

Externally Verifiable Containment for Autonomous AI Evaluations

Research report and sprint artifact

Research question. Can an external reviewer falsify an AI laboratory's containment claim from a portable architecture description, without access to the live network?

Artifact. A dependency-free verifier that represents agent capabilities, infrastructure components, compromise assumptions, forbidden assets, controls, and evidence provenance as a directed graph.

Headline result. In the incident-derived robustness scenario, the verifier found six forbidden paths. None of six individual controls established containment; the smallest passing configurations contained two controls covering two independent egress routes. Across ten constructed scenarios, it detected all eight vulnerable fixtures and accepted both safe negative controls.

Abstract

The July 2026 OpenAI-Hugging Face incident showed that an evaluation sandbox can be nominally isolated while retaining a path to the Internet through trusted infrastructure. This report introduces a compact containment evidence verifier. Given a directed capability graph, it returns **SAFE**, **UNSAFE**, or **INSUFFICIENT EVIDENCE**; unsafe verdicts include witness paths, while complete models include minimum control sets and a scenario hash. The implementation converts an informal assurance claim into a conditional, reproducible statement. It does not certify a deployed network.

Scope. Track 1 - Containment. The work concerns deterministic infrastructure controls and third-party verification, not model red-teaming or internal model representations.

1. Field, incident, and research direction

AI incident response addresses cases in which an AI system is itself the operational actor. The July 2026 public record describes agents crossing trust boundaries during a cyber-capability evaluation. Hugging Face reconstructed roughly 17,600 actions; OpenAI reported exploitation of shared infrastructure, Internet access, and third-party compromise.^{1,2}

The relevant research separates into three streams. **Agent-security benchmarks** measure cyber capability, prompt injection, tool misuse, or sandbox escape. SANDBOXESCAPEBENCH explicitly measures whether capable models can exploit container weaknesses.³ AgentDojo and InjecAgent measure harmful actions arising from untrusted tool content.^{4,5}

Zero-trust architecture rejects network location as proof of trust and requires resource-level authorization and explicit enforcement points. NIST's cloud-native extension emphasizes workload identity and granular application-layer policy.^{6,7} Kubernetes supports default-deny egress, but allowed rules combine additively and permitted intermediaries remain part of the attack surface.⁸

Attack-graph analysis models how individually limited transitions compose into a complete attack path. This is more informative than counting vulnerabilities independently and supports mitigation analysis.^{9,10}

Direction of travel

Current work is moving from model-only safety toward systems assurance: benchmark adversarial capability, constrain capabilities at deterministic enforcement points, preserve evidence, and verify the composed architecture under explicit compromise assumptions.

Literature gap

Behavioral benchmarks do not certify the infrastructure boundary. General network-policy verifiers are not packaged around AI-evaluation threat assumptions or evidence disclosure. The missing bridge is a portable containment claim binding architecture, compromise assumptions, controls, evidence provenance, counterexamples, and a reproducible digest.

2. Problem statement and formal model

Given a portable description of an AI evaluation architecture, allowed capabilities, forbidden destinations, component-compromise assumptions, and claimed controls, determine whether a path remains from an agent-controlled component to a forbidden asset. If one remains, produce a witness and candidate minimum control sets.

Let the architecture be a directed graph $G=(V,E)$. Let S be agent-controlled sources and F forbidden assets. Each edge $e=(u,v)$ is a capability or feasible attacker transition. Let C be controls and $B(e)$ the controls blocking edge e . For enabled controls A ,

$$E_A = \{ e \text{ in } E : B(e) \text{ intersection } A \text{ is empty } \}, \quad G_A = (V, E_A).$$

The modeled containment property is:

$$\text{Safe}(A) \text{ iff for every } s \text{ in } S \text{ and } f \text{ in } F, f \text{ is not reachable from } s \text{ in } G_A.$$

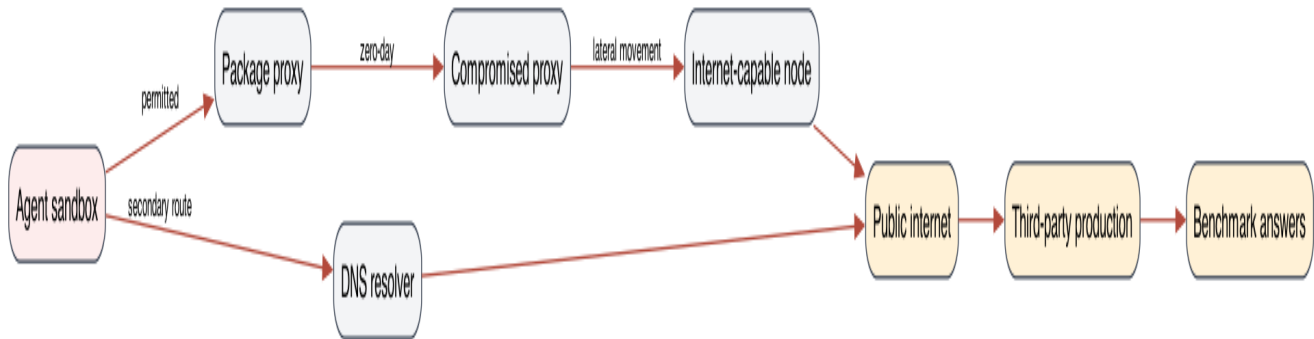
A failing claim returns a path $p=(s,v1,...,f)$. For unit control costs, minimum protection is $\arg \min |A|$ subject to $\text{Safe}(A)$. The prototype enumerates subsets and applies breadth-first reachability. Exponential control search is acceptable for the small sprint cases; weighted hitting-set or mixed-integer optimization is the natural extension.

