www.ijsrnsc.org

Available online at www.ijsrnsc.org

IJSRNSC

Volume-5, Issue-4, August 2017
Research Paper

Int. J. Sc. Res. in Network Security and Communication

ISSN: 2321-3256

# Automated Detection of Legitimate Java Script Code from a Malicious Injected Code and Improvising the Time Efficiency

**R. Saliha Bathool[1*], K.Vijayalakshmi[2]**

[1*]Dept. of MCA, Ethiraj College for Women, Chennai, India
[2]Dept. of MCA, Ethiraj College for Women, Chennai, India

[*]*Corresponding Author: salihabathool_salihabathool15@gmail.com, Phone: +919003189147*

Received 14th Jul 2017, Revised 26th Jul 2017, Accepted 18th Aug 2017, Online 30th Aug 2017

***Abstract***— Automated detection of legitimate java script code from a malicious injected code helps in distinguish between malicious and non-malicious script In order to prevent the users from XSS attack, many client- side solutions have been materialize; most of them being used are the filters that refines the malicious input. Moreover, many of these filters do not provide prevention to the newly designed sophisticated attacks such as multiple points of injection, injection into script etc. This paper proposes and implements an approach based on conceal XSS detection mechanism which bestow JSP detector that scours the vulnerability in the web page that perform automated process to differentiate between java script codes from a malicious injected code and also improvise the efficiency of fast policy check to optimize the speed of XSS attack detection process. It ensures that the file modifications should not entertained by any unauthorized user, if anybody tries to modify it automatically notify the authorized admin but it will contain any changes. Merely, this mechanism is very efficient for detection and security concern.

***Keywords***— Fast Policy Check, Speed Efficiency, JSP Detection, File Detection Speed

## I. INTRODUCTION

Automated detection mechanism that permits an attacker to execute injected JavaScript in victim's web browser in order to gain access to the sensitive resources like cookies, password, credit card numbers etc. XSS is an attack on the client-side web browser, but its capabilities are exploited on the web server side. For the exploitation of XSS vulnerabilities on the web applications, an attacker crafts and injects a malicious JavaScript payload on the web application [5]. This script is injected in such a way that it seems to be began component of the website and at last this script is executed within the domain of the trust of the website. The web server needs a JSP engine, i.e., a container to process JSP pages. The JSP Container is responsible for intercepting requests for JSP pages. This tutorial makes use of Apache which has built-in JSP container to support JSP pages development. A JSP Container works with the Web server to provide the runtime Environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.

### A. JSP Processing

The following steps explain how the web server creates the Webpage using JSP with a normal page, the browser sends an HTTP request to the web server .The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with .jsp instead of .html. The JSP engine loads the JSP page from disk and converts it into servlet content. This conversion is very simple in which all template text is converted to println ( ) statements and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page. The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine. A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format.

 

The output is further passed on to the web server by the servlet engine inside an HTTP response. The web server forwards the HTTP response to your browser in terms of static HTML content. Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

## II. LITERATURE SURVEY

1. Kanpata Sudhakaraina Naman J, Rao, et. al.[ 2015] Introduces XBuster - they client-side defense against XSS, implemented as an extension to the Mozilla Firefox browser[3]. XBuster splits each request parameter into HTML and JavaScript contexts and stores them separately. It defends against all XSS attack vectors including partial script injection, attribute injection and HTML injection.

2. Chokhawala Kirit I., Dr Vinit Kumar et. al.[ 2016]developed a study and analyze the application level attacks for secure web application. An application level attack contains Cross Site Scripting attack, SQL injection attack, Command Injection Attack and Cookie Poisoning attack [9]. A web application is a software application that is accessed over the Internet using Hyper Text Transfer Protocol (HTTP). In a typical web application a client interacts with a web server by exchanging a series of messages that are made up of HTTP requests and responses.

3. V. Nithya1, S. Lakshmana Pandian2 et. al.[ 2015] developed the analysis of detection and prevention of Cross-Site Scripting (XSS) help to avoid this type of attack. They evaluate a technique to detect and prevent this kind of attack [5]. Cross-Site Scripting (XSS) vulnerabilities are being exploited by the attackers to steal web browser's resources by injecting the malicious JavaScript code on the victim's web applications. This feature is useful to enforce the execution of malicious code in a user's Web browser.

