I J C T A, 9(11) 2016, pp. 5307-5322
© International Science Press

BEHAVIOUR ANALYSIS OF ANDROID APPLICATION

Sapna Malik[*] and Kiran Khatter[**]

Abstract: Access Control Lists (ACLs) and Permission Security Models are mechanisms to provide controlled access to the critical resources. Managing ACLs is a complicated process and requires a lot of technical knowledge for its configuration and design which can be done by developers only. In android permission security model, the critical resources are protected by permission system. The user, Android Marketer and Developer play an important role in requesting and granting these permissions. The end user who grants permission requested is not able to understand android permission documentation as it has many flaws including it being too technical for the end user. This work presents a methodology for behavior analysis of android application by analyzing permission requested by android application extracted through AndroData, a tool developed for static and dynamic feature extraction.

Key Words: Android Permission Security Issues, Protection Level, Malware Families

1. INTRODUCTION

Modern Mobile Phone or Smartphone has become minicomputer for the user by providing huge functionality like browsing the internet, editing text, using online banking services. Also, its storage capability allows the user to store his personal data like contact list, multimedia data etc. Android applications in an android phone facilitate user with many attractive features and make user's life easy. The android operating system protects its critical resource access through permission security model. When we install an android application, it requests for permission for accessing critical resources assuming that the user has the knowledge of risks involved in granting these permissions. Due to lack of knowledge to the user, the malware becomes successful in doing intrusion in a mobile device, steals personal data and also affects the user financially. This paper gives insight into the behavior of the malicious application in respect to the permission requested by benign and malicious applications which will help both, the End User to take the right decision while granting the permission and the Researcher to propose a methodology for preventing malicious application intrusion. This paper is organized into six sections. Section II discusses the work related to Permission Security Model. Section III explores the Android Permission Model with its security issues and protection level. Section IV proposes the methodology for behavior analysis of the android application. Section V describes results and observations during analysis of benign and malicious applications from different malware families.

[*] School of Engineering and Technology Ansal University, Gurgaon Haryana, India
Email: sapnadhankhar@gmail.com

[**] School of Engineering and Technology Ansal University, Gurgaon Haryana, India
Email: kirankhatter@gmail.com

2. RELATED WORK

Many researchers have emphasized the importance of permission features. Gomez and Neamtiu [1] explored the four malware families DroidDream, DroidDreamLight, Zsone SMS, Geinimi and done their analysis based on the resources accessed, infiltration techniques and the payload used. Tang et al. [2] proposed the Security Distance Model. He used the threat point to represent the danger level of malware. Grace et al. [3] proposed the method for detecting the permission escalation attack which is the attack caused by a collaboration of malicious applications. Sarma et al. [4] done an analysis of permissions based on the category of applications permission request and the correlation with the applications requested where the permission belonged to the same category. PermissionWatcher [5] is an Android application that classifies malicious application installed on the device based on the custom set.

Dini et al. [6] proposed a multi-criteria evaluation based on the threat score computed with the help of permission requested for accessing the critical resource and the critical operations executed. Sanz et al. [7] proposed a classification of android applications into several categories like entertainment, tools, games and multimedia and communication using machine learning techniques on permission features. DroidRanger [8] is multi criteria based system that uses permission requested, other information in the manifest file, structural analysis of application code and heuristics based filtering for classification of android application. Wei et al. [9] presented the analysis of third party application permission model and proposed several approaches for securing critical data of the device.

Holavanalli et al. [10] proposed Flow Permissions to examine the implicit and explicit flow of permission grant mechanism within the applications. VetDroid [11] is a dynamic analysis platform for analysis of permission usage behavior and permission acquired by the android applications. Rosen et al. [12] presented an approach for the analysis of application based on the API calls and fine grained privacy related behavior. Sato et al. [13] used permission, intent filter and process name for the classification of benign and malicious applications.

Rassameeroj and Tanahashi [14] used network virtualization and clustering algorithms in its permission based security models. Canfora et al. [15] proposed Permission and SystemCall based malware classification technique. Zhu et al. [16] proposed system for classification of dangerous applications based on permission and application description. Aung and Zaw [17] proposed a machine learning based malware detection framework by monitoring permission based features and events executed by the android application.

As discussed in the related work, the Permission is a very important feature for detection of a malicious application, so the proposed work is a contribution to this chain of researches which will help the researchers in a better understanding of the behavior of malicious applications with Permission Android Security Model. A naïve developer and end user will also benefit from it as this work discusses the adverse effects of granting normal and dangerous permission.

