An XSS Attack Detection Model Based on Two-Stage AST Analysis

Qiuhua Wang, Chuangchuang Li, Lifeng Yuan, Dong Wang, Yeru Wang, Yizhi Ren, Weizhi Meng

***Abstract*—When web applications provide convenience to people, they also bring many network security challenges. Cross-site scripting attack is one of the most common cyber attacks, in which attackers can exploit XSS vulnerabilities for purposes such as obtaining users' private information and attacking websites. Many XSS attack detection models based on machine learning and deep learning have been proposed. However, they all ignore the security of the models themselves, so that attackers can successfully bypass these models by using XSS adversarial samples. In response to this challenge, we propose an XSS attack detection model based on the two-stage AST analysis and LSTM neural network, which utilizes the ability of AST in parsing script code to analyze the samples in two stages, thereby efficiently eliminating redundant information in the adversarial samples that interferes with the detection model. First, the JavaScript code is obtained by analyzing the HTML AST. Then, the malicious code fragments are obtained by analyzing the JavaScript AST. Finally, we use the LSTM neural network to train the XSS attack detection model to defend against the adversarial attack. Extensive experiments on the real datasets show that the accuracy rate and F1 score of the proposed model reach 0.991 and 0.998, respectively, when faced with generic XSS samples. In addition, when faced with generic XSS samples, most of the existing XSS attack detection models perform very poorly, but the detection rate of our proposed model can achieve over 0.982, which further proves its effectiveness in defending against XSS adversarial attacks.**

***Index Terms*—Cross-site scripting attack, detection, AST, web application security.**

I. Introduction

WEB applications bring many conveniences to people's lives, but they are also constantly exposed to various network attacks such as SQL injection, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), etc. Among them, XSS is one of the most common network attacks. According to OWASP, the number of XSS injection attacks has increased from 9661 in 2015 to 22041 in 2022, with a rapid growth momentum [1], as shown in Fig. 1. The fast-growing XSS attacks pose a huge threat to web applications and user privacy. Attackers can exploit XSS vulnerabilities to inject malicious code into different users' web pages for purposes such as damaging websites and stealing users' private information.

(Corresponding authors: Yizhi Ren, Weizhi Meng.)
Qiuhua Wang, Chuangchuang Li, Lifeng Yuan, Dong Wang, Yeru Wang, Yizhi Ren are with School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: wangqiuhua@hdu.edu.cn, 211270019@hdu.edu.cn, yuanlifeng@hdu.edu.cn, wangdong@hdu.edu.cn, wangyeru@hdu.edu.cn, renyz@hdu.edu.cn).
Weizhi Meng is with the Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Denmark (e-mail: weme@dtu.dk).

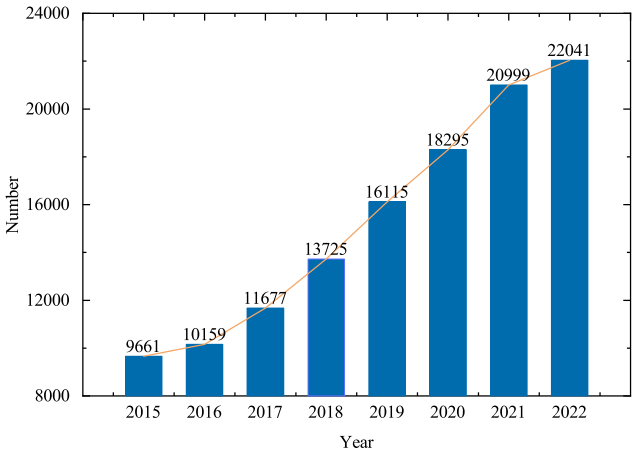

Fig. 1. The increasing trend of the XSS attacks.

Due to the huge threat of XSS, many detection methods on XSS have been proposed. In [2], [3], the semantic analysis-based approach is proposed to find malicious scripts. In [4], [5], the machine learning-based approach is proposed to detect XSS attack, and in [6], [7], the researchers proposed a deep learning-based XSS attack detection model. However, these existing research studies aim to improve the detection rate of the XSS attack detection models, while ignoring the vulnerability of the detection models themselves, which makes it difficult for the models to identify the XSS adversarial samples. In the design of XSS attack detection model, it is crucial to defend against the adversarial attack and to minimize the impact of the adversarial samples [8]. The adversarial attack can lead to misclassification of machine learning models and deep learning models by making certain changes to the original samples [9]. For example, in the field of image recognition, a picture of a panda will be identified as a gibbon by a deep learning network model with high confidence after adding a small perturbation that is undetectable to the human eye. Similarly, the adversarial attack for XSS attack detection model can be used to generate XSS adversarial samples by altering the sample characteristics through various bypass strategies (e.g., adding common strings, encoding, etc.), thereby increasing the confidence of the original XSS samples and significantly reducing the detection rate of the detection model. For example, the adversarial attack model proposed in [10] reduced the detection rate of the detection model from 95% to 1.8% and increased the escape rate of XSS samples from 5% to 98.2%.

In this paper, we propose an XSS attack detection model

based on the two-stage Abstract Syntax Tree (AST) analysis and Long Short-Term Memory (LSTM) neural network, in which the HTML part and JavaScript part of the samples are analyzed in two stages to more effectively detect and eliminate the perturbations affecting the XSS attack detection model in XSS adversarial samples and improve the security of the XSS attack detection model itself. The JavaScript code hidden by the attacker can be obtained by analyzing the HTML AST, and the potentially malicious code fragments can be obtained by analyzing the JavaScript AST. Finally, the XSS samples are classified by training the LSTM neural network. The main contributions of this paper are summarized as follows:

1) We propose a new XSS attack detection model for defending against XSS adversarial attack. The proposed model is highly secure and can effectively detect XSS adversarial samples generated against the model itself.
2) We propose a new XSS sample processing method for processing XSS samples based on the two-stage AST analysis and LSTM neural network, which analyzes both the HTML and the JavaScript parts of the XSS samples, rather than just the JavaScript part. The two-stage AST analysis can effectively find and remove the perturbations that affect the performance of the detection model in the XSS adversarial samples. It also facilitates the subsequent training of the LSTM neural network.
3) Extensive experiments on the real datasets show that the proposed model can effectively defend against the adversarial attack compared with other representative detection models.

The rest of this paper is presented as follows: In Section II, we briefly introduce the related works. In Section III, we describe the proposed approach in detail. We evaluate our method through different experiments in Section IV. Finally, we conclude the paper in Section V.

II. Related Works

Currently, researches about XSS can be divided into two main directions: the XSS attack detection and the XSS adversarial attack. It is worth noting that the attack detection model includes rule and semantic analysis-based attack detection model, machine learning-based attack detection model and deep learning-based attack detection model.

