DETECTION OF SENSORS IN ANDROID APPLICATIONS

M. Prasanna Lakshmi[1], V. Esther Jyothi[2], M. Venkata Rao[3]
[1] Department of Computer Applications, V.R.Siddhartha Engineering College, Vijayawada, India
[2]Department of Computer Applications, V.R.Siddhartha Engineering College, Vijayawada, India
[3]Department of Mathematics, V.R.Siddhartha Engineering College, Vijayawada India

Abstract: *Every mobile device is embedded with some sensors inside, while manufacturing. These sensors ease the working of the applications that are downloaded on to the device. There are basically two types of sensors- physical and composite. Physical sensors are the actual sensors that are embedded in the device such as accelerometer gyroscope and magnetometer. It is not compulsory that every android device should consist of all the three. It depends on the manufacturer and the purpose of the device. Composite sensors are derived by the combination of various physical sensors, such as rotation vector sensor which uses accelerometer, magnetometer and gyroscope. Permissions are set by the android operating system (AOS) to access these sensors. So, for an application to use a sensor, like GPS, it has to request the user to grant permission. Zero permission sensors (ZPS) are those that do not require any permission from the user, and the application can access them without any permission. Such sensors are usually the hardware sensors like accelerometer, gyroscope and magnetometer.*

Keywords— *GPS, AOS, ZPS, gyroscope, magnetometer, accelerometer.*

I. INTRODUCTION

We propose an approach for Android development and android app development played a great role in the success of AOS. The software is allowed to be modified and distributed. The sensor framework in android provides several classes and interfaces that help you perform a wide variety of sensor-related tasks. The sensor framework can be used to determine which sensors are available

on the device; determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution; acquire raw sensor data and define the minimum rate at which you acquire sensor data; and, register and unregister sensor event listeners that monitor sensor changes. An event is created when there is a change in state of sensor. For example, accelerometer provides the values of all x, y, z co-ordinates and if there is a change in those values, an event is created. Every third-party application registers to the AOS for specific sensor events through event listeners. So, when an event is created, the AOS broadcasts the events to all the event listeners that are registered. The applications grab the sensor event information through their listeners and use it accordingly. Android sensor stack [1] describes the components and control flows from applications to sensors and vice versa.

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Due to the open nature of Android, a number of third-party application marketplaces also exist for Android, Android device owners are not given root access to the operating system and sensitive partitions such as systems are read only. However, root access can be obtained by exploiting security flaws in Android, but also by malicious parties to install viruses and malware.

Sometimes an android device can also be exploited by tracking the behaviour of sensors in it. Android devices incorporate many optional hardware components, including still or video cameras, GPS, orientation sensors, dedicated gaming controls, accelerometers, gyroscopes, barometers, magnetometers, proximity sensors, pressure sensors, thermometers, and touch screens. In this project, we analyze ZPSs and how many times they were called in an application.
There has been a lot of research being done on the use of zero permission sensors for data inference. Jun Han, et.al.,[2] demonstrated that accelerometers can be used to locate a device owner to within a 200 meter radius of the true location. Their results are comparable to the typical accuracy for handheld global positioning systems. Yan Michalevsky, et.al., [3] showed that the MEMS gyroscopes found on modern smart phones are sufficiently sensitive to measure acoustic signals in the vicinity of the phone. Yan Michalevsky, et.al., [4] showed that by simply reading the phone's aggregate power consumption over a period of a few minutes an application can learn information about the user's location.

Section I contains the introduction of usage of sensors in android applications. , Section II explains the related work of sensors, Section III explains Frame work of Detection of sensors in Android App, Section VI concludes research work.

II. RELATED WORK

To the best of our knowledge, no existing solution is adequate for what we want to achieve. In what follows we briefly review the relevant sensors.

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in the ambient environment near a device. For example, a game might track readings from a device's gravity sensor to infer complex user gestures and motions, such as tilt, shake, rotation, or swing. Likewise, a weather application might use a device's temperature sensor and humidity sensor to calculate and report the dewpoint, or a travel application might use the geomagnetic field sensor and accelerometer to report a compass bearing.

