Reverse Engineering Android Application

Name: Isha Gupta

TASK: To reverse engineer the given vulnerable APK and find out the solution for each lesson.

PREREQUISITES:

• iglearner.apk

• Genymotion

• Tools required are:

- adb – for connecting device to genymotion
- apktool – for decompiling application
- jadx- for decompiling application and get java codes
- burp suite – to intercept requests
- drozer – to interact with other applications through IPC

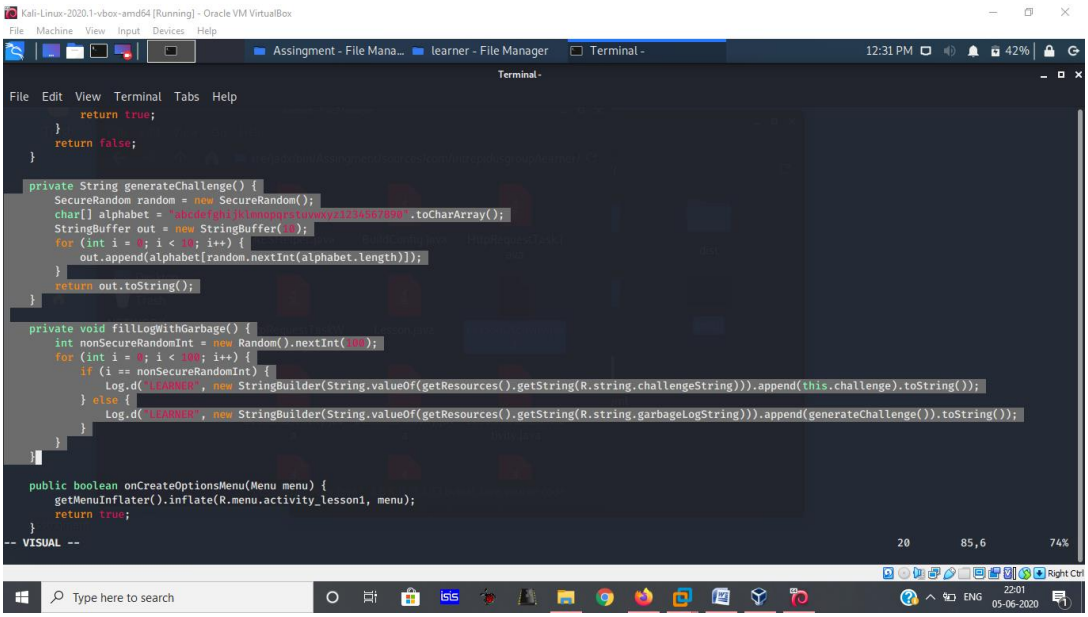
Challenges:

➢ Android logging secrets

Instructions: This lesson just dumped a whole bunch of output to the Android log. The secret code is lurking somewhere in there. Find it all and when ready, enter the code in the provided box and press the Submit button to see if you are correct.

