International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 17 (2017) pp. 6788-6796
© Research India Publications. http://www.ripublication.com

# Combating Cross-Site Scripting Assaults without Proprietary Software

**Ganga Rama Koteswara Rao**
*Research Scholar, Department of Computer Science & Engineering, Rayalaseema University, Kurnool, Andhra Pradesh, India.*

*Orcid: 0000-0003-0972-1165*

**Dr. R.Satya Prasad**
*Research Guide, Associate Professor, Department of Computer Science & Engineering, Acharya Nagarjuna University, Nagarjuna Nagar, Andhra Pradesh, India.*

**Abstract**

Exploiting the security vulnerabilities in web browsers, web applications and firewalls is a principal characteristic of cross-site scripting (XSS) assaults. Dominate part of web population with essential web attentiveness are vulnerable and even expert web users may not see the assault to have the capacity to react so as to neutralize the ill effects of assault . Because of their unpretentious nature, an exploited server, a compromised browser, a impersonated email or a hacked web application tends to keep this type of assaults alive even in the present circumstances. XSS assaults extremely counterbalance the advantages offered by Web based administrations along these lines affecting the worldwide web group. This paper concentrates on guard, recognition and prevention components to be adopted at a variety of different network doorways to defuse cross-site scripting assaults using non Proprietary Software.

**Keywords:** XSS, Web Application Firewall, Web Server Log Files, ModSecurity.

## INTRODUCTION

Exploiting the security vulnerabilities in internet browsers, web packages and firewalls is the key feature of cross-site scripting assaults. Regularly, inexperienced or unsuspicious users fall prey to these kinds of attacks and even specialists might not be aware the attack so that it will reply in time to neutralize the unwell consequences of assault. Intolerable code can be injected into web pages that may exchange the conduct of the web pages, thieve user statistics, and carry out illegal moves the use of false impersonation or cause method instability [8] which effects the web sites. Attacker website sends a script to a victim that steals data from the device

Cross-site scripting vulnerabilities permits the attacker to insert malicious code into customer browser and execute inside the content of the modern-day software[16]. When the victim visits the attackers web site may get hold of the malicious code and if the victim clicks the link and the script may run on the victim server and send the valuable information to the attacker. Attacker may additionally send the e-mail containing a link to the victim , if the victim clicks the hyperlink that directs the user to attacker managed web page and attacker injects script to borrow cookies which are used for session hijacking[7]. The effect of injected XSS script includes.

- Drives traffic to another site by injecting a link or image.
- Records each keystrokes on the victim machine.
- Steal cookies
- Calls a malicious URL for drive by downloading.
- Exploits browser vulnerabilities.
- Modify the content of the web page(inserting words or images and bad reputation)
- Accessing the local file system.

## RELATED WORK

Kerschbaum, F.[12] implemented a solution that prevents reflected cross-site scripting using a gateway at the server

M.I.P. Salas and E. Martins [16] anticipated Penetration Testing and Fault Injection to emulate XSS attack against Web Services.

Engin Kirdaa,, Nenad Jovanovic, Christopher Kruegel and Giovanni Vigna [3] used Noxes Web application firewall to protect XSS attacks.

Isatou Hydara , Abu Bakar Md. Sultan, Hazura Zulzalil and Novia Admodisastro [6] done literature review on various journals related to XSS attacks and found that there is no single solution to mitigate XSS attack

Jayamsakthi Shanmugam and Dr. M. Ponnavaikko[7] surveyed vulnerabilities with current solutions based on client or server side location, analysis type and intrusion detection type and also focused on strengths and weakness of all approaches.

Mike Ter Louw and V.N. Venkatakrishnan [15] implemented design that turned into integrated with numerous famous web programs. The DESIGN changed into evaluated towards a barrage of stress assessments that exhibit strong resistance to assaults, excellent compatibility with net browsers and reasonable performance overheads.

## INJECTING MALICIOUS CODE METHODS

### Reflected XSS(Non-persistent XSS)

Malicious script does not store in the server. The server bounces the original input from the server to the user[19]. The user inputs are reflected back to the browser. The malicious code is inserted into the data stream but not stored.

***Example for Non-Persistent XSS***

**Table 1 :** student.php script

```
<?php
$name = $_GET['name'];
echo "Student Name : $name …<br>";


echo"<ahref='http://yourchoice.com/student.php?name='
". $name . "> Select the page</a>";


?>
```

When the user clicks on the link , victim system may connect to the malicious site where the malicious code will be execute at the client machine. The damage may be caused at the victimized system which cannot be traced by any of the tools since the victim has deliberately initiated the execution of the malicious code.

### Stored XSS (persistent XSS)

A persistent attack where the data is permanently located in a file system or database. The server stores the malicious code and this injected script will not reflected back, permanently located in the file system or database .

***Example for Persistent XSS***

Most of the websites asks the users to post their feedback in the text box and this message will be stored in the database. If the hacker posts malicious code like tracking session ID cookie and if server fails in validating the input provided by the hacker. Any user having cookies appraisal this message and his account is compromised.

