2020 6th International Conference on Web Research (ICWR)

An Efficient Approach For Malware Detection Using PE Header Specifications

Tina Rezaei*, Ali Hamze
Dept. Computer Engineering and Information Technology Shiraz University, Shiraz, Iran
t.rezaei@shirazu.ac.ir, ali@cse.shirazu.ac.ir

***Abstract*— Following the dramatic growth of malware and the essential role of computer systems in our daily lives, the security of computer systems and the existence of malware detection systems become critical. In recent years, many machine learning methods have been used to learn the behavioral or structural patterns of malware. Because of their high generalization capability, they have achieved great success in detecting malware. In this paper, to identify malware programs, features extracted based on the header and PE file structure are used to train several machine learning models. The proposed method identifies malware programs with 95.59% accuracy using only nine features, the values of which have a significant difference between malware and benign files. Due to the high speed of the proposed model in feature extraction and the low number of extracted features, which lead to faster model training, the proposed method can be used in real-time malware detection systems.**

Keywords—Malware Detection, Portable Executable, PE Header, Machine Learning.

I. INTRODUCTION

Malware is software designed to perform unauthorized actions. Each year a large number of malware programs are developed and released. On the other hand, computer systems and the Internet play a key role in our daily lives, and every day a variety of computer applications are downloaded from the Internet on a large scale [1]. Therefore, the security of information and computer systems become increasingly principal for users. Many methods have been presented to tackle the malware detection challenge so far. The most common and oldest method is the signature-based method. A signature is a sequence of bytes that accurately identifies malware instances [2]. In this method, the extracted signature from the intended file is compared to a set of known malware signatures. If any match is found, the corresponding file will be identified as malware. This method can only detect malware samples that their signature is available. Therefore, they are regularly required to keep their signature set up-to-date. Moreover, malware developers use obfuscation techniques to alter their bit patterns, which will lead to different signatures. Therefore, they can be prevented from being detected. For example, some malware programs change their code in each infection, some encrypt themselves [3] or their malicious activity, and some others use packing techniques [4]. So this method fails to detect such malware and newly released malware. Due to the high speed of malware development and disadvantages of the signature-based method, machine learning-based techniques have been proposed to be a potential solution to this problem [5]–[7]. The most important feature of machine learning techniques is their generalization ability[8]. Machine learning methods can recognize patterns in the training data and learn from them, and then they can identify the data similar to the training data. Moreover, malware developers only used to change a small part of the code to produce new variants [9]. Thus machine learning-based methods are capable of identifying such malware. One of the most important parts of the machine learning-based system is the feature extraction phase. Two types of features used in the malware detection area are: 1. Static features, which are obtained from a file without executing it. 2. Dynamic features, which are obtained from executing a file (usually in an isolated environment). Dynamic features are indeed actual actions that are taken by the program at run-time [10]. Although dynamic features may be more effective than static features, they are more time and computational resource consuming to extract and can be easily avoided by applying a time delay [11]. In contrast, extracting static features is much faster and less costly. In addition, given the increase of malware development, presenting a real-time and fast malware detection system is now a critical and challenging task. Therefore, in this work, static features are used. In this method, based on the studies on Microsoft documentation of PE file structure, nine inductive features have been extracted from the header and the structure of the PE file. Afterward, three different machine learning algorithms, Random Forest, Support-Vector Machine (SVM), and k-Nearest Neighbors (kNN), are used to train the models. The proposed method has achieved high accuracy and high speed of detection by using only nine features.

The remainder of this paper is organized as follows. In section 2, related works on malware detection that utilize PE file header and machine learning algorithms, are discussed. In section 3, the proposed method is explained. Section 4 describes the dataset and obtained results. In section 5, discussion and more results are provided. Finally, the conclusion and future work are given in section 6.

II. RELATED WORK

Malware and benign programs are designed for different purposes, which results in differences in their structure. On the other hand, the PE header contains information about file structure. Therefore, the malware program can be distinguished from the benign program using information stored in the PE file header and structural features of the file. Almost every file with executable code that is loaded by windows is in the PE file format, and the other file formats do appear on rare occasions in malware. So the PE file format is the most common file format, which is used by malware. Lots of previous work has also used the information stored in the PE file header and its structure to detect malware. Some of them are discussed here.

