Machine Learning for Analyzing Malware

Zhenyan Liu, Yifei Zeng, Yida Yan,
Pengfei Zhang and Yong Wang

Beijing Key Laboratory of Software Security Engineering Technology,
School of Software, Beijing Institute of Technology,
Beijing 100081, China
E-mail: zhenyanliu@bit.edu.cn; 453624629@qq.com;
1415654264@qq.com; zhpngfei@163.com; wangyong@bit.edu.cn

Received 17 November 2017; Accepted 22 November 2017;
Publication 15 December 2017

Abstract

The Internet has become an indispensable part of people's work and life, but it also provides favorable communication conditions for malwares. Therefore, malwares are endless and spread faster and become one of the main threats of current network security. Based on the malware analysis process, from the original feature extraction and feature selection to malware analysis, this paper introduces the machine learning algorithms such as classification, clustering and association analysis, and how to use these machine learning algorithms to effectively analyze the malware and its variants.

Keywords: Malware analysis, Machine learning, Classification, Clustering, Association analysis.

1 Introduction

Malicious software, or *malware*, plays a part in most computer intrusions and security incidents. Any software that does something that causes harm to a user, computer, or network can be considered malware, including viruses, trojan horses, worms, rootkits, scareware, and spyware [1].

Journal of Cyber Security, Vol. 6_3, 227–244.
doi: 10.13052/jcsm2245-1439.631

In recent years, due to the widespread of malwares in the network, the number of network security incidents is increasing year by year, the relevant statistics show that the number of network security incidents caused by malwares increased by more than 50% per year from the 1990s [2]. These network security incidents not only reflect the vulnerability of system and network security, but also lead huge losses to the current development based on Internet infrastructure. They may cause economic losses to the network users and businesses, even lead to network paralysis.

There are two fundamental approaches to malware analysis: static and dynamic. *Static analysis* involves examining the malware without running it, while *Dynamic analysis* involves running the malware in virtual or real environment [1]. Both approaches analyze the malware based on different features. *Static analysis* uses internal code features and *Dynamic analysis* uses external behavior features on the system or in the network. *Static analysis* is straightforward and can be quick, but it can miss important behaviors. *Dynamic analysis* can capture some important behaviors, but it won't be effective with all malware. Therefore, it is a more common solution how to use *static analysis* together with *dynamic analysis* in order to completely analyze suspected malware.

The number of current malwares is very large and new malwares appear faster and faster, thus traditional detection technology have been unable to cope with the current malware detection because of their speed and efficiency. On the other hand, the traditional method has high maintenance costs, and need a large number of manual experience to carry out sample analysis and extract the rules [3]. In recent years, machine learning for analyzing malware has been widely recognized, which can effectively make up the traditional methods [4–7].

In particular, we first present the framework for analyzing malware by machine learning in Section 2, which is an infrastructure for our review of analyzing malware by machine learning. And then, in Section 3, we provide a critical review of the innovative research developments targeting the malware analysis problem by machine learning, including classification, clustering, and association rules. Finally, we make a prospect on the focus of future research of malware analysis by machine learning in Section 4.

2 The Framework for Analyzing Malware by ML

Machine learning for analyzing malware is still a newcomer, but it has already obtained the huge success in fact. Commonly used machine learning algorithms are clustering, classification and association analysis and so

on. The classification technology can not only be used to detect unknown malware before starting its malicious behavior, but also to label malware family according the exact or fuzzy features of malware [8]. Clustering and association analysis are more useful in how to make family decisions and homology analysis of unknown samples, thus to increase the speed of handling or improve the efficiency of manual analysis.

The framework for analyzing malware by machine learning is shown in the Figure 1.

In this framework there are two subsequent phases that the first phase is to obtain the features of the malware including internal code features and external behavior features on the system or in the network, and then the second phase is to build a machine learning model for analyzing the malware based on these features of the malware.

