A Real-Time PE-Malware Detection System Based on CHI-Square Test and PE-File Features

Mohamed Belaoued(✉) and Smaine Mazouzi

Department of Computer Science
Université 20 août 1955-Skikda, Algeria
{m.belaoued,s.mazouzi}@univ-skikda.dz

Abstract. Constructing an efficient malware detection system requires taking into consideration two important aspects, which are the accuracy and the detection time. However, finding an appropriate balance between these two characteristics remains at this time a very challenging problem. In this paper, we present a real-time PE (Portable Executable) malware detection system, which is based on the analysis of the information stored in the PE-Optional Header fields (PEF). Our system used a combination of the Chi-square (KHI²) score and the Phi (φ) coefficient as feature selection method. We have evaluated our system using Rotation Forest classifier implemented in WEKA and we reached more than 97% of accuracy. Our system is able to categorize a file in 0.077 seconds, which makes it adequate for real-time detection of malware.

Keywords: Malware · Malware analysis · Chi-square test (KHI²) · PE-optional header

1 Introduction

Malware, abbreviation for 'malicious software', is a term used to designate any computer program that is designed to accomplish unauthorized actions without the user's consent. The number of new discovered malware has grown steadily over the past ten years. Therefore, it is crucial to have an efficient protection against this kind of malicious programs. The existing anti-malware techniques can be broadly classified in three classes, which are signature-based, behavioral-based and heuristic-based techniques [1]. Signature-based techniques are widely used by most of commercial antivirus software (AV). These techniques are very accurate for detecting known malware that exist in the signatures' database [1]. However, they are not able to deal with unknown malware or newly launched ones, often developed after discovering a zero-day exploit [2]. Even if the recent AV have become more accurate, they are still very slow to take countermeasures when a new threat is discovered [3, 4].

The behavioral analysis also known as dynamic analysis consists of monitoring the execution of the analyzed program in an isolated environment (i.e. Sandbox or virtual machine) [5]. During the monitoring process, the actions that the program accomplishes (such as API calls, Systems calls, network traffic, etc.) are recorded and used

A. Amine et al. (Eds.): CIIA 2015, IFIP AICT 456, pp. 416–425, 2015.
DOI: 10.1007/978-3-319-19578-0_34

to generate behavior features for categorizing the program (malware or benign). Such techniques are very accurate and they are able to detect unknown malware [5]. However, their main drawback is that the monitoring process is run for a couple of minutes at most, therefore it can't observe the entire capabilities of the program [4]. Moreover, the time required for the monitoring process makes such techniques not suitable for real-time detection.

The heuristic-based analyses investigate different file features such as Opcode instructions, structural information (Such as header information), and API (Application Programming interface) calls [1],[5]. These sets of information are used as features for the classification process, which is generally done using machine learning-based classifiers such as decision trees and Bayes Algorithm [1],[6]. The Heuristic based Anti-malware systems are very accurate and are able to deal with unknown malware [1],[5, 6]. They are also easy to implement compared to the behavioral ones. However, the existing systems suffer from the inconvenient of their high processing overhead, since most of them use a large number of features, which yields to intensive computations. Due to that, most of the existing heuristic techniques are inadequate for real-time detection, which is a very suitable characteristic especially in such sensitive systems.

In this paper, we introduce a real-time PE (Portable Executable, See section 2) malware detection system, which consists of three different components, which are the PE-parser, feature selection module and a decision module. The PE parser was developed using Python language, and it statically (i.e. without executing the analyzed program) extracts the information contained in the PE-Optional header fields (PEF, see Section 2). PE header information (including Optional header ones) are very quick to extract, which is convenient for our real-time purposes. For the same purposes, our analysis was restricted on the Optional header only. We believe that using other types of features such as File-header fields or other structural information will considerably increase the number of features, which will have a direct impact on the detection time. The feature selection module was also developed using Python, and it is based on the KHI² test, which is a statistical method used for hypothesis testing [7]. The decision module is based on Rotation Forest classifier [8] that is available in Waikato Environment for Knowledge Analysis (WEKA) [9].

This paper is organized as fellows: Section 2 introduces the PE file format in order to facilitate the comprehension of the rest of the sections. Section 3 is devoted to most known related works, published in the literature. In section 4, we present our proposed system's architecture. In section 5, we present our experimental results. Section 6 concludes our work and underlines its perspectives.

