Cross Site Scripting (XSS) Attack Prevention And Detection

Yogesh R. Nagargoje, Csmss Chh. Shahu College Of Engineering, Aurangabad, India.

ABSTRACT: Cross-Site Scripting (or XSS) is a security attack that occurs when an attacker uses another's browser to run a malicious script. It is called "cross-site" because it involves the interactions of two or more sites. The "scripting" in the name comes from the injecting of malicious scripts. There are several types of Cross-Site Scripting attacks that can occur: reflected, stored, and DOM-based. With the growing technology, and creation of JavaScript in 1995, hackers began to discover the vulnerabilities of JavaScript. There are solutions to these attacks on the levels of client-side and server-side which can complete each other's to provide protection for the website and web applications to prevent malicious scripts from being implemented.

Keywords – CSS, Detection, Scripts, Cross Site, XSS.

I. INTRODUCTION

Cross site scripting attacks this is likewise known as XSS, it is a more widespread and high risk web application security issue which is used against internet or client web browser. These types of attacks give privilege to attacker to inject client side script into web pages which is seen by the other user. Occur whenever an application contains data that started from a user and transmits it to a Web browser without first properly validating or encoding that content. With the aid of this type of attack attacker execute scripts in the target browser, which in result user session can hijack, deface websites, port scan internal networks, and conduct phishing attacks and access control over the user's browser using scripting malware.

Objective of Project

Cross-site scripting (XSS) is probably the most prevalent high risk web application vulnerability nowadays, and yet it is still one of the most overlooked by developers and defenders alike. So our objective is to make aware people about:

- What is XSS?
- How does a Cross-site scripting attack occur?
- How we can prevent it?

Problem definition

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site. Web application expands its usage to provide more and more services and it became more useful of the essential communication channels between service providers n users. Users mostly use the scripting language is JavaScript and increasing the use of JavaScript also directly increases the serious problem of security vulnerabilities in web application too. Class of Scripting is injected into dynamic pages of trusted sites foe transferring sensitive data of third party. And it avoids same origin policy or cookie protection mechanisms to allow attackers to access confidential data.

II. LITERATURE SURVEY

Mr. Ismail et.al.[1] present an client side proxy for solving both persistent and non persistent attack that he can compare request and response character and if it detects any reflection of malicious characters, then these are disabled. This client side can protect only against a reflected cross site scripting attacks and therefore does not complement our gateway well. It does not prevent cross site request forgery attack and rewrite the server's response.

SessionSafe[2] present a server side solution to session theft via (reflected and stored) cross site scripting attacks, i.e. the solution does not prevent cross site scripting attacks per se, but rather prevent successful attacks from stealing the session. Other attacks via cross site scripting, e.g. request to create malicious transaction are still possible, e.g. by modifying the input. Also cross site request forgery attacks are still possible.

Jovanovic et.al.[3] describe a server side solution for preventing cross site request forgery attacks. It does not prevent cross site scripting attacks, but it does not rely on the refer string and inserted rewrite request and response by adding a token to the URL…….

Server-side Cross-Site Scripting [19] Detection System is based on passive HTTP traffic monitoring and relies upon the strong correlation between incoming parameter and reflected XSS parameter issues. The set of all legitimate JavaScript's in a given web application is bounded. This forms the basis for two novel detection approaches to

 DOI : 10.18231/2454-9150.2018.0284

identify successfully carried out reflected XSS attacks and to discover stored XSS code on the server side.

Scott and Sharp [5] describe a web proxy that is located between the users and the web application, and that makes sure that a web application adheres to prewritten security policies. The main categories of such policy based approaches are that the creation and management of security policies is a tedious and error prone task. Similar to [5], there exists a commercial product called AppShield, which is a web application firewall proxy that apparently does not need security policies.

III. SYSTEM DESIGN

Reflected XSS attack

Summary: The Attacker compromises an application vulnerable to reflected XSS.

Actor: Attacker

Precondition: The Attacker must have access to the application.

