General Notes

https://www.youtube.com/playlist?list=PL6gx4Cwl9DGBsvRxJJOzG4r4k_zLKrnxl

android is built on top of linux.

Android Architecture

Consists of 4 Layers

1 - Application : Contains all application UI (what we see and interact with).
2 - Application Framework
3- Libraries && android Runtime
Libraries contain additional functions that can be used like SQLlite and Webkit libraries.
Android Runtime is the environment in which we have the core libraries and the application runs (dalvik executable file that runs on dalvik vm).
4- Linux Kernel
OS files , drivers and such.

Android Security

Models:

1- Linux based model :
- Privilege control --> each application has a separate process id.
- Every application is given a separate user id and process id
- The user of the application is the one that owns the process and the directory of the application.

2- Android based model :
Permission control --> the user is asked if he allows specific permissions to the application when it is installed or requires it.

Android Application Components

Activity:
Application consists of different screens , similar to html pages in a website.

Intent:
To combine two or more android application components.
Usually used when switching between activities and u want to pass some parameters to the new one.

Content Providers:
Used to store and retrieve data from the database, internet, filesystem .(middle man)

Broadcast receivers:
Listen to and send messages from other applications.

AndroidManifis.xml :
Contains all the permissions declared by the developer.
Has all the properties of the application including required libraries to be loaded.
Can't be modified later on after compilation.

Compilation life Cycle :

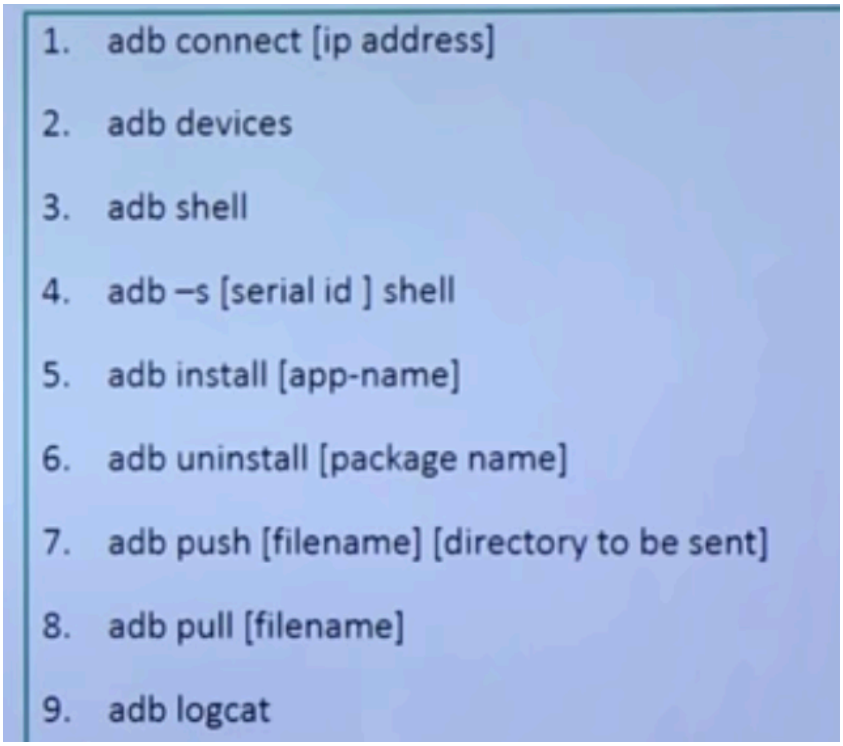
1- Java Source code --> compile with a java compiler
2- Byte code --> compile with a dex compiler
3- Dalvik Byte code --> has extenstion .dex (dalvic executable) , now zipped
4- Apk package

Lab Setup

1- Santoku OS: https://santoku-linux.com/
I think kali linux is sufficient for now.

