

Mobile threats incident handling

Toolset, Document for students





About ENISA

The European Union Agency for Network and Information Security (ENISA) is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. More information about ENISA and its work can be found at www.enisa.europa.eu.

Acknowledgements

Contributors to this report

We would like to thank all our ENISA colleagues who contributed with their input to this report and supervised its completion, especially Lauri Palkmets, Cosmin Ciobanu, Andreas Sfakianakis, Romain Bourgue, and Yonas Leguesse. We would also like to thank the team of Don Stikvoort and Michael Potter from S-CURE, The Netherlands, Mirosław Maj and Tomasz Chlebowski from ComCERT, Poland, and Mirko Wollenberg from PRESECURE Consulting, Germany, who produced the second version of this documents as consultants.

Agreements or Acknowledgements

ENISA wants to thank all institutions and persons who contributed to this document. A special 'Thank You' goes to the following contributors: Anna Felkner, Tomasz Grudzicki, Przemysław Jaroszewski, Piotr Kijewski, Mirosław Maj, Marcin Mielniczek, Elżbieta Nowicka, Cezary Rzewuski, Krzysztof Silicki, Rafał Tarłowski from NASK/CERT Polska, who produced the first version of this document as consultants and the countless people who reviewed this document.

Contact

For contacting the authors please use CERT-Relations@enisa.europa.eu

For media enquires about this paper, please use press@enisa.europa.eu.



Legal notice

Notice must be taken that this publication represents the views and interpretations of the authors and editors, unless stated otherwise. This publication should not be construed to be a legal action of ENISA or the ENISA bodies unless adopted pursuant to the Regulation (EU) No 526/2013. This publication does not necessarily represent state-of-the-art and ENISA may update it from time to time.

Third-party sources are quoted as appropriate. ENISA is not responsible for the content of the external sources including external websites referenced in this publication.

This publication is intended for information purposes only. It must be accessible free of charge. Neither ENISA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this publication.

Copyright Notice

© European Union Agency for Network and Information Security (ENISA), 2013

Reproduction is authorised provided the source is acknowledged.



Table of Contents

| | | |
|----------|--|-----------|
| 1 | What Will You Learn | 1 |
| 2 | Exercise Task | 1 |
| 2.1 | Task 1: Familiarisation with Android, AVD, and ADB | 1 |
| 2.2 | Task 2: Cloning an Application | 8 |
| 2.3 | Task 3: Analysing Cloned Application | 14 |
| 2.4 | Task 4: Analysing Simplelocker | 16 |
| 2.5 | Task 5 (Optional): Analysing Other Artifacts | 16 |
| 3 | Conclusion | 18 |

1 What Will You Learn

Mobile devices add some additional conditions for the investigator. First, access to the device might be difficult (geography, Bring Your Own Device (BYOD), and other privacy issues). Data access and investigation tools used in other environments might not be working. It might be necessary to adjust implemented incident-handling processes to adapt to these specific platforms. This course will serve as an introduction to Mobile Technologies, and Incident Handling within this field.

2 Exercise Task

The Students will be given 5 main tasks:

1. Familiarisation with Android, AVD, and ADB
2. Cloning an Application
3. Analysing Cloned Application
4. Analysing Simplelocker
5. (Optional) Analysing Other Artifacts

2.1 Task 1: Familiarisation with Android, AVD, and ADB

2.1.1 Open Android SDK Manager

- `cd /android/adt-bundle-linux-x86-20140702/sdk/tools`
- `./android`

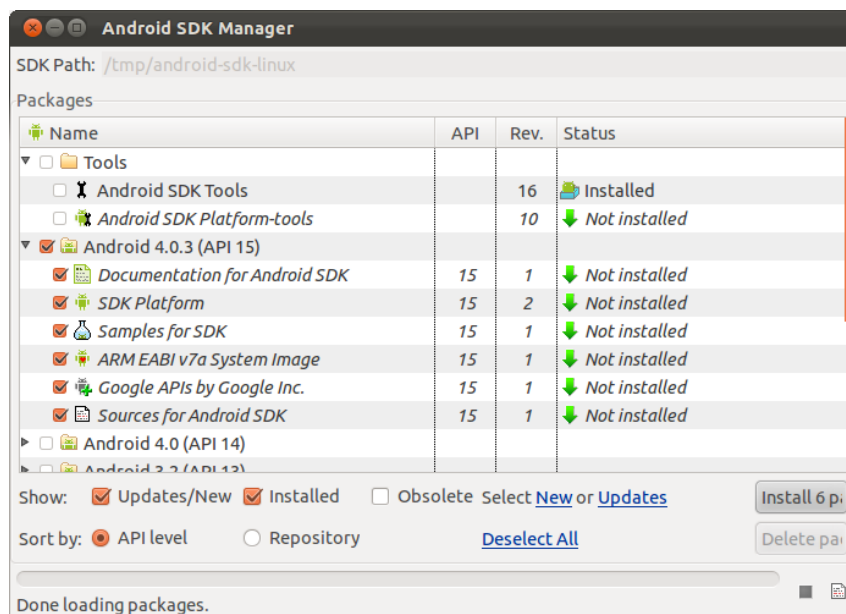


Figure 1: Android SDK Manager

2.1.2 Open Android AVD Manager

- `cd /android/adt-bundle-linux-20140702/sdk/tools/`
- `android avd`

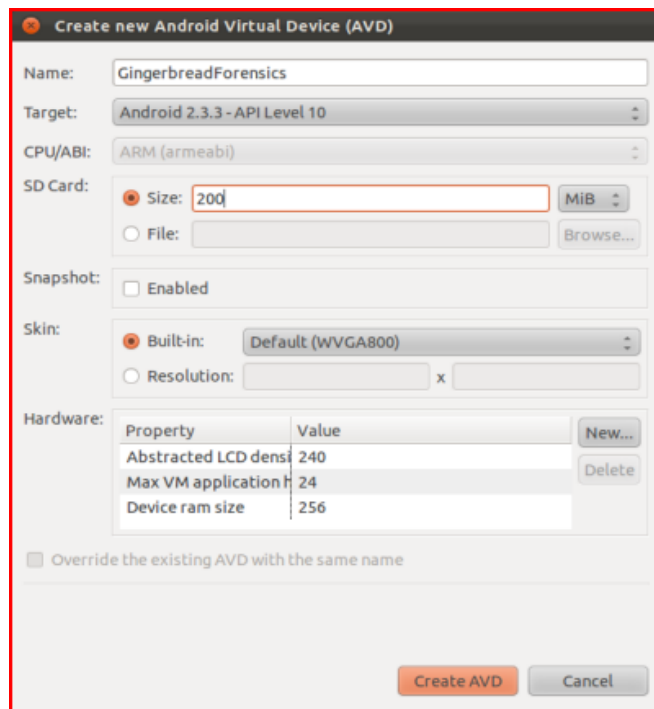


Figure 2: Android AVD Manager

2.1.3 Browse /home/enisa/.android

| Desktop OS | AVD Data Location |
|------------|------------------------------|
| Ubuntu | /home/<username>/.android/ |
| Mac OS X | /Users/<username>/.android |
| Windows 7 | C:\Users\<username>\.android |

