Mobile and Forensics
Vol. 5, No. 1, March 2023, pp. 44-58
ISSN: 2714-6685 (online), 2656-6257 (print)
DOI: http://dx.doi.org/10.12928/mf.v5i1.8914

Impact Analysis of Web Application Firewall on Website-Based Application Security (Case Study PPDB Kak Seto School Website)

Krisna Dewa Pratama[1]* , Nuril Anwar[2]
[1]Universitas Ahmad Dahlan, Daerah Istimewa Yogyakarta, Indonesia
[1]krisnadp23@gmail.com, [2]nuril.anwar@tif.uad.ac.id
*Correspondence email

Abstract

The swift advancement of web-based applications has posed security challenges. Insufficient security awareness among web developers has resulted in a surge of cybercrime incidents due to website vulnerabilities. To counter this, implementing a Web Application Firewall (WAF) is proposed for the vulnerable PPDB Kak Seto School website, aiming to mitigate threats in the public network. The WAF acts as a defense against potential cyber breaches. Employing an experimental approach, this research encompasses identification, observation, literature review, analysis of WAF system requirements, implementation, testing, and pre/post-implementation analysis using ModSecurity as the security system. The study analyzes the impact of WAF adoption and provides recommendations for enhancing security. Findings demonstrate WAF's effectiveness in fortifying the Kak Seto School web application by efficiently identifying and blocking potential attacks, thereby reducing breach success rates. Post-WAF implementation, Pingdom tests show a slight drop in Performance Grade (70 to 69) and a minor increase in Load Time (2.76 to 3.23 seconds). GTmetrix tests reveal a Grade downgrade from B to C and an increase in Largest Contentful Paint time (2.2 to 2.7 seconds). In conclusion, despite minor performance effects, WAF significantly enhances security, as evident in improved loading times during tests.

Keywords: Web Application Firewall, ModSecurity, Website Security, Web Attacks, Website.

INTRODUCTION

In the current era, the progress of web-based applications is rapidly advancing[1]. However, a lack of awareness among web developers regarding website security has triggered the emergence of 'Cyber Crime' incidents[2]. The primary cyber threats in the digital world originate from vulnerabilities inherent in web-based applications[3]. Neglecting the implementation of Penetration Testing during the pre-publication phase has resulted in numerous breaches on vulnerable websites, posing threats to the confidentiality, integrity, and availability of stored data[4]. The dissemination of sensitive information to individuals or the public can lead to substantial losses for users and website owners, given the valuable nature of such information[5]. Consequently, a robust security system is crucial to safeguard data on web servers and prevent malicious attacks[6].

To overcome this challenge, the effective solution of Web Application Firewall (WAF) is introduced to protect web applications from such threats. WAF functions as a barrier between users and web application servers, analyzing and monitoring incoming and outgoing data traffic. WAF has the capability to effectively detect and block suspicious attacks[7]. In the OWASP Top Ten, the most common attacks used by hackers in 2017 and 2021 are illustrated. Cross-site Scripting (XSS) ranked seventh in the OWASP list in 2017, but by 2021, it had dropped to the third position. This study involves simulating attacks on the PPDB Sekolah Kak Seto website,

Article History: *Received December 9, 2022; Revised January 3, 20123 Accepted March 17, 2023*

specifically Cross-site Scripting (XSS) and Security Misconfiguration, to evaluate the effectiveness of Web Application Firewall in handling these attacks. According to OWASP Top Ten, Cross-Site Scripting (XSS) holds the third spot in the Injection category, and Security Misconfiguration ranks fifth out of ten[8].
Cross-Site Scripting, commonly known as XSS, involves injecting malicious code into vulnerable web applications through unvalidated or improperly encoded input and output[7]. Therefore, effective preventive measures are crucial to protect web-based applications from such attacks[9]. On the other hand, Security Misconfiguration occurs due to improper security settings in computer systems or applications, creating vulnerabilities that can be exploited by malicious actors. This can be attributed to weak configurations, poor default passwords, inappropriate access permissions, or other configuration errors. Adhering to best security practices and conducting regular security scans are vital to mitigate the risks associated with Security Misconfiguration[10].
The PPDB Sekolah Kak Seto website is built using the Codeigniter 2 framework with PHP 7.4, making it susceptible to Cross-site Scripting (XSS) and Security Misconfiguration bugs if input filtering measures are not adopted by developers[11]. The website has not undergone Penetration Testing, which aims to assess the security level of the site[12]. The absence of effective security measures against Cross-site Scripting (XSS) and Security Misconfiguration vulnerabilities remains a concern[13]. The website employs the Apache web server. In the context outlined above, this study aims to analyze, implement, and assess the impact of using Web Application Firewall on the security of the PPDB Sekolah Kak Seto web-based application. The Mod Security firewall will be used to measure its effectiveness in mitigating Cross-Site Scripting (XSS) and Security Misconfiguration attacks. This research contributes to enhancing web application security and understanding the efficacy of WAF in protecting against cyber threats.

METHODS

The research employs the Experimental Method, focusing on the impact of implementing a Web Application Firewall in web-based applications. Specifically, the study involves implementing a Web Application Firewall as a security system for the Kak Seto School website. The experimental results will be documented and analyzed to provide recommendations for effective firewall utilization in web application security. The analysis will lead to conclusions regarding the influence, benefits, and strengths of the implemented system. -In this research phase, there are several stages involved. Firstly, identifying the issue and observing the security system of the Kak Seto School web application. Next, conducting a literature review followed by an analysis of the security system requirements for Web Application Firewall implementation. Subsequently, implementing the WAF and performing pre and post-implementation WAF testing[14]. Finally, analyzing the results, as detailed in Fig. 1.

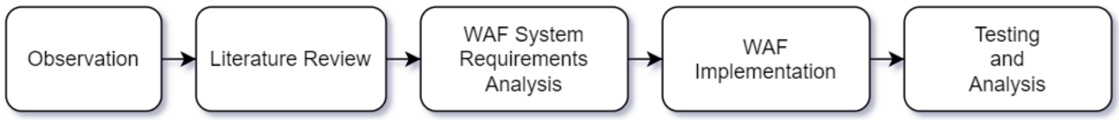


Fig. 1. The stages of the research method

SCENARIO

This research scenario is created to provide an overview of website security using a Web Application Firewall. The research scenario consists of Users, a Web Application Server or Web Server, and a Database as the targeted elements for potential attacks. The scenario depicted in Figure 2 will be conducted on a Web Server that hasn't yet implemented a Web Application Firewall on the Web Application Server, which will be used for attack testing. Requests from both users and potential attackers will be directly received by the server and treated as normal requests due to the absence of security measures on the web server[14]. The research scenario is illustrated in Figures 2.

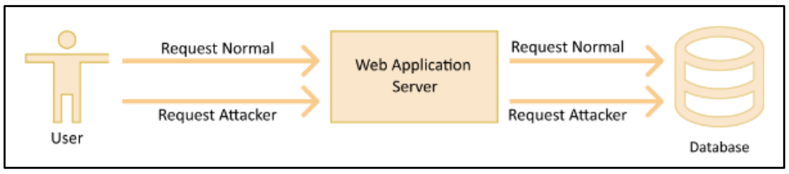


Fig. 2. Scenario without WAF on the Web Server

The second scenario involves Users/Attackers, a Web Application Firewall, Web Application Server or Web Server, and a Database as the targeted elements for potential attacks. In the scenario depicted in Figure 3.2, the Web Application Firewall is expected to filter out potential attacks or unauthorized inputs, allowing only legitimate requests to pass through to the Web server. The research scenario is illustrated in Figure 3.

