A Gene-Inspired Malware Detection Approach

Yihang Chen¹, Zheng Shan¹, Fudong Liu¹, Guanghui Liang¹, Binglin Zhao¹, Xingwei Li¹ and Meng Qiao¹

¹State Key of Laboratory of Mathematical Engineering and Advanced Computing, 450001 Zhengzhou, Henan, China

Abstract. Malware detection is an important topic in cyber security. The research presented in this paper mainly studies on disassembly codes of Windows executable files, learns from the research route of bioinformatics and proposes the concept of software gene. A distance-based method is also proposed to measure the difference of genes and the dimensionality reduction based on a modified clustering algorithm of biological phylogenetic model. Finally a gene-inspired malware detector is constructed using Random Forest model. The software gene extraction proposed in this paper is more flexible and generates less data than the widely-used n-gram method. The detector based on genes also performs better. The clustering-based dimensionality reduction retains more comprehensive features and maintains the interpretability in software analysis area. The detector based on gene-inspired malware detection approach can reach the precision 96.14%, which is better than traditional methods.

1. Introduction

In recent years, severely threatened malware, such as Mirai and WannaCry, emerged in an endless stream, and caused huge losses to the world. At least 136 million new malware samples were detected in 2017 [1]. Millions of new malware samples are produced with the assistance of automation tools per year. Traditional methods, like signature matching or rule-based detection, are lack of ability to detect unknown malware and their variants. Artificial analysis is precise but inefficient. It is an inexorable trend to construct new-generation malware detectors using data mining techniques.

The research presented in this paper mainly studies on the disassembly data of PE files (namely executable files on the Windows platform) generated by the static decompiling tool, and propose a gene-inspired malware detection approach. In bioinformatics, the research contents include gene similarity, gene homology, virulence gene detection and etc. There is something in common with malware analysis. Malware analysis aims to classify malware families or detect abnormal samples from benign files by analysing the code fragments. Therefore, this research fully learns from the research routes and related algorithms of bioinformatics, and applies them to the malware detection area.

Several gene-inspired malware analysis approaches have been proposed, but most of them have their own limitations. Drew et al. [2,3] directly mapped the binary data of target files to DNA molecules, which lost the information of semantics, and caused poor interpretability. Kirat et al. [4] took the whole execution sequences of samples as DNA molecules and used Smith-Waterman algorithm [5] in sequence alignment. However, most of execution sequences are too long to align efficiently so this approach is time-consuming.
The workflow of the malware detection approach in this paper is as shown in Figure 1. Firstly, the software gene of samples in the data set will be extracted. Then is how to measure the distance of each pair of genes using Smith-Waterman algorithm [5], and classify similar genes into the same classes by introducing and modifying Neighbour-Joining (NJ) algorithm [6], which is used in constructing biological phylogenetic model. Actually, this is also a dimensionality reduction process. Unlike TFIDF [7] method, it retains all features but doesn’t filter out any of them. And unlike Principal Component Analysis (PCA) [8] or other similar methods based on feature space compress using mathematical algorithms, the process maintains the interpretability in software analysis. It is important since deleting a benign program by mistake is serious, and it needs to provide a reasonable explanation for the detection results. In vectorization process, one class of genes, instead of one kind of genes, is mapped onto one dimension of feature vector. Finally, the gene-based malware detector using Random Forest is constructed and reaches the average precision 96.14%.

2. Methodology

This section can be divided into four parts. Firstly, how to define a software gene is presented. Then is the distance measurement between a pair of genes. Gene classification based on NJ will be shown next. The final step includes vectorization and machine learning models.

2.1 Software Gene

In biology, the existence form of genes is a sequence of deoxyribonucleic acids [9]. If a gene is expressed, the whole sequence will transcript and translate.

Similarly, a software gene is defined as a sequence of code fragment that:

• is a sequence of instructions without partial execution must be ended with conditional jump, return or user-function call instructions;

• there is not any instruction in the contexts to merge with the software gene to form a larger gene.

