International Journal of Cyber-Security and Digital Forensics (IJCSDF) 7(3): 238-242
The Society of Digital Information and Wireless Communications (SDIWC), 2018 ISSN: 2305-001

Data Mining Classification Approaches for Malicious Executable File Detection

Hassan Najadat, *Assem Alhawari, **Huthifh Al_Rushdan
Jordan University of Science and Technology
Irbid, Jordan
najadat@just.edu.jo, {*aralhawari15,**aahuthifh14}@cit.just.edu.jo

***Abstract*—Classification technique has many applications in executable files detection. The classification method is one of the most popular data mining technique to detect and predict the behavior of executable files. Every executable file has hexadecimal sequence features, which represent the assembly strings sequences of the executable file, and Portable Executable (PE) features like DLL (Dynamic Link Library). These features reflect the behavior and the characteristics of executable files. Classification according to these behavioral patterns is an efficient way to distinguish the malicious executable file from the normal one. We present an extraction code to get specific features of hexadecimal code among thousands of hex codes and DLL. Since there are huge number of extracted features, three different ranker are utilized to select the most worth features. In this paper, among eight classifiers, Neural Networks and KNN achieves a highest accuracy**

Keywords— Classification; Malicious Files; Support Vector Machine; Naïve Bayes; Decisoin Tree, KNN

I. INTRODUCTION

Many applications and software become essential in different fields such as economics, health, social media, markets, and industries. Malware software invades the applications and the internet web pages to control and gather information from computers and mobile operations. Malwares become the most important threat in computers' security. Many techniques are used to protect devices, systems and networking from malware whose goal either to destroy hard disk to eliminate the data or to attack the operating systems like malicious executable files.

The malicious executable files are discovered and analyzed either based on static analysis or dynamic analysis [1]. In the static analysis, a set of patterns from specific features are extracted and used to match the malicious executable files with pre define database. These features include string signature, byte-sequence and syntactic library call. The dynamic analysis uses file execution in special environment to monitor file behavior of the malicious files. Another way to detect malicious files is to utilize data mining techniques to identify the suspicious executable files [2].

This paper provides a compressive work of using eight classifiers to detect malicious files. Different ranker methods are applied to select most worth features that contribute to the detection unwanted files. The rest of the paper is organized as follows; section 2 provides a background and related works in the malware detection and data mining techniques, while section 3 describes the methodology of the features selection and extraction. Experimental results and discussion are presented in section 4. Finally, the conclusion is provided.

II. RELATED WORK

Many data mining approaches including text mining, decision tree, Support Vector Machine (SVM), Naive Bayes (NB), and random forest used to detect malware software. The performance of these approaches depends on quality of dataset features extracted from the target files. The extracted features from the executable files include string code, DLL files, system calls and instructions that used in CPU. Different studies proposed machine-learning techniques for Detecting and Classifying Malware in respect of static and dynamic analysis [1]

The classification technique based on text mining was used by treating CPU instructions as natural language to detect the malicious malware and determine the type of malware. In [3], the researchers utilized Naive Bayes classifier and simplified the attributes according to the correlation between attributes and decisive attributes. Another way to extract features is n-gram-based analysis, which extracts the effective features for classification process [4].

Extracting features from an executable file is a major factor for detection and prediction accuracy, for features analysis some researcher used dynamic analysis, which studied the behavior of executable file that reflects all instructions and system calls in the file. Various classification methods were used to discover malware behavior and dynamic analysis by using VB.Net language to build a suitable dataset for WEKA tool [2].

A combined analytical model was proposed by integrating advantages of both static and dynamic analysis to get better classification accuracy results. Static analysis yielded a high indicator of file behavior without the need for executing the file, but it needs statistical analysis [5].

A detection of malicious executables was studied by proposing a low-complexity host-based technique. Information theoretic divergence measures were utilized to quantify benign and malware executables. Benign models of divergent attributes were applied in cross-correlation and log-likelihood frameworks to classify malicious executables [6]. Multilayer

Perceptron with several learning algorithms were provided based on statistical features of executable file [7]. SVM, Logistic Regression, and Random forest were used as classification approaches to detect malicious files [8].

