International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Client Side Filter Enhancement using Web Proxy

Santosh Kumar Singh[1], Rahul Shrivastava[2]

[1]M Tech Scholar, Computer Technology (CSE) RCET, Bhilai (CG) India,

[2]Assistant Professor, CSE Department, RCET Bhilai (CG) India

Abstract: ***In early days, web pages always use a state for keeping an authentication state between browsers and web applications called cookies, these cookies are sent to the browser by the web server's after the users have been successfully authenticated. Every request that contains the valid cookies will be automatically allowed by the web sites without any further check. The cookies are used to identify and authenticate the client; therefore they are an interesting target for web attackers. Cross Site Scripting attack (XSS) is the popular attacks which is often used to steal the information from a client machines. If any cookie has been stolen by the unauthenticated users then essential and sensitive information will be disclosed. In this paper, we introduce a new technique for securing cookies from unauthorized users called "Dynamic Cookies rewriting", this technique aims to make the cookies meaningless for XSS attacks. Our technique is implemented in a web proxy where it will automatically randomize the cookie value that is sent back and forth between the users and the web applications.***

Keywords: Cookies; Cross Site Scripting; Client Site Attack; XSS Attacks;

1. Introduction

Cross-site scripting vulnerabilities during the early days of the World Wide Web are being very dangerous against sensitive information. Attackers may attack cookies and steal important information stored in the cookies. In the network application, in order to improve the user experience, there is a trend that scripting languages (mostly JavaScript) have been widely used. However, this trend also makes XSS (Cross-Site Scripting attacks) one of the most serious threats to Internet. XSS attacks is to reveal the most direct harm to the user privacy of sensitive information, and make users' personal computers infected with viruses. XSS attacks are essentially referred to the illegal scripts injected in a web page. When a user browses the page suffered XSS attacks, the scripts embedded in the web page will be triggered, resulting in some malicious attack effect.

A. Types of Attacks

There are two well known types of the XSS attacks:
1. Non-persistent
2. Persistent.

(1) Non-persistent (or reflected) XSS

Non-persistent XSS means that malicious code is not persistently stored in a vulnerable server, but it is immediately echoed by the vulnerable server back to a victim. For example, if the victim is accessing www.bank.com in order to do an online transaction, in the same time the victim may also be accessing www.attacksite.com, and be persuaded into clicking on a below link: [1]

```
<a href = "http://www.bank.com/
<SCRIPT>
document. Location ='http://www.attacksite.com/
stealcookie.php?'+document.cookie;
</SCRIPT">> Click here to collect the price.
 </a>
```

Figure 1: Non-persistent XSS attack [1]

When the unauthorized user clicks on the link of the page, the malicious script will be sent to the web server (www.bank.com) as a requested page. Once the web server cannot find the requested page, it will usually return an error page. The web server may also decide to send an identification of the requested site in the error page which is actually the malicious script. When the malicious script is executed on the user's platform, the cookies of the www.bank.com will then be sent to the www.attacksite.com. An owner of the www.attacksite.com may use those cookies to impersonate the victim with respect to the www.bank.com.

(2)Persistent (or stored) XSS

Persistent XSS means that the malicious code is persistently stored in a server's and may later be embedded in an HTML page sent to the victim. For example, a script showed in which it is posted on an online message board of the www.bank.com.

```
<SCRIPT>
document.images[0].src=http://www.attacksite.co
m/
 images.jpg?stealcookie+ document.cookie;
</SCRIPT>
```

Figure 2: Persistent XSS attack [1]

The victim who reads a message will receive the malicious script as a part of the message. The victim's browser will then execute the malicious script which will later send the cookies of the www.bank.com to the www.attacksite.com. Again the malicious script can read the cookies of the www.bank.com because it was loaded from the www.bank.com which has the same origin as the cookies.

