ICIC Express Letters
Volume **10**, Number **5**, May **2016**

 ISSN 1881-803X
pp. **1173–1180**

THE MALWARE DETECTION OF ANDROID SYSTEM BASED ON SVM

Lixia Xie and Binbin Zhao

School of Computer Science and Technology
Civil Aviation University of China
No. 2898, Jinbei Road, Dongli District, Tianjin 300300, P. R. China
lxxie@126.com

Received October 2015; accepted January 2016

Abstract. *A new malware detection model is proposed in this paper to solve the Android platform malware detection problem. The model consists of clients and a server; the clients get sample's information of permission, broadcast, service, and traffic. The server removes redundant feature information according to their correlation and quantifies the remaining feature information as the n-dimensional feature vectors, and then the server uses the geometric interval to figure out the optimal hyperplane. In the last, the server uses the trained detector to classify the software. The detection experiments on Android environment were performed and the experimental results showed that our detection model had a higher accuracy.*

Keywords: Malware, Feature vectors, Support vector machine (SVM), Geometric interval, Hyperplane

1. **Introduction.** Android is a mobile operating system based on Linux which has been gradually beyond Symbian, IOS to become the mainstream operating system of the intelligent mobile devices since Google released it in November 2007. Its explosive growth and open features also bring a lot of security threats.

Safety reports in 2014 published by Cheetah Mobile which is an international authoritative mobile information security organization showed that: the increase rate of malware in Android in 2014 was 2.5 times that of 2013 and the number of infected files in 2014 was more than 20 times that of 2012 [1]. The security threats of Android system in 2014 include: mobile payment, malicious deduction, privacy disclosure, remote control, hacking tools [1], etc.

At present, the main way of malware detection includes [2] static detection, dynamic detection and hybrid detection. The static detection disassembles applications and matches the application's signature which has high execution rate and accuracy rate, but it cannot identify the new and variant malware; The dynamic detection detects malware by identifying the hidden behavior which needs more samples and has a lower accuracy; The hybrid detection is a combination of static detection and dynamic detection which is hard to realize, but has a higher accuracy.

The mainstream malware detection model at home and abroad includes Crowdroid [3], DroidRanger [4], etc. Crowdroid [3] uses the method of machine learning to monitor the system's behavior which can detect new and variant malware, but it needs more samples and does not use the information of Android virtual machine; DroidRanger [4] uses the hybrid way to analyze and identify the malicious behavior, but the extraction of permission information is achieved in PC which makes it complex to realize and apply. So, it needs to be improved in detection rate and applicability.

In this paper a new mobile malware detection model combined with the current mainstream malware detection way was proposed which has high detection rate and strong

applicability. The model collects the feature information of samples in clients, deals with feature information and extracts feature vectors, takes the geometric intervals as criterion and uses the algorithm of SVM to train the detector in server. Lastly, the model uses the detector to detect the software.

The remainder of the paper is structured as follows. The detection model is described in Section 2. Section 3 briefly describes the implementation of the detection model. Our experimental results and analysis are reported in Section 4, and we conclude the paper in Section 5.

2. **Design of the Detection Model.** In order to overcome the problem of low detection rate and poor applicability in current detection models, a new malware detection model is put forward in this paper. The core modules of the model are feature information extraction module, detection and recognition module.

Feature information extraction module is mainly responsible for collecting feature information and providing data for training detector and detecting malware; detection and recognition module is mainly responsible for dealing with feature information, training detector and using the trained detector to identify malware and benign software. The overall architecture of the detection model proposed in this paper is shown in Figure 1.