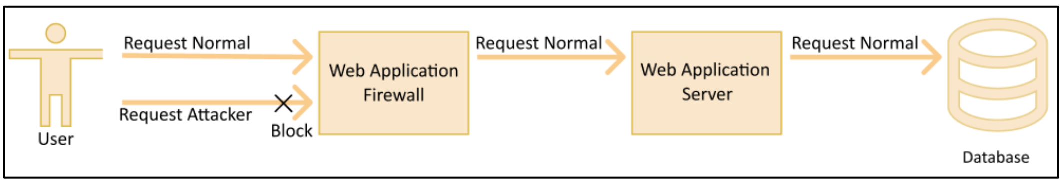


Fig. 2. Scenario with WAF on the Web Server

TOPOLOGY

In the research process outlined in the previous chapter, this step involves implementing the Web Application Firewall as a security measure for the website application to assess the impact of the Web Application Firewall on website application security. Hardware and software components are required to facilitate the implementation of the Web Application Firewall, as discussed earlier[15]. The research topology is illustrated in Figure 4.

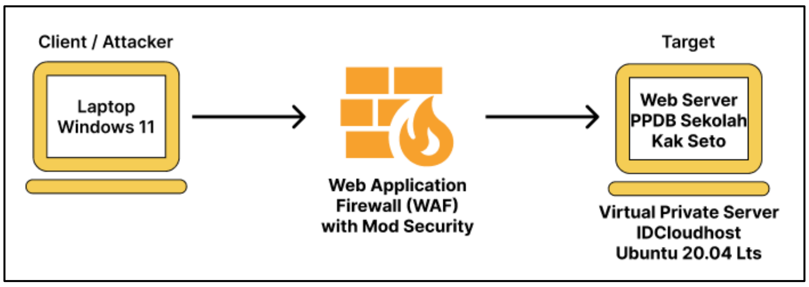

Client / Attacker
Laptop Windows 11
Web Application Firewall (WAF) with Mod Security
Target
Web Server PPDB Sekolah Kak Seto
Virtual Private Server IDCloudhost Ubuntu 20.04 Lts

Fig. 3. System Topology

OBSERVATION

The observation phase is conducted by creating a simulation using the web server of the Kak Seto School website. The security of the web application is implemented as protection against attacks by installing a connected Web Application Firewall (WAF) within the same network as the web server of the web application.

LITERATURE REVIEW

The stage involves studying books, e-books, journals related to computers, web security systems, web applications, and other relevant supporting materials. This process aims to aid the researcher in completing the study effectively[16].

WAF SYSTEM REQUIREMENTS ANALYSIS

Based on the observations and literature review conducted on the web server used for the Kak Seto School Admissions website, this research employs the following specifications for the hardware and software requirements of the:

A. Hardware Requirements

Table 1. Hardware Requirements

No	Hardware	Spesifikasi	
		Client	Server
1	CPU	Processor AMD Ryzen 7 4800H with Radeon Graphics, 2900 Mhz, 8 Core(s), 16 Logical Processor(s)	Intel (Broadwell, IBRS) (2) @ 2.099GHz
2	GPU	NVIDIA Geforce GTX 1650 Ti	00:02.0 Red Hat, Inc. QXL paravirtual graphic card
3	RAM	16 GB	2 GB
4	Storage	512 GB SSD	20 GB Hardisk
5	OS	Windows 11 Home Single Language 64-bit Version 10.0.22621 Build 22621	Ubuntu 20.04.6 LTS x86_64
6	Device	Laptop Lenovo Legion 5	*Virtual Private Server*(*VPS*) IDCloudHost

B. Software Requirements:

Table 2. Software Requirements

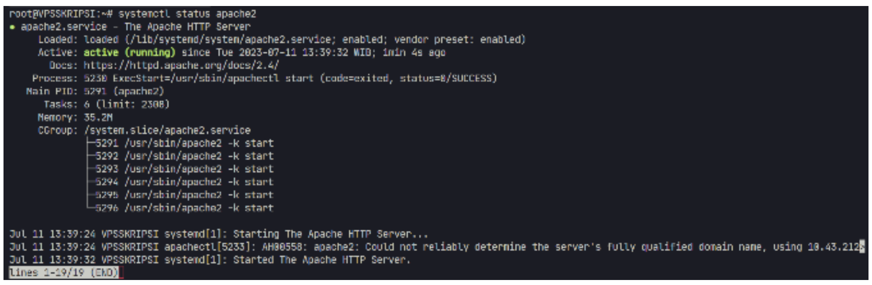
No	Software	Keterangan
1	Web Browser	Tools yang digunakan untuk membuka *Website*
2	*Mod Security*	Tools yang digunakan sebagai *WAF*
3	Visual Studio Code	Tools yang digunakan untuk remote *VPS*
4	Pingdom	Tools yang digunakan untuk menguji response time dari *Website*
5	GTmetrix	Tools yang digunakan untuk menguji response time dari *Website*

WEB SERVER IMPLEMENTATION

Implementing the web server is the initial step in establishing the infrastructure to run a website or web application within the context of this research. Web server implementation involves a series of steps to set up, install, and configure the server to be used. The following is the outcome of implementing the Web Server on the Virtual Private Server.

1. Apache Configuration

The Apache web server serves as the foundational platform for hosting the website in this research. Installed on the VPS web server, it is a critical element in the study's infrastructure. The active (running) status of the Apache service on the Virtual Private Server confirms the successful installation and operational functionality. This status indicates that the server is prepared to receive and process HTTP requests from users. The active Apache service allows easy and reliable access to the hosted website. This accomplishment marks a significant step toward running the web application smoothly and ensuring optimal server performance.



```
root@VPSSKRIPSI:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-07-11 13:39:32 WIB; 1min 4s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 5230 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 5291 (apache2)
      Tasks: 6 (limit: 2308)
     Memory: 35.2M
     CGroup: /system.slice/apache2.service
             ├─5291 /usr/sbin/apache2 -k start
             ├─5292 /usr/sbin/apache2 -k start
             ├─5293 /usr/sbin/apache2 -k start
             ├─5294 /usr/sbin/apache2 -k start
             ├─5295 /usr/sbin/apache2 -k start
             └─5296 /usr/sbin/apache2 -k start

Jul 11 13:39:24 VPSSKRIPSI systemd[1]: Starting The Apache HTTP Server...
Jul 11 13:39:24 VPSSKRIPSI apachectl[5233]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 10.43.212
Jul 11 13:39:32 VPSSKRIPSI systemd[1]: Started The Apache HTTP Server.
lines 1-19/19 (END)
```

Fig. 4. Apache Configuration

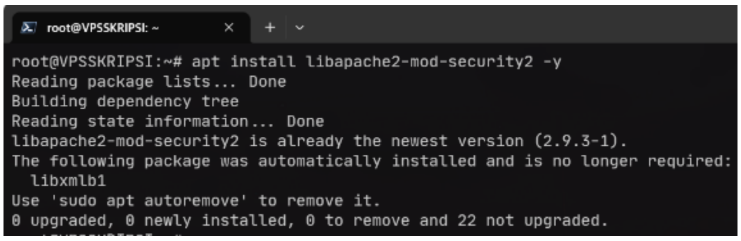
2. **PHP Configuration**
   The PHP installation process aims to facilitate the utilization of a programming language in the web application used on the Kak Seto School Admissions website. Therefore, the installation of PHP version 7.4 is necessary since the Kak Seto School Admissions website is built using Codeigniter 2.
3. **PHPMyAdmin Configuration**
   PHPMyAdmin is a web-based application utilized for managing and accessing MySQL databases. Through its intuitive graphical user interface, PHPMyAdmin simplifies tasks such as creating, modifying, and deleting databases, tables, columns, as well as managing users and access rights.
4. **PHPMyAdmin Configuration**
   The MySQL Server serves as the platform for storing the website data utilized in this research. Within the research context, the MySQL server will be installed on the Virtual Private Server web server. The installation of the MySQL server is achieved by executing the command "sudo apt install -y mysql-server mysql-client". During the installation process, you will be prompted to set a password for the root user. It's important to enter a memorable password during this step.

