GCNXSS: An Attack Detection Approach for Cross-Site Scripting Based on Graph Convolutional Networks

Hongyu Pan[1], Yong Fang[1], Cheng Huang[1*], Wenbo Guo[1], and Xuelin Wan[2]
[1] School of Cyber Science and Engineering, Sichuan University
Chengdu, 610065, China
[e-mail: opcodesec@gmail.com]
[2] China Merchants Bank
Shenzhen, 518057, China
[e-mail: wanxuelin@cmbchina.com]
*Corresponding author: Cheng Huang

Received March 20, 2022; revised October 3, 2022; accepted October 24, 2022; published December 31, 2022

Abstract

Since machine learning was introduced into cross-site scripting (XSS) attack detection, many researchers have conducted related studies and achieved significant results, such as saving time and labor costs by not maintaining a rule database, which is required by traditional XSS attack detection methods. However, this topic came across some problems, such as poor generalization ability, significant false negative rate (FNR) and false positive rate (FPR). Moreover, the automatic clustering property of graph convolutional networks (GCN) has attracted the attention of researchers. In the field of natural language process (NLP), the results of graph embedding based on GCN are automatically clustered in space without any training, which means that text data can be classified just by the embedding process based on GCN. Previously, other methods required training with the help of labeled data after embedding to complete data classification. With the help of the GCN auto-clustering feature and labeled data, this research proposes an approach to detect XSS attacks (called GCNXSS) to mine the dependencies between the units that constitute an XSS payload. First, GCNXSS transforms a URL into a word homogeneous graph based on word co-occurrence relationships. Then, GCNXSS inputs the graph into the GCN model for graph embedding and gets the classification results. Experimental results show that GCNXSS achieved successful results with accuracy, precision, recall, F1-score, FNR, FPR, and predicted time scores of 99.97%, 99.75%, 99.97%, 99.86%, 0.03%, 0.03%, and 0.0461ms. Compared with existing methods, GCNXSS has a lower FNR and FPR with stronger generalization ability.

Keywords: Web security, Cross-site Scripting, Graph Convolutional Networks(GCN)

This work was supported in part by National Natural Science Foundation of China (U20B2045).

1. Introduction

With the continuous development of information technology, Web applications have become more abundant and have penetrated all corners of people's lives. Web applications occupy a higher proportion of people's lives and contain more and more value. Therefore, Web applications have become the main target of attackers. According to Precise Security's research, nearly 40% of all attacks recorded by security experts are XSS attacks [1]. In addition, the OWASP Top 10 released 2017 shows that XSS remains one of the most threatening attack methods [2].

Cross-site scripting (XSS) is a common code-embedding vulnerability in Web applications, with two-thirds of all Web applications being vulnerable to XSS [2]. The attacker can use the XSS vulnerability to construct a malicious script code embedded in the Web page visited by ordinary users. Once ordinary users visit the page embedded with malicious script code, the attacker will construct a malicious script code that can carry out the theft of user accounts, phishing, illegal money transfer, and other malicious operations. An XSS vulnerability is also one of the most harmful web application vulnerabilities. Furthermore, it is one of the primary attack methods of web attacks, causing significant damage to the economy and personal privacy [3]. To defend against code embedded vulnerabilities like XSS, 94% of the applications were tested for some form of injection [4].

XSS attacks are a problem that cannot be ignored in Web security, and many researchers have conducted much research on the field of XSS attacks detection. However, with the continuous development of information technology, XSS attack is transforming and becoming diversified [5], resulting in increasing difficulty in detecting XSS attack.

Traditional XSS attack detection methods need to maintain a rule database, and each rule in the rule database needs to be extracted by security experts, which can take a lot of time. Moreover, the accuracy of the detection method depends on the quality of the rule database, and the poor quality of the rule database will lead to poor results in detecting XSS attacks. Traditional XSS attack detection methods have difficulty dealing with these problems. Since the introduction of machine learning techniques into XSS attack detection, many researchers have conducted related researches and achieved remarkable results, such as applying SVM, Naive Bayes, ADTree, and other methods to the field of XSS attack detection [6].

However, according to the literature review, the existing XSS attack detection methods based on machine learning still have significant shortcomings, such as significant FNR and FPR [6]. The impact caused by FNR is more significant than the increased labor cost caused by FPR. Once an XSS attack evades the security system, it is likely to cause damage to the system. However, the existing researches mostly ignore them. Meanwhile, most existing methods also ignore the efficiency of the XSS attack detection system. Few researchers have paid attention to the spent time processing a large amount of data. Besides, the generalization ability of XSS attack detection methods is an issue. These detection methods often perform well on an experimental dataset, but do not work properly on another dataset. This greatly limits the applicability of machine learning and deep learning based XSS attack detection methods.

The automatic clustering property of GCN is helpful to solve the above problem. In the field of NLP, the results of graph embedding using GCN are automatically clustered in space without any training. Based on graph embedding using GCN, labeled data is used to train GCN and the results will be expected.

In this research, with the help of the GCN auto-clustering feature and labeled data, an XSS attack detection approach (called GCNXSS) is proposed to mine the dependencies between

the units that constitute an XSS payload and solve the above problems. After data preprocessing, GCNXSS converts a URL into a word homogeneous graph based on word co-occurrence relationships. Then GCNXSS inputs the graph into the GCN model for graph embedding and classification.

Various metrics were used to evaluate the proposed method experimentally in this research. The proposed method achieves advanced results on the test dataset. GCNXSS based on word homogeneous graph achieved the best results with accuracy, precision, recall, F1-score, FNR, FPR, and spent time scores of 99.97%, 99.75%, 99.96%, 99.86%, 0.03%, 0.03%, and 0.0461ms. The main contributions are as follows.

