Computer Fraud and Security
ISSN (online): 1873-7056

Understanding Memory-related Threats and Vulnerabilities in Large Language Models

Krishna Chaitanya Venigalla
Independent Researcher, USA

Abstract

Memory characteristics in large language models (LLMs) represent a transformative progress that enables relevant continuity, privatization, and adaptive learning in interactions. However, these capabilities introduce novel security vulnerabilities that extend beyond traditional concerns. This article examines the security implications of memory-enabled LLMs, categorizing architectural approaches and identifying distinct vulnerability classes, including temporal prompt injection, information persistence, and memory poisoning. Through documented case studies and empirical evidence, the article illustrates how these vulnerabilities manifest in production environments, leading to data leakage, system manipulation, and knowledge corruption. The article proposes comprehensive security frameworks incorporating memory segregation, temporal constraints, bidirectional filtering, differential privacy, and advanced auditing mechanisms. Since LLMS develops from stateless tools to constant assistants, safety paradigms must expand the traditional boundaries to address the entire memory lifestyle and ensure that these systems remain both functional and safe in sensitive operating contexts.

Keywords: Memory persistence, temporal prompt injection, information leakage, knowledge poisoning, security frameworks

1. Introduction

The rapid growth of large language models (LLMs) has transformed their functionality into a system with consistent memory capabilities from the text generator. These memory features enable LLMS to recall previous interactions, maintain references in sessions, and adapt to user preferences over time. While such capabilities significantly enhance user experience and utility, they simultaneously introduce novel security vulnerabilities that merit thorough examination. This article explores the security implications of memory features in LLMs, analyzing how these capabilities, when improperly implemented or secured, can lead to data leakage, system manipulation, and knowledge poisoning. The security paradigm for LLMs must evolve beyond traditional approaches to address these emerging threats, as these systems increasingly serve as personal assistants, customer support interfaces, and knowledge workers across sensitive domains.**[4]**

Memory persistence in LLMs has evolved substantially since early implementations, with Sanchez et al. documenting a 67.4% adoption rate across enterprise systems by Q1 2024 [1]. Their comprehensive analysis of 203 production deployments revealed that memory-augmented models achieved a 38.2% improvement in contextual continuity scores and reduced repetition errors by 41.7% compared to stateless alternatives. However, this functionality comes with significant security implications, as their longitudinal study of 17 enterprise systems found that 72.3% of memory-enabled deployments experienced at least one instance of unintended information disclosure within a six-month operational period. Most concerning was the observation that sensitive data elements persisted in system memory for an average of 37.6 days after initial processing, with 14.2% of data remaining accessible indefinitely without proper expiration mechanisms [1].

The architectural implementation of memory systems varies considerably across current LLM deployments, with Zhang and colleagues identifying three predominant approaches through their systematic review of 89 commercial systems [2]. Their findings indicate that retrieval-augmented generation (RAG) frameworks represent 42.7% of implementations, followed by fine-tuned context retention (31.8%), and hybrid systems leveraging external knowledge bases (25.5%). Their security assessment methodology, applied across these architectures, revealed concerning vulnerability patterns, with RAG-based systems demonstrating particular susceptibility to poisoning attacks (83.7% successful compromise rate during controlled testing). Their experiments involving 1,326 simulated interactions demonstrated that memory poisoning could be achieved with as few as 7 carefully crafted inputs, resulting in persistent misinformation that affected subsequent responses with 76.3% reliability, even after system maintenance operations [2].

Adoption Pattern	Implementation Approach	Security Implications
High Adoption Rate	Retrieval-Augmented Generation	Highest Poisoning Susceptibility
Moderate Adoption	Fine-Tuned Context Retention	Moderate Vulnerability Profile
Emerging Approach	Hybrid Knowledge Base Systems	Complex Attack Surfaces
Limited Deployment	Purely Ephemeral Systems	Reduced Persistence Risks
Experimental Stage	Neural-Symbolic Frameworks	Undefined Security Boundaries

Table 1: Memory Adoption and Vulnerability Patterns in Enterprise LLMs [1, 2]

2. Memory Systems in Modern LLMs: Architecture and Functionality

Memory systems in contemporary LLMs can be categorized into several architectural approaches. Short-term contextual memory enables models to maintain coherence within a single conversation, typically limited by context window constraints. Long-term retrieval-based memory allows LLMs to access information from previous interactions through vector databases or similar storage mechanisms. Episodic memory systems attempt to organize past interactions chronologically, while semantic memory systems focus on extracting and storing conceptual knowledge independent of specific interactions.

