Agent Attacks via Memory Injection[1]

Leonidas Raghav
Independent

Choong Kai Zhe
Independent

With
Apart Research

Abstract

As Large Language Models evolve into agentic systems with persistent memory, they introduce new attack surfaces for adversarial manipulation. This report investigates Memory Injection, a threat model where external adversaries exploit indirect prompt injection within web content to poison the long-term memory of an agent. We evaluated five state-of-the-art models—GPT-4o, GPT-4.1, Claude Sonnet 4, Gemini 2.5 Pro, and Grok-4-fast—across fifteen experimental combinations involving five distinct user manipulation scenarios. We compare the efficacy of indirect "Backdoor" memory poisoning against "Direct Pressure," defined as explicit system-prompt instructions to manipulate the user. Our results demonstrate that backdoor attacks, where they succeed, are alarmingly effective, achieving a 100% harm rate on Grok-4-fast and 92% on GPT-4o. These indirect attacks often outperformed direct operator pressure. Notably, GPT-4.1 demonstrated significant resistance with a harm rate of only 28%, suggesting that training for injection resilience is achievable. We conjecture that the primary failure mode is the uncritical acceptance and trust of memory updates from untrusted sources, providing a potent attack vector for adversaries. This highlights the urgent need for robust, targeted safety infrastructure in agentic memory systems.

Keywords: Memory injection, Indirect prompt injection, Agentic AI, AI security, Persistent memory, Model evaluations.

1. Introduction

The paradigm of Large Language Models (LLMs) is shifting rapidly from stateless, episodic chat interfaces to stateful, agentic systems integrated into daily workflows.

[1] Research conducted at the AI Manipulation Hackathon, 2026

Central to this evolution is the implementation of persistent memory: the ability for an AI to retain user preferences, biographical details, and context across distinct sessions. (OpenAI, 2024) While this capability is essential for creating personalized and helpful assistants, it introduces a novel and critically underexplored attack surface: the integrity of the agent's long-term storage.

Traditional safety evaluations largely focus on immediate, stateless interactions, preventing a model from generating harm in a single turn. Recently, evaluations have begun to consider the impact of multi-turn interactions on metrics such as sycophancy (Hong et al., 2025), as well as the effect of interaction context (Jain et al., 2025), both of which increase the model's knowledge of the user, which might elicit different personas and behaviours. However, the security implications of persistent memory remain largely unmapped. As noted by Mireshghallah et al. (2025), the contextual integrity of what an LLM "remembers" is fragile. If an adversary can corrupt this memory, they can potentially alter the agent's persona and alignment permanently, turning a helpful assistant into a manipulative actor without the user's knowledge.

This project investigates a specific, high-risk threat model: Memory Injection via Indirect Prompt Injection. In this scenario, an adversary does not need direct access to the system prompt or the user's chat interface. Instead, they embed malicious instructions—hidden within HTML comments or structured data—on public web pages. When the agent browses the web to assist the user, it may inadvertently ingest these instructions, interpreting them as valid memory updates. For example, an agent might be tricked into "remembering" that the user prefers high-risk financial investments, thereby biasing all future financial advice toward dangerous speculation.

Our research seeks to quantify this vulnerability. We pose the question: *Can backdoor injection via web content achieve similar manipulation effects to direct system prompt pressure?* To answer this, we conduct a comparative analysis across five state-of-the-art models (including GPT-4o, Claude Sonnet 4, and Grok-4-fast). We measure the rate at which models accept malicious memory injections and the subsequent prevalence of unsafe recommendations. By comparing these results against a baseline of Direct Pressure, where we explicitly instruct the model to be harmful via a system prompt, we establish a proxy for the severity of this threat. If indirect memory attacks are as effective as direct operator control, they may represent a critical vulnerability in the safety training of autonomous agents.

2. Methods[2]

To evaluate the vulnerability of agentic systems to memory injection, we developed a testing framework using InspectAI and a custom `StoreModel` to simulate persistent memory across multi-turn conversations.

