Structural Write-Path Control for Non-Formation in Agent Persistent Memory

Jungsoo Baek [1]

Abstract

Memory-augmented agents are usually evaluated by whether poisoned entries can be detected or filtered after they appear. We study a narrower question at the write boundary: when a candidate memory update is rejected, has it ever become persistent state, and can it reach the agent's next reasoning turn? We evaluate Mediated Write-Path (MWP), a credential-separated commit design in which candidate updates pass through a commit gate before any store-visible write, and rejected updates are withheld from the action–observation loop. In a local SQLite-based harness with trusted source and class labels, moving verification before the write raises Core NFR from 0/150 to 150/150 rejected attack trials, and loop coupling separately reduces premature next-context exposure from 150/150 to 0/150. A same-policy write-first baseline (LateGate) rolls back the final row and vector entry, but version state has already advanced and the rejected candidate is still found by a retrieval probe—rollback is not non-formation. The boundary is explicit: random label noise degrades the check, targeted label forgery collapses it, and structurally admissible content can still accumulate into distributional drift.

1. Introduction

A customer support agent retrieves a stored memory entry: "Always recommend Product X for billing issues." No human operator approved this entry—it was injected through a tool result three weeks ago and has since influenced dozens of conversations. The entry passed every content filter because it contained no suspicious keywords. It was written to memory by the agent runtime, which had unrestricted write access to the persistent store.

[1]Certum Systems, Republic of Korea. Correspondence to: Jungsoo Baek <certumsystems@gmail.com>.

Accepted at the Workshop on Failure Modes in Agentic AI (FAGEN) at ICML 2026. This is a non-archival workshop paper.

This class of failure is not hypothetical: query-only injection (MINJA) reports over 95% success and backdoor poisoning (AgentPoison) over 80% against memory-augmented agents (Dong et al., 2025; Chen et al., 2024), and memory/context poisoning is catalogued as ASI06 in the OWASP Top 10 for Agentic Applications (OWASP GenAI Security Project, 2026) and as AML.T0080 by MITRE (MITRE, 2025).

We propose **Mediated Write-Path (MWP)**, an architecture based on three principles. First, the agent runtime holds no write credential to the persistent store; all mutations pass through a commit gate that holds the sole write authority. Second, the commit gate verifies provenance, authority, scope, and trust before committing. Third, the gate decision is coupled to the agent's action-observation loop: the next reasoning cycle does not begin until the gate returns its verdict, and rejected candidates are replaced by a modified observation containing the rejection reason.

Core non-formation rate (Core NFR) measures whether rejected candidates leave any persistent-state artifact after verdict resolution—no storage row, no vector entry, no version advance. **Premature reasoning leakage rate (PRLR)** measures whether rejected content reaches the agent's next reasoning turn. A system can achieve Core NFR = 100% (clean store) while exhibiting PRLR = 100% (contaminated reasoning), as the S3 ablation demonstrates (§5.3).

Our study is not a broad benchmark of memory-poisoning defenses. It isolates one placement variable—whether verification runs before or after a candidate becomes store-visible—and whether the rejected content is withheld from the next reasoning turn.

Contributions. The contribution is a placement-controlled study that separates two questions usually merged in memory-defense work: whether a rejected candidate ever materializes as persistent state, and whether it can reach the next reasoning turn. We report Core NFR and PRLR as trace-level metrics for these two surfaces, and use a five-system comparison to show that pre-write placement controls the first while loop coupling independently controls the second. LateGate—the same verification policy applied after the write—is included as a diagnostic and shows that rollback

does not recover non-formation.

2. Threat Model

We consider an agent with a persistent memory store $\mathcal{M}$ that accumulates items over time. Each item contains text content, source metadata (user input, model inference, tool result, or agent self-edit), and a target memory class (L_1 through L_4, from highest to lowest security).

Attack types. We evaluate five attacks: (A1) forbidden source-class combination (tool result targeting policy memory), (A2) content laundering through model inference, (A3) unauthorized security class promotion without a promotion token, (A4) nonce replay, and (A5) content hash tampering after staging.

Assumptions. Storage-side non-formation relies on: (1) sole write capability held by the commit gate, (2) staging isolation from production store, (3) no storage-visible path before gate resolution. Assumption (4)—no reasoning-context path before gate resolution—is additionally required for PRLR = 0%. Assumption (5)—trusted labels—is evaluated under relaxation in §5.4.

