MGeT: Malware Gene-Based Malware Dynamic Analyses

Jianwei Ding
mathe_007@163.com

Zhouguo Chen
czgexcel@163.com

Yue Zhao
yuezhao@foxmail.com

Hong Su
hongsu@163.com

Yubin Guo
gybglp@163.com

Enbo Sun
bo_lang_2008@163.com

Science and Technology on Communication Security Laboratory
Venture Road No. 8, High-tech Zone
Chengdu, China

ABSTRACT

Malware, as a malicious software, or applications or execution codes, has become the centerpiece of most security threats in such a unceasing open Internet environment. The essential technology of malware analysis is to extract the characteristics of malware, intended to supply signatures to detection systems and provide evidence for recovery and cleanup. The focal point in the malware analysis is how to detect malicious behaviors versus how to hide a malware analyzer from malware during runtime. In this paper, we propose an approach called Malware Gene Topology Model (MGeT) inspired by Biotechnological Genomics that can quickly detect potential malware from a large amount of software or execution codes including metamorphic or new variants of malware. Instead of extracting the signatures from the malware in the execution file level or operating system level, we identify the key malicious behaviors of malware by the underlying instructions, named malware Gene. We evaluate our method based on real-world datasets and the results demonstrate the advantages of our method over the previous studies, validating the contribution of our method.

CCS Concepts

• **Security and privacy → Malware and its mitigation**

Keywords

Security and Protection; Malware analysis; Malware Gene

1. INTRODUCTION

The last decade has witnessed tremendous growths of Internet technology such as cloud computing and big data technology. Numerous malicious computer software, applications and execution codes are able to spread widely, intended to damage host computer's operation, collect sensitive information, unauthorized access and so on, which are all defined as malware[1]. According to malware's attacking pattern[2], it mainly consists of Viruses, Trojan horses, Worms, Spyware, Backdoor, Logic bombs and so on in taxonomy.

ICCSP '17, March 17-19, 2017, Wuhan, China

DOI: http://dx.doi.org/10.1145/3058060.3058065

Although there are many computer security research communities and the related computer security industry making continuous endeavors in the last decades, intended to counter more and more malware threat, yet the number of new malware is still constantly on the rise. According to Chinese well-known security provider 360 security company's technological report[3], approximately 356 million new malware samples are reported just by its own security product in the year 2015, amounting to an average of nearly a million malware samples per day. Furthermore, the worldwide impact of malware can also be realized by the direct and indirect economic losses caused by malware, which are estimated to be more than 400 billion dollars just in the year 2014[4] and still growing. These figures not only depict the scale of the malware problem, but also give the malware developers more motivation to risk countering and evading the solutions provided by the computer security community for detecting and mitigating malware attacks. In other words, it will be a continuous arms race going on between malware developers and the computer security community. The malware analysis method proposed in this paper is also an attempt to be a part of this fight against the malware.

Most of the previous studies on malware analysis are mainly the signature-based methods[1] in which unknown files are searched for specific patterns, or signatures, particular to known malware, which are generated and collected by anti-malware providers after analysis of identified malware and delivered to client computers to update signature database in time. With the explosive growth of varieties of malware, the state-of-the-art malware analysis methods and tools faces more and more challenges. First, anti-analysis and anti-detection technology has a great development such as obfuscation, anti-debugging, anti-instrumentation, and anti-VM techniques [5], which make most of encrypted, polymorphic or metamorphic variants of existing malware become more and more difficult to analysis and detect. Second, most of present signature-based malware analysis methods are so passive that they cannot real-time detect a new type of malware. We analyze the reason of this situation is that, most of the signature-based static and dynamic characteristics almost depict the malware's behaviors in the operating system level, which are hidden or affected by varieties of system's operation, and cannot depict the essential key malicious behaviors in time. Third, signature generation is an often manually assist performed task[1]. Although there have many mature applications of machine learning techniques, such as classification[6], clustering[7], Hidden Markov Models (HMMs)[8], etc., are applied into identify categories of malware so that signatures are able to be generated

on basis category instead of not individual malware, yet keeping in view a large amount of new malware samples every day is still time-consuming for the security providers.

