Generative Adversarial Network-based Approach for Automated Generation of Adversarial Attacks Against a Deep-Learning based XSS Attack Detection Model

Rokia Lamrani Alaoui, El Habib Nfaoui
LISAC Laboratory, Department of Computer Science
University Sidi Mohamed Ben Abdallah, Faculty of Science Dhar El Mahraz, Fez, Morrocco

***Abstract*—Cross Site Scripting attack (XSS) is one of the most famous and dangerous web attacks. In XSS attacks, illegitimate technical methods are used by attackers to disclose sensitive data from web site users, which result in an important finance and reputation loss to the web site's owner. There exist numerous XSS attack countermeasures. Deep Learning has been shown to be effective when used to detect XSS attacks in HTTP web requests. Yet, Deep Learning models are inherently vulnerable to adversarial attacks, which aim to deceive the detection model into mis-classifying malicious HTTP web requests. Thus, it is important to evaluate the robustness of the detection model against adversarial attacks before its deployment to production in real web applications. In this work, we developed a Generative Adversarial Network (GAN) model for automated generation of adversarial XSS attacks against an LSTM-based XSS attack detection model. We showed that the detection model performance drops drastically when evaluated on the XSS instances, originally used in the model development, but modified by the GAN model. We also provided some guidelines to the development of detection models that can defend against adversarial attacks in the particular context of web attacks detection.**

***Keywords*—Deep learning; generative adversarial network; LSTM; web attacks; adversarial attacks; Cross Site Scripting attack**

I. Introduction

Organizations and enterprises are more concerned now than never before about the mitigation of cyber-attacks. Indeed, a successful cyber-attack can cause the enterprise an important financial loss and a reputation damage. Cross Site Scripting Attack is one of the most serious cyber-attacks that target web sites to compromise the confidentiality of the user's data. We distinguish three types of XSS attacks: Stored XSS attacks, Reflected XSS attacks and DOM(Document Object Model) XSS attacks, which consist of executing a malicious code that was injected into the database server, the web server response, or the DOM, respectively. Different methods are used to protect web applications from XSS attacks. However, XSS attacks are still ranked in the top 10 web vulnerabilities since 2017 [13]. Recently, more research works have been devoted to the use of Deep Learning to build Web Applications Firewalls. The results of these works are overall promising (e.g. [2], [4], [5], [6], [7], [8], [9]). Yet, few works have analyzed whether the proposed model is vulnerable to adversarial attacks. Indeed, Deep Learning models can be victim to adversarial attacks which may result in a significant drop in their classification performance. In the case of XSS attacks detection, adversarial attacks attempt to deceive the model into mis-classifying HTTP web requests, by generating HTTP web requests that resemble to normal HTTP web requests, but are in reality malicious HTTP web requests. In this paper, we proposed a Generative Adversarial Network (GAN) model for an automated generation of adversarial attacks against an LSTM-based XSS attacks detection model. The main contributions of our work is two fold:

- We conducted experiments to demonstrate the negative impact of adversarial attacks on the classification performance of an XSS attacks detection model that returned good results upon its evaluation on a public dataset.
- We proposed some guidelines to how to optimize the detection model to defend against adversarial attacks in the particular context of web attacks detection.

To this end, we followed the steps below:

1) We developed an LSTM model to detect XSS attacks. For that, we used a dataset A, and we recorded the classification results of the model on XSS instances.
2) We developed a Feed Forward Neural Network (FFNN)-based GAN model to generate adversarial XSS examples. For that, we used the dataset A and a dataset B.
3) We passed the XSS instances of the dataset A to the trained GAN model, and we obtained a set of XSS samples that we call carftedXSS.
4) We passed craftedXSS to the trained detection model, and we recorded the difference in the results obtained in this step and in the first step.

The remainder of the paper is divided into seven sections: Section II presents research works related to both XSS attack detection based on Deep Learning and, adversarial attacks against DL-based XSS attack detection models. Section III explains the basic concepts behind the present work. Section IV describes the development process of the XSS detection model and the adversarial attack model. Section V reports and discusses the experimental settings and results. Section VI provide some guidelines to the development of XSS attack detection

models that have a good defense against adversarial attacks. Section VII review the contributions and the limitations of the present work and discusses potential future works.

II. Related Work

In this section, we will review research works related to DL-based XSS attack detection models as well as adversarial attacks models on DL-based XSS attack detection models.