- This research proposes an approach for XSS attack detection based on GCN (called GCNXSS) and formulates the XSS attack detection problem as a graph classification task over word homogeneous graphs.
- This research proposes an approach for converting URLs into word homogeneous graphs based on word co-occurrence relationships, enabling GCN to perform graph embedding and mine the dependencies between the units that constitute an XSS payload.
- This research experimentally evaluates the proposed method on the test dataset using various metrics such as precision, recall, FNR, FPR, etc., and compares it with other machine learning methods. Besides, using all the data from the difficult dataset as the generalized dataset, the method also performs well on the generalized dataset.

The remainder of this paper is systematized as follows. Section 2 discusses related work and analyzes the shortcomings of previous studies. Section 3 offers the key details about the proposed method, including preprocessing, graph construction, GCN model, and classifier. Section 4 presents the experimental design, the results of model parameter optimization, and the comparative experimental results of this research. Section 5 discusses and analyzes the results of the experiments. Section 6 concludes this research, focusing on its significance and highlighting key future research directions.

2. Related Work

XSS detection has always been an important research field in Web security. Many researchers have conducted much research on this field in the past ten years and published many research results. These research results can be divided into XSS vulnerability mining and XSS attack detection.

In terms of XSS vulnerability mining, the principle is mainly to discover the vulnerable points of XSS vulnerabilities through mining to fix the corresponding defects as comprehensively as possible. XSS vulnerability mining methods are further divided into static, dynamic, and hybrid analysis based on detection and analysis methods. Static analysis is the source code analysis, which has the advantage of detecting vulnerabilities that an attacker may exploit without executing the application. To detect XSS vulnerabilities in Web applications, Pixy, a static analysis tool for the PHP language, was introduced by Nenad et al. [7]. However, not all security problems come from the source code, but also from software design flaws. Static analysis would not be able to find vulnerabilities that require a deep understanding of the code structure or design, and also has a high false positive rate.

On the other hand, dynamic analysis is based on simulated attacks, and its focus and difficulty lie in generating attack vectors. The quality of the generated attack vectors will directly affect the results of vulnerability mining. Bernhard et al. use the dependencies of the

input parameters to obtain the syntax of the attack vectors and then use the combined testing method to generate the structured attack vectors [8]. Mahmoud et al. propose a syntax-based attack generator to automate the generation of XSS test inputs to evaluate XSS vulnerabilities in target web pages due to mishandling of data encoding [9]. However, dynamic analysis cannot find all vulnerabilities because some of the components are not running at all.

On the other hand, the hybrid analysis combines static and dynamic analysis. William et al. propose a lightweight hybrid approach for detecting DOM-type XSS vulnerabilities with 3.43 times lower computational overhead and detects 94.5% of the vulnerabilities [10]. However, numerous solutions have been proposed, but no single solution can altogether remove the flaws present in the program source code [11].

In terms of XSS attack detection, the principle is mainly to detect whether the user behavior is abnormal to filter out the possible XSS attacks. Client-side detection methods, server-side detection methods and client-server detection methods are the results of the mainstream classification of XSS detection methods. Moreover, machine learning is introduced into the field of XSS attack detection and is heavily used by many researchers. For client-side detection, many researchers have embedded rule-based detector into the front end to filter out a large number of missteps. The XSS Auditor proposed by Danielet al. is used by Google Chrome for many users [12]. Riccardo et al. use a rule-based algorithm and a set of policies to detect XSS attacks [13]. Shashank et al. propose a Google Chrome extension based on contextual dependencies for detecting XSS attacks [14]. Detectors deployed on the client side are useful in that they filter out incorrect input from normal users. But for an attacker with ulterior motives, these detectors can be easily bypassed. Therefore, it is not enough to use client-side detection to defend against XSS attacks, but server-side detection is also required.

For server-side detection, the detector is typically deployed on the server of the Web application. Martin et al. detect reflected XSS attacks by examining the input data and output data [15]. Zhou et al. present a Bayesian network-based attack detection method [5]. The method uses threat intelligence and domain knowledge to construct Bayesian networks and proposes an analysis method to interpret the detection results further. Experiments show that the method outperforms other methods such as SVM, random forest, and decision tree in most cases. Its accuracy reaches 98.54%. However, it is not validated using other performance metrics. Iram et al. propose a detection method using genetic algorithms, statistical inference, and reinforcement learning [16]. This approach applies genetic algorithms, which are widely used in the static analysis, to XSS attack detection, with statistical inference results used to determine the state of vulnerabilities present. It then uses reinforcement learning to adapt to unknown XSS attacks. Its accuracy, precision, recall and F1 values are 99.67%, 99.50%, 99.56%, 99.52% respectively. Fang et al. propose a method called DeepXSS, which is based on circular decoding and LSTM model to extract XSS features for training [17]. Its precision, recall, and F1 values are 99.5%, 97.9%, and 98.7%, respectively. Again, these two articles do not mention FPR, FNR, and running time. Mohammed et al. propose a hybrid feature selection method based on IG and SBS and uses GBDT integrated learning technique for XSS detection, which can provide higher accuracy and detection rate [18]. In addition, a comparison is made with other methods for training time and testing time. The research provides a more comprehensive test of the proposed method with accuracy=99.59%, precision=99.50%, recall=99.02%, false-positive rate=0.20%, false-negative=0.98%, and AUC score= 99.41%.