B. Cookies

Cookies are information which are stored on a client computer. They are designed to hold a limited amount of data specific to a particular user and site, and can be accessed either by the server or the user computer. This allows the

server to send a page tailored to a particular client, or the page itself can contain script which is aware of the information in the cookie and so is able to carry data from one visit to the website (or related site) to another side. [1]
In general the cookies can be categories into two types: session cookies and persistent cookies.

- Session cookies are limited in used; they are discarded when the browser is stopped.
- Persistent cookies can be kept for long time until they expire, they are stored on a local disk and survive across a computer restarts.

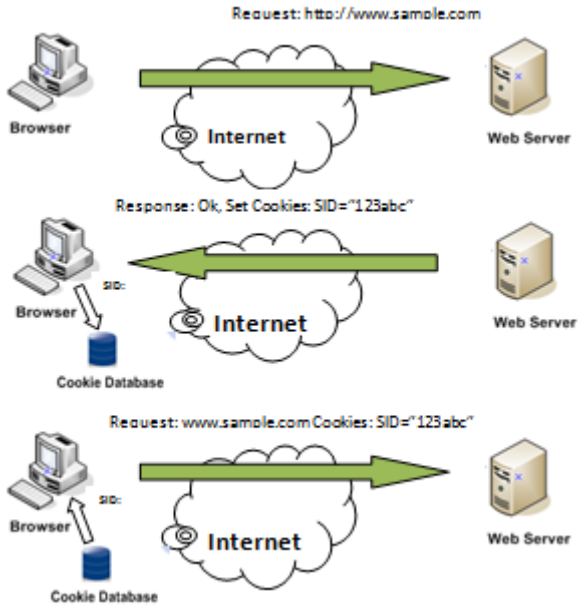

Figure 3: Web server and client exchange the cookies [1]

C. Web Proxy

In computer networks, a proxy is a server (an any comport system or an application) that acts as an intermediary for requests from browser searching resources from other sites. A client may connects to the proxy server, for requesting some services, such as various files, connection setup, web pages, or other resource available from a different sites and the proxy server evaluates the client request as a way to simplify and control its complexity. Proxies were constructed to add structure and encapsulation to distributed services today; most proxies are web proxies, providing access to content on the World Wide Web.

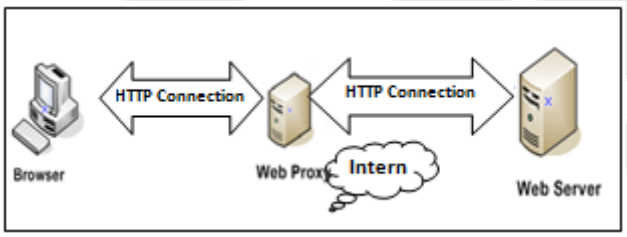

Figure 4: Web Proxy intercepts both HTTP and HTTPs [1]

D. Protection of Cookies

Besides privacy issues, cookies also have various technical lacking point. In particular, they do not always exactly identify users, they can be used for security attacks in the browsers, and they are often at odds with the Representational State Transfer (REST) software architectural style [1].

A) Inaccurate identification
If multiple browsers are used on a computer, each usually has a unique storage area for cookies. Hence cookies do not identify a client, but a combination of a user Id, a computer, and a browser. Thus, a person who uses more than one account, multiple computers, or browsers must have multiple sets of cookies.

B) Inconsistent state on client and server
The use of cookies may create an inconsistency between the state of the user and the state has kept in the cookie. If the client gets a cookie and then clicks the "Back" button of the client, the state on the client is usually not the identical as before that acquisition.

C) Inconsistent support by devices
The trouble with using mobile devices cookies is that most devices do not employ cookies; for example, Nokia only chains cookies on 60% of its mobile, while Motorola only chains cookies on 45% of its set.

2. Background and Related Work

The cookies are always used to keep the information about the session IDs or personal sensitive information in the web applications. Cookies are information which stored in small text files, in client computer.
When a web server sends a web page to a client, the connection is shut down, and the server forgets everything about the client.

