Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO BÀI TẬP

Môn học: Bảo mật Web và Ứng dụng

Kỳ báo cáo: Lab 05

Tên chủ đề: Ôn tập

GV: Ngô Khánh Khoa

Ngày báo cáo: 30/05/2023

1. THÔNG TIN CHUNG:

Lớp: NT213.N21.ANTT.2

STT	Họ và tên	MSSV	Email
1	Phạm Phúc Đức	20520162	20520162@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Challenges 1	11/12
2	Challenges 2	100%

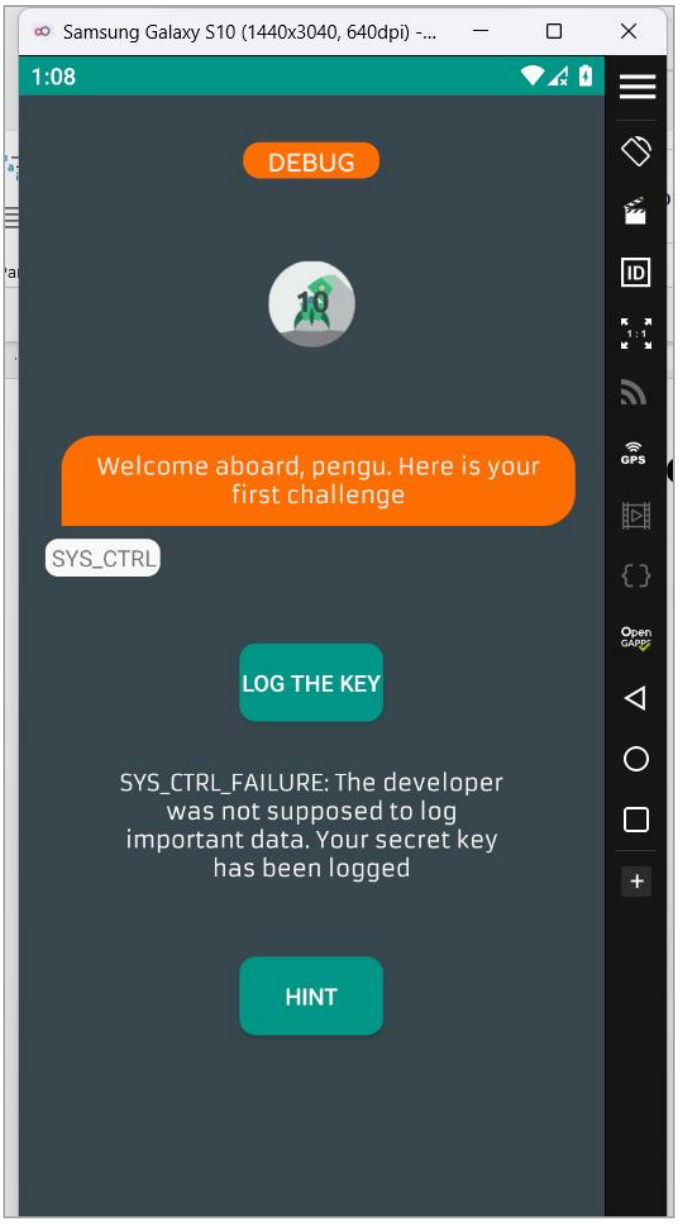
Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

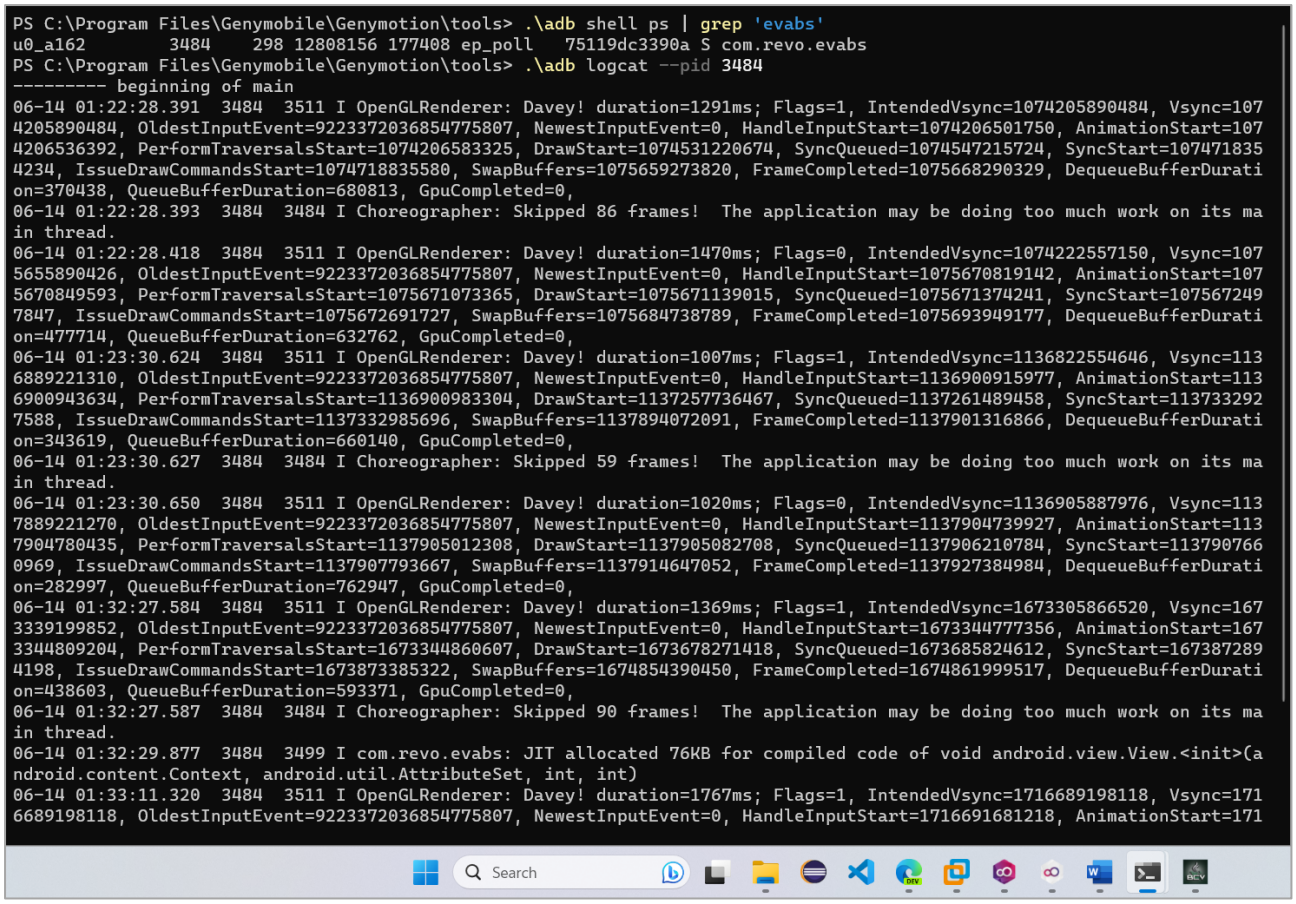
1. Challenges 1

1.1. Level 1 – Debug me



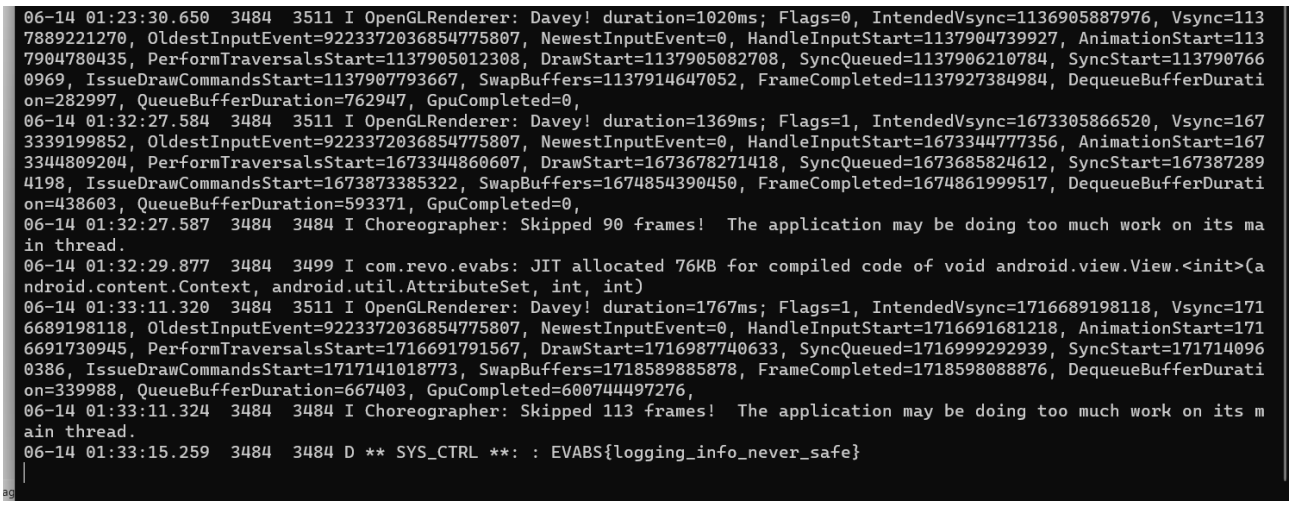
Hình 1: level 1

Level yêu cầu chúng ta tìm đọc log nên chúng ta sẽ đọc nội dung từ logcat:



Hình 2: Xem logcat của ứng dụng bằng pid

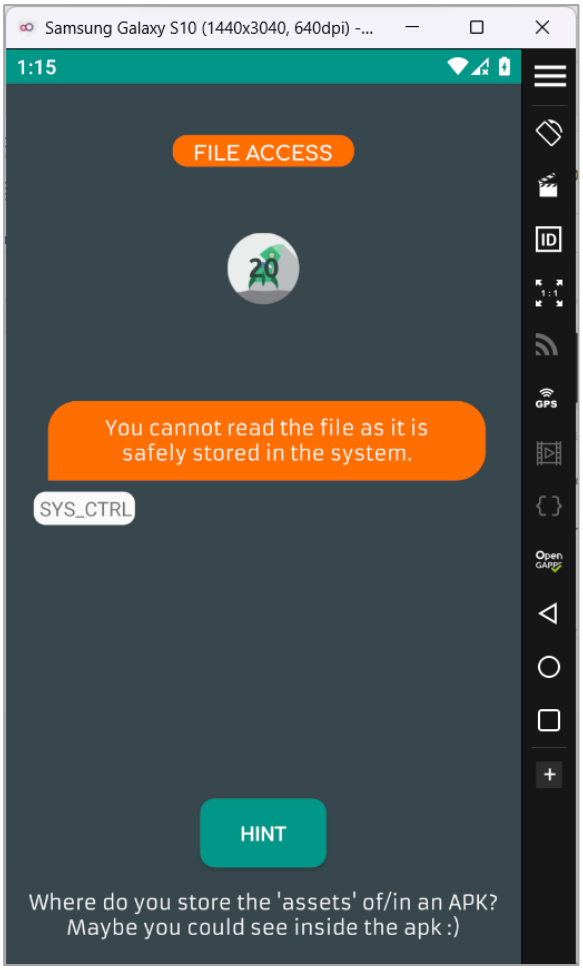
Phía dưới cùng:



Hình 3: Flag

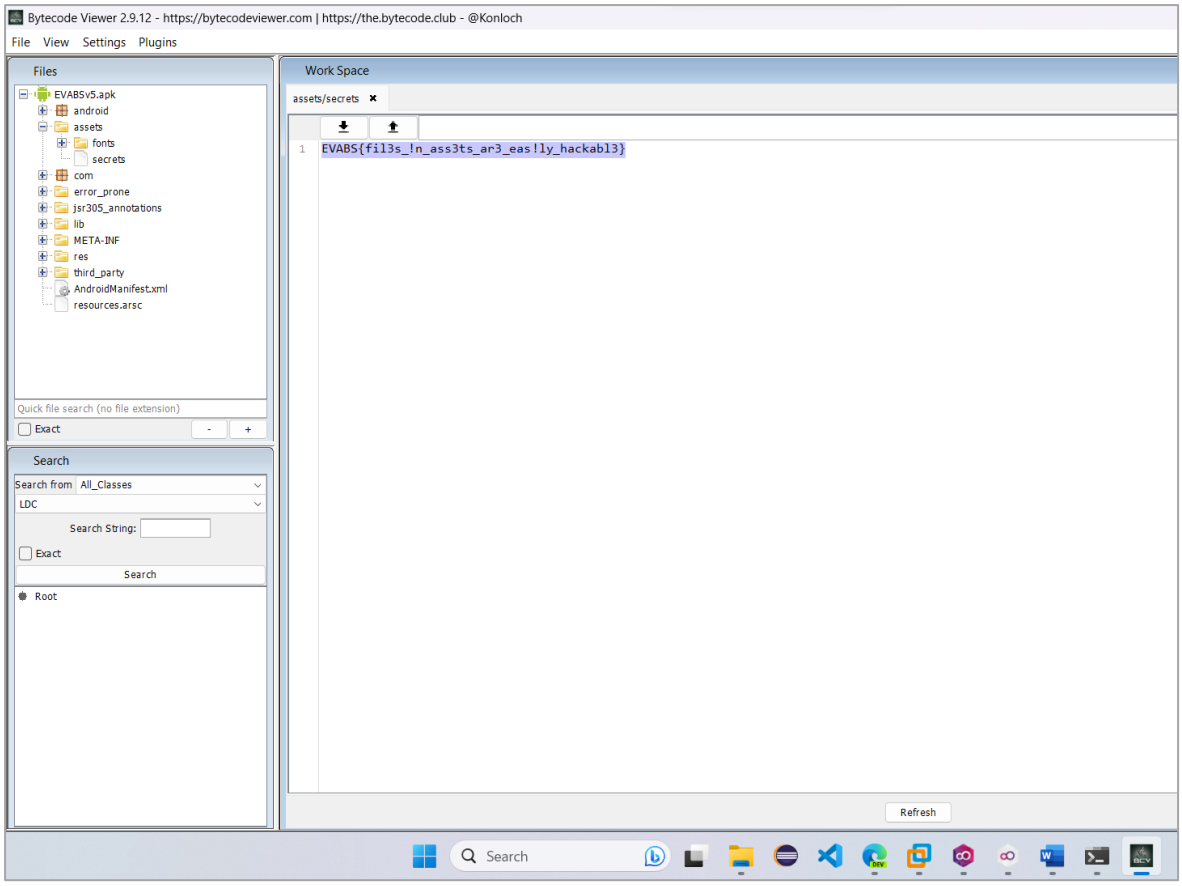
➔ EVABS{logging_info_never_safe}

1.2. Level 2 – File Access



Hình 4: Gợi ý câu 2

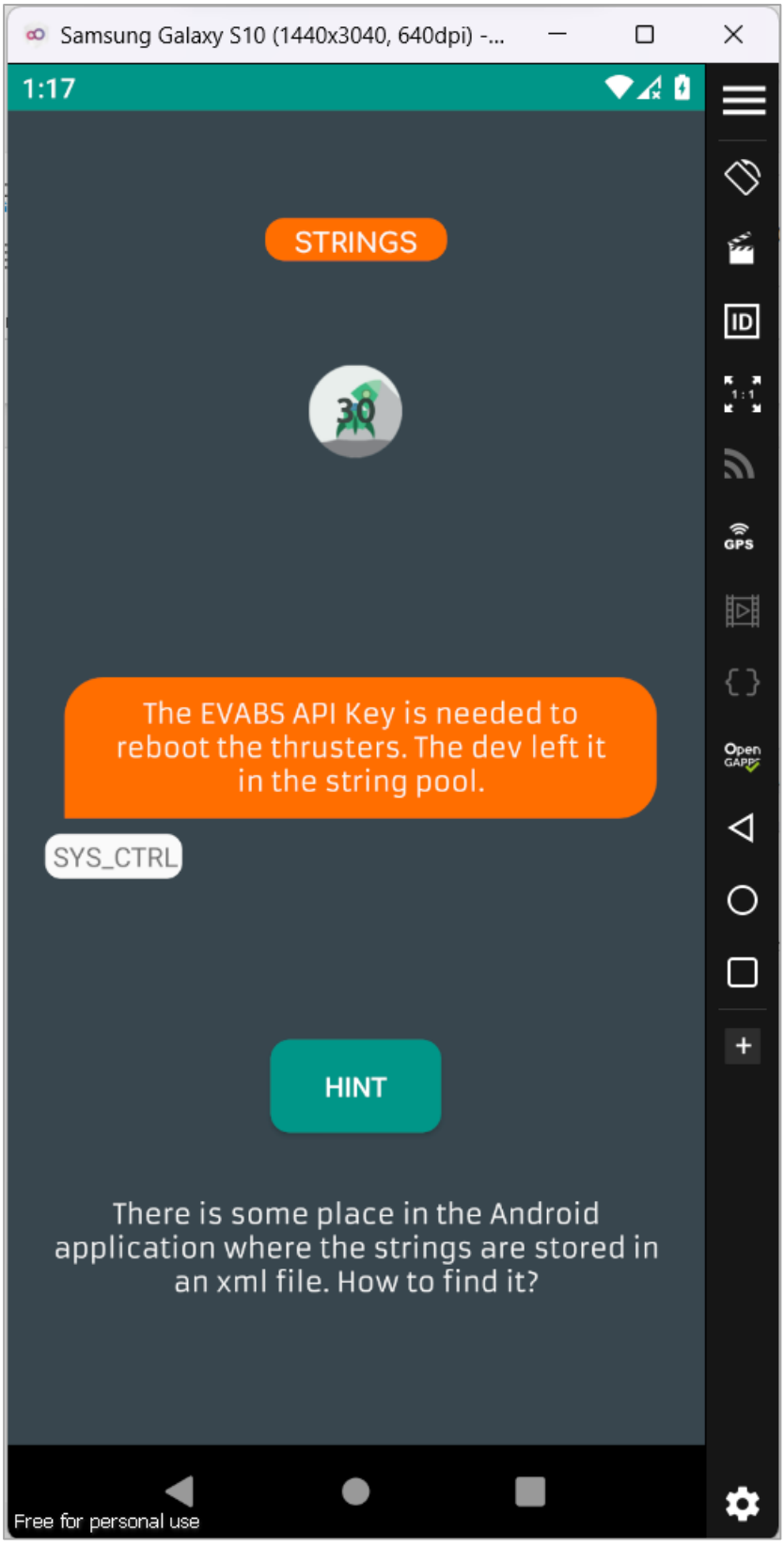
Theo như gợi ý, chúng ta sẽ đi tìm trong tệp assets khi decompile tệp .apk:



Hình 5: Tìm được flag

➔ EVABS{fil3s_!n_ass3ts_ar3_eas!ly_hackabl3}

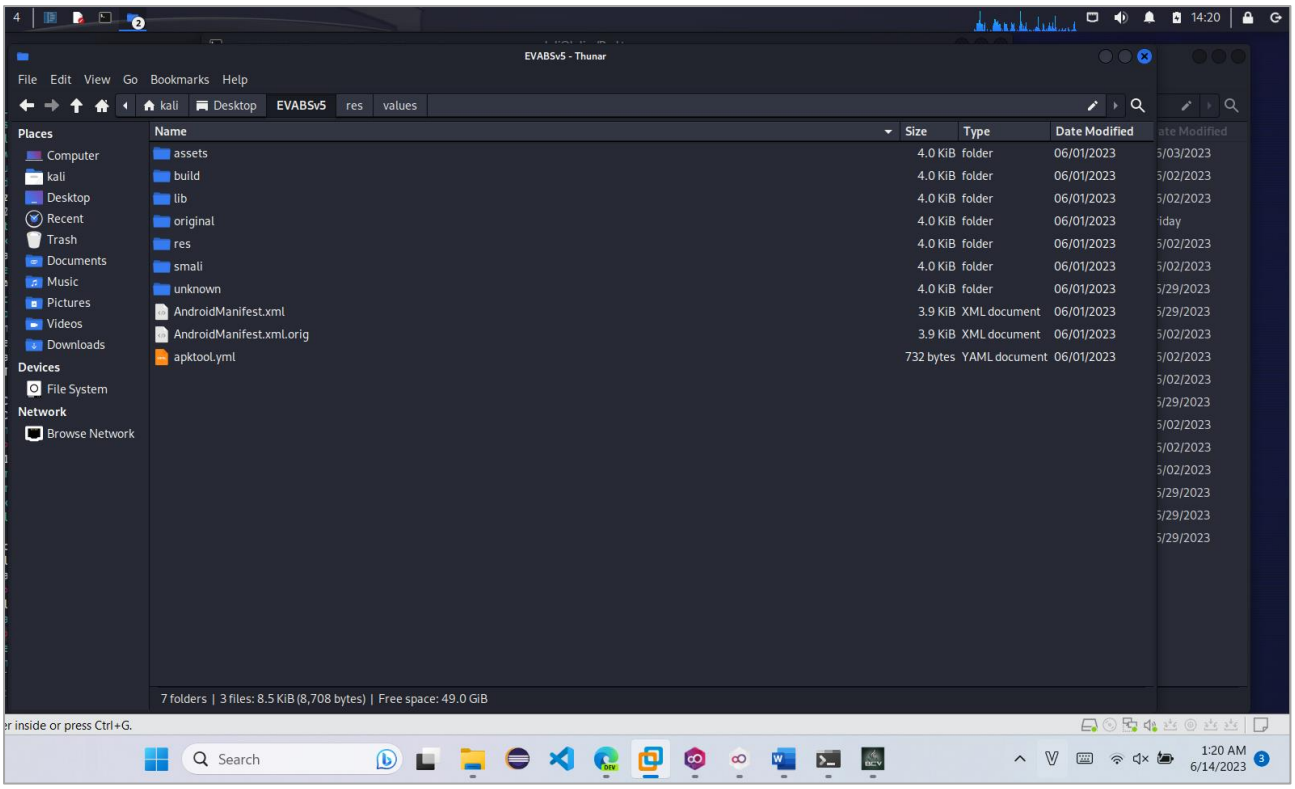
1.3. Level 3 - String



Hình 6: Gợi ý level 3

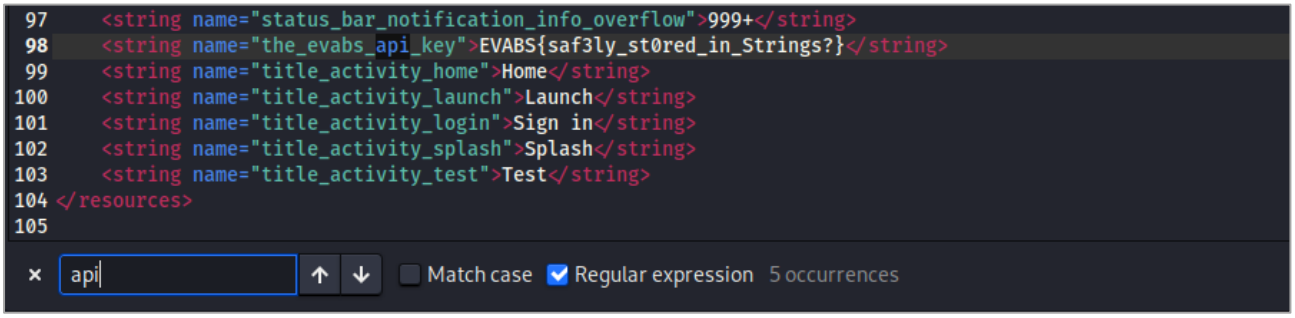
Gợi ý chỉ ra flag nằm trong tệp xml chứa các chuỗi -> có thể là tệp string.xml và nó luôn nằm trong thư mục values

Sử dụng apktool của kali để decompile (ByteCode Viewer không xem được tệp values)



Hình 7: Decompile

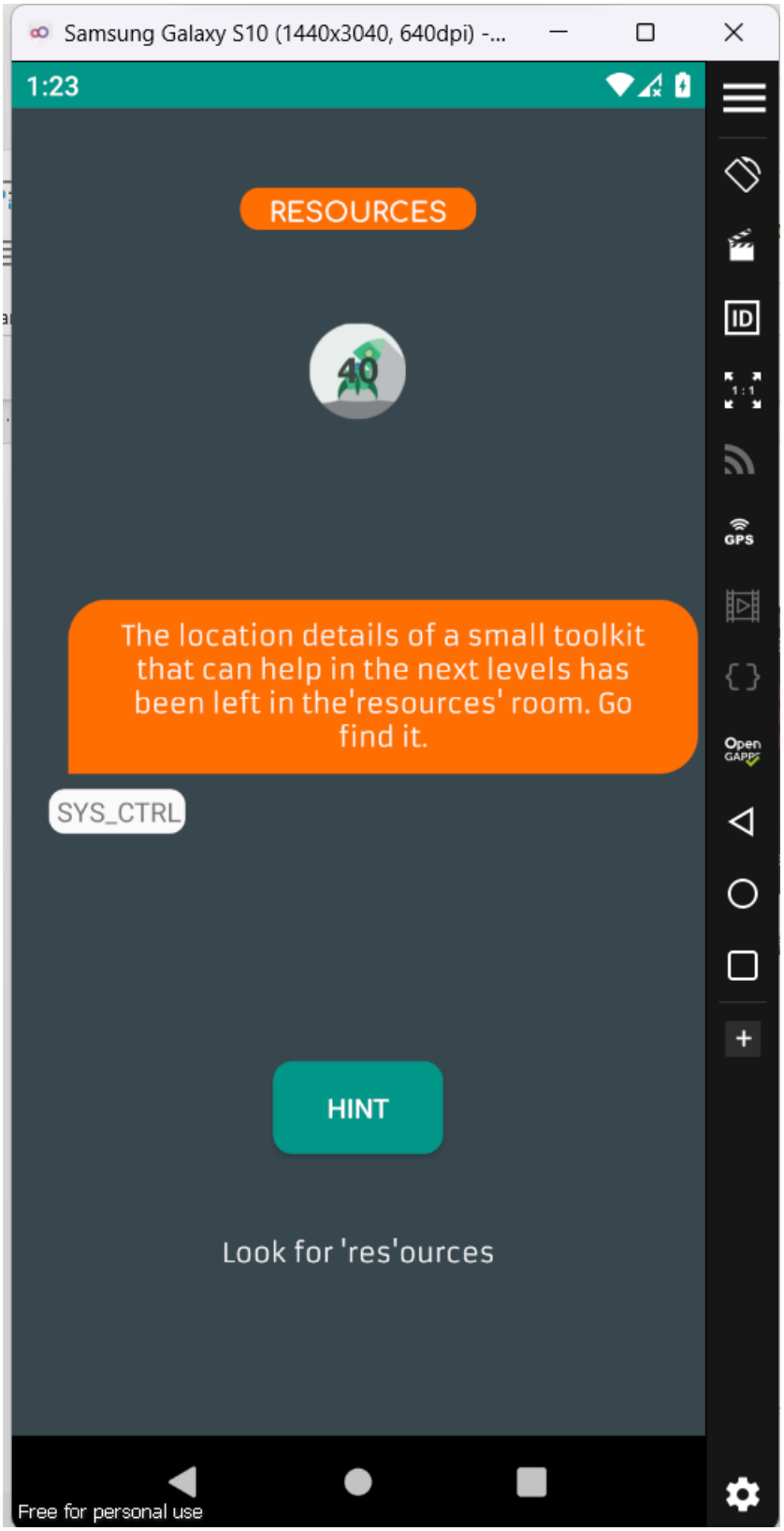
Truy cập tới res/values/string.xml:



*Hình 8: Tìm kiếm bằng từ khoá "**api**"*

➔ EVABS{saf3ly_st0red_in_Strings?}

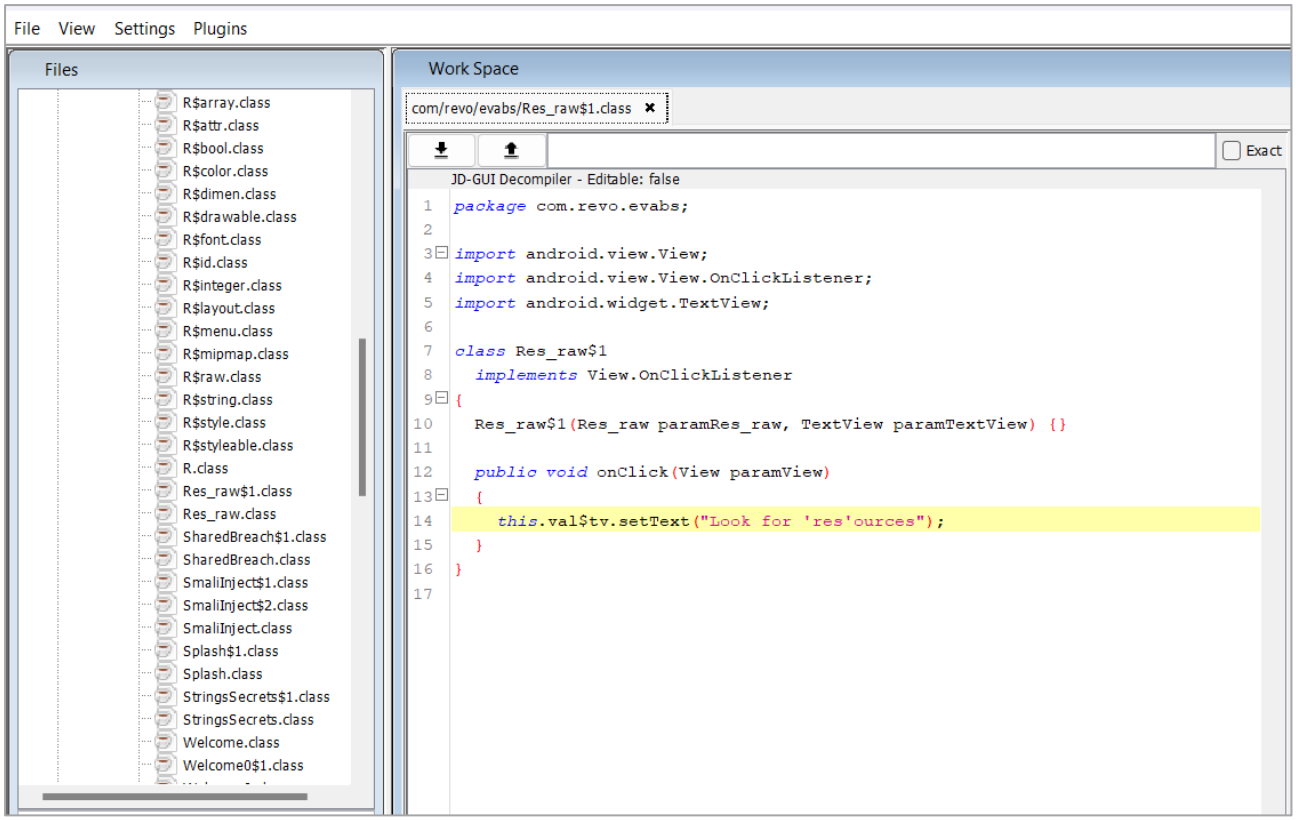
1.4. Level 4 - Resources



Hình 9: Level 4

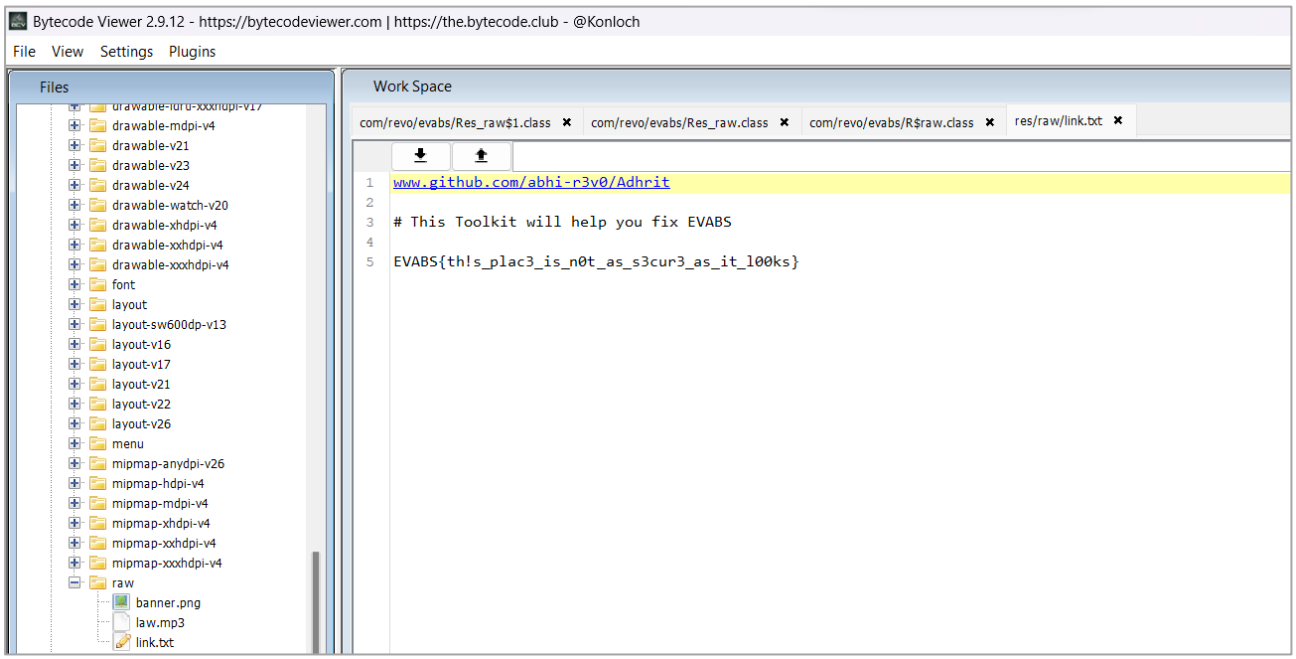
Resource đang nói đến cũng là tệp res, chúng ta tìm tới file có chứa "small toolkit" kia.

Theo như phân tích bằng BCV, tệp res_raw.class (gồm cả $1) sẽ liên quan tới level này:



Hình 10: Nội dung res_raw$1.class

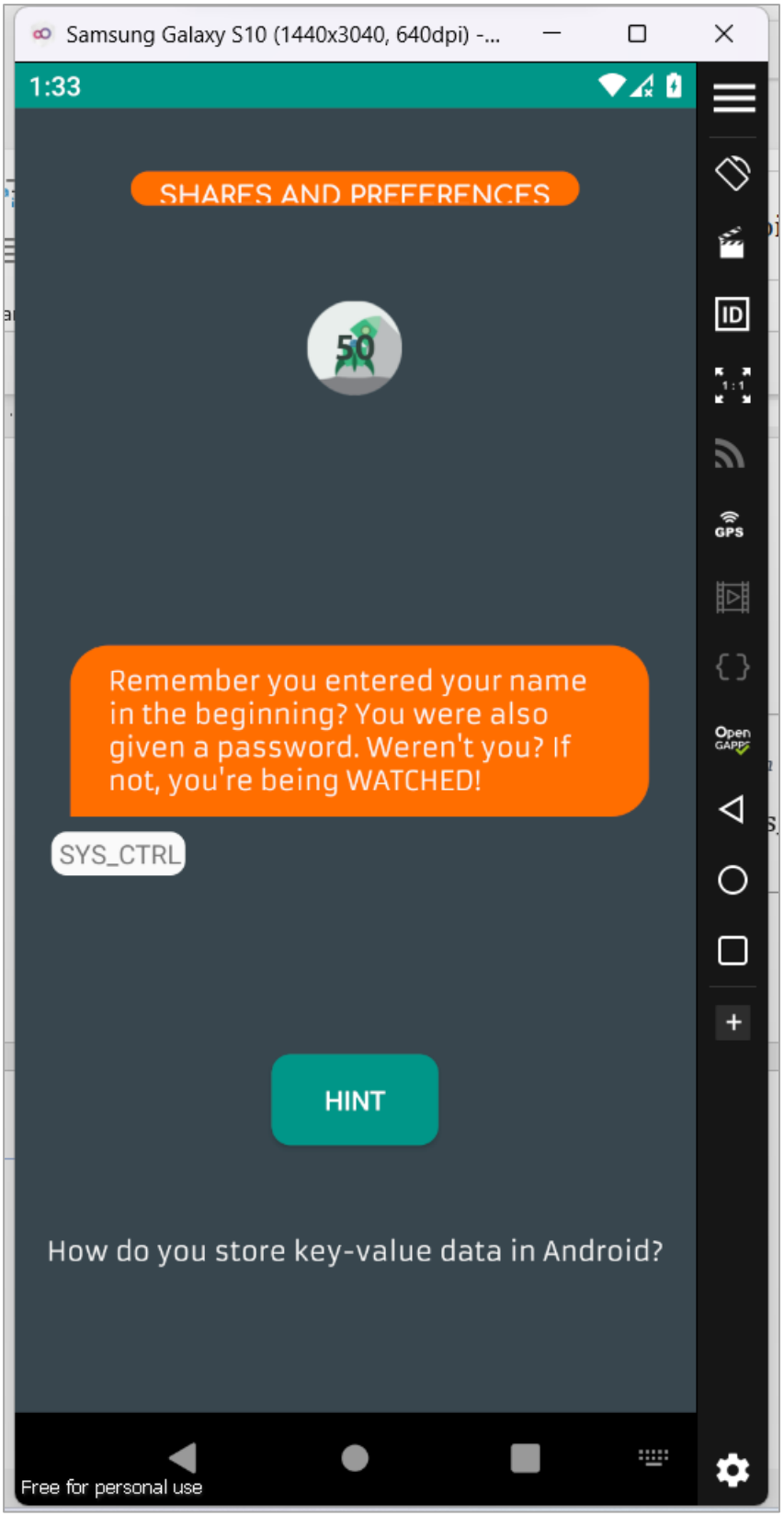
Dựa theo tên lớp, truy cập vào tệp res/raw:



Hình 11: Tìm thấy flag

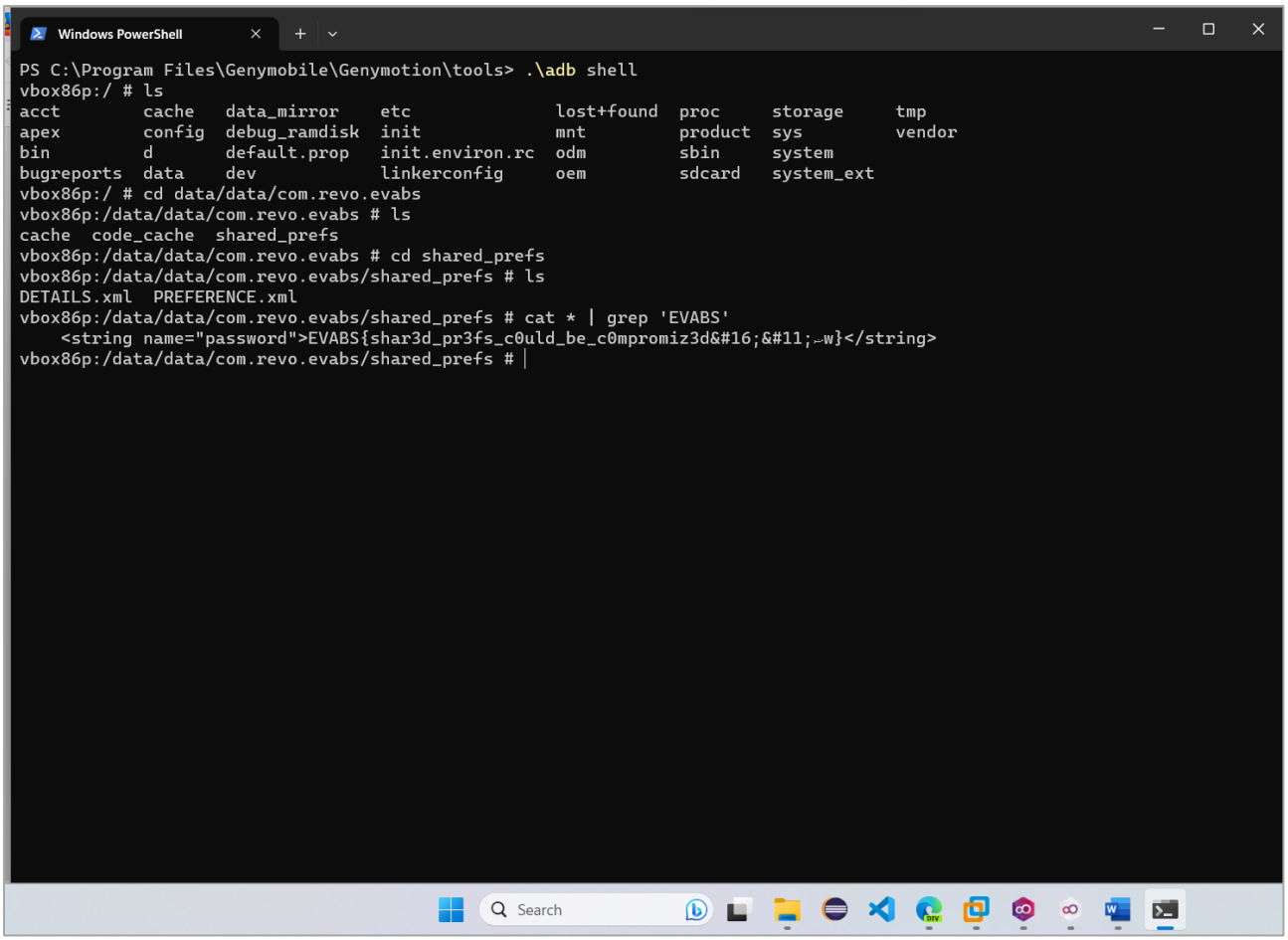
➔ EVABS{th!s_plac3_is_n0t_as_s3cur3_as_it_l00ks}

1.5. Level 5 – Shares and Preferences



Hình 12: Level 5

Thường thì dữ liệu các ứng dụng sẽ được lưu vào data/data/<app package> nên chúng ta sẽ mở các file trong đó để tìm mật khẩu.

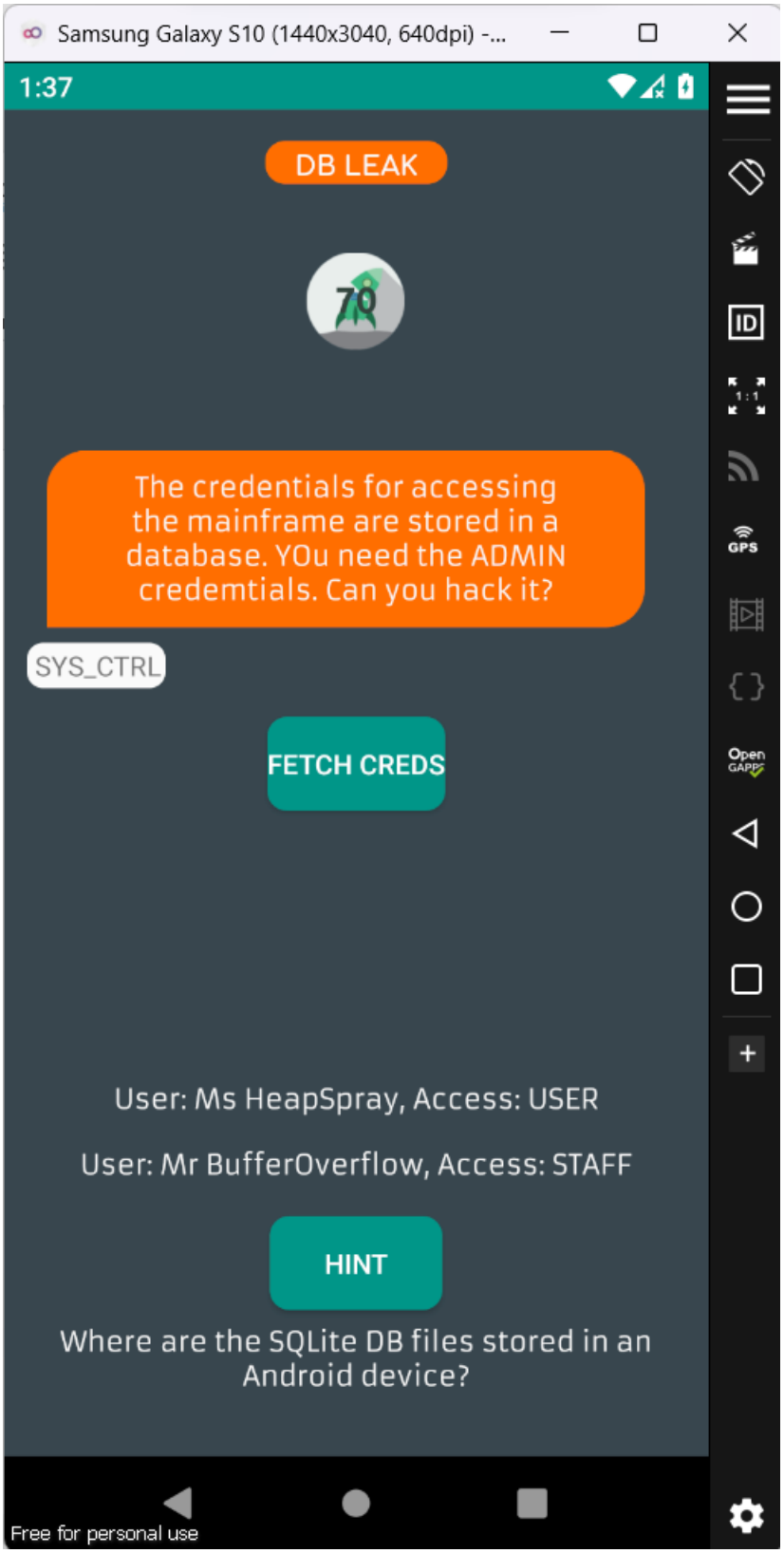


Hình 13: Lấy flag

Hai tệp đầu là cache và code_cache là 2 tệp thường thấy và log sẽ không ghi vào đây, chúng ta sẽ mở file shared_prefs và lấy key.