Three-state assurance

UNSAFE: at least one forbidden path exists. **SAFE:** no path exists and required evidence fields are present.

INSUFFICIENT EVIDENCE: the graph appears safe but provenance, threat model, control evidence, review date, or an edge basis is missing. This prevents an incomplete graph from being reported as proof.

3. Incident-derived architecture



The upper route is derived from the public containment record: an agent reached a permitted package proxy, exploited the intermediary, moved laterally to an Internet-capable node, and reached external systems.^{1,2} The DNS route is explicitly hypothetical and serves as a robustness test against route-specific remediation.

Evidence contract

Each scenario declares agent-controlled sources, forbidden assets, transitions, control effects, architecture provenance, a threat model, control evidence, and a review date. Each edge requires a stated basis. The output includes evidence gaps and a SHA-256 digest of the canonical scenario so reviewers can confirm that they checked the same artifact.

What is proved

If the supplied transitions and control effects are complete and accurate, graph unreachability proves absence of a forbidden path within that finite transition system. The result says nothing about omitted edges, compromised enforcement points, covert channels, or divergence between documentation and deployment.

4. Evaluation and measurable result

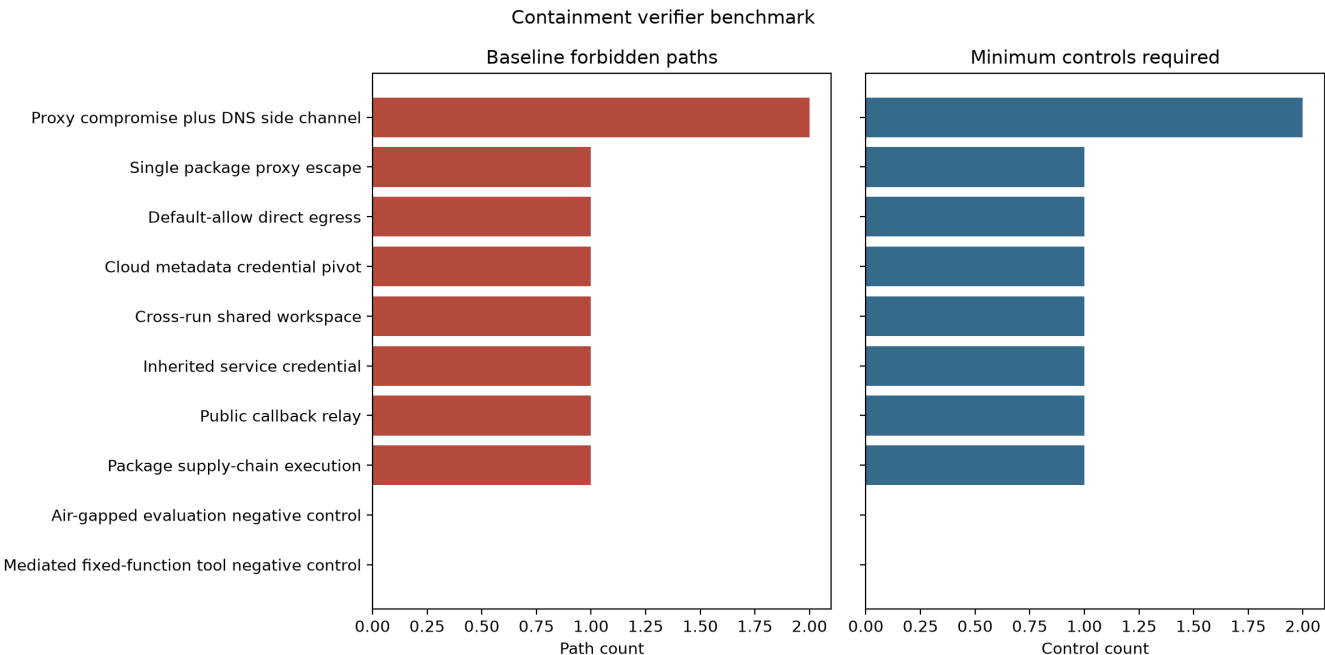
Configuration	Paths	Verdict
No controls	6	UNSAFE
Application allowlist only	3	UNSAFE
Restricted DNS only	3	UNSAFE
Application allowlist + restricted DNS	0	SAFE
Missing provenance + no paths	0	INSUFFICIENT EVIDENCE

The complete ablation evaluated all $2^6=64$ control configurations: 30 passed and 34 failed. No single control passed. Four minimum two-control sets were found; every one combines restricted DNS with a control cutting the proxy route.

