PROTEAN: BINDING CODE, AUTHORITY, AND RECOVERY ACROSS UPDATES IN AGENT RUNTIMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-running LLM agents execute through tools whose implementations and permissions may change while calls are in flight. Yet existing runtimes typically treat code updates, authorization, and crash recovery as separate concerns. This separation creates invocation-level races: a call may be checked against one revision but dispatched to another, rollback may revive revoked authority, and retry after a crash may duplicate an external effect. To eliminate these races, PROTEAN maintains code and authority as independent version histories, yet binds each admitted invocation to one immutable code revision, authority epoch, and recovery identity. This prevents code rollback from resurrecting revoked authority while ensuring that admission, dispatch, and recovery refer to the same invocation. PROTEAN realizes these invariants through a trusted Kernel that (i) admits heterogeneous operations through a fail-closed capability contract, (ii) enforces epoch-scoped authority at every invocation, and (iii) pins execution and journals effects for safe replacement, rollback, and recovery. On AgentDojo, mixed HTTP/MCP/Python execution matches native-tool utility (95.3% vs. 93.7%). Across 1,146 out-of-grant requests, none produces an out-of-envelope effect at the Kernel-mediated boundary. Across 60,000-request replacement and rollback workloads and adversarial interleavings, PROTEAN loses no requests and exhibits none of six constructed lifecycle violations.

1 INTRODUCTION

Long-running agents increasingly execute in mutable tool environments: implementations are replaced, permissions are revoked, and calls fail while external effects remain uncertain. This setting spans MCP servers (Model Context Protocol contributors, 2025), generated skills (Cai et al., 2024; Wang et al., 2024c; Qian et al., 2023; Wölflein et al., 2025; Wang et al., 2024a;b), and stateful APIs (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024a; Yao et al., 2023; Debenedetti et al., 2024). A stochastic planner may correctly diagnose an immediate obstacle yet select an action outside the authority granted for its task, so we treat the planner as an untrusted principal and place the guarantees in the runtime—but even a correct policy does not settle which code, authority, and recovery semantics govern a call whose environment is changing beneath it.

Existing safeguards address these concerns separately. Agent defenses constrain model–tool interaction, untrusted inputs, or per-tool privileges against a fixed tool set (Greshake et al., 2023; Liu et al., 2023; Debenedetti et al., 2026; Shi et al., 2025); dynamic updating preserves code-version consistency (Hicks & Nettles, 2005; Neamtiu et al., 2008); and transactional recovery attributes interrupted effects (Gray, 1981; Mohan et al., 1992). In an agent runtime, however, all three states meet at one invocation. Each state can invalidate another: dispatch to v_2 voids a check made against v_1, a rollback can revive a revoked grant, and a retry can repeat an effect that the interrupted attempt already caused. The core question is thus *which implementation, authority, and recovery identity (the name its effects are journaled under) govern an invocation as its environment changes*.

PROTEAN answers this question by binding the three choices in an immutable *invocation snapshot*. This creates three challenges: OpenAPI, MCP, skill, and code-backed sources expose incompatible contracts that admission must not weaken; an untrusted planner's authority must be delegated per capability and per invocation; and replacement, rollback, revocation, and recovery interleave with admitted calls, risking version tearing, stale authority, or blind replay.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

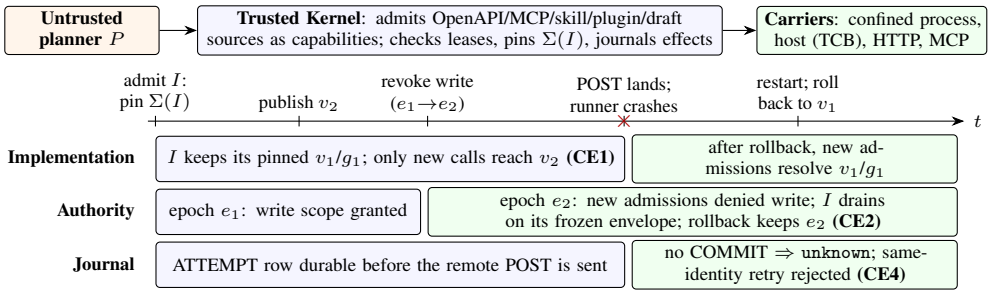


Figure 1: Pinning $\Sigma(I)$ at admission confines v_2, revocation, and rollback to future admissions and turns the crash after the remote POST into `unknown` rather than a duplicate ticket. Top: the untrusted planner reaches carriers only through the trusted Kernel. Bottom: invocation I across three state lanes; CE labels match Table 17.

We present PROTEAN, a capability-oriented runtime mediated by a trusted *Kernel*, a user-space reference monitor rather than an OS kernel. Its central design choice is that authority needs a version history separate from code, so that a code rollback cannot undo a revocation, but must share the invocation identity that pins the code and names its effects, so that the check and the dispatch concern the same call. A fail-closed funnel maps each operation whose contract it can express to a canonical capability and rejects the rest; epoch-scoped leases and explicit delegation edges enforce whatever policy is granted, independently of how precise that policy is. On admission, the Kernel atomically binds an immutable revision, current authority, and durable identity, then records effects against that snapshot in a write-ahead journal. Replacement changes future bindings, revocation advances authority independently of code history, and recovery marks ambiguous non-idempotent effects `unknown` instead of replaying them blindly. Figure 1 traces one invocation through all three hazards; undeclared transitive dependencies and remote server-internal effects lie outside the snapshot.

Contributions. This paper makes three contributions. **(1) Canonical capability admission** (§2.2, §3.1): a fail-closed representation and admission pipeline that maps supported OpenAPI, MCP, Python, and skill/plugin operations into one governed execution plane without silently weakening their declared contracts. **(2) Invocation-bound authority** (§2.3, §3.2): a capability and lease model in which authority evolves independently of code revisions but is frozen per invocation, preventing stale or resurrected privilege across replacement and rollback. **(3) Governed lifecycle semantics** (§2.4, §3.3): a joint model of revision pinning, monotonic authority epochs, and durable effect identity that defines safe replacement, rollback, revocation, and crash recovery. We evaluate these claims through differential execution, planner-in-the-loop authority tests, adversarial lifecycle interleavings, crash replay, and remote MCP experiments; consistent with an untrusted planner, the checks span five planners whose task success ranges from 9% to 95%.

2 DESIGN

PROTEAN is organized as a trusted Kernel mediating a set of governed capabilities. A *capability* is a reusable, persistently named operation: admitted once, invoked by many callers and episodes, and revised as its API spec, MCP server, skill or plugin, or generated code changes. One-shot commands are not admitted individually; they run through the bundled `exec` capability. The running example below introduces the objects that the following subsections define.

2.1 RUNTIME BOUNDARY AND CAPABILITY LAYERS

Running Example. Consider one invocation I throughout this section (Figure 1): an agent calls a tracker capability `issue.create`(*title*) admitted from an OpenAPI operation. At admission, the Kernel freezes a snapshot $\Sigma(I)$ that pins the code revision and the lease envelope. During execution, the operator publishes revision v_2 of the capability and revokes the legacy write scope, and the runner then crashes after the remote POST is received but before the local commit row is written. Without a shared invocation identity, three failures follow: dispatch may reach v_2 under authority vetted for v_1

(check–dispatch drift, which we call version tearing), a later rollback may restore v_1 together with its revoked grant (privilege resurrection), and restart may blindly retry a ticket creation that already succeeded remotely (duplicate effect). The following subsections trace how admission produces the snapshot (§2.2), how the lease bounds its authority (§2.3), and how lifecycle transitions and journaling resolve each hazard (§2.4); the counterexample ladder (Appendix C.3) measures these failures as CE1, CE2, and CE4. Under the default *drain* semantics, I still completes its POST on its frozen envelope; the revocation denies every later admission. The example exercises four layers of a governed capability: the immutable *definition/revision* (contracts, dependencies, provenance) is what v_2 replaces; the *runtime binding*, or *carrier* (confined subprocess, trusted host executor, HTTP endpoint, or MCP connection), is what the POST traverses; the *authority state* (grants, leases, delegation edges, revocation epochs) is what revocation advances; and the frozen *invocation snapshot* is what $\Sigma(I)$ records.

The Trusted Runtime Kernel. The Kernel arbitrates identity, authority, lifecycle transitions, and invocation admission. Stable capability identifiers name objects with immutable revisions; admission resolves the active revision. Epoch-scoped leases grant sets of permissions, each a (resource, operation) pair such as `net.request` on the tracker host or `fs.write` under a path; such pairs cover API scopes as well as filesystem, network, and process authority. The Kernel freezes the resolved binding and authority into an invocation snapshot and tracks mediated effects in a synchronously persisted transaction journal.

2.2 Admitting Heterogeneous Operations as Governed Capabilities

PROTEAN transforms heterogeneous operations into governed capabilities; semantic preservation is tested differentially (§3.1), not proved.

Unified Capability Plane Across Ecosystems. Admission extracts a canonical manifest, validates its contract, and creates an execution binding. In the running example, the manifest for `issue.create` names its *inputs* (`title`), *output* (an issue record), *effects* (one HTTP POST to the tracker endpoint), *carrier* (the HTTP egress carrier), and *provenance* (the OpenAPI operation id and spec revision). OpenAPI operations carry parameter schemas, HTTP encodings, and local egress requirements. MCP bindings isolate protocol and transport state; Python bindings expose declared contracts. Skills and plugins enter through a narrow native adapter (§3.1): each executable entrypoint compiles to a capability IR, while instruction prose such as `SKILL.md` is kept as provenance and grants no authority, so injected instructions cannot widen a lease.

Code-Backed Operations and Staging. Declarative operations (HTTP, MCP) arrive with wire-level contracts the Kernel can inspect directly. Code-backed artifacts instead carry executable logic whose declared effects cannot be verified from the schema alone, so they pass through draft, sandboxed probe, candidate, and admission stages. Drafts are isolated from discovery and grants; a single-use staging lease bounds probes, and admission requires an explicit reuse confirmation, so a failed or discarded draft leaves no identity, discovery entry, or authority (12/12 conformance cases; Appendix A.2). Probing checks contract fidelity; it cannot prove that declared effects are complete, which is the job of confinement.

Execution Substrate Confinement. Representation is insufficient unless the resulting binding preserves the declared effect boundary at execution time. Confined process and microVM carriers apply filesystem, network, and resource restrictions (Watson et al., 2010; Hunt & Larus, 2007; Agache et al., 2020), extending the application-level resource-management substrate pioneered by exokernels (Engler et al., 1995). Host builtin and interpreter carriers execute inside the trusted process and report effects through the effect API; their code is part of the trusted computing base (TCB), so untrusted or synthesized code should run on confined carriers. Generated skill CLI bindings thus run as no-network confined processes exchanging files only via contract payloads.

2.3 Delegating Authority to Capabilities

The Kernel retains root meta-authority and delegates bounded authority to capabilities and invocations.

Kernel Meta-Authority and Leases. Following capability-based protection (Levy, 1984; Miller et al., 2003), the Kernel issues scoped leases bound server-side to a principal, so a leaked lease identifier confers nothing on another caller. An invocation I admitted at time t receives the permission set $A(I) = \text{ResolveAuthority}(P, C, \text{Epoch}_t)$, where P is the caller, C the capability, and ResolveAuthority the permissions that C's declared resolution rules derive from the call's arguments (bounded by C's manifest envelope). Admission fails unless one of P's live, unexpired leases at Epoch_t covers all of $A(I)$; leases are issued only within $\text{Ceiling}(P)$, the maximum scope P may hold, or by explicit approval, so no permission outside the ceiling is granted silently. $\text{RealizedEffects}(I)$ covers mechanically mediated local effects and trusted-host effects reported through the effect API (Assumption A1). For HTTP/MCP, mediation covers local egress and credentials; server-internal effects remain outside this boundary. Out-of-bounds requests are denied at the mediation boundary enforced by interposition and reference-monitor techniques (Hunt & Brubacher, 1999; Krohn et al., 2007; Zeldovich et al., 2006).

Edge-Based Composition Graphs. To prevent ambient privilege propagation across nested tools, PROTEAN enforces explicit edge-based delegation. A parent capability C_1 declares an invocation edge targeting child C_2. The parent's envelope gains only an invoke handle for that edge, never C_2's effect authority. Nested requests re-enter the Kernel, which checks the child's live revision under a freshly read epoch and admits the child only if the parent's frozen envelope $A(I_1)$ contains the handle for that edge and the originating caller's admission-time lease covers the child's own resolved authority $A(I_2)$. A parent thus cannot lend its effect authority to a child, and nesting never exceeds what the originating caller was granted (Hardy, 1988).

Assumption A1 (Mediation Boundary). *All effects of an invocation pass through a Kernel-mediated channel, on a carrier-specific basis: (i) host-registered callables and interpreters are in the TCB and trusted to report effects through the effect-ledger API; (ii) subprocess carriers confine effects by OS-level isolation; (iii) remote HTTP/MCP endpoints are mediated at local egress only; server-internal mutations lie outside RealizedEffects*(I). Clauses differ in *what* guarantees them (trust, OS mechanism, or channel routing); Appendix G formalizes and §3.2 measures them.

Least-Privilege Enforcement. Under Assumption A1, for every enforceable effect at the mediated boundary, $\text{RealizedEffects}(I) \subseteq A(I)$. For a static tool set this containment is close to the definition of a reference monitor; what is non-trivial is that $A(I)$ remains the envelope of the correct epoch while revisions, grants, and bindings change concurrently (§2.4). Following classical protection models (Saltzer & Schroeder, 1975; Lampson, 1974; Sandhu et al., 1996), PROTEAN separates *enforcement soundness* from *policy precision* (tightness of $A(I)$). Requests exceeding principal ceilings enter pending approval: authorized approvers (a deterministic rule, a human, or an LLM judge) can issue scoped one-shot leases or update ceilings under subsequent lease checks.

2.4 GOVERNING CAPABILITY CHANGES

Registration, replacement, revocation, and rollback share explicit authorization, invocation-binding, and recovery semantics, motivated by version-consistent updating (Hicks & Nettles, 2005; Neamtiu et al., 2008).

Invocation Snapshot Pinning. To guarantee execution continuity under concurrent environment changes, the Kernel atomically binds an immutable snapshot upon admission: $\Sigma(I) = \langle \text{CapID}, \text{Revision}, \text{Generation}, \text{Epoch}_t, A(I) \rangle$. Here, Revision numbers the version lineage while Generation identifies the executable provider binding that revision resolves to, which is the object an invocation actually pins and holds references on (rollback restores the target revision's original generation); both are decoupled from the authority epoch Epoch_t. Grants attach to the capability identity and principal, not to a revision. Because rollback re-points implementation bindings to historical revisions while the authority epoch only increases, historical code is admitted only under authority valid at its new admission, which precludes the resurrection of revoked grants. Admitted invocations execute exclusively against their pinned snapshots, while new admissions resolve to updated bindings.

Table 1: Admission is selective and fail-closed. L1 is parse-clean input; L2 is the represented operation, tool, or artifact. L3 is a runtime probe whose unit differs by corpus and is named in each cell, so L3 denominators are not L2 counts. Full rejection taxonomies appear in Appendix B.

Corpus	Input	L1	L2 (rate)	L3 observation
OpenAPI operations	12,948	11,171	4,206 (32.5%)	3,344 wire matches; 1,308 successful synthetic invocations
MCP tools	306	158	158 (51.6%)	50/142 live server handshakes succeed
AgentDojo tools	69	60	49 (71.0%)	69/74 tool-version probes pass
Skills/plugins	644	644	118 (18.3%)	104/401 member-IR calls succeed on generic inputs

Atomic Version and Authority Transitions. The Kernel coordinates replacement, semantic revision, revocation, and rollback atomically via Compare-And-Swap (CAS) over capability metadata. Each transition touches a distinct component of the binding: implementation changes (replacement, semantic revision) mint a new revision and generation; revocation advances `authority_epoch` and denies new admissions while admitted calls drain on frozen envelopes; lease expiry and use-exhaustion instead invalidate only that lease at admission, without advancing the epoch; and rollback re-points the code binding to a historical revision while the authority epoch stays monotonic. No transition silently changes both implementation identity and authority state. Revocation supports *drain* (default: deny new admissions, let in-flight calls complete) and *kill* (SIGKILL; a builtin host callable instead completes and its outcome is denied at commit); an effect-boundary variant is evaluated in Appendix C.14.

Durable Effects and Crash Recovery. Before emitting any side effect, the Kernel synchronously logs an attempt record into a write-ahead journal (WAL); upon confirmation, a commit record seals the transaction (Gray, 1981; Mohan et al., 1992; Garcia-Molina & Salem, 1987). The available durable evidence determines recovery (Table 15): idempotent operations safely re-execute and confirmed external receipts resolve outcomes, whereas ambiguous non-idempotent interruptions become `unknown`, which permanently bars blind duplicate execution. In the running example, the runner dies after the tracker receives the POST but before the local commit record lands; durable evidence therefore classifies the invocation `unknown` and rejects same-identity resubmission rather than blindly recreating the issue. The caller receives `unknown` as a typed outcome; any retry is a fresh, separately journaled job, ideally issued after the caller reconciles (e.g., by searching the tracker). Nine specification-level invariants (I1–I9) are argued by proof sketch, not machine-checked, in Appendix G; implementation conformance is tested in §3, and I2 (authority confinement) is only as strong as Assumption A1.

3 Experimental Evaluation

We organize the evaluation around the three claims rather than individual harnesses: representation and utility (C1), authority enforcement (C2), and lifecycle safety (C3). All lifecycle changes are issued by the harness; model-initiated authoring is outside scope. In the model-in-the-loop runs, AgentDojo's own Python tools execute on the trusted host carrier: there the adversary is the planner, not tool code. Hostile tool code is evaluated separately on each carrier (Table 3).

3.1 Can Heterogeneous Operations Share One Contract Plane?

Admission is deliberately selective: inputs pass through parsing (L1), fail-closed representation (L2), and runtime probing (L3), and unsupported schemas, transports, or entrypoints are rejected rather than approximated. Table 1 therefore reports the boundary of the implemented grammar, not universal coverage. Rejections concentrate at the domain edge rather than in ordinary operation semantics: 6,872 of the 8,742 OpenAPI rejects (79%) are deliberate refusals of unemittable fragment targets or non-JSON media; admission within the parse-clean JSON domain is 53.3%, and every emitted probe preserved its wire shape.

Table 2: Planner independence (one seeded run per model, never pooled). *Utility*: valid pairs of 255 planned, mixed/native (P/N) task success, paired difference, and one-sided 95% lower bound from the preregistered suite-stratified cluster bootstrap (the independence-assuming paired binomial bounds, e.g., -2.7 pp for DeepSeek, are in Appendix C.10). *Scripted escalation*: grants by the planner as LLM approver and realized effects, of 82 goal-aligned requests. *Model escalation*: unapproved lease issuance (SI) and realized out-of-envelope effects (ROE). Grants reflect approver policy, not Kernel mediation.

