Journal of Applied Sciences, Information and Computing
Volume 6, Issue 1, April- May 2025
School of Mathematics and Computing, Kampala International University

ISSN: 1813-3509 https://doi.org/10.59568/JASIC-2025-6-1-17

An enhanced technique for the detection and securing of a Potential vulnerable website

[1]Shukurah Oluwadamilola Yusuf and [2]Adeleke Raheem Ajiboye
[1] Department of Computer Science, Faculty of Communication and Information Sciences, University of Ilorin, Ilorin, Nigeria
[2] Department of Computer Science, School of Mathematics and Computing, Kampala International University, Uganda

Abstract
Enhancing the web application security is very crucial for every organization, but ensuring accurate implementation of vulnerability features can be challenging. Traditional vulnerability scanners struggle due to limited security education for programmers, who often rely solely on web searches for information. However, the use of an innovative techniques such as a dedicated rule engine scanner and a user interface that implements precise vulnerability can be a good guide for programmers. This study focuses on accessing the http security headers in web applications. The detection is actually possible through the security guidelines activated to achieve this purpose. This study also emphasizes the level of accuracy as regards the security header implementation. The resulting output of this study reveals the missing features which was validated using an established scanners and qualitative assessment metrics. Findings from this study reveals some positive outcomes and informed content policies as regards security, features and permissions. The study further reveals better approaches to secure a website from becoming vulnerable, by ensuring the server configurations are regularly checked and hardened the web page through the use of SSL/TLS with a valid certificate.

Keywords: Rule Engine, Security Education, Educational Vulnerability Scanner, Content Security Policy, Feature Policy, Permissions Policy.

1. Introduction
The digital landscape in this era has projected the use of websites as inevitable for businesses, organizations and individual. This has really helped to connect with users for improved services regardless of location. However, the proliferation of the internet, the threat of cyberattacks which are basically targeting vulnerable websites is increasing in multiples. Hackers constantly exploit security weaknesses, leading to data breaches, financial losses, and reputational damage. As cyber threats evolve, traditional security measures are often insufficient to protect websites from sophisticated attacks. This has led to the need for more advanced and proactive techniques to detect vulnerabilities and secure web platforms effectively.

This paper presents an enhanced technique for the detection and securing of vulnerable websites, focusing on innovative approaches to identify security flaws, prevent unauthorized access, and ensure a robust defense against potential cyber

threats. Through an integrated system of real-time vulnerability scanning, behavior analysis, and advanced encryption methods, this technique aims to provide website owners with a comprehensive solution to safeguard their digital assets and maintain user trust.

The study highlights the limitations of traditional vulnerability scanning approaches in providing comprehensive solutions and educating developers about security practices. It emphasizes the lack of active participation in educating developers and the growing prevalence of software, resulting in security vulnerabilities and cyber-attacks. The project aims to tackle these issues by implementing rule engines within web application vulnerability scanners, particularly hard-constraint rule engines.

The study aims to address the broader issue of inadequate security education among developers. It seeks to improve vulnerability scanners' reporting and remediation features by utilizing rule engines, specifically hard-constraint rule engines, to offer tailored and easily understandable security education for developers involved in vulnerability scanning.

The study aims to design an innovative rule-engine-based framework, implement and optimize the created system, and evaluate its efficiency compared to non-educational models to improve the security of vulnerable websites. The study's significance lies in its potential to bring about a new technological advancement and innovation period, surpassing the constraints of traditional vulnerability scanning methods and enhancing security measures for web applications.

The research also emphasizes adaptability, security education, prompt and trustworthy responses. It aims to impact various sectors, including cybersecurity, software development, compliance and regulations, and web applications. The study's scope includes investigating different aspects of enhancing web application vulnerability scanning using rule-engine techniques, particularly emphasizing hard-constraint rule engines. It also covers ethical concerns, potential prejudices, and fairness in educational vulnerability scanning procedures.