A. Rule and Semantic Analysis-based Attack Detection Models

In [11], Chaudhary et al. proposed a rule-based attack detection model for defending against XSS attacks to protect the embedded devices deployed in smart IoT systems. They utilized the Boyer-Moore string matching algorithm to compare the request parameter values with the attack vectors in the blacklist, and reduce the harm of XSS attacks through optimized filtering. The accuracy rate on both test platforms reached over 90%. However, the rule-based attack detection model relies heavily on the blacklist formulated by security experts, and the blacklists are difficult to cover the large numbers of XSS attack patterns. Attackers can bypass such XSS detection systems by applying different HTML tags, using Unicode encoding, URL encoding and HTML encoding for obfuscation, etc.

In [2], Fang et al. proposed an XSS attack detection model based on semantic analysis of JavaScript. They found the obfuscated malicious code through analyzing the AST formed by parsing the JavaScript code, and then used Fast-Text and Bi-LSTM algorithms to classify the JavaScript code to determine whether it is malicious. The experimental results showed that the accuracy rate and the recall rate reached 97.7% and 97.4%, respectively. However, they only considered the JavaScript part and neglected to analyze the HTML part of the samples, where the attackers could hide the malicious JavaScript code through the HTML bypass strategies.

In [3], Chaitin et al. performed semantic analysis on the XSS samples and scored the analysis results. Their approach was able to achieve a high accuracy rate in experiments, but failed to detect XSS attacks with context well. Attackers could also exploit the differences in parsing engines to bypass the detection of XSSChop.

B. Machine Learning-based Attack Detection Models

In [5], Rathore et al. proposed a machine learning-based detection model in which informations were extracted such as the domain and length in the URL, as well as the malicious JavaScript code, and then the random forest algorithm was used to identify XSS attacks against Social Networking Service (SNS). Their experiment achieved a high accuracy rate, but the false positive rate was also high.

In [12], Mokbal et al. proposed XGBXSS, an XSS attack detection model based on the XGBoost algorithm. They proposed a fused feature selection method consisting of information gain and sequence reverse selection which can be used to select the optimal features from the dataset to detect XSS attacks. Their experimental accuracy rate achieved 99.59%, and the recall rate achieved 99.53%.

C. Deep Learning-based Attack Detection Models

In [13], Fang et al. proposed an LSTM-based XSS attack detection model. They decoded the XSS samples by utilizing common decoding methods, followed by extracting the XSS sample features using Word2vec and training with the LSTM neural network. The experiment results showed that the accuracy rate reached 99.5%. In [4], Akaishi et al. converted XSS samples into word vectors by using Word2vec and classified them with various algorithms, showing that CNN and SVM were the best classifiers. However, the dataset they used did not contain the latest HTML5 tags, and the attackers could bypass the detection by encoding the malicious samples.

In [6], Mokbal et al. proposed a dynamic feature extraction approach with an MLP-based XSS attack detection model, obtaining the accuracy rate of 99.32% and the recall rate of 98.35%. However, they parsed HTML and JavaScript through traversal, which is inefficient in the real environment.

In [7], Tekerek et al. proposed a web application attack detection framework based on deep learning. The whole structure of the framework was divided into data preprocessing and CNN. They collected different types of Web attack data

TABLE I
SUMMARY OF EXISTING MODELS

Classification	Authors	Method	Target	Publication year
Rule and Semantic Analysis-based	Chaudhary et al. [11]	Boyer-Moore string matching algorithm and blacklist	Defense	2022
	Fang et al. [2]	Semantic analysis of JavaScript, Fast-Text and Bi-LSTM algorithms	Defense	2020
	Chaitin et al. [3]	Semantic analysis of the XSS samples	Defense	2022
Machine Learning-based	Rathore et al. [5]	Random forest algorithm	Defense	2017
	Mokbal et al. [12]	A fused feature selection approach and XGBoost algorithm	Defense	2021
Deep Learning-based	Fang et.al. [13]	Word2vec and LSTM	Defense	2018
	Akaishi et al. [4]	Word2vec, CNN and SVM	Defense	2019
	Mokbal et al. [6]	A dynamic feature extraction approach and MLP	Defense	2019
	Tekerek et al. [7]	CNN	Defense	2021
	Fang et al. [14]	DDQN algorithm	Attack	2019
	Zhang et al. [8]	Monte Carlo Tree Search (MCTS) algorithm and generative adversarial network	Attack	2020
	Wang et al. [10]	Soft Q-learning algorithm	Attack	2022
	Mondal et al. [15]	Reinforcement learning algorithm	Defense	2022
	Gupta et al. [16]	GeneMiner algorithm	Defense	2022

for training, including SQL injection, CRFL injection, XSS, XXE injection, etc. The experimental results showed that the accuracy rate reached 99.5% and the recall rate reached 97.9%. However, the attacker could use some special HTTP request packets, such as HTTP chunks, to bypass the data preprocessing stage, thereby causing the classifier to output incorrect results.

D. Adversarial Attacks and Detection Models

Recently, the attack methodology against the XSS attack detection model itself has attracted attention, and several approaches have been proposed to defend against XSS adversarial attack.

In [14], Fang et al. proposed a XSS adversarial attack model based on Dueling Deep Q Networks (DDQN), which conducted adversarial attack against different XSS attack detection models and obtained XSS adversarial samples. However, their bypass method is relatively simple and does not take into account unique bypass methods for the XSS attack detection model of deep learning and machine learning. Besides, the DDQN used in their model belongs to the policy-based reinforcement learning algorithm, which only has very few bypass policies for a single XSS sample, resulting in a low escape rate.

In [8], Zhang et al. proposed the MCTS-T algorithm for XSS attack detection based on Monte Carlo Tree Search (MCTS), and utilized generative adversarial network to generate XSS adversarial samples. They optimized the detection rate of the XSS attack detection model by learning the features of the XSS adversarial samples. However, in term of the bypass method, they only employ some encoding and case substitution, and after being encoded, the attack characteristic of the XSS samples may become invalid.

In [10], Wang et al. proposed a Soft Q-learning-based XSS adversarial attack model, which consists of the HTML bypass stage and the JavaScript bypass stage. The results showed that their model could successfully generate adversarial samples against various XSS attack detection models, with an escape rate of more than 85%.

In [15], Mondal et al. utilized reinforcement learning to enhance the defense ability of the XSS attack detection model against the XSS adversarial samples. They first extracted information about the detection model using a reinforcement learning framework, and then trained it using an adversarial strategy. After multiple cycles of attack and detection training, the capability of the detection model to protect itself was improved.

In [16], Gupt et al. proposed GeneMiner, an XSS attack detection framework that uses genetic algorithms to deals with adversarial samples. The framework includes GeneMiner-E and GeneMiner-C, the former is used to extract new features of XSS samples, and the latter is used to classify XSS samples as malicious or benign. Their experiment results showed that the detection rate against the newly generated XSS attack samples could reach 98.5%.

