WebGuard: A Web Vulnerability Scanner for Web Applications

[1]Aniket Maurya, [2]Atharva Sail, [3]Himanshu Sharma, [4]Shailesh Kadam, [5]Sheeba P. S.

[1,2,3,4]Undergraduate Student, [5]Associate Professor

[1]*Department of CSE (IoT and Cyber Security including Blockchain Technology)*,

[1-5]Lokmanya Tilak College of Engineering, Navi Mumbai, India

[1]aniketmaurya_iot_2021@ltce.in, [2]atharvasail_iot_2021@ltce.in, [3] himanshusharma_iot_2021@ltce.in, [4]shaileshkadam_iot_2021@ltce.in, [5]sheebaps@ltce.in

***Abstract*— The increasing reliance on web applications has led to a surge in security vulnerabilities exploited by cybercriminals. This research introduces an automated Web Application Vulnerability Scanner developed in Python to address this issue. The scanner employs web crawling to analyze application components, detects various vulnerabilities including SQL Injection, Server-side Request Forgery, Cross-site Request Forgery, Directory Traversal and Cross-site Scripting, and provides comprehensive reports in JSON and PDF formats. The system offers both web-based and command-line interfaces, detailed attack type information, and remediation strategies, aimed at improving web application security and mitigating potential risks.**

***Index Terms*— Web Application Security, Vulnerability Scanner, Automated Vulnerability Detection, Cross-Site Scripting, Server-side request forgery.**

I. INTRODUCTION

In the contemporary digital landscape, web applications are fundamental to numerous industries, facilitating everything from financial transactions and healthcare services to e-commerce and communication. However, this increasing reliance on web applications has unfortunately coincided with a significant rise in cyber-attacks targeting their inherent security vulnerabilities. These vulnerabilities, which can include weaknesses like SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Server-Side Request Forgery and Directory Traversal, pose substantial risks. Exploitation of these flaws can lead to severe consequences, such as the theft of sensitive data, unauthorized access to systems, and disruption of critical services.

The traditional approach to identifying these vulnerabilities often involves manual testing, a process that is not only time-consuming and labor-intensive but also susceptible to human error. Moreover, manual testing proves to be impractical for the comprehensive and frequent assessments required by modern, large-scale web applications. To address these challenges, there is a clear need for automated solutions that can efficiently and accurately detect security flaws.

This research project introduces the development of a Python-based Web Application Vulnerability Scanner designed to automate the detection of common security vulnerabilities. This scanner aims to provide a robust and efficient solution for identifying weaknesses, generating detailed reports on identified vulnerabilities, and offering actionable remediation recommendations. By automating the vulnerability assessment process, this tool enables developers and security professionals to proactively strengthen web application security, mitigate potential risks, and safeguard sensitive information.

II. LITERATURE REVIEW

Vulnerability scanning is a critical process in securing web applications, which continue to be prime targets for cyberattacks [1], [7]. These attacks often exploit vulnerabilities such as SQL Injection, XSS, and CSRF, leading to severe consequences like data breaches and unauthorized access [1], [5]. Automated vulnerability scanners offer a faster and more scalable alternative to manual testing, making them essential for modern security practices [1], [4].

These scanners operate using static analysis, which inspects code without execution [8], and dynamic analysis, which evaluates application behavior at runtime [3], [6]. However, studies show that scanner performance varies significantly across different vulnerability types. For example, ZAP detects more SQLi and XSS issues, while Arachni performs better in LDAP-related vulnerabilities. No single scanner consistently excels across all OWASP Top 10 risks, suggesting that combining multiple tools may improve detection coverage [7].

Accuracy and false positive rates also differ widely. Acunetix and NetSparker have shown high detection accuracy with minimal false positives when mapped against OWASP and NIST standards [2]. Despite this, most scanners still mainly detect SQLi and XSS, with limited ability to address the full range of web vulnerabilities [1].

New approaches like Black Widow, which uses data-driven black-box scanning, significantly improve code coverage and have successfully identified previously undetected XSS vulnerabilities without false positives [3]. Similarly, AI-based scanners are emerging as powerful tools that enhance detection of unknown threats, reduce false positives, and support real-time scanning—though challenges remain around data quality and model generalization [4].

Integrating automated scanning into the software development lifecycle is considered a best practice, enabling early detection and remediation of vulnerabilities to ensure robust web application security [1], [9].

