Reducing False Notification in Identifying Malicious Application Programming Interface(API) to Detect Malwares Using Artificial Neural Network with Discriminant Analysis

Abbas M. Al-Bakri
College of Computer and Mathematics/ University of Kuffa
Hussein L. Hussein
Dept.of Computer Science/College of Eduucationfor Pure Science(IbnAl-Haitham)/Universityof Baghdad

Received in:7 September 2014 Accepted in: 24 November2014

Abstract

This paper argues the accuracy of behavior based detection systems, in which the Application Programming Interfaces (API) calls are analyzed and monitored. The work identifies the problems that affecting the accuracy of such detection models. The work was extracted (4744) API call through analyzing. The new approach provides an accurate discriminator and can reveal malicious API in PE malware up to 83.2%. Results of this work evaluated with Discriminant Analysis.

Keywords: PE Malwares, Malicious API, ANN, Discriminant Analysis.

Introduction

At present, most widely used malicious executables detections adopt the technique of signature-based pattern matching. It is simple and highly efficient, but only works for the malicious executables which we have already known. For the unknown ones, this approach is ineffective. Thus an alternative way has been brought into focus to resolve this problem-identify the malicious executables based on their behaviors [1]. With the improvement of the techniques used by malwares like virus, Trojan and worm, traditional malware detection approaches based on signatures, were not effectively enough to detect variables of known malwares [2].

Today security threats like malware are more sophisticated and targeted than ever, and they are growing at an unprecedented rate. To deal with them, various approaches are introduced. One of them is Signature-based detection, which is an effective method and widely used to detect malware; however, there is a substantial problem in detecting new instances. In other words, it is solely useful for the second malware attack. Due to the rapid proliferation of malware and the desperate need for human effort to extract some kinds of signature, this approach is a tedious solution; thus, an intelligent malware detection system is required to deal with new malware threats. Most of intelligent detection systems utilize some data mining techniques in order to distinguish malware from sane programs. One of the pivotal phases of these systems is extracting features from malware samples and benign ones in order to make at least a learning model. This phase is called “Malware Analysis” which plays a significant role in these systems. Since API call sequence is an effective feature for realizing unknown malware, this paper is focused on extracting this feature from executable files.

There are two major kinds of approach to analyze executable file.

The first type of analysis is “Static Analysis” which analyzes a program in source code level.

The second one is “Dynamic Analysis” that extracts features by observing program activities such as system requests during its execution time. Static analysis has to traverse the program execution path in order to find called APIs.

Because it does not have sufficient information about decision making points in the given executable file, it is not able to extract the real sequence of called APIs. Although dynamic analysis does not have this drawback, it suffers from execution overhead [3].

Code analysis techniques can be categorized into two main classes: dynamic techniques and static techniques. Approaches that belong to the first category rely on monitoring execution traces of an application to identify the executed instructions and their actions, or behavior. Approaches that belong to the second category analyze the binary structure statically, parsing the instructions as they are found in the binary image and attempting to determine a (possibly over-approximated) set of all possible behaviors. Both static and dynamic approaches have advantages and disadvantages. Static analysis takes into account the complete program, while dynamic analysis can only operate on the instructions that were executed in a particular set of runs. Therefore, it is impossible to guarantee that the whole executable with all possible actions were covered when using dynamic analysis. On the other hand, dynamic analysis assures that only actual program behavior is considered. This eliminates possible incorrect results due to overly conservative approximations that are often necessary when performing static analysis [4].

Recently researchers employ behavior based detection models that depend on API tracking and analyzing to identify suspected PE applications. Those behavior models become more efficient than the signature based detection systems for detecting zero-day malwares. This is because a simple polymorphic or metamorphic malware can defeat signature based detection systems easily. Moreover, it is difficult to keep the signature of all malwares and their variants because the malware number becomes more than five millions, and every day (5000) new malwares or variants are expected over the internet and networks [5].

Related Works

In recent years many malware researchers have focused on neural network and / or data mining to detect unknown malwares. Data mining is the process of analyzing electronically stored data by automatically searching for patterns. Machine learning algorithms have been used widely for different data mining problems to detect patterns and to find correlations between data instances and attributes. Many researchers have used n-grams or API calls as their primary type of feature that are used to represent malware instances in a suitable format for data mining purposes.

Saman Mirza Abdulla1[5] proposed a method using multi-layer neural network to classify the features of API calls sequence by using 6-grams at least to distinguish between malicious and benign , by examining the output of neural network as: out <= 0.3 then benign , or out >= 0.7 the malicious, and the other values will use new method named costimulation.

