Journal Pre-proof

GraphXSS: An Efficient XSS Payload Detection Approach Based on Graph Convolutional Network

Zhonglin Liu, Yong Fang, Cheng Huang, Jiaxuan Han

PII: S0167-4048(21)00420-X
DOI: https://doi.org/10.1016/j.cose.2021.102597
Reference: COSE 102597

To appear in: *Computers & Security*

Received date: 12 October 2021
Revised date: 8 December 2021
Accepted date: 23 December 2021

Please cite this article as: Zhonglin Liu, Yong Fang, Cheng Huang, Jiaxuan Han, GraphXSS: An Efficient XSS Payload Detection Approach Based on Graph Convolutional Network, *Computers & Security* (2021), doi: https://doi.org/10.1016/j.cose.2021.102597

GraphXSS: An Efficient XSS Payload Detection Approach Based on Graph Convolutional Network

Zhonglin Liu
School of Cyber Science and Engineering
Sichuan University
Chengdu, Sichuan,
P.R.China
jungleforsa@gmail.com

Yong Fang
School of Cyber Science and Engineering
Sichuan University
Chengdu, Sichuan,
P.R.China
yfang@scu.edu.cn

Cheng Huang*
School of Cyber Science and Engineering
Sichuan University
Chengdu, Sichuan,
P.R.China
opcodesec@gmail.com

Jiaxuan Han
School of Cyber Science and Engineering
Sichuan University
Chengdu, Sichuan,
P.R.China
zhanSxDrive30i@gmail.com

ABSTRACT

With the rapid development of the Internet age today, Web applications have become very common in modern society. Web applications are often applied to a social network, media, management, etc., and usually contain a large amount of personal privacy information, which makes Web applications a common target for hackers. The most common method for stealing private information from web applications is cross-site scripting attacks. Attackers frequently use cross-site scripting vulnerabilities to steal victims' identity information or hijack login tokens. Therefore, we proposed a cross-site scripting payload detection model based on graph convolutional networks, which could identify the cross-site scripting payload in the content submitted by the user (We termed our implementation of this approach, GraphXSS). We preprocessed the sample, and constructed the processed data into a graph structure, and finally used the graph convolutional network and the residual network to train the cross-site scripting detection model. In experiments, the model based on graph convolutional network (GCN) could achieve AUC value of 0.997 under small sample conditions. Compared with the detection model after adding the residual network structure, the model could converge and stabilize under the multi-layer, and could make the accuracy rate reached 0.996.

KEYWORDS

Cross-site Scripting; Detection; Graph Convolutional Network; Residual network

1 INTRODUCTION

With the advancement of computer technology, the types of script-type programming languages are gradually increasing, and at the same time, more and more security problems are brought about. Cross-site scripting (Abbreviated to XSS) attack is a very common script injection attack, which enables the victim's browser to load and execute malicious JavaScript script after visiting a website or page containing malicious code, thereby being controlled and attacked by malicious script code [1-2].

Attackers often use malicious script codes to hijack the victimâĂŹs browser cookies and get sensitive backstage login addresses to steal usersâĂŹ private information on the Internet or unauthorized login to websites managed by the victim. In addition, the attacker can also exploit XSS to steal the victim's browser information, operating system information, order information on shopping websites, phone numbers, IP addresses, virtual identities, etc., which is very harmful [3-4]. Even the attacker can control the attacked browser to achieve DDOS (Distributed Denial of Service), CSRF (Cross-Site Request Forgery), RCE (Remote Command/Code Execute), and other network attacks against specific targets, resulting in a series of security problems on the attacked host and its related network. A few years ago, Renren worm [5], Samy worm [6], Boonana worm [7], Spaceflash worm [8], and other incidents were all due to the fact that the web application did not detect and process the content submitted by users, which led to the existence of XSS vulnerabilities in the application. The vulnerability made the worm attack effective and had a wide range of impacts, serious information leakage, and immeasurable economic loss.

XSS has various attack modes. Generally, XSS is divided into three categories: Persistent attacks (Storage XSS), non-persistent attacks (Reflected XSS), and DOM-based XSS attacks [9]. XSS vulnerabilities are very easy to exploit, and it's easy for attackers to change the form and structure of XSS attacks, such as by using complex coding, changing sentences, and using browser features to bypass some security detection methods on the server [10]. Therefore, it was very difficult to detect and defend against XSS attacks in a traditional method.

As there are many ways to exploit vulnerabilities, researchers in the past have continuously improved their detection methods. In the early days, researchers generally used regular expressions to identify XSS payloads. But people soon discovered that this method could not match the complex XSS payload. Therefore, researchers began to construct a feature library to identify the payload. This method can match the payload through many features in the feature library, but it cannot accurately identify the payload that has not been collected. In order to improve the ability to recognize

unknown samples, in recent years, researchers have begun to try to use machine learning to solve this problem, such as support vector machines (SVM), convolutional neural networks (CNN), long and short-term memory networks (LSTM), etc. After comparative analysis, we found that deep learning can indeed identify unknown samples more accurately, but the prerequisite is to train the model through many known samples, otherwise, the model will not work well.

In this paper, we proposed an advanced XSS payload detection method based on graph convolutional networks, called GraphXSS. Our main contributions are as follows:

- By learning from the word segmentation method in natural language processing, we divided the preprocessed payload data into multiple characters according to the character structure through regular expressions, and treated each character as a word for subsequent construction of the graph structure.
- We used the relationship between each piece of payload data and words to construct the entire sample data into a graph with multiple vertex and edges. The classification of point types was achieved through deep learning of the graph.
- We designed an advanced XSS payload detection model. This model utilized GCN's ability to analyze graph structures and combined the idea of residual networks to make the model obtained better classification effect under multiple hidden layer structures.

The rest of the paper is organized as follows. In the Section 2, the research results are introduced on XSS detection in recent years. In Section 3, we introduce the basic principles of cross site scripting and implementation methods of XSS vulnerabilities. In Section 4, We will describe in detail the model for XSS payload detection. The results of our experiment and evaluation will be shown in Section 5. In Section 6 is the summary of the work.

2 RELATED WORK

In recent years, many researchers have proposed many detection methods for XSS attacks against XSS vulnerabilities, hoping to reduce the impact of XSS vulnerabilities on network security through efficient and accurate detection methods.

2.1 Rule-based detection methods

In the early days, Zalbina et al. built a security system (called NIDS) sufficient to detect and identify the payload of XSS attacks [11]. But he concluded that this system is only sufficient as the first defensive barrier in the future. This shows that the early use of regular expression matching methods had great limitations, and the detection rate for complex or unknown XSS payloads was very low. MK Gupta et al. proposed a method to detect XSS code files in Web applications through a predictive model based on text mining [12]. This method used a customized tokenization process to extract text features from the source code of a Web application, converted each file into a set of unique text features with related frequencies, and built a predictive model based on these features. But the program was limited to identifying vulnerable code from the source code of the application. The research of Sun et al. proposed a client method to detect JavaScript worm through Firefox plug-in [13]. This method was only for specific clients and could not provide protection for the entire Web application. In addition, it was also vulnerable to simple polymorphic worms, in which the payload signature was actively changed during the entire propagation process of the worm. Ter L. M. et al. developed a tool for protecting end users from XSS attacks, called BLUPRINT [16]. This tool strongly proved that web browsers would not run malicious scripts in web applications with low integrity output. The tool applied a technique that reduces trust in the web browser when understanding untrusted content. It had been used with many actual web applications to protect them from the XSS worm at a relatively low cost, but it did not directly detect malicious code.

