Mobile Application Pentesting-Part 1

Piyush Patil
May 17 · 5 min read ★

I will cover all the most common attacks found in mobile applications.

-Android architecture and filesystem
-Reverse engineering android application(Apktool, Dex2jar, Jadx)
-Signing Application(Keytool, Jarsigner)
-ADB(Android Debugger)
-Insecure Logging(Logcat)
-Hardcoding Issues
-Shared Object Files/Libraries vulnerability
-Insecure Data Storage(Preferences, SQL, Permissions)
-Input Validation Issues(SQL, XSS, Directory Traversal, Webview, Buffer Overflow)
-Access Control Issues(Intent filter vulnerability)
-Drozer(Activities enumeration, Exploiting android broadcast receiver, Content provider enumeration, Service enumeration)
-Andbug and JDB(Java Debugger)
-Android Backup Vulnerability
-Bypassing Certificate Pinning

. . .

Android Architecture

https://developer.android.com/topic/libraries/architecture

The architecture works in such a way that the bottom layer supports the upper layer.

Android runs on a two-tier security model which is a combination of Linux based and Android-based model.

Linux model provides security and each application has its own UID and GID.

Android model provides permissions like accessing SD card or Make calls etc.

Android permissions are defined in filename AndroidManifest.xml file. Its created and specified by the developer. It also contains information like the minimum Android version required to run the application and permission like READ_EXTERNAL_STORAGE and etc.

. . .

Android application components

*Activities

-Visual screens of Android application

-Contains different layouts

-Anything you could interact with

Services

Another important component of an Android application is service. It does not provide the user interface. It does long-running operations in the background. Service doesn't terminate even if the component which initiated it got terminated or switched to another application. A service can be connected to a component which can even do interprocess communication (IPC). For example, when you receive your email updates in inbox it is a service. You get the notification of new e-mail even if you are not using the e-mail app or doing something else

Intents

-In order for activities and services to communicate with each other, we need intents

-To bind different Android components

-Used to perform a different kind of actions

-Changing activities, invoking activities in another application, starting an action, etc.

-Intent Filters

Activity Manager

-SQLite, XML, Plaintext

Shared preferences

-way of storing data in android application

-XML file with name value pairs

-Located in the shared_prefs folder inside the app directory

Broadcast Receivers

-Receives broadcast from various events

-Used by lot malware: can listen to SMS received and trigger some action

. . .

Practical Approach: Learning While Exploiting

We will test all vulnerabilities in the Android application by using vulnerable applications like Diva and InsecureBank.

Let's start…

Diva

Download the application from the following link:

https://github.com/payatu/diva-android

- Profiling apps.

adb devices =>**to see all devices**

Connecting to android phone

adb connect ip

Or

adb -s ip shell

Uploading and Downloading Files

adb push yo.txt /mnt/sdcard/yoyo.txt

adb pull /mnt/sdcard/yoyo.txt yo.txt

Taking backup of any application

adb logcat

adb logcat | grep -i process_id

Starting and Stopping the ADB server

adb start-server

adb kill-server

Installing and Uninstalling Application

adb install appname.apk

Reverse Engineering Android application

> *Dex2jar => Convert APK file into JAR*

Installation

https://sourceforge.net/projects/dex2jar/

unzip -x dex2jar

cd dex2jar

chmod +x *

sh d2j-dex2jar.sh diva-beta.apk

> *JD-GUI => To open jar file*

http://java-decompiler.github.io/

java -jar jd-gui-1.4.3.jar

. . .

