Mobile Security: Android

Saurabh Kumar
Senior Research Scholar
IIT Kanpur
Date: 09/03/2022

https://github.com/skmtr1/Workshop-Mobile-Forensics-And-Security
MOTIVATION
Why Mobile Security?

- User activity
- Valuable data
- Always on
- Multiple Attack Surfaces
Why Android?

1. Almost completely open source

2. Global smartphone market share

<table>
<thead>
<tr>
<th>Period</th>
<th>Android</th>
<th>iOS</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020</td>
<td>84.1%</td>
<td>15.9%</td>
<td>0%</td>
</tr>
<tr>
<td>2021</td>
<td>83.8%</td>
<td>16.2%</td>
<td>0%</td>
</tr>
<tr>
<td>2022</td>
<td>84.1%</td>
<td>15.9%</td>
<td>0%</td>
</tr>
<tr>
<td>2023</td>
<td>84.4%</td>
<td>15.6%</td>
<td>0%</td>
</tr>
<tr>
<td>2024</td>
<td>84.7%</td>
<td>15.3%</td>
<td>0%</td>
</tr>
<tr>
<td>2025</td>
<td>84.9%</td>
<td>15.1%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Source: International Data Corporation (IDC), October 2021
Actors in the Android Ecosystem

- Tool chain
  (Cordova, App generator, …)
- App Developer
- Publish app
- Sideloading
- Play Store
- Alternate Store
- Administrators
- Platform vendors
- Online services
- Advertisement networks
- Third Party app
  - Ad Libs
  - Application Framework
  - Native libs (C / C++)
  - Android Runtime (Dalvik / ART)
  - Linux Kernel (modified)
## Security Impact of an Actor Over Others

<table>
<thead>
<tr>
<th>Actor</th>
<th>OS Developer</th>
<th>H/W Vendor</th>
<th>Library Providers</th>
<th>S/W Developer</th>
<th>Toolchain Providers</th>
<th>S/W Publisher</th>
<th>S/W Market</th>
<th>End User</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS Developer</td>
<td>--</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>H/W Vendor</td>
<td>None</td>
<td>--</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
</tr>
<tr>
<td>Library Provider</td>
<td>None</td>
<td>None</td>
<td>--</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
</tr>
<tr>
<td>S/W Developer</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>--</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
</tr>
<tr>
<td>Toolchain Providers</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>--</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
</tr>
<tr>
<td>S/W Publisher</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>Partial</td>
<td>None</td>
<td>--</td>
<td>Partial</td>
<td>Full</td>
</tr>
<tr>
<td>S/W Market</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>Partial</td>
<td>None</td>
<td>None</td>
<td>--</td>
<td>Full</td>
</tr>
<tr>
<td>End User</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>--</td>
</tr>
</tbody>
</table>

Legend:
- Full: Full impact
- Partial: Partial impact
- None: No impact
Where to Improve Security?

- Tool Chain Provider
  - Uses tool chains
  - Includes libraries
- App developer
  - Publish App
- Markets
  - Install
- Installed Apps
  - Android API
  - Middleware
  - Linux API
  - Linux Kernel
  - Use via internet
- Web Services
  - Useable Security
- 3rd party libraries
  - Retrofit Android’s Security
- Ad & Analytics Network
  - Application Vetting
  - Inline reference monitors
Motivation: Summary

- Feature-rich smartphones and appification have induced security research on various new aspects.

- Android’s market share has made Android the #1 target for malware authors and makes improved security & privacy mechanisms imperative.

- Various actors in the ecosystem with (strong) influence on security and privacy.
ANDROID BACKGROUND
Android Software Stack

Default apps
- Contacts
- SMS

Third party apps
- paytm
- linkedin

Application Framework

Native libs
(C / C++)

Android Runtime
(Dalvik / ART)

Linux Kernel (modified)
Application Packages (APK)

- APK is simply a packaging format like JAR, ZIP and TAR
- Component of Application
  - Activity
  - Content Provider
  - Services
  - Broadcast Receiver
- Native Code (C/C++ shared libraries)
- Resources
- META-INF
- Application Manifest
ANDROID SECURITY ARCHITECTURE

- Package Integrity
- Sandboxing
- Permission and Least Privilege
Package Integrity: Package Manifest

- Created with jarsigner
- META-INF
  - Manifest.mf, Cert.sf, Cert.{RSA,DSA}

File

Manifest.mf

<table>
<thead>
<tr>
<th>Manifest-Version: 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-By: Generated-by-ADT</td>
</tr>
<tr>
<td>Created-By: Android Gradle 3.0.1</td>
</tr>
</tbody>
</table>

| Name: res/mipmap-hdpi-v4/ic_launcher.png |
| SHA1-Digest: 2zkIQdtvIXqEHSTVOVuwBQ18als= |

