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 Injection, 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. 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 Framework) 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 XSSer) 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:

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.
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>
<thead>
<tr>
<th>Attack</th>
<th>OWASP Result</th>
<th>NOOWASP Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tautology</td>
<td>Failed</td>
<td>Success</td>
</tr>
<tr>
<td>Logically</td>
<td>Success</td>
<td>Success</td>
</tr>
<tr>
<td>Union Queries</td>
<td>Failed</td>
<td>Success</td>
</tr>
<tr>
<td>Piggy-backed</td>
<td>Failed</td>
<td>MySQL countered</td>
</tr>
<tr>
<td>Stored Procedure</td>
<td>Failed</td>
<td>MySQL countered</td>
</tr>
<tr>
<td>Blind Injection</td>
<td>Failed</td>
<td>MySQL countered</td>
</tr>
<tr>
<td>Timing Attacks</td>
<td>Failed</td>
<td>MySQL countered</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>OS</th>
<th>OWASP Result</th>
<th>NOOWASP Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kali Linux</td>
<td>Failed</td>
<td>Success</td>
</tr>
<tr>
<td>Back Box</td>
<td>Failed</td>
<td>Success</td>
</tr>
<tr>
<td>Parrot</td>
<td>Failed</td>
<td>Success</td>
</tr>
</tbody>
</table>

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.

Fig. 7 SQLmap detect WAF

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>
<thead>
<tr>
<th>OS</th>
<th>OWASP Result</th>
<th>NOOWASP Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kali Linux</td>
<td>Failed</td>
<td>Success</td>
</tr>
<tr>
<td>Back Box</td>
<td>Success</td>
<td>Success</td>
</tr>
<tr>
<td>Parrot</td>
<td>Success</td>
<td>Success</td>
</tr>
</tbody>
</table>

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.
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>
<thead>
<tr>
<th>OS</th>
<th>OWASP</th>
<th>NOOWASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kali Linux</td>
<td>Failed but detected</td>
<td>Failed</td>
</tr>
<tr>
<td>Back Box</td>
<td>Failed but detected</td>
<td>Failed</td>
</tr>
<tr>
<td>Parrot</td>
<td>Failed but detected</td>
<td>Failed</td>
</tr>
</tbody>
</table>

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.

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>
<thead>
<tr>
<th>Attack Type</th>
<th>OWASP</th>
<th>NOOWASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kali Linux</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>SQLmap</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>BeEF</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>XSSer</td>
<td>Failed</td>
<td>Failed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>OWASP</th>
<th>NOOWASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back Box</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>SQLmap</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>BeEF</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>XSSer</td>
<td>Failed</td>
<td>Failed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>OWASP</th>
<th>NOOWASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parrot</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>SQLmap</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>BeEF</td>
<td>Failed</td>
<td>Failed</td>
</tr>
<tr>
<td>XSSer</td>
<td>Failed</td>
<td>Failed</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Attribute</th>
<th>Average Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Response</td>
<td>5.028 Msec</td>
</tr>
<tr>
<td></td>
<td>4.046 Msec</td>
</tr>
<tr>
<td>CPU Load</td>
<td>0.148 / 5 Minutes</td>
</tr>
<tr>
<td></td>
<td>0.13 / 5 Minutes</td>
</tr>
<tr>
<td>Free Memory</td>
<td>17% / 171 Mbyte</td>
</tr>
<tr>
<td></td>
<td>12% / 121 Mbyte</td>
</tr>
<tr>
<td>Free Disk</td>
<td>2.399 MByte</td>
</tr>
<tr>
<td></td>
<td>2.162 Mbyte</td>
</tr>
</tbody>
</table>

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

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, Cpu 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

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