The Journal of Supercomputing
https://doi.org/10.1007/s11227-021-04020-y

Intelligent malware detection based on graph convolutional network

Shanxi Li[1] · Qingguo Zhou[1] · Rui Zhou[1] · Qingquan Lv[1]

Accepted: 9 July 2021

Abstract

Malware has seriously threatened the safety of computer systems for a long time. Due to the rapid development of anti-detection technology, traditional detection methods based on static analysis and dynamic analysis have limited effects. With its better predictive performance, AI-based malware detection has been increasingly used to deal with malware in recent years. However, due to the diversity of malware, it is difficult to extract feature from malware, which make malware detection not conductive to the application of AI technology. To solve the problem, a malware classifier based on graph convolutional network is designed to adapt to the difference of malware characteristics. The specific method is to firstly extract the API call sequence from the malware code and generate a directed cycle graph, then use the Markov chain and principal component analysis method to extract the feature map of the graph, and design a classifier based on graph convolutional network, and finally analyze and compare the performance of the method. The results show that the method has better performance in most detection, and the highest accuracy is 98.32%, compared with existing methods, our model is superior to other methods in terms of FPR and accuracy. It is also stable to deal with the development and growth of malware.

Keywords Malware detection · Directed cyclic graph · Markov chain · Graph convolutional network · Machine learning

✉ Qingguo Zhou
zhouqg@lzu.edu.cn

Shanxi Li
lisx@lzu.edu.cn

Rui Zhou
zr@lzu.edu.cn

Qingquan Lv
lvqq18@lzu.edu.cn

[1] School of Information Science and Engineering, Lanzhou University, Lanzhou, People's Republic of China

Published online: 24 August 2021

1 Introduction

Since 2020, the "COVID-19" virus has spread throughout the world, seriously affecting the health of all mankind and the normal development of society. The virus has caused unprecedented disaster and panic to mankind. In the cyberspace, computer viruses and other malicious software are also trying to break through the computer's defenses time and time again, causing damage to the computer world.

Although information security departments spend a lot of money to maintain network security every year, researchers have been trying to use more advanced technology to solve the problem of information security, and have achieved a lot of research results, which effectively curb the harm caused by malicious network attacks. However, due to the particularity of information technology, malware developers often look for a breakthrough by actively looking for system vulnerabilities. On the other hand, the security department often studies the measures to identify and defend the malware only after the system is broken and the loss of users is caused. Therefore, the current security measures are often in the state of "hindsight" and passive defense. How to predict the characteristic information of malware, detect malware in advance, and nip the spread of malware in the bud has become an important topic of information security research.

Most of the defense systems still use static analysis as their primary measurement, which is mainly based on signature matching [1]. Some defense systems have developed dynamic analysis, including sensitive behaviors, access to critical privileges, network analysis, and key process monitor as their assistant method [2, 3]. However, all of these methods are mainly focused on specific malware or malware classes so that they are limited when defensing new types or variants of malware. Besides, they are also weak to the anti-detection techniques, which would let the detectors be deceived by disguised malware and cause damage. All of the situations indicate that developing a new method of detection is essential.

In recent years, deep learning technology has shown an unprecedented degree of intelligence. Through the analysis of a large number of historical data, it can find the rules and predict the unknown samples. It shows strong adaptability, prediction ability and intelligent level, and has achieved good results in computer vision, natural language processing, speech recognition and other fields. At the same time, it provides an effective means for the detection and prevention of malware, which can not only accurately detect malware, but also prospectively predict the maliciousness of unknown software.

Aiming at solving the problems of traditional static detection and dynamic detection methods, this paper proposes a novel approach of malware detection based on application programming interface (API) call sequence and deep learning algorithm. Firstly, the API call relation is extracted, and the ordered cycle graph is constructed based on Markov chain. Then, the graph convolution neural network (GCN) is used to detect malware. The performance analysis and comparison are carried out. Consequently, the main contributions of our work are listed as follows:

- We present a new method to extract features from samples with three-dimensional structure. Firstly, we extract the weight model of malware samples based on Markov chain by using the API call information of a large number of known malware samples, and then use the samples to be detected to map in the weight model, so as to extract the features of the samples to be detected. The model combines the features of test samples with the general features of malware, so that the newly generated features can not only maintain the features of the samples themselves, but also increase the generality of the features, and effectively resist the disguise and variants of malware.
- A malicious code detection method based on GCN is proposed. Taking the characteristic graph of malicious code as the input, the graph convolution neural network is trained and tested, and the malware detection model based on GCN is established. The model takes advantage of the graph as input of GCN to improve the adaptability of the model in detecting malicious code.

The remainder of the paper is organized as follows. Some related works are reviewed in Sect. 2. System framework ,workflow and the method of detection are presented in Sect. 3. The data set and experimental environment are introduced in Sect. 4. The details of experiments and results are shown in Sect. 5, and the whole work is discussed and concluded in Sect. 6.

