Malware Detection with Directed Cyclic Graph and Weight Merging

Shanxi Li[1], Qingguo Zhou[1]* and Wei Wei[2]*
[1] School of Information Science and Engineering, Lanzhou University
Lanzhou 730000 - China
[e-mail: lisx@lzu.edu.cn]
[2] School of Computer Science and Engineering, Xi'an University of Technology
Xi'an 710048 - China
[e-mail: weiwei@xaut.edu.cn]
*Corresponding author: Qingguo Zhou and Wei Wei

Received August 18, 2020; revised March 30, 2021; accepted June 14, 2021; published September 30, 2021

Abstract

Malware is a severe threat to the computing system and there's a long history of the battle between malware detection and anti-detection. Most traditional detection methods are based on static analysis with signature matching and dynamic analysis methods that are focused on sensitive behaviors. However, the usual detections have only limited effect when meeting the development of malware, so that the manual update for feature sets is essential. Besides, most of these methods match target samples with the usual feature database, which ignored the characteristics of the sample itself. In this paper, we propose a new malware detection method that could combine the features of a single sample and the general features of malware. Firstly, a structure of Directed Cyclic Graph (DCG) is adopted to extract features from samples. Then the sensitivity of each API call is computed with Markov Chain. Afterward, the graph is merged with the chain to get the final features. Finally, the detectors based on machine learning or deep learning are devised for identification. To evaluate the effect and robustness of our approach, several experiments were adopted. The results showed that the proposed method had a good performance in most tests, and the approach also had stability with the development and growth of malware.

Keywords: malware detection, directed cyclic graph, Markov Chain, machine learning, neural network

http://doi.org/10.3837/tiis.2021.09.010 **ISSN : 1976-7277**

1. Introduction

The concepts of malware could include all binary files that would injure the computing system intentionally. In recent years, malware have more diversities in techniques and destination. Nowadays, many malware adopt the way that mix with multiple attack methods to reach the target, and most of them take obfuscation or encryption method to disguise themselves and avoid detection.

Contract to the development of malware techniques, 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 meeting 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, some researchers have focused on the dynamic detection of malware based on Application Programming Interface (API) call sequences, and have proved their effects [4], [5], [6]. API is a set of instructions used to program software applications. All programs can interact with the Windows API and access pre-defined functions by invoking API calls to make use of facilities provided involving base services to access resources such as system files, processes, threads, devices, and advanced services [7]. The API calls of malware reflect its high-level functionality and can be used to understand its overall behavior. API call sequence analysis is, therefore, an effective method of malware analysis [8].

In this work, we propose a novel approach for feature extraction and detection. The approach extract features from the specific sample and combines the feature with the general characteristics of malware. Besides, we adopt Principal Components Analysis (PCA) for reducing dimensions and consumption of resources. Moreover, some models based on machine or deep learning are devised as detectors to identify the features. The main contributions of our work are listed as follows:

- We propose a novel method for feature extraction from samples with a solid structure. The structure is based on the invoke relationship between API and could maintain most information on the API call sequences.

- To get general features of malware, we design a new weighting model based on Markov Chain, the model can be trained with numerous known malware to decide the weight of each invokes caused in malware. Then the features extracted from a specific sample will be combined with general features of whole malware so that the newly generated features can maintain the characteristics of the sample itself while increasing the generality of the features to defend the disguises and variants of malware.

- We adopt PCA to reduce consumption and improve the performance of the approach. Afterward, we use several models based on machine learning and devise a few other models based on neural networks for detection. The purpose of using various models is to prove the universality of our approach and compare the performance among different models.

- We evaluate the proposed method with a series of experiments and diverse datasets. The results present that our approaches are effective with most detectors and have robustness when meeting the development. However, some problems are also found in the experiments, which need further study.

The remainder of the paper is organized as follows. Some related works are reviewed in Section 2. Our method of feature extraction and detection are presented in Section 3 and 4. The details of experiments and results are shown in Section 5, and the whole work is concluded in Section 6.

2. Related Work

In recent years, static detections of malware have faced severe challenges from the anti-detection technique evolved subsequently, especially encryption and disguise. Therefore, more researchers have concentrated on dynamic detection techniques. One most efficient of them is API call sequences analysis.

Generally, the researches about API-based malware detection are mainly focused on the method of feature extraction such as n-grams, features, and flow graphs, and some researchers combined two of the above of the analysis to propose new approaches. Most of them got good results. Naval et al. [9] proposed a new strategy adopted both graphs and n-grams to generate a unique model called Ordered System-Call Graph (OSCG). Then the graph was transformed into a Feature Vector Table (FVT) using semantically relevant path extraction and was processed with an ensemble-based algorithm. The results showed the accuracy of detection could get up to 95%.

Detections based on Markov chains have also been adopted and developed by some researchers. Onwuzurike et al. [10] proposed a new method named MaMaDroid based on static analysis of API and Markov chains of the call graph to detect malicious Android apps. Ficco [11] improved the approach by exploiting dynamic analysis system calls instead of static. The developed approach was applied in IoT malware detection and get an F-measure up to 89%.