2.1.4 Browse Enisa AVD folder

- **cache.img**: disk image of /cache partition
- **sdcard.img**: disk image of SD card (if created during AVD setup)
- **userdata-qemu.img**: disk image of /data partition

2.1.5 Create an AVD named “SimpleLockerAVD” using avd manager

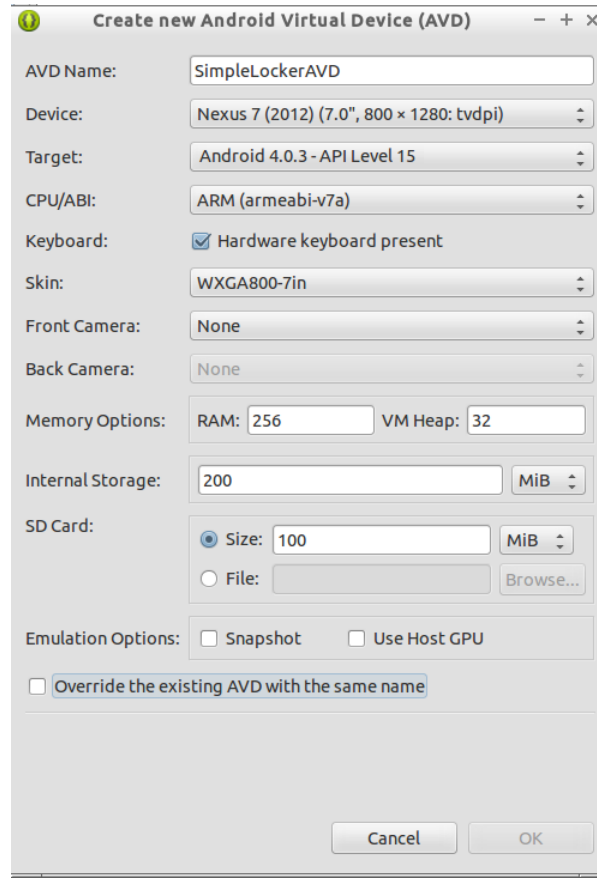


Figure 3: Create AVD

2.1.6 Add Hardware and open AVDs

- Modify Nexus 7 Hardware by Cloning Device and modifying the features
- Add GPS to Enisa AVD (Edit) using AVD Manager

