INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING & APPLIED SCIENCES
ISSN: 2322-0821(O)
ISSN: 2394-9910(P)
VOLUME 9 ISSUE 2
Apr 2021 - Jun 2021
www.irjeas.org

HANDLING VULNERABLE SCRIPT CODE IN WEB ENGINEERING

Meena Deshbhratar[1*], Anurag Srivastava[2]

*[*1]M.Tech Scholar, [2]Asct. Professor*
Department Of Computer Science Engineering, NRI Institute of Research & Technology, Bhopal, India
**Corr. Author - mnadeshtr09@gmail.com*

Abstract- Network protection in our everyday lives is becoming increasingly critical today. Since we cannot live without the Internet, it is important to have a good and security environment for networking. Cross site scripting (XSS) does, however, attack millions of websites. We can use XSS to insert malicious scripting code into apps and then return it to the customer side. If users are using the web browser to visit the injecting place of the malicious script code, it is directly run on the customer machine. The main words of XSS are commonly found in the JavaScript browser or on the server component to filter malicious code. However, it is difficult to collect all keywords in the detecting-list in order to prevent XSS attack. However, it is possible to create various forms of malicious scripting. It is also worthwhile for more people to work on XSS and find more ways to prevent XSS attacks.

Keywords – Script Code, Vulnerability Trends, Web Engineering

I. INTRODUCTION

During the last decade, the Internet has seen a huge growth of the exchange of data by many means, regardless of their distance or location, by volume, nature and channel. The internet has become in particular the main channel by which global businesses operate and are extremely successful in traditional marketing strategies. Nearly every organization today continues to expand beyond its borders; therefore, almost every human endeavour and creation takes on a critical role in the network worldwide. Web apps are one of the best ways to accomplish this vital online presence. Web applications are web technology computer programs for performing tasks on the Internet. Thus, the advent of web applications and other smart devices such as smartphones, tablets and other mobile devices has changed the medium of communication and the exchange of information among platforms. With the widespread and all-round existence of these Internet applications, app developers are forced to reconsider their development strategies and shaping their security issues to avoid targeting hackers and web assailants who are finding inadequate coding practices on the Internet daily to steal sensitive information and perpetuate the Often, with the number of web applications increasing, there are vulnerabilities and a major point of discussion in various development and security forums for web applications.

Web applications usually allow sensitive customer data (such as personal details, credit card numbers and information from social security groups) to be collected, analysed, stored and distributed immediately and repeatedly [1]. As a consequence, web applications have become main objectives of hackers who profit from miscoding, flaws in the application code, insufficient user input permission, or failure of software developers to conform to security standards. These vulnerabilities may either be found on the server side or on the client side more dangerously. SQL injection, cross-site request forgery, information leakage, hijacking of the session and cross-site scripting are the vulnerabilities. The emphasis of this paper is on the detection of cross-site attacks. Cross-site scripting is called malicious code injection in insecure web applications to trick users and redirect them to un-trusted websites (XSS). XSS can also occur when no flaw is in the servers and database engine and is probably one of the most common web applications currently exposed which is shown in Figure 1.

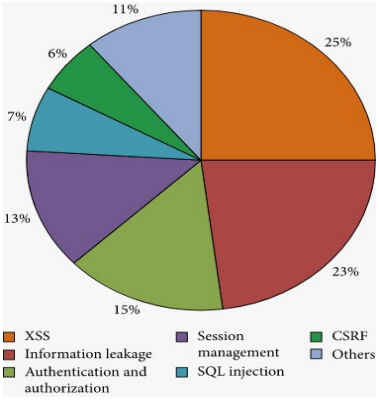

Figure 1.1Centric Application Vulnerability Trends Report

HTML types, cookies, secret fields and get and post parameters are all input sources exploited by attackers.

International Research Journal of Engineering & Applied Sciences, IRJEAS
www.irjeas.org, ISSN(O): 2322-0821, Volume 9 Issue 2, Apr-Jun2021, Page 01-04

Three parties—the intruder and the customer—are part of the ordinary procedure. XSS violations can lead to fraud, theft of identity, regulatory penalties, loss of goodwill, litigation, and client loss.

Many research studies have focused on XSS vulnerability problem solving. In information security testing [2–4], the majority of methods cantered on preventing XSS attacks in web applications. There have been few research activities on its detection [5–6].

II. LITERATURE SURVEY

Current research aims to increase the efficiency of detection by incorporating intelligence. Current browser / device defects can be familiar (using, e.g., vulnerability bases). You can also understand how the results depend on the application's inputs. Any of these approaches are here.Cantered on model inference and evolutionary fuzzing, a novel method for detecting XSS has been developed [7]. This method simply employed a subtracting algorithm motivated by heuristics to create a crawler. They suggested an approach to infer web applications models to shape a grammar of attack. The grammar of the attack produces pieces that reduce the scope of the quest. In order to program Malicious Inputs that are sent to your application, genetic algorithms are then used. As the concept was big, it meant that the application should reset to its initial node, which may not always be useful. The frame also presumed that XSS will only be the product of one fluctuating value.

