Byte-level malware classification based on Markov images and deep learning

YanH 1023041011

***Abstract*—In recent years, malware attacks have become serious security threats and have caused huge losses. Due to the rapid growth of malware variants, how to quickly and accurately classify malware is critical to cyber security. As traditional methods based on machine learning are limited by feature engineering and difficult to process vast amounts of malware quickly, malware classification based on malware images and deep learning has become an effective solution. However, the accuracy rate of existing method based on gray images and deep learning (GDMC) still needs to be improved. Moreover, it is heavily dependent on the amount of training dataset. To improve the accuracy, this paper proposes a byte-level malware classification method based on markov images and deep learning referred to as MDMC. The main step in MDMC is converting malware binaries into markov images according to bytes transfer probability matrixs. Then the deep convolutional neural network is used for markov images classification. The experiments are conducted on two malware datasets, the Microsoft dataset and the Drebin dataset. The average accuracy rates of MDMC are respectively 99.264**

***Index Terms*—Malware, Markov images, Deep learning, Classification**

I. Introduction

Malware is the computer program that infiltrates and damages a computer without the user's consent. With the popularity of internet, malware has become a profitable tool and political weapon for criminals. The cyber attacks and cyber crimes using malware have increased dramatically in recent years1. For example, the global ransomware, WannaCry, infected more than 230,000 computers and caused losses of billion in April 2017 (BBC, 2017). In March 2019, Norway's Norsk Hydro, one of the world's largest aluminum producers, was attacked by a new ransomware, LockerGoga, and forced to temporarily close multiple plants (Ionut, 2019). Malware attacks have become serious security threats to industrial production and have caused huge losses.

According to Rising's 2018 annual report (Rising, 2018), the Rising Cloud Security System intercepted a total of 77.86 million virus samples and found 1.125 billion virus infections. The total number of viruses increased by 55.63 compared with the same period in 2017. Especially in popular malware attacks such as mining malware and ransomware, the number of malware has increased significantly. The vast majority of new malware variants come from known malware with less than 2 code differences between variants according to the study of Greengard and Samuel (Greengard, 2016). It indicates that most malware samples of the same family have certain similarity in functionality, which is an important basis for malware classification. Faced with the endless of malware variants, how to quickly and accurately classify malware variants is critical to cyber security.

The current methods (Catak and Yazı, 2019, 2018) for malware classification are mainly based on static analysis or dynamic analysis to obtain features. Most of these methods analyze the bytecode, assembly code, file structure or dynamic behavior of malware, and then use machine learning algorithms for classification. However, extracting static features of malware disassembly code often meets the problem of reverse analysis. When static analysis is not effective,dynamic analysis can be directly used to analyze malware behaviors. But dynamic analysis can only analyze the behavior of a specific execution path, it is hard to traverse all the paths. Meanwhile, dynamic analysis is time-consuming. These methods are not only limited by reverse analysis and time consumption, but also rely heavily on manually constructing features. Therefore, it is difficult to cope with the explosive growth of malware variants.

It can be applicable to various systems such as windows and android. The main contributions of this paper are summarized as follows:

• A kind of malware image based on markov transfer probability matrix is proposed,

which can generate malware images with fixed size and effectively reduce the redundancy of bytes information.

• The deep convolutional neural network designed in this paper has fewer fully connected layers and lower output dimensions than that of VGG16 (Simonyan and Zisserman, 2014). Therefore, its parameters of fully connected layer are less.

• A framework combined markov images with deep convolutional neural network is proposed for malware classification. It has better performance than GDMC and doesn't rely on pre-trained models. The remainder of this paper is structured as follows:

The Section 2 reviews the related work of malware classification. Then Section 3 details the malware classification method based on markov images and deep learning. Experimental evaluation is given in Section 4. Finally, Section 5 concludes this paper.

II. Related Work

Traditional malware classification methods were mainly based on static analysis or dynamic analysis to obtain features (Gandotra et al., 2014). Then machine learning algorithms were used for classification. On the aspect of malware images classification, Nataraj et al. (2011) first proposed the malware analysis method based on gray images of malware binaries. The binary files were converted to gray images and then the

gist texture features were extracted for malware classification. As deep convolu-tional neural networks such as VGGNet, ResNet, and DesNet have showed significant advantages for large-scale image classification tasks (Russakovsky et al., 2015), traditional malware images classification methods have gradually been replaced by deep learning (Hasanpour et al., 2018). Therefore, it reviews the related work of malware classification from two main aspects: the methods based on feature extraction and machine learning, the methods based on malware images and deep learning.