Another way is to use APKTOOL

> *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. It also makes working with an app easier because of the project like file structure and automation of some repetitive tasks like building apk, etc.

https://ibotpeaches.github.io/Apktool/install/

apktool d appname.apk -o folder

d=>decompile

unzip jadx

cd jadx/bin

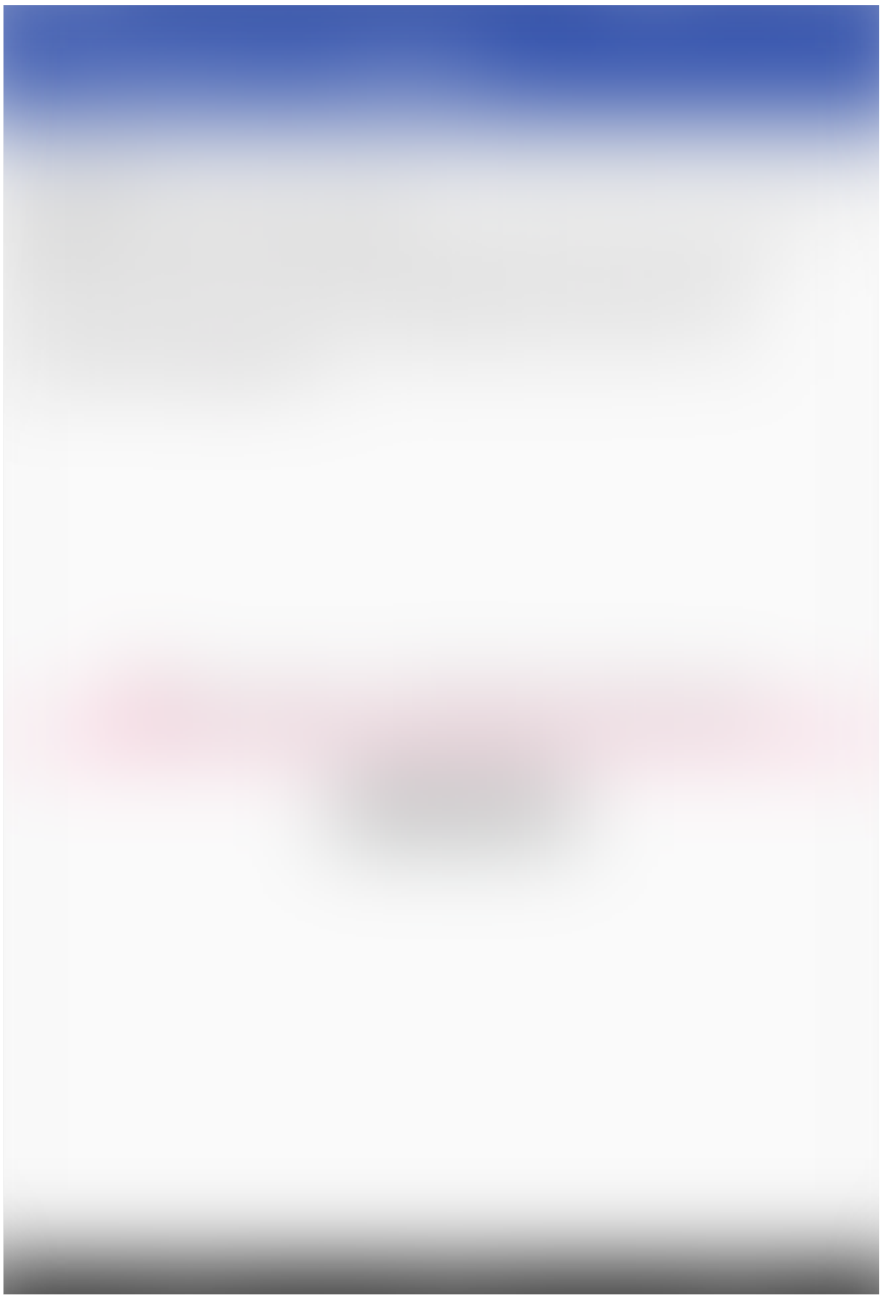
./jadx application.apk -d outputfolder

There will be so many warnings and errors but don't worry, it doesn't matter.

