© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

PREVENTION AGAINST CROSS-SITE SCRIPTING ON SERVER SIDE IN WEB APPLICATIONS

Harshit Punn, Himanshu Sethi,
Department of Information Technology,
Dronacharya college of Engineering, Gurgaon, Haryana

***Abstract*— Cross Site Scripting attacks are a very common vulnerability and are the most popular security problems in modern web applications. Twitter, Facebook, MySpace and many other social networking sites are used by millions of users around the world. Cross site scripting is one of the top most vulnerability exploited by Hackers in modern web applications. Cross Site Scripting permits an attacker to embed mischievous HTML, JavaScript, VBScript, or Flash into an unsafe dynamic web page to fool the user by executing the script on his machine for gathering data. According to OSWAP, these attacks on web applications have experienced an important rise in recent year. This Research paper provides solutions to alleviate cross-site scripting attacks. Current solutions degrade the performance of user's web browsing experience. In this research paper we present a step by step approach to protect cross site scripting on the server side, without degrading user's web browsing experience.**

***Index Terms*—Cross Site Scripting, PHP, Programming approach, Preventing Cross Site Scripting attack**

I. INTRODUCTION

Cross-Site Scripting, also known as XSS. It is a form of attack found in web applications. These attacks occur when an attacker or hacker uses a web application to send harmful code, generally in the form of a browser side script, to a different end user. The script code is downloaded and executed by the web browser, when a user opens a web page. The pernicious script inherits the users' rights, authentication and other data.

XSS can also escape traditional tools such as firewalls and Intrusion Detection Systems (IDS) because they perform through ports used for regular web traffic which usually are open in firewalls. Developers of web-based applications have little or no security background. It is one of the major reasons for the popularity of cross-site scripting. This result in poorly developed code, consisting of security flaws, is deployed and made accessible to the whole Internet. Any web page which passes parameters to a database can be vulnerable to this hacking technique. XSS is usually present in login forms, forgot password forms etc.

XSS flaw can be difficult to identify and remove in web applications. For finding flaws the best way is to perform a security review of the code and search for all places where input from an HTTP request could possibly make its way into the HTML output.

Further in this paper section 2 presents some background and discusses about XSS attack. Section 3 presents ways to prevent Cross site scripting. Section 4 presents the proposed programming approach to deal with cross site scripting. Finally Section 5 gives the conclusion of the paper.

II. XSS Attack

Internet applications today are not static HTML pages. They are filled with ever changing and dynamic content. It is one of the most popular web application-layer attacks. XSS carried out on many websites accounted for roughly 84% of all security vulnerabilities documented by Symantec as of 2007.

Cross Site Scripting frequently targets scripts embedded in a web page which are executed on the client-side (in the user's web browser) rather than on the server-side.

2.1 Types of Cross Site Scripting

In general there are currently three major categories of cross-site scripting. Others may be discovered in the future.

- Reflected Cross-Site Scripting

- Stored Cross-Site Scripting

- DOM based Cross-Site Scripting

Reflected XSS

Reflected Cross-site Scripting (XSS) occur when an attacker or hacker injects browser runnable code within a single HTTP response. This XSS is one of the most common types of cross-site scripting exploit.

Stored XSS

Stored XSS is also known as HTML injection attack. In this XSS attack the injected script is stored permanently on the target server, like in a database, message forum etc. The example of this kind of vulnerability is content management software such as forums and bulletin boards where users are allowed to use raw HTML and XHTML to format their posts.

DOM based XSS

DOM based XSS appear in the Document Object Model. It is also used by the browser for security. This vulnerability appears in the Document Object Model instead of part of the HTML. The harmful results of this XSS flaw include cookie retrieval, further harmful script injection etc.

III. PREVENTING CROSS SITE SCRIPTING ATTACK

Previous section tells us about the cross site scripting and its types. Cross site scripting is very similar to SQL injection. To prevent XSS we need to implement multiple filters along the communication pathway. This section presents the ways to prevent XSS.

3.1 Input Sanitization

User input can come from a variety of sources. A malicious user is not going to announce that he/she is here to attack your website. Because of this reason all input should be checked and validated. Input validation is done to prevent the entry of unsafe data into the web application. We cannot trust the data coming from the user or any third party sources. It focuses on manipulating the data to make sure it is safe by removing unwanted or unsafe bits from the data. If you are expecting a plain text from user input, you may want to remove any HTML Markup from it.

3.2 Input Validation

It is the process of ensuring that your application is running with correct data. Every piece of user data must be validated when it is received to ensure it is of the corrected type, and discarded if it doesn't pass the validation process. For example, if you wanted to validate a phone number, then you would discard any string containing letters as phone numbers contain letters only. Input validation and input sanitization move hand in hand.

IV. PROPOSED PROGRAMMING APPROACH TO DEAL WITH CROSS SITE SCRIPTING ATTACK

This method is explained in PHP programming language and the logic can be implemented in other programming languages too. This method can't be implemented in search engine. This method verifies that the parameters coming through GET and POST method are present in the database or not. In this the parameters are not checked directly but the corresponding values from the database are fetched and the parameter is checked from those values fetched from the database. If the parameter matches or if it is present then the desired SQL query executes and if it is not present then we redirect the user to our desired webpage.

```
If (/* Checking the variables are set or not */)

{

$Variable= /*Storing the parameter coming from GET or POST method */

$Status    = 0 /* this value */

$SQL query          = /* query to fetch the corresponding values */

While (/* fetch corresponding value */)

{

If ($Variable === $fetched value)

{

$Status    =          1 /*if the parameter is equal to corresponding value then Status is equal to 1*/

}

}

If ($Status)

{

/* Run the desired query with fetching the desired results */

}

Else /* if Status is not equal to 1 */

{

/*Redirect it to your desired web page */

}

}
```

V. CONCLUSION

Cross site scripting is one of the most dangerous attacks. Hackers use this attack to sneak into web applications today. Web applications should be secure and stable. It is an attack on the privacy of the clients of a particular website. This paper outlines the methods to securely code your web application. Filters can also be used when we use variables in SQL query.

REFERENCES

1. OWASP https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
2. Wikipedia http://en.wikipedia.org/wiki/Cross-site_scripting
3. Acunetix http://www.acunetix.com/websitesecurity/cross-site-scripting/
4. eSecurity Planet http://www.esecurityplanet.com/browser-security/how-to-prevent-cross-site-scripting-xss-attacks.html