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

Abdelkader Tajtit
University Mohammed First Oujda Morocco
Avenue Mohamed VI, Oujda, Morocco
abdelkader.tajtit.d23@ump.ac.ma

Mohammed Serrhini
University Mohammed First Oujda Morocco
Avenue Mohamed VI, Oujda, Morocco
serrhini@gmail.com

Abstract

The rise of digital financial platforms in Africa has improved access to services but also increased the risk of cyberattacks like Cross-Site Scripting (XSS). XSS attacks inject dangerous JavaScript code into websites, which can steal user data or cause other harm. To help protect users, we developed a Firefox extension that detects malicious scripts in real time. This extension uses a Graph Neural Network (GNN), a type of AI model we have already trained on Control Flow Graphs (CFGs), to find hidden and complex malicious code. The extension shows a clear alert when it detects suspicious code and highlights the dangerous part. It also includes a chatbot assistant based on ChatGPT that explains the code's behavior in simple words. We tested the extension on real African financial websites and with sample data, and it showed good results. This tool combines strong AI detection with easy explanations to improve online safety and user awareness.

1. Introduction

The rapid expansion of digital financial services and e-commerce in Africa has elevated the region's economic growth, but it has also exposed users and organizations to escalating cyber threats. In 2024 alone, the African region experienced over 131.6 million web-based threat detections, including phishing, malware, and exploit attempts, marking a 1.2% increase compared to 2023 [1]. Notably, Kenya, South Africa, and Morocco ranked highest, with nearly 20 million, 17 million, and 12.6 million web threat attempts respectively [1].

On average, African organizations faced 2,960 cyberattack attempts per week in Q2 2024, representing a 37% increase year on year—the highest global rate during that period [2] (see Figure 2).

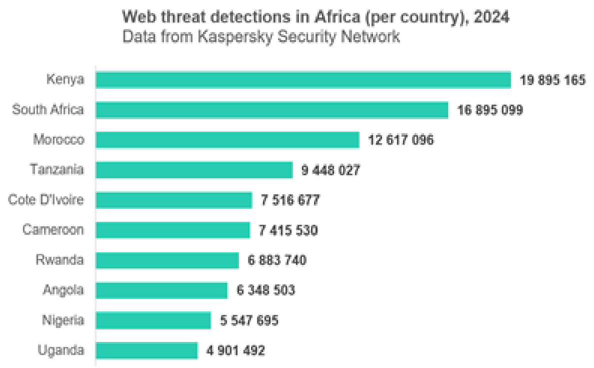
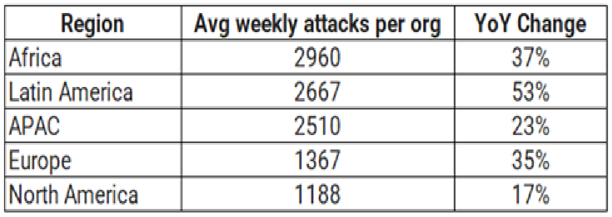


Figure 1. Web threat detection in Africa (2024).



Region	Avg weekly attacks per org	YoY Change
Africa	2960	37%
Latin America	2667	53%
APAC	2510	23%
Europe	1367	35%
North America	1188	17%

Figure 2. Regional Analysis of Cyber Attacks.