In summary, this study aims to carry out some transformations in the field of vulnerability scanning which involve establishing a fresh benchmark for education, adaptability, and availability, emphasizing sustainable approaches to technology advancement. We structured the rest of this paper as follows: In Section 2, the discussion of how website become vulnerable is presented, the section is followed by a discussion of works related to this study. Section 4 shows the methods proposed for this study and the material used is also presented in this section. The findings from this study are shown and discussed in Section 5. This study is concluded in Section 6

.

2. How websites become vulnerable

A vulnerable website is associated with the following vulnerable features:

i.SQL Injection*:* This occurs when an attacker attempts to insert malicious SQL queries into a website's input fields such as search boxes or login forms, which are then executed by the database, potentially revealing sensitive information or corrupting the database.

ii. ***Cross-Site Scripting (XSS)****:* This happens when an attacker injects malicious scripts into web pages viewed by other users. These scripts can steal cookies, session tokens, or other sensitive data.

iii. ***Cross-Site Request Forgery (CSRF)****:* This attack tricks users into performing actions they didn't intend to, like transferring funds or changing account details, by sending unauthorized requests from their unauthorized authenticated session.

iv.*Remote Code Execution (RCE)**:*** This vulnerability allows attackers to execute arbitrary code on the server, which could lead to full server control, data exfiltration, or the installation of malware.

v.*Insecure Authentication and Session Management*: Weak authentication mechanisms, such as weak passwords, or poor session management, like not properly invalidating user sessions, can allow attackers to hijack accounts and impersonate users.

vi. ***Outdated Software or Patches****:* Websites using outdated software, content management

systems (CMS), plugins, or libraries may have known vulnerabilities that can be exploited if not patched or updated.

vii. ***Insecure Communications:*** Websites that don't use secure HTTP which runs on port number 443 is exposes the website or make it vulnerable. This web protocol encrypt data between the server and the user's browser are susceptible to interception or man-in-the-middle attacks.

3 Related Works

Several studies have addressed the critical issue of identifying and mitigating vulnerabilities in web application security. Baako and Umar (2020) conducted a study on the security of Ghanaian e-commerce websites, revealing significant vulnerabilities that could jeopardize user data security. Thus, they advocated for policy changes to strengthen website security. (Abdulghaffar, Elmrabit et al. 2023), introduced a pioneering framework for automating vulnerability testing using multiple scanners, showing improved accuracy compared to individual scanners. Similarly, (Kapodistria, Mitropoulos et al. 2011) proposed a web security tool with high success rates in identifying a wide range of attack types, contributing to the web security arsenal. Machine learning in a combination with some popular web vulnerability scanner has proved to be efficient in bolstering web security (Zangana 2024).

Also, while discussing the security risks for Malaysian small businesses operating online (Buja, Low et al. 2024) , emphasized the need for improved web security practices in this sector. (Fadlil, Riadi et al. 2024) delved into examining SQL injection attacks and suggested the adoption of firewalls as a defense mechanism. Also in a creative approach, the study reported in (Xia, Guo et al. 2024) aimed to fortify browser-based cryptocurrency wallets, attaining a high success rate in uncovering vulnerabilities across a large sample. These wallets, like MetaMask, MyEtherWallet, and others, are prime targets for attackers due to their popularity and exposure to the internet.

The study reported in (Lala, Kumar et al. 2021) provided a different perspective, identifying common vulnerabilities developers should address to protect their applications, with a strong emphasis on OWASP's top 10 vulnerabilities. In a distinct context, (Lathifah, Amri et al. 2022) examined security weaknesses in Sharia crowd-funding websites, proposing enhancements to safeguard user data and financial information. Similarly, (Gandikota, Valluri et al. 2023), stressed the urgent need for robust security measures in web applications and the value of the OWASP framework in identifying and classifying vulnerabilities. In further studies, (Priyanka and Smruthi 2020) explored web application security vulnerabilities, offering preventative measures for developers to mitigate common attacks.

In addition, (Ula, Adek et al. 2023) investigated the security vulnerabilities in e-commerce platforms, highlighting the need for prioritizing security testing during development. Furthermore, Kasmawi et al. (2023) delved into cybersecurity vulnerabilities in college websites, advocating for improved website security practices and emphasizing the necessity of ongoing vulnerability scanning. Meanwhile, (Mushlihudin and Faisal 2023) focused on fortifying web archive security through continuous vulnerability testing using the OWASP method, thereby reducing overall risk. (Setiawan and Setiyadi 2018) emphasized the importance of a holistic approach to website data security, encompassing website, server, and network security.

