A Malware Classification Method Based on Basic Block and CNN

Jinrong Chen[1,2(✉)]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
chenjinrong@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Aiming at solving the three problems ranging from considerable consumption of manpower in manual acquisition, to excessively high feature dimension and unsatisfying accuracy caused by manual feature acquisition, which will occur when using the current malware classification methods for feature acquisition. This paper proposes a malware classification method that is based on basic block and Convolutional Neural Network (CNN). The paper will firstly get the assembly code file of the executable malware sample, then extract the opcodes(such as "mov" and "add") of disassembled file of malware based on the label of basic block, and in the next, it will generate SimHash value vectors of basic blocks through these opcodes and a hash algorithm. Finally, the classification model is trained on the training sample set through using CNN. As we have carried out a series of experiments, and through these experiments, it is proved that our method can get a satisfying result in malware classification. The experiment showed that the classification accuracy of our method can achieve as highest as 99.24%, with the false positive rate being as low as 1.265%.

Keywords: Malware classification · Opcode · Basic block · Convolutional Neural Network

1 Introduction

The rapid development of information technology has not only brought a great deal of convenience to us, but has also brought potential security problems to networked computers. Among these problems, the most typical one is the attack and overflow of malicious code. Malware is a program or code that can spread through storage media and network, and can damage the integrity of computer system without our authorization and authentication [5]. Their types include viruses, worms, trojans, backdoors and spywares, etc. In recent years, the growth rate of malicious code is getting faster and faster, and the amount of malicious code is getting larger and larger. According to the threat report released by McAfee in August 2019 [9], more than 60 million cases of malicious code were

H. Yang et al. (Eds.): ICONIP 2020, CCIS 1332, pp. 275–283, 2020.
https://doi.org/10.1007/978-3-030-63820-7_31

added just in the first quarter of 2019. This has imposed great pressure on security vendors in analyzing and dealing with malicious code. And a fast and efficient method to classify malicious code could reduce the burden for many security vendors to analyze and deal with malicious code.

At present, there are still some problems exist in the malicious code classification technology. Firstly, when the malicious code classification technology uses machine learning to classify the malicious code, it needs to obtain the characteristics of the malicious code manually, and as a result, which will render the workload of obtaining the characteristics quite onerous. Secondly, because the classification model of malicious code relies on features acquired manually, which will lead to deficiencies in the comprehensiveness and effectiveness of feature acquisition, with the final classification accuracy being lower than expected. Therefore, we need to find a simple but effective method for malicious code classification, which could not only reduce the manual work in the process of obtaining classification features, but also ensure that the acquired features are conductive to the final classification accuracy. Aiming at solving problems existing in the current malicious code classification technology, this paper proposes a malicious code classification method that based on gray images that is generated through studying a large number of malicious codes and neural network technology. By considering basic blocks used by malware, we turn the malware classification problem into image recognition problem, then we prove its effectiveness through experiments. The main contributions of this paper are as follow:

(i) Proposed a basic block generation method and a reasonable method for generating gray image based on basic block;
(ii) Based on the gray image generating from malicious code, a suitable hash algorithm is found to generate basic block for classification of malicious code.

The paper is structured as follows. A survey on the related work is presented In Sect. 2. And in Sect. 3, we describe how we extract opcodes and generate the basic block. Then our experiments and discussion are presented in Sect. 4. Finally in Sect. 5 we conclude the paper.

2 Related Work

2.1 Malware Classification

Natalia Stakhanova et al. [16] proposed a malware classification method that is based on the value of network activity in 2011. And experimental study on a real-world malware collection demonstrated that their approach was able to group malware samples that behaved similarly. In 2012, Nikos Karampatziakis et al. [8] proposed a new malware classification system based on a graph induced by file relationships. Experiment showed that their method could be applied to other types of file relationships, and its detection accuracy can be significantly improved, particularly at low false positive rates. In 2013, Rafiqul Islam et al. [7] presented the first classification method of integrating static and dynamic features into a single test. Their approach had improved the previous results that

was based on individual features and had reduced half the time that was needed to test such features separately. In 2016, Ke Tian et al. [17] proposed a new Android repackaged malware detection technique that was based on code heterogeneity analysis. They had performed experimental evaluation through over 7,542 Android apps, and their approach can achieve a false negative rate of 0.35% and a false positive rate of 2.97%. In 2019, Di Xue et al. [18] proposed a classification system Malscore that was based on probability scoring and machine learning. And by carrying out experiments, they proved that the system of combining static analysis with dynamic analysis can classify the malware very efficiently.