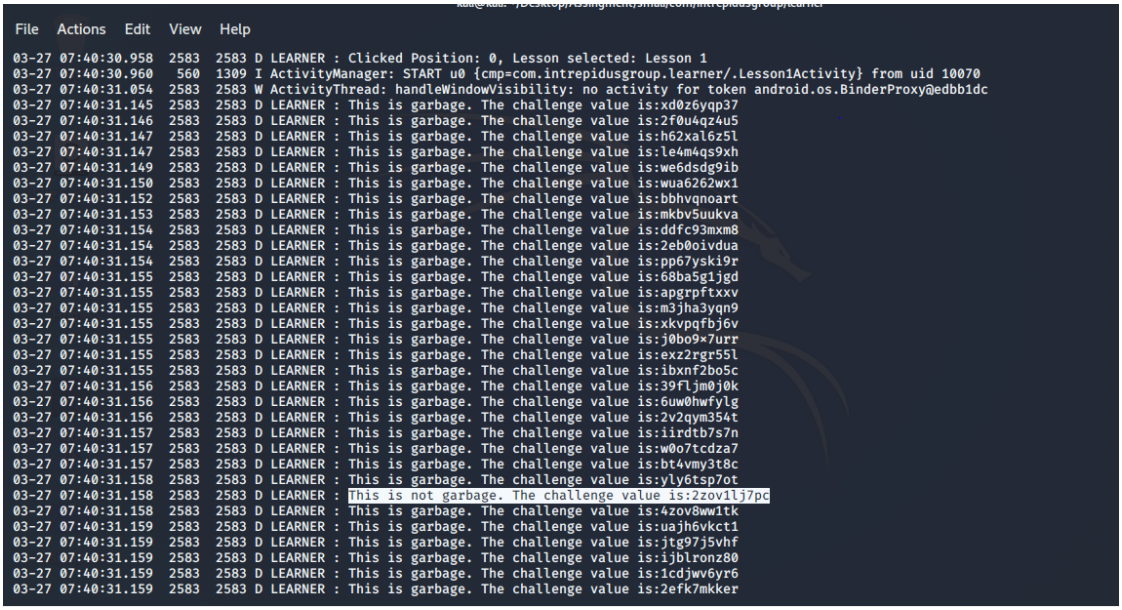
1)Decompile the apk using jadx and apktool.

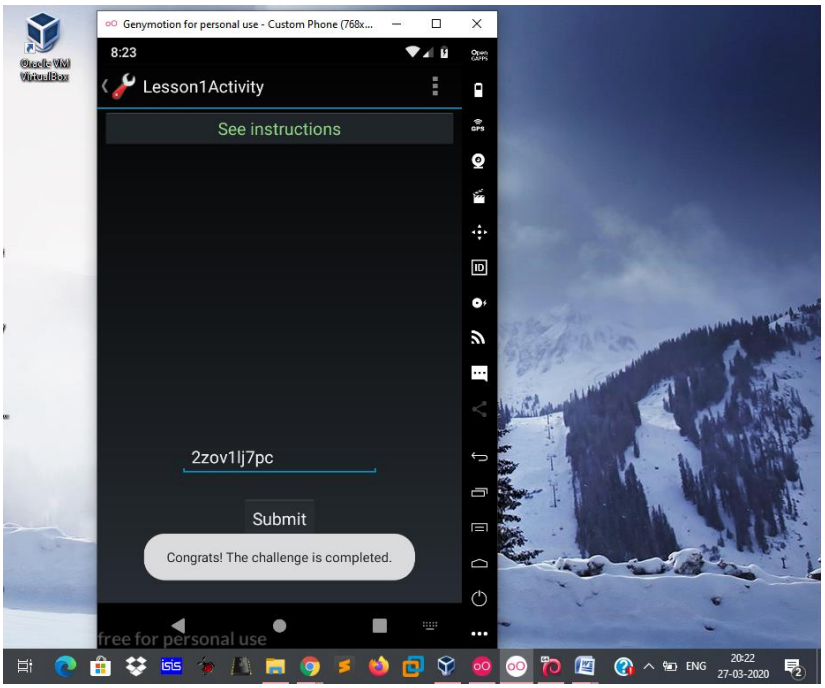
2) We see that the function is creating garbage logs.



3)adb shell

4)adb logcat filter-1 filter-2 … filter-n





➢ Screwy File Permissions

Instructions: in this lesson, we have created a world-writable file somewhere in our app's directory. You need to reverse-engineer the app, find the file that we have created, and enter the name of the file.

1) Open Lesson2Activity.java file. We see that it consist of current date and phone number with a .txt extension.

