mathematics

MDPI

Article

A Malicious Webpage Detection Method Based on Graph Convolutional Network

Yilin Wang, Siqing Xue * and Jun Song

School of Computer Science, China University of Geosciences, Wuhan 430074, China
* Correspondence: xuesiqing@cug.edu.cn

Abstract: In recent years, with the rapid development of the Internet and information technology, video websites, shopping websites, and other portals have grown rapidly. However, malicious webpages can disguise themselves as benign websites and steal users' private information, which seriously threatens network security. Current detection methods for malicious webpages do not fully utilize the syntactic and semantic information in the web source code. In this paper, we propose a GCN-based malicious webpage detection method (GMWD), which constructs a text graph to describe and then a GCN model to learn the syntactic and semantic correlations within and between webpage source codes. We replace word nodes in the text graph with phrase nodes to better maintain the syntactic and semantic integrity of the webpage source code. In addition, we use the URL links appearing in the source code as auxiliary detection information to further improve the detection accuracy. The experiments showed that the proposed method can achieve 99.86% accuracy and a 0.137% false negative rate, achieving a better performance than other related malicious webpage detection methods.

Keywords: malicious webpage detection; GCN; webpage source code; deep learning; URL

MSC: 68T99

Citation: Wang, Y.; Xue, S.; Song, J. A Malicious Webpage Detection Method Based on Graph Convolutional Network. *Mathematics* **2022**, *10*, 3496. https://doi.org/10.3390/math10193496

Academic Editor: Andrea Scozzari

Received: 15 August 2022
Accepted: 21 September 2022
Published: 25 September 2022

1. Introduction

The Internet has been largely popularized, and more and more users are enjoying diverse services. The explosive increase in service types requires various Internet companies to provide a large number of webpages as service carriers. On the one hand, malicious webpages have become a means for many attackers to carry out malicious attacks, seriously threatening our network security. On the other hand, the daily proliferation of webpages makes it impossible to detect every webpage for maliciousness. The 47th Statistical Report on China's Internet Development (February 2021) [1] showed that 38.3% of Internet users expressed that they had suffered from Internet security problems during their Internet access in the past six months. In 2020, the National Internet Emergency Response Center carried out monitoring and found that the number of tampered websites in China was 243,709.

At present, attackers use various obfuscation techniques against traditional malicious web detection methods, which greatly reduce the correct detection rate [2,3]. Traditional methods cannot sufficiently solve new types of malicious web attacks. Therefore, we urgently need to develop effective malicious web detection methods, and thus designing new detection methods has become an important way to solve the problem.

On the one hand, researchers have found that there are still syntactic similarities among the malicious parts in different malicious webpage source codes, which means that malicious webpages can still be effectively detected by analyzing the syntax and semantics [3,4]. On the other hand, as one of the main models of deep learning, graph convolutional networks (GCNs) have been widely used for node classification in recent years

and have shown good performance in text classification [5–7]. In this paper, we propose a GCN-based malicious webpage source code detection method named GMWD. The GCN is able to capture global word co-occurrence information in a corpus of non-contiguous and long-range semantics, making full use of the syntactic and semantic information of the source code. We also propose a new preprocessing strategy to maintain the integrity of the syntactic and semantic information of the code. Compared with other malicious webpage detection methods, this method improves the accuracy of detection.

We make the following contributions:

1. We design a malicious webpage detection method, GMWD, based on a GCN. The model can fully explore and exploit the syntactic and semantic correlations within and among webpages to accurately detect malicious webpages.
2. We propose phrase nodes to replace the word nodes in the text graph. Phrase nodes can effectively maintain the syntactic and semantic information integrity of the source code, which is beneficial for the GCN to capture the features of the malicious codes. We slice the source code into phrases based on point-wise mutual information and left–right entropy.
3. We use the URL links, which are contained in the source code and point to other websites, as auxiliary detection information and use blacklist detection technology to quickly pre-detect them, further improving the overall accuracy of the detection scheme.

A comprehensive evaluation of the proposed method GMWD is provided. We used 52,448 web samples as the dataset for our experiments, containing 26,896 benign samples and 25,552 malicious samples. The dataset was obtained using a common method of crawling websites such as the Malware Domain List. The results show that GMWD can carry out accurate detection, and the accuracy can reach more than 99.8%.

The rest of this paper is outlined as follows: Section 2 introduces related work on malicious webpage detection and deep learning methods. Section 3 describes the theoretical background related to GCNs and illustrates how malicious webpage detection can be performed using GMWD. The experimental results are analyzed in Section 4. Section 5 concludes this paper.

2. Related Work

Machine learning models have been widely used in malicious webpage detection, and these models can learn the difference between malicious and benign webpages. The authors of [8] found co-occurring factors to identify malicious websites using various association rule mining algorithms, not only solving the imbalance in the dataset, but also using feature ranking techniques to rank the importance of features. The authors of [9] designed a machine learning model based on logistic regression to detect malicious URLs. The authors of [10] proposed a malicious webpage detection scheme based on a logistic regression classification algorithm, which can reach 92% detection accuracy by using webpage redirect link-related information as features. The authors of [11] recommended an anti-phishing method, which extracts 19 functions from the client to distinguish malicious websites from benign ones by means of machine learning. The authors of [12] combined feature extraction from abstract syntax trees with a random forest classifier to detect malicious JavaScript instances. The authors of [13] proposed a context-sensitive and keyword density-based approach for webpage classification. In their scheme, three feature extraction methods are used to extract features and information. However, those studies usually required feature selection operations, which are especially expensive since they require domain experts [14–16].

Deep learning is a relatively new field of machine learning research that has been widely used in many scenarios [17], such as target recognition [18] and image classification [19]. Many studies have shown that deep learning performs well in malicious webpage detection. The studies of [12,20,21] detected malicious webpages using neural networks such as CNNs. They extracted features from static information of malicious webpages in multiple dimensions and then detected malicious webpages according to different levels of

features. The accuracy of these methods is generally higher than that of traditional static detection techniques. However, these methods only consider static features without taking into account the syntactic and semantic features of the source code, and it is easy for attackers to de-feature malicious webpages based on the static features extracted from webpages, which greatly reduces the accuracy of detection. There are only a few works similar to ours. The authors of [3,4] extracted features from the semantic level of JavaScript using LSTM to detect malicious JavaScript instances. However, these methods only took advantage of the semantics of partial features, such as local syntax features from JavaScript [4].