2- Genymotion : Very good android emulator

ADB : Android Debug Bridge



adb logcat = access the central repository that the phone stores the logs in

- To get the ip address of phone dont use the one on genymotion title bar , go to the wireless settings instead
- external memory cars is in /mnt/sdcard

Startup process in android

1. Bootloader (boots kernel)
2. Init process starts running. The configuration is got from the /init.rc
3. Zygote process is run under init.
4. Dalvic Virtual machine is run under zygote process.
5. Boot completed Broad cast, lets all applications know that the phone is booted.

Reversing an android application

.smali --> bytecode (intermediate between java and assembly)
.java --> java code (men 2smaha)

apktool : for compiling and decompiling apk's
apktool d <apkfile> --> gets smali output
apktool b <directory> -o <apkname> --> this rebuilds the apk from the directory

jadx : download from github
./jadx -d <apkfile> --> gets java

Application Signing

You generate public and private keys and encrypt your application with the private key

Certificate authority: you buy it
Self signing: create your own custom certificate

Dex file

Dalvik executable file
classes.dex --> contains a lot of info about the application (this runs inside the dalvik vm)
info like

Tools to decompile the dex file:

1. Dex dump
2. 010 Editor
3. d2j-dex2jar
4. JD-GUI

Commands: (go inside the directory where the classes.dex resides)

dexdum -l plain classes.dex
dexdum -l xml classes.dex

dex2jar classes.dex
jd-gui classes_dex2jar.jar

Mobile Traffic Analysis

Passive (Secret) analysis :
Tcp dump

Active analysis:
Burp

To put Burp as a proxy:
go to genymotion --> wireless settings , modify , add proxy
go to burp --> proxy --> option --> edit --> set all interfaces
go to browser --> http://burp --> Download CA certificate (cacert.crt)
find certificate inside : /mnt/sdcard/Download --> rename it like this :
root@android:/sdcard/Download # mv cacert.der cacert.crt
go to settings --> security --> install certificates from sdcard -->

API hooking

Analyzing the methods and arguments that are passed by the application when it does an api call, we can then do api fuzzing

AndBug

git clone https://github.com/swdunlop/AndBug
python setup.py install

root@kali:~/mobile/AndBug# adb shell
root@android:/ # ps | grep insecure
u0_a50 **1171** 182 510832 35620 ffffffff b753e507 S com.android.insecurebankv2

1171 --> process id of insecure bank

now do andbug shell -p 1171

>> classes --> shows all classes of all applications.
>> classes com.android.insecurebankv2 --> loads all the classes of this application. why though ?
>> methods com.android.insecurebankv2.LoginActivity --> shows all methods inside this class
>> method-trace com.android.insecurebankv2.LoginActivity.performLogin()V:0 --> sets hooks on this motion

go to genymotion and login, now u can see what is happening at the backend

Java Debugger

can be used if the application is debuggable (if it uses JDWP, java debug wire protocol)

adb jdwp --> lists all process ids of java apps that used JDWP
Port forwarding
adb forward tcp:<port> jdwp:<processid> --> attaches the port of the process to our jdwp for debugging
jdb -attach localhost:<port>

> methods <classname>
> stop in <method> --> sets a breakpoint
now invoke the method from application (Genymotion)
<1> main[1] locals --> shows variables and arugments
<1> main[1] resume

Other tools :

Introsopy
Cydia Substrate

Drozer

https://labs.f-secure.com/tools/drozer/

framwork that provides comprehensive security audit and attacks on mobile

Has a client server arch. where we install client on kali and server on genymotion (agent).
so any command we execute gets send to the phone.

pip3 install drozer
adb install drozer-agent-2.3.4.apk

adb forward tcp:31415 tcp:31415
set the off to on on the mobile !!!!! --> important we 23adt feha keteer ?2!@#!@
drozer console connect

> ls --> shows available modules
> run app.package.list -->
> run app.package.debuggable
> run app.package.attacksurface com.android.insecurebankv2