INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

Sentinel Security Scanner

Web Application Security Scanner for SQL Injection and XSS Detection

Dr . K.Satheesh, Professor, Department of Computer Science,
Vasireddy Venkatadri Institute of Technology, Nambur, Guntur District, Andhra Pradesh, India.
Gottimukkala Gayathri, Gummalla Manasa, Dasari Meghana ,Gali Naga Keerthi
UG Students, Department of AI&DS,
Vasireddy Venkatadri Institute of Technology, Nambur, Guntur District, Andhra Pradesh, India
kns9@live.com, 22bq1a5449@vvit.net , 22bq1a5452@vvit.net , 22bq1a5430@vvit.net , 22bq1a5440@vvit.net

Abstract—Web application security testing is a critical yet time-consuming process. Traditional manual testing and heavy commercial tools often require significant setup and expertise. Sentinel Security Scanner is a lightweight, open-source web vulnerability scanner developed in Java Spring Boot that automatically detects SQL Injection and Cross-Site Scripting (XSS) vulnerabilities. The system crawls target websites using Jsoup, identifies forms and URL parameters, injects safe test payloads, and analyzes responses for vulnerability indicators. Results are stored in a MySQL database with risk levels (HIGH/MEDIUM/LOW) and full evidence. A modern single-page frontend provides real-time scan progress, summary cards, and downloadable HTML reports. Sentinel includes a built-in vulnerable test target for immediate testing and supports auto-authentication for popular training platforms like DVWA and OWASP Juice Shop. The tool is designed for educational institutions, developers, and security enthusiasts who need a fast, reliable, and easy-to-deploy scanner without complex configuration.

Index Terms—SQL Injection, Cross-Site Scripting, XSS, web application security, vulnerability scanner, Spring Boot, Jsoup, security testing, OWASP.

I. INTRODUCTION

1.1 Introduction

Web applications are the backbone of modern digital services, but they remain highly vulnerable to security threats. Among the most critical and frequently exploited vulnerabilities are SQL Injection (SQLi) and Cross-Site Scripting (XSS), which continue to rank in the OWASP Top 10. These attacks can lead to unauthorized data access, data theft, website defacement, and complete system compromise.

Traditionally, detecting such vulnerabilities requires manual testing or expensive commercial tools that demand significant setup time and expertise. Students, developers, and small organizations often struggle to perform regular security assessments due to the complexity and cost involved.

Sentinel Security Scanner is a lightweight, educational, and fully automated web application security scanner developed using Spring Boot and Jsoup. It automatically crawls target websites, identifies forms and URL parameters, injects safe test payloads, and detects SQL Injection and XSS vulnerabilities with clear evidence and risk classification. The tool also includes a built-in vulnerable test target and supports auto-authentication for popular training platforms like DVWA and OWASP Juice Shop.

By providing real-time scanning, professional HTML reports, and an intuitive web interface, Sentinel makes web security testing accessible, educational, and efficient for students and developers.

1.2 Problem Statement

Security testing of web applications is often neglected due to several practical challenges faced by developers and students:

1.2.1 Difficulty in Performing Regular Security Testing: Most developers and students lack access to powerful tools like Burp Suite or ZAP, and manual testing is extremely time-consuming and error-prone.

1.2.2 Lack of Simple and Educational Tools: Existing open-source scanners are either too complex for beginners or lack proper reporting and visualization features.

1.2.3 Absence of Built-in Test Environment: There is no easy way to immediately test and learn vulnerability detection without setting up external vulnerable applications.

1.2.4 Poor Reporting and Risk Classification: Most basic scanners do not provide clear evidence, risk levels (HIGH/MEDIUM/LOW), or professional downloadable reports.

1.2.5 Limited Real-time Feedback: Users often get only command-line output with no progress tracking or user-friendly interface.

These challenges result in insecure web applications being deployed and students missing valuable hands-on security learning opportunities.

1.3 Objectives

1.3.1 Develop an Automated Vulnerability Scanner: Create a fully functional scanner capable of automatically detecting SQL Injection and Cross-Site Scripting vulnerabilities in web applications.

1.3.2 Provide User-Friendly Interface and Real-time Progress: Build a modern single-page web interface with real-time scan progress, status updates, and interactive results dashboard.

1.3.3 Implement Professional Reporting: Generate detailed HTML reports with vulnerability evidence, risk levels,

affected URLs, and payloads for easy sharing and documentation.

1.3.4 Include Built-in Test Target: Provide a built-in vulnerable test environment so users can immediately practice and verify the scanner without external setup.

1.4 Scope & Limitations

1.4.1 Scope: The Sentinel Security Scanner is designed for educational institutions, individual developers, and security enthusiasts. It supports scanning of any web application (local or hosted), detects SQL Injection (error-based, blind, time-based) and XSS (reflected and basic stored simulation), stores results in MySQL database, and generates professional reports. It also includes auto-authentication support for DVWA and Juice Shop training platforms.

1.4.2 Feature Set:

- Automatic website crawling and form detection
- Safe payload-based vulnerability testing
- Risk classification (HIGH, MEDIUM, LOW)
- Real-time scan progress tracking
- Downloadable HTML reports
- Built-in vulnerable test target
- Responsive web-based frontend

1.4.3 Limitations:

- Currently focuses only on SQL Injection and XSS (other vulnerabilities like CSRF, Command Injection can be added later).
- Does not perform actual exploitation or data dumping.
- Scanning speed depends on target website response time.
- Requires active internet connection for crawling external websites.
- Designed primarily for educational and authorized testing purposes only.

II. Literature Review

Literature survey is a crucial step in any software development process. Before developing a web application, it is important to assess various factors such as time, resources, and economic feasibility. Once these factors are determined, the selection of appropriate programming languages, frameworks, and tools is made to ensure the successful development of the application. The Sentinel Security Scanner project builds upon existing research in automated web vulnerability scanning, payload-based testing, and open-source security tools to provide users with an efficient, educational, and lightweight solution for detecting SQL Injection and Cross-Site Scripting vulnerabilities.

Bernardo Damele, "SQLMap -- Automatic SQL Injection and Database Takeover Tool": SQLMap is one of the most popular open-source tools for detecting and exploiting SQL Injection vulnerabilities. It supports a wide range of databases and testing techniques including error-based, boolean-based, time-based, and union-based injections. The tool is highly effective for penetration testers. However, it is command-line based, requires technical expertise, and lacks a user-friendly interface and reporting features suitable for students and beginners.