3. ANDROID PERMISSION MODEL

In android Security System, the application is executed in its own dalvik virtual machine instance with the UID assigned to it during installation. This is how the android security system isolates the execution of different applications. Another way to provide Security in Android operating system is through Permission model in which each application has to request for permission for accessing android critical resources and accessing components of other application. The android permission is unique text string declared and requested in Android Manifest file of the Android application. Figure 1 shows the format of Android Manifest file. There are 130 android permissions from different groups for accessing critical resources like internet, camera, contact list, dialing a phone number, sending SMS etc.

```
<permission android:description="string resource"
          android:icon="drawable resource"
          android:label="string resource"
          android:name="string"
          android:permissionGroup="string"
          android:protectionLevel=["normal" | "dangerous" |
                              "signature" | "signatureOrSystem"] />
```

Figure 1: Format of Android manifest file

The third party developer can also declare custom permission for accessing their components/services by other applications. These custom permissions must be declared in the manifest file. The other applications which want to access these components/services should request for these custom permissions in their manifest file. The android permission model is based on the decision of all or no permissions. For e.g. at the installation time, application requests for all permissions from the end user that are needed for its successful execution and denying the permissions abort the installation process. Once the permission is granted, it never requests again for accessing a critical resource in its lifetime and the user is trapped by giving permission to a malicious application. The Mandatory Access Control enforcement of reference monitor controls the access to the android resources or the components of the application by evaluating whether the corresponding permission is granted to the application.

3.1 Android Protection Level

Android permission model defines four protection levels for different kind of permissions.

Normal – These permissions are low-risk permissions and granted automatically to the requesting application. These permissions in isolation do not impart any real harm to the End-user. e.g. SET_WALLPAPER

Dangerous – These permissions are high-risk permissions and these are requested by the application explicitly from the End-Users at installation time. These permissions allow access to harmful API calls such as send SMS, access user’s private data and take control of device etc. e.g. SEND_SMS, CALL_PHONE, BRICK.

Signature – These permissions are also automatically granted to the applications that are from same software suite that declare the permission. The application that declares permission and the application that requests permission bear same certificate or signature.

Signature/System – These permissions are same as Signature but these are for Device manufacturers or Operating Systems use. These permissions are automatically granted to the device manufacturers or android operating systems applications.

3.2 Android permission security issues-

Android permission Security issues are divided into two categories Direct Issues and Indirect Issues [19]. Indirect Security Issues are the loopholes in Android Permission Model because of which the direct security issues exist. Direct issues are permission escalation attack, time of check to time of use attack (TOCTOU) and over-claim of permissions which lead to financial loss and leakage of user private information by exploiting the android permission model. The Indirect issues are Coarse Granularity of permission, insufficient Android permission documentation about permission usage for the developer, Incompetent permission administrators who are End users and developers. The indirect issues - Coarse Granularity of Permission, Incompetent permission administrators and Insufficient Android Permission Documentation have explicit relation with direct issue Over Claim of Permission whereas Incompetent Permission Administrator and Insufficient Android Permission Documentation have implicit relation with Permission Escalation Attack and TOCTOU attack issues respectively.

Indirect Security Issues-

Coarse Granularity of permission - The Android permission model has lots of Coarse grained permissions which allow the attacker to gain access to plenty of resources by just getting authorization to one permission. E.g. INTERNET permission is heavily used Coarse grained permission by both malware and benign application which allows an application to send HTTP(S) requests for sending and receiving data from any domain. The Internet permission can be the gateway for the malicious application to do malicious activities like leakage of private data by making HTTP request to specific domains only. The issues of Coarse granularity can be resolved by doing improvement in current framework and enhancing installation time and run time permission grant policy.

Insufficient Android permission documentation - The Android permission model is poorly documented with the lack of permission usage information for the developers and being too technical for End-users to understand the risk involved in granting the permission. In Android permission documentation it is not clearly stated which method of the class requires which permission that leads to defective android apps with over claim of permissions.

Incompetent permission administrators - The Developer, Application Marketer and the End User are the role players in the process of granting permission. The Developer requests for permission in the manifest file of the android application, Application marketer verifies the application and the End user grants the permissions but these three have different interests, Developer wants his application to work either by over-claiming of permissions rendering the end user vulnerable. The end-user wants to install the application by granting all the permissions without knowing the risks involved in granting permissions.

Direct Security Issues

Over-Claim of Permissions - The developers can over claim permissions due to lack of knowledge or for the purpose of doing the malicious activity. A naive developer may request for all the permissions related to the functionality he is designing because of having less experience in android application development leading to violation of the principle of least privilege (PLP) and exposing End users to financial and privacy loss. Permission can be requested by the developer on behalf of its deputy application.

Permission Escalation Attack- Permission Escalation Attack is a collaboration of malicious application with other application for gaining access to the critical resource without requesting its permission explicitly. The permission escalation attack can be a confused deputy attack which occurs due to loopholes in the interface of benign application or Collusion attack which is a collaboration of malicious applications for having a joint set of permissions for doing malicious activities.

TOCTOU Attack- In android permission model there is no constraint or rule imposed on the naming of new permission and because of this flaw TOCTOU attack occurs. For exemplification, suppose a malicious application M2 declares a permission P’ having the same name as the permission P declared by the benign application. The Permission P is used for accessing critical resources. Now, another malicious application M2 requests for permission P' declared by M1. However, due to naming collusion the Application M2 can access critical resources for doing its malicious activities as P' and P have similar names.

