Android Mobile Application Cheatsheet:

# Notes

1. Intent
   * universal interprocess communication for other OS
   * everything we do using intent:
     + #start activity, service, deliver broadcast message(eg turning wifi on, ask to turn signal to on wifi)
   * adb logcat --> show system logs for debug
2. Manifest
   * declare activities, broadcast receivers and services
   * declare permissions, hardware/software needed
3. Drozer
   * framework security assessment, install agent on android
   * communicate through tcp port 31415
4. jadx-gui
   * decompile dex files
   * can decompile to smali (assembler code for android)
   * search through things like:
     + intent communication method calls
     + dangerous network class use 🡪 open backdoor to network socket
     + sms manager use 🡪 utk elak malware from sending info or pull entire addr book to send outside
     + contact/addr book access
5. Method
   * recon – static
   * file system usage - sensitive data storage
   * network communication capture and analysis
     + plaintext vs encrypted traffic
     + api authentication usage
   * application manipulation – drozer
6. Apk Download
   * <https://apps.evozi.com/apk-downloader/>
   * <https://apkpure.com/>
7. Bypass root detection
   * Boleh guna frida or patch manually.signing apk manual patching
     + <https://stackoverflow.com/questions/10930331/how-to-sign-an-already-compiled-apk>
8. Testing guide
   * <https://github.com/OWASP/owasp-mstg>
9. Tools
   * <https://github.com/tanprathan/MobileApp-Pentest-Cheatsheet>
10. Adb Command
    * adb connect [ip-addr]
    * adb devices
    * adb shell
    * adb -s [serial-id] shell
    * adb install [app-name]
    * adb uninstall [app-name]
    * adb push [filename] [directory-to-be-sent]
    * adb pull [filename]
    * adb logcat

# Notes

1. Rooting the phone
   1. <https://magiskmanager.com/>
2. Installing

# Reverse Engineering and Static Analysis:

1. MobSF
2. APKTool - A tool for reverse engineering 3rd party, closed, binary Android apps. It can decode resources to nearly original form and rebuild them after making some modifications.
   * Disassembling Android apk file
     1. apktool d <apk file>
   * Rebuilding decoded resources back to binary APK/JAR with certificate signing
     1. apktool b <modified folder>
     2. keytool -genkey -v -keystore keys/test.keystore -alias Test -keyalg RSA -keysize 1024 -sigalg SHA1withRSA -validity 10000
     3. jarsigner -keystore keys/test.keystore dist/test.apk -sigalg SHA1withRSA -digestalg SHA1 Test
3. Unzip -d <folder-name> <apk file>
   * To use the classes.dex file, dex file nie contains all java code, and also for dexdump
4. Jadx -d <folder-name> <apk file> || jadx-gui -d <folder-name> <apk file>
   * decompile all code in java
5. dexdump classes.dex >> dexvalue.txt
6. d2j-dex2jar.sh classes.dex
   * this will produce jar file
7. jd-gui classes-dex2jar.jar
   * see all the code of java using jd-gui

# Dynamic Analysis:

1. Adb
   1. adb shell 'pm list packages -f'
2. Drozer
   1. run app.package.list -f <appname>
   2. run app.package.info -a <package name>

# Reminder after finished testing

Things that constantly to check to do the mobile pentest:-

* shared preferences
* sqlite db
* internal storage
* external storage
* using network connection

# References

1. <https://www.youtube.com/watch?v=2uwhrfXCl4I>