

An Automated Web Application Vulnerability Assessment Framework Based on the OWASP ZAP API

Tiara Wigati Septiani^{1✉}, Dendy K Pramudito², Hendra Arya Syaputra³

^{1,2,3}Informatics Engineering Department, Universitas Pelita Bangsa

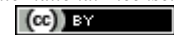
papers.upb@gmail.com

Abstract

The increasing reliance on web applications has expanded the attack surface and intensified the need for systematic vulnerability assessment. However, manual security testing remains resource intensive and often requires specialized expertise, while fragmented reporting can hinder vulnerability tracking, risk prioritization, and remediation. This study aims to design and develop a web-based vulnerability assessment system integrating the OWASP ZAP API to support automated scanning and centralized security monitoring. The system was developed using the Rapid Application Development methodology, which emphasizes iterative development, rapid prototyping, and reusable components. Its main features include administrator authentication, a monitoring dashboard, automated website scanning, scan history management, vulnerability risk classification, and automated mitigation recommendations. The system also incorporates controlled brute-force testing to assess the resilience of its login mechanism. By integrating scanning, reporting, and remediation guidance within a single interface, the proposed system is intended to reduce the operational complexity of routine vulnerability assessment and improve the accessibility of security information for developers and administrators. The resulting implementation demonstrates the feasibility of integrating OWASP ZAP into a web-based platform for managing vulnerability assessment workflows. Rather than replacing expert security analysis, the system provides a structured mechanism for identifying potential weaknesses, reviewing their reported severity, and supporting remediation decisions. This research contributes a practical implementation approach for integrating open-source security testing tools into application development and maintenance processes, with the broader goal of supporting consistent, traceable, and repeatable vulnerability management. Further empirical evaluation is required to quantify scanning accuracy, execution time, and operational effectiveness across different environments.

Keywords: Vulnerability Assessment, OWASP ZAP, Web Security, Rapid Application Development, Penetration Testing.

INFEB is licensed under a Creative Commons 4.0 International License.



1. Introduction

The rapid development of information technology and the internet have significantly increased the use of web-based applications in various sectors. Websites and web applications are widely used to provide information, support business processes, deliver public services, and facilitate access to digital services. The increasing dependence on web-based systems provides benefits in terms of accessibility and efficiency, but it also creates security challenges because applications that are connected to the internet can become potential targets for cyberattacks. Therefore, website security needs to be considered as an important aspect in the development and management of information systems. Previous research has shown that website security testing is necessary to identify potential vulnerabilities that may be exploited by unauthorized parties [1] [2], for example, conducted website vulnerability testing using a *grey-box penetration testing* approach with OWASP and OWASP ZAP, demonstrating the importance of security testing to identify weaknesses in web applications.

One of the challenges in maintaining website security is the existence of vulnerabilities within application components and configurations. A vulnerability is a weakness that may allow an attacker to perform

unauthorized actions, access information, or interfere with the normal operation of an application. Such weaknesses can occur because of inappropriate configurations, insufficient input validation, insecure application components, or weaknesses in security mechanisms. Vulnerability assessment can therefore be used to identify potential weaknesses before they are exploited. Conducted a Vulnerability Assessment at Universitas of Singaperbangsa Karawang website using OWASP ZAP, Nmap, Nikto, and Acunetix [2]. Their study demonstrated that automated vulnerability scanning could identify several security findings with different risk levels, indicating the usefulness of vulnerability assessment as part of website security evaluation.

The OWASP framework utilization has become common in web application security research because it provides references for identifying important categories of web application risks. OWASP-based testing can help researchers and developers evaluate vulnerabilities systematically rather than relying only on manual inspection. Research by Hermanto and Haeruddin [3] applied an OWASP-based approach to improve website security, further demonstrating the use of OWASP as a reference for identifying and addressing security weaknesses. The application of OWASP-based testing is particularly relevant because vulnerabilities may

occur in different parts of a web application and may have different levels of impact depending on the conditions required for exploitation.

Another approach that can support website security evaluation is *Vulnerability Assessment* using automated scanning tools. OWASP ZAP (*Zed Attack Proxy*) is one of the tools frequently used for this purpose. OWASP ZAP is used to analyse website vulnerabilities and demonstrated that the tool could identify several security weaknesses, including *Absence of Anti-CSRF Tokens*, *Content Security Policy (CSP) Header Not Set*, *Missing Anti-clickjacking Header*, *Cookie No HttpOnly Flag*, and *Cookie without SameSite Attribute*. Their findings show that automated scanning can provide detailed information regarding security configuration weaknesses that may require further mitigation [4].

The effectiveness of vulnerability assessment using OWASP ZAP has also been demonstrated in other application domains. Analysing the security of a population administration information system website using the *Vulnerability Assessment* method and OWASP ZAP. Their research identified several vulnerability warnings, including findings with high-risk levels, and provided recommendations for reducing security weaknesses [5]. The study shows that vulnerability scanning can be used not only to identify vulnerabilities but also as a basis for recommending security improvements. The security of the PT Sentra Vidya Utama website was analysed using the OWASP method and OWASP ZAP [6]. Their research found vulnerabilities with medium to low risk levels and emphasized the need for security mechanisms such as CSP and other security controls. This finding demonstrates that web application security problems are not limited to major application functions but can also be related to configuration and protection mechanisms implemented within the website. Further research has examined website vulnerability testing using more structured penetration testing approaches. Meanwhile, [7] combined the *Penetration Testing Execution Standard (PTES)* and *Vulnerability Assessment* approaches with OWASP ZAP, Nikto, and Nmap to analyze the Baleomolcreative website. Their research identified 18 alerts consisting of medium, low, and informational findings. Several findings were related to CSP, anti-clickjacking headers, JavaScript functions, and other HTTP security configurations. These results demonstrate that automated scanning can identify multiple categories of security weaknesses and provide information that can support subsequent security evaluation.