2 PE File Format

PE is an abbreviation for Portable Executable[10], and it represents the common file format for binary executables and DLLs under Windows operating systems. A PE file is structured in layers and it is mainly composed of a DOS Header, PE Header, Section Headers (Section table), and a number of sections, as shown in "figure 1".

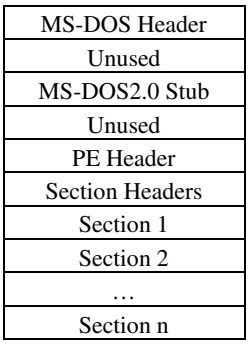

Fig. 1. PE file format

- The DOS Header is used if the file is run from the DOS. So it can then check whether it is a valid executable or not.
- The PE header is an IMAGE_NT_HEADERS data structure, which contains three members: PE-Signature, File Header, and the Optional Header. This latter is the subject of our work and is composed of several fields [10] as illustrated in figure 2. The values of the latter fields will be used as discriminators for the benign-malware categorization process.

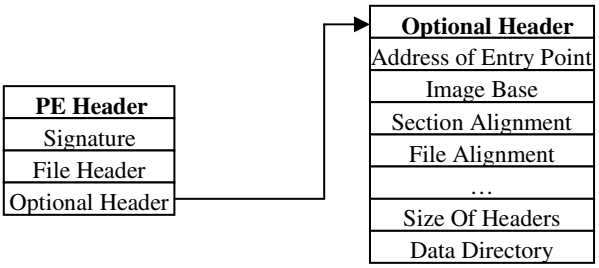

Fig. 2. Members of the PE-Optional Header

3 Related Work

In the last decade, security researchers have introduced new malware detection methods, in order to overcome the limitations of the standard signature-based ones. Schultz et al. [11] were the first authors to introduce a machine learning based malware detection system. The proposed system is based on the analysis of different information contained in the PE file such as strings and API calls. They used a classification method based on Naïve Bayes, and they achieved 97.11% of accuracy.

The method presented in [12] is based on API calls and Naïve Bayes classifier. The extracted APIs were used to construct models of suspicious behaviors, by grouping some APIs according to scenarios that a malware can accomplish, such as obtaining the system's directory, writing malicious data into files, and registry updates. They achieved an overall accuracy of 93.7%.

Ye et al. [2] have introduced a malware detection system that is based on the analysis of the set of APIs called by PE programs. The authors proposed a feature selection method based on the KHI² test. They used an Object Oriented Association (OOA) mining based classification method. Their system achieved an overall accuracy of 67.5% and a detection time of 0.09s.

The system proposed by Salehi et al. [4] is based on analyzing API calls and their arguments. They trained their system using different classifiers and they have obtained an overall accuracy of 98.1%. Extracting APIs arguments requires executing the program; therefore, this method has the inconvenient of dynamic approaches mentioned previously.

4 Proposed Method

Our proposed malware detection system categorizes a file in three different phases, which are the feature extraction, the feature selection, and the decision (classification).

4.1 Feature Extraction

As mentioned previously, our system relies on the analysis of the PE Optional Header fields (PEFs) and in order to extract these features from the analyzed file we developed a module written in Python by using a third party Python module called pefile [13]. PEFs are generated by concatenating the field's name and value (ex. CheckSum0 designates that the feature CheckSum has a value equal to 0).

4.2 Feature Selection

In order to reduce the number of obtained PEFs and keep only the most relevant ones, we developed a feature selection method, which is based on the chi-square (KHI²) test. The KHI² is a statistical method, which is used to determine whether there is a significant association between two qualitative variables. This association is expressed by the distance $\boldsymbol{D}$ between an observed frequency $\boldsymbol{O}$ and an expected one $\boldsymbol{E}$ (which represents the case of independence between the variables) and the greater is that distance stronger is the correlation between the variables. In our case, we will study that association between the variable 'PEF' that has two modalities: "present" and "absent". This variable represents the presence or not of a specific PEF in a PE file. The second variable is "PE" that has also two modalities: "Malware" and "Benign" that corresponds to the two categories of PE files that we used.

