Advances in Computational Research
ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 7, Issue 1, 2015, pp.-233-235.
Available online at http://www.bioinfopublication.org/jouarchive.php?opt=&jouid=BPJ0000187

CROSS-SITE-SCRIPTING - ATTACK AND PROTECTION MECHANISMS

GHODKE A.S.* AND SEN L.

Sinhgad Institute of Business Administration & Computer Application, Lonavala - 410 401, MS, India
*Corresponding Author: Email- ghodke.a@gmail.com

Received: December 18, 2014; Revised: January 05, 2015; Accepted: January 15, 2015

Abstract- In today's world more than 80% of the web applications are vulnerable to XSS threats. User friendly web applications are developed to increase the customer base and hackers utilize the features provided by the web applications. This paper surveys such vulnerabilities with the current solutions. The strengths and weaknesses of all approaches are discussed. Cross Site Scripting (XSS) is a type of computer security exploit where information from one context, where it is not trusted, can be inserted into another context, where it is. The trusted website is used to store, transport, or deliver malicious content to the victim. The target is to trick the client browser to execute malicious scripting commands.
Keywords- Application-level web Security, Cross-site scripting, Computer Security, Security vulnerabilities

Citation: Ghodke A.S. and Sen L. (2015) Cross-Site-Scripting - Attack and Protection Mechanisms. Advances in Computational Research, ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 7, Issue 1, pp.-233-235.

Introduction

Web applications do raise a number of security concerns stemming from improper coding. Serious weaknesses or vulnerabilities, allow hackers to gain direct and public access to databases in order to churn sensitive data. The focus of this research is on the above issue which is the top web application security vulnerability called Cross Site Scripting (CSS or XSS). The Cross Site Scripting is one of the most common application level attacks that hackers use to sneak into web applications. The web site is the target of attack and the user is both the victim and the innocent accomplice. XSS is "Cross-site scripting (XSS)" is a type of computer insecurity vulnerability typically found in Web applications (such as web browsers through breaches of browser security) that enables attackers to inject client-side script into Web pages viewed by other users. the reason of that is the developer trusts user inputs, or miss filtering issues , then send back user input data to the client browser so the malicious code will execute.

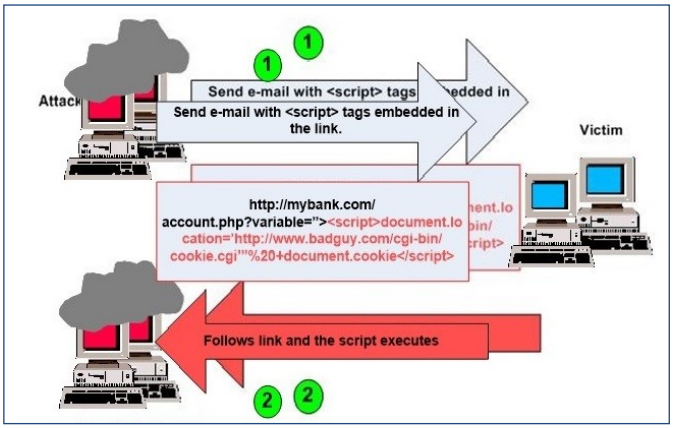
Types of XSS

Three known types

- Reflected (Non-Persistent) : Link in other website or email
- Stored (Persistent) : Forum, bulletin board, feedback form
- DOM based XSS : PDF Adobe Reader , FLASH player

Reflected (Non-Persistent)

The non-persistent (or reflected) cross-site scripting vulnerability is by far the most common type. These holes show up when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to generate a page of results for that user, without properly sanitizing the request.



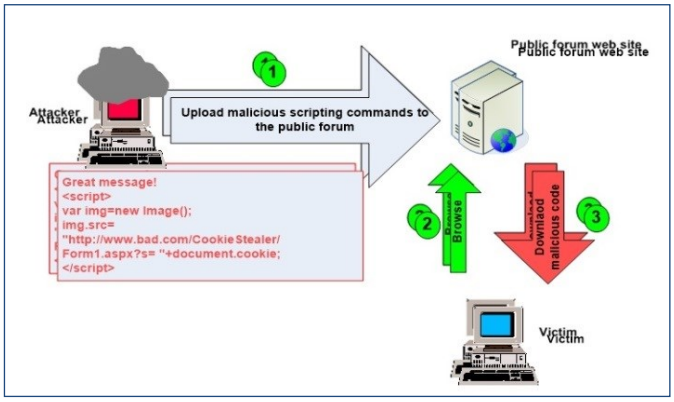
Stored (Persistent)

The persistent (or stored) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. A classic example of this is with online message boards where users are allowed to post HTML formatted messages for other users to read. Simply Persistent XSS is occurs when the developer stores the user input data into database server or simply writing it in a file without a proper filtration , then sending them again to the client browser.