Cookies are made to rectify the problem that how to keep information about client:

- When a client visits a web page, his name can be stored in a cookie.
- Again the client visits the same or different page then the cookie knows his name.

When a client requests a page from a remote server, cookies concerned to the requested page is mentioned to the client request. This way the server accepts the important information to "remember" data about client. Server sends needed data to the user's browser in the form of a cookie. The client machine retrieves the cookies from remote server, and it will be stored as a simple text record on the user's database. Now, when the browser arrives at another link on the site, the user sends same cookie to the remote server for retrieval. If cookie retrieved from client machine, remote server remembers what was stored earlier.

Cookies are a plain text data record of 5 variable-length fields:

- **Expires:** When cookie expires
- **Domain:** Site identification which indicates the name.
- **Path:** the address of the page or site's where the cookie is stored.
- **Secure:** If "secure" word is stored by this column then the session will only be accessed with a secure web server. In case of empty, no such restriction exists.
- **Name=Value:** Value of cookie. Important part of the session.

Cookies were basically designed to keep session ID's and by cookies session data will automatically transferred between client and server, so script on server may write and read cookie values that are stored on the client.

In client side script JavaScript may also manipulate cookies using the *cookie* property of the session object. JavaScript can do all the task about the cookie or cookies that apply to the current web page. Architecture of Exploiting the XSS Vulnerability

XSS attacks are those attacks against the web applications which is often used to steal the cookies from a web browser's database. The following figure1 is an architecture which shows the sequence of steps of exploiting the XSS vulnerability by a malicious attacker.

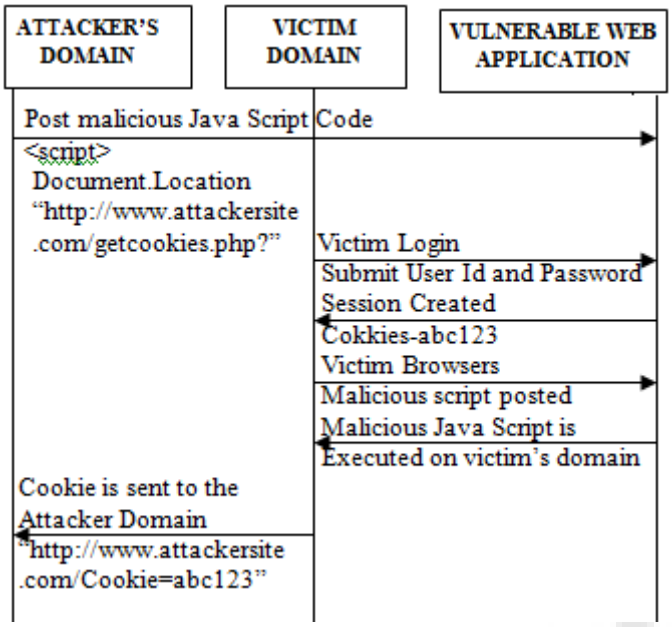

Figure 4: Architecture of Exploiting the XSS Vulnerability. [1]

The above figure shows three useful states i.e. Attacker state, Victim state and Vulnerable Web application state. By using some steps we can understand the working of figure. These steps tell us how exploiting the XSS attack:-

- In first step, the attacker has searched that the web application which is vulnerable to XSS attacks. After then, attacker posts a Java Script Code on the Vulnerable Web page whose function is to hack cookies of the victim's browser.
- In second step, the victim accesses the vulnerable web page by entering the user-id and password. In response web server of web page generates and sends the cookie of that particular session to client's web browser.
- In third step, the client browser get's the malicious Java Script Code and executed on its browser.
- In fourth step, the Java Script Interpreter of the victim browser gets involved and sends the cookies of the client browser to the attacker's domain.
- Now in the last, these cookies will be used by the attacker to access whole information from the client browser and get into the account of victim.