2.1 The methods based on feature extraction and machine learning

According to the different feature extraction methods, malware classification methods based on machine learning can be divided into two categories: based on static features, based on dynamic features. The texture feature of malware images also belongs to a kind of static features.

2.1.1. Based on static features

Most of these methods analyze the bytecode, disassembly code, file structure, etc. Without actual execution, static analysis has the advantages of fast speed and high efficiency. There were various studies of malware classification with static features. For example, Shalaginov et al. (2018) conducted an indepth survey of different machine learning methods for classification of static characteristics of Windows portable executable (PE) files and offered a tutorial on how different machine learning techniques can be utilized in extraction and analysis of a variety of static characteristic of PE binaries. A labeled benchmark dataset, EMBER, was built by Anderson and Roth (2018) for training machine learning models to statically malware analysis. The dataset included features extracted from 1.1M PE binaries: 900K training samples (300K malicious, 300K benign, 300K unlabeled) and 200K test samples (100K malicious, 10 0K benign). Ahmadi et al. (2016) extracted static features from malware binary files and disassembled files. Integrating multi-dimensional features, this method classified 9 malware families with an accuracy rate of 99.77%. In terms of the methods based on texture feature of malware images, it converts the problem of malware classification into the problem of images classification. For instance, Liu and Wang (2016) designed a system to detect and classify the malware by converting disassembly files into gray images. The gray images were compressed by local mean method and mapped to feature vectors. To classify malware, an ensemble learning method based on k-means and diversity selection was proposed. Han et al. (2015) proposed a malware classification method that converted malware binary files into gray images and entropy graphs. The experimental results showed that their method can effectively distinguish malware families. However, extracting static features of malware disassembly code often meets the problem of reverse analysis. The method based on static features would be invalid, when it failed to obtain malware disassembly code.

2.1.2. Based on dynamic features

In dynamic analysis, malware is automatically executed in sandbox or controlled physical environment to observe its behavior. When static analysis is noneffective, dynamic analysis can be directly used to analyze malware behaviors. The dynamic analysis has the advantages of low false alarm rate and high accuracy rate because it obtains the actual runtime behaviors of malware, such as file access, API call, network communication, etc. For example, Lim et al. (2015) presented a malware classification method based on clustering of flow features and sequence alignment algorithms. With real malware traffic, they identified the most appropriate method for classifying malware by the similarity of network activity. In 2017, a methodology was presented by Pektas and Acarman (2017) to build the feature vector by using run-time behaviors and apply online machine learning algorithms for malware classfaction.

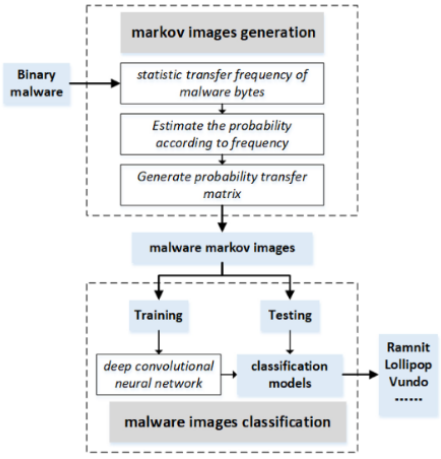

Fig. 1. The framework of malware classification based on markov images and deep learning.

III. The Proposed Method

To improve the accuracy of malware classification, the MDMC is proposed based on markov images and deep learning. The framework of MDMC is shown in Fig. 1, which includes two steps: markov images generation and malware images classification. The first step is converting malware binaries into markov images. The markov images based on bytes transfer probability matrixs are pixel matrixs with fixed size. Compared with gray images, it does not consider the problem of resizing. The second step is malware images classification with deep convolutional neural network. The malware images are divided into training dataset and testing dataset. When training, only the samples in training dataset are used. The testing dataset is used to evaluate the trained models.