The first step to do when conducting a KHI² test, is to define the two hypotheses $\mathbf{H_0}$ and $\mathbf{H_1}$ that one will be accepted, and the other rejected. $\mathbf{H_0}$ and $\mathbf{H_1}$ represent respectively the case of independency and the case of dependency between the two variables. Note that accepting $\mathbf{H_0}$ for a PEF means that it is not specific to any category of PE-files. Therefore, it will be considered as irrelevant and will be removed. In our case, $\mathbf{H_0}$ and $\mathbf{H_1}$ are defined as follows:

- **H_0**: The presence or absence of a PEF is independent of the PE file's type (malware or benign).
- **H_1**: The presence or absence of PEF is related to the PE file's type (malware or benign).

For every PEF, we have a contingency table as shown in table 1.

Table 1. Contingency Table of a PEF.

	PEF: Present	**PEF: Absent**	**Row Total**
PE: Malware	N1	N2	N
PE: Benign	M1	M2	M
Column total	N1+M1	N2+M2	T

N, M, and ***T*** are respectively the total number of malware PE, the total number of benign PE, and the total number of all PE files ($T=N+M$). ***N1*** and ***N2*** are respectively the number of malware PE that have a **PEF** and the number of malware PE that do not have the **PEF**, such as ***N = N1 +N2***. ***M1*** and ***M2*** are respectively the number of benign PE that have a **PEF** and the number of benign PE that do not have the **PEF**, such as ***M = M1 + M2***. The KHI² score (D^2) is calculated using the formula (1):

$$D^2 = \sum \frac{(O_{r,c} - E_{r,c})^2}{E_{r,c}} \tag{1}$$

Where $O_{r,c}$ is the observed frequency count at level r of row variable and level c of column variable. And $E_{r,c}$ is the expected frequency. $E_{r,c}$ is defined by equation (2) .

$$E_{r,c} = \frac{n_r \times n_c}{T} \tag{2}$$

Where n_r, and n_c represent respectively the sum on row r and the sum on column c. After calculating the KHI² values for the obtained PEFs, we have to determine which of two hypotheses are accepted or rejected for every PEF. To do that, we have to compare the obtained KHI² scores of every PEF to a threshold, which represents the theoretical KHI² value (χ^2). That value is obtained by first calculating the degree of freedom (**DF**), and choosing a signification level α that represents the error probability when accepting or rejecting an hypothesis. Considering **DF** and α, the χ^2value is obtained from the KHI² distribution table [14] . **DF** is calculated using the following equation:

$$DF = (R-1) \times (C-1) \tag{3}$$

Where R, and C are respectively the number of modalities of the first and the second variables. After rejecting all the PEFs that are not correlated ($D^2 \leq \chi^2$), we will calculate the φ coefficient using the formula (4) for the remaining ones. The φ coefficient is a normalization of the KHI² score (D^2), which is used to measure the strength of the dependency between the two variables [15]. In our work, that coefficient will be used to generate the different PEFs' subsets, which are grouped according to their correlation's strength (relevance).

$$\varphi = \sqrt{\frac{D^2}{T}} \tag{4}$$

The value of φ ranges between 0 and 1, therefore, the strength of the relationship can be divided in 4 different classes:

- $\varphi \approx 0.25$: Weak correlation.
- $\varphi \approx 0.50$: Medium correlation.
- $\varphi \approx 0.75$: Strong correlation.
- $\varphi \approx 1$: Very strong correlation.

Our obtained PEFs will be divided into non-disjoints subsets according to the φ values mentioned previously.

4.3 Classification

In order to evaluate our malware detection system we have used Rotation Forest classifier [8] that is implemented in WEKA [9]. Therefore, our classification module takes as an input the PEFs subsets represented as an .arff file. The .arff file is the data file format supported by WEKA, and it is automatically generated using a python script. The classifier is then trained and models are generated for each feature subset of the training set. The obtained models are then tested on previously unseen PE-files contained in our test set.

5 Experimentation

5.1 Dataset

We collected a dataset composed of 552 PE files (338 malware and 214 benign programs). This dataset will be split into 80% training set and 20% test set. The infected PE dataset was downloaded from Vxheavens.com and contains 12 different malware categories as shown in table 2.

