Volume 8, No. 5, May-June 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Android Internal Analysis of APK by Droid_Safe & APK Tool

Heena Rawal
M. Tech.,Cyber Security
,Raksha Shakti University
Ahmedabad, Gujarat, India

Mr. Chandresh Parekh
Assistant Professor IT and Telecommunication
Raksha Shakti University
Ahmedabad, Gujarat, India

Abstract: Mobile devices have developed from simple devices, which are used for a phone call and SMS messages to Smartphone devices that can run third party applications. Nowadays, information leak from mobile device is increased and due to this sensitive information like; bank details, identity information and many more personal information. The whole scenario is from source to sink where when data is received from source in between this process sensitive information is leaked by attacker. The leak of sensitive data from those Applications or Android operating systems are being exploited by the attackers who got the capability of penetrating into the mobile systems without user authorization causing compromise the confidentiality, integrity and availability of the applications and the user. This paper, it gave an update to the work done in the project.
We present DroidSafe and Apk Tool a static information flow analysis tools that reports potential leaks of sensitive information in Android applications. Moreover, this paper focuses on the Android Operating System and aim to detect existing Android apps.This Paper chooses several sensitive information leakages from apps with help of these both tools.

Keywords: Smartphone Security; DroidSAfe, APK Tool, Source to Sink, Sensitive Information.

INTRODUCTION

A number of years ago, Smartphone's and tablet have become more common. They provide services such as social networking, banking, etc. Also, they are prepared with many features like Wi-Fi and GPS, make video calls and many more equipment's. With all these features, there also comes a need for security for the mobile phones. This paper mainly focuses on the Android Operating System.
Android, which is open source operating system, will be more popular. At this time, there are over 50 mobile phone companies are developingSmartphone's with Android operating system. Increasing the number of the Android devices causes anxiety in term of user security. McAfee Labs report illustrates that in the first quarter in 2012 to 2017, there is a raise in mobile malware, and the increase was embattled almost only at the Android platform [1]. Figure 1 shows that there were 10 billion application downloaded by the end of the 2012 to 2017. These quick increases in applications download make Android to be the mainly targets for malware.