In 2013, Khorsand et al. [12] presented a method based on the PPM compression algorithm for malware detection. They used PE file headers to build two compression models, one of them is constructed from malware programs, and the other one is constructed from benign programs. The class of

978-1-7281-1051-6/20/$31.00 ©2020 IEEE

the new sample is determined by the model has a higher compression ratio. Although increasing the PPM order can improve accuracy, the high memory consumption constraint increasing PPM order, which is the biggest drawback of this method.

In 2016, Belaoued et al. [11] proposed a real-time malware detection system based on analyzing information stored in PE-Optional Header fields. They used the combination of two feature selection algorithm, Chi-square and Phi coefficient, to select the most effective features. Finally, they used obtained features to train several machine learning classifiers. In their work, they only used the Optional Header part of the PE file, while using only this part is not enough to provide a generalized and efficient model.

In 2017, Vyas et al. [13] investigated static features to propose a real-time network malware detection system. Features are extracted from PE file header and sections and are re-grouped into four categories. File metadata, file packing, imported DLLs, and imported functions. These features are then used to train several machine learning classifiers.

In 2017, Raff et al. [14] used raw bytes of DOS Header, File Header, and Optional Header of the PE file to train a fully connected neural network for malware detection. They demonstrated the ability of the neural network in learning from raw bytes without explicit feature construction. In this work, each byte is converted to a 16-length feature vector using an embedding layer, which causes an increase in the number of network input and, consequently, network training time.

In 2018, Darshan et al. [15] trained several machine learning algorithms using a Hybrid feature set to detect malware. Static features are obtained from DOS Header, File Header, and Optional Header of the PE file. Dynamic features are obtained by extracting API Calls invoked by the Windows executable file during runtime. Then, the LSVC (Linear Support Vector Classification) is used to select the most significant static and dynamic features. Using dynamic features due to the need to execute the program, leads to high time and computational resource consumption.

The reviewed works demonstrated the efficiency of the PE file header in detecting malware by achieving good accuracy. Some of the previous works have used dynamic features to improve the performance of their proposed method, which increases the time and computational complexity. While providing a fast and real-time method for malware detection is a critical and challenging task these days. Most of the mentioned works, and many other works that have utilized PE header[16]–[18], specifically have used the Dos header, File header, and Optional header of the PE file. Although these three parts contain useful and efficient information for distinguishing malware and benign programs, many fields of these three parts are identical in malware and benign programs, and some of them like DOS header fields are historical and offer no information of particular. So all of the fields of these three sections are not useful for malware detection and only increase the number of features, which in turn reduces the efficiency and speed of the model. Therefore, in the proposed method, by extracting only nine features from the PE file header and its structure, the pre-processing time is minimized. Also, it is attempted to identify malware programs with high speed and high accuracy, using the fewest possible features.

III. Proposed Method

The proposed method relies on the structural differences between Malware and Benign programs. These structural differences can be used to extract features that are highly capable of detecting malware. In the proposed method, the PE file header and its structure are examined through Microsoft document, and accordingly, a set of features is extracted from each file. Extracted features are generally different in malware and benign programs. The raw values of the header fields are not considered in this work because the distribution of these fields in a particular dataset may vary for each family, while it may be similar in other datasets or in the real world. Therefore, the raw value of the header fields is not suitable to be used in the real world. So, in the proposed method, by studying the PE file structure, nine efficient features are extracted that are more effective than raw header fields. Before explaining the extracted features, the PE file format is examined.

A. PE File Format

Portable Executable is a file format that is used by windows executables, object code, and DLLs. In Fig 1, an overview of the PE file is given. The PE file format contains a header followed by a series of sections. The header contains metadata about the file itself. The first part of the PE header is the DOS header, which checks whether the file is a valid executable file. The next part is the PE file Signature, which always has the value "PE \ 0 \ 0", and specifies the file as a PE file. Then, the File header and Optional header are placed, which contain useful information like compile-time, the number of sections of the program, size of the code section, size of the file when loaded to the memory. The last field of the optional header is an array of Data_Directory structures. Each row of this array indicates where to find other important components of executable information in the file. Indeed, each file can have some tables such as Import table (table of imported functions), Export table (table of exported functions), ResourcesTable (table of resources such as images embedded in the PE). Each row of Data_Directory gives the address and size of a specific table. After the optional header, is the section table as the last header section, which contains information about the program sections. Each row specifies a section of the PE file. Each section header provides information such as the name of the section, size of

