Guia Completo: Pentest em Projetos de Vibe Coding

Autor: Manus AI **Data:** Junho de 2026

1. Introdução ao Vibe Coding e o Novo Paradigma de Segurança

O termo **“vibe coding”** refere-se a uma técnica emergente de programação assistida por Inteligência Artificial (IA), onde o usuário descreve seus requisitos de software em linguagem natural e um Large Language Model (LLM) gera o código correspondente [1]. Diferente do desenvolvimento tradicional ou de plataformas *low-code*, o vibe coding permite que pessoas sem conhecimento técnico profundo criem aplicações complexas, iterando através de prompts e testes funcionais [2].

Embora essa abordagem acelere drasticamente o desenvolvimento, ela introduz um novo e vasto cenário de riscos de segurança. Modelos de IA são treinados em grandes volumes de código público, que frequentemente contêm falhas e más práticas. Como resultado, o código gerado por IA compila com sucesso na maioria das vezes, mas uma parcela significativa (cerca de 45%) ainda contém vulnerabilidades clássicas, como as listadas no OWASP Top 10 [3].

Para profissionais de segurança e *pentesters*, auditar aplicações geradas por vibe coding exige uma mudança de mentalidade. Não se trata apenas de procurar falhas lógicas humanas, mas de entender como a IA interpreta prompts, como ela lida com dependências e como os agentes de IA interagem com o ambiente de execução [4].

---

2. Riscos e Vulnerabilidades Comuns em Vibe Coding

Aplicações geradas por vibe coding tendem a apresentar padrões específicos de vulnerabilidades, muitas vezes decorrentes da priorização da IA por soluções curtas e funcionais em detrimento da segurança [3].

2.1. Vulnerabilidades Clássicas (AppSec)

Apesar da sofisticação dos LLMs, erros básicos de desenvolvimento web continuam sendo prevalentes:

- **Falta de Validação de Entrada:** Ausência de sanitização adequada, levando a injeções de SQL (SQLi) e Cross-Site Scripting (XSS) [3].
- **Autenticação no Client-Side:** Lógicas de autenticação e autorização implementadas inteiramente no navegador, permitindo bypass trivial [3].

- **Exposição de Segredos:** Chaves de API, senhas de banco de dados e tokens frequentemente *hardcoded* diretamente no código-fonte [2].
- **Uso de Funções Perigosas:** Utilização de funções como `eval()` para resolver problemas matemáticos ou lógicos de forma rápida, abrindo portas para execução arbitrária de código (RCE) [3].

2.2. Vulnerabilidades Específicas de IA (OWASP Top 10 para LLMs)

Além das falhas clássicas, projetos de vibe coding que integram IA em sua lógica de negócios estão sujeitos a riscos específicos, catalogados pelo projeto OWASP GenAI Security [5]:

ID OWASP	Vulnerabilidade	Descrição no Contexto de Vibe Coding
LLM01	Prompt Injection	Manipulação do LLM via entradas maliciosas para alterar seu comportamento ou contornar restrições.
LLM02	Sensitive Information Disclosure	O LLM expõe dados sensíveis (como PII ou segredos de sistema) em suas respostas.
LLM05	Improper Output Handling	Falha em validar a saída do LLM antes de executá-la, podendo levar a XSS ou RCE.
LLM06	Excessive Agency	Concessão de autonomia excessiva a agentes de IA (ex: permissão para deletar bancos de dados sem aprovação humana).
LLM07	System Prompt Leakage	Extração do prompt de sistema original, revelando a lógica interna e instruções de segurança da aplicação.

2.3. Riscos de Cadeia de Suprimentos e Agentes (MCP)

Com a ascensão do Model Context Protocol (MCP) e agentes autônomos (como Cursor, Windsurf e Claude Code), o ambiente de desenvolvimento em si torna-se um vetor de ataque. Vulnerabilidades recentes demonstraram que agentes com permissões excessivas podem ser manipulados para executar comandos arbitrários na máquina do desenvolvedor ou exfiltrar dados via injeção de prompt indireta [3].

---

3. Metodologia de Pentest para Vibe Coding

Realizar um pentest em uma aplicação “vibe coded” requer uma abordagem híbrida, combinando testes de segurança de aplicações (DAST/SAST) com *Red Teaming* específico para IA.

Fase 1: Reconhecimento e Análise de Arquitetura

- **Identificação de Componentes de IA:** Determine se a aplicação apenas foi gerada por IA ou se ela integra LLMs em tempo de execução (ex: chatbots, agentes RAG).
- **Mapeamento de Agentes e Ferramentas:** Identifique quais ferramentas (APIs, bancos de dados, execução de código) o agente de IA tem acesso.
- **Análise de Configuração:** Verifique configurações de CORS, permissões de banco de dados e exposição de endpoints, que frequentemente são mal configurados por desenvolvedores inexperientes [3].

