Chinese Journal of Electronics
Vol.29, No.1, Jan. 2020

Visualization Feature and CNN Based Homology Classification of Malicious Code

CHU Qianfeng, LIU Gongshen and ZHU Xinyu

(*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*)

Abstract — The malicious code brings a serious security threat. Researchers have found that many new types of malicious code are variants of the existing one. The homology classification of the unknown malicious code can find its corresponding family in which all the code share inherent similarities from the database, so that the defenders can make rapid response and processing. We use the algorithm of malicious code visualization to translate the homology classification problem into the image classification problem. A convolution neural network for malicious code image is constructed. We train it to complete the malicious code homology classification on two different datasets. The results show that our work outperforms most of existing work with the accuracy of 98.60%.

Key words — Malicious code, Homology classification, Malicious code image, Convolutional neural network.

I. Introduction

The advent of the Internet era and the rapid development of information technology bring people with a variety of convenience, but also the potential information security issues, especially network security issues. It threatens the security of user information property, and malicious code is one of them.

In recent years, malicious code has caused millions of losses to various types of systems. Security researchers found that many new malicious codes are derived from the existing malicious code variants. The authors use the techniques of deformation, shell, polymorphism, code disruption, *etc.* to deal with the original code to avoid the traditional malicious code detection techniques such as signature matching. And these codes tend to have a highly similar structure, the same function call order and code writing habits. We define the homologous malicious code as the malicious code written by the same code writers or the same team, with the inherent similarity and relevance. The homology classification of the unknown malicious code can identify the homologous malicious code that has been recorded in the library with similar characteristics, so as to make rapid response and processing.

II. Related Work

Early in 1998, Goldberg[1] proposed the theory of constructing a family tree of viruses, as the earliest homologous study of malicious code. After that, researchers attempt to homogenously determine or classify the malicious code by extracting the static and dynamic feature of malicious code. Karim[2] constructed Tokenizer to convert executable files into pseudo-code, using n-grams and n-perms to obtain feature matrix, and using TF-IDF to calculate the similarity of code sequences. Walenstein compare the similarity of malicious code based on the method of maximum π pattern, a PQ tree based feature, and construct a family tree[3]. Wang[4] designed a "BMAT" system for searching matches between data blocks. Kinable[5] used graph matching(smallest graph editing distance) to calculate graph similarity values. The DBSCAN and K-Medoids clustering are used to classify families. Zuo[6] extracted features of node degree of function call graphs, and used specific set operations to obtain the features of the sample functions. Qiao[7] proposed a homology judgement method based on call habits of writers to calculate the similarity.

We try a novel method of homology classification-malicious code images. This idea was first proposed by Nataraj and Karthikeyan of the University of California in 2011[8]. At low-layer of the code, each malicious executable file consists of binary strings (0 and 1). This vector of 0 and 1 can be converted into a matrix by a specific rule, and finally presented in the form of an

Manuscript Received Sept. 3, 2018; Accepted Apr. 11, 2019. This work is supported by the National Natural Science Foundation of China (No.61772337, No.U1736207) and the SJTU-Shanghai Songheng Content Analysis Joint Lab and program of Shanghai Technology Research Leader (No.16XD1424400).

 DOI: 10.1049/cje.2019.11.005

image, which is called malicious code visualization. In Ref.[8], Nataraj and Karthikeyan generated the GIST feature after getting the malicious code image and used the KNN model to complete the classification. After that, researchers have done further researches on malicious code images. Kancherla[9] extracted features based on grayscale and texture from malicious code images and used support vector machines for positive and negative classification. Combining the malicious code image with 12 other features such as n-gram, sequence entropy, operation code, Mansour Ahmadi[10] used XGBoots to complete family classification. Ahmadi[11] calculated the entropy value of each row of pixels of the malicious code image, generated an entropy histogram to compare the malicious code similarity. Yan[12] propose a Pairwise rotation invariant co-occurrence local binary pattern (PRICoLBP) feature, and further extend it to incorporate the TFIDF transform.

III. Methodology

1. Malicious code image

The common malicious code homology classification techniques require feature engineering. They extract feature vectors. As a result, the quality of feature vector extraction methods directly affect the result of homologous determination. With the upgrading of counter-killing techniques for malicious code, obfuscating techniques such as code rearrangement, spam insertion, and anti-tracing has resulted in insufficient or even inaccurate feature vector representations, which has ultimately led to a decrease in the accuracy of classification.

