Journal of Theoretical and Applied Information Technology
10th December 2014. Vol.70 No.1
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A NOVEL DATA MINING METHOD FOR MALWARE DETECTION

[1]* HAMID REZA RANJBAR, [2]MEHDI SADEGHZADEH, [3]AHMAD KESHAVARZ

[1]Department of Computer Engineering, Science and Research branch, Islamic Azad University,

Bushehr, Iran

E-mail: [1] h.r_ranjbar@yahoo.com , [2] sadegh_1999@yahoo.com, [3] ahmad_modarres81@yahoo.com

* Corresponding Author: HAMID REZA RANJBAR, h.r_ranjbar@yahoo.com

ABSTRACT

Losses caused by malware are irrecoverable. Detection of malicious activity is the most challenge in the security of computing systems because current virulent executable are using sophisticated polymorphism and metamorphism techniques. It make difficult for analyzers to investigate their code statically. In this paper, we present a data mining approach to predict executable behavior. We provide an Application Programming Interface (API) which provides sequences captured of a running process with the aim of its predicting intention. Although API calls are commonly analyzed by existing anti-viruses and sandboxes, our work presents for the first time that using an API and the number of iteration as a countermeasure for malware detection in the API. The experiments have shown the effectiveness of our method on polymorphic and metamorphic malware by achieving an accuracy of 93.5% while keeping detection rate as high as 95%.

Keywords: *Malware, Polymorphic, Metamorphic malware, Data Mining, API calls.*

1. INTRODUCTION

Methods for detection of computer viruses which are categorized as malware have been developing since 1983[1] when Cohen introduced computer virus term. Malwares are divided into several groups including viruses, worms, Trojans, spyware and a combination of these groups. It is impossible to exactly distinguish between these groups [2]. According to reports [3], damages resulted from viruses in more than 13 billion dollars annually. Methods based on signature recognition and extended approaches both might be utilized to detect such destructive programs. Despite high speed of signature based methods, malware programmers exploit steganography techniques to avoid detection by conventional algorithms. The failure of signature based methods for detecting this new generation of malware has caused researchers to focus on finding more scalable methods. These methods are supposed to be substituted for signature recognition methods. The method discussed in this paper is based on data mining approach. Recently data mining has been noticed for malware detection; in the next section the previous methods and research works will be presented.

2. MATERIAL AND METHOD

Malware:
Each program which is designed to damage computer system operations is called destructive software or malware. These programs include viruses, worm, Trojan, backdoor, spyware, advertisements and robot. All malwares are sometimes referred to as virus and enterprise anti malware software is still called anti-virus.

3. DETECTION METHOD

Data mining consists of using a complete statistical set and machine learning algorithms in a set of characteristics derived from malware and clean programs [8-11, 20, 32, 36].
Characteristic type sends input data to malware detection system. The source of this data is utilized as a criterion for malware and intrusion detection

systems [5, 21, 26, 37, 38]. If the source is a network, the detection is called network based detection. Characteristics extracted from programs are exploited in host based detection systems. There are diverse reverse engineering methods for extraction of byte sequences, Assembly instructions sequences and API sequences. Network based detection systems use network connection history as data resource.

API is a source code interface which is provided for computer programs by operating system or library so that programs can announce their requests for access to a specific service through APIs. In operating system API means the same as system functions. API call sequence demonstrates program activity. In this study this sequence is utilized to investigate destructive behavior.

The analysis might be performed either in source code or binary level. As mentioned in [4, 12, 19, 20] accessing to the source code for each program is not realistic. The information about execution files can be obtained via their execution or reverse engineering operation or both. Reverse engineering is called static analysis while the actual execution of the program is called dynamic analysis.

Static analysis provides information about data stream and programs control stream and other statistical characteristics without actual execution of the program. Several reverse engineering methods such as decomposition and image compile are used for illustrating executive code. If data format is readable for human, other methods might be utilized as well. The major advantage of static analysis over its dynamic counterpart is independency of runtime overload. The major problem of static analysis is that there is only an approximation of real program. In each decision point the branch must be guessed in the run time.

