I.J. Intelligent Systems and Applications, **2016, 4, 18-25**
Published Online April 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2016.04.02

Detection of Metamorphic Malware based on HMM: A Hierarchical Approach

Mina Gharacheh [1]
[1] School of Electrical and Computer Engineering, University of Science and Arts, Yazd, Iran
E-mail: gharache@stu.sau.ac.ir

Vali Derhami [2], Sattar Hashemi [3], Seyed Mehdi Hazrati Fard [4]
[2] School of Electrical and Computer Engineering, Yazd University, Yazd, Iran
[3, 4] School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran
E-mail: [2] vderhami@yazd.ac.ir, [3] s-hashemi@shirazu.ac.ir, [4] hazrati@cse.shirazu.ac.ir

***Abstract*—**Recent research have depicted that hidden Markov model (HMM) is a persuasive option for malware detection. However, some advanced metamorphic malware are able to overcome the traditional methods based on HMMs. This proposed approach provides a two-layer technique to overcome these challenges. Malware contain various sequences of opcodes some of which are more important and help detect the malware and the rest cause interference. The important sequences of opcodes are extracted by eliminating partial sequences due to the fact that partial sequences of opcodes have more similarities to benign files. In this method, the sliding window technique is used to extract the sequences. In this paper, HMMs are trained using the important sequences of opcodes that will lead to better results. In comparison to previous methods, the results demonstrate that the proposed method is more accurate in metamorphic malware detection and shows higher speed at classification.

***Index Terms*—**Malware detection, metamorphic malware, hidden Markov model.

I. Introduction

Todays, malware is considered a serious threat for personal data security and computer systems and creating large-scale failures. Considering the growth of malware generation and also the various and complex techniques being used by malware designers, such as obfuscation in which the malware keeps its functionality while changing its structure, the significance of malware detection comes before that of taking any other serious action [6]. Malware is a massive form of malicious software and includes Viruses, Worms, Trojans, Adwares, Spywares, etc. [4]. There are two major trends for malware detection: traditional signature-based methods and behavior-based methods.

A malware's signature, which is utilized by the signature-based method to detect the malware, is something like a fingerprint. In other words, it is a unique characteristic and is the sequence of bytes comprising the malware, which should be found in the same malware to maintain its monopoly. Malware writers sought for ways to get around this method. Having considered the intention, polymorphic malware were created. They don't have fixed, unchanging, codes and they encrypt their code by an algorithm per infection; they decrypt the same code during the runtime. The next generation of malware did not even have the fixed encoding and decoding engine and they generally transform their code per spread, so that it is very difficult to recognize them through the traditional signature-based methods and this calls for creating complicated signatures; sometimes detection is impossible. These types of malware are called Metamorphic.

The behavior-based methods are also able to detect malware that use obfuscation techniques, analyzing the programs' behavior. Most methods proposed to achieve this objective use modeling and data mining to detect malware, during which a set of features are extracted from the malware files and an effort is made to learn the malware's behavior using machine learning algorithms, then a model of destructive behavior is created by which malware are separated from benign programs. One of the effective ways in this field is to use HMM in metamorphic malware detection. The research indicates that this method has led to more desirable results [11].

However, some advanced metamorphic malware are able to overcome traditional detection methods based on HMMs. The proposed approach provides a two-layer technique to overcome these challenges. Thus, completely benign files are detected and removed at high speed in the first layer, using the threshold approach, and the rest of the files are sent to the second layer for more accurate identification.

This paper is organized as follows. Section 2 is an overview of the most important previous works based on HMM. Section 3 discusses our proposed method in more detail. In Section 4, experiment results of our proposed method based on HMM are presented. Finally, Section 5 wraps the whole paper up by giving a conclusion.

II. Related Work

In this section, HMM-based malware detection works are reviewed. HMM was introduced in the late 1960s and now it is expanding the range of its applications rapidly. HMMs can have various applications in the field of modeling and learning and they are most well-known for pattern recognition, such as recognizing voice and handwriting, recognizing points and movement, labeling speech and Bioinformatics [7].