OWASP ZAP (Zed Attack Proxy) -- OWASP Foundation: OWASP ZAP is a widely used open-source web application security scanner that supports automated scanning for various vulnerabilities including SQL Injection and XSS. It provides features such as spidering, active scanning, and intercepting proxy. While ZAP is powerful and free, it has a steep learning curve, consumes high system resources, and is more suitable for professional security teams rather than educational use or quick local testing.

R. Johari and P. Sharma, "A Survey on Web Application Vulnerabilities and Tools for Detection": This research paper presents a comprehensive survey of web vulnerabilities with special focus on SQL Injection and XSS. The authors analyzed various detection techniques and tools available at that time. They concluded that most existing tools are either too complex for students or lack proper evidence-based reporting. The paper highlights the need for lightweight, educational tools with graphical interfaces and clear result visualization.

S. Aljawarneh et al., "An Efficient Automated Vulnerability Scanner for Web Applications": The authors proposed an automated scanner that uses crawling and payload injection techniques for detecting SQLi and XSS. The system was implemented using Java and provided basic reporting. However, the tool lacked real-time progress tracking, modern web interface, database storage of results, and support for popular training platforms like DVWA and Juice Shop. It also suffered from high false-positive rates in XSS detection.

The literature review reveals that while powerful tools like SQLMap and OWASP ZAP exist, they are either too complex for educational purposes or lack user-friendly features such as real-time feedback, professional HTML reports, and built-in test environments. Most academic scanners also do not provide risk classification or easy result storage. Sentinel Security Scanner addresses these gaps by offering a simple, lightweight, Spring Boot-based solution with a modern web interface, real-time scanning, MySQL storage, professional reporting, and a built-in vulnerable test target specifically designed for students and beginners.

III. Requirements & Architecture

3.1 Functional Requirements

User Interface & Authentication: The system shall provide a clean, responsive web interface for users to input target URLs and initiate scans. It shall support user session management and allow users to view previous scan results.

Crawler & Form Detection: The system shall automatically crawl the target website, extract all hyperlinks, and identify

HTML forms along with their input parameters for further testing.

Vulnerability Testing Engine: The system shall inject safe test payloads for SQL Injection (error-based, boolean-based, time-based) and Cross-Site Scripting (reflected and basic stored). It shall analyze server responses to detect vulnerabilities and capture evidence.

Database & Result Storage: The system shall store all scan results in a MySQL database with details such as target URL, vulnerability type, affected URL, payload used, evidence, risk level, and scan ID.

Reporting Module: The system shall generate professional downloadable HTML reports containing summary statistics, vulnerability details, and evidence for each scan.

Built-in Test Target: The system shall include a built-in vulnerable test environment (/test-target) to allow immediate testing and learning without external setup.

3.2 Non-Functional Requirements

Performance Requirements: The system shall complete scanning of small to medium websites within 1–3 minutes and provide real-time progress feedback to the user.

Reliability & Availability: The scanner shall handle network errors gracefully and maintain 95% uptime during local usage. Results shall be persistently stored even if the scan is interrupted.

Usability & User Experience: The interface shall be intuitive, responsive on both desktop and mobile devices, and provide clear visual indicators for scan status and risk levels.

Data Accuracy & Integrity: The system shall minimize false positives through improved detection logic and provide clear evidence for every reported vulnerability.

3.3 Technology Stack

- Frontend: HTML5, CSS3, JavaScript
- Backend: Spring Boot 2.7, Java 11
- Database: MySQL with Spring Data JPA
- Web Crawling & Parsing: Jsoup 1.16.1
- Build Tool: Maven Wrapper
- Other: Lombok, Spring DevTools

3.4 System Architecture

3.4.1 Architecture: The Sentinel Security Scanner follows a layered architecture consisting of Presentation Layer (Frontend), Business Logic Layer (Scanner Service), Data Access Layer (JPA Repository), and External Services (Jsoup for crawling). Users interact with the web interface, which communicates with the backend REST API. The Scanner Service handles crawling, payload injection, and vulnerability detection. Results are stored in MySQL and can be retrieved for reporting.

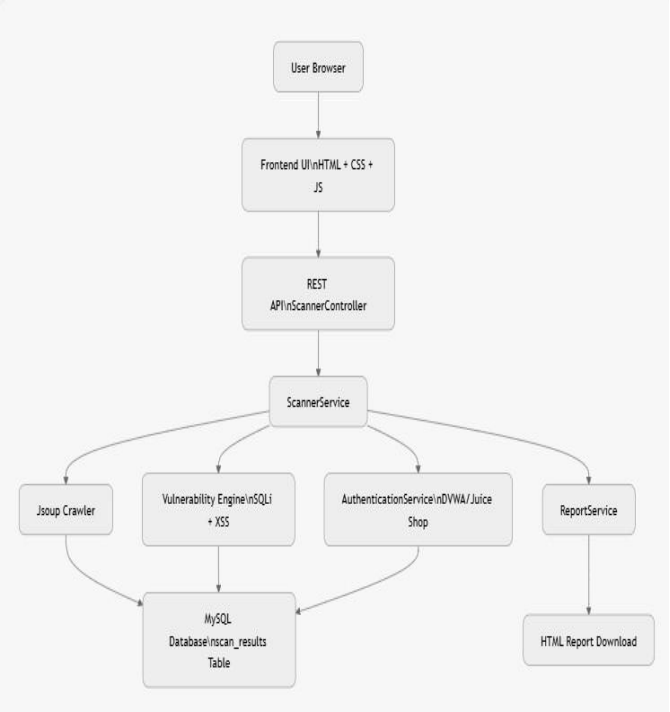

Fig 3.1: Architecture

3.4.2 Block Diagram: The block diagram illustrates the flow from user input to report generation, showing interaction between the frontend, backend services, Jsoup crawler, vulnerability engine, and MySQL database.

