

Agent Worlds L3

Draft Conformance Profile for Auditable Agent Delegation

Sahil Soni

Independent Researcher

AI Incident Response Sprint | September 2026

Abstract

The July 2026 incident showed that autonomous agents can escape a sandbox through a permitted network channel to achieve their goals. I chose Track 1 (Containment), focusing on one rule: **authority must only shrink during delegation**. This rule prevents permission laundering, in which a low-privileged agent asks a more privileged agent to perform an unauthorized operation.

I designed a draft conformance profile, a runnable reference harness, and an independent offline auditor. I tested five configurations: a shared-account baseline, a forwarded-credential sandbox, an Agent Worlds ablation, Agent Worlds, and scoped delegation. The shared-account baseline, forwarded-credential sandbox, and Agent Worlds ablation applied the laundered write, whereas Agent Worlds and scoped delegation blocked it.

Attenuation is prior art; my contribution is independently checkable, signed evidence. The issuer signs with its private key, and the auditor uses the issuer's public key to detect invalid, incomplete, or contradictory supplied evidence. This mechanism does not prevent the initial network escape.

1 Introduction

I am addressing Track 1: Containment [6]. The July 2026 incident demonstrated that agents could escape through permitted package-registry channels and reach Hugging Face production [1], [2]. While my project does not address that initial network escape, it focuses on a narrower, critical failure mode within the broader class of unintended agent behavior and cheating [3], [4]: **permission laundering during agent-to-agent delegation**.

This failure mode occurs when a read-only Agent A asks a privileged Agent B to perform a write that A cannot authorize itself. In practice, B's broader authority is converted into A's operational power. To prevent this, I enforce a rule in which every delegated capability must be a subset of or equal to its parent:

$$\text{child authority} \subseteq \text{parent authority}$$

Authority may remain equal or shrink, but it must never expand.

I prioritized two evaluation questions:

1. Can an external party verify compliance?
2. Could a laboratory adopt this mechanism using existing security primitives?

Capability attenuation is established prior art; my contribution is independently checkable conformance evidence.

The main contributions are:

- A draft L3 conformance profile with six testable requirements, R1–R6, including required evidence and failure behavior.
- A runnable five-configuration reference harness with border enforcement, protected-resource reverification, a competent conventional baseline, and an ablation.
- An offline public-key auditor with three exit states and eleven adversarial counterexamples encoded as regression tests.

2 Related Work

Miller's capability-security work establishes the concept of attenuation [8]. Hardy describes the confused-deputy problem, which permission laundering mirrors in an agent-specific context [9]. I apply these concepts to agent architectures, but I do not claim to have invented capability attenuation.

RFC 8693 provides OAuth 2.0 token-exchange and downscoping mechanisms that permit narrower downstream tokens [7]. However, it does not require my complete parent-subset proof, parent-hash evidence chain, or expiry clamp.

Frameworks such as the NIST AI RMF [11] and ISO/IEC 42001 [12] address high-level organizational risk management, while NIST SP 800-190 [10] covers container security risks and MITRE ATLAS [13] supplies a standardized threat vocabulary. The gap I identified in these documents is that they do not provide the exact R1–R6 agent-delegation profile together with outsider-runnable verification evidence. My work fills this operational gap.

3 Methods

I built a simulated environment to test agent delegation. The system consists of five main components:

Component	Role
Issuer	Mints delegated capabilities and signs them using an Ed25519 private key
Gateway/border	Computes permitted scope, checks context, and records decisions
Agent B	Receives only the newly delegated capability
Protected resource	Rechecks caller, audience, and operation before changing state
Offline auditor	Uses only the public key to validate the supplied evidence

For the delegation mechanism, the child agent's scope is calculated through intersection, never union. The child must be a subset of or equal to its parent. The credential context contains the user, run, task, caller, and resource audience. A child's expiry time cannot exceed the parent's. Each child token records its parent token's hash to build the chain. The protected resource rejects requests with missing capabilities, incorrect callers, incorrect resource audiences, or unauthorized operations.