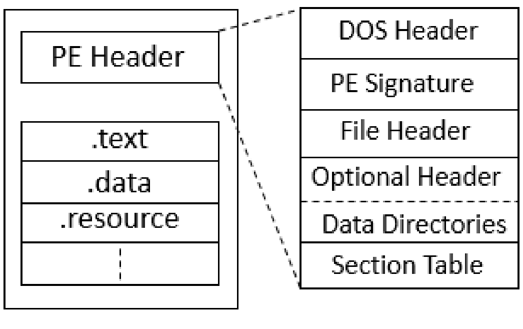


Fig. 1. An Overview of PE File Format

the section, and address of section. After the header, there are actual parts of the file, including code, global data, resources such as photos, icons, and so on.

B. Feature Extraction

In this section, extracted features based on the studies of PE header and structure of the PE file are explained. After describing each feature, the distribution of each feature for malware and benign programs is shown in Fig. 2. The x-axis of each of these diagrams represents the corresponding feature values. The y-axis represents the number of samples that have a certain value of x.

Section Name: Information about each section, including its name, is in the corresponding row in the section table. Benign programs almost always have a standard name for each section, while most malware programs have a random name for some sections. Even some malware programs include an unnamed section. Most of the malware programs use packing tools or nonstandard tools that cause PE files to have nonstandard section name. Packers often use predefined section names and sometimes use empty section names. For example, UPX packer always rename .txt and .data sections to UPX0 and UPX1. So, the section name is a useful feature for distinguishing malware and benign programs. For this purpose, a list of standard names according to the Microsoft documentation is provided, Table. I. The section names of each program are then extracted and compared with this list. Based on this comparison, three features are obtained:

- **NumberOfNonStandardSections:** It is the number of sections of the program whose names do not match the standard list. As is shown in Fig 2.a, the number of nonstandard sections in malware programs is higher than benign programs. For most benign programs, the maximum value of this feature is two, while for most malware programs, this feature is greater than 2.
- **NumberOfStandardSections:** It is the number of sections of the program whose names match the standard list. As is shown in Fig 2.b, the number of standard sections in benign programs is higher than in malware programs. At higher values of this feature, the number of benign programs is significantly larger, and a large number of malware programs have at most two standard sections.
- **EmptySectionName:** It is a Boolean feature. Its value is set to 1 if the program has a section without name otherwise 0, Fig 2.c. As can be seen, none of the benign programs have an unnamed section, and samples with unnamed section are only malware programs.

Data Directories: As mentioned, each file contains a number of data directories, each corresponding to a table, and each table provides different information. The PE file format defines 16 possible data directories, but the number of data directories in each file is not the same. For example, the debug directory contains debug-information that is often used to trace the execution of an executable. So malware writers usually take care to remove their programs of debug-information to avoid possible reverse engineering and detection. Resource directory contains the resources, such as dialog boxes, menus, icons, and so on. Malware developers usually remove resources, debug-information, or other parts of data directories to reduce the size of the compiled executable. In this research, six important data directories are selected: Export Table, Debug Table, Load Config Table, Resource Table, Base Relocation Table, and TLS Table. The presence or absence of each of them is then checked for each file, and accordingly, a feature is extracted:

TABLE I. LIST OF STANDARD SECTIONS NAME

Standard section names								
.text	.data	.data	.rsrc	.idata	.bss	.code	.edata	.rdata

- **DataDirectories**: A Boolean vector with a length of the number of selected data directories (6) is considered. The elements of this vector are set to 0 or 1 based on the absence or presence of the corresponding data directory. Therefore, a 6-bit binary number is created, which would be considered as a feature for each file after being converted to a decimal number, Fig 2.d. Since the maximum value of this feature in most malware programs is 10, it can be concluded that malware programs have fewer data directories than benign programs and also lack data directories that have a higher value in the Boolean vector.