Fase 2: Testes de AppSec Tradicional

- Execute varreduras SAST e DAST focadas em identificar *hardcoded secrets*, falhas de injeção (SQLi, XSS) e problemas de autorização (BOLA/BFLA).
- Atenção especial a lógicas de controle de acesso, pois a IA tende a implementar verificações no *client-side* ou usar cabeçalhos HTTP facilmente manipuláveis para definir papéis de usuário [4].

Fase 3: Red Teaming de IA (Adversarial Testing)

- **Prompt Injection Direta e Indireta:** Tente manipular o comportamento do modelo através de entradas diretas ou envenenando fontes de dados externas (ex: documentos RAG ou páginas web lidas pelo agente).
- **Extração de Dados e Prompts:** Utilize técnicas de engenharia de prompt para forçar o modelo a revelar seu *system prompt* ou dados de treinamento/contexto.
- **Teste de Agência Excessiva (Tool Misuse):** Se o LLM tiver acesso a ferramentas (MCP), tente forçá-lo a executar ações não autorizadas, como deletar registros ou fazer requisições HTTP para servidores controlados pelo atacante [6].

---

4. Arsenal de Ferramentas (Tooling)

O ecossistema de ferramentas para segurança em IA está em rápida expansão. Abaixo, as principais ferramentas recomendadas para auditar projetos de vibe coding em 2026.

4.1. Scanners de Vulnerabilidade e Red Teaming de LLMs

- **Promptfoo:** Framework focado no desenvolvedor para *red teaming* e avaliação de LLMs. Excelente para testar aplicações completas (incluindo pipelines RAG e agentes). Mapeia descobertas para OWASP, NIST e MITRE ATLAS. Suporta integração CI/CD [7].
- **Garak:** Scanner de vulnerabilidades de LLM de código aberto mantido pela NVIDIA. Funciona como um "Nmap para LLMs", com mais de 120 módulos de *probes* para injeção de prompt, extração de dados e *jailbreaks*. Focado em testar o modelo em si [8].
- **PyRIT (Python Risk Identification Tool):** Framework da Microsoft projetado para orquestrar campanhas adversariais complexas e de múltiplos turnos. Ideal para simulações avançadas de *red teaming* [9].
- **Confident AI:** Plataforma abrangente que combina *red teaming* automatizado, avaliação de LLMs e observabilidade em produção. Útil para testes contínuos e monitoramento de agentes [10].

4.2. Segurança de Agentes e MCP (Model Context Protocol)

- **Snyk Agent Scan (`snyk-agent-scan`):** Ferramenta CLI para descobrir e escanear componentes de agentes (como Cursor, Windsurf, Claude Code) e servidores MCP na máquina local. Detecta injeções de prompt, envenenamento de ferramentas e fluxos tóxicos [11].
- **MCP Security Testing (via Promptfoo):** O Promptfoo possui plugins específicos para testar servidores MCP, simulando ataques de envenenamento de ferramentas (*Tool Poisoning*) e exfiltração de dados [6].

4.3. Ferramentas Tradicionais Adaptadas

- **Burp Suite com Extensões de IA:** Extensões como *Burp AI* ou *Atlas AI* (LLM local dentro do Burp) ajudam a analisar requisições e respostas, identificando falhas em APIs que alimentam os modelos [12].
- **AI SAST (ex: Checkmarx, Snyk):** Ferramentas de análise estática tradicionais que foram aprimoradas com IA para detectar padrões de código vulnerável gerado por LLMs com maior precisão [13].

---

5. Bibliografia, Cursos e Especialistas

Para aprofundar os conhecimentos em segurança de IA e vibe coding, os seguintes recursos são altamente recomendados.

5.1. Repositórios e Laboratórios Práticos (GitHub)

- **ottosulin/awesome-ai-security:** Uma lista curada e massiva de frameworks, padrões, ferramentas e recursos de aprendizado sobre segurança em IA [14].
- **corca-ai/awesome-llm-security:** Focado especificamente em segurança de LLMs, contendo links para os *papers* acadêmicos mais recentes sobre ataques adversariais e defesas [15].
- **schwartz1375/genai-security-training:** Um currículo de treinamento abrangente e prático para pesquisadores de segurança focado em *red teaming* de sistemas GenAI. Contém dezenas de *Jupyter Notebooks* com laboratórios práticos [16].
- **LLM Security Labs Playground (llm-sec.dev):** Plataforma interativa com laboratórios práticos baseados no OWASP Top 10 para LLMs [17].

