Malware Analysis and Detection Using Data Mining and Machine Learning Classification

Mozammel Chowdhury(✉), Azizur Rahman, and Rafiqul Islam

School of Computing and Mathematics, Charles Sturt University, Wagga Wagga, Australia
{mochowdhury, azrahman, mislam}@csu.edu.au

Abstract. Exfiltration of sensitive data by malicious software or malware is a serious cyber threat around the world that has catastrophic effect on businesses, research organizations, national intelligence, as well as individuals. Thousands of cyber criminals attempt every day to attack computer systems by employing malicious software with an intention to breach crucial data, damage or manipulate data, or to make illegal financial transfers. Protection of this data is therefore, a critical concern in the research community. This manuscript aims to propose a comprehensive framework to classify and detect malicious software to protect sensitive data against malicious threats using data mining and machine learning classification techniques. In this work, we employ a robust and efficient approach for malware classification and detection by analyzing both signature-based and anomaly-based features. Experimental results confirm the superiority of the proposed approach over other similar methods.

Keywords: Data security · Cyber threat · Malware · Classification · Machine learning

1 Introduction

With the emerging applications of computer and information technology, protecting sensitive data has become a significant challenge all over the world. Hundreds of new malicious programs are released by cyber criminals through the Internet to steal or destroy important data. Malicious software or malware breaches the secrecy and integrity of data and causes unauthorized leakage of information [1]. Such malware threats are expansive, not only in terms of quantity, but also in terms of quality. In recent years, researchers are becoming more concerned about protecting sensitive data from growing cyber-attacks. Hence, securing important data has achieved immense interest in the industry, business, and computer user community.

Over the last decade, researchers have proposed a diversity of solutions in order to defend malware attacks. Most of the solutions generally use two main approaches for malware detection: (i) signature-based approach and (ii) anomaly or behavior-based approach. The signature-based methods are very efficient to detect known malware [2]. A signature-based malware detection system analyses the signature or fingerprint of a file and then compares to a database of signatures of known malicious files. A signature could represent a series of bytes in the file or a cryptographic hash of the file or its

J. Abawajy et al. (eds.), *International Conference on Applications and Techniques in Cyber Security and Intelligence*, Advances in Intelligent Systems and Computing 580,
DOI 10.1007/978-3-319-67071-3_33

sections. However, most of the malwares can generate new variants each time it is executed and a new signature is generated. Therefore, signature based approaches fail to detect unknown or zero-day malwares which are not in the database [17–20]. In contrast, anomaly or behavior-based detection approaches use API call sequences instead of byte sequence matching [3]. Although anomaly-based detection approaches use the knowledge of normal behavior patterns and performs better than the signature based approaches, they have however, high false alarm rate.

To circumvent these challenges of the conventional methods, machine learning has been proposed in recent times to detect malware. Machine learning techniques have been proven to be capable of detecting new malware variants [4]. Machine learning techniques used for detection and classification of malicious entity include support vector machines, decision tree, random forest, and Naive Bayes [3–6]. However, machine learning approaches can have shortcomings of increasing false alarm rate due to weak feature selection and inefficient classifier generation. Moreover, these classification methods require many training samples to build the classification models. Hence, accurate detection of malicious programs is still a key challenge for the cyber community. We therefore, propose a hybrid framework for malware classification integrating a binary associative memory (BAM) with a multilayer perceptron (MLP) neural network by using both signature-based and behavior-based features analysis. In this work, we employ signature-based n-gram features and behavior-based API (Application Programming Interface) call sequences for malware analysis. The main contributions of this work can be summarized as follows:

- We propose a robust and efficient approach for malware classification and detection using a hybrid framework with combination of a binary associative memory (BAM) and a multilayer perceptron (MLP) neural network.
- The BAM network can significantly reduce feature dimensions collected from a large malware dataset.
- We employ hybrid features for malware analysis by integrating both signature-based and behavior-based features that clearly increases classification and detection accuracy.

The rest of the paper is structured as follows. Section 2 demonstrates the proposed approach for malware analysis and classification. Section 3 presents the experimental evaluations and we conclude the paper in Sect. 3.

2 Proposed System Architecture

The proposed scheme for malware classification and detection is consisted of the following major components: (i) Pre-processing, (ii) Features extraction, (iii) Feature refinement/selection, (v) Classification, and (vi) Detection. The architecture of the proposed approach is depicted in Fig. 1.