Given the aforementioned challenges, existing methods for malware analysis and detection do not work well. In this paper, we propose an approach called Malware Gene Topology Model (MGeT) inspired by Biotechnological Genomics that can quickly detect potential malware from a large amount of software or execution codes including metamorphic or new variants of malware. With the help of distribution hardware virtualization extension environment, we first extract a group of instruction behavior sequences (named DNA behavior sequence of the suspected malware sample) for each suspected malware sample. Then, we compare and extract essential malicious instruction behavior sub-sequences (named Gene behavior sequence of the suspected malware sample) from each instruction behavior sequence group of the suspected malware sample. The Gene Topology information of the malware depicts the essential malicious behavior sub-sequence, intended to reflect the real purpose of malware in the bottom of operating system. Meanwhile, the recombination and rearrangement of Gene behavior sequence can produce new type of Gene behavior sequence, which is able to detect new variants or new type of malware. The experimental results show that, our method outperforms the conventional methods when combining a large amount of malware samples.

Our work presents a Malware Gene Topology Model for malware analysis and detection, in the face of a large amount of malware samples. Our main contributions are shown as follows:

- We infer the Malware Gene Topology Model based on a large amount of malware samples, inspired by Biotechnological Genomics. The MGeT model incorporates a group of instruction behavior sequences for each malware sample, then compares and extracts the key malicious behavior sub-sequence in a distributed hardware virtualization extension environment, which is capable of depicting the real malicious purpose of the malware.
- We apply a time series method for the Malware Gene Topology Model's inference, by trying a pool of candidate instruction behavior sub-sequences from all possible malware sample's instruction behavior sequences' segments, and then sorting the top performing segments according to their target prediction qualities. The best discriminative instruction behavior sub-sequences are Gene of the malware.
- We implement the aforementioned methods in the environment of distributed hardware virtualization extension, which offers an external, transparent malware analyzer for instruction-granularity tracing against varieties of anti-analysis technologies effectively.
- We evaluate the effectiveness and efficiency of our method by conducting extensive experiments, using a real-world data set containing a large amount of malware samples. Our method clearly outperforms the baseline methods

The rest of the paper is organized as follows. After reviewing related work in Section 2, Section 3 overviews the notations and definitions in our method. We detail the methodology in Section 4 and evaluate our approach in Section 5. Finally, we conclude this paper in Section 6.

2. RELATED WORK

The key of malware analysis methods is to apply suitable mathematical methods and models to extract and express the characteristics of malware, which belongs to the scope of pattern recognition. Illustrated in Figure 1, According to the extraction means of malware characteristics, malware analysis methods mainly consist of two categories in taxonomy: static analysis methods, dynamic analysis methods and hybrid analysis.

Static analysis [9, 10], is a method that analyzing executable code of malware samples without executing the original file. As shown in Figure 1, it is just looking at the source code and uncovering the information such as the API call graph and byte sequence, which mainly extracted from PE file and assembly file of malware sample. In static analysis, the most basically used information consists of the API function, string information, binary sequence, N-gram sequence, control flow and so on. The advantage of static analysis methods is high coverage percentages of malware samples and low extraction time cost, and meanwhile the disadvantage of static analysis is that it cannot effectively conquer varieties of anti-analysis technologies such code obfuscation [5].

