Malicious Application Traffic Detection and Identification for Mobile Android Devices

Geng Niu
Shaanxi Police College, Xi'an, Shaanxi 710021, China
Email: gn67iq@126.com

Keywords: mobile, Android device, malicious application, detection, identification, traffic feature

Received:

With the popularity of Android devices, the number of malicious applications has been increasing. This paper briefly introduced malicious applications for Android devices, used a sensitivity coefficient-based feature selection method to select traffic features, detected, and identified malicious application traffic with k-means, support vector machine (SVM) and multi-layer perceptron (MLP) methods, and conducted experiments at CIC-AndMal2017. It was found that the accuracy was high when 40 features were selected. The running time of the MLP method was the shortest, 0.02 s. The accuracy of the K-means algorithm was 86.75%, showing poor performance, and the accuracy of the MLP method was 99.87%, showing the best performance. The experimental results demonstrate the effectiveness of the MLP method for monitoring and identifying malicious application traffic. The MLP method can be applied to actual mobile Android devices.

Povzetek: Prispevek se ukvarja z zlonamernimi aplikacijami na operacijskem sistemu Android - jih našteje in podrobno analizira z več metodami.

1 Introduction

With the development of mobile technology [1], devices such as cell phones and tablet computers have been more and more widely used [2] and become an important part of life [3], among which mobile devices using Android have the largest proportion [4]. With the rapid development of Android devices, many new problems have emerged. While the application traffic has increased significantly, the number of malicious applications has also increased significantly due to the open-source characteristic of Android devices [5, 6], which brings huge security problems to the mobile terminal [7]. Mobile applications are more diverse and complex than applications in personnel computers, and the traditional way of traffic detection and identification is not applicable to mobile. Therefore, how to detect and identify malicious application traffic on mobile has received a lot of attention from researchers [8]. Afonso et al. [9] extracted features from Android application programming interface (API) calls and system call traces to identify applications through a machine learning method. They found through experiments that the method had a detection rate of 96.66%. Faruki et al. [10] designed a method called AndroSimilar to detect unknown applications by extracting statistically robust features to generate signatures and finding regions that are statistically similar to known malicious applications. Milosevic et al. [11] proposed two methods to analyze malicious applications, one based on permissions and the other based on source code analysis utilizing a bag-of-words representation model, and found through experiments that the F1 values of these two methods are 95.1% and 89%. Chakraborty et al. [12] designed an integrated clustering and classification (EC2) approach, conducted experiments on the DREBIN dataset, and found that EC20 could accurately detect malicious application families, providing an emerging early warning system. In this paper, the selection of traffic features and the detection and identification of malicious application traffic were investigated, several machine learning methods were compared, and experiments were performed at CIC-AndMal2017 to understand the reliability of the method. This study provides some theoretical support for better implementation of the secure operation of Android devices on mobile.

2 Malicious applications for Android devices

Android is a Linux-based operating system [13] that uses the Dalvik virtual machine and has been very widely used in different brands of mobile devices. The applications for Android devices are very diverse, covering games, music, social communication, office, video, etc., which has led to the flooding of malicious applications [14]. Main malicious applications are shown below.

(1) Privacy theft: Malicious applications can steal users' privacy through the network or short messages, modify permissions of calls and short messages, steal users' personal information, various account passwords, etc., or hide important information of users.

(2) Downloading applications: The user is misled into downloading malicious applications by means of fake application names and forged buttons.

(3) Malicious fee deduction: Users are misled into paying for international calls, sending short messages, and subscribing to paid services in the background.

(4) Malicious propagation: Users install and propagate malicious applications without knowing it.

(5) Remote control: The phone is controlled remotely through applications to download unknown applications, uninstall antivirus applications, and download or destroy the user's data.

The 2019 Android Malicious Software Report released by 360 Security Brain shows the number of new malicious applications by month on the mobile terminal in 2019, as shown in Figure 1.

