Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 9, No. 2, June 2023, pp. 250-267
ISSN: 2338-3070, DOI: 10.26555/jiteki.v9i2.25987

Implementation of Open Web Application Security Project for Penetration Testing on Educational Institution Websites

Nani Sulisnawati, Subektiningsih
Informatics Departement, Faculty of Computer Science, Universitas Amikom Yogyakarta, Yogyakarta 55283, Indonesia

ARTICLE INFO

Article history:

Received March 24, 2023
Revised April 18, 2023
Published April 25, 2023

Keywords:

OWASP;
Cybersecurity;
Penetration Testing;
Website

ABSTRACT

The development of information technology cannot be separated from the development of website applications, as well as the threat of security attacks that will attack website applications. Educational Institution X uses a website application as an important medium in learning activities. Therefore, penetration testing is needed to find security holes in website applications. In this study, penetration testing will be carried out with the target website for student access at Educational Institution X based on the reason that there is sensitive student data that needs to be secure. The method used in this study is an experimental method with the OWASP TOP 10 2021 standard (Open Web Application Security Project). The penetration test results obtained on the website application at Educational Institution X found 11 vulnerabilities that could be tested. Of the 11 vulnerabilities, there is one vulnerability at the medium risk level, 7 at the low risk level, and 3 at the information risk level. The vulnerabilities found relate to token authentication, policy delivery, cookie attribute, cross-site script inclusion, authorization, clickjacking, and weak transport layer security. Based on the penetration testing activities obtained, it can be concluded that the vulnerability gaps found need to be further repaired by the website application system developer, in this case, the Educational Institution X. Therefore, the final result of this study is in the form of a report document containing a list of vulnerabilities, recommendations for vulnerability repairs, and vulnerability mitigation strategies as solutions for handling security systems on website applications to make them even better.

Corresponding Author:

Subektiningsih, Universitas Amikom Yogyakarta, Yogyakarta, 55283, Indonesia
Email: subektiningsih@amikom.ac.id

1. INTRODUCTION

The development of information technology (IT) has become part of the aspects of human daily life. Based on [1], his research stated that IT, which is developing rapidly, has a very important role, with internet usage reaching 53.7% of the Indonesian population. In addition, IT has fueled the development of Industry 4.0, also known as the Industrial Internet of Things, with increased automation and digitization to provide the ability to share and develop sensing, communication, and computing capabilities and increase accurate data generation [2][3][4][5]. This is realized by developing various physical and digital technologies such as the Internet of Things, advanced robotics, artificial intelligence, machine learning, big data analytics, cloud computing, smart sensors, augmented reality, and additive manufacturing [6][7]. However, IT integration also raises new problems and challenges in cybercrime [8][9][10][11][12], specifically the type of cybersecurity.

Cybersecurity is one of the most frequently discussed topics in website applications which is very important to protect system confidentiality and integrity of data, networks, and programs from cyber attacks [13][14][15]. One part of cybersecurity threats related to information security is website application vulnerabilities [16]. Website applications are client-server computer programs that run on web browser applications on the Internet network [17]. According to Badan Siber dan Sandi Negara (BSSN), 1,637,973,022

cyber attacks occurred from 1 January to 31 December 2021. Of the 1,637,973,022 cyber attacks, around 5,940 were website application attacks. However, they did not account for 1% of the total attacks, the Directorate of Operations Cyber Security stated that the results of the analysis of security holes which were the most targeted attacks were the Academic sector with 2,217 attacks [18]. Violating application security for Academics is fatal because it will reduce public trust in Academics, in this case, Educational Institutions. The website application for Educational Institutions functions as a media center and campus information source. The level of website application security must be maintained because it is a widely accessible website application field [19]. Educational Institution X is an Educational Institution in the field of computers that makes website applications an important medium, one of which is a student dashboard that stores sensitive and confidential data or information. Based on [20][21][22], the most common security investigation effort is through penetration testing, carried out by the tester team legally by acting as black hat hackers to find potential exploits and vulnerabilities. Penetration testing is the process of identifying and exploiting system domain vulnerabilities to reduce the risk of attacks on web applications in the cybersecurity field [21][23][24][25][26][27].

A website application-based measurement standard is needed in carrying out the penetration testing process [13]. At the time this research was made, the standard used was OWASP TOP 10 2021 (Open Web Application Security Project), which describes ten security risks [28], namely injection, broken authentication, sensitive data exposure, XML external entities, broken access control, security misconfiguration, cross-site scripting, insecure deserialization, using components with known vulnerabilities, and insufficient logging & monitoring [29]. The OWASP is a nonprofit, which focuses on studying application software security, and provides information on the most recent and popular vulnerabilities related to web applications [30][31][32][33]. In addition, the OWASP TOP 10 is also a guide for website application developers and security teams to manage and predict the weak points of website applications that are vulnerable to cyber attacks [34]. Research [13], utilizes the web-based PHP programming language to create a monitoring dashboard for pentest monitoring activities. This system can highlight application vulnerabilities according to how frequently they occur depending on the test findings. Research [35], describes a vulnerability analysis and evaluation performed using Nikto tools, OWASP ZAP (Open Website Application Security Project), Netcraft, Sparta, and Network Mapper (Nmap). The Kali Linux operating system and search engine were used for testing. The Nikto and OWASP ZAP tools were used to test ten domains for security weaknesses. Based on the test results, it can be seen that when the Nikto and OWASP ZAP tools are juxtaposed, it turns out that the Nikto tool has more vulnerabilities than OWASP ZAP.

Research [36], aims to test the performance of the web server and the results of testing ModSecurity with CRS v.3.2 at the default level, how well it can defend the web server from DoS attacks, and how well the web server performs in terms of throughput (the average number of bytes transmitted per second), transaction rate (hit count), concurrency, and throughput (average number of parallel connections and increases as server efficiency decreases) against the top 10 hazards identified by OWASP. Research [37], aims to determine the security level of a website, whether more protection is needed, and what recommendations should be made for the website, this research evaluates and tests the security of websites with six sub-domains. OWASP TOP 10 is used to perform analysis as penetration testing. Based on the tests carried out, the results obtained are that website security is generally safe. However, there are several subdomains with a security value of 60% which is a level that is vulnerable to manipulation.

Research [38], suggests techniques that seek to automate the work Red Hat hackers usually do to detect and fix bugs. This research examines the suggested method using the Metaesploitable Linux distribution and exploiting the toolchains of various well-known utilities. Based on the tests carried out, the results prove that this method can automatically reduce vulnerabilities that affect widely used services. Research [39], in helping web server administrators implement web server security as needed, a study was conducted using the OWASP scanner approach on web servers. Based on the tests, the results are Cross-Site Scripting attacks with high risk for two website URLs. At the same time, the other vulnerabilities are only notifications and have a low to moderate vulnerability status. Research [40], presents efforts to compile and combine information using the VAPT (Vulnerability Assessment Penetration Testing) method in the IoT field using Nmap, Acunetix, Burpsuite, and Metasploit tools to run tests, analysis reports, and combined results that confirm vulnerability detection. Research [41], presents a cybersecurity investigation of common household devices connected to an IoT network using ExploitDB12, Packet Storm Security13, and the automated tool Metaspoit. Based on the tests, the results of the 22 devices tested showed that 17 vulnerabilities were found and published as new CVEs.

Research [42], focuses on cross-site scripting (XSS), SQL injection (SQLi), and cross-site query forging (CSRF) vulnerabilities. Exploits are done manually and automatically using Sqlmap and Havij. The final results of the comparison of the Sqlmap and Havij tools allow us to draw a certain counter-strategy that should be used while developing web applications to mitigate this type of attack. Research [43], Implementing an

Artificial Intelligence (AI) approach to automatically generate penetration tests for VANET models allowed researchers to examine potential advancements in penetration testing development. Research [44], explains how to conduct VAPT sessions using IoT devices with the use of a brand-new Cyber Kill Chain (KC) dubbed PETIoT. Research [45], the introduction of IoT-PEN, which uses target graphs to discover how a target system could be compromised by an attacker. This research also shows that an attacker can leverage a series of exploits of various vulnerabilities on various hosts to access a target system even when the system is secure under various threat models. Research [46], by implementing a web application firewall tries to defend and reduce malicious attacks that target website applications. The method used in this research is OWASP (Open Web Application Security Project) as the standard step for performing penetration testing. Based on the results of the three trials that have been tested, the web application firewall is 66% effective in protecting vulnerable website applications and can protect against two high-risk vulnerabilities.

