Using Hybrid Model for Android Malicious Application Detection Based on Population (Short Paper)

Zhijie Xiao[1], Tao Li[1,2](✉), and Yuqiao Wang[1]

[1] College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, Hubei, China
544247884@qq.com, litaowust@163.com, leowon@vip.qq.com
[2] Hubei Province Key Laboratory of Intelligent Information Processing and Real-Time Industrial System, Wuhan 430065, Hubei, China

Abstract. In the Android system security issue, the maliciousness of the applications is closely related to the permissions they applied. In this paper, a population-based model is proposed for detecting Android malicious application. Which is in the view of the current disadvantages of missing report, long detection period caused by features redundancy, and the instability of detection rate lead by unbalanced data of benign and malicious samples. Drawing on the idea of population in biology, each app was labeled by preprocessing. And adaptive feature vectors were automatically selected through the feature engineering. Thus the malicious application detection is carried out in the form of hybrid model voting. The experimental results show that feature engineering can remove a large amount of redundancy before classification. And the hybrid voting model can provide adaptive detection service for different populations.

Keywords: Android security · Population · Feature engineering · Security detection

1 Introduction

The various applications running on smartphones are changing people's life and communication mode. At the same time, there are some illegal elements who use malicious programs to carry out malicious charges, remote control, privacy theft and other improper acts, which seriously affect the lives of users [1–4]. Google Bouncer in Google Play can detect malicious programs, but it is not real-time, so malicious applications have been downloaded in large quantities before being detected [5]. Part of the third party application market did not carry out any form of security check before releasing the application [6]. The increasingly severe security situation of Android operating system makes it important to improve the detection efficiency of malware.

The traditional research methods are mainly divided into two categories: static analysis [7] and dynamic detection [8]. Document [9] uses static analysis to extract the function call list of executable linked format files by using the readelf. The classification algorithm is used to classify the extracted samples, so as to achieve the purpose of detecting malware. Literature [10] implements a behavior monitoring system on the

H. Gao et al. (Eds.): CollaborateCom 2018, LNICST 268, pp. 755–766, 2019.
https://doi.org/10.1007/978-3-030-12981-1_52

Android platform, dynamically monitoring the various features and events of the Android system, and classifies it by means of decision tree and regression analysis. However, due to the limited resources and power consumption of the mobile phone system, the implementation of the scheme is rather complicated. Although the traditional dynamic detection methods have their own advantages, they cannot meet the needs of timely detection of a large number of applications.

After analyzing 1100 Android application privileges using the neural network algorithm, it is found that the Android privileges often used by these programs are only a small part [11]. Literature [12] compares the use of permissions to normal software and malware, and finds that access network, reading mobile status, access to network status, and write SD cards are widely used in malicious programs and benign programs. However, malicious programs tend to use SMS related permissions, automatic startup permissions when user starting up, and changes to WIFI status permissions, while benign programs rarely use these permissions.

The above study shows that there are obvious differences in the frequency of use of different Android permissions. Moreover, the combination of normal software and malware in the combination of permissions and the tendency of categories are also quite different. Permission mechanism is the core of Android security mechanism, and the permissions of application are corresponding to the API provided by the system. Therefore, many researchers regard permissions as important detection objects.

Document [13] proposes a multilevel integrated malware detection model which extract Dalvik instructions, permissions and API as features. A three level ensemble classifier based on J48 decision tree is constructed, which has better detection. However, the technology implementation is more complex and is not suitable for detecting large-scale data. Literature [14] extracts the rights of the APK file as the feature, and uses the information gain algorithm to filter the features to implement a Android malware detection model based on the improved random forest algorithm. It achieves better detection results for feature dimensionality reduction, but there is a deficiency in solving the data imbalance, and the detection rate of Android malware is low.

AndroidProtect [15] uses static analysis to mine mass application feature values and uses dynamic target program behavior monitoring to adjust the accuracy of evaluation. Dynamic monitoring can correct deviations, but it is inefficient in dealing with large-scale applications. The author [16] proposed the similarity calculation method combined with the Euclidean distance to evaluate the dangerous trend of the Android application. By using the minimum set of privileges as the security threshold, the distance of the application to the threshold is calculated to represent the dangerous trend of the application. However, the minimum privilege set does not exist in every type of application, and the detection rate based on Euclidean distance is not good enough. To sum up, aiming at the problems existing in the current Android malware detection technology, such as the lack of detection of unknown malware, the imbalance of data in the detection process, and the low detection efficiency, this paper proposes an integrated model based on population for Android malware detection.