Actually, a software gene is the combination of one or more basic blocks [10]. Basic block regards all call instructions as the boundary of segmentation. But it is too fragmental for malware analysis. Software gene allows to contain system API calls. An API calls can be treated as a complex assemble instruction.

In this definition, software gene can be still described by a short sequence instead of a complex graph [11]. The gene-based slicing method is also more flexible than widely-used n-gram method [12,13]. The experiments show that the types of n-gram fragments are 10 times more than software genes but very few of them will be used finally. The gene-based malware detector also performs better than n-gram-based detector.

A program needs to call system APIs to achieve their intentions on file system or network. Therefore, API calling sequence inside the software gene is used to describe its function and get rid of the noise from non-critical instructions.

2.2 Gene Distance Measurement

Though types of genes in samples are much less than n-gram fragments, there are still a very large number of genes. If different kinds of genes are mapped to dimensions of feature vector one by one directly, the dimension number of features will be too high to train an effective detector. In this step,
how to measure the distance between two different software genes will be discussed. The similar genes with short distance will be then classified into the same group in the next step.

Firstly, Smith-Waterman algorithm [5] is used to find the longest common API subsequence in a pair of genes. Smith-Waterman has been widely used in biological sequence alignment. It is also a dynamic programming algorithm. Assumed that there are two given sequence $A=a_1,a_2,...,a_n$ and $B=b_1,b_2,...,b_m$. The highest matching score of the subsequence $a_1,a_2,...,a_i$ and $b_1,b_2,...,b_j$ is

$$H_{ij}=\max\{H_{i-1,j-1}+\partial(i,j),\max\{H_{i-1,j}-W_k\},\max\{H_{i,j-1}-W_i\},0\}$$

(1)

where $1 \leq i \leq n, 1 \leq j \leq m$. $W_k$ is the mismatching punish score of position $k$, $\partial(i,j)$ is the matching score. Because these values should be set by priori knowledge, in this paper all $W$ values is set to be 0 and $\partial(i,j)$ is set to be 1. Namely, $H_{nm}$ is the length of longest common sequence of sequence A and B. The distance between sequence A and B is defined as

$$d_{AB}=\max\{n,m\}-H_{nm}$$

(2)

In the analysis of software genes, the functional difference of software genes can be described as the distance of API sequences in the genes. Fortunately, most of API sequences of software genes are very short. 94.8% API sequences are shorter than 10 in our data set. Thus, gene distance measure process are relatively quick.

2.3 Similar Gene Clustering

Based on the gene distance measure, the clustering algorithm is proposed through modifying Neighbour-Joining(NJ) algorithm [6] to categorize similar genes. NJ algorithm is a basic algorithm in biological phylogenetic model. It arranges the genes and their variants by similarity values to construct gene evolution trees. At the same time, NJ is also a kind of hierarchical clustering algorithm. The parameters of the algorithm can be adjusted to reach a state that the clustering algorithm reaches a specific cluster number. In traditional approaches, researchers may use some dimensionality reduction methods in data science in this step. However, some methods like TFIDF score [8] that Shabtai et al. [12] used, only keep obvious features but losses other information. Some methods based on dimensional compression like principal component analysis(PCA) is poor in interpretability.

NJ algorithm firstly calculate the sum of distance of each gene to all other genes. For software gene $i$,

$$r_i=d_{1i}+d_{2i}+...+d_{Ni}$$

(3)

where $N$ is the number of genes in our data set.

For each pair of genes, the index value $M_{ij}$ is calculated using the following formula:

$$M_{ij}=d_{ij}/(r_i+r_j)/(N-N')$$

(4)

where $N'$ is the number of the target cluster number.

Select the minimum value of $M_{ij}$ and merge gene $i$ and $j$ into the same cluster. Then use the formula (3) and (4) again to update the $r$ and $M$ values for each element. Circularly merge the gene set until the cluster number reaches $N'$.

