

## REVERSING DE SISTEMAS OPERATIVOS MOVILES

```

error OxN      : Display error           rax (64 bits)
address       : Display information about memory    eax (32 bits)
-             : List threads          ax (16 bits)
bl            : List breakpoints        ah (8 bits)
bc            : Cancel breakpoints      al (8 bits)
be            : Enable breakpoints
bd            : Disable breakpoints
bp [Addr]     : Set breakpoint at the address
bm SymPattern : Set breakpoint at the symbol
ba [t!wie] Addr : Set breakpoint on Access
k              : Display call stack
r              : Dump all registers
u              : Disassemble
dn             : Display Where N:
a: ascii chars | u: Unicode char
b: byte + ascii | w: word
M: word + ascii | d: dword
c: dword + ascii | q: qword
b: bin + byte | d: bin + dword
eN Addr Value : Edit memory
.writemem f A S : Dump memory
f: file name
A: Address
S: Size (Lx)
dec hex char dec hex char dec hex char dec hex char
0 0x00 NUL 32 0x20 SP@E 44 0x40 ? 96 0x60
1 0x01 SOH 33 0x21 ! 65 0x41 A 97 0x61 s
2 0x02 STX 34 0x22 * 66 0x42 B 98 0x62 b
3 0x03 ETX 35 0x23 # 67 0x43 C 99 0x63 o
4 0x04 EOT 36 0x24 $ 68 0x44 D 100 0x64 d
5 0x05 ENQ 37 0x25 % 69 0x45 E 101 0x65 e
6 0x06 ACK 38 0x26 & 70 0x46 F 102 0x66 f
7 0x07 BEL 39 0x27 . 71 0x47 G 103 0x67 g
8 0x08 BS 40 0x28 ( 72 0x48 H 104 0x68 h
9 0x09 TAB 41 0x29 ) 73 0x49 I 105 0x69 i
10 0x0A LF 42 0x2A * 74 0x4A J 106 0x6A j
11 0x0B VT 43 0x2B + 75 0x5B K 107 0x7B k
12 0x0C FF 44 0x2C , 76 0x5C L 108 0x7C l
13 0x0D CR 45 0x2D - 77 0x5D M 109 0x7D m
14 0x0E SO 46 0x2E . 78 0x5E N 110 0x7E n
15 0x0F SI 47 0x2F / 79 0x5F O 111 0x7F o
16 0x10 DLE 48 0x30 0 80 0x50 P 112 0x70 p
17 0x11 DC1 49 0x31 1 81 0x51 Q 113 0x71 q
18 0x12 DC2 50 0x32 2 82 0x52 R 114 0x72 r
19 0x13 DC3 51 0x33 3 83 0x53 S 115 0x73 s
20 0x14 DC4 52 0x34 4 84 0x54 T 116 0x74 t
21 0x15 NAK 53 0x35 5 85 0x55 U 117 0x75 u
22 0x16 SYN 54 0x36 6 86 0x56 V 118 0x76 v
23 0x17 ETB 55 0x37 7 87 0x57 W 119 0x77 w
24 0x18 CAN 56 0x38 8 88 0x58 X 120 0x78 x
25 0x19 EN 57 0x39 9 89 0x59 Y 121 0x79 y
26 0x1A SUB 58 0x3A : 90 0x5A Z 122 0x7A z
27 0x1B ESC 59 0x3B ; 91 0x5B [ 123 0x7B [
28 0x1C FS 60 0x3C < 92 0x5C \ 124 0x7C \
29 0x1D GS 61 0x3D = 93 0x5D I 125 0x7D ]
30 0x1E RS 62 0x3E > 94 0x5E ^ 126 0x7E ^
31 0x1F US 63 0x3F ? 95 0x5F DEL 127 0x7F DEL

```

## Máster en Análisis de Malware,

### Reversing y Bug Hunting



**UCAM**  
UNIVERSIDAD  
CATÓLICA DE MURCIA



**ENIIT**  
INNOVA IT BUSINESS SCHOOL

*Ramón Gonzalez Gaztelupe*

## 1.1 - Registro Inseguro

La primera vulnerabilidad obtiene a través de un EditText una entrada de datos la cual se almacena en **cctxr** y por error se logea después parseado a string a través de **Log.e()**.

En el código básicamente se obtiene lo introducido en el textbox creando un objeto de la clase EditText, para después llamar al método de la clase **LogActivity procesCC()** y pasárselo como parámetro lo obtenido haciéndole un cast a string, ya que el método **procesCC()** solo admite como valor un tipo de dato String.

El método **procesCC()** envia una excepción, por lo tanto **checkout()** siempre entrara en **catch()** y mostrara lo almacenado en **cctxt** a través de **getText()** parseado a string:

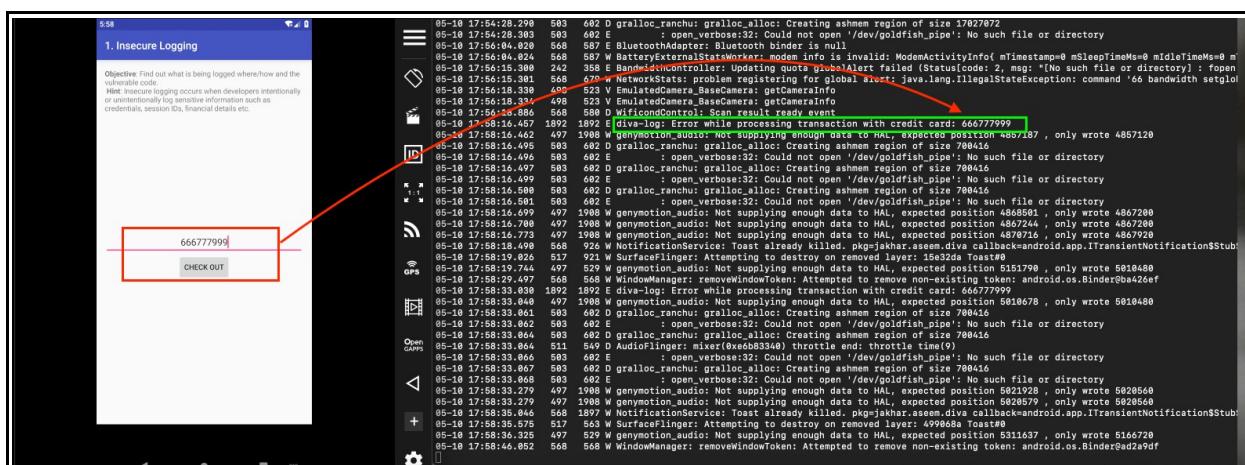
```
/* loaded from: classes.dex */
public class LogActivity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.su
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_log);
    }

    public void checkout(View view) {
        EditText cctxt = (EditText) findViewById(R.id.ccText);
        try {
            processCC(cctxt.getText().toString());
        } catch (RuntimeException e) {
            Log.e("diva-log", "Error while processing transaction with credit card: " + cctxt.getText().toStri
        }
    }

    private void processCC(String ccstr) {
        RuntimeException e = new RuntimeException();
        throw e;
    }
}
```

