APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) p. 1229-1249
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

WEBHUNTER: Web-Based Vulnerability Scanner System

Tan Bo Jan[1], Sofia Najwa Ramli[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: sofianajwa@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.066

Article Info

Received: 11 June 2026
Accepted: 23 June 2026
Available online: 30 June 2026

Keywords

WEBHUNTER, Vulnerability Scanner, Signature-Based Detection, SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), AI Assistant

Abstract

WEBHUNTER is a web-based vulnerability scanner that addresses the increasing demand for secure web applications in today's digital landscape. Existing automated tools such as OWASP ZAP, Intruder, and Invicti often lack user-friendliness, consume significant system resources, require complex configurations, and provide limited interactive support. To overcome these challenges, WEBHUNTER provides an automated, user-friendly, and lightweight vulnerability assessment system that uses signature-based detection to efficiently identify known attack patterns. WEBHUNTER automatically detects common web application vulnerabilities, including SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). The methodology chosen to develop WEBHUNTER is Agile, which allows developing the development in an iterative development, testing continuously, and making some improvements based on the user feedback. Through this system, web developers and security testers conduct vulnerability assessments in a more accessible, automated, and efficient way without requiring extensive security expertise. The implementation of WEBHUNTER helps organizations reduce the likelihood of cyberattacks, enhance web application security, and strengthen overall cybersecurity practices.

1. Introduction

In today's digital era, web applications have become essential tools for businesses, organizations, and individuals. However, they are also prime targets for cyberattacks. According to OWASP Top 10:2021 [1], web application vulnerabilities such as Injection (A03:2021), including SQL Injection (SQLi) and Cross-Site Scripting (XSS), and Broken Access Control (A01:2021), which includes threats like Cross-Site Request Forgery (CSRF), are among the most critical vulnerabilities affecting web applications. Although manual vulnerability assessments, such as code reviews, are effective, they are often time-consuming, prone to human error, and require specialized expertise [2]. Existing automated tools, such as OWASP ZAP, Intruder, and Invicti, improve efficiency but still require manual configuration and technical expertise to produce accurate results [3]. Moreover, these tools often face challenges such as lack of user-friendliness, lack of live support, high resource usage during vulnerability scanning, and complex setup processes [4].

To address these limitations, WEBHUNTER has been developed as a lightweight, automated, and user-friendly web-based vulnerability scanner. It can detect common web application vulnerabilities, including SQLi, XSS, and CSRF, while reducing the risk of human error. The system features an intuitive web interface and performs server-side scanning to minimize client resource usage. It generates comprehensive vulnerability reports that include CVE classifications, impact analysis, and actionable recommendations to assist users in mitigating identified threats. Additionally, the system integrates live support and a real-time AI assistant to help users efficiently resolve issues.

The primary objectives of this project are to design a web-based vulnerability scanner system that detects SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) vulnerabilities, to develop a web-based vulnerability scanner system using signature-based detection on a python web-based platform and to test the functionality, performance and security of the web-based vulnerability scanner system by web developers and security testers through system testing and user acceptance testing (UAT).

WEBHUNTER is developed using Django (Python) for the backend, HTML, CSS, and JavaScript for the frontend, and PostgreSQL as the database. The project adopts the Agile development methodology, which allows iterative development, continuous testing, and improvements driven by user feedback. The system serves two types of users, which are admin and user. The system modules include a login module, user management module, vulnerability scanning module, report generation module, live support and real-time AI assistant module and audit log module.

WEBHUNTER provides significant benefits to developers, organizations, and security testers. For developers, it saves time by automating vulnerability scans and generating easy-to-understand reports with CVE references, enabling faster identification and remediation of security issues. For organizations, the system helps prevent potential cyberattacks by proactively detecting vulnerabilities and supporting vulnerability management practices aligned with ISO/IEC 27001 requirements. Security testers can streamline the process of identifying and analyzing vulnerabilities across multiple web applications, improving the efficiency and effectiveness of security assessments. By automating vulnerability assessments, WEBHUNTER enables organizations to identify and mitigate security risks at an early stage, reducing the likelihood of data breaches, financial losses, and reputational damage. Furthermore, its lightweight server-side scanning approach and integrated AI support make the system accessible, efficient, and user-friendly, thereby enhancing the overall security posture of web applications.

2. Literature Review

This section will discuss SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Common Vulnerabilities and Exposures (CVE), web application vulnerability scanner, signature-based detection, reviewing of existing tools and the comparison between existing tools and the proposed system.

2.1 SQL Injection (SQLi)

SQLi is a significant web application vulnerability in which attackers inject malicious SQL commands into input fields to manipulate a website's database [5]. This attack happens when user inputs are not properly validated and are directly used within SQL queries. When successful, SQLi can lead to unauthorized data access, unlawful data manipulation, or even complete takeover of the database. WEBHUNTER will focus on identifying and mitigating SQLi vulnerabilities in web applications. By using automated scanning and leveraging the CVE database, WEBHUNTER will map identified SQLi vulnerabilities to their corresponding CVE identifiers. This integration allows for a standardized approach to tracking and managing vulnerabilities, ensuring that each identified risk is accurately categorized and prioritized based on its severity.

2.2 Cross-Site Scripting (XSS)

XSS is a vulnerability that enables attackers to inject malicious scripts into web pages, which are then executed in the browsers of unsuspecting users [6]. These scripts can be used to steal sensitive information such as cookies, session tokens, and login credentials. WEBHUNTER is designed to detect and mitigate XSS vulnerabilities in web applications. By using automated scanning and leveraging the CVE database, WEBHUNTER will map identified XSS vulnerabilities to their corresponding CVE identifiers. This integration allows for a standardized approach to tracking and managing vulnerabilities, ensuring that each identified risk is accurately categorized and prioritized based on its severity.

2.3 Cross-Site Request Forgery (CSRF)

CSRF is an attack that tricks a logged-in user into performing actions on a web application without their knowledge or consent [7]. This happens when an application does not properly validate the authenticity of a request. WEBHUNTER will focus on identifying and preventing CSRF vulnerabilities in web applications. By using automated scanning and leveraging the CVE database, WEBHUNTER will map identified CSRF vulnerabilities to their corresponding CVE identifiers. This integration allows for a standardized approach to tracking and managing vulnerabilities, ensuring that each identified risk is accurately categorized and prioritized based on its severity.

2.4 Common Vulnerabilities and Exposures (CVE)

Common Vulnerabilities and Exposures (CVE) is a publicly accessible database that provides standardized identifiers for known security vulnerabilities [8]. MITRE Corporation is responsible for managing the CVE system. It assists security professionals, researchers, and organizations in tracking and addressing vulnerabilities across multiple software and hardware platforms. Each vulnerability is assigned a unique CVE identifier, which allows security tools and professionals to reference and mitigate threats effectively [9]. WEBHUNTER will integrate CVE references into its scanning and reporting processes. By leveraging the CVE database, WEBHUNTER will map detected vulnerabilities to their corresponding CVE identifiers, providing users with an authoritative and standardized method for tracking and managing security risks. This integration enables WEBHUNTER to identify known vulnerabilities in web applications, allowing users to prioritize remediation efforts according to the severity of the vulnerabilities identified.

2.5 Web Application Vulnerability Scanner

Web application vulnerability scanner is specifically designed to detect security flaws in web applications, such as SQLi, XSS and CSRF [10]. These scanners analyze web applications by crawling their structure, testing input fields, and simulating attacks to identify vulnerabilities that malicious users could exploit. WEBHUNTER is focusing on automated detection of web vulnerabilities and categorizing them based on CVE standards. The system will generate comprehensive vulnerability reports, including CVE classifications, impact assessments, risk severity levels, and actionable recommendations to help users prioritize and mitigate identified threats effectively.

2.6 Signature-Based Detection

Signature-based detection works by identifying known patterns or signatures of malicious activity in web requests and responses [11]. This method relies on a database of predefined attack signatures, which are regularly updated to reflect emerging threats. If an incoming request corresponds to a stored signature, the system identifies it as a possible vulnerability or attack. In WEBHUNTER, signature-based detection will leverage the CVE database to enhance signature-based detection. By referencing the CVE identifiers associated with known vulnerabilities, WEBHUNTER can detect vulnerabilities based on established signatures, ensuring a more accurate and up-to-date identification process. This integration enables WEBHUNTER to identify known vulnerabilities in web applications and allows users to prioritize remediation based on the severity of the detected risks.

2.7 Reviewing of Existing Tools

OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner developed by OWASP and originally created in 2010 by Simon Bennetts as a fork of the Paros Proxy project [12]. It has since become one of the most widely used Dynamic Application Security Testing (DAST) tools. ZAP helps both beginners and security professionals detect vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and insecure deserialization by intercepting and analyzing HTTP/HTTPS traffic. It supports passive scanning for non-intrusive analysis and active scanning for deeper testing. Being free, cross-platform, and easy to deploy, ZAP is widely accessible for individuals, educational institutions, and organizations.

Intruder is a commercial web application vulnerability scanner developed by Intruder Security Ltd., founded by Chris Wallis in 2015 [13]. It helps organizations identify and fix security weaknesses in web applications and external-facing systems by offering comprehensive scanning combined with intelligent, risk-based prioritization. The tool detects a wide range of vulnerabilities including SQLi, XSS, Remote Code Execution (RCE), and security misconfigurations. It uses a scanning engine that is continuously updated with the latest threat intelligence. Intruder integrates seamlessly with cloud platforms and development pipelines, enabling automated scans during deployments or infrastructure changes. Its user-friendly dashboard provides detailed reports with CVE identifiers, severity levels, affected systems, and clear remediation guidance.

Invicti, formerly known as Netsparker, is a commercial web application security scanner developed by Invicti Security in 2009 [14]. It is widely recognized for its high accuracy and automation in identifying vulnerabilities across complex web applications, services, and APIs. The tool uses proprietary proof-based scanning technology, which safely verifies vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Remote Code Execution (RCE) by exploiting them in a controlled manner, significantly reducing false positives. Invicti supports both cloud and on-premises deployment and integrates seamlessly with CI/CD tools like Jenkins, GitHub, GitLab, Azure DevOps, and JIRA, making it suitable for DevSecOps workflows [15]. It provides detailed reporting aligned with standards such as OWASP Top Ten, PCI DSS, and ISO 27001, offering clear remediation steps for both technical and non-technical users.