	Utility (AgentDojo, NI)					**Scripted esc.**		**Model esc.**
Planner	valid	P / N	$\hat{\Delta}$	LCB95	NI	grant	realized	SI/ROE
DeepSeek-V4.1-Flash	253	.953/.937	$+1.6$pp	-0.8pp	✓	0	0	0/0
GLM-4-32B[a]	255	.314/.306	$+0.8$pp	-0.4pp	✓	22	22	0/0
Qwen3.8-27B[c]	249	.940/.936	$+0.4$pp	-0.8pp	✓	0	0	0/0
Nemotron-3-Nano-4B[b]	252	.698/.722	-2.4pp	-6.0pp	✓	3	3	0/0
Olmo-3-7B	255	.090/.110	-2.0pp	-4.3pp	✓	0	0	0/0

[a] Canonical GLM run uses the tool-result-fixed chat template. [b] One goal-aligned Nemotron effect was realized within an already-held envelope; it is policy-attributed and ROE $= 0$. Low-utility arms (Olmo, GLM) limit planner activity. [c] Model-escalation run stopped at its request cap (81 episodes).

Within this admitted domain we measure *silent weakening*: a governed call that returns success while its result or post-state differs from native execution. Across 566 successful governed calls over Python, MCP, and HTTP carriers in the 20-tool differential there were none under our canonical comparison predicate, after excluding 15 per carrier traced to a nondeterministic native reference. Across the 49 representable AgentDojo tools, result and state agreement were 0.978 and 0.987 (Table 25); whether an error occurs agrees in 0.990 of cells, but error class matches in only 0.447 because carriers name the same failure differently; the residuals are typed errors, transport encodings, or native timestamps, not silent successes. For task utility we use the 85 of 97 AgentDojo user tasks whose ground-truth tools are all adapter-supported (three seeds each; 255 planned pairs, 253 valid), each episode deterministically splitting the suite's tools across Python (host), MCP, and HTTP (loopback) carriers. Mixed execution is non-inferior to native at the preregistered -10 pp engineering margin (an engineering tolerance, not a task-cost-derived bound): 95.3% versus 93.7% task success, $+1.58$ pp paired difference, one-sided 95% lower bound -0.8 pp (suite-stratified cluster bootstrap), and every tighter margin down to -2 pp, where the verdict depends on the dependence model (Table 9); an earlier 40-pair pilot did not clear -10 pp (-16.9 pp bound), which motivated the larger protocol. The arms are equivalent within ± 5 pp (TOST). Four additional open-weight planners clear the margin in single seeded runs (Table 2); the two low-utility planners (Olmo, GLM) are weakly informative on utility and serve only as security replications.

3.2 Do Leases Enforce the Boundary, or Do Models Merely Behave?

Out-of-envelope requests are common under pressure: when the ceiling deliberately omits the tool a task needs, planners request escalation in 33/40 and 32/40 episodes across two seeds, and the 240 escalation episodes contain 1,146 out-of-ceiling requests. Defense-off ablations and a scripted positive control separate mechanism from model behavior: disabling each check exposes the predicted failure in all eleven attack families, and all 82 scripted requests execute when granted but none under denial, so zero violations reflect active mediation, not a reluctant planner (Table 3); with zero events in 240 episodes, the one-sided 95% upper bound on the violation rate is 1.25%.

The last two rows justify the carrier-specific scoping of Assumption A1 (per-class cells in Table 19): process isolation confines undeclared effects, while an in-process host callable cannot be made safe by an API wrapper alone and can even write the Kernel's own state store; host code is therefore in the TCB (full matrices in Appendices C.6–C.12).

Policy quality remains orthogonal. Enforcement soundness holds within A1 regardless of how tight the granted policy is. An LLM synthesizer that maps a task description to an envelope still covers an injection goal's permissions in 45/321 task–goal pairs, versus 340/340 under a broad policy, while task success moves from 29/34 to 26/34 with an interval crossing zero. Injection exposure is sparse: the 250 separate injection episodes contain only 40 goal-aligned out-of-ceiling requests,

Table 3: Enabled defenses remove the failures exposed by their controls. SI: lease issued without approval; ROE: realized out-of-envelope effect. The trusted-host row is a measured boundary: host executors are inside the TCB, whereas process carriers are mechanically confined.

Evidence	Unit	Control	Observed result
Defense ablation	11 families, ≥ 20 variants each	Intended Kernel check disabled	All 11 failure signatures appear off; 0 escapes with checks on
Scripted escalation	82 requests	Deny vs. executable grant	0 vs. 82 realized
Model escalation	240 episodes	Deny, oracle, and LLM approvers	SI $= 0$; ROE $= 0$ over 1,146 out-of-ceiling (pending) requests
Sandboxed process	70 bypass variants	External host observer	70/70 mechanically intercepted
Trusted host executor	70 bypass variants	Same observer	7/70 intercepted; 63/70 bypass the effect API

Table 4: Lifecycle evidence from two protocols. Continuity columns use the 60,000-request workload (Gap: arrival-to-completion transition window); the safety column summarizes the separate barrier suite (event counts summed over scenarios at 50 trials each, not rates); the S2 row is a separate report condition, not pooled with S1. Baselines are faster; only PROTEAN prevents the safety violations.

Condition	Lost (of 60k)	Gap (ms)	Barrier-trial safety observations
Restart	125	180.2	Not in barrier suite (downtime loss)
Naive	0	30.5	50 torn, 150 post-revocation, 400 unattributed, 50 crash-unclassified
Erlang-2Ver	0	30.5	400 unattributed, 50 purge-loss, 50 crash-unclassified
MCP-advisory (S1)	0	34.5	400 unattributed, 50 crash-unclassified, 150 stale-listing
S2: advisory disable	—	—	250 post-revocation
PROTEAN replace	0	51.7	0 events in all six safety categories
PROTEAN rollback	0	44.2	0 events in all six safety categories

none of which received a grant or realized an attack; the one goal-aligned effect we observed used an already-held grant (Table 2, note b), a policy over-grant rather than a mediation failure. This positions PROTEAN relative to prompt-injection defenses such as CaMeL and Progent (Debenedetti et al., 2026; Shi et al., 2025): they decide *which* actions or flows a policy should permit, and an injection that stays inside the granted envelope is outside our guarantee, so attack success rate is not our primary metric. PROTEAN instead guarantees that whatever policy they produce is the one enforced on each invocation while tools, grants, and bindings change; the two are composable.

3.3 CAN CAPABILITIES CHANGE WITHOUT STOPPING THE TASK?

Both replacement and rollback keep tasks running; what distinguishes PROTEAN is that it does so without lifecycle safety violations. We compare against three author-implemented semantic baselines: NAIVE swaps the implementation in place; ERLANG-2VER follows OTP hot code loading, pinning each call at entry to one of two coexisting versions and killing old-version calls when a third loads; MCP-ADVISORY follows official-SDK live-update semantics, where revocation only hides a tool from listings; RESTART stops and restarts the runtime. These are harness-level semantic reimplementations, not executions of an external Erlang runtime or a production MCP deployment. The continuity and safety suites test different properties and are not pooled. In the 60,000-request continuity workload at 29.85 requests/s, replace and rollback lose no requests; restarting loses 125. Continuity alone does not distinguish the design: the hot-update baselines also lose zero and have shorter arrival-to-completion transition gaps (30.5–34.5 ms vs. 44.2–51.7 ms). They pay for it in the safety suite, which places a fixed barrier at the vulnerable interleaving of each scenario, 50 trials per scenario. Its counts show whether a design has a mechanism for each hazard, not how often the hazard occurs naturally. Only PROTEAN records zero events in all six categories: version tearing, post-revocation admission, unattributed effects, purge loss, unclassified crashes, and stale listings (Table 4; full matrices in Tables 13 and 14).

Table 5: C3-R against MCP servers on another continent, judged from server-side records. Declared drift: post-event calls under description, schema, and removal changes (n=20 episodes $\times$ 3 calls each); behavior-only: issues silently made public. Lost response: official `gitea-mcp` with a create deadline below the round trip. Full tables in Appendix D.

Event	SDK host (cached list)	PROTEAN, pinned
Declared drift	180/180 calls still sent; description change followed silently (60/60)	0/180 sent; refused before `tools/call`
Behavior-only drift	60/60 issues public	60/60 issues public (undetectable by design)
Lost response	retry: 19/20 episodes duplicated (39 issues), all reported timeout	retry refused 20/20; search resolves 20/20 (13 landed, 7 not)
Create p50 (`gitea-mcp`)	1.5 s	3.8 s (re-lists 55 tools per call)

A counterexample ladder that adds one primitive per rung, under 50 randomized timing injections per cell, explains why: pinning alone closes the check–dispatch race but not rollback reviving authority or a crash duplicating an effect, durability alone cannot repair a decoupled check and dispatch, and only joint admission prevents all four (CE1, CE2, CE4, CE5); parent pinning does not freeze an undeclared child dependency (Table 17, Appendix C.3).

Robustness beyond the primary workload. The stress tests agree with this result. In a 330-cell bounded-queue matrix, the PROTEAN replace and rollback arms keep at least 99.6% of sub-capacity requests correct within one second (the revocation cells retain less by design: in-flight requests are legitimately rejected, not lost); above capacity its synchronous journal saturates like the no-change arm while journal-free Naive does not, and the overload surfaces as backpressure rather than silent loss (Table 24). All 16 physical crash-cut cases classify recovery correctly, real HTTP and filesystem crash traces reconcile effect records without blind same-identity replay, and pinning and revocation hold across a mixed skill–builtin–MCP task chain (Appendix C.4).

Against servers we do not control. The results above use carriers and servers the harness controls. C3-R repeats the lifecycle events against two MCP servers on another continent, one of them the official `gitea-mcp` in front of a real Gitea, and judges every outcome from server-side records (Table 5). The results cut both ways. A definition pin, added for this study, hashes the tool's `tools/list` entry at install and re-checks it before each call; it refuses every declared change before any request leaves the client, including the description change that an SDK host follows silently. The pin guards the declared definition only: between the re-check and the `tools/call` the server can still change, and a server that changes behavior behind an unchanged definition passes every arm, the pin included. After a lost response, an SDK host with an enabled retry-on-timeout policy duplicates the issue while reporting a timeout, whereas PROTEAN refuses the same-identity retry and a pinned search resolves each unknown outcome in both directions. The pin costs one re-list of all 55 tools per call, raising create p50 from 1.5 to 3.8 s over this link.

4 DISCUSSION AND LIMITATIONS

Lifecycle Coupling and Cost. The general lesson extends beyond PROTEAN: version pinning and durable recovery are insufficient when authority has an independent lifetime—a runtime that revokes permissions without binding them to the identity that pins code and names effects admits version tearing, privilege resurrection, or duplicate execution (Table 17). The coupling costs 23.0 ms p50 on the confined process carrier—small next to a planner's seconds per step—with journal `fsync` at 8.3 ms of the 12.9 ms carrier-free p50 (Table 22); relaxing SQLite to `NORMAL` halves the durability-run p50 but can lose the latest ATTEMPT row on power loss, so `FULL` stays the default (Appendix C.9).

Limitations and Trust Boundaries. The Kernel enforces safety under explicit operational boundaries: (1) *Single host*: Targets single nodes; distributed consensus is future work. (2) *Policy precision*: Enforces granted envelopes, but cannot prevent over-privileged policies from synthesizers or le-

nient approvers (Table 2). (3) *Host bypass*: Host executors bypass mediation in 63/70 variants (Table 19), leaving 1,776 LOC of host and interpreter executor code inside the trusted computing base. (4) *Egress visibility*: Mediates local egress, but networked process carriers share the host network all-or-nothing, and remote server-internal mutations, including an MCP server changing behavior behind an unchanged tool definition (Appendix D), remain unobservable. (5) *Revocation limits*: drain mode completes in-flight calls, kill mode cannot recall remote mutations, and ambiguous crashes become `unknown`. (6) *Evaluation scope*: Lifecycle changes are harness-issued, the draft path for generated code is conformance-tested only, and end-to-end utility is preregistered on AgentDojo only (descriptive τ^2-bench replication in Appendix E; per-call fidelity is benchmark-independent).

5 Related Work

Agent execution control and capability security. Classical capability and reference-monitor systems provide explicit authority, confinement, and mediated effects (Levy, 1984; Miller et al., 2003; Shapiro et al., 1999; Watson et al., 2010; Zeldovich et al., 2006; Krohn et al., 2007; Hardy, 1988). Recent agent systems enforce such boundaries at execution time: IsolateGPT and ACE isolate LLM-integrated apps and enforce capability barriers between planning and execution (Wu et al., 2025; Li et al., 2026); Conseca derives just-in-time, task-specific policies from trusted context (Tsai & Bagdasarian, 2025); AgentSpec enforces customizable runtime rules over agent actions (Wang et al., 2026); Progent enforces revisable, argument-level privilege policies on tool calls (Shi et al., 2025); CaMeL separates trusted control flow from untrusted data flow (Debenedetti et al., 2026); tracked capabilities type an agent's authority with information-flow-style constraints (Odersky et al., 2026); HCP brokers per-invocation grant matching for MCP calls (Liu, 2026); Agent libOS admits operations through process-scoped capabilities and task ceilings (Zhang, 2026); and GoEX explores post-facto validation and undo (Patil et al., 2024b). All of these decide what an agent may do or where its data may flow at the moment of the call; in the mechanisms we examined—section-pinned for the three closest systems in Table 29—none specifies and evaluates whether the binding admitted for that call remains valid while the runtime changes underneath it. PROTEAN occupies this complementary cell: once an invocation is admitted, its implementation revision, authority epoch, and recovery identity must remain jointly valid despite replacement, revocation, rollback, and failure.

Transactional agent runtimes and lifecycle consistency. Cordon wraps multi-step agent execution in task-level semantic transactions that stage external effects, track result lineage, and carry delegated authority and recovery metadata at the task boundary (Chen et al., 2026). Atomix gives transactional tool-use semantics per resource: calls carry epochs, a progress frontier and predicates decide when effects may commit, bufferable effects are delayed, and externalized reversible effects are compensated (Mohammadi et al., 2026); SagaLLM brings saga-style compensation and independent validation to multi-agent planning (Chang & Geng, 2025). These systems coordinate *when* workflow effects may settle; PROTEAN coordinates *which* implementation and authority govern each admitted invocation while capability definitions and grants change concurrently; its journal resolves crash ambiguity, not workflow-level atomicity. Classical dynamic software updating (Hicks & Nettles, 2005; Neamtiu et al., 2008; Arnold & Kaashoek, 2009; Hayden et al., 2012; Giuffrida et al., 2013; Armstrong, 2007) and transactional recovery (Gray, 1981; Garcia-Molina & Salem, 1987; Mohan et al., 1992; Denning, 1976) decide which code version executes and how interrupted effects are attributed, but do not specify, in the mechanisms we examined, whether the authority checked before an update still governs the dispatched or recovered invocation; PROTEAN commits all three at admission. Broader tool-use ecosystems and agent benchmarks are summarized in Appendix F.

6 Conclusion

We have presented PROTEAN, which treats each admitted invocation as a joint commitment to an implementation, an authority envelope, and durable effect evidence. Evaluation across heterogeneous tool surfaces, adversarial authority tests, live updates, and crash recovery shows that this coupling preserves utility while preventing the lifecycle failures caused by independently managed code, authority, and effects. PROTEAN does not make an over-privileged policy correct, observe remote server-internal mutations, or provide distributed consensus. Within a single-host mediated runtime, however, continuous execution need not sacrifice control over which code executes, which authority it exercises, or how its effects are recovered.

REPRODUCIBILITY STATEMENT

The appendix specifies experimental denominators, measured working points, and mediation assumptions. An anonymized artifact accompanies the submission, organizing harnesses, frozen configurations, archived reports, and analysis code by claim. All deterministic suites—the representation-funnel recomputes, authority and sandbox probes, the defense-off threat matrix, bounded-queue continuity, the safety matrix, and crash-replay soaks—run without model API access; only the model-task evaluations (C1-X Phase B and its G5 multi-planner replications, the C1-T τ^2-bench replication, the C2-ESC and C2-PC LLM arms, and the C3-V caller pilot) require a model endpoint. Native-adapter manifests retain artifact membership and contract provenance alongside execution traces.

ETHICS STATEMENT

Adversarial authority and sandbox tests use synthetic fixtures and mocked credentials. MCP interoperability probes contact public servers; task evaluations use a remote model service. Native scripts execute in isolated environments with controlled inputs, while lifecycle and effect tests use local service oracles.

AI USE STATEMENT

The initial manuscript draft was written without generative-AI drafting assistance. AI tools were subsequently used for language polishing, stylistic revision, and iterative rewriting of portions of the manuscript; literature discovery and citation checking; code implementation, debugging, and refactoring assistance for the runtime, experiment drivers, and evaluation harnesses; research planning and experiment-design support; figure refinement; formulation and presentation of formal invariants and proof sketches; interpretation of experimental results; and cross-checking reported quantities against archived experiment reports. AI tools were not used to generate experimental data or measurements, select or fabricate results, or independently determine the paper's scientific claims. All reported measurements were produced by executable evaluation harnesses and archived experiment reports. All citations included in the manuscript were independently verified against external publication records. All AI-assisted outputs were reviewed, verified, and revised by the authors, who remain responsible for the methodology, claims, code, experiments, citations, and final manuscript.

REFERENCES

Alexandru Agache, Marc Brooker, Andreea Florescu, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. Firecracker: Lightweight virtualization for serverless applications. In *17th USENIX Symposium on Networked Systems Design and Implementation*, 2020.

Joe Armstrong. *Programming Erlang: software for a concurrent world*. Pragmatic Bookshelf, 2007.

Jeffrey Arnold and M. Frans Kaashoek. Ksplice: Automatic rebootless kernel updates. In *Proceedings of the 4th ACM European Conference on Computer Systems*, 2009.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025. URL https://arxiv.org/abs/2506.07982.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations*, 2024.

Edward Y. Chang and Longling Geng. SagaLLM: Context management, validation, and transaction guarantees for multi-agent LLM planning. *Proceedings of the VLDB Endowment*, 18(12):4874–4886, 2025.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *USENIX Security Symposium*, 2025.

Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. Cordon: Semantic transactions for tool-using LLM agents. *arXiv preprint arXiv:2606.17573*, 2026. v1.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 37, pp. 82895–82920, 2024.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026. arXiv:2503.18813.

