ANDROID MALWARE DETECTION AND FORENSICS BASED ON API CALLS

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Arpitaben Shah

FALL
2016
ANDROID MALWARE DETECTION AND FORENSICS BASED ON API CALLS

A Project

by

Arpitaben Shah

Approved by:

__________________________________, Committee Chair
Dr. Jun Dai

__________________________________, Second Reader
Dr. Xiaoyan Sun

__________________________________
Date
Student: Arpitaben Shah

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

______________________, Graduate Coordinator

Dr. Ying Jin

Department of Computer Science
Abstract

of

ANDROID MALWARE DETECTION AND FORENSICS BASED ON API CALLS

by

Arpitaben Shah

In recent world, mobile devices play an important role towards immense information sharing. As mobile smartphones become more widespread and powerful, they store more personal data and may leak it carelessly or maliciously. Research shows that Android is widely used operating system among many smartphones. The growth of Android users infatuates attackers to target more Android smartphone devices by using malicious software.

To defend against expansion of Android malwares, researchers propose many analysis, detection and classification techniques. This paper introduces a dynamic analysis approach to intercept API calls at runtime, extract logs, and analyze them. It helps to understand runtime behavior of installed applications and use of API calls for malicious purpose. By using this method, analysts may get to know if the application is benign or malicious by comparing its actual behavior and expected behavior.
This research will offer essential help to malware researchers to quickly understand the activities and internal workings of unknown applications.

________________________, Committee Chair
Dr. Jun Dai

________________________
Date
ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Jun Dai for his valuable guidance and support to accomplish this project.

In addition, I would also like to thank my second advisor Dr. Xiaoyan Sun for taking the time and effort to assess my project work.

I would also like to thank the Department of Computer Science at California State University for providing me with the opportunity to pursue my Master of Science program.

I would like to thank my family and well-wishers for undefined support and encouragement throughout my Master of Science program.
# TABLE OF CONTENTS

| Acknowledgements | vii |
| List of Tables | xi |
| List of Figures | xii |

**Chapter**

1. INTRODUCTION ................................................................. 1
2. BACKGROUND .................................................................. 3
   2.1 Android System Architecture ........................................... 3
      2.1.1 Linux kernel ......................................................... 3
      2.1.2 Libraries .............................................................. 3
      2.1.3 Android Runtime .................................................... 4
      2.1.4 Application Framework ............................................. 4
      2.1.5 Applications .......................................................... 5
   2.2 Application Program Interface (API) ................................. 5
   2.3 Android application Malware ........................................... 7
      2.3.1 Types of Malware ..................................................... 7
         2.3.1.1 Trojan ............................................................. 7
         2.3.1.2 Spyware ......................................................... 8
      2.3.2 Application malware detection techniques .................... 8
         2.3.2.1 Static Analysis ................................................ 8
         2.3.2.2 Dynamic Analysis .......................................... 8
3. DYNAMIC ANALYSIS OF MALICIOUS APPLICATIONS ............... 10
3.1 Methodology ................................................................. 10

3.2 Design and Implementation .............................................. 12
  3.2.1 Environment Setup .................................................. 12
  3.2.2 Execute Experiments ............................................... 13
  3.2.3 Analysis of the API Call Logs ................................... 14
    3.2.3.1 Generate CSV files ......................................... 15
    3.2.3.2 Graphical Analysis ......................................... 15

3.3 Experiment Results ....................................................... 16
  3.3.1 Dynamic Analysis of Data Peek .................................. 16
  3.3.2 Dynamic Analysis of SpyCam .................................... 21
  3.3.3 Dynamic Analysis of Call Tracker .............................. 24
  3.3.4 Dynamic Analysis of Puzzle2048 ............................... 27
  3.3.5 Dynamic Analysis of QuxiFlashlight ........................... 31

4. CONCLUSION ..................................................................... 35

5. FUTUREWORK .................................................................. 36

Appendix A. Changed code of APK-Instrumenter ....................... 37
Appendix B. Configuration File for API-Instrument ...................... 47
Appendix C. Script to Covert log file into CSV file .................... 48
Appendix D. Script to collect crucial data from log file and save into CSV file ........................................ 51
Appendix E. Script for Crucial Data Graph ................................ 56
Appendix F. Script for Method Count Graph .............................. 62
Appendix G. Script for Method Details Graph ............................ 64
Appendix H. Script to remove blanks from API call log ................ 67
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Major Components of Application Framework</td>
<td>5</td>
</tr>
<tr>
<td>2. Categories of API calls</td>
<td>6</td>
</tr>
<tr>
<td>3. Commands to run experiment</td>
<td>13</td>
</tr>
<tr>
<td>4. API call analysis graphs</td>
<td>16</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Android System Architecture</td>
<td>4</td>
</tr>
<tr>
<td>2. API Analogy</td>
<td>6</td>
</tr>
<tr>
<td>3. Malware Application Detection Flow</td>
<td>11</td>
</tr>
<tr>
<td>4. Sample API call log snippet</td>
<td>15</td>
</tr>
<tr>
<td>5. Example of Method Detail Graph for Data Peek Application</td>
<td>17</td>
</tr>
<tr>
<td>6. Application Flow Graph for Data Peek Application</td>
<td>18</td>
</tr>
<tr>
<td>7. Method Count Graph of Data Peek Application</td>
<td>19</td>
</tr>
<tr>
<td>8. Crucial Data Graph for Data Peek Application</td>
<td>20</td>
</tr>
<tr>
<td>9. Example of Method Detail Graph for SpyCam Application</td>
<td>21</td>
</tr>
<tr>
<td>10. Application Flow Graph for SpyCam Application</td>
<td>22</td>
</tr>
<tr>
<td>11. Method Count Graph for SpyCam Application</td>
<td>23</td>
</tr>
<tr>
<td>12. Crucial Data Graph for SpyCam Application</td>
<td>24</td>
</tr>
<tr>
<td>13. Example Method Detail Graph for Call Tracker application</td>
<td>24</td>
</tr>
<tr>
<td>14. Application Flow Graph for Call Tracker Application</td>
<td>25</td>
</tr>
<tr>
<td>15. Method Count Graph for Call Tracker Application</td>
<td>26</td>
</tr>
<tr>
<td>16. Crucial Data Graph for Call Tracker Application</td>
<td>27</td>
</tr>
<tr>
<td>17. Example of Method Detail Graph for Puzzle2048 Application</td>
<td>28</td>
</tr>
<tr>
<td>18. Application Flow Graph for Puzzle2048 Application</td>
<td>30</td>
</tr>
<tr>
<td>19. Crucial Data Graph for Puzzle2048 Application</td>
<td>31</td>
</tr>
<tr>
<td>20. Example of Method Detail Graph for QuxiFlashlight Application</td>
<td>32</td>
</tr>
</tbody>
</table>
21. Application Flow Graph for QuxiFlashlight Application ........................................ 33
22. Crucial Data Graph for QuxiFlashlight Application ............................................... 34
Chapter 1

INTRODUCTION

Smartphones and tablets make huge transformation in our day-to-day life. They change the way we consume information and communicate. With the growing popularity of smartphones, the focus of modern mobile computing has moved from laptops to mobile devices. According to recent report, there are more than 50 billion android applications downloaded since the release of first Android device [1].

In current era, there are two known operating systems namely iOS and Android by Apple Inc. and Google Inc. respectively. Among two mentioned operating systems, many hardware dealers adopt Android as it is an open source tool. Additionally, Developers can create their individual applications. Android also allows users to download and install applications from various third party application stores. As a result, the candidness of the Android marketplace attracts both malicious and benign developers.

Moreover, Google allows everyone to publish applications on the Android market. The application will be removed from Android market if it is reported as malicious by users. In February 2012, Google announced a service called Bouncer to check application for malware. It checks the application when it is submitted to Google Play Store. However, researchers found that the malware detection rate of Bouncer is very low and malicious applications can easily bypass Google Play Store [1].

According to latest research, nearly two-thirds of people own smartphones. Most of them use smartphones to send or receive texts, take picture, send or receive emails, do online banking, play games, play music, record or watch videos, and use social networking applications to post
photos or videos. People also use smartphones to store personal and confidential information. These massive use of smartphones motivates malicious developers to create malwares and steal personal information.

Researcher found that more than 50,000 malicious applications are installed by malwares per day which display millions of malicious advertisements. According to latest research, malicious applications earn more than $300,000 per month [2]. Researches are more interested in coming up with more and more malware analysis and detection techniques because of drastic increase of android malwares. Static and dynamic analyses are the two main standpoints for malware detection.

In this project, dynamic analysis approach has been used to distinguish between malicious applications and benign applications. API calls and its contextual information like package name, parameter values, return values, etc., have been extracted at application’s runtime. Based on API-level data analysis, users get to know overall working flow of an application and get valuable insights about malware behavior.

The further outline of this document is as follow. Chapter 2 provides relevant background related to Android architecture, Application Program Interface (API), overview of malware and types of malware, and malware detection techniques. Chapter 3 discusses overall methodology and implementation process to analyze malicious applications and experiment results. Finally, Chapter 4 highlights conclusion and Chapter 5 provides suggestions for possible future work.
Chapter 2

BACKGROUND

Before discussing detailed framework of the project, it is important to know Android architecture, basics of Application Program Interface (API), and Android malware. Section 2.1 describes various layers of Android architecture. Section 2.2 gives overview of API. Finally, Section 2.3 provides overview of malware and its detection techniques.

2.1 Android System Architecture

Figure 1 illustrates architecture of Android software. Here, blue components describe applications and its frameworks. They are written in Java to understand and execute the Dalvik VM. Green components are libraries which are written in C or C++. Red components at the bottom represent Linux kernel [1][3][4]. Following subsections briefly describe various abstraction layers.

2.1.1 Linux kernel

This layer includes important features like memory system management, binder IPC driver, and wake locks (a software mechanisms for power management which restrict your Android device to go into deep sleep) for Android embedded system [1].

2.1.2 Libraries

The libraries are written in native C/C++. By using their components, libraries are exposed to Android Runtime and Application Framework. These libraries plays important role in memory consumption, device’s audio and video codecs, and CPU optimization [1].
2.1.3 Android Runtime

Android runtime is a middleware component and runtime territory for applications which contains Dalvik Virtual Machine code and core libraries. It is also capable of compiling application at the time of installation [1].