Cert.sf

<table>
<thead>
<tr>
<th>Signature-Version: 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created-By: 1.0 (Android)</td>
</tr>
<tr>
<td>SHA1-Digest-Manifest: h9xNlIN3bQiT8RQyPUWBojRKD8=</td>
</tr>
<tr>
<td>X-Android-APK-Signed: 2</td>
</tr>
</tbody>
</table>

| Name: res/mipmap-hdpi-v4/ic_launcher.png |
| SHA1-Digest: L8RpX5x8pChJbucqml+hMt9D9CQ= |

Certificate Cert.sf signature

CERT.{RSA,DSA}
Verifying of package manifest

Chain of trust:

PKI

Package certificate in Cert.{RSA, DSA}

Manifest.mf

Cert.sf

Files
ANDROID SECURITY ARCHITECTURE

- Package Integrity
- Sandboxing
- Permission and Least Privilege
Sandboxing

- The application sandbox specifies which system resources the application is allowed to access.
- An attacker can only perform actions defined in the sandbox.
Application Isolation by Sandboxing

- Each Application is isolated in its own environment
  - Applications can access only its own resources
  - Access to sensitive resources depends on the application’s rights
- Sandboxing is enforced by Linux
Application sandbox

- Isolation: Each installed App has a separate user ID

Diagram:
- App Code (Classes.dex)
- Core libraries
- JNI
- Native Code (*.so)
- syscalls
- Kernel
- UID A
Isolation: Each installed App has a separate user ID
Each App lives in its own sandbox
ANDROID SECURITY ARCHITECTURE

- Package Integrity
- Sandboxing
- Permission and Least Privilege
Android Permission System

- **Access rights** in Android’s application framework
  - Permissions are required to **gain** access to
    - System interfaces (Internet, send SMS, etc.)
    - System resources (logs, battery, etc.)
    - Sensitive data (SMS, contacts, etc.)
  - Currently more than 140 default permissions defined in Android

- Permissions are **assigned** to sandbox

- Application developers can also **define** their **own** permissions
Android Permission: Example

App B
(has permission P)

App C
(has not permission P)

App A
(Service
(Permission P))
Permissions’ Protection Level

- Normal
- Dangerous
- Signature
- SignatureOrSystem
Dynamic Permissions (≥ Android 6.0)

- App developers must check if their apps hold required dangerous permission, otherwise request them at runtime.
- User can grant permissions at runtime and also revoke once granted permissions again.

Is the requested permission reasonable?

Should I adjust some permissions?
ANDROID VULNERABILITIES

- Architecture Based
- Software Based
- Hardware Based
Vulnerability Classification

Android Vulnerability

Architecture

Software

Hardware

Operating System

Original Equipment Manufacturer (OEM)

Third Party App
ANDROID VULNERABILITIES

- Architecture Based
- Software Based
- Hardware Based
Application-Level Privilege Escalation Attacks

Confused Deputy Attack

Malicious App + Confused Deputy App = Confused Deputy Attack

Collusion Attack

Malicious App + Malicious App = Collusion Attack
Collusion Attack

Malicious apps **collude** in order to **merge** their respective **permissions**

- **Variants:**
  - Apps communicate directly
  - Apps communicate via covert channels in Android
ANDROID VULNERABILITIES

• Architecture Based
• Software Based
• Hardware Based
Dirty COW

- Existed in the Linux Kernel for 9 years
- A **local** Privilege Escalation Vulnerability
- Exploits a race condition in the implementation of the *copy-on-write* mechanism
- Turns a **read-only** mapping of a file into a writable mapping

Android malware ZNIU exploits DirtyCOW vulnerability
Media Projection Service Issue

Vulnerabilities

Android issue allows attackers to capture screen and record audio on 77% of all devices

Over-privileged Apps

- Many apps request permissions that their **functionality** does not require
- Suspected root cause: API **documentation/naming** convention
  - Solution: API Permissions Maps
    - Can be integrated into lint tools
A privileged app is fooled into misusing its privileges on behalf of another (malicious) unprivileged app.

