Detecting Blind Cross-Site Scripting Attacks Using Machine Learning

Gurpreet Kaur
Department of Information Systems Security and Assurance Management
Concordia University of Edmonton,
Edmonton, Alberta, Canada
gkaur2@student.concordia.ab.ca

Yasir Malik
Department of Information Systems Security and Assurance Management
Concordia University of Edmonton,
Edmonton, Alberta, Canada
yasir.malik@concordia.ab.ca

Hamman Samuel
Department of Information Systems Security and Assurance Management
Concordia University of Edmonton,
Edmonton, Alberta, Canada
hamman.samuel@concordia.ab.ca

Fehmi Jaafar
Department of Information Systems Security and Assurance Management
Concordia University of Edmonton,
Edmonton, Alberta, Canada
fehmi.jaafar@concordia.ab.ca

ABSTRACT
Cross-site scripting (XSS) is a scripting attack targeting web applications by injecting malicious scripts into web pages. Blind XSS is a subset of stored XSS, where an attacker blindly deploys malicious payloads in web pages that are stored in a persistent manner on target servers. Most of the XSS detection techniques used to detect the XSS vulnerabilities are inadequate to detect blind XSS attacks. In this research, we present machine learning based approach to detect blind XSS attacks. Testing results help to identify malicious payloads that are likely to get stored in databases through web applications.

CCS Concepts
• **Security and privacy → Web application security;**

Software and its engineering → Software creation and management.

Keywords
Software Security; Web Security; Cross-Site Scripting (XSS); Machine Learning; Vulnerability Detection

1. INTRODUCTION
The popularity of web applications is growing rapidly, and a variety of applications and services are available for user's convenience, e.g. healthcare, banking, shopping, socializing, etc. While these services and applications greatly improve productivity and convenience, they also introduce several new security vulnerabilities. Exploitation of such vulnerabilities is prone to cyber attacks and an alluring focus for digital criminals that could lead to severe impacts for organizations. The major cause of attacks on the Web is mostly due to security vulnerabilities in their application design, inappropriate input validations, inadequate security controls, etc. Among the Web security vulnerabilities, Cross-Site Scripting (XSS) is the most commonly exploited vulnerability [1]. XSS is a type of security vulnerability, where an attacker injects malicious scripts into a legitimate web application. By leveraging XSS, an attacker would misuse an input vulnerability within a web application and used it to propagate malicious code. These attacks are more dangerous as it provides a surface for other types of attacks such as phishing, keylogging, cookie theft, etc. that can be executed on the network. There are numerous variations of XSS attacks that are implemented to get access to confidential data of legitimate users. These attacks are mainly categorized into three groups. i.e. reflected XSS, stored XSS, and DOM- based XSS. The stored XSS vulnerabilities are harder to detect as compared to reflected XSS.

In this research, we focused on analyzing and detecting blind XSS attacks (a subset of stored XSS), using machine learning techniques. Compare to generic stored XSS, blind XSS attacks are difficult to detect as the execution time of the malicious payloads are unknown, which makes it difficult to detect these attacks. Moreover, there are very few resources available to review the execution of attacks in different web applications as they present different behavior from other types of stored XSS. Our approach leverages machine learning techniques to study the behavior of blind XSS. We implemented a linear Support Vector Machine (SVM) classifier for detecting blind XSS. The linear classification of SVM helps to determine the difference between the features of blind and stored XSS as there is a fine line between both attacks. Testing results help to identify malicious payloads that are likely to get stored in databases through web applications.

The rest of the paper is organized as follow. In section 2, we present the review of related research work on different detection solutions. The proposed approach of blind XSS attack and detection model is presented in section 3. Section 4 presents the discussion on experiments and results. Finally, in section 5, we conclude the paper, with recommendations for future work

SPML '18, November 28–30, 2018, Shanghai, China

DOI: https://doi.org/10.1145/3297067.3297096

2. RELATED WORK

Numerous detection solutions have been proposed to detect the Cross XSS vulnerabilities. In this section, we present a review of related work similar to our approach. Choi et al. in [2] proposed a detection based on N-gram and SVM classifier to identify malicious code in the input string. Here N-gram is the continuous subsequence of N consecutive tokens in a stream of tokens which is used for the feature extraction. SVM algorithm is used for nonlinear separation. The tokenizer replaces the test data, splits test data into tokens and interprets it. However, the tokenizer needs continuous training for the new malicious codes.

