International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

XSS Vulnerability Assessment Procedure and Mitigation for Web Application

Y. Prathyusha Reddy, SK. Althaf Rahaman, K.Yasudha

Abstract: ***CSS is one of the foremost routine vulnerabilities that affect many web applications. XSS attacks are essentially malicious injections (client-side) that are added to an internet page or app through user comments, form submissions, and so on. The most danger behind XSS is that it allows attackers to inject content into the online app. The injected content can modify how it's displayed, forcing the browser to execute the attacker's code. Web vulnerabilities are developed for scanning whole webpage of internet sites. Vulnerability Assessment is that the process of identifying vulnerabilities in your application's environment. Vulnerability is defined as a weakness or flaw within the system that permits an attacker or insider to access the system during a way they're not authorized.***

Keywords: ***XSS, Vulnerability, Mitigation, Attack, Malicious.***

I. INTRODUCTION

CSS is one of the security vulnerabilities where the attacker could inject his own client side scripting to our website by using any input fields available on our website. So, when the attacker enters the script into the input fields it's sent to the server and therefore the server thinks it as a traditional input then sends back to the client browser. Now the client browser takes it as a script that must be rendered than data got to be displayed then executes the attacker's code. By using XSS the attackers could deface our websites.

- Getting login into the appliance.
- When server gives response to you it contains two parts:
  a. Static: Input Fields like username, mobile number, balance, email, etc….
  b. Dynamic: Data entered into input Fields.

If user submits the code (html/JavaScript) rather than data into input fields. It goes to the server and if it reflected back to the browser and Browser consider data as code and starts executing an equivalent.

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Y. Prathyusha Reddy *, PG Student, Department of CS, GIS, GITAM (Deemed to be University), Visakhapatnam, India, prathyu661@gmail.com
Sk.Althaf Rahaman, Department of CS, GIS, GITAM (Deemed to be University),Visakhapatnam, India, rahmanalthaf@gmail.com
K Yasudha, Department of CS, GIS, GITAM (Deemed to be University),Visakhapatnam, India, yasudha.p@gmail.com

Key Concepts of XSS:

- XSS could also be a web-based attack performed on vulnerable web applications.
- In XSS attacks, the victim is that the user and not the appliance.
- In XSS attacks, JavaScript is employed for the dispatchment of malicious content to users.

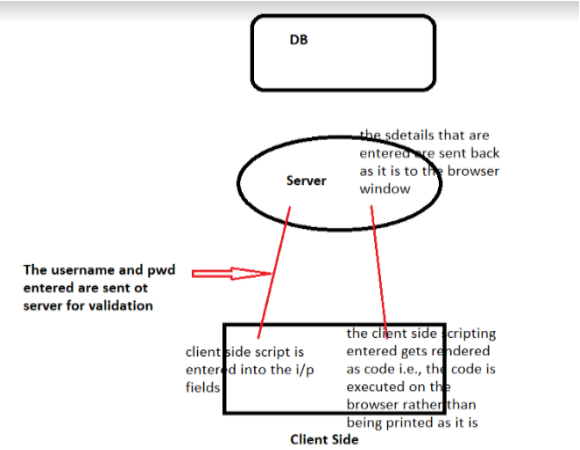


Fig. 1.XSS functionality of Web Application.

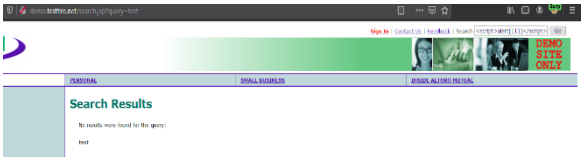
II. XSS (CROSS SITE SCRIPTING) VULNERABILITIES AND MITIGATIONS

XSS (Cross Site Scripting) occurs whenever an application takes untreated data and sends it to the client browser with none validation. This permits attackers to execute malicious scripts within the victim's browser which can end in user sessions hijack; defacing websites or redirect the user to malicious sites. There are three primary kinds of cross-site scripting:

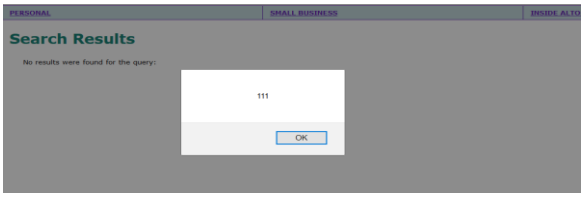
- Reflected XSS occurs when malicious script is shipped from this HTTP request.
- Stored (or persistent) XSS occurs when malicious script is shipped from the website's database.
- Document Object Model (or DOM) based XSS occurs when the vulnerability is on client-side code instead of server-side.

A. Reflected XSS

The type of XSS vulnerability where the attacker's script gets executed after crossing through the server i.e. The user's request for login goes to the server then the injected script gets executed back at the browser.
Example:



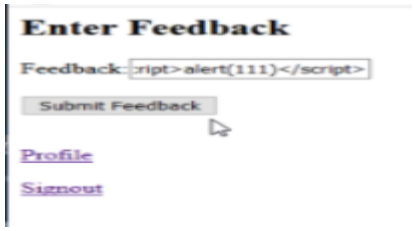
And finally it's got reflected



B. Stored XSS

The type of XSS vulnerability where the injected script doesn't get executed immediately and gets stored within the database is named stored XSS. In this sort of vulnerability; the script that's entered within the fields gets stored within the database and only gets executed when a particular page is accessed.

Example:



And finally it got reflected in database

email	feedback	balance
abc@abc.com	<script>alert(111)</script>	1106
dhinesh@abc.com	Test1	2620

C. DOM (Document Object Model)

The Document Object Model (DOM) could also be a programming interface for HTML and XML (Extensible Mark-up language) documents. DOM defines the logical structure of documents and thus the way a document is accessed and manipulated. DOM could also be because of represent the webpage within the structured hierarchical way so as that it will become easier for programmers and users to glide through the document.

This happens, when the DOM environment is being changed, but the code remains an equivalent .In this sort of vulnerability, the script is entered within the fields doesn't attend the server/database it checks with the DOM based code and shows the response there itself.

Features of DOM

- DOM uses object oriented methods.
- DOM is employed to navigate the resources that you simply are using on the element.
- Here when a user logins with wrong parameters it reflects back on the browser and executes itself.
- DOM Based XSS simply means a Cross-site scripting vulnerability that inherits sight within the DOM (Document Object Model) instead of a neighbourhood of the HTML. In reflective and stored Cross-site scripting attacks you will see the vulnerability payload within the response page but in DOM based cross-site scripting, the HTML ASCII document and response of the attack are getting to be precisely an equivalent, i.e. the payload cannot be found within the response.

Example:

When login with wrong parameters

And click on Transfer

It will be reflected back at the browser page itself without getting to the server side saying that:

Krishna isn't variety

By checking with the DOM code

And the following is that the code for the DOM Based reflection on the online page.

```
function checkinp()
{

var x=document.forms["transfer"]["tacno"].value;

if(isNaN(x))
{

document.write(x+" is not a number.");
return false;
```

III. XSS MITIGATIONS FOR WEB APPLICATION

Mitigations are nothing but the way to stop the malicious attacks on the online applications like Script alerts, injecting malicious codes, performing unauthorized activities and etc.

Types of Mitigations are:

A. Input Validation

Input validation is completed at the server side instead of at the client side.

Reason: I even have something called proxy and it's a feature called Interception in proxy.

B. Client-Side Validation

Validation wiped out the browser is named client side validation.

Reason: What attacker will do is rather than submitting<Krishna he will remove < and just submit Krishna. Here Krishna is a valid user now, client side validation is checked whether is correct or not.

