Summary

[List of useful commands 2](#_Toc154781016)

[General 2](#_Toc154781017)

[Important methods 2](#_Toc154781018)

[Intent handling 2](#_Toc154781019)

[Activity handling 4](#_Toc154781020)

[Android Manifest handling 8](#_Toc154781021)

[Service handling 9](#_Toc154781022)

[Content Provider handling 10](#_Toc154781023)

[List of useful tools 11](#_Toc154781024)

[JADX 11](#_Toc154781025)

[Ghidra 11](#_Toc154781026)

[IDA 11](#_Toc154781027)

# List of useful commands

## General

* Make apps run via a Python script checker

python3 challenge\_checker.py victimapp\_apk\_path malapp\_apk\_path

* Create tags to be visible as errors
  + Create a variable inside the class

*private final static String TAG = "MOBIOTSEC";*

* Handle exceptions
  + Log.e(TAG, "Exception: " + e);
* Handle messages
  + Log.i(TAG, "Hello world”);

## Important methods

* + onCreate

*protected void onCreate(Bundle savedInstanceState) {*

*super.onCreate(savedInstanceState);*

*}*

* + onStart

*protected void onStart() {  
 super.onStart();*

## Intent handling

* + Get data from intents

*Intent intent = getIntent();*

*// Get the data from the received intent  
Uri data = intent.getData();*

* + Read data from intents

InputStream input = getContentResolver().openInputStream(data);

BufferedReader reader = new BufferedReader(new InputStreamReader(input));  
StringBuilder stringBuilder = new StringBuilder();

* + Build result intent and put extra

*Intent resultIntent = new Intent();  
resultIntent.putExtra("hash", hash);  
setResult(Activity.RESULT\_OK, resultIntent);*

* Handling of explicit intents

*@Override*

*protected void onStart() {*

*super.onStart();*

*Intent intent1 = new Intent();*

*intent1.setComponent(new ComponentName("com.example.victimapp", "com.example.victimapp.PartOne"));*

*if (intent1.resolveActivity(getPackageManager()) != null) {*

*startActivityForResult(intent1, 1);*

*}*

*}*

*@Override*

*protected void onActivityResult(int requestCode, int resultCode, Intent data) {*

*super.onActivityResult(requestCode, resultCode, data);*

*…..*

*if(requestCode == 1){*

*Intent intent2 = new Intent("com.example.victimapp.intent.action.JUSTASK");*

*if (intent2.resolveActivity(getPackageManager()) != null) {*

*startActivityForResult(intent2, 2);*

*}*

* Handle explicit internet inside MainActivity

*public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 Context context = getApplicationContext();  
 Intent explicitIntent = new Intent(context, LocationService.class);  
 context.startService(explicitIntent);  
 }*

## Activity handling

* Class behaving as an Activity

*public class HashMe extends Activity*

* Launch activities (Java – not deprecated)

The old way:

*public void openSomeActivityForResult() {*

*Intent intent = new Intent(this, SomeActivity.class);*

*startActivityForResult(intent, 123);*

*}*

*@Override*

*protected void onActivityResult (int requestCode, int resultCode, Intent data) {*

*if (resultCode == Activity.RESULT\_OK && requestCode == 123) {*

*doSomeOperations();*

*}*

*}*

The new way (Java):

*public void openSomeActivityForResult() {*

*Intent intent = new Intent(this, SomeActivity.class);*

*someActivityResultLauncher.launch(intent);*

*}*

*// You can do the assignment inside onAttach or onCreate, i.e, before the activity is displayed*

*ActivityResultLauncher<Intent> someActivityResultLauncher = registerForActivityResult(*

*new ActivityResultContracts.StartActivityForResult(),*

*new ActivityResultCallback<ActivityResult>() {*

*@Override*

*public void onActivityResult(ActivityResult result) {*

*if (result.getResultCode() == Activity.RESULT\_OK) {*

*// There are no request codes*

*Intent data = result.getData();*

*doSomeOperations();*

*}*

*}*

*});*

* An example of this in action (handling multiple activities)

package com.example.myapplication;

import androidx.activity.result.ActivityResultLauncher;

import androidx.activity.result.contract.ActivityResultContracts;

public class MainActivity extends AppCompatActivity {

private static final String TAG = "MOBIOTSEC";

private static final int FIRST\_ACTIVITY\_REQUEST\_CODE = 1;

private static final int SECOND\_ACTIVITY\_REQUEST\_CODE = 2;

private static final int THIRD\_ACTIVITY\_REQUEST\_CODE = 3;

private static final int FOURTH\_ACTIVITY\_REQUEST\_CODE = 4;

private final ActivityResultLauncher<Intent> startForResult = registerForActivityResult(

new ActivityResultContracts.StartActivityForResult(),

result -> handleActivityResult(result.getResultCode(), result.getData())

);

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

}

@Override

protected void onStart() {

super.onStart();

startPartOneActivity();

}

private void startPartOneActivity() {

Intent intent = new Intent();

intent.setComponent(new ComponentName("com.example.victimapp", "com.example.victimapp.PartOne"));

if (intent.resolveActivity(getPackageManager()) != null) {

startForResult.launch(intent);

