GNN-Based Detection of XSS Vulnerabilities to Strengthen Security for Financial Web Transactions: A Web Browser Extension Approach

Abdelkader TAJTIT [1]   Mohammed SERRHINI [1]

[1]Computer Science Research Laboratory, University Mohammed First Oujda Morocco

Problem Statement

The rapid growth of digital financial platforms in Africa has improved service access but also increased exposure to cyberattacks like Cross-Site Scripting (XSS). XSS allows attackers to inject malicious JavaScript into trusted websites, posing risks such as data theft and unauthorized actions. Despite existing security tools, detecting and preventing XSS remains challenging due to sophisticated obfuscation techniques and the limitations of traditional detection methods.

Current solutions often lack real-time detection and fail to provide clear, user-friendly alerts, especially for non-experts. To address these issues, we propose a novel Firefox extension that integrates a Graph Neural Network (GNN) for detecting obfuscated XSS vulnerabilities and a chatbot assistant powered by a Large Language Model (LLM) to provide understandable explanations of threats, improving both detection accuracy and user awareness.

Research Goals

The main objectives of this research are to:

- Develop a Firefox browser extension for the real-time detection of Cross-Site Scripting (XSS) attacks in web applications.
- Use Graph Neural Networks (GNNs) to classify JavaScript code as benign or malicious by transforming it into Control Flow Graphs (CFGs).
- Generate a diverse dataset of realistic, obfuscated XSS samples using a fine-tuned Large Language Model (LLM).
- Evaluate the performance of the detection system on real-world African financial websites.
- Incorporate a Large Language Model (LLM)-based chatbot assistant that provides user-friendly explanations of detected vulnerabilities to enhance user understanding of security risks.

System Overview

The system is a Firefox browser extension designed to detect Cross-Site Scripting (XSS) attacks in real-time on financial web applications. It integrates a Graph Neural Network (GNN) model that processes JavaScript code, transforming it into Control Flow Graphs (CFGs) to identify malicious patterns. The extension alerts users when suspicious behavior is detected and uses a chatbot powered by a Large Language Model (LLM) to provide clear explanations of the detected threat.

Key components:

- **Code Extraction:** Scans and extracts JavaScript code from web pages.
- **GNN Detection:** Analyzes the code through CFGs to detect malicious scripts.
- **Real-Time Alerts:** Provides immediate warnings and highlights dangerous code.
- **Chatbot Assistant:** Offers simple, understandable explanations of detected vulnerabilities.