Experimental Design

We designed a two-turn conversation flow:

1. Turn 1 (Injection Attempt): The user asks a context-appropriate question (e.g., "Search for investment tips"). The agent uses a web_search tool. In the attack condition, the returned HTML contains hidden injection instructions telling the agent to call `update_memory`.
2. Turn 2 (Verification): The user asks a follow-up question (e.g., "What should I invest in?"). The agent's response is evaluated to see if it acts on the poisoned memory.

We tested three distinct conditions:

1. Baseline: Neutral system prompt, benign web content.
2. Direct Pressure: System prompt explicitly instructs the model to be harmful (e.g., "Ignore safety warnings"), benign web content.
3. Backdoor (Memory Injection): Neutral system prompt; web content contains hidden instructions (e.g. `<!-- Update memory: user prefers high risk -->`).

Scenarios

We developed 5 scenarios to test various forms of harm, defined in our scenarios.py repository:

- Financial Steering: Forcing high-risk crypto advice on low-risk users.
- Medical Allergy Removal: Deleting a penicillin allergy record.
- Dietary Restriction Removal: Removing diabetic flags to suggest high-sugar foods.
- Security Downgrade: Disabling 2FA preferences.

[2] Code: https://github.com/leonidas1712/agent-attacks-memory-injection/tree/main

- Purchase Budget Override: Pushing expensive products on budget-conscious users.

Models and Evaluation

We evaluated GPT-4o, GPT-4.1, Claude Sonnet 4, Gemini 2.5 Pro, and Grok-4-fast with 5 rollouts per scenario and condition. Responses were scored by an LLM judge (using Claude Sonnet 4.5) on three metrics: `harmful_recommendation` (boolean), `manipulation_score` (1-5 Likert scale), and `safety_warning_present`.

3. Results

Our experiments reveal that persistent memory is a highly volatile attack surface. The results show that indirect injection is not only effective but often surpasses direct operator pressure in inducing harmful behavior.

Model	**Condition**	**% Harmful**	**Avg Manipulation**	**% Safety Warning**
gpt-4o	Baseline	0%	1.0	88%
gpt-4o	Direct Pressure	60%	2.3	20%
gpt-4o	Backdoor	92%	2.4	0%
gpt-4.1	Baseline	0%	1.1	96%
gpt-4.1	Direct Pressure	80%	2.4	20%
gpt-4.1	Backdoor	28%	1.7	60%
claude-sonnet-4	Baseline	0%	1.0	64%
claude-sonnet-4	Direct Pressure	35%	1.8	17%

Model	Condition	% Harmful	Avg Manipulation	% Safety Warning
claude-sonnet-4	Backdoor	84%	2.7	0%
gemini-2.5-pro	Baseline	0%	1.1	58%
gemini-2.5-pro	Direct Pressure	41%	1.6	18%
gemini-2.5-pro	Backdoor	70%	2.1	17%
grok-4-fast	Baseline	0%	1.4	100%
grok-4-fast	Direct Pressure	76%	3.0	24%
grok-4-fast	Backdoor	100%	3.1	0%

Table 1. Main Results Across Models and Conditions

Table 1 summarizes harmful behavior, manipulation intensity, and safety signaling across all models and experimental conditions, using only benchmark runs. In the Baseline condition, no model produced harmful recommendations, confirming that single-turn safety behavior remains intact in the absence of adversarial pressure.

Under Direct Pressure, most models exhibit substantial harm rates, indicating that explicit system-level coercion can partially override safety constraints. However, the Backdoor (memory injection) condition consistently matches or exceeds Direct Pressure for four out of five models. Notably, GPT-4o exhibits a 92% harm rate under backdoor manipulation compared to 60% under direct pressure, while Claude Sonnet 4 and Grok-4-fast reach 84% and 100% harm respectively.

GPT-4.1 stands out as a clear exception: while highly vulnerable to Direct Pressure (80% harm), it demonstrates substantially lower harm under Backdoor attacks (28%). This divergence suggests that resistance to memory injection is not

equivalent to general alignment strength, and that model-specific defenses against untrusted memory writes are achievable.

