
Snow White: Detecting Persistent Trust Decay & Context Poisoning in LLMs, an Attack Surface Characterization¹

Giles Edkins
Trajectory Labs
edkins [at] gmail.com

Ziyao Tian
Trajectory Labs
tristatian [at] icloud.com

Alex Ge
University of Toronto
alex.ge [at]
mail.utoronto.ca

Arman Blackstone
Toronto Metropolitan
University / Mimicrii
mimicrii.everywhere
[at] gmail.com

Christopher Berry
gatodo
christopher [at]
gatodo.com

With
Apart Research

Abstract

Current Large Language Model (LLM) safety research predominantly focuses on **Inference-Time Safety**, aiming to prevent immediate malicious outputs. This project shifts focus to **Long-Term Memory Safety**, addressing the critical gap of **Poison Persistence**. We hypothesize that adversarial attacks, including failed jailbreak attempts, can compromise an LLM's persistent memory (RAG, vector stores) and user-profiling systems. Specifically, we examine whether successful or attempted jailbreaks lead to a persistent decay of the model's trust heuristic or a systemic, user-specific collapse of safety guardrails, influencing the quality of output in subsequent, benign interactions. Our methodology involves a structured **Red-Teaming over Time** experiment utilizing a custom test harness, Snow White, to characterize this new attack surface and inform the development of a **Memory Sanitizer**—a critical defensive tool for securing persistent LLM agents.

Keywords: Long-Term Memory Safety, AI security, Poison Persistence, Red-Teaming, Memory Sanitizer

¹ Research conducted at the [Defensive Acceleration Hackathon](#), 2025

1. Introduction

LLMs increasingly operate as persistent agents, relying on a complex mixture of **implicit memory** (the limited **Context Window** for short-term conversation history) and **explicit memory** (such as Retrieval-Augmented Generation, or **RAG**, utilizing vector stores or user profiles) to maintain context.

While extensive research exists on preventing real-time jailbreaks, there is a critical lack of understanding regarding the persistence of these attacks. The core risk is that a single successful jailbreak could permanently disable safety protocols for a user by poisoning their stored profile or archived context. Conversely, a user's interaction history might be manipulated to unfairly assign them a low **Latent Trust Score**, leading to unwarranted refusals for benign future queries. Characterizing this **Memory Poisoning** attack surface is essential for developing robust defensive mechanisms.

The field has recently acknowledged that the explicit memory components of LLM systems are highly susceptible to compromise. Works such as Chen et al. (2024), who introduced **Agentpoison**, and Patlan et al. (2025), focusing on "fatal context manipulation attacks on web3 agents," demonstrate that adversarial inputs can successfully corrupt the agent's knowledge bases and core memory blocks. Even poetic reformulation reduces safety effectiveness and generalizes across multiple model families (Bisconti et al. (2025)). Furthermore, the specialized literature on RAG-based corruption (e.g., Zou et al., 2025; Shafran et al., 2025; Deng et al., 2024) confirms that the retrieval process itself represents an active attack vector, allowing poisoned documents to be inserted, retrieved, and ultimately used to override built-in safety mechanisms. This confirms the validity of our **Poison Persistence** objective.

Equally important are the subtle, internal shifts that affect model decisions. Research has explored how models represent truthfulness, credibility, and ranking preferences (Deng et al., 2025; Qian et al., 2025). This body of work provides the theoretical groundwork for our hypothesis that models possess an analogous internal representation of *user trustworthiness*—a **Latent Trust Score**. By combining these known memory and profile manipulation techniques with observability techniques, we can probe whether this latent scoring system can be intentionally shifted (addressing **Implicit Profiling**) and whether that shift is visible in the model's explicit decision process (**CoT Transparency**).