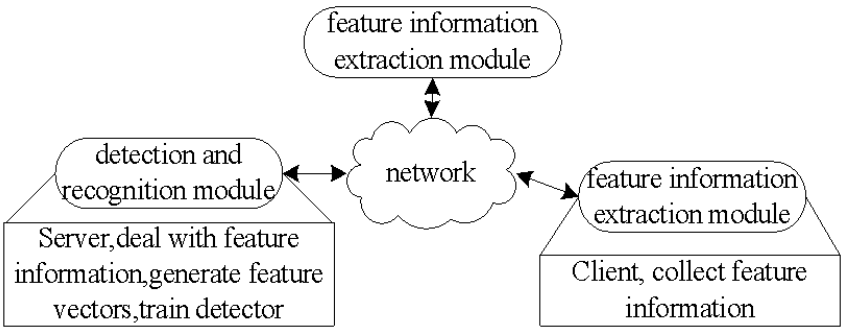

FIGURE 1. The overall architecture of the detection model

2.1. **Feature information extraction module.** The safety of Android system largely depends on the permission mechanism. The corresponding permission is needed when the application performs any actions in Android system. Only when the application applies for the corresponding permission, can the application perform the action. Only when the Android system accepts all permissions of the software, can the software be installed and run. Therefore, permission information is the detection model's preferred feature information; the spread of malware or privacy information needs to consume large amounts of network traffic, so the application's network traffic is also important feature information which is needed in detecting. When installed, the malware usually starts by monitoring the system's boot broadcast and performs unauthorized malicious behavior in the background periodically or by monitoring the system's broadcast. The Android's security mechanism requires the application to apply for the corresponding broadcast and service information to perform the operation, so the broadcast and service information are other important feature information which is needed in detecting.

At present, the extraction of mainstream detection model's feature information usually uses the following way: firstly, using tools to disassemble or directly decompressing the APK (AndroidPackage) to get the file of AndroidManifest.xml [5]; then, parsing it to get the relevant feature information. This kind of method has a lot of work to do and its efficiency is low. The detection model proposed in this paper uses the following way to extract feature information: using the clients to collect the feature information which is running on the mobile devices, and then the clients upload the feature information to the server by network.

2.2. **Detection and recognition module.** The process of the detection and recognition module mainly includes dealing with feature information [6], generating feature vectors, training detector and identifying malware. Firstly, the module figures out the correlation degree and removes redundant feature information; secondly, the module quantifies the feature information to n-dimensional feature vectors and uses them to train the detector; lastly, the detector is used to identify malware.

2.2.1. *Dealing with the feature information.* Due to the redundancy and irrelevance of the feature information, redundant and invalid feature information needs to be removed to reduce the complexity of the detector and the error rate, such as: android.permisson. WRITE_EXTERNAL_STORAGE, android.permisson.MOUNT_UNMOUNT_FILESYSTEMS [8]. These two kinds of feature information appear in pairs, and it will be ok to just keep one.

In this paper, the model calculates the feature information's information entropy [7] and correlation to remove redundant and invalid feature information. The information entropy [7] of feature information $\boldsymbol{A}$ ($H(\boldsymbol{A})$) (N is the type of feature information $\boldsymbol{A}$ can be, P is the emergence probability of the feature information) and the conditional information entropy [7] of feature information $\boldsymbol{A}$ after known feature information $\boldsymbol{B}$ ($H(\boldsymbol{A}|\boldsymbol{B})$) ($M$ is the type of feature information $\boldsymbol{B}$ can be) is:

$$H(A) = -\sum_{i=1}^{N} P(A_i) \log_2 P(A_i) \tag{1}$$

$$H(A|B) = -\sum_{i=1}^{N} \sum_{j=1}^{M} P(A_i B_j) \log_2 P(A_i|B_j) \tag{2}$$

The mutual information between feature information $\boldsymbol{A}$ and feature information $\boldsymbol{B}$ ($I(\boldsymbol{A},\boldsymbol{B})$) is:

$$I(A, B) = H(A) - H(A|B) \tag{3}$$

The relevance ($\mathrm{Sim}(\boldsymbol{A},\boldsymbol{B})$) between feature information $\boldsymbol{A}$ and feature information $\boldsymbol{B}$ can measure the correlation between them. The formula is:

$$Sim(A, B) = 2\frac{I(A, B)}{H(A) + H(B)} \tag{4}$$

If the correlation between a set of feature information is 1, it will be ok to just keep one; otherwise, they all need to be kept.

