Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

Article

Implementation of Permission Management Method for Before and After Applications the Update in Android-based IoT Platform Environment

In-Kyu Park [1] and Jin Kwark [2,*]

[1] Department of Computer Engineering, Ajou University, Suwon 16499, Korea; ikpark.isaa@gmail.com
[2] Department of Cyber Security, Ajou University, Suwon 16499, Korea
* Correspondence: security@ajou.ac.kr

Abstract: The Android-based IoT platform just like the existing Android provides an environment that makes it easy to utilize Google's infrastructure services including development tools and APIs through which it helps to control the sensors of IoT devices. Applications running on the Android-based IoT platform are often UI free and are used without the user's consent to registered permissions. It is difficult to respond to the misuse of permissions as well as to check them when they are registered indiscriminately while updating applications. This paper analyzes the versions of before and after an application the update running on the Android-based IoT platform and the collected permission lists. It aims to identify the same permissions before and after the update, and deleted and newly added permissions after the update were identified, and thereby respond to security threats that can arise from the permissions that is not needed for IoT devices to perform certain functions.

Keywords: Android permissions; Android IoT platform; Android update; Android application

1. Introduction

The Android-based IoT platform was first unveiled to the public as the developer preview version on December 13, 2016. The Android-based IoT platform provides the technology to develop applications that run on IoT devices based on the Android operating system. It makes it easy to develop applications while leveraging existing Android development tools, Android APIs and Google infrastructure services.

Applications that run on the Android-based IoT platform have much in common with those that run on existing Android-based Smartphone. Both applications running on the IoT device and smartphone register permissions to provide users with certain functions. If an application is used differently from its original purpose or asks additional permissions rather than using given permissions to provide certain functions for the user, it can perform malicious activities such as collecting excessive information or leaking personal information [1]. For example, if an IoT device that provides temperature and humidity registered permissions such as location information, camera, package installation and deletion, etc., it would perform functions different from the original purpose through the newly registered permissions.

This paper collects permission lists for the versions of an application running on the Android-based IoT platform before and after the update. It aims to respond to future security threats by identifying the same, deleted, and added permission information compared to the update based on the collected permission lists.

The structure of this paper is as follows. Section 2 discusses the Android-based IoT platform, the AndroidManifest.xml file, and the Android permission protection level. Section 3 performs permission analysis on the application to identify permission differences before and after the update. Finally, section 4 concludes this study.

© 2017 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

2. Related Works

2.1. Android-based IoT platform

The Android-based IoT platform named "Android-Things" was first unveiled by Google. It is the first platform dedicated to IoT devices. "Android-Things" is an upgraded version of the existing Google's Internet platform, Brillo. Unlike the C/C++ language used in Brilo, it enables Android developers to easily develop IoT products [2, 3] by using existing Android development tools such as Android Studio, JAVA language, Android SDK in the same way. In addition, the hardware of "Android-Things" includes Intel Edison, Pico NXP, Raspberry Pi 3, etc. Each hardware is equipped with SOC (System On Chip), RAM, and wireless communication devices. "Android-Things" basically supports various sample code examples such as Doorbell and Bluetooth Audio, making it easier for developers to access.

