ISBN 978-981-11-0008-6
Proceedings of 2016 6th International Workshop on Computer Science and Engineering (WCSE 2016)
Tokyo, Japan, 17-19 June, 2016, pp. 110-114
doi: 10.18178/wcse.2016.06.019

Automatic Analysis of Malware Behavior with SVM

Xiang Jing, Biao Qi, Jianguo Jiang, Bin Lv [+]

Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

Abstract. Malicious binaries prevail in the networked systems, such as computer viruses, Trojan horses, and Internet worms cause a brief threat to the security of computer networks. At the beginning of analyzing malware, static malware analysis methods employ byte-level content to detect malware binaries; however, polymorphism and obfuscation make static malware analysis methods be out of function. Dynamic analysis is more suitable than static analysis to recognize malware for monitoring malware behaviors which are vital to identify and difficult to conceal. Actually homogeneous malware families share similar behavioral features and heterogeneous malware families have different characteristics. We propose a method to classify malware families based behavior. Our method includes two parts: (1) we transform raw malware data into vector space using bag of word model. (2) We employ support vector machine (SVM) to classify malware data into correspondent families. The result shows that the accuracy of algorithm is over 90%.

Keywords: dynamic analysis, bag of word model, SVM.

1. Introduction

Various malware such as computer viruses, Trojan horses, and Internet worms poses a brief threat to the security of computer networks, affects the normal operation of the Internet. Moreover, proliferation and deformation of malware make the computer network at risk seriously [1]. Keeping antivirus products up-to-data continually seems to ensure security of computer network, however, Profit-driven persons or firms will make computer viruses updates faster than anti-virus software. Besides, malware writers usually make their efforts to stop detection by hiding techniques, tens of thousands of known malware variants from some malware family appear, and this leaves professional safety personnel a more daunting work to identify malware. So detecting and recognizing malware are still particularly complex and important.

Before anti-malware software relied on byte-content analysis [2] to confront malware. But this was easily defeated by malware writers by using obfuscation and polymorphism. And other static analysis methods were so [3]. Dynamic analysis of malware binaries based on behaviors is more successful compared to static analysis to distinguish malware, for the reason that malware shares intrinsic dangerous operational characteristics. That is also the reason why so many security researchers pay more attention to dynamic analysis.

In fact judging a computer file whether it is malware or not is difficult for some procedures, moreover, we intend to point out it belongs to which malware families. First of all, we need to run it in a controlled environment such as virtual machine, thus we can collect relevant data and then must gain some important behaviors such as open file or move file. Recently a general approach to complete it is make use of CWSandbox to record all system-level behavior by monitoring the performed system calls, in this way a behavior report created concludes the whole operational view about the file [4]. More precisely, a large amount of malware was massively produced by machines, so nearly every malware belongs to different malware families, and we can classify majority malware into corresponding families by analysis its dynamic behavioural essence.

[+] Corresponding author. Tel.: + 8618911017537.
E-mail address: lvbin@iie.ac.cn.

Our main work in this paper is to employ a popular machine learning method to classify malware into relevant malware families. The difficulties for us include data representations and algorithm analysis, more specifically, we must transform XML-based forms created by CWSandbox into vector space for our algorithms are based on distance metric and numeric analysis and non-numeric data such as system call instructions and other strings can't make a comparison to submit to the ideology that samples in same class own smaller distances while in different class have bigger distances.

This paper is organized as follows: related work is introduced in Section 2, and Section 3 briefly elaborates our approach. Our experiments are presented in Section 4, finally, we conclude our paper in Section 5.

2. Related Work

In recent years, dynamic malware analysis based behaviors are more and more attracting people's attention. In 2008, Konrad proposed a supervised learning method to learn and classify malware based behaviors, its prominent point was comprehensively expounding the whole analysis process from data acquisition to classification explanations [5]. Bailey et al. firstly employed clustering algorithms to consider program behavior of malware for identifying related samples [6]. Gu et al. correlate spatial-temporal relations in botnet communication using [7]. To some extent, dynamic analysis based behaviors with clustering algorithms can help find new malware families, and it is essential to tag new malware families to fight them at early stage. Besides, we don't know malware belongs to which families in the beginning; it deserves to cluster malware binaries into different clusters. In addition, classification algorithms also play effective roles in recognizing different malware, for the best posterior is obtained by training a training patterns set, and they have a good generation performance. While classification methods need sample labels compared to clustering methods [8]. Actually to deal with the increasing amount and diversity of malware, the combination of dynamic analysis and machine learning techniques based statistic characters will be the research focus, so many literatures have studied these in different scenarios [9], [10].

