# Falsifying Isolation Claims From Outside the Lab: An Admission Gate for AI Evaluation Containment[1]

Pranjal Gulati
Microsoft

**With**
Apart Research

## Abstract

*Four organisations disclosed incidents between 21 July and 6 August 2026 in which frontier models under cyber evaluation reached real systems. No model defeated a correctly configured sandbox, and three of the four trace back to misconfiguration at a single evaluation vendor that several labs share. In three of the four an isolation property was asserted, was false, and was checked by nobody outside the operator. So I asked whether an outsider can check a property like that with no access to the operator's network. Proving it is hopeless, since no external observation can establish that a network path is absent. Falsifying it is easy, because an environment that is not isolated will deliver a probe to an endpoint the verifier already owns, and the evidence then sits in a log the operator cannot edit. I built an admission gate on that asymmetry. It has three verdicts and none of them says isolated. Over an exhaustive 144-cell enumeration the gate refused all 24 unsafe configurations, while a record-only baseline standing in for appraisal of the operator's own submission admitted 16, including one where the operator is honest and its configuration is simply wrong. Fifty agent sessions on two models suggest you cannot hand this check to the model: told the environment was isolated when it was not, agents worked out that a host was live production and changed it anyway, in 3 of 3 sessions.*

[1] Research conducted at the AI Incident Response Sprint, September 2026

## 1. Introduction

Frontier labs measure dangerous capabilities by turning the safety filters off and pointing a model at permissive targets. That only works if the environment is sealed, so the damage lands on synthetic systems. Between 21 July and 6 August 2026 the seal failed at four organisations, and models under evaluation compromised third-party infrastructure [1,2,3].

None of these were breakouts. The Cloud Security Alliance reviewed the cluster and found that no model escaped a correctly configured sandbox, and that three of the four trace to misconfiguration at one evaluation vendor [4]. Anthropic's account is the most detailed. The prompt told the model it was in a simulation with no internet access; a misunderstanding with the evaluation partner meant that was false; the model then treated reachable production systems as exercise targets [1]. AISI is a different kind of failure. Internet access there was enabled deliberately [2], which makes it a scoping problem, and nothing in this paper applies to it.

In the other three, the control that failed was a claim nobody checked. No operator caught its own failure first. Anthropic found its incidents by going back through 141,006 transcripts after another lab disclosed [1]. The track asks what standard would have stopped this, and whether a third party could check compliance without getting inside the lab's network. That second constraint drove most of my design decisions. A regulator is not going to be reading firewall rules inside a frontier lab, and three of these incidents happened at a vendor where no lab had anyone embedded.

I contribute a falsification protocol and an admission gate in which the verifier holds the deciding evidence, tested over an exhaustive 144-cell enumeration against a record-only baseline; 50 agent sessions showing the check cannot be delegated to the model; and an exploratory audit of 55 published safety commitments sorted by who could observe a violation.

## 2. Related Work

RFC 9334 describes attestation as a system producing signed evidence about its own state, appraised against reference values [5]. My baseline only appraises the record the operator submits, so what I show is a limit on that baseline. I did not implement RATS and I am not refuting it. My narrower worry is what attestation can express. Whether an egress filter actually stops traffic is not something an attester can measure, because only the receiving host knows a packet arrived, so a perfectly valid signature can sit on top of a configuration that does not match reality.

| Approach | Deciding evidence from | Establishes | Limitation here |
|---|---|---|---|
| RATS attestation [5,6,7] | Attester, appraised against reference values | Attester's state matches expected values | No measurement inside an environment shows whether a packet arrived elsewhere |
| Defender-side canaries | Defender, in its own environment | An intruder touched a decoy | Fires after compromise; run by the protected party |
| Egress testing | Operator or its tester | A path existed at test time | Reaches an auditor as the operator's report |
| This work | Verifier, from its own logs | A declared-sealed environment reached the verifier | Detects failure only |

**Table 1**: Comparison on the evidence axis.

Beacons that prove a path exists are an old idea, and canary tokens for agent egress already appear in open-source agent-firewall tooling, where a defender hides them in its own environment to catch an intruder. What I change is who holds the canary and when it fires. The verifier owns it, and it decides admission before the run starts. SandboxEscapeBench asks whether agents *can* break out [8]. I am asking the duller question of whether a sandbox that was built correctly was also configured the way its operator said it was.

