VOL 2 (2018) NO 4

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

SQL Injection and Cross Site Scripting Prevention Using OWASP Web Application Firewall

Robinson[#], Memen Akbar[#], Muhammad Arif Fadhly Ridha[#]

[#]*Informatics Engineering Study Program, Politeknik Caltex Riau, Pekanbaru, Indonesia*
e-mail: robeansonit@gmail.com, memen@pcr.ac.id, fadhly@pcr.ac.id

***Abstract*— Web Application or website are widely used to provide functionality that allows companies to build and maintain relationships with their customers. The Information stored by web applications is often confidential and, if obtained by malicious attackers. Its exposure could result in substantial losses for both consumers and companies. SQL Injection and Cross Site Scripting are attacks that aiming web application database vulnerabilities. Its can allow malicious attackers to manipulate web server database that can cause various data lost, information thieving, and inconsistent of data. Therefore, this research propose the Open Web Application Security Project (OWASP) ModSecurity Core Rule Set which can help administrator securing the web servers. OWASP operate by blocking IP Address which try to breaking the security rule, monitoring network traffic and preventing suspicious network requesting from outside.**

***Keywords*— web application, SQL Injeciton, Cross Site Scripting, Open Web Application Security Project.**

I. Introduction

Web application or website are widely used to provide functionality that allows companies to build and maintain relationships with their customers. The information stored by web applications is often confidential and, if obtained by malicious attackers, its exposure could result in substantial losses for both consumers and companies. (Mate Vibhakti, 2014). There are a lot of technique that commonly use by the attackers such as SQL Injection, Cross Site Scripting, Brute Force, Worm, deface, etc to intrude web application. By using methods which are specifically aimed at exploiting potential weak spots in the web applications, The attackers were not easily detected by the System with sufficient accuracy.

SQL Injection and Cross Site Scripting (XSS) are attacks that aiming web application database vulnerabilities. These two kinds of attacks allow the malicious attackers to manipulate web server database that can cause various data lost, information thieving, and inconsistent of data. Commonly administrator using a secondary database to backup data or information from main database. When there was an attack, the administrator will retrieve the database by recovery it using the backup one. But this kind of plan can't prevent the data lost or information thieving. One of information thieving damage example is the attackers can get the username and password from database, and using it for login to the website as an administrator. It will giving the attackers the privillage of administrator to control the web application.

Open Web Application Security Project (OWASP) ModSecurity Core Rule Set (CRS) is a web application firewall that can help administrator securing the web servers. OWASP operate by blocking IP Address which try to breaking the security rule, monitoring network traffic, and preventing suspicious network requesting from outside. ModSecurity works by gathers malicious payloads from various web sources and consolidates them into a blacklist. ModSecurity rules then use a fast pattern matching algorithm to inspect outbound html for signs of this malicious code. ModSecurity can then alert/block/clean the malicious code to prevent infecting web application.

OWASP ModSecurity Core Rule Set is a solution for covering web application vulnerabilities and securing web application or website from malicious attacks that could result in substantial losses for consumers and companies. OWASP ModSecurity Core Rule Set also will be implemented to Politeknik Caltex Riau web application especially for academic and student sites.

II. Literature Review

A. *Cross Site Scripting*

Cross-site scripting (XSS) occurs when dynamically generated web pages display input that is not properly validated. This allows an attacker to embed malicious

JavaScript code or script into the generated page and execute the script on the machine of any user that views that site. Cross-site scripting could potentially impact any site that allows users to enter data. This vulnerability is commonly seen on:

a) Search engines that echo the search keyword that was entered
b) Error messages that echo the string that contained the error
c) Forms that are filled out where values are later presented to the user
d) Web message boards that allow users to post their own messages.

An attacker who uses cross-site scripting successfully might compromise confidential information, manipulate or steal cookies, create requests that can be mistaken for those of a valid user, or execute malicious code on the end-user systems.

B. SQL Injection

SQL Injection is a vulnerability that results when you give an attacker the ability to influence the Structured Query Language (SQL) queries that an application passes to a back-end database. By being able to influence what is passed to the database, the attacker can leverage the syntax and capabilities of SQL itself, as well as the power and flexibility of supporting database functionality and operating system functionality available to the database. Any code that accepts input from an untrusted source and then uses that input to form dynamic SQL statements could be vulnerable.

These are the classification of SQL injection types according to Halfond, Viegas and Orso researches (Justin, 2009) .

1. Tautology
2. Logically incorrect queries
3. Union queries
4. Piggy-backed Querie
5. Stored Procedure
6. Blind Injection
7. Timing Attacks

Some impact if attackers success injecting SQL to web servers :

- An attacker can use SQL Injection to bypass authentication or even impersonate specific users.
- An SQL Injection vulnerability could allow the complete disclosure of data residing on a database server.
- An attacker could use SQL Injection to alter data stored in a database. Altering data affects data integrity and could cause repudiation issues.
- An attacker could use an SQL Injection vulnerability to delete data from a database. Even if an appropriate backup strategy is employed, deletion of data could affect an application's availability until the database is restored.
- An attacker could use SQL Injection as the initial vector in an attack of an internal network that sits behind a firewall.