This research discusses the effectiveness of penetration testing in protecting a system. In this case, it focuses on how effective penetration testing is in protecting student dashboard website application systems at Educational Institution X as a target in finding security holes with the domain to be tested, namely xxx.xxxxxx.ac.id. The method used in this study is an experimental method with the OWASP TOP 10 2021 standard (Open Web Application Security Project). From this research, the final results will be obtained in the form of an analysis report document containing a list of vulnerabilities, recommendations for repairing vulnerabilities, and vulnerabilities mitigation strategies as materials for improving the website application system at Educational Institution X.

2. METHODS

In supporting this research, the important components used consist of hardware and software. The hardware used is a laptop, while the software used is the Kali Linux operating system and penetration testing tools. The following specifications of the software used in this study are described in Table 1.

Table 1. Software Spesifications

Software Specifications	
Virtual Machines	VirtualBox
Operating System	Kali Linux
Scan Tool	Nmap
Scanning and Vulnerability Assessment	OWASP ZAP and Acunetix
Exploit Tool	Burp Suite, Editor Cookies, XXS Hunter

Systematic research is a process used to conduct research. Research systematics can be used to verify whether the desired problem-solving objectives have been achieved. The systematics of this research uses the OWASP TOP 10 2021 standards. The OWASP TOP 10 2021 standard is used as a stage of the penetration testing process with the resulting vulnerability results categorized based on the OWASP TOP 10 2021. This research uses the information gathering stage, the scanning and vulnerability assessment stage, the exploitation stage, and the reporting stage [37][46]. The following is systematic research in conducting penetration testing on the X Educational Institution website application using the OWASP TOP 10 2021 method, which can be seen in Fig. 1.

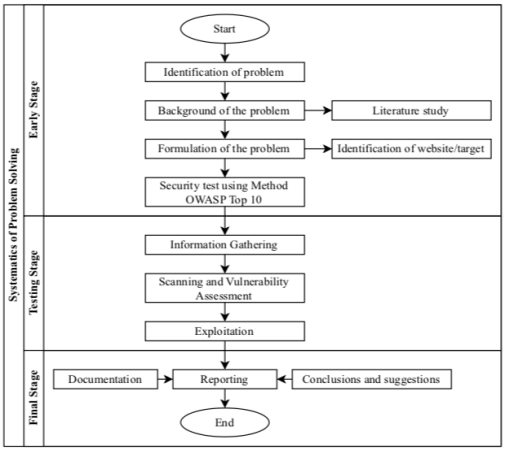

Fig. 1. Systematics of Problem Solving

2.1. Early Stage

In the early stages of conducting this research, define the problems encountered. In this case, the problem faced is finding vulnerabilities in the Internet application of Educational Institution X, which will then be tested based on existing literature studies.

2.2. Testing Stage

At the testing stage, the design of a testing strategy is carried out. At this stage, also prepare tools for research needs such as hardware and software and the tools needed for testing. Then the data and information collection stage will be carried out about the vulnerabilities found in the website application and investigate potential vulnerabilities that allow attackers to damage and change existing data in the website application. After obtaining the vulnerability, an attempt is made to attack the target website. At this stage, an analysis will be carried out on the results of the target website's vulnerability test and the results of its exploitation. The vulnerability analysis in this study was carried out using the OWASP TOP 10 2021 standard.

2.3. Final Stage

At this final stage, all data found in the penetration testing activity is in the form of a report in the form of an accountability document to Educational Institution X which contains a list of vulnerabilities, recommendations for repairing vulnerabilities, and vulnerability mitigation strategies as solutions for handling security systems on website applications to make them even better.

3. RESULTS AND DISCUSSION

3.1. Information Gathering

At this stage, the author as a penetration tester tries to collect data or information on the target website. The tool used for information gathering is Nmap.

3.1.1. Test result using Nmap

Fig. 2 is the result of scanning the target website using Nmap. The scanning results show that port 80/TCP with HTTP service which is a network protocol that connects clients to web servers has an open status and port 443/TCP with HTTPS service which functions as connectivity when needing access HTTPS has an open status. In the next stage, the port is used for exploitation activities with the Burp Suite tool. Then, the Nmap results also get the IP Address address on the target website domain, namely xxx.xxxxxx.ac.id which is used in the next stage for vulnerability scanning activities with the OWASP ZAP tool and Acunetix tool.

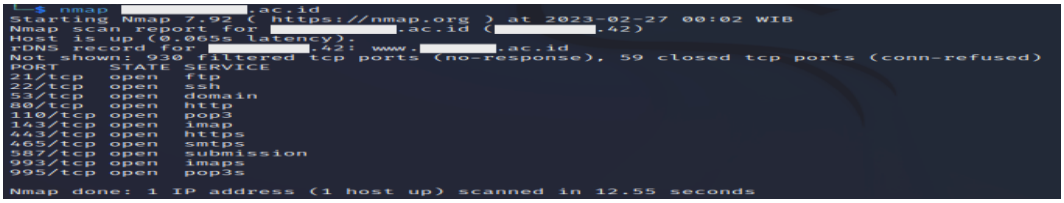

Fig. 2. Test result using Nmap

3.2. Scanning and Vulnerability Assessment

At this stage, the author as a penetration tester tries to gather information about security vulnerabilities found in the website application and investigates potential security holes that allow attackers to damage and change existing data in the website application. This study uses the tools used to perform vulnerability scanning OWASP ZAP and Acunetix tools.

3.2.1. Test result using OWASP ZAP

After a vulnerability scan is performed using the OWASP ZAP tool, a vulnerability gap with High, Medium, Low, and Informational risk levels will be obtained. The vulnerabilities found by the OWASP ZAP tool are described in Table 2, categorized based on the OWASP TOP 10 2021.

Table 2. Vulnerability of OWASP ZAP result

Vulnerabilities	Risk Level
Absence of Anti-CSRF Tokens	Medium
Content Security Policy (CSP) Header Not Set	Medium
Cookies without SameSite Attribute	Low
Cookies Whitout Secure Flag	Low
Cross-Domain JavaScipt Secure File Inclusion	Low
Loosely Scoped Cookies	Informational

3.2.2. Absence of Anti-CSRF Tokens

Without an Anti-CSRF Tokens attack, the victim is made to send HTTP requests to the target location without their knowledge or consent so that the attacker can act in the victim's stead. The main reason is that the app's functionality uses repeatable and predictable URL/form operations. The nature of the attack is that CSRF exploits user trust in a website. Cross-site scripting (XSS), on the other hand, preys on user trust in a website. CSRF attacks are not always cross-site, like XSS, although they can be. CSRF, XSRF, one-click attacks, session riding, confused deputy, and sea surf are other names for cross-site request forgery. This vulnerability is related to token authentication [47]. The impact of missing Anti-CSRF Tokens could be that it is easy for someone to steal another person's account by changing their email address and password or, if the attack is launched from an administrator account, surreptitiously adding a new admin user account. This vulnerability was found at the URL https://xxx.xxxxxx.ac.id.

3.2.3. Content Security Policy (CSP) Harder Not Set

This vulnerability was found in the OWASP ZAP and Acunetix tools. Cross-site scripting (XSS) and data injection attacks are just two examples of the threats that the Content Security Policy (CSP) helps to detect and counter. These attacks are employed for various purposes, from site defacement to virus dissemination to data theft. With the help of the CSP, website owners can designate which types of content—including JavaScript, CSS, HTML frames, fonts, pictures, and embedded objects like Java applets, ActiveX controls, audio files, and video files—browsers should be permitted to load on a given page. This vulnerability is related to policy delivery [48]. The impact is that attacks requiring embedding harmful resources, attacks requiring malicious iframes, such as clickjacking attacks, attacks requiring embedding malicious resources, and others can all be prevented and/or mitigated using CSP. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/assets/, https://xxx.xxxxxx.ac.id/assets/js/, https://xxx.xxxxxx.ac.id/img-sys/, https://xxx.xxxxxx.ac.id/assets/js/guidance/, https://xxx.xxxxxx.ac.id/assets/js/claim/, https://xxx.xxxxxx.ac.id/assets/js/dashboard/, https://xxx.xxxxxx.ac.id/assets/js/email/, https://xxx.xxxxxx.ac.id/assets/js/foto_profil/, https://xxx.xxxxxx.ac.id/assets/js/kuliah/, https://xxx.xxxxxx.ac.id/assets/js/panduan/, https://xxx.xxxxxx.ac.id/assets/js/pempayan/, https://xxx.xxxxxx.ac.id/mailman/archives/, https://xxx.xxxxxx.ac.id/assets/js/pendadaran/, https://xxx.xxxxxx.ac.id/assets/js/perpus/, https://xxx.xxxxxx.ac.id/assets/js/profile/, https://xxx.xxxxxx.ac.id/js/quest_prep/, https://xxx.xxxxxx.ac.id/assets/js/questioner/, https://xxx.xxxxxx.ac.id/assets/js/ujian/, https://xxx.xxxxxx.ac.id/assets/js/validasi_no_ponsel/, https://xxx.xxxxxx.ac.id/assets/js/wisuda/, and https://xxx.xxxxxx.ac.id/assets/js/yudisium/.