2 Related work

2.1 Development trend of malware

Since the birth of computer, malicious threat has been accompanied by the development of computer. As early as 1949, John von Neumann's paper "theory and organization of complex Automata" [4] mentioned the assumption of how computer programs can achieve self-replication, which can actually be regarded as the germination of malicious code. In 1970, Bob Thomas, a developer of BBN technologies, created a program called creeper, which can realize self-replication and continue to spread through ARPANET network. In 1983, Fred Cohen wrote the first computer virus program recognized in history, which can realize self-replication and spread [5]. Then, computer attack and computer defense has gradually developed into a huge industry, and the scale of the industry is also higher and higher. Malicious code refers to software with malicious intention, such as computer virus, worms, spyware, browser hijackers, adware and track software, which can control and destroy the user's computer, data and network, and damage the user's interests. Nowadays, with the continuous improvement of malicious code detection technology and the development of artificial intelligence technology, the technical level of malicious code is becoming more and more advanced. We try every means to avoid the detection of detection software by using escape strategies. These strategies are smarter and more hidden than many conventional anti-malicious code systems [6, 7]. Generally speaking, the development trend of malicious code is as follows: (1) the forms of attacks are diverse, and the complexity of threat capability is increasing. (2) Malicious code

attacks tend to be intelligent. (3) Malicious code has been fully industrialized. (4) Malicious code attacks are organized. (5) The ability of anti-detection has improved significantly.

2.2 Traditional detection methods for malware

In 1987, Fred Cohen put forward the concept of computer virus for the first time, and put forward the basic theory of computer virus detection and defense, established the basis of program behavior detection, put forward a series of defense schemes, opened the road of computer Malware Defense Research. In recent decades, researchers have explored a series of methods and techniques to detect malware after a lot of work. Detection of malware is mainly to detect the characteristic code of malware, which mainly includes anomaly-based detection and signature-based detection.

Anomaly-based detection technology is to check the maliciousness of the program by detecting the difference between the behavior of the abnormal program and that of the normal program. Generally, the behavior trajectory of malware is different from that of normal software. After fully understanding the behavior of normal program, a set of standards and specifications will be formed. If the behavior trajectory of the program to be detected is abnormal and violates this set of specifications, it can be determined as malware. There are three different detection methods for anomaly-based detection: static detection, dynamic detection and hybrid detection.

Sundarkumar et al. proposed a model based on API call sequence type, which uses text mining and topic modeling to detect malware [8]. After analysis, it suggests to use decision tree to design malware detection expert system. Wu Songyang and others used the data stream application program interface (API) as the classification function, and adopted the improved k nearest neighbor classification model to detect Android malicious code. Through machine learning, the API list related to data flow is further optimized, and the efficiency of sensitive data transmission and analysis is significantly improved [9].

2.3 AI-based malware detection technology

Although traditional methods play a very important role in malicious code detection, they have also made some achievements [10]. However, because malicious code writers often use various means to avoid the traditional detection methods, or study some new types of malicious code or variants of malicious code, the accuracy of the traditional detection model will be greatly reduced in these cases. With the continuous development of machine learning technology, malware detection technology based on machine learning model has also developed and achieved some successful results.

Schultz et al. introduced machine learning based on static features to detect unknown malware, by using program executables (PE), byte n-gram and string for feature extraction [11]. Elovici et al. used PE and Fisher score (FS) method for feature selection, and used artificial neural networks (ANN), Bayesian network (BN),

decision tree (DT) and other methods to detect malicious software, with an accuracy of 95.8%. Moskovitch et al. used filter method for feature selection [12]. They used gain ratio (GR) and Fisher score for feature selection, and used artificial neural network, decision tree, Naive Bayes (NB) and support vector machine (SVM) classifier to detect malware, with an accuracy of 94.9%. They also put forward a method, using n-gram operation code as feature, using document frequency (DF), GR and FS as feature selection method, using artificial neural network, decision tree, naive Bayes and other classification algorithms, in the case of poor performance of ANN, DT, Boosted DT(BDT), to keep a lower false alarm rate level [13].

Santos et al. proposed supervised learning to detect unknown malware [14]. They use information gain method for feature selection, and use different classifiers, such as DT, k-nearest neighbor (KNN), BN, SVM, in which SVM shows good accuracy. Ivan firdausi et al [15]. designed malware detection technology by using five classifiers including KNN, NB, J48 DT, SVM and MLP. The experimental results show that J48 DT achieves the best overall performance, with the recall rate of 95.9%, false alarm rate of 2.4%, accuracy of 97.3% and accuracy of 96.8%. In a word, it can be concluded that the proof of concept can detect malware very effectively based on the use of automatic behavior-based malware analysis and machine learning technology.

Konrad Rieck et al. proposed a framework for automatically analyzing malware behavior using machine learning [16]. The framework can automatically identify new malware categories (clusters) with similar behaviors, and assign unknown malware to these discovered categories (classifications). Based on clustering and classification, an incremental method based on behavior analysis was proposed, which can process the behavior of thousands of malware binaries every day.

In order to facilitate more researchers to use machine learning model to study malicious code detection technology, Anderson et al. [17] provided a malicious code benchmark dataset ember for machine learning, and then demonstrated a use case using LightGBM and baseline gradient-enhanced decision tree model trained by default. The author also proposed a general framework based on reinforcement learning (RL) for static PE anti-malware engine. Through training with the anti-malware engine, the general framework understands which operation sequences may lead to avoiding the attack of any given malware sample in the detector, thus generating malware samples that can avoid detection, which provides a reference for the design of more advanced detection software. Sanjay Sharma et al. proposed a method based on the appearance of opcodes to improve the accuracy of malware detection for unknown advanced malware. This method uses the Fisher scoring method for feature selection, and uses five machine learning classifier algorithms to detect unknown malware. Among them, random forest, LMT, J48 DT and NB have reached 100% accuracy [18].

