

Security Testing of the Systematic UNISI Website using the Penetration Testing Method

Agung Ramadhan^{1*}, Abdul Muni², Bayu Rianto³

^{1,2,3}Department of Information Systems, Faculty of Engineering and Computer Science, Universitas Islam Indragiri
adhan2361@gmail.com^{1*}, abdulmuni@gmail.com², rianto.bayu91@gmail.com³

Abstract

Academic websites require continuous security evaluation because they support information publication and institutional services that are accessible through the internet. This study evaluates the security of the Sistemasi UNISI website using a non-destructive penetration testing method with OWASP Zed Attack Proxy (ZAP). The research stages consisted of literature study, planning, reconnaissance, scanning, vulnerability testing, report analysis, conclusion, and remediation formulation. Data were collected through literature review and observation of the target website, while testing was performed by entering the Sistemasi UNISI URL into the Automated Scan feature of OWASP ZAP. The detailed scan report identified 19 findings consisting of four medium-risk findings, eight low-risk findings, and seven informational findings, with no high-risk finding reported. The medium findings were absence of Anti-CSRF tokens, Content Security Policy header not set, missing anti-clickjacking header, and vulnerable JavaScript library. The results indicate that the website still requires security hardening, especially in browser security headers, cookie configuration, JavaScript dependency management, HTTPS consistency, and information disclosure control. The recommended improvements can reduce exploitation opportunities and strengthen the security posture of the website.

Keywords: Cybersecurity, OWASP ZAP, Penetration testing, Security headers, Vulnerability assessment

1. Introduction

Universitas Islam Indragiri (UNISI) is an academic institution that uses digital platforms to support education, publication, and institutional services. One of the web-based platforms used in this environment is Sistemasi UNISI, a website that provides access to academic and scientific information. Because it can be accessed from inside and outside the institution, the website must be managed not only from the perspective of usability and content availability, but also from the perspective of security. A public website that is not properly secured can expose the institution to illegal access, content manipulation, data leakage, and service disruption [1][2].

The use of web technology produces clear benefits because users can reach the system through a browser without installing a specific application. However, this flexibility also increases the attack surface. A website normally includes a web server, programming framework, database, user interface components, session cookies, security headers, and third-party resources. A weakness in one component may not directly compromise the system, but it can assist attackers in reconnaissance or can enlarge the impact of another weakness. For this reason, security evaluation should be performed periodically, especially on websites that support academic services and public information [3][4].

Website security is closely related to confidentiality, integrity, and availability. Confidentiality requires that sensitive data be protected from unauthorized access. Integrity requires that information and system behavior are not modified without permission. Availability requires that the service remains accessible for legitimate users. In a university environment, the loss of one of these properties can reduce user trust and damage institutional credibility. Security controls such as HTTPS, secure cookies, browser security headers, access control, and safe input handling are therefore necessary parts of responsible web management [5][6].

Penetration testing is an authorized and structured method for evaluating system security by imitating attacker-oriented techniques within an agreed scope. The objective is not to damage the system, but to identify vulnerabilities before they are abused by unauthorized parties. A penetration testing activity commonly includes planning, information gathering, reconnaissance, scanning, vulnerability identification, analysis, reporting, and remediation planning. The result of a good penetration test should help administrators understand the weaknesses found and the actions required to improve security [7][8].

OWASP Zed Attack Proxy (OWASP ZAP) is an open-source web application security testing tool developed under the OWASP community. ZAP can function as a proxy between a browser and a web application, allowing HTTP requests and responses to be observed,

inspected, and tested. The tool provides automated scanning, spidering, passive scanning, active scanning, alert classification, and reporting features. In academic research, OWASP ZAP is frequently used because it is accessible, repeatable, and able to detect common web security issues such as missing headers, cookie weaknesses, vulnerable libraries, and information disclosure [9][10].

Previous studies show that web applications in education, government, and business sectors often experience recurring weaknesses. Rosaliah et al. identified vulnerabilities related to authentication, sensitive data exposure, security misconfiguration, and clickjacking [33]. Ghozali et al. used OWASP-based assessment to determine risk levels using likelihood and impact [34]. Sofyan et al. found high, medium, and low vulnerabilities on an institutional website [35]. Fachri et al. demonstrated that penetration testing can reveal serious weaknesses on academic web servers [36]. These studies indicate that automated and structured security testing is useful as an initial baseline for improvement.

The research gap addressed in this study is the need for a focused security analysis of the Sistemasi UNISI website using the penetration testing method. The study does not only list alerts generated by OWASP ZAP, but also interprets the severity level, potential impact, and recommended controls based on the findings. Therefore, this research aims to identify security vulnerabilities on the Sistemasi UNISI website, analyze the effectiveness of OWASP ZAP as a testing tool, and provide technical recommendations that can be used by website administrators to strengthen security.

1.1. Literature foundation

A website is a collection of connected pages containing information in the form of text, images, audio, animation, and other digital content. A web-based application can be accessed through a browser and depends on internet connectivity, making it flexible for users across different operating systems [12][13]. A system information, on the other hand, is a set of interconnected components used to process transactions, support management activities, and provide reports for stakeholders [15]. When a website supports information system functions, security must be considered from the interface, server, database, and communication layers.

Penetration testing is defined as a legitimate attempt to evaluate and exploit weaknesses in a computer system in order to improve its protection [16]. In practice, penetration testing uses procedures and techniques to simulate attacks and identify security gaps [18]. The findings obtained from testing can be converted into recommendations for correction and prevention [17]. This principle is consistent with the present study, because the scan results are used to design practical solutions for the Sistemasi UNISI website.

