A SURVEY OF MALWARE DETECTION BASED ON MACHINE LEARNING FOR STATIC PE FILES

Abstract- In this era of digital technology expansion, malware attack becomes a rising threat. Hence, the computer users are aware of data loss due to unknown malware attack. Therefore, it is essential to develop suitable machine learning language code to detect several types of malware attack.In this survey,there is an in-depth study of machine learning classifiers such as Random Forest, Decision tree, linear regression, Adaboost, Gradient Boosting and Gaussian algorithm that can be used to detect benign or malicious file.There are various performance metrics to measure the performance.the aim to select static analysis is to overcome two important drawbacks of dynamic analysis that is scanning overhead and heavy resource consumption.Total 54 features are extracted from binary exectutable using PEFILE.Identification of features set is from 96724 legitimate files and 41323 malicious files. The accuracy of static malware analysis is 99.38%.In the experimental study,common data is used to demonstrate the accuracy,confusion matrix and different performance metrics.the essential element to measure performance is confusion matrix to detect malware.

Keywords : Malware detection,PE files,machine learning,static analysis

I. Introduction

Malware attack becomes a rising threat in accordance with the development of internet. Using several software programs anonymous hackers is performing several malicious activities over the internet to steal confidential information from any database. With this increasing amount of malicious software programs, anti-virus programs are not able to fulfil the needs of giving full protection to the private network and database. According to the report of Kaspersky Lab, in 2015 approximately 4000000 unique malware programs have been identified that are responsible for almost 6563145 host attacks. Therefore, it is required to implement updated security software program using python. Hence, by viewing this concerning threat, this study is going to evaluate the use of the best possible machine learning algorithm to detect malware programs.

II. machine learning algorithms used for static malware detection

Machine learning algorithm is the creation of computer programming algorithm that can work according to the decision made by artificial intelligence (AI). Hence, this machine learning algorithm is used to execute a program automatically to serve the purpose. In this context, in order to detect newly developed malware programs using python PE file structure, the following algorithms can be used:

Linear Regression

This algorithm is used to make a prediction based on the independent variables. Hence, with the help of this linear regression algorithm it is possible to find a linear relationship between the dependent (y) and independent (x) variables [1].

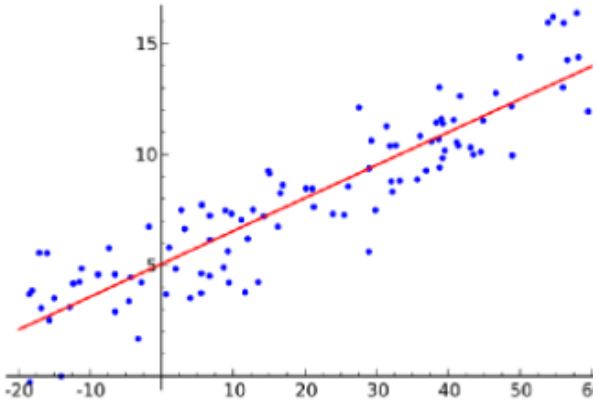

Figure 1: Linear Regression

Hence, by finding the relationship between dependent and independent variables, a regression line can be drawn. Hypothesis of this algorithm can be developed by the equation:

$y = \theta 1 + \theta 2 * x$

Where $\theta 1$= intercept

$\theta 2$= coefficient of x

Hence, by determining the best possible $\theta 1$ and $\theta 2$ values, it can be possible to construct the best fitted line. Hence, by constructing this regression line, it can be possible to minimise the error while predicting the values. In this context, by creating the best fitted regression line it can be possible to detect the malware programs with best prediction.

Random Forest

Random forest classifier has been used to make choice of most accurate decision. In this algorithm, it can be possible to get more accurate decision based on higher number of trees exists in the forest.

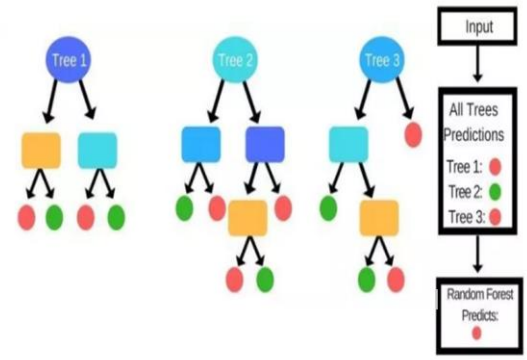

Figure 2: Random Forest Algorithm

In order to make a valid prediction, this algorithm serves a simple but worthy task to develop a decision by evaluating the decisions of each individual decision-making tree. Hence, by the analysis of each individual decision, it can be possible to construct such a decision that can get maximum votes. Hence, this algorithm seems best suitable for the detection of malware, as each of the individual trees produces the best outcomes by eliminating individual errors. Hence, with the use of this algorithm, more accurate results can be obtained. Moreover, in this algorithm, wrong decision made by some of the trees can be mitigated by the effective prediction made by the other trees [2]. Hence, the overall decision made with the help of this random forest algorithm seems accurate enough. The performance of random forest was observed to be more escalated based upon standalone classification. Random forest selects randomly selects subsamples and trains itself for reduction in classification error.