2.1.4 Application Framework

This layer allows to develop applications to use fundamental reusable packages and libraries. Most of the components of this layer are run in background as a part of applications on the device [1]. The major components of this layer can be seen in the following table:

Figure 1. Android System Architecture [4]
Table 1. Major Components of Application Framework

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Manager</td>
<td>It supervises running applications. By using an interface offered by this component, user can cooperate with the running application.</td>
</tr>
<tr>
<td>Intent/Notification Manager</td>
<td>It helps components to communicate with other components by acting as a messenger.</td>
</tr>
<tr>
<td>Content Manger</td>
<td>It provides an interface and helps applications to share data with others.</td>
</tr>
<tr>
<td>Telephony Manager</td>
<td>It provides telephony service information like the IMEI number from the device.</td>
</tr>
<tr>
<td>Location Manager</td>
<td>It allows applications to use and obtain device’s topological location periodically.</td>
</tr>
</tbody>
</table>

2.1.5 Applications

This layer is responsible for interaction between the device and end-users. It is built on top of Application framework and distributed as Android Package files (.apk).

2.2 Application Program Interface (API)

An application program interface (API) is a code with set of functions, commands, and protocols that allows software program to interact with other software program or with an exterior system [5].

It helps developers by providing standard commands to execute common operations so they don’t need to write the code from scratch. The main objective of API is to create a path for programmers to write code which requests services from an Android OS or another application. At any time, it will not stop to function if the underlying system is upgraded [6].
Figure 2. API Analogy

Basically, APIs are implemented by function calls. Whenever developer or tool wants to access data from an application, they have to call related APIs. The syntax for API call is described in the target application. Developers can include API calls and make request through their code. Basically, an API is supposed to grant access to the application by providing resources for requesting services of the program. Following table shows the API calls categorized by resources and utilities [7].

Table 2. Categories of API calls

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Method Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Resolver</td>
<td>It provide access to content providers.</td>
<td>Insert(), query()</td>
</tr>
<tr>
<td>Context</td>
<td>It helps to get universal data like classes,</td>
<td>openFileOutput(),</td>
</tr>
<tr>
<td></td>
<td>resources, assets, etc.</td>
<td>getApplicationInfo()</td>
</tr>
<tr>
<td>Intents</td>
<td>It allows applications to launch other events,</td>
<td>setDataAndType(),</td>
</tr>
<tr>
<td></td>
<td>activities, and services. It also allows</td>
<td>addFlags(),</td>
</tr>
<tr>
<td></td>
<td>applications to interact with device’s hardware.</td>
<td>setDataAndType()</td>
</tr>
<tr>
<td>ActivityManager</td>
<td>It provides access to intercommunicate with</td>
<td>getRunningServices(),</td>
</tr>
<tr>
<td></td>
<td>other running activities of the device.</td>
<td>getMemoryInfo()</td>
</tr>
<tr>
<td>PackageManager</td>
<td>It helps to get information related to installed</td>
<td>getInstalledPackages()</td>
</tr>
<tr>
<td></td>
<td>packages on the target system.</td>
<td></td>
</tr>
<tr>
<td>SmsManager</td>
<td>It helps to manage SMS operations.</td>
<td>sendTextMessage()</td>
</tr>
<tr>
<td>TelephonyManager</td>
<td>It helps to fetch information related system’s</td>
<td>getDeviceId(),</td>
</tr>
<tr>
<td></td>
<td>telephony services.</td>
<td>getLine1Number()</td>
</tr>
<tr>
<td>ConnectivityManager, NetworkInfo,</td>
<td>It provide network related information</td>
<td>getNetworkInfo(),</td>
</tr>
<tr>
<td>WifiManager</td>
<td></td>
<td>isConnected(),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getWifiState()</td>
</tr>
<tr>
<td>HttpURLConnection</td>
<td>This class allows to establish connection with</td>
<td>setRequestMethod(),</td>
</tr>
<tr>
<td></td>
<td>remote servers, also send and receive data.</td>
<td>getInputStream()</td>
</tr>
<tr>
<td>OS package</td>
<td>It allows IPC services, process management, and message passing.</td>
<td>sendMessage(), killProcess()</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>IO Package</td>
<td>This class provides IO process related services.</td>
<td>readLines(), mkdir()</td>
</tr>
<tr>
<td>String, StringBuilder, StringBuffer</td>
<td>It allows to create and manipulate strings.</td>
<td>indexOf(), Append()</td>
</tr>
<tr>
<td>Timer</td>
<td>It schedule tasks for future execution.</td>
<td>schedule()</td>
</tr>
<tr>
<td>Crypto</td>
<td>This class is used for data encryption, decryption, and key agreement.</td>
<td>getInstance(), doFinal()</td>
</tr>
</tbody>
</table>

2.3 Android application Malware

The malware is also called as malicious software which is purely designed to damage or deactivate the target systems like tablets and smartphones. They also allow malicious users to steal personal information or remotely control targeted device. Many ways malware can enter into the target system such as while copying files from other devices or downloading contents from the web. After entering into the system, it infects the system by checking its weaknesses. Today, the way malwares spread across the internet is a global concern.

2.3.1 Types of Malware

The majority of Android malware are either Trojan or Spyware. Both types of malwares use social engineering techniques to cheat users.

2.3.1.1 Trojan

It is also known as SMS Trojans or Fake installer. These malwares feign to be genuine application installer. When user tries to install this kind of application on device, it executes and asks for service agreement. Once user agrees, it runs as background process which sends premium rated text messages. It also obtains permissions to user’s personal information [3]. This malicious
version contains repackaged applications. It provides equivalent functionality as the original applications, but they have other code which secretly send messages in background [1].

2.3.1.2 Spyware

The user’s private data is invaded if this kind of malware attacks the device. The information of the user’s private data could be sent to malware’s private server. Some of the examples of user’s private data could be phone details, geographical locations, sensitive information like credit card numbers and bank passwords, so on. If malware is a kind of botnet then it could start specific activities through its server [1].

2.3.2 Application malware detection techniques

In recent past, the main concentration of the researchers is to come up with malware detection techniques. The two main classification techniques for malware detection are static analysis and dynamic analysis.

2.3.2.1 Static Analysis

The analysis of the code is performed to identify the malicious application when Static analysis is used. Analysis of the code is not done by running the application on either an Android emulator or device. But the code is analyzed statically and features are extracted to check for malicious applications. The static analysis is less time consuming and low computation cost. But this analysis may fail to detect unknown malware [3].

2.3.2.2 Dynamic Analysis

On contrast, Dynamic analysis technique detects malware by running the application on either an Android emulator or the device. This technique loads the code dynamically. The
application behavior is recorded during runtime and hence it is a very useful technique. But, sometimes depending on the code volume, this technique may not be successful. One other possibility is that the dynamic analysis may fail to catch the malicious applications during feature recording. As there is a bottleneck of running the application when compared to static analysis, this method is not sometimes preferred [3].
Chapter 3

DYNAMIC ANALYSIS OF MALICIOUS APPLICATIONS

This chapter describes methodology used to extract dynamic features in Section 3.1, project design and implementation in Section 3.2, followed by results of experiments in Section 3.3.

3.1 Methodology

This section describes a method to detect malicious applications by analyzing API calls dynamically. Each application contains numbers of APIs to do various kind of activities. So, analysis of application by using API calls has become one of the important technique to know its behavior.

In this project, AndroidHooker has been used to hook APIs of the running applications. AndroidHooker provides various applications and tools to intercept and modify API calls from which EmulatorCreator, HookerXp and APK-Instrumenter frameworks have been used for this project. Here, EmulatorCreator contains a list of scripts to create emulator and installs various necessary applications automatically. HookerXp is a tool written in python to control the android device. It also triggers installation and stimulation of an application on the device. APK-Instrumenter is an Android application that must be installed prior to the analysis on an Android device [8].

One major limitation of AndroidHooker is that it doesn’t intercept APIs which are not declared in APK-Instrumenter application and fails to intercept APIs of external packages like javax.mail. In this project, new functions have been added in APK-Instrumenter hooker code to intercept additional APIs which can be found in Appendix A.
This project is done into three phases: 1) Create emulator and install required app to intercept API calls, 2) Install and run malicious applications on emulator, and extract intercepted API calls log, 3) Analyze log, check application behavior, and represent it graphically. Figure.3 describes the process in detail.

![Figure.3 Malware Application Detection Flow](image)

The first phase is to create emulator and install applications namely APK-Instrumenter, SuperSu and Substrate. Here, SuperSu application is used to get root access of the device. Substrate application helps to get access of personal information and specific APIs, and modify access of
specific methods. In this project, Substrate application is used to intercept API calls and aggregate all their contextual information such as parameters, returned values, package name, etc.

In second phase, open created emulator by using hooker script with valid configuration file and then install applications either via configuration file or manually from emulator shell. During application’s runtime, API calls with other related information are intercepted and being stored in log file. The log file can be found in emulator’s SD card folder. The last step of second phase is to fetch intercepted log before closing emulator.

The third phase is to simplify and analyze intercepted log. Hooker records all activities of running applications on emulator and saves them into log file. Log size generated through Hooker might be too large to analyze. This project uses python scripts which help to fetch important information from the log for particular application and save them into csv format. These csv files are being used by another python scripts which uses Graphviz software, an open source graph visualization software, to simplify results and know overall behavior of the application.

3.2 Design and Implementation

3.2.1 Environment Setup

This project uses virtual environment to run and monitor suspicious applications. First, VMware workstation 11 have been installed which is a virtualization tool to create and run multiple operating systems on a desktop. Next step is to install Kali Linux2.0 64 bits in VMware as it is easy to get paths and parameters from Linux system. In Kali Linux, first install Android Studio to understand, modify and build APK-Instrumenter application code. After installing Android Studio, open SDK manager and install latest SDK APIs. Also, consequently install Python and Graphviz to run all the scripts and plot graphs respectively. At the end, install AndroGuard to analyze Android
manifest file for permissions and activities, unpacking of apks, decode Dalvik bytecode, and convert decoded code to Java bytecode for further analysis.