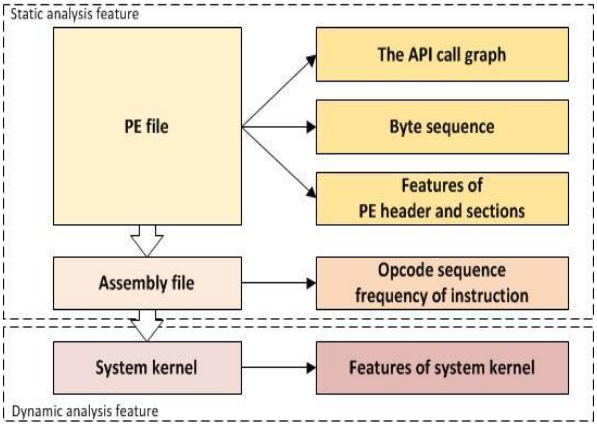

Figure 1. Sources of data collected for malware analysis.

Dynamic analysis[11, 12], is a method that monitoring and analyzing the malware's behaviors in the process of executing the file. The key difference between static analysis and dynamic analysis is if executing the malware sample. After execution of malware, we monitor its behavior and see how much it affects the host computer. Hence, dynamic analysis is also called behavioral analysis. In dynamic analysis, sandbox such as Norman Sandbox[13], simulator such as Anubis[14], and virtual machine such as Polyglot[15], are widely applied. The advantage of dynamic analysis is that it is capable of conquering varieties of anti-analysis technologies especially code obfuscation and the disadvantage is that high extraction time cost and big distinguishing difficulty between behaviors of malware and hosted computer's behaviors.

Hybrid analysis[16], is a hybrid methods combining both static and dynamic analysis, which incorporates the signature part from static analysis, and the behavioral part of dynamic analysis within it. As compared with the above two analysis, hybrid analysis is more efficient, but it also has to maintain the limitation of both static and dynamic analysis.

3. TERMINLOGY AND NOTATION

3.1 Terminology

As shown in Figure 2, a biological Gene sequence is corresponding to a type of biological species, and the same biological species have the homologous biological Gene sequence though they have different biological DNA sequences.

Similarly, as a computer software, its working mode is transformed to a piece of executing binary codes, intended to accomplish some mission or purpose such as calculation software for calculating, by means of a group of computer operation behaviors. An operation behavior sequence of software is analogy to a biological DNA sequence. Although the operation behavior sequences are not totally the same in every executing process of software, yet the essential functions of software are the same, which means that the key operation behavior sub-sequences are the same, intended to accomplish the same function. The key operation behavior sub-sequences are defined as **the Gene of the software**.

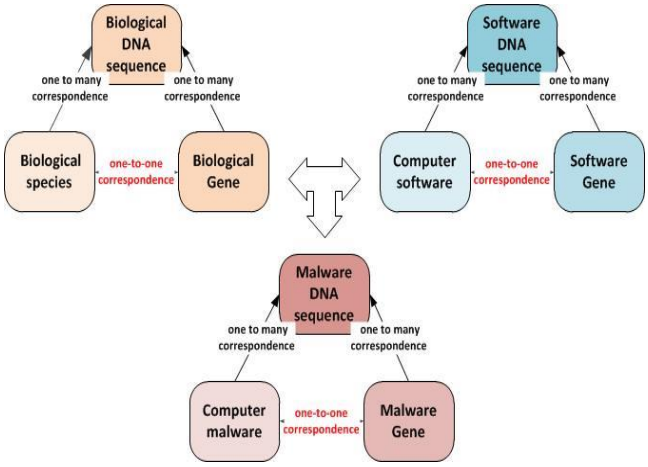

Figure 2. Relationship analogies among biological species, computer software and malware.

Furthermore, as a type of software with malicious purpose, every malware's executing process will produce an operation behavior sequence (named **malware DNA sequence**), which are similar to the biological DNA sequence. The key malicious operation behavior sub-sequences are **the Gene of malware** such as collecting sensitive information or authorized access, which depict the real malicious purpose from the computer instruction's perspective. The Gene of malware is corresponding to a type of basis malware, which consists of many variants samples associated with different malware DNA sequences.