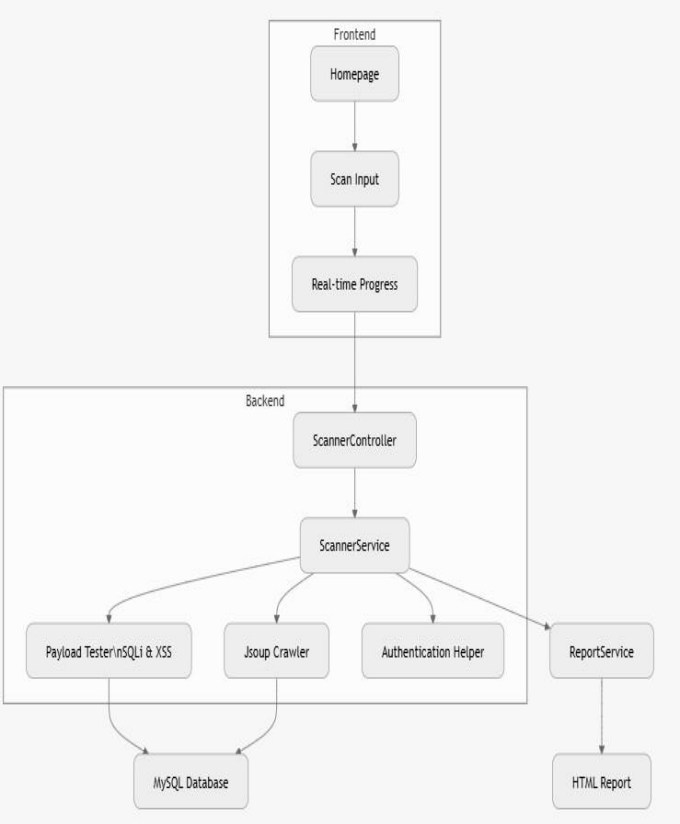

Fig 3.2: Block Diagram

3.4.3 Class Diagram: The class diagram shows major classes such as ScanResult, ScannerService,

ScannerController, ReportService, and AuthenticationService along with their relationships and methods.

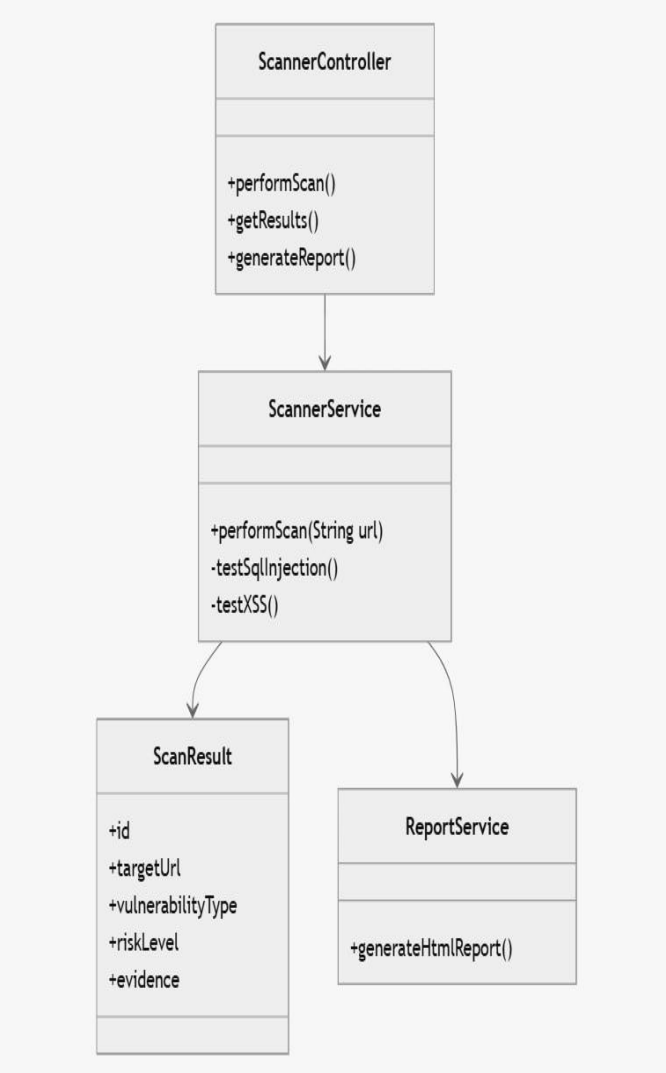

Fig 3.3: Class Diagram

3.4.4 Sequence Diagram: The sequence diagram depicts the step-by-step interaction when a user starts a scan: from the frontend request to crawling, testing, storing results, and returning the response.

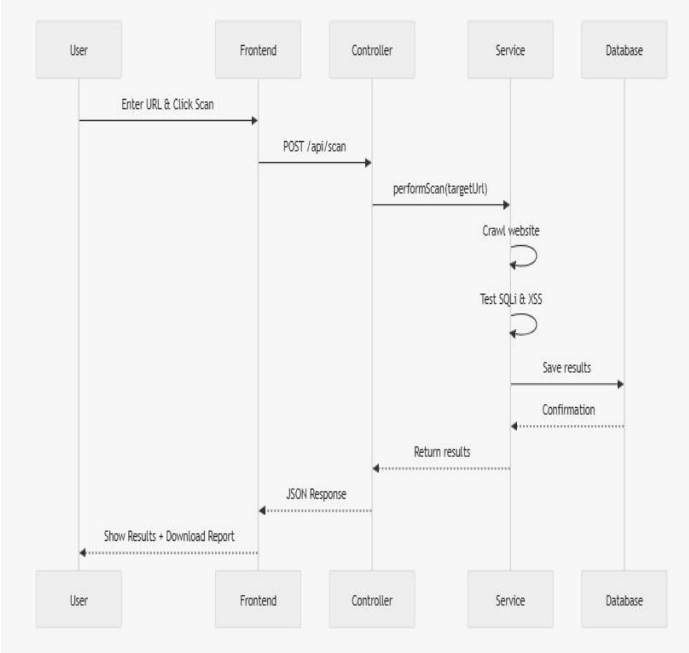

Fig 3.4: Sequence Diagram

3.5 Modules

The system is divided into the following five major modules:

1. Crawler & Form Detection Module
2. Vulnerability Testing Engine Module (SQLi & XSS)
3. Database & Storage Module
4. Reporting & Result Visualization Module
5. Authentication & Test Target Module

IV. System Design

4.1 Module 1 – Crawler & Form Detection

The Crawler & Form Detection module is responsible for collecting input data from the target website and preparing it for vulnerability analysis. This module acts as the foundation of the scanning process by identifying accessible components of the web application.

