

Web Page XSS Attack Detection from Malicious JavaScript using Machine Learning: A Survey

Sanjukta Mohanty¹ and Arup Abhinna Acharya²

^{1,2}School of Computer Engineering, KIIT University, Bhubaneswar, India-751024
Email: mailtorani.sanjukta@gmail.com, aacharyaacs@kiit.ac.in

Abstract—The usage of web application is increased in a rapid rate which leads to different kinds of cyber attacks. Cross Site Scripting (XSS) is one of the application layer code injection attack of web application which allows the invader to execute the malicious code (typically JavaScript) in a victim's browser to steal the cookies, sensitive data, spread malwares, deface the web page, hijack the session etc. Malicious JavaScript and malicious links are the most usual ways of performing the XSS attack. So identifying the script as malicious or benign is an important issues in security testing of web application. In this research article we survey the different effective detection technique for XSS using machine learning algorithms. This study employs static detection with machine learning approaches to analyze and classify the JavaScript malware features to be used in different machine learning classifiers and ensemble machine learning techniques for the prediction of XSS attack.

Index Terms— XSS, web page, machine learning, malicious, java scripts.

I. INTRODUCTION

The modern web applications are developed by several advanced technologies like java scripts, Ajax, PHP, ASP.net etc. [1] to provide ease of access to different web services in a cheapest and fastest way to the internet user in their day to day activities like online shopping, e-commerce, blog, forum, social networking sites, internet banking etc. So to provide these facilities to users web application must be interactive, dynamic, attractive and user-friendly and this is possible only because of different client and server side scripts [4]. The client side JavaScript (JS) is a dynamic and lightweight scripting language added with different functionality to make the web sites more interactive. As this technology allows the client-side scripts to interact with user so it reduces the client-server interaction and other advantages are immediate feedback and rich interfaces [9]. Despite it's various advantages, JS has become the most conventional ways of performing the different cyber attacks like XSS [14] because the intruder makes use of these scripts to craft different attack vector to do harm to the users directly or indirectly. XSS has become one of the major attack vector [4]. It is caused due to the lack of verification or improper validation of client input resulting in web page deformation, phishing, stealing of session id embedded within the cookie, spreading of malware, keylogging, capturing the clipboard content etc. The rest of this paper is organized as follows: Section II gives an overview of relevant aspects of JavaScript and XSS attack scenario. Section III discusses related work on machine learning and XSS. Section IV details the discussion. Section V gives the direction of future work and conclusion.

II. BACKGROUND DETAILS OF XSS AND JAVA SCRIPTS

Cross-site scripting (XSS) is a type of web security vulnerability typically found in Web applications that accepts user inputs. XSS enables attackers to insert client-side script into web pages viewed by other users. Cross-site scripting uses known loopholes present in web-based applications, servers or plug-in systems on which they rely. Exploitation of one of above loopholes allows insertion of malicious content into the contents being delivered from the compromised site. When the resulting compromised content arrives at the user web browser, it is considered as it is delivered from the trusted source, and thus operates under the permissions granted to that access data associated with source website. Attacker can gain privileged access to highly sensitive data, such as session cookies, and other information stored by browser on user behalf by finding different ways to inject malicious script into web pages. Cross-site scripting attacks are special case of code injection. XSS commonly targets scripts embedded in a page which are executed on the client-side (in the users web browser) rather than on the server side. XSS is a threat arises due to internet security weaknesses of various client-side technologies such as HTML, JavaScript, VBScript, ActiveX and Flash. Presence of weakness in these technologies is a main reason for the exploit. XSS is triggered by manipulation of client-side scripts present in web application in a way as anticipated by attacker. Such manipulation embeds script in page which is executed wherever that page is loaded or associated event is performed. In a classic XSS attack the attacker infects authentic web page with his malicious client-side script. When a user visits this web page the script is downloaded to his browser and executed [1].