III. METHODOLOGY

Our methodology has three phases, 1) data set collection and extraction from the data a set of features to build training dataset, 2) reduction the dataset features to get the effective attributes that contribute to a best detection classification, and 3) applying different classifiers to specify the best classifiers that yield a high accuracy. Fig. 1 shows our methodology.

A. Dataset Collection and Features Extraction

1) Dataset Collection

There are few resources to collect malwares and they are dangerous to deal with. We collect our samples from different websites such as http://vxheaven.org, http://leetupload.com/dbindex2, http://dasmalwerk.eu, and http://www.malwareblacklist.com, http://thezoo.morirt.com.

Our data set consists of 14998 executable files divided into 12593 malicious executable files and 2405 normal executable files. In this paper, the dataset represent the malware of win32, which invades the system as executable file and uses the system calls and DLL files from windows system for specific unwanted reasons.

2) Features Selection

We use the hexadecimal features that reflect the behavior of the malware and some DLL files that are used by the malware. We get the same hexadecimal features in [8], which are 503 features, each one is 4 bytes (2 digits hexadecimal number for each byte) and we add 39 DLL features to build the attributes of our dataset. Table I shows a sample of the features. The hexadecimal features are represented from one to 503 while the DLL features are 504 to 543.

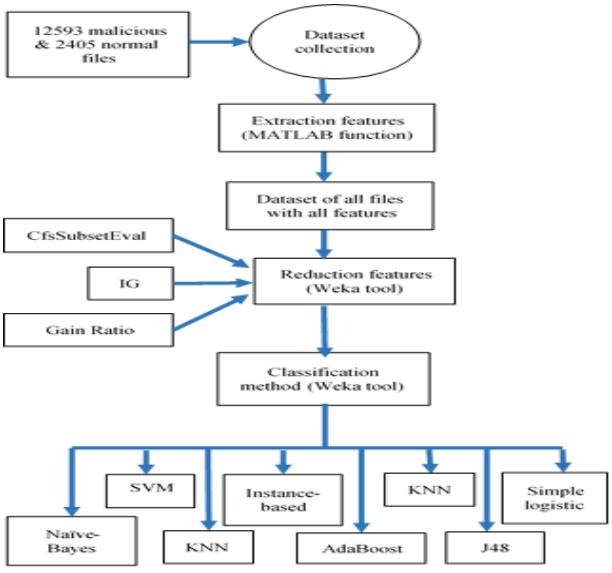

Fig. 1. Methodology Phases

3) Features Extraction

Since there are thousands of hexadecimal sequence in each exe file, we build a Matlab code that searches every file to extract 543 features as shown in Fig. 2. The result of the above code is a dataset represented as a binary vector for each file with setting one for existing feature and zero for missing feature. The class attribute is either normal or malicious. Fig. 3 represents the flowchart of features extraction.

B. Dataset Reduction

Selection of the best attributes is introduced in this paper in order to remove irrelevant features using three different metrics including Information Gain (IG), Gain Ratio and CfsSubsetEval. We utilized Weka tool in this step by setting a cross-validation test to 10 to rank the attributes from the best one to the worst according to classification effectiveness.

CfsSubsetEval is a function in Weka to evaluate the worth of a subset of attributes. Top 50 attributes are selected because these attributes represent the highest gain information that contribute in classifying the files into normal and malicious.

C. Classification process and experiment environment

We applied eight type of classification algorithms using Weka tool: NaiveBayes, SVM, k-Nearest Neighbors (KNN), instance-based method, AdaBoost, Neural network, decision tree (J48) and simple logistic approach. The testing mode is set to 66% of the dataset as a training set and the remaining as a testing set.

These algorithms adapt different approaches for classification. Some these classifiers are lazy which store training data and wait until it is given a test tuple, usually less time in training but more time in predicting such as instance-based and kNN.