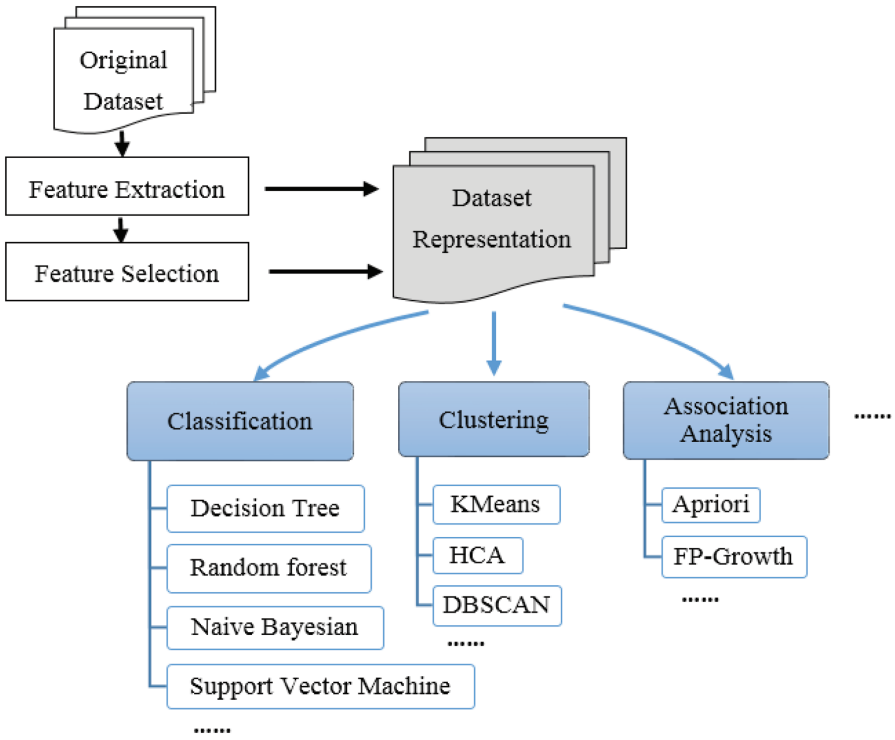


Figure 1 The framework for analyzing malware by machine learning.

The quality of the feature has a greater impact on the performance of the machine learning model, so the first phase in this framework is very critical [9]. In this phase, feature acquisition is generally divided into two steps, one is *original feature extraction*, and the other is *feature selection*. But according to different practical task requirements, sometimes the *original feature extraction* is only used, and sometimes the *feature selection* is also incorporated into this phase.

The main task of the *original feature extraction* is to extract various types of features. Many studies focused on internal code features of the malware such as byte n-grams [5, 6], OpCode [10–12] and PE (Portable Executable) features [13, 14]. In addition, a few researchers utilized external behavior features on the system such as creating file, hiding service, opening port [15], and some works also devoted to employ external behavior features in the network such as DNS (Domain Name System) answer and TTL (Time To Live) value [16].

However, many features from *original feature extraction* are generally redundant, many of which not contributing to build machine learning model (or even degrading its performance). So it is very necessary to select the most important and indispensable features (i.e. *feature selection*). *Feature selection* aims to select a subset of features which are the best for building machine learning model [17]. There are various *feature selection* methods employed in malware detection field, including Document Frequency [18], Information Gain [19], and chi-square [20].

In Kaggle [21], feature engineering was a very key part, which directly determined the game's performance. In Kaggle 2015, we can see some first-class participants used the frequency of the PE header file, the designated dll, opcode, call to select features, it's also worth mentioning that the third place used the L1 regularization of SVM (Support Vector Machine) to remove roughly some irrelevant features, and then utilized the *feature_importances* parameter of Random Forest to make a better choice.

Based on the features obtained, the original samples are represented, and then the representative vectors are the input for a learning model. Next step is the second phase in this framework which is to build a machine learning model, such as classification, clustering and association analysis. In the following section, we will introduce in detail the leading algorithms related to machine learning for analyzing malware.

3 Machine Learning Algorithms

Various machine learning algorithms, including classification, clustering and association analysis have been commonly used for analyzing malware. The most popular classification algorithms are Decision Tree, Random forest, Naive Bayesian, Support Vector Machine, etc. In clustering, we often see KMeans, HCA (hierarchical cluster analysis) and DBSCAN (Density-Based Spatial Clustering of Applications with Noise). In association analysis, Apriori and FP-Growth (Frequent Pattern-Growth) are frequently used.