The Android platform supports three broad categories of sensors:

Motion sensors

These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

Environmental sensors

These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

Position sensors

These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

You can access sensors available on the device and acquire raw sensor data by using the Android sensor framework. The sensor framework provides several classes and interfaces that help you perform a wide variety of sensor-related tasks. For example, you can use the sensor framework to do the following:

Determine which sensors are available on a device.

Determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution.

Acquire raw sensor data and define the minimum rate at which you acquire sensor data.

Register and unregister sensor event listeners that monitor sensor changes.

Motion Sensors

The Android platform provides several sensors that let you monitor the motion of a device.

The sensors' possible architectures vary by sensor type:

The gravity, linear acceleration, rotation vector, significant motion, step counter, and step detector sensors are either hardware-based or software-based.

The accelerometer and gyroscope sensors are always hardware-based.

Most Android-powered devices have an accelerometer, and many now include a gyroscope. The availability of the software-based sensors is more variable because they often rely on one or more hardware sensors to derive their data. Depending on the device, these software-based sensors can derive their data either from the accelerometer and magnetometer or from the gyroscope.

Motion sensors are useful for monitoring device movement, such as tilt, shake, rotation, or swing. The movement is usually a reflection of direct user input (for example, a user steering a car in a game or a user controlling a ball in a game), but it can also be a reflection of the physical environment in which the device is sitting (for example, moving with you while you drive your car). Motion sensors by themselves are not typically used to monitor device position, but they can be used with other sensors, such as the geomagnetic field sensor, to determine a device's position relative to the world's frame of reference.
All of the motion sensors return multi-dimensional arrays of sensor values for each SensorEvent. For example, during a single sensor event the accelerometer returns acceleration force data for the three coordinate axes, and the gyroscope returns rate of

rotation data for the three coordinate axes. These data values are returned in a float array (values) along with other SensorEvent parameters. The accelerometer and gyroscope motion sensors that are available on the Android platform.

Sensor: TYPE_ACCELEROMETER

Sensor event data & Description:

SensorEvent.values [0]: Acceleration force along the x axis (including gravity).

SensorEvent.values [1]: Acceleration force along the y axis (including gravity)

SensorEvent.values [2]: Acceleration force along the z axis (including gravity)

Units of measure: m/s^2

Sensor: TYPE_GYROSCOPE

Sensor event data & Description:

SensorEvent.values [0]: Rate of rotation around the x axis.
SensorEvent.values [1]: Rate of rotation around the y axis.
SensorEvent.values [2]: Rate of rotation around the z axis.

Units of measure: rad/s

III FRAME WORK

The frame work consists of 3 stages:
1) Downloading and decompiling the apks,
2& 3) Feed the decompiled apk to a web page and iterate each .java file in the source code using the developed python code.
Stage 1: Downloading and decompiling APKs: An apk data set of 60 selected apks is used in this project. Each apk is decompiled using a tool called jade-gui [5], which decompiles all the source code and also gives us an option to save it in our required directory. The final source code consists of all the .java files that are used to build the respective application.
Stage 2: Python code: A python code is developed to analyze the source code of every application. With just one command, it loads the respective application's source code, and gives the counter values of specific key words that occurred in the code.
Key words include Sensor Manager, Accelerometer, Gyroscope and Magnetometer etc. It gives the output of number of times these keywords occurred. This helps in knowing the type of sensors used, number of times they are called, number of times the application registered & unregistered for event listeners.
Stage 3: Python code is incorporated with in an HTML web page, which takes in the input and gives the output. The output would be the data related to number of times sensor is called.

3.1 Algorithm for Python code

Below is the algorithm for the code:
1. Takes the path of the directory and prompts the user for the folder.
2. Searches for .java files in all the sub-folders and iterates through each of them to convert them into .txt files.
3. Gets the new path of the .txt file, loads the file into a temporary .txt file to search all through it for certain bunch of keywords. (a keywords .txt file) is priory created and loaded in the function.
4. It increments the respective counters of each of the keywords into a list, along with the application name.
5. The list is passed into a function which creates a .csv file out of it.

Frame work Description:

1) **Android OS: Android** is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as smart phones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input.

2) **APK:** An Android **Package** Kit (APK for short) is the **package** file format used by the Android operating system for distribution and installation of mobile apps. Just like Windows (PC) systems use an .exe file for installing software, the APK does the same for Android.

