Security Analysis of Village Government Website Against Cross-Site Scripting Attacks using Penetration Testing

Dinda Aulia Rizki
Department Informatics
Universitas Ahmad Dahlan
Yogyakarta of Indonesia

Imam Riadi
Department Information System
Universitas Ahmad Dahlan
Yogyakarta of Indonesia

ABSTRACT
The internet is the main source for obtaining various information, both useful and not. A website, also known as a site or portal, is a digital platform consisting of a collection of pages designed to present information in various formats, such as text, still and moving images, animations, and sound. This website serves as an important platform for public services, so it is crucial to protect it from cyber threats. This research aims to analyze and improve the security vulnerabilities of websites against Cross-Site Scripting (XSS) attacks using the Penetration Testing method. The research methodology used in this study includes four essential steps to address potential XSS vulnerabilities on the Purwobakti website. The first step is Preparation, which involves a thorough analysis of the security issues and the development of action plans to address any identified threats. The second step, Scanning, involves a comprehensive scan of all data collected in the previous phase. The third step is Testing, where an in-depth analysis is conducted to identify and evaluate the existing security weaknesses on the website. Finally, the Reporting phase compiles the security testing results into a comprehensive report that provides a complete overview the website security status. The results of this study identified 8 findings: 1 high-risk threat of Server-Side Template Injection (Blind), 1 medium-risk of Content Security Policy (CSP) Header Not Set, Absence Of Anti-CSRF Tokens, and Missing Anti-clickjacking Header, one low-risk threat of Strict-Transport-Security Header Not Set, and 3 informational-level risks, including User Controllable HTML Element Attribute (Potential XSS), Re-examine Cache-control Directives, and Modern Web Application.

Keywords
Cross-Site Scripting, Information security, Open Web Application Security Project, Penetration Testing, Website

1. INTRODUCTION
In the digital era, the internet has made access to information easier. In this ever-evolving digital age, the internet has become an inseparable part of daily life, facilitating access to information and global communication. However, alongside this convenience, serious challenges arise in terms of security, especially for websites that store sensitive data. The Purwobakti village government website is one of the platforms that serves as an important tool for providing public services to the community. Therefore, protecting this website from cyber threats becomes very crucial, especially from Cross-Site Scripting (XSS) attacks that could compromise the integrity of the site and steal important data. The main issue faced is that the Purwobakti village government website has not undergone a comprehensive security evaluation, particularly concerning the threat of XSS attacks. Such attacks can allow an attacker to inject malicious scripts into web pages, which can lead to data manipulation, theft of sensitive information, or even further exploitation of the system.

The lack of adequate security protocol implementation, such as Content Security Policy (CSP) and Anti-CSRF Tokens, exacerbates this situation, making the website vulnerable to various cyber attacks. This research focuses on the analysis of the security of the Purwobakti village government website against XSS attack threats using penetration testing methods that refer to OWASP standards.

(Open Web Application Security Project).This method is applied to identify and address existing security vulnerabilities on the website. The results of this research found eight vulnerabilities on the website, including one with a high risk level, three with medium risk, and four others with low and informational risk. Several vulnerabilities identified include Blind Server-Side Template Injection (SSTI), Content Security Policy (CSP) Header Not Set, and Absence of Anti-CSRF Tokens.By implementing the improvement recommendations generated from this research, it is hoped that the Purwobakti website managers can strengthen their security system. The proposed steps aim to protect the site from XSS attacks and ensure that the public services provided remain safe and reliable. In addition, this research also aims to raise awareness of the importance of information security in today's digital world, particularly in the context of village governance..

2. RELATED WORKS

2.1 Previous Study
Fahmi Fachri, Abdul Fadlil, and Imam Riadi (2021) titled "Analysis of Web Server Security Using Penetration Testing" evaluate the shortcomings and vulnerabilities of web servers in academic information systems (AIS) at universities through penetration testing. The methods used include information gathering, vulnerability assessment, and access testing. The results identify three levels of deficiency: high, medium, and low. This finding helps to understand the patterns of hacker attacks and provides a foundation for web server security measures [6].

Huzain Azis, Farniwati Fattah (2019) on "Analysis of Security Services for Electronic Transaction Card Systems Using Penetration Testing Method" discusses the application of penetration testing on Magnetic Stripe Cards as a means of electronic transactions in amusement parks, compared to RFID. The test results indicate that the system has good security in terms of confidentiality and availability, and it has proven to be quite safe for transactions at that location [7].

Yosua Ade Pohan, Yuhandri Yunus, Sumijan (2021) on "Enhancing the Security of Regional Tax Reporting Application Web Servers Using the Penetration Testing

Execution Standard Method" discusses research that employs the Penetration Testing Execution Standard method, developed by the Pentest Organization, to analyze system security. Previous studies, such as those on the Internet of Things, indicate vulnerabilities across three layers: application, network, and perception, as well as weaknesses in unencrypted passwords and risks of sniffing and spoofing. This result emphasizes the importance of local tax application managers enhancing system security to protect sensitive data and financial transactions. Penetration testing can help identify vulnerabilities and strengthen system protection [8].

Syarif Hidayatulloh, Desky Saptadiaji (2021) on "Penetration Testing on the ARS University Website Using the Open Web Application Security Project (OWASP)," This research analyzes the stages of penetration testing and attacks, with a report based on the OWASP Top 10-2017. From the five subdomains tested using TheHarvester, the results indicate that the security of the ARS University website is rated satisfactory, based on the CIA TRIAD aspects. The website uses SIAKAD Cloud and NiagaHoster servers, which play a role in maintaining adequate security. [9].

Ichsan Ichsan Octama Riandhanu (2022) "Analysis of the Open Web Application Security Project (OWASP) Method Using Penetration Testing on Attendance Website Security" The best method to assess a company's information security is through penetration testing, which helps identify new vulnerabilities. This research demonstrates that web application security analysis using the OWASP method is effective in identifying vulnerabilities in the attendance application sub.domain.com, with 1 critical vulnerability, 3 high, 4 medium, and 7 low. Suggestions for improvement are anticipated to assist the development team in enhancing the application's security and safeguarding sensitive data from potential attacks [10].

3. METHODOLOGY

This research method combines three main approaches: literature study, observation, and interviews. Through the literature review, the research gathers in-depth information on various methods and tools used in web application security, particularly those related to Cross-Site Scripting (XSS) attacks. This review includes an examination of previous studies that identify the best techniques for conducting penetration testing as well as the implementation of OWASP security standards. Additionally, the literature review aims to understand the latest developments in security technology and effective mitigation strategies against XSS attacks. Interviews with experts in web application security were conducted to gain direct insights into recommendations and best practices for protecting web applications from cyber threats. By integrating the findings from both the literature review and interviews, the study gains a better understanding of the most common security vulnerabilities and the strategies to address them. After understanding the theory and concepts, the penetration testing method was practically applied to analyze the security of the Purwobakti village government website. The results of this approach not only revealed eight vulnerabilities on the website, including those with high to informational risk levels but also provided guidelines for strengthening the site's defenses against XSS attack threats.

