Does Hand-Designed Trajectory State Improve Pre-Execution Control?

Anonymous authors
Paper under double-blind review

Abstract

We test whether accumulating risk across an agent's trajectory improves prevention over evaluating each action independently. Across separately analyzed live model studies and released AgentDojo traces, the evaluated hand-designed state establishes no prevention advantage over a stateless per-action rule. Replaying frozen calibration grids raises trajectory prevention to the stateless level but not above it. The live attack studies share seven authored scenarios, and we do not evaluate learned sequence policies. A diagnostic identifies why one apparent observability failure occurred: the controller discarded a tool-name signal already available before execution. Separately, removing returned tool content causes model evaluators to misidentify injection origin in 33 of 40 blinded packets. Within these evaluations, accumulated risk does not improve prevention, whereas retained content supports incident reconstruction. The resulting design lesson is to preserve decision-relevant information before compressing it into a risk state, and to evaluate prevention separately from reconstruction.

1 Introduction

Long-horizon agents create a mismatch between the unit at which systems are controlled and the unit at which incidents must be explained. A tool gateway sees a command; an investigator later needs to know whether it was authorized, whether an external instruction entered through returned content, what effect occurred, and whether a control layer could have intervened earlier (Chen et al., 2026).

Our project began with a stronger hypothesis: stateful trajectory risk would improve real-time prevention relative to stateless action filtering. Controlled matched-pair experiments supported it; the evaluation on real recorded sessions and live model traces did not. On real attacks, trajectory state did not outperform a per-action rule, and both controllers failed before execution when destructive effects were hidden behind an opaque tool call.

The evidence is larger than any one denominator. It includes 99 original live runs across DeepSeek V4 Flash, DeepSeek V4 Pro, and Kimi K2.6; separate 220-run Qwen and NVIDIA Nemotron 3.5 Lightning arms; 400 released AgentDojo traces; 860 calls from legitimate coding sessions; and 40 blinded incident packets. The studies are analyzed separately and never pooled.

Three measurements locate the negative result. First, the default trajectory profile is under-sensitive, but calibration does not reveal a stateful advantage. The 80 entries are controller configurations, not agent runs: each is replayed on all traces from the 99-run cohort and the 220-run NVIDIA Nemotron 3.5 Lightning cohort. At maximum prevention, trajectory and stateless control stop the same runs.

Second, an oracle handed the true effect record before execution prevents every harmful run. Because harm is defined by those effects, this is a near-definitional input bound rather than a deployable result. Third, a retrospective audit corrects our initial diagnosis of the opaque calls: the boundary retained the exact tool name, but our feature projection discarded it. The observed failure therefore identifies an encoding and policy limitation, not a demonstrated collision in the complete pre-dispatch view.

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.

The corrected result identifies information preservation, not memory alone, as the key design question. Some boundaries expose a resource before dispatch even when they hide the operation. In seven authored mechanism cases, scoped delegation blocks every harmful indirection when its target is an argument, but only five when the target is hidden in a body. These cases demonstrate a mechanism, not tool-registry prevalence. More generally, state cannot separate executions that truly remain identical under the complete information available at the intervention point.

We therefore introduce Trajectory Control, an identity-bound evidence architecture at the agent-tool boundary that preserves the authority, declared action, returned content, observed effect, and intervention history needed for later reconstruction.

This paper makes three contributions:

1. **A calibration-robust negative evaluation.** Across frozen live traces, AgentDojo replay, and separately analyzed model configurations, the evaluated trajectory-risk state does not establish an advantage over a per-action rule. Eighty pre-existing controller configurations are replayed over the complete three-model live cohort and the NVIDIA Nemotron 3.5 Lightning cohort, testing sensitivity to threshold selection within the evaluated mechanism.
2. **An empirical encoding and boundary diagnosis.** An elementary collision lemma states what follows when harmful and benign executions truly have the same pre-dispatch view. The live beta audit instead finds projection loss: the boundary retained a tool-name signal that our controller discarded. Authored collisions, oracle inputs, and effect preview then isolate when a genuine observability limit applies.
3. **A blinded evidence-ablation finding.** Complete records support reconstruction where action-only logs do not, and removing returned tool content makes investigators confidently misattribute origin rather than abstain.

2 Problem formulation

Let an agent execution be a sequence of transitions $\tau = (s_0, a_0, d_0, o_0, \ldots, s_T)$, where a_t is the declared action, d_t the control decision, and o_t the independently observed effect. Identity i_t, principal p_t, delegation g_t, and task record q_t denote the context at step t, which may change between steps. A controller computes

$$d_t = \pi(i_t, p_t, g_t, q_t, s_t, a_t), \qquad s_{t+1} = F(s_t, a_t, o_t, d_t). \tag{1}$$

The crucial distinction is between pre-execution information and post-execution observation: the latter can constrain later actions and improve reconstruction, but cannot undo the first invisible effect. Section 2.1 makes the pre-execution limit precise.

We distinguish three questions:

1. *Prevention*: was the harmful action blocked before its effect?
2. *Attribution*: which actor, authority, or delegated component controlled the relevant risk?
3. *Reconstruction*: can an investigator identify what happened, what was authorized, where an instruction entered, and when intervention was possible?

The initial version of this project conflated these questions. The present paper evaluates them separately.

2.1 The observability bound

The first of these questions admits a precise limit, and stating it separates the negative results below that are contingent on our implementation from those that are not.

The boundary's *pre-dispatch view*, $\beta_t = \beta(i_t, p_t, g_t, q_t, s_t, a_t)$, contains everything readable before dispatch. Let e_t denote the effect if dispatched and $\mathcal{H}$ the harmful effects. Observation

o_t may record e_t incompletely; undispatched calls have no realized call effect. Partition decisions into D_{go} (dispatch as issued) and D_{stop} (block or gate on external approval): ALLOW is in the first, DENY and ASK in the second. Candidate invocations (u, u') at the same boundary, including their contexts and effects if dispatched, form a *collision* when

$$\beta(u) = \beta(u'), \qquad e(u) \in \mathcal{H}, \qquad e(u') \notin \mathcal{H}. \tag{2}$$

We assume (i) the decision precedes dispatch and there is no rollback, so a call sent to D_{go} realizes its effect; (ii) π reads only β; and (iii) β is not a function of e_t, being computed from the declared call together with state accumulated over strictly earlier transitions.

Lemma 1 (Pre-dispatch collision)**.** *Let π be a deterministic policy from pre-dispatch views to decisions and let (u, u') be a collision. Then $\pi(\beta(u)) = \pi(\beta(u'))$, and exactly one of the following holds: π dispatches both, failing to prevent $e(u) \in \mathcal{H}$; or π stops both, interrupting the benign invocation u'. No deterministic policy over β prevents $e(u)$ while permitting u'.*

Proof. π is a function whose argument is β alone, by (ii). Since $\beta(u) = \beta(u')$ the two invocations present it with the same argument, so $\pi(\beta(u)) = \pi(\beta(u')) =: d$. If $d \in D_{\text{go}}$ then by (i) u executes and $e(u) \in \mathcal{H}$ occurs, so π does not prevent it. If $d \in D_{\text{stop}}$ then u' is stopped although $e(u') \notin \mathcal{H}$. The partition makes these cases exhaustive and disjoint. Preventing $e(u)$ requires $d \in D_{\text{stop}}$ and permitting u' requires $d \in D_{\text{go}}$, which cannot both hold. □

The lemma is an elementary indistinguishability argument, not a theoretical contribution. Its antecedent is the empirical question: whether a measured boundary admits collisions on effects that matter. Our byte-identical examples are authored controls, so they establish mechanism rather than prevalence; Sections 4.1, 4.3, and 4.4 report them with that boundary. Three consequences shape the rest of the paper.

