www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 3 March 2014 Page No. 4066-4068

Cross Site Scripting (XSS): The dark side of HTML

Junaid Latief Shah[1], *Asif Iqbal Khan*[2]

[1]Dept of Computer Science, University of Kashmir,
Srinagar, India
junaidlatiefshah@gmail.com

[2] Dept of Computer Science, University of Kashmir,
Srinagar, India
khanasifiqbal7@gmail.com

Abstract: ***In recent times, web remains the preferred platform for users to carry out their business activities. The migration of applications to web has been rapid ranging from applications like E-commerce, Public forum, E-governance, E-banking, Shopping Portals or any other applications running on the web. Web Applications have increased its usage because of easy accessibility to different users around the world. But as the usage of the web has increased, it has also given an undesirable or dark side to the usage of html. Cross-site scripting (XSS) attacks continue to remain the topmost threat to web apps, databases and websites around the world for a considerable amount of time now. A survey of about 15 million cyber attacks in the third quarter of 2012 has revealed that most of these attacks are XSS based. Although attacks like SQL Injection, CSRF and Phishing are also common, XSS still remains the preferred technique for hackers to carry out malicious activities on web. This paper discusses about XSS attacks, their operation and different categories of XSS attacks. The paper also highlights the mitigation scenario and techniques possible for prevention.***

Keywords: Cross Site Scripting (XSS), SQL Injection, CSRF, Phishing, Cyber Attacks.

1. Introduction

In the past, enterprise software would be located in trusted areas of a company's network. A company's Application would remain on single system in the office or on all the machines with their own software copy with some communication between these computers or no communication at all. As a result these applications were very less vulnerable to different attacks and hackers. But the world is changing. Web based applications are gaining more and more popularity and usage day by day. Today Web Applications have become most important communication channel between the service provider and the users. Today, Web Applications are gaining more and more popularity as we can build very beautiful and user interactive pages by the extensive use of some client site scripting languages e:g JavaScript. And this growing use of JavaScript is increasing serious security vulnerabilities in web application like SQL injection and Cross Site Scripting or XSS, later being the topmost threat.

SQL Injection is an attack where the victim is the database on the server in which an attacker can inject some code into the database as a part of query and can sneak into the database to find the data like passwords, usernames etc**.** According to the WASC (Web Application Security Consortium) reports, about 9% of the totals hacking incidents reported until 27th July 2006 were due to SQL Injection [1]. Recent research from Acunetix also shows that 50% of the web applications scanned every year are susceptible to SQL Injections. SQL injection also have a variant [3][4] known as Blind SQL Injection that aims to ask database true or false questions and determines the answer based on what application responses. This technique is a hit and trial method which a hacker uses to get access to victims database. Cross Site Scripting (XSS) is different from SQL injection in the way that XSS targets the client's browser i:e the victim here is not the database server but the client browser.XSS is an attack in which an attacker injects some scripting code into the output of a web application which is then sent to a victim's web browser where the scripting code gets executed.

2. Cross Site Scripting (XSS)

Cross Site Scripting (XSS) vulnerabilities have been the nightmare of Web applications for considerable amount of time now. The research carried out at WASC [6] shows that about 100,059 XSS vulnerabilities have been checked by analyzing 31,373 Web sites. Cross Site Scripting (XSS) vulnerabilities attack web applications by inserting client side code or script into web pages that are accessed by users. A number of popular websites including Face book, Twitter, McAfee, MySpace, eBay and Google have been the prime targets of XSS exploits. The attack exploits improper coding of your web applications allowing a hacker to inject malicious script into a web form to allow them to gain access or tamper your application.ie improper sanitization or filtering of user input. The executable code of XSS is normally written in popular scripting and programming languages like JavaScript, vbscript, php etc. The pseudo code and the figure 1 below show little demonstration of an XSS attack