OWASP provides guidance and tools for improving application security. OWASP ZAP is one of the tools that supports both beginner and advanced testers. It can be used for spidering, passive analysis, active scanning, and report generation. In the context of this research, OWASP ZAP is selected because the study focuses on web application vulnerabilities and because the tool can detect issues related to security headers, cookies, sensitive information, authentication patterns, session management, and vulnerable client-side components [21][23][24].

2. Research method

This study used a descriptive security assessment approach with a penetration testing procedure. The testing was non-destructive and limited to identifying vulnerabilities on the Sistemasi UNISI website. The main tool used in the research was OWASP ZAP. The research process followed several stages, namely planning, reconnaissance, scanning, vulnerability testing, report analysis, conclusion, and solution formulation. These stages were selected to keep the evaluation systematic and to ensure that each finding could be connected to a specific recommendation [37].

The object of the study was the Sistemasi UNISI website at <https://sistemasi.ftik.unisi.ac.id>. The website was selected because it is part of the academic digital environment of Universitas Islam Indragiri and is publicly reachable. The scope of the assessment was restricted to web application scanning using OWASP ZAP. The study did not conduct destructive attacks, denial-of-service testing, brute-force attacks, social engineering, or unauthorized data modification. These limitations were applied to maintain ethical and safe testing boundaries.

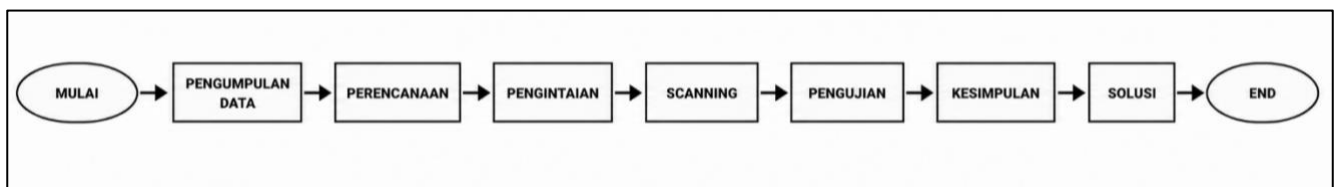


Fig. 1: Research flow of the penetration testing procedure

2.1. Data collection and testing stages

Data collection was carried out through literature study and observation. The literature study reviewed journals, books, proceedings, final projects, and digital sources related to website security, cybersecurity, penetration testing, OWASP, and OWASP ZAP. The literature was used to support the interpretation of web vulnerabilities and to formulate appropriate recommendations. Observation was conducted by accessing and examining the Sistemasi UNISI website before testing, including the visible structure, changes in appearance, and general behavior of the website.

The planning stage determined the scope, target, tool, and testing limitation. The reconnaissance stage collected initial information about the target website without harming the system. The scanning stage used OWASP ZAP to inspect web responses, links, cookies, headers, and other web application behavior. The vulnerability testing stage produced alerts that were then analyzed according to risk level. The final stages were conclusion and solution formulation, in which the findings were summarized and translated into security improvement recommendations.

2.2. Testing environment

Table 1: Research materials and testing environment

Component	Description
Hardware	Lenovo 82h7 laptop with 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80 GHz processor and 8 GB RAM.
Operating system	Windows 11, 64-bit operating system, x64-based processor.
Testing software	OWASP ZAP, used to scan and analyze web application vulnerabilities.
Target website	Sistemasi UNISI website at https://sistemasi.ftik.unisi.ac.id .
Testing type	Non-destructive penetration testing and vulnerability assessment using automated scan features.

2.3. OWASP ZAP testing procedure

The practical testing procedure began by opening the OWASP ZAP application. The tester then selected the Automated Scan feature from the Quick Start interface and entered the URL of the Sistemasi UNISI website in the URL to attack field. OWASP ZAP then performed discovery and analysis on reachable pages and generated an alert report. The dashboard interface during the research is shown in Fig. 2.

The scan involved spidering and alert generation. Spidering was used to browse reachable links and resources within the target scope. Passive analysis reviewed traffic and responses without modifying requests, while automated active testing submitted test requests to detect selected vulnerability patterns. The resulting alerts were classified into risk categories, namely medium, low, and informational. The findings were then reviewed and rewritten into a structured discussion to make the results clearer for the journal article.