A. XSS Attacks Detection Models

We find few research works about the detection of XSS attacks using Deep Learning models [1]. We cite the following papers [2], [3], [4], [5], [6], [7], [8], [9], and [10], which all use common Deep Learning models like GRU or LSTM based encoder-decoder, CNN, LSTM, DFFN, stacked generalization ensemble model, along with classical vectorization techniques, like word2vec, glove, fasttext, and n-gram, for the conversion of HTTP web requests to numerical vectors. According to the experimental results reported in the cited papers, the models achieve excellent classification results. Yet, the detection of XSS attacks is still challenging, which rises the question of what are the reasons behind the fine or zero utilization of the models proposed by the scientific community in real web sites and applications.

B. Adversarial Attacks Models

Independently of their application domain, Deep Learning models are known to be vulnerable to adversarial attacks. In the following, we present the research papers that have studied adversarial attacks on DL-based XSS attack detection models. [11] used greedy search to find the minimum number of transformations needed for the detection model to mis-classify URLs. They showed that adversarial training can improve the model robustness by 7% while adversarial attacks undermine the model classification performance by 56%. [12] used reinforcement learning to build HTTP web requests that evade an XSS attacks detection model based on a pre-defined set of escaping rules. They also leveraged the adversarial attack model to improve the detection model defense capability. Indeed, after adversarial training, the detection model was able to detect 91.75% of XSS examples and miss out only 8%. [14] used a technique called Soft Actor-Critic (SAC) reinforcement learning algorithm to select the most appropriate strategies to build HTTP web requests that escape the XSS attack detection model. They showed that the adversarial model achieves an escape rate of more than 92%. [15] proposed an adversarial attack model based on a reinforcement learning algorithm (Soft Q-learning) to evade different XSS attack detection models. They experimentally showed that the proposed adversarial model bypassed the detection model in 85% of cases. While all existing research works propose a manual approach for creating adversarial attacks against XSS detection models, the present work use GANs model to automate this process. However, the semantic of generated XSS attacks is guaranteed in the manual approach more than in the automatic approach.

III. Background

A. Problem Definition

Deep Learning based XSS attack detection models can be used as part of Web applications Firewalls to protect web sites and applications from XSS attacks. Yet, because Deep Learning models are victim to adversarial attacks, the XSS attack detection model can yield a high false negative rate as it classifies malicious HTTP requests as normal requests. Before we dive into the approach we propose to demonstrate how adversarial attacks can deceive an efficient XSS attack detection model into mis-classifying XSS attacks as normal HTTP web requests, we will hover, in this section, the basic notions underlying our proposed approach.

B. XSS Attacks

In XSS attacks, the attacker injects malicious scripts in a way that allows him to steal sensitive information from users when they visit a web page or click on a link that includes the malicious code. There are three types of XSS attacks:

- Stored XSS attack: is triggered when the user visits a web page that includes a malicious code that was previously stored in the server database because of the lack of user input validation.
- Reflected XSS attack: occurs when the user clicks on a web link that contains malicious code, and the server sends back this code to the client in an HTTP response.
- DOM XSS attack: it does not involve the server, and it is completely handled by the Document Object Model to attack users.

C. Adversarial Attacks

An adversarial attack is an attack on data with the aim to deceive an already trained model or in training model. It introduces a subtle modification to the data so that a human eye could not notice the difference between the original and the noisy data, but causes the attacked model to output a wrong classification of the input data. Adversarial attacks can be classified into three main categories:

- Evasion: try to evade trained models by altering samples (e.g. HTTP web requests) so that the target model returns the wrong classification.
- Data Poisoning: attempt to contaminate training dataset such that the learning process of the model is undesirably impacted. For instance, after the training phase, the model would learn the wrong features that characterize for example malicious HTTP web requests. As a result, it would classify as normal HTTP web requests what it should be classified as malicious.
- Model extraction: aim at extracting the maximum of information about the model properties in order to rebuild the model and use it for personal use or as part of an adversarial attacks model.

Attackers usually target white-box or black-box threat model when they run an adversarial attack. In white-box model, the adversary knows the model's parameters and can get the classification of input data. In black-box model, the attacker can also acquire labels for input data but he does not know the model's parameters and structures.

D. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are Deep Learning based generative models. Unlike discriminative models, which assign a class label to each input data, generative models objective is to create data that maintains the statistical input data distribution.
GAN was first described by [16] and formalized by [17] in a standardized approach called Deep Convolutional Generative Adversarial Networks (DCGAN). Most GANs that are proposed nowadays are based on DCGANs.
GANs are basically composed of two sub-models (Fig. 1):

- Generator model: generates new samples from the problem domain.
- Discriminator model: classifies the generated samples as fake or real.

