Acta Technica *62*, No. 3B/2017, 371–380 © 2017 Institute of Thermomechanics CAS, v.v.i.

Android malware detection method based on system call sequence feature weighting[1]

Xueli Hu[2], Shuai Yi[2], Zhenxing Wang[2], Liancheng Zhang[2]

Abstract. The amount of information carried by smartphones grows tremendously, which makes them one of the attack targets. The malware detection technique intended for smartphones has become a research hotspot. Feature selection is an important process in both supervised and unsupervised learning. However, many malware detection studies tend to perform their feature extraction based on very limited size samples, which can easily lead to the selection of unbalanced samples. Nonetheless, the unbalanced samples problem in malware detection techniques has not raised enough awareness and attention yet. In this paper, we propose an approach that studies the system call sequence samples generated by both malicious and normal and uses TF-RF relevance category feature weighting approach to extract features. The proposed approach can effectively retain the majority classes features (positive classes) and also has a good features classification capability for minority classes (rare classes), which improves malware detection accuracy.

Key words. Android, DF, malware, SVM, TF-RF.

1. Introduction

With increasing popularity of smartphones, malware which have been targeting PCs for several decades have quietly extended their scope to them. The malwares have caused severe damages to smartphones which greatly affected their users. Due to the smartphones fragmentation and compatibility problems, and malware techniques such as transformation, encryption shell and anti-analysis, malware detection is a very challenging task.

Previous studies show that if an application with malicious behavior wants to per-

[1]This research was performed under the project: Studies on Active Flow Association Technique Based on Package Program Flow Watermarking, supported by the National Nature and Science Fund (Fund No. 61402526).

[2]State Key Laboratory of Mathematical Engineering and Advanced Computing, Henan, 45000, China; e-mail: success_receive@hotmail.com

form its tasks, it must request the relevant services after being executed. If system calls are monitored and behavior trails of the entire lifecycle system calls are acquired, then system calls can be analyzed and the behaviors can be examined. Thus it can be concluded whether the codes are malicious or not. References [1–5] selected system calls as features for malwares detection. And some other related research results extract different features as features for malwares detection [6–8].However, in [3] only the frequencies of a single system call are considered, whereas the dependencies between certain system calls and their orders are not considered. Reference [4] only considered part of system call events during the features selection, so the detection is not comprehensive.

Due to the rapid development of malware, malware sample set update is not timely, leading to obsolete samples, easy to lead to training phase of unbalanced data sample sets problem. Malware detection is a typical unbalanced sample set classification problem. Unbalanced data means that some classes have very few instances in the dataset (i.e., minority classes), while other classes have many instances (i.e., majority classes). According to Gary M Weiss [9], unbalanced data set can cause a series of problems, such as data sparseness, data fragmentation, and inductive deviation. These problems might reduce the performances of the traditional classification methods. To improve the unbalanced data sets and improve the system efficiency, [10] used sparse matrices extracted from local singular value decomposition in order to reduce the system load. But singular value decomposition cannot reduce the impact of the unbalanced feature selection on the detection results.

Aiming at the above research problems, this paper proved a method. The paper contributions of research presented in this paper are as listed below:

1. Proved an approach based on two anti-debugging techniques, the App self-attachment and the dynamic process additional state, is proposed. the method tracks all the system call sequences of the App progress and obtains the system call traces of the entire lifecycle.

2. Most of the existing Android malware detection techniques are based on occurring frequency of each individual system call, whereas the dependencies between multiple system calls are neglected. However, that shortcoming is compensated in the proposed approach by sorting out of some implicit features from the N-Gram data sets. Experimental results have demonstrated that these features are highly effective in malware detection.

3. The TF-RF relevance category feature weighting method is adopted. Through feature computation and inter-class correlation based on features selection, the approach can effectively select as many beneficial minority class features as is possible, while maintaining the majority class features.

2. System architecture

According to the fact that the malware uses some certain system call sequences to execute its malicious behavior and that these system call sequences rarely appear in the normal codes, they can be used to extract the malware features. In order to achieve malware detection, TF-RF algorithm was used for features extraction

and the KNN clustering algorithm was used for malware detection model training. The proposed system consists of three components: preprocessing stage, features processing stage, and model-based detection stage, as shown in Fig. 1.