Graph convolutional networks (GCNs) have shown extensive performance in node classification, social recommendation, and link prediction. The authors of [6] first applied GCNs to text classification. The authors proposed an efficient layer-wise propagation rule that is based on a first-order approximation of spectral convolutions on graphs and applied it to graph structure data. Experiments showed that the proposed GCN model is able to encode both the node features and graph structure in a way that is useful for semi-supervised classification. Furthermore, [7] improved the work of [6] by constructing a single large graph from an entire corpus, which contains words and documents as nodes and captures global word co-occurrence information using a GCN.

As CNNs and LSTM focus on locality and sequentiality, other schemes using these deep learning models can sufficiently capture semantic and syntactic information in local consecutive word sequences. However, non-consecutive and long-distance semantic information is equally important for learning similarities between malicious webpage source codes, but capturing this is difficult for models such as LSTM. With the help of a GCN, it is possible to capture not only local syntactic and semantic information, but also global syntactic and semantic information, such as global word co-occurrence information, which could further improve the accuracy of malicious webpage detection.

3. Methodology

Here, we provide the detailed framework design and specific methods of GMWD, a malicious web detection system based on GCN and blacklist detection. Figure 1 shows the overall framework of GMWD. It can be seen that the whole detection system is roughly divided into two stages, namely, blacklist detection and GCN detection. Specifically, our input dataset is text files, and each sample contains only its webpage source code without other information, such as its own URL. During the blacklist detection stage, the URL links pointing to other webpages are first extracted from the webpage source code, and then the URLs are quickly detected item by item using the blacklist detection method. If the webpage is judged to be malicious, it is output directly, and if it is unknown, the source code of this webpage is preprocessed and the sliced phrases are added to the corpus. Next, a large and heterogeneous text graph is built based on the whole corpus, then graph convolutional text classification is performed, and, finally, the judgment results are obtained.

We will address each of these two stages in the following two subsections.

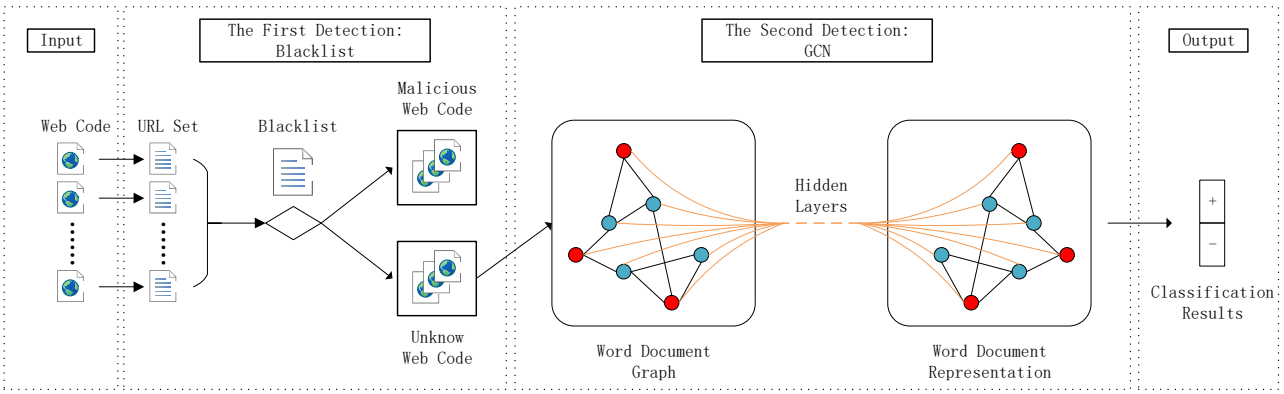

Figure 1. Overall framework of GMWD.

3.1. Quick Detection by Blacklist

The source code of a webpage contains not only the content of this webpage, but also many links pointing to other webpages, some of which are benign and others malicious. When a webpage source code contains URL links pointing to other malicious webpages, we assume that this webpage is probably also malicious or malicious-oriented, and in any case, it is a very dangerous webpage and should be detected by the detection system.

Therefore, we need to detect each URL link contained in the source code of the webpage to be detected. On the other hand, it is relatively reasonable to spend a small amount of time and resources on detecting these URLs, so we choose the blacklist detection method. The advantage of the blacklisting method is that it has a very high accuracy rate and a very low resource and time overhead, which makes it very suitable for the first round of screening in a malicious web detection system.

Firstly, URL links are retrieved from the webpage source code, and then these URLs are detected by the blacklist item by item. Once a URL is found to be a malicious link, the webpage is judged as a malicious webpage. If no malicious link is detected, the webpage is determined as an unknown webpage. After all samples are detected by the blacklist, the unknown webpages are collectively input to the next GCN-based classification stage. The blacklist detection is briefly depicted in Algorithm 1.

Algorithm 1. Detection By Blacklist

Input: A set of webpage source codes *codes*
Output: A set of malicious webpages *malicious_webpages*, A set of unknown webpages *unknown_webpages*

```
for ∀code_i ∈ codes do
    URLs ← get_url(code_i)
    if Blacklist(URLs) then
        Put code_i into malicious_webpages
    else
        Put code_i into unknown_webpages
    end if
end for
return malicious_webpages, unknown_webpages
```

3.2. Detection by GCN

3.2.1. Phrase Segmentation

The presentation of code has a more obvious syntactic format. There are many phrases in code that consist of multiple words together to describe an integrated semantic meaning, and the collocation of these words is relatively fixed. For example, in CSS, when the words "display" and "none" appear together, they generally combine to describe a semantic meaning of "invisible". One way of attack is to set the "display" attribute in the "style" of malicious elements and modules to "none", so that malicious links can be hidden, making it easier for Internet users to click on malicious links, visit them by mistake, and then be attacked by the network. Therefore, in order to maintain the integrity of the syntactic and semantic information of the code and to make full use of the information, it is more reasonable to divide "display" and "none" into a whole phrase rather than into two separate words.

According to [22], during the blacklist detection stage, our work performs phrase slicing based on point-wise mutual information (PMI) and left–right entropy.