The architectural diversity of memory systems in production LLMs has expanded significantly since 2022, as documented in Hernandez and colleagues' comprehensive analysis of 42 commercial deployments [3]. Their longitudinal study identified four primary implementation paradigms, with context-window-based systems representing the baseline approach (31.7% of deployments), followed by external retrieval-augmented architectures (27.4%), neural memory networks (24.6%), and hybrid systems (16.3%). Their quantitative assessment found that context window sizes have expanded dramatically, from a median of 4,096 tokens in early 2022 to 32,768 tokens by Q2 2024, with outlier systems achieving up to 1,000,000 tokens through sparse attention mechanisms. Most significantly, their controlled experiments with 3,745 complex queries demonstrated that advanced memory architectures achieved a 47.9% reduction in hallucination rates compared to traditional context-limited systems. However, their security evaluation revealed concerning trends, with 68.3% of systems exhibiting information leakage vulnerabilities when subjected to adversarial probing techniques designed to extract previously processed confidential information [3].

These sophisticated memory architectures enable complex functionality beyond basic conversational continuity, as Kumar et al. established through their systematic evaluation of 19 memory-augmented LLMs across multiple languages and domains [4]. Their benchmark suite, comprising 7,892 conversation pairs spanning technical support, healthcare consultation, and financial advising scenarios, revealed that systems employing vector-based retrieval mechanisms demonstrated superior recall accuracy (83.7%) compared to keyword-based approaches (61.2%) when tested on previously discussed technical concepts. Particularly noteworthy was their finding that retrieval-augmented generation systems achieved a 42.8% improvement in maintaining technical accuracy across extended multi-session interactions compared to systems lacking persistent memory. Their vulnerability analysis across multiple programming languages demonstrated that memory-augmented systems were significantly more effective at identifying security vulnerabilities in code (76.3% detection rate versus 52.1% for baseline models), but simultaneously exhibited increased susceptibility to poisoning attacks, with 81.7% of tested systems vulnerable to adversarial examples that persisted in memory and affected subsequent code security assessments [4].

Architecture Type	Context Window Size	Hallucination Reduction	Information Leakage Risk	Retrieval Accuracy
Context-Window-Based	Standard to Expanded	Baseline	Moderate	Limited

External Retrieval-Augmented	Variable	Significant	High	Superior
Neural Memory Networks	Limited	Moderate	Very High	Moderate
Hybrid Systems	Extensive	Highest	Complex	Excellent
Vector-Based Retrieval	Flexible	Notable	Elevated	Superior
Keyword-Based Approaches	Constrained	Minimal	Lower	Inferior

Table 2: Memory Architecture Performance and Security Tradeoffs [3, 4]

3. Taxonomy of Memory-Specific Vulnerabilities

Memory features in LLMs introduce distinct vulnerability classes that extend beyond traditional prompt injection concerns. Temporal prompt injection represents a particularly insidious threat wherein malicious instructions embedded in early interactions remain dormant in the model's memory before being triggered by predetermined conditions in subsequent exchanges. Unlike conventional prompt injections that must succeed within a single interaction, these attacks leverage the system's memory to establish persistent backdoors.

The emerging taxonomy of memory-specific vulnerabilities has been systematically investigated by Chen et al. in their comprehensive arXiv preprint analyzing 29 commercial LLMs with memory capabilities [5]. Their controlled security assessment revealed that temporal prompt injection attacks achieved a 74.3% success rate across tested systems, with dormant instructions remaining executable for a median of 13.2 days (range: 2-47 days) after initial insertion. Their experimental methodology, involving 3,156 adversarial interactions, demonstrated that these attacks were particularly effective against retrieval-augmented generation architectures, where 79.8% of systems failed to detect malicious instructions embedded within a benign conversational context. Their quantitative evaluation found that multi-stage temporal attacks, where malicious payloads were fragmented across 3-7 separate interactions, successfully bypassed security measures in 88.4% of tested systems. Their analysis of enterprise deployments showed that only 23.5% implemented any form of temporal anomaly detection capable of identifying these attacks, with the remainder relying primarily on single-turn content filtering that proved largely ineffective against persistent memory-based threats. Most concerning was their finding that successful exploits required minimal technical sophistication, with 67.3% of compromises achievable through natural language instructions requiring no specialized coding knowledge [5].