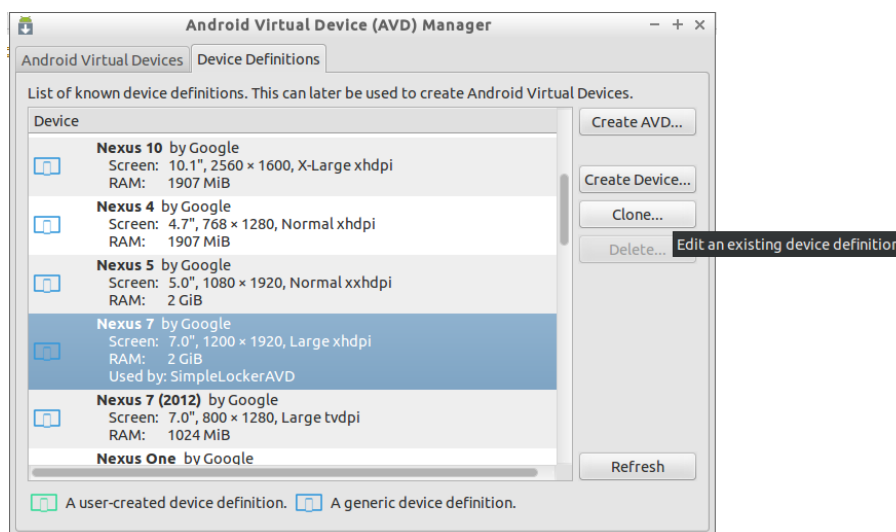


Figure 4: Device Definitions

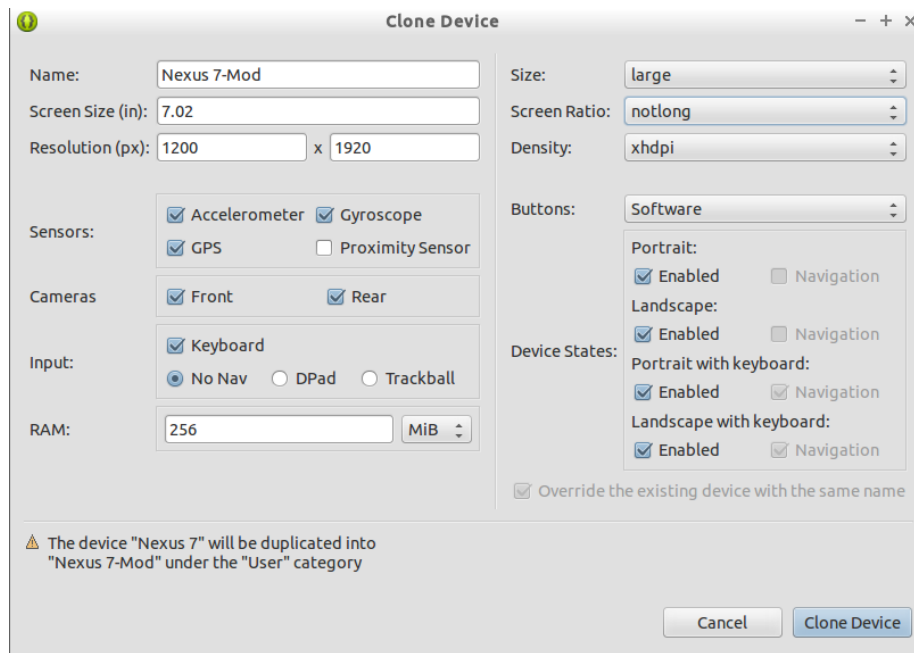


Figure 5: Nexus 7-Mod

- Assign New Device to SimpleLockerAVD

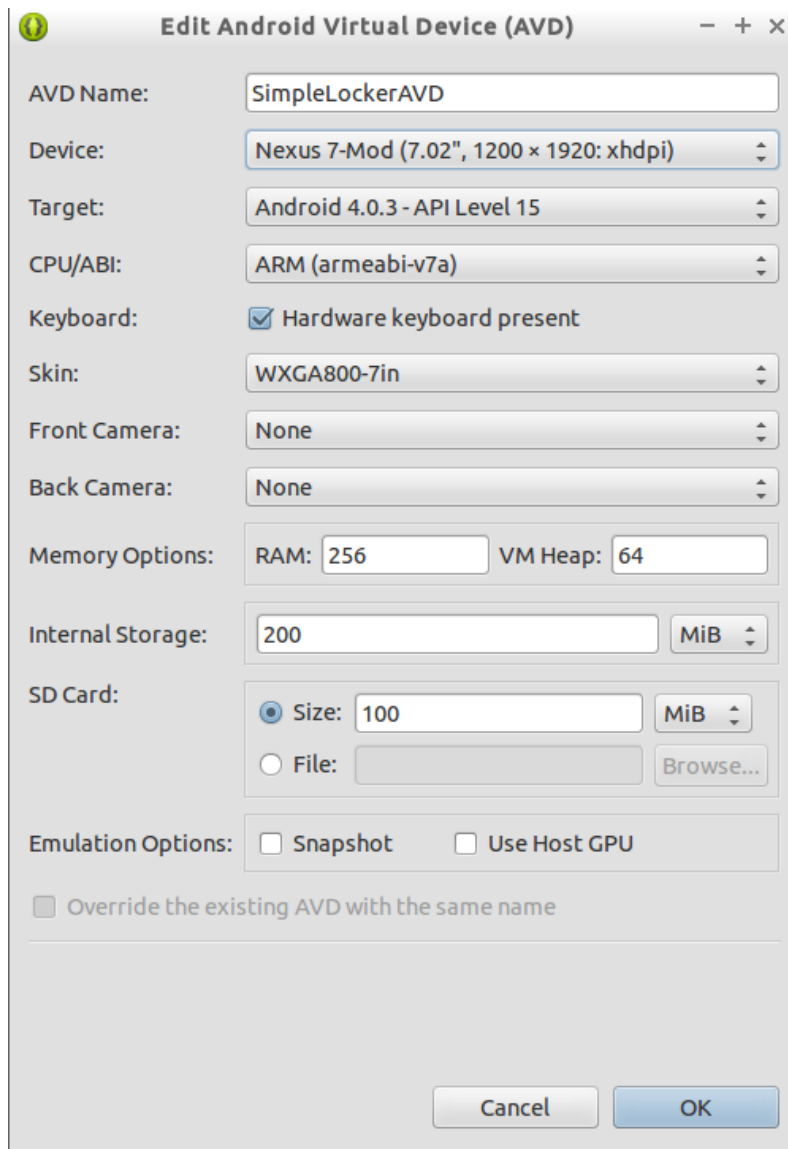


Figure 6: Modified Device

Note: Open AVDs (use separate terminals)

- `cd ./android/adt-bundle-linux-20140702/sdk/tools/`
- `./emulator -avd Enisa -tcpdump dump.pcap`
- `cd ./android/adt-bundle-linux-20140702/sdk/tools/`
- `./emulator -avd SimpleLockerAVD`

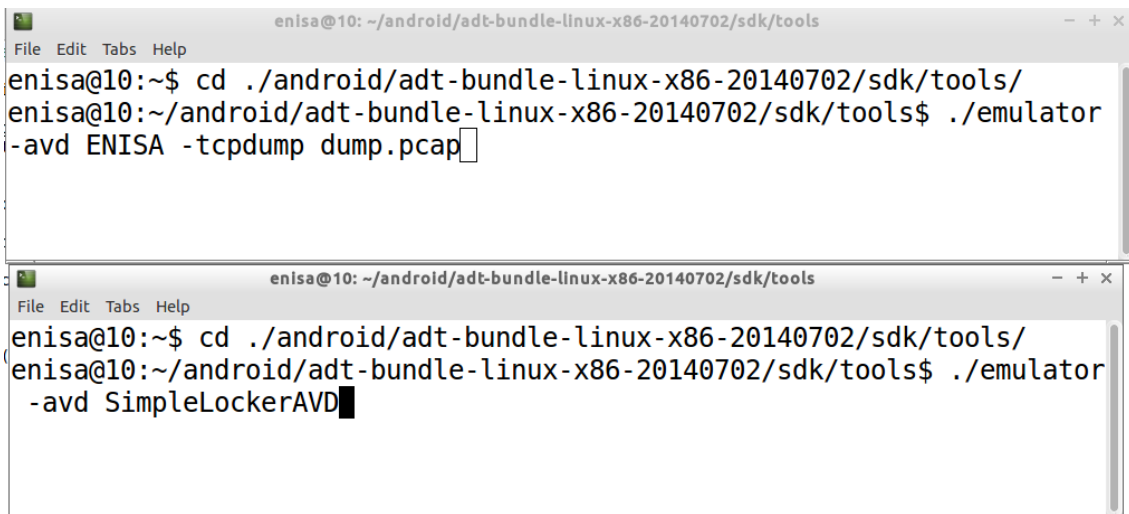
The image shows two overlapping terminal windows. The top window has a title bar that reads "enisa@10: ~/android/adt-bundle-linux-x86-20140702/sdk/tools". The terminal content shows the user navigating to the SDK tools directory and running the command: `./emulator -avd ENISA -tcpdump dump.pcap`. The bottom window has a similar title bar and shows the user running the command: `./emulator -avd SimpleLockerAVD`. Both windows have a menu bar with "File", "Edit", "Tabs", and "Help".

Figure 7: Launch AVDs in different terminals

Note: emulators must be opened on different terminals. These Terminals will be dedicated to the AVD, do not close or write commands in them as this will close the emulator.