Client-side detection and server-side detection are not mutually exclusive; in practice, the two are typically used jointly in deployments. For client-server detection, Trevor et al. resist XSS attacks by embedding a policy in the web page [19]. The policy only supports the browser to run fixed scripts, while the others cannot be executed. This policy can be ideally enforced

by browsers that know when to run scripts. Wassermann et al. propose a method for detecting XSS vulnerabilities based on detecting input data [20]. The method defends against XSS attacks by detecting whether user input causes the browser's JavaScript engine to be invoked. Van Gundy et al. propose a method to distinguish unreliable content in Web pages by randomizing the HTML tags and attributes in each Web page, called Noncespaces [21]. Trusted content in Web pages can be easily distinguished by the client from unreliable content constructed by the attacker as long as the random mapping is not broken by the attacker.

GCN may be used to solve the problems mentioned above. GCN is a generalized form of CNN that extends convolution to graphs while using several convolutional layers instead of circular iterations of the original GNN to achieve convergence of the whole graph. Thus, GCN is a deep neural network that can learn directly on graph data, enabling the extraction of information about the graph structure in addition to semantic features. GCN has an excellent performance in computer vision, natural language processing, program verification, and program reasoning [22]. In particular, in the field of NLP, GCN has an automatic clustering property, and the results of graph embedding using GCN can be automatically clustered in space without any training [23]. XSS attack detection is similar to text classification tasks in the field of NLP, and this property of GCN is useful. Moreover, on the basis of graph embedding using GCN, the expected results are obtained by training the GCN using labeled data.

At present, GCN is divided into spectral-based GCN and spatial-based GCN. The basic principle of spectral-based GCN is to treat the graph as a signal for processing, and the convolution process is the process of noise removal for the signal using filters. At present, the more representative spectral-based GCN models are ChebNet [24], 1stChebNet [23], Spectral CNN [25] and so on. The idea of spatial-based GCN is based on information propagation over space. The more representative spatial-based GCN models are GraphSAGE [26], MPNN [27], PATCHY-SAN [28], DCNN [29], etc. Spatial-based GCN has received more attention from researchers compared to spectral-based GCN. The main reason is that spectral-based GCN requires a full graph Laplacian function, which becomes very difficult to handle for large graphs. In contrast, spatial-based GCN, based on the spatial spread of information, can process some nodes without processing the whole graph at once, which has higher efficiency, flexibility, and versatility.

3. Methodology

Receiving inspiration from GCN research results in the field of NLP [23, 30], this research proposes an XSS attack detection method based on GCN called GCNXSS. The proposed approach proceeds as follows. First, GCNXSS performs preprocessing on the data, including generalization and tokenization. A URL is divided into a set of words. Then, GCNXSS performs graph construction, which transforms a set of words into a word homogeneous graph based on word co-occurrence relationships. This part is divided into edge relation extraction, which extracts PMI from the preprocessed data as the edge weight of the graph, and word embedding, which obtains the node features (Word2Vec) of the graph through training. Finally, GCNXSS inputs the constructed graph into GCN for graph embedding and trains a classifier to distinguish XSS samples from benign samples. **Fig. 1** shows the framework of the proposed method.

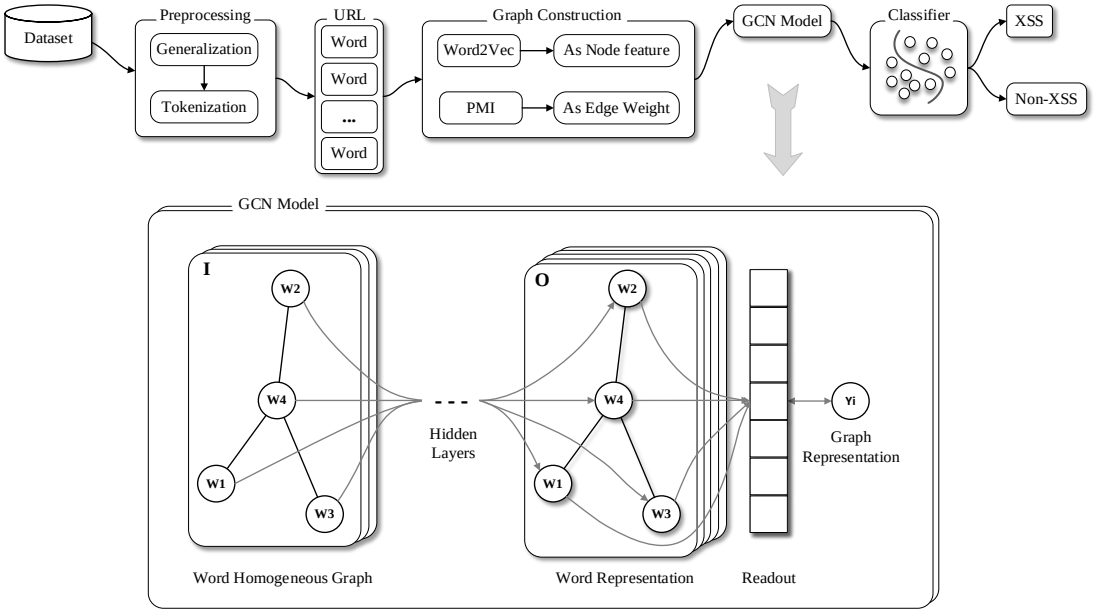

Fig. 1. The framework of the proposed method

3.1 Preprocessing

3.1.1 Generalization

The confrontation between attackers and defenders has been going on for many years. Attackers try to bypass the defenses of security systems by encoding the code, inserting useless HTML tags and parameters into the payload, etc. In addition, domain information and numeric information in the payload is not helpful in detecting XSS attacks. As a result, the input data contains much redundant information. To reduce redundant information, this research uses the following measures to generalize the data: First, all characters in the input data are changed to lowercase. Then, the input data is decoded. Moreover, all the digits in the input data are replaced with '0'. Finally, all URLs in the input data are replaced with 'http://u'.

