Android Apps Security Evaluation System in the Cloud

Hao Wang[1], Tao Li[1,2(✉)], Tong Zhang[1], and Jie Wang[1]

[1] School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, Hubei, China
{1593487967, zt1996816, 909901326}@qq.com, litaowust@163.com

[2] Hubei Province Key Laboratory of Intelligent Information Processing and Real-Time Industrial System, Wuhan 430065, Hubei, China

Abstract. It is an uncertain problem that evaluating the security of Android Apps. We can't be sure of the danger with sensitive permissions in an individual of Apps. Permissions are an important factor in security decisions of Apps. For the Apps security evaluation, the paper proceed from the Android permission mechanism, proposes a classified dynamic security evaluation method. Apps security evaluation system include the large-scale permissions capturing and classification risk evaluation algorithm. The system could find the minimum permissions which are the common features of Apps. The minimum permissions can be dynamically changed according to different classified Apps. We adopt Euclidean distance-based similarity calculation algorithm to evaluate risk. The difference value determines the APP's malicious risk. Experiments prove that the system has reference value to the APP security assessment.

Keywords: Android · App security · Evaluation system · Similarity calculation

1 Introduction

At the first part of 2014, the new malware on the Internet is more than 367 k. 99 % of them are on the Android system [1]. The permissions on android system is the key factor of the Apps security. Because of the different permissions and features, Apps security is an uncertain problem. It can increase the difficult of Apps security evaluation. So we can know that the permissions of one App aren't able to solve the security problem. For example, both of WeChat and flashlight read the contacts. WeChat is a kind of sociality Apps, so we think it is normal. Flashlight is a kind of tools, we think it is dangerous to have the permission. According to the example, we can't judge whether App is dangerous by one kind of permission.

Currently, there are two ways to evaluate the android malware. One is the static analysis method, the other one is dynamic detection. Dynamic detection [2] is monitoring the system calls. It can identify the malicious software by clustering analysis. Static analysis [3] uses the static analysis tools to find out the method called and get the goal by classification algorithm. The two ways have good accuracy. There is in low

S. Guo et al. (Eds.): CollaborateCom 2015, LNICST 163, pp. 151–160, 2016.
DOI: 10.1007/978-3-319-28910-6_14

efficiency when the number of Apps are large. We must use the features of the Android system to deal with. Yue Zhang [4] chose two parts which are the permissions feature selection and permissions feature weighting. It is hard to assess the security of App by the individual.

The paper presents a set of Security Classification Evaluation System. The dynamic system is based on Android App permissions correlation. We put similar Apps analog to a population, and find the set of the minimum permissions in the population with the idea of big data, the other hands the system determine the difference between each individual and the minimum permissions by Euclidean distance-based similarity calculation algorithm.

2 Overview

2.1 Evaluation System

Mobile Android platform provide privacy protection mechanism to prevent the user privacy to leakage. The permission is an important factor in App security. It is difficult to discuss malicious tendencies for App individuals. We put similar Apps likened to a population from the perspective of Big Data. We assume that population has individual which is in line with the minimum security permissions rules. In our opinion, Apps security risk classification evaluation standard should be adjusted dynamically with the Apps updated. Then we identify differences in each App and analyze malicious tendency of App by Euclidean distance algorithm.

2.1.1 Structure

Security Classification Evaluation System includes capture tools and App security classification and evaluation algorithms. Figure 1 is a system configuration diagram.

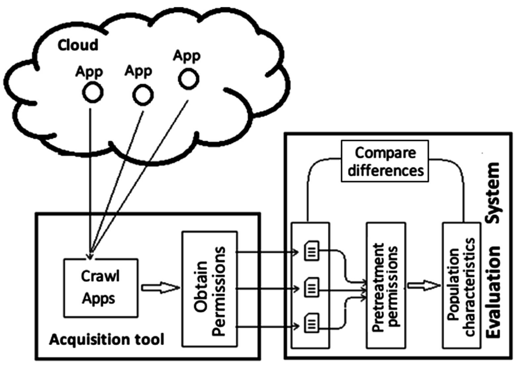

Fig. 1. System structure

2.1.2 Capture Tool

Capture tools should deal with the following key issues.