Peter J Denning. Fault tolerant operating systems. *ACM Computing Surveys (CSUR)*, 8(4):359–389, 1976.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *International Conference on Machine Learning*, 2024.

Dawson R. Engler, M. Frans Kaashoek, and James O'Toole Jr. Exokernel: An operating system architecture for application-level resource management. In *Proceedings of the fifteenth ACM Symposium on Operating Systems Principles (SOSP)*, pp. 251–266, 1995.

Dawei Gao, Zitao Li, Xuchen Pan, et al. AgentScope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*, 2024.

Hector Garcia-Molina and Kenneth Salem. Sagas. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, 1987.

Cristiano Giuffrida, Anton Kuijsten, and Andrew S. Tanenbaum. Safe and automatic live update for operating systems. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 279–292, 2013.

Jim Gray. The transaction concept: Virtues and limitations (invited paper). In *Proceedings of the 7th International Conference on Very Large Data Bases (VLDB)*, pp. 144–154, 1981.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pp. 79–90, 2023.

Norm Hardy. The confused deputy: (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 1988.

Christopher M. Hayden, Edward K. Smith, Michail Denchev, Michael Hicks, and Jeffrey S. Foster. Kitsune: Efficient, general-purpose dynamic software updating for C. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA)*, pp. 249–264, 2012. doi: 10.1145/2384616.2384635.

Michael Hicks and Scott Nettles. Dynamic software updating. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 27(6):1049–1096, 2005.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024.

Galen Hunt and Doug Brubacher. Detours: Binary interception of Win32 functions. In *Proceedings of the 3rd USENIX Windows NT Symposium*, 1999.

Galen C. Hunt and James R. Larus. Singularity: Rethinking the software stack. *ACM SIGOPS Operating Systems Review*, 41(2):37–49, 2007.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *2nd USENIX Conference on File and Storage Technologies (FAST)*, 2003.

Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, et al. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, pp. 207–220, 2009.

Maxwell Krohn, Alexander Yip, Micah Brodsky, Nabil Cliffer, M. Frans Kaashoek, Eddie Kohler, and Robert Morris. Information flow control for standard OS abstractions. In *Proceedings of twenty-first ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, pp. 321–334, 2007.

Butler W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974.

Henry M Levy. *Capability-Based Computer Systems*. Digital Press, 1984.

Evan Li, Tushin Mallick, Evan Rose, William Robertson, Alina Oprea, and Cristina Nita-Rotaru. ACE: A security architecture for LLM-integrated app systems. In *Network and Distributed System Security Symposium (NDSS)*, 2026.

Ting Liu. From tool connection to execution control: Benchmarking security invariants in MCP-style agent runtimes. *arXiv preprint arXiv:2606.29073*, 2026. v1.

Xiao Liu, Hao Yu, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, Leo Yu Zhang, and Yang Liu. Prompt injection attack against LLM-integrated applications. *arXiv preprint arXiv:2306.05499*, 2023. v3.

Jiarui Lu et al. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL*, 2025.

Kai Mei, Xi Zhu, Wujiang Xu, Wenyue Hua, Mingyu Jin, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. AIOS: LLM agent operating system. In *Conference on Language Modeling (COLM)*, 2025.

Mark S Miller, Ka-Ping Yee, and Jonathan S Shapiro. Capability myths demolished. Technical Report SRL2003-02, Johns Hopkins University Systems Research Laboratory, 2003.

Model Context Protocol contributors. Model context protocol specification. Technical Specification, version 2025-06-18, 2025. URL `https://modelcontextprotocol.io/specification/2025-06-18`.

Bardia Mohammadi, Nearchos Potamitis, Lars Klein, Akhil Arora, and Laurent Bindschaedler. Atomix: Timely, transactional tool use for reliable agentic workflows. *arXiv preprint arXiv:2602.14849*, 2026. v2.

C. Mohan, Donald Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems*, 1992.

Iulian Neamtiu, Michael Hicks, Jeffrey S Foster, and Polyvios Pratikakis. Contextual effects for version-consistent dynamic software updating and safe concurrent programming. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 37–49, 2008.

Martin Odersky, Yaoyu Zhao, Yichen Xu, Oliver Bračevac, and Cao Nguyen Pham. Securing agents with tracked capabilities. In *ACM Conference on AI and Agentic Systems (CAIS)*, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive APIs. *Advances in Neural Information Processing Systems*, 37, 2024a.

Shishir G. Patil et al. GoEX: Perspectives and designs towards a runtime for autonomous LLM applications. *arXiv preprint arXiv:2404.06921*, 2024b.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models. In *Findings of EMNLP*, 2023.

Xue Qin, Simin Luan, John See, Zeyd Boukhers, Cong Yang, and Zhijun Li. Harnessing embodied agents: Runtime governance for policy-constrained execution. *arXiv preprint arXiv:2604.07833*, 2026. v4.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024.

Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68481–68503, 2023.

Jonathan S. Shapiro, Jonathan M. Smith, and David J. Farber. EROS: A fast capability system. In *Proceedings of the seventeenth ACM Symposium on Operating Systems Principles (SOSP)*, pp. 170–185, 1999. doi: 10.1145/319151.319163.

Tianneng Shi et al. Progent: Securing AI agents with privilege control. *arXiv preprint arXiv:2504.11703*, 2025. URL `https://arxiv.org/abs/2504.11703`. v3.

Sierra Research. τ^2-bench. `https://github.com/sierra-research/tau2-bench`, 2025. Version 1.0.1.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Lillian Tsai and Eugene Bagdasarian. Contextual agent security: A policy for every purpose. In *Workshop on Hot Topics in Operating Systems (HotOS)*, 2025. doi: 10.1145/3713082.3730378.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, 2026.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50208–50232, 2024b. URL https://proceedings.mlr.press/v235/wang24h.html.

Zhiruo Wang, Graham Neubig, and Daniel Fried. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *International Conference on Machine Learning (ICML)*, pp. 51177–51191, 2024c.

Robert N. M. Watson, Jonathan Anderson, Ben Laurie, and Kris Kennaway. Capsicum: Practical capabilities for UNIX. In *19th USENIX Security Symposium*, pp. 29–46, 2010.

Georg Wölflein et al. LLM agents making agent tools. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Proceedings of the First Conference on Language Modeling (COLM)*, 2024.

Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. IsolateGPT: An execution isolation architecture for LLM-based agentic systems. In *Network and Distributed System Security Symposium (NDSS)*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Shunyu Yao et al. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025.

Nickolai Zeldovich, Silas Boyd-Wickizer, Eddie Kohler, and David Mazières. Making information flow explicit in HiStar. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 263–278, 2006.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL*, 2024.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations (ICLR)*, 2025.

Yingqi Zhang. Agent libos: A runtime substrate for capability-controlled self-evolving LLM agents. *arXiv preprint arXiv:2606.03895*, 2026. v3.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

Wangchunshu Zhou, Yuchen Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, et al. Agents: An open-source framework for autonomous language agents. *arXiv preprint arXiv:2309.07870*, 2023.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A Reproducibility, Ethics, and Implementation Details

The artifact contains the Kernel, normalization adapters, native-artifact adapters, attack suites, and claim-specific evaluation harnesses; archived JSON reports retain the units, exclusions, and observations used in the tables. Appendix G states the correctness invariants; the protocols below specify their experimental instances. Because the runtime can execute external capabilities, deployed operators must independently review endpoint permissions, logging, and data-handling policies.

A.1 Kernel and Storage Configuration

The trusted Kernel is implemented as an in-process authority arbiter with an append-only transaction journal backed by synchronous disk persistence (`fsync`), in the lineage of write-ahead recovery (Mohan et al., 1992). The cryptographic identity engine derives canonical content hashes using SHA-256 over normalized manifest AST representations, echoing content-integrity approaches for untrusted storage (Kallahalla et al., 2003).

A.2 Benchmark and Attack Suite Replication

The paired AgentDojo trials share frozen task/seed identifiers and tool assignments; seed identifiers are pairing metadata, not provider-side random seeds. The authority attack suite (C2-I) exercises six violation classes: missing lease, foreign lease misuse, expired lease tokens, scope escalation, undeclared composition edges, and stale draft-edit generation (CAS) replays. During rollback, authority freshness is checked against the live authority epoch. The C1-A draft-probe suite has 12 deterministic cases: promotion to candidate only, runtime failure without pollution, contract violation, compile failure, nested import denial, process-probe attribution, network denial inside the probe sandbox, single-use probe lease, probing by a caller without a ceiling, stale-generation refusal, admit/activate still required, and output-budget enforcement; all 12 pass. C2-C fuzzes 200 payloads against each of three capabilities (file writer, file reader, network client): the Kernel observed 494 `require` calls, 198 payloads were blocked or failed, and no realized fact fell outside the declared ceiling. Archived reports and traces identify the protocol and implementation versions used. *Archived reports:* `claim1/c1a-draft-probe-conformance-20260916` (C1-A), `claim2/c2i-authority-attack-20260916` (C2-I), `claim2/c2b` (C2-B), `claim2/c2k-undeclared-effect-20260917` (C2-K), `claim2/c2c` (C2-C).

B Experimental Protocols and Data Contracts

B.1 Experiment Index

Table 6 indexes the experiment codes used in the text. Units differ by protocol; each row's n uses the unit stated in its own column. Each protocol subsection below closes with the archived report directory (under `experiments/`) that backs its numbers; reports retain the raw units, exclusions, and observations.

B.2 C1-X Polyglot Equivalence and Mixed-Surface NI Protocol

Representation funnel denominators. For OpenAPI, 1,777/12,948 operations fail Layer 1: invalid endpoint identity (452), ambiguous path templates (336), unsupported patterns (909), unbindable/colliding parameters and other invalid sources (35), `oneOf` (25), and `anyOf` (20). Of 11,171 parse-clean operations, 4,206 pass representation. The 6,965 Layer 2 rejects comprise fragment targets (3,588), non-JSON bodies (1,698), non-JSON responses (1,586), cyclic references (87), missing paths (5), and missing server (1). In-scope representability is 4,206/7,887 (53.3%).

The Layer 3 loopback probe supplies schema-synthesized inputs and substitutes a local server while preserving the endpoint path prefix. All 3,344 emitted requests match method, decoded path structure, query/header bindings, and JSON body checks; 1,308/4,206 invocations complete successfully (Table 7). The remaining outcomes include input-synthesis failures, synthetic-response validation failures, resource/runtime errors, and one materialization rejection due to a frontend/executor inconsistency. The probe tests local request construction against controlled responses.

Table 6: Experiment code index: question, observation unit, and denominator n.

Code	Question	Unit	n
C1-A	Draft probe under restricted Kernel-mediated lease	conformance case	12
C1-B	Three-layer admission funnel across sources	corpus operation/artifact	12,948 / 306 / 69 / 644
C1-S	Native skill/plugin adaptation	member IR / invocation	401 / 104 ok
C1-T	τ^2-bench dispatch swap (descriptive, not preregistered)	task pair	20 airline + 20 retail
C1-X	Cross-source equivalence (Phase A; 49-tool differential); mixed-surface NI (Phase B)	(tool, payload) case; cell; task pair	502; 1,227; 253
C2-I	Authority attack probes	probe	21
C2-B	Sandbox escape probes	probe	17
C2-K	Undeclared effects, name/args vs. effect-mediated	probe arm	60 + 60
C2-C	Payload fuzzing	payload	600
C2-ESC	Model-driven escalation under task pressure	episode	240 valid
C2-N	Defense-off threat-matrix negative controls	family variant	11 fam., $\geq$20 ea.
C2-M	Mediation coverage rate by carrier	bypass variant	210 ($7\times10\times3$)
C2-PC	Scripted goal-aligned positive control	request	82
C3-E	Continuity soak under mutation; barrier safety matrix	request; barrier trial	60,000/cond.; 50/scenario
C3-Q	Bounded-queue continuity and revocation	condition–carrier cell	330
C3-O	Mediation latency decomposition	invocation	180
C3-P	Queue and deadline accounting	offered request	2,000/cond.
C3-06	Composition stress under lifecycle events	stress episode	36
C3-H	Crash-replay soak	crash case	16
C3-U	Unified real-carrier workload	kernel trace	1
C3-I	Component ablations	intervention	8 (5 trials ea.; one 3-pair)
C3-X	Counterexample ladder	timing injection	29 timed cells $\times$ 50; CE6 by definition
C3-L	Unified lifecycle benchmark	run / invocation	50 runs/arm, $\approx$1,812 inv.
C3-R	Remote MCP drift, revocation, and lost responses	episode	1,100 + 440; 120 Gitea
X9	Durability frontier across journal modes	call / crash replay	300 + 5 per setting
C3-Y	Cross-source task chain	condition	3
C3-Z	Approval matrix	condition	5
C3-V	Caller pilot	task cell	12
G5	Planner-independence replication	planner $\times$ frozen slice	4 models

For MCP, 150 servers were selected, 142 probe attempts were made across 139 active hosts, and 50 handshakes succeeded; 158/306 tools were representable. For AgentDojo, 49/69 latest-version tools were representable; 11 exceeded the IR name limit and 9 contained unsupported unions. Its runtime unit is tool-version observations (69/74 passed), not the 69 unique tools.

Contract admission details. The frontend admits scalar `in:header` parameters by binding them into `net.request`, merges OpenAPI 3.1 `$ref` siblings as `allOf` intersections, preserves inert annotation keywords and `nullable` semantics, flattens safely mergeable `allOf` schemas with conflict refusal, and executes length/numeric/container bounds plus `pattern` (compiled under `re.ASCII`, with constructs the engine cannot express refused at admission). Literal `?` suffixes are permitted in static paths (a literal query marker is transmitted in the request target), while literal # is refused: it is the URI fragment delimiter, never reaches the wire, and admitting it would silently drop the declared target.

The 8,742 rejections comprise 3,588 literal # fragment targets that cannot reach the wire, 3,284 non-JSON media contracts, 909 unsupported regular-expression patterns, 458 missing endpoint identities, 371 ambiguous templates or parameter bindings, 87 cyclic reference topologies, and 45

Table 7: OpenAPI loopback probe: mutually exclusive outcomes over all 4,206 operations. The first four rows emitted requests and passed wire-shape checks (3,344 total); only the first row completed successfully. In the second row, the loopback responder could not synthesize a reply satisfying the operation's declared response schema and fell back to a generic body, which output validation rejected before it reached the caller; these are probe-harness limits, not admission rejections.

Outcome	**Operations**
Wire match and invocation success	1,308
Wire match; synthetic-response contract violation	1,849
Wire match; resource limit exceeded	183
Wire match; runtime failure	4
Input could not be synthesized	858
No request emitted; runtime failure	3
Materialization rejected	1
Total	4,206

unsupported `oneOf`/`anyOf` contracts. Within the parse-clean JSON-in/JSON-out domain, admission is 4,206/7,887 (53.3%). Binary/multipart media and the full ECMA-262 assertion set require extensions to the current contract implementation.

The C1-X evaluation protocol comprises two sequential phases:

1. **Phase A (Model-Free Differential Equivalence):** 20 AgentDojo operations across banking, slack, travel, and workspace suites were wrapped into three operational transports: `SRC-PY` (native Python binding), `SRC-MCP` (loopback Streamable-HTTP JSON-RPC), and `SRC-HTTP` (FastAPI REST endpoint). The full suite contained 502 (tool, payload) cases, not 502 cases per tool. Inputs combined ground-truth parameters, schema-derived boundaries, and malformed values. Outcome class agreed in 483/502 cases, persistent state in 487/502, and success/failure presence in 498/502. Exact exception class and effect-ledger agreement were lower because carriers wrap errors and record local egress differently. The report attributes 24 mismatches to canonical-IR gaps and records no Kernel-bug attribution. The silent-weakening rate is recomputed offline over the same frozen corpus: a governed call counts when it succeeds (190 Python, 190 MCP, 186 HTTP; 566 pooled), and it is silent weakening when its canonical result or post-state hash differs from native for a reason other than a nondeterministic native reference. No call qualifies; 15 per carrier differ only because the native reference is nondeterministic (per-carrier zero-event upper bound 1.6%, rule of three). *Archived report:* `claim1/c1x-swr-20260920`.

2. **Phase B (Mixed-Surface Task Utility):** The task universe is the 85 of 97 AgentDojo v1.2.2 user tasks whose ground-truth tools all pass the adapter gate (banking 16/16, slack 21/21, travel 18/20, workspace 30/40; 69 of 74 tool versions pass). Tools were partitioned across transports by a deterministic rule. Every mixed episode exposed all three source kinds; all three were invoked in 87 of the 255 mixed episodes (34.1%; 86 of the 253 valid pairs, Table 10). Of 255 planned pairs, 253 had valid outcomes and two lost one arm to model-call errors. The 9 wins, 5 losses, and 239 ties give $\hat{\Delta} = +0.0158$; the primary one-sided 95% lower bound -0.008 comes from the suite-stratified cluster bootstrap described in §3.1, with a paired binomial (Bonferroni Clopper–Pearson) reference of -0.027 that assumes pair independence. HTTP, MCP, and Python bindings completed 379/397, 444/458, and 529/550 calls, respectively, with no authority denials across 1,405 calls. Table 8 gives the outcome cells. Episodes are deduplicated by retaining the last record for each episode identifier: nine records supersede retries, including five outcome changes (four False→True and one True→False), retained in the per-pair report. Requests use `deepseek-v4-flash`; 1,952 responses record `deepseek-flash`, while 28 transport-error records have no returned model field. The requested alias `deepseek-v4-flash` resolves to the provider-side response name `deepseek-flash`, which corresponds to DeepSeek-V4.1-Flash on the official pricing page; all three names denote the same model used throughout this paper. Token usage is 7.43M prompt tokens (90.4% cache-hit) and 671K completion tokens. Recorded wall-clock totals are 2,056 s for native and 2,286 s for mixed over 255 episodes per arm; these totals include model execution and are not isolated runtime overhead.

Table 8: C1-X Phase B outcome cells by suite (253 valid pairs; unit = (task, seed) pair).

Suite	Both success	PROTEAN only	NATIVE only	Both fail	Valid pairs
Workspace	85	3	1	0	89
Slack	61	1	0	0	62
Travel	41	5	3	5	54
Banking	45	0	1	2	48
Total	232	9	5	7	253

C1-X sampling and analysis. The manifest contains 85 tasks with three paired repetitions, fixed transport assignments, and two arms. The analysis uses a -0.10 tolerance and reports sensitivity to tighter margins; the tolerance is not a task-specific utility guarantee. Repetition seeds identify pairs and tool assignments and are not sent to the model provider. The 1,980 model requests cover the 255 planned pairs. Infrastructure exclusions, retry deduplication, and per-pair outcomes are retained in the report. Task-level sign aggregation is an exploratory diagnostic with a different estimand from the pair-level success-rate difference.