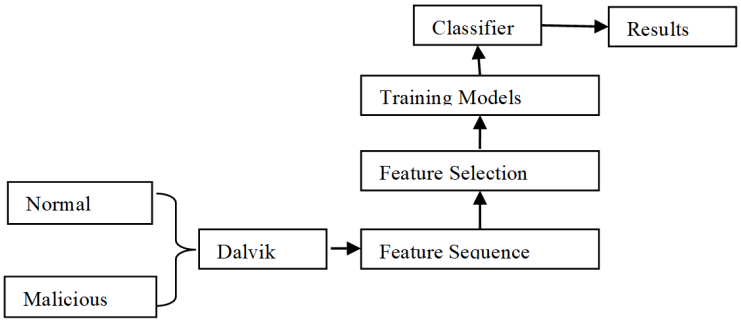

Fig. 1. Proposed system architecture

3. Detection theory

The main task of the preprocessing stage is to extract system call sequences. The most direct approach for that is to run the executable file under Dalvik environment and to use the compiled "Strace "tool to track the App system call paths in the entire lifecycle.

3.1. Anti-debug technique

Strace is a Linux tool for system calls tracking and execution process signals receiving [15]. The Strace tool is used to track system call sequences in order to extract complete application system call sequences. However, malicious codes use some techniques to monitor their codes in order to check whether the code is being debugged. In order to overcome this challenge of malicious code anti-analysis techniques, a solution, based on two frequently used malicious code anti-analysis techniques on Android platform, is presented.

(1) Anti-debug technique by self-attaching Advanced malware can make multiple processes mutually attached to each other and sometimes make one of them attached to itself to prevent other debugging process to attach it, which fulfills the purpose of anti-debugging. Regarding the App self-attaching, this study revises the code of the Ptrace system call under package, bionic/libc, in Android's source code, and generates a customized ROM. In the case that customized ROM detects that some process is trying to attach to itself, the function will return 0, which is a common resolution for the anti-debugging techniques.

(2) Anti-debug techniques by the process status dynamic monitoring Regarding the anti-debugging techniques by process status dynamic monitoring at runtime, the Native Hook technique is used in this paper. This technique hooks the I/O APIs file under bionic/libc, such as fopen function. When it finds a process is trying to access

its own process status file, it will redirect that process to other normal process to resolve this anti-debugging technique.

3.2. Feature processing

Feature processing stage mainly consists of two parts, features extraction and features selection. For features extraction, the N-Gram is used in order to obtain system call sequence. Then, the TF-RF features weighting algorithm is employed to conduct the features selection.

(1) N-Gram feature sequence extraction Due to its advantages such as simplicity, efficiency and easy implementation, the N-Gram's feature sequence extraction, called the N-Gram [11–14] statistic model, has been widely applied. The N-Gram statistic model can be used to mine the implicit relational features among the N-Gram data sets. These features are extremely efficient in unknown malware detection.

The parameters of the N-Gram model are

$$p(w_i|w_1w_2\dots w_{i-1}) = \frac{C(w_1w_2\dots w_{i-1}w_i)}{C(w_1w_2\dots w_{i-1})}\,, \tag{1}$$

where $C(w_1w_2\dots w_{i-1})$ denotes the occurring times of $w_1,\ w_2,\ \dots,\ w_i$ in the training data set.

(2) TF-RF relevance category feature weighting Traditional feature selection method might cause the majority class results' bias due to unbalanced data processing, thus, the predicted performances could be unsatisfactory for the minority class. Currently, most of studies that use feature selection method to improve the performance of unbalanced data classification are based on the feature weighting method.

In research presented in this paper, the TF-RF [16] algorithm was used and the correlation between features and classes was employed in order to enhance the importance of the minority class features. Furthermore, feature weighting and feature extraction from N-Gram system call sequence were obtained.

The symbol A denotes the number of t_k that appear in class c_i. Symbol B denotes the number of tk that do not appear in class c_i. Symbol C denotes the number of t_k that appear in class c_j. Finally, D denotes the number of t_k that do not appear in class c_j and $N = A + B + C + D$. Then

$$\text{tfrf} = \text{tf}\log\left(2 + \frac{A}{\max(1, C)}\right)\,. \tag{2}$$

Term frequency (TF) refers to the number of times that a specific term appears in the document. Since there is $D \gg A, B, C$, RF does not contain the value of D.

3.3. Malware detection model training

In model-based detection stage, the extracted feature vectors are taken as sample data sets, the classification algorithm is used to train the detection model, the KNN,

the decision tree, and the support vector machine were used here to conduct the supervised classification. and finally the obtained model is used to detect the test samples and to generate the detection report.

4. Simulation and analysis

The experimental data set contained 1000 normal applications from Android application market and 200 malicious applications from Debrin [7]. The smartphone Samsung SGH-I747M with 4.1.1 Android version was used in the experiment. From 1000 normal Android applications, 600 application were used for training and 400 for testing, similarly, from 100 malicious application, 120 application were used for training and 80 for testing. The experiment was obtained by comparison of three classification methods, the KNN, the SVM and the decision tree methods.