4. Monika Rohilla, et. al.[ 2016.] In this paper XSS attacks have been discussed with their classifications. Selection of victim web application which is vulnerable for XSS attack and some vulnerability scanners are also discussed [7] the techniques are discussed in detail with real life case studies and guidelines to prevent them are also evaluated in this paper.

5. D. K. Patil K. et. al.[ 2016] In this paper, they implemented a novel client-side XSS sanitizer that prevents web applications from XSS attacks[6]. The sanitizer is able to detect cross-site scripting vulnerabilities at the client-side. It secures the web browser, because modern web browsers do not provide any specific notification alert or indication of security holes or vulnerabilities and their presence in the web application.

6. Manisha S. et. al.[ 2014] This survey paper focuses on various security tools and prevention techniques are available to mitigate attacks due to Cross-site Scripting (XSS) and Cross-site request forgery (CSRF) vulnerabilities[8]. Cross-site request forgery (CSRF), SQL Injection those are used to exploit by the hackers for malicious purposes. merely there is a need for API's/automated security tools to identify and to prevent these vulnerabilities before the application goes live00000.

7. Neha Gupta,[ 2015] This paper presents a new XSS defense approach which is based on the OWASP guidelines available for prevention of XSS attacks. In this approach for XSS defense there is an XSS checker [1] that will check for the unauthorized characters in each parameter in the input and block them on both client side and server side of a web application. Client side solutions reduces the run time overhead and server side solutions are more reliable as any attack occurring when request is going from client to server will be detected by server side solution only but it incurs runtime overhead. So a combination of both will be more robust as it can prevent most of the attacks and manage runtime overhead effectively. This approach is tested on a prototype. It is found that this approach covers major categories of XSS attacks i.e. reflected and stored and will require no additional frameworks.

8. Shashank Gupta, B.B.Gupta,[2015] this paper presents an enhanced XSS defensive methodology for the cloud platforms[4]. They have implemented the framework which initially scans the HTTP requests for the embedded URI links that points towards the links of external JS files and which may contain malicious XSS payload. they design also explores the HTTP response for extracting the script content and compares this content with the script content retrieved from the URI links. Any resemblance observed will be extracted and set of scripts would be considered as malicious XSS worm.

## III. SYSTEM ARCHITECTURE

 

XSS Detection mechanism helps in Comparing between legitimate java script codes from a malicious injected code. It determines the detection process in comparing the malicious java script code by the non malicious java script code. The web server needs a JSP engine, i.e., a container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. This tutorial makes use of Apache which has built-in JSP container to support JSP pages development.

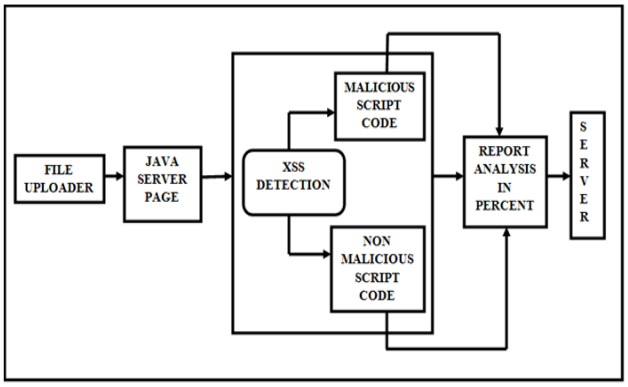


Fig: 1 Efficient detection process of malicious java script

### *A.* File up loader

File up loader is a concept of unloading file in the web Uploading File to server using Servlet and JSP is a common task in Java web application. Before coding your Servlet or JSP to handle file upload request, you need to know little bit about File upload support in HTML and HTTP protocol [16]. If you want your user to choose files from file system and upload to server than you need to use <input type=”file”/>. This will enable to choose any file form file system and upload to server. Next thing is that form method should be HTTP POST with encrypt as multipart/form-data, which makes file data available in parts inside request body. Now in order to read those file parts and create a File inside Servlet can be done by using Servlet Output Stream.

