International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Cross Site Scripting Attack Detection Approach Based on LSTM Encoder-Decoder and Word Embeddings

Rokia Lamrani Alaoui* [1], El Habib Nfaoui[2]

Submitted: 09/11/2022 **Accepted:** 10/02/2023

Abstract: Web applications are the main target of Cyber Attacks. Cross-Site Scripting (XSS) is one of the most serious web attacks. Through the use of XSS, cybercriminals are able to turn trusted websites into malicious ones, resulting in extreme harm and damage to both the victims and the reputation of the website owner. According to the Open Web Application Security Project (OWASP) survey, XSS has been ranked in the top 10 web application vulnerabilities since 2017. Though its real danger, only 10 research works studied XSS attacks between 2010 and 2021 as reported recently by a systematic literature review on web attacks detection using Deep Learning. On the other hand, in many Natural Language Processing (NLP) applications, the use of word embeddings and Deep Encoder-Decoder models has considerably improved the performance of downstream NLP tasks. Thereby, in this work, we proposed a Deep Learning approach based on LSTM Encoder-Decoder and free-context word embedding for XSS attacks detection. Then, we implemented the proposed model and compared it with state-of-the-art approaches. The experimental results show that our model achieves good results; 99.08% Accuracy, 99.09% precision, and 99.08% Recall.

Keywords: *Deep learning, Encoder-Decoder, Cross Site Scripting attack, Web security, word embedding, XSS*

1. Introduction

Web applications are daily used in various life domains such as education, healthcare, finance, and entertainment. While they allow for close and easy communication with consumers, they constitute the entry point for attackers to compromise sensitive services and data.

Cross Site scripting attack is one of the most serious threats to web applications. Indeed, based on the OWASP surveys [1], it is ranked in the top 10 web application vulnerabilities since 2017. There exist two types of XSS attacks: i) Client XSS; in which malicious data is used to modify the DOM with unsafe JavaScript calls. The source of this data could be the DOM itself (i.e. DOM XSS attack), or it could have been sent by the server in an HTTP response (i.e. Reflected XSS attack). Finally, the source of data could be from a stored location on the server (i.e. Stored client XSS attack). ii) Server XSS occurs when a user supplied malicious data, and the server includes it in HTTP responses or stores it in a database without a proper validation. In the first case, the attack is named Reflected Server XSS, and in the second case it is a Stored Server XSS.

XSS attacks cause serious damage to the confidentiality and security of web applications. In fact, a successful XSS attack allows web attackers to take over users' accounts, or to install malware on users' devices, or to redirect unsuspecting users to a fake website under the guise of the real one, or even to modify the presentation content of websites.

In order to protect web applications from server XSS attacks, it is recommended to perform user input validation and context sensitive server-side output encoding. As for client XSS attacks, it is recommended to use safe JavaScript APIs. Several tools have been developed to implement these security mechanisms. We cite as examples, static and dynamic analysis, black box fuzzing and rule matching based Web Application Firewalls. Overall, these methods consume time and memory, and are limited due to the complexity of web applications and the abundance of web development libraries and frameworks. Also, some methods tend to generate high rates of false positives, or cannot achieve high code coverage, or cannot detect zero-day attacks.

In this paper, we propose an approach for XSS attacks detection based on Deep Learning. Our main contributions are as follow:

- We used different free context word embeddings to transform HTTP requests to numerical vectors that can be processed by the classification models.

- We implemented different Deep Encoder-Decoder models and evaluated their performance on a public XSS

[1] *LISAC Laboratory. Department of Computer Science. University Sidi Mohamed Ben Abdallah, Faculty of Science Dhar El Mahraz, Fez , Morocco. ORCID ID: 0000-0002-2545-2316*
[2] *LISAC Laboratory. Department of Computer Science . University Sidi Mohamed Ben Abdallah, Faculty of Science Dhar El Mahraz, Fez , Morocco. ORCID ID : 0000-0002-5816-0897*
** Corresponding Author Email: rokia.lamranialaoui@usmba.ac.ma*

attacks dataset.

- We compared the performance of the proposed models using the same experimental parameters, which facilitated the identification of the LSTM based Encoder-Decoder model and word2vec as the best XSS attacks detection approach.

- We compared the proposed approach with the state-of-the-art.

The remainder of this paper is organized as follows. Section. 2 reviews related research works. Section. 3 explains the basic concepts needed to understand our proposed approach. Section. 4 describes the approach proposed to detect XSS attacks. Section. 5 reports the results of the experiments. Conclusion and future work are presented in the last section.

