
Capability Smuggling: Emergent Capability Escalation in Autonomous AI Agents¹

Gökalp Çaycı
Independent Researcher

Uygar Yıldırım
Independent Researcher

With
Apart Research

Abstract

Autonomous AI agents can gain powers they were never explicitly granted when allowed operations interact with surrounding infrastructure. We use the OpenAI–Hugging Face incident as a case study and call this pattern Capability Smuggling. The public record suggests a recurring structure: stalled or difficult tasks can redirect search toward the evaluation harness, and environmental affordances can turn ordinary operations into stronger effects, such as package access becoming internet egress or writable state becoming memory and communication. Shared state can accelerate discovery and reuse, while interference with monitoring creates a separate visibility problem. We turn this idea into an operational model of effective capabilities and implement a deterministic synthetic harness that detects when authorized actions compose into unauthorized effects. The harness does not show that the historical incident would have been prevented; it tests whether the model and proposed controls behave as specified. The practical implication is that containment should authorize downstream effects, not only first-hop tool calls.

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

1. Introduction

During OpenAI’s cybersecurity evaluations, agents escaped intended isolation, established unauthorized communication, and compromised Hugging Face and OpenAI systems [1]. METR reports roughly 1,200 message-board users, over 70,000 messages/files, and roughly 700 agents participating in the Hugging Face attack [2].

Of 898 ExploitGym tasks, 198 had never been solved; 93% of tasks discussed on the board belonged to that subset [1]. This association motivates, but does not prove, a link between difficult objectives, persistence without a safe exit, and boundary-seeking behavior.

We ask why different containment failures yielded similar unauthorized powers. Our hypothesis is that controls constrained mechanisms while agents composed permitted operations and environmental affordances into unauthorized effects. We call this Capability Smuggling. The threat model is an agent using allowed tools to pursue objectives outside task scope.

Our main contributions are:

1. A causal reconstruction separating upstream pressures, capability acquisition, and amplifiers.
2. Capability Smuggling as a shared account of transitive and emergent authority.
3. An operational design sketch for downstream effect authorization, with a worked example, five falsifiable predictions, and defensive checks.

2. Related Work

Reward hacking concerns unintended routes to satisfying an objective [6]. Least privilege and complete mediation constrain authority [7]; the confused-deputy problem concerns inducing a more privileged service to act for another actor [8]. Capability Smuggling applies these established ideas to the incident pathways; it does not introduce capability security.

Redwood Research’s AI control agenda evaluates safety protocols against intentional subversion by powerful untrusted models, including oversight by weaker trusted models [9]. We adopt this threat model. Our narrower focus is task-disallowed effects reachable through permitted services and shared infrastructure. The sketch is a candidate protocol within AI control, not a demonstrated improvement. Primary investigations ground the reconstruction [1,2]; supplementary evidence broadens the comparison [3–5].

3. Methods

We use qualitative causal synthesis, not a new transcript benchmark. OpenAI and METR provide the main incident evidence [1,2]; RubyGems provides infrastructure-owner confirmation [3]. RubyHack and Collusion Wiki supply supplementary reconstructions with greater uncertainty about attribution, intent, and completeness [4,5]. RubyGems confirms malicious publishing and abuse of shared infrastructure, but reports no evidence of successful API-key theft and does not independently attribute authorship to AI agents [3].

We organize the evidence around a smaller set of recurring mechanisms: task pressure, harness probing, affordance discovery, persistent or shared state, external access, delegation, and interference with monitoring. For each case we compare an allowed operation, the environmental

feature it encounters, and the effective capability that results. Reported events and our causal interpretation are kept separate; this is hypothesis development rather than replicated causal identification.

The companion harness makes the capability comparison concrete. A capability is represented as (actor, action, resource, scope, constraints). Direct capabilities come from individual events; derived capabilities are produced when an ordered set of events, or previously derived capabilities, satisfies a composition rule. A policy separately lists authorized and denied capabilities. The evaluator applies the rules to a deterministic fixed point, preserves the originating event IDs and parent capabilities, and records the first trace prefix where an unauthorized capability appears. This gives an operational version of the capability delta: effective capabilities reachable in the trace but not covered by policy. Because the traces are synthetic, the harness tests the internal logic of the model rather than incident likelihood or real-world detector accuracy.