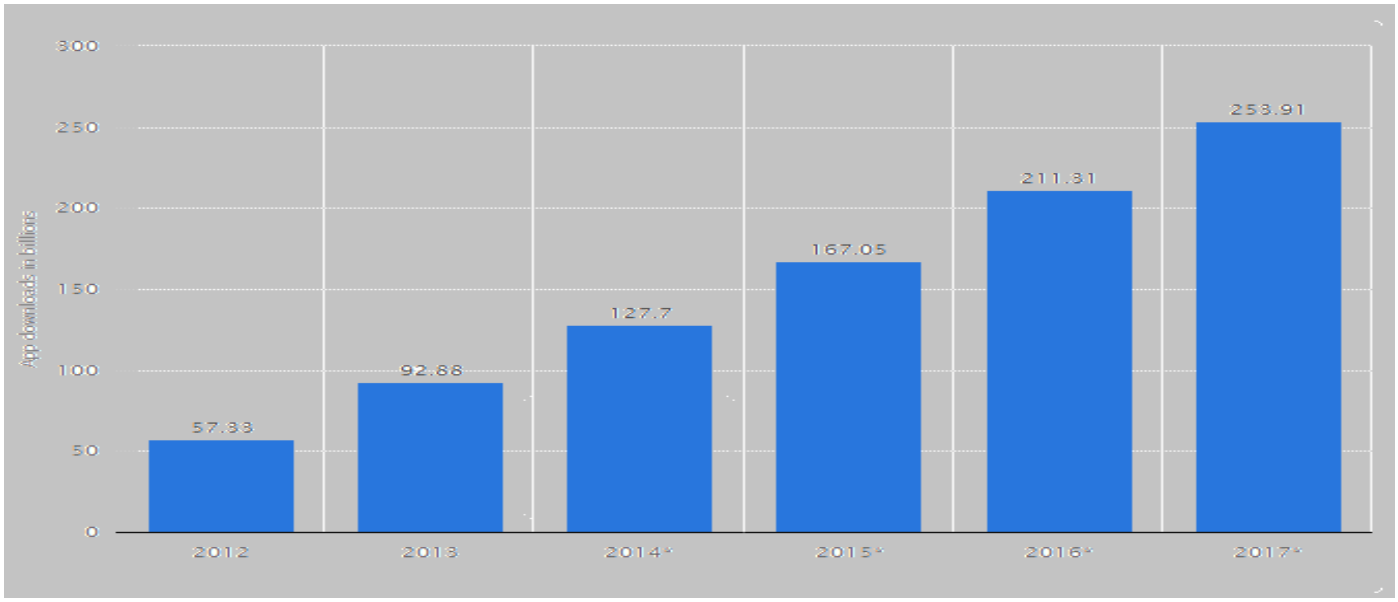

Fig.1. Android Market Growth

In this paper, we are generating two Device IDs APK's and run on two tools (1) APK TOOL: which gives us folder containing SMALI code and (2) we run same Device IDs on Droid safe; it also gives different folders and now by matching strings we extract sensitive data from source to sink. And learning this malware can target the Android phones and how it could be installed and activated in the device by performing a malware analysis using static and

dynamic tools to understand the malware operations and functionalities. To achieve these errands, it is necessary to understand the Android architecture and its security model.

The rest of this testimony provide an explanation of the project and is organized as follows: Section II presents a general idea of Android architecture Section III describe Android security model. After that, Section IV explains Android application Section V Tools for data extraction followed by analysis result in Section VI. Section VII shows the detection results with two Device IDs running on two tools. Section VIII discusses one way for outlook enhancement. Lastly, it winds up the paper in Section VIII.

In Android application sensitive information leaks, as implemented by malicious or misused code, form one of the most well-known security threats to the Android ecosystem. Now android supports a fine-grained information security model in which users grant applications the right to access sensitive information. This model has been less than doing well at eliminating information leaks, in part because many applications need to legally access sensitive information, but only for a specific limited purpose — for example, an application may legally need to access location information, but only with the right to send the information to authorized mapping servers.

Static analysis frameworks attempt to analyze the application before it executes to discover all potential sensitive flows. Standard issues that make problems the constructionof such systems are the challenges of 1) scaling to largeapplications and 2) maintaining precision in the analysis suchthat it does not report too manyflows that do not actuallyexist in the application. One mostly prominent issue withdeveloping static analyses for Android applications is the size,richness, and complexity of the Android API and runtime,which typically comprises multiple millions of lines of codeimplemented in multiple programming languages. Becausesensitiveflows are often generated by complex interactionsbetween the Android application, API, and runtime, any staticanalysis must work with an accurate model of this runtime toproduce acceptably accurate results.

Accuracy is critical for a static analysis seeking to calculate security properties of an application; any inaccuraciesin the execution model provide a motivated attacker with theopportunity to insert maliciousflows that will not be capturedby an analysis. Also, indistinctness in a model could lead toresults that are impractical due to too many fake positives;another target for a motivated attacker. To the best of ourknowledge, the difficulty of obtaining an acceptably accurateand precise Android model has significantly limited the abilityof previous systems to successfully detect the full range ofmalicious information flows in Android applications.

OVERVIEW OF ANDROID ARCHITECTURE

Android is open source platform for mobile improvement developed by Google. The Android architecture as shown Figure2 can be divided into five layers. The first layer from the bottom is the kernel, which is based on the Linux 2.6 kernel. It is used as hardware abstraction layer. The motive of Googleis using Linux is because it provides memory organization, process management, security model, networking, a bunch of core operating system infrastructure that are tough and have been deep-rooted over time. The next level up is a native or central library, which is on paper in C and C++. Further we go to nextlevel that is the Android runtime. The main section in the Android runtime is the Dalvik virtual machine. It was designed particularly for Android to assemble the need of running in an embedded environment where you have inadequate battery, inadequate memory, inadequate CPU. The Dalvik virtual machine runs somewhat called DEX files. Those files are bytes' codes that are outcome of converting at java. classes and .jar files. Those files when are converted to .DEX files become much more well-organizedbytes' code that can run very well on small processors. They make use of memory very effectiveness. The next level up from that are the core libraries. They are written in java encoding language. It contains all of the gathering classes, utilities, I/O, etc. The upper level is the application framework. This is also writing in java programming language. It provides concepts of the original native libraries and Dalvik abilities to application. Each of the Android applications runs on its Dalvik virtual machine. [4]

