Creation and Prevention of Persistent XSS Attack

Nilesh Kunhare
Research Scholar, CSE Department
MANIT
Bhopal, India
nilesh954@gmail.com

B.N.Roy
Assistant Professor, CSE Department
MANIT
Bhopal, India

Aishwary Pandey
Research Scholar, CSE Department
UIT, BU
Bhopal, India

***Abstract*— Web applications are often vulnerable to attacks. Research data shows that over 80% of the applications are vulnerable to Cross Site Scripting (XSS) attacks. It commonly targets scripts embedded in a page which are executed on client side (on the user's web browser) rather than on server side. It involves three parties- attacker, client and the website. The goal of XSS is to steal client cookies and any other sensitive information, which can identify the client with the website. There are two ways of XSS attack. Non-persistent (First Order or Reflected XSS) attack and Persistent (Second Order or Stored XSS) attack. Persistent attack is considered to be the most dangerous types of XSS attack because the attacker can directly supply the malicious input without tricking users into clicking on a URL. In this paper we will explain how persistent XSS attack takes place and understand the behavior of attackers by simply creating an environment where attacker intrudes some malicious code and the code is executed when the user visits on that infected web page and hence the attacker successfully leads to exploit the trust relationship between victim's browser and server's location. We will also discuss about the approaches that can be used for preventing XSS attack types.**

***Keywords*— First Order, Second order, Reflected, Stored, Persistent, XSS, and Cookies.**

I. INTRODUCTION

Web applications use cookies to associate a unique account with a specific user. Moreover, most retail, auction, and banking sites use cookies for authorization and authentication purposes for the accountability of many user accounts on their collective websites. A typical web application involves the exchange of two authentication tokens namely- username and password, the values stored in a cookie then used as an authentication token. It is understood that user's web session is vulnerable to hijacking if an attacker captures that user's cookies. Perhaps the cross-site scripting is considered to be the most popular scheme for stealing Internet cookies which can be used by the attacker to gain control of the user account.

II. CROSS-SITE SCRIPTING

XSS allows user to unintentionally send malicious data to him through that application. It is an attack that forces a website to echo attacker-supplied executable code, which then loads into the user's browser. Attackers perform XSS exploitation by crafting malicious URLs and tricking users into clicking on them. The links created by attackers cause client site scripting languages (such as PHP, CSS, JavaScript, VBScript, HTML etc.) to execute on the victim's browser. The attacker then able to steal the cookies and other sensitive information of user belong to that web site. XSS is a code injection attack which is performed to attack other users of the web application by crafting some malicious code, injecting them to run on legitimate user's browser. There are two ways by which users infected by XSS attack. Users either tricked into clicking on a crafted link (Non-Persistent or First order XSS attack) or, unknowingly attacked by simply visiting a web page embedded with malicious code (Persistent or Second order XSS attack).We will discuss both types of XSS attack in next sections.

A. Non-Persistent XSS Attack

In non-persistent XSS attack, the attacker provided script is embedded in the web page is eventually executed by the server as an immediate response of request. This is the reason why it is also known as Reflected attack because the malicious code is directly reflected back to the user by use of third party mechanism. It is combined with other techniques such as social

engineering and phishing [6] in order to steal cookies, and other sensitive information of user (like credit card number) These attacks are most commonly found in search engines like AOL, Ebay, Google, Amazon etc.

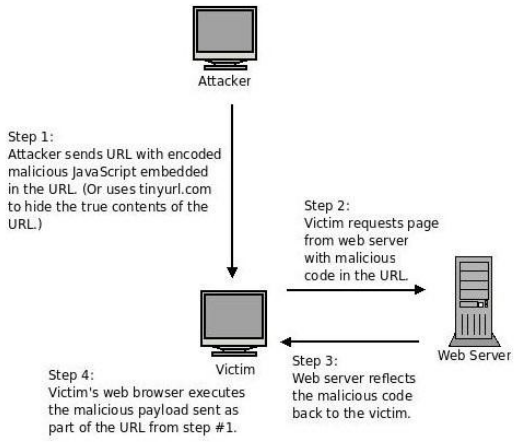

Figure 1 the block diagram of how non-persistent XSS attack takes place.

Let us call the site under attack- http://myserver.com/attack.jsp. Suppose that the script is named attack.jsp and its parameter is "name". It can be operated in this way.

<%out.println("Welcome " + request.getParameter("name")); %>

Then the response would be:

Welcome Nilesh

The attacker then writes some malicious code such as

http://myserver.com/attack.jsp?name=<script>alert("Hacked")
</script>

The victim's browser would interpret this response as an HTML page containing a piece of JavaScript code. This code, when executed is allowed to access all cookies belonging to that site. At this point the hacker will continue to modify this URL to include more sophisticated XSS attack to exploit users. It can steal credential, deface a website, and create a fake page or spam mail and many more.