1. Crawl App automatically to improve response time.
2. Module should be efficient, stable and no-repeated. And it can install quickly and large-scale.
3. Capture for permissions is efficient.

To deal with them, we produce a crawler tool that focus on key words with Scrapy and crawl automatically. Then Apps are installed into testing machine by ADB-tools. Android offers PackageManger class for developers. We get and store all information.

2.1.3 Pretreatment Permissions

All information capture tools to obtain are discrete. And matrix structure we chose makes data storage for analysis. Android official website lists 146 kinds of the permissions for API 19 (Android 4.3 System) [5]. Permissions categories are divided into Normal, Dangerous, Signature and Signature System. We set risk values for all involved permissions. Risk value range: *0 <= Risk value <=100*. And the more dangerous permission is, the higher value is. For example, risk values in the dangerous are greater than in the normal. When i-th App has j-th permissions, i-th App set risk value for permission j.

2.1.4 Safety Evaluation

We find the minimum permissions set of App with the minimum set of permissions search algorithm. The minimum set is a common feature of the App within the population. It is the smallest of privileges for Apps. We identify differences in each App and analyze malicious tendency of App by Similarity Computing algorithm. The minimum set of permissions is determined by population evolution of Apps. So evaluation standard are automatically adjusted. The system can handle out changing App Store.

2.2 Case

We choose flashlight class that features is more specific in tools class. 725 kinds of flashlight App were crawled by capture tool. Then we searched the minimum set of permissions by algorithm. According to the result of experiments, we found out 725 flashlight Apps involved 95 permissions, which contain internet, contact and call. Then total of 578 Apps called network permissions and total of 14 Apps called contacts permission. Similarity calculation algorithm calculated differences from each flashlight and minimum permission set. However, we compared results with downloads in top 10, and found they totally do not coincide. So stores need our evaluation system.

3 The Model of Apps Security Classification and Evaluation System

App has different types and different functions, calls different permission. App individuals themselves permission to discuss the risk judgment malicious tendencies is very difficult, which is also an important factor in App security. Danger of App and its function showed great correlation. For example, video software and malware contains network access, but video software is less dangerous than malware. It is difficult to evaluate for the individual. We assume that population has individual which is in line with the minimum security permissions rules. Then we analyze malicious tendencies by searching the minimum set of permissions and comparing difference.

3.1 The Minimum Set of Permissions

The minimum set of permissions is determined by population evolution of App. We define the concept of permissions and populations before defining the concept of the minimum set of permissions.

3.1.1 Definition

Permissions of App: $Permission_{app} = \{pi \mid pi \in permissions\ of\ Android\}$. It indicates that it Indicates that the set of permissions is subset all of privileges of Android. Permissions storage structure: $PermissionMartix = \{p_{ij} \mid i = 1,2,\ldots,m;\ j = 1,2,\ldots,n;\}$; In PermissionMatrix, if App_i has a permission j, $p_{ij} = 1$; if not, $p_{ij} = 0$;

The minimum set of permissions: $MinPermission = Permission_1 \cap Permission_2 \cap \ldots \cap Permission_i$. It means the minimum set of permissions is the intersection all of set of App in population. Non-essential set of permissions: $nPermission = Permission_{app} - MinPermission$. It represents individual differences.

Population: $P = (Class, Permissions, MinPermission)$. Class is the name of population, Permissions stores all information of each App permissions, and MinPermission is the minimum set of permissions in this class. MinPermission is also called population characteristics. Population operation: $\exists P' = (A_0, A_1, A_2)$, $P = (B_0, B_1, B_2)$, if $A_0 = B_0$, $P' + P = (A_0, A_1 + A_2, A_2 \cap B_2)$. The operation shows how population is operated.