C. Web Application Firewall

To prevent malicious attacks, there are network appliances that are added to the computer network such as Intrusion Prevention System (IPS) and Intrusion Detection System (IDS). Both IPS and IDS help monitor the network but are only limited to detecting and notifying administrators about the abnormal network behaviour and can still succumb to complex attacks or attacks that may not have been recognized by the system. IPS checks the signature of the attacks and must rely on patterns to determine if there is an attack. IPS do not have the ability to understand web application protocol logic and also cannot fully distinguish if a request is normal or malformed at the application layer. When IPS interrogate traffic against signatures and anomalies, WAF interrogate the behaviour of logic of what is request and returned.

The web application firewall is a type of firewall that checks the data level of the packets to protect the application layer of the OSI model. By checking the data portion of the packets, more detailed information is revealed which is referred to as the granularity of a packet. For example, inside the HTTP header there would be http requests and inside http request would be user agents, cookies and more. Now being able to see this information, a more informed decision is now made in regards to the security controls for specific packets passed to the application.

The Web Application Firewall is installed as a running service in the web server or system it needs to protect, particularly the application layer level. Its main purpose is to check all incoming HTTP traffic, then accepts and drops the incoming HTTP traffic according to the rules that was set by the network administrator. The administrator through a text editor configures the rule-sets of the Web Application Firewall. A manual is provided for the syntax and format of the rules. The structure of the rules has the keywords "allow" or "reject" as its basis for the decision, followed by the different options of HTTP request headers and the value which the administrator wants to be checked in the payload. (Endraca, King, Nodalo, Maria, & Sabas, 2013).

D. OWASP ModSecurity Core Rule Set

Open Web Application Security Project or OWASP ModSecurity Core Rule Set (CRS) is a set of generic attack detection rules for use with ModSecurity or compatible web application firewalls. The CRS aims to protect web applications from a wide range of attacks, including the OWASP Top Ten, with a minimum of false alerts. (Curphey & Groves, 2006).

E. Attack Application

Attack Application that used in this research are:

1. SQLMap

Sqlmap is an open source command-line automatic SQL injection tool that was released under the terms of the GNU GPLv2 license by Bernardo Damele A. G. and Daniele Bellucci.

2. BeeF XSS Framework

BeEF (Browser Exploitation Framwork) is a browser-based exploit package that "hooks" one or more browsers as beachheads for launching attacks. A user can be hooked by accessing a customized URL and continue to see typical web traffic, while an attacker has access to the user's session. BeEF bypasses network security appliances and host–based, antivirus applications by targeting the vulnerabilities found in common browsers, such as Internet Explorer and Firefox.

3. XSSer

Cross Site “Scripter” (aka X S Ser) is an open source penetration testing tool that automates the process of detecting, exploiting and reporting XSS vulnerabilities in web-based applications.

F. *Cpanel*

CPanel is a web hosting control software that can be accessed using any modern web browser. It is designed to make the complex task of managing hosting account easy. (Pedersen, 2006).

G. *Simple Network Management Protocol*

Simple Network Management Protocol (SNMP) is a protocol used to monitor and control networks from elsewhere (remote) (Syafrizal, 2005).

H. *Research Method*

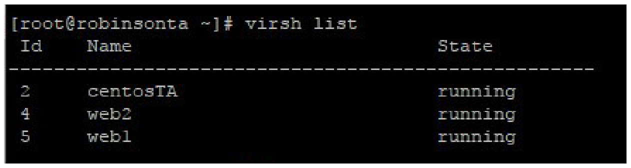
Research method that been used in this final project are:

1. Literature Review
   The collection of references in this research come from the journals, books, e-books, articles which have correlation with this final project topic.
2. Library Study
   The collection of references in this research come from the final project reports and papers which have correlation with this final project topic.
3. Implementation
   OWASP ModSecurity Web Application Firewall will be implemented and configure on Politeknik Caltex Riau web application server Cpanel.
4. Testing
   Testing activities will be done after web application firewall already been implemented on web application server by trying some several vulnerabilities testing.
5. Analyze and Conclusion
   Analyzing and taking conclusion of the testing activities results.

