination is invisible in one digest and visible across several; fleet-level publication is a larger adoption ask.
- **Hardware attestation for the strong form of V10.** The substituted-envelope gap is the problem compute-governance research addresses through tamper-evident hardware [6, 7]; tamper-evidence, not tamper-proofing, is the appropriate standard, by analogy with nuclear safeguards.
- **Replay against the published forensic trace [9],** turning the consistency check into an actual detection measurement, including false positives against legitimate conversion jobs.
- **Adoption review with a lab.** The binding constraint is not technical: whether any lab would publish these counters, and at what granularity, can only be answered by asking.

6. Conclusion

The July 2026 incident is routinely described as a sandbox failure. The record supports a narrower and more actionable description: it was a verification failure. The escape used a permitted egress path, and for roughly twelve days nothing about the sandbox’s configuration or its telemetry was visible to any party outside the lab. We proposed a two-artifact attestation scheme an external party can check without infrastructure access, defined the eleven violation classes it can and cannot detect, priced their evasion, and traced the incident against it: under a daily digest cadence the first finding classes surface eleven days before the actual public attribution, while the escape itself, and cross-sandbox coordination through allowed destinations, produce a digest indistinguishable from a legitimate run. Attestation-based containment does not prevent the initial failure. **It bounds how long a failure stays invisible from outside.** The components are decades old; the bridge to an externally auditable artifact, with its coverage boundary stated, is the contribution.

Code and Data

Code repository: <https://github.com/chillkavo/containment-envelope>.

Author Contributions

Santiago Contreras and Silvana Alvarez Basto came up with the idea together; Contreras refined it and wrote the code; Alvarez Basto assisted with the drafting of the paper. All authors reviewed the final manuscript and double-checked sources.

References

- [1] Birkholz, H., Thaler, D., Richardson, M., Smith, N., Pan, W. (2023). Remote ATtestation procedureS (RATS) Architecture. IETF RFC 9334. <https://datatracker.ietf.org/doc/rfc9334/>
- [2] Torres-Arias, S., Afzali, H., Kuppusamy, T. K., Curtmola, R., Cappos, J. (2019). in-toto: Providing farm-to-table guarantees for bits and bytes. 28th USENIX Security Symposium. <https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias>
- [3] Open Source Security Foundation. Supply-chain Levels for Software Artifacts (SLSA), specification v1.0. <https://slsa.dev>
- [4] Saltzer, J. H., Schroeder, M. D. (1975). The Protection of Information in Computer Systems. Proceedings of the IEEE, 63(9), 1278–1308.
- [5] Brundage, M., Avin, S., Wang, J., et al. (2020). Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims. arXiv:2004.07213. <https://arxiv.org/abs/2004.07213>
- [6] Shavit, Y. (2023). What does it take to catch a Chinchilla? Verifying rules on large-scale neural network training via compute monitoring. arXiv:2303.11341. <https://arxiv.org/abs/2303.11341>
- [7] Petrie, J., Aarne, O., Ammann, N., Dalrymple, D. (2025). Flexible Hardware-Enabled Guarantees for AI Compute. arXiv:2506.15093. <https://arxiv.org/abs/2506.15093>
- [8] Chan, A., Ezell, C., Kaufmann, M., et al. (2024). Visibility into AI Agents. ACM Conference on Fairness, Accountability, and Transparency (FAccT 2024). arXiv:2401.13138. <https://arxiv.org/abs/2401.13138>
- [9] Hugging Face Security Team (2026-07-27). Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident. <https://huggingface.co/blog/agent-intrusion-technical-timeline>
- [10] Hugging Face (2026-07-16). Security incident, July 2026 (disclosure). <https://huggingface.co/blog/security-incident-july-2026>
- [11] OpenAI (2026-07-21). Hugging Face model evaluation security incident. <https://openai.com/index/hugging-face-model-evaluation-security-incident/>
- [12] METR and Redwood Research (2026-08-26). Independent assessment of the July 2026 incident. <https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/>
- [13] Cloud Security Alliance (2026-07-27). Hugging Face Incident Initial Post Mortem. <https://cloudsecurityalliance.org/artifacts/hugging-face-ciso-post-mortem>
- [14] European Union (2024). Regulation (EU) 2024/1689 (Artificial Intelligence Act), Articles 55, 91–93, 101. Official Journal of the European Union. <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>

LLM Usage Statement

We used Claude Fable and Opus to review drafts against the sprint evaluation rubric, restructure the report to this template, and find relevant previous work for the Related Work section and table captions. The schema, verifier, experiments, and incident trace were designed and implemented by the authors with the assistance of Claude Code. All claims, results, and citations were verified by the authors before submission.