***I. J. Computer Network and Information Security*, 2019, 9, 9-17**
Published Online September 2019 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2019.09.02

An Improved Method for Packed Malware Detection using PE Header and Section Table Information

Nahid Maleki
Computer Engineering Faculty, Najafabad Branch, Islamic Azad University, Najafabad, Iran
E-mail: n.maleki2006@gmail.com

Mehdi Bateni
Department of Computer Engineering, Sheikhbahaee University, Isfahan, Iran
E-mail: bateni@shbu.ac.ir

Hamid Rastegari
Computer Engineering Faculty, Najafabad Branch, Islamic Azad University, Najafabad, Iran
E-mail: rastegari@iaun.ac.ir

Received: 16 September 2018; Accepted: 16 October 2018; Published: 08 September 2019

***Abstract*—**Malware poses one of the most serious threats to computer information systems. The current detection technology of malware has several inherent constraints. Because signature-based traditional techniques embedded in commercial antiviruses are not capable of detecting new and obfuscated malware, machine learning algorithms are applied in identifing patterns of malware behavior through features extracted from programs. There, a method is presented for detecting malware based on the features extracted from the PE header and section table PE files. The packed files are detected and then unpacke them. The PE file features are extracted and their static features are selected from PE header and section tables through forward selection method. The files are classified into malware files and clean files throughs different classification methods. The best results are obtained through DT classifier with an accuracy of 98.26%. The results of the experiments consist of 971 executable files containing 761 malware and 210 clean files with an accuracy of 98.26%.

***Index Terms*—**Static Malware Analysis, PE Header, Section Table, Classification, Machine Learning.

I. Introduction

Malware is a malicious computer software designed to perform unauthorized actions [1]. On anual many malicious programs are being developed and expanded, and making the anti-virus software developers to seek manner in protecting their customers by devising more signatures and timely updating of software. Despite the variety of antivirus software, malware detection is a major problem and their attacking manner is on a constant change. The most commonly adopted method for detecting a malware sample in commercial anti-malware systems is the application of malware signatures, while the increased severity of malware makes it difficult to detect them. Anti-malware software applies a self signature database [2] in detecting.

An example of a signature is a sequence of bytes always present inside a malicious executable file and files previously infected by that malware. Determination of a new malicious executable file and providing the right solution to it by professionals is only possible after its identification, which occurs after malware infection is spread. The suspicious files can be analyzed by comparing thaqeir bytes with the signatures list. If a similarity is found between them, the file under test will be identified as a malicious executable file. This approach is effective only for pre-known threats. Signature-based identification methods have become popular according to their performance and stability in commercial systems so far. This approach is not able to detect various types of malware modified by obfuscation methods and fail to analyze (Fig. 1).

In normal mode, if the malware detection software intends to detect different types of malware, it must store all the malicious signatures in its database so that in future analyzes it will be able to identify malware with different signatures. This approach increases the database size of the signatures there of [9,10], which can not easily be identified if a new version malware generated with the possibility of reducing the speed of analysis due to the increased volume of signatures in the malware detection database. If this phenomenon changes the file beyond the change of one byte, then devising a unit signature that is capable of detecting similar malware would face a serious problem. Moreover, even adding and deleting or reordering commands can devise different signatures in presence of signature generation methods, which will increase the complexity of the problem. Signature-based methods are of two basic drawbacks: they can not identify obfusecation codes and can not detect previously unobserved malware [3].

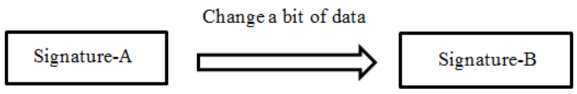

Fig.1. Change the signature of malware by changing one bit

Another method in preventing malware detection is obfusecation, which is on a constant evolution. Obfuscated programs are programs that malware writers try to hide their execution. There exist techniques that obfuscate the code in order to make the analysis and identification of malware difficult. These thechniques are of three groups: polymorphism, metamorphism and packing [11]. In this context, packing is the most common [22]. Polymorphism is an encryption technique that changes the static binary code to prevent detection. Metamorphism is a technique that changes dynamic binary code to prevent detection [21].

Packed programs are a subset of obfuscated programs where a malicious program is packed and can not be analysed [12]. According to [23], over 92% of malwares belong to the packed group. Packing means a packed executable file inside another executable file. In order to detect the malware, this pack must be unpacked. Packing of malwares is the first problem that an analyzer must go through. If unpacking is impossible as well, analysis is impossible because the codes are not understood [24]. Malware writers use obfuscation techniques to hide malware behavior [5-8].