With the development of deep learning, some researchers also attempted to apply the technique of neural networks to malware identification. For example, Ganesh et al. [12] extracted features from APK samples and created a 12x12 vector image, then the image was detected by CNN models. The results showed an accuracy of 93%. However, detection approaches based on deep learning methods also are confronted with risks, specifically attack to deep learning models. Chen et al. [13] proposed an adversarial way to attack deep learning detectors based on Jacobian-based Saliency Map Attack (JSMA) and optimal perturbations onto Android APK. Then the approach was evaluated with MaMaDroid and Derbin. The results showed that the attacking method could attack of MaMaDroid and Derbin effectively. The research verified the seriousness of adversarial attacks towards the detection methods based on deep learning, which should be valued in all related researches.

3. Data Processing

In our work, a new detection method is proposed that consists of feature extraction, weighting, data compression, and detectors based on machine learning and deep learning. In this section, the implementation of data processing will be presented, and the details of the detection method would be put in the next section.

3.1 Weighted Directed Cycle Graph

Directed Acyclic Graph (DAG) is a continuous graph with a finite set of points S and a set of directed edges E, and any edge e from vertex s_i to another vertex s_j $(s_i, s_j \in S)$, and any vertex cannot return itself and the directed edge sequence [14]. 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.

DAG has been widely used in many fields. However, some graph nodes may access themselves. In this case, DAG is not practical and needs to be represented by Directed Cyclic Graph (DCG).

In the calling of software API, an API may call other APIs many times, or it may call itself. Therefore, we use DCG diagram to describe the calling relationship of software API. In the DCG graph of API, we define that the edges of DCG graph will be weighted according to the calling relationship of API. In other words, the weight of the directed edge between S_i and S_j is the number of times API S_i calls API S_j. In particular, the weight of the directed edge between S_i and S_i is also the structure of S_i calling S_i. As shown in **Fig. 1**, it describes the calling relationship between vertex (API) and vertex.

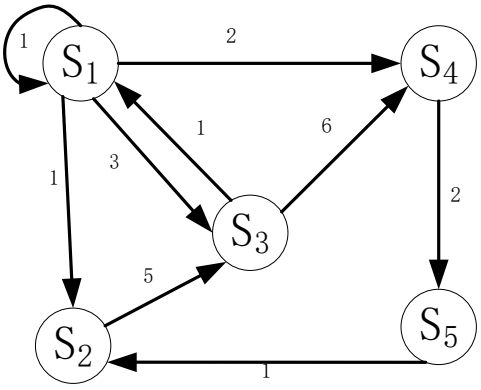


Fig. 1. Sturcture of DCG

We use adjacency matrix to describe a DCG graph called by API. Both rows and columns are used to represent API. The value of *i* row and *j* column in the matrix represents the number of times API S_i calls API S_j, which represents the weight of DCG edges. The adjacency matrix of DCG graph shown in Fig. 1 can be represented by **Table 1**, the weight of the edge in the graph corresponds to the value of the matrix one by one.

Table 1. Adjacent matrix of the DCG in Fig. 1

	S_1	S_2	S_3	S_4	S_5
S_1	1	2	3	2	0
S_2	0	0	5	0	0
S_3	1	0	0	6	0
S_4	0	0	0	0	2
S_5	0	1	0	0	0

3.2 Weighting from Markov Chain

Markov chain is a set of discrete random variables with Markov property[15]. Markov chain is widely used in the classification and processing of sequential data, especially in the dynamic detection of malicious code [16], [17].

In our method, in order to identify and quantify the sensitivity of API calls, we calculate the weight of each call from one API to another based on Markov chain. Therefore, we assume that the API call is independent of the previous process and do not consider the sequence of API calls when calculating the weight. Therefore, when we consider the API calling process of the sample, we only consider the number of each API calling each other, not its sequence. In this way, the API calls related to time series can be represented by DCG diagram, and then the weight can be calculated. The specific calculation process is shown in **Algorithm 1**.

Algorithm 1 weight calculation algorithm based on Markov chain

Input: malicious sample data set s = S1, S2, ⋯, Sn;
Output: The weight matrix w of malicious samples;

$W \leftarrow 0$
for each $i \in \{1, 2, \ldots, n\}$ **do**
    Obtain a sample S_i
    Obtain weight matrix w_i of sample S_i
    Obtain the API call sequence matrix M_i of sample $S_i = [E_1, E_2, \cdots, E_m]^T$, where $E_1 = (e_{11}, e_{12}, \bullet\bullet\bullet, e_{1m})$, ⋯, $E_m = (e_{m1}, e_{m2}, \cdots, e_{mm})$
    for $j \in \{1, 2, \cdots, m\}$ **do**
        $S \leftarrow \sum_{k=1}^{m} e_{jk}$
        for $k \in \{1, 2, \cdots, m\}$ **do**
            $w_{jk} \leftarrow \frac{e_{jk}}{S}$
        end for
    end for
    $W \leftarrow W + w_i$
end for
for $j \in \{1, 2, \ldots, m\}$ **do**
    $S \leftarrow \sum_{k=1}^{m} e_{jk}$
    for $k \in \{1, 2, \ldots, m\}$ **do**
        $W_{jk} \leftarrow \frac{w_{jk}}{S}$
    end for
end for
return W