3.1 Object Study

The research object is the website purwobakti.desa.id, which serves as a platform for village residents to view the transparency of activities carried out by the village government and where the budget is being allocated.

3.2 Research Materials and tools

The selection of appropriate tools and materials is crucial in this research to ensure the smoothness of the process and the validity of the results. This section presents a list of hardware and software used. Here are the tools and materials used, Table 1 lists the hardware, and Table 2 lists the software.

Table 1 : Hardware used in Research

Hardware	Spesification
Laptop	*Acer Aspire A514-53G, RAM 8GB,* Intel Corel i7-1065G7, *256GB SSD, 1TB HDD*

Tabel 2 : Software used in Research

Software	Spesification
Kali Linux	Kali Linux 2.6 (x64bit)
OWASP ZAP	OWASP ZAP 2.15.0 Version

3.3 Research Phase

In conducting web security analysis research using Penetration Testing, several tools are required, namely the software and hardware needed and used during the research. Here are the specifications of the tools and materials that will be used during the research as support for the study to be conducted. By employing these specific tools, the research aims to effectively identify and address potential vulnerabilities in the web application under examination.

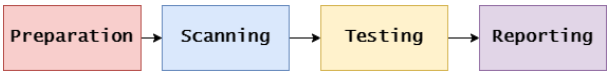

Figure 3 : Research Phase

Figure 3 show the stages of research conducted during the research process.

3.3.1 Preparation

At this stage, the researcher develops a core understanding of the existing problem and designs an appropriate action plan to resolve it, while also involving the collection of data and information about the target, including domain, IP address, host, and other relevant factors for the next stage.

3.3.2 Scanning

At this stage, all the data collected from the previous phase is scanned. In this stage, a scanning process is carried out to identify vulnerabilities present on the Purwobakti regional website. The method used is to search for, discover, and exploit vulnerabilities on regional websites using the OWASP ZAP tool.

3.3.3 Testing

This stage is an advanced phase of Scanning to analyze security weaknesses from various aspects present on the website after using OWASP.

3.3.4 Reporting

At this final stage, the results of the exploitation process are complete, and a report is created to serve as a guideline for web developers in addressing the security vulnerabilities found on the website, so they can determine whether Cross-Site Scripting is possible.

3.4 Scenario Simulation Study

The research scenario is designed to detail the steps that will be taken in the study, with the aim of gaining a clear understanding of the process that will be undertaken.

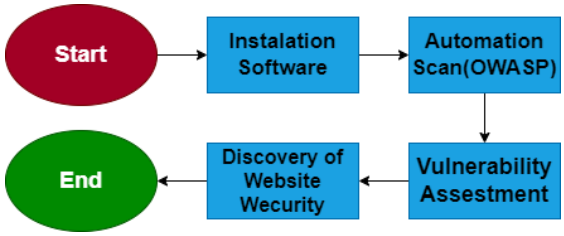

Figure 4 : Ilustration Simulation Study

Figure 4 show an illustration of the scenario or simulation that will be conducted. This illustration includes the stages and overall flow of the research, which consists of three main phases: Pre-Testing, Testing, and Post-Testing. The details of this case simulation illustration are as follows:

3.4.1 Pretesting

In the first stage of the Pre-Testing, it begins with obtaining written permission from the owner of the website to avoid legal violations or accessing the website illegally. At this stage, the author requests permission from the Website Manager to obtain research permission for the website purwobakti.desa.id. Then proceed to the process of downloading and installing OWASP ZAP from the official OWASP website.

3.4.2 Testing

In the second stage of testing, run ZAP after the installation process is complete. Next, open the web browser that will be tested. Configure the proxy settings in the web browser to use ZAP as the proxy. Then, in the Manual Scan feature, set the web browser to route all traffic through ZAP. Scanning by starting the recording of browser activity by pressing the "Attack" button in ZAP and browsing the website using the available features and functions. After finishing exploring the website, press the "Stop" button in ZAP to end the recording process. In addition, you can also simply press the "Attack" button on the Automated Scan feature to perform an automatic scan. The automatic scan referred to here is OWASP ZAP, which will directly explore the target website without the need to manually navigate through the features available on the websitel.

3.4.3 Post Testing

in the third stage of Post-Testing, analyze the scanning results by examining the ZAP scan results that focus on Cross-Site Scripting vulnerabilities, taking into account the severity level and potential impact. After completing the task, prepare a report that explains the findings, including details of vulnerabilities, associated risks, and recommendations for corrective actions. The report was then shared with the owner or developer of the website so that necessary steps could be taken to address the detected vulnerabilities.

4. RESULT AND DISCUSSION

Data collection in this research includes Observation, Literature Study, and Interviews. The observation was conducted using the OWASP ZAP tool to identify information related to vulnerabilities present on the Purwobakti website.

4.1 Implementation

The results of the reporting in the research include Preparation, Scanning, Testing, Reporting (analysis of test results), and recommendations for improvement. In the Scanning section, the researcher used OWASP tools. The type of scanning used in the OWASP tools is automated scanning to identify vulnerabilities on the Purwobakti website.

4.1.1 Preparation Testing

At this stage, it serves as a crucial first step in ensuring the success of the testing process on the subject being examined.

4.1.1.1 Preparation

The image at this stage is the initial step in preparing all the necessary requirements to conduct penetration testing on the subject to support the success of this research. One of them is the installation of the OWASP software, which is the default application already installed on Kali Linux that can be used to find vulnerabilities using automated scanning. If it is not installed, it can be done by following the guide on its official website. https://www.zaproxy.org/download/.

4.1.1.2 Scanning

The scanning process is a crucial stage in identifying potential security vulnerabilities in web-based applications. This research combines automatic and manual scanning methods to obtain comprehensive results. Start active scanning using ZAP to directly evaluate the security of the target website. Provide adequate time for ZAP to complete the scanning process thoroughly. After the scan is complete, the results of the study to identify potential vulnerabilities. Focus the analysis of the scan results on findings related to Cross-Site Scripting vulnerabilities, considering the level of risk and its potential impact. Next, conduct manual testing on the detected vulnerabilities to verify the accuracy and validity of the findings. This process aims to provide a comprehensive and accurate security evaluation of the tested website. The Automatic Scanning stage is the process of identifying vulnerabilities using the Owasp ZAP tool. At this stage, enter the target URL to begin the scan. There are two spider options to choose from, namely the Traditional spider and the Ajax spider. Although the Ajax spider is slower than the traditional spider, it is more effective in the scanning process. The Automatic Scanning Process is show in Figure 5.

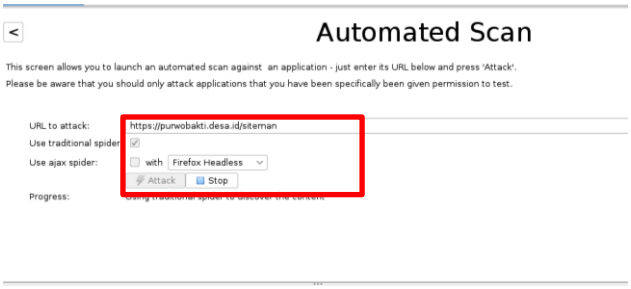

Figure 5 : Scanning Owasp

The scanning process conducted on Figure 5 produces an output displayed in the Alerts section, as illustrated in image 6. Each alert that appears indicates a specific classification of the type of vulnerability detected. Scanning and Testing At this stage, the author conducted testing on the website to scan and test for vulnerabilities present on the website using the features available in OWASP ZAP. Below are the tests conducted using both Automated Scan and Manual Scan.