E. Summary of existing models

In this section, we summarize the existing models by considering the characteristics of classification, authors, method, target, and publication year, as shown in Table I.

III. THE XSS ATTACK DETECTION MODEL BASED ON TWO-STAGE AST ANALYSIS

In this section, we will discuss our proposed two-stage AST analysis based XSS attack detection model in detail.

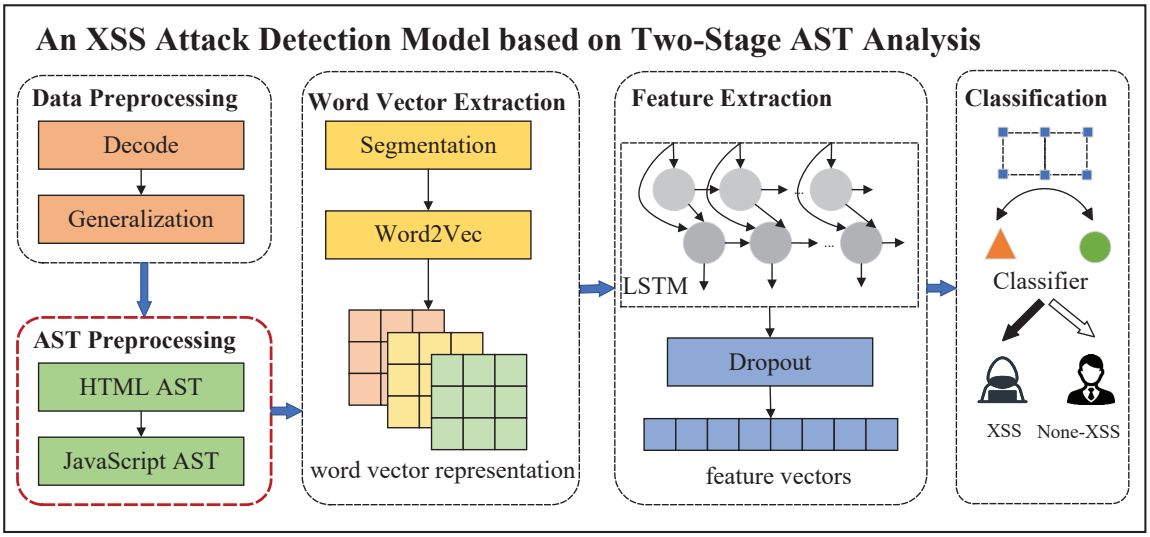

Fig. 2. The overview of our proposed XSS attack detection model. After the data preprocessing and AST preprocessing stages, we obtain the samples with the interference information removed. Then, we use Word2Vec to get the vector representation of the samples and input to LSTM for training, and finally obtain the classification result of the samples: XSS or None-XSS.

\u003cimg%20src=1%20onload=alert(123/\u003e ⟶ ⟶

Fig. 3. An example of decoding and generalization. After the samples are decoded by several decoding methods, the numbers and the URLs in them are replaced with '0' and 'http://u', respectively.

The whole structure of our proposed model is outlined in Fig. 2, which consists of four modules: data preprocessing, AST preprocessing, word vector extraction, and feature extraction and classifier. Firstly, the model preprocesses the samples, including decoding and generalization. Subsequently, in the AST preprocessing stage, the HTML part and JavaScript parts of the samples are parsed into HTML AST and JavaScript AST, respectively. Some invalid information within them will be removed, and the JavaScript AST and HTML AST will be reconstructed into JavaScript and HTML and form the simplified samples. Then, the model performs word segmentation on the simplified samples and obtains the word vectorized representation of them using Word2vec. Finally, an XSS classifier is obtained by training the word vectors with an LSTM neural network, which classifies the samples into XSS samples and None-XSS samples. After the above steps, the model can not only detect the generic XSS samples, but also detect the XSS adversarial samples efficiently.

A. Data Preprocessing

When the browser displays a web page, it will automatically decode and parse the code, which can be exploited by the attackers to bypass the XSS attack detection model by encoding the malicious scripts. After decoding, the XSS samples may also contain numbers and URLs, which are irrelevant to the XSS attack detection model and may affect the detection performance due to their excessive number. Therefore, the XSS samples need to be decoded and generalized during the data preprocessing stage. The decoding approaches include URL decoding, HTML entity decoding, Unicode decoding, and Base64 decoding. Then the generalization operations include uniformly replacing the number with '0' and the URL with 'http://u'. Fig. 3 shows an example of decoding and generalization.

B. AST preprocessing

The main purpose of the AST preprocessing is to find the script actually executed by the XSS samples and eliminate some invalid data that may affect the judgement of the XSS attack detection model. A simplified XSS sample can be obtained after analyzing a complicated and obfuscated XSS sample with HTML AST and JavaScript AST, as shown in Fig. 4.

Our proposed two-stage AST analysis method can be divided into HTML AST preprocessing and JavaScript AST preprocessing. Fig. 5 illustrates the HTML AST preprocessing process of an XSS sample. We use parse5 as the parser in this paper, because parse5 is currently the fastest HTML parser with full HTML processing abilities [17]. When a malicious XSS sample that has been obfuscated several times is analyzed, we can obtain the data in JSON format, which can be traversed to directly find the part of the XSS sample executing the specific JavaScript script. As shown in Fig. 5, there exists malicious JavaScript code in the 'onerror' attribute of 'img' (the red part in the figure). Since in the XSS attack detection environment, the detection model only needs to focus on the part of the XSS sample that executes JavaScript code, the other HTML attributes will be removed.

The JavaScript AST preprocessing will be performed on the JavaScript part of the XSS samples by traversing the JavaScript

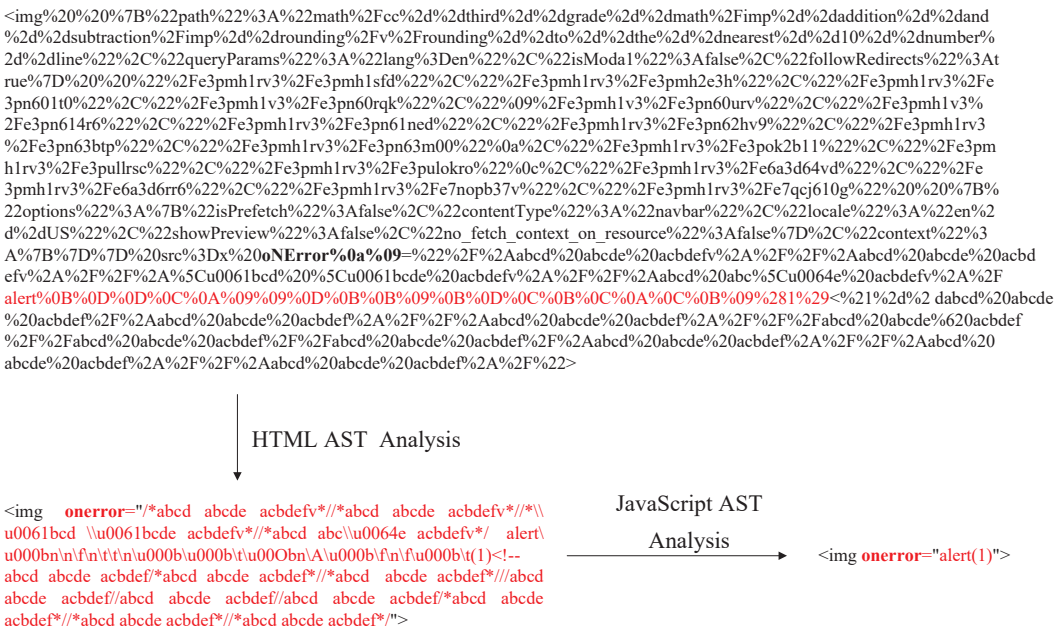

Fig. 4. An example of the two-stage AST analysis.

