Permission Request Characterization on Android

Chit La Pyae Myo Hein
University of Computer Studies, Monywa
chitlapyae86@gmail.com

Khin Mar Myo
Central Institute of Civil Service (Lower Myanmar), Yangon.
kmmyo.ag@gmail.com

Abstract

Mobile malware performs malicious activities like stealing private information, sending messages, SMS, reading contacts can harm by exploiting the data. Malware spreads around the world infects not only end user applications, but also large organization service provider's systems. Malware characterize is a vital component that works together with malware identification to prepare the correct effective malware antidote. Malware feature category is also important to reduce costs and time for malware identification. They may have many features in every mobile android application. This system proposed a score-based detection for Android malware. The advantage of this system is that it uses only manifest files to detect malware. Therefore, in this work is to explain the criteria for characterize, this system need to process characterize different features from the manifest file of permission request. According to the experiment on different categories; the results show that the proposed features characterize is applicable as a lightweight approach.

1. Introduction

One of the most common uses is access to the Internet. Users can download malicious software by repackaging applications using reverse engineering tools. The attacker changes the code in order to incorporate the malicious code, then repackages the application and publishes them in

the application market. Users usually cannot differentiate between the malware application and the legitimate application, thereby end up installing malware.

Mobile applications are rapidly growing segment of the global mobile market. Android is an open-source mobile phone operating system with Linux-based platform which consists of the operating system, middleware, and user interface and application software as shown in Figure 1.

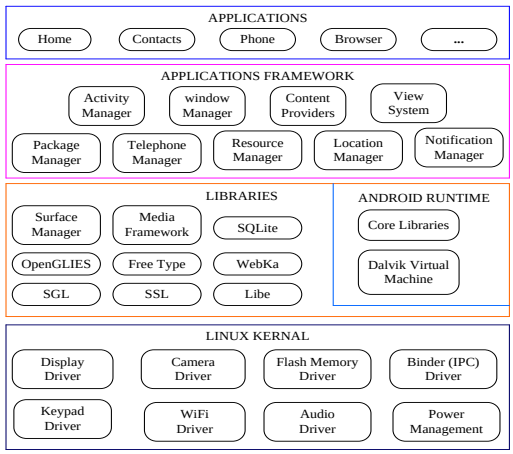

Figure 1. Architecture of Android

Android is about to become the most widely used OS on mobile phones, but with Android comes a security vulnerability that few users take into account. On the Android Market, users can download and upload thousands of applications without having special security checking up knowledge. Security plays a vital role in today's mobile world.

The rest of the paper is organized as follows. Section 2 presents the related works.Section 3 explains the malware detection system, detailing

the process of building the application to collect and give information about the malware detection system. Section 4 presents the results of the malware detection testing and evaluation methods. Section 5 concludes and gives possible future work to reduce the limitations of the system proposed.

2. Literature Review

Many researchers propose complicated extensions to fortify the Android's security framework. They mainly focus on protecting the user data and mitigating some types of privilege escalation attacks. This section recent some of the most well-known approaches to extract the malware list in android technologies.

Alternative research has focused on using machine learning techniques to identify malware. Sanz et al. (2012) applied several types of classifiers to the permissions, ratings, and static strings of 820 applications to see if they could predict application categories. They applied this by using the category scenario as a stand-in for malware detection. [4] Shabtai et al.(2010) similarly built a classifier for Android games and tools, as a proxy for malware detection.[1] Ryo Sato, Daiki Chiba and Shigeki Goto propose a new method for detecting Android malware by analyzing only manifest files based on malware score. [13] Zhou et al. (2012) found real malware in the wild with DroidRanger, a malware detection system that uses permissions as one input. [14] DroidMat (2013) focuses on using attributes of the manifest to trace API calls requiring permissions. [12] N. Peiravian and X. Zhu (2013) proposes a rule-based security mechanism designed to prevent malware at install-time. [11] Explores the use of machine learning algorithms for malware detection using permissions and API calls. The studies in Mila (2009) and android (2015) Android Content License "URL" www.source.android.com /license.html retrieved focus on efficient, scalable, and accurate malware detection in large Android markets.

3. Proposed Feature Selection Method

According to Wu, Mao, Wei, Lee and Wu. (2012) there have been a lot of methods and techniques for feature selection. Most of the techniques are based on machine learning technique. [6] Meanwhile, other papers Peiravian and Zhu (2013) Shabtai, Fedeland Elovicitried (2010) out another light weight approaches to classify mobile malware. Their approach [11] [3] is only based on manifest file analysis rather than applying machine learning algorithms. This paper also proposes a feature selection method based on the manifest file analysis approach. The process flow of our propose method is described in Figure 2.

