Effective Ransomware Detection Method Using PE Header and YARA Rules

S. Hashwanth and S. Kirthica

1 Introduction

Nowadays, information security has become crucial [1]. The number of cyber-crimes has significantly increased in recent years [2]. Ransomware is one of the most serious risks (or) dangers to computer systems and data, and its prevalence has grown with the rise of digital currencies [3]. An illustration of a malicious program which encodes (or) encrypts data, prevents users from accessing the computer or their information (or) data, and then requests payment for the ransom is called ransomware. To prohibit a person or company from accessing files on a computer, ransomware is created [4]. Ransomware attacks are mainly focused on big tech giants where the data/information is more crucial aspect. These companies are more prone to pay a ransom because they are more likely to need immediate access to files. Without any predetermined targets, random attacks are launched over the Internet to infect as many systems as possible.

Recent ransomware attacks have had a significant negative impact on several businesses, crippled local services, and affected hospital's ability to provide vital services. In May 2021, Colonial Pipeline, a major US fuel pipeline operator, was hit by a ransomware attack that forced the company to shut down its operations. The attacker stole nearly 100 gigabytes of data from the company's server and demanded ransom in return. In this paper, ransomware is being detected with the help of the respective portable executable (PE) headers. In order to classify the application, initially, feature selection is done over the dataset. Feature selection is used to select

S. Hashwanth
Vellore Institute of Technology, Chennai, India

S. Kirthica (✉)
Department of Computer Science and Engineering, College of Engineering Guindy, Anna University, Chennai, India
e-mail: s.kirthica@gmail.com

J. K. Mandal et al. (eds.), *Proceedings of International Conference on Network Security and Blockchain Technology*, Lecture Notes in Networks and Systems 738,
https://doi.org/10.1007/978-981-99-4433-0_16

essential data and reduce the data ultimately leading to lesser computation [5]. Feature selection is nothing but selecting the attribute that is more than enough to compute the model and give more accurate results. Once the features are selected, the dataset is split into train and test data. Once the data is split, various machine learning models are trained over training data. Once the models are trained, it can be used to classify the legitimate and malware files. During live testing of the models, there are two ways for this feature extraction from the application [6]. One of the methods is through dynamic analysis. To extract the features, the dynamic approach needs to run the application. This method extracts features in accordance with the operations of the program as it runs. Many ransomware applications do not exhibit their true nature for a while after being launched, so they avoid detection and easily enters the system. During execution, how long the program should be watched over with this approach is unknown [6, 7]. Also, dynamic analysis cannot be done when the file is not executed. Thus, static analysis can be used to perform in this situation. Execution of the application is not required for static analysis. Recent studies have shown that static feature-based techniques can accurately identify ransomware [11].

In this paper, static analysis is used to get more accurate and precise results. The core objective of this proposed work is to detect the ransomware more efficiently and easily without any complex distributed modules. The proposed work has produced great accuracy and speed. Bitcoin is preferred by the attackers as a payment method since it is an anonymous payment system that hides their identity and location. So, along with the PE header classification, YARA rules are implemented to find whether a Bitcoin address is available in the file. This helps to significantly boost the model accuracy. Here, the final prediction result is provided to the user with the malware status of that application.

2 Related Works

Manavi et al. [8] discussed a method where an image based upon PE header is constructed, and CNN is used to extract and classify the features from those images. They performed static analysis to extract feature. Static analysis helps to extract the features without the program being executed. Their proposed method is complex, and the file is not scanned for crypto signatures.

Vinayakumar et al. [9] proposed a method using shallow and deep networks. They use the prevalence of API requests to distinguish ransomware from other malware families and benign malware. Simple MLP networks are utilized to save high computational costs; however, they perform less well than more complicated design choices when it comes to producing outcomes.

Homayoun et al. [10] discussed a dynamic strategy for ransomware imprisonment. They kept a collection of logs and used the collections to extract features. Finally, they classified the gathered features using CNN and LSTM. They demonstrated the existence of unique recurring patterns within various ransomware families. Since this

approach is dynamic, during execution, how long the program should be watched over with this approach is unclear.

Vidyarthi et al. [11] suggested a technique for ransomware detection based on executable file's PE header. They retrieved features for each application by using a few headers field values from the executable files. They performed static analysis and unpacked packed applications. This approach was not efficient since they used lesser number of attributes for determining the final prediction result. Also, crypto signatures could have been detected for more precise results.

