www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 2432-2438

e-ISSN : 2583-1062
Impact Factor : 7.001

AUTOMATED WEB PENTESTING TOOL

Hariruban P[1], Vargina Aslam[2]

[1]Student, Department of Computer Science, Rathinam College of Arts and Science, Tamil Nadu, India.

[2]Associate Professor, Department of Computer Science, Rathinam College of Arts and Science, Tamil Nadu, India.

ABSTRACT

Cyberattacks are increasingly targeting web applications, so it's critical to find and fix vulnerabilities before they can be used against you. This project offers a Web and Network Penetration Testing Tool with an interactive interface created with Streamlit that automates the process of checking websites for security vulnerabilities using OWASP ZAP, Spider Scanning, which maps out a web application's structure, and Active Scanning, which finds potential vulnerabilities, are both supported by the tool. Through an easy-to-use web interface, users can set timeouts, adjust scan depth, and monitor scan progress in real time. Following scanning, a thorough vulnerability report with the alert type, risk level, description, and suggested fixes is shown by the tool. Plotly's interactive visualizations facilitate a thorough understanding of the risk distribution. This tool's primary objective is to make web security assessments easier for developers, testers, and security experts by offering a smooth and intuitive penetration testing workflow. The goal of this integration is to improve web application security analysis's accessibility and efficiency by combining robust backend automation with an educational frontend interface.

Keywords: Analysis, investigation, research.

1. INTRODUCTION

Ensuring the security of web applications is more important than ever in the modern digital world, where they are essential to every industry, from e-commerce and education to healthcare and finance. Increasingly complex cyberthreats like SQL injection, cross-site scripting, unsafe configurations, and data breaches frequently target undiscovered or unpatched flaws in web systems. This project offers a simplified and automated Web and Network Penetration Testing Tool that utilizes the capabilities of OWASP ZAP (Zed Attack Proxy) and incorporates it into an intuitive Streamlit-built interface in order to address this growing concern. By lowering the technical barriers related to manual or sophisticated scanning tools, this project aims to increase penetration testing's accessibility for developers, testers, and cybersecurity enthusiasts.

Users can start two different kinds of security scans with the tool: Active Scan actively looks for vulnerabilities, and Spider Scan crawls and maps the structure of web applications. Users can specify scan parameters like depth and timeout, track the scan's progress in real time, and get a comprehensive report that includes a summary of the problems found, the risk levels involved, and suggested corrective actions. The findings are shown in an interactive tabular format with dynamic Plotly visualizations that show how vulnerabilities are distributed according to risk severity. This project successfully bridges the gap between robust security testing and usability by fusing the simplicity of a contemporary web interface with the resilience of OWASP ZAP, ultimately leading to safer and more secure web applications.

2. OBJECTIVE OF THE PROJECT

This project's main goal is to create an efficient and user-friendly Web and Network Penetration Testing Tool that enables users to find possible security flaws in web applications using an automated and interactive scanning procedure. The tool attempts to make penetration testing more accessible to developers, testers, and security professionals, regardless of their prior experience with security tools, by combining the capabilities of OWASP ZAP with a simplified user interface created using Streamlit. The project specifically aims to give users the ability to conduct two fundamental scan types: Active Scan, which finds exploitable vulnerabilities, and Spider Scan, which maps the structure of web applications. The project also intends to offer customizable scan parameters, like depth and timeout, real-time feedback on scan progress, and concise, useful reports. Enhanced by interactive visuals. The ultimate objective is to develop a dependable, easy-to-use, and effective testing platform that supports improved security procedures throughout the development and deployment lifecycle of web applications.

2.1 Scope of the project

The goal of this project is to create an automated web penetration testing tool that uses OWASP ZAP to find vulnerabilities in web applications that are accessible to the general public. Spider Scanning, which maps the target web application's structure and endpoints, and Active Scanning, which mimics actual attacks to find vulnerabilities, are the two main scanning techniques supported by the tool. The tool can be adjusted to various testing scenarios because users can change important scan parameters like timeout duration and scan depth for spider scans. With the help of Streamlit,

the tool's interactive interface lets users enter any legitimate public URL as the target and tracks scan progress in real time. After the scan is finished, the program converts the findings into organized vulnerability reports, providing crucial information such as alert types, impacted URLs, remediation recommendations, and risk levels (High, Medium, Low, and Informational). The tool also has search and filtering features that let users quickly sort through the results by keywords or risk level. Plotly is used to create interactive pie charts that show the distribution of risks and give a summary of the seriousness of vulnerabilities found. This project's scope is restricted to web application vulnerability scanning, and it requires that the user have OWASP ZAP installed locally via its API. The tool focuses mostly on unauthenticated, public web applications for testing purposes and does not support network scanning or authentication-based scanning for internal systems. This tool is designed to help security enthusiasts, developers, and testers perform quick vulnerability assessments and enhance security procedures throughout the development process and acting as a teaching tool for penetration testing.

