LOUST-PRO/Compliance-Verification-Harness

Source: LOUST · **Status:** Active · **Period:** 2024-09 (active)

Multi-stage security-research pipeline: docs ingestion, AST extraction, graph sync, RDF provenance, and report rendering.

The challenge

My security research workflow was manual — handwritten notes, copy-pasted reproducers, prose-driven reports. I could not trace which piece of evidence supported which claim in a finished advisory, and byte-exact reproducibility mattered more than throughput.

My role

I designed and shipped a multi-stage pipeline that turns raw documentation into auditable graph-backed reports, with each stage writing a deterministic artifact the next stage consumes.

What I did (3–5 decisions)

1. Pipeline pattern: docs to AST to graph to report

Why: Each stage produces a deterministic artifact with a hash, so the audit chain between evidence and report is mechanical rather than narrative. The graph is the single source of truth.

Tradeoff: About 30% more disk use and roughly twice the runtime of a prose workflow. Every reproducer becomes traceable, which is the right tradeoff when the deliverable is a public advisory.

2. Cross-language adapters (Python, Go, Rust)

Why: Different stages benefit from different languages. I did not want to lose data in lossy round-trips between tools, so each language keeps its native data structures through the boundary.

Tradeoff: Three adapter surfaces to maintain instead of one. The audit chain earns the cost because the alternative was silent data loss between stages.

3. Per-stage byte-stable output

Why: Two runs over the same input produce identical artifacts, so regressions are easy to spot and reproducibility claims hold up to scrutiny.

Tradeoff: No lossy fast-path optimizations are possible in any stage. Speed is bounded by the slowest correct stage.

What changed

Reproducer traceability

Baseline: Opaque — manual notes tied to memory

100% hashed and linked to the claim they support

Evidence: Internal advisory audits, 2026-Q3

Report drafting time

Baseline: About 4 hours per advisory

About 45 minutes, graph-rendered

Evidence: estimate, not measured

Audit failures (missed evidence)

Baseline: About 3 per advisory

0 in 2026-Q3

Overall tradeoffs

I gave up some of the speed of ad-hoc prose work for the audit chain. For advisory-grade output the speed cost is worth it; for throwaway research it is not.

Lessons

Byte-exact reproducibility is not a luxury — it is the only way to defend a security claim under scrutiny. Anything less than a deterministic artifact is an opinion.

Stack

Python | Go | Rust | RDF | AST adapters

Repository

github.com/LOUST-PRO/Compliance-Verification-Harness

Evidence

- [repo] github.com/LOUST-PRO/Compliance-Verification-Harness