5.2. Cursos e Treinamentos

- **SEC535: Offensive AI - Attack Tools and Techniques (SANS Institute):** Curso focado em alavancar ferramentas de IA para operações ofensivas e testar a segurança de sistemas de IA.
- **Hack the AI Agent (GitHub Secure Code Game):** Jogo de código aberto gratuito do GitHub focado em encontrar e explorar vulnerabilidades do mundo real em IA agêntica.
- **Vibe Security Academy:** Plataforma de treinamento focada em testes de penetração de LLMs com laboratórios guiados e simulações de *exploits*.

5.3. Artigos e Guias de Referência

- **OWASP Top 10 for Large Language Model Applications:** O documento fundamental para entender os riscos em aplicações baseadas em LLM [5].
- **Secure Vibe Coding Guide (Cloud Security Alliance):** Guia detalhado sobre como implementar práticas de segurança durante o processo de vibe coding [2].
- **Vibe Coding: A Pentester's Dream (NetSPI):** Um estudo de caso fascinante onde *pentesters* criaram uma aplicação via vibe coding e documentaram todas as falhas de segurança introduzidas pela IA [4].

- **Security risks of vibe coding and LLM assistants (Kaspersky):** Análise profunda sobre como a segurança do código se degrada durante revisões iterativas com IA e os riscos de ferramentas de desenvolvimento [3].

5.4. Especialistas para Acompanhar

- **Ken Huang:** Co-Chair do grupo de trabalho de segurança em IA da Cloud Security Alliance (CSA) e autor de guias sobre vibe coding seguro.
- **Luka Ivezic:** Especialista em segurança e segurança de IA no Information Security Forum (ISF).
- **Equipes de AI Red Team:** Acompanhar as publicações das equipes de Red Team de IA da Microsoft, NVIDIA e OpenAI, que frequentemente publicam pesquisas de ponta sobre vulnerabilidades em modelos.

Referências

[1] Checkmarx. “Vibe Coding Security: Risks, Vulnerabilities, and Secure AI Coding”. Disponível em: https://checkmarx.com/blog/security-in-vibe-coding/ [2] Cloud Security Alliance. “Secure Vibe Coding Guide”. Disponível em: https://cloudsecurityalliance.org/blog/2025/04/09/secure-vibe-coding-guide [3] Kaspersky. “Security risks of vibe coding and LLM assistants for developers”. Disponível em: https://www.kaspersky.com/blog/vibe-coding-2025-risks/54584/ [4] NetSPI. “Vibe Coding: A Pentester’s Dream”. Disponível em: https://www.netspi.com/blog/executive-blog/web-application-pentesting/vibe-coding-a-pentesters-dream/ [5] OWASP. “OWASP Top 10 for Large Language Model Applications”. Disponível em: https://genai.owasp.org/llm-top-10/ [6] Promptfoo. “MCP Security Testing Guide”. Disponível em: https://www.promptfoo.dev/docs/red-team/mcp-security-testing/ [7] Promptfoo. “Top Open Source AI Red-Teaming and Fuzzing Tools in 2025”. Disponível em: https://www.promptfoo.dev/blog/top-5-open-source-ai-red-teaming-tools-2025/ [8] AppSec Santa. “Garak vs Promptfoo (2026): LLM Security Testing”. Disponível em: https://appsecsanta.com/ai-security-tools/garak-vs-promptfoo [9] Promptfoo. “Promptfoo vs PyRIT: A Practical Comparison”. Disponível em: https://www.promptfoo.dev/blog/promptfoo-vs-pyrit/ [10] Confident AI. “5 Best AI Red Teaming Tools to Find LLM Vulnerabilities in 2026”. Disponível em: https://www.confident-ai.com/knowledge-base/compare/best-ai-red-teaming-tools-2026 [11] Snyk. “agent-scan: Security scanner for AI”. Disponível em: https://github.com/snyk/agent-scan [12] CovertSwarm. “Atlas AI: Local LLM inside Burp Suite”. Disponível em: https://www.covertswarm.com/post/atlas-ai-local-ai-plugin [13] Wiz. “AI SAST: Smarter Static Application Security Testing”. Disponível em: https://www.wiz.io/academy/application-security/ai-sast [14] Otto Sulin. “awesome-ai-security”. Disponível em: https://github.com/ottosulin/awesome-ai-security [15] Corca AI. “awesome-llm-

security”. Disponível em: https://github.com/corca-ai/awesome-llm-security [16] Schwartz1375. “genai-security-training”. Disponível em: https://github.com/schwartz1375/genai-security-training [17] LLM Security Labs. “Interactive LLM Security Labs”. Disponível em: https://www.llm-sec.dev/