International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 9, No.2 (Mar-2020)
http://dx.doi.org/10.12785/ijcds/090204

Efficient Algorithm for Malware Classification: n-gram MCSC

Myung-Jae Lim[1], Jae-Ju An[1], So-Hee Jun[1] and Young-Man Kwon[1*]

[1]Department of Medical IT, Eulji University, Seongnam, Korea
**: corresponding author*

Received 27 Sep. 2019, Revised 22 Feb. 2020, Accepted 23 Feb. 2020, Published 01 Mar. 2020

Abstract: In this paper, we proposed n-gram MCSC. This method extracts n-gram opcode from execution file and use Simhash to make image of them. We measured and compared the performance metrics of n-gram MCSC and existing MCSC such as accuracy, loss, precision and AUC value of PR curve and ROC curve. To verify whether the difference of accuracy is significant statistically or not, we made experiments of it thirty times and did the ANOVA analysis. We found it was significant. As the result of post-hoc analysis, n-gram MCSC showed better result than existing MCSC in accuracy. The 2-gram MCSC showed the better result than 3-gram MCSC in terms of accuracy, precision, AUC value of PR curve and ROC curve.

Keywords: Malware classification, Malware detection, Malware visualization, Simhash, N-gram, Deep learning

1. INTRODUCTION

Malware is malicious program designed to disrupt and damage a computer system. This malware has increased significantly in recent years and even caused global problems. Therefore, detecting malware is considered as the critical factor in computer security and has achieved remarkable improvement as the result of various studies. However, it still requires a lot of research as malware continues to be evolved by using technologies that create variants and avoid detection.

For detecting malware, there are two main technologies: static analysis and dynamic analysis [1]. First, static analysis is a method of detecting malware by using the static characteristics of execution file without running it. Next, dynamic analysis is a method of analyzing and detecting malware by using the behavioral characteristics of execution file while running it in a limited environment. Both methods have recently been combined with AI (artificial intelligence) algorithms.

In section 2, we will present related works about the two methods for malware detection, MCSC, n-gram and performance metrics. In section 3, we propose N-gram MCSC system for generating malware image and explain details of the system at each step. In section 4, we explain our experimental setting and result of our experiment. We conclude the performance of proposed method according to experimental results. In section 5, we summarize the proposed system and experimental results.

2. RELATED WORKS

A. Static analysis and Dynamic analysis

To detect malware, we need to know whether file has malicious activity or not. To do this, there are two main approaches for analyzing execution file: static analysis and dynamic analysis.

Static analysis is the method of analyzing executable files without running them to determine whether they are malware or not. This method uses their characteristics such as string signature, byte sequence n-grams and opcode or opcode distribution etc. However, the disadvantage of this method is vulnerable to obfuscation (hiding the original algorithm, data structures or the logic of the code) because it does not run the program.

The dynamic analysis is the method of analyzing behavioral characteristics of execution file while running it. Because a file must be executed, we do it in a limited environment like a virtual environment, simulator, sandbox etc. It monitors behavior like API call or tainting others. As malicious programmers recently use obfuscation to avoid static methods, dynamic analysis is getting interest because it can detect what execution files actually do. However, the disadvantage of this method is time-consuming, and it needs limited environments to do.

E-mail: lk04@eulji.ac.kr, anjaeju@gmail.com, gkgkthgml@gmail.com, ymkwon@eulji.ac.kr

B. MCSC (Malware Classification using Simhash and CNN algorithm)

In classification for detecting malware, there are various methods to apply the method of natural language processing or apply the method of image processing. To utilize image processing, we need the binary execution file must be converted to image.

There are several ways to make the execution file as image. Nataraj [2] mapped the binary code of it to 8-bit vector and convert the 8-bit vector to grayscale image. Han [3] makes image using Simhash and dbj2. He transformed the opcodes from a file into color image matrices by locating with Simhash and giving color with dbj2. Sang Ni [4] makes image by using Simhash and called it MCSC.