3.1 Classification

Classification is a supervised technique which is usually divided into two phases: The first phase is to train a classifier using classification algorithm based on the labeled samples, and then the subsequent phase applies this classifier to assign a class to each new data item [22]. The training data and the test data are represented on the basis of the same feature space. A number of classification algorithms applied to analyze malware can be found in the literature. In this section, we will introduce these classification algorithms in detail.

3.1.1 Decision tree

Decision tree algorithms construct a model of decisions made based on actual values of attributes in the data. Each non-leaf node in the tree (including the root node) corresponds to a test of a non-category attribute in the training sample set, and a test result for each branch of the non-leaf node, and each leaf node represents a class or class distribution. A path from the root node to the leaf node forms a classification rule [23]. Decision tree algorithm has many variants, such as ID3 (Iterative Dichotomiser 3) and C4.5, but the basis is similar. C4.5 algorithm has been used more in the field of malware detection.

Zhu et al. [24] proposed a new method for detecting unknown malwares under the Windows platform: the API (Application Program Interface) function dynamically invoked with PE file as the research object, using the sliding window mechanism to extract the feature, and adopt the decision tree C4.5 Algorithm to detect unknown malwares. The results showed that the decision tree C4.5 was higher than the minimum distance classifier and the Naive Bayesian algorithm, and the stability of C4.5 was better than the other two algorithms.

Zhao et al. [16] developed a new system to detect APT (Advanced Persistent Threat) malware which relied on DNS to locate command and control servers. In this system, a classifier of *Malicious DNS Detector* was trained by a series of feature related DNS, which employed J48 decision tree algorithm and obtained the true positive rate of 96.3% and the false positive rate of 1.7%.

Perdisci et al. [25] presented a novel system called FluxBuster for detecting and tracking malicious flux network via large-scale passive DNS analysis. The system, using hierarchical clustering algorithm to group the domains and C4.5 classifier to classify these clusters, was capable of accurately detecting previously unknown flux networks days or weeks in advanced before they appear in blacklists.

3.1.2 Random forest

The Random Forest is an ensemble learning method that constructs a multitude of decision trees and outputs a prediction that is the mode of the classes of the individual trees. A subset of the training dataset (local set) is chosen to grow individual trees, with the remaining samples used to estimate the goodness of fit. Trees are grown by splitting the local set at each node according to the value of a random variable sampled independently from a subset of variables [26].

Tian et al. [27] presented a malware classification technique based on string information using several well-known classification algorithms, including Random Forest, IB1(Instance Based 1), AdaBoost. The experiments showed that the IB1 and Random Forest classification methods were the most effective for this domain.

Zhao et al. [28] defined the features according to both the statistical information and the topology of the function call graph, and used the J48, Bagging, Random Forest and other algorithms to detect viruses, Trojans and worms. The comparative experiments showed that Random Forest was better than other algorithms with 96% accuracy.

Shabtai et al. [29] used the DF method to select different quantities of OpCode N-grams as a feature set, and then used Random Forest, ANN(Artificial Neural Network), SVM, Naive Bayesian and other methods, finally reported that Random Forest was the best algorithm with more than 95% correct rate.

3.1.3 Naive Bayesian

The Naive Bayesian classifier is based on Bayes' theorem, which provides a way of calculating the posterior probability, and works on assumption called class conditional independence. A Naive Bayesian model is easy to build, but often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Zhu et al. [30] proposed a parameter valid window model for system call which improved the ability of operation sequence to describe behavior similarity. On this basis, they presented a malware classification approach based on Naive Bayesian and parameter valid window. The experiment results showed that this approach was effective, and the performance and accuracy of training and classification were improved through parameter valid window.

Sayfullina et al. [31] presented a scalable and highly accurate method for malware classification based on features extracted from APK (Android application package) files. They explored several techniques for tackling independence assumptions in Naive Bayes and proposed Normalized Bernoulli Naive Bayes classifier that resulted in an improved class separation and higher accuracy.

