ANDROID MOBILE SECURITY

TOPIC TO BE COVERED

- Basics of Android
- Data Storage on Android
- Static Mobile Application Analysis
- Dynamic Mobile Application Analysis
- Mobile OWASP Top 10 - 2016
- Mobile Malware

Basics of Android

BASICS OF ANDROID

- What is an Android App
- Android Architecture
- Android Components
- What is an APK file
- What is an Android Emulator

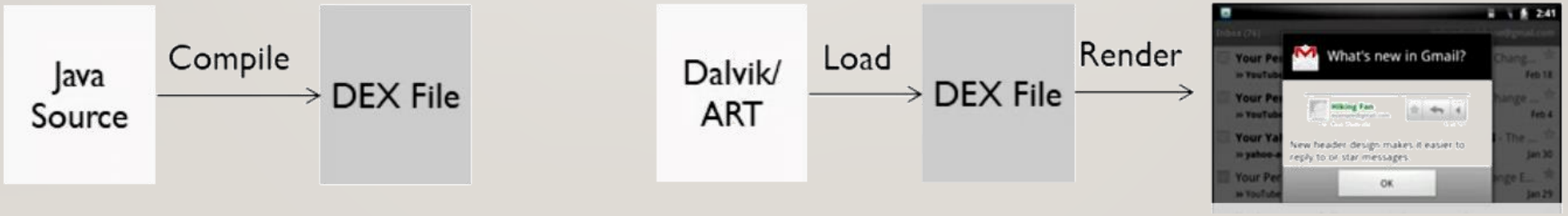
WHAT IS AN ANDROID APP

In Android, Apps are written in Java.

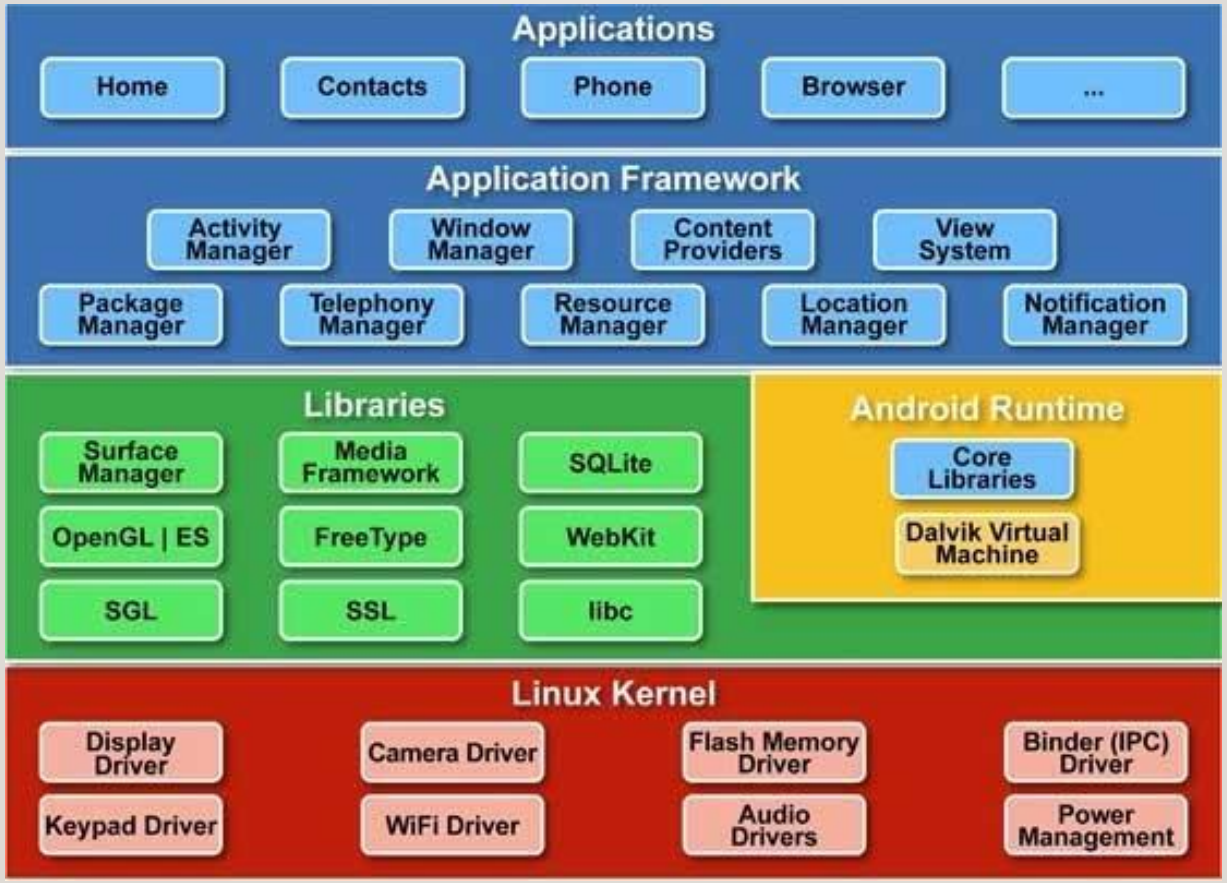
- Java as a language but not a runtime

Java apps are bytecode compiled into DEX format , platform-independent:

- Bytecode compiled apps are interpreted by Android virtual machine
  - Dalvik (Until Android 4.4,Kitkat)
  - ART (Android 5, Lollipop)



ANDROID ARCHITECTURE



ANDROID ARCHITECTURE

- **Application Layer :**The applications are at the topmost layer of the Android stack. An average user of the Android device would mostly interact with this layer.
- **Application Framework Layer** : Our applications directly interact with these blocks of the Android architecture. These programs manage the basic functions of phone like resource management, voice call management etc.
- **Native Libraries Layer** : The next layer in the Android architecture includes Android's native libraries. Libraries carry a set of instructions to guide the device in handling different types of data. For instance, the playback and recording of various audio and video formats is guided by the Media Framework Library.
- **Kernel Layer** : At the bottom of the Android stack is the Linux Kernel. It never really interacts with the users and developers, but is at the heart of the whole system.

ANDROID COMPONENTS

- **Activity**: An Activity provides a screen with which users can interact in order to do something. Users can perform operations such as making a call, sending an SMS, etc. Example: Login screen of your Facebook app.
- **Intent** : Used to launch an Activities , trigger Broadcast Receivers or communication with services.eg. Click on button.
- **Services** : Runs in background to perform long operations. Don't have user interface.
- **Broadcast Receivers** : Respond to announcements broadcasted by system or apps . Originate from system or apps. Don't have user interface. Gateway to other apps.eg. Battery Low, boot completed, headset plug etc.
- **Content Providers**: Allow applications to have centralized data and modify it.(Its like a DB) Eg. Contacts , calendar , system dictionary, photos etc.

WHAT IS AN APK FILE

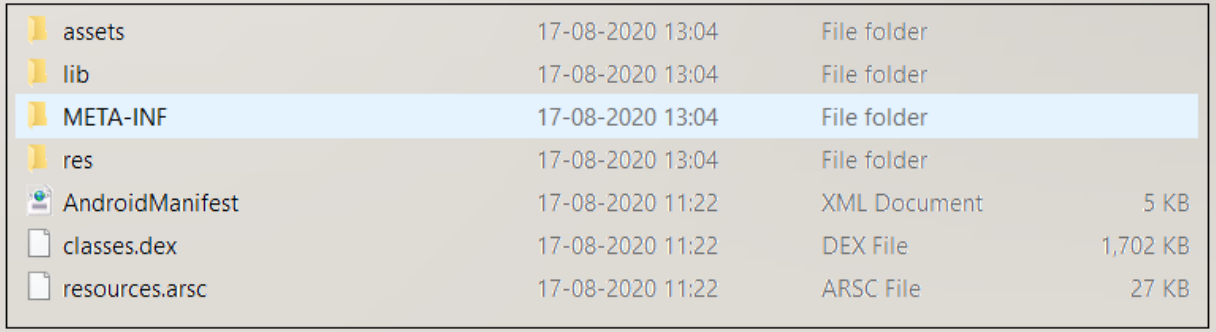
APK stands for Android Package Kit (also Android Application Package) and is the file format that Android uses to distribute and install apps. It contains all the elements that an app needs to install correctly on your device.

Just like EXE files on Windows, you can place an APK file on your Android device to install an app. Manually installing apps using APKs is called sideloading. Normally when you visit Google Play to download an app, it automatically downloads and installs the APK for you.

APK FILE CONTAINS

An APK file is an archive that usually contains the following files and directories:

- META-INF directory:
  - MANIFEST.MF: the Manifest file
- lib: the directory containing the compiled code that is platform dependent; the directory is split into more directories within it:
  - armeabi-v7a: compiled code for all ARMv7 and above based processors only
  - arm64-v8a: compiled code for all ARMv8 arm64 and above based processors only
  - x86: compiled code for x86 processors only
  - x86_64: compiled code for x86 64 processors only



APK FILE CONTAINS

- res: the directory containing resources not compiled into resources.arsc.
- assets: a directory containing applications assets, which can be retrieved by Asset Manager.
- AndroidManifest.xml: An additional Android manifest file, describing the name, version, access rights, referenced library files for the application. This file may be in Android binary XML that can be converted into human-readable plaintext XML with tools such as apktool, or Androguard.
- classes.dex: The classes compiled in the dex file format understandable by the Dalvik virtual machine and by the Android Runtime.
- resources.arsc: a file containing precompiled resources, such as binary XML for example.

WHAT IS AN ANDROID EMULATOR

The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.

The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.

Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.

The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

WHAT IS AN ANDROID EMULATOR

There are various emulator available for Android.

Below are some commonly use emulators

- Android Studio's Emulator
- Genymotion
- NOX
- MEmu
- KoPlayer
- BlueStacks

Question & Answers

QUESTION & ANSWERS

1. Which of the following is an Android component ?
   a. Activity Manager b. Activity
   c. Display Driver d. SQLite
2. How many layers are there in Android Architecture ?
3. What is an APK file contains ?
4. What is the use of android emulator ?

Data Storage on Android

DATA STORAGE ON ANDROID

- Some Important File Locations for Android
- Android Data Storage
- Android Logs
- Android Backup

SOME IMPORTANT FILE LOCATIONS FOR ANDROID

- **/data/data** : All applications data installed by user
- **/data/app** : APKs of applications installed by user
- **/system/app** : System Applications
- **/data/system** : Configuration of your phone, Pattern lock etc.
- **/data/local/tmp** : Writeable (without root)

ANDROID DATA STORAGE

Android applications can store data in many different locations and formats:

- Shared Preferences
- SQLite DB
- Internal and External Storage

Important to analyze them for sensitive data

ANDROID DATA STORAGE: SHARED PREFERENCES

Used for small pieces of data

- Structured as key-values pairs
- Stored in XML format

Can be encrypted through Android Security library

- Keysets hold data encryption keys
- Keysets protected by master key

Common security issues

- Cleartext storage of username and password

```
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs # ls -al
total 24
drwxrwx--x 2 u0_a68 u0_a68 4096 2021-03-01 05:45 .
drwxr-x--x 6 u0_a68 u0_a68 4096 2021-03-01 05:45 ..
-rw-rw---- 1 u0_a68 u0_a68  152 2021-03-01 05:45 jakhar.aseem.diva_preferences.xml
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs # cat jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">passwd</string>
    <string name="user">test</string>
</map>
```

