Detect Cross-Site Scripting Attacks Using Average Word Embedding and Support Vector Machine

Fawaz Mahiuob Mohammed Mokbal[1,2], Dan Wang[1], and Xiaoxi Wang[3]
(Corresponding author: Dan Wang)
College of Computer Science, Faculty of Information Technology, Beijing University of Technology[1]
Beijing, Beijing 100124, China
Email: wangdan@bjut.edu.cn
Faculty of Computer Science, ILMA University, Karachi 75190, Pakistan[2]
State Grid Management College, State Grid Management College, Beijing, China[3]
(Received Nov. 26, 2020; Revised and Accepted May 6, 2021; First Online Nov. 5, 2021)

Abstract

Web applications are still the preferred target for cybercriminals, as these applications have achieved widespread popularity among individuals and companies. The cross-site scripting attack (XSS) is one of the most severe concerns highlighted at the forefront of information security experts' reports. In this study, we proposed the NLP-SVM method to detect web-based XSS attacks. The method used Natural Language Processing (NLP) for processing text payload attacks and the SVM model for the detection task. The XSS attack payloads were converted into vectors at payload-level instead of word embedding-level. Subsequently, the generated vectors were used for modeling using a support vector machine (SVM) algorithm. The proposed method has successfully surpassed double-check tests include the 10-fold cross-validation approach and the hold-out dataset approach. Extensive analyses of the results determine that the proposed method can efficiently and effectively detect the XSS-based attacks with minimum FN and FP rates. Furthermore, the proposed method had many worthy advantages over the well-known eight algorithms that used the same data. It achieved promising and state-of-the-art results with accuracy, precision, detection probabilities, F-Score, FN rate, FP rate, Misclassification, and AUC-ROC scores of 99.44%, 99.54%, 99.64%, 99.59%, 0.4%, 1.0%, 0.56%, and 99.33%, respectively.

Keywords: Attack Detection; Cross-Site Scripting attack (XSS); Embedding Vectors; Natural Language Processing (NLP); Support Vector Machine (SVM); Web Application Security

1 Introduction

With the development of technology and the Internet's availability everywhere, web applications have occupied a prominent place among the people, and a growing interest by profit and non-profit organizations. Simultaneously, cybercriminals' desire has increased to target these applications to obtain informational or financial gains. According to the Common Vulnerability Scoring System (CVSS), the overall number of new vulnerabilities increased by 17.6% (from 17,308 in 2018 to 20,362 in 2019) [16]. Consequently, this increase represents the keen security concern for various applications, Notably, those carried out in high priority facilities or high-availability operations, such as medical care, banking, e-commerce, *etc.* [13].

One of the most concerned and high-risk cyber-attacks on web applications is Cross-Site Scripting (XSS). The cybercriminals exploit vulnerabilities in the web application to inject their malicious code into the HTML pages, typically in script form executed on users browsers. The cyber-criminals can then extract personal information or steal user cookies to hijack an identity in a fraud session. Consequently, They can steal confidential data or even take hold of those devices [13, 15, 22].

Recently, XSS vulnerabilities continued to grow and ranked as the second most common vulnerability at 14% among 2018 web application vulnerabilities, and they still the second most common vulnerability in 2019. Furthermore, the XSS vulnerabilities were the most common by 44.6% (703) in 2019 of all the 1530 WordPress vulnerabilities [17]. Additionally, XSS enlisted as the most attack vectors, nearly 40% of all attacks used against web applications in 2019, as per PreciseSecurity research [18].

Despite many tools existing to check malicious code existence and the awareness of these vulnerabilities, their

number will not decrease in the future. The immediate impact of the exploit of these vulnerabilities, as well as the lack of preconditions required to exploit them in most cases, could be the main reason for this.

Many researchers in the web security domain have used different techniques such as traditional-based and machine learning-based to mitigate and detect this type of cyber-attack, operating them on the server-side, client-side, or both [27]. However, they all still face some limitations present to

1) Have remarkable missing cases, that is, false negative (FN) rate;

2) Produce an unignorable issue of false positive (FP rate).

The FN issue is crucial since the real threat will pass the system undetected, leading to a complete penetration of the system, losing a lot, such as reputation or financial point of view. In contrast, the FP problematic will cause the legitimate user to be deprived of the service. Further, a heavy burden will add to the security system specialists that will investigate, waste much time, and lose user's confidence and, therefore, companies losing money.