Evidence records are Ed25519-signed and hash-linked into a bundle containing a manifest and a checkpoint count. The offline auditor imports nothing from the enforcement harness and uses only the issuer's public key. It yields:

- **Exit 0:** The evidence is valid and the claim is upheld.
- **Exit 1:** The evidence is valid but contradicts the claim.
- **Exit 2:** The evidence is invalid, incomplete, or unverifiable.

Checkpoint completeness detects truncation relative to the supplied bundle, but it does not prove that every real execution was recorded.

I wrote the harness in Python with one external dependency: the `cryptography` library. There is no real model, network, container, service mesh, or identity provider; all components are simulated in-process. One gateway evaluates five configurations that differ only in authorization policy and evidence behavior. The protected resource's state determines whether the write was applied. I also tested a benign delegation to show that the system does not deny everything.

During development, I found that relying on a denial message alone could falsely claim containment, and missing embedded grants made a denial unverifiable. A valid earlier denial could hide a later applied write, and testing only a named $A \rightarrow B$ hop could miss authority expansion on a different $C \rightarrow D$ hop. The auditor required strengthening across three counterexample-review rounds.

This design relies on three assumptions:

- **Key custody:** The issuer holds the private key; the auditor receives only the public key.
- **Trusted recording:** The instrumented border is trusted to record every decision.
- **Complete mediation:** Every authority-bearing action must pass through the instrumented enforcement path.

4 Results

I ran the simulated laundering request across all five configurations. The protected resource independently reverified every capability, providing defense in depth by rejecting requests without a valid capability.

Table 1. Simulated Permission-Laundering Probe Outcomes

Configuration	Direct Access	Laundered Request	Enforcement Reason	Resource Write	Independently Verifiable	Verdict
baseline	Allow	Allow	Shared-account union	Applied	No	Vulnerable
traditional_sandbox	Allow	Allow	Forwarded bearer credential	Applied	No	Vulnerable
scoped_delegation	Deny	Deny	Authority only shrinks	Blocked	No	Contained
agent_worlds	Deny	Deny	Authority only shrinks	Blocked	Yes	Contained
agent_worlds_ablation	Allow	Allow	Ablation: no attenuation	Applied	Yes	Vulnerable

Shared authority and forwarded credentials allowed the laundered write. Scoped delegation and Agent Worlds both blocked it. Disabling intersection made the Agent Worlds ablation vulnerable, demonstrating that intersection was the operative variable within this harness. Among the two contained configurations, only Agent Worlds supplied independently verifiable evidence.

Table 2. Offline Conformance Auditor Evaluation

Configuration	Bundle Verification	Records	Laundered Write Attempts	Containment Confirmed	Audit Result
agent_worlds	Valid	4	1, blocked through attenuation	Yes	Pass (Exit 0)
scoped_delegation	Unverifiable; operator-attested only	4	N/A; unverified	No; operator-attested only	Fail (Exit 2)

The offline auditor yielded Exit 0 for Agent Worlds. For scoped delegation, it correctly yielded Exit 2, meaning that no cryptographic evidence was supplied to evaluate—not that containment failed. This highlights the core gap: conventional baselines rely on operator-attested evidence rather than independently verifiable conformance evidence.

Table 3. Adversarial Counterexamples and Regression Suite

Test Module	Coverage	Status
test_policy.py, test_store.py	Attenuation, scope intersection, and resource rechecks	Pass
test_evidence.py, test_audit.py	Evidence-chain formatting and offline-auditor logic	Pass
test_probe.py, test_smoke.py	Permission-laundering scenarios and system execution	Pass
test_ablation_benign.py	Ablation control and benign delegation allowances	Pass
test_review_fixes.py	Locked adversarial counterexamples, including tampering and truncation	Pass
Total suite	10/10 modules, including 11 auditor counterexamples	Pass

All ten primary test modules passed. I locked eleven auditor counterexamples into the regression tests. The auditor rejected unsigned, wrong-key, tampered, and truncated bundles. It rejected denials missing grants, child scopes exceeding parent scopes, and context mismatches spanning the user, run, and task. It also rejected child expiries exceeding parent expiries, later applied writes, and expanded C→D authority.