3.1.2 Tokenization

XSS payloads are constructed with certain rules, and the pre-processing phase requires segmenting its constituent elements, such as function names and tag names, for feature extraction. In order to tokenize the input data after generalization, the rules of the tokenization is shown below:

- The context between ' and '', such as 'attack'.
- URL, such as http and https.
- Script label, such as <script>.
- Start label, such as <h1.
- Function name, such as topic=.
- Function body, such as alert(.
- Words make up of alphanumeric characters, such as user12.

After this operation, each sample in the input data will be divided into a set of segments called words. Some examples are shown in **Table 1**.

Table 1. Some examples of preprocessing

Origin URL	Words
site=message&msg=<script>alert(1)</script>	site= message msg= <script> alert(0) </script>
q=%3Ciframe+src%3D%22http%3A%2F%2Fxssed.com%22%3E	q= <iframe src= http://u >
Itemid=%22onmouseover=alert%28document.cookie%29%20bad=%22	itemid= onmouseover= alert(document.cookie) bad=
sid=&ring=hentff98&id=&list usernum=3614006703	sid= amp ring= hentff0 amp id= amp list usernum= 0

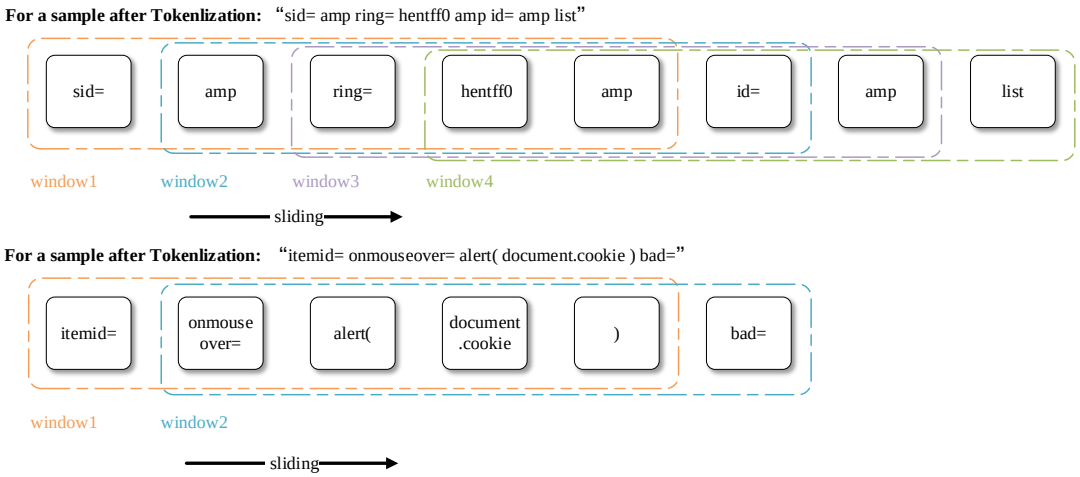

Fig. 2. Some examples of sliding window

3.2 Word Homogeneous Graph Construction

Word homogeneous graph is a homogeneous graph with words as nodes after operations in Section 3.1.2. A sample, whether it is an XSS malicious sample or a benign sample, will be transformed into a word homogeneous graph, as shown in **Fig. 1**. The number of nodes in the word homogeneous graph is the number of words in a sample. The graphs use Word2Vec [31] as the feature of the nodes. This research also found using Word2Vec [31] achieves better results than using GloVe [32] in the preliminary experiments. Each word embedding vector corresponds to a node feature. Point-wise mutual information (PMI) is used to measure the semantic correlation before two word nodes in the graph. When the PMI value is less than or equal to 0, then there is no correlation between the corresponding two word nodes. When the PMI value is greater than 0, then the corresponding two word nodes possess correlation. The higher the correlation, the larger the value of PMI; the lower the correlation, the smaller the value of PMI. In this research, only edges are added between the two word nodes that possess correlation. In addition, PMI will be calculated by using a sliding window of fixed size, as shown in **Fig. 2**. The PMI is calculated as shown in the following equation.

$$\mathrm{PMI}(i,j) = \log \frac{p(i,j)}{p(i)p(j)} \tag{1}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{2}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{3}$$

where *#W(i)* is the number of sliding windows in the dataset that contain word i, *#W(i,j)* is the number of sliding windows that contains both words i and j, and *#W* is the total number of sliding windows in the dataset.

3.3 GCN Layer

For the word homogeneous graph, this research chooses 1stChebNet [23], the most representative model in spectral-based GCN, for graph classification. According to the result in the literature [23, 30, 33], the performance of a two-layer GCN is good enough as far as feature extraction is concerned. GCN with three, or even more, layers do not bring additional performance improvement. Therefore, a two-layer GCN is able to extract enough features from the constructed word isomorphism graph for downstream tasks. Moreover, python package DGL has encapsulated the graph convolution proposed in the literature [23], the simple two-layer GCN is easily implemented. In addition, the input dimension is the word vector dimension, the hidden layer dimension is 20, and the output dimension is the same size as the label set (XSS, Non-XSS).

3.4 Classifiers

After the graph embedding is done from the graph data using GCN, the graph representation is fed to a classifier for the classification task. In this article, GCNXSS only uses a simple binary classifier as shown in **Algorithm 1**. If the detection results of the proposed method are better than existing methods in this case, then it is more indicative of the usability of GCN in the field of XSS attack detection.

