Journal of Computers Vol. 27, No. 3, 2016, pp. 21-31
doi:10.3966/199115592016102703003

A Method of Android Malware Detection Based on BP Neural Network Combining Static and Dynamic Features

Li Zhu[1] Genying Wang[1,*] Junsong Fu[1] ZhongDing Dong[1,2]

[1] Department of Electronic and Information Engineering, Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing Jiaotong University, Beijing, 100044, China
13120193@bjtu.edu.cn, gywang@bjtu.edu.cn, 14111005@bjtu.edu.cn

[2] Department of Network Security, China's Information Security Evaluation Center, Beijing, 100085, China
denniel_narco@sina.cn

Received 1 July 2015; Revised 27 July 2015; Accepted 10 August 2015

Abstract. At present, Android smart mobile terminals have been becoming more and more popular, meanwhile, due to its property of high openness, the Android platform has become the main target of the attackers. In order to effectively detect the malicious software, this paper presents an approach based on BP neural network. This approach not only considers the static features of the APK of the application, but also the running characteristics of the applications. In addition, after completing collecting features, to reduce the processing capacity and the complexity of the classification algorithm, dimension reduction technique is introduced into our approach, at the same time, comparative analysis is conducted between two dimension reduction methods. Then these features are used to train and test the classification algorithm with ten -fold cross validation. The empirical results and comparative analysis demonstrate that our approach can detect unknown Android malware accurately.

Keywords: Android malware, BP neural network, dimension reduction, PCA, information gain

1 Introduction

With the popularity of the Android smartphone, Android platform faces more and more attacks in recent years. According to the report of Fortinet (November 2011), approximately 2000 Android malware samples had been discovered, and these malware belonged to 80 different families [1]. Furthermore, according to a Finnish security company F-secure, among 301 mobile malware samples that arose in 2012, 238 of them targeted Android platform [2]. Since the first Android malware was discovered in 2010, more and more sophisticated malware families have emerged, which are able to evade traditional signature-based detection [3].

The vast majority of users often download apps from third party markets, and these platforms have no strict safeguards to prevent the producers releasing malware. Therefore, the mobile terminals are very easily to be attacked. As a result, this paper proposes a novel method to effectively detect Android malware. Due to the wide application of neural network and its favorable classification characteristics, this paper employs it to the detection of Android malware. These features applied to detect malware contain two parts. The first part is static features extracted from the installation package; the second part is dynamic behavior-based characteristics, which are collected from Android virtual device when the malicious application is running in it. The combination of static and dynamic features results in a favorable detection result.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 presents the system structure of our approach, and the core classifying approach is presented in Section 4. The

* Corresponding Author

approach is conducted simulation in Section 5. Finally, we conclude this paper and present some future research plans in Section 6.

2 Related work

Traditionally, signature-based detection [4-7] is a high-frequently-used method to detect mobile malware. Followed by the emergence of more sophisticated malware, which is able to evade traditional signature-based detection, the method becomes invalid.

In recent years, to detect Android malware, there are mainly two kinds of methods: static analysis and dynamic methods. Static analysis refers to recognize the application benign or malicious by analyzing the APK without running the application. On the contrary, dynamic methods need to install the application and run it. During its running, the required features information is collected. Moreover, dynamic methods can be further subdivided into behavior-based detection and taint-analysis-based detection.

The advantage of static analysis [8-15] lies in comprehensive analysis, as it is difficult to disguise malicious behavior during static analysis. ComDROID [8] is used to detect application vulnerabilities in communication, and DroidChecker [9] is applied to finding privacy leakage problem of Android applications. In addition, some previous researches ever employed static analysis to detect malicious properties, like SCANDAL [10], AndroidLeaks [11], and the framework presented in [12], but they only focus on privacy leakage problem, whereas the issues we target is far more than privacy issues.

Behavior-based detection [16-22] comparatively analyzes the current behavior and established attack pattern. When the smart phone is being used, the method detects abnormal behavior by monitoring memory usage amount, SMS, battery consumption amount, etc. event information. By monitoring the condition of some terminal properties in real time, Shabtai, Kanonov, Elovici, Glezer, & Weiss [20] put forwards anomaly detection method, which applies machine learning. By inspecting the power consumptions of terminals in real time, Buennemeyer, Nelson, Lagett, Dunning, Marchany, & Tront [21] builds an outstanding virus detection model. In this paper, to detect malware more effectively, we select some terminal properties as dynamic features.

TaintDroid [23-26] is an information-flow tracking system by monitoring real-time privacy information on smart phones. The application and development of this method are hindered by the limited overhead of smart phones in real environment.