3) **SensorManager:** SensorManager lets you access the device's sensors.

Public abstractclass SensorManager extends Object
java.lang.Object
android.hardware.SensorManager

Third Party Applications: Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Due to the open nature of Android, a number of third-party application marketplaces also exist for Android, either to provide a substitute for devices that are not allowed to ship with Google Play Store, provide applications that cannot be offered on Google Play Store due to policy violations, or for other reasons. Examples of these third-party stores have included the Amazon Appstore, GetJar, and SlideMe. F-Droid, another alternative marketplace, seeks to only provide applications that are distributed under free and open source licenses.

Application Frame Work:

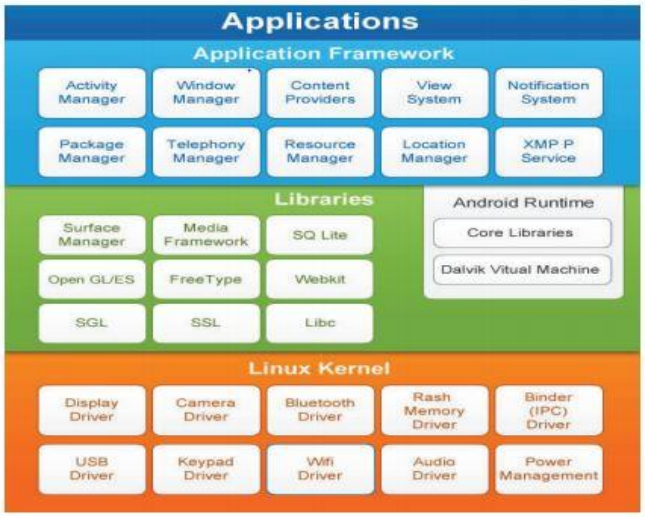

Figure 1: Application Frame Work in Android Application.

Application framework layer provides a much higher level of service applications in the form of Java classes. Developers are allowed to use these services in developing applications.

Applications At the supreme layer are applications. They are developed and implemented exclusively in this layer. Examples of such applications are Contact Manager, Web Browser, Games, Email clients and others.

APK Decompiler:
In computing, reverse engineering is the process of understanding how things work and reusing the information to do something. This is applicable even to Android apps. You might reverse engineer Android apps for many reasons.

1. Read another's code
2. Find vulnerabilities in the code
3. Search for sensitive data hardcoded in the code
4. Malware Analysis
5. Modifying the functionality of an existing application

Decompilation v/s Disassembly

Decompilation is the process of converting software binaries to clear text format in a high-level language in which the source code is written so that developers can read it.

Disassembler, on the other hand, won't convert the binary to a high-level language text. It is only a one-to-one translation of bytes to text and gives us instruction mnemonics, which again can be understood by humans, but it is a little difficult when compared to reading original source code.

APK file extraction

Extraction of the APK file is a process of file decompression from one to multiple readable objects. After the APK file extraction, that is a simple unzip or extract here operation, depending on the compression tool available at the moment (7zip, WinRaR, etc.) many readable objects are already present such as multimedia files used by the application (icons, photos, video files, etc.), although all of the application logic and graphics interface of the application are hidden in the binary archives such as classes. dex and AndroidManifest.xml. For these files to be readable further decompiling is needed.

IV CONCLUSION

This paper concentrates on the sensor functioning, specifically zero permission sensors, and analyses the code statistically to provide the number of times a keyword has occurred. These numbers, combined with the sampling rate to access the sensor data, help in determining the functioning of an application with regard to the zero permission sensors. The python code, which gives the derived values, might help in performing the data flow analysis, specifically for ZPS, in any future applications.

REFERENCES

[1] https://source.android.com/devices/sensors/sensor-stack

[2] "ACComplice: Location Inference using Accelerometers on Smartphones", JunHan, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, Joy Zhang, IEEECOMSNETS, 2012.

[3] "Gyrophone: Recognizing speech from gyroscope signals", Yan Michalevsky, Dan Boneh, and Gabi Nakibly, USENIX Security, 2014

[4] "PowerSpy: Location Tracking using Mobile Device Power Analysis", Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly, USENIX Security, 2015.

[5] https://github.com/skylot/jadx/releases/tag/v0.6.1