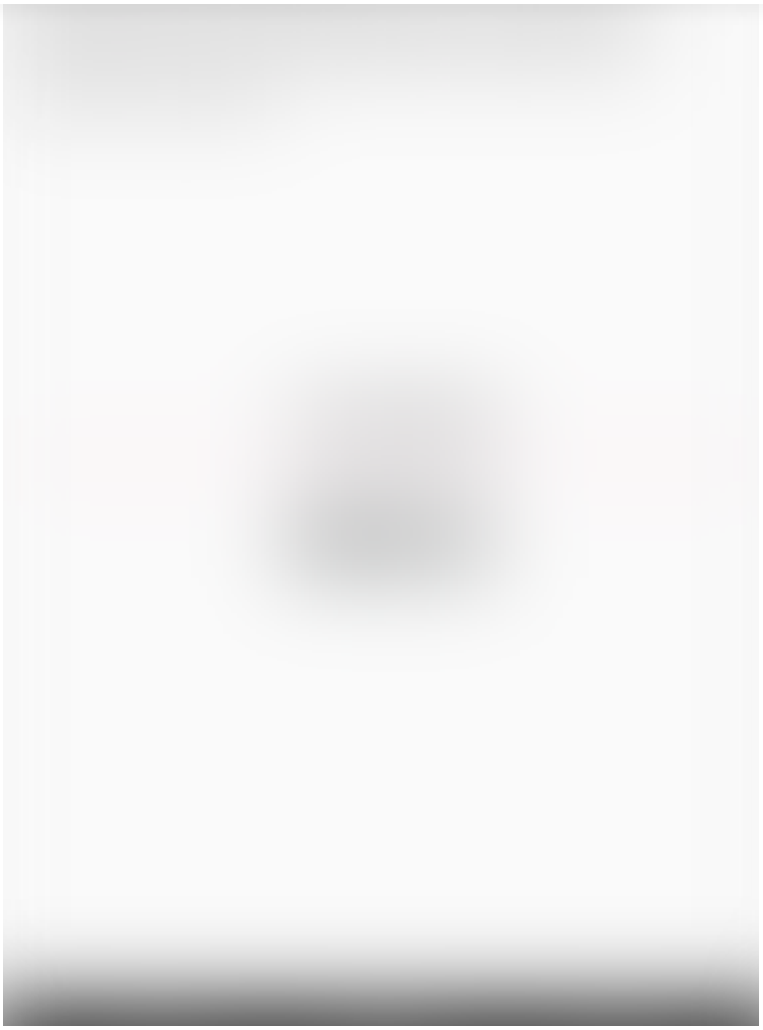
cd diva

So you must be confused, which tool to use to reverse engineer application?

Actually, it depends, but this is what I usually suggest:



Finding Process ID of Application



Now we have entered the credit card number, let's check the log if we can see it or not.