More specifically, there are two stages to process malware classification. Firstly we need to transform XML analysis reports into a suitable and identifiable form. Philipp Trinius et.al. proposed an instruction set for behavior-based analysis, in this way could change CWSandbox representations to a malware instruction set by hashing process [11]. The design of the instruction set is various and vital. Secondly, we can choose supervised or unsupervised machine learning if we get the above procedure, this have been many literatures studied [12], [13].

3. Our Approach

After the introduction of the previous section, we can have a rough understanding of dynamic malware analysis based behaviors. In this section we briefly expound our methodology. Firstly, we employ bag of word model [14] to transform XML form analysis reports into vector space. The bag of word model has been widely used in the field of text classification. Its main function is to convert a text sample to a vector and the feature in the vector represents the number of corresponding word in the primitive text. In this way we completely obtain a vector that can express equivalent XML file. Secondly, we employ linear LibSVM to classify malware files [15]. Our algorithms are as follows in the process:

We need to introduce LibSVM exhaustively. LibSVM was a simple and practical software package developed by Lin et al. [16]. It was mainly used to pattern classification and regression, and it supports parameters optimization framework and cross validation. LibSVM is an achievement of SVM algorithm and code optimization in practice. We then briefly recommend SVM algorithm. Because we obtain high dimension data after data representation, we just attempt to linear SVM without kernel SVM [17]. We mainly state binary classification here.

Binary classification is frequently performed by using a real-valued function $\mathrm{f}: \mathrm{X} \subseteq R^n \longmapsto R$ in the following way: $\mathrm{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$, will be assigned to the positive class, if $\mathrm{f(x)} > 0$, and otherwise to the negative. For f(x) is a linear function, we can write it as

$$f(x) = \omega^T x + b$$

$$= \sum_{i=1}^{n} \omega_i x_i + b$$

The primal model is

$$\min \frac{1}{2}(\| \omega \|)^2 + C\sum_{i=1}^{n} \xi_i$$

$$s.t.\ y(\omega^T x + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0; i = 1, 2, \dots, n$$
$$s.t.\ \sum_{i=1}^{n} \omega_i \alpha_i = 0$$

And the dual optimal problem is

$$\min \frac{1}{2} \sum_{i,j=0}^{n} y_i y_j \alpha_i \alpha_j x_i x_j - \sum_{i=1}^{n} \alpha_i$$
$$s.t.\ \sum_{i=1}^{n} \omega_i \alpha_i = 0$$
$$0 \leq \alpha_i \leq C, i = 1, 2, \dots, n$$

It is a quadratic programming and we can get decision function

$$g(x) = \mathrm{sgn}(\omega^T x + b)$$
$$= \mathrm{sgn}(\sum_{i=1}^{n} \alpha_i y_i x_i \cdot x + b)$$

As the Fig. 1 shows, from the start we gain malware analysis reports in XML form by CWSandbox, and then we transform XML files into vector space by bag of word model. When we get numerical vectors instead of malware, we can perform SVM to classify. It demonstrates our approach is feasible and effective through some experiments.