Description:

- The attacker creates a link containing malicious script targeting the vulnerable application and makes
- it available to the attacker e.g. by sending an email to the victim containing the link
- The victim clicks on the malicious link.
- The application executes the script in the victims' application.
- The attacker then compromises the victim.

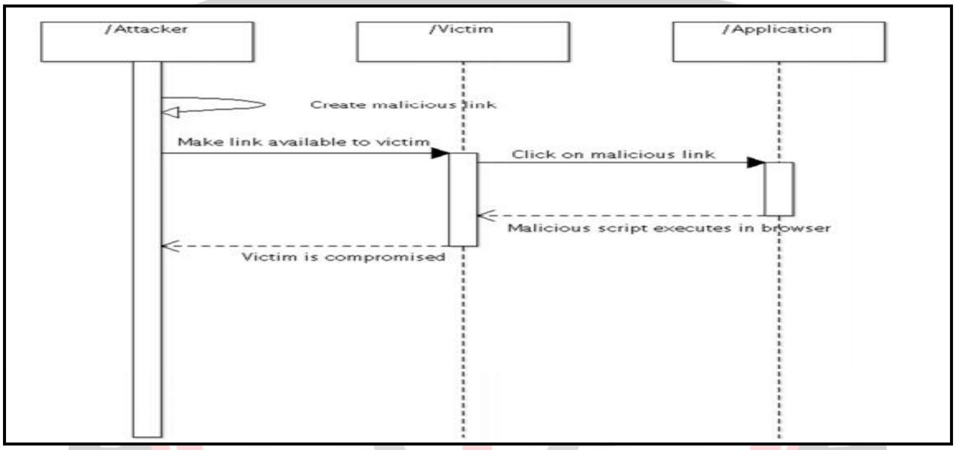

Figure 1: Reflected XSS attack

Stored XSS attack

Summary: The Attacker compromises an application vulnerable to stored XSS.

Actor: Attacker

Precondition: The Attacker must be able to submit input into the application.

Description:

- The attacker submits a text-based script to the application.
- The application stores the script.
- The victim visits an infected page of the application.
- The application executes the script in the victim's browser.
- The attacker then compromises the victim's application.

Post condition:

The victim, who is an application user, is compromised: XSS can be used to steal sensitive information such as usernames and passwords, perform session hijacking, remotely control or monitor the user's browser, poison cookies, impersonate a web page used to gather information, including credit card numbers or used as a pivoting point for other attacks.

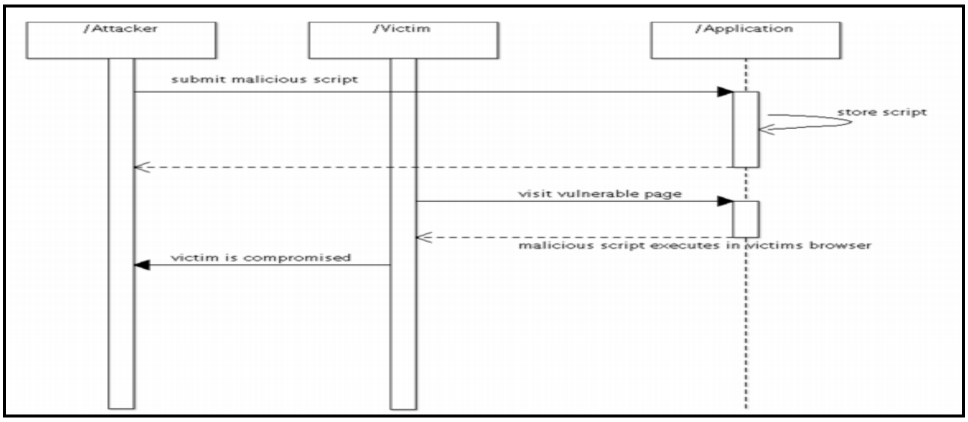

Figure 2: Stored XSS attack

 DOI : 10.18231/2454-9150.2018.0284

ISSN : 2454-9150 Vol-04, Issue-02, May 2018

IV. IMLEMENTATION

XSS REFLECTED

This type of vulnerability compromises the security of the user and not the server. Consists in to inject HTML or Javascript code into a web, in order for a user's browser to execute the code injected at the moment of seeing the altered page when you access it. The form known as reflected is commonly produced in sites that receive information via GET (although the case via POST), such as: searcher.php? d = string_to_search If you suffer this type of vulnerability you could inject code in the browser as follows:

search.php? d = <script type = "text / javascript"> script (); </ script>

In this way the inserted code would not be displayed persistently, but still a user malicious could create a URL that executes the malicious code to later send it to a person and that when executing it, it can subtract cookies or even its password (Pishing).

Exploiting Vulnerability

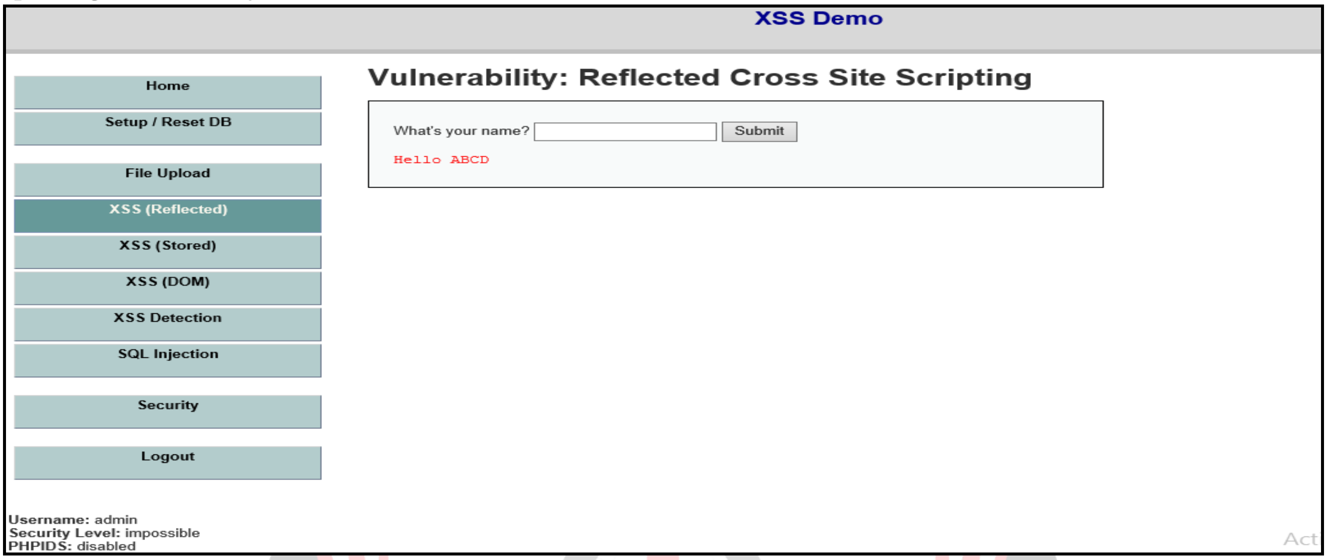

Figure 3 Example of using the web form.

`To exemplify this type of vulnerability, presents a page requesting a name through a form. Once sent, the script shows the text string "Hello sent_name ".

Our code for XSS Reflected only validates that the entered name is not an empty string.

Set Security to low:

Now have a look over a small script which would generate an alert window. So in the given text field for **"name"** I will inject the script in the server.

<script>alert("hello")</script>

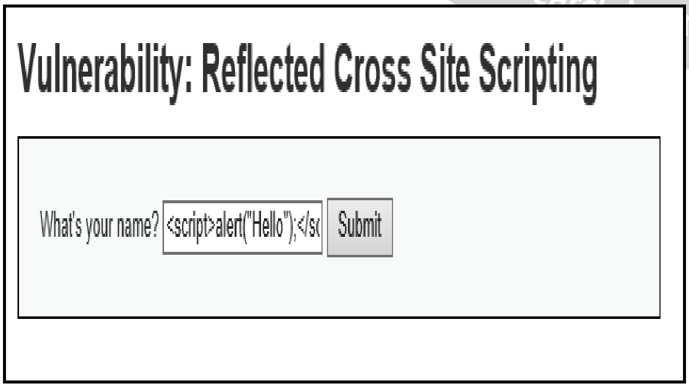

Figure 4: Example of Reflected XSS

Browser will execute our script which generates an alert prompt as showing following screenshot.