In this paper, two methods, the document frequency (DF) and the TF-RF class correlation features, were used for features extraction. The DF features extraction results are illustrated in Fig. 2. The TF-RF feature extraction results are illustrated in Fig. 3.

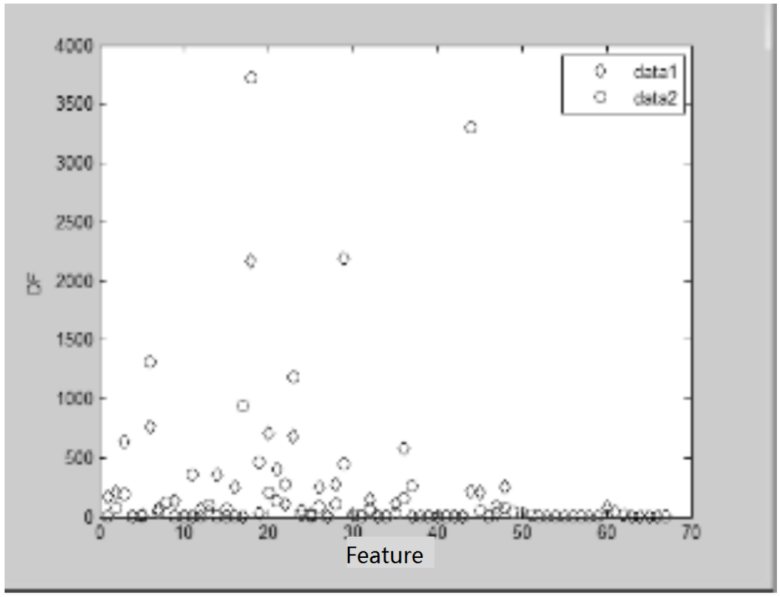

Fig. 2. DF

As can be seen in Fig. 2, the number of circles (20170119data2) is much higher than the diamonds (data1). Since the DF method only considers the document frequency of certain term class feature without consideration of the feature sequence's occurring frequency in the application, the unbalanced sample feature extraction maybe lead to extract more normal features, and fewer malicious features. It could easily cause great ambiguity in the extracted features, which would negatively affect

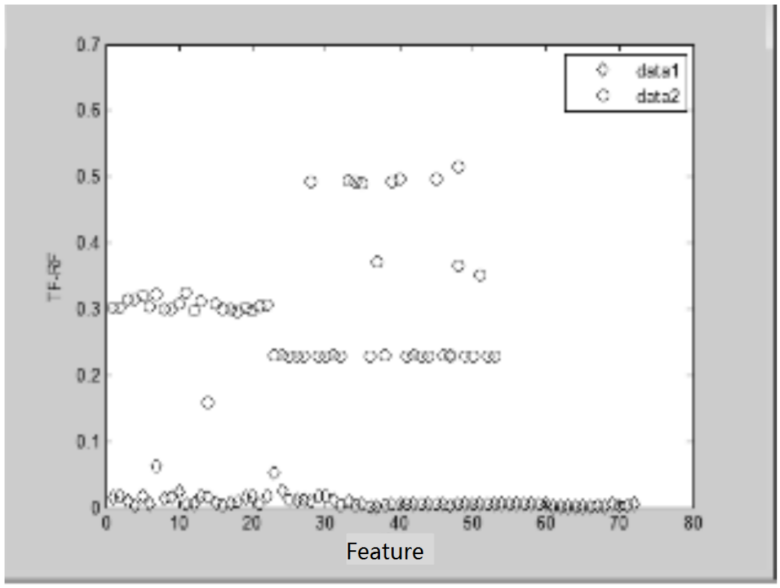

Fig. 3. TF-RF

the classification results.

It can be seen in Fig. 3 that data1 and data2 are basically balanced, since TF-RF has nothing to do with the feature of document frequency and it was only related to the size of relevance of the category, which means that not only features with high correlation to the majority class are kept, but also sufficient minority class features are selected. Thus, the selection between classes is more obvious, which has great benefit to the classification. TF-RF method can be avoid of imbalance problem from DF method.

5. Classification models

In the paper, k-Nearest Neighbor, Decision Tree and Support Vector Machine were used and the experimental results were shown in Figs. 4 and 5. In order to evaluate the detection results for the classification models effectively, the accuracy, true positive rate, false positive rate was adopted for method estimation.