4.2.2 Scanning and Testing

At this stage, the author conducts testing on the website https://purwobakti.ac.id/ to scan and test for vulnerabilities present on the website using the features available in OWASP ZAP. Below are the tests carried out using both Automated Scan and Manual Scan.:

4.2.2.1 Automated Scan

In an Automated Scan, OWASP ZAP will automatically scan all the features of a website to identify vulnerabilities. This process involves mapping all pages, forms, and other functions of the website, followed by a series of tests to identify various types of security vulnerabilities. With this method, OWASP ZAP can detect issues such as SQL injection, Cross-Site

Scripting (XSS), and other weaknesses without manual intervention, allowing the tester to focus on analysis and remediation.

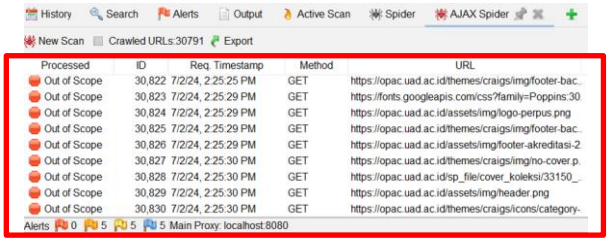

Figure 6 : Scanning to Website

Figure 6 show the process of scanning features and parameters on the website https://purwobakti.desa.id/ using OWASP ZAP to identify any features on the purwobakty website. Besides, also identifies the potential entry point of the attack, vulnerabilities to the features and the parameters that are present on the Purwobacty site. From the scanning shows that the site has some linked parameters such as assets, themes and others. These parameters are linked to each other on the website https://purwobakti.desa.id/ which is then identified by OWASP ZAP to find vulnerabilities to these parameters.

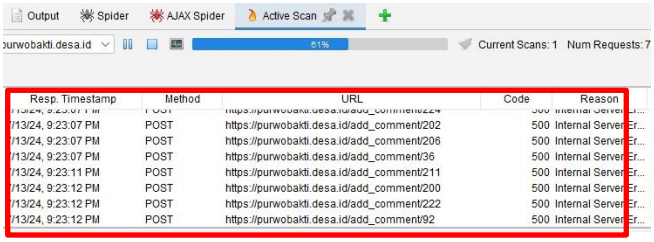

Figure 7 : Scanning Active on Feature Automated Scan

Figure 7 show an active scan of the features and parameters using OWASP ZAP. This process involves a series of actions from OWasp ZAP such as sending a request to an identified entry point through a previous scan to test the vulnerabilities at that entry point. After that, ZAP will display the results of the scans and tests that have been performed. This active scan aims to identify potential security vulnerabilities on various emerging indicating the presence of specific classifications of the type of vulnerability detected. The scan results provide a comprehensive overview of the various security weaknesses that may exist in the systems being analyzed. The vulnerability classification shown helps security professionals quickly identify and prioritize areas that require immediate attention. With this detailed information, security teams can design and implement appropriate mitigation measures to address each type of vulnerability identified. The list of vulnerability alerts show in Figure 8.

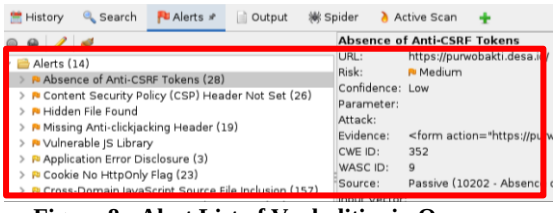

Figure 8 : Alert List of Vurbalities in Owaspzap

4.2.2.2 Testing

In the results of the identification in the Vulnerability Assessment using Owasp Zap, several security vulnerabilities were found. The first one is Server-Side Template Injection (Blind), which is a security vulnerability where an attacker can inject malicious code into a template processed on the server side, without seeing the direct output of that injection. In this attack, the attacker must use inference techniques or timing to determine whether their injection was successful, as there is no direct feedback from the serverIn the results of the identification in the Vulnerability Assessment using Owasp Zap, several security vulnerabilities were found. The first one is Server-Side Template Injection (Blind), which is a security vulnerability where an attacker can inject malicious code into a template processed on the server side, without seeing the direct output of that injection. In this attack, the attacker must use inference techniques or timing to determine whether their injection was successful, as there is no direct feedback from the server.

1. *Server-side template injection (blind)*
   Server-Side Template Injection (SSTI) is a type of security vulnerability that occurs when user input is not properly validated and is directly inserted into server-side templates to be evaluated by the template engine. In the case of "Blind SSTI," this vulnerability occurs without any immediate results visible to the attacker, making it harder to detect. When user input is evaluated by the template engine instead of being used solely as an argument in rendering, it can lead to remote code execution.

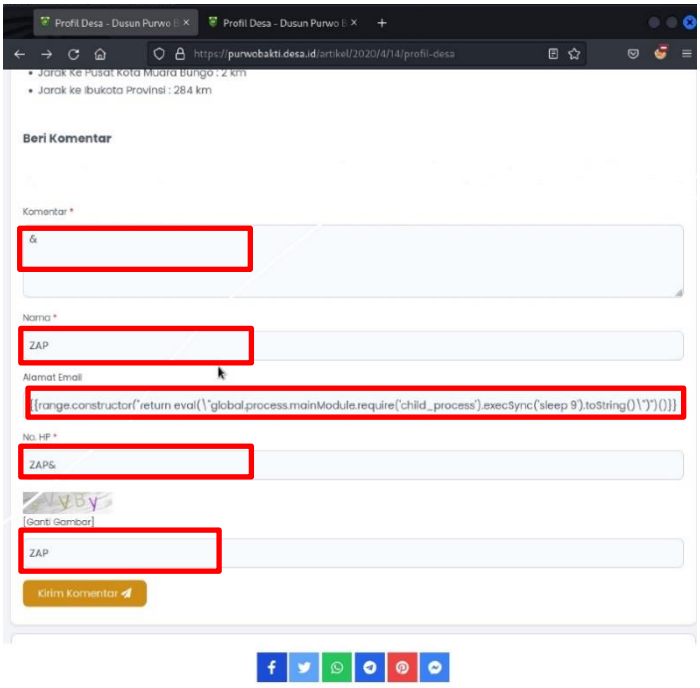

Figure 9 : Try Server-Side Template Injection (Blind) in the Email Column

Figure 9 show test was conducted by entering a script into the email input field to check whether the input parameter is vulnerable to Server-Side Template Injection attacks. (Blind). This test aims to identify whether the web application processes user input unsafely, particularly regarding how that data is used in server-side template rendering. If the script that is entered can be executed, it indicates a potential vulnerability that allows an attacker to run malicious code on the server.