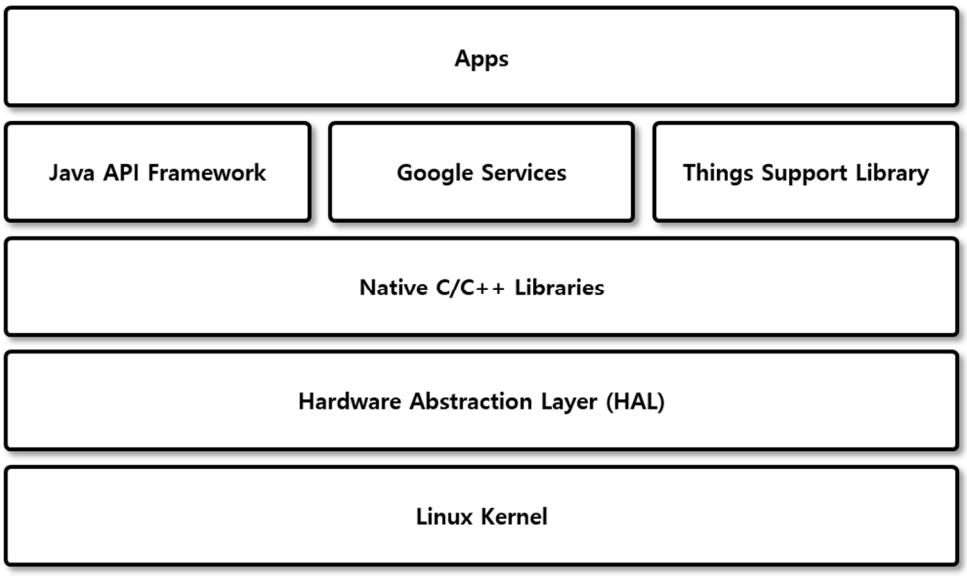

Figure 1. Android based IoT platform

2.2. AndroidManifes.xml file

The AndroidManifest.xml file of an application used in the Android-based IoT platform environment has a similar structure to that in the conventional Android smartphone. The AndroidManifest.xml file contains information on the application including <activity>, <Intent-filter>, and <uses-permission> [4-6]. This paper analyzes permissions of the versions of before and after application the update by analyzing the AnadroidManifest.xml file. The following Table 1 shows the structure of the AndroidManifest.xml file for a sample application provided for the use in the Android-based IoT platform environment.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

Table 1. The AndroidManifest.xml file structure of an application in the Android based IoT platform environment

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.androidthings.bluetooth.audio">
    <uses-permission android:name="android.permission.BLUETOOTH" /
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.BLUETOOTH_PRIVILEGED"
    />
    <application
        android:allowBackup="true"
        android:icon="@android:drawable/sym_def_app_icon"
        android:label="@string/app_name">
        <uses-library android:name="com.google.android.things"/>
        <activity android:name=".A2DPSinkActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.IOT_LAUNCHER"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

2.3. *Android permission protection level*

Android applications must register their permissions in the AndroidManifest.xml file to gain access to the information on the Android device and obtain the user's consent to the use of permissions. The permission protection level for registered permissions can be specified by the developer. It is classified into Normal, Dangerous, Signature, and SignatureOrSystem. Table 2 below lists the four permission protection levels and its definition [4, 7, 8].

Table 2. Define type of permission protection level

Permission Protection Level	Meaning
Normal	- a low risk permission granted to an application with less security threats. - granted to an application without notifying the user or asking for the user's consent at installation time
Dangerous	- a high risk permission granted to an appication with a higher risk than Normal - unlike Normal, notify the user of a requesting permission at installation time and check the user's consent

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

Signature	- a permission granted to an application that is signed with the same certificate as the platform - granted without notifying the user
SignatureOrSystem	- a permission granted to an application that are in the Android system image or that is signed with the same certificate as the platform - typically used when multiple manufacturers need to share specific features when building applications together - like Signature, granted without notifying the user

3. Implementation of permission management method for before and after applications the update

3.1. Analysis flowchart for change of permissions before and after the update

The first step in the analysis sequence to compare permissions before and after the application update is to find the AndroidManifest.xml file and then perform an analysis on the file. The persmissions used by before and after an application the update are first identified based on the analyzed information. After this, the same, deleted, and added permissions in the versions of before and after application the update are checked through the identified information. Figure 2 below shows an analysis flow chart to analyze the permission differences before and after the update.

