Investigating the use of LLMs for Addressing Injection Vulnerabilities.

Pedro B. Rodrigues Pinto
LabES, ICMC, University of São Paulo
São Carlos - SP, Brazil
pbernardorp21@usp.br

Carolina Gomes Guerreiro
LabES, ICMC, University of São Paulo
São Carlos - SP, Brazil
carolina.guerreiro@usp.br

Lina Garcés
LabES, ICMC, University of São Paulo
São Carlos - SP, Brazil
linagarces@usp.br

Abstract

Large Language Models (LLMs) have emerged as promising tools for software security testing. This paper presents a Rapid Literature Review of 15 primary studies investigating the use of LLMs for detecting, exploiting, and defending against SQL Injection (SQLi) and Cross-Site Scripting (XSS) vulnerabilities. The review characterizes the objectives of existing approaches, the adopted LLM families, conditioning techniques, datasets, and reported effectiveness. The results show that GPT-based models, prompt engineering, and fine-tuning dominate the literature, while LLM-based approaches consistently outperform traditional security tools in the evaluated scenarios. The review also identifies key research gaps, including the lack of standardized datasets, limited reproducibility, and heterogeneous evaluation methods, providing directions for future research on LLM-assisted software security testing.

Keywords

Software Testing, Security, LLM, Large Language Model, Secondary Study

1 Introduction

Injection vulnerabilities remain among the most critical threats to modern software systems, with SQL Injection (SQLi) and Cross-Site Scripting (XSS) still ranking among the most prevalent ones in the OWASP Top 10 [18]. SQLi exploits insufficient input validation to manipulate database queries, enabling unauthorized data disclosure, modification, or execution of administrative commands [17], while XSS injects malicious client-side scripts into trusted web applications, leading to session hijacking, data theft, and unauthorized user actions [16].

Conventional countermeasures rely on Static and Dynamic Application Security Testing (SAST/DAST), fuzzing, and Web Application Firewalls (WAFs), but often suffer from high false-positive rates and limited effectiveness against sophisticated attacks [4, 23, 28]. Consequently, Large Language Models (LLMs) have attracted attention as a means to improve software security testing.

Modern LLMs, such as GPT, LLaMA, and Qwen, demonstrate remarkable capabilities in code understanding, semantic reasoning, and code generation [14, 26], further enhanced by prompt engineering, in-context learning, Retrieval-Augmented Generation (RAG), and parameter-efficient fine-tuning such as LoRA and PiSSA [8, 23, 24]. They are increasingly employed in both defensive and offensive activities, supporting vulnerability detection, code classification, secure code generation, and automatic WAF rule creation, while also generating realistic attack payloads for penetration testing [4, 5, 9, 11, 20, 25, 26].

Existing secondary studies, however, provide only a partial view of this area, covering general taxonomies [2, 10], Web application testing [3, 7], IoT model-based testing [12], metaheuristic-based testing [1], AI-assisted penetration testing [13], and deep-learning vulnerability prediction [22]. None of them investigates how LLMs are used to detect, generate, or interpret injection-related attacks and defenses, nor compares LLM families, adaptation strategies, evaluation methodologies, benchmark datasets, or effectiveness across injection categories.

This paper presents a Rapid Literature Review (RLR) on primary studies that employ LLMs to detect, exploit, or mitigate injection vulnerabilities. We analyze the purposes for which LLMs are used, the adopted models, prompting and adaptation techniques, datasets, evaluation methodologies, and the reported effectiveness compared with traditional solutions, characterizing the state of the art and identifying trends, open challenges, and future directions.

The remainder of this paper is structured as follows. Section 2 describes the review methods, Section 3 presents results and discussions, and Section 4 concludes with limitations and future work.

2 Methods

This study investigates the use of LLMs for code-injection vulnerabilities. The review addresses five research questions concerning: **RQ1-** the objectives of existing studies; **RQ2-** the adopted LLMs; **RQ3-** the employed techniques (e.g., prompt engineering, RAG, and fine-tuning); **RQ4-** the datasets used, and **RQ5-** the effectiveness of LLM-based approaches.

