DETECTING MALWARE IN PORTABLE EXECUTABLE FILES USING MACHINE LEARNING APPROACH

Tuan Nguyen Kim[1], Ha Nguyen Hoang[2] and Nguyen Tran Truong Thien[3]

[1]School of Computer Science, Duy Tan University, Da Nang, Vietnam
[2]University of Sciences, Hue University, Vietnam
[3]Cybersecurity Center, Duy Tan University, Da Nang, Vietnam

ABSTRACT

There have been many solutions proposed to increase the ability to detection of malware in executable files in general and in Portable Executable files in particular. In this paper, we rely on the PE header structure of Portable Executablefiles to propose another approach in using Machine learning to classify these files, as malware files or benign files. Experimental results show that the proposed approach still uses the Random Forest algorithm for the classification problem but the accuracy and execution time are improved compared to some recent publications (accuracy reaches 99.71%).

KEYWORDS

PE file, PE header, Feature, Malware, Random Forest Algorithm.

1. INTRODUCTION

The term “malicious malware”, or malware, is used to refer to computer softwares that is developed for illegal purposes such as stealing data, corrupting data, damaging computers and computer systems of certain individuals and organizations. Malwares can hide/attach in any file, device on computers, and computer networks. In this article, we only focus on the types of malware hidden in executable files on the Windows operating system environment.

In recent years, malware has become a significant threat to security in cyberspace. Malware may survive on terminals, migrate over networks, or be attached to/hidden in executable files, particularly Windows PE files. Currently, there are two methods for detecting malware [1-2]. Despite its high accuracy, signature-based detection confronts several challenges due to the diversity and morphing capabilities of today's malware. This difficulty may be solved using non-signature-based detection approaches, which are frequently employed to detect "unknown" malware variants, which are potentially hazardous. When used in combination with a machine learning approach, this technology helps to classify and detect malware with high efficiency today [3-4].

PE (Portable Executable) files are executable files for the Windows operating system. These files can either be executable or contain binary code that can be used by other executable files. The format information area of PE files contains the information that the operating system needs to regulate the execution of the files once they've been loaded into main memory [5]. Because all PE files have the same structure and number of fields in the PE header, we can use these fields as input features in the process of utilizing Machine Learning algorithms to construct malware classification models for these files.

As we all know, the Windows operating system uses a standardized structure for the information included in the PE header of benign PE files. If a PE file has data in its PE header fields that is "different" from data found in benign PE files, it is almost certainly a malware. By looking at the data contained in the fields of a PE file's header, we could identify it as whether malware or benign. In addition, the number of fields in the PE header is large, the data in the fields is also related to each other, and most of the fields can be "different" at various levels. Therefore, the Malware Detection challenge should be approached utilizing Machine Learning algorithms to achive the highest possible accuracy.

We can collect many PE header samples of benign and malicious files, extract the characteristics of each field, and compare the results to discover the most significant differences between the benign and malicious files, which can then be used to classify the files. This is the method we use in this article to experiment and propose.

Although this study is only aimed at detecting malware hidden in executable files in the Windows operating system environment, this approach can be applied to executable files on other operating system environments such as: Linux, Mac, Android, and so on if we know the header structure of these files and have a trusted data set.

2. Related Works

There are many approaches to the malware classification problem using machine learning techniques. In this section, we will analyze the results of recently published approaches in terms of accuracy, detection rate, and training speed.

- In [5], Rushabh Vyah and partners proposed a procedure to detect malware in PE files on the network environment. They applied four different supervised learning algorithms, Decision Tree, K-NN, SVMs, and Random Forest, on the same data set, with only 28 static features. Vyas chose the Random Forest model in his research. The average malware detection rate for backdoor, virus, trojan, and worn of this model is 98.7% and the positive detection rate of that is 1.8%.
- Jinrong Bai and partners proposed an approach for malware detection in PE files by mining the format information of these files [6]. The “in-depth analysis” skill was chosen to analyze the format information field of PE files. Firstly, they extract 197 features from this format information field, then perform feature selection to reduce the number to 19 or 20 features. Then, the selected feature set will be trained by fours classification algorithms J48, Random Forest, Bagging, and Adaboost. Experimental results show that this approach achieved the highest accuracy of 99.1% when using the Random Forest classification algorithm.
- The approach proposed by Hellal and Lotfi Ben Romdhane [7] is a combination of two techniques which are static analysis and graph mining. They have proposed a new algorithm that can automatically extract common and distinct, but repeatable, patterns of malware behavior from suspicious files. This proposal is concerned with saving memory space and reducing scan time by generating a limited number of signatures, which is distinctive from existing methods. The approach in [7] achieved high recognition rate and low false positive rate with 92% accuracy.
- The author group then extracted the icons from the PE file to identify the most prevalent and misleading ones in the malware. Yibin Liao [8] examined the proposed approach on a dataset of 6875 samples, which included 5598 malicious and 1237 benign executable file header samples. The results showed that in less than 20 minutes, this method obtained a detection rate of more than 99% with fewer than 0.2% false positives. According to the author, malware can be detected by examining a few major features/fields in the PE header of PE files or by

looking at the common icons, which are false symbols encoded in these files. This reduced the time taking to identify malware in PE files.