- Yes it is correct input.
- Initiates the request it reaches to the interceptor.
- Opens up the request for me to edit.
- Add <Krishna and send it to the server.
- So, it reflects back and executes.

C. Server-Side Validation

Validation done at server side is called Server side validation.

Reason: Whether you have it in the client side doesn't matter to us from functionality perspective it may requires. At client side we have burp suite to intercept the request. This is how we bypass client side protection mechanisms. Types of server side input validation are:

- When we keep it at server side --Input validation.
- So, what all things i need to avoid <>"'$*.
- When you tell what is not allowed it becomes an unlimited list.
- And you can never put this list.
- This is called black listing.

D. Listing

1. Black Listing: It can be easily bypassed by using esoteric language that we input.

Procedure:

- Picking up dangerous characters what are allowing.
- Identifying them.
- And the characters that are allowed with the help of those characters we use esoteric languages to convert them into specific language set and bypass the security checks, input validations.
- So, always go with white listing.

2. White Listing: If you specify what is allowed then it is called White listing.

Procedure:

- Check it at Server side in php code.
- Easier way to implement the validations or checks with Regex Expressions.
- Instead of if loops, substrings.

IV. CONCLUSION AND FUTURE SCOPE

Now a day's web applications are using altogether round the world. CSS is studied as a deliberate vulnerability in Web applications. A deliberate consequences of those worms on web applications like stealing cookie details, MasterCard number, and password and data breaches. Case studies of Anthem Breach, Face book, eBay and Apple pay are examined with their rules to avoid these attacks. Input encoding can be protected beside more than just XSS. Before storing the information in a database SQL injection and command injection can also be determined. As per the definition of XSS data getting confused as code to avoid confusion between data and code we can encode and send it, which is called output encoding.

As technology moves onward and brings new scenarios, tools, models and methods to boost security levels, hackers are going to be a part of this never end game.

The action to see the code manually and modify the security holes has given significant results for the online site integrity. Such adaptation has changed the condition of the after-test results and therefore the pieces of code that needed to be altered accompanying to the found vulnerability had skilled in activity to be hacking resilient.

Finally, prevention from XSS being exploited is-

- Input validation-At server side.
- Output encoding based on context.

REFERENCES

1. Martin and Justus Winter, "Protecting the Intranet against JavaScript Malware and Related Attacks", volume 4579 of LNCS, pp. 40-59. Springer, July 2007.
2. Klein, A., "DOM Based Cross Site Scripting or XSS of the Third Kind," Technical paper - Web Application Security Consortium, Describes DOM-based (Type 0) XSS.D.
3. XSS(cross side scripting)

   Hint: https://portswigger.net/web-security/cross-site-scripting.
4. Duraisamy, Kannan, &Selvamani. (2010). Protection of Web Application from Cross Site Scripting Attack in Browser Side. IJCSIS, 229-236.
5. Shalini, &Usha. (2011). Prevention of Cross Site Scriptig Attack (XSS) on Web Application in the Client Side. IJCSI International Journal of Computer Science Issue.
6. Nithya, p.lakshmana, and c.malarvizhi, "a survey on detection and prevention of cross-site scripting attack," international journal of security and its applications, vol.9, no.3, pp.139-152, 2015.

AUTHORS PROFILE

Y. Prathyusha Reddy, pursuing Master of Computer Applications, Department of CS, GIS, GITAM (Deemed to be University), Visakhapatnam. Her main area of interest includes Information Security, Application Security and Ethical Hacking.

SK. Althaf Rahaman is currently working as Assistant Professor in the Department of Computer Science, GIS, GITAM (Deemed to be University). His main area of research includes Big Data Analytics, Artificial Intelligence and Computer Security.

K. Yasudha, is currently working as Assistant Professor in the Department of Computer Science, GIS, GITAM (Deemed to be University). Her main area of research includes Machine Learning and Data Mining.