2.2 Machine learning based detection methods

In recent years, with the popularity of machine learning algorithms, more and more researchers use machine learning to improve detection efficiency. BA Vishnu et al. extracted features from normal URLs, malicious URLs, and JS function points, and used three machine learning algorithms (SVM, J48 decision tree, and naive Bayes) to predicted whether the sample was a cross-site scripting attack code [14]. Similarly, S Rathore et al. used URLs, social network services, and web pages as feature items, and selected ten machine learning classification algorithms to divide the data set into two categories, XSS or non-XSS [15]. However, the above-mentioned research was based on traditional machine learning for feature learning. In the early stage, manual preprocessing of data and features was required, and there were certain limitations.

With the deepening of deep learning technology research, more and more researchers have begun to use new machine learning methods to implement classification problems. In 2019, Chen et al. proposed an XSS attack detection framework based on machine learning [17]. They also tried a variety of traditional machine learning algorithms, such as support vector machines, decision trees, and naive Bayes algorithms. In the end, they proved that the machine learning model could more effectively identify the unknown XSS payload. In the same period, Fang's team proposed a deep learning-based XSS attack detection method called DeepXSS [18]. They used word2vec to extract payload features, and then used the LSTM algorithm to build the model. In the experiment, the recall rate of the model reached 0.98, which means that it had a powerful ability to find XSS payload. In addition, the team also proposed a Bidirectional-RNN detection method based on the attention mechanism for XSS attacks in emails, and the accuracy of the final experimental results was as high as 99

Through the analysis of these studies, we could find that the use of deep learning technology could effectively improve the accuracy of unknown XSS detection, but a large number of labeled samples were required for model training before predicting samples. In actual environments, such samples are expensive to obtain. In addition, the XSS payload itself could be regarded as text content composed of characters and symbols. Refer to Thomas N. Kipf et al.'s idea that semi-supervised learning with graph convolutional networks could effectively perform text classification through a small number of samples [20]. This paper proposed a XSS detection model based on graph convolutional network. It was hoped

that through a small number of samples, the XSS payload could be accurately and quickly identified.

3 BACKGROUND AND MOTIVATION

3.1 Principle

The XSS (Cross-site scripting) is an attack method. The attacker uses a flexible method to inject malicious instruction code into the web page, so that the victim can load and execute the malicious code in the web application, and the attacker finally controls the browser. These malicious web programs are usually JavaScript, but in fact they can also include Java, VBScript, ActiveX, Flash or even plain HTML. After the attack is successful, the attacker may get a variety of content including but not limited to higher permissions (such as performing some operations), private web content, sessions, and cookies [21].

The common attack process of cross-site scripting is shown in Figure 1. The whole process is initiated by the attacker. The first step is to actively inject malicious scripts into web applications with XSS vulnerabilities, or trick victims into accessing the constructed URL, and passively inject malicious scripts into web applications. In the second step, when the malicious script is successfully injected, when the victim visits the web application server through the browser, the server returns the response containing the malicious script to the victim's browser, the browser will parse and execute it, and finally make the attacker successful perform an attack. The attacker can steal a lot of private information through the victim's browser and can even conduct further attacks through the victim's browser.

Through our analysis, we found that vulnerabilities are often caused by web application developers failing to check and filter the content entered by the user on the web or the parameters submitted by the URL, and directly display the requested content on the front-end page, leading to the loading and execution of malicious scripts, thus causing vulnerabilities. Some applications will also store user input in the database, such as forums (BBS) and e-mail, etc. [22]. This will lead to more serious consequences. Every time a victim visits a page with a malicious script injected, the malicious script code in the database will be loaded, making the visitor continue to be attacked by XSS (the type of xss attack will be described in detail in 3.2). Table 1 illustrates an example.

3.2 Type of XSS attack

In XSS attacks, according to the method of injecting malicious scripts into web applications, we mainly divided the types of XSS into three categories: Continuous Attacks (Stored XSS), Non-Continuous Attacks (Reflective XSS), and DOM-based XSS Attacks [23]. The characteristics of these three types of XSS and the ways of exploiting vulnerabilities are different.

- **Continuous Attacks**. Also known as stored XSS. The code is stored in the server, such as injecting script code in personal information or publishing articles. If there is no filtering or the filtering is not strict, then these codes will be stored in the server, and the code will be triggered whenever a user visits this page. This kind of XSS is extremely harmful and can easily cause worms, making information stolen. Email XSS attack is one of them. For example, adding script code to the rich text part. If the email system does not filter the email content, when the recipient receives the email and views it, it will trigger the XSS vulnerability and execute the script. Yahoo was exposed to two storage-type vulnerabilities from 2013 to 2016, which caused 40 million user mailboxes to be compromised [19].
- **Non-Continuous Attacks**. Also known as reflective XSS. The attacker makes the attack link in advance, and needs to trick the victim to visit the link to trigger the XSS code (there is no such page and content in the server). This kind of vulnerability is generally easy to appear on the search page. As shown in the example in Table 1, when the user entered a search keyword, the page would display this keyword on the page. If the web application did not properly filter the keyword, the entered keyword $< script > alert('xss') < /script >$would be injected into the front-end code, and the browser would follow the script of the content in the script tag Language to parse and execute, which leads to XSS vulnerabilities.
- **DOM-based XSS Attacks**. DOM is an interface that has nothing to do with the platform and programming language. It allows programs or scripts to dynamically access and update the content, structure and style of the document, and the processed result can become a part of the displayed page. The client-side script program can dynamically check and modify the page content through the DOM. It does not rely on submitting data to the server. The data in the DOM obtained from the client is executed locally. If the data in the DOM is not strictly confirmed, it will generate DOM-based XSS vulnerability. The use of DOM-based XSS and reflective XSS are similar. Both need to trick users into opening the constructed URL, but the content submitted by the URL is an interactive request locally in the browser and does not need to communicate with the server.

Since the use of DOM-based XSS is only local to the browser, there is no need to request the server. Therefore, XSS in this way is also difficult to detect and defend, just can only use other plug-ins, or optimize the code during development. Thus, the data detected in this paper is mainly for the other two types of XSS Payload data.

3.3 Attack vector

As major Internet companies pay more attention to network security, they have gradually enhanced the security of Web applications, and judged and filtered the input content [24]. Therefore, to ensure that cross-site scripting vulnerabilities can still be exploited by attackers, some variant construction methods are tried to be used by attackers, which can bypass the security detection of web applications. We summarize these methods of constructing cross-site scripting vulnerability mutations into the following four categories. Table 2 shows examples of various mutation methods.

