International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 15169-15178
© Research India Publications. http://www.ripublication.com

Static Analysis Tool for Identification of Permission Misuse by Android Applications

Karthick S
PG Scholar, Department of Computer Science,
Christ University, Hosur Road, Bengaluru, Karnataka, India.

Orcid Id: 0000-0002-1176-4714 & Researcher Id: Q-2455-2017

Sumitra Binu
Assistant Professor, Department of Computer Science,
Christ University, Hosur Road, Bengaluru, Karnataka, India.

Orcid Id: 0000-0001-7658-3829

Abstract

Android is one of the most important and widely used mobile operating systems in the world. The Android operating system utilizes the permission-based model, which permits Android applications to get user data, framework data, gadget data and other assets of Smartphone. These permissions are affirmations declared by the developer of an application. The permissions granted varies from one application to another, depending on its functionality. During installation, permissions to access the resources of the smartphone are requested by apps. Once the client grants the permission, the apps are allowed to access the granted resources as per its requirement. Android OS is susceptible to different security issues owing to the loopholes in security. This paper mainly focuses on identifying how the permissions granted to a specific application is misused by another application using SharedUserID. The paper also proposes a security tool that identifies a list of applications which are misusing the permissions in a user's Android smartphone. The viability of the tool is tested by using a Proof-of-Concept (PoC) implementation of the security tool.

Keywords: Android, Over-claiming of Permissions, Permission Misuse, Security Attacks, Security Threats, Security Tool App, SharedUserID.

INTRODUCTION

A flexible operating system (OS) enables PDAs, tablets, PCs, and different devices to run applications and activities. Usage of smartphones are not only restricted to calling and messaging services but also encompasses various tasks such as Internet banking, Online payment, etc. Though there are different types of mobile operating systems available in the market, the most frequently utilized operating systems are Android, iOS, Windows and BlackBerry OS. Android is the most widely used, amongst the mainstream operating systems for smartphones. At the first quarter of 2017, the total number of verified applications available in Google play store was 2.8 Million [1], and a total number of Android operating system-based smartphones sold was 2.1 Billion [2]. It is predicted that the sales of Android-based smartphones will exceed 5 Billion by 2019 [2]. The market shares of Android OS in the last quarter of 2016 was 86.2% whereas iOS, Windows, BlackBerry, and others hold 12.9%, 0.6%, 0.1% and 0.2% respectively [3]. These statistics throws light on the fact that the Android operating system has the most extensive market when compared to other mobile operating systems. Android is an open-source operating system based on Linux-Kernel released by Google under Apache license targeted for smartphones and tablets. iOS (iPhone OS) created by Apple Inc. is utilized just by Apple gadgets, such as iPhone, iPad, and iPod touch. It is the second most prevalent operating system alongside Android.

The rest of the paper is organized as follows: Section II describes related work. Section III and IV discuss various security threats and security attacks in Android, respectively. Section V examines how applications can misuse permissions using SharedUserID. Section VI includes a discussion of different types of Android Permissions and a brief comparison of Android and iOS. Section VII discusses the proposed method for addressing permission escalation using SharedUserID. Section VIII and IX include the methodology and implementation of Security tool app. Section X concludes the work.

RELATED WORK

Android is an open source operating system. In Android, it is possible to install an application from unknown sources. An application whose source is anonymous and is not verified can be installed in the smartphone by unchecking the "Unknown Sources" option, available in the Settings. Google Play Store is extensively used to publish applications worldwide. Even though it is widely used to download the validated apps, not all applications available in Google Play Store are completely secure. As per the reports published by McAfee, the presence of three Android Malware was detected in many games and apps uploaded on the play store [4]. Also, a vast majority of apps request for a host of permission to access information and functions that are not required and thereby misused. The main functionality of the application "FlashLight" released by Happy Hollow Studio available in Google Play Store [5] is to provide flashlight. The permissions demanded by this application are Location, Photo/Media/Files, Camera, WiFi Connection and Device ID & Call Information. The only permission required to perform flashlight functionality is "Camera." However, this application requires the user to accept few more permissions which are entirely irrelevant for its functioning. Hence, it can be understood that the fact that the app is available in Google play store does not make it completely secure and that there are apps that over claims the permissions from the user. Sometimes, the same developer may develop more than one application with the same digital signature to sign these applications and SharedUserID set in AndroidManifest.xml file [6]. This kind of apps can cause the misuse of permissions granted to one app by another app which in turn can result in permission escalation attacks [6].

The security loopholes in Android Operating System are being exploited by hackers to steal confidential information stored in the smartphones by using different malware embedded in various apps.