Due to an increase in devising new malware on dayly basis, their recognition based on their signatures is difficult, this fact indicates that this recognition is not. Hence, it is not entirely dependent on antivirus programs to fight malware. An alternative mechanism is needed to identify new and unknown malware. It is assential to develop new techniques by applying malwares analyses. Data mining techniques are more effective, efficient and fast and accelerate discovery and detection of malware. Various features of programs' binary code are extracted and applied to develop descriptive models. Through these features, a malware file can be separated among a clean file.

In this article, the PEiD tool is used to determine if the malware is packed. If the malware is packed,first, it will be unpacked, next, The PE header and section tables information of all malware and clean files are extract and stored in the features database. By applying forward feature selection method 8 features are selected from PE header and section tables, through which the classifier performance is measured. The test files are examined through DT, NN, ID3, NB and SVM classifiers.

This article is arranged as follow: the overview of PE format is introduced to facilitate understanding of other parts in Sec 2. The literature review is summarized in the subject in Sec 3. The proposed malware detection system is described the proposed malware detection and classification system in Sec 4. The evaluation and result is described the classification process with experimental results in Sec 5. The dicussion is followed the discussion and comparison with existing techniques in Sec 6 and the conclusion is concluded the paper in Sec 7.

II. Overview of PE Format

The PE file format is designed by Microsoft Corporation and standardized by Tool Interface Standard Committee in 1993. This format is the file format for the executables, object code and Dynamic Link Library (DLL). A PE file mainly consists of a MS-DOS header, a PE header and a section header/section table, Fig. (2), followed by a brief description.

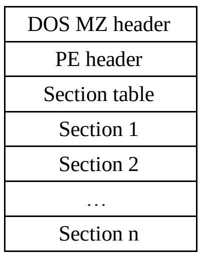

Fig.2. PE file format

- All PE files begin with the DOS header. If the file is executed in the DOS operating system, the DOS header can identify it as a valid executable file. The DOS header actually checks whether a file is a valid executable file.
- PE header is another name for the IMAGE_NT_HEADERS record. This record contains the fields used by the loader of PE file. The PE header has the three fields of: PE-Signature, File Header, and Optional Header.
- Section table is an array of type IMAGE_SECTION_HEADER. These headers are applied in describing each section of the PE file. Each table contains information about a section in PE file, like their attributes and virtual offsets. The compiler in general generates and names the parts of an executable file, where the user has little control over these names. The section count is the second part of file header (6 bytes from the begining of PE header). If the file has 4 sections, this structure will be repeated four times in the table [10]. Each structure is 40 bytes in length.

III. Literature Review

Due to the inability in diagnosing today's commercial anti-viruses, significant concerns are raised in identifying unknown malware. For malware detection, static analyzes are run in a variety of manner, but none assure the complete detection of malware. There exist many studies are different classifications malware detecting regarding [25]. In [10, 32] the techniques applied in identifying malware are classified into two: anomaly-based detection and signature-based detection. The anomaly-based detection method applies recoverable information of clean software, first, to obtain the characteristics of safe behaviors and next, to it identify any significant deviation from this specification as suspicious behavior. A specific type of anomaly-based detection is the specification-based detection. Specification-based methods consist of some specifications or rules, where if the program lacks these specification, it is considered as malicious. A signature-based diagnosis applies features known to be malware to decide whether a program is under malicious software inspection. A signature-based detection applies features known to be malware to decide whether a program of under inspection is malware.

A malware signatures by templates is provided by [33]. Each template has 3 attributes of the instruction, variable and symbolic constraint. The function of this templates is to generalize the signature of a malware sample while maintaining the behavior of malicious codes. Three steps are needed to identify whether PUI is malicious: First, PUI is converted into a platform independent intermediate representation (IR) which is a variant of x86 language, second, a control flow graph is computed for the intermediate representation of PUI, which is compared to that control flow graph of the template and third, comparison is done through the def-use pairs. If for each def-use pair found in the template, there is a corresponding def-use pair in the IR of the PUI, the program is malicious. The results of this study indicate that their template based approach is able to detect malware variants with zero false positives. The binary file entropy graph drawing method is adopted by [25] in order to identify malware. They first converted the binary file into a bitmap file and by drawing the entropy graph followed by comparing their similarities, they identified the similar and polymorphism malware. Because the packed programs and malware may contain the same entropy graph, the detection might be wrong, this, the drawback of this method.