3.2 Notation in Our Proposed Method

We model a program's execution at the machine instruction level, denoted as **P**, which is able to access memory and CPU directly. Hence, a system state consists of the contents of memory and CPU registers, and then let $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{I}$ denote the set of all possible memory states, the set of all possible CPU states and all possible CPU instructions, respectively. Especially, each instruction i($i \in \mathbf{I}$) can be considered a machine recognizable combination of opcode and operands stored at a particular address in memory, which defines how it updates the memory and CPU states. Formally, an execution of an instruction is defined as a transition function $\sigma: \mathbf{I} \times \mathbf{M} \times \mathbf{C} \rightarrow \mathbf{I} \times \mathbf{M} \times \mathbf{C}$, incorporating the machine-level execution semantics. Furthermore, a program's execution is defined as an instruction behavior sequence $P = \{i_1, \dots, i_n\}$, where $\sigma_P(i_l, m_l, c_l) = (i_{l+1}, m_{l+1}, c_{l+1})$. In this paper, we will use integers to denote the instruction behaviors in the instruction set, with each instruction i taking a value from $1, \dots, I$, where i is the number of unique instruction in the instruction set **I**\footnote{Actually, the CPU instruction set is finite set.}.

Malware Dataset A malware dataset consists of malware samples, which is also composed of N malware DNA sequences, and for notation ease we assume that each sequence contains Q-many ordered values, even though our model also works on variable sequence lengths. Hence, the malware dataset is denoted as $I^{N \times Q}$, where the sequence's target is a nominal variable $Y \in \{1, \dots, C\}^N$ having C categories.

Malware Gene A malware Gene of length L($L \ll Q$) is an ordered sub-sequence of values extracted from the malware DNA sequence, which semantically depicts intelligence on how to discriminate the target variable of a malware dataset. The \$K\$-most informative malware Gene are denoted as $G \in R^{K \times L}$.

Distances Between Malware Gene and Malware DNA sequence The distance between the n-th sequence P_n and the k-th Gene G_k is defined as the minimum distance[17] $M_{n,k}$ (illustrated in Equation 1), intended to compare the distances between the Gene G_k and each sub-sequence $j(j = 1, \dots, J; J =: Q - L + 1)$ of P_n. Intuitive speaking, it is the distance of a Gene to the most similar sub-sequence of P_n.

$$M_{n,k} = \min_{j=1,\dots,J} \frac{1}{L} \sum_{l=1}^{L} (P_{n,j+l-1} - G_{k,l})^2 \qquad (1)$$

4. TERMINLOGY AND NOTATION

4.1 Malware Gene Topology Model

Illustrated as Figure 3, our proposed MGeT model first extracts malware DNA sequence at the execution of malware samples, and then searches the best K malware DNA sub-sequences as malware Gene with the calculation of equation 1. The calculation of distances between the Gene candidates and each DNA sub-sequences is too time-consuming. Hence, we iteratively optimize the malware Gene by minimizing a classification loss function (shown in Equation2), instead of searching among possible malware Gene candidates from all the malware DNA sub-sequences.

Minimum distances to malware Genes can be regarded as a transformation of the malware DNA sequence $P \in R^{N \times Q}$ into a new representation $M \in R^{N \times K}$, intended to reduce the dimensionality of the original malware DNA sequence as typically $K < Q$. Based on this idea, we apple a linear learning model to discriminate best category values $\hat{Y} \in R^{N \times K}$, via the distances $M_{n,k}$ associated with linear weights $W \in R^K$ (plus bias $W_0 \in R$), as shown in Equation 2.

$$\hat{Y} = W_0 + \sum_{k=1}^{K} M_{n,k} W_k \qquad (2)$$

For the sake of explanation, the model introduced in this Section will be focused just on binary labels $Y \in \{0,1\}^N$ associated with a fixed malware Gene length L. Actually, we can convert the binary classification problem to C-many one-to-all sub-problems[18]. Hence, we apply a logistic regression classification model, intended to estimate predicted binary labels as probabilistic confidences. The logistic regression model operates by minimizing the logistic loss, defined in Equation 3, between true labels Y and estimated ones $\hat{Y}$.