In 2012, Barrera et al.in their work, has deliberated on how the SharedUserID concept works [7]. In 2013, Sbirlea et al. indicated how SharedUserID concept is implemented [8]. In 2013, Ahmad et al. proposed a comparative study between Android and iOS operating systems and claimed that iOS is more secure than Android [9]. In 2014, Kaur et al. proposed a security tool application called PeMo – Permission Modificator, which is used to revoke granted irrelevant permissions from installed applications [10]. In 2014, Z. Fang et al. thoroughly analyzed Android OS permissions issues and countermeasures. Their findings included various permission escalation attacks and existing countermeasures to solve those problems [11]. In 2014, Ratazzi et al. in their work have discussed the problem of permissions being misused by Android apps using SharedUserID. During installation of an Android application, permission sets for each user are stored in a separate xml file called packages.xml and installation status is stored in an xml file called package-restrictions.xml. For applications, whose SharedUserID are set, permissions from the apps sharing the ID are combined within the <shared-user> block in the packages.xml file. During booting, the package manager loads this permission list into hashmap mPackage, rendering it impossible to differentiate the individual permissions from the corresponding SharedUserID application. Therefore, when apps are developed by the same developer and are signed by the same digital certificate, those apps acquire the combination of permissions from all of the SharedUserID apps installed on the particular platform [12]. In 2015, Ankita et al. identified various security threats like Virus, Worm, Trojan horse, Cyber-attack, Botnet, Rootkit, Privilege escalation vulnerability etc. [13]. In 2015, Faruki et al. classified the Android permissions into normal and dangerous permissions. A developer must declare dangerous permissions explicitly whereas normal permissions are granted automatically. They also identified Android threats like version update threats, malicious apps to steal the personal information and security attacks like Trojan, Backdoor, Worm, etc., [14]. In 2015, B. Rashidi et al. has mentioned that more than 70% of applications collect data that is irrelevant to their functionality [15]. In 2015, J.K Park et al. indicated that excessive authorization, abuse of app permissions and information leakage are possible by using SharedUserID [16]. In 2016, Google has released a white paper on Android Security [17], but there was no solution mentioned for addressing the misusing of app permissions.

In 2017, Karthick et al. [18] in their work have deliberated on how Android apps can abuse the permissions using SharedUserID and proposed a methodology to solve the related problems. This paper is an extension of the work in which a new algorithm is proposed to identify the applications that misuse the app permissions using SharedUserID. The Proof of Concept (PoC) of the algorithm is implemented by developing a Security Tool App which detects complete information about the applications that exploit SharedUserID for accessing permissions not granted to them. SharedUserID is an attribute in the AndroidManifest.xml file. If this attribute is assigned the same value for one or more applications and if the same certificate signs both the applications, then, they can share the resources granted to each other [18].

Android versions 6.0 and above provides explicit notification to the user whenever an app tries to access dangerous permissions. Thus, explicit notification is provided whenever an app seeks to access sensitive resources such as contacts, messages etc. and also it is possible to turn on or off the crucial permissions of each app in the app settings. However, Marshmallow 6.0 is available only on 7.5 percent of Android devices [21]. Nougat 7.0 is available only on 4.5% of Android devices, and Nougat 7.1 is available only on 0.4% of Android devices [21,22]. The Android OS updates are not available for most of the older devices. Therefore, security issues related to application permissions are still not solved.

In the reviewed literature, authors fail to identify works that addresses identification of permission misuse by apps developed by the same developer and signed by same digital signature. The proposed work addresses these concerns by designing an algorithm for permission misuse identification. The viability of the algorithm is verified by a proof of concept (PoC) implementation of the same.

Security Threats in Android

This section briefly discusses a few relevant security threats in Android Operating System.

A. Repackaging

Repackaging is one of the significant security threats in Android. Every Android application is installed using .apk file. Repackaging is the procedure of dismantling or decompiling .apk files by utilizing reverse-engineering techniques and by including or infusing vindictive code into the actual source code. Repackaging methods can be used on the Android platform to allow malicious code to be veiled as an ordinary application. It is difficult to recognize a repackaged malignant code and a genuine application because the repackaged application typically seems to work in the same way as, the actual one [15].

B. Information Leakage

In Android Operating System, significant resources of a smartphone are accessed using permissions. Acceptance of the listed permissions by the user, allows the application to access the permitted resources whenever needed. According to the survey conducted by Jae-Kyung Park [16], it has been identified that more than 70% of apps leaks and misuse the information [16]. The survey results also reveal that only 3% of the Android users are cautious about app permissions and its importance [16].

C. Privilege Escalation

Privilege escalation happens when a client accesses a larger number of resources or functionality than they are normally permitted and such rise or changes ought to have been thwarted by the application. It is caused by an imperfection or flaw in the application. The outcome is that the application performs activities with more privileges than those are anticipated by the system administrator [15].

D. Data Theft

Data theft or privacy leakage occurs at the point when the client allows hazardous or dangerous permissions to malevolent applications and accidentally enables privilege to access sensitive information and ex-filtrate them without client consent [14].

E. Version Update Issues

The updates and upgrades released by an Android OS are more frequent than desktop OS. Android apps access the resources based on permissions granted to it. The older version of an application is modernized by adding new functionalities based on the update of the operating system. An application developed for the earlier version can be over claimed to utilize the hazardous or dangerous permission(s) added in the higher version release of an app. During the update, Android does not verify the newly added permissions in the updated application [14].