This study proposed the NLP-SVM method to detect web-based XSS attacks to overcome the FN and FP issues, which is highly efficient and has the tendency to defeat such cyber-attacks. The proposed method processes the attacks text payload using Natural Language Processing (NLP). It provides the attack vectors at payload-level instead of word embedding-level to fit the support vector machine algorithm for modelling. More than 20200 samples of XSS text payloads were used, and the method was tested using double-check. that is, our method was tested using 10-fold cross-validation and hold-out testing dataset. Extensive analyses of the results determine that the proposed method able to detect the XSS-based attacks efficiently and effectively with minimum FN and FP. Furthermore, the proposed method had many worthy advantages compared to eight algorithms that used the same data. It achieved promising and state of the art results with accuracy, precision, detection probabilities, F-Score, FN rate, FP rate, Misclassification, and AUC-ROC scores of 99.44%, 99.54%, 99.64%, 99.59%, 0.4%, 1.0%, 0.56%, and 99.33%, respectively.

The rest of this study is organized as follows: Section 2 discusses the most recent related work. Section 3 presents the essential details about this study's methodology and techniques. While Section 4 offers the experimental design and evaluation mechanism strategy, including in-deep details about results, discussion, and comparing the proposed with different eight algorithms. Finally, section 5 concludes this study with its significance and highlights future research directions.

2 Related Work

Many researchers using traditional-based XSS attack mitigation including pattern filtering [28] sanitization-based [2], browser extension [11, 23, 26], proxy-based [8] signatures of script code [21], and Content security policy (CSP) [6]; More details can be find in [20]. These traditional methods are still facing some limitations, especially in the FP rate. The researchers have been moved to a machine learning technique to improve such an attack's detection rate.

In the study presented in [1] the authors used the word2vec tool to determine the occurrence frequency and coincidence of XSS scripts payloads. The occurrence frequency vectors are used for modelling using different machine learning models. However, using the frequency of occurrence and coincidence resulting in large and sparse vectors (mostly 0 values) describes script but not the meaning of the words.

The dynamic feature extraction from content and integrates it with Artificial neural networks (MLP) for the detection task has been proposed in [13]. However, the detection rate of 98.35% needs to improve.

A study proposed by the authors in [4] presents the Browser-based method as defenses against XSS attacks using tokens authentication. Their method hypothesis was based on 2464 cookies gathered from 215 ranked websites; Then, the problem was formulated as a binary classification, and different ML algorithms were used for classification. However, using only tokens authentication is not enough to defeat such attacks. Further, the best detection rate obtained with a random forest model was 83%. In [25], the authors proposed a model for malicious code detection using the stacked denoising autoencoder technique, which resulted in big dimensions, forcing the authors to use sparse random projections for dimensions reduction. However, the FP and the detection rate were 4.2% and 93%, respectively, which is inadequate for detecting malicious attacks.

In the study [19], the authors proposed the XSS attacks classification model for social sites. They applied various algorithms that trained on 100 samples only. However, the dataset is too small and a detection rate score of 0.949.

Wang *et al.* [24] introduced a hybrid analysis method to detect malicious web pages. They used three gropes of features includes URL, HTML, and JavaScript, to classify malicious pages. The result of the detection rate was 88.20%, which is inadequate for detecting malicious attacks.

3 Detection Methodology

3.1 XSS Payload-Level Vector

Since the XSS payloads data are usually in script form, machine learning requires that we first represent the text numerically. The straightforward bag-words model encoding schemes such as frequencies and word counts gen-

Table 1: Examples of payload attacks in the dataset

No	Payload	Class
1	<link rel=import href=data:text/html,<script>alert(1)</script>	XSS
2	&";!–>XSS <=&{()}	XSS
3	<SCRIPT>alert('XSS');</SCRIPT>	XSS
4	<svg xmlns=http://www.w3.org/2000/svg" onload="alert(document.domain)"/>	XSS
5	<script/src=data:,eval(atob(location.hash.slice(1)))//#alert(1)	XSS
6	<a href="/wiki/Social_software" title="Social software"¿Social information systems 	XSS
7	Dreyfus' critique of artificial intelligence :	XSS

erate huge and sparse vectors. Therefore, the word vectors also called the word embeddings technique is introduced to represent each word numerically so that the vector corresponds to how that word is used or what it means. Word embeddings are learned by considering the context. That is, getting the meaning of words from their appearance in context. The word vector approach further improves straightforward bag-words model encoding schemes. The words that appear in similar contexts will have similar vectors. However, using this method resulting in 94-dimensional for each word. Since we do not have word-level labels and only have XSS attack payload-level labels, ML models won't be able to use the word-level embedding. Therefore, the vector representation at payload-level for each attack sample is needed. To deal with this issue, we averaged each word's vectors (word-level) for each attack payload and obtained a single vector at payload-level for each sample. Then, these vectors are used as input for modelling. The NLP is used to pre-processing the XSS attack payload. In particular, we used the current and efficient framework in NLP called spaCy [9]. It is a free and open-source library for advanced (NLP) in Python specifically built for production use and helps create applications that comprehend and process large volumes of text. It can create knowledge extraction, natural language comprehension systems, and pre-process text for ML or Deep learning.

spaCy provides embedding learned from a model called Word2Vec [12] and can calculate the average XSS attack vectors obtaining by doc.vector class. These attack vectors are passed to scikit-learn models. The steps of our method are presented with (Algorithm 1).