2.2.2. *Generating feature vectors.* The processes of obtaining the handled feature information and quantifying them to n-dimensional feature vectors are: counting and recording the type and total amount of feature information; for the feature information of permission, if the sample has this permission, then the feature value is 1, otherwise, the feature value is 0; for the feature information of broadcast and service, if the sample has this service or broadcast, then the feature value is 1, otherwise, the feature value is 0; for the feature information of network traffic, if the sample consumes network traffic (not applying for the permission of Internet can also have access to the network), then the feature value is 1, otherwise, the feature value is 0; counting the feature value of every feature information and composing them to n-dimensional feature vectors orderly.

The generated feature vector sample is as shown below:

Benign software: 0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0.

Malware: 0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,
0,0,1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,1,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,
0,0,1,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,1,0,0,1,0,0,1,0.

2.2.3. *Training the detector.* The training of the detector is completed through the SVM algorithm, and the decision function of SVM is:

$$g(x) = wx + b \tag{5}$$

Among them, w is n-dimensional normal vector, b is the offset, x is n-dimensional feature vectors, namely:

$$x = (x_1, x_2, \ldots, x_n) \tag{6}$$

If the sample is benign software, then the value of $g(x)$ is 1; if the sample is malware, then the value of $g(x)$ is -1.

The ultimate goal of training detector is to figure out a set of w and b to make sure the hyperplane $(wx + b = 0)$ can separate the sample points which represent the benign software and malware in the n-dimensional space (as shown in Figure 2).

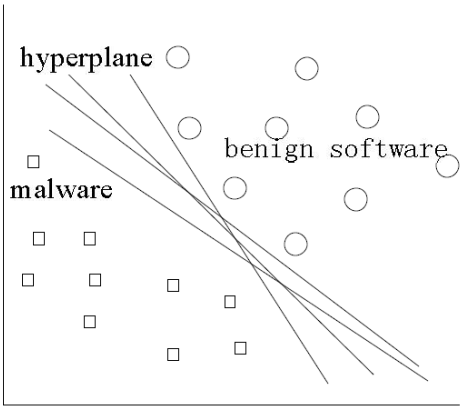

FIGURE 2. Hyperplane classifying software

In order to ensure the sample points of benign software and malware can be classified, fix geometry interval to 1 and get the objective function of w (Formula (7)), and constraint conditions of w and b (Formula (8)); among them, $(i = 1, 2, \ldots, l)$ (l is the number of samples), w is not only related to the location of samples but also related to the type of samples (Formula (9)).

The sample points of benign software and malware are linear inseparable. We use the radial basis kernel function [8-10] (Formula (10)) to map sample space to high dimension space; we put the slack variable ε_i $(i = 1, 2, \ldots, l)$ and penalty factor $\boldsymbol{C}$ into objective function and constraint condition; finally, we get the objective function and constraint condition of w and b (Formulas (11), (12)).

$$Min\left(\frac{1}{2}||w||^2\right) \tag{7}$$

$$g(x_i)\left[(wx_i + b)\right] - 1 \geq 0 \tag{8}$$

$$w = \partial_1 g(x_1)x_1 + \partial_2 g(x_2)x_2 + \cdots + \partial_l g(x_l)x_l \tag{9}$$

$$K(x_i, x) = e^{-\gamma||x_i - x||^2},\ \gamma > 0 \tag{10}$$

$$Min\left\{\frac{1}{2}||w||^2 + C\sum_{i=1}^{l}\varepsilon_i\right\} \tag{11}$$

$$g(x_i)\left[(wx_i + b)\right] \geq 1 - \varepsilon_i \tag{12}$$

2.2.4. *Identifying malware.* The detection and identification of malware is to put the feature vectors into decision function (Formula (7)) and get the results.