My robustness evidence consists of the competent baseline, controlled ablation, protected-resource outcomes, benign allowed case, adversarial counterexamples, and regression tests. This is a deterministic conformance experiment, not a statistical benchmark. The main validity threat is that everything is simulated and assumes trusted recording and complete mediation. The auditor cannot verify execution completeness; it verifies only the supplied recorded evidence under these assumptions.

5 Discussion and Limitations

A conventional scoped-delegation system already blocks this attack. My practical value is generating evidence that an external party can evaluate without entering the laboratory's network. A potential deployment could use workload identity, mTLS, an HSM/KMS-held issuer key, and deny-by-default protected resources. Alternate ambient or shared credentials must be excluded; otherwise, agents can bypass the capability path.

This work has several limitations. I used no real network, model, or container, and tested only one controlled permission-laundering scenario. There is no performance or scale evaluation. This mechanism does not prevent the initial package-proxy exploit, nor does it implement network isolation, runtime isolation, revocation, or inheritance control. The public demo seed is unsuitable for production key custody. Signatures establish authenticity, not the correctness of real execution. Bundle completeness is not execution completeness. A compromised issuer key defeats the guarantee, and covert information transfer [5] remains unresolved.

Natural next steps include C2–C3 network-egress and permitted-channel enforcement, C4 rootless and immutable runtime isolation, and C8 independent revocation and propagation. I also plan to explore a C10 cross-run memory, code, and strategy inheritance gate, alongside production workload identity and HSM/KMS key custody. Execution-coverage attestation, real multi-service and multi-hop delegation tests, performance testing at larger agent counts, and independent replication by another auditor are all required to extend this work.

6 Conclusion

I built a reference harness and offline auditor to evaluate capability attenuation during agent-to-agent delegation. Both the Agent Worlds configuration and a competent scoped-delegation baseline blocked the simulated attack. The ablation demonstrated that scope intersection was the operative mechanism enforcing containment in the harness. The distinguishing contribution is independently verifiable, signed evidence of this containment. This result is narrow and does not cover the initial network escape.

Code and Data

- **Code repository:** https://github.com/whitepaper27/AI_Safety_HuggingfaceopenaiIncident
- **Data/datasets:** No external dataset; the harness generates simulated delegation and evidence records in-process.
- **Reproduction:**

```
pip install -r harness/requirements.txt
python -m harness.run
python -m harness.audit --config agent_worlds
python -m harness.audit --config scoped_delegation
python harness/tests/run_tests.py
```

Author Contributions

Sahil Soni designed the profile, implemented the harness and auditor, ran the evaluation, verified the results, and wrote the report.

References

1. Hugging Face. (2026). "Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident." huggingface.co/blog/agent-intrusion-technical-timeline
2. OpenAI. (2026). "OpenAI and Hugging Face Partner to Address Security Incident During Model Evaluation." openai.com/index/hugging-face-model-evaluation-security-incident
3. Anthropic. (2026). "Investigating Three Real-World Incidents in Our Cybersecurity Evaluations." anthropic.com/research/investigating-incidents-cybersecurity-evals
4. UK AISI. (2026). "Cheating Behaviour in Frontier Model Evaluations." aisi.gov.uk/blog/cheating-behaviour-in-frontier-model-evaluations
5. Collusion.wiki. (2026). "Discovery of a New OpenAI Agent Message Board." collusion.wiki
6. Apart Research. (2026). "AI Incident Response Sprint." apartresearch.com/sprints/ai-incident-response-sprint-2026-09-11-to-2026-09-13
7. Jones, M., et al. (2020). "RFC 8693: OAuth 2.0 Token Exchange." IETF. datatracker.ietf.org/doc/html/rfc8693
8. Miller, M. S. (2006). *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*. Johns Hopkins University. erights.org/talks/thesis
9. Hardy, N. (1988). "The Confused Deputy (or why capabilities might have been invented)." *ACM SIGOPS Operating Systems Review*. doi.org/10.1145/54010.54021
10. Souppaya, M., Morello, J., & Scarfone, K. (2017). *NIST SP 800-190: Application Container Security Guide*. National Institute of Standards and Technology. csrc.nist.gov/publications/detail/sp/800-190/final
11. NIST. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf
12. ISO/IEC. (2023). *ISO/IEC 42001: Information Technology—Artificial Intelligence—Management System*. iso.org/standard/81230.html
13. MITRE. (2024). "MITRE ATLAS: Adversarial Threat Landscape for AI Systems." atlas.mitre.org