Furthermore, cybercriminal tactics targeting data theft also surged: spyware attacks rose by 14%, while password-stealer malware detections surged by 26% between 2023 and 2024 [3]. Among the most persistent web-based threats is Cross Site Scripting (XSS)[4] a type of vulnerability that allows malicious JavaScript injection, posing severe risks to confidential data and financial integrity particularly on user-centric web applications. According to OWASP, XSS remains one of the top 3 vulnerabilities globally, accounting for approximately 30% of web application flaws, and has been exploited in over 76% of organizations as of 2023 [5]. Despite familiarity with XSS, up to 90% of vulnerabilities still evade detection even by advanced IT professionals [6]. Against this backdrop, web-based financial platforms which often handle sensitive user credentials and transaction data are especially vulnerable. Existing security tools often lack real-time, client-side detection and do not provide user-friendly explanations. In response, this paper presents a Firefox browser extension that performs real-time detection of malicious JavaScript using a machine learning model we have already developed. This model is based on a Graph Neural Network (GNN)[7][8] trained on control flow graph (CFG)[9] representations of obfuscated scripts to accurately identify complex threats. When suspicious behavior is detected, the extension automatically generates an alert containing the extracted malicious code snippet. In addition, the extension includes a conversational assistant powered by a Large Language Model (LLM)[10], enabling users to ask questions, receive semantic explanations, and gain contextual security guidance about the detected threat. This combined approach integrating a developed GNN-based detection model with natural language explanations improves both accuracy and usability, offering a practical defense solution for financial web applications in Africa's growing digital economy.

2. System Overview

This system is a browser extension for Firefox that helps users detect and understand dangerous JavaScript code on websites, especially code that can cause Cross-Site Scripting (XSS) attacks. We have already developed and trained the detection model using Graph Neural Networks (GNNs), and this trained model is now integrated into the Firefox extension. The system also uses Large Language Models (LLMs) like ChatGPT to explain detected threats in simple language. At the center of the system is a code extraction module. This part of the extension scans the web page and collects all JavaScript code. Then, the code is sent to two components:

- The GNN detection engine: It converts the code into a graph and uses the trained model to decide if the code is safe or dangerous.
- The ChatGPT assistant: If the system finds dangerous code, it highlights it and activates a chatbot. The user can ask questions like "Why is this code dangerous?" and get clear answers in real time.

One important feature of the system is its focus on the user. Instead of just blocking code or showing confusing alerts, it explains the risk clearly. This helps users understand what is happening and learn about web security. Here are some real examples of how this system works:

- Scenario 1: Protecting a Bank Account Page

  Figure 7 illustrates a step-by-step cyberattack using Cross-Site Scripting (XSS) against an online banking website. The target is a bank in Africa that provides digital services such as account balance checks, fund transfers, and bill payments. The attack begins when a cybercriminal creates malicious JavaScript code (Step 1) and sets up a separate "drop site" at xxx.com to collect stolen data (Step 2). Exploiting a security flaw in the bank's website (bank.com), the attacker injects the harmful code into its pages (Step 3). When a customer named Ali visits the bank's site (Step 4), the page loads the malicious script, which runs silently in his browser (Step 5). Without his knowledge, the script sends sensitive information, such as session cookies, from bank.com to the attacker's server (Step 6). Without protection, this could allow the attacker to take control of Ali's account, transfer money, or misuse his personal data.

  In this case, a browser extension detects the malicious JavaScript, highlights the dangerous code, and warns Ali. The ChatGPT assistant explains that the script is trying to send private data to an untrusted server. Understanding the risk, Ali avoids logging in and remains safe. This scenario shows the importance of secure coding, regular website security audits, and user awareness, especially for online banks in Africa where digital banking adoption is growing but some systems still lack modern security protections.

- Scenario 2: Unsafe African Government or Finance Website
  Figure 8 illustrates a step-by-step cyberattack using Cross-Site Scripting (XSS) against an official African government or finance website. The target is a site where users can submit documents or manage mobile money accounts. The attack starts when a cybercriminal crafts malicious JavaScript code that steals session cookies and sets up a remote server to collect this data. Exploiting a vulnerability in the website, the attacker injects the harmful script into a user-editable form or profile field. When a legitimate user visits the page, the embedded script runs silently in their browser. Without the user's knowledge, the script transmits the session cookie (PHPSESSID=...) to the attacker's server, which can then hijack the user's session and perform unauthorized actions.

This scenario highlights the critical need for secure coding practices, vigilant monitoring of user inputs, and user awareness tools especially on government and finance websites in Africa, where digital services are expanding but security measures can lag behind evolving threats.