Several HMMs are being trained based on phonemes in speech recognition as a solution and the input signal similarity is computed by the trained models. In this solution, the input signal is divided into fixed-sized, overlapping, frames and each frame is analyzed by HMMs and is classified as a phoneme [8]. The solution can be applied, with little change, to detect metamorphic malware.

It is challenging to classify malware automatically. Annachhatre et al used HMM and cluster analysis to solve the problem. They evaluated the HMMs according to a scoring technique and they clustered by K-mean algorithm and achieved acceptable results [2].

The creators of metamorphic viruses endeavor to frustrate methods based on HMM, using dead benign codes. Vinod et al presented a new approach to detect unseen malware and benign samples, using the discriminating linear analysis to rank and produce the most prominent features of opcode which can improve detection rate compared to general scanners [5].

Wong et al utilized the threshold approach to distinguish malware, using HMM successfully. They demonstrated that the approach presents a practical solution for some metamorphic viruses which cannot be detected by the signature-based method. In this method, the opcode sequence is considered unique in the destructive software and is learned by the HMM. Then the probability of observing the opcode sequences in every new file is examined and determined based on its similarity with the sequences learned from this model. It is classified as virus if the probability is more than the determined threshold, otherwise it is classified as benign [12]. The most important benefit of this method is that the analysis is performed at a high speed, as it uses just a single HMM. However, some of the metamorphic malware such as MWOR are able to escape this detection approach.

Kalbhor et al used the dual HMM as a tool to detect metamorphic viruses and they could detect them with a high precision. However, the approach causes much overload and the identification time is about twice the threshold approach, as it creates a separated model for each virus family and each family of benign files rather than using only one HMM. Then, for each file it computes the probability of observing opcode sequences based on each model separately and matches the file to the family that the model corresponding to it has the highest probability [11].

III. Proposed Method

In recent years, different approaches have been developed using some parts of executable files to detect malware. Although every part of an executable contains important information about the file, all of the information does not facilitate the detection of destructive behavior. The sequences of opcodes extracted from an executable file explain the file's behavior and can be shown as a set of simpler tasks through a few opcodes [3]. Every unique sequence of opcodes makes up part of an executable file that lead to a specific behavior which, in some cases, could be similar to the resulting behavior of another unique sequence.

A malware program includes various sequences of opcodes, some of which facilitate detection and the rest just interfere with and hinder detection. As a result, if commands could be separated from each other and the HMM be trained based on the important commands, it would lead to better results. However, the problem is that we are not aware of the locations of important sequences of opcodes in the malware file.

The proposed method uses less important sequences of opcodes in malware to extract the important sequences. It can be done based on the fact that less important sequences of opcodes are the ones which have more similarity to benign files.

Therefore, we break down malware into overlapping parts equal in size and call each new part a frame. In fact, each frame unique sequences of opcodes. Afterwards, each frame is fed to the HMM, which is trained based on benign files, as input in order for the HMM to score the extent of similarity. In other words, the similarity of each frame to benign files is evaluated and we get rid of the frames which show higher similarity because of the fact that such frames bear less significance in the process of malware detection. Thus, the remaining frames are those containing sequences with higher significance. Consequently, by training an HMM based on these more important sequences, we would be able to detect malware faster and more accurately.

As mentioned previously, HMM is a statistical model that using statistical features of signals calculates the similarity. In this proposed method, the signal comprises of the sequences of malware opcodes based on which the model is trained.

The presented method requires explaining three fixed coefficients: the size of the frames, the importance threshold and the classification threshold. The frame size is the opcode sequence length extracted from the destructive software. Thus, if the frame size is considered three, each extracted frame includes only three opcodes. In this method, we use the sliding window technique to extract the frames [10]. The importance threshold is the cutting point by which the unique and important part of the malware is separated from the less important part. If the importance threshold is zero, which is very large, malware code is extracted completely and if it is selected very small, only a small portion of unique sequences of opcodes is extracted. The classification threshold is the cutting point which determines if the existing files in the test set should be classified as malware or benign. In the following sections the process of signal production and each of these constants are explained in length.