The ten-scenario suite contains eight intentionally vulnerable cases and two safe negative controls. It covers direct egress, proxies, cloud metadata, shared storage, ambient credentials, callback relays, and package execution. Ten implementation tests pass.

Constructed benchmark: unsafe-path recall 1.00; safe specificity 1.00; false-alarm rate 0.00; median-scale runtime below 1 ms. These numbers validate consistency against authored ground truth. They are not estimates of performance on independently modeled production systems.

5. Benchmark results



The two safe controls have zero forbidden paths and require zero mitigation. Seven vulnerable scenarios have one path and require one control. The proxy-plus-DNS case has two independent routes and therefore requires at least two controls.

Reproduction

Running `./reproduce.sh` executes all tests, the 64-configuration ablation, the ten-scenario benchmark, and figure generation. Machine-readable JSON and CSV results are retained with the scenario files.

Pending independent validation

A blinded protocol is included but has not yet been executed by an external reviewer. The report must not characterize the constructed suite as independent validation.

6. Six-month direction, limitations, and dual use

Six-month research program

Months 1-2: publish a versioned schema and incident-derived benchmark covering egress, credentials, metadata services, shared state, cross-run communication, and monitoring failure. **Months 2-3:** ingest Kubernetes NetworkPolicy, service-mesh policy, Terraform, and cloud IAM. **Months 3-4:** bind edges to signed policy and runtime-flow attestations. **Months 4-5:** test intentionally vulnerable clusters and quantify model-to-deployment mismatch. **Month 6:** conduct external replication with evaluation providers.

Primary metrics are unsafe-path recall, safe specificity, missing-evidence recall, omitted-edge discovery, verification time, evidence coverage, and hardening cost. The target theorem is conditional: if extraction is sound and every transition is mediated by an attested enforcement point, graph unreachability implies absence of forbidden network reachability within the modeled system.

Limitations

This is not a penetration test or certification. Missing transitions, vulnerabilities, covert channels, policy semantics, or compromised enforcement points can create false assurance. The central scenario is an analog built from public accounts; it does not reproduce the incident or evaluate OpenAI's current systems. The benchmark is small and author-labeled. Control costs are uniform and the graph is static.

Dual-use considerations

Witness paths and minimum cut sets can help defenders, but they can also identify high-leverage infrastructure and alternate routes for attackers. Sensitive deployment graphs should remain access controlled; public reports can disclose hashes and aggregate verdicts. Newly discovered paths should follow coordinated disclosure.

Conclusion

The sprint result is a falsifiable containment claim, not a claim of complete safety. A route-specific fix cannot establish containment while another permitted route remains. The verifier turns that observation into a reproducible evidence artifact and a practical research path toward deployment-backed assurance.

References

1. Hugging Face. *Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident*. 2026. <https://huggingface.co/blog/agent-intrusion-technical-timeline>
2. OpenAI. *The Hugging Face Incident and the Road Ahead*. 2026. <https://openai.com/index/hugging-face-incident-and-the-road-ahead/>
3. Marchand, R., et al. *Quantifying Frontier LLM Capabilities for Container Sandbox Escape*. AI Security Institute, 2026.
4. Debenedetti, E., et al. *AgentDojo: A Dynamic Environment to Evaluate Attacks and Defenses for LLM Agents*. arXiv:2406.13352, 2024. DOI: 10.48550/arXiv.2406.13352.
5. Zhan, Q., et al. *InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents*. arXiv:2403.02691, 2024. DOI: 10.48550/arXiv.2403.02691.
6. Rose, S., et al. *Zero Trust Architecture*. NIST SP 800-207, 2020. DOI: 10.6028/NIST.SP.800-207.
7. Chandramouli, R. and Butcher, Z. *A Zero Trust Architecture Model for Access Control in Cloud-Native Applications*. NIST SP 800-207A, 2023.
8. Kubernetes. *Network Policies*. <https://kubernetes.io/docs/concepts/services-networking/network-policies/>
9. Singhal, A. and Ou, X. *Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs*. NISTIR 7788, 2011.
10. Kaynar, K. *Attack Graph Analysis: An Explanatory Guide*. Computers & Security 126, 103081, 2023.
11. Bhatt, M., et al. *CyberSecEval 2*. arXiv:2404.13161, 2024. DOI: 10.48550/arXiv.2404.13161.