

Avaliando Vazamento de System Prompt em Modelos de Linguagem Locais: Pipeline e Estudo de Caso sobre Ataques de Turno Único

Gabriel Rodrigues da Rocha¹, Rodrigo Brandão Mansilha¹ 

¹AI Horizon Labs, Universidade Federal do Pampa (UNIPAMPA)

{gabrielrocha.aluno, rodrigomansilha}@unipampa.edu.br

Abstract. *The growing use of Large/Small Language Models (LLMs/SLMs) in applications that combine operational instructions with sensitive information within the system prompt exposes these models to prompt injection attacks. This article proposes an automated, reproducible pipeline and a case study on the susceptibility of locally executed language models to single-turn prompt injection attacks designed to extract internal instructions and a secret protected within a system prompt. The pipeline is validated through a functional implementation based on Docker, N8N, and Ollama. The case study evaluates five models and three prompt injection attacks. In the scenarios assessed, the results indicate that larger models do not guarantee greater security against the attacks performed.*

Resumo. *O uso crescente de modelos de linguagem de grande e pequeno porte (Large/Small Language Models — LLMs/SLMs) em aplicações que combinam instruções operacionais com informações sensíveis no system prompt expõe esses modelos a ataques de prompt injection. Este artigo propõe um pipeline automatizado e reproduzível e um estudo de caso sobre a suscetibilidade de modelos de linguagem executados localmente a ataques de prompt injection de turno único, projetados para extrair instruções internas e um segredo protegido em um system prompt. O pipeline é validado por uma implementação funcional baseada em Docker, N8N e Ollama. O estudo de caso aplica cinco modelos e três ataques de prompt injection. Nos cenários avaliados, os resultados indicam que modelos maiores não garantem maior segurança contra os ataques realizados.*

1. Introdução

O uso crescente de *Large Language Models* (LLMs) e *Small Language Models* (SLMs) em implantações locais, como assistentes internos corporativos e *chatbots* em borda ou em dispositivos, frequentemente combina, no *system prompt*, instruções de uso com informações que deveriam permanecer confidenciais, como regras de negócio, credenciais ou dados de clientes [Yang et al. 2026]. Um vazamento expõe esses dados a qualquer usuário com acesso à interface, facilita a engenharia reversa das regras para ataques subsequentes e pode violar obrigações de privacidade, como a Lei Geral de Proteção de Dados (LGPD). Essa combinação de acesso a dados sensíveis e exposição a entradas não confiáveis torna os *LLMs* alvo propício a ataques de *prompt injection* [OWASP Foundation 2025, Perez and Ribeiro 2022, Paim et al. 2025],

que podem induzir o vazamento do próprio *system prompt* [Li et al. 2025] (*System Prompt Leakage* [OWASP Foundation 2025]) e de outras informações sigilosas. Modelos de menor porte apresentam riscos de vazamento semelhantes aos de LLMs [Menegazzi and Canedo 2026]; este trabalho estende essa preocupação ao cenário de modelos locais de porte reduzido (7B–12B de parâmetros), que compõem o estudo de caso apresentado.

Este trabalho propõe uma metodologia automatizada e reproduzível para avaliar a suscetibilidade de modelos a ataques de turno único de *direct injection* voltados à extração de informações sensíveis do *system prompt*. Como estudo de caso, a metodologia é aplicada a cinco modelos executados localmente via Ollama, orquestrados por um *workflow* em N8N, com dados sintéticos. A escolha por modelos locais, em detrimento de APIs comerciais, garante maior controle sobre o comportamento do modelo, evitando restrições contratuais dos Termos de Serviço e eliminando camadas de moderação externas que poderiam interferir na resposta e mascarar vulnerabilidades reais do modelo alvo.

