Marathwada Shikshan Prasarak Mandal’s

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

**Seminar Report**

**On**

**Cross Site Scripting**

Submitted By

**Onkar Shinde (36129)**

**Dr. Babasaheb Ambedkar Technological University**

**Lonere (M.S.)**



Department of Computer Science and Engineering

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

(2019- 2020)

**CERTIFICATE**

This is to certify that the Seminar entitled “**Cross Site Scripting**” submitted by **Onkar Shinde** is a bonafide work completed under my supervision and guidance in partial fulfillment for the award of Bachelor of Technology (Computer Science and Engineering) Degree of Dr. Babasaheb Ambedkar Technological University, Lonere.

Place: Aurangabad

Date:

**Mr. Pankaj Durole Sir Mr. S.B. Kalyankar**

**Guide Head**

**Dr. Ulhas D. Shiurkar**

**Director,**

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

**Abstract**

Cross site scripting presents one entry point for attackers to access and manipulate control system security. It takes advantages of web servers that return dynamically generated web pages or allow users to post viewable content in order to execute arbitrary HTML and active content such as Javascript, ActiveX and VBScript on a remote machine browsing the site within the context of client-server session. This potentially allows the attacker to redirect the Web pages to a malicious location, hijack the client-server session, engage in network reconnaissance and plant backdoor programs. Cross Site Scripting is the most common security vulnerability that can be found in web applications of today. Any web applications that is generating an output based on the users input but without validating the content is virtually exposed to XSS. The users input validation by filtering and escaping is the most effective way to prevent the XSS attacks.

###### Contents

1. **INTRODUCTION**
2. **LITERATURE SURVEY**

1. **VULNERABILITIES**
2. **IMPACT**
3. **PREVENTION**
4. **TESTING**
5. **CONCLUSION**
6. **REFERENCES**
7. **ACKNOWLEDGEMENT**

**Introduction to Cross Site Scripting**

Cross site scripting (XSS) occurs when an attacker uses a web application to gather the data from a user. Attackers inject Javascript into an application into an application to fool a users to get data from them. Every month roughly 10-25 XSS holes are found in commercial products and advisiories are published explaining the threat.

Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. Out of the three parties, the victim is the only one who actually runs the attackers code. The website is just a vehicle for an attack and is jot typically affected. An XSS attack can be carried out in a number of ways. As an example, the attacker sends the victim a maliciously crafted URL through e-mail or some other medium via web. When the victim opens the URL in a web browser, the web site renders the page and the script is executed on the victim’s computer. Cross Site Scripting (XSS) attacks occur when a website fails to properly prevent attackers from inserting malicious code into an area that takes user provided data, such as form fields on a webpage, HTTP headers, URLs etc.

This code is then used to attack other users, rather than the actual website or the server it's hosted on. This leads many to underestimate the dangers of XSS, despite the fact that an XSS flaw could allow attackers to record what users are typing, redirect users to malicious sites to capture their credentials and change the content of pages to defraud users, to name but a few. There are three main types of XSS attacks, reflected, stored and DOM based. Reflected are the most common, whilst stored are the most deadly.

**Types of Cross Site Scripting**

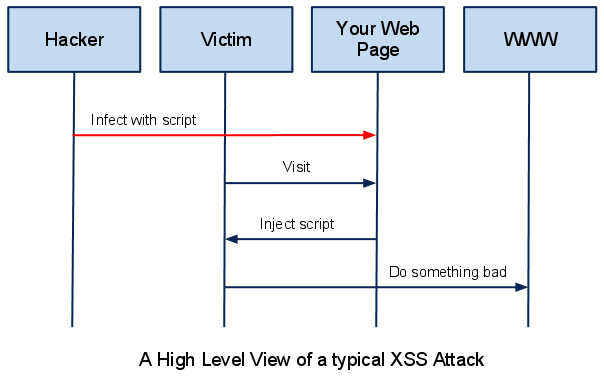
There are mainly three types of cross site scripting attacks. They are :-

1. 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. 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. 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.
2. 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. 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.
3. 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.

## Literature Review

There are two stages to a typical XSS attack:-

1. To run malicious JavaScript code in a victim’s browser, an attacker must first find a way to inject malicious code (payload) into a web page that the victim visits.
2. After that, the victim must visit the web page with the malicious code. If the attack is directed at particular victims, the attacker can use social engineering and/or phishing to send a malicious URL to the victim.



For step one to be possible, the vulnerable website needs to directly include user input in its pages. An attacker can then insert a malicious string that will be used within the web page and treated as source code by the victim’s browser. There are also variants of XSS attacks where the attacker lures the user to visit a URL using social engineering and the payload is part of the link that the user clicks.

