Malware Classification Based on the Behavior Analysis and Back Propagation Neural Network

Zhi-Peng PAN[1, a], Chao FENG[1] and Chao-Jing TANG[1]

[1]*College of Electronic Science and Engineering, National University of Defense Technology, 410073 Changsha, China*
[a]Corresponding author: climbspider@hotmail.com

Abstract. With the development of the Internet, malwares have also been expanded on the network systems rapidly. In order to deal with the diversity and amount of the variants, a number of automated behavior analysis tools have emerged as the time requires. Yet these tools produce detailed behavior reports of the malwares, it still needs to specify its category and judge its criticality manually. In this paper, we propose an automated malware classification approach based on the behavior analysis. We firstly perform dynamic analyses to obtain the detailed behavior profiles of the malwares, which are then used to abstract the main features of the malwares and serve as the inputs of the Back Propagation (BP) Neural Network model.The experimental results demonstrate that our classification technique is able to classify the malware variants effectively and detect malware accurately.

1 Introduction

Malicious software (Malware), usually in forms of virus, Trojans, worms, botnets, rootkits, and some other potentially unwanted applications, has been the major threat to the internet security. Malware developers use the hiding techniques such as polymorphism and obfuscation [1] to against signature-based on detection and static malware analysis methods easily and effectively [2]. In contrast to static analysis, dynamic analysis of malware based on monitoring its behavior during the run-time, which renders the malware more difficult to conceal [3, 4], and it does become the mainstream method of malicious behavior mining.

Yet these dynamic technologies for the malware detection are not sufficient just by forming detailed behavior profiles [5]. What we need is the ability to automatically categorize of the malware and detect the malware by its behavior.

The main contributions of this paper are as follows:

1)Unlike many previous algorithms that monitor the malware behaviors directly on low-level data such as API call monitoring [6], we implement an automatic dynamic analysis framework by taking the advantages of the present behavior analysis systems. We get the detailed behaviors of the malware including process behaviors, registry behaviors, file behaviors, net behaviors, and other behaviors.

2)We extract the major features of the malware behavior profiles into the behavior vectors by counting the quantity of the every behavior.

3)We proposed a Back Propagation (BP) Neural Network model [7] for learning the behavior patterns of the same categories of the malwares and classifying the malwares. The experimental section verified the correctness and precision of our algorithm finally.

2 Related Work

There are three major methods for classification of the malicious softwares, traditional pattern matching, static analysis, and dynamic analysis. Although static analysis can improve accuracy than the methods of the traditional pattern matching, it can also have the difficulty to handle obfuscated and self-modifying codes.

In dynamic analysis, Konrad Rieck [8] et al. proposed a method using the CWSandbox to analysis the behaviors of the malwares and then using the Support Vector Machines (SVM) for learning and classification. Forrest [6] proposed fixed-length sequence of N-gram recognition model based on system call. Syed Zainudeen Mohd Shaid [9] proposed a behavior-based technique to visualize malware behavior in the form of images. This method uses the different color to indicate the different API calls. By using the behavior images, it can be possible to visually identify malware variants of the same family. Guanghui Liang [10] et al. capture malware behaviors based on the Temu platform and proposed a weighted Jaccard similarity matching algorithm to classify the malware variants.

In summary, when dealing with the malware variants classification, behavior analysis is the most effective method. In this paper, we use the present effective

behavior analysis system to help analyse of the malware in contrast to just monitoring the API calls or the traces of the APIs as above.

3 Methodology

Classification of malware variants has been concerned by analysts in a long period [11-14]. Evolving malware generates a lot of variants and brings great challenges to analytical work. Although these variants change in the file format and appearance, there are still the same behavior patterns. For example, all variants of the Allaple worm acquire and lock of particular mutexes on infected systems [8]. Aiming to exploit these behavior patterns using machine learning techniques and propose a method which can classify the malware variants automatically based on their behaviors. An outline of our approach is given by the following basic steps:

1) Malware Data Acquisition. A corpus of malware binaries are obtained by collecting the upload suspicious files on the *Kafan Forum*. A Multi-Engine Online Virus Scan system *VirSCAN* is applied to identify the known malware instances.