III. Result and Analyze

A. *Result of Implementing KVM Virtualization.*

3 nodes that been installed on physic server will show in the Figure 1 :

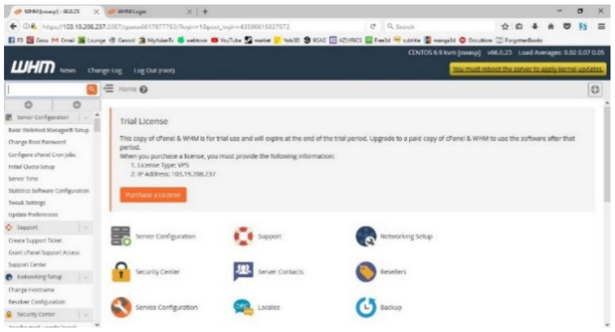


```
[root@robinsonta ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 2     centosTA                       running
 4     web2                           running
 5     web1                           running
```

Fig. 1 KVM Virtualization

B. *Result of CPanel Installation and Configuration*

Cpanel have been installed on web1 server and web2 server. After the installation finished, client can access the web server configuration through port 2083 for Cpanel Interface and port 2087 for WHM interface. The detail Interface will be show in the Figure 2 and Figure 3.



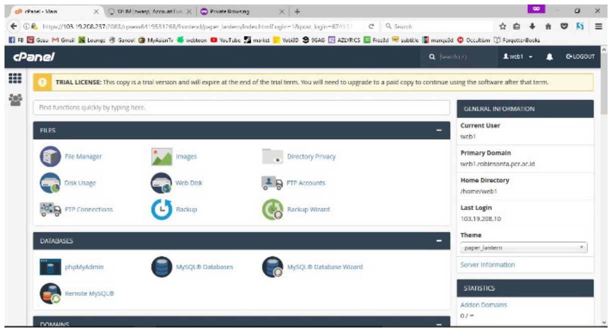
CENTOS 6.9 kvm [owasp] v66.0.23 Load Averages: 0.02 0.07 0.05

You must reboot the server to apply kernel updates.

Trial License

This copy of cPanel & WHM is for trial use and will expire at the end of the trial period. Upgrade to a paid copy of cPanel & WHM to use the software after that period.
When you purchase a license, you must provide the following information:
1. License Type: VPS
2. IP Address: 103.19.208.237

Fig. 2 WHM Index Interface (port 2087)



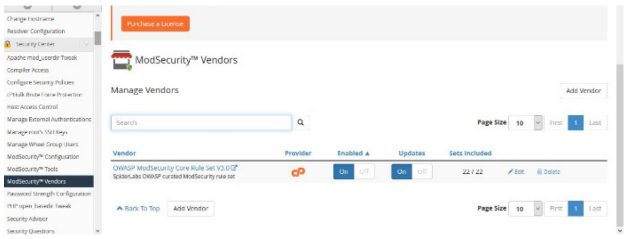
TRIAL LICENSE: This copy is a trial version and will expire at the end of the trial term. You will need to upgrade to a paid copy to continue using the software after that term.

Current User: web1
Primary Domain: web1.robinsonta.pcr.ac.id
Home Directory: /home/web1
Last Login: 103.19.208.10
Theme: paper_lantern

Fig. 3 Cpanel Index Interface (port 2083)

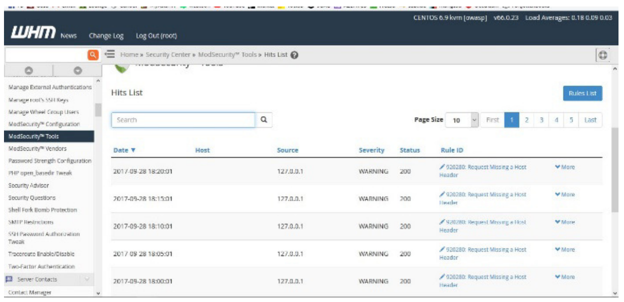
C. *OWASP ModSecurity Configuration*

On web1 server, ModSecurity will be set to active as show in the Figure 3. So we can see the Hit List or web server Logs if there is an attack as show in the Figure 4.



Vendor	Provider	Enabled	Updates	Sets Included
OWASP ModSecurity Core Rule Set V3.0 SpiderLabs OWASP curated ModSecurity rule set		On	On	22 / 22

Fig. 4 ModSecurity Vendor Activation

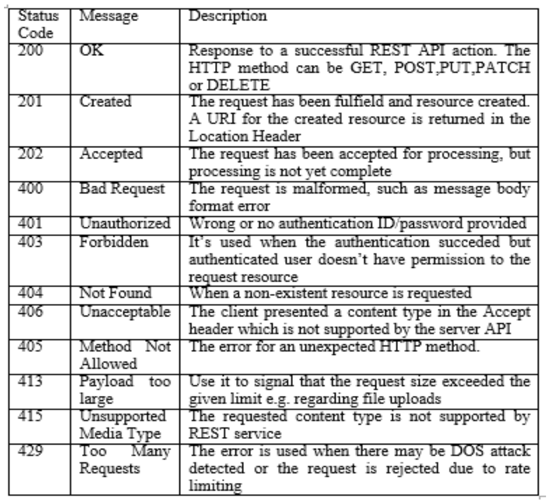


CENTOS 6.9 kvm [owasp] v66.0.23 Load Averages: 0.18 0.09 0.03

Date	Host	Source	Severity	Status	Rule ID
2017-09-28 18:20:01		127.0.0.1	WARNING	200	920280: Request Missing a Host Header
2017-09-28 18:15:01		127.0.0.1	WARNING	200	920280: Request Missing a Host Header
2017-09-28 18:10:01		127.0.0.1	WARNING	200	920280: Request Missing a Host Header
2017 09 28 18:05:01		127.0.0.1	WARNING	200	920280: Request Missing a Host Header
2017-09-28 18:00:01		127.0.0.1	WARNING	200	920280: Request Missing a Host Header

