Zero-Day Malware Detection

Ekta Gandotra
Department of Computer Science and Engineering
Chitkara University Institute of Engineering and Technology,
Chitkara University, India
ekta.gandotra@gmail.com

Divya Bansal
Department of Computer Science and Engineering
PEC University of Technology
Chandigarh, India
divya@pec.ac.in

Sanjeev Sofat
Department of Computer Science and Engineering
PEC University of Technology
Chandigarh, India
sanjeevsofat@pec.ac.in

***Abstract*—The increasing volume and variety of malware is posing a serious security threat to the Internet today and is one of the main apprehensions for the security community for the last few years. The traditional security systems like Intrusion Detection System/Intrusion Prevention System and Anti-Virus (AV) software are not able to detect unknown malware as they use signature based methods. In order to solve this issue, static and dynamic malware analysis is being used along with machine learning algorithms for malware detection and classification. The main problems with these systems is that they have high false positive and false negative rate and the process of building classification model takes time (due to large feature set) which hinders the early detection of malware. Thus, the challenge is to select a relevant set of features, so that, the classification model can be built in less time with high accuracy. In this paper, we present a system that addresses both the issues mentioned above. It uses an integration of both static and dynamic analysis features of malware binaries incorporated with machine learning process for detecting zero-day malware. The proposed model is tested and validated on a real-world corpus of malicious samples. The results show that the static and dynamic features considered together provide high accuracy for distinguishing malware binaries from clean ones and the relevant feature selection process can improve the model building time without compromising the accuracy of malware detection system.**

Keywords-malware detection; static malware analysis; dynamic malware analysis; feature selection; machine learning

I. INTRODUCTION

A software program that purposefully fulfils the harmful intent of an attacker is usually known as malicious software or malware [1]. These are constantly emerging in volume, variety and velocity. These are becoming more sophisticated, targeted and stealthy with time [2]. According to Internet Security Threat Report released by Symantec in 2016 [3], over 317 million new malware specimens were discovered in the year 2014, means about 1 million new malware threats released on daily basis.

978-1-5090-2541-1/16/$31.00 ©2016 IEEE

This number had increased to 430 million in the year 2015, about 36% from the previous year. Moreover, the existing anti-malware systems like IDS/IPS and AV software are not able to detect unknown malware as these are based on signature based methods. In order to deal with such issues, the researchers have developed various automated malware classification systems using machine learning approaches which are based on the features obtained from static and dynamic analysis of malware. Static analysis is the technique to analyze malware sample without executing it. It is performed by examining the malicious code and finding obvious external feature such as fingerprinting, header information, packer detection, strings and import functions etc. The drawback of this approach is that it is unable to detect obfuscated and polymorphic malware specimens. This problem is resolved by performing the dynamic analysis which includes executing it in a controlled environment and monitoring various run-time activities like registry changes, network activities, file system changes etc. Though the dynamic analysis is more effective as compared to static one but it is time exhaustive and resource consuming. The problem with dynamic analysis is that the malicious samples which are executed in the artificial environment may behave differently in order to fool the analysts or sometimes they may trigger their behavior only under certain conditions.

Due to pros and cons of both approaches, it is obvious that in order to improve the results of malware detection and classification, features from both static and dynamic analysis need to be integrated. Moreover, a large number of features if considered can take more time for building classification model thus hindering the early detection of malware. Thus a relevant set of features needs to be selected so that the classification model can be built in less time with high accuracy. Feature selection is a method of identifying top ranked features. It detects the relevant features thus making it easy to discard the irrelevant ones. A perfect selection of the features can improve the learning speed as well as generalization capacity of the model.

In this paper, we present a system for detecting zero-day malware, which makes use of relevant malware features extracted from both static and dynamic malware analysis along with the machine learning algorithms present in WEKA (Waikato Environment for Knowledge Analysis) library [4]. The experimental results obtained from classification models

are compared Before Feature Selection (BFS) and After Feature Selection (AFS) to show that feature selection process can improve the model building time without compromising the accuracy of malware detection system.

II. Related Work

This section provides the research work carried out in classifying malware binaries into their existing classes or identifying those that show different behavior and are the potential candidate for closer analysis. Various static and dynamic malware features along with machine learning algorithms are used for this purpose.

Schultz et al. [5] were the first to present the idea of data mining for malware detection. They use the static features such as strings, byte sequence and features from PE and used the Naïve Bayes method for classification purpose. Their results were upgraded by Kolter et al. [6] who make use of n-gram and various classification methods to detect and classify malware samples. Classification using Boosted decision tree provide the best accuracy.

