Research on Malware Detection and Classification Based on Artificial Intelligence

Li-Chin Huang[1], Chun-Hsien Chang[2], and Min-Shiang Hwang[3,4]
(Corresponding author: Min-Shiang Hwang)

Department of Information Management, Executive Yuan, Taipei 10058, Taiwan[1]
Department of Management Information Systems, National Chung Hsing University, Taiwan[2]
Department of Computer Science & Information Engineering, Asia University, Taichung, Taiwan[3]
500, Lioufeng Rd., Wufeng, Taichung 41354, Taichung, Taiwan, R.O.C.
Department of Medical Research, China Medical University Hospital, China Medical University, Taiwan[4]
(Email: mshwang@asia.edu.tw)

(Received Apr. 13, 2020; Revised and Accepted July 8, 2020; First Online July 9, 2020)

Abstract

Malware remains one of the major threats to network security. As the types of network devices increase, in addition to attacking computers, the amount of malware that affects mobile phones and the Internet of Things devices has also significantly increased. Malicious software can alter the regular operation of the victim's machine, damage user files, steal private information from the user, steal user permissions, and perform unauthorized activities on the device. For users, in addition to the inconvenience caused by using the device, it also poses a threat to property and information. Therefore, in the face of malware threats, if it can accurately and quickly detect its presence and deal with it, it can help reduce the impact of malware. To improve the accuracy and efficiency of malware detection, this article will use deep learning technology in the field of artificial intelligence to study and implement high-precision classification models to improve the effectiveness of malware detection. We will use convolutional neural networks and long and short-term memory as the primary training model. When using convolutional neural networks for training, we use malware visualization techniques. By converting malware features into images for input, and adjusting the input features and input methods, models with higher classification accuracy will be found; in long-term and short-term memory models, appropriate features and preprocessing methods are used to find Model with high classification accuracy. Finally, the accuracy of small sample training is optimized by generating features for network output samples. In the above training, all of us want to use malware as a sample that affects different devices. In this article, we propose three research topics: 1). When importing images, high-precision models are used to study malware. 2). When importing non-images, a high-precision model will be used to study the malware. 3). By using this model, the generated adversarial network is optimized for small sample malware detection.

Keywords: Android Malware; Deep Learning; Machine Learning; Malware; Ransomware Detection

1 Introduction

Today, malware is still one of the significant cybersecurity threats [2,10]. According to Symantec's 2018 Cybersecurity Threat Report [23]: In terms of traditional malware that affects computers, the total number of malware variants discovered in 2017 was as high as 669 million, an increase of 87.7% from the amount found in 2016; In terms of malware that affects mobile devices such as cell phones, the number of new variants has increased from 17,000 to 27,000, an increase of about 55%. Due to the advancement and popularization of IoT technology, the number of malware that affects IoT devices has increased significantly in recent years. The report mentions that malware variants affecting IoT devices have grown by about 600% in recent years. It is the main threat to the currently accessible IoT device environment.

Common malware currently includes Kotver Trojans, worms, ransomware, spyware, and Coinminer. Most malware infection methods are that attackers use system and software security vulnerabilities to implant malware into the victim's device or trick the victim into downloading a file containing malware, and then implant the malware into the victim's computer. Different malware can have different effects on infected devices. Usually, it may affect the regular operation of the device, damage user files, steal user's private information, steal user rights, and perform unauthorized activities on the device. Most ransomware encrypts users' files and requires a ransom to unlock files, which poses a severe threat to users' data and property. Besides, due to the prevalence of virtual currencies such as Bitcoin, malware that secretly uses the computing power of the victim's computer for mining operations is also one of the widespread malware in recent years.

Traditionally, malware detection methods can be roughly divided into static analysis and dynamic analysis. Static analysis can be realized by byte sequence analysis, control flow chart and operation code frequency distribution to achieve malware identification; Currently, dynamic analysis is based on the use of appropriate monitoring tools for API call monitoring and program behavior monitoring in controlled environments (such as sandboxes or virtual machines). Due to the development of artificial intelligence technology, the identification of malware has excellent potential. At present, many studies aiming at this aspect aim to improve accuracy and efficiency. Currently, most artificial intelligence technologies are used to improve the detection of static analysis.

In order to achieve the purpose of detecting and distinguishing malware, many methods have been proposed in different studies. In 2011, Nataraj *et al.* proposed a method for classifying malware after visualization is proposed [17]; In 2011, Santos *et al.* A method for detecting malware using the opcode sequence of malware is proposed. In 2013, based on the 2011 proposed method and the method of tracking the behavior of malware after execution, combined static analysis and dynamic analysis to detect malware [20, 21]; In 2014, Zolotukhin *et al.* proposed to use the n-gram method to find the essential features in the opcode sequence, and then use support vector machines for training to detect malware [26]. In previous studies, only a small amount of opcode extraction can reduce the performance overhead. However, when this method is used for a small number of training sets, accuracy may be severely affected. Therefore, Zhang *et al.* proposed a method to convert Opcode sequences to images in 2017 was introduced to improve this problem [25]; In the same year, Kwon *et al.* proposed to use API call patterns to identify the type of malware [15]; In 2018, Ni *et al.* proposed to use the SimHash method for hashing and converting it into an image as a sample for training [18]; Kim *et al.* proposed a method for detecting Android malware using multi-feature input as training is proposed [14]; For malware affecting the Internet of Things, Hamed *et al.* uses LSTM as training to detect malware that affects IoT devices in 2018 [8]; Sajad *et al.* digitized ransomware running records, and then use CNN and LSTM to train a classifier to achieve ransomware recognition [9].