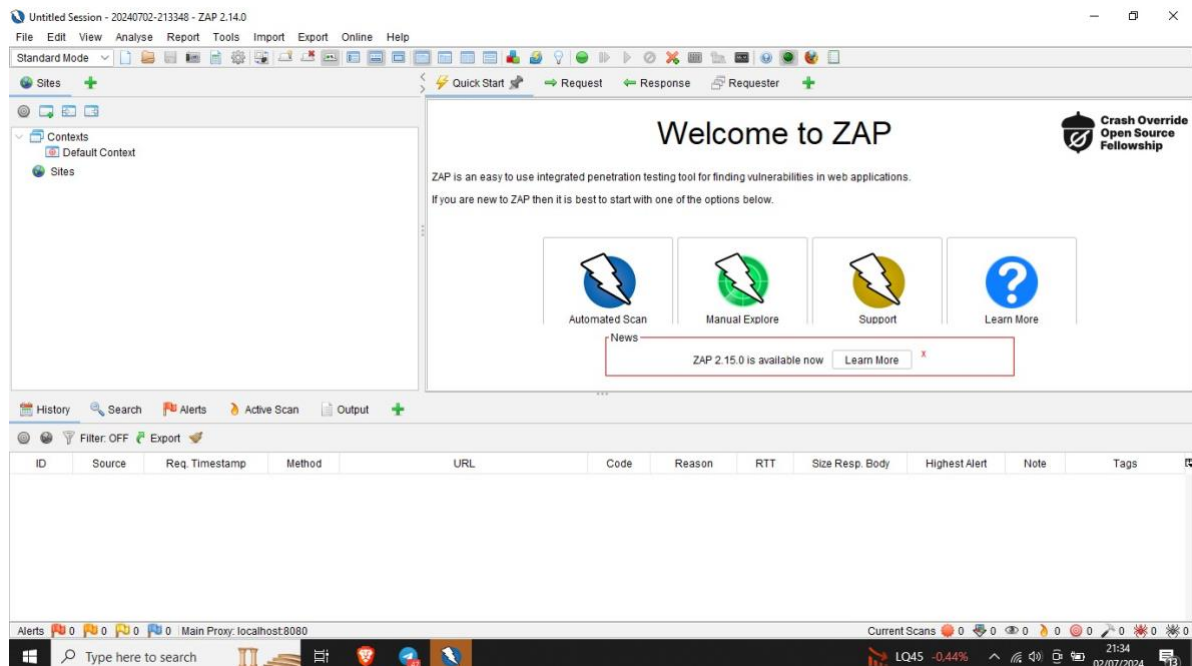


Fig. 2: OWASP ZAP dashboard and Automated Scan interface

3. Results and discussion

The results of the OWASP ZAP testing show that the Sistemasi UNISI website still has several security weaknesses that need attention. The report interface generated by the tool is shown in Fig. 3. The thesis report states that the testing produced alert information from OWASP ZAP and then presented the detailed findings in a table consisting of medium-risk, low-risk, and informational categories. In this article, the detailed findings are normalized into 19 finding entries based on the names, risk levels, descriptions, and solutions contained in Chapter IV of the thesis. No high-risk finding was presented in the detailed report. This is a positive indication within the scope of the automated scan, but it does not mean that the website is fully secure. Automated scanning cannot fully evaluate business logic, role-based authorization, source code quality, or authenticated workflows. Therefore, the results should be interpreted as a baseline for hardening rather than as a final security certificate.

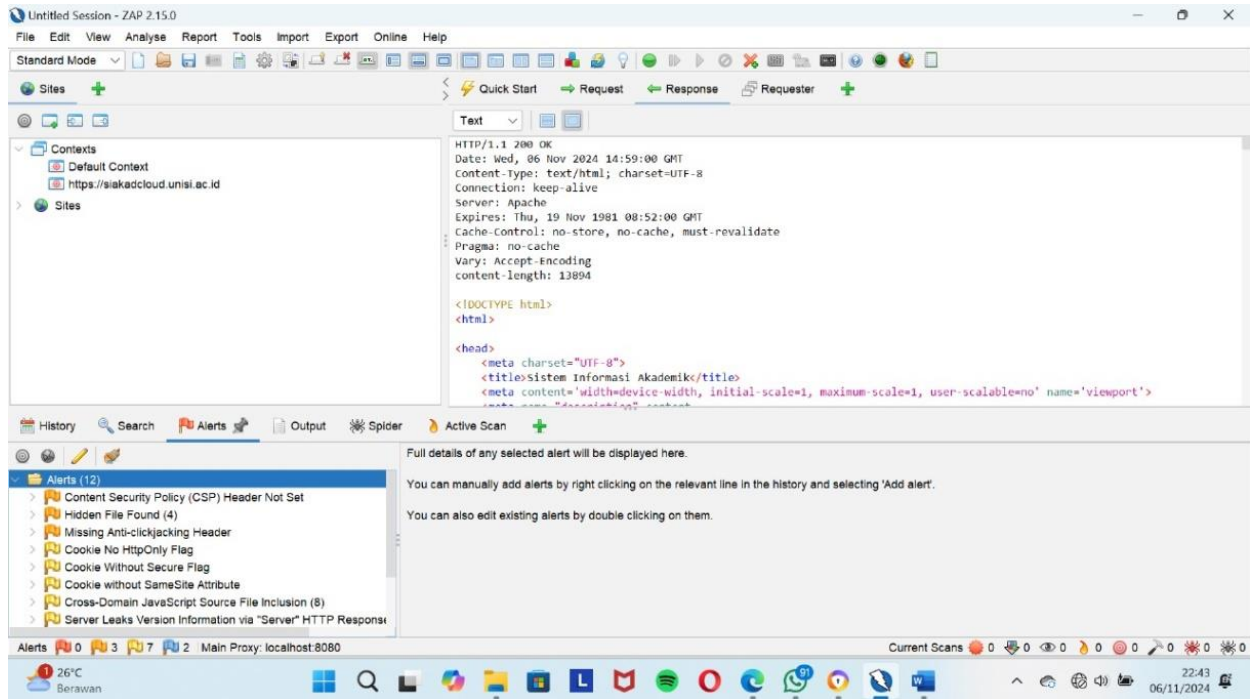


Fig. 3: OWASP ZAP result interface used during the assessment

3.1. Scan summary

Table 2: Summary of OWASP ZAP findings by severity

Severity	Number	Percentage	Interpretation
High	0	0.00%	No high-risk finding was listed in the detailed thesis report.
Medium	4	21.05%	Important weaknesses requiring early remediation because they can support CSRF, XSS impact, clickjacking, or vulnerable dependency exploitation.
Low	8	42.11%	Configuration and information-exposure weaknesses that can assist attack chains and weaken defensive controls.
Informational	7	36.84%	Observed application behaviors that require contextual review, manual validation, or administrative attention.
Total	19	100%	Detailed finding entries summarized from the OWASP ZAP report table in Chapter IV.