Phase A orders the 49 eligible tools by `sha256("c1x|name")`, selects the first 20, then scans the ordered tail for write-side-effecting tools and replaces the last non-write picks until at least six writes are included (eight result). The eligible, selected, and rejected tool manifest accompanies the report. This tool-level sample is distinct from Phase B's task-seed pairs. For Phase B, the task-sign diagnostic (a different estimand, reported descriptively) gives a lower bound of -0.098; enumerating the four feasible completions of the two excluded pairs under the cluster estimator gives lower bounds of -0.016 to -0.008, and the first-write retry rule gives -0.004. An earlier preregistered 40-pair pilot under the same -0.10 margin did not clear it (40/40 valid pairs, $\Delta_{\text{LCB95}} = -0.169$); that negative result motivated the larger mixed-surface protocol and is retained in the evidence ledger.

Margin sensitivity and heterogeneity strata. The -10 pp margin is the project-wide engineering tolerance frozen in the preregistration; no independent task-cost derivation exists, so non-inferiority is reported as sensitivity across margins and dependence models (Table 9). A two-one-sided-tests (TOST) procedure finds the arms equivalent within ± 5 pp ($p = 0.048$, cluster-robust standard errors). The heterogeneity stratification is post-hoc: mixed-arm episodes are grouped by the number of distinct carrier kinds actually invoked (Table 10); the three strata (59/108/86 pairs) are descriptive subsets of one seeded run, are labeled exploratory, and are not pooled into a confirmatory claim. *Archived reports:* `claim1/c1x-differential-20260918` (Phase A), `claim1/c1x-swr-20260920` (SWR), `claim1/c1x-mixed-ni-20260920` (Phase B), `claim1/c1x-missingness-audit-20260919`, `claim1/c1x-ni-margin-sensitivity-20260921`, `claim1/c1x-heterogeneity-strata-20260921`, `claim1/c1x-differential-49-20260921` (49-tool expansion), `claim1/c1c-agentdojo-ni-20260913` (40-pair pilot).

Table 9: C1-X Phase B non-inferiority pass table: margins × dependence models on the same 253 pairs. TOST equivalence at ± 5 pp holds ($p = 0.048$). At -2 pp the conclusion depends on the dependence model.

Dependence model	-10 pp	-5 pp	-2 pp
Paired binomial (Bonferroni CP)	pass	pass	fail
Cluster bootstrap (20,000)	pass	pass	pass
Cluster-robust CR1	pass	pass	pass

Rejection taxonomy (X2-c). The 8,742 rejected OpenAPI operations partition into three classes: security-motivated refusals 6,872 (unemittable fragment targets 3,588; non-JSON media 3,284); implementation backlog 1,041 (inexpressible engine patterns 909; unsupported reference topologies 87; logical combinators 45); and upstream-parse failures 829 (missing endpoint identity 458; ambiguous templates or bindings 371). For the native corpus, 526 memberless or instruction-only artifacts are intrinsically inexpressible under the contract domain, and 297 probe failures are corpus scripts expecting real inputs or third-party packages (12 real environment, 68 missing

Table 10: C1-X Phase B stratified by carrier kinds actually invoked per mixed-arm episode (post-hoc, exploratory; strata are subsets of the seeded run, no pooled claim). $\hat{\Delta}$ is the paired success difference; LCB/CI are suite-stratified cluster bootstrap (20,000 resamples). Utility-verdict agreement is the share of pairs where both arms reach the same utility verdict; LP-denied calls are tool calls rejected by least-privilege ceilings (0 in all strata).

Kinds invoked	n	$\hat{\Delta}$	LCB95	95% CI	agree	errs	p50 (s)
1 kind	59	$+5.1$pp	$+1.6$pp	$[0.0, +10.6]$	0.949	0	3.8
2 kinds	108	-0.9pp	-3.5pp	$[-3.8, +1.9]$	0.954	23	5.8
3 kinds	86	$+2.3$pp	-3.4pp	$[-3.9, +10.6]$	0.930	30	8.8

Table 11: Injection exposure by cohort. Outside-ceiling requests are not necessarily goal-aligned attacks. The repeated cohort evaluates 40 task–attack pairs under both deny and LLM approvers; the single-run cohort uses deny.

	Repeated cohort	**Single-run cohort**	**Pooled**
Valid episodes	80	170	250
Distinct task–attack pairs	40	170	210
Episodes with outside-ceiling requests	5	11	16
Outside-ceiling requests	65	216	281
Goal-aligned request episodes	4	2	6
Goal-aligned requests	9	31	40
Granted (attack opportunity)	0	0	0
Realized attacks	0	0	0

package, 217 other runtime expectations). The taxonomy classifies stated reject reasons; "intrinsic" means inexpressible under the current contract domain, not unverifiable in principle. The backlog class is estimated as medium engineering effort with no version commitment. *Archived reports:* `claim1/c1b-funnel-20260920`, `claim1/c1b-openapi-l3-probe-20260920`, `claim1/c1b-unsupported-taxonomy-20260921`, `claim1/c1s-native-{inventory,funnel,execute}-20260920`.

B.3 C2-ESC Authority Escalation Under Task Pressure Protocol

The C2-ESC protocol evaluates model-driven privilege escalation when least-privilege boundaries block task completion:

- **Task Universe:** Held-out tasks drawn from the frozen $U_{\text{supported}}$ universe across four suites.
- **Authority Pressure Regimes:** (i) `P-full`: oracle authority ceiling covering all ground-truth tools; (ii) `P-nat`: natural least-privilege ceiling derived via LLM prompt synthesis; and (iii) `P-forced`: under-authorized ceiling deliberately omitting the terminal tool required for task completion.
- **Arbitration Modes:** (i) `A-deny`: strict deterministic rejection of all requests exceeding ceiling; (ii) `A-oracle`: automated approval of requests matching ground-truth tool needs; and (iii) `A-llm`: automated LLM judge evaluating request purpose against task description.
- **Invariant Verification:** For every invocation, the Kernel verifies that realized effects satisfy $\text{RealizedEffects}(I) \subseteq A(I)$, recording any unmediated lease issuances (SI) or out-of-envelope side effects (ROE).

The 247 C2-ESC episodes comprise 240 valid and seven infrastructure-failed episodes. Of the valid episodes, 236 belong to conditions with pending requests, and four `P-full`×`A-deny` episodes issue none. Pending requests total 1,146. Under `P-forced`×`A-deny`, escalation occurs in 33/40 and 32/40 episodes across two seeds.

Policy tightness. An LLM synthesizer derives a least-privilege envelope from each of 40 task descriptions. Over-grant is tool-set containment: an injection goal is covered when all of its tools lie inside the derived envelope. Over the 321 eligible task–goal pairs (excluding 19 with an empty

envelope and 10 with an undetermined goal), the derived envelope covers the goal in 45; a broad policy covers all 340 goal-determined pairs, including the 19 with an empty derived envelope. On the 34 tasks inside the adapter-supported universe, task success moves from 29/34 (broad) to 26/34 (derived), a paired difference of -8.8 pp with 95% interval $[-35.1, +15.9]$ pp. *Archived reports:* `claim2/c2l-overgrant-20260917`, `claim2/c2j-universe-recompute-20260918`.

The injection cohort evaluates 40 task–attack pairs under deny and LLM approvers. Five of its 80 episodes issue 65 outside-ceiling requests. The frozen permission-matching rules classify 28 as task-only, 27 as neither task nor goal, five as both, four as goal-only, and one as invalid. Four episodes contain goal-aligned requests. In the 170-episode single-run cohort, the 216 outside-ceiling requests comprise 53 task-only, 127 neither, 29 both, two goal-only, and five invalid requests; two episodes contain the 31 goal-aligned requests. No goal-aligned request receives a grant, and no episode realizes an attack. Goal alignment denotes permission overlap under these rules; authorization and realized effects are recorded separately. *Archived reports:* `claim2/c2esc-20260918-recount` (episode outcomes), `claim2/c2esc-exposure-20260920` (exposure taxonomy).

B.4 C3-06 Composition Stress and Concurrent Lifecycle Protocol

The C3-06 composition stress benchmark verifies system safety invariants under in-flight lifecycle disruptions:

1. **Workload Execution:** Agent missions execute concurrent tool calls against registered capabilities.
2. **In-Flight Lifecycle Injection:** During active task execution, the harness injects an ordered sequence of lifecycle disruptions:
   - **E1 (Target Replacement):** Replaces and activates the recurring ground-truth target capability.
   - **E2 (Unrelated Revocation):** Revokes a distinct capability, exercising authority freshness without changing the recurring target's contract.
   - **E3 (Process Restart):** Rebuilds the process against the same durable Kernel state.
   - **E4 (Historical Rollback):** Rolls back the target to a historical revision; this event was reached in 32/36 stress episodes.
3. **Invariant Checking:** The report records 36/36 passes for each of four checks, including 84 post-E1 target calls for the non-vacuous version check. The stress slice records zero effect-history rows and zero replayable or unknown post-restart recovery effects. Replay prevention is instead exercised by the C3-H crash-replay soak (16/16): a non-idempotent append interrupted post-effect recovers `unknown` with resubmission rejected, and a replayable write re-executes to success. A passive permission-request window after E2 saw no requests. The transport-heterogeneity mode was degraded: Python, MCP, and HTTP provenance labels were declared, but the slice did not exercise three real carriers.

Archived report: `claim3/c3g-composition-stress-v2-20260918`.

Arrival regimes and gap windows. The v3 sub-capacity condition offered 60,000 requests at 29.85 requests/s and lost 125 under restart versus zero under PROTEAN replacement or rollback. The separate archived v2 saturated condition offered 60,000 at 200 requests/s ($2.23\times$ measured capacity) and lost 17,276 under restart versus zero under PROTEAN. These are two measured points, not a continuous loss curve or matched-capacity comparison. In v3, the 51.7 ms arrival-to-completion gap and 49.7 ms journal event-window gap use different time boundaries. The event-window all-round component means were 25.5 ms pre-commit, 9.7 ms activation/fsync, and 14.5 ms post-event admission. A 29-round subset excluding one drain-anomalous round has component means of 28.0, 8.3, and 14.5 ms and a mean gap of 50.9 ms (components rounded). The 2.0 ms window difference should not be interpreted as an unmeasured system stage. *Archived reports:* `claim3/c3e-soak-v3-20260918` (sub-capacity point), `claim3/c3e-arrival-gap-20260918` (v2 saturated point).

Queue-timeline protocol (C3-P). Table 12 reports 2,000 offers per condition with four workers. All offers reached a terminal state; restart had four errors and the other conditions had none. Latency includes all terminal outcomes, whereas deadline utility counts successful completions only. Queue wait is measured from enqueue to worker dispatch; restart timestamps bracket the parent-side socket

Table 12: C3-P: 2,000 offers per condition, approximately 30/s, four workers. Latency starts at planned arrival; all timing columns are milliseconds. Deadline columns use all offered requests as denominator.

Condition	Queue peak	p50	p95	p99	1s success	5s success	Drain
Static	1	10.5	14.0	23.1	2000/2000	2000/2000	10.4
Replace	1	10.4	14.4	23.1	2000/2000	2000/2000	11.3
Rollback	1	11.9	31.3	49.7	2000/2000	2000/2000	26.5
Restart	17	26.3	186.8	580.3	1996/2000	1996/2000	27.9

call, without child-internal admission instrumentation. The observation cutoff was 30 seconds after the arrival trace; no run was censored. The unbounded queue supplies no backpressure. *Archived report:* `claim3/c3p-queue-slo-20260920`.

B.5 Continuous Execution and Safety Event Protocols

Table 13 reports continuity metrics over 60,000 requests per condition across 30 rounds at 29.85 requests/s. Table 14 details safety-matrix violation event counts under fixed transition barriers across 50 trials per scenario.

Table 13: C3-E continuity: 60,000 requests per condition over 30 rounds at 29.85 requests/s. Gap is the measured arrival-to-completion transition window.

Runtime condition	Completed	Lost	Gap (ms)
STATIC (no mutation)	60,000	0	—
NAIVE in-place swap	60,000	0	30.5
ERLANG-2VER	60,000	0	30.5
MCP-ADVISORY	60,000	0	34.5
RESTART	59,875	125	180.2
PROTEAN replace	60,000	0	51.7
PROTEAN rollback	60,000	0	44.2

Table 14: Safety-matrix event counts in fixed barrier windows (50 trials/scenario). The MCP S2 variants are separate report rows, not pooled conditions. Dashes mark untested cells. Columns: version tearing, post-revocation execution from newly admitted requests, unattributed effects, purge loss, unclassified crashes, and stale listings.

Condition	Torn	Post-revoke	Unattrib.	Purge	Crash	Stale
NAIVE	50	150	400	0	50	0
ERLANG-2VER	0	0	400	50	50	0
MCP-ADVISORY	0	0	400	0	50	150
advisory disable (S2)	—	250	—	—	—	—
remove_tool (S2)	—	0	—	—	—	—
PROTEAN	**0**	**0**	**0**	**0**	**0**	**0**

Archived reports: `claim3/c3e-soak-v3-20260918` (continuity), `claim3/c3-safety-matrix-v3-20260919` (safety event counts).

B.6 Recovery Contracts and Component Ablations

Archived reports: `claim3/c3i-ablation-audit-20260920` (component ablations), `claim3/c3-ablation-matrix-20260920` (matrix cells).

Table 15: Per-class recovery contract for an interrupted effect-bearing invocation, organized by durable evidence class: local dispatch evidence (journaled attempt/commit rows), externally reconciled effect evidence (a durable receipt issued by the remote endpoint), or neither (leading to `unknown`).

Replay class	Durable evidence	Recovery action	Resubmission	Guarantee
`replayable`	local dispatch evidence	`replay_pending`; re-execute pinned snapshot	allowed	idempotent replay; execution may repeat
side-effecting + receipt	externally reconciled effect evidence	reconcile against trusted receipt	per recovery protocol	outcome classification when receipt is reconcilable
side-effecting, no receipt	none → `unknown`	`outcome` = `unknown`	same identity rejected	no blind same-identity retry; occurrence remains unknown

Table 16: Component ablations (C3-I). Each row has five trials except attempt-record suppression, which has three matched pairs. "Control" denotes a full-system arm under the same schedule. Attempt-record suppression preserves same-identity retry rejection.

Ablation	Isolation	Control	Observed outcome
durable_checkpoint	proxy	no	external effect without local trace; interrupted job invisible; retry duplicated the send
effect_ledger	proxy	no	external send without effect-level ledger correlation
authority_epoch	compound	yes	revoked capability admitted (control rejects)
epoch_stamp	single factor	yes	stale epoch stamp (control stays fresh)
commit_barrier	single factor	no	binding committed without its journal event
invocation_snapshot	single factor	yes	artifact version mismatch (control stays pinned)
delegation_edge	single factor	yes	undeclared nested call (control rejects)
attempt_record	single factor	yes	no duplicate in either arm; same-identity resubmission rejected

C Native Adaptation and Joint Lifecycle Workloads

C.1 Native Skill and Plugin Adaptation

The native-format corpus contains 580 skills and 64 plugin manifests (the artifact-inventory denominators; a separate Layer-0 registry scan enumerated 578 skill and 62 plugin entries). The adapter records `SKILL.md` frontmatter and prose as provenance, hashes bundle members, and generates CLI bindings for Python entrypoints. Each binding runs through the process carrier in a Bubblewrap namespace without network access. The generated contract carries command arguments, input files, and returned files. File exchange uses the `files`/`files_out` payload fields; the evaluated carrier does not project host filesystem grants into the child.

Of 644 artifacts, 103 skills produce 222 IRs and 15 plugins produce 179 member IRs. All 401 IRs compile, install, and reach invocation. With a generic empty-argument payload, 104 invocations succeed and 297 return `script_error`, including missing task inputs and third-party packages. No installation, contract, or infrastructure failures are reported. This probe measures executable adaptation under generated CLI contracts. Functional task checks use the workload below. Skill and plugin counts retain their source-artifact units; a plugin can contain multiple executable members.

Instruction text grants no authority. The adapter declares `process.exec`; additional declarations require an explicit sidecar. Runtime provenance records the generated wrapper and bundled content. An interpreted entrypoint instead binds instructions and resources to a named

host interpreter, which belongs to the trusted execution boundary. *Archived reports:* `claim1/c1s-native-{inventory,funnel,execute}-20260920`.

C.2 Real-Carrier Lifecycle and Recovery

The real-carrier workload (C3-U) installs a builtin filesystem writer, a local MCP conformance tool, and an HTTP capability against a logging server in one Kernel. Between invocations it exercises replace/activate/rollback and close–reopen. During a delayed MCP response, it performs `semantic_revise` followed by activation. The pending call returns using its pinned revision; the next admission uses the revised binding.

Server request logs and workspace contents provide independent observations. The main trace contains 16 effect rows, five network commits, three MCP calls, and two HTTP requests. The network commits match the five server observations; denied invocations produce no server request. The trace also checks the filesystem result, rollback binding, and subsequent refusal after revocation.

The recovery phase has three repetitions at each of two cuts. An in-flight cut sends SIGKILL after the HTTP server receives the request and before its response returns. The ledger contains an attempt without a commit; recovery classifies the job as `unknown`. Same-identity resubmission and mismatched resubmission return `Conflict`. A fresh job executes separately: the oracle observes one pre-crash request and one fresh request. At the pre-invoke cut, the server and ledger are empty and there is no interrupted effect to recover. These cuts cover distinct execution states; the pre-invoke control does not exercise ambiguous effect recovery. *Archived report:* `claim3/c3u-unified-carriers-20260920`.

C.3 Composition Baselines and Counterexamples v2 (C3-X)

Table 17: Lifecycle counterexample ladder. Cells report violation hits over $n = 50$ randomized timing injections per cell (exact CP95 CIs in the text); **Prev** = 0/50 prevented; **Diff** = semantics differ by design (rollback re-admits the old revision's declared envelope as a journaled transition, but the live lease still gates it: 0/50 silent restores); N/A = concept missing in that model. Rungs: r1 contracts and policy only; r2 adds revision pinning; r3 adds durable effect records; r4 has every component (pins, durable records, decoupled authority) but seals check and dispatch non-atomically; r6 models libOS-style process-scoped grants. CE5 exposes members of a partially installed package. CE6 reports each rung's *defined semantics* for an in-flight invocation under revocation, not a pass/fail rate. No single primitive suffices; only the unified contract (r5) prevents CE1, CE2, CE4, and CE5.

