# Android Mobile Pentest 101

# Lecture 8 – Tools time

Goal: Speed up our pentest process

# Why?

- Assume that we are so noob, the app is just too hard for us, cannot reverse, cannot patch, cannot hook, etc... So, give up?
- Don’t worry, maybe tool will help you. I always use tool first, if fail, the manual phase come next ☺
- I will introduce tool aim for bypass root detection, emulator detection and ssl pinning

# Root Detection Bypass

- To bypass root detection, we are going to use RootCloak
- It is a module for Xposed Framework
- It uses a variety of methods, completely hide root from the app
- This includes hiding the su binary, superuser/supersu apks, processes run by root, adb, and more.

# Root Detection Bypass -> Install

- Since it is a module for Xposed Framework, we have to install Xposed first
- Install via MobSF script:

https://github.com/MobSF/Mobile-Security-Framework-MobSF/blob/master/scripts/mobsfy.py

- Type command:

python3 mobsfy.py -i 192.168.56.101:5555 -t 1

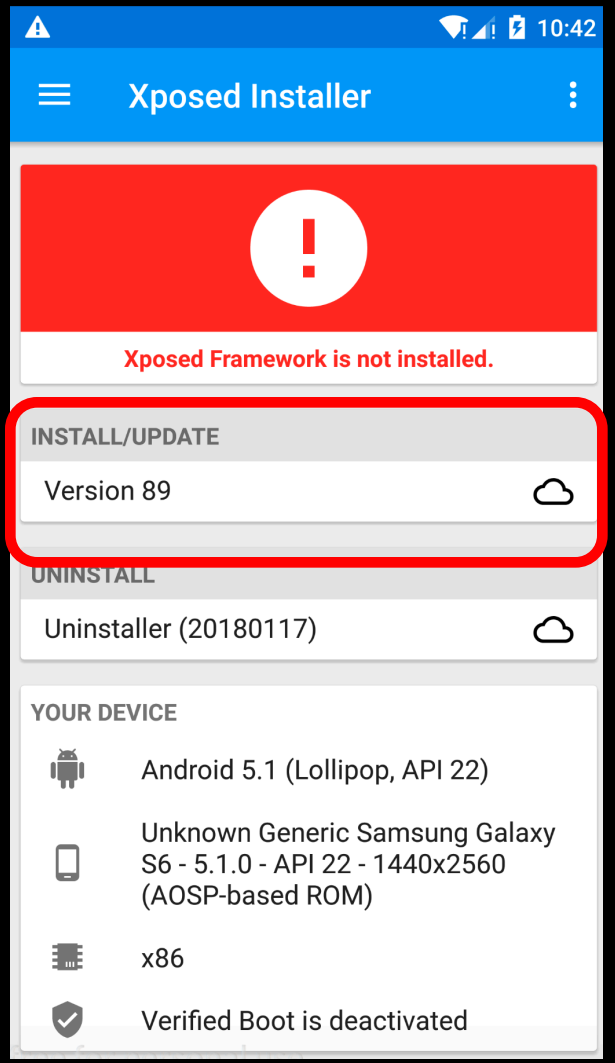
The ip is ip address of our virtual phone, the value of option –t equal to 1 to specify it's the virtual, 2 is device

