XTrap: Trapping Client and Server Side XSS Vulnerability

Ankit Shrivastava
Department of CSE, SCIT
Manipal University Jaipur
Rajasthan, India
ankitshrivastavaieee@gmail.com

Vivek Kumar Verma
Department of IT, SCIT
Manipal University Jaipur
Rajasthan, India
Vermavivek123@gmail.com

Venkatesh Gauri Shankar
Department of IT, SCIT
Manipal University Jaipur
Rajasthan, India
venkateshgaurishankar@gmail.com

***Abstract*— This paper proposes new technique for distinctive and foreclosing Cross Site Scripting (XSS) Attack are projected by suitably checking the cookies and sessions utilizing keen operators. For this reason, distinctive intelligent operators which might screen treats, sessions, mutilating of websites, addition of threatening substance and phishing attacks viably are projected and actualized. This is proficient by conveying four sorts of savvy specialists to be specific treat screen operator, session screen operator, content screen specialist and phishing screen specialist. All of which are considered in charge of fifty four** viable **perceptive and controlling of abuses and attacks in the client side. Not with standing these client side operators, two server side specialists to be specific coordination specialist and information get to specialist are created and sent within the server side. In this paper, we are concentrating on injection, recognition, and expectation of put aside based XSS reflected XSS and DOM oriented XSS.**

Keywords- XTrap, XSS, Vulnerability, Web Application, DOM.

1 Introduction

In this projected strategy, client and server side arrangements are given to alleviate cross site scripting attacks utilizing sensible specialists. The first most well-liked viewpoint of the server side arrangement is that it with success decreases cross site scripting attacks at the server. In addition, the client side arrangement anticipates, deals with treats, sessions and phishing attacks within the client side. The projected client cum server side arrangement keeps all attacks adequately and shrewdly without depending on web application suppliers. Since, this projected arrangement depends on each client and server sides. This postulation clarifies the arrangements gave at the client as well as server sides and subsequently explains on the systems accustomed channel. Malignant scripts in web applications on server side utilizing rules as a section of growth to the protection at the client side by wise operators. Besides, the cross site scripting attacks are in light-weight of the chance of embedding harmful scripts into web site pages that are to be gave the impression to totally different clients. Therefore another winnowing system is important for separating the malignant scripts that are infused by malevolent clients in web applications.

Cross Site Scripting could be a security bug which will have an effect on internet applications. This bug permits an attacker to inject their own malicious code into markup language pages that are flaunted to the users. On eminent execution of the malicious code, the system or web site action or behavior may be utterly modified. It can also steal user's personal information or may be performed on behalf of the user and one of the utmost communal application-layer internet attacks, it targets scripts embedded in an exceedingly page, executed on the client-side instead of on the server-side [1]. XSS could be a threat that happens due to security flaws of Client Side Scripting languages like JavaScript and markup language. The model of XSS is to handle client-side scripts of a web-app to execute within the order most well-liked by the malicious manipulator. These varieties of manipulations will plant a script passing page that might be dead every and each time once it's loaded, or whenever an associated event is performed.

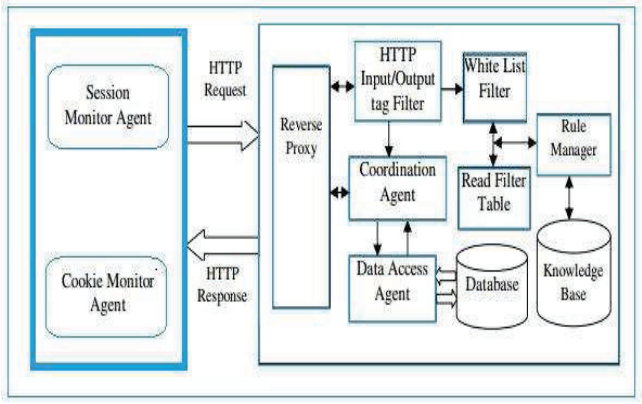

Fig1: Preventing XSS Attacks on Client and Server Side

If a web page accepts information from users and includes it dynamically in web content with no validation then XSS could arise. If a maligned person will do eminent attack then he will gain access to victim's account. Though the maligned person will exploit the victim using the malicious script and bypass the protection among the user's session only. Another methodology to inject XSS is by creating a malicious link with an extreme level. The link does not appear harmful due to the encryption of the malicious portion of code. Generally, attackers send it by doing email, web message board and await clicking on it by victim [2].

