Review of Machine Learning Methods for Windows Malware Detection

Saima Naz
Computer Science and Engineering Department
MNNIT Allahabad
Prayagraj, India
saimanaz0011@gmail.com

Dushyant Kumar Singh
Computer Science and Engineering Department
MNNIT Allahabad
Prayagraj, India
dushyant@mnnit.ac.in

***Abstract*—Malicious software or malware is one of the most critical cyber threats intentionally designed to cause damage, disrupt and gain unauthorized access to the system. The system can be a computer, server and computer network. Windows operating systems are widely used operating systems. It is easy for hackers to spread malware in these operating systems and exploit its vulnerabilities. Malware detection has always been a challenging issue and major concern for the data privacy. Many signature-based malware detection methods have been presented that work at a certain level and fail to detect unknown malware executable files, therefore the aim is to investigate a novel approach which can detect the new and unseen malware. In this paper, a simple and efficient malware detection model is introduced which distinguish between benign and malicious executable files by extracting features from the PE (portable executable) headers. Different machine learning methods such as Support Vector Machine, Decision Tree, Random Forest and Naive Bays classifiers are used for the classification. Random forest classifier among the different classifiers has achieved the highest accuracy result with the dataset of file, optional and section header.**

***Index Terms*—Windows, Portable Executable (PE), Features, Static Analysis, Malware, Benign, Classification.**

I. Introduction

Nowadays, the demand of windows operating systems is increasing rapidly. Windows operating system provides an easy interface to deal with computer operations so that a normal person can perform functions easily. Windows operating systems are very popular due to the availability of numerous features such as hardware features independence, support for third party software, open source software, entertainment purposes and many other features. However, the popularity of windows OS attracts the attackers to spread new types of malware on these systems to compromise the confidentiality and integrity of our data.

Malware is malicious software that has been specifically designed to attack the systems. The main aim of spreading the malware is to steal the sensitive and confidential information. It is a piece of code/program which is purportedly designed to disrupt the functioning of the system and can create digital chaos [1,2]. Malware is generally written to serve the purpose of gaining remote access to the infected system by the attacker. Malware can be theoretically divided into several categories according to their malicious goals and behaviours. Adware, Scareware, Trojan, Spyware, Virus, Botnet and Worms are some of the malware types depending on their working and host type.

The Internet has become an important part of our everyday lives and we are becoming dependent on it. However, its increasing use has also left the attackers to misuse and abuse it. In the internet world, there is a dramatic increase in the spread of malware on applications such as web browsers, media players and many other applications. When a user installs an application, it asks the user to accept all the permissions. Attackers usually ask for the permission through which they can access the private information. Users are unaware of this purpose of attackers and they accept all the permissions at the time of application installation and then they become the victim of the attack. Hence, we need some method to protect the confidential data from any threat.

It is found out that many researchers used static analysis [1,3,17,18] for the malware detection and many researchers used dynamic analysis [2,4,6,7]. The static analysis which is also called signature based analysis finds the malicious behaviours in binary source code segments without executing the applications. Dynamic analysis method can detect malware while execution of malicious application. It is generally performed by observing the malware characteristics while running on a host system. This type of analysis is generally performed in a sandbox environment to prevent the malware from actually infecting production systems. Dynamic analysis consumes more resources and also having the high cost. Static analysis is safer than dynamic analysis because the file is not executed and cannot result in bad system consequences.

Current malware detection techniques are often deficient and fail to detect new types of malware. It is a serious concern that despite the development of anti-malware method there is growth of new malware and is continuously growing day by day. Therefore, effective and efficient automated malware detection and response techniques are of critical importance to the data safety.

In this paper, we proposed a simple, efficient and effective malware detection system to classify malware families by taking the features from PE header and Section table of the executable file. The system is able to detect new and unseen malware in the executable file. The key executable file format for windows operating systems is the PE file format. It is used for the variety of purposes. It is Microsoft centric including DLLs [16]. Executable files are dissembled with the IDA-pro dissembler for debugging and dissembling purposes. In this process, binary instructions are converted to code mnemonics and higher level constructs through which desired and important features are extracted.