In another work, Gupta et al. [3] suggested using machine learning based solution to extract the apropos features to categorize the vulnerable from the legitimate code. Their solution implements SVM, NB, Bagging, J48 and JRip classifiers to extract user-input context features and some basic features that shows the characteristics of input, output, validation and sanitization routines. Shar et al. in[4] proposed prediction models based on classification and clustering techniques to analyze XSS vulnerabilities. Static analysis is used for classification of nodes that provide a clear security-related purpose. Since the classification of nodes that invoke user-defined or language built-in matching function cannot be categorized by static code analysis, a dynamic method was used. The hybrid attributes were identified that are useful to predict SQL as well as XSS vulnerabilities by using supervised and unsupervised learning.

In another work [5] authors extended their prediction technique to increase the detection accuracy, while leveraging the advantages provided by existing static and dynamic taint analysis approaches. However, more experiments are required to determine the feasibility and advantages of integrating multiple approaches in the prediction model. Gupta et al. in [6] suggested an automated detection system that helped to detect JavaScript malicious injections vulnerabilities within client-side. The system was evaluated using three key agents (XSS Attack Vector Injector, HTML Crawler, and Script Locator.) of PHP web application.

Krishnaveni et al. in [7] emphasize classification of XSS web page attacks by using a feature extraction method of web page and URL using numerous data mining techniques. The experiments were coordinated to three types of XSS attacks using machine learning techniques like Naive Bayes (NB), Decision Tree (DT) and Multi- Layer Perception (MLP). However, this method has some limitations for the other variations of stored XSS, such as blind XSS.

The review of literature helps us to identify some gaps that are still unsolved that makes web applications still vulnerable to numerous types of XSS attacks [8]. Some of the studies mentioned above focused on detecting the vulnerabilities of stored XSS reflected and DOM-based attacks. However, stored XSS vulnerabilities are harder to detect compared to reflected XSS vulnerabilities, where script execution is immediately reflected. Moreover, in the case of blind XSS, the execution of malicious payloads can take a long time, thus cannot be tested using traditional methods. In this research, we focus on detecting blind XSS. In the next section, we highlight the limitation of existing approaching in detecting blind XSS and introduce our detection approach.

3. BLIND XSS DETECTION

The blind XSS is kind of stealth-based attack in which the data retrieval point is not accessible by the attacker, due to lack of privileges or some action/event required for payload execution. The exploitation technique of blind XSS is different from other classic XSS vulnerabilities. i.e., instead of sending the malicious URL to the site administrator with XSS payload, an attacker needs to wait until the site administrator opens administrator panel for malicious code to be triggered or execute. This is mainly because the malicious script is not the same as the content displayed to the victim. The detection of blind XSS is still a challenge because its payload execution time is unknown. Moreover, there are fewer resources available to review attack execution in different web applications.

This research contributes to the detection of blind XSS attacks in the web application. We adopted machine learning techniques to study the behavior of blind XSS. The machine learning classifier provides a computer system with the ability to learn with data by using statistical techniques. The Support Vector Machine (SVM) model is used for both classification and regression problems, but for our research, we use SVM for linear classification as there is a fine line between blind and stored XSS attacks.

In machine learning classification technique, feature extraction is a crucial phase. In our analysis, the features of blind XSS are extracted based on JavaScript events such as (onclick, onload, onplay, onshow, onselect, etc.) used by attackers to insert vulnerable payloads. By using these events, the execution time of payloads is delayed, mostly unknown. The classifier decides the vectors of each training data in feature space and the parameter that decides hyperplane, which can separate training data into classes. The linear classification helps to determine the difference between the features of blind and stored XSS and to separate blind XSS attacks from stored XSS attacks by features extracted from the dataset.