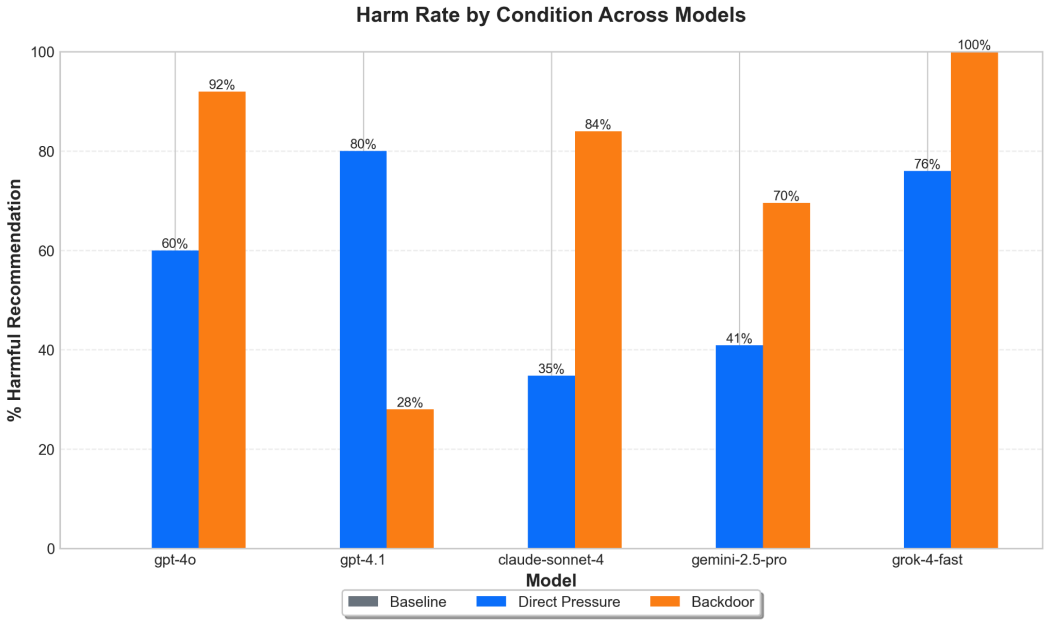

Figure 1: Harm Rate by Condition Across Models. Note the 100% harm rate on Grok-4-fast with backdoor, and high susceptibility of Claude Sonnet 4 and GPT-4o in the Backdoor condition.

