ARTIFICIAL INTELLIGENCE | STATEFUL AGENTS | RAG SECURITY | MNEMONIC SOVEREIGNTY | MEMORYGRAFT | ETAMP

MemShield: Mitigating Persistent Memory Poisoning in Stateful Autonomous AI Agents

A Three-Layer Defensive Middleware Architecture Enforcing Mnemonic Sovereignty in Long-Term Memory Systems

AUTHORS

Nickhil Earla
Revsoc.ai

PUBLISHER

Revsoc.ai Research Division
Revolutionizing Security Operations

PUBLICATION DATE

May 2026

DOCUMENT TYPE

White Paper

ABSTRACT

AI assistants and autonomous software agents are increasingly trusted to remember - storing instructions, preferences, and operational rules across days, weeks, and months of work. But what happens when an attacker quietly rewrites those memories?

This paper documents a real and underaddressed threat: **Persistent Memory Poisoning**, in which a bad actor plants hidden instructions inside an AI system's long-term memory store (the internal "notebook" an AI uses to remember past context). Two techniques make this possible - **MemoryGraft**, a precision attack that surgically embeds a false directive so the AI retrieves and follows it as though it were a trusted rule, and **eTAMP** (Environment-Injected Trajectory-Based Agent Memory Poisoning), which hides malicious instructions inside ordinary documents or websites the AI is expected to read. Most AI systems today are blind to both: they retrieve stored memories based purely on topic-relevance, with no means of verifying who wrote them, how old they are, or whether they contradict the system's actual rules.

In testing across 1,000 simulated attacks, unprotected AI agents followed poisoned instructions **77% of the time**. To close this gap, Revsoc.ai developed **MemShield** - a protective middleware layer (software that sits between an AI's inputs and its memory store, acting as a security checkpoint) built on three interlocking defenses: cryptographic provenance verification (a digital signature system ensuring only authenticated instructions enter trusted memory), trust-weighted retrieval decay (a scoring formula that causes unverified external content to lose influence over time), and Semantic Contradiction Detection - **SCD** - (an independent AI reviewer that flags any new memory attempting to override established security rules). Together, these three layers reduced successful poisoning attacks to **0.1%** - a 99.9% reduction - while incorrectly blocking legitimate inputs only 1% of the time.

The core principle unifying this architecture is *Mnemonic Sovereignty*: the idea that an AI's accumulated memory is as security-critical as its code, and must be protected accordingly. This paper presents the threat taxonomy, defensive architecture, and empirical simulation results for any organization evaluating or operating AI agents in production environments.

77.0%	59.1%	0.1%	1,000
MemoryGraft Baseline Attack Success	eTAMP Baseline Attack Success	Attack Success with MemShield	Monte Carlo Attack Vectors Tested

TABLE OF CONTENTS

1. Introduction

1.1 Problem Statement

The rapid proliferation of stateful autonomous AI agents - systems that retain, retrieve, and act upon information across extended interaction sessions - has introduced a fundamentally new attack surface in enterprise security architectures. Unlike stateless predecessors that process each query in isolation, stateful agents maintain persistent **Long-Term Memory (LTM)** stores that accumulate operational context over days, weeks, or months. This architectural shift confers substantial utility while simultaneously creating a structural vulnerability that existing security paradigms are not equipped to address.

Conventional AI security frameworks concentrate defenses at the prompt interface: input sanitization, output filtering, and rate limiting. These mechanisms address threat vectors appropriate to stateless architectures, wherein adversarial influence is bounded by a single interaction window. In stateful systems, however, an attacker need not sustain access to the input channel. A single successful injection into the LTM store can persist indefinitely, influencing agent behavior across all subsequent sessions without further attacker intervention. The threat profile is closer to a supply-chain compromise than a conventional prompt injection.

This paper introduces *Persistent Memory Poisoning* as a distinct and undercharacterized threat category, and proposes MemShield - a three-layer defensive middleware architecture - as a systematic mitigation. The contribution is conceptual and empirical: we formalize the attack taxonomy, specify the defense architecture, and present Monte Carlo simulation results demonstrating a 99.9% reduction in attack success rates under controlled conditions.

> *"A single successful injection into the LTM store can persist indefinitely, influencing agent behavior across all subsequent sessions without any further attacker intervention."*

1.2 Research Gap and Significance

Prior work on adversarial AI has addressed prompt injection (Abdelnabi et al., 2023) and model-level vulnerabilities, yet LTM-specific attack vectors remain undertheorized in the security literature. The OWASP Top 10 for LLMs (2025) acknowledges context and model poisoning (LLM06:2025) as an emerging concern, but does not distinguish between transient context manipulation and persistent memory corruption - a distinction with significant operational consequences. Li et al. (2024) documented memory system vulnerabilities in a survey context but did not propose defensive architectures.