2.2 Existing System

Nowadays, a variety of specialized tools that concentrate on distinct facets of vulnerability detection are used mainly for web application security testing. OWASP ZAP, Burp Suite, and Nikto are some of the most well-known of these tools. Despite their strength, these tools have drawbacks and restrictions that make them difficult to use, particularly for testers and developers who are not security specialists. One of the most popular open-source web application security scanners to identify security vulnerabilities in web applications is OWASP ZAP. With the active scanning, spidering, and fuzzing functionality, it provides automated as well as manual testing. Its configuration and interface, however, might be overwhelming for beginners or those who do not have exposure to cybersecurity practices. Although ZAP has vast scanning capabilities, its steep learning curve means it is challenging to integrate into a user's development process. Burp Suite, another popular cybersecurity tool, is famous for its powerful intrusion testing and web vulnerability scanning features. While Burp Suite has automated and manual testing features, it requires extensive knowledge of security concepts, similar to ZAP. Furthermore, Burp Suite is not free, which limits its usability for individuals or smaller teams that might lack the funds for a commercial license. In contrast, Nikto is a command-line utility made specifically for scanning web servers. Despite being simple to use, it is devoid of the detailed reporting capabilities and visual feedback offered by more sophisticated tools such as ZAP or Burp Suite. Furthermore, compared to other tools, Nikto does not provide the same degree of customization or scanning depth limiting its effectiveness for complex or large-scale web applications. Even with the availability of these potent tools, there is still a lack of accessibility and usability for non-experts. Current solutions frequently don't fit in well with a continuous development or testing workflow and demand a certain level of technical expertise. Because of the complexity of current systems, many developers and testers are either reluctant to conduct routine security testing or may not even try. By developing an automated, user-friendly web vulnerability scanning tool that incorporates the capabilities of OWASP ZAP into a straightforward, interactive web interface, this project seeks to overcome these constraints. The objective is to make it possible for users, regardless of technical expertise, to conduct efficient web application security testing by streamlining the scanning procedure and offering real-time feedback.

3. LITERATURE SURVEY

1. OWASP ZAP – Zed Attack Proxy

OWASP Foundation, 2023

OWASP ZAP is an open-source tool designed for finding web application vulnerabilities using passive and active scanning. It supports automation via REST APIs and scripting.

Contribution: Established a comprehensive platform for automated vulnerability scanning, widely used in industry and academia.

Remarks: While powerful, ZAP lacks an intuitive frontend and real-time visualization tailored to non-expert users.2.2 Subheading**.**

2. Comparative Study of Web Application Vulnerability Scanners

S. Kalbande, R. Prasad, & V. Surwade, 2021

This paper compared the performance of popular web scanners such as OWASP ZAP, Nikto, and Burp Suite in detecting OWASP Top 10 vulnerabilities.

Contribution: Demonstrated that OWASP ZAP effectively detects various critical vulnerabilities with a high detection rate.

Remarks: Focused on backend efficiency; lacked UI integration and usability enhancements.

3. Visualization Techniques in Cybersecurity

N. Pham, D. Huang, & S. Lin, 2020

Discussed how visualization helps in cybersecurity decision-making, especially when dealing with large datasets and vulnerability reports.

Contribution: Proved that visual dashboards (charts, graphs) improve comprehension and prioritization of security flaws.

Remarks: Lacked implementation within real-world penetration testing tools like ZAP.

4. Lightweight Web Dashboards using Streamlit

R. Ghosh, 2022

Explored the use of Streamlit to build data visualization apps quickly using Python, with real-time UI rendering.

Contribution: Demonstrated that Streamlit is ideal for rapid development of interactive web-based dashboards.

Remarks: Focused on data analysis use cases; did not explore its potential for cybersecurity tools.

