# On the Design of Supervised Binary Classifiers for Malware Detection using Portable Executable Files

Hrushikesh Shukla, Sonali Patil, Dewang Solanki, Lucky Singh, Mayank Swarnkar, *Member, IEEE*, Hiren Kumar Thakkar, *Member, IEEE*

***Abstract*—Executable files such as .exe, .bat, .msi etc. are used to install the software in Windows-based machines. However, downloading these files from untrusted sources may have a chance of having maliciousness. Moreover, these executables are intelligently modified by the anomalous user to bypass antivirus definitions. In this paper, we propose a method to detect malicious executables by analyzing Portable Executable (PE) files extracted from executable files. We trained a supervised binary classifier using features extracted from the PE files of normal and malicious executables. We experimented our method on a large publicly available dataset and reported more than 95% of classification accuracy.**

**Keywords: Machine Learning, Malware Analysis, Feature Extraction, Portable Executable.**

## I. INTRODUCTION

In the past few years, there has been a significant increase in the demand for data analysis in various fields such as Healthcare [1], Network analysis [2] etc. The malware detection analysis in networks and executable files is also becoming a prominent research area considering the threat of malware attack [3–5] to computer users. It is reported that 670 million new malware files were added in 2017 while 246 million new malware files were added in 2018. Fig. 1 shows the generation of new malware files from 2014 to 2018. Malware can cause a lot of damage to the system such as data loss, data leakage, hardware damage, etc. The usual practice of malware analysis by antivirus systems is to first manually examine the malicious file and generate the corresponding signature. Later, the signatures are forwarded to the client in regular updates. However, manual inspection of malicious files to generate the signature is a very time consuming, tedious, and error-prone approach. Further, it requires to recruit domain expertise to generate the signature. To overcome the aforementioned limitations, Machine Learning (ML) techniques are applied to automate the task of classifying an executable file as malicious or benign.

There are mainly two approaches to analyze malicious files such as dynamic and static analysis. The Dynamic approach includes executing the file in a protected environment such as sandbox to collect the behavioral information of a file and corresponding changes in the environment made by it. Dynamic analysis is more reliable. However, it requires a very complex setup. Moreover, the dynamic analysis is a bit slower over the static analysis. On the contrary, the static analysis includes examining the modules of an executable file without executing it. The typical components of an executable file include the metadata, assembly commands (i.e., machine code instructions), and binary data. The aforementioned components are easy to analyze and required a simple setup. However, it is to note that the static, as well as dynamic approaches, are equally effective.

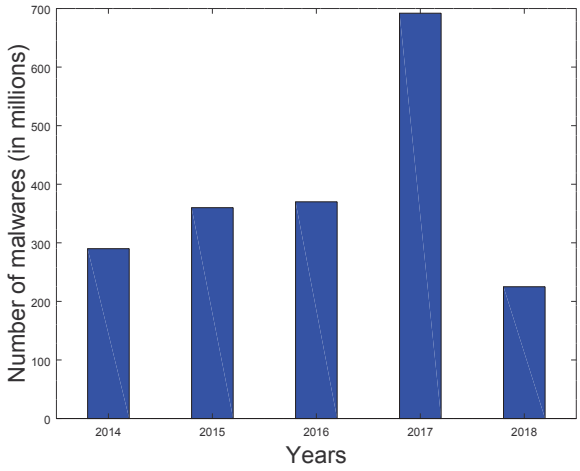


Fig. 1: Number of new malware samples generated in 2014-2018.

In this paper, we use a static analysis approach. To confirm the given file is malicious, we primarily use the corresponding Portable Executable (PE) file for the analysis purpose. PE file of any executable file is given to the windows loader. This file contains the information such as the size of the code, size of overlay, the order in which different sections of the file are executed. The aforementioned information helps us to understand the execution flow of any PE file. Besides, it also helps us to determine malicious files. To be specific, our contributions are as follows.

- We retrieve the PE file from the given executable files.
- We performed feature extraction on PE files and extracts the required features for the analysis.

Hrushikesh Shukla, Sonali Patil, and Dewang Solanki are associated with the department of Computer Science and Engineering, Smt. Indira Gandhi College of Engineering, India. Email: stanhrishi@gmail.com, sp865590@gmail.com, dewangsolanki3@gmail.com.

Lucky Singh is associated with the department of Computer Science and Engineering, Mody University of Science and Technology, India. Email: imluck121@gmail.com

Mayank Swarnkar and Hiren Kumar Thakkar are associated with department of Computer Science and Engineering, Bennett University, India. Email: mayank.swarnkar@bennett.edu.in, hirenkumar.thakkar@bennett.edu.in.