In low security it will easily bypass the injected script when an attacker injects it in the text field given for **"name"** which should be not left empty according developer.

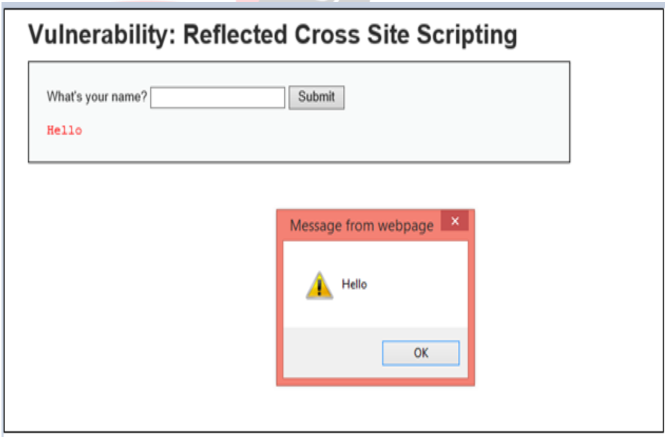

Figure 5: Reflected XSS attack

Set to security High:

In high security the level of security increased where you can easily find preg-replace PHP function is used to perform regular expression to disable the java script.

Preg_replace – Searches string for matches to pattern and replaces them with replacement.

Now above technique will fail as you can see it will search for each and every valid input character for text field and replace invalid character into blank space.

 DOI : 10.18231/2454-9150.2018.0284

To bypass high security level use element of HTML, as you can see I have use image source tag to generate the string inside the web server.

<img src=x onError=alert('xss')>

Prevention for XSS Reflected:

To avoid this type of vulnerability it is recommended to always filter the information coming from the user before making use of it. Usually filtering the characters "<" and ">" would be sufficient, although It is also recommended to filter the names of the labels that can be dangerous in this type of attack like <script>, <object>, <applet>, <embed> and <form>. You can also perform checks to verify that the type of data entered and the length of each field correspond to what was expected. For its part, at the high security level makes use of the htmlspecialchars () function to convert special characters to HTML entities.

In order to better understand its functionality, some examples are shown:

- & (Et) becomes & amp;
- "(Double quotes) becomes & quot; when ENT_NOQUOTES is not established.
- '(Single quote) becomes & # 039; (or & apos;) only when ENT_QUOTES issettled down.
- <(Less than) becomes & lt;
- (Greater than) becomes & gt;

Another countermeasure focused on the users of web pages is to have the browser as much as updated possible. For example, the Internet Explorer browser from its version 8, introduces as novelty an Anti XSS filter that detects possible manipulations of the page by injecting code in a parameter of the URL.

XSS STORED

XSS stored, also called direct or persistent XSS, consists of embedding HTML code or Java script in a web application and can even modify the interface of a website (defacement) As in the case of XSS reflected, this vulnerability compromises the security of the user and not the server.

Detection Rule:

<u>Classification rule:</u>

The algorithm for the detection of cross site scripting attack. We focus attention on the characters which are included in cross site scripting attacks. Now let us prepare to define the detection algorithm. We denote an input string by $l_i$the length of $l_i$by $|l_i|$ (i=1,2,...,I), respectively.

Let us call s_1, s_2 ,...,s_{33} attack feature characters. We define other character as s_{33}, i.e. $s_{33}$are character such as alphabet, number & infrequently symbols in cross site scripting attack. Let us denote$|s_j|$as an appearence frequency in l_i.

