

A Hybrid Machine Learning Model to Detect Reflected XSS Attack

Beraat Buz, Berke Gülçiçek, and Şerif Bahtiyar*


Abstract—Since web technologies are getting more advanced with longer codes, the number of vulnerabilities has increased considerably. Cross-site scripting (XSS) attacks are one of the most common attacks that use vulnerabilities in web applications. There are three types of cross-site scripting attacks namely, reflected, stored, and DOM-based attacks. Reflected XSS attacks are the most common type that is usually implemented by injecting a malicious code into the URL and then sending the URL to the targeted system by using phishing methods, which is a significant threat for recent web applications. Our motivation is the lack of a high-performance detection method of reflected XSS attacks with high accuracy. In this paper, we propose a hybrid machine learning model to detect vulnerabilities related to reflected XSS attacks for a given URL of a website. Our model uses a scanner to discover vulnerabilities in a web site and convolutional neural networks to predict the most common vulnerabilities that may be used for reflected XSS attacks, which makes the proposed model hybrid. We analyzed the model experimentally. Analyses results show that the proposed model is able to detect vulnerable attack surfaces with 99 % accuracy.

Index Terms—Deep Learning, Detection, Reflected XSS, N-gram, Vulnerability, XSS Scanner.


I. INTRODUCTION

RECENTLY, THE number of web applications have increased dramatically with the rapid proliferation of the Internet. More and more applications, even complex ones, are converted into web applications. A lot of new ideas are also implemented using web technologies. People with different levels of expertise are working to develop web applications


BERAAT BUZ, is with Department of Computer Engineering of Istanbul Technical University, Istanbul, Turkey, (e-mail: buz16@itu.edu.tr).

 <https://orcid.org/0000-0002-9455-1537>

BERKE GÜLÇİÇEK, is with Department of Computer Engineering of Istanbul Technical University, Istanbul, Turkey, (e-mail: gulcicek16@itu.edu.tr).

 <https://orcid.org/0000-0002-2282-5404>

ŞERİF BAHTİYAR, is with Department of Computer Engineering of Istanbul Technical University, Istanbul, Turkey, (e-mail: bahtiyars@itu.edu.tr).

 <https://orcid.org/0000-0003-0314-2621>

*Corresponding Author

Manuscript received April 25, 2021; accepted July 27, 2021.

DOI: [10.17694/bajece.927417](https://doi.org/10.17694/bajece.927417)

that case has created many additional vulnerabilities. This circumstance takes attentions of adversaries therefore web application security has become more significant than ever. Analyses also show that 32% of the web applications have extremely poor security levels, and 23% of web applications have poor security levels [1]. Even websites that seem secure may have vulnerabilities. For instance, an XSS security flaw was discovered in the UK Parliament website (Gupta, 2014) [2].

One of the most common types of web application attacks is cross-site scripting attacks. According to OWASP, crosssite scripting attacks are the 7th most common type of web application security risk [3]. In cross-site scripting attacks, an adversary may inject malicious code into some parts of the web application. The vulnerable parts of the website are also parts that involve user input. The malicious code is usually a JavaScript script that steals information of other users. The vulnerable parts of web applications allow attackers to exploit the website by using session hijacking, misinformation, defacing web site, inserting hostile content, phishing attacks, taking over user's browser, pop-up-flooding, steal personal information and access to business data [4].

Cross-site scripting attacks can be categorized into three types. Reflected cross-site scripting attacks, stored cross-site scripting attacks, and DOM-based cross-site scripting attacks as shown in Figure 1. In this paper, our focus is on reflected cross-site scripting attacks, which are also called nonpersistent XSS attacks. These attacks are the most common XSS attack type. It can be implemented by injecting malicious code into the URL of the website, or in a form element of the website where user input is taken. Either way, when the victim opens the website, a malicious script is run.

The initial step of a reflected cross-site scripting attack is to find vulnerabilities in the targeted system. The vulnerabilities usually reside in the parts where user input is taken. It can be the URL of the website, or an HTML form. The attacker injects the malicious code in these areas. For example, in the case of the URL, the attacker sends URL containing malicious code using phishing methods. When the victim opens the URL, the victim's browser will start executing the malicious code.

