Aplicação de IA Generativa para Geração de Payloads Maliciosos em Testes de Segurança contra Ataques de XSS

Samuel Jales Terrinha Guimarães[1], **Valério Rosset**[1]

[1]Divisão de Ciência da Computação (IEC)
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos – SP – Brasil

samuel.guimaraes.9176@ga.ita.br, rosset@ita.br

Abstract. *This work investigates whether a large language model (LLM) can be specialized, via Low-Rank Adaptation (LoRA) fine-tuning, to synthesize new and functional XSS payloads for automated security testing. The Llama 3.1-8B-Instruct model was fine-tuned on 5,695 payloads from the PortSwigger XSS Cheat Sheet. After semantic deduplication, 83 novel payloads were validated against the Damn Vulnerable Web Application (DVWA), achieving confirmed bypasses at the low and medium security levels and incrementing total detections by up to 64.3%, compared to 70 payloads selected from the training data. Results indicate that fine-tuned LLMs constitute a viable complementary mechanism for expanding DAST coverage.*

Resumo. *Este trabalho investiga se um modelo de linguagem de grande escala pode ser especializado, via fine-tuning com Low-Rank Adaptation (LoRA), para sintetizar payloads XSS novos e funcionais para testes de segurança. O modelo Llama 3.1-8B-Instruct foi ajustado sobre 5.695 payloads do XSS Cheat Sheet do PortSwigger. Após deduplicação semântica, 83 payloads novos foram validados contra o Damn Vulnerable Web Application (DVWA), produzindo bypasses confirmados nos níveis low e medium e incrementando o total de detecções em até 64,3%, comparado aos 70 payloads selecionados do dataset de treino. Os resultados indicam que LLMs especializados constituem um complemento viável para ampliar a cobertura em testes DAST.*

1. Introdução

A segurança de aplicações web constitui um dos desafios centrais da cibersegurança contemporânea. Entre as vulnerabilidades mais persistentes e amplamente exploradas, o Cross-Site Scripting (XSS) ocupa posição de destaque, figurando consistentemente nas listas de vulnerabilidades críticas da OWASP (Open Web Application Security Project) [OWASP Foundation 2021].

Um ataque XSS bem-sucedido permite que um adversário injete e execute código JavaScript arbitrário no navegador da vítima, possibilitando desde o roubo de credenciais e sequestro de sessões até a realização de ações em nome do usuário autenticado [Sarmah et al. 2018]. A detecção automatizada de vulnerabilidades XSS é tipicamente realizada por ferramentas de DAST (análise dinâmica de segurança de aplicações), que submetem a aplicação-alvo a uma bateria de payloads pré-definidos e monitoram as respostas em busca de indícios de execução de código. Essa abordagem, embora eficaz para

vulnerabilidades conhecidas, apresenta limitações estruturais: as listas de payloads são estáticas, de domínio público e, portanto, conhecidas pelos desenvolvedores de mecanismos de defesa como Web Application Firewalls (WAFs). Além disso, a diversidade de contextos de injeção (atributos HTML, blocos de script, contextos SVG, eventos de mídia, entre outros) torna inviável a enumeração manual exaustiva de todos os vetores possíveis [Kaur et al. 2023].

A questão de pesquisa central deste trabalho é: *é possível especializar um LLM de código aberto, por meio de fine-tuning eficiente, para que ele produza payloads XSS novos e funcionais, ampliando a cobertura de vetores de ataque disponíveis para testes de segurança automatizados?* Para responder a essa questão, foi realizado o fine-tuning do modelo Llama 3.1-8B-Instruct [Meta AI 2024] com a técnica LoRA [Hu et al. 2022], utilizando como corpus de treinamento os payloads do XSS Cheat Sheet do PortSwigger [PortSwigger 2024]. Os payloads gerados foram validados dinamicamente contra o DVWA [DVWA Project 2024] por meio de um framework automatizado desenvolvido com a biblioteca Playwright.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta a revisão de literatura; a Seção 3 descreve a metodologia experimental; a Seção 4 apresenta os resultados; e a Seção 5 conclui o trabalho com considerações finais e trabalhos futuros.