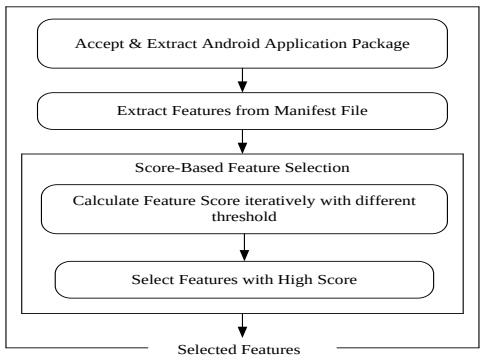

Figure 2. Flow of Proposed Score-Based Feature Selection

The nature of mobile android application (APK) file, how to extract the features from mobile applications are described in this section. The detail explanation of how to process the proposed score-based feature selection is also described in the section below.

3.1. Android Application Package and Manifest File

An Android application package (apk) usually includes the components as described in Figure 3, below, in which a manifest file is also included.

Figure 3. An Android Application Package

Every application must have an android Manifest.xml in its root directory. The manifest presents essential information about the application to the Android system. The information system must have this manifest into before it can run any of the application's code.

Applications must declare in their manifest file which permissions they request or require. When an application is installed, the Android system will present the various malicious applications uploaded in Google market, which misuse the deficiencies in the malware detection framework making the user decide to allow the installation or not.

Android permissions control the access to sensitive resources and functionalities. Android defined permissions are available to third party applications [17]. The permission mechanism should be used to secure the various components in an application. This effect is achieved primarily by associating permissions with the relevant component in its declaration in the manifest. Additionally, applications having Android automatically enforces for the existence of the permissions in the relevant scenarios.

3.2. Features Extracted from Manifest File

There is the same manifest file in both benign and malware applications. The information extracted from manifest file can be categorized into six categories. They are: (1) Permission, (2) Intent filter (action), (3) Intent filter (category), (4) Process name, (5) Intent filter (priority) and (6) Number of redefining permission.

Many applications require several permissions to function properly. These permissions must be listed explicitly in the application's Android Manifest.xml file and accepted by the user during installation. Therefore the only permission based information is extracted from manifest file as our feature set. Table 1 shows the examples of permission keywords in a manifest file.

Table 1. Permission Keywords in a Manifest File

```
<user-permission android name="”

android.permission.ACCESS_WIFI_STATE/>
<user-permission android name="”

android.permission.READ_PHONE_STATE/>
<user-permission android name="”
                android.permission.WRITE_SMS/>
<user-permission android name="”
                android.permission.SET_WALLPAPER/>
```

3.3. Score-Based Feature Selection

The detail process flow of the proposed feature selection method is described in this section. The features are extracted from manifest file of android application package (.apk) as explained in section 3.1 and 3.2. The sample extracted features are described (in Table 2 below).

The score for each extracted feature is calculated using the following formula.

$$MS = E - UE / A \qquad \text{eq (1)}$$

Where; **MS**= malware score

E= number of feature existence

UE= number of feature un-existence

A = total number of applications

The decision of malware score feature should be selected or rejected, which are made by adjusting the score value with different thresholds. The classifier 'Weka Tool' is used to calculate and validate the classified results. If the feature set malware score, which can give the highest classification results are selected, then these feature sets within the appropriate threshold are chosen as selected features. The remaining features under the threshold level are rejected.

Table 2. A Table Showing the Permissions Used by each of Android Applications

Android Applica-tions	Read SMS (F1)	Call Phone (F2)	Write SMS (F3)	F4	F5	F6	……..	F-134
Apk1	0	1	0	1	0	0	……..	0
Apk2	1	0	1	0	0	0	……..	1
Apk3	0	1	0	0	1	0	……..	0
Apk4	0	1	0	1	0	0	……..	0
Apk5	1	0	1	0	0	0	……..	1
Apk6	1	0	0	1	0	0	……..	1
Apk7	0	1	0	1	0	0	……..	0

4. Experiment and Evaluation

We tested our system against a collection of many benign and malicious Android applications. For each data point, we selected a random subset of the training (benign) applications and performed using training set. A dataset made of 250 trusted and 250 malware Android applications was collected. The trusted applications, by different categories were downloaded from the common Android Markets.