The PAYL, a tool which calculates the expected payload for each service (port) on a system is presented by [34]. A byte frequency distribution is arranged which allows for a centroid model to be developed for each of the host's services. This centroid model is calculated during the learning phase. The detector compares the incoming payloads with this centroid model, measuring the Mahalanobis distance between the two. If the incoming payload is too far from the centroid model (a great Mahalanobis distance value), the payload is considered to be malicious. A combination of learning method using k-Medoids-based clustering technique and naïve bayes classification technique is applied by [35]. Since the k-Medoids clustering technique represents the data distribution actual scenario, the proposed method of entire data classified into corresponding clusters with greater accuracy than that of k-Means with better classification results.

Propose A technique which would dedicate some of the processors' resources to assure only secure instructions are executed is proposed by [36]. In this technique, instruction block signatures are verified at runtime. Instruction block signatures are encrypted by a secret processor key unique to each processor. The signatures for the basic blocks are determined through a function named the Multiple Input Signature Register (MISR). A method for identifying malware based on dynamic behavior is provided by [37], where, all the executable paths of an executable file are run on a virtual environment and its results are stored in databases while one then used to detect new malware.

There, the data mining techniques the features of PE files like PE header features, section tables etc. are examined in detecting malware. In recent years, researchers have focused on data mining techniques for malware detection. Machine learning and data mining techniques are applied to overcome signature-based detection restriction. Data mining techniques outperforms traditional techniques like signature-based detection and anomaly-based detection techniques. Data mining techniques are more effective, efficient and fast. Researchers apply data mining techniques to increase malware detection rate.

A static malware detection system applying data mining techniques is proposed by [38]. This system is based on PE header information, DLL names, API functions call inside the executable file to detect malware and malicious codes. A PE-miner program is developed to parse the PE format of the Windows executable file in the dataset. PE-miner extracts all the mentioned information in the PE file, and stores all the extracted data in the feature database. According to the results of the analysis, finally, 88 PE headers, 130 DLLs and 2453 API functions are selected to train the system with higher efficiency of classifiers and better performance. To reduce the count of features, they activate the PCA function and eventually select 39 PE headers, 75 DLLs, and 307 API functions to evaluate the results. They deduced that the PE header feature, an its own, can effectively be applying in zero-day malware detection. Moreover, classifiers with PE header feature can have the smallest processing overhead. The drawbacks of this research is in ignoring packed malware and applying to much information in detecting, this, an increase in detection time.

A classification framework where both static and dynamic features are applied in detecting malware files among clean files is proposed in [39] applying to much. In the static analysis, they considered the count of suspicious sections and the frequency of functions call. In the dynamic analysis, they applying network and file activities in the run executable file; these features constitute ICMP[1] requests and host IPs count. The results obtained from the experiments indicate that when the static and dynamic features are combined, classification accuracy improves with the respective false negative and false positive values, as compared to when these features

[1] Intenet Control Message Protocol

are considered separately.

A malware detection method based on extracting information from PE files is introduced by [40]. Based on a deep analysis of the statical PE file format information, 17 features are extracted from PE files format information and the feature selection methods are applied to reduce the dimensions of the features and achieve a high acceptable performance. They compared different feature selection methods and deduced that the wrapper method obtained better results in comparison with the filtering method. They run Three experiments to assess the detection scheme performance and detection capabilities of new and unknown malware. Each one of the three experiments consists of three steps: 1) Feature extraction, 2) Feature selection and 3) Classification. This detection method, based on previously observed instances, detects the unknown and new malware while it keeps a low false positive rate.

A simple, fast and scalable method for distinguishing malware files among clean files based on the extracted features from Windows PE executable files is proposed by [10]. The applied static features in this work consist of suspicious sections and functions call frequency. After auto extracting the features of the executable files, machine learning algorithms available in WEKA are applied to classify malware files and clean files. Based on the obtained results, it can be deduced that functions call frequency obtained from the static analysis method contribute in detecting malware files among clean files relative to the count of the suspect sections. When both features are combined, the results of classifying all classifiers will be improved.

IV. The Proposed Malware Detection System

This newly proposed system detects malware through a few data mining techniques. The overall framework of the proposed malware detection system is shown in Fig. (3). The PEiD tool is used to determine if the malware is packed. If the malware is packed, it will be unpack through the Quick Unpack tool. In this system, PE header information and table sections of all executable files are extracted as raw attributes. There, eight features of PE header and section tables are selected through the forward selection method. One of the innovation of this research is the use of section table and PE header features in a hybrid method. The other innovation of this study is the reduction of the number of features that are used in detection process. Learning algorithms are applied to obtain the classification result. The test files are examined by applying DT, NN, ID3, NB and SVM classifiers. Each one of these steps is described as follow in a flowchart:

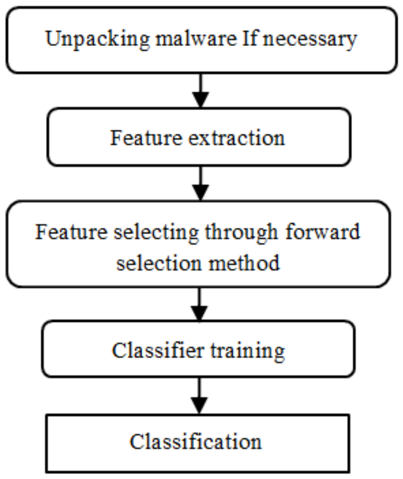
Unpacking malware If necessary
Feature extraction
Feature selecting through forward selection method
Classifier training
Classification

Fig.3. The proposed malware detection system

A. Unpacking

The first step in this process is to examine whether the given file is packed, next, the PEiD tool is used to determine the type of compiler or collected files packer type. The PEiD tool is the best available in determining compiler or packers detections [19] and it is the first tool used in unpacking [20]. The advantages of this consist of: rapidity, free of charge and easy to apply. The disadvantages of this tool sophisticated malware that are usually used by custom packing. After detecting the packed file, it is packed file detection, unpacked through Quick Unpack tool.

B. Feature extraction from PE header and section table files

After the packed files are unpacked, the PE header and section table features are extracted from the files database. These features are extracted from malware files and clean files in a statical manner (i.e. without executing executable file). This system is designed to scan executable files and obtain information about PE header and section table features. A total of 30 features considered as initial set of feature are collected from malware files and clean files data sets. The extracted features from PE header in Table 1 and the extracted features from section tables in Table 2 are tabulated and described (some of these features are named arbitrary). This system puts all the extracted information in database of features.

Table 1. Extracted some features of PE header files

No	Feature	Field	Description
1	NumberOfSections	File Header	Exhibiting the executable sections' count in the file
2	Relocation_stripped_flag	File Header	Removes swap information from file
3	Excutable_Image_flag	File Header	Marker of credit and applicable executable file
4	Line_numbers_flag	File Header	Deleting number of executable lines from file
5	Local_Symbols_flag	File Header	Having a symbol table
...	...	...	...

C. Feature Selection

Choosing a feature is of three objectives: 1) improving classifier prediction performance, 2) presentation of faster and more cost-effective classifier and 3) gaining a deeper insight into basic processes that generate datas. The data must be pre-processed to select a subset of the optimal features at learning, because in many situations, many features use in learning programs that some of them - Perhaps the overwhelming majority - are clearly insignificant or redundant.

Table 2. Extracted some features from section table files

No	Feature	field	Description
1	NumAPI_Kernel32	Export Table	Counts the number of Kernel32 library functions
2	NumAPI_User32	Import Table	Counts the number of User32 library functions
3	Num_Import_DLL	Import Table	Number of libraries linked to executable file
4	SizeOfResourceTable	Resource Table	resource table size
5	Debug_Directory_Size	Section Table	Displaying the debug directory size
...	...	...	...

In this system, feature selection is run through the forward selection method, where most relevant features are selected through very efficient execution. The forward selection method begins by empty selection from features and it adds each unused features in each round. For each added feature, function is estimated through internal operators, for example, cross-validation is the only feature that adds the highest performance increment to the selection. Then a new round begins with a modified selection. The features selected through this method are tabulated in Table 3. Some of these features are named arbitrarily.

Table 3. Selected features by forward selection method

No	Feature	Header	Explanation
1	Byte reversed low flag	PE Header	LSB precedes MSB in memory
2	Debug_info_flag	PE Header	Remove debug info. From the file
3	Num_Executable_Section	PE Header	Count the number of executable sections in the file
4	NumberOfSection	PE Header	Count the number of sections in the file
5	No multiprocessor system flag	PE Header	The file design for multi-core processors
6	DllCharacteristics	PE Header	Displaying the executable file features
7	Debug_Directory_Size	Section Table	Displaying the debug directory size

D. The Classification Process

In order to verify the malware detection system, DT, NN, ID3, NB and SVM classifiers are applied. Data mining algorithms consist of training and test sets. The test set is applying in checking the accuracy of classification on unseen instances. In detecting malware, it is important to maintain low time in decision making for diagnosis. Decision tree algorithm has the least decision time, In Fig. (4).