4. BEHAVIOR ANALYSIS OF MALICIOUS APPLICATIONS

4.1 Dataset

As in this work we have done the behavior analysis of both benign and malicious applications, the datasets of two projects Drebin[20] and Androtracker[21] are taken for this purpose. The Drebin Project provided the dataset of apk files of Malicious Android Applications Whereas Androtacker Research project provided the dataset of Benign Applications. The Apk files used in the proposed work's size varies from 3kb-20MB. The Android application taken for the experimental result is from all Android Application categories like games, tools, education, utility, entertainment etc. In this work 527 malicious android applications and 533 benign applications are used for doing the analysis.

4.2 Features extraction

For doing a successful analysis of Android Applications we have extracted permission feature of the benign and malicious android applications. For extracting the permission features of android application we have used self-developed tool Androdata[22]. The AndroData tool is light weight automatic tool written in shell scripting language which can be used on both Smartphones and Android Emulator. The Androdata tool is used to extract two features of Android Application- Permissions Requested and System Calls Called (in terms of frequency). In AndroData tool, the permission a static feature is extracted from the Android manifest.xml of Android Application with aapt command. The System call a dynamic feature is extracted with strace command after simulating the app with monkey tool. After running AndroData tool we have a dataset of 1060 android applications with their 220 types of android and custom permissions requested by the applications. The dataset has malicious applications from 82 malware families.

5. RESULT AND OBSERVATION

5.1 Malware Families

As written earlier, we have extracted the features of malicious applications from 82 malware families. The graph shows the no. of malicious applications from top 20 families. The following section throws lights on a few of the android malware families.

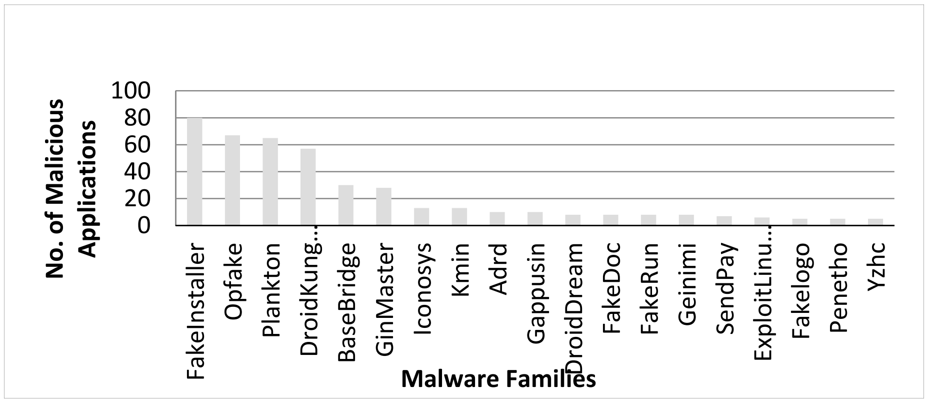

Figure 2: Top 20 Malware families and their Applications Samples in dataset

Fake Installer Malware Family

Fake Installer is a most common malware family that appears to be an installer of a popular android app and sends premium-rate SMS messages to services owned by malware author during installation. This malware hides in a repackaged code of a popular android app. This attack is prevalent in Eastern Europe, Russia and Asia.

OpFake Malware Family

OpFake is the second most common malware family named for being the fake downloader of Opera Mini Mobile Browser. The attacker first created a fake website to lure the customers to download Opera Mini Browser and named its apk file similar to opera. After downloading and installation, it sends premium SMS messages stealthily without victim’s knowledge. It also steals the device information like Country location, Operator name, OS version, Phone Type and IMEI no. of the device and sends the information to the malware author. During the installation, it requests for permissions related to SMS, Contacts and SD card. After their successful installation, it redirects the victim to the authorized page to download Opera Browser to install Browser. The Victim sees the Opera Browser Icon where the malware is running in the background. This family also bundles their malicious codes in other popular android app codes to spread more.

Plankton – Malware Family

The malware of this family runs as a background service named *AndroidMDKProvider* of the third party host application. This background service was written and started in OnCreate ( ) method of main activity of host application. This background service has the capability of bypassing the third party verification process and communicate with the remote server through HTTP and sends information like Application ID, Brand, Build number, Developer ID, Device Display Metrics, IMEI, Locale, Protocol Version, SDK version, Source IP, User Agent, User ID, Version Release and List of Permissions granted to host the application. This malware also has the capability of installing other malicious codes from the predefined URL and sends Premium SMS text messages.

DroidKungFu Malware Family

DroidKungFu is one the most difficult to detect and control malware family. It gains root access privileges to steal sensitive data like device ID, OS version device and send this information to remote servers. It also has the capability to turn the device in a BOT to perform malicious activities without user’s knowledge and permission.