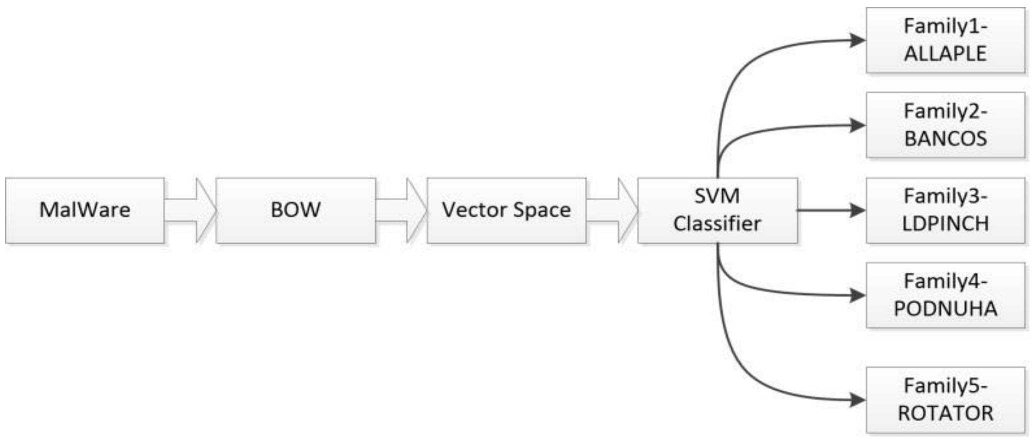

Fig. 1: Malware classification process flow diagram.

4. Experiment

4.1. Data Processing

Malware analysis or classification is based on malware data to a great degree, for different data feeds different algorithms, data is crucial. Especially in our case we should collect raw data using honey pot or spam-trap, and employ CWSandbox to acquire XML analysis reports, but we emphasize our effective approach. So we just make use of the XML file data in the literature [5], and the network link is http://pi1.informatik.unimannheim.de/malheur. We need using numeric type data, so we just extract several kinds of data and build new practical data set by the bag of word model as Table I.

Table I: Malware Dataset

Malware families name	Data size	Data Dimension

ALLAPLE	300	85
BANCOS	48	85
LDPINCH	209	85
PODNUHA	300	85
ROTATOR	78	85

4.2. Experiment

In our whole operation, we perform two experiments, firstly we use all data to take part in training and testing, and secondly we just make use of ALLAPLE, LDPINCH and PODNUHA to complete our simulations. The experiments are conducted on a PC with Intel Core i7-410U (2.00GHz) and RAM 8.00GB memory. The software is based on the LIBSVM-3.1 MATLAB Toolbox, which is available at http://www.matlabsky.com/thread-17936-1-1.html. The experiments procedures are as following Fig. 2. Through the foregoing management, we get numeric data, so we can extract training set, training labels and testing set, testing labels, in addition, we need to normalize our data to (0,1), then reduce data dimensions by 95%, and lastly we employ LibSVM to train the optimal model for ten-fold cross validation. Besides, we set the step size of grid search as 0.1, we perform our two experiments ten times and we use the average as the final results.

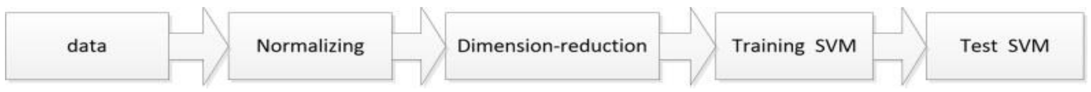

Fig. 2: Experiment procedures.

4.3. Experiment Results and Analysis

The previous section illustrates our specific experiment procedures, and this section mainly demonstrates our experiment results and analysis. We perform two experiments and the results show in the Table II.

Table II: Experiment Results

Malware Families Names	ALLAPLE	BANCOS	LDPINCH	PODNUHA	ROTATOR	Accuracy
Accuracy rate1(%)	92.95	56.67	82.4	97.44	63.83	86
Accuracy rate2(%)	88.57	#	95.45	89.47	#	90.23

In the first experiment, we perform the dataset training directly and the classification results of BANCOS and ROTATOR are not good. The reason of that will be thought of class imbalance from the amount of the data [18]. We get total accuracy of the first experiment according to the following formula:

$$\text{Accuracy} = \sum_{i=1}^{n} \frac{NP_i}{N}$$

N represents the total of samples and NP_i is the ith accuracy in the ith class sample. It stands for the number of malware families. The total accuracy of the first experiment is 86%. In order to keep away from class imbalance, we choose the ALLAPLE, LDPINCH and PODNUHA as the dataset, and we can get the total precision up to 90.23%. It turns out our approach is effective and feasible.