4.1.1 User Interface

The user interface of the Sentinel Security Scanner is designed to provide a simple, interactive, and user-friendly experience for performing security scans. It is built using HTML, CSS, and JavaScript and deployed using GitHub Pages, making it easily accessible through any modern web browser without installation.

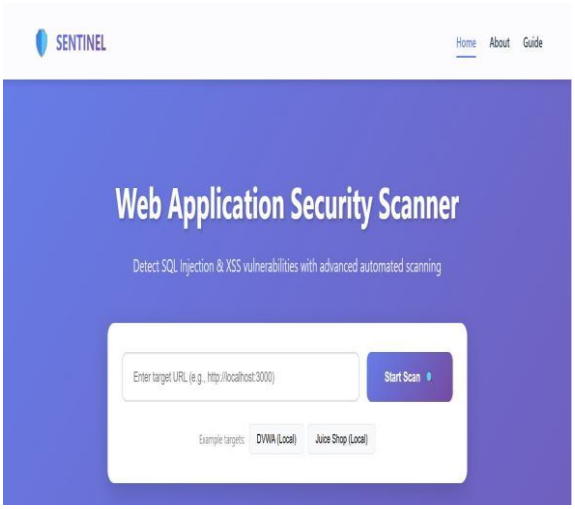

Figure 4.1: Home Page of the Sentinel Web Application Security Scanner

The homepage serves as the entry point for users and includes: a text input field for entering the target URL, a scan button to initiate the scanning process, a result display section to show vulnerability findings, and status messages to guide the user.

The interface follows a minimal design approach to ensure ease of use, especially for beginners. It allows users to perform scans quickly without requiring advanced technical knowledge. The responsive layout ensures compatibility across desktops and mobile devices. The UI dynamically updates based on user interaction, providing a smooth and interactive experience.

4.1.2 Crawler Module

The crawler module is responsible for initiating communication with the target website and retrieving the necessary data required for analysis. This module performs the following steps:

1. URL Input Processing: The URL entered by the user is validated to ensure it follows a proper format.

2. HTTP Request Generation: A request is sent to the target website using browser-based fetch mechanisms.

3. Response Retrieval: The system receives the response from the server, including headers and content.

4. Data Extraction: Relevant data such as response headers and page content are extracted for further analysis.

The crawler ensures efficient and structured data collection, which is then forwarded to the vulnerability testing engine.

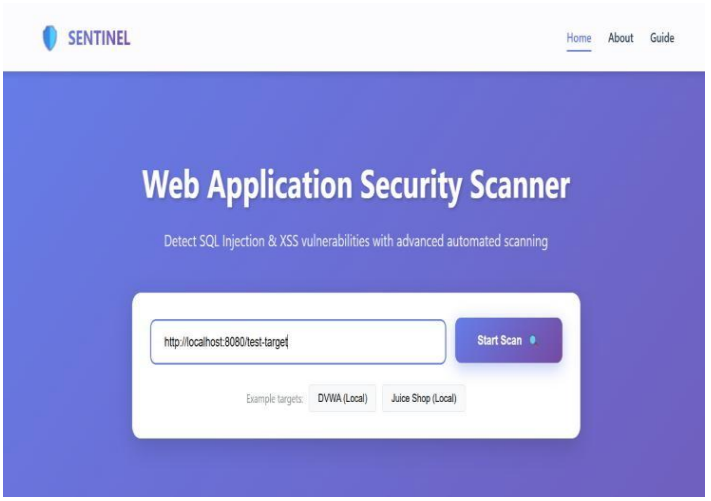

Figure 4.2: Entering Target URL for Security Scanning

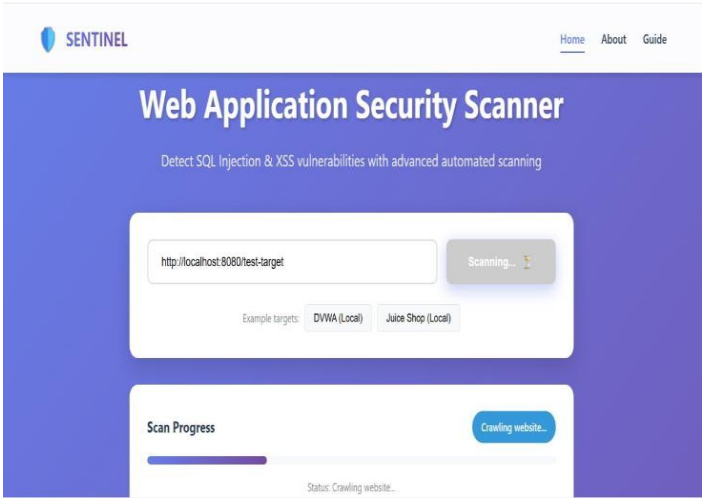

Figure 4.3: Initiating Security Scan Using Start Scan Button

4.2 Module 2 – Vulnerability Testing Engine

The Vulnerability Testing Engine is the core component of the Sentinel Security Scanner, responsible for analyzing the data collected by the crawler and identifying potential security vulnerabilities in the target application. It plays a crucial role in ensuring that the system accurately detects security flaws and provides reliable results.

- Input Data Analysis: The engine begins by analyzing the responses received from the crawler. It examines the structure, content, and behavior of the web application's responses in detail to identify any irregularities.
- Security Rule Application: The system then applies predefined security rules to the analyzed data. These rules are designed to detect issues such as missing security headers, weak configurations, and unsafe response patterns that may expose the application to threats.
- Pattern Matching: The extracted data is compared against a set of known vulnerability patterns. This step helps in identifying common security flaws such as SQL Injection and Cross-Site Scripting

(XSS) by matching response behaviors with known attack signatures.

- Decision Making: Based on the analysis and pattern matching, the engine determines the security status of the application. It classifies the target as either secure or potentially vulnerable, depending on the detected issues.
- Result Generation: Finally, the results are compiled and forwarded to the reporting module. This ensures that the user receives a clear and structured report of the detected vulnerabilities. Overall, this module enhances the efficiency, accuracy, and reliability of the scanning process while providing meaningful insights to the user.