Entropy: Lots of malware programs utilize packing techniques. Packing is one of the obfuscation techniques used by malware writers to make analysis harder, prevent reverse engineering and hider malware detection. Packaging leads to an increase in entropy. So entropy can be used as an indicator of malware programs. Also, some benign programs use packing. Their goal, however, is to prevent reverse engineering in order to protect copyright or to reduce file size in order to accelerate download time and maintain user bandwidth. In general, a higher percentage of malware programs use packing. So the file entropy and the entropy of the two important sections, .text and .data, which are present in almost every file, are calculated. Accordingly, the following three features are extracted for each file:

- **FileEntropy:** It is the calculated entropy for the whole file. As Fig 2.e shows, at higher entropy values, the number of malware is always greater than the benign files. So it can be well understood that most malware programs use packing.
- **TextSectionEntropy:** It is the calculated entropy for the *.text* section. If an intended file does not have this section, the value of this feature is set to -1. As can be seen in Fig 2.f, the large number of malware programs do not have this section, and only malware programs have a value between 7 to 8 for this feature, which determines a high entropy.
- **DataSectionEntropy:** It is the calculated entropy for the *.data* section. If an intended file does not have this section, the value of this feature is set to -1. As can be seen in Fig 2.g, a significant number of malware programs do not have this section, and at high entropy value (between 7 to 8) number of malware programs is higher than benign programs.

Checksum: Each file has a checksum field in the optional header that is used to validate the executable file at load time and can be used as an integrity check. According to the studies, it has been observed that the value of this field is zero for many malware programs, while benign programs have significantly larger values for this field. Therefore, a feature is extracted:

- **Checksum:** It is a Boolean feature, which is set to 1 if the checksum field is zero otherwise is set to 0. Fig 2.h shows that the value of this feature for lots of malware programs is zero, while many benign programs have a nonzero value for this feature.

FileInfo: Each PE file can contain a set of string fields such as *ProductName*, *ProductVersion*, *FileVersion*, *FileDescription*, and *CompanyName* stored in the PE file *FileInfo*. Table. II, lists all the string fields that can be stored in this set. Each file has a different number of these fields. Since benign programs are valid programs produced by reliable individuals or companies, they usually contain these fields, while many malware programs lack this set or have only a small number of these fields. Therefore, for each file, based on the fields stored in FileInfo, a feature is obtained:

- **NumberOfFileInfoFields:** It is the number of string fields stored in the FileInfo field. As can be seen in Fig 2.i, this feature is set to zero for lots of malware programs, indicating that many malware programs lack this information, while most of the benign files contain a lot of this information.

C. Learning Algorithm

In the last step of the proposed method, the extracted features are used to train several machine learning models to detect malware. Specifically, they are Random Forest, SVM, and KNN. In building the SVM model, the linear kernel is deployed. To classify samples using KNN, only K = 1,3,5 are used that are the most commonly used values for k.

IV. Dataset And Experinments

A. Dataset

The evaluation of the proposed method is performed on a dataset of 2460 PE file, containing 1230 malware samples and 1230 benign samples. Indeed, from all malware samples in this dataset, 1230 malware samples are randomly selected. Benign samples are also collected from the "Program Files" and "System32" folders of a standard Windows XP machine. All benign samples were scanned with ESET NODE32 antivirus to ensure there is no malicious file among them.

B. Experimental Setup and Evaluation Metrics

All the experiments are conducted in an environment with the following specifications: Intel(R) Core(TM) i3-2350M CPU @2.30GHz8 with 8GB of RAM and Debian 10 as the operating system.

In order to evaluate the proposed method, K-fold Cross-Validation is used. In this method, the investigated data is divided into k equal parts. Then one part is used as test data, and the other k-1 parts are used as training data. This operation is repeated for k rounds, and the final result is obtained by averaging results obtained from K rounds. For all the experiments performed here, the value of 10 is considered for k. The performance of the proposed method is evaluated using four very well-known measures in the machine learning context, including accuracy, precision, recall, and f measure.

C. Results and Comparison with Other Methods

The obtained results of the proposed method on different measures with the evaluation of 10-fold cross-validation are given in Table III. All three models are able to detect malware with good accuracy. Random forest performed better than the other two models in all cases, and it gained an accuracy of 95.5%. Random Forest employs ensembling technique. Actually, it is a collection of decision trees and the majority vote of the forest is selected as the predicted output. So the trees protect each other from their individual errors and give a more generalized solution.

