DeepXSS: Cross Site Scripting Detection Based on Deep Learning

Yong Fang
College of Cybersecurity
Sichuan University
Chengdu, Sichuan,
P.R.China
yfang@scu.edu.cn

Yang Li
College of Electronics
and Information
Engineering
Sichuan University
Chengdu, Sichuan,
P.R.China
liyang0282019@gmail.com

Liang Liu[*]
College of Cybersecurity
Sichuan University
Chengdu, Sichuan,
P.R.China
liangzhai118@163.com
(Corresponding Author)

Cheng Huang
College of Cybersecurity
Sichuan University
Chengdu, Sichuan,
P.R.China
opcodesec@gmail.com

ABSTRACT

Nowadays, Cross Site Scripting (XSS) is one of the major threats to Web applications. Since it's known to the public, XSS vulnerability has been in the TOP 10 Web application vulnerabilities based on surveys published by the Open Web Applications Security Project (OWASP). How to effectively detect and defend XSS attacks are still one of the most important security issues. In this paper, we present a novel approach to detect XSS attacks based on deep learning (called DeepXSS). First of all, we used word2vec to extract the feature of XSS payloads which captures word order information and map each payload to a feature vector. And then, we trained and tested the detection model using Long Short Term Memory (LSTM) recurrent neural networks. Experimental results show that the proposed XSS detection model based on deep learning achieves a precision rate of 99.5% and a recall rate of 97.9% in real dataset, which means that the novel approach can effectively identify XSS attacks.

CCS Concepts

• Security and privacy → Browser security; Security and privacy → Web application security; Security and privacy → Intrusion detection systems; Security and privacy → Firewalls; Security and privacy → Penetration testing; Security and privacy → Web protocol security

Keywords

Cross Site Scripting (XSS); Word2vec; LSTM; Web Security; Deep Learning

1. INTRODUCTION

Due to the rapid advancement of the Internet and the people's heavy dependence on the Internet and the increasing services of the Web applications, making the network application technology has become more sensitive to cyber-attack. According to the Open Web Application Security Project (OWASP) published the Top 10 lists for the year 2017 [1], Cross Site Scripting (XSS) as the 7th most critical Web application security risks, which means that XSS attacks detection is still a crucial part of cybersecurity. XSS attacks are one of the most common malicious script injection attacks, which occur when an attacker injects a malicious JavaScript payload into a Web page to be executed by user's browser [2].

Malicious scripts executed on the victim's browser may cause hijacking user sessions, misinformation, tampering Web pages, inserting malicious content, phishing attacks, controlling user's browser and accessing business data [3]. More seriously, XSS attacks that combined with other vulnerabilities such as Cross Site Request Forgery (CSRF), Remote Code Execution (RCE), may lead to the host being compromised and threatening the victim's intranet security.

XSS attacks have been traditionally classified into three categories: Persistent XSS attacks, Non-persistent XSS attacks, and DOM-based XSS attacks [4].It is usually easy to implement XSS attacks, but because of the high flexibility encoding schemes and the differences between various browsers, XSS attacks are difficult to prevent by using traditional methods of detection [5].

In this paper, we present a novel XSS attacks detection approach based on deep learning called DeepXSS, and a demo website is accessible at http://www.deepxss.tk/. Our major contributions are as follows:

- We propose a decoder, which can effectively detect obfuscated malicious script attacks by restoring original data from obfuscated data.
- We use word2vec to build the word vector, which can extract the semantic information of XSS attacks payloads.
- We propose a novel XSS attacks detection approach based on LSTM, which achieves a precision rate of 99.5% and a recall rate of 97.9% in real dataset.

The rest of the paper is organized as follows. Related work is presented in section 2, and in section 3 we give a detailed description of deep learning based XSS attacks detection model. In section 4, we conduct the experiments and evaluation results. Finally, in section 5, we summarize our work and discuss further work.

2. RELATED WORK

In recent years, a number of approaches have been proposed to detect the Cross Site Scripting vulnerabilities.