**Table 2:** PHP code for stealing cookie

```
<SCRIPT>
document.location=
'http://hackerhost/cgi-
bin/cookiesteal.cgi?'+document.cookie
</SCRIPT>
```

## DOCUMENT OBJECT MODEL

This attack appears in Document Object Model. DOM based attack will happen when the Javascript in the page accesses a URLparameter and uses this information to write HTML to the page[3].It has been observed that more than the 50% of the web sites are vulnerable to DOM based XSS attack.

## CROSS SITE SCRIPTING DEFENSE

Security in general can never be fool proof, there is always vulnerability. There is no guarantee that the system won't be attacked. Today there is a need to intercept all type of communication including SSL traffic (HTTPS). It is necessary to check if the system is under attack. System may be under attack, if the huge number of request coming from internet for a particular web service.

- Defense mechanism at Client side
  - Disable Java Script: Client browsers can be configured to modify their security decisions based on properties of the script such as its origin, execution period, frequency of interacting with client etc.
  - Verify email: Email attachments or links sent via emails are the most prevalent mechanism for the spread of XSS. It is imperative that before a user clicks on an attachment / link in an email, the user is completely sure that the sender's credentials are correct and that the message itself is authentic.
  - Always update: Browser, Operating System, Antivirus and relevant software have to be kept up to date in order to be able to detect ever

growing threats. These software are themselves under constant improvement therefore enhancing the risk of exposing newer vulnerabilities

- Defense mechanism at Server side
  - Input validation (Black listing VS White listing): With the expanse of web applications, customizing access policies are necessity on a per-case basis at application level. Rules can be framed to allow particular forms of requests, originating from specific clients etc (White listing) or disallowing particular type of requests, originating from specific clients etc (Black listing).
  - Encode all meta characters send to the client: To avoid being victimized by XSS attacks such as Non-persistent (Reflective) XSS, all outgoing data can be encoded using html meta characters which offers a certain degree of defense against this type of attacks.
  - keep track of user sessions: XSS attacks frequently attempt to hijack user sessions. As user session management is the responsibility of the web server, the rules and protocols to be used, their validity periods etc. can be strengthened to check XSS attacks.
  - Web application firewall: Tailor made rule sets for individual web applications are the best mechanisms to defend web applications from XSS attacks to cope up with the increasing range of web applications and the scope of web services being offered by them.
  - Always test: As newer web applications are developed, proper and thorough testing against all known and possible types of XSS vulnerabilities is the best practice to insulate a web application from all three forms of XSS attacks.

## DETECTION

XSS attacks are detected at application layer, because web applications run at this layer. This paper lays emphasis on detecting XSS attacks in the following locations of the network.

- Web Application Firewall
- Web Server Log Files

### DETECTION AT WEB APPLICATION FIREWALL

Typical firewalls examine traffic based on protocols like TCP, UDP and ICMP, because they operate at network layer and transport layer[9] .TCP/IP was not initially proposed to be utilized for business and person to person communication applications and administrations. Responsibility was not a noteworthy concern and secrecy became de facto identity for Internet and its applications such as WWW. As the number and scope of applications of the Internet grew, it became evident that normal firewall security is inadequate. It is important to secure individual web applications from vulnerabilities existing in TCP/IP. Web application server traffic can be controlled by defining rules for decision making applying specific filters. Since this firewall operates at the web application server level, it is the best place for detecting XSS attacks.

### *DETECTION AT WEB SERVER LOG FILES*

Web Servers store a lot of information regarding incoming requests and outgoing responses in the form of web server logs**.** The log data contains user activities and confidential information about the user[4]. However, even the web server logs tend to omit certain critical pieces of information such as data sent to HTTP header and also due to the volume of data being served by the web server; the logs can contain information about only a partial set of outgoing web traffic.

Specific details of problems pertaining to the website can emerge by probing key areas in the server logs. Following are the information for extracting information from the log files.

- Most and least traffic on the web pages.
- Pages viewed by the visitor
- Details of the visitor(domain name and request of the visitor)
- Request’s date and time.
- Method invoked, URL requested and number of bytes transferred.

### *ALGORITHM TO ANALYZE WEBSERVER LOG FILES*

```
For each request do
 If visitor’s domain name is in black_listed_domains_list then
         Send alert to server administrator
          Exit
End if
Identify the number of hits on each web page
If number of hits on a particular web page exceeds the
```

```
specified limit then
        Identify visitor's location (IP Address, Domain name)
        Add visitor to black_listed_domains_list
        Send alert to server administrator
        Exit
End if
Identify the total bytes transferred for particular IP.
If total no of bytes exceeds maximum limit then
        Add IP Address to black_listed_domains_list
        Send alert to server administrator
End if
End
```

## PREVENTION

Older browsers had little or no vulnerabilities as their function was limited to make http request, receive html response and render the output for the user's view. As WWW became more popular, the functionality of web browsers also enhanced. Programmability of web browsers and extensibility is a common feature of all modern browsers. This exposes the browsers to hackers who tend to exploit this feature as vulnerability[17]. It is very complex to mitigate XSS attacks such as persistent XSS due to this very vulnerability. Poor programming in developing web sites also helps the attacker to exploit their malicious scripts and inject code into the user's browser[13]. There is a need to use filtering mechanism at different networking devices in the network. This paper focus on two methods for preventing the XSS attacks.