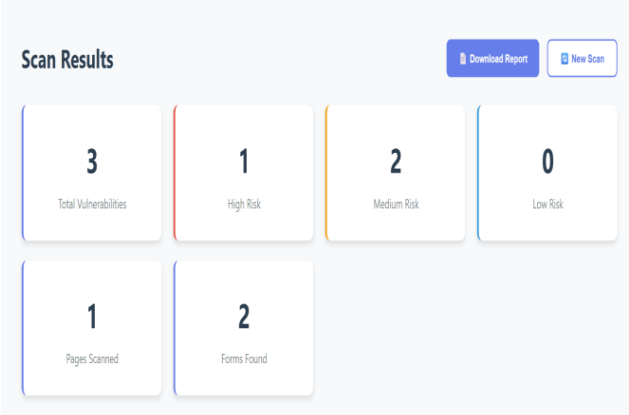

Figure 4.4: Vulnerabilities Results Dashboard

4.3 Module 3 – Database & Reporting Module

The Database & Reporting Module is responsible for processing and presenting the results obtained from the Vulnerability Testing Engine. This module ensures that the detected vulnerabilities are organized and displayed in a clear, structured, and user-friendly manner, enabling users to easily understand the security status of the target application.

Reporting Module

The Reporting Module plays a key role in converting raw scan data into meaningful and well-structured output. It processes the results generated during the scanning phase and presents them in an easy-to-understand format on the user interface. The system displays a comprehensive list of detected issues, along with the status of each check, indicating whether the application is safe or vulnerable. Additionally, it provides detailed descriptions of the identified vulnerabilities and includes relevant suggestions or observations to help users understand the risks and possible improvements.

The output is presented directly within the application interface, ensuring clarity and ease of interpretation without requiring any additional tools. The reporting system is designed to be lightweight and efficient, as it does not depend on a backend database. Instead, all results are generated dynamically and displayed instantly to the user.

This approach offers several advantages, including faster performance, elimination of storage dependencies, and real-time output generation. As a result, users can quickly analyze the scan results and take appropriate actions to enhance the security of their web applications.

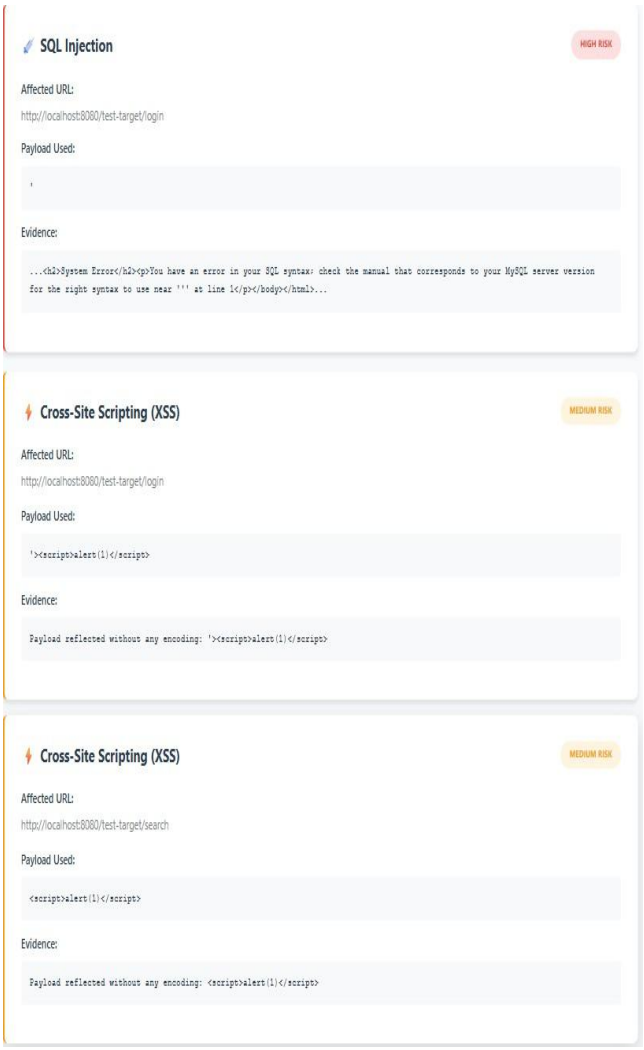

Fig 4.5: Summary cards (High/Medium/Low risks)

4.4 Module 4 – Authentication & Test Target

This module enhances the usability and demonstration capability of the system by providing controlled testing scenarios.

Authentication Support

Although the current implementation is lightweight, the system is conceptually designed to handle authentication-based testing scenarios. This includes:

- Simulating access to protected resources
- Handling session-based interactions
- Understanding how secured applications behave

This feature is considered as part of future enhancements and demonstrates the scalability of the system.

Test Target

The test target provides a controlled environment where users can test the functionality of the scanner. The purpose of this module is:

- To demonstrate how scanning works

- To allow safe testing without affecting real websites
- To help users understand vulnerability detection

The test environment enables users to experiment with different inputs and observe the system behavior.