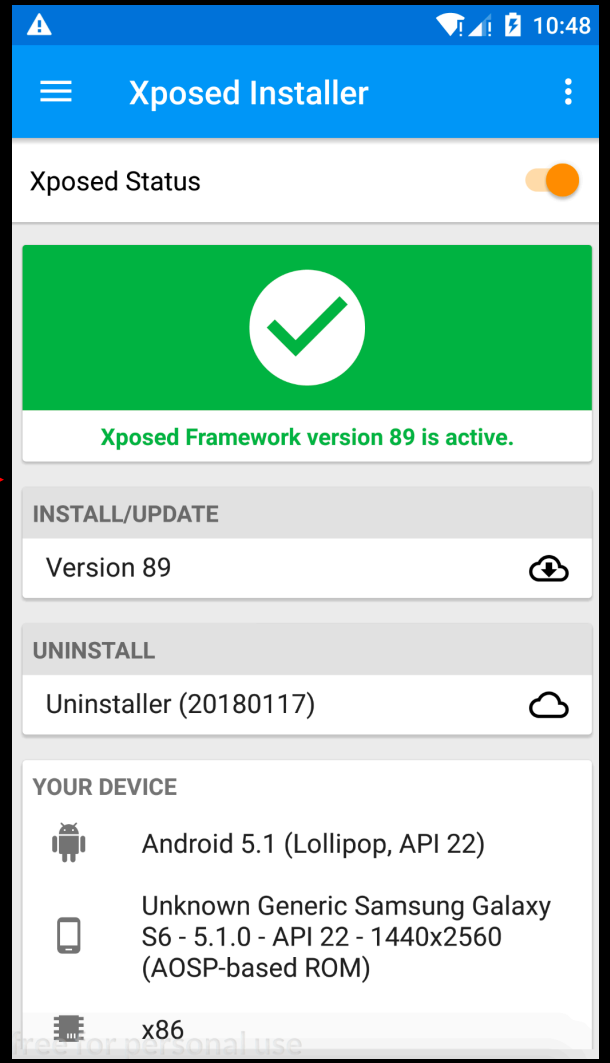
```
[INFO] Executing Command - /Users/tsug0d/Desktop/mobile/tools/Mobile-Security-Framework-MobSF/scripts/../DynamicAnalyzer/tools/adb/mac/adb connect 192.168.56
.101:5555
adb server version (40) doesn't match this client (39); killing...
adb E 09-25 09:35:02  4717 409269 usb_osx.cpp:152] Unable to create an interface plug-in (e00002be)
error: could not install *smartsocket* listener: Address already in use
ADB server didn't ACK
* failed to start daemon *
error: cannot connect to daemon
```

- Fail because we are dealing with genymotion, we have to use geny adb, replace this adb with genymotion adb and we are all done

# Root Detection Bypass -> Install

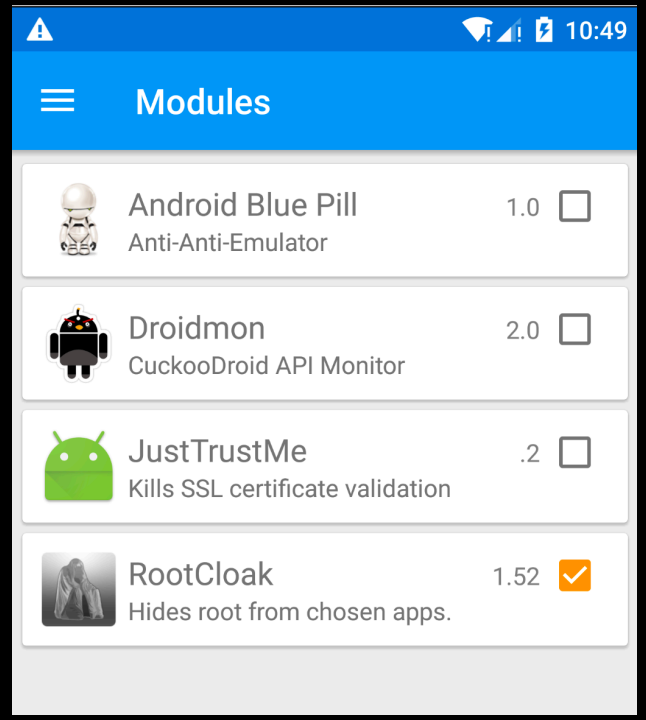
- Run it again, Xposed appear in our phone, but still not install, we have to click install option