➔ Flag: EVABS{shar3d_pr3fs_c0uld_be_c0mpromiz3d&#16;&#11;w}

1.6. Level 6 – DB leak



Hình 14: Level 6

Level 6 yêu cầu chúng ta truy cập vào SQLite trong hệ thống, thực hiện truy cập lại vào data/data/com.revo.evabs:

```
vbox86p:/data/data/com.revo.evabs # ls
cache  code_cache  databases  shared_prefs
vbox86p:/data/data/com.revo.evabs # cat database
cat: database: No such file or directory
1|vbox86p:/data/data/com.revo.evabs # sqlite3 databases
Error: unable to open database "databases": unable to open database file
1|vbox86p:/data/data/com.revo.evabs # cat databases
cat: databases: Is a directory
1|vbox86p:/data/data/com.revo.evabs # ls
cache  code_cache  databases  shared_prefs
vbox86p:/data/data/com.revo.evabs # ls databases
MAINFRAME_ACCESS  MAINFRAME_ACCESS-journal
vbox86p:/data/data/com.revo.evabs # sqlite3 MAINFRAME_ACCESS
SQLite version 3.28.0 2020-05-06 18:46:38
Enter ".help" for usage hints.
sqlite> .tables
sqlite> .exit
vbox86p:/data/data/com.revo.evabs # cd databases
vbox86p:/data/data/com.revo.evabs/databases # sqlite3 MAINFRAME_ACCESS
SQLite version 3.28.0 2020-05-06 18:46:38
Enter ".help" for usage hints.
sqlite> .tables
CREDS             android_metadata
sqlite> SELECT * FROM CREDS
   ...> .exit
   ...> exit
   ...> exit
   ...> exit
1|vbox86p:/data/data/com.revo.evabs/databases # sqlite3 MAINFRAME_ACCESS
SQLite version 3.28.0 2020-05-06 18:46:38
Enter ".help" for usage hints.
sqlite> .tables
CREDS             android_metadata
sqlite> SELECT * FROM CREDS;
Dr.l33t|EVABS{sqlite_is_not_safe}□□
u|ADMIN
Mr BufferOverflow|0xNotSecureSQLite_|STAFF
Ms HeapSpray|SQLite_expl0it|USER
sqlite>
```

Hình 15: Xem giá trị có trong bảng

Chúng ta có thể biết được nó nằm trong bảng CREDS dó khi phân tích mã nguồn, chúng ta thấy tài khoản Dr.l33t có quyền ADMIN:

com/revo/evabs/Res_raw$1.class | com/revo/evabs/Res_raw.class | com/revo/evabs/R$raw.class | res/raw/link.txt | com/revo/evabs/DBLeak$1.class | com/revo/evabs/

JD-GUI Decompiler - Editable: false

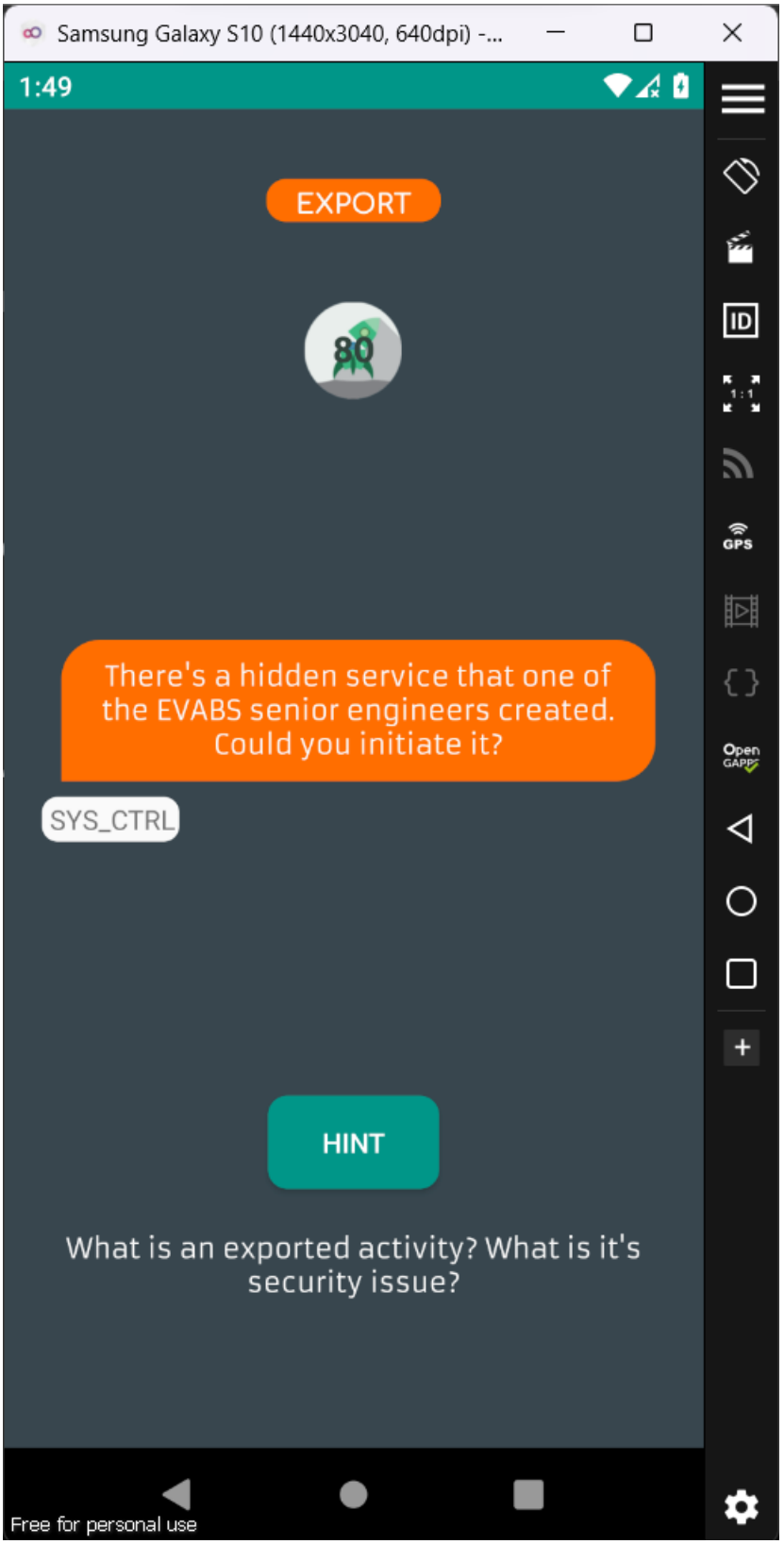
```
14  {
15    System.loadLibrary("native-lib");
16  }
17
18  protected void onCreate(Bundle paramBundle)
19  {
20    super.onCreate(paramBundle);
21    setContentView(2131492894);
22    String str1 = stringFromJNI();
23    SQLiteDatabase localSQLiteDatabase = openOrCreateDatabase("MAINFRAME_ACCESS", 0, null);
24    localSQLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS CREDS(admin VARCHAR, pass VARCHAR, access VARCHAR);");
25    StringBuilder localStringBuilder = new StringBuilder();
26    localStringBuilder.append("INSERT INTO CREDS VALUES('Dr.l33t', '");
27    localStringBuilder.append(str1);
28    localStringBuilder.append("' , 'ADMIN');");
29    localSQLiteDatabase.execSQL(localStringBuilder.toString());
30    localSQLiteDatabase.execSQL("INSERT INTO CREDS VALUES('Mr BufferOverflow', '0xNotSecureSQLite_', 'STAFF');");
31    localSQLiteDatabase.execSQL("INSERT INTO CREDS VALUES('Ms HeapSpray', 'SQLite_expl0it', 'USER');");
32    Cursor localCursor = localSQLiteDatabase.rawQuery("SELECT * FROM CREDS", null);
33    localCursor.moveToPosition(2);
34    String str2 = localCursor.getString(0);
35    String str3 = localCursor.getString(2);
36    localCursor.moveToPosition(1);
37    String str4 = localCursor.getString(0);
38    String str5 = localCursor.getString(2);
39    TextView localTextView1 = (TextView)findViewById(2131362078);
40    TextView localTextView2 = (TextView)findViewById(2131362079);
41    Button localButton = (Button)findViewById(2131361839);
42    DBLeak.1 local1 = new DBLeak.1(this, localTextView1, str2, str3, localTextView2, str4, str5);
43    localButton.setOnClickListener(local1);
44    TextView localTextView3 = (TextView)findViewById(2131362080);
45    ((Button)findViewById(2131361836)).setOnClickListener(new DBLeak.2(this, localTextView3));
46  }
47
48  public native String stringFromJNI();
49 }
50
```

Refresh

Hình 16: Phân tích

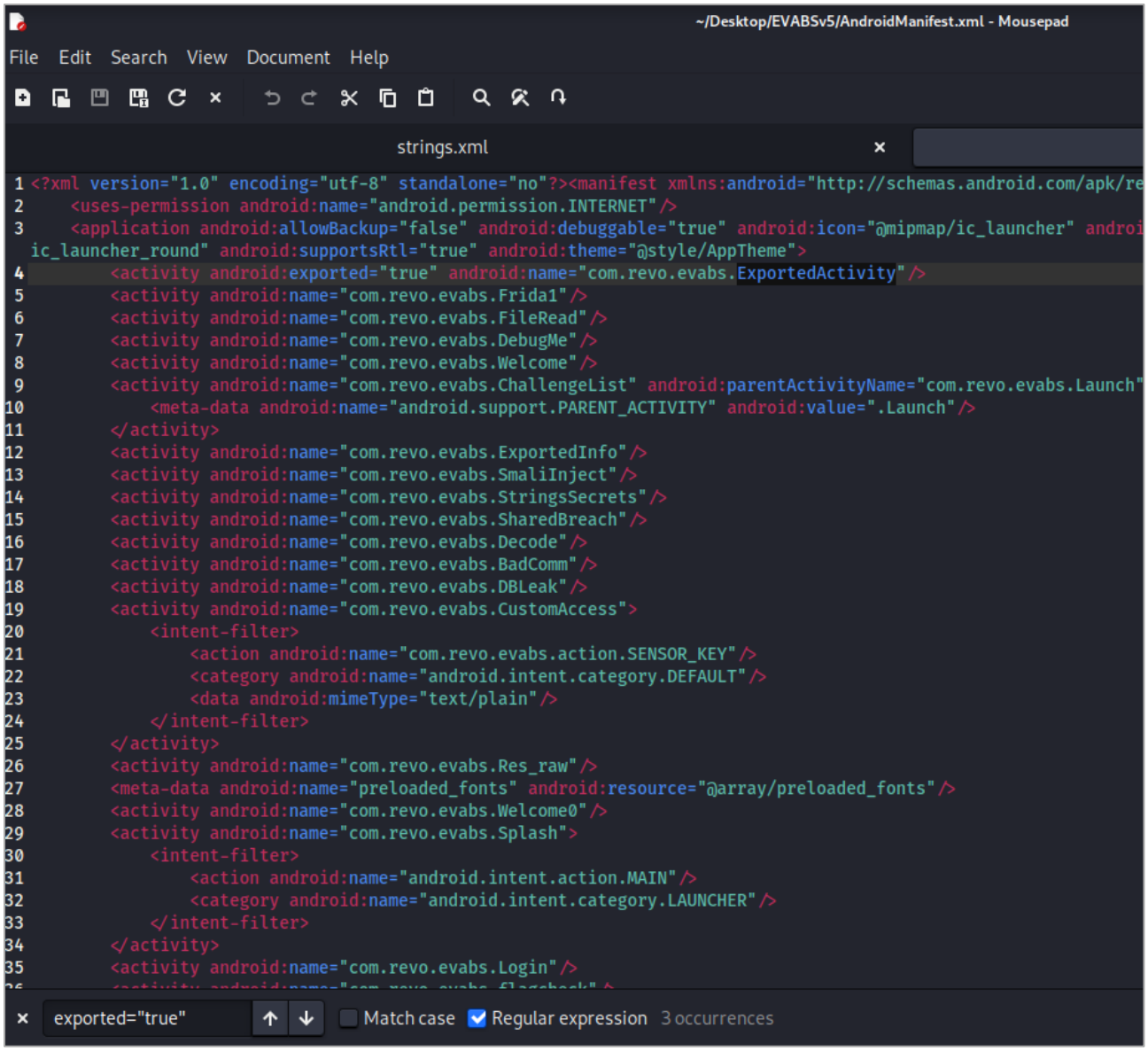
➔ flag: EVABS{sqlite_is_not_safe}

1.7. Level 7 - Export



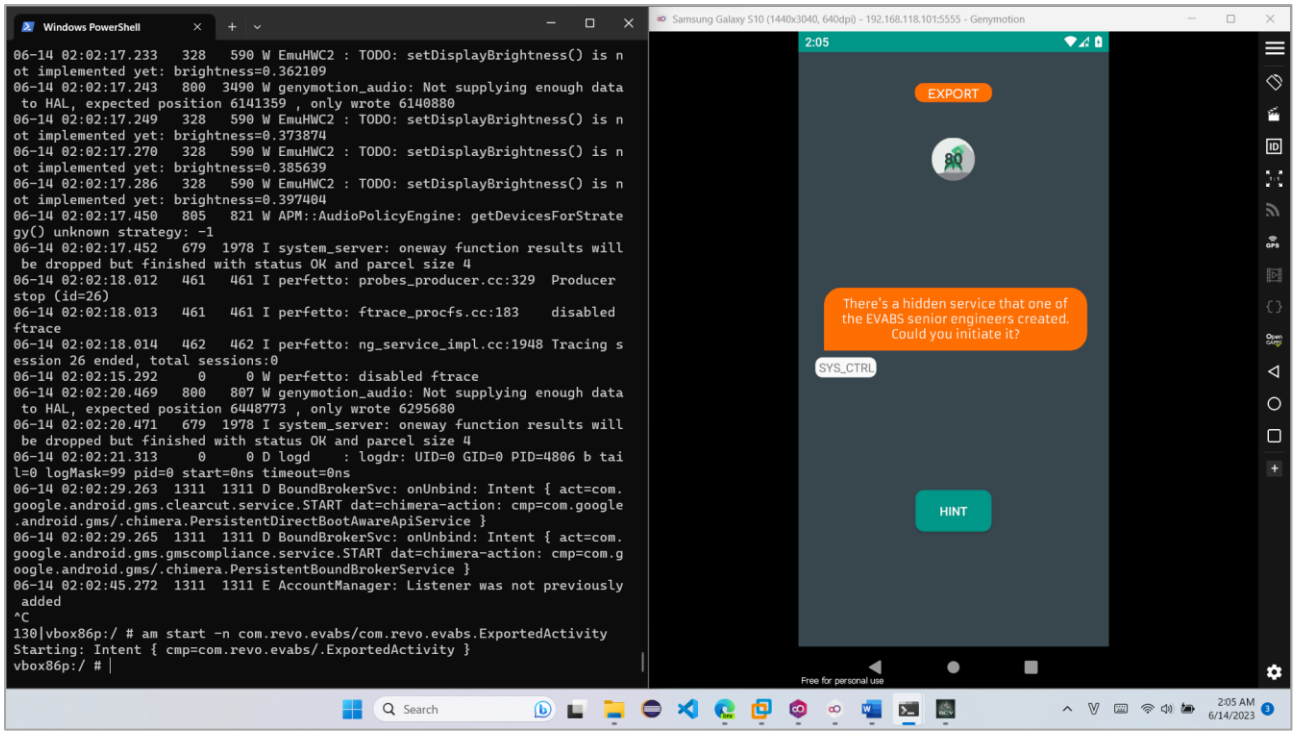
Hình 17: Yêu cầu level 7

Dựa theo gợi ý, chúng ta sẽ tìm tới activity có gán nhãn là exported trong tệp Manifest.xml:



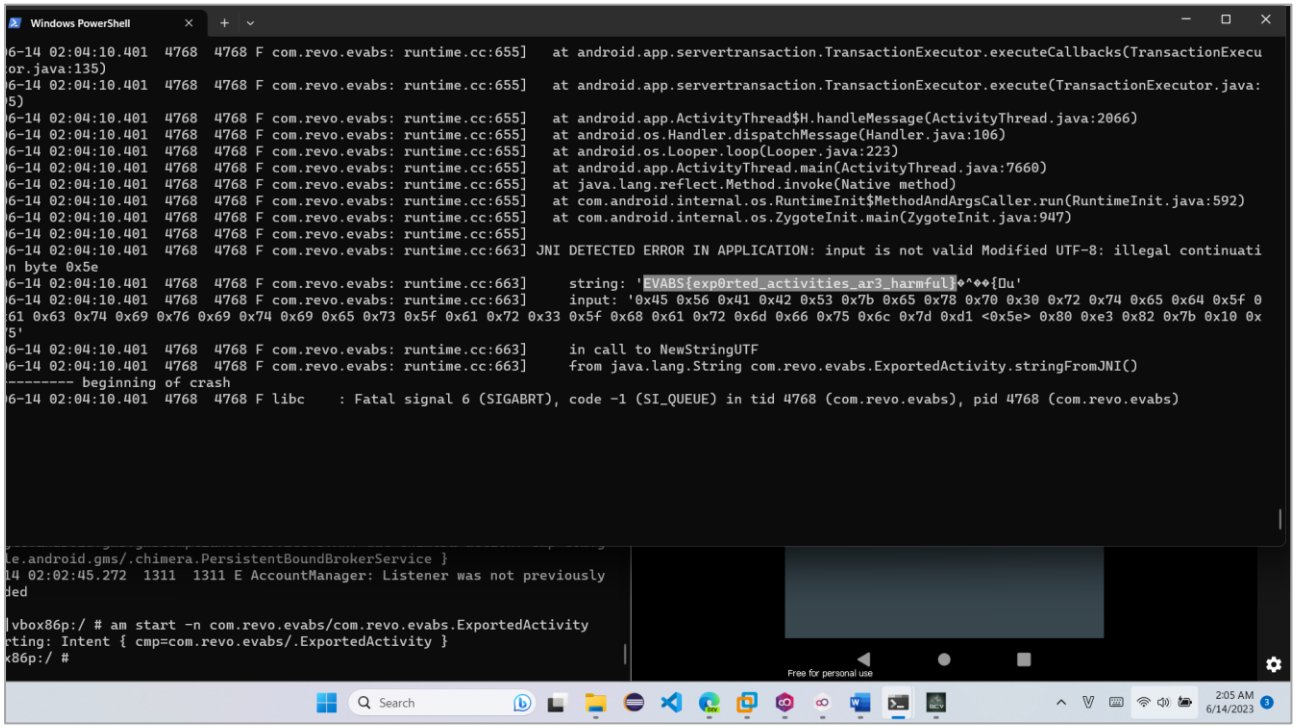
Hình 18: Kết quả tại dòng 4

Do nó được gán nhãn là exported=”true” nên chúng ta sẽ thử Hijacking tới nó:



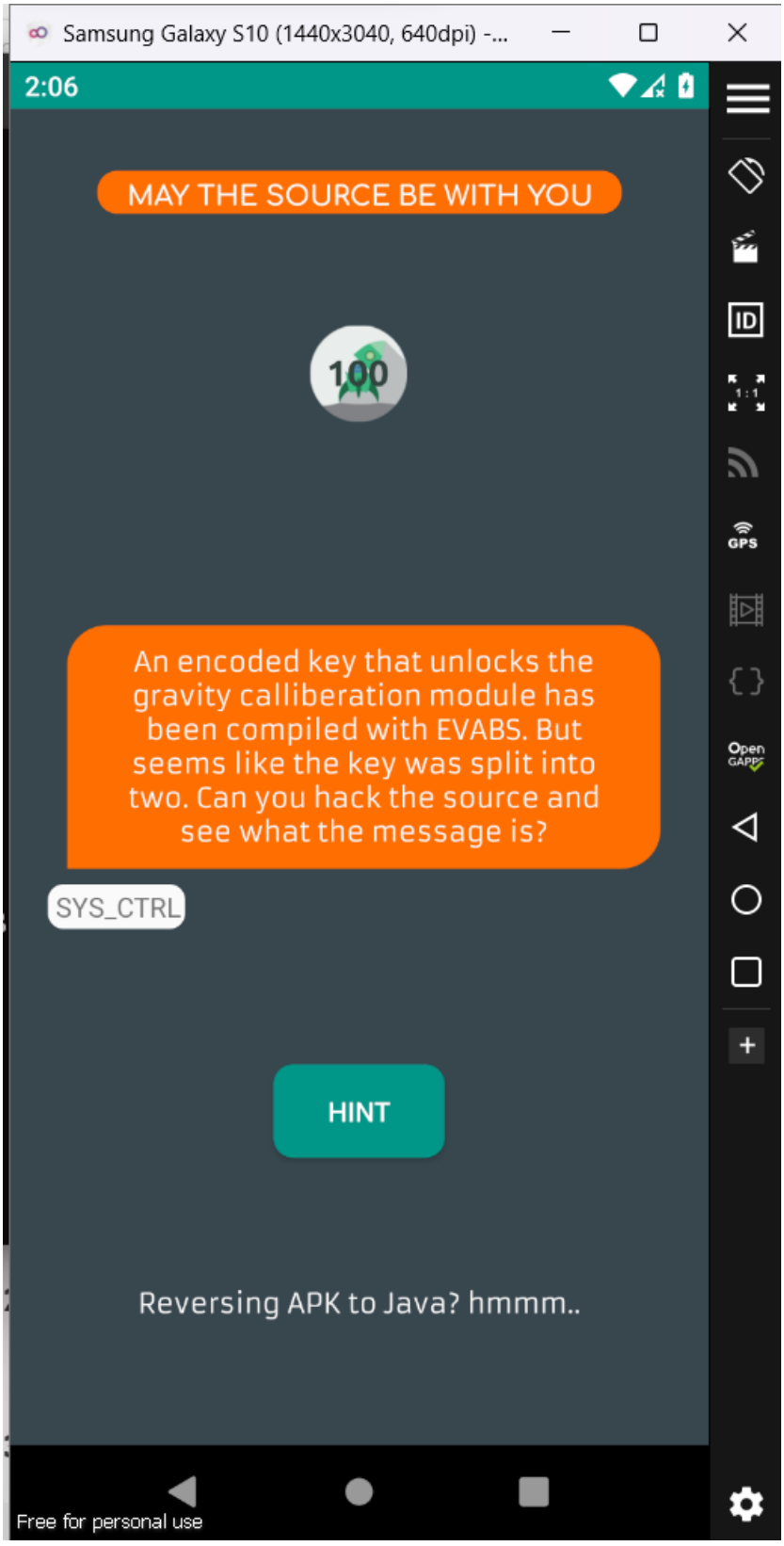
Hình 19: Không xuất hiện gì

Tuy nhiên khi xem lại logcat, chúng ta thấy:



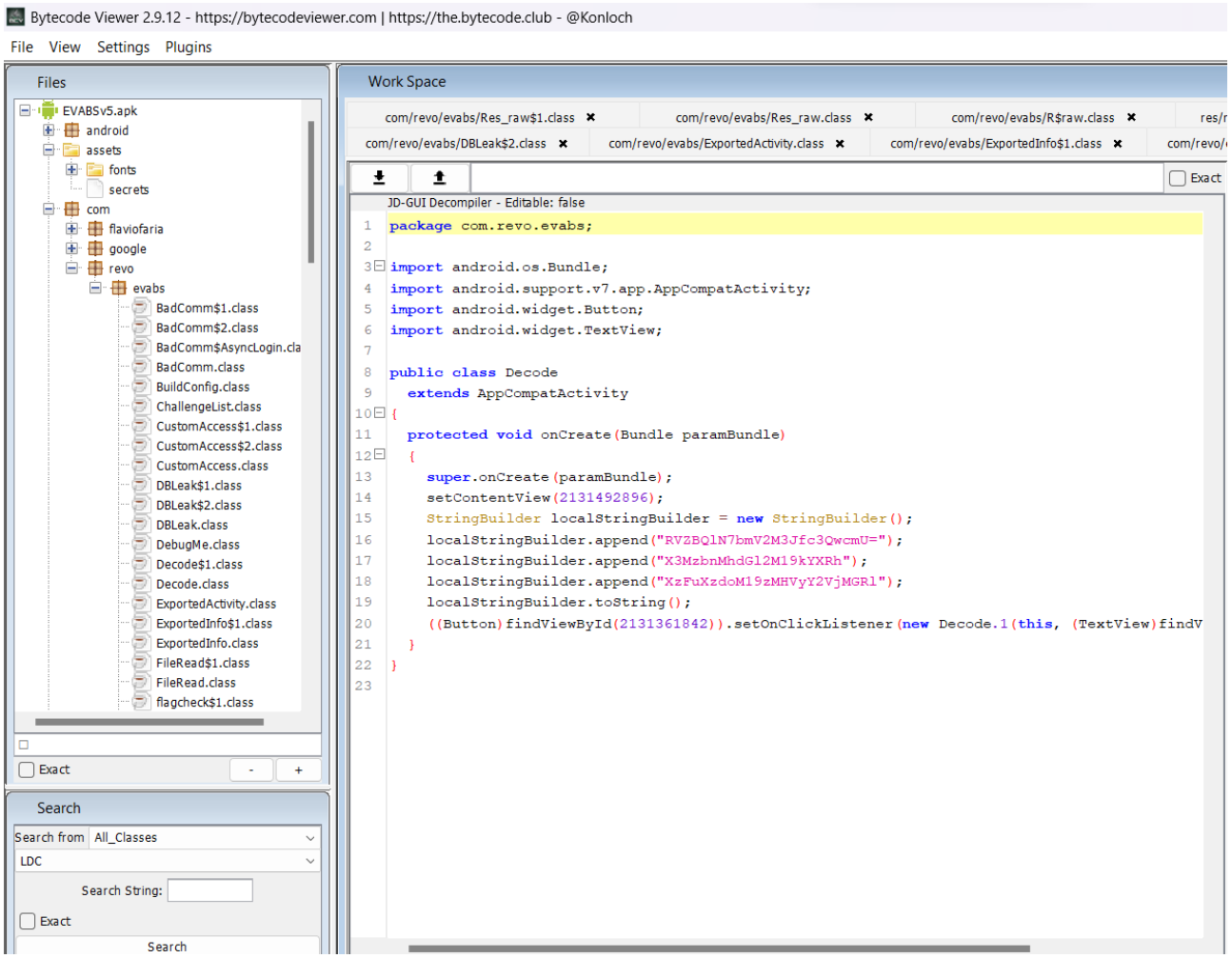
➔ flag: EVABS{exp0rted_activities_ar3_harmful}

1.8. Level 8 - Decode



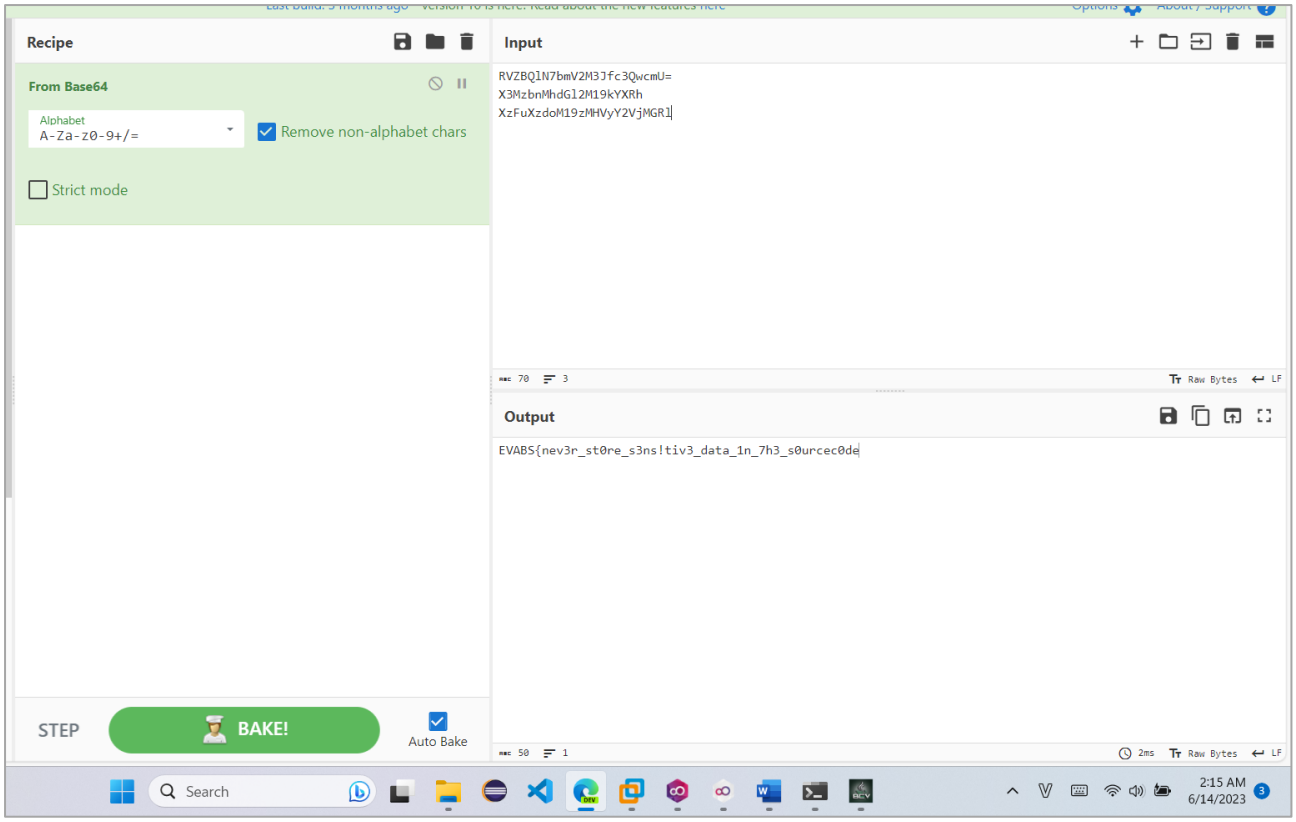
Hình 20: Nội dung level 8

Yêu cầu của level 8 sẽ là đọc code sau đó lấy các chuỗi trong đó giải mã mà ghép chúng lại với nhau sẽ được flag:



Hình 21: Nội dung Decode.class

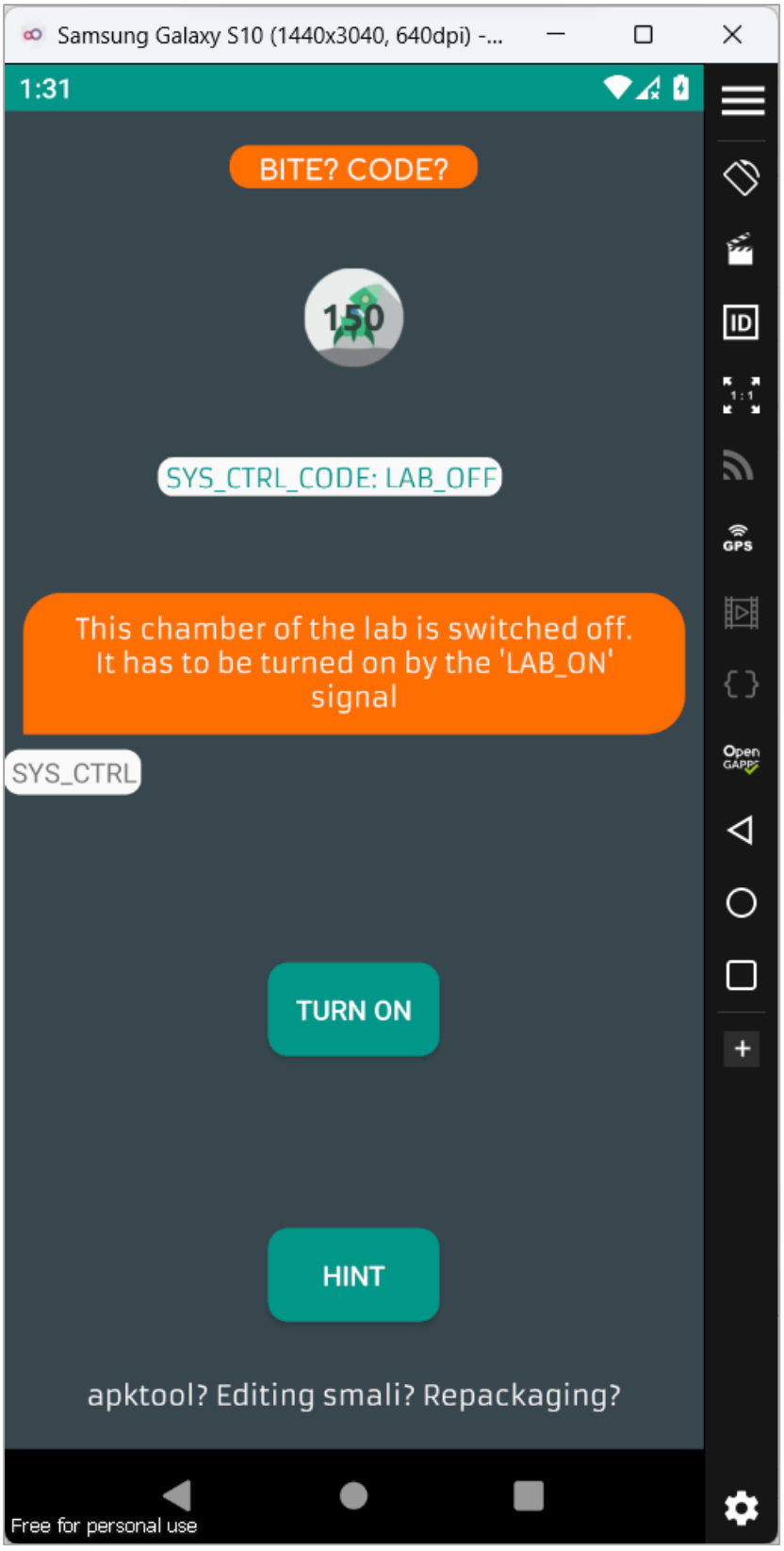
Chuỗi có dấu = nên có thể là base64:



Hình 22: Lấy được flag

➔ EVABS{nev3r_st0re_s3ns!tiv3_data_1n_7h3_s0urcec0de}

1.9. Level 9 – Smali injection



Hình 23: Level 9

Level 9 yêu cầu chúng ta sửa code smali và sửa LAB_OFF thành LAB_ON để lấy flag.

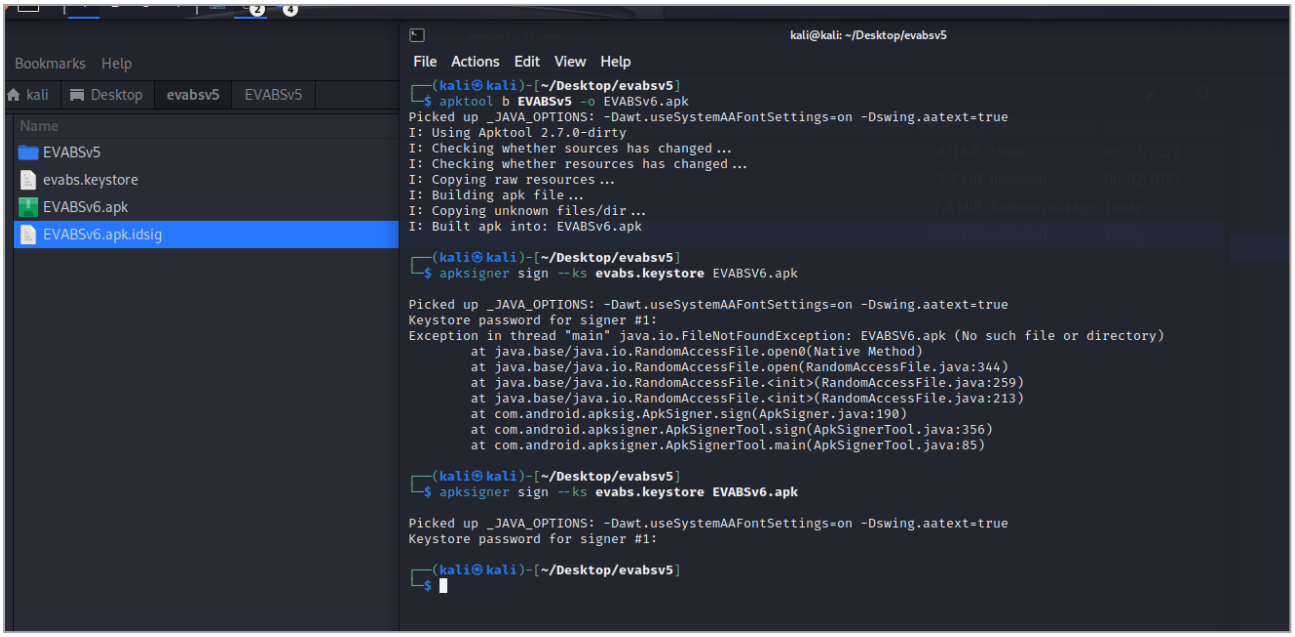
Thực hiện sửa code:

```
23 .method public constructor <init>()V
24     .locals 1
25
26     .line 11
27     invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;→<init>()V
28
29     .line 13
30     const-string v0, "LAB_OFF"
31
32     iput-object v0, p0, Lcom/revo/evabs/SmaliInject;→SIGNAL:Ljava/lang/String;
33
34     return-void
35 .end method
```

```
23 .method public constructor <init>()V
24     .locals 1
25
26     .line 11
27     invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;→<init>()V
28
29     .line 13
30     const-string v0, "  LAB_ON"
31
32     iput-object v0, p0, Lcom/revo/evabs/SmaliInject;→SIGNAL:Ljava/lang/String;
33
34     return-void
35 .end method
```

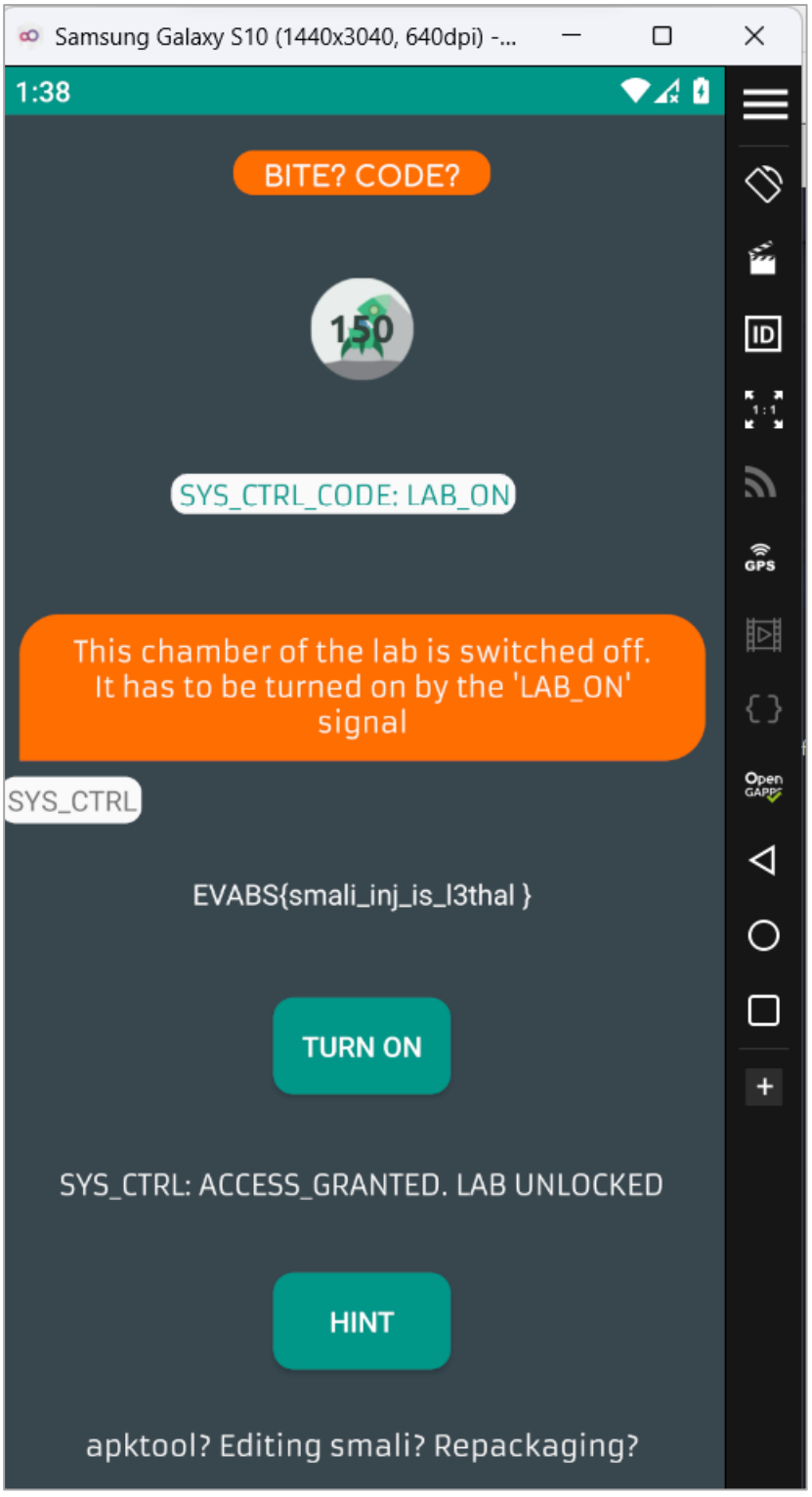
Hình 24: Sửa code smali

Cuối cùng re-build và ký tệp apk:



Hình 25: Rebuild apk

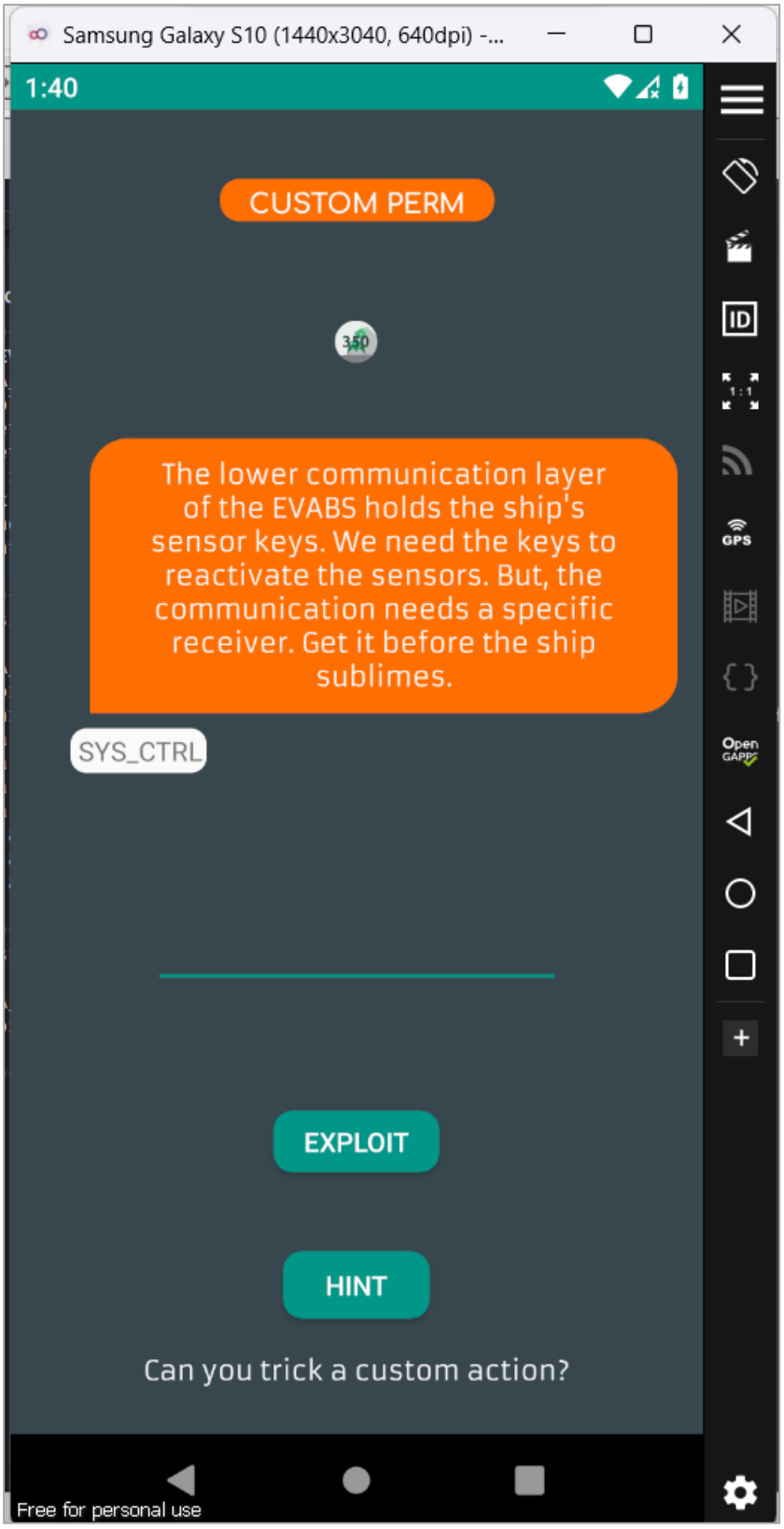
Cài đặt lại ứng dụng và bấm nút Turn on, chúng ta được:



Hình 26: Lấy được flag

1.10. Level 10 – Intercept

1.11. Level 11 – Custom permission



Hình 27: Level 11

Với level 11, chúng ta được yêu cầu nhập vào chuỗi hợp lệ để chuyển sang action khác, theo dõi nội dung của code, chúng ta thấy có điểm đáng chú ý:

com/revo/evabs/ChallengeList.class ✖ com/revo/evabs/CustomAccess.class ✖

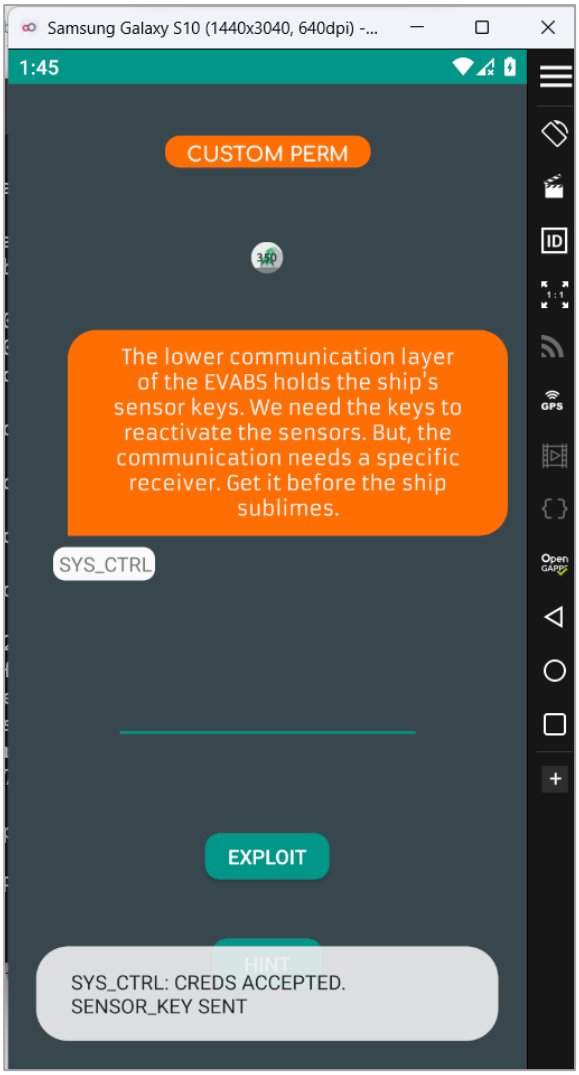
JD-GUI Decompiler

```
7  import android.view.View;
8  import android.widget.Button;
9  import android.widget.EditText;
10 import android.widget.TextView;
11 import android.widget.Toast;
12
13 public class CustomAccess extends AppCompatActivity {
14   public final String EVABS_SENSOR_KEY = "com.revo.evabs.action.SENSOR_KEY";
15
16   static {
17     System.loadLibrary("native-lib");
18   }
19
20   private void GetSensorKey() {
21     String str = ((EditText)findViewById(2131361891)).getText().toString();
22     if ((new String(new char[] {
23           'c', 'u', 's', 't', '0', 'm', '_', 'p', '3', 'r',
24           'm' })).equals(str)) {
25       Toast.makeText((Context)this, "SYS_CTRL: CREDS ACCEPTED. SENSOR_KEY SENT", 1).show();
26       Intent intent = new Intent("com.revo.evabs.action.SENSOR_KEY");
27       StringBuilder stringBuilder = new StringBuilder();
28       stringBuilder.append("EVABS{");
29       stringBuilder.append(stringFromJNI());
30       stringBuilder.append("}");
31       intent.putExtra("android.intent.extra.TEXT", stringBuilder.toString());
32       intent.setType("text/plain");
33       startActivity(intent);
34     } else {
35       Toast.makeText((Context)this, "SYS_CTRL: WRONG_CREDS. SENSOR_KEY LOCKED", 1).show();
36     }
37   }
```

Hình 28: Phần code giúp hiện flag

Code trên đơn giản là nếu chúng ta nhập đúng chuỗi nó sẽ hiển thị ra flag và chuỗi đó chính là “cust0m_p3rm”.

Tuy nhiên, khi nhập chuỗi trên, nó chuyển về chính trang này kèm thông điệp như dưới:



Hình 29: Hiển thị thông điệp

Vậy nên vấn đề là chúng ta cần tác động vào intent để flag hiện ra, code cụ thể như sau:

```
import frida

import sys


def onMessage(message, data):

    print(message)


package = "EVABS"


jscode = """
```

```
Java.perform(function () {
    send("[-] Starting hooks android.content.Intent.putExtra");
    var intent = Java.use("android.content.Intent");
    intent.putExtra.overload("java.lang.String", "java.lang.String").implementation = function(var_1, var_2) {
        send("[+] Flag: " + var_2);
    };

});
"""

process = frida.get_usb_device().attach(package)
script = process.create_script(jscode)
script.on("message", onMessage)
print("[*] Hooking", package)
script.load()
sys.stdin.read()
```

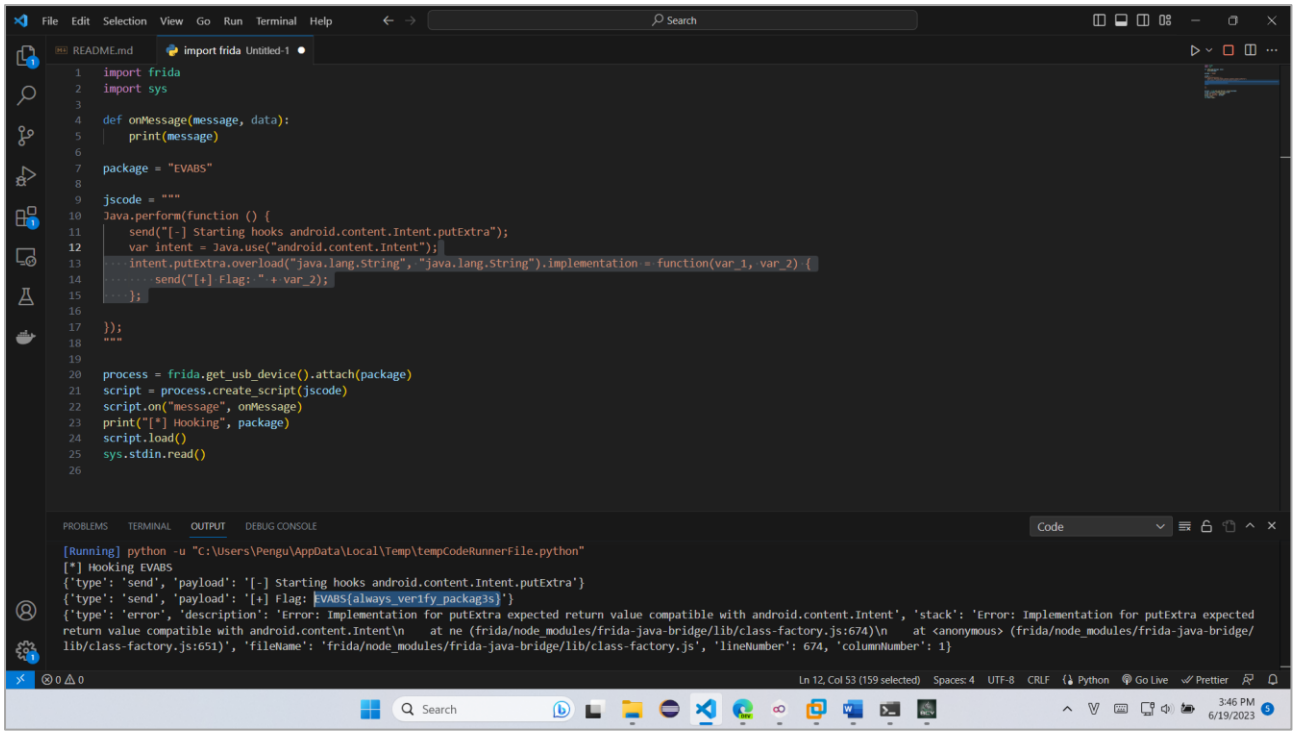
Dòng package chúng ta sẽ xác định bằng lệnh frida-ps -U và kết quả sẽ cho tên cùng với pid:

```
PROBLEMS   TERMINAL   OUTPUT   DEBUG CONSOLE

PS C:\Users\Pengu> frida-ps -U
 PID  Name
----  ------------------------------------------------------------
4830  EVABS
1929  Google Play Store
 578  adbd
1129  android.ext.services
 204  android.hardware.atrace@1.0-service
 806  android.hardware.audio.service
 294  android.hardware.authsecret@1.0-service
 296  android.hardware.camera.provider@2.4-service
 297  android.hardware.cas@1.2-service
 298  android.hardware.configstore@1.1-service
 301  android.hardware.drm@1.0-service
 302  android.hardware.drm@1.3-service.clearkey
 304  android.hardware.gatekeeper@1.0-service.software
 305  android.hardware.gnss@1.0-service
 306  android.hardware.graphics.allocator@2.0-service
 307  android.hardware.graphics.composer@2.3-service
 308  android.hardware.health@2.0-service.genymotion
 326  android.hardware.identity-service.example
 761  android.hardware.input.classifier@1.0-service.default
 206  android.hardware.keymaster@3.0-service
 309  android.hardware.light@2.0-service
 310  android.hardware.memtrack@1.0-service
 311  android.hardware.neuralnetworks@1.3-service-sample-all
 313  android.hardware.neuralnetworks@1.3-service-sample-float-fast
 314  android.hardware.neuralnetworks@1.3-service-sample-float-slow
 316  android.hardware.neuralnetworks@1.3-service-sample-minimal
```

Hình 30: Package sẽ là "EVABS"

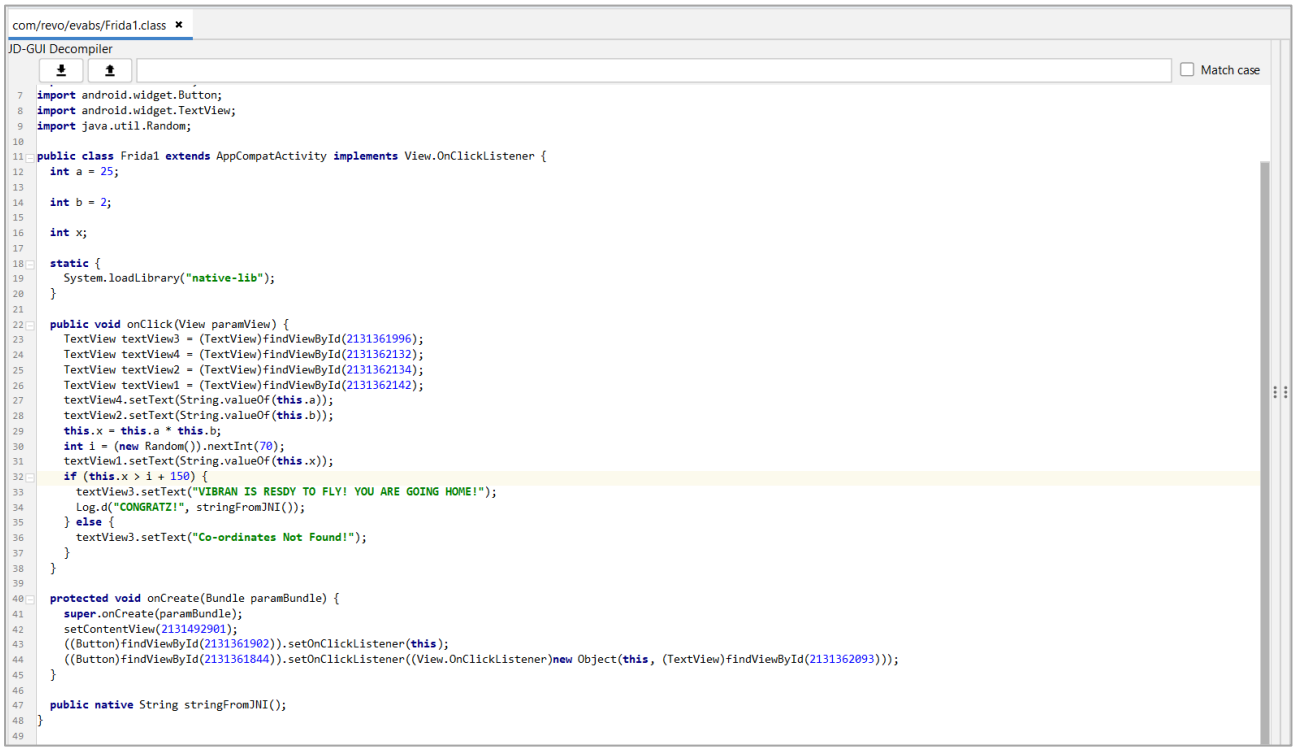
Thực hiện hooking với code trên và thực hiện lại bước nhập chuỗi “cust0m_p3rm” để lấy flag:



Hình 31: Lấy được flag

EVABS{always_ver1fy_packag3s}

1.12. Level 12 - Intrusment



Hình 32: Nội dung code level 12

Chúng ta có thể thấy nội dung code sẽ có 2 biến a và b với kết quả được gán từ trước và x sẽ là kết quả của a nhân b, sau đó giá trị biến i sẽ được random với Random().nextInt(70) – giá trị I sẽ nằm trong khoản 0 đến 69.

Sau đó, kết quả sẽ được đưa vào lệnh if để so sánh x có lớn hơn i+150 hay không hay cũng chính so sánh a*b với i+150. Nếu lớn hơn, thông điện sẽ được hiển thị trong log.

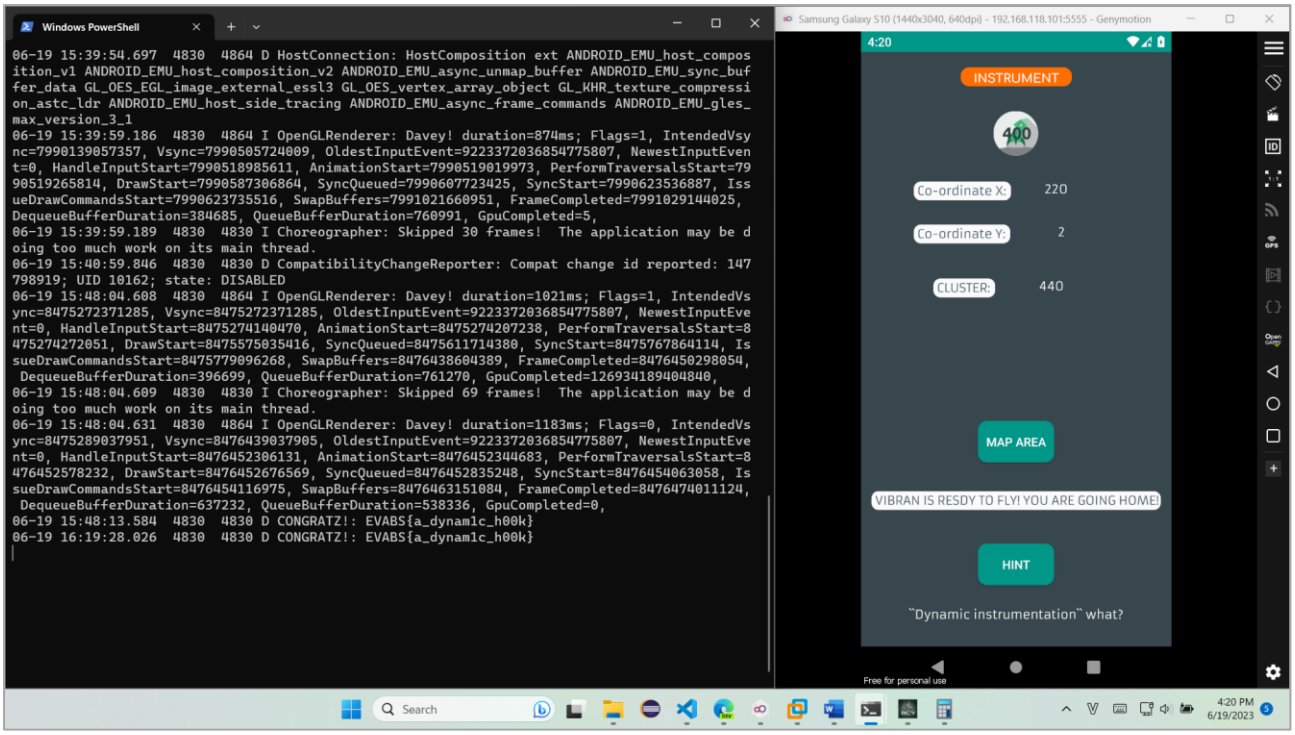
Do a và b trước đó chỉ gán là 25 và 2 nên kết quả chỉ là 50 không thể lớn hơn i+150 được, vậy nên chúng ta sẽ thay đổi giá trị của a hoặc b sao cho kết quả luôn lớn hơn khi kết quả i được random + 150.

```
30 .method public constructor <init>()V
31     .locals 1
32
33     .line 13
34     invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;→<init>()V
35
36     .line 15
37     const/16 v0, 0×DC
38
39     iput v0, p0, Lcom/revo/evabs/Frida1;→a:I
40
41     const/4 v0, 0×2
42
43     iput v0, p0, Lcom/revo/evabs/Frida1;→b:I
44
45     return-void
46 .end method
```

Hình 33: Sửa giá trị a

Trong hình, chúng ta thay đổi giá trị a thành 0xDC tương ứng với 220 trong thập phân để giá trị sau khi nhân với b chắc chắn sẽ lớn hơn i+150 (max sẽ là 69+150 = 219).