The generator and discriminator models are, in general, trained separately. The discriminator is trained until it reaches a high classification performance, and then the generator is trained until the discriminator classifies as real a high percentage of fake samples. After the training process, the discriminator model is discarded and the generator model is kept. In the

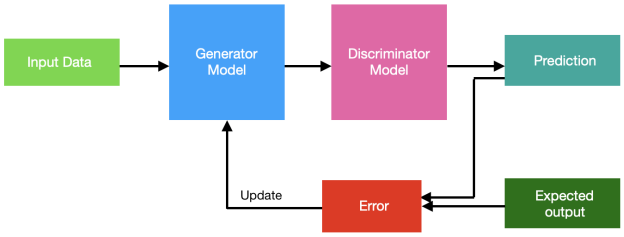

Fig. 1. GAN architecture.

context of adversarial attacks, the GAN model is usually composed of:

- The generator model that can be any Deep Learning based model.
- The discriminator model that is the attacked model.

In this case, the generator model is trained with the purpose of generating adversarial samples that can evade the discriminator model (or the attacked model), which results in a mis-classification of potential XSS attacks.

E. Deep Learning and LSTM Neural Network

Deep Learning is a subfield of machine learning that makes predictions on data based on deep features extraction. Long Short Term Memory or LSTM is a well known neural network especially used in sequence prediction problems. It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies. It was proposed by [18] to resolve the vanishing and exploding gradients problem inherent to the use of RNNs. Fig. 2 describes the basic functioning of LSTMs. They consist of a four types of gates: the forget gate (f_t), the input gate (i_t), the cell gate (g_t) and the output gate

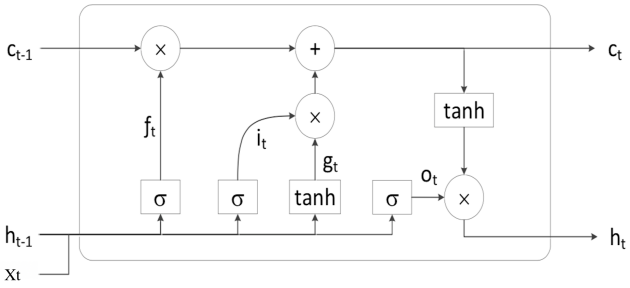

Fig. 2. LSTM architecture.

(o_t). LSTM uses these cell gates and associated activation functions, to decide what to retain and what to forget. The following equations describe the calculations performed at each gate, where W_f,W_i,W_g,W_o are the weight matrices for the forget gate, the input gate, the cell gate, and the output gate, respectively while b_f,b_i,b_g,b_o are the corresponding bias:

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

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

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

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

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t \quad (5)$$

$$h_t = o_t \cdot tanh(c_t) \quad (6)$$

F. Word Embedding: Word2vec

Word embedding is the technique that allows for the application of deep learning models to textual classification problems, as it transforms words to fixed-size numerical vectors. Word2vec is one of the most commonly used word embeddings techniques. CBOW and skip-gram are the main implementations of word2vec. They consist of a Feed Forward Neural Network composed of an input layer, a hidden layer, and an output layer. The CBOW model predicts the contextual words given the main word, while the skip-gram model predicts the target word given the surrounding words.

IV. Proposed Method

A. Overview

Fig. 3 presents an overview of the proposed model architecture. Overall, The GAN model takes as input XSS attacks instances - that the XSS detection model would classify as such if they were not altered by the GAN model-, and then creates a corresponding XSS attack instances, that the detection model could not recognize as such, resulting in a mis-classification of XSS attacks as normal HTTP web requests. In Section V-E, we provide some concrete examples of XSS attacks generated by the GAN model.

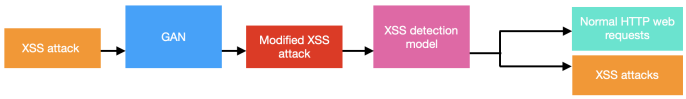

Fig. 3. Overview of the proposed model architecture.

B. XSS Attacks Detection Model

We developed an LSTM neural network to classify HTTP web requests into normal and XSS attacks. It was trained using a dataset A that contains XSS attacks and normal HTTP web requests. We used Word2vec to transform the HTTP web requests into numerical vectors. The choice of LSTM is justified by the fact that the majority of research works about adversarial attacks against Deep Learning based XSS attack detection model, have also selected the LSTM model, which leave a possible space for comparison. Fig. 4 describes the proposed XSS attack detection model.

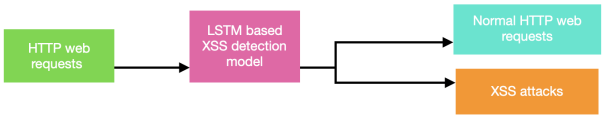

Fig. 4. XSS attacks detection model.