Random forest technique has shown extensive and stable performance in higher dimension such as UCI public datasets.[20]

Decision Tree

In order to solve classification and regression problems, this decision tree algorithm is used. Hence, with the help of this algorithm all the attributes of a problem can be presented by each node of the decision tree and each leaf node is assigned to a class label.

For the prediction of the best decision using this decision tree, it is required to start from the root of the tree. Thereby, this root attribute values are compared with the other recorded attributes. Hence, by a sequential process, it is required to compare each of the attributes by following a branch of the tree [3]. During this comparison, it can be possible to jump to the next node in case; the obtained accuracy is maintained to an acceptable level.

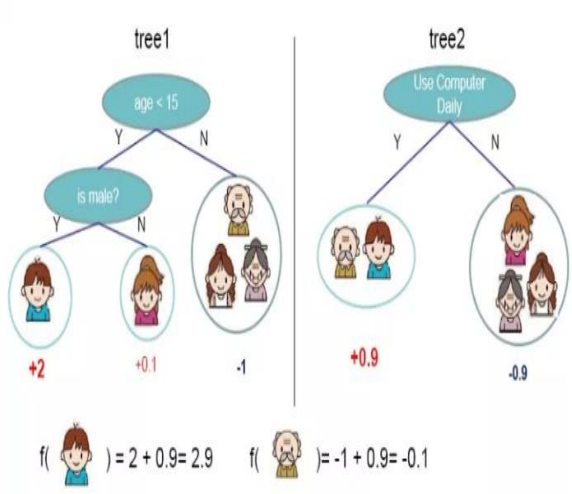

Figure 3: Example of decision tree algorithm

From the above example it can be sensed that, accurate decision can be made to measure the number of computer users (male or female users) by analysing each of the nodes of this decision tree.

Most decision algorithm will not performing good when problem having diagonal separating[21].

AdaBoost

This Adaboost or Adaptive Boosting algorithm can be used to make a viable decision within a short period of time. After developing a decision tree, this algorithm can be used to evaluate the performance of the tree by analysing each of the instances used in the tree. Hence, this AdaBoost algorithm is mainly used to boost the performance of decision tree with the help of binary classification [4]. Therefore, this algorithm can be used in case of detecting malware programs within a short period of time.

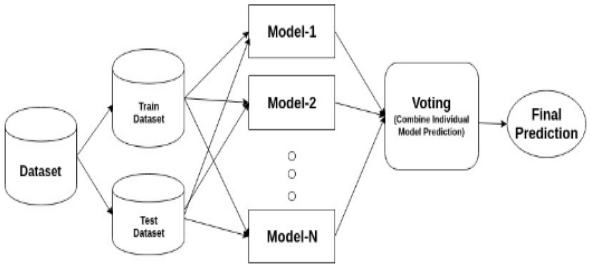

Figure 4: AdaBoost algorithm

With the use of this algorithm, dataset of each individual instance has been weighted to make the prediction process faster. Dataset of each instance can be measured by the following calculation:

Weight (xi)= 1/n

Where, n= number of instance

xi= i'th number of instances

Gaussian

In order to develop a viable prediction by the reduction of rows in linear equation, this algorithm has been used. Hence, in this algorithm, an augmented matrix has been developed from a linear equation. Thereafter, by performing elementary row operation, the augmented

matrix is compressed to an upper triangular matrix. In the machine learning process, this algorithm is used to predict a value from unseen dataset with the use of kernel function. Kernel function is used to determine the similarities between several unseen points of dataset [5]. Thereby, with the help of this algorithm principle, it can be possible to predict the best possible outcomes in a lazy learning process.

Gradient Boosting

Boosting is the process to convert the weak elements into a strong element. In this boosting, in order to get entry to the dataset, each of the new trees requires a modification to become best fitted. Hence, in this algorithm, it can be possible to evaluate the weight of the dataset of each of the tree for making a viable prediction [6].