2 System Framework

We propose a method of Android application security detection based on the applied population, using feature engineering and mixed model voting. The model consists of 3 main modules: Feature extraction, feature engineering and safety inspection (Fig. 1).

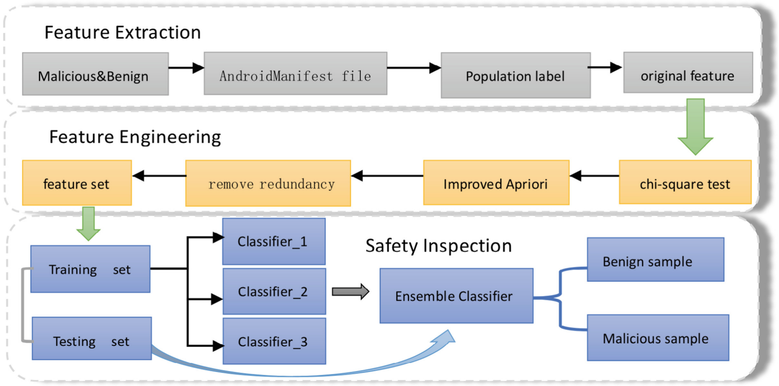


Fig. 1. The picture shows the framework of the entire hybrid detection model.

In the data preprocessing module, we optimize the code for the Scrapy frame crawler used in the article [16]. Using the application category label provided by the web site, the application is crawled by category and stored in the cloud database.

In the issue of Android system security, the permission to be applied is an important object for security analysis and evaluation. Because the same type of application implements similar functions, the required system privileges are similar. In the data preprocessing module, we annotate the collected malicious and benign samples, compare the same type of application into a population and give them a population label. Then, according to the population as a unit, the APK file is reverse processed to get the list file AndroidManifest.xml. According to the population as a unit, the APK file is reverse processed to get the list file AndroidManifest.xml. The permission information of the application is stored in the AndroidManifest.xml file. In order to improve the efficiency of application rights acquisition and carry on the batch extraction of permission, this paper uses the python program and combines the AAPT (Android Asset Packging Tool) to write the AndroidManifest.xml file parsing code. The experiment obtains the application permissions of each population according to the population, thus forming the original feature data set and storing it in the cloud database in the form of 0–1 matrix.

Chi-Square Test. The correlation of two features can be quickly calculated by chi square test. In the detection, we only need to know whether the sample features are available or not, and do not consider the number of occurrence. Therefore, it is very suitable for the processing of Android privilege characteristics. We give the definition of the population label and the specific information for each population:

Definition 1. Class $= \{C1, C2, C3\ldots Cx\}$*Cx are category labels for each population, such as flashlights, cameras, readers, players, social chat and so on.*

Definition 2. Population $= (Cx, Permission1, Permission2, Permission3\ldots Permissionm)$*. Permissionm represents the m permissions feature of the population.*

We suppose two classified variables X and Y, their ranges are {Malicious sample, Benign sample} and {Contain Permissioni, Not Contain Permissioni}, For each population, we count five frequency characteristics: MP, MN, BP and BN (Table 1).

Table 1. The table describes the meaning of each item in the chi square test.

	Contain Permissioni	Not Contain Permissioni	Sum
Malicious	MP	MN	
Benign	BP	BN	
Sum			N

So we calculate the formula:

$$\chi^2 = \frac{(\mathrm{MP} * \mathrm{BN})^2 * \mathrm{N}}{(\mathrm{MP} + \mathrm{MN}) * (\mathrm{BP} + \mathrm{BN}) * (\mathrm{MP} + \mathrm{BP}) * (\mathrm{MN} + \mathrm{BN})} \tag{1}$$

The m permissions of each population are calculated in turn to get the chi-square value χ^2. The greater the value of χ^2, the greater the possibility that the relationship between X and Y will be established. According to the check level a, find the chi square value table, get the critical value and compare it. If the lookup table value is less than the chi square value, then the permissions are redundant and need to be removed. After chi square test, a new population feature vector set D is obtained.

InG Algorithm. After the initial dimension reduction by chi square test, the information gain algorithm is applied to feature selection to further remove redundancy. In information gain, the criterion is how much information the feature can bring to the classification system. The more information it brings, the more important the feature is. For a feature, whether a system has a characteristic information quantity will change, and the difference between the front and rear information is the information quantity brought by this feature to the system. The amount of information is entropy. Information gain describes the ability of attribute X to discriminate sample Y.