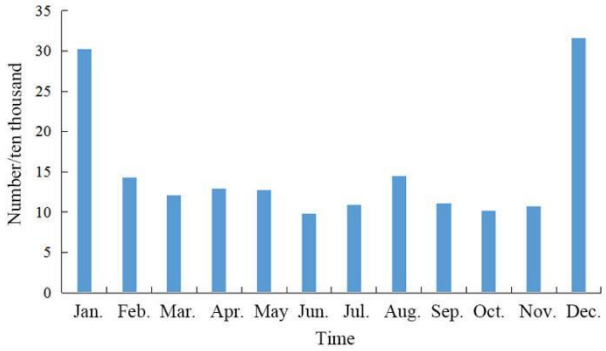

Figure 1: Number of new malicious applications by month

It was seen from Figure 1 that in January and December, the number of new malicious apps was the highest, above 300,000, probably because people used their mobile Android devices for social entertainment activities more frequently around the Chinese New Year holidays, giving malicious applications an opportunity to spread. Among the malicious applications that appeared, the proportion of fee-consuming applications was the highest, reaching 46.8%, followed by privacy theft applications (41.9%). They are two main malicious applications.

Malicious applications generate traffic during the operation. Malicious applications can be found by processing and analyzing the traffic generated by applications [15].

3 Detection and identification method for malicious application traffic

3.1 Traffic feature selection

When using algorithms for detection and recognition, if all the features are put into algorithms, it will cause a dimensional disaster that will consume a lot of time to make the algorithm converge [16]. In order to obtain better detection and recognition performance, it is necessary to select traffic features and reduce the number of features, thus improving calculation efficiency and reducing difficulties.

This paper uses a sensitivity coefficient-based method. It is assumed that in the dataset, the number of samples that belongs to class c_j is $|c_j|$ and the number of feature t_i in c_j is $n(t_i, c_j)$. The occurrence frequency of feature t_i in class c_j is written as:

$$CR(t_i, c_j) = \frac{n(t_i, c_j)}{|c_j|}.$$

Let the malicious application class be c_m, then the occurrence frequency of feature t_i in c_m is written as:

$$MR(t_i) = \frac{n(t_i, c_m)}{|c_m|}.$$

Let the normal application class be c_b, the occurrence frequency of feature t_i in c_b is written as:

$$BR(t_i) = \frac{n(t_i, c_b)}{|c_b|}.$$

The frequency difference of feature t_i in two classes is written as:

$$MBR(t_i) = \frac{MR(t_i)}{MR(t_i)+BR(t_i)}.$$

Ultimately, the sensitivity coefficient of feature t_i is written as:

$$SC(t_i) = \frac{2\times MBR(t_i)\times MR(t_i)}{MBR(t_i)+MR(t_i)}.$$

The sensitivity coefficient of different features is ranked. The larger the value of the sensitivity coefficient is, the higher the rank of the feature is.

3.2 Detection and identification method

(1) K-means clustering [17]

The principle of K-means is to classify the dataset into k clusters according to the size of the distance between samples to achieve the classification of samples. It is assumed that a dataset is divided into k clusters, C_k, then the objective of the algorithm is to minimize square error E, i.e.,

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - u_i\|_2^2$$

$$u_i = \frac{1}{|C_i|} \sum_{x \in c_i} x$$

(2) Support vector machine

Support vector machine (SVM) is a commonly used classification algorithm [18]. There is a sample set, $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, $x_i \in R^n$, $y_i \in (-1,1)$. In order to classify the samples, it is necessary to find the hyperplane: $w^T x + b = 0$, where w is the weight value and b is the threshold value. In order to maximize the classification interval,

$$min \frac{1}{2}\|w\|^2,$$

$$s.t.\, y_i(w^T x_i + b) \geq 1.$$

The above equation is solved using the Lagrange method. Let α_i be the Lagrange multiplier, then

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i x_j,$$

$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1,2,\cdots,m.$$

Ultimately, the classification function of the SVM is written as:

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i x_i^T x + b.$$

(3) Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a kind of neural network [19], and the simplest MLP has a three-layer structure. If the number of nodes in the input layer is P, whose excitation function is the identity function, the number of nodes in the hidden layer is Q, whose excitation function is the Sigmoid function, and the number of nodes in the output layer is L, whose excitation function is the linear function, then the output of the hidden layer is expressed as: $f(w_1x + b_1)$, and the output of the output layer is expressed as: $F(w_2Y + b_2)$, where w_1 and w_2 are weights, b_1 and b_2 are biases, and Y is the output of the hidden layer. Ultimately, the MLP model is written as:

$$F(x) = F\left(\mathrm{b}^{(2)} + \mathrm{w}^{(2)}\left(\mathrm{S}\left(\mathrm{b}^{(1)} + \mathrm{w}^{(1)}\mathrm{x}\right)\right)\right).$$

The MLP method is trained by comparing the output of the model with the desired output and updating the weight according to the error. The update formula is:

$$\nabla w = \eta(d - w^T x)x,$$

where η is the learning factor, d is the expected value, w is the initial weight, and x is the input value.