The point-wise mutual information reflects the degree of intra-word string binding tightness and represents the degree of interdependence between two variables. Taking the binary point-wise mutual information as an example, the PMI value between words i, j is computed as

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

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

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

where W denotes the total number of sliding windows, $W(i)$ denotes the number of sliding windows containing word i in the whole corpus, and $W(i,j)$ denotes the number of sliding windows containing both word i and word j in the whole corpus. The higher the PMI value, the higher the correlation between word i and word j, and the higher the possibility of word i and word j forming a phrase; conversely, the lower the PMI value, the lower the correlation between word i and word j, and the higher the possibility of a phrase boundary between word i and word j.

Left–right entropy can determine the unstructured nature of the phrase expression and can determine the external boundary of the multiword phrase expression. Left–right entropy is divided into left entropy for the left boundary of a multiword phrase expression and right entropy for the right boundary, which are computed as

$$\mathrm{E_L}(W) = -\sum_{\forall a \in A} P(aW|W) \cdot log_2 P(aW|W), \tag{4}$$

$$\mathrm{E_R}(W) = -\sum_{\forall b \in B} P(Wb|W) \cdot log_2 P(Wb|W), \tag{5}$$

where $\mathrm{E_L}$ and $\mathrm{E_R}$ denote the left entropy and right entropy of the word string, respectively; W denotes the N-gram word string, and $W = \{w_1, w_2, \cdots, w_n\}$; A denotes the set of all words appearing on the left side of the word string, and a denotes a word appearing on the left side; B denotes the set of all words appearing on the right side of the word string, and b denotes a word appearing on the right side. If the values of $\mathrm{E_L}$ and $\mathrm{E_R}$ are larger, i.e., the more words appear on the left and right of the string W, it is more likely that W is a complete multiword expression.

The word strings with PMI higher than P and both left entropy and right entropy higher than N are sliced out as phrases, and the rest are sliced out as words to be used as phrases (words with independent semantics can also be considered as phrases), where P and N are preset values. Then, we add all the sliced results of the webpage source code to a corpus.

3.2.2. Text Graph Building

We build a heterogeneous text graph, which contains two types of nodes: phrase nodes representing all the phrases in the corpus vocabulary, and document nodes representing the source code of all webpages in the corpus.

To capture global phrase–phrase correlations and document–phrase correlations, we build edges among document nodes and phrase nodes (document–phrase edges) and among phrase nodes (phrase–phrase edges). Document–phrase edges are built based on phrase occurrence in documents and edge weights computed by the term frequency–inverse document frequency (TF-IDF) method. Additionally, phrase–phrase edges are built based on local phrase co-occurrence within sliding windows in the corpus, with edge weights computed by PMI. We have already introduced PMI in the previous subsection, which reflects the degree of intra-word string closeness and represents the interdependence between two variables. Formally, the weights of the edges between nodes i and j are defined as

$$A_{ij} = \begin{cases} \mathrm{PMI}(i,j) & i,j \text{ are words, } \mathrm{PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where the formula for PMI(i, j) is given in Formulas (1)–(3) in the previous subsection. The TF-IDF value of a phrase i and a document j can be obtained by

$$\text{TF-IDF}_{ij} = \text{TF}_{ij} * \text{IDF}_i, \tag{7}$$

$$\text{TF}_{ij} = n_{ij}, \tag{8}$$

$$\text{IDF}_i = log \frac{|D|}{\left|\left\{j : t_i \epsilon d_j\right\}\right|}, \tag{9}$$

where n_{ij} denotes the number of occurrences of phrase i in the webpage source code j, $|D|$ denotes the total number of webpage source codes in the corpus, and $\left|\left\{j : t_i \epsilon d_j\right\}\right|$ denotes the number of webpage source codes that contain the phrase t_i.

3.2.3. GCN-Based Classification

Our graph neural network model is adapted from a GCN [6], a semi-supervised learning model for graph-structured data that can operate directly on graphs. Formally, consider a graph $G = (V, E, A)$, where $V(|V| = n)$ represents sets of graph nodes, E represents sets of graph edges, and $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix of G. Since each node should be a self-loop, the diagonal element of A is set to 1. In GCN learning, hidden layer representations are obtained by encoding both features of the nodes and graph structure. The new k-dimensional node feature matrix can be obtained by

$$H^{(l+1)} = f\left(H^{(l)}, A\right) \tag{10}$$

where $H^{(l)} \in \mathbb{R}^{n \times k}$ denotes the feature matrix of the lth layer. Additionally, a commonly used layer-wise propagation rule is

$$H^{(l+1)} = f\left(H^{(l)}, A\right) = \rho\left(\widetilde{A} H^{(l)} W^{(l)}\right) \tag{11}$$

where $\widetilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix; D is the diagonal node degree matrix with $D_{ii} = \sum_j A_{ij}$; $H^{(0)} = X$, $X \in \mathbb{R}^{n \times m}$ is a matrix containing all n nodes with their m-dimensional initial input features; $W^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is a weight matrix, where d_l is the number of features for each node in the lth layer; and ρ is an activation function, such as ReLU.

In this paper, we build a two-layer GCN model. The second layer is connected to the softmax classifier to perform node classification, and the second layer's node embeddings have the same size as the labels set:

$$Z = \text{softmax}\left(\widetilde{A}\ \text{ReLU}\ (\widetilde{A} X W^{(0)}) W^{(1)}\right), \tag{12}$$

where $\text{softmax}(x_i) = \frac{1}{Z} \exp(x_i)$ with $Z = \sum_i \exp(x_i)$. The loss function is defined as the cross-entropy error over all labeled webpage source code documents:

$$\mathcal{L} = -\sum_{w \in y_W} \sum_{f=1}^{F} Y_{wf} \ln Z_{wf} \tag{13}$$

where $\mathcal{Y}_W$ is the set of labeled webpages, and F is the dimension of the output vector, which is equal to the number of classes, i.e., 2 (webpages are either malicious or benign).

In summary, we slice the source code into phrases. Then, we build a large text graph and input it into the GCN. Finally, the GCN model is used to learn the syntactic and semantic correlations within and among webpage source codes to detect malicious webpages. The GCN detection is briefly depicted in Algorithm 2.