Passerini et al. [32] developed a system named FluXOR which could detect and monitor fast-flux service networks. FluXOR was divided into three components: collector, monitor and detector. The detector fed the set of collected features of the suspicious hostname to the naive Bayesian classifier for the classification.

3.1.4 Support vector machine

Built on structural risk minimization and VC dimension theory of Statistical Learning, the basic classification idea of SVM is to determine the optimal separating hyperplane, and to find this hyperplane can be converted to solve an equivalent quadratic optimization problem. The result of the optimization process is a set of coefficients to determine the optimal separating hyperplane.

Li et al. [33] studied a malware detection scheme for Android platform using an SVM-based approach, which integrated both risky permission combinations and vulnerable API calls and used them as features in the SVM algorithm. The experiments showed that the proposed malware detection scheme was able to identify malicious Android applications effectively and efficiently.

Mcgrath et al. [34] presented a method of examining fast flux, DNS flux and double flux in the phishing context through classification algorithm that Support Vector Machines, using the features such as the number of IP address and associated ASN that collected various information from each URLs.

By analyzing the patterns of the DNS queries from the fast-flux botnets, Yu et al. [35] extracted six features for constructing the weighted SVM in order to distinguish the normal network domain access from the fast-flux botnet domain access. The evaluation suggested that the approach was effective in detecting the fast-flux botnets.

3.2 Clustering

Clustering is an unsupervised technique which can group data using some measure of inherent similarity between instances, in such a way that samples in one cluster are very similar (compactness property) and samples in different clusters are different (separateness property). After the clusters are created, a labeling phase takes care of annotating each cluster with the most descriptive label [36]. Numerous clustering algorithms applied to analyze malware are available in the literature. In this section, we will introduce these clustering algorithms in detail.

3.2.1 KMeans

KMeans is very popular for cluster analysis. KMeans aims to partition n samples into k clusters in which each sample belongs to the cluster with the nearest mean, serving as a prototype of the cluster. KMeans is very simple and can be easily implemented in solving many practical problems. There exist a lot of extended versions of K-Means such as XMeans [37].

Sharma [38] grouped the executables on the basis of malware sizes by using Optimal KMeans Clustering algorithm and used these obtained groups to select promising features for training (Random Forest, J48, etc) classifiers to detect variants of malwares or unknown malwares.

Dietrich et al. [39] proposed a mechanism to detect DNS C&C (Command and Control) in network traffic and described a real-world botnet using DNS C&C. The mechanism used k-Means clustering and Euclidean Distance based classifier with the feature related DNS, which could obtain higher true positive rates.

Antonakakis et al. [40] proposed a novel detection system called Pleiades to identify DGA(Domain Generation Algorithm) bots through monitoring traffic below the local recursive DNS server and analyzing NXDomain

(Non-Existent Domain) responses. In Pleiades, Xmeans clustering algorithm was used to group domain subsets into larger clusters.

3.2.2 HCA

HCA is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for HCA generally fall into two types: Agglomerative and Divisive. The agglomerative HCA is a "bottom up" approach which each sample starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. And the divisive HCA is a "top down" approach which all samples start in one cluster, and splits are performed recursively as one moves down the hierarchy [41].

Perdisci et al. [42] presented a network-level behavioral malware clustering system using *single-linkage agglomerative hierarchical clustering*, which focused on HTTP-based malware and clustered malware samples based on a notion of structural similarity between the malicious HTTP traffic they generated.

Chatzis et al. [43] proposed a method that using *divisive hierarchical clustering* to divide machines into worm-infected machines and non-infected machines according to the similarities in their DNS communication patterns. And the FP rates and FN rates remained under 1% and 6% in overall email worm detection.

Based on DGA domains and NXDomain traffic, Thomas et al. [44] constructed a system that computed pairwise DNS similarity measures and subsequently performed *single-linkage agglomerative hierarchical clustering*. The resulting clusters grouped the DGA domains to distinctive clusters based on their DNS traffic and contained only domains relevant to that particular variant.

3.2.3 DBSCAN

DBSCAN is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away) [45].

To distinguish the variations of malicious code, Qian et al. [46] studied the malicious behavior of malwares, then computed the similarity of characteristics and the call graphs which were extracted by disassembly tools. They employed DBSCAN to discover the family of malicious code.