- **keywords Substitute**. In order to fight against XSS attacks, some web applications include keywords, tags, and function names that are not used in the application into the blacklist. Once the submitted content is detected as being in the blacklist, it will be replaced or deleted. Because the content of the blacklist is incomplete, attackers can use

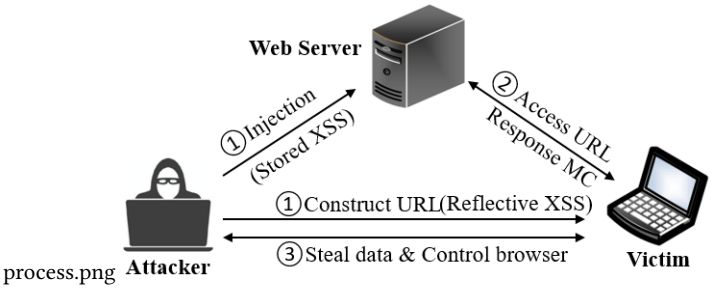


Figure 1: XSS Attack Process

Table 1: Examples of XSS vulnerabilities

PHP Source code	<? php $keyword = $_GET['k']; echo 'Searching: '.$keyword; ?>
Trigger	Access URL: url/test.php?a=<script>alert('xss')</script>
Consequent	The visitorâĂŹs browser will pop up a prompt box showing "xss"

keyword substitution and letter case changes to bypass the blacklist detection. For example, if "<script>" is in the blacklist, an attacker can use the onerror event in "<img>" instead of "<script>" to execute JavaScript code.

- **Encoding**. Browsers can actively decode and escape certain strings under certain circumstances, so that attackers can use this feature to encode attack vectors, bypassing the security detection of web applications, without affecting the original function of the code. For example, "%253C" can become "<" after two URI decoding. Common encoding types include HTML encoding, Unicode encoding, and UTF-7 encoding.
- **Position Transformation**. Regular matching systems and detection models often read and analyze the input string sequence from left to right. Therefore, the attacker bypasses security detection by changing the position of attribute formats. For example, if there are multiple attributes in a tag of an HTML page, swapping the positions of the attributes does not affect the browser's resolution, but this detection strategy may be bypassed.
- **Special Characters**. An attacker can bypass security detection by adding special characters or strings to the payload to disrupt its own word structure. In some special cases, these payloads can still take effect. For example, add a newline character in the event; add any string before and after the label; add a space after the event name; add a comment after the function name; use regular expressions to replace quotation marks.

4 PROPOSED APPROACH

In this part, we will discuss in detail our proposed GCN-based detection method for XSS attacks. Including how to preprocess the data, how to compose the data points into a graph structure, and how to use algorithms to perform deep learning of the graph structure, and finally realize the XSS payload detection.

4.1 Overview

Our overall solution is shown in Figure 2. First, all data is preprocessed, such as decoding and generalization, and then all payloads are divided into multiple segments through regular expressions, and each segment of the string is regarded as a word. The data preprocessing module is to reduce unnecessary parameters in the data to interfere with the detection results, and at the same time, it can also reduce the computational cost of subsequent machine learning and increase the efficiency of the algorithm. The composition module connects words and data in a regular manner to form a graph with multiple points and edges. The deep learning module uses the GCN-based algorithm to learn and optimize the parameters of the graph structure obtained in the composition module, and finally obtain the classification probability distribution of each point, and realize the classification of the data point type.

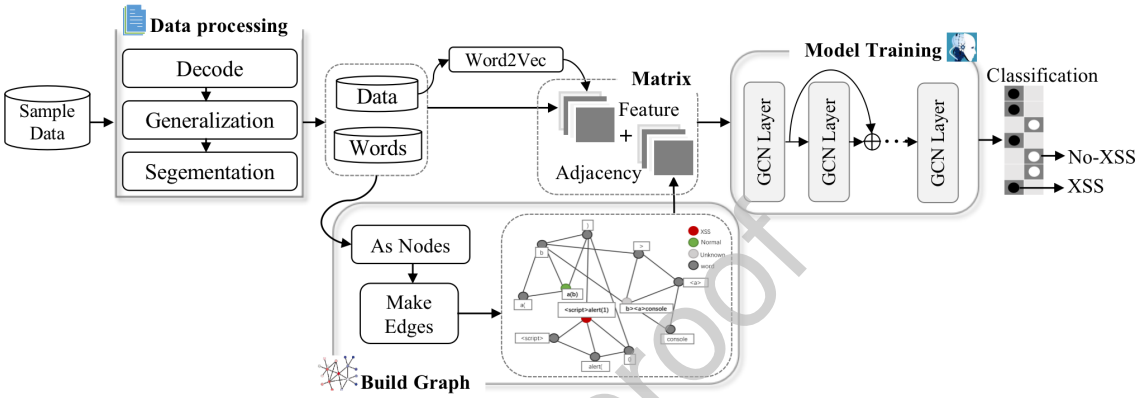
4.2 Data processing

4.2.1 Decode. When the browser displays the webpage, it will automatically decode and parse the code, which also allows attackers to use this feature to utilize encoding techniques to bypass WAF (Web Application Firewall), because the encoded payload can easily bypass the regular expression of WAF [26]. Therefore, in the process of payload detection, it is necessary to simulate the browser to decode the code into the final parsed form, so that the payload can be identified more accurately.

In the process of data preprocessing, we use several common decoding methods such as URL decoding, HTML entity encoding, Unicode decoding, and base64 decoding. The URL decoding and HTML entity encoding are still in their original form after decoding the unencoded character string, so the program decodes the URL and HTML twice by default. The Unicode encoding and Base64

Table 2: Examples of mutation types

Mutation Type	Example
keywords substitute	<aPplet cODe="jAvAscRipt:confirm(document.cookie);" >
Encoding	%3Cscript%3Econsole.log(%2Fxss%2F.source)%3C%2Fscript%3E
Position transformation	<img onerror=alert(1) src=1>
Special characters	aaa<img src=x %00 onerror%0a= console.log(1)><!–



arc.png

Figure 2: The system architecture of GraphXSS

encoding have obvious encoding characteristics, so the program can be used to identify whether these two encodings exist in the payload, and if they exist, the encoding part will be decoded accordingly. If the decoding fails in special circumstances, the original character form is retained. All data will be detected and decoded by these four decoding programs. Examples of encoding and decoding are shown in Table 3.

4.2.2 Generalization. In data preprocessing, a very important step is data conversion. There are three common data conversion methods: Data Standardization, Data Discretization, Data Generalization. Data standardization is to scale data to a specific interval. Data discretization is to replace data with intervals or categories. Data generalization is to abstract data to a higher conceptual level, which is more suitable for the method proposed in this paper. Use data generalization to abstract the non-critical parts of the data into a simpler representation.

Through our research on the cross-site scripting code, we found that after the encoding procedure mentioned in 4.2.1, the number in the payload has no direct effect on detecting whether the payload is cross-site scripting, the Javascript function or method that converts these numbers is the key detection object. In addition, the domain and path information in the payload is also meaningless for detection, how to reference foreign resources is our concern. Therefore, in order to reduce the computational cost, remove interference items, and improve the efficiency of the GCN network to extract high-dimensional features, we replaced the number that appeared in the decoded data with "0", and replaced the host address of the domain with "http://u", as shown in Table 4.

4.2.3 Segmentation. XSS payload is generally composed of closed symbols, HTML tags, events, attributes, JavaScript functions, calling methods, URLs and other parts. In order to detect payload, we need to separate its components and find the potential connection characteristics between components. Then, after decoding and generalizing the payload data, we used the regular expression tokenizer of the NLTK module (Nature Language Toolkit) in the Python library to divided the payload data into multiple segments based on the payload structure. We treated each segmented character as a word (the segmented character string will be referred to as "word" in the following text). Nature Language Toolkit is one of the most commonly used Python libraries in the field of NLP (Nature Language Processing), developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania [27]. We applied this tool to construct a set of regular expressions for word segmentation of payload data. Examples of word segmentation are shown in Table 5. Divide all the data, and then de-duplicate the results, and store them in a fixed order.

