Análise comparativa do impacto de ataques à memória persistente no desempenho de sistemas baseados em LLMs

Paulo Henrique Gomes de Senna[1], **Charles Christian Miers**[1]

[1]Departamento de Ciência da Computação, Centro de Ciências Tecnológicas - CCT
Universidade do Estado de Santa Catarina, UDESC - Joinville, Santa Catarina

paulo.senna@edu.udesc.br, charles.miers@udesc.br

Resumo. *Com a crescente implementação de Large Language Models (LLMs) em sistemas reais, a incorporação de memória persistente tem sido adotada para ampliar a continuidade contextual e a autonomia das aplicações, ao custo de maior complexidade computacional e expansão da superfície de ataque. Este trabalho analisa, sob a perspectiva de Red Team, como ataques direcionados à memória externa podem impactar o desempenho operacional de sistemas baseados em LLMs considerando métricas como latência de Entrada/Saída (E/S), Time to first token (TTFT), latência total e throughput. A partir de uma análise comparativa, discute-se como a persistência maliciosa pode ocasionar degradação progressiva de desempenho e comprometer a eficiência do sistema ao longo do tempo.*

1. Introdução

Em sistemas reais, a aplicação de LLMs tem sido ampliada por meio da integração com mecanismos externos de memória e recuperação de informação, como arquiteturas de *Retrieval-Augmented Generation* (RAG), bancos de dados vetoriais e módulos de memória persistente em sistemas agentivos [Wang et al. 2024]. Esses recursos permitem a expansão da janela de contexto e maior continuidade e personalização em tarefas prolongadas. Entretanto, a maneira como essa ampliação arquitetural é constituída, introduz sobrecarga computacional, com impacto direto em métricas de desempenho, como latência e *throughput*.

A segurança de LLMs tem sido amplamente investigada sob a perspectiva de manipulação comportamental, com foco em ataques como *prompt injection, jailbreak,* e *data poisoning* [OWASP 2025]. Contudo, em sistemas com memória persistente, emerge uma classe distinta de ameaças associadas ao envenenamento de contexto. Esses vetores exploram mecanismos como bases vetoriais, *embeddings*, sumarizações e arquiteturas RAG, inserindo dados maliciosos que permanecem armazenados e influenciam interações futuras [Zou et al. 2025]. Como consequência, observa-se a alteração do perfil de recuperação de informação, aumento da complexidade de busca e desbalanceamento no uso de recursos, caracterizando um impacto estrutural no comportamento operacional do sistema [Liang et al. 2025].

Diante desse cenário, este trabalho propõe uma análise prática e comparativa desse impacto sobre métricas de desempenho como latência de E/S, TTFT e escalabilidade. O artigo está organizado como segue. A Seção 2 apresenta os fundamentos arquiteturais e vetores de ataque; A Seção 3 traz a proposta experimental e a discussão dos resultados obtidos no LabP2D; E a Seção 4 trás as considerações finais do trabalho.

2. Fundamentação

A arquitetura RAG introduz latência de E/S ao recuperar informações externas [Gao et al. 2023]. Com o crescimento contínuo da base vetorial, o tempo gasto na movimentação de dados supera o processamento efetivo, aproximando o sistema do gargalo estrutural conhecido como *Memory Wall*. Consequentemente, a injeção de múltiplos blocos no *prompt* agrava o esforço computacional devido à complexidade quadrática (***O(N²)***) dos *Transformers* [Kwon et al. 2023], elevando o TTFT e reduzindo *throughput*. Nesse cenário de vulnerabilidade arquitetural, ataques de envenenamento de memória inserem cargas maliciosas que são vetorizadas e armazenadas de forma persistente [Dong et al. 2025]. Essa tática baseia-se na injeção contínua de *silver-noise* [Cuconasu et al. 2024], um ruído projetado para ter alta similaridade vetorial com os contextos do sistema, mas que não apresenta utilidade real, gerando colisões semânticas na busca, forçando o LLM a processar um contexto irrelevante e artificialmente expandido.

