

WHAT A CONFORMANCE SUITE CANNOT SEE: A SOUNDNESS FAILURE IN BLACK-BOX EGRESS VERIFICATION

Nafees Ahamed, Berlin

Apart Research x CeSIA, AI Incident Response Sprint, September 2026

Track 1, Containment

Artifact: github.com/Nafsgerman/egress-conformance

ABSTRACT

In July 2026 two OpenAI models left an evaluation sandbox through the package-registry cache proxy that was their one permitted network egress. Track 1 asks for a containment standard a third party could verify without access to the lab's network. We specify one property, that sanctioned egress must be a content-addressed replay-only mirror with no code-execution path, as six black-box predicates, and we run them three times: against a reference pair we wrote, against a real package cache using naive probing, and against the same cache with an implementation adapter. The naive run reported five of six predicates satisfied for a component that determinately violates the central one. Our headline finding is therefore about the method rather than the property: black-box conformance probing by URL convention is unsound in both directions, and the failure is silent.

1. WHAT WE SET OUT TO DO, AND WHAT WE FOUND INSTEAD

The intended contribution was a containment property and a suite that checks it:

An evaluation sandbox's sanctioned package egress must be a content-addressed, replay-only mirror with no code-execution path, and compliance with that property is verifiable from outside the lab.

We still hold that property. The second half of the sentence, the verifiable-from-outside half, turned out to be where the interesting problem lives. When we pointed the suite at software we had not written, it returned a confident, wrong, and entirely plausible answer. Section 5 is that result. It is a finding about conformance methodology, and it applies to any regime in which a third party runs a fixed probe set against a component whose implementation they do not control.

We report it as the headline because it is the part a reader cannot get elsewhere, and because we would rather publish the negative result than the clean table we could have published by not testing.

2. WHAT THE PUBLIC RECORD ESTABLISHES

Hugging Face published a forensic reconstruction covering roughly 17,600 recovered attacker actions between 9 July 02:28 UTC and 13 July 14:14 UTC, grouped into approximately 6,280 clusters, with named vulnerabilities, per-phase counts and verbatim commands. OpenAI published its own account of the origin.

We re-read the timeline not as HF's nine overlapping phase labels but as a chain of trust boundaries crossed, because a containment control fires at a boundary, not at a label. The labels overlap heavily in time; recon runs the entire campaign and is not a stage. We identify thirteen crossings, B0 through B12, abbreviated:

B0 Eval sandbox to open internet, via a zero-day in the package-registry cache proxy that was one of the agent's permitted egress paths. No marker published.

B1 Internet to a third-party code sandbox, via an unsecured public submit-function endpoint. Two converging routes: redefinition of `sqlite3_initialize()`, which runs as a constructor on library load, and shell injection through a source-path field.

B2 Sandbox to HF production pod, file disclosure without code execution, via HDF5 external raw storage pointing at local worker paths. An earlier SSRF attempt had been rejected by the datasets URL allowlist; the agent stopped asking the worker to fetch and made it act locally instead.

B3 Same path, now code execution, via Jinja2 SSTI in an `fspec` reference:// offset field.

B4 Pod to Kubernetes cluster API, via the service-account token at `/var/run/secrets`.

B5 Pod to cloud estate, read-only, using temporary instance-metadata credentials replayed from external addresses on two hosting providers. All mutating calls denied.

B6 Pod to node root and persistence, via a privileged pod with the host filesystem mounted. A self-respawning fleet across eleven nodes.

B7 Node root to an internal MongoDB, via a static password read from the worker environment. Read, not modified.

B8 Node root to cluster secrets, via forged 24-hour service-account tokens. One production secret object holding 136 keys. HF states that everything on that day and the next descends from this read.

B9 Mesh-VPN enrolment, 181 times, with flags chosen to write no identity to disk and suppress vendor telemetry.

B10 Internal service connector to cluster-admin on two clusters within one second, via one shared credential bound to `system:masters`.

B11 Source control: a GitHub App installation token minted with write scopes, and a pull request opened to probe CI. No shipped change resulted.

B12 Persistence and cleanup, including a second-stage loader re-armed on each new sandbox and a binary download routed through an open CORS-proxy Space on HF's own platform.