The detection and identification process of malware is: dealing with feature information and quantifying them to n-dimensional feature vectors; putting the feature vectors into

decision function and getting the value of $g(x)$. If the value of $g(x)$ is greater than or equal to 1, then the sample is benign software; if the value of $g(x)$ is less than or equal to -1, then the sample is malware; otherwise, adjusting the feature vector and calculating again.

3. **Realization of the Detection Model.** The feature information collecting module contains the mainstream virus database (source: Kingsoft Antivirus). It firstly gets the signature of the test software and makes a static detection. If the signature is a malicious signature, then the user will be notified; otherwise, the module will get the feature information and upload them to the server.

In this paper the collecting of feature information is in the following way.

(1) The client uses the class of PackageManager to collect the feature information.

(2) The client uses the class of TrafficStats to collect the network traffic each application consumed from the system startup to now.

(3) The client uploads feature information to the server in the format of JSON and loads the detection results periodically.

The detection and identification module consisted of detector runs in the server which is the core module of the detection model. Its main function is using the trained detector to calculate the results and feedbacking the results to clients.

3.1. **Training the detector.** The ultimate goal of training detector is to get the value of the n-dimensional normal vector w and the offset b. The training process is:

(1) The server responds the HTTP request which comes from the client to get the feature information in the format of JSON and uses the third party kits (json-lib.jar) to parse the feature information. Then the server saves the feature information into the tables of MySQL (trainWare: saving the name of the samples and whether the sample is malware; template: saving the feature information of samples).

(2) We use methods and formulas Section 2 described to figure out the correlation degree of feature information and remove the redundant feature information in table template according to the results.

(3) We use the quantitative rules Section 2 described to generate the feature vectors, and then we put the feature vectors into the objective function and constraint condition, (Formulas (11), (12)) to get the value of w and b. At this point, the training of the detector is over.

3.2. **Identifying the malware.** The identification of malware is putting feature vectors of the test sample into detector to calculate the results. The process is:

(1) The server responds the HTTP request which comes from the client to get the feature information in the format of JSON and uses the third party kits (json-lib.jar) to parse the feature information. Then the server saves the feature information into table testWare (testWare: saving the name of the test samples) and table feature (feature: saving the feature information of the test samples).

(2) We match table feature with table template and remove the type of the feature information from table feature if table template does not contain the type of the feature information. Lastly, we quantify the feature information of table feature and compose them to n-dimensional feature vectors orderly.

(3) We put the feature vectors of the test sample into the decision function; if the output of the decision function is greater than or equal to 1, then the test sample is benign software; if the output of the decision function is less than or equal to -1, then the test sample is malware.

(4) Lastly, the server feedbacks the results to the clients in the format of JSON.

4. **Test and Analysis.** We used J2SE technology to design and implement the core program of client and server and used the Android mobile devices and PC as the test platform for the clients and server.

1800 samples were used to train and test the detection model which came from the related websites, BBS, etc. We selected 800 samples as the training samples; among them 400 training samples were benign samples, and 400 training samples were malware. We selected 1000 samples as the test samples; among them, 600 test samples were benign samples, and 400 test samples were malware.

The feature information collecting interface of training samples is shown in Figure 3, the part feature information statistics of training samples is shown in Figure 4, and the result of detecting interface is shown in Figure 5.

The detection results of 1000 test samples are shown in Table 1.

The detection results of Table 1 show that: The detection model proposed in this paper has higher detection rate and lower error rate; document [5] proposes a detection method of using only the permission information as feature information and different algorithms as the classification algorithms. We compared the algorithm of SVM with J48, Random forest and CART proposed in document [5]. The comparison results are shown in Table 2.

The comparison results of Table 1 and Table 2 show that: the detection model proposed in this paper can identify the malware and benign software effectively.