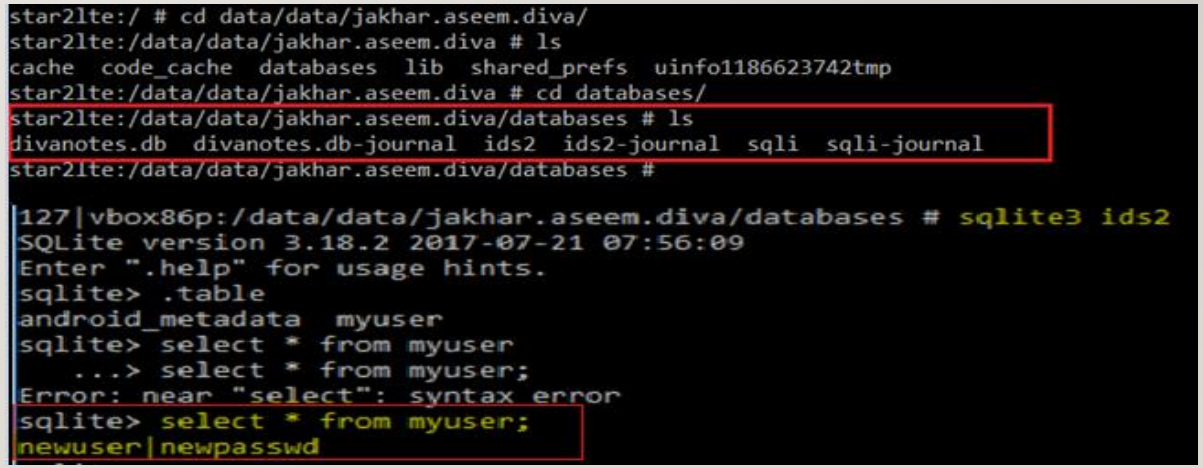
ANDROID DATA STORAGE: SQLITE DB

Android offers an SQLite API to save data in SQLite

- Useful for structured data
- No encryption by default

Full database can be encryption done using SQLCipher

- Key derived from passphrase



ANDROID DATA STORAGE: INTERNAL AND EXTERNAL STORAGE

File can be stored in internal or external storage

Internal Storage	**External Storage**
Accessible only to App itself	Accessible to every App
Deleted if app is uninstalled	Not deleted if app is uninstalled
Avoid plaintext storage of sensitive data res/values/strings.xml Build config file(local.properties)	Application need to request permission to store data in external storage Found in Android Manifest

ANDROID LOGS

Android provides a logging mechanism used by many mobile apps

Apps can use the Log class to create different debug messages

Security issues arise when sensitive information is included in logs

- In Android < 4.1, applications have access to logs of the applications
- In Android > 4.1, still possible on rooted phones or when android:debuggable is True

Use the following tools to access device logs:

- adb logcat
- dmesg
- dumpsys

```
# adb logcat
09-02 15:47:56.986  6159  6333 W VideoCapabilities: Unrecognized profile 4 for video/hevc
09-02 15:47:57.016  6159  6333 W VideoCapabilities: Unrecognized profile 2130706433 for
video/avc
09-02 15:47:57.033  6159  6333 I VideoCapabilities: Unsupported profile 4 for video/mp4v-
es
09-02 15:47:57.840   901  6353 D NetworkMonitor/NetworkAgentInfo [WIFI () - 103]:
PROBE_DNS www.google.com 7ms OK 172.217.19.196
09-02 15:47:57.846   901  6354 D NetworkMonitor/NetworkAgentInfo [WIFI () - 103]:
PROBE_DNS connectivitycheck.gstatic.com 12ms OK 172.217.168.195
09-02 15:47:58.063  6159  6269 D o        : queryLoginServer, url = https://cloud-
us.sanstest.org/api/login?user=testUser&password=qwerty123Secret
09-02 15:48:00.051  6159  6270 I CrashlyticsCore: Crashlytics report upload complete:
5D4BEBA400DE-0001-1699-850FF2A356D6
```

ANDROID BACKUP

Android 4 and later backup feature

- Required USB Debug

Optional AES-256 encryption backup with passphrase

Backed up without password, extract header to reveal tar file

```
C:\Users\jwright>adb backup -all
Now unlock your device and confirm the backup operation.

C:\Users\jwright>dir backup.ab
06/13/2017  10:55 PM           224,024 backup.ab
```

ANDROID BACKUP ACCESS

```
# ls -l backup.ab
-rw-r--r-- 1 root root 224024 Jun 13 23:18 backup.ab
# dd if=backup.ab bs=24 skip=1 | openssl zlib -d > backup.tar
9333+1 records in
9333+1 records out
224000 bytes (224 kB) copied, 0.141182 s, 1.6 MB/s
# tar xf backup.tar
# ls apps
android                                com.android.music
com.android.browser                    com.android.protips
com.android.calculator2                com.android.providers.applications
com.android.calendar                   com.android.providers.calendar
com.android.camera                     com.android.providers.downloads
com.android.contacts                   com.willhackforsushi.lunarlander
com.android.launcher                   jp.co.omronsoft.openwnn
com.android.mms
```

Question & Answers

QUESTION & ANSWERS

1. On which file location you can find user installed application data ?
2. Which of the file location you can be accessed on non rooted device ?

   a. /data/app b. /data/system

   c. /system/app d. /data/local/tmp

3. Which format data get stored in share preference file ?
4. What kind of issue can be find in android application log ?

Static Mobile Application Analysis

STATIC MOBILE APPLICATION ANALYSIS

- **Manual Static App Analysis**
- **Automating Static App Analysis**
- **Obfuscation**

STATIC MOBILE APPLICATION ANALYSIS

Evaluating an app by examining the executable itself

- No need to execute the binary to evaluate it

During static analysis, the mobile app's source code is reviewed to ensure appropriate implementation of security controls.

Below approaches are used for static mobile application analysis:

- Manual Static App Analysis
- Automating Static App Analysis

STATIC MOBILE APPLICATION ANALYSIS

- Manual Static App Analysis : Manual static analysis of apps gives you the best security details but its time consuming and required skill set.

- Automating Static App Analysis : Automating static analysis of apps with third-party tools are faster but often limited.

Manual Static App Analysis

MANUAL STATIC APP ANALYSIS : REVERSE ENGINEERING OF ANDROID APPLICATION

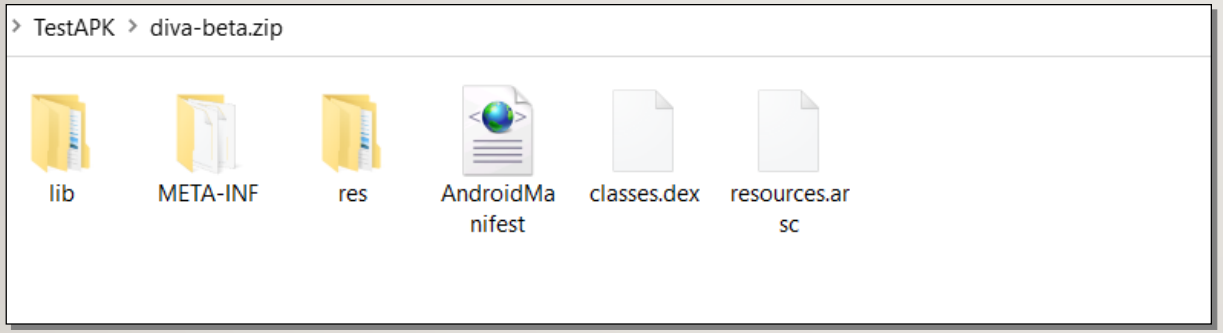
Reverse engineering an android application can give an understanding of how the application really works in the background and how it interacts with the actual phone. This knowledge would assist in the process of discovery vulnerabilities that exist in the code.

In order to start the reversing process of the APK file, we need some reverse engineering tools:

- Dex2jar
- JD-GUI
- APKTool
- JADX
- ADB

REVERSE ENGINEERING TOOLS: DEX2JAR

Android binaries stored in APK format. In order to see the files inside the APK we need to change its extension from .apk to .zip and extract it

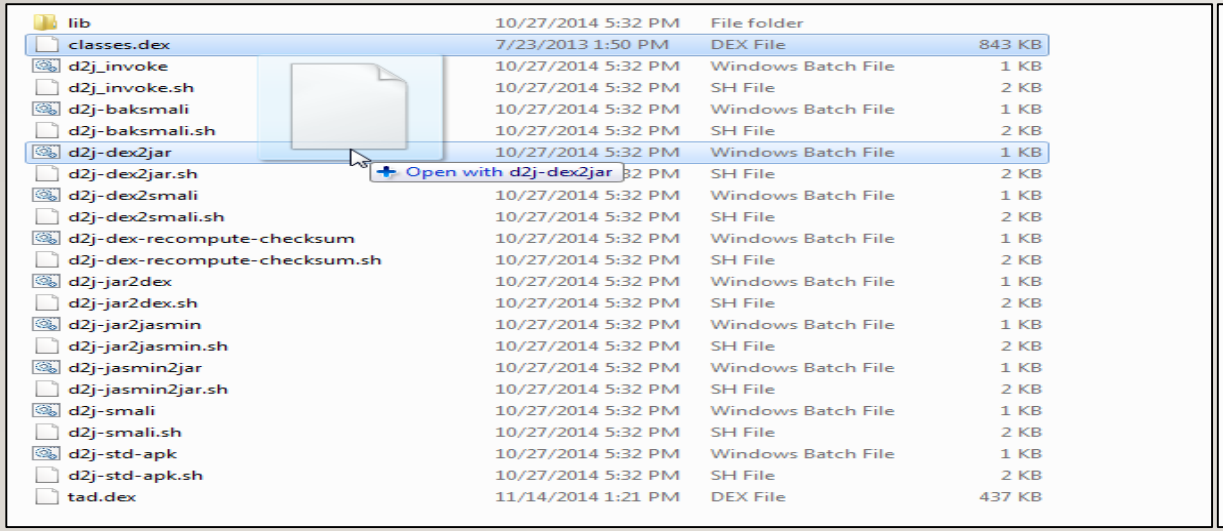


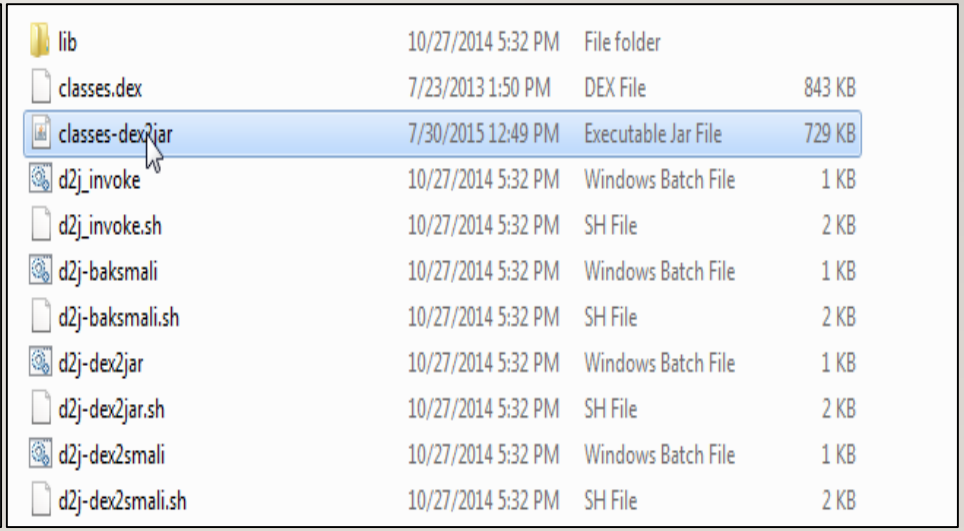
The dex files are Dalvik executable files format, and are not human readable. To convert classes.dex we can use an open source tool "dex2jar".

