Detection Model For XSS Attack

Leena Jacob[#1],Madhumita Chatterjeer[#2], Virginia Mary Nadar[#3]

[#1,2]*Department of Computer Engineering*

[#3]*Department of Information Technology*

Mumbai University

PIIT,New Panvel,India

[1]leenajacob@mes.ac.in

[2]mchatterjeee@mes.ac.in

[3]virginianadar@mes.ac.in

***Abstract*—Nowadays, Internet is at wide use by many organizations due to which the rise of web applications on the internet is also increasing which ultimately give rise to various kinds of attacks on these web applications. Due to feature rich HTML content provides a way for including malicious attack vectors for web-related attacks. The Cross-site scripting attack exploits the security problem in modern web applications. The occurance of these attacks is due to the injection of malicious scripts as the user-injected input is not validated which ultimately leads to the exploitation of vulnerabilities present in the source code of web applications. We propose a detecting model for Persistent as well as Non-Persistent XSS attacks that works both at the client and server side.**

Keywords— Cross-site Scripting , XSS, vulnerability, Persistent XSS, Non-Persistent XSS

I. INTRODUCTION

Web Applications can be defined as a program that is installed and running in remote web server that responds to requests via HTTP protocol. Web developers create the web applications but they are not well aware about the security concerns which thereby create a loophole for the vulnerability. A vulnerability means existing weakness in the system which allows the attacker to access the data or gain complete control of the application. The main motive of Cross-site Scripting attacks is to take hold of sensitive information such as to steal the cookies, password theft and other private credentials of the user.It is essential to secure the web applications from XSS attack thereby ensuring protection to the confidential details of the client.In this paper we propose a system that is capable of detecting Persistent as well as Non-Persistent XSS attack both at the client and the server side.

Cross-site Scripting attack is a code injection attack. It exploits weakness present in websites or web applications which allows the malicious users to infix their client side code into those web pages. When this malicious code along with original web page gets displayed in the client's browser thereby allowing the attacker to control the page. XSS attacks consists of three types namely Persistent XSS attack ,Non-Persistent XSS attack which is also known as Reflected persistent XSS attack and DOM-Based XSS attack. The persistent cross-site scripting attack in which the origination of malicious string is from the website's database. The Non-Persistent XSS attack in which the origination of malicious code is from the victim's request. The DOM-Based cross-site scripting attack is also known as 'Type 0' attack where the vulnerability can reside either in the client-side code rather than in the server-side code.

The organization of the paper is as follows: Section I gives introduction Cross-site scripting attack and its types. Section II about Literature Survey,Section III explains the proposed system .Section IV gives the conclusion of the study.

II. LITERATURE SURVEY

We highlight the relevant related works on XSS attack detection which is either at client or the server side. The literature summary provides different detection approaches either for persistent or non-persistent cross-site scripting attacks.

The authors [1] has proposed first client side solution for mitigating against XSS attacks. The Noxes tool acts as a web proxy and utilizes both manual and automatically generated rules. The user is allowed to create the filter rules for web requests.The rules can be created in three ways: manual creation, firewall prompts and snapshot mode.The disadvantage of this tool is it suffers from low reliability and the inclusion of benign HTML is prohibited. It requires user intervention to accept or deny requests.

Prithvi Bisht and V.N.Venatakrishnan [2] provides a prevention mechanism for XSS attacks on the server side.The shadow pages are generated for every HTTP response in this approach. The purpose of shadow pages is to obtain intended set of authorized scripts that match up with HTTP response.The disadvantage of this approach is it attempts to sanitize unsafe output but influences the web browser parsers to infer unsafe HTML data.It is vulnerable to threats that utilizes browser parse quirks.

Peter Wurzinger,Christian Platzer,Christian Ludl , Engin Kirda and Christopher Kruegel[3] has proposed a server-side mechanism for detection and prevention against XSS attacks.This technique comprises of a reverse proxy which will intercept all the HTML responses.It overcomes the disadvantage of XSS-Guard by detecting malicious embedded javascript code. It can detect the aberrant between benign and injected javascript code. This approach is not suitable for high performance web-service and it is limited to detect maliciously injected content to javascript.

Hossain Shahriar and Mohammad Zulkernine [4] has proposed a server side framework for detection of XSS attack based on boundary injection and policy generation. It suffers from zero false negative.This technique is efficient and does not require any modification of server and client side entities but response delay increases due to increment of policy checking.

Imran Yusof and Al-Sakib Khan Pathan [5] has proposed a client side solution by applying the pattern filtering approach to prevent persistent XSS attack. This approach is effective but these rules does not work for non-persistent XSS attack. If the filtering module fails o detect any malicious script then it will stored and executed in the database.

Shashank Gupta, B.B Gupta [6]proposed a server side approach for detection and mitigation of XSS attacks in javascript code.It is based on injecting features, generating rules and allows insertion of sanitization routines for the discovery of XSS attacks. The drawback of XSS-Safe is it detects the relationship between stored and injected features in the source code of Javascript.

