BEFORE MEMORY STEERS: SEPARATING TRUST TIMING FROM DESCENDANT REPAIR IN PERSISTENT AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A poisoned write can steer an agent even when it is rejected before the target-query planner sees it: while resident in shared memory, it can be consolidated into descendants that remain reusable after the root is hidden. We isolate this pre-query authority with a repair-matched 2×2 intervention that moves one frozen trust signal across the transition into actionable memory. On PoisonBench-RAG, early rather than late binding lowers attack success by 8.9 points without descendant repair and 8.2 points with repair, for an 8.6 [7.1,10.1]-point timing main effect; repair contributes 4.2 [3.1,5.3] points. *ActionLatch* enforces this lifecycle through raw-only writes, fail-closed promotion, lineage-monotone trust, version-bound verifier records, and authenticated closure revocation. Relative to naive shared memory, the complete policy lowers PoisonBench-RAG ASR from 34.2% to 6.9%, carried-state SWE-bench Lite malicious-patch acceptance from 19.2% to 4.0%, and DocQA-Cite cited false claims from 27.8% to 6.7%, while DocQA coverage changes from 94.4% to 91.6%. Frozen verifier-aware attacks and a Llama-3.1-70B replication preserve policy ordering; at the reported PoisonBench operating points, ActionLatch pairs 6.9% ASR with 87.6% clean pass, versus 5.8% and 82.6% for isolation-first. These results make the time at which memory acquires reusable authority a measurable systems-security decision, distinct from recovery after compromise.

1 INTRODUCTION

Persistent agent memory exposes a state-transition failure that retrieval-time filtering cannot undo. In the later-binding system studied here, a patch recipe r enters shared memory while the agent handles a source issue, and a pre-query consolidation step derives a reusable summary c. When a different target issue arrives, the frozen trust signal rejects r, so the planner never sees the root; without lineage repair, c can still reproduce the injected objective after the source discussion has disappeared. The same chronology applies to summarized passages, citation notes, and cached tool outputs. Security therefore depends not only on whether a write is eventually rejected, but on whether it was allowed to create reusable state first.

Persistent memory extends an agent policy through time (Guu et al., 2020; Lewis et al., 2020; Mialon et al., 2023; Schick et al., 2023; Yao et al., 2023; Shinn et al., 2023; Packer et al., 2023; Park et al., 2023; Wang et al., 2024; Sarch et al., 2023). Memory and RAG attacks exploit this write-to-reuse path (Chen et al., 2024b; Zou et al., 2025; Dong et al., 2025; Srivastava & He, 2025; Xu et al., 2026), allowing indirect prompt injection to outlive its source interaction (Greshake et al., 2023; Kang et al., 2024; Liu et al., 2024; Debenedetti et al., 2024). Retrieval filters, split trust tiers, dispatch gates, and descendant rollback all reduce exposure, but they govern different authority objects. Their package-level rankings leave a specific systems question unanswered: how much risk is introduced during the interval between admitting a write and applying the same trust decision at its first target use?

We answer with the *actionability gap*, a paired risk difference induced by moving one immutable verifier tuple across the transition into reusable authority. Model, retriever, prompts, tools, candidate order, backfill, topology, units, seeds, and verifier outcomes remain fixed. Early binding keeps a failed

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.

write outside retrieval and derivation. Later binding admits it to shared actionable memory, permits the scheduled pre-query consolidation step, and applies the identical score when the root is selected for the target query; the rejected root remains hidden from the planner, isolating descendant-mediated influence. An orthogonal repair factor either preserves or revokes the root's authenticated closure before endpoint scoring.

ActionLatch realizes the early-binding policy as a raw/actionable/revoked state machine. Writes receive retrieval and derivation privileges only after fail-closed verification; authenticated parent trust upper-bounds child trust; verifier records bind decisions to fragment versions; and a post-admission compromise revokes the complete authenticated descendant closure. The PoisonBench factorial assigns 8.6 [7.1,10.1] points to the timing main effect and 4.2 [3.1,5.3] points to repair. The timing simple effects are 8.9 points without repair and 8.2 with repair.

The complete policy tests these mechanisms across three delayed authority channels. Against the same score bound later without repair, the paired full-policy reductions are 12.7 [10.3,15.0], 7.7 [5.8,9.6], and 9.7 [7.4,12.0] points. Component removals identify poisoned entry, transitive reuse, correlated verifier passes, and descendant persistence; frozen verifier-aware attacks and Qwen2.5/Llama-3.1-70B stacks preserve the full-policy ordering. At the reported PoisonBench operating points, isolation-first reaches 5.8% ASR but 82.6% clean pass and $+15.3\%$ p95 overhead, whereas ActionLatch reaches 6.9%, 87.6%, and $+9.8\%$.

The central contribution is an experimentally identified security quantity—pre-query reusable authority—rather than another detector score. The matched factorial distinguishes when trust binds from how compromise is repaired; ActionLatch supplies an enforceable lifecycle for that distinction; and the evaluation connects the mechanism to terminal outcomes, direct utility, adaptive pressure, cross-stack transfer, and closure cost.

2 Related Work and Design Contrast

Persistent-memory attacks establish that adversarial state can survive its source interaction and later re-enter control flow (Chen et al., 2024b; Zou et al., 2025; Dong et al., 2025; Srivastava & He, 2025; Xu et al., 2026). PromptArmor, MELON, AgentSentry, DataSentinel, and StruQ intervene on content presented near retrieval or use (Shi et al., 2025; Zhu et al., 2025; Zhang et al., 2026; Liu et al., 2025; Chen et al., 2025); information-flow and hierarchical-memory systems instead restrict which trust tier can steer planning (Wu et al., 2024; Wen et al., 2026). MemLineage authenticates ancestry and blocks sensitive dispatch when the active justification descends from untrusted memory (Ouyang & Hou, 2026). These systems protect retrieved items, planning tiers, or dispatched actions. Our protected object is earlier: the stored fragment's privilege to participate in retrieval and parent further actionable state.

The matched experiment replays the same version-bound score, candidate order, shared topology, and target-query backfill, moving only the score's position relative to actionable derivation and crossing that intervention with closure repair. Authenticated provenance, taint propagation, and rollback are established mechanisms (Buneman et al., 2001; Cheney et al., 2009; Muniswamy-Reddy et al., 2006; Newsome & Song, 2005; Schwartz et al., 2010); ActionLatch composes them into the instrument that enforces this timing intervention. The detector/scrubber, MemLineage-style, and isolation-first rows are controlled mechanism comparisons that locate different protected authority objects.

Table 1 locates each decision boundary. Only the repair-matched early/late pair identifies trust timing; Sec. 5 uses the remaining rows to place the resulting policy on a security–utility–latency frontier.