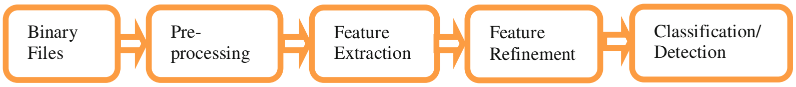


Fig. 1. Architecture of the proposed malware detection system.

2.1 Pre-processing

The collected files are raw executable files; they are stored in the file system as binary code. To make them suitable for our work, we have preprocessed them. First, we unpack the executables in a restricted environment called virtual machine (VM). In order to unpack the packed executables automatically, we use the tool PEid [7].

2.2 Feature Extraction

Features extraction is carried out by analyzing executable files based on static and dynamic analysis. In this work, we extract two types of features from each of the malware and cleanware executable files: N-gram features and Windows API calls.

2.2.1 N-Gram Features

The n-gram features are the sequences of substrings with a length of n-bytes extracted from an executable file. The benefit of using n-gram technique is that it can capture the frequency of words having a length of n-grams. Empirically we find that n-grams of size 5 (each sequence is exactly 5 bytes) produce the most overall accurate results. We extract the n-gram features using the n-gram feature extraction method [8].

2.2.2 Windows API Calls

API call information shows how malware behaves. API list can be extracted from PE format of the executable files. The Portable Executable (PE) header contains information about how the operating system manages a resource allocated to a program. In this work, we employ the most reliable disassembly tool Interactive Disassembler Pro (IDA Pro) [9] to disassemble the binary file to analyze and extract the Windows API calls. IDA Pro can disassemble all types of non-executable and executable files (such as ELF, EXE, PE, etc.). It automatically recognizes API calls for various compilers and provide the hooks to call custom defined plugins resulting in incredibly powerful implementation with flexible levels of analysis and control. IDA Pro loads the selected file into memory to analyze the relevant portion of the program and creates an IDA database. IDA Pro generates the IDA database files into a single IDB file (.idb) by disassembling and analyzing the binary of the file. IDA Pro provides access to its internal resources via an API that allows users to create plugins to be executed by IDA Pro. We have used *idapython* [10] system that runs the disassembly module automatically for generating the .idb database. The *ida2sql* plugin is used to export .idb database into MySQL database (.db) for better binary analysis.

ida2sql plugin generates 16 tables (such as, Address comments, Address reference, Basic blocks, callgraph, control_flow_graph, data, function etc.) for every binary executable [11]. Each of them contains different information about the binary content.

For example, Function table contains all the recognizable API system calls and non-recognizable function names and the length (start and the end location of each function). Instructions table contains all the operation code (OP) and their addresses and block addresses.

We extracted the list of API calls using Function table. The Microsoft Developer Network (MSDN) [12] is used for matching and in identifying the windows API's. A program is implemented to compare and match the API from MSDN and the API calls generated in the database for the malware sample set. To list all the API calls that are associated with malcode are collected using machine opcodes such as Jump and Call operations as well as the function type.

2.3 Feature Selection/Refinement

After the extraction of the n-gram features, the most informative features are selected and the best one is examined based on the calculation of the classifier accuracy that corresponds with the number of features that are selected using different feature selection methods. In this work, we use Principal Components Analysis (PCA) [13] for feature selection. The PCA is employed for increasing computation speed due to its capacity for dimensionality reduction. It is based on converting a large number of variables into a smaller number of uncorrelated variables by finding a few orthogonal linear combinations of the original variables with the largest variance.

Furthermore, we perform the Class-wise document frequency (DCFS) [14] based feature selection measure to get the relevant API calls for each malware category separately to increase the detection and classification accuracy.

2.4 Malware Classification and Detection

Classification process is divided into two stages: training and testing. In the training phase, a training set of malicious and benign files is provided to the system. The learning algorithm trains a classifier. The classifier learns from the labeled data samples. In the testing phase, a set of new malicious and benign files are fed into the classifier and classified as malware or cleanware.

In this work, we propose a hybrid framework for malware classification integrating a binary associative memory (BAM) with a multilayer perceptron (MLP) neural network, as shown in Fig. 2. The BAM works for dimensional reduction of the feature matrix to make the classification faster and more efficient. The MLP with backpropagation algorithm is used to train it with the selected features to classify and detect the malware. The selected features of the malware datasetare fed into the input layer of the BAM networks which provided reduce features as output and it's output is fed into the MLP neural netwwork. The number of nodes in the MLP output layer equals to the number of malwares in the dataset to classify. The number of epochs for this experiment was 50,000 and the minimum error margin was 0.002.