A. Non-persistent/Reflected XSS Attack

Non-persistent attacks are types of attack in which user provided data present in request to server is reflected partially or completely in the form of error message, search result or any other type of response. Non-persistent attacks are delivered to victim in various ways such as in an e-mail message or link present in other web site. When user is deceived to click on malicious link which result in submission of specially crafted form or just browsing of malicious site which will reflect attack back to user browser and browser will execute that response treating as if it came from trusted site[15]. The attacker will include the scripts like “http://webapplication.com?search=”>” When victim will click on that link then request is sent to webapplication.com with given data search parameter that are pointing to the script stored at attacker server that will steal cookie. When webapplication.com sends response then that script will gets executed at victims browser resulting in cookie stealing attack.

B. Persistent/Stored XSS attack

Stored XSS attacks are more dangerous than other types of XSS. Because malicious scripts injected by attacker are stored permanently by server in databases or web pages and served back to other users as normal pages coming from trusted application and user is interpreting them without proper HTML sanitization. Various sinks like database, message forum, comment fields and visitor logs are used to store such malicious scripts permanently, and sent to user browser when request is made for particular content.

C. DOM based XSS attack

DOM XSS is type cross site scripting attack which arise due to improper handling of data related with DOM (Document Object Model) present in HTML page. DOM is standard for accessing and manipulating objects in HTML document. Every HTML entity present in page can be accessed and modified by using DOM properties such as document.referrer, document.url and document.location. Attacker can manipulate or access DOM properties to execute such type of attack. In the DOM-based XSS, the malicious script does not reach to the web server. It is executed at client side only. DOM based XSS attack occurs when user provided untrusted data is interpreted as JavaScript using methods such as eval(), document.write or innerHTML. The main culprit for these type of attacks is JavaScript code.

III. RELATED WORK

For detecting XSS attack pattern from java script malwares different approaches have been proposed. Most of the existing studies detected XSS attack from java script malwares depended on static analysis with machine learning techniques or dynamic analysis with behavioral abnormality checking.

In 2009, Likarish *et al.* [7] proposed an approach for classifying malicious java script codes based on obfuscation scripts features where the frequency of occurrence of symbols were tracked in benign and malicious java scripts.