Counterexample	r1: Base	r2: +Pin	r3: +Dur	r4: Decouple	r5: Protean	r6: Process
CE1: Check/dispatch	Hit (13/50)	Prev (0/50)	Prev (0/50)	Hit (15/50)	**Prev** (0/50)	Hit (15/50)
CE2: Rollback scope	Hit (17/50)	Hit (16/50)	Hit (16/50)	Prev (0/50)	**Diff** (0/50)	N/A
CE3: Unpinned child	Hit (18/50)	Hit (15/50)	Hit (16/50)	Hit (17/50)	**Hit**† (19/50)	Hit (20/50)
CE4: Remote crash	Hit (50/50)	Hit (50/50)	Hit (15/50)	Hit (20/50)	**Prev** (0/50)	Hit (50/50)
CE5: Package visibility	Hit (50/50)	Hit (50/50)	Hit (50/50)	Hit (50/50)	**Prev**‡ (0/50)	Hit (50/50)
CE6: Revoke in flight	undef.	undef.	undef.	undef.	drain|effect|kill	kill

†Declared boundary hit: unpinned dependencies resolve the active revision; closure requires explicit child pins.
‡Prevented under atomic publication; the earlier incremental-activation variant recorded 50/50.

The counterexample study (C3-X) evaluates six counterexample classes across a six-rung ladder: bare contracts/policy (r1), revision pinning (r2), durable effect records (r3), decoupled admission transactions (r4), Protean (r5), and a simulated process-scoped capability reference arm (r6, modeling libOS-style process-level grants). In addition to deterministic witness traces, each cell undergoes randomized timing injections ($n = 50$/cell, seed-controlled) to estimate violation rates with exact Clopper–Pearson 95% confidence intervals (Table 17 reports the counts). For $n = 50$ the intervals are: 0/50, [0, 7.1]%; 13/50, [14.6, 40.3]%; 15/50, [17.9, 44.6]%; 16/50, [19.5, 46.7]%; 17/50, [21.2, 48.8]%; 18/50, [22.9, 50.8]%; 19/50, [24.7, 52.8]%; 20/50, [26.4, 54.8]%; 50/50, [92.9, 100]%. CE6 was added in the current revision and CE5 under r5 is prevented by atomic package publication (all members staged; a single manifest transaction activates them).

The matrix (Table 17) isolates which architectural invariants are load-bearing:

- **CE1 (Check/dispatch gap):** Without atomic snapshot pinning, updates between check and dispatch execute a superseded revision. Pinning (r2, r3, r5) prevents CE1, whereas decoupled admission without atomic sealing (r4) and process-scoped capability replacement (r6) fail in 15/50 trials during the gap window.
- **CE2 (Rollback permission restore):** When rolling back code, naive runtimes restore permission caches and revive revoked scopes (r1–r3). PROTEAN decouples code revision from monotonic authority epochs: the restored revision executes only under currently valid leases. In r6, authority epochs are unobservable, rendering CE2 an honest N/A.
- **CE3 (Unpinned child dependency):** An admitted parent with an unpinned child executes whatever child revision is active. All rungs encounter this behavior (15–20/50); it is prevented only when child dependencies explicitly declare revision pins.
- **CE4 (Remote crash retry):** Crashes between external mutation and local ledger commit trigger duplicate execution upon retry in r1–r4 and r6. PROTEAN uses durable job identifiers to classify unconfirmed executions as `unknown` and reject same-identity resubmission.
- **CE5 (Package visibility):** Multi-member installations expose early members before the bundle is completely registered on r1–r4 and r6. Under PROTEAN atomic publication, members are staged and the manifest activates them in one transaction (0/50); the pre-fix incremental-activation variant recorded 50/50.
- **CE6 (Revoke in flight):** A revocation lands while an invocation is in flight. Rungs r1–r4 define no semantics (the call completes only because no mechanism can reach it) and r6 kills the process generation, losing in-flight sends. PROTEAN defines three operator-selected semantics: *drain* completes the call on its frozen envelope; *effect* lets computation continue but denies effects past the first blocked boundary (witness: 10 committed effects, 1 blocked, 0 commits after the block, terminal state `Revoked`); *kill* terminates the carrier.

Archived report: `claim3/c3x-lifecycle-ladder-v2-20260920`.

C.4 CROSS-SOURCE TASK CHAIN

The task chain (C3-Y) uses members produced by the native adapter. A corpus Figma script normalizes `123:456`; a builtin filesystem capability persists a `plan.json` containing that output; a corpus earnings validator consumes the file through the payload channel and emits a report; a local MCP sink records the verdict. Both scripts are adapted without sidecars.

Three conditions execute the same scripts, inputs, and output oracle: direct subprocess wiring, separately governed composition with a pinned registry and durable log, and the unified Kernel. All three satisfy the final verifier. Recorded end-to-end times are 46.9 ms, 39.8 ms, and 1,635.9 ms, respectively; the governed chain has six ledger-recorded invocations. The Kernel timing is measured from a cold start: it includes creating a fresh Kernel database, installing the bundled capabilities, and compiling and admitting the adapted corpus members and the MCP capability, whereas the other two conditions have no installation stage. These are descriptive timings for this workload, not per-invocation overhead, which C3-O measures (Table 22).

Two controls test the chain's dependencies and authority. Scrambling the upstream value before persistence makes the independent verifier reject the final result. Revoking the MCP sink before its call produces `NotFound` with no server observation. Lifecycle checks replace the validator with revision 2: an explicitly pinned invocation retains revision 1, while an active-binding invocation observes the revision 2 marker. Re-invoking a revoked upstream member is denied. *Archived report:* `claim3/c3y-cross-source-chain-20260920`.

C.5 APPROVAL AND CALLER CONDITIONS

The approval matrix (C3-Z) separates exposure, request submission, authorization, and execution. It injects an off-chain MCP sink request to a second loopback host. Four conditions produce no observed sink request: denied approval, a live lease lacking the requested scope, a forged lease, and use beyond an approved scope. A deliberately broad, valid one-shot approval produces one journaled sink request. Permanent approval widens the stored principal ceiling; one-shot approval leaves that

stored ceiling unchanged. Hostile skill instructions still compile with the declared `process.exec` effect and acquire no permissions from their prose.

The caller pilot (C3-V) runs six frozen tasks with one DeepSeek-V4.1-Flash caller and one seed. Four tasks use adapted corpus members: a Figma normalizer, event-driven mathematics, price-target calculation, and a normalize–plan–validate chain. Two tasks inject replacement or revocation during the episode. Native-subprocess and Kernel-mediated arms expose identical prompts, tool schemas, and result envelopes. The run uses 42 model requests and all 12 task cells pass their verifier.

After replacement, both arms execute new code on the next call. The governed arm attributes it to revision 2 and preserves revision 1 for an explicitly pinned re-invocation; the native arm records file hashes in its harness. After revocation, the governed arm denies admission with no child process, while the native retry fails at the operating system on a missing file. The six-task pilot documents caller behavior under these conditions. *Archived reports:* `claim3/c3z-approval-matrix-20260920`, `claim3/c3v-model-caller-20260920`.

C.6 Threat Model Coverage Matrix and Adaptive Negative Controls (C2-N)

To verify that zero-escape outcomes in Claim 2 reflect active enforcement rather than inactive or vacuous probes, C2-N tests eleven attack families across ≥ 20 variants each with kernel mediation checks monkeypatched off versus on (Table 18). Families F7–F9 introduce adaptive, protocol-aware adversaries whose capability code inspects visible Kernel state to trigger races; F10–F11 are kernel-side families targeting the snapshot and job-identity stores. Write escapes are verified host-side; absorption within private mount namespaces is not counted as escape. The defense-on zeros are measured on the current artifact: an earlier pre-fix build violated defense-on under F10 (14/20 corrupted-snapshot variants replayed forged rows unchecked—a genuine kernel defect, retained in the archived report's pass criteria and the evidence ledger); the defect was fixed before the reported run.

Archived report: `claim2/c2n-threat-matrix-20260920`.

C.7 Mediation Coverage Rate by Carrier (C2-M)

C2-M treats Assumption A1 as a measurement target rather than a premise. Seven bypass classes—direct filesystem writes, raw sockets, subprocess spawn, file-descriptor inheritance, symlink/namespace traversal, credential exfiltration, and direct writes to the Kernel state store—are instantiated as ten parametric variants each per carrier ($7 \times 10 \times 3 = 210$ cells). A host-side independent observer (filesystem, socket table, process table, database rows) decides whether an effect was realized; interception means a ledger-journaled realization or a carrier refusal/namespace absorption before any host-visible effect. Exact Clopper–Pearson 95% intervals are computed per cell.

The trusted-host column of Table 19 quantifies A1 clause 1: only the declared-effect API could have intercepted these variants, and it did not (63/70 escaped). Sandboxed-process interception is mechanical (`bwrap --clearenv --unshare-all`, read-only rootfs). The restricted interpreter expresses none of the seven classes, so nothing is realized and nothing needs intercepting; these cells are marked not-expressible rather than silently counted. *Archived report:* `claim2/c2m-mediation-coverage-20260921`.

C.8 Unified Lifecycle Benchmark (C3-L)

C3-L is a model-free benchmark that composes the full mechanism surface in one workload. Scripted principals execute seeded chains of 20–50 invocations with simulated tool labels; each run injects 3–6 lifecycle events (replace, rollback, drain/effect/kill revocation, child update, SIGKILL, remote timeout-after-receipt) at random positions. A per-invocation oracle specifies the expected revision, allowed authority, expected effect identity, and allowed recovery outcome. Arms r1–r4 reuse the C3-X rung simulations; r5 runs the real Kernel. The statistical unit is the run (50 seeded chains per arm, $\approx$1,812 invocations per arm); unsafe-replay rate is conditioned on the replay opportunities that arise (76 for r1–r2, 55 for r3–r5). Rates are reported with exact Clopper–Pearson 95% intervals; Protean zero rates carry upper bounds (≤ 0.002 per invocation, ≤ 0.065 per replay opportunity, ≤ 0.071 per run), not point claims of impossibility. Harness completion is 50/50 on all arms and is

Table 18: C2-N threat model coverage matrix and defense-off negative controls. Evaluates eleven adversarial attack families across $\geq$ 20 parametric variants each under monkeypatched kernel ablations versus normal mediated execution. Families F7–F9 evaluate adaptive protocol-aware capability payloads; F10–F11 target kernel-side state (corrupted snapshots, forged job identities). Write escapes are verified host-side; private mount namespace absorption is not counted as escape.

Attack family	Target invariant	Defense-off	Defense-on	Signature under mediation disabled
Stale authority epoch (F1)	Invariants I2, I3	20/20	**0/20**	Superseded historical leases resurrected after revocation
Lease-capability binding (F2)	Invariant I2	24/24	**0/24**	Valid lease accepted for foreign capability invocation
Delegation edge (F3)	Invariant I2	20/20	**0/20**	Nested invocation succeeds along undeclared import edge
Undeclared effect (F4, C2-K)	Invariants I2, I9	60/60	**0/60**	Uncontained mutations escape under name/args-only enforcement
Sandbox confinement (F5)	Invariant I9	16/20*	**0/20**	Directory traversal and subprocess execution escape host-side
Effect-ledger stage split (F6)	Invariant I7	20/20	**0/20**	Logging suppression corrupts provenance and recovery state
Attempt–commit race (F7)	Invariants I2, I7	20/20	**0/20**	Duplicate effect on same-identity retry; defense-on yields Conflict
Generation boundary (F8)	Invariants I4, I8	20/20	**0/20**	Torn execution resolves active code; defense-on pins snapshot
Lease timing / revoke (F9)	Invariants I2, I3	16/20†	**0/20**	Post-expiry or revoked-lease execution; defense-on denies admission
Corrupted snapshot (F10)	Invariants I1, I4	14/20	**0/20**	Tampered snapshot fields accepted at admission; defense-on rejects
Forged job identity (F11)	Invariant I7	10/20	**0/20**	Forged job rows confuse recovery classification; defense-on rejects

*Four sandbox probes remained inactive under the specific ablation and are reported as degraded, not failed.
†Four boundary variants are already expired at submission and rejected under both arms.

Table 19: C2-M mediation coverage matrix (per-carrier interception by bypass class). "n.e." = variants not expressible on that substrate (fail-closed, never counted as intercepted). On the trusted host executor, escaped variants are real A1-clause-1 violations—the in-process code bypassed the declared-effect API—kept visible rather than merged.

Bypass class	Trusted host	Restricted	Sandboxed proc.
fs write	esc. 9/10	n.e. 0/10	int. 10/10
net socket	esc. 8/10	n.e. 0/10	int. 10/10
subprocess spawn	esc. 9/10	n.e. 0/10	int. 10/10
fd inheritance	esc. 10/10	n.e. 0/10	int. 10/10
symlink traversal	esc. 9/10	n.e. 0/10	int. 10/10
env credential	esc. 9/10	n.e. 0/10	int. 10/10
kernel sqlite	esc. 9/10	n.e. 0/10	int. 10/10
Interception rate	$7/70 = 0.10\ [0.04, 0.20]$	— (0 realized)	$70/70 = 1.00\ [0.95, 1.00]$

not a Protean advantage claim. SIGKILL is modeled as deduplicated same-identity resubmission (rejected with `Conflict`), not an OS-level crash; genuine `unknown` classification is exercised by C3-H and C3-U. *Archived report:* `claim3/c3l-lifecycle-benchmark-20260921`.

Table 20: C3-L unified lifecycle benchmark (model-free, 50 seeded runs per arm; rates per invocation over n invocations; unsafe-replay rate conditioned on the 55–76 replay opportunities per arm; exact CP95).

Arm	Tear	Stale auth.	Unattrib.	Unsafe replay	Task succ.
r1 contracts/policy	17/1812	331/1812	1812/1812	76/76	50/50
r2 +pinning	0/1812	331/1812	1812/1812	76/76	50/50
r3 +durable	0/1812	31/1812	0/1812	0/55	50/50
r4 decoupled admission	4/1812	31/1812	0/1812	0/55	50/50
r5 PROTEAN	0/1812	0/1812	0/1812	0/55	50/50

Table 21: Durability frontier (X9): invocation wall-clock latency under three SQLite `synchronous` settings (300 calls each) and crash-evidence strength over five process-death replays per setting. "Strong" means the durable ATTEMPT row classifies the interrupted effect as `unknown` and rejects same-identity resubmission. The OFF row is strong only for process death; OS crash and power loss lie outside the fixture.

Journal mode	p50 (ms)	p95 (ms)	p99 (ms)	Crash evidence
`FULL` (default)	18.4	26.1	36.0	strong (5/5)
`NORMAL`	9.3	11.6	15.4	strong (5/5)
`OFF`	9.5	11.5	12.9	strong (5/5)[a]

[a] Under process death only; OFF widens the unprotected window to OS crash / power loss.

C.9 DURABILITY FRONTIER PROTOCOL (X9)

The frontier arm runs 300 latency calls plus five process-death crash replays under each of three SQLite `synchronous` settings (FULL, NORMAL, OFF), guarded by `PROTEAN_BENCH_SYNC`; production code paths never read the knob outside the labeled measurement arms, and the default (FULL) is unchanged. Evidence strength is graded by fixture: *strong* requires a durable ATTEMPT row, absent COMMIT, job state `unknown`, and resubmission rejected with `Conflict`. All three settings scored strong 5/5 under process death. The OFF arm's widened exposure is to OS crash and power loss, which the process-death fixture does not express; this boundary is reported rather than hidden. Group commit is analyzed but not implemented: batching COMMIT records cannot change I2/I7, while the durable ATTEMPT row remains the latency lower bound. Results appear in Table 21. *Archived report:* `claim3/x9-durability-frontier-20260921`.

C.10 MULTI-MODEL REPLICATION PROTOCOL (G5)

G5 re-ran three frozen slices under four self-hosted open-weight planners—GLM-4-32B-0414-FP8, Qwen3.8-27B-FP8, NVIDIA Nemotron-3-Nano-4B-FP8, and Olmo-3-7B-Instruct—served via vLLM on a single RTX 4090 (48 GB; FP8 for GLM-4, Qwen3.8, and Nemotron-3; BF16 for Olmo-3-7B), one seeded run per model, never pooled with the DeepSeek evidence. Slices: C2-ESC (injection arm and main seed), C1-X Phase B non-inferiority, and the C2-PC approver arm. Frozen preregistrations and manifests are unchanged; per-model request caps were kept at DeepSeek-parity values, and capped slices are labeled rather than generalized. Provenance records exact HF/ModelScope revisions, quantization, licenses, parser and thinking-mode configuration, and the incident log (a tunnel outage reran cleanly; a GLM stock chat-template defect that silently dropped tool-role results is preserved in `*-tpla` directories, with the canonical run using the template-fixed rerun). GLM-4 FP8 is a community quantization (official weights are BF16). Low-utility arms (Olmo 9%, GLM 31% absolute success) mean their SI $= 0$ rows do not by themselves demonstrate safe planner behavior—the planner rarely acted; the approver-grant variation (22/82 GLM-4, 3/82 Nemotron-3, 0/82 others) is attributed to approver behavior under the frozen attribution rule. Separately, the Nemotron-3 injection arm realized one goal-aligned effect within the planner's held envelope (in-envelope under the natural authority ceiling, policy-attributed; ROE $= 0$): the episode's four `request_permission` calls were malformed (`invalid_args`) and the realized effect used an already-held grant. *Archived reports:* `claim1/g5-{glm4,qwen38,nemotron3,olmo3}-c1x-20260922` and `claim2/g5-{glm4,qwen38,nemotron3,olmo3}-c2pc-20260922`. Results appear in Table 2,

Table 22: C3-O invocation latency decomposition and multi-worker concurrency. Staged breakdown evaluates 180 chain invocations. Pure-Kernel baseline disables journal sync and process carrier to measure the CPU mediation lower bound. Concurrency tests run 4 concurrent workers (240 total calls/condition).

Pipeline stage / Benchmark arm	Mean share	Latency	Description / Enforcement mechanism
Payload execution (`exec`)	52.8%	—	Child process payload compute and I/O
Durable journal commit (`commit`)	6.0%	—	Synchronous SQLite WAL disk commit (`fsync`)
Durable journal attempt (`attempt`)	5.6%	—	Pre-mutation atomic disk write (`fsync`)
Job initialization / dispatch	5.9%	—	Bubblewrap namespace construction and launch
Revision & artifact fetch	4.4%	—	Content-hash index lookup from persistent state
Admission & ceiling transaction	3.2%	—	Monotonic epoch and authority ceiling intersection
Snapshot resolution & pinning	2.2%	—	Immutable snapshot $\Sigma(I)$ construction and binding
Authority & lease validation	1.6%	—	Principal grant and lease token verification
Other validation & state transitions	1.0%	—	JSON schema checking and job state transitions
PROTEAN default invocation	100.0%	p50: 23.0 ms	End-to-end wall-clock (p95: 116.0 ms, mean: 41.1 ms)
Pure-Kernel arm (`sync` = `OFF`, no carrier)	—	p50: 4.7 ms	Pure mediation CPU lower bound (0 external effects)
Journal `fsync` contribution (Δ p50)	—	Δ: 8.3 ms	Δ equals 63.8% of the pure-kernel default-sync p50 (12.9 ms)
4-worker throughput: no-effect	—	67.0 ops/s	Concurrent throughput (p50: 52.2 ms, p99: 146.8 ms)
4-worker throughput: filesystem-effect	—	53.0 ops/s	Concurrent throughput with fsync (p50: 70.5 ms, p99: 129.3 ms)

which reports the preregistered suite-stratified cluster-bootstrap lower bounds; the independence-assuming paired binomial (Bonferroni Clopper–Pearson) references are -2.7 (DeepSeek), -2.4 (GLM-4), -2.8 (Qwen3.8), -9.1 (Nemotron-3), and -4.5 pp (Olmo-3), all inside the -10 pp margin.