Fig. 5 OWASP ModSecurity Hit List

Each number of the status on the Figure 5 has its own definition which will be describe on the Figure 6



Status Code	Message	Description
200	OK	Response to a successful REST API action. The HTTP method can be GET, POST,PUT,PATCH or DELETE
201	Created	The request has been fulfield and resource created. A URI for the created resource is returned in the Location Header
202	Accepted	The request has been accepted for processing, but processing is not yet complete
400	Bad Request	The request is malformed, such as message body format error
401	Unauthorized	Wrong or no authentication ID/password provided
403	Forbidden	It's used when the authentication succeded but authenticated user doesn't have permission to the request resource
404	Not Found	When a non-existent resource is requested
406	Unacceptable	The client presented a content type in the Accept header which is not supported by the server API
405	Method Not Allowed	The error for an unexpected HTTP method.
413	Payload too large	Use it to signal that the request size exceeded the given limit e.g. regarding file uploads
415	Unsupported Media Type	The requested content type is not supported by REST service
429	Too Many Requests	The error is used when there may be DOS attack detected or the request is rejected due to rate limiting

Fig. 6 Status Code Description

D. Attack Result

1. SQL Injection
   After 15 times of attack testing for each kind of SQL Injection using 3 differences on both of the web servers, the result will be show in the table 1.

TABLE I
SQL INJECTION RESULT

Attack	Result	
	OWASP	NOOWASP
Tautology	Failed	Success
Logically	Success	Success
Union Queries	Failed	Success
Piggy-backed	Failed	MySQL countered
Stored Procedure	Failed	MySQL countered
Blind Injection	Failed	MySQL countered
Timing Attacks	Failed	MySQL countered

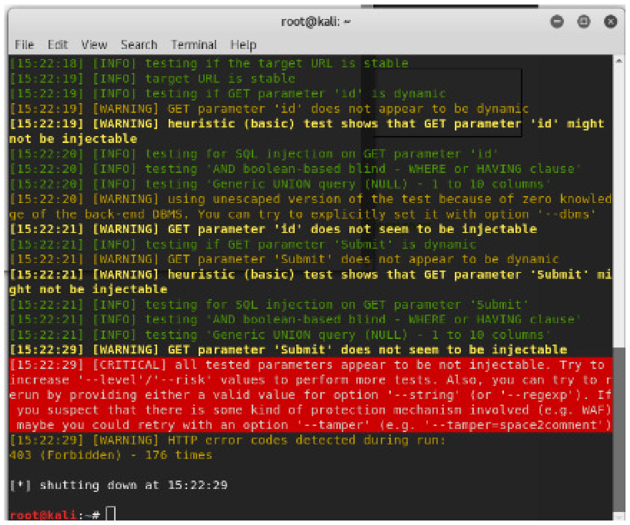
From Table 1, Failed status mean OWASP ModSecurity success securing web server and detect attacks log/ Hit List from those attacks. Success status mean the attackers success injecting SQL query into web server and MYSQL countered mean those SQL query already been patched by the MySQL itself.

2. SQLmap exploitation
   The second attacks testing was using SQLmap exploitation tools which will scan the web server and inject thousand of SQL query into web server.
   Each attack was done 10 times (OWASP 10 times, NOOWASP 10 times) on 3 kinds of Operating System. The result will be show in the Table 2.

TABLE II
SQLMAP EXPLOITATION RESULT

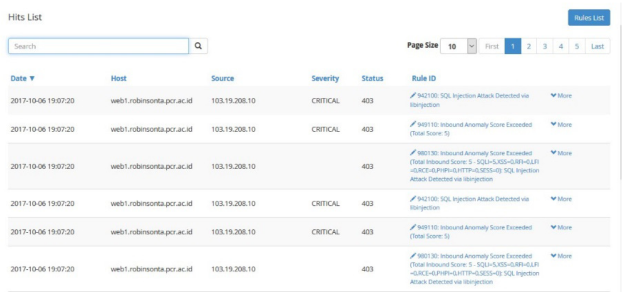
OS	Result	
	OWASP	NOOWASP
Kali Linux	Failed	Success
Back Box	Failed	Success
Parrot	Failed	Success