Imran Yusof and Al-Sakib-Pathan[7] proposed a technique that blocks malicious scripts from loading and executing with a variation of the content security policy (CSP), which provides server administrators with a white list of accepted and approved resources. This approach protects the website visitor from unauthorized downloading of the sensitive data stored in their browsers. The drawback of this approach is it will work only if the browser supports CSP and primary defense must involve validating the user inputs and encode user outputs.

III.PROPOSED SYSTEM

The existing approaches mostly focus on detection XSS attack either at client side or at the server side. So there is a need to come up with a solution that has the ability to detect Persistent as well as Non-Persistent XSS attack which will work both at the client and server side.We have proposed a detection model that will validate the user provided inputs at the client side and the response pages from the server is also validated and then given back to the client.The proposed system is a detection model for XSS attack consisting both Persistent and Non-Persistent cross-site scripting attack. The proposed model has different architecture for client and server side. The Figure 1 shows proposed detection system .

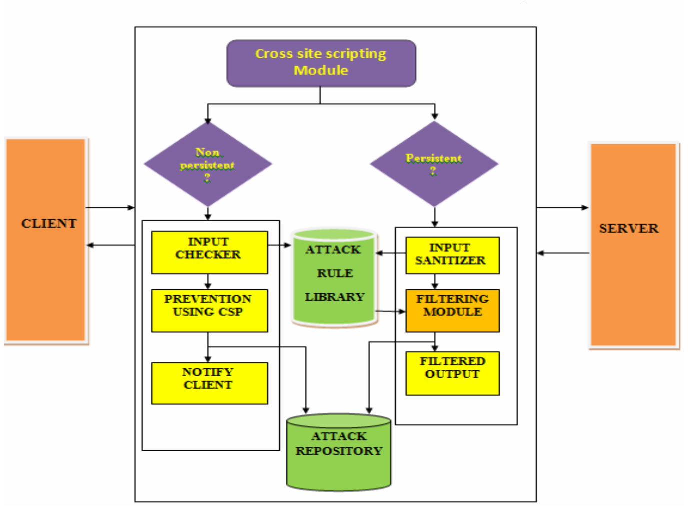

Fig.1 Detection Model for XSS Attack

The Figure 1 shows that detection model has separate modules under Persistent and Non-Persistent XSS attack. Whenever the client sends the HTTP request then the detection system will validate the user input before forwarding it to the server. At the server again the response will be checked and then final HHTP response will be given back to the client.

A. Detection of Persistent XSS Attack

The Persistent XSS attack consists of five modules : Input Sanitizer, Filtering Module ,Filtered Output ,Attack Rule Library and Attack Repository.

1. Input Sanitizer
This block will check the incoming request and determine whether it contains any malicious scripts. The input sanitizer will check the presence of malicious code, if no then request is allowed else it is passed on to the Attack Rule Library.

2.Filtering Module
The filtering module receives input from the Input Sanitizer. The filtering module is capable to filter malicious scripts present in the Event Handlers, Data URI, Insecure Keywords , Character Escaping, Common words in XSS Payload and filtering XSS Buddies.

3.Filtered Output
After the filtering module completes its processing the resultant output will generated by this module.

4.Attack Repository
This block is responsible for storage of log records. After the input is sanitized ,it will be passed to the filtering module and finally filtered output will be generated. The Attack repository will maintain the logs of all the filtered outputs for future use.

The main work of this block is to store the rules.Some examples of rules stored in the library will be as follows :

- <script>...DO NOT PUT UNTRUSTED DATA ..</script> directly in a script.
- <!_...DO NOT PUT UNTRUSTED DATA ..._> inside an HTML comment.
- <div ...DO NOT PUT UNTRUSTED DATA ...=test /> in an attribute name.
- Perform escaping of the URL before inserting untrusted or malicious data into HTML url parameter values.

B.Detection of Non-Persistent XSS Attack

The detection of Non-Persistent XSS attack consists of five blocks : Input Checker, Prevention using CSP(Content Security Policy), Notify Client ,Attack Rule Library and Attack Repository.

1.Input Checker
The input checker accepts the incoming request and checks for any malicious scripts from the Attack Rule Library. Suppose the incoming URL has malicious contents written inside script tag then the Attack Rule Library has a rule where the contents inside a script tag needs to be validated.

2.Content Security Policy
CSP is used to restrict the browser viewing your page so that it can only utilize resources downloaded from trusted sources. It has a white list of trusted source and browser contents. The main aim of CSP is to check whether to permit the browser from loading the website or not from its list of white sources.

3.Notify Client
This module will send an alert message to the client indicating that the website is not trusted and so the browser blocks it from execution.The rejected URL's will be stored in the attack repository for future use.

4.Attack Repository
This block is responsible for storage of log records. After the input is checked for presence of malicious scripts or tags ,it will be passed to CSP module and finally the client is notified with alert message in case of malicious website.The Attack repository will maintain the logs of all the blocked URL's for future use.