978-1-5090-3669-1/16/$31.00 ©2016 IEEE

As we have three forms of Cross-Site Scripting attacks: persistent or stored, non-persistent or mirrored and DOM-based. In DOM primarily based and Non-persistent attack the attackers need a user who visits their malicious web content or click on a malicious link. The hypertext transfer protocol request posts automatically without the information of the victim if the application sends the request using POST technique solely. During this case, vulnerable code are going to be submitted simply through request [3]. Upon submitting the malicious form or clicking on the malicious link, the XSS code can get executed once more and it'll get displayed on user's browser form. In Persistent attacks assailant injects once malicious code using the web type and it's keep within the information. Example message board, review form, webmail messages, internet chat software package and any user input type. The trustful user isn't essentially have to be compelled to move with any extra link, he will merely read the online page containing code.

2 Literature Review

There are many studies and analysis has been worn out previous couple of years to seek out and stop cross-site scripting connected problems within the web application or network requests. Researchers have created multiple solutions however XSS vulnerability still exists within the web application. Web applications still facing several XSS attacks and also the most significant one is session hijacking, attackers steal the victim's cookies details and that they will use an equivalent session.

A technique is employed that embrace security throughout the application life cycle. The application lifecycle phases like design, development, deployment and runtime execution ought to be followed the OWSAP security guideline in a very precise manner. Generally developer lack of information is the reason behind security failure [4].

The ethics of vulnerability analysis explored several security problems that are compromising with user's credentials protection over the network. They advised that the programmers ought to perceive the initial security so we will take away it initially level. And conjointly focuses on risk analysis of every and each situation at initial level [5].

Tools and mechanism are fancied to insure the web application security; it defines the policy that helps developers for developing a secure application. It says cross-site scripting may be a results of improper filtering of user input and instructed careful removal of unwanted scripts or markup language tags that may be executed on the web application. and that they additionally instructed content filtering on the server level and consumer level [6].

An automatic technique is additionally accustomed produce input that exposed the XSS vulnerabilities. an automation tool ANDRILLA is introduced. it's supported input generation technique; it creates the malicious inputs and finds the protection vulnerabilities together with XSS [7].

The attackers attack on the server by manipulating the script, thus content filtering is applied that's using input filtering methodology. However in java script typically filter methodology fails to acknowledge the vulnerable input contents [8].

Sanitization technique is additionally accustomed improve and stop the user input. It consists of DOM, Input filed capture, input sanitizer, links, text area sanitizer and XSS notification. The system follows the bedded approach to stop the vulnerable contents [9].

3 Proposed System Set-up

The projected system could be a study concerning cross-site scripting, there are many ways and approach to preventing the XSS attack however still, and it's not prevented utterly. Therefore we tend to use a graded approach throughout the event method to secure the web application for each client side and server side as a result of one methodology and approach don't seem to be comfortable to handle it.

The proposed levels:

1. Primary level security testing is strongly advised using each black box and white box testing methodology.

2. We can use security testing tools to detect XSS, like Burp Suite is a powerful tool, using secure web application proxy. It can also detect and prevent the security vulnerabilities at first level.

3. As for Client side we should use the signature based mechanism to find the malign java script and verify the known exploits.

4. It uses application level firewall that handles the request between client and server based communication.

5. When we are sending the request we can assign a unique identifier for each request that will differentiate between client and server, and it will be destroyed after completion of a request.