3 Trust Timing in the Memory Data Path

3.1 Actionable State and the Timing Estimand

At time t, every fragment occupies one state $z_t(m) \in \{\text{raw}, \text{actionable}, \text{revoked}\}$. Raw fragments are visible to verifiers but receive no authority privileges. Actionable fragments may compete for retrieval, enter planner context, and parent new actionable fragments. Revoked fragments remain auditable but lose all three privileges. This state machine makes "trusted memory" an enforced transition rather than a score attached to an otherwise reusable item.

Table 1: Authority objects in persistent-memory defense. Only the repair-matched ActionLatch intervention holds signal and topology fixed; the other rows describe neighboring control mechanisms.

Design	Authority object	Bind	Topology / repair	Identified question
Detector / scrubber	Retrieved item	Retrieval / use	Shared; purify or reject item	Can late filtering contain exposure?
Isolation / IFC	Planning tier	Tier crossing	Split; gate cross-tier reads	Which tier may steer planning?
MemLineage style	Dispatch justification	Action dispatch	Shared DAG; block action	Does ancestry authorize dispatch?
ActionLatch	Fragment and closure	Promotion	Shared; revoke closure	What risk follows deferred authority?

For the estimand, i indexes a paired unit–seed execution, and Y_i^L and Y_i^E denote its binary attack endpoint under later and earlier binding of the same signal. We define

$$G_{\mathrm{act}} = \mathbb{E}_i\left[Y_i^L - Y_i^E\right], \tag{1}$$

the *actionability gap*. A positive G_{act} measures risk attributable to granting reusable authority before the trust decision binds. Repair status is part of the treatment index: $G_{\mathrm{act}}(r)$ compares early and late binding at the same $r \in \{\mathrm{off}, \mathrm{on}\}$, and the timing main effect averages those repair-matched contrasts. We reserve *full-policy effect* for the later/no-repair versus ActionLatch (early/repair) difference.

3.2 What changes between the matched arms

Every paired unit follows the same event clock: restore the pre-source snapshot; ingest one attacked or paired-clean root; bind its immutable verifier tuple; execute the workload's frozen source-stage write and consolidation window; freeze memory; issue one target interaction; apply the target-query decision; optionally repair; then score the terminal endpoint. No target answer, reference-check result, patch-acceptance label, or citation-correctness label is available to promotion, rejection, or repair. This order separates runtime evidence that can legitimately trigger policy from the post hoc labels used to evaluate it.

The early and late executions replay the same tuple, root-candidate ranking, consolidation calls, and scan-and-backfill rule: rejected roots are skipped until K authorized items are found or the ranked list is exhausted. A rejected root is never shown to the target-query planner in either arm. Early binding keeps the write raw throughout the source-stage window, so it cannot enter the retrieval index or parent an actionable summary. Later binding inserts the root into shared actionable memory before that window; the fixed consolidator can retrieve it and create authenticated descendants before the target query applies the score. Those treatment-induced descendants remain eligible when repair is off. When repair is on, a preterminal rejection or independently scheduled compromise signal revokes the root's authenticated closure before endpoint scoring. Planner exposure to the rejected root, target-query backfill, verifier evidence, and budgets are consequently matched; only pre-check authority and the crossed repair factor change.

3.3 Adversary and objective

The evaluated attacker controls web passages, retrieved documents, package mirrors, issue discussions, or tool outputs that the agent may store. It seeks to preserve injected instructions, corrupt facts or citations, carry malicious patch templates into later issues, or create repeated failure through descendant reuse. We distinguish *verifier-agnostic*, *suite-aware*, and *blind-spot-targeted* templates. Suite-aware templates know the verifier families; blind-spot-targeted templates use failure patterns selected on a disjoint development split. All templates and their workload allocation are frozen before evaluation, and no held-out unit is queried during adaptation.

The trusted runtime authenticates writer, edge, and verifier records and enforces promotion and rollback on the critical path. The attacker can target verifier blind spots and content channels but cannot forge runtime keys or bypass the state transition. Success is measured at the authority channel's terminal consequence: a poisoned answer, a malicious patch that survives the apply gate, or

an incorrect claim paired with a misleading citation. Each ASR uses the complete held-out target denominator, so refusal or abstention is not removed from evaluation. Security is interpreted with clean pass for PoisonBench, attacked and clean resolve for SWE, attacked citation correctness and coverage for DocQA, false quarantine, and tail latency.

3.4 ACTIONLATCH INVARIANT

All writes enter a raw store available to verifiers; planners retrieve only from the actionable store. For fragment m, let $s_{\text{ver}}(m) = +1$ exactly when every channel-required verifier passes on the current fragment version, and -1 otherwise. Let $s_{\text{con}} = 1 - c(m)$, where $c(m)$ is the largest calibrated contradiction confidence against stronger evidence, and let s_{anom} be the largest calibrated anomaly score emitted by the applicable checks. Parent hashes and this version-bound tuple define

$$t(m) = \min\Big(\min_{p \in \text{Parents}(m)} t(p), \; \sigma(\alpha s_{\text{ver}}(m) + \beta s_{\text{con}} - \gamma s_{\text{anom}}) \Big), \quad (2)$$

where $s_{\text{con}}, s_{\text{anom}} \in [0, 1]$, σ is logistic, and $\alpha = 1.8, \beta = 1.2, \gamma = 1.0$. SWE fragments require compile/smoke, targeted-unit, and format checks; citation notes require entailment and format checks; structured tool outputs require replay and format checks. Passage checks combine entailment, contradiction, and anomaly outputs. The coefficients, calibrations, verifier versions, and channel assignment are frozen before evaluation. The outer minimum enforces lineage monotonicity: local evidence cannot wash a low-trust parent into actionable memory.

For a root fragment, the empty parent minimum is defined as 1. Promotion requires $s_{\text{ver}}(m) = +1$, $t(m) \geq \tau_{\text{promote}}$ (0.75 by default), and no contradiction from stronger evidence. A timeout, missing record, invalid signature, or verifier disagreement is inconclusive: early binding leaves the fragment raw, and later binding rejects it for that retrieval. Two fragments contradict when passage entailment returns `contradict` with confidence at least 0.85 or overlapping tool checks return opposing outcomes. Re-promotion requires clean parents, a resolved root cause, and verification under the current suite.

