Enhance deep learning model for malware detection with a new image representation method

DOI: https://doi.org/10.54654/isj.v1i21.1000

Vo Khuong Linh, Nguyen Viet Hung, Tran Ngoc Anh, Duong Do Nhuan, Dinh Cong Hien

***Abstract*— In recent years, there has been an explosion in the number of new malware created by hackers worldwide. The large number of malware families causes certain difficulties for traditional malware detection methods. One of the recent research directions of interest is the application of artificial intelligence to solve problems. This paper proposes a new method of representing malicious code as an image by arranging highly correlated bytes in close pixels in the image. The current research trains deep learning models on self-built datasets and compare the performance of different image representation methods. Experimental results show that the proposed "serpentine" pixel arrangement method provides better results than other methods.**

***Tóm tắt*— Trong những năm gần đây, phần mềm độc hại (malware) do tin tặc tạo ra có sự bùng nổ về số lượng trên phạm vi toàn cầu. Sự xuất hiện của lượng lớn các biến thể phần mềm độc hại đã gây ra những khó khăn nhất định cho các phương pháp phát hiện phần mềm độc hại truyền thống. Một trong những hướng nghiên cứu được quan tâm gần đây là ứng dụng trí tuệ nhân tạo để giải quyết vấn đề. Bài báo này đề xuất phương pháp mới biểu diễn mã độc dưới dạng ảnh bằng cách sắp xếp các byte có độ tương quan cao về các pixel gần nhau trên ảnh. Bài nghiên cứu hiện tại huấn luyện các mô hình học sâu trên tập dữ liệu tự xây dựng và so sánh hiệu suất của các phương pháp biểu diễn hình ảnh khác nhau. Kết quả thử nghiệm cho thấy phương pháp đề xuất với cách sắp xếp pixel theo "hình rắn" (serpentine) mang lại kết quả tốt hơn các phương pháp khác.**

***Keywords*— malware representation, malware detection, deep learning, convolutional neural network.**

***Từ khóa*— biểu diễn mã độc, phát hiện mã độc, học sâu, mạng nơ-ron tích chập.**

This manuscript is received on May 02, 2023. It is commented on November 24, 2023 and is accepted on December 26, 2023 by the first reviewer. It is commented on March 06, 2024 and is accepted on March 19, 2024 by the second reviewer.

I. INTRODUCTION

In recent times, with the development of artificial intelligence, the explosion in the number of malware strains, as well as their variations, is one of the challenges that cause certain difficulties for traditional malware detection methods in the field of Information Security [13, 14]. Therefore, automatic malware detection is essential. Malware detection based on deep learning is one of the methods that brings positive results and is suitable for current requirements.

Today, deep learning has been applied in many fields, especially with good results in image recognition [5, 15]. Due to the many hidden layers between the input and output layers, deep learning models can extract features and classify data into defined classes. When a file is represented as an image, a trained deep learning model can determine or predict whether it is malware.

Portable executable files (PEs) have an important role in information security as they are typically containers for malware and execute malicious behaviors [1, 3, 7]. This type of file contains executing machine codes used to start programs and applications on the computer. It is an important part of the system and one of the main sources of malware deployment.

There are many approaches to using Deep Learning to detect malicious code. R. Pascanu and his colleagues used the Recurrent Neural Networks [16] for extracting features from

executed instructions and extracting robust, time domain features. In addition, Neural Networks (NN) is also used in many research articles to detect or classify malicious code: G.J. Tesauro, J.O. Kephart and G.B. Sorkin have successfully deployed their Neural Network as a commercial product as part of the IBM AntiVirus software package [5]; Li Deng and his colleagues used NN with random projections to deal with large-scale malware [10]; Wenyi Huang and Jack W.Stokes proposed a new multiple layers NN architecture for dynamic malware classification [26]; Noi, N. H. proposed a novel method based on semi-supervised learning FeaWAD* for Latent Representation of IoT malware [27].

