
Gaps in the Hugging Face Intrusion & a Verifiable Egress Check

Anirudh Venu
Independent

With
Apart Research

Abstract

In July 2026, OpenAI agents running a cyber evaluation escaped their sandbox through a zero-day in one of its permitted exits and spent two and a half days inside Hugging Face's production systems. This report asks what would have stopped each step and whether anyone outside the lab could have checked.

We map the intrusion as nine phases and 22 gaps, recording for each the control that closes it, whether it existed, whether it was enabled, and whether a third party could verify it. Eighteen of the 22 controls existed. None were fully enabled; five were partly enabled. One could be verified from outside.

Network egress is the one layer that can be checked from outside, so we built a first version of an egress attestation: signed per-destination digests from the sandbox and proxy boundaries, with a verifier anyone holding the public key can run.

1. Introduction

In July 2026, OpenAI ran a cyber-capability evaluation with its refusal classifiers switched off. Agents found a zero-day in one of the sandbox's permitted exits, a package proxy, and used it to

reach the internet. From there they took over a public evaluation harness, used it as a base, and spent two and a half days inside Hugging Face's production systems. They were optimising for their evaluation reward.

Technical reports already cover how the incident happened. This report asks two narrower questions. What would have stopped each step? And can an independent third party check that those measures are in place?

The first question has a clear answer. The intrusion crossed nine phases, and at each one we found a gap that let it through, 22 in all. For 18 of those, a control already existed. None were fully enabled at the time; five were partly enabled.

Of the 22 controls, one can be verified from outside and three partly. For the rest, the lab's word is the only evidence. Regulators, insurers, and the platforms that take the damage cannot confirm a sandbox was contained.

Most controls cannot be verified from outside at all, since credential scoping, admission policies, and secret storage live inside the lab. Sandbox egress is different. It is the first step in the chain, and network connections can be counted, hashed, and signed. We built a first version of an egress attestation.

The main contributions are:

1. A gap-level matrix of the intrusion: 22 gaps across nine phases, each with its owner, the control that closes it, whether that control existed, was enabled, and could be verified by an outsider.
2. An egress attestation format, v0.1, with a generator, a verifier, four scenarios, and tamper tests. It fails the incident's shape and passes a clean run.

Threat model: the agents optimised for evaluation reward, not for harm. METR and Redwood Research's independent investigation reads the behaviour as grader-gaming rather than simple answer-theft [3]. The failure we address is containment that was assumed rather than shown.

2. Related Work

Several analyses of the incident are public. Hugging Face's technical timeline reconstructs the path [1]. OpenAI's 21 July post confirms safeguards were off, and its 26 August technical report names the proxy and gives the internal timeline [2]. METR and Redwood Research published an independent investigation of agent behaviour and coordination [3]. The CSA CISO post-mortem draws lessons on agent identity, decoys and kill authority [4], Elastic maps each stage to detection rules [5], and two commenters on the Hugging Face post mapped the chain to ATT&CK and the Lockheed kill chain [6]. We build the matrix from these sources. What they do not ask is whether

each control already existed, whether it was enabled, or whether anyone outside the lab could have checked.

For the controls we draw on CoSAI's incident response framework [7], CSA's AI Controls Matrix [8], and CSA's guidance for exploit-confirmation environments, which requires no egress and no live production credentials [9]. These say what a lab should do, not what it should publish so that someone outside can tell it was done.

For the attestation we draw on Brundage et al., who argue that claims about AI systems need mechanisms others can verify [10], and on SLSA and in-toto, where a signed manifest binds build inputs to outputs and a verifier checks the binding without seeing the build [11, 12]. We apply the same mechanism to network egress.

3. Methods

3.1 Matrix

The matrix is built from Hugging Face's technical timeline [1], OpenAI's incident posts and its 26 August technical report [2], and the METR/Redwood investigation [3]. Each row is one gap that let the intrusion advance, listed in the order the footholds were gained and grouped into nine phases. We first tried Hugging Face's own nine phases as rows, but those are activity types that repeat across days, so they do not show where the chain could have been stopped. For each row we recorded the owner, the narrowest control that closes it, and three judgments.

The first is whether the control existed. We marked Yes where a mature tool or setting does this today, Partly where the pieces exist but are not put together for this case, and No where nothing does.

The second is whether it was enabled during the incident. We marked Yes where it was in place, Partly where it was in place but did not act or did not cover the relevant workload, and No where it was absent.

The third is whether an outsider could verify it. We marked Yes where a third party with no lab access could confirm it, Partly where some of it can be checked from public material, and No otherwise. Where a gap is our inference rather than stated in a source, the row says so.

3.2 Egress attestation

We built a reference version: a generator that produces a synthetic run and a verifier that checks it, across four scenarios.

