International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 03 | Mar -2017 www.irjet.net p-ISSN: 2395-0072

CROSS SITE SCRIPTING ATTACKS AND PREVENTIVE MEASURES

Dr. G. Rama Koteswara Rao[1], K.V.J.S. Sree Ram[2], M. Akhil Kumar[3], R. Supritha[4], S. Ashfaq Reza[5]

[1]Professor, [2,3,4,5]IV/IV B.Tech
Dept. of Information Technology
VELAGAPUDI RAMAKRISHNA SIDDHARTHA ENGINEERING COLLEGE, Kanuru, A.P, India

---

Abstract - *Cross site scripting is an injection type attack where an attacker injects malicious scripts into the web pages. These scripts will be inserted into the pages through search fields, comment fields, guest books etc. The core intention of this attack is to steal the sensitive data of the other users who are visiting the same website. This is one of the most prevalent vulnerabilities in web applications and also a browser exploit that takes advantages of malicious JavaScript. Input validations and Code filtering are the most important areas where most of the executions of malicious script can be blocked.*

Keywords: Cross site scripting, vulnerability, attacks, input validation, malicious script, code filtering.

1. INTRODUCTION

Web applications are generally classified into two types; they are static web applications and dynamic web applications. Static web applications are those which does not interact with server (or database) and display the static content to the users. Dynamic web applications are those which interact with the server and satisfy the request of the client, for example, a sample login page which verifies the username and password of the user by interacting with the database in which the user credentials are stored [1].

Cross site scripting attacks are the type of attacks which enables the attackers to steal the client side sensitive information like cookies etc.. These kind of attacks are generally done by injecting the client side vulnerable scripts into the areas which communicate with the servers or the databases like search fields, comment box etc.. By stealing user sensitive information attackers can bypass the access controls like same origin policy [2].

1.1 TYPES OF CROSS SITE SCRIPTING ATTACKS

There are mainly three types of cross site scripting attacks. They are:

i. **Non persistent Attacks:** It is the most common type of web vulnerability and is also termed as reflected XSS attack or type 1 XSS because the attack is carried out in a single request/response cycle [3]. This attack is done mostly in HTTP query parameters given by the users and is used by scripts on the server side and display the results without sanitizing the query[4]. These attacks are easy to identify and attacker initially checks whether a particular web application is vulnerable or not by performing these attacks. These attacks are not so devastating since these do not show impact on the server.

ii. **Persistent Attacks:** It is the more dangerous type of XSS attack and is commonly termed as stored XSS attack or type 2 XSS because the attack is carried out in two requests one for injecting the malicious code and store it in the web server and the other for the users(victims) to load the page which is malicious[5]. In this attack, the attacker stores the malicious script on the server side permanently and when the users unknowingly or without proper knowledge make the script active he/she will be a victim of the attack[4].

iii. **DOM based Attacks:** In these attacks, the vulnerability appears in the document object model. In type 1 and type 2 XSS, the dangerous payloads are in the response page but in this type of attack, the dangerous payload is not in the response page and the source code of the HTML page is similar to the response page. These attacks are done by the use of document.write() and other such similar functions[6].

2. LITERATURE SURVEY

In 2012, Takeshi Matsuda worked on "Cross Site Scripting Attacks Detection Algorithm Based on the Appearance Position of Characters" [7]. In order to prevent XSS attacks, they proposed a new detection algorithm which works on extracting an attack feature considering the appearance position and symbol frequency. The disadvantage of this approach is it requires learning of detection threshold and since this algorithm works best after testing against training test samples we cannot completely ensure the web application is secured.

In 2013, Michelle E Ruse et al. proposed a two-phase technique to detect XSS vulnerabilities and prevent XSS attacks[8]. In the initial phase, the web application is translated into a language for which recently developed concolic testing tools are available which also identifies input and output variables that are helpful in generating test cases of determining input/output dependencies in the application. In the second phase, monitors are used to check

the vulnerabilities at the run time. The disadvantages are this technique is useful for the web applications developed in java and doesn't work well for those applications in other programing languages like PHP etc..