Dynamic analysis requires the program to be executed and monitored in a real environment or a virtual machine. Although in this method actual information is achieved from data and control stream, it suffers from execution header. When a part of code fails in static analysis, hybrid analysis[18] might be utilized.

Data mining have attracted great attention in recent years for detecting unknown viruses [1, 2, 3]. A few classifiers are designed and high precision is achieved. The most common method where data mining is used for virus detection is generating a set of characteristics [15, 18]. These characteristics might be sequence of hexadecimal bytes (N-Gram), instruction sequence, system function sequences and so on. Usually numerous characteristics are extracted from files. There are methods[39] for selecting the best characteristic. Some other characteristics might be dynamic link libraries information used by programs. In this section we want to investigate research works conducted on virus detection by data mining.

4. RELATED WORK

The first prominent work done using data mining methods for malware detection [22-31] was signature extraction automation for viruses. Viruses were executed in a safe environment to infect trap programs. Candidate signatures with variable lengths were extracted using analysis of infected regions in the programs. Signatures with lower fake positive probability are chosen as the best signature. To deal with longer signatures 3-Gram approach used to be utilized where a sequence was divided into three sections. Then using an approximation formula, long sequence probability is estimated by combining estimated iteration of smaller sequences.

5. PROPOSED METHOD

In this section the method by which malwares are detected in this research is presented and algorithms of each part are introduced [6, 7, 13, 19, 33, 40].

The data mining process is explained in five stages.

- Problem definition
- Data collection
- Data preprocess
- Model estimation
- Model explanation

As it is evident, the problem is detecting malwares using software classification. As it was mentioned in related work section the essential part of malware detection is data collection stage. The major contribution of this study is extraction of system functions from execution files during runtime which has not been performed in previous research works. Furthermore, in preprocess stage a set of these functions is selected as characteristics. The benign and destructive files are introduced as more distinct characteristics.

The problem of detecting a program as destructive or benign is interpreted as a classification problem. Learning theory explicates such problems where the procedure of learning from data is divided into two steps:

- generating the model based on input sample data
- predicting new data based on generated

model

The proposed method is data mining based approach. For the first time, our suggested method utilizes APIs during runtime and considers their number of iterations as a parameter effective on malware detection.
Our proposed system is divided into four steps:

- Collecting executable files and executing them
- Registering interactions between execution files and the system
- Extraction of API functions, selecting them as characteristics and choosing the number of iterations as the value of characteristics
- Using different classifications for detection

The two first stages are data collection stage. Besides, two other stages are data preprocess and system learning respectively.

6. DATA COLLECTION

When the problem is determined the corresponding data must be collected to perform other stages.
Executable files
The main analysis units are computer files. The files are divided into two groups based on content. Data files are those which store the information and executable files which are a set of instructions leading to a particular task. A computer program includes at least one executable file and a set of data files as complementary. Since the malwares are programs they include executable files as well. Worms, Trojans and spywares all are programs executed independently; whereas, viruses inject themselves to other programs and infect them.
In this study the malwares of Windows operating system are merely considered. The executable files of windows are called Portable execution file (PE) whose format is presented in figure2.
The PE file is simply divided into two parts; header and body. The header consists of information about the file such as number of sections and beginning address while the body includes main content of the file.
1197 PE files were collected among which 806 files were malware and 391 files were benign programs.
Benign programs are mostly Windows xp programs and collected viruses are mostly transformation and polymorph viruses.