Stored cross-site scripting attacks, which are also called persistent XSS attacks, are implemented by adding a malicious code snippet into the database of a website. When a user enters that page, the malicious code is executed in the user's browser. On the other hand, DOM-based XSS attacks use

vulnerabilities found in web browsers.

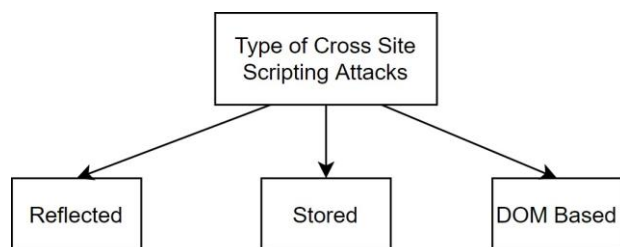


Fig. 1. Types of XSS Attacks

In this paper, we propose a hybrid model, two-step solution, implemented as a tool. In the first step, vulnerable parts of a web application or a web site is analyzed to discover potential vulnerabilities that allow reflected-XSS attacks. Specifically, the tool crawls the website for potentially vulnerable areas. In the next step, potential reflected-XSS attacks are detected. Particularly, the payloads that managed to successfully attack are run through the deep learning model that is trained with payloads beforehand. The deep learning method tries to guess the dangerous payloads. In the payload file, there are thousands of payloads.

To the best of our knowledge, the proposed model is a unique two-step hybrid solution that detect potential reflected XSS attacks with 99% of accuracy. The proposed model is adaptive to newly found web vulnerabilities that provides better accuracy than existing solutions. Experimental analyses verify that our model provides high accuracy with adaptive nature.

The rest of the paper is organized as follows. Section II is about XSS attacks and machine learning algorithms. In the next section the proposed hybrid model to detect reflected-XSS attacks is presented. We analyze the proposed model in Section IV. The last section is devoted to conclusion.

II. CROSS-SITE SCRIPTING ATTACKS AND MACHINE LEARNING

There are many researches about XSS attacks and their prevention methods. For instance, Sarmah, Bhattacharyya, and Kalita explain the type of XSS attacks, which have occurred for two decades [5]. In the research, both client side and server-side XSS attacks detection approaches are analyzed, including static analysis, dynamic analysis, and hybrid analysis mechanisms. On the other hand, Liu et al. explains what makes XSS attacks dangerous by explaining of XSS attacks [6] with static and dynamic analysis methods to detect the vulnerabilities in the system.

Cookies and cookie theft, as well as analysis of tools are significant about XSS attacks detection researches [7]. Types of XSS attacks and in which conditions they arise, how are cookies used, what type of cookies are used, how they can be stolen, which detection tools are common, and how the detection tools work are explained in [7]. However, there is always a detection accuracy of XSS attacks with these tools.

Galan et al. discuss a scanner used to detect vulnerabilities about stored XSS in a website [8]. In the architecture, the Webpage parser agent is the first to be launched. Script injector agent is the next agent, and it takes the attack point repository created by the previous agent as input. Finally, a verification agent, which may not be run until the injector agent, has created a list to launch attacks. Li and Wei create a model for a more efficient automatic XSS detection tool by using SVM algorithm, which is used to determine whether parameters submitted by users are malicious or not in case of XSS attacks [9]. Also, they use DQN algorithm for reinforcement learning for bypassing the rule-based WAF system.

From another point of view, Syaifuddin et al. explain how to prevent XSS attacks with a honeypot [10]. In this approach, when the program detects anomalies on the URL request packet, the honeypot records its log and the URL request. Then, according to the log file, they implement a snort rule.

A dynamic detection technique for XSS attacks is explained with dynamic detection algorithm that contains five steps, namely crawler, feature construct, attacks simulation, results in detection, and report generation [11]. Authors summarize common detection methods that include dynamic analysis based on black-box testing, static analysis based on white box testing, and fuzzing test. They compare three typical XSS attack detection tools which are XSS-ME, Wapiti and Punk.