4.3 Build graph structure

After finishing the each data preprocessing, this paper proposed a scheme to transform the text content into a graph structure. The solution is to treat each piece of payload data and a word content as a point, connect these points according to the rules to construct a graph. First of all, a connection is established between each piece of data and its corresponding word, and then connect the words points according to the adjacent relationship of the data struction. In this way, the relationship between each word and the corresponding data can be established, and the connection between words can

Table 3: Decoding example

Type	Source code	Decode
URL	%3Cscript%3Ealert(1)%3C%2Fscript%3E	<script>alert(1)</script>
HTML		<img src="logo.png/">
Base64	test	<a href="data:text/html;base64, <img src=x onerror=alert(1)>"> test
Unicode	\u003cimg%20src=1%20onload=alert(123)/\u003e	<img%20src=1%20onload=alert (123)/>

Table 4: Data generalization example

Before	After
topic=http://ABCgroup.weIthr.edu/techniques/ index.php?topic=<script>alert(document.cookie)	topic=http://u=<script>alert(document.cookie)
siteID=';alert(String.fromCharCode(88,83,83))// \';alert(String.fromCharCode(88,83,83))//"; alert(String.fromCharCode(88,83,83))//\"; alert(String.fromCharCode(88,83,83))//–>">'> <SCRIPT>alert(String.fromCharCode(88,83,83))	siteid=';alert(string.fromcharcode(0,0,0))//'; alert(string.fromcharcode(0,0,0))//"; alert(string.fromcharcode(0,0,0))//"; alert(string.fromcharcode(0,0,0))//–>">'> < script>alert(string.fromcharcode(0,0,0))

Table 5: Example of word segmentation in preprocessing

Payload data	After segmentation
topic=http://u=<script>alert(document.cookie)	['topic=', 'http://u', '<script>', 'alert(', 'document.cookie', ')']
js='"–>< script>alert(/meeh infected/)< marquee>< h1>XSS :)< marquee>< strong> < blink>XSS TEST< h1>XSS :)	['js=', '>', '< script>', 'alert(', '/', 'meeh', 'infected', '/', ')', '< marquee>', '< h0>', 'xss', ')', '< marquee>', '< strong>', '< blink>', 'xss', 'test', '< h0>', 'xss', ')']

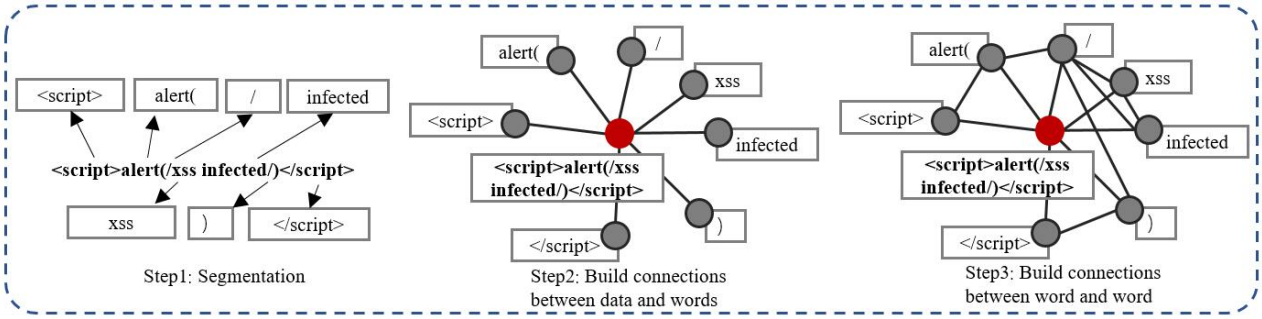
also express the sequential connection of word meaning. Because it is possible that the same words are composed of different orders, the payload will express different meanings. In addition, the use of undirected connection to compose the graph, blur the sequence relationship between adjacent words, can better represent the data structure and find feature information from a more generalized adjacent relationship easily, although this cannot accurately express every detail of the payload, the final experimental results show that this does not affect the classification performance of the detection model. For example, we segmented "<script>alert(/xss infection/)<script>" and constructed a graph. An example of the composition process and graph structure would be as shown in Figure 3.

Because in different payload data, there must be some common words, such as brackets, "<script>", "<img", etc. Therefore, such a composition scheme can connect all data through these common points to form a larger and more complex graph structure, which is convenient for subsequent algorithm calculations. In order to show the effect of composition more intuitively, we randomly selected 150 XSS payloads and 100 no-XSS data from the data set for preprocessing and composition. The result is shown in Figure 4. The red points in the figure are XSS payload data points, the green points are normal data points, and the blue points are word points. Obviously, our composition scheme can construct almost all the data in a graph, in the actual experiment, we only marked a small part of the red and green points. Through model training, the feature value of each output point can be used to classify unlabeled data points, so as to realize whether there is an XSS payload in the data.

4.4 Deep learning algorithm

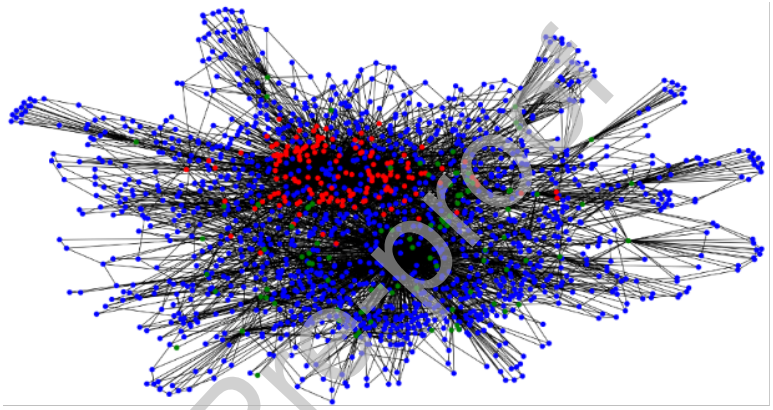
4.4.1 Graph convolutional network. Graph Convolutional Network (referred to as GCN) is a kind of neural network that operates on graphs [20]. Graph in Graph Convolutional Network is a topological graph that uses vertices and edges to establish corresponding relationships in mathematics (Graph Theory). In deep learning algorithms, convolutional neural networks and recurrent neural networks are used to process data in Euclidean space. The characteristic of European spatial data is that it is structured. However, the structure of the graph is generally very irregular, and can be considered as an infinite dimensional data, so the researchers proposed to use GCN to extract graph data features [32].

In GCN, given a graph G = (E, V), E means Edge and V means Vertex, the following features can be used:



graph-new.jpg

Figure 3: Example of building graph structure



graph.png

Figure 4: Build a graph example from 250 pieces of data (This part of the data is the small sample experimental data in section 5.3.1)

Node Feature: Each node i has its feature x_i, which can be expressed by the matrix X_{N*D}. N is the number of nodes. D indicates the number of features per node.

Graph structure features: The information on the graph structure can be represented by an N^*N adjacency matrix A, where $A_{ij} = 1$, if node i and node j are connected, otherwise $A_{ij} = 0$.