The result show that OWASP successfully secure web server from all SQLmap exploitations and detect the attack on the Hit List as show in the Figure 7 and 8.



```
[15:22:18] [INFO] testing if the target URL is stable
[15:22:19] [INFO] target URL is stable
[15:22:19] [INFO] testing if GET parameter 'id' is dynamic
[15:22:19] [WARNING] GET parameter 'id' does not appear to be dynamic
[15:22:19] [WARNING] heuristic (basic) test shows that GET parameter 'id' might
not be injectable
[15:22:20] [INFO] testing for SQL injection on GET parameter 'id'
[15:22:20] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[15:22:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[15:22:20] [WARNING] using unescaped version of the test because of zero knowled
ge of the back-end DBMS. You can try to explicitly set it with option '--dbms'
[15:22:21] [WARNING] GET parameter 'id' does not seem to be injectable
[15:22:21] [INFO] testing if GET parameter 'Submit' is dynamic
[15:22:21] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[15:22:21] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' mi
ght not be injectable
[15:22:21] [INFO] testing for SQL injection on GET parameter 'Submit'
[15:22:21] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[15:22:21] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[15:22:29] [WARNING] GET parameter 'Submit' does not seem to be injectable
[15:22:29] [CRITICAL] all tested parameters appear to be not injectable. Try to
increase '--level'/'--risk' values to perform more tests. Also, you can try to r
erun by providing either a valid value for option '--string' (or '--regexp'). If
 you suspect that there is some kind of protection mechanism involved (e.g. WAF)
 maybe you could retry with an option '--tamper' (e.g. '--tamper=space2comment')
[15:22:29] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 176 times

[*] shutting down at 15:22:29

root@kali:~# 
```

Fig. 7 SQLmap detect WAF



Date ▼	Host	Source	Severity	Status	Rule ID	
2017-10-06 19:07:20	web1.robinsonta.pcr.ac.id	103.19.208.10	CRITICAL	403	942100: SQL Injection Attack Detected via libinjection	More
2017-10-06 19:07:20	web1.robinsonta.pcr.ac.id	103.19.208.10	CRITICAL	403	949110: Inbound Anomaly Score Exceeded (Total Score: 5)	More
2017-10-06 19:07:20	web1.robinsonta.pcr.ac.id	103.19.208.10		403	980130: Inbound Anomaly Score Exceeded (Total Inbound Score: 5 - SQLI=5,XSS=0,RFI=0,LFI =0,RCE=0,PHPI=0,HTTP=0,SESS=0): SQL Injection Attack Detected via libinjection	More
2017-10-06 19:07:20	web1.robinsonta.pcr.ac.id	103.19.208.10	CRITICAL	403	942100: SQL Injection Attack Detected via libinjection	More
2017-10-06 19:07:20	web1.robinsonta.pcr.ac.id	103.19.208.10	CRITICAL	403	949110: Inbound Anomaly Score Exceeded (Total Score: 5)	More
2017-10-06 19:07:20	web1.robinsonta.pcr.ac.id	103.19.208.10		403	980130: Inbound Anomaly Score Exceeded (Total Inbound Score: 5 - SQLI=5,XSS=0,RFI=0,LFI =0,RCE=0,PHPI=0,HTTP=0,SESS=0): SQL Injection Attack Detected via libinjection	More

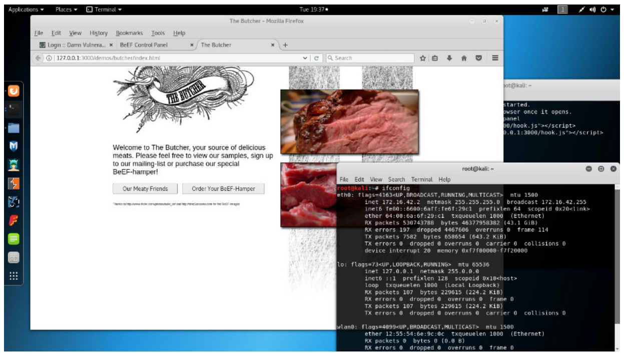
Fig. 8 ModSecurity Hit List

3. BeEF Exploitation
   BeEF is an application for testing cross site scripting stored type which will hook a target PC that will cause the attacker can control the target PC.
   The result of the attacks will be show in the table III.

TABLE III
SQLMAP EXPLOITATION RESULT

OS	Result	
	OWASP	NOOWASP
Kali Linux	Success	Success
Back Box	Success	Success
Parrot	Success	Success

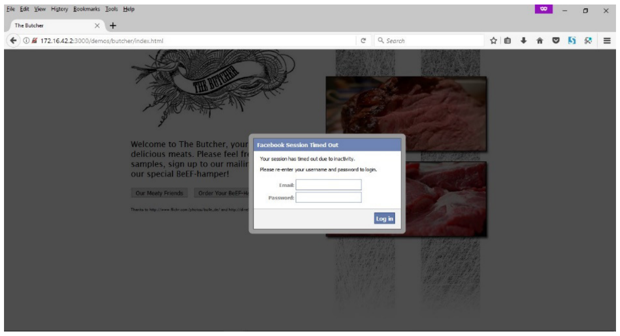
The result show that OWASP failed to secure the web server from XSS (Stored) which cause both of the web server redirecting page into Hook Page that prepared by the Attacker (BeEF) as show on the Figure 9, 10 and 11.