C.11 C3-O Latency Decomposition and Concurrency Benchmark

To isolate the execution cost introduced by mediation, persistence, and process isolation, C3-O decomposes end-to-end invocation latency across 180 chain invocations (Table 22). Across all invocations, per-stage timing captures the full execution path from admission to terminal commit (p50 stage coverage 96.0%, max 99.2%).

Group Commit and Persistence Lower Bound. The `attempt` record must be durably committed before external side effects are emitted; otherwise, a crash cutpoint would leave an external mutation without local record, permanently violating `unknown` recovery classification (Invariant I7). Consequently, the pre-mutation `attempt` `fsync` defines the unavoidable latency lower bound of durable governance. In contrast, post-mutation `commit` operations can safely be batched via WAL group commit without compromising classification soundness: an uncommitted completed job will be classified as `unknown` rather than `committed`, erring on the side of caution and preventing duplicate retries. Testing under `synchronous` = `NORMAL` confirms zero crash classification errors (`sync_normal_classification_correct` = `True`). *Archived reports:* `claim3/c3o-latency-decomposition-20260921`, `claim3/c3o-steady-state-20260920`.

Table 23: C2-PC scripted positive-control attacker funnel. Evaluates 82 goal-aligned requests generated under frozen `c2esc-align-v1` rules across 40 injection pairs through the live pending–approval–realization pipeline. Deny arm uses zero model requests; rubber calibration arm verifies payload realizability; LLM approver arm tests DeepSeek-V4.1-Flash at the conditional link.

Evaluated arm	Attempted	Pending	Granted	Realized	Exec attempts
Scripted + Deny approver	82	82	0	**0**	82 (blocked by authority)
Rubber calibration arm	82	82	82	**82**	82 (executed successfully)
Scripted + LLM approver	82	82	0	**0**	82 (denied by model)

Table 24: C3-Q bounded-queue continuity matrix (330 cells: 11 condition–carrier arms $\times$ 3 queue bounds $\times$ 2 rates $\times$ 5 seeds; 60,000 requests offered per arm). Flow conservation balances across all cells. CorrectRetention within 1 s is reported for sub-capacity (30/s) and over-capacity (400/s) arrival rates. The over-capacity column additionally reflects per-arm service-capacity differences (e.g., Naive has no journal or backpressure); cross-arm continuity comparisons are made at sub-capacity.

Condition	Carrier	Done	Rejected	Lost/Torn	CR@1s(Sub)	CR@1s(Over)
Static	none	43,036	16,964 bp	0 / 0	1.000	0.041
Protean replace	builtin	43,826	16,174 bp	0 / 0	0.999	0.036
Protean rollback	builtin	42,387	17,613 bp	0 / 0	0.996	0.025
Restart	none	38,448	16,225 bp	5,327 lost	0.998	0.056
Naive in-place	none	60,000	0	185 torn	1.000	0.994
Erlang-2Ver	none	50,722	9,278 bp	0 / 0	1.000	0.422
Protean soft revoke	builtin	16,550	34,590 rev†	0 / 0	0.500	0.018
Protean hard revoke	process	15,427	35,500 rev†	71 canc*	0.374	0.010
Protean hard revoke	builtin	16,317	34,709 rev†	45 canc*	0.499	0.016
Protean hard revoke	restricted	13,225	36,203 rev†	10 canc*	0.015	0.008
Protean effect revoke	builtin	4,422	55,463 rev†+bp	115 fenced‡	0.003	0.001

Over-capacity retention is bounded by the synchronously journaled admission path: the no-change Static arm retains 0.041, like the PROTEAN arms. Naive and Erlang-2Ver keep no journal or effect ledger, so their higher over-capacity retention buys no durability and is not comparable on that axis.
bp = backpressure rejected; rev = rejected by revocation (†plus 8.8k–9.0k queue backpressure).
*In-flight requests cancelled by hard revocation: process and restricted carriers reach quiescence via SIGKILL; on the non-preemptible builtin carrier the callable runs to completion and the commit barrier denies its outcome.
‡In-flight jobs fenced at the next effect boundary after revocation (115 blocked ledger rows, zero commits after the first block; last allowed commit $\leq$13 ms).

C.12 SCRIPTED POSITIVE-CONTROL ATTACKER FUNNEL (C2-PC)

To address underpowered injection-induced attempt rates in C2-ESC, C2-PC evaluates 82 goal-aligned escalation requests through the live pending–approval–realization pipeline (Table 23).

Archived reports: `claim2/c2pc-positive-control-20260920` (deny/rubber arms), `claim2/c2pc-llm-approver-20260920` (LLM approver arm).

C.13 BOUNDED-QUEUE CONTINUITY MATRIX (C3-Q)

The C3-Q experiment evaluates continuity and revocation in a 330-cell matrix: 11 condition–carrier arms (9 conditions; hard revoke evaluated on builtin, restricted, and process carriers, and the effect-boundary revoke arm on builtin) $\times$ 3 queue bounds (unbounded, 128, 32) $\times$ 2 offered arrival rates (sub-capacity 30/s, over-capacity 400/s) $\times$ 5 random seeds, offering 60,000 requests per arm (Table 24).

Archived report: `claim3/c3q-bounded-continuity-20260920` (330-cell matrix).

Table 25: C1-X carrier differential equivalence across all 49 representable AgentDojo tools (1,227 test cells × 4 execution legs) compared side-by-side with the frozen 20-tool subset. Clopper–Pearson 95% confidence intervals are reported in brackets.

Metric	49-Tool All	49-Tool CI95	20-Tool Subset	20-Tool CI95
Result agreement (EQ_{result})	0.978 (1,200/1,227)	[0.968, 0.985]	0.962 (483/502)	[0.942, 0.976]
State agreement (EQ_{state})	0.987 (1,211/1,227)	[0.979, 0.992]	0.970 (487/502)	[0.951, 0.982]
Error presence ($EQ_{error_presence}$)	0.990 (1,215/1,227)	[0.983, 0.994]	0.992 (498/502)	[0.980, 0.997]
Error class agreement (EQ_{error})	0.447 (548/1,227)	[0.419, 0.475]	0.371 (186/502)	[0.329, 0.414]
Effect agreement (EQ_{effect})†	0.174 (214/1,227)	[0.154, 0.197]	0.171 (86/502)	[0.141, 0.207]
Attributed mismatches:	Transport: 4,732; Nondeterminism: 180; IR gap: 75; **Kernel bugs: 0**			

†Effect ledgers differ structurally across transport legs by design; semantic side effects remain identical. Error class disagrees when carriers name the same failure differently (a Python exception class, an MCP error code, or an HTTP status); whether an error occurs agrees in 0.990 of cells. Planners therefore see the same failures under different names, which may explain part of the per-suite utility variation.

C.14 C3-Q Bounded-Revocation Invariants and Distributions

The C3-Q experiment formalizes and evaluates revocation semantics across preemptible (process, restricted) and non-preemptible (builtin) execution carriers. Soft, hard, and effect revoke are the *drain*, *kill*, and effect-boundary modes of §2.4:

- **Soft Revocation Invariant:** For any revocation stamped at epoch t_{rev}, every admission request arriving at $t > t_{rev}$ is unconditionally rejected. Across all 30 soft-revocation cells, 34,590 admissions were rejected, and post-revocation new-admission side effects were strictly zero (new_admission_effects $= 0$). In-flight invocations admitted prior to t_{rev} drain within their frozen authority envelope without Kernel-imposed timeout (builtin carrier drain delay: p50 17.9 ms, p95 63.2 ms, max 94.4 ms).
- **Hard Revocation on Preemptible Carriers:** When active cancellation is triggered, every in-flight process is sent SIGKILL. In all 60 hard-revocation cells on preemptible carriers, quiescence was strictly bounded (`quiescence_bounded_all` $=$ `true`), with zero failed cancellations (`cancel_failed` $= 0$) and zero post-quiescence effects. Quiescence delay distributions were:
  - Process carrier (30 cells): p50 99.4 ms, p95 151.5 ms, max 166.8 ms.
  - Restricted carrier (30 cells): p50 110.0 ms, p95 200.7 ms, max 228.6 ms.
  - Builtin carrier (30 cells, drain-by-completion): p50 81.7 ms, p95 106.6 ms, max 107.7 ms.
- **Effect-Boundary Revocation (builtin):** In-flight invocations are not torn; each subsequent effect request after revocation lands a blocked ledger row and the job fails. Across its 30 cells, 115 ledger rows were blocked and no commit followed a block; the last allowed commit came at most 13 ms after revocation.

All interrupted jobs under hard revocation are deterministically classified as committed, aborted, or unknown. *Archived report:* `claim3/c3q-bounded-revocation-20260921`.

C.15 C1-X 49-Tool Differential Expansion

To verify that carrier equivalence generalizes beyond the initial 20-tool subset, C1-X was expanded to evaluate all 49 representable AgentDojo tools across Python, MCP, and HTTP carriers (1,227 cells × 4 transport legs = 4,908 leg executions; Table 25).

Across 4,987 total recorded mismatches, all were conclusively attributed: 4,732 arose from transport serialization differences (e.g., JSON string encoding vs Python objects), 180 from native nondeterministic ISO timestamps in `create_file`, and 75 from IR schema gaps. Zero mismatches were caused by Kernel bugs. Re-aggregating the frozen 20-tool subset from the same run confirms exact identity (k/n match) with the archived report, verifying that equivalence is robust across the complete tool suite.

D REMOTE MCP LIFECYCLE END TO END (C3-R)

The MCP evidence elsewhere in the paper uses loopback servers (C1-X, C3-U) or protocol handshakes against public servers (C1-B). C3-R asks whether the lifecycle properties survive against *real remote* MCP servers that PROTEAN does not control, and it states which ones do not. *Archived report:* `claim3/c3r-remote-mcp-20260923`.

Servers and network. The client is a laptop-class workstation (Intel Core i7-13700H, 16 GB, Linux 6.6 under WSL2, Python 3.14). Two servers run on a cloud VM (aarch64, 2 vCPU) in a different region, on a different continent from the client. All server ports bind to loopback on the VM and are reached through SSH local port forwarding, so traffic crosses the WAN inside an SSH session rather than as direct TLS; measured latencies include the tunnel's overhead. (i) *Tracker*: an issue tracker written against the official MCP Python SDK (`mcp` 2.2.0, specification version 2025-06-18, stateless Streamable HTTP). An operator control route outside `/mcp` ships tool updates behind the client's back or injects a fault, and every request that reaches the app is appended to a JSONL log *before* dispatch. (ii) *Gitea*: the official `gitea-mcp` server (v1.7.0, 55 tools, SSE replies only) in front of a Gitea 1.24.6 instance. Ground truth never comes from a client. For the tracker it is the server's issue store and request log; for Gitea it is Gitea's REST API, queried on the VM so that its token never leaves that host. The tracker episodes also run on loopback (n=50 per cell) to separate protocol effects from WAN timing. Over the WAN, one `tools/list` round trip to `gitea-mcp` takes 1.68 s at the median (five samples at run start).

Definition pin. C3-R introduced one mechanism. A capability compiled from a discovered MCP tool may carry a digest, sha256 over the canonical JSON of the tool's `tools/list` definition fields (`name`, `title`, `description`, `inputSchema`, `outputSchema`, `annotations`). Before each call the `net.request` effect re-lists the server's tools under the same `net.connect` authority, *before* its ATTEMPT row, and raises `ContractViolation` unless the served definition hashes to the pin. A refused call therefore leaves no journal attempt and sends no `tools/call`. The check is an annotation on the effect node, not a branch on the capability's source type (the source-agnostic runtime test still passes). On the pinned path, SSE replies are decoded, and a JSON-RPC error or an `isError` result is a failure rather than output. Unpinned capabilities keep the earlier byte-identical program. The pin binds the *declared* contract only: a server that changes behavior behind an unchanged definition, or answers `tools/list` differently from `tools/call`, is outside what a client can observe.

Conditions and events. The native arms use the official SDK `ClientSession`. *Cached* lists tools once per session. *Re-list* lists before every call and skips a tool whose name disappeared; this is the strongest check the protocol offers without an out-of-band pin. *Retry* re-sends a timed-out call once. The PROTEAN arms install the discovered descriptor with or without the pin. A tracker episode makes six `issue_create` calls, and the event fires after call 3. E1a–E1d deploy a behavior-only change (issues silently become public), a description change, a new required `labels` field, or the tool's removal. E2 revokes the capability (drain mode) while call 3 is in flight. In E4 the server commits call 4 and then stalls past the client deadline, so the effect lands and the response is lost; the harness, acting as the caller, then retries with the same job and invocation identity, which PROTEAN must refuse, and reconciles through the pinned `issue_search` tool. Deadlines are 3 s on loopback and 10 s over the WAN, with stalls of 6 and 15 s.

Findings. *Declared drift is refused before it reaches the server.* Under the description, schema, and removal events, the pinned arm sent no post-event `tools/call` in any episode on either path. The cached SDK arm kept calling: the schema change and the removal fail at the server as tool errors, and the description change, which is the case a tool-poisoning update relies on, is followed silently. Re-listing catches only the removal. *Behavior-only drift is not detected by any arm, including the pin* (E1a): every post-event issue was filed publicly. This is the boundary stated in Limitation (4), now measured. *Revocation* (E2) let the in-flight call 3 complete under drain and denied calls 4–6 locally, and the server log shows no post-revocation request. *Lost responses* (E4): the enabled retry-on-timeout host policy duplicated the issue in every episode while reporting success; PROTEAN refused the same-identity retry (`Conflict`) in every episode, and reconciliation found exactly the one issue the server holds. *What the unpinned program does and does not report.*

Table 26: C3-R tracker episodes, judged at the server. For calls 4–6 (after the event): *Srv*, `tools/call` requests the server received; *Eff*, issues created; *FS*, calls reported successful with no issue created; *Dup*, duplicate issues per title. Denominators are $3n$ calls per cell, with n=50 on loopback and n=20 over the WAN.

		Loopback (n=50)				**WAN** (n=20)			
Event	**Arm**	Srv	Eff	FS	Dup	Srv	Eff	FS	Dup
E1a behavior only	SDK, cached list	150	150	0	0	60	60	0	0
	SDK, re-list	150	150	0	0	60	60	0	0
	Protean, unpinned	150	150	0	0	60	60	0	0
	Protean, pinned	150	150	0	0	60	60	0	0
E1b description	SDK, cached list	150	150	0	0	60	60	0	0
	SDK, re-list	150	150	0	0	60	60	0	0
	Protean, unpinned	150	150	0	0	60	60	0	0
	Protean, pinned	0	0	0	0	0	0	0	0
E1c schema	SDK, cached list	150	0	0	0	60	0	0	0
	SDK, re-list	150	0	0	0	60	0	0	0
	Protean, unpinned	150	0	150	0	60	0	60	0
	Protean, pinned	0	0	0	0	0	0	0	0
E1d removal	SDK, cached list	150	0	0	0	60	0	0	0
	SDK, re-list	0	0	0	0	0	0	0	0
	Protean, unpinned	150	0	150	0	60	0	60	0
	Protean, pinned	0	0	0	0	0	0	0	0
E2 revoke in flight	Protean, unpinned	0	0	0	0	0	0	0	0
	Protean, pinned	0	0	0	0	0	0	0	0
E4 lost response	SDK, cached list	150	150	0	0	60	60	0	0
	SDK, retry	200	200	0	50	80	80	0	20
	Protean, unpinned	150	150	0	0	60	60	0	0
	Protean, pinned	150	150	0	0	60	60	0	0

The unpinned PROTEAN program checks transport and JSON-RPC status and passes the tool result through unchanged, including an `isError` result. Its invocation status therefore means "delivered and answered," not "the tool succeeded." Under the schema and removal events every post-event call was recorded as completed although the server created nothing, whereas the SDK arms surface these as tool errors. No effect escaped authority, and none was duplicated, but a record of completion does not imply a server-side effect. Nor is this silent weakening in the sense of §3.1: the returned result, including its `isError` flag, and the server state are the same as under the SDK, and only the status classification differs. Separately, a genuine compatibility defect: the unpinned program sends a JSON-only `Accept` header, which `gitea-mcp` rejects (HTTP 400), so unpinned PROTEAN cannot call that server at all. The pinned path reads `isError` as a failure and accepts SSE; the unpinned program is left byte-identical so that earlier artifacts replay unchanged. The pin re-list costs one extra round trip per call: on loopback, in the E1a cell where no call is blocked, the p50 latency rises from 14.5 ms to 24.0 ms (the SDK client takes 2.7 ms), and over the WAN by roughly one `tools/list` round trip.

Gitea track. G1 is a create followed by a search. G4 sets the create deadline below the WAN round trip, so the response is lost naturally rather than by an injected fault. Whether the create lands before the client disconnects is left to the network, so the *Issues* column measures what actually happened, not a target rate. In G1 the SDK arm and the pinned arm each create and find the issue in 20/20 episodes. The pinned create takes 3.8 s at p50 against 1.5 s for the SDK, because it re-lists all 55 tools before each call; this is the pin's cost on a slow link. Unpinned PROTEAN fails in 20/20 (HTTP 400, see Findings). In G4 the SDK reported a timeout in all 20 episodes, yet 19 of those issues exist. With the enabled retry-on-timeout host policy, 19/20 episodes left two issues (39 in total), and every one of them still reported a timeout. The pinned arm's create landed in 13/20 episodes. Its same-identity retry was refused in 20/20, no episode holds two issues, and in 20/20 the pinned search result matched

Table 27: C3-R against the official `gitea-mcp` server and a real Gitea instance over the WAN, 20 episodes per row. *Issues* counts the issues Gitea holds with the episode's title. *Recon.* is the number of unknown-outcome episodes whose pinned `search_issues` reconciliation matches Gitea's count. p50 is the create-call latency in ms. The G4 deadline is 1.1 s for the SDK arms and 1.1 s plus the median `tools/list` round trip for the pinned arm, so that its deadline also expires after the call is sent.