Algorithm 2. Detection By GCN

Input: A set of webpage source codes *codes* // i.e., *unknown_webpages* in the output of Algorithm 1
Parameter: Weight matrixes $\{W^{(0)}, W^{(1)}\}$, initial feature matrix X, The number of training epochs *n_epoch*
Output: A set of labels of each webpage Z

for $\forall code\_n \in codes$ **do**
  $phrases \leftarrow get\_phrase(code\_n)$
  for $\forall phrase\_i \in phrases$ **do**
    $phrase\_code\_pair \leftarrow$ TF-IDF$(phrase_i, code_n)$
    for $\forall phrase\_j \in phrases$ **do**
      $phrase\_phrase\_pair \leftarrow$ PMI$(phrase_i, phrase_j)$
    end for
  end for
end for
$V \leftarrow phrases + codes$
$E \leftarrow get\_edge(phrase\_code\_pair, phrase\_phrase\_pair)$
$A \leftarrow get\_adjacency\_matrix(phrase\_code\_pair, phrase\_phrase\_pair)$
for t in $range(n\_epoch)$ **do**
  put $G = (V, E, A)$ into GCN:
  $Z \leftarrow \text{softmax}(\widetilde{A}ReLU(\widetilde{A}XW^{(0)})W^{(1)})$
  $\mathcal{L} \leftarrow -\sum_{w \in y_W} \sum_{f=1}^{F} Y_{wf} \ln Z_{wf}$
  Updating $W^{(0)}$ and $W^{(1)}$ using $\frac{\partial \mathcal{L}}{\partial W^{(0)}}$ and $\frac{\partial \mathcal{L}}{\partial W^{(1)}}$
end for
return Z

4. Experiments

In this section, we analyze the accuracy of the proposed method, GMWD, in different situations. All experiments were conducted on Windows 10, with a 3.6 GHz Intel i5 CPU, 32 GB of RAM, and 1050 NVIDIA GPU.

4.1. Dataset

There is currently no benchmark dataset in the field of malicious webpage detection research. Therefore, we collected experimental datasets from the Malware Domain List, VirusTotal, and other websites by referring to existing research in the field of malicious web detection [23–26]. This is the same method as their experimental dataset acquisition.

The sample size of the dataset used in this paper was 52,448 webpage samples, including 26,896 benign samples and 25,552 malicious samples, as shown in Table 1. The benign samples included 9458 labeled samples from the dataset of the 2017 China Network Security Technology Challenge, and 17,438 test samples that were crawled from the benign platform Chinaz. The malicious samples were all obtained by crawling malicious URLs from the Malware Domain List. We randomly selected 9458 samples as labeled samples, which is the same as the number of benign labeled samples. Additionally, the remaining 16,094 samples were used as test samples. The malicious webpage blacklist was picked up from VirusTotal and contained 300,000 URL data.

4.2. Evaluation Indexes

The purpose of this method is to detect malicious samples, so we consider all malicious samples as positive samples and all benign samples as negative samples.

Table 1. Statistical information of the experimental dataset.

	Benign Samples		Malicious Samples		Blacklist
	Labeled Samples	Test Samples	Labeled Samples	Test Samples	
Source	Competition [1]	Chinaz [2]	Malware Domain List [3]		VirusTotal [4]
Number	9458	17,438	9458	16,094	300,000

[1] 2017 China Network Security Technology Challenge; this competition is organized by the National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC). [2] Chinaz, https://top.chinaz.com (accessed on 3 March 2021). [3] Malware Domain List, http://www.malwaredomainlist.com/mdl.php (accessed on 1 March 2021). [4] VirusTotal, https://www.virustotal.com (accessed on 5 March 2021).

In classification tasks [5,7] and in the field of malicious webpage detection [3,4,27,28], some metrics are commonly used to evaluate the accuracy of detection systems, e.g., ACC, precision, TPR (also known as recall), TNR, FPR, FNR, and F1. In this paper, we use all these commonly used evaluation metrics to demonstrate the effectiveness of our method. Among them, ACC, TPR, and TNR can reflect the detection accuracy of the model to a large extent. Therefore, these three metrics will be highlighted so as to help better focus on the variation in the model's ability to detect malicious and benign samples under different conditions. Descriptions and calculations of the specific metrics are as follows:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}), \tag{14}$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}), \tag{15}$$

$$\text{TPR (Recall)} = \text{TP} / (\text{TP} + \text{FN}), \tag{16}$$

$$\text{TNR} = \text{TN} / (\text{FP} + \text{TN}). \tag{17}$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN}), \tag{18}$$

$$\text{FNR} = \text{FN} / (\text{TP} + \text{FN}), \tag{19}$$

$$\text{F1} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}), \tag{20}$$

where true positive (TP) is the number of malicious samples that are correctly classified as a malicious webpage, true negative (TN) is the number of benign samples that are correctly classified as a benign webpage, false positive (FP) is the number of benign samples that are incorrectly classified as a malicious webpage, and false negative (FN) is the number of malicious samples that are incorrectly classified as a benign webpage.

4.3. Detection by Blacklist

The source code of a webpage contains not only the content of the webpage, but also the URL links pointing to other webpages. This URL information can help us detect malicious webpages. When the source code of a webpage contains URL links pointing to a malicious webpage, the webpage is often a malicious webpage as well. Therefore, we need to extract the URLs from the source code of each webpage and then carry out blacklist detection for each URL. Blacklist detection has the advantages of high accuracy and fast detection, which can quickly and correctly detect some malicious webpages. The blacklist used in this paper was from VirusTotal, with a total of 300,000 malicious URL data.

We performed URL extraction on a total of 33,532 webpages in all test sets and matched them with the blacklist item by item, detecting 476 malicious samples with a detection rate of about 2.96%, all of which were detected correctly. This shows that the scheme to detect malicious webpages based on URL information in webpages is feasible. It is worth noting that the correctness rate of this experiment is greatly limited by the size and characteristics (e.g., old or new) of the blacklist. Choosing the right blacklist can effectively increase the detection rate and thus improve the overall correctness of the detection system.

4.4. Detection by GCN

The dataset we used was the total dataset of 51,972 samples after removing the malicious samples screened by the blacklist detection. The source text was first cleaned, e.g., by removing comments from the code and removing low-frequency phrases that are contained in less than 0.1% of the total source text. Then, we preprocessed the dataset to perform phrase slicing on the source text.