Algorithm 1 NLP-SVM steps

1: Begin
2: Initial dataset upload:
3: Initialize the *observations storage.*
4: **for** each XSS text payload in the dataset **do**
5: Tokenized payload into the sequence of token
6: **for** each token **do**
7: Get the vector representation
8: **end for**
9: Calculate the average vectors for each token in the payload
10: Combine all the tokens vectors into a single payload vector
11: Append payload vector with its label
12: **end for**
13: Use payload vectors as input to training ML models
14: Validation ML using the 10-fold cross-validation method
15: Test the fully training model on a hold-out data set
16: End

3.2 Collection XSS Dataset

The XSS payload attacks consist of 3944 samples were collected from PortSwigger Research [17] and Github repository [7]. The benign payloads consist of 6313 samples were gathered from [10]. Table 1 present a few payload samples.

Looking at sample No.1 in Table 1 as an example, when the XSS payload is converted into a word vector, we got a vector for each word, each vector with 96-dominations. That is 27 vectors, each with 96-dimensions, making machine learning model inability to learn. Furthermore, the payloads within the dataset do not have word-level labels and only have the payloads-level labels. Therefore, each payload's vector representation is obtained using word's vectors' average and used the labels at payload-level. Later, we obtained 20257 vectors, each with 96-dimensions for the entire dataset. Table 2 shows the word level vectors and the payload-level vector dimensions.

3.3 Machine Learning Model

Although machine learning has gained a prominent place in cybersecurity, most XSS-based attack detection models still suffer some limitations. Precisely, there are limitations in low detection rate, high false-positive alerts, or high false-negative alerts. Therefore, machine learning models need clean and accurate data to be able to detect attacks efficiently. One of the most popular algorithms is the support vector machine (SVM). SVM is a robust supervised learning algorithm used for classification, re-

Table 2: Payload attack tokens with word-level and payload level vector-dimensions

Payload	**Payload tokens**	**World victors-dimensions**	**Payload vector-dimensions**	**Entire dataset dimensions**
<link rel= import href=data:text/html,<script> alert(1)</script>	<, link, rel= import, href, = data, :, text, /, html, &comma,;, &lt,;, script, &gt,; alert(1),&lt,; &sol,; script, &gt,;	(27, 96)	(1, 96)	(20257, 96)

gression, and outlier detection [22].

Given training vectors $x_i \in \Re^n, i = 1, ..., N$, and their corresponding labels from two classes $y_i \in \{-1, 1\}, i = 1, ..., N$, the classification problem is formulated as Equation (1).

$$y_i(x) \quad = \omega^T \phi(x) + b. \tag{1}$$

Where ϕ is the feature-space transformation function, w is vectors and b is the linear classification bias.

The SVM objective is to find the optimal hyper-plane $w \in R^n$ in N-dimensional space and distinctly classify the data points by maximizing the nearest positive and negative samples' margin. This procedure is expressed in Equation (2):

$$\min_{wb} \max \quad = \tfrac{1}{2} \; subject\, to : y_i(x) \quad = \omega^T \phi(x) + b. \tag{2}$$

However, calculating $\phi(x)$ is very ineffectual and could be impossible because of the introduction of the Lagrange multipliers $\alpha = \{\alpha_i\}, i = 1, ..., N$. Therefore, the former minimization problem is converted into a maximization problem [5]. Further, the feature space can be high-dimensional and may have infinite dimensions. The kernel function is introduced to implicitly define the feature space and efficiently compute very high dimensional spaces. Furthermore, the controller parameters, also known as soft-margin, that allow the violation of the margin constraint are introduced to solve the optimization problem. The kernel function with its parameters is shown in Equation (3). In our experiment, we used the Radial basis function (RBF) defined in Equation (4).

$$\begin{gathered} \max_{\alpha} D_\gamma \; (\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j \kappa_\gamma \; (x_i x_j), \\ subject to := \begin{cases} \sum y_i \alpha_i = 0 \forall i \\ 0 \leq \alpha_i \leq C \forall i \end{cases} \end{gathered} \tag{3}$$

where κ_γ refers to the RBF function kernel is defined as in Equation (4), and C refers to a regularization term that controls the allowed misclassification-level for the training samples. The $\max_\alpha D_\gamma \; (\alpha)$ is the quantity of an upper bound on misclassification probability of kernel κ_γ .

$$\kappa_\gamma \; (x, y) \quad = \exp^{-\gamma^{(x-y)^2}} \tag{4}$$

We tuned γ parameter (gamma) to 0.01 and C parameter to 5 based on practical experiments.