When we calculate the weight, firstly, we need a Data Set which only contains malicious code and is rich enough to cover the general characteristics of malicious code. Then, all APIs in the Data Set are numbered (S_1 to S_M), and the corresponding DCG graph and adjacency matrix of samples are generated based on Markov chain. We define the DCG graph based on

Markov chain as "weighted graph", denote it as matrix *M*, and generate its weight graph. In the weight graph, the weight W_{ij} of edge e_{ij} is calculated as follows:

$$w_{i,j} = \frac{N(e_{i,j})}{\sum_{k=1}^{n} N(e_{i,k})} \tag{1}$$

where $N(e_{i,j})$ means the number of calls that from API s_i to s_j, and $\sum_{k=1}^{n} N(e_{i,k})$ refers to the sum of all calls that 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:

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

In the weighting stage, the weight of each sample is calculated first. After the weight of all samples is calculated, the weight matrix of each sample is accumulated, and the final weight graph is normalized by using the above method again to obtain the adjacency matrix of the final weight graph. The adjacency matrix can express the importance of API call and the characteristics of samples.

3.3 Feature extraction of detection samples

After getting the weight graph of malicious code Data Set, we need to use it to extract the feature graph of detection samples. The specific process is as follows:

Firstly, the DCG graph of the detection sample is generated.

Secondly, the DCG graph and the weight graph are used for point multiplication to obtain the feature graph of the detection sample.

The process of feature map extraction is shown in **Fig. 2**. From the above two steps, the DCG graph of detection samples and the weight graph of malicious samples can be obtained respectively. Both of them are represented by adjacency matrix. It is not difficult to find that the dimensions of the two matrices are the same, which are $m \times m$ dimension matrices. In feature extraction, we multiply the two matrices, that is, multiply the corresponding elements, and get the feature graph of the detection sample, which is also $m \times m$ dimension matrix. Similarly, the elements of the adjacency matrix of the feature graph are normalized, and the processing formula is as follows:

$$W_{ij} = w_{ij} \bullet n_{ij} \tag{3}$$

Where n_{ij} is the value of the *i* row *j* column in the adjacency matrix *M* of the sample to be detected, that is, the number of times e_i is called. w_{ij} is the value of *ith* row *jth* column in the weight matrix *W*. In the figure, the calculation principle is represented by a simple example, and the feature graph extracted from the figure is represented as an adjacency matrix, as shown in **Table 2**.

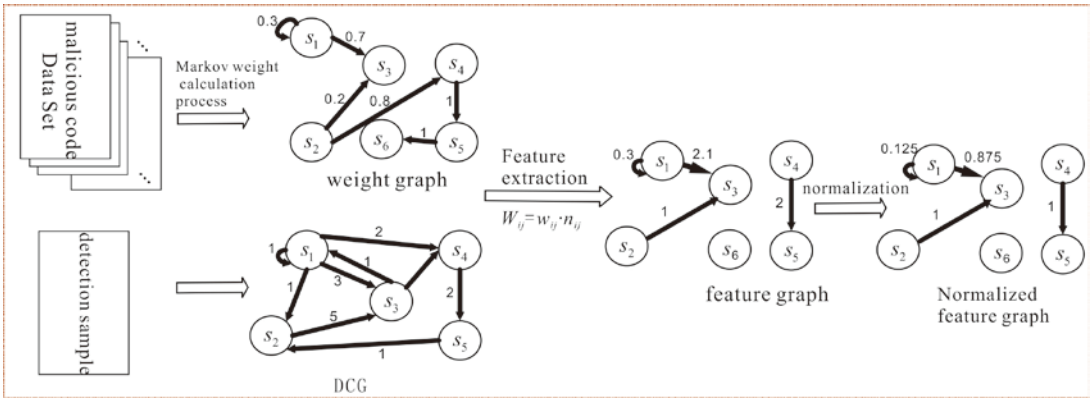


Fig. 2. Merge process of DCG and weighting graph

Table 2. Adjacent matrix of the Merged graph

	S_1	S_2	S_3	S_4	S_5	S_5
S_1	0.125	0	0.875	0	0	0
S_2	0	0	1	0	0	0
S_3	0	0	0	0	0	0
S_4	0	0	0	0	1	0
S_5	0	0	0	0	0	0

3.3 Principal Components Analysis

To improve the performance of the evaluation and reduce time consumption, PCA technique is adopted to centralize the information and reduce the dataset volume.

PCA is a prevalent statistic processing technique for data analysis and preprocessing, which has been applied widely in data processing and mining [18], [19], [20]. The main propose of PCA is to reduce the dimension of analyzed data while maintaining the most information [21]. Generally, PCA can transform an m-dimension dependent variable into an n-dimension independent one ($m < n$) under the premise of keeping most of the information, and the transformed variables are called the principal components (PCS).