2.8 Comparison Between Existing Tools and Proposed System

Table 1 shows the comparison between the proposed WEBHUNTER system and three existing systems include OWAPS ZAP, Intruder and Invicti. First, OWASP ZAP, Intruder, Invicti, and WEBHUNTER are web application vulnerability scanners. OWASP ZAP primarily targets issues such as compromised authentication, exposure of sensitive data, security misconfigurations, SQLi, XSS, and insecure deserialization. Intruder focuses on SQLi, XSS, Remote Code Execution (RCE), and security misconfigurations. Invicti focuses on SQLi, XSS, Command Injection, and Remote File Inclusion (RFI). WEBHUNTER focuses on SQLi, XSS and CSRF. Then, OWASP ZAP and WEBHUNTER are all open-source tools, meaning they are available for free. On the other hand, Intruder and Invicti are commercial products with a free trial, making them paid solutions for users looking for more enterprise-level features and support.

OWASP ZAP employs signature-based and heuristic-based detection, while Invicti utilizes signature-based, heuristic-based, and policy-based detection. Both Intruder and WEBHUNTER rely on signature-based detection. Additionally, WEBHUNTER and Intruder are notable for their web-based interfaces, allowing access from any device with a browser. This makes it more versatile compared to desktop applications like OWASP ZAP and Invicti, which require installation on specific operating systems such as Windows, macOS, and Linux. In terms of setup, OWASP ZAP, Intruder, and WEBHUNTER are relatively easy to set up, allowing users to quickly deploy the system with minimal configuration. However, Invicti requires a moderate setup, which may involve more complex configuration and system adjustments, making it more challenging to install and use for less experienced users.

Next, Invicti offers a graphical user interface (GUI), making it highly user-friendly for those who prefer visual interaction. OWASP ZAP supports both a GUI and a command-line interface (CLI), providing greater flexibility according to user preferences. On the other hand, Intruder and WEBHUNTER provide web-based interfaces, allowing users to access the tools through a browser. This offers convenience and ease of access without the need to install software, making it more accessible and adaptable across different platforms and devices. Additionally, WEBHUNTER provides live support through the integration of CometChat. WEBHUNTER also provides an AI assistant to offer users real-time assistance and improve the overall user experience by quickly resolving issues that arise. However, OWASP ZAP primarily relies on community forums for support, while Intruder provides support through an AI assistant, live chat, and email via its official website. Invicti provides support through email.

In terms of system resource consumption, OWASP ZAP is moderate in resource consumption, while Invicti is known for being resource intensive. This means that these tools require a lot of system resources to run vulnerability scans directly on the user's device, which can affect system performance during the scan. On the other hand, Intruder and WEBHUNTER have low system resource consumption because they run the vulnerability scans on the server side. This server-based scanning allows WEBHUNTER to perform comprehensive vulnerability assessments without burdening the user's device, making it ideal for users with limited computing resources. Furthermore, OWASP ZAP and WEBHUNTER are targeted at web developers and security testers, providing tools suitable for those working directly in web application development and security testing. Intruder and Invicti are targeted at web developers and enterprise security teams, providing advanced web application vulnerability scanning capabilities for larger-scale environments.

In terms of programming languages, OWASP ZAP is written in Java. The programming languages used for Intruder and Invicti are undisclosed, as the developers have not publicly provided this information. WEBHUNTER is built in Python, a more flexible and user-friendly language that can be modified, integrated, and enhanced more easily. Moreover, the security mechanism in OWASP ZAP is used for communication encryption, while the security mechanisms in Intruder are used for communication encryption, endpoint protection (anti-virus and anti-malware), full-disk encryption and two factor authentication. The security mechanisms in Invicti are used for same-origin policy (SOP), http strict transport security (HSTS), cookie security flags and role-based access control (RBAC), while the security mechanisms in WEBHUNTER are used for password hashing (bcrypt), two-factor authentication (2FA) with Gmail and password and email OTP, Google reCAPTCHA V3, rate limiting and account lockouts after multiple failed attempts.

Lastly, the modules in OWASP ZAP include vulnerability scanning, report generation and support (community forums), while the modules in Intruder include login, vulnerability scanning, report generation, and support (email, live chat, AI assistant). The modules in Invicti include login, vulnerability scanning, report generation, user management, and support (email), while the modules in WEBHUNTER include login, vulnerability scanning, report generation, user management, audit log, and live support (CometChat) and real-time AI assistant (OpenAI).

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1229-1249

Table 1 *Comparison Between Existing Tools and Proposed System*

Features	OWASP ZAP	Intruder	Invicti	WEBHUNTER
Types of Vulnerability Scanners	Web Application Vulnerability Scanner	Web Application Vulnerability Scanner	Web Application Vulnerability Scanner	Web Application Vulnerability Scanner
Primary Focus	Compromised authentication, exposure of sensitive data, security misconfiguration, SQL Injection (SQLi), Cross-Site Scripting (XSS), and insecure deserialization.	SQL Injection (SQLi), Cross-Site Scripting (XSS), Remote Code Execution (RCE), and security misconfigurations.	SQL Injection (SQLi), Cross-Site Scripting (XSS), Command Injection, and Remote File Inclusion (RFI).	SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF)
License	Open Source	Commercial	Commercial	Open Source
Types of Vulnerability Detection Techniques	Signature-Based Detection, Heuristic-Based Detection	Signature-Based Detection	Signature-Based Detection, Heuristic-Based Detection, Policy-Based Detection	Signature- Based Detection
Platform Support	Windows, macOS, Linux	Web-based	Windows, MacOS, Linux	Web-based
Ease of Setup	Easy	Easy	Moderate	Easy
User Interface	GUI and CLI	Web-based	GUI	Web-based
Types of Supports	Community Forums	AI Assistant, Email,Live Chat	Email	Live Chat (CometChat), Real-Time AI Assistant (OpenAI)
System Resource Consumption	Medium	Low	High	Low
Target Users	Web Developers Security Testers	Web Developers, Enterprise Security Teams	Web Developers, Enterprise Security Teams	Web Developers, Security Testers
Programming Language	Java	Undisclosed	Undisclosed	Python
Security Mechanisms	Communication Encryption	Communication Encryption, Endpoint Protection (Anti-Virus & Anti-Malware), Full-Disk Encryption, Two Factor Authentication	Same-Origin Policy (SOP), HTTP Strict Transport Security (HSTS), Cookie Security Flags, Role-Based Access Control (RBAC)	Password Hashing (Bcrypt), Two-Factor Authentication (2FA) with Gmail and Password and Email OTP, Google reCAPTCHA V3, Rate Limiting, Account Lockouts After Multiple Failed Attempts
Modules	Vulnerability Scanning, Report Generation, Support (Community Forums)	Login, Vulnerability Scanning, Report Generation, Support (Email, Live Chat, AI Assistant)	Login, Vulnerability Scanning, Report Generation, User Management, Support (Email)	Login, Vulnerability Scanning, Report Generation, User Management, Audit Log, Live Support (CometChat) and Real-Time AI Assistant (OpenAI)

3. Methodology

Agile methodology is an iterative and flexible approach to software development that emphasizes teamwork, continuous user feedback, and the rapid delivery of functional features. It breaks the development process into short cycles called sprints, allowing teams to respond quickly to changing requirements and enhance the system continuously [16]. This methodology was selected for the development of the WEBHUNTER web-based vulnerability scanning system due to its adaptability and overall efficiency. Fig. 1 illustrates the Agile methodology used in this project, which comprises six phases, including planning, design, development, testing, deployment, and review. Each sprint follows this structured workflow to ensure steady progress, support ongoing improvement, and strengthen the overall quality and security of the system.

Fig. 1 *Agile Methodology*

The planning phase serves as the foundation for the project. In this stage, the objectives, scope, functional and non-functional requirements, and system specifications are clearly defined, considering the security needs and expectations of WEBHUNTER. Interviews with web developers and security testers are conducted to gather accurate requirements and gain insights into the desired system features and capabilities. A Gantt Chart is created to organize tasks, set deadlines, and establish a clear timeline, allowing the project team to monitor progress effectively and ensure tasks are completed in a structured and timely manner.

The design phase focuses on transforming the collected requirements into a detailed blueprint system that guides development. UML diagrams are created to visually represent the system architecture, workflows, and interactions. The database is designed to ensure proper data organization and security. The user interface is designed using Canva, emphasizing a clean layout, intuitive navigation, and responsive elements for optimal usability across devices. User feedback is collected on the initial wireframes and incorporated into the final interface to enhance the overall user experience.

In the development phase, the WEBHUNTER system is implemented based on detailed design specifications. Django is used for backend development due to its strong security features and scalability. Python handles core backend functionalities such as user authentication, vulnerability scanning, and report generation. The front end is built using HTML, CSS, and JavaScript to provide a responsive and user-friendly interface. PostgreSQL serves as a relational database to efficiently manage user data, scan results, and system logs. Additional features such as CometChat for real-time communication and an AI assistant powered by OpenAI are integrated to improve user support. Key security features including bcrypt password hashing, Two-Factor Authentication (2FA) via Gmail and email OTP, Google reCAPTCHA, rate limiting, and account lockouts after multiple failed login attempts are implemented. Development follows iterative sprint cycles, allowing continuous testing, debugging, and refinement.

The testing phase evaluates the system's functionality, security, and overall performance. The test plan is executed to ensure each component works as intended. System testing assesses the overall functionality and security of WEBHUNTER, while UAT ensures the system meets user requirements and expectations. Continuous debugging is performed to identify and resolve issues promptly. This iterative testing approach ensures that vulnerabilities, usability issues, and performance concerns are addressed before deployment, guaranteeing a robust and reliable system.

The deployment phase involves launching WEBHUNTER into a live production environment. The system is deployed on a secure server with all backend, database, and frontend components properly configured. Security hardening measures, including HTTPS implementation and the disabling of unnecessary services, are applied to

reduce the system's attack surface. Final validation and live testing ensure that critical features such as authentication, vulnerability scanning, live chat, and the AI assistant function correctly in the real-world environment. Any issues discovered during this phase are resolved promptly, ensuring the system is stable, secure, and ready for end-users.

