Detection of Android malicious applications based on APIs

Xu Jiang1, Baolei Mao2*, Jun Guan1,

1 Northwestern Polytechnical University, Xi’an, Shanxi, 710000, China
2 Zhengzhou University, Zhengzhou, Henan, 450000, China
*Corresponding author’s e-mail: maobaolei@zzu.edu.cn

Abstract. Machine learning technology is widely used to detect Android malware, among which the selection of features is particularly important. Compared with the permission applied, the APIs invoked by applications can better reflect the behavior of the application. Therefore, many researchers choose APIs as the features of machine learning classification algorithm to detect Android malware. Generally, researchers simply select APIs as features, rather than focus on how to select features more beneficial for detection. This paper proposes an Android malware detection method based on APIs frequent pattern selection and use Relief and particle swarm optimization algorithm to jointly select feature subset for detection.

1. Introduction
At present, data mining technology applied in the detection of Android malicious application is very popular. It is necessary to obtain as much features as possible, which can be extracted through static analysis and dynamic analysis [1-3]. The number of features extracted by 1M size application package is as high as 20,000. Provided that high-dimensional fine-grained features are selected, the efficiency of machine learning classifier will definitely be reduced, and not all features are important. Features may contain a lot of irrelevant information, significantly reducing the performance of machine learning classifier. Generally, large data sets can obtain better results, but results are not necessarily better than small data sets. In addition, high dimensional features also lead to the failure of many algorithms and even first-class machine learning algorithms, which are attributed to the "dimension disaster". Therefore, although fine-grained features can obtain more classification information, feature dimensions must be processed.

Android provides a large number of APIs for developers, such as network function, accessing hardware function, etc. The benign or malicious applications implement their function based on the execution of corresponding APIs [4][5]. Therefore, APIs can reflect the basic behavior of the application. However, with a large number of APIs, the problem of data overload must be faced [6]. How to select the data subset to represent the whole data set is the focus of researchers [7].

Based on these insights, we propose and implement an Android malware detection framework that uses static method to extract all the APIs of benign and malicious applications, then use the FP-growth algorithm mining interdependence between different types of benign applications and the malware families separately, combine the two categories of APIs frequent items, get the final API maximum frequent item. Then it uses the Relief algorithm to select features in order to remove irrelevant and noise features, and finally uses the discrete particle swarm optimization (DPSO) algorithm to find the optimal feature subset.
2. Methodology

To detect Android malware, our method requires a set of APIs that enables to determine typical indications of malicious activity. The process of selecting APIs is illustrated in figure 1.

![Figure 1. The process of selecting APIs.](image)

This paper selects applications from Android malware families as malicious samples, extracts APIs invoked by these applications, and constructs the maximum frequent item set of APIs by using FP-growth algorithm. Meanwhile, different types of applications are downloaded from the application market as samples of benign applications. APIs invoked by these application samples are extracted, and the maximum frequent item set of APIs is obtained by using the FP-growth algorithm. Finally, the frequent item sets of APIs of benign and malicious applications are combined (removing duplicate APIs), the elements in the set generate corresponding vectors and form the vector set. Subsequently, the Relief algorithm is used to reduce its dimension, generating the feature vector set of APIs. Then, DPSO is used to select the feature vector set of selected APIs.

3. Evaluation

The experiment was performed on a machine with 16G memory and Inter(R) Core (TM)i7-4720HQ CPU 2.60GHZ. The APIs features are extracted by the program written in Python language. The implementation of the algorithm is mainly completed by the WEKA tool.

In the experiment, the ten-fold cross validation method is adopted. Classification accuracy and F1 are used as evaluation indexes of classifier performance, among which F1 is a comprehensive index of Precision and Recall. Parameters of the discrete particle swarm optimization algorithm are set as shown in table 1.

| Table 1. Parameters of DPSO. |
|-----------------------------|
| DPSO | Parameter |
| --- | --- |
| 20  | Particle number |
| 100 | Iterations     |
| 2   | c1 & c2        |
| [0.5,1] | Inertia weight |

3.1. Data sets

The dataset contains 1,018 malicious applications and 987 benign applications. Malware from 25 malicious application families are selected from AndroMalShare. Benign applications are downloaded from the xiaomi app store through crawler programs, and the applications cover 16 categories, including games, practical tools, audio-visual, chat and social networking, and book reading.

3.2. The selection of APIs

We extract all APIs invoked by malware families to build feature vector sets, such as DroidDream, tapsnake, etc. When an application invokes an API, the corresponding feature vector value is 1,
otherwise it is 0. Then, all the frequent APIs of each malicious application family are set and repeated APIs are removed to obtain the final set of APIs of malware, a total of 4986. Similarly, we adopted the same method to obtain the frequent APIs of different categories of benign application, and combined them, totaling 5782 APIs. The APIs of malicious and benign applications are merged, totaling 8691.