Hanqi et al. [12] proposed a method which includes transforming opcode sequences from ransomware samples to N-gram sequences. Then calculation of TF-IDF is performed for each N-gram to select feature N-grams. To extract features without program being executed, static analysis is performed. It is difficult to extract opcodes from files if the files are packed.

Bahrani et al. [13] proposed an approach that employs process mining to discover ransomware by first extracting the process model from the events logs, following which features are retrieved and combined with classification algorithms. The features used to classify are not sufficient, thus making it less efficient.

El-Kosairy et al. [14] proposed a method where using a deception system based on honey files and honeytokens, any intrusion or ransomware attempting to compromise private files is detected. This approach is easy to deploy but this only works with NTFS file systems and cannot work with the FAT32 file systems.

Rezaei et al. [15] proposed a method where different machine learning models are trained using information derived from the PE file structure and header to identify malware applications. They used static analysis, which is an advantage to extract features even when program is not executed. The accuracy of their proposed work is lesser and could have used more attributes to classify.

Manavi et al. [16] used LSTM network to process the executable file header and build a detection model. The approach performs well by consuming less time for delivering the results. Only con is that it is complex and gives less accuracy.

Belaoued et al. [17] suggested a technique for detecting malware in real-time that uses information stored in PE-optional header fields in 2016. To choose the most useful features, they combined the Chi-square and Phi coefficient feature selection algorithms. Finally, they trained several machine learning classifiers using the features they had obtained. This model only utilized the optional header section of the PE file for their research; however, this is insufficient to offer a thorough and useful model.

Vyas et al. [18] suggested a real-time system for detecting the network malware by investigated static features. The header and sections of PE files are mined for features, which are then reorganized into 4 categories. The 4 categories are file packing, DLL imports, imported functions, and file metadata. This proposed work uses 28 features which can be minimized in to give more accurate results or lesser computations.

By observing the previous works, it can be noticed that most of the previous proposed works undergo dynamic approach toward feature selection which is not efficient since the file needs to be executed for that [11]. In few cases, opcodes were taken for the detection of ransomware. In these works, it is difficult to get the opcodes

if the file is packed [13]. Most research papers which have provided efficient results lack with respect to complexity of its architecture [9–11].

3 Proposed Method

The proposed method is divided into 3 modules—data pre-processing, model training, and static analysis. Further, YARA rules will be implemented to check for bitcoin address in the file. The dataset consists of the metadata from the PE file.

3.1 PE Structure

The portable executable file format is used by Windows executables, object code, and DLLs [15]. A header is followed by a number of sections in the PE file structure. The header contains details about the file itself. The next element is the PE file signature, which always contains the value "PE 0 0". The file header and optional header are then included. These headers provide crucial information. In Fig. 1, insight over PE file structure is shown.

The locations of additional significant executable file information are indicated in each row of this array. In fact, each file may contain a few tables, such as the resources table and import table. The size and address of a particular table are provided in each row of the Data Directory. The section table, which includes details on the program sections, comes as the final header section after the optional header. Each row describes a specific area of the PE file.

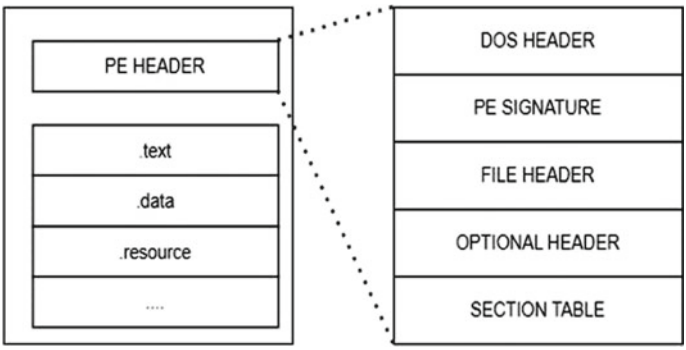

Fig. 1 PE file structure

3.2 Data Pre-processing

The dataset is imported with the help of panda's library. The dataset is visualized to get detailed insights over the data. Once the data is loaded, the first step is feature selection which is carried out to select essential features required for the computation [19]. Less resources are needed to carry out computations or actions when characteristics are minimized. Thus, computer can complete more work with less storage space and computation time.