# Root Detection Bypass -> Install

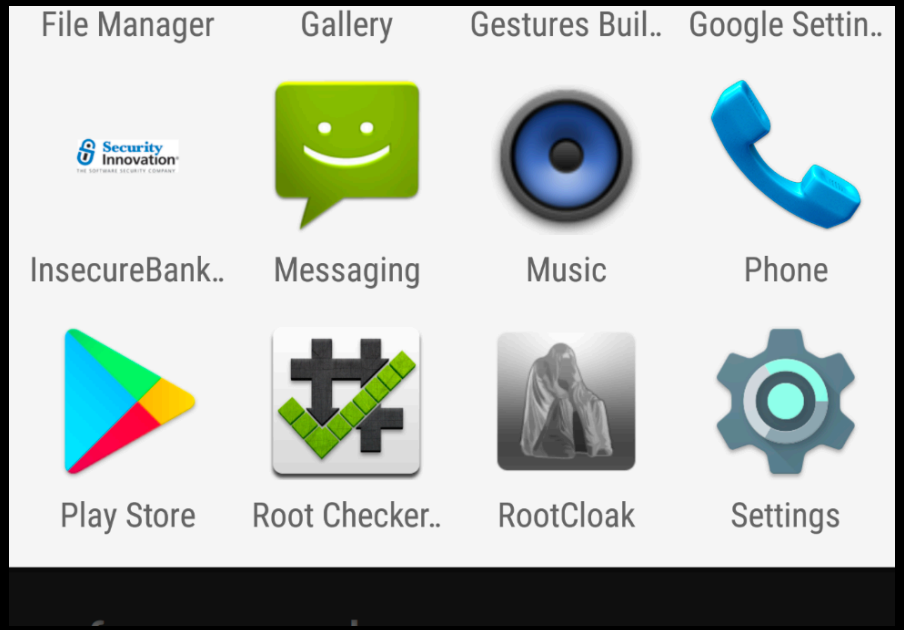
- Now go to Modules task and tick on RootCloak