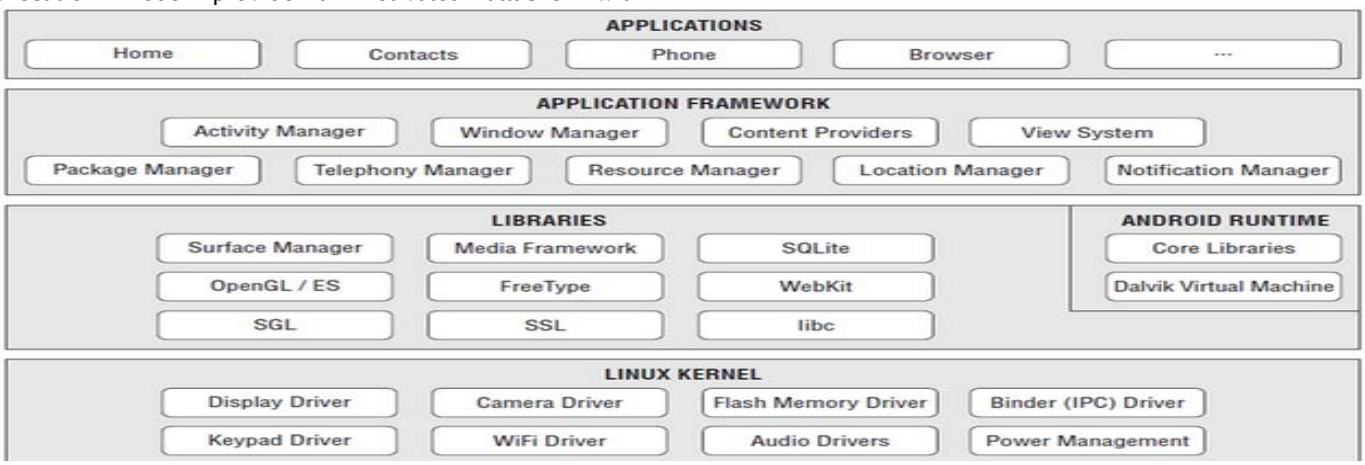

Fig 2. Android Architecture [3]

ANDROID SECURITY MODLE

The entire idea behind mobile platform is the actuality that the user can run a lot and a lot of different applications on the device. The user might be installing and downloading a banking application that can be doing some sensitive data. On furtherpass the user may be installing a game application right after to previous application and running on the same

device. The user obviously does not want the game application to be able to access the sensitive data that banking application is operation on. So, to get this Android platform makes sure that severalapplicationsare isolated from each other. Basically, when the user download and install an application, it will be given a unique UID. In addition, each oneof application will run on separate process on separate virtual machine. Therefore, application cannot read other application private data [4].

As it was mentioned on Section II, Android was built on the top of the Linux, so the Linux file consent are applied. Consent allows the user to protect his/her sensitive data that are stored on the device. Also, it protects access to content provider, which mostly in a database in the device. Consents are requested by an application at install time and they are approved or denied once at the install time which requires the user approval [4].

ANDROID APPLICATION

Android application has an extension file. apk which is put set for Android package. It is mainly an archive file contains all the required files and folder in an application. Each application is divided to four main workingsi.e. Activities, Service, Broadcast Receivers and Content provider [4].

- Activity is essentially just a part of User Interface (UI). So, any visual screens that allow user to see and act together with in an Android application. It can consist of views such as Button View, Text view Table view, etc.
- Intent Receiver which is a way for which an application to register some code that will not be running until it's triggered by some external event. Developer can write some code through XML and register it to be running when a bit happens, e.g. network connectivity is established at a certain time, or when the phone is ring.
- Service is a chore that does not have any user interfaces. It is a module running in the background. For example, when lunching Device IDs application, the first display is an activity. But as soon as selecting a button and that will move to other application, the service keeps running in the background.
- Content Provider is data storage, which allows the applications to share the data with other application.

Each application contains a manifest file named Androidmanifest.xml. This file declares applications components, specifies the application requirements, and contains the permissions. These permissions will be shown to the user, when he would be installing the application. Figure2 is an example of the Androidmanifestfile.xml.