The main idea of the malicious code image is to map the malicious code into a corresponding grayscale image, and use the texture features in the image to analyze and detect the malicious code. Compared to artificially extracted feature vectors, malicious code images contain rich, almost all malicious code information. Either through direct analysis of image textures (feature descriptor extraction) or higher levels analysis after abstraction (image classification based on deep learning), the malicious code image can minimize the influence of the obfuscation technique and provide new ideas for the research of homology classification.

2. Malicious code visualization

When we analyze the binary file, it is found that the corresponding hexadecimal value range of each byte segmented is [00, FF] and the corresponding decimal value is 0-255. It covers the entire range of gray values, with 0 for black and 255 for white. Disassemble the malicious executable file to get its binary string, divide it into bytes to get the pixel value into the array, map the binary string into the array of pixel gray value, and visualize the array as the malicious code image. Algorithm 1 shows the malicious code visualization algorithm.

Fig.1 shows malicious code images from two families generated from the above algorithm. According to a simple visual observation, it can be found that different malicious code samples from a family show similar images and are clearly distinguishable from malicious images of different families. As mentioned earlier, this may be owing to the fact that the new malicious code originates from existing malicious code, which have inherent similarities. At this point, the problem of homologous classification of malicious code is transformed into the problem of malicious code image classification. And the research of computer image processing and classification technology becoming more and more mature provides technical supports.

Algorithm 1 Malicious code visualization

In: str: malicious code binary string of length m;
Out: img: malicious code image
  initial $i = 0$;
  for $j = 0$ to m by 8 do
    turn $str[j : j + 8]$ into dec; /* $dec \in [0, 255]$ */
    $img_{\text{array}}[i] = dec$;
    i=i+1;
  end;
  generate img from img_{array};
  return img;
  end;

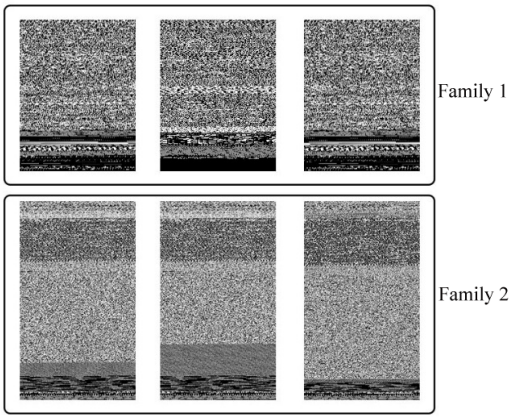

Fig. 1. Images from two families generated from Algorithm 1

3. Deep learning model selection

Based on the malicious code visualization algorithm, we combine the malicious code image with deep learning maturely developed in recent years. Deep learning provides feature learning, which can obtain higher-level abstract features that may not be recognized by human beings. And these higher-level abstractions can help capture relevant and invariable features.

Among deep learning models, Convolutional neural network (CNN) has shown its outstanding performance

in the fields of image and pattern recognition due to its characteristics such as local connection and multi-convolution kernel. We select the convolutional neural network as the homologous classification model. Compared to other models utilizing the malicious code image, such as Ref.[12] using local binary pattern, the CNN can extract deeper statistical characteristics. Actually, the multi-convolution kernels we apply in the CNN network can be trained to recognize the local pattern from multiple dimensions. Furthermore, as the number of layers grows, the extracted patterns will be unexplainable but rich of features from the original images.

1) Convolution and subsampling

CNN is a multilayer neural network model. Figure 2 is the model structure used in our experiment. Taking the malicious code image as the input (INPUT), the model applies the Convolutional (C) layers followed by the Subsampling (S) layer to reduce the dimensions of the feature maps. The number of layers is set as the hyperparameter and tuned in the experiment. Finally, the full connection layer and output (softmax) layer is linked to generate the prediction. After enormous experiments, we get the final structure with 10 convolutional layers and 5 subsampling layers.

