XSS ATTACK DETECTION AND MITIGATION USING MULTI-LAYER SECURITY MECHANISM (MLSM)

[1]Hartono [2]Sriyanto
hartono@umko.ac.id, sriyanto@darmajaya.ac.id

[1]Universitas Muhammadiyah Kotabumi, [2]Institut Informatika dan Bisnis Darmajaya

Abstrak: BSSN menyebut bahwa terdapat 12,9 juta ancaman siber di Indonesia selama tahun 2018. Pada Januari – April 2020, jumlah serangan siber meningkat tinggi. Dalam empat bulan tersebut, jumlah serangan siber mencapai 88 juta. Metode, aplikasi, dan teknik serangan yang digunakan tidak dapat terindentifikasi secara mudah. Namun, menurut data dari OWASP Top Ten Tahun 2017 dan 2021 (*statistics based proposal),* terdapat 10 celah keamanan website yang paling sering dieksploitasi. XSS menjadi salah satu celah keamanan yang masuk dalam daftar tersebut. Dampak XSS sangat fatal, karena penyerang dapat melakukan *account takeover,* pencurian data pribadi, dan sebagainya. Terdapat beberapa penelitian yang telah mengimplementasikan mekanisme mengatasi serangan XSS. Namun, implementasi tersebut belum mendapatkan hasil yang efektif dan holistik. Mekanisme yang diujicobakan penelitian sebelumnya tidak dapat mengatasi setiap jenis serangan XSS dan dilakukan secara tidak representatif. Salah satu penyebab kegagalan metode sebelumnya adalah penggunaan mekanisme *single layer security*. Oleh karena itu, tujuan penelitian ini adalah menguji efektivitas mekanisme *multi layer security* (MLSM) dalam mendeteksi dan memitigasi serangan XSS. MLSM terdiri dari lima lapisan, yaitu OWASP ModSecurity, Framework/CMS *Security Feature, HTTP Middleware, Templating Engine*, dan *Data Sanitizer*. Untuk menguji tingkat keamanan MLSM, peneliti melakukan simulasi serangan menggunakan aplikasi Arachni dan ZAP pada *sample website* yang memiliki 170 celah keamanan XSS. Berdasarkan uji coba serangan ke website non MLSM, Arachni sukses mengeksekusi 168 dari 170 (98,82%) dan ZAP mengeksekusi 103 dari 170 (60,58%) serangan XSS. Namun, setelah mengimplementasikan MLSM pada website, serangan Arachni dan ZAP gagal dalam melakukan serangan XSS, baik *stored, reflected,* dan DOM *based* XSS. Tidak ada satu jenis pun serangan XSS yang dapat dilakukan pada website MLSM.

Kata kunci: Cross Site Scripting, XSS, Cyber Security, Multi Layer Security, OWASP

Abstract: *BSSN stated that there were 12.9 million cyber threats in Indonesia during 2018. In January - April 2020, the number of cyber-attacks increased. In those four months, the number of cyberattacks reached 88 million. The methods, applications, and attack techniques used cannot be identified easily. However, according to data from the OWASP Top Ten in 2017 and 2021 (statistics-based proposal), there are 10 website security vulnerabilities that are most often exploited. XSS is one of the security holes included in the list. In addition to being a loophole that is often found, the impact of XSS is very fatal, because it allows attackers to do account takeovers, theft of personal data, and so on. There are several studies that have implemented mechanisms to detect and mitigate XSS attacks. However, the implementation has not yet obtained effective and holistic results. The mechanism tested by previous research still leaves a security problem that allows attackers to execute XSS attacks. One of the things that cause this problem is the use of a single-layer security mechanism. Therefore, the purpose of this study is to test the effectiveness of the multi-layer security (MLSM) mechanism in detecting and mitigating XSS attacks. MLSM consists of five layers, namely OWASP ModSecurity, Framework/CMS Security Feature, HTTP Middleware, Templating Engine, and Data Sanitizer. To test the security level of MLSM, the researchers conducted a simulation of attacks*

using the Arachni and ZAP applications on a sample website that had 170 XSS security vulnerabilities. Based on test attacks on non-MLSM websites, Arachni successfully executed 168 of 170 (98.82%), and ZAP executed 103 of 170 (60.58%) XSS attacks. However, after implementing the MLSM feature on the website, Arachni and ZAP attacks failed to perform XSS attacks, both stored, reflected, and DOM-based XSS. There is no single type of XSS attack that can be carried out on MLSM websites.

***Keywords**: Cross Site Scripting, XSS, Cyber Security, Multi Layer Security, OWASP*

1. INTRODUCTION

Based on data from the National Cyber and Crypto Agency (BSSN), there were around 12.9 million cyber threat attempts to Indonesia during 2018. A total of 513,900 of the total attacks were malware (*Serangan Siber Ancam Indonesia - Infografik Katadata.co.id*, 2019). Not only that, during January - April 2020, BSSN recorded that there were around 88,414,296 cyber-attack activities in Indonesia (*Rekap Serangan Siber (Januari – April 2020) | Bssn.Go.Id*, n.d.). Besides being motivated by various motivations, the technology used is also varied, sophisticated, and difficult to detect. When viewed from the type, there are three types of cyber threats, namely attacks, crimes, and cyber terrorism (Giri, 2019). These three cyber threats have a very dangerous potential for national security. Based on a review of the attack methods used, the three types of cyber threats can actually use the same method (Buchanan, n.d.). It means that even one attack method can be used to carry out all three types of cyber threats. Therefore, in an effort to overcome these problems, the detection and mitigation of cyber threat methods must be formulated well, holistically, and comprehensively.