2.2 Deep Learning

The recent success in deep learning research and development has drawn a lot of our attention [15]. In 2015, Google released TensorFlow [1], which is a framework of realizing deep learning algorithm. Deep learning is a specific type of machine learning with a lot of work on it [6,10]. The most well-known deep networks is convolutional neural network (CNN). CNN is composed of hidden layers, fully connected layers, convolution layers, and pooling layers. The hidden layers are used to increase the complexity of the model. And CNN is now being widely applied to the image recognition, and has achieved a good performance [2–4,19].

3 Model Construction

3.1 Extracting Basic Block

In this part, we mainly describe how to extract the basic block from the disassembled file of malware.

```
loc_415A81:
33 C0          xor eax, eax
locret_415A83:
C3             retn
```

```
sub_414076 proc near
33 C0          xor eax, eax
C3             retn
Sub_414076 endp
```

```
__ismbbkalnum_1:
8B FF          mov edi, edi
5D             pop ebp
C3             retn
```

Fig. 1. Some labels of basic block

The malware we are studying is given by Microsoft [12]. And according to our study, a disassembled file of malware mainly contains multiple subroutines, and most of them start with “proc near”, and end with “endp”. Inside the subroutine, its branches are usually marked with “loc_” or “locret_ ”. Outside the subroutine, the basic block is often marked with “XXX: ”(X is a letter or symbol), as can be seen in Fig. 1.

We first extract these identifiers and opcodes that belong to them from the disassembled file according to the labels.

3.2 Applying Hash Algorithm to the Basic Block

We use SimHash algorithm to process the opcode to generate the basic block of malware. Simhash is a fingerprint generation algorithm or fingerprint extraction algorithm mentioned in [11]. And it is widely used by Google in the filed of removing duplicate pages. As a kind of locality sensitive hash, its main idea is to reduce dimension. Take the following popular case for an instance. After Simhash dimensionality reduction, a certain number of text content may only get a 32 or 64 bit binary string composed of 0 and 1.

For example in this paper(please see the right side of Fig. 1), we extract the opcodes that belong to a basic block of malware, then perform a 4-bit hash function h to explain the SimHash calculation process, Supposing this below:

$$D = (w_1 = \text{“mov”}, w_2 = \text{“pop”}, w_3 = \text{“retn”})^T \tag{1}$$

Then, we can get each above keyword's hash value as follow:

$$\mathrm{h}(\mathrm{w}_1) = (1, 0, 0, 0)^T \tag{2}$$
$$\mathrm{h}(\mathrm{w}_2) = (1, 0, 0, 1)^T \tag{3}$$
$$\mathrm{h}(\mathrm{w}_3) = (1, 0, 1, 0)^T \tag{4}$$

In the SimHash algorithm of our paper, the opcode of each instruction belonging to the basic block is regarded as keywords when being referred to SimHash. To achieve simplicity, we treated each opcode on the same footing. Therefore, all have the same weight of 1. Then, we can get the weight vector (WV) through weight and hash value as follows:

$$WV(w_1) = (1, -1, -1, -1)^T \tag{5}$$
$$WV(w_2) = (1, -1, -1, 1)^T \tag{6}$$
$$WV(w_3) = (1, -1, 1, -1)^T \tag{7}$$

Then, we got a SimHash vector by adding up each WV and converting it into binary SimHash. In this example, we obtained the SimHash vector and SimHash value that is equal to “1000”.

$$\text{SimHash Vector} = (3, -3, -1, -1)^T \tag{8}$$

Finally, opcode sequence (OpcodeSeq) of each basic block of the malware is encoded to an n-bit SimHash value that has the same length relating to the selected hash algorithm.