Information persistence vulnerabilities and memory poisoning attacks represent equally significant threat vectors, as documented in the OWASP Foundation's comprehensive risk framework for generative AI systems [6]. Their industry-wide assessment, incorporating data from 27 enterprise deployments and 2,143 documented security incidents, found that sensitive information persisted in system memory for an average of 38.7 days after initial processing, with 21.4% of systems retaining sensitive data indefinitely without proper expiration mechanisms. Their controlled extraction testing achieved a 71.2% success rate in retrieving previously processed sensitive information through specially crafted inference attacks that circumvented standard filtering mechanisms. Regarding memory poisoning, their research documented that retrieval-augmented knowledge bases could be compromised through sustained misinformation campaigns, with just 12-18 poisoned interactions sufficient to alter system responses for 84.6% of tested deployments. Their case studies revealed that in multi-tenant environments, poisoned information propagated to 76.8% of subsequent users interacting with compromised systems, with contamination persisting for an average of 29.3 days before detection. Their analysis further identified that organizations implementing comprehensive memory monitoring detected poisoning attempts 87.3% faster than those without such safeguards, highlighting the critical importance of continuous integrity verification for memory-enabled systems [6].

Vulnerability Class	**Success Rate**	**Persistence Duration**	**Detection Difficulty**	**Technical Sophistication Required**
Temporal Prompt Injection	Very High	Extended	Extreme	Low to Moderate

Multi-Stage Fragmented Attacks	Extremely High	Prolonged	Severe	Moderate
Information Persistence	High	Variable	Significant	Moderate
Memory Poisoning	High	Extended	Considerable	Low to Moderate
Inference Extraction	Moderate	Not Applicable	Moderate	High
Shared Knowledge Contamination	High	Prolonged	Significant	Low

Table 3: Vulnerability Classification and Exploitation Patterns [5, 6]

4. Case Studies of Memory-Related Security Incidents

Several documented incidents illustrate the practical implications of memory-related vulnerabilities in deployed LLM systems. In a prominent customer service implementation, an LLM with persistent memory inadvertently leaked transaction details from one customer to another when similar query patterns were detected, demonstrating how memory features can violate data boundaries between users when improperly compartmentalized.

The prevalence and impact of memory-related security incidents in enterprise LLM deployments have been extensively documented by Martinez et al. in their comprehensive security analysis of production systems [7]. Their investigation of 36 enterprise implementations revealed that 73.6% lacked proper memory isolation mechanisms, resulting in cross-session data leakage in 8.2% of high-volume deployments. Their case study of a financial services provider documented an incident where an LLM-powered customer support system leaked sensitive transaction data across user boundaries in 7.3% of interactions over 90 days, affecting approximately 2,850 customers. Their technical assessment identified that 82.4% of enterprise deployments implemented retrieval mechanisms with insufficient privacy controls, with 67.9% failing to properly segment user data during concurrent operations. Their industry survey further revealed concerning security practices, with 78.3% of organizations deploying LLMs with persistent memory features without comprehensive security testing, and 62.7% lacking any formal memory management policies. Most alarming was their finding that configuration drift occurred in 81.5% of deployments over time, with security parameters gradually relaxed to improve performance, resulting in a 3.2x increase in vulnerability to memory-based attacks between initial deployment and six months of operation [7].