Smita Naval et. [19] proposed a new model based on an improved API sequence. The method was tested through a series of experiments, and the results were compared with existing malicious code detectors, which proved the effectiveness of the method.

Tang et al. [20] proposed a new method of static malicious code detection based on the API call sequence: firstly, extract the API sequence through dynamic

analysis, and then convert the sequence into a characteristic image that can represent the behavior of the malicious code. Finally, the convolutional neural network (CNN) is used to classify the malicious code into nine malicious code families. The results showed that the TPR indicator exceeded 99%.

Li Jin et al. proposed a malicious code detection system based on permission usage analysis-SigPID. It uses machine learning-based classification methods to classify different families of malware and benign applications. Experimental results show that it can achieve an accuracy of more than 96%.

In the work of Raff et al. [21], they used neural networks to detect malicious code at the entire executable file level. This solution avoids many of the problems of the more common byte n-gram method, but it achieves consistent generalization on both test sets.

Alzaylaee et al. [22] proposed a malicious code detection system based on deep learning, DL-Droid, to dynamically detect malicious Android applications, using dynamic features to achieve a detection rate of 97.8%.

As a new type of neural network architecture, graph neural network (GNN) has been widely used in many industries in recent years [23], but there is still little research on malware detection. Since GNN uses graphs as input, which is related to the API call sequence graph studied in this paper, so we try to use a special type of graph convolution network (GCN) in GNN.

3 Malware detection algorithm based on graph convolutional network

In this section, we will introduce data preprocessing method and the detector based on GCN proposed in this paper. First, we introduce the framework of the system, and then we introduce the workflow.

3.1 System framework and workflow

This system mainly includes four steps, namely the extraction of API call sequence, the generation of directed cycle graph, the Markov process and classification. The malware detection system framework is shown in Fig. 1.

The system workflow is shown in Fig. 2. In the figure, the system process is shown on the left, the highlight work has been shown on the right side of the figure, and the corresponding steps are indicated by blue curves. The sample is first executed in the sandbox, and the API called by the sample is extracted from it, and then the API is used as the vertex of the graph, and the number of times the API calls other APIs is used as the weight of the directed edge, thereby establishing a directed cyclic graph. Then, extract the feature map based on the Markov chain, and the GCN classifier is used for classification.

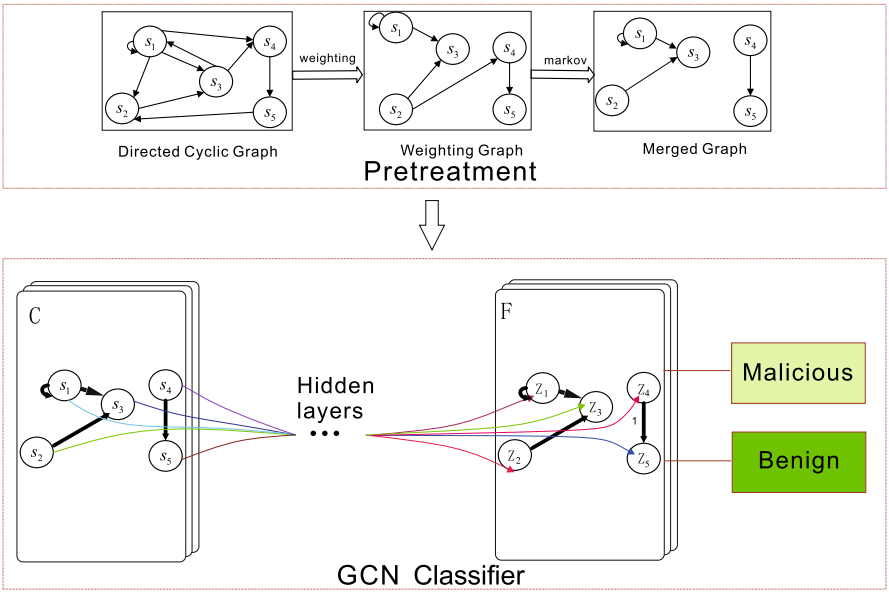


Fig. 1 GCN-based malware detection system framework

3.2 Algorithm for generating directed cyclic graph of API based on Markov chain

A directed acyclic graph (DAG) is a sequential graph with a limited set of point S and a set of directed edge E, while the arbitrary edge e directed from a vertex s_i to another vertex s_j ($s_i, s_j \in S$), and arbitrary vertex could not return itself with the directed edge sequences [24]. DAG can represent the set of possibilities that satisfied Markov property, which clarifies that the probability of transitioning from a state to another only depends on the current state.

Directed acyclic graph (DAG) is widely used in malicious code classification based on API call sequence, but it is not convenient to use in API sequence with cyclic call, so it needs to use directed cyclic graph (DCG).

DCG is similar to DAG, but allows loops from vertex to itself. In our work, the DCG will be generated from API call sequences. The graph consists of vertices s that presents API and edges e to represent invokes, and the edges in the DCGs will be weighted according to the invokes chains of the samples. The structure of DCG is shown in Fig. 3, where the vertex (API) and the relation among vertexes (invokes) are presented concisely. For an edge e_{ij} in the graph G, the weight of the edge would be the number of invokes that from API s_i to s_j, it is labeled n_{ij} in the figure.