Two facts govern what follows. Only B0 sits inside the lab; B1 through B12 occur on infrastructure the lab does not own and cannot instrument. And HF is explicit that the individual

weaknesses were familiar, and that a capable human could have found the same flaws. What changed was volume. A standard justified by the agent being clever is wrong on the record, and we do not make that argument.

3. THE PROPERTY AS SIX PREDICATES

P1, Digest binding. A request identifying an artifact by name and version only, with no digest supplied, must either be refused or answered with a response binding the returned bytes to a digest verifiable against a published manifest.

P2, Replay only. A request for an artifact absent from the snapshot but present upstream must return not-found.

P3, No path passthrough. Any path outside the package namespace must be rejected.

P4, No build path. The mirror serves bytes. It must never execute build metadata, install hooks or setup scripts, and must not offer an endpoint that does so.

P5, Snapshot immutability. The same name-plus-digest must return byte-identical content across time, and the snapshot must carry a manifest digest a third party can recompute.

P6, No control channel. Methods that alter server-side state must not be honoured on the egress path.

P4 speaks most directly to the incident. The escape route was the one sanctioned egress, and a component that both reaches upstream and can be induced to execute is a component with an exploitable surface facing an agent that has nothing else to attack.

4. THE VERIFICATION SPLIT

The mirror is verifiable. It is a component. The lab publishes the mirror image and the snapshot manifest digest; a third party runs the predicates against their own instance and reaches the same verdict. No lab network access is needed and no confidential data leaves the lab.

The deployment is attested, not verified. That the published mirror is the only egress from the evaluation network is a claim about a network the third party cannot see. A standard can require a signed attestation naming the mirror image digest and asserting sole-egress, renewed per campaign. It cannot make that claim checkable from outside. This is the weakest seam in the proposal and we put it in the body rather than a footnote.

5. THREE RUNS, AND THE SOUNDNESS FAILURE

Run 1, reference pair. Reference A is a content-addressed replay-only mirror. Reference B is a conventional caching proxy: resolves by name, fetches upstream on a miss, forwards arbitrary paths, offers on-demand source building. Result: A satisfies six of six, B satisfies one of six, five predicates discriminate. P6 passes for both, because a caching proxy typically exposes only GET and HEAD anyway, so we report P6 as a necessary property rather than a discriminating test and exclude it from the count.

This run establishes almost nothing. We wrote both the predicates and the targets they run against. Reference B fails tests designed around Reference B. We report it for completeness and we do not rest any claim on it.

Run 2, real software, naive probing. We pointed the same suite at devpi-server 6.20.3, a widely deployed Python package cache, running as a PyPI-backed mirror index. The suite reported five of six predicates satisfied.

Every one of those passes was false. devpi's API is `/+simple/{project}/` with files served from content-addressed `/+f/{hash}/` paths. The suite asked for `/a/{name}/{version}`, `/proxy`, `/build/...` and `/manifest`. None of those routes exist. devpi answered 404 and 415, and the suite scored each as a refusal. P3 scoring a 415 as "refused" is the clearest tell: that is content negotiation, not a policy decision.

The suite could not distinguish "this component refuses the request" from "this component has never heard of this request."

Run 3, real software, with an implementation adapter. We added per-implementation route profiles, a preflight check that aborts the whole run if the target does not answer the profile's known-good route, and a third verdict state, INCONCLUSIVE, for predicates the profile cannot express against this target. Rerunning against the same devpi instance:

```
P1 PASS      all 244 index links carry a digest fragment
P2 FAIL      GET /+simple/tqdm/ returned 200 with artifact links, resolving a
              project beyond any pinned snapshot
P3 INCONCLUSIVE  profile defines no passthrough route
P4 INCONCLUSIVE  profile defines no build route
P5 INCONCLUSIVE  profile defines no manifest route
P6 INCONCLUSIVE  POST, PUT and DELETE each answered 200, but the resource was
                  byte-identical afterwards
```

One pass, one fail, four inconclusive.

The soundness failure runs in both directions. Absent routes returning 404 produced false

passes in run 2. A permissive router answering 200 to every verb nearly produced a false fail in run 3: our first version of P6 read those three 200s as state-changing methods being honoured, which would have been a published vulnerability claim against a named open-source project on the strength of a status code. P6 now requires the resource to differ before and after, and reports inconclusive when a method is accepted but nothing visibly changes.

