Analysis of Browser Defenses against XSS Attack Vectors

Shital Dhamal
Department of Computer Engineering
Lokmanya Tilak College of Engineering
Koparkhairne,Navi Mumbai ,Maharashtra,India

Manisha Mathur
Department of Computer Engineering
Lokmanya Tilak College of Engineering
Koparkhairne,Navi Mumbai ,Maharashtra,India

ABSTRACT

With the up gradation of technology came World Wide Web and now it has become part of our everyday life. Our increasing dependency on web applications has made us more susceptible to web based attacks .According to OWASP [1] (Open Source Web Application Security Project) Structured Query Language (SQL) injection, Cross Site Scripting Attack (XSS) and Cross-Site Request Forgery (CSRF) are the most popular attack techniques used by evil-minded user for monetary gains or in some way harm the unsuspecting user. Cross site scripting has been on top of the list of web security threats of late. To deal with the cross site scripting on server side is not always possible because of security unawareness of web developers. Hence it becomes imperative to implement client side defenses. In this paper we are going to assess the defenses of existing browsers and study their limitations. For analyzing the defenses provided by different browsers we have created detailed test cases of vulnerabilities and designed a vulnerable web site for testing the browsers capability to resist against the exploits.

General Terms

Browser Security

Keywords

Cross Site Scripting, XSS, Browser Security

1. INTRODUCTION

Cross Site Scripting attacks exploits the user's trust in website [2]. Cross-site scripting (XSS) [3] is one of the most prevalent and dangerous vulnerabilities in web applications. XSS are those attacks against web applications in which an intruder gets control of a user’s browser in order to execute a malicious script (usually an HTML/JavaScript code) within the trusted web site and the victim remains gullible of attackers malignant intentions. As a result, if the embedded code is successfully executed, the attacker might then be able to access any sensitive browser information associated to the web application (e.g., cookies, session IDs, etc.). Dealing with the cross site scripting on server side is not completely possible because web developers are not completely aware of security. Hence client side defenses are implemented by most of the browsers. In this paper the defenses of existing browsers are evaluated like Internet Explorer 8, Firefox, Google Chrome and Safari. To test the defense mechanisms of various browsers against XSS attacks. A vulnerable web application with no defense mechanism has been created which can be easily exploited.

This paper is divided into following sections: Section 2 describes the three major categories of XSS Attacks. Section 3 describes the defenses presented by different browsers for different XSS attack vectors, Section 4 presents the results of evaluation, Section 5 mentions the related work, section 6 concludes this paper, Section 7 presents the future scope and then referenced materials are listed.

2. TYPES OF XSS ATTACKS

2.1 Reflected XSS Attack:

Among the different type of XSS attacks reflected XSS is by far the most common type. This type of vulnerability is exploited when the input provided by a client is processed by server-side scripts without properly verifying or sanitizing the input. In addition the server should reflect this input along with some other data as HTTP response. A very common example is a site search engine. A search engine returns the input given along with the search results. If the input given is a script then the same script is reflected by the server resulting in a reflected XSS .When a attacker inject his mischievous script into a search query, a search box, or the end of an url , it's called Reflected XSS Attack as the script gets executed and is reflected along with the server response.

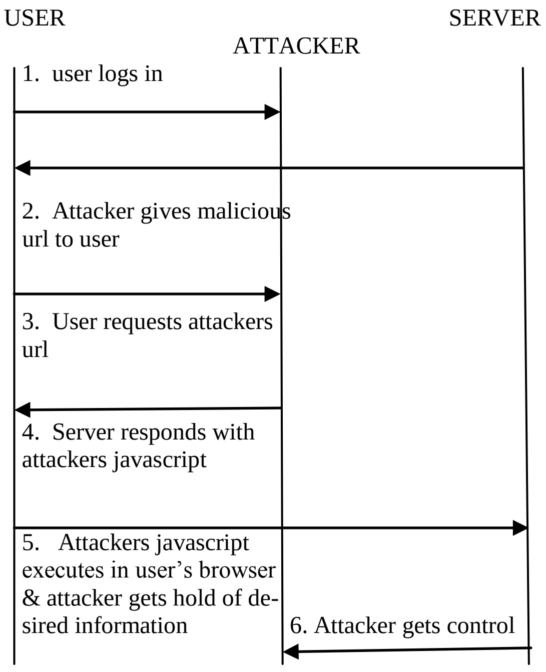
USER
ATTACKER
SERVER
1. user logs in
2. Attacker gives malicious url to user
3. User requests attackers url
4. Server responds with attackers javascript
5. Attackers javascript executes in user’s browser & attacker gets hold of de-sired information
6. Attacker gets control