- Web Application Firewall
- Code Filtering

### Web Application Firewall

Web Application Firewall rules are defined based on the security policies to identify and block malicious code. It examines the payload to detect malicious traffic in addition to examining IP address, Protocol and port number. Incoming user data via post, get and cookie values are disinfected. Requests to URLs not defined in filtering rule sets are disallowed.

Web application firewall is one method to prevent the attack; we must have a clear understanding of the operation of the web application and the services offered by it and configure the firewall carefully. Firewall filters HTTP traffic based on the rules defined[2]. Web application firewall operates at the application layer, identifies and blocks XSS attacks. The blocking rule is placed in the firewall configuration file.

Implementing the web application firewall rules helps to

- Block malicious and inappropriate traffic.
- Examines all inbound and outbound traffic for malicious Redirect URLs and Malicious JS Payloads.
- Stop the IP address of the user that tries to attack the website.
- Expert tuning and configuration management
- Ongoing performance and availability management
- Checks outgoing HTTP responses and verify that it stops the attack

### *Mod Security*

ModSecurity is an web application firewall (WAF) module which is an open source and cross-platform. It filters incoming and outing going data to stop malicious traffic and also analyze and detect malicious traffic. All the details of the malicious traffic are recorded in the log file and alert messages are sent to the administrator. It defines rules on HTTP data such rules will have access to full HTTP header information[1].

Operating as an Apache Web server module, the purpose of ModSecurity is to increase web application security, protecting web applications from known and unknown attacks.

Modsecurity prevents attacks by monitoring HTTP traffic based on rules define in configuration file. If the modsecurity is not configured properly it cannot detect the attacks. The detection of attacks are mainly based on the configuration.

*Steps involved in Configuring the Web Application Firewall*

Set the mod_security module, logging functions, cookie parser version and folder for persistent storage.

Set SecFilterScanPost ON for retrieve request payload

Verify the encoding of URL request by setting SecFilterCheckURLEncoding ON

Set the SecAuditEngine ON to log all HTTP requests.

Set the location of the log file with SecAuditLog /log/mod.log

Allows scanning of POST request with SecFilterScanPOST

Reject all invalid requests with status 403 by setting SecFilterDefaultAction "deny,log,status:403".

```
        Disallows directory traversal with SecFilter "\.\./"
        Deny all request containing /bin/bash string with SecFilter /bin/bash
```

*Algorithm for blocking Cross-Site Scripting assaults using modsecurity firewall*

The modsecurity webserver firewall examines REQUEST_HEADER:Referer, REQUEST_URI and REQUEST_METHOD

```
for each request do
        if (REQUEST_HEADER:Referer contains
                blacklisteddomainname) then
                Block the request.
                Store the requester_info in the log file.
        endif
        if(REQUEST_URI  contains “script”|”alert”|script
            methods) then
                Block the request.
                Send the alert message to the administrator.
        end if
        If(REQUEST_METHOD does not match the pattern
          “\^(GET|HEAD)$”) then
          Block the values of the web page
          Restricts request method to either GET or HEAD
          Send  alert to administrator
        Endif
End
```

**Code Filtering**

The Cross-site scripting vulnerability occurs due to unseemly refining of user inputs[6]. To prevent XSS attacks always validate input fields. The input filtering should uncover Cross-site scripting vulnerabilities in web applications[14]. There are functions like htmlspecialchars() and strip_tags() in PHP to block XSS attacks.

There are two approaches to filter XSS attacks that are input and output filtering. The main data like URL, HTTP referrer objects, GET and POST parameters from a form, Window.location, document.referrer, document.location must be properly filtered before being used on websites because users’ data without proper validation leads to XSS attack[18]. The input filtering ensures proper web application behaviour[20].

***Algorithm for filtering from XSS attacks***

```
    for each request do
      if(request method is GET) then
          Examine the URL string for malformed characters.
              if(malformedcharacter are found) then
                  reject the request
              endif
      endif
      if(request method is POST) then
          Examine the protocol in the request.
              if( request protocol is not secured) then
                  block the request
              else
                  Process the SSL.
              endif
                  examine the URL string
              if(input parameter contains script methods)
              then
                  reject the request
              endif
      endif
if (input contains
 Window.location|Document.referrer|document.location) then
      Reject the Input
endif
if(input contains CookieName.getValue != “ “) then
      CookieName , setValue = “”;
Endif
If(input contains Headerdata) then
Block the header
endif
end
```

***Output Filtering***

Output filtering is done at server side before sending to client browser which is similar to input filtering. There is a need to ensure that the special characters are not filtered at output filtering. The user data need to be filtered such as encoding the characters before sending it back to the browser[10]. There are two ways to set encoding, response header and meta tags. Don't let the web browser guess at a web pages content encoding[15].