F. Malicious Apps

In Android, it is possible to install an application from unknown sources. The application launched by anonymous sources is dangerous because its source code is not verified and it may contain some malicious code to access confidential information from the device and to send the information to its server without the consent of user [14].

Security Attacks in Android

A few security attacks in Android are explored in this section.

A. Collision or Colluding Attack

Colluding is a client-side assault. Android OS supports SharedUserID [6,11,12,14,15,16] which is the main reason for Colluding or Collision and Permission Escalation attacks. It occurs when a set of applications are, signed with the same digital certificate, assigned same value in the SharedUserID attribute and are installed on the same device. In this attack, clients install applications created by the same developer and signed with the same digital signature and allow distinctive sorts of permissions including sensitive and non-sensitive. These applications can exploit a common SharedUserID and access union of permissions granted to each of them [12,14,15,18].

B. Permission Escalation Attacks

It enables a noxious application to work together with different applications to gain access to critical resources without asking for corresponding permissions explicitly [18].

C. Denial of Service Attack

It happens when the application overuses the sparse resources such as CPU, memory, battery and bandwidth [14,15].

D. Time of Check to Time of Use Attack (TOCTTOU)

It occurs due to naming collision. No naming standard or imperative is applied to new permission declaration. Besides, permissions in Android are given as strings, and any two permissions with a similar name string are treated as same even if they belong to different applications [11,18].

E. Spyware

Spyware is a kind of malware. It is an apk file, downloaded automatically when the client visits the vindictive site and installs the applications from unidentified sources. In Android, other than from google play store, it is viable to install the applications from unknown sources. Spyware is one of the primary reasons behind significant security dangers in the Android operating system. Spyware is hidden and collects the data and transmits it to the remote server [11,14,15,18].

F. Ransomware

Ransomware can lock the client gadget to make it unavailable until some ransom amount is paid online. It displays the fake malware cautions to lure the user to install the swindle malware [14].

G. Aggressive Adware

Aggressive adware can make shortcuts on the home screen, take bookmarks, change the default web settings and can push irrelevant notifications to thwart the gadget utilization [14].

H. Botnet

Botnet applications make a remote server to control a victim's device through a series of commands. Bot also has commands to download vindictive apps automatically [14].

I. Back Door

Backdoor permits other malware to enter the device, which overrides the security mechanisms quietly. It can perform root exploits to acquire the superuser privilege. By using superuser privilege, the malware can be hidden from anti-malware scanning software [14].

J. Worm

Worm application can make exact or comparative duplicates of itself and spreads them by using networking or removable Medias [14].

K. Trojan

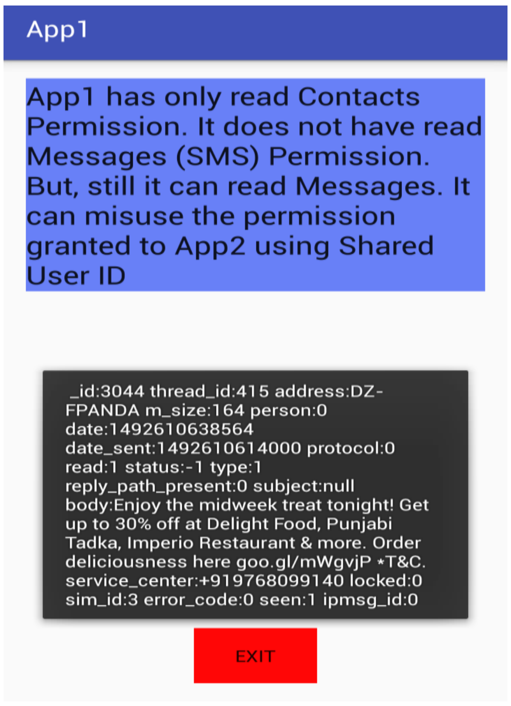
Trojans take on the appearance of useful applications. However, they perform unsafe exercises without the consent of the clients.

Permission Misuse between Android Applications

This section shows how two applications can misuse the permissions granted to each other using SharedUserID attribute.

In the Android operating system, all the information related to an application is available in AndroidManifest.xml file. It is an xml file, which comprises various information about the application like appName, appIcon, Permissions, Activities, Package Name, etc., Android uses permission-based model, and its permissions are classified into Normal and Dangerous Permissions. The app needs to request the user to accept the dangerous permissions whereas the normal permissions are granted by default. The dangerous permissions must be declared in the AndroidManifest.xml file prior to their usage.

To gain an understanding of "Permission Misuse" by apps, consider two applications developed by the same developer and named as App1 and App2. App1 has only Read Contacts permission and hence can read the user contacts stored on the smartphone. App2 has only Read SMS (Short Message Service) permission and it can read the SMS stored in the smartphone.