The following evaluation criteria are commonly used to evaluate the effectiveness of this research topic:

1) Accuracy, recall rate, precision, and F-measure: These indicators are used to analyze the results of machine learning. Table 1 shows the calculation method for each parameter.

   Accuracy: The proportion of correct samples classified for the classifier to the total number of samples:

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

   This result indicates the classifier's ability to distinguish the entire sample set. However, in some cases, this indicator will fail. For example, if there are 10,000 A samples and 100 B samples in the data set, and the classifier will judge all samples as A, the accuracy is still 99%.

   Recall: The proportion of positive samples correctly classified by the classifier among all positive samples:

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

   Precision: The proportion of all classified samples classified as classified samples by the classifier:

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

   The result indicates the actual accuracy of predicting positive samples.

   F-measure: If the Recall is as important as Precision in the model, the F-measure can be used as an indicator. The calculation also considers two indicators: Precision and Recall:

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

2) Operating cost: Compare the time spent training and distinguishing samples with the required operating resources.

Table 1: The evaluation criteria

	Actual True	Actual False
Prediction True	True Positive (TP)	False Positive (FP)
Prediction False	False Negative (FN)	True Negative (TN)

2 Research on Malware Input by Image

2.1 Motivations

Many types of machine learning models can identify malware. At present, the performance is the best, and the most popular are various neural network models. Among them, a convolutional neural network (CNN) can be said to be one of the representatives. CNN is very suitable for the recognition of image processing. When identifying malware, if the training samples are input as images, CNN can achieve a good classification effect.

Nataraj *et al.* proposed the use of image processing technology to visualize and classify malware in 2011 [17]. This method converts binary malware files into grayscale images for classification. Kancherla *et al.* proposed in 2013 to convert malware executable files into grayscale images called byteplot. And use a Support Vector Machine (SVM) as a training tool [11]. The method proposed by Zhang *et al.* in 2017 decompiled the binary executable file and converted the Opcode sequence into an image. Then identify whether the file is malware [25]. Ni *et al.* proposed to use the SimHash method in 2018 to hash decompiled malware for sequence similarity comparison. Then use CNN for training to achieve the effect of distinguishing different types of malware [18].

From the above research, we can see that after visualizing the malware and then analyzing it, a lot of research has been done in this area. But the methods are different, so we think it is still possible to find higher accuracy and efficiency in visualizing and analyzing malware. In addition to the above research, we also need to focus on converting different data from malware and then performing machine learning, different preprocessing of image input, and the impact on training. In the first research topic, we propose to use several different feature input methods to convert to images, and then try to use different preprocessing methods to process the images. Finally, the best feature selection and preprocessing methods in the experiment are obtained.

2.2 Related Works

The malware is based on images as training samples. After the malware is visualized, it can be processed for related research to identify the malware. Nataraj *et al.* proposed a method for visualizing malware, and performing automatic classification in 2011 is proposed [17]. Figure 1 is a schematic of the study. Since it was observed that the malware of the same family would have similarities in the texture and layout of the image, the study first converted binary malware files into 8-bit vectors. Then convert the 8-bit vector into a grayscale image. Finally, the k-nearest neighbor of Euclidean distance is used for classification. The 9458 sample classifications have 98% classification accuracy among its 25 categories.

The method proposed by Zhang et al. in 2017 decompiled the binary executable file into an Opcode sequence. After converting it into an image, a convolutional neural network was used to compare the target image with the image generated by known malware. Then determine whether the file is malware [25]. The flow chart of its method is shown in Figure 2.

First, after decompressing the unknown file, find out its opcode sequence and frequency of occurrence. Next, use "Information Gain" to find out which function to use in training. After the selected function is converted to an image, training will be conducted to achieve the effect of identifying malware. In this study, the data set used included ten types of malware, with a total of 9168 samples. The classification accuracy of the research results is about 93.7% 96.7%.

In 2018, Ni *et al.* proposed an MCSC (Malware Classification using SimHash and CNN) method [18]. Use the SimHash method to hash the decompiled malware to achieve the effect that similar functions can hash similar sequences. After hashing, the results are converted into grayscale images, which are then trained using CNN to classify the malware. Figure 3 shows the research structure.

This method first decompiles the malware file into Opcode and then executes SimHash. The calculation method of SimHash is shown in Figure 4.

The results obtained by SimHash will be converted into grayscale images and used as samples for CNN training. The method proposed in this study maintains a stable classification accuracy rate for relatively few malware categories in the sample, with an average accuracy rate of 98.86%.

2.3 Malware Detection Based on the High-precision Model for Image Input

This research topic, malware implemented with the high-precision model during image input, focuses on models that can obtain the highest accuracy and performance when using malware as training input. The research architecture is shown in Figure 5.

This research topic will use different methods to deal with the steps of converting samples into images. Evaluate the impact of selecting the appropriate function or directly converting to an image on accuracy. Then, after converting to an image, explore whether different image processing methods can improve accuracy. Finally, adjust the training model to obtain a classifier that can be accurately classified.

This research topic will collect malware samples that affect different devices—for example, traditional computers used for research, mobile phones, Internet of Things devices, etc. Then, when mirroring the malware, we will use different methods to deal with malware. One is to extract features from malware and then convert the features into images. The extraction is mainly based on opcodes; the other is to image the malware directly. After processing the image, the image is used as a training sample.