REVERSE ENGINEERING OF ANDROID APPLICATION: DEX2JAR

Just DRAG the classes.dex file and DROP on **d2j-dex2jar.bat file**. It creates a jar file for the corresponding app as shown below Or type the following code in command window –

d2j-dex2jar classes.dex



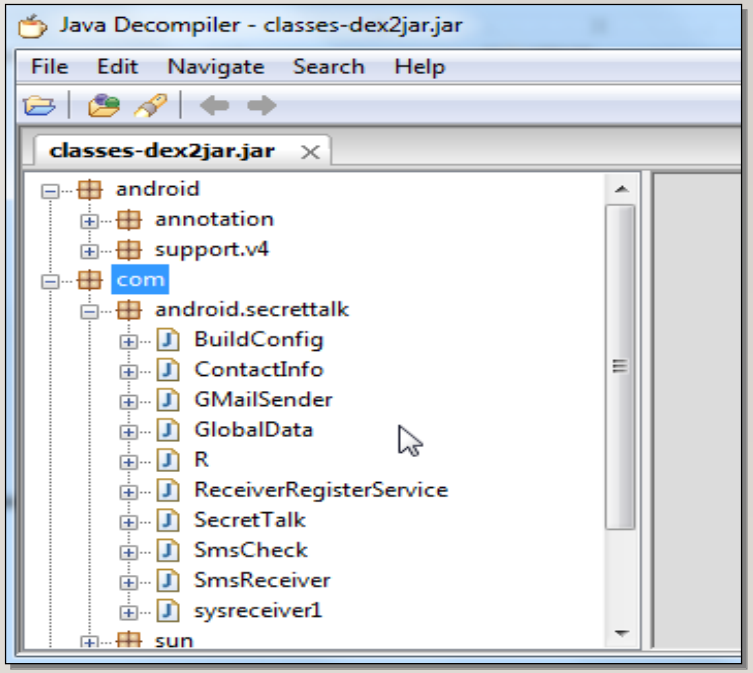


REVERSE ENGINEERING TOOLS: JD-GUI

JD-GUI is a standalone graphical utility that displays Java source codes of ".class" files. You can browse the reconstructed source code with the JD-GUI for instant access to methods and fields.

We just need to open the classes-dex2jar.jar file

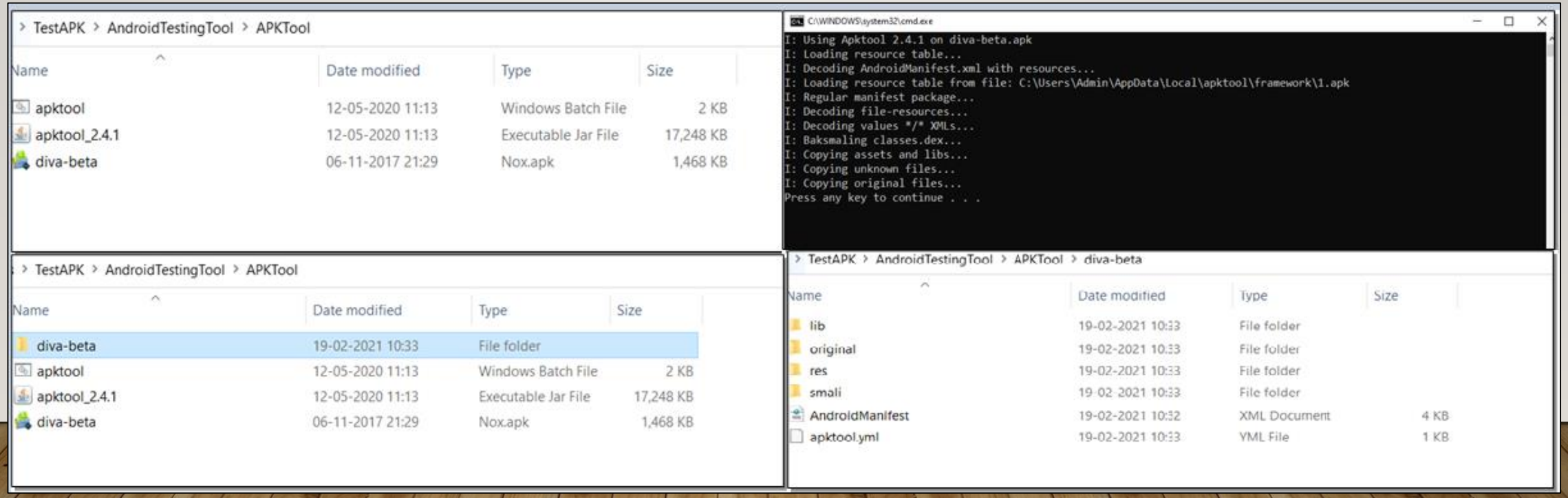
in JD-GUI tool.



REVERSE ENGINEERING TOOLS: APKTOOL

APKTool is a java-based tool which compile and decompile android APK. It works on Windows, Linux, and MacOS.

Just Drag and drop APK file on APKTool windows batch file

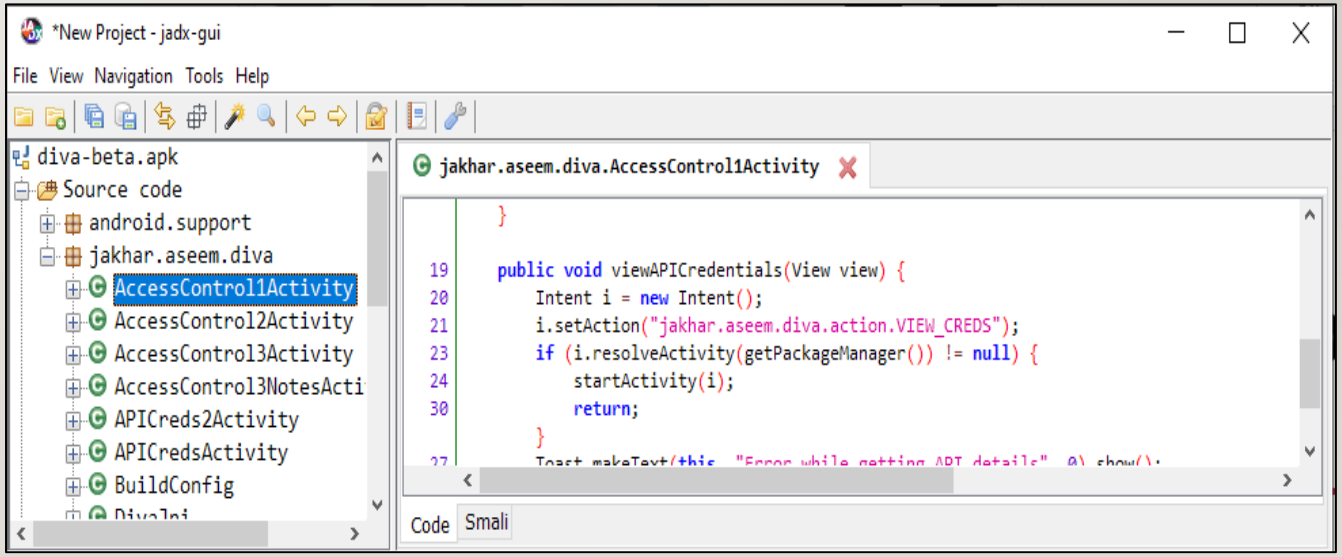


REVERSE ENGINEERING TOOLS: JADX

JADX decompile an APK file. It has convenient GUI and command line tool and it supports Windows, Linux and macOS

- Produces reconstructed source from the Android Application
- Supersedes older tools: DEX2JAR, JDGUI

We just need to open apk in JADX tool

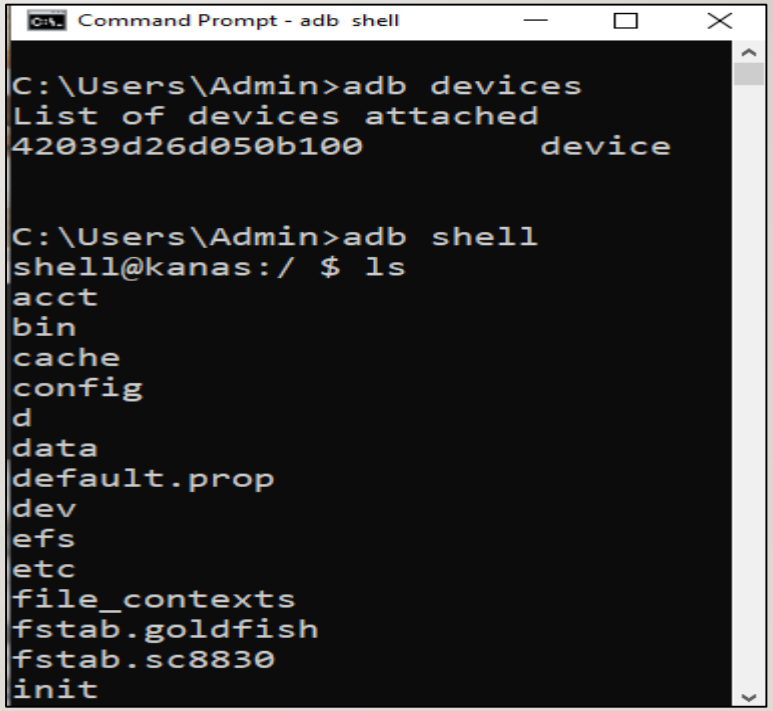


ANDROID DEBUG BRIDGE (ADB)

Android Debug Bridge (**adb**) is a versatile command-line **tool** that lets you communicate with a device. The **adb** command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device.

Commands of ADB

- adb devices
- adb connect ip.address
- adb shell
- adb install apkname
- adb uninstall apkname
- adb push file/directory
- adb pull filename
- adb logcat
- And many more



Automating Static App Analysis

AUTOMATING STATIC APP ANALYSIS : STATIC ANALYSIS TOOLS

Static analysis tools focus on the source code

Below are the some commonly used automated tools for static analysis:

- MobSF
- AndroBugs
- QARK
- Androsim

Each tool has strong and weak point

STATIC ANALYSIS TOOLS: MOBSF

Mobile Security Framework(MobSF) aims to be an all-in-one solution for static and dynamic analysis.

Gives a good overview of the application

- (Exported) Activities, Services, Receivers and Providers

Highlights potential issues

- Debug/backup enabled, unprotected receivers, sensitive information in logs etc.

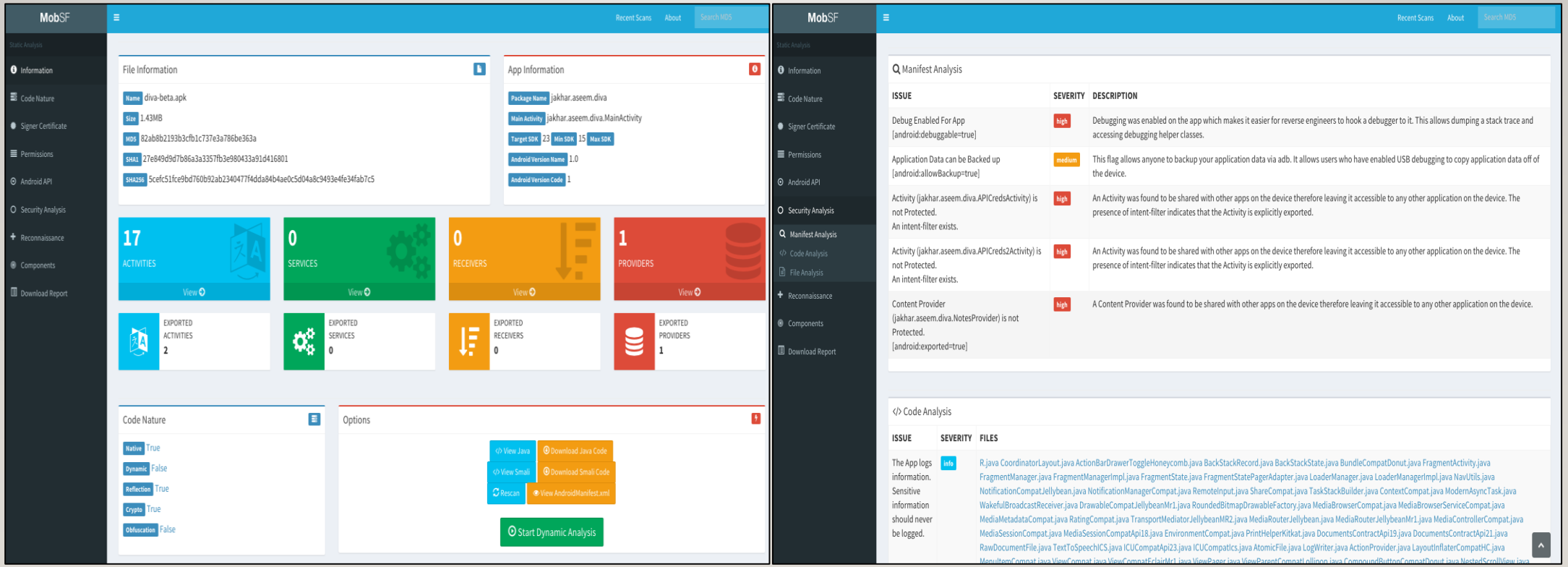
Typically many false positive, but still a great way to start an application analysis.