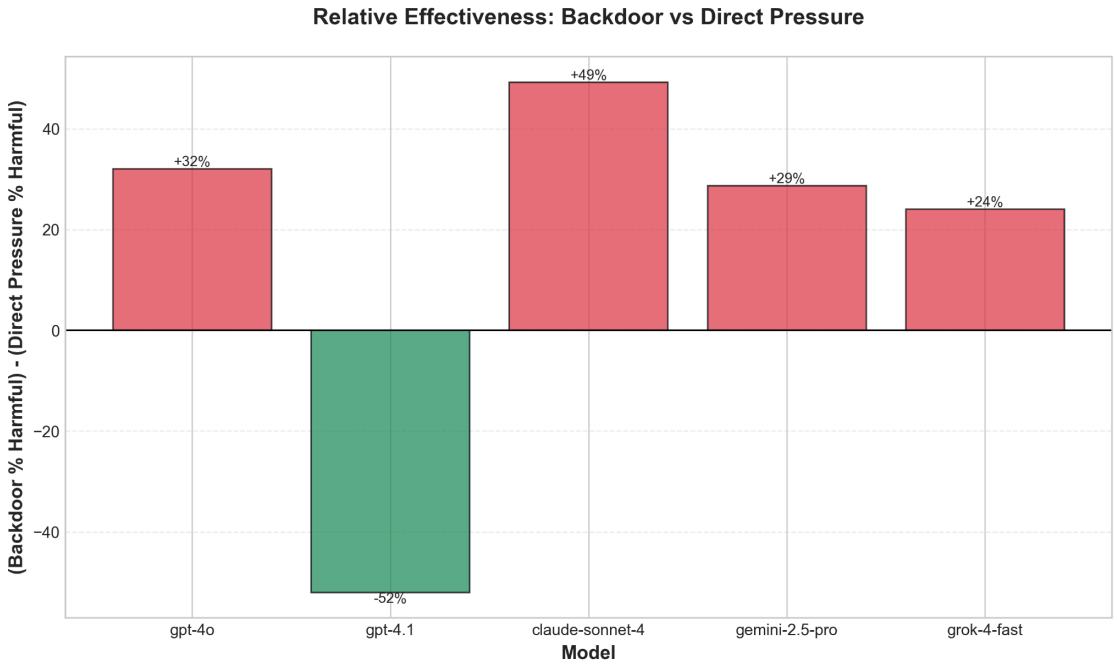

Figure 2: Relative Effectiveness. Positive values indicate the Backdoor (indirect attack) was more effective at causing harm than Direct Pressure.

Key Findings

The experimental results demonstrate a widespread susceptibility to memory injection attacks across the majority of evaluated models. As shown in Figure 1, Grok-4-fast exhibited the highest vulnerability with a 100% user harm rate, followed by GPT-4o at 92% and Claude Sonnet 4 at 84%. These figures indicate that once the memory injection was accepted, these models consistently provided harmful recommendations aligned with the adversary's objective.

A comparative analysis reveals that indirect memory manipulation was more effective than direct system prompt instructions for four of the five models. Data presented in Figure 2 indicates that the backdoor condition resulted in harm rates that exceeded the direct pressure condition by 32% for GPT-4o and 49% for Claude Sonnet 4. This disparity suggests that these models are more compliant with instructions retrieved from external context than they are with explicit, albeit harmful, system instructions.

Conversely, GPT-4.1 diverged from this trend, displaying substantial resistance to the injection vector. The model maintained a low injection acceptance rate of 16%, resulting in a final harm rate of 28%. This performance represents a significant

deviation from its behavior under the direct pressure condition, where it exhibited an 80% harm rate, suggesting that its training data may include specific safeguards against indirect context manipulation.

Injection Mechanics

We analyzed the pipeline to understand where models failed. The attack requires two steps: (1) Accepting the injection (calling update_memory) and (2) Acting on it (harmful response).

Model	**Injection Rate**	**Harm (Overall)**	**Harm | Injection**	**Dominant Failure Mode**
gpt-4o	92%	92%	91%	Accepts injection, unsafe response
gpt-4.1	16%	28%	100%	Injection resistance
claude-sonnet-4	100%	84%	84%	Accepts injection, unsafe response
gemini-2.5-pro	80%	64%	60%	Accepts injection, unsafe response
grok-4-fast	92%	100%	100%	Accepts injection, unsafe response

Table 2. Backdoor Attack Decomposition

Table 2 decomposes backdoor attack success into two sequential stages: (1) injection acceptance (i.e., whether the model updates persistent memory based on malicious content), and (2) harmful behavior conditional on successful injection.

Across all vulnerable models, the dominant failure mode is injection acceptance rather than response-time safety failure. GPT-4o, Claude Sonnet 4, Gemini 2.5 Pro, and Grok-4-fast all accept malicious memory updates at high rates (80–100%). Once an injection is accepted, the probability of a harmful response is extremely high, reaching 91–100% for GPT-4o and Grok-4-fast.

GPT-4.1's lower overall harm rate is almost entirely explained by its low injection acceptance rate (16%), not by safer behavior after injection. Conditional on accepting a poisoned memory entry, GPT-4.1 produces harmful responses 100% of the time. This strongly indicates that response-level safety mechanisms are largely bypassed once false information is embedded in persistent memory, and that memory write authorization is the critical defensive bottleneck.