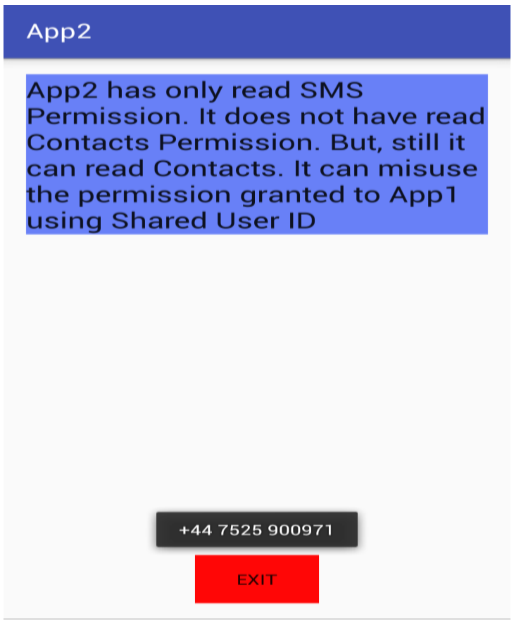


App1 has only read Contacts Permission. It does not have read Messages (SMS) Permission. But, still it can read Messages. It can misuse the permission granted to App2 using Shared User ID

```
 _id:3044 thread_id:415 address:DZ-
FPANDA m_size:164 person:0
date:1492610638564
date_sent:1492610614000 protocol:0
read:1 status:-1 type:1
reply_path_present:0 subject:null
body:Enjoy the midweek treat tonight! Get
up to 30% off at Delight Food, Punjabi
Tadka, Imperio Restaurant & more. Order
deliciousness here goo.gl/mWgvjP *T&C.
service_center:+919768099140 locked:0
sim_id:3 error_code:0 seen:1 ipmsg_id:0
```

Figure 1: App1 misusing permissions of App2

However, App1 can also read SMS which is stored in the user's smartphone even though it does not have ReadSMS permission implicitly. Similarly, App2 can read the Contacts, which are stored in the user's smartphone even though it does not have Read Contacts permission implicitly. As illustrated in Figure 1 and Figure 2, App1 and App2 can exploit the permissions granted to each other.



App2 has only read SMS Permission. It does not have read Contacts Permission. But, still it can read Contacts. It can misuse the permission granted to App1 using Shared User ID

Figure 2: App2 misusing permissions of App1

Hence, it is evident that App1 and App2 can use the permissions granted to each other because SharedUserID is same for both the applications in their respective manifest files and also the same developer has signed both the applications.

Android Permissions and Comparison of Android, IOS

The Android OS utilizes the permission-based model to get different resources and data. These permissions are not demands and they are affirmations specified in AndroidManifest.xml file. As shown in Fig. 3, these permissions are classified into Normal Permissions [19] and Dangerous Permissions [20]. Due to various security threats and attacks on misusing app permissions, Android versions 6.0 and above, provides explicit notification to the user whenever an app tries to access dangerous permissions or sensitive resources. Since the Android OS updates are not available for most of the older devices, security issues related to application permissions are yet to be resolved.

A. Normal Permissions

Normal permissions do not particularly invade the user's privacy. It is not required to declare normal permissions in the AndroidManifest.xml file. Thus, users need not explicitly accept these permissions for successful installation of an application. Normal Permissions are granted automatically. A few examples include:

KILL_BACKGROUND_PROCESSES

REQUEST_INSTALL_PACKAGES

SET_ALARM

SET_WALLPAPER

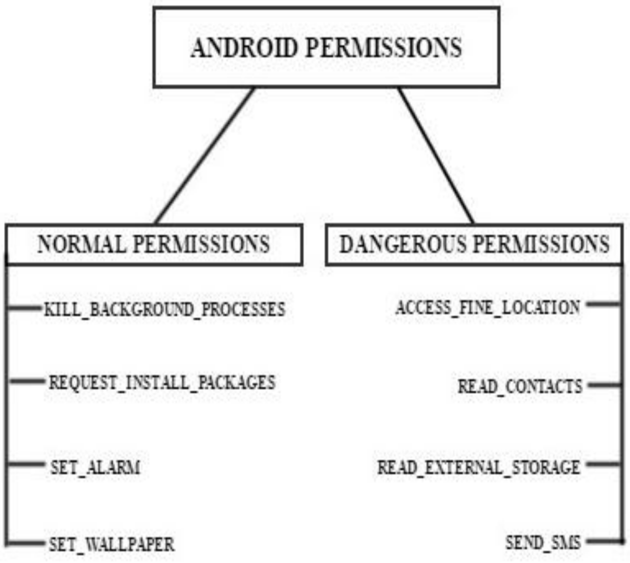


Figure 3: Android Permissions

B. Dangerous Permissions

Dangerous permissions question the user's privacy and developers need to declare dangerous permissions in the AndroidManifest.xml file, prior to their usage. Dangerous Permissions are not provided automatically and users need to grant these permissions explicitly for successful installation of an application. Android versions 6.0 and above provides explicit notification to the user whenever an app tries to access dangerous permissions. A few examples of dangerous permissions include:

ACCESS_FINE_LOCATION

READ_CONTACTS

READ_EXTERNAL_STORAGE

SEND_SMS

The most extensively used mobile operating system in the world is Android, but, from a security perspective, an iOS operating system is considered to be more secure than Android [9]. Understanding the differences between iOS and Android helps to identify the security problems related to Android. The following Table 1, illustrates a comparison between Android and iOS mobile operating systems on the basis of various parameters.