Currently, we have not found a method, an approach or a model that is considered to be the most generic and optimal for detecting and classifying malware using Machine learning with the highest accuracy. Therefore, we propose a different approach, focusing on the high-impact fields in the PE header of PE files, as a small contribution to this research direction.

3. THE PROPOSED APPROACH

Our proposed approach is evaluated on a huge dataset, which includes 140,297 PE header samples, 44,214 malware samples, and 96,083 benign samples. We collected this data from the virusshare.com website and benign PE files on the Windows operating system.

We apply machine learning algorithms such as AdaBoost, Gradient Boosting, Decision Tree, Extra Tree, and Random Forest to develop classification models of PE files - malware or benign files - from this dataset based purely on the majority of the variables in the PE headers. The purpose of the experiment is to find a machine learning classification model with high accuracy and an acceptable training time.

We deleted the fields least affected by malware, such as LoaderFlags, NumberOfRvaAndSizes, SizeOfHeapCommit, SizeOfHeapReserve..., from the dataset using the information gained from surveying the fields in the PE Header section of these files, keeping only 44 fields. This is completely consistent with the results obtained using the Random Forest and Extra Tree methods to assess the influence of fields, specific features, in the PE headers of 140,297 PE samples in the dataset. According to Random Forest, the following table illustrates the influence of fields:

Table 1. The Influence of Fields in PE headers of PEs files according to Random Forest algorithms

	Fields in PE header	Level of affect
1	ImageBase	0.193689
2	SizeOfStackReserve	0.103419
3	VersionInformationSize	0.075304
4	MinorImageVersion	0.065888
5	ResourcesMinSize	0.058338
6	Characteristics	0.052923
7	ExportNb	0.052831
8	Subsystem	0.049870
9	MajorOSVersion	0.045429
10	ResourcesNb	0.037733
…	…	…
44	ImportsNbOrdinal	0.001600
45	LoadConfigurationSize	0.001275
46	FileAlignment	0.001175
47	SectionAlignment	0.001167
48	SizeOfHeaders	0.001088
49	SizeOfUninitializedData	0.001036
50	BaseOfCode	0.000832
51	SizeOfHeapReserve	0.000401
52	SizeOfHeapCommit	0.000225
53	NumberOfRvaAndSizes	0.000008
…	…	…

Reducing some fields of each PE header sample helps to reduce the size of the dataset, then resulting in a reduction in system resources used in the classification model building program. This reduction of fields also leads to a reduction in model training time, 13.04s and 12.52s for 54 features and 44 features, respectively.

The remaining part of our approach is conducted in the order of the four experiments listed in the following section.

4. EXPERIMENTAL RESULTS

4.1. Experiment 1

We randomly divide the dataset into 2 parts, 80% is the training set (Training set) and 20% is the test set (Test set). These two data sets are used to evaluate the accuracy and training time of Machine learning models according to 5 different algorithms. The results are given in Table 2.

Table 2. The accuracy and training time of Mlmodels

Algorithm	Accuracy	Training time
AdaBoost	99.12%	12.83 s
GradientBoosting	99.30%	30.76 s
DecisionTree	99.34%	0.98 s
ExtraTree	99.69%	9.74 s
RandomForest	99.71%	13.17s

This experiment shows that the model built by the Random Forest algorithm gives the highest accuracy, up to 99.71%, as compared to the other 4 algorithms, with the average training time. The Extra Trees model achieves a faster training time, but lower accuracy than the Random Forest model. The Decision Tree algorithm for the model has a very high training speed, but the accuracy is not as expected.

4.2. Experiment 2

Although the method of randomly dividing the data set into two parts as in experiment 1 is not complicated, the model's accuracy may be affected if overfitting occurs. To tackle the overfit/unoverfit problem in this experiment, we apply the k-fold technique [9], with K = 10. Table 3 displays the acquired results.