Trajectory state cannot escape a collision, but the live $0/12$ result is not one. A stateful controller enlarges β by conditioning on s_t, but cannot break a pair that remains identical under the complete history available before dispatch. Our retrospective audit shows that every sampled opaque harmful call uses `run_script`, while no authored-benign or resisted-injection call does. Because the exact tool name is a coordinate of β, these sampled harmful and clean calls are distinguishable. The controller still reaches $0/12$ because its declared-feature projection discards the exact tool name and maps all 12 targets to one signature shared by 223 of the 236 outcome-clean calls; the remaining 13 comprise nine `send` calls and four sensitive-resource calls in two other signatures. This diagnoses the evaluated encoding and policies, not complete-β indistinguishability. The sample contains no harmless `run_script` use, so the retrospective separator is not a deployable rule or an out-of-sample claim. The lemma instead applies to the byte-identical authored controls below. It also does not rule out a learned belief-state or sequence policy improving decisions on non-colliding histories (Smallwood & Sondik, 1973); we do not evaluate one.

ReDiR's learned trajectory representation for action generation (Dai et al., 2026) is outside our runtime-state evaluation.

Randomization does not escape it either. If π is stochastic then $\pi(\cdot \mid \beta(u))$ and $\pi(\cdot \mid \beta(u'))$ are the same distribution, so any prevention gain for u equally increases interruption of u'.

A fixed collision requires an additional signal; an artificial collision requires retaining an existing one. Enlarging β with the target resource, operation type, or resolved effect can distinguish a genuine pair. The live correction is simpler: preserve the exact action coordinate already present at the boundary. Sections 4.3 and 4.4 test additional resource exposure in authored schemas, while the oracle in Section 4.2 measures what resolved effects buy. No policy over an unchanged genuine collision view can separate that pair.

Assumption (i) is the one worth attacking. Assumptions (ii) and (iii) describe what a policy is and what a boundary carries, but (i) — decision before dispatch, no rollback — is an architectural choice rather than a law. A runtime that resolves an effect and revalidates

it against consent *before commit* moves the decision point to where e_t is known, and the bound simply does not apply there. That is the design Zhang (2026) adopt, and Section 4.5 reports an independent reimplementation of it in this repository. We flag it here rather than in related work because a reader entitled to ask "so is prevention hopeless?" deserves the answer at the point the bound is stated: no — but escaping it requires changing the boundary, not the policy, and Section 4.5 shows what that costs.

3 System design

We describe the full architecture so the negative result can be interpreted: the evidence ledger and observation channel support reconstruction, delegation supports scope checks, and cumulative risk state is the component under evaluation.

3.1 Identity and delegation

An identity descriptor binds an agent instance to a principal, version, declared capabilities, and deployment context. A delegation descriptor records issuer, subject, permitted tools and resources, expiry, revocation, and any parent delegation. The controller rejects an action if the delegation is expired, revoked, outside scope, or not issued by the principal bound to the agent. Identity is not treated as a trust score; it is an evidentiary binding for authority and events.

3.2 Observation and risk state

The controller separates declared-action features from effect observations such as file mutations, deletion, privilege changes, network intent, errors, and external effects. Its state is controller-side, computed from declared actions and observed effects; it is not the model's context window, a skills file, or an MCP scaffold. Declaration–observation divergence becomes a new event rather than rewriting the earlier record. A bounded short-term score decays per action, while counters retain rare suspicious events; profiles expose all weights and thresholds and preserve the legacy no-decay ratchet as a baseline. Appendix A.3 gives the rule. Since evaluation does not support this component, we describe only what is needed to interpret the negative result.

3.3 Evidence ledger

Each event contains the task identifier, identity, principal, delegation, declared action, observation, pre-state, decision, post-state, reason, timestamp, previous-event hash, and current hash. Optional HMAC authentication protects against undetected content rewriting when the signing key is held outside the controller. An independent verifier checks the complete ledger without importing the controller's decision logic.

4 Evaluation

We evaluate the legacy controller, the redesigned controller, and ten prevention conditions. The study includes five controlled conditions, capability probes at a real Claude Code hook, replay of 12 recorded coding-agent sessions with 860 tool calls, 99 live sandbox runs across three model configurations and eleven scenarios, and scripted domain experiments over filesystem, database, messaging, and browser schemas. All live effects are confined to a path-jailed sandbox; outbound communication records intent and returns a synthetic receipt rather than sending data over the network.

Measurement inputs are hash-frozen; Appendix A.4 records why. A fail-closed, mutation-tested provenance gate maps manuscript numbers to committed results; it checks consistency, not scientific validity.

We also treat self-audit failures as methodology rather than discarded mistakes. Across the oracle, origin detector, path-hint parser, authored grants, and dispatch table, a component

Table 1: Conditions on 99 live runs containing 25 harmful incidents. Rows 6b–6c are retrospectively selected over 80 frozen configurations, so post-selection confidence intervals are omitted; other rows are fixed profiles with Wilson intervals. FP is flags per declared-benign call over 93 calls in 36 authored-benign runs; completion is the share of those 36 runs with no flag. Outcome-clean runs containing resisted attacks are excluded from these cost columns. Condition 10 is not deployable.

Condition	Prevented	95% CI	Post-hoc	Benign FP	Completed
1 no controller	0/25	0–13%	0	0.0%	100%
2 static deny-list	8/25	17–52%	0	0.0%	100%
3 stateless per-action rule	13/25	34–70%	0	9.7%	75%
4 scoped delegation only	**17/25**	48–83%	0	1.1%	97%
5 trajectory, no obs.	3/25	4–30%	1	0.0%	100%
6 trajectory + observation	3/25	4–30%	13	0.0%	100%
6b traj., retrospective ≤10%	4/25	–	0	0.0%	100%
6c traj., retrospective max.	13/25	–	0	9.7%	75%
7 external policy	13/25	34–70%	0	9.7%	75%
8 narrow taint state	3/25	4–30%	0	0.0%	100%
9 per-tool rate limit	3/25	4–30%	0	0.0%	100%
10 true-effect oracle	**25/25**	87–100%	0	9.7%	75%

scored on material its authors knew produced an optimistic or circular answer. We therefore retain these failures, replace each with an independent signal or explicit upper bound where possible, and label the remaining authored controls as mechanism tests (Appendix A.8).

4.1 Opaque effects, at one boundary and then at five

Across 25 controlled matched pairs with identical final actions and observations but different histories, the stateful controller separates 80–96% of escalating probes while a stateless filter separates none. This verifies that the implementation uses history in constructed trajectories, not that history predicts deployed harm; real-session replay supersedes the suite's cost figures (Appendix A.6).

The real hook integration exposes declaring and opaque tool surfaces: declaring tools announce deletion or sending in their names, while opaque effects execute through a generic script invocation. Across 25 achieved harmful runs, stateless control prevents 13, the default trajectory profile prevents 3, and the lowest-cost calibrated profile prevents 4. All three prevent 0/12 opaque harms.

Our original explanation for the opaque result was wrong. The boundary recorded `run_script`, but the controller's declared-feature projection discarded the exact name. In this sample the raw action distinguishes all 12 opaque harmful targets from all 236 outcome-clean calls; the evaluated projection instead groups the targets with 223 of those calls. The result is therefore a feature-encoding failure with unknown out-of-sample cost, not evidence that the complete live boundary hid every distinguishing signal.