Shultz et al. [6] proposed a method using data mining techniques for detecting new malicious executables. Three different types of features are extracted from the executables, i.e. the list of DLLs used by the binary, the list of DLL function calls, and number of different system calls used within each DLL. Also they analyze byte sequences extracted from the hex dump of an executable. The data set consisted of 4,266 files out of which 3,265 were malicious and 1,001 were legitimate or benign programs. A rule induction algorithm called Ripper [7] was applied to find patterns in the DLL data. A learning algorithm Naïve Bayes (NB), which is based on Bayesian statistics, was used to find patterns in the string data and n-grams of byte sequences were used as input data for the Multinomial Naïve Bayes algorithm. A data set is partitioned in two data sets, i.e., a test data set and a training data set. This is to allow for performance testing on data that are independent from the data used to generate the classifiers. The Naïve Bayes algorithm, using strings as input data, yielded the highest classification performance with an accuracy of 97.11%. The authors compared their results with traditional signature-based methods and claimed that the data mining-based detection rate of new malware was twice as high in comparison to the signature-based algorithm.

Dataset description

Our dataset consists of a total of 4744 Windows programs in PE file format that are 1736 malicious and 3008 clean programs. There were no duplicate API vectors in our dataset. The malicious and benign excel sheet were obtained from [8], Windows' system files and our local laboratory's program files [8].We only consider non-packed Windows binary files in PE format. Some malware writers compress their code to make them undetectable by antivirus programs.

This work argued the problems and challenges that may affect the accuracy of API calls behavior detection models. The work analyzed the API calls using supervised Neural Network (NN).

The work will compare the results for all these classifier techniques with the result that are obtained from NN with the Discriminant Analysis (DA).

Methods for Detecting Unknown Malware

We selected the static analysis method as the focus of this research. This is due to the technical limitations of the dynamic method, and also to the fact that constant monitoring on every computer is necessary to prevent malware. To develop a detection method based on static analysis, we must overcome a number of challenges, such that.

- Most viruses today feature a mechanism for self-encryption to circumvent simple file checking or detection by pattern matching.

- There are only two ways to detect such viruses when these are unrecognized: by cracking the encrypted code or by allowing the viruses to activate decryption on their own.

To determine whether an executable program will exhibit malicious behaviour (corruption of files, mailing of confidential documents, etc.), it is necessary to analyse the special function calls requested by the program from the operating system. Operating systems such as Windows and UNIX protect files and other resources from direct manipulation by ordinary programs. Even malwares must call functions from the operating system in order to perform any given malicious action. A mechanism is therefore required to identify sets of function calls that are indicative of malicious behaviours [9].

Research Motivation

In this section, we will review the core computational intelligence approaches that have been proposed to solve malware detection problems. We shall discuss data collection, artificial neural networks, as soft computing and discriminant analysis as data mining method as shown in fig-1.

-Data Collection and Pre-Processing

The work analyzed (4744) of PE applications. Samples have been collected from [8].As the work will classify the API functions of each suspected application.

- As a result of this stage, the window size of each vector is ranged from 4 to 8 API elements, the vectors which are collected according to this research are :-
- 938 vectors of API calls have been prepared after duplicated vectors have been deleted.
- 1218 vectors of API calls have been prepared after duplicated vectors have been deleted.
- 1410 vectors of API calls have been prepared after duplicated vectors have been deleted.
- 1520 vectors of API calls have been prepared after duplicated vectors have been deleted.
- 1602 vectors of API calls have been prepared after duplicated vectors have been deleted.

-Artificial Neural Networks

This work tries to do two tests on the prepared vectors of API calls to check the conditions that affect the accuracy of classifiers and to check their accuracy rates.
An artificial neural network (ANN) consists of a collection of processing units called neurons that are highly interconnected in a given topology. ANNs have the ability of learning-by-example and generalizing from limited; they have been successfully employed in a broad spectrum of data intensive applications. In this section, the research will review the contribution to and performance in the malware detection domain.

To predict the status of an API call vector, this work has applied vectors to a supervised neural network that depend on Feed forward Back-Propagation (FFBP) as a training algorithm. The structure of the neural network contains (4-8) nodes at input layer, two hidden layers ,the first hidden layer ranged from 4 to 15 nodes and the second hidden layer ranged from 8 to 30 nodes hidden , and one node at output layer . The output range of the neural network will be between (0, 1); the case (0) indicates beingn and (1) indicates malwares.