One method that is often used to exploit website security vulnerabilities is the Cross-Site Scripting (XSS) method. This method takes advantage of the XSS security vulnerability, because the website does not validate or filter the submitted input, either through forms, URLs, or DOM (Document Object Manipulation). The XSS method uses Javascript codes to inject the page through the client-side. This security vulnerability is not a new security vulnerability. However, in reality, XSS is still one of the security holes that has been included several times in the list of Open Web Application Security Project (OWASP) Top-10 Web Vulnerabilities ("OWASP Top-10 2021, Statistically Calculated Proposal.," n.d.). OWASP is an international organization that focuses on research and development of security systems, as well as handling internet security issues from cyber threats (Marchand-Melsom & Nguyen Mai, 2020). It shows that XSS security vulnerabilities are important to research

Table 1. OWASP Top-10 Vulnerabilities 2017 and 2010

OWASP Top 10 2017		OWASP Top 2021 Proposal	
A1	Injections	A1	Injections
A2	Broken Authentification	A2	Broken Authentification
A3	Sensitive Data Exposure	A3	Cross-Site Scripting (XSS)
A4	XML External Entities (XXE)	A4	Sensitive Data Exposure
A5	Broken Access Control	A5	Insecure Deserialization
A6	Security Misconfiguration	A6	Broken Access Control
A7	Cross-Site Scripting (XSS)	A7	Insufficient Loggin & Monitoring
A8	Insecure Deserialization	A8	Server Side Request Forgery (SSRF)
A9	Known Vulnerabilties	A9	Known Vulnerabilities
A10	Insufficient Logging & Monitoring	A10	Security Misconfiguration

Based on data from OWASP Top-10 data in 2017, XSS occupies the 7th position in the list of security vulnerabilities that are most often found in web-based applications worldwide (Søhoel, n.d.). Not only that, in the OWASP Top-10 2021 (statistics-based proposal), the position of the XSS security vulnerability rose to third ("OWASP Top-10 2021, Statistically Calculated Proposal.," n.d.). This fact shows that XSS is a security vulnerability that is still a problem on many websites. Judging from the risks, XSS can cause very serious problems, such as theft of user login data, transfer of confidential information, manipulating website content, and even stealing accounts belonging to website users (Gan et al., 2021; Gupta & Chaudhary, n.d.; Rodríguez et al., 2020). XSS cases identified on the eBay site can even make attackers enter a malicious code in the item description field in the item auction feature (Mahmoud et al., 2017).

In addition, XSS attacks aimed at government sites can allow attackers to perform the ATO (Account Takeover) method, so this attack has a large enough potential to disrupt national security stability. Based on these problems, both website developers and web managers must be able to detect, anticipate, and mitigate XSS attacks. Actually, there are several methods, techniques, or tools to carry out the detection and mitigation (Mahmoud et al., 2017). However, in practice, XSS attacks are quite complex and can be carried out through a variety of methods, techniques, and applications. A single security defense system or mechanism (single security layer) has proven to be unable to overcome the massive XSS attacks on the internet (Akbar & Ridha, n.d.; Mahmoud et al., 2017). Therefore, the purpose of this research is to test the implementation of MLSM (Multi-Layer Security Mechanism) to improve the detection and mitigation of websites from XSS attacks. In this mechanism, every layer of security in MLSM can synergize with each other and cover every security gap, so that the website can be more secure.

2. PREVIOUS RESEARCH

There are several studies which have discussed or tested some various methods, applications, or techniques to detect and overcome XSS attacks. Research (Akbar & Ridha, n.d.) analyzed web security from XSS attacks using the OWASP Web Application Firewall Mod Security (OWASP ModSecurity) module. OWASP ModSecurity is used to block attempted or attempted XSS attacks on websites that are on the same web server. Research (Wijayarathna & Gamagedara Arachchilage, 2019) tested the implementation of OWASP ESAPI Output Encoding to perform XSS attack filtering and measure its success rate. In addition, to overcome XSS attacks, research (Yulianingsih, 2017) applies the Meta Character method and research (Putra et al., 2020) uses the Advanced Encryption Standard (AES) algorithm. Each of these studies certainly has different results. However, because the method, application, or technique used is a single security mechanism, the security mechanism is still not effective. The security mechanism only addresses (patching) one side of the loophole or security issue and is not comprehensive. The previous research can overcome reflected XSS, but cannot overcome XSS attacks of stored XSS type. Therefore, MLSM is implemented to address every type of XSS security vulnerability.

In additio, in order to dealing with XSS attacks, there are several studies conducted to detect or measure the level of vulnerability of XSS security vulnerabilities. Research (Ahad et al., 2018; Kurniawan & Setianto, 2020; Sari & Putra, 2017) conducted an analysis or

measured the level of vulnerability on the web from XSS attacks. Meanwhile, studies (Mahmoud et al., 2017; Salas & Martins, 2014; Udayana University, Bali, Indonesia et al., 2019) made a comparison of several single security mechanisms to overcome and detect XSS attacks. Based on a review of mitigation efforts, research (Mahmoud et al., 2017; Mateo Tudela et al., 2020; Rodríguez et al., 2020; Wibowo & Sulaksono, 2021) discusses how to mitigate efforts for web creators or managers from XSS attacks. In addition, the implementation of layered security mechanisms has been carried out by research (Mokbal et al., 2019) using the perception algorithm. This method utilizes an artificial neural network model to detect XSS attacks based on training data. However, research(Mokbal et al., 2019) was carried out for the purpose of testing or measuring the success rate of detection, not for the purpose of addressing and mitigating.