## RISKS FROM XSS ATTACKS

According to a survey by Web Application Security Consortium, over sixty percent of websites are prone to XSS attacks. Only a keen and perceptive perspective can reveal a XSS attack victim[12]. The target machine(s) and the software is the major resource for computing system of attack[11]. Confirming round the clock continuous service to the clients becomes a top priority to the organization [5]. Hackers have been exploiting the cross site scripting vulnerability right from the development of scripting languages which aid in enhancing the web users' browsing experience.

**Table 3 :** Analysis from security firms on XSS

| Security Firm | Analysis |
|---|---|
| Trust wave's Spiderlabs | Websites vulnerable to cross site scripting attacks > 80% |
| CWE by MITRE(Common Weakness Enumeration) | XSS is one of the most prevailing, fixed, and risky vulnerabilities found in websites. |
| WhiteHat Security | XSS notorious for being the most common vulnerability of websites |

The vulnerabilities found in the application layer are XSS, Content Injection, Cross-Site Request Forgeries (CSRF), Session Management, Information Leakage, Authentication, SQL Injection and Other Injection. The vulnerabilities discovered by Edgescan[21] in 2014 for web application layer is shown in figure below.

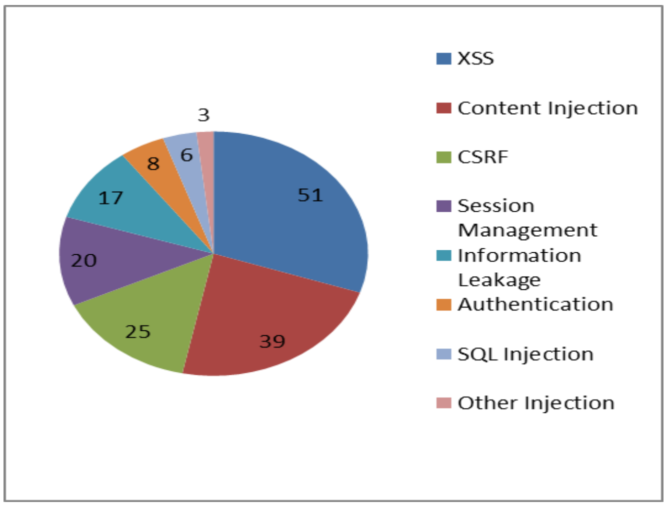


**Figure 1:** Vulnerabilities discovered in 2014 in web application layer

All public and private establishments regardless of the nature of their operations have taken to web enabled information technology platforms. IT services have become the de facto standard for employees and their employers to perform their regular functions. Social networking is set to revolutionize the way everybody interacts with everybody else. The way to conduct one's business is fast becoming online. XSS attack impact on various segments employing web enabled IT services is shown below.

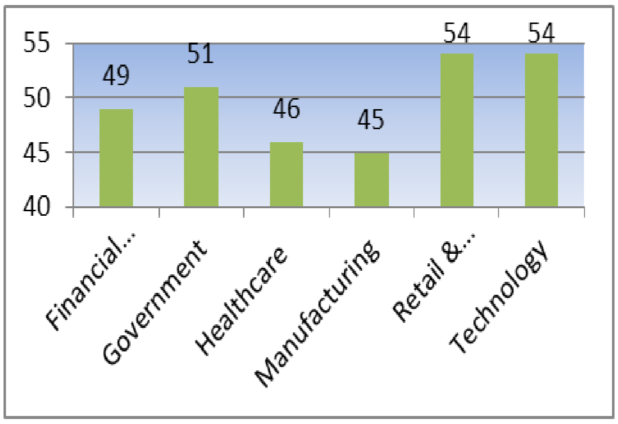


**Figure 2:** The impact of cross site scripting attacks on different organizations

Web applications security vulnerability depends both on the programming language chosen as well as the deployed platform. Any combination of the web technologies employed by the organizations is not completely immune to security threats. The major percentage of software platforms used in the Manufacturing, Healthcare, Government and Financial Services are .NET and Java. A segment wise view of the web technologies employed by these sectors is shown in the figure below.