As the main purpose of XSS attack is to execute malicious JavaScript in the victim's browser, and there are few fundamentally different ways of achieving that goal.

•**Reflected XSS**: In reflected XSS, the malicious string is part of the victim's request to the website. It might seem harmless a sit requires the victim himself to actually send a request containing a malicious string. But attackers may trick the victims to send the malicious script without informing them. When the attacker targets a specific individual, group the attacker can send the malicious URL to the victim (using e-mail or instant messaging, or social networking link or link to his own site for example) and trick them into visiting it and stealing or changing their confidential information stored in cookies.

•**Persistent XSS** : In persistent/stored XSS, the malicious string originates from the website's database. 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. Here, malicious code is inputted by attackers into vulnerable web pages and is then stored on the web server for later use. The payload may be served back to other users browsing web pages and is executed in their context, at a later stage. Thus, the victims do not need to click on a malicious link in order to run the payload (as in the case of Non-Persistent XSS); they simply have to visit the vulnerable web page, serving back un-sanitized user input from other web sessions.

•**DOM-based XSS**: DOM XSS is a type of cross site scripting attack which relies on inappropriate handling, in the HTML page, of the data from its associated DOM. Among the objects in the DOM, there are several which the attacker can manipulate in order to generate the XSS condition, and the most popular.

**Various attacks On Vulnerability**

All organizations who maintain a web presence are at risk of being attacked. However, the level of risk is different for each organization with respect to intellectual property or personally identifiable information stored by the organization [26]. The purpose of a web based attack is significantly different than other attacks. XSS is most commonly associated with execution of malicious javascript through web applications . This vulnerability is also utilized as a platform for launching other types of attacks. The other types of attacks include Session Hijacking, Denial of Service(DoS), Defacement, Account Hijacking and Cross Site Request Forgery(CSRF).For the case study of these attacks, a dummy web application is selected. Below described attacks are performed on this dummy web application manually.Test environment was created by installing XAMPP server on our local machine to make it a web server to host our dummy vulnerable application.

When this vulnerability is given some other script to execute, it runs it with same efficiency and shows the confidential information which can be further used to steal the passwords and to perform other malicious activities.

1. Defacement: This is a very popular attack on the web and in this attack, an attacker alters the content of web site with offensive or erroneous graphics and/or text. An attacker can also change the appearance of the page or silently redirect a client to malware hosting server. With Stored XSS vulnerability, the attacker is allowed to store the malicious script at the server which gets executed every time a user visits that page. But here, the script is such that which redirects the user to another page, and displays the page of another web site instead of the original page
2. Session Hijacking: This is again a popular attack on the web and possible to perform with XSS vulnerability. In this attack, attacker uses the session ID received from the Cookies of the user to login without the requirement of the user account and password. Attacker changes the session ID using browser functionalities as shown in figure7, and replaces it with the one received as discussed before to login as that victim. By that way, attacker may easily make some changes to the data and even password of the user, which can further deny the victim to login.

**Impacts Of Cross Site Scripting**

The impact of an exploited XSS vulnerability on a web application varies a lot. It ranges from user's Session Hijacking, and if used in conjunction with a social engineering attack it can also lead to disclosure of sensitive data, CSRF attacks and other security vulnerabilities. By exploiting a cross-site scripting vulnerability an attacker can impersonate the victim and take over the account. If the victim has administrative rights it might even lead to code execution on the server, depending on the application and the privileges of the account.

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.

At Dionach we have experienced a few situations when reporting XSS in penetration test reports as a critical or high risk issue, and the client would come back and say "Ok, you managed to pop up an alert box in the browser, how's that a risk at all for our customers?!?".  Although Dionach’s penetration test reports detail the risks associated with all vulnerabilities, a screenshot is worth a thousand words, therefore the alert box is what the client remembers from XSS.

Here we aim to give some practical examples of how attackers leverage XSS to launch devastating attacks leading to the full compromise of the web application or users' computers and how to minimise the risk.

Cross-site scripting is a flaw that allows users to inject HTML or JavaScript code into a page enabling arbitrary input. There are two main variants of XSS, stored and reflected. Stored XSS allows an attacker to embed a malicious script into a vulnerable page, which is then executed when a victim views the page. Reflected cross-site scripting relies on a victim being socially engineered into clicking on a malicious link, sent via email for example.

## How to prevent XSS attacks

Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.