The newly proposed DllCharacteristics feature, constribute the root of this DT. DllCharacteristics, first, displays the features of executable file like: executable and Applicability implementation file, executable System file etc., next it is split into two subfolders, if DllCharacteristics feature is greater than 17056, then it counts the executable sections. If the count of executable sections is greater than 1500, it checks the file sections count. The leaf nodes exhibit the debugging information removed from file and the executable sections' count. If DllCharacteristics feature is less than or equal to 17056, it computes Debug_Directory_Size feature that shows debug directory size. If this feature is less than or equal to 14,500, it identifies the file as malware; otherwise, it will check the Debug_info_flag feature. If debugging information is removed, it considers the file as malware; otherwise, it counts the file sections.

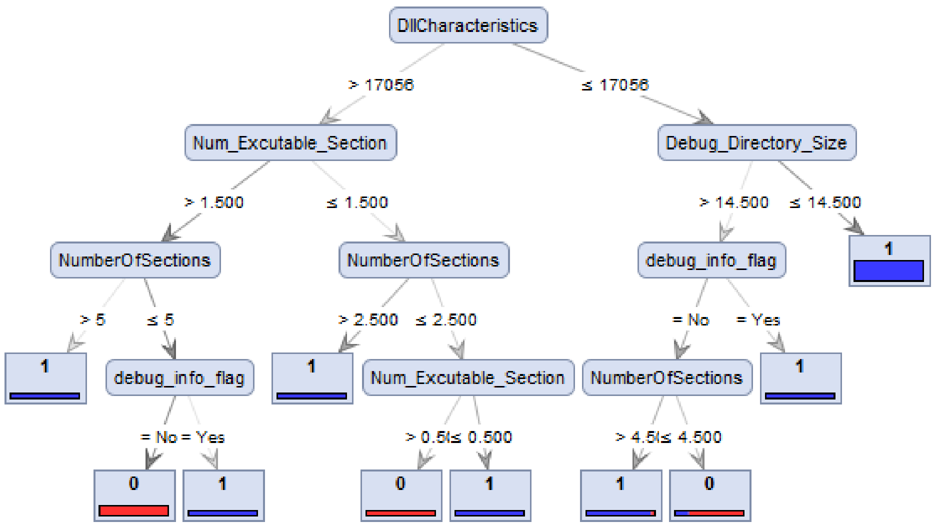

Fig.4. Decision Tree for malware classification

V. Evaluation and Results

A. Terms of testing

These experiments are run to evaluate the proposed method on a system with dual-core processor and 4 GB memory. There, the PEiD v0.95 version is applied in order to identify compiler or file packer and Quick Unpack v2.2 version is applied in order to unpack the packed files. The 7.4000 Rapidminer software is applied in order implement and obtain the test results.

B. Data set

A total of 971 PE files are applied in testing this system, which contains 761 malware files and 210 clean files. The samples of malware are downloaded from VxHeavens website [41]. These data are used in many researches [18,38,40] in this field. Various types of malware like: Backdoor, Hacktool, Trojan, Email-Worm, Virus, Worm, etc. are tabulated Table 4. The clean files are collected from Windows XP installation location. After malware is unpacked to assure their proper selection, the entire dataset is scanned through ESET Antivirus. Dataset is proportionated by 80% of training set and 20% of test set.

C. Experiment results

In this experiment, the accuracy of classification is tested through: DT, NN, ID3, NB and SVM classifiers. The values of experimental results for the existing classifications are tabulated in Table 3. Experiment the following criterion are of concern: false positive (FP), false negative (FN), detection rate (DR), recall, Precision and accuracy (Acc), Table 5.

Table 4. The type and amount of malware used in the tests

No	Malware Type	Normal	Packed Malware	Count
1	Backdoor	26	22	48
2	Email-Worm	25	21	46
3	Exploit	36	23	59
4	Hacktool	22	20	42
5	Net-Worm	28	19	47
6	P2P-Worm	27	21	48
7	Trojan	50	37	87
8	Trojan-Downloader	36	23	59
9	Trojan-Dropper	33	31	64
10	Trojan-Spy	35	28	63
11	Virus	55	41	96
12	Worm	53	49	102
Total = 761				

Table 5. Experimental results

Classifier	Classification Results					
	FP (%)	FN (%)	DR (%)	Re (%)	Pr (%)	Acc (%)
DT	0.058	0.004	99.54	98.46	94.12	98.26
NN	0.072	0.004	99.54	98.46	92.75	97.92
ID3	0.115	0.018	98.17	93.85	88.41	95.83
NB	0.140	0.018	98.15	93.85	85.92	95.14
SVM	0.140	0.018	98.15	93.85	85.92	95.14

The performance of experiments reveal that NB and SVM classifiers are of the worst accuracy at 95.14%. The DT classifier is of the highest accuracy at 98.26%. The NB and SVM classifiers reveal the precision at 85.92% and the DT classifier is of the highest precision at 94.12%. The detection rate of NB and SVM classifiers are similar at 98.15% representing the lowest value, and the DT and NN classifiers representing the highest value at 99.54%. The comparison of classifiers according to desired features in these experiments are barcharted in Fig. (5).