3. CROSS SITE SCRIPTING

Cross-Site Scripting (XSS) is an attack which occurs on the client-side through the injection of Javascript-based code into web pages. Most XSS attacks will be carried out via HTML forms. Starting from input in the form of text input, numbers, text areas, and so on (Mokbal et al., 2019). XXS attacks can be carried out because the input from the client is not properly validated. XSS attacks are divided into 3 types, namely (1) Stored XSS; (2) Reflected XSS; and (3) DOM Based XSS. By implementation, most cases of XSS attacks will be executed through five intermediaries, namely the search engine that reflects the search keywords entered, the error message that reflects the string containing the error, the form that can be filled out and submitted by the user, the form stored in the database, and the dashboard. web-based messaging or chat pages that allow users to send messages through the system (Joshi, n.d.; Mahmoud et al., 2017). These attack techniques will certainly continue to develop along with technological developments that occur (Shrivastava et al., 2016; *XSS Attacks Cross Site Scripting Exploits and Defense*, n.d.).

3.1. *Stored XSS*

A stored XSS attack occurs when an attacker enters data or form fields in the form of javascript code into the database storage, without going through the validation process or filtering the fields. The data is directly sent and stored in the database. As a result, when the user displays the database through the browser, the XSS code is then executed by the browser. An overview of the stored XSS code is below.

```
# HTML code <textarea> comments

<textarea id="comments " name="comments
"></textarea>
# XSS code injected into comment <textarea> form
saved to database

# XSS code is executed when the user opens comment
<textarea id="comments" name="comments”>
   <script>/* Embeded XSS Code*/</script>
</textarea>
```

3.2. *Reflected XSS*

Technically, a reflected XSS attack is similar to a stored XSS attack, in that it is submitted via a form and executed when the browser accesses an infected page. However, in this attack, the website does not store the XSS code in the database. The data is directly reflected or displayed on the infected web page. Therefore, in this type of attack, most XSS code will be embedded via URL

```
# HTML Code <input>
<input type="text" name="q" id="keyword" />

# XSS Code is injected to URL
https://<web-target-
address>.com/index.php?q=<injection-location>
```

3.3. *DOM Based XSS*

DOM Based XXS is mostly devoted to Javascript-based web applications or those that use Javascript to process and display data entered by the user (JS Rendered Page). As with previous types of XSS attacks, the data is not filtered and validated, so attackers can insert XSS code into the target web. Attackers usually take advantage of the browser console to carry out this attack. The following is an example of the code used by Dom Based XSS.

```
# HTML Code <input>

<input type="text" name="q" id="keyword" />

# XSS code is injected into DOM
var keyword =
document.getElementById(keyword).value;
var result  = document.getElementById('result');
result.innerHTML = 'You search for ' + <injection-
location>;
```

3.4. Mekanisme *Multi Layer Security*

The security mechanism implemented in this research consists of five layers, namely OWASP ModSecurity, HTTP Middleware, Templating Engine, Data Sanitizer, and Framework or CMS Default Security Features. All components are integrated to build a solid security wall that can withstand various attacks, especially XSS attacks. Each component works on its own scope and forms a firewall layer that protects each other. This mechanism also ensures that attacks can be detected, resolved, and mitigated effectively. Therefore, this security mechanism does not depend on the framework, CMS, or technology used by the website application..

3.4.1. *OWASP ModSecurity*

OWASP ModSecurity (OWASP Web Application Firewall Mod Security) is an Apache web server application or module that runs as a service and acts as a firewall. This firewall works by scanning every HTTP traffic that goes through the web server (Alamsyah, n.d.). Based on the traffic of packet, OWASP ModSecurity accepts or drops requests depending on the configured rules. The rules, in this case, are called as core rule set (Akbar & Ridha, n.d.).

3.4.2. *HTTP Middleware*

HTTP Middleware is a web application component that sits one level above routing. The main task of this layer is to managing web traffic. In this case, request and response will be managed in this type of layer (Varghese, 2015). Through this HTTP Middleware layer, web developers could implement algorithms wich check whether requests and responses contain XSS.

3.4.3. *Templating Engine*

The main task of A layout engine or templating engine is an application component that is to render application layouts or themes. If the templating engine is properly configured, XSS code can be detected, rejected, or converted into safe HTML format before being displayed to web visitors..

3.4.4. *Data Sanitizer*

In many modern web applications, this security layer has become standard function, so it is usually provided by the programming language or web framework. For example, Wordpress uses the function name esc_html(html) or PHP uses htmlspecialchars(html) to filter out HTML code that contains XSS codes. This function can be used to filter the form fields or web content before they are displayed.

3.4.5. *Framework/CMS Security Features*

The security components at this layer are in the form of applications packaged in the form of a web framework or CMS. Each web framework and CMS has its own characteristics and security features (Kaluza et al., 2019). However, in this layered security

mechanism, the web framework or CMS have to meet several fundamental minimum requirements: (1) having HTTP middleware, hook, or the similar features; (2) having data sanitizer functions which could be activated at the template and database levels; and (3) having template engine which has an escape or filter.

4. METHODS

4.1.1. *Mechanism and Stages*

This study applies a layered security mechanism to solve XSS attacks. As shown in figure 1, the attacker must knock out five layers of defensive walls to carry out an attack. Each layer has its own role and scope of work. With layered security mechanisms, XSS attacks become very difficult to carry out. Layer 1 protects all websites that are on the same web server. Layer 2 ensures that every request and response traffic does not contain XSS code. In the last layer, layers 3, 4, and 5 eliminate XSS code execution on the webview.