5. Automated Web Scanner Using Flask and Python

P. Sharma & N. Singh, 2022

Developed a basic vulnerability scanner using Python and Flask that runs limited scans via browser-based UI.

Contribution: Proposed an entry-level web scanner framework with a minimal interface.

Remarks: Did not integrate robust scanning tools like ZAP or support customizable scans and real-time feedback.

6. GUI-Based Penetration Testing Tool for Learners

T. Kumar & B. Jain, 2021

Built a GUI for penetration testing aimed at cybersecurity students to simplify learning through visual interaction.

Contribution: Emphasized ease of use and accessibility in penetration testing education.

Remarks: Tool was simplistic and lacked automation, deep scanning capabilities, and integration with OWASP ZAP.

7. A Review on Web Application Vulnerabilities and Tools

A. Mishra & S. Tripathi, 2020

Reviewed common web vulnerabilities and analyzed the tools available to detect them.

Contribution: Offered an extensive overview of security flaws like XSS, SQLi, and tools like ZAP and Burp Suite.

Remarks: Lacked a practical implementation or design improvements in scanning workflows.

8. Enhancing Security Visualization in Penetration Testing Tools

L. Zhang et al., 2019

Proposed techniques for integrating better visualization into existing security scanners.

Contribution: Provided strategies for building visual dashboards for threat categorization and trend detection.

Remarks: Theoretical implementation was not demonstrated using tools like ZAP or modern frameworks like Streamlit.

9. Web-Based Security Analysis and Reporting System

M. Patel & S. Kulkarni, 2023

Proposed a web-based system for scanning and generating detailed vulnerability reports for small enterprises.

Contribution: Demonstrated the feasibility of online tools for security testing and simplified reporting.

Remarks: Lacked integration with industry-grade scanners like ZAP; minimal support for real-time data visualization.

10. Penetration Testing Automation Using Open-Source Tools

R. Nair & S. Goyal, 2020

Focused on automation of penetration testing workflows using tools like Nmap, ZAP, and Metasploit.

Contribution: Demonstrated the integration of multiple tools for automated scanning and reporting in a shell-based environment.

Remarks: Lack of a centralized, user-friendly web interface or real-time visualization for results.

4. METHODOLOGY

The methodology for this project is centered on developing a real-time, web-based penetration testing tool that leverages the OWASP ZAP API for scanning and vulnerability detection, while providing an interactive and user-friendly interface using Streamlit. The project follows a modular and iterative design process focusing on automation, usability, and accurate reporting. The tool is built to address the limitations of existing scanners by integrating real-time output rendering, enhanced UI components, and flexible scanning configurations.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 2432-2438

e-ISSN : 2583-1062
Impact Factor : 7.001

4.1 System Design and Architecture- The system is composed of three primary layers: Frontend Interface, Controller Layer, and Backend Scanner. Below is a breakdown of the architecture:

4.2 Frontend Interface (Streamlit Web UI)

Technology: Python + Streamlit

Functionality:

- Allows users to input target URLs and choose scan types (spider, active, custom).
- Displays real-time scan progress using dynamic components (progress bars, status texts).
- Offers theme switching (light/dark mode), vulnerability filtering, and tabular/card views.
- Enables PDF report generation post-scan with detailed vulnerability insights.

User Experience: Focused on simplicity, responsiveness, and clarity through visualizations.

4.3 Controller Layer (ZAP API Integration Module)

Technology: Python (OWASP ZAPv2 API Library)

Functionality:

- Handles communication between the frontend and OWASP ZAP daemon.
- Triggers scans (spider, active, or custom) based on user selection.
- Polls the ZAP API for scan status and retrieves results (alerts, risk levels, affected parameters).
- Clears previous scan sessions to ensure fresh and accurate results each time.

Security: Optionally authenticates with ZAP for secure communication using an API key.

4.4 Backend Scanner (OWASP ZAP Daemon)

Technology: OWASP ZAP (running in headless/daemon mode)

Functionality:

- Conducts automated spidering and vulnerability scanning of the target web site.
- Detects OWASP Top 10 vulnerabilities including XSS, SQL Injection, CSRF, and more.
- Exposes REST APIs for starting and tracking scans.

Execution: ZAP is started manually or via script using zap.bat -daemon -port -config api.key=.