3.2.2 Execute Experiments

In this phase, the API calls have been extracted dynamically during application runtime. First, open emulator instance which is created as previously mentioned in setup phase. Then, install applications on emulator and execute to start recording of API calls. Next step is connect to the emulator instance using the Android Debug Bridge (adb). Table 3 shows the list of commands to run this experiment.

Table 3. Commands to run experiment

<table>
<thead>
<tr>
<th>Command To</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate to the android-sdk folder</td>
<td>cd /usr/share/android-sdk/</td>
</tr>
<tr>
<td>Check available android targets</td>
<td>cd tools/ android list target</td>
</tr>
<tr>
<td>Create Emulator</td>
<td>cd tools/emulatorCreator</td>
</tr>
<tr>
<td></td>
<td>python HookerInstaller.py -s &lt;sdk-path&gt; -a &lt;emulator-name&gt; -t &lt;android-target&gt; -d &lt;avd_directory-path&gt;</td>
</tr>
<tr>
<td>Install application</td>
<td>cd /usr/share/android-sdk/platform-tools/</td>
</tr>
<tr>
<td></td>
<td>./adb install ApplicationToInstall.apk</td>
</tr>
<tr>
<td>Run the experiment</td>
<td>python hooker_xp.py -c APIcallAnalysisConfigFile.conf</td>
</tr>
<tr>
<td>Pull log from emulator’s SD card</td>
<td>cd /usr/share/android-sdk/platform-tools/</td>
</tr>
<tr>
<td></td>
<td>./adb pull /sdcard/ FoldernameTo CopyLog</td>
</tr>
</tbody>
</table>

Table 3 shows commands to run experiment. Following are detailed execution steps for mentioned process in Figure.3.

1) Open terminal and navigate to the android-sdk folder
2) Check available targets installed on the system

3) Launch the automatic script of hooker for an easier creation of an emulator. Go to hooker emulatorCreator folder and run HookerInstaller script. For Example,

```
python HookerInstaller.py -s /usr/share/android-sdk/ -a APIcallEmulator -t 4 -d /root/.android/avd/
```

4) Once emulator is up, install SuperSu and Substrate applications

5) Install APK-Instrumenter hooker application using ADB and restart emulator. For example, go to platform-tools folder of android-sdk and run:

```
./adb install /root/Desktop/API-Hooker.apk
```

6) Close your emulator.

7) To run experiment, execute hooker_xp.py script with valid configuration which contains emulator name, SDK path, AVD path, AndroGuard path, etc. For example,

```
python hooker_xp.py -c APIcallAnalysisConfigFile.conf
```

8) Install application which user wants to test via adb. For Example,

```
./adb install ContactUpload.apk
```

9) Before closing emulator, pull log from emulator’s SD card

```
./adb pull /sdcard/hooker/events.logs /root/Desktop/Logs/Interceptedevents.logs
```

### 3.2.3 Analysis of the API Call Logs

The captured log is very big in size even if you run an application for small amount of time which contains all contextual information like parameter type, parameter value, return type, return value, package name, class name, etc., along with APIs. It is difficult to go through whole log to
know the process of an application. It needs to be converted into more readable format for quick analysis. Figure.4 shows sample log format.

Figure.4 Sample API call log snippet

### 3.2.3.1 Generate CSV files

Considering big log file size, it needs to be converted into csv for better readability and analysis. Here, two scripts have been written to read log file, fetch particular data, and save into csv file. First script fetches method name, parameter values and return values from the log. While the second script searches for all personal and network related information like package name, method name, email, username, password, port number, hostname, receiver port, receiver host, sending port, sender host, local port, socket address, device Id, network operator name, sim serial number, sim country ISO, SD card return value, and URI. These generated csv files help to understand which kind of data application is trying to fetch from the device and send to the network.

### 3.2.3.2 Graphical Analysis

After generating all csv files, the next step is to generate graph to know behavior of installed applications. To do so, various python scripts have been written which post-process generated csv files and convert data into graphical representation using Graphviz. Normally, Graph
and Digraph are two classes of Graphviz module. These scripts use Digraph to create graph
descriptions in DOT language and generate results in graphical format [9]. Table 4 shows four
types of graphs generated through python scripts.

Table 4. API call analysis graphs

<table>
<thead>
<tr>
<th>Graph</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detailed graph</td>
<td>This graph represents very detailed information about methods being called by target application. It plots graph for each method which includes information related to parameters and return values.</td>
</tr>
<tr>
<td>API Count Graph</td>
<td>This graph provides information for number of times the APIs called by an application.</td>
</tr>
<tr>
<td>Application Flow Graph</td>
<td>This graph uses API count and detailed graph to come up with application flow by doing static analysis.</td>
</tr>
<tr>
<td>Crucial Data Graph</td>
<td>This graph represents all personal and network related information used by an application.</td>
</tr>
</tbody>
</table>

3.3 Experiment Results

In this project, five malicious applications have been analyzed. The first three applications
namely Data Peek, Spycam, and Call Tracker were developed by Aditi Kulkarni from CSUS. The
remaining applications were found online named Puzzle 2048 and QuxiFlashlight. All subsections
of section 3.3 include mentioned four types of graphs in Table.4. Each application calls many
methods which are not possible to show in Method Detail graph. For that reason, only one method
detail graph is shown for each tested application.

3.3.1 Dynamic Analysis of Data Peek

By analyzing all graphs for Data Peek application, the conclusion is that the application
tries to steal contact information from the device. First, it reads all contact information like ID,
display name, photo URI, and phone number. It also tries to create database and table named ContactManager and ContactTable respectively to save contact information. Later, it reads all stored information from the table and sends outside using SMTP server. Following are the graphs used to analyze Data Peek application.

Figure.5 Example of Method Detail Graph for Data Peek Application
Figure 6 Application Flow Graph for Data Peek Application
Figure 7 Method Count Graph of Data Peek Application
Figure 8 Crucial Data Graph for Data Peek Application
3.3.2 Dynamic Analysis of SpyCam

By analyzing all graphs for SpyCam application, the conclusion is that it uses front or back camera to take pictures, saves them in gallery and uploads to server without letting the user know. First, SpyCam application tries to get camera information if the device has camera feature. Once it finds that feature is available, it opens device’s camera in background, sets all camera features, takes picture every few seconds, and saves them into device’s internal storage. It fetches pictures from sdcard and sends it to sever (192.168.78.129) using SavePhoto.php file. Following are the graphs used to analyze SpyCam application.

Figure.9 Example of Method Detail Graph for SpyCam Application
Figure 10 Application Flow Graph for SpyCam Application
Figure 11 Method Count Graph for SpyCam Application
3.3.3 Dynamic Analysis of Call Tracker

By analyzing all graphs for Call Tracker application, the conclusion is that the application tries to record incoming or outgoing calls, and uploads recorded call to specific server. First, application checks PHONE_STATE and starts required services. Then, it records all incoming and outgoing call of the device, creates “.raw” and “.wav” file of the recording, stores in sdcard, and uploads this .wav file to sever (192.168.78.129) using UploadToSever.php file. Following are the graphs used to analyze Call Tracker application.

![Figure.12 Crucial Data Graph for SpyCam Application](http://192.168.78.129/SavePhoto.php)

**Figure.13 Example Method Detail Graph for Call Tracker application**
**Figure.14 Application Flow Graph for Call Tracker Application**
Figure 15 Method Count Graph for Call Tracker Application
3.3.4 Dynamic Analysis of Puzzle2048

By analyzing all graphs for Puzzle2048 application, the conclusion is that the application first checks permissions like Internet, Wi-Fi, read phone state, etc. It also tries to fetch information like phone number, country information, Android version, API level, phone model, phone type, display information, SIM serial number, device ID, list of installed packages, account details, list of running tasks, etc. Then it encrypts fetched data from the device and uploads encrypted data to specific URLs. Following are the graphs used to analyze Puzzle2048 application.
Figure 17 Example of Method Detail Graph for Puzzle2048 Application
Figure 18: Application Flow Graph for Puzzle2048 Application
Figure 19: Crucial Data Graph for Puzzle2048 Application

3.3.5 Dynamic Analysis of QuxiFlashlight

By analyzing all graphs for QuxiFlashlight application, the conclusion is that the application tries to fetch phone type, SIM serial number and SIM state, device ID, accounts information, country information, network operator, list of installed packages, account details, list
of running tasks, etc., and sends them to specific server. After installation, it hides its icon from device screen after sometime but keeps running in background. It also flashes advertisement frequently. Following are the graphs used to analyze QuxiFlashlight application.

Figure 20: Example of Method Detail Graph for QuxiFlashlight Application
Figure 22: Crucial Data Graph for Quxi Flashlight Application
Chapter 4

CONCLUSION

The main objective of this project was to detect Android application malware by using dynamic analysis technique. Section 3.2 represents methodology to intercept API calls at application runtime. By analyzing graphical results of intercepted API calls, users can get to know application flux and predict whether it is legitimate or perverted. For this analysis, analysts must depend on API calls and related information to achieve better precision. Researchers can use this methodology to detect new application malwares. They can also identify detailed behavior of known suspicious applications using mentioned techniques in this project.
Chapter 5

FUTUREWORK

The analyzed results of intercepted API calls of running applications have recognized to be very effective to detect malicious application. However, this approach has prospects and could be explored more. For future enhancement of this project, more research is required on malicious applications and its APIs. Newly found APIs can be added to existing code to enhance current project. Another enhancement of this project can be achieved by mapping API calls with permissions. At the time of installation, application requests multiple permissions. This feature can help to understand activities of the requested permissions. It also helps to conclude whether particular permission is really required or not.
APPENDIX A

Changed code of APK-Instrumenter

Filename: Hooker.java

protected void vent(InterceptEvent event, Context appContext) {
    if (event == null) {
        return;
    }