Severity distribution of OWASP ZAP findings

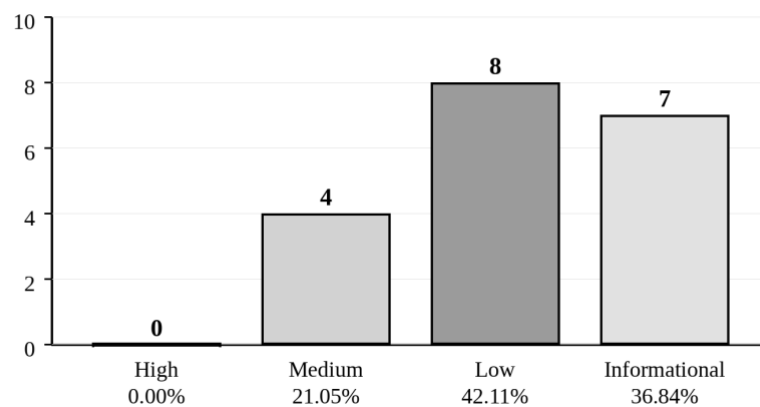


Fig. 4: Severity distribution based on 19 detailed finding entries

The distribution shows that low-risk findings formed the largest group. Even though these findings are not immediately critical, they are still relevant because attackers may combine low and medium weaknesses. For example, missing security headers can increase the impact of script injection, weak cookie attributes can expose session data, and information disclosure can help attackers map the system. Therefore, remediation should be organized by priority and verified through repeated testing.

3.2. Detailed OWASP ZAP findings

Table 3: Detailed analysis and recommendations for OWASP ZAP findings

No.	Finding	Risk	Observed weakness	Recommended control
1	Absence of Anti-CSRF Tokens	Medium	No Anti-CSRF token was found in HTML submission forms, creating potential exposure to forged requests from authenticated users.	Add unpredictable per-request or per-form CSRF tokens, validate them on the server, avoid state-changing actions through GET, and review XSS risks because XSS can bypass CSRF defenses.
2	Content Security Policy Header Not Set	Medium	The response did not define a Content Security Policy to restrict scripts, frames, styles, and other browser-loaded resources.	Deploy a restrictive Content-Security-Policy, start with Report-Only mode, define trusted sources, and use frame-ancestors to support clickjacking prevention.
3	Missing Anti-clickjacking Header	Medium	The response did not include X-Frame-Options or CSP frame-ancestors to prevent unauthorized framing.	Use X-Frame-Options DENY or SAMEORIGIN and implement CSP frame-ancestors on all relevant pages.
4	Vulnerable JS Library	Medium	A JavaScript library used by the website was identified as vulnerable or outdated.	Upgrade the affected library to the latest safe version, remove unused dependencies, and maintain dependency monitoring.
5	Big Redirect Detected	Low	The server returned a large redirect response, which may expose content that should not be sent with a redirect.	Ensure redirect responses contain minimal body content and do not reveal sensitive information.
6	Cookie Without SameSite Attribute	Low	A cookie was set without an explicit SameSite attribute, allowing cross-site requests to carry the cookie.	Set SameSite=Lax or SameSite=Strict as appropriate and combine it with CSRF tokens for state-changing requests.
7	Cross-Domain JavaScript Source File Inclusion	Low	The page included one or more JavaScript files from third-party domains.	Load scripts only from trusted sources, document external dependencies, apply Subresource Integrity when possible, and restrict sources through CSP.
8	Server Leaks Version Information via Server Header	Low	The Server HTTP response header disclosed web server or application server information.	Suppress or generalize server banners and keep the server, modules, and framework patched.
9	Strict-Transport-Security Header Not Set	Low	The website did not instruct browsers to enforce future HTTPS connections through HSTS.	Enable Strict-Transport-Security after confirming that all pages and subresources work correctly over HTTPS.
10	X-Content-Type-Options Header Missing	Low	The response did not set X-Content-Type-Options: nosniff, allowing MIME sniffing in some browsers.	Set accurate Content-Type headers and add X-Content-Type-Options: nosniff to responses.
11	Cookie No HttpOnly Flag	Low	A cookie was set without the HttpOnly flag, allowing access through client-side JavaScript.	Set HttpOnly for session and sensitive cookies to reduce the impact of XSS on session theft.
12	Secure Pages Include Mixed Content	Low	A secure page loaded content through HTTP instead of HTTPS.	Serve all scripts, images, styles, and third-party resources through HTTPS and remove insecure subresource links.
13	Authentication Request Identified	Informational	ZAP identified a request pattern associated with authentication.	Review authentication manually, including HTTPS enforcement, error messages, rate limiting, and account protection.
14	Information Disclosure - Sensitive Information in URL	Informational	A request appeared to contain sensitive information in the URL.	Do not place sensitive data in URI parameters; use secure request bodies and server-side session mechanisms.
15	Information Disclosure - Suspicious Comments	Informational	The response contained suspicious comments that may help an attacker understand the system.	Remove comments that reveal implementation details, credentials, file paths, or debugging information.
16	Modern Web Application	Informational	ZAP detected behavior typical of a modern web application that may require Ajax Spider for deeper crawling.	Use Ajax Spider or authenticated crawling in future testing when dynamic content must be explored.
17	Re-examine Cache-control Directives	Informational	Cache-control headers were missing or not configured properly for sensitive content.	For sensitive content, use no-cache, no-store, must-revalidate; for public assets, use controlled caching policies.
18	Session Management Response Identified	Informational	ZAP identified a response containing a session-management token.	Review session renewal, expiration, logout invalidation, fixation resistance, and secure cookie configuration.
19	User Controllable HTML Element Attribute (Potential XSS)	Informational	User-controlled input appeared in an HTML attribute context, requiring review for possible XSS.	Validate input and encode output before writing user-controlled values into HTML attributes.