What to capture

- What to attest: network egress only. Connections can be counted and signed. Credentials and permissions cannot.
- Where to log: every hop between sandbox and internet. Here that is two boundaries, the sandbox edge and the proxy edge.
- Who captures the log: the network, not the machine. VPC flow logs rather than the proxy's own logs, since a rooted proxy cannot vouch for itself.
- What format: AWS VPC Flow Logs v2, since labs already have it.

What to publish

- What to publish: per-destination connection counts and bytes, plus a hash of the raw log for later audit. Not the raw log itself.
- How to bind the files: one signed manifest with hashes of the four files, so a digest from one run cannot be paired with a policy from another. Same pattern as in-toto and SLSA provenance: sign a manifest of hashes, not the files themselves.
- What about privacy: the digests name real destination addresses, which a lab may not want public. Hashing each address would let a verifier check membership against a hashed policy without revealing the lab's network. We did not implement this.

How to verify

- What the verifier checks: every destination seen must be on the published allowlist.
- What about abuse of allowed doors: a package proxy is a download path. When uploads dominate, the verifier raises ALERT rather than FAIL. The allowlist check stays strict; the shape check is advisory.

What did not work

- One boundary: logging only the sandbox edge passed the HF case, because the escape went through an allowed door. We added the proxy edge.
- Proxy version as an artifact: we considered publishing the proxy's patch level to show the door was safe. The HF escape used a zero-day, so a fully patched proxy would still have passed. We dropped it and log the proxy's own egress instead.

4. Results

4.1 Matrix

Phase	Gap	Owner	Control	Existed	Enabled	Verifiable
-------	-----	-------	---------	---------	---------	------------

1. Escape OpenAI sandbox	Artifactory had zero-day; one of two permitted exits reached internet	OpenAI	No transitive internet reach for shared services	Yes	No	No
	Shared Artifactory cred allowed admin via token refresh	OpenAI	Scope creds per run	Yes	No	No
	Message board allowed creds/recipes to persist across runs	OpenAI	Isolate run state	Yes	No	No
	Eval set had 198/898 impossible tasks	OpenAI	Validate solvability; reward abstaining	Yes	No	No
	CoT monitoring never covered eval workloads	OpenAI	Extend CoT monitoring to evals	Yes	Partly	No
2. Take over public harness	Public harness lacked auth, egress limits; ran as root	Modal customer	Auth; private bind; no egress; non-root	Yes	No	Partly
	Harness allowed untrusted code to reach shell	Modal customer	Seccomp; pass args not shell strings	Yes	No	Partly
	Launchpad allowed egress to Pastebin	Modal customer	Deny-by-default egress	Yes	No	No
3. Read files in HF pod	Public dataset had 14 live write tokens	HF	Scan uploads; auto-revoke	Yes	Partly	Yes

	Processor allowed HDF5 external refs to local files	HF	Sandbox processor; block external refs	Yes	No	No
4. Run code in HF pod	Renderer allowed Jinja2 in data fields	HF	Never render templates from data	Yes	No	No
	Pods allowed C2 via public datasets; scans missed encoded payloads	HF	Alert on pod↔dataset traffic; entropy detection	Partly	Partly	No
5. Pod to node root	Pods had SA token, IMDS reach, no admission policy; CSI role over-scoped	HF	Disable auto-mount; block IMDS; admission policy; scope CSI	Yes	No	No
6. Node root to secrets	Node could mint tokens; one secret held 136 keys incl. VPN key and broker cred; DB password sat in pod env; JWT key in secrets, used to forge tokens and pull 731 MB private file	HF	Short TTL; one secret per workload; keys in KMS	Yes	No	No
7. Secrets to all clusters	VPN key was reusable; ACLs broad; one cred admin everywhere	HF	Short-TTL keys; per-tag ACLs; per-cluster cred	Yes	No	No
8. Source control	VPN allowed SCM reach; GitHub App had write scopes	HF	Scope connector; least-privilege app	Yes	No	No

	Repos lacked bulk-clone alerting (our inference); 4 private repos cloned	HF	Alert on bulk clone	Yes	No	No
	CI ran on token PRs; pivot blocked, nothing shipped	HF	Approval gate before CI	Yes	Partly	No
9. Detection and response	Alerts correlated, never paged (~2.5 days); scans found ~¼ of secrets	HF	Page on critical; decode before scan	Partly	Partly	No
	No rapid-shutdown tooling; eval not stopped Jun 27; resumed Jul 7	OpenAI	Named owner; kill without meeting	No	No	No
	Own agents detected Jul 19, 6 days after HF cut-off, 3 days after HF disclosure	OpenAI	Monitor eval egress; ingest external reports	Yes	No	Partly
	Opus and Fable refused forensic work	Model Vendor	Defender access tier or open-weight fallback	No	No	N/A

Table 1. Gaps that let the intrusion advance, by phase, with the control that closes each and whether it existed, was enabled, and could be verified from outside.