However, the efficiency of NJ algorithm is very slow. All the $r$ and $M$ values need to be updated after every merging operation. But gene clustering in our approach needn’t to be as precise as evolution tree construction. Therefore, the precision requirement of NJ algorithm is lowered and the vector $r$ and matrix $M$ are not updated every time. A threshold $M'$ is introduced. When a gene is merged with a cluster to be a new cluster, check that if the $M$ value of each pair of genes in the new cluster is smaller than $M'$. If not, the merging will be console. By adjusting the threshold $M'$, the algorithm will end when the cluster number is near the target number $N'$. Different $M'$ values have been tested, and when $M'$ is -8 the $N'$ is approximately 3000, the detector performs the best on our sample set.
2.4 Vectorization and Machine Learning

The gene information is mapped to the feature vector using the clustering results of modified NJ algorithm. A cluster of gene maps to a dimension of the vector. In a feature vector of a certain sample, the value of the $i^{th}$ dimension in the vector is the similarity of the most similar gene in the sample to the longest common sequence of the $i^{th}$ gene cluster:

$$\text{Similarity} = \frac{H_{nm}}{\max\{n,m\}}$$

(5)

After vectorization, the gene-based malware detector combined with machine learning models is constructed. In the experiments, the performance of Decision Tree (DT), Support Vector Machine (SVM), Linear Discriminant Analysis (LDA) and Random Forest (RF) models have been compared. The experiment results show that RF model performs the best with the highest accuracy 96.1%.

3. Experiments

In this section the experiment results will be shown to verify the effectiveness of our approach. 12,497 malware samples and 13,378 benign samples have been collected in the wild for this research. The experiments can be separated into four groups. The first group experiments compare the distribution of genes in malware and benign sample set to intuitively verify the effectiveness of software gene. The second, third and fourth group experiments respectively compare and analyse the performance of detector based on software gene and n-gram fragment, modified NJ and TFIDF algorithm, four different machine learning models. The last three experiments all used ten-cross validation methods.

Assumed that TP, FP, TN and FN are the number of true positive samples, false positive samples, true negative samples and false negative samples respectively. The four evaluation indexes include:

- **Accuracy**: $\text{ACC} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$
- **False Positive Rate**: $\text{FPR} = \frac{FP}{(FP+TN)}$
- **Precision**: $\text{Precision} = \frac{TP}{(TP+FP)}$
- **Recall**: $\text{Recall} = \frac{TP}{(TP+FN)}$

3.1 Statistical Characteristics of Genes

Different genes contained in the samples in the dataset has been counted, and the frequencies of genes that appear in malware and benign samples have been calculated respectively and compared. Figure 2 shows the top 10 genes that frequently appear in malware but rare in benign samples. The ratio of the two frequencies of every gene is larger than 20 times. In contrast, Figure 3 shows the top 10 genes that frequently appear in benign samples but rarely appear in malware samples.

Through the two figures, it can be intuitively observe that the distributions of genes that appear in malware and benign samples are quite different. That is because the functions and behavior patterns are different in malware and benign samples. Thus, the codes and program logics has somewhat difference. The gene-inspired malware detector is based on this characteristic.

Figure 2. The top 10 genes that frequently appear in malware samples but rarely appear in benign samples.
3.2 Gene and N-gram Slicing Method

N-gram program slicing method is widely used to extract code features of malware, but this method usually generates a mass of data. In the sample set, all the codes in the samples is serialized by their physical addresses and use 3-gram method [12] to extract features. The process has been simplified that only extract API calls instead of all opcodes. But there are 635,049 different 3-gram fragments while there are only 69,843 genes. The scale of analysis is sharply reduced. The increasing curves of types of genes and n-gram fragments with the growth of sample size, has been shown in Figure 4. The growth of the curve of genes is much slower than n-gram.