Schiavoni et al. [47] presented a mechanism called Phoenix based on the DBSCAN clustering algorithm, which could not only tell DGA- and

non-DGA-generated domains apart using a combination of string and IP-based features, characterize the DGAs behind them, but also find group of DGA-generated domains that are representatives of the respective botnets.

Xiao et al. [48] proposed a behavior-based malware cluster method inspired by the characteristic that Android malwares in a same family behaved similarly. With collected feature data, DBSCAN was employed to cluster malwares into various families. The evaluation results showed that the accuracy rate of malware clustering can reach 91.3% at best, and the correct prediction rate is 82.3% for evaluated samples.

3.3 Association Analysis

Finding implicit relationships between items from large-scale data set is called association analysis or association rule learning. Transactions may have a certain degree of law and relevance, malware is the same, between the behavior and family, there have a certain relationship, so you can tap the useful association rules for the malware analysis. There are many algorithms for mining association rules cited in the literature. In this section, we will introduce these algorithms in detail.

3.3.1 Apriori

Apriori is a classical algorithm which can mine frequent itemsets for Boolean association rules. Apriori employs an iterative approach known as a *level-wise* search, where k-itemsets are used to explore (k+1)-itemsets. To improve the efficiency of the *level-wise* generation of frequent itemsets, an important property called the Apriori property, i.e. all nonempty subsets of a frequent itemset must also be frequent, is used to reduce the search space [49].

Studying the rules of API calling sequence of some viruses and their variants in the implementation process, Zhang et al. [50] used Apriori algorithm to extract valuable association rules from the known virus API call sequence. And according to balancing the false positive rate and false negative rate, the best confidence threshold to guide the virus detection was acquired.

In [51], malware code was disassembled and preprocessed into sequential data, an Apriori-like algorithm was used to discover sequential pattern and remove normal pattern, and the result pattern set can be used to detect unknown malware. Experimental result showed that the method had high accuracy rate and low false positive rate.

Adebayo et al. [52] presented a classification system of Android malware using candidate detectors generated from Apriori algorithm improved with particle swarm optimization to train three different supervised classifiers. In this method, features were extracted from Android applications byte-code through static code analysis, selected and used to train supervised classifiers.

3.3.2 FP-Growth

The Apriori algorithm needs to scan the database once every iteration, generate a large number of candidate sets, spend a lot of time on the I/O, and the algorithm is inefficient. FP-growth algorithm [53] only needs to scan the database twice, which greatly speeds up the algorithm. Therefore, the application of FP-Growth algorithm in malware analysis is more extensive and in-depth.

In order to identify malicious web campaigns, Kruczkowski et al. [54] constructed a FP-SVM system which employed the FP-growth algorithm and the SVM method. This system could be successfully used to analyze a huge amount of dynamic, heterogenous, unstructured and imbalanced network data. In this system, the FP-growth algorithm was applied to produce the training dataset.

In [55], the frequent itemsets were obtained by the FP-Growth algorithm which extracted the URLs of the malware events, then mined the association rules from them, linked all the associated rules to the graphs, finally used the modular approach to divide the graph into different groups and analyzed a malware web site map.

In [56], a weighted FP-Growth frequent pattern mining algorithm was proposed for malicious code forensics. Different API call sequences were assigned different weights according to their threaten degree to obtain frequent patterns of serious malicious codes and more accurate analysis results. Compared with the original FP-Growth algorithm, the weighted algorithm can obtain higher accuracy when used for evidence analysis.

From the above analysis, we can see that all kinds of methods in different scenarios have their own advantages, and there are no absolute merits of the points. The performance of the algorithm often depends on the application of the scene, as well as the selected features, or the characteristics of the samples. And no algorithm can have any advantage in any application scenario, so the combination of the actual scene is the most critical. Moreover, it is worth mentioning that the original feature extraction and feature selection are very critical, which determine the final performance of the machine learning model.