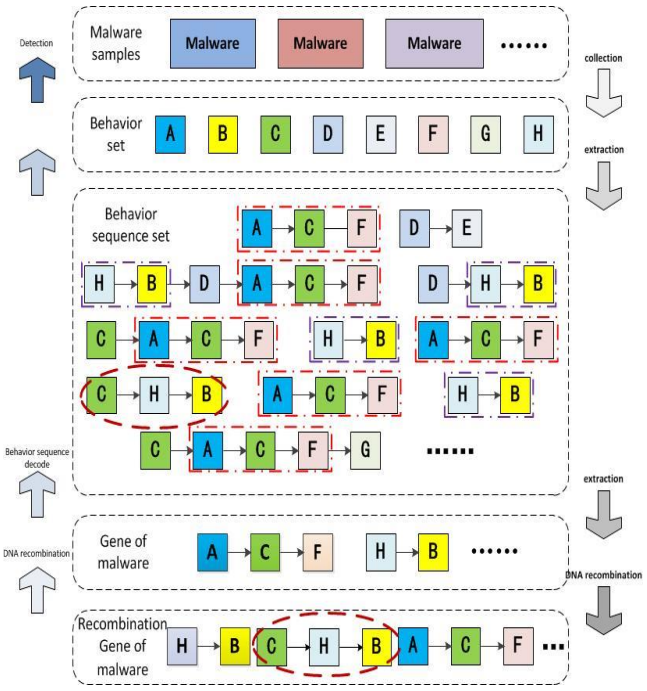

Figure 3. The illustration of Malware Gene Topology Model.

$$\mathcal{L}(Y,\hat{Y}) = -Y\ln\varphi(\hat{Y}) - (1-Y)\ln(1-\varphi(\hat{Y})) \quad (3)$$

In Equation 3, the logistic function $\varphi(Y) = (1+e^{-Y})^{-1}$. The logistic loss functions $\mathcal{L}(Y,\hat{Y})$ together with regularization terms represent the regularized objective function, denoted as $\mathcal{F}$ in Equation 4. The idea of this paper is to jointly learn the optimal malware Gene G and the best linear hyper-plane W that minimize the classification objective $\mathcal{F}$.

$$\arg\min_{G,W} \mathcal{F}(G,W) = \arg\min_{G,W} \sum_{n=1}^{N} \mathcal{L}(Y_n, \hat{Y}_n) + \lambda_W ||W||^2 \quad (4)$$

4.2 The Inference of MGeT

The inference of MGeT model is to infer the Equation 4. In this paper we apply a stochastic gradient descent approach on the decomposed objective function $\mathcal{F}_n$ shown in Equation 5, corresponding to a division of the objective of Equation 4 into per-instance losses for each malware DNA sequence.

$$\mathcal{F}_n = \mathcal{L}(Y,\hat{Y}) + \frac{\lambda_W}{N}\sum_{k=1}^{K} W_k^2 \quad (5)$$

In Equation 5, there are two variables G and W, and then we need calculate the gradients of the objective function with the respect to them.

The gradient for malware genes $G_{k,l}$ First, the gradient of point l in malware Gene k with respect to the objective of the n-th malware DNA sequence is calculated in Equation 6, which is derived through the chain rule of derivation.

$$\frac{\partial \mathcal{F}_n}{\partial G_{k,l}} = \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial \hat{Y}} \frac{\partial \hat{Y}_n}{\partial M_{n,k}} \sum_{j=1}^{J} \frac{\partial M_{n,k}}{\partial D_{n,k,j}} \frac{\partial D_{n,k,j}}{\partial G_{k,l}} \quad (6)$$