3.2.4. Cookies without SameSite Attribute

Since a cookie has not been created with the SameSite property may be delivered in response to a "cross-site" request. Timing attacks, cross-site script inclusion, and cross-site request forgeries can all be prevented using the SameSite property. This vulnerability is related to the cookie attribute [49]. The impact is that a Cross-site Request Forgery (CSRF) attack can result from a cookie without the SameSite Attribute. Declaring whether the cookie should be limited to a first-party or same-site context is possible using the "SameSite" attribute. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/robots.txt, and https://xxx.xxxxxx.ac.id/sitemap.xml.

3.2.5. Cookies Without Secure Flag

This vulnerability was found in the OWASP ZAP and Acunetix tools. The Secure flag is not set on one or more cookies. When the Secure flag is set on a cookie, the browser is told that only secure SSL/TLS channels may be used to access the cookie. This is a crucial session cookie security measure. This vulnerability is related

to the cookie attribute [49]. The impact of this vulnerability is that unencrypted Channels can be used to send cookies. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_kuliah_personal, https://xxx.xxxxxx.ac.id/1perkulian/, https://xxx.xxxxxx.ac.id/1support/master_prodi, https://xxx.xxxxxx.ac.id/1support/, https://xxx.xxxxxx.ac.id/1presensi/kiri_kode_presensi, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_kuliah_prodi, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_ujian_personal, https://xxx.xxxxxx.ac.id/1presensi/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/1perkulian/, https://xxx.xxxxxx.ac.id/1presensi/, https://xxx.xxxxxx.ac.id/1support/, https://xxx.xxxxxx.ac.id/1support/master_prodi, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_kuliah_personal, https://xxx.xxxxxx.ac.id/1perkuliahan/gulung_kuliah_prodi, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_ujian_personal, and https://xxx.xxxxxx.ac.id/1presensi/kiri_kode_presensi.

3.2.6. Cross-Domain JavaScript Secure File Inclusion

One or more script files from a third-party domain are on the page. This vulnerability relates to Cross-Site Script Inclusion (XSSI) [50]. The impact is a security alert called cross-domain JavaScript source file inclusion that may apply to a web application that uses one or more Javascript files originating from a different domain. The third party's harmful material may be added and performed on the victim's web application, intentionally or accidentally. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, and https://xxx.xxxxxx.ac.id.

3.2.7. Loosely Scoped Cookies

Cookies can be restricted by path or domain. Only domain scope is considered in this check. Which domains can access a cookie depends on the domain scope applied to it. A cookie, for instance, can be loosely or strictly scoped to a parent domain, such as nottrusted.com or xxx.nottrusted.com. In the latter scenario, the cookie is accessible through any nottrusted.com subdomain. Mega-applications like Google.com and Live.com frequently use cookies with a broad reach. The browser only transmits set cookies from a subdomain, such as an app.foo.bar, to that domain. Yet, a parent-level domain or any of its subdomains may receive cookies that are specific to that domain. This vulnerability is related to the cookie attribute [49]. The impact is which domains can access a cookie depends on the domain scope applied to it. A cookie, for instance, can be loosely or rigorously scoped to a parent domain, like scanrepeat.com, such as xxx.scanrepeat.com. In the latter scenario, the cookie is accessible from any scanrepeat.com subdomain. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id, https://xxx.xxxxxx.ac.id/robots.txt, https://xxx.xxxxxx.ac.id/robots.txt, https://xxx.xxxxxx.ac.id/sitemap.xml, and https://xxx.xxxxxx.ac.id/sitemap.xml.

3.2.8. Test result using Acunetix

After a vulnerability scan is performed using the Acunetix tool, a vulnerability gap with High, Medium, Low, and Informational risk levels will be obtained. The vulnerability that the Acunetix tool found is described in Table 3 which has been categorized based on the OWASP TOP 10 2021.

Table 3. Vulnerability of Acunetix result

Vulnerabilities	Risk Level
Directory listings	Medium
Clickjacking: X-Frame-Options header	Low
Cookies with missing, inconsistent or contradictory properties	Low
Cookies without Secure flag set	Low
HTTP Strict Transport Security (HSTS) is not implemented	Low
TSL/SSL certificate about to expire	Low
Content Security Policy (CSP) is not implemented	Informational
Session cookies are scoped to parent domain	Informational

3.2.9. Directory listings

Directory listing is when an index file is missing from a given website directory, the directory contents are shown using the web server. Leaving the web server's use of this function unattended puts users' information at risk of a leak. This vulnerability is related to authorization [51]. The impact is a list of all the files from the affected directories that can be viewed by a user, potentially revealing sensitive information. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/assets/, https://xxx.xxxxxx.ac.id/assets/js/, https://xxx.xxxxxx.ac.id/assets/js/guidance/, https://xxx.xxxxxx.ac.id/assets/js/claim/, https://xxx.xxxxxx.ac.id/assets/js/dashboard/, https://xxx.xxxxxx.ac.id/assets/js/email/, https://xxx.xxxxxx.ac.id/assets/js/foto_profil/, https://xxx.xxxxxx.ac.id/assets/js/kuliah/, https://xxx.xxxxxx.ac.id/assets/js/panduan/, https://xxx.xxxxxx.ac.id/assets/js/pempayan/, https://xxx.xxxxxx.ac.id/assets/js/pendadaran/, https://xxx.xxxxxx.ac.id/assets/js/perpus/, https://xxx.xxxxxx.ac.id/assets/js/profile/, https://xxx.xxxxxx.ac.id/assets/js/quest_prep/, https://xxx.xxxxxx.ac.id/assets/js/questioner/, https://xxx.xxxxxx.ac.id/assets/js/ujian/, https://xxx.xxxxxx.ac.id/assets/js/validasi_no_ponsel/, https://xxx.xxxxxx.ac.id/assets/js/wisuda/, and https://xxx.xxxxxx.ac.id/assets/js/yudisium/.

3.2.10. Clickjacking: X-Frame-Options header

Clickjacking (also known as User Interface redress attack, UI redress attack, or UI redressing) is a malicious technique used to trick a Web user into clicking on something other than what they think they are clicking on, potentially disclosing private information or taking control of their computer while they are clicking on what appear to be innocent web pages. This website may be vulnerable to a clickjacking attack since the server did not return an X-Frame-Options header with the value DENY or SAMEORIGIN. Optional X-Frames A browser's ability to render a page inside of a frame or iframe can be controlled using the HTTP response header. By making sure that their material is not integrated into unreliable sites, websites can use this to protect themselves from clickjacking assaults. The affected web application determines the impact. This vulnerability is related to clickjacking [52]. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/img-sys/ and https://xxx.xxxxxx.ac.id/mailman/archives/.

3.2.11. Cookies with missing, inconsistent or contradictory properties

The cookie is invalid or incompatible with another property of the same cookie, with the environment it is being used in, or with at least one of the following properties. Although this is not a vulnerability per se, it will probably cause the program to behave unexpectedly, which could result in other security problems. This vulnerability is related to the cookie attribute [49]. The impact is website browsers won't submit or keep cookies. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_kuliah_personal, https://xxx.xxxxxx.ac.id/1perkulahan/, https://xxx.xxxxxx.ac.id/1support/master_prodi, https://xxx.xxxxxx.ac.id/ 1support/, https://xxx.xxxxxx.ac.id/1presensi/kiri_kode_presensi, https://xxx.xxxxxx.ac.id/1perkuliahan/gulung_kuliah_prodi, https://xxx.xxxxxx.ac.id/1perkulian/schedule_ujian_personal, https://xxx.xxxxxx.ac.id/1presence/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/1perkulahan/, https://xxx.xxxxxx.ac.id/1presence/, https://xxx.xxxxxx.ac.id/1support/, https://xxx.xxxxxx.ac.id/1support/master_prodi, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_kuliah_personal, https://xxx.xxxxxx.ac.id/1perkuliahan/gulung_kuliah_prodi, https://xxx.xxxxxx.ac.id/1perkulian/schedule_ujian_personal, and https://xxx.xxxxxx.ac.id/1presensi/kiri_kode_presensi.