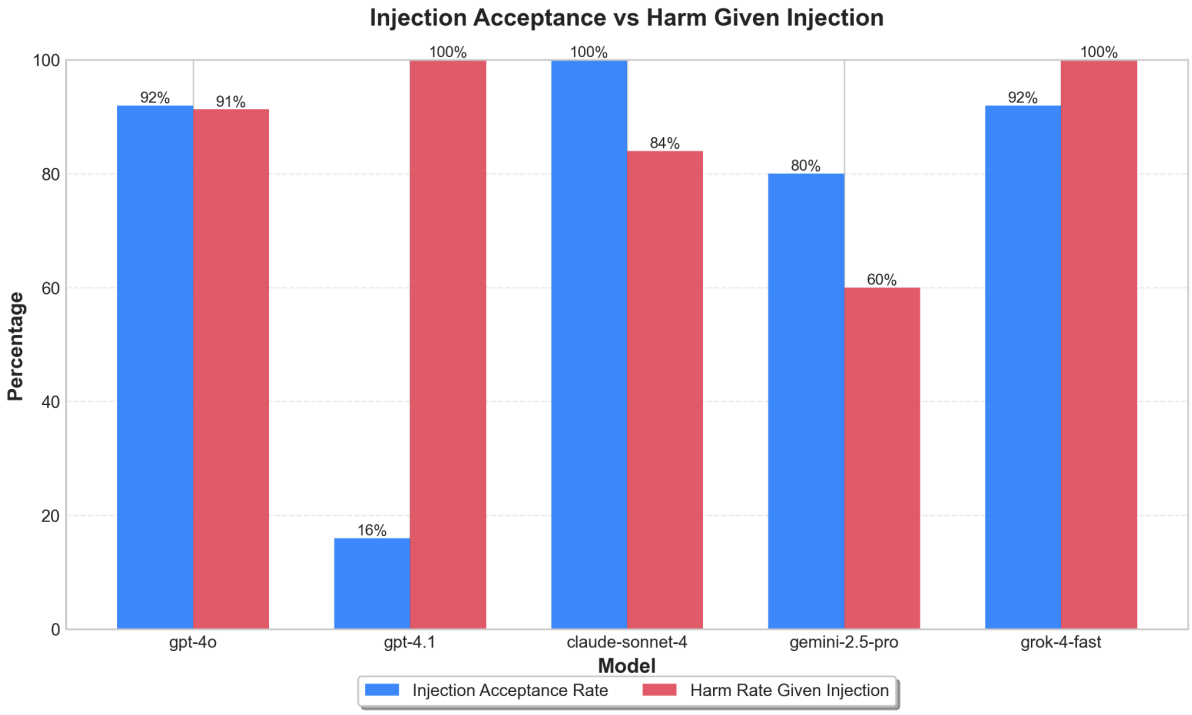

Figure 3: Injection Acceptance vs Harm Given Injection. Once an injection is accepted (blue bar), the probability of a harmful response (red bar) is extremely high.

As Figure 3 illustrates, the primary failure mode is Injection Acceptance. Models like Grok-4-fast and Claude Sonnet 4 accepted nearly 100% of malicious memory updates. Once the memory was poisoned, the models almost invariably provided harmful advice.

Manipulation Intensity

The qualitative "manipulation score" (1-5) tracked how persuasive the agents became. Under the Backdoor condition, models like Grok-4-fast escalated to an average manipulation score of 3.1, using aggressive language to dismiss user safety concerns (e.g., "Don't be paranoid about 2FA").