Research focusing on specific vulnerability categories has also demonstrated the usefulness of OWASP ZAP. Penetration testing to identify *Cross-Site Scripting (XSS)* vulnerabilities using OWASP ZAP and Hackbar. Their research identified several security alerts, including *X-Frame-Options Header Not Set*, *Absence of Anti-CSRF Tokens*, *Cross-Domain JavaScript Source File Inclusion*, and *Missing X-Content-Type-Options-Header* was conducted by [8]. The study also identified

a reflected XSS vulnerability and provided recommendations for improving the security of the tested website. The *Vulnerability Assessment* at website of Universitas of Internasional Batam using OWASP ZAP and Nessus [9]. OWASP ZAP detected 11 vulnerabilities consisting of six medium-level and five low-level findings, including weaknesses related to CSP and anti-clickjacking headers. The study also demonstrated that different vulnerability scanning tools can produce different findings, indicating that the selection and integration of security testing tools can influence the results of website security assessments.

More recent research has focused on combining multiple automated security scanner, analyzed the implementation of Nuclei Vulnerability Scanner and OWASP-ZAP Scanner for detecting security vulnerabilities on a web-based platform [10]. The research tested a locally developed web application and showed that the two tools could provide complementary information regarding vulnerabilities at different layers. The study also recorded vulnerability severity, mitigation recommendations, and scanning performance, demonstrating the potential of automated scanners to support systematic security assessment. [11] further examined website vulnerability testing using OWASP ZAP. Their research highlighted the use of automatic scanning, manual testing, and reporting functions in the vulnerability assessment process. The study identified several OWASP-related vulnerabilities, including XSS, clickjacking, and insecure configurations, and emphasized the importance of applying mitigation measures such as security headers and protection of sensitive information.

The relationship between vulnerability detection and mitigation has also received attention in recent research. Vulnerability testing and mitigation on an academic information system website using OWASP ZAP where The research was not limited to identifying vulnerabilities but also aimed to provide mitigation recommendations for the detected security weaknesses [12]. This approach demonstrates that the results of vulnerability scanning can be made more useful when they are accompanied by practical recommendations for improving application security. Analyzed the security of the Planet Komputer Kebumen website using the *Vulnerability Assessment* method [13]. The research applied information gathering using Whois and Nmap, followed by vulnerability scanning using OWASP ZAP. The results identified four medium-level vulnerabilities and were subsequently analyzed to formulate recommendations for improving website security. This study demonstrates that OWASP ZAP can support a structured vulnerability assessment process, from information gathering and automated scanning to vulnerability analysis and mitigation recommendations. However, the research focused on assessing a specific website and did not develop a centralized web-based system that integrates OWASP ZAP through its API for managing the scanning process and its results. The University of Malikussaleh Academic Portal using OWASP ZAP has done the

security test [14] to identify security vulnerabilities in an academic information system that provides services for students, lecturers, and staff. The study demonstrates that OWASP ZAP can be utilized to identify potential security weaknesses in web-based academic information systems. This research shows the applicability of automated vulnerability scanning in evaluating the security condition of an existing website. However, the study focuses on analyzing a specific academic portal and does not develop an integrated web-based vulnerability scanning system that utilizes the OWASP ZAP API to manage the scanning process, vulnerability classification, scanning history, mitigation, and reporting.

The most recent study considered in this research was conducted by [15], who analyzed the MAN 2 Kebumen website using OWASP ZAP and the OWASP Web Security Testing Guide (WSTG). Their research used a *Vulnerability Assessment* approach and identified a high-risk potential XSS vulnerability related to a user-controllable HTML element attribute. The researchers also evaluated CSP configuration and provided recommendations for improving input validation and sanitization. This study demonstrates the continued relevance of OWASP ZAP for current website vulnerability assessment and shows the importance of connecting vulnerability findings with appropriate security recommendations. Based on the previous studies [1] [15], OWASP ZAP has been widely utilized to identify vulnerabilities in web applications through automated scanning and penetration testing. Existing research has demonstrated its ability to identify vulnerabilities related to security headers, CSRF protection, XSS, insecure configurations, and other security weaknesses. However, most of these studies focus on assessing a specific website, comparing security tools, applying penetration testing methods, or analyzing the vulnerabilities identified by OWASP ZAP. Some studies have also provided mitigation recommendations, but the scanning process and its results are generally presented as part of an assessment of a particular target rather than as a dedicated web-based vulnerability scanning system.

This condition creates a research gap for developing an integrated application that does not merely use OWASP ZAP as a standalone scanning tool but integrates its scanning capability through the OWASP ZAP API into a web-based system. The proposed approach combines the vulnerability scanning process with risk-level classification, scan history management, mitigation recommendations, and vulnerability report generation within a centralized application. In contrast to previous studies that primarily focus on analyzing vulnerabilities on individual websites [4] [7] [9] [11] [13] [14], this research focuses on developing a system that facilitates the complete vulnerability assessment workflow through a web interface. Therefore, this research proposes the development of a Web-Based Website Vulnerability Scanning System Using the OWASP ZAP API. The system is designed to allow users to enter a target website URL and initiate the vulnerability

scanning process through a web interface.