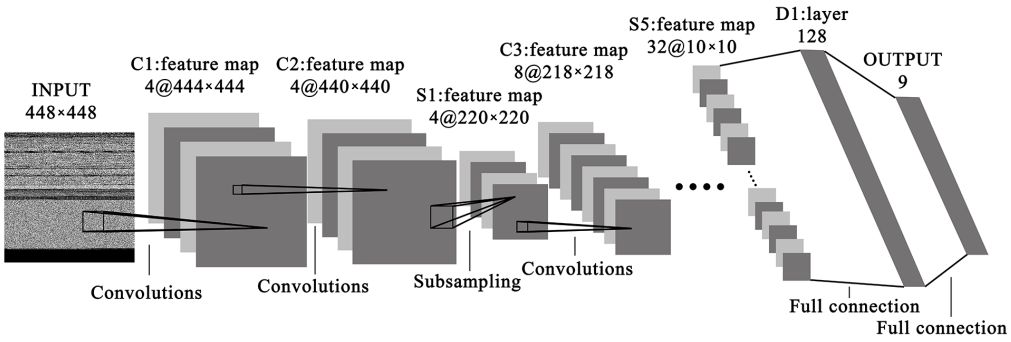

Fig. 2. Model structure detail of CNN used in our experiment

The convolutional layer and the subsampling layer are the core parts of the CNN network, shown in Fig.3.

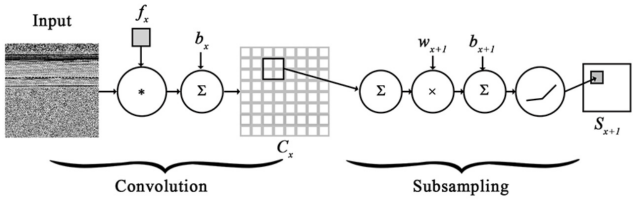

Fig. 3. Model structure detail of CNN used in our experiment

In a convolutional layer, the feature map of the previous layer is convoluted with a learnable kernel, and output through an activation function, shown by Eq.(1):

$$x_j^l = f(\sum_{i \in M_j} x_i^{l-1} \cdot k_{ij}^l + b_j^l) \tag{1}$$

where M_j is the set of input feature maps, k_{ij} denotes the weights and b_j denotes bias.

The subsampling layer is given by Eq.(2):

$$x_j^l = f(\beta_j^l \cdot down(x_j^{l-1}) + b_j^l) \tag{2}$$

where $down(.)$ denotes a sub-sampling function (maximum pool sampling is used here), b_j represents bias.

2) CNN training process

After the initialization of the CNN model, the model is trained. The training process is similar to the BP algorithm and is divided into two stages: forward propagation and backward propagation.

i) Forward propagation

In the forward propagation phase, each neuron calculates its output value based on the input fed to it. The loss function is used to calculate the error between the predicted value and the ground truth value. The square toot error cost function is a commonly used loss function for multi-classification problems. Assuming there are a total of N training samples, which are classified into C categories.

$$x^l = f(u^l), \text{ with } u^l = W^l \cdot x^{l-1} + b^l$$

$$E = \frac{1}{2}\sum_{k=1}^{C}(t_k - y_k)^2 = \frac{1}{2}\left\|t - y\right\|_2^2$$

$$E^N = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}(t_k^n - y_k^n)^2$$

ii) Backward propagation

The backward propagation is an iterative process that moves from the last level to the first level. Backward propagation gives a way to determine the output error of the previous layer, given the output of the current layer. Based on this method, the entire network weight can be

updated after iteration.

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial b} = \delta$$

$$\delta^l = (W^{l+1})^T \delta^{l+1} \circ f'(u^l)$$

$$\frac{\partial E}{\partial w^l} = x^{l-1}(\delta^l)^T$$

$$\Delta W^l = -\eta \frac{\partial E}{\partial w^l}$$

iii) Framework selection

As for the framework, we selected the deep learning library Keras to build the CNN network model. Keras is a highly modular deep learning network library that provides a simple and consistent API for direct user calls and provides GPU-accelerated services using the CUDA development environment to speed up model-training.

IV. Experiment and Analysis

Based on the visualization algorithm and deep learning model of CNN, we conduct experiments through training stage and test stage, as shown in Fig.4. In the training stage, all the training data are changed into image using our algorithm and sent into the CNN network. After training, the mature model serves as the homology classification tool of the system. And in the testing stage, when a piece of unknown malicious code arrives, we send it into the system, get the result, and conduct system performance analysis aiming to improve the model further.