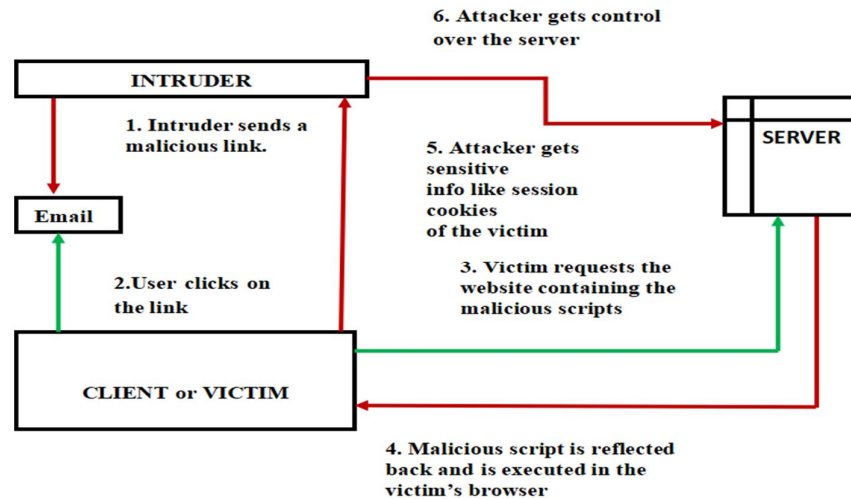


Figure 1. An overview of XSS attack

The classifiers Naive Bayes, Alternating Decision Tree (ADTree), Support Vector Machines (SVM) and the RIPPER rule learner were trained and evaluated using 10-fold cross validation giving precision of 0.92% in SVM classifier. In 2010, Cova *et al.* [10] presented a novel approach for detecting and analyzing the malicious java scripts using machine learning(Naïve Bayes) and anomaly detection method, mainly detected the Drive-by-Download attack of JS based attacks. They implemented their approach in a tool, JSAND (JavaScript Anomaly-based Analysis and Detection) which was able to detect more attacks and projected a low number of false positives. Their approach identified 10 features characterizing intrinsic events of a drive-by download attack. Then machine learning technique Naïve Bayes(NB) is used to detect malicious samples by utilizing these 10 features which are used for the purpose of redirection, de-obfuscation, environmental context and exploitation. Their analysis technique automatically created the deobfuscated version of malicious JavaScript code. In 2011, Komiya *et al.* [3] detected malicious web code by classifying the user input with the help of machine learning techniques (SVM, Naïve-Bayes, k-nearest Neighbor). Two types of methods have been proposed for feature extraction one is blank separation, and other one is tokenizing. In the first method the input may have many terms separated by spaces, so a feature weight is calculated by counting each term. In the second method malicious code contains tokens that explains the features of malicious web code, each term is counted to calculate feature weights. Using this feature extraction technique with SVM gave accuracy of up to 98.95%. The disadvantage of the first method is that malicious script terms might be separated by characters other than spaces, which would generate an incorrect feature weight. In 2012, Nunan *et al.* [6] adopted machine learning classifiers (SVM and Naïve Bayes) to automatically classify XSS attack from the malicious web pages extracting the features from web document content and URL. The features were classified into three types: obfuscation based, suspicious patterns and HTML/JavaScript schemes. The experiments were conducted using three datasets namely Dmoz and ClueWeb09(benign scripts) and XSSed.com (malicious scripts). The machine learning classifiers were evaluated and found accuracy 98.58% with Dmoz dataset, and 99.89% with the ClueWeb09 dataset. Though this technique gives high accuracy, but it is depended on a single source for malicious scripts and concentrated on obfuscated JavaScript. In 2013, Wei-Hong *et al.* [4] used a combined approach of static detection with machine learning technique(SVM) to analyze and conduct the malicious feature extraction to classify script as malicious or benign scripts. The features are extracted basing upon the functions like eval function, the escape function and DOM-modifying functions which are used more in malicious scripts than benign scripts. The abnormal use of special keyword, strings and tags are also took into account. The SVM machine learning detection technique shows the accuracy of up to 94.38% on the test set. In 2014, Vishnu *et al.* [5] proposed an approach for predicting XSS attack vector based on the features extracted from URL and java script code. Three machine learning algorithms such as Naïve Bayes, Support Vector Machine and J48 Decision Tree are used to classify the malicious scripts from benign scripts. J48 Decision Tree gave better performance with respect to FPR(False Positive rate) for identifying XSS attack based on URL and java script. In 2015, Wang *et al.* [8] proposed a hybrid approach of static machine learning and dynamic program analysis. Malicious java scripts were classified using machine learning techniques(AD Tree, Random Forest, J48,Naïve Bayes) using the

predictive features of textual information, program structure and risky function calls and from the malware behavior able to classify 8 different types of attack considering the different attack feature vector or dynamic execution traces in dynamic program analysis and achieved scalability and high true positive rate. They adopted pattern analysis and build a malicious JS attack model using only JavaScript function calls and textual contents. In 2016, Yi *et al.* [11] proposed multiple classifier system (MCS) approach for detection of JavaScript malware with the help of machine learning techniques.(linear regression, logistic regression, decision tree classifier, and perceptron.) Here features are extracted from abstract syntax tree(AST) for training of the classifier. To increase the accuracy of classification result, the four classifiers are trained by original training set. Then the malicious and benign scripts are decided by the classifiers' votes. They found that linear regression was not appropriate for malware detection. In 2017, Patil *et al.* [12] employed a static analysis approach in web pages to classify the JavaScript as malicious or benign script using 7 machine learning classifier(NB, J48 Decision Tree, RF, SVM, AdaBoost, REPTree and ADTree). Collected malicious and benign data from different benchmark sources (Alexa, PhisTank, GeeksOnSecurity-GitHub) and extracted 77 static features of JS among which 45 are novel features and prepared a labeled dataset. The 7 machine learning classifiers were evaluated with training dataset and their results were compared with 9 antivirus software and found that the classifiers achieved good detection rate between 97%-99%, with very low FPR(False Positive Rate) and FNR(False Negative Rate), as compared to antivirus software. In 2018, Mereani *et al.* [2] adopted three classifiers like SVM, k-NN, and Random Forest to detect XSS which are coded in java script giving high accuracy (up to 99.75%) and precision (up to 99.88%) when applied to a large real world data set. The features selected in this article are divided into two groups like structural and behavioural. In the first category the complete set of symbols used in java scripts are considered and in the second group contains the commands and functions used in the java scripts. So it enables the classifiers to find an appropriate pattern based on these two features groups. A binary measure has been used to all the features which has provided high accuracy and precision in comparison to earlier experiments where weighted measures were used. In 2019, Ndichu *et al.* [9] employed Abstract Syntax Tree (AST) for code structure representation and Doc2vec, a feature learning approach to malicious JavaScript(JS) content and a classifier for AST feature classification. To improve the feature learning they employed two types of AST-JS implementations, AST-level merging and AST-subtree realignment. The features learned with Doc2vec were of low dimensions hence the detection rate was faster. Their approach was evaluated using the dataset D3M (Drive-by-Download Data by Marionette) for malicious JS codes and JSUNPACK plus Alexa top 100 websites datasets for benign JS codes. The advantage of this article is, it lessened the loads of manually selecting the functions, features and rules, also Doc2vec was able to detect the new and variant JS codes. Their feature learning approach gave better accuracy and fast classification in malicious JavaScript code.