### B. Java server page (JSP)

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language, jstl etc. JSP page consists of HTML tags and JSP tags [4]. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc.

### C. XSS Detection

XSS detection mechanism helps in distinguish between legitimate java script codes from a malicious injected code [10]. It determines the detection process in comparing the malicious java script code by the non-malicious java script code. XSS detector process by evaluating the java script code if the client or server uploads the file in any other script except than java the file will be called as non-malicious.

## IV. METHODOLOGY

### A. Methods and Procedure

JSP dynamically generates web page based on html, xml or other document types. JSP allows java code and certain predefined actions to be interacted with the static markup content.

- Start the server web page.
- Put the JSP file in a folder and deploy on the server. Upload the file by clicking file up loader.
- XSS detector detects the vulnerability available in the web page and regenerates.
- If the uploaded file contains java script language then the file is considered as non-malicious java script file
- Else, if the uploaded file contains other language is marked as malicious java script file.
- It improves the fast check policy by installing the timer which detects the Exacts time taken in detection and uploading file.
- It contains a check box in the design view that notifies the admin if somebody tries to modify or change the content of the web page.

## V. RESULT

File uploading mechanism helps in uploading various files. XSS detector perform the process by evaluating the java script code if the client or server upload the file in any other script except than java the file will be called as non-malicious. It helps in distinguish between legitimate java script codes from a malicious injected code.

 

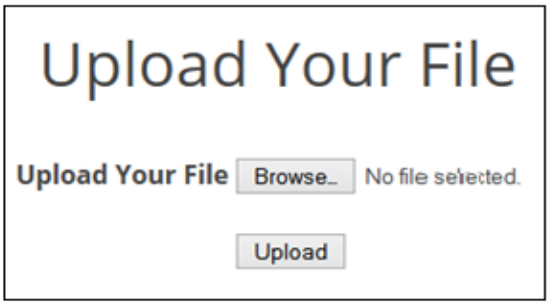


Fig: 2 File uploading mechanism

| File Name | Timing | File Size | View File |
|---|---|---|---|
| t8.txt | 1.67 | 0.0032720565795898438 | Click Here |
| t39.txt | 0.04 | 0.005087852478027344 | Click Here |
| t8.txt | 0.04 | 0.0032720565795898438 | Click Here |
| p27_events.html | 0.15 | 2.5988983154296875E-4 | Click Here |
| new.html | 0.03 | 7.591247558593754-4 | Click Here |
| t8.txt | 2.11 | 0.0032720565795898438 | Click Here |

Fig: 3 Updated Table of files

File updated table of malicious and non malicious which shows File Name, Timing, File size. Both malicious and non malicious file are saved in same table.

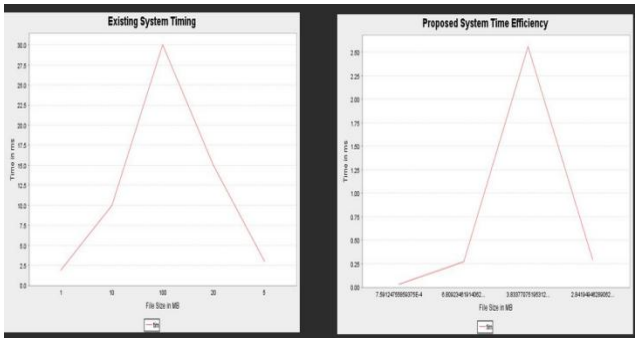


Fig:4 File Detection speed of maicious script

File detection speed of malicious and non malicious script uses a dynamic graph it contains a code reusability which helps in reducing calling time of every file while uploading files of malicious and non malicious script