3.1. Markov images generation

In most current malware classification methods based on malware images, the bytes of malware binaries are directly used as pixel points in gray images. The influence of information loss in the pre-processing of truncation or scaling of malware images on the final accuracy is not considered. Moreover, it does not take into account the negative impact of large amount of information redundancy brought by this direct conversion. To alleviate the above problems, the malware markov images based on bytes transfer probability matrixs are proposed. The bytes of malware binaries are viewed as

bytes stream which can be represented as a stochastic process as follow:

$$Byte_i, i \in \{0, 1, \ldots, N-1\} \tag{1}$$

where N refers to the length of malware binary. Because most new malware comes from known malware with minimal code differences, malware of the same family also has great similarity in bytes distribution. Assuming that the probability of Bytei is only related to Bytei1, then the random variable Bytei is a markov chain:

$$P(Byte_{i+1}|Byte_0, \ldots, Byte_i) = P(Byte_{i+1}|Byte_i) \tag{2}$$

Since the value of each byte ranges from 0 to 255, Bytei has 256 possible states, Bytei 0, 1, ..., 255. The markov transfer probability matrix is generated based on the transfer probability of each state. If Pm,n indicates the transfer probability that the subsequent byte of m is n, then the formula for Pm,n is as follow:

$$p_{m,n} = P(n|m) = \frac{f(m,n)}{\sum_{n=0}^{255} f(m,n)} \tag{3}$$

where f(m, n) indicates the frequency at which m is followed by n, and m, n ∈ 0, 1, ..., 255. Considering the probability that all bytes are transferred to each other, the transfer probability matrix M is formed:

$$M = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,255} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,255} \\ \vdots & \vdots & \ddots & \vdots \\ p_{255,0} & p_{255,1} & \cdots & p_{255,255} \end{bmatrix} \tag{4}$$

Malware markov images are generated based on the matrix M. In markov images, each transfer probability Pm,n corresponds to a pixel point at Mm,n. The schematic diagram of converting malware binaries into markov images is shown in Fig. 2.

The steps of markov images generation are summarized as follows:

1. Step 1: The malware binaries are viewed as byte streams. The transfer frequency of each byte to other bytes is counted.

2. Step 2: The probability is estimated by frequency according to the formula 3. The transfer probability of each byte to other bytes is calculated.

3. Step 3: The byte transfer probability matrix is generated according to the formula 4. The markov images of malware are generated based on the matrix M.

Because the size of malware markov images is fixed, they can be directly used as input when classification with deep learning. On the other hand, the pixel values of malware markov images are byte transfer probabilities rather than the actual values of bytes. They represent the distribution characteristics of malware binaries and can reduce the information redundancy to some extent.

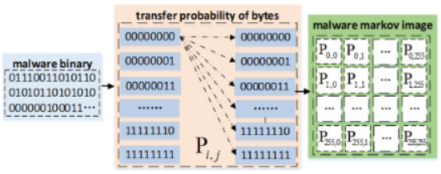

Fig. 2. The schematic diagram of converting malware binaries into markov images.

TABLE I
THE MICROSOFT DATASET AND DREBIN DATASET.

Dataset	Family	ClassID	Samples
Microsoft	Ramnit	1	1541
	Lollipop	2	2478
	Kelihos_Ver3	3	2942
	Vundo	4	475
	Simda	5	42
	Tracur	6	751
	Kelihos_Ver1	7	398
	Obfuscator.ACY	8	1228
	Gatak	9	1013
	Total		10868
Drebin	Plankton	1	624
	DroidKungFu	2	667
	GinMaster	3	339
	FakeDoc	4	132
	FakeInstaller	5	925
	Opfake	6	613
	BaseBridge	7	329
	Kmin	8	147
	Iconosys	9	152
	Geinimi	10	92
	Total		4020

IV. EXPERIMENTS

The MDMC is based on markov images and deep learning, which can effectively improve the accuracy of malware classification compared with GDMC. Secondly, most current methods (Bhodia et al., 2019; Gibert et al., 2019; Kalash et al., 2018; Ni et al., 2018; Rezende et al., 2017) only consider the case that the training samples are relatively sufficient. For example, when the training dataset accounts for 90% or 80% of the whole dataset, these methods can achieve high accuracy. Therefore, it also carried out experimental analysis when lack of training samples. Experiments are conducted on two kinds of malware datasets with the same structure of DCNN. The following is detailed introduction from three aspects: the datasets and experimental settings, evaluation indicators, and experimental results.