B. Persistent XSS Attack

Persistent XSS attack do not require specially crafted link for execution. A hacker submits XSS exploit code to an area of a website that is likely to be visited by other users. This area could be message board posts, chat rooms, user reviews, comments, HTML email and numerous other locations. Whenever the user visits the infected web page, execution is automatic. This is the reason why persistent attack is considered to be more dangerous than non persistent attack because the user has no mean of defending himself from that infected page.

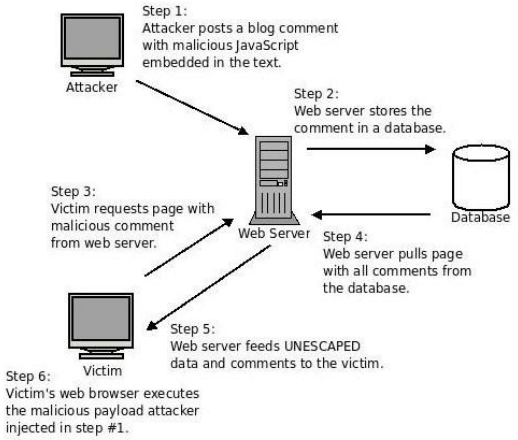

Figure 2 the block diagram of how Persistent XSS attack takes place.

Let us consider a scenario where a user visits a vulnerable website named

http://mywebpage/login.jsp

When authorized user accesses the database of a company by entering username and password of that site, it displays the related web page of that site. But whenever that authenticated user want to update some records he have to visit on next page by clicking on some command button, as the user clicks on that button, the user visited to the infected page. This is because the attacker crafted some malicious script like-

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

This malicious code is executed whenever user visits that particular infected web page. This is shown in the figure 3 that how persistent XSS attack takes place as user visits the web page. The attacker's target is to steal the cookies and other sensitive information of the legitimate users of web page. The fig 2 shows the block diagram of persistent XSS attack and describes how it executes the stored malicious code.

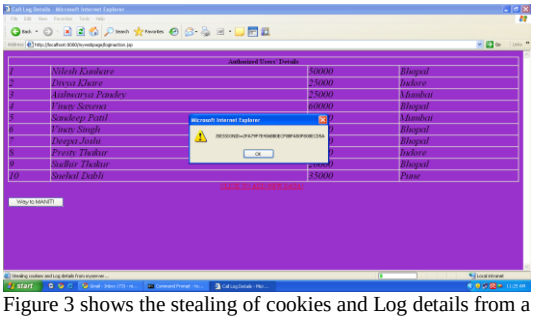

Figure 3 shows the stealing of cookies and Log details from a web-page.

Persistent attack occurs mostly at community driven web applications which involves the visiting of web pages. It neither requires attacker to publish any crafted URLs nor requires social engineering. Such attack can happen only when the user visits that infected page, as we experienced in our example. This is the reason that persistent attack can easily be hidden from the victims. Hence they are more vicious and lethal than non-persistent attack. SAMY worm [5] is one of the famous persistent XSS attacks that happened on MySpace.com, spread by exploiting a persistent XSS vulnerability on personal profile web page. Samy, the author update his profile by exploiting JavaScript code. He was successful in uploading his code by using some filter-bypassing technique. Whenever an authenticated MySpace user visits Samy's profile, the worm payload using XHR, forced the user's Web browser to add Samy as friend and alter the user's profile with a copy of malicious code. Another form of persistent XSS attack recently found in Facebook [7] where the attackers locate the vulnerability in application publishing form. Researchers found that attacker permanently inject the malicious code into scamp Facebook application pages. The attackers able to hijack user's cookies and performs unauthorized actions because malicious code was executed by the web browser in the context of the domain.

III. PREVENTION TECHNIQUES

We discussed about two types of XSS attack. Now we will discuss about the solutions which can be used for preventing it. To prevent against non-persistent XSS attack, the application user's need to check anchors before clicking on them, and application need to modify or reject input values that contain malicious code. If the input characters are sanitized then it can restricts the attacker from using some special characters which are not needed by authorized users. There are various approaches [9 and 11] used for preventing against the types of cross-site scripting attacks, some of them are discussed below.

1). Filtering

One of the common methods to prevent XSS attack is filtering of JavaScript. Any JavaScript code in the input must be transformed in way it is not execute by a web browser if sent to it.