Habibi and Surantha briefly explain XSS attack detection methods, which are created by using Support Vector Machine (SVM), K-Nearest Neighbour (KNN), and Naive Bayes (NB) techniques with N-gram method which is a method for detecting similarities between two sentences [12]. Dong et al. explore possible XSS vulnerabilities in HTML5 [13]. They introduce a XSS attack detection tool that produces a large number of test emails by configuring each checkpoint of a test email with an attack vector derived from their repository and then connects to the SMTP server in the detection process. The tool automatically sends test emails to target mailboxes after they have been checked and approved by SMTP. When these test emails are successfully opened in target mailboxes, they may quickly evaluate if the attack vectors on each checkpoint have been filtered.

Li et al. represent their XSS attack detection approach based on the attention mechanism of Long Short-Term Memory (LSTM) recurrent neural network [14]. The proposed XSS attack detection model is based on LSTM. In addition, recall and precision are weighted harmonic means, and the F1 metric is a weighted harmonic mean of these two metrics [15].

These researches provide some solutions to detect XSS attacks. However, they barely satisfy increasing number of vulnerabilities about XSS attacks with high accuracy that depends on precision and recall. In the light of all this information, we propose a hybrid detection model that is more accurate according to two parameters, namely precision and recall. Vulnerabilities about reflected XSS have been a huge threat for societies that use web applications and web sites.

III. A HYBRID MODEL TO DETECT REFLECTED-XSS ATTACKS

We have proposed a hybrid model that has two important features. The first one is a scanner of potential vulnerabilities that allow reflected XSS attacks and the other one is a detection mechanism for XSS attacks. We also implemented the model as a tool called XSS-Guard.

XSS-Guard takes an URL of a website in the first step about reflected XSS. This URL may be on a web server or on a local server. There are two options to scan the website, with a crawler or without a crawler. In the crawler mode, the tool searches all usable links that are not used before. Then, these links are scanned to find vulnerabilities related to reflected XSS attacks.