ICCAI 2018, March 12–14, 2018, Chengdu, China
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6419-5/18/03$15.00
https://doi.org/10.1145/3194452.3194469

MK Gupta et al. [6] proposed an approach to detect XSS vulnerable code files in the Web applications through text-mining based predictive models. This method extracts the text features from the source code of Web application using customized tokenizing process, transforms each code file into a set of unique text features with correlated frequency, and builds prediction models based on these characteristics. But, this method is limited to identifying vulnerable code from the source code of the application.

S Goswami et al. [7] proposed an XSS attacks detection method based on unsupervised attribute clustering, using cross-entropy Monte Carlo algorithm as the rank aggregation to rank clusters into two categories: malicious scripts and benign scripts. This approach uses a divergence measure to check the client for XSS vulnerabilities initially. If it exceeds a threshold value, then the request is discarded, on the contrary, it is forwarded to the proxy for further processing. However, this method needs to be detected on the client and proxy servers, lacking in flexibility and interfering with the customer's usage.

BA Vishnu et al. [8] used features based on normal and malicious URLs and JavaScript to predict Cross Site Scripting attacks using three machine learning algorithms (Naive Bayes, SVM, and J48 decision trees). Similarly, S Rathore et al. [9] proposed a method based on machine learning for detecting XSS attacks on SNS websites. This method uses ten different machine learning classification algorithms to classify the dataset into XSS or non-XSS which is based on three features: URLs, webpage, and SNSs. But, the features of these methods based on traditional machine learning are artificial extraction, and there are some limitations.

Instead of using the manual-designed features like above methods, we use deep leaning to extract features that can extract deeper-level features, which can illustrate the representation of data better without artificial participation.

3. PROPOSED APPROACH

In this section, we introduce the overview of deep learning based XSS detection method, describe how to preprocess the input data, and then explain how to use Word2vec to extract XSS payloads into feature vectors, furthermore, describe how LSTM model is designed and learned from sequential data to detect XSS malicious attacks.

3.1 Overview

The proposed approach proceeds as follows. Firstly, we preprocess the input data, which includes decoding, generalization, and tokenization. Then, we extract features by using a deep learning model of word2vec. Finally, we put these word embedding into LSTM to train a classifier for distinguishing XSS samples from benign samples. Figure 1 shows the steps followed by the proposed method of classifying XSS from non-XSS.

3.2 Processing

3.2.1 Decoder

As the XSS Filter Evasion Cheat Sheet [10] states, attackers may be able to evade traditional filters or validation mechanisms which are based on regular expressions by using encoding techniques such as Unicode encoding, URL encoding, HTML entity encoding, Hex encoding, Base64 encoding, UTF-7 encoding, etc. For this reason, we propose a decoder that recursively tries all the decoding possibilities for the input data and converts it into the original form. For example, after the decoder's processing, an obfuscated XSS payload by using HTML entity encoding "<script>&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41;</script>" is restored to "<script>alert(1)</script>".

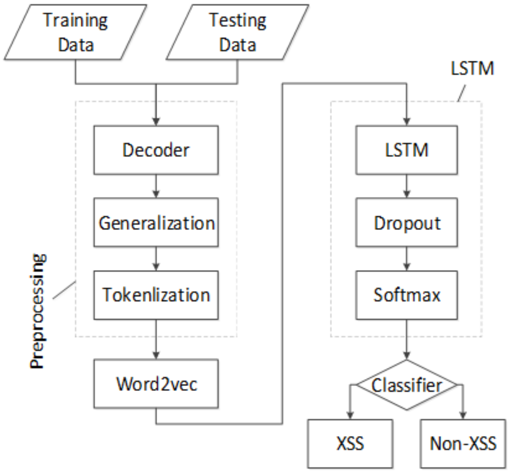

Figure 1. Flow graph of proposed approach.