The main contributions of this paper lie in the following three aspects.

Firstly, these features applied to detect malware combine the static features extracted from the APK with dynamic behavior features collected from the Android smartphone itself when the application is running.

Secondly, to reduce the processing capacity and the complexity of the classification algorithm, dimension reduction of the combined features adopts two kinds of methods: the PCA (principal components analysis) and the algorithm of information gain, they are used to select top-ranked features and the results are comparatively analyzed.

Thirdly, this paper utilizes BP neural network to recognize the application benign or malicious with ten-fold cross validation. Meanwhile, in order to raise the detection rate, and according to the principle that the minority is subordinate to the majority, multiple neural networks vote on the classification results. What is more, the average detection rate is compared with a few existing Android security tools and some related works.

3 System structure

As presented in Fig.1, the APK is respectively sent to two modules for processing. In the module of ‘APK Analysis’, the APK is decompressed into separate folders, including the Manifest file, the .dex file and other resource subfolders. We especially utilize the Manifest file and the .dex file. The Manifest file is used to extract the permissions the application involves, and the .dex file contains code properties of the application. Afterwards, we convert the Manifest file to readable format with AXML2jar. The .dex file is disassembled using Baksmali [25]. Baksmali is a disassembler for the .dex format in Dalvik. Baksmali disassembles .dex files into multiple files with .smali extensions. Each .smali file contains only one class information, which is equivalent to a Java .class file. Subsequently, these files are used to ex-

tract relevant features to construct classification models based on BP neural network. The sub module of the module 'The Android Emulator' is presented in detail as following Fig.2.

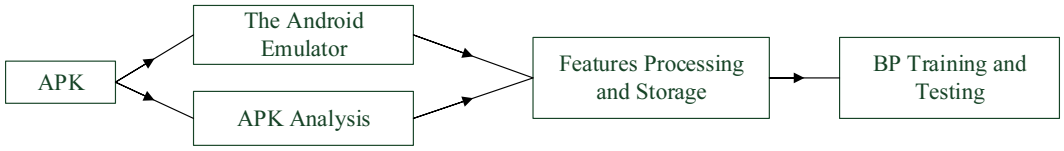

Fig. 1. System structure of classification algorithm

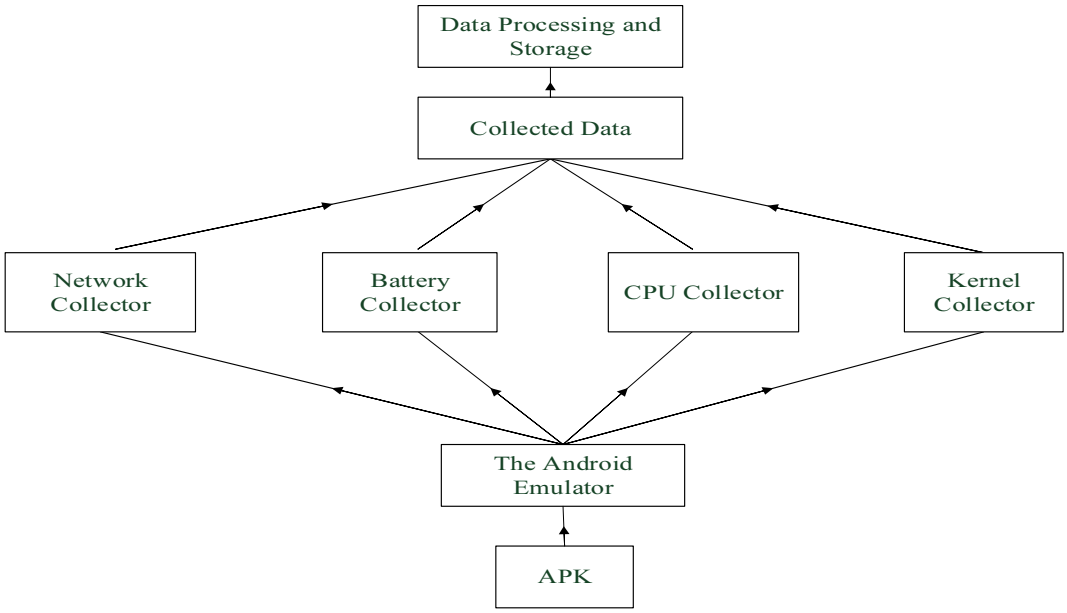

Fig. 2. The framework of features collection by using the Android simulator

Firstly, the application is installed in the Android emulator. While the application is running, we collect dynamic behavior features by some collectors such as network collector, battery collector and so on. These collected data is gathered together, and in order to be addressed as the data format the classifier can handle, they are then sent to the module "Data Processing and Storage". The following chart shows the dynamic features extracted.