3.3. Analysis of medium-risk findings

The first medium-risk finding was the absence of Anti-CSRF tokens. Cross-Site Request Forgery occurs when a victim who has an active session is forced to send an HTTP request to a target application without intending to perform the action. The threat becomes important when the application has predictable state-changing forms or URLs. If a user is already authenticated, a forged request can potentially perform an action using the user privileges. The recommended control is to generate unique and unpredictable tokens, place them in forms, and validate the tokens on the server before accepting any state-changing request.

The second medium-risk finding was Content Security Policy Header Not Set. Although the thesis table duplicated part of the Anti-CSRF description under this item, the correct technical interpretation is that the website did not define a browser policy that restricts where executable scripts, styles, images, objects, or frames may be loaded from. CSP is a defense-in-depth mechanism because it can limit the impact of XSS and unauthorized resource loading. The policy should be tested gradually through Report-Only mode before enforcement so that legitimate resources are not blocked unexpectedly.

The third medium-risk finding was Missing Anti-clickjacking Header. This issue occurs when a page can be embedded in a frame by another website. Attackers may use transparent layers or deceptive interfaces to trick users into clicking a hidden control on the real website. This risk is especially relevant for pages that require authentication or perform data-changing actions. The recommended solution is to apply X-Frame-Options with DENY or SAMEORIGIN and use the CSP frame-ancestors directive for modern browser control.

The fourth medium-risk finding was Vulnerable JS Library. Client-side libraries are commonly used to simplify web development, but outdated libraries may contain publicly known vulnerabilities. Even when the server code is secure, a vulnerable library can provide an entry point for client-side attacks or compatibility weaknesses. The administrator should identify the affected library version, upgrade it, remove unused scripts, and create a regular dependency maintenance process.

3.4. Analysis of low-risk findings

The low-risk findings were dominated by configuration and hardening issues. Big Redirect Detected indicates that the server returned a redirect response with a large body. Redirects should generally contain only minimal information because unnecessary response content can reveal sensitive information, paths, or application behavior. The administrator should review redirect responses and ensure that no confidential content is sent in them.

Cookie Without SameSite Attribute and Cookie No HttpOnly Flag show that cookie hardening still needs improvement. SameSite controls whether a browser sends cookies in cross-site requests, while HttpOnly prevents JavaScript from reading a cookie. These controls protect different aspects of session security. SameSite helps reduce CSRF exposure, whereas HttpOnly reduces session theft if an XSS weakness exists. For sensitive cookies, the configuration should include HttpOnly, Secure, an appropriate SameSite value, a narrow path, and no unnecessary domain scope.

Cross-Domain JavaScript Source File Inclusion means that the website loaded scripts from external domains. This practice is common, but it creates dependency risk because JavaScript from another domain executes with the privileges of the page that imports it. If an external source is compromised, the website can be affected without a direct change to its own server. The safest approach is to keep an inventory of external scripts, remove unnecessary dependencies, load scripts from trusted sources, and combine this control with CSP.

Server Leaks Version Information through the Server header is a reconnaissance issue. Product and version information can help attackers select exploits or narrow their research toward a specific platform. Hiding version information does not replace patching, but it reduces unnecessary exposure. Server banners should be minimized, detailed error pages should be disabled for public users, and the server and modules must be updated regularly.

Strict-Transport-Security Header Not Set and Secure Pages Include Mixed Content indicate that transport security needs strengthening. HSTS instructs browsers to use HTTPS for future connections, reducing downgrade risk after the policy is received. Mixed content occurs when an HTTPS page loads HTTP resources, which weakens the protection provided by TLS. The administrator should ensure that every page and asset is served through HTTPS and then enable HSTS with a suitable max-age.

X-Content-Type-Options Header Missing indicates that the application did not send the nosniff directive. Some browsers may attempt to infer the content type of a response. When an incorrect or ambiguous content type is interpreted as executable script, the risk of content execution can increase. Setting X-Content-Type-Options: nosniff and sending accurate Content-Type headers are simple but important hardening measures.

3.5. Analysis of informational findings

Informational findings are not direct proof of vulnerabilities, but they identify areas that require contextual review. Authentication Request Identified shows that ZAP recognized an authentication-related request. This alert should be followed by manual testing of login behavior, password handling, generic error messages, rate limiting, lockout policy, HTTPS enforcement, and account recovery mechanisms. Automated scanning alone cannot determine whether all authentication controls are strong.

Information Disclosure - Sensitive Information in URL indicates that sensitive values may have been placed in a URL. URLs are often stored in browser history, server logs, proxy logs, bookmarks, and analytics systems. For this reason, credentials, session identifiers, tokens, or personal information should not be passed through query strings. Sensitive data should be processed through secure forms, request bodies, or server-side sessions.

Information Disclosure - Suspicious Comments means that the web response contained comments that may provide useful information to attackers. Comments may reveal unfinished code, file names, debugging notes, old paths, developer assumptions, or references to internal components. The solution is to remove all unnecessary public comments and keep operational notes inside internal documentation rather than in delivered HTML.

Modern Web Application indicates that the site may contain dynamic behavior that is not fully explored by a standard spider. In future testing, Ajax Spider or authenticated crawling can be used to reach pages and functions that depend on JavaScript interaction. This is important because automated scanners can miss content that is not linked statically or that requires a user role. Re-examine Cache-control Directives indicates that cache headers are missing or not configured correctly. Sensitive pages should instruct browsers and proxies not to store private content. At the same time, public static assets may use controlled caching to improve performance. The important point is that caching behavior must match the sensitivity of the content.