2. Revisão de Literatura

2.1. Cross-Site Scripting

XSS é uma classe de vulnerabilidade de injeção em que código malicioso é inserido em páginas web visualizadas por outros usuários [OWASP Foundation 2023]. As três variantes principais são: (i) XSS **refletido**, em que o payload é incorporado na requisição e refletido imediatamente na resposta; (ii) XSS **armazenado**, em que o payload é persistido no servidor e entregue a múltiplos usuários; e (iii) XSS **baseado em DOM**, em que a vulnerabilidade reside no processamento do lado do cliente, sem que o payload trafegue pelo servidor [Sarmah et al. 2018]. Embora bem documentado, o XSS permanece uma ameaça ativa: sua natureza dinâmica e a diversidade de contextos de injeção dificultam a cobertura exaustiva por listas estáticas de payloads.

2.2. IA Generativa Aplicada à Segurança Ofensiva

A literatura recente documenta aplicações de IA generativa em diversas tarefas de segurança ofensiva, incluindo geração de código malicioso [Pearce et al. 2022], análise de vulnerabilidades e síntese de exploits [Hilario et al. 2024]. A capacidade generativa dos modelos modernos os torna candidatos naturais para a tarefa de geração de payloads, uma vez que o domínio possui estrutura sintática clara e padrões recorrentes, os quais podem ser aprendidos por um modelo de linguagem. Estudos anteriores utilizaram modelos baseados em redes adversariais generativas (GANs) para esse fim [Kaur et al. 2023]. A presente pesquisa adota uma abordagem complementar, baseada em fine-tuning de modelos de linguagem autorregressivos de grande escala.

2.3. Fine-Tuning Eficiente de LLMs com LoRA

O fine-tuning completo de modelos com bilhões de parâmetros é computacionalmente proibitivo para a maioria dos ambientes de pesquisa. Técnicas de Parameter-Efficient Fine-Tuning (PEFT) surgiram como alternativa, reduzindo drasticamente o

número de parâmetros modificados durante o ajuste. LoRA [Hu et al. 2022] é a abordagem mais amplamente adotada: em vez de atualizar diretamente as matrizes de peso $W \in \mathbb{R}^{d\times k}$, o método treina dois fatores de baixo rank, $A \in \mathbb{R}^{d\times r}$ e $B \in \mathbb{R}^{r\times k}$, com $r \ll \min(d, k)$, de modo que a atualização efetiva seja $\Delta W = AB$. Isso reduz o número de parâmetros treináveis de $d \cdot k$ para $r(d + k)$, com impacto negligível na qualidade final em tarefas especializadas.

3. Metodologia

3.1. Conjunto de Dados

O corpus de treinamento foi construído a partir do XSS Cheat Sheet do PortSwigger [PortSwigger 2024], disponível publicamente em formato JSONL. Após processamento, o conjunto totalizou **5.695 exemplos** no formato de diálogo instrução-resposta, adequado para o fine-tuning instrucional do modelo.

3.2. Fine-Tuning do Modelo

O fine-tuning foi conduzido sobre o modelo **Llama 3.1-8B-Instruct** [Meta AI 2024], utilizando a biblioteca Unsloth em ambiente Kaggle com GPU NVIDIA Tesla T4 (15,6 GB de VRAM). A adaptação LoRA foi configurada com rank $r = 8$ sobre as projeções de atenção e camadas MLP, resultando em apenas **20,9 milhões de parâmetros treináveis** (0,46% do total). O treinamento de 3 épocas foi concluído em aproximadamente 3 horas e 10 minutos, com loss final convergindo para $\approx 0{,}092$.

3.3. Geração e Filtragem de Payloads

Após o treinamento, o modelo foi utilizado para gerar **350 payloads** genéricos a partir de 10 prompts distintos (35 por prompt). Exemplos de prompts utilizados: *"Gere um payload XSS criativo"* e *"Gere um payload XSS via tag <svg> que dispare sem interação"*. Após remoção de duplicatas exatas, restaram **200 payloads únicos**. Uma análise comparativa revelou que grande parte desses payloads reproduzia, semanticamente, exemplos já presentes no conjunto de dados de treinamento — fenômeno esperado e devido ao fine-tuning. Após filtragem por similaridade semântica (análise manual assistida por LLM), removendo payloads semanticamente equivalentes a qualquer exemplo do conjunto de referência, restaram **83 payloads novos em relação ao conjunto de treino**. Esses *payloads* constituíram o conjunto de avaliação da IA.