Considering a data set X with the number of data M and the dimension N, so that $X = [x_1, x_2, ..., x_i, ..., x_N]^T$ and $x_i = [v_{i,1}, v_{i,2}, ..., v_{i,N}]$. The PCA process could be presented as following steps:

- Before calculation, each variable $v_{i,j}$, $j = 1, 2, ..., N$ of the vector x_i was rescaled as

$$\mu_i = \frac{1}{M}\sum_{j=1}^{m} v_{i,j} \tag{4}$$

$$\theta_{i,j} = v_{i,j} - \mu_i \tag{5}$$

  each variable $v_{i,j}$ would be replaced with $\theta_{i,j}$, and the vector x_i finish the rescalation.

- After the rescalation, the covariance matrix is further calculated and eigenvalue decomposition is performed as:

$$R = X^T \cdot X = P \cdot \Lambda \cdot P^T \tag{6}$$

where P is the matrix formed by the eigenvector, Λ is the eigenvalue matrix and is arranged as follows by size:

$$R = (1-N-1) X^T \cdot X = P \cdot \Lambda \cdot P^T \tag{7}$$

where $PP^T = P^T P = I_M$.

$$R = [P_q P_{M-q}] \begin{bmatrix} \Lambda_q & 0 \\ 0 & \Lambda_{M-q} \end{bmatrix} \begin{bmatrix} P_q^T \\ P_{M-q}^T \end{bmatrix} \tag{8}$$

where Λ is the eigenvalues matrix of R with decreasing order.

4. Detection Method

In this section, several detection methods based on machine learning and deep learning are proposed.

4.1 Machine Learning Approaches

For the detectors that based on machine learning, Support Vector Machine (SVM), Decision Tree (DT), Random Forests (RF), and Naive Bayes (NB) are adopted to evaluate our approaches. All those models are supervised learning models, which need to be trained with lots of labeled data to obtain a model for classification. Notably, the principals of each model are very different, so that they will have distinct results in malware detection.

Support Vector Machine is a valid binary classifier that is applied widely in the task of two-class identification [22]. The target of SVM is to get a hyperplane that could identify a binary class with maximum margin from support vectors so that the hyperplane could work well in classify new input data.

The principal of Decision Tree is to build a tree that consists of a non-leaf node for representing attribute and leaf-node for labeling [23]. The method can learn the characteristics from the training data by using higher information gain, and the attributes will be used for classification.

Random Forest is a combination of several self-determining decision tree [24]. Each tree in the method will process independently, and all results of the trees will be collected and finally voted for the ultimate result. Because the outcome of Random Forest is based on various trees, it usually performs better than a single Decision Tree.

Naive Bayes is a classification method based on Bayes theorem, which assumes that the features of the data are independent among them. The model calculates the conditional probability of each feature in the training phase, and the possibility will be adopted for computing posterior probability and then classification.

4.2 Deep Learning Approaches

Besides the models of machine learning, some models based on CNN and RNN are also used to detect malware. Each model is customized to fit the detection that effective for our method.

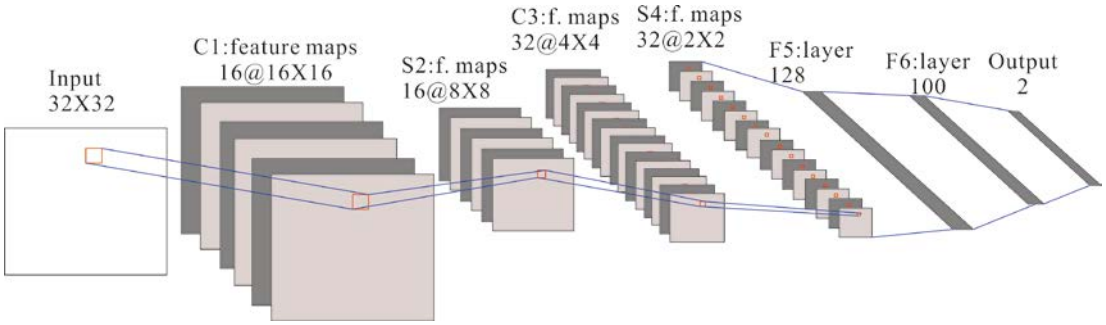


Fig. 3. Sturcture of CNN

The structure of CNN is shown in **Fig. 3**. The input layer of the model is a matrix generated from PCA. The output size of each layer is marked above it. C1 is convolution layer, which contains 16 convolution cores with a size of 5 × 5 and a step size of 1; C3 is convolution layer, which contains 32 convolution cores with a size of 5 × 5 and a step size of 1; S2 and S4 are pooling layers with the same size of 2 × 2; the number of neurons in the full connection layer is 128 and 100; the final output layer is softmax, the output is a two-dimensional variable for classification.

The architecture of RNN is shown in **Fig. 4**. RNN is a sequence-based model that output a state h_t at each step t, and each state of current step will depend on both current input x_t and previous state h_{t-1} [26], [27].