In order to reduce the time spent on file classification, this method exploits two different layers which are both based on HMM; in a way that the first layer, using the threshold approach, detects files which are certainly benign and gets rid of them, resulting in an increased efficiency. And then, the rest of the files, the classification of which are more difficult, are sent to the second layer for a more precise evaluation. What follows describes these two layers.

A. The First Layer

As already explained, the first layer using the threshold approach excludes the certainly benign files from the set which lack any similarity to malware files. The similarity of each frame to benign files is evaluated only in order to exclude the files which are more similar to benign files and contain less important sequences, ultimately to detect important sequences of malware files. Therefore, benign files which show more similarity to malware files are required. If such files could be detected and the similarity of each frame to them could be evaluated, we would, certainly, reach more accurate results and the time spent on classification would also be decreased. It is the first layer that enables such improvement.

The threshold approach proposed in [12] trains an HMM based on malware files. Then, using this trained HMM, every benign or malware file in the test set is scored. Afterwards, a threshold is determined to separate benign files from malware files which is compared against the score that each file has received. If a files scores above the threshold, the file is classified as malware otherwise as benign.

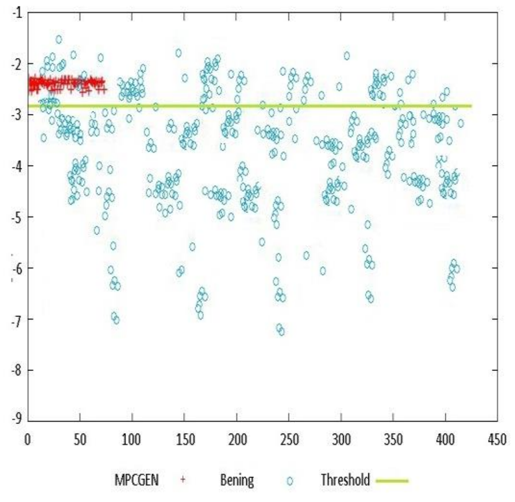


Fig.1. First layer performance for MPCGEN family with -2.92 as threshold

This proposed method makes a few changes to the Wong approach in a way that for each family of malware an HMM is trained and the threshold is determined in such a way that files scoring below that number are classified as certainly benign. Fig. 1 indicates the performance of the threshold approach for MPCGEN malware family.

As can be seen in the Fig. 1, the files below the threshold are certainly benign and bear no resemblance to MPCGEN family and it can be asserted the play no role in the extraction of important sequences. Thus, they can be excluded and the rest of the files are ready to be sent to the next layer.

B. The Second Layer

The second layer aims at distinguishing different malware commands and training an HMM based on significant commands in order to acquire better results. The significance or importance of commands is evaluated based on their lack of similarity to benign files because not every part of malware program does not signify destructive essence and this justifies the elimination of certainly benign files in the first layer. A brief explanation of the process of training and classification in the second layer is presented:

Training:

1. Benign files are converted into a signal and an HMM is trained based on that. This model is capable of calculating the extent of similarity between input files and benign files.
2. Malware files are also converted into a signal and the signal is broken into overlapping parts. For this purpose, a frame size is determined and all frames are extracted from the signal.
3. Then, every frame is fed to the model created in step 1 as input in order to determine the similarity of each frame to benign files.
4. Using the defined threshold, more important opcode sequences are separated from the less important ones. In other words, frames with higher similarity to benign files are excluded from the set and consequently frames with more important opcode sequences remain in the set.
5. Then, a new HMM is trained using the resulting important sequences. This model enables accurate evaluation of the similarity between input files and malware files.