TABLE I. A SAMPLE OF FEATURES

Attribute 500	Attribute 502	Attribute 503	Attribute 504	Attribute 506
61007400	FFFFFF7F	30	USER32.DLL	kernel32.dll

```
1. Start
2. Get the features exe file
3. Repeat for each exe file
   3.1 Open exe file and read all hexadecimal
       sequences and DLL systemfile
   3.2. Repeat for every feature
       Find the feature in exe file
         If feature exists then
             Set the value of the attribute 1
         Else set the value of the attribute 0
       End loop
   3.3. Create a binary vector for the exe file
   3.4. Save the binary vector in matrix objects
4. End of for loop
```

Fig. 2. Matlab Code for File Features Extraction

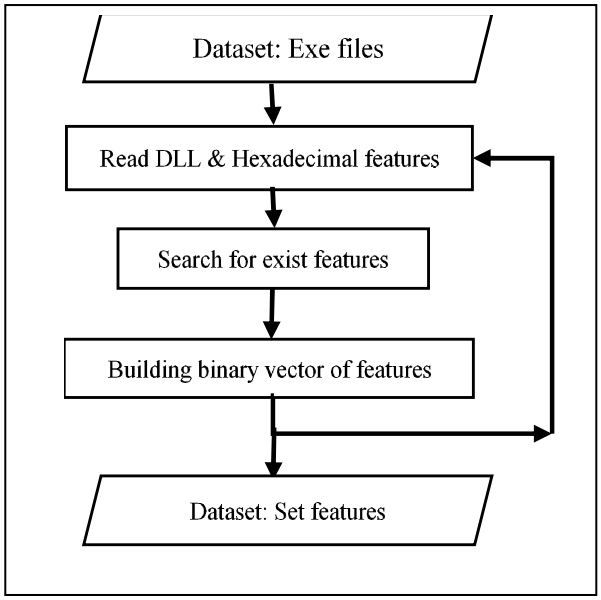

Fig. 3. Flowchart of features extraction

Neural network uses multi layers of inputs and outputs and updates the weights of each inputs until it reaches the optimal nonlinear separating function for classification. Decision tree builds classification model by building a tree structure. It breaks down a dataset into smaller and smaller subsets until it can predict the value of a target variable based on several input variables.

IV. Experimental Results and Discussion

A. *CfsSubsetEval selected attributes technique*

Table II shows the result of the eight classifiers. The first column represent the classifier name, the second column represents true positive recognition rate for malicious and normal exe files, TP, which is called sensitivity. M represents malicious while N represents normal. We use Root Mean Squared Error (RMSE) in the third column. Each classifier has different time for building model (TBM) and time for testing (TTM).

Fig. 4 shows the accuracy of each classifier. The problem was some classifier takes a long time to build the model like neural network; on the other hand, some classifier takes a long time with evaluating the test set on the model like Kstar and KNN.

Neural Network yields the highest accuracy with 103.11 seconds. KNN has close accuracy to neural network but it takes 128.89 seconds for every test. According to the time consuming, simple logistic achieves a good accuracy comparing with other classifiers based on 9.86 seconds.

B. *Gain Ratio selected attributes technique*

Gain Ratio is a method to rank the attributes from the best one to the worst. Table III shows the result of the eight classifiers with Gain Ratio, first fifty attributes were chosen.

TABLE II. APPLYING CLASSIFIERS BASED ON CFSSUBSETEVAL ALGORITHM

classifier	TP		RMSE	ACC%	TBM	TTM	Time
	M	*N*					
NB	0.945	0.824	0.247	92.6	0.41	0.41	0.82
SVM	0.948	0.83	0.265	92.96	11.86	0.03	11.89
KNN	0.96	0.805	0.219	93.57	0.03	128.89	128.9
KStar	0.968	0.64	0.235	91.76	0.01	726.64	726.7
AdaBoost	0.948	0.657	0.248	90.25	3.01	0.08	3.09
Neural network	0.959	0.81	0.217	93.63	103.11	0.17	103.3
J48	0.956	0.804	0.23	93.23	2.57	0.06	2.63
Simple logistic	0.959	0.804	0.2237	93.53	9.83	0.03	9.86