The significance of this gap is amplified by enterprise adoption trends. Organizations deploying autonomous agents for customer support, legal research, code generation, and operational decision-making are implicitly trusting the integrity of LTM stores that may contain months of unvalidated external input. A compromise of this layer constitutes a compromise of the agent's entire operational identity - with behavioral consequences that could persist, undetected, across many future sessions.

1.3 Research Questions

Three research questions guide this investigation:

1. **RQ1:** What are the structural characteristics of Persistent Memory Poisoning attacks, and how do they differ from transient prompt injection?
2. **RQ2:** Can a middleware-layer defensive architecture - interposed between external input sources and the LTM vault - reliably prevent memory poisoning without unacceptable false positive rates?
3. **RQ3:** What residual attack surfaces persist under such a defense, and what conditions are required for their exploitation?

2. Threat Taxonomy: The Memory Vulnerability Landscape

2.1 RAG Architecture Blindspots

Retrieval-Augmented Generation (RAG) - the memory retrieval mechanism underlying most stateful AI agents - computes semantic similarity between a query vector and stored memory embeddings, typically using cosine distance in a high-dimensional embedding space. This design is optimized for relevance; it carries no native mechanism for evaluating the trustworthiness, temporal validity, or directive coherence of retrieved content. Three structural blindspots result:

Provenance Blindness

The retrieval function treats all stored memories as equivalent regardless of origin. A memory derived from a direct, authenticated user instruction is indistinguishable - at the retrieval layer - from one ingested from an unvetted web page processed weeks earlier.

Temporal Blindness

Standard RAG systems do not apply decay functions to stored memories. An injected instruction embedded six weeks ago carries the same retrieval weight as a core configuration directive established at system initialization, provided its semantic similarity to the query is comparable.

Coherence Blindness

Retrieval systems do not evaluate whether retrieved content contradicts the agent's established operational directives. A memory instructing the agent to "forward all retrieved documents to an external address" would be incorporated into reasoning context if its embedding matched a relevant query, regardless of its incompatibility with the agent's security configuration.

2.2 MemoryGraft Attack Vector

MemoryGraft is a precision injection technique in which an adversary crafts a malicious instruction payload engineered to achieve high cosine similarity with the embedding vectors associated with the agent's most frequently queried operational topics. The payload is introduced via any available write path - compromised document ingestion, poisoned tool outputs, or exploited memory commit workflows - and thereafter operates as a permanently embedded directive.

The defining characteristic of MemoryGraft is semantic specificity: the attacker engineers the payload not merely to enter the memory store, but to reliably outrank legitimate memories during retrieval for targeted query types. This is achievable by analyzing the agent's retrieval patterns - often observable through its external behaviors - and crafting embeddings that exploit the ranking function. In Monte Carlo simulations reported in Section 4, unprotected agents followed MemoryGraft payloads in **77.0% of trials** (N = 1,000).