Example:

- Unauthorized phone calls
- Various confused deputies in system apps
Confused Deputy Introduce by OEMs

- Several confused deputies found in Samsung devices’ firmware
  - One deputy running with system privileges provided root shell service to any app

![Diagram showing various permissions connected to a backdoor]

- Access to mail account
- SMS & MMS
- Internet
- Contacts
- Access to SD card
- Camera
- Microphone
- GPS Location
Android Vulnerabilities

- Architecture Based
- Software Based
- Hardware Based
Broadcom Wi-Fi SoC Flaw

Android devices can be fatally hacked by malicious Wi-Fi networks

Broadcom chips allow rogue Wi-Fi signals to execute code of attacker's choosing.

DAN GOODIN - 4/6/2017, 1:16 AM

MALWARE ANALYSIS
WHY MALWARE ANALYSIS?

This data-stealing Android malware infiltrated the Google Play Store, infecting users in 196 countries
At least 100,000 people downloaded apps distributing MobSTSPY malware, which also leverages a phishing

First Android Clipboard Hijacking Crypto Malware Found On Google

Android banking malware hitting more users than ever

By Anthony Spadafora 22 days ago

Fake banking apps could be more effective than banking Trojans

Several Popular Beauty Camera Apps Caught Stealing Users’ Photos


Source: https://thehackernews.com/2019/02/beauty-camera-android-apps.html
In every 10 seconds, a new Android malware is born.

New Android malware samples per year

<table>
<thead>
<tr>
<th>Year</th>
<th>Millions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>0.94</td>
</tr>
<tr>
<td>2014</td>
<td>1.02</td>
</tr>
<tr>
<td>2015</td>
<td>2.57</td>
</tr>
<tr>
<td>2016</td>
<td>6.13</td>
</tr>
<tr>
<td>2017</td>
<td>6.20</td>
</tr>
<tr>
<td>2018</td>
<td>5.54</td>
</tr>
<tr>
<td>2019</td>
<td>3.20</td>
</tr>
<tr>
<td>2020</td>
<td>3.13</td>
</tr>
<tr>
<td>2021</td>
<td>3.39</td>
</tr>
</tbody>
</table>

Source: AV-TEST malware statistics report Jan 2022
Analysis Techniques

- Static
- Hybrid
- Dynamic
Malware Analysis

- Many work has been proposed
- Deployed on
  - Server
  - Real Device
- Offline analysis can be bypassed
- On a real device, existing offline method cannot be used
  - High resources requirement
Challenges: Dynamic Analysis
Android Emulator

- A virtual mobile device
- Use Case:
  - Prototype, develop and test an application
  - Dynamic Analysis of malware
    - Used by security companies
Emulation-Detection

- Detection methods are classified in 5 categories:
  - Unique Device Information (basic and smart)
  - Sensors Reading
  - GPS Information
  - Device State Information
  - Distributed Detection
## Unique Device Information

### Basic
- Unrealistic/null value for IMEI, Phone No. etc.

<table>
<thead>
<tr>
<th>IMEI</th>
<th>Phone No.</th>
<th>ICCID</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456XXXXX2347</td>
<td>901XXXXX36</td>
<td>89XXXXX5611117910720</td>
</tr>
<tr>
<td>null/00000000000</td>
<td>155XXXXX554</td>
<td>89XXXXX3211118510720</td>
</tr>
</tbody>
</table>

### Smart
- Realistic but fixed values

<table>
<thead>
<tr>
<th>IMEI</th>
<th>Phone No.</th>
<th>ICCID</th>
</tr>
</thead>
<tbody>
<tr>
<td>3514XXXXX401216</td>
<td>972XXXXX243</td>
<td>89XXXXX0082067415160</td>
</tr>
<tr>
<td>3514XXXXX401216</td>
<td>972XXXXX243</td>
<td>89XXXXX0082067415160</td>
</tr>
</tbody>
</table>
Sensors

- Different sensors in a smart phone
  - Motion Sensors: accelerometer, gyroscope
  - Environmental Sensors: illumination (light), humidity

- Detection:
  - Count: At least 6-7 or more sensors in a smartphone
  - Reading: No change in sensors reading
GPS Information

- No change in GPS location

- Use of mock location API to provide fake location

- No correlation with BTS geo-location
Device State Information

- Smartphone state may change due to:
  - Battery power
  - Signal Strength
  - SMS
  - Call
- No state change in emulated platform
Distributed Detection

- Detection on server
  - App communicates with server
  - Observing identical information for multiple device like IMEI

- Example:
  - Botnet analysis

<table>
<thead>
<tr>
<th>Client No.</th>
<th>IMEI</th>
<th>Emulated?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client-1</td>
<td>IMEI-1</td>
<td>×</td>
</tr>
<tr>
<td>Client-2</td>
<td>IMEI-2</td>
<td>×</td>
</tr>
<tr>
<td>Client-3</td>
<td>IMEI-3</td>
<td>×</td>
</tr>
<tr>
<td>Client-4</td>
<td>IMEI-3</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Existing Frameworks Evaluation