Because of using the PE header in this research, Farrokhmanesh's [19] approach and Kumar's [18] approach have been selected for comparison. Also, they both used machine learning methods to classify malware and benign programs. In [19], by converting raw bytes of the header of each file to an audio file and extracting musical patterns as features, several machine learning algorithms are trained to classify malware and benign programs. In [18], they used a combination of PE header fields raw value and derived values for the malware detection task. Derived values are obtained by an analysis of the standard values of the PE header fields. The comparison results are presented in Table IV. As can be seen, the proposed method performs better than Farrokhmanesh's approach. In addition, producing audio files and extracting musical patterns is a time-consuming process. The proposed method also performs better than Kumar's method. They also used the header fields' raw value besides derived values. While in the proposed method, raw values of header fields are not considered because they can not be a proper feature in the real world. They used 68 features, which is more than the number of features used in this research. So, the proposed method by using far fewer features have achieved higher accuracy than Kumar's method.

V. DISCUSSION AND MORE RESULT

In this research, it has been attempted to detect malware using features extracted from the PE header and its structure.

TABLE II. LIST OF THE STRING FIELDS IN THE FILEINFO FIELD

String Fields of the FileInfo field		
SpecialBuild	PrivateBuild	Comments
LegalTrademarks	LegalCopyright	OLESelfRegister
OriginalFilename	FileDescription	FileVersion
ProductName	ProductVersion	CompanyName
InternalName		

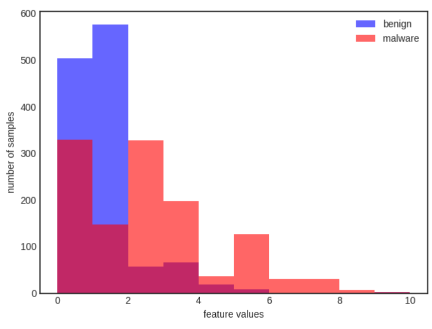
TABLE III. RESULTS OF THE PROPOSED METHOD

classifier	***Accuracy***	***Precision***	***Recall***	***F-measure***
Random Forest	**95.5**	**95.7**	95.4	**95.5**
SVM	94.4	93.5	**95.6**	94.5
KNN(K = 1)	94.0	94.6	93.5	94.0
KNN(K = 3)	93.7	94.5	92.8	93.6
KNN(K = 5)	93.7	94.6	92.8	93.6

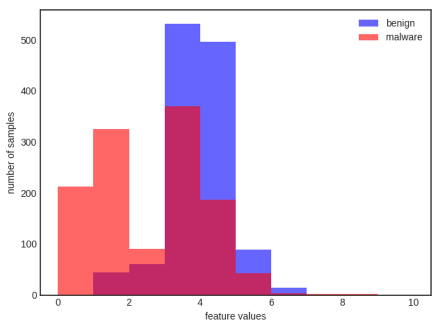
The proposed method demonstrated the high capability of the header in the malware detection task, and by extracting only nine features, it was able to classify malware and benign programs with an accuracy of 95.59%. These features are extracted based on the study of the structural specification of the PE file. Most of these features are extracted from the PE header. On the other hand, as mentioned in Section 2, lots of previous PE header-based works, specifically used the Dos Header, File Header, and Optional Header parts of the PE file. In order to compare the performance of the features extracted in this research against the raw fields of the header, all the DOS Header, file header, and Optional header fields are extracted. These 3 parts contain 55 fields, but e_res and e_res2 fields of the DOS header are removed because they are reserved. All the extracted fields are then given to Random Forest, SVM, and KNN. However, the features extracted in this research are far fewer than those extracted from these three parts. Therefore, in another experiment to make a fairer comparison, where the number of features is equal, the chi-square feature selection algorithm was used to select the best 9 features from these 3 parts. The selected features are then given to the mentioned classifiers. The results of these experiments are presented in Table V. Since Random Forest had the best results for each experiment, only Random Forest results are shown in Table V. As can be seen, the proposed method performed better than the other two experiments. The proposed method has higher accuracy than the case where all header fields are used, in addition to using a much smaller number of features. Therefore, the superiority of the extracted features over raw header fields is well demonstrated. It should be noted that the results of the best 9 selected features are only 1.5% lower than all the header fields. It indicates that all of the header fields are not effective, and by selecting a smaller number of header fields a good accuracy can be gained.