7. VIRTUALIZATION

Our approach is based on dynamic analysis. As mentioned before, dynamic analysis means real execution of executable files. As a consequence of destructive property of the most of the malwares they cannot be executed in a normal and unprotected environment as they may cause data theft and delete data files. Therefore, virtualization is employed to provide a controlled environment. Virtualization is a process which separates software and hardware components [14, 16, 35] on which the software is executed. One of the most popular types of virtualization is operating system virtualization [17, 27, 34, 39]. In this method various operating systems can be simulated using a super layer. This super layer is a virtual platform which is responsible for managing access to physical resources for virtual machines. Virtual machine is a set of virtual hardware which have standards of x86 and x64 computers. Super layers are divided into two groups. The first one is simple metal super layer which is directly installed on the hardware and host super layer which is installed on conventional operating systems. In our study host virtual operating systems such as VMware Workstation are utilized which are a controlled environment. Executing each file may influence on the results of other files execution. Thus, after each execution, system is refreshed using instant image capability. It will protect virtual system from damages as well as isolating the results of one file execution from other executed files. Software data contact with three elements: humans, processes and products. Humans include programmers, testers, project managers and users. Processes consist of different stages of development and activities such as requirements, design, implementation, test, debugging, malware detection, maintenance and installation. Products may either have structure such as source code or may be without structure such as documentation and error report.
Software data are divided into three groups:
A. sequence: including paths collected during runtime, static sequences collected from source code or binary file.
B. graph: including graphs collected during runtime, static graphs collected from source code or binary file.
C. text: including error report, e-mails, code comments and dominations.

Data mining is utilized by software engineers for modification of software quality and by executable file analyzers for detecting abnormalities in

algorithms. For instance, some algorithms might be employed to detect APIs called in a program while they even do not have documentation or source code. For maintenance expression, one may say that some algorithms are able to determine whether a part of code should affect the other part in case any changes or not. Furthermore, software engineers might exploit data mining algorithms for error detection.
According to introduced types of data in software engineering, this study aims to extract API functions sequence from executable files. Finally, the goal of our study is to prove our claim regarding richness of this type of data and using them for learning classification so that malwares could be detected.

8. REGISTERING SYSTEM FUNCTIONS

Programing interface (API) is a source code interface which is provided by operating system or libraries for computer programs so that they could request special services from operating system or library. Each program requires API functions of operating system to perform its operations such as changing files, registry, network and so on. Hence, when a program runs it calls a sequence of system functions which is known as software behavior. In order to register system functions trapping operation or debugging might be used. There are numerous methods [59] by which malwares could detect debuggers and deny performing debugging on themselves. As a result, in this study the second method is utilized for monitoring executable files.
Hook is a region in messaging mechanism of a system where a program can install a function and monitor all messages of a process. To monitor an executable file, one should access its address space. In Windows operating system each process has its own address space. When pointers are exploited for access to memory, the value of pointer is transferred to the memory address in the range of address space associated with the same process. A process cannot assign a pointer to an address which belongs to another process. Hence, if the program has errors and it wants to randomly rewrite a part of memory which is used by another process, it would not be able. Separate address space is considered an advantage for both programmers and users. For programmers it is advantageous as the system may want to read irrelevant memories. Moreover, it is beneficial for users as the operating system would be more robust; because a program cannot interfere in operating system or another program. Nevertheless, the cost of this robustness is paid by providing a program which is capable of contacting other processes.
The situations where it should be possible to change permissible range of a process to access another process include the following.
- When it is desired to perform a window access operation by another process- e.g. one may want to read password field of another process which can be done by that processor.
- When debugging is needed- for example, when one needs to know which DLLs are utilized by other process.
- when trapping must be done in other processes- for instance when one wants to know which APIs are called by one process.
For this purpose DLL injection operation might be used [60]. DLL injection means that one may put desired code in a DLL and inject it to the address space of target process. When our DLL codes are inside address space of another process, we could completely access that process. In this circumstance we may either monitor or even destroy that process. There are several methods for injecting DLL to address space of another process.

9. DATA PREPROCESS

After system functions sequence is extracted from executable files and before applying classification algorithms to them, they should be formed in an appropriate manner and fed to classification algorithms. With each API and its number of iterations a primary database is formed. For each sequence a tag is considered which demonstrates whether it is benign or destructive. Despite relational data, behavior of software which is a sequence of APIs does not have any predefined characteristics. Thus, in sequential database, extracting characteristics is a prominent part of data mining process. Here, APIs and their iterations are considered as characteristics. It will be seen in experiments and studied cases section that using these characteristics as distinct ones will provide acceptable results. Besides, proper precision for classification might be achieved.
Using LIBSVM and random forest models diverse models are proposed used for prediction.
The Algorithm 1 illustrates classification framework of destructive and clean files.