Settings. In the construction of the text graph, we set the embedding size of the initial phrase embeddings to 15 and set the sliding window size for statistical word co-occurrence to 20, as used in [7]. In the training process, the dropout rate was 0.5. A maximum of 400 epochs and the Adam optimizer with a learning rate of 0.02 were used, and training was stopped early if the validation loss did not decrease for 10 consecutive epochs.

We used GMWD for malicious webpage detection experiments to verify its performance. The experimental results are shown in Table 2. The experimental results show that GMWD achieved good results in the detection ability of both malicious web samples and benign web samples, with TPR, TNR, and ACC exceeding 99.8%, which demonstrates the effectiveness and accuracy of the proposed method in malicious webpage detection.

Table 2. Result of detection by GCN.

Malicious Samples		Benign Samples		ACC	Precision	TPR	TNR	F1
TP	FN	TN	FP					
15,595	23	17,416	22	0.9986	0.9986	0.9985	0.9987	0.9986

The main reasons why GMWD performs well are as follows. (1) The model inherits the advantages of a GCN for node classification: graph convolution can capture both document–phrase and global phrase–phrase relationships while combining its own information with that of its multi-order neighbor nodes through the advantage of information transfer from the graph model. (2) The single word corrupts the semantic information of the code to a certain extent, while the phrase better maintains the semantic information of the code text.

Effect of the Number of Labeled Data. To evaluate the effect of the number of labeled data, we tested GMWD with seven sets of labeled data of different orders of magnitude, with 30, 500, 2000, 4000, 6000, 8000, and 9458 positive and negative samples each. The test results are presented in Figure 2. We can see that the overall TPR, TNR, and ACC roughly show an increasing curve as the number of labeled data increases. Specifically, when there are only 500 positive and negative samples each, the TPR reaches 99.15%, although the TNR is only 96.65%, which is still a relatively good result if we consider that the success rate of detecting malicious webpages is more important in practical applications. When the labeled data reach 4000 positive and negative samples each, TPR, TNR, and ACC all exceed 99%. When the number of positive and negative samples reaches 9458, TPR, TNR, and ACC all exceed 99.85%.

Effect of Different Sliding Window Sizes. In order to evaluate the effect of different sliding window sizes, we tested GMWD with different sliding window sizes, and to make the comparison of the results more obvious, the tests were performed under the condition that the labeled data were 4000 positive and negative samples each. The test results are presented in Figure 3, which shows that the test accuracy first increases as the window size becomes larger, but the average accuracy decreases when the window size is larger than 8. This indicates that too small a window size causes the GCN to acquire semantic information between phrases ineffectively, while too large a window size may cause the GCN to overlearn the correlation between less closely related nodes.

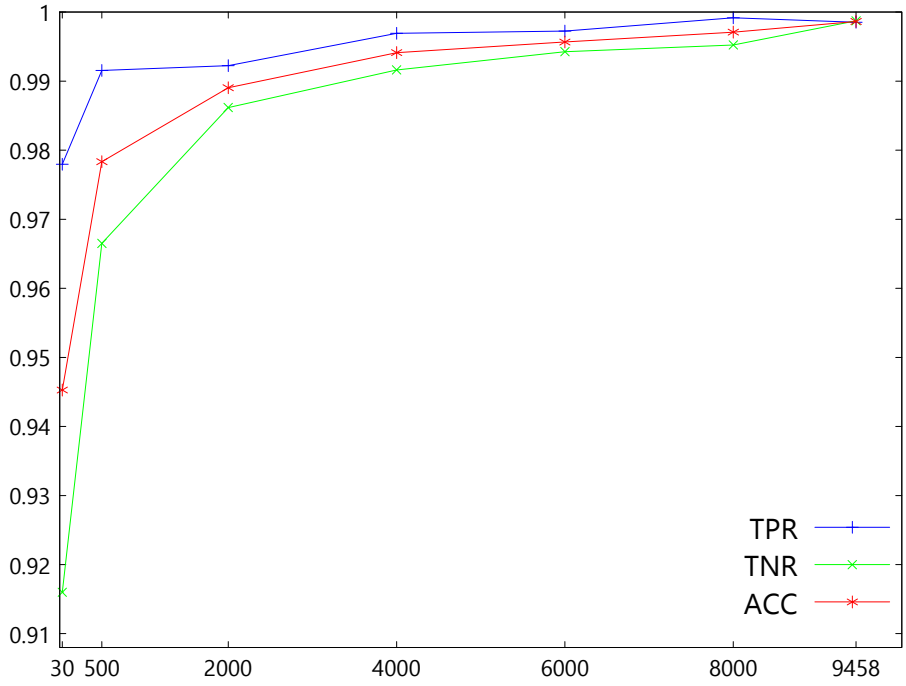

Figure 2. Test accuracy on labeled samples with different orders of magnitude.

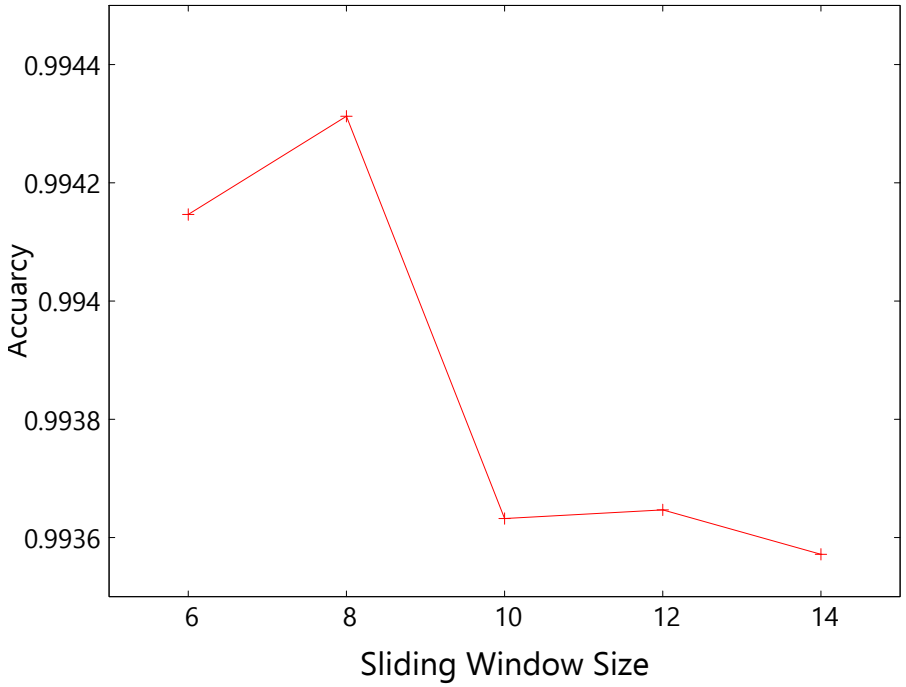

Figure 3. Test accuracy with different sliding window sizes.