syntax tree and eliminating some invalid data, such as comment data, some special characters added through obfuscation methods, etc. Fig. 6 shows the AST analysis process of a malicious JavaScript script, with some comment messages such as "/*abcd abcde acbdefv*/", "<! –abcd abcde acbdef/*abcdabcde acbdef*//*abcd abcde acbdef*///abcd abcde acbdef//abcd abcde acbdef//abcdabcde acbdef/*abcd abcde acbdef/*abcd abcde acbdef*//* abcd abcde acbdef*//*abcd abcde acbdef*/", as well as some characters added using XSS obfuscation methods will be removed.

C. Word Vector Extraction

The word vector extraction consists of two steps: word segmentation and word vector acquisition. The main purpose of word segmentation is to divide the XSS samples according to certain rules and input them to a word vector representation training model. Fig. 7 shows the process of word segmentation of an XSS sample. After the two-stage AST analysis, we use regular expression matching to segment the results. Since the work of decoding and eliminating some redundant data have already been performed in Sections 3.1 and 3.2, the word separation can directly divide the XSS samples into tags, sensitive functions, executed JavaScript, URLs, brackets, etc.

We choose Word2vec to obtain the word vector representation of the XSS samples, which was proposed by Mikolov et al. [18] in 2013 to transform a given textual content into a k-dimensional word vector through the combination of neural network and combinatorial probabilities. We can choose either the Continuous Bag of Words (CBOW) or the Skip-gram model for training. These two models are shown in Fig. 8 (the window size is set to 2, indicating the maximum distance between the current word and the predicted word in one sentence). CBOW calculates the probability distribution of the central word by the context, e.g., given the central word $w(t)$ within the window of $w(t-2)$, $w(t-1)$, $w(t+1)$, $w(t+2)$, the probability of the central word $w(t)$ is predicted, whereas the Skip-gram model predicts the probability of the contextual distribution of the current central word based on the current word, e.g. given the central word $w(t)$, the probabilities of $w(t-2)$, $w(t-1)$, $w(t+1)$, $w(t+2)$ within the prediction window are predicted. Skip-gram and CBOW are both bag-of-words models. However, compared with the CBOW model, the Skip-gram model is not constrained by the window size, it allows certain phrases to be skipped, which can better represent the relationship between the current word and the words beyond the window, so as to obtain a higher quality word vector. Therefore, we choose the Skip-gram model in this paper. Fig. 9 shows an example of the word vector generation using Word2vec for a segmented XSS sample.

D. Feature Extraction and Classifier

We build the XSS attack detection model based on LSTM [19]. LSTM is a kind of RNN with special structure that can solve the gradient disappearance and gradient explosion problems easily occurred during the training process of RNN, and it has the ability to preserve information for a long time, which can maximize the use of the feature vectors of the samples and achieve better performance. The overall structure of the model is shown in Fig. 10, which consists of an input layer, a hidden layer, a softmax layer and an output layer.

In the input layer, the word vectors x_1, x_2, x_3, ..., xt generated by Word2vec are used as input, and the LSTM hidden layer transforms the data by layer with features and obtain the corresponding output h_1, h_2, h_3, ..., h_t.

The output of the LSTM hidden layer will be used as the input of the Dropout layer, which aims to prevent overfitting caused by the lack of feature dimensions during the training of the XSS attack detection model. The Bernoulli function

<img%20%20%7B%22path%22%3A%22math%2Fcc%2d%2dthird%2d%2dgrade%2d%2dmath%2Fimp%2d%2daddition%2d%2dand
%2d%2dsubtraction%2Fimp%2d%2drounding%2Fv%2Frounding%2d%2dto%2d%2dthe%2d%2dnearest%2d%2d10%2d%2dnumber%
2d%2dline%22%2C%22queryParams%22%3A%22lang%3Den%22%2C%22isModa1%22%3Afalse%2C%22followRedirects%22%3At
rue%7D%20%20%22%2Fe3pmh1rv3%2Fe3pmh1sfd%22%2C%22%2Fe3pmh1rv3%2Fe3pmh2e3h%22%2C%22%2Fe3pmh1rv3%2Fe
3pn601t0%22%2C%22%2Fe3pmh1v3%2Fe3pn60rqk%22%2C%22%09%2Fe3pmh1v3%2Fe3pn60urv%22%2C%22%2Fe3pmh1v3%
2Fe3pn614r6%22%2C%22%2Fe3pmh1rv3%2Fe3pn61ned%22%2C%22%2Fe3pmh1rv3%2Fe3pn62hv9%22%2C%22%2Fe3pmh1rv3
%2Fe3pn63btp%22%2C%22%2Fe3pmh1rv3%2Fe3pn63m00%22%0a%2C%22%2Fe3pmh1rv3%2Fe3pok2b11%22%2C%22%2Fe3pm
h1rv3%2Fe3pullrsc%22%2C%22%2Fe3pmh1rv3%2Fe3pulokro%22%0c%2C%22%2Fe3pmh1rv3%2Fe6a3d64vd%22%2C%22%2Fe
3pmh1rv3%2Fe6a3d6rr6%22%2C%22%2Fe3pmh1rv3%2Fe7nopb37v%22%2C%22%2Fe3pmh1rv3%2Fe7qcj610g%22%20%20%7B%
22options%22%3A%7B%22isPrefetch%22%3Afalse%2C%22contentType%22%3A%22navbar%22%2C%22locale%22%3A%22en%2
d%2dUS%22%2C%22showPreview%22%3Afalse%2C%22no_fetch_context_on_resource%22%3Afalse%7D%2C%22context%22%3
A%7B%7D%7D%20src%3Dx%20**oNError%0a%09**=%22%2F%2Aabcd%20abcde%20acbdefv%2A%2F%2F%2Aabcd%20abcde%20acbd
efv%2A%2F%2F%2A%5Cu0061bcd%20%5Cu0061bcde%20acbdefv%2A%2F%2F%2Aabcd%20abc%5Cu0064e%20acbdefv%2A%2F
alert%0B%0D%0D%0C%0A%09%09%0D%0B%0B%09%0B%0D%0C%0B%0C%0A%0C%0B%09%281%29<%21%2d%2 dabcd%20abcde
%20acbdef%2F%2Aabcd%20abcde%20acbdef%2A%2F%2F%2Aabcd%20abcde%20acbdef%2A%2F%2F%2Fabcd%20abcde%620acbdef
%2F%2Fabcd%20abcde%20acbdef%2F%2Fabcd%20abcde%20acbdef%2F%2Aabcd%20abcde%20acbdef%2A%2F%2F%2Aabcd%20
abcde%20acbdef%2A%2F%2F%2Aabcd%20abcde%20acbdef%2A%2F%22>