4.1. The datasets and experimental settings

The comparative experiments are conducted on two malware datasets, the Microsoft dataset (Ronen et al., 2018) and the Drebin dataset (Arp et al., 2014), as shown in Table 1. The Microsoft dataset has 10,868 labeled samples that come from 9 malware families. The malware samples in the Microsoft dataset are unpacked. When converting malware into markov images, only the binary file of each malware is used. In the Drebin dataset, the top 10 malware families are selected and a total of 4020 android malware samples are used for experiments. To eliminate the influence of irrelevant

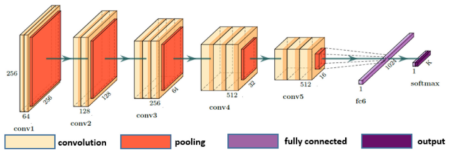

Fig. 3. The structure of deep convolutional neural network for malware images classification.

data, only the executable, .dex file, is used for malware markov images generation.

In terms of experimental settings, the DCNN is built in Python with keras2 and tensorflow3. The network is trained on an ubuntu server with 4 GPUs. As for hyperparameters, the learning rate is 1e3 which is consistent with the previous studies (Kalash et al, 2018; Simonyan and Zisserman, 2014). According to the memory size of GPU and time consumption of training, it selected suitable values for other hyperparameters. The batch size is 32, the training epoch is 250 and the decay is 1e 6. The problem of hyperparameter optimization is not further studied because it does not improve classification accuracy significantly.

4.2. Evaluation indicators

To quantitatively evaluate the experimental results, the accuracy and logarithmic loss (logloss) are used as indicators to compare the performance of various methods on the whole. The accuracy refers to the fraction of correct predictions. The logloss is the cross entropy between the distribution of the true labels and the predicted probabilities, as shown in Eq. 5:

$$\log \text{loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \tag{5}$$

where N is the number of observations, M is the number of class labels and log is the natural logarithm. If observation i is in class j,y_{ij} is 1. Otherwise, it is equal to 0. The p_{ij} represents the predicted probability that observation i is in class j.

Because the distribution of malware families is imbalanced, the accuracy and logloss are difficult to reflect the classifier's performance on each class, especially the minority. Therefore, precision, recall and f1-score are also used as indicators for each malware class, so as to comprehensively evaluate the performance.

4.3. Experimental results

Similar to most studies on malware classification, it firstly compares the methods, MDMC and GDMC, when training dataset accounts for 90% and testing dataset only accounts for 10%. Two different types of malware datasets are used for experimental analysis. On each dataset, the accuracy and logloss of two methods are compared firstly. Then the precision, recall and f1-score are used to evaluate the performance on each malware class. It also fully considers the case of insufficient training samples by gradually reducing the proportion of training dataset and testing dataset. To make experiments comparable, the experiments are conducted under the same settings of DCNN.

4.3.1. Experiment on Microsoft dataset As shown in Fig. 4, it is the change trend of accuracy with training epochs.

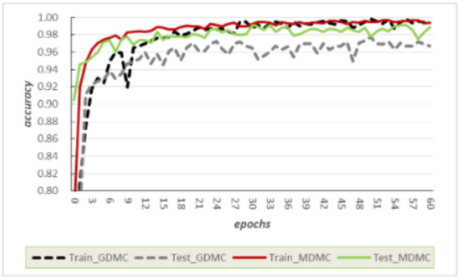

Fig. 4. The change trend of accuracy with training epochs on Microsoft dataset.

The experimental datas are recorded by TensorBoard4. To clear display, the experimental datas of the first 60 epochs are used. The horizontal axis indicates training epochs, and the vertical axis indicates accuracy. The black and gray dashed lines represent the training accuracy and testing accuracy of GDMC respectively. The red and green solid lines represent the training accuracy and testing accuracy of MDMC respectively. It can be seen from the figure that the training accuracy of two methods converges rapidly with the increase of training epochs. Compared with GDMC, the training accuracy of MDMC converges faster. Secondly, when the training accuracy is stable, the testing accuracy of GDMC can reach about 96%. However, the testing accuracy of MDMC fluctuates in the range of 98% to 100%.

V. Conclusion

Because static reverse analysis and dynamic analysis respectively have their own limitations, traditional machine learning algorithms are generally difficult to process massive unknown malware samples. This paper proposed a byte-level malware classification method called MDMC which converted malware binaries into markov images and classified malware markov images with deep learning. Only the binaries of malware were used without the reverse analysis and dynamic analysis. MDMC could be applicable to various systems such as windows and android. Compared with similar methods such as GDMC, MDMC could significantly improve the accuracy of malware classification.

Declaration of Competing Interest

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