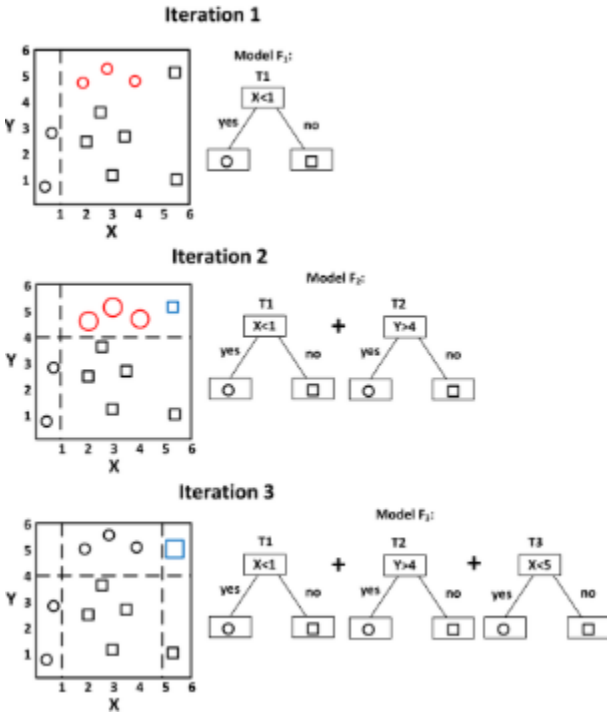

Figure 5: Gradient boosting algorithm

Hence, based on the weight classification error can be calculated easily by increasing or decreasing the weight of each of the decision tree accordingly.

III. PE file structure

The PE or portable executable file structure is a data structure that contains essential information for the operating program that can help to manage executable code. There are mainly two parts of a PE file format such as header and section.

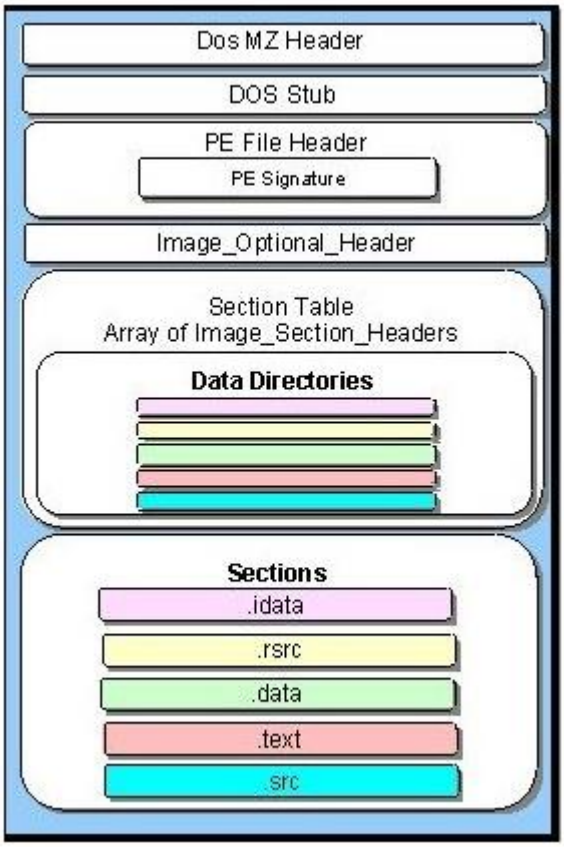

Figure 6: PE file format

DOS Header

Each PE file contains a DOS header of 64 bytes. The most common structure of a DOS header is ifanew and magic. In order to identify compatibility with MS-DOS file type, e_magic is used. Offset of this header is set to 0 [7].

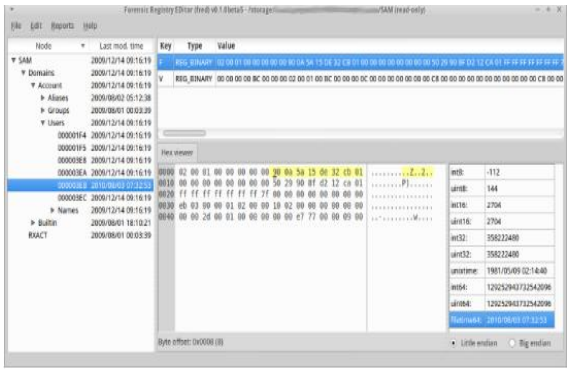

Figure 7: DOS Header

However, this offset value can be varied in CFF explorer. On the other hand, ifanew is used to convert the exe file into PE file. This ifanew can help to jump into the PE header by avoiding the DOS stub [8].

DOS Stub

In order to print a string this DOS stub is generally used. This DOS stub can help to develop an application for the Windows operating system. This DOS stub can be executed on the Windows platform rather than running in DOS mode. An instruction called winstub.exe has been sent to an executable file. In this executable file, binary instructions have been stored with the help of linker. This executable file is stored in 0x3c address [9].

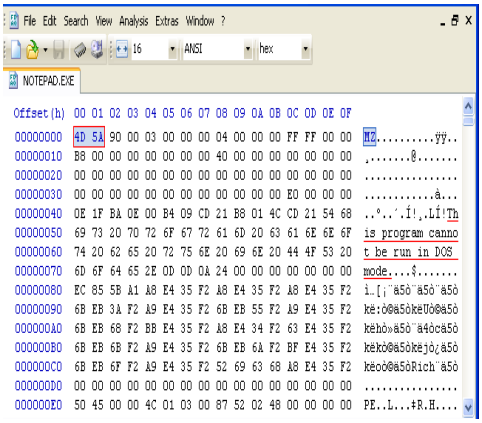

Figure 8: DOS stub

This address is the offset of the PE header that is next to this address. This PE header address starts from the address F8000000 and its offset 000000F8. Therefore, by knowing the starting address of PE header and its offset address, it can be possible to determine the correct address to start the execution [10].