HTML AST Analysis


```
{
  "nodeName":"img",
  "tagName":"img",
  "attrs":[
    {
      "name":"{\"path\":\"math",
      "value":""
    },
    {
      "name":"cc--third--grade--math",
      "value":""
    },
    {
      "name":"rounding--to--the--nearest--10--number--
line\",\"queryparams\":\"lang",
      "value":"en\",\"isModal\":false,\"followRedirects\":true)"
    },

      ......
    {
      "name":"onerror",
      "value":"/*abcd abcde acbdefv*//*abcd abcde acbdefv*//*\\u0061bcd
\u0061bcde acbdefv*//*abcd abc\\u0064e
acbdefv*/alert\uooob\n\n\f\n\t\t\n\u0oob\u0oob\t\u0oob\n\f\uoob\f\n\f\u0oob\t(1)<!-
-abcd abcde acbdef/*abcd abcde acbdef*//*abcd abcde acbdef*///abcd abcde
acbdef//abcd abcde acbdef//abcd abcde acbdef/*abcd abcde acbdef*//*abcd abcde
acbdef*//*abcd abcde acbdef*/"
    }
    ]
}
```


Fig. 5. An example of HTML AST analysis.

is mainly used in dropout layer employs, which generates a vector of 0 or 1 randomly, where 0 represents dropping the neuron and 1 represents keeping the neuron. In the formulas (1) and (2), $r_j^{(l)}$ indicates whether the jth neuron in the lth layer is enabled, 1 means enabled and 0 means dropped, d stands for the output of the Dropout layer, h represents the output of the LSTM hidden layer, and $r^{(l)}$ denotes the state of all neurons in the dropout layer.

$$r_j^{(l)} \sim Bernoulli\,(p) \tag{1}$$

$$d = r^{(l)} \times h \tag{2}$$

$$o_s = softmax\,(wd + b) \tag{3}$$

$$y = argmax\,(o_s) \tag{4}$$

In the softmax layer, the model will output the final classification result. In formula (3), w and b represent the weight and bias of the softmax, and o_s represents the classification probability of the output. We can obtain the final output y by formula (4) with a value of 0 or 1, where 1 represents that the input sample is an XSS sample and 0 represents that the input sample is a normal sample.

IV. Experiments and Discussions

In order to evaluate the effectiveness of the proposed XSS attack detection model, we conduct extensive experiments on three real datasets, focusing on the following questions:

1) Q1: Compared with the representative approaches, how does the proposed model perform against generic XSS samples?
2) Q2: Compared with the representative approaches, how does the proposed model perform against XSS adversarial samples?
3) Q3: How does the proposed two-stage AST analysis affect the performance of the detection model?

TABLE II
INTRODUCTION OF THE THREE DATASETS

Dataset	Source	Description
DeepXSS's Dataset [13]	XSSED [20] and DMOZ	There are 33,426 malicious samples derived from the XSSED database and 31,407 normal samples obtained from the DMOZ database.
Wang's Dataset [10]	PortSwigger [21], XSSED [20] and their own samples	Their dataset is derived from three parts, including PortSwigger, XSSED and their own samples constructed according to ECMAScript 2020 syntax. They take into account more HTML tags and differences between different JavaScript parsing engines, making these samples more threatening to the detection models.
RLXSS's Dataset [14]	XSSED [20]	There are 33,426 XSS samples derived from the XSSED database.

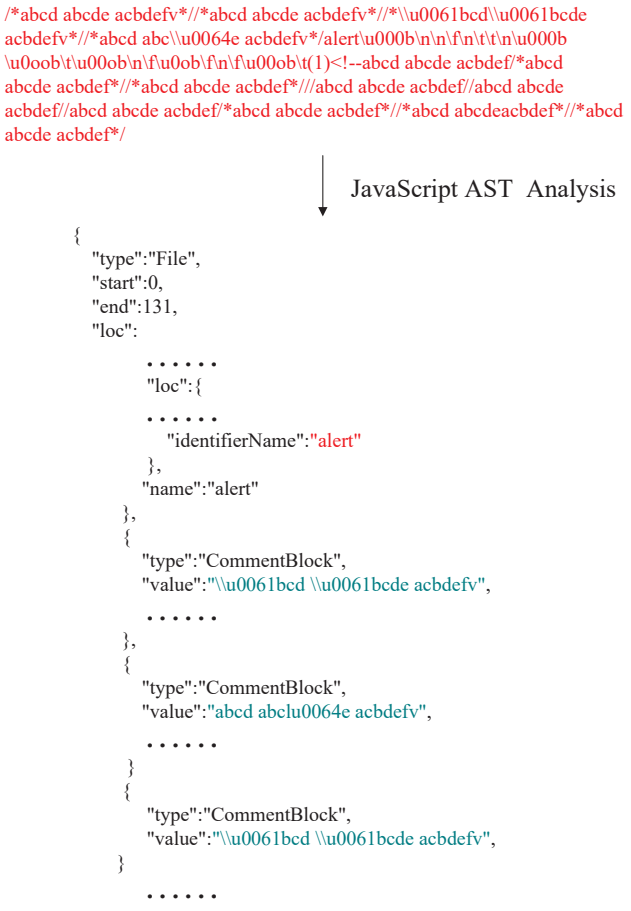

Fig. 6. An example of JavaScript AST analysis.

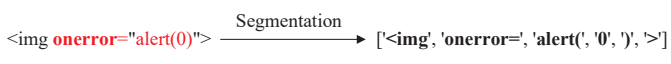

Fig. 7. An example of word segmentation.

A. The Datasets and Experimental Environment

In this paper, we used three datasets to evaluate the proposed XSS attack detection model, namely DeepXSS's dataset [13], Wang's Dataset [10] and RLXSS's Dataset [14]. Table II shows a brief description of them.

The dataset used for evaluating the capability of the XSS attack detection models against generic XSS samples is from [13], which contains 40,637 XSS samples and 31,407 benign samples. It is referenced to as the DeepXSS's dataset in this paper.