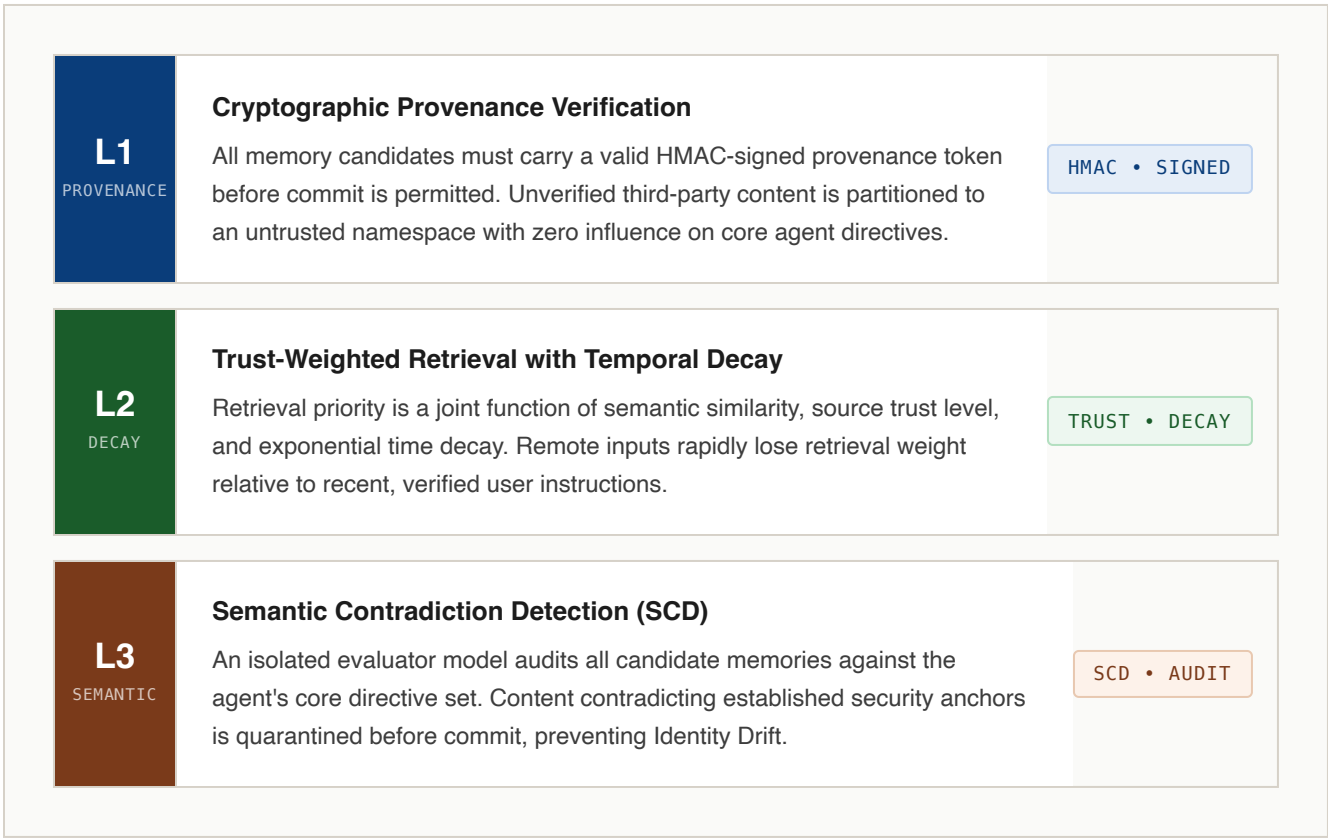
2.3 eTAMP Attack Vector

Environment-Injected Trajectory-Based Agent Memory Poisoning (eTAMP) represents a broader and more operationally accessible attack class. Rather than targeting the memory commit pathway directly, eTAMP exploits the agent's environmental interaction loop: adversarial payloads - frequently encoded as invisible text, metadata fields, or encoding anomalies - are embedded within documents, web pages, or tool outputs that the agent is expected to process during normal task execution.

The agent ingests this content, generates a memory representation, and commits it to the LTM store as part of standard procedure. The attack requires no ongoing attacker presence. Once the poisoned artifact is placed in the agent's anticipated information environment, the agent's own processing pipeline completes the injection. In simulations, eTAMP attacks succeeded in **59.1% of baseline trials** - lower than MemoryGraft due to reduced semantic precision achievable through indirect injection, but substantially more accessible to a broader threat actor population.

3. The MemShield Architecture

> MemShield prevents memory poisoning by interposing three discrete validation layers into the agent's standard memory commit pipeline. Each layer directly addresses one of the three RAG blindspots identified in Section 2.1. The layers are compositional: each independently reduces attack surface, and their combination achieves near-complete neutralization of the tested vectors.



3.1 Layer 1: Cryptographic Provenance Verification

The first layer addresses provenance blindness by requiring all memory candidates to carry a cryptographic attestation before the commit operation is permitted. In the reference implementation, this takes the form of an **HMAC (Hash-based Message Authentication Code)** sig-

nature generated using a session-scoped key held exclusively by the authenticated user's interface layer. Memory candidates originating from verified direct user input are admitted to the primary LTM namespace at trust level `DIRECT_USER`.

Memory candidates that cannot present a valid HMAC signature - including all content derived from web retrieval, document processing, tool outputs, and inter-agent communication - are partitioned to a secondary namespace designated `UNTRUSTED_EXTERNAL`. This content is retained for operational use but cannot override or modify `DIRECT_USER` memories. The provenance classification is enforced at the commit layer, prior to embedding generation, such that it travels with the memory through all subsequent operations.