Table 2. Used malware dataset

N°	Malware Type	Counts	N°	Malware Type	Counts
1	Backdoor	27	7	Trojan	59
2	Email-Worm	19	8	Trojan-Downloader	24
3	Exploit	28	9	Trojan-Dropper	32
4	Hacktool	22	10	Trojan-Spy	18
5	Net-Worm	16	11	Virus	42
6	P2P-Worm	17	12	Worm	34
TOTAL = 338					

The benign PE files include some utility software downloaded from Softpedia.com and some Windows system files collected from a clean installation of windows XP. We scanned the whole dataset by more than 40 AV available on the website Virus-Total.com, in order to make sure that they are correctly labeled (malware, benign).

5.2 Results and Evaluation

In this subsection, we will present the obtained experimental results from the feature extraction phase until the decision phase. We first start by the obtained PEFs after the feature extraction phase. As presented in Table 3, we have obtained 590 PEFs with their corresponding frequencies in malware and benign PE (observed frequencies).

Table 3. Overview of the obtained PEFs list and their corresponding frequencies

			Frequency	
N°	**Optional Header field**	**Value**	**Malware (271)**	**Benign(172)**
1	BaseOfCode	4096	271 (100%)	172 (100%)
2	BaseOfData	102400	4 (1%)	1 (1%)
…	…	…	…	…
86	CheckSum	0	259 (96%)	5 (3%)
87	CheckSum	102910	1 (1%)	0 (0%)
…	…	…	…	…
589	Subsystem	2	226 (83%)	106 (62%)
590	Subsystem	3	45 (17%)	66 (38%)

We will calculate the KHI^2 and φ values (as presented in the subsection 4.2) for the obtained PEFs and remove the non-relevant ones that have $KHI^2 < 3.84$ (3.84 is the X^2 value for DF=1 and alpha =0.05). The obtained results are presented in table 4.

Table 4. KHI^2 scores and φ values of the selected PEFs

N°	**PEF**	***KHI²***	***φ***
1	CheckSum0	375.21	0.92
2	MajorImageVersion0	370.57	0.91
3	DllCharacteristics0	355.91	0.9
4	MajorOperatingSystemVersion5	346.02	0.88
5	MinorOperatingSystemVersion0	341.92	0.88
…	…	…	…
50	SizeOfInitializedData28672	3.86	0.09

As presented in Table 4, we have obtained a final list of 50 PEFs with their corresponding KHI^2 scores and φ values. We will divide these features into different groups (subsets) according to their φ values. At the end of the feature selection phase, we have obtained three different subsets: G1, G2, and G3 that contain PEFs that have

respectively $\varphi \geq 0.75$, $\varphi \geq 0.5$, and $\varphi \geq 0.25$. We have respectively 11, 14, and 22 PEFs in G1, G2, and G3. We have used a forth subset G4 that contains the complete 590 extracted PEFs, the aim from that is to see whether our feature selection method have improved the obtained results or not.

Next, we will evaluate our system's performance by training the Rotation Forest classifier using different features subsets and see which subset will generate the best results. The performance of a classifier is generally evaluated by calculating three different metrics which are Detection rate (DR), False Alarm rate (FA), and Accuracy (AC) and they are calculated using the equations 5, 6 and 7 respectively:

$$DR = \frac{TP}{TP + FN} \times 100\% \tag{5}$$

Where TP (true positive) and FN (false negative) represent respectively malware that were correctly classified as malware and malware that were wrongly classified as benign.

$$FA = \frac{FP}{FP + TN} \times 100\% \tag{6}$$

TN (true negative) and FP (false positive) represent respectively benign programs that were correctly classified as benign, benign program that were wrongly classified as malware. The accuracy (AC) represents the rate of files that were correctly classified in their class.

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{7}$$

The fourth metric that we will use to evaluate our system's performance is the detection time (DT), which represents the average time required for categorizing a file and it is expressed in seconds per file. DT includes the feature extraction time, .arff file generation time, and the classification time. The obtained results are presented in table 5.

Table 5. Experimental results

Group	φ	PEF Counts	DR	*FA*	AC	DT
G1	≥0.75	11	98.51%	7.14%	96.33%	0.075
G2	**≥0.50**	**14**	**100.00%**	**7.14%**	**97.25%**	**0.077**
G3	≥0.25	22	98.51%	9.52%	95.41%	0.079
G4	-	590	97.01%	7.14%	95.41%	0.116

From the results presented in the above table, we can see that our proposed feature selection method was able to increase the accuracy of our system by +1.84% (from 95.41% with G4 to 97.25% with G2) and that using only 14 PEFs. It was also able to reduce the categorization time by 33% (from 0.116s with G4 to 0.077s with G2). Note that the feature extraction phase took 0.037s, the .arff file generation also required 0.037s, and the classification phase took 0.003s.