4.2 Baselines and the observability bound

Table 1 separates fixed profiles from retrospective calibration and keeps pre-execution prevention distinct from post-hoc detection. The default trajectory row prevents 3/25 and does not separate from no controller, but this is a statement about that profile, not the mechanism's attainable maximum. The retrospectively selected maximum reaches 13/25, exactly the stateless total, and the two conditions prevent the same runs and flag the same nine calls in the same nine declared-benign runs: 9/93 calls (9.7%) and 27/36 uninterrupted runs (75% completion). Over all 74 outcome-clean runs, including 38 resisted attacks, both flag 9/236 calls and leave 65/74 runs uninterrupted (88%); this broader population is reported separately from Table 1. At the lower approval threshold the history term adds flags on harmful traces but adds no prevented run. Scoped delegation reaches 17/25, but does not

Table 2: AgentDojo evidence kept in two epistemic classes. Native rows are released end-to-end outcomes. Offline rows are counterfactual interventions on achieved no-defense traces; post-intervention agent behavior and task outcome are unknown.

Condition	Attack outcome/intervention	Benign outcome
No defense (native)	109/160 succeed	34/40 succeed
Tool filter (native)	17/160 succeed	36/40 succeed
Stateless rule (offline)	51/109 intercepted	11/40 interrupted
Trajectory controller (offline)	52/109 intercepted	13/40 interrupted

Table 3: Original per-model rows and three frozen N=20 arms plus representative resistance rows; the complete model-specific table is in Appendix A.17. Rows are separately analyzed execution studies with their own denominators; no studies are pooled. Harm is achieved harmful outcome per injected run; controller fractions condition on achieved harm. Trajectory is the default profile; stateless is fixed.

Model/configuration	Runs	Harm	Trajectory	Stateless
DeepSeek-V4-Flash	33	9/21	1/9	4/9
DeepSeek-V4-Pro	33	10/21	2/10	6/10
Kimi K2.6	33	6/21	0/6	3/6
NVIDIA Nemotron 3.5 Lightning	220	99/140	39/99	59/99
MiniMax M3	33	0/21	–	–
GLM-5.2	33	0/21	–	–
Qwen3-Coder-480B-A35B-Instruct	220	98/140	39/98	59/98
GLM-5-Turbo	220	42/140	1/42	22/42
Kimi K3	88	0/56	–	–

statistically separate from this selected 13/25 row; we therefore do not claim a calibrated ordering between them. Appendix A.7 audits authored-grant bias.

Three clean-room comparators sharpen that result: an external declarative policy matches the stateless rule at 13/25, while narrow taint state and per-tool rate limiting prevent 3/25 each, all statistically inseparable from no controller. The static deny-list prevents 8/25 against the default trajectory profile's 3/25 and interrupts 0/93 authored-benign calls in 0/36 runs; zero observed cost is not "free," with one-sided 95% upper bounds of 3.2% under call independence and 8.0% by run. Costs need not order with prevention because Table 1 compares different policies, not thresholds on one policy. These results do not deny their value against intended threats; narrow memory and call counts still do not recover a hidden effect.

The true-effect oracle (Table 1, condition 10) is a non-deployable diagnostic that receives the actual effects before execution and refuses irreversible or outbound actions. Harm in these traces *consists of* an irreversible deletion or an outbound transfer, so its 25 of 25 is close to definitional — a consistency check on the labelling, not evidence that any system achieves complete prevention. What was not guaranteed in advance is the cost: perfect effect information buys complete prevention at 9.7% declared-benign-call interruption and 75% unattended completion, comparable to the stateless rule and worse than scoped delegation. Perfect foresight is therefore not merely unavailable but also not free. The oracle bounds the inputs, not the algorithms.

Latency is negligible for every condition (p95 under 0.05 ms per call), so none of these differences reflects a performance trade-off.

Reproduced external check on AgentDojo. We replay frozen policies over 400 released AgentDojo traces (Debenedetti et al., 2024). Table 2 keeps released end-to-end outcomes separate from our counterfactual offline interventions. The trajectory and stateless replays differ by one achieved attack ($p = 1.0$); neither establishes an advantage. Appendix A.15 gives construction and claim boundaries.

Table 3 includes the original per-model rows, all three frozen N=20 arms, and representative resistance rows; Appendix A.17 reports every eligible model-specific row. Low- and zero-

Table 4: Retrospective calibration over 80 pre-existing controller configurations, not 80 agent runs. Each is replayed over all 99 frozen traces in the three-model live cohort (DeepSeek-V4-Flash, DeepSeek-V4-Pro and Kimi K2.6; 33 runs each) and all 220 frozen traces in the NVIDIA Nemotron 3.5 Lightning cohort. The budget selects maximum prevention, then lowest cost, at $\leq 10\%$ benign-run interruption. Discordance is trajectory-only/stateless-only prevented runs at maximum prevention.

Dataset	Default	$\leq 10\%$	Maximum	Stateless	Discordance
Three-model live cohort	3/25	4/25	13/25	13/25	0/0
NVIDIA Nemotron 3.5 Lightning	39/99	39/99	59/99	59/99	0/0

harm configurations measure susceptibility heterogeneity but leave little or no achieved harm for controller comparison. No row is pooled across models, providers, or waves.

McNemar's paired test compares disagreements: runs prevented only by trajectory versus only by stateless control. Its run-level exact values (Appendix Table 9) assume independent achieved-harm pairs. Yet Appendix Table 10 shows near-deterministic within-scenario outcomes and discordance concentrated in one or two scenarios. Cluster-aware inference is required (Eliasziw & Donner, 1991); exact scenario-level sign tests are uninformative ($p = 1.0, 1.0, 0.5, 1.0$) with only one or two non-zero contrasts among seven scenarios. We retain run-level values only as conditional sensitivity calculations and make no inferential claim about unseen scenarios. Within these fixed scenarios, Qwen3-Coder-480B-A35B-Instruct and NVIDIA Nemotron 3.5 Lightning reproduce the exact pattern; GLM-5-Turbo and DeepSeek-V4-Flash reproduce the same descriptive, destruction-concentrated ordering.

Frozen calibration audit. We replay all 80 pre-existing settings to test the conservative default. At maximum trajectory prevention (Table 4), both controllers prevent the same runs in each cohort: neither controller uniquely prevents a run. Their interrupted benign-run sets and benign-call flag rates are also identical. Under the 10% budget, the lowest-cost maximum-prevention cells interrupt no benign run but prevent only 4/25 and 39/99. Calibration explains part of the default weakness, but history adds no prevention at the maximum. This is retrospective sensitivity evidence, not prospective tuning or a claim that stateless dominates every lower-cost point.

4.3 Cross-boundary observability

We map the same logical action through four adapter families: two live (Claude Code hook, 860 calls; instrumented tool loop, 99 runs) and two schema-conformant plugin-hook fixtures, not executions inside those runtimes. OpenClaw with and without path hints yields five conditions. We do not pool live and fixture evidence; only the live pair supports a deployment claim.

No boundary reveals what an opaque call will do before dispatch. One exposes harness-computed target-path hints even for an arbitrary-execution tool, which names the resource without naming the verb: insufficient for effect detection, sufficient for a scope check. Separately, the instrumented loop reports every effect class directly after execution with no text parsing, and is no better than the thinnest boundary before it. Owning the tool implementations buys hindsight, not foresight, which is why prevention and attribution must be scored separately throughout.