In the linear SVM classification process, the classifier classifies test data using $(a1, b1) \cdots (an, bn)$, Where ai is either 1 or -1, which indicates the class to which the $a1$ points belong bi is a p-dimensional real vector. Here we want to know the maximum margin hyperplane that separates the group of points ai for Which bi = 1 from the group of points for which $bi = -1$. The hyperplane is written for the group of points ai as shown in

equation 1.

$$a.\, x - b = 0 \qquad (1)$$

Here x is a normal vector to the hyperplane. For our research, the training dataset is linearly separable. For the implementation of the SVM classifier, we use two arrays as described below in equation 2 and 3:

$$x = x1 \qquad (2)$$

$$\text{y} = y1 \qquad (3)$$

where $x1$ represents the predicting values in array x. $y1$ stands for the target values that help to classify the blind XSS attack. As the SVM classifier takes the binary values for an array, we converted the string values into binary values through the LabelEncoder () function. The dataset is ready and can predict the data through the SVM classifier. As mentioned above, we are using a linear separable dataset.

$$clf = SVC(kernel = linear\ ) \qquad (4)$$

$$cl\,f : f\,it(array1;\ array2) \qquad (5)$$

$$cl\,f : predict(value1; value2) \qquad (6)$$

Equation 4 and 5 determines array1, array2 samples, and classes. After the values are collected, the trained model can now predict new values. The result of the geometric expression in equation 6 specifies whether the result falls into a blind XSS attack or not. In the next section, we show our experiment and implementation results.

4. EXPERIMENT AND RESULTS

We experimentally evaluated the feasibility of blind XSS attack and its detection by using machine learning algorithm. To evaluate our approach, we simulated the blind XSS on Mutillidae (an open-source, vulnerable web application) [9]. The experimental setup includes three virtual machines attacker, firewall and server. The attacker machine is configured with a Windows operating system and all other machines are configured with the Ubuntu operating system. The detection technique relies on the extracted feature and trained dataset which is collected from various sources, including GitHub repositories [10], GBHackers [11] and XSS-payloads [12].

In our experiment, an attacker tries to insert the malicious payload in the comment section of a web page which is vulnerable to blind XSS attack. However, the execution of the malicious payload is unknown. The script is stored in the database and will execute when the vulnerable webpage is visited. When the condition of attack is triggered, the victim visits the malicious blog and is redirected to the link, set by the attacker. The attacker gets a callback through email, when victim accesses the attacker's blog, along with the session cookies, user ID and password.

The attacker initiates the attack by inserting JavaScript with a malicious payload in the comment section of website. Our approach detects the malicious payload before it gets executed. The learning model is trained by assigning values to two different parameters of the SVM algorithm. The first step of the detection technique relies on the JavaScript events of the dataset script. Once the script is inserted by an attacker in the malicious website, the prediction model configured on web application firewall starts predicting the attack based on features and trained dataset. The data matrix is generated based on the prediction result. The support vector performs the linear classification of predicted data, and output is generated based on defined criteria. Figure 1 represents the attack detected by the prediction model using the SVM linear classification model. The attack detected by the prediction model using the SVM linear classification model. The graph is represented in a two-dimensional matrix, where the bar with prediction value at the right hand determines the prediction capability of the SVM prediction model implemented to determine blind XSS attack. The prediction value of blind XSS payloads is represented within 0 to 40 value, which further represents the number of scripts analyzed and predicted by the SVM algorithm. The three other quadrants of the graph represent the regular payloads analysis of scripts (stored XSS), which are not blind. The predictability of the SVM algorithm is represented in the color bar range and shown in the confusion matrix to represent the linear difference in regular (stored) and blind XSS payloads prediction. We performed an analysis by collecting 1400 payloads. Table 1 shows the accuracy of our detection model. The false positive of our detection approach is 0.111. The detection accuracy of our approach is 95.4%, and the remaining 4.6% payloads are unknown because of their variation in signature of attacks.