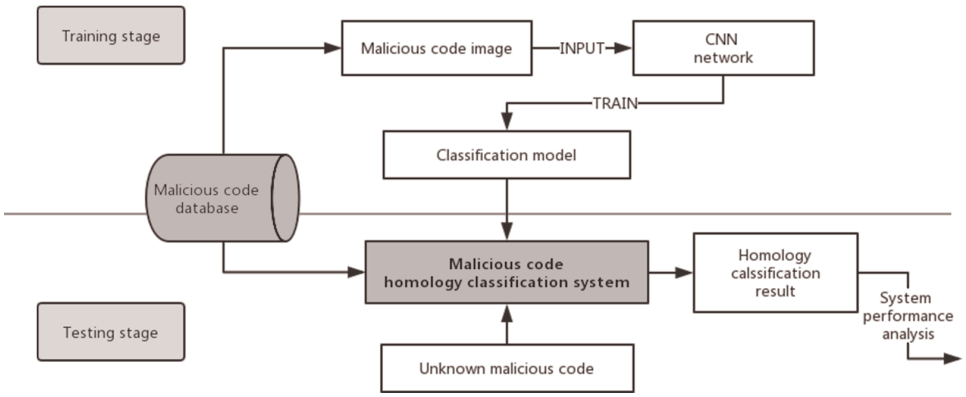

Fig. 4. Stage map of homology classification process

1. Data preprocessing

We do the experiments on two different datasets. The first dataset contains malicious code samples made public by the Kaggle platform under Microsoft Corporation[13]. The data set is tagged by family. More than 10,000 samples are divided into 9 families. Each malicious code sample in the data set has been disassembled by IDA. Corresponding binary files (∗.bytes) and assembly files (∗.asm) provide more options for data analysis and processing. And the other dataset contains malicious code collected from the VirusShare[14] website. They are all executable files. We tag them by the anti-virus software and randomly select 12 families for homology classification.

In the experiments performed with the Kaggle dataset, the assembly file (∗.asm) is selected as the processing object. The structure of assembly files is shown in Fig.5. The hex string is extracted using the regular expression match, and unnecessary information is left out, such as the "CC" string corresponding to the alignment instruction "align 10h". We use the Numpy library in Python to complete the generation of malicious code images, which provides numerical calculation extensions that show good performance when processing and storing large matrices.

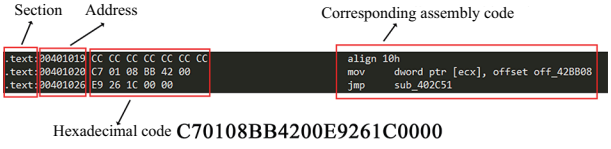

Fig. 5. The assembly file of malicious code

Since the CNN network requires the same size of image input, the images need to be cut or supplemented so that all malicious code images have the same size. We handle the size of image using the following strategy.

• If the length of the hexadecimal string is smaller than the predetermined image size, the zero value is added at the spare position (the black pixel block).

• If the length of the hexadecimal string is larger than the predetermined image size, the redundant part is cut.

The length of the malicious code can also be used as a criterion for the homologous classification, so we suggest the supplementary operation has little influence on the result. We conduct experiments with different image size input to determine which size to use. Fig.6 shows the training accuracy and validating accuracy of the corresponding best model using images of different size {128×128, 256×256, 448×448} in Experiment A. We do not try images larger than 448×448 for too much time and space consuming. Eventually we choose 448×448 as the predetermined image size.