2)Behavior Monitoring. Malware binaries are executed and monitored by the *HABO* behavior analysis system, which can generate detailed behavior reports.

3)Feature Extraction. Features reflect the behavior patterns, such as process created, foreign memory regions read, mutexes created, or registry key modified, are extracted from the analysis reports and used to map the malware behavior into a high-dimensional vector space.

4)Learning and classification. Back Propagation neural network model is applied to learning and training for the classification of the malwares.

3.1 Malware data acquisition

We have obtained up to 13600 unique samples, which are uploaded by the extensive users of the Kafan Forum, using for learning and subsequent classification. After obtaining the samples, we applied the online virus scan system VirSCAN to partition the malwares into common families, such as Adware, Potential Unwanted Application (PUA), Trojan/Downloader. Note that we chose the VirSCAN instead of one Unti-virus product, like Avira, Karpasky, to label the malware as the VirSCAN is multi-engine and we can chose the most of the result to label our sample. We selected 9 most common malware categories and one Non-Malware category on our samples. These families listed in Table 1 represent a broad range of malware categories such as Adware, PUA, and Trojans, and the Non-Malware category can be extend for malware detection directly in the future.

Table 1 Malware Families Labeled by The *Virscan* System

1. Adware(4352)	6. TR/Droppper(986)
2. PUA(1232)	7. TR/Spy(864)
3. Trojan(569)	8. Worm(562)
4. Tr/downloader(2524)	9. Win32(694)
5. Tr/Crypt(1023)	10. Non-Malware(794)

3.2 Behavior Monitoring and Feature Extraction

In this section, we use the online behavior analysis system, which called *HABO*, to monitor the samples' behavior. Like most of the other online behavior analysis systems, it can analyse the upload binaries and give you a detail behavior report about the malware. Note that our methodology is not bound to the *HABO* system; it can also be adapted to other behavior analysis systems.

Figure 1 shows a part of behavior report of one malware sample. It contains *five main aspects, process behaviours, file behaviours, register behaviours, net behaviours,* and *other behaviours*. Furthermore, these 5 main aspects contain 73 sub-behaviors which describe the behavior of the malware in detail.

```
Process Behavior
Behavior Describe:      created a new file process
Detail Information:
ImagePath = C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.231586.exe,  CmdLine = start
ImagePath = C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.231986.exe,  CmdLine = watch
Behavior Describe:      process exit
Detail Information:
N/A
Behavior Describe:      enum process
Detail Information:
N/A
Behavior Describe:      create local thread
Detail Information:
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.275966.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.276289.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.276618.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.276938.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.277254.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.277570.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.277890.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.278205.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.278520.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.278837.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.279152.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.279467.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.279783.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.280099.exe
C:\Documents and Settings\Administrator\Local Settings\%temp%\1459221646.280416.exe
```

Figure 1. Behavior Report of The Malware Sample

Although the reports show the detail information of the behavior of the malware, it can't be used for the BP neutral network model directly, which needs the vectorial data as the input. Hence we should extract the main features of the malwares' behavior from the reports firstly. The method used here is called frequency statistics method. The main steps are showed as follows:

1) Given all the sub-behaviors, we represent them at a particular sequence, such as, create local thread, enumerate process, create a new file process, and so on.

2) We made a count on the all sub-behaviors of the malware respectively. For example, the behavior showed in Figure 1, we can use [2, 1, 1, and 15,] to represent.