III. PROBLEM DEFINITION

Web applications are increasingly vulnerable to cyber-attacks due to security flaws such as SQL Injection, Cross-Site Scripting, Server-Side Request Forgery, Cross-Site Request Forgery, and Directory Traversal. Traditional methods for detecting these vulnerabilities are time-consuming, require expert knowledge, and are prone to errors. This research aims to develop an automated

web application vulnerability scanner to efficiently identify these flaws and provide developers with the means to secure their applications.

IV. METHODOLOGY

The research will commence with a thorough requirement analysis to define the project's objectives and specify the scope of vulnerabilities the scanner will detect; this will specifically include SQL Injection, Cross-Site Scripting, Server-Side Request Forgery, Cross-Site Request Forgery, and Directory Traversal. Following this, a comprehensive research and review phase will be conducted, examining existing web vulnerability scanning tools, methodologies, and frameworks to understand current approaches to vulnerability detection and reporting. The next critical step involves tool selection, where appropriate programming languages, libraries, and tools will be chosen for the scanner's various components, including web crawling, vulnerability detection (specifically for the identified vulnerabilities), report generation, and user interface development; key technologies will encompass Python (Flask), HTML, CSS, and libraries such as argparse, requests, BeautifulSoup, and Scrapy.

The implementation phase will then proceed with the development of the scanner's core functionalities. This will include web crawling to discover URLs and web components, the creation of vulnerability detection modules meticulously designed to identify SQL Injection, Cross-Site Scripting, Server-Side Request Forgery, Cross-Site Request Forgery, and Directory Traversal, and the establishment of reporting mechanisms to generate detailed security reports in formats like JSON and PDF. User interfaces, both web-based and command-line, will also be developed to facilitate user interaction.

Subsequently, a rigorous testing and validation phase will be carried out to ensure the scanner's accuracy and effectiveness in detecting the specified vulnerabilities, potentially involving testing against benchmark applications like DVWA (Damn Vulnerable Web Application). Finally, the scanner's performance will be analyzed in the optimization and tuning phase to identify areas for enhancement, focusing on optimizing scanning speed and tuning detection algorithms to minimize false positives and negatives in the detection of SQL Injection, Cross-Site Scripting, Server-Side Request Forgery, Cross-Site Request Forgery, and Directory Traversal.

V. FEATURES

A. Vulnerability Detection:

The scanner is capable of detecting five common web application vulnerabilities: SQL Injection, Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF), Cross-Site Request Forgery (CSRF), and Directory Traversal/Path Traversal.

B. Web Crawling:

The scanner includes web crawling functionality to automatically discover URLs, links, forms, and other relevant web components within the target application.

C. User Interface:

It offers flexibility with both a web-based user interface (UI) and a command-line interface (CLI). The web-based UI provides an interactive and user-friendly experience, while the CLI allows for direct command execution.

D. Attack Type Selection:

Users can specify the types of vulnerabilities they want to scan for, allowing for targeted assessments. The system provides descriptions and references for each attack type to aid user understanding.

E. Automated Scanning:

The scanner automates the process of identifying security flaws, reducing the need for manual testing.

F. Reporting:

The scanner generates detailed and structured reports in JSON and PDF formats. Reports include information on identified vulnerabilities and recommended remediation steps.

G. Remediation and Recommendations:

The system provides security best practices and mitigation strategies to help users address identified vulnerabilities.

H. Learning:

The scanner can be designed to incorporate machine learning techniques to improve vulnerability detection rates over time. This learning capability allows the scanner to adapt to new and emerging threats.

VI. IMPLEMENTATION SNAPSHOTS

FIGURE 1. Visit the Web App

(a) (b)