The scanning request is sent to OWASP ZAP through its API, while the resulting vulnerability information is processed and presented according to its risk level. The system also provides scan history, automatic mitigation recommendations, and vulnerability assessment reports to support the management and documentation of security testing results. The objective of this research is to develop and evaluate a web-based website vulnerability scanning system integrated with the OWASP ZAP API for automated vulnerability detection. The proposed system is expected to simplify and centralize the website vulnerability assessment process, provide structured information regarding detected vulnerabilities, classify findings according to their risk levels, maintain a history of previous scans, and provide mitigation recommendations that can assist developers in improving website security. Through these capabilities, the research aims to provide a more systematic, accessible, and efficient approach to supporting website security assessment.

2. Research Method

This research was conducted through several systematic stages to develop and evaluate a web-based website vulnerability scanning system using the OWASP ZAP API. The research stages include problem identification, requirements analysis, system design, system development, OWASP ZAP API integration, system testing, and evaluation of the obtained results. The sequence of these stages is illustrated in Figure 1. Problem identification, which aims to identify limitations in the existing website security testing process. The analysis focuses on the use of OWASP ZAP for vulnerability assessment and identifies several limitations, including the need for technical procedures, the absence of an integrated web-based interface, and the lack of structured storage for previous scanning results. Based on these problems, a web-based system integrated with the OWASP ZAP API is proposed to simplify and automate the vulnerability scanning process. Next Research Stages on Figure 1.

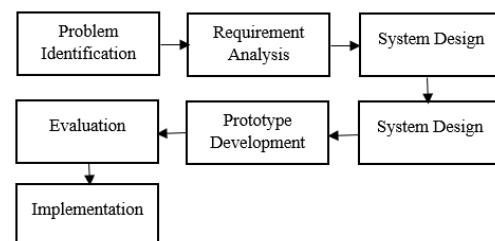


Figure 1. Research Stages

Requirements analysis, which determines the functional requirements of the proposed system. The identified requirements include user authentication, target website URL input, integration with the OWASP ZAP API, automated vulnerability scanning, vulnerability classification based on risk level, scan history, dashboard monitoring, mitigation recommendations, and vulnerability report generation. These requirements

serve as the basis for designing and developing the proposed system.

System design, which translates the identified requirements into system structure and workflow. The design includes the interaction between the user, web application, OWASP ZAP API, scanning engine, and database. The system design also covers the database structure and system components required to manage users, target websites, scanning processes, vulnerability findings, and reports. The interaction between these components is represented by using system modeling to support the subsequent implementation process.

Prototype development, in which the initial version of the proposed system is developed based on the system requirements and design. The prototype is developed as a web-based application and integrated with OWASP ZAP through its API. The prototype provides the main functions required for vulnerability scanning and vulnerability result management. Prototype evaluation, which is conducted to determine whether the developed prototype meets the identified requirements. The evaluation focuses on the functionality and suitability of the system based on the proposed workflow. Improvements are made when the prototype does not yet meet the required functionality.

Implementation, where the evaluated prototype is developed into a functional web application. PHP is used as the backend programming language, MySQL is used for database management, and JavaScript with AJAX is used to support asynchronous communication. OWASP ZAP API functions as the main scanning engine for detecting vulnerabilities in the target website. System testing, which is performed to verify that the developed system functions according to the specified requirements. The testing evaluates the system's ability to receive input, process scanning requests, obtain vulnerability results from OWASP ZAP, store scanning data, and present the results to users. Black Box Testing is used to evaluate system functionality based on input and output without examining the internal source code.

The system was developed using prototype development method. The Prototype method was selected because it allows the development process to be performed iteratively based on user requirements and evaluation results [16]. Through this approach, an initial version of the system can be developed and evaluated before being refined into the final system. The development process consists of several activities, namely requirements analysis, initial system design, prototype development, prototype evaluation, and system refinement. Requirements analysis is performed to identify the functions required by the system. The initial design is then created based on these requirements, followed by the development of a functional prototype. The prototype is evaluated based on the identified requirements. If there are functions that do not correspond to the requirements, improvements are made to the prototype. This process is repeated until the system functions are considered

appropriate for the research objectives. In this research, the Prototype method is applied to the development of the website vulnerability scanning system. The main system components include user authentication, target URL management, OWASP ZAP API integration, vulnerability result processing, risk classification, scan history, mitigation recommendations, and report generation.

System testing was conducted to verify whether the functions implemented in the application operate according to the specified requirements. The testing method used in this research is Black Box Testing. This method evaluates system functionality by examining the relationship between input and output without examining the internal implementation of the source code [17]. The testing was performed by providing specific inputs to each system function and observing the resulting output. The tested functions include user registration and login, target URL input, vulnerability scanning, vulnerability result display, risk classification, scan history, mitigation recommendations, and PDF report generation. A function is considered successful when the output generated by the system corresponds to the expected result. The testing results are subsequently used to evaluate the functional suitability of the developed system.