In 2014, Guowei Dong, YanZhang, XinWang, Peng Wang, Liangkun Liu worked on "Detecting Cross Site Scripting Vulnerabilities Introduced by HTML5" [9]. They have done a systematic analysis on tools and attributes and identified XSS attack vectors related to HTML5. A XSS repository is constructed and a dynamic tool is implemented depending on these vectors. The disadvantage of this is: Since it is based on analysis, although all the Webmail systems have a respective XSS vulnerability filtering mechanism, if a new XSS vector appears this mechanism doesn't respond

In 2015, Shashank Gupta and B.B.Gupta conducted a survey on the various journals on "Cross Site Scripting attacks and Defense mechanism" [10]. They have analyzed the major concerns for web applications and Internet-based services which are persistent in several web applications and highlighted some of the serious vulnerabilities found in the modern web applications.

3. PROPOSED ALGORITHM

3.1 Script filtering Algorithm

This algorithm works best because here the mechanism implemented deals with input given by the user. Whatever is the input given by the user is sanitized properly and displayed to the user.

Step 1: consider user input

Step 2: while(given user input)

If(user input contains any HTML specific tags)

Sanitize the input and store in the database

If(user input contains any special symbols)

Sanitize the input and store in the database

If(user input contains any script tags)

Sanitize the input and store in the database

If(user input contains any DOM objects)

Sanitize the input and store in the database

If(user input contains window objects or document objects)

Sanitize the input and store in the database.

If(user input contains any styling related code)

Sanitize the input and store it in the database.

Step 3: Take the user input and goto step 2

Step 4: Display the results.

3.2 Flow diagram:

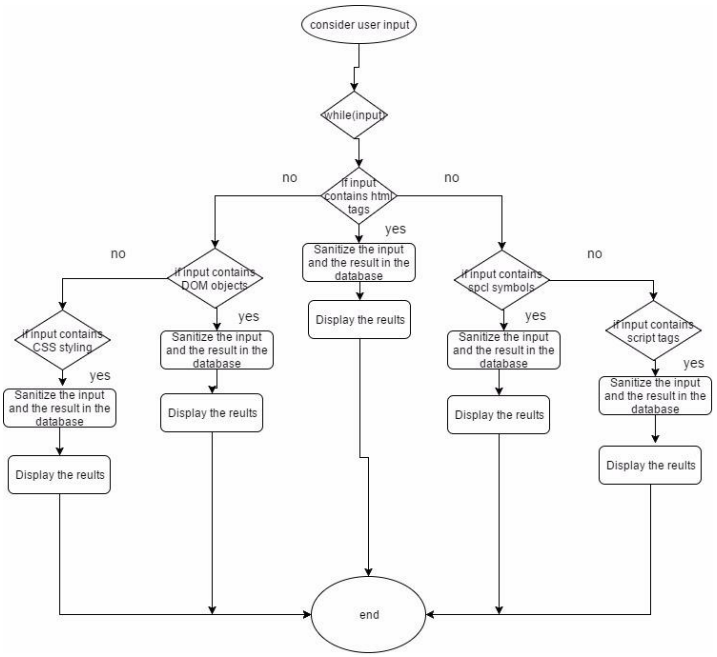

Fig: 1 flow chart for script filtering algorithm

3.3 Algorithm implementation

For an attack to happen, the attacker tries to find the user input areas. The user input is given such priority because it is the only way for the user or client to interact with the server. So if the attacker can be successful in injecting the malicious code into the server an attack is guaranteed to happen. In order to prevent the attacker to have that privilege, we sanitize the user input. As shown in Fig 1, we initially consider the user input. If the user input contains any HTML specific tags like "<i>,
, <a> etc.. " we sanitize the request and store it in the database. If the user input contains any special symbols which are generally used in script functions, they should be sanitized. If the user input contains any script tags which are one of the most serious ways of an attack to be possible, they should be properly sanitized. If the user input contains any styling related code then filter the code and store it in the database. Finally, we have restricted the redirection of a specific web application page to some other page through which we can stop most of

the attacks. This can be done by sanitizing the user input if it contains any window.location or document.refferer methods. If the above methods are not followed, the attacker tries to steal the valuable information of the users like cookies. Usually, if we consider any login page example sessions will be created for every user. The flaw of any browser is that it stores the session id in the form of a cookie. So, if the attacker steals this cookie he can enter into the web application as an authorized user and the results can be more devastating.

