

Article

RLXSS: Optimizing XSS Detection Model to Defend Against Adversarial Attacks Based on Reinforcement Learning

Yong Fang ¹, Cheng Huang ^{1,*} , Yijia Xu ¹ and Yang Li ²

¹ College of Cybersecurity, Sichuan University, Chengdu 610065, Sichuan, China

² College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, Sichuan, China

* Correspondence: opcodesec@gmail.com

Received: 18 July 2019; Accepted: 12 August 2019; Published: 14 August 2019

Abstract: With the development of artificial intelligence, machine learning algorithms and deep learning algorithms are widely applied to attack detection models. Adversarial attacks against artificial intelligence models become inevitable problems when there is a lack of research on the cross-site scripting (XSS) attack detection model for defense against attacks. It is extremely important to design a method that can effectively improve the detection model against attack. In this paper, we present a method based on reinforcement learning (called RLXSS), which aims to optimize the XSS detection model to defend against adversarial attacks. First, the adversarial samples of the detection model are mined by the adversarial attack model based on reinforcement learning. Secondly, the detection model and the adversarial model are alternately trained. After each round, the newly-excavated adversarial samples are marked as a malicious sample and are used to retrain the detection model. Experimental results show that the proposed RLXSS model can successfully mine adversarial samples that escape black-box and white-box detection and retain aggressive features. What is more, by alternately training the detection model and the confrontation attack model, the escape rate of the detection model is continuously reduced, which indicates that the model can improve the ability of the detection model to defend against attacks.

Keywords: reinforcement learning; cross-site scripting; adversarial attacks; double deep Q network

1. Introduction

With the increasing popularity of the Internet and the continuous enrichment of web application services, various network security problems have emerged gradually. The endless web attacks have a serious impact on people's daily work and life. Common web attacks include Structured Query Language (SQL) injection, file upload, XSS, Cross Site Request Forgery (CSRF), etc. Web attackers often target sensitive data or direct control of the website. Most web vulnerabilities rely on website functionality, such as SQL injection, which depends on database services, file upload vulnerabilities, which depend on upload services, and so on. In this part, the XSS vulnerability relies on a browser, which can be attacked by XSS as long as you use it. Therefore, the attacks, often being the first step of other advanced attacks, directly threaten user privacy and server security, resulting in information disclosure, command execution, and so on [1,2]. There have already been many research teams that have introduced machine learning and deep learning algorithms into XSS attack detection [3].

With the development of attack detection technology, adversarial attack technologies have emerged for detection models based on AI algorithms. Attackers attempt to attack the detection models by generating confusing and aggressive countermeasure samples, misleading models to classify malicious attack types into benign ones, so as to escape attack detection of the detection models. Generative Adversarial Networks (GAN) add inconspicuous noise to a panda image, and

the image is still visible as a panda by the human eye. However, the GoogleLeNet classification model judges the modified image as a gibbon with 99.3% confidence [4]. The one-pixel attack changes the classification result of deep neural networks in an extreme-limit scenario where only one pixel can be modified [5]. What is more, there are some studies on the adversarial attack of cybersecurity detection models, which aim at malware detection research. Rosenberg et al. [6] proposed a black-box adversarial attack based on the Application Programming Interface (API) for calling machine-based malware classifiers, which was based on generating adversarial sequences of combined API calls and static features, thus misleading classifiers and not affecting the malware functions.

Reinforcement learning has developed rapidly in recent years, and its powerful ability for self-evolution is well known. Wu C et al. [7] proposed Gym-plus, which is a model for generating malware based on reinforcement learning. It retrains the detection model with newly-generated adversarial malware samples to improve its ability to detect unknown malware threats.

Researchers have made some achievements in applying GAN and reinforcement learning to malware detection. However, there are few studies on how to use it in XSS detection to improve the model. It is of great significance to design a method that can effectively improve the defensive ability of the detection model against adversarial attack.

In this paper, we propose an XSS adversarial attack model based on reinforcement learning. By marking the countermeasure samples as XSS malicious samples alternating the training detection model and adversarial attack model, we can continuously enhance the ability of the detection model to defend against attack. Our major contributions are as follows:

- We propose a model of XSS adversarial attack based on reinforcement learning (called RLXSS), which converts the XSS escape attack into the choice of escape strategy and the best escape strategy according to the state of the environment.
- We propose four types of XSS attack escape techniques, including encoding obfuscation, sensitive word replacement, position morphology transformation, and special character adding. RLXSS chooses the best escape strategy according to the environment state to mine the adversarial sample that escapes black-box and white-box detection and retain aggressive features. We have found common XSS escape strategies for SafeDog and XSSchop, which are widely-used real security protection software packages.
- We use RLXSS to mine the adversarial samples, by marking the adversarial samples as malicious samples and retraining the detection model. We alternately train the detection model and adversarial attack model, so as to improve the ability of the detection model to defend against adversarial attacks continuously.