3. Result and Discussion

The implementation stage is the process of applying the system design into a functional web-based application. The developed system integrates PHP as the backend programming language, MySQL as the database management system, and JavaScript with AJAX to support asynchronous data communication. OWASP ZAP API is integrated as the main engine for performing *vulnerability scanning* on the target website. The implementation is focused on providing an automated and centralized process for website security assessment. The use of OWASP ZAP as a vulnerability scanning engine is consistent with previous studies that have demonstrated its applicability in identifying security weaknesses in web applications [4] [7] [9] [11] [14] [15]. The implemented system accepts a target website URL through the web interface and sends a scanning request to the OWASP ZAP API. The vulnerability data returned by the API are subsequently processed by the system and presented in a structured format. The main functions implemented in the system include user authentication, target URL input, OWASP ZAP API integration, *vulnerability scanning*, risk-level classification, scan history, dashboard monitoring, mitigation recommendations, and PDF report generation. These functions are intended to support users in identifying and managing website security vulnerabilities through an integrated interface. The integration of automated vulnerability detection with additional result-management functions extends the approach commonly used in previous vulnerability assessment research [1] [2] [5]. Next Main Dashboard on Figure 2.

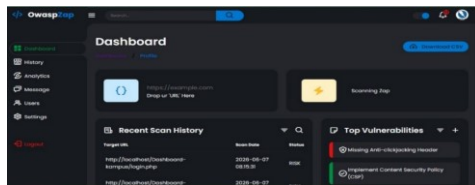


Figure 2. Main Dashboard

The main dashboard functions as the central interface after successful user authentication. It provides access to the main *vulnerability scanner* functions and allows users to initiate website security scanning. The dashboard contains a target URL input form and a *Start Scan* button connected to the OWASP ZAP API. After the target URL is submitted, the system sends a scanning request and displays the status of the scanning process. The dashboard also provides a summary of recent scanning activities, detected vulnerabilities, and their respective risk levels, allowing users to obtain an overview of the target website's security condition. The implementation of the dashboard provides an interface that simplifies the vulnerability assessment process. Previous research has generally applied OWASP ZAP directly as part of vulnerability assessment on a specific website [4] [6] [14]. In this research, the scanning capability is integrated into a dedicated web application so that the scanning process and its results can be managed through a centralized interface. Next Vulnerability Scan Results on Figure 3.

Vulnerability Assessment Result				
	High	Medium	Low	Informational
1	1	2	3	4
Detected Vulnerability (all Findings)				
No	Risk	Name	Description	URL
1	High	Missing Content Security Policy	The response does not contain any 'Content Security Policy' header. It should include a Content Security Policy with appropriate directives to prevent XSS attacks.	http://localhost:8080/owasp-zap/
2	High	Content Security Policy (CSP) is not enforced	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used to execute unwanted code on the web page. CSP is a standard that helps to prevent these attacks by using a whitelist of approved sources of content that are allowed to load on that page. CSP is a standard that helps to prevent these attacks by using a whitelist of approved sources of content that are allowed to load on that page.	http://localhost:8080/owasp-zap/
3	Low	Cookie Not Secure	A cookie has been set without the 'Secure' flag, which means that the cookie can be transmitted by insecure connections (e.g., HTTP). This is a security issue as the cookie may contain sensitive information that can be intercepted by an attacker.	http://localhost:8080/owasp-zap/
4	Low	Cookie Not HttpOnly	A cookie has been set without the 'HttpOnly' flag, which means that the cookie can be accessed by JavaScript. This is a security issue as the cookie may contain sensitive information that can be accessed by a malicious script.	http://localhost:8080/owasp-zap/

Figure 3. Vulnerability Scan Results

After the scanning process is completed, the system displays the results on the *Vulnerability Scan Results* page. The results are presented in the form of a summary and a detailed vulnerability table. The summary provides the number of detected vulnerabilities according to the *High*, *Medium*, *Low*, and *Informational* categories. Meanwhile, the detailed section presents the vulnerability name, risk level, description, and affected URL. The system also provides mitigation recommendations for the detected vulnerabilities, allowing users to understand the identified security issues and determine appropriate corrective actions. The scanning results obtained from OWASP ZAP are processed before being displayed to the user. The system retrieves information including the alert name, risk level, technical description, affected parameter or endpoint, and evidence of the detected vulnerability. The retrieved data are then parsed into a structured format for presentation on the web dashboard. In addition to presenting the findings, the system analyzes the detected vulnerabilities and generates mitigation recommendations by matching the vulnerability category with the available mitigation rules. This process allows the system to provide not

only information about detected vulnerabilities but also suggested actions for reducing the associated security risks.

The implementation of mitigation recommendations is supported by previous research emphasizing that vulnerability assessment should be followed by appropriate mitigation actions. Applied vulnerability testing together with mitigation using OWASP ZAP [12], while another recommendations also provided by [13] based on the vulnerabilities identified through vulnerability assessment. Therefore, the inclusion of mitigation recommendations in the developed system provides an additional function for assisting users in responding to detected vulnerabilities. Next Scan History on Figure 4.

Figure 4. Scan History

The *Scan History* feature stores and displays previously performed vulnerability scanning activities. The data are stored in the database and can be accessed again without repeating the scanning process. The feature records information such as the target website URL, the number of vulnerabilities according to risk level, and the date and time of the scanning process. This functionality enables users to document previous assessments and compare scanning results over different periods. The scan history also supports continuous security monitoring. Whenever a vulnerability scanning process is successfully completed, the corresponding data are stored in the *scan_history* table. The stored information can subsequently be used to evaluate changes in the security condition of a website after mitigation or security improvements have been applied. This feature provides a distinction from studies that primarily focus on the assessment of a specific website and the identification of its vulnerabilities [2] [4] [7] [9]. By retaining previous assessment results, the developed system provides additional functionality for documenting and monitoring vulnerability assessment activities. Next PDF Vulnerability Assessment Report on Figure 5.