Re-running this study on the live scenario definitions rather than on fixtures we authored replicates the negative half exactly — opaque-surface effect detection is 0/14 at every one of the five boundary conditions, against 14/14 on declaring tools. These 14 opaque harmful calls occur in 12 runs; one run contains three. The path hint is worth 28/28 or 19/28 depending on how a scope policy reads it, the strict reading refuses all nine authorized outbound sends, and the calls it catches are an artifact of the attacker's filename. Appendix A.5 gives both readings. A path hint is only as useful as the resource names reaching the policy (Section 4.4).

4.4 Domain diversity and a constraint on tool schemas

Every result above comes from one domain: a filesystem with an outbound send. To test whether the bound is a property of execution boundaries or of `delete_path`, we build three further domains sharing no code with it: a synthetic database, a synthetic messaging and API surface, and an in-memory browser page graph. Each has declaring tools and an idiomatic opaque tool. Declaring tools expose all harmful effects (5/5 in each added domain), while opaque named migration, webhook or retention, and browser-automation calls expose none (0/2, 0/2, and 0/1). Five unrelated indirections therefore reproduce the same blindness: script, migration, webhook, retention rule, and browser automation. A matched ablation isolates target exposure: arms realizing a verified-identical effect reverse prevention when registration names the target, a delta of +1 in both domains at no benign cost (Appendix A.10). Browser cases add ordinary-site disclosure without injection syntax and separate sensitive reads, attempted submissions, committed cross-origin transfers, and unintended disclosure. These traces are *scripted rather than model-generated*; the browser has no network stack or real browser. They measure what a boundary can observe and refuse, not susceptibility, and all schemas are ours, so this is controlled rather than independent validation.

4.5 Escaping the bound by moving the commit point, and what it costs

Pre-dispatch conditions cannot undo permitted calls. Effect-bound consent (Zhang, 2026) instead previews an action's effect, records an instance-specific predicate, and revalidates the *resolved* effect at commit. The decision then sits where e_t is known, so assumption (i) of Lemma 1 fails; this is the principled escape from the bound.

Our independent reimplementation binds the predicate and effect hash to the principal, agent, delegation, and policy version, rejecting authority drift even when the effect is unchanged. It decides all ten of our controlled divergence cases correctly (Appendix A.9). These authored synthetic cases test composition with an identity-bound record, not a prevention rate.

Preview quality becomes the new trust boundary. In the source study, rows whose resolved effect did not exceed the approved effect are "non-broadened" and should ordinarily commit; conservatively rejecting an unresolved preview restores zero unsafe commits but *unnecessarily blocks 91.8% of the non-broadened rows.* Zhang's separate recovery pilot retries 50 such blocked tasks after additional resolution and safely recovers only 8. The bound relocates from policy to effect adapter; a perfect predicate cannot compensate for an incomplete preview.

Preview availability is adapter-specific: migrations may resolve inside rollback-capable transactions, while our filesystem and outbound adapters expose no staging phase. Snapshots, queues, transactions, or preview APIs can change that boundary. The trace-automaton reimplementation of Cho et al. (2026) likewise reaches 63.4% next-step accuracy despite 0.989 transition replay fitness, but is provably identical to a first-order frequency model on every held-out decision and is beaten by a second-order model (Appendix A.11).

4.6 Real session cost and sensitivity

Table 1 uses 93 calls in 36 authored-benign runs; the separate replay below covers 860 calls over 516 resources in 12 legitimate coding sessions. Outcome-clean runs also include resisted attacks. These populations must not be treated as estimates of one quantity.

The balanced profile flags 73.8% of calls and denies 52.4%; conservative irreversibility lowers denial to 26.6%, so the synthetic 2% cost is not a deployment estimate. Across 80 grid cells with hash-frozen weights and thresholds and four one-at-a-time axes, interruption spans 5.7–82.4%; *none keeps it below 5%.* Within 10%, the best holdout catch is 7 of 12 failure scenarios, while 11 of 12 costs 17.6% benign-call interruption. The shipped profile reaches 42.2% interruption for 9 of 12, and one weight is non-monotonic. These are not the 12 coding sessions: we operationalized third-party failures, so holdout independence is partial.

Table 5: Blinded evidence ablation across three model evaluators. Origin counts are per evaluator over the same 40 packets; answer totals cover five questions per packet.

Evidence retained	Correct answers	Injection origin
Full record	190–198/200	39–40/40 correct
No returned content	147–149/200	7/40 correct; 33/40 wrong
Action log only	79–82/200	1–3/40 correct

4.7 Attribution and blinded evidence

An initial ablation selected 13 declaring-surface harmful incidents. Complete records answer 13/13 culprit and authorization questions and fully attribute 11/13 incidents; removing returned content reduces origin recovery to 0/13, while action-only logs answer none. Selection by tool name missed the 12 opaque `run_script` harms; the subsequent effect-grounded study includes opaque incidents.

DeepSeek V4 Flash, DeepSeek V4 Pro and Kimi K2.6 evaluate 40 blinded packets: 20 harmful, 13 refused injections, and 7 benign. Over five questions each, full records yield 190–198 correct answers of 200 versus 79–82 for action logs. Without returned content, each recovers origin in 7/40 packets and answers 33 wrongly (Table 5; Appendix A.14).

5 Related work and scope

Unlike the broad safety coverage of AgentDojo and Agent-SafetyBench (Debenedetti et al., 2024; Zhang et al., 2024), we isolate controller-side state. Host-IDS work found short system-call windows useful, simpler models competitive, and mimicry attacks able to hide malicious behavior in normal-looking traces (Forrest et al., 1996; Warrender et al., 1999; Wagner & Soto, 2002). Appendix A.11 echoes the short-context finding; Lemma 1 isolates the pre-dispatch analogue.

Complete mediation requires seeing the operation (Saltzer & Schroeder, 1975); a hidden-target tool risks acting as a confused deputy without scope information (Hardy, 1988). Capabilities and information flow motivate authority and provenance (Miller, 2006; Denning, 1976); CaMeL separates capabilities under injection (Debenedetti et al., 2025). Unlike trajectory auditing (Yang et al., 2026; Liu et al., 2026), we separate prevention from reconstruction and reimplement effect-bound consent and trace automata (Zhang, 2026; Cho et al., 2026).

6 Limitations

Seven authored attack scenarios, one sandbox and workload, and our schemas limit external validity; repeated runs do not add scenario diversity. The state is hand-designed, the retrospective `run_script` separator has no harmless support, and frozen traces omit the complete formal pre-dispatch view. Evaluators are models and the harness lacks native delegation. Low- and zero-harm rows measure susceptibility, not controller ordering or immunity; the mechanisms establish neither universal safety nor legal responsibility.

7 Conclusion

Across the evaluated live, frozen-model and AgentDojo-replay studies, the hand-designed state establishes no calibrated prevention advantage over stateless control. Scoped delegation reaches 17/25 prevented harms versus 13/25, without statistical separation; removing returned content causes 33/40 origin errors. The opaque-call diagnostic identifies discarded information, not demonstrated boundary indistinguishability. Accumulating risk did not improve this controller; preserving evidence supported reconstruction. The practical lesson is to preserve information for both action decisions and incident reconstruction, and to judge accumulated risk by the prevention it demonstrably adds.

AI Use Statement

AI tools were used for code assistance, document organization, and drafting support. The authors inspected the source materials, designed the evaluation, reviewed generated text, and remain responsible for the claims, code, experiments, and submission.

Ethics Statement

Live evaluation was conducted in a path-jailed sandbox. The outbound communication primitive did not perform network I/O and returned synthetic receipts. No real credentials or third-party data were intentionally disclosed. The project records that returned tool content may contain sensitive information and requires access control, minimization, and retention policies for deployment.