In this paper, only some data points had known label information, then the graph could be regarded as a standard structured feature classification problem. Each hidden layer in the network can be represented by the following non-linear function, formulaically describing the basic network layer as formula (1) and the layer conduction rule as formula (2):

$$H^{(l+1)} = f(H^{(l)}, A) \tag{1}$$

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{2}$$

l indicates the number of hidden layers, the input layer is $H^{(0)} = X$; A represents the adjacency matrix; W is the weight of the feature H in the upper layer; Different network structures will use different functions $f()$. σ is the activation function, "ReLU" is used as the activation function in this paper. Multiplying the adjacency matrix A and the feature H is equivalent to adding the features of the adjacent nodes through the adjacency matrix to obtain the input of the next hidden layer. The multiple hidden layers approximate the use of information from multiple layers of neighbor nodes.

Furthermore, in order to solve the problem that the node itself is connected to itself (it has an edge with itself) and the adjacency matrix is not generalized, the problem that nodes with many neighbor nodes tend to have larger eigenvalues [20]. We added self-connection features to the adjacency matrix A and perform symmetric normalization, so that the final GCN network layer feature propagation formula is shown in formula (3):

$$f(H^{(l)}, A) = \sigma(\widetilde{D}^{(-1/2)}\widetilde{A}\widetilde{D}^{(-1/2)}H^{(l)}W^{(l)}) \tag{3}$$

In formula (3), $\widetilde{A} = A + I$.The formula adds the adjacency matrix A and the identity matrix I, so that the adjacency matrix introduces the node self-connection feature. At the same time, similar to the construction of the frequency domain transformation in the Fourier transformation, use $\widetilde{D}^{(-1/2)}\widetilde{A}\widetilde{D}^{(-1/2)}$ to generalize the adjacency matrix A. In the formula (3), $\widetilde{D}$ can be expressed by formula (4):

$$\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij} \tag{4}$$

$\widetilde{D}$ is a diagonal matrix. The value on the diagonal of the matrix is the degree of the corresponding node. In this method, the use of multi-layer for training can use the information of multi-layer

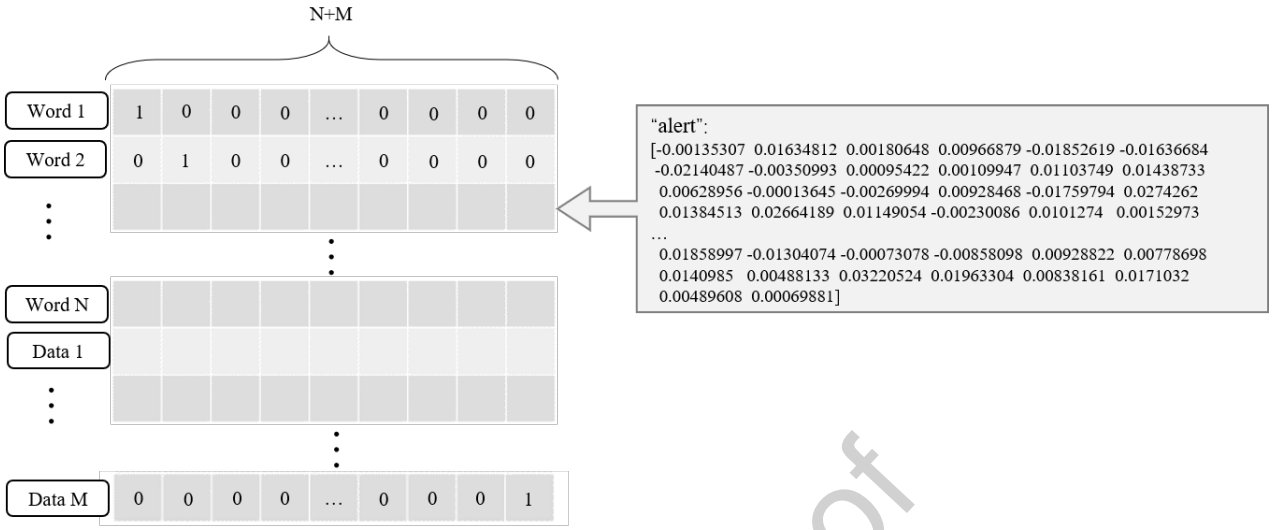


Figure 5: Example of feature matrix structure

neighbor nodes to allow the model to learn the information between data and words.

Word2Vec is a tool released by Google in 2013, specifically for word vector calculation [28]. Word2Vec can run efficiently in a million-level corpus, and the generated word vector can express the relationship between words. Therefore, we decided to use Word2Vec as a feature extraction tool for data segmentation points.

As we seen from formula (3) that the input data of the entire model is the feature matrix H and the adjacency matrix A of the graph. In this paper, we integrated the N words after data preprocessing with the features of M pieces of data and arranged them in a matrix to construct a (N+M)*(N+M)-dimensional feature matrix. First of all, suppose this feature matrix is an identity matrix, and each row of the matrix represents the feature vector (N+M dimension) of a word or a piece of data. Then we used the word vector tool (Word2Vec) in third-party library Gensim of Python to replace the word feature vector (originally one-hot code) of the corresponding word with the word vector (Referring to Zhang's research on XSS detection [25], in this paper the word vector was set to 200 dimensions, N+M>200), and filled the vacant part with "0". The structure of the feature matrix and the word vectors are shown in Figure 5, where the word vector takes the word "alert" as an example.

First, we used the Word2Vec pre-training model to train all the segmented words and data samples. Since we hope to classify data through a small number of samples in this paper, we only selected 3000 high-frequency segmentation words in the word segmentation dictionary for training. We can obtain the word vectors of some words through Word2Vec, and then replaced these word vectors with the feature vectors of the corresponding words (rows) in the original feature matrix, and finally get the feature matrix $H^{(0)}$ of the input layer in the GCN network.

4.4.2 Residual network. In deep learning, as the depth increases, there will be a degradation problem: When the network depth becomes deeper, the training accuracy rate will tend to be flat, but the training error will become larger [29]. To solve this problem, Kaiming He et al. proposed a residual network to solve this problem [30].

A residual network is composed of multiple residual blocks. The definition of the residual block is $y = F(x, W_i) + x$, where x and y are the input and output of the residual block, and $F(x, W_i)$ is the residual mapping that needs to be learned. The residual network is formed by the fusion of multiple shallow networks, which avoids the problem of gradient disappearance. The shallow network will not have the disappearing gradient problem during training, so it can accelerate the convergence of the network.