VI. CONCLUSION

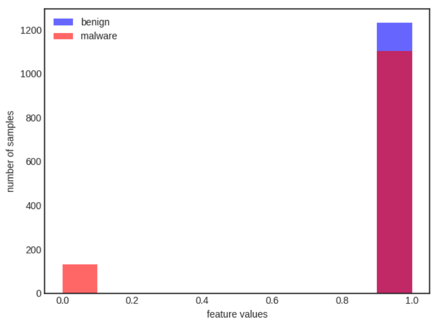
In this paper, a malware detection system was developed based on the analysis of the PE file header and its structure. Due to the differences in the intent of the malware and benign files and different way of building them, there are several differences in the structure of the PE file. Therefore, by analyzing the PE file structure, nine features were extracted from the header and the structure of each file, which was significantly different for malware and benign programs . Extracted features are the features that are



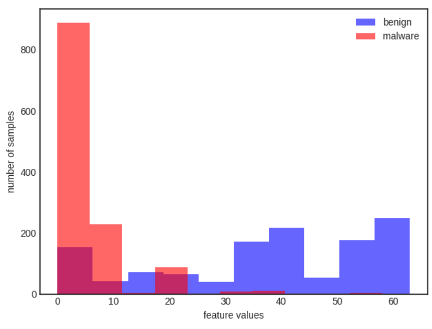
(a) NumberOfNonStandardSections



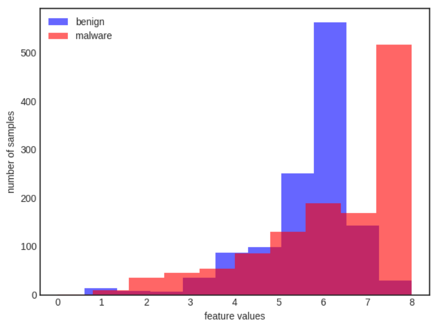
(b) NumberOfStandardSections



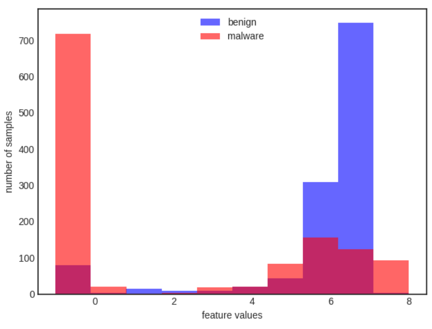
(c) EmptySectionName



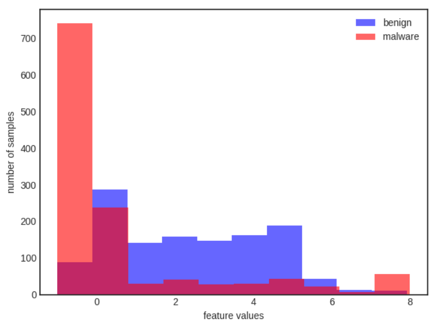
(d) DataDirectories



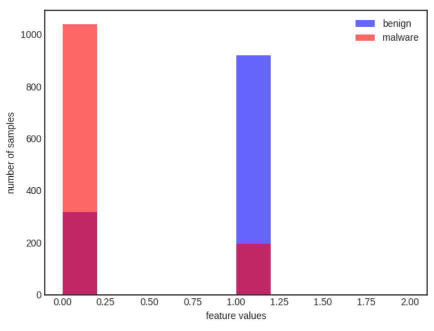
(e) FileEntropy



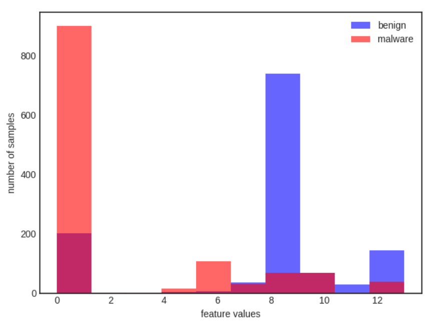
(f) textSectionEntropy



(g) DataSectionEntropy



(h) Checksum



(i) NumberOfFileInfoFields

Fig.2. Distribution of the extracted features values for benign and malware programs (continued)

TABLE IV. COMPARING THE PROPOSED METHOD WITH OTHER METHODS

classifier	*Accuracy*	*Precision*	*Recall*	*F-measure*
Farrokhmanesh's [18]	92.2	91.5	93.1	92.2
Kumar's [19]	94.9	95.5	94.4	94.9
Proposed method	**95.5**	**95.7**	**95.4**	**95.5**