4 Conclusion

Malware analysis has become a hot topic in recent years in network security. The emergence of plenty of new malwares makes the traditional analysis method no longer completely effective. How to extract the most representative features of malware, and further to maximize the speed and accuracy of malware analysis still need to intensive study. Therefore, it is necessary to use a more efficient and intelligent approach – machine learning, to detect and analyze unknown malwares.

The future development of malware analysis by machine learning will focus on three major research areas. Firstly, the existing machine learning algorithm should be improved and combination with specific application. Secondly, the improvement of the feature extraction and feature selection method should be considered to be expanded based on the semantic level, so that we can get more accurate features. The third more valuable research direction can be considered is to make a combination of feature and instance selection methods to study the scalability of malware database.

Acknowledgements

This work was financially supported by National Key R&D Program of China (2016YFB0801304).

References

[1] Sikorski, M., and Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. San Francisco, CA: no starch press.

[2] Liao, G., and Liu, J. (2016). A malicious code detection method based on data mining and machine learning. *J. Inf. Secu. Res.* 2, 74–79.

[3] Huang, H. X, Zhang, L., and Deng, L. (2016). Review of Malware Detection Based on Data Mining. *Computer Sci.* 43, 13–18.

[4] Lee, D. H., Song, I. S., and Kim, K. J., et al. (2011). A Study on Malicious Codes Pattern Analysis Using Visualization. In *International Conference on Information Science and Applications. IEEE Computer Society*, 1–5.

[5] Kolter, J. Z., and Maloof, M. A. (2006). Learning to Detect and Classify Malicious Executables in the Wild. *J. Mach. Learn. Res.* 6, 2721–2744.

[6] Schultz, M. G., Eskin, E., and Zadok, E., et al. (2000). Data Mining Methods for Detection of New Malicious Executables. Security and

Privacy, 2001. S&P 2001. In *Proceedings. 2001 IEEE Symposium*. IEEE, 38–49.

[7] Lai, Y. (2008). A Feature Selection for Malicious Detection. In *Ninth Acis International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/distributed Computing. IEEE Computer Society,* 365–370.

[8] Mao, M., and Liu Y. (2010). Research on Malicious Program Detection Based on Machine Learning. Software Guide, 9, 23–25.

[9] Domingos, P. (2012). A Few Useful Things to Know about Machine Learning. *ACM*, 55, 78–87.

[10] Karim, M. E., Walenstein, A., and Lakhotia, A., et al. (2005). Malware Phylogeny Generation Using Permutations of Code. *J. Computer Virology*, 113–23.

[11] Bilar, D. (2007). Opcodes as Predictor for Malware. *Int. J. Electronic Security & Digital Forensics*, 1, 156–168.

[12] Santos, I., Brezo, F., and Ugarte-Pedrero, X., et al. (2013). Opcode Sequences as Representation of Executables for Data-Mining-Based Unknown Malware Detection. Information Sciences, 231, 64–82.

[13] Perdisci, R., Lanzi, A., and Lee, W. (2008). Classification of Packed Executables for Accurate Computer Virus Detection. *Pattern Recognition Letters* 29, 1941–1946.

[14] Ding, Y., Yuan, X., and Tang, K., et al. (2013). A Fast Malware Detection Algorithm Based on Objective-Oriented Association Mining. *Computers & Security*, 39, 315–324.

[15] Lu, Y. B., Din, S. C., and Zheng, C. F., et al. (2010). Using multi-feature and classifier ensembles to improve malware detection. *J. Chung Cheng Institute of Technology*, 39, 57–72.

[16] Zhao, G., Xu, K., and Xu, L., et al. (2015). Detecting APT malware infections based on malicious DNS and traffic analysis. *IEEE Access,* 3, 1132–1142.

[17] Liang, C. (2012). Research on the Main Techonologies in Malware Code Detection. Yangzhou University.

[18] Moskovitch, R., Feher, C., and Tzachar, N., et al. (2015). Unknown Malcode Detection Using OPCODE Representation. In *Intelligence and Security Informatics, First European Conference, EuroISI 2008*, Esbjerg, Denmark, 204–215.

[19] Kolter, J., and Maloof, M. (2006). Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.* 7, 2721–44.