978-1-7281-4392-7/19/$31.00 ©2019 IEEE

- The machine learning classifiers are implemented to classify the malicious and benign executable.
- We reported the results for the best performing machine learning classifier.

Rest of the paper is organized as follows. In Section II, we describe the related works. In Section III, we describe our methodology followed by experimental details in Section IV. The concluding remarks are made in Section V.

## II. RELATED WORK

In this section, a few of the relevant studies are described. In [6], different machine learning methods on static analysis of malware files are presented. In [7], Shafiq et al., have employed data mining techniques on PE files to determine malicious files. The dataset is obtained from VX heavens and Malfese and the corresponding accuracy was 99%. Ranveer and Hiray performed malware detection using static, dynamic and hybrid analysis of the executable [8]. Shabtai et al. synthesized a taxonomy that classified detection methods for malicious codes by machine learning techniques using static features extracted from executables [9]. Santos et al. used n-grams signature. They obtained a dataset of 100GB out of which they used 1000 malware and 1000 benign samples training and testing purposes[10]. Moskovitch at el. took a dataset of 30,000 samples and performed byte sequence n-gram representation with a possibility of achieving 95% accuracy with lesser than 20% malicious samples[11]. Further, they evaluated Opcode n-gram representation and claimed that accuracy more than 99% is possible with a malicious dataset lesser than 15% which is more than their previous experience [11]. Singhal and Raul proposed an antivirus engine that extracts API calls and applies machine learning algorithms to detect harmful executable [12]. Xiao et al. Proposed a method to extract API calls and analyze them using data mining techniques for malware detection. In their method, they implemented data mining algorithms like objective-oriented associate mining (OOA) for mining association rules. To get association rules with high discrimination power, they presented an improved algorithm for frequent item generation. This algorithm is dependent mainly on support and classification capability[13]. Vyas at el. Proposed a method to detect portable executable files on a network using machine learning methods. In this method they shortlisted 28 features that were extracted from metadata, packing, imported from DLL files. Their system achieves an accuracy of 98.7% and 1.8% false-positive rate[14]. Yuxin and Siyi, in their study, proposed a model that represents a malware as a sequence of opcodes and then use deep belief network (DBN). They used unsupervised learning methods to pre-train multi-layer generator models which as per them can better represent the characteristics of the data samples. Markel at el. Worked on the detection of malware based on statistical properties of executable files using heuristic classification scheme which is hypothesis-based and adjustable to different application scenarios. As the heuristic approach applies limitations on the evaluation of static features, they feel that this approach is optimal for malware detection [15]. Acs and Jurecek in their thesis proposed a model to detect a malicious file using machine learning. They used features extracted from PE files as input to their method. The accuracy that they achieved from their machine learning technique is 95%. There are few other traffic classification techniques [16], [17], [18] which analyze network traffic which in turn can be used to detect anomalies in the network

## III. METHODOLOGY

In this section, we elaborate on our proposed model for the detection malware file. Our main focus in this project is to analyze the features extracted from the PE of an executable. PE of an executable is a collection of data elements that are required by the windows loader. PE file contains various elements like the size of code, size of data sections, overlay number. With the help of the PE file, one can understand how a program is going to execute. Fig. 2 shows a small part of the PE file which we extracted from an executable file and converted it into a text file for the ease of extracting features.

### A. System Architecture

In Fig. 3, we show the architecture of our system. We took a dataset and trained our model. After Training is complete we considered the executable files, extracted features from it and passed it through the trained model for evaluation purposes. The metric we use for our evaluation of our model is accurate. Our model is a binary classifier that takes input as an executable file and gives output as a label showing whether a file is a malware or not.

```
DOS Header
    Magic number:                    0x5a4d (MZ)
    Bytes in last page:              144
    Pages in file:                   3
    Relocations:                     0
    Size of header in paragraphs:    4
    Minimum extra paragraphs:        0
    Maximum extra paragraphs:        65535
    Initial (relative) SS value:     0
    Initial SP value:                0xb8
    Initial IP value:                0
    Initial (relative) CS value:     0
    Address of relocation table:     0x40
    Overlay number:                  0
    OEM identifier:                  0
    OEM information:                 0
    PE header offset:                0xb8
```

Fig. 2: A sinppet of PE file.

### B. System Components

The system components are described as follow.