The use of scanner ++ reported in (Yin, Xu et al. 2023), was a collaborative framework to enhance vulnerability detection in web applications. The study highlight the potential of collaboration in vulnerability detection. (Srivastava, Raghuvanshi et al. 2023), underscored the criticality of security in e-commerce websites, prioritizing customer and website data protection. They also provided a roadmap for securing e-commerce websites using established practices and tools

In a different approach, (Sitohang, Asnar et al. 2024), focused on improving web application security testing, proposing mutation testing as a valuable tool, particularly in detecting Cross-Site Request Forgery (CSRF) vulnerabilities. It can be

inferred from the review work that the literature has provides valuable insights into the evolving landscape of web application security, showcasing a range of innovative approaches, sector-specific considerations, and the critical need for proactive and collaborative strategies to identify and mitigate vulnerabilities.

4 Material and Methods

In this section, we describe the process of integrating a rule engine into a vulnerability scanner. The focus is on utilizing rule engine methods to strengthen vulnerability scanning, reduce reliance on web searches, and improve overall system effectiveness. Other tasks performed in this section includes: data preparation, software design, information gathering, performance evaluation, and features integration. The system analysis is further broken into a number of some subsections in the following orders:

4.1 System Analysis

i. Existing systems

The existing vulnerability scanning methods have shown improvement in effectiveness, but they still face challenges in adaptability and security education. Collaborative efforts from different fields are required to create more beginner-friendly and educational solutions**.**

ii. Proposed system framework

The proposed system represented in Figure 1, combines hard-constraint rule engines with user interface methods to transform vulnerability scanning. It provides advanced security education through a rule engine and prioritizes user experience with intuitive interfaces, setting a new benchmark for security education and ease of use for beginners.

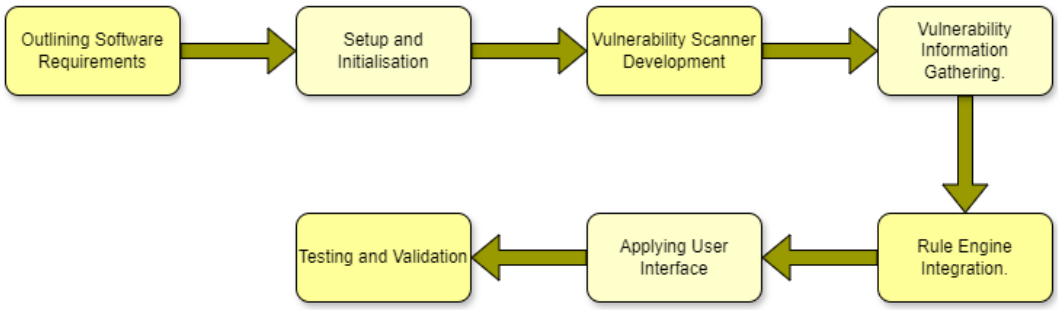

Figure 1. The framework of the proposed system

In the framework of the proposed system, equation is formulated to represent the relationship between the vulnerability detection rate, the mitigation success rate, and the overall website security level:

$S = f(V,M,T)$ (1)

where:

S is the overall security level of the website.

V is the vulnerability detection rate (how effectively vulnerabilities are detected).

M s the mitigation success rate (how well security measures prevent or neutralize identified threats).

T is the time or adaptability factor (the time it takes to detect and secure the website, or how adaptable the technique is to emerging threats).

This equation describes how the security of a website depend on both how well vulnerabilities are identified (V) and how effectively the threats are mitigated (M). The time factor (T) can play a role in determining how quickly the website adapts to new vulnerabilities or emerging attack techniques.

