A New approach for Malware Detection Based on Evolutionary Algorithm

Farnoush Manavi
Department of Computer Science and Engineering & IT
Iran
F.Manavi@cse.shirazu.ac.ir

Ali Hamzeh
Department of Computer Science and Engineering & IT
Iran
Ali@cse.shirazu.ac.ir

ABSTRACT

Malware is a malicious code which intends to harm computers and networks. Each year, a huge number of malicious programs are released. Therefore, detecting malware has become one of the most important challenges for the security of computer systems. Various methods have been defined for detecting and classifying malware, such as signature-based and heuristic-based techniques. This paper proposes a new malware detection method based on the operational codes (OpCodes) within an executable file by using the evolutionary algorithm. There are several steps in the proposed method, which includes disassembling the executable files, generating a graph of OpCodes and using the evolutionary algorithm to find the most similar graph to each suspicious instance. Finally, the label of each suspicious instance is detected based on the most similar graph obtained from the evolutionary algorithm with each class (family of malware and benign). The results show that, the proposed method can be used as a method for malware detection and malware category.

CCS CONCEPTS

•Theory of computation → Evolutionary algorithms • Security and privacy → Malware

KEYWORDS

Classification, Evolutionary algorithm, Malware detection, OpCode, Graph.

ACM Reference format:

F. Manavi and A. Hamzeh. 2019. SIG Proceedings Paper in word Format. In *Proceedings of ACM GECCO conference, Prague, Czech Republic, July 2019 (GECCO'19)*, 6 pages.
https://doi.org/10.1145/3319619.3326811

1 INTRODUCTION

Malware is a program that is developed with malicious purpose, such as harming computer systems or doing unwanted actions on a computer system [1]. Virus, Worm, Trojan, Spyware, Adware, Rootkit, Backdoors, and etc. are among different types of malware [2]. The number of malware attacks has increased dramatically. Therefore, identifying malware has been more critical in computer systems [3]. Softwares, such as anti-virus and anti-malware programs or devices used to try and protect a user's computer against activity identified as malicious, and to recover from attacks. However, malware writers apply various concealment strategies to deceive malware detectors such as anti-malware, when they realize their malware are going to be detected. Therefore, there is an intensive competition between malware and anti-malware [2, 4, 5].

A branch of computer security is malware detection which attempts to analyze suspicious programs and detect malware [2]. There are many approaches to detect malware, which are usually divided into three categories [6]:

Signature-based techniques: This approach is based on the assumption that malware can be described through patterns which is called signature. This detection scheme identifies the presence of a malware infection or instance by matching at least one signature of the software in question with the database of signatures of known malicious programs. This method is the most commonly used technique for anti-malware systems that is relatively fast but, it is not able to detect unknown malware [7].

Behavior-based techniques: This approach monitors behaviors of a program to extract features and determine whether it is a malicious or not. Therefore, each application must run in a virtual environment and the actual behavior of the program is observed. Majority of malware do not perform their actual behavior for some time after the program is running as to deceive the detection method [6-8]. Due to its low speed, this method is not suitable for rapid detection.

Heuristic-based techniques: This approach uses operational codes (Opcodes) or file bytes to detect malware. Typically, this approach utilizes machine learning methods [2, 6].

Considering that signature-based methods compare a signature, these methods are only able to detect known malware and those are weak to detect new (or unknown) ones. Behavior-based methods also have quite low speed, which makes them inefficient for fast detection. Based on the shortcomings mentioned, this research focuses on malware detection by heuristic-based techniques.

Biological behaviors have always been a good source of inspiration for computer scientists. Among them, Darwinian evolution have shown a good potential for being a model for search algorithms. Nowadays, their ideas apply to various problems like optimization, search, or problems from machine learning. Evolutionary algorithms (EAs) are generic population-based metaheuristic optimization algorithms. EAs usually have better performance in challenging search spaces than alternative methods. This property of EAs has made them an efficient way to approximate solutions [9]. Therefore, in this research instead of using machine learning methods like previous approaches evolutionary algorithms are used, and it attempts to detect malware based on discovered malware.