    // Compute relative timestamp
    long relativeTimestamp = event.getTimestamp() -
    this.startTimestamp;
    event.setRelativeTimestamp(relativeTimestamp);

    if (SubstrateMain.NETWORK_MODE || SubstrateMain.FILE_MODE) {
        SubstrateMain.log("FILE_MODE :");
        this.sendEventToCollectService(event, appContext);
    }

    if (SubstrateMain.DEBUG_MODE) {
        SubstrateMain.log("DEBUG_MODE :");
        //Arpita
        this.sendEventToCollectService(event, appContext);
    }
}
private void attachOnCursorClass()
{
    Map<String, Integer> methodsFromLocationToHook = new HashMap<String, Integer>();
    methodsFromLocationToHook.put("getColumnIndex", 2);
    methodsFromLocationToHook.put("getColumnName", 2);
    methodsFromLocationToHook.put("getString", 2);
    methodsFromLocationToHook.put("moveToNext", 2);
    methodsFromLocationToHook.put("moveToFirst", 2);
    methodsFromLocationToHook.put("moveToLast", 2);
    methodsFromLocationToHook.put("requery", 2);
    try
    {
        hookMethods(null, "android.database.Cursor", methodsFromLocationToHook);
        SubstrateMain.log("hooking android.database.Cursor methods sucessful");
    }
    catch (HookerInitializationException e) {
        SubstrateMain.log("hooking android.content.Context methods has failed", e);
    }
}
private void attachOnCursorClass()
{
    Map<String, Integer> methodsFromLocationToHook = new HashMap<String, Integer>();
    methodsFromLocationToHook.put("getColumnIndex", 2);
    methodsFromLocationToHook.put("getName", 2);
    methodsFromLocationToHook.put("getCount", 2);
    methodsFromLocationToHook.put("moveToNext", 2);
    methodsFromLocationToHook.put("moveToFirst", 2);
    methodsFromLocationToHook.put("moveToLast", 2);
    methodsFromLocationToHook.put("requery", 2);
    try
    {
        hookMethods(null, "android.database.Cursor",
                    methodsFromLocationToHook);
        SubstrateMain.log("hooking android.database.Cursor methods sucessful");
    }
    catch (HookerInitializationException e) {
        SubstrateMain.log("hooking android.content.Context methods has failed", e);
    }
}

private void attachOnCommandMap()
{
    final String className = "javax.activation.CommandMap";
    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();
    //methodsToHook.put("createCommandMap", 0);
    //methodsToHook.put("getMimeTypes", 0);
    //methodsToHook.put("getPreferredCommands", 0);
    methodsToHook.put("getDefaultCommandMap", 0);
    try
    {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                        .append(" methods sucessful").toString());
    }
    catch (HookerInitializationException e) {
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                        .append(" methods has failed").toString(), e);
    }
private void attachOnMailcapCommandMap() {
    final String className = "javax.activation.MailcapCommandMap";
    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();
    methodsToHook.put("addMailcap", 0);
    methodsToHook.put("createDataContentHandler", 0);
    methodsToHook.put("getAllCommands", 0);
    methodsToHook.put("getCommand", 0);
    methodsToHook.put("getMimeTypes", 0);
    methodsToHook.put("getPreferredCommands", 0);
    methodsToHook.put("getNativeCommands", 0);
    try {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                .append(" methods sucessful").toString());
    } catch (HookerInitializationException e) {
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                .append(" methods has failed").toString(), e);
    }
}

private void attachOnBodyPart() {
    final String className = "javax.mail.BodyPart";
    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();
    methodsToHook.put("getParent", 0);
    try {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                .append(" methods sucessful").toString());
    } catch (HookerInitializationException e) {
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                .append(" methods has failed").toString(), e);
    }
}
private void attachOnMultipart() {
    final String className = "javax.mail.Multipart";

    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();

    methodsToHook.put("addBodyPart", 0);
    methodsToHook.put("getBodyPart", 0);
    methodsToHook.put("getCount", 0);
    methodsToHook.put("getParent", 0);
    methodsToHook.put("removeBodyPart", 0);
    //methodsToHook.put("setMultipartDataSource", 0);
    methodsToHook.put("setParent", 0);
    methodsToHook.put("writeTo", 0);

    try {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                            .append(" methods successful").toString());
    } catch (HookerInitializationException e) {
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                               .append(" methods has failed").toString(), e);
    }
}

private void attachOnPasswordAuthentication() {
    final String className = "javax.mail.PasswordAuthentication";

    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();

    methodsToHook.put("getPassword", 0);
    methodsToHook.put("getUserName", 0);

    try {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                            .append(" methods successful").toString());
    } catch (HookerInitializationException e) {
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
                               .append(" methods has failed").toString(), e);
    }
}
private void attachOnSession() {
    final String className = "javax.mail.Session";

    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();

    methodsToHook.put("addProvider", 0);
    methodsToHook.put("getDebug", 0);
    methodsToHook.put("getDebugOut", 0);
    methodsToHook.put("getDebugInstance", 0);
    methodsToHook.put("setProvider", 0);
    methodsToHook.put("setProtocolForAddress", 0);
    methodsToHook.put("setPasswordAuthentication", 0);
    methodsToHook.put("setDebugOut", 0);
    methodsToHook.put("setDebug", 0);
    methodsToHook.put("requestPasswordAuthentication", 0);
    methodsToHook.put("getTransport", 0);
    methodsToHook.put("getStore", 0);
    methodsToHook.put("getProviders", 0);
    methodsToHook.put("getProvider", 0);
    methodsToHook.put("getProperty", 0);
    methodsToHook.put("getPasswordAuthentication", 0);
    methodsToHook.put("getProperties", 0);
    methodsToHook.put("getInstance", 0);
    methodsToHook.put("getFolder", 0);

    try {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className).
                        append(" methods sucessful").toString());
    }
    catch (HookerInitializationException e) {
        SubstrateMain.log(new StringBuilder("hooking ").append(className).
                        append(" methods has failed").toString(), e);
    }
}

private void attachOnTransport() {
    final String className = "javax.mail.Transport";

    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();

    //methodsToHook.put("addTransportListener", 0);
    //methodsToHook.put("notifyTransportListeners", 0);
    //methodsToHook.put("removeTransportListener", 0);
    methodsToHook.put("send", 0);
    //methodsToHook.put("sendMessage", 0);
try {
    hookMethods(null, className, methodsToHook);
    SubstrateMain.log(new StringBuilder("hooking ").append(className)
        .append(" methods sucessful")).toString();
} catch (HookerInitializationException e) {
    SubstrateMain.log(new StringBuilder("hooking ").append(className)
        .append(" methods has failed")).toString(), e);
}

private void attachOnInternetAddress() {
    final String className = "javax.mail.internet.InternetAddress";
    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();
    
    methodsToHook.put("clone", 0);
    methodsToHook.put("equals", 0);
    methodsToHook.put("getAddress", 0);
    methodsToHook.put("getGroup", 0);
    methodsToHook.put("getLocalAddress", 0);
    methodsToHook.put("getPersonal", 0);
    methodsToHook.put("getType", 0);
    methodsToHook.put("hashCode", 0);
    methodsToHook.put("isGroup", 0);
    methodsToHook.put("parse", 0);
    methodsToHook.put("parseHeader", 0);
    methodsToHook.put("setAddress", 0);
    methodsToHook.put("setPersonal", 0);
    methodsToHook.put("toString", 0);
    methodsToHook.put("toUnicodeString", 0);
    methodsToHook.put("validate", 0);
    
    try {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
            .append(" methods sucessful")).toString();
    } catch (HookerInitializationException e) {
        SubstrateMain.log(new StringBuilder("hooking ").append(className)
            .append(" methods has failed")).toString(), e);
    }
}
private void attachOnMimeBodyPart() {
    final String className = "javax.mail.internet.MimeBodyPart";

    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();

    methodsToHook.put("addHeader", 0);
    methodsToHook.put("addHeaderValue", 0);
    methodsToHook.put("attachFile", 0);
    methodsToHook.put("getAllHeaderLines", 0);
    methodsToHook.put("getAllHeaders", 0);
    methodsToHook.put("getContent", 0);
    methodsToHook.put("getContentID", 0);
    methodsToHook.put("getContentLanguage", 0);
    methodsToHook.put("getContentMD5", 0);
    methodsToHook.put("getContentStream", 0);
    methodsToHook.put("getContentType", 0);
    methodsToHook.put("getDataHandler", 0);
    methodsToHook.put("getDescription", 0);
    methodsToHook.put("getDisposition", 0);
    methodsToHook.put("getEncoding", 0);
    methodsToHook.put("getFileName", 0);
    methodsToHook.put("getHeader", 0);
    methodsToHook.put("getInputStream", 0);
    methodsToHook.put("getLineCount", 0);
    methodsToHook.put("getMatchingHeaderLines", 0);
    methodsToHook.put("getMatchingHeaders", 0);
    methodsToHook.put("getNonMatchingHeaderLines", 0);
    methodsToHook.put("getNonMatchingHeaders", 0);
    methodsToHook.put("getRawInputStream", 0);
    methodsToHook.put("getSize", 0);
    methodsToHook.put("isMimeType", 0);
    methodsToHook.put("removeHeader", 0);
    methodsToHook.put("saveFile", 0);
    methodsToHook.put("setContent", 0);
    methodsToHook.put("setContentID", 0);
    methodsToHook.put("setContentLanguage", 0);
    methodsToHook.put("setContentMD5", 0);
    methodsToHook.put("setDataHandler", 0);
    methodsToHook.put("setDescription", 0);
    methodsToHook.put("setDescription", 0);
    methodsToHook.put("setDisposition", 0);
    methodsToHook.put("setFileName", 0);
    methodsToHook.put("setHeader", 0);
    methodsToHook.put("setText", 0);
    methodsToHook.put("updateHeaders", 0);
    methodsToHook.put("writeTo", 0);

    try {
        hookMethods(null, className, methodsToHook);
        SubstrateMain.log(new StringBuilder("hooking ").append(className))
catch (HookerInitializationException e) {
    SubstrateMain.log(new StringBuilder("hooking 
").append(className)
        .append(" methods has failed").toString(), e);
}