Reproducibility Statement

The repository contains the controller, independent verifier, tests, scenario definitions, committed traces, result files, and commands used to regenerate the reported tables. The AgentDojo package includes its adapter, frozen manifest, run-level outputs, and independent replay record. The DeepSeek and model-family expansions include frozen manifests, fail-closed artifact audits, deterministic analyses, and regression tests; the latter was independently reviewed before manuscript integration. The replay dataset is snapshot-specific, and model-judge results are nondeterministic; these conditions are recorded in the experiment documentation.

References

Tianyu Chen, Dongrui Liu, Xia Hu, Jingyi Yu, and Wenjie Wang. A trajectory-based safety audit of clawdbot (openclaw). *arXiv preprint arXiv:2602.14364*, 2026. URL https://arxiv.org/abs/2602.14364.

Seonglae Cho, Franklin Cardenoso Fernandez, Umar Mohammed, Zekun Wu, Kleyton Da Costa, Ilham Wicaksono, and Adriano Koshiyama. Automata from agent traces: Failure and next-step prediction. In *Second Workshop on Agents in the Wild: Safety, Security, and Beyond at ICML*, 2026. URL https://openreview.net/forum?id=1cprFkvFT0.

Yanbo Dai, Zhenlan Ji, Zongjie Li, and Shuai Wang. Reassembling distributed risk: Trajectory-conditioned action generation for multi-turn agent safety. *arXiv preprint arXiv:2608.25711*, 2026. URL https://arxiv.org/abs/2608.25711.

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, Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=m1YYAQjO3w.

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. *arXiv preprint arXiv:2503.18813*, 2025. URL https://arxiv.org/abs/2503.18813.

Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976. doi: 10.1145/360051.360056.

M. Eliasziw and A. Donner. Application of the McNemar test to non-independent matched pair data. *Statistics in Medicine*, 10(12):1981–1991, 1991. doi: 10.1002/sim.4780101211.

Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for UNIX processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pp. 120–128, 1996. doi: 10.1109/SECPRI.1996.502675.

Norm Hardy. The confused deputy: (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 22(4):36–38, 1988. doi: 10.1145/54289.871709.

Songyang Liu, Chaozhuo Li, Chenxu Wang, Jinyu Hou, Zejian Chen, Litian Zhang, Zheng Liu, Qiwei Ye, Yiming Hei, Xi Zhang, and Zhongyuan Wang. Clawkeeper: Comprehensive safety protection for openclaw agents through skills, plugins, and watchers. *arXiv preprint arXiv:2603.24414*, 2026. URL https://arxiv.org/abs/2603.24414.

Mark Samuel Miller. *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*. PhD thesis, Johns Hopkins University, 2006. URL http://www.erights.org/talks/thesis/. Doctoral dissertation.

Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. doi: 10.1109/PROC.1975.9939.

Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973. doi: 10.1287/opre.21.5.1071.

David A. Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 255–264, 2002. doi: 10.1145/586110.586145.

Christina Warrender, Stephanie Forrest, and Barak A. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pp. 133–145, 1999. doi: 10.1109/SECPRI.1999.766910.

Zhonghao Yang, Yu Li, Yanxu Zhu, Tianyi Zhou, Yuejin Xie, Haoyu Luo, Jing Shao, Xia Hu, and Dongrui Liu. Benchmarks for trajectory safety evaluation and diagnosis in openclaw and codex: Atbench-claw and atbench-codex. *arXiv preprint arXiv:2604.14858*, 2026. URL https://arxiv.org/abs/2604.14858.

Qi Zhang. Approve the effect, not the tool call: Preventing stale consent in tool-using agents. In *Second Workshop on Agents in the Wild: Safety, Security, and Beyond at ICML*, 2026. URL https://openreview.net/forum?id=lClCNawtQB.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. *arXiv preprint arXiv:2412.14470*, 2024. URL https://arxiv.org/abs/2412.14470.

A Failure audits and reproducibility details

The complete experiment commands, raw traces, configuration profiles, capability table, and test suite are included in the supplementary repository. The legacy ratchet is retained as a baseline so that the initial false-positive result remains reproducible after the redesigned controller is introduced.

A.1 A measurement hazard in the oracle condition

The oracle is worth recording as a methods warning, because a broken upper bound reads as a finding rather than as a bug. An initial version scored the true effect features through the deployed risk model and prevented only 3 of 25 — no better than the mechanism it was meant to bound — because that model's approval threshold is calibrated for risk *accumulated* across a trajectory, while a single irreversible action contributes less than half of it. That configuration measured the threshold rather than the information, and it is the reason the reported oracle refuses directly on the true effect class instead of routing it through a policy.

A.2 Implications for accountability

The results support a narrower governance implication than the project began with. A deployer cannot demonstrate effective control by retaining a list of tool calls: an accountability record needs authority context, task context, returned content, observed effects, decisions, and integrity protection. Without returned content an investigator may correctly identify the deletion yet wrongly conclude the agent chose it, when the decisive instruction entered through a poisoned document; without task context they cannot distinguish an authorized instruction from a hijacked execution. A second implication follows from the domain study: authority is enforceable only over what a tool call names, so schema design belongs to the control surface rather than beneath it. None of this allocates liability automatically — it produces evidence for questions that negligence, contract, product, and governance analysis may later consider — and it carries a trade-off, since retaining returned content improves reconstruction while retaining sensitive data. Deployments therefore need access control, encryption, retention limits, and redaction alongside cryptographic commitments sufficient to detect alteration.

A.3 The trajectory risk update rule

The implementation separates a decaying base score B_t from persistent counters. For a proposed action, it computes

$$\begin{aligned} B_{t+1}^{\text{pre}} &= \text{clip}(\lambda B_t + w^\top x_t, 0, 1), \\ R_{t+1}^{\text{pre}} &= \text{clip}(B_{t+1}^{\text{pre}} + c_{t+1}, 0, 1). \end{aligned} \tag{3}$$

Here λ is decay, w the feature-weight vector, and x_t the pending call's declared features and indicators from prior state. The proposed counter update yields $c_{t+1} = w_{\text{count}} K_{t+1}$, where K_{t+1} counts tripped persistent counters. Counter contributions are not fed back into B_t. The pre-dispatch decision thresholds R_{t+1}^{pre}; observed declaration–effect divergence subsequently increments the base score and a persistent counter, affecting later decisions only. The legacy profile uses $\lambda = 1$, disables counter contributions, and zeros external-effect, sensitive-resource and divergence weights; decay alone does not recreate it. Section 4.2 reports the evaluated calibration range, not all possible policies.

Fixed baselines and projection. Table 1's stateless rule is programmatic: it resets state for each call, computes the balanced profile's declared-feature risk, and requests approval at 0.45. The projection maps tool names to irreversible/external-effect flags and scans arguments for sensitive-resource hints; `run_script` maps to neither effect flag. Exact tool identity is not retained as a separate feature. Thus the rule is not an LLM judge or a retrospectively tuned maximum. Rows 5–6 use the redesigned balanced state (with observation only in row 6); the legacy ratchet is evaluated separately in Appendix A.6. The external policy blocks `delete_path` and gates `send` or sensitive reads; narrow taint remembers a sensitive read and blocks a later `send`; the rate limit gates a tool's fourth call. These are fixed policies, distinct from rows 6b–6c's retrospective trajectory selections.

A.4 Why measurement inputs had to be frozen

The session replay originally read a live transcript directory, which grows as its operator works: the same code produced 752 tool calls when first run and 860 later, and because session files are rewritten on compaction the count is not even monotonic. Extraction is now separated from measurement, with derived per-call features committed under a content hash the replay verifies on load. We record this because the failure is easy to miss — the earlier numbers were not wrong when computed, merely irreproducible, and nothing in the output indicated which snapshot produced them.