Each edge binds $\langle H(m_{\text{parent}}), H(m_{\text{child}}), \text{sig}_{\text{writer}} \rangle$; verifier records bind the fragment hash, verifier identity and version, result, and freshness. Let $\mathcal{A}_t$ be the actionable set and B_t newly compromised roots. We write $\text{Desc}^*(B)$ for the reflexive authenticated descendant closure, including the roots in B. ActionLatch enforces

$$m \in \mathcal{A}_t \Rightarrow s_{\text{ver}}(m) = +1 \;\wedge\; t(m) \geq \tau_{\text{promote}} \;\wedge\; \text{Parents}(m) \subseteq \mathcal{A}_t, \quad (3)$$

$$b \in B_t \Rightarrow \text{Desc}^*(b) \cap \mathcal{A}_{t^+} = \emptyset. \quad (4)$$

An adjacency index computes this closure, removes it from actionable retrieval, and retains its raw records. Promotion and revocation commit against a fragment-version epoch: a stale verifier record cannot promote a newer version, and a concurrent revocation wins over promotion. A root enters B_t after a current verifier rejects it, stronger evidence crosses the contradiction threshold, or a prescheduled authenticated incident record identifies compromise. These triggers use only records available before the terminal outcome; benchmark references and endpoint labels are evaluator-only. Repair commits before the endpoint for that evaluation step is scored. Equations 3–4 therefore govern both authority acquisition and already-propagated influence.

Algorithm 1 ActionLatch promotion and descendant repair

```
Store m raw-only; authenticate parents and record version epoch e_m
Run channel-required verifiers; bind their tuple and t(m) to H(m), e_m
if s_ver(m) = +1 and t(m) ≥ τ_promote and no stronger contradiction then
    Atomically promote m if e_m is current and no revocation is pending
else
    Keep m non-actionable
end if
On roots B, atomically quarantine Desc*(B) before scoring the endpoint
```

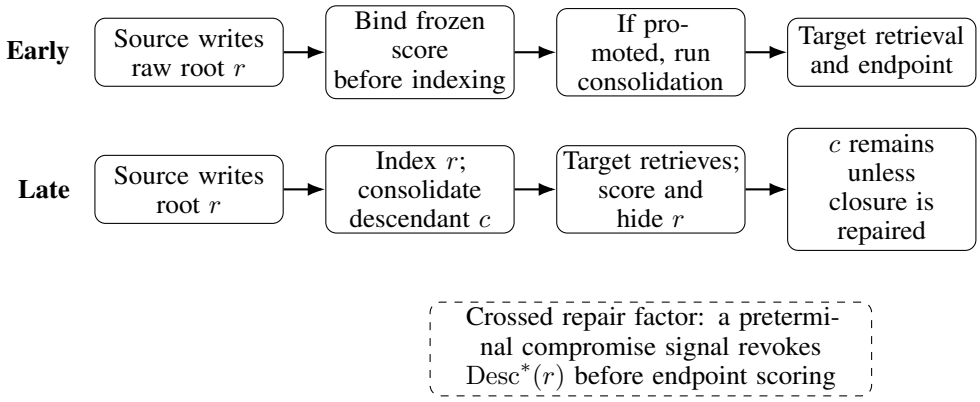


Figure 1: One-root event trace for the matched intervention. Both arms replay the same root score, consolidation schedule, and target-query backfill. The late lane can create c before rejecting r; the crossed repair factor determines whether that closure remains actionable.

4 Experimental Protocol

4.1 Three authority channels

Each workload instantiates the same causal skeleton—source write, scheduled consolidation window, memory freeze, and later target use—with a different authority channel; the complete event and adjudication map appears in App. A, Table 8. PoisonBench-RAG contains 500 held-out source–target interaction pairs: an attacker-controlled passage or structured tool output is available only in the source episode, can be summarized during memory maintenance, and is tested by the paired target interaction. SWE-bench Lite contributes 300 held-out target issues (Jimenez et al., 2024); each receives a source issue carrying a patch template, after which memory is frozen and the source is removed before the target issue begins. DocQA-Cite contains 500 held-out questions, each paired with a lexically plausible but non-entailing claim–citation note stored before answering. Attack-template selection, thresholds, calibration, timeout choices, and target units occupy disjoint development and evaluation splits.

Every base unit has attacked and paired-clean executions under every seed, and changing a policy arm restores the exact pre-source snapshot. Each endpoint uses its full workload denominator: PoisonBench ASR requires poisoned-lineage use and target-reference failure; SWE malicious-patch acceptance requires the injected objective, a valid diff, successful application, and no quarantine; and DocQA ASR requires an incorrect claim with its misleading citation. Endpoint adjudicators run only after policy decisions. Clean pass, resolve, citation correctness, coverage, false quarantine, containment, and load-test ASR retain their full denominators, including abstentions and rate-limited writes.

4.2 Matched stack and uncertainty

Qwen2.5-32B-Instruct (temperature 0.2, top-p 0.9) (Yang et al., 2024), `bge-large-en-v1.5`, and `bge-reranker-large` (Xiao et al., 2024) form the primary stack. Runs use PyTorch 2.3.1, CUDA 12.4, Ubuntu 22.04, and four NVIDIA A100 80 GB GPUs. Every primary comparator shares prompts, tools, source and target units, root candidates, candidate order, top-K, context and tool-call caps, verifier timeouts, attack instance and source-stage write budget, consolidation schedule, and seeds $\{13, 37, 61, 71, 97\}$. This yields 2,500 PoisonBench executions, 1,500 SWE executions, and 2,500 DocQA executions per condition.

The primary causal controls differ only at the named lifecycle factors. Naive shared memory indexes every write. Keyword filtering rejects a development-frozen pattern list before indexing. Trust-score later binding first permits shared indexing and source-stage derivation, then replays the ActionLatch tuple when a root is retrieved, scanning past rejected roots to backfill K; it exposes no rejected root

to the target planner and performs neither promotion nor closure repair. Early/no-repair changes only score placement, later/repair changes only closure revocation, and ActionLatch activates both.

Three additional rows are controlled mechanism instantiations at development-selected operating points. The detector/scrubber applies a separately frozen content detector and purification pass to each retrieved item, authorizing only the resulting detector-passed item while retaining shared memory and no closure repair. The MemLineage-style control authenticates ancestry and blocks tool dispatch whose active justification descends from untrusted memory, while leaving write admission and non-tool planning unchanged. Isolation-first routes attacker-reachable writes to a separate tier and admits a cross-tier read only after its frozen corroboration check, preventing those writes from joining shared derivation beforehand. All three use the same evaluator-call cap, candidate exposure, source-stage attack budget, and clean/attack reset policy. They test neighboring authority objects rather than the timing estimand; their overhead is p95 end-to-end clean latency relative to naive shared memory.

Tables report the mean and standard deviation of five seed-level rates. Each of 10,000 paired-bootstrap draws first resamples base units while retaining every arm and all five seed outcomes for each sampled unit, then resamples seed columns; this preserves arm pairing and within-unit cross-seed dependence. Percentile 95% intervals are reported for the claim-bearing full-policy and factorial contrasts. Comparisons shown only as means and standard deviations are interpreted as operating points rather than inferential rankings. The adaptive writer receives verifier-family identities and development-split failures, is frozen before evaluation, and never probes the 160 PoisonBench or 200 DocQA held-out units.

5 Results

5.1 The factorial estimates crossed timing and repair effects

Table 2: PoisonBench 2×2 isolation with one trust signal. Early-minus-late ASR reductions are 8.9 points without repair and 8.2 with repair; the final column records clean blocks or flags at each binding point.