6. We can never allow direct deployment of data into the script, HTML notes, field name, element name, CSS.

7. As the part of special character we may use escape method for HTML, field, java script, JSON, CSS, URL values, to prevent those characters that can be used to insert the script, like &, ‘. ", /, \, <, >, &lt, &gt, &amp, &quot, &#x27, &#x2F and etc, or we can also use auto-creating template system.

8.on other hand, we can apply use of removal method to clean up HTML predefined text.

9. Use of HTTP and add-on based cookies

10.You may advice to the use of spring MVC to develop java based application. Spring is a robust tool and manages many securities parameters.

11. Updated the browser as per the handy use.

4 Web Experiment

The browsers do not allow intersection between sites (cross-site access). Therefore, attacker use many techniques to create a cross- site attack.

Example: We deploy malicious script in our system to steal the cookie on runtime and this could be performed by passing following script:

<script>

window.location = 'http://google.co.in/?cookie = '+document.cookie

</script>

With the navigation of this script the application browser to google.co.in. The URL combines the victim's browser cookie as a query argument. Once the attacker gets the URL feedback with cookie he/she can use it to control the victim's session.

5 Result Analysis

Our projected technique and result are supported the study of past analysis and their execution within the real state of affairs. We tend to tested several web applications and located high priority cross web site scripting problems. Though we've several approaches obtainable currently to stop XSS, however several internet applications are still vulnerable.

On other hand, in past approach commonly developer's uses java script validation or user input filter to forestall client side malicious inputs, for the output they use escape methodology and sanitation methodology. Some safer web applications use application level firewall to filter user request. But still, we tend to don't seem to be ready to stop XSS attacks.

In our analysis, we tend to found that it's not ample within the prevention of additional dangerous XSS payloads. The utilization of one or two existing approaches will stop the direct malicious inputs from the online browser, however not sturdy enough to handle middleware attacks.

In our projected analysis we are exploitation java script validation for user input, java script signature mechanism to spot valid java script, assignment a singular token for client-server request throughout communication, exploitation escape technique to stop script characters, clean-up technique to clean-up hypertext markup language text.

Conjointly we advise the utilization of robust application level firewall, use of HTTPOnly cookie flag, use of correct security testing and conjointly the utilization of scanner tool to notice XSS payloads all told parameters, headers and path, use of robust SPRING tool to develop java application and use of updated application.

This projected system facilitate developer within the handling of XSS problems and it additionally facilitate tester in detection. we tend to instructed this graded approach as a result of one level of security isn't ample within the bar of cross site scripting problems. Our system provides an efficient result for bar of XSS vulnerabilities.

Steps to steel cookie:

1. We are creating a form of our application to deploy the script into the database.

```
<script>
windowlocation="http//google.co.in/
/"cookies'+documentcookie
</script>
```

SAVE ACCOUNT

Fig 2: User input form

2. Next to this, we request a consolidated page from the website.

3. The website contains the malicious code snippet from the DB in the feedback and sends it to the victim.

4. The browser of victim's propagates the malicious script inside the response, sending the victim's cookies to the attacker's side server section.

Fig 3: URL redirection with session id

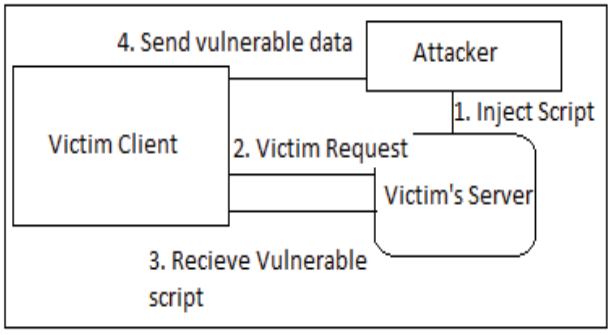

Fig 4: Reflected XSS attacks

Stealing Cookies From Test.Mail.In Using Reflected XSS:

Step 1. Propagating a navigation to the victim:
Button Text: Click Here To Get You Lucky Now
http://mail.test.in/xss/Link_XSS.html
mail.test.in - Victim Website
XSS- Folder that contains JSP file & HTML file

Step: 2 By navigating on above link, the user will redirect to the given link http://mail.test.in/xss/Link_XSS.html, after the JSP file and the HTML file will be retrieved from attacker side server. This link will display cookie as a parameter.