Convolutional neural network (CNN) is the most widely used deep learning technique in image processing [1]. We will input the samples obtained in the previous step to CNN for training. CNN is roughly composed of a convolutional layer, pooling layer, and fully connected layer, as shown in Figure 6.

C1 and C2 in Figure 6 are convolutional layers. The convolution layer consists of convolution units. The input malware image and function detector (filter) are con-

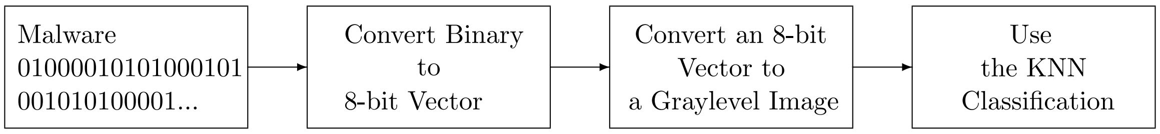

Malware
010000101010001010010101000001...
Convert Binary to 8-bit Vector
Convert an 8-bit Vector to a Graylevel Image
Use the KNN Classification

Figure 1: Method for visualizing and automatically classifying malware [17]

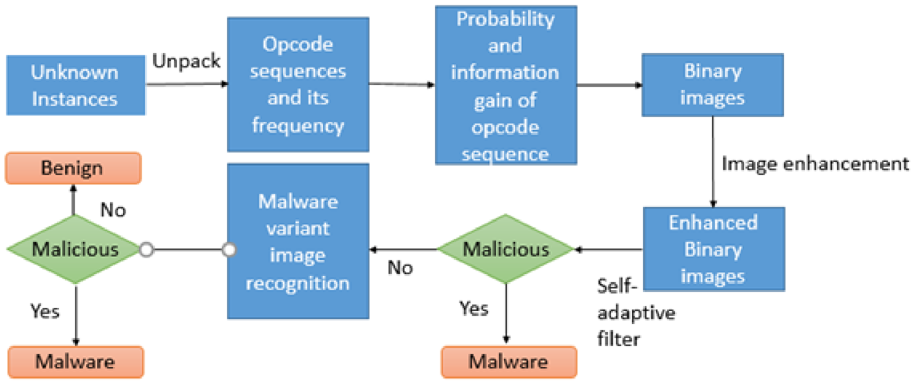


Figure 2: Flow chart of Zhang et al.'s method [25]

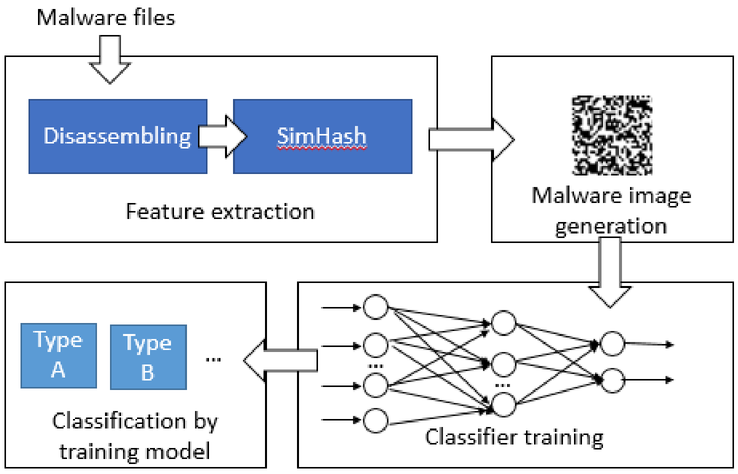


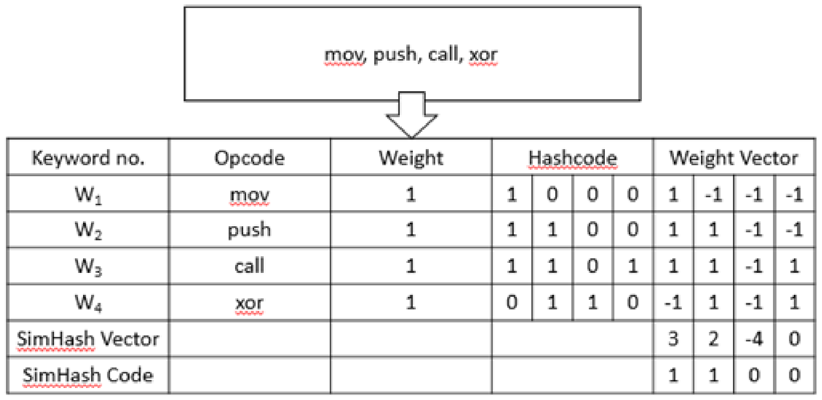
Figure 3: MCSC architecture

volved. Adding the appropriate excitation function (such as ReLU), we get the resulting output. The convolution operation can extract input features and repeatedly remove high-level, sophisticated features from low-level features such as lines in a multi-layer network.

The pooling layer will process the output of the convolutional layer. The pooling layer can be seen as a subsampling process. Taking the maximum pool as an example, if the maximum pool is 2 × 2, the output will be the maximum value in each 2 × 2 block. The data size, after processing by the pooling layer, will become smaller. Therefore, the number of parameters will also be reduced, which helps reduce the phenomenon of overfitting.

Figure 7 is an example of convolution operation and maximum pooling. The yellow box in the figure is the convolution operation of the 3 × 3 area and the feature detector; The red block is the largest pool in the 2 × 2 area.