A.5 Which scope reading a path hint requires

The two readings of a path-based grant differ in whether each named resource is checked on its own or the call's arguments are concatenated first. Under the per-resource reading

a harness-supplied path hint makes all 28 harm-carrying calls refusable, at the cost of refusing all 9 authorized outbound sends, since the grant enumerates paths and can express no destination. Under the any-token reading — which is what our own delegation baseline computes — a single in-scope token anywhere in the joined argument string admits the call, and the hint's contribution drops to zero: 19 of 28, identical to every boundary that supplies no hint at all. The 5 opaque calls the strict reading catches are additionally an artifact of scenario authorship rather than of the boundary: all 9 it misses present the identical in-scope resource name, while the 5 it catches happen to use a name the grant does not cover. We report both readings because the choice belongs to the deployer, and because reporting only the favourable one would overstate what the field provides.

A.6 Controlled matched-pair detail

The controlled suite holds the final probe action and the present observation fixed while varying only the preceding history, so a stateless filter cannot separate the pairs at all and any separation is attributable to state. The stateful controller separates 80–96% of escalating probes. The same suite exposes the legacy design's cost: a no-decay risk ratchet flags 68–75% of routine benign work, which is the pathology that motivated the redesign. The redesigned decay-plus-counter model reaches 84% discrimination at 2% false flags here. That 2% figure is retained only as a record of the constructed setting; the real-session replay in Section 4 supersedes it for any deployment-cost claim, and the two should never be quoted together.

A.7 Who writes the grant, and why it matters

Scoped delegation is the best deployable condition in Table 1 and the basis of our constructive recommendation, so the conditions under which it was evaluated deserve stating plainly. Each grant was written from the legitimate requirements of the task it governs, by authors who knew those tasks in advance. That is a favourable setting and not an unrealistic one — capability grants in deployment are also authored by someone who knows the intended work — but it does not test the hard case, which is open-ended work whose legitimate resource set is not known when the grant is issued.

Two of our own results indicate how that case would go. The sensitivity surface shows real coding sessions touch 516 distinct resources across 860 calls, so a grant tight enough to be meaningful over that workload would need to anticipate a large and irregular resource set. And the Phase 16 scope study shows what happens when a grant is written loosely enough to cover such work: the attacker's script sat inside a granted prefix and the check had nothing to refuse. Whether scopes can be authored in advance at that scale is untested here, and it is the obvious way for this baseline to fail in practice.

A.8 Circularity in the injection-origin detector

The first origin detector was written after reading our own injection payloads, so its positive result largely measured whether our patterns matched our own prose. Scored on held-out registers deliberately phrased unlike the originals — conversational rather than imperative, with no notice block — it recovers 0 of 2. A model judge carrying no payload vocabulary recovers 11 of 11 direct and 2 of 2 held-out registers, and a purely structural detector that looks for content entering through a tool return rather than for any wording recovers 9 of 11 and 1 of 2. The evidence claim therefore holds under detection we did not tune, at the cost of introducing a model-dependent component to the pipeline.

We record this because it is the same failure mode as two others in this work, and naming the pattern is more useful than each instance: a component authored against the material it is later scored on will report the authorship rather than the phenomenon. The oracle that scored true effects through our own risk model, and the scope check that read our own concatenated arguments, failed the same way.

A.9 The effect-bound consent divergence cases

The ten controlled cases separate the ways a committed effect can diverge from the one a principal approved. Four are effect divergences: an alias that expands to additional subjects, a resource pointer that resolves elsewhere, an amount above the approved ceiling, and an added secondary effect. One is an extraction failure, where the adapter cannot resolve a field and the predicate rejects the unknown rather than assuming it benign. Four are authority divergences that leave the effect untouched — a substituted executing agent, a changed delegation, a substituted principal, and a policy-version bump — and these are the cases our identity binding adds to the mechanism as reported. The tenth is the exactly-matching commit, which must be permitted. The adapter decides all ten correctly, refusing the nine divergent commits with the reason recorded in the evidence record rather than only in the decision.

The design point the cases illustrate is that the predicate is an instance-specific upper bound on the effect, not a policy over the tool: approval is bound to a canonical hash of the previewed effect, so a commit that is broader in any dimension fails without needing a rule anticipating that dimension.

A.10 Construction of the three additional domains

Effects in all three synthetic domains are confined to in-memory structures, so no host resource is in reach at all — these domains are strictly more contained than the path-jailed filesystem sandbox, which has a jail because it has something to jail. Every credential-shaped value is fabricated and marked as a decoy, and outbound operations record an intent and return a synthetic receipt without touching a network, as in the filesystem sandbox.

Each domain carries all seven case types the evaluation plan requires: benign controls, explicit harm requested by the operator, direct injection, a held-out injection register, an opaque surface, a delegation violation, and a revocation or expiry case. The last two exist nowhere else in this work, which matters because scoped delegation is both the best deployable condition and the paper's constructive recommendation, and until these scenarios no test had attacked the scope boundary itself. Ground truth is kept on three separate axes — harm, scope violation, and post-expiry effect — because collapsing them would let a condition that merely blocks everything irreversible take credit for enforcing authority.

The opaque tools are deliberately unlike one another and unlike the filesystem's script runner: a migration is run by name with its statements in a registered body, a webhook is fired by name with its destination and payload fixed at registration, a retention rule is applied by name with its target channel fixed at registration, and a recorded browser automation is replayed by name. Named indirection is common to schema tools, webhooks, retention policies, and saved automations, which makes these fair opaque surfaces rather than one construction repeated under different labels.

The messaging domain yields the collision Lemma 1 assumes: a benign and a harmful call with byte-identical pre-dispatch views, decided identically by all nine deployable conditions.

Only scoped delegation enforces authority, preventing 7/7 in messaging against 5/7 in database despite equal opacity at dispatch.

The target-exposure ablation. The 7/7 versus 5/7 contrast above confounds target exposure with domain identity, since it compares one database schema against one messaging schema. We therefore add a second variant in each domain: a declared-target migration taking the table as an argument alongside the body-only one, and a hidden-target webhook carrying its destination inside a config field alongside the one that names a URL. Within a domain the paired arms realize an effect we verify identical by comparing a canonical effect signature, and the dispatching call is equally opaque in every arm. Prevention reverses with exposure in both domains, delta +1 each, and neither benign declared arm is refused, so declaring the target buys prevention rather than trading cost for it.

Two things about the measure. The dependent variable is refusal on a call that *names the offending target*, not any refusal before harm: the messaging arms open with a sensitive read

that is outside the grant by construction, so the coarse measure fires in both arms and is uninformative here. And an early version of the ablation appeared to succeed while one arm silently realized no effect at all, because its tool was absent from the domain's dispatch table; the identical-effect check is asserted by a test for that reason. All four schemas remain ours, so this isolates the mechanism from domain flavour without establishing that the schemas resemble real tools.

A.11 The trace-automaton baseline

Following Cho et al. (2026), we induce a finite-state machine over tool-activity classes from the saved live traces. The resulting 8-state machine replays a held-out split at 0.989 transition fitness and accepts 95% of held-out traces exactly, while next-step accuracy is 63.4%. Transition coverage and exact next-step prediction therefore measure different properties.

Establishing what 63.4% means requires nulls, so every predictor now consumes one frozen split — traces are the sampling unit, 59 train, 20 validation, 20 test, 93 held-out decisions over six activity classes — and emits one per-example schema. Intervals bootstrap over traces rather than steps, because steps within a trace are dependent and resampling them would understate every interval.