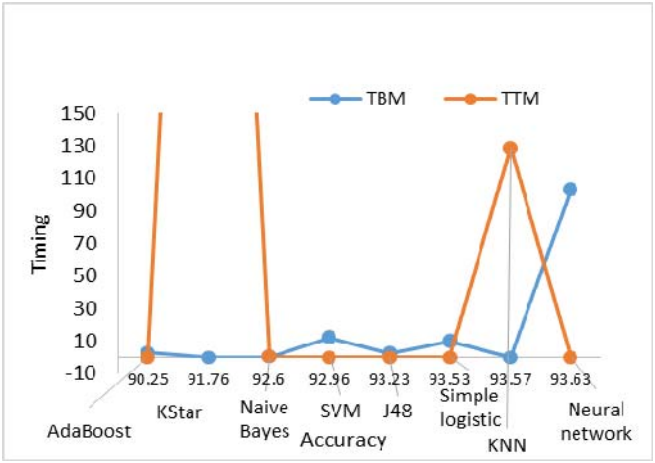

Fig. 4. Accuracy of classifiers based on CfsSubsetEval algorithm

TABLE III. APPLYING CLASSIFIERS BASED ON GAIN RATIO ALGORITHM

Classifier	TP		RMSE	ACC%	TBM	TTM	Time
	M	N					
NB	0.91	0.7	0.344	87.3	1.22	0.14	1.36
SVM	0.97	0.74	0.256	93.4	566	0.08	566
KNN	0.99	0.82	0.154	96.4	0.03	97	97
KStar	0.99	0.8	0.174	96.2	0	1125	1125
AdaBoost	0.94	0.66	0.255	89.4	5.13	0.03	5.16
Neural Network	0.99	0.83	0.175	96.4	509	0.21	510
J48	0.99	0.81	0.186	96	12.6	0.13	12.8
Simple logistic	0.98	0.75	0.22	94.4	49	0.05	49.1

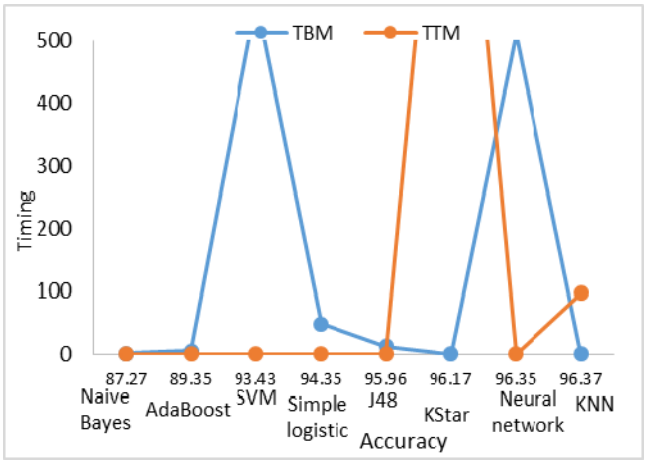

Fig. 5. Accuracy of classifiers based on Gain Ratio algorithm

The accuracy is between 87.27 and 96.37. The classifiers, which take a long time to build the model, are neural network and SVM. On the other hand, some classifier takes a long time with evaluating the test set on the model like KStar. Fig. 5 shows classifiers time in respect of accuracy rate.

According to Fig. 5, J48 achieves a good accuracy comparing with other classifiers and it takes 12.77 seconds to build and test the model. However, if we want to get high accuracy and low time consuming, KNN is the best in this experiment.

C. Information Gain (IG) selected attributes technique

IG is a measurement function to rank the attributes from the best one to the worst with specific threshold. Table IV shows the result of the eight classifiers based on the top of 50 attributes.

The accuracy is between 84.91 and 96.96.9. The classifiers, which take a long time to build the model, are neural network, SVM and simple logistic On the other hand, some classifier takes a long time with evaluating the test set on the model like KStar. Fig. 5 shows classifiers time in respect of accuracy rate.

If we want to decide, which model is the best according to Fig. 5, J48 has good accuracy comparing with other classifiers and it takes 6.48 seconds to build and test the model. However, if we want to get high accuracy and low time consuming, KNN is the best in this experiment, it takes 30.51 seconds to test and build the model.

Table V shows the accuracy of each classifier in the three selected measurements, Gain Ratio, IG, and CfsSubsetEval. NaiveBayes and SVM achieve the best sensitivity in the CfsSubsetEval algorithm. Generally, NN, KNN, and Kstar gain the highest sensitivity. However, NN takes a long time to build the model while Kstar takes a time to test the model. Simple logistic gives in CfsSubsetEval higher sensitivity than Gain Ratio, but the accuracy of KNN in Gain Ratio achieves the highest accuracy