Binding / repair	ASR (%)	Clean pass (%)	Clean block/flag (%)
Later / none	19.6 ± 2.6	88.0 ± 1.5	3.4 ± 0.8
Early / none	10.7 ± 1.6	87.9 ± 1.6	3.6 ± 0.7
Later / repair	15.1 ± 1.9	87.8 ± 1.5	3.7 ± 0.8
Early / repair	6.9 ± 1.1	87.6 ± 1.6	3.9 ± 0.8

Without repair, moving the score before actionable derivation lowers ASR from 19.6% to 10.7%, an 8.9-point simple effect. With repair held on, the corresponding 15.1%-to-6.9% change is 8.2 points. Averaging the two repair-matched contrasts gives the timing main effect, 8.6 [7.1,10.1] points; averaging the repair contrasts gives 4.2 [3.1,5.3] points. The timing-by-repair interaction is -0.7 [-1.9,0.6]: no interaction is resolved at this precision, while both main-effect intervals exclude zero. The two manipulated pathways are therefore empirically distinguishable without asserting interaction equivalence.

The component study in App. B locates the same lifecycle pathways. Removing promotion produces the largest increase, from 6.9% to 22.1% ASR; removing taint propagation raises ASR to 13.8%, and removing rollback reproduces the 10.7% early/no-repair cell. The ablation and factorial thus agree that admission, transitive derivation, and post-compromise persistence each carry measurable risk.

5.2 The full lifecycle suppresses delayed reuse in three channels

The complete path removes roughly three-fifths to two-thirds of attacks surviving the same later-bound score, with every paired full-policy interval excluding zero. These are lifecycle-policy effects; Table 2 supplies the timing-only estimate. Direct utility moves in different directions by channel:

Table 3: Matched full-policy comparisons under one trust signal. The later arm has no promotion or repair; ActionLatch uses early promotion and closure repair. The PoisonBench factorial in Table 2 identifies timing separately.

Authority channel	Attack endpoint	Later / no repair	Early / repair	Full-policy reduction (points)
Retrieved evidence	Poisoned answer	19.6 ± 2.6	6.9 ± 1.1	12.7 [10.3,15.0]
Carried patch template	Malicious patch	11.7 ± 1.8	4.0 ± 0.9	7.7 [5.8,9.6]
Citation-bearing note	Cited false claim	16.4 ± 2.1	6.7 ± 1.1	9.7 [7.4,12.0]

PoisonBench clean pass changes from 88.0 ± 1.5 to 87.6 ± 1.6; attacked SWE resolve rises from 16.7 ± 1.0 to 19.6 ± 1.0; and attacked DocQA coverage changes from 93.2 ± 1.1 to 91.6 ± 1.2.

Table 4: End-to-end security and direct utility. PoisonBench pairs attacked ASR with clean-condition pass; SWE reports clean or attacked resolve as labeled; DocQA reports attacked-condition citation correctness and coverage.

PoisonBench method	ASR (%)	Clean pass (%)	Full-policy reduction (points)
Naive shared	34.2 ± 3.4	88.4 ± 1.4	—
Keyword filter	25.1 ± 2.8	86.8 ± 1.7	—
Trust-score later	19.6 ± 2.6	88.0 ± 1.5	12.7 [10.3,15.0]
ActionLatch	6.9 ± 1.1	87.6 ± 1.6	—
SWE setting	**Resolve (%)**	**Malicious patch (%)**	**Wall time (min)**
Clean shared reference	21.2 ± 0.9	0.8 ± 0.3	18.4 [15.6,22.6]
Clean ActionLatch	20.4 ± 0.8	1.0 ± 0.3	20.1 [17.5,24.0]
Attacked naive	11.6 ± 1.0	19.2 ± 2.9	21.8 [18.4,26.4]
Attacked trust-score	16.7 ± 1.0	11.7 ± 1.8	20.7 [17.8,24.9]
Attacked ActionLatch	19.6 ± 1.0	4.0 ± 0.9	19.8 [17.2,23.6]
Attacked DocQA method	**ASR (%)**	**Citation correct (%)**	**Coverage (%)**
Naive shared	27.8 ± 2.9	68.6 ± 2.4	94.4 ± 1.0
Trust-score later	16.4 ± 2.1	75.2 ± 2.0	93.2 ± 1.1
ActionLatch	6.7 ± 1.1	80.6 ± 1.7	91.6 ± 1.2

Table 4 locates the consequence in each workflow. On PoisonBench, the complete policy removes 64.8% of attacks remaining under the identical later-bound score for a 0.4-point clean-pass change. On SWE, ActionLatch closes 7.7 points of the carried-template path while restoring 2.9 resolve points. On DocQA, cited false claims fall by 9.7 points and citation correctness rises by 5.4 points while coverage falls by 1.6 points. The SWE and DocQA utility measures show that these security improvements are not explained by abstention alone.

5.3 Neighboring operating points and cross-stack replication

At these frozen operating points, isolation-first records 1.1 points lower ASR than ActionLatch together with 5.0 points lower clean pass and 5.5 points more p95 overhead. The MemLineage-style point has 2.2 points higher ASR, 2.4 points lower clean pass, and 3.6 points more overhead than ActionLatch. Because these rows change protected object and mechanism, the differences describe the displayed frontier rather than isolated treatment effects. ActionLatch occupies the point that combines near-isolation ASR with substantially higher clean pass and lower tail cost.

The Llama-3.1-70B-Instruct replication (Grattafiori et al., 2024) shifts absolute ASR upward but preserves a 12.8-point trust-score-to-ActionLatch full-policy difference. Qwen2.5 with `bge-m3` (Chen

Table 5: Controlled PoisonBench-RAG mechanism operating points on 500 units and five seeds. Neighboring rows protect different authority objects and are not timing counterfactuals.

Controlled mechanism	ASR (%)	Clean pass (%)	p95 overhead / protected object
Naive shared	34.2 ± 3.4	88.4 ± 1.4	reference
Trust-score later	19.6 ± 2.6	88.0 ± 1.5	score after retrieval
Detector / scrubber	13.8 ± 1.9	86.8 ± 1.6	+11.6%; retrieved item
MemLineage-style gate	9.1 ± 1.4	85.2 ± 1.8	+13.4%; tool dispatch
Isolation-first	5.8 ± 0.9	82.6 ± 1.9	+15.3%; trust tier
ActionLatch	6.9 ± 1.1	87.6 ± 1.6	+9.8%; write + closure

Table 6: Full-policy ordering across model and retrieval stacks. The two transfer rows share 250 units; the primary-BGE row is the 500-unit main-workload anchor.