2.1.7 Sending commands through telnet:

Connect:

- **telnet localhost 5554**

Commands:

- **power status full**
- **power status charging**
- **gsm call 0123456789**
- **sms send 12345 hello test sms**
- **geo fix 48 51**

2.1.8 Try shortcut keys

| Emulated Device Key | Keyboard Key |
|--|---|
| Home | HOME |
| Menu (left softkey) | F2 or Page-up button |
| Star (right softkey) | Shift-F2 or Page Down |
| Back | ESC |
| Call/dial button | F3 |
| Hangup/end call button | F4 |
| Search | F5 |
| Power button | F7 |
| Audio volume up button | KEYPAD_PLUS, Ctrl-F5 |
| Audio volume down button | KEYPAD_MINUS, Ctrl-F6 |
| Camera button | Ctrl-KEYPAD_5, Ctrl-F3 |
| Switch to previous layout orientation (for example, portrait, landscape) | KEYPAD_7, Ctrl-F11 |
| Switch to next layout orientation (for example, portrait, landscape) | KEYPAD_9, Ctrl-F12 |
| Toggle cell networking on/off | F8 |
| Toggle code profiling | F9 (only with <code>-trace</code> startup option) |
| Toggle fullscreen mode | Alt-Enter |
| Toggle trackball mode | F6 |
| Enter trackball mode temporarily (while key is pressed) | Delete |
| DPad left/up/right/down | KEYPAD_4/8/6/2 |
| DPad center click | KEYPAD_5 |
| Onion alpha increase/decrease | KEYPAD_MULTIPLY(*) / KEYPAD_DIVIDE(/) |

Figure 8: Basic Emulator Commands¹

2.1.9 Enable USB debugging on emulator

- Developer Options>USB debugging

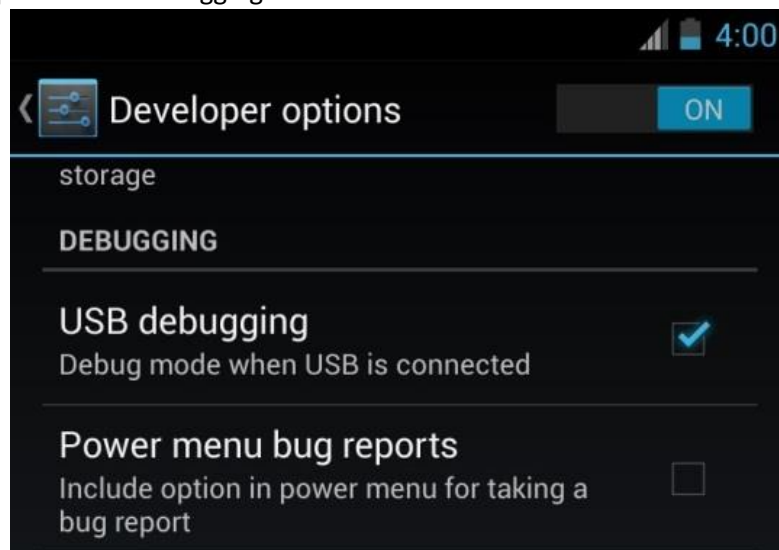


Figure 9: Developer Options

¹ <http://developer.android.com/tools/help/emulator.html>

2.1.10 List Devices

- `cd /android/adt-bundle-linux-x86-20140702/sdk/tools`
- `adb devices`

2.1.11 Run command on specific device

- `adb -s emulator-5554 shell`

2.1.12 Browse device folders using shell

- `adb shell`
- `cd /data/data`
- `ls`
- `cd /data/app`
- `ls`
- `exit`

2.1.13 Push and pull file to and from Emulator

- Create file on desktop (test.txt)
- `adb push /home/Enisa/Desktop/test.txt /sdcard/testpush.txt`
- `adb pull /home/Enisa/Desktop/testpull.txt /sdcard/testpush.txt`

2.2 Task 2: Cloning an Application

2.2.1 Run the Android emulator

In terminal run commands:

- `cd /android/adt-bundle-linux-20140702/sdk/tools/`
- `./emulator -avd Enisa -tcpdump dump.pcap`

2.2.2 Download and Install MySuperAV

In Android Emulator:

- Go to url: `10.0.2.2/Good.html` on Browser
- Click on the download link

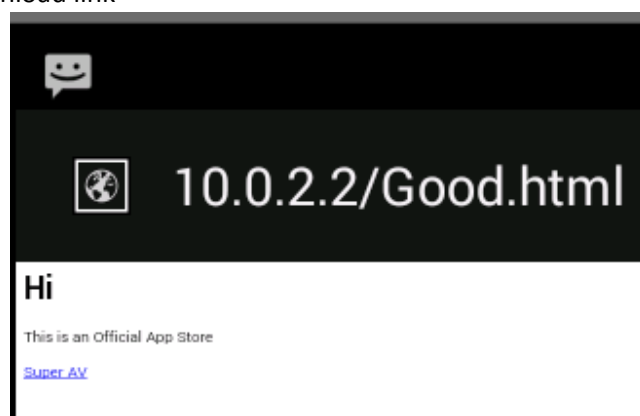


Figure 10: Good Store

2.2.3 Install the app

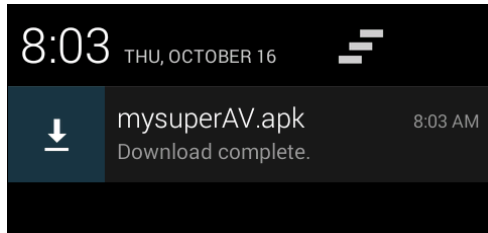


Figure 11: Download

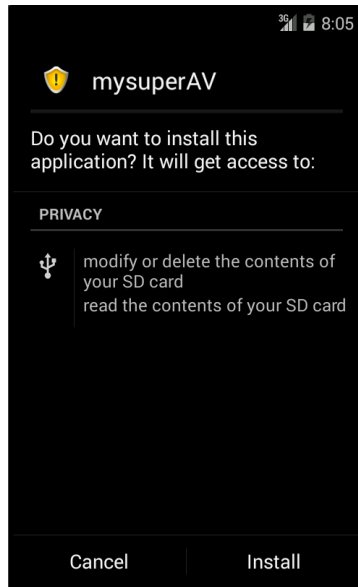


Figure 12: Install

2.2.4 Run the app

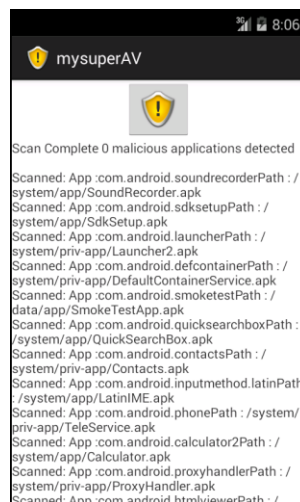


Figure 13: App Screenshot

2.2.5 Task: use adb pull to get apk file

2.2.6 Uninstall the app

2.2.7 Decode the app

- `apktool d mysuperAV.apk`

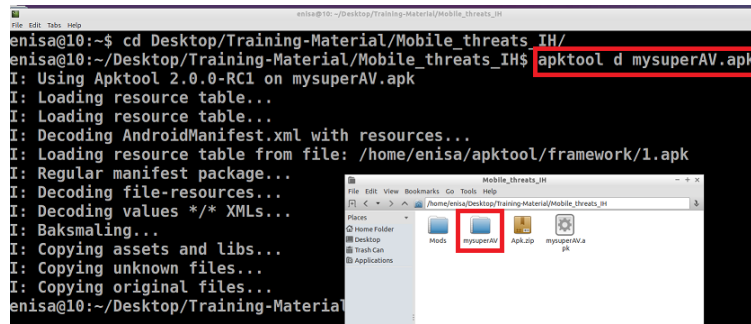


Figure 14: ApkTool Decode

2.2.8 Inject Code - Classes

- Copy the custom smali folder ('Mods/smali/code') into the decoded app's ('smali/com') folder

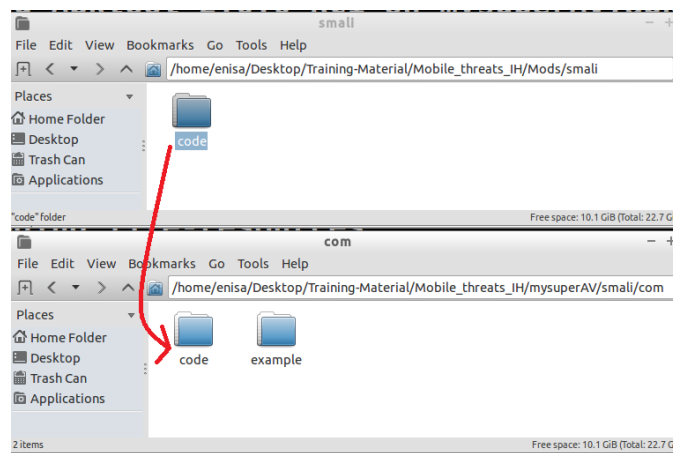


Figure 15: Inserting Code

2.2.9 Inject Code - Manifest

- Copy the **contents** of **Mods/manifest/Mods_AndroidManifest.xml** into the decoded app's Manifest (**mysuperAV/AndroidManifest.xml**) (use leafpad and read instructions on **Mods_AndroidManifest.xml**)