### 3.3. The feature subset selection

Firstly, Relief, information gain (IG), DPSO and regularization algorithm are used to select 1200, 1000, 800, 600, 400, 200, 100 dimensional features from the 8691 APIs extracted by FP-growth algorithm, respectively, and input them into SVM classification. The accuracy curve is shown in Figure 2.

It can be seen from Figure 2, no matter how the number of features changes, the accuracy of the feature subset selected by DPSO is the best when input to the SVM classifier. The accuracy of the feature subset selected by Relief algorithm is better than that of IG and regularization algorithm. When the number of features is larger than 400, the accuracy of the feature subset extracted by DPSO is close to the effect of inputting all features into the SVM classifier. When the features are selected from 200 or 100 dimensions, the accuracy is relatively low, but it is still the best compared with other algorithms. It is worth noting that the efficiency is very low, time-consuming for using DPSO to extract feature subset. When the number of features more than 400, the feature subset selected by Relief algorithm as input to SVM classifier, the accuracy is close to that of DPSO. Considering that when the Relief algorithm selects 400-dimensional features, the accuracy of the classifier is close to DPSO, and the time consumption is far less than that of DPSO. Therefore, this paper uses Relief algorithm to extract 400-dimensional features from the feature set as the feature subset firstly.

![Figure 2](image-url)

**Figure 2.** The accuracy of different algorithms selecting the same number of feature

### 3.4. Detection Performance

DPSO is used to conduct secondary screening of 400-dimensional features and selects 200 and 100 dimensional features respectively. As shown in Figure 3, Relief algorithm select 400-dimensional features and the accuracy is 0.9636. Then, DPSO is used for secondary screening. When 200-dimensional features are selected, the accuracy is 0.9806, but the corresponding accuracy of Relief algorithm is 0.9434. When 100-dimensional features are selected, the accuracy is 0.9769, while the corresponding accuracy of Relief algorithm is 0.9171.

It is worth noting that even if all the 1200 dimensional features selected by DPSO, the accuracy is only 0.9741. The experimental results shows that the method of selecting feature subsets based on Relief algorithm and DPSO proposed is efficient and ensures high accuracy.
3.5. Compared with the randomly selected APIs

In order to verify the advantage of FP-growth association rules algorithm for feature selection, the experiment randomly chooses 8691 APIs from Android SDK Function, Java, javax for comparison. For avoiding the accident of the experimental results, this experiment randomly selects the two set of APIs of the same dimensions as the features and uses the same method to extract 200-dimensional features, which are then input to SVM classifier for classification. The accuracy of the different sets is shown in Table 2. The result shows the importance of using FP-growth to extract APIs frequent item sets of malicious and benign application API.

Table 2. The comparison among our method, Random set 1 and Random set 2

| Feature set      | Accuracy       | F1   |
|------------------|----------------|------|
| Our method       | 98.056%        | 0.981|
| Random set 1     | 95.701%        | 0.957|
| Random set 2     | 96.461%        | 0.965|

4. Conclusion

Although the method proposed in this paper has achieved good results, it still has some limitations. In terms of feature types, this paper only selects APIs, and does not consider other features, such as system call, component information, string characteristics, etc. In the future, with respect to the detection efficiency, more valuable features will be added, and different classifiers will be adopted for different types of features, to obtain higher classification accuracy.

References

[1] Wu D J, Mao C H, Lee H M, et al. (2012) DroidMat: Android Malware Detection through Manifest and API Calls Tracing. In: Information Security. Tokyo. pp. 62-69.
[2] Moonsamy V, Rong J, Liu S. (2014) Mining permission patterns for contrasting clean and malicious android applications. Future Generation Computer Systems, 36(3):122-132.
[3] Arp D, Spreitzenbarth M, Hubner M, et al. (2014) Drebin: Effective and explainable detection of android malware in your pocket. In: Ndss. San Diego. pp. 23-26.
[4] Peiravian N, Zhu X. (2014) Machine Learning for Android Malware Detection Using Permission and API Calls. In: International Conference on TOOLS with Artificial Intelligence. Limassol.
pp. 300-305.

[5] Schmidt A D, Bye R, Schmidt H G, et al. (2009) Static Analysis of Executables for Collaborative Malware Detection on Android. In: International Conference on Communications. Dresden. pp. 1-5.

[6] Kantardzic M. (2011) Data mining: concepts, models, methods, and algorithms. John Wiley & Sons. New York.