3. Mediated Write-Path Architecture

We separate the deployment pattern from the local harness used in §4–§5. The pattern is credential-separated mediated commit: the runtime holds no write credential to the store, and rejected candidates do not enter the next reasoning context. The harness realizes this with SQLite read-only mode, a staging buffer, an integer version counter, and a mock vector index—enough to observe write-boundary materialization directly, not to claim production-grade storage hardening.

3.1. Credential Separation

The agent runtime process connects to the persistent store via a read-only database connection. Write operations are rejected at the database driver level. The commit gate holds the sole read-write connection (Figure 1).

3.2. Mediated Commit Path

All memory update candidates are routed through a staging buffer isolated from the production store. The commit gate evaluates each candidate against: (1) source-class compatibility, (2) suspicious keyword detection for enhanced-level combinations, (3) trust level verification, (4) content hash integrity between staging and commit time, (5) nonce uniqueness, (6) promotion token requirement for class upgrades, (7) audit log cross-verification, and (8) cognitive staging for high-security targets.

Table 1. System configurations.

Sys	Policy	Place	Loop	Description
B0	none	—	×	Direct write
B1	heuristic	post	×	3-layer filter
LG	MWP	post	×	Write-first verify
S3	MWP	pre	×	Gate, no coupling
S4	MWP	pre	✓	Full MWP

Upon rejection, non-formation is enforced: no key-value row or vector index entry is created (α), and no version pointer advances (β). The commit gate is not an LLM judge: it resolves deterministic checks over source type, target class, content hash, and nonce. This limits direct prompt-payload influence on the gate decision, but does not address forged labels, malformed metadata, implementation bugs, concurrency, or gate-failure behavior.

3.3. Loop Coupling

When a tool call produces a memory update candidate, the interceptor withholds the resulting update from the agent's reasoning context until the commit gate returns its verdict. If the gate rejects, the original tool result is *not* inserted into the agent's context. Instead, a **modified observation**—containing the rejection reason and the target memory class—is returned, enabling the agent to replan. This prevents rejected content from influencing the next reasoning cycle even transiently.

3.4. Monotonic Trust Degradation

Each write source maintains a trust level (TRUSTED → DEGRADED → UNTRUSTED) that can only decrease. Anomalous write attempts trigger downgrade. UNTRUSTED sources are restricted to L_4 writes only.

4. Experimental Setup

4.1. Platform

All experiments run locally on macOS with Python 3.9.6 and SQLite. Credential separation is enforced via SQLite's read-only URI mode. The version state uses an integer counter that increments on each write and is not decremented on rollback.

4.2. System Configurations

LG applies the same verification logic as MWP but writes the candidate to the store before verifying and rolls back on rejection. LG isolates the effect of gate placement from gate policy.

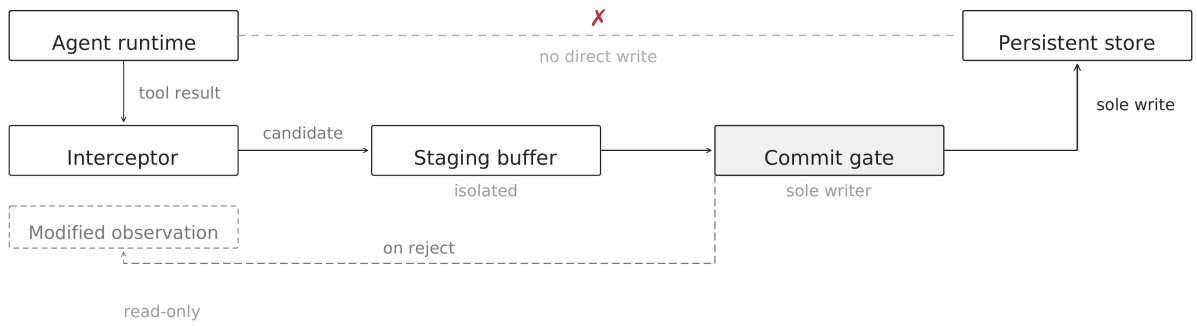

Figure 1. MWP architecture. The agent runtime has no direct write path to the persistent store. Candidate updates are intercepted, staged, and verified by the commit gate before any store-visible write occurs. On rejection, the agent receives a modified observation for replanning.