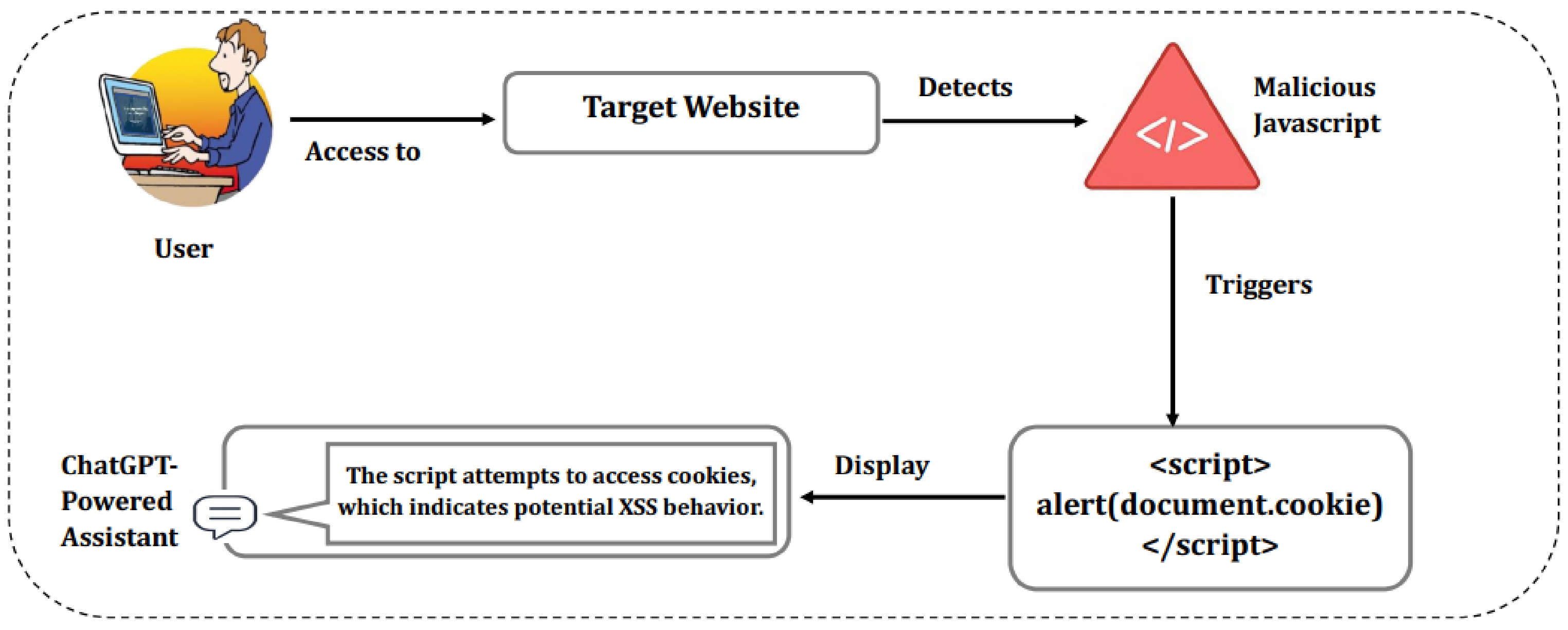


Figure 1. System architecture: JavaScript extraction, GNN detection, alerts, and chatbot.

Experimental Implementation

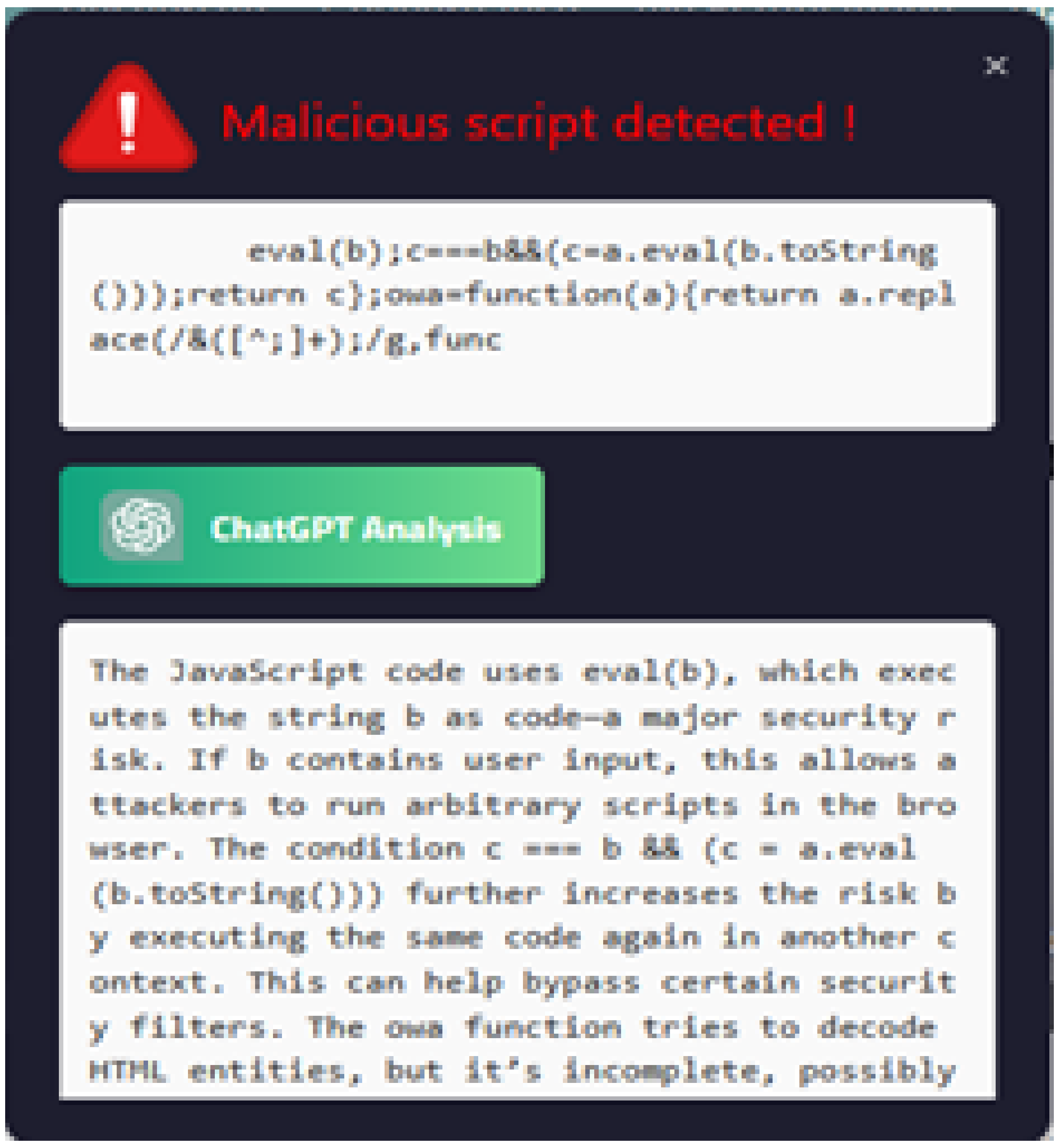
We implemented a functional Firefox extension and trained the GNN model using a curated dataset consisting of 37,605 samples, including 12,038 benign and 7,321 malicious scripts. The dataset was preprocessed to remove duplicates, normalize data, and ensure quality. Each script was converted into a Control Flow Graph (CFG) to capture the structural properties of the code.