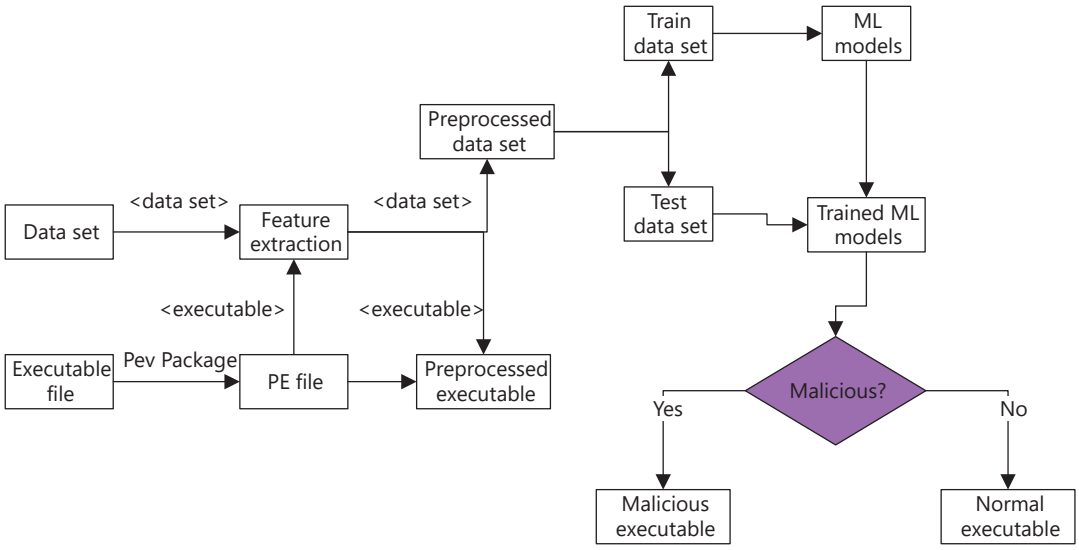


Fig. 3: System Architecture.

*1) Dataset:* We trained our model using a publicly available dataset. The dataset contains features extracted from PE files and stored in a CSV file.

*2) PE Files:* This block helps us when we input an executable file in our system. It extracts the PE file of the input executable and arranges that file in the text file in a defined way. This arrangement of PE components into a text file helps us to extract the required features from the file. The features are then passed on to the Feature Extraction block for the extraction of features.

*3) Feature Extraction:* The input to this block can be a sample from the dataset or it can be PE file in text format. This block extracts the features from the input data and does the required preprocessing.

*4) ML Classifiers:* The preprocessed dataset is then divided into training and testing sets respectively. We varied the size of training and testing sets to get different results. We even applied multiple machine learning algorithms and compared their accuracy obtained on the testing set.

*5) Trained model:* We save the best-trained model and then pass the testing files through the model. This saved model can help us to determine whether the input file is malware or not.

## IV. EXPERIMENTAL RESULTS

In this section, we are going through different experimental results we obtained by comparing various machine-learning algorithms.

### *A. Data Description and preprocessing*

We obtained our dataset of malware files through Kaggle. The dataset consists of features extracted from the PE file of an executable file. All the file samples are described by 75 different features like the size of code, size of overlay, etc. The last column of the dataset represents the label of the file, which identifies whether the file is malware or benign. As seen in Fig. 4, there are in total 19,612 samples, out of which, 14,599 malware file samples and 5,012 benign samples. After going through the dataset, we concluded that nearly 25 of the features are have repeated values and are redundant. Even the number of malware samples is more than benign samples.

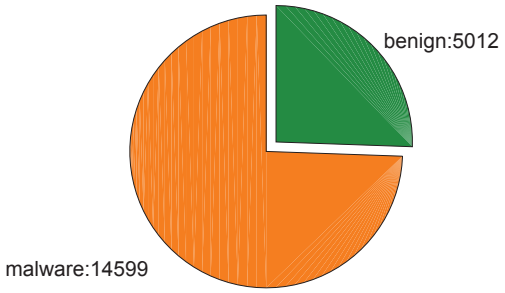


Fig. 4: Number of malware and benign samples in our dataset.

As mentioned previously, our dataset contains nearly 75% of malware files 25% of benign files which might create a bias in the machine. To overcome this problem we randomly sampled malware files and selected the number of malware

files equal to the benign files. The number of malware and benign files are 5000 each.

Out of 75 features, we came across 25 of the features were having common or repeated values. As a result, we made these features redundant and removed them from our dataset. The features had very sparse values that might not train our model well. For this purpose, we normalized the dataset values between 0 to 1.

### *B. Experimental setup and results*

Our machine learning model takes the features extracted from PE file and classifies whether a file is malicious or not.