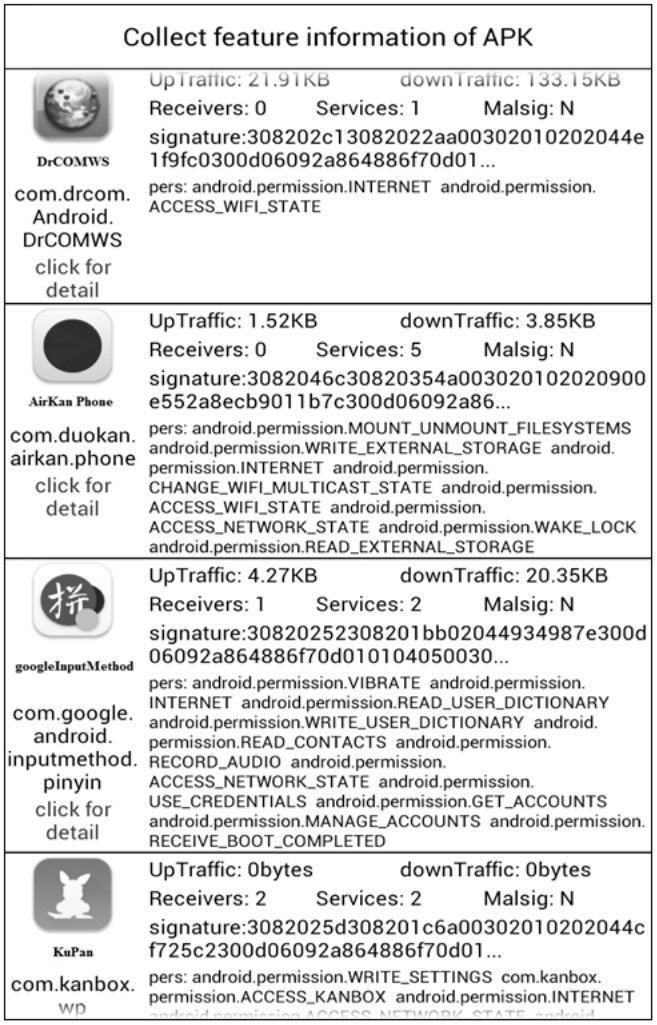

FIGURE 3. Feature information

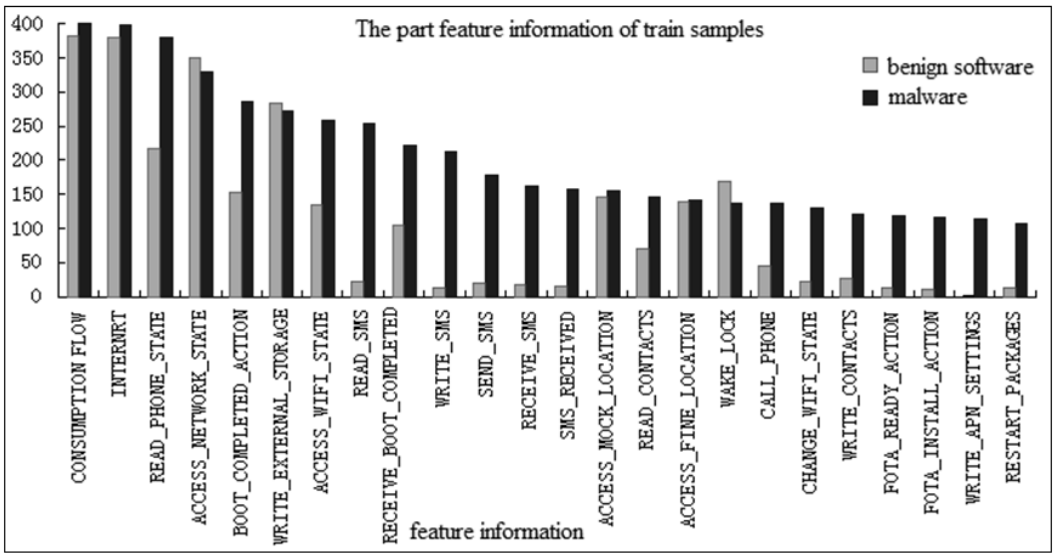

FIGURE 4. Part feature information of training samples

FIGURE 5. Detection results

5. **Conclusions.** This paper proposes a malware detection model for the malware detection problem. In the model, the clients are used to detect the software preliminarily, extract the feature information and upload the feature information to the server; the server handles the feature information and generates the feature vectors, takes the geometric interval as the rules and uses the SVM to train detector; lastly, the detector is used to detect software.

Although this model can effectively detect malware, it is still insufficient. For example, the collecting period of some feature information is longer, the number of training sample is large, and it is difficult to train the detector.

TABLE 1. Detection results

Sample	Number	CIN	EIN	CR	ER
benign	600	551	49	91.8%	8.2%
malicious	400	369	31	92.25%	7.75%
total	1000	920	80	92.0%	8.0%

CIN: Correct Identified Number;
EIN: Error Identified Number;
CR: Correct Rate; ER: Error Rate.

TABLE 2. Comparison of detection results

Result	J48	Random forest	CART
CR	90.72%	91.75%	90.72%
ER	9.28%	8.25%	9.27%

CR: Correct Rate; ER: Error Rate.

We will make some improvement in increasing the number of samples, increasing the types of feature information (such as: the information of CPU, memory and process), optimizing the training algorithm and so on.

Acknowledgement. This work is partially supported by the National Natural Science Foundation of China (NSFC). We would like to thank NSFC and people for their supports.

REFERENCES

[1] *2014 Half Year Security Report*, http://www.cmcm.com/blog/2014-07-18/186.html.
[2] S. Michael, S. Thomas, E. Florian, A. Daniel et al., Mobile-Sandbox: Combining static and dynamic analysis with machine-learning techniques, *International Journal of Information Security*, vol.14, no.2, pp.141-153, 2015.