Basebridge Malware Family

Basebridge is Trojan malware family discovered in 2011. This malware family has the capability of sending SMS, remove SMS from inbox, dial phone call and sending critical data like IMSI, IMEI the device information to the remote servers and acts as a spyware for the malware author. It also blocks the data consumption monitoring system of the mobile so that it can perform its malicious activities covertly. It also kills the detection process of antivirus.

GingerMaster Malware Family

GingerMaster is a Trojan malware family which has the dangerous root access capability and it takes control of the device. It can install and uninstall malicious apps without user permission. It makes SQLite database having information about the phone number, android version, IMEI, Network type, a list of installed applications on the device and sends it to the remote server. It often bundles itself in the benign application and hide its identity.

Iconosys Malware Family

Iconosys is another malware family that acts as a sniffer to the android user as it watches your all browsing activities, tracks locations, reads contact list, access text messages, credit card details, account credentials and sends it to the malware author. It can read SMS and write SMS, can process ongoing calls, record audio, call any number without user's knowledge and can read and write external storage on the device.

5.2 Permission Feature

In the dataset generated by AndroData tool, the permission feature of benign and malicious applications have 220 Android Permissions and Custom permissions. As permission requested is an important feature in the analysis of the behavior of the malicious application, this is observed that malicious application requests for more no. of permissions than benign applications. The Figure 3 shows there are maximum 32 types of Permissions requested by the benign application whereas the malicious applications have requested a maximum of 110 types of permissions. Further, the Figure 4 shows these permissions requested are from all Permissions category i.e. Normal, Dangerous, System and Signature category. The Normal, Signature and System permissions are granted automatically whereas Dangerous permissions are granted by the End User. It is observed that the malicious application requests more no. of permissions from almost each category.

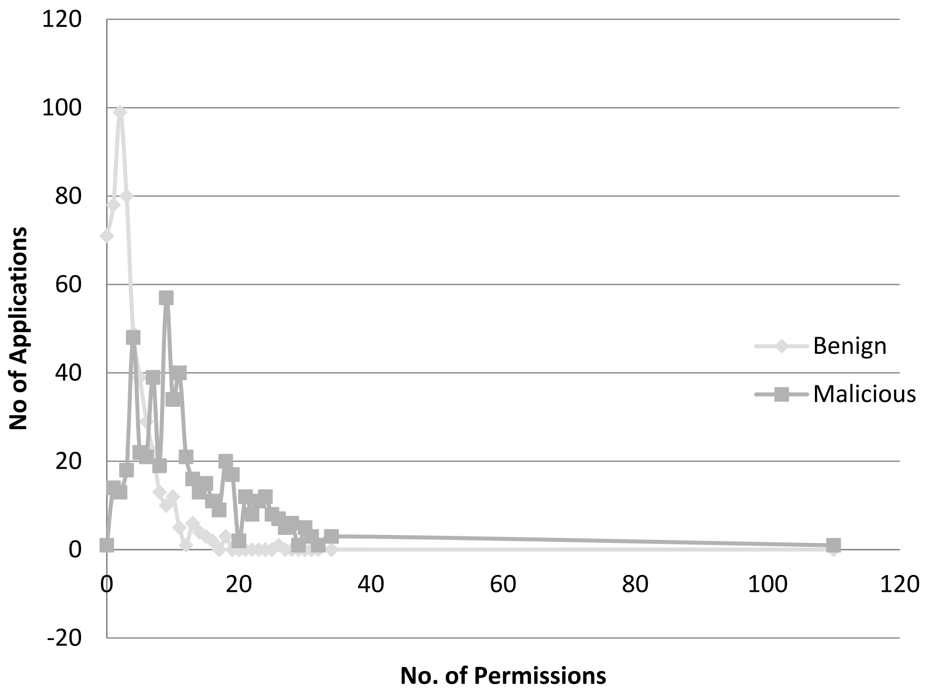

Figure 3: Types of Permission and their frequency in Dataset

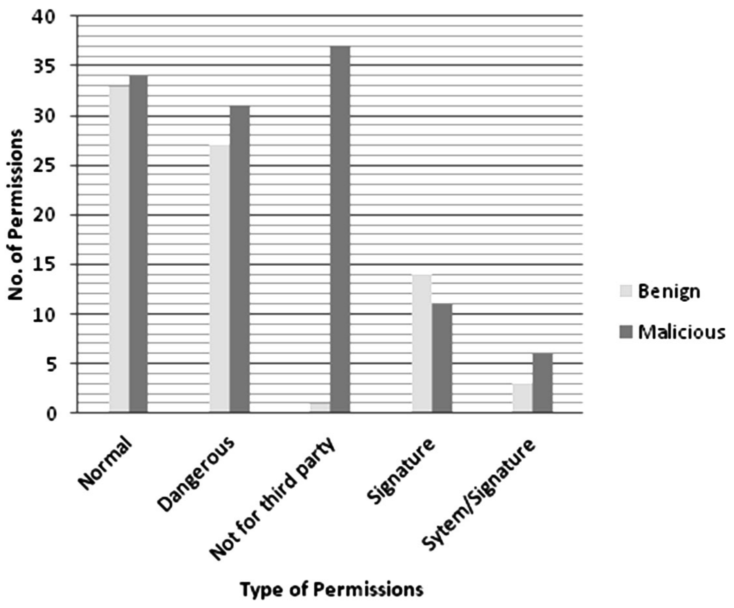

Figure 4: Application Permission Request Frequency

The 28.5% permissions in the dataset of malicious applications and 34.6% permissions in the dataset of benign applications are custom permissions. The custom permission is the permission declared by the application for authorizing the other application for accessing its components. The user defined permission can be a reason for Permission Escalation Attack and TOCTOU attack as discussed earlier.