The datasets used for evaluating the capability of the XSS attack detection models against XSS adversarial samples are from [10] and [14], respectively. In [10], Wang et al. first collected XSS samples containing primarily HTML tags and corresponding triggered JavaScript events from the PortSwigger XSS spoofing table [21], and then extracted 20 malicious JavaScript samples from the XSSED [20] database. They also constructed some malicious samples based on the ECMAScript 2020 syntax. Their dataset is referred to as Wang's Dataset in this paper.

In [14], Fang et al. collected 33,426 samples as XSS samples from the XSSED database. Their dataset is referred to as the RLXSS's Dataset in this paper.

The above samples are all real and effective XSS samples from the cyber environment.

The computer configuration used for the experiments is: MacOS 10.15.4, Dual-Core Intel Core i7 CPU @2.5GHz, 16G RAM. And the programming environment is: Python 3.7, TensorFlow 1.6.0 build, Keras 2.1.5.

B. Evaluation Criteria

To evaluate the effectiveness of the XSS attack detection model, we use the recall rate, the precision rate, the F1 score, and the accuracy rate as evaluation metrics. The higher these metrics score, the better the model performs.

The recall rate refers to the probability that the actual XSS sample is predicted to be XSS sample, indicating the prediction accuracy of XSS samples. The formula is shown as follows:

$$recall = \frac{TP}{TP + FN} \tag{5}$$

The precision rate refers to the probability that all samples predicted to be XSS samples are actually XSS samples,

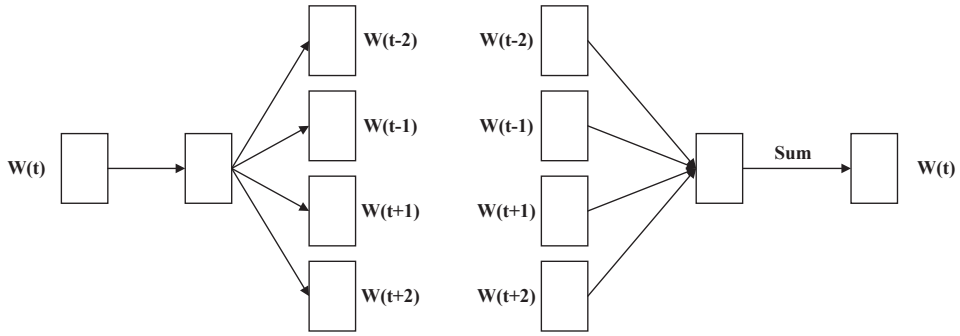

Fig. 8. Skip-gram Model(left) and CBOW Model(right).

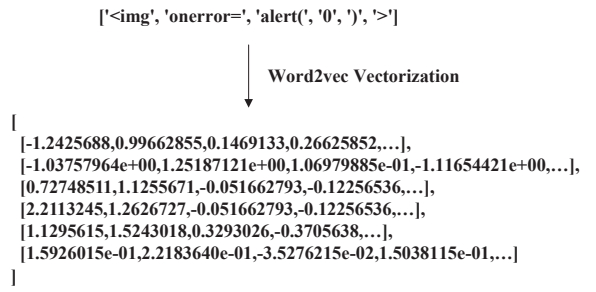

Fig. 9. An example of obtaining the word vector representation of an XSS sample.

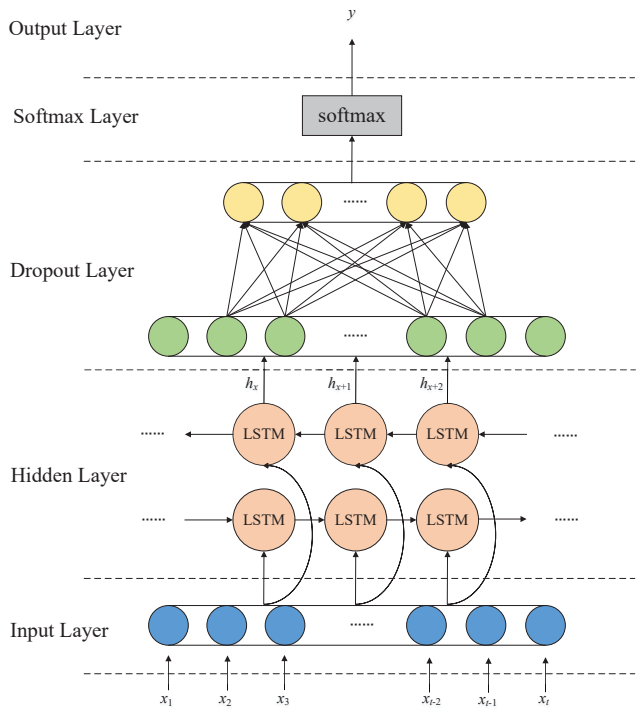

Fig. 10. The overall structure of the classification model.

indicating the prediction accuracy in the XSS samples results. The formula is shown as follows:

$$precision = \frac{TP}{TP + FP} \tag{6}$$

The F1 score takes into account both precision rate and recall rate and is the harmonic mean of the both. The formula is shown as follows:

$$F1 = \frac{2 * precision * recall}{presicion + recall} \tag{7}$$

The accuracy rate refers to the proportion of the correctly classified samples to the total number of samples. The formula is shown as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

In the above formulas, TP (True Positive) denotes the number of XSS samples accurately classified, FP (False Positive) denotes the number of benign samples misclassified as malicious samples, TN (True Negative) denotes the number of benign samples correctly classified, and FN (False Negative) denotes the number of XSS samples wrongly classified as benign samples.

Furthermore, we also use two metrics, DR (Detection Rate) and ER (Escape Rate), to evaluate the defense capability of the XSS attack detection model against XSS adversarial samples. The DR refers to the proportion of the number of XSS adversarial samples that are still classified as XSS samples after using the bypass method to the total number of XSS adversarial samples. The lower the DR is, the stronger the attack capability of the XSS adversarial attack model is, and the higher the DR is, the stronger the defense capability of the XSS attack detection model is. The ER refers to the proportion of the number of XSS adversarial samples classified as normal samples to the total number of XSS adversarial samples after using the bypass method. The higher the ER is, the stronger the attack capability of the XSS adversarial attack model is. Also, the lower the ER is, the stronger the defense capability of the XSS attack detection model is.

C. Experimental Settings

We employ TensorFlow to train the proposed XSS attack detection model. The word vector dimension k of Word2vec is set to 128. The weights of the LSTM neural network are randomly initialized, and the parameters are optimized and updated using the Adam optimizer with a learning rate of 0.1. When training the model, epoch is set to 200 and batch size is set to 128. The setup of the above training parameters is based on the experimental results. The experiment was conducted

TABLE III
GROUPS AND CORRESPONDING REPRESENTATIVE MODELS

Group	Representative models	Description
Group(A)	SVM, MLP and LSTM (The comparison models we build in this paper.)	After obtaining the corresponding features of the XSS samples using Word2vec, we built XSS detection models based on SVM, MLP and LSTM, respectively. The models based on machine learning and deep learning can maximize the utilization rate of the features of XSS samples.
Group(B)	MLPXSS [6] and DeepXSS [13]	MLPXSS is an MLP-based XSS attack detection model which detects XSS samples by dynamically extracting XSS sample features. DeepXSS is an LSTM-based XSS attack detection model, which first decodes XSS samples and then extracts XSS sample features. The two approaches represent the best deep learning based XSS attack detection models.
Group(C)	SafeDog [22] and XSSChop [3]	SafeDog and XSSChop are two commercial models that can detect XSS attacks.