Another classification system was designed by Kong et al. [7], who used the structure information of malware binaries for automated classification process. Tian et al. [8-9] used function length and its frequency [8] and printable string information [9] as the features for classifying malware specimens. In [10], authors used suspicious section count and frequency of import functions to discriminate malicious files from benign ones. Firdausi et al. [11] obtained the behavioral features from Anubis [12] and pre-processed these into sparse vector models for classification using machine learning algorithms available on WEKA. Nari et al. [13] used network behavior for malware classification using machine learning algorithms available in WEKA library. Lee et al. [14] used the behavioral profile created using dynamic malware features like registry, file and network activities.

Authors of [15], [16], [17] and [18] proposed the hybrid techniques which include features from both static and dynamic analysis of malware and provide more accurate results as compared to the systems using features from static analysis or dynamic analysis alone.

From the review of open literature, it is found that researchers have focused on improving the accuracy of malware detection system but there is a need to address the issue related to improving the building time of classification model while maintaining high accuracy.

This paper presents a system which uses an integration of both static and dynamic analysis features of malware binaries for detecting zero-day malware. It uses a filter approach for selecting relevant features for building classification models using machine learning algorithms available in WEKA library.

III. Proposed Malware Classification System

This section presents the proposed system for distinguishing malware programs from benign ones. Figure 1 illustrates the design flow of automated malware detection and classification system. The description of various steps involved in the system is as follows:

A. Data Acquisition

A large corpus of malicious samples targeting Windows OS is collected from Virusshare (https://virusshare.com) database which is available in the public domain after free registration. These are then scanned using AVG AV (http://www.avg.com) to endorse their maliciousness. The clean files used are collected manually from System directories of successive versions of Windows Operating system.

B. Automated Malware Analysis

All these specimens are then made to execute in an automated analysis environment set up at Cyber Security Research Centre, PEC University of Technology, Chandigarh, India using a modified version of Cuckoo sandbox (Brad Accuvant) [19-20]. The system is configured to generate the analysis reports in Java Script Object Notation (JSON) format [21] after executing a specimen in it.

C. Feature Extraction

The JSON reports generated by Cuckoo sandbox are then parsed to obtain the various malware features including both static and dynamic features. These include file size, packer used, section count, suspicious section count, commands executed, network activities, file system activities, registry activities, services created/started, processes created etc. The dataset so obtained contains very large number of features and is not suitable for building the classification model. This data is prepared to have a feature set of 18 malware attributes which can be used for building the classification models. These features are: packer_used, file_size, section_count, susp_sec_count, network_activity, dropped_count, filesread_count, filemod_count, filesdel_count, mutex_count, execomm_count, regread_count, regmod_count, regdel_count, importfunt_count, service_creat_count, service_started_count, processcount.

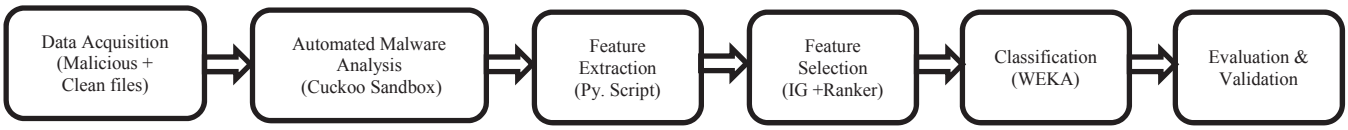

Fig. 1. Malware detection and classification system.

D. Feature Selection

Building a classification model from the training data is a time consuming task for a data set with high dimensionality. So, the top ranked features are selected from the set of 18 features using Information Gain (IG) method [22]. It is a method for feature evaluation which is broadly used in machine learning and is based on entropy (degree of randomness in the data).

$$Entropy = -\sum_{i} P_i \log_2 P_i$$

where P_i is the probability of the class.

IG is the amount of information gained from the value of an attribute and is computed by subtracting the entropy of distribution in the data after split from the entropy of distribution before split [22]. We use the ranker algorithm for ranking the malware attributes (shown in figure 2). Top 7 features are selected which are used for classification purpose. These are: file_size, importfunt_count, service_creat_count, regread_count, regmod_count, susp_sec_count, packer_used.