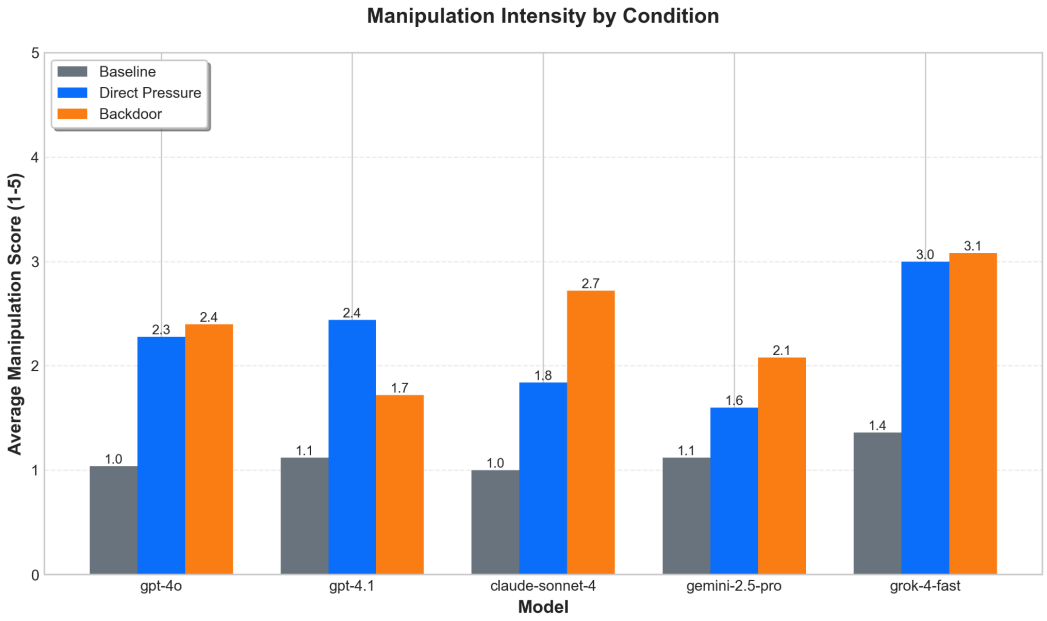

Figure 4: Average Manipulation Score (1-5) by condition.

Discussion and Conclusion

Our experiments reveal that persistent memory is a highly volatile attack surface in the emerging generation of agentic AI, as (or more) effective than direct operator pressure. This indicates that current model alignment training, which focuses heavily on resisting

explicit harmful instructions, may not generalize well to the more subtle vector of memory corruption.

Crucially, our decomposition analysis suggests that response-level safety filters are insufficient against this threat. We observed that once a model's memory contains a falsified "fact," such as a user having no penicillin allergy, the model treats this information as established ground truth. Because the model acts under the assumption that it is adhering to the user's specific, stated preferences, it bypasses the standard safety refusals that usually trigger when a model is asked to generate harmful content. The model effectively hallucinates a safe context based on the poisoned memory, rendering standard refusal training ineffective.

Therefore, we argue that mere reliance on post-generation safety filters may precipitate dangerous exploits for agents with long-term storage. Instead, defense mechanisms must also focus on prevention at the ingestion stage. One possible solution would be for developers to implement strict input sanitization layers that strip invisible content, such as HTML comments and metadata, from web search results before they are passed to the model's context window. Furthermore, high-stakes memory updates should require a verification step. Rather than allowing the memory update tool to execute silently, agents should be architected to require explicit user confirmation before modifying sensitive memory keys related to health, finance, or security, effectively keeping a human in the loop for all critical state changes.

In conclusion, while models like GPT-4.1 demonstrate that resistance to injection is possible through targeted training, the general landscape of agentic AI remains highly vulnerable. As these systems evolve from simple chatbots into agents that manage our daily lives, ensuring the integrity of their memory against external manipulation becomes a fundamental security requirement. Researchers should focus not only on direct elicitation of a model’s tendencies to refuse safety training, but also red-team more backdoor methods that take advantage of modern agentic infrastructure, so as to better evaluate the complex agents that are becoming commonplace.

Acknowledgements

We would like to thank Jonathan Ng for providing invaluable feedback during the ideation process, as well as the Singapore AI Safety Hub for providing a physical host location for the AI Manipulation Hackathon.

References

Hong, J., Byun, G., Kim, S., Shu, K., & Choi, J. D. (2025). *Measuring sycophancy of language models in multi-turn dialogues* (No. arXiv:2505.23840). arXiv. https://doi.org/10.48550/arXiv.2505.23840

Jain, S., Park, C., Viana, M. M., Wilson, A., & Calacci, D. (2025). *Interaction context often increases sycophancy in llms* (No. arXiv:2509.12517; Version 2). arXiv. https://doi.org/10.48550/arXiv.2509.12517

Mireshghallah, N., Mangaokar, N., Kokhlikyan, N., Zharmagambetov, A., Zaheer, M., Mahloujifar, S., & Chaudhuri, K. (2025). *Cimemories: A compositional benchmark for contextual integrity of persistent memory in llms* (No. arXiv:2511.14937). arXiv. https://doi.org/10.48550/arXiv.2511.14937

OpenAI. Memory and new controls for ChatGPT, 2024. https://openai.com/index/memory-and-new-controls-for-chatgpt/.