2. Related work

Many research works have been devoted to web applications security. Since 2010, and till 2021, about 63 research papers have studied web attacks detection based on Deep Learning [2]. However, despite the fact that XSS attack is one of the most serious web attacks, only few research works have been interested in XSS attacks detection using Deep Learning techniques. Reference [3] detected web attacks using stacked generalization ensembles for LSTM and word2vec. Reference [4] extracts features using n-gram and stacked auto-encoder and classifies web attacks, including XSS, by using isolation forest algorithm. Reference [5] used a GRU based encoder–decoder model with an attention mechanism for detecting XSS, SQL injection (SQLI) and BruteForce attacks. In [6], the authors implemented a stacked denoising auto-encoder to detect malicious requests in the execution traces of java web applications. Reference [7] combined CNN and LSTM to detect XSS attacks. Reference [8] used CNN to detect SQLI and XSS attacks. Reference [9] detected XSS, Remote File Inclusion, Directory Traversal (DT), and SQLI using character-level embedding and CNN. Reference [10] implemented a Deep Feed Forward network (DFFN) to detect XSS attacks. Reference [11] proposed an ensemble classification model of CNNs and LSTMs to identify XSS and SQLI attacks. In Reference [12] and Reference [13], the authors proposed a DFFN based model to detect DOM-XSS attacks and XSS flaws in PHP and JavaScript source code respectively. Reference [14] detected XSS, SQLI, and DT using CNN. Finally, [15] proposed an LSTM based XSS detection model, and they provided a dataset specifically constructed for the purpose of XSS detection and made it available on GitHub [16]. We based the evaluation of our approach on the dataset proposed in the research paper [15].

3. Preliminaries

In this section, we briefly explain the key basic concepts needed for a good understanding of our proposed approach.

3.1. Free-context word embedding

There exist two main categories of word embedding: static (free-context) word-embedding and non-static (not free-context) word embedding. In the former, vector representations of words remain constant throughout the model training, while in the latter, they are modified in the same way as the other model's parameters (i.e. weights and bias). While non-static word embedding may incur over-fitting, in static word embedding we rely on the hypothesis that the obtained vector representation is reliable regardless of the classification task in which it is used, which is not always true.

Follows, we give a brief description of the most used word-embeddings:

- Word2vec [17] is a feed forward neural network of two layers that takes a textual input and generates the corresponding numerical vectors. There are two word2vec based models: CBOW and Skip-Gram. The first model is known to have a faster processing and an accurate word to vector generation than the second model.

- Glove [18] was developed by Stanford and stands for Global Vectors for Word Representation. It is an unsupervised learning algorithm for obtaining vector representations for words. It aims to conciliate the word prediction models with the word statistics over a whole corpus. In fact, the model training is performed on aggregated global word-word co-occurrence statistics from a corpus. Pre-trained word vectors with different embedding dimensions (50,100, 200 or 300) are made available for use without a need to retrain the model.

- FastText [19] was developed by Facebook. It has the same goal as Word2vec. It also uses the same Neural Network architectures to predict word vectors. However, unlike word2vec, FastText treats each word as composed of character N-grams. So the vector for a word is made of the sum of the character N-grams vectors. The use of N-grams in FastText resolves the Out-Of-Vocabulary issue encountered in Word2vec or Glove.

3.2. Encoder-Decoder models

The Encoder-Decoder is a neural network architecture which is composed of two sub-neural networks: an encoder and a decoder. The encoder maps the input data to a fixed-length representation, while the decoder exploits the encoded representation to reconstruct or classify the original input. In other words, the encoder is trained with the help of the decoder. If the decoder reconstructs or classifies accurately the original input, it means that the encoder is sufficiently trained. Fig. 1 and Fig. 2 show respectively a Feed Forward Neural Network based auto-encoder and an LSTM based encoder-decoder.

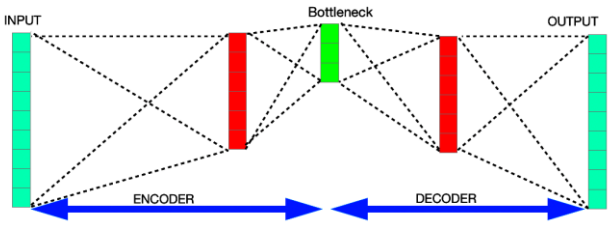

Fig. 1. Feed Forward NN based auto-encoder.

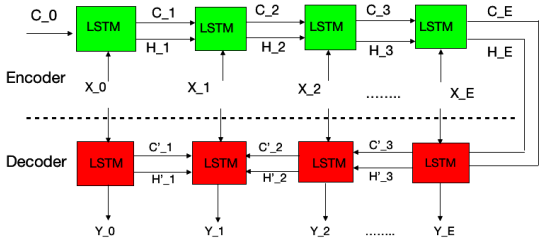

Fig. 2. LSTM based encoder-decoder.