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

2. Related Work

At present, there are many research works on cross-site scripting, which are mainly divided into cross-site scripting attack detection and cross-site scripting vulnerability discovery. These two main research directions have been developed in recent years with many new research results. Many researchers have developed efficient XSS detection models or XSS discoverers. When referring to their research, we also think about how to optimize them.

In terms of cross-site scripting attack detection, Vishnu B A et al. [8] proposed a method of detecting XSS attacks using machine learning algorithms, extracting the characteristics of URLs and JavaScript code and using three machine learning algorithms (naive Bayes, SVM, and J48 decision trees) to detect XSS. Similarly, Rathore S et al. [9] proposed a method for XSS attack detection on a social networking services (SNS) website based on machine learning algorithms. The method extracts

three characteristics of the URL, web page, and SNS website and classifies the dataset into XSS and non-XSS by using 10 different machine learning classification algorithms. What is more, with the development of deep learning, we published the “DeepXSS: Cross Site Scripting Detection Based on Deep Learning” in the 2018 International Conference on Computing and Artificial Intelligence [10], which proposed to extract word vectors with semantic information based on Word2Vec, and Long Short-Term Memory (LSTM) algorithm based deep learning technology was used to extract the deep features of cross-site scripting attacks automatically.

In terms of cross-site scripting vulnerability discovery, the research on XSS vulnerability discovery focuses on how to generate XSS attack vectors. Due to improper data encoding, Mohammadi M [11] proposed a grammar-based attack generator that automatically generates an XSS test case to evaluate cross-site scripting vulnerabilities in the target page. Duchene F et al. [12] proposed a black-box fuzzy tester based on the genetic algorithm to generate malicious input detection XSS automatically, which was named KameleonFuzz. Guo et al. [13] proposed a method for mining XSS vulnerabilities based on the optimized XSS attack vector library, which constructs the XSS attack vector grammar, and built the attack vector pattern library, resource library, and mutation rule library based on the attack vector grammar to generate the XSS attack vector library.

Due to the complex and varied web application environment, it is difficult to exploit fully the vulnerability of XSS vulnerabilities based on attack vector generation and automated testing. More importantly, most of the current research focuses on attack detection or vulnerability discovery, but the research on the security of the XSS detection model itself is lacking. Therefore, how to optimize the XSS detection model’s ability to defend against adversarial attacks through reinforcement learning will be the focus of this paper.

3. Proposed Approach

In this section, we introduce the overview of the method based on reinforcement learning, which optimizes the XSS detection model to defend against adversarial attacks, how to mine the adversarial sample of the black-box and white-box XSS detection model through reinforcement learning, and how to optimize the detection model’s ability to defend against adversarial attacks.

3.1. Overview

RLXSS consists of an adversarial model and a retraining mode. Figure 1 shows the steps followed by RLXSS to mine the adversarial sample and optimize the detection model’s ability to defend against adversarial attacks. It proceeds as follows:

- (a) The adversarial model is designed to mine the adversarial samples that retain the XSS attack function and successfully escape the black- and white-box model detection. Firstly, the training sample data and test sample data are input into the black- and white-box detection environment, and the state information is transmitted to the agent based on DDQN (dueling deep Q networks) according to the sample of the detection model. Secondly, the agent chooses the corresponding escape technology, and the modifier modifies the sample according to the selected action. Then, it will be transferred to the detection environment for detection again. The environment obtains the state of the detection results, and the corresponding reward value is fed back.
- (b) The retaining model is designed to optimize the detection model’s ability to defend against adversarial attacks. The adversarial samples are marked as malicious samples. When the detection model (XSS classifier) is retrained, the adversarial model and detection model are alternately trained to improve the ability of the detection model to defend against attacks continuously.