We have divided the data set into two parts; first part has number of vectors used during training phase, while remain vectors has used for testing. Each point in this figure indicates how many vectors that labelled as normal or malware has the same prediction output. The threshold used for each neural network is different according to the output. Cases in such a situation probably lead to generate False Alarm (FA) as shown in fig-2.

-Discriminant Analysis (DA)

Discriminant function analysis is used to determine which variables discriminate between two or more naturally occurring groups. Computationally, discriminant function analysis is very similar to analysis of variance (*ANOVA*).

Improving API Behavior Based Models

Going through API classifying problems and doing more evaluations will direct such models toward improving their accuracy and minimizing False Alarm (FA) rates. Therefore, minimizing the number of cases in this area means minimizing errors and will improve the accuracy of identification and classification rate.

To do that, this work employed a double checking or scanning process on the same suspected vectors of API calls, but in two different directions. This ides has been taken out from the Co-stimulation phenomenon that occurred inside the Human Immune System (HIS). The process of Co-stimulation provides a very good discrimination procedure to HIS. It displaces the self-cells from non-self-cells with zero rates of FA. It achieves that by involving two different Immune cells in checking a foreign body. The first cell checks the normality, while the second cell checks the viral degree of the same suspected foreign body. After that, both cells will send back their results to Immune System (IS). Then, the final decision to identify or classify a foreign body will depend on the checking results that came from both Immune cells [5].

Our model framework that simulates as Co-stimulation phenomenon is illustrated in Figure 4. The model has two main lines. Each line, differently, will scan API call vectors. After that, results that obtained from both lines will be sent to Co-stimulation unit. There, both results will be considered for identification or classification.

-The First Scanning method

In this method, the sequence of API functions that called by a suspected application will be scanned. Each time, the scanner will read number of elements (n-grams) as vector; the n-grams will be rated from 4 to 8 elements. Later, the vector will be compared with many predefined normal and abnormal vectors. A threshold set to classify vectors into three groups. Vectors will be labelled as Normal, if their output result (output<=threshold-1), or as Malware *if* (output>=threshold-2), otherwise they are considered as ambiguous vectors and labelled as *FA*. Then, the mean percentage of each group will be calculated. Another thing that should be kept at output stage is the vectors in FA for further checking. To achieve this scanning line, this work has employed a *Feed forward Back-Propagation* (FF-BP) Neural network tool, because it can solve very complex and nonlinear problems. The structure of the used Neural Network is illustrated in Figure-2 which determined that the input layer will be ranged from 4 to 8 input elements in each vector while the first and second hidden layer will be ranged from 4 to 15 for first hidden layer and from 8 to 30 for second hidden layer. Through this line of scanning, the normality and viral degree of a suspected file somehow can be estimated. This work tests the first line using (1736) malicious API calls; vector still needs improvement, because the possibility of FA generation is high.

- The Second Scanning method

The aim of the second scanning is to improve or to support what have been obtained through the first line. Here, vectors will be compared with behaviors that a malware may perform during execution. We can classify the behaviors that are expected by using a special data mining method named Discriminant Analysis (DA) which results are obtained by tables (1, 2, 3, 4 and 5) for different input vectors.

Conclusions

The problem of accurate discrimination between normal and malware (API) function calls have been explained in this paper. This problem challenges the most malware behavior detection models that depend on (API) tracing and analyzing.
The problem affects the accuracy of detection and increases the rate of (FA).
Based on the biological Co-stimulation process, this paper advised a new detection model that scans any suspected application in two methods. The model established a conferment process between both scan lines. The results show that the proposed model can detect unseen malware and can increase accuracy of (API) behavior detection systems between 77%- 83.2%.