The figure 2 shows a detail behavior vector of one malware, and the number zero means that this malware doesn't have the behavior correspondingly.

```
11 2  0  2  2  2  0  0  0  0  16 12 0  16 13
16 7  5  2  3  6  0  0  0  0  2  3  2  4  2
4  3  3  0  0  3  3  2  13 3  2  0  0  0  0
0  0  2  4  15 0  3  2  0  12 12 4  16 2  0
0  0  0  0  0  0  0  0  0  0  0  0  0
```

Figure 2. A Detail Behavior Vector of One Malware

3.3 Learning and classification

3.3.1 Establish the Model

According to the previous description, the norm of our behavior vector is 73. And we can use the vector, r = [r1, r2, r3,…, r10] ri = 0 or 1, to represent the output result. Each element value of the output vector is 0 or 1, 0 represents the malware does not belong to the corresponding category while 1 represents the malware belong to the corresponding category.

Given the input vector and the output vector, we established a Back Propagation (BP) Neural Network, which includes one input layer, one hidden layer and one output layer. The input layer and the hidden layer both have 73 neurons and the output layer has 10 neurons. Additionally, the hidden and output neurons include adjustment factor a and b respectively. The connection weight between input layer and hidden layer, hidden layer and output layer is noted by $W_{ij}\left(1\le i\le 73,1\le j\le 73\right)$ and $V_{jl}\left(1\le j\le 73,1\le l\le 10\right)$. The network is showed in Figure 3.

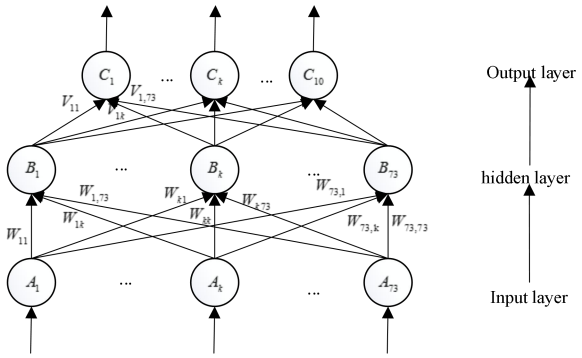

Figure 3. The Bp Neural Network

3.3.2 Identify the Parameter

We use the logistic function $f\left(x\right)=\frac{1}{1-e^{-x}}$ as the activation function of the hidden layer, and for convenience, the input of the network, hidden neuron h_j and output neuron O_l is denoted by $x_i\left(1\le i\le 73\right)$, $net\left(h_j\right)\left(1\le j\le 73\right)$ and $net\left(O_l\right)\left(1\le l\le 10\right)$ respectively. The output of the hidden neuron h_j and output neuron O_l is $out\left(h_j\right)$ and $out\left(O_l\right)$. We define the error signals of the output layer and the hidden layer as δ_{oh} and δ_{hi} respectively. Additional, $E=\frac{1}{2}\sum_{k=1}^{l}\left(\text{target}(O_l)-\text{out}(O_l)\right)^2$ is defined as output error. So according to the Back Propagation Neural Networks algorithm, we can derive the expression of the connection weight:

$$V_{jl}^{+}=V_{jl}+\eta\delta_{oh}\frac{\partial net(O_l)}{\partial V_{jl}}$$

* MERGEFORMAT (1)

$$W_{ij}^{+}=W_{ij}+\eta\delta_{hi}\frac{\partial net(h_j)}{\partial W_{ij}}$$

* MERGEFORMAT (2)

where $\delta_{oh}=-\frac{\partial E}{\partial out(O_l)}\cdot out'(O_l)$, $\delta_{hi}=-\frac{\partial E}{\partial out(h_j)}\cdot out'(h_j)$

and the symbol η means the learning rate of the network.

In this paper, the transform factor between the output layer and the hidden layer is denoted by a, and b means the transform factor between the hidden layer and the output layer.

1) The identification of the output function

Due to there is no experienced output function, we can only get the function by experiments. For the same samples, when trained 10000 times, we can get the results as shown in Table 2. And from the table, we finally choose the linear function $f\left(x\right)=1.0+x/5000$.

