International Journal of Safety and Security Engineering
Vol. 12, No. 2, April, 2022, pp. 201-208
Journal homepage: http://iieta.org/journals/ijsse

Mitigation Web Server for Cross-Site Scripting Attack Using Penetration Testing Method

Abdul Fadlil[1], Imam Riadi[2], Fahmi Fachri[3*]

[1] Department of Electrical Engineering, Universitas Ahmad Dahlan, Yogyakarta 55164, Indonesia
[2] Department of Information System, Universitas Ahmad Dahlan, Yogyakarta 55164, Indonesia
[3] Department of Informatics, Universitas Ahmad Dahlan, Yogyakarta 55164, Indonesia

Corresponding Author Email: fahmi2007048017@webmail.uad.ac.id

https://doi.org/10.18280/ijsse.120208

Received: 3 February 2022
Accepted: 29 March 2022

Keywords:
web server, cross-site scripting, cyberattack, penetration testing, log

ABSTRACT

The increasing number of user-oriented applications uploading all their information to the web is causing cyber-attacks and data theft. One of the most prevalent vulnerabilities is Cross-Site Scripting (XSS). Intruders take advantage of these attacks to access sensitive user data. This study aims to mitigate XSS attacks by using the penetration testing method as an official effort to improve web server security. The subject of this research uses the login form from the academic information system web server. This study offers a mitigation system prototype against XSS using the penetration test method and the secure code algorithm. This method plays a role in obtaining vulnerability data and security code as a prevention system. The results of this study indicate three categories of web server weaknesses: five at the high level, 164 at the medium level, and 52 vulnerabilities at the low level. Mitigation measures use secure code by denying repeated failed login attempts. These results provide a strategy for web managers to improve security and consider the risk of cyberattacks.

1. INTRODUCTION

Cross-Site Scripting (XSS) attacks threaten industries, organizations, companies, and governments. The attack targets website applications that collect various data and information. XSS attacks allow attackers to obtain data from users for malicious purposes that harm the relevant agencies. Attackers can legitimately enter the system and send a backdoor virus to the user's machine or a central server.

The website application is a medium to deliver information regarding the profile, vision, mission, types of products, and company data [1]. A website application is a task-oriented application used on a web server; this is essential because the webserver is required to maintain and protect the integrity of the information submitted to website users [2].

The use of more and more websites does not guarantee security; increased cyber attacks and theft of sensitive data are the main topics that are often discussed at this time; based on the Global report by Cyber Imperva throughout 2020, web server security in the world has increased by 33% from 2019 [3]. Based on these data, web developers need to consider the security side and risk assessment in designing applications based on these data [4, 5]. When launched without prior security testing, the web becomes an attractive target for attackers [6]. Every user who uses this vulnerable application falls prey to hackers to obtain private data and confidential information [7, 8].

Many security vulnerabilities in web pages result from proven issues Cross-Site Scripting (XSS). Cross-Site Scripting (XSS) is a type of web attack wherein malicious web code, usually in a script, is delivered or executed from a browser on the victim's computer through a user's web application. This execution can filter personal information or steal users [9].

The most common type of attack is cross-site scripting (XSS) [3]. According to top10 2020 (OWASP) Open Web Application Security project statistics [10], XSS attack is one of the top web application vulnerabilities in recent years; more than 60% of websites are still vulnerable to XSS attacks [11]. Cookies serve as an objective to hijack identities in fraudulent sessions, thus offering attackers the possibility of stealing sensitive data or even taking control of certain devices. XSS delivers 40% attack attempts, SQL injection (SQLi) 24%, so-called cross-section attacks 7%, local file inclusion (LFI) 4% and in last place in distributed denial of service (DDoS) with 3% [12].

Several studies have tested web server security against cross-site scripting attacks with various methods, including Machine Learning and n-Gram Methods [13]; *Multilayer Perceptron Technique* [14]; *OWASP Security Shepherd* [15]. In addition, there is also research that uses the penetration testing method [16]. However, so far, most of the penetration testing on cross-site scripting attacks is only limited to vulnerability assessment, exploitation surveys, and identification of system loopholes. While this research uses five stages of penetration testing, according to chipper [17], which not only assesses system security but also includes the process of system improvement (post-exploitation) and reporting of results before and after repairs.

Evaluating information security systems on websites consists of three main things: hosting providers, website protocols such as HTTP/HTTPS, and website platforms. The evaluation aims to determine the vulnerability of the system [18]. This research focuses on mitigating XSS attacks through the website platform. The vulnerability evaluation process uses penetration testing methods intending to help management glimpse the security of their web pages from an

attacker's point of view [19].

In a security case research, this research aims to find Vulnerability or security holes on the webserver against Cross-Site Scripting attacks using the Penetration Testing Method to be mitigated immediately.

2. LITERATURE REVIEW

Here are some studies that have made improvements to web servers against Cross-site scripting attacks.