Fig 5: URL redirection with session id

Step: 3 Only hijacker can view this session id on his server console.

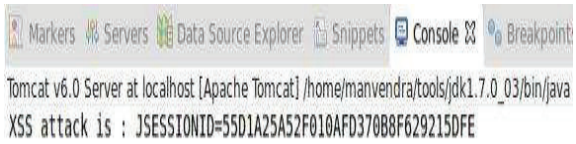

Fig 6: Attacker server receives the session id

Extreme value of the **SetComFonNam** request parameter is added into a JavaScript code snippet which is combined in single quotation marks.

The payload **62370'%3balert(1)//821637fdb** was added in the SetComFonNam parameter. This input was called as **62370';alert(1)//821637fdb** in the application's response.

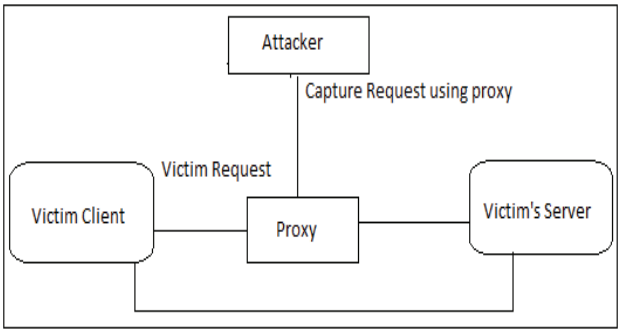

Fig 7: DOM based XSS attacks

The following weblink http://test.mail.org.in contains the below code:

```
<script>
document.write("<b>Current URL<b> : " +
document.baseURI);
</script>
```

http://test.mail.org.in/test.html#<script>alert(1)</script>, it's strong JavaScript code that will get executed, If we look at the view of the page, we won't see <script>alert(1)</script> because it's all happening in the DOM and done by the running JavaScript code.

DOM based XSS using an example of test.mail.org.in.

Step 1: Reload web page and retrieve request in burp suite.

Step 2: Fetch request in burp suite.

Step 3: Modify and create 84960';alert(1)//467ffe1de this as a parameter and forward the request.

Step 4: Hence XSS is executed with script.

Severity:	**High**
Confidence:	**Certain**
Host:	**http://10.11.13.111**
Path:	**/2013/admin/ajax/IntermediateAddUser.jsp**

Fig 8: XSS alert on user's web browser

7 Conclusion

Cross-site scripting is one among the foremost risky web site vulnerabilities. It's accustomed damage internet applications and users. It harms users by causation or inserting malicious code into application info, largely its accustomed perform attacks like session hijacking. We have a tendency to conjointly understand that mending XSS is feasible however we can never be fully certain that nobody can break our filter. Attackers forever realize their ways that to interrupt application security. If we actually need to form a hard-to-crack XSS filter, we've got to investigate additional on all XSS patterns then we will use our interference technique

expeditiously. Once economic analysis and mistreatment higher interference technique, we will stop or fix the damaging xxx web application vulnerabilities. We must always not rely on one technique or approach, we must always use a multilayer security and conjointly we must always have targeted on initial level security. Use of SSL should be counseled to insure secure interaction between client and server.