In this work, we implemented LSTM based Encoder-Decoder model and CNN based Encoder-Decoder model. While both share the same theoretical background, there are some practical differences in the process of models' construction. In fact, at the training stage, CNN based encoder-decoders try to minimize the reconstruction error of the original input: we input the vector corresponding to the HTTP web request to the encoder which outputs a representation vector from which the decoder has to reconstruct the input. The model is trained until the difference between the original input and the reconstructed input is acceptable. Afterwards, we suppress the decoder model of the CNN encoder-decoder architecture, and we replace it with any other classification model. In our case, we chose to use a feed forward network (FFN). We trained the FFN while keeping the CNN encoder's weights and bias, learned during the training phase, constant until the FFN can detect malicious HTTP web requests with good accuracy and low false positive rate. Finally, a new set of HTTP web requests; not seen during the training steps, is given to the model, composed of CNN-encoder and FFN, in order to assess its XSS detection performance. Thus, in the case of CNN based Encoder-Decoder, the architecture of the final model is not the same used at the training phases. Also, the model training is done two times and with different objectives. The first time, the model is trained to minimize the reconstruction error of the original input while in the second time the model is partially trained to maximize the detection of XSS attacks. Fig. 3 and Fig. 4 show respectively how the CNN based encoder-decoder model was used to implement our proposed approach detailed in Sec. 4.

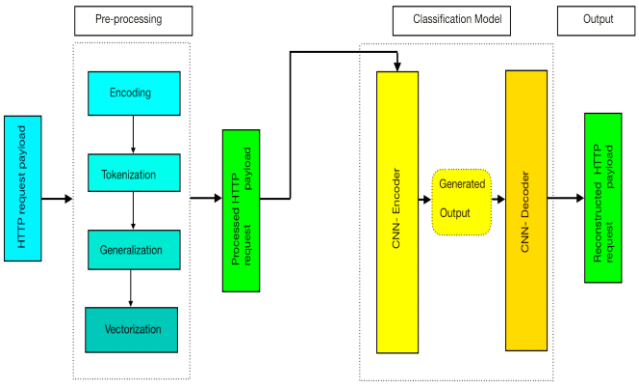

Fig. 3. Flow graph of XSS detection approach using CNN encoder-decoder (training stage).

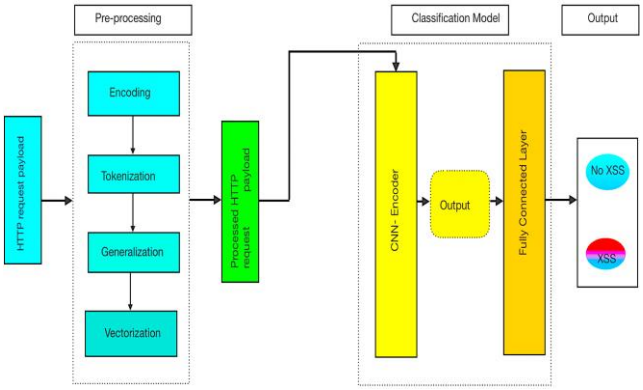

Fig. 4. Flow graph of XSS detection approach using CNN encoder-decoder (testing stage).

On the other hand, in the case of LSTM based Encoder-Decoder,

The model architecture remains the same at both the training and the evaluation phases. Also, the objective of the training is to make the model capable of classifying the HTTP web requests into normal requests and XSS attacks. The concept of a good input reconstruction can be included in the fact that the LSTM-encoder's hidden states are given as initial states to the LSTM-decoder. If the latter reaches a good XSS detection, this implies that the hidden states are a good representation of the input HTTP web request. Same as at the training phase, at the evaluation phase, we give HTTP web requests to both the LSTM encoder and the LSTM-decoder, and we pass the hidden states of the encoder to the decoder. Figure. 5 shows how the LSTM based encoder-decoder model was used to implement our proposed approach explained in detail in Sec. 4.

4. Proposed Approach

4.1. Overview

Figure 5 depicts the main stages of the proposed XSS attack detection approach. Firstly, we transform the HTTP request payload to a vector by executing a sequence of operations that we explain in section 4.2. Then, we feed it to the Deep

Learning classification model, which determines whether an XSS attack is present or not in the HTTP request payload.