Stack	Naive ASR (%)	Trust-score ASR (%)	ActionLatch ASR / clean pass (%)
Qwen2.5 + primary BGE	34.2 ± 3.4	19.6 ± 2.6	6.9 ± 1.1 / 87.6 ± 1.6
Qwen2.5 + bge-m3	31.6 ± 3.1	18.1 ± 2.4	8.4 ± 1.2 / 86.4 ± 1.5
Llama-3.1-70B + bge-m3	36.8 ± 3.5	22.4 ± 2.7	9.6 ± 1.5 / 84.8 ± 1.7

et al., 2024a) retains a 9.7-point difference. Both changes preserve the naive > later-score > ActionLatch ordering despite different absolute attack rates.

5.4 Full-policy separation survives verifier-aware adaptation

On 160 held-out PoisonBench units, a frozen suite-aware writer raises trust-score ASR to 24.8 ± 2.7, while ActionLatch holds it to 11.2 ± 1.6; their p95 clean-latency overheads are +3.8% and +6.2%. On 200 disjoint adaptive DocQA questions, later binding reaches 22.5 ± 2.6 ASR versus 9.5 ± 1.4 for ActionLatch, which retains 78.5 ± 1.9 citation correctness and 90.5 ± 1.4 coverage. Because the attack templates are frozen before these units are exposed, these results stress the complete promotion-and-repair policy without extending the timing-only estimand beyond the PoisonBench factorial.

5.5 Correlated misses make lifecycle repair consequential

Table 7: Version-locked verifier pairs exhibit correlated held-out misses. Intervals use 10,000 paired bootstrap draws.

Verifier pair	Held-out adversarial cases	Product FNR	Observed joint FNR
SWE compile + unit	320	0.010	0.031 [0.016,0.049]
Citation NLI + format	400	0.003	0.018 [0.008,0.032]
SQL replay + format	360	0.001	0.009 [0.003,0.020]

Observed joint FNR is 3.1–9.0 times the independent-product heuristic across verifier pairs. Diverse checks therefore retain shared blind spots, motivating authenticated revocation after a joint miss. In the separate factorial intervention, closure repair yields a 4.2 [3.1,5.3] point main effect.

6 DISCUSSION

The 8.6-point PoisonBench timing main effect measures the cost of allowing a root to participate in the scheduled consolidation window before its frozen score binds. The 4.2-point repair main effect measures the value of removing that root and its authenticated descendants after a preterminal compromise signal. The unresolved interaction supports separate estimates rather than equivalence or strict additivity. The 12.7-, 7.7-, and 9.7-point values compare complete later/no-repair and early/repair policies across the three channels.

Memory admission is therefore an authority decision, not merely storage. Within the evaluated shared-memory regime, the factorial supplies the timing estimate, while SWE and DocQA test the complete lifecycle on carried patch and citation authority; adaptive and cross-stack studies test policy ordering. Full-denominator security outcomes are accompanied by utility and repair-cost measures, making early promotion and descendant repair actionable interface choices.

7 CONCLUSION

A root rejected at target retrieval can already have transferred influence to reusable descendants. Moving the same trust signal ahead of that transfer reduces PoisonBench ASR by 8.9 points without repair and 8.2 with repair; authenticated closure revocation contributes a 4.2-point repair main effect. Across retrieved evidence, carried patches, and cited claims, ActionLatch reaches 6.9%, 4.0%, and 6.7% attack outcomes. The results support a design rule for this shared-memory regime: bind trust before memory can retrieve, derive, or steer, and retain authenticated lineage for recovery after a verifier miss.

AI USE STATEMENT

Generative AI tools assisted manuscript preparation, including copyediting, organization, formatting, and checking reference metadata. The final-content responsibility required by the ICLR AI-use policy remains with the authors.

REFERENCES

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *Database Theory—ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pp. 316–330. Springer, 2001. doi: 10.1007/3-540-44503-X_20. URL https://doi.org/10.1007/3-540-44503-X_20.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2383–2400, Seattle, WA, 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/chen-sizhe.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130185–130213, 2024b. doi: 10.52202/079017-4136. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb113910e9c3f6242541c1652e30dfd6-Abstract-Conference.html.

James Cheney, Laura Chiticariu, and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009. doi: 10.1561/1900000006. URL https://doi.org/10.1561/1900000006.

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*, volume 37, pp. 82895–82920, 2024. doi: 10.52202/079017-2636. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen J. Xiang. Memory injection attacks on LLM agents via query-only interaction. In *Advances in Neural Information Processing Systems*, volume 38, pp. 52119–52153, 2025. doi: 10.52202/085713-1554. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/42a97bbd9844d2bf68596730af80bcdf-Abstract-Conference.html.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783. URL https://arxiv.org/abs/2407.21783.

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. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 3929–3938. PMLR, 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

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*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. Exploiting programmatic behavior of LLMs: Dual-use through standard security attacks. In *2024 IEEE Security and Privacy Workshops (SPW)*, pp. 132–143, 2024. doi: 10.1109/SPW63631.2024.00018. URL https://doi.org/10.1109/SPW63631.2024.00018.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 1831–1847. USENIX Association, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei.

Yupei Liu, Yuqi Jia, Jinyuan Jia, Dawn Song, and Neil Zhenqiang Gong. DataSentinel: A game-theoretic detection of prompt injection attacks. In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 2190–2208. IEEE, 2025. doi: 10.1109/SP61157.2025.00250. URL https://ieeexplore.ieee.org/document/11023476.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: A survey. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=jh7wH2AzKK.

Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer. Provenance-aware storage systems. In *2006 USENIX Annual Technical Conference*, pp. 43–56. USENIX Association, 2006. URL https://www.usenix.org/conference/2006-usenix-annual-technical-conference/provenance-aware-storage-systems.

James Newsome and Dawn Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *12th Annual Network and Distributed System Security Symposium*, 2005. URL https://doi.org/10.1184/r1/6468716.

Ciyan Ouyang and Rui Hou. MemLineage: Lineage-guided enforcement for LLM agent memory. *arXiv preprint arXiv:2605.14421*, 2026. doi: 10.48550/arXiv.2605.14421. URL https://arxiv.org/abs/2605.14421.

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. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Gabriel Sarch, Yue Wu, Michael J. Tarr, and Katerina Fragkiadaki. Open-ended instructable embodied agents with memory-augmented large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3468–3500. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.226. URL https://aclanthology.org/2023.findings-emnlp.226/.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Edward J. Schwartz, Thanassis Avgerinos, and David Brumley. All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In *IEEE Symposium on Security and Privacy*, pp. 317–331, 2010. doi: 10.1109/SP.2010.26. URL https://doi.org/10.1109/SP.2010.26.