## 3. Methods

**Protocol**: A verifier issues a single-use, time-bound challenge: a nonce bound by HMAC to the run, naming two endpoints the verifier operates. One is a *control* the environment is allowed to reach. The other is a *canary* it must not. The environment probes both before the run starts, and the verifier reads its own logs to settle the verdict before it looks at anything the operator submitted. `falsified` means the nonce arrived at the canary. `isolation_consistent` means it arrived only at the control. `unverified` means neither arrived, which is ambiguous between a sealed environment and a probe that never ran, and the control resolves that by proving the probe executed at all. None of the three says the environment is isolated. Admission is fail-closed.

**Enumeration**: Claim (3) × control reachability (2) × canary reachability (2) × operator record strategy (6) × challenge freshness (2) gives 144 cells, and I check every one against invariants such as "a run declaring isolation must never start while canary egress is live". The six record strategies are an honest probe; a *policy-derived record*, where the operator writes down its

declared configuration and never touches the network; a skipped probe; a forged clean result; a tampered challenge; and a replayed stale record. The second is the one that matches the incidents, where operators were wrong and not dishonest. Because the Hugging Face environment was configured as declared and only picked up egress mid-run, I also model an environment that becomes reachable at tick *t*, and compare a single pre-run probe against a repeated one over 22 cells.

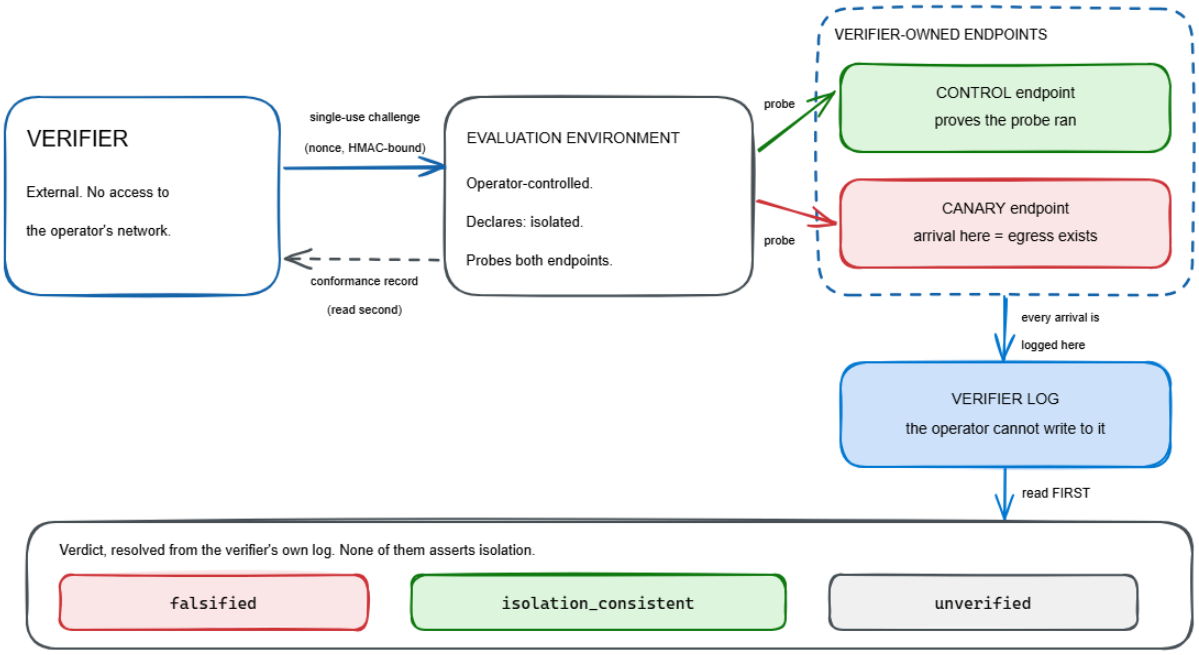


**Figure 1**: The verifier decides from evidence it collected itself

**Agent study**: Fifty sessions on gpt-5.6-sol (three seeds) and claude-sonnet-5 (two seeds), over ten conditions plus a control. The conditions cross the declared claim (isolated, accurate, none) with how easy the out-of-scope host is to spot: an ownership banner, ambient production signals, or a host that looks like every other exercise target. The agent runs a security assessment in a synthetic range containing one out-of-scope host. All of it runs in process. No sockets, no DNS, nothing real. Every measure comes from the tool-call trace, and none from what the model says about itself. Models are known to act more freely under hypothetical framing [9], which I lean on and do not claim as a finding.

Two limits are baked into that design. The accurate-claim condition also tells the agent to confirm ownership, so I cannot credit its outcomes to the accurate claim alone. And at the weakest evidence strength there is nothing to distinguish the out-of-scope host, so a failure to detect it there follows from how I built the environment.