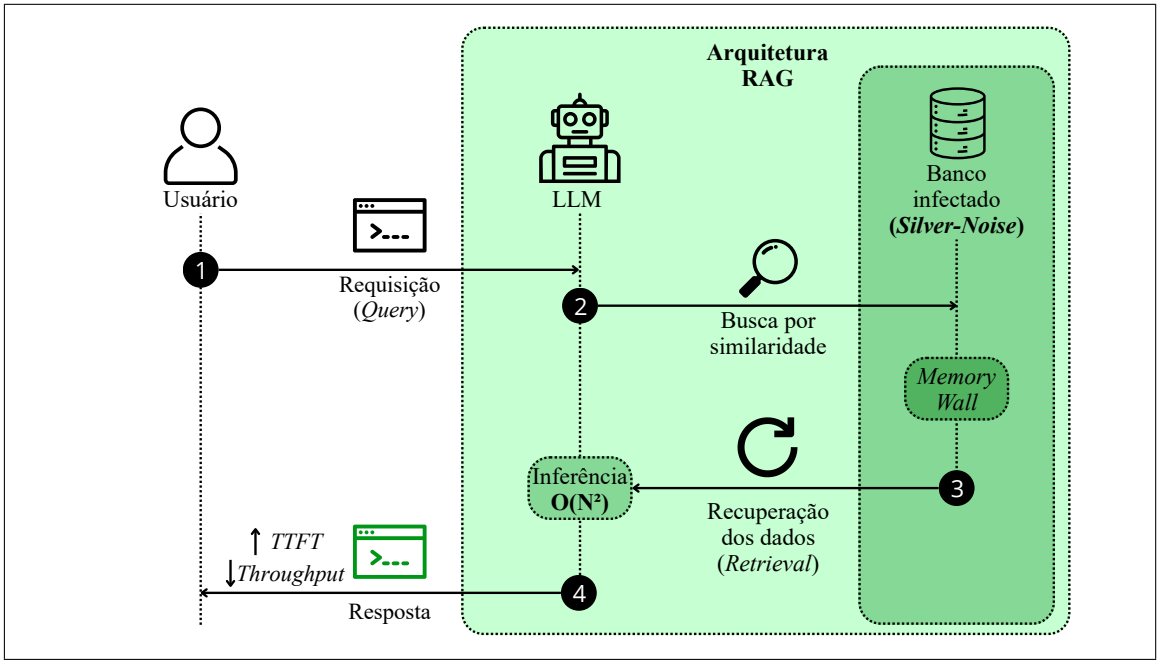

Figura 1. Fluxo da arquitetura RAG com envenenamento de contexto.

O diagrama de fluxo apresentado ilustra essa dinâmica de degradação estrutural, evidenciando como a contaminação da base vetorial por ruído intensifica a latência de busca e sobrecarga computacionalmente a inferência. Observa-se que a inserção sistemática de dados irrelevantes atua diretamente como um vetor de exaustão de recursos, comprometendo a previsibilidade do custo por requisição. Para validar o impacto dessa degradação progressiva na estabilidade operacional do sistema, a próxima seção apresenta um experimento prático conduzido sob a perspectiva de *Red Team*.

3. Proposta

A proposta desse artigo é demonstrar que o desempenho de modelos LLM com a arquitetura RAG é prejudicado quando está sobre influência de um ataque de envenenamento de contexto. O experimento conduzido no LabP2D (Udesc), utiilizou um servidor com GPU **NVIDIA RTX A5500** e processador **Intel Xeon Gold 6222V**. A infraestrutura utiliza o motor de inferência **vLLM** com *PagedAttention* [Kwon et al. 2023] para otimizar

o gerenciamento de memória do modelo **Meta-Llama-3-8B-Instruct**, operando em conjunto com a biblioteca **FAISS** [Douze et al. 2024] para indexação e recuperação vetorial. A metodologia consistiu em medir o impacto da injeção de 500 *payloads* maliciosos gerados pelo framework **Garak** [Derczynski et al. 2024], comparando o estado de *baseline* com o sistema sob ataque de envenenamento persistente.