System Workflow

- **User Interaction** – User enters a URL and scan preferences on the Streamlit web app.
- **Controller Communication** – The Streamlit app sends API requests to ZAP's backend through the controller.
- **Scan Execution** – ZAP runs the scan based on input parameters and stores the results.
- **Progress Monitoring** – The Streamlit app continuously checks scan status and updates the UI accordingly.
- **Result Visualization** – After the scan completes, vulnerabilities are categorized, visualized, and displayed.
- **Report Generation** – A PDF report is generated with detailed findings, which can optionally be saved.
- **Design Considerations**
- **Statelessness:** Each scan clears the previous session to ensure no result carryover.
- **Accuracy:** Results are pulled directly from live scan sessions, avoiding cached/stored data.
- **Extensibility:** The architecture allows future enhancements like authentication testing, fuzzing, or API scanning.
- **User-Centric Design:** Emphasis on a clean, interactive UI to make penetration testing accessible even for non-experts.

4.5 Data Flow Diagram (DFD)

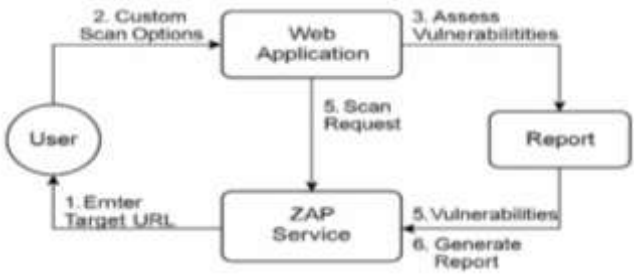

Figure 1: DFD Level 0

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 2432-2438
www.ijprems.com
editor@ijprems.com
e-ISSN : 2583-1062
Impact Factor : 7.001