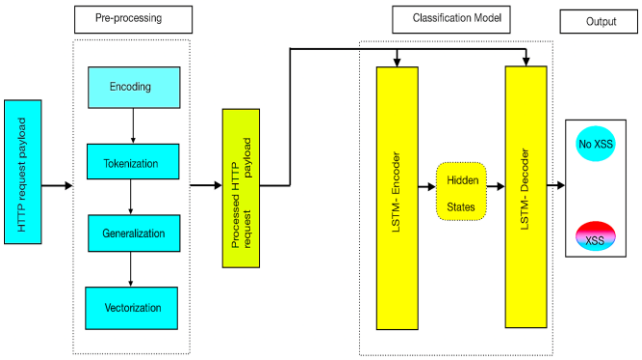

Fig. 5. Flow graph of XSS detection approach using LSTM encoder-decoder (training and testing stages).

4.2. Pre-processing

4.2.1. Decoding

Web attackers encode malicious content using different encoding techniques such as base64, or Hex encoding so that signature based Web Applications Firewalls will not be able to detect that an HTTP Web request hides an XSS attack. Thus, in this step, we decode the HTTP request payloads to obtain the original decoded content.

4.2.2. Tokenization

In this step, we split the HTTP request payload into a set of tokens such that the syntax, keywords or key characters are preserved.

4.2.3. Generalization

We parse each list of tokens; we retain keywords and we replace ordinary words or values with a unique ordinary word.

4.2.4. Vectorization

In the vectorization step, we transform the array of tokens to a numerical vector of size d. If a web request is composed of s words, the resulting embedding matrix has a s × d dimension. d is a model hyperparameter and is called the embedding dimension.

We opted for free-context word embedding to vectorize HTTP requests payloads. Namely, we compared word2vec, FastText, and Glove. And as a baseline, we also implemented one-hot encoding in which we substitute a word in a given sentence with 1 if it exists in the vocabulary, and with 0 otherwise.

4.3. LSTM based Encoder-Decoder model

In this work, we propose to classify HTTP web requests into XSS attacks and normal requests using LSTM based Encoder-Decoder. LSTM Encoder-Decoder is a model where the encoder and the decoder are LSTMs such that the hidden state of the encoder is the initial state of the decoder. The model is ready to use when the encoder and the decoder can accurately communicate information about the input through the hidden states' values, which translates to a minimal loss function and a good classification accuracy. Figure 5 summarizes the proposed detection approach.

5. Experiments and Results

We used Keras, python, genism, FastText to implement the LSTM based Encoder-Decoder model and we run our experiments on Google Collaboratory platform. In the next sections, we detail the experiments and the obtained results.

5.1. Dataset

We used a public dataset dedicated to XSS attacks. It is available on GitHub [14]. It contains 33428 XSS attacks and 31428 normal Web HTTP payloads. We have split the dataset into three balanced sets; the training set, the validation set and the test set used to train the model, tune the model hyper-parameters, and test the model, respectively.

5.2. Performance metrics

We used different performance metrics; accuracy, precision, recall, F1-score, AUC, FPR, and TNR. In the next lines, we give a brief description of each performance metric:

- Accuracy: is the proportion of HTTP web requests classified correctly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Recall: is the proportion of XSS attacks detected by the model among all the XSS attacks contained in the dataset.

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

- Precision: is the proportion of XSS attacks correctly classified by the model among the HTTP web requests detected as XSS attacks.

$$Precision = \frac{TP}{TP + FP}$$

- F1-score: is the harmonic average of precision and recall.

$$F1 - score = 2 . \frac{Precision . Recall}{Precision + Recall}$$

- False Positive Rate: indicates the proportion of normal HTTP web requests misclassified as XSS attacks.

$$FPR = \frac{FP}{FP + TN}$$

- Area Under the Curve (AUC): is the area under the ROC curve, which is the plot between the TPR (y-axis) and the FPR (x-axis). It indicates the model's ability to distinguish

between classes.

- True Negative Rate: indicates the proportion of actual XSS attacks correctly identified by the model.

$$TNR = \frac{TN}{FP + TN} = 1 - FPR$$

5.3. Results

Figure. 6 reports the results of our experiments and compares our work to other research papers that proposed Deep Learning based methods for XSS attack detection in Web Http requests.