Session Management Response Identified shows that ZAP recognized a token related to session management. Session handling should be manually evaluated to ensure that identifiers are unpredictable, regenerated after login, regenerated after privilege changes, expired after inactivity, and invalidated during logout. Session fixation and reuse after logout should also be tested. User Controllable HTML Element Attribute (Potential XSS) identifies a location where user input may appear inside an HTML attribute. This is not automatically a confirmed XSS vulnerability, but it requires review. The application should validate input and encode output according to the HTML context before writing user-controlled values into attributes. This control helps prevent script execution through malformed attribute values.

3.6. Security impact and remediation priority

The findings can affect security objectives in different ways. Anti-CSRF and SameSite issues are related to the integrity of user actions. CSP, vulnerable JavaScript libraries, and user-controllable HTML attributes are related to browser-side execution and may affect confidentiality or integrity if exploited. Cookie issues and session-management indicators are related to session confidentiality and authenticity. HSTS and mixed content are related to transport confidentiality and integrity. Server banners, suspicious comments, and sensitive information in URLs primarily assist reconnaissance and information disclosure.

A practical remediation plan should begin with the medium-risk findings because they represent missing preventive controls and known vulnerable components. The next group should address cookie and transport configuration because these controls protect session data and secure communication. Information disclosure and cache configuration should then be reviewed to reduce unnecessary exposure. Finally, the website should be retested after each remediation batch to verify whether the alerts have been reduced or eliminated.

Table 4: Remediation priority and verification criteria

Priority	Control area	Actions	Responsible role	Verification evidence
1	Medium-risk controls	Add Anti-CSRF tokens, deploy CSP, apply anti-clickjacking headers, and update vulnerable JavaScript libraries.	Developers and web administrators	Repeat scan and verify response headers, token validation, and library versions.
2	Cookie and session security	Set HttpOnly and SameSite attributes, review session regeneration, expiration, and logout invalidation.	Application developers	Inspect cookies in the browser and test session behavior manually.
3	HTTPS and transport security	Remove mixed content and enable HSTS after confirming full HTTPS support.	Infrastructure administrators	All pages and assets load through HTTPS and HSTS header is present.
4	Information disclosure control	Suppress server version banners, remove suspicious comments, and avoid sensitive data in URLs.	Web administrators and developers	Public responses no longer reveal unnecessary technical information.
5	Cache and dynamic testing	Set proper cache-control directives and use Ajax Spider or authenticated testing for dynamic pages.	Security tester and application owner	Cache behavior matches data sensitivity and future scans reach dynamic pages.
6	Continuous verification	Repeat OWASP ZAP scans after fixes and major updates; document closed findings and accepted residual risks.	Security tester and system owner	A comparison report shows remediation status and remaining risks.

3.7. Comparison with previous studies

Table 5: Comparison with related security assessment studies

Study	Object and method	Main result	Relation to this study
Rosaliah et al. [33]	Website security testing with penetration testing and OWASP Top 10.	Authentication, sensitive data exposure, misconfiguration, and clickjacking issues were found.	The present study also found browser protection and configuration weaknesses, especially Anti-CSRF, CSP, and clickjacking controls.
Ghozali et al. [34]	OWASP-based web application risk rating.	Risk levels were derived from likelihood and impact.	The present study uses OWASP ZAP severity and adds remediation verification criteria.
Sofyan et al. [35]	Institutional website penetration testing.	One high, three medium, and six low vulnerabilities were reported.	The present study recorded no high finding in the detailed table, but medium and low hardening issues remain important.
Fachri et al. [36]	Academic web server penetration test.	High and medium server weaknesses and open service exposure were identified.	The present study focuses on web application scanning and browser-facing configuration rather than network exploitation.
Muni et al. [2]	Academic information system assessed using OWASP ZAP.	Multiple vulnerabilities and recommendations were identified.	The present study is aligned with the same institutional security concern and provides detailed interpretation for Sistemasi UNISI.

3.8. Research limitations

This research has several limitations. First, the assessment depended on OWASP ZAP. Although ZAP is useful for web vulnerability detection, it does not cover all possible weaknesses. Different tools such as Burp Suite, Nmap, Nessus, source-code analyzers, or dependency scanners may produce additional findings. Therefore, the result should be interpreted as an OWASP ZAP-based baseline rather than a complete security audit.

Second, the research was limited to non-destructive testing. It did not include denial-of-service simulation, credential attacks, exploitation that changes data, or social engineering. This limitation is appropriate for an academic and ethical assessment, but it also means that some risks were not tested. Third, access context affects the scan result. Pages that require administrator roles, lecturer roles, or authenticated workflows may not be fully evaluated if the scanner does not operate with those credentials.

Fourth, the result is time-dependent. Website security can change when software is updated, features are added, server configuration is modified, or dependencies are replaced. The findings therefore represent the condition observed during the research period. Future work should include authenticated testing, manual verification of business logic and access control, comparison with additional tools, and retesting after remediation.

4. Conclusion

The penetration testing activity using OWASP ZAP identified 19 detailed security findings on the Sistemasi UNISI website. The findings consisted of four medium-risk findings, eight low-risk findings, and seven informational findings. No high-risk finding was presented in the detailed report. The medium-risk findings were Absence of Anti-CSRF Tokens, Content Security Policy Header Not Set, Missing Anti-clickjacking Header, and Vulnerable JS Library. These findings show that the website requires improvement in preventive browser controls, request validation, and client-side dependency management.