Welcome to The Butcher, your source of delicious meats. Please feel free to view our samples, sign up to our mailing-list or purchase our special BeEF-hamper!

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.42.2  netmask 255.255.255.0  broadcast 172.16.42.255
        inet6 fe80::6600:6aff:fe6f:29c1  prefixlen 64  scopeid 0x20<link>
        ether 64:00:6a:6f:29:c1  txqueuelen 1000  (Ethernet)
        RX packets 530743788  bytes 46377958382 (43.1 GiB)
        RX errors 197  dropped 4467606  overruns 0  frame 114
        TX packets 7582  bytes 658654 (643.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xf7f00000-f7f20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 107  bytes 229615 (224.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 107  bytes 229615 (224.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 12:55:54:6e:9c:0c  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
```

Fig. 9 BeEF Hook Page



Facebook Session Timed Out

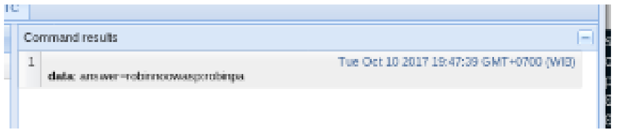
Your session has timed out due to inactivity.

Please re-enter your username and password to login.

Email:

Password:

Fig. 10 BeEF fake form login into target PC



```
Command results
1                                     Tue Oct 10 2017 19:47:39 GMT+0700 (WIB)
data: answer=robinnowaspcrobinpa
```

Fig. 11 BeEF got the username and password from target PC

4. XSSer Exploitation

XSSer is an application for scanning a site page and try to injecting hundred of CSS query.

The result of this attack will be show in the table 4.

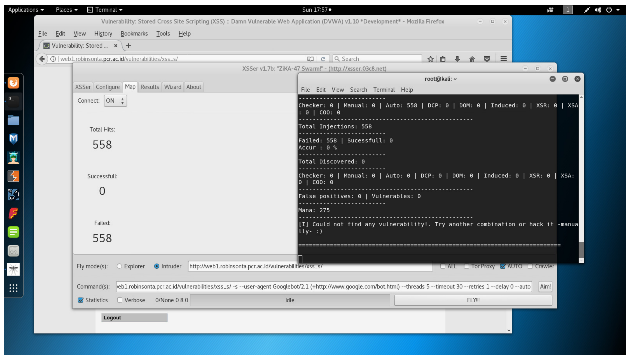
TABLE IV

XSSER EXPLOITATION RESULT

OS	Result	
	OWASP	NOOWASP
Kali Linux	Failed but detected	Failed
Back Box	Failed but detected	Failed
Parrot	Failed but detected	Failed

XSSer application failed to inject CSS query into both of the web server. This may be caused by the Damn Vulnerable Web Application sites isn't show the .php at the URL. XSSer on its work, need the .php at the end of the URL link for injecting CSS script.

Even XSSer was failed to inject CSS Script, but OWASP ModSecurity was success to detect all of XSSer inject action and was display in Hit List as show in the Figure 12 and 13.



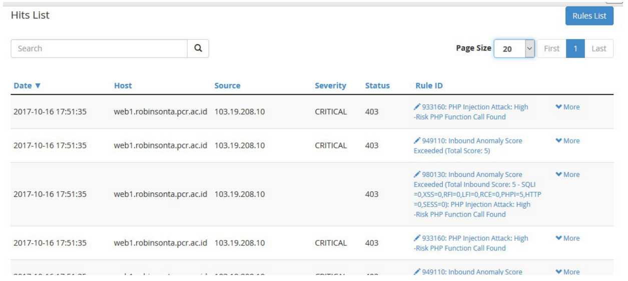
```
-------------------------
Checker: 0 | Manual: 0 | Auto: 558 | DCP: 0 | DOM: 0 | Induced: 0 | XSR: 0 | XSA
: 0 | COO: 0
--------------------------------------------------
Total Injections: 558
-------------------------
Failed: 558 | Sucessfull: 0
Accur : 0 %
-------------------------
Total Discovered: 0
-------------------------
Checker: 0 | Manual: 0 | Auto: 0 | DCP: 0 | DOM: 0 | Induced: 0 | XSR: 0 | XSA:
0 | COO: 0
--------------------------------------------------
False positives: 0 | Vulnerables: 0
-------------------------
Mana: 275
--------------------------------------------------
[I] Could not find any vulnerability!. Try another combination or hack it -manua
lly- :)