References

1. Jacob, G., Debar, H., and Filiol, E. (2008) "Behavior detection of malware: from a survey toward an established taxonomy", Journal in Computer Virology, August 2008, 4,. , springer. :251-266
2. Forrest, S., et al. Longstaff (1996)"A Sense of Self for Unix Processes". Proceedings of IEEE Symposium on Security and Privacy[C],USA:IEEE Computer Society Press, The ACM Digital Library,: 120~128.
3. Mojtaba Eskandari , Zeinab Khorshidpour and Sattar Hashemi(2013)"HDM-Analyser: a hybrid analysis approach based on data mining techniques for malware detection", J Comput Virol Hack Tech,: 9:77–93.
4. Giovanni Vigna(2007)" static Disassembly and Code Analysis",springer.
5.Saman Mirza Abdulla1, Assoc. Prof. Dr. Miss Laiha Mat Kiah, and Assoc. Prof. Dr. Omar Zakaria (2012)"Minimizing Errors in Identifying Malicious API to Detect PE Malwares Using Artificial Costimulation" , International Conference on Emerging Trends in Computer and Electronics Engineering, March 24-25Dubai.
6. Schultz, M. G., Eskin, E., and Stolfo, S. J.(2001) "Data mining methods for detection of new malicious executables," in Proceedings of the IEEE Symp. on Security and Privacy,. 38-49.
7. Cohen, W.(1995) "Fast effective rule induction,." Proc. 12th International Conference on Machine Learning, 115-23, San Francisco, CA: Morgan Kaufmann Publishers.
8.http://csminig.org/index.php/malicious-software-datasets-.html.
9. Mori Akira; Sawada Toshimi; Izumida Tomonori and Inoue Tadashi*(2005)*"Detecting Unknown Computer Viruses - A New Approach",*Journal of the National Institute of Information and communications Technology .52 Nos.1/2 ,75-88.*
10. Shelly Xiaonan Wu* and Wolfgang Banzhaf(2010)"The use of computational intelligence in intrusion detection systems: A review", Applied Soft Computing 10 :1–35.

Table No.(1): DA using 4 API in each vector

Classification Results

			Predicted Group Membership		
		4 API	0	1	Total
Original	**Count**	0	519	42	561
		1	168	209	377
	%	0	92.5	7.5	100.0
		1	44.6	55.4	100.0

77.6% of original grouped cases correctly classified.

Table No.(2): DA using 5 API in ea vector.

Classification Results

			Predicted Group Membership		
		5 API	0	1	Total
Original	**Count**	**0**	721	46	767
		1	207	244	451
	%	**0**	94.0	6.0	100.0
		1	45.9	54.1	100.0

79.2% of original grouped cases correctly classified.

Table No.(3):DA using 6 API in each vector.

Classification Results

			Predicted Group Membership		
		6 API	*0*	*1*	*Total*
Original	***Count***	***0***	*839*	*56*	*895*
		1	*215*	*300*	*515*
	%	***0***	*93.7*	*6.3*	*100.0*
		1	*41.7*	*58.3*	*100.0*

80.8% of original grouped cases correctly classified.

Table No.(5): DA using 8 API in each vector.

Classification Results

			Predicted Group Membership		
		8 API	0	1	Total
Original	Count	0	956	48	1004
		1	231	367	598
	%	0	95.2	4.8	100.0
		1	38.6	61.4	100.0

82.6% of original grouped cases correctly classified.