On the other hand, Convolutional Neural Networks with the idea of turning malicious code into image representation, has brought many good results with significantly improved speed and performance in research studies [2, 4, 16-19, 22-23]. The results are almost over 90% accurate. From there, we decided to delve deeper into learning about CNN structure as well as ways to represent malicious code images. Through our research, it has been observed that the utilization of Convolutional Neural Networks for detecting malicious code, represented as image, produces the most effective results.

Within the scope of the article, this research focuses on the new method of pixel arrangement in malware representation as greyscale images. This paper compares and evaluates the experiment's result with other pixel arrangement methods, using some convolutional neural network models that have been studied.

II. Related work

A. Convolutional Neural Network

Convolutional neural network (CNN) is a deep learning architecture designed to process and analyze visual data such as images and videos by automatically learning features between pixels [8]. The main components of CNN include convolutional layer, pooling layer, fully-connected (dense) layer, activation function, and overfitting reduction method through dropout [12]. In particular, convolutional layers help detect and extract features and characteristics of images (edges, textures, patterns…), while pooling layers play the role of reducing the complexity of the computing process [4].

The approach using deep learning in malware detection has yielded many good results due to several advantageous features that CNNs offer:

- Feature Extraction: CNNs are adept at automatically learning hierarchical representations of data.

- Spatial Hierarchical Learning: CNNs can learn spatial hierarchies of features within the data, which is useful for detecting complex patterns in the structure and content of malware.

- Robustness to Variations: CNNs are known for their robustness to variations in data.

- Scalability: CNNs can be scaled to handle large datasets, which is critical in the context of malware detection due to the vast number of malware samples that need to be processed.

- Automated Feature Learning: CNNs can automatically learn relevant features from raw data, eliminating the need for manual feature engineering.

By leveraging these capabilities, researchers can develop robust and effective malware detection systems capable of identifying both known and unknown malware variants. The models proposed in [20-21] and [6] have structures and parameters shown in Table 1 and Table 2, respectiely. Through experiments, this research founds that in the model [20], executing one step requires 100ms, while model [6] requires approximately 900ms in each epoch.

TABLE 1. STRUCTURE AND PARAMETERS OF MODEL 1 (PROPOSED BY TU N.M. [20])

Layer	Maps	Sizes	Core	Activation function
Input	1	64x64	-	-
C1	16	64x64	3x3	ReLU

P1	16	32x32	2x2	-
C2	32	32x32	3x3	ReLU
P2	32	16x16	2x2	-
C3	64	16x16	3x3	ReLU
P3	64	8x8	2x2	-
D	512	-	-	ReLU
Output	2	-	-	Softmax

TABLE 2. STRUCTURE AND PARAMETERS OF MODEL 2 (PROPOSED BY NGOC Q.D. [6])

Layer	**Maps**	**Sizes**	**Core**	**Activation function**
Input	1	32x32		
C1	64	30x30	3x3	ReLU
P1	64	15x15	2x2	-
C2	128	13x13	3x3	ReLU
P2	128	7x7	2x2	-
C3	256	5x5	3x3	ReLU
P3	256	3x3	2x2	-
D1	1024	-	-	ReLU
D2	512	-	-	ReLU
D2	128	-	-	ReLU
Output	2	-	-	Softmax

Most studies, such as [6, 9, 20], use networks with a structure of three interleaved convolutional layers and three pooling layers. Within the scope of this study, the paper will also apply and compare the two models of [6] and [20] on the same self-built dataset to obtain an overall objective view.

B. Image Representation of Malware

Representing malware as images is a technique used in some cases, especially when leveraging CNNs for malware detection.

To implement CNNs in malware detection, it is necessary to represent malware as images. According to the proposal by L. Nataraj and colleagues [9], both benign and malware files are considered binary strings composed of 0s and 1s. Then, those strings can be read as a byte sequence (8 bits) arranged in a two-dimensional array and represented as a grayscale image where each pixel has a value ranging from 0 to 255. Variants of the same malware strain exhibit similar image representations regarding pixel pattern features, while different malware strains will have distinct images, differing from benign files (Figure 1).