According to the characteristics of the SimHash algorithm, each basic block will hash to similar SimHash values. Then we convert each SimHash bit into a pixel value ($0 \rightarrow 0$, $1 \rightarrow 255$). That is, when the bit value is 0, then the pixel value will be 0; and when the bit value is 1, then the pixel value will be 255. Then by arranging the n pixel dots in a matrix(each row of the matrix is a basic block), we convert the SimHash bits into a gray scale image. And in order to find out the appropriate hash algorithm for generating basic blocks of malicious code, we use some different hash algorithms (including multiple cascading hash functions) to generate the basic block, as shown in Table 1.

Table 1. Different hash algorithms versus image width

SimHash type	Cascading mode	Image width
SimHash128	MD5	128
SimHash256	SHA256	256
SimHash384	SHA384	384
SimHash512	SHA512	512
SimHash768	SHA512+SHA256	768
SimHash896	SHA512+SHA256+MD5	896
SimHash1024	SHA512+SHA384+MD5	1024

4 Evaluation

4.1 Malware Dataset

We use the dataset of [12], with each sample in the data set containing two files. Of which, one is hexadecimal file and the other is disassembled file. And as the disassembled file is generated from the IDA, we don't need to disassemble malware samples. The dataset has 10,868 disassembled files in 9 large malware families, and these samples have already been labelled the type of their families.

4.2 Evaluation Metrics

To evaluate the classifier's capability, True Positive Rate (TPR), False Positive Rate (FPR) and Accuracy are measured. TPR is the rate of malware samples correctly classified. FPR is the rate of malware samples falsely classified. The formulas of True Positive Rate (TPR), False Positive Rate (FPR) and Accuracy are given by Eq. 9, Eq. 10 and Eq. 11.

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

$$FPR = \frac{FP}{FP + TN} \tag{10}$$

$$Accuracy(\%) = \frac{TP + TN}{TP + FN + FP + TN} * 100 \tag{11}$$

Where TP is the number of malware samples correctly classified in their class, FP is the number of malware samples incorrectly classified in another class, FN is the number of malware samples incorrectly classified in their class, and TN is the number of malware samples correctly classified in other classes.

4.3 Experimental Results and Discussion

The CNN structure [14] is demonstrated in Fig. 2. And it is clearly shown in Fig. 2, as an input, each malware image needs to go through three convolution layers, two subsampling layers and three full connection layers. And over the processes of convolutions (C1, C2, C3), each convolutional layer will involve 32 learnable filters of the size of 2×2. During the processes of subsamplings (S1, S2), the max pooling with window size 2×2, is applied to reducing training parameters. And after each max pooling, a dropout layer with probability 0.5 could avoid overfitting from happening. After the second subsampling layer (S2), we flatten the output feature map, then link it to other three fully connected layers of dimensions 512, 256, and 9 (number of malware categories) respectively. Then, the first two fully connected layers will take *tanh* as activation function with the last one utilizing $softmax$ as activation function.