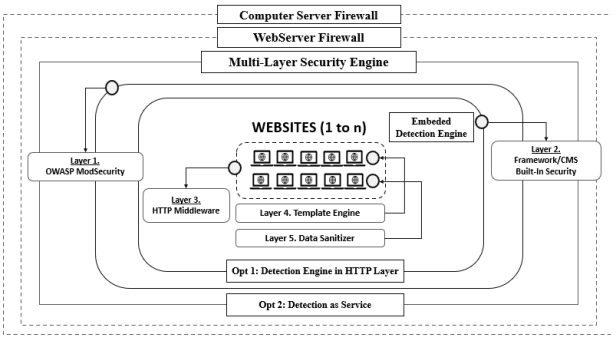


Figure 1. MLSM Implementation to Protect Entire Websites on the Web Server

This study applies a layered security mechanism on a website based on the Django framework. Testing or implementation of security is carried out in two stages. The level of web security in both stages was tested by two different attacking applications. In the first stage, XSS attacks are carried out using two attacker applications (Zap and Arachni) on websites that have not implemented MLSM. Then, in the second stage, another XSS attack is carried out using two applications (Zap and Arachni) and the same target website. However, in the second stage, the researcher activated the MLSM security.

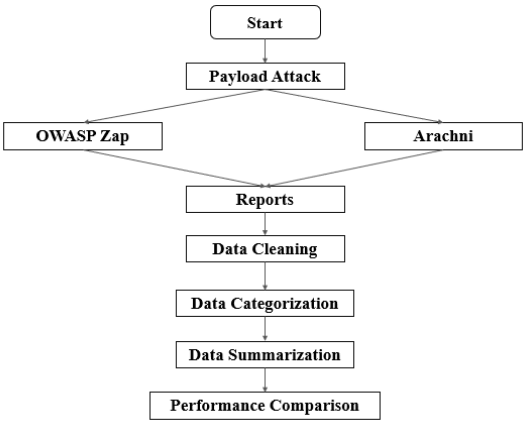


Figure 2. Stages of Application Security Tests that Have Not and Have Implemented MLSM

The results of these two stages are automatically recorded in the form of a report by each attacker application. Based on the results of the attack report, the researcher then took three actions, namely (1) data cleaning to separate the results of XSS and non XSS attacks; (2) categorizing the XSS vulnerabilities found by type; (3) recapitulate the findings of XSS security vulnerabilities; and (4) to compare the effectiveness of the findings previous studies.

4.1.2. XSS Vulnerable Attacker

The tools used to test XSS vulnerabilities are Arachni and OWASP Zap. Acachni is a feature-rich and modular Ruby-based application that is often used by penetration testers (pentesters) to evaluate the security of web applications (Institute of Engineering & Management, Y-12 Saltlake Electronics Complex, Sector V, Kolkata -91 INDIA et al., 2014). Arachni version used is 2.0. OWASP ZAP is an application that is used to scan website security holes comprehensively and holistically, because it is able to detect security holes, even those on

pages that require authentication (Prasad, 2016). In this study, researchers used ZAP version 2.10.0.

4.1.3. Method for Detecting Attacks

When an XSS attack is sent, the security layer performs 2 main tasks, namely: detecting and responding to attacks.

a) OWASP ModSecurity Layer

OWASP ModSecurity is a service that runs in conjunction with a web server. This ModSecurity detects and overcomes attacks based on predetermined rules. In this study, researchers used the default XSS rule to overcome XSS. The full version of the rule can be accessed on the page owasp-modsecurity-crs.

b) HTTP Middleware Layer

At the HTTP Middleware layer, researchers use middleware code that aims to detect XSS attacks, then direct the request from the attack to a 404 page or not found.

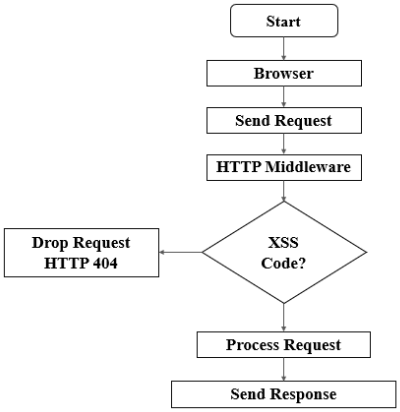


Figure 4. XSS Attack Detection and Blocking Flow by HTTP Middleware

c) Templating Engine Layer

The task of this layer is to protect the website from stored XSS attacks. This attack is executed when database data is displayed on a web page without going through a validation process. Through the implementation of the templating engine, XSS code cannot be executed.

d) Data Sanitizer Layer

This layer is to filter data before it is stored in the database and before it is displayed on web pages. Each web framework or programming language has its own function and syntax to filter data. To apply the sanitizer data, the researchers used Django-Bleach.

e) Framework/CMS Layer

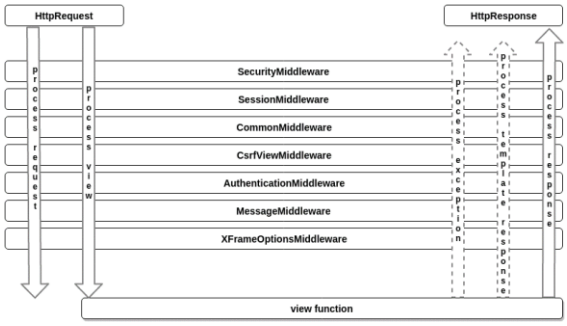


Figure 4. Django Default Security Based on HTTP Middleware

At the CMS or Framework layer, Django has its own security features. The security vulnerabilities that are open in Django will mostly occur on the application side that is developed separately by web developers.