We use Y to represent the random variable of the application category, Y= {Malicious, Benign}. For each permission Permissioni in a population, its information entropy H (x) is:

$$\mathrm{H}(\mathrm{X}) = -\sum\nolimits_i P(X_i) lb P(X_i) \tag{2}$$

The conditional information entropy H (X|Y) of X after the known variable Y is:

$$\mathrm{H(X|Y)} = -\sum\nolimits_j P(y_i)\sum\nolimits_i P(x_i|y_j)lbP(x_i|y_j) \tag{3}$$

Mutual information between variables X and Y is:

$$\mathrm{MI(X,Y)} = \mathrm{H(X)} - \mathrm{H(X|Y)} = \sum\nolimits_{x,y} P(x,y)lb\frac{P(x,y)}{P(x)*P(y)} \tag{4}$$

The greater the information gain of the privilege feature Permissioni, the greater the correlation between the feature and the category. After calculating the information gain of each privilege feature, the features are arranged in descending order according to information gain. The feature set of each population is selected adaptively through experiments to train and test the classifier. At this point, we get the data set S for each population.

3 Security Detection Algorithm

3.1 SVM Algorithm Based on Bagging

Android malicious sample is difficult to collect compared to the benign sample, so it is easy to appear the imbalance between the benign sample and the malignant sample, which makes the classifier more biased in the benign sample, and eventually leads to the low detection rate. At the same time, the SVM algorithm is sensitive to the samples at the boundary. If there are misplaced samples on the boundary, the stability of the classifier will be greatly affected.

Using Bagging to construct balanced data sets for benign and malicious samples can reduce the impact of imbalanced data on experimental results. In order to improve the detection efficiency, this paper adopts the Linear Support Vector Machine to design the SVM algorithm based on Bagging. The concrete steps are as follows:

1. Randomly divide 70% data from S as training set S_train and 30% data as test machine S_test, and the range of data set S is {Maliouse, Benign}.
2. Using Bagging to extract m benign samples and M malicious samples to form a new training data set $D_i = (X_i, Y_i)$, the digital m is generated by a random number generator, in which $X_i = (X_{i1}X_{i2}X_{i3}\ldots X_{im})$, that is, the information of an application sample. $Y_i = \{0, 1\}$, of which 0 represents benign samples, and 1 indicates malignant samples.
3. The SVM classifier is represented as:

$$\begin{cases} \min_\lambda \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i^T x_j - \sum_{i=1}^n \lambda_i \\ \sum_{i=1}^n y_i\lambda_i = 0, 0 \le \lambda_i \le \alpha, i = 1, 2, \ldots n \end{cases} \tag{5}$$

A is a penalty term. The two step programming method is used to solve the model and get the Lagrange multiplier. Then the LSVM parameter is solved as follows:

$$\omega = \sum\nolimits_i \lambda_i y_i x_i \tag{6}$$

$$\lambda_i \left(y_i \sum\nolimits_i \lambda_j y_j x_i^T x_j + c - 1 \right) = 0 \tag{7}$$

$$\mathrm{C(x)} = sign\left(\sum\nolimits_{i=1}^{n} \lambda_i y_i x_i^T + c \right) \tag{8}$$

4. Repeat 3 times steps 2 and 3, get three LSVM base classifiers, and combine 3 base classifiers into SVM ensemble classifier CC. The test sample X in S_test is put into the integrated classifier CC, and the result is cast. The voting formula is:

$$\mathrm{CC(x)} = voting(C_1(x), C_2(x), C_3(x), \ldots C_k(x)) = \delta\left(\sum\nolimits_i singn(C_i(x) = y) \right) \tag{9}$$

If $C_i(x) = y$, in that way sign(r) = 1, conversely, sign(r) = −1.
If $\sum_i singn(C_i(x) = y) > 0$, $\delta(\mathrm{q}) = 1$, samples is malware, otherwise $\delta(\mathrm{q}) = 0$, indicating benign.

3.2 A Classifier Based on Improved Naive Bayes

The idea of naive Bayes is to obtain a priori probability by training samples, and the posterior probability of events is obtained according to the prior probabilities and sample data information. Finally, the event is attributed to the maximum of the posterior probability. The Bias classifier is derived from the formula, so it has a stable classification efficiency, low sensitivity to real data and high algorithm efficiency.