10. RESULTS AND DISCUSSION

In this section the experiments conducted on database are presented and obtained results are discussed.

The models are tested using test data. Utilizing achieved response, complexity matrixes are derived for each classifier. The undergoing measures constitute the complexity matrix.

- True positive: number of destructive programs correctly detected.
- False positive: number of destructive programs detected incorrectly.
- True negative: number of benign programs detected correctly.
- False negative: number of benign programs detected incorrectly.

The performance of each classifier is evaluated using detection rate, false caution rate and total precision, which are defined as follows:

- Detection rate: percentage of correct detection of destructive programs

$$Detection\ Rate = \frac{TP}{TP + FP}$$

- False caution rate: percentage of incorrect detection of benign programs

$$FalseAlarm\ Rate = \frac{FN}{TN + FN}$$

- Total precision: percentage of correct detection of whole programs

$$OverallAccuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The proposed method is completely implemented as the following figure. Marvelous results were achieved.

In the system function registration step, three crucial DLLs of the operating system were monitored. These include advapi32.DLL, comct32.DLL and ws2_32.DLL which are usually used for sockets, Graphical User Interface components and registry. Additionally, several classifiers are used for building the model and the best results belong to random forest. To evaluate our proposed method, tenfold cross validation method was utilized. Our database includes 806 malware (which are mostly polymorph) and 391 benign programs. Database is divided into ten equal parts out of which 9 parts are used for experiments and one for test each time.

Obtained results revealed efficiency of our method for detection of polymorph and transformation malwares. The best precision is 93.5% and the best detection rate is 95%.

Considering increasing growth of malwares and their damages, the request for detection of malwares has also increased. Although enterprise antiviruses can use signature of malwares to detect them, they are inefficient for new viruses with polymorph and transformation capabilities.

In this thesis a novel malware detection system based on dynamic APIs was proposed. The main point regarding our proposed method was extraction of derived APIs and choosing them as characteristics which is utilized for the first time. Furthermore, the trapping capability is exploited which overcame debugging deficiencies.

As future work more complicated behavioral patterns of malwares might be extracted in both dynamic and static conditions. These extracted patterns can also be utilized as characteristics for classification learning.

REFRENCES:

[1] F. Cohen, "Computer Viruses," PhD thesis, University of Southern California, 1985.

[2] P. Szor, the Art of Computer Virus Research and Defense. New Jersey: Addison Wesley for Symantec Press, 2005.

[3] 2007"Malware Report: The Economic Impact of Viruses, Spyware, Adware, Botnets, and Other Malicious Code," computer economics, 2007.

[4] G. P.-S. W. Frawley, and C. Matheus. (1992) Knowledge Discovery in Databases: An Overview. AI Magazine. 213–228 .

[5] Mehdi Bahrami, Mohammad Bahrami ,"An overview to Software Architecture in Intrusion Detection System", International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 1, No. 1, pp. 1-8, 2011, Doi: 10.7321/jscse.v1.n1.1

[6] R. A. a. R. Srikant, "Mining sequential patterns," presented at the ICDE, 1995.

[7] R. A. a. R. Srikant., "Fast algorithms for mining association rules," presented at the VLDB, 1994.

[8] S.-C. K. D. Lo, and C. Liu, "Efficient mining of iterative patterns for software specification discovery," presented at the KDD, 2007.

[9] J. a. K. Han, Micheline, Data Mining:Concepts and Techniques, 2 ed.: Morgan Kaufmann, 2006.

[10] I. H. a. F. Witten, Eibe, Data Mining : Practical Machine Learning Tools and Techniques: Morgan Kaufmann, 2005.

[11] Shah, Satish K., Pooja S. Suratia, and Nirmalkumar S. Reshamwala. "Comparative Performance Analysis of ANN Based MIMO Channel Estimation for downlink LTE-Advanced System

employing Genetic Algorithm." proceeding of International Conference On Soft Computing and Software Engineering [SCSE'13]..