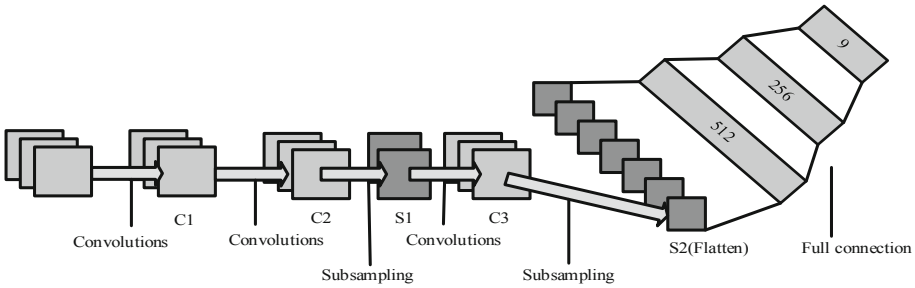

Fig. 2. CNN structure for malware image training

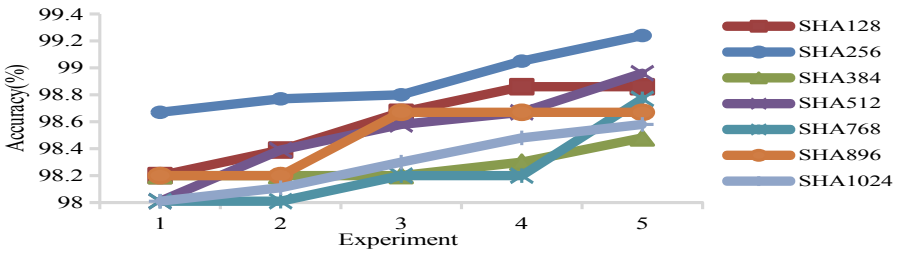

Fig. 3. Accuracy of hash algorithm

After preprocessing, 10,798 samples remained. 90% of them will be used for training, with the rest for testing. Experimental programs are written in Python, and the hardware environment is Intel®Xeon(R) CPU E5-2640 v3 @ 2.60GHz $\times$ 32 with 62.8 GB main memory. We repeat a number of times for each hash algorithm and select the top5 experimental results from them, as shown in Fig. 3, Fig. 4 and Fig. 5.

Table 2. Experimental results of SHA256

Experiment	Accuracy	TPR	FPR
1	98.67	99.17	**0.675**
2	98.77	99.61	0.740
3	98.80	99.59	1.718
4	99.05	99.60	1.290
5	**99.24**	**100**	1.265

From Fig. 3, we could know that in terms of accuracy, the best result is the hash algorithm SHA256, ranging from 98.6% to 99.3%, and the highest result is 99.24%. The experimental results of SHA256 are shown in Table 2. The worst result is the hash algorithm SHA768, and the accuracy of the rest of the hash algorithms(SHA128, SHA384, SHA512, SHA986 and SHA1024) are between 98% and 99%. According to our enlargement of Fig. 3, the order of the hash algorithms in terms of accuracy is as follows: $SHA256 > SHA128 > SHA512 > SHA896 > SHA1024 > SHA384 > SHA768$.

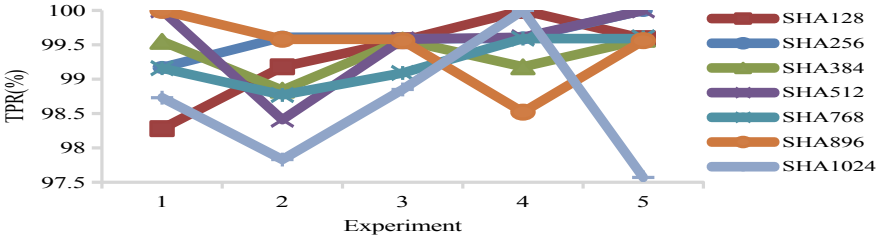

Fig. 4. True positive rate of hash algorithm

From Fig. 4, we could know that in terms of TPR, the best result seems to be achieved by hash algorithm SHA256, and the worst result is obtained by the hash algorithm SHA1024, and the TPR of other hash algorithms is between 98% and 100%. From Fig. 5, we could know that in terms of FPR, the best result is obtained by the hash algorithm SHA256, with its average FPR being 1.1376%. And the FPR of other hash algorithms is between 0.5% and 4%. From the analyses of Fig. 3, Fig. 4, and Fig. 5, we could know that SHA256 is the most suitable hash algorithm to generate the basic block of malicious code.

From Fig. 3, we could know that the difference between SHA256 and SHA384 is greater than that between SHA896 and SHA1024. From Table 1, we could know that the only difference between SHA896 and SHA1024 is that the former has cascaded SHA256, while the latter has cascaded SHA384. Therefore, it could be seen from Fig. 3 and Table 1, SHA256 is more negatively affected than SHA384 in the cascading hash function. A comparison of the gap between

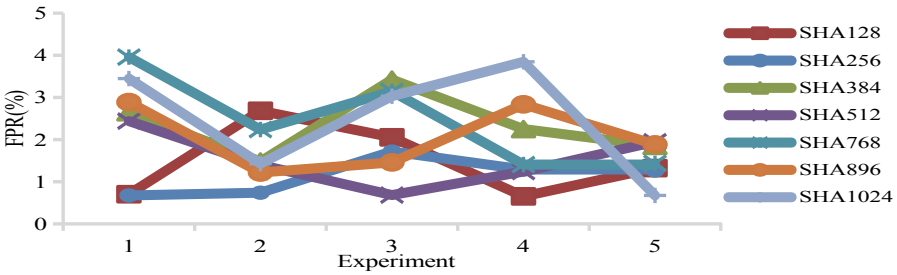

Fig. 5. False positive rate of hash algorithm

SHA256 and SHA768, as well as the comparison of the gap between SHA512 and SHA768 can also lead to this conclusion.