In Both Dataset, Benign and Malicious some common permissions are requested. Table 1 shows the percentage of Normal permissions requested by benign and malicious applications. Normal permissions are granted to the applications automatically. Granting some permissions in the table does not harm the device e.g. SET_WALLPAPER, FLASHLIGHT, Set_WALLPAPER_HINT, EXPAND_STATUS_BAR, BATTERY_STATS but granting some permissions for e.g. INTERNET, ACCESS_NETWORK_STATE, RECEIVE_BOOT_COMPLETED, WAKE_LOCK, VIBRATE, GET_TASKS etc. can make malicious applications to misuse these permissions.

The INTERNET permission which is requested by 74.2 % benign and 95.1 % malicious applications allow the applications to access the internet via opening network sockets for data transfer but a malicious application can misuse this permission for sending confidential data to unknown URL and breaching the End User privacy.

The ACCESS_NETWORK_STATE requested by 51.0 % of benign applications and 67.7 % of malicious applications allows the application to access information about types of network available, types of network devices connected, roaming or local networks and no. of failed connection attempts. This basic information is needed by the application before connecting to network however this can be misused by malicious application for user profiling.

ACCESS_WIFI_STATE requested by 11.4 % benign and 44.1 % malicious applications allows the application to access information about wifi network which may result in hacking the wifi network by using this information. BLUETOOTH permission requested by 1.7 % benign and 3.0 % malicious applications and BLUETOOTH_ADMIN requested by 1.5 % benign and 2.1 % malicious application are the permissions for connecting to Bluetooth devices and also pairing a new Bluetooth device respectively. These permissions can be used by the malicious application for sending data to unknown mobile via Bluetooth.

Table 1
Normal Permission Request % in Benign and Malicious Application

Android Permission	*Benign* (%)	*Malicious* (%)
INTERNET	74.2	95.1
ACCESS_NETWORK_STATE	51.0	67.7
RECEIVE_BOOT_COMPLETED	9.1	47.1
ACCESS_WIFI_STATE	11.4	44.1
WAKE_LOCK	17.1	37.7
VIBRATE	19.5	30.4
INSTALL_SHORTCUT	2.8	28.1
CHANGE_WIFI_STATE	2.5	16.1
UNINSTALL_SHORTCUT	0.8	15.4
RESTART_PACKAGES	2.3	13.3
GET_TASKS	1.7	12.6
ACCESS_LOCATION_EXTRA_COMMANDS	2.1	11.8
SET_WALLPAPER	2.5	8.8
DISABLE_KEYGUARD	1.7	7.5
CHANGE_NETWORK_STATE	1.1	6.9
BLUETOOTH	1.7	3.0
SET_ALARM	0.2	2.8
KILL_BACKGROUND_PROCESSES	0.8	2.3
BLUETOOTH_ADMIN	1.5	2.1
FLASHLIGHT	1.5	1.7
BATTERY_STATS	0.6	1.5
MODIFY_AUDIO_SETTINGS	1.3	1.3
Set_WALLPAPER_HINTS	0.2	1.1
EXPAND_STATUS_BAR	0.2	1.1
GET_PACKAGE_SIZE	0.0	1.1
BROADCAST_STICKY	0.6	0.8
INSTALL_SHORTCUT	0.0	0.8
WRITE_SYNC_SETTINGS	0.2	0.6
PERSISTENT_ACTIVITY	0.2	0.6
REORDER_TASKS	0.0	0.6
CHANGE_WIFI_MULTICAST_STATE	0.2	0.4
READ_SYNC_SETTINGS	0.2	0.2
READ_SYNC_STATS	0.2	0.2
SET_TIME_ZONE	0.0	0.2
SUBSCRIBED_FEEDS_READ	0.0	0.2

WAKE_LOCK of SYSTEM_TOOLS Group another permission requested by 17.1 % benign applications and 37.7 % malicious applications helps in preventing the malicious application to stop while the phone is in sleep mode. VIBRATE requested by 19.5 % benign applications and 30.4 % malicious applications can prevent the notification of functionality of the malicious application and does its malicious activity without user's knowledge.