Classification:

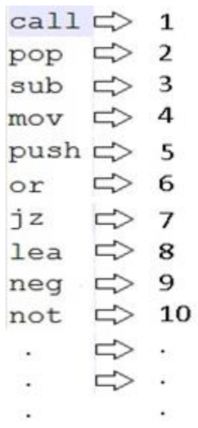
1. Malware and benign files remaining in the test set are fed to the new HMM which trained only on specific parts of malware in order to determine the similarity of each file to malware files.
2. Then, exploiting the defined classification threshold, benign files are separated from malware and so they are classified.

Acquired results show that the proposed method features high speed and accuracy in metamorphic malware detection.

C. Signal Generation Process

As mentioned, the proposed model is created and trained based on the input signal that it receives and the input signal is actually the opcode sequences of benign and malware files.

In order to generate the signal based on opcode sequences, it is required that all unique opcodes in the whole data set be detected and then each opcode be assigned a corresponding unique number. Fig. 2 shows the manner in which this operation is done.



call	1
pop	2
sub	3
mov	4
push	5
or	6
jz	7
lea	8
neg	9
not	10
.	.
.	.
.	.

Fig.2. Mapping opcodes to unique numbers in order to generate the signal

Next, the opcode sequences of the relating file is ascribed with these numbers and, as seen in Fig. 3, the signal of the file is generated.

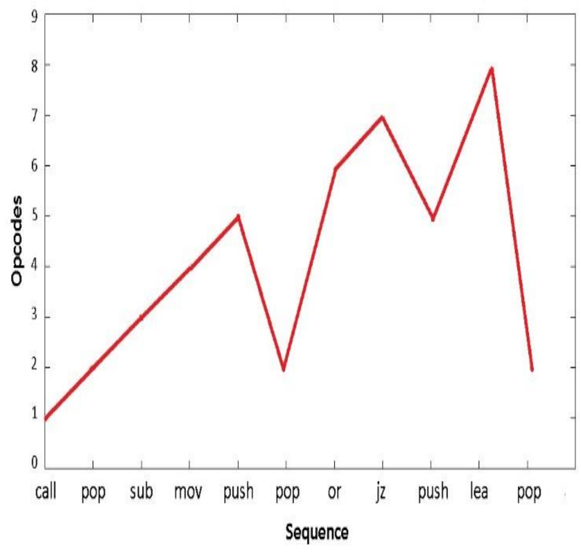


Fig.3. A sample signal generated from opcode sequences

Fig. 4 is an actual signal generated from opcode sequences of NGVCK malware family files and as is apparent, the length of the signal is about 9000 meaning that it comprises of 9000 opcodes which include about 120 unique opcodes.

D. Classification Threshold

To determine the value of the threshold, first the number of false positives and the number of false negatives of each model are to be calculated. Next, these numbers are to be studied according to different values of the threshold and, depending on the desired balance, the value of the threshold is determined [12].

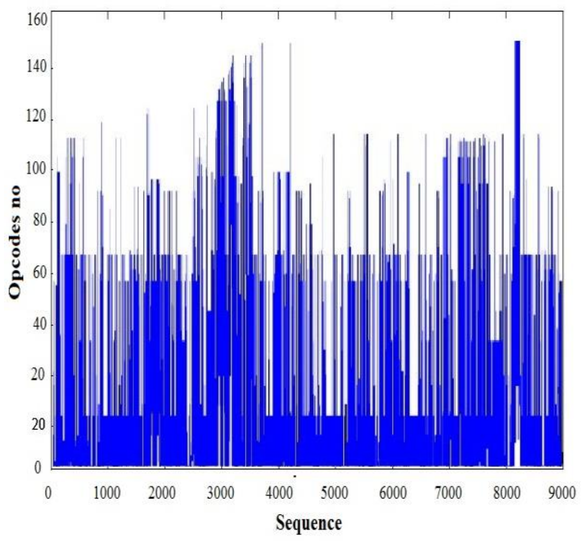


Fig.4. The signal generated from opcode sequences of NGVCK malware family

The classification threshold is a value using which the files in the test set are classified as benign or malware. If the value is too low, most of the files are classified as benign and, obviously, if the value is too high, the opposite happens. The threshold value is calculated by experimenting with different values on the test data, through the explained process in the previous section.