Rodríguez et al. conducted mitigation research on Cross-Site Scripting vulnerabilities with the results of being able to mitigate these attacks showing that the trend is increasing in proposals that analyze web page content (13.20%), as well as those that serve as toolkits for web browsers (16.98%) using artificial intelligence techniques [12]. His research explained that XSS vulnerabilities could be exploited by stealing the victim's cookie and making the cookie login remotely, namely by utilizing the CORS (Cross-Origin Resource Sharing) working principle on di browser web [20]. Gunawan et al. conducted research on XSS attacks by sending HTTP GET requests to a web server using Java Script that connected all web browsers to the BeFF tool on Kali Linux. The attack exploited the victim's web server and gained remote access to the victim's machine [21]. Omer et al. conducted mitigation research showing that released patches for existing vulnerabilities at the operating system (OS) level and in Software programs do not entirely prevent XSS cyber attacks.

On the other hand, producing company-specific patches and fixing Software bugs by realizing that software running on each system can give better results [22]. Lei et al. research describes three stages of XSS vulnerability repair using the Short-Term Long Memory (LSTM) method, namely: first, the data needs to be processed using decoding technology to return the XSS code to an unencoded state to improve code readability. The second uses word2vec to extract the XSS payload features and maps them to vector features. Third, improve the LSTM model by adding a more effective attention mechanism. The experimental results show that the proposed XSS detection model based on the LSTM achieves a 99.3% resistance rate and a 98.2% recall rate in the data set collected. This method can help minimize website security system vulnerabilities from XSS attacks [23].

2.1 Cross-site scripting (XSS)

XSS is an attack that exploits a security vulnerability to insert malicious scripts into web pages.

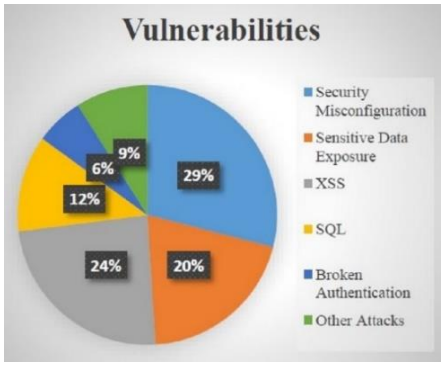

Figure 1. Survey on current vulnerability

The script will hand redirect the user to a website that has been designed to be able to retrieve the user's cookies or session [24]. This attack will have originated from the site or a reliable source. As a result, until it has complete access to the web system, a malicious script can access any information stored in the browser, such as cookies, session tokens, or sensitive information [25].

Figure 1 provides information that the Open Web Application Security Project (OWASP) has summarized the current top web application vulnerabilities; more than 24% of websites are still vulnerable to XSS attacks and are at the second-largest vulnerability level [26]. XSS is generally divided into 3 three types of attacks, namely:

2.1.1 Reflected XSS

For attackers, reflected XSS is the most prevalent and essential form of XSm [23]. Attackers employ social engineering to get visitors to visit a website that contains harmful malware. Next, the Intruder gains access to legitimate users' cookies and carries out his malicious purposes, as shown in Figure 2.

One way to prevent this attack is to verify each input data and ensure that the data complies with the system requirements—conditions such as maximum and minimum character length, use of special characters, and use of numbers. If the data does not match, the system will display a warning and generate a warning log. The warning log becomes an evaluation material to increase the next system update.

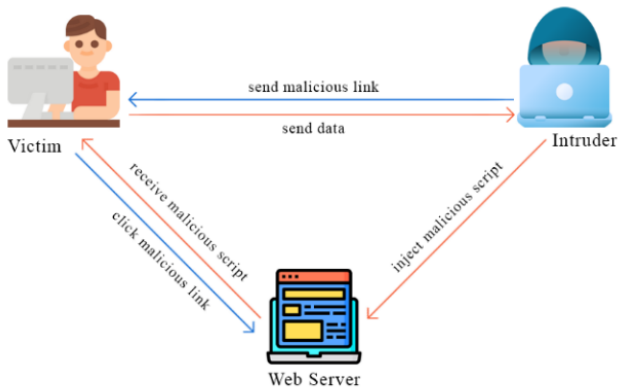

Figure 2. Attack type reflected XSS

2.1.2 Stored XSS

Persistent (or stored) XSS issues are a more dangerous version of a typical attack. When attacker data gets into the server system, users continually access and execute XSS code unknowingly by luring users to third-party websites. These attacks allow theft of user data through search engines and execute a backdoor virus. The virus will give attackers access to carry out their malicious purposes. Persistent XSS vulnerabilities can be more severe than other varieties. In addition, persistent XSS can spread to all accounts, create client-side worms, login legitimately, and will not stop even if the user restarts the machine. Persistet stored XSS refers to Figure 3.

2.1.3 DOM based XSS

DOM-Based XSS (or as it is called in some texts, "type-0 XSS") is an XSS attack in which the attack is implemented as a result of modification of the DOM "like environment" in the

victim's browser used by the original client-side script so that client-side code running in an "unexpected" way. To deliver a DOM-based XSS attack, one needs to put data into a source that propagates to the sink and causes arbitrary JavaScript execution. The XSS DOM is the URL that typically accesses the window location. The object can generate a link to lead the victim to a vulnerable page with a payload in the query string and part of the URL fragment. Like targeting a 404 page or a PHP website, the payload can also be placed in the path. If an error occurs, the XSS script cannot run, and the payload will be the error or delay, as shown in Figure 4.