3.4 Experimental and Evaluation

3.5 Dataset Subdivisions

The dataset consists of 20,257 text payload samples, in which 6,313 are benign, and 13,944 are malicious. The dataset has been split randomly and separately into two parts with a ratio of 70%: 30%: for training and testing sets, respectively. The details of the complete dataset are shown in Table 3.

Table 3: Dataset subdivisions

Name	**Benign**	**Malicious**	**Total**
Training dataset	4392	9787	14179
Hold-Out dataset	1921	4157	6078
Total dataset	6313	13944	20257

3.6 Performance Evaluation Metrics

In this research, accuracy, precision, detection rate (DR), Error Rate, false positive, false negative, F-score, and ROC-AUC curves are selected to evaluate the performance of the proposed scheme. These measurements are based on confusion matrix [14]. The TN represents whether the normal case is correctly classified as normal or not. FP or type I error means the normal cases that are incorrectly labeled as an XSS attack. FN or type II error represents an XSS payloads attacks that are incorrectly identified as normal. TP means that an attack payload is correctly identified as an attack. The detailed derivation of the selected performance metrics are shown in the following equations:

$$\begin{aligned} Precision \quad &= \left(\frac{TP}{TP + FP} \right) \\ Detection_{Rate} \quad &= \left(\frac{TP}{TP + FN} \right) \\ TP_{Rate} \quad &= \left(\frac{FP}{TN + FP} \right) \\ FN_{Rate} \quad &= \left(\frac{FN}{TP + FN} \right) \\ F - score \quad &= 2 \left(\frac{Recall \times precision}{Recall + precision} \right) \end{aligned}$$

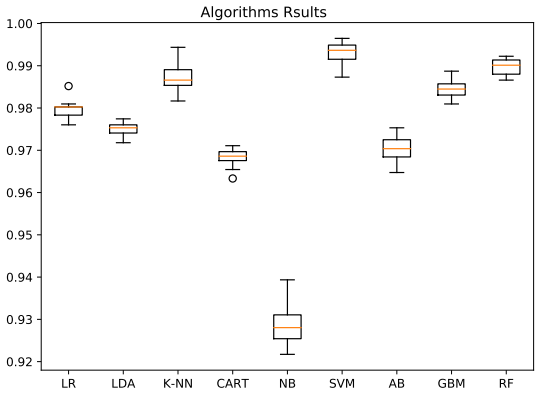


Figure 1: Comparison of algorithms results

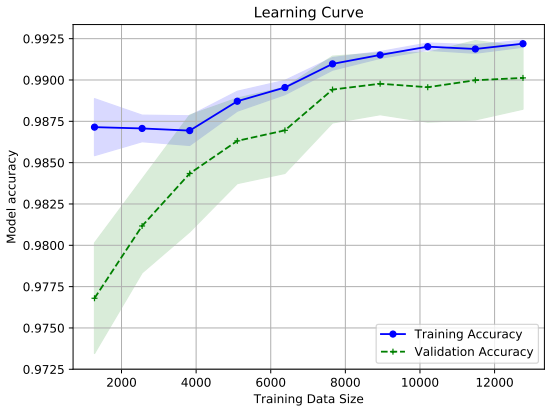


Figure 2: NLP-SVM learning and validation curves with 10-fold

$$Misclassification_{Rate} = \left(\frac{FP+FN}{TP+TN+FP+FN}\right)$$

$$ROC-AUC = \frac{1}{2}\left(\frac{TP}{TP+FN}\right)+\left(\frac{TN}{TN+FP}\right)$$

3.7 Results and Discussion

In this study, SVM alongside eight ML algorithms are implemented, including Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Neighbors (KNN), Decision Tree (CART), GaussianNB (NB), AdaBoost (AB), Gradient Boosting (GB), and Random Forest (RF) [3]. The entire experiments were done on the operating platform LinuxMint-19-tara, 16 GB RAM, Intel Xeon CPU E-5-2620 v3@ 2.40GHz, GPU NIVIDA (Quadro K220). SpaCy version is V2.0, and the Python version is 3.6.7. For strict validation of our model's proposal, we used double-check. All models are trained and tested using 10-fold cross-validation at the first check. Then the Hold-Out test set is used to assess the performance of the final and fully trained models. Overall, the results achieved by all the algorithms were promising. However, the SVM model achieves superior results in both 10-fold cross-validation and the hold-out dataset test. Figure 1 shows that the SVM performance comes in the first position, within an accuracy of 0.9929 and 0.0028 standard deviation. Followed by RF, KNN, GBM, LR, LDA, CART, and NB score of 0.989632, 0.987023, 0.984484, 0.979759, 0.975034, 0.968193, and 0.929121, respectively.