### *C. Experimental setup*

The machine learning classification algorithms we used, include Naive Bayes, Decision Tree, Random Forests, Logistic Regression, Artificial Neural Networks(ANN).We trained our model on different ratios of training and testing size. First, we took the training size of 67% of the balanced dataset and 33% of the same for testing purposes. Then we took 90of balanced data for training and 10% of the data for testing purposes. As the number of samples is very large even 1000 (10% of total data) were sufficient for testing purposes.

### *D. Results and discussion*

Fig. 5 and 6 show the accuracy of the algorithms before and after normalizing the data, respectively. We kept the training size as 67% and testing size as 33%.

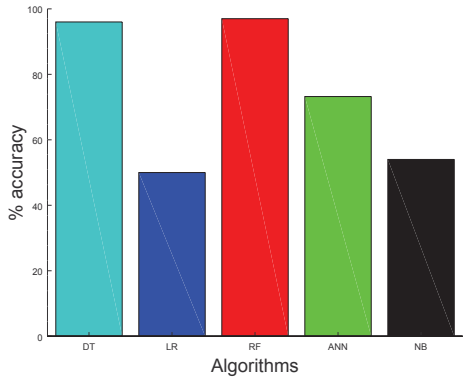


Fig. 5: Performance evaluation of classifiers for not normalized data set with train and test set of 67% and 33%.

Fig. 7 and 8 shows the percentage accuracy of the algorithms before and after normalizing the data when the size of training is fixed to 90% while the size of testing is fixed to 10%. Fig. 9 shows the accuracy after performing 5-Fold cross-validation on the dataset. Before performing the cross-validation we normalized the dataset.

We can observe that there is great change in the performance of ANN model after the dataset is normalized. The Naive Bayes algorithm does not perform well. The performance of Naive Bayes is not affected by normalization or by increasing the training size of the dataset. The accuracy obtained from

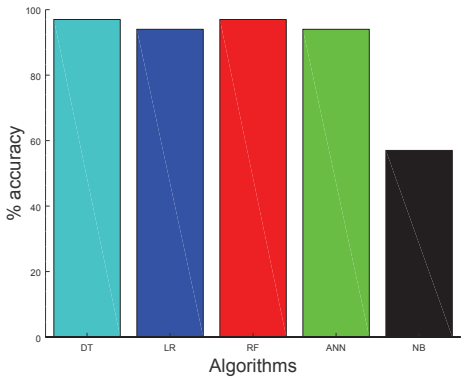


Fig. 6: Performance evaluation of classifiesr for balanced data set with train and test set of 67% and 33%.

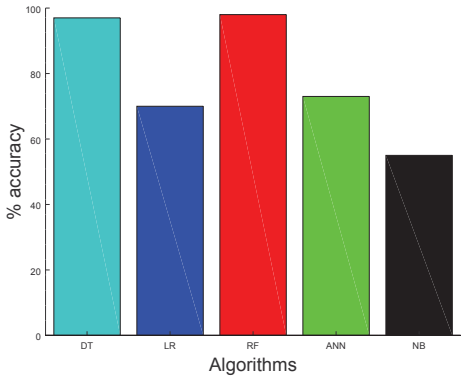


Fig. 7: Performance evaluation of classifiers for not normalized data set with train and test set of 90% and 10%.

Logistic Regression increases a lot if we apply normalization. The performance of Random forest is as consistent as of Decision trees but Random forest performs a little better than Decision trees. Hence, we selected Random forest as our final model and used it for testing purposes. Further, to extensively evaluate the different machine learning classifiers, 5-fold cross-validation is performed. The process of the 5-fold cross-validation is briefly described as follows. The entire training dataset is divided into five sub-datasets with equal training samples. Later, the classifiers are trained on four sub-datasets and subsequently validated on the one sub dataset. The aforementioned process is repeated and each time one sub dataset is kept aside for the validation purpose. Fig. 9 reports the performance of different ML classifiers for the 5-fold cross-validations. On the line of our previous observations, the random forest classifier achieves superior performance.