============================================================================
```

Total Hits: 558

Successful: 0

Failed: 558

Fig. 12 XSSer failed to inject 558 CSS Query into web server.



Hits List

Date ▼	Host	Source	Severity	Status	Rule ID	
2017-10-16 17:51:35	web1.robinsonta.pcr.ac.id	103.19.208.10	CRITICAL	403	933160: PHP Injection Attack: High -Risk PHP Function Call Found	More
2017-10-16 17:51:35	web1.robinsonta.pcr.ac.id	103.19.208.10	CRITICAL	403	949110: Inbound Anomaly Score Exceeded (Total Score: 5)	More
2017-10-16 17:51:35	web1.robinsonta.pcr.ac.id	103.19.208.10		403	980130: Inbound Anomaly Score Exceeded (Total Inbound Score: 5 - SQLI =0,XSS=0,RFI=0,LFI=0,RCE=0,PHPI=5,HTTP =0,SESS=0): PHP Injection Attack: High -Risk PHP Function Call Found	More
2017-10-16 17:51:35	web1.robinsonta.pcr.ac.id	103.19.208.10	CRITICAL	403	933160: PHP Injection Attack: High -Risk PHP Function Call Found	More
					949110: Inbound Anomaly Score	More

Fig. 13 OWASP ModSecurity Hit List detected XSSer Injection

5. Implementation

After several testing was done to both servers, now the OWASP will be implement into pcr old mahasiswa website. Several attacks also done for testing the security.

The result of the attack testing after implementation will be shown at table V, VI, and VII.

TABLE V

KALI LINUX OS ATTACK RESULT

	Kali Linux	
Attack Type	OWASP	NOOWASP
SQL Injection	Failed	Failed
SQLmap	Failed	Failed
BeEF	Failed	Failed
XSSer	Failed	Failed

TABLE VI

BACKBOX OS ATTACK RESULT

	Back Box	
Attack Type	OWASP	NOOWASP
SQL Injection	Failed	Failed
SQLmap	Failed	Failed
BeEF	Failed	Failed
XSSer	Failed	Failed

TABLE VII

PARROT OS ATTACK RESULT

	Parrot	
Attack Type	OWASP	NOOWASP
SQL Injection	Failed	Failed
SQLmap	Failed	Failed
BeEF	Failed	Failed
XSSer	Failed	Failed

6. Performance Testing

An application call Web Stress were used to simulated 100 clients visit with the conditions below :

- Number of Users are 100 and Test run for 1 minutes.
- 2 Urls that tested : web1.robinsonta.pcr.ac.id & web2.robinsonta.pcr.ac.id

Comparison of the Http Response, Cpu Load, Free Memory and Free Disk of both servers will be display in table VIII:

TABLE VIII
PERFORMANCE RESULT AFTER WEB STRESS SIMULATION

Attribute	Average Result	
	OWASP	NOOWASP
HTTP Response	5.028 Msec	4.046 Msec
CPU Load	0.148 / 5 Minutes	0,13 / 5 Minutes
Free Memory	17% / 171 Mbyte	12% / 121 Mbyte
Free Disk	2.399 MByte	2.162 Mbyte

The result from the table above that convert to chart mode will be show in the Figure 14.

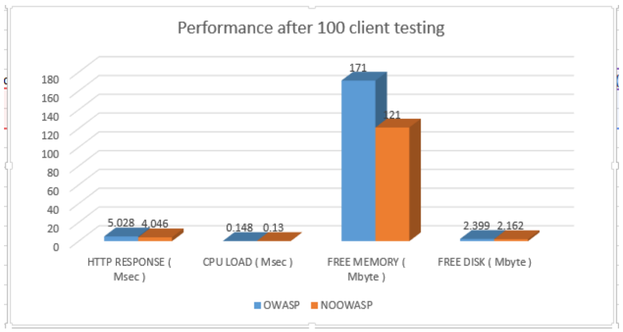


Fig. 14 Performance after 100 client testing.

7. Analyze

a. After testing 7 kinds of SQL Injection on web servers, the result was some attack type already counter or covered by the MySQL patching itself. (based on the results reference to SQL Injection Attack Testing)
b. SQLmap Exploitation sometimes even already detected there was a WAF/IDS/IPS its still force to inject dozens of query that can cause OWASP Hit List flooding. (based on the results reference to SQL Injection using SQLmap Tool)
c. There are 3 type of Cross Site Scripting, DOM, Reflected and Stored type. BeEF is a tool that used to do cross site scripting stored type. Stored type cross site scripting can't been detected by OWASP ModSecurity Web Application Firewall. This may be caused by some web application actually save the information directly into database without filtering the parameter input by user. The solution is filter all the input parameter from user and exception on input form field during Web Application Development time. (based on the results reference to XSS Testing using BeEF Framework Tool).
d. XSSer only can be done for the web application which has shown page extention such as .php, .html , etc . Its can be cover during Web Application Development time by hiding the page extention. (based on the results reference to XSS using XSSer tool)
e. OWASP didn't affect too much the HTTP Response time, CpPU load, Free Memory and Free Disk. (based on the results reference to Performance Comparison & Performance Testing)
f. 100 Clients Performance testing only affect HTTP response time and available memory cause of the testing was only aim port 80 of the web application. (based on the result reference to Performance Testing)

IV. CONCLUSION AND ADVICE

Conclusions of this Final Project are :

1. OWASP ModSecurity success 100% detect and secure web application from SQL Injection after 15 times testing using 3 difference Operating Systems.