4.1.4. Web Framework for Testing

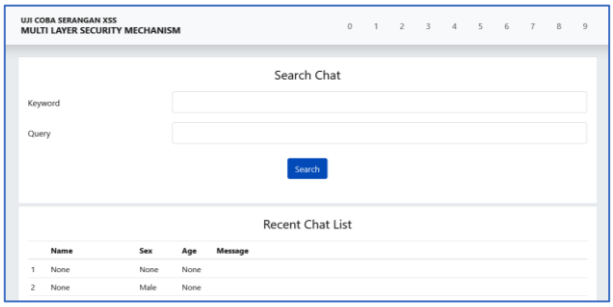


Figure 5. Vulnerable Chat Application as a Test Target Web (Has 170 XSS Vulnerabilities)

The website used to conduct this research is a web based on the Django framework which is activated on the Apache2 webserver. Django was chosen for three reasons, namely: (1) it has HTTP Middleware security features that can be configured quickly and easily; (2) regular expression-

based routing configuration, thus simplifying URL filtering; and (3) has a built-in template engine that can apply XSS code-indicated content filtering.

Before the process of security testing, the Django-based web framework was filled with fake chat applications. Researchers have disabled the security features of HTTP Middleware and Data Sanitizer (model and templating engine). The following are the details of the configuration of the test website used before using the layered security mechanism.

HTTP Middleware	:	Off
Template Engine	:	Auto Escape Off
Model Sanitizer	:	False
Vulnerabilities	:	170

5. RESULT AND DISCUSSION

After testing to the target web running on the Apache 2.4 web server on the Linux Ubuntu 20.04.2 (WSL Version) operating system, various XSS attack results or reports were found from Arachni and ZAP. This happens because each attacker application has a different character in imple-menting the attack algorithm. The following are the results of the MLSM implementation of testing.

5.1. Arachni and ZAP Attacks on Non-MLSM Websites (Unimplemented)

A. Arachni Attack

After testing or simulating attacks against web targets using Arachni, there are 168 of 170 (98.82%) security holes that Arachni can attack. Based on the analysis of the attack reports, the attack process by Arachni took 2 hours 51 minutes.

Version	2.0dev
Seed	e73401431dab818a74789981b6863a64
Audit started on	2021-07-19 13:31:59 +0700
Audit finished on	2021-07-19 16:23:16 +0700
Runtime	02:51:17

Based on the type of attack, the details of the XSS attacks that were successfully carried out were 124 attacks in the form of DOM based XSS, 22 stored XSS, and 22 reflected XSS. The following is a breakdown of the XSS vulnerabilities found by Arachni..

Tabel 2. XSS Security Vulnerabilities Found by Arachni

	Type of Attack	Total
1	Dom Based XSS	124
2	Stored XSS	22
3	Reflected XSS	22
	Total	**168**

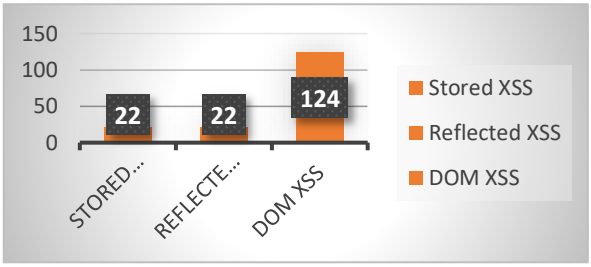


Figure 7. Percentage of XSS Security Vulnerabilities Found

B. ZAP Attack

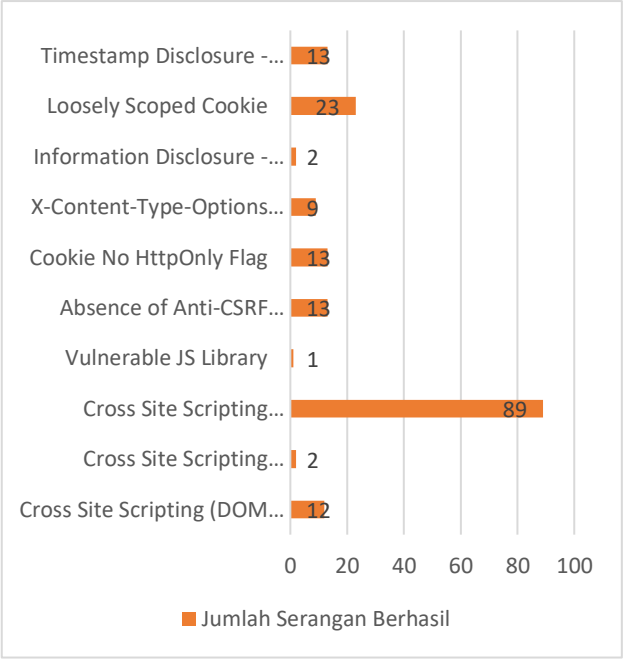


Figure 8. Percentage of XSS Security Vulnerabilities Found

Based on the results of the ZAP attack, there are 177 attack that can be done. From the 177 vulnerabilities, the number of XSS vulnerabilities that were successfully attacked was 103 out of 170 provided (60.58%). Based on the type of XSS, the XSS vulnerabilities consist of 12 XSS-based DOM attacks and 2 stored XSS, and 89 reflected XSS. In this case, the XSS security vulnerability is at a HIGH risk level which means that the vulnerability is dangerous for the target web.