The NLP-SVM model then was tested under 10-fold cross-validation. The SVM parameters of gamma and C were tuning to 0.01 and 5, respectively. The NLP-SVM testing result demonstrated the ability and effectiveness of the model to detect the XSS payloads. Figure 2 shows the NLP-SVM learning and testing curve. The convergence and smoothness between the learning curve and the verification curve indicate that the model learns very well, especially after 4000 sample size.

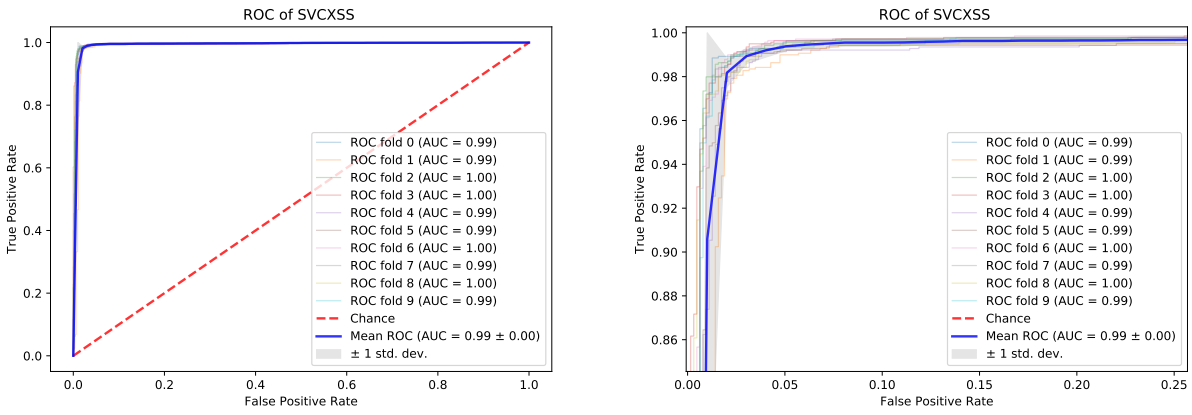
To clarify the NLP-SVM discriminative robustness, we evaluated it with the ROC curve (receiver operating characteristic). The ROC curve is a crucial measure for any classification model's performance that visualizes classification efficiency and provides a full sense of its performance. Verification of NLP-SVM was applied at fold-level, and the mean of 10-fold was calculated. The Figure 3(a) and 3(b) shown the ROC curves performance of NLP-SVM. The mean area under the ROC curve was 0.99 with ?00, proving the model's robustness and ability to detect XSS attacks.

Furthermore, The NLP-SVM model was evaluated by a score and scalability functions. Figure 4 shows the model's results evaluation consisting of (A) Score Vs. Training samples, (B) Scalability Vs. Fit-times, and (C) the performance Vs. Training samples. The results clear that the model has a powerful ability to detect XSS-based attacks. All evaluation results are very consistent and harmonic, which is another strong proof of our proposal outstanding performance.

The second evaluation was done by testing all models on the hold-out dataset. This process was performed after all models were training. The confusion matrix of all models is shown in Table 4. We provided in-depth details with extensive evaluation measurements to compare all the models' results in Table 5. Although the performance of all algorithms was efficient, there are some different crucial points. Compared to SVM, we can see that some models are selective behave. They have higher precision than the detection rate, resulting in an increased FN rate, which means that the real threats will pass through the system undetected. The LR, LDA, CART, NB, AB, and GBM are examples of this category. Another observation is that some models have low precision and low detection rate, resulting in increased FP and FN simultaneously. The NB and CART are examples of this category.

Furthermore, some models have difficulty distinguishing between benign and attack samples where the XSS attack class results are very well, but the benign class results are not. Therefore, they classify many benign samples as attacks, leading to an increasing FP rate. The RF, GBM, AB, NB, CART, LDA, and LR are Fall in this category.

On the other hand, the proposed model is an ideal



(a) (b)

Figure 3: The ROC curves performance of NLP-SVM. (a)SVM model ROC curve for each fold and the mean with a standard deviation, (b) SVM model ROC curve for each fold and the mean with a standard deviation Zoning on the top left.

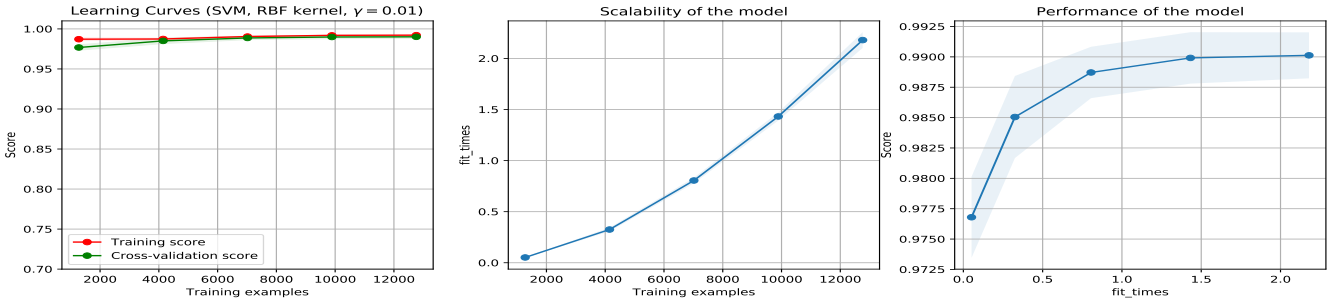


Figure 4: (A) Model learning curve score, (B) show the scalability curve of the model, and (C) show the Model performance curve