Boyer–Moore matching string algorithm was used for the detection technique in a solution proposed by Saleh et al. [8]. The characteristics of the input pattern are contrasted with the webpage characters from right to left with the heuristics referred to as the bad-type shift and the good-suffix transition. The module's core concept is to fulfil the necessary requirements, which can search character by character for the input pattern from right to left. But the scanner takes a long time to complete its scan when the duration of the URL is long.

A working algorithm called 'NUIVT' was suggested by Abbass and Nasser[9] (novel user input validation test). There are three stages in the algorithm: The first phase analyses the fields of user input and input type detection. In the second stage input forms are transformed into regular phrases. The third phase tests are done to detect vulnerability for the invalid input, but results are saved from each scan to enhance the system intelligence which leads to repeat comparisons and is time consuming and tedious

Koli et al. [10] suggested SQL and XSS architecture. They have developed a SQL injection and the XSS detection method which uses HTTP request filters to look for attack signatures. To decide whether or not the script tag is present, a detection component is used. As user response, the result is stored in a database. In order to evaluate its effectiveness, they compared their work with well-known vulnerability scanners. The major downside of your research was that the tool cannot effectively detect the attack if its pattern is not stored in its database.

III. PROPOSED WORK

We propose a safe browser/ browser plug-in with its inbuilt support for JavaScript interpretation. We implement the interpreter to detect the client-side JavaScript based attacks so that the browser will not execute the unsafe statements thereby preventing the user security from getting compromised. The proposed safe browser will be able to detect various JavaScript based attacks and will not let them execute on client side. The interpreter will contain signatures of most of the well-known attacks, and it will employ regular expressions for identifying the presence of any malicious JavaScript code in the current webpage.

As an example, in cross-site scripting (XSS) attacks, a malicious web application gathers confidential data from a user. A typical example of a XSS attack is when a user is tricked into clicking on a link hosted by a malicious host (e.g., www.evil.com).

The link, appears to be pointing to a resource on a trusted site (e.g., http://www.bank.com/accounts.html, but, instead, it contains JavaScript code as the resource name. When the resource is requested by the user's browser, the code is sent as part of the HTTP request to the destination of the link (i.e., the trusted web server). Since the requested resource does not exist, the trusted web server returns an error message that contains the name of the resource that could not be accessed. As a result, the page containing the error message is interpreted by the client's browser as a page from the trusted site containing some JavaScript code. Therefore, the code is executed in the context of the trusted site and has access to the cookies previously set by the trusted site, including session identifiers and authentication tokens.

A. The Proposed Scheme

A new method for the detection and prevention of the malicious code in the scripting site is provided. The method is the combined architecture with no rules generation to detect various scripts in the database. The proposed method works as follows: Cross site scripting attacks usually taken place at the user validation or in the user input query execution.

In the proposed method the input query is clearly investigated, whether it consists of any malicious code. The following steps explains the working procedure of the method,

Classifier	Evaluation Criteria			
	Training time (sec)	Testing time (sec)	Training Accuracy (%)	Testing Accuracy (%)
Proposed Approach	0.1250	0.0313	0.7784	0.9962
ELM Kernel	0.1796	0.0065	0.7218	0.9740

Table 3.1- Comparison of proposed approach an elm-kernel

Step 1: Inputthe user Query in the available validation control.

Step 2: Identify for any script within the input entered by the user.

Step 3: Split the input values into two characters for each tag. (Eg.)

Step 4: Check the occurrence of the scripts such as Step 5: Check the given input with the predefined tags specified namely, <# …>,<=…>,

IV. RESULT AND IMPLEMENTATION

Project classification algorithms, such as the Extreme Learning Machine (ELM) and the Compare to Kernelized Extreme Learning machine were implemented (KELM). These managed algorithms for learning are carried out via MATLAB 2012. The dataset can be divided into a training set. For training purposes 80% of data was split into training data from the final dataset such that 240 records out of the 300 records are included in the training data. In addition to the checked data, there are also 20% of the final data, and the test collection includes 60 records of the 300 records. In these cases, two algorithms like the Approach Proposed and the Kernelized Extreme Learning Machine (KELM) are compared according to time and precision of learning. Even if the degree of precision is closer, the time for each analysis varies. MATLAB is used to apply the Extreme Learning Machine (ELM). Here the malicious webpage has been classified by the ELM.