MCSC uses Simhash algorithm to make image from execution file. Simhash [5] is a method that represents document as a value, which can be used to check whether two documents are similar or not. This method can reduce time-cost by comparing each value of documents rather than comparing the texts in them one by one. Simhash consists of tokenizing given data, hashing all tokenized data and fiving weight, calculating weighted vectors into a calculated value and converting the calculated value into a binary value. For more details, we give a naïve example in Fig. 1.

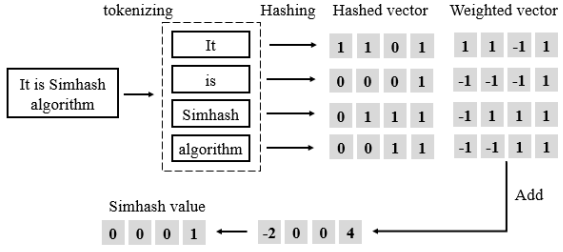

Figure 1. Simhash algorithm

First, tokenizing separates given data into each word. Next, the words are hashed, and it gives us n-bit length of binary value, regardless of the length of data. For hashing, there are various hash method including md-5, sha-128, sha-256 and so on [6]. Hashed n-bit binary values are changed by weight, which can be any number. When weighting, the number 1 changes to 1*w, 0 changes to -1*w. After adding all weighted values, we get a calculated value. Then, converting the value with following rule: The 1 if positive else the 0. Finally, we can get n-bit binary value that represents given data, which can be converted into decimal or hexadecimal.

We displayed MCSC algorithm in Fig.2. This consists of three procedures (steps): feature extraction, making image (visualization) and training composed of feature extraction of neural net and classification. In this system, we use the execution file instead of plain text file. In the window system, the main format of execution file is the PE (portable execution).

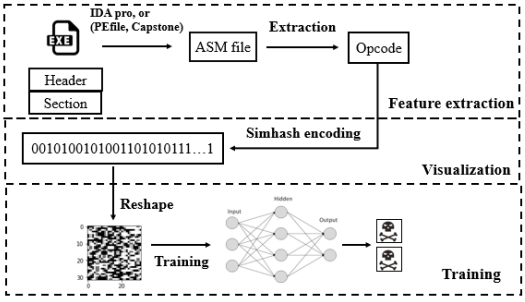

Figure 2. The overall procedure of MCSC

In the step of feature extraction, it disassembles execution file to get PE format by disassembler like IDA pro. In addition, it extracts only opcodes in the PE format. In the step of visualization, the extracted opcodes are encoded by Simhash function to get the fixed length of Simhash binary value. As it has fixed length of binary value regardless of the number of the opcodes, MCSC has advantage in using all information of the opcodes. In the step of training, it converts the fixed length of binary value into X*Y image, where X*Y is depending on the types of Simhash. Classifier uses CNN model, learns features of each malware image and classifies it to classes where it belong.

C. N-gram

N-gram [7] is a method of grouping data into N-size chunks when there is a given data. An n-gram of size 1 is referred to as a "unigram", size 2 is a "bigram" or "digram" and size 3 is a "trigram". N-gram is not only used in natural language processing, but also used effectively malware detection area. N-gram tokenizes given data into several words. Then, it made new word composed of n-size consecutive words to get context information. For example, 2-gram example is shown in Fig. 3.

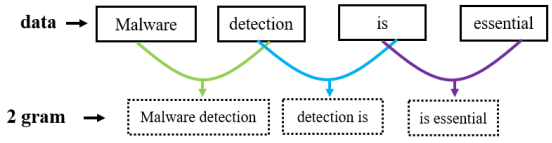

Figure 3. 2-gram example

Igor Santos [8] used a method to get file signatures that are made up of the set of n-grams for every file in author's dataset. It got the best result 74.37% when n-size is 4 and false positive ratio is 0%. In this paper, we apply n-gram to opcode sequence of each file and use the n-gram opcode sequence.