Input filtering: The script code must be transformed into a text string that is displayed by the web browser instead of being execute, rather deleting all JavaScript code during input filtering. It can be achieved by HTML encoding like

<script> → <script>

Output filtering: It is a process when an application itself is using JavaScript in HTML documents to remove any script code in the final HTML document before it is sent to the user's browser. The following table shows HTML tags which is considered to be dangerous when executed.

HTML-Tag	Use	Risk
<APPLET>	Embedding of Java-Applets	Execution of malicious code
<XML>	Embedding of Meta statements for HTML	Execution of malicious code
<SCRIPT>	Embedding of executable script code (JavaScript, Jscript, VBScript)	Execution of malicious code
<OBJECT>	Embedding of external objects. ActiveX-Controls, Applets, Plug-Ins	Execution of malicious code
<STYLE>	Embedding of formatting Instructions(Style sheets) for HTML	May contain JavaScript type=text/javascript""(JavaScript Stylesheets)
<EMBED>	Embedding of external objects. Plug-Ins, executable code.	Execution of malicious code

2). Cookie Security

We know that XSS attack targets to steal cookie value of web applications because they are used for track activities, store and retrieve information etc. To mitigate this particular threat, we can tie session cookies to the IP address of the user who originally logged in, and only permit that particular IP to use that cookie.

3). Using BBCode

Most of the web-applications provide the facility to the users to enhance and format the text. However, these formatted texts are vulnerable to XSS attack. The web application can be used with BBCode [8] to provide user with enhanced

and formatted text and hence can protect against such attack. Consider the example of BBCode.

[url =http://mywebserver.com] Persistent XSS attack [/url]
to
< a href =http://mywebserver.com”>Persistent XSS attack.
The above illustrate a normal anchor tag BBCode. However, if we place a JavaScript into the BBCode then it can be still a BBCode, as mentioned below.
[url=javascript: alert(‘Nilesh’)] Persistent XSS attack [/url]
To
<a
href=javascript:alert(‘Nilesh’)”>http://mywebserver.com”>Persistent XSS attack.
As illustrated above, the BBCode is used with placing JavaScript to prevent against such attacks.

4). Disabled Cookie
Another option to prevent against XSS attack is disabling cookie of web browser. Cookies are accountable for authorization and authentication of legitimate users. But an alternative must be developed which provide the functionalities that required for users association with web application than depending on it.

5). Disabled JavaScript
Cross-site scripting uses the JavaScript code to harm users. We can prevent XSS attack by disabling JavaScript in web applications. This approach act as a white list approach where only trusted website has JavaScript enabled to prevent any damage from occurring. Disabling JavaScript means that newer site which only deploys JavaScript may not be accessible.

IV. CONCLUSION

We discussed in this paper about the XSS attack and its types. We reached on a conclusion that Cross-site scripting (XSS) attacks are likely to originate on popular websites with community-driven features such as blogs, Web mail, chat-rooms, message boards, social networking sites, Wikis and user reviews. We observed that the XSS attack will be challenge to spot because the network behavior of infected browsers remains relatively unchanged and the JavaScript exploit code is hard to distinguish from normal Web page. We observed the behaviors of types of XSS attack. We found that Persistent XSS attack is more destructive than Non-persistent XSS attack as the application need to reject or sanitize input values that may contain script code. We also discussed about the techniques used for preventing the types of cross-site scripting attacks like Filtering, Disabling Cookie and many more which can be helpful to prevent against the XSS attacks.

REFERNCES

[1] The Evaluation of Cross-site Scripting Attacks by David Endler, 2002
[2] Simple Cross-site Scripting Attack Prevention
Florian Kerschbaum SAP Research Karlsruhe, Germany
[3] Cross-site scripting Explained
Amit Klein, Sanctum Security Group
[4] Automatic Creation of SQL Injection and Cross-Site Scripting Attacks
Adam Kie˙zun (MIT), Michael D. Ernst (University of Washington)
Philip J. Guo and Karthick Jayaraman (Stanford University)
[5] MySpace Worm Explanation,
[6] Jagatic, T., Johnson, N., Jakobsson, M., and Menczer, F. Social Phishing. To appear in Communications of the ACM
[7] Facebook Spam Worm Propagates via Persistent XSS Vulnerability (http://news.softpedia.com/news/Facebook-Spam-Worm-Propagates-via-Persistent-XSS-Vulnerability-188934.shtml)
[8] BBCode Wikipedia
[9] Filtering JavaScript to prevent to cross site scripting.
[10] Web Security Detection Tools by Abhishek Agashe (San Jose State University)
[11] Solution to Cross-site scripting (http://hungred.com/web-development/solutions-crosssite-scripting-xss-attack/)