Finally, the fully connected layer flattens the previous output and connects it to the neural network, and obtains the output after passing through the neural network. In this research topic, we first use the traditional CNN model. The training affected malware samples from different devices. After finding a better image input and processing method, adjust the CNN model to obtain better accuracy and performance.



mov, push, call, xor

Keyword no.	Opcode	Weight	Hashcode				Weight Vector			
W_1	mov	1	1	0	0	0	1	-1	-1	-1
W_2	push	1	1	1	0	0	1	1	-1	-1
W_3	call	1	1	1	0	1	1	1	-1	1
W_4	xor	1	0	1	1	0	-1	1	-1	1
SimHash Vector							3	2	-4	0
SimHash Code							1	1	0	0

Figure 4: Example of SimHash method

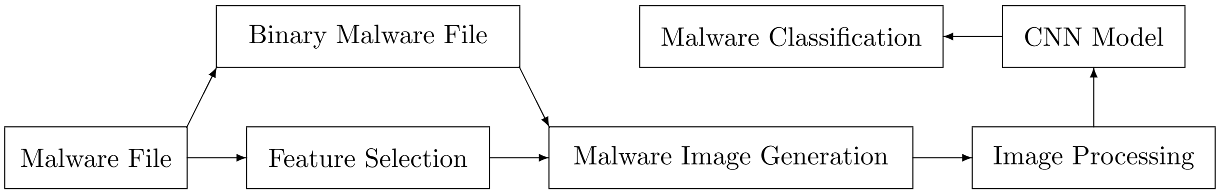


Figure 5: A research framework for the malware detection based on the high-precision model for image input

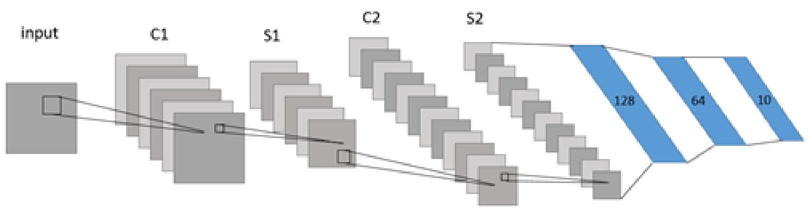


Figure 6: Example of CNN architecture

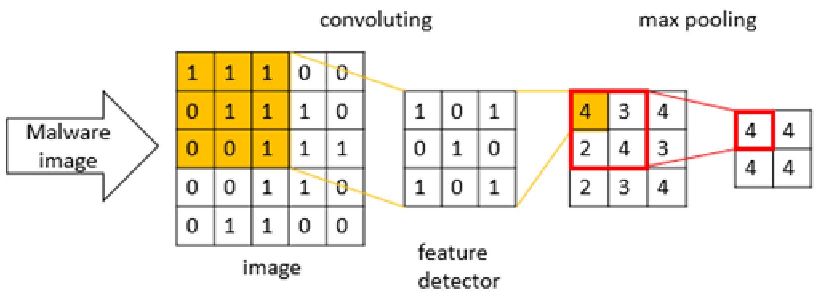


Figure 7: Convolution operation and maximum pooling

There are currently many sample sets open to the outside world. However, new malware samples that have recently appeared may be missing. There are two leading solutions: One is to cooperate with the malware collection platform to obtain fresh samples. The second is to establish a platform and encourage users to upload malware samples. Several malware data sets are currently open to the public and are expected to be used in research:

1) Microsoft Malware Classification Challenge [16];

2) The ultimate gaming malware research benchmark [4];

3) Android malware data set [24];

4) AndroZoo [3];

5) CTU-13 [22].

3 Research on Malware Input by Non-image

3.1 Motivations

In addition to using image input for machine learning of malware samples, directly using opcodes or character string sequences as training inputs is also one of the main methods. However, just as the input image is used as the training sample, the selection of the input features of the sample, the preprocessing of the data, and the adjustment of the model all affect the accuracy and efficiency of the classification.

Kim *et al.* proposed a multi-mode deep learning method using multiple types of features to detect Android malware. The method uses information obtained from decompiled apk files to extract various kinds of features—for example, strings, permissions, and calls API, and so on. Then use the Multimodal Neural Network (MNN) for training. Finally, it is used to determine whether the input file is benign software or malware [14]. Hamed et al. used RNN (Recurrent Neural Network) and LSTM (Long Short Term Memory) in 2018 to take Opcode of IoT malware as input to train a model that can be judged to be benign or malware [8]. Sajad et al. used to record and digitize the actions of the ransomware when it was running, and then used LSTM and CNN to train separately to classify benignly and ransomware and their types [9].

The subject of this research is to test different types of samples (traditional computer malware, mobile malware, etc.) based on their selectable characteristics. Find the combination of the best accuracy and efficiency, and compare it with the best performing images obtained in the previous stage.

3.2 Related Works

The subject of this study uses non-image methods as a sample input. This section will introduce research on different types of malware (such as Android, IoT, and ransomware).

In 2018, Kim *et al.* proposed an Android malware detection framework that uses multi-mode deep learning methods with multiple input features to detect Android malware [14]. Its architecture is shown in Figure 8. First, decompile the apk file to extract various types of functions. This study is divided into seven categories: String functions, method opcode functions, method API functions, shared library function opcode functions, permission functions, component functions, and environment functions. After generating the feature matrix from the features, a multimodal neural network will be used for training. Finally, input files can be distinguished as benign software or malware. The study used two sets of samples, namely 1,075 malware and 19,417 benign software, and 1,209 malware and 1,300 benign software. The model under study achieved 98% accuracy and 0.99 F measurement in the first set of samples. In the second set of samples, 99% accuracy and 0.99 F measurement were obtained.