TABLE IV. APPLYING CLASSIFIERS IN RESPECT OF INFORMATION GAIN ALGORITHM

Type	TP		RMSE	ACC %	TBM	TTM	Time
	M	*N*					
NB	0.877	0.696	0.378	84.91	0.94	0.11	1.05
SVM	0.976	0.787	0.231	94.67	417.64	0.04	417
KNN	0.985	0.868	0.17	96.67	0.03	30.48	30.51
KStar	0.986	0.871	0.157	96.82	0.01	1209	1209
AdaBo	0.944	0.834	0.24	92.7	3.96	0.04	4
NN	0.985	0.883	0.165	96.9	741.07	0.18	741.25
J48	0.986	0.858	0.1757	96.6	6.45	0.03	6.48
Simple logistic	0.979	0.82	0.198	95.45	121.1	0.18	121.28

V. CONCLUSION

In this paper, we present a methodology for extracting executable file features and apply eight classifiers using Weka tool, and we provide a comparison between these classification algorithms. Choosing the best classifier depends on size of the dataset and reduction algorithm. Neural Network achieves the highest accuracy based on IG ranker. When Gain Ratio is used as a selection attribute ranker, KNN gains the highest accuracy.

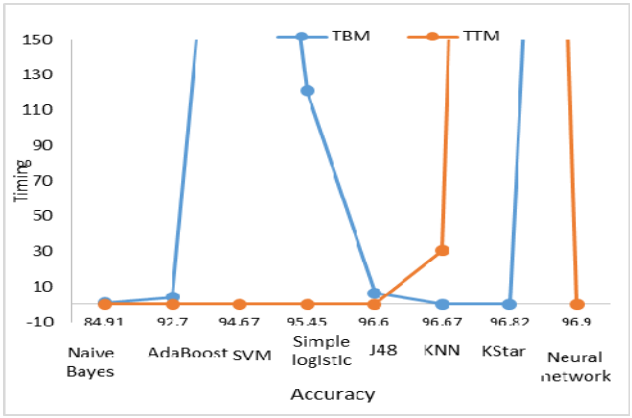

Fig. 6. Timing and accuracy of classifiers with Information Gain algorithm

TABLE V. CLASSIFIERS ACCURACY OF THREE SELECTED ATTRIBUTES ALGORITHMS

Classifier	CfsSubsetEval	Gain Ratio	IG
Naïve Bayes	92.6	87.27	84.91
SVM	92.96	93.43	94.67
KNN	93.57	96.37	96.67
KStar	91.76	96.17	96.82
AdaBoost	90.25	89.35	92.7
Neural network	93.63	96.35	96.9
J48	93.23	95.96	96.6
Simple logistic	93.53	94.35	95.45

REFERENCES

[1] E. Gandotra, D. Bansal and S. Sofat, "Malware Analysis and Classification: A Survey", *Journal of Information Security*, vol. 05, no. 02, pp. 56-64, 2014.

[2] M. Norouzi, A. Souri and M. Samad Zamini, "A Data Mining Classification Approach for Behavioral Malware Detection", *Journal of Computer Networks and Communications*, vol. 2016, pp. 1-9, 2016.

[3] P. Kierski, M.Okoniewski and P.Gawrysiak, "Automatic Classification of Executable Code for Computer Virus Detection", in Conference: Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM'03, Zakopane, Poland, 2003, pp. pp 277-284.

[4] T. Lee and J. Kwak, "Effective and Reliable Malware Group Classification for a Massive Malware Environment", *International Journal of Distributed Sensor Networks*, vol. 2016, pp. 1-6, 2016.

[5] O. Samantray, S. Narayan Tripathy, S. Kumar Das and B. Panda, "CAM: A Combined Analytical Model for Efficient Malware Classification", *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 1, 2016.

[6] H. Khan, F. Mirza and S. Khayam, "Determining malicious executable distinguishing attributes and low-complexity detection", *Journal in Computer Virology*, vol. 7, no. 2, pp. 95-105, 2010.

[7] L. Gonzalez and R. Vazquez, "Malware Classification using Euclidean Distance and Artificial Neural Networks", in *Artificial Intelligence (MICAI),12th Mexican International Conference on Artificial Intelligence*, Mexico City, Mexico, 2013.

[8] P. AnastaRumao, "Using Two Dimensional Hybrid Feature Dataset to Detect Malicious Executables", *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 7, 2016.