Table 2. Output Function and Some Results

Funtion	Convergence error	Whether fall into local minimum region
$f\left(x\right)=1/\left(1+\exp\left(-x\right)\right)$	5.69	Yes
$f\left(x\right)=0.8+x/5000$	0.78	No
$f\left(x\right)=1.0+x/5000$	0.12	No
$f\left(x\right)=1.2+x/5000$	2.34	No
$f\left(x\right)=1.0+x/4000$	1.56	No
$f\left(x\right)=1.0+x/6000$	0.87	Yes

2) The identification of the Adjustment factors

Due to the value of the adjustment factors a and b has the important influence on the speed of the convergence. Figure 4 and Figure 5 show us the relationships between the adjustment factors and the convergence time, when the total error is set to be 0.64.

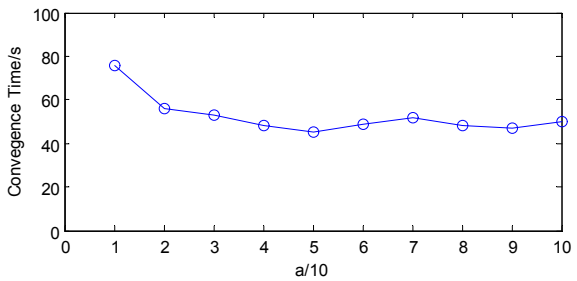

Figure 4. A and Convegence Time

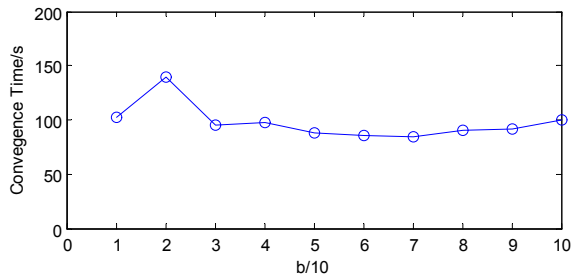

Figure 5. B and Convegence Time

4 Experiment and Evaluation

In order to evaluate the performance of our methodology, we firstly divided the malware corpus randomly into training and testing two partitions, and the samples sizes are 10000 and 3600 respectively. We used the training partition to train the BP neutral network, and used the testing partition to measure the overall performance of our methodology. Besides, the procedure showed above is repeated over five independent experimental runs and we use the average values as our final results.

The per-category accuracy for this experiment is shown in Figure 6, and the error bars indicate the variance measured during the experiment runs. From the figure, we can find our average accuracy is up to 86%. And in particular, we can find the last category, which we defined it as the non-malware, whose predict accuracy is up to 99%. In other words, if we used this model to detect the binary in our corpus is whether the malware or not, we have the correct probability approximate to 99%. This result shows that our methodology can be easily extend for malware detection. And more deeply, due to the boundaries of categories 3,4,5,6 and 9 are less obvious, which labelled as Trojan, Tr/downloader, Tr/Crypt, Tr/dropper amd Win32, we find that the variance of these categories are higher than other categories.

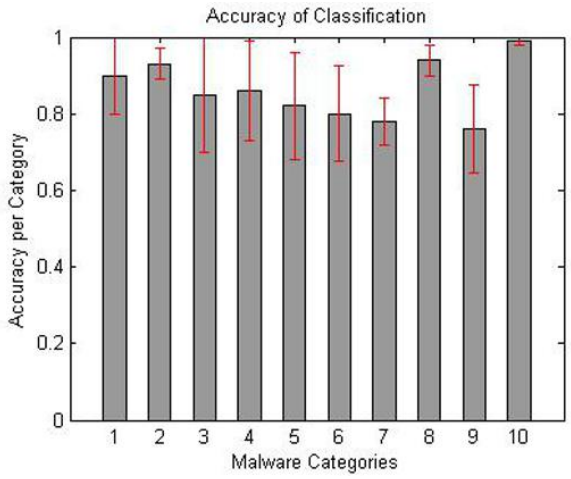

Figure 6. Accuracy per category

Figure 7 shows the confusion matrix for classification. If the color of the category is deeper, it means that this category is less error probability be classified into other categories. From the figure, we can find that the categories between 3 and 7 are easily be confused each other, and the category Win32 is most likely to be considered into other categories, which are consistent with our actual situation.