In this paper, impurity-based feature importance is calculated using tree-based estimators, and this is then used to eliminate unnecessary features. After eliminating the unnecessary feature, the remaining features are used to get the most efficient and accurate results after classification. After feature selection, the attributes count is reduced from the 57 features to 14 features. These 14 features can be used for further model training to get more precise results.

3.3 Model Training

The final data from the previous step is used for training the model. The data is fed into different classification models to get the classification results. For classification, the data is split into train and test set. Once the data is divided into train and test set, the train set is then fed into each model to get the classification score of the record being a legitimate file or malware. Models included in the research are decision trees, gradient boosting, random forest, adaptive boosting, and Naïve Bayes, logistic regression, and KNN. Few of them are commonly used ones [20].

3.4 Static Analysis

Once the model is training, it can be tested against live applications. The applications to be classified are taken, and these applications undergo static analysis in order to get their respective PE headers. Static analysis, also known as static code analysis, examines a computer program to identify issues without running the application [21]. In order to understand the structure of the code and subsequently detect errors in it, static analysis is often conducted on the source code of the program using tools that turn the program into an abstract syntax tree (AST).

In this proposed work, PE headers are extracted from the file based on which the model predicts it as legitimate or malware. Windows 25 executables, and DLLs and object code all employ the portable executable (PE) file format. The Windows OS loader needs information from the PE file format, which is a data structure, to manage the wrapped executable code. PE headers are extracted using a Python package "pefile". A multi-platform Python module called "pefile" is used to parse and

operate with portable executable (PE) files. Most of the data in the PE file headers, as well as all the metadata and data in the sections, are all accessible. Once the required features are taken out from the source application, the data is passed into the trained model so that the final classification output can be obtained whether the file is legitimate file or a malware.

3.5 YARA Rules Implementation

After successfully classifying the applications, further YARA rules are used to identify whether a file contains any Bitcoin address. Rules are written in order to check for any Bitcoin address presence in that file. A Bitcoin address is composed of 26–35 alphanumeric characters that start with 1 or 3.

4 Results and Discussion

4.1 Dataset

The dataset used in this project is taken from Kaggle [23]. The data in the dataset is obtained by conducting statistical analysis on 138,047 application files. The objective is to classify files as malware or legitimate. Legitimate files are software that does not behave like malware and are useful and harmless to the users. The dataset constitutes 41,323 binaries (exe, dll) which are legitimate and 96,724 malware files from virusshare.com. The statistical analysis extracted PE information and calculated the entropy of different sections of these files. Finally, the dataset consisted of 138,047 entries with 57 columns.

4.2 Evaluation Metrics

The proposed work is evaluated using the metrics: accuracy, precision, *F*-measure, and recall [22].

In order to calculate these metrics, the confusion matrix must be visualized as shown in Fig. 2, and the true (or) false positive and true (or) false negative values should be taken from the matrix. These values are used for the calculation of the metrics. In these metrics, true positive (TP) refers to the number of samples that are classified rightly as ransomware, true negative (TN) refers to the number of samples that are classified rightly as the legitimate files, false positive (FP) refers to the number of samples that are falsely classified as ransomware, and false negative (FN) refers to the number of samples that are falsely classified as legitimate files.

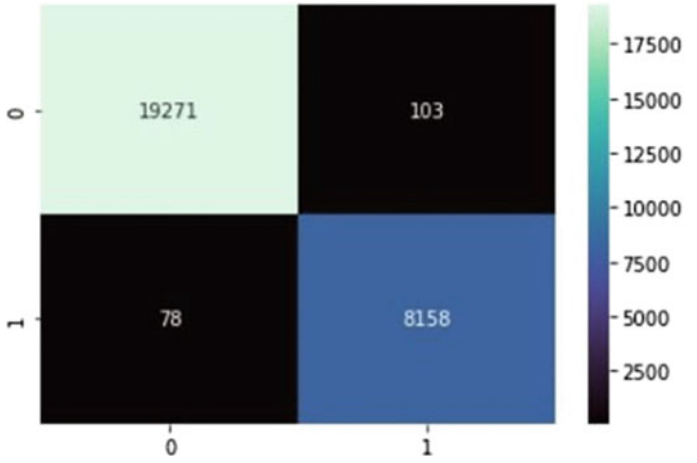

Fig. 2 Confusion matrix obtained using random forest model

4.3 Final Evaluation

After performing evaluation with the models, random forest turned out to give the highest prediction accuracy and score out of all the other models. Random forest gave a whopping accuracy of 99.3% in classification. The final score of the model is 99.34%. Compared to the previous works, the classification accuracy is the highest among them all as shown in Table 1. All the other evaluation metrics (recall, precision, and $F1$-score) are cross verified with the related works and visualized as shown in Fig. 3 and Table 2.