[12] L. Breiman, "Possible Virus Attacks Against Integrity Programs and How to Prevent Them," presented at the 6th Internation Virus Bulletin Conference, 1996.

[13] Mansor, Muhammad Naufal, et al. "PCA-Based Feature Extraction and LDA algorithm for Preterm Birth Monitoring." Journal of Soft Computing and Software Engineering (2012).

[14] Choudhari, Jitender, and Ugrasen Suman. "Code Change Approach for Maintenance using XP practices." The International Journal of Soft Computing and Software Engineering [JSCSE] 3.3 (2013): 131-136.

[15] M. D. J. Bergeron, J. Desharnais, B. Ktari, M. Salois, and N. Tawbi, "Detection of Malicious Code in COTS Software: A Short Survey," presented at the 1st International Software Assurance Certification Conference, 1999.

[16] Ranjbar, Hamid Reza. "A Review on Software Engineering Methods for Distributed Systems."

[17] Bahrami, Mehdi. "Cloud Template, a Big Data Solution." arXiv preprint arXiv:1307.4716 (2013).

[18] N. I. a. A. P. Mathur, "A Survey of Malware Detection Techniques," 2007.

[19] V. Bontchev, "Possible Virus Attacks Against Integrity Programs and How to Prevent Them," presented at the 6th Internation Virus Bulletin Conference, 1996.

[20] Lusa, Sofian, and Dana Indra Sensuse. "Study of Socio-Technical For Implemetation of Knowledge Management System." International Journal of Soft Computing and Software Engineering [JSCSE] 2.1 (2012): 1-10.

[21] D. W. a. D. Dean, "Intrusion detection via static analysis ",presented at the IEEE Symposium on Security and Privacy, 2001.

[22] R. K. J. Rabek, S. Lewandowski, and R. Cunningham, "Detection of injected, dynamically generated, and obfuscated malicious code.," presented at the ACM workshop on Rapid malcode, 2003.

[23] M. D. J. Bergeron, M. M. Erhioui, and B. Ktari, "Static Analysis of Binary Code to Isolate Malicious Behavior," presented at the 8th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises, 1999.

[24] M. D. J. Bergeron, J. Desharnais, M. M. Erhioui, Y. Lavoie, and N. Tawbi, "Static Detection of Malicious Code in Executable Programs," presented at the Symposium on Requirements Engineering for Information Security, 2001.

[25] J. X. A. H. Sung, P. Chavez, and S. Mukkamala, "Static Analyzer of Vicious Executables," presented at the 20th Annual Computer Security Applications Conference, 2004.

[26] S. F. S. Hofmeyr, and A. Somayaji, ""Intrusion detection using sequences of system calls," Journal of Computer Security, pp. 151–180, 19.98

[27] Mehdi Bahrami ,"Cloud Template, a Big Data Solution", International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 2, pp. 13-17, 2013, Doi: 10.7321/jscse.v3.n2.2.

[28] D. J. M. a. M. D. Smith, "Host-based detection of worms through peer-to-peer cooperation," presented at the ACM workshop on Rapid malcode, 2005.

[29] Uhr, Patrick, André Klahold, and Madjid Fathi. "Imitation of the Human Ability of Word Association." International Journal of Soft Computing and Software Engineering (JSCSE) (2013).

[30] M. G. M. Debbabi, L. Poulin, M .Salois, , and N. Tawbi, "DynamicMonitoring of Malicious Activity in Software Systems," presented at the Symposium on Requirements Engineering for Information Security, 2001.

[31] D. W. Yanfang Ye, Tao Li, and Dongyi Ye, "IMDS: intelligent malware detection system," presented at the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, 2007.

[32] H. R. Ashkan Sami, Babak Yadegar, Naser Peiravian, Sattar Hashemi, Ali Hamze, "Malware Detection Based On Mining API Calls," presented at the ACM Symposium on Applied Computing-Data Mining Track, Sierre, Switzerland, 2010.