Dynamic analysis is less stable

MobSF can be download from below link

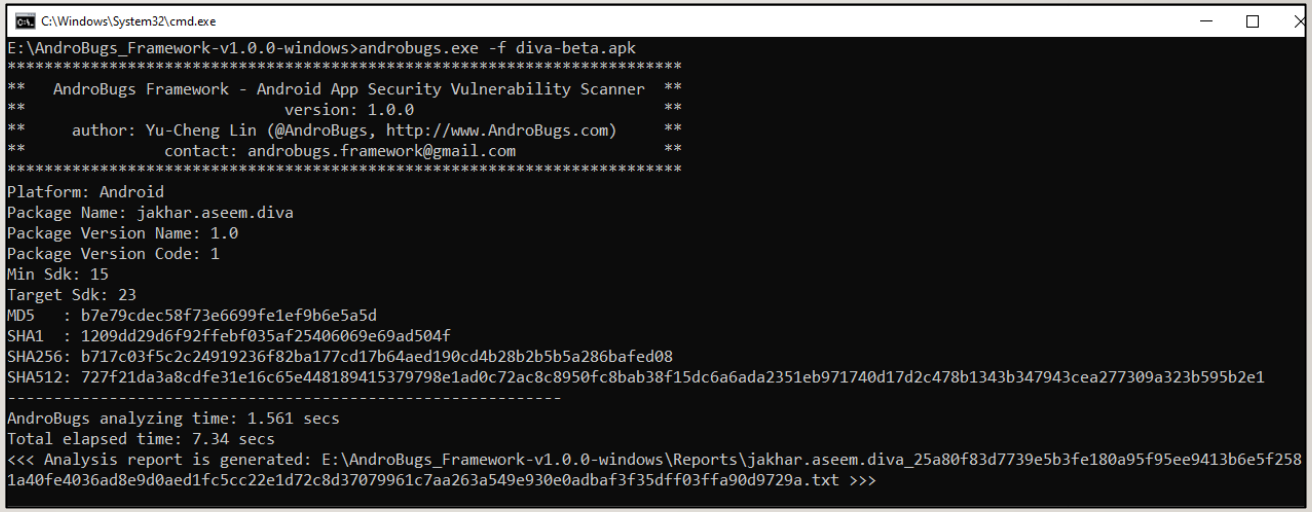
https://github.com/MobSF/Mobile-Security-Framework-MobSF

STATIC ANALYSIS TOOLS: MOBSF ANALYSIS RESULT



STATIC ANALYSIS TOOLS: ANDROBUGS

AndroBugs Framework is an Android vulnerability analysis system that helps developers or hackers find potential security vulnerabilities in Android applications. No splendid GUI interface, but the most efficient and more accurate.



AndroBugs Framework can be download from below link

https://github.com/AndroBugs/AndroBugs_Framework

STATIC ANALYSIS TOOLS: ANDROBUGS ANALYSIS RESULT

jakhar.aseem.diva_25a80f83d7739e5b3fe180a95f95ee9413b6e5f2581a40fe4036ad8e9d0aed1fc5cc22e1d72c8d37079961c7aa263a549e930e0adbaf3f35dff03ffa90d9729a.txt

```
********************************************************************************
**   AndroBugs Framework - Android App Security Vulnerability Scanner  **
**                            version: 1.0.0                          **
**     author: Yu-Cheng Lin (@AndroBugs, http://www.AndroBugs.com)     **
**               contact: androbugs.framework@gmail.com               **
********************************************************************************
Platform: Android
Package Name: jakhar.aseem.diva
Package Version Name: 1.0
Package Version Code: 1
Min Sdk: 15
Target Sdk: 23
MD5   : b7e79cdec58f73e6699fe1ef9b6e5a5d
SHA1  : 1209dd29d6f92ffebf035af25406069e69ad504f
SHA256: b717c03f5c2c24919236f82ba177cd17b64aed190cd4b28b2b5b5a286bafed08
SHA512: 727f21da3a8cdfe31e16c65e448189415379798e1ad0c72ac8c8950fc8bab38f15dc6a6ada2351eb971740d17d2c478b1343b347943cea277309a323b595b2e1
Analyze Signature: 25a80f83d7739e5b3fe180a95f95ee9413b6e5f2581a40fe4036ad8e9d0aed1fc5cc22e1d72c8d37079961c7aa263a549e930e0adbaf3f35dff03ff
--------------------------------------------------------------------------------------------------------
[Critical] <Debug> Android Debug Mode Checking:
           DEBUG mode is ON(android:debuggable="true") in AndroidManifest.xml. This is very dangerous. The attackers will be able to sniff
           the debug messages by Logcat. Please disable the DEBUG mode if it is a released application.
[Critical]  AndroidManifest ContentProvider Exported Checking:
           Found "exported" ContentProvider, allowing any other app on the device to access it (AndroidManifest.xml). You should modify th
           attribute to [exported="false"] or set at least "signature" protectionalLevel permission if you don't want to.
           Vulnerable ContentProvider Case Example:
             (1)https://www.nowsecure.com/mobile-security/ebay-android-content-provider-injection-vulnerability.html
             (2)http://blog.trustlook.com/2013/10/23/ebay-android-content-provider-information-disclosure-vulnerability/
             (3)http://www.wooyun.org/bugs/wooyun-2010-039169
                provider => jakhar.aseem.diva.NotesProvider
[Warning]  External Storage Accessing:
           External storage access found (Remember DO NOT write important files to external storages):
              => Ljakhar/aseem/diva/InsecureDataStorage4Activity;->saveCredentials(Landroid/view/View;)V (0x26) --->
                 Landroid/os/Environment;->getExternalStorageDirectory()Ljava/io/File;
[Warning]  AndroidManifest Exported Components Checking:
           Found "exported" components(except for Launcher) for receiving outside applications' actions (AndroidManifest.xml).
           These components can be initilized by other apps. You should add or modify the attribute to [exported="false"] if you don't wan
```

STATIC ANALYSIS TOOLS: QARK

QARK is a command line static analysis tool for Android applications

Reads APK files or source code and detects common vulnerabilities

- Use of world-readable or world-writable files
- Inadequately protected components
- Bad use of cryptography

Reports in different formats

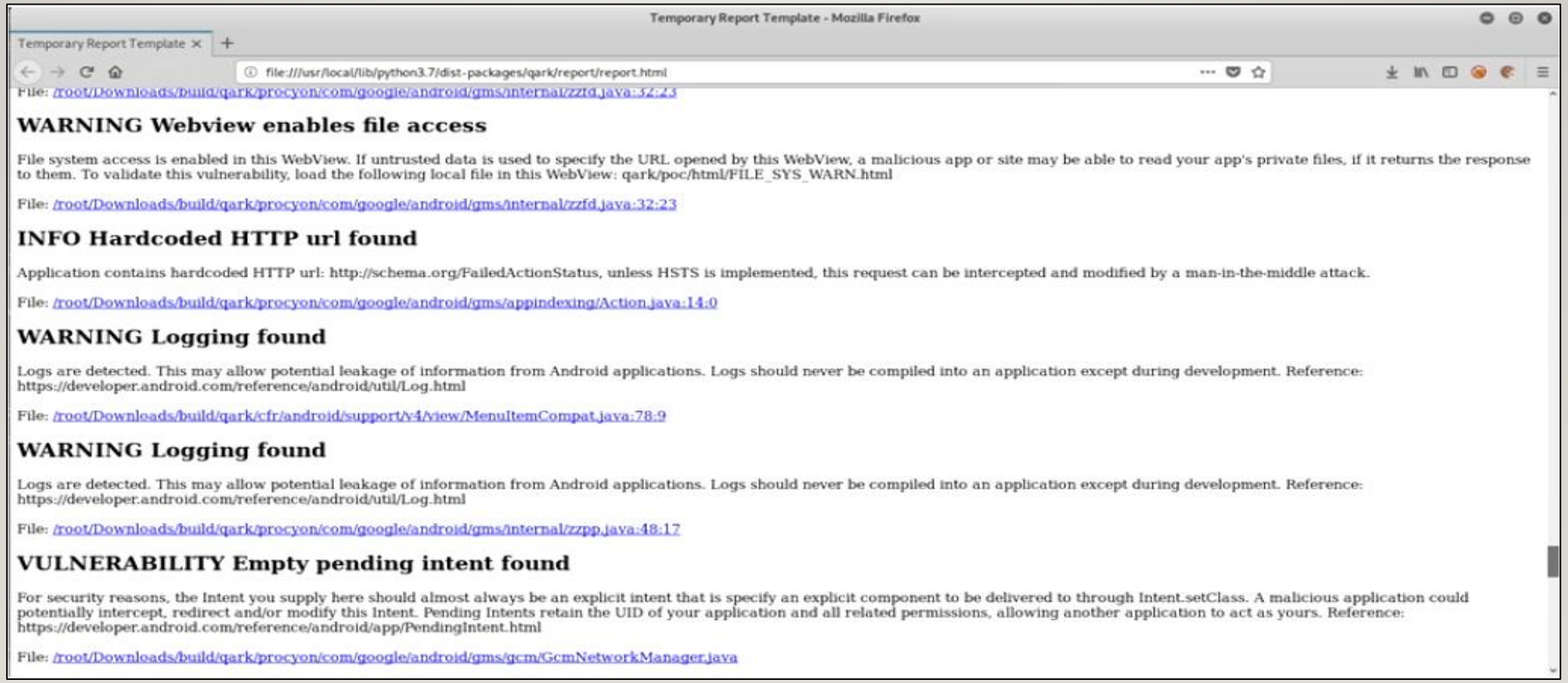
- HTML,XML etc.

Can create exploit APKs for detected vulnerabilities

QARK can be download from below link

https://github.com/linkedin/qark

STATIC ANALYSIS TOOLS: QARK ANALYSIS RESULT



STATIC ANALYSIS TOOLS: ANDROSIM

Androsim is a tool that compare two APKS

Useful for evaluating app rip-offs

- "How much of this app is copied from this legitimate app?"

Useful for evaluating changes between apps

- "How much changes were made between version 1.0 and 1.1?"
- "What's the difference between these two malware samples?"