## V. CONCLUSION

In this paper, we designed a robust binary classifier, which consistently classifies the files into malicious and benign with 97.2% accuracy. The proposed model is limited only to detect whether the file is malware or not. The model does not provide any information about the family from which the malware is derived. The model might not give satisfactory

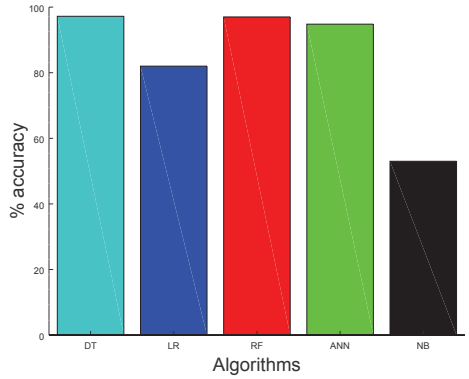


Fig. 8: Performance evaluation of classifiers for normalized data set with train and test set of 90% and 10%.

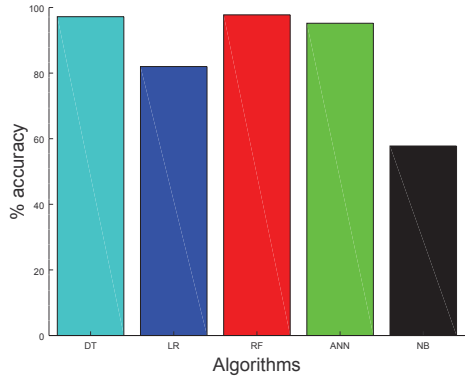


Fig. 9: Performance evaluation of classifiers across 5-fold cross validation for normalized data set with train and test set of 90% and 10%.

results if the executable file is encrypted.

The future scope of the paper might include actions to be taken when the file is detected to be malicious.

## REFERENCES


[1] P. Sahoo, H. Thakkar, W.-Y. Lin, P.-C. Chang, and M.-Y. Lee, "On the design of an efficient cardiac health monitoring system through combined analysis of ecg and scg signals," *Sensors*, vol. 18, no. 2, p. 379, 2018.

[2] P. K. Sahoo and H. K. Thakkar, "Tls: traffic load based scheduling protocol for wireless sensor networks," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 30, no. 3, pp. 150–160, 2019.

[3] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, "Malware dynamic analysis evasion techniques: A survey," *arXiv preprint arXiv:1811.01190*, 2018.

[4] M. Swarnkar and N. Hubballi, "Ocpad: One class naive bayes classifier for payload based anomaly detection," *Expert Systems with Applications*, vol. 64, pp. 330–339, 2016.

[5] ——, "Rangegram: A novel payload based anomaly detection technique against web traffic," in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*. IEEE, 2015, pp. 1–6.

[6] H. V. Nath and B. M. Mehtre, "Static malware analysis using machine learning methods," in *International Conference on Security in Computer Networks and Distributed Systems*. Springer, 2014, pp. 440–450.

[7] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "Peminer: Mining structural information to detect malicious executables in realtime," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2009, pp. 121–141.

[8] S. Ranveer and S. Hiray, "Comparative analysis of feature extraction methods of malware detection," *International Journal of Computer Applications*, vol. 120, no. 5, 2015.

[9] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," *information security technical report*, vol. 14, no. 1, pp. 16–29, 2009.

[10] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based file signatures for malware detection." *ICEIS (2)*, vol. 9, pp. 317–320, 2009.

[11] R. Moskovitch, D. Stopel, C. Feher, N. Nissim, and Y. Elovici, "Unknown malcode detection via text categorization and the imbalance problem," in *2008 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 2008, pp. 156–161.

[12] P. Singhal and N. Raul, "Malware detection module using machine learning algorithms to assist in centralized security in enterprise networks," *arXiv preprint arXiv:1205.3062*, 2012.

[13] X. Xiao, D. Yuxin, Z. Yibin, T. Ke, and D. Wei, "Malware detection based on objective-oriented association mining," in *2013 International Conference on Machine Learning and Cybernetics*, vol. 1. IEEE, 2013, pp. 375–380.

[14] R. Vyas, X. Luo, N. McFarland, and C. Justice, "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)*. IEEE, 2017, pp. 941–946.

[15] R. Merkel, T. Hoppe, C. Kraetzer, and J. Dittmann, "Statistical detection of malicious pe-executables for fast offline analysis," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2010, pp. 93–105.

[16] G. Sun, L. Liang, T. Chen, F. Xiao, and F. Lang, "Network traffic classification based on transfer learning," *Computers & electrical engineering*, vol. 69, pp. 920–927, 2018.

[17] N. Hubballi and M. Swarnkar, "*bitcoding*: Network traffic classification through encoded bit level signatures," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2334–2346, 2018.

[18] ——, "Bitcoding: Protocol type agnostic robust bit level signatures for traffic classification," in *GLOBECOM*

*2017-2017 IEEE Global Communications Conference*.
IEEE, 2017, pp. 1–6.