- Then reboot the phone

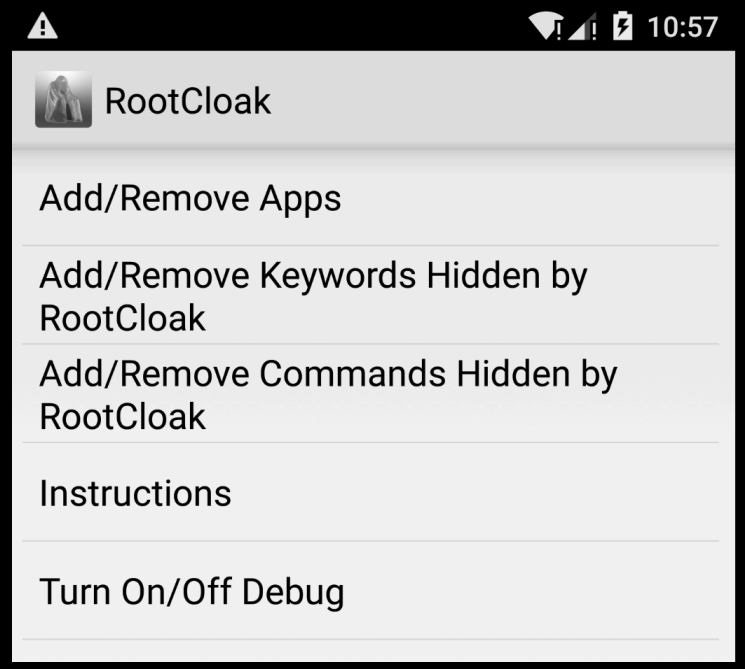
# Root Detection Bypass -> Install

- RootCloak appear ☺



# Root Detection Bypass -> Using

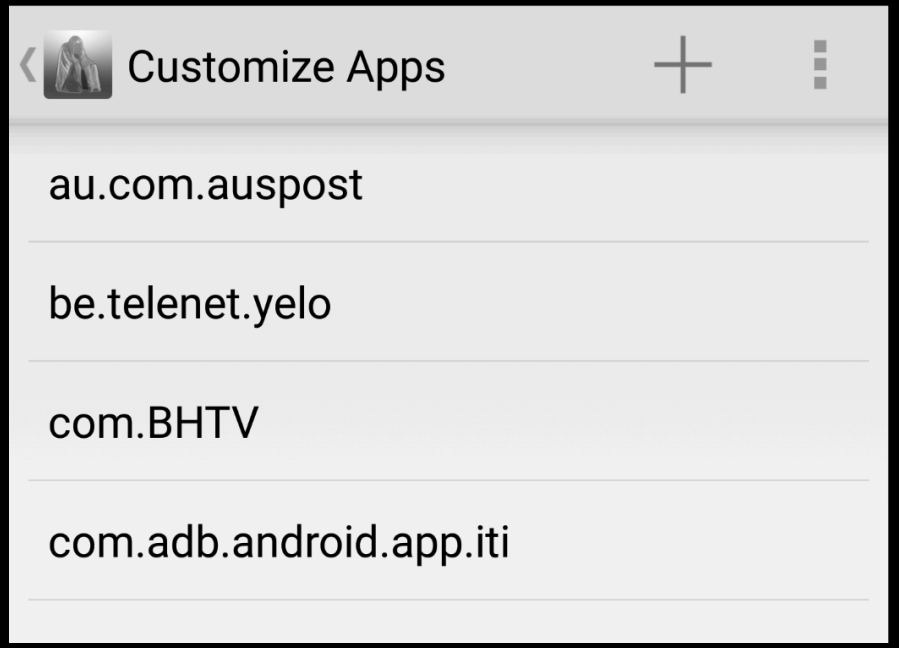
- To use it, click on the app icon, it look like this:




- Choose Add/Remove Apps

# Root Detection Bypass -> Using

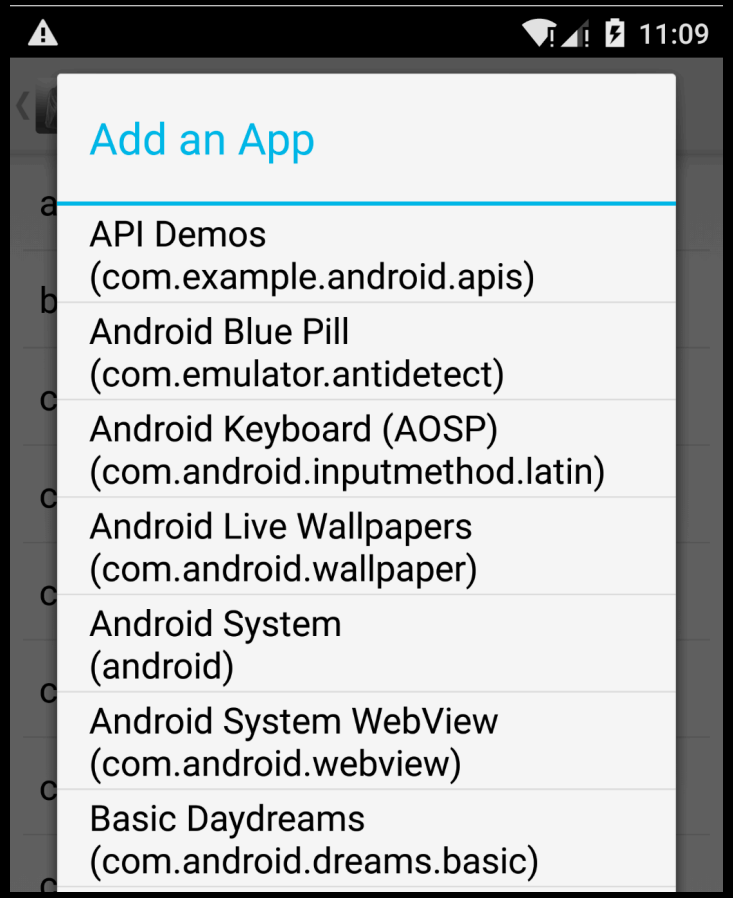
- To use it, click on the app icon, it look like this:




- Click on plus symbol

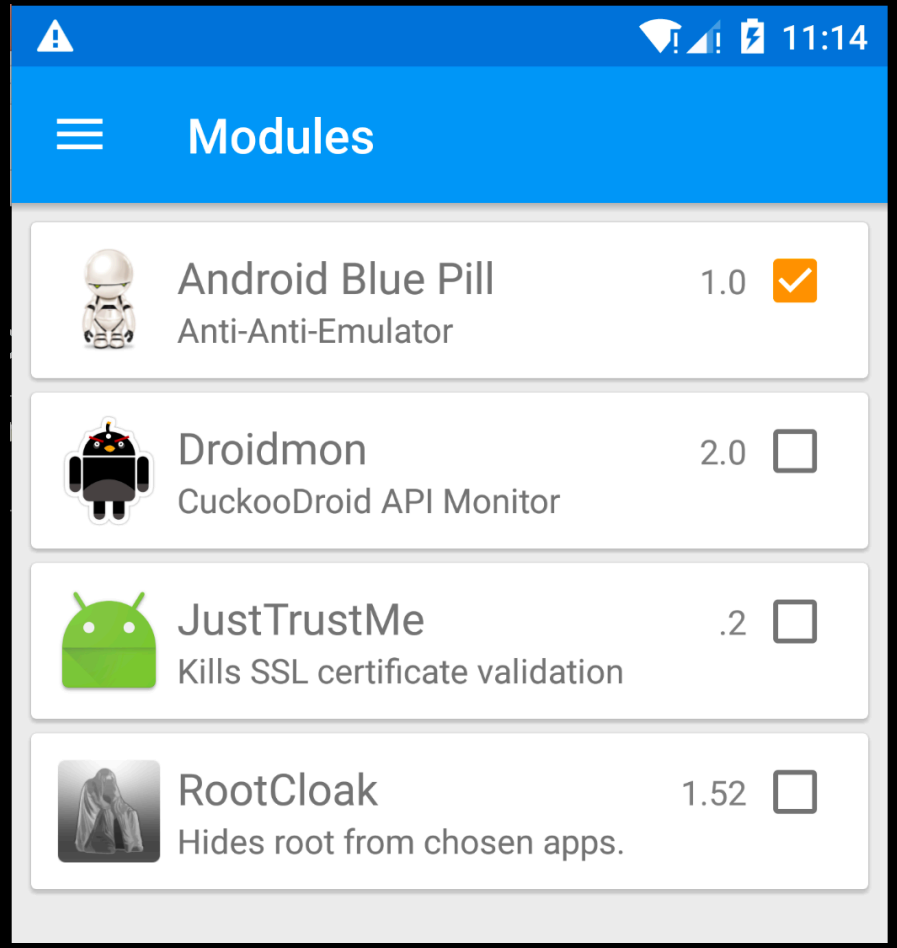
# Root Detection Bypass -> Using

- Add your app to hide root from it, you are all set



# Emulator Detection Bypass

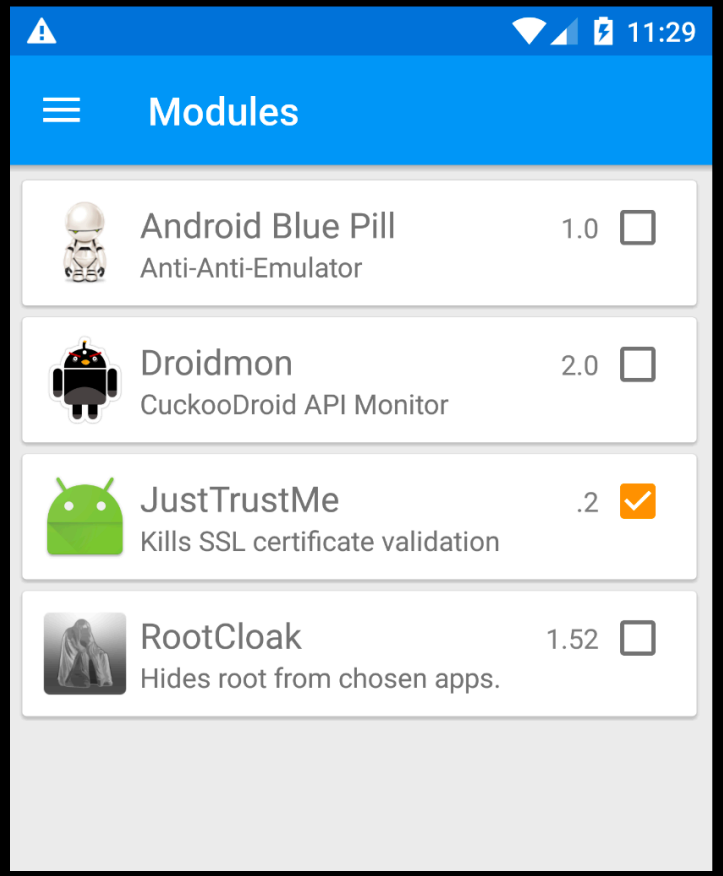
- Same as root bypass, we select Android Blue Pill in Xposed




- Reboot the phone, done ( Just introduce, I always fail when using it ☺ )

# SSL Pinning Bypass -> JustTrustMe

- In this section, i'll introduce 2 tools I always use, the first is JustTrustMe from Xposed Modules




- Tick on it, reboot, done
- Since it is too old, I prefer the second one!

# SSL Pinning Bypass -> Objection

- Remember last slide of lecture 6? We are going to use Objection - a runtime mobile exploration toolkit, powered by Frida, to bypass ssl pinning
- Objection was built with the aim of helping assess mobile applications and their security posture without the need for a jailbroken or rooted mobile device.

**Note:** This is not some form of jailbreak / root bypass. By using objection, you are still limited by all of the restrictions imposed by the applicable sandbox you are facing.

# SSL Pinning Bypass -> Objection

- To Install, simply type command:

pip3 install objection

- Test if it is installed:

```
 ~/Desktop/mobile/tools/objection/ objection
Usage: objection [OPTIONS] COMMAND [ARGS]...

      _     _         _   _
  ___| |_  |_|___ ___| |_|_|___ ___
 | . | . | | | -_|  _|  _| | . |   |
 |___|___|_| |___|___|_| |_|___|_|_|
         |___|(object)inject(ion)

      Runtime Mobile Exploration
         by: @leonjza from @sensepost

  By default, communications will happen over USB, unless the --network
  option is provided.