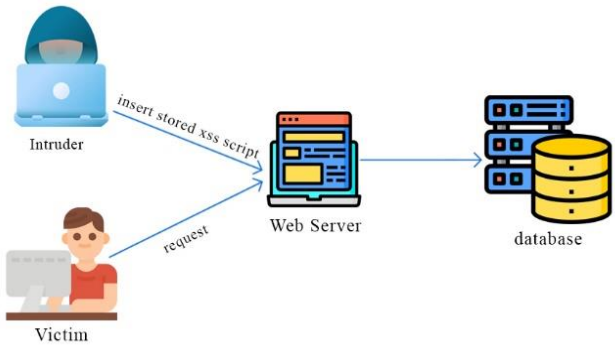

Figure 3. Attack type stored XSS

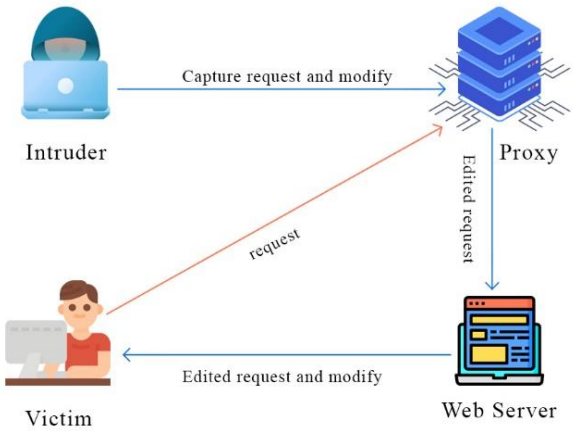

Figure 4. Attack type DOM based XSS

2.2 System models

In this study, virtualization is carried out in a cloud network environment, and penetration testing is used to complete the mitigation process.

2.2.1 Cloud network environment

Cloud Network Environment [24] is a technology that allows users to share hosts, access distributed environments, and virtualize—using the VirtualBox program to create a virtual environment to conduct valid and realistic assault trials. This virtual server and attacker operating systems are available [25]. They are making it possible to do Mitigation to prevent web server vulnerabilities. The environmental structure design in this research is shown in Figure 5.

Figure 5 provides information on the virtual environment infrastructure, consisting of three parts: the Intruder as the attacker, the web server acts as the initial target of the attack before obtaining essential data, and the database serves as the primary purpose of cyberattacks on the overall data storage.

The following is the network configuration data for each device, as shown in Table 1.

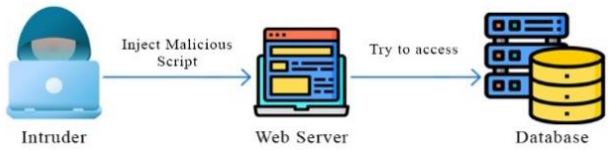

Figure 5. Environment structure

Table 1. Configuration network

Device	OS	IP Address	Port
Server	Ubuntu 20.04	192.168.1.14	Enp0s3
Attacker	Parrot OS	192.168.1.20	eth0

Table 1 shows that this environment consists of two devices: Linux Ubuntu as the server and Parrot Os as the attacker.

2.2.2 Penetration testing

Penetration testing is a structured process to test an organization's computing base, including hardware, software, and people. This process includes analyzing the organization's overall computing system, looking for vulnerabilities such as system configuration, software, and hardware errors, and its operational processes to identify weaknesses [27]. Penetration testing aims to secure an organization by imitating what attackers do. This helps determine the various vulnerabilities that can damage the system and fix them before they happen.

3. THE PROPOSED APPROACH

3.1 Research subject

The research subject studied in this research was a web server on the Academic Information System. Researchers analyzed system security using the Penetration Testing method. Researchers also analyzed the tools' ability to discover what vulnerabilities were found on the webserver. Penetration Testing is expected to improve and enhance the web server's security.

3.2 Experiment research stages

This experimental study is to mitigate against XSS attacks using two secure code algorithms. This research consists of three stages: the first stage of gathering information about the webserver, which involves searching for vulnerabilities, determining the type of attack, and the attack process. The second stage includes corrective actions against web server vulnerabilities using a mitigation system.

The third stage is to test the attack again on the system to determine which Mitigation can work to minimize attacks. Each stage has its role. The first stage focuses on identifying vulnerabilities, the second stage on improvement efforts based on vulnerabilities, and the third stage as a verification step against improvement efforts in the previous stage [28]. The flow chart of the experimental stages in this research refers in Figure 6.