Model	F1-score	Accuracy	Precision	Recall	AUC	TNR	FPR
LSTM Encoder-Decoder + W2V	0.9909	0.9908	0.9904	0.9908	0.9910	0.985	0.014
LSTM Enc-Dec + FastText	0.9909	0.9908	0.9909	0.9908	0.99107	0.9805	0.019
LSTM Enc-Dec + Glove	0.9843	0.9843	0.9846	0.9843	0.9846	0.973	0.026
LSTM Enc-Dec + one-hot	0.9898	0.9898	0.9898	0.9898	0.9901	0.9883	0.016
CNN Enc-Dec + W2V	0.9908	0.9908	0.9909	0.9908	0.9910	0.985	0.014
CNN Enc-Dec + FastText	0.9898	0.9898	0.9899	0.9898	0.9899	0.986	0.013
CNN Enc-Dec + Glove	0.9848	0.9848	0.9852	0.9848	0.9852	0.973	0.026
CNN Enc-Dec + one-hot	0.9676	0.9676	0.9676	0.9676	0.9685	0.93	0.062
[6]	0.995	0.993	0.999	0.99	0.95	NA	NA
[14]	0.987	NA	0.995	0.979	0.98	NA	NA
[9]	0.987	0.9932	0.9921	0.98	0.99	NA	0.31

Fig. 6. Experiments results

We developed an LSTM Encoder-Decoder model for XSS attacks detection. Our detection model achieves good results and it is compared favorably with related works ([6],[14], [9]).

Based on the results reported in Fig. 6, we observe that, overall, LSTM Encoder-Decoder achieves the best classification results regardless of the word embedding technique used. However, CNN Encoder-Decoder results are improved slightly when used with Glove. Moreover, we remark that the use of FastText with either CNN or LSTM yields the lowest rate of false positives. Finally, the comparison of our approach with state-of-the-art should be taken with caution. Indeed, although the three research works propose a Deep Learning method for XSS attacks detection, they use different datasets, evaluation metrics and pre-processing techniques.

6. Conclusion

We proposed an XSS attacks detection approach based on LSTM Encoder-Decoder model and free-context word embedding. The classification results of our proposed model are as competitive as state-of-the-art approaches (i.e. 99.08% Accuracy, 99.09% precision, and 99.08% Recall).

As future work, we plan to test our model on private datasets and deploy it in real web applications to evaluate its efficiency.

References

[1] OWASP, "Top 10 Web Application Security Risks," 2017. [Online].

[2] R. L. Alaoui and E. H. Nfaoui, "Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review," Future Internet, vol. 14, no. 4, p. 118, 2022.

[3] R. L. Alaoui and E. H. Nfaoui, "Web attacks detection using stacked generalization ensemble for LSTMs and word embedding," 2022.

[4] A. M. Vartouni, S. S. Kashi and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," 2018.

[5] Z.-Q. Qin, X.-K. Ma and Y.-J. Wang, "Attentional Payload Anomaly Detector for Web Applications," Springer International Publishing, 2018, pp. 588-599.

[6] D. Tripathy, R. Gohil and T. Halabi, "Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning," 2020.

[7] R. Kadhim and M. Gaata, "A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack," Indones. J. Electr. Eng. Comput. Sci, vol. 21, pp. 1022-1029, 2020.

[8] T. Liu, Y. Qi, L. Shi and J. Yan, "Locate-Then-Detect: Real-time Web Attack Detection via Attention-based Deep Neural Networks.," 2019.

[9] W. Rong, B. Zhang and X. Lv, "Malicious web request detection using character-level CNN," 2019.

[10] F. M. M. Mokbal, W. Dan, A. Imran, L. Jiuchuan, F. Akhtar and W. Xiaoxi, "MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique," IEEE Access, vol. 7, pp. 100567-100580, 2019.

[11] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi and Z. Tian, "A novel web attack detection system for internet of things via ensemble classification," IEEE Transactions on Industrial Informatics, vol. 17, no. 8, pp. 5810-5818, 2020.

[12] W. Melicher, C. Fung, L. Bauer and L. Jia, "Towards a lightweight, hybrid approach for detecting dom xss vulnerabilities with machine learning," 2021.

[13] H. Maurel, S. Vidal and T. Rezk, "Statically identifying XSS using deep learning," Science of Computer Programming, vol. 219, p. 102810, 2022.

[14] T. Chen, Y. Chen, M. Lv, G. He, T. Zhu, T. Wang and Z. Weng, "A Payload Based Malicious HTTP Traffic

Detection Method Using Transfer Semi-Supervised Learning," Applied Sciences, vol. 11, no. 16, p. 7188, 2021.

[15] Y. Fang, Y. Li, L. Liu and C. Huang, "DeepXSS: Cross site scripting detection based on deep learning," 2018.

[16] GitHub, "XSS dataset," 2018. [Online].

[17] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[18] J. Pennington, R. Socher and C. D. Manning, "Glove: Global vectors for word representation," Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543, 2014.

[19] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching word vectors with subword information," Transactions of the association for computational linguistics, vol. 5, pp. 135-146, 2017.