private void attachOnMimeMessage() {
    final String className = "javax.mail.internet.MimeMessage";
    Map<String, Integer> methodsToHook = new HashMap<String, Integer>();
    methodsToHook.put("addFrom", 0);
    methodsToHook.put("addHeader", 0);
    methodsToHook.put("addHeaderValue", 0);
    methodsToHook.put("addRecipients", 0);
    methodsToHook.put("createInternetHeaders", 0);
    methodsToHook.put("createMimeMessage", 0);
    methodsToHook.put("getAllHeaderLines", 0);
    methodsToHook.put("getAllHeaders", 0);
    methodsToHook.put("getAllRecipients", 0);
    methodsToHook.put("getContent", 0);
    methodsToHook.put("getContentID", 0);
    methodsToHook.put("getContentLanguage", 0);
    methodsToHook.put("getContentMD5", 0);
    methodsToHook.put("getContentStream", 0);
    methodsToHook.put("getContentType", 0);
    methodsToHook.put("getDataHandler", 0);
    methodsToHook.put("getDescription", 0);
    methodsToHook.put("getDisposition", 0);
    methodsToHook.put("getEncoding", 0);
    methodsToHook.put("getFileName", 0);
    methodsToHook.put("getFlags", 0);
    methodsToHook.put("getFrom", 0);
    methodsToHook.put("getHeader", 0);
    methodsToHook.put("getInputStream", 0);
    methodsToHook.put("getLineCount", 0);
    methodsToHook.put("getMatchingHeaderLines", 0);
    methodsToHook.put("getMatchingHeaders", 0);
    methodsToHook.put("getMessageID", 0);
    methodsToHook.put("getNonMatchingHeaderLines", 0);
    methodsToHook.put("getNonMatchingHeaders", 0);
    methodsToHook.put("getRawInputStream", 0);
    methodsToHook.put("getReceivedDate", 0);
    methodsToHook.put("getRecipients", 0);
    methodsToHook.put("getReplyTo", 0);
    methodsToHook.put("getSender", 0);
    methodsToHook.put("getSentDate", 0);
    methodsToHook.put("getSize", 0);
    methodsToHook.put("getSubject", 0);
    methodsToHook.put("isMimeType", 0);
methodsToHook.put("isSet", 0);
methodsToHook.put("parse", 0);
methodsToHook.put("removeHeader", 0);
methodsToHook.put("reply", 0);
methodsToHook.put("saveChanges", 0);
methodsToHook.put("setContent", 0);
methodsToHook.put("setContentID", 0);
methodsToHook.put("setContentLanguage", 0);
methodsToHook.put("setContentMD5", 0);
methodsToHook.put("setDataHandler", 0);
methodsToHook.put("setDescription", 0);
methodsToHook.put("setDisposition", 0);
methodsToHook.put("setFileName", 0);
methodsToHook.put("setFlags", 0);
methodsToHook.put("setFrom", 0);
methodsToHook.put("setHeader", 0);
methodsToHook.put("setRecipients", 0);
methodsToHook.put("setReplyTo", 0);
methodsToHook.put("setSender", 0);
methodsToHook.put("setSentDate", 0);
methodsToHook.put("setSubject", 0);
methodsToHook.put("setText", 0);
methodsToHook.put("updateHeaders", 0);
methodsToHook.put("updateMessageID", 0);
methodsToHook.put("writeTo", 0);
try {
    hookMethods(null, className, methodsToHook);
    SubstrateMain.log(new StringBuilder("hooking ").append(className)
              .append(" methods sucessful").toString());
} catch (HookerInitializationException e) {
    SubstrateMain.log(new StringBuilder("hooking ").append(className)
              .append(" methods has failed").toString(), e);
}
}
APPENDIX B

Configuration File for API-Instrument

# Path to reference AVD, emulators are clones of this reference
referenceAVD=/root/.android/avd/AHooker

# Path to the Android SDK
androidSDKPath=/usr/share/android

# Path to your local Android temporary directory
androidTemporaryPath=/tmp/android/

# Path to androguard framework
androguardPath=/root/androguard

# Type of device (real or emulated)
device=emulated

# Name of the analysis
name=Hooker_API_Call

type=manual
apks=../tools/sampleApps/IncomingCallSample.apk
prepareApks=../tools/APK-contactGenerator/ImportContacts.apk
elasticsearch_mode=false
file_mode=true
#!/usr/bin/python

import gzip, sys, getopt, re, csv, operator
import os.path
import fileinput

outfile=''
logfile=''

#-------------------Definition to print error-------------------#
def errorMsg():
    print ('The format should be')
    print ('python ConvertToCSV.py -l <log unix path> -o <output name without any extension> -p <packagename>\n')
    sys.exit()

# I give credits to kasey.fr, sourceware, ethicalhackers
# to read and check commandline in python
#-------------------Check arguments-------------------#
try:
    opts, args =
    getopt.getopt(sys.argv[1:],"hl:o:p:","log=","out=","package=")
except:
    errorMsg()

if len(opts):  
    for option, arg in opts:
        print ('python ConvertToCSV.py -l <log unix path> -o <output name without any extension> -p <packagename>\n')
        sys.exit()
        if option == '-h':
            print ('python ConvertToCSV.py -l <log unix path> -o <output name without any extension> -p <packagename>
')
            sys.exit()
        elif option in ("-l", "--log"):
            logfile = arg
        elif option in ("-o", "--out"):
            outcsv = arg
        elif option in ("-p", "--package"):
            package = arg
        else:
            errorMsg()
    else:
        errorMsg()

#------------------- set array-------------------#
Parameters = []; MethodName = []; PackageName = []; Return= [];
Timestamp = [];

#------------------- LINE PARSER-------------------#
Parameters.append('Parameters')
MethodName.append('MethodName');
PackageName.append('PackageName');
Return.append("Return")
Timestamp.append('Timestamp');
remove previous CSV if exist

```python
CSV_NAME = outcsv, '.csv'
CSV = ''.join(CSV_NAME)
if os.path.exists(CSV):
    os.remove(CSV)

line_number = 0
if os.path.isfile(logfile) == True:
    #os.system('python blank_remove.py -l ' + logfile + ' -o ' + CSV)
    os.system('py blank_remove.py -l ' + logfile + ' -o ' + CSV)
inputfile = open(logfile, 'r')
lines = inputfile.readlines()
lines.sort()
while line_number < len(lines):
    line = lines[line_number]
    if line.strip():
        line = line.rstrip()
        new_line = re.sub('[{}]', '', line)
        split_line1 = re.split('"Parameters":', new_line)
        parameters = split_line2[0].replace('""', '"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
final_data = map(list, zip(Timestamp, PackageName, MethodName, 
Parameters, Return))

os.remove(CSV)
with open(CSV, 'w') as out:
    output = csv.writer(out)
    for i in final_data:
        output.writerow(i)

# I give credit to johnroach and grokbase for read and write file information
# sort csv by timestamp
data = csv.reader(open(CSV), delimiter=',')
sortedlist = sorted(data, key=operator.itemgetter(0))
# now write the sorte result into new CSV file
with open(CSV, "wb") as csvf:
    fileWriterCSVfnm = csv.writer(csvf, delimiter=',')
    for row in sortedlist:
        fileWriter.writerow(row)
APPENDIX D

Script to collect crucial data from log file and save into CSV file

#!/usr/bin/python

import gzip, sys, getopt, re, csv
import os.path

outfile=''
logfile=''

#####--------------------------Definition to print error--------------------------####
def errorMsg():
    print ('The format should be')
    print ('python ConvertToCSV_Crucialdata.py -l <log unix path> -o <output name without any extenstion> -p <packagename>

sys.exit()

# I give creadits to kasey.fr, sourceware, ethicalhackers for read and check commandline information
#####-----------------Check arguments-----------------------------####
try:
    opts, args = getopt.getopt(sys.argv[1:],"hl:o:p:","log=","out=","package=")
except:
    errorMsg()

if len(opts):
    for opt, arg in opts:
        if opt == '-h':
            print ('python ConvertToCSV_Crucialdata.py -l <log unix path>

sys.exit()
    elif opt in ('-l', "--log"):
        logfile = arg
    elif opt in ('-o',"--out"):
        outcsv = arg
    elif opt in ('-p',"--package"):
        package = arg
    else:
        errorMsg()
else:
    errorMsg()

#####---------------- set array------------------------####
MethodName = []; PackageName = []; User=[]; Password = [];
Port = []; Host=[]; Recv_port = []; Recv_host = []; Send_port = [];
Send_host = []; DeviceId = []; NetworkOperatorName=[];
SimSerialNumber=[]; SimCountryIso=[]; SdcardReturnValue=[];
localPort=[]; Socketaddress=[]; Email=[];URI=[];
### remove previous CSV if exist

```python
CSV_NAME = outcsv, '.csv'
CSV = ''.join(CSV_NAME)
if os.path.exists(CSV):
    os.remove(CSV)

line_number = 0

if os.path.isfile(logfile):
    os.system('python blank_remove.py -l ' + logfile + ' -o ' + CSV)
inputfile = open(CSV, 'r')
lines = inputfile.readlines()
lines.sort()
while line_number < len(lines):
    line = lines[line_number]
    if line.strip():
        strUser = ''; strPassword = ''; strHost = '';
        strRecv_port = ''; strRecv_host = ''; strSend_port = '';
        strSend_host = ''; strDeviceId = '';
        strNetworkOperatorName = '';
        strSimSerialNumber = ''; strSimCountryIso = '';
        strSdcardReturnValue = ''; strlocalPort = '';
        strSocketaddress = ''; strEmail = ''; strUri = '';

        line = line.rstrip()
        new_line = re.sub('[{}]', '', line)
        split_line4 = re.split('"HookerName":"', new_line)
        split_line5 = re.split('"MethodName":"', split_line4[1])
        split_line6 = re.split('"PackageName":"', split_line5[1])
        pname = re.sub('"', '', split_line6[0])
        split_line7 = re.split('"InstanceID":"', split_line6[1])
        packname = re.sub('"', '', split_line7[0])

        # Find Network Operator Name if exist
        if pname == 'getNetworkOperatorName':
            NetworkOperatorNameColumn = line.find("ReturnValue":"")
            NetworkOperatorNamevalue = line[NetworkOperatorNameColumn, len(line)]
            strNetworkOperatorName = line[NetworkOperatorNameColumn + len("ReturnValue":"") : NetworkOperatorNamevalue]