3.1.2 The Minimum Set of Permissions Search Algorithm

Input: Capture tool provides a class of permission information with each App;
Output: The minimum set of privileges such App;

```
1. Define the minimum set of permissions:
   MinPermission = Android's Permissions;
2. Traverse all App:
   foreach Appi in Permissions.
3. Calculate the intersection of MinPermission and App_i:
   MinPermission = MinPermission ∩ Appi;
4. Return MinPermission;
```

According to the definition of the minimum set of permissions, the minimum set of permissions is the intersection all of set of App in population. The minimum set of permissions search algorithm traverse all Apps permissions in the current population information. And calculates the intersection of App permission, which is the minimum set of permissions. The time complexity is *O (n)* because of the intersection of operation.

3.1.3 Dynamic Evolution of the Minimum Set of Permissions

App security risk classification evaluation standard should be adjusted dynamically with the App updated. We presented updated minimum set of permissions algorithm based on population calculation.

Population characteristics update algorithm:

Input: Capture tool to capture new population P', the current system all populations P;

```
1. Traverse current population P:
   foreach Pi in P:
2. Analyze the new population P' whether existed within current system:
   if Pi.Class == P'.Class
3. Update population:
   Pi.Permissions = Pi.Permissions + P'.Permissions;
   Pi.MinPermission = Pi.MinPermission + P'.MinPermission;
4. Return Pi;
5. The system adds new population P ':
   P.Add(P');
```

Since the App of the App Store constantly updated iteration, the evaluation criteria for App is updated dynamically. Algorithms developed App evaluation standard dynamic update rules. It is mainly to detect whether a certain group of newly captured App is an existing population. If the population doesn't exist in system, we should develop a standard for it. Of course, if existing, we should update the minimum set of permissions. The main operation of the algorithm is to determine whether the newly captured populations already exist in the system. So the time complexity is $O_{(1)}$ in best case, and the time complexity is $O_{(n)}$ in the worst case.

3.2 Evaluation of Similarity Calculation

3.2.1 Individual Differences in Similar App

Population mutation is uncertain. We identified the differences App and population characteristic by Euclidean distance-based similarity calculation algorithm [6, 7]. The larger the difference is, the greater malicious tendency is, and the more dangerous App is. There is are eigenvector $(I_0, I_1, \ldots, I_n)$ and discrete distribution point $(A_0, A_1, \ldots, A_n)$ in traditional Euclidean distance similarity calculation algorithm. So the similarity distance d is calculated as:

$$\mathrm{d} = \sqrt{(A_0 - I_0)^2 + (A_1 - I_1)^2 + \ldots + (A_n - I_n)^2} \quad (1)$$

We propose the improved algorithm based on Euclidean distance to calculate the similarity. We have developed the following rules.

∃ *population P = (Class,Permission,MinPermission), and Permission = {P_i | P_i is a set of permissions in each App in the same class}*; *$nPermission_i$* is non-essential set of permissions for each App. And *$nPermission_i$ = {p | p∈$Permission_i$ − MinPermission, 0 < i < n}*. So every value of similar distance can be calculated.

$$\mathrm{di} = \sqrt{nPermission_i} = \sqrt{p_0^2 + p_1^2 + \ldots p_k^2},\ (\mathrm{p}_0,\ \mathrm{p}_1,\ \cdots,\mathrm{p}_\mathrm{k} \in \mathrm{nPermissioni}); \qquad (2)$$

d_i determines the difference App and population characteristic. The higher d_i is, the lower similarity is and the higher malicious tendency is. And App is more dangerous. The minimum set of permissions is the basis of similarity calculation. This paper selects the similarity analysis of the Euclidean distance algorithm is due to the algorithm can effectively distinguish the similarity of App. The main function of the algorithm is the difference between the performance of each App behavior and the minimum set of permissions. And a lot of similarity algorithms are based on distance such as K-means and K- center algorithm.