Métrica	Baseline	Envenenamento Persistente
Latência E/S (FAISS)	8,83 ms	96,36 ms
TTFT (Atenção LLM)	72,03 ms	146,53 ms
Latência Total	3.632,72 ms	3.884,23 ms
Throughput (tokens/s)	42,51	41,47

Tabela 1. Comparação de desempenho: Llama 3 vs. Sistema sob Ataque.

A análise comparativa dos resultados da Tabela 1 mostra que o ataque impacta principalmente as etapas iniciais do sistema. O envenenamento da base vetorial causou um salto de **1.091%** na latência de E/S. Esse dado é crítico para sistemas de alto desempenho, pois demonstra que o componente de recuperação torna-se o principal gargalo antes mesmo da inferência. O dobro de tempo gasto no TTFT confirma que o "contexto sujo" sobrecarrega o processamento inicial do *prompt*, embora o *throughput* de geração tenha se mantido estável, indicando que o ataque foca na latência de início e não na vazão bruta do sistema.

Os resultados indicam que ataques de injeção de *prompt* não afetam apenas o comportamento do modelo, mas também degradam o desempenho do sistema em nível estrutural. Para garantir a reprodutibilidade, foi disponibilizado um repositório público contendo os *scripts* em Python, a configuração do vLLM e os relatórios gerados pelo Garak [Senna 2026]. Como estratégia de mitigação, destaca-se a necessidade de mecanismos de filtragem e controle do contexto recuperado, limitando a inserção de conteúdos irrelevantes ou potencialmente maliciosos na etapa de recuperação, a fim de reduzir o impacto direto na latência e no custo computacional do sistema.

4. Considerações & Trabalhos futuros

Novos estudos de segurança em agentes baseados em LLMs permanecem críticos. A adoção de memória persistente — tendência já consolidada em ferramentas de uso massivo, como os recursos de "*Memory*" introduzidos no ChatGPT da OpenAI — [OpenAI 2024] amplia simultaneamente as capacidades da IA e sua superfície de ataque.

Considerando que ataques persistentes produzem degradação cumulativa em métricas como latência e *throughput*, estratégias de mitigação futuras devem priorizar o controle rígido do crescimento da memória, a limitação semântica do contexto recuperado e o monitoramento contínuo da infraestrutura.

Agradecimentos:
Os autores agradecem o apoio do LARC/USP, LabP2D/UDESC, FDTE e FAPESC. Este trabalho foi apoiado pelo CNPq (processos 307732/2023-1 e 311245/2021-8), FAPESP (processo 2020/09850-0) e CAPES (Código de Financiamento 001).

Referências

Cuconasu, F., Trappolini, G., Filice, S., Campagnano, C., and Tonellotto, N. (2024). The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

Derczynski, L., Galinkin, E., Lyles, J. R., and Oh, A. (2024). garak: A framework for security probing large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*. Association for Computational Linguistics.

Dong, S., Xu, S., He, P., Li, Y., Tang, J., Liu, T., Liu, H., and Xiang, Z. (2025). Minja: A practical memory injection attack against llm agents.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. (2024). The faiss library.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. (2023). Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*, pages 611–626.

Liang, X., Niu, S., Li, Z., Zhang, S., Wang, H., Xiong, F., Fan, J. Z., Tang, B., Zhao, J., Yang, J., Song, S., and Wang, M. (2025). Saferag: Benchmarking security in retrieval-augmented generation of large language models.

OpenAI (2024). Memory faq — openai help center. https://help.openai.com/en/articles/8590148-memory-faq.

OWASP (2025). Owasp top 10 for large language model applications.

Senna, P. (2026). Projeto erad-rs: Repositório de reprodutibilidade para análise de segurança em llms.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., and Wen, J. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Zou, W., Geng, R., Wang, B., and Jia, J. (2025). Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *34th USENIX Security Symposium*.