In this paper, we learned from the idea of residual network. For traditional GCN algorithms generally only have 2 to 3 layers, we proposed to add residual network structure between hidden layers of GCN (increase the number of hidden layers to 6). We added the feature matrix output by the first layer of the hidden layer of the GCN network and the feature matrix output by the second layer as a new feature matrix and passed it to the third hidden layer for calculation, and so on, until the output of the last layer (It turns out that the second layer directly uses the output result of the first layer for calculation). According to our observations, using this scheme in the XSS payload identification based on the GCN algorithm, compared with the GCN network without adding the residual network structure, can effectively accelerate the convergence of the model, and at the same time can avoid the problem of the disappearance of the gradient caused by the multilayer structure. In this way, the XSS payload detection model finally combines the hidden layer feature propagation form of the GCN network with the residual network structure as formula (5) (6), and the residual structure is shown in

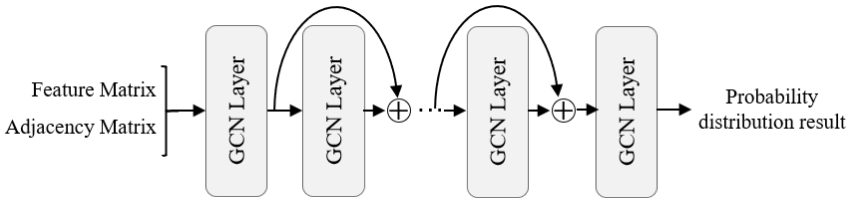


Figure 6: Structure of the algorithm after introduction of the residual network

Table 6: Python libraries

Module	Library and Version
Data processing	Nltk == 3.5
	Re (Built-in)
	HTML (Built-in)
	Base64 (Built-in)
	Urllib (Built-in)
	CSV (Built-in)
	Collections (Built-in)
Feature extraction	Gensim == 3.8.3
	Numpy == 1.16.0
Machine learning	Keras == 2.2.4
	Torch == 1.6.0
	Scikit-learn == 0.23.2
	Tensorflow-gpu == 1.13.1

Figure 6.

$$H^{(l+1)} = f(H^{(l-1)} + H^{(l)}, A) \tag{5}$$

$$H^{(l+1)} = \sigma(\widetilde{D}^{(-1/2)}\widetilde{AD}^{(-1/2)}(f(H^{(l-1)}, A) + H^{(l)}, A))W^{(l)}) \tag{6}$$

5 EXPERIMENT

5.1 Environment

The experimental environment designed in this paper was all running under the Windows 10 64-bit operating system, and all the functional modules in the content were developed and ran under the Python 3.6 environment. All the involved Python libraries and their versions are shown in Table 6.

5.2 Dataset

The GCN-based XSS detection model is a semi-supervised learning method, so a part of the labeled data set is still needed for model training. However, public data in the field of network security is very scarce. In order to ensure the authenticity of the data and practical application capabilities, we crawl the latest XSS payload on the *xssed.com* [31] website as a data set. The data set covered the above three types of XSS attack payloads. As shown in Table 7, we obtained 5,000 XSS payload data through the crawler program as a positive example. For the other part of the normal request data, we selected 9,000 normal HTTP GET request records from the laboratory servers as a negative sample. In the experiment, We toke 2,000 pieces of normal data and 1,500 pieces of XSS payload data in the dataset as the large sample data set; toke 150 pieces of normal data and 100 pieces of XSS payload data as the small sample data set. In addition, we divided the experimental data into training

Table 7: Dataset

Type	Large sample	Small sample
No-XSS data	2,000 (from 9,000)	150 (from 9,000)
XSS payload	1,500 (from 5,000)	100 (from 5,000)

data and test data, and used the 10-fold cross-validation method to verify the experimental results. In the entire calculation process, we only considered the predicted classification of data points.

5.3 Experiment and analysis

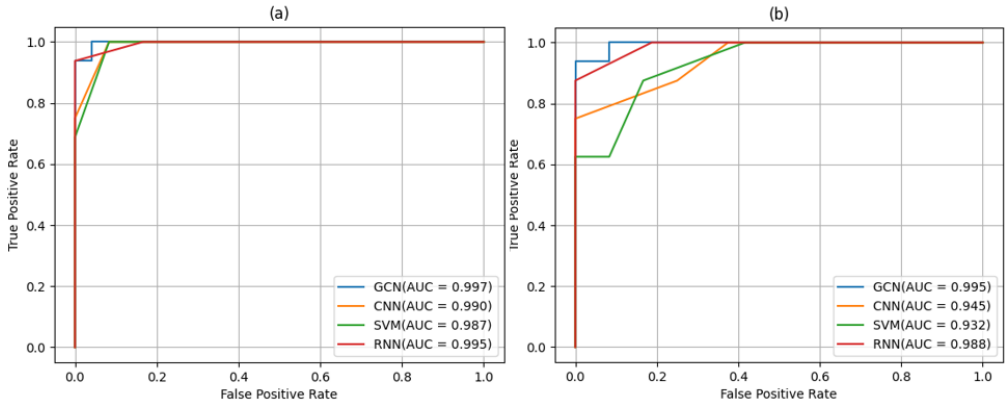
The deep learning methods proposed by many researchers to detect XSS payloads are very effective, with an accuracy rate of more than 0.98, but they rely on a large number of labeled data samples to train the model. So, we designed two experiments based on this condition to verify the validity and advancement of the model proposed in this paper.

The first experiment is to compare the effects of different sample sizes on different detection models. We compared the AUC values of the results to determine which model is least affected by the sample size. The second experiment is to compare how different factors affect the results of the model in terms of composition strategies and model building methods. The effects of different factors are shown through the final accuracy value of the model and the speed of accuracy improvement. Finally, we displayed the classification results of the model proposed in this paper in the first experiment on a two-dimensional plane through dimensionality reduction to observe whether the algorithm is effective in distinguishing different point features.

In a binary decision problem, the result of each classification may either be positive or negative. As shown in Table 8, the decision can be represented in a structure called as a confusion matrix.The confusion matrix has four categories:T_p (True Positive) indicates the amount of XSS samples correctly classified, and F_p (False Positive) indicates the amount of normal samples classified as XSS. Likewise, T_n (True Negative) indicates the amount of normal samples correctly classified, and F_n (False Negative) indicates the amount of XSS samples classified as normal. Based on confusion matrix, we evaluate the experimental results with accuracy, recall, F1 score, and their formulas criteria are shown in following equations.

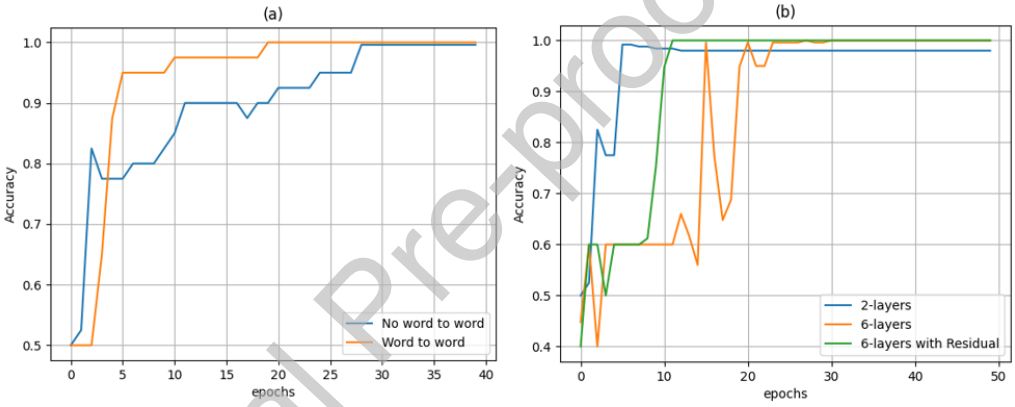
$$Accuracy = \frac{T_p + T_n}{(T_p + F_n) + (F_p + T_n)} \tag{7}$$

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



roc.png

Figure 7: The ROC curve of different models under different sample sizes ((a) is under the condition of a large sample, (b) is under the condition of a small sample)



exp.png

Figure 8: (a) The influence of establishing a connection between words; (b) The influence of different model structures on recognition accuracy

Table 8: Confusion matrix

	Predicted XSS	**Predicted No-XSS**
Actual XSS	T_p	F_n
Actual No-XSS	F_p	T_n

$$Precision = \frac{T_p}{T_p + F_p} \tag{9}$$

$$F1 = \frac{2Precision * Recall}{Precision + Recall} \tag{10}$$

5.3.1 Sample size experiment. In this experiment, we first selected 2,000 XSS payload data and 1,500 normal data as the data set to train the model. Then reduced the number of data sets, and selected only 150 XSS payload data and 100 normal data as the data set. In this way, we can observe the effect of reducing the number of samples on the model prediction results.