The structure of this paper is organized as follows. Section 2 provides related work in malware detection. Section 3 is focused on the detailed description of the proposed method, overview of the system, graph extraction and evolutionary algorithm. The system evaluation and discussion is presented in Section 4. Finally, Section 5 provides conclusion.

2 RELATED WORK

Some methods for malware detecting rely on features extracted from API calls, strings, byte n-grams, OpCode, etc. Santos et al. [10] proposed a method to identify critical windows malware based on the frequency of appearance of OpCode sequences. They showed the cosign relevance of two OpCodes based on the frequency with which it appears in malware and benign files. They computed the mutual information between each OpCode and provided a method which uses weighted similarity function for OpCode relevance. They created a vector for each file that each of its elements represent the Weighted Term Frequency related to an OpCode sequence. Finally, these vectors are utilized to train a machine learning model by the purpose of classifying unknown instances. Runwal et al. [11] proposed a metamorphic malware detection method based on the OpCodes within an executable file. They generated a weighted directed graph according to OpCode diagram, which is created based on counting the number of OpCode pairs that appeared in the OpCode sequence. Finally, they classified each file as malware or benign by measuring the similarity between the extracted graphs. Rad et al. [12] build a database of different variants of the morphed virus and used the histogram of OpCodes as a feature to determine classification of metamorphic virus family variants. Hashemi et al. [5] proposed a malware detection method based on the OpCodes within an executable file. They generated a graph of OpCode within an executable file and then embedded this graph into eigenspace using "Power Iteration" method. They converted an executable file as a linear combination of eigenvectors proportionate to their eigenvalues, which is beneficial to train machine learning classifiers such as k-nearest neighbor and support vector machine. Manavi et al. [2] proposed a malware detection method based on the OpCodes within an executable file by using image processing techniques. They generated a graph of OpCodes from an executable file and converted this graph to an image and then using "GIST" method in order to extract features from each image and then used machine learning methods such as Support Vector Machine, K-Nearest Neighbor, Ensemble to classification. Darabian et al. [13] used sequential pattern mining technique to detect most frequent OpCode sequences of malicious IoT applications. They used maximal frequent patterns (MFP) of OpCode sequences to differentiate malicious from benign IoT applications. Srivastava et al. [14] proposed a feature extraction process based on Genetic process for malware detection. They used the existing features to generate new features; These newly generated features are then evaluated using a fitness function. Then, they used the extracted features to train the classifier to Malware Detection process.

3 PROPOSED METHOD

As it was mentioned, this research is focused on malware detection based on heuristic techniques. This section describes how to construct adjacency graph for malware and benign files, and explains how to determine malware or benign files based on adjacency graph using an evolutionary algorithm.

3.1 The Overview of the System

The proposed method consists of two main preprocessing steps. At first, the database of executable files is disassembled. Second, for each file based on binary combinations of consecutive OpCodes a graph is constructed. In this graph weight to each edge represents the number of binary combinations of consecutive OpCodes. These graphs will be the initial population of the evolutionary algorithm. For the target file, OpCodes are extracted and our system creates graph based on these extracted OpCodes. Using the evolutionary algorithm, the most similar graph to each class (family of malware and benign) is constructed. Then, the graph of the suspicious file is compared with the obtained graphs from the designed evolutionary algorithm. After these comparisons, we can classify the target file to the most similar class. The overview of the system is illustrated in Fig. 1.

3.2 Graph Adjacency Matrix

After disassembling the executable files, the sequence of the file OpCodes is extracted. For each file, a graph of the possible binary combinations of consecutive OpCodes is built. This graph shows a directed and weighted graph which represents frequency of 2-gram OpCodes in a program. In this graph, each node represents an OpCode of files, and each directed edge from node v1 to node v2 means that OpCode v2 is after the OpCode v1, and the weight on the edge shows the number of these occurrences. Table 1 shows a small part of the instructions and OpCodes of malware called *Worm.Win32.Doomer* and Fig. 2 and Table 2 gives a graph and adjacency matrix of graph which is shown in Table 1.