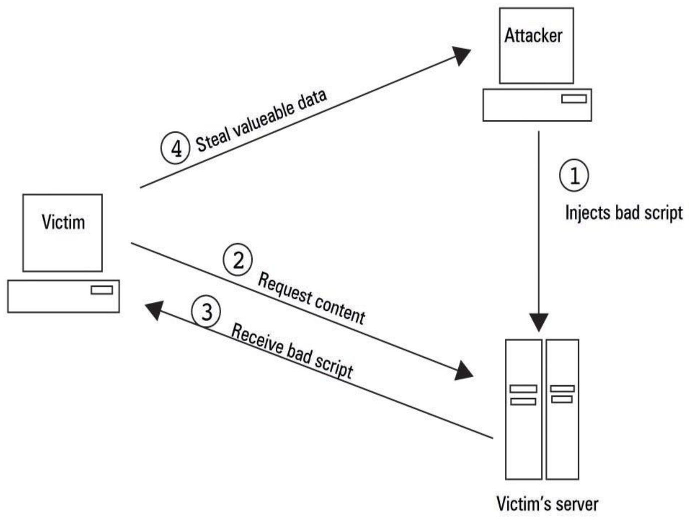

Figure 1 Operation of XSS

Suppose there's a public forum or blog or discussion website where people can ask Questions regarding some subject. Each question asked by users is stored in a database and rendered as a list, if someone requests the relevant section of the page. Such a list might look like this (no XSS code embedded here):

```
<html>
<head>
<title> Discussion Forum – Technology Section</title>
</head>
<body>
 List of questions:
<p> Question: "Which is the advantage of <i> cloud computing</i> in current times?"
</p>
<p> Question: "What are the attributes of distributed operating systems?"</p>
</body>
</html>
```

When a malicious user visits this page he will immediately notice that the text cloud computing is rendered italic in his browser and concludes that the user that posted the question added the corresponding tags himself. Now the malicious user might post a question like this:

Code with embedded XSS attack:

```
<html>
<head>
<title> Discussion Forum – Technology Section</title>
</head>
<body>
List of questions:
<p> Question: "Which is the advantage of <i> cloud computing</i> in current times?"
</p>
<p> Question: "What are the attributes of distributed operating systems?"</p>
<p> Question: "<Script>alert ('Welcome to technology…') ;< /script>"</p>
</body>
</html>
```

Now, every time a user visits this page, a pop-up will be generated every time and appear in that user's browser that displays the words "Welcome to technology…" While only some technology literate users will actually consider this as attack, other naive users will surely not pay any heed and consider it as a normal pop up. By this way of injecting malicious code into web pages, an attacker can gain high access-privileges to sensitive page content, website cookies and numerous other information stored by the browser on behalf for user, making Cross-site scripting attacks, therefore a unique case of malicious code injection [6].

3. Types of XSS

There is basically no standard classification of Cross site scripting but usually experts divide these attacks in two main types Persistent and non Persistent. Some experts classify XSS threats as Type 0 (DOM Based), Type 1 (Reflected) and Type 3 (Persistent).

3.1 Type 0 or DOM based Attack

This form of XSS vulnerability is also referred as **DOM-based** (Document object Model) attack. In DOM based cross-site scripting attack, the vulnerability is present within a page's client-side script itself. For example, if a piece of JavaScript accesses any object like ***document.url*** and uses this data to write some HTML code to its own page, and if this data is not properly encoded using HTML tags, an XSS loop hole will likely be existing, since this written data will be re-interpreted by browsers as HTML which could include additional client-side script

3.2 Type 1 or Non Persistent Attack

Non persistent or reflected threat occurs when the input data provided by user via form submission or query string is immediately used by the server to display the result back to the user or we can say the input is written back unaltered. If the user supplied data is not filtered or validated, some of the results could include a client-side script that is executed in the browser instead of HTML.

3.3 Type 2 or Persistent Attack

Persistent or stored XSS attack is called persistent because it gets stored somewhere on the server and the effect of the attack is not immediate. In this type of attack the data provided by the attacker gets stored on the server usually in the database and then displayed as normal to all other users. An example of this type of attack is when someone writes a HTML formatted review or comments on a review for other users to read. When some user reads the review the code gets executed on the user's browser and does some unwanted stuff like stealing cookies, redirect to some other page etc.The operation is shown in the figure 2 below.