```

# SSL Pinning Bypass -> Objection

- To use Objection, you need two things:

1. Objection installed
2. A patched APK installed to your android device and with the device connected and authorized to your computer via USB

- We already got the first thing, so we will create a patched APK, type command:

objection patchapk --source InsecureBankv2.apk

(More about this: https://github.com/sensepost/objection/wiki/Patching-Android-Applications)

- Result:

```
 ~/Desktop/mobile/tools/objection/ objection patchapk --source InsecureBankv2.apk
No architecture specified. Determining it using `adb`...
Detected target device architecture as: x86
Using latest Github gadget version: 12.2.5
Patcher will be using Gadget version: 12.2.5
Unpacking InsecureBankv2.apk
App already has android.permission.INTERNET
Reading smali from: /var/folders/h1/rxkqmv9d69vg7j1cw8k1d13m0000gn/T/tmp8vw27ag0.apktemp/smali/com/android/insecurebankv2/LoginActivity.smali
Injecting loadLibrary call at line: 24
Writing patched smali back to: /var/folders/h1/rxkqmv9d69vg7j1cw8k1d13m0000gn/T/tmp8vw27ag0.apktemp/smali/com/android/insecurebankv2/LoginActivity.smali
Creating library path: /var/folders/h1/rxkqmv9d69vg7j1cw8k1d13m0000gn/T/tmp8vw27ag0.apktemp/lib/x86
Copying Frida gadget to libs path...
Rebuilding the APK with the frida-gadget loaded...
Built new APK with injected loadLibrary and frida-gadget
Signing new APK.
Signed the new APK
Performing zipalign
Zipalign completed
Copying final apk from /var/folders/h1/rxkqmv9d69vg7j1cw8k1d13m0000gn/T/tmp8vw27ag0.apktemp.aligned.objection.apk to InsecureBankv2.objection.apk in current
directory...
Cleaning up temp files...
```