}

}

private void startNextActivity(String actionOrClassName, int requestCode) {

Intent intent = new Intent(actionOrClassName);

if (intent.resolveActivity(getPackageManager()) != null) {

startForResult.launch(intent);

}

}

private void handleActivityResult(int resultCode, @Nullable Intent data){

Log.i(TAG, "resultCode: " + resultCode);

Log.i(TAG, "IntentData: " + data);

if (resultCode == FIRST\_ACTIVITY\_REQUEST\_CODE) {

startNextActivity("com.example.victimapp.intent.action.JUSTASK", SECOND\_ACTIVITY\_REQUEST\_CODE);

} else if (resultCode == SECOND\_ACTIVITY\_REQUEST\_CODE) {

startNextActivity("com.example.victimapp.PartThree", THIRD\_ACTIVITY\_REQUEST\_CODE);

} else if (resultCode == THIRD\_ACTIVITY\_REQUEST\_CODE) {

startNextActivity("com.example.victimapp.intent.action.JUSTASKBUTNOTSOSIMPLE", FOURTH\_ACTIVITY\_REQUEST\_CODE);

} …..

}

* Right way to start activities in Kotlin

val launchActivity =

registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result: ActivityResult ->

if (result.resultCode == Activity.RESULT\_OK) {

// There are no request codes

val data: Intent? = result.data

// Handle the result here

}

}

val intent = Intent()

intent.component = ComponentName("com.example.victimapp", "com.example.victimapp.SerialActivity")

launchActivity.launch(intent)

Log.i("MOBIOTSEC", "Launched intent!")

}

}

* Get Data from Bundles

Bundle bundle = data.getExtras();

for (String p : bundle.keySet()) {

Log.i(TAG, "key: " + p + " value: " + bundle.get(p));

}

* Exploit Java Reflection
  + Means you leverage vulnerabilities of a called class making a method accessible

FlagContainer fc = (FlagContainer) bundle.get("flag");

Method method = fc.getClass().getDeclaredMethod("getFlag");

method.setAccessible(true);/

Log.i(TAG, "Result: " + method.invoke(fc).toString());

## Android Manifest handling

Everything has to be done inside the “<application>” tag.

* Declare intent filters
  + Done inside an activity

*<activity android:name=".MainActivity">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>*

* Declare multiple activities (they can have intent filters too)

*<activity android:name=".MainActivity">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
        <activity android:name=".PartOne" android:exported="true"/>  
        <activity android:name=".PartTwo">  
            <intent-filter>  
                <action android:name="com.example.victimapp.intent.action.JUSTASK"/>  
                <category android:name="android.intent.category.DEFAULT"/>  
            </intent-filter>  
        </activity>  
        <activity android:name=".PartThree" android:exported="true"/>  
        <activity android:name=".PartFour">  
            <intent-filter>  
                <action android:name="**com.example.victimapp.intent.action.JUSTASKBUTNOTSOSIMPLE"/>  
                <category android:name="android.intent.category.DEFAULT"/>  
            </intent-filter>  
        </activity>*

## Service handling

* Declare it in the Manifest file (intent filter)

<service

android:name=".LocationService"

android:exported="true">

<intent-filter>

<action android:name="com.mobiotsec.intent.action.GIMMELOCATION" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</service>

* Make the class extend Service and launch service inside onCreate

public class LocationService extends Service {  
 private static final String TAG = "MOBIOTSEC";  
 private FusedLocationProviderClient fusedLocationClient;  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);  
 startForegroundServiceNotification(); // Start foreground service with a notification  
 }  
  
 @Nullable  
 @Override  
 public IBinder onBind(Intent intent) {  
 return null;  
 }

* Service returns reply to the system

Intent i = new Intent();  
i.setAction("com.mobiotsec.intent.action.LOCATION\_ANNOUNCEMENT");  
i.putExtra("location", currLoc);  
sendBroadcast(i);

## Content Provider handling

* Query package, all the data and get data in Cursor format

class MainActivity : ComponentActivity() {  
override fun onCreate(savedInstanceState: Bundle?) {  
super.onCreate(savedInstanceState)  
  
val contenturi = Uri.parse("content://com.example.victimapp.MyProvider/joke")  
  
val projection = arrayOf("author", "joke")  
  
val selection = "author = ?"  
val selectionArgs = arrayOf("elosiouk")  
  
val cursor = contentResolver.query(contenturi, projection, selection, selectionArgs, null)

# List of useful tools

## JADX

* Either open jadx-gui over a specific apk
* Or use the cli line with syntax
  + jadx -d outputfoldername app.apk
  + E.g. jadx -d out babyrev.apk
* Usually, the folder to look is inside
  + out > sources > com > mobiotsec > appname
* Other useful files to look into
  + *strings.xml*
    - app.apk->Resources->resources.arsc->res->values->strings.xml
* Other useful things to note
  + Look for methods like checkFlag/pinCheck
  + Many times, you will be asked to write some kind of decrypting logic yourself

## Ghidra

* Open Ghidra and click on File -> New Project, choose Non shared project , pick a directory and the name for your project then click on finish. Also on File options pick Import File and then choose the PE file you want to reverse engineering
* Have fun looking at its files and its overall code in pseudocode format natively

## IDA

* Open IDA and just open a whatever desired file
* Have fun looking at the code (ps. press F5 to look into pseudocode)(