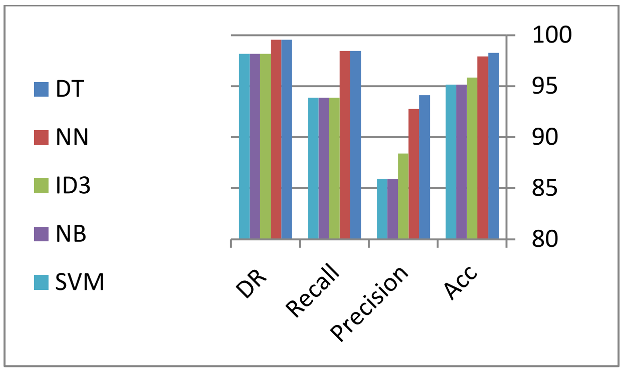

Fig.5. Comparison of different classifiers

VI. Discussion

The results obtained from available experiments in this context are which compared with the results of this proposed method, Table 6, when the accuracy varies within 1.01% to 30.76%. Fewer and more effective features are selected in this proposed method which has gained more accuracy.

The authors in [13] were the first to introduce a malware detection system based on machine learning. Their proposed system is based on analysis of various information contained in strings and system calls in the executable files. Their objective is to discover the count of standard data mining techniques for calculating precise detectors for new binary files (unseen). They extracted a binary profile from each sample in a dataset in an automatical manner, they extracted the binary profiles in a statical sense to be applied in classifiers and to generate diagnostic models. They applied NB classifier and obtained 97.11% accuracy.

The n-gram byte sequence is applied in [14] in order to deal with problem of malware classification. They classified malware based on text classification concept and obtained 95% accuracy. Their method can reveal accurate unknown malicious code detection based on previously observed instances, while it keeps low levels of false alarms.

The fact that usually a obfuscated file does not contain any strings containing composed words or sentences is observed by [15], where, available printable string information in the executable file for classification of malware consist of 1367 datasets. They selected five algorithms for classification and each was applied separately. In each conjunction, they applied the AdaBoost increase technique. They developed WEKA interface, Moreover they developed a five-part validation test based on training and testing sets as well. Their experiments reveal that IB1 and Random Forest classification methods are most effective in this domain. Their obtained accuracy is 97%.

A system calls is assessed by [16] where the NB classifier is applied. The extracted system calls is applied in building the model based on suspicious behaviors by grouping some system calls, because malware can run scenarios like obtaining system directory, writing malicious data for files and updating registry, with a 93.7% accuracy.

A malware detection system based on analysis of a set of system named called PE programs is introduced by [17], where the feature selection method is based on KHI test. They applied object oriented mining society classification method, with a 67.5% accuracy.

A PE malware detection system based on saved information analysis in PE-Optional Header Fields (PEF) proposed by [18], which consists of three parts: PE-parser, feature selection module and decision module. A PE-parser is developed in a statical sense by applying Python language which extracts available information in the optional header fields. Their system applied the Chi-square score (KHI) and the Phi (φ) coefficient as a feature selection method. They evaluated their system through Rotation Forest classifier implemented in WEKA and obtained and accuracy more than 97%.

Table 6. Comparison of our method with existing work

Method	Data set	Feature Type	Acc (%)
Schultz et al. [13]	4,266	Strings	97.11
Moskovitch et al. [14]	30,601	byte n-gram	95
Tian et al. [15]	1,367	Printable string information	97.5
Wang et al. [16]	714	API call sequence	93.71
Ye et.al [17]	50,000	API call sequence	67.50
Belaoued and Mazouzi [18]	552	Optional header	97.25
Our method	**971**	**PE header And Section header**	**98.26**

VII. Conclusion and Future Work

A method based on data mining techniques is proposed here, where, features of the PE header and the PE file sections table are applied to improve the accuracy of malware detection and reduce the detection error rate. Detection of packed malware by its signature is very difficult. In order to packed malware detection, its must be unpacked. In this study 8 features are applied as static features in detecting malware files among clean files. To identify these features the a forward selection method is adopted. These features can be applied as inputs for machine learning algorithms for malware classification. The experiments are run on 971 executable files revealed 98.26% accuracy in the DT classifier. The results of this classification can be applied in anti-virus programs to improve of malware detection rate. As a future work, many dynamic features in the context can be combined in order to increase the accuracy in malware detection system.