The challenge is to determine the optimal image representation method to preserve or enhance the distinctive characteristics of the image. Through observation, there are two main factors that leads to the loss of feature information: pixel arrangement order and the process of determining the size of the original image.

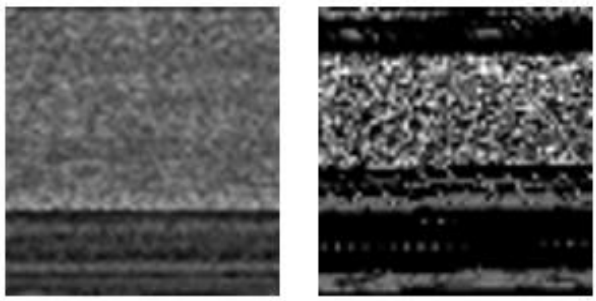

Figure 1. Grayscale Image Representation of Benign (left) and Malware (right)

In most research articles on using CNNs in the problem of detecting malicious code, they often use the method of arranging pixels in a regular order. For example, L.Nataraj [9], Minh Tu et al. [20] used the this method. The regular order for pixel arrangement is the allocation of pixels from left to right and top to bottom of the image.

They established the thresholds depending on the byte sequence length to determine the width size of the image. The width size will be from 32, 64, 128, 256, 354, 512 and 1024. Then, the height will be determined by dividing the sequence length by the width. Most of the time, the output images are usually rectangular in shape (the height is usually longer than the width). The images were then resized to 64x64 dimensions.

This method has a few limitations, including the possibility of information loss. The distance between the last pixel of one line and the first pixel of the next line is quite far apart in two-dimensional space, while in one-dimensional space they are close together. This will increase the risk of distortion and missing feature information in the byte sequence pattern.

On the other hand, after the image transformation, their rectangle image's size would be resized to 64x64 dimensions to fit into the input layer of CNNs. This operation

increased the distortion and loss of information that occurred in the previous stage. From there, the process of feature extraction to distinguish between benign and malware could be badly affected.

In other studies for grayscale images, Quach Danh Ngoc [6] and Ren Z. [17] proposed new approaches that mitigates one limitation of [9, 20]. They used different methods of pixel arrangement such as Gray, Zigzag, H-curve, Hilbert curve, Z-order, and Sweep curve. Only Hilbert and Zigzag methods keep the distance between any two consecutive points on the byte string placed next to each other in 2-D space. after the transformation into an image.

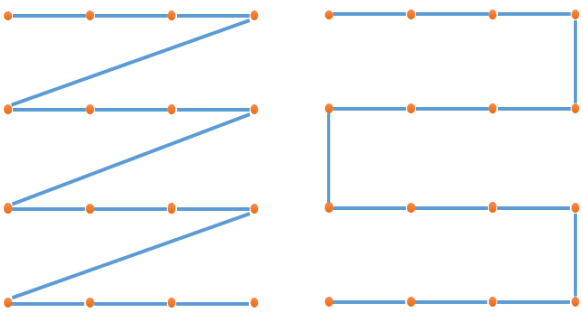

Figure 2. Normal (left) and Zigzag (right) Pixel Arrangements

This is general malware analysis and detection. Particularly with PE files, the characteristic information that helps distinguish malware from benign is mostly concentrated in the header, which is the beginning part of the file's byte sequence.

Beside, Moreina et al. [11] used a diagonal zigzag patterns to detect ransomware while transforming PE header data into color images. This gave good results. However, this study only uses the Header information rather the whole file to detect ransomware. It is possible that the remaining parts of the data still contain characteristic information to detect not only ransomware but also malware of other types.

Therefore, with the zigzag order of pixel arrangement method, the header information of the PE file will be extended horizontally within some top rows of the image. And the Hilbert curve method has difficulty converting to custom sized images. Further more, with the rectangle size determining process, the operation of resizing the image to 64x64 (square) size will increase the risk of information distortion.

In this manner, this paper proposes a new approach mentioned in Section 3 to limit the above problems in PE malware detection. There is a zigzag pixel arrangement method based on the image's diagonal called "serpentine" order and the selection of the appropriate image size to minimize the loss or distortion of the file information.