Finally, the review phase focuses on post-deployment evaluation. Feedback is collected from end-users to assess whether the system meets predefined objectives, functional requirements, and user expectations. Continuous monitoring of system performance and regular review of server logs help detect unusual activity or potential performance issues. Based on collected feedback and monitoring results, enhancements or new features are identified for future updates. This phase ensures that WEBHUNTER remains secure, reliable, and aligned with user needs while promoting continuous improvement even after deployment.

4. Analysis and Design

This section presents the system requirements analysis and design for WEBHUNTER, including the flowchart of the signature-based vulnerability scan process, use case diagram, sequence diagram for scanning and report generation, activity diagram, and class diagram.

4.1 Flowchart of the Signature-Based Vulnerability Scan Process

Fig. 2 illustrates the flowchart of the signature-based vulnerability scan process implemented in WEBHUNTER. This process allows both administrators and users to perform automated vulnerability scans on selected target web applications. The process begins when the administrator or user selects the vulnerability scan page. The admin or user is required to enter the target name, target URL, and a description of the scan. After that, the system validates the entered URL to ensure it is in the correct format and accessible. If the URL is invalid, the system will display an error message and return to the vulnerability scan page. If the URL is valid, the system proceeds to perform the vulnerability detection process.

During the vulnerability detection stage, three detection modules, including SQL injection detection, XSS detection, and CSRF detection, are executed simultaneously to reduce scanning time and improve efficiency. In the SQL injection detection module, the system runs an SQL injection script that injects various SQL payloads into the target URL to test possible injection vulnerabilities. The responses received from the server are compared with known SQL error patterns. For example, the scanner may test https://example.com/product?id=123 with the payload id=123' OR '1'='1. If the server returns a response containing the string "You have an error in your SQL syntax," the system identifies it as a potential SQL injection vulnerability and saves the result in the database. If no match is found, the system checks whether all SQL injection scripts have been executed. If not, it continues to run the remaining script until completion. Once all scripts have been executed, the system proceeds to display the scan results.

In the XSS detection module, the system runs an XSS injection script that inserts XSS payloads into the target URL to detect reflected XSS vulnerabilities. The responses are then analyzed and matched against known XSS reflection patterns. For example, the scanner may test https://example.com/search?q=shoes with the payload q=<script>alert(1)</script>. If the server returns a body containing " Search results for <script>alert(1)</script> ", the system identifies it as a potential reflected XSS vulnerability and saves the result in the database. If no match is found, the system checks whether all XSS scripts have been executed. If there are remaining scripts, the process continues until all have been completed, after which the scan results are displayed.

For the CSRF detection module, the system executes a CSRF detection script to determine whether the target web application implements proper CSRF protection mechanisms. The script first analyzes web forms to check for CSRF token fields or related protection patterns. Then, it performs active verification by submitting test requests such as removing the token, using an invalid token, or sending forged Origin and Referer headers and observing the server's responses. For example, the scanner may test a form at https://example.com/update-profile by submitting the form without a CSRF token. If the server responds with a 403 Forbidden status or an error message indicating a missing or invalid token, the system concludes that CSRF protection is in place. However, if the application accepts the request and updates the profile without validating the token, it is identified as vulnerable, and the result is stored in the database. This process continues until all detection scripts have been executed, after which the system displays the overall vulnerability scan results.

After the vulnerability detection process is complete, both administrators and users can download the vulnerability scan report. This report includes detailed information such as the vulnerability name, CVE reference number, CVSS base score, mitigation suggestions, and impact analysis. This comprehensive process ensures that WEBHUNTER provides accurate and efficient detection of common web vulnerabilities through a signature-based scanning approach.

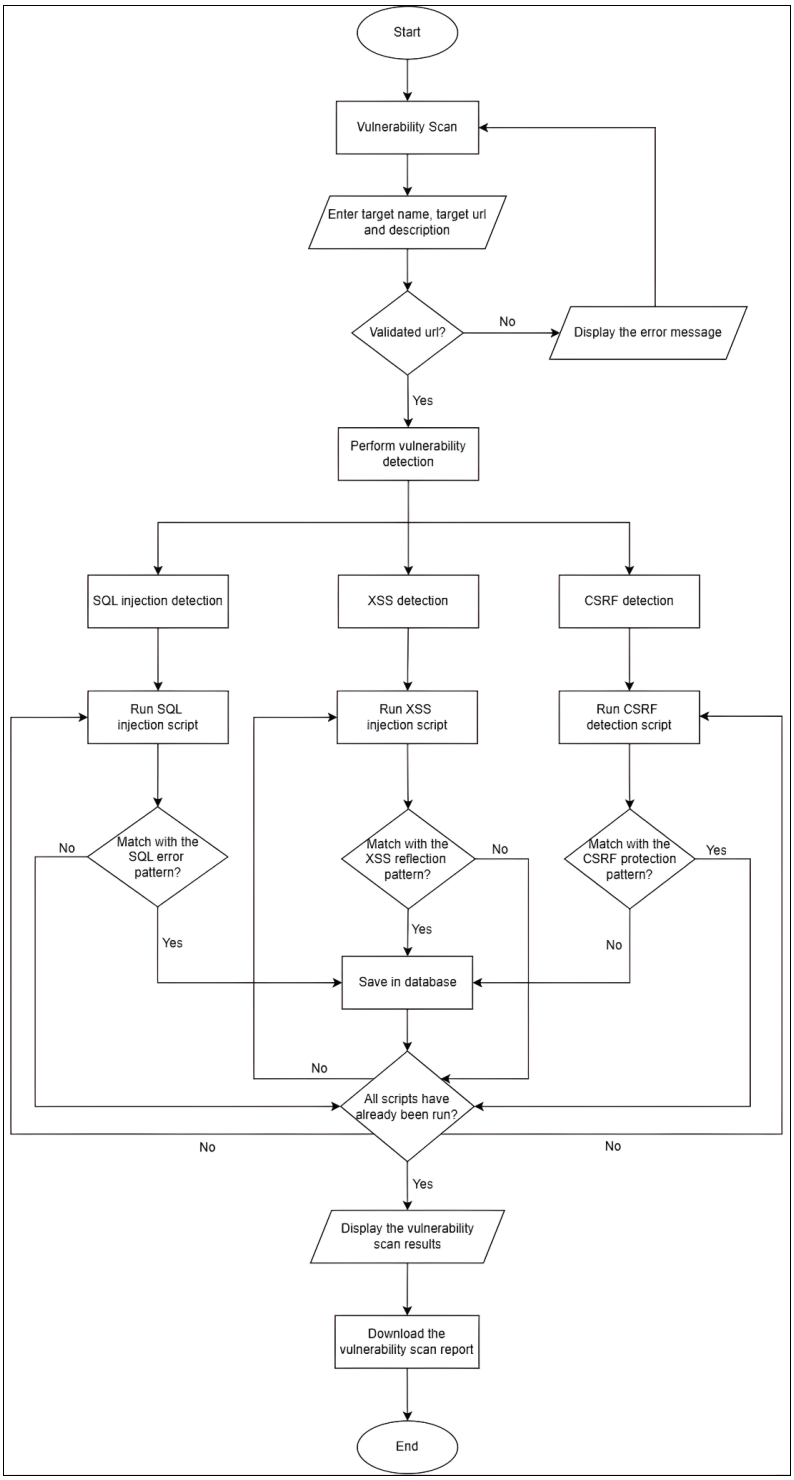


Fig. 2 *Flowchart of the Signature-Based Vulnerability Scan Process*

4.2 Use Case Diagram

A use case diagram is a visual representation that describes the interactions between users and the system to achieve specific goals. Fig. 3 shows use case diagram for WEBHUNTER. There are two types of users in the system, which are admins and users. Each type of user is associated with specific system use cases. Users are involved in login, vulnerability scanning, viewing vulnerability scan results, downloading vulnerability scan reports, live support and AI assistant. Then, admin is involved in login, user management, vulnerability scanning,

viewing statistics on the number of registered users, visitors, and vulnerability scans, viewing vulnerability scan results, downloading vulnerability scan reports, live support, audit logs and AI assistant.

