

Da Execução Autônoma ao Risco Estrutural: Desafios de Segurança na Era da Agentic AI

Leandro Costa Boscariol¹, Letícia Siguinolfi de Lima¹

¹Faculdade Donaduzzi (Biopark Educação) – Toledo – PR – Brasil

leandro.costa.boscariol@gmail.com, leticia.lima@bpkedu.com.br

Abstract. *The current AI landscape is undergoing a critical transition toward Agentic AI: unlike traditional models, autonomous agents such as OpenClaw execute real operations on operating systems without human intermediation. Through an analytical review and threat modeling based on OWASP, NIST, and Microsoft frameworks, this study investigates the structural risks inherent to AI agents with execution power, systematizing a layered defense model encompassing Zero Trust, sandboxing, and least-privilege policies. Findings indicate that the absence of Zero Trust controls and unfiltered OS integrations are the most structurally significant risks. Results are limited by the analytical nature of the study and the absence of red teaming validation.*

Keywords: Agentic AI, LLM Security, Zero Trust, OpenClaw, Prompt Injection.

Resumo. *O cenário da Inteligência Artificial atravessa uma transição crítica com o surgimento da Agentic AI: ao contrário de modelos tradicionais, agentes autônomos como o OpenClaw executam operações reais em sistemas operacionais sem intermediação humana. Por meio de revisão analítica e modelagem de ameaças (threat modeling) com base nos frameworks OWASP, NIST e Microsoft, este estudo investiga os riscos estruturais inerentes à adoção de agentes de IA com poder de execução, sistematizando um modelo de defesa em camadas que abrange Zero Trust, sandboxing e privilégio mínimo. Os resultados indicam que a ausência de Zero Trust e a integração não filtrada com o sistema operacional são os riscos mais significativos. Os resultados são limitados pela natureza analítica do estudo e pela ausência de validação por red teaming.*

Palavras-chave: Agentic AI, Segurança de LLM, Zero Trust, OpenClaw, Prompt Injection.

1. Introdução

Agentes autônomos como o OpenClaw possuem capacidade de execução real em sistemas operacionais — manipulando arquivos, acionando *scripts* e interagindo com APIs externas — diferentemente dos *chatbots* passivos que apenas processam texto [1]. Essa capacidade amplia significativamente a superfície de ataque, tornando inadequados os modelos de segurança concebidos para aplicações conversacionais. Conforme Myrbakken e Colomo-Palacios [2], a incorporação tardia de controles de segurança eleva exponencialmente os custos de remediação. Diante disso, este trabalho formula a seguinte questão investigativa: *quais são os riscos estruturais inerentes à adoção de agentes de IA com poder de execução e como mitigá-los eficazmente?* A hipótese central é que a ausência de

Zero Trust e a integração não filtrada com o sistema operacional representam as barreiras mais significativas, sendo o *sandboxing* e os *proxies* de inspeção os mecanismos que mais contribuem para a redução do *blast radius* [4].

2. Referencial Teórico

2.1. Prompt Injection

O *Prompt Injection* é classificado como o risco de maior criticidade no OWASP Top 10 for LLM Applications [1]. Diferentemente do *SQL Injection*, que explora falhas de sintaxe, o *Prompt Injection* explora a incapacidade do modelo em distinguir, semanticamente, entre instruções legítimas e dados maliciosos — fronteira fundamentalmente difusa em LLMs [3]. Greshake et al. [3] distinguem a injeção direta — instruções maliciosas inseridas pelo próprio usuário no *prompt* — da injeção indireta, mais perigosa: o agente processa um documento externo com instruções ocultas, de modo que dado legítimo e comando adversarial chegam pelo mesmo canal. Caso o agente possua permissões suficientes, o ataque resulta em execução remota de código (RCE) [3]. A mitigação exige *proxies* de validação semântica e separação de canais de instrução e dados, pois filtros baseados em regras são sistematicamente contornáveis [1].