FIGURE 2. Explore how web attacks are performed

FIGURE 3. Learning section to learn about various web vulnerabilities

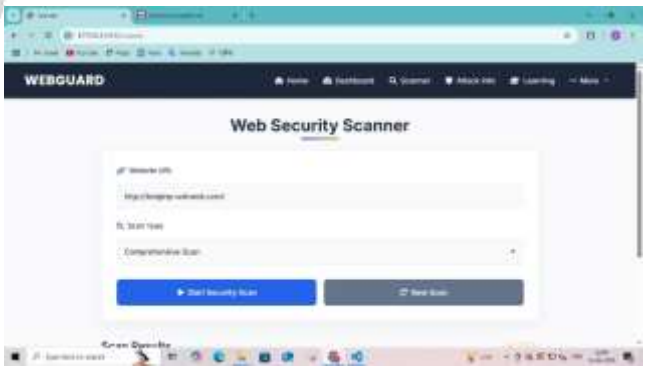

FIGURE 4. Enter the target website URL for scanning

FIGURE 5. View the live scanning process

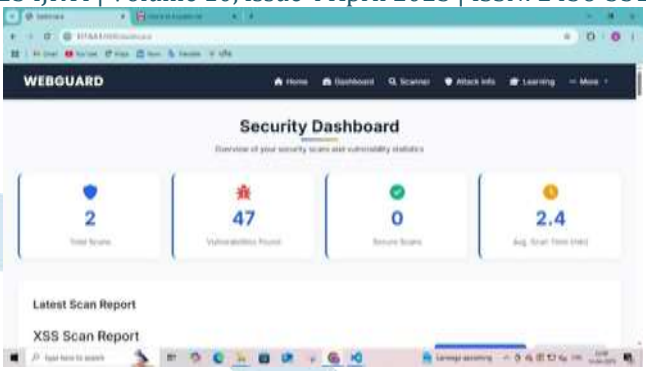

FIGURE 6. Scan results on the security dashboard

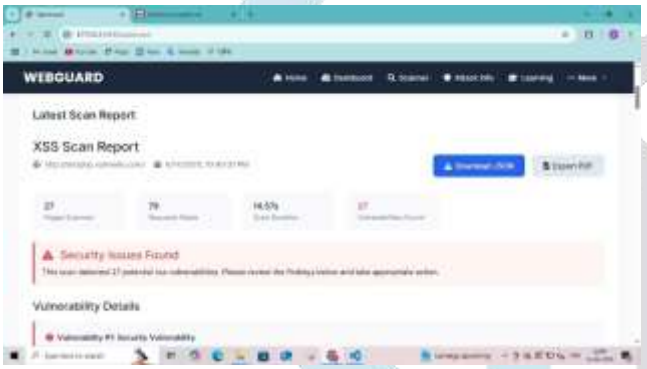

FIGURE 7. Scan results on the security dashboard

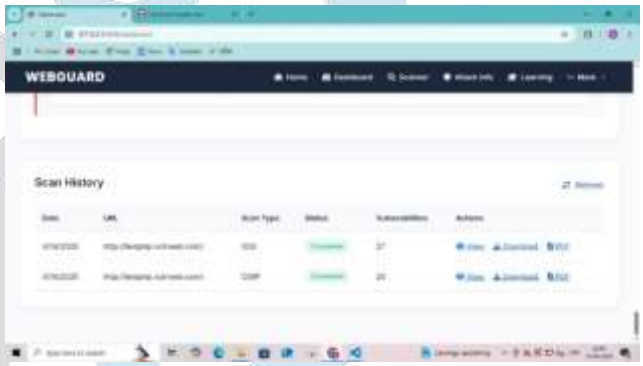

FIGURE 8. Download scan reports in JSON and PDF formats

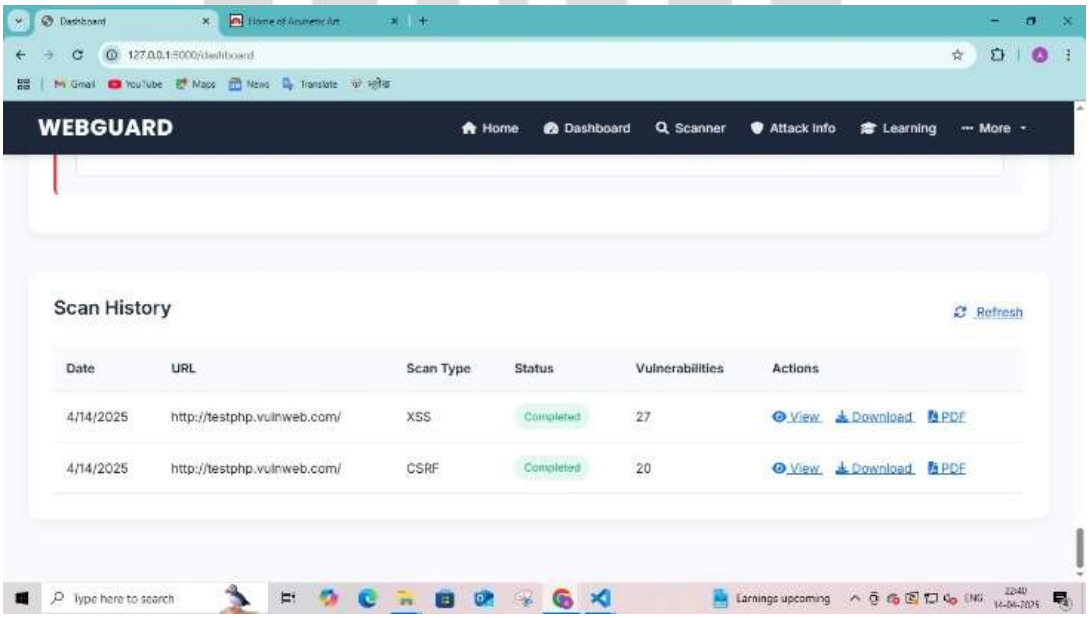

FIGURE 9. Access previous scan history anytime

VII. CONCLUSION

In today's digital landscape, securing web applications is of utmost importance due to the increasing complexity of cyber threats. This project introduces an automated web vulnerability scanner designed to detect and analyze security flaws, including prevalent ones like SQL Injection, XSS, CSRF, Directory Traversal, and SSRF. The developed system offers flexibility for various users by providing both a web-based interface and a command-line interface. It automates the detection of common web vulnerabilities, such as SQL injection and Cross-Site Scripting (XSS). The scanner performs web crawling, conducts targeted vulnerability assessments, and generates detailed reports in JSON and PDF formats, ensuring a structured and efficient security analysis process. By leveraging established security frameworks and methodologies like OWASP and NIST guidelines, the system enhances the reliability of vulnerability detection.

Compared to existing security tools, this system aims to provide a cost-effective, user-friendly, and customizable approach to web application security. It offers detailed, user-friendly reports to streamline vulnerability management and remediation. The inclusion of comprehensive risk analysis and remediation suggestions empowers organizations to proactively address vulnerabilities before they