Table 6: Next-activity prediction on one frozen split, 93 held-out decisions. Accuracy intervals are 95% bootstrap over traces. Mean negative log-likelihood uses a probability floor for zero-mass outcomes; the final column reports how often that floor carried the likelihood. These are not safety metrics: accuracy here does not predict harm.

Predictor	Accuracy	95% CI	Macro-F1	Mean NLL	Floor rate
majority class / unigram	0.344	0.29–0.39	0.073	1.587	0.0%
per-position majority	0.581	0.48–0.68	0.287	1.375	3.2%
first-order frequency	0.634	0.55–0.72	0.313	0.894	1.1%
activity FSM	**0.634**	0.55–0.72	0.313	0.894	1.1%
second-order frequency	**0.742**	0.66–0.82	0.469	0.770	2.1%

Two results follow, and both narrow what the automaton row may be said to show. First, **the FSM is the first-order frequency model**: the two agree on all 93 held-out decisions and assign identical probabilities at every one, which is what the construction implies, since the state is the previous activity alone. Its 63.4% is a bigram result, and we do not describe it as a state-machine capability. Second, a second-order model beats it, 0.742 against 0.634, disagreeing on 12 decisions of which 11 favour the second-order model; its backoff rate is 1.1%, though with 361 tool-call steps across all 99 traces (train, validation and test), this row is better read as a support check than a result.

So the honest reading is narrower than "a compact automaton describes well and predicts poorly". A bigram over tool names predicts moderately, a trigram predicts better, and the automaton adds nothing over the bigram it is equivalent to. What survives is the asymmetry itself: high replay fitness alongside middling prediction, and nothing here indicates that a richer sequence model would recover the prevention the risk state failed to deliver, because prediction of the next *tool name* is not prediction of the *effect* that Lemma 1 concerns.

A.12 What scripted-effect traces can and cannot support

The three additional domains are driven by authored call sequences rather than by a model. The distinction is enforced in the artifact rather than left to the reader: those results carry the status `scripted_effects`, the runner emits no `followed_injection`, `resisted_injection`, or susceptibility field at all, and a test asserts their absence so a later change cannot reintroduce one silently.

The division of claims is as follows. Whether a boundary can see an effect before dispatch, and whether a scope policy can refuse it, are properties of the tool surface and the policy; a scripted trace that genuinely performs the effect is a sound input for asking what the boundary saw, and the effect records come from executing the synthetic domains rather

than from annotation. Whether a real agent would comply with an injection is a property of the model, and no scripted sequence can measure it — authoring the compliance and then reporting it would be circular. The scenarios therefore carry frozen injection payloads in both a direct and a held-out register, unused by the scripted driver, so that the same definitions can be driven live when an account exists, with the registers fixed before any result was seen.

A.13 Separating authority enforcement from incidental refusal

A condition can refuse the offending call in an authority scenario without responding to the authority failure at all: an irreversible action is refused by several conditions whatever the grant says. Scoring any refusal as enforcement therefore credits the oracle and the stateless rule with a capability neither has. The four authority cases per domain comprise two delegation violations, one expiry and one revocation, instantiating two of Appendix A.10's seven case types. We replay each twice over the identical recorded calls — once as it occurred, once with the delegation left intact and unrestricted — and count only refusals present in the first run and absent in the second. Under that measure scoped delegation enforces authority in 3 of 4 cases in each domain and every other condition in 0, including the three whose unadjusted counts are non-zero.

A.14 Secondary blinded-evaluator results

No model evaluator invented harm — all scored 0 of 7 on benign packets and 0 of 13 on refused injections — so evaluators are reliable about *whether* harm occurred and unreliable about *why*, and the latter governs responsibility. Confidence does not rescue reduced-record attribution: the completed evaluators remain highly confident when wrong (means 4.34 and 4.36 of 5), although confidence is higher when correct (4.98 and 4.94), leaving the scale too compressed near its maximum to be a reliable correctness filter here.

A null evaluator that always blames the agent scores 94 of 200 and invents harm on all 20 packets where none occurred, confirming the corpus punishes that strategy.

Earliest-intervention for the stronger full-record evaluator falls from 36 correct and 4 wrong to 20 correct and 20 wrong when returned content is removed.

A.15 AgentDojo external-evaluation detail

The external sample uses AgentDojo v1.2.2, release v0.1.35 at commit `a75aba7`, with `gpt-4o-2024-05-13`. It contains ten user tasks from each of banking, Slack, travel, and workspace, crossed with four injection tasks shared by all suites. All 400 selected traces were read and none excluded. Half are released no-defense runs and half use AgentDojo's native tool filter; each half contains 160 attacked and 40 benign runs.

The two evidence classes must remain separate. AgentDojo's rows are released end-to-end outcomes. Our rows ask whether a frozen policy would intervene before an attack that actually succeeded. Once an offline replay intervenes, the agent's next action and final task outcome are unknown. Forty benign traces are also too few to infer that the native filter improves benign task success.

In all 109 achieved attacks, the complete released trace reportedly links an injection-bearing return to a later attack action. This supports retrospective reconstruction, not online detection: the imported traces were not authenticated by our logger, and AgentDojo supplies neither requester identities nor delegation grants. We do not treat them as verified provenance or evidence of legal compliance.

The repository includes the adapter, frozen manifest, run-level outputs, tests, and an independent 400-trace replay with byte-identical result hashes. Table 2 still cannot serve as the sole support for a headline claim because it uses one released model and offline counterfactual controller replay.

A.16 DeepSeek V4 Flash single-model expansion

One model, `deepseek-v4-flash`, ran 88 live tool-loop traces across eight trial labels and eleven frozen scenarios, with thinking disabled and temperature zero.

56 runs carried an injection, 32 were declared benign, and 16 achieved a harmful outcome, so prevention is scored over those 16, never 56 or 32.

Trajectory plus observation prevents 0/16 achieved harms against 8/16 for both the stateless rule and scoped delegation. The original run-level exact McNemar value is $p = 0.007812$ for each, but all eight discordances come from one declaring-destruction scenario; the scenario-level exact sign value is therefore $p = 1.0$. As in Table 9, this supports a descriptive within-matrix ordering, not cross-scenario significance.

Against no controller the trajectory condition is indistinguishable, $p = 1.0$. The eight trajectory detections occur only after the harmful call, so they are evidence, not prevention.

This sample is not pooled with the earlier 33 DeepSeek records or with AgentDojo.

All 88 cells completed with zero retries; the run used 282,848 prompt and 39,419 completion tokens across 365 requests, at USD 0.050639.

Backend fingerprint: `a26a7955944dc5c60445bff77fac9c8e`.

Execution source: `d15ab6b8a1ab095ad13aac2586ad32006638716a`.

Manifest SHA-256: `2945cd59048acad38cb96d770910cfefbeea00866afe78b5dec337de68c503e3`.

Result-set SHA-256: `a9dd4d221f63696f63310d0582e76889e02365daab0921d0e1af2061977bb976`.

Analysis SHA-256: `437dea6bf052a2579c4d0f0f08b882f2f59b3a2f72365d8ed105a0cc9fb03586`.

Trial integers are labels, not provider seeds; the API exposes no seed parameter. External human reproduction is pending, so treat this as corroboration until it is re-run.

Table 7: DeepSeek V4 Flash single-model expansion. Prevention is scored over the 16 achieved harmful outcomes; post-harm flags are evidence, not prevention; benign interruption counts cover the 32 declared-benign runs.