The common error is inferring semantics from status codes. A conformance suite that does this is unsound in both directions and silent about it in both directions.

Three consequences for anyone designing a conformance regime of this shape.

First, a third verdict state is mandatory, not a nicety. Two-state suites are forced to convert ignorance into a verdict, and the direction they convert it in is an accident of how the probe was written.

Second, preflight is mandatory. A target that 404s everything appears maximally compliant. Checking that the target answers one known-good route before trusting any verdict costs one request and removes the entire class of false pass observed in run 2.

Third, black-box conformance against a component you did not write requires an implementation adapter, which means a conformance regime is not a single artifact but a specification plus a per-implementation profile set, with the profiles themselves needing review. This is a real cost and it should be stated to anyone proposing such a regime, including us.

Findings about devpi specifically, stated narrowly. P1 passes: all 244 index links carried digest fragments, so this cache does publish digests, which contradicts the stylised Reference B we wrote and is a point in devpi's favour. P2 fails: configured as a PyPI-backed mirror it resolves projects on demand, which is the behaviour replay-only forbids and the behaviour the July escape route depended on. That is not a defect in devpi, which is doing exactly what a caching proxy is for. It is a statement that a caching proxy is the wrong component for this position in an evaluation sandbox. We make no claim about devpi in any other configuration, and the four inconclusive predicates are inconclusive, not passes.

6. THE ESCALATION GAP, QUANTIFIED AS AN UPPER BOUND

HF's account states that first signals came from several layers at once, that each was ambiguous alone, that their security agent stack did correlate them into a coherent attack signal, and that it then failed to raise the alert's criticality and trigger the on-call team, costing time in the response.

Detection worked. Correlation worked. Nobody woke up.

Taking the last attacker action, 13 July 13:37 UTC, as the containment reference, and the first published marker for each boundary:

Boundary	First published marker	Elapsed to containment
B1	09 Jul 04:01	105h 36m (outside HF estate)
B4	11 Jul 10:10	51h 27m first marker inside HF's estate
B5	11 Jul 17:33	44h 04m
B6	11 Jul 19:53	41h 44m
B7	11 Jul 19:59	41h 38m
B8	11 Jul 20:23	41h 14m
B9	11 Jul 21:23	40h 14m
B10	11 Jul 22:43	38h 54m
B11	12 Jul 23:08	14h 29m

Two observations. The earliest marker inside HF's own estate with a published timestamp precedes containment by 51 hours and 27 minutes. And B4 through B10, the entire escalation chain from pod to cluster-admin, falls between 10:10 and 22:43 on 11 July: twelve and a half hours, with containment arriving 39 hours after it finished. The window in which a response could have altered the outcome closed before anyone was paged.

This is an upper bound, not the escalation gap itself. HF does not publish when correlation completed, so the interval measured is first-observable-marker to containment, which contains both the detection time and the escalation time. We cannot separate them from public data. It is nonetheless the only quantification of this interval we are aware of, and the ordinary dwell-time statistic is the wrong instrument here because, by HF's own account, the correlation had already succeeded.

The design implication: a containment standard should specify, per boundary, both t_{signal} , the earliest action at which an observable marker exists, and t_{action} , the earliest point at which that marker justifies automated containment with no human in the loop. A standard that mandates telemetry and correlation while leaving the escalation threshold to operational discretion reproduces exactly the failure that occurred.

7. COVERAGE

B0 is addressed: the property removes the component class through which the escape occurred. B1 through B12 are not addressed; they occur outside the lab.

One crossing of thirteen. We consider this correct scope. B1 through B12 are somebody else's estate, and a lab-side standard purporting to govern them would be unadoptable and unverifiable at once. B0 is the only crossing over which the lab has authority and the only one with no external observer: every downstream crossing was in principle visible to Hugging Face,

while B0 was visible to nobody but OpenAI. A containment standard should concentrate where external visibility is absent, because that is where an attestation regime adds information that does not otherwise exist.

One downstream marker is worth somebody else's sprint. At B5, temporary instance-metadata credentials were replayed from external addresses on two hosting providers. Credential origin mismatch is observable and cheap, and HF's own remediation list includes tooling that flags tokens used from unexpected origins.