This example shows why such tools are important in regions where websites may not follow modern security practices. The system helps protect users and also teaches them about security threats.

Figure 3 illustrates the architecture of the system, highlighting the interaction between the extraction module, ChatGPT assistant and the user interface components.

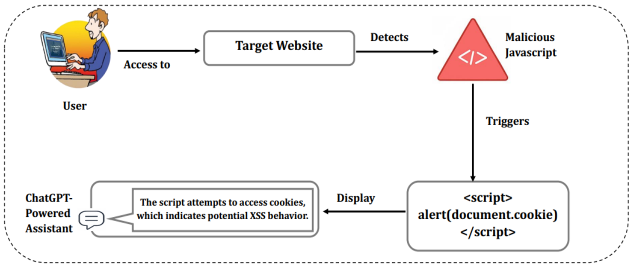
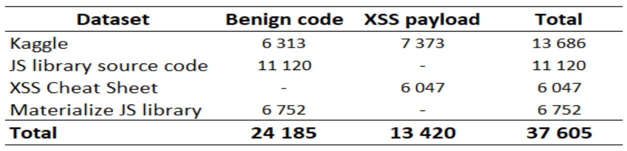


Figure 3. Architecture of the system.

3. Experimental Implementation

To validate our proposed architecture, we developed a functional Firefox browser extension integrating a Graph Neural Network (GNN) model for real-time detection of malicious JavaScript code, particularly Cross-Site Scripting (XSS) attacks. The model was trained on a curated dataset initially containing 37,605 samples, sourced from Kaggle, the OWASP XSS Cheat Sheet, and common JavaScript libraries. After preprocessing (deduplication, normalization, and quality filtering), the final dataset consisted of 19,359 samples, including 12,038 benign (62.18%) and 7,321 malicious (37.82%) scripts. Figure 4 presents the composition and origin of the collected data.



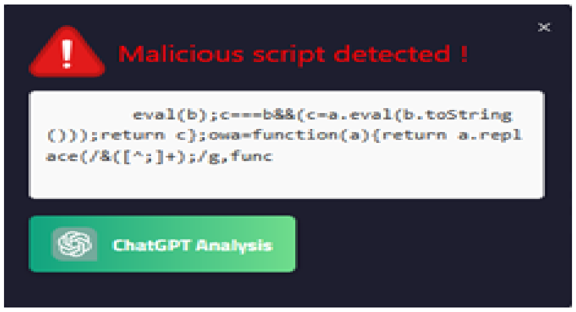
Dataset	Benign code	XSS payload	Total
Kaggle	6 313	7 373	13 686
JS library source code	11 120	-	11 120
XSS Cheat Sheet	-	6 047	6 047
Materialize JS library	6 752	-	6 752
Total	**24 185**	**13 420**	**37 605**

Figure 4. Sources and composition of the collected dataset.

To better simulate real-world scenarios, we enriched the dataset with synthetic obfuscated XSS payloads generated using a controlled language model. Each script real or synthetic was transformed into a Control Flow Graph (CFG), allowing the model to capture structural and semantic properties of the code. The dataset was split into 80% training and 20% testing sets, and the encoded CFGs were used to train the GNN classifier embedded in the extension.

Upon detecting suspicious behavior, the system triggers two main mechanisms:

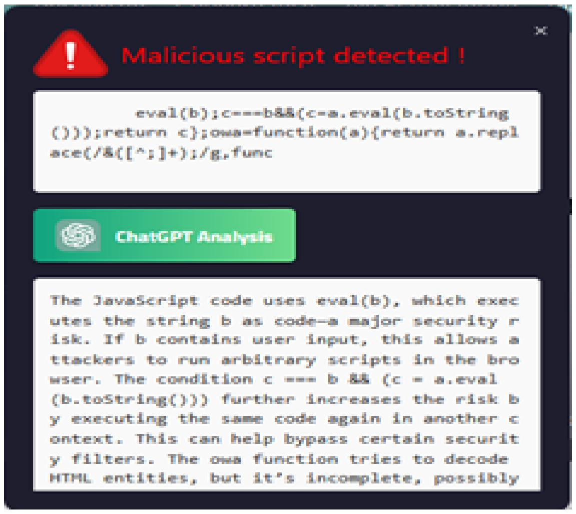
- Real-time alert system: The extension displays a warning to the user, clearly highlighting the presence of a potentially malicious script and identifying its source within the web page (see Fig 5).