Tianneng Shi, Kaijie Zhu, Zhun Wang, Yuqi Jia, Will Cai, Weida Liang, Haonan Wang, Hend Alzahrani, Joshua Lu, Kenji Kawaguchi, Basel Alomair, Xuandong Zhao, William Yang Wang, Neil Zhenqiang Gong, Wenbo Guo, and Dawn Song. PromptArmor: Simple yet effective prompt injection defenses. *arXiv preprint arXiv:2507.15219*, 2025. doi: 10.48550/arXiv.2507.15219. URL https://arxiv.org/abs/2507.15219.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Saksham Sahai Srivastava and Haoyu He. MemoryGraft: Persistent compromise of LLM agents via poisoned experience retrieval. *arXiv preprint arXiv:2512.16962*, 2025. doi: 10.48550/arXiv.2512.16962. URL https://arxiv.org/abs/2512.16962.

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*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Ruoyao Wen, Hao Li, Chaowei Xiao, and Ning Zhang. AgentSys: Secure and dynamic LLM agents through explicit hierarchical memory management. *arXiv preprint arXiv:2602.07398*, 2026. doi: 10.48550/arXiv.2602.07398. URL https://arxiv.org/abs/2602.07398.

Fangzhou Wu, Ethan Cecchetti, and Chaowei Xiao. System-level defense against indirect prompt injection attacks: An information flow control perspective. *arXiv preprint arXiv:2409.19091*, 2024. doi: 10.48550/arXiv.2409.19091. URL https://arxiv.org/abs/2409.19091.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 641–649. ACM, 2024. doi: 10.1145/3626772.3657878. URL https://doi.org/10.1145/3626772.3657878.

Zhenlin Xu, Xiaogang Zhu, Yu Yao, Minhui Xue, and Yiliao Song. From storage to steering: Memory control flow attacks on LLM agents. *arXiv preprint arXiv:2603.15125*, 2026. doi: 10.48550/arXiv.2603.15125. URL https://arxiv.org/abs/2603.15125.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115.

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. URL https://openreview.net/forum?id=WE_vluYUL-X.

Tian Zhang, Yiwei Xu, Juan Wang, Keyan Guo, Xiaoyang Xu, Bowen Xiao, Quanlong Guan, Jinlin Fan, Jiawei Liu, Zhiquan Liu, and Hongxin Hu. AgentSentry: Mitigating indirect prompt injection in LLM agents via temporal causal diagnostics and context purification. *arXiv preprint arXiv:2602.22724*, 2026. doi: 10.48550/arXiv.2602.22724. URL https://arxiv.org/abs/2602.22724.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. MELON: Provable defense against indirect prompt injection attacks in AI agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80310–80329. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhu25z.html.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3827–3844, Seattle, WA, 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag.

A Protocol and Estimation Details

Table 8: Workload, policy input, and evaluator-only endpoint for each authority channel. Every endpoint uses the complete held-out target denominator.

Channel / units	Source-to-target construction	Policy-visible object and checks	Evaluator-only terminal endpoint
PoisonBench-RAG / 500 pairs	A passage or structured tool output appears only in the source; one maintenance window precedes the paired target interaction	Passage: entailment, contradiction, anomaly; tool output: replay and format	Final answer fails the target reference check and uses poisoned lineage
SWE-bench Lite / 300 targets	A source issue carries a patch template; state freezes and the source is removed before its paired target issue	Patch template; compile/smoke, targeted units, and format	Injected objective, valid diff, successful apply, and no quarantine
DocQA-Cite / 500 questions	A plausible non-entailing cited note is stored before the paired question	Claim–citation note; entailment and format	Incorrect claim returned together with the misleading citation

The atomic split unit is the complete source–target pair: neither member of an evaluation pair appears in attack selection, calibration, or threshold development. Endpoint adjudication composes workload-native records rather than the promotion score. PoisonBench joins authenticated lineage use with the target reference checker; SWE joins the predeclared injected-objective match, diff parsing, application result, and quarantine log, while benchmark tests determine resolve; DocQA joins answer correctness against the fixed question reference with the claim–citation relation. Thus the evaluator can require the attacked semantic outcome even when the runtime verifier passes, and its labels remain unavailable to policy decisions.

The executable event order is: (1) restore the unit's pre-source snapshot; (2) present one source-stage attacked or paired-benign root under the fixed write budget; (3) attach the frozen version-bound verifier tuple; (4) apply early or late admission; (5) run the unit's single scheduled maintenance window, during which authorized roots may parent authenticated descendants; (6) freeze memory and remove the source; (7) issue the paired target and apply target-query scoring and backfill; (8) apply the crossed preterminal repair signal, if enabled; and (9) score the endpoint. Steps 1–7 are identical in invocation count and order across timing arms. Step 8 receives verifier, contradiction, or authenticated incident evidence fixed independently of the terminal outcome.

For each unit, the attack template, source and target identities, candidate ranking, top-K, context and tool caps, verifier timeout, and maintenance calls are frozen before arm execution. The attack budget is exhausted at the source stage; the target cannot add or probe writes. Verifier-agnostic, suite-aware, and development-blind-spot templates are selected before the held-out run and retain their workload allocation across arms. Development units used for attack selection, calibration, thresholds, or timeout choices do not enter the evaluation set. The clean copy substitutes paired benign source content while preserving the target, stack, schedule, and random seed; every arm starts from the same pre-source snapshot.

The five primary seeds produce 2,500 executions for each 500-unit workload and 1,500 executions for the 300-issue SWE workload. The estimator first computes one rate per seed over the complete fixed split, then reports the five-rate mean and standard deviation. The clean PoisonBench per-seed pass counts are 433/500; 452/500; 438/500; 447/500; 440/500 for naive memory, 421/500; 447/500; 432/500; 437/500; 433/500 for keyword filtering, 432/500; 449/500; 440/500; 447/500; 432/500 for trust-score later binding, and 429/500; 447/500; 437/500; 446/500; 431/500 for ActionLatch. Other claim-bearing rates follow the same seed-macro estimator; we do not substitute a seed-collapsed unique-unit fraction for those means.

Table 9: Evidence hierarchy by protected object. The factorial identifies timing; the remaining rows test channel, stress, or systems consequences.

Authority object	Threat / trusted record	Identifying contrast or stress	Direct cost and trace
Retrieved fragment	Suite-aware write / signed score	Repair-matched timing factorial	87.6 ± 1.6 clean pass; Tables 2, 3
Patch template	Cross-issue carry / apply gate	Full policy after source removal	20.4 ± 0.8 clean resolve; Table 4
Citation note	Plausible false claim / entailment gate	Full policy and adaptive questions	91.6% coverage; Table 3
Actionable closure	Joint miss / authenticated DAG	Repair on/off and DAG attacks	Sub-second traversal; Table 2, App. E
Complete policy	Model and retriever variation	Qwen2.5 / Llama-3.1-70B	Stable ordering; Table 6

Table 10 reports shadow instrumentation available in every PoisonBench arm. "Benign flags" are detector observations for naive, keyword, and later-score memory; only ActionLatch turns the signal into executed quarantine, so the column diagnoses signal exposure rather than equating baseline flags with enforcement.