Due to the rapid increase in malware targeting IoT devices, Hamed *et al.* used LSTM to detect IoT malware [8] in 2018. Figure 9 is its research architecture. After decompressing and decompiling the sample, the opcode is taken out, and then the opcode is selected as the feature to generate the feature vector. Then use RNN and LSTM for training. The study used 281 malware samples and 270 benign software samples. And use three different LSTM configurations for training (Layers 1 to 3). Finally, 100 samples not used in training are used to evaluate the performance of the model. Finally, compare with random forest, support vector machine, KNN, and other classification models. The two-layer LSTM configuration has the highest accuracy. The accuracy rate is 98.18%.

Sajad *et al.* proposed the DRTHIS (Deep ransomware threat hunting and intelligence system) method in 2018 [9]. The research architecture is shown in Figure 10. It is roughly divided into three parts: data conversion, threat detection (whether detection is ransomware), and what kind of ransomware is detected. This method records and digitizes the motion information within 10 seconds after the file is executed. The digitized data will be merged with the label into a training data set. Then through the two models of CNN and LSTM, the binary classifier and the ransomware classifier are trained. Finally, a system capable of judging benign and malicious software and distinguishing the types of ransomware is obtained. In this study, the LSTM model obtained relatively good results. In the experiment, three different types of ransomware samples (Locky, Cerber, TeslaCrypt) were used in the experiment. Each type of ransomware used 220 and 219 benign samples for training. The result of the F-measure is 0.96, and the true positive rate is 97.2%. Also, the study used other types of ransomware not used for training. 99% of CryptWall samples, 75% of TorrentLocker samples, and 92% of Sage samples were correctly classified.

3.3 Malware Detection Based on the High-precision Model for Non-image Input

For this research topic, we focus on research when non-images are used as a sample input. The research architecture is shown in Figure 11.

The input samples are adjusted according to different feature selection methods and combinations. At the same time, there are many variations of LSTM. For example, GRU (Gated Cycling Unit), *etc.* The accuracy and performance of the image are also worth discussing.

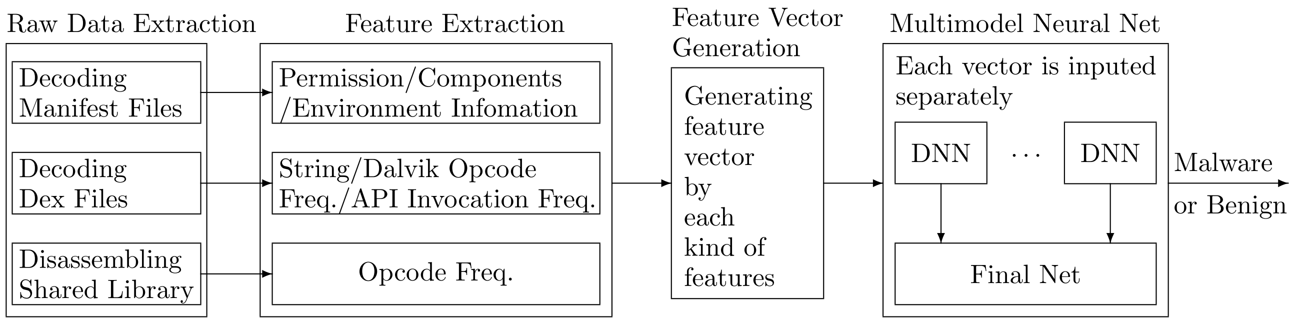

Raw Data Extraction
Decoding Manifest Files
Decoding Dex Files
Disassembling Shared Library
Feature Extraction
Permission/Components /Environment Infomation
String/Dalvik Opcode Freq./API Invocation Freq.
Opcode Freq.
Feature Vector Generation
Generating feature vector by each kind of features
Multimodel Neural Net
Each vector is inputed separately
DNN
· · ·
DNN
Final Net
Malware or Benign

Figure 8: The framework of Kim *et al.* [14]

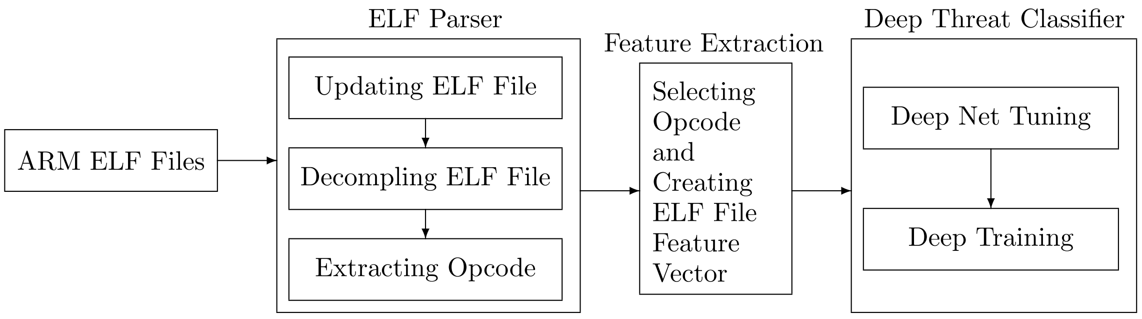


Figure 9: Using LSTM to detect IoT malware [8]