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
- <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="org.cert.echoer">
  - <application android:allowBackup="true" android:icon="@drawable/ic_launcher" android:label="@string/app_name" android:theme="@style/AppTheme">
    - <activity android:label="@string/app_name" android:name="org.cert.echoer.MainActivity">
      - <intent-filter>
          <action android:name="android.intent.action.SEND" />
          <category android:name="android.intent.category.DEFAULT" />
          <data android:mimeType="text/plain" />
        </intent-filter>
      </activity>
    - <activity-alias android:label="MainActivity_VIEW" android:name="MainActivity_Alias" android:targetActivity="org.cert.echoer.MainActivity">
      - <intent-filter>
          <action android:name="android.intent.action.VIEW" />
          <category android:name="android.intent.category.DEFAULT" />
          <data android:scheme="http" />
        </intent-filter>
      </activity-alias>
    </application>
  </manifest>
```

Fig 3 Example of Android Manifest of XML.s

The Android Manifest file also helps a user in determining whether an application is a legitimate one or it is a malicious one. For example, a game application does not need permissions such as SEND_SMSM, READ_CONTACTS. In this case, it should be known that if the application is a legitimate or not.

A. MALWARE INFECTIVITY METHODS

There are quite a few methods that the Android devices could be contaminated with malware. The following are four different methods which malware can be installed on the phone [1]:

1) Repackaging legitimate application

This is one of the most ordinary methods used by the invaders. They may put and download genuine popular application from the market, disassemble it, inserta malicious code and then re-assemble and present the new apps to the official or alternative Android market. Users could be vulnerable by being attract to download and install these contaminated applications. It was found that 86.0% repackaged legitimate application including malicious payloads after analyzing more than 1,200 Android malware samples [1].

2) Exploiting Android's application bug

There could be a virus in the application itself. The attacker may use this susceptibility to compromise the phone and fix the malware on the device.

3) Fake applications

It was also bare that there are fake applications formed to include malware which allows attacker to contact your mobile device. Attackers upload on the market fake applications that seems are lawful to users but they are infected by themselves. For example, Spy eye's fake security tool was found in the market which is a malware.

4) Remote Install

The malware could be installed in the user phone distantly. If the attacker could compromise users' certificate and pass them in the market, then in this case, the malware will be installed into the device without the user permissions. This application will restrain malicious codes that permit attacker to access personal data such as contacts list [1].

TOOLS FOR APPLICATION ANALYSIS

DroidSafe:

DroidSafe is a tool for accurately and precisely analyzing sensitive unambiguous information flows in large, real-world Android applications. DroidSafe footpaths informationflows from sources (Android API calls that injectsensitive information) to sinks (Android API calls that mayleak information). We estimate DroidSafe on twoDevice ID to freely generate a sootOutput folder that has been increased with malicious information flow leaks by three hostile Red Team organizations. The goal of these organizations was to develop information leaks that would either evade detection by static analysis tools or overcome static analysis tools into producing improper results (by, for example, manipulating the tool into reporting an overwhelming number of false positive flows). DroidSafe exactly detects all of the malicious flows in these applications (while reporting a manageable total number of flows). A current stateof-the-art Android information-flow analysis system, Flow- Droid [6] + IccTA [7], in contrast, detects only few malicious flows, and has a larger ratio of total flows reported to true malicious flows reported. DroidSafe fails to report only the implicit flows. Finally, we assess DroidSafe on a suite of two Device IDs Android explicit information-flow developed by us; DroidSafe achieves 100% accuracy and correctness for the matching set, compared to FlowDroid + IccTA's 34.9% accuracy and 79.0% precision. One advantage of working with applications that contain known inserted malicious flows is the ability to characterize the accuracy of our analysis (i.e., measure how many malicious flows DroidSafe was able to detect). As these results illustrate, DroidSafe implements an analysis of unparalleled accuracy and precision. To the finest of our knowledge, DroidSafe provides the first exploitable information-flow analysis for Android applications. [5]

APK TOOL:

A tool for reverse engineering 3rd party, closed, binary Android apps. It can decode resources to nearly original form and rebuild them after making some modifications. It also makes working with an app easier because of the project like file structure and automation of some repetitive tasks like building apk, etc.An android application comes in Android package (.apk) documentation. This. apk file is nothing but a zip package of AndroidManifest.xml, classes.dex and other resources and folders. For extracting these features, we primarily need to reverse engineering of. apk files. This is done using the APK TOOL .The AndroidManifest.xml file contains a lot of features that can be used for static analysis. While running the (.apk) on this tool; it generates SMALI code. And by matching same strings of both Device IDs we steal the sensitive information from source to sink.

While running APK tool it shows like figure 3

Fig.4. Running Status of APK tool

PROPOSED DESIGN

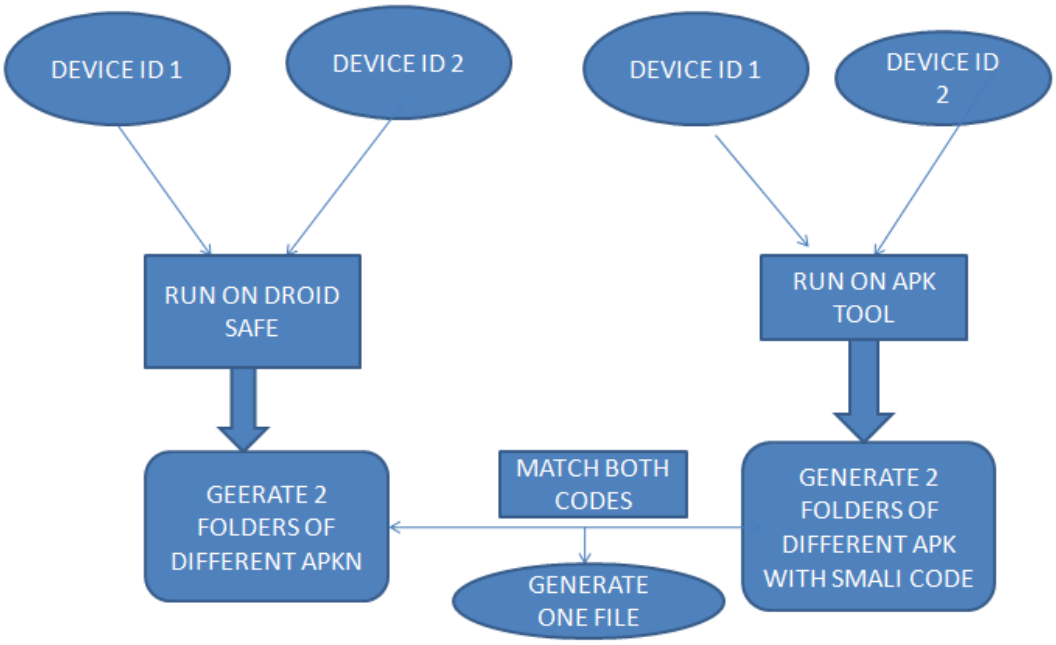

Fig.5. Flow of Working tools

ANALYSIS OF RESULTS