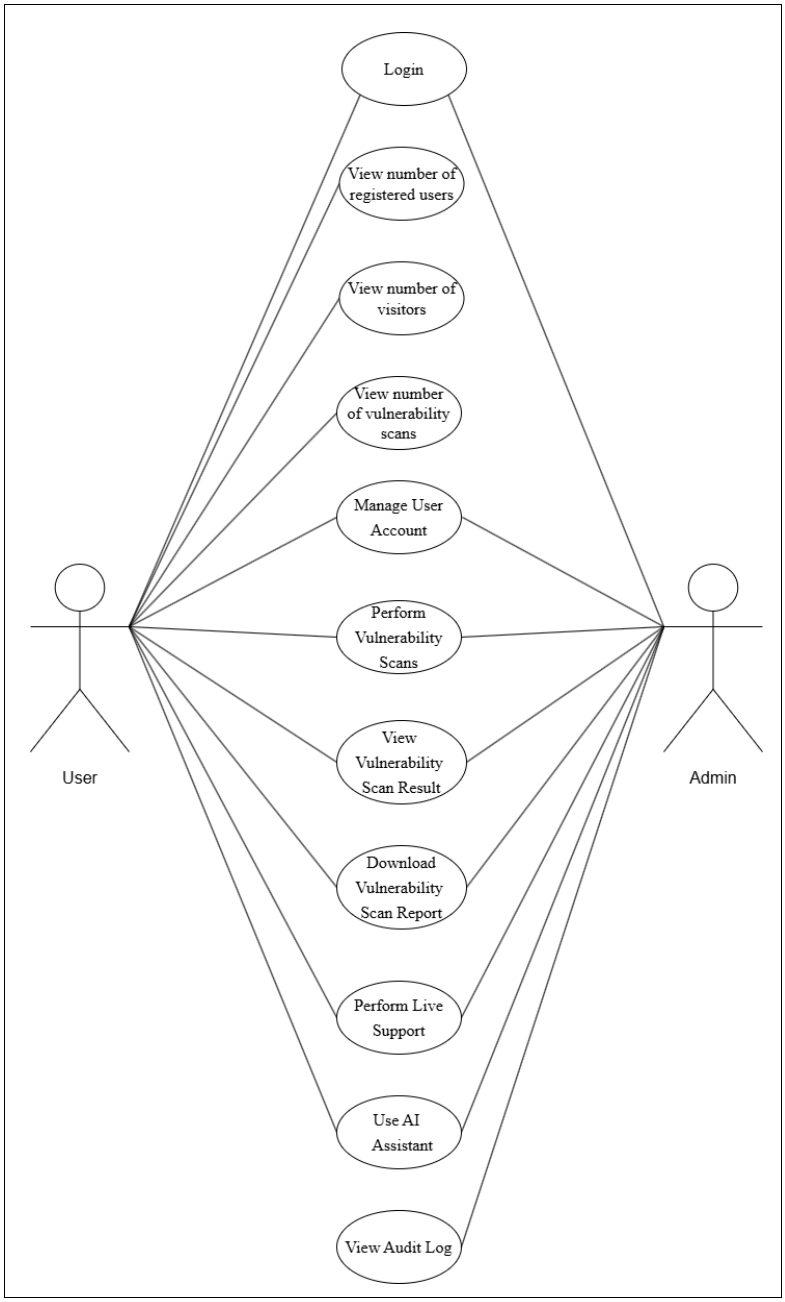


Fig. 3 *Use Case Diagram*

4.3 Sequence Diagram for Vulnerability Scanning and Report Generation

Fig. 4 shows the sequence diagram for vulnerability scanning and report generation in WEBHUNTER. The diagram includes four key objects, which are vulnerability scanning, vulnerability detection, report generation and database. The process begins when the user enters the target name, target URL, and description on the vulnerability scanning page. The system verifies the validity of the target URL. If the target URL is invalid, a verification failure message is sent to the user. If the target URL is valid, the database saves the target name, target URL, and description. Next, the system sends the target URL to the vulnerability detection module, which performs the vulnerability detection process. This process includes SQLi Detection, XSS Detection, and CSRF Detection. If any vulnerabilities are detected, the database saves the corresponding results. After the detection process, the user can request to view and download the scan report. The system retrieves the target information and scan results from the database and then automatically generates the scan report. Finally, the system sends the generated scan report to the user.

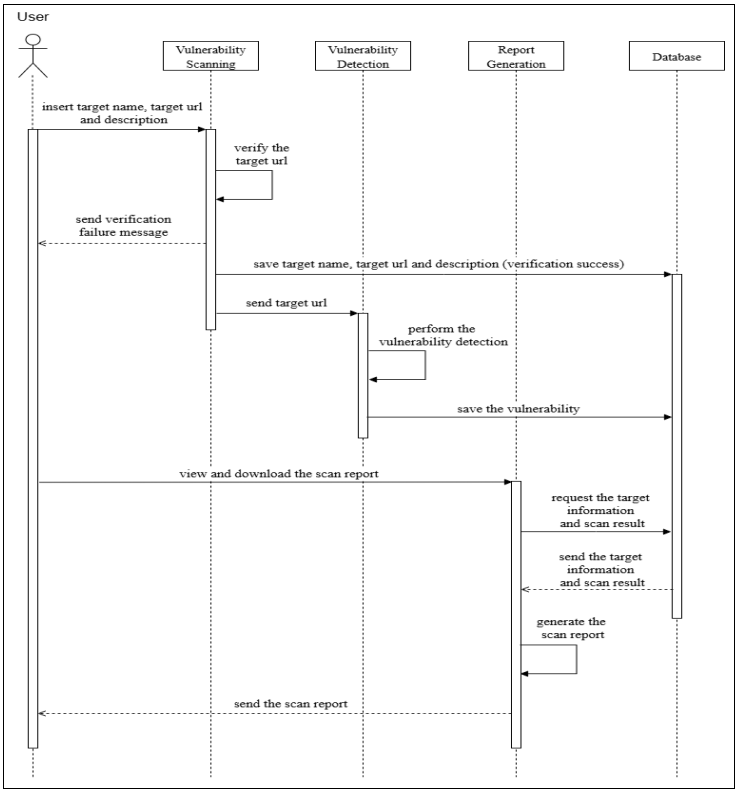


Fig. 4 *Sequence Diagram for Vulnerability Scanning and Report Generation*

4.4 Activity Diagram

An activity diagram is a flowchart that illustrates the steps involved in a process or workflow within a system. Fig. 5 shows the activity diagram for WEBHUNTER. The process begins when the user accesses the system, which first displays the login page. The system will verify the user's credentials. If the credentials are correct, the user is redirected to the home page. If the credentials are incorrect, the process returns to the login page. Then, the user can enter the target name, target URL, and description to initiate a vulnerability scan. After clicking submit, the system validates the target URL. If the URL is invalid, the process returns to the home page. Once the target is validated, the user can proceed to select the scan and click the view button to display the scan results. The user can also download the scan report by clicking the download button. If the user requires assistance, they can access the live support feature, enter a message, and click send. The message will be sent to the admin, and once the admin replies, the user will receive the response through the live chat interface. Lastly, users can access the AI Assistant page to receive real-time assistance. They can enter a message and click send, and the AI will respond to the message immediately.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1229-1249

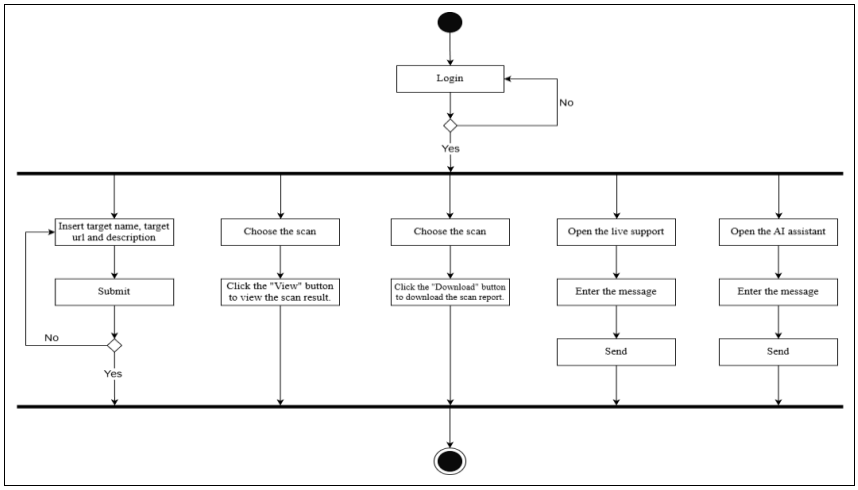


Fig. 5 *Activity Diagram*

4.5 Class Diagram

Fig. 6 shows the class diagram for the WEBHUNTER, which includes classes such as admin, user, otp, live support, audit log, scan, vulnerability and cve.

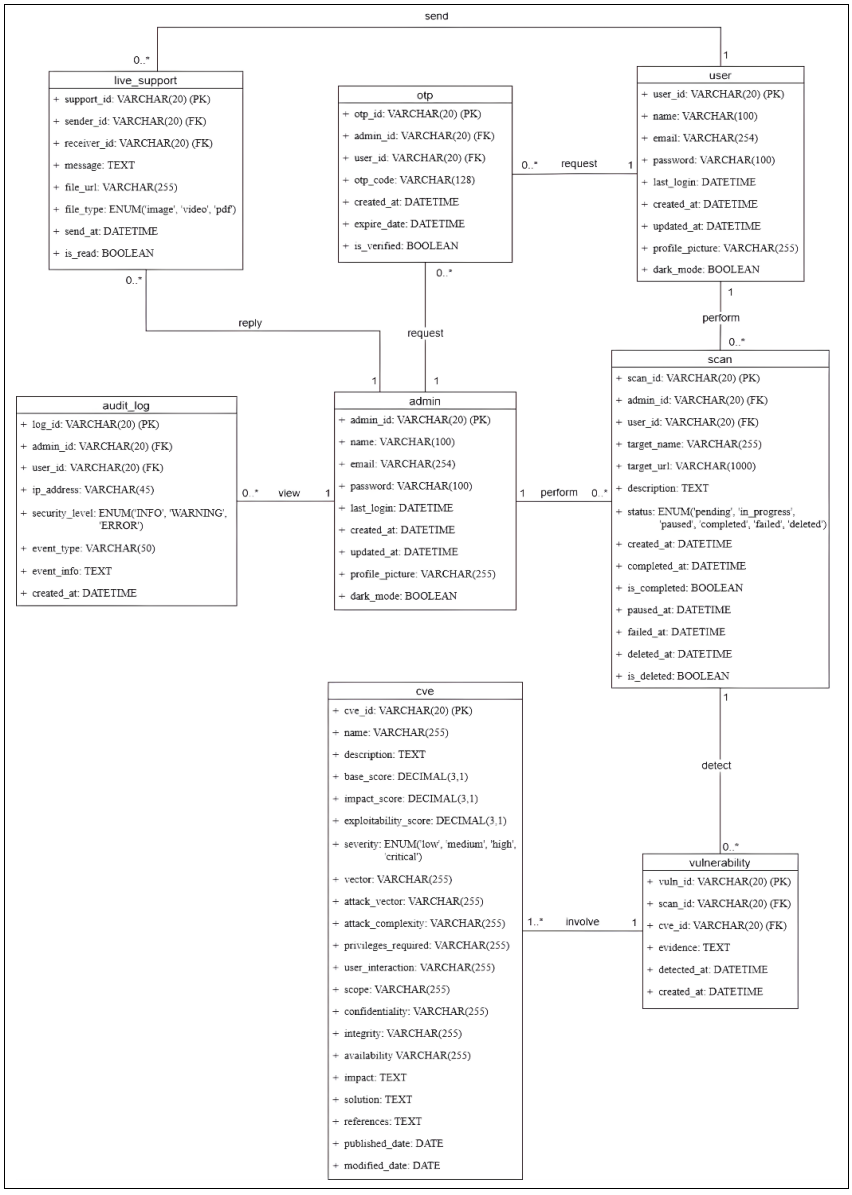


Fig. 6 *Class Diagram*

5. Implementation and Testing

This section will discuss the implementation and testing of WEBHUNTER.