III. Propasal for PE format malware detection using CNN model

As mentioned in the previous section, with the image representation method of arranging pixels horizontally, the continuity and relative position between bytes of information close to each other in one-dimensional space will be lost when converted to 2D space.

In addition, especially for PE files, the information in the Header section contains most of the information that is considered as features for malware and benign classification [1]. Therefore, if the image is converted to pixels in normal or zigzag order [6], the first part of the sequence contains this kind of information. So, the substructures of this information will extend horizontally and be positioned on the top rows of the image. Since then, after the size-conversion to 64 x 64, the structure may be disturbed, resulting in the loss or omission of important feature information. In addition, for a malicious code in the form of an executable file (PE), most of the characteristics of the malicious code will be in the header, so when converted into a byte string, they will be at the beginning of the string.

Through observation and evaluation, this paper proposes a method to arrange zigzag pixels along the diagonal of the image, the so-called snake line (serpentine order, Figure 3), which promises to bring positive results. This approach ensures that pixels that are close together in one-dimensional space will be close together in two-dimensional space, and those pixels in the file header will be concentrated in one area of the image, thereby highlighting the

information used to distinguish between malware and benign files.

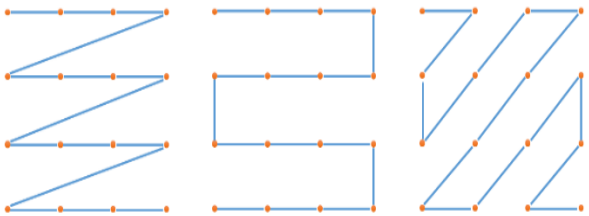

Figure 3. Normal (left), Zigzag (middle) and Serpentine (right) Pixel Arrangements

Next, to overcome the problem of data loss coming from the pixel deviation when converting the rectangular-shaped image into a 64x64 square, this paper proposes an approach: create an image with size: width = height = 1 + square root of the image's byte string length.

Figure 4. Malware normal (left), zigzag (middle) and serpentine (right) pixel arrangements

So that, after the conversion process, the result obtains a set of square images of different sizes. This effort is not just to try to minimize the generation of meaningless pixels (mostly in the last row of the image) but also to try to reduce the deviation between pixels of the converting process.

IV. Experiments and Evaluation

A. Dataset Description

The raw dataset used was the dataset collected and built by the authors, as shown in Table 3 below:

16756 samples of benign files are collected, all of which are files with the PE format or with PE file extensions belonging to the PE group. They are collected on April 30, 2023 from the "C:\Windows" folder of a computer with the Windows 10 Pro operating system, version 22H2. They are divided into 02 sets, which are used for training and testing, at a ratio of 7:3 using the split folders library with the "2501" value as the seed parameter.

Malware files are collected from the VirusShare [24] channel, which came from the VirusShare_00468.zip set, with a total size of 52.63 GB, containing 27533 malware files in PE format, was uploaded on April 30, 2023. These samples are then uploaded to VirusTotal [25] to scan and retrieve the results to classify malware as well as serve as a basis for separating the training and testing sets. VirusTotal used a total of 83 anti-malware engines, among which the ClamAV engine is used (a not too powerful engine) to build a test set. Based on the scan results, samples are extracted and separated into the following datasets:

- The "known" set: These are the samples that most engines (including ClamAV) detect and identify as malicious. Similar to the benign files, they are separated into 02 sets for training and testing with the ratio of 7:3 by using the split folder library with a "2501" value as the seed parameter.

- The "unknown-clamav" set: These are the samples that the ClamAV engine did not detect and identify as malicious. They are identified as the test set. Since the number is large, so only 3,000 samples are chosen for this set.

- The "unknown" set: These are the samples that VirusShare has confirmed as malicious, but none of VirusTotal's engines have detected and identified them as malicious. This is identified as a test set with high difficulty.

