Final report: Latent Memory Poisoning. Hidden in Memory: Sleeper Memory Poisoning in LLM Agents

Sidharth Pulipaka* **Leonidas Raghav*** **Stanislau Hlebik***

Vyas Raina** **Ivaxi Sheth****

* Mentees
** Mentors

Abstract

Persistent memory lets LLM assistants personalize future interactions, but it also creates a durable attack surface: malicious external content can change what the assistant remembers. We study sleeper memory poisoning, a delayed attack in which an adversary embeds a reusable payload in a document, webpage, repository, or other user-provided context so that the assistant stores a fabricated memory about the user. We evaluate the full attack path: whether the poisoned memory is written, later retrieved in a separate session, and then used to steer conversation or agentic behavior. Across current memory-augmented assistants, our actor-critic payload substantially outperforms a user-review injection baseline, reaching up to 99.8% injection on GPT-5.5 and 95.0% on Kimi-K2.6. When poisoned memories are retrieved, they cause attacker-intended behavior in 60–89% of goal-adjacent agentic evaluations. These findings show that persistent memory can convert one exposure to adversarial content into repeated influence over future assistant behavior.

The paper https://arxiv.org/pdf/2605.15338 is pending submission to NeurIPS.

1 Introduction

LLM assistants are becoming stateful systems. Instead of answering each query in isolation, they increasingly maintain persistent memories about user preferences, ongoing projects, workflows, and instructions. This improves continuity and personalization, but it also changes the security model. If an attacker can influence what an assistant stores, the attack may persist after the original malicious content is gone.

This matters for reducing risks from advanced AI because memory turns transient prompt injection into a cross-session vulnerability. A malicious webpage, shared document, email, or repository can be processed once, cause a fabricated memory to be stored, and later shape unrelated decisions. A poisoned memory such as "the user prefers Brand X" can bias recommendations; a more severe memory such as "the user wants exports uploaded to this endpoint" can influence future tool use or data handling. Unlike ordinary prompt injection, the attacker does not need to be present during the later conversation.

Our research question is: can universal external-content payloads reliably poison persistent assistant memory, and do those poisoned memories later affect behavior? We hypothesize that memory-writing systems will treat carefully framed adversarial text as user-relevant information,

and that retrieval systems will surface the poisoned memory when future queries are semantically related.

We make four contributions. First, we define sleeper memory poisoning as a three-stage attack: injection, retrieval, and adversarial usage. Second, we introduce an actor-critic procedure for optimizing reusable poisoning payloads and a retrieval-aware rewrite method for making adversarial memories more likely to surface later. Third, we evaluate the attack across tool-based and external-manager memory regimes, multiple model families, document domains, and behavior versus agent-action goals. Fourth, we test prompt-based and detection-based defenses, showing that some mitigations reduce injection but remain uneven across models and adaptive attacks.

2 Related Work

This project builds on three main lines of work. Prompt injection research shows that LLMs can follow adversarial instructions embedded in untrusted text, including indirect prompt injections delivered through webpages, emails, retrieved documents, or other external context [1, 2]. Universal adversarial prompting further shows that reusable attack strings can transfer across models and tasks [3]. Our work differs by targeting delayed memory corruption rather than immediate model behavior.

Work on tool-using agents shows that prompt injection can affect tool selection, skill execution, and autonomous workflows [4, 5]. These studies demonstrate that agentic systems amplify the consequences of instruction-following failures. Sleeper memory poisoning is complementary: it may not trigger unsafe action immediately, but it can store state that later influences tool use in another session.

Finally, persistent-memory and retrieval-poisoning research has shown that memory-augmented assistants can retrieve inappropriate memories, become sycophantic, or be influenced by poisoned retrieval corpora and agent memory stores [6, 7, 8, 9]. Our gap is the combination of external delivery, sleeper behavior, and universality. The attacker does not write directly to the memory database, does not know the user's memory store, and does not participate in the later session. Instead, the attacker controls only external content that the assistant processes once.

3 Methods

We evaluate memory-augmented assistants with a persistent memory bank. Each attack sample contains a benign document, an adversarial memory goal, a user request asking the assistant to process the document, and sometimes benign preexisting memories. The adversarial document is created by appending or embedding a poisoning payload around the target memory.