STATIC ANALYSIS TOOLS: ANDROSIM ANALYSIS RESULT

```
C:\TEMP\Cajino-Baidu>androsim.exe -i Cajino_A.apk Cajino_B.apk -c ZLIB -n -d >Cajino.txt

Elements:
        IDENTICAL:    5726
        SIMILAR:      11
        NEW:          7
        DELETED:      0
        SKIPPED:      0
       --> methods: 99.984410% of similarities
SIMILAR methods:
       Lca/ji/no/method10/BaiduUtils; getCallLog (Landroid/content/Context;
Ljava/lang/String;)V 248
              --> Lca/ji/no/method2/BaiduUtils; getCallLog (Landroid/content/Context;
Ljava/lang/String;)V 252 0.0625
IDENTICAL methods:
       Lorg/apache/commons/logging/LogFactory$5; run ()Ljava/lang/Object; 53
              --> Lorg/apache/commons/logging/LogFactory$5; run ()Ljava/lang/Object; 53
NEW methods:
       Lca/ji/no/method2/BaiduUtils; call (Landroid/content/Context;
Ljava/lang/String;)V 48
       Lca/ji/no/method2/BaiduUtils; downloadFile (Landroid/content/Context;
Ljava/lang/String;)V 59
```

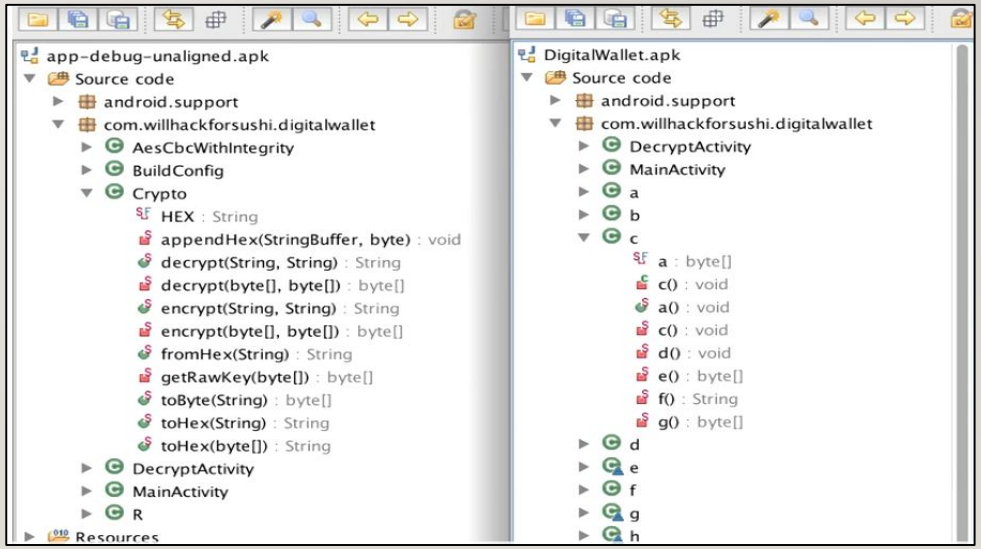
Androsim output edited and emphasized for display

OBFUSCATION

Obfuscation is the process of modifying an executable so that it is no longer useful to a hacker but remains fully functional. While the process may modify actual method instructions or metadata, it does not alter the output of the program. Obfuscation helps protecting your application against reverse engineering by others.

Below commonly used obfuscation tools for android app:

- ProGuard
- DexGuard
- DexProtector



Question & Answers

QUESTION & ANSWERS

1. ADB stands for -

   a. Android debug bridge
   b. Android delete bridge
   c. Android destroy bridge
   d. Android dos bridge

2. What are the tools required to perform manual static testing of an android mobile app ?

3. How to prevent reverse engineering of your mobile app ?

4. Which of the static testing approach should be use for android app analysis ?

   a. Manual static analysis
   b. Automated static analysis

Dynamic Mobile Application Analysis

DYNAMIC MOBILE APPLICATION ANALYSIS

- Some Important Concepts
- Intercepting SSL/TLS Traffic
- Modifying Mobile Application Code
- Mobile Application Runtime Manipulation
- Android Dynamic Analysis with Drozer

DYNAMIC MOBILE APPLICATION ANALYSIS

Static evaluation is valuable, but can become very complex

- Hard to follow code path
- Application may load dynamic code
- Application may require server interaction

Dynamic analysis can solve these issues

- Monitor application behavior by executing it
- Inspect local storage and platform interaction
- Modify the behavior of the application at runtime

DYNAMIC MOBILE APPLICATION ANALYSIS: SOME IMPORTANT CONCEPTS

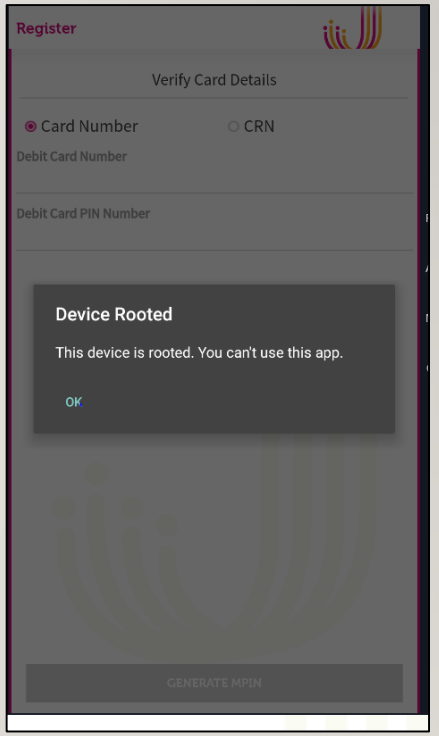
Before starting dynamic application analysis we should know below important concepts:

- Root Detection
- SSL Pinning
- Hooking

ROOT DETECTION

"Rooting" is the process by which one gains access to the administrative commands and functions of an operating system. It gives the ability (or permission) to alter or replace system applications, files, and settings, removing pre-installed applications, run specialized applications ("apps") that require administrator-level permissions.

Root-detection process is used in Android to prevent users from using their app on a rooted device. An APP/APK will implement different checks to determine whether the device is rooted or non-rooted, if the phone is rooted then the APK will display some message like "This device is rooted, you cannot use this app"



COMMON ROOT DETECTION CHECKS

Installed Packages: Check for the any of the below-installed packages on the mobile device at runtime

- supersu.apk
- Busybox
- Root Cloak
- Xpose framework
- Magisk

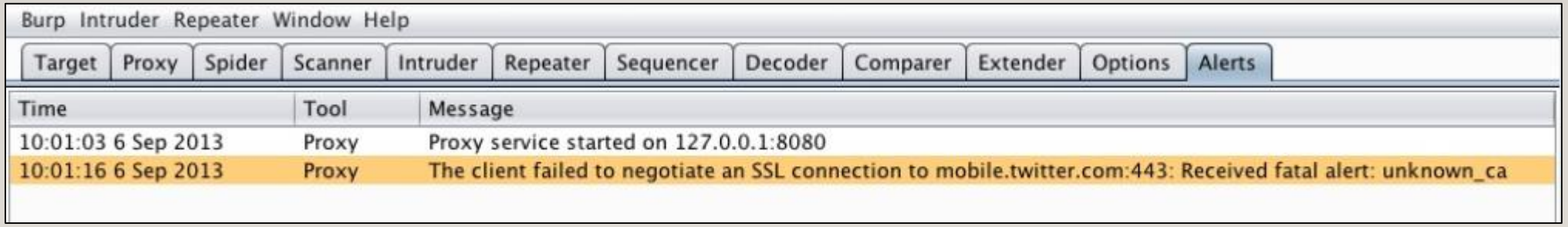
COMMON ROOT DETECTION CHECKS

Installed Files: Check for the installed files and directories on the device and its permissions

Superuser	Supersu
/su	/system/app/Superuser.apk
/system/bin	/system/bin/su
/system/sd/xbin	/system/xbin/su
/system/xbin	/data/local
/data/local/bin	/data/local/xbin
/sbin	/system/bin/failsafe
/vendor/bin	

SSL PINNING

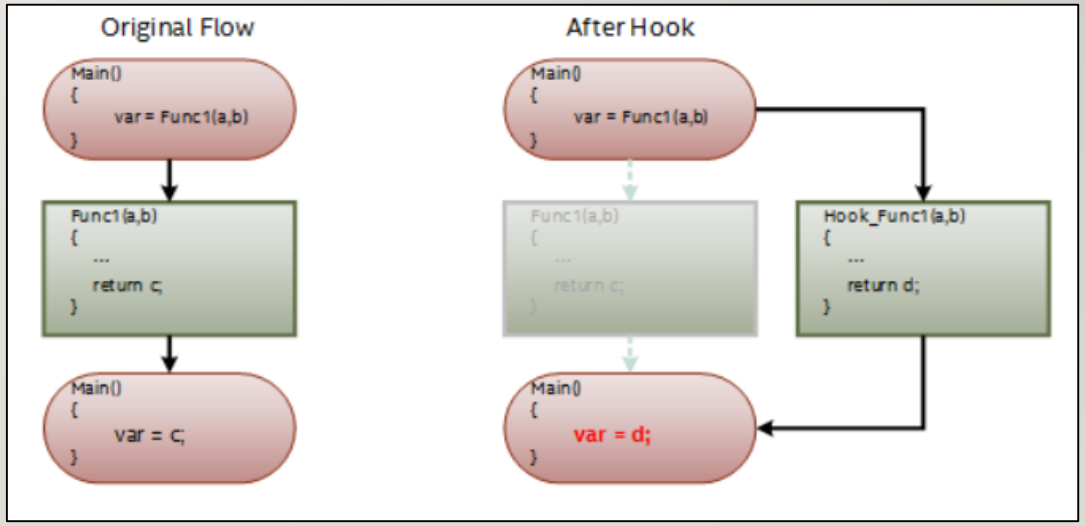
SSL Pinning is a technique that is used in the client side to avoid man-in-the-middle attack by validating the server certificates again even after SSL handshaking. The developers embed (or pin) a list of trustful certificates to the client application during development and use them to compare against the server certificates during runtime. If there is a mismatch between the server and the local copy of certificates, the connection will simply be disrupted, and no further user data will be even sent to that server. This enforcement ensures that the user devices are communicating only to the dedicated trustful servers.



HOOKING

Hooking is a term for a range of code modification techniques that are used to change the behavior of the original code running sequence by inserting instructions into the code segment at runtime

Hook can change running sequence for program



Intercepting SSL/TLS Traffic

INTERCEPTING SSL/TLS TRAFFIC

Intercepting SSL/TLS traffic from an application installed on our own device is very useful during mobile pen test:

- Inspect contents of HTTPS traffic
- Manipulation HTTPS requests

We can configure our device to send traffic to a proxy tools:

- Easier to do on rooted devices

Interception can be difficult when SSL pinning is used

INTERCEPTING SSL/TLS TRAFFIC USING PROXY TOOL: BURP SUITE

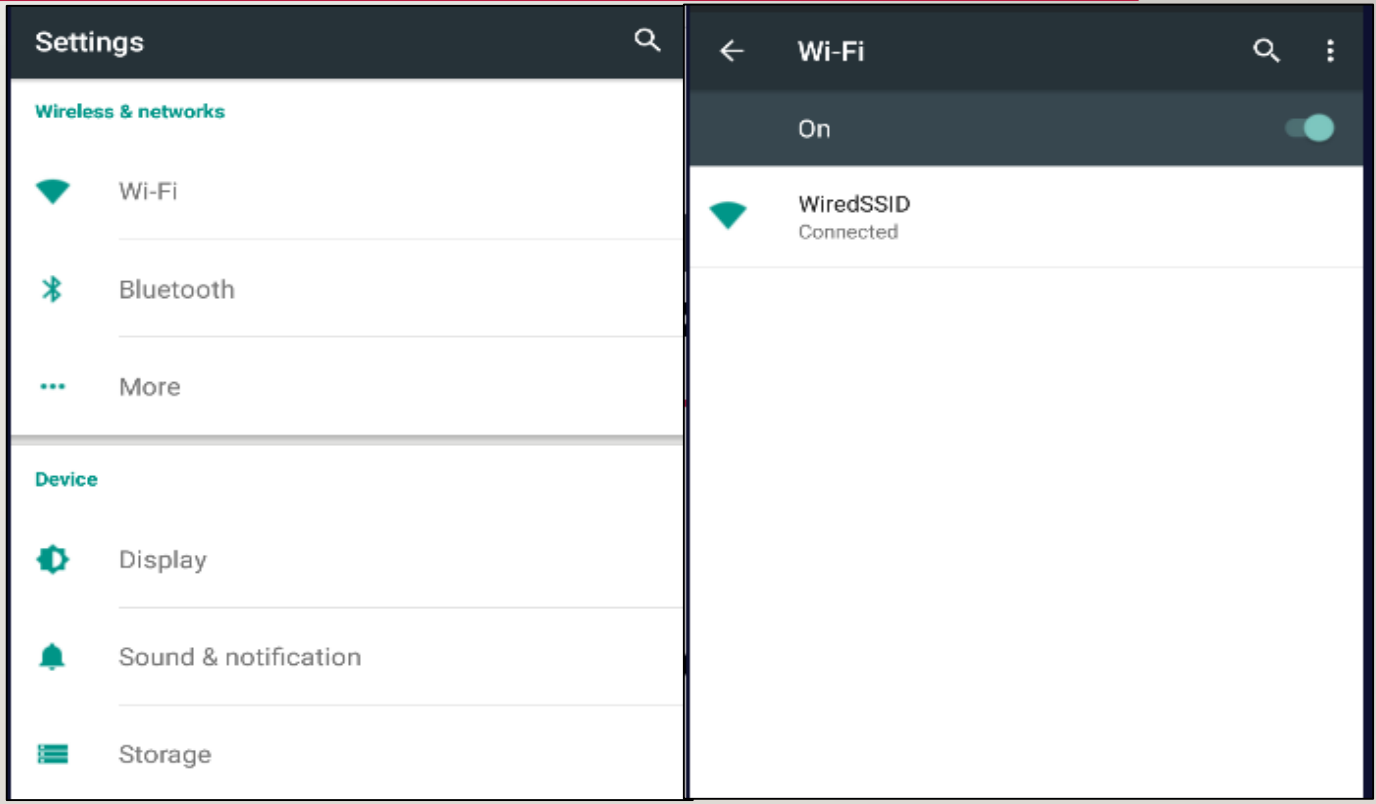
Burp Suite can be classified as an Interception Proxy. While using target mobile application, a penetration tester can configure their mobile device to route traffic through the Burp Suite proxy server. Burp Suite then acts as a (sort of) Man In The Middle by capturing and analyzing each request to and from the target mobile application so that they can be analyzed.