        # Find DeviceID / IMEI number if exist
        if pname == 'getDeviceId':
            DeviceIdColumn = line.find("ReturnValue":"")
            DeviceIdvalue = line[DeviceIdColumn, len(line)]
            strDeviceId = line[DeviceIdColumn + len("ReturnValue":"") : DeviceIdvalue]
```
# Find User in line if exist
if "user=" in line:
    userColumn = line.find("user=")
    uservalue = line.find("," , userColumn, len(line))
    strUser = line[userColumn+len("user="):uservalue]

# Find password in line if exist
if "password=" in line:
    pwdColumn = line.find("password=")
    pwdvalue = line.find("," , pwdColumn, len(line))
    strPassword = line[pwdColumn+len("password="):pwdvalue]

# Find Port number in line if exist
if ",port=" in line:
    portColumn = line.find(",port=")
    portvalue = line.find(",", portColumn+1, len(line))
    strPort = line[portColumn+len(",port="):portvalue]

# Find Host name in line if exist
if ",host=" in line:
    hostColumn = line.find(",host=")
    hostvalue = line.find(",", hostColumn+1, len(line))
    strHost = line[hostColumn+len(",host="):hostvalue]

# Find Recieve Port in line if exist
if ",recv_port=" in line:
    recv_portColumn = line.find(",recv_port=")
    recv_portvalue = line.find(",", recv_portColumn+1, len(line))
    strRecv_port = line[recv_portColumn+len(",recv_port="):recv_portvalue]

# Find Recieve Host in line if exist
if ",recv_host=" in line:
    recv_hostColumn = line.find(",recv_host=")
    recv_hostvalue = line.find(",", recv_hostColumn+1, len(line))
    strRecv_host = line[recv_hostColumn+len(",recv_host="):recv_hostvalue]

# Find Send Port in line if exist
if ",send_port=" in line:
    send_portColumn = line.find(",send_port=")
    send_portvalue = line.find(",", send_portColumn+1, len(line))
    strSend_port = line[send_portColumn+len(",send_port="):send_portvalue]
# Find Send Host in line if exist
if ",send_host=" in line:
    send_hostColumn = line.find("recv_host=")
    send_hostvalue = line.find("," ,send_hostColumn+1,
    len(line))
    strSend_host = line[send_hostColumn+len(",send_host="):send_hostvalue]

# Find Send Host in line if exist
if "localPort=" in line:
    localPortColumn = line.find("localPort=")
    localPortvalue = line.find("," ,localPortColumn+1,
    len(line))
    strlocalPort = line[localPortColumn+len("localPort="):localPortvalue]
    strlocalPort = strlocalPort.strip('"')

# Find Sim Serial Number if exist
if pname=="getSimSerialNumber":
    SimSerialNumberColumn = line.find("ReturnValue\":"")
    SimSerialNumbervalue = line.find("",SimSerialNumberColumn, len(line))
    strSimSerialNumber = line[SimSerialNumberColumn+len("ReturnValue\":") :SimSerialNumbervalue]

# Find Sim Serial Number if exist
if pname=="getSimCountryIso":
    SimCountryIsoColumn = line.find("ReturnValue\":"")
    SimCountryIsovalue = line.find("",SimCountryIsoColumn, len(line))
    strSimCountryIso = line[SimCountryIsoColumn+len("ReturnValue\":") :SimCountryIsovalue]

# If reading anything from sdcard
if "ReturnValue\":"/mnt/sdcard/" in line:
    sdcardColumn = line.find("ReturnValue\":"/mnt/sdcard/")
    sdcardvalue = line.find("",sdcardColumn, len(line))
    checkExtension = line[sdcardColumn+len("ReturnValue\":"):/mnt/sdcard/":sdcardvalue]
    if checkExtension.find(">")!=-1:
        strSdcardReturnvalue = checkExtension

# Socket Address/IP info
if "Socket[address=" in line:
    SocketaddressColumn = line.find("Socket[address=")
    Socketaddressvalue = line.find(",",SocketaddressColumn, len(line))
    strSocketaddress = line[SocketaddressColumn+len("Socket[address="):Socketaddressvalue]
    strSocketaddress = strSocketaddress.replace("\\","")
# I give credit to dss.gov.au for information to find Email

# Find Emails
if re.findall(r'\w+[.\w]\w+@\w+[.\w]+', line):
    strEmail = sorted(set(re.findall(r'\w+[.\w]\w+@\w+[.\w]+', line)))  # remove duplicates from list

# Find URI
if "uri=htt" in line:
    SocketaddressColumn = line.find("uri=")
    Socketaddressvalue = line.find(",",SocketaddressColumn, len(line))
    strSocketaddress = line[SocketaddressColumn+len("uri="):Socketaddressvalue]

# Append Values
if (packname == package) and (strUser or strPassword or strPort or strlocalPort or strHost or strRecv_port or strRecv_host or strSend_port or strSend_host or strDeviceId or strNetworkOperatorName or strSimSerialNumber or strSimCountryIso or strSdcardReturnValue or strSocketaddress or strEmail):
    PackageName.append(packname)
    MethodName.append(pname)
    User.append(strUser)
    Password.append(strPassword)
    Port.append(strPort)
    localPort.append(strlocalPort)
    Host.append(strHost)
    Recv_port.append(strRecv_port)
    Recv_host.append(strRecv_host)
    Send_port.append(strSend_port)
    Send_host.append(strSend_host)
    DeviceId.append(strDeviceId)
    NetworkOperatorName.append(strNetworkOperatorName)
    SimSerialNumber.append(strSimSerialNumber)
    SimCountryIso.append(strSimCountryIso)
    SdcardReturnValue.append(strSdcardReturnValue)
    Socketaddress.append(strSocketaddress)
    Email.append(strEmail)
    URI.append(strUri)

else:
    print ('Please enter valid log file')

final_data = map(list, zip(PackageName,MethodName, Email, User, Password, Port, Host, Recv_port, Recv_host, Send_port, Send_host, localPort,Socketaddress, DeviceId, NetworkOperatorName, SimSerialNumber, SimCountryIso, SdcardReturnValue,URI))
o.s.remove(CSV)
out = open(CSV, 'w')
output = csv.writer(out)
for i in final_data :
    output.writerow(i)
import os.path
import sys, getopt, csv, re
from graphviz import Digraph
from subprocess import check_call

import pydot
from PIL import Image

csvfile = ''

def errorMsg():
    print ('The format should be')
    print ('python Graph_CrucialData.py -c <input csv filename> -p <packagename>\n')
    sys.exit()

# I give creadits to kasey.fr, sourceware, ethicalhackers for read and check commandline information

def errorMsg():
    print ('The format should be')
    print ('python Graph_CrucialData.py -c <input csv filename> -p <packagename>\n')
    sys.exit()

if os.path.isfile(csvfile) == True:
    with open(csvfile) as cfile:
        readCSV = csv.reader(cfile, delimiter=',',)

        dot = Digraph(comment='No Comments Here')
        dot.graph_attr['rankdir'] = 'LR'

        # Code
        if os.path.isfile(csvfile) == True:
            with open(csvfile) as cfile:
                readCSV = csv.reader(cfile, delimiter=',',)

                dot = Digraph(comment='No Comments Here')
                dot.graph_attr['rankdir'] = 'LR'
package = {}
packageseen = set()
MethodNameseen = []
Emailseen = []
Userseen = []
Passwordseen = []
Portseen = []
Hostseen = []
Recv_portseen = []
Recv_hostseen = []
Send_portseen = []
Send_hostseen = []
localPortseen = []
Socketaddressseen = []
DeviceIdseen = []
NetworkOperatorNameseen = []
SimSerialNumberseens = []
SimCountryIsoseen = []
SdcardReturnValueseen = []
URIseen = []

p=0

for itm in readCSV:
    if itm[0] not in packageseen:
        MethodNameseen.append([])
        Emailseen.append([])
        Userseen.append([])
        Passwordseen.append([])
        Portseen.append([])
        Hostseen.append([])
       Recv_portseen.append([])
       Recv_hostseen.append([])
        Send_portseen.append([])
        Send_hostseen.append([])
        localPortseen.append([])
        Socketaddressseen.append([])
        DeviceIdseen.append([])
        NetworkOperatorNameseen.append([])
        SimSerialNumberseens.append([])
        SimCountryIsoseen.append([])
        SdcardReturnValueseen.append([])
        URIseen.append([])

    packageseen.add(itm[0])
    package[itm[0]] = p
    dot.node('P%d' % p, label=itm[0], style="filled",
    fillcolor="lightyellow", shape="box")

    p += 1
if ((itm[1] not in MethodNameseen[package[itm[0]]]) and
       itm[1]):
    MethodNameseen[package[itm[0]]].append(itm[1])
if ((itm[2] not in Emailseen[package[itm[0]]]) and
       itm[2]):
    Emailseen[package[itm[0]]].append(itm[2])
if ((itm[3] not in Userseen[package[itm[0]]]) and itm[3]):
    Userseen[package[itm[0]]].append(itm[3])
if ((itm[4] not in Passwordseen[package[itm[0]]]) and
       itm[4]):
    Passwordseen[package[itm[0]]].append(itm[4])
if ((itm[5] not in Portseen[package[itm[0]]]) and itm[5]):
    Portseen[package[itm[0]]].append(itm[5])
if ((itm[6] not in Hostseen[package[itm[0]]]) and itm[6]):
    Hostseen[package[itm[0]]].append(itm[6])
if ((itm[7] not in Recv_portseen[package[itm[0]]]) and
       itm[7]):
    Recv_portseen[package[itm[0]]].append(itm[7])
if ((itm[8] not in Recv_hostseen[package[itm[0]]]) and
       itm[8]):
    Recv_hostseen[package[itm[0]]].append(itm[8])
if ((itm[9] not in Send_portseen[package[itm[0]]]) and
       itm[9]):
    Send_portseen[package[itm[0]]].append(itm[9])
if ((itm[10] not in Send_hostseen[package[itm[0]]]) and
       itm[10]):
    Send_hostseen[package[itm[0]]].append(itm[10])
if ((itm[11] not in localPortseen[package[itm[0]]]) and
       itm[11]):
    localPortseen[package[itm[0]]].append(itm[11])
and itm[12]):
    Socketaddressseen[package[itm[0]]].append(itm[12])
if ((itm[13] not in DeviceIdseen[package[itm[0]]]) and
       itm[13]):
    DeviceIdseen[package[itm[0]]].append(itm[13])
if ((itm[14] not in NetworkOperatorNameseen[package[itm[0]]]) and
       itm[14]):
    NetworkOperatorNameseen[package[itm[0]]].append(itm[14])
if ((itm[15] not in SimSerialNumberseen[package[itm[0]]]) and
       itm[15]):
    SimSerialNumberseen[package[itm[0]]].append(itm[15])
if ((itm[16] not in SimCountryIsoseen[package[itm[0]]]) and
       itm[16]):
    SimCountryIsoseen[package[itm[0]]].append(itm[16])
if ((itm[17] not in SdcardReturnValueseen[package[itm[0]]]) and
       itm[17]):
    SdcardReturnValueseen[package[itm[0]]].append(itm[17])
for i in range(0, p):
    labeldata = ''
    if MethodNameseen[i]:
        labeldata = labeldata + '<MethodName> MethodName|'
    if Emailseen[i]:
        labeldata = labeldata + '<Email> Email|'
    if Userseen[i]:
        labeldata = labeldata + '<User> User|'
    if Passwordse
    if Portseen[i]:
        labeldata = labeldata + '<Port> Port|'
    if Hostseen[i]:
        labeldata = labeldata + '<Host> Host|'
    if Recv_portseen[i]:
        labeldata = labeldata + '<Recv_port> Recv_port|'
    if Recv_hostseen[i]:
        labeldata = labeldata + '<Recv_host> Recv_host|'
    if Send_hostseen[i]:
        labeldata = labeldata + '<Send_port> Send_port|'
    if Send_hostseen[i]:
        labeldata = labeldata + '<Send_host> Send_host|'
    if localPortseen[i]:
        labeldata = labeldata + '<localPort> localPort|'
    if Socketaddressseen[i]:
        labeldata = labeldata + '<Socketaddress>'
    if DeviceIdseen[i]:
        labeldata = labeldata + '<DeviceId> DeviceId|'
    if NetworkOperatorNameseen[i]:
        labeldata = labeldata + '<NetworkOperatorName>'
    if SimSerialNumberseen[i]:
        labeldata = labeldata + '<SimSerialNumber>
    if SimCountryIsoseen[i]:
        labeldata = labeldata + '<SimCountryIso>
    if SdcardReturnValueseen[i]:
        labeldata = labeldata + '<SdcardReturnValue>
    if URIsseen[i]:
        labeldata = labeldata + '<URI> URI|'

    labeldata = labeldata[1:]

    dot.node('HeaderP%d' % i, label=labeldata,
              shape='record', style="filled", fillcolor="thistle1")
    dot.edge('P%d' % i, 'HeaderP%d' % i)
if MethodNameseen[i]:
    dot.node('Method%d' % i,
    label='|'.join(MethodNameseen[i]), shape='record')
    dot.edge('HeaderP%d:MethodName' % i, 'Method%d' % i)
if Emailseen[i]:
    dot.node('Email%d' % i, label='|'.join(Emailseen[i]),
    shape='record')
    dot.edge('HeaderP%d:Email' % i, 'Email%d' % i)
if Userseen[i]:
    dot.node('User%d' % i, label='|'.join(Userseen[i]),
    shape='record')
    dot.edge('HeaderP%d:User' % i, 'User%d' % i)
if Passwordseen[i]:
    dot.node('Password%d' % i, label='|'.join(Passwordseen[i]),
    shape='record')
    dot.edge('HeaderP%d:Password' % i, 'Password%d' % i)
if Portseen[i]:
    dot.node('Port%d' % i, label='|'.join(Portseen[i]),
    shape='record')
    dot.edge('HeaderP%d:Port' % i, 'Port%d' % i)
if Hostseen[i]:
    dot.node('Host%d' % i, label='|'.join(Hostseen[i]),
    shape='record')
    dot.edge('HeaderP%d:Host' % i, 'Host%d' % i)
if Recv_postseen[i]:
    dot.node('Recv_port%d' % i,
    label='|'.join(Recv_postseen[i]), shape='record')
    dot.edge('HeaderP%d:Recv_port' % i, 'Recv_port%d' % i)
if Recv_hostseen[i]:
    dot.node('Recv_host%d' % i,
    label='|'.join(Recv_hostseen[i]), shape='record')
    dot.edge('HeaderP%d:Recv_host' % i, 'Recv_host%d' % i)
if Send_hostseen[i]:
    dot.node('Send_port%d' % i,
    label='|'.join(Send_hostseen[i]), shape='record')
    dot.edge('HeaderP%d:Send_port' % i, 'Send_port%d' % i)
if Send_hostseen[i]:
    dot.node('Send_host%d' % i,
    label='|'.join(Send_hostseen[i]), shape='record')
    dot.edge('HeaderP%d:Send_host' % i, 'Send_host%d' % i)
if localPortseen[i]:
    dot.node('localPort%d' % i,
    label='|'.join(localPortseen[i]), shape='record')
    dot.edge('HeaderP%d:localPort' % i, 'localPort%d' % i)
if Socketaddressseen[i]:
    dot.node('Socketaddress%d' % i,
    label='|'.join(Socketaddressseen[i]), shape='record')
    dot.edge('HeaderP%d:Socketaddress' % i, 'Socketaddress%d' % i)
```
if DeviceIdseen[i]:
    dot.node('DeviceId%d' % i,
    label='|'.join(DeviceIdseen[i]), shape='record')
    dot.edge('HeaderP%d:DeviceId' % i, 'DeviceId%d' % i)
if NetworkOperatorNameseen[i]:
    dot.node('NetworkOperatorName%d' % i,
    label='|'.join(NetworkOperatorNameseen[i]), shape='record')
    dot.edge('HeaderP%d:NetworkOperatorName' % i, 'NetworkOperatorName%d' % i)
if SimSerialNumberseen[i]:
    dot.node('SimSerialNumber%d' % i,
    label='|'.join(SimSerialNumberseen[i]), shape='record')
    dot.edge('HeaderP%d:SimSerialNumber' % i, 'SimSerialNumber%d' % i)
if SimCountryIsoseen[i]:
    dot.node('SimCountryIso%d' % i,
    label='|'.join(SimCountryIsoseen[i]), shape='record')
    dot.edge('HeaderP%d:SimCountryIso' % i, 'SimCountryIso%d' % i)
if SdcardReturnValueseen[i]:
    dot.node('SdcardReturnValue%d' % i,
    label='|'.join(SdcardReturnValueseen[i]), shape='record')
    dot.edge('HeaderP%d:SdcardReturnValue' % i, 'SdcardReturnValue%d' % i)
if URIseen[i]:
    dot.node('URI%d' % i, label='|'.join(URIseen[i]), shape='record')
    dot.edge('HeaderP%d:URI' % i, 'URI%d' % i)