Figure 6 shows how the workflow experiment in this study is used to implement Mitigation. There are five steps to follow:

1. *Intelligence Gathering*, gathering information about web servers.
2. *Vulnerability analysis* is carried out by scanning for vulnerabilities and determining the types of attacks that can be exploited.
3. *Exploitation* attacked the vulnerabilities found and tested whether they could be exploited.
4. *Post-Exploitation* this stage, improvements are made, solutions are applied to vulnerabilities, and a retest is carried out.
5. *Reporting*: This is the report generation section based on the analysis results in vulnerability testing results before and after repairs.

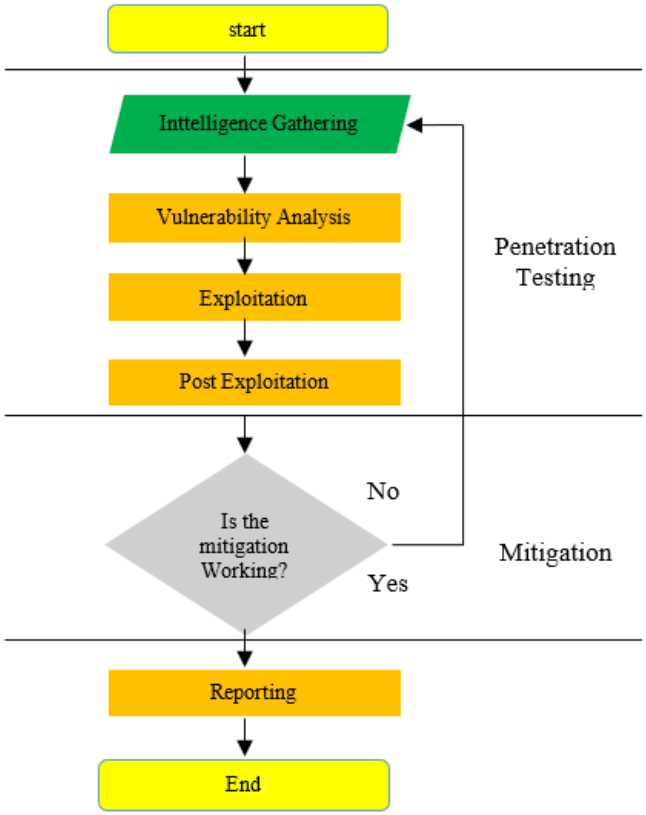

Figure 6. Flowchart of experiment research stages

3.3 Forensic tools

This section defines the forensic tools that play an essential role in simulating attacks against web servers, as shown in Table 2.

Table 2. Forensic tools

No	Tools	Version	Function
1	Parrot	5.10	OS Attacker
2	Ubuntu	20.05	OS Server
3	VirtualBox	6.1.32	Research Environment
4	NMAP	7.9.1	Intelligence Gathering
5	Whois	Web App	Intelligence Gathering
6	Nikto	2.1.6	Vulnerability analysis
7	Acunetix	11.0	Vulnerability analysis
8	Wireshark	3.4.3	Capture Network

Table 2 presents a list of data from tools and materials to perform a series of simulations. The attacker carries out the attack process using the Parrot Security Operating System. Parrot OS provides complete tools to attack web servers and perform Penetration Testing, namely NMAP, Whois, Nikto, and Acunetix.

4. RESULT AND DISCUSSION

Penetration testing in this research is done by simulating, reviewing, and evaluating the Academic Information System on the server. The penetration testing analysis used in this research includes several stages, namely intelligence gathering, vulnerability analysis, exploitation, post-exploitation, and reporting.

4.1 Intelligence gathering

The first step is Intelligence Gathering, which is a process of direct action against the target taken [29]. Intelligence gathering aims to investigate the problem and determine the right tools for the next phase. The information-gathering process is divided into active and passive information gathering [30]. Collecting information using active information gathering techniques using NMAP (Network Exploration or Security Auditing) tools, while passive information gathering using tools which is lookup.

4.1.1 NMAP (Network exploration or security auditing)

```
┌[fahmi@parrot]-[~]
└─────$nmap -sV -p- -sT 192.168.1.14
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-25 00:01 WIB
Nmap scan report for 192.168.1.14
Host is up (0.020s latency).
Not shown : 65533 filtered ports

PORT        STATE       SERVICE             VERSION

22 /tcp     open        ssh                 OpenSSH 8.2p1 Ubuntu 4ubuntu0.2
80/tcp      open        http                Apache httpd 2.4.41 (Ubuntu)
Service Info:    OS:     Linux; CPE: cpe:/o:Linux: linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done:  1 Ip address  (1 host up) scanned in 135.44 seconds
```

Figure 7. Information web server system

Based on Figure 7, it is known that the scan results using Nmap on the Academic Information System server display information about ports on the system that are still open, one of which is port 22 SSH (Secure Shell) and port 80 HTTP (Hypertext Transfer Protocol) running on IP 192.168.1.14.

4.1.2 Who is lookup

Whois is an internet service that provides information about a domain. *Whois Lookup* aims to display information on a particular domain, such as status, registration, address, contact, name server, and other information [31]. In addition, this application also provides information related to a domain that is available or has been taken by someone else, Figure 8.