TABLE V. COMPARING THE FEATURES EXTRACTED IN THE PROPOSED METHOD WITH HEADER RAW FIELDS AND THE 9 BEST SELECTED

classifier	*Accuracy*	*Precision*	*Recall*	*F-measure*
Proposed method	**95.5**	95.7	**95.4**	**95.5**
Header Raw Fields	94.9	**95.8**	94.0	94.9
The best 9 selected fields	93.4	94.2	92.5	93.3

generally different in malware and benign programs, unlike the raw header fields whose distribution in each data set can be different. The proposed method, with only nine features, was able to detect malware with high accuracy and speed. Further work can be done on the other specifications of the structure of the PE file and therefore extract more features.

REFRENCES

[1] K. Sethi, S. K. Chaudhary, B. K. Tripathy, and P. Bera, "A Novel Malware Analysis Framework for Malware Detection and Classification using Machine Learning Approach," in *Proceedings of the 19th International Conference on Distributed Computing and Networking - ICDCN '18*, 2018, pp. 1–4.

[2] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 12, no. 2, pp. 59–67, 2016.

[3] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*, 2013, pp. 113–120.

[4] W. Fleshman, E. Raff, R. Zak, M. McLean, and C. Nicholas, "Static Malware Detection & Subterfuge: Quantifying the Robustness of Machine Learning and Current Anti-Virus," *MALWARE 2018 - Proc. 2018 13th Int. Conf. Malicious Unwanted Softw.*, pp. 3–12, 2019.

[5] D. Carlin, P. O'Kane, and S. Sezer, "A cost analysis of machine learning using dynamic runtime opcodes for malware detection," *Comput. Secur.*, vol. 85, pp. 138–155, 2019.

[6] Z. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," *Proc. 2017 Des. Autom. Test Eur. DATE 2017*, pp. 169–174, 2017.

[7] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Comput. Inf. Sci.*, vol. 8, no. 1, 2018.

[8] Z. Kan, H. Wang, G. Xu, Y. Guo, and X. Chen, "Towards Light-Weight Deep Learning Based Malware Detection," *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 1, pp. 600–609, 2018.

[9] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *J. Comput. Virol. Hacking Tech.*, Aug. 2018.

[10] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malware detection and classification using system-call groups," *J. Comput. Virol. Hacking Tech.*, vol. 13, no. 1, pp. 29–46, Feb. 2017.

[11] M. Belaoued and S. Mazouzi, "A chi-square-based decision for real-time malware detection using PE-file features," *J. Inf. Process. Syst.*, vol. 12, no. 4, pp. 644–660, 2016.

[12] Z. Khorsand and A. Hamzeh, "A novel compression-based approach for malware detection using PE header," *IKT 2013 - 2013 5th Conf. Inf. Knowl. Technol.*, pp. 127–133, 2013.

[13] R. Vyas, X. Luo, N. McFarland, and C. Justice, "Investigation of malicious portable executable file detection on the network using supervised learning techniques," *Proc. IM 2017 - 2017 IFIP/IEEE Int. Symp. Integr. Netw. Serv. Manag.*, pp. 941–946, 2017.

[14] E. Raff, J. Sylvester, and C. Nicholas, "Learning the PE Header, Malware Detection with Minimal Domain Knowledge," *Proc. 10th ACM Work. Artif. Intell. Secur. - AISec '17*, pp. 121–132, 2017.

[15] S. L. Shiva Darshan and C. D. Jaidhar, "Windows malware detection system based on LSVC recommended hybrid features," *J. Comput. Virol. Hacking Tech.*, 2018.

[16] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "PE-miner: Mining structural information to detect malicious executables in realtime," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5758 LNCS, no. January, pp. 121–141, 2009.

[17] J. Bai, J. Wang, and G. Zou, "A malware detection scheme based on mining format information," *Sci. World J.*, vol. 2014, 2014.

[18] A. Kumar, K. S. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *J. King Saud Univ. - Comput. Inf. Sci.*, 2016.

[19] M. Farrokhmanesh and A. Hamzeh, "A novel method for malware detection using audio signal processing techniques," *2016 Artif. Intell. Robot. IRANOPEN 2016*, pp. 85–91, 2016.