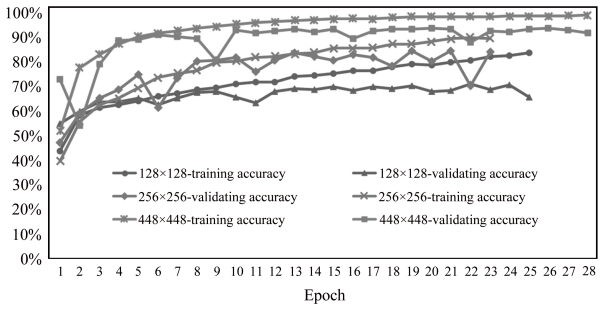

Fig. 6. Model accuracy with input images of different size

2. Experiment

1) Experiment A: Kaggle dataset

In the experiment, a 5-fold cross-validation method was adopted. The iteration round was controlled by the callback function Earystopping with 8. The loss function is Categorical_crossentropy; the optimizer is Rmsprop. We adjust the network structure, parameters and the input image size to complete the contrast experiment. According to a large number of contrast experiments, the model consists of 10 convolutional layers, 5 layers of pooling layers, and a fully connected layer structure.

After adding the callback function EARLYSTOPPING, the training is stopped when the verification loss no longer decreases in 8 consecutive iterations, and the final iteration number of experiments is 28. Fig.7 give the training accuracy/loss and the validation accuracy/loss respectively.

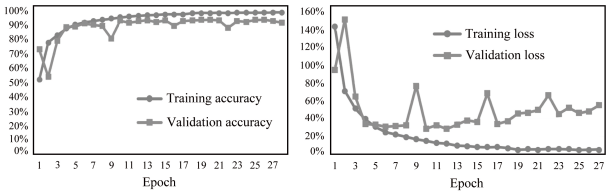

Fig. 7. The training accuracy/loss and the validation accuracy/loss

Class activation map (CAM) is generated for the sample, as shown in Fig.8. CAM indicates the input regions whose change would most contribute towards maximizing the output of class indices, that is, visualizes attention over the input. Through the CAM, we assume that the classification is mainly based on the structural characteristics of malicious code, such as the code length, the organizational structure of the data segment.

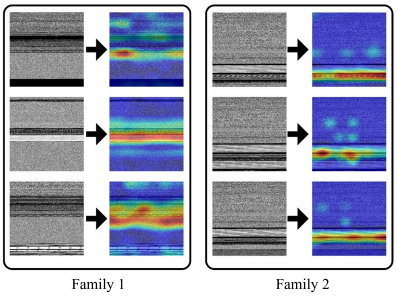

Fig. 8. Model CAM

All samples were judged according to the established model and the confusion matrix was generated as shown in Fig.9. The vertical axis indicates the actual family label of the sample, and the horizontal axis indicates the malicious code family number determined by the model. The block in the figure represents the percentage of a sample determination result with a gray value. For example, among malicious codes with an actual family number 2, there are 0.65% codes determined to belong to family 1, 99.03% family 2, and the rest family 4, 6, and 8 respectively. Analyzing the No.5 family with low accuracy, we find that the sample number is far less than other families. The training effect is poor. Dataset expansion can improve such low-accuracy problem. But the commonly used dataset expansion methods (horizontal flipping, random zooming, shearing, *etc.*) do not apply to malicious code image datasets well. Therefore, further experimental data collection is needed to improve the model. In general, our method can complete the homologous classification of malicious code with high accuracy.

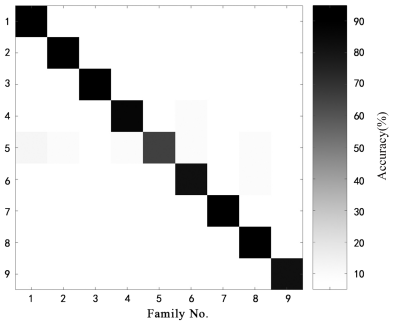

Fig. 9. The confusion matrix of Experiment A

2) Experiment B: Kaggle dataset

The assembly files are tagged with different fragments. Actually, a complete piece of malicious code includes ".text", ".data", ".idata", ".rdata", ".rsrc". The ".text" fragment contains program code and the ".data",

".idata", ".rdata" fragment includes all the variables and constants used in the program. In experiment B, we separate the text fragment and data fragment(including ".data", ".idata", ".rdata"), shown in Fig.10. Two different CNN models are trained to classify them seperately. We assume that classification by separate fragment may improve the performance because of coherent features inside the fragments.