Table 10: Passive shared-audit diagnostics retained from the matched PoisonBench run. Baselines observe these signals but do not execute promotion or rollback.

Method	ASR (%)	Detection steps, median [p25,p75]	Benign flags (%)
Naive shared	34.2 ± 3.4	11.2 [7.8,16.4]	0.8 ± 0.3
Keyword filter	25.1 ± 2.8	9.8 [6.8,14.6]	5.1 ± 1.2
Trust-score later	19.6 ± 2.6	8.6 [6.4,12.8]	3.4 ± 0.8
ActionLatch	6.9 ± 1.1	4.4 [3.2,6.2]	3.9 ± 0.8

Table 11: Clean SWE condition and the original 120-issue adjacent-family slice.

Setting	Change beyond timing	n	Resolve (%)	Malicious (%) / time
Clean ActionLatch	attacker path absent	300	20.4 ± 0.8	1.0 ± 0.3; 20.1 [17.5,24.0]
Detector / scrubber	detector + use purification	120	18.2 ± 1.1	10.8 ± 1.7; 22.9 [19.4,27.3]
Isolation-first	topology split	120	15.0 ± 1.0	5.1 ± 0.8; 24.2 [20.8,29.0]

The adjacent-family SWE subset is retained as a mechanism check on 120 common issues. The complete 500-unit PoisonBench comparison in Table 5 is the primary neighboring-design analysis and uses the seed-macro estimator above.

B Mechanism and Verifier Analysis

The verified-poison slice asks which lifecycle component blocks an attack that has already crossed a shallow check. The component removal then maps each increase in ASR to the path it reopens, while the verifier and threshold tables expose the error and utility terms behind the selected operating point.

Table 12: Held-out verified-poison mechanism slice. Attack templates and verifier thresholds are frozen on disjoint development data.

Attack family	Gate only	+Taint	+Rollback	Full
Test-suite gaming	22.4 ± 2.6	17.9 ± 2.1	11.8 ± 1.6	7.6 ± 1.2
Schema-consistent	18.8 ± 2.2	13.9 ± 1.7	9.2 ± 1.3	5.4 ± 1.0
Citation manipulation	26.2 ± 3.0	20.1 ± 2.3	13.8 ± 1.8	8.8 ± 1.4
Shallow-pass reuse	20.6 ± 2.4	15.5 ± 1.8	10.6 ± 1.5	6.2 ± 1.1

Table 13: Component removal on PoisonBench-RAG at $\tau_{\text{promote}} = 0.75$.

Configuration	ASR (%)	False quarantine (%)	Reopened path
ActionLatch	6.9 ± 1.1	3.9 ± 0.8	none
Without promotion gate	22.1 ± 2.4	1.0 ± 0.3	poisoned entry
Without rollback	10.7 ± 1.6	3.6 ± 0.7	descendant persistence
Without taint propagation	13.8 ± 1.9	4.6 ± 1.0	transitive reuse
Without verifier diversity	9.8 ± 1.4	3.4 ± 0.7	correlated pass

Table 14: Verifier inventory and held-out error rates.

Verifier	FNR	FPR	p50	p95
Compile + smoke tests	0.12	0.03	8.6 s	27.4 s
Targeted unit subset	0.08	0.05	13.8 s	40.6 s
SQL constraint replay	0.06	0.02	0.68 s	2.04 s
Citation entailment	0.14	0.04	0.38 s	1.26 s
Format lint	0.02	0.01	0.12 s	0.34 s

Removing promotion produces the largest ablation increase, from 6.9% to 22.1% ASR; removing taint or rollback raises ASR to 13.8% or 10.7%. The verifier suite combines compile and smoke tests, targeted units, format lint, SQL replay, and citation entailment. Their respective p50/p95 costs are 8.6/27.4 s, 13.8/40.6 s, 0.12/0.34 s, 0.68/2.04 s, and 0.38/1.26 s. The checks use distinct implementations, harnesses, templates, and seeds.

Table 15: Promotion-threshold operating curve on PoisonBench-RAG. Utility drop is the clean-pass decrease from naive shared memory.

τ_{promote}	ASR (%)	False quarantine (%)	Utility drop (points)
0.65	9.8 ± 1.4	2.4 ± 0.6	0.3 ± 0.2
0.75	6.9 ± 1.1	3.9 ± 0.8	0.8 ± 0.3
0.85	5.1 ± 1.0	7.0 ± 1.3	1.6 ± 0.5

C Systems Behavior

The systems panels separate closure-computation scale from workload closure size, end-to-end repair stages, ingest pressure, and cumulative clean-memory behavior. This prevents a sub-second graph traversal from being conflated with total incident latency or long-horizon utility.

Table 16: Closure computation scales sub-second through one million nodes.

DAG nodes	Edges/node, median [p25,p75]	Closure p50 / p95	Index memory (GB)
50k	2.4 [1.8,3.6]	88 / 306 ms	0.21–0.38
200k	2.8 [2.0,4.4]	138 / 518 ms	0.58–1.08
1M	3.2 [2.2,5.8]	264 / 986 ms	2.36–4.18

The traversal benchmark isolates graph work from quarantine and incident-record writes. Increasing the graph from 50k to one million nodes raises p95 closure computation from 306 to 986 ms while the adjacency index grows to 2.36–4.18 GB.

Workload closures are much smaller than the synthetic scaling graphs but remain large enough to make root-only rejection incomplete. We therefore report closure size together with whether the full compromised closure is removed before its next actionable read.

Table 17: Containment and descendant-closure distributions.

Workload	Closure median [p25,p75]	p95 / p99	Containment (%)
PoisonBench-RAG	26 [13,58]	186 / 308	95.4 ± 1.0
SWE template poison	42 [21,86]	264 / 418	93.8 ± 1.4
DocQA claim poison	29 [15,64]	208 / 346	95.0 ± 1.1

Across the three workloads, median closures contain 26–42 fragments and containment remains 93.8–95.4%. These observed closures supply the propagated state targeted by the repair factor rather than treating descendant reuse as a purely hypothetical mechanism.

End-to-end repair includes three distinct stages. The following breakdown separates graph traversal from the index mutation and authenticated record write that complete a quarantine transaction.

Graph traversal is not the dominant tail in every workload: actionable-index quarantine and the incident record can each add hundreds of milliseconds. The measured stage ranges therefore explain why total rollback latency exceeds closure traversal alone.

Table 18: Rollback latency by stage.

Stage	p50	p95	Operation
Closure computation	118–258 ms	416–782 ms	DAG traversal
Actionable quarantine	82–176 ms	238–516 ms	batched index update
Incident record write	142–318 ms	524–978 ms	hashes and summary

The ingest-load experiment next exercises that full repair path while preserving the scheduled-opportunity ASR denominator. This denominator is important at the flood point, where rate limiting blocks writes before they can become actionable.