As for the vulnerability detection rate (V), this this could be modelled using a detection rate, say D(t), which changes over time t, representing how quickly vulnerabilities are detected. This might be modelled with an exponential growth function, assuming the detection algorithm improves over time as shown in equation 2.

$$D(t) = D_0(1 - e^{-kt}) \quad (2)$$

https://doi.org/10.59568/JASIC-2025-6-1-17

where
D_0 = maximum detection rate
k = constant rate of improvement
t = time.

The mitigation success rate (M) could be modelled based on the percentage of succesfully mitigated vulnerabilities over time, t. This often involves a success factor that decreases as more vulnerabilities are deteected as shown in equation 3.

$$M(t) = \frac{V_{mitigated}(t)}{V_{total}\,(t)} \text{ x } 100 \qquad (3)$$

where
M(t) = Mitigation Success Rate at time, t (in percentage)
$V_{mitigated}$ (t) is the number of vulnerabilities mitigated by time, t
V_{total} (t) is the total number of vulnerabilities identified at time, t

4.2 System development tools and procedures

The vulnerability scanner system was developed using a range of tools and technologies to ensure the system perform effectively. The development is simplified in compliance with security protocols. The essential tools and technologies utilized include Python, Django, SQLite, Bootstrap, and OWASP.

The procedures basically involve setting up the system's development area, obtaining necessary scripts and resources, and verifying the installation of required libraries and other necessary packages using a package manager like pip for installation. Also, the requests library and urllib play important roles in the educational vulnerability scanner project. The requests library simplifies the HTTP request process and supports complex scenarios such as session management and authentication. On the other hand, urllib, as part of the Python standard library, provides essential capabilities for making HTTP requests and handling of URLs. The vulnerability scanner methodology also involves structured header scanning functionality, which was implemented in the scan-headers function using the requests library.

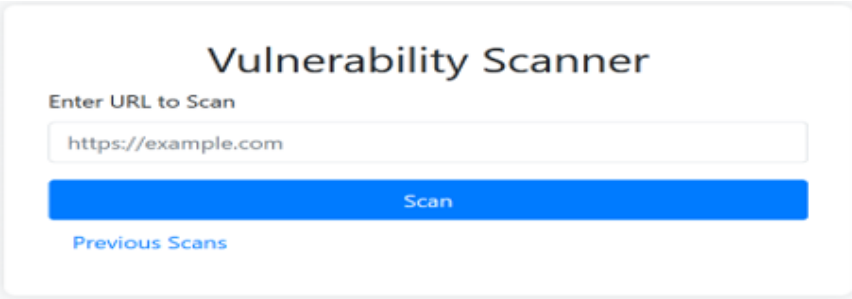

Figure 2. A view of the input section

4.3 The Use Case Diagram

The use case diagram as shown in Figure 3, illustrates how users interact with the vulnerability scanner system to scan a URL, review scan history, and schedule regular scans for specific URLs. It depicts the interaction between users and the system's features, considering internal and external factors influencing the system's behavior.

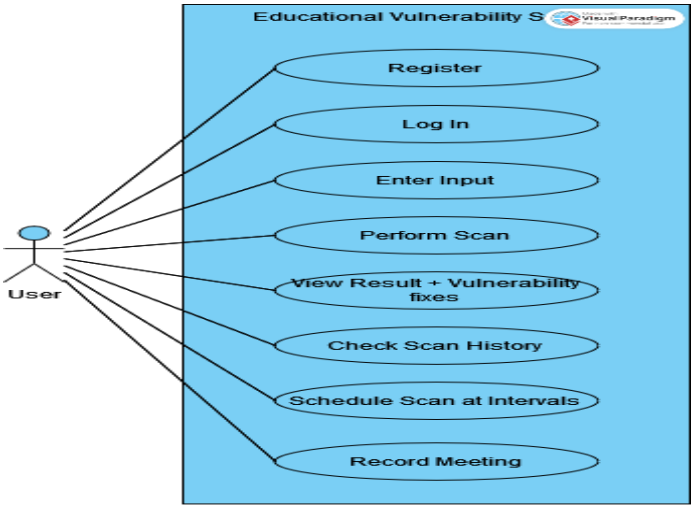

Figure 3. Illustration of user interaction with the vulnerability scanner system

4.3 Vulnerability Information Gathering

The development also includes gathering comprehensive rules for the rule engine from reputable sources such as OWASP and integrating these rules into the vulnerability scanner.