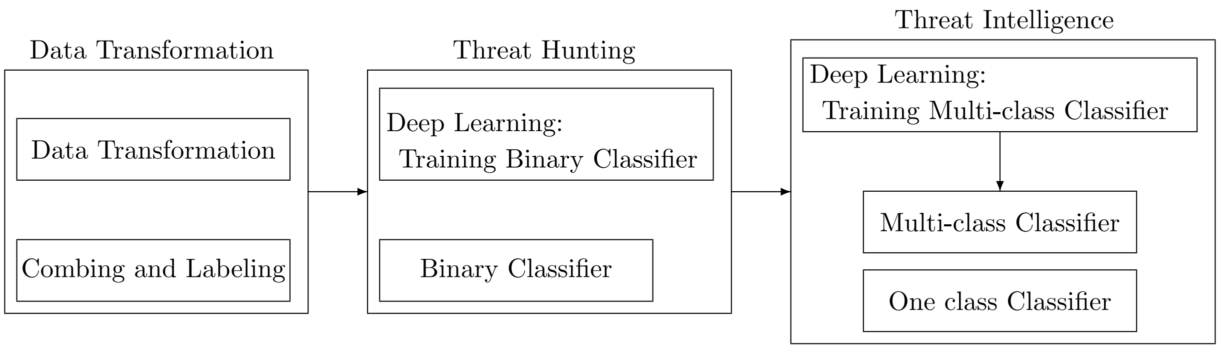


Figure 10: DRTHIS architecture [9]

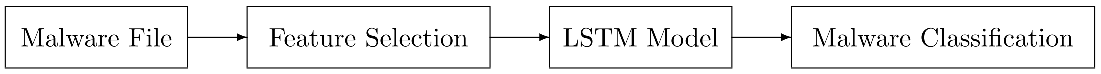


Figure 11: Research structure for the malware implemented with the high-precision model for non-image input

First, we select the features of malware samples based on their category characteristics. For example, malware that affects mobile phones will have functions such as permission requirements, which can also be used as features. At this stage, we use LSTM and its deformation as the training model.

When dealing with non-image input, especially the input of sequence data, LSTM will be a very suitable model. Since the traditional RNN will have a connection phenomenon between the next node and the previous node (The vanishing gradient problem for RNNs), the weight of the self-loop can be changed through the input gate, output gate, and forget gate:

- The function of the input gate is to determine whether to add the current input to the long-term memory.
- The function of the output gate is to determine whether to add the current input to the output.
- The function of the forget gate is to determine whether the incoming information of the upper layer should be kept in memory or forgotten.

Therefore, when the model parameters are fixed, the problem of gradient disappearance or expansion can be avoided. Figure 12 shows the architecture of the LSTM model.

There are many variations of LSTM. One of them is GRU (Gated Circulation Unit). GRU merges the input gate and forgets the gate in LSTM into one update gate. Compared with LSTM, GRU has the advantage of simplifying the calculation. If the prediction performance used is similar to LSTM, it is possible to improve performance.

4 Research on Malware and Generative Adversarial Networks

4.1 Motivations

In the current research, the results of machine learning are related to the number of samples. However, in the face of new malware threats, you must limit the number of samples in a short period. The Generative Adversarial Network (GAN) that have emerged in recent years has considerable potential in this regard [6, 7, 12]. GAN is composed of the Generative model and the Discriminative model. Generative model random samples of the network as input, and the output should be as close as possible to the real samples in the training and deception network; The discriminative model identifies the real samples of the network or generates the output of the system and distinguishes the non-real samples as much as possible. These two networks face each other and adjust the parameters, which ultimately enables the generating network to generate samples, thus making the discriminating network unable to identify authenticity.

Most GANs are used to process image samples. When this method is applied to the classification of malware, we convert the samples into image input methods and study the accuracy of classifying a small number of samples using GANs. In addition, there are currently several variants of GANs. Improving the accuracy or efficiency of malware classification is also one of the research directions.

4.2 Related Works

In 2017, Kim et al. proposed the use of generative adversarial networks for malware classification [13]. In 2018, the research continued and discussed the protection measures for zero-day attacks [12]. Figure 13 is its research architecture.

Traditional malware detection mechanisms usually rely on existing functions. The defense effect against zero-day attacks is limited. Therefore, the method proposed in this study first visualizes the malware and then uses the generative adversarial network (GAN) for training. Pseudo samples, the GAN generates pseudo samples and adjusts the parameters through continuous confrontation with the discriminant network. Finally, the generated fake samples are similar to the actual samples, but not the same. It can mimic a variant of the virus.

Part of the detector uses an autoencoder to learn the features of the malware and feed it back to the generation network to train the generator stably. The classification accuracy rate is 95.74%. At the same time, in the experiments of this study, noise-added sample images were used to simulate virus variants. The SSIM (Structural Similarity Index) of the sample after adding noise and the original sample is $0.6 \sim 0.69$. The accuracy rate is between 98.16% and 98.99%, which indicates that it also has good detection ability for the new variant virus.

4.3 Malware Detection Based on GAN for Generating Small-sample Malware

For this research topic, we used the best performing image input and processing methods in the first research topic to process the samples to be used at this stage. Then use GAN for training to improve the training effect of a small number of samples. It can be expected that when new types of malware appear, they can be effectively detected even with a small number of samples.

The research architecture at this stage is shown in Figure 14. In this research stage, we used a malware data set with a small number of samples and extracted a certain amount of samples from the previous samples as samples at this stage. Then use GAN as a model for this training phase.

The concept of GAN was proposed by Goodfello *et al.* in 2014 [7]. Figure 15 shows the underlying architecture of the GAN.