Table 1: Differences between Android and iOS

SL No	**Difference**	**Android**	**iOS**
1	Application Downloads	Android applications can be downloaded from any source. Android applications can be published in various ways like publishing through email, website, Google play store, etc. This facilitates the downloading of Android app even from unknown sources [6,9].	iOS apps can be downloaded only from Play store. It is not possible to publish iOS apps through email, website, etc., except via Play Store. Play Store contains only verified apps. Therefore, iOS applications can be downloaded only from Play Store [6,9].
2	Open Source and Closed Source	Android is an open source OS.	iOS is closed Source.
3	Storage	It supports both internal and external storage.	It supports only internal storage.
4	Signing of Apps	Self-signing used.	Code-signing used.
5	Installation of Apps	Applications from unknown sources can be installed.	Applications downloaded from Play Store only can be installed.
6	Version Updates	Most of Android OS updates are not available for older devices.	Version updates are easily available even for older devices.

PROPOSED WORK

Over-claiming of app permissions is the main reason for permission misuse. Identifying whether an app requires the requested permission is a very challenging task and android SharedUserID is one of the primary reasons for misusing app permissions. Due to SharedUserID, permissions granted to one app can be accessed by another app if and only if both have the SharedUserID value set same and are signed by the same certificate. The users are unaware of the applications which are making an attempt to over-claim and misuse permissions. The discussed work addresses these concerns by proposing an algorithm for permission misuse identification and verifies its feasibility by using a proof of concept (PoC) implementation of the same.

Figure 4 illustrates the system proposed for Permission Misuse Identification that identifies permission misuse by applications installed in the user's smartphone.

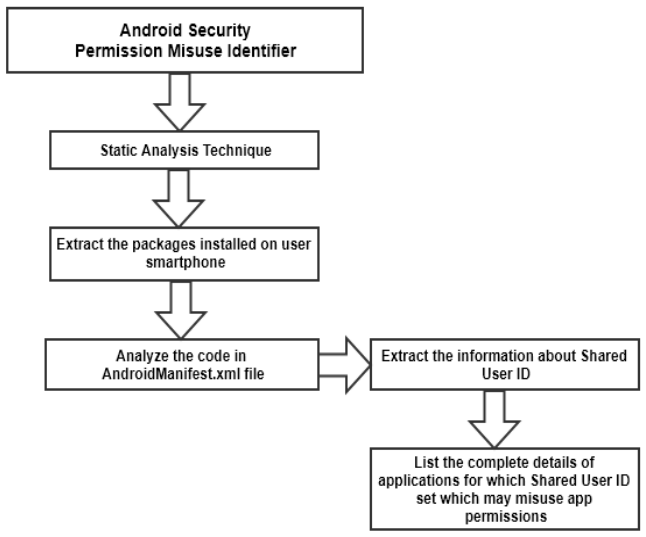

Android Security
Permission Misuse Identifier
Static Analysis Technique
Extract the packages installed on user smartphone
Analyze the code in AndroidManifest.xml file
Extract the information about Shared User ID
List the complete details of applications for which Shared User ID set which may misuse app permissions

Figure 4: Block Diagram of the Proposed System

The proposed system uses Static analysis technique to identify permission misuse by the applications. In Android, all the applications have unique package name. It is possible to extract the information of all the installed applications programmatically. The proposed model analyzes the code in AndroidManifest.xml file of each application to identify whether the SharedUserID is set or not. If the SharedUserID is set, then, it may misuse the permissions granted to other apps. The tool displays the details such as appName, packageName, SharedUserID, etc. of applications that misuse permissions.

METHODOLOGY

The main reason for permission escalation attacks is misusing the app permissions. The proposed work attempts to solve this issue by using:

Permission Misuse Identifier - Android security tool developed to identify and display the list of apps misusing permissions. Algorithm for identifying the list of apps misusing permissions can be explained as follows:

Purpose: To provide a list of installed applications which misuses app permissions using SharedUserID.

Input:

packList	- An object that contains a list of all the installed applications in the Smartphone.
Size	- Stores the total number of installed applications.
packInfo	- An object which stores the detailed information about each package.
SharedIDApp	- Stores shared ID of the application.
PackageName	- Stores the package name of the application.
AppName	- Stores the name of the application.

Output:

Complete details of applications, which misuse the permissions' using SharedUserID in 'AllSharedApps'.

General Terms:

getPackageManager() – It is a part of PackageManager class which is used to get the information related to the packages available in the Smartphone.

getInstalledPackages() – It is a part of getPackageManager() which is used to get the information related to installed applications available on the Smartphone.