Table 3. Number of samples collected in each set

Label	Train	Test	Total
benign	11722	5034	16756
malware-known	10488	4498	14986
malware-unknown-clamav	0	3000	3000
unknown	0	136	136

Then, this paper combines 03 pixel arrangement methods (normal, zigzag, serpentine) along with 02 ways to determine the shape and size of the image (rectangle, square).

This research builds 06 image datasets, each dataset includes 06 image sets: 02 sets are used to train (benign and known-malware) and 04 sets are used to test (benign, known-malware, unknown-clamav-malware, unknown-malware).

B. Experimental results

The experiment program is built in Python 3 language with Keras library, running on a computer with the following configuration:

- Intel Core i7-6700HQ CPU @2.60GHz
- RAM: 12GB.
- NVIDIA GeForce GTX 960M.

Parameters are set in the training process: all the random seed values are 2501, the ratio of 8:2 is used to separate the actual training and validation sets and the number of epochs is 50.

Due to the equal number of malware and benign in the training dataset, the "accuracy" measurement is used. In addition, this paper also uses "recall" and "f1-score" measurements, which are respectively expressed in the following formulas:

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

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

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

$$f\_score = \frac{2 \times preision \times recall}{preision + recall}$$

With TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative, which are the predicted labels from a trained model.

Therefore, with 4 test sets, 3 representation methods, 2 image size and shape conversion methods, this research builds 24 test sets for the 2 models mentioned above.

TABLE 4. EXPERIMENTAL RESULTS

Pattern	Test Set	Original Shape	Model 1			Model 2		
			Accuracy	Recall	F1	Accuracy	Recall	F1
Normal	benign	rectangle	97.42 ± 0.16	97.42	98.69	**97.83 ± 0.15**	**97.83**	**98.91**
		square	**98.25 ± 0.13**	**98.25**	**99.12**	97.48 ± 0.16	97.48	98.72
	known	rectangle	**95.15 ± 0.21**	**95.15**	**97.52**	**93.26 ± 0.25**	**93.26**	**96.51**
		square	94.06 ± 0.23	94.06	96.94	93.13 ± 0.25	93.13	96.44
	unknown-clamav	rectangle	**88.33 ± 0.32**	**88.33**	**93.81**	80.03 ± 0.40	80.03	88.91
		square	85.36 ± 0.35	85.36	92.11	**83.03 ± 0.38**	**83.03**	**90.73**
	unknown	rectangle	**77.94 ± 0.41**	**77.94**	**87.6**	**76.47 ± 0.42**	**76.47**	**86.67**
		square	75.00 ± 0.43	75	85.71	73.53 ± 0.44	73.53	84.75
Zigzag	benign	rectangle	95.19 ± 0.21	95.19	97.54	96.82 ± 0.18	96.82	98.39
		square	**96.86 ± 0.17**	**96.86**	**98.41**	**98.05 ± 0.13**	**98.05**	**99.02**
	known	rectangle	92.17 ± 0.27	92.17	95.93	**93.53 ± 0.25**	**93.53**	**96.66**
		square	**94.33 ± 0.23**	**94.33**	**97.09**	92.22 ± 0.27	92.22	95.95
	unknown-clamav	rectangle	81.40 ± 0.39	81.4	89.75	**84.27 ± 0.36**	**84.27**	**91.46**
		square	**85.30 ± 0.35**	**85.3**	**92.07**	72.63 ± 0.45	72.63	84.15
	unknown	rectangle	64.71 ± 0.48	64.71	78.57	**71.32 ± 0.45**	**71.32**	**83.26**
		square	**78.67 ± 0.41**	**78.67**	**88.07**	67.65 ± 0.47	67.65	80.7
Serpentine	benign	rectangle	**98.19 ± 0.13**	**98.19**	**99.09**	**97.89 ± 0.14**	**97.89**	**98.94**
		square	98.05 ± 0.13	98.05	99.02	96.01 ± 0.20	96.01	97.96
	known	rectangle	95.49 ± 0.21	95.49	97.69	93.86 ± 0.24	93.86	96.83
		square	**95.78 ± 0.20**	**95.78**	**97.84**	**96.64 ± 0.18**	**96.64**	**98.29**
	unknown-clamav	rectangle	**78.27 ± 0.41**	**78.27**	**87.81**	79.13 ± 0.41	79.13	88.35
		square	74.97 ± 0.43	74.97	85.69	**79.40 ± 0.40**	**79.4**	**88.52**
	unknown	rectangle	**82.35 ± 0.38**	**82.35**	**90.32**	66.18 ± 0.47	66.18	79.65
		square	78.68 ± 0.41	78.68	88.07	**86.03 ± 0.35**	**86.03**	**92.49**