In this paper, we apply a supervised machine learning method to classify malware families, our method has two procedures: in the first process, we use bag of word model to transform XML form files obtained by CWSandbox into vector space, and we get whole numerical type malware data; in the second process, we employ SVM to train the optimal model and then test the testing samples, through experiment simulations, it demonstrates our classification precision is up to 90.23%. However, this work just is our preliminary attempt to malware analysis, and we hope that afterwards we can thoroughly connect malware features to actual and specific behaviors and furthermore, we can recognize attacks behaviors and objectives of malware. Those will be our subsequent work.

5. Conclusion

In this paper, we apply a supervised machine learning method to classify malware families, our method has two procedures, in the first process, we use bag of word model to transform XML form files obtained by CWSandbox into vector space, and we get whole numerical type malware data; in the second process, we employ LibSVM to train the optimal model and then test the testing samples, through experiment simulations, it demonstrates our classification precision is up to 90.23%. We hope that afterwards we can thoroughly connect malware features to actual and specific behaviors and furthermore, we can recognize attacks behaviors and objectives of malware. Those will be our subsequent work.

6. References

[1] Microsoft, “Microsoft security intelligence report (sir),” Microsoft Corporation, Tech. Rep., 2009.

[2] M. E. Karim, A. Walenstein, A. Lakhotia, and L. Parida, “Malware phylogeny generation using permutations of code,” Journal in Computer Virology, vol. 1, no. 1-2, pp. 13–23, 2005.

[3] M. Christodorescu and S. Jha, “Static analysis of executables to detect malicious patterns,” DTIC Document, Tech. Rep., 2006.

[4] C. Willems, T. Holz, and F. Freiling, “CWSandbox: Towards automated dynamic binary analysis,” IEEE Security and Privacy, vol. 5, no. 2, pp. 32–39, 2007.

[5] K. Rieck, T. Holz, C. Willems, P. D¨ussel, and P. Laskov, “Learning and classification of malware behavior,” in Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2008, pp. 108–125.

[6] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, “Automated classification and analysis of internet malware,” in Recent advances in intrusion detection. Springer, 2007, pp. 178–197.

[7] G. Gu, J. Zhang, and W. Lee, “Botsniffer: Detecting botnet command and control channels in network traffic,” 2008.

[8] M. R. Anderberg, Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks. Academic press, 2014, vol. 19.

[9] K. Rieck, P. Trinius, C. Willems, and T. Holz, Automatic analysis of malware behavior using machine learning. TU, Professoren der Fak. IV, 2009.

[10] F. Ahmed, H. Hameed, M. Z. Shafiq, and M. Farooq, “Using spatiotemporal information in api calls with machine learning algorithms for malware detection,” in Proceedings of the 2nd ACM workshop on Security and artificial intelligence. ACM, 2009, pp. 55–62.

[11] P. Trinius, C. Willems, T. Holz, and K. Rieck, “A malware instruction set for behavior-based analysis,” 2009.

[12] R. Gentleman and V. Carey, “Unsupervised machine learning,” in Bioconductor Case Studies. Springer, 2008, pp. 137–157.

[13] Y.-H. Wang, Y. Li, S.-L. Yang, and L. Yang, “Classification of substrates and inhibitors of p-glycoprotein using unsupervised machine learning approach.” Journal of chemical information and modeling, vol. 45, no. 3, pp. 750–757, 2005.

[14] C.-F. Tsai, “Bag-of-words representation in image annotation: A review,” ISRN Artificial Intelligence, vol. 2012, 2012.

[15] C.-C. Chang and C.-J. Lin, “Libsvm: A library for support vector machines,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, p. 27, 2011.

[16] C. Chang and C. Lin, “Libsvm: a library for support vector machines (2001), software,” 2005.

[17] V. N. Vapnik and V. Vapnik, Statistical learning theory. Wiley New York, 1998, vol. 1.

[18] R. Longadge and S. Dongre, “Class imbalance problem in data mining review,” arXiv preprint arXiv: 1305.1707, 2013.