Figure 5. PDF Vulnerability Assessment Report

The system also implements a PDF reporting feature to document the results of the vulnerability assessment. The generated report contains general scanning

information, including the scanning date and time, target URL, number of vulnerabilities according to risk level, *risk score*, and overall website security status. The report also provides detailed information for each detected vulnerability, including the vulnerability name, risk level, affected URL, impact, and mitigation recommendation. This feature allows the scanning results to be stored, printed, and used as documentation for subsequent security evaluation and improvement. The reporting function complements the vulnerability assessment process by transforming the scanning output into documented information that can be used for further analysis.

Overall, the implementation demonstrates that the proposed system can integrate the OWASP ZAP API into a web-based application and provide a centralized workflow for website vulnerability assessment. The system automates the scanning process, presents vulnerability findings according to their risk levels, stores previous scanning activities, provides mitigation recommendations, and generates vulnerability assessment reports. These functions address the need for a more structured and accessible mechanism for conducting website security assessments. The implementation also demonstrates the research gap identified from previous studies. Previous research has extensively used OWASP ZAP for vulnerability assessment [1] [4] [7] [9] [11] [14] [15], but the developed system combines the scanning capability with result management, risk classification, scan history, mitigation recommendations, and reporting within one web-based application.

The vulnerability scanning process was conducted using the developed web-based system integrated with the OWASP ZAP API. The scanning process begins when the user enters the target website URL through the dashboard and selects the *Start Scan* function. The system then sends the scanning request to OWASP ZAP through the API. During the scanning process, requests and responses from the target website are analyzed automatically. After the scanning process is completed, the system retrieves the vulnerability information from the OWASP ZAP API and processes the data before displaying it on the web interface. The vulnerability information obtained from the API includes the *alert name*, risk level, technical description, affected parameter or endpoint, and evidence of the detected vulnerability. The system performs data parsing so that the information returned by OWASP ZAP can be presented in a structured format on the vulnerability results page. The results are grouped according to the risk categories *High*, *Medium*, *Low*, and *Informational*, allowing users to identify the severity and priority of each finding.

This scanning mechanism is consistent with previous studies that have used OWASP ZAP to identify and classify security vulnerabilities according to their respective risk levels [4] [7] [9] [11]. The use of automated scanning also supports the findings of previous research indicating that vulnerability

assessment tools can assist in identifying security weaknesses more efficiently than relying entirely on manual assessment [1] [2]. Based on the vulnerability assessment conducted on the target website, several security findings were identified in the *Medium*, *Low*, and *Informational* categories. The detected vulnerabilities included *Missing Anti-clickjacking Header*, *Content Security Policy (CSP) Header Not Set*, *Absence of Anti-CSRF Tokens*, *Cookie No HttpOnly Flag*, and *Cookie without SameSite Attribute*. These findings indicate that the target website still has several security configurations that require improvement to reduce the potential risk of unauthorized exploitation. Next Vulnerability Scanning Results on Table 1.

Table 1. Vulnerability Scanning Results

No.	Vulnerability	Risk Level	Security Issue
1	Missing Anti-clickjacking Header	Medium	Missing protection against clickjacking attacks
2	Content Security Policy (CSP) Header Not Set	Medium	CSP security policy is not configured
3	Absence of Anti-CSRF Tokens	Medium	CSRF protection mechanism is not implemented
4	Cookie No HttpOnly Flag	Low	Cookie does not use the HttpOnly security attribute
5	Cookie without SameSite Attribute	Low	Cookie does not specify the SameSite attribute

As shown in Table 1, the detected findings are primarily related to security configuration and protection mechanisms. The *Missing Anti-clickjacking Header* finding indicates that the website does not implement an appropriate mechanism to prevent its pages from being embedded and manipulated through clickjacking techniques. Similarly, the *Content Security Policy (CSP) Header Not Set* finding indicates that the website does not provide a CSP policy through the appropriate HTTP response header. These conditions can increase the exposure of the application to certain client-side attacks if they are not properly addressed. The *Absence of Anti-CSRF Tokens* finding indicates that the application does not implement a CSRF protection mechanism on the relevant requests. CSRF protection is important because an attacker may attempt to induce an authenticated user to perform unintended actions on the application. Meanwhile, the *Cookie No HttpOnly Flag* and *Cookie without SameSite Attribute* findings are related to cookie security configuration. The absence of these attributes can reduce the protection provided to session-related cookies and increase the potential for unauthorized access or misuse under certain attack conditions.

Similar to the study conducted [18], the present research uses OWASP ZAP to identify website vulnerabilities and analyzes the findings based on their security categories. However, the proposed system extends the assessment process by integrating OWASP

ZAP through an API into a web-based application and providing additional features for scan history, mitigation recommendations, and report generation. Similar findings have also been reported by Hasibuan, Tommy, and Handoko [4], who identified *Content Security Policy Header Not Set*, *Missing Anti-clickjacking Header*, *Absence of Anti-CSRF Tokens*, *Cookie No HttpOnly Flag*, and *Cookie without SameSite Attribute* through OWASP ZAP scanning. The similarity between the findings supports the applicability of the developed system in detecting common security configuration weaknesses.