4 Experimental Evaluation

Based on the system-designing, we did three experiments to evaluate the system. We crawled 725 flashlight Apps and stored on local computer from 360 market with Scrapy framework. On Android 4.1.2, we installed all Apps by ADB-tools and got their permission. According to the result of experiments, we found out 725 flashlight Apps involved 95 privileges.

Experiment 1: With the help of statistical methods, we calculate that how many Apps called permission.

1. Traversing permissions information matrix.
2. Calculating the number of permission to be called.

Figure 2 displays case sensitive permission to call.

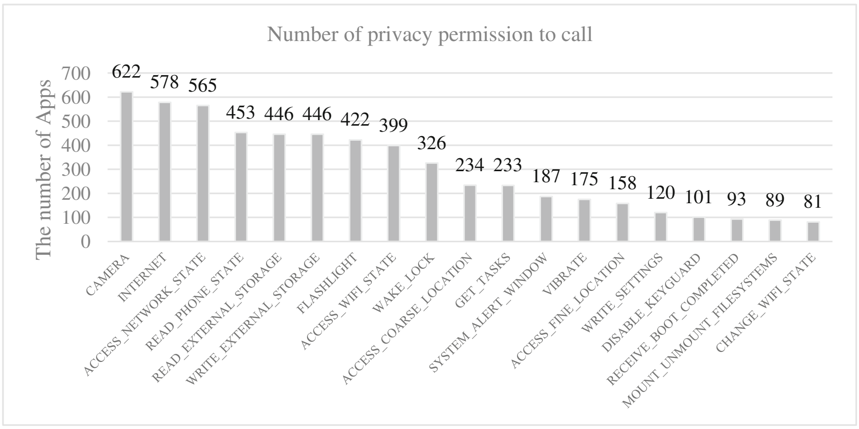

Fig. 2. Number of privacy permission to call

The majority of the 725 flashlight Apps contain some sensitive permission such as CAMERA, INTERNET, ACCESS_NETWORK_STATE, ACCESS_PHONE_STATE. Those permissions couldn't match with the features of flashlight. There is conclusion that the minimum set of permissions is empty. This means flashlight could run with no privilege. So these permissions increased malicious tendency of Apps.

Experiment 2: Calculate similar distance and analyze relations similar distance values with permissions.

1. Traverse permissions martix after pretreatment;
2. Calculate the value of similar distance of the App with minimum permissions in the improved Euclidean distance algorithm.

Figure 3 shows Similar Distance values in descending order.

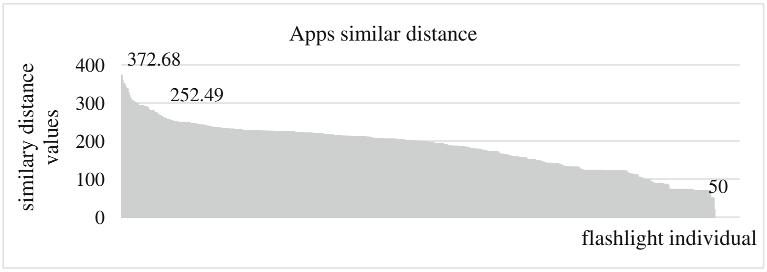

Fig. 3. Apps similar distance

Excerpt of data shows (the value of similar distance of com.sskj.flashlight is 372.68, the value of similar distance of com.mika.flashlight is 252.49 and the value of similar distance of com.htc.flashlight is 50). The larger the value of similar distance, the greater the tendency malicious Apps. We could get more information on Fig. 3. The minimum of the value of similar distance is zero, the Maximum is 372.68 (the result is concerned about permission dangerous scores). It is a clear difference between each App. Euclidean distance algorithm could effectively differentiate the malicious tendencies. We can conclude App risk evaluation system classification is workable and practical.

Figure 4 shows the relations similar distance values with permissions of the App.