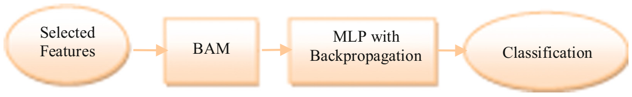


Fig. 2. The proposed hybrid classification approach.

3 Experimental Results and Discussion

In this section, we present the experimental results of our proposed approach for malware classification and detection. Experiments are carried out with the collected datasets of both malware and cleanware. In this work, we employ signature-based n-gram features and behavior-based API (Application Programming Interface) call sequences for malware analysis.

3.1 Data Set

In this work, we have collected 52,185 executable files in total consisting of 41,265 recent malicious files and the remaining being benign files. Table 1 represents the malware and cleanware datasets used in the experiment. The malicious files are collected from VX Heaven [15] and the benign files are collected from online sources: Download.com and Softpedia.com. The malware collection consists of Trojans, backdoors, hack tools, root kits, worms and other types of malware.

Table 1. Malware and cleanware dataset

File type		Number of files
Malware	Backdoor	5,340
	Virus	13,645
	Rootkit	423
	Trojan	11,435
	Worm	9,556
	Exploit	329
	Other	537
Cleanware		10,920
Total		52,185

3.2 Experimental Evaluation

In order to evaluate our proposed method, we use two different datasets: a malware dataset and a benign dataset. For extracting or mining n-grams, we first dump all benign and malware files into separate databases. The two databases are then merged into a new common database with removing the duplicate n-grams. We extract the n-grams features for every file in that dataset for a n-gram length of 5 (n = 5). The number of n-gram features obtained from malware and cleanware datasets is 120,598 features, which is very large. We therefore refine these features using PCA and select the top 1,000 features.

The API function calls found in the DLL imports of the Windows PE structure are used as behavior or anomaly-based features as they provide the best insight into the behavior of a sample. We use static analysis tool IDA Pro to disassemble the binary file of the malware and cleanware sample to analyze and extract the Windows API calls. We obtain a total of 152,641 different API calls from our samples. We refine them using Class-wise document frequency (DCFS) measure to select the top API calls. For each class, we record the top 100 most frequent API calls, creating a database of the top identifying API calls. Using the database of the most frequent API calls of each class, we create a feature vector of 1's and 0's (1 if a certain API call was present, 0 if it was not) for each sample. We combine these two different datasets into one, creating thus an integrated signature-behavior based features dataset. To compare our method, we have also kept the datasets with only the signatures-based features and only the behavior-based features.

In order to compare the performance of our approach with other similar techniques, we run and evaluate the machine learning algorithms in the Waikato Environment for Knowledge Analysis (WEKA) platform [16] using similar features. We compare our approach with four prominent machine learning classifiers including, Support Vector Machine (SVM), Decision Tree (J48), Naïve Bayes, and Random Forest. WEKA uses a unique file format as input, called Attribute-Relation File Format (ARFF) database. This format allows to list all file features and their type (numeric, nominal, string, etc.). We convert both malware and cleanware data sets into WEKA format. We pass training set to the WEKA library to train the classifier and then justify the effectiveness with the test dataset.

To validate each classifier, we perform a k-fold cross validation. Empirically it is found that with k = 10 the classifier provides better performance. In this way, our dataset is randomly divided into 10 different sets of learning and testing, and each set has approximately the same class distribution as the total data sets. Thus, 90% of the total dataset is used for training and the rest 10% of the total data is used for testing. For each validation step, we conducted the learning phase of the algorithms with the training datasets, applying different parameters or learning algorithms depending on the concrete classifier.