[20] Siddiqui, M., Wang, M. C., and Lee, J. (2008). Data Mining Methods for Malware Detection Using Instruction Sequences. In *IASTED International Conference on Artificial Intelligence* and Applications. ACTA Press, 358–363.
[21] http://www.kaggle.com/malware-classification
[22] Fang, Z. (2011). Research and Implementation of Malware Classification. National University of Defense Technology.
[23] Li, W. (2010). Research and Implementation of Mobile Customer Churn Prediction Based on Decision Tree Algorithm. Beijing University.
[24] Zhu, L. J., and XU, Y. F. (2013). Application of c4.5 algorithm in unknown malicious code identification. *J. Shenyang University Chemical Technol.* 27, 78–82.
[25] Perdisci, R., Corona, I., and Giacinto, G. (2012). Early detection of malicious flux networks via large-scale passive dns traffic analysis. *IEEE Transactions on Dependable & Secure Computing* 9, 714–726.
[26] Mistry, P., Neagu, D., Trundle, P. R., et al. (2016). Using random forest and decision tree models for a new vehicle prediction approach in computational toxicology. *Soft. Comput*. 20, 2967–2979.
[27] Tian, R., Batten, L., and Islam, R., et al. (2010). "An automated classification system based on the strings of trojan and virus families", in *International Conference on Malicious and Unwanted Software*. IEEE, 23–30.
[28] Zhao, Z., Wang, J., and Wang, C. (2013). An unknown malware detection scheme based on the features of graph. *Secu. Commun. Netw.* 6, 239–246.
[29] Shabtai, A., Moskovitch, R., and Feher, C., et al. (2012). Detecting unknown malicious code by applying classification techniques on opcode patterns. *Secu. Inf.* 11.
[30] Zhu, K., Yin, B., and Mao, Y., et al. (2014). Malware classification approach based on valid window and naive bayes. *J. Computer Res. Dev.* 51, 373–381.
[31] Sayfullina, L., Eirola, E., Komashinsky, D., Palumbo, P., Miche, Y., Lendasse, A., and Karhunen, J. (2015). Efficient detection of zero-day Android malware using normalized bernoulli naive bayes. In *Trustcom/BigDataSE/ISPA, IEEE*, 1, 198–205. IEEE.
[32] Passerini, E., Paleari, R., and Martignoni, L., et al. (2008). FluXOR: Detecting and monitoring fast-flux service networks. In *Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 186–206.

[33] Li, W., Ge, J., and Dai, G. (2015). Detecting malware for android platform: An svm-based approach. In *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference,* 464–469. IEEE.

[34] McGrath, D. K., Kalafut, A., and Gupta, M. (2009). Phishing infrastructure fluxes all the way. *IEEE Security & Privacy*.

[35] Yu, X., Zhang, B., Kang, L., and Chen, J. (2012). Fast-flux botnet detection based on weighted svm. *Inf. Technol. J.* 11, 1048–1055.

[36] Ceri, S., Bozzon, A., Brambilla, M., Della, V. E., Fraternali, P., and Quarteroni, S. (2013). Web Information Retrieval. Springer Berlin Heidelberg.

[37] Pelleg, D., and Moore, A. W. (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *ICML*, 1, 727–734.

[38] Sharma, A. (2015). Grouping the Executables to Detect Malwares with High Accuracy. In *International Conference on Information Security and Privacy*, 667–674.

[39] Dietrich, C. J., Rossow, C., and Freiling, F. C., et al. (2012). On Botnets that Use DNS for Command and Control. In *Seventh European Conference on Computer Network Defense*. IEEE, 9–16.

[40] Antonakakis, M., Perdisci, R., and Nadji, Y., et al. (2012). From Throw-away Traffic to Bots: Detecting the Rise of DGA-based Malware. In Usenix Conference on Security *Sympo*sium. 24–40.

[41] *Hierarchical Clustering*. Available at: https://en.wikipedia.org/wiki/Hierarchical_clustering

[42] Perdisci, R., Ariu, D., and Giacinto, G. (2013). Scalable fine-grained behavioral clustering of http-based malware. *Computer Netw.* 57, 487–500.

[43] Chatzis, N., Popescu-Zeletin, R., and Brownlee, N. (2009). Email worm detection by wavelet analysis of DNS query streams. In *Computational Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on* 53–60. IEEE.