Algorithm 1: How to classify (hardmax)

```
Input: The array of features for every sample, features
Output: The array of labels for every feature, labels
labels is initialized to a null array ;
for item in features do
        if item[0] > item[1] then
            labels.add('Non-XSS');
        else
            labels.add('XSS');
        end
end
return labels;
```

4. Experiment

This section introduces the datasets and metrics, and then conducts a series of experiments to answer the following research questions:

- **RQ 1.** With the help of GCN and labeled data, does GCNXSS effectively detect XSS attacks and solve problems such as FNR and FPR?
- **RQ 2.** Does GCNXSS outperform existing XSS attack detection methods?

ALL experiments were operated on the PyCharm Community Edition 2021.1.1 x64 platform. The computer used to run the program had 1 CPU of i5-10600KF with 4.10GHZ

and 16GB RAM. At the same time, we program in Python 3.6.13, Pytorch1.5.1, pandas 1.1.5, NumPy 1.19.2, and DGL 0.7.1 software environment.

4.1 Datasets

This research used 151,658 samples collected from the GitHub repository(https://github.com/duoergun0729/1book/tree/master/data), including 16,151 XSS malicious samples and 135,507 benign samples. The dataset is divided into training, validation, and testing datasets in the ratio of 6:2:2. **Table 2** shows the detailed partitions of the datasets. In addition, this research merged the dataset on the Kaggle platform (https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning) with the dataset used in the literature [34] to construct a difficult dataset containing 13,726 samples, with 6,312 benign samples and 7,414 malicious samples. The data for the malicious samples were obtained from OWASP, PortSwigger, and html5sec.org. The difficult dataset was used to perform generalization experiments.

Table 2. Dataset subdivision

Name	Benign	Malicious	Total
Training dataset	81,304	9,690	90,994
Validation dataset	27,101	3,230	30,331
Testing dataset	27,102	3,231	30,333
Total dataset	135,507	16,151	151,658

Table 3. Confusion matrix

	Actual XSS	Actual non-XSS
Predicted XSS	TP	FP
Predicted Non-XSS	FN	TN

4.2 Evaluation Metrics

In this research, the proposed method's performance is employed to analyze the methods using accuracy, precision, recall, F1-score, FNR, and FPR. The malicious sample was denoted as the positive (P) class, and the benign sample was denoted as the negative (N) class. These metrics are computed based on a confusion matrix (see **Table 3**), and they are defined as:

$$\text{accuracy} = \frac{TP+TN}{TP+FN+FP+TN} \tag{4}$$

$$\text{precision} = \frac{TP}{TP+FP} \tag{5}$$

$$\text{recall} = \frac{TP}{TP+FN} \tag{6}$$

$$\text{f1} - \text{score} = \frac{2*\text{precison}*\text{recall}}{\text{precison}+\text{recall}} \tag{7}$$

$$\text{FNR} = \frac{FN}{TP+FN} = 1 - \text{recall} \tag{8}$$

$$\text{FPR} = \frac{FP}{FP+TN} \tag{9}$$

4.3 Experimental Results Analysis

To verify whether GCN can extract features of XSS attacks, this research performs a preliminary experiment using randomly initialized GCN parameters based on fixed composition-related parameters. To enhance the convincing power, this research uses a real

dataset (https://github.com/das-lab/deep-xss) from the GitHub repository containing 64,833 real samples in addition to experiments on the test dataset. After only one training, the results are very satisfactory, as shown in **Table 4**. It is easy to see that the accuracy of GCNXSS based on testing dataset is close to 0.98 after one training, and the accuracy of GCNXSS based on real dataset is close to 0.97. Moreover, all other metrics have good performance. This can deduce that GCN is able to aggregate XSS feature information.

Table 4. Performance of the proposed method in the preliminary experiment after one training

Dataset	Accuracy	Precision	Recall	F1-score	FNR	FPR
Testing dataset	0.9827	0.9093	0.9307	0.9198	0.0693	0.0111
Real dataset	0.9705	0.9674	0.9758	0.9716	0.0242	0.0350

Moreover, this research also found using homogeneous graphs achieves a better embedding result than using heterogeneous graphs in the preliminary experiments. It is expected that the reason for this situation is that too many elements in the heterogeneous graphs make it more difficult for graph embedding based on GCN.

To obtain better performance, the parameters of GCNXSS are optimized and trained. When the epoch is set to 50, the window size is set to 20, the word vector dimension is set to 300, the word embedding method is set to Word2Vec CBOW, the loss function is set to cross-entropy, the optimizer is set to Adam, and the rest of the parameters by default, the performance results of GCNXSS are optimal, as shown in **Table 5**. From **Table 5**, GCNXSS has promising results with low FNR and FPR. Besides, the accuracy, precision, recall, and F1-score of GCNXSS also improves. It is easy to see how the labeled data can help to overcome the problem of significant FNR and FPR on the basis of GCN.

Table 5. Performance of the proposed method based on different datasets after training

Dataset	Accuracy	Precision	Recall	F1-score	FNR	FPR	Predicted Time (ms)
Testing dataset	0.9997	0.9975	0.9997	0.9986	0.0003	0.0003	0.0461
Real dataset	0.9931	0.9979	0.9886	0.9932	0.0114	0.0022	0.0489

In addition, this research also tested the final model on the real dataset, as shown in **Table 5**. The performance of GCNXSS on the real dataset is still good, and the reduction in the metrics is within acceptable limits compared to the performance on the testing dataset.

Answer to **RQ 1**: The experimental results prove that GCNXSS has promising performance with low FNR and FPR. With the help of GCN and labeled data, GCNXSS can effectively detect XSS attacks.