3.2.2 Generalization

In order to reduce the interference of meaningless or non-helpful information, the decoded data is generalized by follows steps: Firstly, we replace the various URLs in the input data with "http://website". Then, the numbers in the input data will be replaced with "0". And the string as a function parameters will be replaced with "param_string". In addition, other special characters such as blank characters or control characters will be removed.

3.2.3 Tokenization

We design a series of customizing regular expressions based on the features of the scripting language to tokenize the input data. And, the Classification of the tokenization is shown in Table 1.

Table 1. Classification of Tokenization

Classification	List
Start Label	<script>, <frame>, <img, <body>, etc
End Label	</script>, </frame>, </body>, etc
Windows Event	onerror=, onload=, onblur=, oncut=, etc
Function Name	alert(, prompt(, String.fromCharCode(, etc
Script URL	javascript:, vbscript:, etc
Others	>,), #, etc

3.3 Word2vec

Word2vec is a deep learning tool released by Google in 2013, including CBOW (Continuous Bag of Words) model and Skip-Gram model. The CBOW model predicts current word based on the context, while the Skip-Gram model predicts surrounding words for given current word [11]. In terms of training time, the CBOW model is usually several times faster than the Skip-Gram model on the same dataset. For this reason, we use CBOW model to construct word representation in the following experiments. The typical CBOW model architecture is shown in Figure 2.

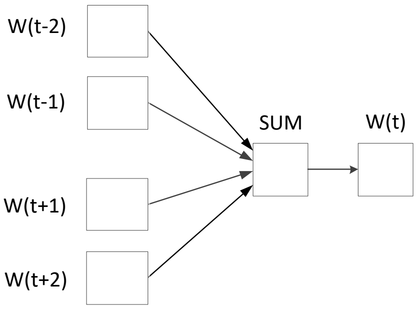

Figure 2. The architecture of CBOW model.

3.4 Deep learning and LSTM

Deep Learning is a branch of algorithmic-based machine learning that attempts to model deep abstractions in data. Moreover, it is an effective tool in predictive analytics because it can model data that contains non-linear characteristics [13]. To overcome the gradient disappearance and gradient explosion problems of RNN algorithm, S Hochreiter et al. [14] proposed the Long Short Term Memory (LSTM) algorithm. Figure 3 describes the basic process of how LSTM works. The typical LSTM network works by using units consisting of a set of gates: the forget gate (f_t), the input gate (i_t), the cell gate (g_t) and the output gate (o_t). LSTM uses these cell gates and associated activation functions, such as: sigmoid activation functions (σ), hyperbolic tangent activation functions (tanh) for proportional select data. The Sigmoid (σ) function is used as an open/close gate, while the tanh function is used as the unit state and output selection.

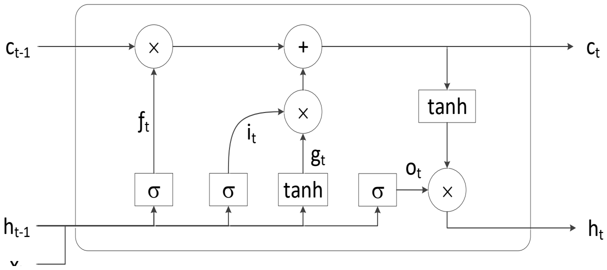

Figure 3. The architecture of LSTM model.

The following equations describe the calculation of the values of these above gates. Where t−1, t are two sequential steps in a sequence. In particular, W_f, W_i, W_g, W_o are weights matrix for forget gate, input gate, cell gate and output gate. Besides, b_f, b_i, b_g, b_o are the corresponding bias.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{3}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4}$$

$$c_t = f_t * c_{t-1} + i_t * g_t \tag{5}$$

$$h_t = o_t * \tanh(c_t) \tag{6}$$

In the experiment, we designed an LSTM neural network detection model that includes an LSTM layer, a Dropout layer, and a Softmax output layer. The LSTM layer is the core layer of this detection model, the Dropout layer is chosen to reduce overfitting problem, and the softmax output layer is designed to predict XSS attacks.