3.3 The Evolutionary Part

Evolutionary algorithms (EAs) are meta-heuristic methods for solving computationally difficult problems. EAs often perform well approximating solutions to all types of problems because those ideally do not make any assumption about the underlying fitness landscape. In this proposed method, the evolutionary algorithm is used to specify the label of each file. First, for all the files in dataset, from the OpCodes of 2-grams a graph is extracted.

Then, the suspicious instance along with the graphs of each class will be as an input of the evolutionary algorithm. In other words, according to target instance, the most similar graph from each class (family of malware and benign) is created. Finally, the most similar graph of the resulting evolutionary algorithm will determine the target label. The evolutionary algorithm which has been used is depicted in Algorithm 1.

Step 1, creates an initial population for each graph of each class. Step 2, selects parents from population for creating new individuals by genetic variation. In step 3, offspring populations are created by crossover and mutation. In step 4, the fitness of each individual (offspring and populations) is calculated and saved best graph. Step 5, replaces N best individual from current population and offspring to next population. Finally returns the best chromosome. If best chromosome satisfies the termination condition or the number of repetitions to be sufficient, the algorithm stops, otherwise the algorithm jumps to step 2. In the following, is described the details of our evolutionary algorithm.

3.3.1 Representation

Each chromosome is represented as a matrix, which represents the graph adjacency matrix of 2-grams of OpCodes. If the number of unique OpCodes in the dataset are N, then the size of the matrix will be N * N. For example, Table 2 gives adjacency matrix of graph which is shown in Table 1.

3.3.2 Initialization

A part of the training set is selected and used for the initial population. The size of the population is proportional to the dataset size between 100 and 200.

3.3.3 The Fitness Function

Since each graph is represented as a N*N matrix, and the goal of the algorithm is to obtain the most similar graph to the target graph, Euclidean distance is used as the fitness function.

3.3.4 Selection

Since tournament selection is not sensitive to the fitness value of population [15], we used this method for parent selection. Often tournaments are held between 2-5 individuals, so 5th tournament selection is used here. Five individuals randomly are chosen from the population and the best individual will be selected as a parent.

3.3.5 Crossover

Uniform crossover is used for the proposed method. In this crossover, each bit from the offspring's genome is independently chosen from the two parents according to a given distribution. In other words, each element of the offspring matrix is selected with equal probability from their parents. The rate of crossover is set to 0.9.

3.3.6 Mutation

For mutation the maximum and minimum allele value in chromosome is determined and randomly an integer of this interval is selected. Then, one positions in the chromosome randomly is selected and its allele value is changed to that integer mentioned. The rate of the mutation is set to 0.1.

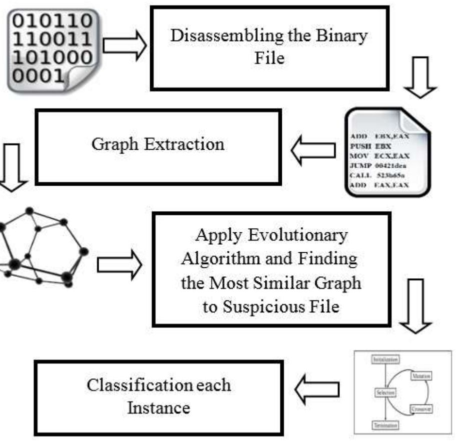
010110
110011
101000
0001
Disassembling the Binary File
ADD EBX,EAX
PUSH EBX
MOV ECX,EAX
JUMP 00421dea
CALL 523b65a
ADD EAX,EAX
Graph Extraction
Apply Evolutionary Algorithm and Finding the Most Similar Graph to Suspicious File
Classification each Instance

Figure 1: Overview of the Proposed Method.