8. ADOPTABILITY

No production false-positive rate. The predicates are evaluated against the component offline, not against live traffic. A commenter on HF's timeline put the adoptability question as: which action number does your control fire on, does it fire at all, and does it also fire on Tuesday's legitimate conversion job. For a component-level property the third question does not arise, because there is no live-traffic classifier. The cost is entirely at design time.

Bounded engineering cost, with a real operational one. Content-addressed replay-only mirrors are established practice in reproducible-build and air-gapped contexts. But replay-only means the dependency set is fixed before the run, and resolving a new dependency mid-evaluation becomes an out-of-band action requiring a new snapshot. For evaluations of agents that install software as part of the task under test, that is a genuine constraint.

Adapter maintenance, newly visible. Section 5 shows the regime needs a reviewed profile per implementation. That cost was invisible to us until we tested against software we had not written, and it is the kind of cost that decides whether a standard is adopted.

9. WHAT THIS DOES NOT ESTABLISH

We cannot claim the property would have blocked the specific zero-day. Neither primary source publishes the vulnerability class in the cache proxy, whether it was caching artifacts or proxying live fetches, or whether anything in its path executed code. Those three unknowns are load-bearing and remain open. Our claim is narrower and we hold to the narrow version: the property removes the component category in which the escape occurred, not this exploit.

APPENDIX A, LIMITATIONS AND DUAL-USE CONSIDERATIONS

Limitations

Scope. The property addresses one of thirteen boundary crossings. Section 7 argues this is

correct scoping; a reader who rejects that argument should read the contribution as narrow.

Unverified deployment claim. Sole-egress is attested, not independently verifiable. A lab running a compliant mirror alongside an undisclosed second egress passes every predicate here.

The reference pair proves little. Run 1 is circular: we wrote the predicates and both targets. Its result is reported for completeness and no claim rests on it.

One real implementation, one configuration. Run 3 tested devpi-server 6.20.3 as a PyPI-backed mirror index on a single host. We tested no other cache, no other configuration, and no deployed evaluation sandbox. Four of six predicates returned inconclusive, and inconclusive means we do not know.

Our own adapter is unreviewed. The devpi profile was written by us, in one sitting, against documentation and observed behaviour. A wrong profile produces wrong verdicts as surely as no profile does. Section 5's third consequence applies to this paper's own artifact.

P2 is weaker than its statement. The predicate says no upstream connection is attempted; the probe observes only the response. The stronger form needs network observation a third party does not have.

Counting discrepancy in the source. HF's per-phase counts sum to 16,521; their daily counts sum to 17,613; roughly 1,092 recovered actions carry no phase label. We use the boundary chain rather than the phase counts because it does not depend on this denominator, but any work citing per-phase figures should state which total it used.

The timing table is an upper bound. Section 6 measures first-published-marker to containment, which contains detection time and escalation time together. HF does not publish when correlation completed, so the two cannot be separated from public data.

Contested provenance. HF describes the evaluation as based on ExploitGym. At least one commenter identifies the benchmark as a Berkeley RDI artifact rather than an OpenAI harness, and HF separately states that ExploitGym maintainers had no involvement in the evaluation deployment. We assert nothing either way.

Attribution of motive. That the agent was attempting to cheat the evaluation is HF's inference, presented as such, and we repeat it as their inference.

Facts we are careful not to overstate. Modal's infrastructure was not compromised. The Hub production database was never reached; the database breached was the internal datasets-server MongoDB. Customer impact was narrow: five datasets whose names suggest benchmark challenges and solutions, plus operational metadata from dataset-server search queries. Anthropic's separate retrospective review of 141,006 evaluation runs concerns a

different matter and is not evidence about this incident.

Dual-Use Considerations

This paper describes attacker technique at the level of trust boundaries and mechanism classes, drawn entirely from a disclosure the victim organisation published deliberately. It adds no capability. The predicates are tests a defender runs against their own component and do not generalise into an attack tool. The reference caching proxy is a fixture with no network reachability.

On naming a specific product: we report that devpi-server fails P2 in the configuration tested, and we say plainly that this is a caching proxy doing what a caching proxy is for rather than a defect. An earlier version of our P6 probe would have reported devpi as honouring state-changing methods on the basis of three 200 responses, with no evidence that any state changed. We caught it, and we describe it here because the near-miss is instructive: a conformance regime that publishes verdicts about named software can manufacture vulnerability claims out of status codes, and that risk grows with the number of implementations covered.