Others malware applications of different categories and malicious intents were downloaded from public databases of antivirus companies. The malware nature of each application was confirmed by antivirus companies. Then the features are extracted from the 250 trusted and malware Android applications. These features are used to evaluate our proposed feature selection method. From these 134 features are obtained applications. The detailed experiment is described in the following section.

4.1. Feature Score Calculation with Different Threshold

The score for each of the 134 features out of the 203 Android applications are calculated using Equation (1) MS=E-UE/A. The score results of some features out of the 203 applications are described in Table 3 as an example.

Table 3. Feature and Their Feature Score

F_ No	Feature Name	E	EU	E-EU /A	M_ Score	%
1	WriteSMS	99	5	94/203	0.4	40
2	Call Phone	76	99	23/203	0.1	10
3	ReadSMS	89	65	24/203	0.1	10
4	Internet	94	86	8/203	0.03	3
5	Read Content	89	42	47/203	0.2	20
6	Read_ PhState	99	45	45/203	0.2	20
7	Battery State	99	65	34/203	0.1	10
8	Delete Package	86	38	42/203	0.2	20
9	Global Search	99	70	29/203	0.1	10
10	Call Privilleg e	88	72	16/203	0.07	7

The following Figure 4 also shows the results of feature score on Malware and Benign applications.

If we take the threshold value 11, the features under the threshold value (2,3,4,7,9,10) are rejected and remaining features (1,5,6,8) are selected see Figure 4. Similarly, feature selection is tested by using different numbers of threshold values. The threshold value, which gives good classification accuracy is chosen and marked for later use.

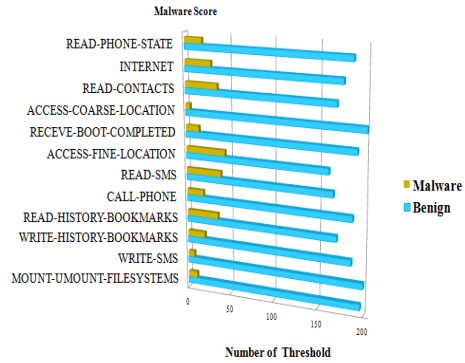

Figure 4. Feature Score on Malware and Benign

How to validate the above selected features can provide good characterize for Malware and Benign application as explained in the following section.

4.2. Categorization of Risky Permission

Permissions have different danger levels depending on the functions they allow the application to perform and are consequently classified in protection level groups. Likewise, through this attribute, it is possible to determine which applications have access to the permission:

Risk1: They pose a risky1 factor and typically only affect the application's scope. Risk1 permissions are granted by the system automatically without explicit approval of the user.

Risk2: They are risky2 permissions that allow costly access to services. The permissions can be granted by the user during installation. If the permission request is denied, then the application is not installed.

Risk3: They are risky3 permissions are only granted if the requesting application is signed by the same developer that defined the permission. Risk3 permissions are useful for restricting component access to a small set of applications trusted and controlled by the developer.

The big problem is that groups can contain both normal, basic permissions as well as more dangerous permissions. For example:

i. **Location**: An app that asks for your approximate, network-based location can now gain permission to track your exact location with your device's GPS.
ii. **SMS**: An app that only needs to receive text messages can now gain the permission to send SMS messages in the background, potentially costing you money.
iii. **Phone**: An app that asks to read your call log can now gain permission to reroute outgoing calls and make phone calls without asking you.
iv. **Photos/Media/Files**: An app that needs to read the contents of your USB storage or SD card can now format your entire external storage device.

The categories of applications are often to characterize an application as malicious or benign. In characterization features are used to make decisions. Application features are required to be informative to produce an accurate decision.

Table 4. Categorization of Permission Risky

Data Category	Data Usage	Permission Request	Private Threat		
			R 1	R 2	R 3
Sensor/ Location	Location Audio Video	ACCESS_COARSE_LOCATION	☹		
		ACCESS_FINE_LOCATION	☹		
		BLUETOOTH_ADMIN	☹		
		ACCESS_NETWORK_STATE	☹		
		ACCESS_WIFI_STATE	☹		
		READ_SOCIAL_STREAM	☹		
		RECORD_AUDIO	☹		
		CAMERA	☹		
External Storage		WRITE_EXTENAL_STORAGE	☹		
		READ_EXTERNAL_STORAGE	☹		
Communication	SMS MMS Voice Wap Push	RECEIVE_SMS		☹	
		READ_SMS		☹	
		RECEIVE_MMS			☹
		PROCESS_OUTGOING_CALLS			☹
		RECEIVE_WAP_PUSH			☹