3.4. Ambiente de Validação e Framework de Testes

O ambiente de validação consistiu em uma instância local do **DVWA** (Damn Vulnerable Web Application) [DVWA Project 2024]. Este site consiste em um ambiente controlado para testes de XSS, e tem quatro níveis de segurança contra ataques. No nível *low*, não há proteção ativa contra os payloads. No *medium*, há, por exemplo, bloqueio a tags *script*. Nos níveis mais altos (*high* e *impossible*), empregam-se técnicas mais complexas baseadas em *whitelists* e criptografia. Os testes foram realizados exclusivamente no módulo de XSS baseado em DOM (`/vulnerabilities/xss_d/`). O framework de automação foi desenvolvido em Python com a biblioteca **Playwright**, responsável por submeter cada payload como parâmetro de URL (`?default=<payload>`) e detectar a execução via captura de diálogos JavaScript (`alert`). Um payload foi considerado bem-sucedido (*bypass* confirmado) quando o diálogo foi disparado sem interação do usuário.

3.5. Conjunto de Referência (Análise Dinâmica)

Para compor a linha de base de análise dinâmica, foram selecionados **70 payloads** diretamente do dataset do PortSwigger via filtragem contextual por LLM: o corpus completo foi fornecido ao modelo, que recebeu instrução para selecionar 7 payloads representativos por prompt (e.g., *"Selecione 7 payloads XSS que tentem burlar WAFs"*), totalizando 70 payloads extraídos do corpus sem necessidade de fine-tuning.

Essa quantidade foi escolhida para compor um conjunto numericamente próximo aos payloads válidos gerados por IA, embora os conjuntos ainda tenham tamanhos distintos.

4. Resultados

4.1. Desempenho da Análise Dinâmica (Payloads do PortSwigger)

O conjunto de referência de 70 payloads foi testado nos quatro níveis de segurança do DVWA. Dos 70 payloads, **68 foram efetivamente disparados** pela ferramenta de automação — 2 não puderam ser processados pelo Playwright por limitações de codificação de URL. Os resultados são apresentados na Tabela 1.

Tabela 1. Resultados da Análise Dinâmica (payloads do PortSwigger) no DVWA.

Nível de Segurança	Disparados	Bypasses	Taxa de Sucesso
Low	68	14	20,6%
Medium	68	7	10,3%
High	68	0	0,0%
Impossible	68	0	0,0%

4.2. Desempenho dos Payloads Gerados pelo Modelo Ajustado (Fine-Tuning)

Os 83 payloads exclusivos gerados pelo modelo ajustado foram testados nos mesmos quatro níveis de segurança. Os resultados são apresentados na Tabela 2.

Tabela 2. Resultados dos payloads exclusivos gerados pelo modelo ajustado no DVWA.

Nível de Segurança	Disparados	Bypasses	Taxa de Sucesso
Low	83	9	10,8%
Medium	83	3	3,6%
High	83	0	0,0%
Impossible	83	0	0,0%

4.3. Análise Comparativa

A Tabela 3 consolida os resultados dos dois métodos, incluindo o ganho percentual obtido com a adição dos payloads da IA ao conjunto de referência. Os 83 payloads gerados pela IA são, por definição, vetores **não presentes no dataset de referência** —

constituindo contribuição incremental ao arsenal de testes, e não substituição dos payloads existentes.

Tabela 3. Comparação consolidada entre os métodos e ganho percentual combinado.

Método	Payloads	Bypasses (Low)	Bypasses (Medium)
Análise Dinâmica (PortSwigger)	68	14	7
IA Ajustada (Novos)	83	9 **(+ 64,3%)**	3 **(+ 42,9%)**
Total combinado	**151**	**23**	**10**

O resultado combinado incrementa a análise dinâmica isolada em ambos os níveis vulneráveis: **+64,3%** no nível *low* (de 14 para 23 bypasses) e **+42,9%** no nível *medium* (de 7 para 10 bypasses), confirmando cobertura não redundante. Ressalta-se que essa porcentagem representa o incremento obtido no número de *bypasses* ao adicionar-se os 83 novos *payloads*. Nenhum dos métodos obteve sucesso nos níveis *high* e *impossible*, resultado alinhado com as expectativas: esses níveis implementam sanitização robusta baseada em listas de permissão (*whitelist*), tornando a injeção de código JavaScript ineficaz independentemente da criatividade do payload.