After all, the attacker gets successes to enter in victim's browser and exploited on the victim domain. It may be dangerous for the victim domain because all information related to side, which stored in cookie will be transferred to attacker's domain and he can steal the sensitive information.

3. Related Methodology and Limitation

Previous methodology presented a method called Dynamic Hash Generation Technique. The main objective of this method is to protect the cookies from attackers. This method is successfully implemented on the server site without major modification required on the client browser. By using this method, web server will generates the random value also called hash value to the browser's domain, therefore client browser will keep the random value of cookies in its memory instead of original value. Now every time, if the client browser rebuilt a connection i.e. active connection, the client must include the random value of cookie into its request therefore the web server must also rewrite all random values into its original format. This original format again used by the server for further work.

Cookies in which three attributes (name, domain and path) are specified for the identifying the cookies uniquely. The table contains both original and hash cookie value of victim's domain:

Table 1: Original and Hash Value of Cookie

Name	Domain	Path	Original Value	Hash Value
Cookie1	www.sample.com	/root1	789pqrs	+$@2g
Cookie2	www.exa.com	/root2	567sdfg	%$^$4

In this paper, the Dynamic Hashing Generation Technique used on the server side, which is used to generate the hash of value of name attribute in the cookie. All the other attributes (i.e. domain and path attributes as shown in the Table 1) will remain same. Following are some of the steps which are used to explain the Dynamic Hashing Generation Technique:-

- The victim from the client side submits the user-id and password to the web server of the web page.
- The web server submits the corresponding information from the victim and generates session cookies.
- Now the web server will dynamically generates the hash value of original value of the name attribute in the session cookie and stores both these values (original as well as hash value) in the form of a table on the server side.
- Subsequently, the web server will send the generated random value of the name attribute in the session cookie to the victim's web browser.
- The victim's browser will store this random value of name filed into its repository.

Since the cookies (hash version) of victim's database now are not valid for the web pages. Therefore XSS attack will not be able to use stolen cookies which are generated into its random form. Now if the victim wants to reconnect to the web server considering the part of the active connection, it has to include session cookie (hash value) with its corresponding request to the web server. The web server will use the database stored in the table to rewrite back the values

of name attribute in the session cookie (sent by the web browser) to the original value created by the web server [1].

4. Limitation

Dynamic hash generation is done in the server site, but most of vulnerability attacks found in client side so if server creates dynamic hash value to the client machine then it would be slow in process and due to remote area server have to monitor the hash value returned from browser each time. Therefore the task of the server increases each time and also travel time will be increase due to long distance between server and client.

5. Proposed Methodology

Our methodology carried out from previous technique where we also create dynamic hash value but in client side as well as we also create dynamic hash value of four fields. We create a web proxy for dynamic hash method which randomizes all four original values in to the dynamic hash value. The creation of four dynamic hash values will deviate the thief who wants to access the cookie value. He will not be able to understand the original cookie name as well as its original value. In this technique all four fields as well as their original values will be randomize before send to the client memory. Further at request time of the client the hash values will be taken from the client and proxy will convert all hash values in the original form and send to the server. If unauthorized person has accessed the fields from client memory and tries to identify the original value then he has to first identify the name of all the fields and then their respective value that would be so tedious and would be time consuming therefore the generation of hash values of four fields will be more secure as compare to single field hash generation. Here are the original values of cookie sent by server:

Table 2: Original value of Cookie

Original Field	Name	Domain	Path	Value
Original Value	Cookie1	www.google.com	./Root	Abc123

Further the web proxy will convert the original value into hash value and send these values to the client machine. As the browser stores the hash value of cookies, so even the XSS attack can steal the cookies from browser‘s database, the cookies cannot be used later to hijack or take off the user‘s session. Cookies will be secured.

Table 3: Hash Value of Cookie

Hash Field	$%^&	&^%^^	%^&&*	#$%^^
Hash Value	$%^45	#$%@#34$%	%^&12	&&^^%$

After randomizing the value of the cookie it is send to the browser’s memory. At the requesting time web proxy will access the hash value which is stored in the browser’s memory and convert it’s into original format and sends again to server.