REFRENCES

[1] J. Raphel and P. Vinod, "Information theoretic method for classification of packed and encoded files", Procedings of the 8th International Conference on Security of Information and Networks - SIN '15, 2015.

[2] P. Morley, Processing virus collections, in: Proceedings of the 2001 Virus Bulletin Conference (VB2001), Virus Bulletin, 2001, pp. 129–134.

[3] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," Information Sciences, vol. 231, pp. 64–82, May 2013.

[4] N. Kuzurin, A. Shokurov, N. Varnovsky, V. Zakharov, On the concept of software obfuscation in computer security, Lecture Notes in Computer Science 4779 (2007) 281.

[5] D. Bruschi, L. Martignoni, M. Monga, Detecting self-mutating malware using control-flow graph matching, Lecture Notes in Computer Science 4064 (2006) 129.

[6] Q. Zhang, D. Reeves, Metaaware: identifying metamorphic malware, in: Proceedings of the 2007 Annual Computer Security Applications Conference (ACSAC), 2007, pp. 411–420.

[7] M. Chouchane, A. Lakhotia, Using engine signature to detect metamorphic malware, in: Proceedings of the 2006 ACM workshop on Recurring Malcode, ACM, New York, NY, USA, 2006, pp. 73–78.

[8] M. Karim, A. Walenstein, A. Lakhotia, L. Parida, Malware phylogeny generation using permutations of code, Journal in Computer Virology 1 (2005) 13–23.

[9] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime," Recent Advances in Intrusion Detection, pp. 121–141, 2009.

[10] Nwokedi Idike and Aditya P. Mathur, "A Survey of Malware Detection Techniques", Technical Report, Purdue University, 2007.

[11] P. OKane, S. Sezer, and K. McLaughlin, "Obfuscation: The Hidden Malware", IEEE Security & Privacy Magazine, vol. 9, no. 5, pp. 41–47, Sep. 2011.

[12] Sikorski M. and Honig A., Practical Malware Analysis: the Hand-On Guide to Dissecting Malicious Software, no starch press, 2012.

[13] Schultz, M.G., Eskin, E., Zadok, E., Stolfo, S.J., "Data mining methods for detection of new malicious executables," in Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001.

[14] Moskovitch, R., Stopel, D., Feher, C., Nissim, N., and Elovici, Y. 2008. Unknown Malcode Detection via Text Categorization and the Imbalance Problem. In *Proceedings of 6th IEEE International Conference on Intelligence and Security Informatics (ISI)*, Taiwan, 2008, 156–161.

[15] Tian, R., Batten, L., Islam, R., and Versteeg, S. 2009. An automated classification system based on the strings of Trojan and virus families. In *Proceedings of 4th International Conference on Malicious and Unwanted Software*, Montréal, Quebec, Canada, 2009, 23-30.

[16] Wang, C., Pang, J., Zhao, R., and Liu, X. 2009. Using API Sequence and Bayes Algorithm to Detect Suspicious Behavior. In *Proceedings of International Conference on Communication Software and Networks*, IEEE Computer Society Washington, DC, USA, 2009, 544–548.

[17] Ye, Y., Li, T., Jiang, Q., and Wang, Y. 2010. 'CIMDS: Adapting Post processing Techniques of Associative Classification for Malware Detection', IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2010, 40, 3 (2010), 298-307.

[18] Belaoued, M., Mazouzi, S., "A real-time pe-malware detection system based on chi-square test and pe-file features", In: IFIP International Conference on Computer Science and its Applications. Springer, pp. 416–425, 2015.

[19] Sikorski M. and Honig A., Practical Malware Analysis: the Hand-On Guide to Dissecting Malicious Software, no starch press, 2012.

[20] Goppit, Portable Executable File Format - A Reverse Engineer View, Code Breakers Journal - Aug 15, 2005.

[21] S. Alam, R. N. Horspool, and I. Traore, "MARD: A framework for metamorphic malware analysis and real-time detection", Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA, vol. 48, pp. 480–489, 2014.

[22] Goppit, Portable Executable File Format - A Reverse Engineer View, Code Breakers Journal - Aug 15, 2005.

[23] T. Brosch and M. Morgenstern, "Runtime Packers: The Hidden Problem," Proc. Black Hat USA, Black Hat, 2006; www.blackhat.com/presentations/bh-usa-06 BH-US-06-Morgenstern.pdf.