INSTALLATION AND CONFIGURATION OF THE WEB APPLICATION

This study will employ the Kak Seto School Admissions website as the testing object for web server security assessments, utilizing a Virtual Private Server (VPS) and the domain kriscode.my.id.

IMPLEMENTATION AND CONFIGURATION OF WAF

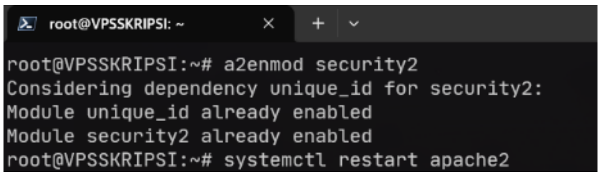
In this phase, the configuration of the Web Application Firewall (WAF) is conducted to ensure optimal security. The configuration includes setting security policies tailored to the needs of the Kak Seto School Admissions website, activating protective features such as defense against Cross-Site Scripting (XSS) and Security Misconfiguration (Directory Listing) attacks, as well as other common attacks. Additionally, security rules are adapted to match the characteristics and requirements of the Kak Seto School web application. The first step involves installing the Web Application Firewall (WAF) using the command "sudo apt install libapache2-mod-security2 -y," as shown in Figure 6.



```
root@VPSSKRIPSI:~# apt install libapache2-mod-security2 -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
libapache2-mod-security2 is already the newest version (2.9.3-1).
The following package was automatically installed and is no longer required:
  libxmlb1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 22 not upgraded.
```

Fig. 5. The installation process of Libapache2 Mod Security

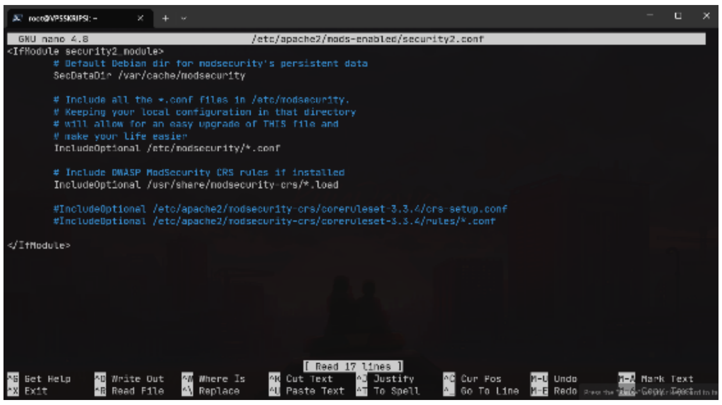
The second step after the installation is complete involves enabling the ModSecurity module using the command "a2enmod security2" and then restarting Apache with the following command: "systemctl restart apache2." The process of enabling Mod Security on the Apache Server is depicted in the Figure 7.



```
root@VPSSKRIPSI:~# a2enmod security2
Considering dependency unique_id for security2:
Module unique_id already enabled
Module security2 already enabled
root@VPSSKRIPSI:~# systemctl restart apache2
```

Fig. 6. The process of Enabling Mod Security

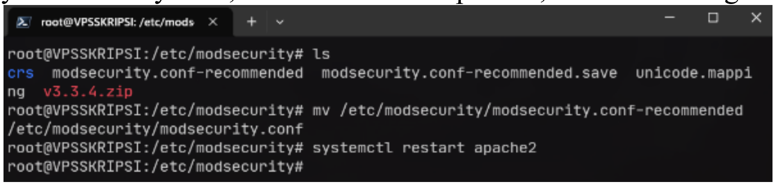
Next, modify the Mod Security configuration file using the command "nano /etc/apache2/mods-enabled/security2.conf." Then, uncomment the line "IncludeOptional /etc/modsecurity/*.conf" as shown in Figure 8.



```
  GNU nano 4.8                      /etc/apache2/mods-enabled/security2.conf
<IfModule security2_module>
        # Default Debian dir for modsecurity's persistent data
        SecDataDir /var/cache/modsecurity

        # Include all the *.conf files in /etc/modsecurity.
        # Keeping your local configuration in that directory
        # will allow for an easy upgrade of THIS file and
        # make your life easier
        IncludeOptional /etc/modsecurity/*.conf

        # Include OWASP ModSecurity CRS rules if installed
        IncludeOptional /usr/share/modsecurity-crs/*.load

        #IncludeOptional /etc/apache2/modsecurity-crs/coreruleset-3.3.4/crs-setup.conf
        #IncludeOptional /etc/apache2/modsecurity-crs/coreruleset-3.3.4/rules/*.conf

</IfModule>

                                  [ Read 17 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo     M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo     M-6 Copy Text
```

Fig. 7. The process of Applying the ModSecurity.conf Configuration

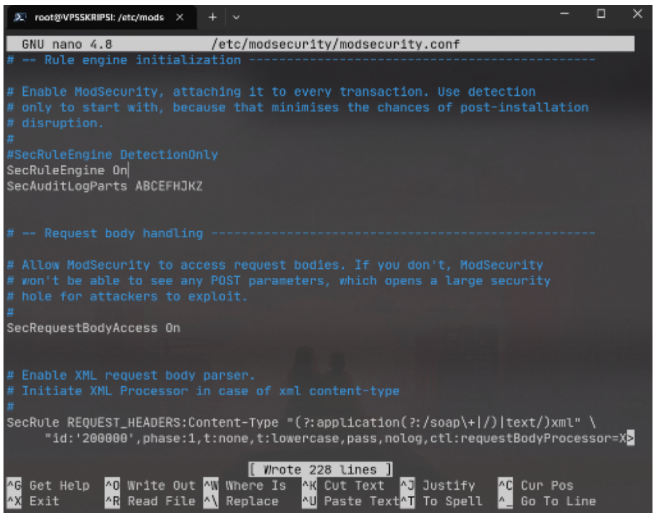
Next, rename the file "modsecurity.conf-recommended" to "modsecurity.conf" in the folder /etc/modsecurity/ using the command "mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf," and then restart apache2, as shown in Figure 9.



```
root@VPSSKRIPSI:/etc/modsecurity# ls
crs  modsecurity.conf-recommended  modsecurity.conf-recommended.save  unicode.mappi
ng  v3.3.4.zip
root@VPSSKRIPSI:/etc/modsecurity# mv /etc/modsecurity/modsecurity.conf-recommended
/etc/modsecurity/modsecurity.conf
root@VPSSKRIPSI:/etc/modsecurity# systemctl restart apache2
root@VPSSKRIPSI:/etc/modsecurity#
```