C. Adversarial Attacks Model

Fig. 5 show that the adversarial attacks model is a Feed Forward Neural Network that takes as input HTTP web requests, and then creates a modified version of the same HTTP web requests, which are then passed to the detection model. The adversarial attacks model is constantly updated until the detection model can barely recognize XSS attacks. The adversarial attacks model is developed based on a dataset B that contains XSS attacks from the dataset A and another dataset C that contains XSS instances only.

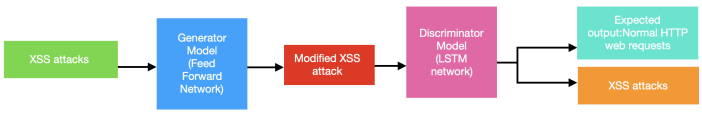

Fig. 5. Adversarial attacks model.

V. Experiments and Discussions

In this section, we present and discuss the settings and results of the experiments conducted in the present work.

TABLE I. XSS Attacks Detection Model Results

Classification results of the LSTM+W2V model	
F1-score	0.98
Accuracy	0.98
Precision	0.98
Recall	0.98
AUC	0.98
FNR	0.002
FPR	0.02

A. Dataset

For the development of the LSTM-based XSS detection model, we used xssed, a public dataset available on GitHub [12], built specifically for XSS attacks. It contains 33428 XSS attacks and 31428 normal HTTP web requests. We splitted the dataset into three balanced sets; the training set, the validation set and the test set, which we used to train the model, tune the model hyper-parameters, and test the model, respectively. As for the adversarial attack model, we collected XSS attacks from the xssed [12] dataset and we added the XSS attacks listed in the online XSS cheat sheet available at [19]. We obtained a dataset of 41542 XSS attacks examples. We splitted the dataset into two sets; the training and the validation sets which are used at the training phase. As for the testing set, it is composed of the XSS attacks contained in the xssed dataset (33427 example), and we used it to evaluate the detection model capability to identify XSS attacks before and after the modification introduced by the GAN model.

B. Performance Indicators

We used traditional performance metrics, namely accuracy, recall, precision, F1-score, AUC, False Positive Rate (FPR), and False Negative Rate (TNR), to evaluate the classification performance of the XSS attack detection model. As regards the adversarial attacks model, we referred to the following indicators to assess its performance:

- DR (detection rate): DR is the ratio of the number of XSS instances that are still classified as XSS examples by the XSS attack detection model to the total examples of XSS attacks.
- ER (escape rate): ER is the ratio of the number of XSS instances classified as normal by the XSS attack detection model after being modified by the adversarial attack model to the total examples of XSS attacks.

C. XSS Attacks Detection Model Results

Table I reports the performance results of the XSS attack detection model. The results indicate that the LSTM-based XSS attacks detection model achieves a high detection accuracy with a low false negative rate (0.002).

D. Adversarial Attacks Model Results

Table II shows that the GAN-based adversarial attack model escape the detection model in 100% of cases (or the detection rate of the detection model is 0%). Because the adversarial attack model uses a large vocabulary, the generated XSS attacks contain a vocabulary that is unknown to the

TABLE II. ADVERSARIAL ATTACK RESULTS

DL Model	Detection rate	Escape rate
LSTM based XSS detection	0	1

detection model as well as the vocabulary that appeared in normal HTTP web requests. Also, the detection model does not account for the semantic of the HTTP web requests. These two factors maximize the escape rate because generated XSS attacks are judged normal HTTP web requests by the detection model. However, this does not mean that the generated XSS attacks if executed will be successful, it will depend on whether they are semantically correct.

E. Examples of Generated XSS Attacks

In this section, we show some examples of XSS attacks generated by the GAN model. Listing 1 shows the original XSS instances, while Listing 2 shows the modified XSS instances. We can clearly see that the generated XSS example widely differ from the original one although it is composed of a vocabulary that is recognized by the detection model. Also, the generated XSS example is not semantically correct as it includes invalid HTML tags. Moreover, the generated XSS example contains some suspicious keywords such as "alert", or "fromcharcode" that appear only in malicious HTTP web requests. Although the human-eye can easily classify the generated instance as malicious, the detection model failed to output the correct classification label.

