Malware Detection Model Based on Deep Convolution Generation Adversarial Network

Cong Tang1, Jiangyong Shi1, Yi Yang1 and Yuexiang Yang1

College of Electronic Science, National University of Defense Technology, Changsha, Hunan Province, 410073, China
E-mail: tangcong18@nudt.edu.cn

Abstract. At present, malware is one of the biggest threats to Internet security. In this paper, a new static malware analysis algorithm MSG is proposed based on DCGAN. The algorithm transforms the disassembled malware code into a gray image based on SimHash, and uses DCGAN to generate countermeasure samples for training to detect unknown malware variants. The experimental results show that the detection rate of our algorithm for malware can reach 96.67%, and the dodge rate of generated malicious samples can reach 0.92 under the detection of CNN discriminator.

1. Introduction
Malware identification is a method to judge the security of computer software, which has always been a hot topic in computer security research. In terms of security, the game of attack and defense has never stopped. Since the existence of malware, the detection of malware has already attracted the attention of researchers. With the continuous innovation of existing technology, both sides are trying to use new technology to achieve breakthrough. Traditional fingerprint based malware detection methods can't deal with the rapid variation of malware detection, so machine learning based method to detect unknown threats has gradually become a consensus in the security industry. In the face of a large number of malware variants on the network, how to detect malware effectively, quickly and accurately is also an important issue.

The main works of this paper are as follows: (1) feature extraction of malware using LSH algorithm; (2) transforming malware into two-dimensional black-and-white image by visualization; (3) training detector with DCGAN generated image to realize the effectiveness of detector against unknown malware variants.

2. Related work
Malware detection methods are usually divided into two categories: static analysis and dynamic analysis. In static analysis, malware binaries are decomposed or decompiled without execution. Therefore, static analysis reveals the behavior of malware and prevents the operating system from malicious damage. However, in most cases, static analysis is not a simple task because attackers use code obfuscation techniques such as binary packers, encryption, or self modifying techniques to evade static analysis. In addition, static analysis does not allow a high degree of automation in the analysis process. In dynamic analysis, the behavior of malware is analyzed during execution in the debugger. At present, dynamic analysis based on sandbox is one of the most promising technologies.
3. Malware identification and detection

3.1. Feature extraction
We found that Sang Ni (2018) [1] has very good results in malware extraction and visualization algorithm for malware analysis. We extract the operating code of executable file of PE file in windows. Because different malware has different length opcodes, it is difficult to compare different length opcodes, so we use SimHash method to process them into hash codes with uniform length. LSH algorithm, namely SimHash, is a locally sensitive hash algorithm proposed by Charikar (2002) [2], which is mainly used for similar text recognition and webpage repetition recognition (Manku et al., 2007) [3]. SimHash can make Hash values comparable. That is, similar content will have similar hash values.

The main idea of SimHash is to use the dimension vector \( v \) of specified digits to represent the document, and the nth bit of \( v \) is determined by calculating the hash values of all keywords in the document. If the number of hash values 1 in the nth bit is greater than the number of hash values of 0, we set the nth bit of SimHash value to 1, otherwise the nth bit of SimHash value is set to 0.

For example, we use the first six operations in Figure 1 and a 5-bit hash function \( h \) to explain the process of SimHash. Supposing that

\[
D = (\omega_1 = "push", \omega_2 = "call", \omega_3 = "add", \omega_4 = "mov", \omega_5 = "test", \omega_6 = "mov")^T
\]

Then, we can get the hash value of each operation above as follows:

\[
h(\omega_1) = (1,0,0,0,1)^T \quad h(\omega_2) = (1,0,0,1,1)^T \quad h(\omega_3) = (1,0,1,1,1)^T
\]

\[
h(\omega_4) = (1,1,1,0,0)^T \quad h(\omega_5) = (0,1,1,0,1)^T \quad h(\omega_6) = (1,1,1,0,0)^T
\]

Each operation is regarded as a document, and for simplicity, if the weight is 1, we can get the weight vector of each operation through the weight and Hash value:

\[
WV(\omega_1) = (1,-1,-1,-1,1)^T \quad WV(\omega_2) = (1,-1,-1,1,1)^T \quad WV(\omega_3) = (1,-1,1,1,1)^T
\]

\[
WV(\omega_4) = (1,1,1,-1,-1)^T \quad WV(\omega_5) = (-1,1,1,-1,1)^T \quad WV(\omega_6) = (1,1,-1,1,-1)^T
\]

Then we get the SimHash vector by adding each WV and converting it to binary SimHash. The SimHash value is 10101.

\[
SimHashVector = (4,0,2,−2,2)^T
\]
similar SimHash values. Therefore, these similar binary values (with equal length) can be converted into similar images of equal size.