Fig. 8. Rename modsecurity.conf

The third step involves activating ModSecurity by executing the command "nano /etc/modsecurity/modsecurity.conf." Then, modify the "SecRuleEngine" from its initial setting "DetectionOnly" to "On." After making this change, save the file and restart apache2 as shown in the figure 10.



```
  GNU nano 4.8                /etc/modsecurity/modsecurity.conf
# -- Rule engine initialization ----------------------------------------------

# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
#SecRuleEngine DetectionOnly
SecRuleEngine On
SecAuditLogParts ABCEFHJKZ


# -- Request body handling ---------------------------------------------------

# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On


# Enable XML request body parser.
# Initiate XML Processor in case of xml content-type
#
SecRule REQUEST_HEADERS:Content-Type "(?:application(?:/soap\+|/)|text/)xml" \
     "id:'200000',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=X>

                          [ Wrote 228 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

Fig. 9. Enabling Mod Security WAF

The fourth step involves configuring ModSecurity rules, as ModSecurity cannot function without rules. In this regard, the rules employed utilize the built-in rules from ModSecurity. These built-in rules include syntax such as "SecRule VARIABLES OPERATOR [ACTIONS]." The created rules need to be placed in the ModSecurity directory to ensure proper application and functionality.

Table 3 illustrates the explanation of Mod Security rules. It includes the "SecRule" command for creating basic rules, "VARIABLES" for initializing variables to be checked, "OPERATOR" for initializing the detected operators, and "[ACTIONS]" for initializing the actions to be taken.

TESTING AND ANALYSIS

Table 3. Mod Security Rules Explanation

Command	Description
SecRule	Basic command to create a rule.
VARIABLES	Initialization of variables to be checked.
OPERATOR	Initialization of operators to be detected.
[ACTIONS]	Initialization of actions to be taken.

In this phase, comprehensive testing is conducted to verify the optimal performance of the system after implementation. This testing encompasses two distinct stages: testing against attacks from parties attempting to compromise the website, and testing the performance of the website with the integrated Web Application Firewall.

A. Attack Testing with Disabled WAF

In this attack testing, the researcher manually conducts it using two testing scenarios: performing testing with WAF disabled and after activating the Web Application Firewall. The following describes the attack testing using the Kak Seto School Admissions website with the Web Application Firewall disabled.

1. Testing Cross-site Scripting (XSS) Attack with Disabled WAF

Testing the Cross-site Scripting (XSS) attack with Disabled WAF utilizes an injection testing mechanism on the Path Parameter present in the target website. The Cross-site Scripting (XSS) attack on the target website exploits the path and parameters located within the URL column. The program code used for the Cross-site Scripting (XSS) attack involves JavaScript programming. Figure 11 illustrates an attempt at a Cross-site Scripting (XSS) attack, where the URL column in the browser can also be used for conducting the XSS attack. In this stage, the attack is carried out by injecting program code into the browser's URL column, such as "https://kriscode.my.id/index.php/ppdb=%3Cscript%3Ealert(document.cookie)%3C/script%3E". The XSS attack is performed by manipulating the path and parameters of the URL of the Kak Seto School Admissions website.

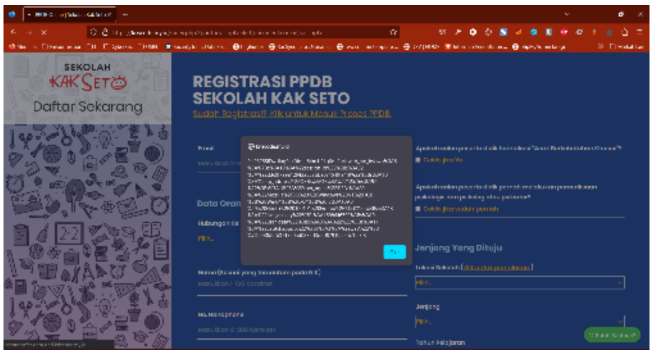


Fig. 10. Testing Cross-Site Scripting (XSS) Attack 1 with WAF Disabled

The second Cross-Site Scripting (XSS) testing involves reinserting the same JavaScript XSS program by leveraging the same path and parameters in the browser's URL column. The injected program code is as follows: "https://kriscode.my.id/index.php/?ppdb=<img

src=x onerror= "alert('XSS Testing');". Figure 12 depicts the attempt of Cross-Site Scripting (XSS) testing.

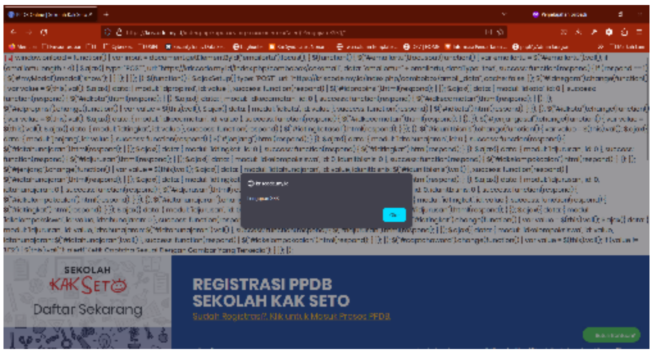


Fig. 11. Testing Cross-Site Scripting (XSS) Attack 2 with WAF Disabled

In the third XSS testing scenario, an attempt was made to insert a JavaScript program code using the Redirect Page method. This was achieved by exploiting the same path and parameters in the browser's URL column. The injected code aimed to redirect users from the vulnerable page to a different page controlled by the attacker. This type of attack, known as XSS Redirect Page, can be perilous as it can manipulate users into performing unintended actions or expose them to harmful content. This testing was carried out with Web Application Firewall (WAF) turned off. The results of these XSS testing scenarios highlight potential vulnerabilities in the website's security. Figure 13 depicts the third XSS testing attempt.

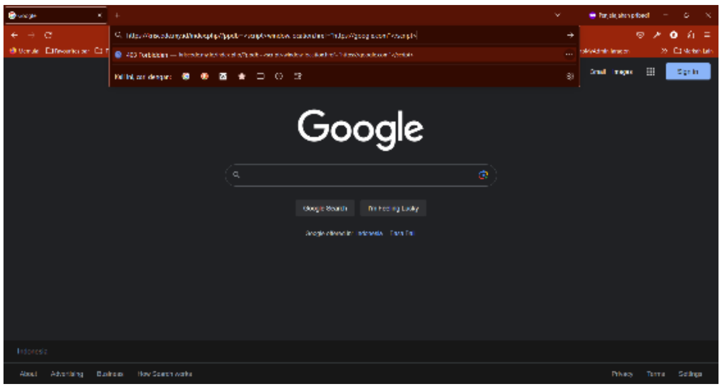
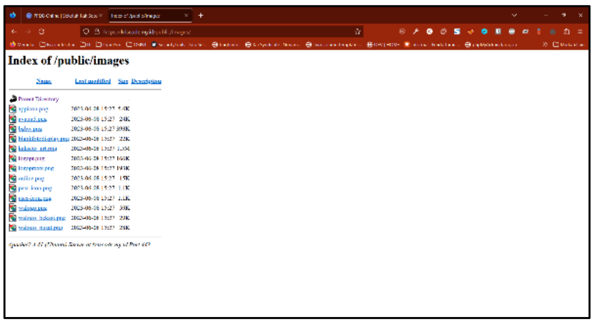


Fig. 12. Testing Cross-Site Scripting (XSS) Attack 3 Redirect Page with WAF Off

2. Testing Directory Listing Attack with WAF Disabled

Testing Directory Listing attack with WAF disabled involves a manual testing mechanism by right-clicking on an image present on the website. Subsequently, the image will be displayed, and the URL link will reveal the "public/images" folder. The attempt to carry out a Directory Listing attack involves accessing an image present on the initial registration page of the PPDB Sekolah Kak Seto website. This attempt successfully revealed the full image through the URL link: https://kriscode.my.id/public/images/logopt.png. Figure 14 illustrates the Directory Listing attack on the "public/images" folder.



Index of /public/images