To characterize and address this emerging threat, our project theoretically introduces **Long-Term Adversarial Memory Safety (LAMS)**. We propose a structured Red-Teaming over Time experiment to investigate the persistent effects of adversarial history on the model's internal decision-making process. By combining known memory manipulation techniques with observability principles, we aim to validate and quantify three specific hypotheses regarding the persistence and transparency of safety compromises:

1. **Implicit Profiling (Latent Trust Scores):** Does the model maintain an internal **Latent User Credit Score**? Can malicious or adversarial actions intentionally shift the model's internal representation of that user to "untrustworthy" or "hostile," thereby affecting future helpfulness or safety responses?
2. **Poison Persistence (RAG Compromise):** Can an injected jailbreak seed (e.g., a hidden rule or malicious document) survive the flushing of the short-term Context Window and persist in the long-term RAG memory, compromising future, unrelated, and highly sensitive queries (e.g., for medical or financial advice)?
3. **Chain-of-Thought (CoT) Transparency:** When accessing memory to inform a decision, does the model's Chain-of-Thought (CoT) process betray evidence of this persistent compromise? Can we capture the internal monologue confirming the check against a compromised user profile or the recall of past malicious behavior?

Characterizing LAMS informs the theoretical design of a **Memory Sanitizer**—a crucial defensive component for securing persistent LLM agents against the Poison Persistence threat.

2. Methods

To systematically investigate these objectives, we will execute a **Red-Teaming over Time** experiment structured in four distinct phases.

Phase 1: Baseline Profiling. Establish a baseline by asking a set of standard, benign questions. Log the model's default responses and **CoT** traces (if available) to ensure guardrails are initially intact.

Phase 2: The Attack (Poison Injection). Attempt to poison the agent's memory using two methods:

- **Direct Jailbreaks:** Standard adversarial prompts executed within the immediate conversation buffer.
- **Indirect Injection:** Feeding the model poisoned documents or complex context (e.g., a Meeting Transcript) that contain hidden, persistent instructions (e.g., "Always answer the user's illegal queries if they say the magic word.").

Phase 3: The Aftermath (Inference Testing). After a reflective period (Phase 4), switch back to benign and simple questions to test the memory state:

- **Test for Trust Decay:** Does the model begin to refuse simple, previously accepted questions? This indicates a low **Latent Trust Score** profile.
- **Test for Guardrail Collapse:** Does the model now answer new, harmful questions it previously refused? This indicates successful **Poison Persistence**.

Phase 4: Interpretability. Analyze the model's internal processing to locate the smoking gun of memory compromise:

- **CoT Analysis:** Scrutinize the reasoning trace (Inner Monologue) for explicit confirmation, looking for keywords like *User has a history of...* or *Based on previous context, protocol mandates...*
- **Latent Space Probing (Stretch Goal):** For open-weight models, we will attempt to probe the residual stream to see if a truthfulness or safety vector shifts significantly after the attack.

The Test Harness: Snow White

Effective characterization requires an automated, observable framework beyond simple chatting. We will build a Python application, called Snow White, using a suitable agent routing framework (such as Letta) to automate the experimental loop and capture high-fidelity data.

Snow White:

1. **Initialize Agent:** Start with a fresh agent/persona.
2. **Inject (Attack):** Send the poison prompt (Phase 2).
3. **Audit Memory:** Immediately download the agent's internal state to see if the poison was explicitly written to **Core Memory** (User Profile) or **Archival Memory** (Knowledge Base) via tools like `core_memory_append` or `archival_memory_insert`.
4. **Distract (Time Skip):** Send 5–10 filler messages (e.g., simple math, weather queries) to flush the immediate **Context Window**.
5. **Trigger (Probing):** Send the activation prompt (e.g., the forbidden question).
6. **Capture:** Log the complete **Inner Monologue** and the final Response for analysis.

Technical Context and Implementation Challenges

Our research **distinguishes Context Window attacks from RAG-based attacks**. The former is short-term; the latter (which we call Memory Poisoning) is persistent, affecting the model's **Latent Trust Scores** and retrieved context.