TABLE IV
EXPERIMENTAL RESULTS OF XSS ATTACK DETECTION MODEL

Models	Recall	Precision	F1	Accuracy
SVM (built by ourselves)	0.9776	0.9984	0.9879	0.9877
MLP (built by ourselves)	0.9837	0.9983	0.9910	0.9908
LSTM (built by ourselves)	0.9842	0.9991	0.9916	0.9914
MLPXSS [6]	0.9835	0.9921	0.9877	**0.9932**
DeepXSS [13]	0.9790	0.9950	0.9870	-
SafeDog [22]	0.9286	0.9972	0.9617	0.9366
XSSChop [3]	0.8243	**0.9999**	0.9036	0.8494
AST+LSTM (ours)	**0.9904**	0.9981	**0.9942**	0.9913

five times and the results presented in the paper are the average of all five runs.

D. Compared Models

We compare the proposed model with some representative models, including the models based on SVM, MLP and LSTM in this paper, the models in [6] and [13], and two commercial models, SafeDog and XSSChop. We classify the above models into three groups as shown in Table III.

E. Experimental Results

We conducted extensive experiments on the above three datasets to answer the three research questions posed: Q1-Q3.

1) Performance of Detection Models Against Generic XSS Samples: We conduct experiments using the DeepXSS's dataset on the seven attack detection models mentioned above as well as on the model proposed in this paper (represented by AST+LSTM) to answer the research question Q1. The results are shown in Table IV.

As shown in Table IV, given the same dataset, the recall rate, the precision rate, the F1 score and the accuracy rate of all XSS attack detection models are higher than 92%. The only exception is XSSChop, which is also higher than 82% for all evaluation metrics. The proposed model performs comparably to the best in all metrics. The experimental results show that these XSS attack detection models can effectively detect XSS samples without considering XSS adversarial samples. This is because the structure of the generic XSS sample is simple, and there is no targeted interference on the XSS attack detection model, so each detection model can achieve good detection results after training.

2) Performance of Detection Models Against XSS Adversarial Samples: We employ the XSS adversarial attack model proposed in [10] to attack each XSS attack detection model based on Wang's dataset and RLXSS's dataset to answer the research question Q2. The results are shown in Table V.

As can be seen from Table V, the XSS attack detection model performs differently against XSS adversarial samples generated based on different datasets. When faced with the XSS adversarial samples generated based on the RLXSS's Dataset, the DR of XSSChop achieves 0.879, indicating that it can still defend against most of these XSS attacks, although its effectiveness is compromised. However, when faced with XSS adversarial samples generated based on Wang's Dataset, the DR of XSSChop is as low as 0.058, indicating that it is no longer able to defend against such XSS attacks. In addition, SafeDog and models in Group(A) have lower DR against XSS adversarial samples generated based on Wang's dataset, suggesting that such XSS adversarial samples pose a higher threat to XSS attack detection models due to the

TABLE V
EXPERIMENTAL RESULTS OF THE XSS ADVERSARIAL ATTACK MODEL

Models	Wang's Dataset		RLXSS's Dataset	
	DR	ER	DR	ER
SVM (built by ourselves)	0.141	0.859	0.198	0.802
MLP (built by ourselves)	0.047	0.953	0.119	0.881
LSTM (built by ourselves)	0.138	0.862	0.241	0.759
SafeDog [22]	0.031	0.969	0.057	0.943
XSSChop [3]	0.058	0.942	0.879	0.121
AST+LSTM (ours)	**0.982**	**0.018**	**0.998**	**0.002**

fact that Wang et al. considered more new features of HTML and JavaScript when constructing their dataset. However, the DR of the proposed model AST + LSTM reaches more than 0.98 on both datasets, which proves that its detection effect is independent of the dataset and its robustness is better than the other five models.

Furthermore, through the Table V, we can find that the DR of SafeDog is less than 0.06 when facing two types of XSS adversarial samples, and the highest DR of the models in Group(A) is only 0.241. The DR of the best performing XSSChop is only 0.058 when faced with XSS adversarial samples generated based on the Wang's dataset. This indicates that these models are no longer effective in protecting web applications and user's privacy information. While the DR of AST + LSTM is 0.982 and 0.998 when faced with two types of XSS adversarial samples based on Wang's Dataset and RLXSS's Dataset, respectively, making the ER of XSS adversarial samples be only 0.018 and 0.002 respectively, which greatly improves the detection effect compared with other models. It illustrates that our proposed model can make up for the defect that the existing detection model cannot effectively detect XSS adversarial samples, and improve the security and reliability of the XSS attack detection model itself. This result is due to the fact that the normal strings and special characters in the XSS adversarial samples interfere heavily with the detection models, while the proposed model discards them through the two-stage AST analysis, allowing the following LSTM neural network to utilize the features of the XSS adversarial samples better without interference, so AST+LSTM can achieve better detection results.

3) Impact of The Two-stage AST Analysis on Detection Models: We conducted two sets of experiments on generic XSS samples and XSS adversarial samples, including LSTM and AST + LSTM, to answer the research question Q3. LSTM refers to the construction of detection models directly using LSTM without two-stage AST analysis, and AST + LSTM refers to the model proposed in this paper. The experimental results are shown in Fig. 11 and Fig. 12, and the more detailed results can be referred to Table IV and Table V.

From Fig. 11 we can see that the AST + LSTM is slightly better than the LSTM in terms of the F1 score when faced with generic XSS samples, but they are nearly identical in terms of the accuracy rate. This is because that the main purpose of the two-stage AST analysis is to efficiently analyze the

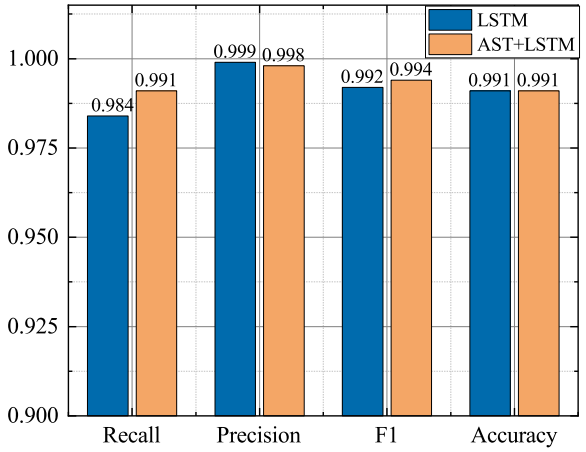

Fig. 11. The performance of LSTM and AST + LSTM against generic XSS samples.