```
Name                     Last modified     Size  Description
Parent Directory                             -
applono.png              2023-06-08 15:27  5.6K
avatar5.png              2023-06-08 15:27   24K
[illegible]              2023-06-08 15:27  393K
blankfotodisplay.png     2023-06-08 15:27   22K
kakseto_art.png          2023-06-08 15:27  1.3M
[illegible]              2023-06-08 15:27  166K
[illegible]              2023-06-08 15:27  193K
online.png               2023-06-08 15:27   15K
[illegible]              2023-06-08 15:27  1.1K
[illegible]              2023-06-08 15:27  1.1K
[illegible]              2023-06-08 15:27   50K
[illegible]              2023-06-08 15:27   29K
[illegible]              2023-06-08 15:27   28K
Apache/2.4.41 (Ubuntu) Server at [illegible] Port 443
```

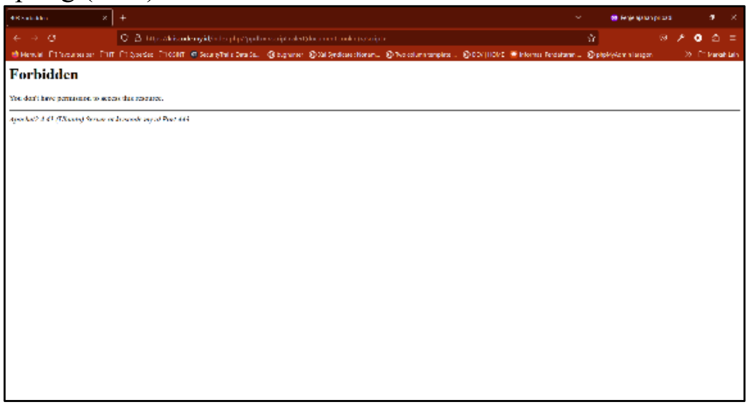
Fig. 13. Attempt of Directory Listing Attack on /public/images

B. Attacks Testing with Active WAF

Testing attacks on the PPDB Sekolah Kak Seto website with WAF enabled was carried out using the same techniques as the testing with WAF disabled. The attacks that were tested include Cross-site Scripting (XSS) and Directory Listing vulnerabilities that may exist in the web application. These attacks were performed on the PPDB Sekolah Kak Seto web application, which had the WAF activated.

1. Testing Cross-site Scripting (XSS) Attack with Active WAF

The attack uses the URL column in the browser to perform XSS attacks. In this phase, the attack is executed by injecting code into the URL column of the browser. The result of this testing indicates that the author was unsuccessful in injecting the Cross-site Scripting (XSS) attack code, instead displaying the message "404 Forbidden – You don't have permission to access this resource." Therefore, every request identified as an attack by the user will be redirected to this page. Figure 15 shows the display after attempting a Cross-site Scripting (XSS) attack with the active WAF.



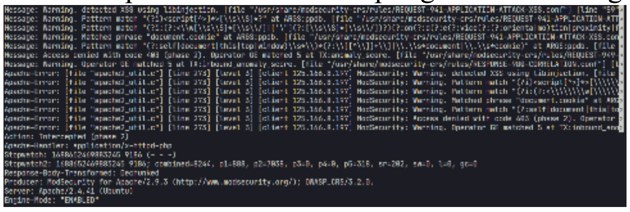
Forbidden

You don't have permission to access this resource.

Apache/2.4.41 (Ubuntu) Server at [illegible] Port 443

Fig. 14. Testing Attack Cross-Site Scripting 1 with WAF On

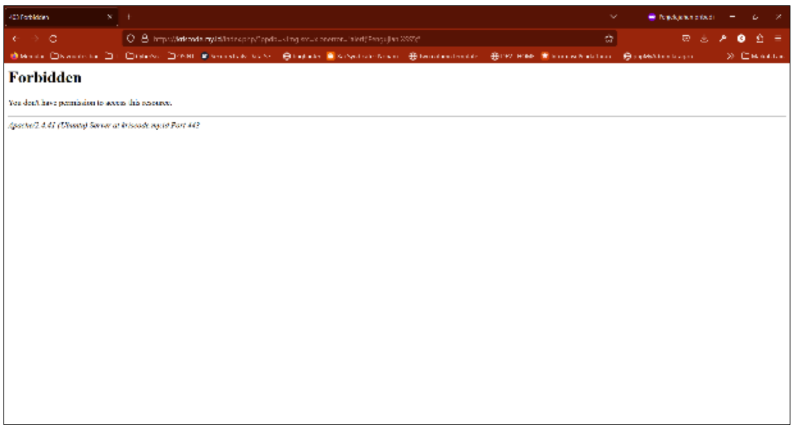
The result of Mod Security Audit Log recorded by ModSecurity as a Web Application Firewall when an attacker exploits Cross-Site Scripting is shown in Figure 16.



```
Message: Warning. detected XSS using libinjection. [file "/usr/share/modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "59"
Message: Warning. Pattern match "(?i)<script[^>]*>[\\s\\S]*?" at ARGS:ppdb. [file "/usr/share/modsecurity-crs/rules/REQUEST-941-APPLICATION-ATT
Message: Warning. Pattern match "(?i:(?:<\\w[\\s\\S]*[\\s\\/]|['\"](?:[\\s\\S]*[\\s\\/])?)(?:on(?:d(?:e(?:vice(?:(?:orienta|mo)tion|proximity|
Message: Warning. Matched phrase "document.cookie" at ARGS:ppdb. [file "/usr/share/modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.con
Message: Warning. Pattern match "(?:self|document|this|top|window)\\s*\\)*(?:\\[[^\\]]+\\]|\\.\\s*document|\\.\\s*cookie)" at ARGS:ppdb. [file
Message: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "/usr/share/modsecurity-crs/rules/REQUEST-949-
Message: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/RESPONSE-980-CORRELATION.conf"] [l
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 125.166.8.197] ModSecurity: Warning. detected XSS using libinjection. [file
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 125.166.8.197] ModSecurity: Warning. Pattern match "(?i)<script[^>]*>[\\\\\\
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 125.166.8.197] ModSecurity: Warning. Pattern match "(?i:(?:<\\\\\\\\w[\\\\\\
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 125.166.8.197] ModSecurity: Warning. Matched phrase "document.cookie" at ARG
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 125.166.8.197] ModSecurity: Warning. Pattern match "(?:self|document|this|to
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 125.166.8.197] ModSecurity: Access denied with code 403 (phase 2). Operator
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 125.166.8.197] ModSecurity: Warning. Operator GE matched 5 at TX:inbound_ano
Action: Intercepted (phase 2)
Apache-Handler: application/x-httpd-php
Stopwatch: 1688652469883245 9186 (- - -)
Stopwatch2: 1688652469883245 9186; combined=8244, p1=888, p2=7035, p3=0, p4=0, p5=318, sr=202, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.3 (http://www.modsecurity.org/); OWASP_CRS/3.2.0.
Server: Apache/2.4.41 (Ubuntu)
Engine-Mode: "ENABLED"
```

Fig. ***15.*** *Mod Security Audit Log 1*

Testing of the second Cross-site Scripting attack with JavaScript was successfully blocked by the Web Application Firewall, resulting in a 404 Forbidden message - You don't have permission to access this resource. Below is Figure 16, illustrating that the testing of the second Cross-Site Scripting Attack.



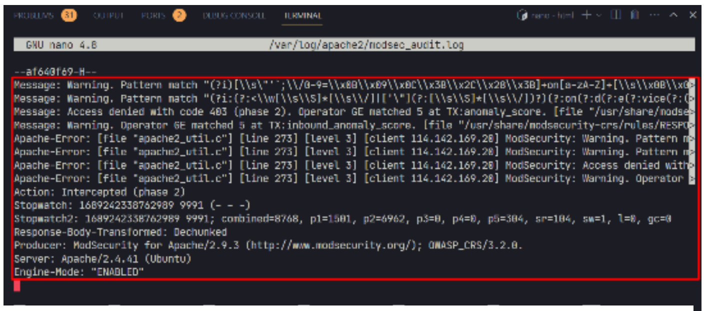
Forbidden

You don't have permission to access this resource.

Apache/2.4.41 (Ubuntu) Server at [illegible] Port 443

Fig. 16. Testing Attack Cross-Site Scripting 2 with WAF On

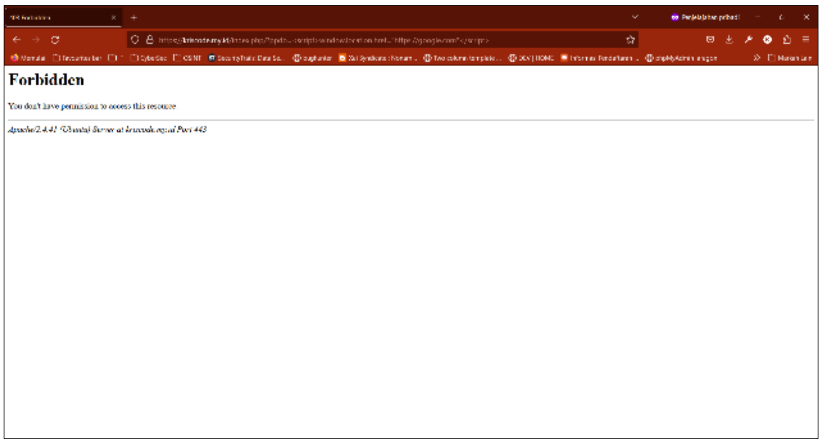
The result of Mod Security Audit Log recorded by ModSecurity as a Web Application Firewall when an attacker exploits Cross-Site Scripting 2 is shown in Figure 18.