3.2.12. HTTP Strict Transport Security (HSTS) is not implemented

A browser is informed by HTTP Strict Transport Security (HSTS) that a website can only be accessed via HTTPS. The website application's failure to implement HTTP Strict Transport Security (HSTS) was discovered because the response's Strict Transport Security header is missing. This vulnerability is related to transport layer security [53]. The impact is that certain man-in-the-middle (MitM) attacks can be avoided or mitigated with the help of HSTS. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/img-sys/ and https://xxx.xxxxxx.ac.id/mailman/archives/.

3.2.13. TSL/SSL certificate about to expire

The server's TLS/SSL certificate is going to expire soon. Most web browsers will show end users a security warning once the certificate has expired and instruct them to verify the legitimacy of your certificate chain manually. Software or automated processes can decline to connect to the server politely. This warning might have been generated by an intermediate certificate rather than the server (leaf) certificate, but that is not guaranteed. To find the impacted certificate, please refer to the alert details' certificate serial number. This vulnerability is related to weak transport layer security [53]. The impact is that an application server may continue processing data as if nothing happened or the connection may be abruptly terminated if it discovers an expired certificate with a machine it is interacting with. The TLS/SSL certificate (serial: xxxxxxxxxee5b1f020e2343a234d7c1) will expire in less than 60 days. The certificate validity period is from Wed Jan 25, 2023, 07:00:00 GMT+0700 (SE Asia Standard Time) to Wed Apr 26 2023 06:59:59 GMT+0700 (SE Asia Standard Time) (58 days left).

3.2.14. Session cookies are scoped to the parent domain

Instead of a sub-domain, one or more session cookies are scoped to the parent domain. When a cookie is restricted to a parent domain, both the parent domain and any other sub-domains of the parent domain can access it. Security issues might result from this. This vulnerability is related to the cookie attribute [49]. This vulnerability was found in the URL https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_kuliah_personal, https://xxx.xxxxxx.ac.id/1perkulahan/, https://xxx.xxxxxx.ac.id/1support/master_prodi, https://xxx.xxxxxx.ac.id/1support/, https://xxx.xxxxxx.ac.id/1presensi/kiri_kode_presensi, https://xxx.xxxxxx.ac.id/1perkuliahan/gulung_kuliah_prodi, https://xxx.xxxxxx.ac.id/1perkulian/schedule_ujian_personal, https://xxx.xxxxxx.ac.id/1presensi/, https://xxx.xxxxxx.ac.id/, https://xxx.xxxxxx.ac.id/1perkulahan/, https://xxx.xxxxxx.ac.id/1presence/, https://xxx.xxxxxx.ac.id/1support/, https://xxx.xxxxxx.ac.id/1support/master_prodi, https://xxx.xxxxxx.ac.id/1perkuliahan/schedule_kuliah_personal, https://xxx.xxxxxx.ac.id/1perkuliahan/gulung_kuliah_prodi, https://xxx.xxxxxx.ac.id/1perkulian/schedule_ujian_personal, and https://xxx.xxxxxx.ac.id/1presensi/kiri_kode_presensi.

3.3. Exploitation

At this stage, the author as a penetration tester tries to carry out further activities to enter into the computer security system after it is known that there are security holes obtained in the scanning process.

3.3.1. Absence of Anti-CSRF Tokens

Fig. 3 shows an experimental attack using the Burp Suite tool with the Response to this request feature. In the script named from action, enter the URL "https://xxx.xxxxxx.ac.id" to prove that the website application has an Absence of Anti-CSRF Tokens loophole.

Fig. 3. Testing Absence of Anti-CSRF Tokens

Fig. 4 shows the results of an experimental attack that displays the login page without any changes. So the Absence of an Anti-CSRF Tokens gap is not found on this page.

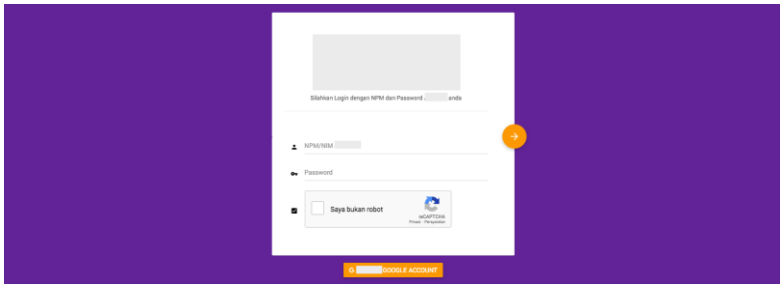

Fig. 4. Test result Absence of Anti-CSRF Tokens

3.3.2. Directory listings

Fig. 5 shows the results of the Acunetix scanning vulnerability that can access urls assets js script that shouldn't be visible. This allows an attacker to study the script and carry out the attack.

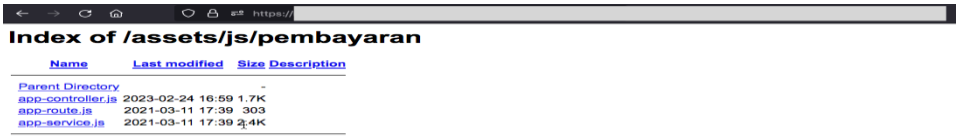

Fig. 5. Test result Directory listings

3.3.3. Cookies without SameSite Attribute

Fig. 6 shows the results of the OWASP ZAP vulnerability scanning with the response results for the status line and header section not having the Set-Cookie SameSite Attribute.

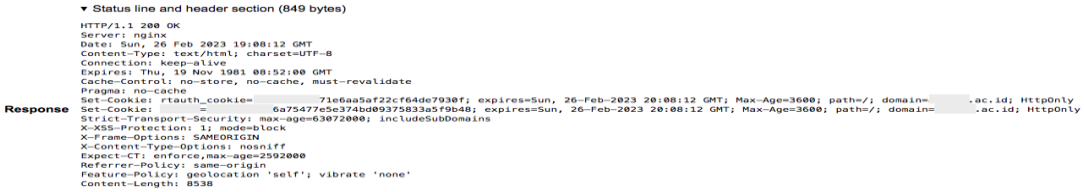

Fig. 6. Testing Cookies without SameSite Attribute

Fig. 7 shows the results of inspecting the website as proof that the Same-Site Attribute is not used.

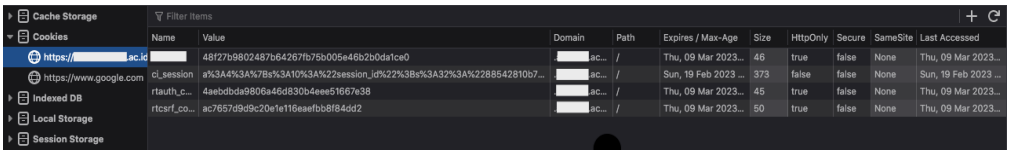

Fig. 7. Testing Cookies without SameSite Attribute

3.3.4. Cookies without Secure flag set

Fig. 8 shows the results of the Acunetix scanning vulnerability with the response results for the Set-Cookie script line not having a Secure Flag.

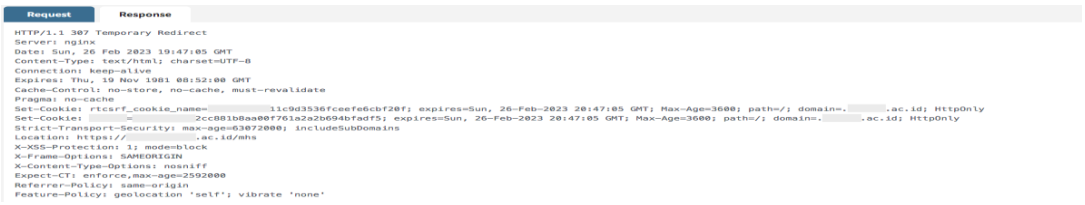

Fig. 8. Test result Cookies with Secure flag set

3.3.5. Cross-Domain JavaScript Secure File Inclusion

Cross-Domain JavaScript Source File Inclusion vulnerability attack using the Cookie Editor, Burp Suite, and XXS Hunter tools. Fig. 9 shows an experimental attack changing the Value of rtkeyaccess with "123456789".