D. Performance metrics

For measuring the performance of classification task, confusion matrix is typical method for this. Confusion matrix has negative and positive predictions on x-axis and negative and positive actual labels on y-axis, we divide each corresponding section as TN, FP, FN, TP. TN (true negative) means classifier predicted negative label for given data, and the actual label of the data was negative. FN (false negative) means classifier predicted negative label for given data, in fact, the actual label of the data was positive. Again, FP (false positive) means that prediction was positive label, but the actual label was negative. TP (true positive) means that prediction was positive label, and the actual label was positive.

		Predicted class	
		Negative	Positive
Actual class	Negative	TN	FP
	Positive	FN	TP

Figure 4. Confusion Matrix

To get the more concise metrics, we use precision and recall. Through confusion metrics, the method of obtaining precision and recall is as follows.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

Equation (1) is precision that is actual positive class rate of total positive predictions. For example, when high precision, if weather forecast predicts tomorrow's weather is sunny, the probability of sunny weather is high. Equation (2) is positive prediction rate of actual positive.

We can draw PR curve [9] with them. PR curve is consisted of recall on x-axis and precision on y-axis. PR curve is very useful when the dataset is highly imbalanced.

ROC curve is drawn True positive rate (TPR) and false positive rate (FPR). TPR is same with recall, and FPR is calculated dividing FP with TN + FP. ROC curve is not accurate in the case of imbalanced dataset. However, it can be used as convenient performance measurement for comparing several models at once.

Area under the curve (AUC) is the bottom area of the PR or ROC curve. The more AUC is close to 1.0, the more classifier classifies well.

3. PROPOSED METHOD

In this paper, we propose the N-gram MCSC algorithm for malware classification based on the existing MCSC algorithm. The proposed system looks like as Fig. 5.

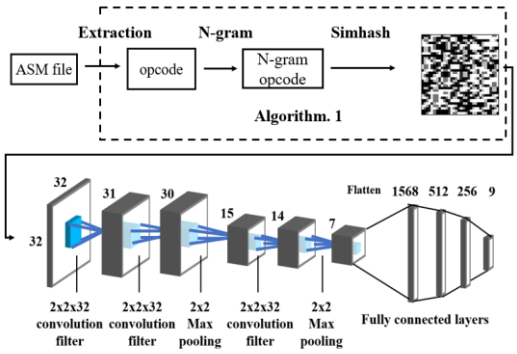

Figure 5. Proposed system: N-gram MCSC

As we know in Fig. 5, it extracts opcodes from ASM files. After that, we make n-gram opcode. It is a method for grouping the opcode sequence into N size chunk sequence using N-gram as explained in Simhash encoding. This is the key idea of our system. The overall procedure of making image from one ASM file is in the following Algorithm. 1 (written by pseudo python code).

Algorithm 1 Making image from ASM file

```
input: ASM file
output: 32 x 32 image
 1. Begin
 2. make opcode_instruction_set
 3. opcode_list = []
 4. for token in ASM file:
 5.     if token in opcode_instruction_set :
 6.         opcode_list.append(token)
 7. intialize n for n-gram
 8. ngram_list = []
 9. list_size = length(opcode_list)
10. for i in range (0, (list_size - n)+1):
11.     out = opcode_list[i]
12.     for j in range (1, n):
13.         out= out + opcode_list[i+j]
14.     ngram_list.append(out)
15. apply Simhash (refer to reference [7])
16. reshape the result with bilinear interpolation
```

In the step of Simhash, we used Simhash function that is in hashlib library in Python. Especially, we used Simhash-768 (SHA-768) that is proposed in MCSC paper because SHA-768 got the best result. The result image of Simhash is 32 x 24, so we used bilinear interpolation in openCV to make them square (32 x 32).