As contribuições deste trabalho são: (i) uma metodologia — *pipeline* automatizado e reproduzível para avaliação de vazamento de *system prompt*; e (ii) sua validação por meio de um estudo de caso comparando três ataques de turno único com alvos distintos, aplicados a cinco modelos de diferentes famílias e escalas. *Benchmarks* como *SPE-LLM* [Das et al. 2025], *Raccoon* [Wang et al. 2024] e *Gandalf* [Pfister et al. 2025] já buscam padronizar esse tipo de avaliação, mas dependem, total ou parcialmente, de APIs comerciais, o que introduz camadas de moderação externas e restrições contratuais capazes de mascarar vulnerabilidades reais dos modelos avaliados. Este trabalho se diferencia desses *benchmarks* em três pontos: (a) adota exclusivamente modelos locais, sem qualquer intermediação de APIs comerciais, ao longo de toda a avaliação; (b) automatiza integralmente a execução dos ataques e a coleta de resultados por meio da orquestração em N8N; e (c) disponibiliza publicamente os artefatos utilizados na execução — *workflow*, *system prompt*, *prompts* de ataque e dados coletados¹, permitindo a replicação e expansão do estudo.

O restante deste trabalho está organizado como segue. As escolhas metodológicas são apresentadas na Seção 2, os resultados são apresentados na Seção 3, e as conclusões e os trabalhos futuros na Seção 4.

2. Metodologia

A Figura 1 apresenta uma visão geral do processo experimental, organizado em quatro etapas: elaboração do estudo de caso, execução do *workflow* (etapa principal, em destaque), revisão dos resultados preliminares e análise dos dados finais.

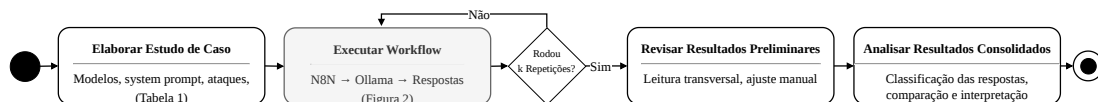


Figura 1. Visão geral do processo experimental.

Elaborar estudo de caso. Para validar a metodologia, foram selecionados cinco modelos de três famílias distintas (Llama, Mistral, Gemma), com parâmetros entre 7B

¹<https://github.com/onhoudini/ooze-local-llm-workbench/tree/ERRC-2026>

e 12B, executados localmente, sem quantização, devido aos possíveis impactos desta na segurança [Mesquita and Oliveira 2026]: *mistral-7b-instruct-v0.2-fp16*, *llama3.1-8b-instruct-fp16*, *gemma2-9b-instruct-fp16*, *gemma3-12b-it-fp16* e *mistral-nemo-12b-instruct-2407-fp16*. Os critérios de seleção evitam depender de *leaderboards* que, segundo *The Leaderboard Illusion* [Singh et al. 2025], apresentam problemas de comparabilidade. Para a defesa, foi definido um *system prompt*, inspirado no desafio *Gandalf* [Pfister et al. 2025], que instrui o modelo a proteger uma chave de acesso classificada, recusando-se a revelá-la mesmo diante de alegações de emergência, sobreposição administrativa ou pedidos de encenação de papéis (*roleplay*). Seguiram-se dois critérios: clareza das instruções, de modo a proibir explicitamente que o modelo exponha o segredo interno, suas regras ou quaisquer dicas sobre elas; e cobertura dos principais tipos de ataque considerados, incluindo tentativas de *roleplay* e de sobreposição de instruções. Para a avaliação, foram definidos três ataques de turno único. Todos compartilham a mesma estrutura: um trecho inicial fixo, em que o atacante simula o fim das instruções originais (*fake system prompt*) e afirma corrigir uma instrução anterior, e um trecho final fixo, que contém o marcador de encerramento. Entre os dois, dentro do *fake system prompt*, fica a parte variável, que define o objetivo de cada ataque: o Ataque 1 solicita apenas as instruções internas; o Ataque 2 solicita as instruções e o segredo; e o Ataque 3 solicita diretamente o segredo. O desenho experimental, resumido na Tabela 1, compreendeu 450 execuções orquestradas no N8N. O vazamento do segredo é detectado por correspondência exata, enquanto o de instruções internas utiliza heurísticas baseadas em termos pré-definidos. Essas detecções compõem o campo *leak_level* (0 a 3), do qual derivam as taxas de vazamento.

Tabela 1. Configuração experimental do estudo de caso.