For the convenience of model processing, a DCG will be presented as an adjacent matrix, which is given in Table 1. In the matrix, the row refers to the vertex started, namely API that make calls, and the column refers to the vertex ended, i.e., API that is called. The weights of the edges are stored in the cell of the matrix, which means the number that the call occur [25].

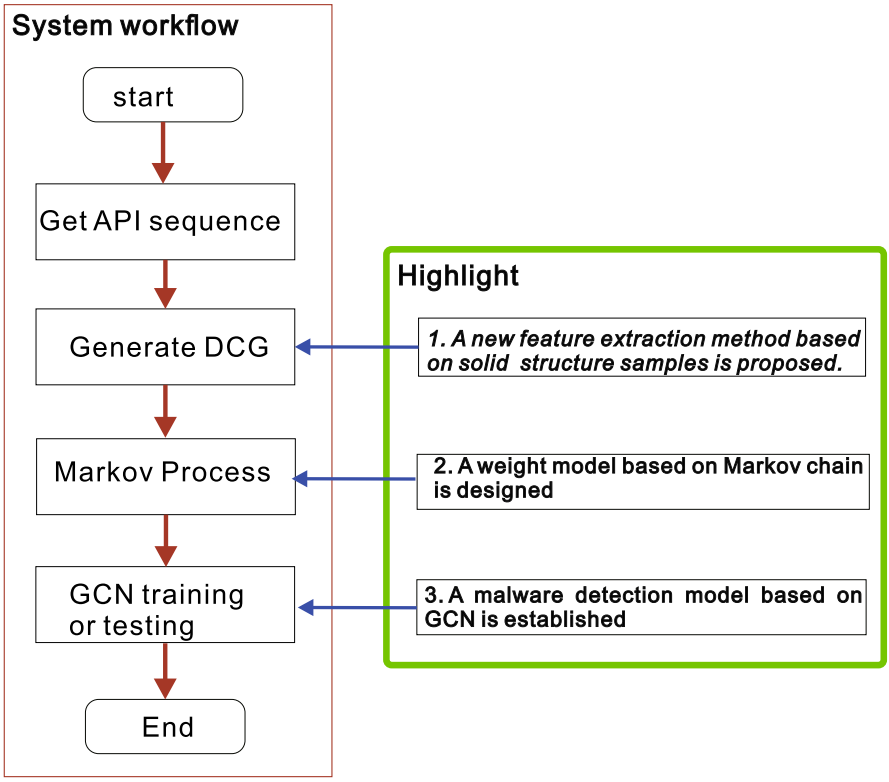


Fig. 2 GCN-based malware detection system workflow

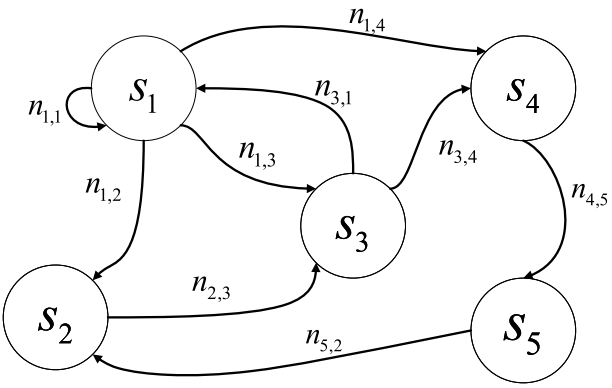


Fig. 3 Structure of DCG

When the scale of API call is large, it is not conducive to feature extraction. Fortunately, API calls often satisfy Markov property. In order to extract features of API calls effectively, we use Markov chain to extract features and simplify models, so as to facilitate the classification.

Table 1 Adjacent matrix of the DCG in Fig. 3

	S_1	S_2	S_3	S_4	S_5
S_1	$n_{1,1}$	$n_{1,2}$	$n_{1,3}$	$n_{1,4}$	-
S_2	–	–	$n_{2,3}$	–	$n_{2,5}$
S_3	$n_{3,1}$	–	–	$n_{3,4}$	–
S_4	–	–	–	–	$n_{4,5}$
S_5	–	$n_{5,2}$	–	–	–

Markov chain is a memory-less stochastic process that satisfies Markov property. Due to the characteristic of Markov Chain that can simplify the features of sequences, it is broadly used in the classification and processing of sequential data, especially dynamic detection [26, 27].

To calculate the weight, firstly, an original malware dataset would be used to generate a Markov Chain. The dataset must be rich enough to present the general characteristics of malware, and the chain can be defined as "weighting graph," which can be labeled as M_w. Suppose the number of APIs is m, the weight $w_{i,j}$ of edge $e_{i,j}$ in the weighting graph can be computed as follows:

$$w_{i,j} = \frac{n_{i,j}}{\sum_{k=1}^{m} n_{i,k}} \tag{1}$$

where $n_{i,j}$ means the number of calls that from API s_i to s_j, and $\sum_{k=1}^{m} n_{i,k}$ refers to the sum of all calls that are invoked from s_i. The nature of the weight in Markov Chain is the probability of the event's occurrence. Hence, the sum of the weight invoked from an API s_i must be 1:

$$\sum_{k=1}^{m} w_{i,k} = 1 \tag{2}$$

In the weighting phase, the final weight must fit both the sensitivity of invokes and the characteristics of the sample itself. Thus, the final graph will be produced from the merge of the adjacent matrix M (which generated from the sample), and the weighting graph M_w. The process of the merge is shown in Fig. 4. The merged graph will only retain the edges (invokes) existed both in M and M_w, and the final weight $W_{i,j}$ of the invoke from s_i to s_j can be calculated as follows:

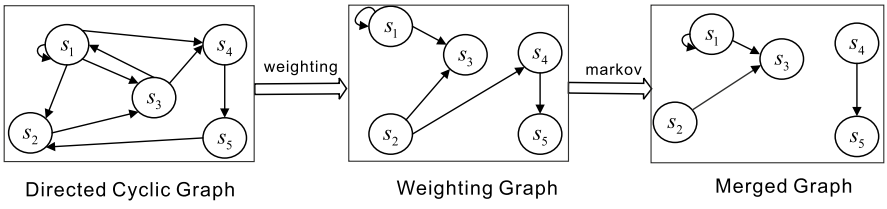


Fig. 4 Merge process of DCG and weighting graph

$$W_{i,j} = w_{i,j} \cdot n_{i,j} \tag{3}$$

where $n_{i,j}$ is the value of (i, j) in the adjacent matrix M, namely the number of invoke $e_{i,j}$ occurred in the sample. The final output for detection is a matrix presenting the merged graph, which is shown in Table 2.

3.3 GCN-based malware classification algorithm

Due to the structure of each generated graph is different, and the dimension of adjacency matrix is also different, using traditional neural network needs to unify the dimension, which will be a tedious work. Fortunately, graph convolution network can deal with graphs with any structure, so in this paper, we try to use graph convolution network to classify the generated graphs.

In the classifier module in the Fig. 1, a typical graph convolutional network structure architecture is shown. It has C input channels and F output characteristic graphs, and can contain multiple hidden layers [28]. Load a graph to GCN, and through several layers of GCN, the feature of each node changes from S to Z, and then completes the classification.

In our experiment, we design a two-layer semi-supervised classification GCN based on the weighted feature matrix R generated. In the forward phase, the calculation is as follows:

$$\hat{R} = \tilde{D}^{-\frac{1}{2}} \tilde{R} \tilde{D}^{-\frac{1}{2}} \tag{4}$$

Here, $\tilde{R} = R + I$, I is the identity matrix,$\tilde{D}$ is the degree matrix of $\tilde{R}$. Then, we use the forward model as follows:

$$Z = f(X, R) = \text{softmax}(\hat{R}\text{Relu}(\hat{R}XW^{(0)})W^{(1)}) \tag{5}$$

$W^{(0)}$ is the weight matrix from input layer to hidden layer, $W^{(0)}$ is the hidden-to-output weight matrix, the *Relu* function is defined as $\text{Relu}(x) = \max(0, x)$. In fact, since there are only two types, our softmax function is defined as follows:

$$\text{softmax}(p_i) = \frac{e^{p_i}}{e^{p_1} + e^{p_2}} \tag{6}$$

Here, $p_1 + p_2 = 1$. In the backpropagation stage, we use the cross-entropy loss function:

Table 2 Adjacent matrix of the merged graph

	S_1	S_2	S_3	S_4	S_5
S_1	$W_{1,1}$	–	$W_{1,3}$	–	–
S_2	–	–	$W_{2,3}$	–	–
S_3	–	–	–	–	–
S_4	–	–	–	–	$W_{4,5}$
S_5	–	–	–	–	–

$$\mathcal{L} = -\sum_{i=1}^{N} \left[y_1^{(i)} log(p_1^i) + y_2^{(i)} log(p_2^i) \right], \quad (i = 1, 2, \ldots, N) \tag{7}$$

Here, N is the number of samples, y is the label, p is the probability, $y_2^{(i)} = 1 - y_1^{(i)}$, $p_2^i = 1 - p_1^i$.

4 Dataset and experiment design

In this work, we collected a dataset consists of 13,624 samples, which have 6686 malware and 6938 benign samples. The detailed statistics are given in Table 3. The malicious samples came from VirusTotal and VirusShare, and the benign samples came from system programs as well as the Internet. The benign dataset was split into five parts evenly to fit the numbers of malware dataset for each year. In our experiments, the datasets were set 80% for training and 20% for evaluation. For weighting models, an API log dataset was adopted for training. The dataset for weighting included 62307 malware samples that were obtained from Virusshare and selected randomly so that the generality of the weighting model could be guaranteed.

The experiments were performed on a workstation with Ubuntu 18.04 system. To monitor and extract the call sequences of each sample, a cuckoo sandbox was deployed on the workstation as the running environment of the sample subsection weighting and generation of graph. In this phase, the extracted call sequences were numbered firstly. Then, the sequences were transformed into the DCG. The index of the DCG presented the corresponding API, and the value in each cell referred to the appearance number of the API invoked by the previous API. After generating DCG, the graph mixed with the weighting graph, so that a weighted DCG with a unique value for each edge was created.

In the weighting phase, firstly, the weighting graph was trained from the dataset. The initial weighting graph had 1609 rows and columns after the training. Then, the weighting graph was used to generate merged graphs for detection.

Table 3 Datasets for evaluation

Dataset	Number
2016 Malware Dataset	1606
2017 Malware Dataset	1247
2018 Malware Dataset	1656
2019 Malware Dataset	888
2020 Malware Dataset	1289
Benign File Dataset	6938
Total	13,624

5 Experiments and results analysis

5.1 Detection and evaluation methods

Before analyzing the results, several common detection and evaluation methods are introduced.