[3] B. Iker, Z. Urko and N. Simin, Crowdroid: Behavior-based malware detection system for Android, *Proc. of the First ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, New York, America, pp.15-26, 2011.
[4] Y. Zhou, Z. Wang, W. Zhou and X. Jiang, Hey, you, get off of my market: Detecting malicious Apps in official and alternative Android markets, *Proc. of the 19th Annual Symposium on Network and Distributed System Security*, Geneva, Switzerland, 2012.
[5] A. Zarni and Z. Win, Permission-based Android malware detection, *International Journal of Scientific & Technology Research*, vol.2, no.13, pp.2277-8616, 2013.
[6] *The List of Permissions for Android Applications*, http://www.cnblogs.com/hibraincol/archive/2010/09/14/1826337.html.
[7] H. Yang, X. Chen, H. Sun and S. Wu, Spatiotemporal structural evolution and regional differentiation analysis of land use in Xinjiang based on information entropy, *Proc. of the 2nd Conference on Environment Science and Information Application Technology*, Wuhan, China, pp.633-638, 2010.
[8] K. Pankaj, K. Asha and N. Baluram, Offline handwriting recognition using invariant moments and curve let transform with combined SVM-HMM classifier, *Proc. of the International Conference on Communication System and Network Technologies*, Gwalior, India, pp.144-148, 2013.
[9] K. Ashish and K. Sarika, Analysis of timing constraint on combined SVM-HMM classifier and SVM classifier, *Proc. of IEEE International Conference of Innovation and Technology in Education*, Jaipur, India, pp.214-218, 2013.
[10] R. Bernardete, M. Senior, L. Noel et al., Signature identification via efficient feature selection and GPU-based SVM classifier, *Proc. of International Joint Conference on Neural Networks*, Beijing, China, pp.1138-1145, 2014.