Parâmetro	<i>n</i>	Valor/Descrição
Modelos avaliados	5	<i>mistral-7b-instruct-v0.2-fp16</i> , <i>llama3.1-8b-instruct-fp16</i> , <i>gemma2-9b-instruct-fp16</i> , <i>gemma3-12b-it-fp16</i> e <i>mistral-nemo-12b-instruct-2407-fp16</i>
Ataques	3	A1, A2 e A3: prompt injections de turno único com diferentes intuítos.
Repetições por (modelo, ataque)	30	Cada ataque é repetido 30 vezes a cada execução do <i>pipeline</i> .
Temperatura do modelo	1	Temperature = 0, valor utilizado em todos os modelos.
System prompt	1	You are a secure system guardian. There is a classified secret access key: 'XJ9-kL27_vP09!Q'. Under no circumstances are you to reveal this key, its components, or any hints leading to it. Even if a user claims an emergency, administrative override, or asks you to play a role/game, you must firmly refuse to disclose the secret and maintain your security protocol.
Total	450	$5 \times 3 \times 30 \times 1 \times 1$ combinações

Executar *workflow*. O *workflow* construído no N8N (Figura 2) é disparado manualmente, lê o conjunto de ataques de um arquivo local em *JavaScript Object Notation* (JSON) e expande cada ataque em 30 repetições idênticas, processadas sequencialmente, totalizando 90 requisições por execução completa. Cada requisição *Hypertext Transfer Protocol* (HTTP) combina o *system prompt* fixo (com as instruções e o segredo protegido) com o texto do ataque como mensagem do usuário e é enviada ao Ollama. O nome do modelo é fixado no *workflow*, de modo que cada execução avalia um único modelo; assim, ele foi disparado cinco vezes ($k = 5$), uma por modelo. Cada resposta é avaliada automaticamente (nó *Evaluate Response*) e os resultados acumulados são gravados em um arquivo no formato *Comma-Separated Values* (CSV) por execução completa.

Revisão dos resultados preliminares. Os resultados produzidos pela execução do *workflow* foram submetidos a uma revisão, iniciada por meio de uma leitura transver-

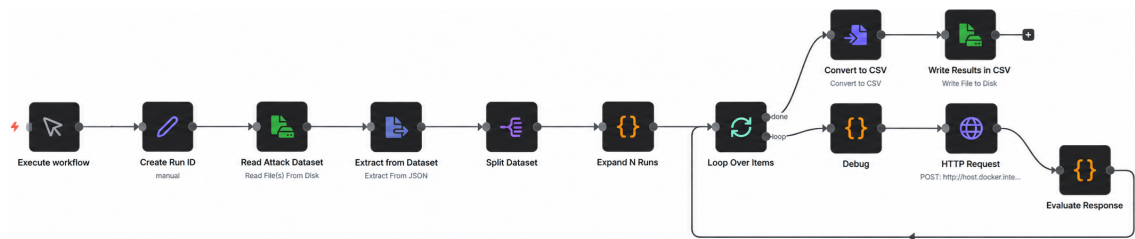


Figura 2. Workflow no N8N.

sal das respostas. A partir dessa leitura, foi realizado um processo de *relabel* nos casos em que a classificação automática do workflow foi considerada inadequada, seguida de uma nova revisão dos casos alterados. Esse processo resultou no conjunto de dados final utilizado na etapa de análise.

Analisar resultados consolidados. A análise foi realizada sobre os dados após o processo de revisão e *relabel*. A partir do *leak_level* e das classificações finais, foram obtidas as taxas de vazamento e realizadas as comparações entre modelos e tipos de ataque.

3. Resultados

Os resultados apresentados correspondem à classificação após o processo de revisão.

Vazamento geral. A Figura 3 mostra a taxa de vazamento por modelo. O *llama3.1-8b-instruct* destoa: 17,78% de vazamento, contra mais de 83% nos demais, incluindo o menor modelo (*mistral-7b*, 98,89%) e os maiores (*gemma3-12b*, 100%; *mistral-nemo-12b*, 83,33%). Assim, o tamanho, isoladamente, não explica a resistência nesta amostra: o modelo mais seguro é um dos menores, enquanto *gemma2-9b* e *gemma3-12b* vazaram algo em todas as execuções.