The proposed model is designed based on static analysis approach. The model is divided into four main parts: Sample Collection, Feature Extraction, Splitting the dataset and Executable file classification. It detects malware before execution of the executable file. For the malware analysis different supervised machine learning based classification methods are used to classify whether the source executable file belongs to benign class or malware class. The performance of the proposed system is evaluated by using the different performance evaluation metrics such as precision, recall, accuracy, true positive rate and false positive rate. Among the different classification methods, random forest classifier has achieved the highest accuracy with combined features of file, optional and section headers.

Rest of the paper is organized as follows: Related work is shown in section 2 where brief overview of different proposed models on malware detection is given. In the section 3, the methodology of the proposed model is explained. Experimental results and comparative analysis is presented in section 4 and section 5 respectively and at last section conclusion is given.

II. Literature review

Researchers proposed many different malware detection techniques in windows operating system. Mithal, et al. [1] proposed a system in which mnemonic n-grams, strings and other features are extracted from the executable file and system is prepared with machine learning methods to detect the malware in windows OS. In a proposed framework [2], authors presented the comparative study with different machine learning based classifiers for the malware detection. Alsulami, et al. [3] proposed a malware detector system. In this system, authors used list of dependency file of prefetch files of windows operating system to differentiate the normal and malicious windows applications. Satrya, et al. [4] introduced a model which can detect 8 types of botnet malware using hybrid analysis that is the combination of both static and dynamic analysis. This model detects malware in the executable files of windows operating system. To detect the unknown malware in windows OS, authors in [5] proposed an active learning framework. SVM machine learning classifier used in this framework.

Rehman et al. [6], recently proposed an android malware detection system where constant strings of the APK (Android Package Kit) file compared with the constant strings of malicious application. If no comparison found between the constant strings, then the user will be notified about the infected application. Extracted keywords and manifest.xml files of the application are compared to know about the status of whether it is malicious or benign. A model named EnDroid [7] introduced based on dynamic analysis. The implementation of this model is based on different types of dynamic features. Here, critical features are extracted that helped to recognise the risky behaviours of the applications and system-level behaviour. After that Stacking applied to implement malware detection model.

For the fast and dynamically detection of malware, a hybrid-multi filter framework [8] investigated. This framework designed by using complementary filters to recognize the run-time behavioural characteristics of malware. The proposed system has developed different types of hybrid algorithms by combining filter approaches like fisher relief F-score, minimum-redundancy and maximum relevance with the SVM wrapper. A heuristics approach [9] proposed for data logs of mine behavioural for the mobile malware detection. Model improves the cyber-resilience which is based on the user behaviour of mobile devices. It is designed to automatically analyse, detect the malware and notifies when malware found in the mobile device. An android malware detection model recently developed which is based on block-chain technology. The model named as consortium block-chain framework [10] works for the detection of public chain shared by users and consortium chain shared by test members. It addresses the issue of detection of malicious codes and extraction of corresponding evidences in mobile devices. For automated detection of malware in android devices, a model [11] introduced which is based on deep learning and API calls of android app packages. One proposed framework [12] detects malware dynamically. In this framework, authors used the run time behaviours of android application by collecting the frequency of applications system calls.

Shah, et al. [2], introduced a dynamic analysis for malware detection by using the API calls of executable files. Authors used the classification methods for the malware classification. Shijo et al. [14] proposed a method to analyse and then classify an unknown executable file. Method is implemented with both the static and dynamic analysis approaches. The feature set is prepared by extracting binary code from executable file through static analysis. Behaviour of executable files are analysed during run time execution which comes under dynamic analysis. Then machine learning classification methods applied for the classification and detection of malware. In the malware analysis survey [15], it is shown that many malware detection novel techniques have been proposed in the literature. Every technique has its own importance and all the techniques having some advantages and disadvantages for detection of malware and other security vulnerabilities. Awan [16] proposed a malware detection method by using static and dynamic features extracted from executable file. In [17, 18], authors presented the work based on static malware analysis. According to the methodology [19], authors used the opcode n-gram, gray-scale image techniques and import functions for the feature extraction process.

We designed an efficient method for the malware detection in windows operating systems. We extracted the different types of features from the executable file. Those features are used as input to the different machine learning based classifiers to classifies the malware executable files. The motivation of the proposed work is to find out the unknown malware in the executable file.