PackageInfo – It is a class, which contains all the information available in AndroidManifest.xml file of each package.

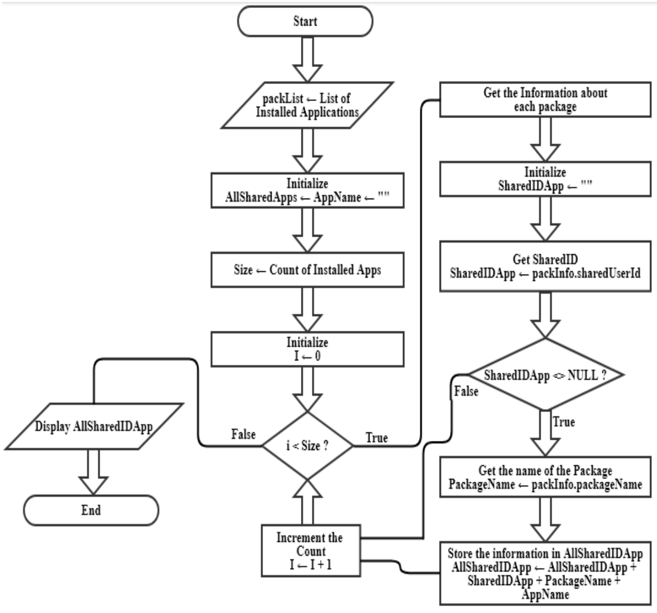

Start
packList ← List of Installed Applications
Initialize AllSharedApps ← AppName ← ""
Size ← Count of Installed Apps
Initialize I ← 0
i < Size ?
False
True
Display AllSharedIDApp
End
Increment the Count I ← I + 1
Get the Information about each package
Initialize SharedIDApp ← ""
Get SharedID SharedIDApp ← packInfo.sharedUserId
SharedIDApp <> NULL ?
False
True
Get the name of the Package PackageName ← packInfo.packageName
Store the information in AllSharedIDApp AllSharedIDApp ← AllSharedIDApp + SharedIDApp + PackageName + AppName

Figure 5: Flowchart of Permission Misuse Identifier

The flowchart in Figure 5 can be explained as follows:

Step 1: Initialize the variables 'AllSharedIDApps' and 'AppName' which is of String type to NULL.

AllSharedIDApps 🡐 AppName 🡐 ""

Step 2: Create an object 'packList', which contains a list of all installed applications in Smartphone. getPackageManager() and getInstalledPackages() used together to create an object which is of List<PackageInfo> type.

List<PackageInfo>packList🡐getPackageManager.getInstalledPackages()

Step 3: Store the total number of installed applications in the variable 'Size' using packList.size() method which returns a total number of installed applications.

Size 🡐 packList.size()

Step 4: Repeat step 4 for I 🡐 0 to Size – 1

Step 5: Get the complete details about each package by using packList.get(I) and store the information in 'packInfo' object which is of PackageInfo type.

PackageInfo packInfo 🡐 packList.get(I)

Step 6: Initialize the SharedIDApp variable to NULL.

SharedIDApp 🡐 ""

Step 7: Get the SharedUserID of an application by using packInfo.SharedUserID and store it in the SharedIDApp.

SharedIDApp 🡐 packInfo.SharedUserID

Step 8: If SharedIDApp = NULL then

goto Step 13

else

goto Step 9.

Step 9: Get the Application name by using packInfo.applicationInfo and store it in the AppName.

AppName 🡐 packInfo.applicationInfo

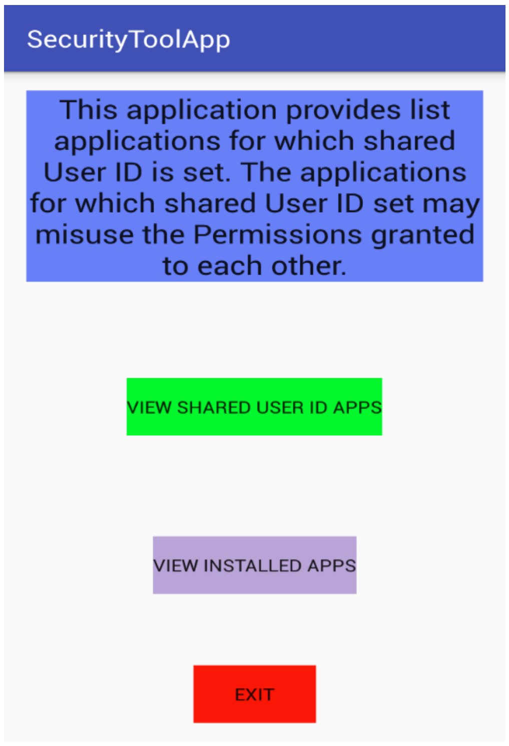
```
Step 10: Get the Package name of an application by using packInfo.packageName and store it in the PackageName.
        PackageName 🡐 packInfo.packageName

Step 11: Combine all the information available in SharedIDApp, PackageName and AppName in AllSharedIDApps
        AllSharedIDApps 🡐 AllSharedIDApps + SharedIDApp + PackageName + AppName
EndIf
[End of Step 4 loop]

Step 12: Display the complete information available in the 'AllSharedIDApps'.

Step 13: Exit
```