Table 3. The accuracy of k-fold ML models with K = 10

Algorithm	Average accuracy	Min accuracy	Max accuracy
AdaBoost	99.11%	99.05%	99.17%
GradientBoosting	99.31%	99.24%	99.37%
DecisionTree	99.34%	99.26%	99.42%
ExtraTree	99.71%	99.67%	99.75%
RandomForest	99.72%	99.66%	99.76%

From the results obtained in Experiment 1 and Experiment 2, we choose the Random Forest algorithm to build a classification model for our proposal, because the accuracy it provides is the highest (99.71% and 99.72%) and with a reasonable training time.

4.3. Experiment 3

In this experiment, we will find out whether increasing the number of Trees in the Random Forest model improves accuracy, and if so, how many Trees are needed for the model to work faster and with higher accuracy.

We first try to create 10 Random Forest models with only one tree, then gradually increase to 500 trees, for each increase we will average the accuracy and training time of 10 models. The results are shown in 2 charts below (Fig. 1a and Fig. 1b):

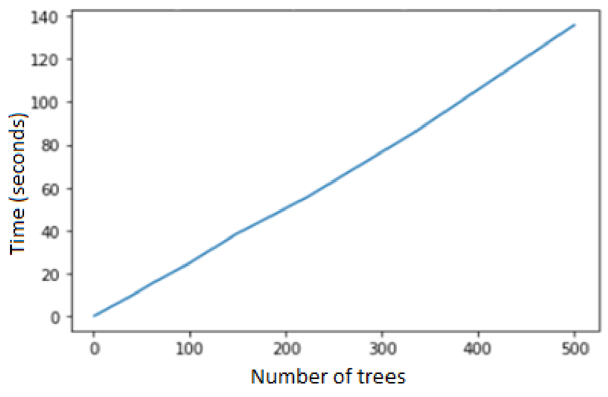

Fig. 1a. The training time when increasing the number of trees

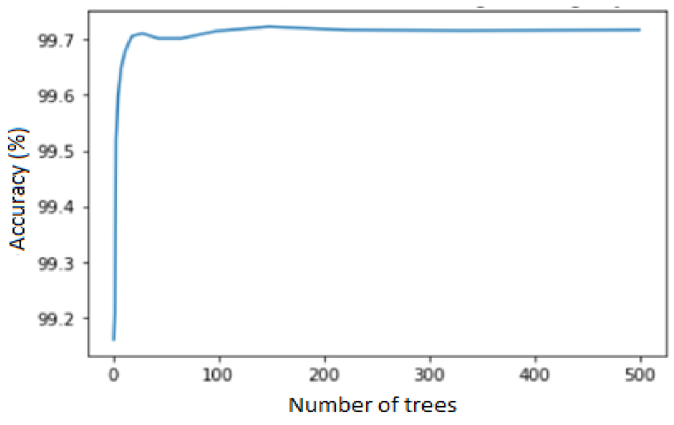

Fig. 1b. The accuracy when increasing the number of trees

The accuracy when the number of Trees is less than 20 is very low. After the Trees number reaches 50 starts, the accuracy starts eventually increasing. When the number of Trees obtain 100, the training time increases with that number. This illustrates that we only need a suficient number of Trees (100 in this case) for the model to achieve a high accuracy. The reduction in the number of trees reduces training time and saves system resources. This is something to be reconised.

4.4. Experiment 4

By selecting only 44 features, equivalent to 44 fields in the PE header of PE files, our

RandomForest machine learning classification model has an average accuracy rate and remarkable training time, 99.72% and 13.17s, respectively. We experiment to further reduce the number of selected features, to see if the accuracy rate and model training time are changed. The results are as follows, when the number of features is selected between 13 and 15, the average accuracy rate is 99.63% and the training time is 3.88s.

As it is shown in this experiment when reducing the number of features as much as possible, the average accuracy rate decreases only by a negligible amount, 0.09%, but the reduction in training time is remarkable, 9.29s (70%), compared to the original. Reducing the number of features also reduces the size of the dataset, reduces the time it takes to extract fields from the PE header of PE files, speeds up malware detection, and increases system performance.

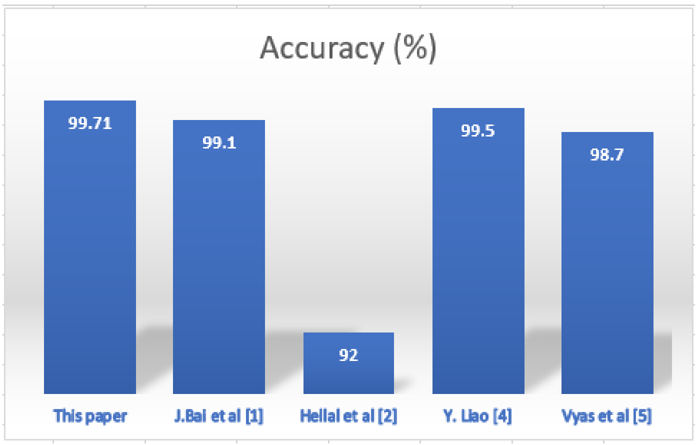

Fig. 2. Comparison between the accuracy in this proposal and that of a recent research

Thus, our approach to classifying malware based on the E header of PE files has achieved the recorded accuracy compared to some recent publications (Figure 2).