In Equation 6, the variable $D_{n,k,j}$ represents the distance between the j-th sub-sequence of the n-th malware DNA sequence and the k-th malware Gene, as shown in Equation 7. Meanwhile, because $\frac{\partial M}{\partial G}$ is actually not differentiable, then we use a soft minimum function (shown in Equation 8) to approaches the real M when the parameter α approaches to∞.

$$D_{n,k,j} := \frac{1}{L}\sum_{l=1}^{L}(P_{n,j+l-1} - G_{k,l})^2 \quad (7)$$

$$M_{n,k} \approx M_{n,k} = \frac{\sum_{j=1}^{J} D_{n,k,j} e^{\alpha D_{n,k,j}}}{\sum_{j=1}^{J} e^{\alpha D_{n,k,j}}}$$

Furthermore, the gradient of the loss with respect to the predicted label $\frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial \hat{Y}}$ is calculated in Equation 9, and the gradient of the minimum distances with respect to the estimated labels $\frac{\partial \hat{Y}_n}{\partial M_{n,k}}$ is illustrated in Equation 10. The gradient of the overall minimum distance with respect to a sub-sequence distance $\frac{\partial M_{n,k}}{\partial D_{n,k,j}}$ is presented in Equation 11 and the gradient of a sub-sequence distance with respect to a malware Gene point $\frac{\partial D_{n,k,j}}{\partial G_{k,l}}$ is derived in Equation 12.

$$\frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial \hat{Y}} = -(Y_n - \varphi(\hat{Y}_n)) \quad (9)$$

$$\frac{\partial \hat{Y}_n}{\partial M_{n,k}} = W_k \quad (10)$$

$$\frac{\partial M_{n,k}}{\partial D_{n,k,j}} = \frac{e^{\alpha D_{n,k,j}(1+\alpha(D_{n,k,j}-M_{n,k}))}}{\sum_{j=1}^{J} e^{\alpha D_{n,k,j}}} \quad (11)$$

$$\frac{\partial D_{n,k,j}}{\partial G_{k,l}} = \frac{2}{L}(G_{k,l} - P_{n,j+l-1}) \quad (12)$$

Gradients for classification weights W The hyper plane weights W can also be inferred by minimizing the classification objective via stochastic gradient descent. Equation 13 shows the partial gradient of updating each weight W_k and Equation 14 presents the bias term W_0.

5. EVALUATION

We conducted the experiments on malware datasets used in [19], which consists of N = 1014 malware instances, divided into eight malware families (C = 8).

5.1 Evaluation Baseline and Metrics

In the experiments, we evaluated three groups of baseline experiments shown as follows:

- **HMM[19]**: applied a hidden markov model (HMM) on the system call graph extracted from the malware samples, intended to detect the predicted category of malware (its hidden states).
- **N-GRAM[20]:** used n-grams of system calls in their two-step process comprising prototype-based clustering followed by nearest prototype classification.
- **DTW-SVM**: we applied dynamic time wrap to calculate the distance between two malware DNA sequences, and then used SVM to classify different type of malware samples.

A 10-fold cross-validation approach was used to compare the methods. The iterations were formed in a statistically balanced manner across different categories of malware samples. Meanwhile, we evaluate three standard performance metrics: precision, recall and F-measure. The experiment mainly inspects the following four evaluation indexes:

$$\textbf{precision} = \frac{\#|\boldsymbol{True\ positives}|}{\#|\boldsymbol{True\ positives}|+\#|\boldsymbol{Flase\ positives}|} \quad (15)$$

$$\textbf{recall} = \frac{\#|\boldsymbol{True\ positives}|}{\#|\boldsymbol{True\ positives}|+\#|\boldsymbol{False\ negatives}|} \quad (16)$$

$$\mathbf{F-measure} = \frac{2\times precision\times recall}{precistion+recall} \quad (17)$$

5.2 Experiment Results

All evaluation results for the datasets are presented in Figure 4. As highlighted in figures, **MGeT** consistently outperforms the other approaches for all the three metrics(precision, recall, and F-measure), demonstrating the importance and effectiveness of extracting malware Gene sequence among a large amount of malware DNA sequences.