III. Proposed Method

The proposed system works on static analysis approach and detects malware before installation of the executable files. The work is accomplished in four main steps: (1) sample collection, (2) feature extraction, (3) splitting the dataset and 4) Executable file classification. Fig 1. illustrates all four steps of the work. Analysis of malware is done by comparing the extracted features of the portable executable file. For the feature extraction, recent researches are relying on prior knowledge of PE file header. Techniques such as n-datagram, grayscale and many others are used for the feature extraction. In our model, we considered the PE file header as a whole to evaluate the features from the executable files. Then these features are used to build the classifier which helps to determine whether the file is malicious or not.

Dataset of features is collected by extracting the features from samples of malicious and benign files by dissembling these sample files through the dissembler. The dataset is given to the data-processing phase and processed for training-testing purpose. To classify the malicious and benign features different classification algorithms are applied and accuracy is evaluated. During the experimentation, it is obtained that for each of them Random forest classifier has achieved the highest accuracy result. Hence, model is prepared on the random forest classifier. The detailed discussion is given in the upcoming sub-sections.

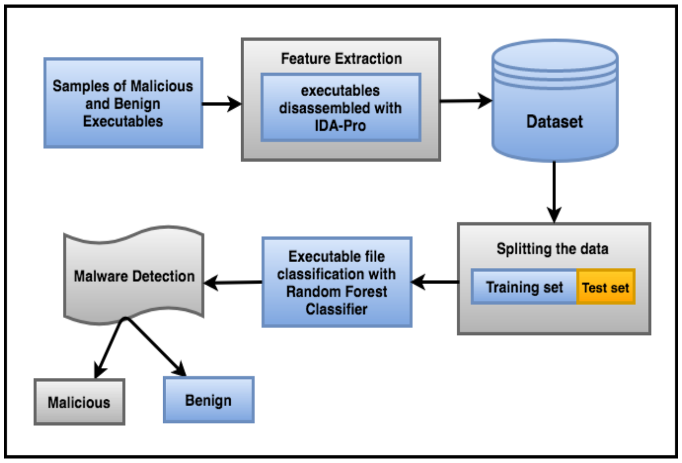
Samples of Malicious and Benign Executables
Feature Extraction
executables disassembled with IDA-Pro
Dataset
Splitting the data
Training set
Test set
Executable file classification with Random Forest Classifier
Malware Detection
Malicious
Benign

Fig. 1. Proposed machine learning malware detection model

A. Experimental Setup

In order to setup the experiment, there is need to create a big representative set of executable files. We downloaded 1530 malicious executable files from Virusshare and Vxhaven websites and 1340 benign executable (.exe and .dll) files are collected from System 32 folder of windows 7 and windows 8.1 operating systems. Both samples of executable files scanned through the virus-total online tool to facilitates the quick detection of worms, viruses, trojans, and all kinds of malware.

B. Feature Extraction

Malware and benign samples are disassembled through IDA-Pro disassembler. We analysed the function flow chart, function blocks and many other things from the dissembled portable executable files. And collected the large amount of information from the dissembled executable file and various properties are stored to understand the executable file. Set of features is formed by extracting values from fields of three main headers: file header, optional header and section header presented in every executable file. Total 40 features are evaluated. Fig 2 illustrates the process of extracting features from the PE headers for each malicious and benign executable file.

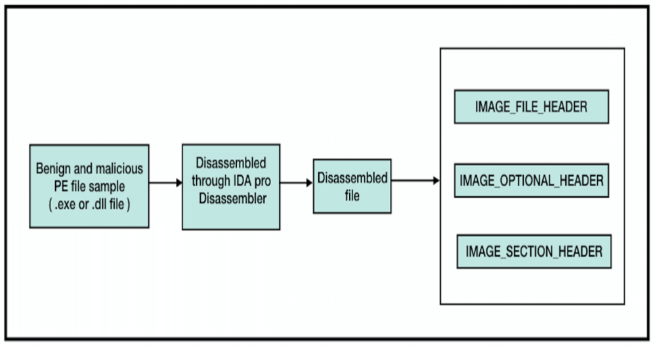
Benign and malicious PE file sample (.exe or .dll file)
Disassembled through IDA pro Disassembler
Disassembled file
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_SECTION_HEADER

Fig. 2. Extracting features from PE file

IMAGE_FILE_HEADER, IMAGE_OPTIONAL_HEADER and IMAGE_SECTION_HEADER of structure type components are analysed from the disassembled file. We developed a module by using the **pefile** library of Python in order to extract features from these analysed header components. The first few hundred bytes of the typical PE file are taken up by the MS-DOS stub header.