4 Results and analysis

4.1 Experimental setup

The experimental operating system was Windows 10. The model number of the CPU was Intel(R) Core(TM) i5-8250U. The memory was 8 G. The programming language was Python. The experimental data set came from the Canadian Android Malware Dataset (CIC-AndMal2017) [20], collected in 2015, 2016, and 2017 Google play market, which included 4354 malicious applications (Malware) and 6500 benign samples (Benign). Malicious applications included four categories, 42 malicious families, as shown in Table 1. The dataset provides network traffic features (.pcap file), and they are more than 80 features extracted using CICFlowMeter-V3 [21]. The dataset used for the experiments is shown in Table 2.

Malicious application category	Malicious family name
Adware	Dowgin family
	Ewind family
	Feiwo family
	Gooligan family
	Kemoge family
	koodous family
	Mobidash family
	Selfmite family
	Shuanet family
	Youmi family
Ransomware	Charger family
	Jisut family
	Koler family
	LockerPin family
	Simplocker family
	Pletor family
	PornDroid family
	RansomBO family
	Svpeng family
	WannaLocker family
Scareware	AndroidDefender
	AndroidSpy.277 family
	AV for Android family
	AVpass family
	FakeApp family
	FakeApp.AL family
	FakeAV family
	FakeJobOffer family
	FakeTaoBao family
	Penetho family
	VirusShield family
SMS Malware	BeanBot family
	Biige family
	FakeInst family
	FakeMart family
	FakeNotify family
	Jifake family
	Mazarbot family
	Nandrobox family
	Plankton family
	SMSsniffer family
	Zsone family

Table 1: Malicious application traffic categories and their families

		Training set	Test set	Total
Malicious application traffic	Adware	26143	8714	34857
	Ransomware	28593	9531	38124
	Scareware	27554	9184	36738
	SMS Malware	25654	8551	34205
Benign traffic	Benign	108820	36274	145094

Table 2: Experimental data set

4.2 Evaluation indicators

The classification problem of the traffic is generally evaluated on the basis of the confusion matrix (Table 3).

		Classification results	
		Normal application	Malicious application
The real situation	Normal application	TP	FN
	Malicious application	FP	TN

Table 3: Confusion matrix

The specific indicators are as follows.

(1) Accuracy: $A = \frac{TP+TN}{TP+TN+FP+FN}$

(2) Precision: $P = \frac{TP}{TP+FP}$

(4) Recall rate: $R = \frac{TP}{TP+FN}$

(5) F1 value: $F1 = \frac{2\times TP}{2\times TP+FN+FP}$

4.3 Analysis of results

First, the accuracy was compared under different numbers of features using the K-means method, and the results are shown in Figure 2.

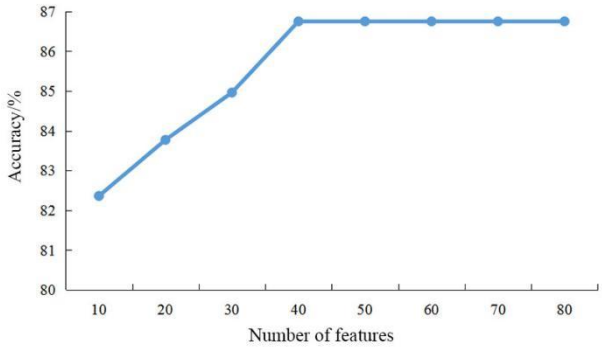

Figure 2: Accuracy under different number of features

It was seen from Figure 2 that the accuracy of the algorithm was 82.36% when the number of features was 10 and 86.75% when the number of features reached 40; after 40 features, the continued increase in the number of features did not improve the accuracy of the algorithm, which always remained at 86.75%. Therefore, in the subsequent experiments, the number of features was determined as 40.

The comparison of the running time of different algorithms under 40 features is shown in Figure 3.