Figures 4 and 5 illustrate the ROC curves of KNN, SVM and DT method. As can be seen in Figs. 4 and 5, the different values of Nlead to different results. When different thresholds are fixed, the detection rate and false alarm rate will vary in a different way. It is apparent that the most consistent detection scheme is the KNN approach, since it is only slightly worse than the SVM approach for detection rates (1 % and 2 %), and the low false alarm rates (below 5 %). Compare feature sequence lengths $N = 3$ and $N = 4$, when the length N was 4 and detection rate of KNN was the best. Thus we selected the feature sequence length as 4.

Through the comprehensive comparison and analysis, it is found that the detec-

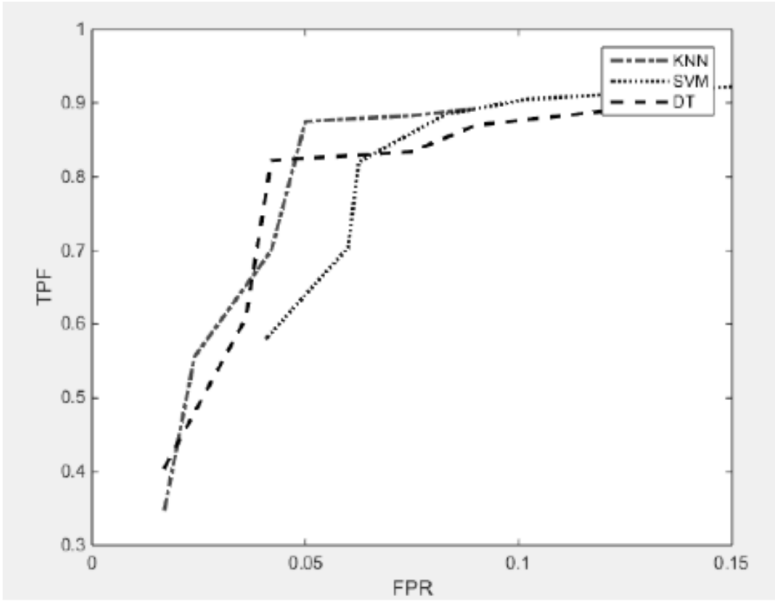

Fig. 4. $N = 3$ ROC

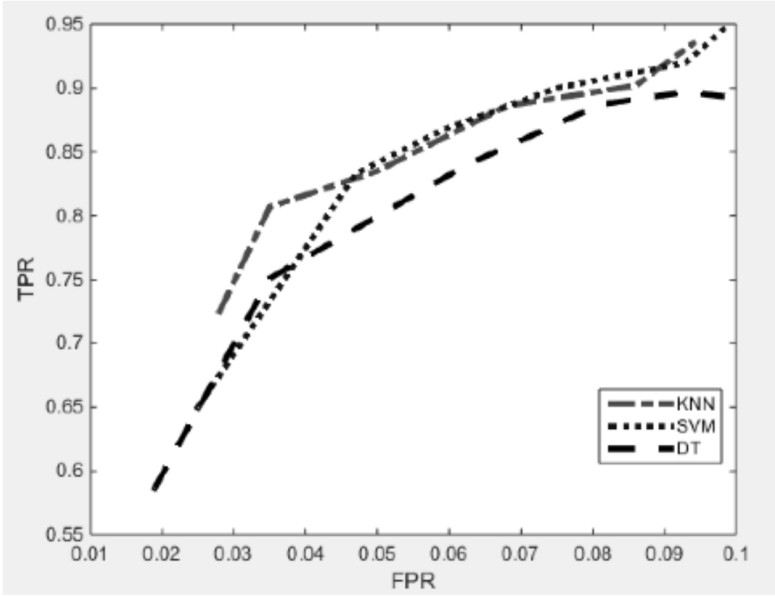

Fig. 5. $N = 4$ ROC

tion rate of TR-RF class association feature weighting method is higher than that of DF feature extraction method, and the false positive rate was significantly lower than that of DF feature extraction method. The overall performance of KNN was

better than DT and SVM.

6. Summary

The elaborated studies on the Android malware detection techniques showed that unbalanced sample set affects the classification accuracy. Therefore, the malware detection is a typical unbalanced samples classification problem. In order to reduce effects of unbalanced data on the classification results, the TF-RF class association feature weighting method was adopted. Firstly, the reverse analysis against the malicious code's analysis technique was used to obtain the system call traces of the complete lifecycle. Then, the N-Gram method was employed to mine the implicit relation features among system call functions. Moreover, the TF-RF feature weighting method was applied to extract the system call sequence features and to determine the weights of the features. Finally, comparison of three classification methods has been obtained, and it was concluded that the KNN clustering method gives the best results in terms of the detection results' evaluation curves and have low false alarm rate. Compared to the tradition DF and TF-RF methods, the class relation feature weighting method improved the classification efficiency of unbalanced samples data and significantly enhanced the detection accuracy.