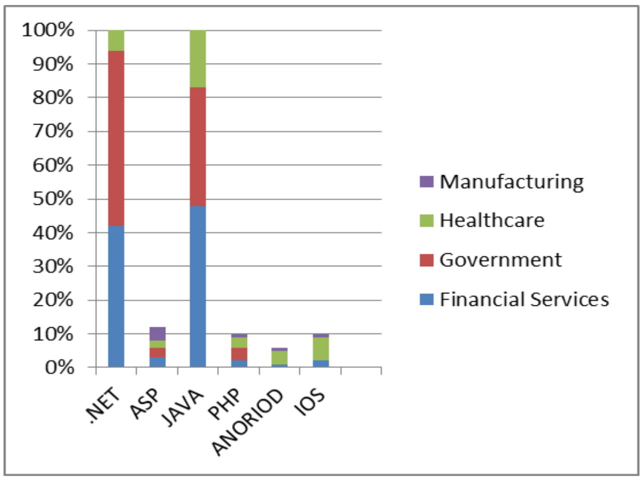
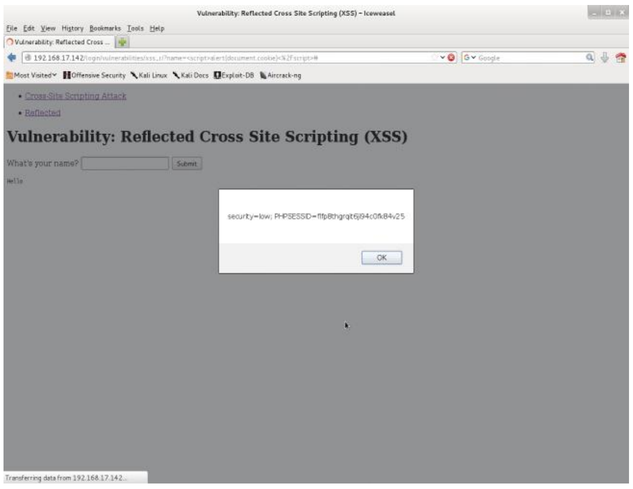


**Figure 3:** Percentage of the software used by different organization web sites

## RESULTS

Cross-site scripting vulnerabilities are misused on the site. Cross-site scripting vulnerabilities empowers an assailant to target different clients of the application, conceivably accessing their information, performing unapproved activities for their benefit, or doing different assaults against them.

The script is injected into the name parameter, the browser will execute the script which generates the popup window as shown in the figure 4 below.


Vulnerability: Reflected Cross Site Scripting (XSS) – Iceweasel

192.168.17.142/login/vulnerabilities/xss_r/?name=<script>alert(document.cookie)<%2Fscript>#

- Cross-Site Scripting Attack
- Reflected

**Vulnerability: Reflected Cross Site Scripting (XSS)**

What's your name? Submit

Hello

security=low; PHPSESSID=flfp8thgrqit6j94c0fk84v25

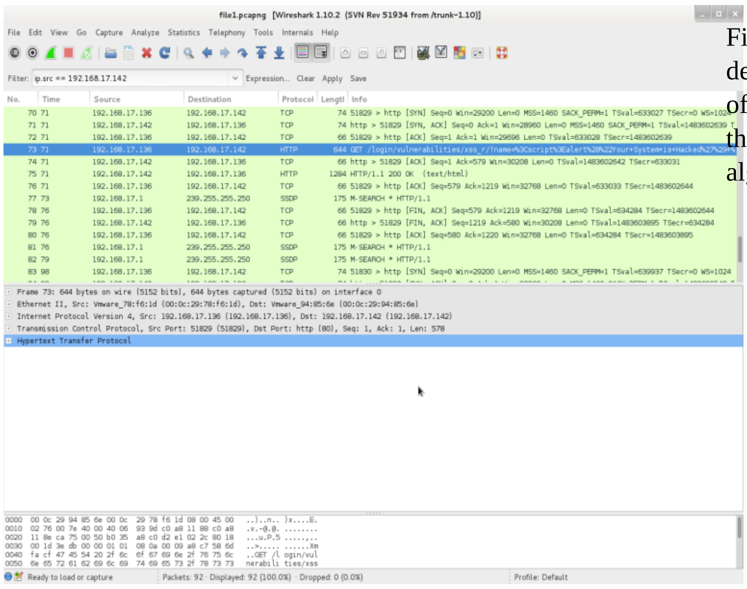
OK

Transferring data from 192.168.17.142...

File Edit View History Bookmarks Tools Help
Most Visited Offensive Security Kali Linux Kali Docs Exploit-DB Aircrack-ng

**Figure 4:** Execution of Reflective Cross site scripting attack

The attack can be captured by the wireshark as shoen in figure 5 below. The packet no 73 shows that the malicious script is injected with the GET request. The packet no 75 shows that the request is accepted and runs the malicious script.


file1.pcapng [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]