5.1 Implementation

Fig. 7 shows the interface of the login page for WEBHUNTER. Users or administrators must enter their email and password to request the OTP. WEBHUNTER implements strong security mechanisms, including password hashing (bcrypt), Two-Factor Authentication (2FA) with Gmail and password and email OTP, Google reCAPTCHA v3, rate limiting, and account lockouts after multiple failed attempts. These features help protect the system from unauthorized access and brute-force attacks. Then, Fig. 8 shows the interface of the Two-Factor Authentication page for WEBHUNTER. Users or administrators are required to enter the OTP sent to verify their identity and gain access to the system. The OTP consists of a 6-digit code sent to the user's Gmail account and will expire within 5 minutes for enhanced security.

Fig. 7 *Interface of Login Page*

Fig. 8 *Interface of Two-Factor Authentication Page*

Fig. 9 shows the interface of the home page for WEBHUNTER. Admins can view the number of registered users, visitors and vulnerability scans, along with these statistics displayed in a line chart. Admins can also download the statistics in PDF format.

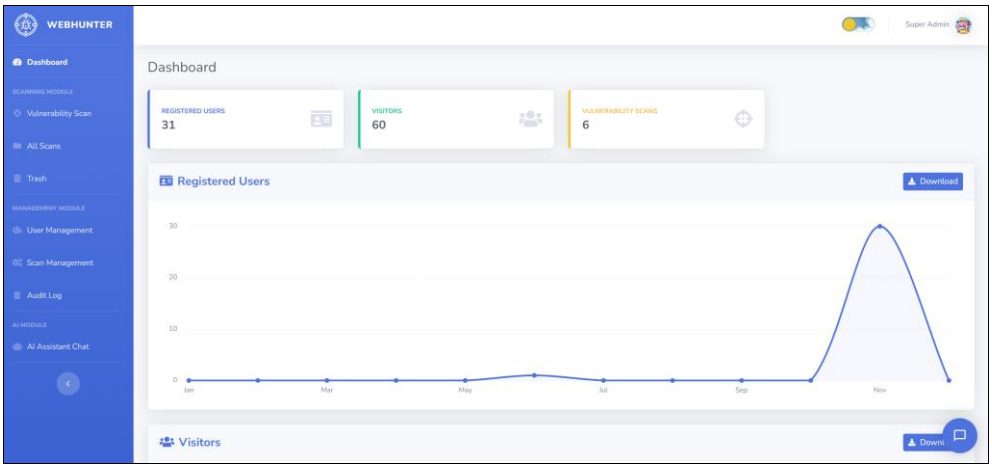


Fig. 9 *Interface of Home Page*

Fig. 10 shows the interface of the vulnerability scan page for WEBHUNTER. Admins can enter a target name, target URL, and description to perform a vulnerability scan. The target name and target URL fields are required. The system will validate the target URL, and if the URL is valid, it will display a success message indicating that the scan has been initiated. Otherwise, it will show an error message.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1229-1249

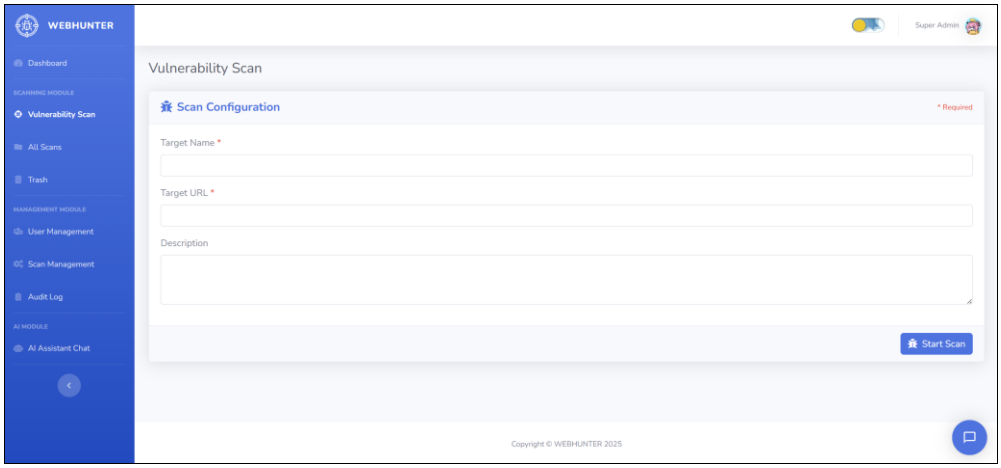


Fig. 10 *Interface of Vulnerability Scan Page*

Fig. 11 shows the interface of the all-scan page for WEBHUNTER. Admins can view the entire history of vulnerability scans. Once a scan is completed, the system will automatically generate the scan report and admins can download the scan report.

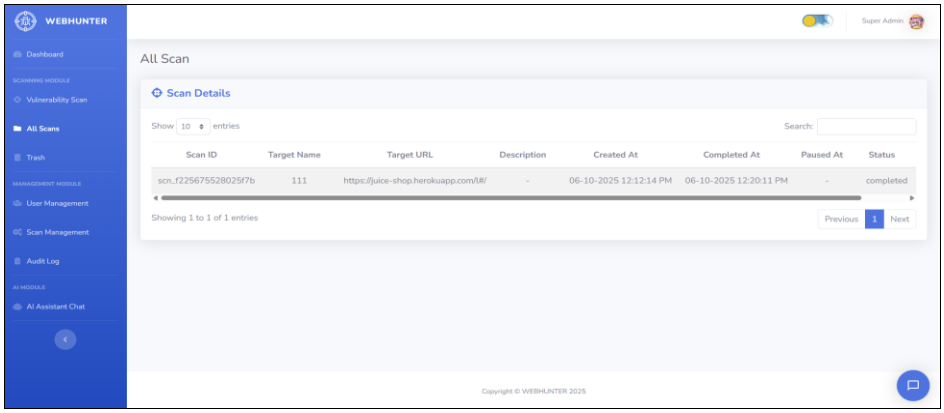


Fig. 11 *Interface of All Scan Page*

Fig. 12 shows the interface of the scan result page for WEBHUNTER. Admins can view detailed scan results, including the vulnerability ID, severity, base score, CVE ID, name, and other related information. On the right side, admins can also view scan details such as the scan ID, target name, target URL, description, start and end date and time, as well as a pie chart showing the severity distribution of the detected vulnerabilities.

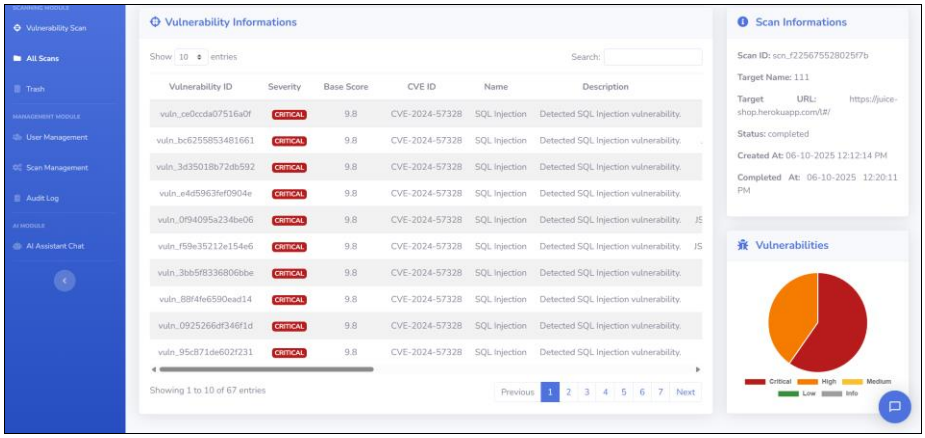


Fig. 12 *Interface of Scan Result Page*

Fig. 13 shows the interface of the vulnerability details page for WEBHUNTER. Admins can view detailed information about detected vulnerabilities, including severity, base score, CVE ID, name, description, impact, solution, evidence and other related information.

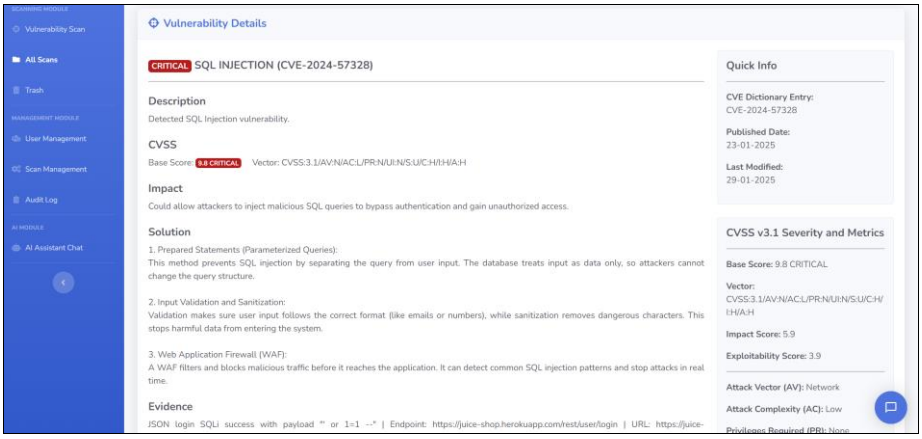


Fig. 13 *Interface of Vulnerability Details Page*

Fig. 14 shows the interface of the scan details page for the scan report. On the scan details page, it displays the total number of detected vulnerabilities based on their severity, along with the scan ID, target name, target URL, and other related information. Next, Fig. 15 shows the interface of the vulnerability details page for the scan report. On the vulnerability details page, it will show detailed information about detected vulnerabilities, including severity, base score, CVE ID, name, description, impact, solution, evidence and other related information.