4.4 Rule Engine Integration

Lastly, the rule engine systematically checks the presence and configuration of important security headers in web applications' HTTP responses and is continuously updated to remain current and comprehensive.

4.5 Applying the User Interfaces

In order to design and implement the user interface for use in the vulnerability scanning application, a number of specific user interface requirements were identified and implemented using HTML, CSS, JavaScript, and Bootstrap. The frontend components were connected to the backend logic, thoroughly tested, and refined based on user feedback.

4.6 Testing and Validation

During testing and validation, the system underwent rigorous assessment to gauge its performance and efficacy so as to guide the programmers in addressing vulnerabilities. The validation process encompassed qualitative evaluations to assess the accuracy, reliability, and overall quality of the vulnerability scanning process and educational content on security. The system was exposed to various types of web applications from the dataset to evaluate its resilience and adaptability under diverse conditions.

4.6.1 *Referrer-Policy:*

X-XSS-Protection is a header that steps pages from loading when they detect reflected cross-site scripting (XSS) attacks. Some selected web pages were thoroughly scanned for missing key features that is paramount to web security. The following are the required steps to fix the detected missing feature These steps were implemented in Python, Node.js, Java and PhP. Each of the program or script written shows capacity to fix the missing feature.

Steps:

1. Set X-XSS-Protection header to “t: mode=block”
2. Disable inline scripting
3. Implement Content Policy (CSP) to mitigate XSS attacks.
4. Use input validation and output encoding to prevent XSS vulnerabilities

4.7 Comparison Analysis

The research compares HTTP security headers across various web applications using two widely recognized vulnerability scanners, Mozilla Observatory and SecurityHeaders.io. Four web application URLs from OWASP’s Vulnerable Web Application directory were chosen for the analysis. The headers and policies assessed included X-Frame-Option Header, Content-Security-Policy, X-XSS-Protection-Content-Type-Options, Referrer-Policy, Feature-Policy, and Permissions-Policy.

5 The Study Findings and Discussion

Findings from this study is discussed based on the consistencies and some discrepancies identified:

1. Consistencies:

i. *Clarity of Mitigation Strategy:* All the vulnerability scanners provided comprehensive information on security education, using clear and succinct language. In an effort to better secure a website from becoming vulnerable, the server configurations should be checked from time to time, the site should be hardened through the use of https with a valid certificate and whatever known issues detected should be patched promptly.

ii. *Availability of Further Resources:* All the vulnerability scanners offered hyperlinks for additional reading, particularly to the scanners' websites. Hyperlink are navigation elements usually found in a digital document that directs the users to a different document or web page. Well-known hyperlinks include Text links, Image links and Button links.

2. Discrepancies:

i. *Actionability:* The Mozilla scanner and securityheader.io scanner do not offer practical instructions. Their security instruction is provided in the form of an article. As a solution, the scanner used in this study provides explicit instructions for each stage, allowing programmers to adhere to the sequence of mitigation procedures in a simple way.

ii. *Depth of Explanation:* The securityheader.io lacks comprehensive explanations. The Mozilla scanner provides a comprehensive analysis but does not delve into the implementation details of security headers in different programming languages. The scanner used in this study provides a solution by demonstrating the implementation of the security header in multiple programming languages.

iii. Inclusion of a Code Snippet: Also, the scanner used in this study includes a code snippet for six different languages and their implementation: Python, node.js, Java, PHP, ASP.NET, and Ruby. Mozilla has an implementation for just HTML, while securityheader.io doesn’t have any.

6 Conclusion and Recommendation

This study aims to address the challenges faced by programmers in handling web application security issues effectively. It involves integrating a rule engine into a vulnerability scanner using Python to extract header requests and automate rule development. The main objectives were to optimize the generation of mitigation rules, enhance user engagement through an intuitive graphical interface, and improve the efficiency and effectiveness of vulnerability identification and mitigation.

