

Governing Untrusted AI — the Agent Harness study

How to turn an untrusted, weak model into governed,
evidenced work with zero patches to the core — a reproducible
study with a full evidence trail you can re-run.

DECLARED
EVERY EFFECT

GOVERNED
AT THE POINT OF ACTION

EVIDENCED
SIGNED & REPLAYABLE

CONTENTS

01 **Executive summary**

02 **The methodology: evidence, not demos**

03 **Crash-proof effects and live authority**

Free sample — selected sections. The full report contains every section on the shelf.

Executive summary

Weak or untrusted models can still do real work — if the work is **governed by construction**, not by trust.

This report documents a reproducible qualification study with one question: can an off-the-shelf, untrusted agent harness — running a small, weak, locally-hosted model — be turned into governed, evidence-backed work **without patching the harness's core**? Across **31 independently-reviewed engineering tasks**, pinned to a single upstream release, the answer is yes.

Every high-consequence action was forced through a Capability Host Protocol (CHP) authority gate. Every real-world effect received a crash-proof idempotency key. Every outcome was written to a tamper-evident, hash-chained evidence ledger held **outside** the disposable agent cell. The decision — **adopt without core patches** — passed six formal veto gates and reproduced **94 of 95** decisive tests in a clean-room container.

Two reframes carry the result

- **The “weak model” problem is a context problem.** Dropped and duplicated steps were first blamed on model capability. Ground-truth benchmarking disproved it: with a *fresh* session, the same small 14B model passed **5/5** tool-use tasks. The lever was engineering — larger context window, native compaction, a leaner persona, sharper tool descriptions — not a bigger model.
- **Governance is an architecture problem, not a capability problem.** All governance was achieved through *public contracts only*, with **zero downstream core patches** — so absorbing the next upstream security release is essentially free.

The model proposes; the harness disposes. Authority never leaves the governed boundary.

The business case is the compounding-failure math. At 95% reliability per tool call, an eight-step task succeeds only about **66%** of the time. Idempotency, fail-closed approval, and verification from ground truth are what make probabilistic agents usable for consequential work — and what make the safety property portable across model swaps, harness upgrades, and cell destruction.

The methodology: evidence, not demos

The study’s core discipline is that nothing is confirmed because it worked once.

Evidence classes

Every claim is tagged **Confirmed** / **Strongly supported** / **Plausible inference** / **Speculative** / **Unknown**, each with a pinned commit, a source path or experiment ID, a date, an owner, and its counterevidence. Confirmation requires five things at once — code-path evidence, runtime evidence, failure-path evidence, compatibility against the pinned baseline, and **independent reproduction**.

A hypothesis may not be marked confirmed solely because a prototype appears to work.

A pre-committed hypothesis register

Sixteen hypotheses (H-001...H-016) each carry an initial status, a named falsification test, and a decision impact. Experiments record their pass/fail thresholds, fixtures, model configuration, expected failure modes, and evaluator **before** running — so results cannot redefine their own success criteria.

Six veto gates as property tests

Governance is enforced by six invariants (INV-001...INV-010), each a randomized property test naming the single component that enforces it:

INVARIANT	WHAT IT GUARANTEES
all_consequential_paths_mediated	No A3/A4 effect bypasses CHP
canonical_state_external	Canonical state lives outside the cell
no_cross_tenant_credentials	A cell cannot reach another tenant
no_duplicate_effects	One idempotency key → at most one effect
exact_action_approval	The approved action is the executed action
detectable_evidence_tampering	Any ledger edit is detectable

The idempotency key is defined precisely as `sha256(capability_id + canonical(arguments))` — deliberately excluding invocation and correlation IDs, so a retry can never double-apply. A failed gate

requires architectural revision and rerun; it cannot be overridden into a pass.

Hard kill criteria

The study pre-declared the conditions under which the answer is “no”: if a consequential action can bypass CHP; if canonical state needs unsupported private-database access; if a compromised cell can reach cross-tenant credentials; if no clean zero-patch path exists; if security updates need pervasive manual reconstruction; or if the harness ontology prevents clean separation of conversation, task, run, effect, and evidence.

Crash-proof effects and live authority

Two experiments carry the whole thesis: effects survive a crash exactly once, and an external authority can veto a live action on an unmodified harness.

Exactly-once under a real crash (EXP-003)

The test does not simulate a failure — it hard-kills the process with `os._exit(137)` after the effect is durably written but before the completion acknowledgement, then recovers from fresh handles across three distinct stores.

```
EXP-003  effect idempotency under crash
journal_row_count_after_crash_only ..... 1
journal_row_count_after_recovery ..... 1
effect_actor_side_effect_count ..... 1
naive_non_idempotent_retry_count ..... 2
recovered_status ..... recovered
2 passed in 0.33s
```

The guarded path applies the effect **once** and records the recovered run as `recovered`, not `succeeded`. The retained un-guarded control applies the same effect **twice** — proving the gate catches the violation rather than passing vacuously.

A live external veto, zero core patches (EXP-014)

CHP acts as the live external approver on an **unmodified** gateway. A requester connection blocks on an approval request; a CHP connection holding `operator.admin` receives it, decides, and resolves it. The result differentiates by policy:

```
exec.approval  live decision (unmodified gateway)
printf chp-live-ok ..... allow-once
rm -rf /tmp/chp-x ..... deny  (fail-closed)
downstream core patches ... 0
```

The bypass we found — and closed (EXP-018)

An adversarial probe found a real hole: the agent used the harness's native `write` tool to create a file, and CHP saw nothing. It was **closed by construction** — a closed-by-default tool allowlist (`exec`, `read`, `ls`, `find`, `grep`) so the only effect path is CHP-gated, plus a boot-time mediation check that fails loud on drift.

Governed file capabilities were then re-added *through the mediated bridge*, every call landing `execution_started` → `fs_write` → `execution_completed` in the evidence ledger.

One idempotency key cannot create two successful equivalent effects.

Figures reproduced from evidence/experiment-results/EXP-003 and EXP-014-ROOT-CAUSE. Both re-run from a clean checkout via the commands in the reproducibility appendix.

The rest is behind the paywall.

This free sample gives away the thesis, the full methodology, and one complete results chapter — real crash, real deny, real numbers. The gate-by-gate results, the weak-model production playbook, the comparative economics, and the full reproducibility appendix are in the complete report.

Read & buy → agency.capabilityhostprotocol.com/thinking/governing-untrusted-ai