Furthermore, we conducted the experiment on parts of the data sets to confirm that the malware Gene sequence does indeed affect the performance of malware detection. For the sake of explanation, we only explain the results of precision of the four methods in Figure 5; similar results of other metrics are observed in Figure 6 and 7. In particular, the precision on 20% of datasets on the MGeT are obviously larger than that of other three baselines, which demonstrates that our proposed MGeT indeed is more capable of detecting new variants of malware or new type of malware. What's more, with the volume of datasets increases, the performance of MGeT consecutively outperforms the other three methods on all the three metrics, and increases with the increase of the datasets volume.

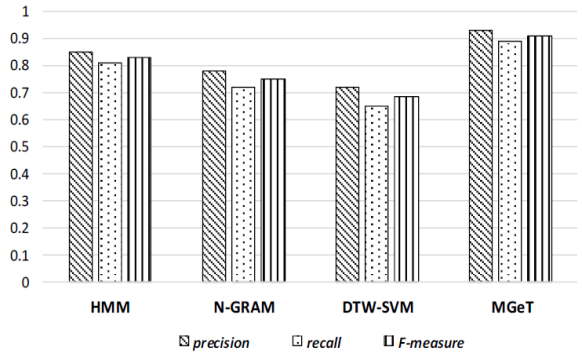

Figure 4: Performance of various approaches.

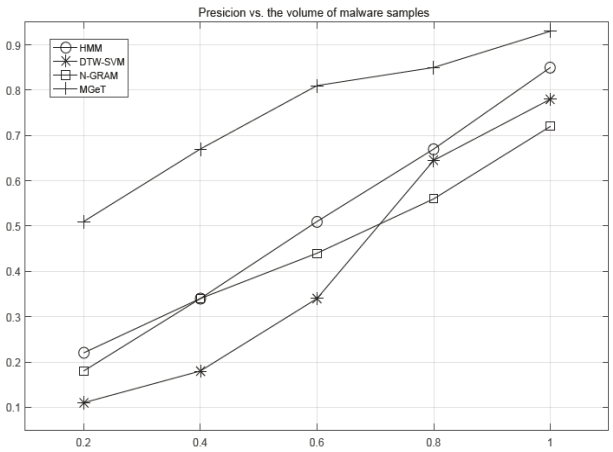

Figure 5: Precision vs. the volume of datasets.

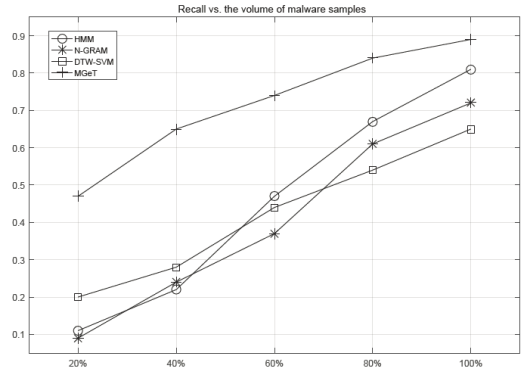

Figure 6: Recall vs. the volume of datasets.

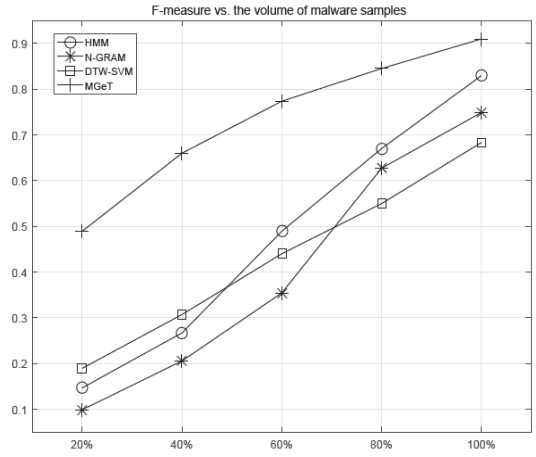

Figure 7: F-measure vs. the volume of datasets.