[44] Thomas, M., and Mohaisen, A. (2014). Kindred domains: detecting and clustering botnet domains using DNS traffic. In *Proceedings of the 23rd International Conference on World Wide Web*, 707–712. ACM.

[45] Density-based spatial clustering of applications with noise (DBSCAN). Available at: https://en.wikipedia.org/wiki/DBSCAN

[46] Qian, Y., Peng, G., and Wang, Y., et al. (2015). Homology analysis of malicious code and family clustering. *Computer Engineering & Applications* 51, 76–81.

[47] Schiavoni, S., Maggi, F., Cavallaro, L., and Zanero, S. (2014). Phoenix: DGA-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* 192–211. Springer, Cham.

[48] Xiao, Y., Su, H., Qian, Y., and Peng, G. A., (2016). Behavior-based family clustering method for android malwares. *Journal of Wuhan University: Natural Science Edition*, 62, 429–436.

[49] Han, J., and Kamber, M., (2011). *Data Mining Concept and Techniques*. Elsevier.

[50] Zhang, W., Zheng, Q., and Shuai, J. M., et al. (2008). New malicious executables detection based on association rules. *Computer Eng*. 34, 172–174.

[51] Wang, X. Z., Sun, L. C., and Zhang, M., et al. (2011). Malicious Behavior Detection Method Based on Sequential Pattern Discovery. *Computer Eng*. 37, 1–3.

[52] Adebayo, O. S., and AbdulAziz, N. (2014). Android malware classification using static code analysis and apriori algorithm improved with particle swarm optimization. In *Information and Communication Technologies (WICT), 2014 Fourth World Congress* 123–128. IEEE.

[53] Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *ACM Sigmod Record* 29, 1–12. ACM.

[54] Kruczkowski, M., Niewiadomska-Szynkiewicz, E., and Kozakiewicz, A. (2015). FP-tree and SVM for Malicious Web Campaign Detection. In *Asian Conference on Intelligent Information and Database Systems* 193–201. Springer, Cham.

[55] Li-xiong, Z., Xiao-lin, X., Jia, L., Lu, Z., Xuan-chen, P., Zhi-yuan, M., and Li-hong, Z. (2015). Malicious URL prediction based on community detection. In *Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC), 2015 International Conference* 1–7. IEEE.

[56] Li, X., Dong, X., and Wang, Y. (2013). Malicious code forensics based on data mining. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2013 10th International Conference* 978–983. IEEE.

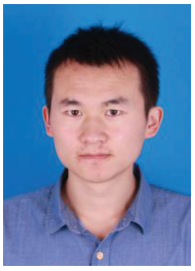
Biographies

Zhenyan Liu works at School of Software, Beijing Institute of Technology. She received her Ph.D. degree in Computer Architecture from Institute of Computing Technology Chinese Academy of Sciences, China. Her current research interests include big data, artificial intelligence, and cyber security. She's a fellow of the China Computer Federation (CCF).

Yifei Zeng is a M.Sc. student at Beijing Institute of Technology since autumn 2017. He attended the Guangdong Ocean University of Zhanjiang, China where he received his B.Sc. degree in Software Engineering in 2016. His M.Sc. work focuses on cyber security and artificial intelligence.

Yida Yan is a M.Sc. student at Beijing Institute of Technology since autumn 2016. She attended the Hebei Normal University, China where she received

her B.Sc. degree in Software Engineering in 2015. Her M.Sc. work focuses on data mining and cyber security. She has obtained Data Mining Senior Engineer Certificate, in China.



Pengfei Zhang is a M.Sc. student at Beijing Institute of Technology since autumn 2017. He attended the Nanjing University of Science & Technology, China where he received his B.Sc. degree in Computer Science in 2015. He has been a member of TP-Link from 2015 to 2016. His M.Sc. work focuses on cyber security and artificial intelligence.

Yong Wang works at School of Software, Beijing Institute of Technology. She received her Ph.D. degree in Computer Science from Beijing Institute of Technology, China. Her current research interests include cyber security and machine learning. She's a fellow of the China Computer Federation (CCF).