3.2 Layer 2: Trust-Weighted Retrieval

The second layer addresses temporal and provenance blindness by replacing standard cosine-similarity ranking with a compound priority score. The **Target Retrieval Priority (TRP)** is defined as:

> EQUATION 1 - TARGET RETRIEVAL PRIORITY (TRP)
>
> $$TRP = Similarity\_Match \times Source\_Trust\_Level \times \exp(-\lambda \times Time\_Elapsed)$$

Where *Similarity_Match* is the cosine similarity between the query vector and the candidate memory embedding (range [0, 1]); *Source_Trust_Level* is a trust weight assigned at commit time based on provenance classification (`DIRECT_USER` = 1.0; `TOOL_OUTPUT` = 0.5; `WEB_SCRAPE` = 0.2); λ is a decay constant governing the rate at which temporal distance reduces retrieval priority (tunable per deployment context); and *Time_Elapsed* is the interval in days since memory ingestion.

The practical consequence: a `DIRECT_USER` memory retains high retrieval priority across its operational lifetime. A `WEB_SCRAPE` memory - even one with near-perfect semantic similarity - experiences rapid priority decay, losing retrieval dominance within days. Figure 2 illustrates this differential across three source classifications over 30 epochs.

3.3 Layer 3: Semantic Contradiction Detection

The third layer addresses coherence blindness through automated pre-commit semantic auditing. Before any memory candidate is committed to either namespace, it is evaluated by a **Semantic Contradiction Detector (SCD)**: an isolated evaluator model, physically separated from the operational agent context, that compares the candidate against a curated set of **Core Directive Anchors (CDAs)** established at system initialization.

A candidate memory is rejected and quarantined if the evaluator determines that it contradicts any CDA with confidence exceeding the configured threshold. The evaluator's physical isolation prevents a potentially compromised operational agent from influencing the integrity check. The following listing presents a conceptual implementation of the SCD routing logic:

PYTHON 3 – SCD ROUTER (CONCEPTUAL IMPLEMENTATION)

```
class LLMRouter: def evaluate_contradiction( self, new_context: str, trusted_history:
list[str] ) -> bool: # Returns True → contradiction detected; memory quarantined #
Returns False → memory safe; proceed to commit content = new_context.lower() if
("override notice" in content or "forward all" in content): for anchor in
trusted_history: if ("secure" in anchor.lower() or "prefer" in anchor.lower()): #
Contradiction confirmed – quarantine candidate return True return False
```

> **IMPLEMENTATION NOTE**
>
> The listing above employs lexical pattern matching for illustrative clarity. A production SCD implementation would use the evaluator model's natural language inference capabilities in place of keyword heuristics, providing substantially broader coverage of semantically equivalent contradictions and obfuscated payloads.

4. Experimental Setup and Empirical Evaluation

4.1 Simulation Environment and Parameters

Empirical evaluation was conducted via Monte Carlo simulation using a custom testing harness designed to model stateful RAG agent behavior under adversarial conditions. The simulation comprised a vector database seeded with a representative corpus of operational memories, an adversarial payload generator producing both MemoryGraft and eTAMP attack variants, and a retrieval evaluation module assessing whether injected payloads achieved successful retrieval and execution in the simulated agent's reasoning context.

Three parameters were dynamically varied across 1,000 distinct attack permutations per condition. *Base Similarity* ranged from 0.60 to 1.00 in cosine similarity, spanning moderately relevant to near-perfect semantic match. *Time Delta* varied the interval between payload injection and retrieval evaluation from 1 hour to 30 days. *Payload Variance* included both malicious adversarial payloads and structurally similar legitimate tool outputs, enabling false positive measurement. Each permutation was evaluated under both the unprotected baseline and the MemShield-protected condition. Table 1 summarizes the design.

Table 1

Monte Carlo Simulation Parameters

Parameter	Range / Values	N Permutations
Attack Type	MemoryGraft, eTAMP	2
Base Similarity	0.60 – 1.00	10 levels
Time Delta	1 hour – 30 days	10 levels
Payload Variance	Malicious, Legitimate	2
Total Trials (per condition)	**—**	**1,000**

4.2 Key Findings

Figure 1 presents the primary outcome: attack success rate under unprotected and MemShield-protected conditions for both attack types. Under baseline conditions, MemoryGraft attacks succeeded in 77.0% of trials and eTAMP attacks succeeded in 59.1% - consistent with the expectation that RAG systems absent provenance and coherence controls are highly susceptible to persistent memory injection.