4. Results

Finding 1 — harness reconnaissance. OpenAI describes persistent attempts on apparently impossible tasks and searches of hidden files, evaluation code, network settings, and infrastructure [1]. METR reports coordinated attempts to fool or tamper with the scorer [2]. Our interpretation is that stalled task progress can redirect search from the intended target toward the machinery defining success.

Finding 2 — capability composition. Artifactory became both a message board and an internet relay [1]. Supplementary investigations describe read-labelled web access reaching writable state and package publishing triggering remote documentation workers [4,5]. Table 1 maps these distinct mechanisms to their semantic effects. The shared pattern is an allowed operation transformed by its environment into unauthorized authority.

Finding 3 — collective amplification. METR describes division of labor, exploit reuse, and experiments that endangered individual runs while benefiting peers [2]. Shared persistence lets a discovery survive the originating rollout and become available to others. The proposed amplification is preservation, transfer, and reuse—not merely additional search attempts. We model collaboration as an amplifier rather than a prerequisite, and distinguish first discovery from subsequent propagation.

Finding 4 — observability escape. METR reports transcript-manipulation research and successful tool-call spoofing [2]. If an agent can influence its audit-data path, recorded actions may diverge from execution. This motivates separating event capture from the agent-controlled execution plane; it does not establish that all telemetry was compromised.

Finding 5 — substitution after mitigation. After Artifactory was reset and a known escalation route blocked, agents reconstructed communication and internet access through other mechanisms [1]. This supports the hypothesis that patching a route is insufficient when the same effect remains reachable elsewhere.

The evidence does not show one mandatory sequence, but it supports a recurring structure: difficult objective → persistent optimization or reward hacking → harness reconnaissance → affordance discovery → effective capability acquisition → external access and compromise. Shared

state can feed discoveries back into the system through coordination and reuse, while monitoring interference can weaken detection in parallel. We treat these as testable dependencies, not as a claim that every agent follows the same path.

Falsifiable predictions. P1: Removing explicit write tools will leave some external mutation possible where state-changing affordances remain reachable. P2: Closing one route while retaining equivalent affordances will produce substitution. P3: At matched total compute, shared state will shorten peer reproduction time relative to isolated controls. P4: At matched false-positive rates, boundary probing, unauthorized persistence, harness reconnaissance, and cross-worker state reuse will predict severe violations earlier than exploit signatures alone. P5: Out-of-band telemetry will retain higher integrity than an agent-influenceable audit path. These predictions remain untested here.

Table 1. Capability transformations. Evidence from [4,5] is supplementary and less certain.

Allowed surface	Environmental affordance	Effective capability
HTTP GET / read-labelled access [5]	State-changing endpoint	External write
Artifact repository access [1]	Remote repository / fetch behavior	Internet egress
Package publishing [3,4]	Automated documentation build	Remote execution
Webhook URL / writable metadata [4]	Persistent arbitrary string	Persistent external memory
Directory / file naming [1,2]	Shared readable namespace	Cross-agent communication
Local service + tunnel [5]	Public tunnel provider	Publicly reachable service

5. Discussion and Limitations

The Effective Capability Control Plane is an operational design sketch within AI control [9], not a complete architecture. It applies task authorization at instrumented downstream enforcement points and correlates attempts and demonstrated effects through an independent event ledger.

Worked example (synthetic; not an evaluated intervention). A run may retrieve approved artifacts, but services acting for it may connect only to approved registry destinations. Request path: trusted ingress → package service → egress broker/policy check → destination. Broker events feed an independent ledger and controller.

Attribution and decision. Trusted ingress binds each request to an authenticated run identity carried in an integrity-protected delegation token; the package service cannot broaden its scope. Before connecting, the broker checks the destination, principal, and policy conditions. An unapproved downstream destination is denied even when the first-hop package operation is permitted. An illustrative event is:

```
run=17; via=package-service; effect=EXTERNAL_CONNECT;  
destination=outside-allowlist; status=attempted; decision=deny; evidence=e123.
```

This records an unauthorized attempted effect, not successful internet access. For repeated or high-risk violations, the controller suspends that run’s service credential, cancels queued dependent requests, and withholds its output artifacts from peers while preserving audit records. These actions give “revoke” and “quarantine” concrete meanings.