It is a non-linear relationship between the number of App permission to call and similar distance values. The same number of permission is not necessarily the same distance. The higher the risk of sensitive permissions, the greater the impact on similar distance (the result is concerned about permission dangerous scores).

Experiment 3: Calculate permissions for each App owned by mathematical methods.

1. Traverse permissions martix after pretreatment;
2. Calculate permissions for each App owned.

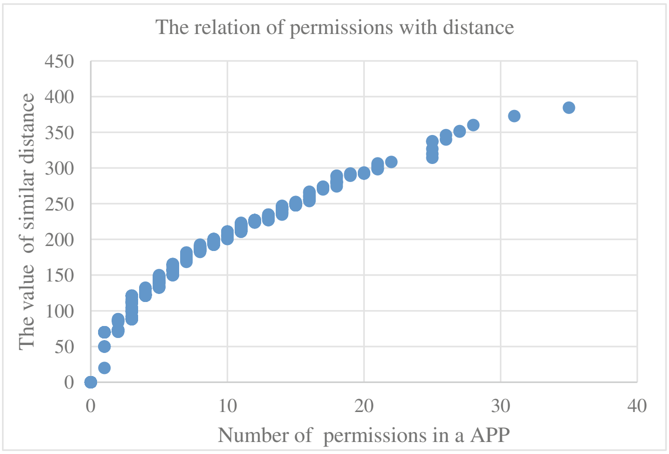

Fig. 4. The relation of permissions with distance

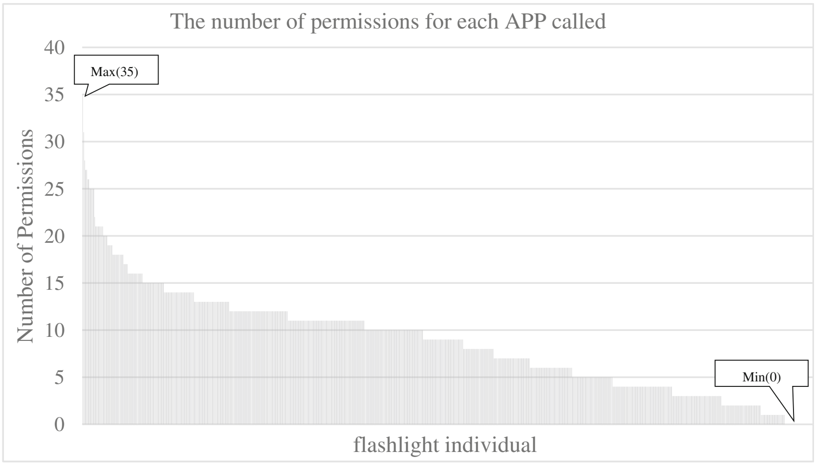

Fig. 5. The number of permissions for each App

Figure 6 could display that the number of Apps within the range of same permissions number.

In Fig. 7, we counted the number of permission that Apps called from 7 to 14.

We could conclude an App can call 35 kinds of permissions and a safe App should have no privilege in flashlight class from Figs. 5, 6 and 7. There are malicious tendency in rare of Apps. These permissions such as READ_EXTERNAL_STORAGER and WRITE_EXTERNAL_STORAGE would be called frequently in the majority of Apps.

From the above chart, we have a comprehensive understanding of the flashlight-type Apps. These Apps are also dependent on a number of other permissions in addition to the basic. Most of the flashlight-type Apps call these permissions purpose