2)adb shell

3)ls –l

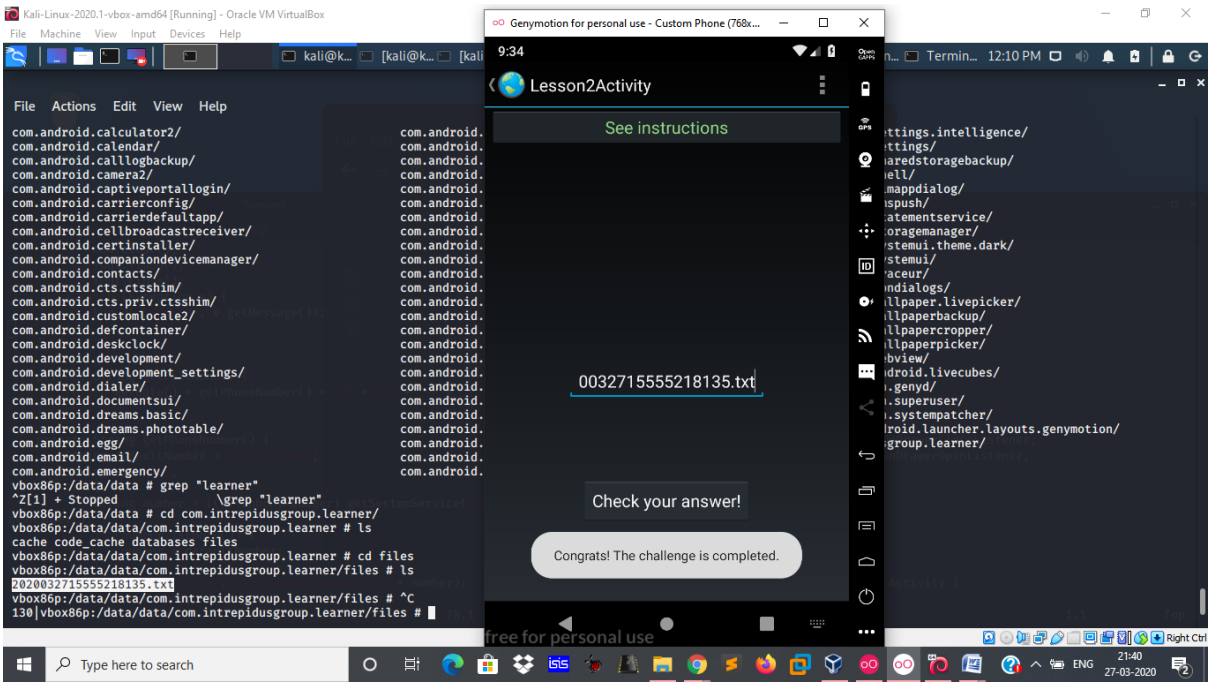
4)cd data

5)cd data

6)cd com.intrepidusgroup.learner/

7)cd files

20200327155555218135.txt



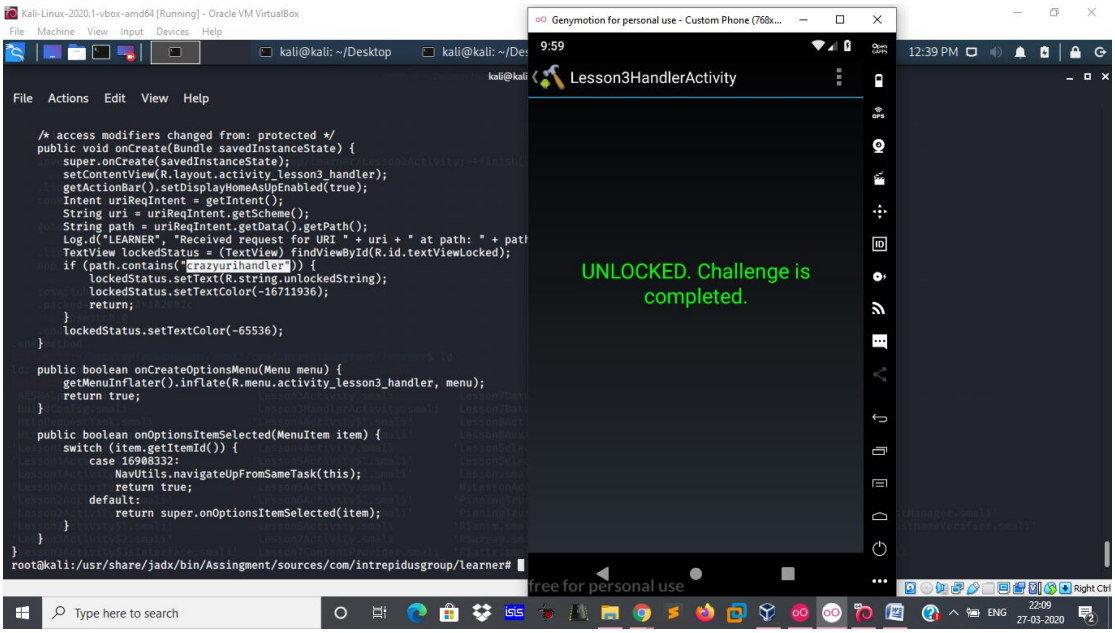
➢ URI handlers Craziness

Instructions: in the WebView that you see on the screen, we registered a handler for the URIs that start with iglearner:// . A sample URI is shown in the box. Go ahead and push the submit button to generate a URI in the WebView, and then click on the link to see what happens. The challenge will be completed