File header has the abstract information about the whole file and comes after the MS-DOS stub. In the file header NumberOfSections, SizeOfOptionalHeader and Characteristics are the important fields of the structure type component IMAGE_FILE_HEADER. In this proposed work, values from these important fields are extracted as features. **NumberOfSection** field denotes the number of sections presented in the section header.

Optional Header is required in every executable file; it is not the optional header for the PE file. It contains the information about how the binaries are loaded such as what is the size of the data segment, the amount of stack to reserve and many other things. Fields of the component IMAGE_OPTIONAL_HEADER contains the critical information. The first eight standard fields contain information which are useful for loading and running the executable file and the 21 fields are the windows specific fields. In windows OS, these fields contain additional information required by the linker and loader.

ImageBase and the subsystem fields are the most important fields of the optional header. **ImageBase** contains the address (which should be of multiple of 64k for every non malicious PE file) of the memory where the linker creates an executable. Every non-malicious executable file contains non-zero value in **SizeOfInitializedData** field while in some malicious files this field contains zero value. **MajorImageVersion** contains higher values in non-malicious files as compared to malicious file which contains almost lesser or zero values. The **FileAlignment** field is also one of the fields of optional header which contains alignment of sections in bytes when they are loaded into the memory.

Section header comes right after the optional header in the PE file header. Values from all VirtualAddress fields from the component of type structure IMAGE_SECTION_HEADER are collected as V_1 to V_n (where n is the total number of sections in section header). In this model, NumberOfSections field of file header is used to calculate the required value as a feature. We calculated minimum, maximum and mean VirtualAddress as minVA, maxVA and meanVA respectively with V_1 to V_n values as shown in equation (1), (2) and (3).

$$minVA = \min(V_1, V_2, ...V_n) \tag{1}$$

$$maxVA = \max(V_1, V_2, ...V_n) \tag{2}$$

$$meanVA = \frac{\sum_{i=0}^{n} V_i}{NumberOfSections} \tag{3}$$

Similarly values R_1 to R_n are collected from the SizeOfRawData fields of section header. And then minimum, maximum and mean SizeOfRawData as minRD, maxRD and meanRD respectively are calculated by using R_1 to R_n values as shown in equation (4), (5) and (6).

$$minRD = \min(R_1, R_2, ...R_n) \tag{4}$$

$$maxRD = \max(R_1, R_2, ...R_n) \tag{5}$$

$$meanRD = \frac{\sum_{i=0}^{n} R_i}{NumberOfSections} \tag{6}$$

C. Splitting the Dataset

Some prominent and pertinent features are extracted from the collected features. The dataset of these prominent features is divided into two sets. We used the Cross-Validation method of machine learning for diving the dataset. Hold-Out method of Cross-Validation is applied on the dataset. By using hold-out method the dataset is divided into the training set and test set. Training set is bigger part which is 75% part of total dataset while test set is the smaller part which is 25% part of the dataset. Model is prepared on the training set and tested over testing set. Now, this trained network model is able to predict whether the new source/input executable file is malicious or not.

D. Executable file Classification

The different classification methods of supervised machine learning are used to find inferences and patterns on the data which already have class labels. Classification methods such as k-Nearest Neighbours, Decision Tree, Random Forest, Naive Bays and Support Vector Machine are applied on the collected prominent features. Here, classification methods are used to classify the data by training the model and applying the new data on the trained model for the prediction. The model learned to train itself to find some patterns and inferences for the prediction with the help of classifiers as a supervisor or learning method. After training the model it is then tested against testing data to find the performance accuracy of the learning method which was used to train the data. The different models are constructed by using these classifiers. The best model is constructed using random forest classifier.