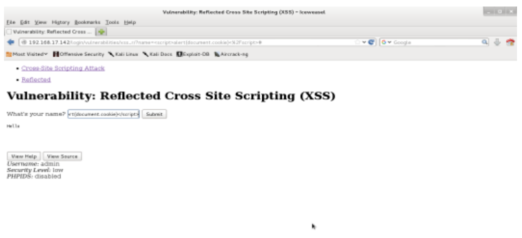
Filter: ip.src == 192.168.17.142

```
No.    Time  Source          Destination      Protocol Length Info
 70 71  192.168.17.136  192.168.17.142   TCP       74 51829 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=633027 TSecr=0 WS=1024
 71 71  192.168.17.142  192.168.17.136   TCP       74 http > 51829 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1483602639 T
 72 71  192.168.17.136  192.168.17.142   TCP       66 51829 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=633028 TSecr=1483602639
 73 71  192.168.17.136  192.168.17.142   HTTP     644 GET /login/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22Your+System+is+Hacked%27%29+
 74 71  192.168.17.142  192.168.17.136   TCP       66 http > 51829 [ACK] Seq=1 Ack=579 Win=30208 Len=0 TSval=1483602642 TSecr=633031
 75 71  192.168.17.142  192.168.17.136   HTTP    1284 HTTP/1.1 200 OK  (text/html)
 76 71  192.168.17.136  192.168.17.142   TCP       66 51829 > http [ACK] Seq=579 Ack=1219 Win=32768 Len=0 TSval=633033 TSecr=1483602644
 77 73  192.168.17.1    239.255.255.250  SSDP     175 M-SEARCH * HTTP/1.1
 78 76  192.168.17.136  192.168.17.142   TCP       66 51829 > http [FIN, ACK] Seq=579 Ack=1219 Win=32768 Len=0 TSval=634284 TSecr=1483602644
 79 76  192.168.17.142  192.168.17.136   TCP       66 http > 51829 [FIN, ACK] Seq=1219 Ack=580 Win=30208 Len=0 TSval=1483603895 TSecr=634284
 80 76  192.168.17.136  192.168.17.142   TCP       66 51829 > http [ACK] Seq=580 Ack=1220 Win=32768 Len=0 TSval=634284 TSecr=1483603895
 81 76  192.168.17.1    239.255.255.250  SSDP     175 M-SEARCH * HTTP/1.1
 82 79  192.168.17.1    239.255.255.250  SSDP     175 M-SEARCH * HTTP/1.1
 83 98  192.168.17.136  192.168.17.142   TCP       74 51830 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=639937 TSecr=0 WS=1024

Frame 73: 644 bytes on wire (5152 bits), 644 bytes captured (5152 bits) on interface 0
Ethernet II, Src: Vmware_78:f6:1d (00:0c:29:78:f6:1d), Dst: Vmware_94:85:6e (00:0c:29:94:85:6e)
Internet Protocol Version 4, Src: 192.168.17.136 (192.168.17.136), Dst: 192.168.17.142 (192.168.17.142)
Transmission Control Protocol, Src Port: 51829 (51829), Dst Port: http (80), Seq: 1, Ack: 1, Len: 578
Hypertext Transfer Protocol

0000  00 0c 29 94 85 6e 00 0c  29 78 f6 1d 08 00 45 00   ..)..n.. )x....E.
0010  02 76 00 7e 40 00 40 06  93 9d c0 a8 11 88 c0 a8   .v.~@.@. ........
0020  11 8e ca 75 00 50 b0 35  a8 c0 d2 e1 02 2c 80 18   ...u.P.5 .....,..
0030  00 1d 3e db 00 00 01 01  08 0a 00 09 a8 c7 58 6d   ..>..... ......Xm
0040  fa cf 47 45 54 20 2f 6c  6f 67 69 6e 2f 76 75 6c   ..GET /l ogin/vul
0050  6e 65 72 61 62 69 6c 69  74 69 65 73 2f 78 73 73   nerabili ties/xss

Ready to load or capture   Packets: 92 · Displayed: 92 (100.0%) · Dropped: 0 (0.0%)   Profile: Default
```

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

**Figure 5:** Reflective XSS attack is examined at the Packet No 73 & 75

After implementing the algorithm 6.1.3 on modsecurity web application firewall the following results are observed. The script is injected in the name parameter as shown in figure 6 below


Vulnerability: Reflected Cross Site Scripting (XSS) – Iceweasel

192.168.17.142/login/vulnerabilities/xss_r/?name=<script>alert(document.cookie)<%2Fscript>#

- Cross-Site Scripting Attack
- Reflected

**Vulnerability: Reflected Cross Site Scripting (XSS)**

What's your name? rt(document.cookie)</script> Submit

Hello

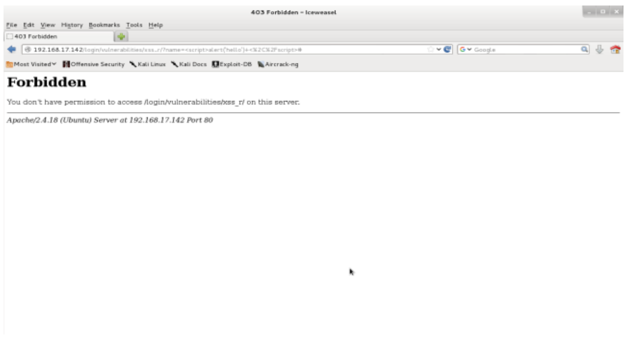
View Help View Source

*Username:* admin
*Security Level:* low
*PHPIDS:* disabled

File Edit View History Bookmarks Tools Help
Most Visited Offensive Security Kali Linux Kali Docs Exploit-DB Aircrack-ng

**Figure 6:** Java Script is injected in the name parameter

Algorithm for blocking Cross-Site Scripting assaults using modsecurity firewall implemented in modsecurity web application firewall blocks the script running in the browser, the browser will execute "Forbidden" error code as shown in figure below.