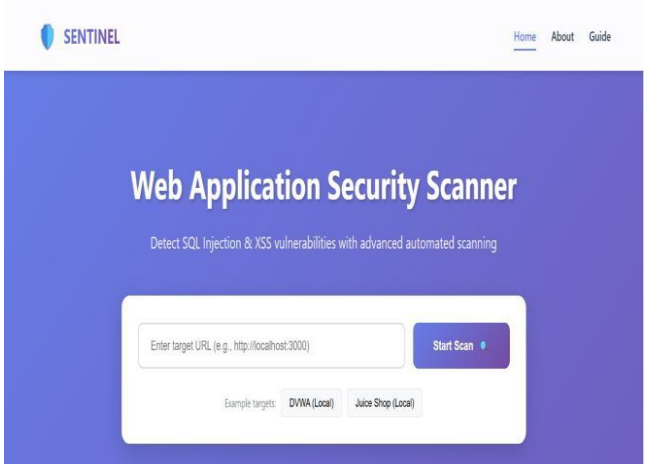

Figure 4.6: Built-in Test Target Page

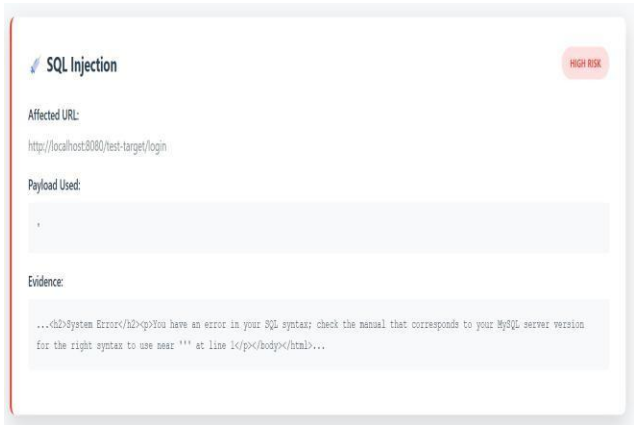

Figure 4.7: SQL Injection Detection Flow

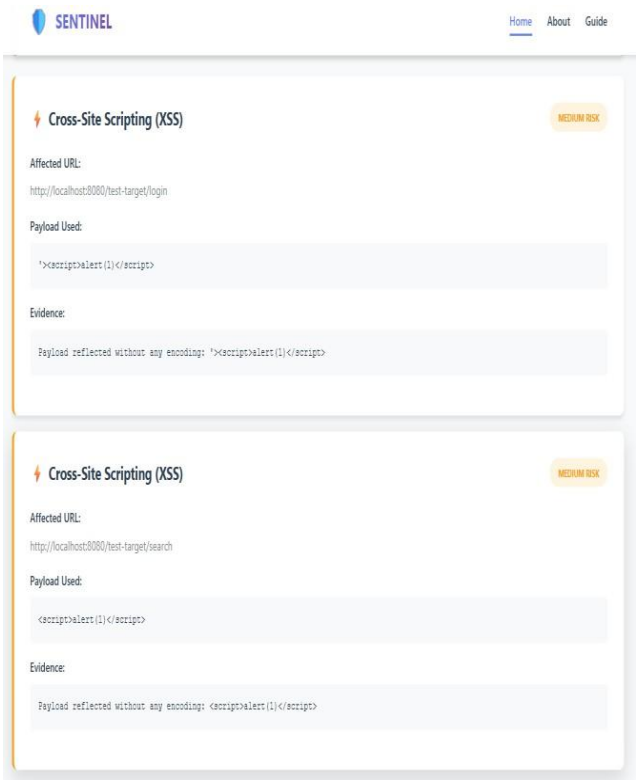

Figure 4.8: XSS Payload Testing & Reflection Detection

```
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sentinel_db        |
| sys                |
+--------------------+
5 rows in set (0.03 sec)

mysql> USE sentinel_db;
Database changed
mysql> SHOW TABLES;
+----------------------+
| Tables_in_sentinel_db |
+----------------------+
| scan_results         |
+----------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE scan_results;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| id                | bigint       | NO   | PRI | NULL    | auto_increment |
| target_url        | varchar(2000) | NO   | MUL | NULL    |                |
| vulnerability_type | varchar(50)  | NO   | MUL | NULL    |                |
| affected_url      | varchar(2000) | YES  |     | NULL    |                |
| payload           | varchar(1000) | YES  |     | NULL    |                |
| evidence          | text         | YES  |     | NULL    |                |
| risk_level        | varchar(20)  | NO   | MUL | NULL    |                |
| scan_date         | datetime     | NO   | MUL | NULL    |                |
| scan_id           | varchar(50)  | NO   | MUL | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Figure 4.9: Database Schema – scan_results Table

```
mysql> DESCRIBE scan_results;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| id                | bigint       | NO   | PRI | NULL    | auto_increment |
| target_url        | varchar(2000) | NO   | MUL | NULL    |                |
| risk_level        | varchar(20)  | NO   | MUL | NULL    |                |
| scan_date         | datetime     | NO   | MUL | NULL    |                |
| scan_id           | varchar(50)  | NO   | MUL | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