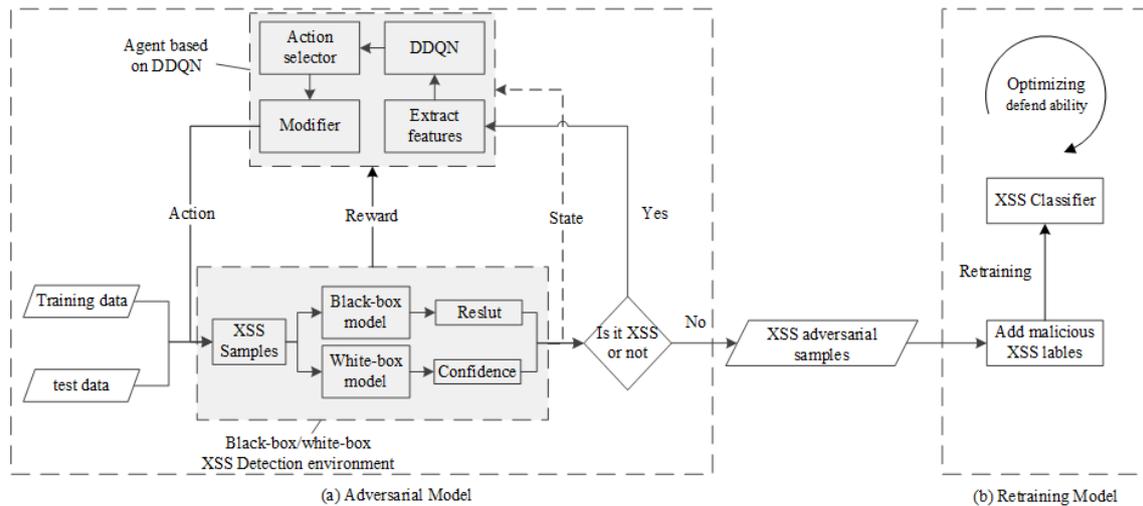


Figure 1. Architecture of reinforcement learning cross-site scripting (RLXSS).

3.2. Mining XSS Adversarial Samples through Reinforcement Learning

3.2.1. Preprocessing

The preprocessing inputs the initial malicious samples into the black-box and the white-box detection model for filtering. For the black-box detection tool, it saves the feedback intercepted samples as a malicious sample set, which is used as the adversarial attack black-box tool. For the white-box detection model, it saves samples with attack detection confidence greater than the threshold as a set of malicious samples, which is used as the adversarial attack white-box model. Finally, it takes the malicious sample block in the two datasets as malicious samples of RLXSS adversarial attack for the black- and white-box XSS detection software.

3.2.2. Black- and White-Box Detection Environment

In order to implement a convenient interface to call different environments, the module encapsulates the black-box detection tool API and the white-box detection model API. The black-box detection tool interface uses the web crawler to carry the detection sample to request the detection web page, and the web page has the black-box detection tool installed to protect against the XSS attack. According to whether the request is blocked, the result is fed back. While the white-box detection model interface pre-processes the detection samples, it will input them in the white-box detection model for detection. Besides, it obtains the confidence of the detection samples classified as XSS attacks and feeds back the results.

The reward function is defined according to the difference between the black- and white-box detection model. The reward of the black-box model mainly depends on whether or not it escapes detection. The reward of the white-box model mainly depends on the extent of confidence reduction. The specific definition of the reward function is shown in formula:

$$r_t = \begin{cases} result * score, & \text{if } is_blackbox = True \\ \frac{1-result}{1-threshold} * score, & \text{if } is_blackbox = False \end{cases} \quad (1)$$

In the formula, the *isblackbox* parameter indicates whether the environment is a black-box detection model; the *result* parameter is the feedback value of the black-box and white-box detection API; the *threshold* parameter is the confidence threshold of the white-box detection model; and the *score* parameter is the reward for a successful escape.

When the modified malicious samples are classified as benign samples and the reward value is not lower than the set threshold, the malicious samples are stored as adversarial samples; otherwise, they will be transmitted to the agent based on DDQN to continue to try the adversarial attacks.

3.2.3. Agent Based on DDQN

In the agent module, the detection samples are processed by word segmentation and vectorization. The word vector and environment state are input into the DDQN algorithm model. According to the output prediction of DDQN, the escape action selector chooses the optimal escape action. Finally, the modifier converts XSS samples based on the selected actions.