Case	Arm	n	First-call status	Issues	Same-id retry	Recon.	p50
G1	SDK, cached list	20	ok 20	20	—	—	1484
G1	Protean, pinned	20	ok 20	20	—	—	3821
G1	Protean, unpinned	20	RuntimeFailure 20	0	—	—	1460
G4	SDK, cached list	20	timeout 20	19	—	—	1102
G4	SDK, retry	20	timeout 20	39	—	—	3452
G4	Protean, pinned	20	ResourceLimitExceeded 20	13	Conflict 20	20/20	2786

Gitea's own count (13 found exactly one, 7 found none), so each unknown outcome was resolved correctly in both directions.

Validity. The SSH tunnel dropped during an earlier attempt. The harness now records every reconnect, and any episode that overlaps a drop (from 3 s before it starts until 30 s after it ends, which covers keepalive detection lag), or that crashes in its plumbing, is discarded and rerun under a fresh tag. Discards are archived with their reason. The loopback run discarded none. Over the WAN the tunnel dropped three times during the Gitea run and in three separate periods during the tracker run, one an outage of six minutes; this discarded 3 Gitea and 20 tracker episodes (13 plumbing crashes during the outage and 7 overlaps). Separately, the official SDK client process itself exited abnormally in 3 of 23 WAN retry episodes (E4), with no tunnel drop nearby and after the server had already created the duplicate; all three had duplicated as well, so rerunning them does not flatter the SDK arm. We traced the exit to a race in the SDK's streamable-HTTP transport (`mcp` 2.2.0): after the read deadline the session sends `notifications/cancelled` and returns, but the POST task for the stalled call stays open, and when the server's late reply arrives after the session has closed, delivering it raises `anyio.BrokenResourceError`, which escapes the transport's task group. The race needs the episode to end near the end of the stall, which the WAN timing (10 s deadline, 15 s stall) permits and the loopback timing (3 s, 6 s) does not. Rerunning this episode 30 times over the WAN reproduced it 4 times, none near a tunnel drop; the other two non-zero exits were plumbing (a harness control request timing out, and a tunnel drop). No episode was left unrecovered. Harness plumbing (control route, listing, ground-truth queries) retries across drops; a measured call never does. Lost responses in E4 and G4 come from the stall or the client deadline, not from packet loss, so they do not depend on the tunnel. Lifecycle events are issued by the harness, not by an operator or a model, and the tracker is our own SDK server rather than a third-party deployment. The Gitea track tests a third-party server but only its create and search tools.

E τ^2-BENCH DESCRIPTIVE REPLICATION (C1-T)

The preregistered utility result (§3.1) uses AgentDojo only. C1-T asks whether routing a second benchmark's tool calls through the Kernel changes its official score. It is a *descriptive replication*. It was not preregistered, and with 20 pairs per domain we claim no non-inferiority margin. *Archived report:* `claim1/c1t-tau2-ni-20260924`.

Harness. We use τ^2-bench 1.0.1 (Barres et al., 2025; Sierra Research, 2025) (airline and retail domains, `base` split) with its official LLM agent, user simulator, orchestrator, and evaluator unchanged. The two arms differ only in the assistant's tool dispatch. In the native arm, the environment calls its toolkit directly. In the PROTEAN arm, each domain tool is installed and activated as a builtin capability, and each call goes through `Kernel.invoke`. Each capability declares one effect, `airline`/`retail.read` or `.write`, taken from τ^2's own `mutates_state` flag. The model principal's lease covers exactly these capabilities. The two resources are registered process-locally for this run; the shipped vocabulary is unchanged. The capability returns τ^2's own JSON serialization of

the tool result, so the tool message the agent sees is byte-identical to the native path. A tool exception is re-raised with its original message, so the agent sees the same `Error:` string. Every episode uses a fresh Kernel and a single host carrier; this is not a mixed-surface run.

Tasks, model, and fidelity checks. Per domain, we take the first 20 tasks whose gold actions include a state-mutating assistant tool; this rule was fixed before any episode ran. That gives airline tasks 7, 8, 11, 12, 14–25, 29, 30, 32, 33 and retail tasks 0–9, 11, 13–21. Each task is run once (seed 1, temperature 0) with `deepseek-v4-flash` (DeepSeek-V4.1-Flash, the planner used in C1-X) as the agent, the user simulator, and the judge for natural-language assertions. τ^2's default judge is `gpt-4.1`, so retail scores are not comparable to the public leaderboard. Both arms are judged identically. Fidelity is checked twice per episode. First, the official evaluator replays the trajectory natively in strict mode, which fails if any recorded mutating output differs. Second, the PROTEAN arm's live database hash must equal the hash of an independent native replay of the same messages.

Table 28: C1-T τ^2-bench paired episodes. *Reward* is τ^2's official task reward (mean over 20 pairs). *P/N/T*: PROTEAN wins, native wins, and ties. p is the exact two-sided McNemar test. *Replay*: episodes whose live DB hash equals an independent native replay (native/PROTEAN). *Kernel*: Kernel-completed invocations out of assistant tool calls in the PROTEAN arm. *Wr.*: `.write` effects committed. *Den.*: Kernel denials. *ms*: median of per-episode mean Kernel call latency, measured with four concurrent episodes in one process.

Domain	Pairs	Native	Protean	P/N/T	p	Replay	Kernel	Wr.	Den.	ms
Airline	20	0.75	0.75	3/3/14	1.00	20/20	181/182	35	0	74
Retail	20	0.95	0.95	1/1/18	1.00	20/20	155/155	24	0	37

Results. Mean reward is identical in both domains (Table 28), and the discordant pairs split evenly (3/3 and 1/1). Temperature 0 does not make the provider deterministic, so we read the discordant pairs as run-to-run variance. Every episode in both arms passes strict replay, and all 80 live hashes equal their native replays. All 80 episodes end with the user simulator stopping the conversation. PROTEAN committed 59 write effects with no denials. No denials is the expected outcome, not evidence of enforcement: the lease covers every domain tool, so no call was ever out of grant. Authority is tested in C2.

Accounting. One airline call (task 25) has no Kernel completion record. The builtin runtime passes a tool's own exception through unwrapped, and the router used for the main run logged only `KernelErrors`. A PROTEAN-only rerun of task 25 over three seeds reproduced the case in two of them. There, `book_reservation` raises τ^2's `ValueError` ("Payment amount does not add up"), the agent receives the same `Error:` string as on the native path, and the episode still earns reward 1 with an equal replay hash. The router now logs such calls as tool errors. Retail's first attempt lost 12 episodes (6 tasks, both arms) at evaluation, when the default judge had no credential. The run was repeated after the judge was pointed at DeepSeek; the superseded records are kept in the archive. Latency includes contention between concurrent episodes and is not comparable to the isolated decomposition in C3-O.

F EXTENDED RELATED-WORK COMPARISON

Tool use and mutable environments. Toolformer, ToolLLM, and Gorilla study tool selection and API invocation (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024a). LATM, CREATOR, ToolMaker, and Voyager study tool creation and reuse (Cai et al., 2024; Qian et al., 2023; Wölflein et al., 2025; Wang et al., 2024a). SWE-bench evaluates repository-level software-engineering tasks (Jimenez et al., 2024), while OSWorld, AppWorld, WebArena, WorkArena, and AgentBench evaluate agents in interactive, stateful software and application environments (Xie et al., 2024; Trivedi et al., 2024; Zhou et al., 2024; Drouin et al., 2024; Liu et al., 2024). Coordination, memory, and scheduling frameworks (Wu et al., 2024; Hong et al., 2024; Gao et al., 2024; Packer et al., 2023; Mei et al., 2025; Zhou et al., 2023) can act as callers of an execution runtime, while MCP supplies a shared tool interface (Model Context Protocol contributors, 2025). These settings motivate managing the permissions and implementation identities of the operations those callers use.

Live updates and execution continuity. Dynamic software updating (DSU) studies when running programs can safely transition between versions (Hicks & Nettles, 2005; Neamtiu et al., 2008). Erlang's two-version code loading provides an influential live-update mechanism for concurrent message-passing actors (Armstrong, 2007). Our safety question differs in scope: an autonomous planner may invoke capabilities whose internal implementation, exposed schema, and delegated grants mutate asynchronously during an episode. Version pinning must therefore be coupled to monotonic authority epochs and receipt-reconciled effect attribution. The comparison in Table 14 tests these properties at fixed transition barriers; its counts demonstrate reachable violations under naive updating and baseline frameworks.

Protection and agent security. Least privilege and capability systems motivate explicit authority and controlled delegation (Saltzer & Schroeder, 1975; Lampson, 1974; Levy, 1984; Miller et al., 2003; Shapiro et al., 1999). Capsicum applies capability restrictions to UNIX processes (Watson et al., 2010), while seL4 verifies a microkernel implementation (Klein et al., 2009). HiStar and Flume track information flow through OS abstractions (Zeldovich et al., 2006; Krohn et al., 2007); PROTEAN's scope-based egress checks provide no separate information-flow policy plane.

AgentDojo studies prompt injection; ToolSandbox and τ-bench evaluate stateful tool use and policy-following; ToolEmu and the Agent Security Bench benchmark agent risk end-to-end; and InjecAgent measures indirect-injection susceptibility (Debenedetti et al., 2024; Lu et al., 2025; Yao et al., 2025; Ruan et al., 2024; Zhang et al., 2025; Zhan et al., 2024). Indirect prompt injection exposes the risk of treating retrieved text as authority (Greshake et al., 2023; Liu et al., 2023). Spotlighting marks untrusted input for the model (Hines et al., 2024); StruQ structurally separates instructions from data (Chen et al., 2025); CaMeL separates control/data flow and applies value-level policies at tool calls (Debenedetti et al., 2026). Our enforcement checks the supplied invocation envelope and records the implementation and effects associated with that decision.

Runtime enforcement. AgentSpec expresses action rules in a DSL (Wang et al., 2026); Progent applies revisable tool-name and argument policies (Shi et al., 2025); GoEX studies sandboxing and undo (Patil et al., 2024b). HCP specifies execution-layer invariants for MCP-style calls (Liu, 2026), and Harnessing Embodied Agents evaluates external governance in simulation (Qin et al., 2026). Our C2-K tests compare name/argument checks with configured effect mediation: the former misses 60 undeclared effects and the latter blocks all 60. The unset-policy behavior observed in our Progent port is specific to that configuration.

Transactions and durable runtimes. Classical sagas and ARIES-style write-ahead recovery supply the compensation and logging vocabulary (Garcia-Molina & Salem, 1987; Mohan et al., 1992). Atomix orders task-level settlement using per-resource progress frontiers (Mohammadi et al., 2026). Cordon stages mutations and external effects, then validates lineage and authority before release (Chen et al., 2026). Their transaction boundaries govern when work may settle. PROTEAN's invocation snapshots govern which implementation and authority execute; interrupted effects follow their replay and receipt contracts.

Agent libOS combines process identity, task ceilings, typed capabilities, and approval with durable evidence (Zhang, 2026). Its prepare–dispatch–settle protocol retains ambiguous outcomes, and its checkpoint restoration preserves authority restrictions. Those concerns converge with PROTEAN's recovery and rollback design. Agent libOS additionally provides information-flow labels and sink clearance. Our principal comparison axes are per-invocation revision pinning during live updates and executable admission across source formats (Table 29).

Table 29 compares the published semantics of PROTEAN against contemporary runtimes for tool-using agents: Agent libOS (Zhang, 2026), Cordon (Chen et al., 2026), and HCP (Liu, 2026). Comparisons are grounded entirely in published specifications and arXiv section pins, classifying each property under a three-valued verdict: **specified** (explicitly defined in cited sections), **not specified** (no statement found in the cited sections), or **semantics differ** (addressed under an alternative abstraction).

Table 30 places the quantities each system reports about itself next to ours. We did not run the external systems; every external number is quoted from the cited version. Workloads, models, denominators, and harnesses differ across columns, so the table shows what each evaluation measures, not a ranking. None of the three evaluates on AgentDojo, and none reports a live-update or revocation-under-load measurement.

Table 29: Published-semantics comparison across contemporary agent runtimes. Citations reference verified arXiv preprint sections: Agent libOS (Zhang, 2026) (arXiv:2606.03895 v3), Cordon (Chen et al., 2026) (arXiv:2606.17573 v1), and HCP (Liu, 2026) (arXiv:2606.29073 v1). Verdicts: **specified** (explicitly addressed), **not specified** (omitted from specification), or **semantics differ** (different abstraction layer). The simulated process-scoped reference arm is r6 in Table 17; no code was cloned or executed for external systems.

Semantic dimension	**Agent libOS (v3)**	**Cordon (v1)**	**HCP (v1)**	**PROTEAN**
(a) Authority binding granularity	**semantics differ**: process-scoped capability + task ceiling (§3.1)	**semantics differ**: transaction scope authority A (§3.1, §3.5)	**specified**: grant matching per invocation (§5)	**specified**: per-invocation lease token $L(I)$ bound to $\Sigma(I)$
(b) Implementation identity frozen per call	**not specified**: process binding + source hash; replace in-flight omitted (§5.2)	**not specified**: tool versioning omitted; versions workspace state (§3.2, §4.4)	**not specified**: manifest identifier only; no versioning (§5)	**specified**: decoupled generation/epoch pinning via immutable snapshot $\Sigma(I)$
(c) Rollback carrying old authority (CE2)	**specified**: checkpoint restore cannot mint authority (§2.4, §5.3)	**not specified**: abort cancels staged effects; grant rollback omitted (§3.4, §4.6)	**not specified**: no rollback concept; grant check is invocation-time (§5, §9)	**specified**: monotonic epoch rejects revoked grants upon code rollback
(d) Revocation semantics on in-flight calls	**not specified**: epoch fencing on crash; in-flight revoke drain omitted (§4.3, §5.4)	**specified**: outbox cancel/compensate for tx; capability revoke omitted (§4.5)	**not specified**: revoked grant denies at gate; in-flight drain omitted (§5, §9)	**specified**: soft revoke drains in-flight; hard revoke bounds quiescence
(e) Crash effect classification	**specified**: prepare–dispatch–settle classifies unknown (§4.2, §5.4)	**specified**: idempotency key + receipt review on crash (§4.5, §4.6)	**not specified**: in-memory audit; durable recovery out of scope (§9)	**specified**: two-phase journal classifies unknown; rejects retry
(f) Heterogeneous admission across sources	**specified**: conjunctive admission across tools/JIT/MCP syscalls (§3.1, §5.2)	**not specified**: single-point interposition; unified IR omitted (§4.2, §5)	**specified**: unified broker grant matching for MCP calls (§5)	**specified**: fail-closed admission across OpenAPI, MCP, and native

G INVARIANT STATEMENTS AND PROOF SKETCHES

The invariants below are properties of the Kernel state-machine specification. The sketches argue them from that specification under Assumption A1; they are neither mechanized nor implementation-level proofs. Whether the implementation conforms is established empirically, by the negative controls, interleaving suites, and crash tests of §3. The strength of I2 is exactly that of A1: on trusted-host carriers, confinement follows from trust rather than mechanism.

The invariants are:

- **I1 (Active Implies Admitted):** A candidate enters the active space only after contract verification and admission arbitration.
- **I2 (Conditional Authority Confinement):** For every invocation I, RealizedEffects$(I) \subseteq A(I)$ under the mediation assumption below.
- **I3 (Revocation Freshness):** After revocation at epoch k, subsequent admissions cannot resolve grants stamped with epoch $j \leq k$.
- **I4 (Generation Consistency):** Each invocation executes against one immutable generation.
- **I5 (Atomic Publication):** A per-capability binding mutation exposes either the prior or next valid generation. Multi-member imports stage all members and publish them atomically in a single manifest transaction.
- **I6 (Rollback Safety):** Historical bindings are restored only with authority checked against the current epoch.
- **I7 (Effect Integrity):** Interrupted non-idempotent effects without a reconcilable receipt are classified as unknown, not blindly replayed.

Table 30: Self-reported quantitative results, quoted from Agent libOS (v3), Cordon (v1), and HCP (v1) with the table or section of the cited paper (external "Tab."/"Fig."/§ numbers refer to that paper); PROTEAN values are from this paper. Numbers in one row come from different workloads and are not directly comparable. "—": not reported.

Quantity	Agent libOS (v3)	Cordon (v1)	HCP (v1)	PROTEAN
Benign utility	33/33 deterministic tasks (Tab. 3); 12/12 canonical real-model runs, model unnamed (Tab. 6); 13/15 fully correct in both paired Skill arms (Tab. 7)	τ-bench subset 87.5% $\rightarrow$ 90.0%, DeepSeek-V4-Pro, n not stated (Tab. 6)	— (no LLM; 18/18 MCP coverage items, Tab. 8)	AgentDojo 253 pairs, 93.7% $\rightarrow$ 95.3%, LCB95 -0.8 pp; τ^2-bench 40 pairs, reward unchanged (Tab. 28)
Authority / attack	0/119 unauthorized effects vs. 20/29 direct wrapper (Tab. 3)	45/45 risky effects intercepted before commit; plain commits 45/45 (Fig. 5)	10/10 modeled attacks blocked vs. 6/10 succeed under mitigation (Tab. 4)	SI $= 0$, ROE $= 0$ over 1,146 out-of-ceiling requests; effect mediation blocks 60/60 undeclared effects
Overhead	— (timings declared diagnostic, §7.3)	Transaction control 22.2–23.4% of task time; rollback 4.17 ms median (Fig. 9, Tab. 5)	0.002–0.097 ms p50, in-memory, no baseline (Tab. 6)	12.9 ms p50 carrier-free Kernel (8.3 ms `fsync`); 23.0 ms process carrier (Tab. 22)
Crash / recovery	6/6 barrier cases; 1,000/1,001/1,000 seeded pending or unreconciled rows reconciled (Tab. 5)	15/15 resumes, no crash injection (Tab. 5)	— (in-memory runtime)	16/16 crash-replay cases; same-identity retry refused 20/20 against remote Gitea (Tab. 27)
Live update / revocation	—	—	—	0 lost of 60,000; 0 events in six safety categories (Tab. 4)

- **I8 (Execution Continuity):** Lifecycle transitions preserve unrelated and already admitted in-flight execution, subject to the drain limitation.
- **I9 (Substrate Confinement):** Mechanically confined subprocesses cannot reach effects beyond the projected envelope.