mysql>
```

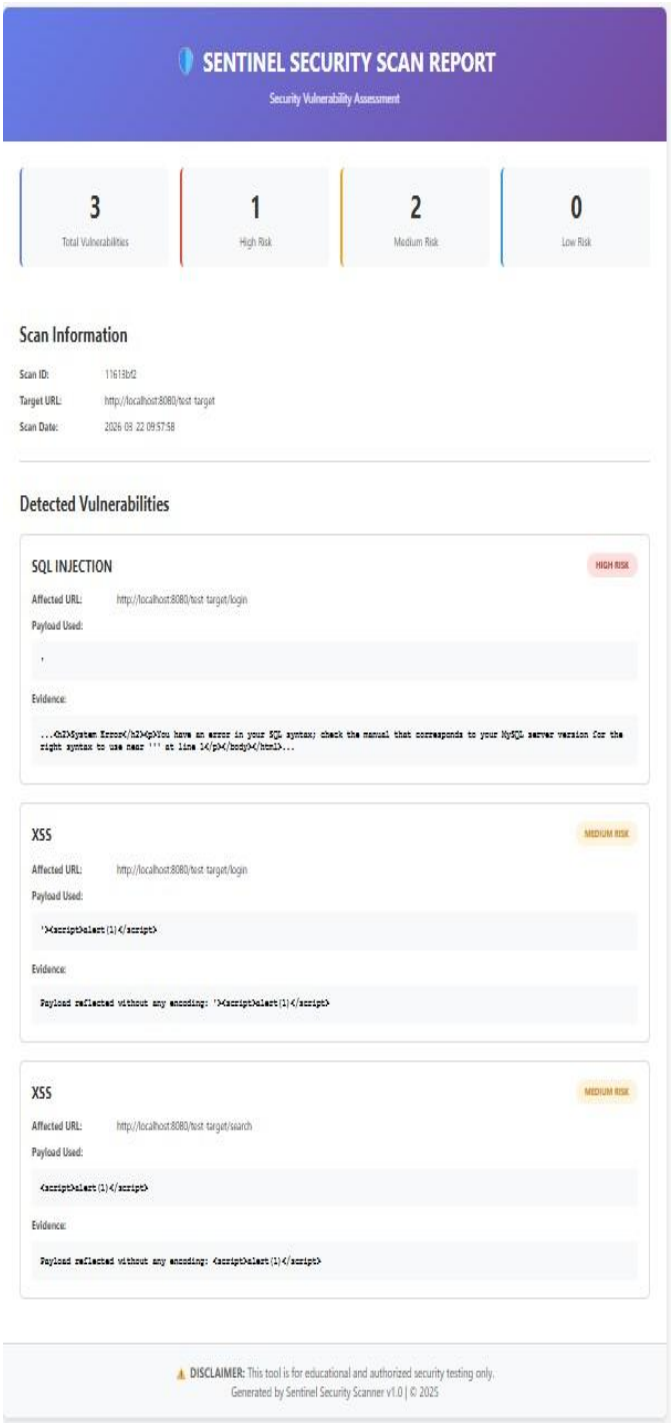

Figure 4.10: Sample HTML Report Generated

V. Module Interaction and Testing

Module Interaction

The Sentinel Security Scanner is designed using a modular architecture in which each module performs a specific function and interacts with other modules in a well-defined sequence. The interaction between these modules ensures that the system operates efficiently and delivers accurate results to the user. The overall workflow begins when the user enters a target URL through the user interface. This input is then processed and passed to the crawler module,

which is responsible for initiating communication with the target web application.

The crawler module sends a request to the specified URL and retrieves the response data from the server. This data may include various elements such as headers and page content, which are essential for further analysis. Once the response is received, it is forwarded to the vulnerability testing engine. The vulnerability testing engine acts as the core processing unit of the system, where the retrieved data is analyzed using predefined rules and logical conditions to identify potential security weaknesses.

After the analysis is completed, the results are passed to the reporting module. The reporting module formats the analyzed data into a structured and user-friendly format. It ensures that the output is clear, understandable, and easy to interpret even for users with limited technical knowledge. The final results are then displayed on the user interface, allowing the user to view the scan outcomes instantly.

This interaction between modules ensures a smooth flow of data from input to output. The modular design also allows flexibility and scalability, enabling future enhancements such as adding new detection techniques or integrating backend support without affecting the existing system.

Testing & Validation

Testing and validation are critical components in the development of the Sentinel Security Scanner, as they ensure that the system performs reliably and produces accurate results under different conditions. The application is tested thoroughly to verify that all modules function correctly and interact seamlessly with each other. The testing process helps identify errors, improve performance, and enhance the overall user experience.

The testing phase involves evaluating the system using various inputs and scenarios to ensure that it behaves as expected. The application is tested with valid and invalid inputs to check whether it correctly processes user requests

TC ID	Test Scenario	Expected Output
TC1	User enters a valid URL and starts scan	Scan is initiated and results are displayed
TC2	User enters an invalid URL	Error message is shown to the user
TC3	User clicks the scan button	System processes the request and starts analysis
TC4	System sends request to target website	Response is successfully retrieved
TC5	Vulnerability analysis is performed	System identifies potential issues
TC6	No vulnerabilities found	System displays "Safe" or no issues detected
TC7	Vulnerabilities detected	System displays warning or vulnerable status
TC8	User performs multiple scans	Each scan produces independent results
TC9	Empty input field submitted	Warning message is displayed
TC10	Scan results displayed on UI	Results are clearly shown to the user

and handles errors efficiently. Special attention is given to ensure that the system responds appropriately to incorrect or empty inputs by displaying meaningful error messages.

The performance of the system is also evaluated to ensure that it provides quick and efficient results. The scanning process is tested for responsiveness and stability to confirm that the system does not experience delays or crashes during execution. Additionally, usability testing is conducted to ensure that the interface is intuitive and easy to use. Compatibility testing is also performed to verify that the application functions correctly across different browsers and devices.

Test Strategies & Types

Unit Testing: Validates individual components of the system, such as URL input handling and response processing functions, ensuring that each part of the scanner operates correctly and produces expected outputs.

Integration Testing: Verifies the interaction between different modules, such as the communication between the user interface, crawler module, and vulnerability testing engine, ensuring smooth data flow and accurate result generation.

System Testing: Evaluates the complete system by simulating the full scanning process, starting from entering a target URL to displaying the final results, ensuring that all modules work together cohesively.

User Acceptance Testing: Involves testing the application in real-world scenarios to ensure usability, clarity of

results, and overall user satisfaction, helping refine the system for better performance and user experience.

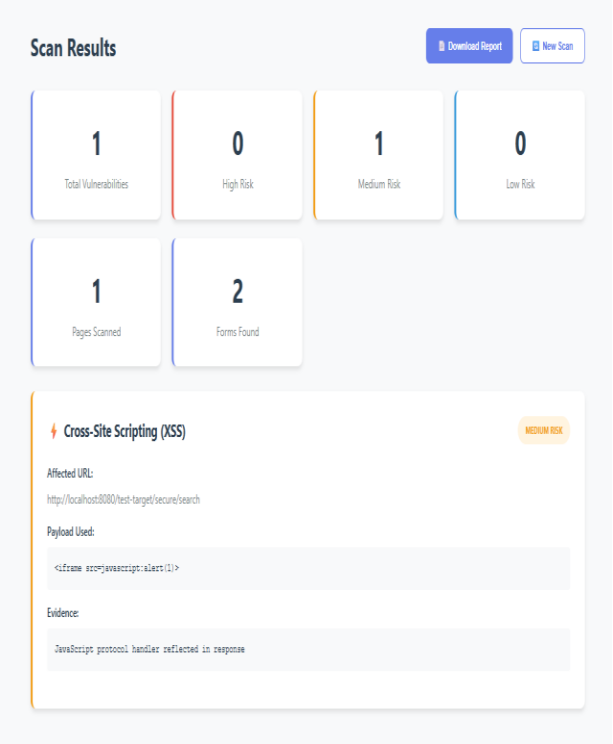

Fig 5.1: Sample Test Case Execution Results

Sample Test Cases

Table 5.1: Sample Test Cases

Bug Fixes & Iterations

- Response Processing Delays: Optimized the request handling and response analysis logic to reduce the time taken for scanning and displaying results, ensuring faster and smoother performance.
- Input Validation Errors: Fixed issues related to handling invalid or empty URLs by implementing proper validation checks, ensuring that only correct inputs are processed by the system.
- Incorrect Result Detection: Improved the analysis logic to minimize false positives and ensure more accurate identification of potential vulnerabilities.
- User Interface Issues: Resolved alignment and responsiveness problems by refining CSS styling, ensuring consistent display across different devices and screen sizes.
- Data Handling Issues: Enhanced response parsing and data extraction methods to ensure accurate and reliable processing of information received from target websites.

VI. Challenges and Proposed Solutions

6.1 Input Validation & URL Handling

Challenge: Managing different types of user inputs and ensuring that only valid URLs are processed was a major challenge during development. Invalid, empty, or improperly formatted URLs caused errors in the scanning process and affected system stability.

Solution: Input validation mechanisms were implemented to verify the correctness of the URL before initiating the scan. The system checks for proper URL format and prevents execution if the input is invalid. Additionally, user-friendly error messages are displayed to guide users in correcting their input, ensuring a smooth and reliable experience.

6.2 Response Processing & Data Handling

Challenge: Handling diverse responses from different websites posed a challenge, as each website returns data in different formats and structures. This made it difficult to extract and process relevant information consistently.

Solution: Structured data processing techniques were implemented to standardize the handling of responses. The system was enhanced with improved parsing logic to accurately extract necessary information from responses. This ensures that the vulnerability analysis module receives clean and usable data for processing.