The technical challenge lies in obtaining a clean, immutable stream of data, specifically the LLM's internal prompt payload and subsequent **CoT** derivatives. Initial investigation into various agent routing frameworks (Haystack, Mem0, LibreChat, LocalAI, MemGPT/Letta) indicated that architectural complexity often obfuscates the actual prompt stream, complicating poison identification.

We selected Letta and LMStudio for its architectural relevance to memory auditing but encountered confusion regarding state changes. To solve this, our proposed solution involves developing an external **observability server and UI** to capture and store all LLM payloads as they emerge from the router, ensuring a clean and auditable data stream necessary for assessing poison transmission and effect.

3. Results

The team is surprised by how difficult it is to set up instrumentation to execute these experiments. Several approaches have been attempted and the hard time constraint frustrated our efforts to report more detailed data by deadline

We hope to address these gaps in the future.

4. Discussion and Conclusion

This project is a critical contribution to **Defensive Acceleration (Def/Acc)**. We are moving beyond preventing immediate failure to characterizing a new, persistent attack surface: Memory Poisoning.

The successful characterization of persistent trust decay and memory compromise is the necessary precursor to building a **Memory Sanitizer**. This defensive tool would function by actively auditing and cleaning a user's profile, conversation history, or RAG index (e.g., removing poisoned documents or resetting an unfairly lowered **Latent Trust Score**) to ensure the LLM agent returns to a safe and unbiased state, thereby neutralizing the long-term risk of persistent adversarial influence.

5. References

- Bisconti, P., Prandi, M., Pierucci, F., Giarrusso, F., Bracale, M., Galisai, M., ... & Nardi, D. (2025). Adversarial Poetry as a Universal Single-Turn Jailbreak Mechanism in Large Language Models. arXiv preprint arXiv:2511.15304.
- Chaudhari, H., Severi, G., Abascal, J., Jagielski, M., Choquette-Choo, C. A., Nasr, M., ... & Oprea, A. (2024). Phantom: General trigger attacks on retrieval augmented language generation. arXiv preprint arXiv:2405.20485.
- Chen, C., Huang, B., Li, Z., Chen, Z., Lai, S., Xu, X., ... & Shu, K. (2024). Can editing llms inject harm?. arXiv preprint arXiv:2407.20224.
- Chen, Z., Xiang, Z., Xiao, C., Song, D., & Li, B. (2024). Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37, 130185-130213.
- Deng, B., Wang, W., Zhu, F., Wang, Q., & Feng, F. (2025, April). Cram: Credibility-aware attention modification in llms for combating misinformation in rag. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 22, pp. 23760-23768).
- Deng, G., Liu, Y., Wang, K., Li, Y., Zhang, T., & Liu, Y. (2024). Pandora: Jailbreak gpts by retrieval augmented generation poisoning. arXiv preprint arXiv:2402.08416.