Effect of Using Phrase Nodes. To evaluate the impact of using phrases instead of words as nodes in malicious webpage detection, we compared GMWD using phrase nodes with that using word nodes and conducted detection experiments with different orders of magnitude of markup data. The results are presented in Figure 4. From the results, we can see that the TPR of the phrase node group is much higher than that of the word node group when the number of labeled samples is small, and the gap gradually decreases as the number of labeled samples increases. The TNR of the word node group is much higher than that of the phrase node group when the number of labeled samples is small, the gap gradually decreases as the number of labeled samples increases, and the two are almost

equal after the labeled data reach 8000 positive and negative samples each. The ACC of the word node group is higher than that of the phrase node group overall.

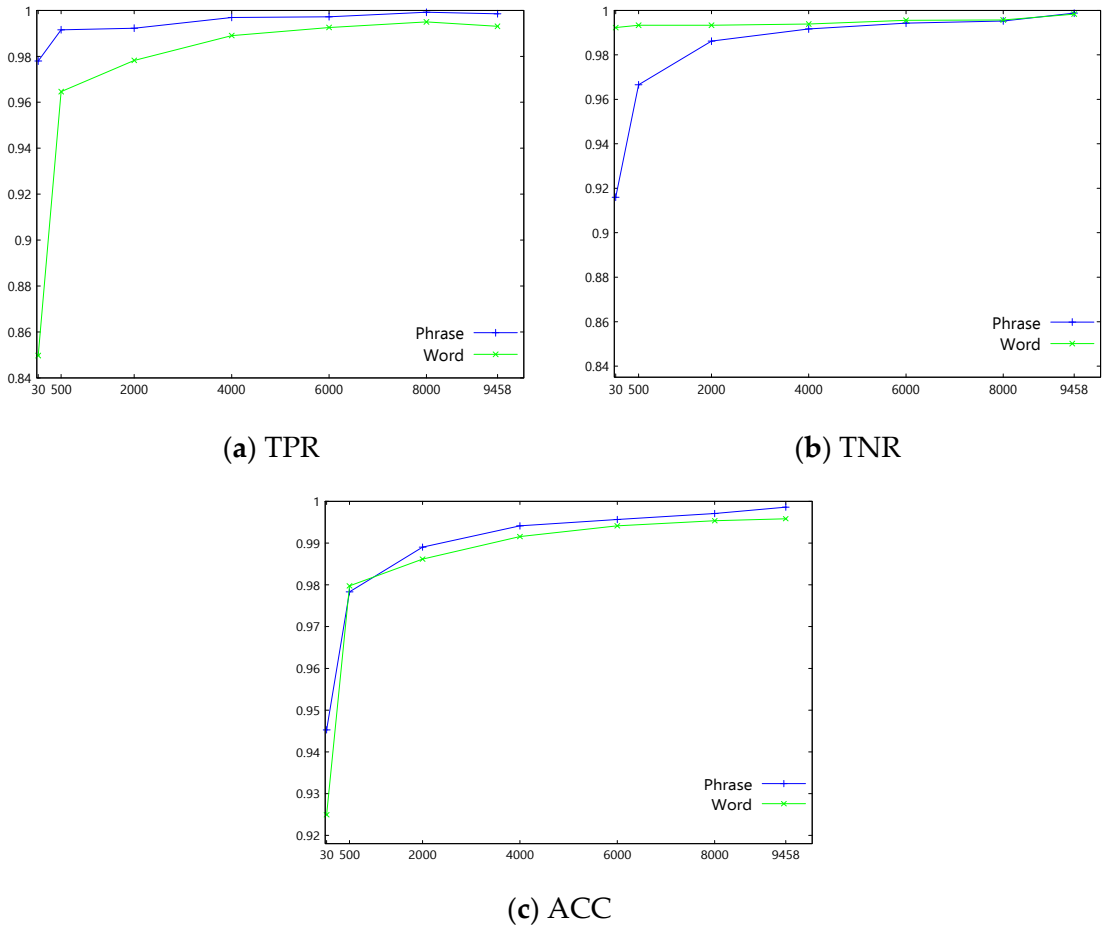

Figure 4. Test performance with different node types, where the blue line represents GMWD with phrases as nodes and the green line represents the control group with words as nodes. (**a**–**c**) show the test TPR, TNR, and ACC with different node types, respectively.

The difference between a malicious webpage source code and benign webpage source code is actually reflected in the presence of malicious code in the malicious webpage, while the rest of the malicious webpage source code is the same as a general benign source code. Therefore, to effectively detect malicious webpages, the features of malicious code must be learned by allowing the GCN to capture the similarities between malicious codes.

The experimental results show that using words as a node causes the system to have a low false alarm rate but a high miss rate when the number of labeled samples is small. This means that the GCN is not able to sufficiently capture and learn the features of malicious code, so it is biased to judge the webpages as benign. Meanwhile, when using phrases as a node, there is a high false alarm rate, but a low miss rate, which indicates that the system is better able to capture and learn the features of the malicious code. This proves our conjecture that slicing the source code into individual words will lose some syntactic and semantic information, and slicing the source code into phrases can better maintain the syntactic and semantic information of the source code. This information enables the GCN to better capture and learn the features of malicious code to detect malicious webpages.

