Volume 9, Issue 3, March – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR2188

A Framework for Detection of Malicious Code by Exploiting Machine Learning Techniques on Portable Executables

Yash Gajjar[*1]; Vaishnavi Sharma[*2]; Sanskruti Bhatt[*3]; Dr. Maitri Jhaveri[4]
[1,2,3,4]Department of Computer Science, Gujarat University – 380009

Abstract:- Executable files coming from the internet bring along with them many potential hazards and vulnerabilities in the form of malware to computer systems. The executables can be of form raw binaries, mnemonics, libraries, and function calls/APIs. They can misguide many of the conventional malware detection techniques. This paper explores the potential of Machine Learning-based methods for malware detection problems. The scope of the work here is currently limited to Static Analysis of Executable files. Various feature selection techniques are implemented to reduce the size of the training data. Machine learning algorithms like K-Nearest Neighbors and Random Forest Classifier were trained on the curated feature sets. The outperforming experiment result was shown by the Random Forest Classifier having an accuracy of 99.5%. We have developed a framework as a two-step module; in the first step, a list of features are extracted from a given executable file, and then for the next step, trained algorithm is integrated into the framework which will classify whether the given executable file is malicious or not. This framework is demonstrated in the form of a Webapp developed in Python. Furthermore, this framework is evaluated based on its performance on a small dataset containing 35 portable executables (.exe) files and it is observed to be retaining the accuracy of the trained algorithm.

Keywords:- *Portable Executables (PE), Malicious Code, Machine Learning (ML).*

I. INTRODUCTION

Computers nowadays are an important part of every sector. In this digital age, the transfer of data, information, software, etc. between computer systems and external networks is a common practice that can introduce malware, vulnerabilities, or other risks. Any program or file that purposefully hurts a computer, network, or server is known as malware or malicious software. Computer viruses, trojan horses, ransomware, worms and spyware are a few examples of malware. These malicious programs can steal, alter, encrypt, hijack, or delete sensitive data, core computing functions and they can even monitor user's computer activity. The way malware harms the users or endpoints can vary depending on its type, ranging from mild and harmless to severe and catastrophic consequences.

Executable files coming from the internet bring the highest vulnerability to any computer system. The executable can be raw binaries, mnemonics, libraries, and API/function calls. They can misguide many of the traditional malware detection techniques such as Signature, Check summing, Reduced Masks, known Plain text Cryptanalysis, Statistical analysis, Heuristics, and Sandboxing. The next-generation techniques include AI/Machine-Learning-Based Static Analysis, NLP-based techniques, Application Whitelisting, Endpoint Detection, and Response. Machine Learning algorithms can replace the rule-based approach of detecting malicious code, where different algorithms can be trained on the dataset consisting of the features of executable files. Such trained models can classify between Legitimate and Malicious files and can reduce the hectic work of analyzing executable files manually. Further, these trained models can be retrained on new datasets for better predictions of malicious files.

This paper focuses on Machine Learning based detection using Portable Executable (PE) files. Windows (both x86 and x64) utilizes the PE file format, which serves as a structured data container that holds the necessary information needed for the Windows OS loader to manage the wrapped executable code. The PE format is a file format for executable, object code, DLLs, FON font files, and core dumps. The kind of code which are malicious is attached to PE files.

The techniques for identifying malware can be categorized into static and dynamic analysis. In static analysis, executable files are not executed but the tools and apps can be used to get the required forensic information and the values of its features can be extracted. While, in dynamic analysis, the executable files are executed in a safe environment and then observed and classified. The work here is currently limited to the Static analysis of PE files. We have developed a framework that extracts features from portable executable files and will then classify these files as legitimate or malicious. We have applied binary classification algorithms on labelled data in this work.

II. RELATED WORKS

(Kim et al., 2020) proposed a static analysis automation technique using machine learning to classify malicious code. Using variety of algorithms like Random Forest Classifier, AdaBoost, Gaussian Naïve Bayes, Logistic Regression and Decision Tree, they extracted and classified several distinctive characteristics, including packer information, PE