In this test, a thorough observation was conducted to see if the application shows signs of being vulnerable to template injection, even though the server's direct response may not be apparent. This indicates that web applications are at risk if the

server template processes input data insecurely. The steps taken in this testing help identify the extent to which the system is vulnerable to Server-Side Template Injection attacks, as well as provide a foundation for security improvements at the application level.

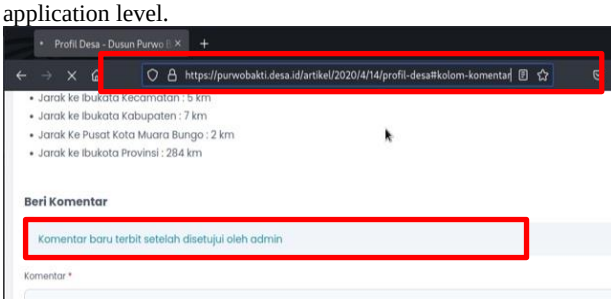

Figure 10 : Server Side Template Injection (Blind) Trial in the Email column

Figure 10 show web responses give an alert "new comment published after admin approval," but when the page is refreshed, the alert does not disappear, indicating that the code is executed by the server. If the user input entered in the email field is processed by the template engine without proper validation, an attacker could inject malicious code that is executed by the server, opening the possibility for server takeover or unauthorized access to sensitive data. Notifications that cannot be dismissed after executing the script indicate that the script affects the performance and responsiveness of the server. This is a direct impact of using the sleep 9 command, which delays all server operations for 9 seconds. This underscores that using eval to execute shell commands opens the possibility for unwanted code execution. In a broader context, this could lead to the server becoming unstable or potentially being exploited for other malicious purposes.

2.Content Security Policy (CSP) Header Not Set

The vulnerability related to Content Security Policy (CSP) does not provide clear details regarding parameters that could potentially create security gaps. Therefore, the researchers decided to shift the focus of the testing to the potential vulnerabilities of Cross-Site Scripting (XSS) elsewhere. However, after further analysis of the provided HTTP headers, it was found that there is no CSP policy explicitly applied.

Typically, the CSP header will appear in the HTTP response as "Content-Security-Policy," which serves to control what resources are allowed in the browser. The absence of this CSP header indicates that the website is at risk of XSS attacks and other types of attacks that can be exploited due to a lack of security policy settings on the content loaded in the browser. Therefore, the researchers recommend that the implementation of CSP policies be carried out comprehensively to strengthen protection against XSS threats and ensure that websites are safer from exploitation that takes advantage of this vulnerability. Based on the scanning results, the vulnerability related to Content Security Policy (CSP) does not provide clear details regarding the parameters that could potentially create security gaps. Therefore, the researchers decided to shift the focus of the testing to the potential vulnerabilities of Cross-Site Scripting (XSS) elsewhere. However, after further analysis of the provided HTTP headers, it was found that there is no CSP policy explicitly applied. Typically, the CSP header will appear in the HTTP response as "Content-Security-Policy," which serves to control which resources are allowed in the browser.

The absence of this CSP header indicates that the website is at risk of XSS attacks and other types of attacks that can be exploited due to the lack of security policy settings on the content loaded in the browser. Therefore, researchers recommend that the implementation of CSP policies be carried out comprehensively to strengthen protection against XSS threats and ensure that the website is more secure from exploitation that takes advantage of this vulnerability.

SIDCSRF stands for Synchronizer Token Pattern for Cross-Site Request Forgery. It is a security technique used to prevent Cross-Site Request Forgery (CSRF) attacks. The Synchronizer Token Pattern works as follows:

1. Strict Transport Security (HSTS) activated.
2. X-Content-Type-Options set to "nosniff".
3. X-Permitted-Cross-Domain-Policies set to "none".
4. Permissions Policy expected token for that session.
5. Cross-Origin Policies (Embedder, Resource, Opener) set to "same-origin".
6. Use of secure cookies with "Secure" and "HttpOnly" flags.
7. SameSite attribute on cookies to prevent CSRF.

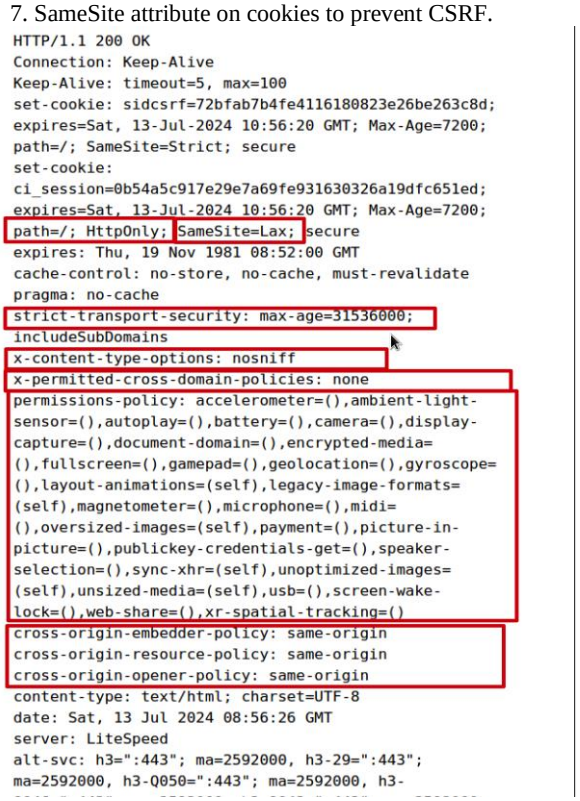

Figure 11 : Respon Body to Website Purwobakti

Figure 11 show security measures are good, adding a CSP will provide an additional layer of security, especially in preventing XSS and the injection of other harmful content. CSP allows you to specify which resources are permitted to be loaded by the browser, which can significantly reduce the risk of XSS attacks. To further enhance security, consider adding a CSP header that aligns with the needs of your web application.

3. Absence Of Anti-CSRF Tokens

The absence of Anti-CSRF tokens is a vulnerability detected in OWASP scanning with a high risk. Based on the explanation from its official website, Owasp ZAP, this vulnerability forces victims to send HTTP requests to a specific destination without their knowledge, exploiting application functionality that uses predictable URL actions or forms. Unlike cross-site scripting (XSS), which exploits the user's trust in a website, CSRF takes advantage of the website's trust in the user. CSRF attacks are effective in certain situations, such as when the victim has an active session, is authenticated via HTTP, or is on the same local network as the target site. Although different from XSS attacks, CSRF techniques can also expose sensitive information, especially when combined with XSS

vulnerabilities. Therefore, researchers are exploring this gap as an aspect of testing for Cross-Site Scripting vulnerabilities Based on image 11, evidence was found in the input form; however, the parameters and attack were not mentioned. Upon investigation, it was found that the form is located on the login form. According to the explanation in the previous paragraph, if this vulnerability is detected due to the absence of CSRF token protection, it has been observed that there is a variable named getCsrfToken. This variable indicates that the CSRF token has been implemented on the website with the name sidcsrf; the token is show in Figure 11. However, OWASP does not capture this variable because the declaration of the token is in the JavaScript tag, as show in Figure 12.