In a population, an application of information $\mathrm{X} = (X_1, X_2, X_3 \ldots X_k)$, C_i represents whether the application is benign or malignant. So we can get the posteriori hypothesis.

$$\mathrm{P}(C_i|\mathrm{X}) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{10}$$

Naive Bayes assumes that the influence of a feature on classification is independent of other attributes, so we can known that:

$$P(X|C_i) = \prod\nolimits_{k=1}^{n} P(X_k|C_i) \tag{11}$$

However, in practical applications, permissions are independent of each other, which is not valid in Android malware detection. After multiple removal of redundancy after the feature engineering module, we get the data set S, which is very representative of the classification, so the naive Bayes can be applied to the detection scene.

$$\mathrm{P}(C_i|\mathrm{X}) = \frac{P(C_i) \prod_{k=1}^{n} P(X_k|C_i) \frac{1}{\chi_k^2} \sum_{i=1}^{n} \chi_i^2}{P(X)} \tag{12}$$

At the same time, in the detection scene, the extent of the influence of different permissions on malicious detection is different. The simple Bias cannot show the difference of the feature [17]. Therefore, we introduce the weight influence factor $\frac{1}{\chi_k^2}\sum_{i=1}^{n}\chi_i^2$ to the posterior probability of the Bias classification, and the influence factor represents the proportion of the influence that the authority brings to the classification.

4 Experimental Results and Analysis

4.1 Experimental Data

At present, we have climbed 62 categories and totaling 325371 Android benign applications from AnZhi [18]. Malicious samples from VirusShare [19]. For the following reasons, we chose two groups of cameras and flashlights as experimental objects. First, the two types of applications, such as cameras and flashlights, have clear functional boundaries. For a app, it is easier to distinguish whether it belongs to a flashlight or camera category from its main permission statement and the application description that is filled in when it is uploaded. Second, flashlights and cameras are widely used by users, while mobile devices are produced with flashlights and cameras, but almost every user will install another flashlight or camera app because of the individual needs.

4.2 Experimental Results

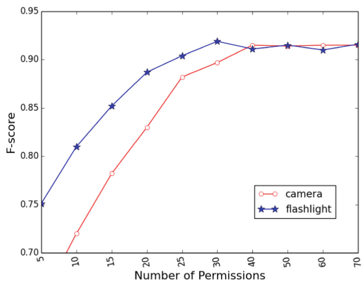
Taking Android permission meaning as background knowledge, chi square test and InG algorithm are used to perform two times of feature de redundancy. In the module of the feature engineering, we take all the rights characteristics of the population as the experimental set, and get 128 feature vectors of the flashlight, and 134 camera population characteristics. After screening, each authority is scored. The higher the score, the greater the information gain and the better the privilege of the categorization result. List 10 rights information with the highest score in two populations.

In the table, the top 10 features of the population and flashlight population information gain are listed. From the table, we can see that there are differences in the distribution of permission characteristics among the two populations, and the information gain of the same permissions in different populations is also different. It can be seen that different populations have different requirements and distributions in terms of authority, which shows that the characteristics of populations are different (Table 2).

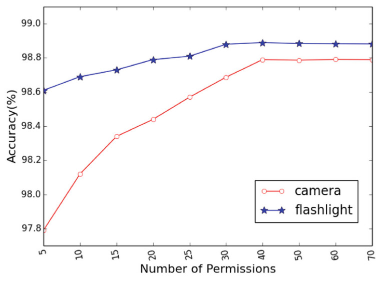
Table 2. The top10 information gain of the flashlight and camera population.

Rank	Flashlight	InG
1	CAMERA	0.538
2	FLASHLIGHT	0.511
3	INTERNET	0.384
4	ACCESS_NETWORK_STATE	0.347
5	ACCESS_COARSE_LOCATION	0.315
6	ACCESS_WIFI_STATE	0.280
7	ACCESS_FINE_LOCATION	0.279
8	GET_TASKS	0.247
9	WAKE_LOCK	0.231
10	WRITE_SETTINGS	0.206
Rank	**Camera**	**InG**
1	WRITE_EXTERNAL_STORAGE	0.442
2	INTERNET	0.418
3	CAMERA	0.397
4	ACCESS_NETWORK_STATE	0.385
5	READ_PHONE_STATE	0.379
6	ACCESS_WIFI_STATE	0.341
7	WAKE_LOCK	0.326
8	ACCESS_COARSE_LOCATION	0.303
9	VIBRATE	0.285
10	GET_TASKS	0.247

In order to find optimal subset, we count the Accuracy and F1-score under the number of different privileges, as shown in the following figure (Fig. 2).