if you manage to manipulate the URI in a way that would change the field below to UNLOCKED.

1)Open Lesson3Activity.java from path
/usr/share/jadx/bin/Assingment/sources/com/intrepidusgroup/learner

2)nano Lesson3HandlerActivity.java

3) In URL replace testme by crazyurihandler



➢ SSL man in the middle

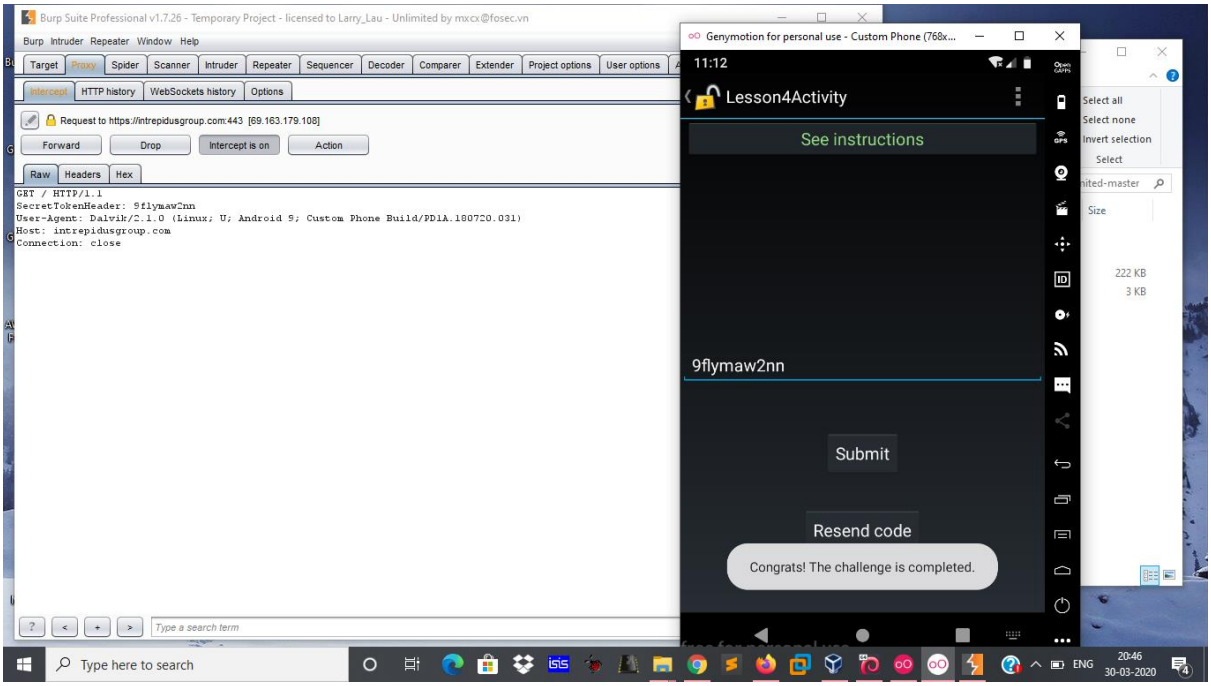
Instructions: in this lesson, you need to intercept the token that we are sending to our web server. Enter the secret token and hit submit to see if you intercepted the correct token.