5.3 Comparison

In this subsection, we will evaluate our system's performance by comparing it with the previously cited methods. The results are presented in table 6.

Table 6. Results of the comparison with the previously cited methods for malware detection

Method	Feature Type	DR	AC
Our method	**PEFs**	**100%**	**97.25%**
Schultz et al. [11]	Strings	97.43%	97.11%
Salehi et al. [4]	APIs+Args	99.2%	98.4%
Wang et al. [12]	APIs	94.4%	93.71%
Ye et al. [2]	APIs	88.16%	67.5%

From the results presented in Table 6, we can see that our system outperforms three of the four presented systems with an improvement in accuracy that varies from 0.14 % to 30%. The system proposed by Salehi et al. [4] is more accurate than our system (+1.15%). However, our system has a better detection rate.

If we consider the detection time (categorization time), we can conclude that our system is adequate for real-time detection. The proposed system is able to categorize a file in 0.077s, which is a very satisfying performance, compared with the system proposed by Ye et al. [2] which categorizes a file in 0.09s. The system proposed by Salehi et al. [4] needs to monitor the analyzed program during 2 minutes in order to extract API calls and their arguments, that represents almost 3000 times the required time by our proposed features extraction method.

6 Conclusion and Future Works

In this paper, we have presented a real-time PE-malware detection system that is based on the analysis of the PE-optional Header information. The proposed system uses an efficient feature selection method, which is based on the KHI² test. This latter allowed us to achieve a high accuracy and a low detection time, using only 2% of the initially extracted features. As future works, we project to combine different types of features such as APIs calls, and Opcode, in order to increase the accuracy of our system.

References

1. Bazrafshan, Z., Hashemi, H., Fard, S.M.H., Hamzeh, A.: A survey on heuristic malware detection techniques. In: Proceedings 2013 5th Conference on Information and Knowledge Technology (IKT), Shiraz, pp. 113–120 (2013)
2. Ye, Y., Li, T., Jiang, Q., Wang, Y.: CIMDS: Adapting postprocessing techniques of associative classification for malware detection. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 40, 298–307 (2010)
3. June, I.: Anti-malware vendors slow to respond. Computer Fraud & Security, 1–2 (2010)

4. Salehi, Z., Sami, A., Ghiasi, M.: Using feature generation from API calls for malware detection. Computer Fraud & Security Bulletin, 9–18 (2014)
5. Aycock, J.D.: Computer viruses and malware. Springer, Heidelberg (2006)
6. Shabtai, A., Moskovitch, R., Elovici, Y., Glezer, C.: Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. Information Security Technical Report 14, 16–29 (2009)
7. Fornasini, P.: The Chi Square test. The Uncertainty in Physical Measurements:An Introduction to Data Analysis in the Physics Laboratory, pp. 187–198. Springer Science & Business Media (2009)
8. Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A New classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1619–1630 (2006)
9. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
10. Pietrek, M.: Peering Inside the PE: A Tour of the Win32 Portable Executable File Format. Microsoft Systems Journal-US Edition 9, 15–38 (1994)
11. Schultz, M.G., Eskin, E., Zadok, E., Stolfo, S.J.: Data mining methods for detection of new malicious executables. Proceedings. In: 2001 IEEE Symposium on Security and Privacy, S&P 2001, Oakland, CA, pp. 38–49 (2001)
12. Wang, C., Pang, J., Zhao, R., Liu, X.: Using API sequence and bayes algorithm to detect suspicious behavior. In: Proceedings of the 2009 International Conference on Communication Software and Networks, ICCSN 2009, Macau, pp. 544–548 (2009)
13. `https://code.google.com/p/pefile/`
14. Koskska, S., Nevison, C.: Statistical tables and formulae. Springer, New York (1989)
15. Farrington, D.P., Loeber, R.: Relative improvement over chance (RIOC) and phi as measures of predictive efficiency and strength of association in 2x2 tables. Journal of Quantitative Criminology 5, 201–213 (1989)