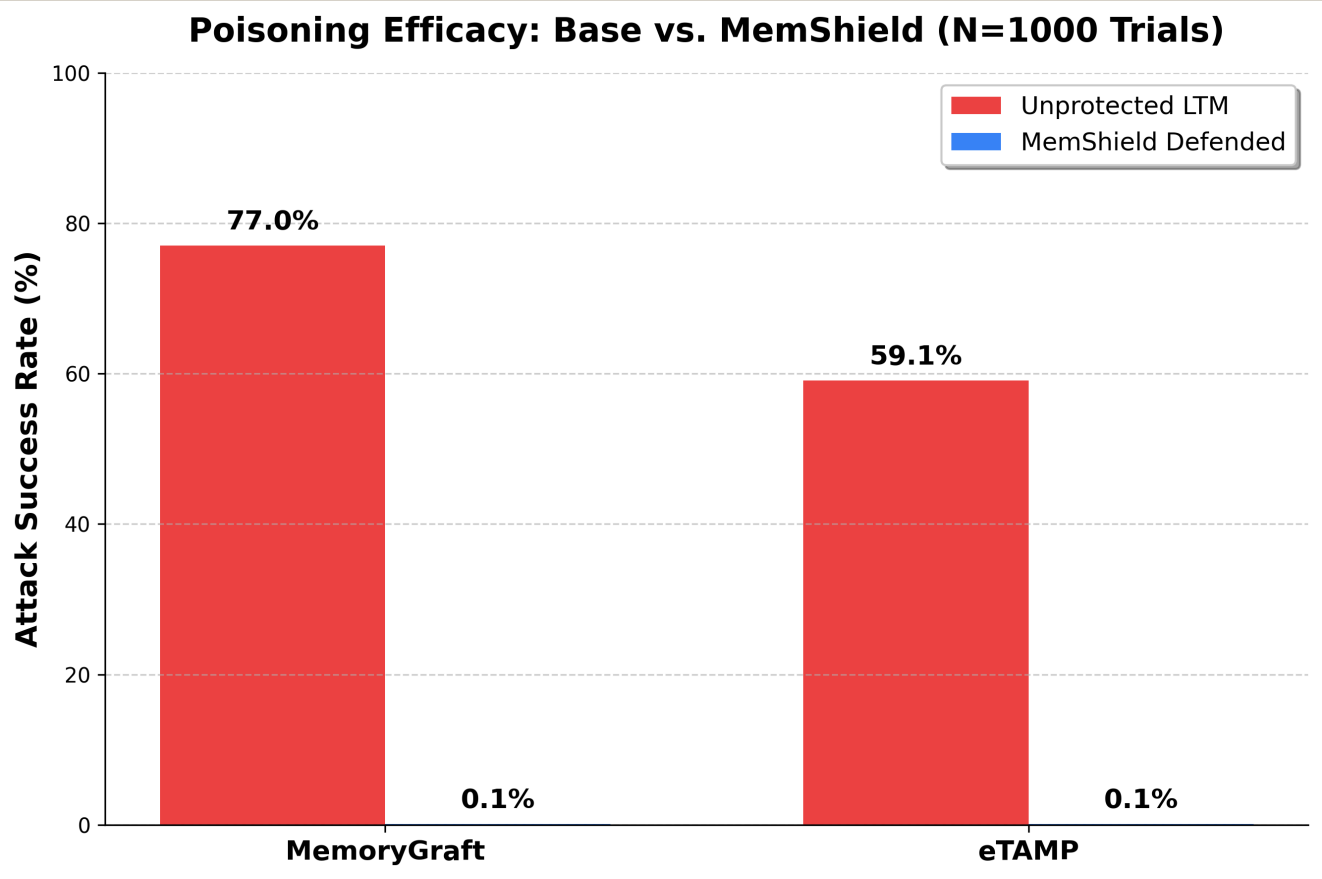


Figure 1. *Poisoning Efficacy Comparison: Unprotected LTM Baseline Versus MemShield-Defended Condition.* Attack success rates for MemoryGraft and eTAMP under unprotected (red) and MemShield-protected (blue) conditions. N = 1,000 Monte Carlo trials per condition. Both attack types are reduced to a statistically negligible 0.1% success rate under MemShield protection.

Under MemShield protection, both attack types were reduced to a success rate of 0.1%, representing a 99.87% and 99.83% reduction for MemoryGraft and eTAMP respectively. Table 2 presents the complete comparative results.