PE file Header

PE file headers consist of location and size of the file like other file format. Among total memory size of the PE header file, DOS stub has taken first few hundred bytes of memory of this PE header. In the DOS header, this PE file has been stored with the idex name e_ifanew [11].

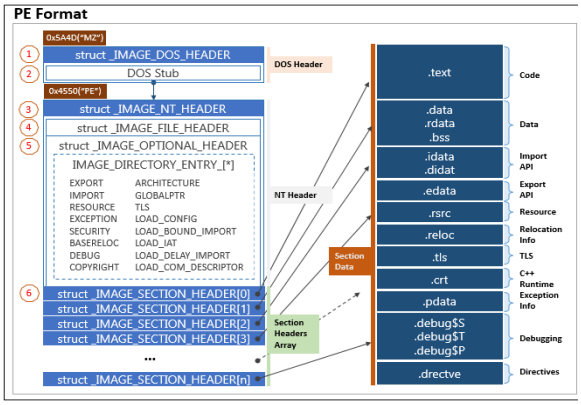

Figure 9: PE file header

Hence, actual memory-mapped address can be determined by viewing the file location in MS DOS header. PE file header has the following characteristics:

Signature- With the help of P.E. followed by two 0's is used to address the signature

Machines- With the help of a number, it represents the type of systems such as AMD and Intel. The number structure with respect to the system is defined as the following:

Intel i860- 0x14d

NumberOfSections- Size of the section table can be addressed by this characteristic. It has been placed immediately after the PE header [10].

SizeOfOptionalHeader- in case of an executable file, it is required to define the size of any optional header. Hence, this optional header has been placed in between the starting of the section table and top of the optional header [11].

Section table

The address of the section table starts immediately after the optional header. Hence, the starting location of the section table can be determined by measuring the location of the first byte after the ending address of the header [12]. The section table header can be structured as follows:

IMAGE_SECTION_HEADER

This section header consists of a minimum of 40 bytes of size in order to record the entries in the section table [10]. These entries are given by the noofsectionfield. Some of the entries in the section table have been given as the following:

Name- It is an 8-byte encoding string

VirtualSize- It determines the size of section data

SizeOfRawData- Section data size of the disk file

PointerToRawData- It is the offset from starting of file to the section data

PE file section

In this section, the body content of a file can be stored. In each of the section data, there is a PE file header and body of the file in which all the executable files; code and program resources can be stored. There are nine predefined sections such as .bss, .text, .data, .rdata, .edata, .idata, .pdata, .tstc and .debug in an application, among which some of the data are used to utilise the memory at the time of execution of a file [12].

IV. Static PE files Feature extraction

A static PE file is generally an executable file or a file with binary code that can help the other executable files for simulation. Based on the PE file structure, extracted features can be classified into four categories such as the following:

PE Header Information

In the PE file header section, there is information regarding file execution process in the Windows platform. Therefore, in the header of a PE file, information of known features of any size can be stored. Hence, from the PE header, it can be possible to extract the same set of features for a large number of PE files [13].

Section names and characteristics

In this section the actual body elements of a PE file can be stored including, file configuration, string, binary code and pictures. In a static PE file, there are similar sections as name, order and characteristics. Based on evasion technique any existing malware can be detected by viewing the changed file structure [14]. Some of the section name and the name of their respective builders or compilers have been given as the following table:

Table 1: SECTION NAME AND THEIR COMPILERS

Section Names	Builders/Compilers/ Packing Infrastructure
Mpres	Mpres Packer
Upx	UPX
Tsuarch	TSULooder
Petite	Petit packer
Rmnet	Ramnit Packer
Nsp	NsPack packer
Aspack	Aspack packer
Themeda	Themida
Adata	Armadillo packer
Vmp	VMProtect
Mew	MEW Packer
Neolit	ImpRec-created section

Section entropy

There are several packed data in the PE file system that is known as entropy. In order to avoid malware attack, actual code of a PE file has been stored with encrypted format. There are several known packet utilities that are used by malicious programs such as UPX, MPress, VMProtect and Themida [15]. Hence, one of the above-mentioned algorithms is used to detect the malware programs in the body of a PE file.

PE imports

It is another feature of a PE filer that it can import program code from other PE file. There are some potential malicious imports:

Table 2: POTENTIAL PE CODE IMPORTS

Import Names	Potential Malicious Usage
KERNEL32.DLL!MapViewOfFile	Code Injection
KERNEL32.DLL!IsDebuggerPresent	Anti-Debugging
KERNEL32.DLL!GetThreadContext	Code Injection
KERNEL32.DLL!ReadProcessMemory	Code Injection
KERNEL32.DLL!ResumeThread	Code Injection
KERNEL32.DLL!ResumeThread	Code Injection
KERNEL32.DLL!WriteProcessMemory	Code Injection
KERNEL32.DLL!SetFileTime	Stealth
USER32.DLL!SetWindowsHookExW	API Hooking
KERNEL32.DLL!MapViewOfFile	Code Injection
ADVAPI32.DLL!CryptGenRandom	Encryption

All of the above features can be stored in the data library. There are some well-known libraries that contain PE code which is used to execute a PE file with the help of python programming language [16]. These python-based libraries are as follows:

Pandas- Easy data structure has been stored in this open-source library

NumPy- This library consists of multidimensional array objects that help to perform mathematical and logical operation in faster way [17].