To intercept mobile device traffic first we need to configure it with burp suite

CONFIGURING AN ANDROID DEVICE TO WORK WITH BURP

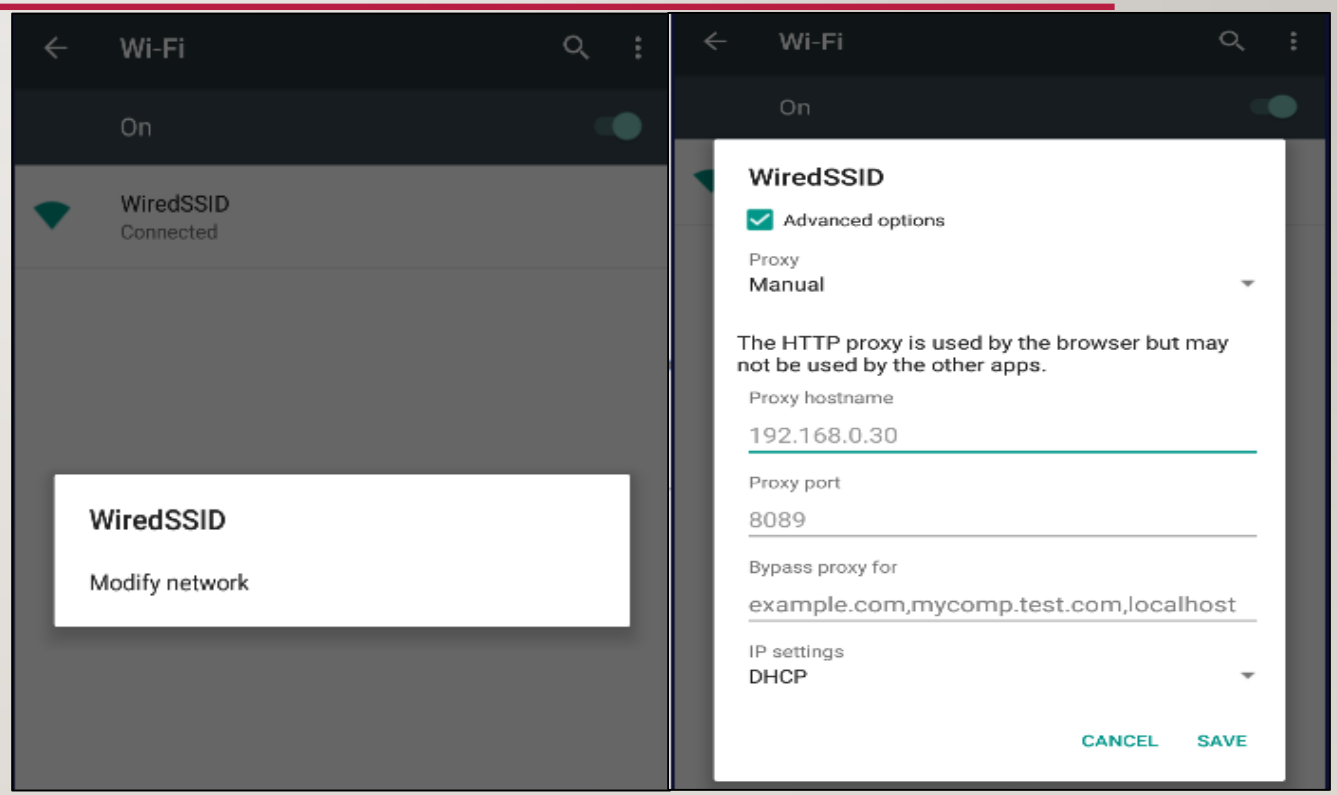
Configure your mobile device to use the proxy

- Open mobile **Settings** and click **Wi-Fi** in your mobile.
- Click on **Wi-Fi**



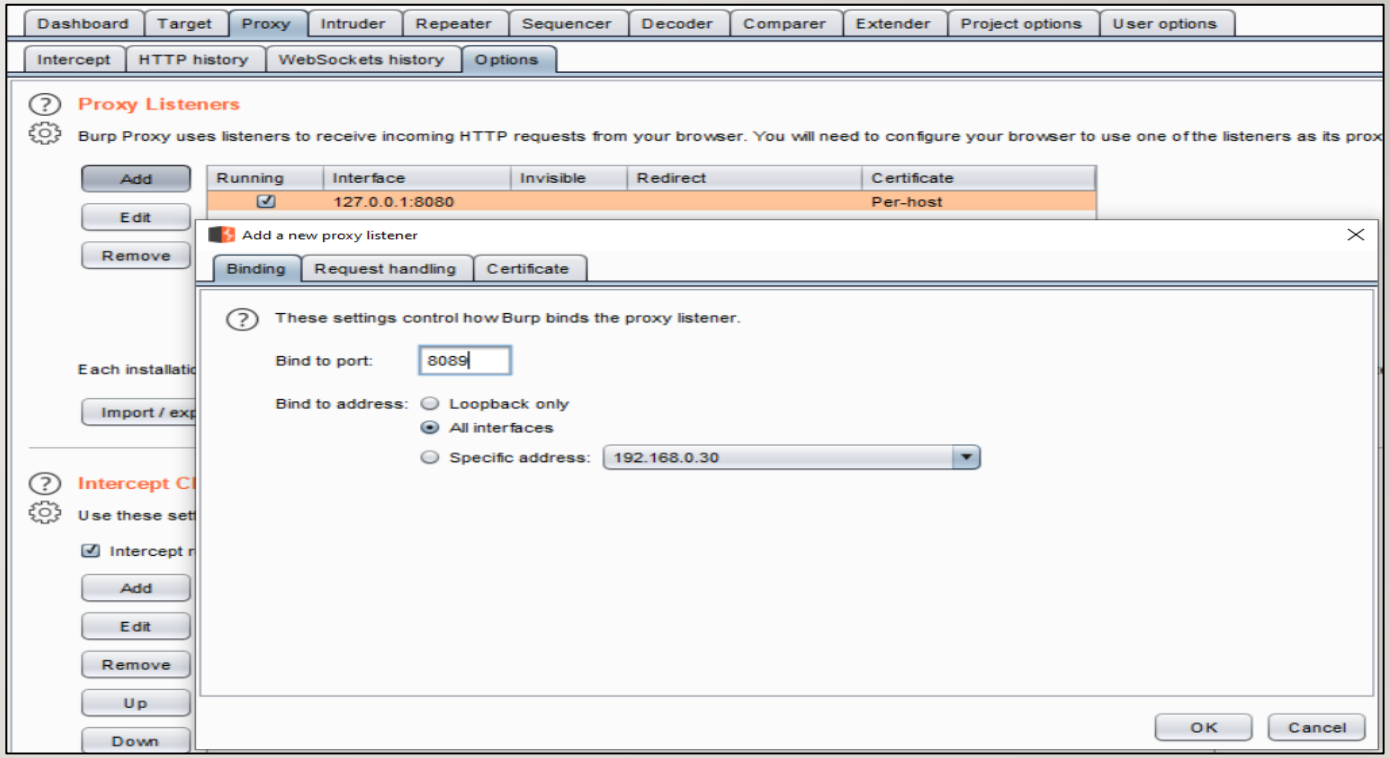
CONFIGURING AN ANDROID DEVICE TO WORK WITH BURP

- Long press on your Connected Network and **c**lick on **Modify network**
- Set **Proxy** as **Manual, Proxy hostname** and **Proxy port** (**Note**: Your need to set same hostname and port in Burp Suite) and click **SAVE**



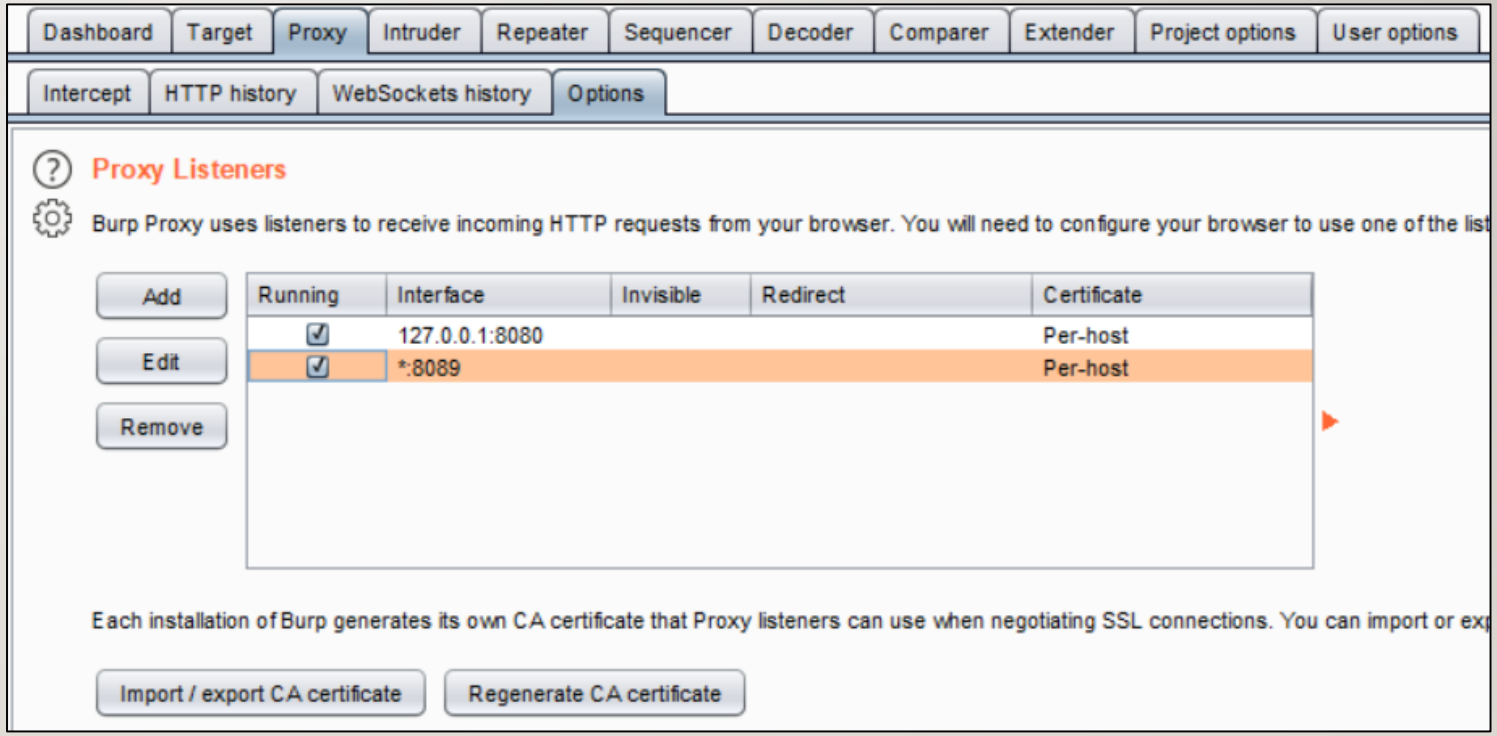
CONFIGURING AN ANDROID DEVICE TO WORK WITH BURP

- In Burp, go to the "**Proxy**" tab and then the "**Options**" tab and in the "**Proxy Listeners**" section, click the "**Add**" button.
- In the "**Binding**" tab, in the "Bind to port:" box, enter a port number that is not currently in use, e.g. "8089". Then select the "**All interfaces**" or "**Specific address**" option, and click "**OK**".