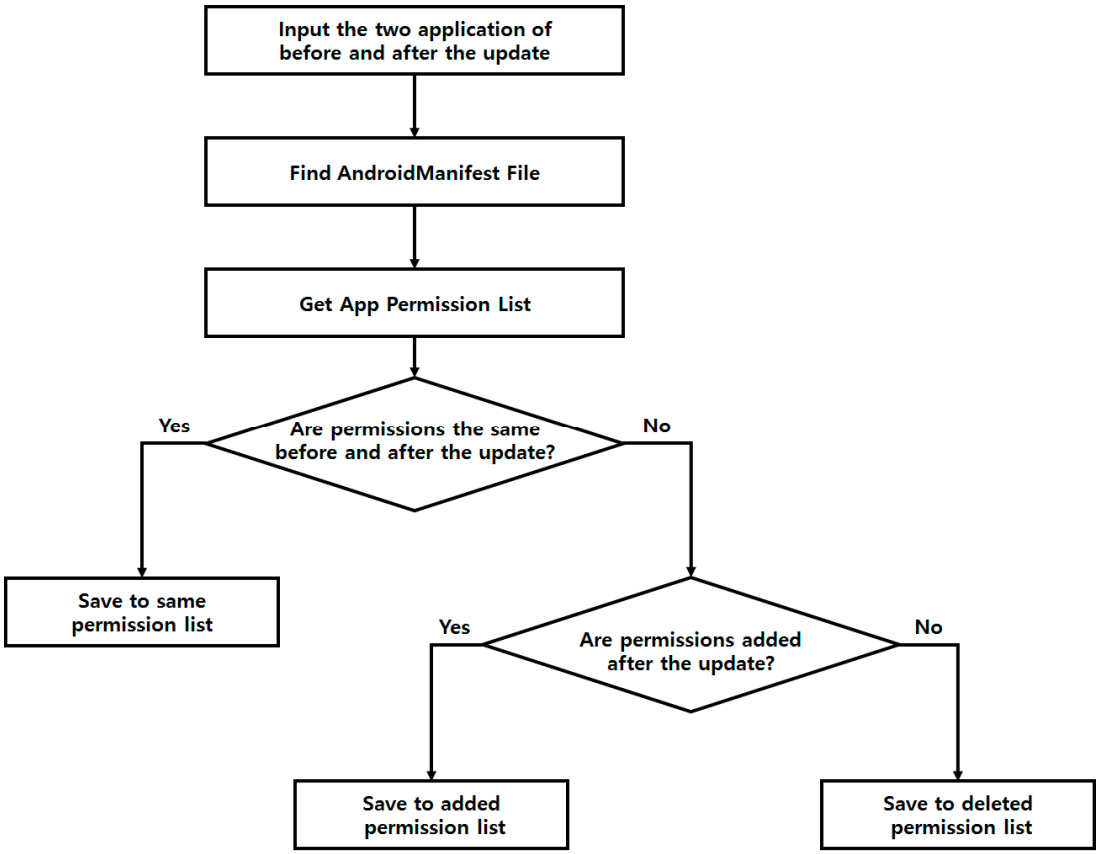

Figure 2. Analysis flowchart for change of permissions before and after the update

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

Permission analysis for before and after application the update consists of four steps. The detailed analysis process is as follows.

1. Input of the application information before and after the update
   Input two versions of the application to analyze before and after application the update.
2. Search of the AndroidManifest.xml
   Search for the AndroidManifest.xml file to analyze permissions for both versions of the application. During this process, find each AndroidManifest.xml file for before and after application the update
3. Check the permissions used by before and after application the update
   Analyze the AndroidManifest.xml file found in step 2 to check and list permissions used in before and after application the update
4. Identify permission differences for before and after application the update
   Based on the analyzed information above, the same, deleted, and added permissions during the update process are identified. Based on the permission information identified through the analysis, respond to security threats such as indiscriminate data collection and data leakage by recognizing them in advance that may occur in Android-based IoT devices.

3.2. Source code for permission analysis before and after the update

Python version 3.5.3 is used to analyze the permissions of the application used in the Android-based IoT platform environment. Search the AndroidManifest.xml file inside the application based on both of the application input information. Analyze both AndroidManifest.xml files to identify the same and changed permissions before and after the update. The content of the source code is explained as follows.

- Line 01~02:
  The variable pwd1 and pwd2 contain the top-level directory name for analyzing both versions of the application.
- Line 04~12:
  Find the AndroidManifest.xml file in the application using the variable pwd1 and pwd2. Generally, the AndroidManifest.xml file is in "/app/src/main/" but sometimes it is not. Therefore, do not always search the same path but search all paths inside the application to find the AndroidManifest.xml file. If the AndroidManifest.xml file is found, open the AndroidManifest.xml file in read mode using the update_before and update_after variable to analyze the information in the AndroidManifest.xml file.
- Line 14~19:
  Check the phrase "android.permission" by reading a file line by line. In case permissions are provided by Android, the phrase basically starts with "android.permission". When this phrase is found, include the permission before and after the update in the update_before and update_after list respectively and identify the deleted or added permissions based on the list information. The identified permissions are kept sorted for the ease of use later.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