Para debug y ver lo que devuelven los Log, que son una forma de detectar los errores en el desarrollo, tenemos que obtener los logs a través de adb: **adb logcat**

Introducimos un numero como “**666777999**” en la aplicación y traceamos dinámicamente



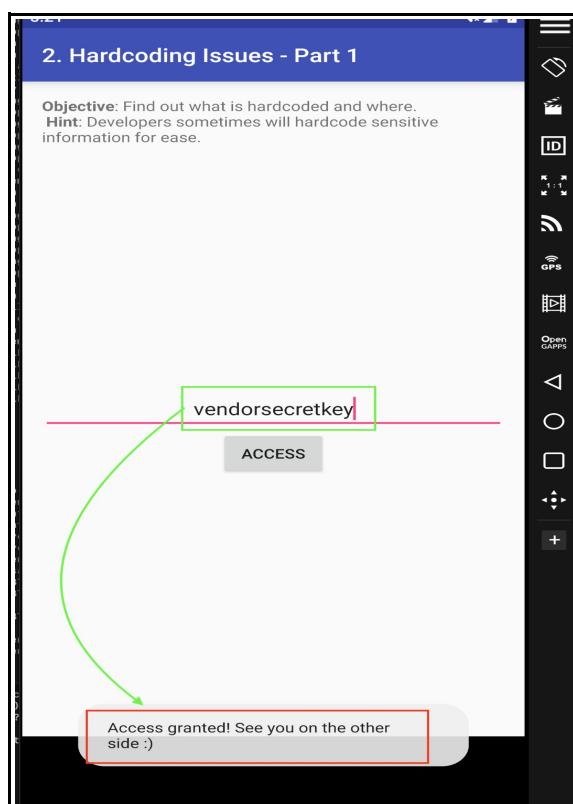
## 1.2 - Datos sensibles hardcodeados

En la clase **HardcodeActivity** se realiza un **equals()** a un string harcodeado en el código **“vendorsecretkey”**:

```
/* loaded from: classes.dex */
public class HardcodeActivity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseF
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode);
    }

    public void access(View view) {
        EditText hcKey = (EditText) findViewById(R.id.hcKey);
        if (hcKey.getText().toString().equals("vendorsecretkey")) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}
```

Si introducimos ese texto en el `textbox` obtendremos un Acces Granted:



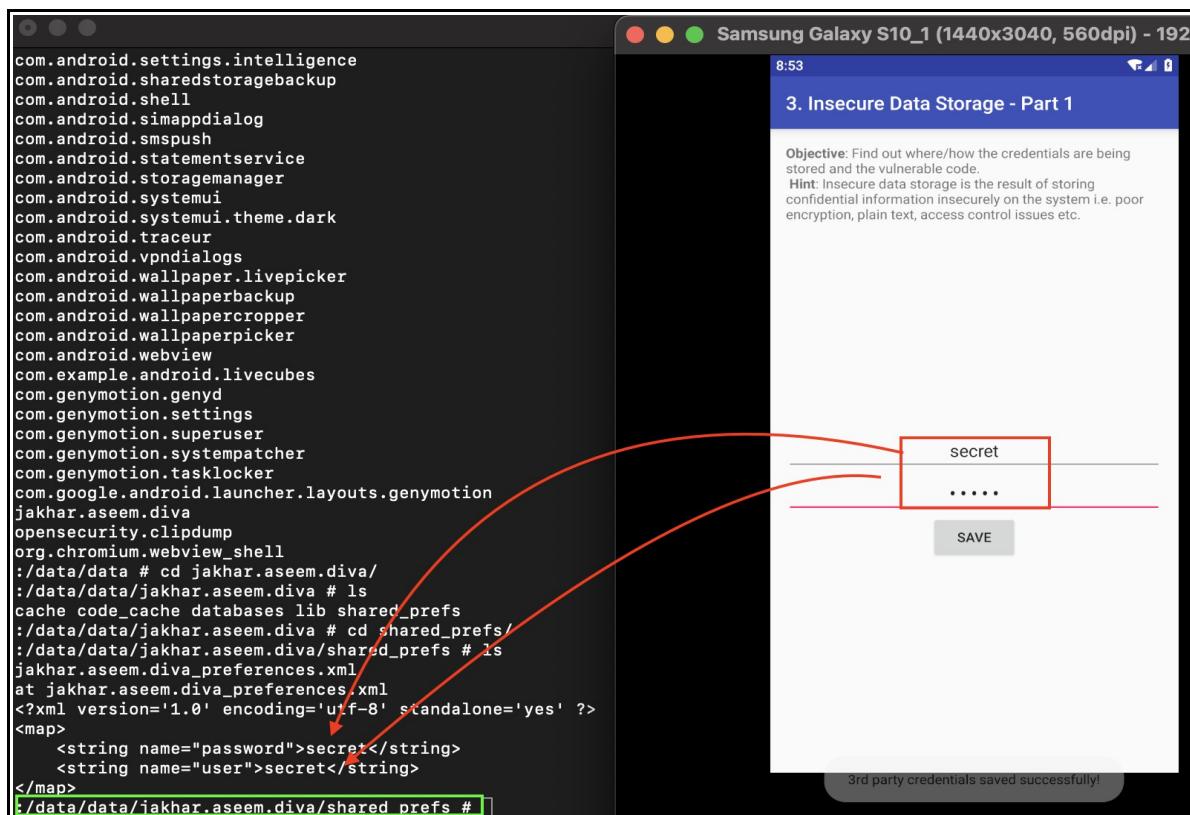
## 1.3 - Almacenamiento de datos inseguro

Android almacena la configuración de preferencias compartidas en un archivo .xml en la carpeta **shared\_prefs** → **data/data/[APP]**.

