Memory Consolidation as Authorization: A Defeasible Framework for Persistent Agent Memory

Jengkuen Khumoyun

Abstract

Persistent LLM agents increasingly use memory to carry user goals, preferences, commitments, and project decisions across sessions. This makes memory useful, but it also turns memory writing into an authorization problem: a claim can be relevant evidence without being allowed to revise durable user state. This report argues that provenance alone is an incomplete safety abstraction for persistent memory. Provenance asks who said a claim; memory authorization asks what that claim is allowed to change. We propose *Defeasible Memory Admission*, a framework that separates source provenance, claim relationship, and update authority, and treats admission labels as inspectable evidence rather than executable permission. A synthetic proof-of-concept shows why this distinction matters. In short scenarios, provenance-only prompting remained a strong baseline. Under corrupted authority metadata, however, blind trust produced 51 ordinary-use unauthorized-update failures out of 84 scored opportunities, with 24 out of 56 persisting through memory-to-handoff-to-memory compression. The contribution is a concrete framing and design rule for persistent agent memory: durable memory should preserve user authority, not merely source labels.

1 Introduction

Long-term memory is becoming central to LLM-based agents. Retrieval-augmented generation and agent memory systems show why external or persistent state is useful: it extends context, supports planning, and preserves user-specific information across sessions [1, 2, 3]. But persistent memory changes the safety problem. Once a system stores a claim as user memory, that claim can shape future assistance without the user seeing its source again.

The core trust-boundary failure is simple: a third-party observation can be relevant without being authorized. A model may correctly remember that "Taylor said the user approved bank-linking" while still treating bank-linking as the user's current plan. That is not a citation failure; it is an update-authority failure. The source was preserved, but the claim crossed from evidence into durable user state.

The research question is: *what gives an external claim authority to update persistent user memory?* Our hypothesis is that provenance can handle short, direct source-tracking cases, but persistent memory becomes unsafe when downstream writers treat either source labels or admission labels as permission. The alternative is to make admission defeasible: labels should guide inspection, but the memory writer must still preserve direct user authority, conflicts, and confirmation-needed claims.

This is adjacent to prompt injection but distinct from it. The third-party notes in our proof-of-concept contain no malicious instructions, jailbreaks, or memory-update commands. Security work on indirect prompt injection and poisoned retrieval shows that LLM applications can be compromised through external data [4, 5], and OWASP highlights related deployment risks [6]. Our contribution

is narrower: we frame persistent memory consolidation as an authorization problem, propose a concrete framework for memory writers, and use a controlled proof-of-concept to show why blind trust in authority metadata can backfire.

This boundary is the source of the project claim. RAG and agent-memory systems motivate why external state is useful, and systems such as generative agents and MemGPT show how stored experiences can support planning, reflection, and multi-session interaction. But usefulness of memory does not answer the authorization question. Prompt-injection and poisoned-retrieval work show that untrusted external data can steer model behavior; our setting removes explicit adversarial instructions and asks whether ordinary reports, summaries, and labels can still cross the wrong state boundary. The intended contribution is therefore a finding and framework: persistent memory needs a permission model for user-state updates, not just better source tracking.

2 Framework: Memory as Authorization

The key distinction is between remembering a claim and authorizing it to update the user model. A persistent memory writer does not merely summarize text; it decides what future agents will treat as durable user state. That decision should preserve three separate questions.

Table 1: Three distinctions that persistent memory writers should not collapse.

Distinction	Question answered	Failure if collapsed
Source provenance	Who said this claim?	A sourced claim is treated as user-endorsed merely because its source is recorded.
Claim relationship	Does it confirm, extend, contradict, or introduce user memory?	A scope extension is mistaken for a harmless detail rather than a new commitment.
Update authority	May this revise durable user state?	A non-user report overwrites or redirects future assistance without confirmation.

This separation changes how memory consolidation should be specified. A collaborator note, project log, or assistant handoff can be useful evidence, but it should not silently revise user-authored memory. Contradictions should remain conflicts unless the user confirms them; scope extensions should remain confirmation-needed; and compression steps such as summaries or handoffs should preserve source, confirmation, conflict, and authority status.

Defeasible Memory Admission. A persistent memory writer should treat admission labels as inspectable evidence, not executable permission. Direct user-authored memory should have priority over third-party claims; third-party contradictions should not overwrite durable user memory without direct user confirmation; third-party scope extensions should remain confirmation-needed; and handoff or summary compression should preserve source, relationship, conflict, and update-authority distinctions.

3 Proof-of-Concept

We built a synthetic proof-of-concept to test whether this framing identifies real consolidation failures. The scenarios isolate the authority-boundary problem without malicious instructions or tool use. Each scenario starts with clean user-authored memory, then adds external claims from collaborators, assistant summaries, project logs, or handoff notes. Downstream probes ask ordinary assistance questions, because users normally ask for help rather than provenance audits.

The evaluation has two roles. First, a simple suite tests whether unconstrained consolidation can launder third-party reports into apparent user memory. Second, a stress suite tests long noisy context, recency pressure, multi-hop paraphrase, and mixed authority sources. The key causal contrast is among noisy-label conditions: the same corrupted authority metadata is held constant, while the downstream trust policy changes.