A. DDQN

In reinforcement learning, the decision maker or learner is called the Agent. The sum of all other interactions with them is called the Environment, and the environment generated such as game scores is called the Reward. DeepMind's VMnih first proposed the deep Q-networks (DQN) network in 2013. DQN uses end-to-end reinforcement learning to learn strategies from high-dimensional input and achieves comparable results to professional players in the many challenges of the Atari2600 game [14]. DQN is a deep Q network based on the Q-learning algorithm. The main improvements are as follows:

- (1) Limit the reward value and the error term to a limited range and define the Q value and the gradient to be within a valid limited range to improve the stability of the model.
- (2) Adopt the experience replay training mechanism. The training of deep neural network requires that the samples be independent and identically distributed. However, the data acquired by reinforcement learning have a strong correlation. Direct training will lead to the problem of training instability. DQN stores the transferred samples of each step into memory D based on the experience replay mechanism, and each time, a mini-batch transfer sample is randomly extracted from D. The parameters are updated according to the gradient descent and randomized by introducing the experience replay sampling, which weakens the correlation between the data, thereby improving the stability of the model.
- (3) Use a deep neural network to approximate the current value function. In addition, use another network to generate a target value function. Every Round, the target value function is updated with the current value function, and the other rounds keep the target value function unchanged.

There is an overestimation problem in the DQN algorithm, which may lead to overestimation of a non-optimal action, such that the Q value exceeds the Q value of the optimal action. Finally, the model cannot obtain the optimal action. In order to solve the problem of overestimation of the DQN algorithm, Van Hasselt et al. [15] proposed the double DQN algorithm (DDQN), which decomposes the target value function into the action selection value function and action evaluation value function. It can solve the problem of overestimation by decoupling action selection and action evaluation.

The neural network part of DQN can be seen as a new neural network with an old neural network. They have the same structure, but their internal parameter updates are delayed. The idea of DDQN is to use another neural network to eliminate some effects of maximum error. Therefore, DDQN uses the Q-estimated neural network to estimate the maximum action value of $Q_{\max}(s', a')$ in Q reality. Then, we use this action estimated by Q to select $Q(s')$ in Q reality. The basic operating principle of DDQN is shown in Figure 2.

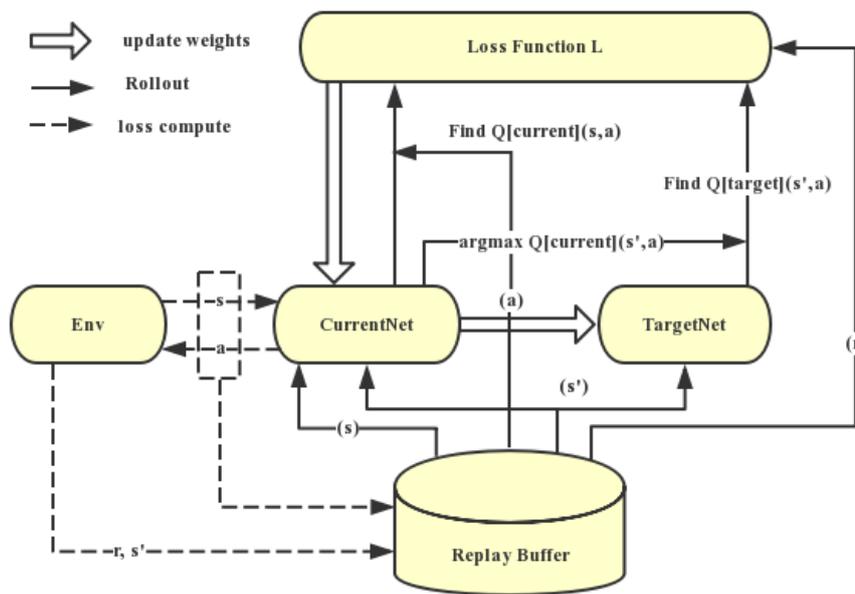


Figure 2. DDQN technical principle framework.

The time and space complexity of RLXSS depends mainly on the DDQN algorithm:

$$Y_t^{DoubleDDQN} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}} Q(S_{t+1}, a; \theta_t) \theta_t^-) \tag{2}$$

In the algorithm, the parameter t represents the current step of the algorithm; the parameter a represents the action selected by the algorithm; S_{t+1} represents the new state of DDQN; θ represents the weight in the new neural network. We can see from the algorithm that DDQN is the time complexity of recursion, and its spatial complexity is twice that of the neural network.

B. XSS escape technology and action space

Figure 3 shows a schematic diagram of the structure of common XSS attack vectors, including tags, attribute expressions, event expressions, content, and other components.

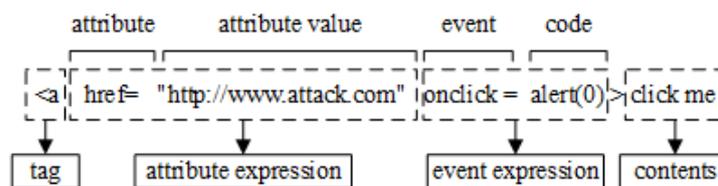


Figure 3. XSS attack vector structure diagram.