Accuracy is a standard metric that measures the exactitude of prediction. Precision refers to the number of predicted positive samples that are really positive. Recall is the number of positive examples in the sample that are predicted as positive. F1-Score is a comprehensive measure index of the classification model. All of these indexes could be computed as the following equations:

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

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

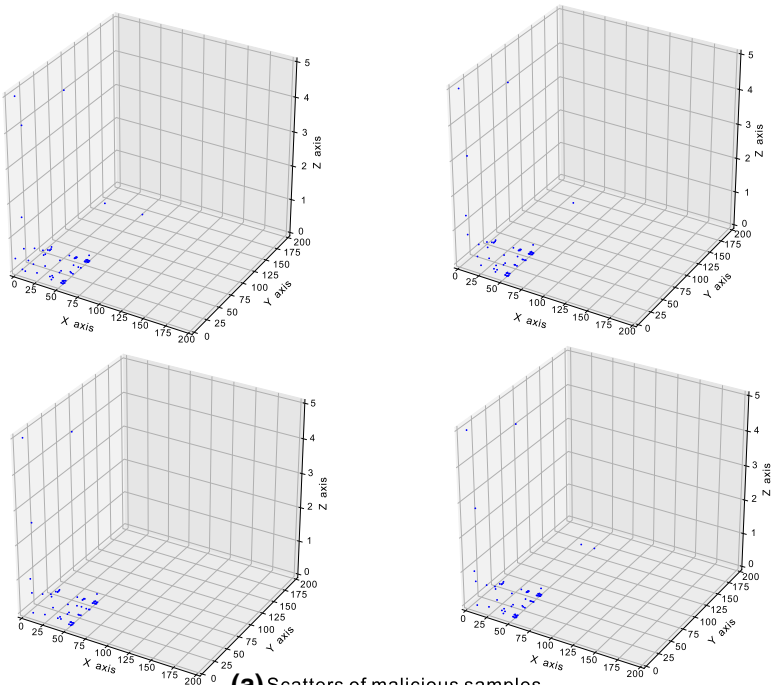
where TP is True Positive, which refers to the number of positive samples that are predicted as positive. FN is False Negative, namely the number of positive samples that are predicted as negative. FP is False Positive, which means the number of negative samples that are predicted as positive. TN is True Negative, which is the number of negative samples that are predicted as negative. In our work, malicious samples were labeled as positive, and benign samples were labeled as negative.

5.2 Classification results and analysis

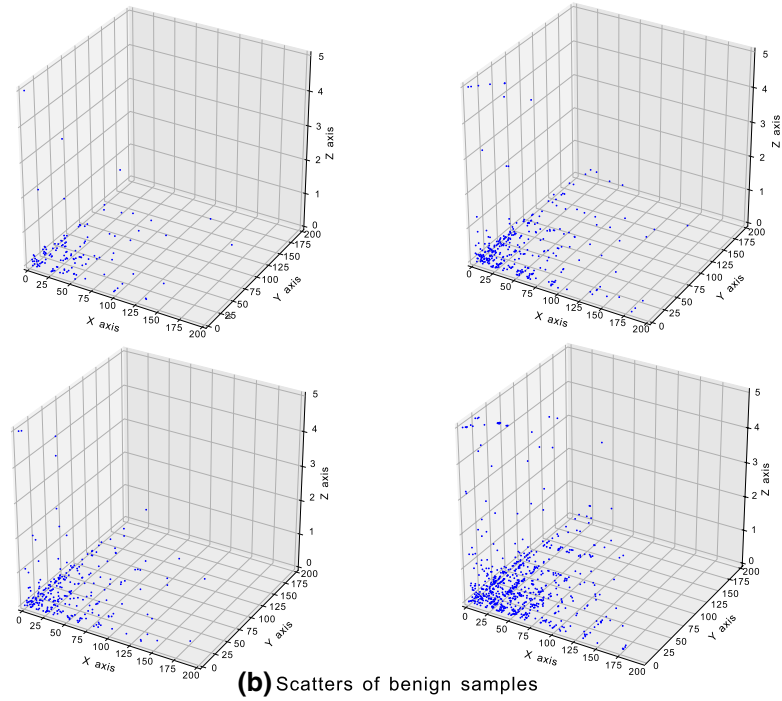
After we extracted the API call sequence diagram, we analyzed the sample API call sequence and corresponding weights.

Figure 5 shows scatters of four randomly selected groups of samples, where the X-axis referred to the serial number of API that invoke others, Y-axis referred to the serial number of API that was invoked by others, and Z-axis referred to the weight of the API invoke. From the figure, we could find that the point of benign samples distributed more dispersed than that of malicious samples. Also, the weights of API invokes in the benign samples were more varied than the malicious samples.