Fig 1 The steps involved in Reflected XSS attack

2.2 Stored XSS Attack or Persistent XSS Attack:

Stored XSS attack is the most destructive of all cross site scripting attacks. Message boards that allow users to post messages provide breeding ground for persistent XSS. A hacker just has to submit XSS exploit code which is designed in a way to cause harm to an area of a web site that is likely to be visited by other users [5]. These areas could be blog comments, user reviews, message board posts, chat rooms, HTML e-mail and numerous other locations. When a user visits the infected web page, the code is automatically executed. This makes persistent XSS much more dangerous than non-persistent or DOM-based because the user has no means of safeguarding himself. Stored XSS attack can effect multiple users. Once a hacker has his exploit code in place who ever visits his page or view the blog will unknowingly get infected. Refer to Figure 2 for the steps involved in stored XSS attack.

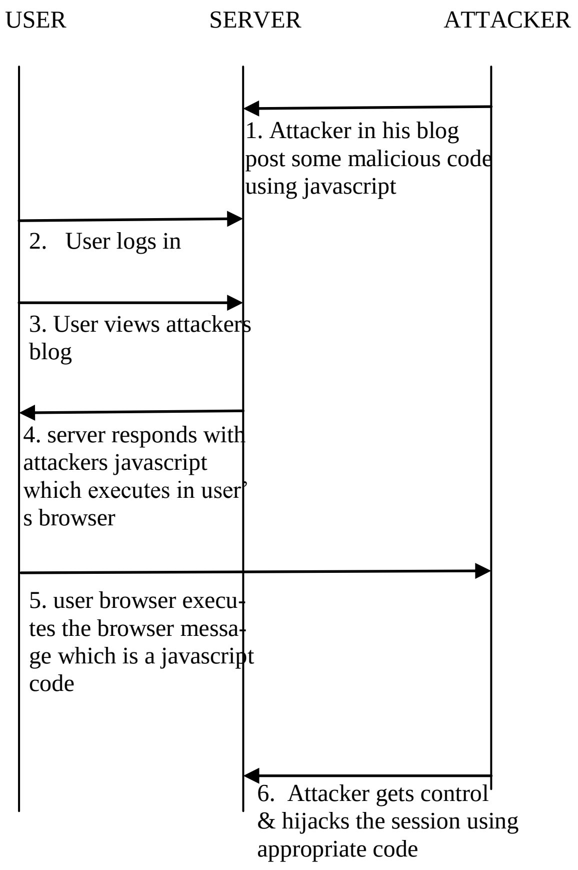
USER
SERVER
ATTACKER
1. Attacker in his blog post some malicious code using javascript
2. User logs in
3. User views attackers blog
4. server responds with attackers javascript which executes in user's browser
5. user browser executes the browser message which is a javascript code
6. Attacker gets control & hijacks the session using appropriate code

Fig 2. The steps involved in Stored XSS attack

2.3 DOM Based XSS:

In DOM based XSS [4] the hostile code written in javascript doesn't need to be posted in a blog or echoed by the website to exploit a user. The code is executed when a client side script access the URL to modify the DOM. The prerequisite is for the vulnerable site to have an HTML page that uses data from the "document.location" or "document.URL" or "document.referrer" (or any various other objects which the attacker can influence) in an insecure manner.

e.g, Consider the following code:

```
<script>var position=document.URL.indexOf("Value=")+6;
var Inputuser=document.URL.substring(position, document.
URL.length); document.write(unescape(Inputuser));</script>
```

If the page is loaded with the 'Value' parameter set to '<script>alert("xss")</script>' instead of the intended input, then the extra script will be added into the page's DOM and executed as the page is loaded and will display alert box with "XSS". Figure 3 gives the sequence of steps involved in DOM based XSS attacks.