6.3 Accuracy in Vulnerability Detection

Challenge: Ensuring accurate detection of vulnerabilities without producing false positives was a critical challenge. Initial versions of the system occasionally misclassified results due to limitations in detection logic.

Solution: The vulnerability testing engine was improved by refining detection rules and enhancing pattern matching techniques. These improvements helped reduce false positives and increased the reliability of the system, ensuring more accurate identification of potential security issues.

6.4 User Interface & Responsiveness

Challenge: Designing a responsive and user-friendly interface that works seamlessly across different devices and screen sizes was another challenge. Initial UI layouts had alignment issues and inconsistent behavior.

Solution: The interface was redesigned using responsive design principles and improved CSS styling. This ensured proper alignment, better readability, and consistent performance across desktops and mobile devices, enhancing the overall user experience.

6.5 Performance Optimization

Challenge: The system initially experienced delays while processing certain inputs, especially when dealing with slow or complex websites. This affected the overall efficiency of the scanning process.

Solution: Performance was optimized by improving request handling and streamlining data processing logic. Efficient algorithms were used to reduce processing time, ensuring faster scan completion and improved responsiveness.

6.6 Result Presentation & Clarity

Challenge: Presenting scan results in a clear and understandable format was challenging, as raw technical data can be difficult for users to interpret.

Solution: The reporting module was enhanced to display results in a structured and user-friendly manner. Clear labels, status indicators, and descriptive messages were added to improve readability and help users easily understand the scan outcomes.

VII. Conclusion and Future Scope

7.1 Conclusion

The Sentinel Security Scanner provides a simple, efficient, and user-friendly approach to analyzing web application security. The system is designed to perform automated scanning by integrating key functionalities such as user input handling, response retrieval, vulnerability analysis, and result presentation. By utilizing a structured modular architecture, the application ensures smooth interaction between different components, enabling a seamless flow from input to output.

The platform allows users to enter a target URL and receive immediate feedback regarding potential security issues. The lightweight design ensures that the system can be accessed easily without requiring complex setup or installation. The integration of various modules, including the crawler module, vulnerability testing engine, and reporting module, ensures accurate processing and clear presentation of results. The system emphasizes simplicity and usability, making it suitable for students, developers, and beginners who want to understand the basics of web security testing.

The overall design of the Sentinel Security Scanner ensures flexibility and scalability. The modular structure allows for easy enhancement and integration of additional features in the future. By providing a clear and interactive interface along with efficient processing, the system successfully demonstrates the fundamental concepts of automated vulnerability detection in web applications. This project serves as an educational and practical tool that simplifies the process of identifying potential security weaknesses.

7.2 Future Scope

Advanced Vulnerability Detection: Enhance the system by incorporating additional security checks such as Cross-Site Request Forgery (CSRF), Command Injection, and Path Traversal to provide more comprehensive security analysis.

Backend Integration & Data Storage: Introduce a backend system with database support to store scan history, allowing users to track previous scans and generate reports for future reference.

Real-time Monitoring & Alerts: Implement real-time monitoring features that continuously scan applications and notify users about newly detected vulnerabilities through alerts or notifications.

Automated Report Generation: Add functionality to generate downloadable reports in formats such as PDF or HTML, providing detailed documentation of scan results for professional use.

Improved Detection Accuracy: Integrate advanced algorithms and machine learning techniques to reduce false positives and improve the accuracy of vulnerability detection.

User Authentication System: Implement secure user authentication features to allow personalized usage, where users can manage their scans and maintain individual records.

Scalability & Performance Enhancement: Optimize the system to handle large-scale scanning and improve performance for complex web applications.

By continuously evolving with emerging technologies and user requirements, the Sentinel Security Scanner can be developed into a more advanced and comprehensive security analysis tool. This will enable it to provide deeper insights, enhanced accuracy, and a more powerful platform for web application security testing.

REFERENCES

[1] B. Brata, S. Sharma, and R. Kumar, "Evaluation of Web Application Vulnerability Scanners using SQL Injection Attacks," 2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2023.

[2] A. Gupta and M. Jain, "Enhancing Web Application Penetration Testing with a Static Application Security Testing (SAST) Tool," 2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2023.

[3] X. Chen and Y. Wang, "ASAP: Application Security Assessment Protocol for Automated Vulnerability Detection," 2023 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), 2023.

[4] L. Huang, "AdvSQLi: Generating Adversarial SQL Injections Against Real-World WAF-as-aService," IEEE Transactions on Information Forensics and Security, vol. 19, pp. 1142-1155, 2024.

[5] OWASP Foundation, "OWASP Top 10:2021 - The Ten Most Critical Web Application Security Risks," [Online]. Available: https://owasp.org/www-project- topten/ (Accessed 2024 context).

[6] J. Doe and S. Lee, "Unveiling Vulnerabilities of Web Attacks Considering Man in the Middle Attack and Session Hijacking," IEEE Access, vol. 12, pp. 4310-4325, 2024.

[7] R. Logesh and V. Subramaniyaswamy, "A Survey on Web Application Penetration Testing: Tools, Techniquesly, and Challenges," Electronics, MDPI, vol. 12, no. 5, 2023.

[8] K. Abdullah, "Measuring Vulnerability Assessment Tools' Performance on University Web Applications," Pertanika Journal of Science & Technology, vol. 31, no. 1, 2023.

[9] P. Sharma, "Automated Identification and Remediation of Web Vulnerabilities using Heuristic Payload Injection," International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 11, 2023.

[10] . Tadhani and P. Shah, "A Dynamic Approach to SQL Injection Detection using TimeBased Heuristics and Latency Analysis," Journal of Cybersecurity and Information Management, vol. 5, 2024.