In the future, we will conduct experiments similar to the approach described in [10-11].

5. Conclusion

This article proposes a different approach for malware detection on PE files. Our proposal is tested on a huge dataset, including headers of 149,297 PE files that consist of 44,214 malware files and 96,083 benign files. As the experimental results show, even without evaluating total fields in the header and removing the least influential fields, the Random Forest algorithm still provides pretty high accuracy as compared to 4 other algorithms.

This accuracy is calculated to be up to 99.71%, with a training average of 13.17s. In addition, the experiments also figure that the accuracy of Random Forest is determined by selecting the appropriate number of Trees rather than a large number of Trees. Finally, the reduction of the number of Trees and their moval of the less important fields improved the model training speed (70% reduction), malware detection speed, and system resources.

In the future, we will research and propose more advanced solutions to improve the accuracy and speed of malware detection on various file types.

REFERENCES

[1] Priya A., Singh K., Tiwari R., (2016) A Review on Malware Analysis by using an Approach of Machine Learning Techniques, Smart Moves Journal Ijosthe, vol. 3, no. 5, pp. 1-5.

[2] Souri A., Hosseini R., (2018) A state-of-the-art survey of malware detection approaches using data mining techniques, Human-centric Computing and Information Sciences, vol. 8, pp. 1-6.

[3] Fabio, Troia D., (2020) Machine learning classification for advanced malware detection. PhD thesis, Kingston University, Section 5-7.

[4] Rosli N. A., Yassin W., Faizal M. A., Selamat S. R., (2019) Clustering Analysis for Malware Behavior Detection using Registry Data, IJACSA, vol. 8, iss. 12, pp. 95–120.

[5] Vyas, Luo R., McFarland X., Justice N., (2017) Investigation of malicious portable executable file detection on the network using supervised learning techniques, IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 941–946.

[6] Bai J., Wang J., Zou G., (2014) A Malware Detection Scheme Based on Mining Format Information, The Scientific World Journal, vol.2014, Article ID 260905, pp.1-11.

[7] Hellal A., Romdhane L. B., (2016) Minimal Contrast Frequent Pattern Mining for Malware Detection, Computers & Security, vol.62, pp.19-32.

[8] Liao Y., (2012) Pe-Header-Based Malware Study and Detection, Security & Privacy Workshop, San Francisco, CA, U.S.A.

[9] Anguita D., Ghelardoni L., Ghio A., Oneto L. and Ridella S., (2012) The ‘K’ in K-fold Cross Validation, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges (Belgium), pp.25-27.

[10] Kumara A., Kuppusamya K. S., Aghilab G., (2019) A learning model to detect maliciousness of portable executable using integrated feature set, Journal of King Saud University - Computer and Information Sciences, vol.31, iss.2, pp.252-265.

[11] Roseline S. A., S. Geetha S., Kadry S., Yunyoung N., (2020) Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm, IEEE Access, vol.8, pp.206303-206324.

AUTHORS

Tuan Nguyen Kim (First Author) was born in 1969, and received B.E, and M.E from Hue University of Sciences in 1994, and from Hanoi University of Technology in 1998. He has been a lecturer at Hue University since 1996. From 2011 to the present (2021) he is a lecturer at the School of Computer Science, Duy Tan University, Da Nang, Vietnam. His main research interests include Computer Network Technology and Information Security.

Ha Nguyen Hoang (Corresponding Author) was born in 1976 in Vietnam. Working in the Faculty of Information Technology, Hue University of Sciences. 1995-1999 Bachelor of Science (B.S) in Science (majoring in COMPUTER SCIENCE), Hue University of Sciences. 2003-2005 Master of Science (M.S) in Computer Science, Hue University of Sciences.2012-2017 Doctor of Science (D.S) in Computer Science, Hue University of Sciences. His main research interests include Cloud Computing, parallel processing, and distributed processing.

Nguyen Tran Truong Thien was born in 1997, and received B.E from Duy Tan University in 2020. He has been a security researcher at Duy Tan University since February 2021. His main research interests include is Network Security, Information Security, and Machine learning for Cybersecurity.