Fig. 14 *Scan Details Page*

Fig. 15 *Vulnerability Details Page*

Fig. 16 shows the interface of the trash page for WEBHUNTER. Admins can view the entire trash history of vulnerability scans and available actions such as “Restore” and "Delete" the scan.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1229-1249

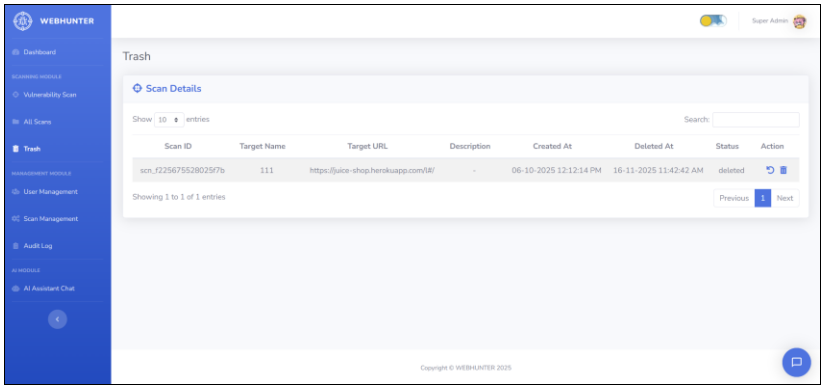


Fig. 16 *Interface of Trash Page*

Fig. 17 shows the interface of the user management page for WEBHUNTER. Admins can create, view, update, and delete user accounts.

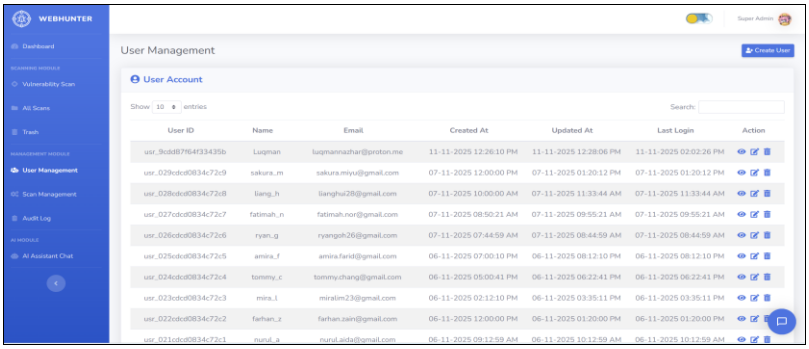


Fig. 17 *Interface of User Management Page*

Fig. 18 shows the interface of the scan management page for WEBHUNTER. Admins can view, download, and delete all users' vulnerability scans.

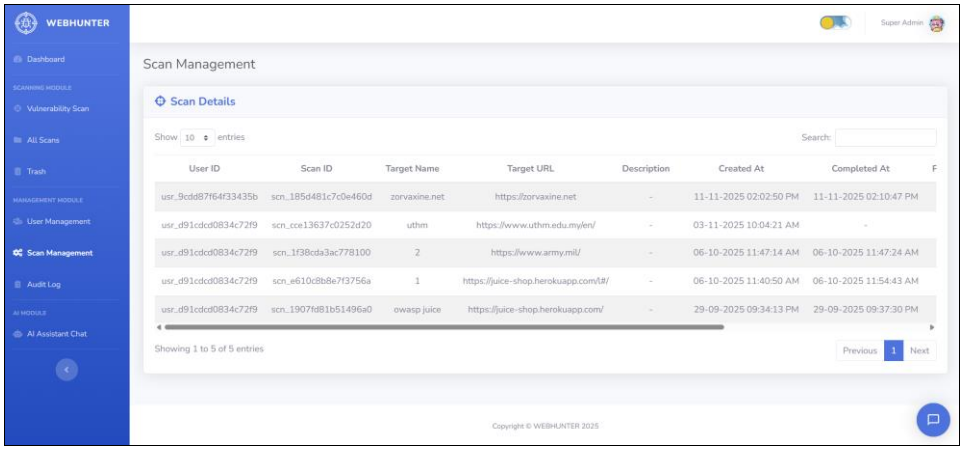


Fig. 18 *Interface of Scan Management Page*

Fig. 19 shows the interface of the audit log page for WEBHUNTER. Admins can view and track all user activities, including IP address, level, event type, event detail, and other related information. This model enables admins to detect suspicious activity and take appropriate action to ensure the security and integrity of the system.

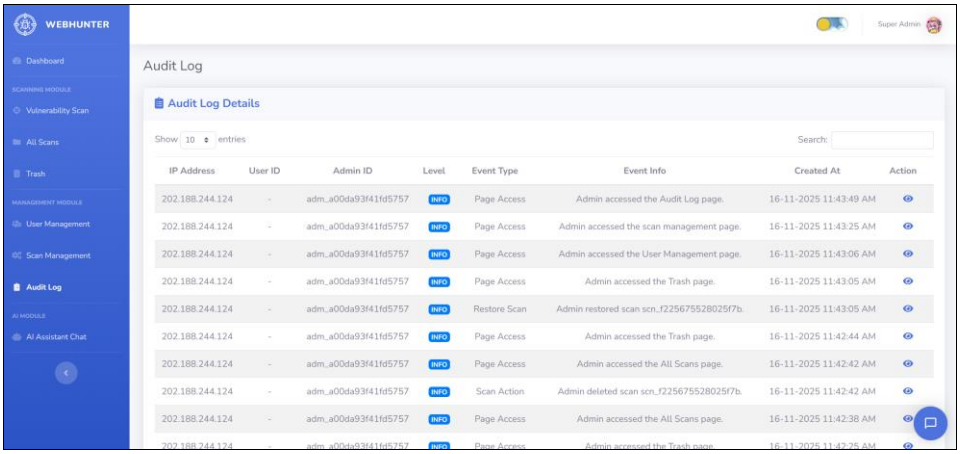


Fig. 19 *Interface of Audit Log Page*

Fig. 20 shows the interface of AI assistant page for WEBHUNTER. Admins can ask questions to the AI, and it will respond immediately.

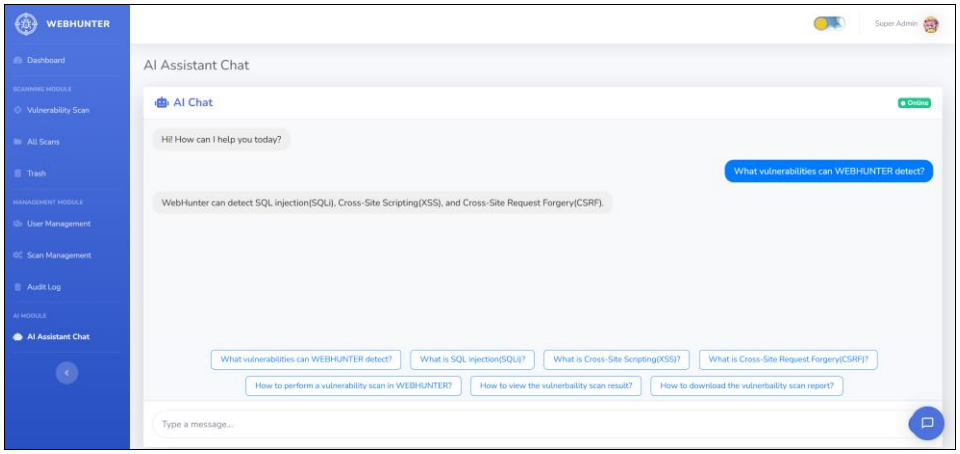


Fig. 20 *Interface of AI Assistant Page*

Fig. 21 shows the interface of the live support page for WEBHUNTER. Admins can view all user messages and respond by sending stickers, photos, videos, or text messages to the users. Both users and admins can initiate a meeting by clicking the camera or phone call button at the top of the page. Next, Fig. 22 shows the interface of the live support meeting page. During the meeting, either the admin or the user can share their screen to demonstrate or resolve issues.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1229-1249

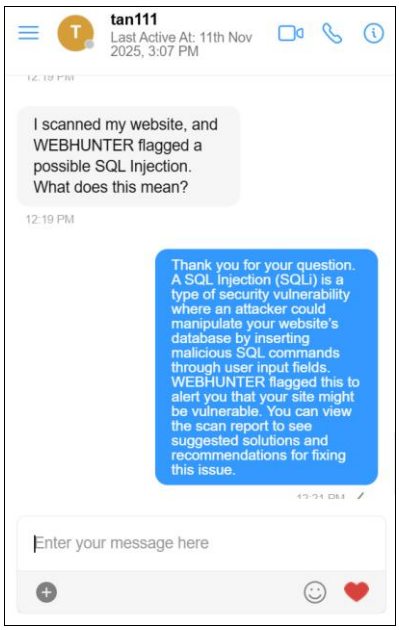


Fig. 21 *Interface of Live Support Page*

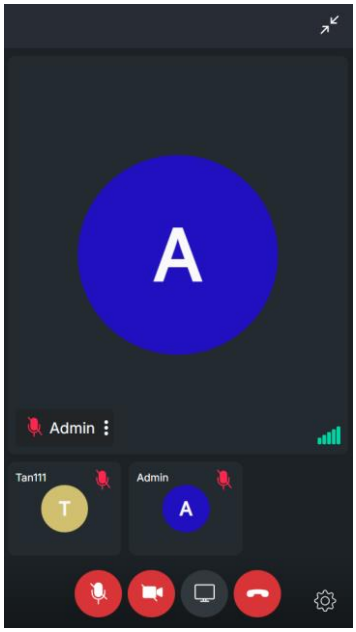


Fig. 22 *Interface of Live Support Meeting Page*

5.2 Testing

User Acceptance Testing (UAT) was conducted to determine whether the WEBHUNTER system meets end-user expectations and functions as intended. A total of 30 users participated in the evaluation, which assessed three key aspects of the system, including system functionality, system usability, and user interface design. The feedback gathered from these users was used to verify the system's overall effectiveness and alignment with user needs.

Fig. 23 shows user satisfaction with the system functionality. The results indicate a highly positive response from the participants. Most respondents selected "Strongly Agree" across all statements, demonstrating strong confidence in the system's functional performance. A small number of respondents chose "Agree" and "Neutral", showing a generally positive sentiment with minor variation. Notably, none of the participants selected "Disagree" or "Strongly Disagree", demonstrating that users consistently found the system reliable, effective, and capable of performing its intended tasks without issues. Overall, the feedback confirms that WEBHUNTER successfully meets users' expectations in terms of system functionality.