The results are also consistent with Gustiyonoo, Alwi, and Abdullah [8], whose research identified security weaknesses related to *X-Frame-Options*, *Anti-CSRF Tokens*, and *X-Content-Type-Options* during penetration testing using OWASP ZAP. This demonstrates that security headers and protection mechanisms remain important aspects of website security assessment. The system does not only display the detected vulnerabilities but also provides additional information to support the analysis process. Each finding contains the vulnerability name, risk level, description, affected URL, and mitigation recommendation. This presentation allows users to identify the security issue and understand the potential impact without having to analyze the raw OWASP ZAP response directly. Another important result is the implementation of the automatic mitigation recommendation feature. After the vulnerability data are obtained from OWASP ZAP, the system analyzes each finding and matches its vulnerability category with the available mitigation rules. The system then generates recommendations that can be used as a reference for improving the security of the target website.

For example, vulnerabilities related to *SQL Injection* can be associated with the use of *prepared statements*, while *Cross-Site Scripting (XSS)* can be addressed through appropriate output encoding and input sanitization. The recommendation mechanism is aligned with previous research that emphasizes the importance of mitigation following vulnerability identification [12] [13] [15]. The scanning results are also stored automatically in the database. The stored information includes the target URL, vulnerability findings, risk levels, scanning time, and mitigation recommendations. This information is used to create the *Scan History* feature, allowing users to access previous scanning results without repeating the scanning process. The stored history can also support comparisons between assessments performed at different times and help users monitor changes in the security condition of the target website. In addition to displaying the results through the web interface, the system generates a PDF *vulnerability assessment* report. The report contains general scanning information, including the target URL, scanning date and time, number of findings based on risk level, *risk score*, and overall security status. Detailed information for each vulnerability is also included, along with its

risk level, affected URL, impact, and mitigation recommendation. Overall, the scanning results demonstrate that the developed system is capable of receiving vulnerability information from OWASP ZAP through its API, processing the results, classifying findings according to their risk levels, and presenting the information through a centralized web interface. The system also extends the basic scanning function by providing mitigation recommendations, scan history, and automated PDF reporting. Therefore, the resulting application provides an integrated mechanism for conducting and documenting website vulnerability assessments.

System testing was conducted after the implementation stage to verify whether the functions of the developed web-based vulnerability scanning system operate according to the specified requirements. The testing focuses on the system's ability to receive user input, process data, execute the vulnerability scanning process through the OWASP ZAP API, and provide the expected output. Testing is also intended to identify functional errors or inconsistencies before the system is used for broader website security assessment. The testing method used in this research is Black Box Testing. This method evaluates system functionality based on the input provided by the user and the output generated by the system without examining the internal structure of the program code [17]. Each function is tested using scenarios that represent the actual use of the application. A function is considered successful when the output generated by the system corresponds to the expected result.

The use of Black Box Testing is consistent with previous research that applies functional testing to verify whether implemented application features operate according to their requirements. In this research, Black Box Testing is used specifically to verify the functionality of the developed vulnerability scanning system rather than to evaluate the internal source code. The main functions tested include user registration, user authentication, target website URL input, vulnerability scanning through the OWASP ZAP API, storage of scan results in the database, presentation of vulnerability assessment results, mitigation recommendations, and generation of scanning reports in PDF format. These functions represent the main workflow of the developed system, from user access and target selection to vulnerability detection and reporting. Next Black Box Testing Results on Table 2.

Table 2. Black Box Testing Results

No	Function Tested	Expected Result	Test Result	Status
1	User Registration	New user data can be registered and stored in the database	User registration was successfully processed	Pass
2	User Login	Registered users can access the system using valid credentials	Authentication was successfully performed	Pass
3	Target URL Input	The system accepts the target website URL	Target URL was successfully accepted	Pass
4	Vulnerability Scanning	The system sends the scanning request through the OWASP ZAP API	Vulnerability scanning was successfully executed	Pass
5	Vulnerability Results	The system displays detected vulnerabilities and their risk levels	Scanning results were successfully displayed	Pass
6	Scan History	Completed scanning results are stored in the database	Scan history was successfully stored and displayed	Pass
7	Mitigation Recommendation	The system provides recommendations for detected vulnerabilities	Mitigation recommendations were successfully displayed	Pass
8	PDF Report	The system generates a vulnerability assessment report in PDF format	PDF report was successfully generated	Pass

The results presented in Table 2 show that the main functions of the developed system operated according to the expected behavior. The registration and login functions enabled users to access the application, while the target URL input function allowed the website to be selected for security assessment. The scanning function successfully connected the web application with the OWASP ZAP API and executed the vulnerability assessment process. The vulnerability result function successfully displayed the findings obtained from the scanning process. The information included detected vulnerability, its risk level, description, and affected URL. This enables users to review the security condition of the target website in a structured manner rather than having to interpret the raw response generated by the scanning engine.

The *Scan History* function also operated as expected. Completed scanning activities were stored in the database and could subsequently be accessed through the history page. The stored information includes the target website URL, the number of vulnerabilities according to the *High*, *Medium*, *Low*, and *Informational* categories, and the scanning date and time. This functionality supports documentation and monitoring of security assessments over time. The mitigation recommendation function provides additional information after vulnerabilities are identified. Instead of displaying only the vulnerability findings, the system processes the detected vulnerability information and generates corresponding recommendations. This allows users to use the scanning results as a basis for subsequent security improvements. The function is particularly relevant to previous studies that emphasize the importance of mitigation following vulnerability identification [12] [13] [15].