Table 1: Extracted OpCodes from Executable File

Line	OpCode	Operand
1	push	ebp
2	mov	ebp, esp
3	push	0FFFFFFFFh
4	push	offset stru_422220
5	push	offset sub_4036E4
6	mov	eax, large fs:0
7	push	eax
8	**mov**	large fs:0, esp
9	**add**	esp, 0FFFFFFF0h
10	push	ebx
11	push	esi
12	push	edi
13	mov	[ebp+ms_exc.old_esp], esp
14	call	ds:GetVersion
15	mov	dword_425764, eax
16	mov	eax, dword_425764
17	shr	eax, 8
18	and	eax, 0FFh
19	mov	dword_425770, eax
20	mov	ecx, dword_425764
21	and	ecx, 0FFh
22	mov	dword_42576C, ecx
23	**mov**	edx, dword_42576C
24	**add**	edx, dword_425770
25	mov	dword_425768, edx
26	mov	eax, dword_425764
27	shr	eax, 10h
28	and	eax, 0FFFFh
29	mov	dword_425764, eax
30	push	0

Table 2: The Adjacency Matrix of Graph

OpCode	add	and	call	shr	push	mov
add	0	0	0	0	1	1
and	0	0	0	0	0	3
call	0	0	0	0	0	1
shr	0	2	0	0	0	0
push	0	0	0	0	4	4
mov	**2**	1	1	2	3	4

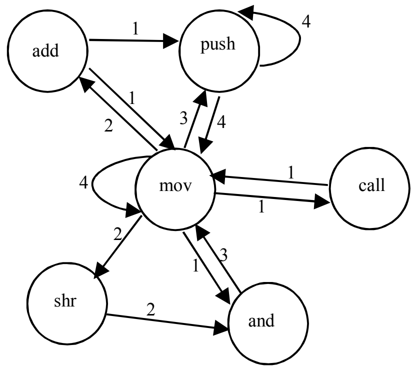

Figure 2: The OpCode Graph Shown in Table 1.

```
Algorithm 1: Evolutionary algorithm.
    Input: The target graph (G) along with the graphs of each
    class (family of malware and benign).
    Output: The most similar graph to the target graph.
    Step1: Create initial population from graphs of each
    class.
    repeat
        Step2: Choose parents from population
        Parent1= tournament(population)
        Parent2= tournament(population)
        Step3: apply Cross over and mutation on parents
        Offspring1, 2= Xover(parent1, parent2)
        Offspring1= Mutation(Offspring1)
        Offspring2= Mutation(Offspring2)
        Step4: Add offspring to the population and save best
        chromosome (the most similar graph to G).
        Step5: Sort chromosomes based on fitness and
        replace N best chromosome as next generation.
    until termination
```

3.3.7 Replacement

For survival, replaces N best individual from current population and offspring to next population is considered.

3.3.8 Stopping Criterion

Proposed algorithm stops when it reaches to the zero fitness. In this case it finds the target graph. Otherwise has been set the number of maximum generation to 500.

4 RESULTS AND DISCUSSION

This section provides information on our experimental results. First, the evaluation metrics, dataset, hardware and software used in the experiments is discussed. Then, main results are presented and proposed method is compared to other relevant methods.

4.1 Evaluation Metrics

To evaluate the competency of proposed method, some common machine learning performance evaluation metrics such as Accuracy and F-measure are used. At first, some required terms for introducing Accuracy and F-measure are defined. The true positive (TP) indicates the number of items which are correctly labeled as the positive class. True negative (TN) represents the number of items which are correctly labeled as the negative class. False positive (FP) indicates the number of items which are incorrectly labeled to be in the positive class. False negative (FN) represents the number of items which are incorrectly labeled as negative. Table 3 shows the relationships between these metrics.

Table 3: Evaluation Metrics Formula

Metrics	Formula
Recall	$\frac{TP}{TP + FN}$
Precision	$\frac{TP}{TP + FP}$
Accuracy	$\frac{TP + TN}{TP + FP + TN + FN}$
F-measure	$2 * \frac{Presision * Recall}{Precision + Recall}$

4.2 Dataset

In order to evaluate the proposed method, three different datasets are used[2]. Each dataset has different features and samples. Dataset number 1 has 3200 files including 1600 malware and 1600 benign samples where malware files are selected randomly from executable files of VX Heavens virus collection which is one of the well-known datasets in malware detection context and includes different kinds of malware such as Virus, Trojan, Backdoor, Hacking tools and Rootkits. Benign files are also selected randomly from original Windows files and other utility software. All benign files in dataset are scanned using ESET NOD32 and KASPERSKY to ensure that they are not contaminated. Dataset number 2 has 4000 APK files including 2000 malware and 2000 benign APK samples. The android malware dataset is a subset of the benchmark dataset called Drebin which provided by Arp et al [16]. Drebin dataset contains 2631 Android non-malicious application and more than 5500 Android malware files. Benign files are also selected randomly from android application markets (Google Play store). All benign files in dataset are scanned using VirusTotal service to ensure that