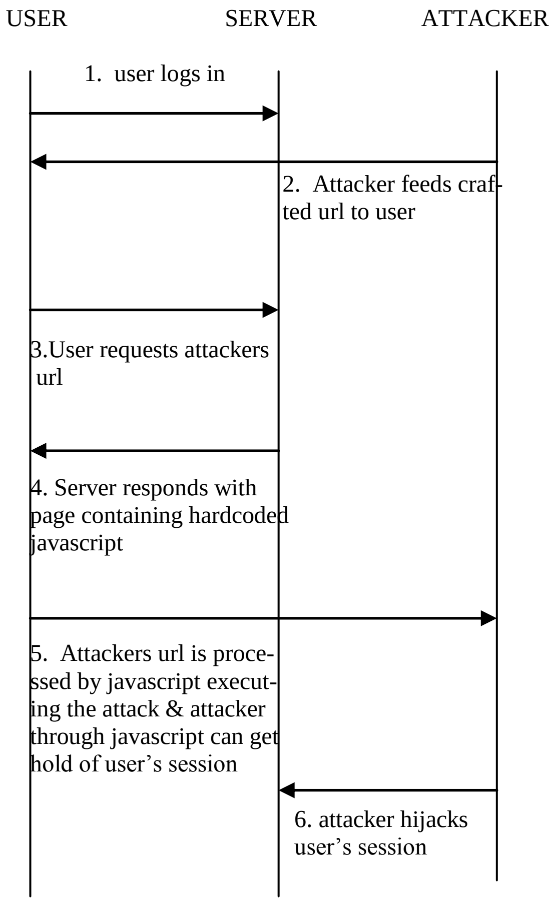
USER
SERVER
ATTACKER
1. user logs in
2. Attacker feeds crafted url to user
3.User requests attackers url
4. Server responds with page containing hardcoded javascript
5. Attackers url is processed by javascript executing the attack & attacker through javascript can get hold of user's session
6. attacker hijacks user's session

Fig 3. The steps involved in DOM based XSS attack

3. BROWSER DEFENSES AGAINST DIFFERENT EXPLOITS

A web application has been created using PHP [6] which contains all the above specified vulnerabilities. A web server "Windows Server 2008" is developed using VMWARE [7] and web site is hosted on the server. Browsers like IE8, Google Chrome, Firefox and Safari are tested for different vulnerabilities. Some of the attack vectors are taken from OWASP XSS Cheat Sheet [1] and also from earlier work of some authors & researchers [5][8] [9] [10] [11].

3.1 Basic Reflection

This is the easiest and common type of injection exploited. Atttackers unhealthy intentions are fulfilled when the web application returns maligned input without making any changes to it or properly verifying it on the server side. A javascript which is sent in the request and is returned by the web application is the most common vector.

The following can be added into the search field:

<script>alert (“XSS”);</script>

A message box will be reflected back.
In real world, the above attack is done by crafting a URL as follows:

http://192.168.1.111/search.php?search=<script>alert(“(XSS”));</script>

- Internet Explorer 8: Gives a warning message “Internet Explorer has modified this page to help prevent Cross Site Scripting”.

Output is mangled

r of script is changed to # and <script> becomes <sc#ipt>

- Firefox: Attack Succesful. An alert box with XSS appears.
- Firefox with NOSCRIPT :Gives warning Mangled output Removes < from <script>
- Google Chrome: Blocked it
- Safari: Blocked it

3.2 HTML Form Injection

A URL using FORM tag can be used by attacker and presented to the victim so that the victim submits confidential details or sensitive personal information.

http://192.168.1.111/search.php?search=<form action=http:// 192.168.1.111/check/info.txt method=”GET”>GMAIL<input type=”text” name=”email”/><br/>password<input type=”pass word” name=”password”/>Submit<input type=”Button” value=”submit”/><br/></form>

- Internet Explorer 8: Gives a warning message “Internet Explorer has modified this page to help prevent Cross Site Scripting”.