En el código podemos ver en el método **saveCredentials()** como se obtiene la ruta al archivo y se edita modificando los datos introducidos en cada *textbox* de la APP con los del .xml

```
19     public void saveCredentials(View view) {
20         SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);
21         SharedPreferences.Editor spedit = spref.edit();
22         EditText usr = (EditText) findViewById(R.id.ids1Usr);
23         EditText pwd = (EditText) findViewById(R.id.ids1Pwd);
24         spedit.putString("user", usr.getText().toString());
25         spedit.putString("password", pwd.getText().toString());
26         spedit.commit();
27         Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
28     }
29 }
```

Por lo tanto las credenciales que guardemos a través de ese View se almacenaran en texto plano en  
**data/data/jakhar.aseem.diva/shared\_prefs/jakhar.aseem.diva\_preferences.xml**



## 1.4 - Almacenamiento de datos inseguro 2

En el código se observa como se almacenan los campos **ids2Usr** y **ids2Pwd** en la tabla **myuser**, el primero en el campo user y el segundo en el campo password, ambos en texto plano.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    try {
        this.mDB = openOrCreateDatabase("ids2", 0, null);
        this.mDB.execSQL("CREATE TABLE IF NOT EXISTS myuser(user VARCHAR, password VARCHAR);");
    } catch (Exception e) {
        Log.d("Diva", "Error occurred while creating database: " + e.getMessage());
    }
    setContentView(R.layout.activity_insecure_data_storage2);
}
```

```
public void saveCredentials(View view) {
    EditText usr = (EditText) findViewById(R.id.ids2Usr);
    EditText pwd = (EditText) findViewById(R.id.ids2Pwd);
    try {
        this.mDB.execSQL("INSERT INTO myuser VALUES ('" + usr.getText().toString() + "', '" + pwd.getText().toString() + "');");
        this.mDB.close();
    } catch (Exception e) {
        Log.d("Diva", "Error occurred while inserting into database: " + e.getMessage());
    }
    Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
}
```

Introduciremos un user y password nuevos a través de la APP:  
**user=rajkit, password=rajkit**

Para acceder a la base de datos SQLite tendremos que acceder a través de adb con una shell, para ello → adb shell

Una vez dentro vamos hasta el directorio **/data/data/[nombre app]/databases** y nos mostrara las BD que tiene la aplicación y comprobamos si existe la DB que nos mostraba el decompilado. “**ids2**”:

```
[1]:/data/data/jakhar.aseem.diva/databases # ls
divanotes.db divanotes.db-shm divanotes.db-wal ids2 ids2-shm ids2-wal
:/data/data/jakhar.aseem.diva/databases #
```

Accederemos a ella a través de sqlite3 desde la misma shell y realizaremos una consulta para que nos vuelque todos los usuarios :

```

:/data/data/jakhar.aseem.diva/databases # sqlite3 ids2
SQLite version 3.22.0 2019-09-03 18:36:11
Enter ".help" for usage hints.
sqlite> . tables;
Error: unknown command or invalid arguments: "tables;". Enter ".help" for help
sqlite> . tables
android_metadata myuser
sqlite> select * from myuser;
admin|admin
rajkit|rajkit
rajkit OR 1=1|123456
sqlite> 

```

## 1.5 - Almacenamiento de datos inseguro 3

Podemos apreciar en el código que el método `saveCredentials()` en este caso obtiene los strings introducidos en ambos EditText y los almacena en texto plano en el directorio propio en un archivo que crea llamado **"uinfo+identificador+tmp"**

```

public void saveCredentials(View view) {
    EditText usr = (EditText) findViewById(R.id.ids3Usr);
    EditText pwd = (EditText) findViewById(R.id.ids3Pwd);
    File ddir = new File(getApplicationContext().dataDir);
    try {
        File uinfo = File.createTempFile("uinfo", "tmp", ddir);
        uinfo.setReadable(true);
        uinfo.setWritable(true);
        FileWriter fw = new FileWriter(uinfo);
        fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
        fw.close();
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    } catch (Exception e) {
        Toast.makeText(this, "File error occurred", 0).show();
        Log.d("Diva", "File error: " + e.getMessage());
    }
}

```

Enviamos los datos a través de la app y comprobamos si podemos obtener nuestras credenciales:

```

:/data/data/jakhar.aseem.diva # cat uinfo4478281446761325151tmp
rajkit:rajkit
:/data/data/jakhar.aseem.diva # ls
cache code_cache databases lib shared_prefs uinfo4478281446761325151tmp
:/data/data/jakhar.aseem.diva # cat uinfo4478281446761325151tmp
rajkit:rajkit
:/data/data/jakhar.aseem.diva #

```

## 1.6 - Almacenamiento de datos inseguro 4

En este view podemos ver en el código como se obtiene el *path* del almacenamiento externo y se crea un archivo oculto con las credenciales en texto plano:

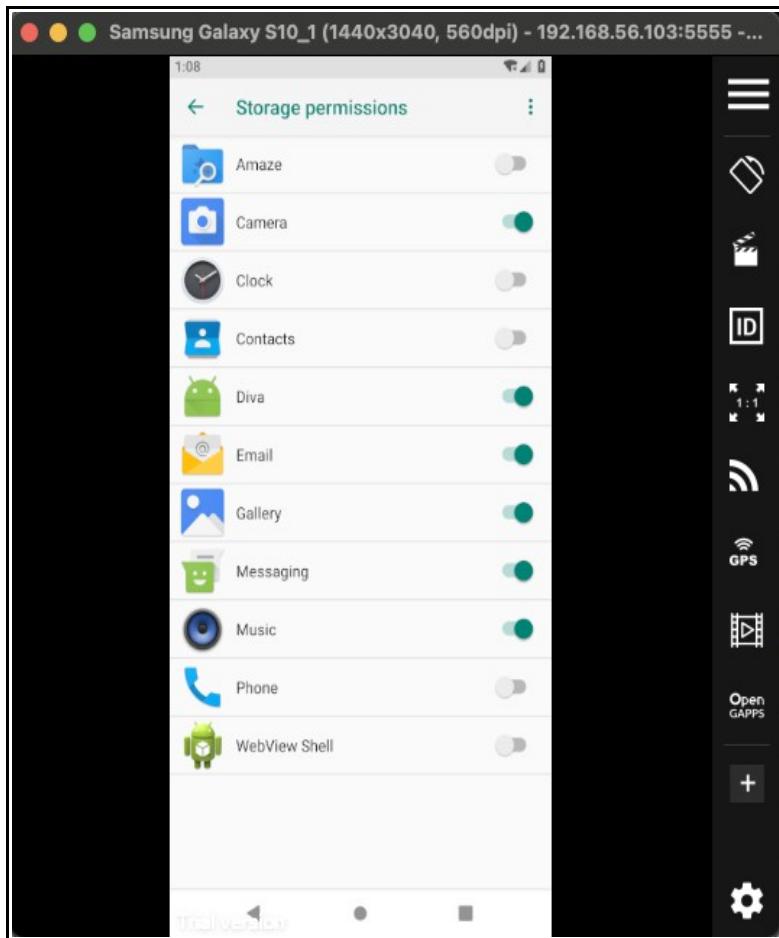
```

/*
 * LOKED FROM: classes.dex
 */
public class InsecureDataStorage4Activity extends AppCompatActivity {
    /* JADY INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, and
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_insecure_data_storage4);
    }

    public void saveCredentials(View view) {
        EditText usr = (EditText) findViewById(R.id.ids4usr);
        EditText pwd = (EditText) findViewById(R.id.ids4pwd);
        File sdDir = Environment.getExternalStorageDirectory();
        try {
            File uinfo = new File(sdDir.getAbsolutePath() + "/.uinfo.txt");
            uinfo.setReadable(true);
            uinfo.setWritable(true);
            FileWriter fw = new FileWriter(uinfo);
            fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
            fw.close();
            Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
        } catch (Exception e) {
            Toast.makeText(this, "File error occurred", 0).show();
            Log.d("Diva", "File error: " + e.getMessage());
        }
    }
}

```

Para poder crear un archivo con los permisos de **lectura/escritura** fuera de los límites de la APP tenemos que darle primero los permisos desde el propio emulador.



Para ver nuestras credenciales accedemos desde adb shell al directorio de la *sdcard*, y mostramos el directorio completo con ls -la:

```
./data/data/jakhar.aseem.diva # cd /mnt/
./mnt # ls
appfuse asec expand media_rw obb runtime sdcard secure shared user vendor
./mnt # cd s
sdcard/ secure/ shared/
./mnt/sdcard # ls
Alarms DCIM Download Movies Music Notifications Pictures Podcasts Ringtones
./mnt/sdcard # ls -la
total 92
drwxrwx--x 11 root sdcard_rw 4096 2023-05-11 13:02 .
drwxr-x---x 4 root sdcard_rw 4096 2023-05-09 15:03 ..
-rw-rw---- 1 root sdcard_rw 14 2023-05-11 13:02 .uininfo.txt
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Alarms
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 DCIM
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Download
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Movies
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Music
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Notifications
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Pictures
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Podcasts
drwxrwx--x 2 root sdcard_rw 4096 2023-05-09 15:03 Ringtones
./mnt/sdcard # cat .uininfo.txt
rajk1:rajk1yt
./mnt/sdcard #
```

## 1.7 - Input validation issues 1

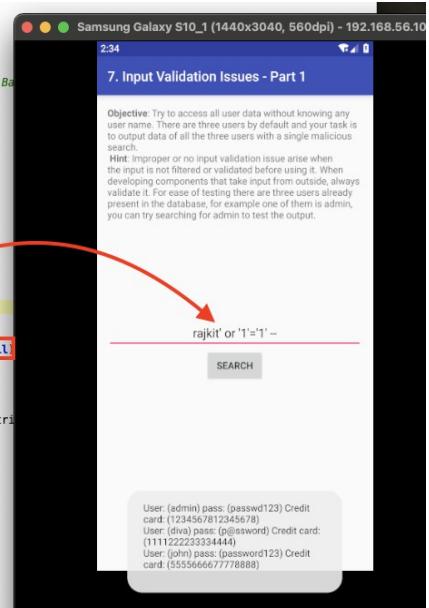
En este view tenemos un error en la integración de los datos recibidos desde el **EditText** con la creación de la consulta SQLite, básicamente no se controla los tipos de caracteres que se obtienen del **TextView** y los integra directamente en la consulta.

La consulta a la BD básicamente vuelca todos los datos del **USER** siempre y cuando coincida con alguno de la tabla, sin embargo podemos continuar esa consulta introduciendo cualquier usuario, cerramos con una comilla y extendemos con un **OR** seguido de **1=1** que equivale a a que la condición se cumpla.

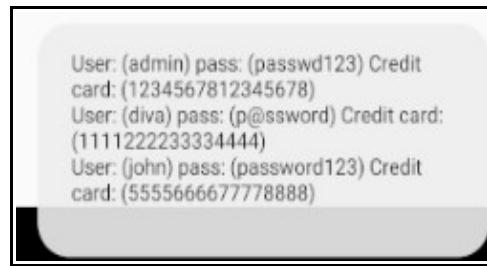
```
/* loaded from: classes.dex */
public class SQLinjectionActivity extends AppCompatActivity {
    private SQLiteDatabase mDB;