403 Forbidden – Iceweasel

192.168.17.142/login/vulnerabilities/xss_r/?name=<script>alert('hello')+<%2C%2Fscript>#

**Forbidden**

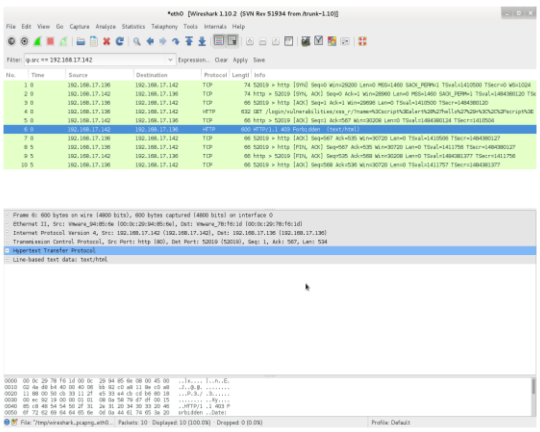
You don't have permission to access /login/vulnerabilities/xss_r/ on this server.

*Apache/2.4.18 (Ubuntu) Server at 192.168.17.142 Port 80*

File Edit View History Bookmarks Tools Help
Most Visited Offensive Security Kali Linux Kali Docs Exploit-DB Aircrack-ng

**Figure 7:** Browser with the Cross-site scripting overcrowding rule

Figure 8 beneath indicates wireshark catch the assault. It demonstrates that the GET request was sent with the injection of cross site script in packet #4. Then, the packet #6 shows that the request is forbidden with help of the defence algorithm.


*eth0 [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]

Filter: ip.src == 192.168.17.142

```
No.  Time Source          Destination     Protocol Length Info
 1 0  192.168.17.136  192.168.17.142  TCP    74 52019 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1410500 TSecr=0 WS=1024
 2 0  192.168.17.142  192.168.17.136  TCP    74 http > 52019 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1484380120 TSe
 3 0  192.168.17.136  192.168.17.142  TCP    66 52019 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1410500 TSecr=1484380120
 4 0  192.168.17.136  192.168.17.142  HTTP  632 GET /login/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%27hello%27%29+%3C%2C%2Fscript%3E
 5 0  192.168.17.142  192.168.17.136  TCP    66 http > 52019 [ACK] Seq=1 Ack=567 Win=30208 Len=0 TSval=1484380124 TSecr=1410504
 6 0  192.168.17.142  192.168.17.136  HTTP  600 HTTP/1.1 403 Forbidden  (text/html)
 7 0  192.168.17.136  192.168.17.142  TCP    66 52019 > http [ACK] Seq=567 Ack=535 Win=30720 Len=0 TSval=1410506 TSecr=1484380127
 8 5  192.168.17.136  192.168.17.142  TCP    66 52019 > http [FIN, ACK] Seq=567 Ack=535 Win=30720 Len=0 TSval=1411756 TSecr=1484380127
 9 5  192.168.17.142  192.168.17.136  TCP    66 http > 52019 [FIN, ACK] Seq=535 Ack=568 Win=30208 Len=0 TSval=1484381377 TSecr=1411756
10 5  192.168.17.136  192.168.17.142  TCP    66 52019 > http [ACK] Seq=568 Ack=536 Win=30720 Len=0 TSval=1411757 TSecr=1484381377

Frame 6: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits) on interface 0
Ethernet II, Src: Vmware_94:85:6e (00:0c:29:94:85:6e), Dst: Vmware_78:f6:1d (00:0c:29:78:f6:1d)
Internet Protocol Version 4, Src: 192.168.17.142 (192.168.17.142), Dst: 192.168.17.136 (192.168.17.136)
Transmission Control Protocol, Src Port: http (80), Dst Port: 52019 (52019), Seq: 1, Ack: 567, Len: 534
Hypertext Transfer Protocol
Line-based text data: text/html

[illegible]

File: "/tmp/wireshark_pcapng_eth0... Packets: 10 · Displayed: 10 (100.0%) · Dropped: 0 (0.0%)   Profile: Default
```

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

**Figure 8:** Blocking of Reflective XSS attack is examined at the Packet No 6

## CONCLUSION

Web applications need to convey web administrations day and night to meet the regularly developing worldwide interest for online administrations over the Internet. XSS assaults utilize known vulnerabilities in existing Internet tools and advances to upset web administrations affecting their clients. All XSS assault strategies have been broke down. Hazard impact factors of XSS assaults on different public and private sectors have been assessed. This paper concentrates on distinguishing and counteracting XSS assaults utilizing web server log records, web application firewalls and code filtering

Defusing Cross-Site Scripting assaults is not a paltry assignment. Mechanism for mitigating the XSS assault has been monitored / identified at various areas, for example, web server log files, web application firewalls and web server requests/responses . Web Server log file has been utilized to distinguish the assault. ModSecurity web application firewall and code filtering has been utilized for both discovery and avoidance of assault.