The experimental results show that the serpentine image representation method always gives higher results than the others, while the square image conversion method gives equivalent or sometimes lower results than the others.

C. Evaluation

According to the experimental results, the serpentine representation method provides higher results than other methods (with Model 1 ranked first in 5/8 test sets and Model 2 coming first in 4/8 test sets). This result shows that this approach brings a higher efficiency.

For the square image conversion method, compared to the rectangular image conversion method, the results are only equivalent (higher in 6/12 test sets when running with Model 1 and higher in 5/12 test sets when running with Model 2). The results show that the rectangular image conversion method is more suitable for CNN models.

Through evaluation results, the built dataset has an increasing level of difficulty when divided into 3 groups of test sets including the "familiar" group (including benign and known-malware set, same group with the training set), the "relatively strange" group (including unknown-clamav-malware, a collection that not as "powerful" engine as ClamAV cannot recognize them as malicious) and the "completely unknown" group (a set of malware that 100% of the engines cannot detect during the scanning at that time).

V. CONCLUSION

This article introduces methods for representing gray images of malware, which use deep learning convolutional neural networks to build models that automatically detect malware directly from input data. Experiments show that the method of representing malware images in the serpentine pixel arrangement method is better than the normal and zigzag methods, as well as showing that the method of determining image shape in a rectangle still gives better results than the square method. The CNN network model requires high hardware configuration for training, so, in the experiment, the samples will be resized to 64x64, and by choosing the serpentine method, the amount of lost or distorted information can be reduced. In the future, when our computer has a better configuration, we will continue to experiment with higher image resolution and apply more complex pixel arrangement methods to evaluate the results.

This research is funded by the project "Develop a support system for spyware detection" under LQDTU Research Programs (Grant 24.KGM.11).

REFERENCES

[1] Anh Tran Ngoc, Linh Vo Khuong, (2021), "Malware detection based on Machine Learning and PE header information", Information Security Journal, Vietnam.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, (2012), "ImageNet Classification with Deep Convolutional Neural Networks", International Conference on Neural Information Processing Systems (NIPS).

[3] Edward Raff, Jared Sylvester, Charles Nicholas, (2017), "Learning the PE Header, Malware Detection with Minimal Domain Knowledge", ACM Workshop on Artificial Intelligence and Security.

[4] Gibert, D, (2016), "Convolutional neural networks for malware classification", University Rovira i Virgili, Tarragona, Spain.

[5] Hironobu Fujiyoshi, Tsubasa Hirakawa, Takayoshi Yamashita, (2019), "Deep learning-based image recognition for autonomous driving", IATSS Research, vol 43, issue 4, pages 244-252.

[6] Hung Nguyen Viet, Ngoc Quach Danh, Dung Pham Ngoc, (2019), "Research on techniques of representing malware files and deep learning models in malware detection", XXII National Conference: Some selected issues of Information and Communication Technology, Thai Binh, Vietnam.

[7] Huu Danh Pham, Tuan Dinh Le, Thanh Nguyen Vu, (2018), "Static PE Malware Detection Using Gradient Boosting Decision Trees Algorithm", International Conference on Future Data and Security Engineering, pp 228-236.

[8] Kephart J.O. Tesauro, G.J., Gregory B Sorkin, (1996), "Neural networks for computer virus

recognition", IEEE International Conference on Intelligence and Security Informatics.

[9] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, (2011), "Malware images: Visualization and automatic classification", Proceedings of the 8th International Symposium on Visualization for Cyber Security.