RECEIVE_BOOT_COMPLETED permission helps the malicious application to start immediately after booting completion which is requested by 9.1 % benign applications and 47.1 malicious applications. GET_TASKS permission of SYSTEM_TOOLS can be misused by malicious application for spying on the activities performed by the user. The dataset shows it is requested by 1.7 % benign and 12.6 % malicious applications. As explained above, the permission with Normal Protection Level can prove dangerous when it is misused by a malicious application.

Table 2
Dangerous Permission Request % in Benign and Malicious Application

Permission	***Benign* (%)**	***Malicious* (%)**
READ_PHONE_STATE	22.6	87.6
WRITE_EXTERNAL_STORAGE	27.5	66.8
SEND_SMS	1.3	51.4
RECEIVE_SMS	0.9	36.2
READ_SMS	0.4	35.6
ACCESS_COARSE_LOCATION	15.4	33.4
ACCESS_FINE_LOCATION	17.1	31.5
READ_CONTACTS	5.1	23.3
WRITE_SMS	0.2	21.4
READ_HISTORY_BOOKMARKS	0.6	20.6
WRITE_HISTORY_BOOKMARKS	0.4	18.6
CALL_PHONE	5.3	11.3
WRITE_CONTACTS	2.1	9.0
GET_ACCOUNTS	4.4	8.4
READ_EXTERNAL_STORAGE	1.9	6.4
CAMERA	6.3	5.1
PROCESS_OUTGOING_CALLS	0.0	5.1
RECEIVE_MMS	0.4	3.9
RECORD_AUDIO	2.8	2.4
RECEIVE_WAP_PUSH	0.0	2.3
WRITE_CALENDAR	0.0	1.3
READ_CALENDAR	0.2	0.9
ACCESS_MOCK_LOCATION	2.3	0.8
WRITE_OWNER_DATA	0.2	0.8
READ_OWNER_DATA	0.0	0.8
AUTHENTICATE_ACCOUNTS	0.4	0.2
MANAGE_ACCOUNTS	0.8	0.2
USE_CREDENTIALS	0.9	0.2
RECORD_VIDEO	0.0	0.2
SUBSCRIBED_FEEDS_WRITE	0.0	0.2
CLEAR_APP_CACHE	0.6	0.9

Table 2 lists the dataset of permissions with dangerous protection level requested by the benign and malicious applications both. Here the role of End User comes, as we know these permissions are granted to the malicious applications by End User himself. READ_PHONE_STATE is the first permission in the table requested by 87.6 % of malicious applications and 22.6% of Benign applications. This permission gives access to critical data of phone like IMEI/IMSI device identifier, Phone Number, Network Operator, Voice Mail Box, SIM ID which can help the malware author to keep track of your phone and can involve your device in malicious activities.

WRITE_EXTERNAL_STORAGE is another permission requested by 66.8 % malicious applications and 27.5 % benign applications which give permission to malicious applications to write their malicious codes on external storage and making their multiple copies. Granting this permission automatically grants the permission READ_EXTERNAL_STORAGE to the application.

The permissions SEND_SMS requested by 1.3 benign applications and 51.4 % malicious applications, RECEIVE_SMS requested by 0.9 % benign applications and 36.2 % malicious applications, READ_SMS requested by 0.4 % benign applications and 35.6 % malicious applications, WRITE_SMS requested by 0.2 % benign applications and 21.4 % malicious applications belongs to COST_MONEY Group that helps the malicious application to read, write and send user's personal information to the malware author.

ACCESS_COARSE_LOCATION requested by 15.4 % of benign applications and 33.4 % malicious applications to access the approximate location of the mobile device from cell towers or Wifi and ACCESS_FINE_LOCATION requested by 17.1% applications and 31.5 % malicious applications is used to access precise location from GPS, Cell towers and WiFi of the mobile device. These two permissions are used by the malicious applications for location based sniffing and used by the benign application to display location based ads.

The READ_CONTACTS requested by 5.1 % benign applications and 23.3 % malicious applications and WRITE_CONTACTS requested by 2.1 % benign applications and 9.0 % malicious applications are the permissions of Personal Info Group that permit the malicious applications to burgle the personal info of victim.

CALL_PHONE requested by 5.3% benign applications and 11.3 % requested by malicious applications allows the malicious applications to make a phone call to the anonymous number without user's knowledge. The PROCESS_OUTGOING_CALLS requested by only 5.1 % malicious applications allows to process outgoing calls by monitoring, modifying and even aborting the ongoing calls.

COM.ANDROID.BROWSER.PERMISSION.READ_HISTORY_BOOKMARKS requested by 0.6% benign applications and 20.6 % malicious applications and COM.ANDROID.BROWSER. PERMISSION.WRITE_HISTORY_BOOKMARKS requested by 0.4% benign applications and 18.6 % malicious applications help the malicious application to spy on the online activities of user's mobile. Table 3 shows the list of System/Signature/Not for third party permissions, requested by the malicious and benign applications. The maximum 15.2 % of malicious applications requested these types of applications as these permissions have the highest protection level but surprisingly malicious applications still request these permissions like INSTALL_PACKAGES, WRITE_SETTINGS, READ_LOGS, WRITE_APN_SETTINGS, SYSTEM_ALERT_WINDOW,MOUNT_UNMOUNT_FILESYSTEMS, DELETE_PACKAGES, WRITE_SECURE_SETTINGS, UPDATE_DEVICE_STATS. Tabel 4 summarizes the Use and Misuse of various Android Permission.