IMPLEMENTATION

The implementation of the tool "Permission Misuse Identifier enables to dynamically show the complete list of applications which are misusing app permissions. The Permission Misuse Identifier works with the help of Package Manager Functionality [24]. PackageManager is a part of Application Framework of Android architecture. The application framework provides services to its upper layer "Application" [25]. The Package Manager is the primary programming interface (API) for Android, which manages the installation, up-gradation, and expulsion of an application. The Package Manager with the assistance of Package Installer does all these tasks [10,24,26]. The Package Manager stores the application details such as the list of permissions and packages in the separate XML file, package.xml [26]. During installation of an Android application, permission sets are stored in the separate xml file packages.xml and installation status is stored in the separate xml file package-restrictions.xml for each user. For SharedUserID set applications, permissions from each app are combined and stored within the <shared-user> block in the packages.xml file. During booting, the package manager loads this permission list into hashmap mPackage, since it is impossible to segregate the individual permissions of each SharedUserID application. Therefore, when apps are developed by the same developer and signed by same digital certificate, each app gains the combination of permissions from all of the SharedUserID apps installed on the particular platform [12].

CONCLUSION

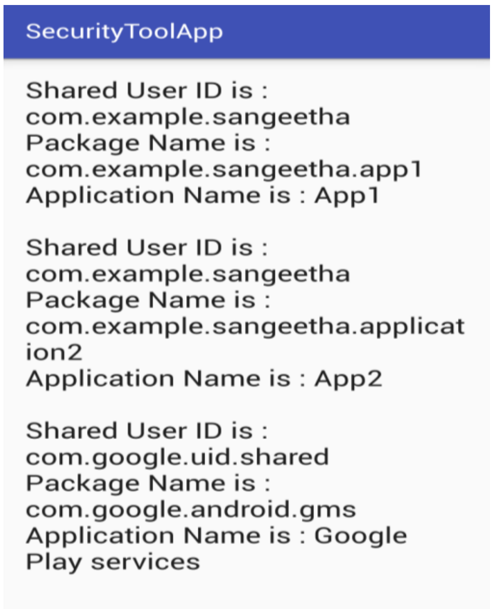
The "Permission Misuse Identifier tool discussed in this work, dynamically shows the complete list of applications which are misusing app permissions. It takes the list of packages installed on the smartphone as input and produces the list of apps for which setting of SharedUserID may result in misuse of the app permissions. The users may not be aware of the apps, which are misusing the granted permissions. It is also difficult for the end users to check the manifest file to find out whether the SharedUserID is set for an application or not. To solve these issues, a security tool viz. Permission Misuse Identifier is developed. All the information about the app is available in the Manifest file. It is possible to get the details of every application installed on the smartphone, by executing the appropriate program code. In Android, every application has its unique ID indicating the package name. It is not at all possible to have two applications with same package name in an Android smartphone. PackageManager and PackageInfo classes are used to get the information related to all the application packages installed on the user device and information about the particular package. The Permission Misuse Identifier does not require any dangerous permissions to perform these tasks.


SecurityToolApp

This application provides list applications for which shared User ID is set. The applications for which shared User ID set may misuse the Permissions granted to each other.

Figure 6: Main page of SecurityToolApp Permission Misuse Identifier

It provides complete details such as SharedUserID value, package name and application name about each application for which SharedUserID is set. These applications may misuse the permissions granted to each other. Figure 6 and Figure 7 show the implementation of Permission Misuse Identifier. The tool has options to view list of installed applications and to view list of applications which are misusing the app permissions by exploiting SharedUserID. Figure 7 shows complete details of the apps which are misusing app permissions.


SecurityToolApp

Shared User ID is :
com.example.sangeetha
Package Name is :
com.example.sangeetha.app1
Application Name is : App1

Shared User ID is :
com.example.sangeetha
Package Name is :
com.example.sangeetha.application2
Application Name is : App2

Shared User ID is :
com.google.uid.shared
Package Name is :
com.google.android.gms
Application Name is : Google Play services

Figure 7: View SharedUserID Permission Misusing Apps

Android is a widely used mobile operating system. Ensuring the unauthorized access to resources of a smartphone, preventing permission misuse and avoiding over claiming of app permissions are all the more relevant for protecting Android applications. This paper discusses a security tool that dynamically identifies the list of apps that attempt to over claim and misuse permissions.

ACKNOWLEDGEMENTS

This research is conducted at **Christ University**, Bengaluru, Karnataka, India. We would like to thank **Dr. Surendra Umesh Kulkarni**, Dean of Sciences for his support for the research work. We would also like to thank **Prof. Joy Paulose**, HOD, Department of Computer Science for his consistent guidance, support and encouragement throughout the research work. We would also like to thank **Dr. Rohini V**, Coordinator of MCA for her support for the research work.