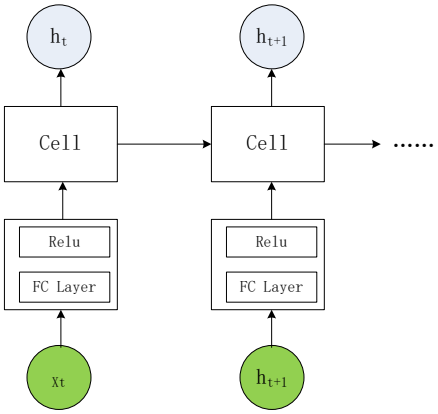


Fig. 4. Sturcture of RNN

In this model, we combine a full connection layer, a Relu unit and an RNN cell as a network structure at one time. The full connection layer and Relu unit are used to extract sample features, RNN is used to identify the attributes of data.

5. Experimental Classification Results and Analysis

5.1 Datasets and Environment

Dataset: In this work, we collected a dataset consists of 13624 samples, which have 6686 malware and 6938 benign samples. The detailed statistics were presented 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, 10-fold cross-validation method is used to train and verify the model. The method is as follows: first, all samples are randomly divided into 10 subsets, each time 9 subsets are selected as training samples, and another subset is taken as test samples. In this way, the model can be trained ten times, and the remaining validation sample can be used to evaluate the accuracy of the model after training. Finally, average the ten test results as the final test results. 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.

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	13624

Environments: 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 samples.

5.2 Experiment Steps

A. Weighting and Generating 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.

B. Detection and Evaluation

After data processing, the final data was detected with the proposed models. To evaluate the performance of each model, the index of accuracy, precision, recall, and F1-score were adopted.

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:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

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.

Besides, the receiver operating characteristic (ROC) curve was also be implemented as a measure to evaluate the models. The curve used False Positive Rate (FPR) as the X-axis and True Positive Rate (TPR) as the Y-axis. Both values could be computed as (14) and (15). Moreover, the Area Under Curve (AUC) was calculated to estimate the overall accuracy of the models [28].

$$TPR = \frac{TP}{TP + FN} \tag{14}$$

$$FPR = \frac{FP}{FP + TN} \tag{15}$$

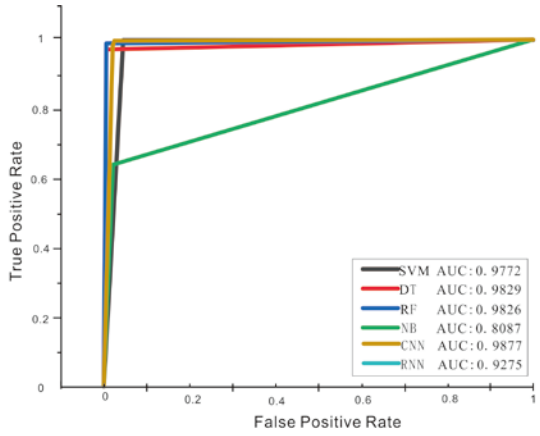
5.3 Detection Result and Analysis

Table 4 shows the evaluation results of the model based on the datasets of different years. The results show that both machine learning model and deep learning model have achieved good prediction results, and the performance of each model is relatively close, which does not show that the model is the best. It shows that our feature extraction method is effective, and some main machine learning models based on this feature extraction method have good detection results, which shows that our proposed feature extraction method is universal.

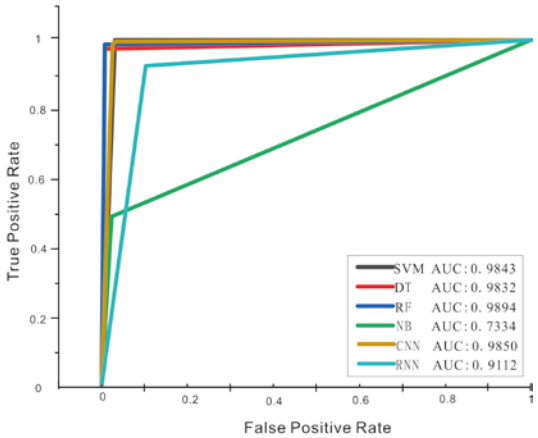
It can be seen that with the increase of years, the prediction accuracy of the model shows a downward trend. We think that this is because with the increase of years, the design level of malicious code is getting higher and higher, and it has more powerful anti detection ability, which is a big challenge to our model. In the future research, we will try to improve the self optimization ability of the model to solve the problem.

Fig. 5 showed the ROC curves of the evaluation, where DT, RF, and NB are the abbreviation of Decision Tree, Random Forest, and Naive Bayes. The AUC of each model was presented in the legend of the figure. It could be seen that most models had a good effect in evaluation, except Naive Bayes that had a poor performance and RNN that performed very unstably in the detection.

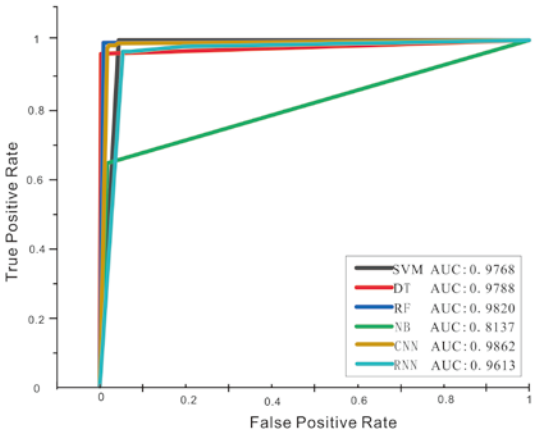
In summary, the result indicated that our approach had a good effect on the general malware detection with the most models based on machine or deep learning, which could prove the effectiveness of our approach.