Table 2. ZAP Attackable XSS Security Vulnerabilities

XSS Type	Risk Level	Attack
DOM Based XSS	High	12
Stored XSS	High	2
Reflected XSS	High	89
Jumlah Celah Keamanan XSS		**103**

Unlike other security vulnerabilities, XSS vulnerabilities that can be attacked by ZAP are cla-ssified as security vulnerabilities with a high level of risk. The following are the differences and recapitulation of the level of risk of security vulnerabilities found by ZAP.

Table 3. ZAP Attackable XSS Security Vulnerabilities (Complete View)

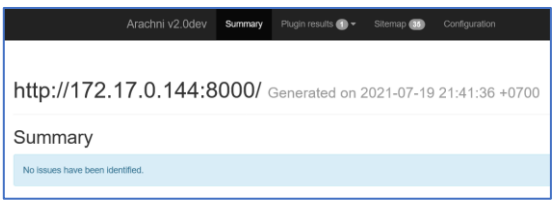
XSS Type	Risk Level	Attack
Cross Site Scripting (DOM Based)	**High**	**12**
Cross Site Scripting (Persistent)	**High**	**2**
Cross Site Scripting (Reflected)	**High**	**89**
Vulnerable JS Library	Medium	1
Absence of Anti-CSRF Tokens	Low	13
Cookie No HttpOnly Flag	Low	13
X-Content-Type-Options Header Missing	Low	9
Information Disclosure - Suspicious Comments	Informational	2
Loosely Scoped Cookie	Informational	23
Timestamp Disclosure - Unix	Informational	13

5.2. Arachni and ZAP Attacks on Non-MLSM Websites (Implemented MLSM)

After carrying out an attack on a website that does not implement MLSM, the attack is then carried out on a website that has implemented MLSM. This is to see how effective the MLSM implementation is in detecting and overcoming XSS attacks. In this attack trial, the researchers used the same attacker's website and application. In addition, before conducting the experiment, the researcher resets the database so that traces of the previous attack are lost and do not affect the second stage of the attack.

A. Arachni Attack

The attack time spent when website has activated MLSM is 1 hour 24 seconds. This time is shorter than non-MLSM website attacks, which take 2 hours 51 minutes 17 seconds. This happens because MLSM has blocked access to attacks and closed security holes on the website. The following is a screenshot of the attack report showing that the XSS attack cannot be carried out.



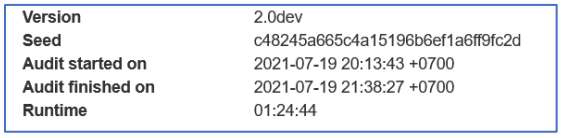


Figure 9. XSS Arachni Attack on MLSM Website Cannot Be Executed

B. ZAP Attack

The same as Arachni's attack. ZAP is also not capable of XSS attacks. None of the XSS attacks were successfully carried out on websites that enabled the MLSM feature. ZAP only managed to perform attacks on security vulnerabilities other than XSS. The recap. of ZAP attacks can be seen in Figure 10 and table 4.

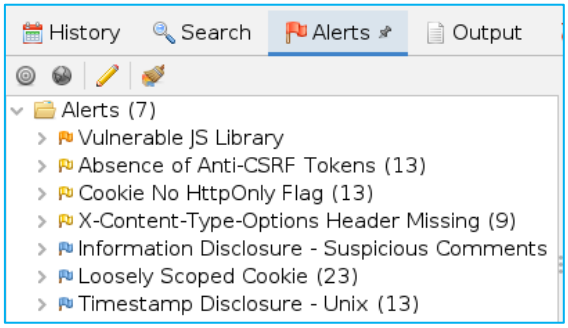


Figure 9. ZAP Attack Result Screenshot (XSS not Found)

In addition to figure 9 above, a recapitulation of security vulnerabilities that can be attacked by ZAP can be seen in table 4. According to the data in the table, ZAP cannot perform XSS attacks on websites that have implemented MLSM.

Table 4. XSS Security Vulnerabilities Arachni Can Attack

Attack Name	Risk Level	Attack Total
Vulnerable JS Library	Medium	1
Absence of Anti-CSRF Tokens	Low	13
Cookie No HttpOnly Flag	Low	13
X-Content-Type-Options Header Missing	Low	9
Information Disclosure - Suspicious Comments	Informational	2
Loosely Scoped Cookie	Informational	23
Timestamp Disclosure - Unix	Informational	13

C. Attack Result Comparison

After testing Arachni and ZAP attacks on websites that have activated the MLSM feature, MLSM is proven to be able to detect and overcome 100% of XSS attacks. There is no single type of XSS attack that can attack websites. To get higher confidence, researchers have conducted 2 experiments and the results are still the same, namely Arachni and ZAP cannot attack websites that have implemented the MLSM feature..

Table 4. Comparison of MLSM and Non MLSM Web Attack Results

	XSS Attack Type	The Number of Attacks Can Do After and Before MLSM Implementation			
		Arachni		ZAP	
		Unimplemented	Implemented	Unimplemented	Implemented
1	Reflected XSS	124	0	12	0
2	Stored XSS	22	0	2	0
3	Dom Based XSS	22	0	89	0
TOTAL		168	0	103	0

5.3. Comparison of the Effectiveness of MPLS Implementation with Previous Research

As previously stated, there are studies that have tested methods to detect and mitigate XSS attacks. However, the security

mechanisms used in previous studies have not been able to overcome XSS attacks holistically and comprehensively. The method used in previous research is still not able to detect and deal with XSS attacks properly. The following is a comparison of the implementation of MLSM with similar studies also measure the effectiveness of the method to overcome XSS attacks.