[2] In order to access the data, send an email to f.manavi@cse.shirazu.ac.ir

they are not contaminated. Dataset number 3 has 2042 files including nine different malware families where malware families are selected randomly from Kaggle Microsoft Malware Classification challenge (BIG 2015) [17].

4.3 The Experimental Environment

All experiments were done under the environment with following specification: Windows 10 as operating system, AMD FX(tm)-6200 Six-Core Processor 3.80 GHz and 32GB of RAM. MATLAB 2018 and Python 3.3 is used for implementation Evolutionary algorithm and graph extraction task. IDA Pro tools, Baksmali and Androguard are chosen in our work for extracting OpCodes.

4.4 Discussion of the Results

Polymorphic and metamorphic malware employ obfuscation techniques to bypass traditional detection methods. Owing to the fact that, any file has usually a specific sequence of OpCodes in its nature, obfuscation techniques can not be covered fully in this nature. But, extracting high-level features such as Opcode, function calls or program's control flow graph(CFG) alone is not enough. Evolutionary algorithms have an ability to evolve a sample of the population and build new ones. Therefore, evolutionary algorithms can be used to find out the other structures of a malware. The proposed method attempts to detect the structure of new malware due to discovered malware and the capabilities of evolutionary algorithms. The results show that the proposed method has been effective in achieving this goal.

4.5 Comparison with other Methods

The proposed method is compared with two powerful Hashemi et al. [5] and Santos et al. [10] methods which are based on OpCodes. Moreover, in Tables 4-6, the proposed method is compared with Nataraj et al. [18] method which is common method based on raw bytes. Hashemi et al. [5] generated a graph of OpCode within an executable file and then embedded this graph into eigenspace using "Power Iteration" method. Finally, they used eigenspace as representative sample feature-set for training a machine learning model. Santos et al. [10] disassembled files and extracted OpCodes. They generated OpCode profile that is a matrix shows the frequency number of each OpCodes in all files. They showed the cosign relevance of two OpCodes based on the frequency with which it appears in malware and benign files. They computed the mutual information between each OpCode and provided a method which uses weighted similarity function for OpCode relevance. They created a vector for each file that each of its elements represent the Weighted Term Frequency related to an opcode sequence. Finally, these vectors are utilized to train a machine-learning model by the purpose of classifying unknown instances. Nataraj et al. [18] represented executable file as a binary string of zeros and ones. Then, they converted these strings to grayscale images and used GIST [19] to compute texture features. The process ends with a K-Nearest Neighbors classification with Euclidean distance as the distance measure for classification.

The experiments show that our method has good results in comparison with other mentioned methods. Therefore, the proposed method can be used as a method for detecting and categorizing malware.

In Tables 4-6, the maximum value of each row is highlighted in bold. For the Hashemi, Santos and Nataraj methods, the best result of KNN (k=1:10) is expressed.

Table 4: Comparing Results Obtained from Dataset 1

	Proposed method	Santos et al. [10]	Hashemi et al. [5]	Nataraj et al. [18]
Accuracy	85.80	85.72	**86.56**	80.06
F-measure	86.41	**86.61**	86.12	79.84

Table 5: Comparing Results Obtained from Dataset 2

	Proposed method	Santos et al. [10]	Hashemi et al. [5]	Nataraj et al. [18]
Accuracy	85.80	85.30	**86.69**	73.90
F-measure	85.18	85.28	**87.10**	74.78

Table 6: Comparing Results Obtained from Dataset 3

	Proposed method	Santos et al. [10]	Hashemi et al. [5]	Nataraj et al. [18]
Accuracy	**87.67**	80.23	75.39	68.80
F-measure	**86.71**	78.47	70.18	66.50

5 CONCLUSIONS

In this research, we have been looking for a novel approach with high detection rates to detect unknown malware based on their OpCode sequence and Evolutionary algorithm; For this purpose, binary sequences of Opcodes have been extracted from the executable files and have been converted to a graph corresponding to each file. Using the evolutionary algorithm, the most similar graph to each class has been constructed. Then, the graph of the suspicious instance has been compared with the obtained graphs from the designed evolutionary algorithm. After these comparisons, we can classify the target instance to the most similar class.