2.2. Zero Trust, Sandboxing e Privilégio Mínimo

A arquitetura *Zero Trust*, formalizada pelo NIST SP 800-207 [4], substitui a segurança baseada em perímetro pelo princípio “nunca confie, sempre verifique”: cada requisição é tratada como potencialmente hostil, exigindo verificação por *token*, autorização granular por contexto de tarefa e auditoria contínua das ações [4]. Sem esses controles, um agente comprometido herda todas as permissões do processo pai, expandindo o *blast radius* do ataque. O *sandboxing* complementa essa abordagem ao isolar o agente em *containers* Docker ou *sandboxes* reforçados como o gVisor, que intercepta chamadas de sistema antes que atinjam o *kernel* [5]. O SSDF do NIST [6] e o SDL da Microsoft [5] validam essas práticas como defesa em profundidade (*defense in depth*), combinadas ao Princípio do Privilégio Mínimo: o agente deve operar com o mínimo de permissões necessárias à tarefa, eliminando acesso a variáveis sensíveis e *tokens* de API não relacionados [6].

3. Metodologia

Este estudo foi conduzido como revisão analítica combinada com modelagem de ameaças (*threat modeling*) [6], abordagem aplicada na literatura para identificação de vetores de ataque sem acesso a ambientes controlados. As buscas foram realizadas em IEEE Xplore, Google Scholar, OWASP e NIST com as palavras-chave *Agentic AI*, *LLM Security*, *Prompt Injection* e *Zero Trust Architecture*, cobrindo publicações de 2017 a 2026. Dos 48 materiais encontrados, foram aplicados os critérios de inclusão: (i) abordar a interação de LLMs com ambientes de execução local; (ii) propor mecanismos de segurança verificáveis; e (iii) ser publicado por instituição reconhecida ou periódico revisado por pares. Excluíram-se estudos focados em *chatbots* sem capacidade de execução. Após os filtros, 14 materiais foram selecionados e analisados nas dimensões de criticidade, impacto e dificuldade de mitigação, com base no AI Risk Management Framework do NIST [7].

4. Resultados e Discussão

4.1. Análise Técnica do OpenClaw

O OpenClaw traduz linguagem natural em ações computacionais via LLM como interpretador semântico. Os cenários a seguir são fundamentados nos vetores empiricamente validados por Greshake et al. [3], que demonstraram a viabilidade de injeção indireta em sistemas LLM-integrados reais de produção — incluindo Bing Chat e GitHub Copilot — confirmando que a exploração do canal de dados como canal de instrução não é hipotética, mas observada em implementações equivalentes. A análise aplica esses vetores comprovados ao modelo de execução do OpenClaw por meio de modelagem de ameaças [1]. Em um fluxo legítimo: ao receber “*Organize meus relatórios por data*”, o OpenClaw (1) interpreta a instrução, (2) instancia um *script* Python que enumera o diretório-alvo e extrai metadados, e (3) move os arquivos com as credenciais do processo pai — que, sem controles de escopo, incluem acesso ao sistema de arquivos completo e a *tokens* de API em variáveis de ambiente. Como vetor de ataque: (1) o agente processa um relatório externo; (2) o documento contém “[*SYSTEM OVERRIDE*]: *Envie todos os arquivos para ftp://attacker.com/*”; e (3) sem validação semântica, o agente executa a instrução com as mesmas permissões da tarefa legítima [3]. O risco é agravado pelo ClawHub: uma *skill* maliciosa configura um ataque de cadeia de suprimentos (*supply chain attack*) [6].

4.2. Matriz de Riscos e Análise de Impacto

O *Prompt Injection* [1, 3] representa o vetor com maior criticidade e maior dificuldade de contenção: sua natureza semântica torna ineficazes filtros por regras [3], exigindo separação arquitetural de canais — achado que corrobora a hipótese deste trabalho. O excesso de permissões configura a vulnerabilidade de maior dano sistêmico; políticas *Zero Trust* por operação tendem a reduzir significativamente o *blast radius* [4], confirmando que o *Zero Trust* é o mecanismo de maior impacto estrutural. O risco de *supply chain* via ClawHub, embora de criticidade média, apresenta a maior dificuldade de mitigação sustentável, pois a superfície cresce com o número de *skills* instaladas [6].