The low-risk findings show that several hardening controls were incomplete, including cookie attributes, cross-domain JavaScript management, server banner reduction, HSTS, MIME sniffing protection, and mixed-content prevention. The informational findings show areas that require additional review, including authentication behavior, sensitive information in URLs, suspicious comments, dynamic application behavior, cache-control directives, session management, and potential XSS contexts. Overall, the website can still be accessed normally, but its security configuration needs strengthening to reduce exploitation opportunities.

The recommended improvements include adding Anti-CSRF tokens, implementing Content Security Policy, applying X-Frame-Options or CSP frame-ancestors, updating vulnerable JavaScript libraries, configuring HttpOnly and SameSite cookies, enforcing HTTPS and HSTS, removing mixed content, suppressing unnecessary server information, removing suspicious comments, improving cache-control directives, and performing retesting after remediation. Future research should combine OWASP ZAP with manual testing and additional tools to obtain a broader evaluation of the website security posture.

Acknowledgement

The authors would like to thank the Department of Information Systems, Faculty of Engineering and Computer Science, Universitas Islam Indragiri, for academic support during the completion of this research. The authors also acknowledge the guidance provided during the preparation of the thesis and the journal manuscript.

References

- [1] B. Rianto and I. Alsa, "Perancangan Aplikasi Promosi Universitas Islam Indragiri Berbasis Multimedia Interaktif," *Selodang Mayang: Jurnal Ilmiah Badan Perencanaan Pembangunan Daerah Kabupaten Indragiri Hilir*, vol. 7, no. 1, pp. 39-43, 2021.
- [2] A. Muni, B. Rianto, and M. Jalil, "Analisis Kerentanan Keamanan Sistem Informasi Akademik Menggunakan Owasp-Zap Di Universitas Islam Indragiri," *Teknofile: Jurnal Sistem Informasi*, vol. 2, no. 6, pp. 409-420, 2024.
- [3] A. Prahendratno et al., *Strategi Bisnis Digital: Optimalisasi & Otomatisasi Sebuah Bisnis Menggunakan Media Digital*. PT. Sonpedia Publishing Indonesia, 2023.
- [4] A. D. Madya, B. D. Haryanto, D. P. Ningsih, and F. Sinlae, "Keefektifan Metode Proteksi Data dalam Mengatasi Ancaman Cybersecurity," *Indonesian Journal of Education And Computer Science*, vol. 1, no. 3, pp. 127-135, 2023.
- [5] G. Alfaren, "Analisis Serangan Penetration Testing: Sebuah Review Sistematis," *Jiifkom (Jurnal Ilmiah Informatika dan Komputer)*, vol. 1, no. 2, pp. 21-26, 2022.
- [6] M. Huda, *Keamanan Informasi*. Nulisbuku, 2020.
- [7] M. R. Ardiansyah et al., "Analisis Kerentanan Keamanan Website Menggunakan Metode Ptes (Penetration Testing Execution And Standart)," *Nuansa Informatika*, vol. 18, no. 2, pp. 145-153, 2024.
- [8] M. Tahir and M. R. Ardiansyah, "Analisis Keamanan Website Dinas Pemerintahan Yogyakarta Dengan Metode Ptes (Penetration Testing Execution Standard)," *Jurnal Teknik Informatika Unika Santo Thomas*, pp. 118-125, 2024.
- [9] F. Tinambunan, A. Junaidi, and A. M. Rizki, "Pengujian Sistem Informasi Akademik Universitas X Melalui Pendekatan Penetration Testing Berdasarkan Owasp Top 10," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 1, pp. 1062-1069, 2024.
- [10] A. F. Hasibuan and D. Handoko, "Analisis Kerentanan Website Dengan Aplikasi Owasp Zap," *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 2, no. 2, pp. 257-270, 2023.
- [11] K. Nisa, M. A. Putra, R. A. Siregar, and M. D. Irawan, "Analisis Website Tapanuli Tengah Menggunakan Metode Open Web Application Security Project Zap (Owasp Zap)," *Bulletin of Information Technology (BIT)*, vol. 3, no. 4, pp. 216-308, 2022.
- [12] J. N. Ginting, "Perancangan dan Pembuatan Sistem Informasi Penerimaan Mahasiswa Baru Berbasis Website," *Jurnal Nasional Teknologi Komputer*, vol. 2, no. 2, pp. 51-59, 2022.
- [13] W. Wiyanto, S. Fadhillah, and A. Siswandi, "E-Tourism Sebagai Media Wisata Kabupaten Bekasi Berbasis Website," *Journal of Practical Computer*