4.4 Comparison with Other Detectors

To demonstrate the advancement of the methods proposed in this research, this research uses support vector machines (SVM), decision tree (DR), logistic regression (LR), k-nearest neighbor (KNN), random forest (RF), naive bayes, and adaptive boosting (Adaboost) which are machine learning methods compared with GCNXSS. The features used by these machine learning methods are those proposed in the literature [5]. These features are broadly classified into four categories: input length, sensitive characters, sensitive words, and redirected links. These are also the main difference between XSS payload and normal data.

Table 6. Comparison results with other machine learning detectors

Detector	Accuracy	Precision	Recall	F1-score	FNR	FPR	Predicted Time (ms)
SVM	0.9942	0.9881	0.9573	0.9724	0.0427	0.0014	0.8467
Decision Tree	0.9955	0.9951	0.9622	0.9784	0.0378	0.0006	0.0422
Logistic Regression	0.9990	0.9928	0.9979	0.9954	0.0021	0.0009	0.0423
KNN	0.9987	0.9957	0.9924	0.9940	0.0076	0.0005	0.6986
Random Forest	0.9920	0.9936	0.9309	0.9612	0.0691	0.0007	0.0427
Naive Bayes	0.9916	0.9279	0.9988	0.9620	0.0012	0.0092	0.0426
Adaboost	0.9993	0.9969	0.9967	0.9968	0.0033	0.0004	0.0571
GCNXSS	0.9997	0.9975	0.9997	0.9986	0.0003	0.0003	0.0461

Table 7. Comparison results with previous detectors

Detector	Accuracy	Precision	Recall	F1-score	FNR	FPR	Predicted Time (ms)
Fang et al., 2018 [17]	0.9896	0.9966	0.9832	0.9898	0.0168	0.0036	0.2678
Mokbal et al., 2021 [18]	0.9966	0.9958	0.9725	0.9840	0.0275	0.0005	0.0532
GCNXSS	0.9997	0.9975	0.9997	0.9986	0.0003	0.0003	0.0461

The results of the comparison on the testing dataset are shown in **Table 6**. Traditional machine learning methods almost always have a high accuracy, precision, recall, and F1-score. At this point, GCNXSS only has a weak advantage. However, traditional machine learning methods is slightly deficient in FPR and FNR. In contrast, GCNXSS is more advantageous because its FPR and FNR are close to 0. And the predicted time is also in the top. In terms of performance metrics, GCNXSS is better compared to traditional machine learning methods.

To further assess the advantages of the proposed GCNXSS, this research compared with the detectors proposed by the literature [17, 18] on the testing dataset. The literature [17] extracts the features of XSS payloads based on recurrent decoding and word2vec, and trains them using Long Short-Term Memory (LSTM) recurrent neural networks. The literature [18] proposes a hybrid feature selection method based on IG and SBS with GBDT integrated learning technique for XSS detection. The comparison results are shown in **Table 7**. As with traditional machine learning methods, these detectors are also slightly deficient in FPR and FNR, although they have high accuracy, precision, recall, and F1-score. From **Table 7**, it is can see that GCNXSS has a slight advantage over the previous detectors.

To further demonstrate the advantages of the GCNXSS method, this research conducts generalization experiments on the difficult dataset. The detectors that appear in **Table 6** and **Table 7** use a model trained based on the testing dataset with all data from the difficult dataset as input. The results of the generalization experiments are shown in **Table 8** and **Table 9**, which also be seen in **Fig. 3** and **Fig. 4**.

Table 8. Results of generalization experiments with other machine learning detectors

Detector	Accuracy	Precision	Recall	F1-score	FNR	FPR
SVM	0.6356	0.5975	0.9909	0.7455	0.0091	0.7795
Decision Tree	0.6590	0.6138	0.9898	0.7577	0.0102	0.7274
Logistic Regression	0.5920	0.5698	0.9908	0.7235	0.0092	0.8738
KNN	0.6414	0.6016	0.9900	0.7484	0.0100	0.7657
Random Forest	0.5955	0.5720	0.9896	0.7250	0.0104	0.8647
Naive Bayes	0.5503	0.5451	0.9984	0.7052	0.0016	0.9731
Adaboost	0.6618	0.6158	0.9901	0.7593	0.0099	0.7215
GCNXSS	0.8119	0.7583	0.9610	0.8477	0.0390	0.3665

Table 9. Results of generalization experiments with previous detectors

Detector	Accuracy	Precision	Recall	F1-score	FNR	FPR
Fang et al., 2018 [17]	0.6548	0.6138	0.9879	0.7572	0.0121	0.7436
Mokbal et al., 2021 [18]	0.5942	0.5715	0.9864	0.7237	0.0136	0.8639
GCNXSS	0.8119	0.7583	0.9610	0.8477	0.0390	0.3665