[33] Bahrami, Mehdi, Peyman Arebi, and Mohammad Bahrami. "NetQTM: Node Configuration In Network Setup By Quantum Turing Machine." Internation

conference ICWN. Vol. 11.

[34] L. K. Mohammad M. Masud, and Bhavani Thuraisingham, "A scalable multilevel feature extraction technique to detect malicious executables," Information Systems Frontiers, 2007.

[35] Bahrami, Mehdi, and Mohammad Bahrami. "A Review of Software Architecture for Collaborative Software's." Advanced Materials Research 433 (2012): 2372-2376.

[36] M. C. W. Muazzam Siddiqui, and Joohan Lee, "Data Mining Methods for Malware Detection Using Instruction Sequences," presented at the Artificial Intelligence and Applications, AIA, 2008.

[37] R. G. a. J. L. J. Dai, "Efficient Virus Detection Using Dynamic Instruction Sequences," Journal of Computers, 2009.

[38] S.-J. H. Tzu-Yen Wang, Ming-Yang Su, Chin-Hsiung Wu, Peng-Chu Wang, and Wei-Zen Su, "A Surveillance Spyware Detection System Based on Data Mining Methods," presented at the Evolutionary Computation, 2006.

[39] Bahrami, Mehdi, and Mukesh Singhal. "The Role of Cloud Computing Architecture in Big Data." Information Granularity, Big Data, and Computational Intelligence 8.

[40] Bahrami, Mehdi, Marziyeh Shahrazadfard, and Tooba Kerdkar. "NSSA: A New Enterprise Architecture for Network Setup without Any Network Infrastructure." Intelligent Systems, Modelling and Simulation (ISMS), 2011 Second International Conference on. IEEE, 2011.

[41] Asayesh, Mehrzad, et al. "A Novel Multi Routing in Ad-Hoc Networks Based on Maximum Node Energy." Computer Modeling and Simulation (EMS), 2010 Fourth UKSim European Symposium on. IEEE, 2010.

[42] N. Falliere. (2007, Windows Anti-Debug Reference. Available:

[43] Bahrami, Mehdi, and Leila Pashaie Bonab. "CLS QTM: New Model of Node Configuration In Collaboration Learning Systems By Quantum Turing Machine.", International Journal of Advancements in Computing Technology Volume 3, Number 7, August 2011

[44] H. C. David Lo, Jiawei Han, Siau-Cheng Khoo, and Chengnian Sun, "Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach," presented at the KDD'09, Paris, France, 2009.