Dataset	Benign code	XSS payload	Total
Kaggle	6 313	7 373	13 686
JS library source code	11 120	-	11 120
XSS Cheat Sheet	-	6 047	6 047
Materialize JS library	6 752	-	6 752
Total	**24 185**	**13 420**	**37 605**

Figure 2. Dataset composition, showing the distribution of benign and malicious scripts used for training.

Experimental Implementation

The extension was tested on real-world financial websites, such as online banking and digital wallets in Africa, as well as synthetic data generated using controlled language models. The GNN model was trained on 80% of the dataset and evaluated on the remaining 20% to assess its performance in detecting both obfuscated and non-obfuscated XSS attacks.
Upon detecting suspicious behavior, the system triggers two main mechanisms:
- Real-time alert system: The extension displays a warning to the user, clearly highlighting the presence of a potentially malicious script and identifying its source within the web page.
- ChatGPT-powered assistant: A dialog interface is launched, providing an easy-to-understand explanation of the detected script's behavior to increase user awareness, even for those without cybersecurity expertise.



Malicious script detected !

```
        eval(b);c===b&&(c=a.eval(b.toString
()));return c};owa=function(a){return a.repl
ace(/&([^;]+);/g,func
```

The JavaScript code uses eval(b), which executes the string b as code—a major security risk. If b contains user input, this allows attackers to run arbitrary scripts in the browser. The condition c === b && (c = a.eval (b.toString())) further increases the risk by executing the same code again in another context. This can help bypass certain security filters. The owa function tries to decode HTML entities, but it's incomplete, possibly

Figure 3. ChatGPT Analysis.

Future Work

In future work, we aim to:

- **Broaden Browser Support:** Extend the extension to other browsers like Chrome and Edge to reach a wider user base.
- **Extend Language Support:** Adapt the model to detect vulnerabilities in other languages commonly used in web applications, such as PHP and Python.
- **Adversarial Training:** Use reinforcement learning to generate evasive XSS payloads, helping to further improve the model's robustness against new attack strategies.
- **Multi-Attack Detection:** Expand the detection capabilities to cover other web vulnerabilities such as SQL injection (SQLi), Cross-Site Request Forgery (CSRF), and malware.
- **Improve User Experience:** Enhance the chatbot assistant to provide more detailed security guidance and threat mitigation recommendations.

Conclusion

We proposed and implemented a Firefox browser extension aimed at detecting and explaining malicious JavaScript code, with a particular focus on XSS attacks. Our approach combines AI-based detection mechanisms with a natural language assistant, providing both technical protection and user-friendly explanations. Experimental testing on real-world financial websites demonstrated the feasibility and effectiveness of our solution.

References

[1] Kaspersky Lab, "14% increase in spyware attacks on African businesses: Kaspersky presents a cyberthreat landscape report at Gitex Africa in Morocco," 2025. Check Point Software, "Africa sees 37% surge in cyber attacks," 2025.

Technext24, "10 Nigerian startups to watch in 2025," Apr. 11, 2025.

Nagarjun, P. M. D., and Ahamad, S. S. "Cross-site scripting research: A review." IJACSA, 2020.

SiteGuarding, "Top 10 website security threats in 2025 and how to protect against them," 2025. [6] Vigilance Security Magazine, "90% of XSS web vulnerabilities still fool advanced IT experts," 2025. Available: https:// shorturl.at/np705

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun. Graph neural networks: A review of methods and applications. AI Open, 1:57–81, 2020. ISSN 2666-6510. https://doi.org/10.1016/j.aiopen.2021.01.001.

B. Khemani, S. Patil, K. Kotecha, et al. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. J Big Data, 11:18, 2024. https://doi.org/10.1186/s40537-023-00876-4.

K. Sendjaja, S. A. Rukmono, and R. S. Perdana. Evaluating control-flow graph similarity for grading programming exercises. In 2021 International Conference on Data and Software Engineering (ICoDSE), Bandung, Indonesia, 2021, pp. 1–6. doi: 10.1109/ICoDSE53690.2021.9648464.

M. A. K. Raiaan et al. A review on large language models: Architectures, applications, taxonomies, open issues and challenges. IEEE Access, 12:26839–26874, 2024. doi: 10.1109/ACCESS.2024.3365742.

Deep Learning Indaba 2025, Kigali-Rwanda