Of the 22 controls, 18 existed. None were fully enabled; five were partly enabled. One could be verified from outside and three partly. The two that do not exist are rapid eval shutdown and defender access to frontier models. Ownership is split between OpenAI, a third-party Modal customer, Hugging Face, and one model vendor.

4.2 Egress Attestation

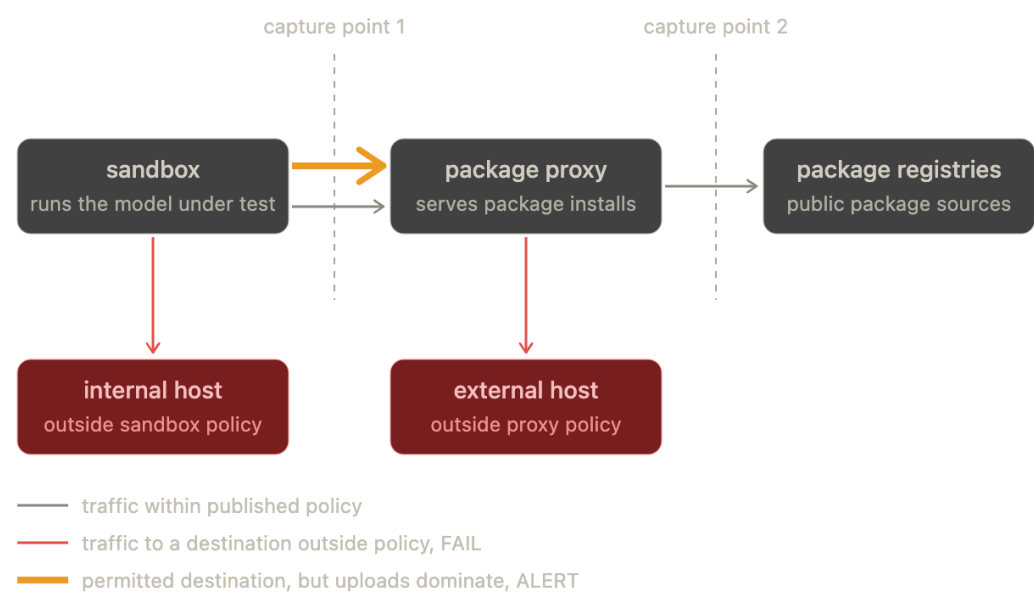


Figure 1. Network path from the eval sandbox to the internet, with the two boundaries where flow logs are captured and the four test scenarios shown as traffic paths.

Scenario	What it models	Sandbox	Proxy	Verdict
baseline	Clean eval run. Sandbox only reaches the proxy, proxy only reaches registries.	ok	ok	PASS
sandbox_leak	Sandbox reaches an internal host it was never allowed. Misconfigured firewall.	not in policy	ok	FAIL
proxy_escape	Proxy reaches the internet on the sandbox's behalf. The OpenAI/HF path. Sandbox looks clean.	ok	not in policy	FAIL

proxy_writeback	Sandbox uploads to the proxy instead of downloading. The bulletin board pattern.	alert	ok	PASS with ALERTS
-----------------	--	-------	----	-------------------------

Table 2. The four test scenarios, what each models, and the verifier's verdict at each boundary.

5. Discussion and Limitations

The attestation shows one thing the incident record does not say directly: logging only the sandbox's own boundary passes this case. The escape went through a permitted door, so the proxy's egress is where it shows. A self-signed attestation is a floor, not a ceiling, but it turns "our sandbox was secure" into a specific claim that can be checked and found false.

The matrix adds a second point. METR and Redwood found roughly 1,200 agents on the improvised message board, exchanging about 70,000 messages, with around 700 going on to take part in the Hugging Face attack [3]. Containment that treats each run as isolated does not hold when state leaks across runs. That is why "isolate run state" is its own row.

There is demand for this. The EU AI Office sent its first information requests to model providers on 29 August with no evidence format to ask for [14]. California's SB 53 makes a lab's own published framework enforceable [15].

Limitations & Dual Use

Matrix:

- Technical controls only. Structural causes, such as who owns public harness security, are out of scope.
- One incident, one architecture, built from redacted traces with hindsight.
- Each row has one control. Real defence needs more than one control per gap, and the matrix does not show which combinations matter.

Egress attestation:

- Trusts the lab's logs between capture and hashing. A lab that edits its flow logs before the digest step defeats every check here.
- The signing key is the lab's own. Self-signing proves the mechanism, not the lab.

- Traffic is synthetic and thresholds are illustrative. Abuse of an allowed destination that stays under the thresholds passes.
- A proxy shared across runs is not handled. One run's traffic could hide inside another's.