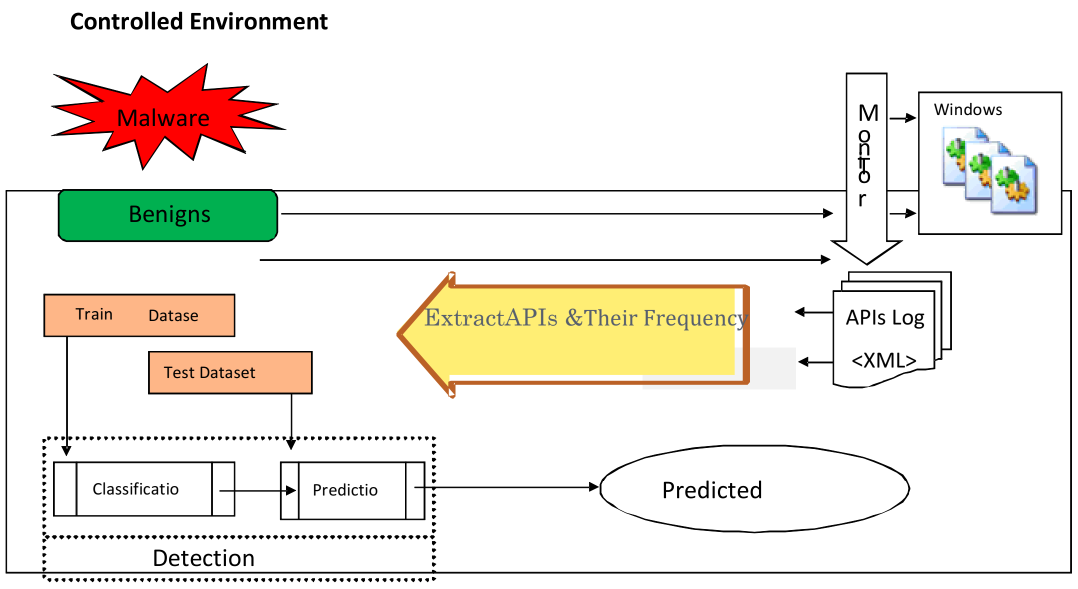

Figure 1. Proposed method

Algorithm 1 Constructing Models with classifiers

Procedure: Model construction

Inputs: PEsDB: A set of malicious and benign files
Output: Classifier: PE classification

```
1: for every PE in PEsDB
2:        PElog = output of PE Monitoring for 2 minutes;
3:        PEslogDB = PEslogDB ∪ PElog;
4: for every PElog in PEslogDB
5:        PESeq = Extract Sequence of APIs from PElog;
6:        PEDataset = PEDataset ∪ PESeq;
7: for every row in PEDataset
8:        F = F ∪ <Name of API, Frequency of the API>;
9: Transform PEDataset into the feature space of F;
10: Classifier = Train a classifier on new PEDataset;
11: return Classifier;
```

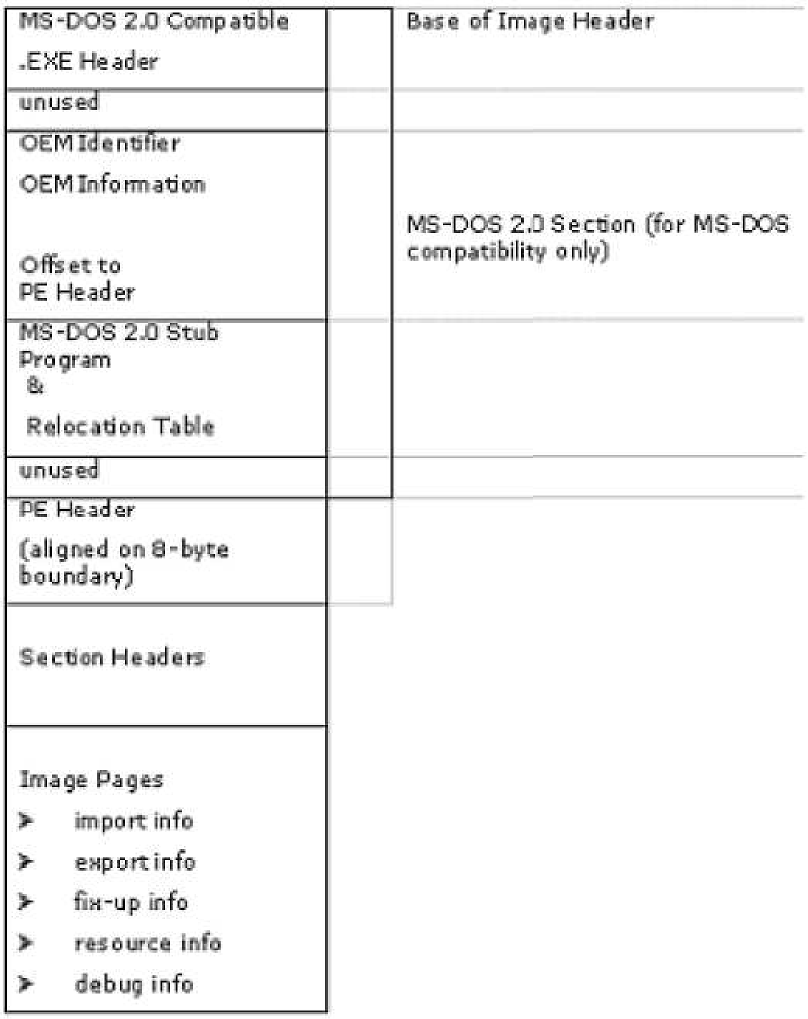

Figure 2. Portable execution file