Random forest is considered as best classifier because it consists of various decision trees which produce overall best classification result. The individual component of random forest that is, decision tree performs dataset splitting in tree-like structure by performing a feature test at its every node that optimizes some particular condition. The splitting algorithm used by random forest and decision tree classifier for splitting criteria is the Gini Index.

$$gini(D) = \sum_{j=1}^{n} P_j^2 \tag{7}$$

$$gini_A(D) = (|D_1|/|D|)gini(D_1) + (|D_2|/|D|)gini(D_2) \tag{8}$$

$$IR = gini(D) - gini_A(D) \tag{9}$$

Gini Index is measure of impurity which performs only binary split in the decision tree. It calculates Gini index for whole dataset (D) using Eq. 7 where P_j is the relative frequency of class j at any node of decision tree. For the attributes on which dataset is split in more than one subset it calculates Gini index using Eq. 8. The attribute with the highest value of **reduction in impurity (IR)** calculated using Eq. 9. is selected to be the node of a decision tree.

IV. Experimental Results

Experimental results and performance of the model is evaluated using the metrics such as true positive rate (TPR), false positive rate (FPR), Accuracy, Precision and Recall [7][12]. In a malware detection problem: TN refers to the number of benign executable files classified as benign, TP refers to the number of malicious executable files that are classified as malicious, FN refers the number of malicious executable files that are miss-classified as benign. FP refers the number of benign executable that are miss-classified as malicious. TPR, FPR, Accuracy, Precision and Recall is calculated as shown in equation (10), (11), (12), (13) and (14) respectively.

TPR (True Positive Rate) is defined as number of true positives divided by total number of malicious executable files.

$$TPR = \frac{TP}{TP + FN} \tag{10}$$

FPR (False Positive Rate) is defined as number of false positives divided by total number of benign executable files.

$$FPR = \frac{FP}{FP + TN} \tag{11}$$

Precision is defined as true positive numbers divided by the sum of true positive numbers and false positive numbers.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall is defined as true positive numbers divided by the sum of true positive numbers and false positive numbers.

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

Accuracy or classification rate is defined as the sum of the true positive numbers and true negative numbers divided by the total number of instances.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{14}$$

We achieved improved detection rates with combined feature set of file header, optional header and section header as compared to feature set of only optional header. Classification performance with optional header features is shown in Table 1. Random forest and decision tree classifiers have achieved the highest accuracy results among other classifiers. Random forest classifier has achieved an accuracy rate of 97.24% that is slightly higher than the 97.12% accuracy achieved by the decision tree classifier. With random forest classifier we got 97.5% precision and 97.9% recall, 97.50% true positive rate and 3.07%. false positive rate.

TABLE I
PERFORMANCE WITH OPTIONAL HEADER FEATURES

Classifier	Evaluation Metric				
	Accuracy	*Precision*	*Recall*	*TPR*	*FPR*
KNN	95.17%	0.9536	0.9500	95.00%	4.61%
GNB	75.17%	0.9493	0.5750	57.50%	3.07%
RF	97.24%	0.9750	0.9795	97.50%	3.07%
DT	97.12%	0.9740	0.9625	96.80%	4.00%
BNB	65.51%	0.6769	0.6125	76.12%	29.23%
ADB	94.48%	0.9500	0.9392	95.00%	6.15%
LR	94.48%	0.9679	0.9250	92.50%	3.07%
SVM	95.17%	0.9557	0.9500	95.00%	4.40%

To improve the accuracy results for more accurate malware detection, we have taken combined features from all the three headers. Classification performance with combined features is shown in Table 2. Here, random forest classifier has achieved the 98.63% accuracy rate which is highest accuracy rate among the different classifiers. As shown in Table 2, with random forest classifier we got 98.68% true positive rate and 1.42% false positive rate.

TABLE II
PERFORMANCE WITH COMBINED FEATURES

Classifier	Evaluation Metric				
	Accuracy	*Precision*	*Recall*	*TPR*	*FPR*
KNN	91.09%	0.9021	0.9203	92.10%	10.00%
GNB	77.39%	0.9761	0.5789	57.89%	1.42%
RF	98.63%	0.9858	0.9869	98.68%	1.42%
DT	96.57%	0.9712	0.9606	96.05%	2.85%
BNB	67.123%	0.7253	0.5658	56.57%	21.42%
ADB	97.260%	0.9584	0.9869	98.68%	4.28%
LR	78.76%	0.9021	0.6579	65.78%	7.14%
SVM	82.57%	0.7489	0.9800	95%	32.85%