```
GNU nano 4.8                    /var/log/apache2/modsec_audit.log

--af640f69-H--
Message: Warning. Pattern match "(?i)[\\s\"'`;\\/0-9=\\x0B\\x09\\x0C\\x3B\\x2C\\x28\\x3B]+on[a-zA-Z]+[\\s\\x0B\\x0
Message: Warning. Pattern match "(?i:(?:<\\w[\\s\\S]*[\\s\\/]|['\"](?:[\\s\\S]*[\\s\\/])?)(?:on(?:d(?:e(?:vice(?:(
Message: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "/usr/share/modse
Message: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/RESPO
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.28] ModSecurity: Warning. Pattern m
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.28] ModSecurity: Warning. Pattern m
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.28] ModSecurity: Access denied with
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.28] ModSecurity: Warning. Operator
Action: Intercepted (phase 2)
Stopwatch: 1689242338762989 9991 (- - -)
Stopwatch2: 1689242338762989 9991; combined=8768, p1=1501, p2=6962, p3=0, p4=0, p5=304, sr=104, sw=1, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.3 (http://www.modsecurity.org/); OWASP_CRS/3.2.0.
Server: Apache/2.4.41 (Ubuntu)
Engine-Mode: "ENABLED"
```

***Fig. 17.** Mod Security Audit Log 1*

Testing a Cross-Site Scripting attack using JavaScript Redirect Page, which was successfully blocked by the Web Application Firewall, resulting in a 404 Forbidden message - You don't have permission to access this resource. Figure 19 shows the results of testing a Cross-site Scripting attack using javascript Redirect Page.



Forbidden

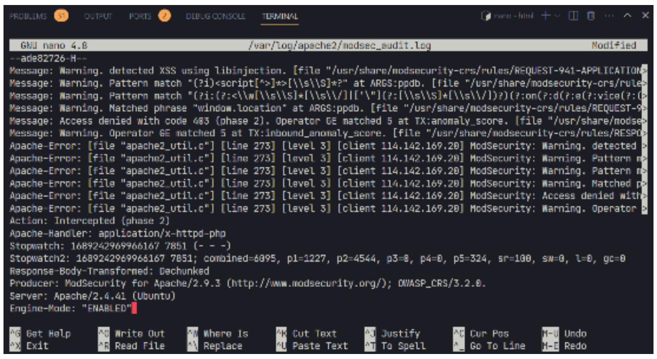
You don't have permission to access this resource.

Apache/2.4.41 (Ubuntu) Server at [illegible] Port 443

Fig. 18. Testing Cross-Site Scripting Attack 3 Redirect Page with WAF On

The Audit Log Mod Security results from the third Cross-Site Scripting attack testing using a different script were successfully recorded by the Web Application

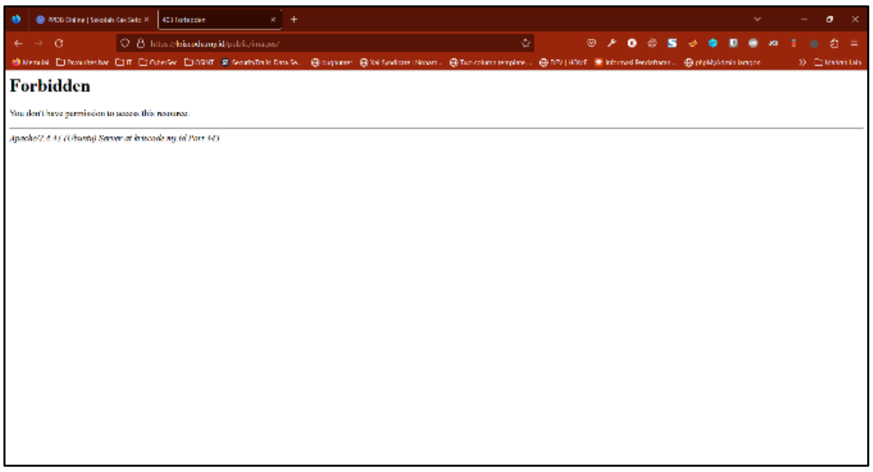
Firewall. A warning for an XSS attack using libinjection was detected and successfully blocked. Figure 20 shows the Audit Log Mod Security results form the third attack.