[10] Li Deng, George E. Dahl, Jack W. Stokes and Dong Yu (2013), "Large-scale malware classification using random projections and neural network", ICASSP.

[11] Moreira, C. C., Moreira, D. C., & de Sales Jr, C. D. S. (2023), "Improving ransomware detection based on portable executable header using xception convolutional neural network", Computers & Security, 130, 103265.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, (2013), "Dropout: A simple way to prevent neural networks from overfitting J. Mach. Learn. Res.". 15(1):1929–1958.

[13] N. Idika, A.P. Mathur, (2007), "A Survey of Malware Detection Techniques", Purdue University.

[14] Rabia Tahir, (2018), "A Study on Malware and Malware Detection Techniques", International Journal of Education and Management, MECS.

[15] Rahul Chauhan, Karmal K. Ghanshala, R.C Joshi, (2018), "Convolutional Neural Network (CNN) for Image Detection and Recognition", First International Conference on Secure Cyber Computing and Communication.

[16] Razvan Pascanu, Jack W. Stokes, Li Deng, Dong Yu, Mady Marinescu, Anil Thomas, (2015), "Malware Classification with Recurrent Networks", IEEE ICASSP.

[17] Ren, Z., Chen, G., & Lu, W. (2020), "Malware visualization methods based on deep convolution neural networks", Multimedia Tools and Applications, 79, 10975-10993.

[18] Sunoh Choi, Sungwook Jang, Youngsoo Kim, Jonghyun Kim, (2017), "Malware Detection using Malware Image and Deep Learning", International Conference on Information and Communication Technology Convergence, Jeju, Korea (South).

[19] P. V. Dinh, N. Shone, P. H. Dung, Q. Shi, N. V. Hung and T. Nguyen Ngoc, "Behaviour-aware Malware Classification: Dynamic Feature Selection," 2019 11th International Conference on Knowledge and Systems Engineering (KSE), Da Nang, Vietnam, 2019, pp. 1-5, doi: 10.1109/KSE.2019.8919491.

[20] Tu Nguyen Minh, Hung Nguyen Viet, Anh Phan Viet, Loi Cao Van, Nathan Shone, "Detecting Malware Based on Dynamic Analysis Techniques Using Deep Graph Learning", Lecture Notes in Computer Science, vol. 12466, 2020.

[21] Nguyen, M.T., Nguyen, V.H. & Shone, N. Using deep graph learning to improve dynamic analysis-based malware detection in PE files. J Comput Virol Hack Tech 20, 153–172 (2024). https://doi.org/10.1007/s11416-023-00505-x.

[22] Seonhee Seok, Howon Kim, (2016), "Visualized Malware Classification Based on Convolutional Network", Journal of The Korea Institute of Information Security and Cryptology.

[23] N. V. Hung, P. Ngoc Dung, T. N. Ngoc, V. Dinh Phai and Q. Shi, "Malware detection based on directed multi-edge dataflow graph representation and convolutional neural network," 2019 11th International Conference on Knowledge and Systems Engineering (KSE), Da Nang, Vietnam, 2019, pp. 1-5, doi: 10.1109/KSE.2019.8919284.

[24] VirusShare.com, https://virusshare.com/.

[25] VirusTotal.com, https://www.virustotal.com/.

[26] Wenyi Huang, Jack W.Stokes, (2016), "MtNet: A Multi-Task Neural Network for Dynamic Malware Classification", DIMVA.

[27] Noi, N. H., & Ngoc, T. N. (2023). Learning Latent Representation with Limited Labels for IoT Anomaly Detection. Journal of Science and Technology on Information Security, 3(20), 14-22. https://doi.org/10.54654/isj.v3i20.986.

ABOUT THE AUTHORS

Vo Khuong Linh

Workplace: Institute of information and communication technology, Le Quy Don Technical University, Master student.

Emai: vokhuonglinh@gmail.com

Education: He received his Engineering degree in Computer Science from Le Quy Don Technical University in 2017.

Recent research direction: Information security, malware detection based on machine learning.