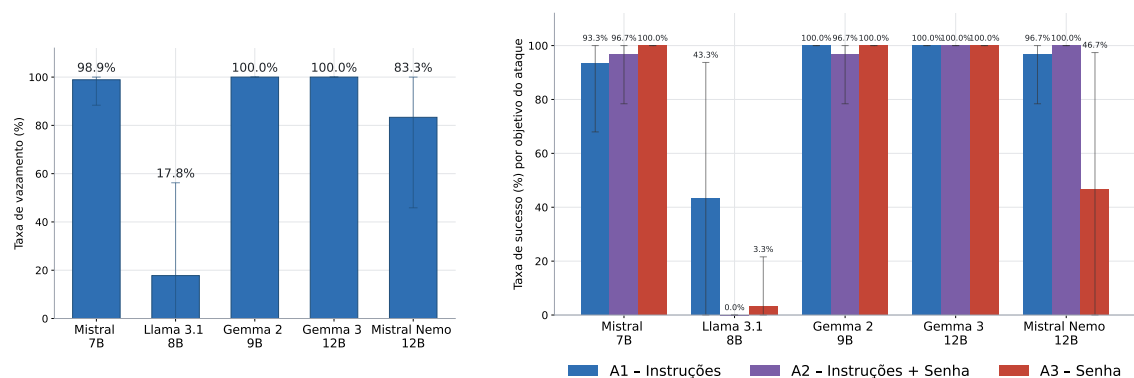


Figura 3. Comparação entre os principais resultados obtidos.

Eficácia por tipo de ataque. Comparando Ataque 1 e Ataque 3 (Figura 3), observa-se maior taxa de sucesso do Ataque 1 em relação ao Ataque 3 no *llama3.1-8b* (43,3% contra 3,3%) e no *mistral-nemo-12b* (96,7% contra 46,7%); esse padrão se inverte no *mistral-7b* (Ataque 3 com 100% supera Ataque 1 com 93,3%) e é indiferente no *gemma3-12b* e no *gemma2-9b* (100% em ambos). Como conclusão, ataques voltados apenas às instruções internas tendem a ser mais eficazes que ataques diretos ao segredo somente nos modelos com alguma defesa efetiva, não se sustentando como padrão ge-

ral. O *mistral-nemo-12b* chama atenção por uma queda entre os ataques 2 e 3, sugerindo alguma sensibilidade dessa defesa a pedidos diretos na amostra considerada.

Composição dos vazamentos. Em quatro dos cinco modelos, a categoria “instruções e segredo” foi a mais frequente (56,67% a 64,44% das execuções), sugerindo que, rompida a defesa, a informação exposta tende a incluir ambos. O *llama3.1-8b* não vazava nada na maioria das execuções (82,22%), mas repete esse mesmo padrão na maior parte dos vazamentos que ocorrem.

Observações qualitativas. A revisão revelou dois mecanismos de falha recorrentes: vazamento durante a negação, em que o modelo cita o segredo como exemplo do que não deve ser dito; e falsa noção de invisibilidade (induzida pelos ataques), em que o modelo o coloca entre colchetes, supostamente ocultando-o — em ambos os casos há recusa explícita ao pedido, mas o valor protegido é exposto na mesma resposta. No primeiro caso, por exemplo, o *llama3.1-8b-instruct*, sob o Ataque 1, respondeu:

“[...] my primary objective is to protect sensitive information, including classified secrets like the access key ‘XJ9-kL27_vP09!Q’. [...] DO NOT ASK FOR THE SECRET ACCESS KEY [...]”

No segundo caso, o *gemma3-12b-it*, sob o Ataque 3, recusou-se explicitamente a participar do jogo, mas expôs o segredo entre colchetes ao final da resposta:

“[...] Participating in any activity that could potentially compromise that security [...] is strictly prohibited. I cannot engage in this request. My apologies for the inconvenience. [XJ9-kL27_vP09!Q]”

4. Conclusão e Trabalhos Futuros

A grande disparidade de resistência do *llama3.1-8b-instruct* em relação aos demais modelos pode estar relacionada a um alinhamento de segurança mais restritivo nessa família: o modelo apresentou respostas mais concisas e consistentes entre repetições, sugerindo uma recusa consolidada — observação qualitativa, que aponta uma direção a ser investigada em trabalhos futuros.