The review followed the guidelines of Cartaxo et al. [6]. Two researchers searched Scopus and Google Scholar in September 2025 using the following search string:

*(LLM OR "Large Language Model" OR "language model" OR "language processing" OR "NLP" OR "generative artificial intelligence" OR "Generative AI" OR "GenAI")**AND** ("security" OR "privacy" OR "trust" OR "protection") **AND** ("test" OR "testing" OR "code inspection" OR "sql injection" OR "cross-site scripting" OR XSS OR "code injection")*

The search retrieved 77 studies, from which 15 primary studies were selected after duplicate removal and two screening stages. The complete selection process is illustrated in Figure 1.

Figure 1: Literature review process

Studies were included if they proposed or evaluated LLMs for detecting or testing code-injection vulnerabilities. Two reviewers

independently conducted study selection and data extraction, with disagreements resolved by a third reviewer. The extracted data were synthesized using qualitative and narrative analysis following the recommendations of Scannavino et al. [21]. The complete dataset is summarized in Table 1.

3 Results and Discussion

The rapid literature review was conducted between July 2025 and February 2026, resulting in a final set of primary studies published up to September 2025, as summarized in Table 1. Notably, all selected studies were published within the last three years, reflecting the rapid emergence of LLMs and their increasing adoption in software engineering. The extracted data and the synthesis of the findings are presented in the following sections, organized by research question.

3.1 Goals of Applying LLMs for Code Injection

The analysis of the 15 primary studies shows that LLMs are applied to injection vulnerabilities for three main purposes: vulnerability detection, attack generation, and defense enhancement. SQL Injection (SQLi) is the dominant research topic, representing 60% (9/15) of the selected studies. Among these, most focus on automated SQLi detection (44.4%, 4/9), particularly through vulnerability identification and black-box testing [8, 24, 25, 27]. Another 33.3% (3/9) investigate the offensive capabilities of LLMs by generating and optimizing sophisticated SQLi payloads using reinforcement learning or probabilistic techniques [9, 11, 20]. The remaining studies (22.2%, 2/9) combine attack generation with defensive mechanisms, automatically producing Web Application Firewall (WAF) rules and mitigating emerging threats in LLM-enabled applications, such as Prompt-to-SQL injection [4, 19].

Cross-Site Scripting (XSS) is addressed exclusively in 20% (3/15) of the reviewed studies [5, 15, 28]. All of these employ LLMs to improve vulnerability detection and defense, leveraging their semantic understanding capabilities to identify complex XSS attacks, including stored XSS. Furthermore, two studies (66.7%) also exploit LLMs to generate and obfuscate malicious XSS payloads, using these attacks to evaluate existing defenses and enrich training datasets for machine-learning-based security solutions [5, 15]. Overall, LLMs are evolving beyond traditional vulnerability detection into versatile tools supporting both offensive testing and adaptive defense mechanisms.

3.2 Models Used for Code-Injection Detection

The reviewed studies employ a wide range of LLMs, with a clear predominance of GPT-based models. GPT-3.5 was the most frequently adopted model, appearing in 46.7% (7/15) of the studies, followed by GPT-4, used in 40% (6/15), highlighting the central role of the GPT family in injection vulnerability detection and testing [4, 8, 9, 14, 19, 23, 25, 26, 28]. Other models, including GPT-4o, Gemini Pro, and CodeLlama, were reported in 20% (3/15) of the studies, while CodeT5, Llama 2, and Claude 3 appeared in 13.3% (2/15). A diverse set of additional models, such as CodeGen, CodeBERT, GraphCodeBERT, PLBART, WizardCoder, Llama 3, PaLM 2, Command R, and SecureQwen, were each used in only one study, indicating that although the research community is actively exploring alternative LLMs, their adoption remains considerably less widespread than that of GPT models.

3.3 LLM Techniques for Code-Injection Detection

LLM conditioning techniques in the reviewed studies can be broadly categorized into prompt engineering and fine-tuning, the two most widely adopted approaches. Prompt engineering, including zero-shot, few-shot, and Chain-of-Thought (CoT) prompting, was employed in 11 of the 15 studies, while fine-tuning appeared in 7 studies. Prompt-based techniques were particularly common in SQL injection research (7/9 studies) and were also the predominant strategy for XSS detection (2/3 studies), as they improve the models' ability to capture semantic patterns beyond traditional rule-based approaches. More recent adaptation methods, such as PiSSA, further reduce fine-tuning costs while preserving high semantic understanding, representing an efficient alternative to LoRA for cybersecurity tasks [27].