6. SUMMARY

Rapid detection of potential malware in a vast amount of malware candidates is important to the malware analysis. In this paper, we modeled the key malware instruction behavior sequences using the **MGeT** model, which is able to discriminate the malware instances even though new variants or new type of malware. Employing the MGeT model has promising performance in experimental studies, which leads us to conclude that our proposed MGeT can indeed detect more potential malware instances than previous approaches. Our future work includes model optimization, model training, and experiments on different data sets. In addition, further analysis of the malware Gene pattern detected by MGeT is an interesting topic worthy of further study.

7. REFERENCES

[1] Egele, M., Scholte, T., Kirda, E., and Kruegel ,C. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR), 44(2):6, 2012.

[2] Szor, P. The art of computer virus research and defense. Pearson Education, 2005.

[3] Company, S. Internet security report of china in the year 2015. http://www.360doc.com/content, 2015.

[4] McAfee, N. L. Estimating the global cost of cybercrime. economic impact of cybercrime ii, mcafee, junio de 2014.

[5] Moser, A., Kruegel, C., and Kirda, E. Limits of static analysis for malware detection. In Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual, pages 421–430. IEEE, 2007.

[6] Ye, D. Wang, T., Li, D. Y., and Jiang, Q. An intelligent pe-malware detection system based on association mining. Journal in computer virology, 4(4):323–334, 2008.

[7] Ramadass, S. Malware detection based on evolving clustering method for classification. Scientific Research and Essays, 7(22):2031–2036, 2012.

[8] Annachhatre, C., Austin, T. H., and Stamp, M. Hidden markov models for malware classification. Journal of Computer Virology and Hacking Techniques, 11(2):59–73, 2015.

[9] Kong, D. and Yan, G. Discriminant malware distance learning on structural information for automated malware classification. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1357–1365. ACM, 2013.

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

[11] Anderson, B., Quist, D., Neil, J., Storlie, C., and Lane, T. Graph-based malware detection using dynamic analysis. Journal in Computer Virology, 7(4):247–258, 2011.

[12] Nari, S., and Ghorbani, A. A. Automated malware classification based on network behavior. In Computing, Networking and Communications (ICNC), 2013 International Conference on, pages 642–647. IEEE, 2013.

[13] Sandbox, N. Norman sandbox whitepaper, 2010.

[14] Bayer, U., Kruegel, C., and Kirda, E. Anubis: Analyzing unknown binaries, 2009.

[15] Caballero, J., Yin, H., Liang, Z., and Song, D. Polyglot: Automatic extraction of protocol message format using Dynamic binary analysis. In Proceedings of the 14th ACM conference on Computer and communications security, pages 317–329. ACM, 2007.

[16] Elhadi, A. A. E., Maarof, M. A., and Osman, A. H. Malware detection based on hybrid signature behaviour application programming interface call graph. American Journal of Applied Sciences, 9(3):283, 2012.

[17] Ye, L., and Keogh, E. Time series shapelets: a new primitive for data mining. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 947–956. ACM, 2009.

[18] Hou, L., Kwok, J. T., and Zurada, J. M. Efficient learning of time series shapelets. In Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[19] Imran, M., Afzal, M. T., and Qadir, M. A. Using hidden markov model for dynamic malware analysis: First impressions. In Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on, pages 816–821. IEEE, 2015.

[20] Rieck, K., Trinius, P., Willems, C., and Holz, T. Automatic analysis of malware behavior using machine learning. Journal of Computer Security, 19(4):639–668, 2011.