For comparison with our method, we choose the method proposed in Nataraj et al. [13] which has adopted GIST features of malware images and K-Nearest Neighbor (KNN, K = 3) classification. In the algorithm, GIST feature of 512 dimensions and KNN are used to classify these samples. The algorithm repeats 5 times for random sampling, and in the end, average accuracy of GIST-KNN algorithm on the test set of 9 families is 95.595%, the best accuracy of GIST-KNN algorithm gets 95.974%, and the average FPR of GIST-KNN algorithm is 4.31%. Compared with their method, ours is obviously more accurate.

5 Conclusion

Malware classification has become a major topic for research and concern owing to the continuing growth of malicious code in recent years. Aiming at tackling the problems caused by manual feature acquisition in the current malicious code classification method, we propose a method of malware classification that is based on basic block and convolutional neural network. Specifically, we propose a method for representing malware that relies on gray image that is generated from opcodes of the basic block. And experiments show that this method could achieve brilliant accuracy as high as 99.24%. Based on the method, the newly discovered malicious code samples can be accurately classified and their analysis efficiency can be improved.

References

1. Abadi, M., et al.: TensorFlow: a system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016), pp. 265–283. USENIX Association, Savannah (2016)
2. Alex, K., Ilya, S., Hg, E.: ImageNet classification with deep convolutional neural networks, pp. 1097–1105, January 2012
3. Gibert, D., Mateu, C., Planes, J., Vicens, R.: Classification of malware by using structural entropy on convolutional neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

4. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2014)
5. Grimes, R.A.: Malicious Mobile Code. Oreilly & Associates Inc. (2001)
6. Heaton, J., Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. Genet. Program. Evol. Mach. **19**(1–2), 1–3 (2017)
7. Islam, M.R., Tian, R., Batten, L., Versteeg, S.: Classification of malware based on integrated static and dynamic features. J. Netw. Comput. Appl. **36**, 646–656 (2013). https://doi.org/10.1016/j.jnca.2012.10.004
8. Karampatziakis, N., Stokes, J.W., Thomas, A., Marinescu, M.: Using File Relationships in Malware Classification. Springer, Heidelberg (2012)
9. Labs, M.: Mcafee labs threat report. McAfee Labs Threat Report (2019). https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf
10. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
11. Manku, G.S., Jain, A., Das Sarma, A.: Detecting near-duplicates for web crawling. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 141–150. Association for Computing Machinery, New York (2007). https://doi.org/10.1145/1242572.1242592
12. Microsoft: Microsoft malware classification challenge. Microsoft Malware Classification Challenge (2015). http://arxiv.org/abs/1802.10135
13. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.: Malware images: visualization and automatic classification, July 2011. https://doi.org/10.1145/2016904.2016908
14. Ni, S., Qian, Q., Zhang, R.: Malware identification using visualization images and deep learning. Comput. Secur. **77**(AUG), 871–885 (2018)
15. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
16. Stakhanova, N., Couture, M., Ghorbani, A.A.: Exploring network-based malware classification. In: 2011 6th International Conference on Malicious and Unwanted Software (2011)
17. Tian, K., Yao, D., Ryder, B., Tan, G.: Analysis of code heterogeneity for high-precision classification of repackaged malware. In: 2016 IEEE Security and Privacy Workshops (SPW), pp. 262–271, May 016
18. Xue, D., Li, J., Lv, T., Wu, W., Wang, J.: Malware classification using probability scoring and machine learning. IEEE Access **PP**(99), 1 (2019)
19. Yan, Z., et al.: HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In: IEEE International Conference on Computer Vision (2016)