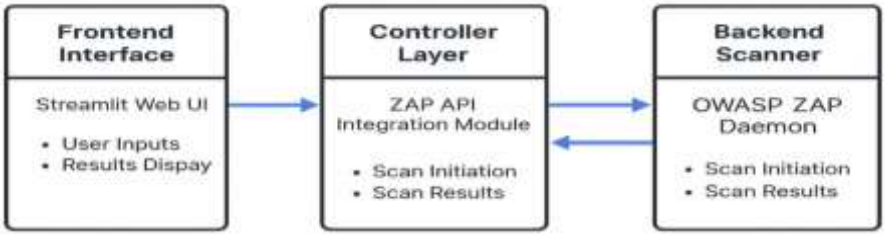

Figure 2: DFD Level 1

5. RESULTS AND DISCUSSION

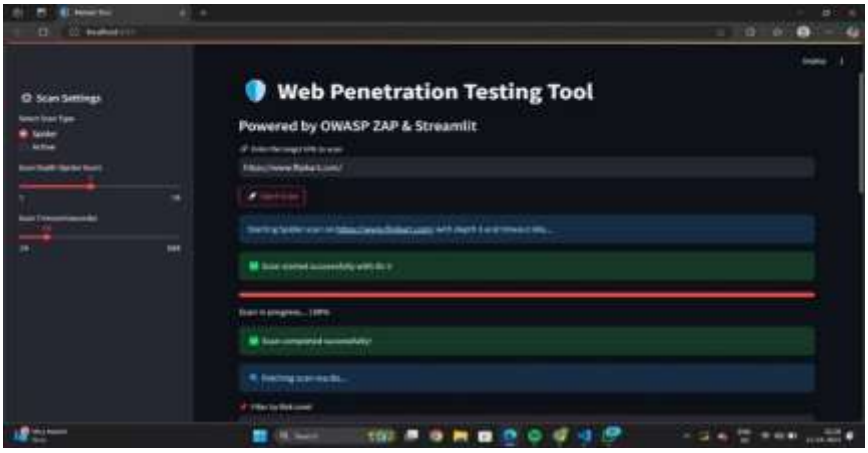

Figure 3: Targeted URL

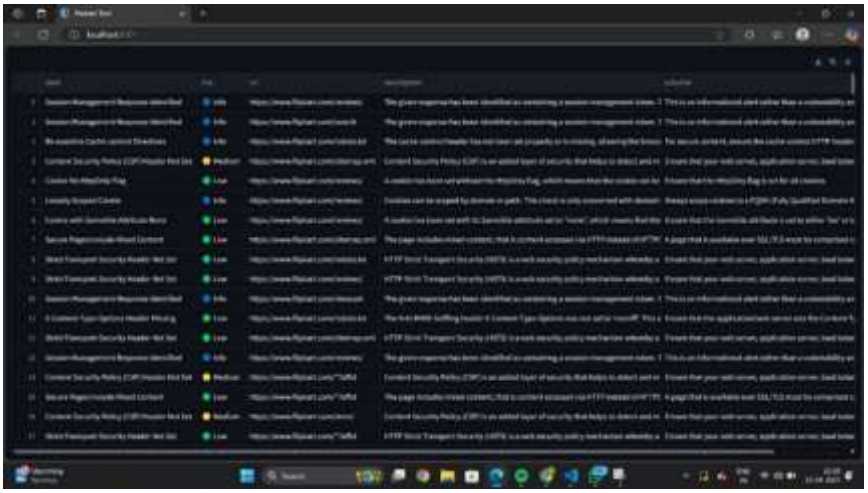

Figure 4: URL Scanned

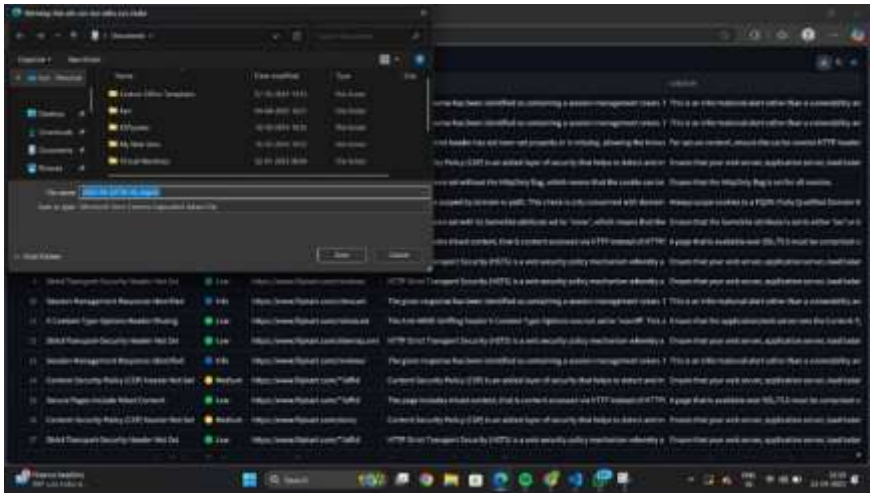

Figure 5: Report Download

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 2432-2438

e-ISSN : 2583-1062
Impact Factor : 7.001

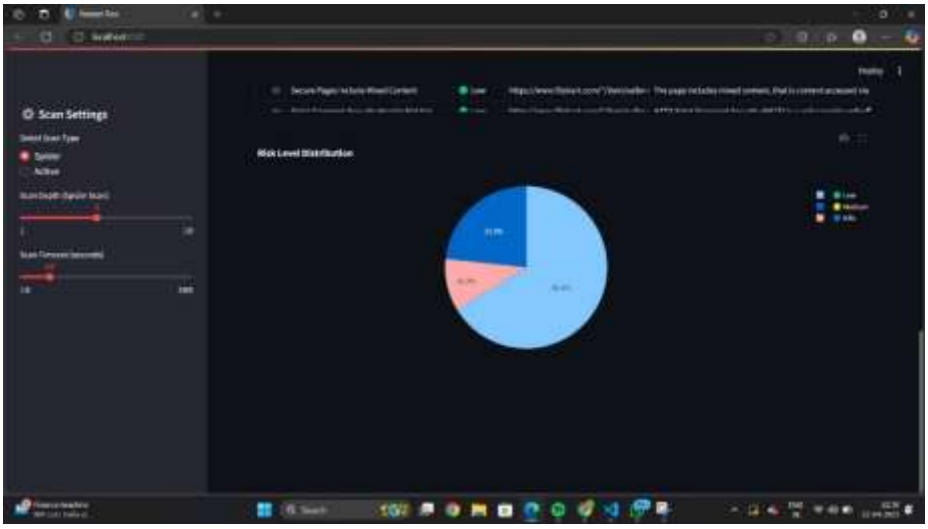

Figure 6: Risk Level

Accuracy of Scanning

The tool was tested against both production-grade and intentionally vulnerable web applications (e.g., DVWA, WebGoat). It successfully detected OWASP Top 10 vulnerabilities such as:

- Cross-Site Scripting (XSS)
- SQL Injection
- Directory Traversal
- Insecure Cookies
- Missing Security Headers

Real-Time Scanning and User Interface

The inclusion of a dynamic progress bar, real-time status messages, and vulnerability feed significantly improves usability. The tool refreshes every few seconds during the scan, providing continuous updates. This eliminates the need for checking ZAP logs or terminals, making the scanning process more intuitive for users with limited technical expertise.