- Science, vol. 2, no. 1, pp. 1-14, 2022, doi: 10.37366/jpcs.v2i1.1035.
- [14] B. Rianto, "WESITE PENYEDIAN INFORMASI RUMAH KOS KAB. INHIL Studi Kasus: Sebagian Rumah Kos Yang Ada Di Tembilahan," JUTI UNISI, vol. 3, no. 1, pp. 35-42, 2019.
 - [15] B. Rianto, L. Lidya, and G. W. Nurcahyo, "Pemodelan Arsitektur Enterprise Menggunakan Metode Togaf ADM Studi Kasus Dinas Kesehatan Kabupaten Indragiri Hilir," Jurnal Komputer Terapan, vol. 2, no. 1, pp. 55-68, 2016.
 - [16] S. Hidayatulloh and D. Saptadjadi, "Penetration Testing pada Website Universitas ARS Menggunakan Open Web Application Security Project (OWASP)," Jurnal Algoritma, vol. 18, no. 1, pp. 77-86, 2021.
 - [17] D. F. Priambodo, A. D. Rifansyah, and M. Hasbi, "Penetration Testing Web XYZ Berdasarkan OWASP Risk Rating," Teknika, vol. 12, no. 1, pp. 33-46, 2023.
 - [18] H. H. Muhammad, A. I. Hadiana, and H. Ashaury, "Pengamanan Aplikasi Web Dari Serangan Sql Injection Dan Cross Site Scripting Menggunakan Web Application Firewall," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 7, no. 5, pp. 3265-3273, 2023.
 - [19] D. S. Wibowo, M. Nishom, and T. Abidin, "Pengumpulan Informasi pada Situs Web Dengan Menyusun Kerangka Kerja Keamanan Siber NIST," Jurnal Informatika: Jurnal Pengembangan IT, vol. 9, no. 1, pp. 1-6, 2024.
 - [20] M. Idris, I. Syarif, and I. Winarno, "Web Application Security Education Platform Based On OWASP API Security Project," EMITTER international journal of engineering technology, pp. 246-261, 2022.
 - [21] G. Guntoro, L. Costaner, and M. Musfawati, "Analisis Keamanan Web Server Open Journal System (Ojs) Menggunakan Metode Issaf Dan Owasp (Studi Kasus Ojs Universitas Lancang Kuning)," JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika), vol. 5, no. 1, pp. 45-55, 2020.
 - [22] Y. Mulyanto, E. Haryanti, and J. Jumirah, "Analisis Keamanan Website Sman 1 Sumbawa Menggunakan Metode Vulnerability Asesement," Jurnal Informatika Teknologi dan Sains (Jinteks), vol. 3, no. 3, pp. 394-400, 2021.
 - [23] P. Jarupunphol, S. Seatun, and W. Buathong, "Measuring Vulnerability Assessment Tools' Performance on the University Web Application," Pertanika J. Sci. Technol., vol. 31, no. 6, 2023.
 - [24] Y. Yudianta, A. Elanda, and R. L. Buana, "Analisis Kualitas Keamanan Sistem Informasi E-Office Berbasis Website Pada STMIK Rosma Dengan Menggunakan OWASP Top 10," CESS (Journal of Computer Engineering, System and Science), vol. 6, no. 2, pp. 185-191, 2021.
 - [25] N. Albalawi, N. Alamrani, R. Aloufi, M. Albalawi, A. Aljaedi, and A. R. Alharbi, "The Reality of Internet Infrastructure and Services Defacement: A Second Look at Characterizing Web-Based Vulnerabilities," Electronics, vol. 12, no. 12, p. 2664, 2023.
 - [26] H. Poston, "Mapping The OWASP Top Ten to Blockchain," Procedia Comput. Sci., vol. 177, pp. 613-617, 2020.
 - [27] K. Qian, R. M. Parizi, and D. Lo, "Owasp Risk Analysis Driven Security Requirements Specification For Secure Android Mobile Software Development," in 2018 IEEE Conference on Dependable and Secure Computing (DSC), IEEE, 2018, pp. 1-2.
 - [28] S. Khandelwal, "Critical RCE Flaw Discovered In Blockchain-Based EOS Smart Contract System," The Hacker News, 2018.
 - [29] I. F. Ashari, V. Oktarina, R. G. Sadewo, and S. Damanhuri, "Analysis of Cross Site Request Forgery (CSRF) Attacks on West Lampung Regency Websites Using OWASP ZAP Tools," Jurnal Sisfokom (Sistem Informasi dan Komputer), vol. 11, no. 2, pp. 276-281, 2022.
 - [30] I. Riadi, A. Fadlil, and M. A. Mu'min, "OWASP Framework-based Network Forensics to Analyze the SQLi Attacks on Web Servers," MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer, vol. 22, no. 3, pp. 481-494, 2023.
 - [31] G. H. Editya and S. Mulyati, "Aplikasi Mobile One Time Password Menggunakan Algoritma MD5 dan SHA1 untuk Meningkatkan Keamanan Website," SKANIKA: Sistem Komputer dan Teknik Informatika, vol. 1, no. 2, pp. 618-623, 2018.
 - [32] J. T. Santoso, "Teknologi Keamanan Siber (Cyber Security)," Penerbit Yayasan Prima Agus Teknik, pp. 1-173, 2023.
 - [33] Y. T. A. Rosaliah, J. Jayanta, and B. Hananto, "Pengujian Celah Keamanan Website Menggunakan Teknik Penetration Testing dan Metode OWASP TOP 10 pada Website SIM xxx," in Prosiding Seminar Nasional Mahasiswa Bidang Ilmu Komputer dan Aplikasinya, 2021, pp. 752-761.
 - [34] B. Ghazali, K. Kusriani, and S. Sudarmawan, "Mendeteksi Kerentanan Keamanan Aplikasi Website Menggunakan Metode Owasp (Open Web Application Security Project) Untuk Penilaian Risk Rating," Creative Information Technology Journal, vol. 4, no. 4, pp. 264-275, 2019.
 - [35] H. Sofyan, M. Sugiarto, and B. M. Akbar, "Implementation of Penetration testing on Websites to Improve Security of Information Assets UPN Veteran Yogyakarta," Telematika: Jurnal Informatika dan Teknologi Informasi, vol. 20, no. 2, pp. 153-162, 2023.
 - [36] F. Fachri, A. Fadlil, and I. Riadi, "Analisis Keamanan Webserver menggunakan Penetration Test," J. Inform, vol. 8, no. 2, pp. 183-190, 2021.
 - [37] H. S. Abdullah, "Evaluation of opensource web application vulnerability scanners," Academic Journal of Nawroz University, vol. 9, no. 1, pp. 47-52, 2020.