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR2188

metadata, and hash value. (Kumar et al., 2019) proposed a technique that uses static analysis to extract features with lower time and resource requirements than dynamic analysis. By combining raw and derived features based on various PE file header field values, they have produced an integrated feature set that has the classification accuracy of 98%. (Shijo & Salim, 2015) have developed an integrated approach using both static and dynamic features for malware detection and their results show that the support vector machine (SVM) algorithm is best equipped to classify the data. (Chaudhary, 2021) have identified the most suitable features to detect malicious executable files using both static and dynamic analysis techniques. A simpler and faster method to distinguish between malware and legitimate .exe files by analyzing some key features from MS Windows PE headers was proposed by (Liao, 2018). He also performed icon extraction to identify malware by extracting the embedded icons such as the prevalent or misleading. (Abdessadki & Lazaar, 2019) extract features from the header of each file, which are then used as input for machine learning algorithms for classifying PE files without executing them. (Baldangombo et al., 2013) developed a PE-Miner program to parse the PE format of the Windows executable in their dataset. The PE Miner extracts all PE header information, DLL names, and API function calls inside each DLL contained in a PE file. They utilize data mining techniques such as Information Gain and PCA transformation, due to which the system extracts valuable features from Windows PE files and achieves a high detection rate using machine learning and data mining concepts. (Schultz et al., 2001) extracted the information using PE files and proposed a framework based on ML to detect the malicious PE files. The author's dataset contained 4266 samples from which 3265 are malicious and 1001 are benign files. They have used three Machine Learning algorithms – Ripper, Naïve Bayes, and Multi-Naïve Bayes out of which Multi-Naïve Bayes had the highest accuracy and detection rate of about 97.76%. Their framework automatically detects malicious executables, significantly improving detection rates compared to traditional methods.

III. AVAILABILITY OF DATA AND MATERIALS

The raw data was gathered from the malware security partner of Meraz'18, the annual techno-cultural festival of IIT Bhilai. The information extracted from several PE files in the form of 55 features, is contained in the raw data (CSV data). In our work, we have used two datasets namely, dataset-1 (75,502 Legitimate and 140,848 Malicious) and dataset-2 (41,323 Legitimate and 96,724 Malicious). The data is a mixture of categorical and continuous values.

Table 1: Preview of both Datasets

	ID	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData
0	1	34404	240	8226	14	12	78848	57856
1	2	332	224	8450	14	12	43520	10752
2	3	332	224	8226	48	0	424960	2048
3	4	34404	240	8226	14	12	414720	314880
4	5	332	224	8450	14	13	123392	45056

IV. RESEARCH METHODOLOGY

We propose a machine learning-based model for detecting malicious executable files. The problem is implemented as a classic supervised learning problem that classifies an input file into either of two classes, i.e., Malicious or Legitimate.

A. Preprocessing and Feature Selection

Preprocessing steps include the removal of string feature 'md5', which does not contribute to the classifier model. Detailed examination of the dataset reveals that out of 55 features, only a limited number of features have the information that can differentiate between malicious and legitimate files. And hence, feature selection becomes an important part of the process. We have performed feature selection by following methods:

- *Correlation Coefficient Method*

Correlation calculates the linear relationship between variables. Features which are important should be highly correlated with the target variable. Furthermore, if two variables are highly correlated then we can drop the one which has low correlation coefficient value with the target variable, as the model only needs one of them and the second one does not add any information.

- *Chi-Square (χ^2) Method*

The Chi-square (χ^2) test can be used as a feature selection method when our dataset contains categorical features. The chi-square distribution is a sampling distribution and is a family of probability distributions based on the number of degrees of freedom (df). A chi-square variable cannot be negative and the area under each chi-square distribution is equal to 1.00, or 100%. Chi-square value is calculated between each feature and the target variable. The features with the best Chi-square values are selected according to one's necessity. The purpose of chi-square analysis is not to identify the exact nature of a relationship between nominal variables but to simply test whether the variables could be independent of each other.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR2188

➢ *Gini Impurity-based Method*

This approach involves assessing the reduction in Gini impurity for each feature as it's utilized to partition the data. The extent of this reduction is determined by the proportion of data points influenced by the split. Features leading to greater drops in Gini impurity are considered more significant. To ensure that feature importance values sum up to 1, they are normalized. The formula for calculating Gini importance for a given feature Xi involves various parameters related to the nodes and splits in the decision trees (Breiman, 1984). This implementation utilizes the scikit-learn Python package for training and extracting feature importance from Random Forest Classifier.