Then , we see that$|l_i| = \sum_{i=1}^{33} |s_i|$

Example 1: Let l be

<imglowsrc=javascript:alert(2)>

Then, we have $|s_2|=1$, $|s_4|=1$, $|s_5|=1$, $|s_6|=1$, $|s_8|=1$, $|s_{10}|=1$, $|s_{11}|=1$, & $|s_{33}|=25$.

We define

$$x_i = \frac{\sum_{j=1}^{j+1} s_j \hat{a}_j}{\sum_{j=1}^{j+1} s_j a_j}$$

for all input l_i. Here a_j is non negative real number &

$$\hat{a} = \begin{cases} a_j & (if\ l_i\ otherwise\ s_j) \\ 0 & (otherwise) \end{cases}$$

For some. $\alpha \in [0,1]$.We assume that the input l_i is a cross site scripting attack (resp. l_i is normal)if $x_i > \alpha$ (resp.$X_i \leq \alpha$)

Variable	Candidates of symbols
s1	“(double quotation mark)
s2	>(greater than sign)
s3	/(slash)
s4	<(less than)
s5	space
s6	=(equal)
s7	'(single quotation mark)
s8	:(colon)
s9	.(period)
s10	((left parenthesis)
s11	)(right parenthesis)
s12	-(hyphen)
s13	;(semicolon)
s14	(yen sign)
s15	&(ampersand)
s16	{(left brace)
s17	}(right brace)
s18	#(hash mark)
s19	+(plus)
s20	!(excramation mark)
s21	,(konma)
s22	@(atmark)
s23	?(question mark)
s24	](right bracket)
s25	[(left bracket)
s26	_(underscore)
s27	(tilde)
s28	*(asterisk)
s29	|(vertical bar)
s30	^(caret)
s31	%(percent)
s32	$(dollars)

Table 1: Special character use to calculate threshold

Calculation of important degree of symbol

The calculation method of important degree of characters. we can assume that $l_i$is composed of $|l_i|$ symbol $l_{i,0}$, $l_{i,1}$ ..., $l_{i,|l1|}$. The case that $l_i$include A_j symbol S_j is considerable.

 DOI : 10.18231/2454-9150.2018.0284

ISSN : 2454-9150 Vol-04, Issue-02, May 2018

So we define the symbol $s_j^{(1)}, s_j^{(2)}, s_j^{(a)}, \dots, s_j^{(Aj)}$ at the appearance position by the order of appearing in l_i.

Example 2:

Let l_i be

¡SCRIPT SRC=http://ha.ckers.org/xss.js¡/SCRIPT.

Then, we show the label symbol of l_i in the following table. The input l_i include two symbols S_{18} (<),so we have $s_{18}^{(1)} = l_{i,1}$,(k=1) and $s_{18}^{(2)} = l_{i,40}$ (k=40)

Table: Example of the labeled character in l_i

$$If\ l_{i,k} = s^a{}_j (1 \le k \le |l_i|), then$$

$$E(s^a{}_j, l_{i,i_0}) = \frac{|l_i| - |k - i_0|}{\frac{1}{2|l_i|}\{|l_i| + 2i_0(i_0 - 1)\}}$$

theabove equation is based on the related word extraction algorithm. We define the important degree of symbol

$$E(s_j, l_i) = A_j \times \frac{1}{|l_i|} \sum_{a=1}^{A_j} \sum_{i_0=1}^{|l_i|} E\left(s_j^{(a)}, l_{i,i_0}\right)$$