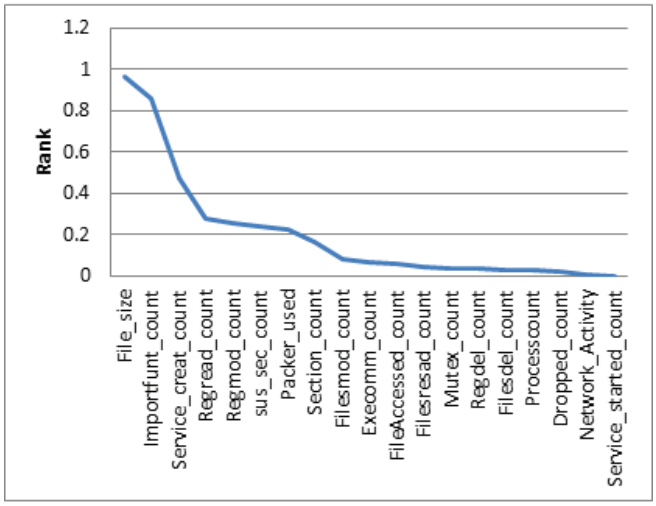

Fig. 2. Malware attribute ranking.

E. Classification

The selected 7 features are then used to build the classification model using machine learning algorithms available in WEKA library [5]. WEKA is a free data mining tool available under the GNU General Public License and is implemented in Java. We use 3.6.10 version of WEKA for Windows OS. We use seven classifiers i.e. IB1, Naïve Bayes (NB), J48, Random Forest (RF), Bagging, Decision Table (DT) and Multi-Layer Perceptron (MLP) from WEKA library for distinguishing malicious files from benign ones. The model build time is observed while conducting the experiments using both the datasets i.e. BFS and AFS.

F. Evaluation and Validation

The training data is required for classification algorithms to build the models and testing data is required to test the models so built. Validation of the models is done by using cross validation technique which is used for evaluating the results by generating independent datasets. In our experiments, we use 10-fold cross validation, as it is a well-accepted method to evaluate the predictions over unseen data for malicious and benign files. The machine learning algorithms are evaluated by using following performance measures which are computed using the fields of confusion matrix as shown in table I:

TABLE I. CONFUSION MATRIX FIELDS ALONG WITH THEIR DESCRIPTION

Field	Description
True Positive (TP)	No. of malicious files classified correctly as malicious
True Negative (TN)	No. of benign files classified correctly as benign.
False Positive (FP)	No. of benign files classified incorrectly as malicious.
False Negative (FN)	No. of malicious files incorrectly classified as benign.

- **True Positive Rate (TPR):** Rate of correctly identified malicious files (also known as recall or sensitivity). It is a measure of completeness or quantity.

$$TPR = \frac{TP}{TP + FN}$$

- **False Positive Rate (FPR):** Rate of incorrectly identified benign files.

$$FPR = \frac{FP}{FP + TN}$$

- **Precision:** Rate of Detection. It is a measure of exactness or quality.

$$Precision = \frac{TP}{TP + FP}$$

- **F-Measure:** It is the harmonic mean of precision and recall.

$$F - Measure = \frac{2 * \mathrm{TP}}{2 * \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

- **Accuracy (%):** Percentage of correctly identified files (both malicious and benign)

$$Accuracy\ (\%) = \frac{TP + TN}{TP + FN + TN + FP} * 100$$

IV. EXPERIMENTAL RESULTS ANALYSIS

This section provides the experimental results and their analysis along with the visualizations. In order to validate the proposed system, a total of 3130 portable executable (PE) files are considered which include 1720 malicious and 1410 clean files. As mentioned in the previous section, all the files are executed in the sandbox to obtain malware attributes which are used to build the classification models using WEKA. The

results obtained for these classifiers are given in table 2. It shows that all the classifiers except NB and MLP perform very well in distinguishing malicious files from benign ones. RF gives the maximum accuracy of 99.97% followed by J48, DT and Bagging. NB gives the minimum accuracy of 95.24%.

TABLE II. CLASSIFICATION RESULTS BEFORE FEATURE SELECTION (WITH 18 FEATURES)

Classifiers	Performance Evaluation Measures				
	TPR	*FPR*	*Precision*	*F-Measure*	*Accuracy (%)*
IB1	0.99	0.01	0.99	0.99	98.98
NB	0.952	0.057	0.955	0.952	95.24
J48	0.998	0.002	0.998	0.998	99.84
RF	1	0	1	1	99.97
Bagging	0.997	0.004	0.997	0.997	99.68
DT	0.998	0.002	0.998	0.998	99.78
MLP	0.981	0.018	0.982	0.981	98.15

Afterwards, the top 7 features are selected using IG method as explained in the previous section and the classification models are again built using the same set of machine learning algorithms for which the result are given in table III. It shows that the maximum accuracy is given by RF followed by DT. For both the classifiers, the accuracy remains same BFS and AFS which means the set of features which are discarded after applying filtering approach are irrelevant. For NB, MLP and Bagging also there is a decrease in accuracy but it is not significant. In case of IB1, there is an increase in accuracy. A comparison of accuracy (%) for these classifiers BFS and AFS is given in figure 3.