5.3.1. MLSM and OWASP ModSecurity

OWASP *Web Application Firewall* had been individually applied on research (Akbar & Ridha, n.d.). To test the security level of XSS, researchers used 2 applications, namely BeEF Explo-itation and XSSer Exploitation. Based on the trials that have been carried out, OWASP ModSecurity Firewall is able to overcome reflected XSS attacks, but fails to overcome stored XSS attacks. In contrast to this research, MLSM is able to overcome the attacks of three types of XSS attacks, namely stored, reflected, and DOM based XSS..

5.3.2. MLSM and Metode Metacharacter

This research implement application protection using the Meta-Character Method. In this study, researchers only conducted a trial of reflected XSS which was sent through the search box. In addition, the trial website used is also too simple so that it does not represent the actual website. The researcher also did not use the attacker application, only sent 2 XSS queries in the search box. Meanwhile, MLSM is able to detect and overcome 3 types of XSS and use a more complex and representative test web(Yulianingsih, 2017).

5.3.3. MLSM and Metode Advanced Encryption Standard (AES)

Research implemented the AES algorithm to overcome the XSS vulnerability. To test the website's resilience from XSS attacks, the study used one attacker application, namely Acunetix. Based on the test results, there were only 2 XSS attacks found by Acunetix and resolved by the AES method, namely stored XSS. The method offered in this study cannot detect DOM-based and reflected attacks. Meanwhile, MLSM used 2 attacker applications and was able to find 100 XSS vulnerabilities (Putra et al., 2020).

5.3.4. MLSM and Native PHP Function

Based on the results of applying the proposed method in research (Kurniawan & Setianto, 2020), there are 8 out of 21 XSS security vulnerabilities that cannot be handled by Native PHP Function. The study only conducted a partial trial and used a simple form so that it did not represent the actual web. Meanwhile, MLSM is able to handle 100% of the XSS vulnerabilities committed by two attacking applications.

To strengthen the application of the proposed method, this study calculates the ratio of the types of XSS that can be handled and the XSS security vulnerabilities provided. In addition, researchers also calculated the level of effectiveness between methods. The formula used is as follows..

Ratio Calculation Formula By Attack Type XXS

$$RJS = \frac{\sum JST}{JSG}$$

Information:
RJS = Attack Type Ratio
ANN = Types of Identified Security Vulnerabilities
JSG = Highest Vulnerability

Ratio Calculation Formula Based on Successfully Handled Gaps

$$RCK = \frac{(CK0 - CK1)}{CKG}$$

Information:
RCK = Vulnerability Ratio
CK0 = Unimplemented Vulnerability
CK0 = Implemented Vulnerability
CKG = Highest Vulnerability Handled Gap

Table 5. Comparison of the Effectiveness of MLSM with Other Methods

	Method Used	Type of XSS			RJS	XSS Available	Identified Vulnerable		RCK	Effectivity Level (RJS x RCK)
		R	S	D			Un	Im		
1	MLSM	1	1	1	1	170	168	0	1	1

2	OWASP Fiewall (Akbar & Ridha, n.d.)	1	0	0	0,3	3	3	1	0,011	0,0035
4	Metacharacter (Yulianingsih, 2017)	1	0	0	0,3	1	1	0	0,005	0,0017
5	AES (Putra et al., 2020)	0	1	0	0,3	2	2	0	0,011	0,0035
6	Native PHP (Kurniawan & Setianto, 2020)	1	0	0	0,3	1	1	0	0,005	0,0017

***) If more than 1 attacker application, the largest number of post-implementation loopholes is taken**

6. Conclusion

Based on the testing and implementation carried out by this research, MLSM is proven to be able to overcome XSS attacks from two attacking applications, namely Arachni and ZAP. MLSM is also able to overcome three types of XSS attacks, namely stored, reflected, and DOM based XSS. This study also proves that the application of a single security mechanism, as has been applied to research (Akbar & Ridha, n.d.), (Yulianingsih, 2017), (Putra et al., 2020), and (Kurniawan & Setianto, 2020), are ineffective against all three types of XSS attacks. According to table 5, MLSM proved to be more effective than previously method.

The implementation of MLSM has been proven to be able to detect and mitigate 100% of XSS security vulnerabilities, starting from stored, reflected, and DOM based XSS. This proof has been carried out comprehensively and holistically by two applications by activating the attack mode. This test has been carried out two times and got the same results, namely the implementation of MLSM successfully overcomes and mitigates 100% of XSS security vulnerabilities. Therefore, MLSM is an effective method for dealing with XSS attacks.

REFERENCES

Ahad, D. S., Akbar, M., & Ulfa, M. (2018). ANASLISIS KERENTANAN TERHADAP ANCAMAN SERANGAN PADA WEBSITE PDAM TIRTA MUSI PALEMBANG. *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION*, *2*(4), 10.

Akbar, M., & Ridha, M. A. F. (n.d.). *SQL Injection and Cross Site Scripting Prevention Using OWASP Web Application Firewall*. 7.

Alamsyah, A. (n.d.). *ANALISA KEAMANAN INFORMASI PADA APLIKASI BERBASIS WEB MENGGUNAKAN TEKNIK WEB APPLICATION FIREWALL MODSECURITY*. 12.