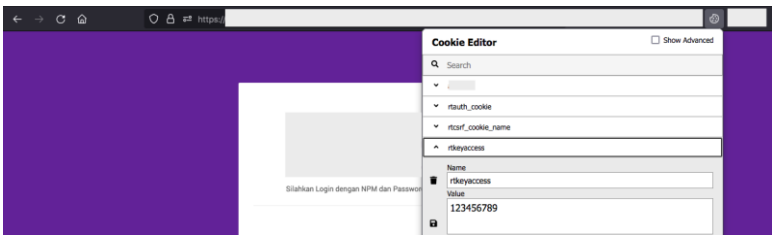

Fig. 9. Testing Cross-Domain JavaScript Source File Inclusion

Fig. 10 shows the experimental attack changing the Javascript Libraries line with the Javascript from the XXS Hunter website.

Fig. 10. Testing Cross-Domain JavaScript Source File Inclusion

Fig. 11 shows the results of the experimental attack on the XXS Hunter website successfully displaying the Value content of rtkeyaccess that has been entered in the Cookie Editor, namely "123456789". So this vulnerability is dangerous because it can steal sensitive information such as cookies, session IDs, personal data, etc.

Fig. 11. Test result Cross-Domain JavaScript Source File Inclusion

3.3.6. Clickjacking: X-Frame-Options header

Fig. 12 and Fig. 13 show an experimental attack using Burp Suite with the Burp Clickbandit feature. On Console, paste Clickbandit then click Run. Then click on the red button with the words Click on the website page with a gray box.

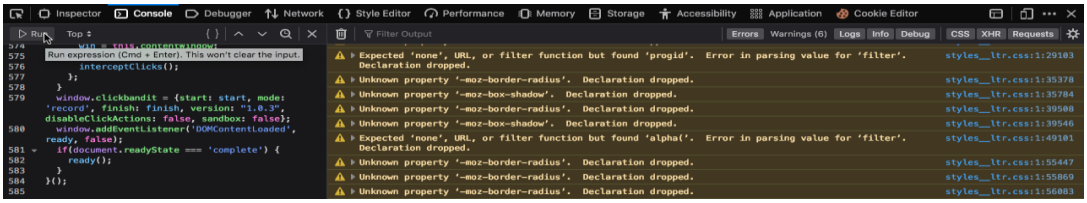

Fig. 12. Testing Clickjacking: X-Frame-Options header

Fig. 13. Testing Clickjacking: X-Frame-Options header

Fig. 14 shows the results of the experimental attack showing the words “You've been clickjacked!”. So Clickjacking: X-Frame-Options header loophole.

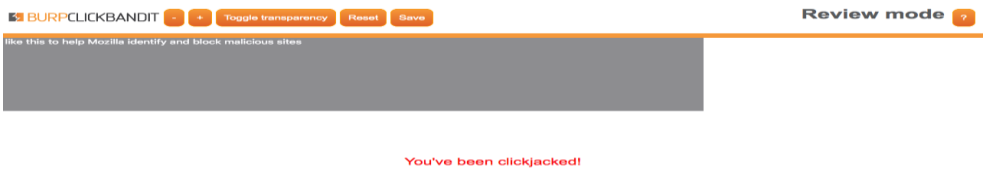

Fig. 14. Test result Clickjacking: X-Frame-Options header

3.3.7. Cookies with missing, inconsistent or contradictory properties

Fig. 15 shows the results of the Acunetix scanning vulnerability with the response results for the Set-Cookie script line not having the Same-Site Attribute such as “Strict”, “Lax”, or “None”.

Fig. 15. Test result Cookies with missing, inconsistent or contradictory properties

3.3.8. HTTP Strict Transport Security (HSTS) is not implemented

Fig. 16 shows the results of Acunetix's vulnerability scanning with the results reporting that two urls do not have HSTS such as Strict-Transport-Security.

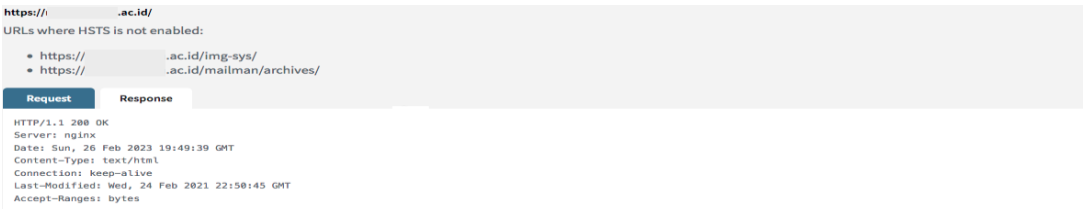

Fig. 16. Test result in HTTP Strict Transport Security (HSTS) not implemented

3.3.9. TSL/SSL certificate about to expire

Fig. 17 shows the results of the Acunetix scanning vulnerability with the report results that the TSL/SSL certificate will expire soon.

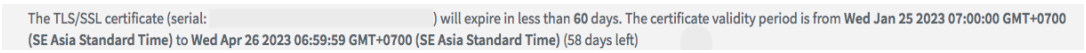

Fig. 17. Test result TSL/SSL certificate about to expire

3.3.10. Loosely Scoped Cookies

Fig. 18 shows the results of the OWASP ZAP vulnerability scanning with the response results for the status line and header section. In the Set-Cookie line, there is a script domain=.xxxxxx.ac.id; which means that the script for the domain is non-specific so not all subdomains follow. The proof is seen in the Session cookies scoped to parent domain vulnerability which displays the same Set-Cookie subdomain.

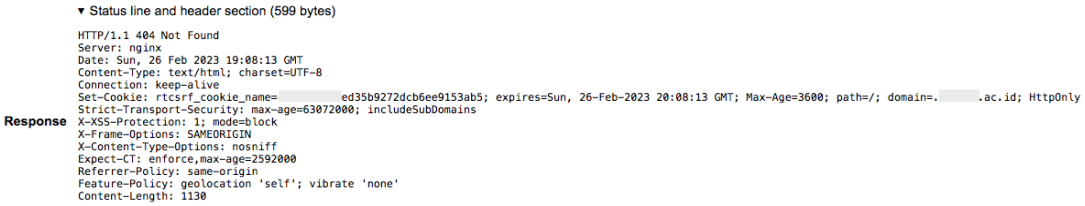

Fig. 18. Test result Loosely Scoped Cookies

3.3.11. Content Security Policy (CSP) is not implemented

Fig. 19 shows the results of the Acunetix scanning vulnerability that can access urls assets js script that shouldn't be visible. This allows an attacker to study the script and carry out the attack.

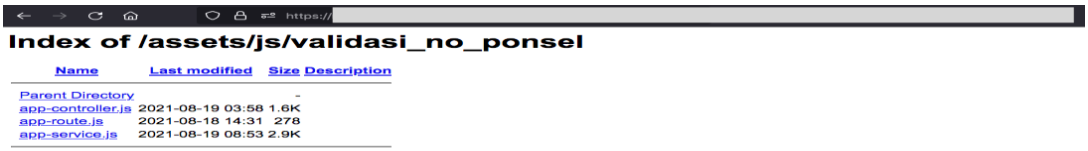

Fig. 19. Test result Content Security Policy (CSP) not implemented

3.3.12. Session cookie are scoped to parent domain

Fig. 20 shows the results of Acunetix's vulnerability scanning with the report showing that one or more session cookies are scoped to the parent domain, not a sub-domain. So that leads to a report showing that one or more Session cookies are scoped to the parent domain, not a sub-domain.

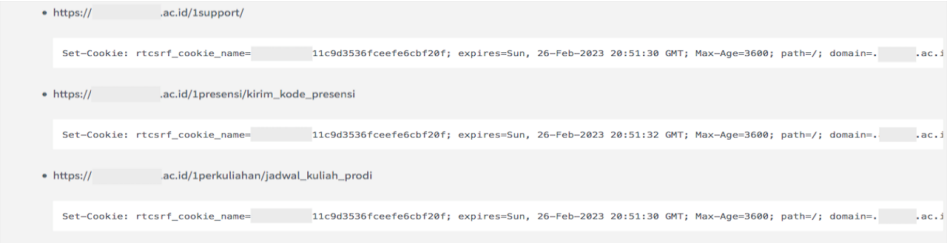

Fig. 20. Test result Session cookies are scoped to the parent domain

3.4. Reporting

The last stage in doing penetration testing is reporting. In this study, the standard vulnerability in penetration testing activities used is the OWASP TOP 10 2021. So the results of the research that been done are described in Table 4.