Output is distorted

r of FORM is replaced by #

and = is changed to #

- Firefox: Attack Succesful.
- Firefox with NOSCRIPT :Gives warning, Blocked the attack Remove all < tags
- Google Chrome: Attack successful
- Safari: Attack successful

3.3 IFRAME Injection

To avoid visual detection by website owners, normally invisible or hidden zero pixel IFRAMEs are used to hide the embedded malicious injected content which can easily compromise the system.

e.g <IFRAME SRC=http://192.168.1.111/check/iframe.html> </IFRAME>

- Internet Explorer 8 : R of IFRAME is changed to #
- Firefox : Attack successful
- Firefox with NOSCRIPT :Blocked it with mangled output and removes all < tags
- Google Chrome: Attack successful
- Safari: Attack successful

3.4 XSS Attack Vector Using <IMG> Tag

<IMG> tag can be used for XSS attack

<IMG SRC=javascript:alert(String fromCharCode(88,83,83)

- Internet Explorer 8 : r of script is changed to #
- Firefox: Blocked it.
- Firefox with NOSCRIPT: Blocked it.
- Google Chrome: Blocked it
- Safari: Displays?

3.5 <IMG> With Mouseover

<IMG SRC=#onmouseover=”alert;(‘XSS’)”>

- Internet Explorer 8: Block it and after SRC is omitted.
- Firefox :Block it and after SRC is omitted
- Firefox with NOSCRIPT :Blocked it with warning and removed all < tags
- Google Chrome: Blocked it
- Safari: Displays? and after SRC is omitted.

3.6 Hyper Link Insertion

This kind of injection inserts links into a particular vulnerable web page. There is high chance of click rate when a link is inserted in to the web page which is a part of trusted web site. The link could point to any website and can lead to any sensitive operation if user is already logged into that website.
e.g An attacker can send a link through an e-mail to the victim directing him to his bank login page (generated using phishing techniques) and get all sensitive information from him .

3.7 <a> With Mouseover

<a onmouseover=”alert(document.cookie)”>XSS LINK</a>

- Internet Explorer 8: Gives a warning message “Internet Explorer has modified this page to help prevent Cross Site Scripting”.
  o of onmouseover is changed to #
- Firefox: Attack successful

- Firefox with NOSCRIPT: Blocked it and removed all < tags
- Google Chrome: Blocked it
- Safari: deformed output

3.8 Stored XSS Attack

To evaluate browser defenses against persistent XSS attack a blog named blogger has been designed using Drupal-7.22(open source content management),using which it's members can post comments and messages. The attack vector <script>alert("XSS");</script> is considered and is posted as a message in one of the blogs. All the members of the blogger who visit the web site get infected and an alert box is displayed. Similar type of attacks can be used to cause more severe harm to unsuspecting user.

- Internet Explorer 8:Attack successful
- Firefox: Attack successful
- Firefox with NOSCRIPT: Blocked it
- Google Chrome: Attack successful
- Safari: Attack successful

3.9 DOM Based Attack

For DOM based XSS attack we have considered the following example :

```
<script> var  pos=document.URL.indexOf("search=")+7;
var   user=document.URL.substring(pos,document
.URL.length); document.write(unescape(user));</script>
```

- Internet Explorer 8: Doesn't execute the script & gives warning
- Firefox : Attack successful
- Firefox with NOSCRIPT: Blocked it
- Google Chrome: Blocked it
- Safari: Blocked it

4. RESULTS OF EVALUATION

Study of the behavior of different browsers against XSS attacks shows the following outcome:

Firefox does not provide any protection by itself, it uses a plug-in NOSCRIPT to provide defense against XSS attack. Firefox with NOSCRIPT plug-in installed is also considered as well. Firefox is unable to provide resistance against most of the attack vectors but with NOSCRIPT it is able to stop most of the attacks.

For most of the attack vectors Chrome filter XSS Auditor is able to block them.

The filter in IE 8 monitors all of the requests and responses made by the browser and automatically disables XSS attacks when they're detected. When an attack is blocked, users will be alerted with a modified version of the requested page.

Safari like Chrome is able to block some attacks and some attacks were successful also. Refer to Table 1 given below for results of evaluation.