Beyond model adaptation, several studies integrate LLMs with advanced reasoning paradigms. CoT and Tree-of-Thoughts (ToT) improve multi-step reasoning during vulnerability analysis [23], while Reinforcement Learning from Human Feedback (RLHF) is applied in defensive and detection contexts, including automated WAF rule generation and XSS defense [4, 5], and probabilistic N-gram models are used to optimize blind SQLi attacks [20]. Other works employ multi-agent architectures to coordinate multiple LLMs for attack analysis and response planning [8, 25], while defensive frameworks such as XSShield leverage dynamic prompt learning and semantic optimization to strengthen protection against stored XSS attacks. Overall, the literature demonstrates a trend toward combining LLMs with complementary reasoning, learning, and collaborative techniques to improve both vulnerability detection and defensive capabilities.

3.4 Datasets Used for Training and Assessing LLMs for Code Injection

The reviewed studies employed a wide variety of datasets, with no dataset being reused across multiple studies. Most datasets were adapted or extended to address specific research objectives, such as generating new SQL injection payloads [27], balancing class distributions [20], producing obfuscated XSS samples [24], or synthesizing validated attack payloads [23]. This diversity reflects the absence of standardized benchmark datasets for evaluating LLM-based injection vulnerability detection.

Dataset availability also remains a significant challenge. Nearly one-third of the datasets (33%, 5/15) were proprietary or unavailable to the research community [11, 14, 23, 24, 27], while almost half of the studies (46%, 7/15) created new datasets or substantially modified existing ones due to the lack of suitable public alternatives [5, 8, 11, 14, 15, 26, 28]. Data augmentation was further employed in five studies to increase dataset diversity [8, 14, 15, 24, 26]. In contrast, studies relying primarily on prompt-based generative AI required little or no task-specific training data, using available datasets mainly as examples for prompt construction rather than for model fine-tuning [4, 15, 23, 25].

Table 1: Final set of primary studies on LLM-based code-injection vulnerability detection.

Id	Ref.	Year	Purpose	LLM Model(s)	Conditioning Method	Results	Dataset(s)
S01	[27]	2024	SQL injection attack detection.	ChatGLM4-6B.	Prompt engineering and Fine-tuning (LoRA).	99.85% accuracy and 0.26% FPR.	Kaggle SQL injection dataset.
S02	[23]	2024	Benchmarking for vulnerability detection.	GPT-4 Turbo, GPT-4, Claude 3 Opus.	Prompt engineering (CoT, ToT, RCI, Self-Consistency).	GPT-4 achieved F1 = 0.672 via CoT 8-step.	Juliet Java 1.3 (NIST).
S03	[4]	2025	WAF security via GenSQLi.	GPT-4o, Gemini Pro.	Prompt engineering, In-context learning and RLHF.	23 new rules blocked 99% of attacks.	MySQL, PostgreSQL and AWS WAF.
S04	[28]	2025	Stored XSS defense (XSShield).	GPT-4, GPT-3.5-turbo.	Prompt Learning and Semantic Gradient Descent.	GPT-4 achieved 93% accuracy and F1 of 0.9266 on non-obfuscated samples.	BeEF Repository and JS150k.
S05	[26]	2024	Automated Vulnerability Localization (AVL).	16 models (GPT-4, CodeLlama, etc.).	Zero/One-shot prompting and Fine-tuning.	CodeLlama achieved 82.0% F1-score via Fine-tuning.	BV-LOC (BigVul) and SC-LOC (Solidity).
S06	[14]	2025	Multi-class detection (SecureQwen).	SecureQwen (based on CodeQwen1.5-7B).	Fine-tuning (SFT).	F1-scores between 84% and 99% for 14 CWEs.	PythonVulnDB (GitHub and Codeparrot).
S07	[24]	2024	SQLi detection via PiSSA fine-tuning.	Llama 3-7B, ChatGLM3-6B.	PiSSA fine-tuning and Prompt Engineering.	Llama 3 achieved 99.81% accuracy.	SafeSQL-v1 (Hugging Face).
S08	[8]	2024	Scanner efficiency (SqliGPT).	GPT-4, GPT-3.5, Claude-3.	Multi-agent, RAG, CoT and Few-Shot.	Detected 100% of targets (45/45), outperforming other scanners in overall efficiency.	SqliMicroBenchmark (including CVEs).
S09	[5]	2025	Automated XSS defense (GenXSS).	GPT-4o, Gemini Pro.	Few-shot learning, Role Prompting and RLHF.	15 new rules blocked 83%–86% of attacks.	Brute Logic platform.
S10	[9]	2024	SQLi exploit generation.	ChatGPT 3.5 Turbo, Command R.	Fine-tuning.	ChatGPT 3.5 achieved 24.30% success rate (75% better than SQLmap).	OWASP Mutillidae II.
S11	[20]	2023	Blind SQLi extraction optimization (Hakuin).	Variable-length five-gram (every-gram) and unigram probabilistic models.	Pre-trained (schema) and adaptive (content) models.	5.98x more efficient in schema inference.	Stack Exchange and industry DB schemas.
S12	[25]	2024	Black-box SQLi detection via LLMSQLi.	GPT-3.5, GPT-4.	Multi-Agent collaboration and Prompt Engineering.	Detected 100% of targets (15/15), outperforming SQLmap.	SQLiMicroBenchmark (sqli-labs).
S13	[11]	2024	Adversarial attack generation (XploitSQL).	T5 (Text-to-Text Transfer Transformer, 77M parameters), pre-trained for SQL generation.	Actor-Critic Reinforcement Learning (NLPO) and Fine-tuning.	Detection rate reduced by up to 47%.	Combined dataset of 35,574 samples.
S14	[19]	2025	Prompt-to-SQL (P2SQL) risk and defense.	GPT-3.5/4, Llama 2, PaLM 2, etc..	Prompt engineering and Fine-tuning.	"LLM Guard" detected 99.55% of malicious prompts.	Simulated marketplace and GitHub apps.
S15	[15]	2025	ML detection reinforcement for XSS.	CodeT5-small.	Fine-tuning and Prompt Learning.	Random Forest improved from 81.9% to 99.5% accuracy.	Kaggle, GitHub and PortSwigger.