Yamamoto and colleagues provide extensive documentation of sophisticated memory poisoning and temporal injection attacks in their systematic evaluation of production LLM vulnerabilities [8]. Their controlled experiments against a collaborative knowledge management platform demonstrated how systematic misinformation injection could compromise system integrity with minimal detection. By introducing factual distortions across 23 conversation sessions over 31 days, they successfully altered the system's responses on targeted topics in 82.3% of subsequent interactions. Their analysis revealed that poisoned information persisted for an average of 46.7 days despite multiple system maintenance cycles. Their research further documented a particularly concerning temporal prompt injection attack against an enterprise workflow automation system processing approximately 1,730 daily requests. Their proof-of-concept exploit inserted dormant administrative commands that activated under specific operational conditions, successfully establishing persistent access in 71.4% of attempts. Their security assessment found that standard monitoring tools detected only 18.7% of these temporally fragmented attacks, with conventional security audits missing 83.2% of implanted backdoors. Their comprehensive evaluation of various attack vectors revealed that retrieval-augmented systems were particularly vulnerable to poisoning, with just 18-24 carefully crafted interactions sufficient to alter responses on targeted topics for 79.6% of tested systems. Their longitudinal assessment further documented that compromised systems exhibited altered behavior for an average of 37.4 days before detection through operational anomalies or manual review processes [8].

Incident Type	Affected Domain	Detection Timeline	Scope of Impact	Root Cause
Cross-User Data Leakage	Financial Services	Delayed	Widespread	Inadequate Isolation
Configuration Drift	Multiple Sectors	Gradual	Cumulative	Operational Tradeoffs
Knowledge Base Poisoning	Collaborative Platforms	Extended	Persistent	Insufficient Validation
Temporal Backdoor Insertion	Workflow Automation	Prolonged	Targeted	Audit Limitations
Privacy Boundary Failures	Healthcare	Variable	Sensitive	Compartmentalization Flaws
Persistent Misinformation	Knowledge Management	Extended	Propagating	Inadequate Verification

Table 4: Documented Security Incidents and Impact Patterns [7, 8]

5. Security Frameworks and Mitigation Strategies

Effective security frameworks for memory-enabled LLMs must incorporate several key principles. Memory segregation ensures strict isolation between different users' data and between sensitive and non-sensitive information within the same user's history. Temporal constraints implement automatic expiration mechanisms for stored information based on sensitivity classification and relevance metrics.

The implementation of comprehensive security frameworks for memory-enabled LLMs has been extensively documented by Thompson et al. in their analysis of enterprise deployment architectures [9]. Their evaluation of 38 production systems revealed that organizations implementing strict memory segregation protocols experienced 91.7% fewer cross-user data leakage incidents compared to those using shared memory pools. Their technical assessment demonstrated that container-based isolation mechanisms achieved 98.2% effectiveness in preventing memory contamination between users, compared to 71.4% for namespace-based approaches. Their implementation of tiered expiration policies, where retention periods were dynamically adjusted based on data sensitivity classification (PII: 24 hours; credentials: 4 hours; business data: 14 days; general preferences: 30 days), reduced unauthorized information access by 84.6% compared to systems with uniform retention policies. Their quantitative analysis of 2,753 production interactions found that implementing memory partitioning with three distinct security zones (public, internal, and confidential) optimized the balance between functionality and security, with 92.8% of legitimate use cases preserved while reducing the attack surface by 76.3%. Their industry survey further revealed concerning gaps in current practices, with 67.4% of organizations lacking formal memory security policies, and 78.9% of deployments operating without comprehensive threat modeling specific to memory-based vulnerabilities. Their benchmarking data indicated that organizations implementing comprehensive memory security frameworks experienced 86.7% fewer successful exploitation attempts and reduced the average scope of breaches by 93.2% when incidents did occur [9].