B. Classification Framework

The classification framework is built as a web application. To build the framework, machine learning models like Random Forest Classifier, Decision tree Classifier, K-Nearest Neighbors, Support Vector Machine, and Gaussian Naive-Bayes are trained on various feature sets curated based on feature selection. These algorithms are being trained and tested on CSV datasets. Outperforming combinations of feature sets and trained models are then wielded for further integration with the framework. The work here is split up into two modules; the first module acts as an extractor that extracts the values of features from the portable executable taken as input and then passes the extracted values to the other module where the trained model is integrated for classification.

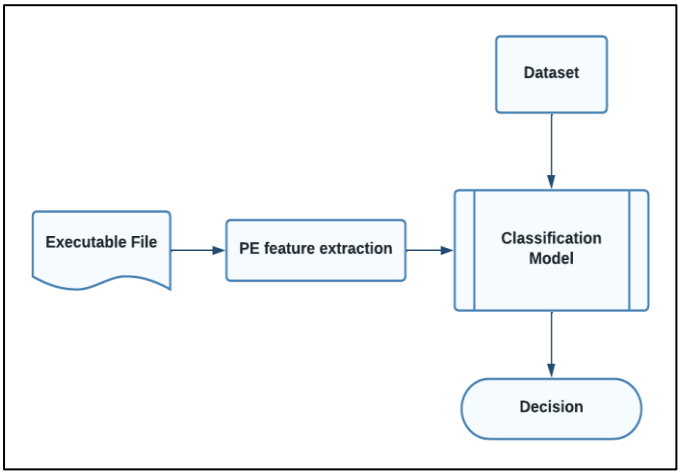

Fig 1: Flowchart of the Classification Framework

C. Experimentation

We have conducted several experiments; in the first experiment, we chose to train five classifiers namely Decision Tree Classifier, Random Forest Classifier, Gaussian Naïve Bayes, K-Nearest Neighbors, and Support Vector Machine. The data used for training has all the features (54 features) except md5. The train-test split criterion is kept the same for all the experiments conducted on dataset-1, i.e., 70% of the data is used for training and 30% for testing. The accuracy scores obtained by above mentioned algorithms in *experiment-1* are shown in Table 2.

Table 2: Accuracy Scores of Algorithms from Experiment 1

Algorithm	Accuracy
Decision Tree Classifier (DTC)	98.21%
Random Forest Classifier (RFC)	98.93%
Gaussian Naïve Bayes (GNB)	65.06%
K-Nearest Neighbors (KNN)	97.71%
Support Vector Machine (SVM)	65.06%

Out of all the trained models, the accuracy of KNN and RFC is very high. Random Forest is a combination of n decision trees (n - hyperparameter) and hence we are not considering the results based on DTC. Therefore, in the succeeding experiments, RFC and KNN are chosen, for training.

In *experiment 2*, the feature set used for training contains 10 features selected based on the Correlation method, and the training & testing are done on dataset 1. In *experiment 3*, the feature set used for training has 10 features selected based on χ^2 -test which shows how much a nominal feature is dependent on the targets. Here as well, dataset-1 is used for training and testing. In *experiment 4*, the feature set used for training has 12 features (top 10 from the Correlation method and top 2 from χ^2 -test). In *experiment 5*, the same feature-set of experiment 4 is used but the training is done on dataset-1 and testing is done on dataset-2. In *experiment 6*, the same feature set of experiment 4 is used but dataset-1 was balanced by under-sampling malicious class for training and then testing is done on dataset-2. Finally, in *experiment 7*, an updated feature set is curated which has 12 features (top 8 from the Correlation method and top 4 from the χ^2 -test). For training, balanced dataset-1 is used, and for testing dataset-2. The accuracy scores of both classifiers in above mentioned experiments are shown in the figure below.

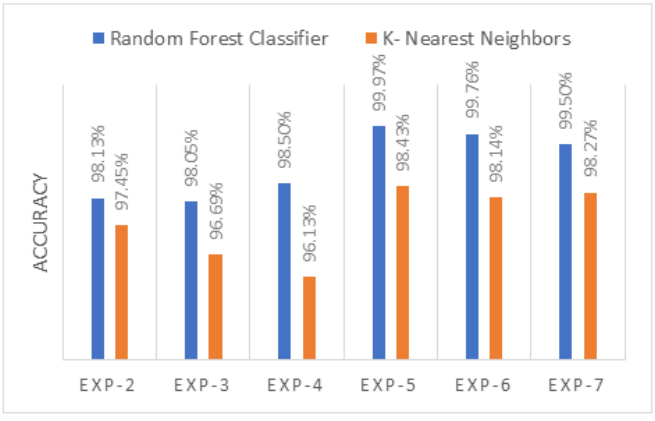

Fig 2: Accuracy scores of RFC and KNN from Experiment 2 to Experiment 7

V. RESULTS

In the creation of the web application, the trained model Random Forest Classifier from *experiment 7* is selected which has an accuracy of 99.50%. We selected this model for integration in Webapp as the features used while training in this experiment, are computationally convenient to extract

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR2188

from the raw PE file as compared to other features. The features selected for *experiment 7* and eventually for the framework are shown in Table 3.