Listing 1: original XSS instances

```
form.search_text=Dell%22%3E%3Cscript%3
    Ealert(/xss-Bulgari<br/>a/.source)%3C/
script%3E&form.hardware_category=LAPTOP
```

Listing 2: generated XSS instances

```
<section,ic_querytext=,<marque,</q>,ing.
    fromcharcode(,<shadow,),<applet,l(,<p
    <code,querycontext0=,query=,</keygen
    >,<header,video
header>,<noembed,script=,charcode(,section
    ,0aalert(,</tr>,mcharcode(,dd,0t(,<br
```

VI. GUIDELINES

Adversarial attacks constitute a real danger to the reliability of Deep Learning models. In particular, if the Deep Learning model is used to secure a web application, adversarial attacks can increase the false negative rate which results in a low detection rate of XSS attacks. Based on the outcome of this work, we advise the following guidelines to the development of a detection model that has a good defense against adversarial attacks:

- The HTTP web request should be decoded into its original form. Indeed, web attackers obfuscate HTTP web requests by using different encoding techniques.
- The detection model should be trained on a large dataset that include the maximum number of HTML and JavaScript tags.
- The detection model should not restrict its classification decision to the lexical words that compose the HTTP web request, but also include the semantic validity of HTTP web requests.
- The detection model should discard any HTTP web request that exceeds a certain threshold of unknown words.
- It is important to include adversarial learning in the development process of the detection model in order to optimize its defense against adversarial attacks.

VII. CONCLUSION, LIMITATIONS AND FUTURE WORK

In this work, we developed a GAN-based adversarial attacks model to generate adversarial examples that can bypass an LSTM-based XSS attack detection model. The results of our experiments show that the detection model performance drops drastically when comes to the classification of adversarial XSS examples. Indeed, while the detection model classified correctly 32904 out of 33427 XSS attacks, it could not classify correctly any of the corresponding adversarial examples. Moreover, we provided some guidelines to optimize the defense of XSS attack detection models against adversarial attacks. The present work presents the following limitations:

- The adversarial model can not guarantee the semantic validity of the generated adversarial XSS examples.
- Although the proposed adversarial attacks model is applicable to other Deep Learning models, it was applied to only LSTM models.

As future work, we are going to improve the adversarial attacks model in order to generate XSS attacks that are semantically correct.

REFERENCES

[1] 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.

[2] 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. 100 567–100 580, 2019.

[3] R. L. Alaoui *et al.*, "Web attacks detection using stacked generalization ensemble for lstms and word embedding," *Procedia Computer Science*, vol. 215, pp. 687–696, 2022.

[4] A. Vartouni, S. Kashi, and M. Teshnehlab, "An anomaly detection method to detect web attacks using stacked auto-encoder," vol. 2018-January, 2018, pp. 131–134.

[5] Z.-Q. Qin, X.-K. Ma, and Y.-J. Wang, "Attentional payload anomaly detector for web applications," in *Neural Information Processing*. Springer International Publishing, 2018, pp. 588–599. [Online].Available: "https://doi.org/10.1007.

[6] 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.

[7] W. Melicher, C. Fung, L. Bauer, and L. Jia, "Towards a lightweight, hybrid approach for detecting dom xss vulnerabilities with machine learning," in *Proceedings of the Web Conference 2021*, 2021, pp. 2684–2695.

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

[9] Y. Fang, Y. Li, L. Liu, and C. Huang, "Deepxss: Cross site scripting detection based on deep learning," in *Proceedings of the 2018 international conference on computing and artificial intelligence*, 2018, pp. 47–51.

[10] R. L. Alaoui *et al.*, "Cross site scripting attack detection approach based on lstm encoder-decoder and word embeddings," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 2, pp. 277–282, 2023.

[11] B. Rasheed, A. Khan, S. A. Kazmi, R. Hussain, M. J. Piran, and D. Y. Suh, "Adversarial attacks on featureless deep learning malicious urls detection," *Computers, Materials and Continua*, vol. 68, no. 1, pp. 921–939, 2021.

[12] Y. Fang, C. Huang, Y. Xu, and Y. Li, "Rlxss: Optimizing xss detection model to defend against adversarial attacks based on reinforcement learning," *Future Internet*, vol. 11, no. 8, p. 177, 2019.

[13] OWASP, "Top 10 Web Application Security Risks,"https://owasp.org/www-project-top-ten/.

[14] L. Chen, C. Tang, J. He, H. Zhao, X. Lan, and T. Li, "Xss adversarial example attacks based on deep reinforcement learning," *Computers & Security*, vol. 120, p. 102831, 2022.

[15] Q. Wang, H. Yang, G. Wu, K.-K. R. Choo, Z. Zhang, G. Miao, and Y. Ren, "Black-box adversarial attacks on xss attack detection model," *Computers & Security*, vol. 113, p. 102554, 2022.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[18] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," *Advances in neural information processing systems*, vol. 9, 1996.

[19] W. S. Community, "Cross-site scripting (xss) cheat sheet," https://portswigger.net/web-security/cross-site-scripting/cheat-sheet.