References

[1] I. Burguera, U. Zurutuza, S. Nadjm-Tehrani: *Crowdroid: Behavior-based malware detection system for android.* ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), 17–17 October 2011, Chicago, Il, USA, Published ACM New York, USA (2011), 15–26.

[2] T. Bläsing, L. Batyuk, A. D. Schmidt, S. A. Camtepe, S. Albayrak: *An android application sandbox system for suspicious software detection.* International Conference on Malicious and Unwanted Software, 19–20 October 2010, Nancy, France, IEEE Conference Publications (2010), 55–62.

[3] T. Isohara, K. Takemori, A. Kubota: *Kernel-based behavior analysis for android malware detection.* International Conference on Computational Intelligence and Security (CIS), 3–4 December 2011, Hainan, China, IEEE Conference Publications (2011), 1011–1015.

[4] A. Bose, X. Hu, G. S. Kang, T. Park: *Behavioral detection of malware on mobile handsets.* International Conference on Mobile Systems, Applications, and Services (MobiSys), 17–20 June 2008, Breckenridge, CO, USA, Published ACM New York, USA (2008), 225–238.

[5] M. Christodorescu, S. Jha, C. Kruegel: *Mining specifications of malicious behavior.* European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering(ESEC-FSE), 3–7 September 2007, Dubrovnik, Croatia, Published ACM New York, USA (2007), 5–14.

[6] K. Chen, P. Wang, Y. Lee, X. F. Wang, N. Zhang, H. Huang, W. Zou , P. Liu: *Finding unknown malice in 10 seconds: Mass vetting for new threats at the google-play scale.* USENIX Conference on Security Symposium (SEC), 12–14 August 2015, Washington, D.C., Published USENIX Association Berkeley, CA (2015) 659–674.

[7] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck: *Drebin: Effective and explainable detection of android malware in your pocket.* Network and Distributed System Security Symposium (NDSS), 23–26 February 2014,

San Diego, CA, USA, www.internetsociety.org/doc/drebin-effective-and-explainable-detection-android-malware-your-pocket.

[8] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. G. Chun, L. P. Cox, J. Jung, P. McDaniel, A. N. Sheth: *TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones.* ACM Transactions on Computer Systems *32* (2014), No. 2, Article 5, 1–29.

[9] G. M. Wiess: *Mining with rarity: A unifying framework.* ACM Sigkdd Explorations Newsletter *6* (2004), No. 1, 7–19.

[10] B. Wolfe, K. Elish, D. Yao: *High precision screening for android malware with dimensionality reduction.* International Conference on Machine Learning and Applications (ICMLA), 3–6 December 2014, Detroit, MI, USA, IEEE Conference Publications (2014), 21–28.

[11] P. F. Brown, P. V. de Souza, R. L. Mercer, V. J. Della Pietra, J. C. Lai: *Class-based n-gram models of natural language.* Journal Computational Linguistics *18* (1992), No. 4, 467–479.

[12] J. O. Kephart: *A biologically inspired immune system for computers.* In Artificial Life IV: International Workshop on the Synthesis and Simulation of Living Systems, Brooks and Pattie Maes, MIT Press, Cambridge, Massachusetts (1994), 130–139.

[13] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden aP. G. Bringas: *Idea: Opcode-sequence-based malware detection.* Engineering Secure Software and Systems, Springer Berlin Heidelberg, Lecture Notes in Computer Science *5965* (2010), 35–43.

[14] P. Venu, D. V. R. Prasad: *N-gram analysis in SMV training phase reduction using dataset feature filtering for malware detection.* International Journal of Science and Research (IJSR), Impact Factor (2012): 3.358 *3* (2014), No. 9, 550–554.

[15] M. Frye: *Effects of suction/blowing on steady boundary layer stagnation-point flow and heat transfer towards a shrinking sheet with thermal radiation.* Retrieved August 26 2006 from www.redhat.com/magazine/ 010aug05/features/strace/ (2006).

[16] M. Lan, C. L. Tan, J. Su, Y. Lu: *Supervised and traditional term weighting methods for automatic text categorization.* IEEE Transactions on Pattern Analysis and Machine Intelligence *31* (2009), No. 4, 721–735.

Received April 30, 2017