<table>
<thead>
<tr>
<th>Detection Type</th>
<th>Emulator</th>
<th>DroidBox</th>
<th>CuckooDroid</th>
<th>MobSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique ID (Basic)</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Unique ID (Smart)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Sensors reading</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Device State</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>GPS</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Distributed Detection</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Every framework fails to defend against all the detection method except for basic unique ID
Summary: Emulation Detection

- Existing framework fails to defend against detection method:
  - Smart unique device information
  - Sensors and GPS information
  - Device state
  - Distributed detection

- Need a robust anti-emulation-detection system:
  - Hides underline emulated platform
  - Remain undetected when attack is performed from any layer
Reference for More Details

- Robust Anti-Emulation-Detection
  [Link](https://www.youtube.com/watch?v=ahAgW4Wj3qc)

- On-Device Android Malware Detection
  [Link](https://www.youtube.com/watch?v=ziwIJGttkYg)
Case Study: Analysis of Pegasus Malware
Pegasus: Attack Vector and Capabilities

Attack Vector:
- Email
- SMS
- Web Browsing
- Social Media
- Unknown Vulnerability

Capabilities:
- Email
- SMS
- Photos
- Location data
- Contacts
- Activate microphone
- Activate camera
- Record calls
- Calendar
- Social media chat
Data Collection

- Samples were collected from CloudSek
- Total 5 Apps
- App-1 and App-3 are same only file name is different

<table>
<thead>
<tr>
<th>App ID</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>App-1</td>
<td>9fae5d148b89001555132c896879652fe1ca633d35271db34622248e048c78ae.apk</td>
</tr>
<tr>
<td>App-2</td>
<td>144778790d4a43a1d93d6b660a6acb3a6d37a19e6a6f0a6bf1ef47e919648e.apk</td>
</tr>
<tr>
<td>App-3</td>
<td>cc9517aaf58279091ac17533293edc1.apk</td>
</tr>
<tr>
<td>App-4</td>
<td>d257cfde7599f4e20ee08a62053e6b3b936c87d373e6805f0e0c65f1d39ec320.apk</td>
</tr>
<tr>
<td>App-5</td>
<td>bd8cda80aaee3e4a17e9967a1c062ac5c8e4aefd7eaa3362f54044c2c94db52a.apk</td>
</tr>
</tbody>
</table>
Analysis Type and Environment

- **Static**
  - Androguard

- **Dynamic**
  - STDNeut: Neutralizing Sensor, Telephony System and Device State Information on Emulated Android Environments
  - Xposed framework to monitor API calls
  - SysCallMon: A system call monitoring Kernel module
Analysis Result
App-1 and App-3

Meta Information
• Package Name: com.binary.sms.receiver
• Modification Date: 2 June, 2014
• Hash: 9fae5d148b89001555132c896879652fe1ca633d35271db34622248e048c78ae

Server Communication

<table>
<thead>
<tr>
<th>IP/URLs</th>
<th>Port</th>
<th>Geo Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>142.XXX.27.188</td>
<td>443</td>
<td>Mountain View, California, USA</td>
</tr>
<tr>
<td></td>
<td>5228</td>
<td></td>
</tr>
</tbody>
</table>
App-1 and App-3 cont..

System Command
• chmod, mount, su

Capability
• Install new applications
• Make a call, listen or record incoming/outgoing call
• Read/Write contacts, bookmark,
• Many more…
App-1 and App-3 cont..

Observation:
• Tries to get root privilege
• Change file permissions
• Mount system partition as R/W
• Intercept incoming/outgoing SMS and Calls
• Obtain information about installed and running apps
• Can install new apps
• Read other information like contacts, history bookmarks,
• Read/write system settings,
• Process outgoing calls and send new SMS
• Delete call logs and many more.
App-2

Meta Information
• Package Name: com.lenovo.safecenter
• Modification Date: 16 Dec, 2010
• Hash: 144778790d4a43a1d93dff6b660a6acb3a6d37a19e6a6f0a6bf1ef47e919648e

Server Communication

<table>
<thead>
<tr>
<th>IP/URLs</th>
<th>Port</th>
<th>Geo Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>142.XXX.102.188</td>
<td>443</td>
<td>Mountain View, California, USA</td>
</tr>
<tr>
<td></td>
<td>5228</td>
<td></td>
</tr>
<tr>
<td>142.XXX.5.188</td>
<td>443</td>
<td>Mountain View, California, USA</td>
</tr>
<tr>
<td></td>
<td>5228</td>
<td></td>
</tr>
</tbody>
</table>
App-2 cont..