According to the characteristics of XSS attack, we present four types of XSS attack escape technologies proposed in this paper, including encoding obfuscation, sensitive word substitution, position morphology transformation, and adding special characters. The attack vectors are generated based on the escape strategy, which is used for adversarial attack the detection tools or models to mine adversarial samples that escape detection and retain aggressive features. On the basis of four types of escape techniques, the escape action space of the XSS attack is defined as follows:

- Add an assignment expression before the event
- Replace the alert function with another function
- Use the top function to modify the alert function

- Add a random string at the end of the label
- Add a random string at the end of the sample
- General the event replacement tag for any character
- Add a comment between the function and the parameter
- Add a calculation expression before the function
- Replace the brackets
- Replace the space
- Replace the event
- Add blank characters after the event
- Randomly convert the uppercase and lowercase letters of the label and event character words
- Attribute expression and event expression positional transformation
- Unicode encoding
- HTML entity encoding

Regular XSS statements are often killed by detection software, but these action spaces provide escape methods for XSS attack. These action spaces will be used by the Agent to mutate the XSS code, which can produce a valid XSS payload.

3.2.4. Mining XSS Adversarial Samples

The XSS adversarial attack model is constructed based on the DDQN reinforcement learning algorithm. The malicious samples are input into the black-box or white-box XSS detection tool, and the corresponding detection results are obtained. When the inspection result and the current sample are input into the Agent, the adversarial model selects the optimal escape action in the action space based on the environmental state. After transforming the malicious samples, they are re-input into the black-box or white-box XSS detection tool for detection. Feedback rewards are given according to whether the escaping results are successful or exceed the maximum number of attempts. Otherwise, it continues to find the optimal escape strategy. What is more, the samples of successful escape detection are saved as XSS adversarial samples.

3.3. Optimizing the Ability of XSS Detection Models to Defend against Adversarial Attacks through Retraining

Based on the DDQN reinforcement learning algorithm, we built the adversarial attack of the black-box and white-box detection model. We mined the adversarial attack samples with successful escape detection and retaining attack function, thus verifying the effectiveness of the attack model. However, the purpose of our research is to optimize the detection model rather than implement the attack. Therefore, this paper proposes a retraining model, marking the adversarial sample as a malicious sample and then retraining the detection model. By continuously training the adversarial model and the detection model, the ability of the detection model to defend against adversarial attack is improved. In summary, we try to improve the DeepXSS model's ability to defend against adversarial attacks through the RLXSS model.

4. Experiments and Evaluations

4.1. Dataset

The XSSed project was created in early February 2007. It provides information on all things related to cross-site scripting vulnerabilities and is the largest online archive of XSS vulnerable websites. In the experiment, we used 33,426 samples from the XSSed database (<http://www.xssed.com/>) as XSS malicious samples and as the initial dataset for adversarial attacks. The data we used are real and valid attack samples collected by www.xssed.com over the past 10 years. Therefore, we believe that we can generate samples more suitable for the real environment by training on this dataset.

4.2. Experimental Environment

RLXSS was programmed based on Python3, Keras-rl, and OpanAIGym. Keras-rl [16] is an open-source toolkit for deep reinforcement learning based on Keras. OpanAIGym [17] is an open-source toolkit for developing and comparing reinforcement learning algorithms, which is usually used to provide an environment for reinforcement learning. We chose two common website security software packages, SafeDog [18] (Apache Version V4.0) and XSSChop [19] (version: b6d98f6; update date: 35 January 2019) as the black-box model for the adversarial attacks' target. We chose DeepXSS as the target of the white-box model for the adversarial attack and as the final target model of optimizing defense capability. The detail experiment environment is listed in Table 1.

Table 1. Experiment environment.

System	Ubuntu 16.04.4 LTS
RAM	16G
CPU	i7-7700 CPU @ 3.60 GHz
GPU	NVIDIA GeForce GTX 1060 6 GB
Versions of Python and Extension Library	Python 3.6.7 keras-rl==0.4.2 gym==0.9.5 requests==2.18.4 tensorflow==1.13.1 keras==2.0.9 gensim==3.2.0 h5py==2.9.0 sklearn==0.20.3

4.3. Evaluation Method

4.3.1. Evaluation Criteria

In order to evaluate the experiment objectively, the DR (detection rate) and ER (escape rate) were used. Their definitions are shown in the formulas below.

$$DR = \frac{\text{Number of malicious samples detected}}{\text{Total number of malicious samples}} \quad (3)$$

$$ER = \frac{\text{Number of escaping Success}}{\text{Total number of malicious samples}} = 1 - DR \quad (4)$$