4. EXPERIMENTS AND RESULTS

An attacker initially checks whether a web application is vulnerable or not as shown in Fig 2.

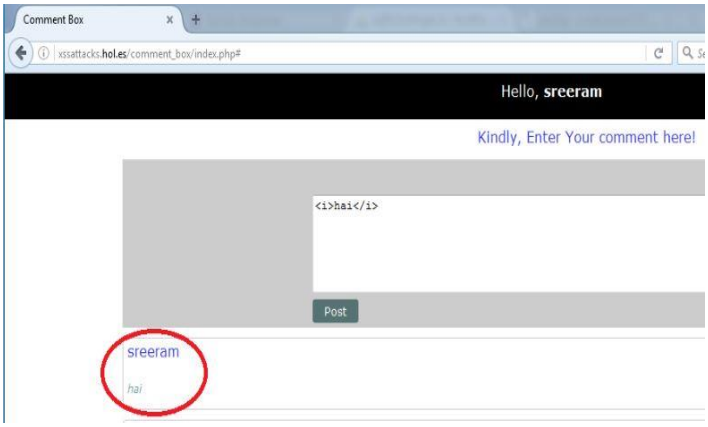

Fig: 2 checking for vulnerability

Since the web application is vulnerable, the attacker tries to inject the malicious code into the server which is a persistent XSS and can be devastating as shown in Fig 3.

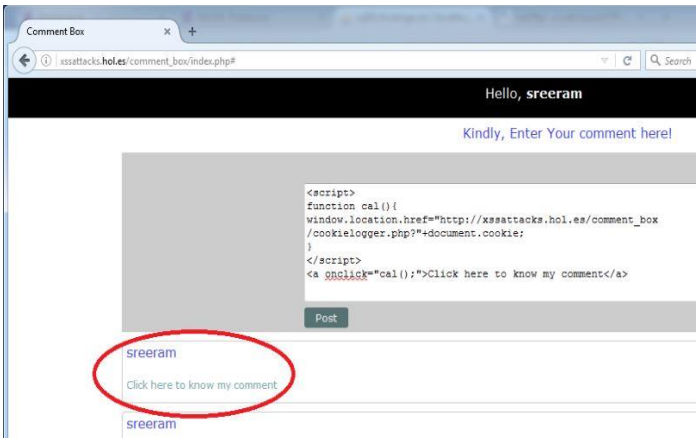

Fig: 3 Injecting malicious script

When an authorized user is logged into the web application as shown in Fig 4, and unknowingly clicks the link he will be redirected to a blank page which he feels like a dummy comment but the attack is done as shown in Fig 5.