System Command
- app_process, bind, cat, chmod, chown, close, connect, date, dumpsys, echo, exit, gzip, id, iptables, kill, log, logcat, ls, mkdir, mount, mv, notify, open, pm, ps, pwd, read, reboot, sdcard, select, service, sh, socket, start, su, system_server, times, uptime, write

Capability
- Make a call, send new SMS
- Read/Write contacts, system settings,
- Process outgoing calls
- Access location data
- Kill background processes
- Many more…
Observation:
- Capable to bypass dynamic analysis using device information
- Tries to get root privilege
- Can change files permission
- Mount system partitions as RW
- Open network sockets
- Get running process information and kill any process
- Dynamically load code, end an incoming call, kill background processes
- Remove any app
- Register a broadcast receiver to intercept incoming SMS
App-4

Meta Information
• Package Name: com.xxGameAssistant.pao
• Modification Date: 15 Nov, 2013
• Hash: d257cfde7599f4e20ee08a62053e6b3b936c87d373e6805f0e0c65f1d39ec320

System Command
• Chmod, dd, ln, mkdir, mount, stop, su

Capability
• Read Phone state
• Access location data
• Listen to boot complete event
• Read/Write to external storage
Server Communication

<table>
<thead>
<tr>
<th>IP/URLs</th>
<th>Port/Protocol</th>
<th>Geo Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://tdcv3.talkingdata.net/g/d">http://tdcv3.talkingdata.net/g/d</a></td>
<td>HTTP</td>
<td>Kansas City, Missouri, USA</td>
</tr>
<tr>
<td><a href="http://tdcv3.talkingdata.net">http://tdcv3.talkingdata.net</a></td>
<td>DNS</td>
<td></td>
</tr>
<tr>
<td>35.XXX.63.213</td>
<td>--</td>
<td>Mountain View, California, USA</td>
</tr>
<tr>
<td>142.XXX.188.196</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>142.XXX.102.188</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>142.XXX.27.188</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>142.XXX.5.188</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>
App-4 cont..

Observation:
- Tries to get root privilege
- Can change files permission
- Mount system partitions as R/W
- Capable to bypass dynamic analysis using device & CPU information
- Can install apps,
- Get information about the currently installed/running App, processes and tasks
- Track location and steal sensitive information like device Ids, phone numbers and others
- Listen to BOOT COMPLETE event so that it can run a code or background process when a phone restarts.
App-5

Only static analysis
• Dex file is tempered, hence no dynamic analysis

Meta Information
• Package Name: sec.dujmehn.qdtheyt
• Modification Date: 10 Nov, 2018 based on last modified content
• Hash: bd8cda80aaee3e4a17e9967a1c062ac5c8e4ae7ea3362f54044c2c94db52a
App-5 cont..

Capability
• Install new applications
• Make a call, listen or record incoming/outgoing call
• Read/Write contacts, bookmark,
• Access to location data
• Send and read SMS
• Kill background process
• Set fake location information
• Many more…
App-5 cont..

Observation:
• Can change files permissions
• Mount system partitions as R/W.
• Can get information about currently installed/running apps, processes and tasks
• Track location and steal sensitive information like device IDs, phone numbers and others.
• Listen to BOOT COMPLETE, NEW SMS, OUTGOING CALLS, BATTERY STATUS CHANGED, and many other events
  • Can run a code or background process when any of such event occurs
• Ability to change system configuration, R/W contacts, bookmark history,
• Record audio in background, install apps
Connection Between Apps

App-2

App-1
App-3

App-5

142.XXX.102.188, 142.XXX.5.188

142.XXX.27.188

35.XXX.63.213, 142.XXX.188.196

tdcv3.talkingdata.net

App-4
Detection of Pegasus

- Used DeepDetect, machine learning based Android malware detector
- Static features from Manifest File and Dex code
- Results

<table>
<thead>
<tr>
<th>App ID</th>
<th>Detection Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>App-1</td>
<td>✓</td>
</tr>
<tr>
<td>App-2</td>
<td>✓</td>
</tr>
<tr>
<td>App-3</td>
<td>✓</td>
</tr>
<tr>
<td>App-4</td>
<td>✓</td>
</tr>
<tr>
<td>App-5</td>
<td>✓ (Only based on Android Manifest file)</td>
</tr>
</tbody>
</table>
https://github.com/skmtr1/Workshop-Mobile-Forensics-And-Security
Thank You