Figure 12 : Token sidcsrf Website purwobakti

SIDCSRF stands for Synchronizer Token Pattern for Cross-Site Request Forgery. It is a security technique used to prevent Cross-Site Request Forgery (CSRF) attacks. The Synchronizer Token Pattern works as follows:

1. A unique token (usually a random string) is generated for each user session
2. This token is inserted into the HTML form as a hidden field.
3. When the form is submitted,
4. The server checks whether the token sent matches the expected token for that session
5. If the token does not match or is not present, the server rejects the request.

This is effective because CSRF attackers cannot know the value of a valid token, as this token is unique for each session and cannot be predicted. After further investigation, researchers found that SIDCSRF is often associated with the CodeIgniter framework, a popular PHP framework. In the context of CodeIgniter:

1. SID may refer to "Session ID" or session identification.
2. CSRF tentu mengacu pada Cross-Site Request Forgery.

CodeIgniter has built-in security features to prevent CSRF attacks, which use synchronization tokens. This feature can be enabled in the security configuration of CodeIgniter

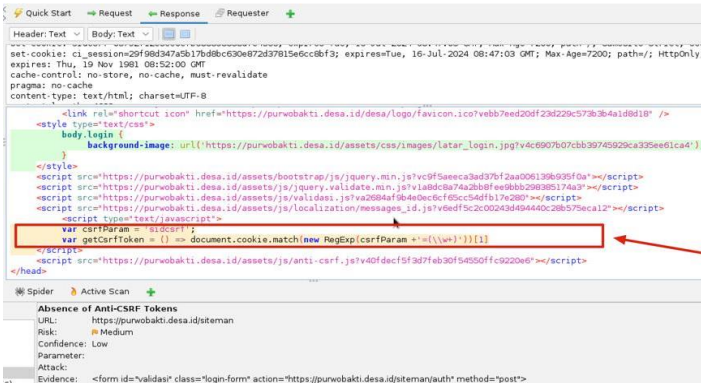

Figure 13 : Discovery Token sidcsrf and Evindence on the Website Purwobakti

Figure 13 show what has been explained in the paragraph above, it is assured that the website has implemented CSRF tokens and that the web is protected by those tokens. Next, the researchers tested the input form by entering the input <script>alert(document.cookie)</script> to observe the response from the object. The web response is show in image 13, where the website rejects the script because it has the potential to capture a session cookie if a user is currently accessing the site. In addition, the website has also implemented rate limiting, which restricts the number of requests that users or client applications can make to the server within a certain time period, as show in Figure 14. Here is a complete explanation of rate limiting:

1. Definition: Rate limiting is a mechanism that controls the rate and frequency of requests received by a web server or API.
2. Here are the objectives regarding rate limiting:
   1. Preventing service abuse
   2. Protecting the server from overload
   3. Ensuring service quality for all users
   4. Saving server resources
   5. Preventing DDoS attacks (Distributed Denial of Service)

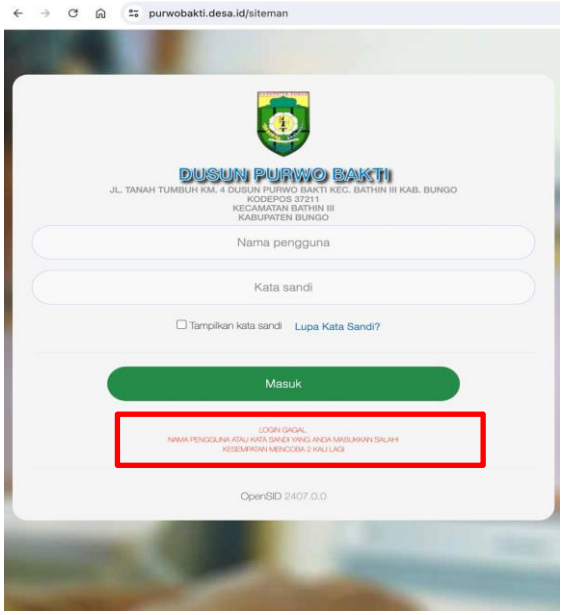

Figure 14: Web Response to XSS Script for Display Cookie

4) Missing Anti-clickjacking Header

The missing Anti-clickjacking Header is a response that does not include Content-Security-Policy with the 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks. The Content-Security-Policy (CSP) with the 'frame-ancestors' directive specifies which sources are allowed to frame content, while X-Frame-Options controls whether a page can be framed by other sites.

Without these two protections, web applications become vulnerable to ClickJacking attacks, where an attacker can trick users into clicking on seemingly legitimate page elements. but in reality interacting with different pages, often with malicious intent. The results of the testing regarding the Missing Anti-clickjacking Header are show in Figure 15.

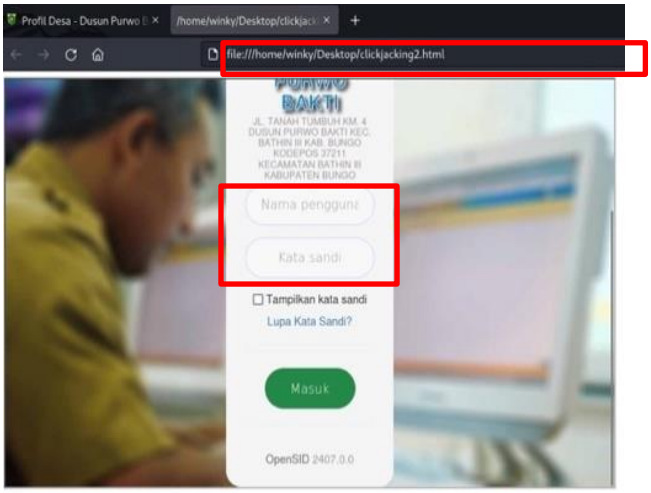

Figure 15 : Missing Anti-Clickjacking

5) Strict-Transport-Security Header Not Set

HTTP Strict Transport Security (HSTS) is a web security policy mechanism that allows a web server to declare that compliant user agents (such as web browsers) should only interact with it using a secure HTTPS connection (that is, HTTP layered with TLS/SSL). HSTS is a protocol established by the IETF and described in RFC 6797.

In the identified case, the to robots.txt does not include the Strict-Transport-Security header, making the website vulnerable to attacks that exploit insecure data transmission.

6) User Controllable HTML Element Attribute (Potential XSS)

A security vulnerability that focuses on analyzing input provided by users, whether through query string parameters or POST data, to identify potential manipulation of HTML element attribute values. This process aims to detect vulnerable points to XSS attacks. (cross-site scripting).

Although this examination may reveal potentially risky areas, the findings require further evaluation by security analysts. The ultimate goal is to determine whether the detected vulnerabilities can actually be exploited, allowing developers to take appropriate preventive measures and enhance the overall security of the application.