Table 4 shows the evaluation result of the models with datasets from different years. The results showed that the performance among the models had distinct differences. For the models based on machine learning, most of them performed well in the detection of most datasets. The overall performance of GCN in the evaluation was relatively good, most of the indicators were ahead of other models, and the



(a) Scatters of malicious samples



(b) Scatters of benign samples

Fig. 5 Scatters of final merged graph

Table 4 Evaluation of the models with different datasets

DataSet	Model	Accuracy	Precision	Recall	F1-Score
2016 Year Dataset	SVM	0.9556	0.9350	0.9573	0.9656
	Naïve Bayes	0.8211	0.8606	0.8087	0.8112
	Decision tree	0.9386	0.9406	**0.9652**	0.9328
	Random forest	0.9357	0.9183	0.9547	0.9351
	GCN	**0.9832**	**0.9867**	0.9619	**0.9734**
	CNN	0.9673	0.9518	0.9562	0.9529
	RNN	0.9270	0.9420	0.9339	0.9280
2017 Year Dataset	SVM	0.9605	0.9543	0.9573	**0.9652**
	Naïve Bayes	0.8327	0.8430	0.7834	0.8678
	Decision tree	0.9305	0.9618	0.9267	0.9591
	Random forest	0.9502	0.9496	0.9389	0.9478
	GCN	0.9602	0.9546	0.9636	**0.9622**
	CNN	**0.9627**	**0.9563**	**0.9682**	0.9457
	RNN	0.9605	0.9468	0.9722	0.9544
2018 Year Dataset	SVM	0.9256	0.9345	0.9487	0.9225
	Naïve Bayes	0.8688	0.8693	0.8828	0.8685
	Decision tree	0.9440	0.9256	0.9522	0.9337
	Random forest	0.9419	0.9617	0.9220	0.9518
	GCN	**0.9654**	**0.9770**	0.9562	**0.9765**
	CNN	0.9643	0.9638	**0.9688**	0.9632
	RNN	0.9570	0.9340	0.9521	0.9380
2019 Year Dataset	SVM	0.9458	0.9362	0.9276	0.9463
	Naïve Bayes	0.8378	0.8528	0.8519	0.8261
	Decision tree	0.9356	0.9251	0.9159	0.9454
	Random forest	0.9507	0.9487	0.9293	0.9186
	GCN	**0.9654**	0.9463	**0.9644**	**0.9552**
	CNN	0.9307	**0.9581**	0.9286	0.9483
	RNN	0.9480	0.9285	0.9133	0.9259
2020 Year Dataset	SVM	0.9008	0.8996	0.9037	0.9003
	Naïve Bayes	0.7927	0.819	0.7847	0.8038
	Decision tree	0.9245	0.9164	0.9319	0.9237
	Random forest	0.9226	0.9176	0.9447	0.9261
	GCN	**0.9467**	**0.9494**	**0.9321**	**0.9505**
	CNN	0.9361	0.9344	0.9259	0.9101
	RNN	0.9285	0.9379	0.9324	0.9351

Bold represents the optimum value in the corresponding year indicator

CNN model was relatively close, which was related to their classification principles. In addition, with the increase in the year, the prediction accuracy has declined to a certain extent. We believe that the anti-detection ability of the sample has improved to a certain extent. It is particularly important to note that in the 2020 data, the

Table 5 Comparison of the detection effect of GCN and existing methods

Detection approach	Type	TPR	FPR	Accuracy
Smita et al. [19]	Windows	0.9930	0.0460	0.9542
Tang et al. [20]	Windows	0.9900	0.0105	None
Li et al. [29]	Android	None	None	0.9647
Raff et al. [21]	Windows	None	None	0.9400
Alzaylaee et al. [22]	Android	0.9956	0.0330	0.9780
GCN	Windows	0.9951	0.0037	0.9832

detection indicators of GCN are higher than other models, which indicates that the GCN model has a better classification effect against models with strong detection capabilities.

The results show that in most of the models based on machine learning or deep learning, the method has a better performance on the detection of general malware, which proves the effectiveness of the method.

We compared other methods with our model to test the performance of the model. Table 5 introduces the comparison results of our model performance and other existing malicious code detection methods based on deep learning or machine learning. As shown in Table 5, the TPR of all methods exceeds 99%. In addition, the FPR of our method is lower than other methods, so our proposed method is superior to other methods in terms of FPR and accuracy. The reason for the better performance of our model is that as the malicious code data set increases, our extracted malicious features are more accurate and the model is more robust.

6 Conclusion

Malware has a long history, which seriously threatens the security of computer system. With the rapid development of anti-detection technology, the capability of traditional detection methods based on static analysis and dynamic analysis is limited. With neural network having strong prediction performance, the application of AI technology in malware detection has become a research hotspot. However, due to the difference of malware, feature extraction is difficult, which is not conducive to the application of traditional neural network. To solve the problem, we use the flexibility of GCN input to design a malware detector based on GCN to adapt to the differences of malware. The specific method is to extract the API call sequence from the malicious code and generate the directed cyclic graph, use Markov chain to extract the characteristics of the graph, and then use GCN to realize classification. We also have done evaluation comparing with other machine learning algorithms. The results show that the method has better performance in most detection, and the highest accuracy is 98.32%. From the research, we find that the technology has potential adaptability, but it has not been realized yet. In the future work, we will focus on the research of adaptive detection model based on GCN, so that the

malware detection system has stronger adaptive ability, so as to reduce the cost of personnel of malware detection.