The ER (escape rate) reflects the proportion of the target detection model or tool classifying malicious samples into benign samples after escape transformation. The higher the proportion, the more vectors represent escape, which indicates that the defection of the model of defense adversarial attack is greater. The DR (detection rate), which reflects the escape detection model or tool, can still detect the proportion of malicious attack samples. The higher the detection rate, the stronger the ability of the model or tool to defend against attacks. The DR reflects the proportion of malicious attack samples after escape confusion, and the detection model or tool can still detect malicious attack samples. For example, the higher the detection rate, the stronger the ability of the model or tool to defend against the adversarial attack.

4.3.2. Evaluation Model

In order to test the detection rate and escape rate of the adversarial model, we not only used the most popular XSS detection software (SafeDog and XSSChop), but also trained the LSTM model for evaluation. To get a better LSTM model, the paper tuned the Size, Iter, Window, and Negative parameters in Word2Vec. Through the control variable method, only one parameter was modified at a time, and the effects of different parameters on the recall rate, accuracy, accuracy, and F1 value of the LSTM detection model were compared. The experimental results of the Word2Vec parameter tuning are shown in Figure 4.

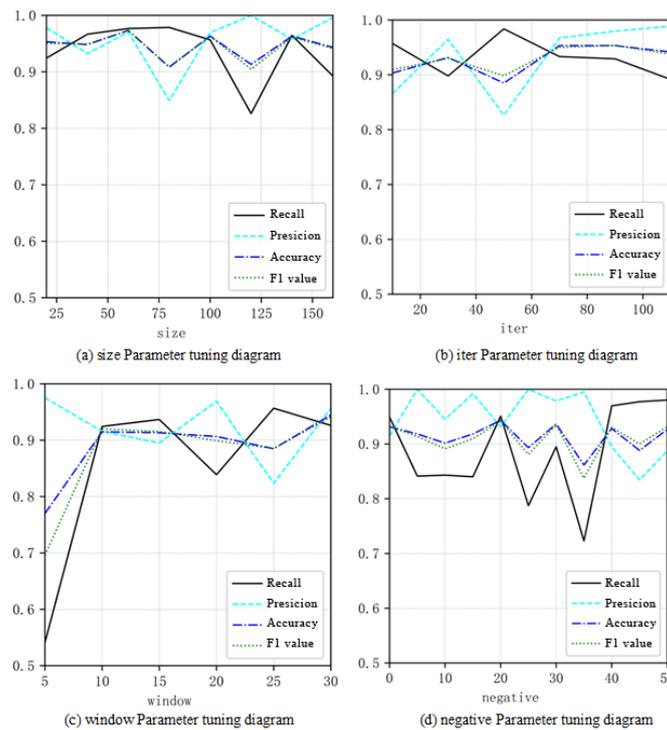


Figure 4. Word2Vec parameter tuning relationship diagram for LSTM.

The experiment comprehensively considered the recall, precision, accuracy, F1 value, and training time to adjust the parameters. Finally, we decided to set the training parameters of Word2Vec as follows: “size” to 60; “windows” to 10; “negative” to 20, and “iter” to 70.

To evaluate the LSTM model trained in the experiment objectively, we compared the model with the traditional machine learning algorithm based on ADTree and AdaBoost proposed by WangRui. Their method used the same XSS malicious sample dataset as this paper. The paper also selected the SafeDog and XSSChop for comparative experiments. The comparing results are shown in Table 2.

Table 2. Comparative experiment of the XSS detection model for LSTM.

Model	Precision	Recall	F1 Value
LSTM	0.995	0.979	0.987
ADTree	0.938	0.936	0.936
AdaBoost	0.941	0.939	0.939
SafeDog	0.997	0.902	0.948
XSSChop	1.0	0.616	0.762

From the result, the accuracy of the LSTM-based XSS attack detection model was 99.5%; the recall rate was 97.9%; and the F1 value was 98.7%. The performance in terms of accuracy, recall, and F1 was superior to the traditional machine learning algorithms ADTree and AdaBoost. The accuracy of the LSTM model in this paper was slightly lower than SafeDog and XSSChop, but the accuracy rates of the three were all more than 99.5%. Besides, the LSTM detection model trained in the experiment was superior to the website SafeDog and XSSChop in terms of recall rate and F1 value. In summary, the LSTM-based detection model trained in the experiment had obvious advantages in accuracy, recall rate, and F1 value, which proves that the model can effectively detect cross-site scripting attacks.