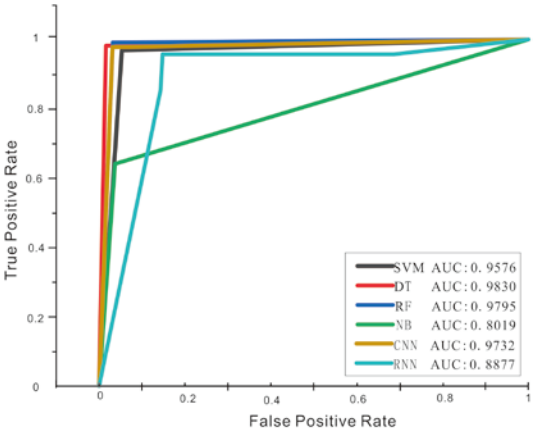
(a) Test results of data sets in 2016



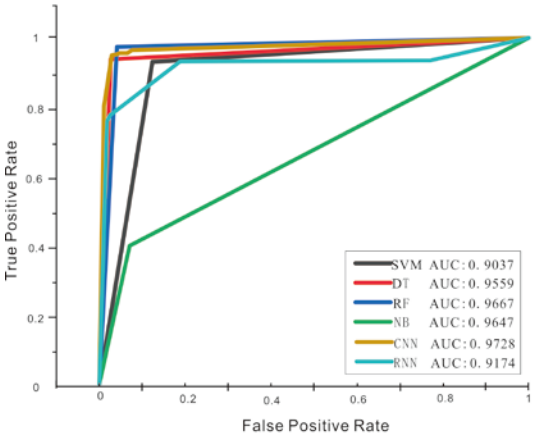
(b) Test results of data sets in 2017



(c) Test results of data sets in 2018



(d) Test results of data sets in 2019



(e) Test results of data sets in 2020

Model	AUC				
	2016	2017	2018	2019	2020
SVM	0. 9772	0. 9843	0. 9768	0. 9576	0. 9037
DT	0. 9829	0. 9832	0. 9788	0. 9830	0. 9559
RF	0. 9826	0. 9894	0. 9820	0. 9795	0. 9667
NB	0. 8087	0. 7334	0. 8137	0. 8019	0. 9647
CNN	0. 9877	0. 9850	0. 9862	0. 9732	0. 9728
RNN	0.9275	0. 9112	0. 9613	0. 8877	0. 9174

(f)AUC statistics of several models

Fig. 5. ROC Curve of evaluation

Table 4. Evaluation of the models with different datasets

DataSet	Model	Accuracy	Precision	Recall	F1-Score
2016 Year Dataset	SVM	0.9556	0.935	0.9573	0.9656
	NB	0.8211	0.8606	0.8087	0.8112
	DT	0.9386	0.9406	**0.9652**	0.9328
	RF	0.9357	0.9183	0.9547	0.9351
	DNN	0.9618	**0.9536**	0.9627	**0.9648**
	CNN	**0.9673**	0.9518	0.9562	0.9529
	RNN	0.927	0.942	0.9339	0.928
2017 Year Dataset	SVM	0.9605	0.9543	0.9573	**0.9652**
	NB	0.8327	0.843	0.7834	0.8678
	DT	0.9305	**0.9618**	0.9267	0.9591
	RF	0.9502	0.9496	0.9389	0.9478
	DNN	0.9506	0.9544	0.9473	0.9625
	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
	NB	0.8688	0.8693	0.8828	0.8685
	DT	0.944	0.9256	0.9522	0.9337
	RF	0.9419	0.9617	0.922	0.9518
	DNN	0.9353	0.9270	0.9445	**0.9656**
	CNN	**0.9643**	**0.9638**	**0.9688**	0.9632
	RNN	0.957	0.934	0.9521	0.938
2019 Year Dataset	SVM	0.9458	0.9362	0.9276	0.9463
	NB	0.8378	0.8528	0.8519	0.8261
	DT	0.9356	0.9251	0.9159	0.9454
	RF	**0.9507**	0.9487	0.9293	0.9186
	DNN	0.9434	0.9267	0.9141	0.9253
	CNN	0.9307	**0.9581**	**0.9286**	**0.9483**
	RNN	0.948	0.9285	0.9133	0.9259
2020 Year Dataset	SVM	0.9008	0.8996	0.9037	0.9003
	NB	0.7927	0.819	0.7847	0.8038
	DT	0.9245	0.9164	0.9319	0.9237
	RF	0.9226	0.9176	**0.9447**	0.9261
	DNN	0.9162	0.9095	0.9228	**0.9409**
	CNN	**0.9361**	0.9344	0.9259	0.9101
	RNN	0.9285	**0.9379**	0.9324	0.9351

6. Discussion and Conclusion

In this work, we propose a new malware detection approach. The approach mainly consists of the generation of Directed Cyclic Graph and the weighting based on the Markov Chain. To improve the performance and accuracy of the method, Primal Component Analysis is applied to reduce the volume of data as well as retain the primary information related to the detection. Finally, some models based on machine learning and deep learning are adopted to evaluate the effect of our approach. The result shows that the method had an excellent performance in most of the detection, with the highest accuracy of 96.73%. The AUC of most models could keep above 90% except NB, which could confirm the robustness and universality of the approach.