We set our model parameter epoch to 50, loss function to "nll_loss" (Negative Log Likelihood loss), and optimization function to "Adam" (learning rate=0.01). As a control group, we selected the RNN-based model proposed by Fang's team [19], the CNN-based model proposed by Foxscheduler [33], and the traditional machine learning SVM model for comparative testing with the same sample set we collected. Finally, to evaluate different models, we plotted the ROC curve and calculated the AUC value, as shown in figure 7.

From Figure 7 we can see that as the number of training samples decreases, the AUC values of the traditional SVM algorithm and the CNN-based model are affected and decreased, which is lower than the original AUC value (the original AUC is greater than 0.98), it is also significantly lower than the GCN-based model. However, the RNN-based model is least affected by the reduction in the number of samples, and the AUC value can still reach 0.99. According to our analysis, this is because deep learning is more suitable for processing high-dimensional data sets than traditional machine learning, and the algorithmic characteristics of GCN and GNN make them more suitable for processing text classification. Under

Table 9: More detail of the algorithm

	Accuracy	Recall	F1 score
GraphXSS	0.996	0.997	0.997
Foxscheduler-CNN	0.955	0.971	0.971
SVM	0.850	0.975	0.912
Fang-RNN	0.991	0.987	0.889

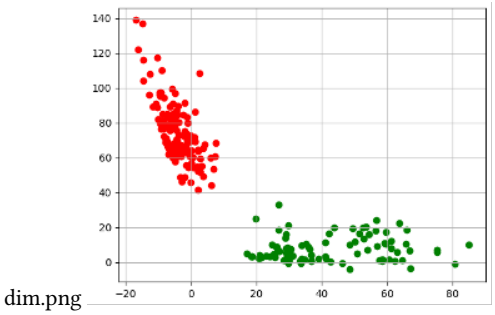
the conditions of two kinds of data sets, the accuracy and recall rate of the final prediction results of the detection model proposed in this paper have exceeded 0.99, which shows that the model we designed is very effective. In a small sample condition, a more detailed assessment is shown in Table 9.

5.3.2 Build model strategy experiment. In the second experiment, we tried to use the same set of data to compare the composition strategy and model building method, and how different factors would affect the model results. First of all, in the process of constructing the graph structure, we compared the effect of establishing and non-establishing the connection between words on the model; Then we compared the traditional 2-layer GCN model, the 6-layer GCN model and the 6-layer GCN model with residual network proposed in this paper. We hope that through this experiment can prove that the multi-layer structure model is more conducive to model learning, and the model structure combined with the residual network is better than the traditional GCN model in XSS payload dectection. Therefore, in the experiment, we recorded the changes in the accuracy of model prediction results of different models as the epoch increased, and plotted the accuracy curve as shown in Figure 8.

Through the observation of (a) in Figure 8, we could find that in the process of composition, establishing the connection between words could speed up the convergence of the model, and the final model accuracy rate can reach 0.996. However, without establishing a graph of the connection between words, the training results converge slowly, and the final model detection accuracy rate is only 0.996. This shows that the connection between words helps express the association between words, which is conducive to the model's acquisition of information in the graph structure.

We observe (b) in Figure 9 and we could see that increasing the number of hidden layers in the model would increase the computational cost and made the data structure more complex, so the accuracy rate rose more slowly. Comparing the calculation results of the 2-layer structure and the 6-layer structure, the multi-layer structure could make the model obtain higher recognition accuracy, indicating that the multi-layer structure could make the model obtain more effective information. Furthermore, after adding the residual network structure, the model could converge the results faster and more stably, and the accuracy rate reached 0.996, while the accuracy rates of the 2-layer and 6-layer models reached 0.985 and 0.996. Thus it can be seen that the model proposed in this paper is superior to the traditional GCN model and other detection models.

5.3.3 Classification effect. In addition to the above-mentioned control experiments, we used the dimensionality reduction method to convert the model calculation results under the small sample condition in the first experiment to convert multi-dimensional data into two-dimensional data for display in a two-dimensional plane. As shown in Figure 9, the final classification result can clearly distinguish the two types of points, while the same type of data is clustered together. The red dots in the figure represent XSS payload data, and the green dots represent the data of a common request. It can be concluded that the detection model we proposed is able to effectively classify samples such as XSS payload.



dim.png

Figure 9: 2-dimensional display of data categories

6 CONCLUSION

In this paper, an XSS detection model based on deep learning was proposed. This model could detect whether there is an XSS attack payload in the data submitted by the user. We optimized the performance of the detection model by combining the graph convolutional network and the residual network, and proved this through experiments. This article first summarized the basic principles and research motivation of XSS attacks. Then we proposed the method of transforming text content into graph structure for detection. Furthermore, the XSS detection framework based on graph structure was proposed. Finally, the experiment proved that the model has excellent classification and recognition performance. The model we proposed could achieve 100% accuracy of sample recognition even with a small number of training samples, which is significantly better than other detection schemes. In the real world, we can use the detection model to detect whether the user submitted content contains XSS payload. If so, we can filter the submitted content and issue a system warning, which effectively protects Web application resources and user privacy information security from XSS attacks.

ACKNOWLEDGMENTS

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

REFERENCES

[1] OWASP, OWASP Top 10 - 2017. Available at https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_(en).pdf.pdf. Accessed 22, October, 2020.
[2] M. Mohammadi, B. Chu and H. R. Lipford, "Detecting Cross-Site Scripting Vulnerabilities through Automated Unit Testing," 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), 2017, pp. 364-373.
[3] V. Nithya, S.L. Pandian, C. Malarvizhi. A survey on detection and prevention of cross-site scripting attack. International Journal of Security and Its Applications, 2015, 9(3): 139-152.

[4] C. Liu, X. Cui, Z. Wang, X. Wang, Y. Feng and X. Li, "MaliceScript: A Novel Browser-Based Intranet Threat," 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), 2018, pp. 219-226, doi: 10.1109/DSC.2018.00039.

[5] L. Constantin, New Chinese social networking worm discoverd. Available at http://news.softpedia.com/news/New-Chinese-Social-Networking-Worm-Discovered-120021.shtml. Accessed 22, October, 2020.

[6] Technical explanation of The MySpace Worm. Available at https://samy.pl/popluar/tech.html. Accessed 5, June, 2018.

[7] G. Cluley. Cross-platform Boonana Trojan targets Facebook users. Available at https://nakedsecurity.sophos.com/2010/10/28/cross-platform-worm-targets-facebook-users/. Accessed 22, October, 2020.

[8] Hackagon. XSS attack. Available at http://hackagon.com/xss-attack/. Accessed 12, Augest, 2016.