1)Configure burp suite listeners configured to all interfaces by going to proxy → options

2) Install the CA certificate in device to intercept the request in burp.

3) In your mobile set the proxy to manual and enter the IP host and the port that you want burp to listen.

4)Turn on the intercept in burp and generate the request for the token from the virtual device and you'll get token header.



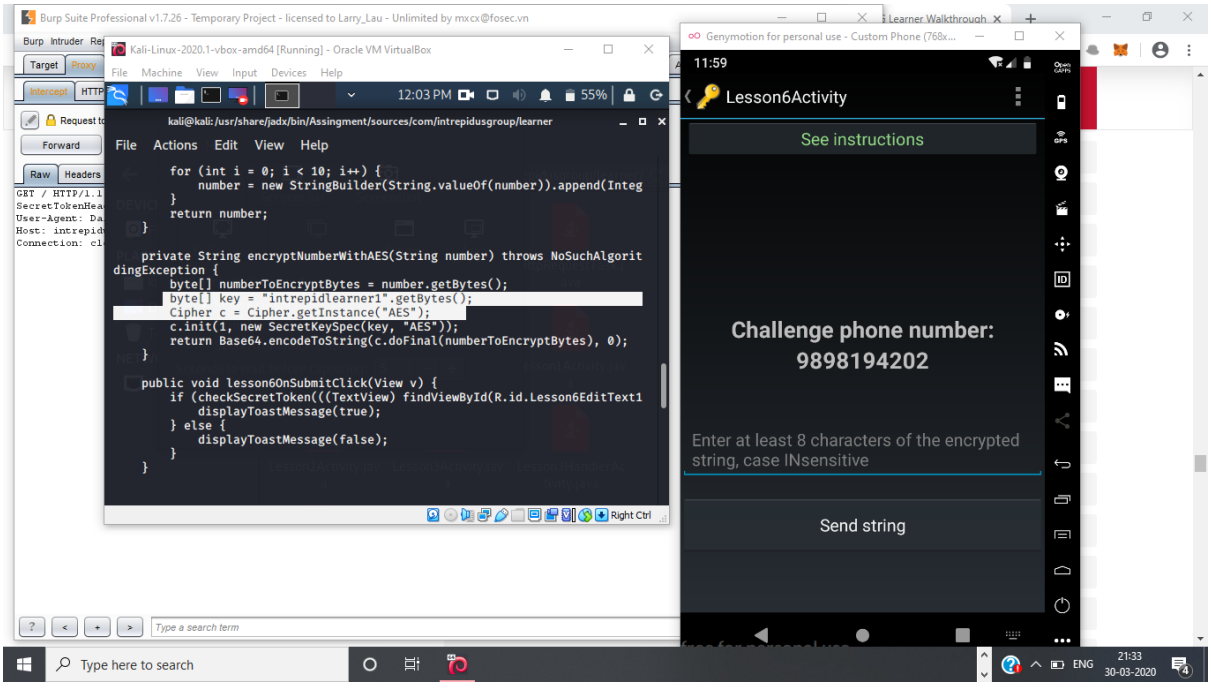
➢ Encryption vs Encraption

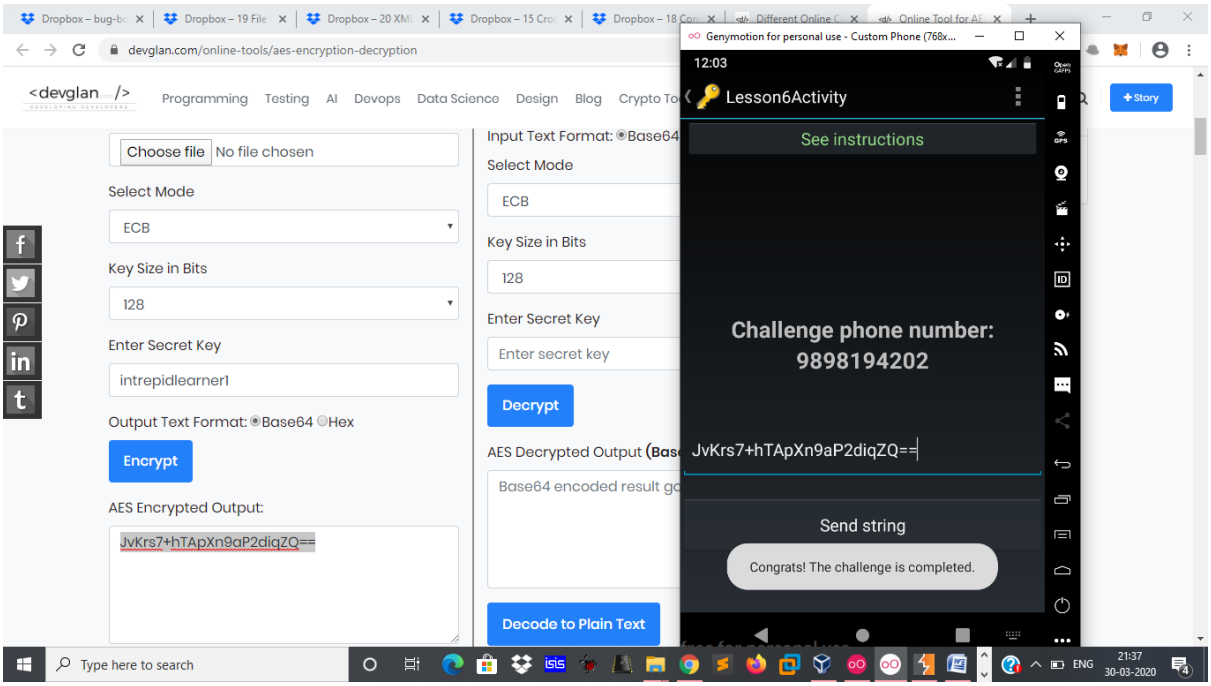
Instructions: this lesson logged out an authentication token that was created based on something stored locally, specifically, your phone number and a static key. The goal is to figure out what encryption algorithm was used and locate the key, and then enter an authentication token for another number in the provided box. Hint: if decompiling and reversing doesn't help you figure out how to create the token, there are a few shortcuts.

1) open Lesson6Activity.java. The code tells that AES encryption is being used for encryption purpose.

2) copy the number that has been generated in the app along with the hard coded key.

3) use any online AES encryption tool and encrypt that number and generate the base64 encoded encrypted text.