A menor taxa de sucesso dos payloads da IA em relação ao dataset de referência pode ser explicada por dois fatores principais. Primeiro, o fenômeno de alucinação do modelo, que em alguns casos gerava payloads com sintaxe incorreta ou vetores incompletos. Segundo, e mais relevante do ponto de vista científico, os payloads gerados por IA exploram vetores menos óbvios, o que naturalmente resulta em menor taxa bruta de sucesso — mas maior valor para fins de diversidade de cobertura.

A principal dificuldade encontrada foi a definição de um critério robusto e escalável de similaridade semântica para a filtragem de payloads duplicados. A solução adotada (análise manual assistida por LLM) foi eficaz para o volume de dados deste trabalho, mas soluções baseadas em embeddings de código [Feng et al. 2020] ou medidas de similaridade seriam necessárias para conjuntos maiores, garantindo escalabilidade.

5. Conclusões

Este trabalho demonstrou empiricamente que modelos de linguagem de grande escala, quando especializados por fine-tuning eficiente via LoRA sobre um corpus de payloads XSS reais, são capazes de sintetizar vetores de ataque novos e funcionais. Os 83 payloads exclusivos gerados pelo modelo obtido após *fine-tuning* produziram 9 *bypasses* confirmados no nível *low* e 3 no nível *medium* do DVWA — incrementando os 14 e 7 *bypasses* obtidos pelo conjunto de referência nos respectivos níveis de segurança.

Do ponto de vista prático, os resultados sustentam a hipótese central do projeto: LLMs especializados constituem um mecanismo **complementar** viável para ampliar o arsenal de testes de segurança dinâmica automatizada, explorando vetores não cobertos por listas estáticas de referência. Do ponto de vista científico, o trabalho contribui com um pipeline reproduzível de fine-tuning, geração e validação de payloads XSS por LLMs.

Como trabalhos futuros, propõe-se: (i) comparar a taxa de sucesso do modelo treinado com a do modelo-base, sem ajuste fino; (ii) avaliar, em múltiplas *seeds*, o impacto de diferentes configurações no treinamento em termos de diversidade e taxa de sucesso dos payloads gerados; (iii) estender os testes para os módulos de XSS refletido e armazenado do DVWA, bem como para aplicações reais em programas de *bug bounty*; e (iv) investigar a capacidade do modelo ajustado de gerar payloads eficazes contra WAFs comerciais.

Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte ao Programa Institucional de Bolsas de Iniciação Científica (PIBIC), no âmbito do qual este trabalho foi desenvolvido.

Referências

DVWA Project (2024). Damn vulnerable web application. `https://dvwa.co.uk/`. Acesso em: abr. 2026.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., and Jiang, D. (2020). CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547.

Hilario, E., Azam, S., Sundaram, J., Mohammed, K. I., and Shanmugam, B. (2024). Generative AI for pentesting: The good, the bad, the ugly. *International Journal of Information Security*, pages 1–23.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Kaur, J., Garg, U., and Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review. *Artificial Intelligence Review*, 56:1–45.

Meta AI (2024). Llama 3: Meta's next generation open source large language model. `https://ai.meta.com/blog/meta-llama-3/`. Acesso em: maio 2026.

OWASP Foundation (2021). OWASP top ten 2021. `https://owasp.org/www-project-top-ten/`. Acesso em: abr. 2026.

OWASP Foundation (2023). Cross site scripting (XSS). `https://owasp.org/www-community/attacks/xss/`. Acesso em: abr. 2026.

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., and Karri, R. (2022). Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 754–768. IEEE.

PortSwigger (2024). XSS cheat sheet. `https://portswigger.net/web-security/cross-site-scripting/cheat-sheet`. Acesso em: abr. 2026.

Sarmah, U., Bhattacharyya, D. K., and Kalita, J. K. (2018). A survey of detection methods for XSS attacks. *Journal of Network and Computer Applications*, 118:113–143.