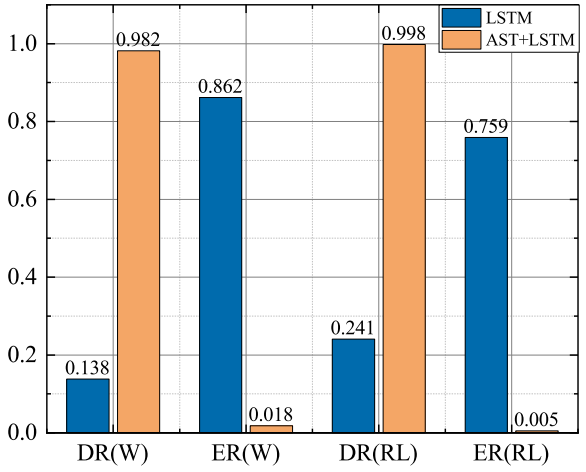

Fig. 12. The performance of LSTM and AST + LSTM against XSS adversarial samples. Where the words in brackets represent the dataset used, W stands for the Wang's Dataset and RL stands for the RLXSS's Dataset.

structure of samples and find malicious scripts hidden by the attackers through various bypass strategies, while the generic XSS samples may not all perform this bypass step. Therefore, the LSTM can also perform well without performing a two-stage AST analysis. However, it can also be seen from Fig. 11 that the two-stage AST analysis effectively improves the recall rate of the detection model, which is often the most important

metric for the network attack detection model as it represents the ability to discover network attacks.

From Fig. 12 we can see that AST + LSTM achieves a very significant improvement over LSTM when faced with XSS adversarial samples, whether based on Wang's Dataset or RLXSS's Dataset. When faced with XSS adversarial samples generated based on the Wang's Dataset, the DR of AST + LSTM and LSTM are 0.982 and 0.138, respectively. When faced with XSS adversarial samples generated based on the RLXSS's Dataset, the DR of AST + LSTM and LSTM are 0.998 and 0.241, respectively. This can be attributed to the fact that the XSS attack detection model based on LSTM alone has difficulty in eliminating the impact of the attacker's various bypass strategies on the XSS samples, while the two-stage AST analysis can easily do so by providing in-depth analysis of the XSS sample structure. The results in Fig. 12 fully illustrate the effectiveness of the two-stage AST analysis for detecting the XSS adversarial samples.

4) Discussion: From the above analysis, we can see that AST + LSTM performs very well against XSS adversarial samples. However, there are still some ways that can make the model perform better. Firstly, by expanding the number of samples in the training set as much as possible, it can make the model predict more accurately. Secondly, in the AST preprocessing module, if the parsing engine can be updated in real time to support the latest HTML and JavaScript grammars, it can effectively prevent attackers from using the latest grammars to construct XSS adversarial samples to bypass the XSS attack detection model. In addition, we plan to leverage the powerful representational learning capability of Graph Convolutional Networks (GCN) to better defend against XSS adversarial attacks.

V. Conclusion

In this paper, we proposed an XSS attack detection model based on the two-stage AST analysis and the LSTM neural network, which is very effective in the face of XSS adversarial samples. The model first goes through two stages of HTML AST and JavaScript AST to analyze the XSS samples and find potentially malicious code fragments, and then obtains the XSS attack detection model by training an LSTM neural network. After analyzing the HTML part of the XSS samples, the proposed model can eliminate the interference of various irrelevant information, find the malicious code that may be hidden by the attackers. Extensive experiments on real datasets demonstrated that, compared with representative detection models, our proposed model performed better, especially in defending against XSS adversarial attack, which can achieve a detection rate of over 0.982. For the future work, we plan to utilize techniques such as graph neural network to provide better performance for defending against XSS adversarial attacks.

Acknowledgments

This research was supported by "Pioneer" and "Leading Goose" R&D Program of Zhejiang (Grant No. 2023C03203, 2022C03174, 2023C03180).

References

[1] (2022) CVE. [Online]. Available: https://cve.mitre.org/
[2] Y. Fang, C. Huang, Y. Su, and Y. Qiu, "Detecting malicious JavaScript code based on semantic analysis," *Computers & Security*, vol. 93, p. 101764, 2020.
[3] (2022) XSSChop. [Online]. Available: https://xsschop.chaitin.cn/
[4] S. Akaishi and R. Uda, "Classification of XSS attacks by machine learning with frequency of appearance and co-occurrence," in *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2019, pp. 1–6.
[5] S. Rathore, P. K. Sharma, and J. H. Park, "XSSClassifier: an efficient XSS attack detection approach based on machine learning classifier on SNSs," *Journal of Information Processing Systems*, vol. 13, no. 4, pp. 1014–1028, 2017.
[6] 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.
[7] A. Tekerek, "A novel architecture for web-based attack detection using convolutional neural network," *Computers & Security*, vol. 100, p. 102096, 2021.
[8] X. Zhang, Y. Zhou, S. Pei, J. Zhuge, and J. Chen, "Adversarial examples detection for XSS attacks based on generative adversarial networks," *IEEE Access*, vol. 8, pp. 10 989–10 996, 2020.
[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
[10] 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.
[11] P. Chaudhary, B. B. Gupta, and A. Singh, "Securing heterogeneous embedded devices against XSS attack in intelligent IoT system," *Computers & Security*, vol. 118, p. 102710, 2022.
[12] F. M. M. Mokbal, W. Dan, W. Xiaoxi, Z. Wenbin, and F. Lihua, "XGBXSS: an extreme gradient boosting detection framework for cross-site scripting attacks based on hybrid feature selection approach and parameters optimization," *Journal of Information Security and Applications*, vol. 58, p. 102813, 2021.
[13] 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.
[14] 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.
[15] B. Mondal, A. Banerjee, and S. Gupta, "XSS Filter detection using Trust Region Policy Optimization," in *2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC)*. IEEE, 2023, pp. 1–4.
[16] C. Gupta, R. K. Singh, and A. K. Mohapatra, "Geneminer: a classification approach for detection of XSS attacks on web services," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.
[17] (2022) "parse5,"npm. [Online]. Available: https://www.npmjs.com/package/parse5
[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.
[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[20] (2012) XSSed: XSS attacks information. [Online]. Available: http://www.xssed.com/archive
[21] (2021) Cross-Site Scripting (XSS) Cheat Sheet - 2021 Edition — Web Security Academy. [Online]. Available: https://portswigger.net/websecurity/cross-site-scripting/cheat-sheet
[22] (2020) Safedog: web attack detection engine. [Online]. Available: http://www.safedog.cn/

VI. Biography Section