2. OWASP ModSecurity failed to secure web application from Cross Site Scripting Stored type that caused the attacker successfully hook target PC.
3. OWASP ModSecurity success 100% detect and secure web application from SQLmap exploitation tools which was done using 3 differences Operating Systems.
4. OWASP ModSecurity success 100% detect and secure web application from XSSer exploitation tools which was done using 3 differences Operating Systems.
5. Comparison result show that there wasn't any big affect that will influence web application performance after using OWASP ModSecurity.

In current research, it still has many shortcomings due to time constraints, budget cost and thoughts of the author. Many things can be studied and developed more deeply. Advice expected for future development includes:

1. Extend security coverage using OWASP ModSecurity Web Application Firewall such as Scanner Detection, DOS Protection, IP Reputation, etc.
2. Extending Core Rule Set of OWASP ModSecurity for securing some cross site scripting attacks that didn't provided by OWASP ModSecurity
3. Research on another WAF and make a comparison of the result with OWASP ModSecurity.

V. ACKNOWLEDGEMENT

We would like to thank Causal Productions for permits to use and revise the template provided by Causal Productions. Original version of this template was provided by courtesy of Causal Productions (www.causalproductions.com).

VI. REFERENCES

[1] BackBox | A Free Open Source Community Project. (n d). Retrieved from Backbox: https://backbox.org
[2] Crhis,A. (2002). Advanced SQL Injection in SQL Server Application. *Software Insigth Security Research (NISR) Publication.*
[3] Conallen, J. (1999). *Building Web Application with UML.* Addison Wesley.
[4] Curphey, M., & Groves, D. (2006). Retrieved from Open Web Application Security Project :

https://www.owasp.org/index.php/OWASP_Modsecurity_Core_Rule_Set_Project

[5] Dharma, M. H. (2011). *Simulasi dan Analisa Keamanan Jaringan Menggunakan Honeypot.*Pekanbaru: Politeknik Caltex Riau.

[6] Endraca, A, King, B., Nodalo, G., Maria, M. S., & Sabas, I.(2013). Web Application Firewall (WAF). *International Journal of e-Educiation, e-Business, e-Management and e-Learning.*

[7] Feri, S. (2014). *Implementasi Firewall Aplikasi Web untuk Mencegah SQL Injection Menggunakan Naxsi.* Yogyakarta : Universitas Islam Negeri Sunan Kalijaga.

[8] Justin, C. (2009). *SQL Injection Attacks and Defense.* Burlington: Syngress Publishing, Inc.

[9] Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution. (n.d.). Retrieved from kali.org: http://kali.org

[10] Mate Vibhakti, T. M. (2014). Building A Secure & Anti-Theft Web Application By Detecting and Preventing Owasp Critical Attacks. *International Journal of Engineering Research and Applications (IJERA).*

[11] Muniz Jospeh, L. A. (2013). *Web Penetration Testing with Linux.* Birmingham, UK: Packt Publishing Ltd.

[12] *Parrot Project.* (n.d.). Retrieved from Parrot OS: www.parrotsec.org

[13] Pedersen, A. (2006). *cPanel User Guide and Tutorial.* Birmingham: Packt Publishing.

[14] Pritchett Willie L, S. D. (2013). *Kali Linux Cookbook.* Birmingham,UK: Packt Publishing Ltd.

[15] Randhe Kirit, M. V. (2012). Defense against SQL Injection and Cross Site Scripting Vulnerabilities. *International Journal of Science and Research (IJSR).*

[16] Rudi, R. (2011). *Membangun Server yang Tahan Terhadap Serangan Brute Force Menggunakan Fail2ban pada Debian 2.0 Squeeze*. Pekanbaru: Politeknik Caltex Riau.

[17] Shah Junaid Latief, K. A. (2014). Cross Site Scripting (XSS) : The dark side of HTML. *International Journal of Engineering and Computer Science*, 4066-4068.

[18] Spett, K. (2005). Cross-Site Scripting. *SPI Dynamic, Inc.*

[19] Syafrizal, M. (2005). *Pengantar Jaringan Komputer.* Yogyakarta: ANDI

[20] Vogt Philipp, N. F. (n.d.). Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis. *Secure Systems Lab Technical University Vienna.*

[21] Wardana, H. K., Novan, S., & Handoko. (2009). Jurusan Teknik Elektro Fakultas Teknik Universitas Kristen Satya Wacana. *Aplikasi Penggunaan Simple Network Management Protocol (SNMP) dalam Jaringan*, 93-108.

[22] Yulianingsih. (2017). Melindungi Aplikasi dari Serangan CrossSite Scripting (XSS) dengan Metode MetaCharacter. *FakultasTeknik dan MIPA Universitas Indraprasta PGRI*, Vol.03 , No.01.