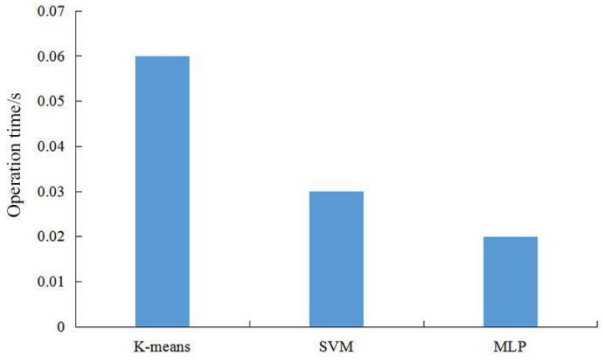

Figure 3: Comparison of the running time between different algorithms

It was seen from Figure 3 that when detecting and identifying malicious application traffic, the K-means method required the longest running time, 0.06 s, the SVM method was the second-longest, 0.03 s, which was 50% shorter than the K-means method, and the MLP method requires the shortest running time, 0.02s, which was 66.67% shorter than the K-means method and 33.33% shorter than the SVM method. These results indicated that the MLP method was the most efficient and could achieve the detection and identification of malicious application traffic in the shortest time.

The performance of different algorithms for detecting and identifying malicious application traffic is shown in Figure 4.

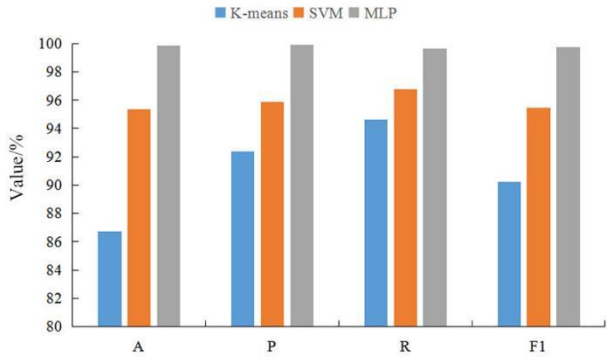

Figure 4: Performance comparison of different algorithms

Figure 4 shows that the accuracy of the K-means, SVM, and MLP methods were 86.75%, 95.36%, and 99.87%, respectively, and the MLP method had the highest accuracy rate, which was 13.12% higher than the K-means method and 4.51% higher than the SVM method; the precision of the three algorithms were 92.36%, 95.89%, and 99.91%, respectively, and the MLP method also had the highest precision; the recall rate of the K-means, SVM and MLP methods were 94.65%, 96.77%, and 99.65%, respectively, and the recall rate of the MLP method was 5% higher than that of the K-means method and 2.88% higher than that of the SVM method; the F1 value of the MLP method was 99.77%, which was 9.56% larger than the K-means method and 4.31% larger than the SVM method.

These results suggested that among the three algorithms, the MLP method had the best performance in detecting and identifying malicious application traffic and could classify malicious application traffic effectively.

5 Conclusion

This paper focuses on the problem of detecting and identifying malicious application traffic on mobile Android devices. A sensitivity coefficient-based approach was designed to select traffic features, and the performance of three algorithms, K-means, SVM, and MLP, in detecting and identifying traffic was compared. It was found through experiments on CIC-AndMal2017 that:

(1) the accuracy of the algorithm has reached stability when 40 features were used;

(2) the MLP method had the shortest running time, 0.02 s, when performing detection and recognition;

(3) compared with the K-means and SVM methods, the MLP method had higher accuracy, precision, recall rate, and F1 value, showing better performance.

The experimental results verified the reliability of the MLP method in detecting and identifying malicious application traffic. The MLP method can be further promoted and applied in actual mobile Android devices, which is conducive to better secure use of Android devices.

References

[1] Zhu F (2018). Research on intelligent english oral training system in mobile network. *Informatica: An International Journal of Computing and Informatics*, 42, pp. 259-264.

[2] Kang B J, Yerima S Y, Sezer S, Mclaughlin K (2016). N-gram Opcode Analysis for Android Malware Detection. *IJCSA*, 1, pp. 231-255. https://doi.org/10.22619/IJCSA.2016.1001011

[3] Cen L, Gates CS, Si L, Li N (2015). A Probabilistic Discriminative Model for Android Malware Detection with Decompiled Source Code. *IEEE Transactions on Dependable and Secure Computing*, 12, pp. 400-412. https://doi.org/10.1109/TDSC.2014.2355839

[4] Almin S B, Chatterjee M (2015). A Novel Approach to Detect Android Malware. *Procedia Computer Science*, 45, pp. 407-417. https://doi.org/10.1016/j.procs.2015.03.170

[5] Aresu M, Ariu D, Ahmadi M, Maiorca D, Giacinto G (2015). Clustering android malware families by http traffic. *In 2015 10th International Conference on Malicious and Unwanted Software (MALWARE'15)*, 2015, pp. 128-135.