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

Figure 5. Real-time Alert.

- 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 (see Fig 6).



```
     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 exec
utes the string b as code—a major security r
isk. If b contains user input, this allows a
ttackers to run arbitrary scripts in the bro
wser. The condition c === b && (c = a.eval
(b.toString())) further increases the risk b
y executing the same code again in another c
ontext. This can help bypass certain securit
y filters. The owa function tries to decode
HTML entities, but it's incomplete, possibly
```

Figure 6. ChatGPT Analysis.

Additionally, we conducted experiments on real-world websites, including publicly accessible pages of financial platforms and banking institutions operating in Africa. These tests covered online services such as money transfers, digital wallets, and user authentication forms. The goal was to evaluate the extension's

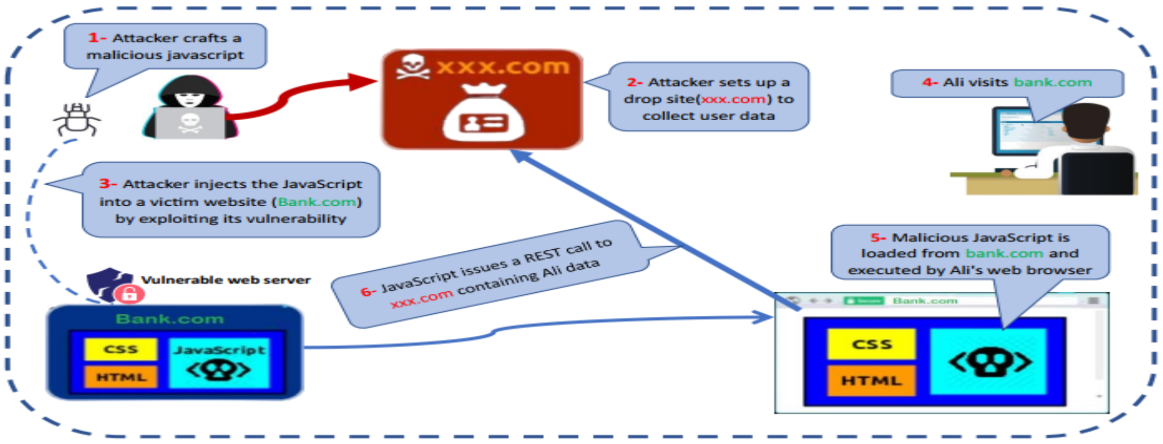

1- Attacker crafts a malicious javascript
xxx.com
2- Attacker sets up a drop site(xxx.com) to collect user data
4- Ali visits bank.com
3- Attacker injects the JavaScript into a victim website (Bank.com) by exploiting its vulnerability
5- Malicious JavaScript is loaded from bank.com and executed by Ali's web browser
Vulnerable web server
6- JavaScript issues a REST call to xxx.com containing Ali data
Bank.com
CSS
HTML
JavaScript
Bank.com
CSS
HTML

Figure 7. Scenario of Bank Account Page Attack.