For the classification, the results are compared based on four factors such as training time, testing time, training accuracy and testing accuracy. There are two algorithms, for example Basic-ELM and ELM-Kernel. The analysis between 2 algorithms is shown in Table 3.1

The predictive precision shown by the Basic Proposed approach to ELM-Kernel is higher in terms of prediction accuracy, compared to the ELM kernel in the above-mentioned comparative analysis (Table3.1). It takes more than the proposed approach algorithm for the model to be created via ELM-Kernel. Fig4.1 Displays the diagram of the results of the study between two algorithms

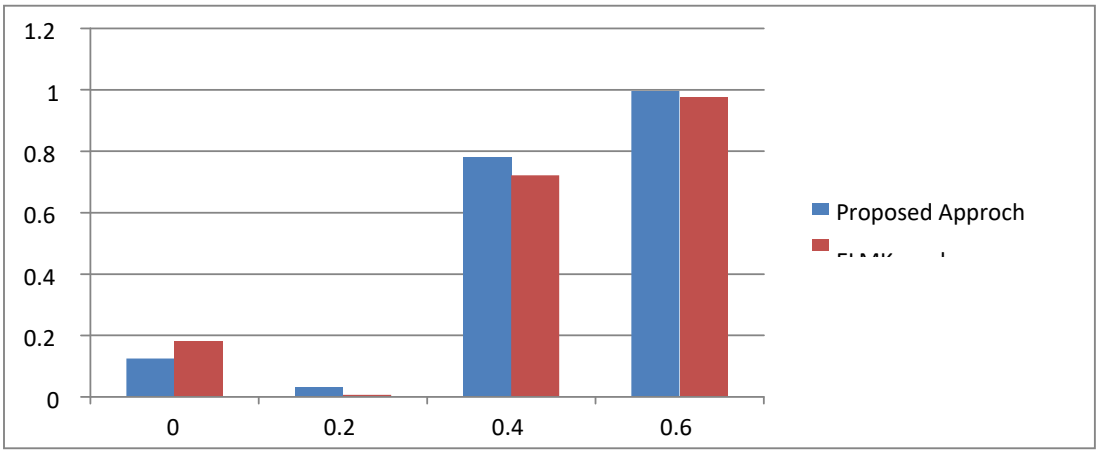

Figure 4.1: Comparison chart for Proposed Approach and KELM

V. CONCLUSION

The JavaScript language is used to boost the presentation of web pages on the client side. JavaScript is downloaded and run by an embedded interpreter in the browser on-the-fly. Browsers have sandboxing mechanisms to prevent JavaScript code from compromising the security of a client environment but unfortunately there are a variety of attacks that can be used to steal user credentials, e.g. cross-site scripting attacks (e.g., phishing attacks). We suggest an approach to resolving this problem based upon browser control of the execution of JavaScript code. The trained model was created with the aid of an extreme learning machine and kernels. The performance of the trained models is assessed by ten times cross validation based on prediction precision and time, and the results are analyzed. It has been observed that ELM-based prediction model shows around 99

percent predictive accuracy. For the prediction of malicious web pages, the training time and accuracy of the test play a major role in deciding the model's results. For the classifications of cross site scripting (XSS) web sites, this research work has successfully implemented Extreme Learning Machine (ELM). For the potential improvement of this study

REFERENCE

[1]H.Hibshi, Composite Security Requirements in the Presence of Uncertainty. Societal Computing Institute for Software Research, Carnegie Mellon University, Pittsburgh, PA, USA, 2016.

[2]P. Sharma, R. Johari, and S. S. Sarma, "Integrated approach to prevent SQL injection attack and reflected cross site scripting attack," International Journal of System Assurance Engineering and Management, vol. 3, no. 4, pp. 343–351, 2012.View at: Publisher Site | Google Scholar

[3]Y. Sun and D. He, "Model checking for the defence against cross-site scripting attacks," in Proceedings of the 2012 International Conference on Computer Science and Service System, pp. 2161–2164, Maui, Hawaii, January 2012.View at: Google Scholar

[4]M. Van Gundy and H. Chen, "Noncespaces: using randomization to defeat cross-site scripting attacks," International Journal of Computer Security, vol. 31, no. 4, pp. 612–628, 2012.View at: Publisher Site | Google Scholar

[5] H. Isatou, S. Abubakr, Z. Hazura, and A. Novia, "An approach for cross site scripting detection and removal based on genetic algorithms," in Proceedings of the Ninth International Conference on Software Engineering Advances: France, pp. 227–232, Nice, France, October 2014.View at: Google Scholar

[6]P. Bathia, B. R. Beerelli, and M. Laverdière, "Assisting programmers resolving vulnerabilities in Java web applications in CCIST," Communications in Computer and Information Science, vol. 133, no. 1, pp. 268–279, 2011.View at: Publisher Site | Google Scholar

[7]F. Duchene, R. Groz, and S. A. Rwat, "Vulnerability detection using model inference assisted evolutionary fuzzing," in Proceedings of the IEEE Fifth International Conference on Software Testing, pp. 815–817, Montreal, QC, Canada, 2012.

[8] A. N. Abbass and M. Nasser, "Presentation of a pattern to counteract the attacks of XSS Malware," International Journal of Computer Applications, vol. 143, no. 2, pp. 78–88, 2016.View at: Google Scholar

[9] M. Koli, S. Pooja, H. K. Pranali, and N. G. Prathmesh, "SQL injection and XSS vulnerabilities countermeasures in web applications," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4, no. 4, pp. 692–695, 2016.

[10] M. Koli, S. Pooja, H. K. Pranali, and N. G. Prathmesh, "SQL injection and XSS vulnerabilities countermeasures in web applications," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 4, no. 4, pp. 692–695, 2016