Table 4. Reporting

OWASP TOP 10 Vulnerability Categories 2021	Vulnerabilities	Test result	Recommendations	Mitigation
A01 Broken Access Control	Absence of Anti-CSRF Tokens	Not successful.	Make sure to use a library or framework that has been tested in preventing CSRF vulnerability, use anti-CSRF software, for example, such as OWASP CSRFGuard, and avoid using cross-site scripting because most of the CSRF defenses can be overcome by scripts controlled by the attacker.	Use built-in or existing CSRF implementations for CSRF protection, synchronizer token pattern, double submit cookie, SameSite cookie attribute, verifying origin with standard headers, use of custom request headers, and user interaction-based CSRF defense.
	Directory listings	Success, assets js script can be accessed.	To prevent the disclosure of sensitive information, users should check the web server settings and consider restricting directory listings.	Ensures that backup files and file metadata are not in the web root and disables web server directory listing.
	Clickjacking: X-Frame-Options header	Success is proven to detect iframes on other website pages.	Set up the CSP header using the frame-ancestors directive and the X-Frame-Options header on a web server. For more details on the potential values for this header, consult online resources.	Set up the CSP header.
	Cookies without SameSite Attribute	It works, the Header is shown not to be using the SameSite Attribute.	Ensure that all cookies have the SameSite Attribute set to "lax" or ideally "strict".	Sets SameSite Attribute for the flags “none”, “lax”, or “strict”.
	Cookies with missing, inconsistent or contradictory properties	It worked and proved not to use SameSite Attribute.	Make that the cookie configuration conforms to the relevant regulations.	Sets SameSite Attribute for the flags “none”, “lax”, or “strict”.

OWASP TOP 10 Vulnerability Categories 2021	Vulnerabilities	Test result	Recommendations	Mitigation
	Session cookies are scoped to parent domain	Success, proven Set-Cookies on subdomains have the same cookie value.	The scope of the session cookies should ideally be limited to a single sub-domain.	Set up domain attributes on a single sub-domain.
A02 Cryptographic Failures	Directory listings	Success, assets js script can be accessed.	To prevent the disclosure of sensitive information, users should check the web server settings and consider restricting directory listings.	Ensures that backup files and file metadata are not in the web root and disables web server directory listing.
A03 Injections	Not found	-	-	
A04 Insecure Design	Clickjacking: X-Frame-Options header	Success is proven to detect iframes on other website pages.	Set up the CSP header using the frame-ancestors directive and the X-Frame-Options header on a web server. For more details on the potential values for this header, consult online resources.	Set up the CSP header.
	Content Security Policy (CSP) is not implemented	Success, assets js script can be accessed.	Implementing Content Security Policy (CSP) into the website application is advised. The Content-Security-Policy HTTP header is added to a web page during configuration to govern the resources the user agent is permitted to load for that page.	Measurable security improvement, disable unsafe APIs, remote attack vectors, and target classes of bugs.
A05 Security Misconfiguration	Directory listings	Success, assets js script can be accessed.	To prevent the disclosure of sensitive information, users should check the web server settings and consider restricting directory listings.	Ensures that backup files and file metadata are not in the web root and disables web server directory listing.
	Cookies with missing, inconsistent or contradictory properties	It worked and proved not to use SameSite Attribute.	Make that the cookie configuration conforms to the relevant regulations.	Sets SameSite Attribute for the flags “none”, “lax”, or “strict”.
	Cookies without Secure flag set	It works, the Header is shown not to be using the Secure Flag.	If at all possible, ought to set these cookies Secure flag.	Sets up the HttpOnly flag on the cookies it creates, indicating that cookies should not be accessed by clients.
	HTTP Strict Transport Security (HSTS) is not implemented	It works, it's proven not to use HSTS.	It is advised that website applications incorporate HTTP Strict Transport Security (HSTS). Web resources might be consulted for more information.	Set up Strict-Transport-Security on the URL obtained.
	Session cookies are scoped to parent domain	Success, proven Set-Cookies on subdomains have the same cookie value.	The scope of the session cookies should ideally be limited to a single sub-domain.	Set up domain attributes on a single sub-domain.
	TSL/SSL certificate about to expire	Success, the TSL/SSL certificate will expire soon.	Contact your Certificate Authority to renew the SSL certificate.	Service configuration validation.

OWASP TOP 10 Vulnerability Categories 2021	Vulnerabilities	Test result	Recommendations	Mitigation
A06 Vulnerable and Outdated Components	Content Security Policy (CSP) is not implemented	Success, assets js script can be accessed.	Implementing Content Security Policy (CSP) into your online application is advised. The Content-Security-Policy HTTP header is added to a web page during configuration to govern the resources the user agent is permitted to load for that page.	Measurable security improvement, disable unsafe APIs, remote attack vectors, and target classes of bugs.
	Directory listings	Success, assets js script can be accessed.	To prevent the disclosure of sensitive information, users should check the web server settings and consider restricting directory listings.	Ensures that backup files and file metadata are not in the web root and disables web server directory listing.
	Cookies with missing, inconsistent or contradictory properties	It worked and proved not to use SameSite Attribute.	Make that the cookie configuration conforms to the relevant regulations.	Sets SameSite Attribute for the flags “none”, “lax”, or “strict”.
	Cookies without Secure flag set	It works, the Header is shown not to be using the Secure Flag.	If at all possible, ought to set these cookies Secure flag.	Sets up the HttpOnly flag on the cookies it creates, indicating that cookies should not be accessed by clients.
	Session cookies are scoped to parent domain	Success, proven Set-Cookies on subdomains have the same cookie value.	The scope of the session cookies should ideally be limited to a single sub-domain.	Set up domain attributes on a single sub-domain.
	TSL/SSL certificate about to expire	Success, the TSL/SSL certificate will expire soon.	Contact your Certificate Authority to renew the SSL certificate.	Service configuration validation.
	Content Security Policy (CSP) is not implemented	Success, assets js script can be accessed.	Implementing Content Security Policy (CSP) into your online application is advised. The Content-Security-Policy HTTP header is added to a web page during configuration to govern the resources the user agent is permitted to load for that page.	Measurable security improvement, disable unsafe APIs, remote attack vectors, and target classes of bugs.
A07 Identification and Authentication Failures	Not found	-	-	
A08 Software and Data Integrity Failures	Cross-Domain JavaScript Secure File Inclusion	Success, the website page is proven to load scripts from third-party domains.	Make sure that JavaScript source files are loaded from only reliable sources and that application end users cannot control the sources.	Sets up the charset attribute of the <script> tag with UTF-16 encoding.
	Loosely Scoped Cookies	It works, it's proven that the script domain is not	Scope cookies to an FQDN at all times (Fully Qualified Domain Name).	Set up attribute domain.

OWASP TOP 10 Vulnerability Categories 2021	Vulnerabilities	Test result	Recommendations	Mitigation
		specific so not all subdomains follow.		
A09 Security Logging and Monitoring Failures	Not found	-	-	
A10 Server-Side Request Forgery	Not found	-	-	

Based on the penetration testing that has been carried out, the results obtained from this study are the discovery of vulnerabilities in website applications with the domain xxx. xxx-xxx.ac.id at Educational Institution X. The number of vulnerabilities that were successfully scanned using the OWASP ZAP tool is 6 vulnerabilities and the Acunetix tool is 8 vulnerability loopholes. However, there are 2 vulnerability gaps with the same category, so the total vulnerability gaps after being combined become 12 vulnerability gaps. Of the 12 vulnerabilities, 2 vulnerabilities are at the medium risk level, 7 are at the low-risk level, and 3 are at the informational risk level. The types of vulnerabilities obtained are Absence of Anti-CSRF Tokens which has the impact that it is easy for someone to steal other people's accounts by changing their email address and password, Content Security Policy (CSP) Harder Not Set has the impact of attacks that require dangerous iframes such as clickjacking attacks, Cookies without SameSite Attribute has the impact of causing a Cross-site Request Forgery (CSRF) attack, Cookies Without Secure Flag has the impact that unencrypted channels can be used to send cookies, Cross-Domain JavaScript Secure File Inclusion has the impact of a security warning when including JavaScript source files cross domains, Loosely Scoped Cookies have an impact on which domains can access cookies depending on the scope of the applied domain, Directory listings have the impact of listing all files of the affected directory visible to the user and potentially exposing sensitive information, Clickjacking: X-Frame-Options header has the impact of affecting the website on the iframe, Cookies with missing, inconsistent or contradictory properties has the impact that the website browser does not send or store cookies, HTTP Strict Transport Security (HSTS) is not implemented has the impact that certain man-in-the-middle attacks can be avoided or mitigated with the help of HSTS, TSL/SSL certificate about to expire has the impact that the connection can be terminated abruptly if it encounters an expired certificate with the machine it interacts with, and Session cookies are scoped to the parent domain has the impact when cookies are restricted to the parent domain, either the parent domain or other sub-domains of the parent domain can access it.