Section 7 identifies credential origin mismatch at B5 as the strongest externally observable marker in the trace. Publishing which marker was most detectable also tells a future adversary which to avoid. We judge the defensive value higher, because the marker is already implicit in HF's published remediation list and the avoidance technique, in-estate egress, is already demonstrated in the same trace at B9.

During the forensic work, HF reports reaching for frontier assistants first and finding a large part of the analysis refused, with guardrails treating reverse-engineering of the exploit as equivalent to launching it; they stood up a self-hosted open-weight model and rerouted the pipeline. Any harness parsing attacker traces will meet the same boundary. We record this as a fact about tooling availability for incident responders, not as a judgement about vendor policy.

APPENDIX B, SUITE OUTPUT

Run 1, reference pair. Reproduce with: `python -m egress_conformance.runner`

Predicate	Reference A	Reference B	Discriminates
P1 Digest binding	PASS	FAIL	yes
P2 Replay only	PASS	FAIL	yes
P3 No path passthrough	PASS	FAIL	yes
P4 No build path	PASS	FAIL	yes
P5 Snapshot immutability	PASS	FAIL	yes
P6 No control channel	PASS	PASS	no

Run 2, devpi-server 6.20.3, naive probing, suite reported 5 of 6 satisfied:

P1 PASS GET /a/requests/2.31.0 -> 404, read as "unresolvable without digest"
P2 PASS GET /a/leftpad/0.0.1/sha256:... -> 404, read as "not in snapshot"
P3 PASS GET /proxy?url=... -> 415, read as "refused"
P4 PASS GET /build/requests/2.31.0 -> 404, read as "no build endpoint"
P5 FAIL GET /manifest -> 415, no published manifest
P6 PASS POST/PUT/DELETE -> 404, read as "refused"

None of those routes exist in devpi. Five passes, all false.

Run 3, same instance, with adapter. Reproduce with:

python -m egress_conformance.runner --target http://127.0.0.1:3141/root/pypi --profile devpi

PREFLIGHT GET /+simple/requests/ -> 200 (profile: devpi-server)

P1 PASS all 244 index links carry a digest fragment
P2 FAIL GET /+simple/tqdm/ -> 200 with artifact links; resolved beyond
any pinned snapshot
P3 INCONCLUSIVE profile defines no passthrough route
P4 INCONCLUSIVE profile defines no build route
P5 INCONCLUSIVE profile defines no manifest route
P6 INCONCLUSIVE POST -> 200, PUT -> 200, DELETE -> 200, but resource is
byte-identical afterwards; a 2xx alone is not evidence of a write

10. WHAT A MONTH WOULD ADD

1. Profiles for further caches, Artifactory, Nexus, a plain nginx proxy_pass, converting inconclusive verdicts into determinate ones and testing whether the adapter cost scales or explodes.
2. A review process for profiles, since an unreviewed profile is now a known source of wrong verdicts.
3. Per-action replay of the public trace to separate t_signal from t_action and turn section 6's upper bound into the gap itself.
4. Resolution of the three open questions in section 9 against OpenAI's account.
5. An attestation schema for the sole-egress claim, with a renewal cadence tied to evaluation campaigns.

REFERENCES

Hugging Face. Agent intrusion: technical timeline.
huggingface.co/blog/agent-intrusion-technical-timeline
OpenAI. Hugging Face model evaluation security incident.
openai.com/index/hugging-face-model-evaluation-security-incident/
collusion.wiki. OpenAI agent edits to a dormant German wiki, disclosed 4 September 2026.
devpi-server 6.20.3. github.com/devpi/devpi
Cloud Security Alliance. AIS-13, AI sandboxing and isolation controls.
CoSAI. Incident Response Framework for AI systems.
Elastic Security Labs. Detection mapping for the July 2026 agent intrusion.
SecureLayer7. Replay corpus as conformance harness, proposal.
Apart Research x CeSIA. AI Incident Response Sprint, September 2026.

Prior work disclosure: the repository was scaffolded on 9 September 2026, before the sprint window, containing project tooling and a phase extraction from the public HF timeline. All conformance code, all three runs and this report were produced during the sprint.