Pickle- In order to perform serialize operation of an object structure, a python object is converted into a character stream

SciPy- This open-source python library is used to hold the codes for mathematical and engineering operations

Pefile- PE file header information has been stored in this multi-platform python library

Scikit- This free machine learning python-based library consists of various algorithm features such as random forest, support vector and k-neighbours [18]

V. Experiments and results

In order to detect malware using PE file structure, it is essential to develop the confusion matrix. Therefore, it can be possible to calculate TPR and TNR from the FPR and FNR values. All these values are given in the following table [17]

Table 3: CALCULATION OF TRUE POSITIVE RATE

Measures	Percentage (%)
FPR	0.534094
FNR	0.792793
TPR	0.207207
TNR	0.465906

TPR= True Positive Rate

TNR= True Negative Rate

FPR= False Positive Rate

FNR= False Negative Rate

TPR= 1- FNR

TNR= 1- FPR

Therefore, in this context, *accuracy or efficiency* of all used algorithms have been given as the following:

Table 4: ACCURACY OF USED ALGORITHMS

Algorithm	Accuracy (%)
Gaussian	69.85
Decision Tree	99
Random Forest	99.48
AdaBoost	98.59
Gradient Boosting	98.84

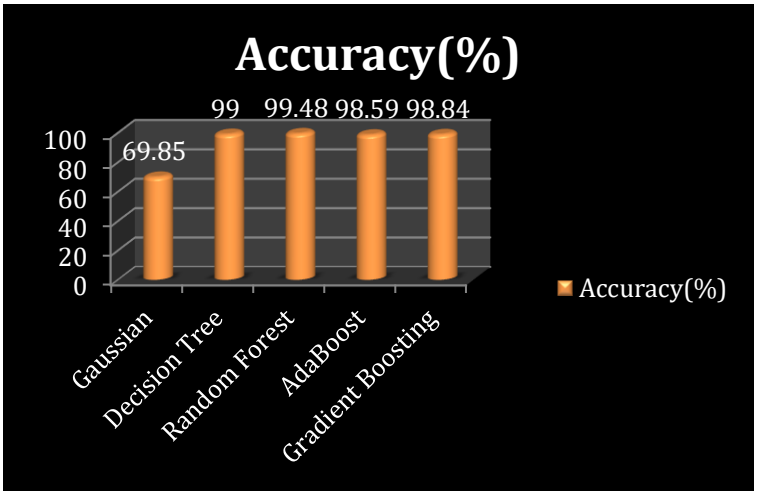

Figure 10: Accuracy percentage

Hence, from this experiment, it can be sensed that, RandomForest algorithm seems best suitable for the detection of malware program, as it has given the best accuracy to detect malware programs.

Confusion matrix can be formed as follows:

5: FORM OF CONFUSION MATIX

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

TP= True Positive

TN= True Negative

FP= False Positive

FN= False Negative

In this context a total of 54 features are used,

Here, n= 27610

Table 6: CALCULATION OF CONFUSION MATRIX

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	19260	85
Class 2 Actual	57	8208

Accuracy of an algorithm can be calculated by the below-mentioned formula:

Accuracy= (TP+TN)/ (TP+TN+FP+FN) %

Therefore, in this context, accuracy of this malware detection process has been determined as the following:

Accuracy=(19260+8208)/(19260+8208+57+85) = 99.485%

Precision of an algorithm can be calculated by the below mentioned formula:

Precision= (TP)/ (TP+FP) %

Therefore, in this context, precision of this malware detection process has been determined as the following:

Precision=(19260)/ (19260+57) = 99.70%

Recall of an algorithm can be calculated by the below mentioned formula:

Recall= (TP)/ (TP+FN) %

Therefore, in this context, precision of this malware detection process has been determined as the following:

Precision=(19260)/ (19260+85) = 99.56%

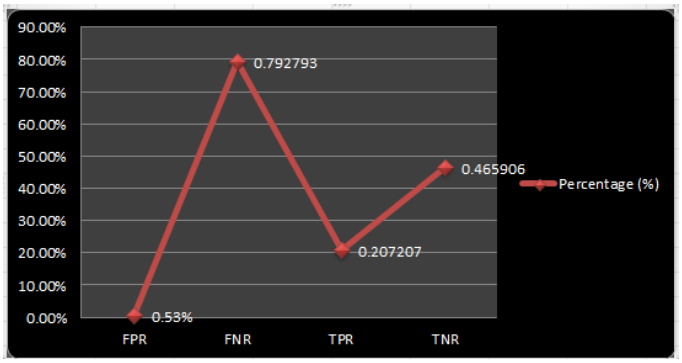

Figure 11: FPR, FNR, TPR and TNR percentage

VI. Conclusion

It has been sensed from the above study that, detection of malware program using PE files can be performed with the help of several machine learning algorithms such as RandonForest, Decision tree, linear regression, Gaussian, Gradient Boosting and AdaBoost algorithm. Hence, by applying one of these or by applying all of these

above mentioned algorithms it can be possible to detect the presence of malware program. Moreover, by measuring the accuracy level of each of the algorithms, the best suited one has been chosen for performing the task. In this context, RandomForest seems the best algorithm to run, as this algorithm gives the highest level of accuracy (99.478450%). However, in every context, a single algorithm is not sufficient for all types of malware detection operation and thereby, proper algorithm has been selected based on the accuracy level. Furthermore, in order to determine the accuracy level, it is essential to develop the confusion matrix with four parameters such as TP, TN, FP and FN. all of these parameters are essential to determine the percentage of accuracy of an algorithm while detecting a malware program.

References

[1] Zhang, Z., Lai, Z., Xu, Y., Shao, L., Wu, J. and Xie, G.S., 2017. Discriminative elastic-net regularized linear regression. *IEEE Transactions on Image Processing*, *26*(3), pp.1466-1481.

[2] Lin, W., Wu, Z., Lin, L., Wen, A. and Li, J., 2017. An ensemble random forest algorithm for insurance big data analysis. *Ieee Access*, *5*, pp.16568-16575.

[3] Dai, Q.Y., Zhang, C.P. and Wu, H., 2016. Research of decision tree classification algorithm in data mining. *International Journal of Database Theory and Application*, *9*(5), pp.1-8.

[4] Lu, K., Zhang, W. and Sun, B., 2017. Multidimensional data-driven life prediction method for white LEDs based on BP-NN and improved-adaboost algorithm. *IEEE Access*, *5*, pp.21660-21668.

[5] Cheng, R., Jin, Y., Narukawa, K. and Sendhoff, B., 2015. A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling. *IEEE Transactions on Evolutionary Computation*, *19*(6), pp.838-856.

[6] Zhang, F., Du, B. and Zhang, L., 2015. Scene classification via a gradient boosting random convolutional network framework. *IEEE Transactions on Geoscience and Remote Sensing*, *54*(3), pp.1793-1802.

[7] Anderson, H.S. and Roth, P., 2018. Ember: an open dataset for training static PE malware machine learning models. *arXiv preprint arXiv:1804.04637*.

[8] Narouei, M., Ahmadi, M., Giacinto, G., Takabi, H. and Sami, A., 2015. DLLMiner: structural mining for malware detection. *Security and Communication Networks*, *8*(18), pp.3311-3322.

[9] Belaoued, M. and Mazouzi, S., 2015, May. 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* (pp. 416-425). Springer, Cham.

[10] Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C. and Roli, F., 2018, September. Adversarial malware binaries: Evading deep learning for malware detection in executables. In *2018 26th European Signal Processing Conference (EUSIPCO)* (pp. 533-537). IEEE.

[11] Yang, J.H. and Ryu, Y., 2015. Design and development of a command-line tool for portable executable file analysis and malware

detection in IoT devices. *International Journal of Security and Its Applications*, *9*(8), pp.127-136.

[12] Anderson, H.S., Kharkar, A., Filar, B., Evans, D. and Roth, P., 2018. Learning to evade static PE machine learning malware models via reinforcement learning. *arXiv preprint arXiv:1801.08917*.

[13] Soeder, D.A., Permeh, R., Golomb, G. and Wolff, M., CylanceInc, 2016. *Static feature extraction from structured files*. U.S. Patent 9,262,296.

[14] Sun, B., Li, Q., Guo, Y., Wen, Q., Lin, X. and Liu, W., 2017, December. Malware family classification method based on static feature extraction. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)* (pp. 507-513). IEEE.

[15] Mithal, T., Shah, K. and Singh, D.K., 2016. Case studies on intelligent approaches for static malware analysis. In *Emerging Research in Computing, Information, Communication and Applications* (pp. 555-567). Springer, Singapore.

[16] Awan, S. and Saqib, N.A., 2016, November. Detection of malicious executables using static and dynamic features of portable executable (pe) file. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage* (pp. 48-58). Springer, Cham.

[17] Machine Learning (2018).Machine learning repository : https://github.com/prk54/malware-detection-machine-learning-approach.

[18] Soeder, D.A., Permeh, R., Golomb, G. and Wolff, M., CylanceInc, 2018. *Static feature extraction from structured files*. U.S. Patent 9,959,276.

[19] Hassen, M., Carvalho, M.M. and Chan, P.K., 2017. Malware classification using static analysis based features. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*(pp. 1-7). IEEE.