```
 GNU nano 4.8                /var/log/apache2/modsec_audit.log                Modified
--ade82726-H--
Message: Warning. detected XSS using libinjection. [file "/usr/share/modsecurity-crs/rules/REQUEST-941-APPLICATION>
Message: Warning. Pattern match "(?i)<script[^>]*>[\\s\\S]*?" at ARGS:ppdb. [file "/usr/share/modsecurity-crs/rule>
Message: Warning. Pattern match "(?i:(?:<\\w[\\s\\S]*[\\s\\/]|['\"](?:[\\s\\S]*[\\s\\/])?)(?:on(?:d(?:e(?:vice(?:(>
Message: Warning. Matched phrase "window.location" at ARGS:ppdb. [file "/usr/share/modsecurity-crs/rules/REQUEST-9>
Message: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "/usr/share/modse>
Message: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/RESPO>
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.20] ModSecurity: Warning. detected >
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.20] ModSecurity: Warning. Pattern m>
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.20] ModSecurity: Warning. Pattern m>
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.20] ModSecurity: Warning. Matched p>
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.20] ModSecurity: Access denied with>
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.20] ModSecurity: Warning. Operator >
Action: Intercepted (phase 2)
Apache-Handler: application/x-httpd-php
Stopwatch: 1689242969966167 7851 (- - -)
Stopwatch2: 1689242969966167 7851; combined=6095, p1=1227, p2=4544, p3=0, p4=0, p5=324, sr=100, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.3 (http://www.modsecurity.org/); OWASP_CRS/3.2.0.
Server: Apache/2.4.41 (Ubuntu)
Engine-Mode: "ENABLED"

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

Fig. 19. Mod Security Audit Log 3

2. **Testing Directory Listing Attack with Active WAF**
   Testing of Directory Listing attack on the Web Server of Enrollment Kak Seto School website using the Web Application Firewall's security measures. In this test, the firewall operates based on rules and the result obtained is that the firewall successfully blocked the testing and displayed a 404 Forbidden information message. Figure 21 illustrates the testing of Directory Listing attack.



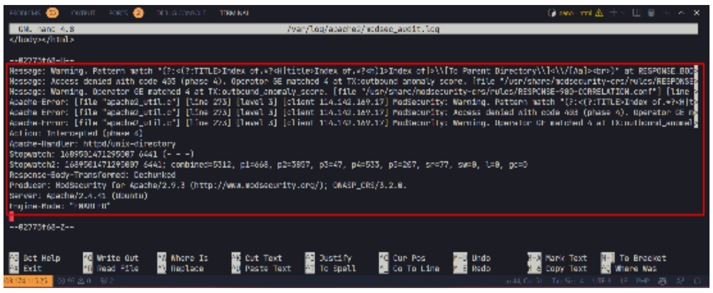
Forbidden

You don't have permission to access this resource.

Apache/2.4.41 (Ubuntu) Server at [illegible] Port 443

Fig. 20. Testing Directory Listing Attack with WAF On

Mod Security Audit Log results recorded by ModSecurity as a Web Application Firewall when an attacker attempted to exploit Directory Listing. Figure 4.39 shows the Mod Security Audit Log results recorded by ModSecurity.



```
 GNU nano 4.8                /var/log/apache2/modsec_audit.log
</body></html>

--82773f68-H--
Message: Warning. Pattern match "(?:<(?:TITLE>Index of.*?<H|title>Index of.*?<h)1>Index of|>\\[To Parent Directory\\]<\\/[Aa]><br>)" at RESPONSE_BOD>
Message: Access denied with code 403 (phase 4). Operator GE matched 4 at TX:outbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/RESPONSE>
Message: Warning. Operator GE matched 4 at TX:outbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/RESPONSE-980-CORRELATION.conf"] [line >
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.17] ModSecurity: Warning. Pattern match "(?:<(?:TITLE>Index of.*?<H|t>
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.17] ModSecurity: Access denied with code 403 (phase 4). Operator GE m>
Apache-Error: [file "apache2_util.c"] [line 273] [level 3] [client 114.142.169.17] ModSecurity: Warning. Operator GE matched 4 at TX:outbound_anomal>
Action: Intercepted (phase 4)
Apache-Handler: httpd/unix-directory
Stopwatch: 1689581471295087 6441 (- - -)
Stopwatch2: 1689581471295087 6441; combined=5312, p1=668, p2=3857, p3=47, p4=533, p5=207, sr=77, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.3 (http://www.modsecurity.org/); OWASP_CRS/3.2.0.
Server: Apache/2.4.41 (Ubuntu)
Engine-Mode: "ENABLED"

--82773f68-Z--

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo      M-A Mark Text   M-] To Bracket
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo      M-6 Copy Text   ^Q Where Was
```

Fig. 21. Mod Security Audit Log 4

RESULT AND DISCUSSIONS

Based on the conducted attack testing results, it is concluded that the Website of PPDB Sekolah Kak Seto, deployed on a Virtual Private Server with the domain https://kriscode.my.id, lacks input validation and filtering, allowing the website to respond to inputs, execute the provided code, and obtain cookies from users of the PPDB Sekolah Kak Seto website. Table 4 below presents the outcomes of the Cross-Site Scripting (XSS) and Security Misconfiguration (Directory Listing) attack testing conducted on the PPDB Sekolah Kak Seto website.

Table 4. Presents the results of Web Server exploitation

No	Jenis Serangan	Domain	Kode Program Exploit	WAF Disabled	WAF Active
1	**Cross-site Scripting (XSS)**	kriscode.my.id	https://kriscode.my.id/index.php/?ppdb=<script>alert(document.cookie)</script>	Success	Not Successful
2			https://kriscode.my.id/index.php/?ppdb=<img src=x onerror=”Pengujian XSS’);”	Success	Not Successful
3			https://kriscode.my.id/index.php/?ppdb=<script>window.location.href="https://google.com"</script>	Success	Not Successful
4			https://kriscode.my.id/index.php/psb_calonsiswa	Success	Not Successful
5			https://kriscode.my.id/index.php/psb_calon_asesmen	Not Successful	Not Successful
6			https://kriscode.my.id/index.php/psb_calon_ortu	Not Successful	Not Successful
7	**Security Misconfiguration (Directory Listing)**		https://kriscode.my.id/public/	Success	Not Successful
8			https://kriscode.my.id/public/images/	Success	Not Successful
9			https://kriscode.my.id/public/css/	Success	Not Successful
10			https://kriscode.my.id/public/fonts/	Success	Not Successful
11			https://kriscode.my.id/public/js/	Success	Not Successful
12			https://kriscode.my.id/uploads/	Success	Not Successful
13			https://kriscode.my.id/uploads/calonsiswa/	Success	Not Successful
14			https://kriscode.my.id/uploads/captcha/	Success	Not Successful
15			https://kriscode.my.id/uploads/material/	Success	Not Successful
16			https://kriscode.my.id/uploads/onlinekronologis/	Success	Not Successful
17			https://kriscode.my.id/uploads/pegawai/	Success	Not Successful

Table 5. Pingdom Test Results

No	Indicator	Description	
		Disabled WAF	Active WAF
1	Performance Grade	70 (D)	69 (D)
2	Page Size	3.6 MB	3.6 MB

3	Load time	2.76 s	3.23 s
4	Request	69	69

The data presented in Table 5 reveals that the load time results exhibit only a marginal decrease when utilizing a Web Application Firewall (WAF), where implementing a WAF does not significantly alter the website's load time. Similarly, the performance grade outcome exhibits a mere difference of 1 point. Pingdom's analysis of page size and request indicators showed no noteworthy disparities between using and not using a WAF; resource demands and requests remained consistent. Upon analyzing these results, it becomes evident that WAF implementation does not impart substantial changes in page size or request quantity. However, it does result in a 16.7% increase in site load time, rising from 2.76 seconds to 3.23 seconds. Despite this elevated load time, the impact remains within acceptable ranges.
It is important to consider that although load time increased slightly after WAF application, the impact remains manageable. Continuously monitoring and making necessary adjustments can mitigate this effect, ensuring optimal website performance. Ultimately, while WAF provides additional layers of protection against attacks like Cross-Site Scripting (XSS), Directory Listing, and other web-based threats, the decision to employ it should be based on more comprehensive security considerations. While this testing shows minor performance differences or lack thereof when utilizing a WAF, the choice to incorporate WAF remains crucial to overall web application security enhancement. WAF serves as a preventive measure against attacks and safeguards web applications from potential vulnerabilities.