Table 3 : Updated Combined Feature Set Used in Experiment 7 and in our Framework

Feature names	
Machine	SectionsMeanEntropy
SizeOfOptionalHeader	SectionsMaxEntropy
Characteristics	SizeOfStackCommit
MajorSubsystemVersion	SizeOfStackReserve
Subsystem	ImageBase
DllCharacteristics	CheckSum

VI. DISCUSSION

From the results obtained out of all the experiments, it can be concluded that the accuracy of the model Random Forest Classifier which is an ensemble model, is comparatively better which is integrated into our framework. Up until now, the models have been tested on CSV files and hence, the framework needs to be evaluated based on performance on PE files. A total of 35 PE files were downloaded, among them 16 files are legitimate, obtained from www.exefiles.com [accessed on 12 August 2022], and 19 files are malicious, obtained from www.tekdefense.com [accessed on 12 August 2022]. The web app was able to predict all 35 files correctly which shows that on such a small dataset model can maintain the accuracy of 99.50%.

VII. CONCLUSION

We observe that the Machine Learning-based classification algorithms are successfully able to classify an executable file into malicious or legitimate. Our best model is the Random Forest Classifier with 12 features (fusion of features selected from Correlation and Chi-square method) having an accuracy of 99.50% which is further integrated into our framework. The developed framework for detecting malicious files is found to be robust. Current work which focuses on static analysis of executable files, might be applied further to the executables of different extensions as well.

FUTURE SCOPE

The authors intend to execute the work using Deep Learning Algorithms to have a better efficiency of the developed web application. Furthermore, this work can be stretched for multi-class classification of malware and to the executables of different extensions.

- *Data can be Accessed through: https://www.kaggle.com/competitions/malware-detection/data.*

ACKNOWLEDGEMENTS

The work in this paper is done on Google Colab notebooks and the authors are especially thankful to Ero Carrera for pefile library (Carrera Ventura, 2022).

REFERENCES

[1]. Abdessadki, I., & Lazaar, S. (2019). A New Classification Based Model for Malicious PE Files Detection. International Journal of Computer Network and Information Security, 11(6), 1–9. https://doi.org/10.5815/ijcnis.2019.06.01
[2]. Baldangombo, U., Jambaljav, N., & Horng, S. (2013). a S Tatic M Alware D Etection S Ystem U Sing. 4(4), 113–126.
[3]. Breiman, L. a. (1984). In Classification and Regression Trees. Taylor \& Francis.
[4]. Carrera Ventura, E. (2022). pefile (2022.5.30). https://github.com/erocarrera/pefile
[5]. Chaudhary, P. (2021). PE File-Based Malware Detection Using Machine Learning PE File-Based Malware Detection Using. January. https://doi.org/10.1007/978-981-15-4992-2
[6]. Kim, S., Yeom, S., Oh, H., Shin, D., & Shin, D. (2020). Automatic malicious code classification system through static analysis using machine learning. Symmetry, 13(1), 1–11. https://doi.org/10.3390/sym13010035
[7]. Kumar, A., Kuppusamy, K. S., & Aghila, G. (2019). A learning model to detect maliciousness of portable executable using integrated feature set. Journal of King Saud University - Computer and Information Sciences, 31(2), 252–265. https://doi.org/10.1016/j.jksuci.2017.01.003
[8]. Liao, Y. (2018). PE-Header-Based Malware Study and Detection. 4.
[9]. Schultz, M. G., Eskin, E., Zadok, E., & Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, February 2001, 38–49. https://doi.org/10.1109/secpri.2001.924286
[10]. Shijo, P. V., & Salim, A. (2015). Integrated static and dynamic analysis for malware detection. Procedia Computer Science, 46(Icict 2014), 804–811. https://doi.org/10.1016/j.procs.2015.02.149