4. CONCLUSION

Based on the entire penetration testing process that has been carried out previously, it can be concluded that this research resulted in penetration testing using the experimental method with the OWASP TOP 10 2021 (Open Web Application Security Project) standard which has proven effective in finding vulnerabilities in website applications with the domain xxx.xxxxxx.ac.id at Educational Institution X. The number of vulnerabilities that were successfully scanned was 12 vulnerabilities. However, only 11 vulnerabilities can be tested. Of the 11 vulnerabilities, there is 1 vulnerability that is at a medium risk level including Directory listings, 7 vulnerabilities at a low-risk level including Cookies without SameSite Attribute, Cookies without Secure flag set, Cross-Domain JavaScript Secure File Inclusion, Clickjacking: X-Frame-Options header, Cookies with missing inconsistent or contradictory properties, HTTP Strict Transport Security (HSTS) is not implemented, TSL/SSL certificate about to expire, and 3 vulnerability holes at the level of informational risk Among them are Loosely Scoped Cookies, Content Security Policy (CSP) is not implemented, and Session cookies are scoped to parent domains. Therefore, monitoring website applications regularly and improving security by using a WAF (Web Application Firewall) is recommended. The limitation of this research is that it allows downtime on website pages, so it is highly recommended to have permission to do penetration testing, in this case, Educational Institution X. The suggestions that can be given for further research include using a different method from the OWASP TOP 10 standard to get a complete guide on vulnerability assessment and penetration testing and using different tools to calculate the resulting vulnerability gaps more accurately.

REFERENCES

[1] H. Ismail, F. Febiyanto, Kevin, and J. V Moniaga, "Methods to prevent privacy violations on the internet on the personal level in Indonesia," *Procedia Comput. Sci.*, vol. 216, 2022, pp. 650–654, 2023, https://doi.org/10.1016/j.procs.2022.12.180.

[2] M. Lezzi, M. Lazoi, and A. Corallo, "Cybersecurity for Industry 4.0 in the current literature: A reference framework," *Comput. Ind.*, vol. 103, pp. 97–110, 2018, https://doi.org/10.1016/j.compind.2018.09.004.

[3] V. Mullet, P. Sondi, and E. Ramat, "A review of cybersecurity guidelines for manufacturing factories in Industry 4.0," *IEEE Access*, vol. 9, pp. 23235–23263, 2021, https://doi.org/10.1109/ACCESS.2021.3056650.

[4] J. Whitter-Jones, "Security review on the Internet of Things," *2018 3rd Int. Conf. Fog Mob. Edge Comput. FMEC 2018*, pp. 163–168, 2018, https://doi.org/10.1109/FMEC.2018.8364059.

[5] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of Industrial Internet of Things security: Requirements and fog computing opportunities," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020, https://doi.org/10.1109/COMST.2020.3011208.

[6] P. Kumar, J. Bhamu, and K. S. Sangwan, "Analysis of barriers to Industry 4.0 adoption in manufacturing organizations: An ISM approach," *Procedia CIRP*, vol. 98, pp. 85–90, 2021, https://doi.org/10.1016/j.procir.2021.01.010.

[7] A. Ustundag and E. Cevikcan, *Industry 4.0: Managing the digital transformation*, Springer Nature, 2022, https://doi.org/10.1007/978-3-319-57870-5.

[8] A. Abdullah, R. Hamad, M. Abdulrahman, H. Moala, and S. Elkhediri, "Cybersecurity: A review of Internet of Things (IoT) security issues, challenges and techniques," *2nd Int. Conf. Comput. Appl. Inf. Secur. ICCAIS 2019*, pp. 1–6, 2019, https://doi.org/10.1109/CAIS.2019.8769560.

[9] A. Hashim, R. Medani, and T. A. Attia, "Defences against web application attacks and detecting phishing links using Machine Learning," *Proc. 2020 Int. Conf. Comput. Control. Electr. Electron. Eng. ICCCEEE 2020*, pp. 1–6, 2021, https://doi.org/10.1109/ICCCEEE49695.2021.9429609.

[10] S. Manoj Kumar, R. Gokula Santhiya, V. Jeni, and J. Joshika Bhavna, "Web Application Security on Top of Public Cloud," *Proc. - 2022 2nd Int. Conf. Interdiscip. Cyber Phys. Syst. ICPS 2022*, pp. 210–215, 2022, https://doi.org/10.1109/ICPS55917.2022.00045.

[11] M. Alawida, A. E. Omolara, O. I. Abiodun, and M. Al-Rajab, "A deeper look into cybersecurity issues in the wake of Covid-19: A survey," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 8176–8206, 2022, https://doi.org/10.1016/j.jksuci.2022.08.003.

[12] M. Azzam, L. Pasquale, G. Provan, and B. Nuseibeh, "Forensic readiness of industrial control systems under stealthy attacks," *Comput. Secur.*, vol. 125, p. 103010, 2023, https://doi.org/10.1016/j.cose.2022.103010.

[13] Y. S. Wijaya and I. Ramadhani, "Web-based dashboard for monitoring Penetration Testing activities based on OWASP Standards," *J. Ilm. Tek. Elektro Komput. dan Inform.*, vol. 6, no. 1, pp. 36–41, 2020, https://doi.org/10.26555/jiteki.v16i1.17019.

[14] O. Ojagbule, H. Wimmer, and R. J. Haddad, "Vulnerability analysis of content management systems to SQL Injection using SQLMAP," *Conf. Proc. - IEEE SOUTHEASTCON*, pp. 1–7, 2018, https://doi.org/10.1109/SECON.2018.8479130.

[15] R. Leszczyna, "Review of cybersecurity assessment methods: Applicability perspective," *Comput. Secur.*, vol. 108, p. 102376, 2021, https://doi.org/10.1016/j.cose.2021.102376.

[16] A. Sharma, A. Singh, N. Sharma, I. Kaushik, and B. Bhushan, "Security countermeasures in web based application," *2019 2nd Int. Conf. Intell. Comput. Instrum. Control Technol. ICICICT 2019*, pp. 1236–1241, 2019, https://doi.org/10.1109/ICICICT46008.2019.8993141.

[17] A. F. Maskur and Y. Dwi Wardhana Asnar, "Static code analysis tools with the taint analysis method for detecting web application vulnerability," *Proc. 2019 Int. Conf. Data Softw. Eng. ICoDSE 2019*, pp. 1–6, 2019, https://doi.org/10.1109/ICoDSE48700.2019.9092614.

[18] Badan Siber dan Sandi Negara, "Laporan tahunan monitoring keamanan Siber 2," *BSSN*, 2022. https://cloud.bssn.go.id/s/Lyw8E4LxwNiJoNw.

[19] I. G. N. Mantra, M. S. Hartawan, H. Saragih, and A. A. Rahman, "Web vulnerability assessment and maturity model analysis on Indonesia higher education," *Procedia Comput. Sci.*, vol. 161, pp. 1165–1172, 2019, https://doi.org/10.1016/j.procs.2019.11.229.

[20] Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. Benefits and drawbacks. Modern tendency," *Mod. Probl. Radio Eng. Telecommun. Comput. Sci. Proc. 13th Int. Conf. TCSET 2016*, vol. 1, pp. 488–491, 2016, https://doi.org/10.1109/TCSET.2016.7452095.

[21] S. Chaudhary, A. O'Brien, and S. Xu, "Automated Post-Breach Penetration Testing through Reinforcement Learning," *2020 IEEE Conf. Commun. Netw. Secur. CNS 2020*, pp. 1–2, 2020, https://doi.org/10.1109/CNS48642.2020.9162301.