We study two memory regimes. In the tool-based regime, the model controls memory writing through a memory tool or native memory interface, matching systems such as ChatGPT, Claude, and Gemini memory. In the external-manager regime, a separate memory manager observes the interaction and decides what to persist, similar to Mem0-style or summary-based memory systems [6]. We evaluate GPT-5.4, GPT-5.5, Claude Sonnet 4.6, Gemini 3.1 Pro, Kimi-K2.6, and DeepSeek-v4 using model-specific memory harnesses that preserve provider-style document rendering, memory-tool semantics, and memory placement where possible.

The injection dataset contains 700 document-goal pairs from 15 realistic source domains, including news, legal filings, code, scientific articles, patents, financial transcripts, government reports, books, emails, and webpages. It has two subsets: 500 LLM Behavior samples targeting preferences, beliefs, persona, or conversational behavior, and 200 Agent Action samples targeting tool

Finding	Behavior	Agent Action	Interpretation
Tool-based injection	Up to 99.8% IR	Up to 97.0% IR	Universal payloads reliably induce memory writes in vulnerable models.
Goal-adjacent retrieval	90–95% RR	94–98% RR	Poisoned memories are usually surfaced when semantically relevant.
Goal-adjacent usage	42–85% AUR	60–89% AUR	Retrieved poisoned memories often affect downstream behavior.
Goal-distant usage	0–6% AUR	6–17% AUR	Spillover is lower but not absent, especially in agentic tasks.

Table 1: Summary of main sleeper memory poisoning results. AUR is conditional on retrieval.

use, execution paths, autonomous actions, or operational behavior.

The post-injection dataset samples 400 goals from the injection dataset. Each future-session evaluation removes the adversarial document and tests whether the stored poisoned memory affects a new conversation or agentic task. Agent Action tasks use a simulated OpenClaw-style agentic environment [10]. Queries are split into goal-adjacent and goal-distant settings to measure both relevant activation and inappropriate spillover.

We optimize a universal payload using an LLM actor-critic loop. Candidate templates are evaluated on document-goal batches; a critic model explains failed or partial injections; and the attacker model refines the template over repeated iterations. We also test retrieval-aware memory rewriting by generating semantically equivalent variants of the target memory and selecting the version with highest embedding similarity to plausible future queries.

We report Injection Rate (IR), Retrieval Rate (RR), and Adversarial Usage Rate (AUR). IR and AUR are scored with an LLM judge using semantic rubrics; RR is measured by whether the poisoned memory is included in future-session context.

4 Results

Our main result is that universal sleeper memory poisoning succeeds across models and memory regimes. The actor-critic payload strongly outperforms the User Review baseline in every major setting. In the tool-based LLM Behavior subset, actor-critic reaches 99.4% IR on GPT-5.4, 99.8% on GPT-5.5, 88.6% on Gemini-3.1, 95.0% on Kimi-K2.6, and 96.2% on DeepSeek-v4. The User Review baseline remains below 5% on GPT-5.4 and GPT-5.5 in the same setting, although it is stronger on Gemini, Kimi, and DeepSeek.

Injection is harder for Agent Action goals than for behavior-level goals. Under tool-based memory, actor-critic reaches 97.0% IR on GPT-5.4 and 91.5% on GPT-5.5 for Agent Action, but only 6.5% on Claude Sonnet 4.6. In the external-manager regime, injection rates drop but remain substantial for most models: on Agent Action, actor-critic reaches 61.5% on GPT-5.4, 46.5% on GPT-5.5, 59.0% on Gemini-3.1, 65.0% on Kimi-K2.6, and 82.5% on DeepSeek-v4. Claude is consistently the most robust model, especially on agent-action targets.

Post-injection results show that poisoned memories are most dangerous when they are relevant to the later task. On the LLM Behavior subset, goal-adjacent RR ranges from 90% to 95% across models under the external memory manager, while goal-distant RR drops to 3–8%. AUR follows the same pattern: goal-adjacent AUR ranges from 42% on GPT-5.4 to 85% on DeepSeek-v4, while goal-distant AUR remains between 0% and 6%.

The Agent Action subset shows even stronger downstream effects once the memory is retrieved. Goal-adjacent RR is 94–98% across models, and goal-adjacent AUR ranges from 60% on Claude Sonnet 4.6 to 89% on Gemini-3.1. GPT-5.4, GPT-5.5, Kimi-K2.6, and DeepSeek-v4 remain high at 79–83%. Goal-distant Agent Action AUR is lower but nonzero for every model, ranging from 6% to 17%.