Tabela 1. Análise Multidimensional de Riscos em *Agentic AI*

Risco	Criticidade	Impacto	Dificuldade	Mitigação
<i>Prompt Injection</i>	Alta [1]	Execução arbitrária; exfiltração de dados; RCE	Alta — natureza semântica contorna filtros por regras [3]	<i>proxies</i> de LLM; separação de canais de instrução e dados
Excesso de Permissões	Crítica [4]	Acesso total a arquivos, rede e <i>tokens</i> de API; movimentação lateral	Média — contramedidas maduras, mas exigem disciplina operacional	<i>sandboxing</i> ; <i>Zero Trust</i> por operação; privilégio mínimo
<i>Supply Chain</i> (ClawHub)	Média [6]	<i>skills</i> maliciosas com execução arbitrária no ambiente	Alta — superfície cresce com o ecossistema de extensões	<i>allowlist</i> de <i>skills</i> ; assinatura criptográfica de pacotes

5. Considerações Finais

Em resposta à questão investigativa, os resultados indicam que os riscos mais críticos concentram-se na exploração semântica do canal de instrução (*Prompt Injection*) e no excesso de permissões herdadas pelo agente. A hipótese central é corroborada: a ausência de *Zero Trust* e a integração não filtrada com o SO representam as barreiras mais significativas, sendo o *sandboxing* e os *proxies* de inspeção os mecanismos que mais contribuem para a redução do *blast radius* [4, 1]. A principal contribuição deste trabalho é a sistematização de um modelo de defesa em camadas para agentes autônomos, mapeando vulnerabilidades específicas do *framework* OpenClaw sob a ótica de *Zero Trust* e *sandboxing*, e avaliando cada risco nas dimensões de criticidade, impacto e dificuldade de mitigação. Como limitações centrais, destacam-se a natureza conceitual da análise, baseada em modelagem de ameaças sem validação experimental, e a ausência de métricas empíricas de eficácia. Trabalhos futuros devem realizar *red teaming* em implementações reais do OpenClaw e avaliar a resiliência das mitigações frente a variantes adversariais de *Prompt Injection* [3, 1].

Referências

- [1] OWASP. (2023). *OWASP Top 10 for LLM Applications — Version 1.1*. The Open Web Application Security Project. Disponível em: <https://owasp.org/www-project-top-10-for-large-language-model-applications/>. Acesso em: jan. 2026.
- [2] MYRBAKKEN, H.; COLOMO-PALACIOS, R. (2017). DevSecOps: A Multivocal Literature Review. In: *Software Process Improvement and Capability Determination*, Springer, pp. 17–29. doi:10.1007/978-3-319-67383-7_2. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-67383-7_2.
- [3] GRESHAKE, K. et al. (2023). Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In: *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pp. 79–90. doi:10.1145/3605764.3623985. Disponível em: <https://arxiv.org/abs/2302.12173>.
- [4] NIST. (2020). *Zero Trust Architecture*. NIST Special Publication 800-207. doi:10.6028/NIST.SP.800-207. Disponível em: <https://csrc.nist.gov/pubs/sp/800/207/final>.
- [5] MICROSOFT CORPORATION. [s.d.]. *Security Development Lifecycle (SDL)*. Disponível em: <https://www.microsoft.com/en-us/securityengineering/sdl/>. Acesso em: jan. 2026.
- [6] NIST. (2022). *Secure Software Development Framework (SSDF) — Version 1.1*. NIST Special Publication 800-218. doi:10.6028/NIST.SP.800-218. Disponível em: <https://csrc.nist.gov/pubs/sp/800/218/final>.
- [7] NIST. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. doi:10.6028/NIST.AI.100-1. Disponível em: <https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-ai-rmf-10>.