This evaluation is important because not all findings are critical, so the results of the analysis must be accompanied by manual verification to ensure their relevance to actual security risks. The ultimate goal is to help developers understand potential threats and take appropriate preventive measures, such as improving input validation or implementing stricter security policies. Thus, the security of web applications can be significantly improved, minimizing the risk of XSS attacks that could compromise system integrity.

This process aims to detect vulnerable areas where an attacker can inject malicious scripts to control HTML elements. Although the results of this examination may indicate at-risk areas, further evaluation by a security analyst is needed to assess whether the detected vulnerabilities can actually be exploited. allowing developers to take appropriate preventive measures and enhance the overall security of the application. Thus, the security of web applications can be significantly improved, minimizing the risk of XSS attacks that could compromise system integrity

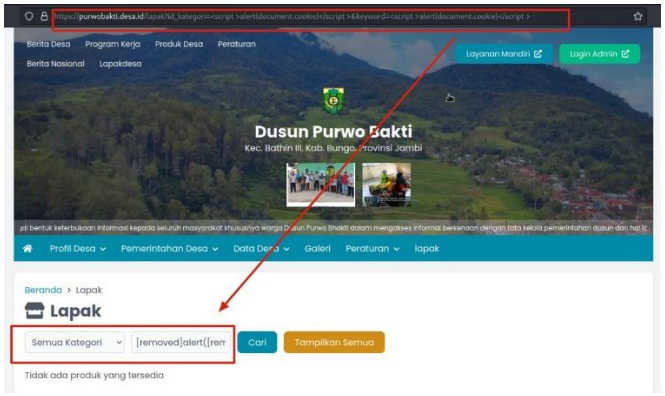

Figure 16 : Testing using the XSS script on the category and keywords parameters

Figure 16 show testing using Scanning OWASP ZAP, the detected vulnerability lies in the parameter id_kategori. However, the response is indicated by a keyword that shows that the field is a search parameter. The researcher entered an XSS payload to determine the response from the web, as shown in image 16. The response [removed]alert(123)[removed] appeared in the web input form after attempting to insert JavaScript code for the attack. Reflected XSS has significant implications for web security:

1. Filtering indication: The appearance of "[removed]" indicates that the website has an active filtering or input sanitization mechanism. The website's security system has detected and removed potentially harmful content.
2. XSS Prevention: The original JavaScript code (alert(123)) has been removed, preventing the execution of unwanted scripts in the user's browser. This is a preventive measure against reflected XSS attacks.
3. Feedback Security feedback: By displaying "[removed]," the website provides feedback that an attack attempt has been detected and blocked, without revealing the details of its security implementation.
4. Further testing needs: This indicates the necessity for additional testing with various XSS payloads to ensure the effectiveness of the protection.
5. Security implementation confirmation: Indicating that the web developer has implemented several security measures, although improvements may be necessary.

Although this indicates the presence of protection, it is important to conduct comprehensive testing to ensure that there are no other security vulnerabilities that could be exploited.

7. Re-examine Cache-control

The Cache-control directive is an important mechanism in web content management that regulates how browsers and proxy servers store and use cached data. In the identified case, the robots.txt file has a cache-control header set as "public, max-age=31536000". This setting indicates that the file can be cached publicly for a full year. Although aggressive caching strategies like this can significantly improve performance for static content, these strategies also carry potential risks, especially if the files contain sensitive information or require more frequent updates. The result shows that there is a discovery in the website directory, show in Figure 17.

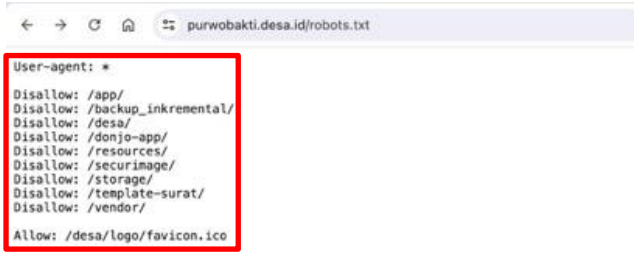

Figure 17 : File robot.txt from website

Improper use of cache-control can lead to several security and functional issues. For example, if the robots.txt contains important instructions about the site's structure or restricted areas, long-term caching can hinder rapid changes in the site's crawling policy.

In addition, if this file accidentally reveals sensitive information, loose cache settings can prolong the exposure of that information. Therefore, it is very important to carefully balance performance needs with security and flexibility requirements when configuring cache-control directives, especially for files that have security implications or are frequently updated.

8) Modern Web Application

This application appears to be a modern web application, indicating that the latest technologies and frameworks have been used in its development. This often involves the use of dynamic rendering techniques and asynchronous content loading, which can enhance the user experience by speeding up page load times and allowing for more responsive interactions. In the context of automated testing and exploration, conventional methods like standard spiders may not be able to effectively handle such dynamic loading.

Links like this are often implemented through JavaScript to enhance flexibility and control over behavior navigation without having to reload the entire page. However, the use of this technique also adds an additional layer of complexity in terms of automated testing and security, as testing tools must be able to understand and interact with the elements manipulated by JavaScript.

4.2 Discussion

The discussion in the research includes an analysis of the results from the testing phase conducted by the testers and provides recommendations for improvements regarding the vulnerabilities that have been successfully identified. The results of the testing analysis (reporting) and recommendations will later be given to the Purwobakti website management, which will serve as evaluation material regarding the security of the website.

Analysis od the tes result reporting

Based on the results of the website security testing, Purwobakti tends to be safe. The detected vulnerabilities, such as the absence of a content security policy (CSP), are not present on the website; however, its security adds several layers that can protect against potential exploitation. The vulnerability of the Absence Of Anti-CSRF Tokens is not proven because the website has already implemented CSRF tokens, as previously explained. Therefore, the validated vulnerability does not exist. User Controllable HTML Element Attribute Vulnerability (Potential XSS) also indicates one of the vulnerabilities that can be exploited, but the website also blocks scripts inputted by researchers, thus mitigating the attack. The vulnerabilities based on the risks obtained using OWASP ZAP scanning will be explained presented in Table 3.

Table 3 : Test Reporting use Owaspzap

No	Types of Threats	Description	Risk Level	Test Result
1	Server Side Template Injection (SSTI) Blind	Inserting malicious code into a server template without directly viewing the output.	*High*	Success, a notification that cannot be dismissed.
2	Content Security Policy (CSP) Header Not Set	The Content Security Policy (CSP) is not implemented on the website.	*Medium*	It didn't work, but there's no CSP.
3	Absence Of Anti-CSRF Tokens	The HTML form is not equipped with an anti-CSRF token to prevent attacks.	*Medium*	It didn't work, the sidcsrf website was applied.
4	Missing Anti click jacking Header	The response does not include protection against ClickJacking attacks with CSP or X-Frame-Options.	*Medium*	Success, X-Frame-Options is not used so the website can be framed.
5	Strict-Transport-Security Header Not Set	The website has not implemented the Strict-Transport-Security header.	*Low*	It didn't work, the website has already been configured.
6	User Control HTML Element Attribute (Potential XSS)	User input validation in query parameters and Post data.	*Information*	It didn't work, the result is "[removed] " in the input field.
7	Re-examine Cache-control Directives	The cache-control header is not set correctly, allowing sensitive content to be cached.	*Information*	It was found that the robots.txt file has a cache-control header set.
8	Modern Web Application	A modern web that uses the latest technology.	*Information*	Not found. The website does not have attributes.