[20] C. Barreyrea, B. Laurent, J.-M. Loubesc, B. Cabona and L. Boussouf,2017,December. Multiple testing for outlier detection in functional data. HAL Id: hal-01651191.https://hal.archives-ouvertes.fr/hal-01651191

[21] S. B. Kotsiantis, 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering, Supervised Machine Learning: A Review of Classification Techniques (pg 3-24) ACM.

Appendix

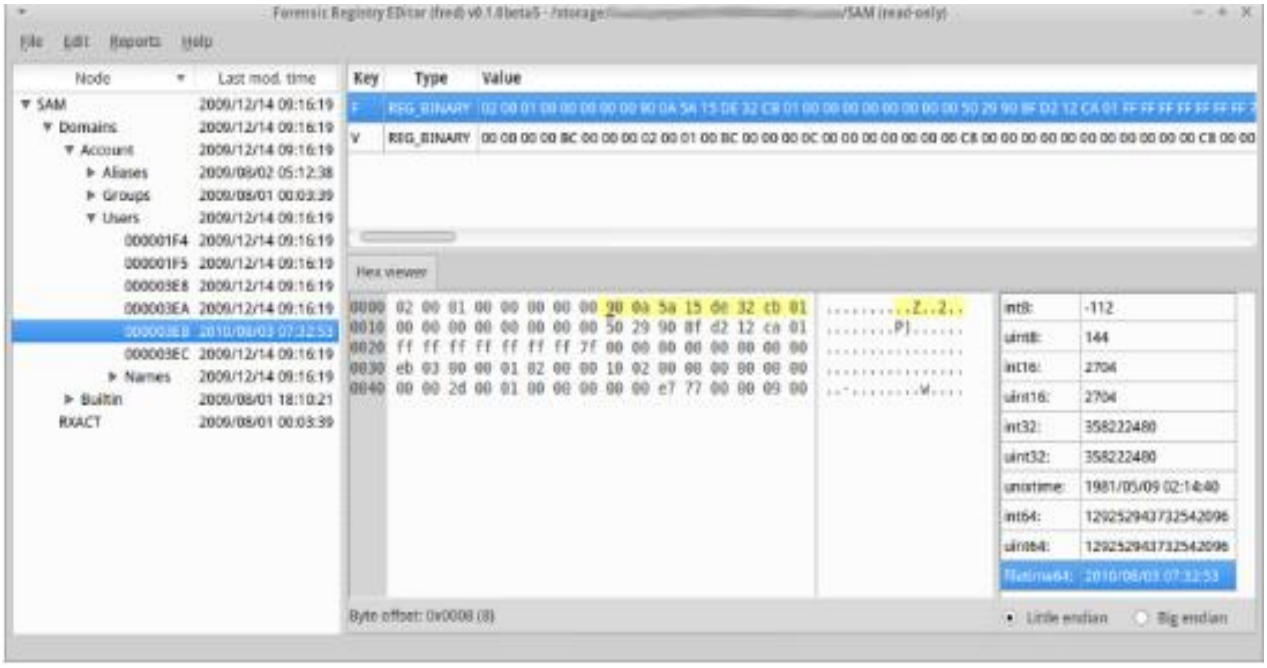

Figure 7: DOS Header

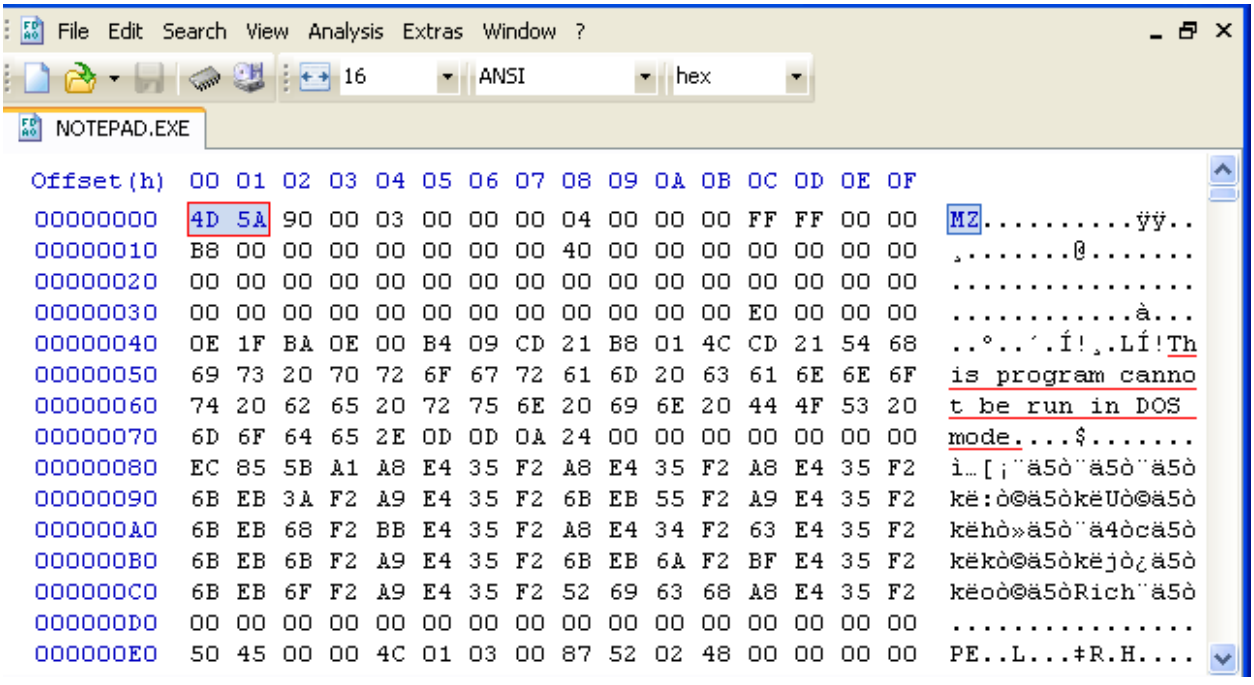

Figure 8: DOS stub

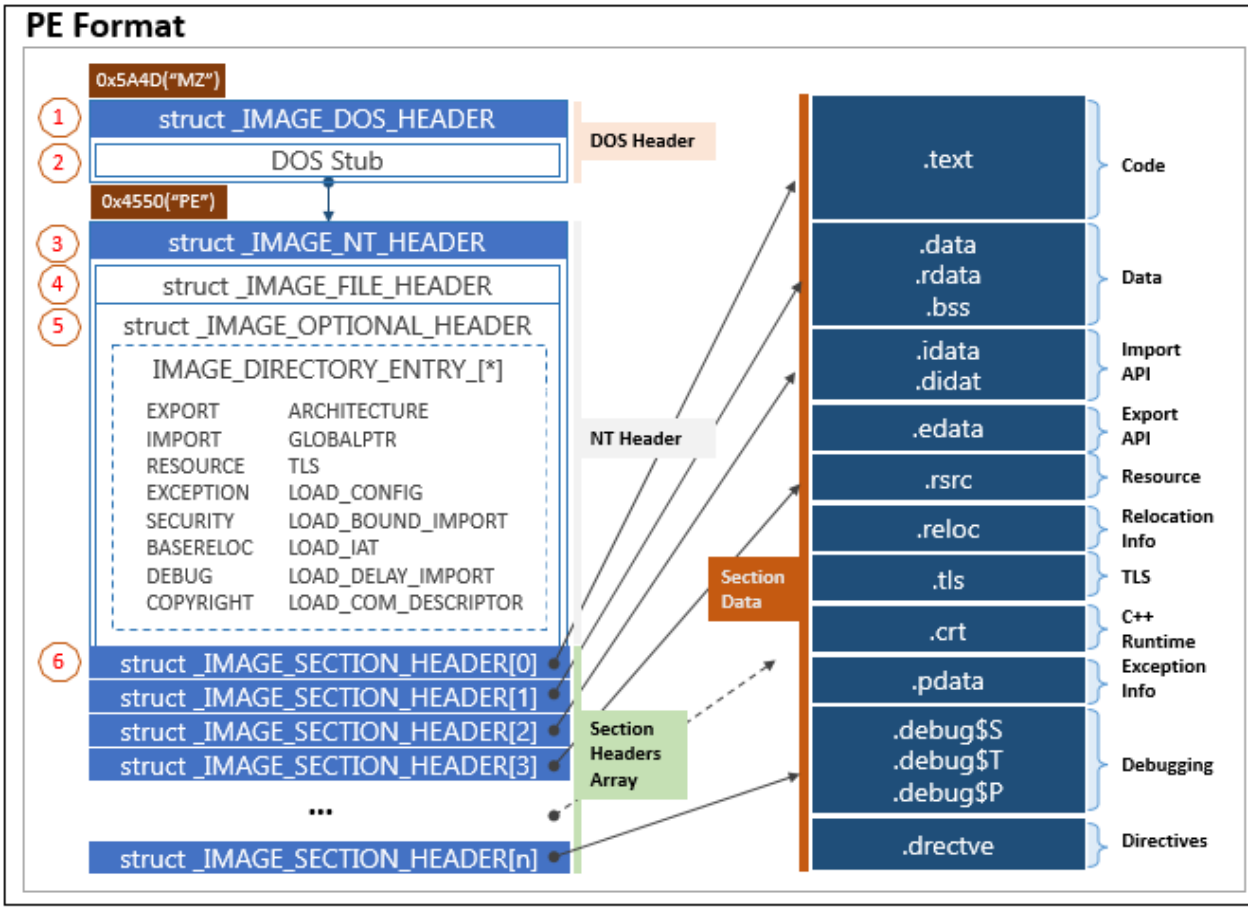

Figure 9: PE file header