Buchanan, B. (n.d.). *The Cybersecurity DIlemma: Hacking, Trust, and Fear Between Nations*. 300.

Gan, J.-M., Ling, H.-Y., & Leau, Y.-B. (2021). A Review on Detection of Cross-Site Scripting Attacks (XSS) in Web Security. In M. Anbar, N. Abdullah, & S. Manickam (Eds.), *Advances in Cyber Security* (pp. 685–709). Springer. https://doi.org/10.1007/978-981-33-6835-4_45

Giri, S. (2019). *Cyber Crime, Cyber threat, Cyber Security Strategies and Cyber Law in Nepal. 9*(2249), 11.

Gupta, B. B., & Chaudhary, P. (n.d.). *Cross-Site Scripting Attacks*. 171.

Institute of Engineering & Management, Y-12 Saltlake Electronics Complex, Sector V, Kolkata -91 INDIA, Mukhopadhyay, I., Goswami, S., & Mandal, E. (2014). Web Penetration

Testing using Nessus and Metasploit Tool. *IOSR Journal of Computer Engineering*, *16*(3), 126–129. https://doi.org/10.9790/0661-1634126129

Joshi, S. (n.d.). *SQL Injection Attack and Defense*. 24.

Kaluza, M., Kalanj, M., & Vukelić, B. (2019). A Comparison of Back-end Framework for Web Application Development. *Zbornik Veleučilišta u Rijeci*, *7*(1), 317–332. https://doi.org/10.31784/zvr.7.1.10

Kurniawan, M. F., & Setianto, W. (2020). *Optimasi Metode Otomatisasi Penghilangan Kerentanan Terhadap Serangan XSS Pada Aplikasi Web*. *2*, 8.

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. *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 36–42. https://doi.org/10.1109/INTELCIS.2017.8260024

Marchand-Melsom, A., & Nguyen Mai, D. B. (2020). Automatic repair of OWASP Top 10 security vulnerabilities: A survey. *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 23–30. https://doi.org/10.1145/3387940.3392200

Mateo Tudela, F., Bermejo Higuera, J.-R., Bermejo Higuera, J., Sicilia Montalvo, J.-A., & Argyros, M. I. (2020). On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications. *Applied Sciences*, *10*(24), 9119. https://doi.org/10.3390/app10249119

Mokbal, F. M. M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F., & Xiaoxi, W. (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

OWASP Top-10 2021, statistically calculated proposal. (n.d.). *Wallarm Blog*. Retrieved June 19, 2021, from https://lab.wallarm.com/owasp-top-10-2021-proposal-based-on-a-statistical-data/

Prasad, P. (2016). *Mastering modern web penetration testing: Master the art of conducting modern pen testing attacks and techniques on your web application before the hacker does!* http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1409187

Putra, Y., Yunus, Y., & Sumijan, S. (2020). Meningkatkan Keamanan Web Menggunakan Algoritma Advanced Encryption Standard (AES) Terhadap Seragan Cross Site Scripting. *Jurnal Sistim Informasi dan Teknologi*. https://doi.org/10.37034/jsisfotek.v3i2.110

Rekap Serangan Siber (Januari – April 2020) | bssn.go.id. (n.d.). Retrieved July 19, 2021, from https://bssn.go.id/rekap-serangan-siber-januari-april-2020/

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

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

Sari, W. P., & Putra, I. N. A. P. (2017). Analisis Serangan Hacker Menggunakan Honeypot High Interaction. *Jurnal Tiarsie*, *14*(1), 13–18.

Serangan Siber Ancam Indonesia—Infografik Katadata.co.id. (2019, March 25). https://katadata.co.id/ariayudhistira/infografik/5e9a5513db5c6/serangan-siber-ancam-indonesia

Shrivastava, A., Choudhary, S., & Kumar, A. (2016). XSS vulnerability assessment and prevention in web application. *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, 850–853. https://doi.org/10.1109/NGCT.2016.7877529

Søhoel, H. M. (n.d.). *OWASP top ten—What is the state of practice among start-ups?* 94.

Udayana University, Bali, Indonesia, Wiradarma, A. A. B. A., & Sasmita, G. M. A. (2019). IT Risk Management Based on ISO 31000 and OWASP Framework using OSINT at the Information Gathering Stage (Case Study: X Company). *International Journal of Computer Network and Information Security*, *11*(12), 17–29. https://doi.org/10.5815/ijcnis.2019.12.03

Varghese, S. (2015). HTTP Middleware. In S. Varghese (Ed.), *Web Development with Go: Building Scalable Web Apps and RESTful Services* (pp. 99–120). Apress. https://doi.org/10.1007/978-1-4842-1052-9_6

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. https://doi.org/10.24002/ijis.v3i2.4192

Wijayarathna, C., & Gamagedara Arachchilage, N. (2019). *Fighting Against XSS Attacks. A Usability Evaluation of OWASP ESAPI Output Encoding*. Hawaii International Conference on System Sciences. https://doi.org/10.24251/HICSS.2019.877

XSS Attacks Cross Site Scripting Exploits and Defense. (n.d.). 482.

Yulianingsih, Y. (2017). Melindungi Aplikasi dari Serangan Cross Site Scripting dengan Metode Metacharacter. *Jurnal Nasional Teknologi dan Sistem Informasi*, *3*(1), Article 1. https://doi.org/10.25077/TEKNOSI.v3i1.2017.83-88