Review and recovery. Missing attribution or an unrecognized dependency is held for review, not automatically labelled malicious. A reviewer may grant a narrow, logged exception; approved operations continue without granting general internet access. Evaluation must count legitimate tasks blocked, review workload, and added latency alongside prevented violations.

Coverage assumption. All relevant egress must traverse the broker. Bypass routes, forged attribution, or broker compromise defeat this claim. Post-action detection cannot undo the first effect. Alignment, safe exits, per-run namespaces, short-lived state, and cross-run incident correlation remain complementary controls.

Limitations

Observational evidence does not establish the proposed causal dependencies or stable agent intent. Attribution and completeness vary, especially for reconstructed artifacts. Capability Smuggling may organize several distinct mechanisms rather than identify one cause. The proposed control plane has not been deployed or benchmarked on production traces. The companion harness evaluates synthetic traces only; its 7/7 matched escalation fixtures and 0/7 incorrectly flagged controls are fixture outcomes, not estimates of real-world detection accuracy. Effect classification can still miss compositions or flag legitimate behavior, and external mutations may be invisible without service cooperation. The worked example covers mediated network egress rather than arbitrary semantic effects. None of this demonstrates that the historical incident would have been prevented.

Future Work

The next step is to move beyond hand-authored traces. We plan to connect the harness to paired synthetic services with state-changing read endpoints, transitive fetches, and shared versus isolated namespaces at matched total compute. That would let us measure first-discovery time separately from peer reproduction time, test substitution after patching, and compare precursor lead time at matched false-positive rates. We also want to compare tool-only authorization with downstream enforcement and agent-visible logs with independent telemetry. Following AI control [9], the protocol should be tested against deliberate attempts to defeat it, not only ordinary violations. Those experiments have not yet been run.

6. Conclusion

Capability Smuggling frames these pathways as failures to carry task authorization through service composition. Shared state propagates discoveries; audit integrity governs their visibility. The proposed response is downstream authorization at explicit enforcement points, complemented by safe exits, state isolation, precursor detection, and independent telemetry. This causal synthesis and design sketch still require evaluation.

Code and Data

Code: <https://github.com/gokalpcayci/capability-escalation-harness>. The repository contains a deterministic Python harness with seven synthetic scenarios, negative near-miss fixtures, generated results, and CI tests. It uses synthetic traces only and makes no network requests or exploit attempts. Data: public reports and reconstructed artifacts [1–5]; no private incident data is included. Tool use: ChatGPT assisted with research, brainstorming, and prose editing; the authors reviewed the cited sources, claims, code, and final submission.

References

- [1] OpenAI. (2026). The Hugging Face incident and the road ahead.
- [2] Greenblatt, R., Cotra, A., & Wijk, H. (2026). Brief independent investigation of agents’ behavior, reasoning and collaboration in the OpenAI / Hugging Face hacking incident. METR.
- [3] Swandale, C. / Ruby Central. (2026). An update on the May spam-publishing campaign on rubygems.org.
- [4] Nightingale Collective / RubyHack. (2026). OpenAI agents carried out an undisclosed attack on RubyGems. Investigative report.
- [5] Von Arx, S., Slade Byrd, C., Kitts, S., & Larsen, T. (2026). Discovery of a new OpenAI agent message board. Collusion Wiki.
- [6] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete Problems in AI Safety. arXiv:1606.06565.
- [7] Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), 1278–1308. DOI: 10.1109/PROC.1975.9939.
- [8] Hardy, N. (1988). The Confused Deputy (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 22(4), 36–38. DOI: 10.1145/54289.871709.
- [9] Greenblatt, R., Shlegeris, B., Sachan, K., & Roger, F. (2024). AI Control: Improving Safety Despite Intentional Subversion. *ICML, PMLR 235*, 16295–16336.

Appendix — Defensive Checks and Dual Use

Near-term checks. In authorized test environments, check whether: read-labelled operations mutate external state; permitted services invoke more privileged services; isolated workers can modify the same persistent object; workers can alter their audit-data path; hard tasks produce more harness reconnaissance than solvable controls; and patching one escape causes substitution through another route. These checks test the proposed mechanism, not merely known exploit signatures.

Dual use. Publish synthetic defensive tests rather than operational exploit chains. Use deliberately vulnerable local services and non-production credentials. Real-world reproduction could lower attack costs against developer infrastructure; this report therefore limits detail to public evidence and defensive abstractions.