can be exploited. Furthermore, the scanner offers continuous monitoring to help organizations proactively manage security risks and is adaptable to emerging threats, ensuring up-to-date protection. Ultimately, this project contributes to strengthening cybersecurity measures by making vulnerability assessment more accessible, automated, and efficient, and by significantly reducing the risk of data breaches and cyberattacks, thereby enhancing the overall security and reliability of web applications.

REFERENCES

[1] S. Alazmi and D. Conte De Leon, "A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners," IEEE Access, vol. 10, pp. 33200-33219, 2022.

[2] M. Qasaimeh, A. Shamlawi, and T. Z. Khairallah, "Black Box Evaluation of Web Application Scanners: Standards Mapping Approach," Proc. Int. Conf. Cybersecurity (ICoCS), vol. 96, pp. 4584-4596, 2018.

[3] B. Eriksson, G. Pellegrino, and A. Sabelfeld, "Black Widow: Blackbox Data-driven Web Scanning," IEEE Symposium on Security and Privacy (SP), 2021.

[4] P. Devadiga, S. Varankar, S. Kumari, and N. Mishra, "AI-Based Web Vulnerability Scanner: A Comprehensive Review," SSRN Journal, 2024.

[5] OWASP Foundation, OWASP Top 10: Critical Web Application Security Risks, [Online]: https://owasp.org/www-project-top-ten/.

[6] PortSwigger, Burp Suite Vulnerability Scanner Documentation, 2025. [Online]: https://portswigger.net/burp/vulnerability-scanner.

[7] B. Mburano and W. Si, "Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark," Proc. IEEE Int. Conf. Web Services (ICWS), 2018.

[8] C. C. Nnaemeka and O. Ehichoya, "Evaluating Security Vulnerabilities in Web-Based Applications Using Static Analysis," arXiv , 2022.

[9] Invicti, How Web Vulnerability Scanners Work, 2025. [Online]: https://www.invicti.com/blog/web-security/vulnerability-scanners-work/

[10] Pentest-Tools.com, *Advanced Website Vulnerability Scanning Techniques*, [Online]: https://pentest-tools.com/website-vulnerability-scanning.