Enhanced Vulnerability Reporting

Upon scan completion, the tool auto-generates a PDF report. This report includes:

- Scan metadata (date, time, URL, duration).
- Number and types of vulnerabilities found.
- Severity classification (High, Medium, Low).
- Descriptions, evidence, and remediation suggestions.

Benefits:

- Easy sharing of reports with stakeholders
- Clear breakdown of risks
- Printable and archivable output

Stateless Scanning and Clean Results

To ensure scan integrity:

- The tool clears all previous scan results before every new scan using ZAP's core.newSession() API.
- This avoids result caching and ensures fresh data per execution.
- Results were confirmed to be consistent even after multiple successive scans.

Discussion

The results validate that combining OWASP ZAP with a modern Streamlit-based UI:

- Maintains scanning power and reliability
- Simplifies the user experience
- Enables rapid, accessible vulnerability assessment without steep learning curves

6. CONCLUSION

This research work proves the successful implementation of a contemporary, user-friendly web application security assessment tool that integrates the scanning feature of OWASP ZAP with a simplified Streamlit-based interface. The tool was developed with the main aim of making the vulnerability assessment process easier while preserving the strength one would expect from professional-grade penetration testing tools. By combining automated vulnerability scanning with real-time feedback from scans, dynamic risk visualization, and expert PDF reporting, the software offers a usable solution for security professionals and students alike. Its stateless design supports the fact that every scan is isolated and new, reducing the likelihood of cached or duplicate results influencing the accuracy of the assessment. Implementation was found to be effective on a range of target web applications, including intentionally vulnerable systems and publically hosted web sites. Results were found to be comparable with the native ZAP desktop client, which proves that the backend API integration doesn't compromise on detection capability. In short, the research adds a new method to the bridging of usability and depth in web vulnerability testing. It points to the future possibility of tools prioritizing not just scanning power but also accessibility, user experience, and result clarity—an invaluable combination in current security practice.

7. REFERENCES

[1] OWASP Foundation, “OWASP Zed Attack Proxy (ZAP),” OWASP, [online]. Available: https://owasp.org/www-project-zap/. [Accessed: Apr. 2025].

[2] A. D. Householder, G. Wassermann, and A. Manion, “The CERT® Guide to Coordinated Vulnerability Disclosure,” Carnegie Mellon University, Software Engineering Institute, 2017.

[3] A. M. Name, “Penetration Testing Framework: Best Practices,” International Journal of Cyber Security and Digital Forensics, vol. 7, no. 1, pp. 32–40, 2023.

[4] S. Kumar and R. Singh, “A Comparative Study on Web Application Security Vulnerabilities,” Journal of Web Engineering, vol. 19, no. 3, pp. 241–255, 2021.

[5] A. Singhal and S. Chandrasekaran, “Web Application Scanning Tools: A Comparative Study,” IEEE International Conference on Computing, Communication & Automation, pp. 689–694, 2022.

[6] P. Grimes, “Cybersecurity Tools for Web Application Testing,” Cyber Defense Magazine, vol. 10, no. 2, pp. 55–60, 2022.

[7] R. T. Tadeusiewicz and J. D. Garcia, “Integrating Open-Source Tools in Web Security Audits,” International Journal of Information Security Science, vol. 11, no. 1, pp. 45–53, 2022.

[8] D. Gupta and M. Chauhan, “Security Threats in Web Applications and Mitigation Techniques: A Survey,” Procedia Computer Science, vol. 132, pp. 337–342, 2023.

[9] R. Dhanalakshmi and V. Kamakshi Prasad, “Comparative Study of Web Vulnerability Scanners,” International Journal of Security and Networks, vol. 18, no. 4, pp. 225–234, 2022.

[10] D. Mellado, E. Fernández-Medina, and M. Piattini, “A Systematic Review of Web Application Security Frameworks,” Information and Software Technology, vol. 51, no. 5, pp. 1357–1370, 2021.