From this result, random forest is said to perform the best and give more accurate and precise results. Further, YARA rules are implemented to find whether the file contains any Bitcoin address or not. This helps to get an insight over the file to check for any Bitcoin address which will help to confirm with the ransomware.

Table 1 Final model classification scores

Model	Score
Logistic regression	70.17
KNN	98.97
Decision tree	99.05
Random forest	99.34
Naïve Bayes	70.17
Gradient boosting	98.88
AdaBoost	98.56

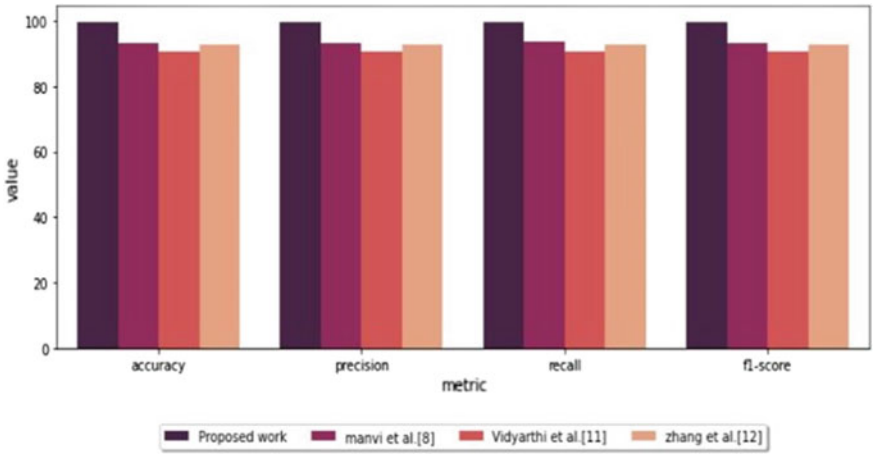

Fig. 3 Comparison of evaluation metrics of proposed work with the previous works

Table 2 Comparison of evaluation metrics of proposed work with related works

Metric Proposed work	Accuracy	Precision	Recall	*F*-score
Proposed method	99.34	99.59	99.46	99.53
Manvi et al. [8]	93.33	93.33	93.40	93.34
Vidyarthi et al. [11]	90.43	90.43	90.67	90.40
Zhang et al. [12]	92.75	92.75	92.78	92.74

4.4 Testing

The test bed is taken from dike dataset available in GitHub. This dataset is a labeled one which contains the benign PE and OLE files. The test bed consists of 750 benign files. And, also the model is tested against different setup files of many programs. Once the files are taken, it underwent static analysis in order to get those 14 features selected using feature selection during data pre-processing. After getting the required metadata, the data is tested with the loaded model (random forest). Then, the desired result is obtained. Further, YARA is compiled by importing YARA-Python, then the rule to find the bitcoin address is defined manually. The classified file is further process with this rule in order to check for presence of any Bitcoin address in that file. If presence of bitcoin is found, 1 is returned or else 0 is returned.

5 Conclusion

In this study, a ransomware classification system was created by analyzing structure and header of a PE file using static analysis. Static analysis helped us to extract the required data without the execution of the program. Random forests can handle large datasets with high dimensionality and can avoid overfitting by building multiple decision trees and aggregating their results. Random forest gave the highest accuracy. This model is stored and loaded during static analysis for classification of the application. Further, set of rules (YARA rules) were defined in order to check whether the file consists of any bitcoin address or not. The advantage of the proposed work over the previous studies is that the result gave a whopping accuracy of 99.34% which shows the efficiency of the proposed work. With this proposed work, the future impacts of ransomwares can be mitigated in large scale. Additional implementation of YARA rules in checking for Bitcoin address to get furthermore insights over the file. Future works are discussed in the below section.

The future work includes improvising the data to give a slightly higher performance. Further, model can be trained with more benign files in order to increase the accuracy of the project. Also, Bitcoin detection is not just enough to predict ransomware applications. In that case, the future work also comprises the implementation to find the crypto signatures in the executable file to provide a more reliable result.