model by all measures, which have minimal false negatives and false positives simultaneously. The false negatives score is 0.004 (0.4%), and the false positives score is 0.010 (1.0%), as shown in Figure 5. This ideal result is crucial in such security systems. The FNs are posing real threats to the system. They may lead to a complete penetration of the system, losing a lot, such as reputation or financial point of view. Therefore, it must be taken into consideration very well.

Due to the FP, the legitimate user may deprive of the service. A heavy burden will add to the security system specialists that will investigate, waste much time, and lose users and money's confidence. Therefore, the FP must be reduced to the maximum extent. Other pointers to our proposed model's robustness are the ROC carve and F-score measurements. The area under the ROC curve reaches 99.33%, and the F-score to 99.35%, reflecting the harmony of accuracy and recall and implies the model's power.

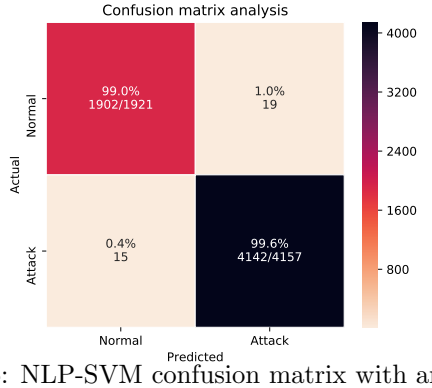


Figure 5: NLP-SVM confusion matrix with analyzing

Table 4: Confusion matrix of all models on hold-out dataset

Model	TP	FN	TN	FP	Total Samples
LR	1841	48	4109	80	6078
LDA	1840	67	4090	81	6078
K-NN	1903	60	4097	18	6078
CART	1816	95	4062	105	6078
NB	1616	136	4021	305	6078
AB	1820	74	4083	101	6078
GBM	1855	37	4120	66	6078
RF	1881	24	4133	40	6078
SVM	1902	15	4142	19	6078

3.8 Comparison with Previous Works

This section compares our work with the reported results of three previous related works in [1, 19, 24], as shown in Table 6. The studies we compare our work with were discussed in more detail in the related work section.

The comparison results demonstrate our proposed work performance is better than others in most measurement. Notably, NLP-SVM's detection rate is the highest, which is critical in such attacks detection systems. Furthermore, we assessed our proposed work on various performance metrics not mentioned in the relevant works.

4 Conclusions

This study presents NLP-SVM model using average word embedding method to detected web-based XSS attacks. Our proposal is used NLP for processing text payloads attacks and the SVM model for the detection task. The detection model has been proved efficient to achieve higher accuracy and a remarkable detection rate with minimal False Positive and Negative rates. The NLP-SVM model adopted a large dataset for training and testing. Numerous analyses have been performed to test the proposed model at various stages. The experimental results confirm the efficiency and idealism of the NLP-SVM method with significant perfection and harmony of performance on multiple measurements of both classes, compared to eight ML algorithms. Moreover, the proposed model outperforms all models in all aspects.

Although our method has proven highly efficient in detecting such attacks, the attack payloads should be extracted from the content to detect the attack automatically. This task will be our future work, as we plan to propose a mechanism for the automatic extraction the attack payload from the content and integrated with NLP-SVM model.

References

[1] S. Akaishi and R. Uda, "Classification of xss attacks by machine learning with frequency of appearance and co-occurrence," in *The 53rd Annual Conference on Information Sciences and Systems (CISS'19)*, pp. 1–6, 2019.

[2] P. Biswajit, G. Tyler, and M. Priyanka, "Handling cross site scripting attacks using cache check to reduce webpage rendering time with elimination of sanitization and filtering in light weight mobile web browser," in *First Conference on Mobile and Secure Services (MOBISECSERV'15)*, pp. 1–7, 2015.

[3] G. Bonaccorso, *Machine Learning Algorithms: Popular Algorithms for Data Science and Machine Learning*, 2018. ISBN: 1789347998.

[4] S. Calzavara, G. Tolomei, A. Casini, M. Bugliesi, and S. Orlando, "A supervised learning approach to protect client authentication on the web," *ACM Transactions on the Web (TWEB'15)*, vol. 9, no. 3, pp. 1–30, 2015.