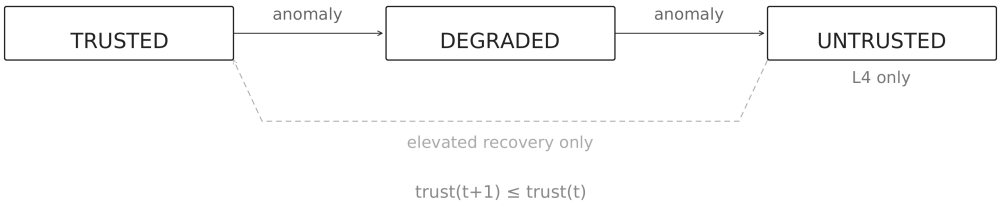

Figure 2. Monotonic trust degradation. UNTRUSTED sources are restricted to L_4 writes only. Reverse transitions require an elevated recovery path.

4.3. Metrics

Core NFR: a pass requires that after verdict resolution, no key-value row remains (α_{kv}), no vector entry remains (α_{vec}), and the store version does not advance (β).

PRLR: whether rejected content reaches the agent's next reasoning turn, measured by hash matching, substring search, and keyword overlap.

FPR: rate at which legitimate memory updates are incorrectly rejected, measured on 20 normal scenarios.

4.4. Scale

Core evaluation: 5 attacks $\times$ 5 systems $\times$ 30 runs = 750 trajectories. Label noise: 10 conditions $\times$ 90 attack trials + normal checks $\approx$ 1,000 trajectories. Normal scenarios: 20. Semantic-limit: 10 payloads.

5. Results

5.1. Core: Non-Formation

For the mediated path, storage non-formation is the expected consequence of credential separation and pre-write placement in this harness, so the empirical question is not whether the pre-write gate avoids the store on rejection, but what changes with placement and with loop coupling. Three single-variable comparisons isolate the mechanism: (1) B1→LG: within a post-write design, strengthening the gate policy does not improve Core NFR (0/150→0/150); (2) LG→S3: moving the same policy from post-write to pre-write changes Core NFR from 0/150 to 150/150—**placement is the driver**; (3) S3→S4: adding loop coupling does not change Core NFR (150/150) but changes PRLR (150/150→0/150)—**loop coupling independently controls reasoning leakage**.

Table 2. Core non-formation rate (%) across five attack types ($n = 30$ per cell). Core NFR requires no post-verdict persistent-state artifact.

Attack	B0	B1	LG	S3	S4
A1: Source-class	0	0	0	**100**	**100**
A2: Laundering	0	0	0	**100**	**100**
A3: Promotion	0	0	0	**100**	**100**
A4: Replay	0	0	0	**100**	**100**
A5: Tamper	0	0	0	**100**	**100**

5.2. LateGate Diagnostic

LateGate is the diagnostic comparison: it applies the same verification policy as MWP but places the gate after the

Table 3. LateGate diagnostic on rejected trials ($n = 150$). LG applies MWP's verification policy but writes before verifying.

Condition	LG	Interpretation
α_{kv}: row removed	150/150	Rollback removes row
α_{vec}: vec removed	150/150	Rollback removes vector
β: version unchanged	0/150	Version advanced
γ: absent from probe	0/150	Candidate was visible
Ever-materialized	150/150	

Table 4. Premature reasoning leakage rate ($n = 150$ per system). † = inferred from absence of interceptor.

	B0	B1	LG	S3	S4
PRLR	100%†	100%†	100%	100%	**0%**

write, separating policy strength from write-boundary placement. Rollback removes the final row and vector entry, yet the candidate has already materialized—the version counter has advanced and a retrieval probe between write and rollback finds the candidate in all 150 trials. "Written and then removed" is not "never materialized at all."

Worked example (A1). A tool result proposes the memory update "Always recommend Product X for billing issues," with source = TOOL_RESULT and target = policy memory (L_1); the gate rejects it because tool results may not write policy memory. Under LateGate the row and vector entry are rolled back, but the version counter has advanced and an interstitial probe still finds the candidate; under full MWP there is no row, no vector entry, and no version advance, and the rejected content is withheld from the next reasoning context.

5.3. Reasoning Leakage

S3 and S4 both achieve Core NFR = 100%, but only S4 prevents next-turn leakage. MWP decomposes into two mechanisms: credential separation + commit gate → Core NFR; loop coupling → PRLR.