4.4. Adversarial Attack Experiment Results

We entered the XSS dataset into the adversarial model and got a batch of XSS adversarial samples after training. We organized these samples and named them the “adversarial dataset”. The adversarial

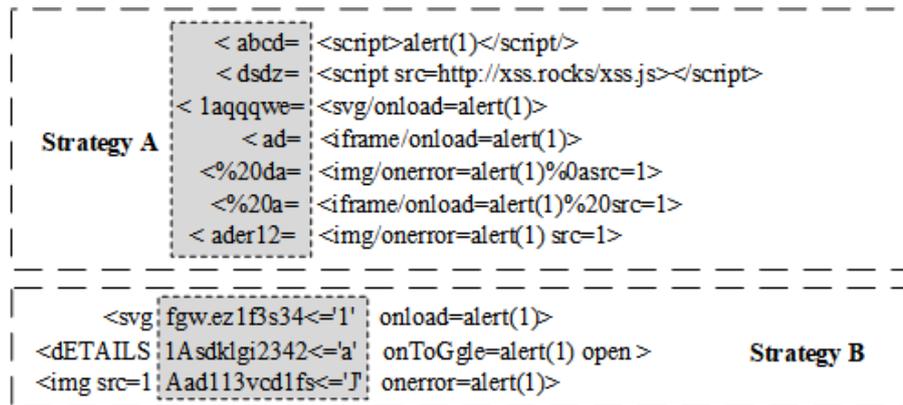


Figure 6. Examples of generic XSS escape strategies for XSSChop.

On the one hand, Strategy A is as follows. A string was added before the attack vector, and its structure satisfied: “<” + “space” + “any length of letters or numbers” + “=”; at the same time, if the attack vector contained events such as onload, onerror, and so on, we needed to ensure that there was no “space” before the corresponding event. Finally, we could change the position of the event expression in the attack vector and replace the space with “/”. An attack vector bypassing XSSChop detection was constructed. On the other hand, Bypass Strategy B was as follows. According to the escape action, we added an arbitrary assignment expression before onload, ontoggle, onerror, onclick, and other events of the attack vector, so that its structure satisfied “any length of letters or numbers” + “<=” + “single or double quotes” + “any length of letters or numbers” + “single or double quotes (need to be same as the previous ones)”. The constructed attack vectors maintained the attack function of the constructed vector, and the attack vectors bypassed XSSChop detection.

4.5. Optimizing the Experimental Results

The adversarial samples were marked as malicious samples of XSS attacks, and the DeepXSS model was retrained. The experimental results of 10 rounds of alternate training between the detection model and the adversarial model are shown in Figure 7.

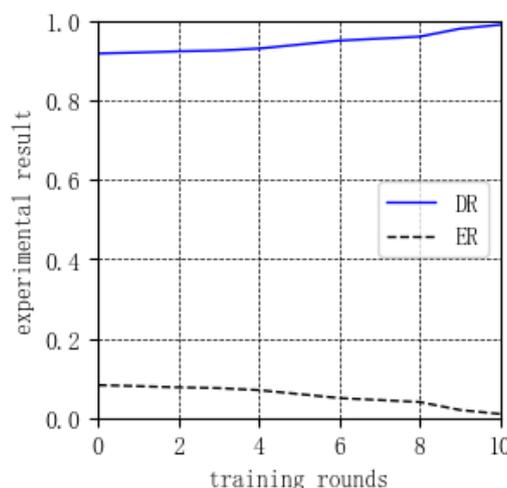


Figure 7. Optimizing experiment results.

The experimental results showed that the detection rate of DeepXSS was continuously increased, and the escape rate was continuously reduced. Finally, the escape rate of the DeepXSS model was reduced to less than 1%. This proved that the ability of the defense adversarial attack of the detection model can be improved by alternate training of the detection model and adversarial model.

In summary, RLXSS can effectively mine adversarial samples. By marking adversarial samples as malicious samples, the detection model was retrained. The adversarial sample space of the detection model was compressed, so that the detection rate was improved and the escape rate reduced. Thus, the ability of the detection model to defend against adversarial attack was effectively optimized.