The results of the website security analysis conducted in Table 3 reveal several vulnerabilities and important findings that require serious attention. From the eight types of threats tested,

several vulnerabilities were found with varying risk levels, ranging from high to informational

4.3.2 Recommendation for improvement

Recommendations will be presented in the form of a table containing vulnerability gaps from the testing results that have been confirmed as vulnerabilities that could endanger the system and user. The recommendations are expected to assist the website manager or admin in taking preventive measures against the gaps identified by the tester presented in Table 4.

Table 4 : Recommendation Repair Result Testing

No	Type of Threat	Risk Level	Improvement Recommendation
1	Server Side Template Injection (SSTI) Blind	*High*	Validate and sanitize user input rigorously. Use a secure template engine. Limit access to dangerous functions. Apply the principle of least privilege. Conduct security testing regularly.
2	Content Security Policy (CSP) Header Not Set	*Medium*	Although the security on the Purwobakti website is already good, researchers still recommend adding a Content Security Policy (CSP) that will provide an additional layer of security, especially in preventing XSS and the injection of other harmful content.
3	Absence Of Anti-CSRF Tokens	*Medium*	None, because the website has already implemented Synchronizer Token Pattern protection against Cross-Site Request Forgery. (sidcsrf)
4	Missing Anti-clickjacking Header	*Medium*	Implementing X-Frame-Options with appropriate values (such as 'DENY' or 'SAMEORIGIN')
5	Strict-Transport-Security Header Not Set	*Low*	None, the website has implemented Strict Transport Security (HSTS) which is activated as show in Figure 11
6	User Control HTML Element Attribute (Potential XSS)	*Information*	There is nothing; the website already has an active filtering or input sanitization mechanism. The website's security system has detected and removed potentially harmful content.
7	Re-examine Cache-control Directives	*Information*	Cache-control configuration to ensure that no sensitive information is exposed and crawling policies remain effective.
8	Modern Web Application	*Information*	This is an informational notice, so no changes are necessary.

Recommendations for improvements from vulnerabilities that have been identified through Automated Scans as well as Manual Testing during the testing phase and also the analysis of the test results. Here is a series of mitigations that can be implemented against the identified vulnerabilities.

1. Server Side Template Injection (SSTI) Blind

It is a high-risk vulnerability that requires serious attention. To mitigate this risk, the recommended steps include strict validation and sanitization of user input. This means that every input received from the user must be checked and sanitized to ensure that no harmful code can be executed. The use of a secure template engine is also highly recommended, as these engines typically have built-in mechanisms to prevent injection.

2. Content Security Policy (CSP)

Although the security on the Purwobakti website is already quite good, the addition of a Content Security Policy (CSP) is still highly recommended. CSP serves as a highly effective additional layer of security, especially in preventing Cross-Site Scripting (XSS) attacks and the injection of other harmful content. By implementing CSP, administrators can specifically determine which resources are allowed to be loaded by the user's browser.

3. Absence of Anti-CSRF Tokens

Regarding the Absence of Anti-CSRF Tokens, no corrective recommendations are necessary. The Purwobakti website has implemented adequate protection by using the Synchronizer Token Pattern to prevent Cross-Site Request Forgery (CSRF).

4. Missing Anti-clickjacking Header

To address the vulnerability of Missing Anti-clickjacking Header, the recommended step is to implement the X-Frame-Options header with an appropriate value. Setting the value to 'DENY' or 'SAMEORIGIN' on the X-Frame-Options header can effectively prevent the website from clickjacking attacks. The implementation of this header is a simple yet highly effective step in enhancing website security against framing-based attacks.

5. Strict-Transport-Security Header

Regarding the Strict Transport Security Header, no improvement recommendations are necessary. The Purwobakti website has correctly implemented Strict Transport Security (HSTS), as shown in figure 16. This step demonstrates a strong commitment to the security of communication between users and the server.

6. User Controllable HTML Element Attribute

Regarding the User Controllable HTML Element Attribute (Potential XSS), no remediation recommendations are necessary. The website has demonstrated good capability in handling potential XSS attacks by implementing active input filtering or sanitization mechanisms. The website security system has successfully detected and removed potentially harmful content, as indicated by the response "[removed]" to the attempt at malicious code injection.

7. Re-examine Cache-control Directives

Regarding the User Controllable HTML Element Attribute (Potential XSS), no remediation recommendations are needed. The website has shown good capability in managing potential XSS attacks by implementing active input filtering or sanitization mechanisms. The website security system has successfully detected and removed potentially harmful content, as indicated by the response "[removed]" to the attempt at malicious code injection.

8. **Modern Web Application**

Modern Web Application Findings, this is an informational notice that does not require immediate corrective action. Identifying a website as a modern web application indicates the use of the latest technologies in the development and structure of the website. This indicates that the security testing approach may need to be adjusted, for example by using an AJAX spider instead of a standard spider for a more comprehensive analysis.

5. CONCLUCION

Based on the research conducted by the researcher titled "Analysis of the Security of Village Government Websites Against Cross-Site Scripting Attacks Using Penetration Testing," the results stem from the stages of research that have been carried out, from information gathering, testing, analysis, to reporting. In the research, penetration testing was conducted using OWASP ZAP, which is used to identify security vulnerabilities and aims to enhance the security of the website. The test results found that the website has varying levels of security, with 1 low risk categorized as High, 1 low risk categorized as Low, and there are also 3 risks classified as medium and informational. Of the 5 vulnerabilities or risks, the website https://purwobakti.id/ still has a gap or potential for Server-Side Template Injection (SSTI) Blind attacks, which poses a very high risk of attack. And the implementation of the OWASP method in analyzing and identifying vulnerabilities present on the website, It runs well, as evidenced by the testing of the website, which has been proven to be secure due to the absence of vulnerabilities such as Content Security Policy (CSP) on the website. However, its security adds several layers that can protect against potential exploitation. The vulnerability of Absence Of Anti-CSRF Tokens has not been proven because the website has already implemented CSRF tokens, as previously explained; therefore, the validated vulnerability does not exist. User Controllable HTML Element Attribute Vulnerability (Potential XSS) also indicates one of the vulnerabilities that can be exploited, but the website also blocks scripts inputted by researchers, thus mitigating the attack.

6. REFERENCES

[1] J. J. B. H. Yum Thurfah Afifa Rosaliah, “Pengujian Celah Keamanan Website Menggunakan Teknik Penetration Testing dan Metode OWASP TOP 10 pada Website SIM,” Senamika, vol. 2, no. September, pp. 752–761, 2021.

[2] J. T. Elektro and P. N. Medan, “Perancangan Website Pada Pt. Ratu Enim Palembang,” pp. 15–27,

[3] Muhammad Isfa Hany, Adhitya Bhawiyuga, and Ari Kusyanti, “Implementasi Cross Site Scripting Vulnerability Assessment Tools berdasarkan OWASP Code Review,” J. Pengemb. Teknol. Inf. dan Ilmu Komput, vol. 5, no. 9, pp. 3745–3753, 2021.