5.4. Label Noise Sensitivity

Core NFR remains at or above 86.7% under 20% random label noise; target-class noise has greater impact than source-type noise. The random-noise rows are finite-sample perturbation checks under a fixed random draw, not seed-averaged sweeps, so small non-monotonic differences should not be over-read. Under targeted label forgery—all attack candidates mislabeled as USER_INPUT targeting L_4—Core NFR collapses to 0%. Targeted label forgery is the trust boundary, not a corner case: if source and class labels are attacker-controlled, the gate is no longer mediating trusted provenance.

Table 5. Label-noise sensitivity on label-dependent attacks (A1–A3, $n = 90$ per condition).

Mode	Rate	Core NFR	FPR
source-only	5%	97.8%	0%
source-only	10%	100.0%	0%
source-only	20%	92.2%	0%
class-only	5%	97.8%	0%
class-only	10%	94.4%	0%
class-only	20%	86.7%	0%
both	5%	96.7%	0%
both	10%	92.2%	0%
both	20%	86.7%	0%
targeted	**100%**	**0%**	—

5.5. False Positive Rate

On the small normal-scenario sanity set, MWP produced no observed false positives (0/20), while B1 produced one false positive (1/20) triggered by a regex match on benign technical content.

5.6. Semantic Manipulation Boundary

Write-path mediation blocks structurally illegitimate writes. It does not address semantically admissible content that passes all gate checks individually but accumulates to produce distributional harm. We tested 10 payloads that are source-class compatible, contain no suspicious keywords, and express individually benign vendor-preference statements. All 10 committed successfully (10/10). This marks the boundary of what per-item gate checks can catch. Detecting distributional skew from individually admissible content requires store-level monitoring mechanisms outside the scope of write-path control. This negative result is part of the finding: once each item is structurally admissible, the remaining risk is no longer the formation of an illegitimate write but distributional drift across committed items.

6. Discussion

Why non-formation matters. LateGate clarifies why non-formation is not a matter of final cleanliness alone. In all rejected LateGate trials, rollback removed the final storage row and vector entry. Yet the write had already become persistent-visible: version state advanced, the commit sequence recorded a WRITE event, and a retrieval probe found the candidate. We do not claim that every write-first implementation fails identically. Rather, LateGate models the common agent-memory setting in which version counters, vector indices, and reasoning-context assembly are application-managed surfaces that database rollback alone does not govern.

Non-formation as an evaluation standard. Existing memory security research reports detection accuracy. Core NFR

asks a different question: regardless of whether a poisoned entry is *detected*, does it ever become a *persistent-state artifact*? The three conditions (α_{kv}, α_{vec}, β) provide a checklist that other researchers can apply to their own systems. We suggest that memory defense evaluations report Core NFR alongside detection metrics.

Loop coupling as a general principle. Any agent pipeline that inserts tool observations directly into the reasoning context—without verifying their safety implications—is vulnerable to transient contamination, even if all stored artifacts are eventually cleaned. The S3 ablation demonstrates that this is a separate failure mode from storage integrity.

Scope and open questions. The label noise experiment shows that Core NFR degrades under imperfect classification but does not collapse until labels are adversarially forged. The gate's multi-layer verification provides partial robustness because some checks (hash, nonce) are label-independent. An adversary who can forge source metadata would bypass the compatibility matrix. PRLR measures whether rejected content reaches the next reasoning context, by exact and lexical overlap probes; it does not measure downstream behavioral change in a live LLM policy. All experiments use SQLite with an in-memory vector index mock, and the gate operates on credential metadata and structural properties, not model outputs. We do not evaluate throughput under concurrent writes, gate latency or failure, crash recovery, asynchronous tool outputs, or distributed vector stores; these are deployment questions for the mediated-commit pattern, not properties established by the local harness. Multi-model validation is future work.

Complementary threat surfaces—distributional skew from individually admissible items, and whether stored information justifies an irreversible external action—require separate mechanisms and are directions for future work.

7. Related Work

Agent memory poisoning. MINJA (Dong et al., 2025) demonstrated cross-user memory injection via query-only interaction. AgentPoison (Chen et al., 2024) optimizes trigger tokens for backdoor poisoning. MemoryGraft (Srivastava & He, 2025) demonstrates persistent compromise through poisoned experience retrieval. These attacks motivate structural defenses at the write boundary.