Table 1. The dynamic behavior-based features

Resource	Features
CPU	CPU Usage Rate
Telephone	The Number, Contents of Send/Receive Calls
SMS	The Number, Contents of Send/Receive SMS
Network	Rx Packets, TX Packets, Rx Bytes, TX Bytes
Process	Process ID, Process Name, Running Process, Context Switches
Battery	Voltage, Temperature

These features we need are then sent to the module "Features Processing and Storage" stored as a unified matrix format. Afterwards, the data set was divided into training set and testing set with ten-fold cross validation. Eventually, for each BP neural network, we conduct the training and testing ten times, and the final result is voted by ten BP neural networks, and we take the average detection rate as the final result.

4 Core algorithm of classifier

The data set for classifying contains two parts: the features data and the class label of the application. To reduce the processing capacity and the complexity of the classification algorithm, this paper adopts two kinds of dimension reduction methods: the PCA (principal components analysis) and the algorithm of information gain to select top-ranked features. Afterwards, we utilize BP neural network for training and testing. Unlike a single BP neural network, according to the minority is subordinate to the majority, ten neural networks vote on the classification results.

In this paper, 1000 malicious apps from 28 malware families are utilized, the categories and their respective number of samples are shown in Table 2. The malware samples are mainly obtained from the Android Malware Genome Project [3]. The sample set of 1000 benign apps covers a wide variety of application. The categories include system tools, health and fitness, entertainment, news and magazines, sports, finance, music and audio, education, business, games and so on. The benign apps from third party market were identified through a series of security software.

Table 2. Malware families and their respective number

Family	No. of samples	Family	No. of samples
AnserverBot	75	Geinimi	104
BaseBridge	100	GoldDream	37
Bgserve	10	Kmin	43
Carberp	12	Pjapps	25
Cawitt	31	RootSmart/Bmaster	15
CruseWin	23	Saiva	33
DroidDream	30	Scavir	10
DroidDreamlight	25	TigerBot	20
DroidKungFu	112	UpdtBot	45
Extension/Monad	17	Uxipp	17
FakeFlash	21	Vdloader	13
FakeInst	13	YZHC	48
FakePlayer	36	Zeahache	16
Gamex	49	Zsone	20

4.1 The algorithm of dimension reduction

In this paper, two dimension reduction techniques are employed, the principles of these two algorithms is detailed below.

(1) PCA

Let a column vector represents the features set of an application as following.

$$X = (\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_p)^T$$

The letter p represents the number of all extracted features, and the value of x_i is 1or 0, which represents having the feature or not.

$$X_i = (\mathrm{x}_{i1}, \mathrm{x}_{i2}, ..., \mathrm{x}_{ip})^T, i = 1, 2, ..., n.$$

The letter n is the number of application samples and n>p. After constructing the samples matrix, the following standardized transformation is implemented to each element of the sample matrix.

$$Z_{ij} = \frac{X_{ij} - \overline{X}_j}{S_j}, i = 1, 2, ..., n; j = 1, 2, ..., p. \quad \textbf{(1)}$$

In the above formula,

$$\overline{X}_j = \frac{\sum_{i=1}^{n} X_{ij}}{n} \quad \textbf{(2)}$$

$$S_j^2 = \frac{\sum_{i=1}^{n} (X_{ij} - \overline{X}_j)^2}{n-1} \tag{3}$$

Z is just the standard matrix. Then solve the correlation coefficient matrix of the standard matrix Z as following.

$$R = \frac{Z^T Z}{n-1} \tag{4}$$

Obtain the p characteristic roots according to $|R - \Upsilon I_p| = 0$, determine the principal components by the following methods.

$$\frac{\sum_{j=1}^{m} \Upsilon_j}{\sum_{j=1}^{p} \Upsilon_j} \geq percent \tag{5}$$

Determine the value of m according to the above formula making the information utilization rate reach more than percent.

For each Υ_j, $j = 1, 2, \ldots, m$, solve the unit eigenvector b_j^o from equation $Rb = \Upsilon_j b$. Then transform the standardized index variables into the main components by $U_{ij} = Z_i^T b_j^o$, $j = 1, 2, \ldots, m$. As a result, U_k represents the K_{th} main component.