Another observation is that some applications of permissions are requested by such malware applications in the Cantagio dataset. Malware are more favor of changing the settings and use money-related services such as short message service (SMS). Changing settings, especially changing the network settings, generally is the first step before a malware performs any malicious activity. Sometimes malware even try to kill background processes, which could help them avoid being detected by anti-virus applications. Characterization system can see that the usage pattern of SMS related permissions is

quite different between the benign applications and the malware applications and many malware applications attempt to request SMS related permissions. SMS is also a risky permission of private threat (Risk3) that is more likely requested by malware applications. Describe the data usage of SMS include the data category of communication as shown in Table 4.

5. Conclusion and Future Work

The advantage of this system is that it uses only manifest files to detect malware. Manifest files are required in all Android applications, and thus, the proposed system is applicable to all Android applications. This system proposed a score-based detection for Android malware. The results show that the proposed method can detect malware and benign samples that are undetectable by a simple static approach. Similarly malware characterized accuracy is also evaluated using different permission of feature. Future work will emphasize testing of already tested malware applications to deduce the characterized of their performance. Testing of other major applications has been done in the Android Market and discovers additional mobile device vulnerabilities.

References

[1] A. Shabtai, Y. Fledel, and Y. Elovici, "Automated Static Code Analysis for Classifying Android Applications Using Machine Learning" In Proceedings: Computational Intelligence and Security (CIS), 2010 International Conference

[2] A. P. Felt, K. Greenwood, and D. Wagner, "The effectiveness of application permissions," in Proceedings of the USENIX Conf. on Web Application Development, ser. WebApps, 2011.

[3] A. P. Felt, E. Chin, S. Hanna, D. Song. D. Wagner. Android permissions demystified. In the proc. of the 18th ACM conference on Computer and communications security, CCS 11, ACM, 627-638,(2011)

[4] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. Bringas, "On the automatic categorisation of android applications," in Proceedings of the 9th IEEE Consumer Communications and Networking Conf. (CCNC), 2012.

[5] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P.G. Bringas y G. "MAMA: Manifest Analysis for Malware Detection in Android" published in Cybernetics and Systems-Intelligent Network Security And Survivability 1,October,2013

[6] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee and K.-P. Wu. Droidmat: Android malware detection through manifest and api calls tracing. In 2012 Seventh Asia JCIS, pages 62–69. IEEE, August 2012.

[7] D. Barrera, H. G. u. c. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in ACM CCS '10.

[8] G. Holmes, A. Donkin, and I.H. Witten, ―Weka: a machine learning workbench,‖ August 1994, pp. 357-361.

[9] J. Sahs and L. Khan "A Machine Learning approach to Android malware detection," 2012 European Intelligence and Security Informatics Conference, 2012.

[10] L. Batyuk, M. Herpich, S. A. Camtepe, K. Raddatz, A. Schmidt, S. Albayrak, "Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within android applications," Malicious and Unwanted Software (MALWARE 2011), 6th International conference, 2011.

[11] N. Peiravian and X. Zhu. Machine learning for android malware detection using permission and api calls. In Proc. of IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pages 300–305. IEEE, November 2013.

[12] R. Sato, Daiki Chiba and Shigeki Goto "Detecting Android Malware by Analyzing Manifest Files" Proceedings of the APAN – Network Research Workshop 2013

[13] W. Enck, M. Ongtang, and P. McDaniel. On lightweight mobile phone application certification. In Proc. of the 16th ACM conference on CCS, pages 235–245. ACM, November 2009.

[14] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, You, Get off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets," in NDSS'2012.

[15]Google Android Retrieved http://developer.android.com/guide/basics/what-is-android.html

[16] Google Android source code. Available: http://source.android.com/source/downloading.html

[17] Google Inc. Android Developers. Manifest. permission. http://developer.android.com/reference/android/Manifest.permission.html.
[18] Google Inc. Android Developers. Security and Permissions http://developer.android.com/guide/topics/security/security.html.
[19] Permission>Android Developer - API Guides – Android Manifest.http://developer.android.com/guide/topics/manifest/permission-element.html
[20] Cantagio mobile malware-mini-dump http://contagiominidumo.blogspot.com