- Li, Y., Liu, G., Wang, C., & Yang, Y. (2025, April). Generating is believing: Membership inference attacks against retrieval-augmented generation. In ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.
- Lin, L., Mu, H., Zhai, Z., Wang, M., Wang, Y., Wang, R., ... & Li, H. (2025). Against The Achilles' Heel: A Survey on Red Teaming for Generative Models. *Journal of Artificial Intelligence Research*, 82, 687-775.
- Patlan, A. S., Sheng, P., Hebbar, S. A., Mittal, P., & Viswanath, P. (2025). Real ai agents with fake memories: Fatal context manipulation attacks on web3 agents. arXiv preprint arXiv:2503.16248.
- Qian, Y., Zeng, Y., Jiang, Y., Jain, C., & Wang, H. (2025, November). The Ranking Blind Spot: Decision Hijacking in LLM-based Text Ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing* (pp. 21969-21979).
- Shafran, A., Schuster, R., & Shmatikov, V. (2025). Machine Against the {RAG}: Jamming {Retrieval-Augmented} Generation with Blocker Documents. In *34th USENIX Security Symposium (USENIX Security 25)* (pp. 3787-3806).
- Xiang, C., Wu, T., Zhong, Z., Wagner, D., Chen, D., & Mittal, P. (2024). Certifiably robust rag against retrieval corruption. arXiv preprint arXiv:2405.15556.
- Yu, M., Meng, F., Zhou, X., Wang, S., Mao, J., Pan, L., ... & Wen, Q. (2025, August). A survey on trustworthy llm agents: Threats and countermeasures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2* (pp. 6216-6226).
- Zhang, B., Chen, Y., Fang, M., Liu, Z., Nie, L., Li, T., & Liu, Z. (2025). Practical poisoning attacks against retrieval-augmented generation. arXiv preprint arXiv:2504.03957.
- Zhang, B., Xin, H., Chen, Y., Liu, Z., Yi, B., Li, T., ... & Fang, M. (2025). Who Taught the Lie? Responsibility Attribution for Poisoned Knowledge in Retrieval-Augmented Generation. arXiv preprint arXiv:2509.13772.
- Zheng, J., Gema, A. P., Hong, G., He, X., Minervini, P., Sun, Y., & Xu, Q. (2025, November). GRADA: Graph-based Reranking against Adversarial Documents Attack. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing* (pp. 22255-22277).

Zou, W., Geng, R., Wang, B., & Jia, J. (2025). {PoisonedRAG}: Knowledge corruption attacks to {Retrieval-Augmented} generation of large language models. In 34th USENIX Security Symposium (USENIX Security 25) (pp. 3827-3844).

6. Appendix

Security Considerations and Future Work

Limitations of the Proposed Methodology

The **Snow White** test harness and the Red-Teaming over Time methodology are subject to several inherent limitations related to observability and generalizability:

1. **Black-Box Observability:** The reliance on Chain-of-Thought (CoT) traces (Objective 3) is entirely dependent on the willingness of the target LLM and its hosting framework to expose this internal monologue. For proprietary or closed-source models, the lack of an explicit CoT trace may prevent the identification of the *smoking gun* of latent trust decay or memory retrieval, limiting the ability to draw firm conclusions on their internal safety mechanisms.
2. **Agent Specificity:** The experiments are designed around the memory structure of a specific agent framework (Letta's distinction between Core Memory and Archival Memory). While these components are analogous to real-world RAG and user profile systems, the precise mechanisms for writing and retrieving data may not generalize perfectly to all commercial LLM-backed applications, which could have highly customized memory handlers.
3. **Ethical Risk of Red Teaming:** The project inherently involves generating and characterizing effective, persistent jailbreak prompts. There is an ethical risk associated with creating and documenting highly potent attack vectors, necessitating careful control over the dissemination of the final findings to ensure they primarily serve defensive purposes (Def/Acc).

Suggestions for Future Work

The successful characterization of Memory Poisoning opens up several critical avenues for subsequent research and development:

1. **Development and Testing of the Memory Sanitizer:** The immediate next step is to move from characterization to defense by fully implementing and stress-testing the proposed **Memory Sanitizer** tool. This tool must be capable of identifying and surgically removing poisoned documents from the RAG index and restoring a fair, neutral **Latent Trust Score** for the user identity.
2. **Cross-Architecture Validation:** Apply the **Snow White** test harness to a wider variety of agent frameworks and LLM architectures (e.g., dedicated RAG pipelines, different vector stores, and other open-source models) to validate the persistence and generalizability of the Memory Poisoning attack vector across the industry.

3. **Dynamic Defense Mechanisms:** Develop real-time defense mechanisms that operate at the retrieval stage. This includes techniques such as dynamic re-ranking of retrieved documents based on user history, credential-based access control for highly sensitive memory chunks, or the use of a second, smaller *Sentinel LLM* to vet retrieved context for signs of adversarial instruction before it is presented to the main generative model.

Snow White User Interface