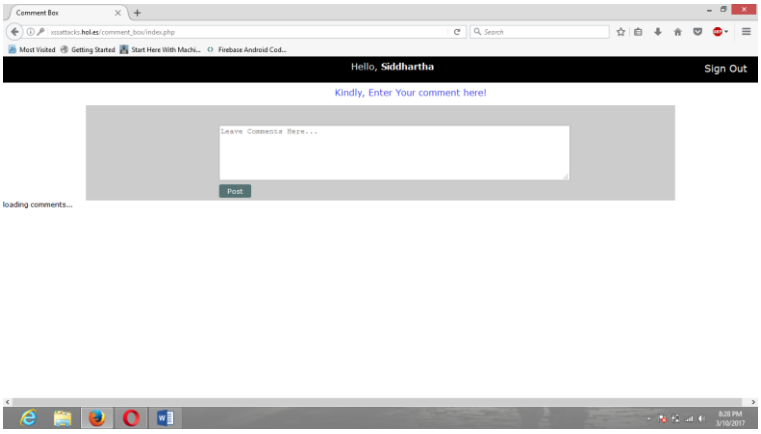

Fig: 4 Authorized user login

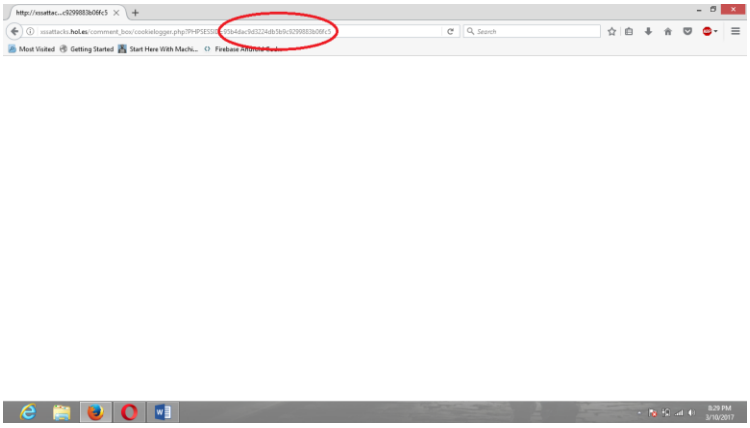

Fig: 5 Stealing cookies

This cookie is stored in attacker's database as shown in Fig 6 and the attacker can use this cookie to login as the authorized user.

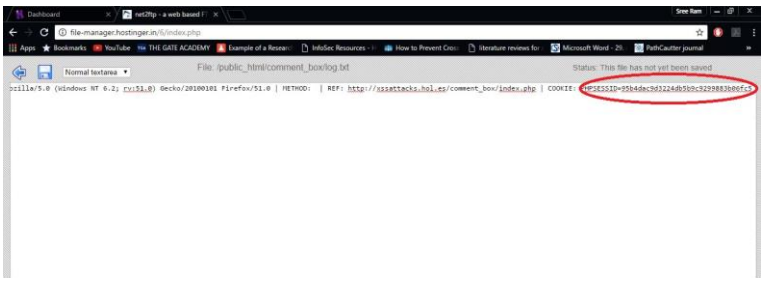

Fig: 6 cookies stored at attacker side

In order to prevent this kind of attacks, we use code filtering algorithm which converts the given text into a plain text format and displays the result as shown in Fig 7.

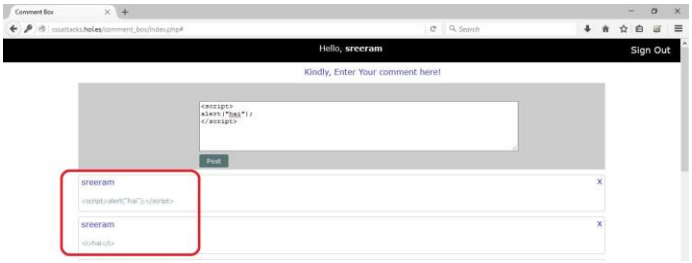

Fig: 7 Filtered script

5. CONCLUSIONS

In this paper, we tried to restrict the XSS attacks with the help of code filtering algorithm. This algorithm works fine because it allows no script to store in the database and thus no script can be made executed. But, in this paper we made our efforts to reduce the XSS attacks by means of cookie stealing which is not the only way of performing XSS attacks. We would like to implement the same algorithm to restrict attacks done through key logging etc..

REFERENCES

[1] Okin, Jonathan Robert. The information revolution: the not-for-dummies guide to the history, technology, and use of the World Wide Web. Ironbound Pr, 2005.

[2] Barth, Adam. "The web origin concept." (2011).

[3] http://www.acunetix.com/blog/articles/non-persistent-xss as accessed on 11 March 2017.

[4] Jayamsakthi Shanmugam, Dr M. "Cross Site Scripting-Latest developments and solutions: A survey." Int. J. Open Problems Compt. Math 1.2 (2008).

[5] http://www.acunetix.com/blog/articles/persistent-xss as accessed on 11 March 2017.

[6]http://www.acunetix.com/blog/articles/dom-xss-explained as accessed on 11 March 2017.

[7] Matsuda, Takeshi, Daiki Koizumi, and Michio Sonoda. "Cross site scripting attacks detection algorithm based on the appearance position of characters." *Communications, Computers and Applications (MIC-CCA), 2012 Mosharaka International Conference on*. IEEE, 2012.

[8] Ruse, Michelle E., and Samik Basu. "Detecting cross-site scripting vulnerability using concolic testing." *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*. IEEE, 2013.

[9] Dong, Guowei, et al. "Detecting cross site scripting vulnerabilities introduced by HTML5." *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on*. IEEE, 2014.

[10] Gupta, Shashank, and B. B. Gupta. "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art." *International Journal of System Assurance Engineering and Management* (2015): 1-19.