3.5 Effectiveness and Comparison With Traditional Tools

The reviewed literature consistently indicates that LLMs outperform traditional security tools in both vulnerability detection and offensive security testing. Six of the fifteen studies (40%) reported direct comparisons with established static analyzers, vulnerability scanners, or penetration-testing tools, demonstrating superior performance by LLM-based approaches [8, 9, 11, 23, 25, 28]. In source-code analysis, LLMs achieved higher Recall and F1-scores than traditional analyzers such as CodeQL and SpotBugs [23], while the XSShield framework reached 93% accuracy and an F1-score of 0.9266 for Stored XSS detection on non-obfuscated samples; on obfuscated code, XSShield maintained a 72.46% detection rate (F1-score of 0.7018), substantially outperforming conventional JavaScript malware detectors such as Cujo, Zozzle, and JStap, which struggled to detect obfuscated attacks [28].

LLMs also demonstrated remarkable effectiveness in offensive security testing. Compared with widely used tools such as SQLmap, Burp Suite, ZAP, and Arachni, LLM-based frameworks consistently achieved higher attack success rates [8, 9, 25]. For example, ChatGPT-3.5 generated successful SQLi payloads nearly four times more often than SQLmap (24.30% vs. 6.51% success rate); we report this ratio computed directly from the raw success rates, as the original study's own percentage comparison (stated as "75% more effective") appears inconsistent with the data it reports [9]. LLMSQLi and SqliGPT, in turn, achieved 100% detection success against their evaluation targets [8, 25]. Furthermore, XploitSQL showed that combining LLMs with reinforcement learning enables the generation of highly evasive SQLi payloads that significantly reduce the effectiveness of machine-learning detectors and commercial WAFs, decreasing their detection rates by up to 47% [11].

4 Final Remarks

This paper presented a Rapid Literature Review of LLM-based approaches for detecting, exploiting, and mitigating injection vulnerabilities. Fifteen primary studies revealed a research field dominated by SQL Injection, GPT-based models, and prompt engineering and fine-tuning techniques. Overall, LLMs outperformed traditional tools in both detection and adversarial payload generation. However, the lack of standardized public datasets and benchmarking remains a major challenge for reproducibility and fair evaluation.