Figure 8 shows the results of passive *information gathering*. The front of the web is a page where users can see and interact directly at the beginning of a visit before entering the system and getting all the information contained therein.

Figure 8. Front end website login

4.2 Vulnerability analysis

In this second step, take a systematic and proactive approach to find vulnerabilities by identifying, classifying, and prioritizing vulnerabilities in the system. Generally, this section is helpful to websites, applications, and network infrastructure [31]. This research uses Nikto and Acunetix tools. Nikto can assess website server security systems to identify specific security holes. Acunetix can detect and notify various kinds of vulnerabilities in website applications.

4.2.1 Nikto

Nikto is a web application security scanner tool that functions to find security gaps. Nikto is also a vulnerability assessment (VA) application designed as a hidden tool for evaluating systems or finding system weaknesses [32]. The results of the analysis using the Nikto application can be seen in Figure 9.

```
┌[fahmi@parrot]-[~]
└─────$nikto –h 192.168.1.14
-Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP          : 192.168.1.14
+ Target Hostname    : 192.168.1.14
+ Start Time         : 2021-12-25 00.05.48 (GMT7)
---------------------------------------------------------------------------
+ Server: Apache/2.4.41 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined.
+ The X-Content-Type-Options header is not set.
+ /: Output from the phpinfo() function was found.
+ /index.php: Output from the phpinfo()function was found.
+ No CGI Directories found (use '-C all ' to force check all possible dirs)
+ Web Server returns a valid response with junk HTTP methods.
```

Figure 9. Scanning vulnerability Nikto

Nikto's scanning results produce information about several vulnerabilities, namely:

a. *Anti-clickjacking*
b. *X-XSS (Cross-site Scripting)*
c. *X-Frame Header Options are Missing*
d. *Indikasi file /index.php*
e. */: Output the phpinfo() was found*
f. *HTML Form without CSRF Protection*

4.2.2 Acunetix

Figure 10 provides information on the acunetix application, which can detect and notify various vulnerabilities in applications built on various platforms such as WordPress, PHP, ASP.NET, Java Frameworks, and Ruby on Rails [5]. Figure 10 shows the results of a vulnerability scan using the acunetix application, namely:

a. 5 Vulnerabilities are at High Severity Vulnerabilities
b. 164 Vulnerability is at Medium Severity Vulnerabilities
c. 52 Vulnerabilities are at Low Severity Vulnerabilities

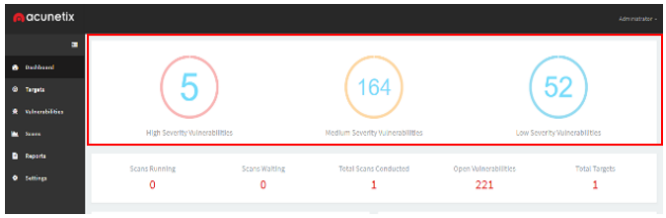

Figure 10. Report scan acunetix

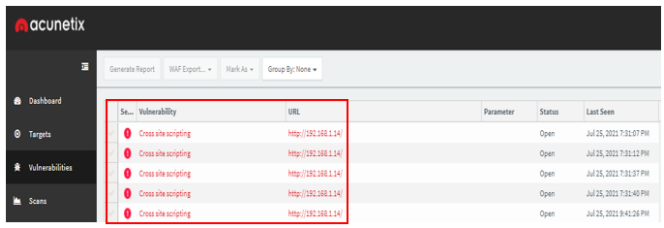

Figure 11. Result scan vulnerability

Figure 11 shows that the webserver security level in Acunetix tools is at a dangerous (high) level. At this level, hackers are very easy to exploit, which may be done from viewing, changing, deleting data, creating new accounts to endanger the web, and taking over the website's function completely [33]. Based on the results of scanning with both tools, it can be concluded that the vulnerability to Cross-site scripting attacks is in the High category. Therefore, it is necessary to analyze to maintain system security from outside attacks.

4.3 Exploitation

The third step is to take advantage of the XSS vulnerability by injecting the payload into the webserver. This exploitation stage focuses on all possible XSS attacks, thereby minimizing the vulnerability to reappearance [34].

4.3.1 Wireshark

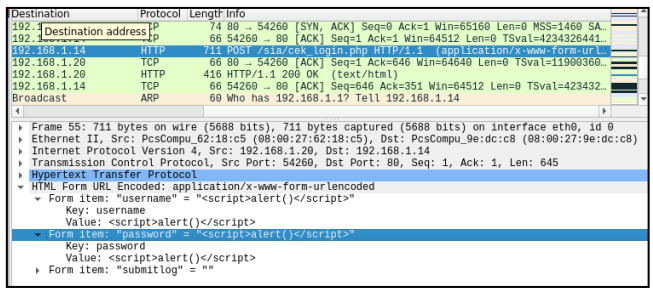

Figure 12. Network traffic log

The network traffic log in Figure 12 shows a cross-site script attack, starting with the attacker launching an attack on the server using a malicious script on the user and password. The use of target port is port 80, the length of the batch is 416, and the HTTP and IP protocols are 192.168.1.20, namely computers with operating systems using Parrot OS.