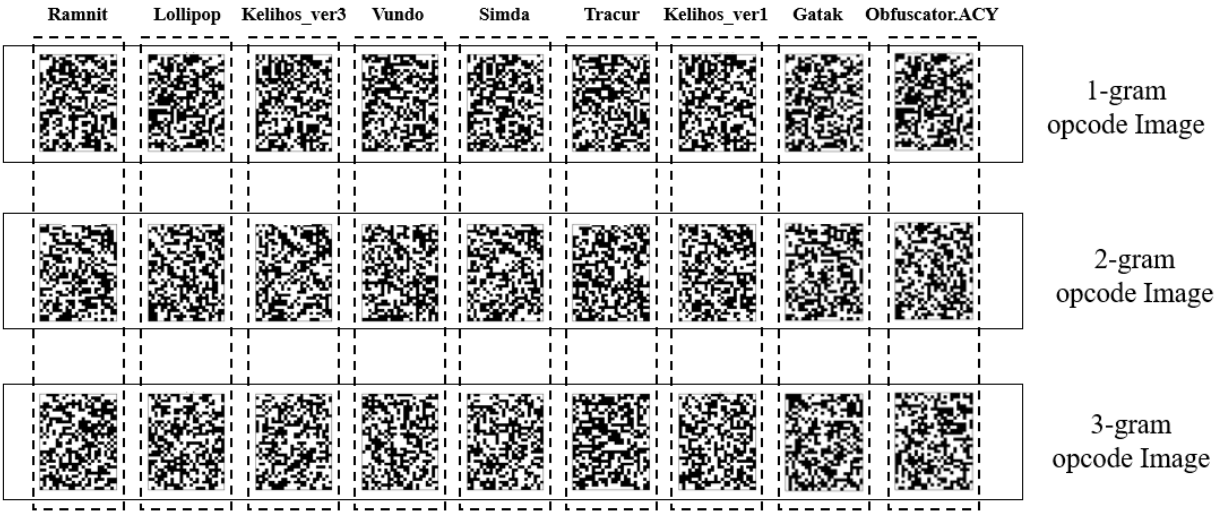

Figure 6. Image of n-gram opcode for malware family

In the final step, neural network of MCSC paper consists of three convolutional layers with tangent hyperbolic activation function and two Maxpoolling layers with dropout layer. After that, it flattens the output of convolution layers and used the fully connected layers. Those consist of 1568, 512, 256 and 9 nodes sequentially.

4. Experimental Results

For dataset, we used Microsoft Challenge dataset [10] that has nine classes, total 10,868 asm files. Among them, we can use 10,734 files during extracting opcode.

At first, for comparing feature image of n-gram opcode to the method of MCSC (1-gram), we generated images of each malware family by using algorithm 1 for 1-gram, 2-gram and 3-gram opcode. Some result of them is in the Fig. 6. As you can see, feature images maintained distinct features compared with other families and similar feature compared within same family according to n-gram.

We implemented the proposed method with Python 3.6 and Pytorch 1.1.0 version. We used parameters of neural network, batch size as 128, epoch as 1500, learning rate as 0.009, dropout rate as 0.7 and used weighted-cross entropy loss that give high weight to low number of class [11] because Microsoft challenge dataset is highly imbalanced.

We run our method for the 1-gram, 2-gram and 3-gram thirty times and measured accuracy, loss and max accuracy. The mean of them are in the table 1. The boxplot of accuracy is in the left side of Fig. 7. We also show the accuracy curve per epoch is in the right side of Fig. 7.

As we can see, the accuracy of MCSC is 0.954 and the 2-gram and 3-gram methods are 0.983 and 0.981 each. Standard deviation of them are similar. For loss, MCSC recorded mean loss over 7, while the others under 1.3. According to the loss, MCSC show unstable standard deviation than others. The max accuracy is 0.988 for 2-gram MCSC.

TABLE I. Accuracy & Loss

Name	Accuracy		Loss		Max accuracy
	mean	*std*	*mean*	*std*	
MCSC	0.954	0.0016	7.740	0.466	0.961
2-gram MCSC	0.983	0.0019	1.230	0.021	0.988
3-gram MCSC	0.981	0.0017	0.950	0.018	0.986