Table 3. Source code for analyzing the used permissions

```
pwd1 = "./" + var1
pwd2 = "./" + var2

for path, dirs, files in os.walk(pwd1):
    for file in files:
        if (os.path.join(path, file).find("AndroidManifest.xml") > -1):
            update_before = open(os.path.join(path, file), 'r')
for path, dirs, files in os.walk(pwd2):
    for file in files:
        if (os.path.join(path, file).find("AndroidManifest.xml") > -1):
            update_after = open(os.path.join(path, file), 'r')

for count in update_before:
    if count.find("android.permission") > -1:
        before_list.append(count[count.find("\""):count.find("/>")])

for count in update_after:
    if count.find("android.permission") > -1:
        after_list.append(count[count.find("\""):count.find("/>")])

after_list.sort()
before_list.sort()
```

3.3. Analysis results

When the analysis of two versions of the application is completed, the same permissions before and after the update are first printed out on screen. Next, the deleted and newly added permissions after the update are printed out in order. Figure 3 shows the results of analyzing the permissions of before and after application the update. Permissions from [1] through [3] in Figure 3 show the same permissions that exist in both versions of before and after application the update. [4] through [6] indicate permissions that existed in the version of before application the update but were deleted after the update. [7] - [14] shows newly added permissions that did not exist in the version of before application the update but were added in the update process. The permissions that have been deleted or added after the update can be identified through the analysis.

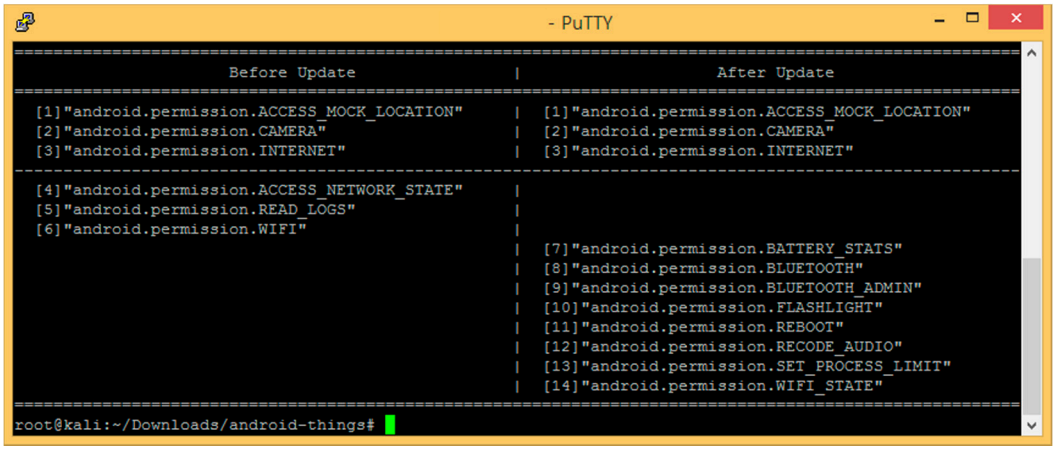

Figure 3. Analysis result of the permissions of the before and after application the update

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

3.4. Security threats

IoT devices can carry out malicious activities such as collecting personal information indiscriminately or leaking personal information when permissions not related to performing certain functions are added during the update process. To prevent IoT devices from performing such malicious activities, there is a need to analyze threats that may arise from permissions to be added during the application update. Information on permissions that exist in many applications that perform malicious activities has been continuously analyzed through many researches. Table 4 below shows the list of permissions that exist in the malicious applications that have been previously studied [4, 5, 9, 10]. It is sorted in the order most used of permission in the malicious application. Restrictions on the use of permissions in the process of analyzing security threats should be considered since there may be restrictions on using permissions according to IoT devices. Based on the previously researched permission information and the results analyzed in section 3.3, it is necessary to respond to security threats in advance by analyzing them that may occur due to added permissions while updating an application. For example, if an IoT device that provides temperature or humidity asks permissions to control the location information or the device, it is necessary to respond to security threats that can arise from this.