Inevitably, our method also has some problems and limitations, which need to be further studied. First of all, we can see that our method needs PCA method to reduce the dimension, so as to meet the requirements of fixed input of the model, and it is not adaptive when facing the malicious code with large differences. In the future work, our research will focus on the adaptive learning of features and models, so that the features of malware detection can adapt to

the development of malware and related technologies, without human interference. In addition, in view of the increasing trend of adversarial attacks based on machine learning model, we will further study the related defense measures.

Acknowledgment

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-2020-sp02, lzujbky-2019-kb51 and lzujbky-2018-k12, National Natural Science Foundation of China under Grant No. 61402210, State Grid Corporation of China Science and Technology Project under Grant No. SGGSKY00WYJS2000062, Science, National key R&D Program of China under Grant NO. 2018YFB0203901, the Key Research and Development Program of Shaanxi Province(No.2018ZDXM-GY-036), Shaanxi Key Laboratory of Intelligent Processing for Big Energy Data(No.IPBED7) , Technology Plan of Qinghai Province under Grant No.2020-GX-164, Google Research Awards and Google Faculty Award. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Jetson TX1 used for this research.

References

[1] A.Saeed, Imtithal, Ali Selamat, and Ali M. A. Abuagoub, "A Survey on Malware and Malware Detection Systems," *International Journal of Computer Applications*, 67(16), 25–31, 2013. Article (CrossRef Link).

[2] Bazrafshan, Zahra, Hashem Hashemi, Seyed Mehdi Hazrati Fard, and Ali Hamzeh, "A Survey on Heuristic Malware Detection Techniques," in *Proc. of The 5th Conference on Information and Knowledge Technology*, shiraz, Iran, 113–120, 2013. Article (CrossRef Link)

[3] Ye, Yanfang, Tao Li, Donald Adjeroh, and S. Sitharama Iyengar. 2017, "A Survey on Malware Detection Using Data Mining Techniques," *ACM Computing Surveys*, 50(3), 1–40, 2017. Article (CrossRef Link).

[4] Tang, Mingdong, and Quan Qian, "Dynamic API Call Sequence Visualisation for Malware Classification," *IET Information Security*, 13(4), 367–377, 2019. Article (CrossRef Link).

[5] Xiaofeng, Lu, Jiang Fangshuo, Zhou Xiao, Yi Shengwei, Sha Jing, and Pietro Lio, "ASSCA: API Sequence and Statistics Features Combined Architecture for Malware Detection," *Computer Networks*, 157, 99–111, 2019. Article (CrossRef Link)

[6] Ma, Xin, Shize Guo, Wei Bai, Jun Chen, Shiming Xia, and Zhisong Pan, "An API Semantics-Aware Malware Detection Method Based on Deep Learning," *Security and Communication Networks*, 2019. Article (CrossRef Link)

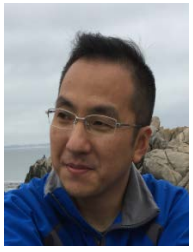
[7] Alqurashi, Saja, Omar Batarfi, Saudi Arabi, "A comparison between API call sequences and opcode sequences as reflectors of malware behavior," in *Proc. of 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE*, 2017. Article (CrossRef Link)

[8] Mira, Fahad, "A Review Paper of Malware Detection Using API Call Sequences," in *Proc. of 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, Riyadh, Saudi Arabia, 1–6, 2019. Article (CrossRef Link).

[9] Naval, Smita, Vijay Laxmi, Muttukrishnan Rajarajan, Manoj Singh Gaur, and Mauro Conti, "Employing Program Semantics for Malware Detection," *IEEE Transactions on Information Forensics and Security*, 10(12), 2591–2604, 2015. Article (CrossRef Link).