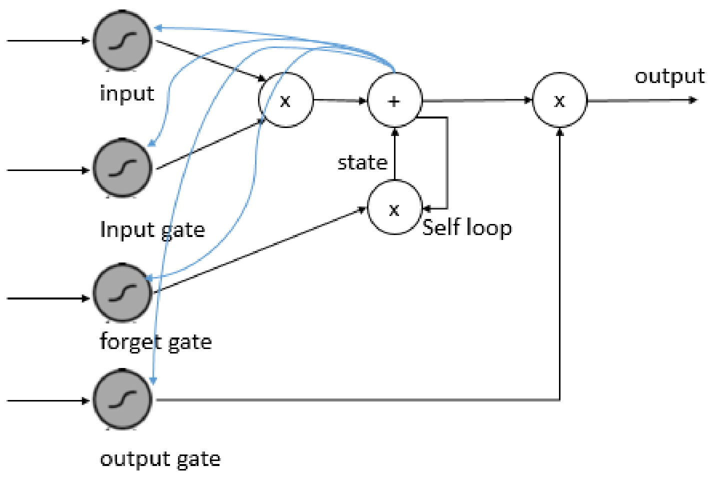


Figure 12: LSTM model

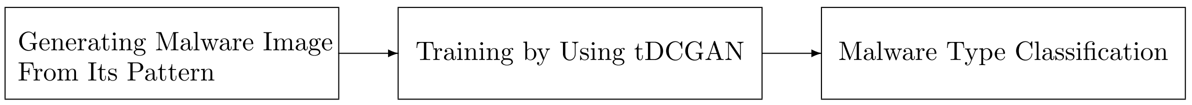

Generating Malware Image From Its Pattern
Training by Using tDCGAN
Malware Type Classification

Figure 13: The research framework of Kim *et al.* [12]

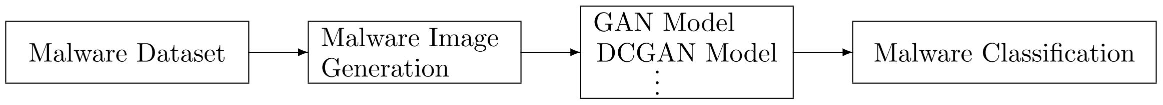


Figure 14: Research structure of malware detection based on GAN for generating small-sample malware

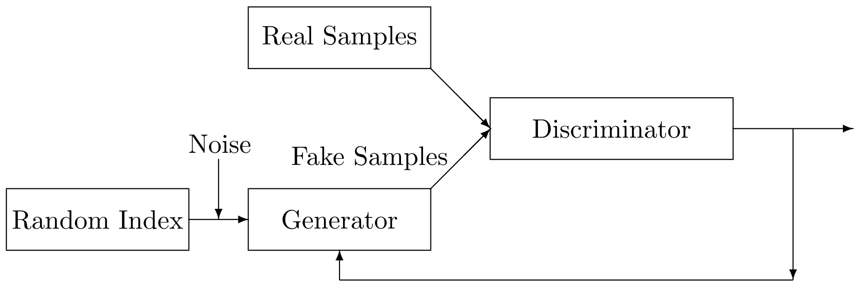


Figure 15: Generating an adversarial network model

GAN consists of a generating network and a discriminative network. The generating network generates random samples of the network as input and produce an output that can deceive and distinguish the network; The discriminative network inputs real samples or generates network output and distinguishes non-real samples as much as possible. The two networks struggle with each other and are adjusting parameters. The final sample generated by the network is almost real. Using this technology, you can amplify the number of samples analyzed by malware and a small amount of samples, and combine the results of the previous two research topics to optimize the accuracy of malware analysis for a small number of samples. GAN is not suitable for learning discrete data, such as text. Therefore, in this part, the image will still be used as the sample input form.

In addition, GAN has conducted many studies to propose its deformation. For example, DCGAN proposed in [19], WGAN proposed in [5], and so on.

5 Conclusions

In this article, we have proposed three research topics: 1) Research on malware input by image: Malware detection based on the high-precision model for image input; 2) Research on malware input by non-image: Malware detection based on the high-precision model for non-image input; 3) Research on malware and generative adversarial networks: Malware detection based on GAN for generating small-sample malware.

This article collected and used malware samples that affected different devices for training. It contains malware that affects computers, malware, and ransomware that affects mobile phones. Malware that affects different methods has various features. For example, due to the apk file structure of mobile phones, the malware that affects mobile phones has many different characteristics compared to the malware of cellular phones; Another example, the permission functions and component functions mentioned in the related research. The goal of this article is to propose a high-precision model for different types of samples.

Acknowledgments

This research was partially supported by the Ministry of Science and Technology, Taiwan (ROC), under contract no.: MOST 108-2410-H-468-023 and MOST 108-2622-8-468-001-TM1.

References

[1] D. S. Abdul Minaam and E. Amer, "Survey on machine learning techniques: Concepts and algorithms," *International Journal of Electronics and Information Engineering*, vol. 10, no. 1, pp. 34–44, 2019.

[2] A. A. Al-khatib, W. A. Hammood, "Mobile malware and defending systems: Comparison study," *International Journal of Electronics and Information Engineering*, vol. 6, no. 2, pp. 116–123, 2017.

[3] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting millions of android apps for the research community," in *IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR'16)*, 2016.

[4] H. S. Anderson, P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," *arXiv preprint*, arXiv: 1804.04637, 2018.