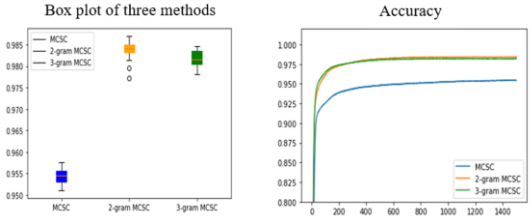

Figure 7. Boxplot and mean accuracy per epoch

As we can see the boxplot in the left side of Fig. 7, N-gram MCSCs (2-gram and 3-gram MCSC) have better accuracy than MCSC. For three methods to have difference statistically, we made analysis of variance (ANOVA). As the result of it, there is a significant difference (p value = 7205e-90 < 0.01). Therefore, we did post-hoc-analysis. In the case between 2-gram MCSC and 3-gram MCSC, the result showed significant difference (p = 0.000089 < 0.01). The mean accuracy per epoch in the right side of Fig. 7, it shows N-gram MCSCs can learn faster than MCSC.

We presented the performance of precisions in Fig. 8 for each malware family. MCSC showed good precision for most of classes except Obfuscator.ACY class. On the other hand, 2-gram and 3-gram MCSCs show huge improvement at Obfucator. In addition, 2-gram MCSC show good precision for each class. However, 3-gram MCSC got the lowest precision at Simda class, though it can detect most of classes well. Therefore, we can conclude the 2-gram method get the best precision among them.

Whenever the positive class is rare, the PR curve is important. Therefore, we presented PR curves for each method in Fig. 9. They are the result when we treat each malware as binary class. MCSC shows more unstable curves at most of the classes than 2-gram and 3-gram MCSC, which means that MCSC cannot detect malware properly. However, 2-gram and 3-gram MCSC show stable curve, which means they can detect malware, properly when even thresholds are high.

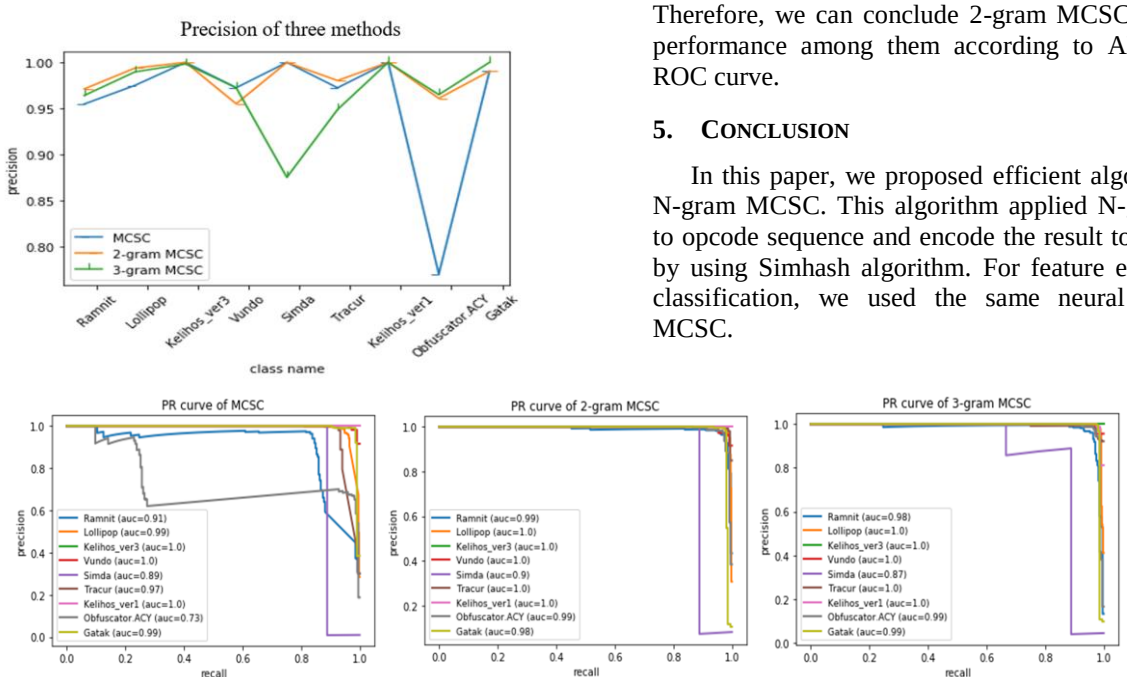

Figure 9. PR curve of compared methods

To quantify the performance of PR curve, we measured the AUC (area under curve) value of them and showed them within Fig. 9. The mean of PR-AUC value for MCSC is 0.942, for 2-gram is 0.984 and for 3-gram is 0.979. Therefore, we can conclude 2-gram MCSC got the best performance among them according to AUC value of PR curve.