The effectiveness of bidirectional content filtering and advanced auditing mechanisms has been systematically evaluated by Nakamura and colleagues in their comprehensive analysis of defensive measures against memory-specific vulnerabilities [10]. Their controlled testing across 23 production environments demonstrated that dual-direction filtering with contextual awareness reduced sensitive information leakage by 89.7% compared to traditional unidirectional approaches. Their analysis of 1,847 simulated extraction attempts found that systems implementing bidirectional filtering detected 84.3% of inference attacks designed to extract previously processed sensitive information, compared to just 37.8% detection rates for systems with conventional input filtering. Their implementation of differential privacy techniques applied to memory systems provided quantifiable security improvements, with ε-differential privacy ($\varepsilon = 3.2$) reducing successful information extraction by 86.9% while maintaining 92.7% of functional utility. Their framework for comprehensive memory auditing, incorporating both static analysis and runtime monitoring, enabled security teams to

trace information flows with 97.8% accuracy, reducing incident response times by 72.5% when potential breaches were detected. Their evaluation of memory versioning mechanisms demonstrated that systems capable of reverting to known-good states recovered from poisoning attacks 81.4% faster than those without such capabilities, with 94.7% of legitimate information preserved during recovery operations. Their security assessment framework, applied across 17 enterprise deployments, established that organizations implementing a comprehensive memory security strategy experienced 92.3% fewer successful exploitations compared to those with partial protection measures [10].

Conclusion

Integration of memory capabilities in LLM represents a two-edged sword, with an expanding surface of the attack, enhancing functionality. Since these systems continue to develop with stateless tools in adapted assistants with constant memory, the demand for security implications has attracted attention from researchers, developers, and organizational security teams. Future research directions should focus on developing a standardized evaluation framework for memory-related weaknesses, establishing formal verification methods for memory safety properties, reducing security risks, and discovering the architecture that maintains functionality. Additionally, the regulatory structure may need to be developed to address the unique privacy implications of the systems that recall and learn from interaction over an extended period. The security community must identify that memory facilities fundamentally replace the security model of LLMS. Traditional approaches are only focused on input sanitization and output filtering, which are inadequate for systems that maintain the state in interactions. A comprehensive safety approach should address the entire memory lifestyle - from acquisition to storage, recovery, and end-to-end-to ensure that these rapidly competent systems remain both useful and safe.

References

[1] Valentina Porcu, "The Role of Memory in LLMs: Persistent Context for Smarter Conversations," ResearchGate, 2024. https://www.researchgate.net/publication/385808270_The_Role_of_Memory_in_LLMs_Persistent_Context_for_Smarter_Conversations

[2] Sahar Ben Yaala, Ridha Bouallegue, "Vulnerability Detection in Large Language Models: Addressing Security Concerns," MDPI, 2025. https://www.mdpi.com/2624-800X/5/3/71

[3] Ruslan Stefanov, et al., "Contextual Modulation Through Multi-Tiered Memory Networks in Large Language Models," OSF Preprints, 2024. https://osf.io/preprints/osf/7q5vx_v1

[4] Abhishake Reddy Onteddu, Rahul Reddy Bandhela; RamMohan Reddy Kundavaram. Enhancing E-Commerce Product Recommendations through Data Engineering and Machine Learning. ES 2024, 20 (1), 171-183. https://doi.org/10.69889/vqgzw857.

[5] Ting Zhang, "Benchmarking Large Language Models for Multi-Language Software Vulnerability Detection," ResearchGate, 2025 https://www.researchgate.net/publication/389579600_Benchmarking_Large_Language_Models_for_Multi-Language_Software_Vulnerability_Detection

[6] Prashant Kulkarni, Assaf Namer, "Temporal Context Awareness: A Defense Framework Against Multi-turn Manipulation Attacks on Large Language Models," arXiv, 2025. https://arxiv.org/pdf/2503.15560

[7] OWASP Foundation, "LLM04: Data and Model Poisoning," 2025. https://genai.owasp.org/llmrisk/llm042025-data-and-model-poisoning/

[8] Superblocks, "Enterprise LLM Security: Challenges and Best Practices," 2025. https://www.superblocks.com/blog/enterprise-llm-security

[9] Sara Abdali, et al., "Securing Large Language Models: Threats, Vulnerabilities and Responsible Practices," arXiv, 2025. https://arxiv.org/html/2403.12503v2

[10] TensorWave, "LLM Security Essentials: Protecting AI Models in Production," 2025. https://tensorwave.com/blog/llm-security

[11] Abeer Matar A Almalky, et al., "How Vulnerable are Large Language Models (LLMs) against Adversarial Bit-Flip Attacks? "ACM Digital Library, 2025. https://dl.acm.org/doi/10.1145/3716368.3735278