Table 1 Performance of SVM Classifier

Classifier	Precision	Recall	Accuracy	False Positive
SVM	0.956	0.951	95.4	0.111

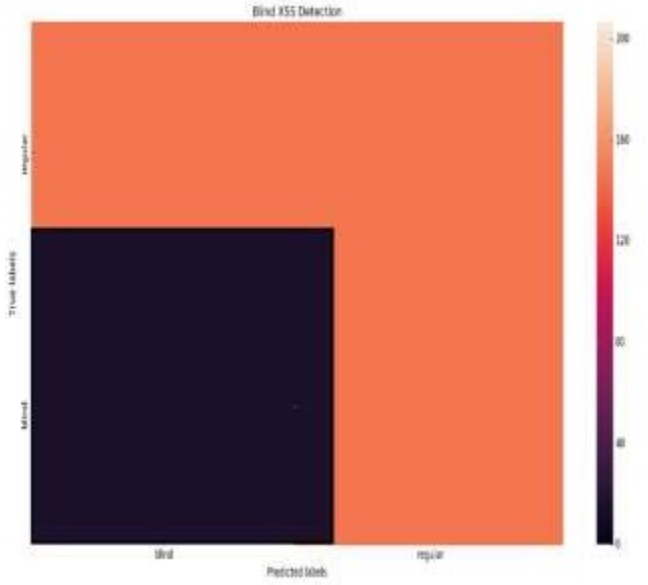

Figure 1 Blind XSS Attack Detection Using SVM

5. SECTIONS

In this paper, we studied and analyzed blind XSS attack and its detection by using machine learning techniques that will help to security professionals to prevent or secure the web applications from such attacks. We implemented the SVM for linear classification to determine the difference between the features of blind and stored XSS. Our research helps to classify malicious payloads by extracting various features of payloads that are likely to get stored persistently in databases About 1400 payloads are tested to detect blind XSS attacks, and the accuracy of tested results shows that our approach can detect blind XSS attacks with 95.4% of accuracy. For future work, we are interested to analyze new dataset to extract that can be used as a benchmark for the OWASP community to extend web security research. Moreover, by adding or removing the extracted blind XSS features the detection result varies.

6. ACKNOWLEDGMENTS

This paper has been partly funded by Concordia University of Edmonton and by the Computer Research Institute of Montréal (CRIM). We gratefully thank the Ministère de l'Économie, Science et Innovation du Québec (MESI) for its generous support.

7. REFERENCES

[1] Isatou Hydara, Abu Bakar Md. Sultan, Hazura Zulzalil, Novia Admodisastro, Current state of research on cross-site scripting (XSS) - *A systematic literature review, In*

Information and Software Technology, Volume 58, 2015, Pages 170-186, ISSN 0950-5849

[2] J. Choi, H. Kim, C. Choi and P. Kim, "Efficient Malicious Code Detection Using N-Gram Analysis and SVM," *2011 14th International Conference on Network-Based Information Systems*, Tirana, 2011, pp. 618-621.

[3] M. K. Gupta, M. C. Govil and G. Singh, "Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications," *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Songkhla, 2015, pp. 162-167.

[4] L. K. Shar, H. Beng Kuan Tan and L. C. Briand, "Mining SQL injection and Cross-site scripting vulnerabilities using hybrid program analysis," *2013 35th International Conference on Software Engineering (ICSE)*, San Francisco, CA, 2013, pp. 642-651.

[5] L. K. Shar, L. C. Briand and H. B. K. Tan, "Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning," in *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 6, pp. 688-707, Nov.-Dec. 1, 2015.

[6] Shashank Gupta and B. B. Gupta, "Automated discovery of JavaScript code injection attacks in PHP web applications", *International Conference on Information Security & Privacy (ICISP)*, Nagpur, INDIA, 11-12 December 2015, Elsevier, Procedia Computer Science, vol. 78, pp.82 – 87, 2016

[7] Patents.justia.com. (2018). US Patent for Persistent cross-site scripting vulnerability detection Patent (Patent # 9,948,665 issued April 17, 2018) - *Justia Patents Search*. [online] Available at: https://patents.justia.com/patent/9948665 [Accessed 14 Jun. 2018].

[8] S. K. Mahmoud, M. Alfonse, M. I. Roushdy and A. B. M. Salem, "A comparative analysis of Cross-site Scripting (XSS) detecting and defensive techniques," *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, Cairo, 2017, pp. 36-42

[9] https://sourceforge.net/projects/mutillidae/ [online] [Accessed: 1 Jun 2018].

[10] GitHub. (2018). *swisskyrepo/Payloads All The Things*. [online] Available at: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/X SS%20 injection [Accessed:1 Jun. 2018].

[11] GBHackers On Security. (2018). *Top 500 Most Important XSS Cheat Sheet for Web Application Pentesting*. [online] Available at: https://gbhackers.com/top-500-important-xss-cheat-sheet/ [Accessed 1 Jun. 2018].

[12] *Cross Site Scripting Payloads ≈ Packet Storm*. [online] Available at: https://packetstormsecurity.com/files/112152/Cross-Site-Scripting- Payloads.html [Accessed 1 Jun. 2018].