The ROC curves for each malware family according to n-gram is shown in Fig. 10. They are also the result when we treat each malware as binary class. In the case of Lollipop, Kelihos series, Vundo and Gatak, 3 methods showed very stable curve. However, in the case of Ramnit, Simda and Obfuscator.ACY, MCSC showed unstable curve, while others still showed stable curve. Therefore, we can conclude 2-gram and 3-gram MCSC are better method than MCSC.

To quantify the performance of ROC curve, we measured the AUC value of them and showed them within Fig. 10. The mean of ROC-AUC value for MCSC is 0.987, for 2-gram is 0.997 and for 3-gram is 0.995. Therefore, we can conclude 2-gram MCSC got the best performance among them according to AUC value of ROC curve.

5. CONCLUSION

In this paper, we proposed efficient algorithm that is N-gram MCSC. This algorithm applied N-gram method to opcode sequence and encode the result to make image by using Simhash algorithm. For feature extraction and classification, we used the same neural network of MCSC.

To prove the performance of proposed method, we compared the result of MCSC, 2-gram MCSC and 3-gram MCSC through accuracy, loss, precision, AUC value of PR curve and ROC curve. In the view of accuracy and loss, we concluded the difference of three methods is significant by using ANOVA and post-hoc analysis. The 2-gram MCSC got the best result among three methods. In the view of precision, we concluded 2-gram MCSC showed the best performance. In the view of the PR-AUC value, we concluded the 2-gram MCSC got the best result among them. In the view of the ROC-AUC value, we concluded the 2-gram MCSC got the best result. In the all kinds of view, we concluded that 2-gram method showed best performance among the N-gram.