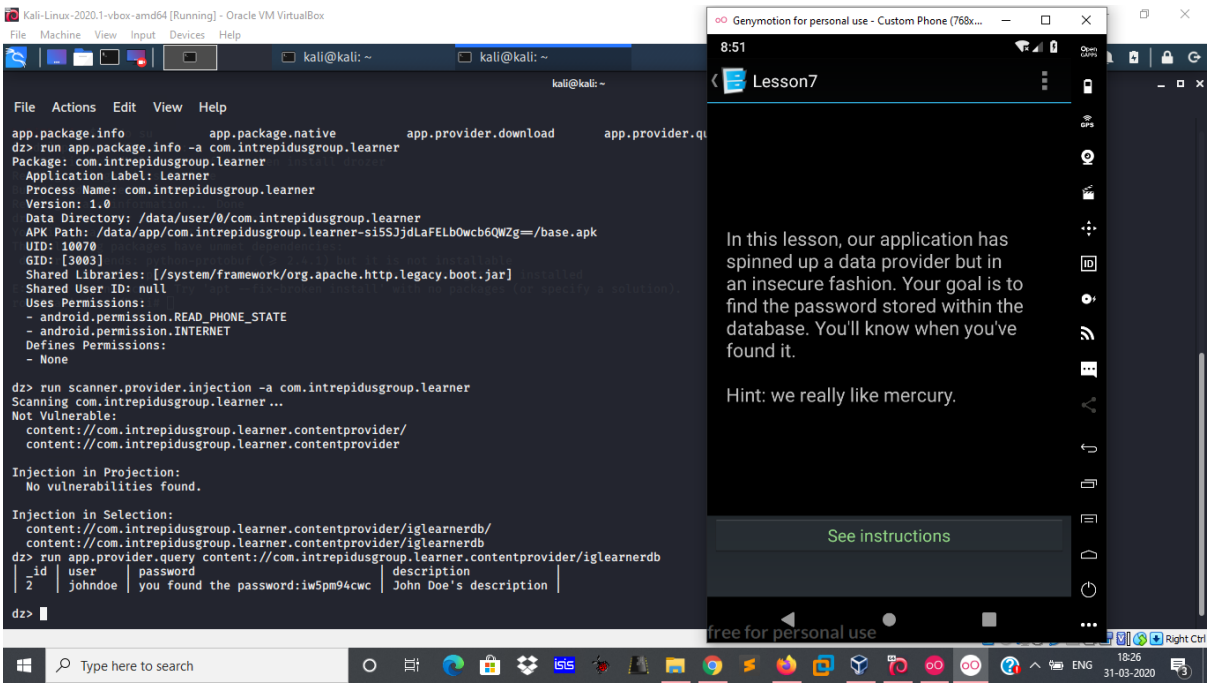
➢ Providers shared with the world

Instructions: in this lesson, our application has spinned up a data provider but in an insecure fashion. Your goal is to find the password stored within the database. You'll know when you've found it.

1)Look at Lesson7Activity.java file and Lesson7ContentProvider.java file and there we get the content provider's URI.

2)Enter adb forward tcp:31415 tcp:31415 to connect to the drozer agent then
drozer console connect which opens console from where we can see all the applications

3)Now enter the following fetch the information of the URI that we have taken from Lesson7ContentProvider.java.
run app.provider.query content://com.intrepidusgroup.learner.contentprovider/iglearnerdb
This gives the required information



➢ Malcious-Intents

Instructions: in this lesson, we'll go over the importance of securing exported application activities. The lesson activity is going to advertise an intent filter that can be invoked by another application. Interception of this intent tat is sent correctly will result in the app displaying a hidden menu inside the application. Change the code of the aux8.apk application to cause this app to display a hidden screen.

1)look at the code of Lesson8AuxActivity.java file which tells getAction() needs to be equal to something that is being referenced by "2131099692". getAction().

2)to run the activity we need the package name, activity name and the action to be performed which we get from the AndroidManifest.xml

3)Now we write our query to run the aux8.apk on our virtual device, use drozer console here to run our query to perform the action.

4)The query is:
run app.activity.start –component com.intrepidusgroup.learner com.intrepidusgroup.learner.Lesson8AuxActivity –action com.intrepidusgroup.learner.custom.intent.action.SEND –extra string 20200331 dummyData