(a)



(b)

Fig. 2. (a) and (b) respectively represent the Accuracy and F1-score of two different populations when increasing the number of permissions.

From the graph, it is found that in the flashlight population, the set of the top 30 permissions is the optimal set. In the camera population, the set of the top 40 permissions is the optimal set, because at this time the two populations have reached a higher and more stable accuracy. Considering that we normally tend to select the smallest subset of permissions. Adding more permissions does not help on improving but may introduce redundant, noisy and irrelevant information (Table 3).

Table 3. The F1 score of the four classification algorithms on two populations. Top20 represents the first 20 features of the information gain rankings.

		NB	SVM	RF	Mixture
Flashlight	Top 20	82.3	88.0	88.4	88.7
	Top 30	84.2	89.2	89.7	92.0
	Top 50	88.4	93.7	90.6	91.8
	Top 90	92.3	91.5	91.0	91.2
	Data S	90.7	92.1	90.9	91.0
Camera	Top 20	81.4	86.5	85.1	83.0
	Top 40	83.6	88.1	87.6	91.5
	Top 60	88.9	90.2	89.7	91.5
	Top 90	90.5	90.9	91.1	91.3
	Data S	90.8	91.3	89.9	90.8

The table shows two groups of flashlights and cameras. The purpose is to evaluate the effect of feature engineering. The classification algorithms are naive Bayes, LSVM, RF and the hybrid voting algorithm in this paper(Mixture), in which LSVM and RF use the default parameters in the Weka tool.

The 10 cross validation method is used to train the classifier and get the result.

It can be seen from the table that the classification results of RF and mixture are better. In the flash population, when the number of permissions reaches the first 30, the F1 value gradually stabilizes in the 92.0%. In the camera population, when the number of permissions reaches 40, the F1 value is gradually stabilized at the 91.5%.

According to Occam's Razor, Entities should not be multiplied unnecessarily, for these two populations, we respectively take the top 30 and the top 40 of the rights for classification. This shows that the feature engineering screening method proposed in this paper can eliminate a large number of redundant features and improve the efficiency of detection.

For non-equilibrium data, 3 sets of data sets are designed to test the effectiveness of the proposed method. After the characteristic engineering module was processed, the flashlight population data set SF and the camera population data set SC were obtained. SF contains 300 malicious applications, 1200 benign applications. And 280 malicious applications and 1250 benign applications in SC. Three data sets of two populations, A, B, and C, are extracted randomly by no return (Table 4).

Table 4. Data distribution of data sets A, B, C in two populations. 280(M)+625(B) represents 280 malicious samples and 625 benign samples.

Population	A	B	C
Camera	280(M)+280(B)	280(M)+625(B)	280(M)+1250(B)
Flashlight	300(M)+300(B)	300(M)+600(B)	300(M)+1200(B)

Three samples of A, B and C in two populations were sampled without return. Each dataset was divided into 10 averages, of which 8 were used as training sets and 2 as test sets. Then, our algorithm and the method of Wang [16], Zhang [15] are used to classify these data sets simultaneously. For the results obtained, the classification indexes Precision, Recall, and Accuracy are calculated. Repeat the 10 cross validation to get the average value. The results are shown in the following table (Table 5).

Table 5. The results of different methods are compared to the A, B, C data sets. F represents the flashlight population, and C represents the camera population.

Data	Method	Precision		Recall		Accuracy	
		F	C	F	C	F	C
A	Mixture	96.7	96.2	98.9	98.8	97.8	98.1
	Wang	95.4	94.8	97.4	96.3	96.0	97.5
	Zhang	96.5	97.3	98.5	97.4	97.1	97.9
B	Mixture	96.8	96.5	97.9	98.5	97.6	97.7
	Wang	94.3	93.3	94.6	97.1	95.3	96.3
	Zhang	95.9	98.0	92.2	92.5	96.5	96.7
C	Mixture	96.3	93.1	98.7	98.4	97.6	97.6
	Wang	93.3	92.9	95.6	95.7	95.1	95.5
	Zhang	94.6	96.4	95.7	92.3	96.1	96.7

For the two unbalanced data sets of B and C, the detection rate of the three methods decreased when the imbalances increased, but the variation of Mixture in the detection rate was smaller and stable at a better level. For a detection method, the negative impact of malware to be detected as a benign software is far greater than that of the benign software being misrepresented as malware, so the Recall is a very important index. A slight reduction in accuracy and improvement in Recall are of great importance in the detection of Android applications.