(2) Information Gain

The sample matrix of applications is expressed the same as PCA. Differently, we calculate the mutual information (MI) [24] or information gain of each feature x_i related with the class variable C. Let C be a random variable representing the application class.

$$C \in \{\text{malicious}, \text{benign}\}$$

As the goal is to select the most relevant features, we calculate the information gain as follows.

$$\begin{aligned} MI(x_i, C) &= \sum_{x \in \{0.1\}} \sum_{c \in \{mal, ben\}} P(x_i = x; C = c) \cdot \log_2 \left(\frac{P(x_i = x; C = c)}{P(x_i = x) P(C = c)} \right) \\ &= \sum_{x \in \{0.1\}} \sum_{c \in \{mal, ben\}} P(x_i = x) \cdot P(C = c \mid x_i = x) \cdot \log_2 \left(\frac{P(C = c \mid x_i = x)}{P(C = c)} \right) \end{aligned} \tag{6}$$

The higher the information gain is, the more relevant the feature. After calculating the information gain for each feature x_i the features is then ranked in descending order. As a result, in order to maximize the classification accuracy, the top n most relevant features are selected for training the classifier.

4.2 BP neural network

This paper utilizes favorable classifying property of BP neural network. It is kind of multilayer feed-forward neural network training by error back propagation algorithm. Without prior to reveal the mathematical equations to describe this mapping, BP neural network is able to learn and store large amounts of input-output mapping relationships. Its learning rule is the method of steepest descent, to achieve the minimal mean square error (MSE) of the network, the network weight value and threshold value is constantly adjusted by back propagation. The topology structure of a BP neural network includes input layer, hidden layer and output layer.

Note that ten-fold cross-validation is employed in this paper. Thus, random 1600 samples are used in the training, whereas the remaining 400 samples are applied for testing. Hence, in the experiment process of ten-fold cross validation, a BP neural network uses ten different training and testing sets. This strategy helps classifier to learn more comprehensive, and provides a wider range of samples to test the classifier's ability to detect unknown malware. In conclusion, this method contributes to improve the performance of the classifier.

5 Experiment results and discussing

In this part, we conduct experiments on benign and malicious applications, including features processing, the training and testing of the classifier, comparative analysis.

5.1 Features processing and BP training

After extracting the dynamic behavior-based features and static mixed permissions and code properties, these features are then implemented the process of dimension reduction. These characteristics after the dimension reduction for subsequent classification processing are shown in Table 3 and Table 4.

Table 3. Scope of principal components and their respective percentage

Scope of Principal Components	(1,5)	(1,10)	(1,15)	(1,20)	(16,20)	(6,20)
Information Utilization Rate	0.62	0.73	0.81	0.92	0.11	0.3

Table 4. Static features and the descending order of information gain

Mixed permissions and code properties	Benign	Malicious	Total	Infogain
READ_SMS	31	761	792	0.44135
getSubsciberId(TelephonyManager)	22	603	625	0.38911
WRITE_SMS	13	473	486	0.26153
getDeviceId (TelephonyManager)	314	861	1175	0.22954
SEND_SMS	375	895	1270	0.21012
READ_PHONE_STATE	36	457	493	0.20966
getSimSerialNumber (TelephonyManager)	32	452	484	0.19874
RECEIVE_SMS	14	396	410	0.19615
chmod	87	540	627	0.18205
.apk	15	390	405	0.18134
intent.action.BOOT_COMPLETED	8	336	344	0.17987
READ_CONTACTS	77	491	568	0.17236
abortBroadcast	65	467	532	0.16878
Runtime.exec()	7	301	308	0.16365
WRITE_APN_SETTINGS	113	496	609	0.15316
/system/app	39	375	414	0.14752
/system/bin	4	251	255	0.14013
getLine1Number (TelephonyManager)	174	552	726	0.13516
RECEIVE_BOOT_COMPLETED	8	183	191	0.13079
CALL_PHONE	182	503	685	0.12879

The result of Table 3 is obtained according to the algorithm of PCA, and it just lists the top 20 most relevant features. It is clear that the amount of information of the first fifteen features accounts for 81% of all the characteristics, and the information utilization rate of the first twenty features is 92%. The percentage of the lowest five ranked features occupying is far less than the top five features. The same result applies to the situation between the top fifteen and the lowest fifteen ranked features, moreover, the information utilization rate of the lowest fifteen ranked features is less than half of the percentage of the top five features. The content of Table 4 is computed using the formula (6), and to undertake the following comparative analysis, it accordingly presents the top 20 ranked features.