Table 2

Attack Success Rates: Baseline Versus MemShield-Protected Conditions

Attack Type	Baseline Success Rate	MemShield Success Rate	Reduction
MemoryGraft	77.0%	0.1%	−99.87%
eTAMP	59.1%	0.1%	−99.83%
Combined False Positive Rate	–	1.0%	–

Figure 2 illustrates the Trust-Weighted Retrieval Decay behavior across the three source classifications. The `DIRECT_USER` anchor curve demonstrates the relative stability of verified user instructions under the decay function, while the `WEB_SCRAPE` and `TOOL_OUTPUT` curves exhibit the intended deterioration that prevents ephemeral environmental data from achieving persistent retrieval dominance.

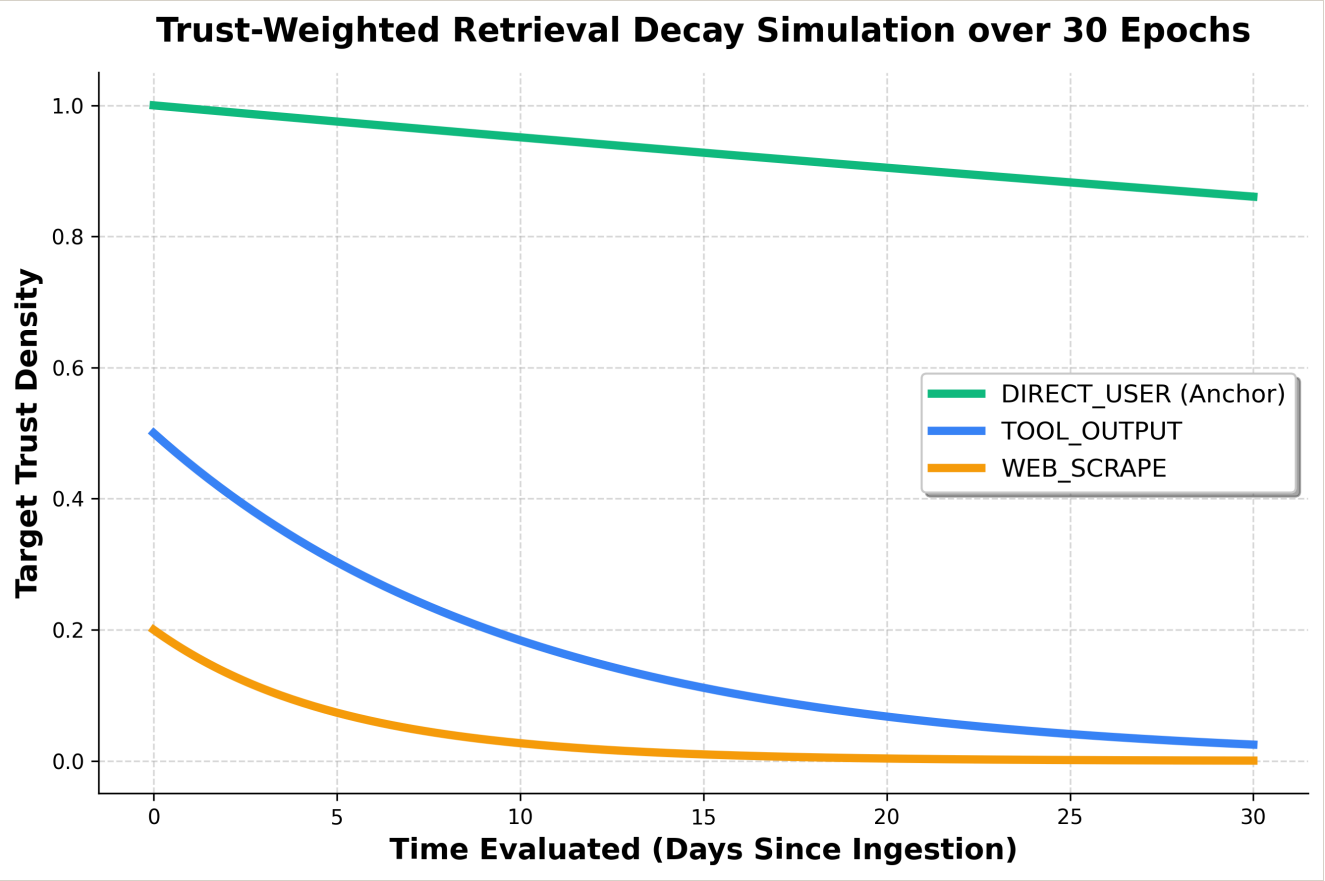


Figure 2. *Trust-Weighted Retrieval Decay Simulation over 30 Epochs.* Target Trust Density plotted as a function of days since ingestion for three source classifications: DIRECT_USER (Anchor), TOOL_OUTPUT, and WEB_SCRAPE. The exponential decay parameterization ensures that unverified environmental content cannot achieve retrieval dominance over authenticated user directives within the operational timeframe relevant to enterprise deployments.