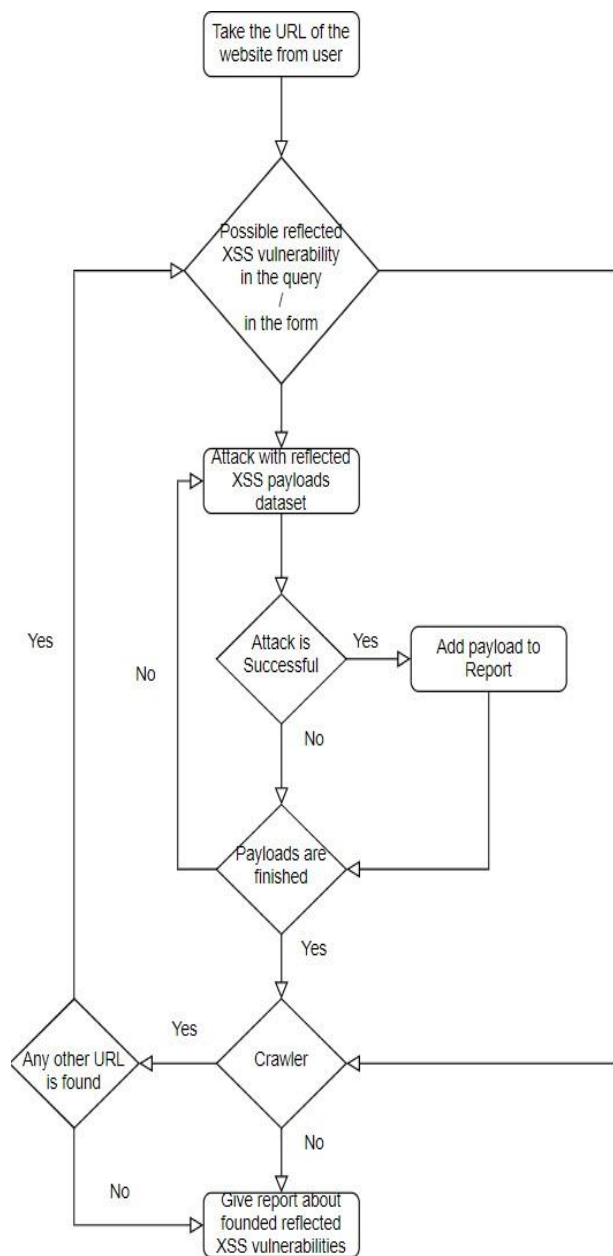


Fig. 2. The algorithm of the scanner

When the website is ready, and there are suitable places for reflected XSS attacks, the tool starts to attack with static reflected XSS payloads. Our payload dataset contains 8000 lines of script codes. These scripts are found from various sources. If a script run on targeted system, this means that there is at least one vulnerability that allows reflected XSS attacks. Vulnerabilities about this script are added to the vulnerability list of the scanner. Figure 2 shows the algorithm of this process.

In the detection mechanism, we have used a convolutional neural network (CNN) that is usually used for image classification, medical image analysis, and natural language processing. CNN provides an effective approach to prevent reflected XSS attacks. We have used a built-in Keras to determine if the script is reflected XSS attack data or benign data.

The proposed model starts by reading the dataset from a csv file where payloads and labels are stored. Then, each payload is converted to an ASCII value array and the array is resized. Next, we convert the data in the form that it is suitable to be processed by CNN. Moreover, we split the dataset as train payloads, train labels, test payloads, and test labels.

We use sequential approach in our model. At the beginning, we have three convolution layers (2D) and three max pooling layers. To flatten data in matrix form, we use a flatten layer. Next, the model is finished with four dense layers. We compile the proposed model with Adam optimizer. We tested many optimizers and we found out that Adam optimizer provides better results in our approach.

In our model, the batch size is 128 and the number of epochs is 10. After we fit the model, we move on to the test-set phase. We accept values greater than 0.5 as attack data, and values less than 0.5 as benign data. Finally, we calculate the accuracy, the precision and the recall values.

Our dataset contains approximately 13000 data with different payloads that are used in the scanner. This dataset has two columns. The first column is script data and the second column is label data in which "0" means XSS attack data, and "1" means benign script data.

- `<inputtype = imagesrc = 1onerror = alert(1) >`
- `<htmlontouchstart = alert(1) >`
- `<xonmouseover = alert(1) >`
- `%3Cxonxxx = alert(1)`

In the list above, you can see a few example payloads from the dataset. Each XSS attack payload is a JavaScript that tries to pop an alert. In this way, we check the success of the attack if the payload successfully popped an alert in the website. This makes easy to automate the testing process.

The script may be inside of different HTML tags, such as img, html, body, form, and etc. It may also be in different HTML events, such as onmouseover, ondrag, and etc. Attackers can use such tricks to get the browser to execute the code, therefore we take into account these tricks. The script may also be encoded to escape simple sanitation techniques.

After the training with CNN, N-gram model is used to detect XSS attacks as shown in Figure 3. N-gram algorithm is

used to find the repetition rate in a consecutive sequence. The variable expressed with n represents the value by which the repetition is controlled. The gram corresponds to the weight of this repeated value in the array. N-gram model requires a payload dataset therefore we have used all XSS attacks scripts and get the high-frequency words. Specifically, we use 9-grams and 10-grams which means that words have only 9 or 10 characters. Since these words may contain benign words like the "javascript" word, we have passed these high-frequency words through the model. Finally, restricted words are obtained.

After completing the scanning step, the model moves to the detection step. Initially, the created scripts are tested with CNN model. Then, the restricted words are used to determine benign actions or XSS attacks. Finally, XSS-Guard creates a report about the results.

IV. ANALYSIS OF PROPOSED MODEL

We create a reflected XSS attack detection tool, called XSS-Guard, on Ubuntu 20.04 LTS Operating system by using python3. The foremost requirements are Keras, Selenium, PyQt5, and OpenCV.