Table 6. GTmetrix Comparison Results

No	**Indicator**	**Description**	
		Disabled WAF	**Active WAF**
1	*GTmetrix Grade*	B	C
2	*Performance Score*	83%	78%
3	*Largest Contentful Pain*	2.2s	2.7s
4	*Total Blocking time*	0ms	0ms
5	*Cumulative Layout Shift*	0	0

Based on Table 6, the testing results show that the utilization of a Web Application Firewall (WAF) doesn't significantly affect the scores of all measured indicators. The GTmetrix Grade experienced a decline of 16.7%, transitioning from B to C, resulting in a Performance Score of 83%. However, after implementing the WAF, the Performance Score dropped to 78%, representing a 6.0% decrease from the initial 83%. Notably, the Largest Contentful Paint indicator saw a 22.7% increase in time, from 2.2 seconds before WAF implementation to 2.7 seconds after. The time difference without WAF was only 6ms compared to using WAF. Largest Contentful Paint measures the time taken to load the largest content element, such as the main image or title text, within a page. Meanwhile, Total Blocking Time and Cumulative Layout Shift remained unchanged. These results in Table 4.5 were obtained from the condition of the PPDB Sekolah Kak Seto website deployed on a Virtual Private Server (VPS).

CONCLUSIONS

The analysis of the impact of utilizing a Web Application Firewall (WAF) on security in web-based applications, with the case study of the PPDB Sekolah Kak Seto website, reveals several crucial findings. Firstly, the implementation of a WAF substantially enhances the security of web-based applications, including the PPDB Sekolah Kak Seto website. The WAF effectively safeguards applications from cyber threats that could compromise data integrity, confidentiality, and availability. Secondly, integrating the Web Application Firewall into the PPDB Sekolah Kak Seto website yields positive changes in countering Cross-Site Scripting and Security

Misconfiguration (Directory Listing) attacks. The WAF successfully prevents these attacks by identifying and blocking potentially harmful exploitation attempts, bolstering the defense of the website against cyber threats. Lastly, deploying the Web Application Firewall impacts website performance. Pingdom testing shows a Performance Grade decrease from 70 to 69 and a slight increase in Load Time from 2.76 seconds to 3.23 seconds post-WAF implementation. GTmetrix testing indicates a grade reduction from B to C and a rise in Largest Contentful Paint time from 2.2 seconds to 2.7 seconds with WAF. Nonetheless, despite the performance impact, particularly evident in increased load times in Pingdom and GTmetrix evaluations, the benefits gained in heightened security, including the prevention of Cross-Site Scripting and Security Misconfiguration attacks, remain significant.

REFERENCES

[1] F. O. Sonmez and B. G. Kilic, "Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results," *IEEE Access*, vol. 9, pp. 25858–25884, 2021, doi: 10.1109/ACCESS.2021.3057044.

[2] M. Fahlevi, M. Saparudin, S. Maemunah, D. Irma, and M. Ekhsan, "Cybercrime Business Digital in Indonesia," *E3S Web Conf.*, vol. 125, no. 201 9, pp. 1–5, 2019, doi: 10.1051/e3sconf/201912521001.

[3] N. Sun *et al.*, "Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives," *IEEE Commun. Surv. Tutorials*, vol. PP, p. 1, 2023, doi: 10.1109/COMST.2023.3273282.

[4] E. A. Altulaihan, A. Alismail, and M. Frikha, "A Survey on Web Application Penetration Testing," *Electron.*, vol. 12, no. 5, 2023, doi: 10.3390/electronics12051229.

[5] D. Kalla, F. Samaah, S. Kuraku, and N. Smith, "Phishing Detection Implementation using Databricks and Artificial Intelligence," *Int. J. Comput. Appl.*, vol. 185, no. 11, pp. 1–11, 2023, doi: 10.5120/ijca2023922764.

[6] R. Riska and H. Alamsyah, "Penerapan Sistem Keamanan Web Menggunakan Metode Web Aplication Firewall," *J. Amplif. J. Ilm. Bid. Tek. Elektro Dan Komput.*, vol. 11, no. 1, pp. 37–42, 2021, doi: 10.33369/jamplifier.v11i1.16683.

[7] S. Alazmi and D. C. De Leon, "A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners," *IEEE Access*, vol. 10, pp. 33200–33219, 2022, doi: 10.1109/ACCESS.2022.3161522.

[8] "OWASP Top Ten | OWASP Foundation." https://owasp.org/www-project-top-ten/ (accessed Aug. 01, 2022).

[9] B. I. Dewangkara, K. S. Santi, V. A. Putri, and I. M. E. Listartha, "Penerapan Analisis Kerentanan XSS dan Rate Limiting pada Situs Web MTsN 3 Negara Menggunakan OWASP ZAP," *J. Inform. Upgris*, vol. 8, no. 1, pp. 1–6, 2022, doi: 10.26877/jiu.v8i1.10266.

[10] M. A. Mu'min, A. Fadlil, and I. Riadi, "Analisis Keamanan Sistem Informasi Akademik Menggunakan Open Web Application Security Project Framework," *J. Media Inform. Budidarma*, vol. 6, no. 3, p. 1468, 2022, doi: 10.30865/mib.v6i3.4099.

[11] A. W. Marashdih, Z. F. Zaaba, and K. Suwais, "An Enhanced Static Taint Analysis Approach to Detect Input Validation Vulnerability," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 2, pp. 682–701, 2023, doi: 10.1016/j.jksuci.2023.01.009.

[12] F. Nurelia, S. Putri, Y. B. Utomo, and U. I. Kadiri, "Analisa Celah Keamanan Pada Website Pemerintah Kabupaten Kediri Menggunakan Metode Penetration Testing Melalui Kali Linux," vol. 7, pp. 52–59, 2023.

[13] P. Panwar, H. Mishra, and R. Patidar, "An Analysis of the Prevention and Detection of Cross Site Scripting Attack," *Int. J. Emerg. Trends Eng. Res.*, vol. 11, no. 1, pp. 30–34, 2023, doi: 10.30534/ijeter/2023/051112023.

[14] R. Rizal and Y. Sumaryana, "Peningkatan Keamanan Aplikasi Web Menggunakan Web Application Firewall (WAF) Pada Sistem Informasi Manajemen Kampus Terintegrasi,"

J. ICT Inf. Commun. Technol., vol. 20, no. 2, pp. 323–330, 2021, doi: 10.36054/jict-ikmi.v20i2.416.
[15] K. Dhiatama Ayunda *et al.*, "Implementation and Analysis ModSecurity on Web-Based Application with OWASP Standards," *Jurnal.Mdp.Ac.Id*, vol. 8, no. 3, pp. 1638–1650, 2021, [Online]. Available: https://jurnal.mdp.ac.id/index.php/jatisi/article/view/1223
[16] M. L. N. Suryana, N. R. S. Muda, D. Minggu, R. Agustiady, and C. Herkariawan, "Implementation of firewall for web server access management based on application gateway for TNI AD website," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1098, no. 2, p. 022105, 2021, doi: 10.1088/1757-899x/1098/2/022105.

AUTHORS BIBLIOGRAPHY

Krisna Dewa Pratama is an undergraduate student in the Department of Informatics at Universitas Ahmad Dahlan. His research interest is centered around Cyber Security.
Email: rachmad1900018326@webmail.uad.ac.id

Nuril Anwar, is a lecturer in Departement of Informatics at Universitas Ahmad Dahlan. His research interest is centered on Computer Networks & Security, Digital Forensics.
Email: nuril.anwar@tif.uad.ac.id