[6] Wu Q, Zhu X, Liu B (2021). A Survey of Android Malware Static Detection Technology Based on Machine Learning. *Mobile Information Systems*, 2021, pp. 1-18. https://doi.org/10.1155/2021/8896013

[7] Jang J W, Yun J, Mohaisen A, Woo J, Kim H K (2016). Detecting and classifying method based on similarity matching of Android malware behavior with profile. *Springerplus*, 5, pp. 273. https://doi.org/10.1186/s40064-016-1861-x

[8] Wang Z, Li C, Yuan Z, Guan Y, Xue Y (2016). DroidChain: A novel Android malware detection method based on behavior chains. *Pervasive and Mobile Computing*, 32, pp. 3-14. https://doi.org/10.1016/j.pmcj.2016.06.018

[9] Afonso V M, Amorim M, Grégio A R A, Junquera GB, de Geus P L (2015). Identifying Android malware using dynamically obtained features. *Journal of Computer Virology & Hacking Techniques*, 11, pp. 9-17.

[10] Faruki P, Laxmi V, Bharmal A, Gaur M S, Ganmoor V (2015). AndroSimilar: Robust signature for detecting variants of Android malware. *Journal of Information Security & Applications*, 22, pp. 66-80. https://doi.org/10.1016/j.jisa.2014.10.011

[11] Milosevic N, Dehghantanha A, Choo K (2017). Machine learning aided Android malware classification. *Computers & Electrical Engineering*, 61, pp. 266-274. https://doi.org/10.1016/j.compeleceng.2017.02.013

[12] Chakraborty T, Pierazzi F, Subrahmanian V S (2017). EC2: Ensemble Clustering and Classification for Predicting Android Malware Families. *IEEE Transactions on Dependable & Secure Computing*, 17, pp. 262-277. https://doi.org/10.1109/TDSC.2017.2739145

[13] Ahmed, Abukmail, Paul, et al. (2014). Automatic Android-based Wireless Mesh Networks. *Informatica: An International Journal of Computing and Informatics*, 38, pp. 313-320.

[14] Singh J, Gera T, Ali F, Thakur D, Singh K, Kwak K S (2021). Understanding Research Trends in Android Malware Research Using Information Modelling Techniques. *Computers, Materials and Continua*, 66, pp. 2655-2670. https://doi.org/10.32604/cmc.2021.014504

[15] Chen X R, Shi S S, Xie C L, Yang Z, Guo Y J, Fang Y, Wen W P (2021). SUIP: An Android malware detection method based on data flow features. *Journal of Physics: Conference Series*, 1812, pp. 1-8. https://doi.org/10.1088/1742-6596/1812/1/012010

[16] Wang L, Gao Y, Gao S, Yong X (2021). A New Feature Selection Method Based on a Self-Variant Genetic Algorithm Applied to Android Malware Detection. *Symmetry*, 13, pp. 1290. https://doi.org/10.3390/sym13071290

[17] Fan C (2022). Evaluating employee performance with an improved clustering algorithm. *Informatica: An International Journal of Computing and Informatics*, 46, pp. 123-128. https://doi.org/10.31449/inf.v46i5.4079

[18] Dey A, Chowdhury S (2020) Probabilistic weighted induced multi-class support vector machines for face recognition. *Informatica: An International Journal of Computing and Informatics*, 44, pp. 456-467. https://doi.org/10.31449/inf.v44i4.3142

[19] Mansour M, Alsulamy S, Dawood S (2021). Prediction of implementing ISO 14031 guidelines

using a multilayer perceptron neural network approach. *PLoS ONE*, 16, pp. 1-18. https://doi.org/10.1371/journal.pone.0244029

[20] Lashkari A H, Kadir A F A, Taheri L, Ghorbani A A (2018). Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification. Proceedings of the 52nd IEEE International Carnahan Conference on Security Technology (ICCST), Montreal, Quebec, Canada, pp. 1-7.

[21] Habibi Lashkari A, Draper Gil G, Mamun MSI, Ghorbani AA (2017). Characterization of Tor Traffic using Time based Features. Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP, Porto, Portugal, pp. 253-262.