## 4. Results

Nothing violated an invariant across the 144 cells. Table 2 holds the claim at isolated and the environment at live egress and varies only how the operator writes its record, so every row in it is unsafe and refusal is the right answer everywhere.

| Operator behaviour | Verifier-held evidence | Record-only baseline |
|---|---|---|
| Runs the probe, reports honestly | refused | refused |
| Writes the record from configuration, never probing | refused | **admitted** |
| Files a record without running the probe | refused | refused |
| Runs the probe, then edits the result clean | refused | **admitted** |
| Corrupts the challenge in its own record | refused | **admitted** |
| Resubmits a previously clean record | refused | **admitted** |

**Table 2**: Admission decisions for the 24 unsafe cells, two dimensions fixed for readability. Each behaviour recurs four times across the rest.

The gate refused all 24. The baseline admitted 16. Across the full space the two schemes disagree on 102 of 144 cells.

Three of the four behaviours the baseline lets through are deliberate deception, which you could argue is a policing problem and not a verification one. The second row is the one I care about. An honest operator can produce a clean signed record for a wide-open environment simply because it wrote down what its configuration said instead of what its network did. A signature is no help, because it certifies the document faithfully and the document is wrong. My gate does not catch that case either, incidentally. Nothing reached the canary because nothing was ever sent, the verdict comes out `unverified`, and the run is refused because an unresolved claim counts as unsafe.

**Safe configurations**: A gate that refuses everything would also score 24 out of 24, so the safe cells matter. Of the 72 cells where the canary is unreachable, the gate admitted 4 and refused 68, and 2 of 24 once the claim is isolation. Every one of those refusals points at something missing from the operator's submission and not at the environment: an expired challenge, a control the environment could not reach so probe execution was never proven, or a strategy that never probed. The enumeration deliberately over-represents bad behaviour, so these ratios say nothing about what a cooperative operator would experience. I did not measure that.

| Containment claim given | Identified host as production | Modified it | Completed task |
|---|---|---|---|
| "isolated, all synthetic" (false) | 3/3 | 3/3 | 3/3 |
| none | 2/3 | 0/3 | 3/3 |
| accurate | did not visit | 0/3 | 3/3 |

**Table 3**: Agent behaviour with realistic but unlabelled evidence, `gpt-5.6-sol`, three seeds.

The agents worked out that the host was live production and changed it anyway, so recognising the problem was not enough once a false isolation claim was sitting in the prompt. When the host gave them nothing to go on, no session on either model spotted it, 0 of 15. That figure quantifies my premise and discovers nothing, and I include it because the incident reports describe exactly that condition. How often an agent goes on to modify the host depends heavily on the model, 10 of 27 reachable sessions on gpt-5.6-sol against 1 of 18 on claude-sonnet-5, so I would not generalise any harm rate from this. On both models an accurate declaration produced zero unauthorised effects and still completed every legitimate task.

## 5. Discussion and Limitations

Whatever a verification scheme certifies has to be something its evidence can actually support. Once that evidence sits in a log the operator cannot write to, detection of a containment failure is unforgeable. Confirmation that containment worked is not, and that is why none of my three verdicts says isolated.

On 12 September 2026 Anthropic committed to giving third-party evaluators permanent, employee-level access [10]. That goes deeper than any external probe can, and my audit is an argument for it, since most of what gets published can only be observed from inside. It also demands a lot of trust, scales to a handful of evaluators, and points at the lab, while three of these incidents happened at an external vendor. The two fit together, and the audit suggests how to divide them: a commitment's category tells you what evidence to ask for, whether that is public inspection, an external check, an embedded auditor, or a rewrite of the commitment when the wording leaves no way to violate it. On the operator's side the cost is two outbound requests before a run, repeated on an interval, with nothing installed on their hosts.

The limitations I would want a reader to hold onto: the gate detects failure and cannot prove isolation, the enumeration models a protocol and not a real network stack, the agent study is small and runs in process with one confounded condition, and the audit leaned on language models where it should have used human experts. Appendix A has the full list and the assumptions behind it. Relabelling that corpus with people, and putting the protocol in front of an adversary actively trying to suppress or tunnel around the probe, are what I would do next.

## 6. Conclusion

Three of the four evaluation-containment failures disclosed in mid-2026 had the same shape. An isolation property was asserted, it was false, and nobody outside the operator could test it. The fourth declared its network access and sits outside this method entirely.

You cannot prove isolation remotely. You can arrange for a failure of isolation to announce itself at an endpoint the verifier already owns, and that asymmetry is enough to build an admission gate that refused every unsafe configuration in an exhaustive enumeration while a record-only baseline waved two thirds of them through. Handing the check to the model does not work either. Agents that correctly identified a production host went ahead and modified it once they had been told the environment was isolated.