Analysis of Phrases. We used the t-SNE tool [29] to visualize the second layer's learned phrase embeddings, as shown in Figure 5. The orange dots represent the nodes of phrases with higher values classified as malicious than as benign. We note that phrases with malicious labels are close to each other rather than randomly distributed, which means that phrases with malicious labels are always closely related to each other rather than

existing separately. In addition, we show a list of the nine phrases with the largest values of malicious labels in Table 3. We found that the phrases are interpretable. For example, "wp-content" is often found in malicious links. According to our statistics on the dataset, this phrase was present in 21 of the 25,076 benign samples in the dataset, a total of 93 times, while it was present in 3888 of the 26,896 malicious samples, a total of 12,115 times.

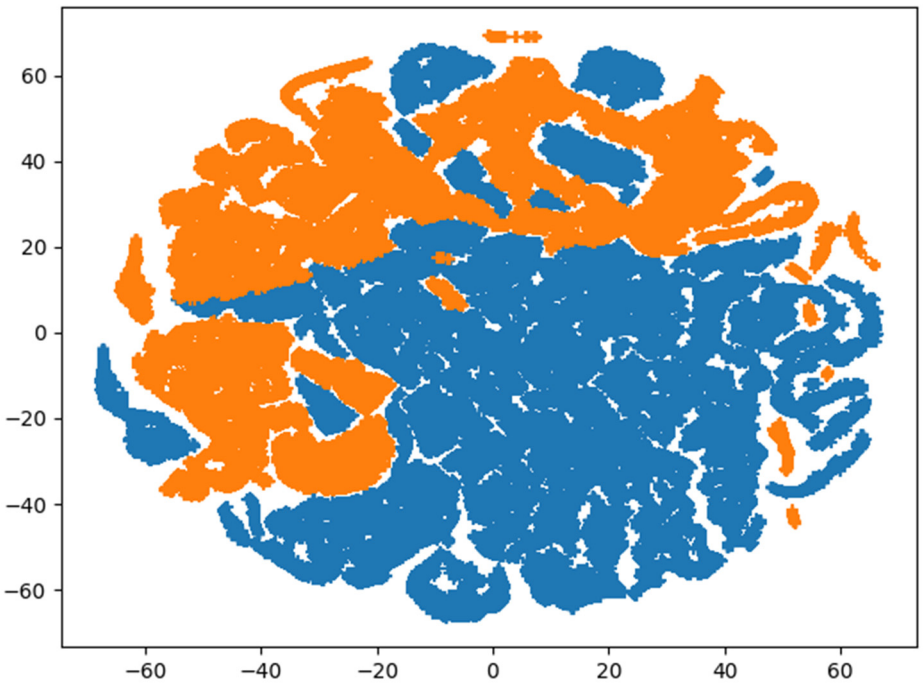

Figure 5. The t-SNE visualization of the second layer's learned phrase embeddings. The orange dots represent the nodes of phrases with higher values classified as malicious than as benign.

Table 3. Nine phrases with the largest values of malicious labels.

Top 9 Phrases with Malicious Labels		
1. wp-content	2. quot	3. menu-item
4. elementor-element	5. path	6. important
7. data-element_type	8. not	9. border-color

4.5. Comparison of Related Work

We compared GMWD with other related malicious web detection methods. The authors of [4] proposed a malicious JavaScript detection model based on LSTM. The model extracts features from the semantic level of bytecode and optimizes the word vector approach. The authors of [12] combined features extracted from abstract syntax trees with random forest classes to detect malicious JavaScript instances. The authors of [21] proposed a multi-dimensional feature phishing detection method based on deep learning, mainly for webpage URLs and webpage source code. The authors of [25] detected malicious webpages by operating directly on language-independent token streams that are extracted directly from static HTML files using simple regular expressions. In order to present the comparison results more comprehensively, we list all the common evaluation indicators. The details are as follows.

The result shows that our scheme can accurately perform the malicious webpage detection task. As shown in Table 4, our scheme outperforms the methods proposed by [4] [12], and [21] in all metrics. The reason is that the network models they use, such as LSTM, only capture the local semantic information. In contrast, with the help of the GCN, we capture not only local semantic information but also global semantic information, which could further improve the accuracy of malicious webpage detection. The FPR of [25]

was able to reach 0.1%, which is lower than that of our method. The low FPR of [25] has to be taken with a grain of salt; like the majority of anti-virus systems, they rather traded a low false positive rate for a higher false negative rate. In other metrics, our approach performs better.

Table 4. Accuracy of different malicious web detection models. The best indicates in boldface.

Method	ACC (%)	Precision (%)	FNR (%)	FPR (%)	F1 (%)
Ref. [4]	99.51	99.55	3.79	——	98.37
Ref. [12]	99.50	——	0.54	0.52	——
Ref. [21]	98.99	99.41	1.43	0.59	99.0
Ref. [25]	97.50	——	——	**0.1**	——
GMWD	**99.866**	**99.859**	**0.137**	0.126	**99.860**

5. Conclusions and Future Work

With the widespread popularity of the Internet, malicious webpages are becoming one of the main ways to seriously threaten user information security and privacy. In this work, we proposed a GCN-based malicious webpage detection method, GMWD. This method, unlike traditional malicious webpage detection methods, is not interfered with by attacker obfuscation techniques. We replace word nodes in the text graph with phrase nodes to better maintain the syntactic and semantic integrity of the webpage source code. In addition, we also utilize the URLs appearing in the webpage source code as auxiliary detection information. The experimental results show that GMWD has higher accuracy and a lower false negative rate compared to other existing malicious webpage detection schemes.

This paper used the GCN model for detection, which requires preprocessing and building a text graph of the webpages in the dataset at first. During subsequent detection, additional pages cannot be dynamically inserted into the graph, which means that pages not included in the initial dataset cannot be detected. Therefore, one of our future works is to explore how to detect single webpages outside the dataset. Moreover, there are many types of malicious codes, and the syntax and semantics of different types of malicious codes differ. The labeled samples used in our experiments were randomly selected, and there may be cases where the labeled samples do not contain all malicious types, which brings certain limitations and difficulties to improving the experimental effect. Generating a suitable labeled sample dataset, especially when the size of the labeled sample is small, is a way to improve the accuracy of the model in the future.