4. EXPERIMENTS AND EVALUATIONS

4.1 Dataset

In the experiment, we used Web crawlers to crawl real malicious and normal samples from the Internet. There are 40,637 malicious samples derived from the XSSed database (http://www.xssed.com/), after removing the sensitive information and duplicate information, a total of 33,426 malicious samples are retained. In addition, there are 31,407 normal samples obtained from the DMOZ database (http://dmoztools.net/).

4.2 Experimental Environment

This experiment was conducted on an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 16G RAM, and NVIDIA GeForce GTX 1060 6GB. Keras is a high-level neural networks API interface developed with Python, which can run on top of TensorFlow, CNTK, and Theano [15]. Because of Keras has the advantages of fast, simple, user-friendliness, modularity, and easy extensibility, we used Keras on top of TensorFlow to construct and evaluate the proposed detection method.

4.3 Evaluation and Result

In the experiment, we evaluated the performance of the proposed approach using 10-fold cross validation. In 10-fold cross validation, the dataset is randomly divided into ten equal size subsets. Nine subsets are retained as training data, and the final subset is used as the validation data for testing the model. The cross-validation process is repeated ten times, and each of the ten subsets is used exactly once as validation data, with the final result is obtained by the average of the result.

In a binary decision problem, the result of each classification may either be positive or negative. As shown in Table 2, the decision can be represented in a structure called as a confusion matrix.

Table 2. Confusion matrix

	Actual XSS	Actual Non-XSS
Predicted XSS	T_P	F_p
Predicted Non-XSS	F_n	T_n

The confusion matrix has four categories: T_P (True Positive) indicates the amount of XSS samples correctly classified, and F_p (False Positive) indicates the amount of Non-XSS samples classified as XSS. Likewise, T_n (True Negative) indicates the amount of Non-XSS samples correctly classified, and F_n (False Negative) indicates the amount of XSS samples classified as Non-XSS. Based on confusion matrix, we evaluate the experimental results with precision, recall, F1 score, and their formulas criteria are shown in following equations.

$$\text{Precision} = \frac{T_p}{T_p + F_p} \tag{7}$$

$$\text{Recall} = \frac{T_p}{T_p + F_n} \tag{8}$$

$$\text{F}_1 = \frac{2\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

In order to objectively evaluate our proposed model, we compared our XSS attacks approach with the method proposed by R Wang [16], which uses traditional machine learning methods (ADTree and AdaBoost algorithms) to detect XSS in OSN; what's more, their datasets also come from the XSSed and the DMOZ. The results of Precision, Recall, and F1 score are shown in Table 3.

Table 3. Evaluating results

Classifier	Precision	Recall	F1
DeepXSS	0.995	0.979	0.987
ADTree	0.938	0.936	0.936
AdaBoost	0.941	0.939	0.939

The table 3 shows that the performance of deep learning model is better than ADTree and AdaBoost algorithms. The precision rate of our proposed method is up to 99.5%, the recall rate is up to 97.9%, and the F1 score is up to 98.7%. To demonstrate the detection performance of our proposed method, a receiver operating characteristic (ROC) curve is plotted as shown in Figure 4. We earned a practically perfect ROC curve with the area under curve (AUC) value of 0.98. And, when the false positive rate close to 0.01, the corresponding True Positive Rate is almost 1.0. This means that our proposed method based on deep learning is very effective in detecting XSS attacks.

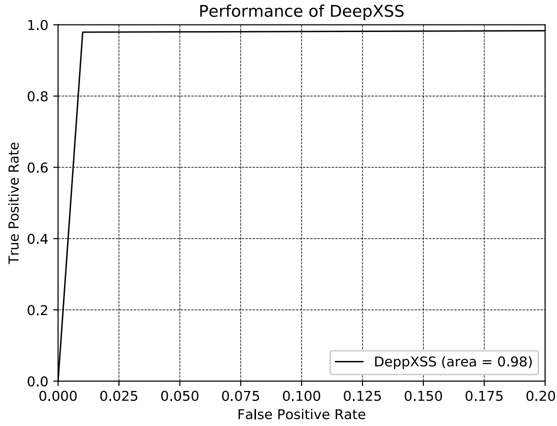

Figure 4. ROC curve.