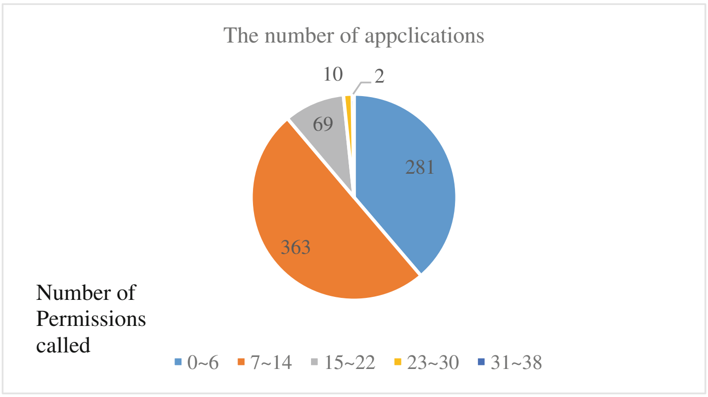

Fig. 6. The number of Apps (Color figure online)

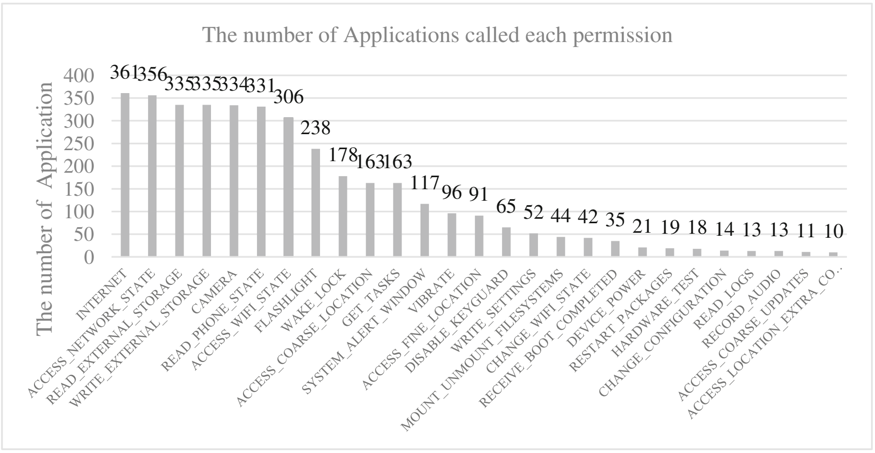

Fig. 7. The number of Apps called permissions

to advertise and market. However, the existence of a very small number of malicious App to read user contacts (Similar distance largest App). This sample may indicate that the Euclidean distance algorithm is effective and App security risk classification evaluation can be applied for evaluating App security.

5 Conclusion

Experimental results show that the Euclidean distance algorithm operability and practicability of App hazard classification. But there are still insufficient. For example, capture tools crawl App certain failure rate, whether the Euclidean distance algorithm is the most Appropriate. So there are to be further optimization in the evaluation model.

Acknowledgement. Authors are partially supported by Colleges and Universities in Hubei Provincial College Students' Innovative Entrepreneurial Training Program (No.201510488006), National Natural Science Foundation of China (No.61273225), Humanities and Social Sciences Foundation of Education Ministry of Hubei Province (No.2012D111).

References

1. Internet environmental remediation report at 2014 first half[R/OL] 10 September 2014
2. Burguera, I., Zurutuza, U., Nadjm–Tehrani, S.: Crowdroid: behavior-based malware detection system for Android. In: Proceedings of the 1st ACM Workshop on Sercurity and Privacy in Smartphones and Mobile Devices, pp. 15−26. ACM, New York (2011)
3. Schmidt, A.D., Bye, R., Schmidt, H.G., et al.: Static analysis of executable for collaborative malware detection on Android. In: Proceedings of the 2009 IEEE International Conference on Communications, pp. 631−635. IEEE Press, Piscataway (2009)
4. Zhang, Y., Yang, J.: Android malware detection based on permissions. Comput. Appl. **34**(5), 1322–1325 (2014)
5. Google.Mainfest.permission[EB/OL], 01 November 2013. http://developer.android.com/reference/android/Manifest.permission.html
6. Song, Y., Zhang, Y.,, Meng, H.: Research Euclidean distance clustering method based on weighted. Computer Engineering and Applications (2007)
7. Liu, R.: Weighted Euclidean distance and its application. Mathematical Statistics and Management (2002)