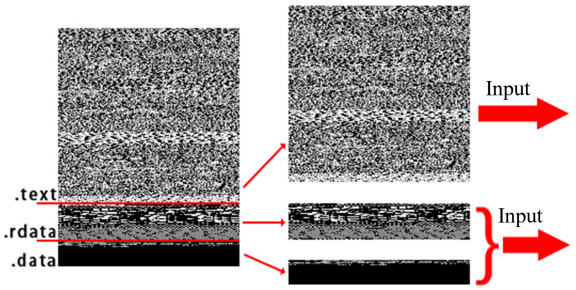

Fig. 10. Image fragment as input into two separate model

For a single piece of malicious code, we get the resulting array $y_{.\text{text}}$ and $y_{.\text{data}}$ after the softmax. For text fragment, $y_{.\text{text}} = [y_{t0}, y_{t2}, \cdots, y_{t8}]$; for data fragment, $y_{.\text{data}} = [y_{d0}, y_{d2}, \cdots, y_{d8}]$. Each array element y_{ti} or y_{di} represents the probability of the text/data fragment belonging to Family i. Eventually, the homology classification result is determined by Eq.(3):

$$family\_no. = \max\_index[\max(y_{.\text{text}}) > \max(y_{.\text{data}})?y_{.\text{text}} : y_{.\text{data}}] \quad (3)$$

where $\max\_index(\cdot)$ represents the array index of the maximum element. $* > *?* : *$ is the ternary operator for condition judgment. It means we choose the fragment as the final classification input whose maximum element is larger than the other's.

Separately, the text-model get the accuracy of 94.52%, and the data-model get the accuracy of 96.43%. But when we combine two results together according to Eq.(3), the accuracy approves to 97.77%.

3) Experiment C: Virus-share dataset

In Experiment C, we use Virus-share dataset to conduct the homology classification. All the samples are collected from the VirusShare Website. We tag them with the family name by the anti-virus software. The dataset information is listed in Table 1.

Following the process, we turn all the samples to malicious code image and train the CNN model. The optimal model performs the accuracy of 93.73%. The CNN model is deeper than the one in Experiment A and B for the larger dataset. It consists of totally 12 convolution layers, 4 maxpooling layers and 2 full-connected layers.

Table 1. Sample information of Virus-share dataset

Trojan.Antifw	8752
Trojan.Falsesign	6288
Adware.Bp-installer	6113
Adware.Multiplug	5002
Trojan.Generic	4576
Adware.Imali	4393
Adware.Lollipop	4093
Trojan.Outbrowse	3392
Trojan.Inject	3342
Trojan.Fakeav	3062
Trojan.Badur	2690
Trojan.Startpage	2673

3. Contrasts

On the experimental dataset published by Microsoft Kaggle platform, our system achieves as high accuracy as 98.6% using the image as a integrity. Experiment B using the fragment method provide a new idea to analyze malicious code, that is focusing on different fragments. And in Experiment C, another dataset is used to prove the algorithm effectiveness. The accuracy is a bit lower than that on Kaggle dataset. Several reasons may count: The family information is tagged by an anti-virus software which may cause mistake. The CNN model does not reach the top classification accuracy. We can try deeper CNN model to further improve the classification result.

We then do the lateral comparison among other classification systems using the malicious code visualization algorithm, the accuracy of which is shown in Table 2.

Table 2. Accuracy of malicious code visualization systems

Classification model	Accuracy	Dataset	Remarks
KNN[8]	98.00%	Anubis	Dataset not available
SVM[9]	95.00%	Offensive computing	/
XGBOOST[10]	99.77%	Kaggle	Image feature together with other 12 features
XGBOOST	97.36%	Kaggle	Using image feature alone
Histogram Similarity[11]	97.90%	VX Heavens	/
PRICoLBP[12]	98.60%	Kaggle	/
Our Model(CNN)	98.60%	Kaggle	/

Our CNN decision model is only slightly lower than the classification model implemented in Ref.[10]. Mansour's system based on XGBOOST gets the accuracy of 97.36% when using the image feature alone. When they try the combination of totally thirteen features, it owns the highest accuracy, including malicious code images, n-grams, sequence entropy, and operation codes. But it consumes high complexity of time and space. In the actual situation, we usually require fast access to the classification result without waiting for extra analysis and

feature extraction. Our CNN-based system outperforms it with a higher degree of automation.