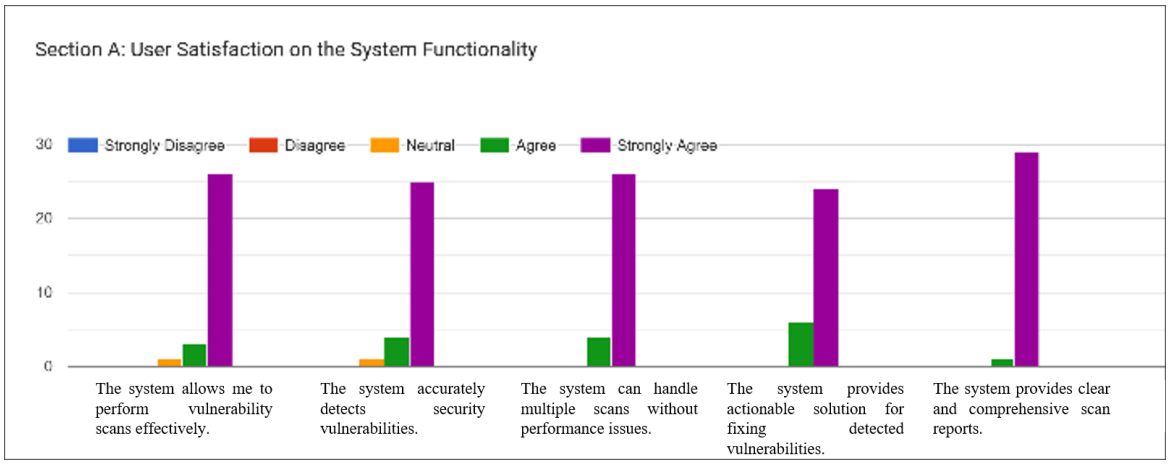


Fig. 23 *User Satisfaction on the System Functionality*

Fig. 24 illustrates user satisfaction on the sysystem'ssability. The results show a highly positive response from participants. Most respondents selected "Strongly Agree", indicating that the system is user-friendly, intuitive, and easy to operate. A smaller number chose "Agree" and "Neutral", reflecting generally positive sentiment with minimal variation. No participants selected "Disagree" or "Strongly Disagree", suggesting that users consistently found the system accessible, straightforward, and convenient to use across devices. Overall,

the feedback confirms that WEBHUNTER provides smooth and effective user experience, supporting ease of navigation and usability.

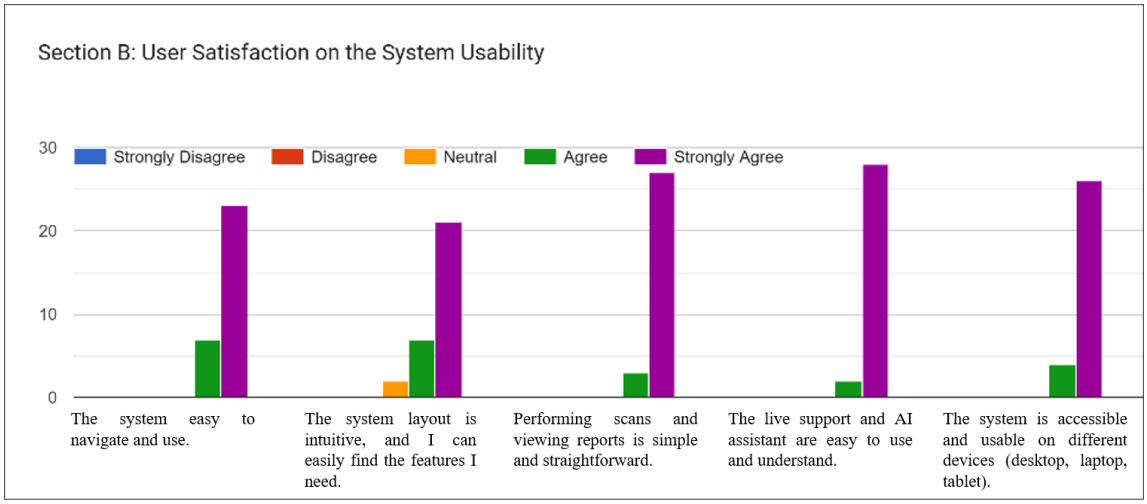


Fig. 24 *User Satisfaction on the System Usability*

Fig. 25 presents user satisfaction on the system user interface design. The results indicate a highly positive response from participants. Most respondents selected "Strongly Agree", reflecting strong approval of the system's visual appeal, organization, and responsiveness. A smaller portion chose "Agree" and "Neutral", showing generally positive feedback with minor variation. No respondents selected "Disagree" or "Strongly Disagree", demonstrating that users consistently found the interface professional, intuitive, and enjoyable to use. Overall, the feedback confirms that WEBHUNTER's user interface design effectively enhances the user experience, providing a visually appealing and well-organized environment for performing vulnerability scans.

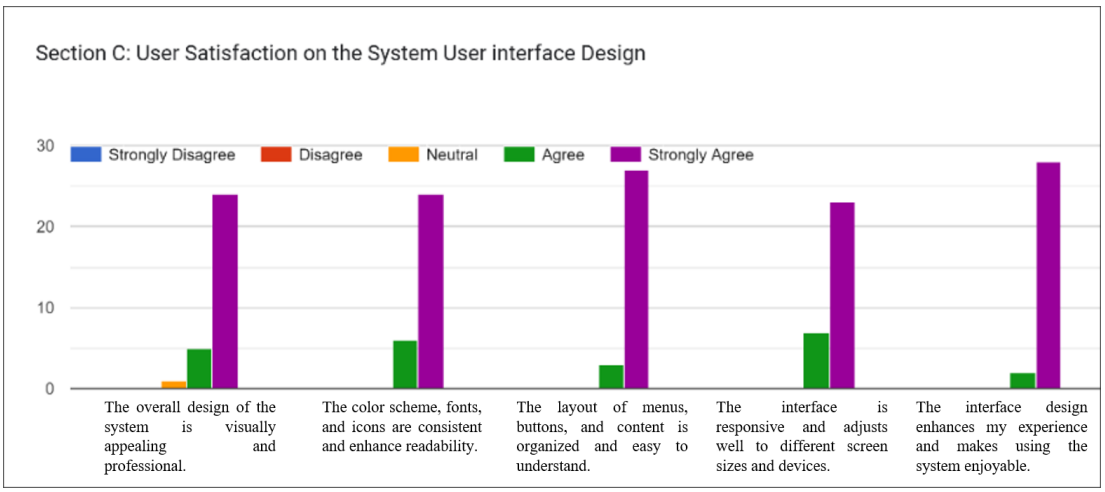


Fig. 25 *User Satisfaction on the System User interface Design*

Fig. 26 illustrates user comments, suggestions, and feedback about WEBHUNTER. The participants provided positive feedback, highlighting the AI assistant's responsiveness, smooth system performance, and the clarity of scan results. Some users suggested adding more types of vulnerability scans in future updates, reflecting interest in further enhancing the system's functionality. A lighthearted comment that "the hamster is very cute" also emphasized that the system's design and interactive elements contribute to a positive and enjoyable user experience. Overall, the feedback provides valuable insights for potential improvements and confirms that WEBHUNTER is well received by its users.

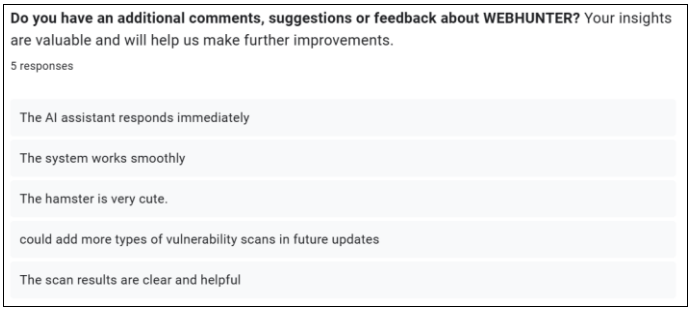


Fig. 26 *Comments, Suggestions, and Feedback about WEBHUNTER*

Accuracy is used to measure the overall correctness of a system's predictions [17]. A total of 50 URLs were analyzed, of which 40 were secure and 10 were identified as vulnerable, to determine whether the system could detect these vulnerabilities. Fig. 27 shows the accuracy testing result for SQLi demonstrates that the system achieved 96% accuracy, indicating a high level of correctness in identifying both vulnerable and secure URLs for SQL Injection. Fig. 30 presents the WEBHUNTER SQLi scan report for "https://juice-shop.herokuapp.com/#/", which was correctly identified as vulnerable to SQL injection (SQLi). The scan detected a successful SQLi attack (Vulnerability ID: vuln_284fb6bd431ccf5) during a JSON-based login request using the payload "' OR 1=1--". The vulnerability was exploited through the endpoint "https://juice-shop.herokuapp.com/rest/user/login", confirming the presence of SQL injection in the login functionality.

Next, Fig. 28 shows the accuracy testing result for XSS demonstrates that the system achieved 94% accuracy, indicating a high level of correctness in identifying both vulnerable and secure URLs for Cross-Site Scripting. Fig. 31 shows the WEBHUNTER XSS scan report for "http://testphp.vulnweb.com", which was also correctly identified as vulnerable to reflected XSS. In this case, the vulnerability (Vulnerability ID: vuln_9af933eecce1865) was detected using the payload "<input type=text value=alert(1)>" through the search functionality at "http://testphp.vulnweb.com/search.php". The reflection of the injected payload in the server response verified the existence of an XSS vulnerability within the application.