4.3.2 Cross-site scripting

In this case, the attacker's payload becomes part of the request sent to the webserver. This is then reflected such that the HTTP response includes the payload of the HTTP request. The reflected XSS Payload is then executed in the user's browser. The exploitation web refers to Figure 13.

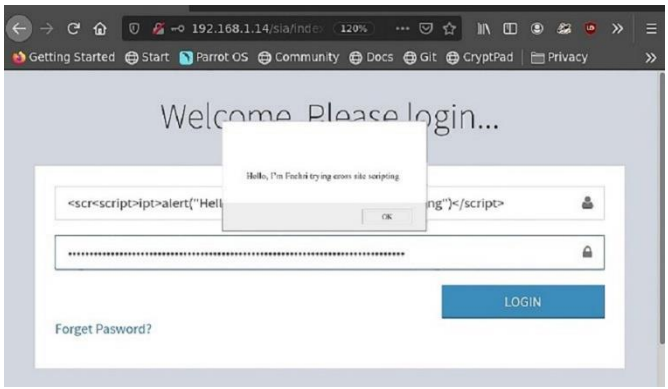

Figure 13. Exploitation web target

The result of Figure 13 is an example of a web server system that was successfully attacked in a cross-site scripting security vulnerability. The web server vulnerability gets an alert message with the script *<scr<script>ipt>alert("Hello, I'm Fachri trying cross-site scripting")</script>*. Attackers use social engineering to invite users to click on URLs embedded with malicious code. This helps the Intruder obtain the user's cookies, which can then be used to hijack the user's session. This result follows Buyukkayhan et al. research [35], which states that cross-site scripting vulnerabilities are dangerous in tampering with web server security.

4.4 Post exploitation

The next stage in penetration testing is post-exploitation. At this stage, a repair process is carried out on the webserver based on the weaknesses found, and the appropriate solution is determined to overcome the vulnerability [36]. The best way to solve this problem is to provide an anti-injection source code configuration, i.e., username and password do not use special characters. Post-exploitation consists of two stages. The first stage is adding a secure code algorithm to the webserver. The second stage is retesting to ensure the application of the Algorithm has functioned as intended.

4.4.1 Mitigation web server

Based on the results obtained in the above exploitation stage, the mitigation process needed aims to stop the spread of hackers against cross-site scripting attacks by configuring the server and adding the security code of the Algorithm. Fixing the vulnerability to cross-site Scripting attacks is done by configuring the server. The Mitigation refers to Figure 14.

```
<?php
session_start ();
include "konek/koneksi.php";
include "konek/anti_sql_injection.php";

//$kode=anti_sql_injection($_POST['kode']);
$username=anti_sql_injection($_POST['username']);
$pass=anti_sql_injection(md5($_POST['username']));

$injeksi_username = mysqli_real_escape_string($konek, $username);
$injeksi_password = mysqli_real_escape_string($konek, $pass);
if (!ctype_alnum($injeksi_username) OR !ctype_alnum($injeksi_password)){
        echo "sorry, you can't attack this website";
```

Figure 14. Mitigation web server with algorithm1

Figure 14 provides information on the implementation of algorithm1 to prevent injection through special characters in the username and password form. If the user uses one or more special characters, the Algorithm will reject the incoming data process, thus preventing the injection process of the database. The following is the application of Algorithm2 to validate the input data, as shown in Figure 15.

```
InputValidation() {
        for     (let I = 1; i < this.Input.length; i++) {
                const currentData = this.Input[i];
                const previousData = this.Input[i - 1];
                if (currentData.hash ! == currentData.calculateHash() ) {
                        return false;}
                if (currentData.previousHash !== previousData.hash) {
                return false;}}
                        return true; }
```

Figure 15. Mitigation web server with algorithm2

Figure 15 the InputValidation() code tests the validity of the input made and ensures that the data loaded has not changed. The Algorithm will check the input data by matching them to the system. Valid data will produce True output, and if invalid data, it will give False output.

4.4.2 Attack test simulation

Test attacks are carried out when the server conditions have been improved. Attack trials were carried out from the attacker's host and simulated cross-site scripting attacks on the server, as shown in Figure 16.

Figure 16. Rejection attack on a web server

4.5 Reporting

The fifth or final stage of the Penetration testing method is the reporting stage. This shows the effectiveness of the penetration testing method by comparing the results of exploitation with the post-exploitation stage. The vulnerability testing results produce an effective solution by rejecting suspicious data input into the system. This solution is a mitigation effort by adding a secure code algorithm to the source code on the webserver so that if an injection attack attempts on the webserver occur, the system will reject the attempt.

Table 3. Result report mitigation

No	Mitigation	Function	Result	Status
1	Algorithm1	Identification Special Character	Block	Success
2	Algorithm2	Validation Input Data	Block	Success

Table 3 provides information on the implementation of secure code mitigation using two algorithms. Algorithm1 functions to prevent injection via special characters (*, %, #, <), and algorithm2 focuses on preventing data input that does not match the database system. The result column with the "Block" indicator means that the XSS attack test was

successfully rejected. The status column with the "success" indicator shows that the implementation of Mitigation using secure code is running successfully to prevent injection attacks against the webserver.