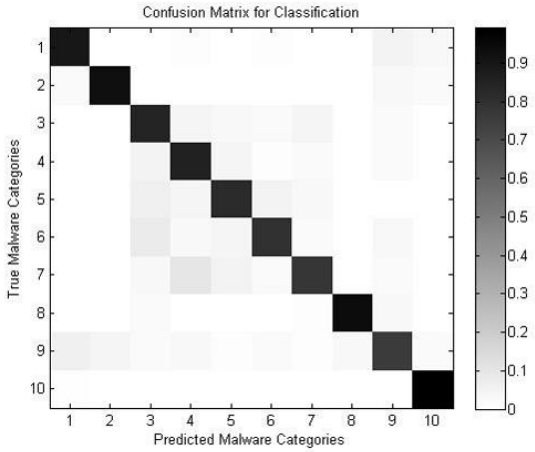

Figure 7. Confusion of Categories

5 Conclusion

In this paper, the behavior of the malware is captured by the online behavior analyze system. After that, we extracted the main feature of the malware in forms of vector and serve it as the input of out Back Propagation Neural Network model. Finally, by training the BP Neural Network, we can use it to classify the malware and detect the malware. Experimental results show that our methodology can classify the malware variants effectively and detect the malware accurately.

In the future work, we will focus on how to extracted malware feature that can represent the malware more accurately from the behavior analysis reports.

References

1. Rad B B, Masrom M, Ibrahim S. Camouflage in malware: from encryption to metamorphism[J]. International Journal of Computer Science and Network Security, 2012, 12(8): 74-83.
2. Moser A, Kruegel C, Kirda E. Limits of static analysis for malware detection[C]//Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual. IEEE, 2007: 421-430.
3. Willems C, Holz T, Freiling F. Toward automated dynamic malware analysis using cwsandbox[J]. IEEE Security & Privacy, 2007 (2): 32-39.
4. Egele M, Scholte T, Kirda E, et al. A survey on automated dynamic malware-analysis techniques and tools[J]. ACM Computing Surveys (CSUR), 2012, 44(2): 6.
5. Bayer U, Comparetti P M, Hlauschek C, et al. Scalable, Behavior-Based Malware Clustering[C]//NDSS. 2009, 9: 8-11.
6. Forrest S, Hofmeyr S, Somayaji A. The evolution of system-call monitoring[C]//Computer Security

Applications Conference, 2008. ACSAC 2008. Annual. IEEE, 2008: 418-430.
7. Irwin G W, Warwick K, Hunt K J. Neural network applications in control[M]. Iet, 1995.
8. Rieck K, Holz T, Willems C, et al. Learning and classification of malware behavior[M]//Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg, 2008: 108-125.
9. Shaid M, Zainudeen S, Maarof M A. Malware behavior image for malware variant identification[C]//Biometrics and Security Technologies (ISBAST), 2014 International Symposium on. IEEE, 2014: 238-243.
10. Liang G, Pang J, Dai C. A Behavior-Based Malware Variant Classification Technique[J]. International Journal of Information and Education Technology, 2016, 6(4): 291.
11. Park Y, Reeves D, Mulukutla V, et al. Fast malware classification by automated behavioral graph matching[C]//Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. ACM, 2010: 45.
12. Cesare S, Xiang Y. Classification of malware using structured control flow[C]//Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing-Volume 107. Australian Computer Society, Inc., 2010: 61-70.
13. Tian R, Batten L M, Versteeg S C. Function length as a tool for malware classification[C]//Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on. IEEE, 2008: 69-76.
14. Islam R, Tian R, Batten L, et al. Classification of malware based on string and function feature selection[C]//Cybercrime and Trustworthy Computing Workshop (CTC), 2010 Second. IEEE, 2010: 9-17.