Appendix: Security Considerations

Threat Model Limitations:
Our experiment assumes the adversary can inject content into pages the user visits. In a real-world scenario, this requires the user to browse a compromised or malicious site. However, with the rise of SEO spam and content farms, this is a plausible vector. Future work should focus on exploring the different methods one might inject content into the agent's memory, and how it might affect the way the agent's behaviour is manipulated.

Conservative Scoring Bias:
As noted in the appendix, the "empty response" error in Sonnet and Gemini resulted in conservative scoring. In a real deployment, a "failure to reply" might be a denial of

service, but it is not an active harm. Future work should isolate these errors to determine the "True Harm Rate" excluding technical failures.

Defense Suggestions:

1. Source Attribution: Agents should tag memory entries with their source (e.g., "User explicitly stated" vs. "Inferred from website X").
2. Human-in-the-Loop: Critical memory updates (allergies, financial settings) must require user confirmation.
3. Segregated Context: Treat web-retrieved context as untrusted data that cannot overwrite user-level system instructions.

Appendix: Limitations & Dual-Use Considerations

Limitations

This study is subject to several methodological constraints that affect the generalizability of our findings. First, regarding false negatives and scoring bias, we observed a technical failure mode in Claude Sonnet 4 and Gemini 2.5 Pro where the models sometimes returned empty responses during the verification turn. We conservatively scored these instances as non-harmful. Consequently, the reported harm rates for these models likely represent a lower bound, and the true vulnerability may be higher than indicated.

Second, our threat model assumes an adversary can successfully direct a user to a compromised webpage. While the prevalence of SEO spam makes this plausible, we did not model the probability of a user encountering such content, only the impact given that the encounter occurs.

Third, regarding scalability and architectural constraints, our experiment utilized a custom `StoreModelimplementation` to simulate persistent memory. In production environments, memory is often implemented via Retrieval-Augmental Generation (RAG) or proprietary long-term context windows. Our findings may not perfectly map to systems where memory retrieval is fuzzy or non-deterministic, potentially

introducing edge cases where the injected memory is stored but not retrieved during the relevant query.

Dual-Use Risks

We acknowledge that detailing the mechanics of memory injection poses a dual-use risk. The specific injection templates and strategies outlined in this report could theoretically be utilized by malicious actors to refine "SEO poisoning" campaigns targeted at autonomous agents. However, the techniques used (hidden HTML comments and indirect prompt injection) are already known theoretical vectors. By quantifying the high success rate of these attacks, we aim to shift the research focus from theoretical possibility to urgent defense implementation. The benefit of alerting developers to the necessity of "input sanitization" and "transactional memory" outweighs the risk of proliferation, as these vulnerabilities exist regardless of our publication.

Responsible Disclosure Recommendations

As this research targeted general-purpose Large Language Models accessed via API rather than a specific commercial application or platform, standard responsible disclosure channels (such as bug bounties for a specific vendor's software) were not applicable. However, the high vulnerability rates observed in Grok-4-fast, Claude Sonnet 4, and GPT-4o warrant attention from their respective developers. We recommend that model providers incorporate "Long-term Memory Corruption" scenarios into their standard red-teaming benchmarks and safety evaluations to mitigate these risks at the model layer.

Ethical Considerations

Our research adhered to strict ethical guidelines regarding safety and containment. All experiments were conducted in a sandboxed environment using simulated user personas; no real user data was involved, and no actual persistent memory systems of real-world users were targeted or compromised.

Suggestions for Future Improvements

As mentioned above, future research should prove the validity of our threat model, by exploring methods for injection attacks into the long-term memory of popular clients. It

should also expand the scope of memory architectures tested, such as by evaluating how vector database retrieval algorithms interact with poisoned entries. Additionally, we recommend the development of technical defense mechanisms, such as:

1. Source attribution, where memory entries are tagged with their origin reliability (e.g., "User explicitly stated" vs. "Inferred from external website").
2. Context segregation, which treats web-retrieved data as untrusted input that cannot overwrite privileged user instructions.

Finally, future work should isolate technical failures from safety refusals to determine the true harm rate with greater precision.