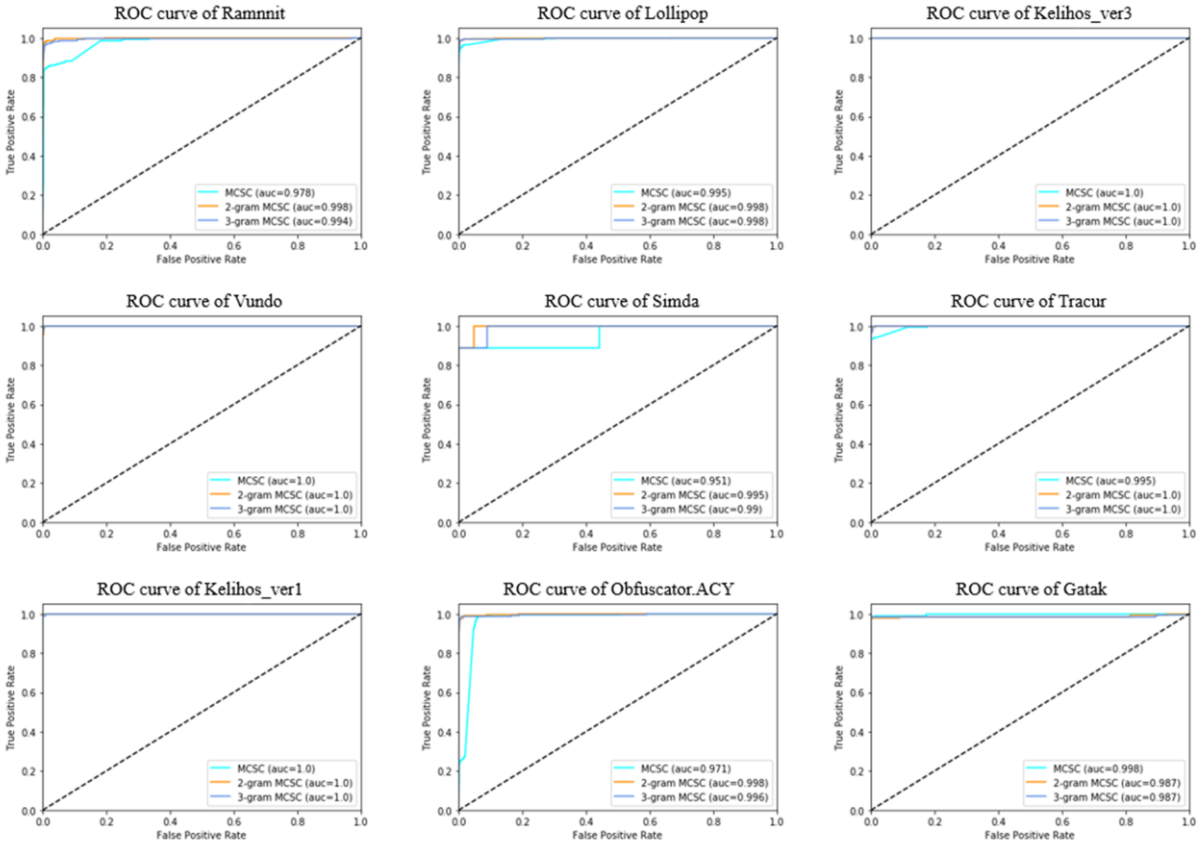

Figure 10. ROC curve for each malware family

ACKNOWLEDGEMENT

This work was supported by R.O.K. National Research Foundation under grant NRF-2017R1D1A1B03036372 in 2019.

REFERENCES

[1] E. Gandotra, D. Bansal and S. Sofat, "Malware Analysis and Classification: A Survey", Journal of Information Security, vol. 05, no. 02, pp. 56-64, 2014.

[2] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang. A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, pages 21–30. ACM, 2011.

[3] K. Han, J. H. Lim, and E. G. Im. Malware analysis method using visualization of binary files. In Proceedings of the 2013 Research in Adaptive and Convergent Systems, pages 317–321. ACM, 2013.

[4] S. Ni, Q. Qian, and R. Zhang, ‘‘Malware identification using visualiza-tion images and deep learning,’’Comput. Secur., vol. 77, pp. 871–885,Aug. 2018.

[5] M. Charikar. Similarity estimation techniques from roundingalgorithms. InSTOC, 2002.

[6] P. Gallagher, (2008). Secure Hash Standard (SHS).

[7] I.H. Witten and E. Frank, Data Mining-Pracitcal Machine Learning Tools and Techniques with JAVA Implementations. Morgan Kaufmann, 2000

[8] Igor Santos, Yoseba K. Penya, Jaime Devesa and Pablo G. Bringas. “N-GRAMS-BASED FILE SIGNATURES FOR MALWARE DETECTION”,S3Lab, Deusto Technological Foundation, Bilbao, Basque Country fisantos, ypenya, jdevesa, pgbg@tecnologico.deusto.es

[9] M. Buckland and F. Gey, “The Relationship between Recall and Precision,” J. Am. Soc. for Information Science, vol. 45, no. 1, pp. 12-19, 1994.

[10] "Microsoft Malware Prediction | Kaggle", Kaggle.com, 2019.

[11] Yue, S., 2017. Imbalanced Malware Images Classification: a CNN based Approach.arXiv preprint arXiv:1708.08042.[6]LeCun, Y., 2015. LeNet-5, con

Myung-Jae Lim
1998.2 Ph.D. in Department of Computer science, Chung Ang university, Korea.
1992.3 ~ Professor in Eulji university, Korea.

Jae-Ju An
2014.3 ~ Junior in Department of Medical IT, Eulji university, Korea.

So-Hee Jun
2015.3 ~ Senior in Department of Medical IT, Eulji university, Korea

Young-Man Kwon
1985.2 M.S. in Department of Electric and Electronic engineering, KAIST, Korea.
1998.3 Doctor course completion, in Department of information and communication engineering, KAIST, Korea.
2007.2 Ph.D. in Department of Electronic engineering, Kwangwoon university, Korea
1993.3 ~ Professor in Eulji university, Korea.