To evaluate the performance of the algorithms, we estimate the following evaluation metrics:

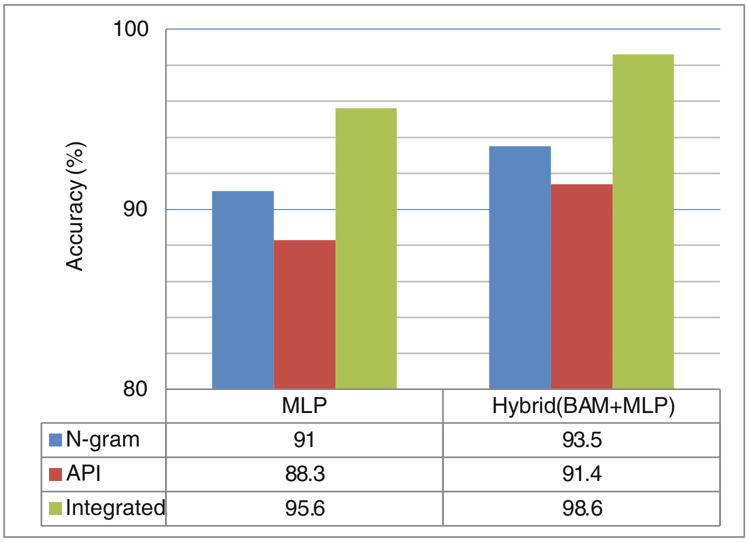
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

$$FPR = \frac{FP}{FP + FN} \tag{3}$$

where,

TP (true positive) = No of malware files correctly identified as malware,
FP (false positive) = No of malware files incorrectly identified as cleanware,
TN (true negative) = No of cleanware files correctly identified as cleanware,
FN (false negative) = No of cleanware files incorrectly identified as malware,
TPR = True Positive Rate (sensitivity), and FPR = False Positive Rate.



	MLP	Hybrid(BAM+MLP)
🟦 N-gram	91	93.5
🟥 API	88.3	91.4
🟩 Integrated	95.6	98.6

Fig. 3. Accuracy of the proposed method with integrated and individual features.

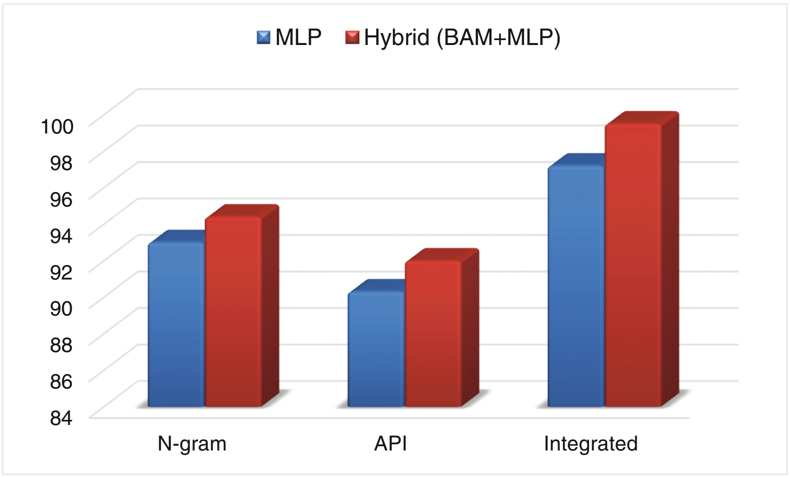


Fig. 4. TPR results of the proposed method with integrated and individual features.

We test our algorithm with integrated features and individual features separately. Experimental results of our proposed method are reported in Figs. 3, 4 and 5 for accuracy, TPR and FPR, respectively. The results show that the proposed classifier gives better accuracy in case of using integrated features and employing hybrid network (BAM with MLP). The performance results for different approaches are reported in Table 2, which clearly indicate the superiority of our proposed method.

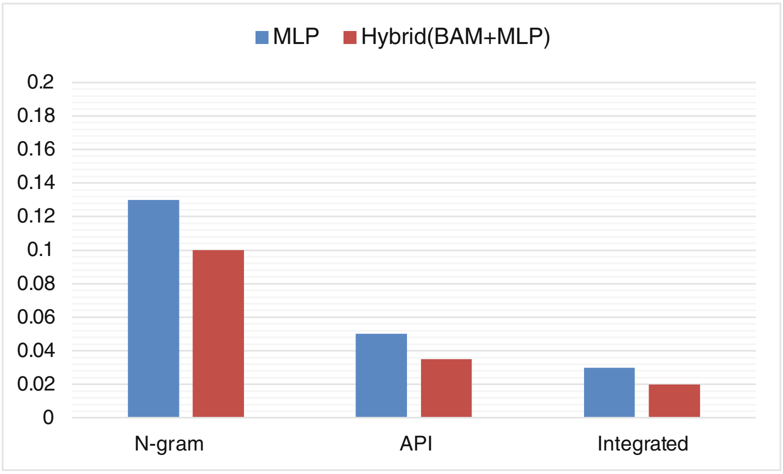


Fig. 5. FPR results of the proposed method with integrated and individual features.