pnm = packnm.rsplit('.', 1)[-1]
fp = open('Output/Graph_%s_Crucialdata.dot'%pnm, 'w')
fp.write(dot.source)
fp.close()
check_call(['dot', '-Tpng', 'Output/Graph_%s_Crucialdata.dot'%pnm, '-o','Output/Graph_%s_Crucialdata.png'%pnm])
else:
    print ('Csv file not available')
```
import os.path
import subprocess
import sys, getopt, csv, re
from graphviz import Digraph
from numpy import array
from subprocess import check_call

import pydot
from PIL import Image

csvfile = ''

def insertNewlines(text, lineLength):
    if len(text) <= lineLength:
        return text
    else:
        return text[:lineLength] + '\n' +
        insertNewlines(text[lineLength:], lineLength)

####-------------------Definition to print error-------------------####
def errorMsg():
    print ('The format should be')
    print ('python Graph_MethodDetails.py -c <input csv filename> -p <packagename>\n')
    sys.exit()

# I give credit to kasey.fr, sourceware, ethicalhackers for read and check commandline information
###-------------------Check arguments-------------------###
try:
    opts, args =
    getopt.getopt(sys.argv[1:],"hc:p:v",["help","csv","package"])
except:
    errorMsg()

if len(opts):
    for opt, arg in opts:
        if opt in ('-h','--help'):
            print ('python Graph_MethodDetails.py -c <input csv filename> -p <packagename>\n')
            sys.exit()
        elif opt in ('-c','--csv'):
            csvfile = arg
        elif opt in ('-p','--package'):
            package = arg
        else:
            errorMsg()
else:
    errorMsg()}
MethodNameseen = []
methodindex = {}

dot = Digraph(comment='No Comments Here')
dot.graph_attr['rankdir'] = 'LR'
dot.node('P1', label=package, style="filled",
    fillcolor="lightyellow",
    shape="box")

readCSV = csv.reader(cfile, delimiter=',')

for item in readCSV:
    if (item[1] == package) and item[2]:
        MethodNameseen.append(item[2])

# counter duplicate lines
counts = collections.Counter(zip(MethodNameseen))
j = 0
for key, count in sorted(counts.items()):
    MethodNameseen = key
    dot.node('M%d' % j, label=''.join(MethodNameseen),
             style="filled", fillcolor="thistle1")
    dot.edge('P1', 'M%d' % j, label=str(count),
             fontcolor="red")
    j = j + 1

nm = package.rsplit('.', 1)[-1]
fp = open('Output/Graph_%s_MethodCount.dot' % nm, 'w')
fp.write(dot.source)
fp.close()

call_check(['dot', '-Tpng', 'Output/Graph_%s_MethodCount.dot' % nm, '-o',
            'Output/Graph_%s_MethodCount.png' % nm])
else:
    print 'Csv file not available'
import os.path
import subprocess
import sys, getopt, csv, re
from graphviz import Digraph
from numpy import array
from subprocess import check_call
from pydot import Image

csvfile = ''

def insertNewlines(text, lineLength):
    if len(text) <= lineLength:
        return text
    else:
        return text[:lineLength] + '\n' +
        insertNewlines(text[lineLength:], lineLength)

###---------------------Definition to print error---------------------###
def errorMsg():
    print ('The format should be')
    print ('python Graph_MethodDetails.py -c <input csv filename> -p <packagename>\n')
    sys.exit()

# I give credits to kasey.fr, sourceware, ethicalhackers for read and
check commandline information
###----------------Check arguments----------------###
try:
    opts, args =
    getopt.getopt(sys.argv[1:],"hc:p:v",["help","csv","package"])
except:
    errorMsg()

if len(opts):
    for opt, arg in opts:
        if opt in ('-h', '--help '):
            print ('python Graph_MethodDetails.py -c <input csv filename>
        -p <packagename>\n')
            sys.exit()
        elif opt in ('-c', '--csv '):
            csvfile = arg
        elif opt in ('-p', '--package '):
            package = arg
        else:
            errorMsg()}
else:
    errorMsg()

# Code
if os.path.isfile(csvfile) == True:
    with open(csvfile) as cfile:
        i=0
        k=0
        MethodNameseen = []
        methodindex = {}
        readCSV = csv.reader(cfile, delimiter=',',)
        for item in readCSV:
            if (item[1]==package) and (item[2] not in MethodNameseen):
                MethodNameseen.append(item[2])

            for j in range(len(MethodNameseen)):
                para = []
                retval = []

                with open(csvfile) as cfile1:
                    readCSV1 = csv.reader(cfile1, delimiter=',',)
                    for PKdata in readCSV1:
                            if PKdata[3] not in ('['',]'','[',','][',','][',','):
                                PKdata[3] = re.sub('/\*/\','',PKdata[3])
                                PKdata[3] = PKPKdata[3].replace("\n", " ")
                                if (PKdata[3] not in para):
                                    PKdata[3] = insertNewlines(PKdata[3],150)
                                    para.append(PKdata[3])

                            if PKdata[4] not in ('[]',']','[',',','][',','][',',') and
                             x<50:
                                PKdata[4] = re.sub('/\*/\','',PKdata[4])
                                if (PKdata[4] not in retval):
                                    PKdata[4] = insertNewlines(PKdata[4],150)
                                    retval.append(PKdata[4])
                                x+=1

                        else:
                            errorMsg()
```
dot = Digraph(comment='No Comments Here')
dot.graph_attr['rankdir'] = 'LR'
dot.node('P1', label=package, style="filled",
      fillcolor="lightyellow", shape="box")
if para:
    parametervalue.append(para)
    dot.node('Para%d' % j, label='|'.join(para),
      shape='record')
dot.edge('M%d' % j, 'Para%d' % j)
if retval:
    returnvalue.append(retval)
    dot.node('RetVal%d' % j, label='|'.join(retval),
      shape='record')
dot.edge('M%d' % j, 'RetVal%d' % j)
dot.node('M%d' % j, label=''.join(MethodNameseen[j]),
      shape='record', style="filled", fillcolor="thistle1")
dot.edge('P1', 'M%d' % j)