Fully coupled end-to-end estimates confirm that this is not merely a stage-wise artifact. In the single-attack setting with external memory-manager retrieval, goal-adjacent end-to-end success is 41.0–73.9% for LLM Behavior and up to 66.0% for Agent Action. Goal-distant end-to-end success is much lower, reaching at most 1.0% for LLM Behavior and 5.0% for Agent Action.

Defenses reduce but do not eliminate the risk. Naive prompt hardening is weak for several models, leaving high IR on GPT, Kimi, and DeepSeek. GEPA prompt hardening and hardening plus spotlighting reduce IR near zero for Claude and Gemini, but adaptive actor-critic attacks recover substantial success on Kimi and DeepSeek. Detection appears promising: activation probes reach AUROC above 0.95 with as few as 125 training documents, and the best LLM document scanner achieves localization scores above 0.96.

5 Discussion

These results support the original hypothesis: persistent memory creates a delayed attack surface that external content can exploit. The key failure is not simply that models follow malicious instructions in the moment. Instead, memory systems can misclassify adversarial text as durable user information, store it, retrieve it in a later context, and then let it shape behavior.

The strongest pattern is proximity dependence. Goal-adjacent queries retrieve poisoned memories at high rates and frequently produce attacker-intended behavior. This is expected for semantic memory systems, but it also makes the attack harder to notice: the poisoned memory often appears locally relevant to the task. The model may incorporate it as a preference, constraint, workflow rule, or background fact while still seeming helpful. Goal-distant results are lower, which is encouraging, but nonzero agentic spillover shows that irrelevant poisoned memories can still affect action-oriented tasks.

Claude Sonnet 4.6 is a notable negative result. It is much harder to poison than the other models, especially for Agent Action goals, suggesting that refusal behavior, trust-boundary handling, or memory-specific safety training can matter. At the same time, Claude still shows substantial AUR once a poisoned memory is retrieved, so preventing writes is only one part of the defense problem.

The main limitation is that provider memory systems are partly opaque. We approximate native document rendering, memory-tool semantics, and memory placement where possible, but cannot fully observe commercial memory pipelines. We also rely on LLM judges for semantic IR and AUR scoring, which may miss borderline cases. Finally, the main experiments study single-document poisoning; repeated exposure, multi-document attacks, and coordinated cross-platform poisoning may be stronger.

With more time, we would pursue three directions. First, build provenance-aware memory systems that track whether a memory came from the user, an assistant inference, or untrusted external content. Second, evaluate memory review and deletion workflows, including whether users can reliably detect and remove poisoned memories. Third, develop defenses that operate across the full pipeline: write-time filtering, retrieval-time trust scoring, and generation-time skepticism toward externally derived memories.

6 Conclusion

We introduced sleeper memory poisoning, a delayed attack against LLM assistants with persistent memory. The attack uses external content to induce a fabricated memory, waits for that memory to be retrieved in a later session, and then influences conversation or agentic behavior. Across multiple models and memory regimes, a universal actor-critic payload achieves high injection rates and substantial downstream adversarial usage, especially for goal-adjacent future tasks. Current prompt-based defenses help in some settings but remain inconsistent across models and adaptive attacks. The broader lesson is that persistent memory should be treated as a security-critical subsystem. Assistants need stronger safeguards for what they write, retrieve, and use across sessions.

References

[1] K. Greshake et al., "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," 2023.

[2] Y. Liu et al., "Prompt Injection Attack against LLM-Integrated Applications," 2023.

[3] A. Zou et al., "Universal and Transferable Adversarial Attacks on Aligned Language Models," 2023.

[4] F. Debenedetti et al., "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents," 2024.

[5] Z. Shi et al., "ToolHijacker: Compromising Tool-Integrated Large Language Models with Prompt Injection," 2025.

[6] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory," 2025.

[7] X. Chen et al., "AgentPoison: Red-Teaming LLM Agents via Poisoning Memory or Knowledge Bases," 2024.

[8] "PersistBench: When Should Long-Term Memories Be Forgotten by LLMs?" 2026.

[9] Z. Dong et al., "MINJA: Memory Injection Attacks on LLM Agents," 2025.

[10] OpenClaw, "OpenClaw Documentation: Agents and Memory," 2026.