Lastly, Fig. 29 shows the accuracy testing result for CSRF demonstrates that the system achieved 86% accuracy, indicating a high level of correctness in identifying both vulnerable and secure URLs for Cross-Site Request Forgery. Fig. 32 presents the WEBHUNTER CSRF scan report for "https://demo.testfire.net/login.jsp", which was correctly identified as vulnerable to CSRF. The scan detected a CSRF vulnerability (Vulnerability ID: vuln_e23e02b4e9b681b) where a login request was accepted without a CSRF token. The vulnerability was observed in a POST request submitted to the action endpoint "https://demo.testfire.net/doLogin", confirming that the application does not implement adequate CSRF protection for the authentication process.

$$TP = 5,\ TN = 43$$
$$TP + FP + TN + FN = 50$$
$$\text{Accuracy for SQLi} = \frac{(5 + 43)}{50} = \frac{48}{50} = 0.96 = 96\%$$

Fig. 27 *Accuracy Testing Results for SQL Injection (SQLi)*

$$TP = 6,\ TN = 41$$
$$TP + FP + TN + FN = 50$$
$$\text{Accuracy for XSS} = \frac{(6 + 41)}{50} = \frac{47}{50} = 0.94 = 94\%$$

Fig. 28 *Accuracy Testing Results for Cross-Site Scripting (XSS)*

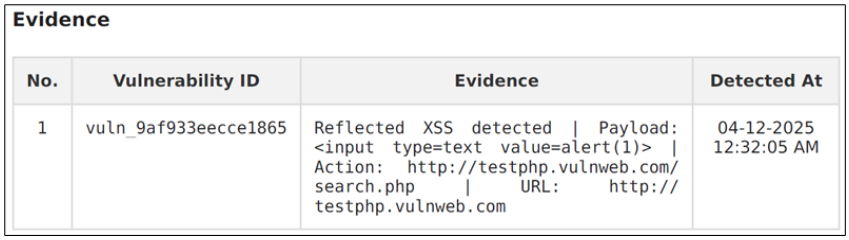
$$TP = 1,\ TN = 42$$
$$TP + FP + TN + FN = 50$$
$$\text{Accuracy for CSRF} = \frac{(1 + 42)}{50} = \frac{43}{50} = 0.86 = 86\%$$

Fig. 29 *Accuracy Testing Results for Cross-Site Request Forgery (CSRF)*

Evidence

No.	Vulnerability ID	Evidence	Detected At
1	vuln_284fb6bd431ccf5	JSON login SQLi success with payload "' OR 1=1--" | Endpoint: https://juice-shop.herokuapp.com/rest/user/login | URL: https://juice-shop.herokuapp.com/l/login	04-12-2025 12:28:37 AM

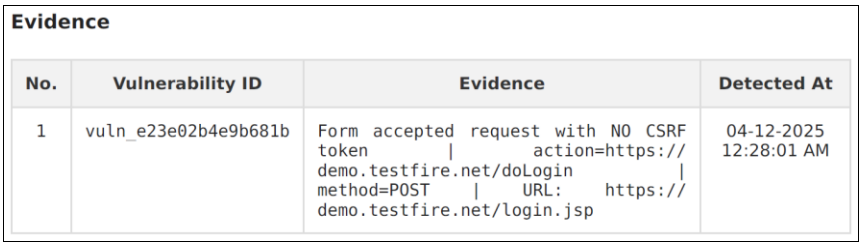
Fig. 30 *WEBHUNTER SQLi Scan Result for https://juice-shop.herokuapp.com/#/*



Evidence

No.	Vulnerability ID	Evidence	Detected At
1	vuln_9af933eecce1865	Reflected XSS detected | Payload: <input type=text value=alert(1)> | Action: http://testphp.vulnweb.com/search.php | URL: http://testphp.vulnweb.com	04-12-2025 12:32:05 AM

Fig. 31 *WEBHUNTER XSS Scan Result for http://testphp.vulnweb.com*



Evidence

No.	Vulnerability ID	Evidence	Detected At
1	vuln_e23e02b4e9b681b	Form accepted request with NO CSRF token | action=https://demo.testfire.net/doLogin | method=POST | URL: https://demo.testfire.net/login.jsp	04-12-2025 12:28:01 AM

Fig. 32 *WEBHUNTER CSRF Scan Result for https://demo.testfire.net/login.jsp*

Fig. 33 shows the accuracy results for Command Injection detected by OWASP ZAP, which includes vulnerabilities such as SQL Injection (SQLi) and Cross-Site Scripting (XSS). The tool achieved an overall detection accuracy of 77%, based on 251 total tests, with 193 passes and 58 fails [18]. While OWASP ZAP is a widely used and reliable vulnerability scanner, this indicates that a notable number of vulnerabilities were not detected. In comparison, WEBHUNTER achieved significantly higher detection accuracy, with SQLi detection at 96% and XSS detection at 94%. These results suggest that WEBHUNTER is more effective in identifying critical web application vulnerabilities, providing more reliable coverage, and reducing the likelihood of false negatives.

Section	Total Tests	Passes	Fails	Score
Command Injection	251	193	58	77%

Fig. 33 *Accuracy Result for OWAPS ZAP*

6. Conclusion

This project presents the successful development of WEBHUNTER, designed to provide an automated web application vulnerability scanning solution. All objectives were successfully achieved, including the detection of common web vulnerabilities, generation of detailed vulnerability reports, and offering an intuitive, user-friendly interface.

The system offers several advantages, including highly accurate detection of SQLi, XSS and CSRF, automated reporting with CVE identifiers, severity levels, and recommended remediation steps, robust security features, and real-time AI assistance with live chat support. These features enhance security, improve usability, and provide users with actionable insights to strengthen their web applications. In comparison, OWASP ZAP achieved an overall detection accuracy of 77% for SQLi and XSS vulnerabilities. These results indicate that WEBHUNTER provides significantly higher detection accuracy, effectively identifying both vulnerable and secure web applications while minimizing false positives and false negatives.

Despite its strengths, WEBHUNTER has some limitations. The system's detection accuracy for CSRF is slightly lower than for other vulnerabilities, it currently covers a limited range of web vulnerabilities, and its AI assistance relies on third-party services and internet connectivity. These areas provide opportunities for future improvements.

Future enhancements could include broadening the range of detectable vulnerabilities to address advanced threats such as SSRF, broken authentication, and complex logic flaws. Additional improvements may involve increasing CSRF detection accuracy, incorporate offline or alternative AI support, and integrating machine learning-based detection techniques to more effectively identify new and unknown threats.

Overall, WEBHUNTER is a reliable, secure, and user-centric platform that successfully strengthens web application security through its combination of accurate detection, automated reporting, usability, and robust security features.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *B. J. Tan, S. N. Ramli;* ***data collection:*** *B. J. Tan, S. N. Ramli;* ***analysis and interpretation of results:*** *B. J. Tan, S. N. Ramli;* ***draft manuscript preparation:*** *B. J. Tan, S. N. Ramli. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Upadhyay, D., & Ware, N. R. (2023). International Journal of Engineering Technology and Management Sciences Evolving Trends in Web Application Vulnerabilities: A Comparative Study of OWASP Top 10 2017 and OWASP Top 10 2021. https://doi.org/10.46647/ijetms.2023.v07i06.038

[2] Shahid, J., Hameed, M. K., Javed, I. T., Qureshi, K. N., Ali, M., & Crespi, N. (2022). A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. Applied Sciences (Switzerland), 12(8). https://doi.org/10.3390/app12084077

[3] Ali, N. (2023, September 10). 7 Limitations of Vulnerability Scanners. Beagle Security, https://beaglesecurity.com/blog/article/limitations-of-vulnerability-scanners.html

[4] Miguel, P. G. (2025, March 21). 25 Best Vulnerability Scanning Software Reviewed in 2025. The CTO Club. https://thectoclub.com/tools/best-vulnerability-scanning-tools/

[5] Paul, A., Sharma, V., & Olukoya, O. (2024). SQL injection attack: Detection, prioritization & prevention. Journal of Information Security and Applications, 85. https://doi.org/10.1016/j.jisa.2024.103871

[6] Weamie, S. J. Y. (2022). Cross-Site Scripting Attacks and Defensive Techniques: A Comprehensive Survey. International Journal of Communications, Network and System Sciences, 15(08), 126–148. https://doi.org/10.4236/ijcns.2022.158010

[7] Siahaan, C. N., Rufisanto, M., Nolasco, R., Achmad, S., & Siahaan, C. R. P. (2023). Study of Cross-Site Request Forgery on Web-Based Application: Exploitations and Preventions. Procedia Computer Science, 227, 92–100. https://doi.org/10.1016/j.procs.2023.10.506

[8] Martin, R., Christey, S., & Baker, D. (2021, April 5). A Progress Report on the CVE Initiative. https://apps.dtic.mil/sti/trecms/pdf/AD1125343.pdf

[9] Sangfor Technologies. (2022, November 25). What is CVE? – Understanding common vulnerabilities & exposures. Sangfor Technologies. https://www.sangfor.com/blog/cybersecurity/what-is-cve-common-vulnerabilities-exposures

[10] Alazmi, S., & de Leon, D. C. (2022). A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners. In IEEE Access (Vol. 10, pp. 33200–33219). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2022.3161522

[11] Nidecki, T. A. (2022, July 18). How scanners find vulnerabilities. Acunetix. https://www.acunetix.com/blog/web-security-zone/how-scanners-find-vulnerabilities/

[12] Severns, R. (2025, March 6). OWASP ZAP: Open Source App Security Testing. StackHawk. https://www.stackhawk.com/blog/guide-to-zap-application-security-testing/

[13] Intruder. (2025). Web Application Vulnerability Scanner. Intruder. https://www.intruder.io/web-application-vulnerability-scanner

[14] Invicti. (2025). What is Invicti? | Application Security Scanner. Invicti. https://www.invicti.com/support/what-is-invicti/

[15] Invicti. (2025). Seamless DevSecOps integration: AppSec that fits your Dev workflow. Invicti. https://www.invicti.com/blog/web-security/dast-first-devsecops-integration

[16] Laoyan, S. (2025, February 20). What is agile methodology? (A beginner's guide). Asana. https://asana.com/resources/agile-methodology

[17] Armah, G. K., Luo, G., & Qin, K. (2014). A Deep Analysis of the Precision Formula for Imbalanced Class Distribution. International Journal of Machine Learning and Computing, 4(5), 417–422. https://doi.org/10.7763/ijmlc.2014.v4.447

[18] OWASP. (2025). ZAP vs OWASP Benchmark. OWASP. https://www.zaproxy.org/docs/scans/benchmark/