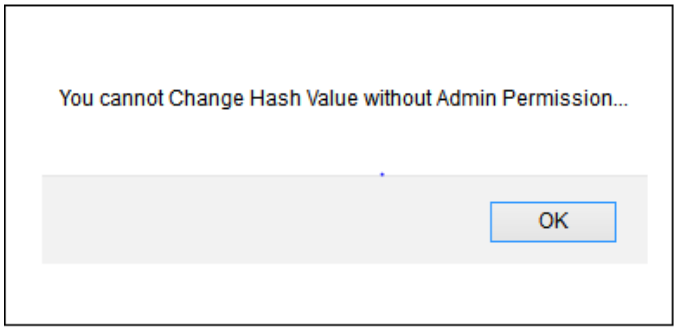


Fig: 5 File modifying message box

File modification will not happen if any unknown user will tries to modify the content of files. If the hash value has been try to change in any file, automatically it will notify the admin about the changes.

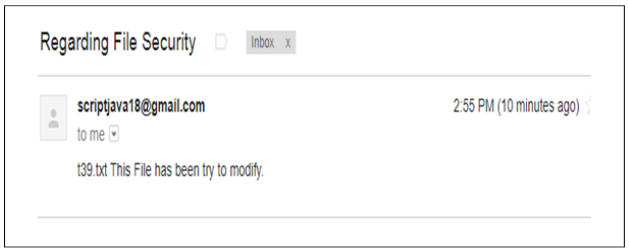


Fig: 6 File Security Notifications

While the file has been tries to modify it will notify the admin by sending a security message for the security concern.

## VI. CONCLUSION

The main objective of this system is to replenish a secure web page by the sustenance of automated detection of legitimate java script code from a malicious injected code. It contains java server page which dynamically loads the uploaded data with the file extension .java. XSS detector performs the role of scrutinized and detects the XSS vulnerability and split it in to two parts malicious code and non-malicious code. It enables the process of efficient speed policy check and improvises the detection speed.

## REFERENCES

[1]. Neha Gupta, "*XSS Defense: An Approach for Detecting and Preventing Cross Site Scripting Attacks*", International journal of advanced computer technology, Vol.4, Issue.3, 2015.

[2]. Shashank Gupta, BB Gupta. "*Cross-Site Scripting attacks and defense mechanisms*", classification and state-of-the-art,Int J Syst Assur Eng Manag- DOI 10.1007/s13198-015-0376-0.

 

[3]. Kanpata Sudhakara Rao, Naman Jain, Nikhil Limaje, Abhilash Gupta, Mridul Jain, Bernard Menezes, "*Two for the price of one: A combined browser defense against XSS and click jacking*", IEEE Publication Department of Computer Science and Engineering, IIT Bombay, Mumbai, India, 2016.

[4]. Shashank Gupta, B.B.Gupta, "*Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment*", International Conference on Emerging Trends in Engineering Science and Technology 2015.

[5]. V.Nithya1,S.Lakshmana,Pandian,e "*A Survey on Detection and Prevention in Cross Site Scripting Attack*", International Journal of Security and Its Applications, Vol. 9, No.3 (2015).

[6]. D. K. Patil K. R. Patil, *"Client side Automated Sanitizer for XSS Vulnerabilities",* International Journal of Computer Applications, Vol.121 - No.20, July 2016.

[7]. Monika Rohilla, Rakesh Kumar, Girdhar Gopal, "*XSS attack analysis, Detection and prevention*", International journal of advanced research in computer science, Vol.6, Issue.6, June 2016.

[8]. Manisha S. Mahindrakar, "*Prevention to Cross site Scripting Attacks: A Survey*", IJSR, Volume 3 Issue 7, July 2014.

[9]. ChokhawalaKirit I., Dr. Vinit Kumar Chuabay, Dr. A. R. Patel, "*Secure Web Application: Preventing Application Injections"*, IJSRSET,Vol.2, Issue.1, 2016.

**Author Profile**

*Ms.* Saliha Bathool pursed Bachelor of information systems management from Justice Basher Ahmed College for Women in 2011 and Master of computer application from Ethiraj College for women in the year 2016. She is currently pursuing her M Phil in Ethiraj College for women and also did her Master of Business Administration from Annamalai University. She has presented a paper on XSS Refiner Technique for Web Vulnerability in DRBCH College. She did her Research work in cryptography and network security.

 