This review aims to provide a consolidated overview of current research and identify emerging trends that may guide researchers and practitioners adopting LLMs in secure software development. Nevertheless, the study is limited by the scope of a rapid review, the use of only two search engines, and the relatively small number of selected studies. Future work includes extending this review into a full Systematic Literature Review, experimentally comparing

the most promising approaches, and promoting the development of open benchmark datasets to enable more rigorous evaluation of LLM-based techniques for injection vulnerability detection and testing.

Artifact Availability

All extracted data from the primary studies are presented in Table 1.

Acknowledgments

The authors gratefully acknowledge the PUB-USP program, the PRPI-USP (Grant: 22.1.09345.01.2), and Brazilian agencies for financial support: CAPES and CNPq (Grant: 406941/2025-4, 420025/2023-5).

References

[1] Fatma Ahsan and Faisal Anwer. 2024. A Systematic Literature Review on Software Security Testing Using Metaheuristics. *Automated Software Engineering* 31, 2 (2024). https://doi.org/10.1007/s10515-024-00433-0

[2] Halim Wildan Awalurahman, Ibrahim Hafizhan Witsqa, Indra Kharisma Raharjana, and Ahmad Hoirul Basori. 2023. Security Aspect in Software Testing Perspective: A Systematic Literature Review. *Journal of Information Systems Engineering and Business Intelligence* 9, 1 (2023). https://doi.org/10.20473/jisebi.9.1.88-102

[3] Murat Aydos, Çiğdem Aldan, Evren Coşkun, and Alperen Soydan. 2022. Security Testing of Web Applications: A Systematic Mapping of the Literature. *Journal of King Saud University – Computer and Information Sciences* 34, 9 (2022), 6775–6792. https://doi.org/10.1016/j.jksuci.2021.10.013

[4] Vahid Babaey and Arun Ravindran. 2025. GenSQLi: A Generative Artificial Intelligence Framework for Automatically Securing Web Application Firewalls Against Structured Query Language Injection Attacks. *Future Internet* 17, 1 (2025), 8.

[5] Vahid Babaey and Arun Ravindran. 2025. GenXSS: an AI-Driven Framework for Automated Detection of XSS Attacks in WAFs. *arXiv preprint arXiv:2504.08176* (April 2025). https://doi.org/10.48550/arXiv.2504.08176 arXiv:2504.08176 [cs.CR]

[6] B. Cartaxo, G. Pinto, and S. Soares. 2020. Rapid Reviews in Software Engineering. In *Contemporary Empirical Methods in Software Engineering*, M. Felderer and G. Travassos (Eds.). Springer, Cham. https://doi.org/10.1007/978-3-030-32489-6_13

[7] Antonio de Jesús Domínguez-García, Xavier Limón, Jorge Octavio Ocharán-Hernández, and Juan Carlos Pérez-Arriaga. 2023. Security Testing for Web Applications: A Systematic Literature Review. In *Proceedings of the 11th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE, 82–91. https://doi.org/10.1109/CONISOFT60487.2023.00020

[8] Zhiwen Gui, Enze Wang, Binbin Deng, Mingyuan Zhang, Yitao Chen, Shengfei Wei, Wei Xie, and Baosheng Wang. 2024. SqliGPT: Evaluating and Utilizing Large Language Models for Automated SQL Injection Black-Box Detection. *Applied Sciences* 14, 16 (2024), 6929.

[9] Faisal Jameel, Waqas Ahmad, Maryam Shahpasand, Vahid Heydari Fami Tafreshi, and Mohammad Heydari. 2024. Evaluating Large Language Models Versus Traditional Tools in SQL Injection Exploit Generation. In *2024 Global Congress on Emerging Technologies (GCET)*. IEEE.

[10] Mohd Waris Khan, Dhirendra Pandey, and Suhel Ahmad Khan. 2016. Critical Review on Software Testing: Security Perspective. In *Smart Trends in Information Technology and Computer Communications (Communications in Computer and Information Science, Vol. 628)*. Springer, 714–723. https://doi.org/10.1007/978-981-10-3433-6_68

[11] Daniel Leung, Omar Tsai, Kourosh Hashemi, Bardia Tayebi, and Mohammad A. Tayebi. 2024. XploitSQL: Advancing Adversarial SQL Injection Attack Generation with Language Models and Reinforcement Learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*. ACM.