E. Significance Threshold

The significance threshold is the cutting point using which less important opcode sequences are separated from the more important ones in malware files. If the value of the significance threshold is too big, malware code is extracted completely which is undesirable and if the value is too small only small part of the unique opcode sequences are extracted which is also undesirable. Fig. 5 gives a view to the process.

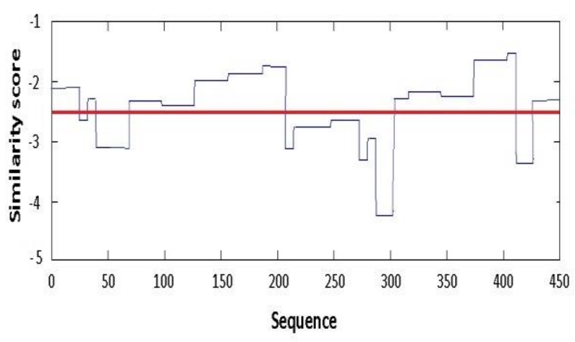


Fig.5. Performance of significance threshold with the value of -2.5

The sequences that fall below the threshold are those with less significance that are more similar to benign files and bring about interference in detection. Thus, such sequences are eliminated and important sequences above the threshold are separated and packed together to create a new signal.

F. Frame Size

As previously mentioned, to extract important sequences of malware, files are first to be converted into a signal. Then, the signal is broken into overlapping parts, called frames, which are equal in size. In fact, every frame contains a unique opcode sequence and the size of the frame determines the length of each sequence extracted from the malware. So that if the size is assumed to be 50, each extracted sequence contains only 50 opcodes.

This method exploits a technique similar to the sliding window technique because the frames need to overlap. This is due to the fact that we do not know which part of sequences are the important ones that we are looking for. Therefore, by having overlapping frames, the chance of finding more significant frames increases. For this purpose, after converting malware files into signals, we break them down into overlapping frames equal in size. Fig. 6 shows how the process is done with frame size of 50.

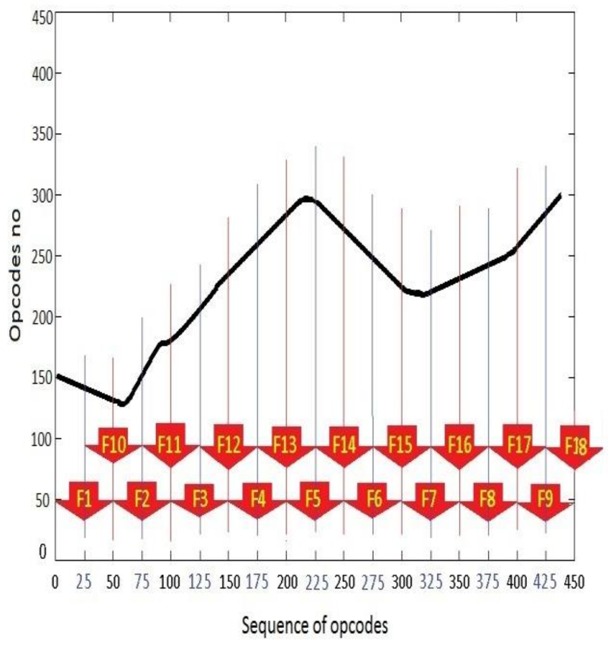


Fig.6. Signal segmentation with frame size of 50

As seen in Fig. 6, a signal with the length of 480 is broken into 18 frames with the size of 50. If the size is too high, for example if the frame size is 450 in Fig. 6, only one frame is extracted from malware file which will, certainly, not prove effective at all in the results.