    /* JADY INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.Back
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            this.mDB = openOrCreateDatabase("sql", 0, null);
            this.mDB.execSQL("DROP TABLE IF EXISTS sqiluser;");
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqiluser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");
            this.mDB.execSQL("INSERT INTO sqiluser VALUES ('admin', 'password123', '1234567812345678');");
            this.mDB.execSQL("INSERT INTO sqiluser VALUES ('diva', '@$sword', '111122223334444');");
            this.mDB.execSQL("INSERT INTO sqiluser VALUES ('john', 'password123', '555566667778888');");
        } catch (Exception e) {
            Log.d("Diva-sqli", "Error occurred while creating database for SQLI: " + e.getMessage());
        }
        setContentView(R.layout.activity_sqlinjection);
    }

    public void search(View view) {
        EditText srchtxt = (EditText) findViewById(R.id.ivilesearch);
        try {
            Cursor cr = this.mDB.rawQuery("SELECT * FROM sqiluser WHERE user = '" + srchtxt.getText().toString() + "'", null);
            StringBuilder strb = new StringBuilder("");
            if (cr != null && cr.getCount() > 0) {
                cr.moveToFirst();
                do {
                    strb.append("User: (" + cr.getString(0) + ") pass: (" + cr.getString(1) + ") Credit card: (" + cr.getStri
                } while (cr.moveToNext());
            } else {
                strb.append("User: (" + srchtxt.getText().toString() + ") not found");
            }
            Toast.makeText(this, strb.toString(), 0).show();
        } catch (Exception e) {
            Log.d("Diva-sqli", "Error occurred while searching in database: " + e.getMessage());
        }
    }
}
```

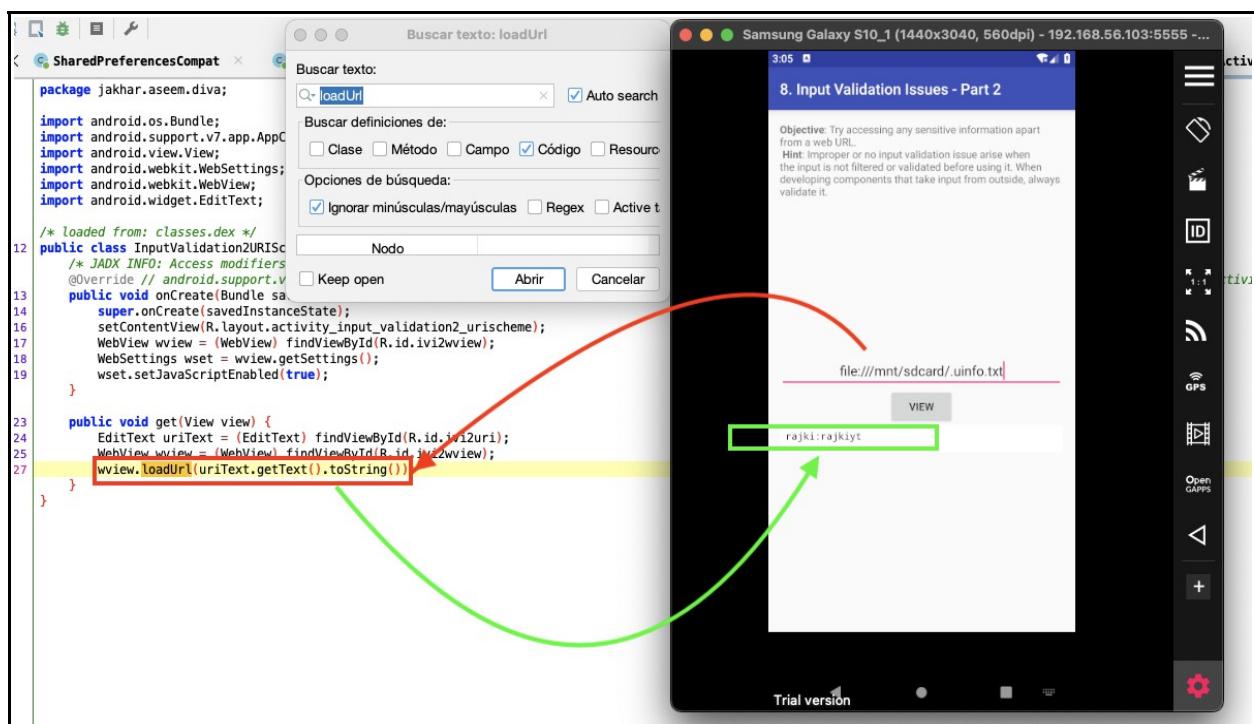


Y nos vuelca toda la tabla ***sqluser***:



## 1.8 - Input validation issues 2

En este **view** método **get()** obtiene el string del **EditText** y sin ningún tipo de verificación lo pasa como parámetro a **loadUrl()**, por lo tanto podemos acceder a cualquier parte de la que aplicación pueda acceder dentro del teléfono, servidores FTP, etc..



## 1.9 - Acces Control Issues 1

En Android los Intent son objetos que tienen información que el sistema usa para determinar que componente debe iniciar.

Tenemos 2 tipo de Intent:

- **Explícito**, estos nos permiten invocar en nuestra APP servicios de los que no disponemos pero sin embargo se nos permite iniciarla en la APP.
  - **Implicito**, estos son los que enviamos a otra aplicación con unos parámetros determinados que necesite la otra APP para invocar `startActivity()`

Si pulsamos en el View de la app “VIEW API CREDENTIALS” podemos observar en el logcat como se acciona el Intent al pulsar el botón, el problema es que en el código no se realiza ningún tipo de control de acceso por lo que en teoría también deberíamos de poder accionarlo desde fuera de la APP.

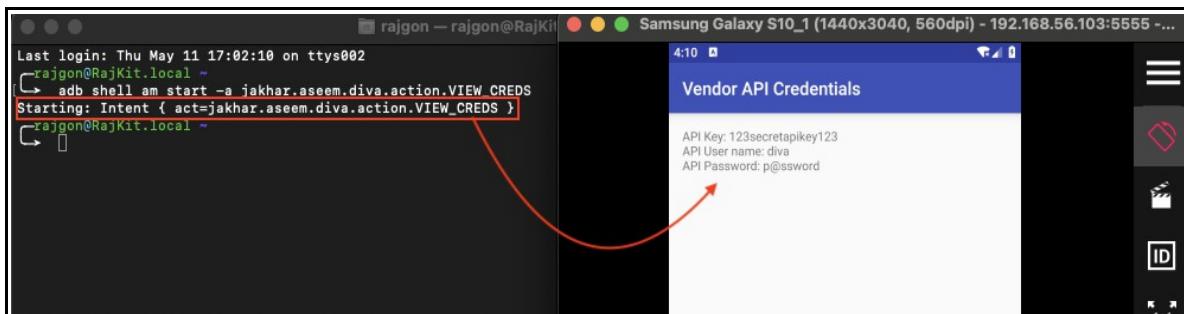
En el código se ve claramente que no hay ninguna estructura de control de flujo que determine si un acceso es lícito o no, lo único que verifica es que no contenga un valor NULL:

```

public void viewAPICredentials(View view) {
    Intent i = new Intent();
    i.setAction("jakhar.aseem.diva.action.VIEW_CREDS");
    if (i.resolveActivity(getApplicationContext()) != null) {
        startActivity(i);
        return;
    }
    Toast.makeText(this, "Error while getting API details", 0).show();
    Log.e("Diva-acil", "Couldn't resolve the Intent VIEW_CREDS to our activity");
}

```

Iniciaremos la activamente a través adb shell enviando “**am start -n**” y el nombre del componente tal y como esta definido en *manifest*.



## 1.10 - Input Validation Issues 2

Desde el entorno virtual que e creado para las dependencias de python y con *drozer agent* arrancado en el emulador de android escuchando en el puerto 31415, nos conectamos:

```

[rajon@RajKit.local ~/drozer
└─$ adb forward tcp:31415 tcp:31415
31415

```

```

[rajon@RajKit.local ~] Connection refused
[rajon@RajKit.local ~] ↵ drozer console connect
[rajon@RajKit.local ~] /Users/rajon/drozer/android-drozer/lib/python2.7/site-packages/OpenSSL/crypto.py
[rajon@RajKit.local ~] Warning: This code is no longer supported by the Python core team. Support for it is now deprecated in crypt
[rajon@RajKit.local ~] from cryptography import utils, x509
[rajon@RajKit.local ~] :0: UserWarning: You do not have a working installation of the service_identity module. Install it from <https://pypi.python.org/pypi/service_identity> and make sure all of its dependencies are installed.
[rajon@RajKit.local ~] Twisted can perform only rudimentary TLS client hostname verification. Hostnames which
[rajon@RajKit.local ~] e rejected.
[rajon@RajKit.local ~] Selecting 5d7bfec0acef51 (Genymotion Galaxy S10 9)

[rajon@RajKit.local ~] ..
[rajon@RajKit.local ~] ..0...
[rajon@RajKit.local ~] ..a... . .... . ..nd
[rajon@RajKit.local ~] .ro..idsnemesisand..pr
[rajon@RajKit.local ~] .otectorandroide
[rajon@RajKit.local ~] .,sisandprotectorandroide.
[rajon@RajKit.local ~] ..nemesisandprotectorandroide..
[rajon@RajKit.local ~] .emesisandprotectorandroide.
[rajon@RajKit.local ~] ..isandp,,,rotectorandroe,,idsnem
[rajon@RajKit.local ~] .isisandp..rotectorandroide.snemisis.
[rajon@RajKit.local ~] .andprotectorandroide.nemisisandprotec
[rajon@RajKit.local ~] .torandroide.nemesisandprotectorandroide.
[rajon@RajKit.local ~] .nemesisandprotectorandroide.nemesisan
[rajon@RajKit.local ~] .dprotectorandroide.nemesisandprotector.

[rajon@RajKit.local ~] drozer Console (v2.4.4)
[rajon@RajKit.local ~] dz> []

```

Podemos observar en el código como se va iniciar un **Intent → startActivity()** en función del checkbox seleccionado: esta actividad es **APICreeds2Activity** la cual va realizar una comprobación del radio button.

Verificara que que no es **null** antes de iniciar la actividad:

The screenshot shows two code snippets side-by-side. The left snippet is in Java and the right is in Python. Red arrows point from the variable declaration in the Java code to its corresponding assignment in the Python code.

```
20     setContentView(R.layout.activity_detailed_credentials);
21
22     public void viewAPICredentials(View view) {
23         RadioButton rbgnew = (RadioButton) findViewById(C0319R.C0321id.aci2rbregnow);
24         Intent i = new Intent();
25         boolean chk_pin = rbgnew.isChecked();
26         i.setAction("jakhar.aseem.diva.action.VIEW_CREDS2");
27         i.putExtra(getString(C0319R.string.chk_pin), chk_pin);
28         if (i.resolveActivity(getApplicationContext()) != null) {
29             startActivity(i);
30         }
31     }
32
33     return;
34
35     Buscar texto: 2131492973
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
167
168
169
169
170
171
172
173
174
175
176
177
177
178
179
179
180
181
182
183
184
185
185
186
187
187
188
189
189
190
191
192
193
193
194
195
195
196
197
197
198
199
199
200
200
201
201
202
202
203
203
204
204
205
205
206
206
207
207
208
208
209
209
210
210
211
211
212
212
213
213
214
214
215
215
216
216
217
217
218
218
219
219
220
220
221
221
222
222
223
223
224
224
225
225
226
226
227
227
228
228
229
229
230
230
231
231
232
232
233
233
234
234
235
235
236
236
237
237
238
238
239
239
240
240
241
241
242
242
243
243
244
244
245
245
246
246
247
247
248
248
249
249
250
250
251
251
252
252
253
253
254
254
255
255
256
256
257
257
258
258
259
259
260
260
261
261
262
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
```

En el *manifest* podemos observar como esa acción esta categorizada como DEFAULT lo que significa que se le va a ocultar al usuario:

```
<intent-filter>
    <action android:name="jakhar.aseem.diva.action.VIEW_CREDS2"/>
    <category android:name="android.intent.category.DEFAULT"/>
```

Realizamos una búsqueda de **chk\_pin** que nos lleva a **strings.xml** donde obtenemos en nombre del *value* que debe recibir false como parámetro para validar el acceso:

```
40 <string name="chk_pin">check pin</string>
41 <string name="err_pin">incorrect pin</string>
```