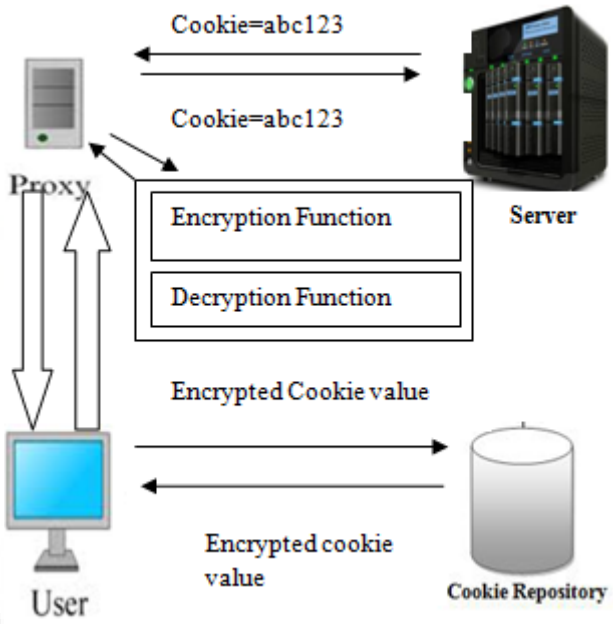

Figure 5: Web Proxy sends Hash value to Browser

6. Result and Analysis

Result of this approach will be increase the security of the cookies. It improves the speed of the transportation of the cookies. If attacker found the cookie of the browser and tries to decrypt the hash values of four fields then he has to identify the sequence of the four fields as well as all corresponding values. This process will be time consuming, till then user will be finished his work and will be off line. In the hash generation we use four fields, which are as follows:

- Server sends new session cookie to the victim and web proxy will generate its random value using random function before sending to victim’s browser.
- After generating in hash format web proxy will send all random values to the victim’s browser.
- Random value will be stored by the computer Memory of the victim.

Strengths

- The proposed method described above does not affect the performance of victim’s browser.
- This method is used by web proxy therefore it is too secure with respect to victim’s browser.
- Even if attacker will perform the XSS attack to steal the cookies from browser’s machine, the attacker will get the hash version of the cookies which are not appropriate to impersonate the users.

7. Conclusion

This paper represents the method which helps browser’s to prevent disclose information stored in the cookies. Because cookies stores random values instead of original values. Randomization of values is good idea to hide the cookie field’s values against the attackers. When an attacker attacks on the client machine and tried to find out the cookies stored in the system and finally he stolen the cookie, but during the reading the values of the cookie he will have to convert all random values in its original format that is almost too critical

and till then user may complete his work and get out from the site. The method may be implemented in various browsers and may protect the system cookies. The result of the technique is beneficial for client side

References

[1] Shashank Gupta, Lalitsen Sharma, Manu Gupta, Simi Gupta," Prevention of Cross-Site Scripting Vulnerabilities using Dynamic Hash Generation Technique on the Server Side" International Journal of Advanced Computer Research2012
[2] Yu Sun, Dake He " Model Checking for the Defense against Cross-site Scripting Attacks" 2012 IEEE
[3] Jonathan R. Mayer, Third Party Web Tracking: Policy and Technology, IEEE 2012
[4] Jyoti Snehi, Dr. Renu Dhir," Web Client and Web Server approaches to Prevent XSS Attacks" IJCT 2013
[5] Daniel Bates, Adam Barth, Collin Jackson, Regular Expressions Considered Harmful in Client-Side XSS Filters. IW3C2 2010
[6] Angelo Eduardo Nunan, Eduardo Souto, Eulanda M. dos Santos, Eduardo Feitosa, "Automatic Classification of Cross-Site Scripting in Web Pages Using Document-based and URL-based Features" IEEE 2012