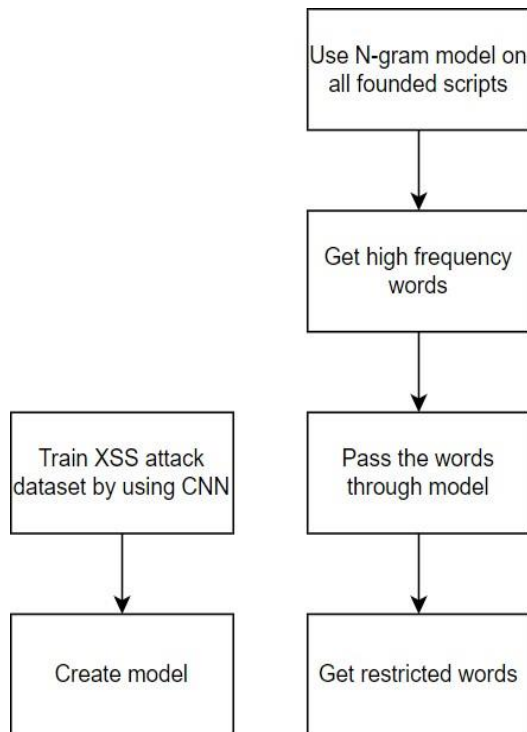


Fig. 3. The algorithm of the detection mechanism

We tested the proposed model with the tool on Web for Pentester from pentesterlab.com. We used a docker version from Github. This website is basically a website designed to be vulnerable against attacks with the purpose to test and practice applications. We used XSS-Guard with one of examples on the website. Our tool is used to attack the URL with around 500 payloads and managed to find 26 of these

payloads successfully, which means the website is vulnerable to reflected-XSS attacks as shown in Figure 4.

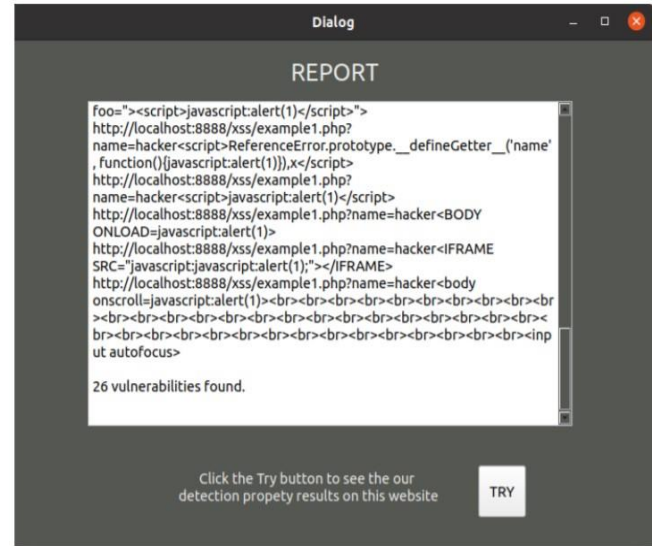


Fig. 4. Some results of XSS-Guard

We have used the following four metrics to specify the detection step of the proposed model with CNN.

- True Positive (TP): The number of attack instances identified as attacks.
- True Negative (TN): The number of instances of non-attacks known as non-attacks.
- False Negative (FN): The number of cases of attack defined as non-attacks.
- False Positive (FP): The number of cases of non-attacks classified as attacks.

We created a confusion matrix with TP, TN, FP, FN. We analyze the detection performance metrics that use the confusion matrix with the classification model. Performance metrics calculated using the confusion matrix are:

Accuracy: The estimated correct classifications are divided by the total number of classifications as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (1)$$

Precision: Determine how accurate is when a positive estimate is obtained from the classification. This is divided by the number of correctly predicted positive instances by the total number of positive predictions, as true or false as follows.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall: It is also known as Sensitivity or True Positive Rate (TPR). Recall is the number of positive predictions divided by the number of positive classified values in the test data calculated as follows.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

These three metrics are used to observe the success of the neural network that is used for the detection mechanism and how the n-gram model boosts it.