Table 3
Signature/System/Not for Third party Permission Requests % in Benign and Malicious Applications

Permission	***Benign* (%)**	***Malicious* (%)**
INSTALL_PACKAGES	0.4	15.2
WRITE_SETTINGS	4.7	10.1
READ_LOGS	0.8	9.6
WRITE_APN_SETTINGS	0.0	9.6
SYSTEM_ALERT_WINDOW	0.6	6.0
MOUNT_UNMOUNT_FILESYSTEMS	0.8	4.5
DELETE_PACKAGES	0.4	4.1
WRITE_SECURE_SETTINGS	0.9	3.9
UPDATE_DEVICE_STATS	0.0	3.0
DELETE_CACHE_FILES	0.0	2.4
ACCESS_CACHE_FILESYSTEM	0.0	2.3
MODIFY_PHONE_STATE	0.0	2.3
DEVICE_POWER	0.2	1.5
SET_PREFERRED_APPLICATIONS	0.2	1.3
STATUS_BAR	0.2	1.1
CONTROL_LOCATION_UPDATES	0.2	0.9
READ_FRAME_BUFFER	0.0	0.8
BROADCAST_SMS	0.0	0.8
BROADCAST_WAP_PUSH	0.0	0.8
INTERNAL_SYSTEM_WINDOW	0.4	0.6
CHANGE_COMPONENT_ENABLED_STATE	0.0	0.6
REBOOT	0.0	0.6
CALL_PRIVILEGED	0.4	0.4
HARDWARE_TEST	0.4	0.4
SET_ORIENTATION	0.4	0.4
BRICK	0.0	0.4
GLOBAL_SEARCH_CONTROL	0.0	0.4
MOUNT_FORMAT_FILESYSTEMS	0.0	0.4
BIND_WALLPAPER	0.2	0.2
BROADCAST_PACKAGE_REMOVED	0.2	0.2
CLEAR_APP_USER_DATA	0.0	0.2

Table 4
Use and Misuse of Permissions in Android Application.

ANDROID PERMISSION	*USE*	*MISUSE*
INTERNET	Allows the applications to access the internet via opening network sockets for transferring data	Granting this permission can be misused by sending confidential user data to unknown URL.
ACCESS_NETWORK_STATE	Allows the application to access information about type of network available, type of network devices connected, roaming or local network and no. of failed connection attempts.	Misused by malicious application for maintaining user profile regarding his network information.
ACCESS_WIFI_STATE	Allows the application to access information about Wi-Fi network	Can help the malicious application in hacking the Wi-Fi network and sending user data by using this information
BLUETOOTH,BLUETOOTH_ADMIN	Connecting to Bluetooth devices and also pairing a new Bluetooth device	Used by the malicious application for pairing with unknown device and sending data to it or blocking the Bluetooth connectivity.
WAKE_LOCK	Control the mobile device's Sleep Mode	Can be used by malicious application for preventing the device go into sleep mode. Thus malicious code runs continuously and drains out the device battery.
VIBRATE	Control the device's Vibration Mode.	Prevents the notification of functionality of malicious application and carries out its malicious activity silently without user's knowledge
RECEIVE_BOOT_COMPLETED	Notifies about the completion of Booting of Mobile Device	Helps the malicious application to start in background immediately after Boot Completion
GET_TASKS	Allows an application to get information about the currently or recently running tasks.	Misused by malicious application for spying on the activities performed by the user
READ_PHONE_STATE	Gives access to critical data of phone like IMEI/IMSI device identifier, Phone Number, Network Operator ,Voice Mail Box, SIM ID etc.	Helps the malware author to keep track of your phone and can involve your device in malicious activities using this information.
WRITE_EXTERNAL_STORAGE, READ_EXTERNAL_STORAGE	Allow to read or write external storage	Malware can read confidential data of user and write its malicious code on external storage.
SEND_SMS, RECEIVE_SMS, READ_SMS, WRITE_SMS	Allow the activities related to SMS	Help the malicious application read, write and send user's personal information to the malware author.
ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION	Permissions to access location related information of mobile device	These two permissions are used by the malicious applications for location based sniffing
READ_CONTACTS, WRITE_CONTACTS	Allow to read and write contact list	Permit the malicious applications to burgle the personal info of victim
CALL_PHONE, PROCESS_OUTGOING_CALLS	Permissions to do phone call and control ongoing calls	Make phone call to anonymous number without user's knowledge
READ_HISTORY_BOOKMARKS, WRITE_HISTORY_BOOKMARKS	Permissions to read and write History Bookmark files of Internet Browsing activities.	Malicious application can spy on the internet browsing activity of user using this information.