[5] M. U. Diwekar, "Introduction to applied optimization," *Springer Optimization and Its Applications*, vol. 22, 2020.

[6] I. Dolnák, "Content security policy (CSP) as countermeasure to cross site scripting (XSS) attacks," in *The 15th International Conference on Emerging eLearning Technologies and Applications (ICETA'17)*, pp. 1–4, 2017.

[7] GitHub.com. *Cross Site Scripting (XSS) Vulnerability Payload List*, 2020. (`https://github.com/payloadbox/xss-payload-list`)

[8] S. Goswami, N. Hoque, D. K. Bhattacharyya, and J. Kalita, "An unsupervised method for detection of xss attack," *International Journal of Network Security*, vol. 19, no. 5, pp. 761–775, 2017.

[9] M. Honnibal I. and Montani, "SpaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To Appear*, vol. 7, no. 1, pp. 411–420, 2017.

[10] Kaggle.com. *Cross Site Scripting XSS Dataset for Deep Learning*, 2020. (`https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning`)

[11] B. Mewara, S. Bairwa, J. Gajrani, and V. Jain, "Enhanced browser defense for reflected cross-site scripting," in *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, pp. 1–6, 2014.

[12] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," *Computation and Language*, 2017. arXiv:1712.09405.

[13] F. M. M. Mokbal, D. Wang, I. Azhar, J. Lin, F. Akhtar, and X. Wang, "MLPXSS: An integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique," *IEEE Access*, vol. 7, no. 1, pp. 100567–100580, 2019.

[14] F. M. M. Mokbal, D. Wang, X. Wang, and L. Fu, "Data augmentation-based conditional wasserstein generative adversarial network-gradient penalty for

Table 5: Experiments results analysis for all models on a hold-out dataset

Model	Class	P	DR	F-S	AC	FN	FP	ER	ROC
LR	XSS	0.9809	0.9885	0.9847	0.9789	0.012	0.042	0.0211	9734
	Non-XSS	0.9746	0.9584	0.9664					
	macro avg	0.9777	0.9734	0.9755	(97.89%)	(1.2%)	(4.2%)	(2.11%)	(97.34%)
LDA	XSS	0.9806	0.9839	0.9822	0.9756	0.016	0.042	0.0244	0.9709
	Non-XSS	0.9649	0.9578	0.9613					
	macro avg	0.9727	0.9709	0.9718	(97.56%)	(1.6%)	(4.2%)	(2.44%)	(97.09%)
K-NN	XSS	0.9956	0.9856	0.9906	0.9871	0.014	0.009	0.0192	0.9881
	Non-XSS	0.9694	0.9906	0.9799					
	macro avg	0.9825	0.9881	0.9852	(98.71%)	(1.4%)	(0.9%)	(1.29%)	(98.81%)
CART	XSS	0.9748	0.9771	0.9760	0.9664	0.023	0.055	0.0336	0.9612
	Non-XSS	0.9503	0.9453	0.9478					
	macro avg	0.9625	0.9612	0.9619	(96.64%)	(2.3%)	(5.5%)	(3.36%)	(96.12%)
NB	XSS	0.9295	0.9673	0.9480	0.9274	0.033	0.159	0.0726	0.9043
	Non-XSS	0.9224	0.8412	0.8799					
	macro avg	0.9259	0.9043	0.9140	(92.74%)	(3.3%)	(15.9%)	(7.26%)	(90.43%)
AB	XSS	0.9759	0.9822	0.9790	0.9712	0.018	0.053	0.0288	0.9648
	Non-XSS	0.9609	0.9474	0.9541					
	macro avg	0.9684	0.9648	0.9666	(97.71%)	(1.8%)	(5.3%)	(2.88%)	(96.48%)
GBM	XSS	0.9842	0.9911	0.9877	0.9830	0.009	0.034	0.017	0.9784
	Non-XSS	0.9804	0.9656	0.9730					
	macro avg	0.9823	0.9784	0.9803	(98.30%)	(0.9%)	(3.4%)	(1.7%)	(97.84%)
RF	XSS	0.9904	0.9942	0.9923	0.9895	0.006	0.021	0.0105	0.9867
	Non-XSS	0.9874	0.9792	0.9833					
	macro avg	0.9889	0.9867	0.9878	(98.95%)	(0.6%)	(2.1%)	(1.05%)	(98.67%)
SVM	XSS	0.9954	0.9964	0.9959	0.9944	0.004	0.010	0.0056	0.9933
	Non-XSS	0.9922	0.9901	0.9911					
	macro avg	0.9938	0.9933	0.9935	(99.44%)	(0.4%)	(1.0%)	(0.56%)	(99.33%)

P= Precision, DR= Detection Rate, F-S= F-score, AC= Accuracy overall, FN= FN Rate, FP= FP Rate, ER= Misclassification rate (Error Rate), ROC= AUC-ROC