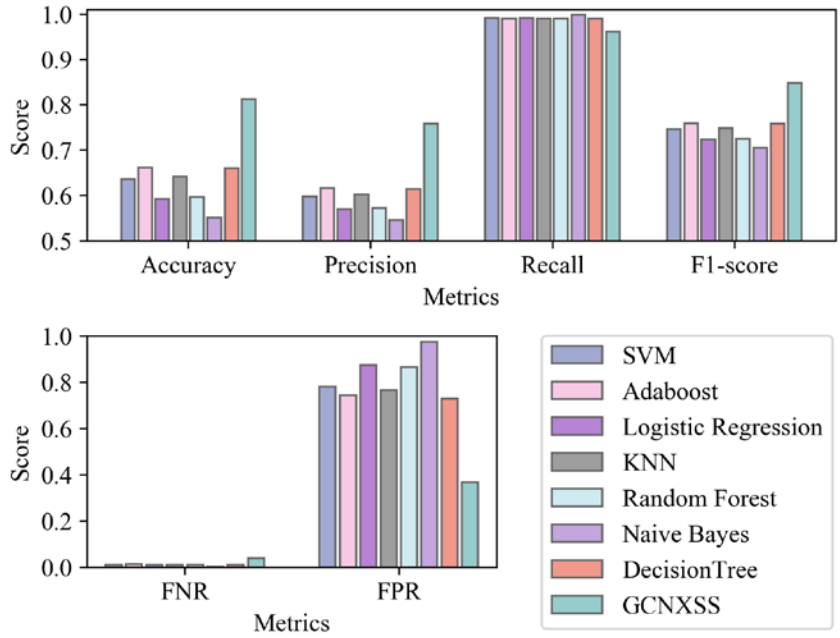

Fig. 3. Results of generalization experiments with other machine learning detectors

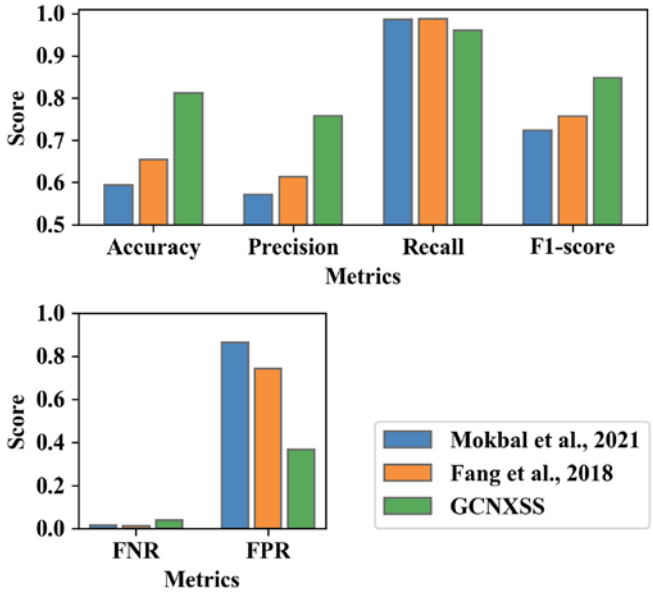

Fig. 4. Results of generalization experiments with previous detectors

After the generalization experiments, it is easy to see that GCNXSS has significant advantages in all performance measures except for the recall and FNR, which do not differ much from other detectors. It can be concluded that GCNXSS has stronger generalization ability than other detectors and has a clear advantage in FPR.

Answer to **RQ 2:** Traditional machine learning methods and previous detectors face poor generalization ability, significant FPR and FNR problems, although they have high accuracy, precision, recall, and F1-score. This research can conclude that GCNXSS has a stronger generalization ability with low FNR and FPR compared to the other methods.

5. Discussion

This research explores the application of GCN in XSS attack detection. Based on the experimental results, this research clarifies the conclusion that GCN can indeed be applied in the field of XSS attack detection. However, although the GCN is in the lead with other machine learning models in terms of predicted time, the time required for graph construction in the experiments is much greater than the time required for feature extraction. From this point on, optimizing composition methods is a subsequent priority in this area. In addition, most of the current XSS attack detection methods only perform well on the experimental datasets. Once the data source is changed, the performance of these methods will be greatly reduced. Using more advanced methods with better quality datasets may be able to help solve this problem. Besides, the advanced results obtained in this research using only GCN extraction results and annotation information, and the performance of the proposed method will be even better after using more complex classifiers and introducing attention mechanisms on top of this method.

6. Conclusion

This research proposes an XSS attack detection method based on GCN called GCNXSS. First, GCNXSS performs a preprocessing operation for the data. Then, GCNXSS performs graph construction, which transforms a URL into a word homogeneous graph based on word co-occurrence relationships. Finally, GCNXSS inputs the constructed graph into GCN for graph embedding and classification. In the experiments, this research first verifies the ability of GCN to extract the features of XSS attacks through preliminary experiments. Finally, the superiority of GCNXSS is demonstrated by comparing it with existing XSS attack detection methods. In the future, the research task is to explore more composition methods in the field of XSS attack detection. The direction of exploration is to construct graph structures that can contain more XSS payload features as a way to increase the generalization ability of detection methods and reduce detection time. In addition, because GCN shows good performance in the field of XSS attack detection, another research task is to apply GCN to more fields of Web attack detection, such as SQL injection detection, JavaScript malicious code detection, and so on.

References

[1] Precise Security, "Cross-Site Scripting (XSS) Makes Nearly 40% of All Cyber Attacks in 2019," Website, 2020. [Online]. Available: https://www.precisesecurity.com/articles/cross-site-script ing-xss-makes-nearly-40-of-all-cyber-attacks-in-2019/

[2] OWASP, "OWASP top 10 - 2017 The Ten Most Critical Web Application Security Risks," Website, 2017. [Online]. Available: https://www.owasp.org/images/7/72/OWASP_To p_10-2017_(en).pdf

[3] J. Fonseca, N. Seixas, M. Vieira, and H. Madeira, "Analysis of Field Data on Web Security Vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 2, pp. 89-100, 2014. Article (CrossRef Link)

[4] OWASP, "OWASP top 10 - 2021 The Ten Most Critical Web Application Security Risks," Website, 2021. [Online]. Available: https://owasp.org/Top10/

[5] Y. Zhou and P. Wang, "An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence," *Computers & Security*, vol. 82, pp. 261-269, 2019. Article (CrossRef Link)