Type of xss attack	Browser				
	IE 8	Google Chrome	Firefox	Firefox with NOScript	Safari
Reflected XSS	Not	Not	Yes	Not	Not
HTML Form Injection	Not	Yes	Yes	Not	Yes
Iframe Injection	Not	Yes	Yes	Not	Yes
XSS attack with <IMG>	Not	Not	Not	Not	Not
<IMG> with mouseover	Not	Not	Not	Not	Not
Hyperlink	Yes	Yes	Yes	Yes	Yes
<a> with mouseover	Not	Not	Yes	Not	Not
Stored attack	Yes	Yes	Yes	Not	Yes
DOM based attack	Not	Not	Yes	Not	Not
Yes : Attack Successful Not : Attack is Not Successful					

Table 1: Results of Evaluation

.

5. RELATED WORK

Riccardo and Sekar [11], in their paper have analyzed open-source XSS filters, XSS Auditor for Google Chrome and NoScript for Firefox for their strengths and limitations. They have presented a new browser-resident defense called XSSFilt.

Bhanu [5] has done browser analysis of IE8 and Google Chrome and concluded that IE8 was unable to provide protection against inscript injection and fragmented injection where as Google Chrome was ineffective for iframe injection, inscript injection and fragmented injection.

Adam and Collin [12] studied the chromium architecture examined recent security vulnerabilities in web browsers and evaluated whether those vulnerabilities exist in Chromium and how they can be resolved.

In their paper Marin,Jakov and Goran [13] explained Chrome architecture and Security.

6. CONCLUSION

After analyzing the different browser defenses for different attack vectors it can be concluded that the defenses employed by the browsers to counter cross site scripting attacks are not complete, even though they are stopping some categories of attacks, still there is lot to be done.

It is the responsibility of the user to be aware of different means by which an attacker can exploit the system and can gain access to sensitive information. We should keep changing our password regularly and never click on suspicious link as it may invite some problem. Less enabled features in the browser will result in a safer surfing of internet.

Here in this paper few attack vectors are considered. There are many more ways by which a wicked user may attack the system. It is essential to develop a secure web application that requires more awareness on the part of web developers.

7. FUTURE SCOPE

Here the defenses provided by different browsers against XSS attacks have been evaluated. Same can be extended to other attacks like SQL Injection attack, phishing, CSRF etc. More work has to be done to enhance the browser defenses against such attacks to have safe and secure browsing experience.

Browser defenses can be analyzed for mobile browsers also like Google Android, Microsoft IE, Skyfire , Bolt etc.

REFERENCES

[1] OWASP Foundation, OWASP Top 10 2013, Creative Commons Attribution 2.0, June 27, 2013

[2] Petko D Petkov Anton Rager Seth Fogie Jeremiah Grossman, Robert Hansen., XSS Attacks-CrossSite Scripting Exploits and Defense. Syngress, 2009.

[3] Imperva’s Web Application Attack Report Edition #1 - July 2011

[4] S.Shalini, S.Usha ,Prevention of Cross-Site Scripting Attacks (XSS) On Web Applications in The Client Side, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011

[5] Bhanu Prakash Valluri, Evaluating Browsers and The HTML5 Standard Against XSS, MTech Thesis, IIT Bombay

[6] W. Jason Gilmore**,** Beginning PHP and MySQL: From Novice to Professional, Apress

[7] Brian Ward,The Book of VMware: The Complete guide to VMware Workstation, No Starch Press 2002

[8] Paco Hope, Ben Walther, Web Security Testing Cookbook Systematic Techniques to Find Problems Fast, O'Reilly Media

[9] Daniel Bates,Adam Barth ,Collin Jackson ,Regular Expressions Considered Harmful in Client-Side XSS Filters, Carnegie Mellon university

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

[11] Riccardo Pelizzi R. Sekar**,** Protection, Usability and Improvementsin Reflected XSS Filter, ASIACCS ’12, May 2–4, 2012, Seoul, Korea.

[12] Adam Barth, Collin Jackson, The Security Architecture of the Chromium Browser

[13] Marin Sili,Jakov Krolo and Goran Dela, Security Vulnerabilities in Modern Web Browser Architecture, MIPRO 2010, May 24-28, 2010, Opatija, Croatia