References

1. Alkhudhayr F, Alfarraj S, Aljameeli B, Elkhdiri S (2019) Information security: a review of information security issues and techniques. In: 2019 2nd international conference on computer applications & information security (ICCAIS), pp 1–6. https://doi.org/10.1109/CAIS.2019.8769504
2. Humayun M, Niazi M, Jhanjhi NZ, Alshayeb M, Mahmood S (2020) Cyber security threats and vulnerabilities: a systematic mapping study. Arab J Sci Eng 1–19
3. Noorbehbahani F, Rasouli F, Saberi M (2019) Analysis of machine learning techniques for ransomware detection. In: 2019 16th international ISC (Iranian Society of Cryptology) conference on information security and cryptology (ISCISC), pp 128–133
4. Al-rimy BAS, Maarof MA, Shaid SZM (2018) Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions. Comput Secur 74:144–166
5. Sethi K, Chaudhary SK, Tripathy BK, Bera P (2018) 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, pp 1–4
6. Shijo PV, Salim A (2015) Integrated static and dynamic analysis for malware detection. Procedia Comput Sci 46:804–811. ISSN: 1877-0509
7. Sgandurra D, Munoz-Gonzalez L, Mohsen R, Lupu EC (2016) Automated dynamic analysis of ransomware: benefits, limitations and use for detection. arXiv Prepr. arXiv:1609.03020
8. Manavi F, Hamzeh A (2020) A new method for ransomware detection based on PE header using convolutional neural networks. In: 2020 17th international ISC conference on information security and cryptology (ISCISC), pp 82–87

9. Vinayakumar R, Soman KP, Velan K, Ganorkar S (2017) Evaluating shallow and deep networks for ransomware detection and classification. In: 2017 international conference on advances in computing, communications, and informatics (ICACCI), pp 259–265
10. Homayoun S, Dehghantanha A, Ahmadzadeh M, Hashemi S, Khayami R (2020) Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. IEEE Trans Emerg Topics Comput 8(2):341–351
11. Vidyarthi D, Kumar CRS, Rakshit S, Chansarkar S (2019) Static malware analysis to identify ransomware properties. Int J Comput Sci Issues 16(3):10–17
12. Zhang H, Xiao X, Mercaldo F, Ni S, Martinelli F, Sangaiah AK (2019) Classification of ransomware families with machine learning based on N-gram of opcodes. Future Gener Comput Syst 90:211–221
13. Bahrani A, Bidgly AJ (2019) Ransomware detection using process mining and classification algorithms. In :2019 16th international ISC (Iranian Society of Cryptology) conference on information security and cryptology (ISCISC), pp 73–77
14. El-Kosairy A, Azer MA (2018) Intrusion and ransomware detection system. In: 2018 1st international conference on computer applications & information security (ICCAIS), pp 1–7
15. Rezaei T, Hamze A (2020) An efficient approach for malware detection using PE header specifications. In: 2020 6th international conference on web research (ICWR), pp 234–239
16. Manavi F, Hamzeh A (2021) Static detection of ransomware using LSTM network and PE header. In: 2021 26th international computer conference, Computer Society of Iran (CSICC), pp 1–5
17. Belaoued M, Mazouzi S (2016) A chi-square-based decision for real-time malware detection using PE-file features. J Inf Process Syst 12(4):644–660
18. Vyas R, Luo X, McFarland N, Justice C (2017) Investigation of malicious portable executable file detection on the network using supervised learning techniques. In: 2017 IFIP/IEEE symposium on integrated network and service management (IM)
19. Benkessirat A, Benblidia N (2019) Fundamentals of feature selection: an overview and comparison. In: 2019 IEEE/ACS 16th international conference on computer systems and applications (AICCSA), pp 1–6
20. Baldwin J, Dehghantanha A (2018) Leveraging support vector machine for opcode density-based detection of crypto-ransomware. In: Cyber threat intelligence. Springer, pp 107–136
21. Hassen M, Carvalho MM, Chan PK (2017) Malware classification using static analysis-based features. In: 2017 IEEE symposium series on computational intelligence (SSCI), pp 1–7
22. Powers DM (2011) Evaluation: from precision, recall and *F*-measure to ROC, informedness, markedness and correlation
23. Jayanth D. PE-header data. https://www.kaggle.com/datasets/dasarijayanth/pe-header-data