In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

* **Filter input on arrival:** At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
* **Encode data on output:** At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.
* **Use appropriate response headers:** To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.
* **Content Security Policy:** As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.
* **Escaping user input:** is one way to prevent XSS vulnerabilities from appearing in applications. This means taking the data an application has received and ensuring it's secure before rendering it for the user. Escaping user input prevents key characters in the data that a webpage receives from being interpreted as executable code. This means preventing the browser from interpreting characters used to signal the start or end of executable code, and it translates them to "escaped." For example, quote characters, parentheses, brackets and some other punctuation marks are sometimes used to set off executable code; "escaping" these characters means converting them from single characters that would not be displayed into strings that the browser interprets as printable versions of the characters.
* **Sanitizing user input**: is another way to prevent cross-site scripting attacks, which is especially helpful on sites that allow HTML markup. This prevention method scrubs the data clean of potentially executable characters, changing unacceptable user input to an acceptable format and ensuring the data received can't be interpreted as executable code.
* **Input validation**: makes certain an application is rendering the correct data and preventing malicious data from harming a website, database and users. Validating input helps prevent XSS in forms because it stops a user from adding special characters into webpage data entry fields by refusing the request. Input validation helps reduce the possibility of harmful effects if an attacker should discover such an XSS vulnerability

### How to Test Against XSS

Firstly, in order to test against XSS attack, black box testing can be performed.

It means, that it can be tested without a code review. However, code review is always a recommended practice and it brings more reliable results too. From my software testing experience, I would like to add, that if a good black box testing technique is selected and performed accurately, then this should be much enough.

While starting testing, a tester should consider which website’s parts are vulnerable to the possible XSS attack.

It is better to list them in any testing document and this way we will be sure, that nothing would be missed. Then, the tester should plan for what code or script input fields have to be checked. It is important to remember, what results mean, that application is vulnerable and it analyzes the results thoroughly.

While testing for possible attack, it is important to check how it is being responded to the typed scripts and is those scripts executed or not etc.

### XSS Testing Tools

As Cross Site Scripting attack is one of the most popular risky attacks, there are a plenty of tools to test it automatically. We can find various scanners to check for possible XSS attack vulnerabilities – like, Nesus and Nikto. Both of which are considered as quite reliable.

From my software testing career, I would like to mention SOAP UI tool. It can be considered as a quite strong tool for checking against the possible XSS attacks. It contains ready templates for checking against this attack. It really simplifies the testing process.

However, in order to test for this vulnerability with SOAP UI tool, API level testing should already be automated with that tool. Another solution to test against XSS can be browser plugins. However, plugins are considered as quite a weak tool to check against this type of attack.

Even while testing automatically, the tester should have good knowledge of this attack type and should be able to analyze the results appropriately.

Good knowledge is also helpful while selecting the testing tool. Also, it is important to know, that while performing scanning for security vulnerabilities with an automatic tool, testing manually is also a good practice and this way the tester will be able to see the results and analyze them.

**Conclusion**

By following the basic guidelines on this page you can avoid the most common vulnerabilities that are introduced into code. In general, spending time on input validation and output sanitization and escaping will make your application safe.

When choosing functions for sanitization and escaping, choose the function that most closely matches your specific use case. If you are outputting data into an HTML attribute, use a sanitization or escaping function specific for HTML attributes. This will give you the best combination of application performance and security.

If you are able to avoid XSS vulnerabilities and secure your application output, you will avoid almost half of all vulnerabilities that might be introduced into your application.

Cross-site scripting is a Web-based attack technique used to gain information from a victim machine or leverage other vulnerabilities for additional attacks. The fact that this technique could be used to specifically target and gain access to control system environments has been described in a detailed hypothetical attack scenario. Similar attacks may be mitigated though the application of the seven practices recommended above. These practices employ policy, people, and technology countermeasures to protect against XSS and other Web attacks. Critical infrastructure control system asset owners are encouraged to appropriately apply these practices in their operating environments.

**References**

[1] https://en.wikipedia.org/wiki/Cross-site\_scripting

[2]<https://www.uscert.gov/sites/default/files/recommended_practices/RP_CaseStudy_XSS_20071024_S508C.pdf>

[3] <https://www.exploit-db.com/docs/english/18895-complete-cross-site-scripting-walkthrough.pdf>

[4] <https://www.grc.com/sn/files/A_Web_Developers_Guide_to_Cross_Site_Scripting.pdf>

**ACKNOWLEDGEMENT**

I would like to place on record my deep sense of gratitude to Prof. Pankaj Durole Sir, HOD-Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and Management Studies Aurangabad, for his generous guidance, help and useful suggestions.

I express my sincere gratitude to Prof. Sanjay Kalyankar, Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and Management Studies Aurangabad, for his stimulating guidance, continuous encouragement and supervision throughout the course of the present work.

I am extremely thankful to Dr. Ulhas Shiurkar, Director, Deogiri Institute of Engineering and Management Studies Aurangabad, for providing me infrastructural facilities to work in, without which this work would not have been possible.

**Signature of Student**

Onkar Shinde Sign