After concluded the test on both Device IDs, we agreed that while sending data from one application to another application (from Source to sink) it becomes very important for every user to check the permissions that any application he/she is downloading really requires access to them or not. One of the most important disadvantages of Android applications is that without agreeing to grant access to all the permissions, an application cannot be installed on the device. For example, the application Game is only supposed to play games and do nothing more. Hence it is understandable that it does not require permission to send messages or receive SMS. Finally, by analysis we can extract data from source to sink while sending SMS or receiving message from one App to another application. Here in Fig. 4 and Fig. 5 both device Ids which are extracting are same.

The Android Manifest file is the single file that has been changed in order to permit the application to access more sensitive data. 83% of the data is misplaced while transfer from source to sink. Therefore, if the developer can have modified Android Manifest file and implement a method or implement an algorithm to detect the application that is installed to the phone, then the risk of those data leakage will be mitigated.

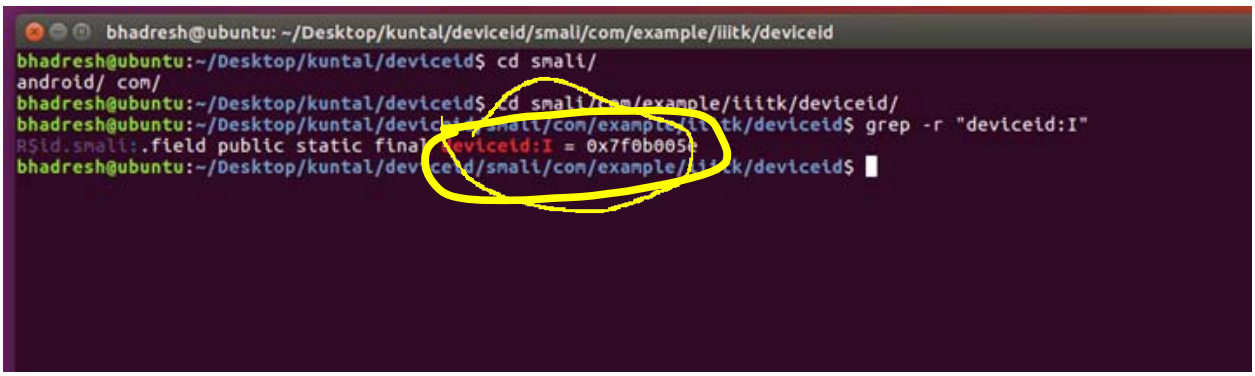

Fig 6. Result of Extracting Device Id

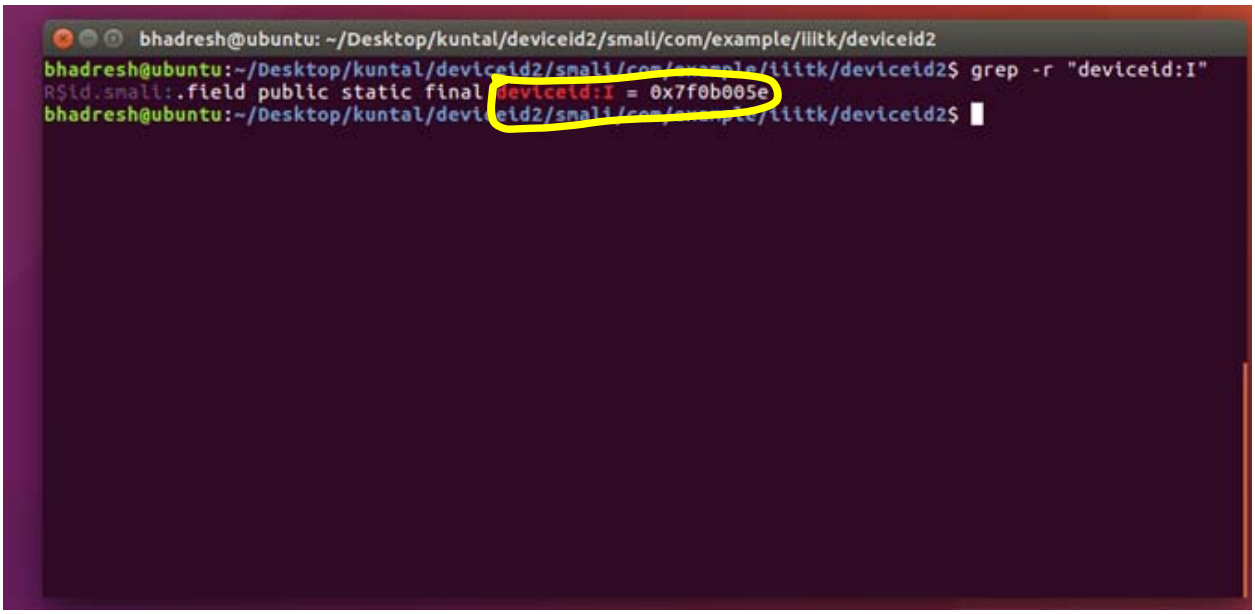

Fig 7. Result of Extracting Device id2

CONCLUSION

In the times of yoreSmartphone users have increased rapidly. There are attackers whose intentionis to target the Smartphone's. The main reason for doing this is that lack of user awareness concerning how their devices can be compromised. Today, Smartphone's like Android are not just used as a manageable telephone. Android devices can right to use the internet, make online bank transmissions, deal with social networks, etc. All these functionalities of a mobile phone appear very attractive for an attacker to gain information of the user and use it to his/her advantage. Therefore, users need to be conscious enough and have full responsibilities to read and understand the permissions requested by the application before agreeing to grant access.

REFERENCES

[1] Yajin Zhou, Xuxian Jiang, "Dissecting Android Malware: Characterization and Evolution," Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012), San Francisco, CA, May2012

[2] Marry van Ullen, Jane Kessler, "Citation App for Mobile Device" The Statistics Portal.[Online]. Available:https://www.statista.com/statistics/241587/number-of-free-mobile-app-downloads-worldwide/. [Accessed: 2-1-2016]

[3] Technology, "Architecture of Android OS" techneology.com, Nov 2011. [Online]. Available:http://www.techneology.com/2011/11/architecture-of- android-os.html.[Accessed:July 25, 2012].

[4] Frank Ableson, "Introduction to Android development the open source appliance platform "ibm.com, 12 May 2009. [Online]. Available: http://www.ibm.com/developerworks/opensource/l library/os-android-devel/. [Accessed: July 25, 2012].

[5] Michael I. Gordon∗, Deokhwan Kim∗, Jeff Perkins∗, Limei Gilham†, Nguyen Nguyen‡, and Martin Rinard∗ "Information-Flow Analysis of Android Applications in DroidSafe", Available: http://people.csail.mit.edu/mgordon/papers/droidsafe-ndss-2015.pdf

[6] S. Arzt et al., "FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps," in PLDI, 2014.

[7] L. Li et al., "I know what leaked in your pocket: uncovering privacy leaks on Android Apps with Static Taint Analysis," CoRR, 2014.