Table 6: Comparison with previous proposed works

Model	Accuracy	Precision	Detection Rate	F-score	FP	FN	ROC
Decision Tree [24]	-	0.9520	0.882	0.916	-	-	0.9479
CNN+SVM [1]	0.9937	0.9978	0.9886	0.9936	-	-	-
Random Forest [19]	0.972	0.977	0.971	0.974	0.087	-	-
NLP-SVM(This Work)	**0.9944**	**0.9954**	**0.9964**	**0.9959**	**0.010**	**0.004**	**0.9933**

xss attack detection system," *PeerJ Computer Science*, vol. 6, p. e328, 2020.

[15] E. Mugaboand and Q. Y. Zhang, "Intrusion detection method based on support vector machine and information gain for mobile cloud computing," *International Journal of Network Security*, vol. 22, no. 2, pp. 231–241, 2020.

[16] NVD.nist.gov. *NVD - vulnerability metrics*, 2020. (`https://nvd.nist.gov/vuln-metrics/cvss`)

[17] PortSwigger research. *Cross Site Scripting (XSS) Research*, 2020. (`https://nvd.nist.gov/vuln-metrics/cvss`)

[18] PreciseSecurity.com. *Cross-site scripting (XSS) makes nearly 40% of all cyber attacks in 2019*, 2020. (`https://www.precisesecurity.com/articles/cross-site-scripting-xss-makes-nearly-40-of-all-cyber-attacks-in-2019/`)

[19] S. Rathore, P. K. Sharma, and J. H. Park, "XSSclassifier: An efficient XSS attack detection approach based on machine learning classifier on SNSS," *Journal of Information Processing Systems*, vol. 13, no. 4, 2017.

[20] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," *Computer Networks*, vol. 166, pp. 106960, 2020.

[21] H. Shahriar and H. M. Haddad, "Client-side detection of clickjacking attacks," *International Journal of Information Security and Privacy (IJISP'15)*, vol. 9, no. 1, pp. 1–25, 2015.

[22] S. Shan, "Support vector machine," in *Machine Learning Models and Algorithms for Big Data Classification*, pp. 207–235, 2016.

[23] A. P. Sivanesan, A. Mathur, and A. Y. Javaid, "A google chromium browser extension for detecting XSS attack in HTML5 based websites," in *IEEE International Conference on Electro/Information Technology (EIT'18)*, pp. 0302–0304, 2018.

[24] R. Wang, Y. Zhu, J. Tan, and B. Zhou, "Detection of malicious web pages based on hybrid analysis," *Journal of Information Security and Applications*, vol. 35, pp. 68–74, 2017.

[25] Y. Wang, W. D. Cai, and P. C. Wei, "A deep learning approach for detecting malicious javascript code," *Security and Communication Networks*, vol. 9, no. 11, pp. 1520–1534, 2016.

[26] C. Wang and Y. Zhou, "A new cross-site scripting detection mechanism integrated with HTML5 and cors properties by using browser extensions," in *International Computer Symposium (ICS'16)*, pp. 264–269, 2016.

[27] H. Yulianton, H. Warnars, B. Soewito, F. L. Gaol, and E. Abdurachman, "Web security and vulnerability: A literature review," in *Journal of Physics: Conference Series*, vol. 1477, pp. 022028, 2020.

[28] I. Yusof, A. S. K. Pathan, "Preventing persistent cross-site scripting (XSS) attack by applying pattern filtering approach," in *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M'14)*, pp. 1–6, 2014.

Biography

Fawaz Mokbal received his BS degree in Computer Science from Thamar University, Yemen, and MS degree in Information Technology from the University of Agriculture, Pakistan. He is currently a PhD researcher in Computer Science and Technology with Beijing University of Technology, China. He also an Assistant Professor with the Faculty of Computer Science at ILMA University, Pakistan. He has served as head of the Technical Team of Information Center Project for the local Authority for 2 years, and Manager of Information Systems at Ministry of Local Administration 5 years. He is the author and reviewer with various SCI, EI, and Scopus indexed journals. His interest area includes Machine and Deep Learning, Artificial Neural Networks, Medical Images, Web Application Security, and IoT security issues.

Wang Dan received the B.S. degree in computer application, the M.S. degree in computer software and theory, and the Ph.D. degree in computer software and theory from Northeastern University, China, in 1991, 1996, and 2002, respectively. She is currently a Professor with the College of Computer Science, Beijing University of Technology. She is the author and reviewer with various SCI, EI, and Scopus indexed journals. Her major areas of interests include trusted software, web security, and big data.

Wang Xiaoxireceived his MS degree in Computer Technology from Beijing University of Technology. He is currently working as an engineer in State Grid Management College. His major area of interest is Computer Network.