Assumption A1 (Mediation Invariants). The Kernel mediates all operations passing through its ABI. The three clauses differ in what guarantees them, how they are measured, and how the trusted surface can be shrunk:

1. Host-registered capability bodies and interpreters belong to the TCB and correctly report their effects through the Kernel effect-ledger interface. *Guarantee:* trust, not mechanism—in-process code can bypass the declared-effect API. *Measured by:* the C2-K arms (name/argument checks miss 60/60 undeclared effects; effect mediation blocks 60/60) and the C2-M mediation-coverage matrix (Appendix K), which quantifies this clause directly: 63/70 host-side bypass variants escape unmediated. *Reduction:* migrating effectful builtins into unprivileged process carriers removes most of this clause (1,776 LOC; Table 32).
2. Subprocess carriers (such as Bubblewrap) enforce filesystem, network, and process isolation strictly within the bounds specified by their launch arguments. *Guarantee:* OS mechanism (private mount namespaces, network unsharing, dropped privileges). *Measured by:* the C2-N sandbox family (defense-off 16/20 escapes; defense-on 0) and C2-M (70/70 variants intercepted or never realized on the sandboxed carrier).
3. Remote service endpoints (HTTP, MCP) execute external effects outside the local host, but all communication with those endpoints is routed through the Kernel's mediated channels. *Guarantee:* channel routing; the Kernel observes local egress (endpoint, method, credentials, parameters), not server-internal mutations. *Measured by:* C2-K egress checks and the C3-Z/C3-U sink reconciliations. *Reduction:* receipt-aware endpoints (externally reconciled effect evidence, exercised by the C3-Z/C3-U sink reconciliations) shrink the unobserved region; a cryptographically signed receipt protocol is future work.

Proof Sketches. Under Assumption A1, the invariants hold as specification-level consequences of the Kernel state-machine definitions:

- *I1 (Active Implies Admitted):* Follows directly from the capability-state transition rule: active bindings are modified only by the `admit` transition, which checks syntactic validity, schema conformance, and staging probes.
- *I2 (Conditional Authority Confinement):* At invocation time, the Kernel computes $A(I) = \text{ResolveAuthority}(P, C, e)$, admits only if a live lease of P covers all of $A(I)$ (leases lie within $\text{Ceiling}(P)$ unless explicitly approved), and stamps $A(I)$ onto the snapshot $\Sigma(I)$. Under A1, carrier enforcement guarantees that realized effects do not exceed $A(I)$. Implementation-level conformance is verified empirically via the C2-N negative-control suite and the C2-M mediation-coverage matrix (§C.7).
- *I3 (Revocation Freshness):* Revocation advances the epoch counter $e \leftarrow e + 1$ and purges matching grants from the state-side lease table. Any subsequent `admit` checking epoch validity strictly rejects grants stamped with $j \leq e_{\text{revoked}}$.
- *I4 (Generation Consistency):* The invocation snapshot $\Sigma(I)$ contains an immutable reference to generation g. All subsequent fetches during that invocation resolve against the content-addressed store using the snapshot's digest, unaffected by concurrent mutation of the active pointer.
- *I5 (Atomic Publication):* The active binding pointer in the state store is updated in a single SQLite transaction with `fsync`. Readers observe either the old or the new pointer.
- *I6 (Rollback Safety):* The `rollback` operation restores code revision r_{target} without advancing the authority epoch. Revocation monotonically advances e and marks revoked grants ineligible, so the restored binding admits only under grants valid at the current epoch; leases superseded by a prior revocation remain invalid.
- *I7 (Effect Integrity):* Follows from the two-phase logging protocol: before any physical mutation, an `attempt` record is fsynced. On crash recovery, any attempt without a corresponding `commit` is marked `unknown`, and duplicate submissions with the same job identity are rejected with `Conflict`.
- *I8 (Execution Continuity):* Active invocations hold reference counts on their pinned generations. Replacement updates only the active pointer for future admissions; existing calls run to completion using their pinned snapshots.
- *I9 (Substrate Confinement):* Verified by sandbox configuration construction: private mount namespaces, network unsharing, and dropped privileges prevent host-level escapes.

H Detailed Capability Lifecycle and State Transitions

The capability lifecycle is governed by a state machine transitioning across three discrete spaces:

1. **Registration and Staging Space:** A raw operation candidate undergoes syntactic extraction, schema normalization, and contract validation.
2. **Active Capability Space:** Following admission, the capability is stamped with generation g_0 and authority epoch e_0. Transitions in this space include:
   - `replace`(C, r_{new}): Derives $g_{\text{new}} \leftarrow g_{\text{old}} + 1$, compiles the new artifact, and commits the updated binding pointer atomically.
   - `rollback`$(C, r_{\text{historical}})$: Re-points the active binding to a non-revoked historical revision without changing the epoch e; authority freshness stays epoch-scoped, so revoked grants are not resurrected.
   - `revoke`(C): Purges active lease tokens and increments epoch $e \leftarrow e + 1$.
3. **Invocation Execution Space:** Each individual invocation transitions through four sequential execution states:

$$\mathcal{S}_{\text{inv}} = \{\texttt{Admitted}, \texttt{Executing}, \texttt{Attempted}, \texttt{Terminal}\}$$

1. **Admission and Snapshot Pinning:** The Kernel resolves the active binding, derives $A(I) = \text{ResolveAuthority}(P, C, e)$, checks that a live lease covers it, assigns a random UUID4 invocation identifier, and persists an immutable snapshot $\Sigma(I) = \langle \text{CapID}, r, g, e, A(I) \rangle$.

2. **Substrate Dispatch:** The invocation dispatches to the carrier matching $\Sigma(I)$. Carrier execution is confined to the declared resource envelope $A(I)$.
3. **Two-Phase Effect Journaling:** Before any observable external side effect is executed, the runtime records an explicit `attempt` row in the persistent SQLite/WAL journal. Following confirmed completion, the runtime journals a corresponding `commit` entry.
4. **Termination and Receipt Reconciliation:** The invocation completes with terminal status (`Success`, `UserError`, or `SystemError`). The Kernel releases the active lease token, updates invocation metrics, and closes the drain reference counter.

I C1-X Polyglot Equivalence and Discordance Analysis

In Section 3.1, Phase B of C1-X evaluated mixed-surface non-inferiority across 255 planned task pairs, completing all 253 valid pairs (with two single-arm infrastructure exclusions due to remote model provider timeouts: `model:ModelCallError`). Here we present the complete discordance breakdown across the four AgentDojo task suites.

Per-Suite Task Discordance. Table 31 details the task outcomes, discordance counts, and invocation distributions across suites. Overall, Protean-Mixed recorded 9 wins, 5 losses, and 239 ties against native execution, recording an observed empirical delta of $\hat{\Delta} = +0.0158$. The primary bound is a suite-stratified percentile cluster bootstrap over the 85 (suite, task) units (20,000 resamples, fixed seed): $\Delta_{\text{LCB95}} = -0.008$ one-sided, 95% CI $[-0.016, +0.051]$, clearing the prespecified margin of -0.10 and every tighter margin down to -0.02. Method sensitivities: cluster-robust (CR1) -0.018; independence-assuming paired binomial reference -0.027. The bound remains ≥ -0.016 under all four feasible completions of the two infrastructure-excluded pairs and is -0.004 under the alternative first-write retry rule.

Table 31: C1-X Phase B full population breakdown across AgentDojo suites (253 valid pairs, DeepSeek-V4.1-Flash). Discordant pairs are classified as Protean wins (+) or Native wins (−).

Suite	Planned	Valid	Wins (+)	Wins (−)	Ties	$\hat{\Delta}$
Banking	48	48	0	1	47	−0.0208
Slack	63	62	1	0	61	+0.0161
Travel	54	54	5	3	46	+0.0370
Workspace	90	89	3	1	85	+0.0225
Total Population	**255**	**253**	**9**	**5**	**239**	+0.0158

Discordance Distribution. Of 14 discordant pairs, eight are in travel, four in workspace, and one each in banking and Slack. These task outcomes do not identify a causal mechanism: they cannot by themselves attribute a win to schema normalization or a loss to transport latency. Phase A measures carrier agreement separately.

Per-Source Kind Call Statistics. The report records 1,405 capability-call attempts across the three bindings:

- `SRC-HTTP`: 397 calls attempted, 379 succeeded (95.47%), 18 returned errors (4.53%), 0 authority denials.
- `SRC-MCP`: 458 calls attempted, 444 succeeded (96.94%), 14 returned typed domain errors (3.06%), 0 authority denials.
- `SRC-PY`: 550 calls attempted, 529 succeeded (96.18%), 21 returned typed domain errors (3.82%), 0 authority denials.

Across these calls, 1,352 succeeded and 53 returned errors; none was denied by authority checks under the supplied policies.

J C3-H CRASH-REPLAY SOAK AND LEDGER STATE TRANSITIONS

The C3-H filesystem soak passes 16/16 cases (2 families $\times$ 2 crash cuts $\times$ 4 iterations), covering recovery classification, idempotent replay, and same-identity retry rejection.

Two-Phase Logical Effect Protocol. The tested effect path records an attempt before the physical mutation and a commit after confirmation:

$$\texttt{Start} \xrightarrow{\text{Pre-effect}} \texttt{Attempt} \xrightarrow{\text{Physical Mutation}} \texttt{Execute} \xrightarrow{\text{Post-effect}} \texttt{Commit}$$

Every side-effecting invocation logs a persistent `attempt` row indexed by $\langle$`effect_id`, `stage` = `attempt`, `job_id`, `invocation_id`$\rangle$ via an atomic synchronous disk write (`fsync`) before executing the underlying physical operation. Upon successful completion of the external mutation, a corresponding `commit` row is written.

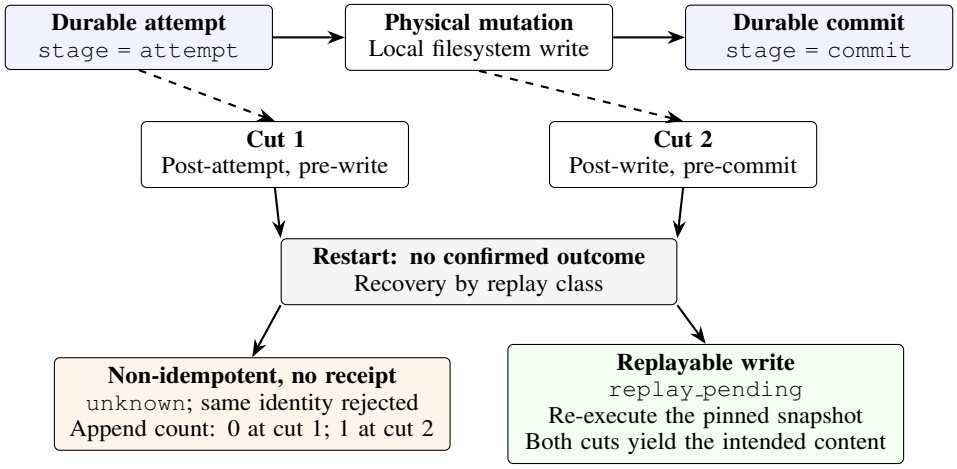


Figure 2: C3-H recovery at both crash cuts. Each cut exercises both replay classes; the crash location does not select the class. Unknown outcomes can mean zero or one physical effect, while idempotent replay may execute more than once.

State Transition and Recovery Outcomes. As illustrated in Figure 2, the harness injects process crashes at two critical boundaries:

- **Cut 1: `after_attempt`** (crash immediately following journaled attempt, before physical effect):
  - *Replayable Write Family*: The journal retains an orphan `attempt` row. Upon Kernel restart, recovery inspects the missing commit, marks the invocation `replay_pending`, and allows a resubmission carrying the identical invocation and effect identity to resume the attempt row and execute to success.
  - *Side-Effecting Append Family*: The journal contains an unconfirmed attempt. Because external side effects cannot be safely re-executed without risk of duplication, the Kernel classifies the outcome as `unknown`. When a resubmission occurs, the Kernel raises `Conflict`, refusing duplicate execution. The external counter verifies that exactly 0 lines were appended.
- **Cut 2: `after_write`** (crash immediately following physical write, before journaled commit):
  - *Replayable Write Family*: The physical mutation succeeded on disk, but the commit record is missing. Recovery classifies the operation as `replay_pending`. Resubmission safely re-applies the idempotent write, commits the journal, and returns success.
  - *Side-Effecting Append Family*: The physical mutation incremented the external state, but the commit record was not flushed before crash. Recovery classifies the interrupted invocation as `unknown`. Upon same-identity resubmission, the Kernel raises `Conflict`. The observed external counter remains at one line.

All 16 cases pass their configured checks on the local filesystem path. A fresh job identity is admitted as separate work. *Archived report:* `claim3/c3h-crash-replay-20260919`.

K THREAT MODEL, ATTACKER CAPABILITIES, AND TCB BOUNDARY ANALYSIS

To ground the security guarantees of Claim 2, we define the threat model, attacker capabilities, and TCB boundary assumptions governing PROTEAN.

Principal Trust Hierarchy. PROTEAN establishes a strict three-tier trust hierarchy:

1. **Trusted Runtime Kernel (TCB):** Formed by the in-process authority arbiter, transaction journal, monotonic epoch engine, and substrate managers. The Kernel is assumed to be correctly implemented and uncompromised.
2. **Untrusted Foundation Model Planner (P):** The upstream language model planner is treated as an untrusted principal. The planner may suffer from alignment failures, prompt-injection attacks, goal misgeneralization, or adversarial manipulation (Greshake et al., 2023; Liu et al., 2023; Zou et al., 2023); its requests therefore cannot be assumed to remain within the authority granted for the task.
3. **Governed Capabilities:** External tools and services vary in trustworthiness. Builtin host callables are trusted components of the host runtime; sandboxed Python substrates are mechanically confined via Linux namespaces (Bubblewrap); remote HTTP and MCP endpoints are untrusted external entities mediated exclusively at the local network boundary.

Attacker Capabilities and Vectors. We evaluate PROTEAN against four distinct adversarial threat vectors:

- **V1: Ambient Escalation and Token Forgery.** An attacker invokes tools without a valid lease, presents expired or revoked credentials, guesses lease identifiers, or reuses leases across capabilities. C2-I rejects 21/21 probes through state-side lease, binding, and epoch checks. UUID4 identifiers supply random bearer credentials; transport and credential storage remain deployment responsibilities.
- **V2: Substrate Escape and Host Compromise.** Malicious capability code attempts to escape execution confinement via directory traversal (`../../etc/passwd`), environment secret exfiltration (`AWS_SECRET_KEY`), or unauthorized subprocess execution (`/bin/sh`). In C2-B, the Bubblewrap-confined substrate intercepted 17/17 escape probes using dedicated mount namespaces and network isolation.
- **V3: Indirect Prompt Injection.** Untrusted web content or user-supplied tool payloads inject adversarial instructions commanding the agent to execute unauthorized operations (e.g., exfiltrating files or transferring funds). In C2-ESC, across the 80 repeated-cohort injection episodes (250 pooled, Table 11), the Kernel prevented silent issuance (SI $= 0$) and out-of-envelope side effects (ROE $= 0$) by holding all requests exceeding the task ceiling in a pending approval state.
- **V4: Temporal and Rollback Exploits.** An attacker attempts to exploit lifecycle transitions by issuing invocations during live updates, executing against revoked tools during drain, or triggering rollbacks to resurrect historical permissions. PROTEAN defeats these exploits by decoupling `provider_generation` from `authority_epoch`, ensuring that rollbacks restore historical code while strictly re-validating authority against the current monotonic epoch.

TCB Boundary and Mediation Boundaries (Assumption A1). Under Assumption A1, Bubblewrap carriers use OS-level process and namespace isolation. Host-registered callables and interpreters belong to the TCB and honor the effect API. HTTP/MCP mediation covers local egress, parameters, and credentials; remote internal effects remain outside that boundary. An independent test sink can detect discrepancies between its observed requests and local records, while effects outside the sink's visibility require another observer. The C2-M protocol measures these boundaries directly (§C.7, Table 19): on the trusted host executor 63/70 bypass variants escape (aggregate MCR $= 0.10$), while the sandboxed process carrier mechanically intercepts all 70 (MCR $= 1.00$, CP95 $[0.95, 1.00]$). In particular, the interpreter case marks variants not expressible under its vocabulary rather than silently counting them as intercepted.

Table 32: TCB quantification and module trust classification across the PROTEAN implementation (12,804 total code LOC). LOC excludes blanks and comments. Non-TCB modules are verified fail-closed at runtime admission.

Component / Module	LOC	Trust	Coverage	Violation consequence	Can exit TCB? / Cost
Kernel core (admission, lease, epoch)	2,829	Yes	F1, F2, F3, F6	Full mediation bypass	No (mediation boundary)
Persistence / journal (SQLite WAL)	1,581	Yes	F6, C3-H	Provenance / recovery loss	No (trusted audit store)
Host builtin executors (effects, fs)	1,429	Yes	F4, F6	Undeclared mutation	Yes (route to process carrier)
Interpreter executor (restricted-Py)	347	Yes	F5, C3-Q	In-process escape	Partially (bwrap isolate)
Sandbox manager (Bubblewrap argv)	593	Yes	F5	Host filesystem escape	No (sandbox configs)
MCP / HTTP client (egress)	145	Yes	F5, C3-Z	Out-of-scope egress	Partially (proxy delegate)
Total Trusted Computing Base	**6,924**	**54.1% of codebase; 0 third-party Python dependencies**			
Frontends, compile, IR, contracts	4,749	No	C1, F3	Rejected at admission	Already outside (fail-closed)
Bundled capability payloads	737	No	F4, F6	Lease envelope confined	Already outside (payload code)
Model-facing gateway	394	No	—	Untrusted caller boundary	Already outside
Total Implementation Codebase	**12,804**	**100.0% of codebase accounted for**			

TCB Quantification and Trust Classification. To evaluate the attack surface of the trusted computing base, Table 32 classifies all modules across the implementation. Code lines of code (LOC) are computed excluding blank lines and comment-only lines. Overall, 6,924 out of 12,804 code LOC (54.1%) reside within the TCB. The entire runtime has zero third-party Python dependencies (relying solely on the standard library), with external runtime dependencies limited to the standard Bubblewrap binary and SQLite.

TCB Reduction Path. The largest reducible portion of the TCB consists of the host builtin executors and interpreter executor (1,776 LOC total). The builtin executors are responsible for the non-preemptible drain behavior observed during revocation; the restricted interpreter was preemptible in C3-Q. Moving these executors out of the TCB would entail routing all builtin operations through isolated Bubblewrap child processes with projected filesystem envelopes. The performance trade-off of this migration is directly quantified by our C3-O decomposition: invoking operations via process carriers adds process launch and inter-process serialization overhead, while yielding a strictly minimal TCB of 5,148 LOC (40.2% of codebase). *Archived report:* `tcb/tcb-loc-20260921`.