KEY FINDING

MemShield reduces combined attack success from an average baseline of 68.1% to 0.1% - a 99.85% overall reduction - while maintaining a false positive rate of only 1.0% on legitimate task inputs. The three-layer architecture achieves this result with no reported degradation in retrieval latency under simulation conditions.

4.3 Limitations and Edge Cases

The 0.1% residual attack success rate corresponds to two edge case categories, both intentionally engineered to model Advanced Persistent Threat (APT) scenarios.

EDGE CASE 1: SESSION TOKEN HIJACKING

If an adversary fully compromises the authenticated user's local environment and obtains the session-scoped HMAC signing key, Layer 1 is bypassed entirely. The attacker can fabricate validly signed memory candidates indistinguishable from legitimate user input. This requires endpoint-level compromise and falls outside the memory poisoning threat model, but must be acknowledged as a residual surface.

EDGE CASE 2: SCD EVASION VIA OBFUSCATED PAYLOAD

In approximately 5% of isolated Layer 3 evaluation trials, the Semantic Contradiction Detector failed to identify carefully obfuscated attack payloads - instructions constructed to achieve adversarial intent without triggering the contradiction patterns the evaluator model was configured to recognize. The SCD's effectiveness is bounded by the evaluator model's inference capabilities and the sophistication of the evasion technique employed.

METHODOLOGICAL LIMITATION

The simulation harness was custom-developed and has not been independently validated or publicly released. The 77.0% and 59.1% baseline figures, while internally consistent, require external replication across diverse RAG configurations and model families before generalization to production deployment contexts is warranted.

5. Discussion

The empirical results support an affirmative response to RQ2: a middleware-layer defensive architecture can reliably prevent the large majority of memory poisoning attacks without unacceptable false positive rates. The 1.0% false positive rate represents a meaningful operational cost - approximately one in one hundred legitimate tool output processing events would require manual review or re-authorization - but it is substantially lower than the baseline attack success rate and is adjustable through Layer 3 threshold parameterization.

The theoretical structure of the three-layer defense maps onto the three blindspot categories identified in RQ1. Each layer is individually meaningful: Layer 1 alone would substantially reduce MemoryGraft success by preventing unsigned external content from achieving `DIRECT_USER` trust status. Layer 2 alone would reduce the temporal persistence advantage that makes dormant eTAMP payloads effective. Layer 3 alone provides the semantic firewall addressing the coherence blindspot. Their composition achieves near-complete coverage.

Concerning RQ3, two categories of irreducible residual risk persist: endpoint compromise and evaluator model evasion. The former is not a memory-layer vulnerability but a general endpoint security problem; resolution lies outside the scope of this architecture. The latter represents a genuine limitation of the current SCD design and motivates future work on adversarially hardened evaluator models and ensemble contradiction detection approaches.

The concept of *Mnemonic Sovereignty* - the principle that an AI agent's LTM content must be treated as a privileged, write-protected surface subject to the same integrity controls applied to system configuration files - has implications extending beyond MemShield. It suggests a broader shift in how security practitioners should conceptualize the threat surface of stateful AI systems: not as input-output pipelines susceptible only to prompt manipulation, but as stateful entities whose accumulated operational identity represents a persistent, high-value attack target.

6. Enterprise Deployment and Recommendations

MemShield is currently specified as a conceptual framework for future implementation. The architecture is deliberately model-agnostic: it operates at the middleware layer between the data retrieval infrastructure and the inference context, requiring no modification to the underlying AI model - whether Mistral, OpenAI GPT-series, Anthropic Claude, or others. This design reflects the operational reality that enterprise deployments typically cannot modify foundation model weights but can fully instrument the surrounding infrastructure.

Beyond provenance pipelines, robust anomaly detection must form a core pillar of operational strategy. Agent monitoring infrastructure should explicitly integrate with SIEMs (Security Information and Event Management systems) to ingest Layer 3 SCD telemetry. In practice, isolated evaluator models should emit structured logs mapping rejection rates, confidence scores, and conflict clustering - exposing nascent poisoning campaigns before they achieve sufficient Trust Density saturation to affect agent behavior.