$$rank(s_j) = \frac{1}{N} \sum_{i=1}^{N} E(s_j, l_i)$$

4. PERFORMANCE ANALYSIS

Behavior of system on different security levels:-

Security Level LOW:-

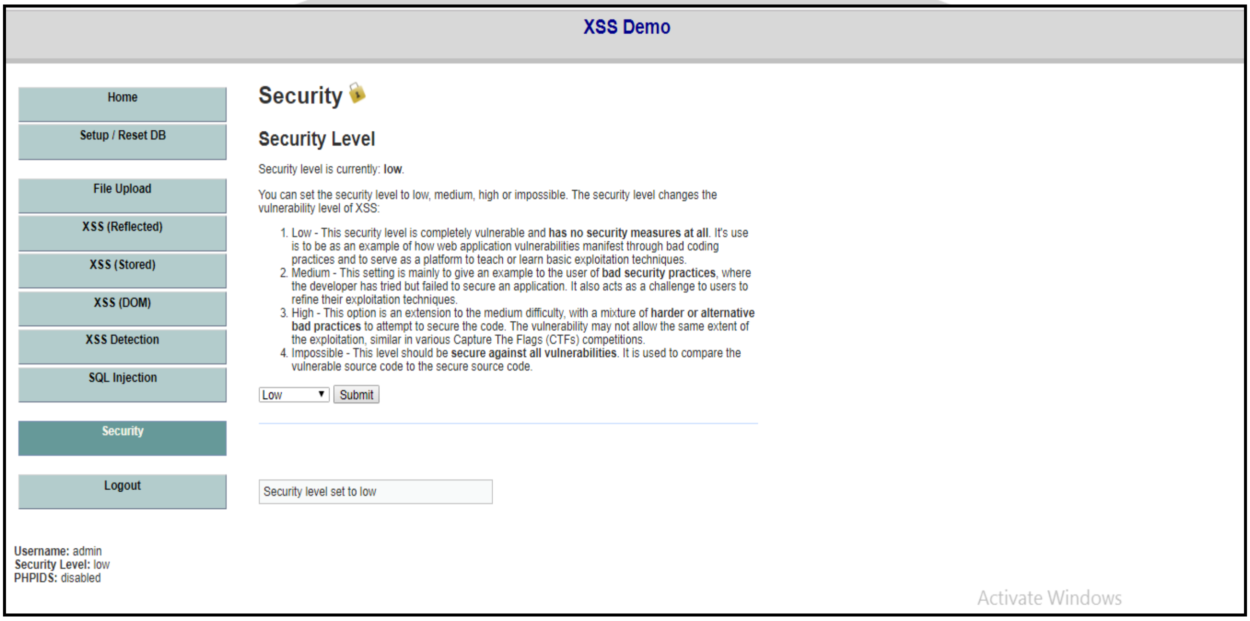

Fig 6 Set Security Level

V. CONCLUSION

Cross-site scripting is one of the most dangerous and most common website vulnerability on the internet. An XSS attack comes in many forms that range from something as small as pop up in a window, to something as destructive as a virus or aworm, and even worse; XSS is capable of compromising a person's identity. Nobody in this world is ever completely safe from it. As XSS vulnerabilities continue to grow, the best way to protect yourself against it is to always be on the alert, and be aware of what you should do when you come across it. In this project we given a demo of different XSS attacks so that everyone get severity of XSS attack. In detection we consider the appearance of position and frequency of characters in input string which detect almost XSS attack through input.

REFERENCES

[1] O.Ismail,M.Etoh, Y.Kadobayashi, and S. Yamaguchi. A proposal and Implementation of Automatic Detection/Collection System for cross site scripting Vulnerability. Proceeding of the International Conference on advance Information Networking and application.2012

[2] M.JohnsSessionSafe: Implementing XSS Session Handling Proceeding of Europeon Symposium on Research in computer security,2006.

[3] N. Jovanovic,E.Kirda, and C.Kruegel. Preventing Cross Site Request Forgery Attacks. Proceeding of IEEE International Conference on Security and Privacy Communication Networks,2006.

[4] Martin Johns, Bjorn Engelmann, and Joachim Posegga, "XSSDS: Server-side Detection of Cross-Site Scripting Attacks," proc. IEEE Computer Security Applications Conference, pp. 335–343, October 2008.

[5] D. Scott and R. Sharp," Abstracting Application-Level Web Security," In Proceedings of the 11th International World Wide Web Conference (WWW 2002), May 2002.

 DOI : 10.18231/2454-9150.2018.0284