[6] U. Sarmah, D. K. Bhattacharyya, and J. K. Kalita, "A survey of detection methods for XSS attacks," *Journal of Network and Computer Applications*, vol. 118, pp. 113-143, 2018. Article (CrossRef Link)

[7] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper)," in *Proc. of IEEE Computer Society*, USA, pp. 258–263, 2006. Article (CrossRef Link)

[8] D. E. Simos, B. Garn, J. Zivanovic, and M. Leithner, "Practical Combinatorial Testing for XSS Detection using Locally Optimized Attack Models," in *Proc. of ICSTW*, pp. 122-130, 2019. Article (CrossRef Link)

[9] M. Mohammadi, B. Chu, and H. R. Lipford, "Detecting cross-site scripting vulnerabilities through automated unit testing," in *Proc. of QRS*, pp. 364-373, 2017. Article (CrossRef Link)

[10] W. Melicher, C. Fung, L. Bauer, and L. Jia, "Towards a Lightweight, Hybrid Approach for Detecting DOM XSS Vulnerabilities with Machine Learning," in *Proc. of the Web Conference 2021*, pp. 2684-2695, 2021. Article (CrossRef Link)

[11] G. Deepa and P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Information and Software Technology*, vol. 74, pp. 160-180, 2016. Article (CrossRef Link)

[12] D. Bates, A. Barth, and C. Jackson, "Regular Expressions Considered Harmful in Client-Side XSS Filters," in *Proc. of ICWWW*, New York, NY, USA, pp. 91–100, 2010. Article (CrossRef Link)

[13] R. Pelizzi and R. Sekar, "Protection, Usability and Improvements in Reflected XSS Filters," in *Proc. of ASIACCS*, New York, NY, USA, p. 5, 2012. Article (CrossRef Link)
[14] S. Gupta and B. B. Gupta, "XSS-immune: A Google chrome extension-based XSS defensive framework for contemporary platforms of web applications," *Security and Communication Networks*, vol. 9, pp. 3966-3986, 2016. Article (CrossRef Link)
[15] M. Johns, B. Engelmann, and J. Posegga, "XSSDS: Server-Side Detection of Cross-Site Scripting Attacks," in *Proc. of ACSAC*, pp. 335-344, 2008. Article (CrossRef Link)
[16] I. Tariq, M. A. Sindhu, R. A. Abbasi, A. S. Khattak, O. Maqbool, and G. F. Siddiqui, "Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning," *Expert Systems with Applications*, vol. 168, p. 114386, 2021. Article (CrossRef Link)
[17] Y. Fang, Y. Li, L. Liu, and C. Huang, "DeepXSS: Cross site scripting detection based on deep learning," in *Proc. of ICCAI*, pp. 47-51, 2018. Article (CrossRef Link)
[18] 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. Article (CrossRef Link)
[19] T. Jim, N. Swamy, and M. Hicks, "Defeating Script Injection Attacks with Browser-Enforced Embedded Policies," in *Proc. of ICWWW*, New York, NY, USA, pp. 601–610, 2007. Article (CrossRef Link)
[20] G. Wassermann and Z. Su, "Static Detection of Cross-Site Scripting Vulnerabilities," in *Proc. of ICSE*, New York, NY, USA, pp. 171–180, 2008. Article (CrossRef Link)
[21] M. Van Gundy and H. Chen, "Noncespaces: Using randomization to defeat cross-site scripting attacks," *Computers & Security*, vol. 31, no. 4, pp. 612-628, 2012. Article (CrossRef Link)
[22] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4-24, 2021. Article (CrossRef Link)
[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844-3852, 2016. Article (CrossRef Link)
[25] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.
[26] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017. Article (CrossRef Link)
[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. of ICML*, pp. 1263-1272, 2017. Article (CrossRef Link)
[28] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. of ICML*, pp. 2014-2023, 2016. Article (CrossRef Link)
[29] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *Advances in neural information processing systems*, vol. 29, 2016.
[30] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proc. of AAAI*, vol. 33, pp. 7370-7377, 2019. Article (CrossRef Link)
[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. of EMNLP*, pp. 1532-1543, 2014. Article (CrossRef Link)
[33] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. of AAAI*, 2018. Article (CrossRef Link)
[34] G. Xu, X. Xie, and S. Huang, "JSCSP: A Novel Policy-Based XSS Defense Mechanism for Browsers," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 862-878, 2022. Article (CrossRef Link)

Hongyu Pan received the B.Eng. degree in cyberspace security from Sichuan University, Chengdu, China, in 2021, where he is currently pursuing the master's degree with the School of Cyber Science and Engineering. His current research interests include Web security and artificial intelligence.

Yong Fang received the Ph.D degree from Sichuan University, Chengdu, China, in 2010. He is currently a Professor with School of Cyber Science and Engineering, Sichuan University, China. His research interests include network security, Web security, Internet of Things, Big Data and artificial intelligence.

Cheng Huang received the Ph.D degree from Sichuan University, Chengdu, China, in 2017. From 2014 to 2015, he was a visiting student at the School of Computer Science, University of California, CA, USA. He is currently an Associate Professor at the School of Cyber Science and Engineering, Sichuan University, Chengdu, China. His current research interests include Web security, attack detection, artificial intelligence.

Wenbo Guo received the B.Eng. degree in cyberspace security from Sichuan University, Chengdu, China, in 2020, where he is currently pursuing the master's degree with the School of Cyber Science and Engineering. His current research interests include Web security and artificial intelligence.

Xuelin Wan received the M.S. degree from Peking University, Beijing, China, in 2010. He is senior engineer working at China Merchants Bank, Shenzhen, China. His current research interests include network security, machine learning and attack detection.