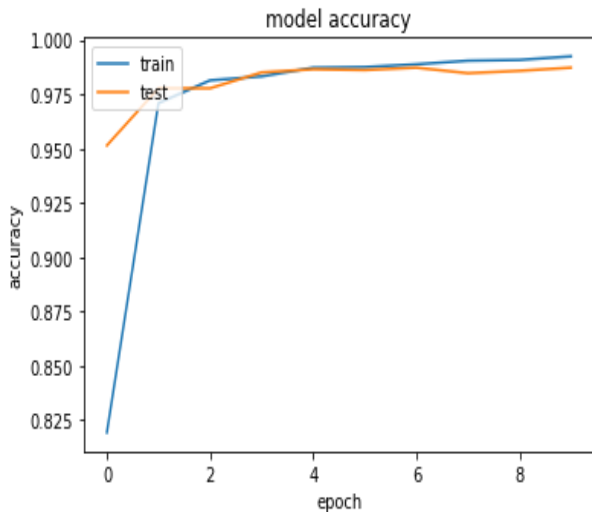


Fig. 5. The accuracy model

We used Adam optimizer to compile the model. Although RMSprop optimizer and Stochastic Gradient Descent optimizer also achieve high accuracy, Adam optimizer provides best performance in our cases. In the experiments, we have 10 epoch sizes and 128 batch sizes to fit the model. In the following figures, visualizations of model accuracy and model loss are shown. These models are created by using CNN without N-gram model.

The accuracy of the proposed model with experimental evaluation is shown in Figure 5. Experimental results show that after the first epoch, the value of accuracy on the training dataset increase considerably. Additionally, the accuracy for both datasets rise till the end of epochs.

We show the experimental results of the loss model in Figure 6. It can be observed that the behavior of loss is the same as the behavior of accuracy. Differently from the accuracy, the loss decreases with the similar behavior. The performance of the loss model for both train and validation datasets are parallel. The performance results for the three metrics are given on Table I related to the validation dataset. These results show that XSS-Guad detects XSS attacks with high performance.

TABLE I
PERFORMANCE RESULTS WITHOUT N-GRAM

| Performance Measures | Values |
|----------------------|--------------------|
| Accuracy | 0.9890430971512053 |
| Precision | 0.9852546916890008 |
| Recall | 0.9945872801082544 |

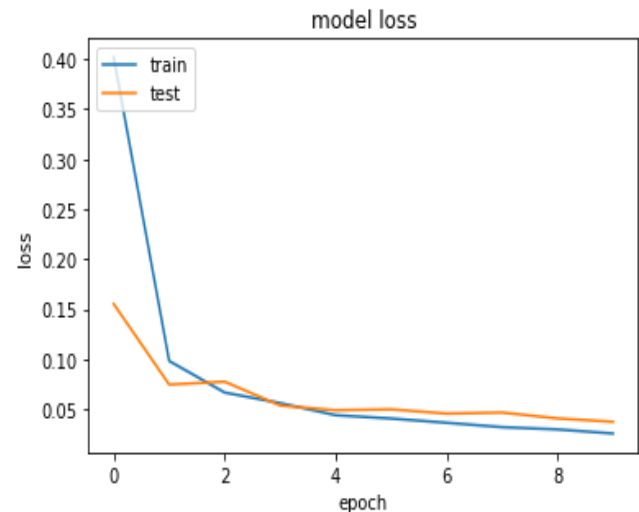


Fig. 6. The loss model

In addition to CNN, we used N-gram model to restrict some words since scripts may be hidden inside many non-sense characters. The performance results with N-gram model is shown in Table II. The performance results show that using Ngram helps to increase the performance of accuracy. Actually, if these payloads are found as benign in our model, we can control these restricted words to label them as XSS attacks. Some of these restricted words we found by using N-gram and passing through the model are as follows:

- ypress = ""
- nstart = ""
- solute;""
- useout = ""
- ofocus ><

TABLE II
PERFORMANCE RESULTS WITH N-GRAM

| Performance Measures | Values |
|----------------------|--------------------|
| Accuracy | 0.9901387874360847 |
| Precision | 0.9872397582269979 |
| Recall | 0.9945872801082544 |

In order to validate the performance of the hybrid model, we have used K Fold Cross Validation. We chose K to be 5. Table III contains the accuracy results of 5 fold cross validation in our dataset with or without N-gram algorithm. These results also show that N-gram provides better accuracy results for the proposed model.