REFERENCES

[1] "Google Play Store: number of apps 2009-2017 | Statistic," Statista. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[2] "Smartphone users worldwide 2014-2020 | statistic," Statista, 2016. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide.

[3] "Latest Gartner data shows iOS vs Android battle shaping up much like Mac vs Windows," 9to5Mac, 18-Aug-2016. [Online]. Available: https://9to5mac.com/2016/08/18/android-ios-smartphone-market-share/.

[4] Cimpanu, C. (2017). Google Play Store Sees Sudden Surge of Malicious Apps. [online] BleepingComputer. Available at: https://www.bleepingcomputer.com/news/security/google-play-store-sees-sudden-surge-of-malicious-apps/.

[5] "Flashlight - Android Apps on Google Play," Google. [Online]. Available: https://play.google.com/store/apps/details?id=com.happyhollow.flash.torchlight&rdid=com.happyhollow.flash.torchlight.

[6] Android Developers. [Online]. Available: https://developer.android.com/guide/topics/manifest/manifest-element.html.

[7] D. Barrera, "Understanding and Improving App Installation Security Mechanisms through Empirical Analysis of Android," ACM, Oct. 2012.

[8] Sbirlea, D. et al. "Automatic Detection Of Inter-Application Permission Leaks In Android Applications", Technical Report TR13-02, Department of Computer Science, Rice University, 2013.

[9] M. S. Ahmad, N. E. Musa, R. Nadarajah, R. Hassan, and N. E. Othman, "Comparison between android and iOS operating system in terms of security," 2013 8th International Conference on Information Technology in Asia (CITA), Jul. 2013.

[10] A. Kaur and D. Upadhyay, "PeMo: Modifying application's permissions and preventing information stealing on smartphones," 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence), Sep. 2014.

[11] Z. Fang, W. Han, and Y. Li, "Permission based Android security: Issues and countermeasures," Computers & Security, vol. 43, pp. 205–218, Jun. 2014.

[12] Ratazzi, Paul et al. "A Systematic Security Evaluation Of Android'S Multi-User Framework". arXiv preprint arXiv:1410.7752 (2014).

[13] Khandelwal, Ankita, and A K Mohapatra. "An Insight Into The Security Issues And Their Solutions For

Android Phones - IEEE Xplore Document". Ieeexplore.ieee.org. N.p., 2015.

[14] Faruki, Parvez et al. "Android Security: A Survey Of Issues, Malware Penetration, And Defenses". IEEE Communications Surveys & Tutorials 17.2 (2015): 998-1022.

[15] B. Rashidi and C. Fung, "A Survey of Android Security Threats and Defenses," Journal of Wireless Mobile Networks, vol. 6, no. 3, pp. 3–35, 2015.

[16] J.-K. Park and S.-Y. Choi, "Studying security weaknesses of Android system," International Journal of Security and Its Applications, vol. 9, no. 3, pp. 7–12, Mar. 2015.

[17] "Google White Paper on Android Security," Google. [Online]. Available: https://www.google.co.in /url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad =rja&uact=8&ved=0ahUKEwjKvpjAqrrTAhUHPI8K HduFChwQFggtMAA&url=https%3A%2F%2Fenterpr ise.google.com%2Fandroid%2Fstatic%2Ffiles%2Fandr oid-for-work-security-white-paper.pdf&usg=AFQjCNFzieFmeDft4YS9PbQ3SvIAa GnIQg&sig2=B--3tuoO6XDXNslOmBI1vQ.

[18] S. Karthick, and Sumitra Binu. "Android Security Issues And Solutions". IEEE - International Conference on Innovative Mechanisms for Industry Applications, Feb. 2017.

[19] "Normal Permissions,". [Online]. Available: https://developer.android.com/guide/topics/security/normal-permissions.html.

[20] "Dangerous Permissions,". [Online]. Available: https://developer.android.

[21] V. Savov, "Only 7.5 percent of Android phones are running marshmallow," The Verge, 2016. [Online]. Available: http://www.theverge.com/circuitbreaker/2016/5/4/11589630/android-6-marshmallow-os-distribution-statistics.

[22] A. Dobie, "Nougat rises to 4.9% share in March Android platform numbers," Android Central, 07-Apr-2017. [Online]. Available: http://www.androidcentral.com/nougat-rises-49-share-march-android-platform-numbers.

[23] "Dashboards," Android Developers. [Online]. Available: https://developer.android.com/about/dashboards/index.html.

[24] "PackageManager," PackageManager | Android Developers. [Online]. Available: https://developer.android.com/reference/android/content/pm/PackageManager.html.

[25] "Android Architecture," www.tutorialspoint.com. [Online]. Available: https://www.tutorialspoint.com/android/android_architecture.htm.

[26] K. Parmar, "In Depth: Android Package Manager and Package Installer - DZone Mobile," dzone.com, 19-Sep-2015. [Online]. Available: https://dzone.com/articles/depth-android-package-manager.