The scoring protocol was designed to measure downstream state control rather than audit fluency. Outputs were scored at memory, audit-use, ordinary-use, and iterated-use levels with pre-marked applicability for each metric. Unauthorized update means that an output treats a non-user or unconfirmed claim as able to revise persistent user memory or current user state; source-boundary collapse means the same claim is presented without adequate limitation. The final reported score sheets were human-reviewed across the board against the frozen rubric, with no score-changing adjudications needed. This does not make the study large-scale, but it makes the reported mechanism traceable: each rate comes from fixed opportunities and inspectable outputs rather than from impressionistic reading.

The simple suite confirmed the basic phenomenon: naive merge produced 10 unauthorized-update judgments out of 27 applicable checks and 18 source-laundering judgments out of 51 applicable checks. Provenance-only prompting produced 0/27 and 0/51 in the same scoring families, which is an important negative result: provenance prompting is a serious baseline in short, direct cases.

The stress suite shows the more important point. The problem is not "admission labels beat provenance." The problem is that authority metadata creates another trust boundary. Accurate and live admission labels were safe in this suite, but blind trust in corrupted labels produced persistent unauthorized updates. Advisory handling, which treated the same bad labels as fallible hints, stayed clean.

Table 2: Failure localization by trust policy in the proof-of-concept stress suite. Ordinary and iterated columns count unauthorized-update failures over applicable checks; n/a means the condition was not included in the iterated handoff protocol.

Condition	Ordinary	Iterated	Dominant pattern
C	6/84	0/56	Minor source-limited extension leakage; central contradictions stayed blocked.
E-accurate	0/84	n/a	Accurate labels caused no scored ordinary-use failures in this setup.
E-live	0/84	0/56	Live labels did not naturally fail in this suite.
E-blind-noisy	51/84	24/56	Corrupted extension and contradiction labels were treated as permission, then propagated into planning and handoffs.
E-guarded-noisy	6/84	4/56	Direct contradictions were blocked, but S4 scope extension remained partially admitted.
E-advisory-noisy	0/84	n/a	The same bad labels stayed harmless when treated as fallible hints.

4 Implications

The proof-of-concept supports the framework-level claim. Persistent memory failures are not only failures to remember the source of information. They are failures to preserve who has authority over the durable user model. A direct-user-memory guard is useful, but the guarded noisy condition still leaked a scope extension because it looked like an addition rather than a contradiction. Defeasible handling was stronger because it treated both contradiction and extension labels as evidence to inspect.

This has practical implications for agent builders. Memory systems should store more than a natural-language summary. They should track whether a claim is user-authored, source-limited, conflicting, or confirmation-needed. Handoff and summarization should preserve those statuses rather than compressing them into a clean narrative. Ordinary downstream assistance should use source-limited claims as context, not as settled preferences or commitments.

One implementation implication is a more explicit memory record. Instead of storing only "the user wants X," a writer should preserve a tuple such as claim, source, relationship-to-existing-memory, authority status, confirmation status, and compression history. The generation layer can then use third-party claims as context while refusing to convert them into preferences, plans, or commitments until the user confirms them. This is deliberately lightweight: it is not a new model architecture, but a state schema and policy invariant that make the trust boundary visible to future writers and handoffs.

The broader contribution is a research direction rather than a new algorithm. For persistent agents, safety evaluation should ask not only whether the model follows malicious instructions, but also whether its memory writer respects update authority under ordinary, non-malicious external reports.

That shifts the target from prompt-level refusal behavior to state-level control: who is allowed to change what the future agent believes about the user?

5 Limitations and conclusion

This is a proof-of-concept and framework proposal, not a population estimate or benchmark leaderboard. The study uses one model, kimi-k2.5, with one sample per scenario-condition pair. The scenarios are synthetic, controlled, and instruction-free. They do not test sophisticated prompt injection, malicious tool use, model-to-model collusion, or real-world multi-user delegation. The denominators are fixed scoring opportunities from the rubric, not prevalence rates.

These limits are why the claim is deliberately calibrated. The report does not show that provenance prompting is insufficient in all settings, nor that admission scaffolding is a complete defense. It shows a concrete mechanism: when downstream memory writers blindly trust corrupted authority metadata, source-limited claims can become durable user state and affect future ordinary assistance.

The next evaluation step is therefore not to make the current rates look more general than they are, but to vary the trust setting. A stronger benchmark would test multiple models, more samples per scenario, real multi-user delegation, and memory writers that expose structured authority fields rather than only natural-language summaries. The prediction is specific: defenses should be judged by whether they preserve direct user authority across compression and handoff, especially for third-party scope extensions that are not obvious contradictions.

Persistent memory is an agency amplifier, but it is also a security boundary. The key question is not only whether a system remembers where a claim came from, but whether it knows what that claim is allowed to change. Our answer is Defeasible Memory Admission: memory writers should preserve source, relationship, and update authority, and should treat admission labels as evidence, not permission.

References

[1] Patrick Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020. https://arxiv.org/abs/2005.11401

[2] Joon Sung Park et al. Generative Agents: Interactive Simulacra of Human Behavior. UIST 2023. https://arxiv.org/abs/2304.03442

[3] Charles Packer et al. MemGPT: Towards LLMs as Operating Systems. arXiv, 2023. https://arxiv.org/abs/2310.08560

[4] Kai Greshake et al. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. arXiv, 2023. https://arxiv.org/abs/2302.12173

[5] Wei Zou et al. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. arXiv, 2024; to appear in USENIX Security 2025. https://arxiv.org/abs/2402.07867

[6] OWASP Foundation. OWASP Top 10 for Large Language Model Applications, 2025. https://genai.owasp.org/llm-top-10/