5. CONCLUSIONS

Implementing source code improvements applied to the webserver is proven to prevent cross-site scripting attacks and block attackers. Researchers also found several vulnerabilities in the Academic Information System server. There are also types of vulnerabilities categorized into three categories, namely five high levels, 164 medium levels, and 52 low levels. Simulation of attacks carried out by injecting scripts on the server has proven to be successful in penetrating the system by displaying an alert message. System improvement is made by configuring the source code contained on the server and has proven successful in preventing attacks and thwarting attackers' access to enter the system. The results of the attack simulation using the Penetration Testing method can provide information on vulnerabilities and accelerate IT in overcoming structured and systematic attacks on Academic Information Systems, especially against cross-site scripting attacks. This research has limitations, namely, only testing on servers that use the reflected XSS feature. Therefore, further research is recommended to test other types of features to increase the generalizability of the results of this research.

REFERENCES

[1] Arnaldy, D., Perdana, A.R. (2019). Implementation and analysis of penetration techniques using the man-in-the-middle attack. In 2019 2nd International Conference of Computer and Informatics Engineering (IC2IE), pp. 188-192. https://doi.org/10.1109/IC2IE47452.2019.8940872

[2] Prajapati, P., Patel, N., Shah, P. (2019). A review of recent detection methods for HTTP DDoS attacks. International Journal of Scientific and Technology Research, 8(12): 1693-1696.

[3] Malviya, V.K., Rai, S., Gupta, A. (2021). Development of web browser prototype with embedded classification capability for mitigating Cross-Site Scripting attacks. Applied Soft Computing, 102: 106873. https://doi.org/10.1016/j.asoc.2020.106873

[4] Toapanta, S.M., Quimis, O.A.E., Gallegos, L.E.M., Arellano, M.R.M. (2020). Analysis for the evaluation and security management of a database in a public organization to mitigate cyber attacks. IEEE Access, 8: 169367-169384. https://doi.org/10.1109/ACCESS.2020.3022746

[5] Surian, R.U., Abd Rahman, N.A., Nathan, Y. (2020). Nscanner: Vulnerabilities detection tool for web application. Journal of Physics: Conference Series, 1712(1): 012018. https://doi.org/10.1088/1742-6596/1712/1/012018

[6] Toapanta, S.M.T., Ochoa, I.N.C., Sanchez, R.A.N., Mafla, L.E.G. (2019). Impact on administrative processes by cyberattacks in a public organization of Ecuador. In 2019 Third World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4), pp. 270-274. https://doi.org/10.1109/WorldS4.2019.8903967

[7] Priyanka, A.K., Smruthi, S.S. (2020). WebApplication Vulnerabilities: Exploitation and Prevention. In 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 729-734. https://doi.org/10.1109/ICIRCA48905.2020.9182928

[8] Chen, Q. (2020). Research on the implementation method of database security in management information system based on big data analysis. In E3S Web of Conferences, 185: 02033. https://doi.org/10.1051/e3sconf/202018502033

[9] Zubarev, D., Skarga-Bandurova, I. (2019). Cross-site scripting for graphic data: Vulnerabilities and prevention. In 2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT), pp. 154-160. https://doi.org/10.1109/DESSERT.2019.8770043

[10] Open Web Application Security Project. (2021). Top 10 Web Application Security Risks.

[11] Salas, M.I.P., Martins, E. (2014). Security testing methodology for vulnerabilities detection of XSS in web services and WS-security. Electronic Notes in Theoretical Computer Science, 302: 133-154. https://doi.org/10.1016/j.entcs.2014.01.024

[12] Rodríguez, G.E., Torres, J.G., Flores, P., Benavides, D.E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks, 166: 106960. https://doi.org/10.1016/j.comnet.2019.106960

[13] Habibi, G., Surantha, N. (2020). XSS attack detection with machine learning and n-gram methods. In 2020 International Conference on Information Management and Technology (ICIMTech), pp. 516-520. https://doi.org/10.1109/ICIMTech50083.2020.9210946

[14] Mokbal, F.M.M., Dan, W., Imran, A., Lin, J., Akhtar, F., Wang, X. (2019). MLPXSS: An integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. IEEE Access, 7: 100567-100580. https://doi.org/10.1109/ACCESS.2019.2927417

[15] Wibowo, R.M., Sulaksono, A. (2021). Web Vulnerability Through Cross Site Scripting (XSS) Detection with OWASP Security Shepherd. Indonesian Journal of Information Systems, 3(2): 149-159. https://doi.org/10.24002/ijis.v3i2.4192

[16] Liu, M., Zhang, B., Chen, W., Zhang, X. (2019). A survey of exploitation and detection methods of XSS vulnerabilities. IEEE Access, 7: 182004-182016. https://doi.org/10.1109/ACCESS.2019.2960449

[17] Chipher. (2020). A complete guide to the phases of penetration testing. 2020 5th International Conference on Computer and Communication Systems, ICCCS 2020, Farmingdale, NY 11735 United States.