# SSL Pinning Bypass -> Objection

- We install the patched apk on the phone:

```
 ~/Desktop/mobile/tools/objection/ adb install InsecureBankv2.objection.apk
InsecureBankv2.objection.apk: 1 file pushed. 87.4 MB/s (10230246 bytes in 0.112s)
WARNING: linker: libhoudini.so has text relocations. This is wasting memory and prevents security hardening. Please fix.
        pkg: /data/local/tmp/InsecureBankv2.objection.apk
Success
```

- Run the app on phone, then type this command to expore it:

```
objection --gadget "com.android.InsecureBankv2" explore
```

```
     _     _         _   _
 ___| |_  |_|___ ___| |_|_|___ ___
| . | . | | | -_|  _|  _| | . |   |
|___|___|_| |___|___|_| |_|___|_|_|
        |___|(object)inject(ion) v1.4.3

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
com.android.insecurebankv2 on (google: 5.1) [usb] #
```

# SSL Pinning Bypass -> Objection

- With Objection, we can easily collect app information, for example the env command will print out the locations of the applications Files, Caches and other directories:

```
com.android.insecurebankv2 on (google: 5.1) [usb] # env

Name                  Path
--------------------  ------------------------------------------------------------
filesDirectory        /data/data/com.android.insecurebankv2/files
cacheDirectory        /data/data/com.android.insecurebankv2/cache
externalCacheDirectory  /storage/emulated/0/Android/data/com.android.insecurebankv2/cache
codeCacheDirectory    /data/data/com.android.insecurebankv2/code_cache
obbDir                /storage/emulated/0/Android/obb/com.android.insecurebankv2
packageCodePath       /data/app/com.android.insecurebankv2-1/base.apk
```

# SSL Pinning Bypass -> Objection

- Or list all of the Activities that the application has:

```
com.android.insecurebankv2 on (google: 5.1) [usb] # android hooking list activities
com.android.insecurebankv2.ChangePassword
com.android.insecurebankv2.DoLogin
com.android.insecurebankv2.DoTransfer
com.android.insecurebankv2.FilePrefActivity
com.android.insecurebankv2.LoginActivity
com.android.insecurebankv2.PostLogin
com.android.insecurebankv2.ViewStatement
com.android.insecurebankv2.WrongLogin
com.google.android.gms.ads.AdActivity
com.google.android.gms.ads.purchase.InAppPurchaseActivity

Found 10 classes
```

- Using the output from the activities list, invoking arbitrary activities is as simple as:

```
com.android.insecurebankv2 on (google: 5.1) [usb] # android intent launch_activity com.android.insecurebankv2.PostLogin
Launching Activity: com.android.insecurebankv2.PostLogin...
Launched: com.android.insecurebankv2.PostLogin
```

# SSL Pinning Bypass -> Objection

- Sorry, I'll back to our topic, to bypass SSL Pinning using Objection, simply type command (although this app have no pinning, just example ☺):

android sslpinning disable

```
com.android.insecurebankv2 on (google: 5.1) [usb] # android sslpinning disable
Job: ddd0adc5-872b-4f22-9c6f-84b1c0150a1a - Starting
[84b1c0150a1a] [android-ssl-pinning-bypass] Custom, Empty TrustManager ready
[84b1c0150a1a] [android-ssl-pinning-bypass] TrustManagerImpl
Job: ddd0adc5-872b-4f22-9c6f-84b1c0150a1a - Started
```

- We are all done! There are many cool features that objection has, feel free to explore it