3.2. Malware image generation
After feature extraction, we get an n-bit SimHash code for each malware, so we convert each SimHash code into an image of the same size. We convert each SimHash bit into a pixel value \(0 \rightarrow 0, 1 \rightarrow 255\). That is, if the SimHash bit is 0, then the pixel value is 0; if the SimHash bit is 1, the pixel value is 255. Then, by arranging the n points into a matrix, we can get an image with only two pixel values. The size of the image is determined by the length of the SimHash value. We list the image sizes of different algorithms.

| Algorithm            | Image size |
|----------------------|------------|
| MD5 (the first 646 bits) | 8 x 8      |
| MD5 (128 bits)       | 8 x 16     |
| SHA-256              | 16 x 16    |
| SHA-512              | 16 x 32    |

In order to train DCGAN model, we need to adjust the image size to 64 x 64. Because we use SHA-256 Hash algorithm, we need to enlarge it by 4 x 4 times. That is, for each point \((I, J)\) in the original image, 16 points are obtained corresponding to the new image. That is, for any point \(Q_i(j)\), it will correspond to a 4 x 4 matrix block, as shown in Figure 4.

4. DCGAN training
Confrontational learning has always been an important research field in deep learning. Since Goodfellow[4] proposed the generation of confrontation network in 2014, its good generative ability has rapidly attracted the attention of a large number of researchers. GAN provides researchers with new ideas in solving problems. It sets two corresponding players as generator and discriminator. The purpose of generator is to learn and capture the potential distribution of real data and generate new samples. Discriminator is a classifier used to correctly determine whether the input data is real or from the generator.
We design a model for generating virtual goodwill samples based on deep convolution generative adversarial network (DCGAN) [5]. Virtual goodwill samples refer to the feature vectors generated by trained generation networks and can be recognized as good samples by the detection system. The schematic diagram of DCGAN is shown in Figure 5. Compared with traditional GAN, dcgan has the following improvements: (1) using convolution and deconvolution instead of pooling layer; (2) adding batch normalization operation in generator and discriminator; (3) removing full connection layer and using global pooling layer instead; (4) Tanh activation function is used in generator output layer and ReLU is used in other layers; (5) all layers of discriminator are activated by LeakyReLU Function.

We take a 100 dimensional random noise vector as the input, and then send it into the network. In the generator, we output a 64 x 64 dimensional feature vector through one feature recombination and four times convolution operations to generate a virtual goodwill sample library. Then, we use the virtual goodwill samples as malicious samples to train the discriminator again for retraining.

5. Experiment
The data set we use is the EMBER dataset [6], in which 800 samples are taken and divided into two parts, one part as the training set and the other as the detection set, and both use SimHash algorithm to generate images. And we use convolutional neural network(CNN) and MSG algorithm to train respectively.

5.1. Avoidance rate test
We randomly selected 100 malicious samples, used MSG algorithm to generate confrontation samples, and used convolutional neural network to detect them. As shown in Figure 6, the higher the complexity of SimHash algorithm is, the more fine the generated image will be. Therefore, the higher the accuracy achieved in the training process, the higher the avoidance rate of generating countermeasure samples to bypass detection And when we choose SHA-798 Hash algorithm, the avoidance rate reaches 0.92, which shows that our MSG algorithm generated confrontation samples are effective.
5.2. **Heavy training detection**

The detection rate of malware detection, also known as the call rate, represents the proportion of all malware samples correctly classified as malware. We use different hash functions and use the same DCGAN model for training. The detection results are shown in Figure 7. After using higher dimensional hash algorithm, the model detection rate will increase. When SHA-768 Hash algorithm is used, the detection rate can reach 96.67%. It can be clearly seen from the graph that the detection rate of the retrained discriminator is significantly higher than that of the discriminator trained only by convolutional neural network.

6. **Conclusion and future works**

In our experiment, windows malware is analyzed effectively, and malware image is generated by extracting malware operation code and SimHash algorithm, and the image generator and discriminator are trained effectively. We also need to carry out the next research in the following directions. (1) Using high-performance computing based on GPU or other parallel technologies to achieve faster malware detection and classification. (2) The proposed method is used in large-scale application environment. (3) Combining the static method with the dynamic analysis, it is easy to expand the robustness and adaptability of the detection system.

**Acknowledgement**

We would like to thank the anonymous reviewers for their valuable feedback. This work is supported by Education Ministry-China Mobile Research Funding (MCM20170404).

**References**

[1] Sang Ni, Quan Qian, Rui Zhang, et al. Malware identification using visualization images and deep learning[J]. Computers & Security 77. 2018: 871-885.

[2] Charikar MS. Similarity estimation techniques from rounding algorithms. Proceedings of the thirty-fourth annual ACM symposium on theory of computing, Montreal, Quebec, Canada; 2002. p. 380–8 . May.

[3] Manku GS, Jain A, Sarma AD. Detecting near-duplicates for web crawling. Proceedings of the international conference on world wide web; 2007. p. 141–50.

[4] Goodfellow I J, Pougetabadie J, Mirza M, et al. Generative Adversarial Networks[J]. Advances in Neural Information Processing Systems, 2014, 3:2672-2680.

[5] Radford A,Metz L,Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks[J]. Computer Science, 2015.

[6] Anderson HS, Roth P, et al. EMBER: an open dataset for training static PE malware machine learning models[J]. arXiv preprint arXiv: 1804.04637v2, 2018.