The functioning of Attack Rule Library is similar to one mentioned in the detection of Persistent XSS attack.

C.Detection of XSS Attack at Server Side

The detection of Cross-site scripting attack at the server side has five modules : Feature Injection,Policy Storage ,Web Server, Output Response Deviator, Sanitization and Feature Removal. The Figure 2 illustrates the framework :

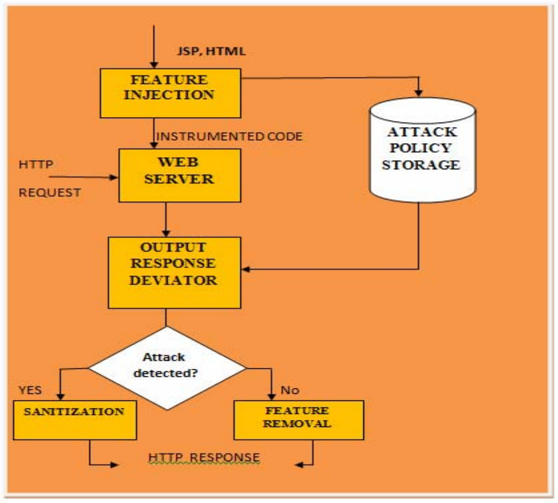

Fig.2 Detection of XSS Attack at Server side.

1.Feature Injection
This module is responsible for inserting an HTML or Javascript comment or features that does not modify the intended HTTP response or behaviours. The evaluation of features is to discover the presence of malicious injected contents.

2.Policy Storage
This module is responsible for storage of policies which represent the expected features such as number of tags, attributes, method names and arguments .The policies are given to the Output Response Deviator to check any variation between the actual and expected features.

3.Web Server
The web server represents the instrumented code with injected features that can be accessed from browsers. The request received by web server provides an initial response which is forwarded to the Output Response Deviator.

4.Output Response Deviator
This module is responsible for analyzing the initial response pages produced by web server .It checks whether any deviation is found between the actual and the expected features. XSS attack is detected if any deviation is found between the actual and expected features.

5.Sanitization
The main of sanitization is to remove the harmful scripts or contents.This module will remove the malicious contents and then give response to the client.

6.Feature Removal
In this step if no attack is detected by the Output Response Deviator then the boundaries or features could be removed and HTTP response is given to the client.

IV. CONCLUSION

Web applications are utilized for security-critical services so they have turned out to be a well-liked and precious target for web-related vulnerabilities.XSS attacks allows the attacker to execute malicious script on the victim’s browser thereby stealing user’s sensitive information. The existing approaches mostly focus on detection XSS attack either at client side or at a server side. So there is a need to come up with a solution that can detect Persistent and Non-Persistent XSS Attack which will work both at the client and the server side.Thus our proposed approach is modelled in such a way that it validates the input at the client side. This technique works for both Persistent and Non-Persistent XSS attack. The server side approach provides validated output.

REFERENCES

[1] “Noxes : A Client-Side Solution for Mitigating Cross-site Scripting Attacks ” by Engin Kirda,Christopher Kruegel,Giovanni Vigna and Nenad Jovanovic , Sac’06 April 23-27,2006,ACM Journal.

[2] “ XSS-Guard : Precise Dynamic Prevention of Cross-site Scripting Attacks” by Prithvi Bisht and V.N.Venatakrishnan , 2008 , Springer.

[3] “Swap: Mitigating XSS Attacks Using Reverse Proxy” by Peter Wurzinger,Christian Platzer,Christian Ludl , Engin Kirda and Christopher Kruegel, SESS’09,May 19,2009,978-1-4244-3725-2/09 , IEEE.

[4] “ S2xS2: A Server Side Approach to Automatically Detect XSS Attacks” by Hossain Shahriar and Mohammad Zulkernine ,2011, 978-0-7695-4612-4/11 , IEEE.

[5] “Preventing Persistent Cross-Site Scripting(XSS) Attack By Applying Pattern Filtering Approach” by Imran Yusof and Al-Sakib Khan Pathan ,2014, IEEE.

[6] “XSS-Safe : A Server-side Approach to Detect and Mitigate Cross-site scripting (XSS) Attacks in Javascript code” by Shashank Gupta, B.B Gupta ,2015, Springer.

[7] “Mitigating Cross-site Scripting Attack with a Content Security Policy. ” by Imran Yusof and Al-Sakib-Pathan ,2016,IEEE.

[8] “Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-site Scripting Attacks,” by Matthew Van Gundy and Hao Chen,Feb 2009, Proc. of NDSS, San Diego.

[9]“Cross-site Scripting(XSS)Attacks and Defense Mechanisms: classification and state-of-art” by Shashank Gupta and B.B Gupta ,14 September,2015, Springer.

[10] “SecuBat : A web Vulnerability Scanner” by Stefan Kals, Engin Kirda,Christoper Kruegel and Nenad Jovanovic , 2006 ,ACM 1-59593-323-9/06/0005