5. CONCLUSION AND FUTURE WORK

Cross Site Scripting detection is an important factor in the Web applications security. Web applications with XSS vulnerabilities may lead to stealing user information, and potentially serious security issues. In this paper, we proposed a novel detection approach of XSS attacks based on deep learning, called DeepXSS. Firstly, we proposed a decoder to restore original data from the obfuscated data, which is a key part of the process of preprocessing. Secondly, we used word2vec to extract the semantic information of the XSS attacks payloads and mapped them to feature vectors. Finally, we used the Long Short Term Memory (LSTM) recurrent neural networks to build classification models. We evaluated the proposed approach using 10-fold cross validation on the real dataset. And the evaluation results show that our approach achieved outstanding performance with a precision rate of 99.5%, a recall rate of 97.9%, and an F1 score of 98.7%. Moreover, DeepXSS can not only effectively identify traditional XSS attacks, but also effectively identify obfuscated XSS attacks.

In the future, we will collect more XSS attacks datasets to train and optimize our model. In addition, we will try to use other deep learning algorithms to detect XSS attacks.

6. REFERENCES

[1] OWASP, OWASP Top 10 – 2017. Available at https://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf.pdf . Accessed November 3, 2017.

[2] Mohammadi M, Chu B, Lipford H R. Detecting Cross-Site Scripting Vulnerabilities through Automated Unit Testing[C]//Software Quality, Reliability and Security (QRS), 2017 IEEE International Conference on. IEEE, 2017: 364-373.

[3] Nithya V, Pandian S L, Malarvizhi C. A survey on detection and prevention of cross-site scripting attack[J]. International Journal of Security and Its Applications, 2015, 9(3): 139-152.

[4] Gupta S, Gupta B B. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art[J]. International Journal of System Assurance Engineering and Management, 2017, 8(1): 512-530.

[5] Hadpawat T, Vaya D. Analysis of Prevention of XSS Attacks at Client Side[J]. Analysis, 2017, 173(10).

[6] Gupta M K, Govil M C, Singh G. Text-mining based predictive model to detect XSS vulnerable files in Web applications[C]//India Conference (INDICON), 2015 Annual IEEE. IEEE, 2015: 1-6.

[7] Goswami S, Hoque N, Bhattacharyya D K, et al. An Unsupervised Method for Detection of XSS Attack[J]. IJ Network Security, 2017, 19(5): 761-775.

[8] Vishnu B A, Jevitha K P. Prediction of Cross-Site Scripting Attack Using Machine Learning Algorithms[C]//Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing. ACM, 2014: 55.

[9] Rathore S, Sharma P K, Park J H. XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs[J]. Journal of Information Processing Systems, 2017, 13(4).

[10] OWASP, XSS Filter Evasion Cheat Sheet. Available at https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet. Accessed December 12, 2017.

[11] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[12] Joulin A, Mikolov T. Inferring algorithmic patterns with stack-augmented recurrent nets[C]//Advances in neural information processing systems. 2015: 190-198.

[13] Aditham S, Ranganathan N, Katkoori S. LSTM-Based Memory Profiling for Predicting Data Attacks in Distributed Big Data Systems[C]//Parallel and Distributed Processing

Symposium Workshops (IPDPSW), 2017 IEEE International. IEEE, 2017: 1259-1267.

[14] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[15] Keras, Keras: The Python Deep Learning library. Available at https://keras.io/. Accessed January 9, 2018.

[16] Wang R, Jia X, Li Q, et al. Machine Learning Based Cross-Site Scripting Detection in Online Social Network[C]//High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on. IEEE, 2014: 823-8.