pnm = package.rsplit('.', 1)[-1]
fp = open('Output/%d_%s%s.dot%(j+1,pnm,MethodNameseen[j]), 'w')
fp.write(dot.source)
fp.close()
check_call(['dot', '-Tpng', 'Output/%d_%s%s.dot%(j+1,pnm,MethodNameseen[j]), '-o', 'Output/%d_%s%s.png%(j+1,pnm,MethodNameseen[j])'])
```

else:
    print ('Csv file not available')
APPENDIX H

Script to remove blanks from API call log

```python
#!/usr/bin/python
import sys, getopt
import os.path
outfile=''
logfile='

def exitMsg():
    print ('The format should be')
    print ('blank_remove.py -l <log unix path> -o <output name without
    any extenstion>
    sys.exit()

# I give creadits to kasey.fr, sourceware, ethicalhackers for read and
check commandline information
#-------------------Check arguments-------------------
try:
    opts, args = getopt.getopt(sys.argv[1:],"hl:o:","log=","out=")
except:
    exitMsg()
if len(opts):
    for opt, arg in opts:
        if opt == '-h':
            print ('python blank_remove.py -l <log unix path> -o
            <output name without any extenstion>
            sys.exit()
        elif opt in ('-l', '--log'):
            logfile = arg
        elif opt in ('-o','--out'):
            outfile = arg
        else:
            print ('Invalid Option')
            exitMsg()
else:
    exitMsg()
if (outfile and logfile):
    if os.path.exists(outfile):
        os.remove(outfile)
    if os.path.isfile(logfile) == True:
        f = open(logfile, "r")
        # omit empty lines and lines containing only whitespace
        lines = [line for line in f if line.strip()]
        f.close()
        lines.sort()
        f = open(outfile, "w")
        f.writelines(lines)
    else:
        print ('Please enter valid log file')
else:
    print ('Please enter valid commandline arguments')
    exitMsg()```
APPENDIX I

Script to remove duplicates from API call log

```python
#!/usr/bin/python
import sys, getopt
import os.path
import fileinput

outfile=''
log=''

def exitMsg():
    print 'The format should be'
    print 'RemoveDuplicate.py -l <log unix path> -o <output name without any extension>
    sys.exit()

# I give credits to kasey.fr, sourceware, ethicalhackers for read and check commandline information
#------------------Check arguments---------------------#
try:
    opts, args = getopt.getopt(sys.argv[1:],"hl:o:",["log=","out="]) except: exitMsg()

if len(opts):
    for opt, arg in opts:
        if opt == '-h':
            print 'python RemoveDuplicate.py -l <log unix path> -o <output name without any extension>
            sys.exit()
        elif opt in ('-l', '--log'):
            log = arg
        elif opt in ('-o', '--out'):
            outfile = arg
        else:
            exitMsg()
else:
    exitMsg()

##-- remove previous output file if exist --##
if os.path.exists(outfile) :
    os.remove(outfile)

if (outfile and log):
    if os.path.isfile(log) == True :
        # Remove blank and duplicate line from File
        r = set()
        for line in fileinput.FileInput(log):
            if line.rstrip():
                r.add(line)
```
out1 = open(outfile, 'w').writelines(r)
else:
    print 'RemoveDuplicate.py : Please enter valid log file'
else:
    print 'RemoveDuplicate.py : Please enter valid commandline arguments'
    exitMsg()
APPENDIX J

Sample DOT file

// No Comments Here
digraph {
    rankdir=LR;
    node [ shape = record]

    M0 [label=startService fillcolor=thistle1 style=filled]
    V0 [label="MyService"]
    M0->V0

    M1 [label="query" fillcolor=thistle1 style=filled]
    V1 [label="content:com.android.contacts/contacts"]
    M1->V1

    M2 [label="<M2_1>getColumnIndex | <M2_2>getString"
        fillcolor=thistle1 style=filled]
    V2 [label="id, display_name, photo_uri, phone_number"]
    M2:M2_1->V2
    V2 -> M2:M2_2

    M3 [label="openOrCreateDatabase" fillcolor=thistle1 style=filled]
    V3 [label="contactsManager"]
    M3->V3

    M4 [label="execSQL" fillcolor=thistle1 style=filled]
    V4 [label="CREATE TABLE contactTable (id INTEGER, phone_number
            TEXT, email TEXT, name TEXT);"]
    M4->V4

    M5 [label="insert" fillcolor=thistle1 style=filled]
    V5 [label="contactTable ( phone_number, email, name )"]
    M5->V5

    M6 [label="rawQuery" fillcolor=thistle1 style=filled]
    V6 [label="<V6_1> SELECT * FROM contactTable; | <V6_2>
        Columns=id,phone_number,email,name"]
    M6->V6:V6_1
    V6:V6_2->M6

    M7 [label="getDefaultCommandMap | addMailcap" fillcolor=thistle1
        style=filled]
    V7 [label="text\html; text\xml; text\plain; multipart\*;
        message\rfc822;"]
    M7->V7
M8 [label="getInstance" fillcolor=thistle1 style=filled]
V8 [label="<V8_1> Properties: mail.smtp.port=465, smtp.host=smtp.gmail.com, smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory, smtp.socketFactory.port=465 \n GmailClient[user=arpitatushit27513@gmail.com,host=smtp.gmail.com,password=arpitatushitshah,port=465,smtp=465] |<V8_2> addressMap/rfc822=smtp,authenticator=com.project.fetch.contactdetails.GmailClient, \n Providers: com.sun.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun Microsystems],\n com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider[STORE,smtp,com.sun.mail.pop3.POP3SSLStore,Sun Microsystems],\n com.sun.mail.pop3.POP3Store=javax.mail.Provider[STORE,smtp,com.sun.mail.pop3.POP3Store,Sun Microsystems],\n com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Sun Microsystems],\n com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,smtp,com.sun.mail.imap.IMAPSSLStore,Sun Microsystems]"]
M8->V8:V8_1
V8:V8_2->M8

M9 [label="setFrom" fillcolor=thistle1 style=filled]
V9 [label="kartiki.aditi@gmail.com"]
M9->V9

M10 [label="setRecipients" fillcolor=thistle1 style=filled]
V10 [label="arpitatushit27513@gmail.com"]
M10->V10

M11 [label="setSubject" fillcolor=thistle1 style=filled]
V11 [label="All contacts you will get .. :P"]
M11->V11

M12 [label="setSentDate" fillcolor=thistle1 style=filled]
V12 [label="Current Date Time"]
M12->V12

M13 [label="setText" fillcolor=thistle1 style=filled]
V13 [label="Contact Details
Id1 Phone Number Name Email"]
M13->V13

M14 [label="send" fillcolor=thistle1 style=filled]
M15 [label="closeSocket" fillcolor=thistle1 style=filled]

P1 [label="com.project.fetch.contactdetails" fillcolor=lightyellow shape=doubleoctagon style=filled]
P1 -> M0
P1 -> M1
P1 -> M2
P1 -> M3
P1 -> M4
P1 -> M5
P1 -> M6
P1 -> M7
P1 -> M8
P1 -> M9
P1 -> M10
P1 -> M11
P1 -> M12
P1 -> M13
P1 -> M14
P1 -> M15
}
REFERENCES