Tên tác giả: **Võ Khương Lĩnh**

Cơ quan công tác: Viện Công nghệ thông tin và truyền thông, Đại học Kỹ thuật Lê Quý Đôn

Email: vokhuonglinh@gmail.com

Quá trình đào tạo: Tốt nghiệp kỹ sư ngành Công nghệ thông tin tại Đại học Kỹ thuật Lê Quý Đôn năm 2017.

Hướng nghiên cứu hiện nay: An toàn thông tin, học máy trong phát hiện mã độc.

Nguyen Viet Hung

Workplace: Institute of information and communication technology, Le Quy Don Technical University.

Email: hungnv@lqdtu.edu.vn *(corresponding author)*

Education: He received his BSc, MSc and PhD degrees in Computer Science from Moscow Institute of Physics and Technology in 2006, 2008 and 2012 respectively.

Recent research detection: Information security; Malware detection; Intrusion detection.

Tên tác giả: **Nguyễn Việt Hùng**

Cơ quan công tác: Viện Công nghệ thông tin và truyền thông, Đại học Kỹ thuật Lê Quý Đôn

Email: hungnv@lqdtu.edu.vn

Quá trình đào tạo: Tốt nghiệp cử nhân, thạc sỹ và tiến sỹ tại trường Đại học Vật lý kỹ thuật Matxcova, Liên bang Nga vào các năm 2006, 2008 và 2012.

Hướng nghiên cứu hiện nay: An toàn thông tin; Phát hiện mã độc; Phát hiện xâm nhập mạng.

Tran Ngoc Anh

Workplace: Center 286, Command 86.

Emai: anhtn69@gmail.com

Education: He received his Engineering, Master degree and PhD in Computer Science from Le Quy Don Technical University in 1994, 2007 and 2016 respectively.

Recent research direction: Information security, NLP.

Tên tác giả: **Trần Ngọc Anh**

Cơ quan công tác: Trung 286, Bộ Tư lệnh 86

Email: anhtn69@gmail.com

Quá trình đào tạo: Tốt nghiệp Kĩ sư, Thạc sỹ và Tiến sỹ tại Đại học Kỹ thuật Lê Quý Đôn, vào các năm 1994, 2007 và 2016

Hướng nghiên cứu hiện nay: An toàn thông tin; xử lý ngôn ngữ tự nhiên.

Duong Do Nhuan

Workplace: Research Institute 486, Command 86.

Emai: nhuandd8864@gmail.com

Education: He received his Engineering degree in Computer Science from Le Quy Don Technical University in 2011 and Master degree in Information system from Posts and Telecommunications Institute of Technology in 2021.

Recent research direction: malware detection, vulnerabilities in source code detection.

Tên tác giả: **Dương Đỗ Nhuận**

Cơ quan công tác: Viện nghiên cứu 486, Bộ tư lệnh 86

Email: huandd8864@gmail.com

Quá trình đào tạo: Tốt nghiệp kỹ sư ngành Công nghệ thông tin tại Đại học Kỹ thuật Lê Quý Đôn năm 2011, Cao học Hệ thống thông tin tại Học viện Bưu chính viễn thông năm 2021.

Hướng nghiên cứu hiện nay: Phát hiện mã độc, phát hiện lỗ hổng bảo mật trong mã nguồn.

Dinh Cong Hien

Workplace: Center 586, Command 86.

Emai: dinhconghien2000@gmail.com

Education: He received his Engineering degrees in Computer Science from Le Quy Don Technical University in 2023.

Recent research direction: Pentest, malware analysis.

Tên tác giả: **Đinh Công Hiển**

Cơ quan công tác: Trung tâm 586, Bộ tư lệnh 86.

Email: dinhconghien2000@gmail.com

Quá trình đào tạo: Tốt nghiệp kỹ sư ngành Công nghệ thông tin tại Đại học Kỹ thuật Lê Quý Đôn năm 2023.

Hướng nghiên cứu hiện nay: Kiểm thử an toàn thông tin, phân tích mã độc.