## REFERENCES

[1] Chung-Huang Yang and Chung-Hsiang Shen, “IMPLEMENT WEB ATTACK DETECTION ENGINE WITH SNORT BY USING MODSECURITY CORE RULES” , The E-Learming and Information Technology Symposium Tainan, TAIWAN, 1 April, 2009

[2] David Gillman et al., “Protecting Websites from Attack with Secure Delivery Networks”, Computer, vol.48, pp. 26-34, Apr. 2015.

[3] Engin Kirda, Nenad Jovanovich, Christopher Kruegel and Giovanni Vigna, “Client-Side Cross-Site Scripting Protection”, Computers and Security Journal, Elsevier, Vol: 28, No: 7, 2009

[4] Dr. G. Rama Krishna, Professor, P.MounikaV and P. Krishna Anne, “Providing Privacy through Search Logs”, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 9, Number 19 (2014) pp. 5569-5582

[5] G.Rama koteswara Rao and A.Pathanjali Sastri "Neutralizing DoS attacks on Linux Servers“ at Networks and Communications (NetCom2013), Lecture Notes in Electrical Engineering Volume 284, 2014, pp 281-293, Springer International Publishing, DOI : 10.1007/978-3-319-03692-2-23 .

[6] Isatou Hydara et al., “Current state of research on cross-site scripting (XSS) – A systematic literature review”, Information and Software Technology 58 (2015) 170–186, Elsevier

[7] Jayamsakthi Shanmugam and Dr. M. Ponnavaikko, “Cross Site Scripting-Latest developments and solutions: A survey”, Int. J. Open Problems Compt. Math., Vol. 1, No. 2, September 2008, pp. 8-28

[8] Jose Fonseca et al., “Analysis of Field Data on Web Security Vulnerabilities”, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 2, MARCH/APRIL 2014

[9] J. Stuart Broderick, “Firewalls e Are they enough protection for current networks?”, Information Security Technical Report (2005) 10, pp. 204-212, Elsevier.

[10] Jyoti Snehi and Dr. Renu Dhir, “Web Client and Web Server approaches to Prevent XSS Attacks”, International Journal of Computers & Technology, Volume 4 No. 2, March-April, 2013, ISSN 2277-3061, pp. 345-352.

[11] K Raja Sekhar , L S S Reddy, U. J. Kameswari. "Secure System of Attack Patterns towards Application Security Metric Derivation." International Journal of Computer Applications , volume 53-No.1,September 2012

[12] Kerschbaum, F. “Simple cross-site attack prevention”, Third International Conference on Security and Privacy in Communication Networks,IEEE,2007

[13] Lwin Khin Shar and Hee Beng Kuan Tan, “Automated removal of cross site scripting vulnerabilities in web applications”, Information and Software Technology 54 (2012) ,Elsevier, pp. 467–478

[14] Lwin Khin Shar and Hee Beng Kuan Tan, “Defending against Cross-Sit Scripting Attacks”, Computer, vol.45, pp. 55-62, March 22.

[15] Mike Ter Louw and V.N. Venkatakrishnan, “BLUEPRINT: Robust Prevention of Cross-site Scripting Attacks for Existing Browsers”, 30th IEEE symposium on Security and Privacy, Oakland, May, 2009.

[16] M.I.P. Salas and E. Martins, “Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security”, Electronic Notes in Theoretical Computer Science 302 (2014) 133–154, Elsevier

[17] Phu H. Phung et al., “Between Worlds: Securing Mixed JavaScript/ActionScript Multi-Party Web Content”, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 12, NO. 4, JULY/AUGUST 2015

[18] Puspendra Kumar and R.K. Pateriya, “DWVP: Detection of Web Application Vulnerabilities using

Parameters of Web Form", Proceedings of Joint International Conferences on CIIT 2013 and itSIP 2013 sponsored by ISTE

[19] Tawfiq S. Barhoom and Sarah N. Kohail , "A New Server-Side Solution for Detecting Cross Site Scripting Attack", International Journal of Computer Information Systems, Vol. 3, No. 2, 2011, pp. 19-23.

[20] Theodoor Scholte et al., "Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis", IEEE 37th Annual Computer Software and Applications Conference,2013.

[21] https://www.edgescan.com/assets/docs/BCC001%20Edgescan%20Stats%20Report_WEB2.pdf