[4] B. Sakti, A. Aziz, and A. Doewes, “Uji Kelayakan Implementasi SSH sebagai Pengaman FTP Server dengan Penetration Testing,” J. Teknol. Inf. ITSmart, vol. 2, no. 1, p. 44, 2016, doi: 10.20961/its.v2i1.620.

[5] I. M. Edy Listartha, I. M. A. Premana Mitha, M. W. Aditya Arta, and I. K. W. Yuda Arimika, “Analisis Kerentanan Website SMA Negeri 2 Amlapura Menggunakan Metode OWASP (Open Web Application Security Project),” Simkom, vol. 7, no. 1, pp. 23–27, 2022, doi: 10.51717/simkom.v7i1.63.

[6] F. Fachri, A. Fadlil, and I. Riadi, “Analisis Keamanan Webserver menggunakan Penetration Test,” J. Inform., vol.8,no.2,pp.183–190,2021,doi:10.31294/ji.v8i2.1085 4.

[7] H. Azis and F. Fattah, “Analisis Layanan Keamanan Sistem Kartu Transaksi Elektronik Menggunakan Metode Penetration Testing,” Ilk. J. Ilm., vol. 11, no. 2,pp.167–174,2019,doi:10.33096/ilkom.v11i2.447.1 67-174.

[8] Y. A. Pohan, “Meningkatkan Keamanan Webserver Aplikasi Pelaporan Pajak Daerah Menggunakan Metode Penetration Testing Execution Standar,” J. SistimInf.danTeknol.,vol.3,pp.16,2021,doi:10.37034/jsisfotek.v3i1.3 6.

[9] S. Hidayatulloh and D. Saptadiaji, “Penetration Testing pada Website Universitas ARS Menggunakan Open Web Application Security Project (OWASP),” J. Algoritm.,vol.18,no.1,pp77–86,2021,doi:10.33364/algoritma/v.18-1.827.

[10] I. O. Riandhanu, “Analisis Metode Open Web Application Security Project (OWASP) Menggunakan Penetration Testing pada Keamanan Website Absensi,” J. Inf. dan Teknol., vol. 4, no. 3, pp. 160–165, 2022, doi: 10.37034/jidt.v4i3.236.

[11] S. Nurul, S. Anggrainy, and S. Aprelyani, “Faktor-Faktor Yang Mempengaruhi Keamanan Sistem Informasi\: Keamanan Informasi , Teknologi Informasi Dan Network (Literature Review Sim),” J. Ekon. Manaj. Sist. Inf., vol. Vol. 3, no. No. 5, pp. 564–573, 2022.

[12] A. H. Harahap, C. Difa Andani, A. Christie, D. Nurhaliza, and A. Fauzi, “Pentingnya Peranan CIA Triad Dalam Keamanan Informasi dan Data Untuk Pemangku Kepentingan atau Stakholder,” J. Manaj. dan Pemasar. Digit., vol. 1, no. 2, pp. 73–83, 2023.

[13] M. Kamil, B. Rahmat, and O. Primadianti, “Perancangan Dan Implementasi Web Server Untuk Pemantauan Kualitas Air Berbasis Iot,” e-Proceeding Eng., vol. 8, no. 6, p. 3515, 2022.

[14] Y. Mulyanto and A. A. Fari, “Analisis Keamanan Login Router Mikrotik dari Serangan Brute Force Menggunakan Metode Penetration Testing,” J. Inform. Teknol. dan Sains, vol. 4, no. No.3, pp. 145–155, 2022.

[15] M. D. Al Vriano, “Pengujian Keamanan Web Juice Shop Dengan Metode Pentesting Berbasis Owasp Top 10,” J. Multidisiplin Saintek, vol. 1, no. 06, pp. 81–90, 2023.

[16] M. Hasibuan and A. M. Elhanafi, “Penetration Testing Sistem Jaringan Komputer Menggunakan Kali Linux untuk Mengetahui Kerentanan Keamanan Server dengan Metode Black Box,” sudo J. Tek. Inform., vol. 1,no.4,pp.171–177, 2022, doi: 10.56211/sudo.v1i4.160.

[17] C. Alderi Jeffta Soewoeh et al., “Analisa Kerentanan Website FMIPA UNSRAT Berdasarkan Open Web Application Security Project Top 10 Framework,” JECSIT J. Eng. Comput. Sci. Inf. Technol., vol. 2, no. 2, pp. 2797–5045, 2022, [Online]. Available:http://jurnal.teknokrat.ac.id/index.php/JECSIT/article/view/251.

[18] M. A. Mu’min, A. Fadlil, and I. Riadi, “Analisis Keamanan Sistem Informasi Akademik Menggunakan Open Web Application Security Project Framework,” J. Media Inform. Budidarma, vol. 6, no. 3, p. 1468, 2022, doi: 10.30865/mib.v6i3. 4099.

[19] H. Haikal Muhammad, A. Id Hadiana, and H. Ashaury, “Pengamanan Aplikasi Web Dari Serangan Sql Injection Dan Cross Site Scripting Menggunakan Web Application Firewall,” JATI (Jurnal Mhs. Tek. Inform., vol.7,no.5,pp.3265–3273,2024,doi:10.36040/jati.v7i5.7320.

[20] B. I. Dewangkara, K. S. Santi, V. A. Putri, and I. M. E. Listartha, “Penerapan Analisis Kerentanan XSS dan Rate Limiting pada Situs Web MTsN 3 Negara Menggunakan

OWASP ZAP," J. Inform. Upgris, vol.8,no.1,pp.92–97,2022,doi:10.26877/jiu.v8i1.102 66.

[21] S. Suroto and A. Asman, "Ancaman Terhadap Keamanan Informasi Oleh Serangan Cross-Site Scripting (Xss) Dan Metode Pencegahannya," Zo. Komput.,vol.11,no.1,pp.1119,2021,[Online].Available:http://www.hackers.com ?yid=

[22] I. M. Suartana, H. Endah Wahanani, and A. Noor Sandy, "Sistem Pengaman Web Server Dengan Application Firewall (WAF)," Scan, vol. X, no. 1, pp. 3–8, 2015

[23] A. S. Hakim, T. A. Cahyanto, and H. Azizah, "Serangan cross-site scripting (XSS) berdasarkan base metric CVSS V.2," J. Smart Teknol., vol. 2, no. 1, 2020.

[24] N. I. Aspriantama, "Pengujian Keamanan Sistem Informasi Uajy Menggunakan Penetration Testing," 2021,[Online].Available:http://ejournal.uajy.ac.id/id/eprint/24753

[25] Harry Dwi Sabdho and Ulfa Maria, "Analisis Keamanan Jaringan Wireless Menggunakan Metode Penetration Testing Pada Kantor PT. Mora Telematika Indonesia Regional Palembang," Semhavok, vol. 1, no. 1, pp. 15–24, 2018.