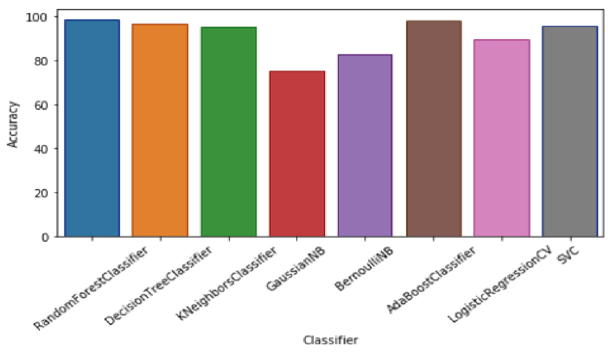

Fig. 3. Accuracy Comparison with combined features

V. COMPARATIVE ANALYSIS

Different performance results compared with two sets of features to achieve more accuracy in malware detection. We employed different classification methods on the dataset to find out the accuracy results. Classification algorithms such as Decision Tree (DT), Random Forest (RF), k-Nearest Neighbours (kNN), Gaussian Nave Bays Classifier (GNB), Bernoulli Naive Bays Classifier (BNB), AdaBoost Classifier (ADB), Logistic Regression (LR) and Support Vector machine (SVM) are applied on two datasets. One dataset is the combination of features from all the three main headers (file, optional and section header). Other dataset is the collection of features from optional header only. As shown in Fig.3, best accuracy result is achieved by applying random forest classifier with combined feature set. With optional header feature set, random

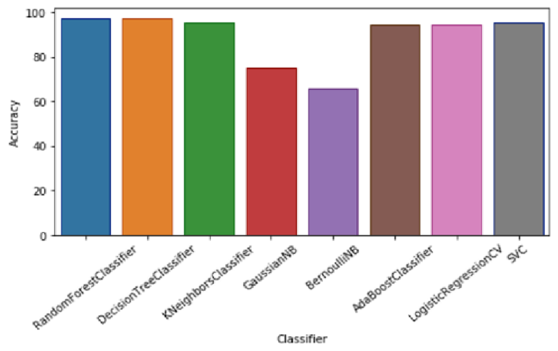

Fig. 4. Accuracy Comparison with optional header features

forest classifier has achieved the highest accuracy among different classifiers as shown in Fig.4. The precision and recall comparison with different classifiers is shown in Fig.5 and Fig.6 respectively.

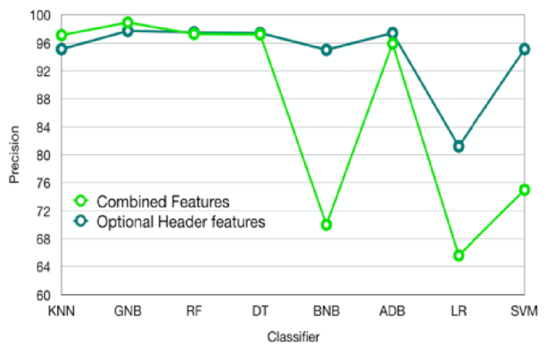

Fig. 5. Precision comparison

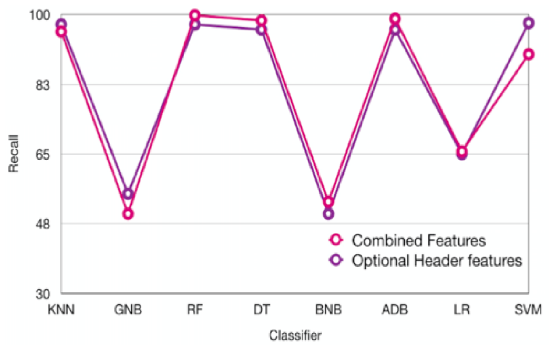

Fig. 6. Recall comparison

VI. Conclusion

In this paper, an efficient and fast approach for the malware detection is presented. The proposed model is based on static analysis approach. By applying machine-learning methods the model learns in which category the given file belongs is whether malicious or not. Features are extracted from file header, optional header and section header of various executable files. Extracted features from optional header are used as input to different classifiers to classify the malware and achieved the highest accuracy of 97.24% with random forest classifier. Model is also trained and tested on combined extracted features of file, optional as well as section header and achieved the highest accuracy of 98.63% with random forest classifier. It is concluded that the accuracy achieved by combined feature set is higher than the accuracy achieved by acquired features from optional header.