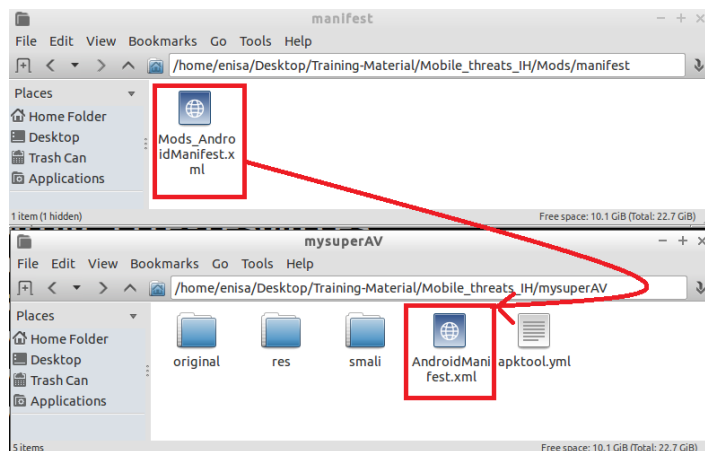


Figure 16: Manifest Files

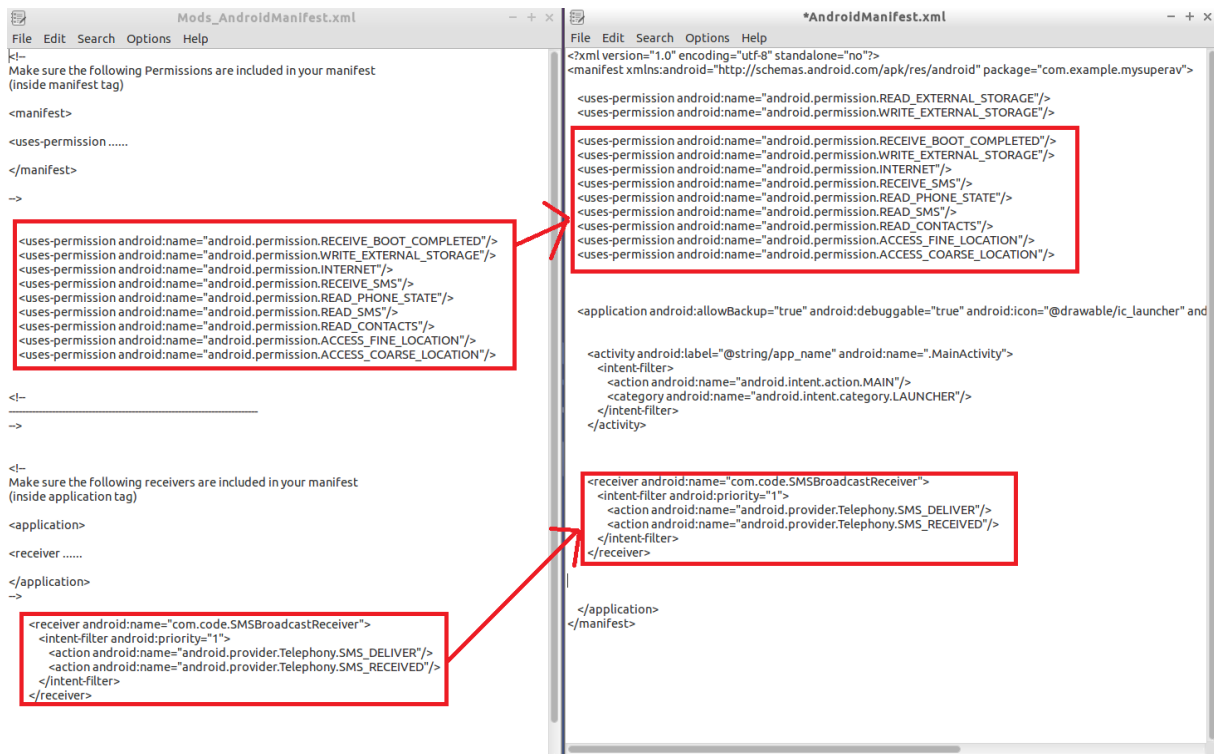


Figure 17: manifest modification

1. Build app

- **apktool b mysuperAV**

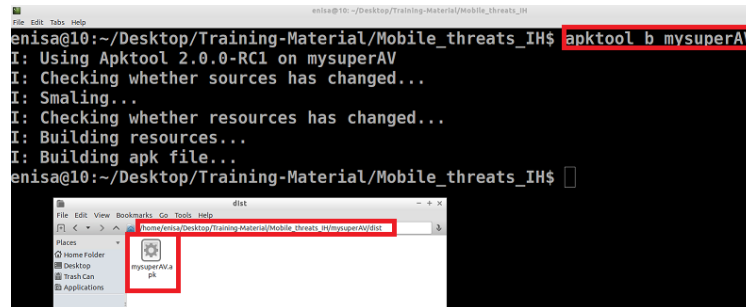


Figure 18: Building App

2. Move apk to signAPK folder

- Copy apk (**mysuperAV/dist/mysuperAV.apk**) and places it in the **Tools/SignAPK** folder

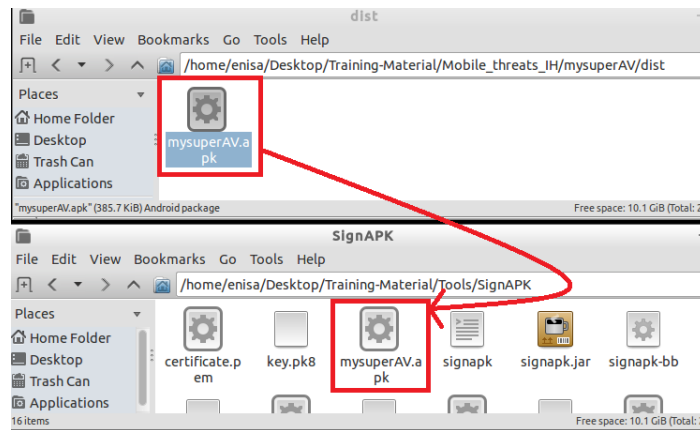


Figure 19: Placing apk into SingAPK

2.2.10 Sign the apk

- `java -jar signapk.jar certificate.pem key.pk8 mysuperAV.apk mysuperAV_signed.apk`

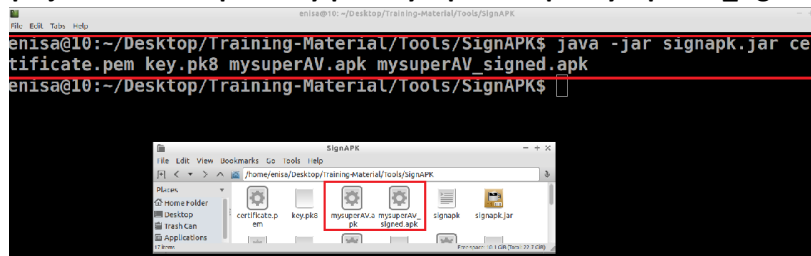


Figure 20: Sign apk

2.2.11 Install the app

In Android Emulator:

- Make sure mysuperAV is not installed
- Go to url: 10.0.2.2/Bad.html on Browser
- Click on the Download link
- Install the app

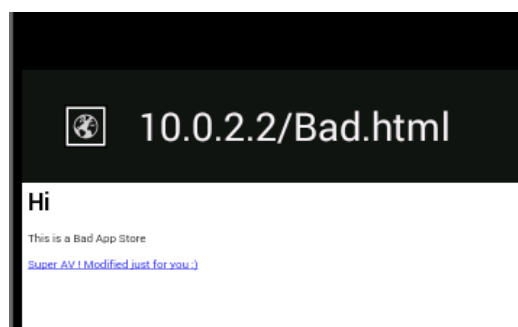


Figure 21: Bad Store

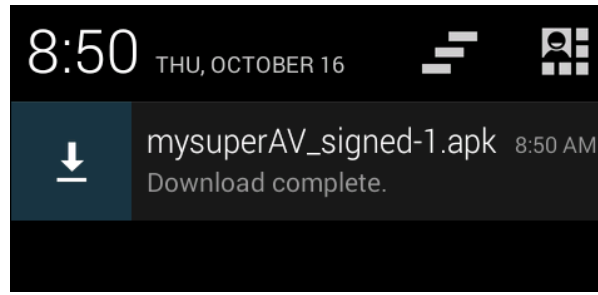


Figure 22: Download apk

2.2.12 Run the app

- Open and run the app. See that all works fine.
- Close the app

2.2.13 Add Contact

- Go to 'People' app
- Click 'Create New Contact'
- Select 'Keep Local'
- Name: Bank
- Phone: +123456789
- Click 'Done'

2.2.14 Send SMS to emulator

- `echo sms send +123456789 'This is Confidential' | nc localhost 5554`
- `echo sms send + 123456789 'your bank pin code is 1234' | nc localhost 5554`

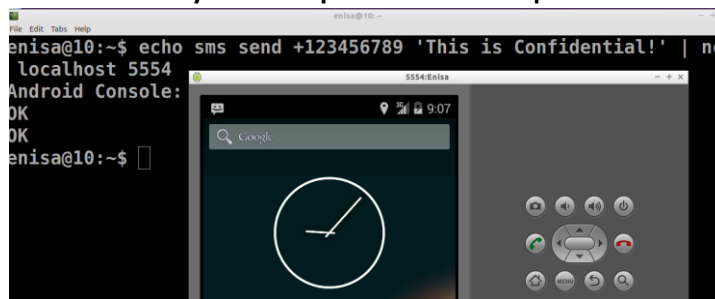


Figure 23: Send SMS

2.2.15 View Stolen Data

- Open text file `"/var/www/html/myfile.txt"` found on the Bad Server! This text file contains the stolen data that was taken from the victims device.
- `sudo leafpad /var/www/html/myfile.txt`