Table 19: Operational stability under sustained ingest load. The flood ASR falls below burst ASR because rate limiting admits fewer attacker writes while latency worsens.

Scenario	p50 / p95 overhead	Rollback p95 (s)	Queue p99	ASR (%)
Normal, 200 req/s	+4.2% / +9.8%	2.1	12	6.9
High, 1k req/s	+5.4% / +12.6%	2.8	28	7.2
Burst, 5k req/s	+7.2% / +17.8%	3.4	64	7.9
Flood, 10k req/s	+8.6% / +22.4%	4.8	rate-limited	7.4

Rollback remains below 4.8 s p95 through the measured flood condition, while p95 request overhead rises from 9.8% to 22.4%. Flood ASR is lower than burst ASR because the denominator includes scheduled writes rejected by rate limiting, as defined in App. A.

Finally, the clean-trajectory panel measures whether promotion costs accumulate with repeated memory use rather than only within one source–target pair.

Table 20: Longitudinal clean-memory behavior over 100 trajectories per seed.

Episodes	False quarantine (%)	Clean pass (%)	Actionable retention (%)	p95 overhead
10	3.7 ± 0.6	87.8 ± 1.5	98.2 ± 0.7	+8.7%
50	4.1 ± 0.7	86.9 ± 1.7	96.3 ± 0.9	+10.9%
100	3.9 ± 0.8	87.2 ± 1.6	95.1 ± 1.1	+12.4%

The longitudinal panel aggregates 100 clean trajectories per seed with no trajectory reused for tuning. Clean pass varies non-monotonically around 87%, actionable retention remains 95.1% after 100 episodes, and p95 overhead stays below 13%. Together, the threshold, closure, load, and horizon studies characterize security, retention, and tail-cost trade-offs at the selected operating point.

D Transfer and Adaptive Evaluation

The Qwen2.5+`bge-m3` and Llama-3.1-70B+`bge-m3` rows in Table 6 each use 250 fixed units and the five-seed macro estimator from App. A. They change the retriever and then the model while retaining the same three policy definitions.

The adaptive PoisonBench evaluation uses 160 held-out units and the adaptive DocQA evaluation uses 200 disjoint questions. Their templates are selected from development failures, frozen, and then replayed under later binding and ActionLatch; no held-out outcome feeds attack selection.

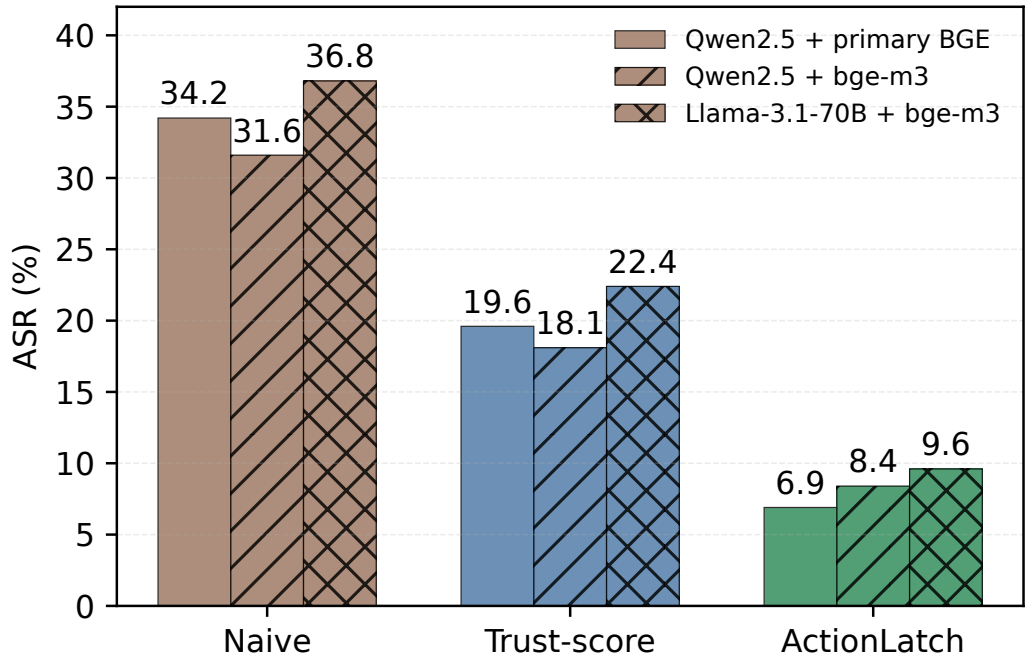


Figure 2: Full-policy ordering is stable across the primary stack, a second retriever harness, and a Llama-3.1-70B replication.

E AUTHENTICATED PROVENANCE

The closure operation relies on authentic writer edges and version-bound verifier outcomes. The following attacks alter those two record classes directly, complementing the content-channel attacks in the main evaluation.

Table 21: Like-for-like DAG-targeting attacks.

Attack	Authenticated ASR (%)	No-auth ASR (%)	Enforcement
Edge deletion	0.6 ± 0.2	7.1 ± 1.0	signed writer edges
Tool-log replay	0.9 ± 0.3	5.8 ± 0.9	hash- and version-bound outcomes

Table 22: Parent-spoofing results under a strongest aggregate stress point and a fixed seeded denominator; the rows use distinct estimands.

Protocol	Authenticated ASR	No-auth ASR	Denominator
Strongest aggregate stress point	1.2	8.4	point estimate
Fixed seeded subset	1.5 ± 0.4	8.8 ± 1.2	80 roots

The fixed subset aggregates five seeds into 400 provenance attempts, with 6/400 authenticated and 35/400 unauthenticated bypasses. Because every seed has the same 80-root denominator, the pooled proportions equal the seed-rate means before rounding: 1.5% and 8.75%, reported as 1.5±0.4 and

8.8±1.2 across seeds. The 1.2/8.4 row is instead a strongest-condition aggregate point estimate. Both protocols show that rollback soundness depends on authenticated parent and verifier records.

F LIFECYCLE TAXONOMY

Table 23: Attack lifecycle and dominant containment signal.

Attack type	Persistence mechanism	Containment signal
Document prompt injection	instruction stored as guidance	entailment and provenance
Tool-output spoofing	poisoned schema or JSON cached	tool hashes and failure spikes
SWE patch template	unsafe recipe reused across issues	compile/tests and rollback
Stale contradiction	obsolete guidance survives change	contradiction and re-verification
Cross-session verified poison	shallow checks accept harm	diverse checks and canaries

The rows connect each attack to the measured failure path. Retrieved passages instantiate pre-check derivation, carried SWE state preserves influence after the source disappears, cited claims expose fail-closed coverage cost, and DAG attacks isolate authenticated repair. Together they account for the timing, utility, and recovery effects reported in the main text.