REFERENCES

[1] Christodorescu, M., Jha, S., Maughan, D., Song, D., & Wang, C. (Eds.). (2007). *Malware detection* (Vol. 27). Springer Science & Business Media.

[2] Manavi, F., & Hamzeh, A. (2017, October). A new method for malware detection using opcode visualization. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)* (pp. 96-102). IEEE.

[3] Farrokhmanesh, M., & Hamzeh, A. (2018). Music classification as a new approach for malware detection. *Journal of Computer Virology and Hacking Techniques*, 1-20.

[4] Jang, J. W., Kang, H., Woo, J., Mohaisen, A., & Kim, H. K. (2016). Andro-dumpsys: anti-malware system based on the similarity of malware creator and malware centric information. *computers & security*, *58*, 125-138.

[5] Hashemi, H., Azmoodeh, A., Hamzeh, A., & Hashemi, S. (2017). Graph embedding as a new approach for unknown malware detection. *Journal of Computer Virology and Hacking Techniques*, *13*(3), 153-166.

[6] Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013, May). A survey on heuristic malware detection techniques. In *The 5th Conference on Information and Knowledge Technology* (pp. 113-120). IEEE.

[7] Mujumdar, A., Masiwal, G., & Meshram, D. B. (2013). Analysis of signature-based and behavior-based anti-malware approaches. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, *2*(6).

[8] Ye, Y., Li, T., Jiang, Q., & Wang, Y. (2010). CIMDS: adapting postprocessing techniques of associative classification for malware detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(3), 298-307.

[9] Wiegand, R. P., & Sarma, J. (2004, September). Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In International Conference on Parallel Problem Solving from Nature (pp. 912-921). Springer, Berlin, Heidelberg.

[10] Santos, I., Brezo, F., Nieves, J., Penya, Y. K., Sanz, B., Laorden, C., & Bringas, P. G. (2010, February). Idea: Opcode-sequence-based malware detection. In International Symposium on Engineering Secure Software and Systems (pp. 35-43). Springer, Berlin, Heidelberg.

[11] Runwal, N., Low, R. M., & Stamp, M. (2012). Opcode graph similarity and metamorphic detection. Journal in Computer Virology, 8(1-2), 37-52.

[12] Rad, B. B., Masrom, M., & Ibrahim, S. (2012, September). Opcodes histogram for classifying metamorphic portable executables malware. In 2012 International Conference on e-Learning and e-Technologies in Education (ICEEE) (pp. 209-213). IEEE.

[13] Darabian, H., Dehghantanha, A., Hashemi, S., Homayoun, S., & Choo, K. K. R. An opcode-based technique for polymorphic Internet of Things malware detection. Concurrency and Computation: Practice and Experience, e5173.

[14] Srivastava, P., & M. Raj, (2018). Feature extraction for enhanced malware detection using genetic algorithm. *International Journal of Engineering & Technology*, 444-449.

[15] Blickle, T., & Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms. TIK-Report 11, TIK Institut fur Technische Informatik und Kommunikationsnetze. Computer Engineering and Networks Laboratory, ETH, Swiss Federal Institute of Technology, Gloriastrasse, 35, 8092.

[16] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. E. R. T. (2014, February). Drebin: Effective and explainable detection of android malware in your pocket. In Ndss (Vol. 14, pp. 23-26).

[17] Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., & Ahmadi, M. (2018). Microsoft malware classification challenge. arXiv preprint arXiv:1802.10135.

[18] Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011, July). Malware images: visualization and automatic classification. In Proceedings of the 8th international symposium on visualization for cyber security (p. 4). ACM.

[19] Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. International journal of computer vision, 42(3), 145-175.