5 Conclusion

This article is based on the idea of biological population. The static feature of Android is removed by feature engineering, so that the feature is pruned, the model training time is shortened, and the detection efficiency is improved. Then we use mixed model voting to detect malicious applications. Experiments on two commonly used flashlight and camera populations show that malware detection rates reach 98.7% and 98.4% respectively on unbalanced datasets. It is limited to detect only the permissions for malicious application detection, but the method is simple and easy to implement. It can detect a large number of applications at the same time. In the case of good detection rate, it also has a high detection efficiency.

Acknowledgement. Authors are partially supported by Major projects of the Hubei Provincial Education Department (No. 17ZD014), Hubei college students' Innovation and Entrepreneurship Training Program project (No. 201610488020), National defense pre research fund of Wuhan University of Science and Technology (No. GF201712) and Colleges and Universities in Hubei Provincial College Students' Innovation Entrepreneurial Training Program (No. 201710488027).

References

1. The development of the China Mobile Internet and its security report (2017). [EB/OL]. http://www.isc.org.cn/zxzx/xhdt/listinfo-35398.html. Accessed 17 May 2017/08 Mar 2018
2. Yi, L., Zhang, N., Liu, D.: Study on mobile malware situation and trends. Inf. Commun. Technol. **7**(2), 75–79 (2013)
3. Jiang, X., Zhou, Y.: A survey of Android malware. In: Jiang, X., Zhou, Y. (eds.) Android Malware, pp. 3–20. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-7394-7_2
4. Chu, J., Zheng, L.: The security analysis of Android OS. Microcomput. Appl. **20**(7), 1–3 (2013)
5. Peng, H., Gates, C., Sarma, B., et al.: Using probabilistic generative models for ranking risks of Android apps. In: ACM Conference on Computer and Communications Security, pp. 241–252. ACM (2012)
6. Zhou, Y., Jiang, X.: Dissecting Android malware: characterization and evolution. In: IEEE Symposium on Security and Privacy, pp. 95–109. IEEE (2012)
7. Feng, Y., Anand, S., Dillig, I., et al.: Apposcopy: semantics-based detection of Android malware through static analysis. In: ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 576–587. ACM (2014)
8. Petsas, T., Voyatzis, G., Athanasopoulos, E., et al.: Rage against the virtual machine: hindering dynamic analysis of Android malware. ACM (2014)
9. Schmidt, A.D., Bye, R., Schmidt, H.G., et al.: Static analysis of executables for collaborative malware detection on Android. In: IEEE International Conference on Communications, pp. 1–5. IEEE (2009)
10. Shabtai, A., Elovici, Y.: Applying behavioral detection on Android-based devices. In: Cai, Y., Magedanz, T., Li, M., Xia, J., Giannelli, C. (eds.) MOBILWARE 2010. LNICST, vol. 48, pp. 235–249. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17758-3_17
11. Barrera, D., Oorschot, P.C.V., Somayaji, A.: A methodology for empirical analysis of permission-based security models and its application to Android. In: ACM Conference on Computer & Communications Security, pp. 73–84. ACM (2010)
12. Zhou, Y.: Dissecting Android malware: characterization and evolution. **4**(3), 95–109 (2012)
13. Zhang, W., Ben, H., Zhang, K., et al.: Malware detection techniques by mining massive behavioral data of mobile Apps. J. Integr. Technol. **5**(2), 29–40 (2016)
14. Yang, H., Xu, J.: Android malware detection based on improved random forest algorithm. J. Commun. **38**(4), 8–16 (2017)

15. Zhang, T., Li, T., Wang, H., Xiao, Z.: AndroidProtect: Android apps security analysis system. In: Wang, S., Zhou, A. (eds.) CollaborateCom 2016. LNICST, vol. 201, pp. 583–594. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59288-6_58
16. Wang, H., Li, T., Zhang, T., Wang, J.: Android apps security evaluation system in the cloud. In: Guo, S., Liao, X., Liu, F., Zhu, Y. (eds.) CollaborateCom 2015. LNICST, vol. 163, pp. 151–160. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28910-6_14
17. Peng, H.: Discussion on the selective weighted Bias classification method. Zhongshan University (2010)
18. Anzhi[EB/OL]. http://www.anzhi.com/
19. VirusShare [EB/OL]. https://virusshare.com/