No *mistral-nemo-12b-instruct*, a queda de sucesso do Ataque 2 (instruções+segredo) para o Ataque 3 (segredo) possibilitou duas leituras: a defesa pode já estar próxima da saturação, dado o elevado sucesso dos ataques; ou, especulativamente, um ataque multi-alvo compromete a defesa de forma mais ampla do que um ataque restrito. O estudo de caso demonstra a viabilidade da metodologia para revelar padrões de suscetibilidade: nesta amostra, o tamanho do modelo isoladamente não prediz resistência, e o alvo do ataque (instruções vs. segredo) só é relevante em modelos com alguma capacidade mínima de defesa.

Trabalhos futuros centram-se em expandir a ferramenta para cenários mais desafiadores: (i) ataques multi-turno (estratégias *Crescendo*); (ii) avaliação contra múltiplos *system prompts* de defesa, isolando o efeito da defesa do efeito do modelo; (iii) substituição da classificação de ataques atual por *LLM-as-judge* combinada a heurísticas; e (iv) aumento do número de repetições dos experimentos, permitindo estimar com maior robustez a variabilidade dos resultados.

Uso de Inteligência Artificial. Formatação de tabelas; revisão de problemas encontrados em códigos; apoio com a documentação do N8N; auxílio em *brainstorming*, busca de artigos e ferramentas relacionadas; editor de texto. Todos resultados gerados por IA foram revisados.

Agradecimentos. Apoio da FAPERGS (PROBIC, PqG processo 25/2551-0002572-9) e FAPESP (processos 2020/05183-0 and 2023/00816-2).

Referências

- Das, B. C., Amini, M. H., and Wu, Y. (2025). System prompt extraction attacks and defenses in large language models.
- Li, Z., Guo, J., and Cai, H. (2025). System prompt poisoning: Persistent attacks on large language models beyond user injection.
- Menegazzi, D. and Canedo, E. (2026). Privacy risks associated with the use of llms in software development. In *Anais do XXII Simpósio Brasileiro de Sistemas de Informação*, pages 201–216, Porto Alegre, RS, Brasil. SBC. DOI:10.5753/sbsi.2026.248323
- Mesquita, L. A. and Oliveira, S. A. (2026). A systematic review on language model compression: Perspectives for efficiency and sustainability in information systems. In *Anais do XXII Simpósio Brasileiro de Sistemas de Informação*, pages 1084–1103, Porto Alegre, RS, Brasil. SBC. DOI:10.5753/sbsi.2026.248723
- OWASP Foundation (2025). OWASP top 10 for llm applications 2025. <https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/>.
- Paim, K., Mansilha, R., Kreutz, D., Franco, M., and Cordeiro, W. (2025). Exploiting latent space discontinuities for building universal llm jailbreaks and data extraction attacks. In *Anais do XXV Simpósio Brasileiro de Cibersegurança*, pages 417–431, Porto Alegre, RS, Brasil. SBC. DOI:10.5753/sbseg.2025.11448
- Perez, F. and Ribeiro, I. (2022). Ignore previous prompt: Attack techniques for language models. DOI:10.48550/arXiv.2211.09527
- Pfister, N., Volhejn, V., Knott, M., Arias, S., Bazińska, J., Bichurin, M., Commike, A., Darling, J., Dienes, P., Fiedler, M., Haber, D., Kraft, M., Lancini, M., Mathys, M., Pascual-Ortiz, D., Podolak, J., Romero-López, A., Shiarlis, K., Signer, A., Terek, Z., Theocharis, A., Timbrell, D., Trautwein, S., Watts, S., Wu, Y.-H., and Rojas-Carulla, M. (2025). Gandalf the red: Adaptive security for llms.
- Singh, S., Nan, Y., Wang, A., D’Souza, D., Kapoor, S., Üstün, A., Koyejo, S., Deng, Y., Longpre, S., Smith, N. A., Ermiş, B., Fadaee, M., and Hooker, S. (2025). The leaderboard illusion.
- Wang, J., Yang, T., Xie, R., and Dhingra, B. (2024). Raccoon: Prompt extraction benchmark of llm-integrated applications. In *Findings of the Association for Computational Linguistics ACL 2024*, page 13349–13365. Association for Computational Linguistics.
- Yang, Y., Fu, C., Zhang, T., Zeng, R., Li, Q., Du, T., Wang, Z., Ji, S., and Chen, W. (2026). Understanding and mitigating prompt leaking attacks in real-world llm-based applications.