TABLE III
CROSS VALIDATION RESULTS

| # Group | Without N-gram | With N-gram |
|---------|---------------------------|---------------------------|
| 1 | Accuracy : 0.982191780821 | Accuracy : 0.988645838609 |
| 2 | Accuracy : 0.987671232876 | Accuracy : 0.990244198456 |
| 3 | Accuracy : 0.986301369863 | Accuracy : 0.989195991534 |
| 4 | Accuracy : 0.989497716894 | Accuracy : 0.991157563924 |
| 5 | Accuracy : 0.977168949771 | Accuracy : 0.980293547470 |

After creating the model and the restricted words, we tested the detection mechanism to prevent XSS attacks on the founded XSS attack scripts by using XSS-Guard, the tool we developed. We detected 25 XSS attacks scripts as XSS attacks and one XSS attack script as benign. The results are shown in Figure 7.

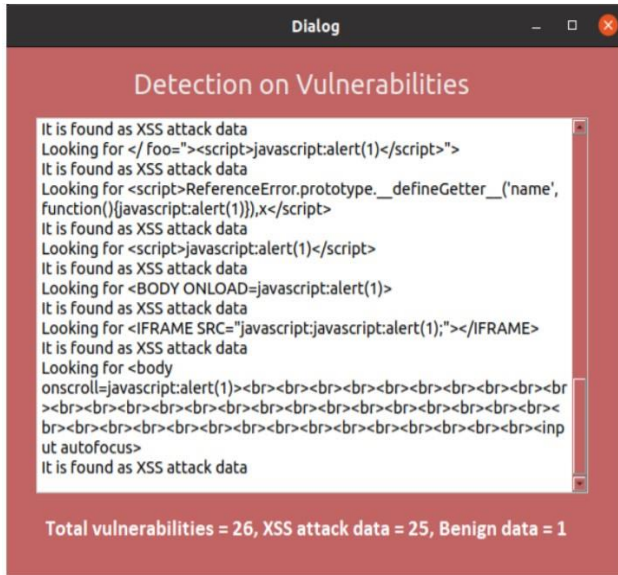


Fig. 7. An example results for reflected XSS attack detection

Overall analyses results show that classifications by using N-gram provides better accuracy for detecting reflected XSS attacks. Additionally, the proposed hybrid model, which implemented on XSS-Guard helps to determine vulnerable websites against reflected-XSS attacks with high accuracy. Thus, the proposed hybrid model is expected to counter reflected XSS attacks more accurately.

V. CONCLUSION

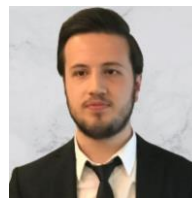
In this paper, we consider one of the most common and significant web vulnerabilities that are used for cross-site scripting attacks. Specifically, we take into account reflected-XSS attacks and related vulnerabilities. We proposed a hybrid model that provides more than 99% accuracy by using web scanning and deep learning methods. The model extracts vulnerabilities in a website and detects potential reflected XSS attacks with deep learning methods. We created a tool, called XSS-Guard, to test our model. The proposed model will help to detect reflected XSS attacks in a more accurate way.

REFERENCES

- [1] "Web Applications vulnerabilities and threats: statistics for 2019." [Online]. Available: <https://www.ptsecurity.com/ww-en/analytcs/web-vulnerabilities-2020/>
- [2] S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," *International Journal of System Assurance Engineering and Management*, vol. 8, no. S1, pp. 512–530, Jan. 2017. [Online]. Available: <http://link.springer.com/10.1007/s13198-015-0376-0>