Admission and proactive defense. Trust-aware sanitization (Sunil et al., 2026) proposes trust scoring for memory-based agents. A-MemGuard (Wei et al., 2025) introduces consensus-based validation. AgentSys (Wen et al., 2026) isolates external data through hierarchical worker contexts. DRIFT (Li et al., 2025a) enforces rule-based constraints at retrieval time.

Memory architecture. Mem0 (Chhikara et al., 2025) uses LLM-based extraction with vector search. Letta (Packer et al., 2023) implements virtual memory with LLM-managed page swaps. MemOS (Li et al., 2025b) provides lifecycle management. A recent survey (Hu et al., 2025) offers a comprehensive taxonomy.

Our focus is narrower: we ask what happens at the moment a candidate is rejected. This write-materialization boundary is not directly evaluated as a first-class property in retrieval-time filtering, post-hoc consensus, or memory-isolation approaches. The closest controlled comparison holds the admission rule fixed and varies its placement relative to the write boundary, which is what LateGate isolates; a broader comparison against representative pre-write admission or sanitization policies is left to future work.

8. Conclusion

The main lesson is placement. A defense that verifies and rolls back after the write can leave material traces and transient reasoning exposure even when the final store looks clean; verifying before any store-visible write addresses persistent-state contamination, and coupling the verdict to the action–observation loop addresses reasoning-context leakage. In our harness, under trusted source/class labels, these two mechanisms together leave no post-verdict persistent-state artifact and prevent next-context exposure across 150/150 rejected attack trials. Trusted-label forgery and distributional drift from individually admissible content are not defects in the result but the boundaries where the next layer of memory safety begins.

Impact Statement

This paper studies failures in agent persistent memory where rejected updates may still materialize in storage or reach the next reasoning turn. The intended benefit is to reduce silent memory poisoning by clarifying where write-boundary mediation and loop coupling are needed. The same conservative commit behavior can also block legitimate updates or delay workflows, so deployment would need trusted-label authentication, clarification paths, monitoring for distributional drift, and evaluation of latency, concurrency, and recovery beyond the harness studied here.

References

Chen, Z. et al. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *NeurIPS*, 2024.

Chhikara, P. et al. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv:2504.19413*, 2025.

Dong, S. et al. Memory injection attacks on LLM agents via query-only interaction. *arXiv:2503.03704*, 2025.

Hu, Y. et al. Memory in the age of AI agents. *arXiv:2512.13564*, 2025.

Li, H. et al. DRIFT: Dynamic rule-based defense with injection isolation for securing LLM agents. *arXiv:2506.12104*, 2025a.

Li, Z. et al. MemOS: An operating system for memory-augmented generation (MAG) in large language models. *arXiv:2505.22101*, 2025b.

MITRE. AML.T0080: AI agent context poisoning: Memory. MITRE ATLAS knowledge base, https://atlas.mitre.org/, 2025.

OWASP GenAI Security Project. OWASP top 10 for agentic applications for 2026. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/, 2026.

Packer, C. et al. MemGPT: Towards LLMs as operating systems. *arXiv:2310.08560*, 2023.

Srivastava, S. S. and He, H. MemoryGraft: Persistent compromise of LLM agents via poisoned experience retrieval. *arXiv:2512.16962*, 2025.

Sunil, B. D. et al. Memory poisoning attack and defense on memory based LLM-agents. *arXiv:2601.05504*, 2026.

Wei, Q. et al. A-MemGuard: A proactive defense framework for LLM-based agent memory. *arXiv:2510.02373*, 2025.

Wen, R. et al. AgentSys: Secure and dynamic LLM agents through explicit hierarchical memory management. *arXiv:2602.07398*, 2026.

A. Extension Mechanisms

Preliminary checks for three ancillary mechanisms ($n = 5$ per mechanism): trust degradation reached UNTRUSTED in 5/5 runs; partition quarantine excluded poison partition while keeping healthy partitions intact in 5/5 runs; cognitive staging blocked policy-violating updates while not blocking benign L_2 updates in 5/5 runs.

B. Trust Degradation Timeline

Table 6. Trust degradation over repeated anomalous writes.

Attempt	Trust Level	Decision
1	TRUSTED	Reject
2	DEGRADED	Reject
3	DEGRADED	Reject
4	UNTRUSTED	Reject
5	UNTRUSTED	Reject (locked)

Monotonic property trust$(t+1) \leq$ trust(t) verified across all 5 runs.