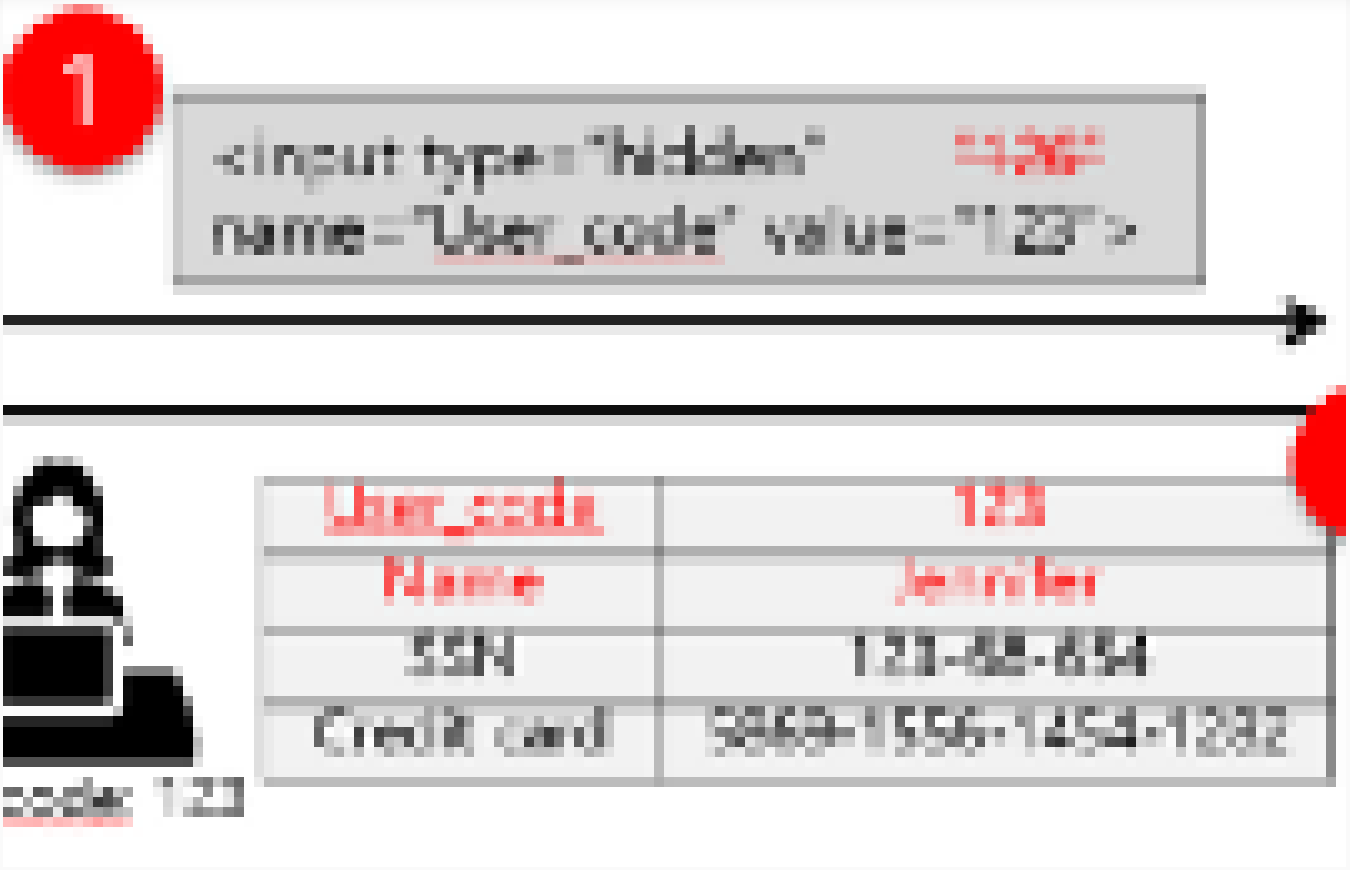
adb shell logcat | grep -i 19586

Reverse Engineering, Penetration Testing(Web, Mobile, IoT, Network, Infra)

Follow

More From Medium

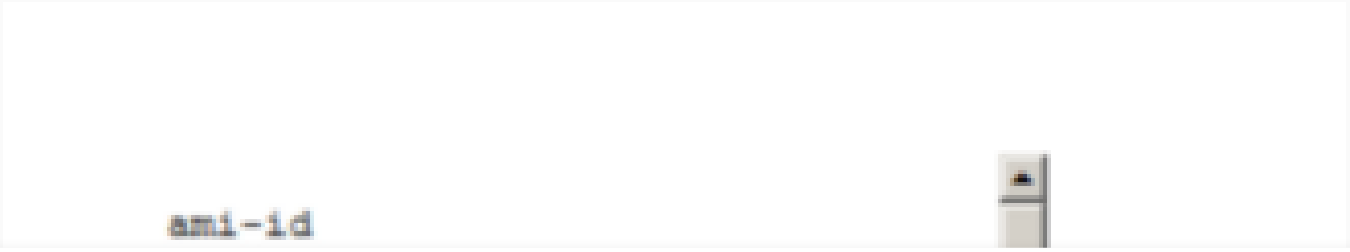
Related reads



what is Parameter Tampering

MRunal
Jun 21 · 7 min read ★

Related reads



The Bugs Are Out There, Hiding in Plain Sight

A Bug'z Life in A Bug'z Life
Jul 15 · 6 min read ★

808