TABLE III. CLASSIFICATION RESULTS AFTER FEATURE SELECTION (WITH 7 FEATURES)

Classifiers	Performance Evaluation Measures				
	TPR	*FPR*	*Precision*	*F-Measure*	*Accuracy (%)*
IB1	0.996	0.004	0.996	0.996	99.62
NB	0.952	0.058	0.955	0.951	95.17
J48	0.997	0.003	0.997	0.997	99.71
RF	1	0	1	1	99.97
Bagging	0.996	0.004	0.996	0.996	99.65
DT	0.998	0.002	0.998	0.998	99.78
MLP	0.975	0.022	0.976	0.975	97.54

The classification building time is also noted down while performing experiment BFS and AFS for all the classifiers (shown in table IV). Their comparison is visualized using figure 3. It depicts that for each classifier, the model building time has been decreased significantly (except NB, where it remains same) AFS. For IB1, the model building time is equal to zero as it is a lazy learner. There is a significant improvement in model building time for MLP. We have not shown it in the figure just to visualize the improvement of other classifiers' comparison clearly.

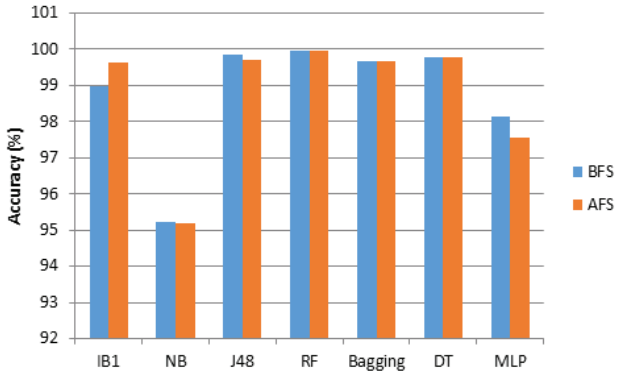

Fig. 3. Comparison of overall accuracy of classifiers before and after feature selection.

TABLE IV. COMPARISON OF MODELS BUILDING TIME (IN SEC.) BFS AND AFS

Classifiers	Time to build the Model (in sec.)	
	BFS	*AFS*
IB1	0	0
NB	0.02	0.02
J48	0.05	0.02
RF	0.11	0.09
Bagging	0.27	0.14
DT	0.83	0.27
MLP	27.42	8.2

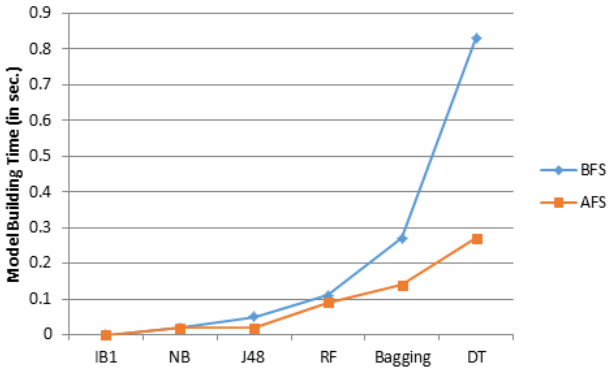

Fig. 4. Comparison of time taken to build the model by different classifiers.

Though very less time is taken to build the classification model for almost all the classifiers but it is to be noted that the experiments have been conducted using a small dataset of 3130 PE files. If the dataset considered is large, it will take large time to build the model and thus the improvement will also seem to be significant.

V. CONCLUSION

This paper presents a malware detection system which is making use of the integrated feature set from both static and dynamic analysis of malware for early detection of zero-day malware with high accuracy. The experimental results show that integrated feature set provides a very good accuracy for all the classifiers except NB. These further demonstrate that the inclusion of filter approach for relevant feature selection can improve the model building time without compromising the accuracy of malware detection system.