Figure 24: Stolen Data

2.3 Task 3: Analysing Cloned Application

2.3.1 Task: use adb pull to get apk file

2.3.2 Analyse network traffic

- `cd /android/adt-bundle-linux-20140702/sdk/tools/`
- `wireshark dump.pcap`
- find http post to bad server (/bad.php)

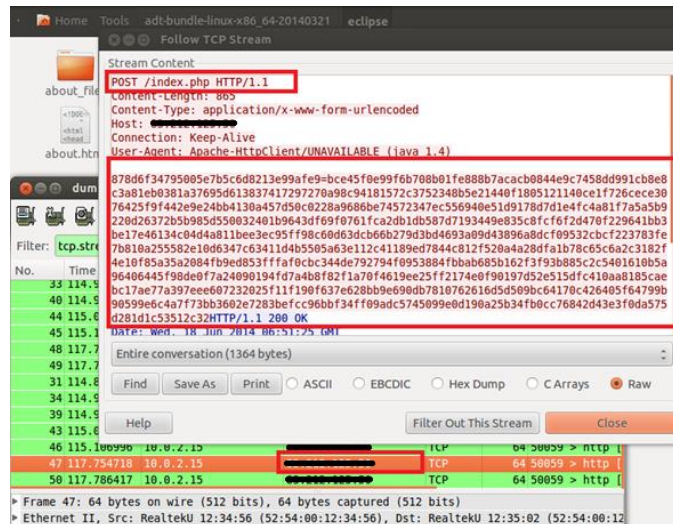


Figure 25: Network Traffic

2.3.3 Create new Investigation Folder

- Place original (**Mobile_threats_IH/mysuperAV.apk**) and cloned/signed (**Tools/SignAPK/mysuperAV_signed.apk**) apks into a new folder “**Mobile_threats_IH/Investigation**”

Note: Alternatively you can also find both apks in path (/var/www/html/). Copy and paste them into the investigations folder.

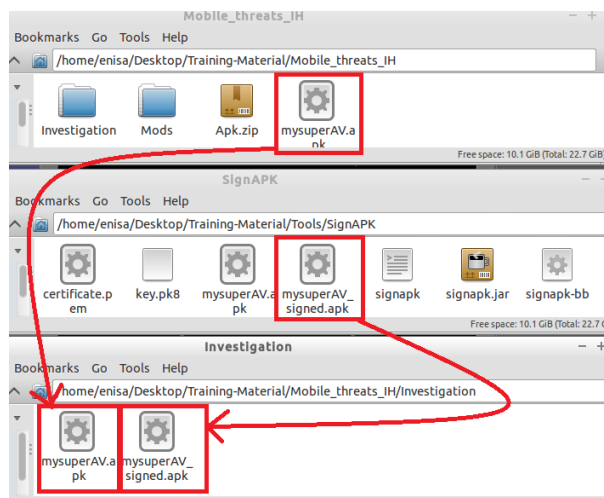
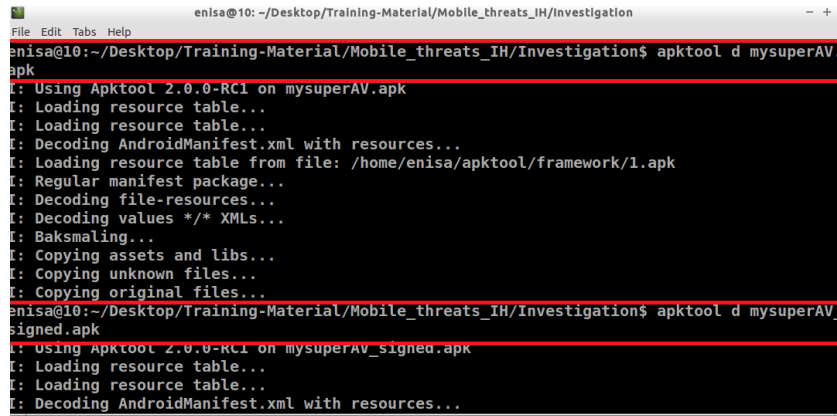


Figure 26: Investigation Folder

2.3.4 Decode both apk files

In terminal run command:

- `cd Desktop/Training-Material/Mobile_threats_IH/Investigation`
- `apktool d mysuperAV.apk`
- `apktool d mysuperAV_signed.apk`



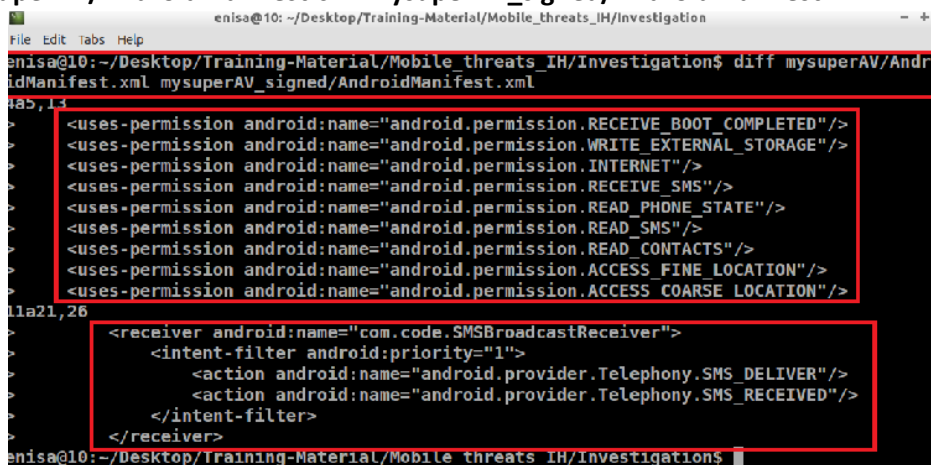
```
enisa@10: ~/Desktop/Training-Material/Mobile_threats_IH/Investigation
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH/Investigation$ apktool d mysuperAV.apk
I: Using Apktool 2.0.0-RC1 on mysuperAV.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/enisa/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH/Investigation$ apktool d mysuperAV_signed.apk
I: Using Apktool 2.0.0-RC1 on mysuperAV_signed.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
```