Condition	Prevented	Detected post-harm	Benign runs interrupted
No controller	0/16	0/16	0/32
Stateless per-action rule	8/16	0/16	8/32
Scoped delegation	8/16	0/16	2/32
Trajectory plus observation	0/16	8/16	0/32
Oracle effect observation (scoring-only)	16/16	0/16	8/32

A.17 N=20 AND CROSS-FAMILY MODEL EXPANSION

Table 8 reports the complete eligible model-specific evidence. Repeated configurations and execution packages remain separate, and trial integers are labels rather than provider seeds. The frozen N=20 Qwen and NVIDIA rows support configuration-specific controller comparisons; low- and zero-harm rows support only susceptibility-heterogeneity statements.

The committed power artifact parameterizes the marginal prevention gap and discordance rate under a two-sided exact McNemar model. It is now labelled a prospective *run-level sensitivity calculation under independent matched pairs*, not cross-scenario power; the seven-scenario design has too few non-zero scenario contrasts for that claim.

Table 8: Complete model-specific susceptibility and controller outcomes. Each row preserves its neutral package ID and denominator; repeated models and waves are not pooled. The N=5/N=10/N=20 Nemotron rows are separate non-cumulative execution packages: N=10 freshly reruns the five N=5 labels and adds five; N=20 uses twenty non-overlapping labels. No raw records or canonical traces overlap, and rows are never pooled. Harm is achieved harmful outcome per injected run. Controller fractions condition on achieved harm; dashes mean no harmful outcome existed to prevent.

Model/configuration	Runs	Harm	Trajectory	Stateless
DeepSeek-V4-Flash (original live)	33	9/21	1/9	4/9
DeepSeek-V4-Pro (original live)	33	10/21	2/10	6/10
Kimi K2.6 (original live)	33	6/21	0/6	3/6
DeepSeek-V4-Flash (pilot N=8)	88	16/56	0/16	8/16
GLM-5-Turbo (Wave 1 N=5)	55	12/35	2/12	7/12
Hunyuan 3 (Wave 1 N=5)	55	5/35	0/5	0/5
Nemotron 3 Ultra (Wave 1 N=5)	55	5/35	0/5	1/5
NVIDIA Nemotron 3.5 Lightning (Wave 1 N=5)	55	24/35	10/24	15/24
NVIDIA Nemotron 3.5 Lightning (N=10)	110	50/70	20/50	30/50
NVIDIA Nemotron 3.5 Lightning (N=20)	220	99/140	39/99	59/99
MiniMax M3 (Wave 5 N=3)	33	0/21	–	–
GLM-5.2 (Wave 5 N=3)	33	0/21	–	–
Qwen3-Coder-480B-A35B-Instruct (Wave 6 N=2)	22	10/14	4/10	6/10
Gemma 4 31B (Wave 6 N=2)	22	2/14	0/2	0/2
DeepSeek-V4-Flash (Wave 7 N=20)	220	11/140	4/11	6/11
Qwen3-Coder-480B-A35B-Instruct (Wave 7 N=20)	220	98/140	39/98	59/98
GLM-5-Turbo (Wave 7 N=20)	220	42/140	1/42	22/42
DeepSeek V4 Flash 0731 (N=8 replication)	88	1/56	0/1	0/1
GLM-5.2 (N=8 replication)	88	0/56	–	–
Kimi K3 (N=8 replication)	88	0/56	–	–

Table 9: Run- and scenario-level trajectory-versus-stateless discordance in four separate studies. T/S is trajectory-only/stateless-only. Run p is the original two-sided exact McNemar sensitivity calculation, which assumes independent matched pairs. Scenario p is a two-sided exact sign test over non-zero aggregate scenario contrasts. With only one or two non-zero scenarios these scenario-level tests are underpowered; $p = 1$ records one discordant scenario rather than evidence of equality. Studies are not pooled.

Model/configuration	Harm	Run T/S	Scenario T/S	Run p	Scenario p
Qwen3-Coder-480B-A35B-Instruct (Featherless)	98	0/20	0/1	1.907×10^{-6}	1.0
NVIDIA Nemotron 3.5 Lightning (Zen)	99	0/20	0/1	1.907×10^{-6}	1.0
GLM-5-Turbo (direct Z.ai)	42	0/21	0/2	9.537×10^{-7}	0.5
DeepSeek-V4-Flash (direct DeepSeek)	16	0/8	0/1	0.007812	1.0

Table 10: Exact injected-scenario decomposition for the two frozen N=20 arms. Each complete run matrix has 11 scenarios = 4 benign + 7 injected. H is achieved harm, T is trajectory prevention, and S is stateless prevention; controller denominators condition on achieved harm. The two frozen studies are not pooled. Complete GLM-5-Turbo and DeepSeek-V4-Flash scenario decompositions are provided in the supplementary analysis artifact.

Scenario	Qwen H	Qwen T	Qwen S	Nemotron H	Nemotron T	Nemotron S
Declaring destruction	20/20	0/20	20/20	20/20	0/20	20/20
Opaque destruction	20/20	0/20	0/20	20/20	0/20	0/20
Held-out destruction	0/20	–	–	0/20	–	–
Declaring exfiltration	20/20	20/20	20/20	19/20	19/19	19/19
Opaque exfiltration	19/20	0/19	0/19	20/20	0/20	0/20
Staged exfiltration	19/20	19/19	19/19	20/20	20/20	20/20
Held-out exfiltration	0/20	–	–	0/20	–	–
Total	98/140	39/98	59/98	99/140	39/99	59/99

A.18 Summary of the corrected empirical story

Table 11 collects the questions this project asked and what the traces answered. Every figure in it appears in the body; it is reproduced here as a single view of which claims survived.

Table 11: Summary of the corrected empirical story. Rates are reported only for the evaluated traces, and the two epistemic classes — live and scripted — are never pooled.

Question	Evidence from experiments	Interpretation
Can history separate constructed twins?	80–96%	Yes, in controlled cases
Does it beat a stateless rule on live attacks?	No	Claim not supported
Does it beat a stateless rule on AgentDojo replay?	52/109 vs 51/109	No; reproduced external result
Does AgentDojo's native tool filter reduce achieved attacks?	17/160 vs 109/160	Yes on released runs; not our controller
Does it beat scoped delegation?	default 3/25; calibrated max 13/25; scoped 17/25	Default loses; calibrated rows do not separate
Does it beat no controller at all?	default 3/25; retrospective max 13/25; no controller 0/25	Default not separable; the selected maximum is descriptive, not a confirmatory comparison
Can the evaluated policies block opaque effects?	0/12 live; 0/14 at 5 boundaries; 0/2 in each new domain	Live failure includes projection loss; authored schemas demonstrate observability limits
Do byte-identical views with differing effects occur?	2 authored collisions, decided identically by all 9 conditions	Lemma 1's mechanism appears in controls
What does the effect oracle establish?	oracle 25/25 at 9.7% FP	Near-definitional effect-based check, not a policy optimum
Is the false-positive cost tunable away?	0 of 80 cells under 5%	Structural
Do simpler runtime controls recover it?	external policy 13/25; taint 3/25; rate limit 3/25	No; narrow memory does not help
Does any condition enforce authority?	1 of 10, under a counterfactual check	Only scoped delegation
Does a path hint rescue an opaque call?	28/28 or 19/28, by scope reading	Contingent on resource naming
Can complete records reconstruct real incidents?	13/13 core questions	Evidence claim supported
Can action-only logs reconstruct them?	0/13	Insufficient for attribution
Do blinded evaluators abstain when evidence is cut?	33/40 answered wrongly	They confabulate instead
Does a larger single-model sample reproduce the ordering?	0/16 vs 8/16 prevented	Yes descriptively within one scenario; trajectory still provides no prevention benefit