Afterwards, these feature data is applied for train and test the classifier. For each ten-fold cross validation, ten BP neural networks are trained and tested ten times with different training and testing sets, and eventually ten testing results of all samples are obtained. Then ten neural networks vote on the testing results. In order to gain more accurate detection rate, ten-fold cross validation is implemented on each BP neural network ten times. The following is the training process of one BP neural network. Arbitrary one among the ten BP neural networks contains an input layer, an output layer and two hidden layers. The number of the hidden layers and the number of neurons which each hidden layer contains have influence on the training performance and the detection rate. The dimension of the input layer rests with the number of input features, the number of nodes of the output layer depends on the number of classification categories.

From Fig.3, it is obviously that gradient descent with momentum and adaptive learn rate is utilized to train BP. This method helps to solve the primary two problems BP confronted with: the speed of convergence and the objective function has a local minimum point.

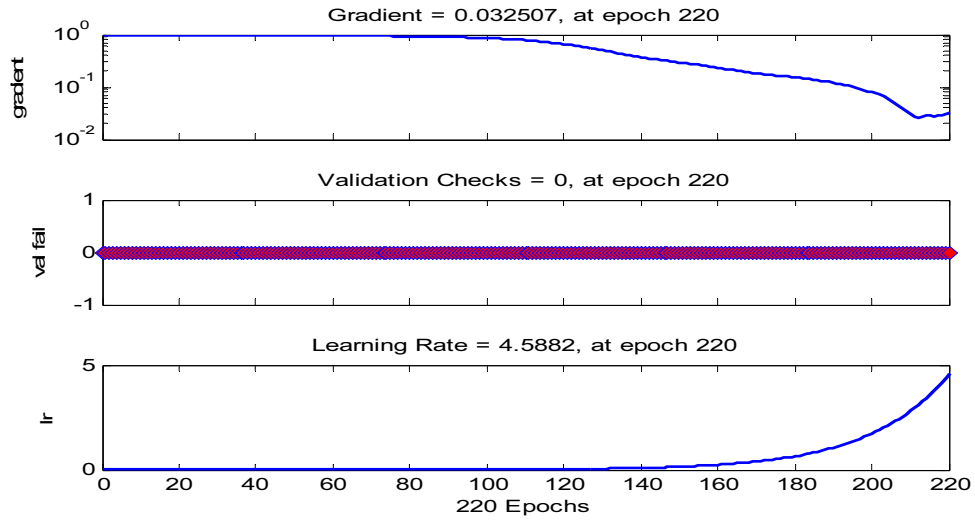

Fig. 3. The training states of one BP neural network

Fig. 4. The regression of the training result

In this paper, the whole of the ten BP neural networks is equivalent to a linear classifier. As a result, the actual outputs and the targets are painted in a picture so that the relationship between the two can be fitted, and the closer the fit line to the line Y = T, the better the performance of the training state. However, the ultimate goal of the classifier does not lie in the training performance, but the testing performance. The lowest mean squared error of the training performance is very likely to result in a problem of overlearning, which directly leads to lower detection rate of unknown malwares. As a result, the relationship between the training performance and the detection rate need to be balanced well. In principle, in the premise of a little difference between them, the detection rate is strived to improve.

5.2 BP testing and comparative analysis

After accomplishing training ten BP neural networks, they can be applied to test and verify with the different testing sets. For the same scope of features, the average detection rate of the ten BP neural net-

works is considered as the ultimate result. Fig.5 and Fig.6 respectively exhibit the accuracy and error rate of detection.