Table 2. Comparison of overall accuracy for different classifiers.

Methods	Accuracy (%)	TPR (%)	FPR (%)
Naïve Bayes	88.5	89	12
J48	91.1	92	8
Random forest	94.8	95.5	6
SVM	95.7	97	3.6
Proposed approach	98.6	99.5	2

4 Conclusion

This paper proposes an efficient and robust malware detection scheme using machine learning classification algorithm. We have explored the variations of parameters and their effect on the accuracy of malware classification. Our approach combines the use of N-gram and API Call features. Experimental evaluation confirms the effectiveness and robustness of our proposed method. In the future work, we aim to use more variant features in conjunction with each other to achieve more detection accuracy and reduce false positive as well.

References

1. Islam, R., Tian, R., Batten, L.M., Versteeg, S.: Classification of malware based on integrated static and dynamic features. J. Netw. Comput. Appl. **36**, 646–656 (2013)
2. Tang, K., Zhou, M.T., Zuo, Z.-H.: An enhanced automated signature generation algorithm for polymorphic malware detection. J Electron. Sci. Technol. China **8**, 114–121 (2010)
3. Tian, R., Islam, R., Batten, L., Versteeg, S.: Differentiating malware from cleanware using behavioural analysis. In: International Conference on Malicious and Unwanted Software: MALWARE 2010, pp. 23–30 (2010)
4. O'Kane, P., Sezer, S., McLaughlin, K., Im, E.: SVM training phase reduction using dataset feature filtering for malware detection. IEEE Trans. Inf. Forensics Secur. **8**(3), 500–509 (2013)
5. Hadžiosmanović, D., Simionato, L., Bolzoni, D., Zambon, E., and Etalle, S.: N-gram against the machine: on the feasibility of the N-gram network analysis for binary protocols. Research in Attacks, Intrusions, and Defenses. Springer, pp. 354–373 (2012)
6. Chowdhury, M., Rahman, A., Islam, R.: Protecting data from malware threats using machine learning technique. In: IEEE Conference on Industrial Electronics and Applications (ICIEA 2017), 18–20 June 2017, Siem Reap, Cambodia (2017)
7. PEid Unpacker. http://www.peid.info/
8. Jain, S., Meena, Y.K.: Byte level N-gram analysis for malware detection. Comput. Netw. Intell. Comput. **157**, 51–59 (2011). (Springer)
9. IDA Pro Disassembler and Debugger, IDA Pro (2010)
10. Idapython (2009). http://code.google.com/p/idapython/
11. Zynamics BinNavi. http://www.zynamics.com/binnavi
12. Windows API Functions, MSDN. http://msdn.microsoft.com/enus/library/aa383749%28VS.85%29.aspx. Accessed Jan 2010
13. Xu, X., Wang, X.: An adaptive network intrusion detection method based on PCA and support vector machines. In: Advanced Data Mining and Applications. Springer, pp. 696–703 (2005)
14. Devesa, J., Santos, I., Cantero, X., Penya, Y.K., Bringas, P.G.: Automatic behaviour-based analysis and classification system for malware detection. In: Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS) (2010)
15. VX Heaven collection, VX Heaven website. http://vx.netlux.org
16. Weka library, Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka
17. Walls, J., Choo, K.-K.R.: A review of free cloud-based anti-malware apps for android. In: Proceedings of 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2015), pp. 1053–1058, 20–22 Aug 2015, IEEE Computer Society Press (2015)
18. Milosevic, N., Dehghantanha, A., Choo, K.-K.R.: Machine learning aided android malware classification. Comput. Electr. Eng. (2017). doi:10.1016/j.compeleceng.2017.02.013
19. Afifi, F., Anuar, N.B., Shamshirband, S., Choo, K.-K.R.: DyHAP: dynamic hybrid ANFIS-PSO approach for predicting mobile malware. PLoS ONE **11**(9), e0162627 (2016)
20. Damshenas, M., Dehghantanha, A., Choo, K.-K.R., Mahmud, R.: M0DROID, android behavioral-based malware detection model. J. Inf. Priv. Secur. **11**(3), 141–157 (2015)