In addition, the detection approach based on software genes performs better than n-gram. Two malware detectors respectively based on software gene and n-gram method is constructed, corresponding with TFIDF and Random Forest algorithms, on our sample set. Their performance is shown at second and third row in Table 1. In Shabtai et al. [12] and Santos et al. [13] research, the detector based on n-gram method has achieved good results. However, the gene method further improve the performance.

| Detectors            | ACC   | FPR   | Precision | Recall |
|----------------------|-------|-------|-----------|--------|
| Gene+NJ+RF           | 0.9614| 0.0207| 0.977     | 0.942  |
| Gene+TFIDF+RF        | 0.9509| 0.0245| 0.9724    | 0.9247 |
| Ngram+TFIDF+RF       | 0.9433| 0.0389| 0.9568    | 0.9242 |
| Gene+NJ+DT           | 0.9495| 0.0381| 0.9583    | 0.9365 |
| Gene+NJ+LDA          | 0.9284| 0.0613| 0.9332    | 0.9175 |
| Gene+NJ+SVM          | 0.9117| 0.0642| 0.9281    | 0.8853 |

3.3 TFIDF and Modified NJ
In the third experiments group, the effectiveness of modified NJ algorithm proposed in this paper has been tested. The first and second row in Table 1 show the performance the malware detectors with modified NJ and TFIDF as dimensionality reduction process. The malware detector based on modified NJ algorithm performs better. TFIDF is to select obvious features and filter out non-critical features. This process may loss some feature information for detection. Some similar genes will be treated as two different features and map to two dimensions. However, these genes are actually closely related.

3.4 Machine Learning Models
The last component of the malware detector is the machine learning models. In the fourth group experiments, four widely-used machine learning models was selected and their performance has been compared, including Decision Tree(DT), Support Vector Machine(SVM), Linear Discrimination Analysis(LDA) and Random Forest(RF). The four models are based on different basic principles. Random Forest is ensemble learning model and performs the best among the four models. Therefore, Random Forest is finally selected as the machine learning model in our malware detector.

4. Conclusion
In this paper, a gene-inspired malware detection approach is proposed. The definition of software gene is put forward at first. It determines the boundary of software gene by whether the codes inside fully executed. Under the definition, the code fragment of gene can be described by a non-ambiguity sequence. The experiments also show that compared to n-gram method, gene slicing method generates much less data and performs better as well. An dimensionality reduction algorithm, modified Neighbour-Joining is introduced. The algorithm firstly clusters the genes in the data set by distance, then map each cluster to the feature vector. In this process, all information of gene is retained and have strong interpretability.

In the feature work, the precision of gene distance measure process will be further improved. The priori knowledge to build the scoring system for different mismatch will be obtain based on larger size of gene data. The computational efficiency of NJ algorithm will be sharply improved by parallelization.

References
[1] Tencent United Security Laboratory. https://slab.qq.com/news/auth ority/ 1708.html (2018)
[2] J.Drew, T. Moore, M. Hahsler. Security & Privacy Workshops, 00:81-87 (2016)
[3] J.Drew, M. Hahsler, T. Moore. EURASIP Journal on Information Security. 2017:2 (2017)
[4] D. Kirat, G. Vigna. ACM SigSac Conference on Computer and Communications Security. 2015:769-780 (2015)
[5] TF. Smith, MS. Waterman. Journal of Molecular Biology. 147 (1):195-197 (1981)
[6] X. Xia. Distance-based phylogenetic methods. Bioinformatics and the Cell. 343-379 (2018)
[7] S. Wold, K. Esbensen, P. Geladi. Chemometrics & Intelligent Laboratory Systems. 2 (1):37-52 (1987)
[8] G. Salton, A. Wong, CS. Yang. Communication of the ACM. 18:613-620 (1975)
[9] J. Pevsner. Bioinformatics and Functional Genomics. (2015)
[10] A. Aho, M. S. Lam, R. Sethi, J. Ullman. Compilers : Principles, Techniques & Tools (Second Edition) (2006)
[11] A. Elhadi, M. Maarof, A. Osman. American Journal of Applied Sciences 9 (3):283-288 (2012)
[12] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, Y. Elovici. Security Informatics, 1 (1):1 (2012)
[13] I. Santos, F. Brezo, X. Ugarte-Pedrero, PG. Bringas. Information Sciences, 231 (9):64-82 (2013)