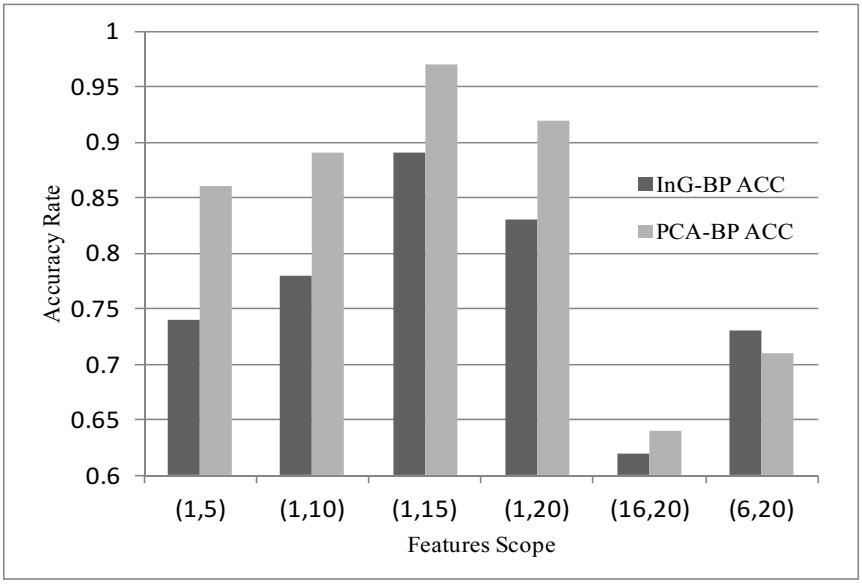

Fig. 5. The accuracy rate of the classifier

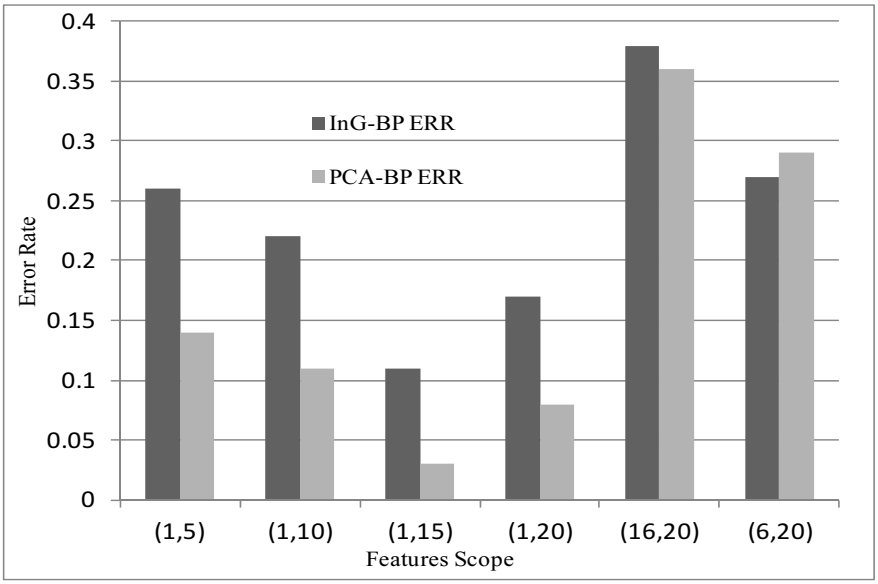

Fig. 6. The error rate of the classifier

In the general case, the classifier based on PCA performs better than that based on InG (Information Gain) except the features is ranked between $6_{th} \sim 20_{th}$. It exhibits the best performance with the top 15 ranked features used for classifying based on PCA. The information utilization rate of characteristics is not necessarily with proportional to the corresponding detection rate, and the accuracy comparison between the top 15 and 20 ranked features is right the verification for that. Under the same number of features, the detection rate with $16_{th} \sim 20_{th}$ is far inferior to that with $1_{th} \sim 5_{th}$. Fig.7 shows the error rate which is complementary to the accuracy, accordingly, it exhibits the best performance with least error rate under the top 15 ranked features.

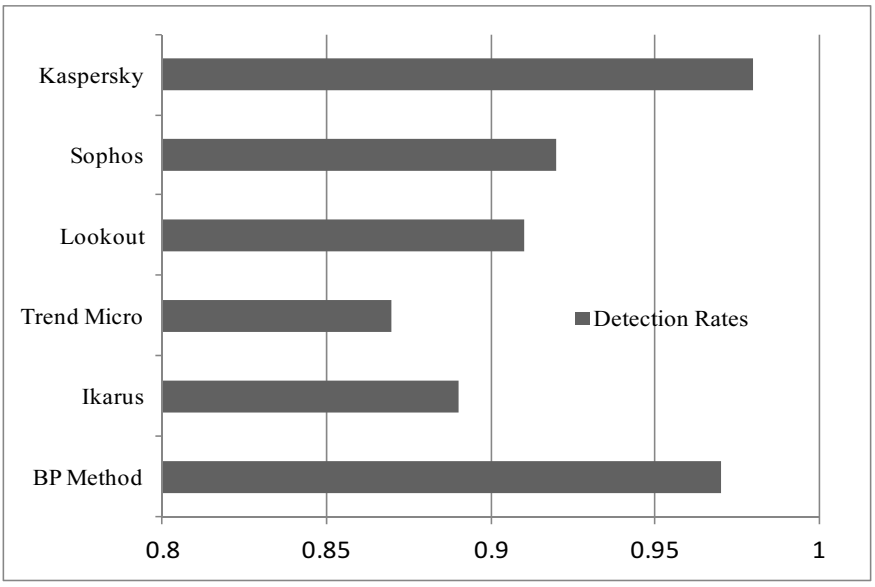

Fig. 7. The comparative analysis with some existing security software

To demonstrate abilities of our classifier to detect unknown Android malware, comparative analysis with some existing security software is undertaken and the result is as shown in Fig.7. Obviously in the figure, the optimal detection rate of our method excels the vast majority of other security software, and the accuracy of our method approximately reaches up to 97%, which is an excellent performance.