[5] M. Arjovsky, S. Chintala, L. Bottou, "Wasserstein GAN," *arXiv preprint*, arXiv: 1701.07875, 2017.

[6] J. Bruner, A. Deshpande, *Generative Adversarial Networks for Beginners*, July 8, 2020. (`https://github.com/jonbruner/generative-adversarial-networks/blob/master/gan-notebook.ipynb`)

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirze, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*, vol. 2, pp. 2672–2680, 2014.

[8] H. HaddadPajouh, A. Dehghantanha, R. Khayami, K. K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88-96, 2018.

[9] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K. K. R. Choo, D. E. Newton, "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Generation Computer Systems*, vol. 90, pp. 94-104, 2019.

[10] S. Islam, H. Ali, A. Habib, N. Nobi, M. Alam, and D. Hossain, "Threat minimization by design and deployment of secured networking model," *International Journal of Electronics and Information Engineering*, vol. 8, no. 2, pp. 135–144, 2018.

[11] K. Kancherla, S. Mukkamala, "Image visualization based malware detection," in *IEEE Symposium on Computational Intelligence in Cyber Security (CICS'13)*, pp. 40-44, 2013.

[12] J. Y. Kim, S. J. Bu, S. B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460–461, pp. 83-102, 2018.

[13] J. Y. Kim, S. J. Bu, S. B. Cho, "Malware detection using deep transferred generative adversarial networks," in *International Conference on Neural Information Processing*, pp. 556-564, 2017.

[14] T. Kim, B. Kang, M. Rho, S. Sezerm, E. G. Im, "A multimodal deep learning method for Android malware detection using various features," *IEEE Transcations on Information Foresics and Security*, vol. 14, no. 3, pp. 773-788, 2019.

[15] I. Kwon, E. G. Im, "Extracting the representative API call patterns of malware families using recurrent neural network," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, ACM, pp. 202-207, 2017.

[16] Microsoft, *Microsoft Malware Classification Challenge (BIG 2015)*, July 8, 2020. (`https://www.kaggle.com/c/malware-classification`)

[17] L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proceedings of the Eighth International Symposium on Visualization for Cyber Security*, pp. 311–320, 2011.

[18] S. Ni, Q. Qian, R. Zhang, "Malware identification using visualization images and deep learning," *Computers & Security*, vol. 77, pp. 871-885, 2018.

[19] A. Radford, L. Metz, S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *Under review as a conference paper at ICLR 2016*, 2016. (`https://arxiv.org/pdf/1511.06434.pdf`)

[20] I. Santos, F. Brezo, X. Ugarte-Pedrero, P. G. Bringas, "Opcode sequences as representation of executables for data mining based malware variant detection," *Information Sciences*, vol. 231, no. 9, pp. 64-82, 2011.

[21] I. Santos, J. Devesa, F. Brezo, J. Nieves, "OPEM: A static-dynamic approach for machine learning based malware detection," in *Proceedings of International Conference (CISIS'12)*, pp. 271-280, 2013.

[22] Stratosphereips Lab., *The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background Traffic*, July 8, 2020. (`https://www.stratosphereips.org/datasets-ctu13/`)

[23] Symantec, *Internet Security Threat Report*, vol.23, 2018. (`https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf`)

[24] F. Wei, Y. Li, S. Roy, X. Ou, W. Zhou, "Deep ground truth analysis of current android malware," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, pp. 252-276, 2017.

[25] . J. Zhang, Z. Qin, H. Yin, L. Ou, Y. Hu, "IRMD: Malware variant detection using opcode image recognition," in *IEEE International Conference on Parallel and Distributed Systems (ICPADS'17)*, pp. 1175–1180, 2017.

[26] M. Zolotukhin, T. Hamalainen, "Detection of zero-day malware based on the analysis of opcode sequences," in *Proceedings of The 11th Annual IEEE CCNC Security, Privacy and Content Protection*, pp. 386-391, 2014.

Biography

Li-Chin Huang received the B.S. in computer science from Providence University, Taiwan, in 1993 and M.S. in information management from Chaoyang University of Technology (CYUT), Taichung, Taiwan, in 2001; and the Ph.D. degree in computer and information science from National Chung Hsing University (NCHU), Taiwan in 2001. Her current research interests include information security, cryptography, medical image, data hiding, network, security, big data, and mobile communications.

Chun-Hsien Chang received B.S. in Department of Mechanical Engineering from National Cheng Kung University, Taiwan in 2017; M.S. in Department of Management Information Systems, National Chung Hsing University, Taiwan, in 2019. His current research interests include artificial intelligence and information security.

Min-Shiang Hwang received M.S. in industrial engineering from National Tsing Hua University, Taiwan in 1988, and a Ph.D. degree in computer and information science from National Chiao Tung University, Taiwan, in 1995. He was a professor and Chairman of the Department of Management Information Systems, NCHU, during 2003-2009. He was also a visiting professor at the University of California (UC), Riverside, and UC. Davis (USA) during 2009-2010. He was a distinguished professor of the Department of Management Information Systems, NCHU, during 2007-2011. He obtained the 1997, 1998, 1999, 2000, and 2001 Excellent Research Award of National Science Council (Taiwan). Dr. Hwang was a dean of the College of Computer Science, Asia University (AU), Taichung, Taiwan. He is currently a chair professor with the Department of Computer Science and Information Engineering, AU. His current research interests include information security, electronic commerce, database and data security, cryptography, image compression, and mobile computing. Dr. Hwang has published over 300+ articles on the above research fields in international journals.