Figure 27: Decode apk files

2.3.5 Compare the content of both Manifests

In terminal run command:

- `cd Desktop/Training-Material/Mobile_threats_IH/Investigation`
- `diff mysuperAV/AndroidManifest.cml mysuperAV_signed/AndroidManifest.xml`



```
enisa@10: ~/Desktop/Training-Material/Mobile_threats_IH/Investigation
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH/Investigation$ diff mysuperAV/AndroidManifest.cml mysuperAV_signed/AndroidManifest.xml
145,147c145,147
< <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
> <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
> <uses-permission android:name="android.permission.INTERNET"/>
> <uses-permission android:name="android.permission.RECEIVE_SMS"/>
> <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
> <uses-permission android:name="android.permission.READ_SMS"/>
> <uses-permission android:name="android.permission.READ_CONTACTS"/>
> <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
> <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
11a21,26
> <receiver android:name="com.code.SMSBroadcastReceiver">
> <intent-filter android:priority="1">
> <action android:name="android.provider.Telephony.SMS_DELIVER"/>
> <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
> </intent-filter>
> </receiver>
```

Figure 28: Modified Manifest

2.3.6 Convert apks into jar files

- `cd Desktop/Training-Material/Tools/dex2jar/`
- `./dex2jar.sh ../../Mobile_threats_IH/mysuperAV.apk`
- `./dex2jar.sh ../../Mobile_threats_IH/mysuperAV_signed.apk`

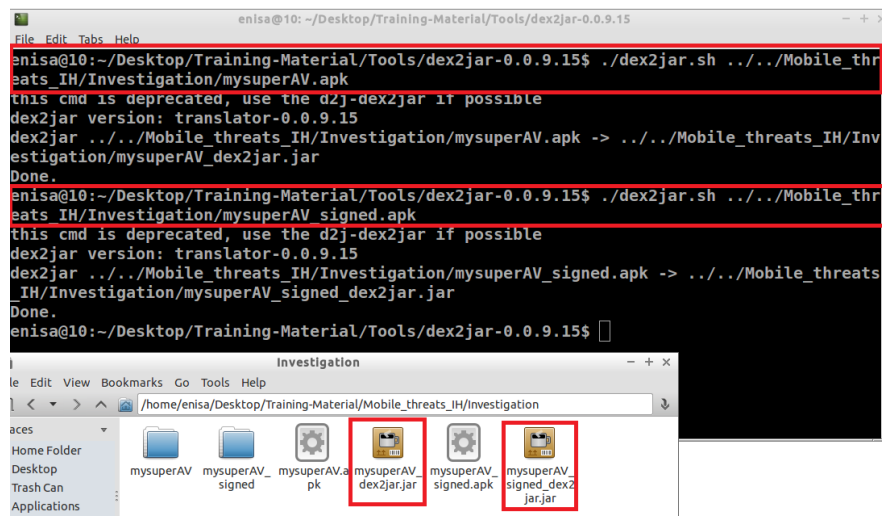


Figure 29: Dex2jar

2.3.7 Analyse classes

- Open JD-GUI (Programming > JD-GUI)
- Open both apks in JD-GUI

2.4 Task 4: Analysing Simplelocker

2.4.1 Task: Install Simplelocker

- **Disable Network connection on VM!!!**
- **cd Desktop/Training-Material/Mobile_threats_IH/**
- **7z e Simplelocker.apk.zip**
- **<ask for password in class>**
- Install using adb install or adb push

2.4.2 Task: Analyse Simplelocker

- Use apktool, analyse manifest
- Use dex2jar to create jar file
- Open jar file with jd-gui
 - **Challenge 1: What encryption algorithm is being used?**
 - **Challenge 2: What is the encryption key**
 - **Challenge 3: What files types are being encrypted**

2.4.3 Task: Remove Simplelocker

- Uninstall simplelocker using adb uninstall

2.5 Task 5 (Optional): Analysing Other Artifacts

Under /home/enisa/Mobile_threats_IH you will find samples of apks (Apk.zip), some of them are malicious while others are not. Some samples are still dangerous; Command and Control servers still active. You may wish to hand them to your students for additional practice.

Since some of the artefacts are still active, in order to analyse the network connections you may wish to use sandbox tools such as Anubis.

If you wish to run the applications locally, you should:

- use a segregated unprotected network, and clean/dummy devices without personal data (accounts, credentials, etc); or
 - use a dns and http server on the host, simulating the C&C servers (a possible approach could be to run the tool in a sandboxed malware analysis tool, check for network connections, and configure dns and http servers accordingly); or
- disable network connection (note that this may not always produce results, as the code may check for network connection prior to running its code)

2.5.1 Task: Flappy Bird

Analyse the code of both Flappy bird APKs. One of them is malware. Which one?

- Go to Mobile Folder:
 - **cd /home/enisa/Desktop/Training-Material/Mobile_threats_IH/**
- Unzip Simple locker file:
 - **unzip Apk.zip**
 - **<ask for password in class>**

2.5.2 Upload apk files to automated tools

- Anubis;
- Mobile Sandbox;
- Joe sandbox mobile;
- Others.
-

2.6 Questions

| No. | Question | Answer |
|-----|--|--------|
| 1 | List any suspicious permissions that are being acquired | |
| 2 | Can you identify any background app components (Services/Broadcast Receivers)? | |
| 3 | Can you identify the initial/starting activity? | |

| | | |
|---|---|--|
| 4 | Do you notice any suspicious network connections initiated by the application? | |
| 5 | From the code, can you see any suspicious Functions? | |
| 6 | If so, can you link these functions to the permissions and/or suspicious network connections? | |
| 7 | List any other useful information (eg: encryption keys, data being saved to database, etc.) | |
| 8 | What is the main purpose of this artefact? | |

3 Conclusion

Depending on the student's previous experience with Mobile Technologies, he/she should be able to answer most of the questions listed in section 2.1.5.1.

**ENISA**

European Union Agency for Network and Information Security
Science and Technology Park of Crete (ITE)
Vassilika Vouton, 700 13, Heraklion, Greece

Athens Office

1 Vass. Sofias & Meg. Alexandrou
Marousi 151 24, Athens, Greece



PO Box 1309, 710 01 Heraklion, Greece
Tel: +30 28 14 40 9710
info@enisa.europa.eu
www.enisa.europa.eu