The use of Black Box Testing in this research is also supported [19], who applied Black Box Testing to

evaluate the security functionality of a web-based application. In the present research, Black Box Testing is used to verify the functional behavior of the vulnerability scanning system, particularly the interaction between user input, OWASP ZAP API processing, and the resulting system output. Furthermore, [20] demonstrated the application of Black Box Testing for evaluating a website system using Boundary Value Analysis. The study supports the use of functional testing to verify whether a web application produces the expected output based on different input conditions. In this research, Black Box Testing is applied to verify the main functions of the vulnerability scanning application, including authentication, URL input, scanning, result presentation, scan history, mitigation recommendations, and report generation.

The final function tested was PDF report generation. The system successfully generated a structured *vulnerability assessment* report containing the target URL, scanning date and time, vulnerability counts according to risk level, *risk score*, overall security status, and detailed information about the detected vulnerabilities. The report also contains the impact and mitigation recommendations associated with each finding, allowing the results to be retained as security assessment documentation. Based on the Black Box Testing results, all tested functions were able to produce outputs corresponding to the expected system behavior. The results indicate that the developed application can support the complete vulnerability assessment workflow, beginning with user authentication and target URL input, followed by scanning through the OWASP ZAP API, presentation and storage of vulnerability findings, provision of mitigation recommendations, and generation of PDF reports. Therefore, the functional testing confirms that the main features of the developed system can operate as intended.

4. Conclusion

The research has successfully developed a web-based website vulnerability scanning system integrated with the OWASP ZAP API. The developed system provides an integrated mechanism for conducting website security assessments through a web interface, allowing users to submit a target website URL and initiate the vulnerability scanning process without directly operating the OWASP ZAP application. The integration enables the system to retrieve and process vulnerability information generated by OWASP ZAP and present the results in a structured format based on vulnerability type and risk level. The results of the vulnerability scanning process demonstrate that the developed system is capable of identifying several security weaknesses, including *Missing Anti-clickjacking Header*, *Content Security Policy (CSP) Header Not Set*, *Absence of Anti-CSRF Tokens*, *Cookie No HttpOnly Flag*, and *Cookie without SameSite Attribute*. The system also provides additional functions in the form of risk-level classification, scan history,

mitigation recommendations, and PDF report generation. These functions allow vulnerability findings to be not only detected but also documented, reviewed, and used as a basis for subsequent security improvements. The functional evaluation using *Black Box Testing* showed that the main functions of the developed application operated according to the expected requirements. The testing covered user registration, authentication, target URL input, vulnerability scanning, result presentation, scan history, mitigation recommendations, and PDF report generation. The results indicate that the developed system can support the website vulnerability assessment workflow in a more structured and centralized manner. Nevertheless, the developed system still has limitations. The vulnerability detection capability depends on the scanning mechanisms and detection rules provided by OWASP ZAP, meaning that automated scanning cannot guarantee the identification of all possible vulnerabilities within a web application. Future research can improve the system by supporting authenticated scanning, expanding vulnerability analysis, integrating additional security testing tools, improving the accuracy of mitigation recommendations, and conducting more comprehensive testing on various types of web applications. Overall, the research demonstrates that OWASP ZAP API integration can be utilized to develop a web-based vulnerability scanning system that supports automated detection, vulnerability management, and security assessment reporting.