IV. Experimental Results

We experiment with three different approaches, the threshold approach, the dueling approach and our proposed approach. We also compare the results of the three approaches in terms of execution time, as well as in terms of detection rate, false-positive rate and overall accuracy.

The detection rate equals the number of viruses detected by the model divided by the total number of viruses in the test set. The false-positive rate, which is related to model features, is obtained through dividing the number of false positives by the total number of benign programs in the test set. The overall precision is defined as the number of correct predictions divided by the total numbers of benign and non-benign programs [12]. These three criteria are computed based on true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) as follows:

$$Detection\ Rate = \frac{TP}{TP+FN} \tag{1}$$

$$False\ Positive\ Rate = \frac{FP}{FP+TN} \tag{2}$$

$$Overall\ Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

We look at each of these in detail in the following sections.

A. Data Sets

The proposed method is examined and tested on the well-known malware MWOR[19] and the malware constructed by kits NGVCK [13], G2[14], MPCGEN[15], MetaPHOR[16] and the random benign executable files which are selected from the Clang [18], Cygwin [19], GCC [20], MingW [21], TASM [24], Turbo C [25]. The total number of malware and benign files in the training and test sets together is 1245.

Classifying a file requires the computation of the probability of its opcode sequence similarity in a model. But the computation is not possible directly as it needs about $2TN^T$. The strength of HMM is due to the fact that there exists an efficient algorithm to achieve the same results. We calculate the score by running the forward HMM algorithm which only needs N^2T multiples [9].

B. Threshold Method

In this method, we train an HMM for every malware family code, then, determine the appropriate threshold value. Afterwards, for any file in the test set, we determine the probability of observing given sequences of opcodes.

If the probability is less than the threshold value, the file is classified as a malware; otherwise it is classified as benign [12].

Table 1. Results for Threshold Approach

	FP %	FN %	Threshold	Detection Rate	FP rate	Overall accuracy
G2	15.29	8.33	-2.71	0.9863	0.54	0.8557
MPCGEN	21.64	3.33	-2.92	0.9940	0.61	0.8062
NGVCK	39.76	9.00	-3.58	0.9343	0.48	0.7008
MetaPHOR	47.29	15.0	-2.66	0.9372	0.70	0.5886
MWOR	12.23	19.0	-2.57	0.9515	0.39	0.8648

Results in Table 1 show that this approach was especially incompetent to detect MetaPHOR and NGVCK. Our results are different from Wong’s [12] because we use a different dataset and train an HMM for each malware family, instead of training an HMM for the whole malware set.

C. Dueling Method

The dueling HMM strategy differs from threshold approach. In this approach, an HMM is trained for each malware family code and also some HMMs are built based on benign codes. For every file in the test set, we determine the probability of observing the sequence of opcodes for each of the malware HMMs and benign HMMs. If the HMM reporting the highest probability represents malware code, the file is classified as a malware [11].

Table 2. Results for the Dueling Approach

	FP %	FN %	Detection Rate	FP Rate	Overall accuracy
G2	0.00	0.00	1.0000	0.00	1.0000
MPCGEN	0.00	0.00	1.0000	0.00	1.0000
NGVCK	2.82	38.00	0.8446	0.08	0.8592
MetaPHOR	3.29	21.00	0.9514	0.15	0.9333
MWOR	1.64	2.00	0.9952	0.06	0.9829

Table 2 shows that the results improve considerably in terms of overall accuracy. However, as a side effect of each file being scored against multiple HMMs the files take longer to be classified.

D. Proposed Approach

In the proposed approach we use the threshold approach. The aim is to eliminate as many files as can be eliminated using the threshold approach (Tier1), thus the proposed approach would need (Tier2) to classify smaller number of files, leading to a considerable gain in terms of execution time in file classification.

We first use the threshold approach to score the files and eliminate any “definitely benign” file; files which score below the threshold .The threshold is different for different malware code.The files which cannot be eliminated using the threshold approach are then scored using our proposed approach. Table 3 shows the results of this approach.