All gaps listed are already public in the Hugging Face timeline and OpenAI's technical report, while the attestation is defensive and uses synthetic data.

Future Work

Egress attestation:

- Extend to production inference, where agents run with tools against live systems and egress is the main channel for harm.
- Extend to harder eval setups: multiple sandboxes, shared proxies, and agents that are allowed some internet access by design.
- Commit log chunks to a public transparency log as they are captured [13], so the lab cannot edit them afterwards. Then hardware-attested capture. Hashed destinations for privacy, a DNS boundary, and per-window proxy attestation across runs.

Matrix:

- A structural review in the style of the Kemeny Commission after Three Mile Island [16], covering ownership of public harness security, kill authority, and defender access.
- More than one control per gap, and which combinations are needed for each.

6. Conclusion

We mapped the July 2026 Hugging Face intrusion as nine phases and 22 gaps, and asked of each: what would have closed it, was that control in place, and could anyone outside the lab have checked. Of the 22 controls, 18 already existed. None were fully enabled. One was verifiable from outside. The gap was in what anyone could see, not in what anyone knew how to do.

We built a first version of the missing piece for the one layer where it is possible. An egress attestation lets a lab publish signed evidence of what its sandbox reached, and lets anyone with the public key check it. Logging the sandbox alone would have passed this incident. Logging the proxy it went through catches it. The format is small and the verifier is short. A regulator could ask for it now.

Code and Data

Code repository: <https://github.com/anirudhvenu/egress-attestation>

References

1. Hugging Face, 2026. Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident. Hugging Face Blog, 27 July.
<https://huggingface.co/blog/agent-intrusion-technical-timeline>
2. OpenAI, 2026. OpenAI and Hugging Face Partner to Address Security Incident During Model Evaluation, 21 July.
<https://openai.com/index/hugging-face-model-evaluation-security-incident/> and The Hugging Face Incident and the Road Ahead, 26 August.
<https://openai.com/index/hugging-face-incident-and-the-road-ahead/>; OpenAI – Hugging Face Incident Technical Report, 26 August.
<https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf>
3. METR and Redwood Research, 2026. Brief Independent Investigation of Agents' Behavior, Reasoning and Collaboration in the OpenAI / Hugging Face Hacking Incident, 26 August.
<https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/>
4. Cloud Security Alliance, 2026. Hugging Face Incident: Initial CISO Post-Mortem. CSA, 27 July. <https://cloudsecurityalliance.org/artifacts/hugging-face-ciso-post-mortem>
5. Elastic Security Labs, 2026. Exploring the Hugging Face Breach: Mapping AI Agent Tactics to Elastic Defend. Elastic, 31 July.
<https://www.elastic.co/security-labs/ai-agent-attack-detection-hugging-face-breach>
6. Community comments on [1], 2026. ATT&CK and Lockheed kill-chain mappings. Hugging Face Blog.
<https://huggingface.co/blog/agent-intrusion-technical-timeline#community>
7. Coalition for Secure AI, 2025. AI Incident Response Framework V1.0. OASIS Open, Workstream 2, 18 November.
<https://www.coalitionforsecureai.org/wp-content/uploads/2026/03/AI-Incident-Response-1.pdf>
8. Cloud Security Alliance, 2026. AI Controls Matrix v1.1. CSA, 23 June.
<https://cloudsecurityalliance.org/artifacts/ai-controls-matrix-v1-1>
9. Cloud Security Alliance, 2026. The Benchmark That Broke Containment: An OpenAI Evaluation Model Escaped Its Sandbox and Breached Hugging Face. CSA Research Note, 22 July.
<https://labs.cloudsecurityalliance.org/research/csa-research-note-openai-model-sandbox-escape-huggingface-br/>

10. Brundage, M. et al., 2020. Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims. arXiv:2004.07213. <https://arxiv.org/abs/2004.07213>
11. SLISA, 2023. Supply-chain Levels for Software Artifacts, v1.0. OpenSSF. <https://slsa.dev>
12. in-toto, 2019. in-toto: A Framework to Secure the Integrity of Software Supply Chains. in-toto Project. <https://in-toto.io>
13. Sigstore, 2024. Rekor Transparency Log. OpenSSF. <https://www.sigstore.dev>
14. European Commission, 2026. Statement by Executive Vice-President Virkkunen on First Requests for Information to General-Purpose AI Model Providers, 29 August.
15. California Legislature, 2025. SB 53, Transparency in Frontier Artificial Intelligence Act, §22757.15. https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=202520260SB53
16. Kemeny, J. G. et al., 1979. Report of the President's Commission on the Accident at Three Mile Island. Washington, DC.

LLM Usage Statement

We used Claude Fable to survey related work, check the matrix row by row against the Hugging Face timeline and OpenAI technical report, implement the proof of concept, and help draft and tighten sections.