The proposed work is limited to malware detection in executable files of windows operating system. As future research, the aim is to extend our method to incorporate feature selection in more hybrid scenarios and design some techniques to extract those features. We can establish the model based on hybrid analysis that is the combination of both static as well as dynamic analysis by addressing the issues of both types of analysis.

References

[1] Mithal, Tulika, Kshitij Shah, and Dushyant Kumar Singh. ”Case studies on intelligent approaches for static malware analysis.”Emerging Research in Computing, Information, Communication and Applications. Springer, Singapore, pp. 555-567, 2016.

[2] Vatamanu, Cristina, et al. ”A comparative study of malware detection techniques using machine learning methods.” Int. J. Comput. Electr. Autom. Control Inf. Eng. pp. 555-567, 2016.

[3] Alsulami, Bander, et al. ”Lightweight behavioral malware detection for windows platforms.”2017 12th International Conference on Malicious and Unwanted Software (MALWARE). IEEE. pp. 75-81, 2017.

[4] Satrya, Gandeva B., Niken DW Cahyani, and Ritchie F. Andreta. ”The detection of 8 type malware botnet using hybrid malware analysis in executable file windows operating systems.”Proceedings of the 17th International Conference on Electronic Commerce 2015. ACM. pp. 5, 2015.

[5] Nissim, Nir, et al. ”Novel active learning methods for enhanced PC malware detection in windows OS.” Expert Systems with Applications. Vol. 41, No. 13, pp. 5843-5857, 2014.

[6] Rehman, Zahoor-Ur, et al. ”Machine learning-assisted signature and heuristic-based detection of malwares in Android devices.” Computers Electrical Engineering. Vol. 69, pp. 828-841, 2018.

[7] Feng, Pengbin, et al. ”A Novel Dynamic Android Malware Detection System With Ensemble Learning.” IEEE Access. Vol. 6, pp. 30996-31011, 2018.

[8] Huda, Shamsul, et al. ”A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection.” Future Generation Computer Systems. Vol. 83, pp. 193-207, 2018.

[9] Nguyen, Giang, et al. ”A heuristics approach to mine behavioural data logs in mobile malware detection system.” Data Knowledge Engineering. Vol. 115, pp. 129-151, 2018.

[10] Gu, Jingjing, et al. ”Consortium blockchain-based malware detection in mobile devices.” IEEE Access. Vol. 6, pp. 12118-12128, 2018.

[11] Karbab, ElMouatez Billah, et al. ”MalDozer: Automatic framework for android malware detection using deep learning.” Digital Investigation. Vol. 24, pp. S48-S59, 2018

[12] Bhatia, Taniya, and Rishabh Kaushal. ”Malware detection in android based on dynamic analysis.” 2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security). IEEE. pp. 1-6, 2017.

[13] Shah, Kshitij, and Dushyant Kumar Singh. ”A survey on data mining approaches for dynamic analysis of malwares.”2015 International Conference on Green Computing and Internet of Things (ICGCIoT). IEEE. pp. 495-499, 2015.

[14] Shijo, P. V., and A. Salim. ”Integrated static and dynamic analysis for malware detection.” Procedia Computer Science. Vol. 46, pp. 804-811, 2015.

[15] Yan, Ping, and Zheng Yan. ”A survey on dynamic mobile malware detection.” Software Quality Journal. Vol. 26, No. 3, pp. 891-919, 2018.

[16] Awan, Saba, and Nazar Abbas Saqib. ”Detection of malicious executables using static and dynamic features of portable executable (pe) file.” International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer. pp. 48-58, 2016.

[17] Shalaginov, Andrii, et al. ”Machine learning aided static malware analysis: A survey and tutorial.” Cyber Threat Intelligence. pp. 7-45, 2018.

[18] Kumar, Ajit, K. S. Kuppusamy, and G. Aghila. ”A learning model to detect maliciousness of portable executable using integrated feature set.” Journal of King Saud University-Computer and Information Sciences (2017).

[19] Liu, Liu, et al. ”Automatic malware classification and new malware detection using machine learning.” Frontiers of Information Technology Electronic Engineering. Vol. 18, No. 9, pp. 1336-1347, 2017.