Table 3. Results for Proposed Approach

	FP %	FN %	Threshold	Eliminated in Tier 1	Detection Rate	FP Rate	Overall accuracy
G2	0	0	-2.78	251/485	1.0000	0.00	1.0000
MPCGEN	0	0	-2.92	287/485	1.0000	0.00	1.0000
NGVCK	0.4	33.5	-3.65	211/625	0.8633	0.01	0.8896
MetaPHOR	0.7	9	-2.81	162/525	0.9791	0.03	0.9771
MWOR	0	0	-2.62	222/525	1.0000	0.00	1.0000

Table 4 lists out the time taken in milliseconds by each approach to classify every malware family. Table 4 shows that the threshold approach is the fastest method followed by the proposed approach. The dueling approach, as expected, is the slowest of the three .The execution time for the dueling approach is two to three times longer than that of the threshold approach on average.

Table 4. Comparison - Performance in terms of time

Timing comparison (ms)			
	Threshold Approach	Dueling Approach	Proposed Approach
G2	556.3584	1125.1585	633.5652
MPCGEN	522.4151	1084.0254	613.4547
NGVCK	561.8427	1185.4857	784.9556
MetaPHOR	658.1485	1425.1258	898.3598
MWOR	4494.1589	11351.7675	9524.4412

Table 5 shows that the threshold method is the best approach in terms of time. The dueling approach is much more accurate than the threshold method; however the time required for execution is long.

The proposed approach on the other hand takes the best of both approaches. The proposed method is able to detect destructive files by 98 percent precision and yet it keeps the false-positive rate at minimum and achieves superior results compared to the dueling HMM strategy.

Table 5. Comparison of detection strategies

	Detection Rate	FP rate	Overall accuracy	Total time (ms)
Threshold Approach	0.9593	0.4755	0.8068	849.2757
Dueling Approach	0.9712	0.0554	0.9664	2021.4453
Proposed Approach	0.9803	0.0058	0.9833	1556.8470

V. Conclusion

This paper proposes the idea of combining different approaches in order to detect malware in a more efficient way in terms of time. As it was stated, some of the metamorphic malware which are able to get around the threshold approach are detected by the dual approach with high precision. However, this approach causes much overload. In addition to taking advantage of the benefits of both approaches, we present a new method by distinguishing important file parts based on their dissimilarity to benign files and by extracting important sequences of opcodes from malware files to overcome the challenge. The results demonstrate that the HMM trained based on the important sequences of opcodes is able to act with higher speed and is, in most cases, more accurate than the dual approach. The improvement ranges from almost 42% to a minimum of 8% percent in terms of time. We hope the proposed method allows us to improve malware detection tools based on HMM.

References

[1] A. Kalbhor, T. H. Austin, E. Filiol and M. Stamp, "Dueling hidden Markov models for virus analysis," Journal in Computer Virology Hack Tech:Springer, 2014.

[2] C. Annachhatre, T. H. Austin and M. Stamp, "Hidden Markov models for malware classification," Journal in Computer Virology Hack Tech:Springer, 2014.

[3] D. Baysa, "Structural Entropy and Metamorphic Malware," M.S. dissertation, Dept. Comp. Sc., Univ. San Jose State, 2013.

[4] J. Aycock, "Computer Viruses and Malware," Advances In Information Security:Springer, 2006.

[5] J. Kuriakose and P. Vinod, "Ranked Linear Discriminant Analysis Features for Metamorphic Malware Detection," IEEE International Advanced Computing Conference, pp. 112-117, 2014.

[6] K. Mathur and S. Hiranwal, "A Survey on Techniques in Detection and Analyzing Malware Executables," International Journal of Advanced Research in Computer Science and Software Engineering, pp. 422-428, 2013.

[7] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, pp. 257-286, 1989.

[8] M. Gales, and S. Young, "The Application of Hidden Markov Models in Speech Recognition," Now Publishers Inc, pp. 195-304, 2008.