[10] Onwuzurike, Lucky, Enrico Mariconti, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon Ross, and Gianluca Stringhini, "MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models (Extended Version)," *ACM Transactions on Privacy and Security*, 22(2), 1–34, 2019. Article (CrossRef Link).
[11] Ficco, Massimo, "Detecting IoT Malware by Markov Chain Behavioral Models," in *Proc. of 2019 IEEE International Conference on Cloud Engineering (IC2E)*, Prague, Czech Republic, 229–234, 2019. Article (CrossRef Link).
[12] Ganesh, Meenu, Priyanka Pednekar, Pooja Prabhuswamy, Divyashri Sreedharan Nair, Younghee Park, and Hyeran Jeon, "CNN-Based Android Malware Detection," in *Proc. of 2017 International Conference on Software Security and Assurance (ICSSA)*, Altoona, PA, 60–65, 2017. Article (CrossRef Link).
[13] Chen, Xiao, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren, "Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection," *IEEE Transactions on Information Forensics and Security*, 15, 987–1001, 2019. Article (CrossRef Link).
[14] Spirtes, Peter L, "Directed Cyclic Graphical Representations of Feedback Models," *arXiv:1302.4982 [Cs]*, February 2013. Article (CrossRef Link).
[15] Xiao, Xi, Zhenlong Wang, Qing Li, Shutao Xia, and Yong Jiang, "Back-Propagation Neural Network on Markov Chains from System Call Sequences: A New Approach for Detecting Android Malware with System Call Sequences," *IET Information Security*, 11(1), 8–15, 2017. Article (CrossRef Link).
[16] Zang, Dong, Jinhai Liu, and Huaizhen Wang, "Markov Chain-Based Feature Extraction for Anomaly Detection in Time Series and Its Industrial Application," in *Proc. of 2018 Chinese Control and Decision Conference (CCDC)*, Shenyang, 1059–1063, 2018. Article (CrossRef Link).
[17] Yong, B., Liu, X., Yu, Q., Huang, L., & Zhou, Q, "Malicious Web traffic detection for Internet of Things environments," *Computers & Electrical Engineering*, 77, 260-272, 2019. Article (CrossRef Link)
[18] Chereau, Jean P., Bruno Scalzo Dees, and Danilo P. Mandic, "Robust Principal Component Analysis Based on Maximum Correntropy Power Iterations," *arXiv:1910.11374 [Cs, Eess, Math, Stat]*, October 2019. Article (CrossRef Link).
[19] Hadri, Amal, Khalid Chougdali, and Rajae Touahni, "Intrusion Detection System Using PCA and Fuzzy PCA Techniques," in *Proc. of 2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, Marrakesh, Morocco, 1–7, 2016. Article (CrossRef Link).
[20] Liu, Liang, Jianchang Liu, Xia Yu, Honghai Wang, and Zhaoqiang Chen, "A multivariate monitoring method based on PCA and Dual Control Chart," in *Proc. of 2019 Chinese Control And Decision Conference (CCDC), IEEE*, 2019. Article (CrossRef Link).
[21] Shlens, Jonathon, "A Tutorial on Principal Component Analysis," *arXiv:1404.1100 [Cs, Stat]*, April 2014. Article (CrossRef Link).
[22] Alam, Saruar, Moonsoo Kang, Jae-Young Pyun, and Goo-Rak Kwon, "Performance of Classification Based on PCA, Linear SVM, and Multi-Kernel SVM," in *Proc. of 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, Austria, 987–989, 2016. Article (CrossRef Link)
[23] Patil, Siddalingeshwar, and Umakant Kulkarni, "Accuracy Prediction for Distributed Decision Tree Using Machine Learning Approach," in *Proc. of 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 1365–1371, 2019. Article (CrossRef Link).
[24] Xiang, Yiyao, Lei Li, and Wanting Zhou, "Random Forest Classifier for Hardware Trojan Detection," in *Proc. of 2019 12th International Symposium on Computational Intelligence and Design (ISCID)*, Hangzhou, China, 134–137, 2019. Article (CrossRef Link).
[25] Zhou, Q., Yong, B., Lv, Q., Shen, J., & Wang, X., "Deep Autoencoder for Mass Spectrometry Feature Learning and Cancer Detection," *IEEE Access*, 8, 45156-45166, 2020. Article (CrossRef Link)

[26] Habi, Hai Victor, and Hagit Messer, "RNN Models for Rain Detection," in *Proc. of 2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, Nanjing, China, 184–88, 2019. Article (CrossRef Link).
[27] Yong B, Shen J, Liu X, et al., "An intelligent blockchain-based system for safe vaccine supply and supervision," *International Journal of Information Management*, 52(2020), 102024, 2020. Article (CrossRef Link).
[28] Bradley, Andrew P, "The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms," *Pattern Recognition*, 30 (7), 1145–1159, 1997. Article (CrossRef Link).



SHANXI LI (Member, IEEE) received the M.S. and PhD degrees in computer sciences from Lanzhou University, in 2009 and 2021, respectively. His research interests include computer networks, computer security, artificial intelligence and machine learning.

Qingguo Zhou received the BS and MS degrees in Physics from Lanzhou University in 1996 and 2001, respectively, and received PhD in Theoretical Physics from Lanzhou University in 2005. Now he is a professor of Lanzhou University and working in the School of Information Science and Engineering. He is also a Fellow of IET. He was a recipient of IBM Real-Time Innovation Award in 2007, a recipient of Google FacultyAward in 2011, and a recipient of Google Faculty Research Award in 2012. His research interests include safety-critical systems, embedded systems, and real-time systems.

Wei Wei (SM'17) received the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2005 and 2011, respectively. His current research interests include the area of wireless networks, wireless sensor networks application, image processing, mobile computing, distributed computing, and pervasive computing, Internet of Things, and sensor data clouds. Dr. Wei is a Senior Member of the China Computer Federation.