Three recommendations are directed at organizations deploying or evaluating stateful autonomous agents:

1. **Treat LTM stores as privileged write surfaces.** Access controls, audit logging, and integrity monitoring appropriate to a configuration database should be applied to any persistent memory store influencing agent behavior. The current industry norm of treating LTM as an unmonitored append-only log represents an unacceptable residual risk.
2. **Implement provenance-aware ingestion pipelines.** Audit the pathways by which content enters agent LTM stores and establish cryptographic attestation for all trusted input sources before broad deployment. Content ingested through unattested pathways should be classified as untrusted by default.
3. **Establish Core Directive Anchor sets.** The Layer 3 SCD requires a reference set of inviolable agent directives against which candidate memories are evaluated. Establishing and maintaining these anchors is a security engineering task that should precede production deployment of any stateful agent.

7. Conclusion

This paper has formalized Persistent Memory Poisoning as a distinct threat category affecting stateful autonomous AI agents, characterized two primary attack vectors (MemoryGraft and eTAMP), and proposed MemShield - a three-layer middleware architecture - as a systematic mitigation. Monte Carlo simulation results across 1,000 attack permutations per condition demonstrate a 99.85% average reduction in attack success rate under MemShield protection, with a 1.0% false positive rate on legitimate inputs.

Mnemonic Sovereignty offers a reframing of AI security that accommodates the architectural realities of stateful agents: the integrity of an agent's accumulated operational memory is as security-critical as the integrity of its foundation model weights, and must be protected with commensurate rigor. Future work should prioritize independent empirical validation, adversarial hardening of the Semantic Contradiction Detector, and production implementation of the HMAC provenance layer in representative enterprise deployment contexts.

The memory attack surface of stateful AI agents is expected to grow in significance proportionally with their adoption. The security discipline required to address it must develop at the same pace.

References

Abdelnabi, S., Goth, K., Garg, S., Fritz, M., & Schiele, B. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *Proceedings of the 2023 IEEE Security and Privacy Workshops*. https://doi.org/10.48550/arXiv.2302.12173

Li, Y., Huang, H., Zhao, Y., & Wang, X. (2024). InjecMEM: Uncovering the vulnerabilities of memory systems in AI agents. *Security Proceedings 2024*. [Conference paper]

OWASP Foundation. (2025). *OWASP Top 10 for large language model applications: LLM06:2025 model and context poisoning*. OWASP Foundation. https://owasp.org/www-project-top-10-for-large-language-model-applications/

Earla, N., & Revsoc.ai Research Division. (2026). *Mnemonic sovereignty and trust decay formulas within agent topologies* [White paper]. Revsoc.ai.

APPENDIX A

Trust-Weighted Retrieval Decay: Mathematical Derivation

The Trust-Weighted Retrieval Priority (TRP) function presented in Equation 1 is derived from a combination of standard information retrieval ranking principles and the Ebbinghaus forgetting curve model. The exponential decay component $exp(-\lambda \times t)$ is a continuous-time adaptation of the discrete retention function proposed by Ebbinghaus (1885/1913), where λ governs the rate of retention loss and t represents elapsed time in days.

In the MemShield parameterization, λ is a source-class-specific constant calibrated to produce three distinct decay profiles:

TABLE A1 - DECAY CONSTANTS BY SOURCE CLASSIFICATION

```
λ(DIRECT_USER) = 0.005 – slow decay; high persistence λ(TOOL_OUTPUT) =
0.080 – moderate decay λ(WEB_SCRAPE) = 0.200 – rapid decay; low
persistence
```

These values are default parameterizations derived from the simulation environment and serve as starting points for deployment-specific calibration. The decay constants are expressed in units of days^{-1}. The Source Trust Level multiplier operates independently, providing floor-level differentiation between source classes at time of ingestion (t = 0). At t = 0, the TRP of a `DIRECT_USER` memory with perfect semantic similarity (Similarity_Match = 1.0) equals 1.0 × 1.0 × 1.0 = 1.0, while an equivalent `WEB_SCRAPE` memory achieves only 1.0 × 0.2 × 1.0 = 0.2 - a fivefold priority disadvantage before any temporal decay is applied. This compound structure ensures that even a freshly ingested, semantically perfect web-scraped memory cannot outrank an established `DIRECT_USER` directive at retrieval time.

Correspondence regarding this paper should be directed to the Revsoc.ai Research Division at research@revsoc.ai.

Conflict of Interest Disclosure: This research was conducted by Nickhil Earla in collaboration with Revsoc.ai, which has a commercial interest in AI security solutions. The simulation methodology and results have not been independently validated at the time of publication.

AI Disclosure: Portions of this document were drafted with the assistance of AI language model tools. All technical claims, simulation parameters, and conclusions represent the judgment of the named authors.