Moreover, we also conduct simple comparative analysis with part of previous works, and the result is presented in Fig.8. In Fig.8, BP Method represents our method, obviously, our detection rate excels the others. What is more, other detection rates are all lower than 0.95, only ours exceeds. The methods in [13-15] are all static analysis, they are respectively based on static code properties, permission correlation, the mixed permissions and code-based properties. Whereas, the proposed means in [22] is a kind of behavior-based detection, which is based on the properties of the mobile terminals; and [26] proposes a taint analysis tool FlowDroid. The aspects these previous works consider are less comprehensive than our method, consequently, their detection rates is inferior to ours. In conclusion, our method has excellent ability to detect unknown Android malware.

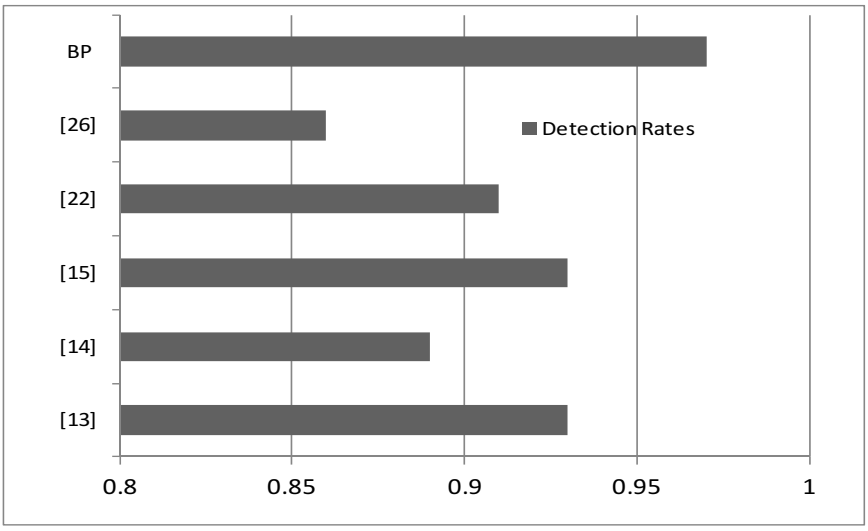

Fig. 8 The comparative analysis with some related works

6 Conclusions

In this paper, a method for detecting Android malware based on BP neural networks is proposed. We not only consider the static but also the running characteristics of the applications in the mobile terminals. We regard ten different BP neural networks as a classifier, and each neural network is conducted ten-fold cross validation ten times. The simulation results demonstrate that our approach can detect unknown Android malware accurately. As our future work, we plan to collect more Android malware samples which can be used to train and test the classifier. In addition, to detect more sophisticated malware and protect the privacy and property of users from being infringed, more efficient and timely detection methods are also deserved to be proposed.

Acknowledgment

This research is supported by Fundamental Research Funds for the Central Universities (2015YJS027).

References

[1] Apvrille, A., & Strazzere, T. (2012). Reducing the window of opportunity for Android malware Gotta catch' em all. *Journal in Computer Virology, 8*(1-2), 61-71.

[2] F-Secure. (2012). *Mobile threat report Q4 2012*. Retrieved from https://www.f-secure.com/documents/996508/1030743/Mobile+Threat+Report+Q4+2012.pdf

[3] Zhou, Y., & Jiang, X. (2012, May). *Dissecting android malware: Characterization and evolution*. Paper presented at the Proceeding IEEE Symposium on Security and Privacy, San Francisco, CA.

[4] Schmidt, A. D., Camtepe, A., & Albayrak, S. (2010, September). *Static smart phone malware detection*. Paper presented at the Proceedings of the 5th Security Research Conference (Future Security 2010), Berlin, Germany.

[5] Bläsing, T., Schmidt, A. D., Batyuk, L., Camtepe, S. A., Albayrak, S. (2010, October). *An android application sandbox system for suspicious software detection*. Paper presented at the 5th International Conference on Malicious and Unwanted Software (MALWARE'2010), Nancy, France.

[6] Kou, X., & Wen, Q. (2011, October). *Intrusion detection model based on android*. Paper presented at the 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), Shenzhen, China.