Here is a PHP code that suffers from Persistent XSS:-

```
<? Php
    If (isset ($_POST ['btnSign']))
        {
        $message=trim ($_POST ['mtxMessage']);
        $name=trim ($_POST ['txtName'])
// sanitize message input
        $message = stripslashes ($message);
        $message = mysql_real_escape_string($message);
// sanitize name input
        $name = mysql_real_escape_string ($name);
        $query = "INSERT INTO guestbook (comment, name) VALUES ('$message','$name');";
        $result=mysql_query ($query) or die ('<pre>'.mysql_error ().'</pre>');
        }
?>
```

The two parameters in that code “message” and “name” are not sanitized properly ,the ,we store these parameters into the guestbook table, so when we displaying these parameters back the client browser, It will execute the malicious JavaScript code.



DOM based XSS

DOM-based vulnerabilities occur in the content processing stages performed by the client, typically in client-side JavaScript. The name refers to the standard model for representing HTML or XML contents which is called the Document Object Model (DOM) JavaScript programs manipulate the state of a web page and populate it with dynamically-computed data primarily by acting upon the DOM. simply that type occurs on the JavaScript code itself that the developer use in client side for example "A typical example is a piece of JavaScript accessing and extracting data from the URL via the location.

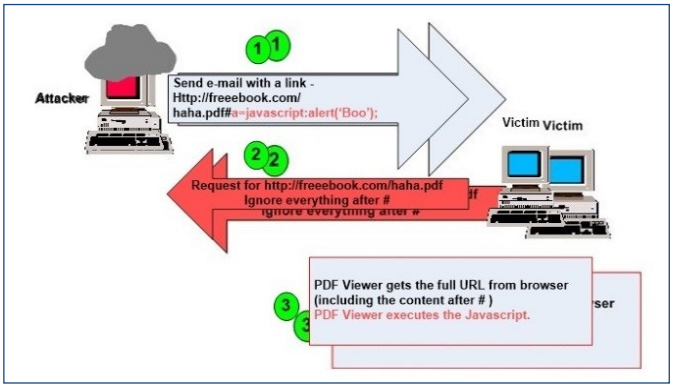
Advanced Techniques

There are some avoidance Techniques can be taken to protect a against XSS exploits but they are not

Implementing well for example

Tons of sites may seem vulnerable but not executing the code that occurs because some kind of

Filtration methods and those may can be bypassed ,we will demonstrate most of them.



Method 1: Replace <script> with null string ""

Here is the vulnerable code that suffers from reflected xss , that has a filtration :

```
<? Php
        If (! array_key_exists ("name", $_GET) || $_GET ['name'] == NULL || $_GET['name'] == '')
        {
        $is empty = true;
        }
        else
        {
        echo '<pre>';
        echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
        echo '</pre>';
        }
?>
```

As you can see, in the previous code, the developer replace the string that called "<script>" with a Null string "" .

Method 2: Magic Quotes Filtration

In this Technique, the developer uses technique that called magic quotes filtration, by using a PHP function called "add slashes()" that add slash before any special chars. So our traditional JavaScript code doesn't work.

Conclusion

- Cross-Site Scripting is extremely dangerous
- Identity theft, Impersonation
- Cause: Missing or in-sufficient input validation
- XSS-Prevention Best Practices
- Implement XSS-Prevention in application
- Do not assume input values are benign
- Do not trust client side validation
- Check and validate all input before processing
- Do not echo any input value without validation
- Use one conceptual solution in all applications

Conflicts of Interest: None declared.

References

[1] Kirda E., Jovanovic N., Kruegel C. & Vigna G. (2009) *Comput-*

ers & Security, 28(7), 592-604.

[2] Bicho D. (2004) *PHP-nuke reviews module cross-site scripting vulnerability*, 10493.

[3] Jovanovic N., Kruegel C. & Kirda E. (2006) *Proceedings of the 2006 ACM workshop on Programming languages and analysis for security*, 27-36.

[4] Kirda E., Kruegel C., Vigna G. & Jovanovic N. (2006) *Proceedings of the 2006 ACM symposium on Applied computing*, 330-337.

[5] Kossel A. (2004) *eBay-Passwortklau*