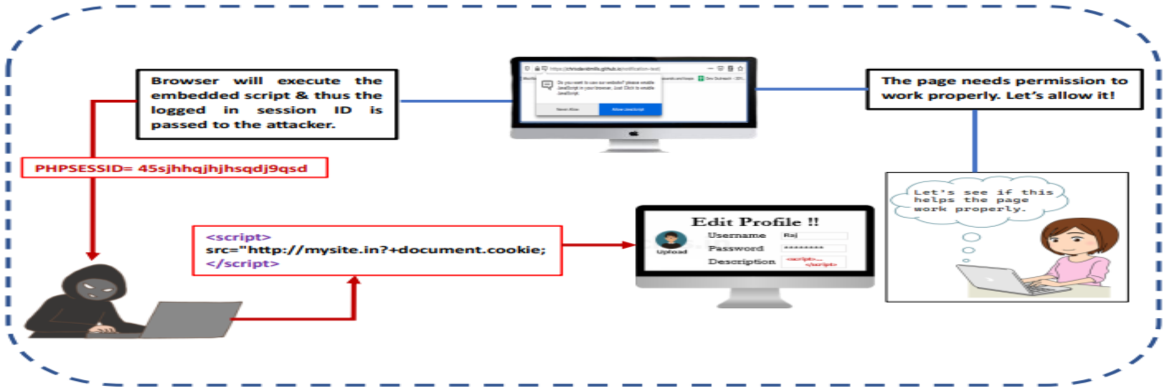

Browser will execute the embedded script & thus the logged in session ID is passed to the attacker.
The page needs permission to work properly. Let's allow it!
PHPSESSID= 45sjhhqjhjhsqdj9qsd
<script> src="http://mysite.in?+document.cookie; </script>
Edit Profile !!
Username
Raj
Password
Description
Upload
Let's see if this helps the page work properly.

Figure 8. Scenario of Unsafe African Government or Finance Website.

performance in high-risk, real-use scenarios typical of the African fintech sector. Initial results showed that integrating an AI model into a browser extension is technically feasible and effective for real-time detection of malicious client-side scripts. The combination of automatic detection with natural language explanations provided by ChatGPT makes the tool especially useful in low-digital-literacy environments, where users may not fully understand security alerts. This experimental implementation demonstrates the practical potential of AI-driven browser security tools in real-world applications, especially in regions where digital finance is growing but user-side protections remain limited.

4. Conclusion

In this work, we proposed and implemented a novel Firefox browser extension aimed at detecting and explaining malicious JavaScript code, with a particular focus on Cross-Site Scripting (XSS) attacks. Our approach combines AI-based detection mechanisms with a natural language assistant powered by ChatGPT, providing both technical protection and user-friendly explanations. The system architecture was designed for real-time operation, seamlessly integrated within the browser environment. Experimental testing on real-world financial websites, including African banking platforms, demonstrated the feasibility and effectiveness of our solution. The extension accurately identified suspicious scripts and alerted users, while also offering valuable contextual insights to promote user awareness and trust. Our work shows the potential of combining machine learning and large language models (LLMs) to address client-side web security challenges in a transparent and explainable way. This is especially important in emerging digital ecosystems like those across Africa, where fintech adoption is growing fast but often lacks adequate end-user protection tools.

References

[1] Kaspersky Lab, “14% increase in spyware attacks on African businesses: Kaspersky presents a cyberthreat landscape report at Gitex Africa in Morocco,” 2025. Available: https://shorturl.at/sJUQd

[2] Check Point Software, “Africa sees 37% surge in cyber attacks,” 2025. Available: https://shorturl.at/FhAlb

[3] Technext24, “10 Nigerian startups to watch in 2025,” Apr. 11, 2025. Available: https://shorturl.at/x072Y

[4] P. M. D. Nagarjun and S. Shakeel Ahamad. Cross-site scripting research: A review. International Journal of Advanced Computer Science and Applications (IJACSA), 11(4), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110481.

[5] SiteGuarding, “Top 10 website security threats in 2025 and how to protect against them,” 2025. Available: https://shorturl.at/HP36i

[6] Vigilance Security Magazine, “90% of XSS web vulnerabilities still fool advanced IT experts,” 2025. Available: https://shorturl.at/np705

[7] 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.

[8] 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.

[9] 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.

[10] 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.