[24] Y. Choi, I. Kim, J. Oh, and J. Ryou, "PE File Header Analysis-Based Packed PE File Detection Technique (PHAD)," International Symposium on Computer Science and its Applications, Oct. 2008.

[25] A. Elhadi, M. A. Maarof and A. H. Osman, "Malware Detection Based on Hybrid Signature Behaviour Application Programming Interface Call Graph", American Journal of Applied Sciences, vol. 9, no. 3, pp. 283–288, Mar. 2012.

[26] K. Mathur and S. Hiranwal, "A survey on techniques in detection and analyzing malware executables", Int. J. Adv. Res. Comp. Sci. and Soft. Eng., vol. 3, no. 4, pp. 422-428, 2013.

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

[28] I. Basu, N. Sinha, D. Bhagat, and S. Goswami, "Malware Detection Based on Source Data using Data Mining: A Survey", American Journal of Advanced Computing, Vol. 03, no. 01, pp. 18-37, 2016.

[29] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques", in Proc. 5th Conf. Inf. Knowl. Technol. (IKT), 2013, pp. 113–120.

[30] S. Alqurashi and O. Batarfi, "A Comparison of Malware Detection Techniques Based on Hidden Markov Model", Journal of Information Security, vol. 07, no. 03, pp. 215–223, 2016.

[31] Ammar Ahmed E. Elhadi, Mohd Aizaini Maarof and Bazara I. A. Barry, "Improving the Detection of Malware Behaviour Using Simplified Data Dependent API Call Graph", International Journal of Security and Its Applications Vol.7, No.5, pp.29-42, 2013.

[32] Ms. Shital Balkrishna Kuber. "A Survey on Data Mining Methods for Malware Detection" International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014.

[33] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, "Semantics-aware Malware Detection", In IEEE Symposium on Security and Privacy, 2005.

[34] K. Wang and S.J. Stolfo, "Anomalous Payload-Based Network Intrusion Detection", Proc. Seventh Int'l Symp Recent Advances in Intrusion Detection, pp. 203-222, 2004.

[35] Chitrakar R., Chuanhe H., "Anomaly based Intrusion

Detection using Hybrid Learning Approach of combining k-Medoids Clustering and Na ïve Bayes Classification", In Proceedings of 8th IEEE International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), pp. 1-5, 2012.
[36] M. Milenković, A. Milenković, and E. Jovanov, "Using instruction block signatures to counter code injection attacks," ACM SIGARCH Computer Architecture News, vol. 33, no. 1, p. 108, Mar. 2005.
[37] C. Hu, X. Wang, N. Li, H. Bai, and X. Jing, "Approach for malware identification using dynamic behaviour and outcome triggering", IET Information Security, vol. 8, no. 2, pp. 140–151, Mar. 2014.
[38] U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A Static Malware Detection System Using Data Mining Methods," International Journal of Artificial Intelligence & Applications, vol. 4, no. 4, pp. 113–126, Jul. 2013.
[39] E. Gandotra, D. Bansal, S. Sofat, "Integrated Framework for Classification of Malware", Proc. of 7th International Conference on Security of Information and Networks, ACM, University of Glasgow, UK. September, 2014.
[40] J. Bai, J. Wang, and G. Zou, "A Malware Detection Scheme Based on Mining Format Information", The Scientific World Journal, vol. 2014, pp. 1–11, 2014.
[41] "VXHeavens," http://vxheaven.org/.

Authors' Profiles

Nahid Maleki, received the B.Sc. degree in computer engineering in 2013 from Islamic Azad University of Najafabad, Isfahan, Iran, and the M.Sc. degree in computer engineering from Islamic Azad University of Najafabad, Isfahan, Iran, in 2017.

Mehdi Bateni, is an assistant professor of computer engineering at the Faculty of Engineering of the Sheikhbahaee University (SHBU). He received his B.Sc. in Computer Engineering in 1997 from University of Isfahan, Isfahan, Iran and his M.Sc. in Computer Engineering from Ferdowsi University of Mashhad, Mashhad, Iran in 2000. He received his Ph.D. in Computer Engineering in 2012 from University of Isfahan, Isfahan, Iran.

Hamid Rastegari, is an assistant professor of computer engineering at Islamic Azad University of Najafabad. He received his Ph.D. in Computer Engineering in 2011 from University of UTM, Malaysia.

How to cite this paper: Nahid Maleki, Mehdi Bateni, Hamid Rastegari, "An Improved Method for Packed Malware Detection using PE Header and Section Table Information", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.9, pp.9-17, 2019.DOI: 10.5815/ijcnis.2019.09.02