Author Contributions: Conceptualization, Y.W. and S.X.; methodology, Y.W., S.X. and J.S.; software, S.X. and J.S.; validation, S.X., Y.W. and J.S.; formal analysis, S.X. and J.S.; investigation, S.X. and J.S.; resources, Y.W. and J.S.; data curation, Y.W.; writing—original draft preparation, Y.W. and J.S.; writing—review and editing, S.X., J.S. and Y.W.; visualization, S.X.; supervision, J.S.; project administration, S.X. and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in our paper:

GCN	Graph convolutional network
GMWD	The GCN-based malicious webpage detection method proposed in this paper
URL	Uniform Resource Location
CNN	Convolutional neural network
LSTM	Long short-term memory
PMI	Point-wise mutual information
TF-IDF	Term frequency–inverse document frequency
TP	True positive
FN	False negative
TN	True negative
FP	False positive
ACC	Accuracy rate
TPR	True positive rate
TNR	True negative rate
FPR	False positive rate
FNR	False negative rate

References

1. China Internet Network Information Center. *The 47th Statistical Report on China's Internet Development*; China Internet Network Information Center: Beijing, China, 2021.
2. Guo, Y.; Marco-Gisbert, H.; Keir, P. Mitigating webshell attacks through machine learning techniques. *Future Internet* **2020**, *12*, 12. [CrossRef]
3. Song, X.; Chen, C.; Cui, B.; Fu, J. Malicious JavaScript detection based on bidirectional LSTM model. *Appl. Sci.* **2020**, *10*, 3440. [CrossRef]
4. Fang, Y.; Huang, C.; Liu, L.; Xue, M. Research on malicious JavaScript detection technology based on LSTM. *IEEE Access* **2018**, *6*, 59118–59125. [CrossRef]
5. Liu, X.; You, X.; Zhang, X.; Wu, J.; Lv, P. Tensor graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), New York, NY, USA, 7–12 February 2020; pp. 8409–8416.
6. Welling, M.; Kipf, T.N. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
7. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), Atlanta, GA, USA, 8–12 October 2019; pp. 7370–7377.
8. Manjeri, A.S.; Kaushik, R.; Ajay, M.; Nair, P.C. A machine learning approach for detecting malicious websites using URL features. In Proceedings of the 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 12–14 June 2019; pp. 555–561.
9. Chiramdasu, R.; Srivastava, G.; Bhattacharya, S.; Reddy, P.K.; Gadekallu, T.R. Malicious url detection using logistic regression. In Proceedings of the 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), Barcelona, Spain, 23–25 August 2021; pp. 1–6.
10. Lee, S.; Kim, J. Warningbird: A near real-time detection system for suspicious urls in twitter stream. *IEEE Trans. Dependable Secur. Comput.* **2013**, *10*, 183–195. [CrossRef]
11. Jain, A.K.; Gupta, B.B. Towards detection of phishing websites on client-side using machine learning based approach. *Telecommun. Syst.* **2018**, *68*, 687–700. [CrossRef]
12. Fass, A.; Krawczyk, R.P.; Backes, M.; Stock, B. Jast: Fully syntactic detection of malicious (obfuscated) javascript. In Proceedings of the International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (ICNSC), Paris, France, 28–29 June 2018; pp. 303–325.
13. Altay, B.; Dokeroglu, T.; Cosar, A. Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection. *Soft Comput.* **2019**, *23*, 4177–4191. [CrossRef]
14. Agor, J.; Özaltın, O.Y. Feature selection for classification models via bilevel optimization. *Comput. Oper. Res.* **2019**, *106*, 156–168. [CrossRef]
15. Liu, W.; Wang, J. A brief survey on nature-inspired metaheuristics for feature selection in classification in this decade. In Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), Banff, AB, Canada, 9–11 May 2019; pp. 424–429.
16. Zhang, C.; Liu, C.; Zhang, X.; Almpanidis, G. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Syst. Appl.* **2017**, *82*, 128–150. [CrossRef]
17. Louati, F.; Ktata, F.B. A deep learning-based multi-agent system for intrusion detection. *SN Appl. Sci.* **2020**, *2*, 675. [CrossRef]
18. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.

19. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
20. Le, H.; Pham, H.Q.; Sahoo, D.; Hoi, S.C. URLNet: Learning a URL representation with deep learning for malicious URL detection. In Proceedings of the PODS 2017: ACM Symposium on Principles of Distributed Computing, Washington, DC, USA, 25 July 2017; pp. 1–13.
21. Yang, P.; Zhao, G.; Zeng, P. Phishing website detection based on multidimensional features driven by deep learning. *IEEE Access* **2019**, *7*, 15196–15209. [CrossRef]
22. HongYing, Z.; Yang, Y.; ZHANG, J.; Jun, Z.; YouLang, J.; ZhenDong, D.; QingChen, W. Application of term library construction based on machine learning and statistical method in intelligent power grid custom service. In Proceedings of the 2019 IEEE 8th International Conference on Advanced Power System Automation and Protection (APAP), Xi'an, China, 21–24 October 2019; pp. 145–149.
23. Liu, Y.; Zhu, C.; Wu, Y.; Xu, H.; Song, J. MMWD: An efficient mobile malicious webpage detection framework based on deep learning and edge cloud. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6191. [CrossRef]
24. Rozi, M.; Ban, T.; Kim, S.; Ozawa, S.; Takahashi, T.; Inoue, D. Detecting Malicious Websites Based on JavaScript Content Analysis. In Proceedings of the Computer Security Symposium, Kamakura, Japan, 21–24 June 2021; pp. 727–732.
25. Saxe, J.; Harang, R.; Wild, C.; Sanders, H. A deep learning approach to fast, format-agnostic detection of malicious web content. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24–24 May 2018; pp. 8–14.
26. Vazhayil, A.; Vinayakumar, R.; Soman, K. Comparative study of the detection of malicious URLs using shallow and deep networks. In Proceedings of the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018; pp. 1–6.
27. Wang, R.; Zhu, Y.; Tan, J.; Zhou, B. Detection of malicious web pages based on hybrid analysis. *J. Inf. Secur. Appl.* **2017**, *35*, 68–74. [CrossRef]
28. Yi, P.; Guan, Y.; Zou, F.; Yao, Y.; Wang, W.; Zhu, T. Web phishing detection using a deep learning framework. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1–9. [CrossRef]
29. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.