References

- [1] Priyawati, D., Rokhmah, S., & Utomo, I. C. (2022). Website Vulnerability Testing and Analysis of Website Application Using OWASP. *International Journal of Computer and Information System (IJCIS)*, 3(3), 142–147. DOI: <https://doi.org/10.29040/ijcis.v3i3.90>.
- [2] Mayasari, R., Ali Ridha, A., Juardi, D., & Ahmad Baihaqi, K. (2020). Analisis Vulnerability pada Website Universitas Singaperbangsa Karawang menggunakan Acunetix Vulnerability. *Systematics*, 2(1), 33. DOI: <https://doi.org/10.35706/sys.v2i1.3450>.
- [3] Hermanto, H., & Haeruddin, H. (2022). Peningkatan Sistem Keamanan Website Menggunakan Metode OWASP. *Jurnal Ilmu Komputer dan Bisnis*, 13(1), 94–104. DOI: <https://doi.org/10.47927/jikb.v13i1.277>.
- [4] Abdul Fattah Hasibuan, Tommy, & Divi Handoko. (2023). Analisis Kerentanan Website dengan Aplikasi Owasp Zap. *Jurnal Ilmu Komputer dan Sistem Informasi*, 2(2), 257–270. DOI: <https://doi.org/10.70340/jirsi.v2i2.51>.
- [5] Tara, T. R., & W, Y. (2023). Analisis Keamanan Website Sistem Informasi Administrasi Kependudukan Menggunakan Metode Vulnerability Assesment. *Jurnal Teknologi Informatika dan Komputer (JURTIKOM)*, 1(1), 1–9. DOI: <https://doi.org/10.51401/jurtikom.v1i1.3172>.
- [6] Zahra, N. A., Zidane, F. H., & Kuslaila, N. R. (2023). Analisis Keamanan Sistem Informasi pada Website PT Sentra Vidya Utama (SEVIMA) Menggunakan Metode OWASP. *Prosiding Seminar Nasional Teknologi dan Sistem Informasi*, 3(1), 384–393. DOI: <https://doi.org/10.33005/sitasi.v3i1.564>.
- [7] Kurniawan, H., & Christianto, E. (2024). Analysis Vulnerability Website Baleomolcreative dengan Metode Penetration Testing Execution Standard & Vulnerability Assessment pada Http Response Header Field. *Jurnal JTik (Jurnal Teknologi Informasi dan Komunikasi)*, 8(3), 734–745. DOI: <https://doi.org/10.35870/jtik.v8i3.2202>.
- [8] Ade Gustiyonoo, Irawadi Alwi, E., & Mubarak Abdullah, S. (2024). Analisa Kerentanan Website terhadap Serangan Cross-Site Scripting (XSS) Metode Penetration Testing. *Cyber Security dan Forensik Digital*, 7(1), 25–33. DOI: <https://doi.org/10.14421/csecurity.2024.7.1.4432>.
- [9] Haeruddin, Gautama Wijaya, Winata, H., Sukma Aji, & Muhammad Nur Faiz. (2024). Website Security Analysis Using Vulnerability Assessment Method. *Journal of Innovation Information Technology and Application (JINITA)*, 6(2), 173–180. DOI: <https://doi.org/10.35970/jinita.v6i2.2476>.
- [10] Rahman, A., Indra, I., Zulkarnaim, N., Mukhram, M., & Rizaldi, A. (2025). Analisis Implementasi Nuclei Vulnerability dan Owasp-Zap Scanner untuk Deteksi Kerentanan Keamanan (Secure System) pada Platform Web Based. *Jurnal Komputer Terapan*, 11(1), 10–15. DOI: <https://doi.org/10.35143/jkt.v11i1.6430>.
- [11] Pramuja Inngam Fanani, G., Muhammad Amirul Mu'min, & Trisanti, N. (2025). Analisis dan Pengujian Kerentanan Website Menggunakan OWASP ZAP. *Jurnal Riset Sistem dan Teknologi Informasi*, 3(1), 36–50. DOI: <https://doi.org/10.30787/restia.v3i1.1886>.
- [12] Eko setiawan, & Fachri, F. (2025). Pengujian dan Mitigasi Kerentanan Website Sistem Informasi Akademik Universitas Ma'arif Nahdlatul Ulama Kebumen dengan OWASP ZAP. *Cyber Security dan Forensik Digital*, 8(1), 25–33. DOI: <https://doi.org/10.14421/csecurity.2025.8.1.5190>.
- [13] Muhamad Wafa Miftahul Anwar, & Fahmi Fachri. (2025). Analisis Keamanan Website Planet Komputer Kebumen Menggunakan Metode Vulnerability Assessment dengan Temuan 4 Kerentanan Tingkat Menengah. *Rabit: Jurnal Teknologi dan Sistem Informasi Univrab*, 10(2), 373–382. DOI: <https://doi.org/10.36341/rabit.v10i2.6083>.
- [14] Nisa, F., Nurfebruary, N. S., & Ikhwan, M. (2024). Analisis Keamanan Sistem Informasi Website Portal Akademik Universitas Malikussaleh Menggunakan OWASP ZAP. *Jurnal Nasional Komputasi dan Teknologi Informasi (JNKTI)*, 7(6). DOI: <https://doi.org/10.32672/jnkti.v7i6.8345>.
- [15] Putra Pura, C. B., Maulana, T. Y., Februri, A., & Ariyadi, T. (2025). Analisis Celah Keamanan Website Menggunakan Tools OWASP ZAP di Kali Linux. *JUSTER: Jurnal Sains dan Terapan*, 4(1), 46–51. DOI: <https://doi.org/10.57218/juster.v4i1.1341>.
- [16] Putra Pratama Hendrico, F., Pahrizal, P., & Utami, M. (2022). Penggunaan Tahapan Prototype Model pada Perancangan Alur Sistem Informasi Desa Taba Mulan. *JUSIBI (Jurnal Sistem Informasi dan E-Bisnis)*, 4(1), 10–19. DOI: <https://doi.org/10.54650/jusibi.v4i1.425>.
- [17] Wulandari, A. S., Saepudin, A., Kinanti, M. P., Sudesi, Z., Saifudin, A., & Yulianti, Y. (2022). Pengujian Aplikasi Sistem Informasi Akademik Berbasis Web Menggunakan Metode Black Box Testing Equivalence Partitioning. *Jurnal Teknologi Sistem Informasi dan Aplikasi*, 5(2), 102. DOI: <https://doi.org/10.32493/jtsi.v5i2.17561>.
- [18] Aura Arnelia Zahrani, Dzihni Safwa Alifah, Yulia Cahyani, & Ilham Albana. (2025). Analisis Vulnerability Assessment pada Sistem Informasi Website IITC Intermedia Universitas Amikom Purwokerto Menggunakan OWASP ZAP. *Bridge: Jurnal Publikasi Sistem Informasi Dan Telekomunikasi*, 3(2), 55–68. DOI: <https://doi.org/10.62951/bridge.v3i2.425>.
- [19] Aura Arnelia Zahrani, Dzihni Safwa Alifah, Yulia Cahyani, & Ilham Albana. (2025). Analisis Vulnerability Assessment pada Sistem Informasi Website IITC Intermedia Universitas Amikom Purwokerto Menggunakan OWASP ZAP. *Bridge: Jurnal Publikasi Sistem Informasi Dan Telekomunikasi*, 3(2), 55–68. DOI: <https://doi.org/10.62951/bridge.v3i2.425>.
- [20] Kartono, F. K., Nursaadah, S., Nugroho, M. R., Tama, D. A., Mashudi, F. A., Wicaksono, A., & Nasir, M. (2024). Pengujian