Table 4. Commonly used permission ranking in malicious applications

Ranking	Permission
1	INTERNET
2	READ_PHONE_STATE
3	ACCESS_NETWORK_STATE
4	WRITE_EXTERNAL_STORAGE
5	SEND_SMS
6	ACCESS_WIFI_STATE
7	RECEIVE_BOOT_COMPLETED
8	RECEIVE_SMS
9	READ_SMS
10	WAKE_LOCK

4. Results

When an application is updated in the Android-based IoT platform environment, it does not require the user's consent to permissions to be added due to the nature of most IoT devices unlike Android smartphone, which might lead to various security threats. In addition, security threats on Android smartphone can occur in applications in the Android-based IoT platform because it, in similar way to the existing Android, provides certain functions and accesses the device information through permissions. This paper comparatively analyzed permissions before and after the application update by examining the AndroidManifest.xml file in the application when it was updated in the Android-based IoT platform environment. The analysis results show that the same permissions before and after the update, deleted and newly added permissions after the update were identified. We should be able to respond to security threats that may arise after the application update through the information on permissions that are identified and exist in many malicious applications that have previously been studied.

In the future, we will build a real-time automatic permission analysis service when an application is updated in the Android-based IoT platform environment by carrying out research on a real-time permission change monitoring system based on the permission management method implemented in this paper.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 May 2017 doi:10.20944/preprints201705.0075.v1

Acknowledgments: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. NRF-2014R1A2A1A11050818).

Author Contributions: InKyu Park is implemented the program and generated the results of the work. Also InKyu Park is coordinated the writing of the manuscript, and contributed to the text. Jin Kwak is conceived and supervised the work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kimberly Tam, Ali Feizollah, Nor Badrul Anuar, and Rosli Salleh, Lorenzo Cavallaro. The Evolution of Android Malware and Android Analysis Techniques. Journal of ACM Computing Surveys **2017.**
2. Yonghong Wu, Jianchao Luo and Lei Luo. Porting mobile web application engine to the Android platform. IEEE International Conference Computer and Information Technology **2010.**
3. Sung Wook Moon, Young Jin Kim, Ho Jun Myeong, Chang Soo Kim, Nam Ju Cha, and Dong Hwan Kim. Implementation of Smartphone Environment Remote Control and Monitoring System for Android Operating System-based Robot Platform. International Conference on Ubiquitous Robots and Ambient Intelligence **2011.**
4. Xuetao Wei, Lorenzo Gomez, Lulian Neamtiu, Michalis Faloutsos. Permission Evolution in the Android Ecosystem. Proceedings of the 28th Annual Computer Security Applications Conference **2012,** ACSAC '12, pp. 31-40.
5. Xiang Li, Jianyi Liu, Yanyu Huo, Ru Zhang, Yuangang Yao. An Android malware detection method based on androidmanifest file. Cloud Computing and Intelligence Systems (CCIS) **2016.**
6. Jignesh Joshi, Chandresh Parekh. Android Smartphone Vulnerabilites : A Survey. Advances in Computing, Communication, & Automation (ICACCA) (Spring) **2016.**
7. Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, Phillipa Gill and David Lie.; Short Paper: A Look at SmartPhone Permission Models. In Proceedings of the 1st ACM Workshop on Security and Privacy in SmartPhones and Mobile Devices **2011,** SPSM '11, pp. 63-58.
8. Mengyu Qiao, Andrew H. Sung and Qingzhong Liu. Merging Permission and API Features for Android Malware Detection. IIAI International Congress on Advanced Applied Informatics. **2016.**
9. Bhaskar Sarma, Ninghui Li, Chris Gates, Rahul Potharaju, Cristina Nita-Rotaru. Android Permissions: A Perspective Combining Risks and Benefits. Proceedings of the 17th ACM symposium on Access Control Models and Technologies **2012,** SACMAT '12, pp. 13-22.
10. Panagiotis Andriotis, Martina Angela Sasse, Gianluca Stringhini. Permissions Snapshots: Assessing Users' Adaptation to the Android Runtime Permission Model. IEEE International Workshop on Information Forensics and Security. **2016.**