[7] Bose, A., Hu, X., Shin, K. G., & Park, T. (2008, June). *Behavioral detection of malware on mobile handsets*. Paper presented at the Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, Washington, DC.

[8] Chin, E., Felt, A. P., Greenwood, K., & Wagner, D. (2011, June). *Analyzing inter-application communication in Android*. Paper presented at the Proceedings of 9th International Conference Mobile Systems, Applications, and Services, Washington, DC.

[9] Chan, P. P. F., Hui, L. C. K., & Yiu, S. M. (2012, April). *DroidChecker: Analyzing android applications for capability leak*. Paper presented at the Proceedings of 5th ACM Conference Security and Privacy in Wireless and Mobile Networks, Tucson, AZ.

[10] Kim, J., Yoon, Y., Yi, K., & Shin, J. (2012, May). *SCANDAL: Static analyzer for detecting privacy leaks in Android applications*. Paper presented at the Proceedings of the Workshop on Mobile Security Technologies (MoST 2012), San Francisco, CA.

[11] Gibler, C., Crussell, J., Erickson, J., & Chen, H. (2012, June). *AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale*. Paper presented at the Proceedings of 5th International Conference Trust and

Trustworthy Computing (TRUST 2012), Vienna, Austria.

[12] Mann, C., & Starostin, A. (2012). *A framework for static detection of privacy leaks in android applications*. Paper presented at the Proceedings of 27th Annual ACM Symposium On Applied Computing (SAC'12), Trento, Italy.

[13] Zhang, H., Wu, J.-L, & Tang, J.-J. (2014). Neighbor Watcher: Detecting piggybacked smart phone applications with their family members. *Acta Electronica Sinica, 42*(8), 1642-1646.

[14] Zhang, R., & Yang, J.-Y. (2014). Android malware detection based on permission correlation. *Journal of Computer Applications, 34*(5), 1322-1325.

[15] Yerima, S. Y., Sezer, S., & McWilliams, G. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security, 8*(1), 25-36.

[16] Schmidt, A. D., Schmidt, H. G., Clausen, J., Yüksel, K. A., Kiraz, O., Camtepe, A., & Albayrak, S. (2008, October). *Enhancing security of Linux-based android devices*. Paper presented at the Proceedings of 15th International Linux Kongress, Hamburg, Germany.

[17] Cheng, J., Wong, S. H. Y., Yang, H., & Lu, S. (2007, June). *SmartSiren virus detection and alert for Smart phones*. Paper presented at the Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, MobiSys '07, Puerto Rico.

[18] Liu, L., Yan, G., Zhang, X., & Chen, S. (2009). Virusmeter preventing your cellphone from spies. *Recent Advances in Intrusion Detection Lecture Notes in Computer Science, 5758*, 244-264.

[19] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011, July). *Crowdroid behavior-based malware detection system*. Paper presented at the Proceedings of the 1st ACM Workshop on Security and Privacy in Smart Phones and Mobile Devices (SPSM '11), New York, NY.

[20] Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). ''Andromaly'' a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems, 38*(1),161-190.

[21] Buennemeyer, T. K., Nelson, T. M., Lagett, L. M. C., Dunning, J. P., Marchany, R. C., & Tront, J. G. (2008, Januagry). *Mobile device profiling and intrusion detection using smart batteries*. Paper presented at the Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS '08), Big Island, HI.

[22] Liu, Y.-B., Jia, S.-S., & Xing, C.-C. (2012). A novel behavior-based virus detection method for smart mobile terminals. *Discrete Dynamics in Nature and Society, 2012*, 1-12.

[23] Fuchs, A. P, Chaudhuri, A., & Foster, J. S. (2011). *ScanDroid: automated security certification of Android applications*. Retrieved from http://www.cs.umd.edu/avik/projects/scandroidascaa/

[24] Enck, W., Gilbert, P., Chun, B. G,, Cox, L. P., Jung, J., McDaniel, P., Sheth, A. (2010, October). *TaintDroid: An information-flow tracking system for real time privacy monitoring on smart phones*. Paper presented at the Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10), Berkeley, CA.

[25] Baksmali: http://code.google.com/p/smali, Accessed June 2013.

[26] Fritz, C., Arzt, S., Rasthofer, S., Eric Bodden, E., Bartel, A., Klein, J., le Traon, Y., Octeau, D., & McDaniel, P. (2013). *Technical report: Highly precise taint analysis for Android application*. Retrieved from http://www.bodden.de/pubs/TUD-CS-2013-0113.pdf