REFERENCES

1. Amit Singh, S Sathappan: A Survey on XSS web-attack and Defense Mechanisms, Department of Computer Science and Engineering LNCTS, Bhopal, India.
2. Vishwajit S. Patil Department of MCA P.R.M.I.T.& R. Bandera, Amravati Dr. G. R. Bamnote Professor & Head, Department of CSE P.R.M.I.T.& R. Bandera, Amravati Sanil S. Nair Department of MCA P.R.M.I.T.& R. Bandera, Amravati: Cross Site Scripting: An Overview, (ISDMISC) 2011.
3. V. Nithya1 , S. Lakshmana Pandian2 and C. Malarvizhi3: A Survey on Detection and Prevention of Cross-Site Scripting Attack, Vol. 9, No. 3 (2015), pp. 139-152.
4. Meier, John D. "Web application security frame." U.S. Patent No. 7,818,788. 19 Oct. 2010.
5. Bruce Schneier: The Ethics of Vulnerability Research, 2008.
6. David Jonathan Scott: Abstracting Application-Level Security Policy for Ubiquitous Computing, 2004.
7. Kieyzun, Adam, et al. "Automatic creation f SQL injection and cross-site scripting attacks." IEEE 31st International Conference on Software Engineering, Canada.
8. Mishra, Nitin, et al. "Solving False Positive Problem in Client Side XSS Filter."
9. D. K. Patil, K. R. Patil, Ph.D.: Client-side Automated Sanitizer for Cross-Site Scripting Vulnerabilities Department of Computer Engineering, VIIT, Pune Affiliated to SPPU University, Volume 121 - No.20, July 2015.
10. S.SHALINI1,S.USHA2: Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side, Vol. 8, Issue 4, No 1, July 2011.
11.. Tejal V. Kasture, Pinaki P. Dixit, Pooja S. Ovhal, Gayatri Sathe, Neelam A. Zambre : Multiple Prevention Techniques for Different Attacks in Web Application, Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438.
12. Tejinder Singh Mehta , Sanjay Jamwal: Model To Prevent Websites From XSS Vulnerabilities, Vol. 6 (2) , 2015, 1059-1067.
13. Santosh Kumar Singh1, Rahul Shrivastava: BYClient Side Filter Enhancement using Web Proxy, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Impact Factor (2012): 3.358 Volume 3 Issue 7, July 2014.
14. Punam Thopate, Purva Bamm, Apeksha Kamble, Snehal Kunjir, Prof S.M.Chawre: Cross Site Scripting Attack Detection & Prevention System, Volume 3 Issue 11, November 2014.
15. V. S. Patil, D. G. R. Bamnote, and S. S. Nair, "Cross site scripting: An overview," *IJCA Proceedings on International Symposium on Devices MEMS, Intelligent Systems and Communication*, no. 4, pp. 19–22, 2011.
16. V. Nithya, S. L. Pandian, 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. 139–151, 2015.
17. J. D. Meier, "Web application security frame," Oct. 19 2010, uS Patent 7,818,788.
18. B. Schneier, *Bruce Schneier on Trust Set*. John Wiley & Sons, 2014.
19. D. J. Scott, "Abstracting application-level security policy for ubiquitous computing," Ph.D. dissertation, University of Cambridge, 2005.
20. A. Kieyzun, P. J. Guo, K. Jayaraman, and M. D. Ernst, "Automatic creation of sql injection and cross-site scripting attacks," in *2009 IEEE*
31st International Conference on Software Engineering. IEEE, 2009, pp. 199–209.
21. N. Mishra, S. Chaturvedi, C. Dewangan, and S. Jain, "Solving false positive problem in client side xss filter," 2014.
22. D. Patil and K. Patil, "Client-side automated sanitizer for cross-site scripting vulnerabilities," *International Journal of Computer Applications*, vol. 121, no. 20, 2015.
23. S. Shalini and S. Usha, "Prevention of cross-site scripting attacks (xss) on web applications in the client side," *IJCSI International Journal of Computer Science Issues*, vol. 8, no. 4, 2011.
24. T. V. Kasture, P. P. Dixit, P. S. Ovhal, G. Sathe, and N. A. Zambre, "Multiple prevention techniques for different attacks in web application."
25. T. S. Mehta and S. Jamwal, "Model to prevent websites from xss vulnerabilities," *IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1059–1067, 2015.
26. S. K. Singh and R. Shrivastava, "Client side filter enhancement using
web proxy," 2014.