REFERENCES

[1] A. Moser, C. Kruegel, E. Kirda, "Exploring Multiple Execution Paths for Malware Analysis," Proc. of IEEE Symposium on Security and Privacy, pp. 231-245. IEEE Computer Society, USA, 2007, doi:10.1109/SP.2007.17.

[2] E. Gandotra, D. Bansal, S. Sofat, "Malware Analysis and Classification: A Survey," Journal of Information Security, vol. 5, pp. 56-65, 2014.

[3] Internet Security Threat Report, Symantec, Volume 21, April, 2016, [online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, "The WEKA Data Mining Software: An Update," ACM SIGKDD Explorations Newsletter, vol. 11, no. 1 pp. 10-18, 2009.

[5] M. Schultz, E. Eskin, F. Zadok, and S. Stolfo, "Data mining methods for detection of new malicious executables," Proc. of 2001 IEEE Symposium on Security and Privacy, IEEE, Oakland, CA, 2001, pp. 38-49, Doi: 10.1109/SECPRI.2001.924286.

[6] J. Kolter, and M. Maloof, "Learning to detect malicious executables in the wild," Proc. of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM New York, NY, USA, 2004, pp. 470–478, doi: 10.1145/1014052.1014105.

[7] D. Kong and G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," Proc. of the ACM SIGMETRICS/ international conference on Measurement and modeling of computer systems, ACM New York, USA, 2013, pp. 347-348, doi: 10.1145/2465529.2465531.

[8] R. Tian, L. Batten, and S. Versteeg, "Function Length as a Tool for Malware Classification," Proc. of the 3rd International Conference on Malicious and Unwanted Software, IEEE, Fairfax, Virginia, 2008, pp. 57-64, doi: 10.1109/MALWARE.2008.4690860.

[9] R. Tian, L. Batten, R. Islam, and S. Versteeg, "An automated classification system based on the strings of Trojan and virus families," Proc. of the 4th International Conference on Malicious and Unwanted Software, IEEE Montréal, Quebec, Canada, 2009, pp. 23-30.

[10] A. Saini, E. Gandotra, D. Bansal, and S. Sofat, "Classification of PE files using static analysis," Proc. of the 7th International Conference on Security of Information and Networks, ACM, University of Glasgow, UK. September, 2014, doi: 10.1145/2659651.2659679.

[11] I. Firdausi, C. Lim, A. Erwin, "Analysis of machine learning techniques used in behavior based malware detection," Proc. of 2nd International Conference on Advances in Computing, Control and Telecommunication Technologies, IEEE, Jakarta, 2010, pp. 201-203.

[12] Anubis, [online]. Available: http://anubis.iseclab.org/

[13] S. Nari, A. Ghorbani, "Automated Malware Classification based on Network Behavior," Proc. of International Conference on Computing, Networking and Communications (ICNC), IEEE,. San Diego, CA, 2013, pp. 642-647, doi: 10.1109/ICCNC.2013.6504162.

[14] T. Lee, J. Mody, "Behavioral classification," Proc. of the European Institute for Computer Antivirus Research Conference, 2006.

[15] I. Santos, J. Devesa, F. Brezo, J. Nieves, PG. Bringas, "OPEM: A static-dynamic approach for machine learning based malware detection," Proc. of International Conference CISIS'12-ICEUTE'12 Special Sessions Advances in Intelligent Systems and Computing, 189, 2013, pp. 271-280, doi: 10.1007/978-3-642-33018-6_28.

[16] R. Islam, R. Tian, .L. Battenb, S. Versteeg S, "Classification of malware based on integrated static and dynamic features," Journal of Network and Computer Application, vol. 36, pp. 546-556, 2013.

[17] Anderson B, Storlie C, Lane T, "Improving Malware Classification: Bridging the static/Dynamic Gap," Proc.of 5th ACM workshop on Security and Artificial Intelligence, ACM, New York, USA, 2012, pp. 3-14.

[18] E. Gandotra, D. Bansal, S. Sofat, "Integrated Framework for Classification of Malware," Proc. of 7th International Conference on Security of Information and Networks, ACM, University of Glasgow, UK. September, 2014, doi: 10.1145/2659651.2659738.

[19] Cuckoo, [Online]. Available: http://www.cuckoosandbox.org/

[20] Modified edition of cuckoo, [Online]. Available: https://github.com/brad-accuvant/cuckoo-

[21] D. Crockford, The JSON data interchange format. Technical report, 2013, ECMA International.

[22] Breadcrumb, Data Mining – Information Gain, [Online]. Available: http://gerardnico.com/wiki/data_mining/information_gainhttp