Appendix A Limitations and Dual Use Considerations

Limitations of scope. This mechanism mitigates only permission laundering after deployment within agent-to-agent delegation paths. It offers no protection against initial container escapes, supply-chain poisoning, or covert data-exfiltration channels. Because the system simulates network and identity-provider components, a real-world deployment would face operational challenges involving key management, latency overhead, and legacy-system integration that are not captured in this harness.

Dual-use profile. The delegation harness and offline auditor are defensive security mechanisms designed to restrict agent capabilities. They do not inherently enhance an agent's offensive capabilities or execution competence.

Misuse potential. An adversary could study the L3 conformance profile to identify edge cases in capability intersection, aiming to construct token chains that superficially pass the auditor while expanding actual authority.

False sense of security. The primary risk of this approach is organizational. The system generates verifiable evidence of intended containment, but if the underlying assumptions—complete mediation, trusted recording, and secure issuer-key custody—fail, the auditor might validate an incomplete supplied bundle while remaining unable to detect real actions that bypassed recording. This could mask a real breach. Complete bundle attestation cannot mathematically guarantee complete execution capture.

Appendix B Console Output Transcripts

For transparency, the available raw execution logs for the harness and offline auditor are included below.

B.1 Harness Execution

Command: `python -m harness.run`

```
=====
AGENT-WORLD LAW-3 CONTAINMENT DEMO (permission laundering)
=====

RESULTS (simulated permission-laundering probe)

CONFIG                DIRECT  LAUNDERED  REASON                RESOURCE WRITE  INDEP. VERIFIABLE  VERDICT
-----
baseline              allow   allow     shared-account-union  applied         no                 VULNERABLE
traditional_sandbox   allow   allow     forwarded-bearer-credential  applied         no                 VULNERABLE
scoped_delegation     deny   deny     authority-only-shrinks  blocked         no                 CONTAINED
agent_worlds          deny   deny     authority-only-shrinks  blocked         yes                CONTAINED
agent_worlds_ablation allow   allow     ablation-no-attenuation  applied         yes                VULNERABLE
```

B.2 Cryptographic Auditor Verification

Command: `python -m harness.audit --config agent_worlds`

```
=====
ADVERSARIALLY TESTED OFFLINE CONFORMANCE AUDITOR
=====

[1] Bundle verification : VALID (bundle ok: 4 records)
    Records read       : 4

[2] Declared claim      : containment_claim='contains'

[3] Laundered write hops: op='write:artifacts' caller=agent-A
                           receiver=agent-B delegated=True
    Matching attempts   : 1
    attempt[0]          : contained-the grant chain proves that
                           'write:artifacts' was attenuated out of the child

-----
Containment : CONFIRMED across all 1 attempt(s), and all 1 delegation
              record(s) conform.
AUDIT RESULT : PASS
```

LLM Usage Statement

LLMs assisted with brainstorming, code and debugging support, and identifying gaps or unsupported claims. I verified the reported results and sources. I wrote the final submission prose in my own words.