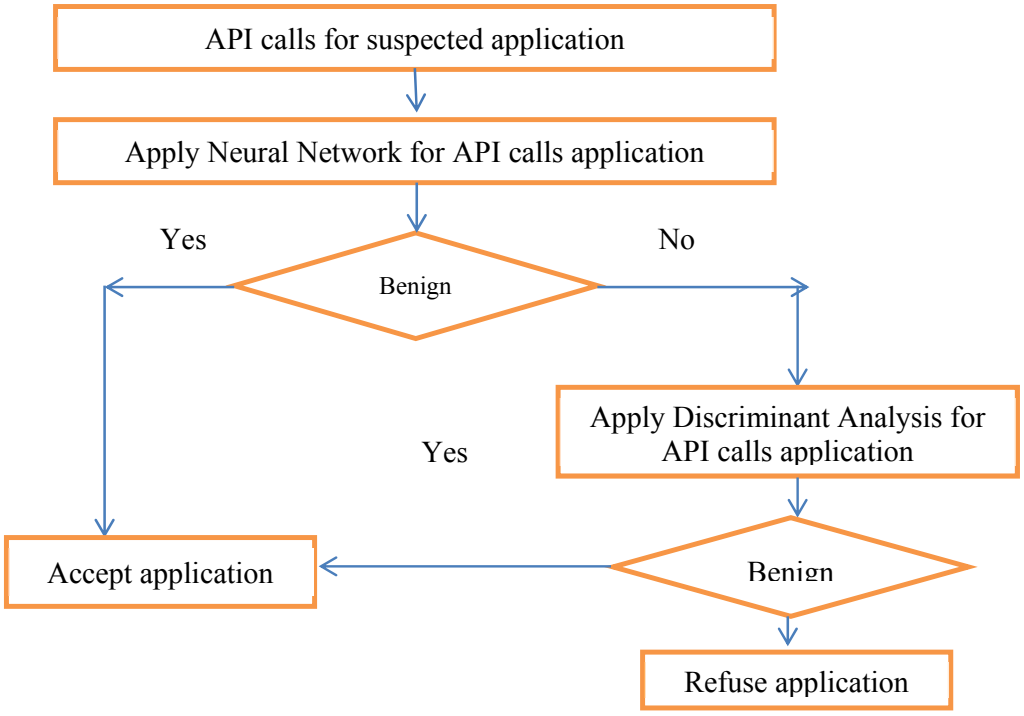

Figure No.(1):Research scanning framework

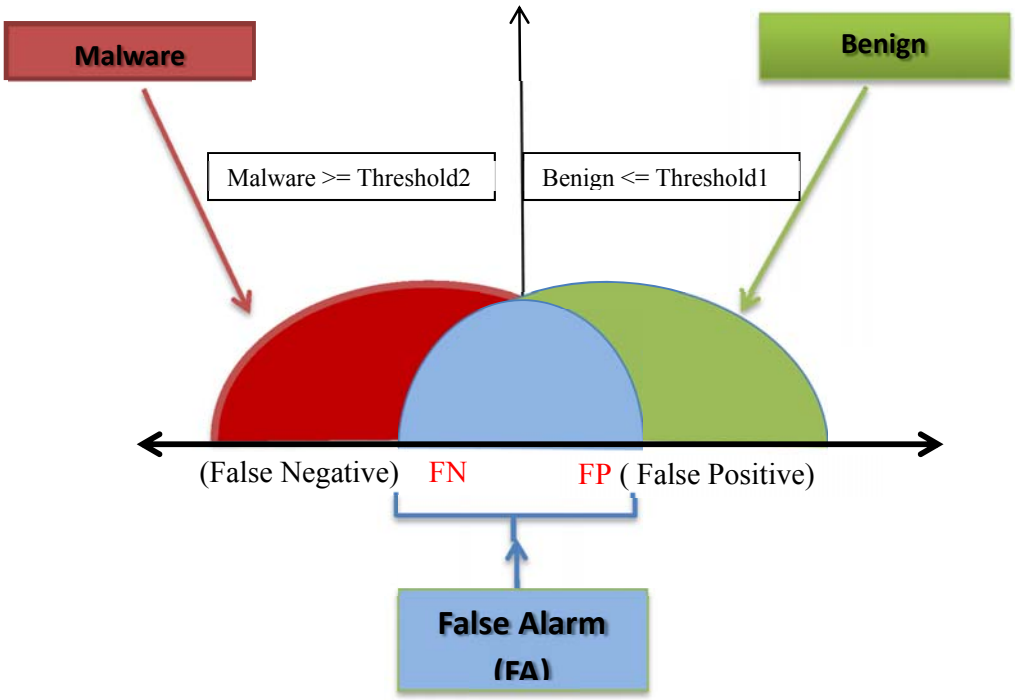

Figure No.(2):False Alarm, False Negative and False Positive.

تقليل التبليغات الكاذبة في تحديد واجهة برامجيات التطبيقات الضارة عند كشف البرمجيات الخبيثة باستعمال الشبكات العصبية مع التحليل المميز

عباس محسن البكري
كلية الرياضيات الحاسوب / جامعة الكوفة
حسين لفتة حسين
قسم علوم الحاسبات /كلية التربية للعلوم الصرفة (ابن الهيثم)/ جامعة بغداد

استلم البحث في : 7ايلول 2014 قبل البحث في: 24 تشرين الثاني 2014

الخلاصة

يناقش هذا البحث دقة أنماط السلوك المعتمدة في أنظمة الكشف والتي يتم تحليلها ورصدها بواسطة واجهات برمجة التطبيقات (API) .هذا العمل يحدد المشكلات التي تؤثر في دقة نماذج الكشف .تم استخراج 4744 (API) في هذا العمل عن طريق التحليل .وتتضمن هذه الطريقة الجديدة زيادة الدقة في كشف (API) الخبيثة في البرمجيات الخبيثة تصل إلى 83.2٪ .علما ان نتائج هذا العمل تم احتسابها و تقويمها باعتماد طبقة تحليل التمايز.

الكلمات المفتاحية: PE Malwares, Mali cious API, ANN