6. CONCLUSION

As discussed earlier, there are four protection levels of permission: Normal, Dangerous, Signature and System. A user has little control in granting Normal, Signature and System permission as these are automatically granted by the Android Permission model but granting Dangerous permission is in the hands of the end user. The Normal permission is harmless is a fallacy. As discussed in this work, this permission can be misused by the malicious application in a number of ways. The Normal Permission that can be misused by the malicious application should be kept in a dangerous or signature category by Android Permission Model. The Android Permission documentation should be improved to prevent security issues like over claim of permissions. The Android Permission Model also needs improvement to avoid security issues like permission escalation attack, naming collision and security breaches due to course granularity of permission.

References

[1] L. Gomez and I. Neamtiu, “A Characterization of Malicious Android Applications,” Technical report, University of California, Riverside, 2011.

[2] W. Tang, G. Jin, J. He, and X. Jiang, “Extending Android security enforcement with a security distance model,” in Internet Technology and Applications (iTAP), 2011

[3] M. C. Grace, Y. Zhou, Z. Wang, and X. Jiang, “Systematic Detection of Capability Leaks in Stock Android Smartphones,” in NDSS, 2012.

[4] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, “Android permissions: a perspective combining risks and benefits,” in Proceedings of the 17th ACM symposium on Access Control Models and Technologies, pp. 13–22, 2012.

[5] E. Struse, J. Seifert, S. U¨ llenbeck, E. Rukzio, and C. Wolf, “PermissionWatcher: Creating User Awareness of Application Permissions in Mobile Systems,” in Ambient Intelligence, pp. 65–80,2012.

[6] G. Dini, F. Martinelli, I. Matteucci, M. Petrocchi, A. Saracino, and D. Sgandurra, “A Multi-Criteria-Based Evaluation of Android Applications,” in Trusted Systems, pp. 67–82, 2012.

[7] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. G. Bringas, “On the automatic categorization of android applications,” in Consumer Communications and Networking Conference (CCNC),2012 IEEE, pp. 149–153, 2012.

[8] 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.

[9] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, “Malicious android applications in the enterprise:What do they do and how do we fix it?[1],” in Data Engineering Workshops (ICDEW),IEEE 28th International Conference on, pp. 251–254, 2012.

[10] S. Holavanalli, D. Manuel, V. Nanjundaswamy, B. Rosenberg, F. Shen, S. Y. Ko, and L. Ziarek,“Flow permissions for android,” in Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, pp. 652–657,2013.

[11] Y. Zhang, M. Yang, B. Xu, Z. Yang, G. Gu, P. Ning, X. S. Wang, and B. Zang, “Vetting undesirable behaviors in android apps with permission use analysis,” in Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 611–622, 2013.

[12] S. Rosen, Z. Qian, and Z. M. Mao, “Appprofiler: a flexible method of exposing privacy-related behavior in android applications to end users,” in Proceedings of the third ACM conference on Data and application security and privacy, pp. 221–232, 2013.

[13] R. Sato, D. Chiba, and S. Goto, “Detecting Android Malware by Analyzing Manifest Files,” Proceedings of the Asia-Pacific Advanced Network, vol. 36, pp. 23–31, 2013.

[14] V. Rastogi, Y. Chen, and X. Jiang, “Droidchameleon: evaluating android anti-malware against transformation attacks,” in Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 329–334,2013.

[1]1

[15] G. Canfora, F. Mercaldo, and C. A. Visaggio, “A classifier of malicious android applications,” in Availability, Reliability and Security (ARES), pp. 607–614, 2013.

[16] J. Zhu, Z. Guan, Y. Yang, L. Yu, H. Sun, and Z. Chen, “Permission-based abnormal application detection for android,” in Information and Communications Security, pp. 228–239,2012.

[17] Z. Aung and W. Zaw, “Permission-based android malware detection,” International Journal Of Scientific & Technology Research, vol. 2, no. 3, 2013.

[18] M.-Y. Su and W.-C. Chang, “Permission-based malware detection mechanisms for smart phones,” in Information Networking (ICOIN), 2014 International Conference on, pp. 449–452, 2014.

[19] Z. Fang, W. Han, and Y. Li, “Permission based Android security : Issues and countermeasures.”Computers and Security, Volume 43,pp-205-218,2014

[20] D. Arp, M. Spreitzenbarth, H. Malte, H. Gascon, and K. Rieck, “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket,” Symp. Netw. Distrib. Syst. Secur., pp. 23–26, 2014.

[21] Hyunjae Kang, Jae-wook Jang, Aziz Mohaisen, and Huy Kang Kim, “Detecting and Classifying Android Malware Using Static Analysis along with Creator Information,” International Journal of Distributed Sensor Networks,pp-7, 2015.

[22] Sapna Malik,Kiran khatter,” AndroData:a static and dynamic feature extraction tool for android application”, International Journal of Applied Engineering Research, Volume 10,Issue 94, pp-98-102,2015