[9] M. Stamp, "A revealing introduction to hidden Markov models," Dept. Comp. Sc., Univ. San Jose State, 2012.

[10] S. Alam, I. Sogukpinar, I. Traore and R. Horspool, "Sliding window and control flow weight for metamorphic malware detection," Journal in Computer Virology Hack Tech: Springer, pp. 75-88, 2015.

[11] T. H. Austin, E. Filiol, S. Josse and M. Stamp, "Exploring hidden Markov models for virus analysis: A semantic approach," Hawaii International Conference on System Sciences, pp. 5039-5048, 2013.

[12] W. Wong, "Analysis and Detection of Metamorphic Computer Viruses," M.S. dissertation, Dept. Comp. Sc., Univ. San Jose State, 2006.

[13] NGVCK. VX Heavens, Retrieved from http://vxheaven.org/vx.php?id=tn02

[14] G2. VX Heavens. Retrieved from http://download.adamas.ai/dlbase/Stuff/VX%20Heavens%20Library/static/vdat/creatrs1.htm

[15] MPCGEN . VX Heavens, http://vxheaven.org/vx.php?id=tm02

[16] MetaPHOR. Retrieved from http://spth.virii.lu/29a6/29A-6.602.txt

[17] Sridhara, S. M., & Stamp, M. (2013). Metamorphic worm that carries its own morphing engine. Journal of Computer Virology and Hacking Techniques, 9(2), 49-58.

[18] Clang. Retrieved from http://clang.llvm.org/

[19] Cygwin. Retrieved from http://www.cygwin.com/

[20] GCC. Retrieved from http://gcc.gnu.org/

[21] MinGW. Retrieved from http://www.mingw.org/

[22] TASM. Retrieved from http://trimtab.ca/2010/tech/tasm-5-intel-8086-turbo-assemblerdownload

[23] Turbo C, Retrieved from http://edn.embarcadero.com/article/20841

Authors' Profiles

Mina Gharacheh received her B.Sc. Degree on computer software from Pasargad University, Shiraz, Iran in 2011. She is currently a M.Sc. student continuing computer software at Science and Arts University, Yazd, Iran.

Her research interests are in the fields of machine learning, Improving simulation performance of system on chips Using GPUs as accelerator and security.

Vali Derhami received the B.Sc. Degree on control engineering from Esfahan University of Technology, Iran, in 1996. He received M.S. and Ph.D. degrees on control engineering from Tarbiat Modares University, Iran, in 1998 and 2007, respectively.

From 2002 to 2007, he worked in the Intelligent Control Systems Laboratory on "intelligent agent based controller design for robot navigation problem". Currently, he is Associate professor in computer and electrical engineering department in Yazd University. His research interests are neural fuzzy systems, intelligent control, reinforcement learning, robotics, search engine, and information technology. He has published more than 100 papers in conferences and journals.

Sattar Hashemi received the PhD degree in computer science from Iran University of Science and Technology in conjunction with Monash University, Australia, in 2008. Following academic appointments at Shiraz University, he is currently an associate professor at Electrical and Computer Engineering School, Shiraz University, Shiraz, Iran.

He is recognized for contributions in the fields of machine learning and data mining. He has published many refereed papers and book chapters on data stream classification, game theory, social networks, database intrusion detection, and computer security.

Seyed Mehdi Hazrati Fard received his B.Sc. degree on computer software from Shiraz Azad University, Shiraz, Iran in 2007 and his M.Sc. degree in the field of artificial intelligence (AI) from Shiraz University, Shiraz, Iran in 2012. He is currently a Ph.D. student continuing AI at Shiraz University, Shiraz, Iran.

He was worked more than 2 years in the malware detection lab in APA center of Shiraz University and then was a member of antivirus project of this unity. He is currently a Faculty member in the Department of Computer science and IT of Pishtazan institutes of higher education, Shiraz, Iran. His research interests are in the fields of image processing, machine learning, data mining and security and has several publications in these fields.