IV. DISCUSSION

We have performed a systematic survey of XSS attack detection from malicious java script codes using machine learning techniques. For building a successful model a good set of feature set is required thus a significant feature selection technique is essential for increasing the accuracy of the model. In most of the articles we have found the features are extracted from functions and symbols used in java scripts but no such feature selection technique is adopted during training time. Hence we propose to design a framework which will implement an evolutionary feature selection technique for reducing the training time and plan to use ensemble machine learning classifiers along with batch machine learning classifiers to enhance the accuracy. Because the ensemble classifiers combine the weak classifiers to build the strong classifiers for obtaining the better classification result.

V. CONCLUSION

Most of the modern web applications are XSS vulnerable due to the improper input validation and sanitization. In this paper we have conducted a detail survey on XSS attacks with malicious java scripts using machine learning techniques. In future we plan to adopt a suitable feature selection technique and ensemble machine learning estimators to build a strong model which can be able to recognize the XSS attack from malicious java scripts effectively with less resources and less time.

TABLE I. COMPARISON OF DIFFERENT EXISTING APPROACHES FOR DETECTION OF XSS ATTACK

Articles	Techniques Used	Dataset	Feature Selection	Objective	Results
Likarish et al.[7], 2009	NB, ADTree, SVM, RIPPER ruler learner	XSSed.com	obfuscation scripts features	To classify malicious java script(JS) codes	Precision: 0.92%
Cova et al [10], 2010	NB	Spam trap, SQL injection,mal ware forum,Wepaw et-bad	!0 number of relevant features	To detect and analyze the malicious JS samples for finding out the Drive-by-Download attack.	False negative: 0.2%
Komiya et al.[3], 2011	SVM, Naïve-Bayes, k-nearest Neighbor	SQLAs XSS dataset	Java scripts based features	Classify the user input into malicious or benign one.	Accuracy: 98.95%
Nunan et al.[6], 2012	SVM and NB	Dmoz, ClueWeb09, XSSed.com	web document content and URL.	To automatically classify XSS attack from the malicious web pages.	Accuracy: 98.58% (Dmoz), 99.89% (ClueWeb09)
Wei-Hong et al. [4], 2013	SVM, Dynamic analysis	VX Heavens dataset	27 java scripts based features	To classify malicious scripts using a combined approach of machine learning with static detection.	Accuracy: 94.38%
Vishnu et al. [5], 2014	NB, SVM and J48 Decision Tree	XSS-URL dataset	URL and java script code.	To propose an approach for predicting XSS attack vector.	FPR(False Positive Rate) is better
Wang et al[8], 2015	AD Tree, RF, J48,NB	XSSed.com	Features are extracted from textual information, program structure and risky function calls	To propose a hybrid approach of static machine learning and dynamic program analysis to classify the malicious java scripts.	High true positive rate
Yi et al [11], 2016	linear regression, logistic regression, decision tree classifier, and perceptron	devtools.netw ork, Alexa	features are extracted from abstract syntax tree(AST)	To detect JavaScript malware using MCS approach	Accuracy: 97%
Patil et al. [12], 2017	NB, J48 Decision Tree, RF, SVM, AdaBoost, REPTree and ADTree	Alexa, PhisTank, GeeksOnSecurity-GitHub	77 javascripts based features are used.	To detect JavaScript code as malicious or benign using static analysis approach.	Accuracy: 93.63%
Mereani et al.[2], 2018	SVM, k-NN, and RF	Alexa, PhisTank	Structural and behavioral features of javascripts	To enable the classifiers to find an appropriate XSS pattern based on two features groups.	Accuracy: 99.75% Precision: 99.88%
Ndichu et al. [9], 2019	AST-JS, DOC2VEC	D3M, JSUNPACK	Java script content features	To detect the new and variant JS codes.	Accuracy: 98%