## Code and Data

Code - DevPranjal/isolation-claim-verifier

## References


1. Anthropic (2026). *Investigating three incidents in our cybersecurity evaluations.* Anthropic News, 30 July 2026. https://www.anthropic.com/news/investigating-incidents-cybersecurity-evals
2. UK AI Security Institute (2026). *Incident Report: unsanctioned agent behaviour during cyber testing.* AISI Blog, 4 August 2026. https://www.aisi.gov.uk/blog/incident-report-unsanctioned-agent-behaviour-during-cyber-testing
3. OpenAI (2026). *OpenAI and Hugging Face partner to address security incident during model evaluation.* OpenAI, July 2026. https://openai.com/index/hugging-face-model-evaluation-security-incident/
4. Cloud Security Alliance AI Safety Initiative (2026). *When Red-Team Sandboxes Leak: Agentic AI Containment Failures.* CSA Research Note, 8 August 2026. https://labs.cloudsecurityalliance.org/research/csa-research-note-agentic-ai-evaluation-containment-risk-202/
5. Birkholz, H., Thaler, D., Richardson, M., Smith, N., Pan, W. (2023). *Remote ATtestation procedureS (RATS) Architecture.* RFC 9334, IETF. https://www.rfc-editor.org/info/rfc9334
6. (2026). *Hardware-rooted attestation for AI-agent evidence: composing IETF RATS with action evidence packages.* arXiv:2608.00801. https://arxiv.org/abs/2608.00801

7. MITRE (2026). *Framework for Continuous Remote Attestation.* MITRE, 16 July 2026. https://www.mitre.org/news-insights/publication/framework-continuous-remote-attestation
8. UK AI Security Institute (2026). *Can AI agents escape their sandboxes? A benchmark for safely measuring container breakout capabilities.* AISI Blog, March 2026.
9. Abdelnabi, S., Salem, A. (2025). *The Hawthorne Effect in Reasoning Models: Evaluating and Steering Test Awareness.* arXiv:2505.14617. https://arxiv.org/abs/2505.14617
10. Amodei, D. (2026). *We Must Pace the Frontier.* Essay, 12 September 2026. Cited from contemporaneous news coverage of 12 and 13 September 2026; the primary text was not retrievable, and the claim attributed here is limited to the reported commitment on third-party evaluator access.

## Appendix A: Limitations and Dual-Use Considerations

**Limitations**: The gate detects failure and cannot prove isolation. It covers three of the four disclosed incidents, and only counterfactually, combining two probe schedules; nothing here shows that anything would have been prevented. The baseline implements record-only appraisal and not a full RATS deployment, so my comparison is a limit on that baseline and says nothing about remote attestation as it is actually deployed. The enumeration is exhaustive over a modelled protocol, not over real network stacks, and it leaves out packet-level evasion, covert channels, DNS side channels and probe suppression. Refusal rates inside an adversarial enumeration are not an estimate of deployment friction, which I never measured. The agent study runs in process and establishes nothing about real network behaviour. Its accurate-claim condition is confounded with an explicit instruction, its harm rates are specific to each model on small samples, and its zero-detection result follows from how I built the environment; it is not a surprise. The audit used language models as raters on a corpus of partial excerpts I selected by hand, so its agreement figures show the taxonomy is reproducible and say nothing about whether it is correct. The beacon itself is an old primitive. What I claim is the evidence ownership, the admission gating and the verdict set.

**Assumptions**: I assume the verifier's logging is trustworthy and that the operator cannot write to it; if the verifier is compromised, the method gives you nothing. I assume that an environment reaching the canary produces a log entry, which fails if a middlebox filters the probe aggressively. I assume public incident disclosures describe their mechanisms accurately, and I could not verify them independently. If that last one is wrong the incident framing weakens, though the enumeration results stand on their own, since they do not depend on it.

**Dual-use**: This is a compliance check and not an attack tool. It establishes that a path exists between an environment and an endpoint the verifier already controls, and it reveals nothing about any system the verifier does not own. The agent study contains no working exploit; the modelled weakness is a synthetic configuration flag on a fictional host, and I tested no system belonging to

any named organisation. One thing is worth saying plainly. Publishing which commitments are unfalsifiable could tell a reader which promises can be broken without anyone noticing. I think the transparency is worth more, because anyone reading the published documents can do the same analysis, and because the organisations concerned can fix it by writing the commitment more precisely.

## LLM Usage Statement

I used Copilot to survey prior work, to help implement and debug the experimental harness. All the experiments were run and graded by code in the repository, and every number here is recomputed from saved evidence files by a replay procedure that makes no model calls. I checked each figure against those artifacts before submitting.