[18] Nirmal, K., Janet, B., Kumar, R. (2018). Web application vulnerabilities-the hacker's treasure. In 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 58-62. https://doi.org/10.1109/ICIRCA.2018.8597221

[19] Pandey, R., Jyothindar, V., Chopra, U.K. (2020). Vulnerability Assessment and Penetration Testing: A portable solution Implementation. In 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 398-402. https://doi.org/10.1109/CICN49253.2020.9242640

[20] Nirmal, K., Janet, B., Kumar, R. (2018). It's more than stealing cookies-exploitability of XSS. In 2018 Second

International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 490-493. https://doi.org/10.1109/ICCONS.2018.8663230

[21] Gunawan, T., Lim, M.K., Kartiwi, M., Malik, N.A., Ismail, N. (2018). Penetration testing using Kali Linux: SQL injection, XSS, Wordpres, and WPA2 attacks. Indonesian Journal of Electrical Engineering and Computer Science, 12(2): 729-737. https://doi.org/10.11591/ijeecs.v12.i2.pp729-737

[22] Aslan, Ö., Samet, R. (2017). Mitigating cyber security attacks by being aware of vulnerabilities and bugs. In 2017 International Conference on Cyberworlds (CW), pp. 222-225. https://doi.org/10.1109/CW.2017.22

[23] Lei, L., Chen, M., He, C., Li, D. (2020). XSS detection technology based on LSTM-attention. In 2020 5th International Conference on Control, Robotics and Cybernetics (CRC), pp. 175-180. https://doi.org/10.1109/CRC51253.2020.9253484

[24] Gupta, S., Gupta, B.B. (2017). Cross-Site Scripting (XSS) attacks and defense mechanisms: Classification and state-of-the-art. International Journal of System Assurance Engineering and Management, 8(1): 512-530. https://doi.org/10.1007/s13198-015-0376-0

[25] Mahmoud, S.K., Alfonse, M., Roushdy, M.I., Salem, A.B.M. (2017). A comparative analysis of Cross Site Scripting (XSS) detecting and defensive techniques. In 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 36-42. https://doi.org/10.1109/INTELCIS.2017.8260024

[26] Patel, K. (2019). A survey on vulnerability assessment & penetration testing for secure communication. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 320-325. https://doi.org/10.1109/ICOEI.2019.8862767

[27] Al Shebli, H.M.Z., Beheshti, B.D. (2018). A study on penetration testing process and tools. In 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT), pp. 1-7. https://doi.org/10.1109/LISAT.2018.8378035

[28] Ye, Y., Guo, J., Xu, X., Li, Q., Liu, H., Di, Y. (2019). High-risk problem of penetration testing of power grid rainstorm disaster artificial intelligence prediction system and its countermeasures. In 2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2), pp. 2675-2680. https://doi.org/10.1109/EI247390.2019.9062097

[29] Kothia, A., Swar, B., Jaafar, F. (2019). Knowledge extraction and integration for information gathering in penetration testing. In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 330-335. https://doi.org/10.1109/QRS-C.2019.00068

[30] Sadigh, D., Landolfi, N., Sastry, S.S., Seshia, S.A., Dragan, A.D. (2018). Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. Autonomous Robots, 42(7): 1405-1426. https://doi.org/10.1007/s10514-018-9746-1

[31] Gautam, R. (2016). Analysis and implementation of WHOIS domain lookup. International Journal of Technical Research & Science, 20(1-2): 21-37.

[32] Rahman, M.A., Amjad, M., Ahmed, B., Siddik, M.S. (2020). Analyzing web application vulnerabilities: an empirical study on e-commerce sector in Bangladesh. In Proceedings of the International Conference on Computing Advancements, pp. 1-6. https://doi.org/10.1145/3377049.3377107

[33] Amankwah, R., Chen, J., Kudjo, P.K., Towey, D. (2020). An empirical comparison of commercial and open-source web vulnerability scanners. Software: Practice and Experience, 50(9): 1842-1857. https://doi.org/10.1002/spe.2870

[34] McKinnel, D.R., Dargahi, T., Dehghantanha, A., Choo, K.K.R. (2019). A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Computers & Electrical Engineering, 75: 175-188. https://doi.org/10.1016/j.compeleceng.2019.02.022

[35] Buyukkayhan, A.S., Gemicioglu, C., Lauinger, T., Oprea, A., Robertson, W., Kirda, E. (2020). What's in an Exploit? An Empirical Analysis of Reflected Server {XSS} Exploitation Techniques. In 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), pp. 107-120.

[36] Rapley, A., Bellekens, X., Shepherd, L.A., McLean, C. (2018). Mayall: A framework for desktop JavaScript auditing and post-exploitation analysis. In Informatics, 5(4): 46. https://doi.org/10.3390/informatics5040046