The comparison investigation demonstrated the proposed method's consistent accuracy and reliability and can be compared to other well-established results in this area. The proposed system identified two additional missing headers; this unique development indicates a greater extent of detection coverage. Additionally, the system was found to be more programmer-friendly than several others in the related studies, allowing programmers versed in various languages to use the implementation codes for their needs.

However, the approach is not without its own limitations, such as the rule engine's reliance on manual input, the focus on vulnerabilities related to website headers, and the potential for false positives and improper implementation flaws.

Recommendation

Recommendations suggested include focusing on a holistic approach to vulnerability scanning, incorporating the educational vulnerability scanner into development environments, and enhancing the system with additional features to improve the user experience and learning process.

Future research opportunities include exploring the integration of machine learning algorithms to autonomously build and optimize rules based on continuous examination of web application security data. Additionally, investigating the incorporation of natural language processing methods to read and analyze the security rules and standards could further enhance the rule engine's abilities.

In summary, future research should focus on refining vulnerability scanners to be more intelligent, adaptable, and educational, while ensuring continual enhancements for the best user experience when identifying and fixing vulnerabilities.

7. References

[1] Abdulghaffar, K., et al. (2023). "Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners." Computers 12(11): 235.

[2] Buja, A. G., et al. (2024). "Analysis of Web Vulnerability Using Open-Source Scanners on Different Types of Small Entrepreneur Web Applications in Malaysia." Journal of Advanced Research in Applied Sciences and Engineering Technology 40(1): 174-188.

[3] Fadlil, A., et al. (2024). "Mitigation from SQL Injection Attacks on Web Server using Open Web Application Security Project Framework." International Journal of Engineering 37(4): 635-645.

[4] Gandikota, P. S. S. K., et al. (2023). "Web Application Security through Comprehensive Vulnerability Assessment." Procedia Computer Science 230: 168-182.

[5] Kapodistria, H., et al. (2011). "An advanced web attack detection and prevention tool." Information Management & Computer Security 19(5): 280-299.

[6] Lala, S. K., et al. (2021). Secure web development using owasp guidelines. 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), IEEE.

[7] Lathifah, A., et al. (2022). Security vulnerability analysis of the sharia crowdfunding website using owasp-zap. 2022 10th International Conference on Cyber and IT Service Management (CITSM), IEEE.

[8] Mushlihudin, M. and D. Faisal (2023). "Vulnerability Analysis and Prevention on Software as a Service (SaaS) of Archive Websites." Buletin Ilmiah Sarjana Teknik Elektro 5(3): 351-358.

[9] Priyanka, A. K. and S. S. Smruthi (2020). Web Application Vulnerabilities: Exploitation and Prevention. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE.

[10] Setiawan, E. B. and A. Setiyadi (2018). Web vulnerability analysis and implementation. IOP conference series: materials science and engineering, IOP Publishing.

[11] Sitohang, B., et al. (2024). Securing Cross-Site Request Forgery Vulnerabilities in Web Applications Using Mutation Analysis. 2024 2nd International Conference on Software Engineering and Information Technology (ICoSEIT), IEEE.

[12] Srivastava, M., et al. (2023). Security and Scalability of E-Commerce Website by OWASP threats. 2023 6th International Conference on Information Systems and Computer Networks (ISCON), IEEE.

[13] Ula, M., et al. (2023). Vulnerability risk assessment using Open Web Application Security Project (OWASP) methodology for e-marketplace. AIP Conference Proceedings, AIP Publishing.

[14] Xia, P., et al. (2024). "WalletRadar: towards automating the detection of vulnerabilities in browser-based cryptocurrency wallets." Automated Software Engineering **31**(1): 32.

[15] Yin, Z., et al. (2023). "Scanner++: Enhanced Vulnerability Detection of Web Applications with Attack Intent Synchronization." ACM Transactions on Software Engineering and Methodology **32**(1): 1-30.

[16] Zangana, H. M. (2024). "Exploring the Landscape of Website Vulnerability Scanners: A Comprehensive Review and Comparative Analysis." Redefining Security With Cyber AI: 111-129.