Additionally, we assume our model possess of stronger extensibility and generality. We can accomplish the code prediction tasks utilizing the visualization algorithm and CNN model, such as malicious code shell class judgement and malicious code detection. We will dive into related researches in the future.

V. Conclusions

We mainly propose the realization of homologous classification system of malicious code based on deep learning. The feature selection focuses on the macroscopic malicious code image. Owing to the malicious code visualization algorithm and the powerful deep learning model, our system achieves excellent accuracy. But this research still encounters some shortcomings and needs to be improved, such as sample set selection, model parameter adjustment, *etc.* The pros and cons of the algorithm need to be further verified on other data sets. Its validity and accuracy will be further improved in deeper researches and experiments.

References

[1] L. Goldberg, P. Goldberg, C. Phillips, *et al.*, "Constructing computer virus phylogenies", *Journal of Algorithms*, Vol.26, No.1, pp.188–208, 1998.

[2] M.E. Karim, A. Walenstein, A. Lakhotia, *et al.*, "Malware phylogeny generation using permutations of code", *Journal in Computer Virology*, Vol.1, No.1-2, pp.13–23, 2005.

[3] M.E. Karim, A. Walenstein, A. Lakhotia, *et al.*, "Malware phylogeny using maximal pi-patterns", *European Institute for Computer Anti-Virus Research Conference*, Malta, pp.156–174, 2005.

[4] Z. Wang, K. Pierce and S. McFarling, "Bmat-a binary matching tool for stale profile propagation", *The Journal of Instruction-Level Parallelism*, Vol.2, pp.1–20, 2000.

[5] J. Kinable and O. Kostakis, "Malware classification based on call graph clustering", *Journal of Computer Virology and Hacking Techniques*, Vol.7, No.4, pp.233–245, 2011.

[6] L.M. Zuo, E.G. Liu, B.G. Xu, *et al.*, "Feature extraction and analysis technology of malicious code group", *Journal of Huazhong University of Science and Technology* (*Natural Science Edition*), Vol.4, pp.46–49, 2010.

[7] Y.C. Qiao, X.C. Yun, Y.Z. Zhang, *et al.*, "An Automatic Malware Homology Identification Method Based on Calling Habits", *Chinese Journal of Electronic*, Vol.44, No.10, pp.2410–2414, 2016.

[8] L. Nataraj, S. Karthikeyan, G. Jacob, *et al.*, "Malware images: visualization and automatic classification", *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, Pittsburgh, Pennsylvania, USA, pp.4, 2011.

[9] K. Kancherla and S. Mukkamala, "Image visualization based malware detection", *Computational Intelligence in Cyber Security*, Singapore, pp.40–44, 2013.

[10] M. Ahmadi, D. Ulyanov, S. Semenov, *et al.*, "Novel feature extraction, selection and fusion for effective malware family classification", *Computer Science*, Vol.8, No.3, pp.183–194, 2015.

[11] K.S. Han, J.H. Lim, B. Kang, *et al.*, "Malware analysis using visualized images and entropy graphs", *International Journal of Information Security*, Vol.14, No.1, pp.1–14, 2015.

[12] H. Yan, H. Zhou and H. Zhang, "Automatic malware classification via PRICoLBP", *Chinese Journal of Electronics*, Vol.27, No.4, pp.852–859, 2018.

[13] Microsoft, "Microsoft malware classification challenge", *https://www.kaggle.com/c/malware-classification/data*, 2015-2-3.

[14] VXShare, "VirusShare", *https://virusshare.com/*, 2018-5-25.

CHU Qianfeng received the B.S. degree in Shanghai Jiao Tong University (SJTU), China, in 2013. He is currently an M.S. candidate of SJTU. His research interest includes natural language processing, malicious code.
(Email: chuqianfeng@sjtu.edu.cn)

LIU Gongshen (corresponding author) received the Ph.D. degree from the Department of Computer Science, Shanghai Jiao Tong University, China, in 2003. He is currently an associate professor of SJTU. His research interests cover natural language processing, social networks. (Email: lgshen@sjtu.edu.cn)

ZHU Xinyu received the B.S. degree in Shanghai Jiao Tong University, China, in 2013. He is currently an M.S. candidate of SJTU. His research interest includes natural language processing, malicious code.
(Email: jasonzxy@sjtu.edu.cn)