REFERENCE

- [1] Gupta, B. B., Shashank Gupta, S. Gangwar, M. Kumar, and P. K. Meena. "Cross-site scripting (XSS) abuse and defense: exploitation on several testing bed environments and its defense." *Journal of Information Privacy and Security* 11, no. 2 (2015): 118-136.
- [2] Mereani, Fawaz A., and Jacob M. Howe. "Detecting cross-site scripting attacks using machine learning." In *International Conference on Advanced Machine Learning Technologies and Applications*, pp. 200-210. Springer, Cham, 2018.
- [3] Komiya, Ryohei, Incheon Paik, and Masayuki Hisada. "Classification of malicious web code by machine learning." In *2011 3rd International Conference on Awareness Science and Technology (ICAST)*, pp. 406-411. IEEE, 2011.
- [4] Wei-Hong, W. A. N. G., L. V. Yin-Jun, C. H. E. N. Hui-Bing, and F. A. N. G. Zhao-Lin. "A static malicious javascript detection using SVM." In *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*. Atlantis Press, 2013.
- [5] Vishnu, B. A., and K. P. Jevitha. "Prediction of cross-site scripting attack using machine learning algorithms." In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, pp. 1-5. 2014.

- [6] Nunan, Angelo Eduardo, Eduardo Souto, Eulanda M. Dos Santos, and Eduardo Feitosa. "Automatic classification of cross-site scripting in web pages using document-based and URL-based features." In *2012 IEEE symposium on computers and communications (ISCC)*, pp. 000702-000707. IEEE, 2012.
- [7] Likarish, Peter, Eunjin Jung, and Insoon Jo. "Obfuscated malicious javascript detection using classification techniques." In *2009 4th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 47-54. IEEE, 2009.
- [8] Wang, Junjie, Yinxing Xue, Yang Liu, and Tian Huat Tan. "Jsdc: A hybrid approach for javascript malware detection and classification." In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pp. 109-120. 2015.
- [9] Ndichu, Samuel, Sangwook Kim, Seiichi Ozawa, Takeshi Misu, and Kazuo Makishima. "A machine learning approach to detection of JavaScript-based attacks using AST features and paragraph vectors." *Applied Soft Computing* 84 (2019): 105721.
- [10] Cova, Marco, Christopher Kruegel, and Giovanni Vigna. "Detection and analysis of drive-by-download attacks and malicious JavaScript code." In *Proceedings of the 19th international conference on World wide web*, pp. 281-290. 2010.
- [11] Yi, Zibo, Jun Ma, Lei Luo, Jie Yu, and Qingbo Wu. "Multiple Classifier Systems for More Accurate JavaScript Malware Detection." In *International Conference on Promotion of Information Technology (ICPIT 2016)*. Atlantis Press, 2016.
- [12] Patil, Dharmaraj R., and J. B. Patil. "Detection of malicious javascript code in web pages." *Indian Journal of Science and Technology* 10, no. 19 (2017): 1-12.
- [13] Goswami, Swaswati, Nazrul Hoque, Dhruva K. Bhattacharyya, and Jugal Kalita. "An Unsupervised Method for Detection of XSS Attack." *IJ Network Security* 19, no. 5 (2017): 761-775.
- [14] Goswami, Swaswati, Nazrul Hoque, Dhruva K. Bhattacharyya, and Jugal Kalita. "An Unsupervised Method for Detection of XSS Attack." *IJ Network Security* 19, no. 5 (2017): 761-775.
- [15] Jena, Sanjay. "XSS attack prevention using DOM based filtering API" , May, 2014.