5. Conclusions and Future Work

Aiming at the risk of escaping adversarial attack in current detection models and tools, this paper proposed an XSS adversarial attack model based on reinforcement learning, called RLXSS. In the RLXSS, the adversarial model generates XSS adversarial samples, and the retraining model uses these samples to optimize the XSS classifier. In this paper, the problem of escaping attack was transformed into the problem of choosing the optimal escaping strategy. We mined the adversarial samples of the black-box and white-box detection model by RLXSS, which can successfully escape detection and retain the attack function. What is more, by marking the adversarial samples as malicious samples, it retrained the detection model and alternately trained the detection model and adversarial attack model, so that the detection model kept excellent detection ability while continuously improving its ability to defend against the adversarial attack. Finally, the experimental results showed that the RLXSS successfully excavated the general XSS escape strategy for SafeDog and the general XSS escape strategy for XSSChop. Through the retraining model, the ability of DeepXSS to defend against adversarial attacks was continuously improved.

In the future, we plan to apply the reinforcement learning-based adversarial attack model to other areas of cybersecurity, such as SQL adversarial attack research work and Distributed Denial of Service (DDoS) adversarial attack research work.

Author Contributions: Conceptualization, Y.F., Y.L. and C.H.; methodology, software, Y.X.; validation, Y.F., Y.X.; formal analysis, Y.F., C.H.; investigation, Y.X.; resources, Y.F., C.H.; data curation, Y.L., Y.X.; writing—original draft preparation, Y.L.; writing—review and editing, Y.X., C.H.; supervision, Y.F.; project administration, C.H.

Funding: This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802900, the Fundamental Research Funds for the Central Universities, and the Sichuan University Postdoc Research Foundation under Grant 19XJ0002.

Acknowledgments: We thank anonymous reviewers and editors for provided helpful comments on earlier drafts of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nirmal, K.; Janet, B.; Kumar, R. Web Application Vulnerabilities-The Hacker's Treasure. In Proceedings of the 2018 International Conference on Inventive Research in Computing Applications, Coimbatore, India, 11–12 July 2018; pp. 58–62.
2. Nithya, V.; Pandian, S.L.; Malarvizhi, C. A survey on detection and prevention of cross-site scripting attack. *IJNSA* **2015**, *9*, 139–152. [[CrossRef](#)]
3. Sarmah, U.; Bhattacharyya, D.K.; Kalita, J.K. A survey of detection methods for XSS attacks. *J. Netw. Comput. Appl.* **2018**, *118*, 113–143. [[CrossRef](#)]
4. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M. Generative adversarial nets. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2014; pp. 2672–2680.
5. Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE T. Evolut. Comput.* **2019**. [[CrossRef](#)]
6. Rosenberg, I.; Shabtai, A.; Rokach, L. Generic black-box end-to-end attack against state of the art API call based malware classifiers. In Proceedings of the 21st International Symposium on Research in Attacks, Intrusions and Defenses, Heraklion, Greece, 10–12 September 2018; pp. 490–510.
7. Wu, C.; Shi, J.; Yang, Y. Enhancing Machine Learning Based Malware Detection Model by Reinforcement Learning. In Proceedings of the 8th International Conference on Communication and Network Security, Qingdao, China, 2–4 November 2018; pp. 74–78.

8. Vishnu, B.A.; Jevitha, K.P. Prediction of cross-site scripting attack using machine learning algorithms. In Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing, Amritapuri, India, 10–11 October 2014; p. 55.
9. Rathore, S.; Sharma, P.K.; Park, J.H. XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs. *JIPS* **2017**, *13*, 1014–1028. [[CrossRef](#)]
10. Fang, Y.; Li, Y.; Liu, L. Deepxss: Cross site scripting detection based on deep learning. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence, Chengdu, China, 12–14 March 2018; pp. 47–51.
11. Mohammadi, M.; Chu, B.; Lipford, H.R. Detecting cross-site scripting vulnerabilities through automated unit testing. In Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security, Prague, Czech Republic, 25–29 July 2017; pp. 364–373.
12. Duchene, F.; Rawat, S.; Richier, J.L. KameleonFuzz: evolutionary fuzzing for black-box XSS detection. In Proceedings of the 4th ACM conference on Data and application security and privacy, San Antonio, TX, USA, 3–5 March 2014; pp. 37–48.
13. Guo, X.; Jin, S.; Zhang, Y. XSS vulnerability detection using optimized attack vector repertory. In Proceedings of the 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Xi'an, China, 17–19 September 2015; pp. 29–36.
14. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
15. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
16. Matthias Plappert. keras-rl. Available online: <https://github.com/keras-rl/keras-rl> (accessed on 10 May 2019).
17. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
18. Safedog. Website Safedog. Available online: http://free.safedog.cn/website_safedog.html (accessed on 10 May 2019).
19. Chaitin. XSSChop. Available online: <https://xsschop.chaitin.cn/demo/> (accessed on 10 May 2019).



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).