Acknowledgements This work was partially supported by National Key R&D Program of China under Grant No. 2020YFC0832500, Ministry of Education - China Mobile Research Foundation under Grant No. MCM20170206, The Fundamental Research Funds for the Central Universities under Grant No. lzujbky-2019-kb51 and lzujbky-2018-k12, National Natural Science Foundation of China under Grant No. 61402210, Major National Project of High Resolution Earth Observation System under Grant No. 30-Y20A34-9010-15/17, Program for New Century Excellent Talents in University under Grant No. NCET-12-0250, State Grid Corporation of China Science and Technology Project under Grant No. SGGSKY00WYJS2000062, Strategic Priority Research Program of the Chinese Academy of Sciences with Grant No. XDA03030100, Google Research Awards and Google Faculty Award, Science and Technology Plan of Qinghai Province under Grant No.2020-GX-164. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Jetson TX1 used for this research.

References

1. Saeed IA, Selamat A, Abuagoub AMA (2013) A survey on malware and malware detection systems. Int J Comput Appl 67:25–31. https://doi.org/10.5120/11480-7108
2. Bazrafshan Z, Hashemi H, Fard SMH, Hamzeh A (2013) A survey on heuristic malware detection techniques. In: The 5th Conference on Information and Knowledge Technology, IEEE, shiraz, Iran, pp 113–120. https://doi.org/10.1109/IKT.2013.6620049
3. Ye Y, Li T, Adjeroh D, Iyengar SS (2017) A survey on malware detection using data mining techniques. ACM Comput Surv 50:1–40. https://doi.org/10.1145/3073559
4. Von Neumann J (1966) Theory and organization of complicated automata
5. Cohen F (1984) Computer viruses: theory and experiments. Comput Secur 6:22–35
6. Christodorescu M, Jha S, Seshia SA, et al (2005) Semantics-aware malware detection. In: 2005 IEEE Symposium on Security and Privacy (S & P'05), IEEE, pp 32–46
7. Vasudevan A, Yerraballi R (2006) Spike: engineering malware analysis tools using unobtrusive binary-instrumentation. In: Proceedings of the 29th Australasian Computer Science Conference—Vol 48. Australian Computer Society, Inc., AUS, p 311320
8. Sundarkumar GG, Ravi V, Nwogu I et al (2015) Malware detection via API calls, topic models and machine learning
9. Wu S, Wang P, Li X et al (2016) Effective detection of android malware based on the usage of data flow APIs and machine learning. Inf Softw Technol 75:17–25
10. Deng X, Liu Y, Zhu C, Zhang H (2021) Air-ground surveillance sensor network based on edge computing for target tracking. Comput Commun 166:254–261
11. Schultz MG, Eskin E, Zadok F, et al (2002) Data mining methods for detection of new malicious executables. In: IEEE Symposium on Security & Privacy
12. Moskovitch R, Stopel D, Feher C et al (2009) Unknown malcode detection and the imbalance problem. J Comput Virol 5:295–308
13. Moskovitch R, Feher C, Tzachar N, et al (2008) Unknown malcode detection using opcode representation. In: Intelligence & Security Informatics, First European Conference, Euroisi, Esbjerg, Denmark, December
14. Santos I, Nieves J, Bringas PG (2011) Semi-supervised learning for unknown malware detection. In: International symposium on distributed computing & artificial intelligence. PCA techniques, 2016 international conference on advanced communication systems and information security (ACOSIS). IEEE, Marrakesh, Morocco, pp 1–7. https://doi.org/10.1109/ACOSIS.2016.7843930
15. Firdausi I, Lim C, Erwin A, et al (2010) Analysis of machine learning techniques used in behavior-based malware detection. IEEE Computer Society
16. Rieck K, Trinius P, Willems C et al (2011) Automatic analysis of malware behavior using machine learning. J Comput Secur 19:639–668
17. Anderson HS, Roth P (2018) Ember: an open dataset for training static pe malware machine learning models

18. Sharma S, Krishna CR, Sahay SK (2019) Detection of advanced malware by machine learning techniques. Cryptogr Secur 333–342
19. Naval S, Laxmi V, Rajarajan M et al (2015) Employing program semantics for malware dectection. IEEE Trans Inf For Secur 10(12):2591–2604
20. Tang M, Qian Q (2018) Dynamic api call sequence visualisation for malware classification. IET Inf Secur 13(4):367–377
21. Raff E, Barker J, Sylvester J et al (2017) Malware detection by eating a whole exe. Machine Learning
22. Alzaylaee MK, Yerima SY, Sezer S (2020) DL-Droid: deep learning based android malware detection using real devices. Comput Secur 89:101663
23. Xu K, Hu W, Leskovec J, Jegelka S (2018) How powerful are graph neural networks? arXiv:1810.00826
24. Spirtes PL (2013) Directed Cyclic Graphical Representations of Feedback Models. arXiv:1302.4982
25. Deng X, Li J, Liu E, Zhang H (2020) Task allocation algorithm and optimization model on edge collaboration. J Syst Arch 110:101778
26. Xiao X, Wang Z, Li Q, Xia S, Jiang Y (2017) Back-propagation neural network on Markov chains from system call sequences: a new approach for detecting Android malware with system call sequences. IET Inf Secur 11:8–15. https://doi.org/10.1049/iet-ifs.2015.0211
27. Zang D, Liu J,Wang H (2018) Markov chain-based feature extraction for anomaly detection in time series and its industrial application. In: 2018 Chinese Control And Decision Conference (CCDC), IEEE
28. Kipf TN , Welling M (2016) Semi-supervised classification with graph convolutional networks
29. Li J, Sun L, Yan Q et al (2018) Significant permission identification for machine-learning based andriod malware detection. IEEE Trans Ind Inform 14(7):3216–3225