- [3] "OWASP Top Ten Web Application Security Risks | OWASP." [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [4] V. Nithya, S. L. Pandian, and C. Malarvizhi, "A Survey on Detection and Prevention of Cross-Site Scripting Attack," *International Journal of Security and Its Applications*, vol. 9, no. 3, pp. 139–152, Mar. 2015.
- [5] U. Sarmah, D. Bhattacharyya, and J. Kalita, "A survey of detection methods for XSS attacks," *Journal of Network and Computer Applications*, vol. 118, pp. 113–143, Sep. 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1084804518302042>
- [6] M. Liu, B. Zhang, W. Chen, and X. Zhang, "A Survey of Exploitation and Detection Methods of XSS Vulnerabilities," *IEEE Access*, vol. 7, pp. 182004–182016, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8935148/>
- [7] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Crosssite scripting (XSS) attacks and mitigation: A survey," *Computer Networks*, vol. 166, p. 106960, Jan. 2020. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1389128619311247>
- [8] E. Galan, A. Alcaide, A. Orfila, and J. Blasco, "A multi-agent scanner' to detect stored-xss vulnerabilities," in *2010 International Conference for Internet Technology and Secured Transactions*, 2010, pp. 1–6.
- [9] L. Li and L. Wei, "Automatic XSS Detection and Automatic Anti-Anti-Virus Payload Generation," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. Guilin, China: IEEE, Oct. 2019, pp. 71–76. [Online]. Available: <https://ieeexplore.ieee.org/document/8945988/>
- [10] S. Syaifuddin, D. Risqiwati, and H. A. Sidharta, "Automation Snort Rule for XSS Detection with HoneyPot," in *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. Malang, Indonesia: IEEE, Oct. 2018, pp. 584–588. [Online]. Available: <https://ieeexplore.ieee.org/document/8752961/>
- [11] X.-Y. Hou, X.-L. Zhao, M.-J. Wu, R. Ma, and Y.-P. Chen, "A Dynamic Detection Technique for XSS Vulnerabilities," in *2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC)*. Wuhan, China: IEEE, Apr. 2018, pp. 34–43. [Online]. Available: <https://ieeexplore.ieee.org/document/8842866/>
- [12] G. Habibi and N. Surantha, "XSS Attack Detection with Machine Learning and n-Gram Methods," in *2020 International Conference on Information Management and Technology (ICIMTech)*. Bandung, Indonesia: IEEE, Aug. 2020, pp. 516–520. [Online]. Available: <https://ieeexplore.ieee.org/document/9210946/>
- [13] G. Dong, Y. Zhang, X. Wang, P. Wang, and L. Liu, "Detecting cross site scripting vulnerabilities introduced by HTML5," in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSE)*. Chon Buri: IEEE, May 2014, pp. 319–323. [Online]. Available: <https://ieeexplore.ieee.org/document/6841888/>
- [14] L. Lei, M. Chen, C. He, and D. Li, "XSS Detection Technology Based on LSTM-Attention," in *2020 5th International Conference on Control, Robotics and Cybernetics (CRC)*. Wuhan, China: IEEE, Oct. 2020, pp. 175–180. [Online]. Available: <https://ieeexplore.ieee.org/document/9253484/>
- [15] D. M. W. Powers, "What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes," *arXiv:1503.06410 [cs, stat]*, Sep. 2019, arXiv: 1503.06410. [Online]. Available: <http://arxiv.org/abs/1503.06410>

BIOGRAPHIES



BERAAT BUZ was born in Istanbul, Turkey in 1998. He is a student at Istanbul Technical University, Department of Computer Engineering. He is working on artificial intelligence and computer security.



BERKE GÜLÇİÇEK was born in Samsun, Turkey in 1998. He is a student at Istanbul Technical University, Department of Computer Engineering. His research

interests include computer security and machine learning.



ŞERİF BAHTİYAR is an associate professor in the Department of Computer Engineering at Istanbul Technical University and he is the vice dean in the Faculty of Computer and Informatics at Istanbul Technical University. He received his BS in Control and Computer Engineering and MS in Computer

Engineering degrees both from Istanbul Technical University respectively, and his PhD degree in Computer Engineering from Boğaziçi University. Dr. Bahtiyar was with MasterCard, TUBerlin in Germany, and National Research Institute of Electronics and Cryptology.

Dr. Bahtiyar is the founder and the director of Cyber Security and Privacy Research Laboratory, SPF LAB, at Istanbul Technical University. His current research interests include cyber security and privacy, mobile systems, trust modeling, and financial systems.