[22] Z. Hu, R. Beuran, and Y. Tan, "Automated Penetration Testing using Deep Reinforcement Learning," *Proc. - 5th IEEE Eur. Symp. Secur. Priv. Work. Euro S PW 2020*, pp. 2–10, 2020, https://doi.org/10.1109/EuroSPW51379.2020.00010.

[23] N. Anantharaman and B. Wukkadada, "Identifying the usage of known vulnerabilities components based on OWASP A9," *2020 Int. Conf. Emerg. Smart Comput. Informatics, ESCI 2020*, pp. 88–91, 2020, https://doi.org/10.1109/ESCI48226.2020.9167645.

[24] J. N. Goel and B. M. Mehtre, "Vulnerability assessment & Penetration Testing as a cyber defence technology,"

Procedia Comput. Sci., vol. 57, pp. 710–715, 2015, https://doi.org/10.1016/j.procs.2015.07.458.
[25] M. N. Zakaria, P. A. Phin, N. Mohmad, S. A. Ismail, M. N. Kama, and O. Yusop, "A review of standardization for penetration testing reports and documents," *Int. Conf. Res. Innov. Inf. Syst. ICRIIS*, pp. 1–5, 2019, https://doi.org/10.1109/ICRIIS48246.2019.9073393.
[26] L. F. De Lima, M. C. Horstmann, D. N. Neto, A. R. A. Grégio, F. Silva, and L. M. Peres, "On the challenges of automated testing of web vulnerabilities," *Proc. Work. Enabling Technol. Infrastruct. Collab. Enterp. WETICE*, pp. 203–206, 2020, https://doi.org/10.1109/WETICE49692.2020.00047.
[27] S. Raj and N. K. Walia, "A Study on Metasploit Framework: A Pen-Testing tool," *2020 Int. Conf. Comput. Perform. Eval. ComPE 2020*, pp. 296–302, 2020, https://doi.org/10.1109/ComPE49325.2020.9200028.
[28] H. Sohoel, M. G. Jaatun, and C. Boyd, "OWASP Top 10 - Do Startups care?," *2018 Int. Conf. Cyber Secur. Prot. Digit. Serv. Cyber Secur. 2018*, pp. 1–8, 2018, https://doi.org/10.1109/CyberSecPODS.2018.8560666.
[29] M. Bach-Nutman, "Understanding the Top 10 OWASP vulnerabilities," pp. 1–4, 2020, [Online]. Available: http://arxiv.org/abs/2012.09960.
[30] G. Su, F. Wang, and Q. Li, "Research on SQL Injection vulnerability attack model," *In 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 217–221, 2018, https://doi.org/10.1109/CCIS.2018.8691148.
[31] K. Patel, "A survey on vulnerability assessment penetration testing for secure communication," *Proc. Int. Conf. Trends Electron. Informatics, ICOEI 2019*, pp. 320–325, 2019, https://doi.org/10.1109/ICOEI.2019.8862767.
[32] V. Mounika, X. Yuan, and K. Bandaru, "Analyzing CVE database using unsupervised topic modelling," *Proc. - 6th Annu. Conf. Comput. Sci. Comput. Intell. CSCI 2019*, pp. 72–77, 2019, https://doi.org/10.1109/CSCI49370.2019.00019.
[33] M. Aydos, Ç. Aldan, E. Coşkun, and A. Soydan, "Security testing of web applications: A systematic mapping of the literature," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6775–6792, 2022, https://doi.org/10.1016/j.jksuci.2021.09.018.
[34] Sunardi, I. Riadi, and P. A. Raharja, "Vulnerability analysis of E-voting application using open web application security project (OWASP) framework," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 11, pp. 135–143, 2019, https://doi.org/10.14569/IJACSA.2019.0101118.
[35] R. S. Devi and M. M. Kumar, "Testing for security weakness of web applications using Ethical Hacking," *Proc. 4th Int. Conf. Trends Electron. Informatics, ICOEI 2020*, pp. 354–361, 2020, https://doi.org/10.1109/ICOEI48184.2020.9143018.
[36] T. D. Sobola, P. Zavarsky, and S. Butakov, "Experimental study of ModSecurity web application firewalls," *Proc. - 2020 IEEE 6th Intl Conf. Big Data Secur. Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conf. High Perform. Smart Comput. HPSC 2020 2020 IEEE Intl Conf. Intell. Data Secur. IDS 2020*, pp. 209–213, 2020, https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00045.
[37] M. Agreindra Helmiawan, E. Firmansyah, I. Fadil, Y. Sofivan, F. Mahardika, and A. Guntara, "Analysis of web security using Open Web Application Security Project 10," *2020 8th Int. Conf. Cyber IT Serv. Manag. CITSM 2020*, pp. 1–5, 2020, https://doi.org/10.1109/CITSM50537.2020.9268856.
[38] E. Filiol, F. Mercaldo, and A. Santone, "A method for automatic penetration testing and mitigation: A Red Hat approach," *Procedia Comput. Sci.*, vol. 192, pp. 2039–2046, 2021, https://doi.org/10.1016/j.procs.2021.08.210.
[39] A. Wijayanto, E. Utami, and A. B. Prasetio, "Analysis of vulnerability webserver office management of information and documentation Diskominfo using OWASP scanner," *2020 2nd Int. Conf. Cybern. Intell. Syst. ICORIS 2020*, pp. 1–5, 2020, https://doi.org/10.1109/ICORIS50180.2020.9320833.
[40] R. Johari, I. Kaur, R. Tripathi, and K. Gupta, "Penetration Testing in IoT network," *Proc. 2020 Int. Conf. Comput. Commun. Secur. ICCCS 2020*, pp. 1–7, 2020, https://doi.org/10.1109/ICCCS49678.2020.9276853.
[41] F. Heiding, E. Süren, J. Olegård, and R. Lagerström, "Penetration testing of connected households," *Comput. Secur.*, vol. 126, p. 103067, 2023, https://doi.org/10.1016/j.cose.2022.103067.
[42] A. K. Priyanka and S. S. Smruthi, "WebApplication vulnerabilities:Exploitation and Prevention," *Proc. 2nd Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2020*, pp. 729–734, 2020, https://doi.org/10.1109/ICIRCA48905.2020.9182928.
[43] P. Garrad and S. Unnikrishnan, "Reinforcement learning in VANET penetration testing," *Results Eng.*, vol. 17, p. 100970, 2023, https://doi.org/10.1016/j.rineng.2023.100970.
[44] G. Bella, P. Biondi, S. Bognanni, and S. Esposito, "PETIoT: PEnetration Testing the Internet of Things," *Internet of Things (Netherlands)*, vol. 22, p. 100707, 2023, https://doi.org/10.1016/j.iot.2023.100707.
[45] G. Yadav, K. Paul, A. Allakany, and K. Okamura, "IoT-PEN: A Penetration Testing framework for IoT," *Int. Conf. Inf. Netw.*, pp. 196–201, 2020, https://doi.org/10.1109/ICOIN48656.2020.9016445.
[46] K. Dhiatama Ayunda *et al.*, "Implementation and analysis ModSecurity on web-based application with OWASP Standards," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 8, no. 3, pp. 1638–1650, 2021, https://doi.org/10.35957/jatisi.v8i3.1223.
[47] OWASP, "Cross-Site Request Forgery Prevention," *owasp.org*. https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html.
[48] OWASP, "Content Security Policy," *owasp.org*. https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html.
[49] OWASP, "Cookies_Attributes," *owasp.org*. Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.
[50] OWASP, "Cross Site Script Inclusion," *owasp.org*. Web_Application_Security_Testing/11-Client-side_Testing/13-

Testing_for_Cross_Site_Script_Inclusion.
[51] OWASP, "Authorization Testing," *owasp.org*. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/.
[52] OWASP, "Clickjacking," *owasp.org*. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/09-Testing_for_Clickjacking.
[53] OWASP, "Weak Transport Layer Security," *owasp.org*. Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/01-Testing_for_Weak_Transport_Layer_Security.

BIOGRAPHY OF AUTHORS

Nani Sulisnawati, S1 Informatics, Faculty of Computer Science, Universitas Amikom Yogyakarta. Email: nanisulisnawati10@students.amikom.ac.id

Subektiningsih, Lecturer at Informatics Department, Faculty of Computer Science, Universitas Amikom Yogyakarta. Have an interest in information security and digital forensics. Like the sound of the camera shutter when capturing, and likes to write on a personal blog. Email: subektiningsih@amikom.ac.id. Orcid: https://orcid.org/0000-0001-8632-3190