CONFIGURING AN ANDROID DEVICE TO WORK WITH BURP

- The Proxy listener should now be configured and running.



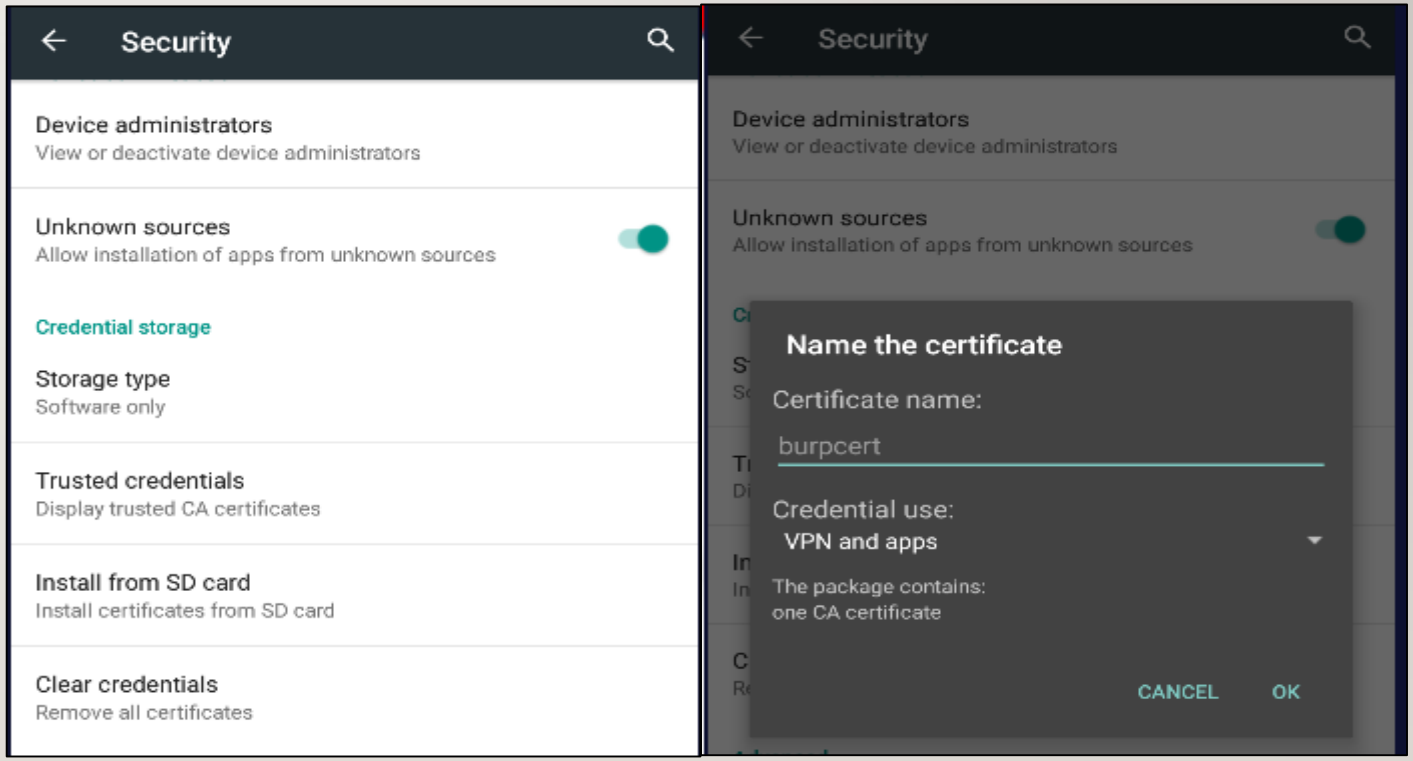
CONFIGURING AN ANDROID DEVICE TO WORK WITH BURP

Installing Burp's CA Certificate in an Android Device

- On your mobile browser with Burp running, visit http://burp and click the "CA Certificate" link. Save the certificate file and rename cacert.der to cacert.cer

CONFIGURING AN ANDROID DEVICE TO WORK WITH BURP

- In the "**Security**" menu click on the "**Install from device storage**"
- Click on downloaded burp certificate **cacert.cer**. Give certificate name , and click "**OK**".



Modifying Mobile Application Code

MODIFYING MOBILE APPLICATION CODE

Code manipulation can help us to analyze a mobile App

We can perform code manipulation on mobile Apps to achieve the following :

- Force an app to use HTTP, not HTTPS
- Disable emulator detection
- Disable root detection
- Remove SSL Pinning

MODIFYING MOBILE APPLICATION CODE

To perform Application manipulation we need to follow below steps:

- Decompile the Application APK using APKTool and JADX
- Analyze source code and perform code changes in any Application file or smali code(Dalvik Byte Code)
- Compile the modified Application code using APKTool
- Sign the modify Application APK

Root Detection Bypass Using Code Manipulation

SSL Pinning Bypass Using Code Manipulation

Question & Answers

QUESTION & ANSWERS

1. What is SSL Pinning ?
2. What is Hooking ?
3. What is Root Detection ?
4. How to setup proxy tool for an Android device ?

Mobile Application Runtime Manipulation

MOBILE APPLICATION RUNTIME MANIPULATION

Modifying and recompiling applications is time-consuming and doesn't allow modifications at runtime

Tools exist for Android to perform runtime manipulation

- No need to redeploy app
- Quick script development
- Live debugging and inspection
- Use application-specific and generic scripts

RUNTIME MANIPULATION WITH XPOSED

Xposed is a framework for modules that can change the behavior of the system and apps without touching any APKs. It allows users to easily apply add-ons (called Modules) to the ROM. Xposed framework can be used to perform below activities:

- Root detection bypass
- SSL Pinning bypass

Xposed framework requires rooted device.

Xposed framework can be download from below link

https://repo.xposed.info/module/de.robv.android.xposed.installer

Root Detection Bypass Using Xposed and SSLUnPinned

SSL Pinning Bypass Using Xposed and Root Cloak

RUNTIME MANIPULATION WITH FRIDA

Frida is a free dynamic instrumentation toolkit that enables software professionals to execute their own scripts in software that has traditionally been locked down ; i.e., proprietary (such as Android applications).

Frida allows you to hook functions to:

- Inspect arguments and return values
- Changes implementations

Frida requires rooted device.

Frida server can be download from below link

https://github.com/frida/frida/releases/

Root Detection Bypass Using Frida

SSL Pinning Bypass Using Frida

Question & Answers

QUESTION & ANSWERS

1. What are the different activities can be perform using application runtime manipulation ?
2. What are the different tools available for application runtime manipulation ?

RUNTIME MANIPULATION WITH OBJECTION

Objection is a runtime exploration toolkit built on top of Frida

Interactive console to interact with filesystem

Implements bypasses for common mobile hardening techniques

- Bypass SSL Pinning
- Bypass and simulate rooted environment

Objection requires rooted device.

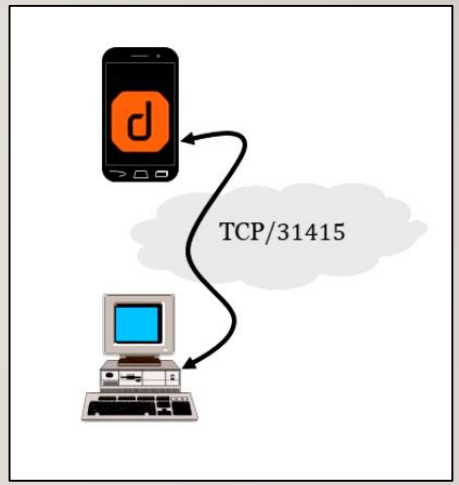
Root Detection Bypass Using Objection

SSL Pinning Bypass Using Objection

Android Dynamic Analysis with Drozer

DROZER FRAMEWORK

Drozer is the leading security assessment framework for the Android platform. Drozer is a Client/server framework for evaluating and exploiting Android applications



- Formerly, Mercury Framework
- Runs in emulator or physical devices

It is like Metasploit includes analysis task and exploit modules.

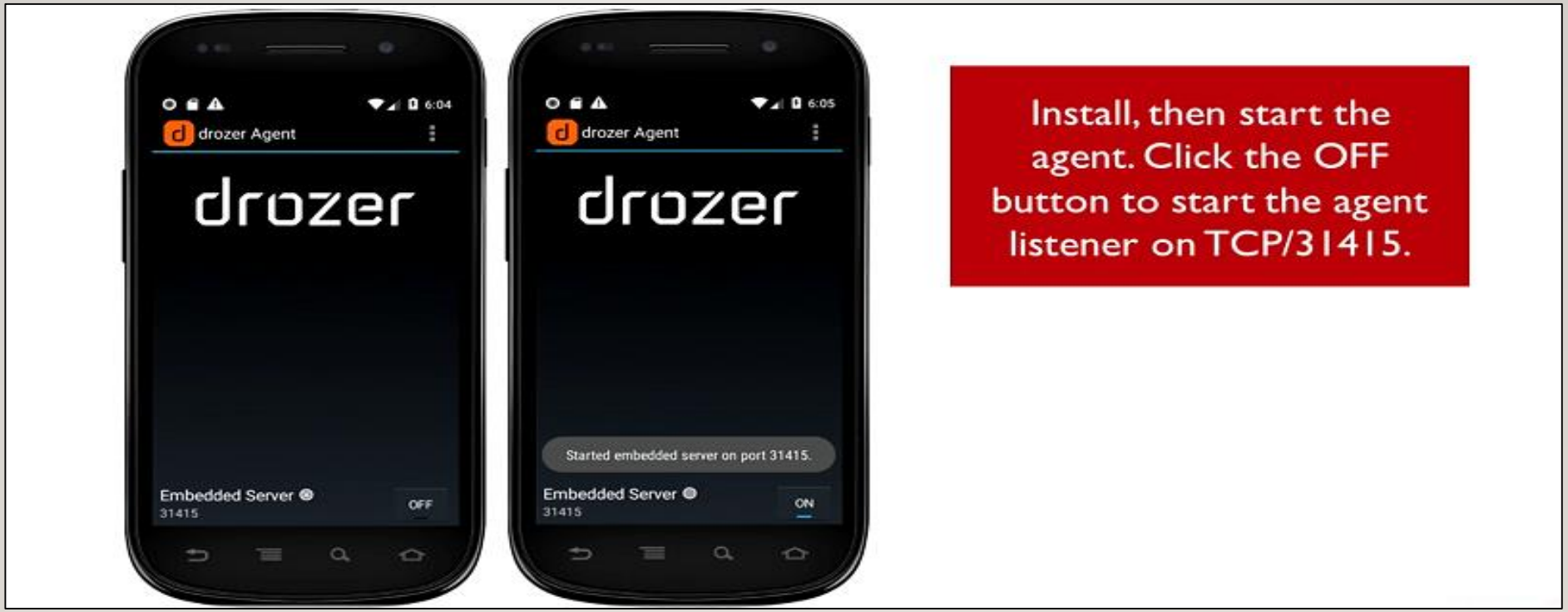
Opportunity to explore Android IPC mechanisms

- Identifying and crafting Intents

Other applications can invoke and manipulate components. Developers need to protect access to sensitive components

Drozer can be download from this link - **https://labs.f-secure.com/tools/drozer/**

DROZER AGENT



Download the latest Drozer Agent from below link

https://github.com/FSecureLABS/drozer/releases/download/2.3.4/drozer-agent-2.3.4.apk

DROZER CONSOLE : STARTUP

We have to do port forwarding before using drozer. As we know drozer by default uses port 31415. We will type command:

- adb forward tcp:31415 tcp:31415

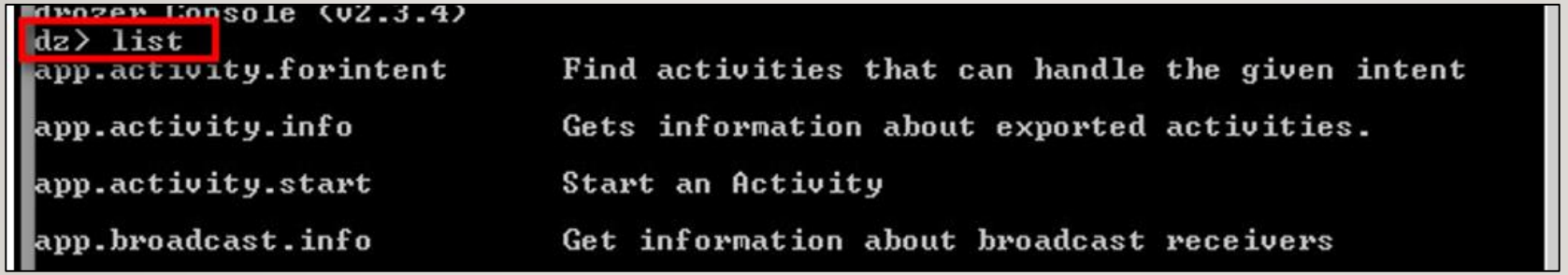
Now for using drozer, first open command prompt, and we will type command :

- drozer console connect (linux) or drozer.bat console connect (windows)

```
C:\>adb forward tcp:31415 tcp:31415
C:\>cd \drozer
C:\drozer>drozer.bat console connect
```

BASIC DROZER COMMANDS

list - shows the list of all Drozer modules that can be executed in the current session



run app.package.list - To list out all the packages installed on the emulator

BASIC DROZER COMMANDS

run app.package.info –a (package identifier) - To see some basic information about the package

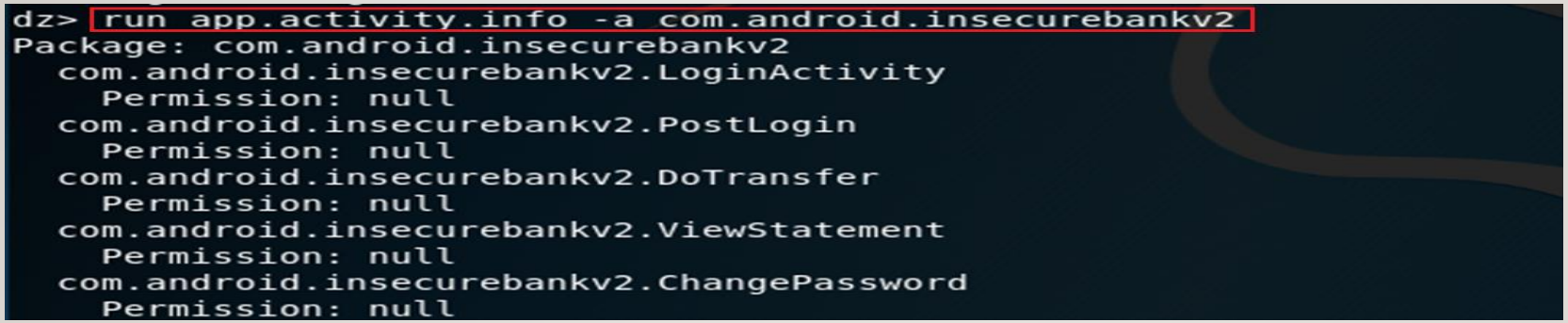
```
dz> run app.package.info -a com.mwr.example.sieve
Package: com.mwr.example.sieve
  Application Label: Sieve
  Process Name: com.mwr.example.sieve
  Version: 1.0
  Data Directory: /data/data/com.mwr.example.sieve
  APK Path: /data/app/com.mwr.example.sieve-1.apk
  UID: 10167
```

run app.package.attacksurface (Package identifier) – To identify the attack surface of our target application

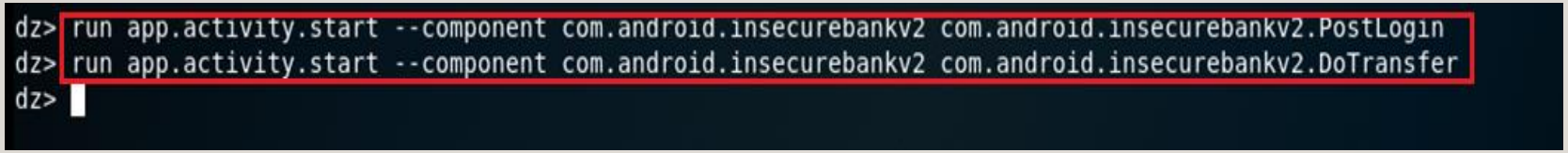
```
dz> run app.package.attacksurface com.android.insecurebankv2
Attack Surface:
  5 activities exported
  1 broadcast receivers exported
  1 content providers exported
  0 services exported
```

BASIC DROZER COMMANDS

run app.activity.info –a (Package identifier) - To get a list activities from a package



run app.activity.start --component (Package identifier) (activity name) - To launch any selected activity



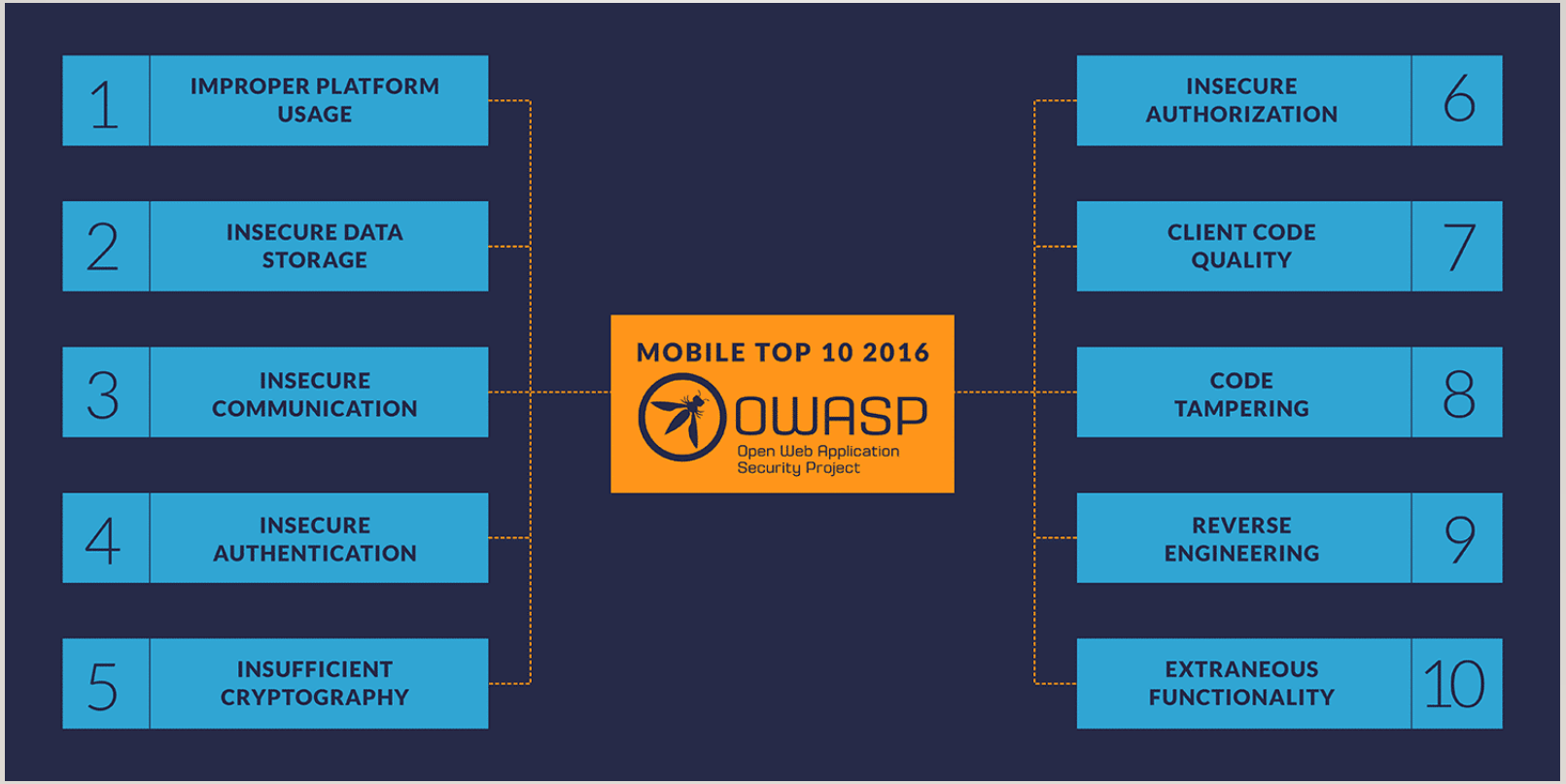
Question & Answers

QUESTION & ANSWERS

1. Which command is used to bypass SSL Pinning in Objection Framework ?
2. What is the default port number used by Drozer ?
3. Which command is used to do port forwarding in drozer ?

Mobile OWASP Top 10 - 2016

MOBILE OWASP TOP 10 - 2016



MOBILE OWASP TOP 10 - 2016

1. Improper Platform Usage : This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of Touch ID, the Keychain, or some other security control that is part of the mobile operating system.

2. Insecure Data Storage : Threats agents include the following: an adversary that has attained a lost/stolen mobile device; malware or another repackaged app acting on the adversary's behalf that executes on the mobile device.

3. Insecure Communication : In mobile application, data is commonly exchanged in a client-server fashion. When the solution transmits its data, it must traverse the mobile device's carrier network and the internet. Threat agents might exploit vulnerabilities to intercept sensitive data while it's traveling across the wire.

MOBILE OWASP TOP 10 - 2016

4. Insecure Authentication : Threat agents that exploit authentication vulnerabilities typically do so through automated attacks that use available or custom-built tools.

5. Insufficient Cryptography : Threat agents include the following: anyone with physical access to data that has been encrypted improperly, or mobile malware acting on an adversary's behalf.

6. Insecure Authorization : Threat agents that exploit authorization vulnerabilities typically do so through automated attacks that use available or custom-built tools.

7. Client Code Quality : Threat Agents include entities that can pass untrusted inputs to method calls made within mobile code. These types of issues are not necessarily security issues in and of themselves but lead to security vulnerabilities. Poor code-quality issues are typically exploited via malware or phishing scams.

MOBILE OWASP TOP 10 - 2016

8. Code Tampering : Typically, an attacker will exploit code modification via malicious forms of the apps hosted in third-party app stores. The attacker may also trick the user into installing the app via phishing attacks.

9. Reverse Engineering : An attacker will typically download the targeted app from an play store and analyze it within their own local environment using a suite of different tools.

10. Extraneous Functionality : Typically, an attacker seeks to understand extraneous functionality within a mobile app in order to discover hidden functionality in in backend systems. The attacker will typically exploit extraneous functionality directly from their own systems without any involvement by end-users.

Refer below link for more details on Mobile OWASP TOP 10

https://owasp.org/www-project-mobile-top-10/

Mobile Malware

MOBILE MALWARE

Mobile device malware is a growing threat for devices

- New opportunities for exploiting
- New opportunities for attack financial gain

Popular distribution method for Android malware is fake installers

Impersonates a legitimate application, bundled with malicious activity

MOBILE MALWARE DEFENCES

Below are the common ways to defend mobile malware

- Antivirus or anti-malware tools are available for Android
- Users should be trained to identify suspicious application
- Download App from trusted sources

Question & Answers

QUESTION & ANSWERS

1. What is OWASP Top 10 2016 mobile vulnerabilities ?
2. What are the common ways to defend mobile malware ?

DO YOU HAVE ANY QUESTIONS ?


Thank
You