```
Button vbutton = (Button) findViewById(C0319R.C0321Id.aci2button);
Intent i = getIntent();
boolean bcheck = i.getBooleanExtra(getString(C0319R.string.chk_pin), true);
if (bcheck == true) goto L6;
apiview.setText("TWEETER API Key: secrettveeterapikey\nAPI User name: diva2\nAPI Password: p@ssword2");
return;
```

```
    @Override // android.support.v4.app.FragmentActivity, android.support.v4.app.FragmentManagerActivity, android.support.v4.app.FragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(C0319R.layout.activity_apicreds2);
        TextView apicview = (TextView) findViewById(C0319R.C0321id.apic2TextView);
        EditText pintext = (EditText) findViewById(C0319R.C0321id.aci2pinText);
        Button vbutton = (Button) findViewById(C0319R.C0321id.aci2button);
        Intent i = getIntent();
        boolean bcheck = i.getBooleanExtra(getString(C0319R.string.chk_pin), true);
if (bcheck == true) goto L6;
        apicview.setText("TVEETER API Key: secrettveeterapikey\nAPI User name: diva2\nAPI Password: p@ssword2");
        return;
    }

L6:
    apicview.setText("Register yourself at http://payatu.com to get your PIN and then login with that PIN!");
    pintext.setVisibility(0);
    vbutton.setVisibility(0);
}

public void viewCreds(View view) {
    Toast.makeText(this, "Invalid PIN. Please try again", 0).show();
}
```

Por lo tanto a partir de aquí deducimos que si enviamos a **APICreds2Activity** el parámetro booleano **check\_pin** como false nos devolverá las credenciales la aplicación, para ello usaremos dropper como inyector:

## 1.11 - Input Validation Issues 3

Observando el código de AC3A se puede ver al final como se realiza un `intent → startActivity()` de `AccessControl2NotesActivity.class`

```
public void goToNotes(View view) {
    Intent i = new Intent(this, AccessControl3NotesActivity.class);
    startActivity(i);
```

Accedemos a esta clase para ver como continua el flujo de ejecución y vemos como se realiza una comparación mediante `equals` del pin, que de ser correcto se realiza una llamada a `CONTENT_URI` dentro de la clase `NotesProvider`:

```
public void accessNotes(View view) {
    EditText pinTxt = (EditText) findViewById(C0319R.C0321id.aci3notesPinText);
    Button abutton = (Button) findViewById(C0319R.C0321id.aci3accessbutton);
    SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);
    String pin = spref.getString(getString(C0319R.string.pkey), "");
    String userpin = pinTxt.getText().toString();
    if (userpin.equals(pin)) {
        ListView lview = (ListView) findViewById(C0319R.C0321id.aci3nlistView);
        Cursor cr = getContentResolver().query(NotesProvider.CONTENT_URI, new String[]{"_id", "title", "note"}, null, null, null);
        String[] columns = {"title", "note"};
        int[] fields = {C0319R.C0321id.title_entry, C0319R.C0321id.note_entry};
        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, C0319R.layout.notes_entry, cr, columns, fields, 0);
        lview.setAdapter((ListAdapter) adapter);
        pinTxt.setVisibility(4);
        abutton.setVisibility(4);
        return;
    }
    Toast.makeText(this, "Please Enter a valid pin!", 0).show();
}
```

En la clase `NotesProvider` vemos como esta el valor de `CONTENT_URI`:

```
/* loaded from: classes.dex */
public class NotesProvider extends ContentProvider {
    static final String AUTHORITY = "jakhar.aseem.diva.provider.notesprovider";
    static final String CREATE_TBL_QRY = "CREATE TABLE notes (_id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT NOT NULL, note TEXT NOT NULL);";
    static final String C_ID = "_id";
    static final String C_NOTE = "note";
    static final String C_TITLE = "title";
    static final String DBNAME = "divanotes.db";
    static final int DBVERSION = 1;
    static final String DROP_TBL_QRY = "DROP TABLE IF EXISTS notes";
    static final int PATH_ID = 2;
    static final int PATH_TABLE = 1;
    static final String TABLE = "notes";
    SQLiteDatabase mDB;
    static final Uri CONTENT_URI = Uri.parse("content://jakhar.aseem.diva.provider.notesprovider/notes");
    static final UriMatcher uriMatcher = new UriMatcher(-1);
```

Podemos realizar una consulta a través de URI con `adb` o `drozer`, también podríamos acceder directamente la base de datos `divanotes.db`:

```
[rajan@RajKit.local ~] drozer
↳ adb shell content query --uri content://jakhar.aseem.diva.provider.notesprovider/notes
Row: 0 _id=5, title=Exercise, note=Alternate days running
Row: 1 _id=4, title=Expense, note=Spent too much on home theater
Row: 2 _id=6, title=Weekend, note=b3333333333333
Row: 3 _id=3, title=holiday, note=Either Goa or Amsterdam
Row: 4 _id=2, title=home, note=Buy toys for baby, Order dinner
Row: 5 _id=1, title=office, note=10 Meetings. 5 Calls. Lunch with CEO
```

Si accedemos a `/data/data/[nombre apk]/databases` tenemos acceso a las bases de datos que utiliza la aplicación, simplemente elegimos la BD que obtenemos en el código de **NoteProvider**, realizamos una consulta de los nombre de las tablas y otra consulta con los datos que queremos que nos muestre de esa tabla:

```

:/data/data # cd jakhar.aseem.diva/
:/data/data/jakhar.aseem.diva # ls
app_textures app_webview cache code_cache databases lib shared_prefs uinfo4211003095055429750tmp uinfo4478281446761325151tmp
:/data/data/jakhar.aseem.diva # cd databases
divanotes.db divanotes.db-shm divanotes.db-wal ids2 ids2-shm ids2-wal sqlite3 sqli-shm sqli-wal
:/data/data/jakhar.aseem.diva/databases # ls
SQLite version 3.22.0 2019-09-03 18:36:11
Enter ".help" for usage hints.
sqlite> .tables
android_metadata notes
sqlite> select * from notes;
1|office|10 Meetings. 5 Calls. Lunch with CEO
2|home|Buy toys for baby, Order dinner
3|holiday|Either Goa or Amsterdam
4|Expense|Spent too much on home theater
5|Exercise|Alternate days running
6|Weekend|b33333333333333
sqlite>

```

## 1.12 - Hardcoding Issues 2

Analizando el código observamos como se inicia una instancia de la clase **DivaJni** y se realiza en el método `acces()` una comprobación:

```

public class Hardcode2Activity extends AppCompatActivity {
    private DivaJni djni;

    /* JADY INFO: Access modifiers changed from: protected */
    @Override // android.support.p003v7.app.AppCompatActivity, android.support.p000v4.app.FragmentActivity, android.support.p000v4.app.BaseFragmentActivityDonut,
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode2);
        this.djni = new DivaJni();
    }

    public void access(View view) {
        EditText hckey = findViewById(C0319R.id.hckey);
        if (this.djni.access(hckey.getText().toString()) != 0) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}

```

Realizando un seguimiento accedemos a la clase **DivaJni** y observamos como el código invoca una librería con el **string** contenido en la variable `soName`:

```

/* loaded from: classes.dex */
public class DivaJni {
    private static final String soName = "divajni";
    public native int access(String str);

    public native int initiateLaunchSequence(String str);

    static {
        System.loadLibrary(soName);
    }
}

```

Esto significa que automáticamente cargará la librería *divajni.so* almacenada en la carpeta **Resources/lib/[arquitectura]**

```
sm. reader. not found
[127]:/data/data/jakhar.aseem.diva # ls -la
total 28
drwxr-xr-x 2 system system 4096 2023-05-10 09:38 .
drwxr-xr-x 3 system system 4096 2023-05-10 09:38 ..
-rwxr-xr-x 1 system system 5164 2015-12-31 14:37 libdivajni.so
:/data/data/jakhar.aseem.diva/lib #
```

Sabemos que este archivo es una biblioteca nativa compilada, así que la descargaremos de android y realizaremos una búsqueda de posibles datos a través de sus secciones:

```
[aigon@RajKit.local ~/drozer
objdump --section-headers libdivajni.so

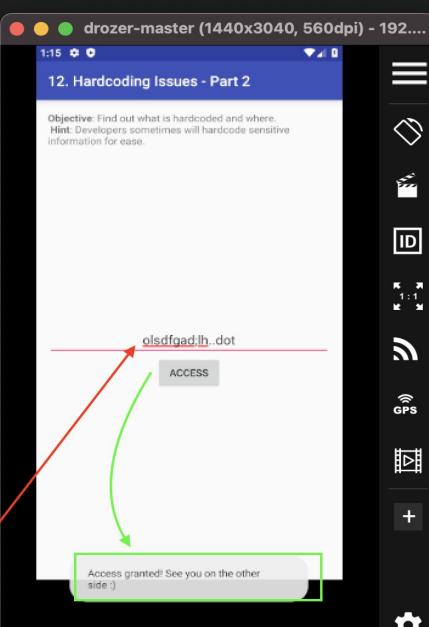
libdivajni.so: file format elf32-i386

Sections:
Idx Name      Size   VMA      Type
0           00000000 0000000000000000
1 .dynsym    000000b0 000000114
2 .dynstr    000000e8 000001c4
3 .hash      00000040 000002ac
4 .rel.dyn   00000010 000002ec
5 .rel.plt   00000020 000002fc
6 .plt       00000050 00000328 TEXT
7 .text      00000186 00000370 TEXT
8 .rodata    00000014 000004f6 DATA
9 .eh_frame  00000150 0000050c DATA
10 .eh_frame_hdr 00000054 0000065c DATA
11 .fini_array 00000008 00001ee0
12 .init_array 00000004 00001ee8
13 .dynamic   000000f8 00001eec
14 .got       00000000 00001fe4 DATA
15 .got.plt   0000001c 00001fe4 DATA
16 .data      00000004 00002000 DATA
17 .bss       00000000 00002004 BSS
18 .comment   00000010 00000000
19 .note.gnu.gold-version 0000001c 00000000
20 .shstrtab  00000003 00000000

[aigon@RajKit.local ~/drozer
objdump -s -j .rodata libdivajni.so

libdivajni.so: file format elf32-i386
Contents of section .rodata:
04f6 6f6c7364 66676164 3b6c6800 2e646f74 olsdfgad;lh..dot.
0506 646f7400
[aigon@RajKit.local ~/drozer
```

Sabemos que la sección **.rodata** se almacenan valores de constantes y se colocan datos inicializados, ademas sus datos nunca son modificados.



The drozer application interface shows the following details:

- Objective:** Find out what is hardcoded and where.
- Hints:** Developers sometimes will hardcode sensitive information for ease.
- Access Granted:** Access granted! See you on the other side :)

```
[aigon@RajKit.local ~/drozer
objdump --section-headers libdivajni.so

libdivajni.so: file format elf32-i386

Sections:
Idx Name      Size   VMA      Type
0           00000000 0000000000000000
1 .dynsym    000000b0 000000114
2 .dynstr    000000e8 000001c4
3 .hash      00000040 000002ac
4 .rel.dyn   00000010 000002ec
5 .rel.plt   00000020 000002fc
6 .plt       00000050 00000328 TEXT
7 .text      00000186 00000370 TEXT
8 .rodata    00000014 000004f6 DATA
9 .eh_frame  00000150 0000050c DATA
10 .eh_frame_hdr 00000054 0000065c DATA
11 .fini_array 00000008 00001ee0
12 .init_array 00000004 00001ee8
13 .dynamic   000000f8 00001eec
14 .got       00000000 00001fe4 DATA
15 .got.plt   0000001c 00001fe4 DATA
16 .data      00000004 00002000 DATA
17 .bss       00000000 00002004 BSS
18 .comment   00000010 00000000
19 .note.gnu.gold-version 0000001c 00000000
20 .shstrtab  00000003 00000000

[aigon@RajKit.local ~/drozer
objdump -s -j .rodata libdivajni.so

libdivajni.so: file format elf32-i386
Contents of section .rodata:
04f6 6f6c7364 66676164 3b6c6800 2e646f74 olsdfgad;lh..dot.
0506 646f7400
```