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

[10] OWASP, XSS Filter Evasion Cheat Sheet. Available at https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet. Accessed Accessed 22, October, 2020.

[11] M. R. Zalbina, T. W. Septian, D. Stiawan, M. Y. Idris, A. Heryanto and R. Budiarto, "Payload recognition and detection of Cross Site Scripting attack," 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), 2017, pp. 172-176, doi: 10.1109/Anti-Cybercrime.2017.7905285.

[12] M. K. Gupta, M. C. Govil and G. Singh, "Text-mining based predictive model to detect XSS vulnerable files in web applications," 2015 Annual IEEE India Conference (INDICON), 2015, pp. 1-6, doi: 10.1109/INDICON.2015.7443332.

[13] F. Sun, L. Xu, Z. Su. Client-side detection of XSS worm by monitoring payload propagation. Proceedings of the 14th European Symposium on Research in Computer Security, Saint-Malo, France, 2009, pp. 539-554.

[14] B. A. Vishnu, K. P. Jevitha. 2014. Prediction of Cross-Site Scripting Attack Using Machine Learning Algorithms. In Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing (ICONIAAC '14). Association for Computing Machinery, New York, NY, USA, Article 55, 1âĂŞ5. DOI:https://doi.org/10.1145/2660859.2660969

[15] R. Shailendra, S. P. Kumar, P. J. Hyuk. XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs. Journal of Information Processing Systems [Internet]. 2017 Aug 31;13(4):1014âĂŞ28.

[16] M. T. Louw, V. N. Venkatakrishnan. "Blueprint: Robust Prevention of Cross-site Scripting Attacks for Existing Browsers," 2009 30th IEEE Symposium on Security and Privacy, 2009, pp. 331-346, doi: 10.1109/SP.2009.33.

[17] X. L. Chen, M. Li, Y. Jiang, Y. Sun. A Comparison of Machine Learning Algorithms for Detecting XSS Attacks. In: Sun X., Pan Z., Bertino E. (eds) Artificial Intelligence and Security. ICAIS 2019. Lecture Notes in Computer Science, vol 11635. Springer, Cham.

[18] 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 IntelligenceâĂŤICCAI, Chengdu, China, 12âĂŞ14 March 2018; pp. 47âĂŞ51.

[19] Y. Fang,Y. Xu, P. Jia, C. Huang, Providing Email Privacy by Preventing Webmail from Loading Malicious XSS Payloads. Appl. Sci. 2020, 10, 4425.

[20] N.K. Thomas, W. Max, Semi-Supervised Classification with Graph Convolutional Networks. International Conference on Learning Representations 2017, 2017.

[21] S. Goswami, N. Hoque, D. K. Bhattacharyya, J. Kalita. An Unsupervised Method for Detection of XSS Attack. Int. J. Netw. Secur. 19 (2017), pp. 761-775.

[22] X. Wang, R. Wu, J. Ma, G. Long, J. Han. Research on vulnerability detection technology for web mail system. In Proceedings of the International Conference of Information and Communication TechnologyàĂŤICTC, Jeju Island, Korea, Volume 131, pp. 124âĂŞ130.

[23] M. K. Gupta, M. C. Govil and G. Singh, "Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: A survey," International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014), 2014, pp. 1-5, doi: 10.1109/ICRAIE.2014.6909173.

[24] Y. Gao, Y. Hu. Web Front-end XSS Filtering Technology. Communications Technology, 2017, 50(3).

[25] J. Zhang, Y. Jou, X. Li. Cross-Site Scripting (XSS) Detection Integrating Evidences in Multiple Stages. In Proceedings of the 52nd Hawaii International Conference on System SciencesâĂŤHICSS, Grand Wailea, Maui, 2019, pp. 7166âĂŞ7175.

[26] Stan1y. How to bypass XSS protection. Available at https://www.freebuf.com/articles/web/245097.html. Accessed 22, October, 2020.

[27] L. Edward, B. Steven. NLTK: The Natural Language Toolkit. In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, 2002, pp. 62âĂŞ69.

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

[29] A. Veit, J.W. Michael, B. Serge. Residual networks behave like ensembles of relatively shallow networks. Advances in Neural Information Processing Systems. 2016, pp. 550-558.

[30] H. Kaiming, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition ,2016, pp. 770-778.

[31] XSSed | Cross Site Scripting (XSS) attacks information and archive. Available at http://www.xssed.com/. Accessed 22, October, 2020.

[32] D. Marcheggiani, I. Titov. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling// Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. 2017, pp. 1506-1515

[33] Foxscheduler. How to use deep learning to detect XSS. Available at https://www.freebuf.com/news/142069.html. Accessed 22, October, 2020.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CREDIT AUTHOR STATMENT

Zhonglin Liu: Conceptualization, Methodology, Software, Writing - Original draft preparation

Yong Fang: Resources, Investigation, Validation

Cheng Huang: Data curation, Formal analysis, Writing - Reviewing and Editing.

Jiaxuan Han: Project administration, Supervision

GraphXSS: An Efficient XSS Payload Detection Approach Based on Graph Convolutional Network

Zhonglin Liu[a], Yong Fang[a], Cheng Huang[a,*] and Jiaxuan Han[a]

[a]School of Cyber Science and Engineering, Sichuan University, Chengdu, China

ARTICLE INFO

Keywords:
Cross-site Scripting
Detection
Graph Convolutional Network
Residual network

ABSTRACT

With the rapid development of the Internet age today, Web applications have become very common in modern society. Web applications are often applied to a social network, media, management, etc., and usually contain a large amount of personal privacy information, which makes Web applications a common target for hackers. The most common method for stealing private information from web applications is cross-site scripting attacks. Attackers frequently use cross-site scripting vulnerabilities to steal victims' identity information or hijack login tokens. Therefore, we proposed a cross-site scripting payload detection model based on graph convolutional networks, which could identify the cross-site scripting payload in the content submitted by the user (We termed our implementation of this approach, GraphXSS). We preprocessed the sample, and constructed the processed data into a graph structure, and finally used the graph convolutional network and the residual network to train the cross-site scripting detection model. In experiments, the model based on graph convolutional network (GCN) could achieve AUC value of 0.997 under small sample conditions. Compared with the detection model after adding the residual network structure, the model could converge and stabilize under the multi-layer, and could make the accuracy rate reached 0.996.

Zhonglin Liu received the M.S. degree from Sichuan University, Chengdu, China, in 2016. He is currently pursuing the Ph.D degree with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China. His current research interests include Web security, machine learning and attack detection.

Yong Fang received the Ph.D degree from Sichuan University, Chengdu, China, in 2010. He is currently a Professor with college of Cybersecurity, 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 Assistant Research Professor at the college of Cybersecurity, Sichuan University, Chengdu, China. His current research interests include Web security, attack detection, artificial intelligence.

Jiaxuan Han obtained his BS degree in Information Security from Guizhou University in 2019. Now he is studying for Ph.D. degree in School of Cyber Science and Engineering at Sichuan University. His research field is cyberspace security. And his research interests include: source code security, vulnerability detection, etc.

jungleforsa@gmail.com (Z. Liu); yongfangscu@gmail.com (Y. Fang); opcodesec@gmail.com (C. Huang); zhanSxDrive30i@gmail.com (J. Han)
ORCID(s):