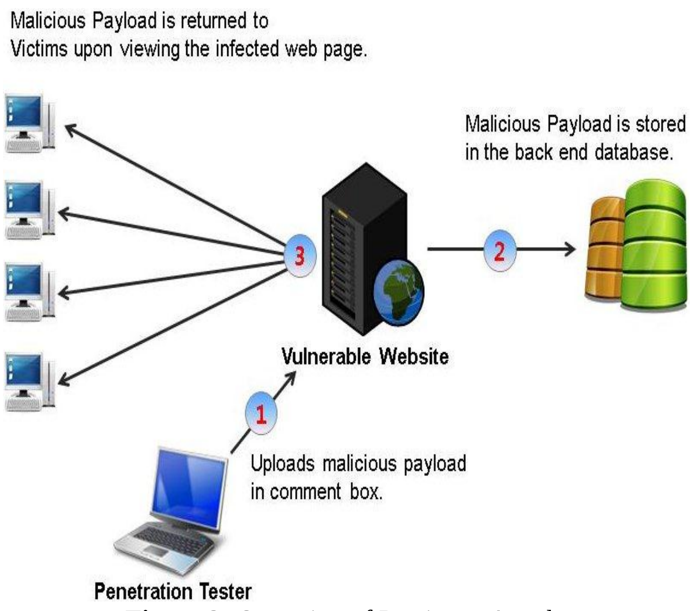
Malicious Payload is returned to Victims upon viewing the infected web page.
Malicious Payload is stored in the back end database.
3
2
1
Vulnerable Website
Uploads malicious payload in comment box.
Penetration Tester

Figure 2: Operation of Persistent Attack

For example the code in the comment or review can be like this
<b>Thank for your review <script>
window.location.href="http://abc.com"</script>
The above message will be stored in the database as it is and when some future user visits the page, the comment will be displayed but immediately the code in the script tag will be executed and the victim will be redirected to "*abc.com*".

4. Mitigation Techniques

A study of majority of web attacks reveals that they are caused due to improper coding of web applications and inability to filter or sanitize input coming into web. The attacks such as XSS and injection attacks occur due to non sanitization of user input. The majority of these web application attacks are mitigated either on the client side or server side. The client side mitigation normally involves input validation techniques. These techniques normally restrict a user from supplying malicious data to the web page. On the other hand, server side mitigation involves filtering the user input or output sanitation. One solution is also to block all JavaScript in your browser but that will restrict the user from developing or viewing interactive web applications .Some researchers have also suggested using browser plug-in that will incorporate some kind of artificial intelligence to restrict or filter user input thereby adding an intelligence factor to browser. Analyzing xss attacks, there are number of client side solutions implemented by the developers all over the world but still to completely secure a web application is still its infancy.

5. Conclusion

This paper carried an extensive survey about cross site scripting attack and discussed about different types of XSS attacks. The information contained in this paper could be very useful for new application/web developers for developing smarter and secure applications running over the web. The paper also lists some of the mitigation scenarios. Although a complete secure application is not guaranteed in the modern world, but still a considerable amount of work and research has been done in this area. Completely securing a web application seems to be a daunting task for developers today.

References

[1]http://www.acunetix.com/websitesecurity/sql-injection/
[2]http://www.computerweekly.com/news/2240168930/XSS-attacks-remain-top-threat-to-web- applications
[3]https://www.owasp.org/index.php
[4]http:// blindsqlinjection.com/
[5]web Application Security Statistics,06(WASC) http://www.webappsec.org/projects/statistics/
[6]Y. Xie and A. Aiken, "Static Detection of Security Vulner-abilities in Scripting Languages," Proc. 15th Use nix Security Symp. (Use nix-SS 06), vol. 15, Use nix, 2006, pp.179-192.
[7]Rohit Dhamankar,MikeDausin,Marc Eisenbarth, and James King. The top cyber security risks http://www.sans.org/top-cyber-security-risks/,2009
[8]http://cwe.mitre.org/top25/(2010).
[9]Rao,T."DEFENDING AGAINSTWEBVULNERABILITIES AND CROSS-SITE SCRIPTING." *Journal of Global Research in Computer Science* 3.5 (2012): 61-64.
[10]O. Hallaraker and G. Vigna. " Detecting Malicious JavaScript Code in Mozilla", In proceedings of the IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), 2005