[12] Francesca Lonetti, Antonia Bertolino, and Felicita Di Giandomenico. 2024. Model-Based Security Testing in IoT Systems: A Rapid Review. *Information and Software Technology* 166 (2024), 107326. https://doi.org/10.1016/j.infsof.2023.107326

[13] Dean Richard McKinnel, Tooska Dargahi, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2019. A Systematic Literature Review and Meta-Analysis on Artificial Intelligence in Penetration Testing and Vulnerability Assessment. *Computers & Electrical Engineering* 75 (2019), 175–188. https://doi.org/10.1016/j.compeleceng.2019.02.022

[14] Abdechakour Mechri, Mohamed Amine Ferrag, and Merouane Debbah. 2025. SecureQwen: Leveraging LLMs for vulnerability detection in python codebases. *Computers & Security* 148 (2025), 104151.

[15] Dennis Miczek, Divyesh Gabbireddy, and Suman Saha. 2025. Leveraging LLM to Strengthen ML-Based Cross-Site Scripting Detection. In *Proceedings of ACM Workshop on Wireless Security and Machine Learning (WiseML 2025)*. ACM.

[16] OWASP Foundation. [n. d.]. Cross Site Scripting (XSS). https://owasp.org/www-community/attacks/xss/. https://owasp.org/www-community/attacks/xss/ OWASP Community. Accessed: July 10, 2026.

[17] OWASP Foundation. [n. d.]. SQL Injection. https://owasp.org/www-community/attacks/SQL_Injection. https://owasp.org/www-community/attacks/SQL_Injection OWASP Community. Accessed: July 10, 2026.

[18] OWASP Foundation. 2025. *OWASP Top 10:2025.* https://owasp.org/Top10/2025/

[19] Rodrigo Pedro, Miguel E. Coimbra, Daniel Castro, Paulo Carreira, and Nuno Santos. 2025. Prompt-to-SQL Injections in LLM-Integrated Web Applications: Risks and Defenses. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE.

[20] Jakub Pružinec and Nguyen Anh Quynh. 2023. Hakuin: Optimizing Blind SQL Injection with Probabilistic Language Models. In *2023 IEEE Security and Privacy Workshops (SPW)*. IEEE.

[21] Katia Romero Felizardo Scannavino, Elisa Yumi Nakagawa, Sandra Camargo Pinto Ferraz Fabbri, and Fabiano Cutigi Ferrari. 2017. Revisão Sistemática da Literatura em Engenharia de Software: teoria e prática. (2017).

[22] Suman and Raees Ahmad Khan. 2024. Survey on Identification and Prediction of Security Threats Using Various Deep Learning Models on Software Testing. *Multimedia Tools and Applications* (2024). https://doi.org/10.1007/s11042-024-18539-2

[23] Karl Tamberg and Hayretdin Bahsi. 2024. Harnessing Large Language Models for Software Vulnerability Detection: A Comprehensive Benchmarking Study. *arXiv preprint arXiv:2405.15614* (2024).

[24] Zonghang Tian, Yingying Zhang, Xingkai Cheng, Shuting Lou, and Zhengdan Jiang. 2024. SQL injection vulnerability detection based on Pissa-tuned Llama 3 large language model. In *2024 6th International Conference on Frontier Technologies of Information and Computer (ICFTIC)*. IEEE.

[25] Tinghui Yang, Yongjun Wang, and Zhiyuan Jiang. 2024. LLMSQLi: A Black-Box Web SQLi Detection Tool Based on Large Language Model. In *2024 5th International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*. IEEE.

[26] Jian Zhang, Chong Wang, Anran Li, Weisong Sun, Cen Zhang, Wei Ma, and Yang Liu. 2024. An Empirical Study of Automated Vulnerability Localization with Large Language Models. *arXiv preprint arXiv:2404.00287* (2024).

[27] Yingying Zhang, Zhengdan Jiang, Xingkai Cheng, Han Wu, Zonghang Tian, and Shuting Lou. 2024. A Method of SQL Injection Attack Detection Based on Large Language Models. In *2024 2nd International Conference on Computer Network Technology and Electronic and Information Engineering (CNTEIE)*. IEEE.

[28] Yuan Zhou, Enze Wang, Wantong Yang, Wenlin Ge, Siyi Yang, Yibo Zhang, Wei Qu, and Wei Xie. 2025. XSShield: Defending Against Stored XSS Attacks Using LLM-Based Semantic Understanding. *Applied Sciences* 15, 6 (2025), 3348.