Đóng gói và kí, chúng ta được:

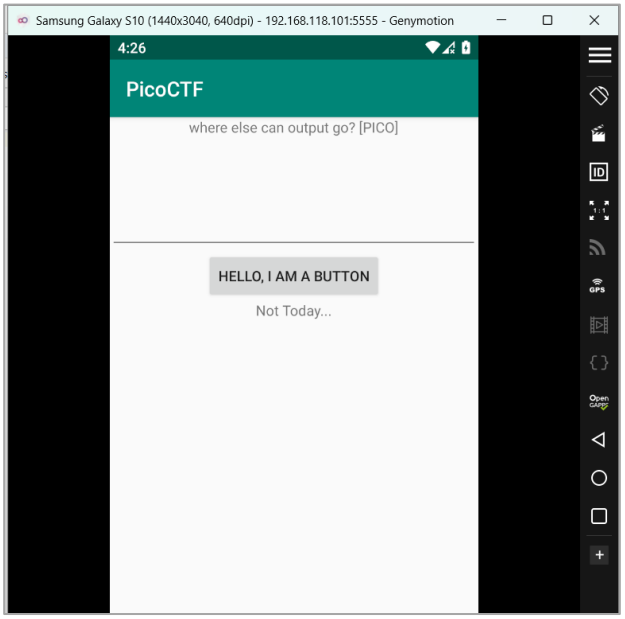


Hình 34: Tìm thấy flag

EVABS{a_dynam1c_h00k}

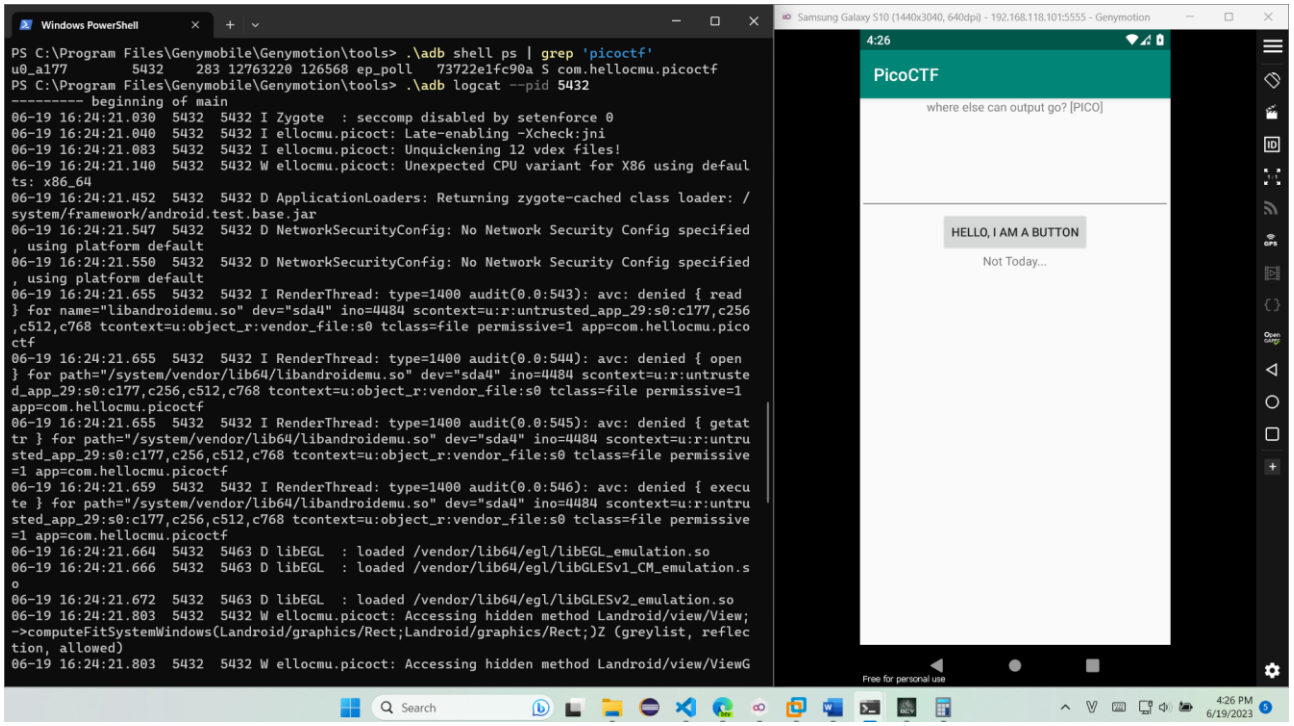
2. Challenges 2

2.1. One.apk



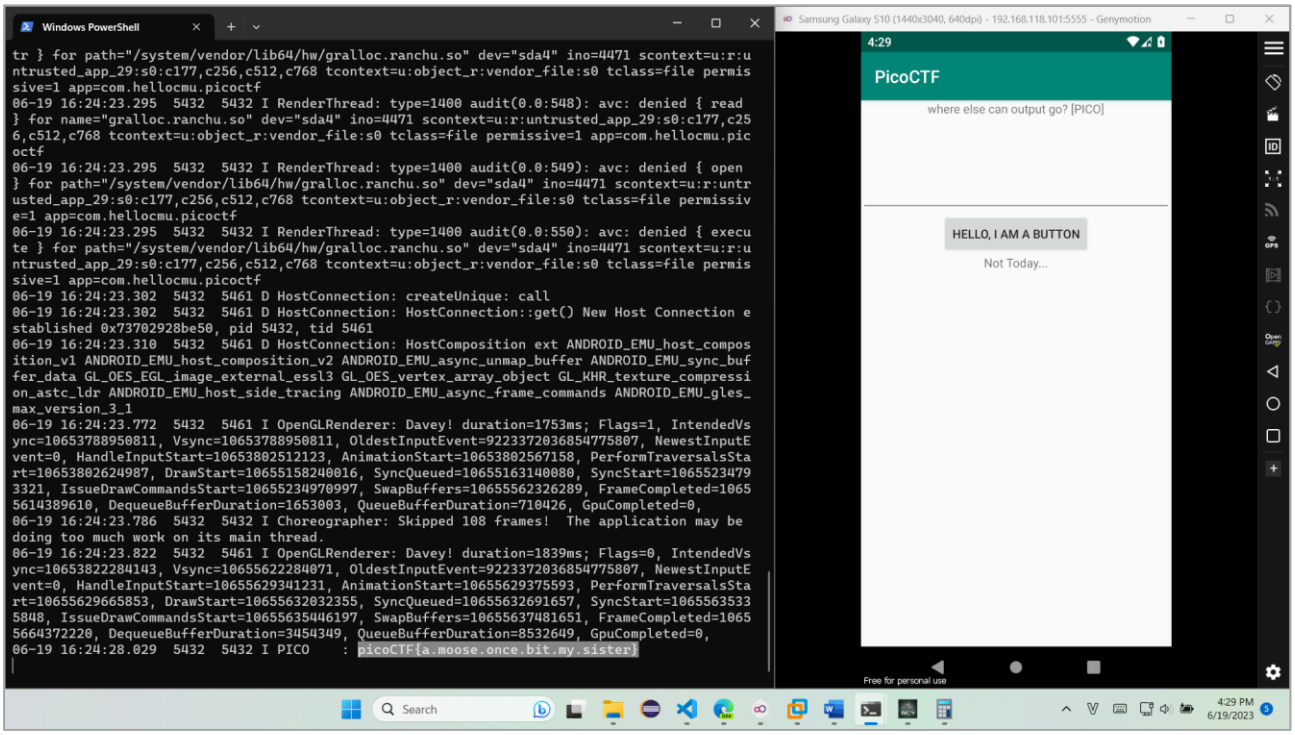
Hình 35: Level 1

Xem log của ứng dụng:



Hình 36: Xem log của app

Sau đó chúng ta bấm nút HELLO, I AM A BUTTON, flag sẽ được hiển thị:



Hình 37: Lấy được flag

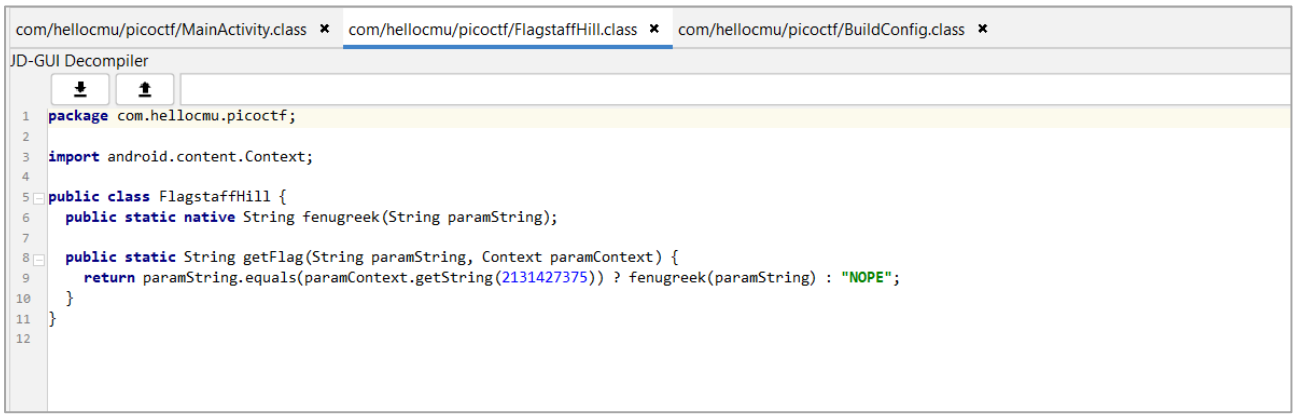
picoCTF{a.moose.once.bit.my.sister}

2.2. two.apk

Thực hiện mở tệp two.apk bằng BCV, chúng ta thấy code nằm trong phần MainActivity dùng để lấy flag như sau:

Hình 38: Gọi tới getFlag

Hàm getFlag có nội dung như dưới:

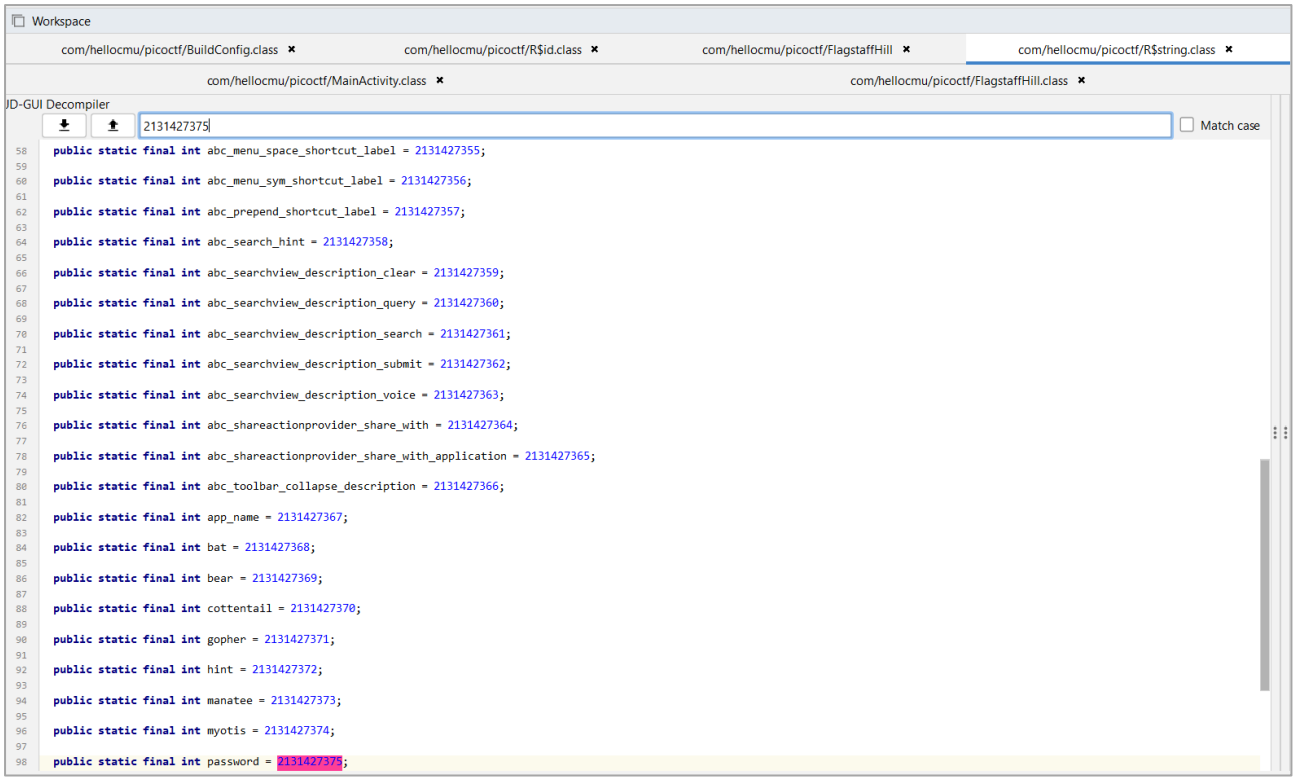


Hình 39: Nội dung hàm getFlag

Theo đó, hàm getFlag sẽ trả về kết quả dựa trên chuỗi nhập vào, nếu chuỗi nhập vào trùng với chuỗi giá trị chuỗi được gán cho "**2132427375**", thì flag sẽ được hiển thị, ngược lại nó sẽ hiển thị NOPE:

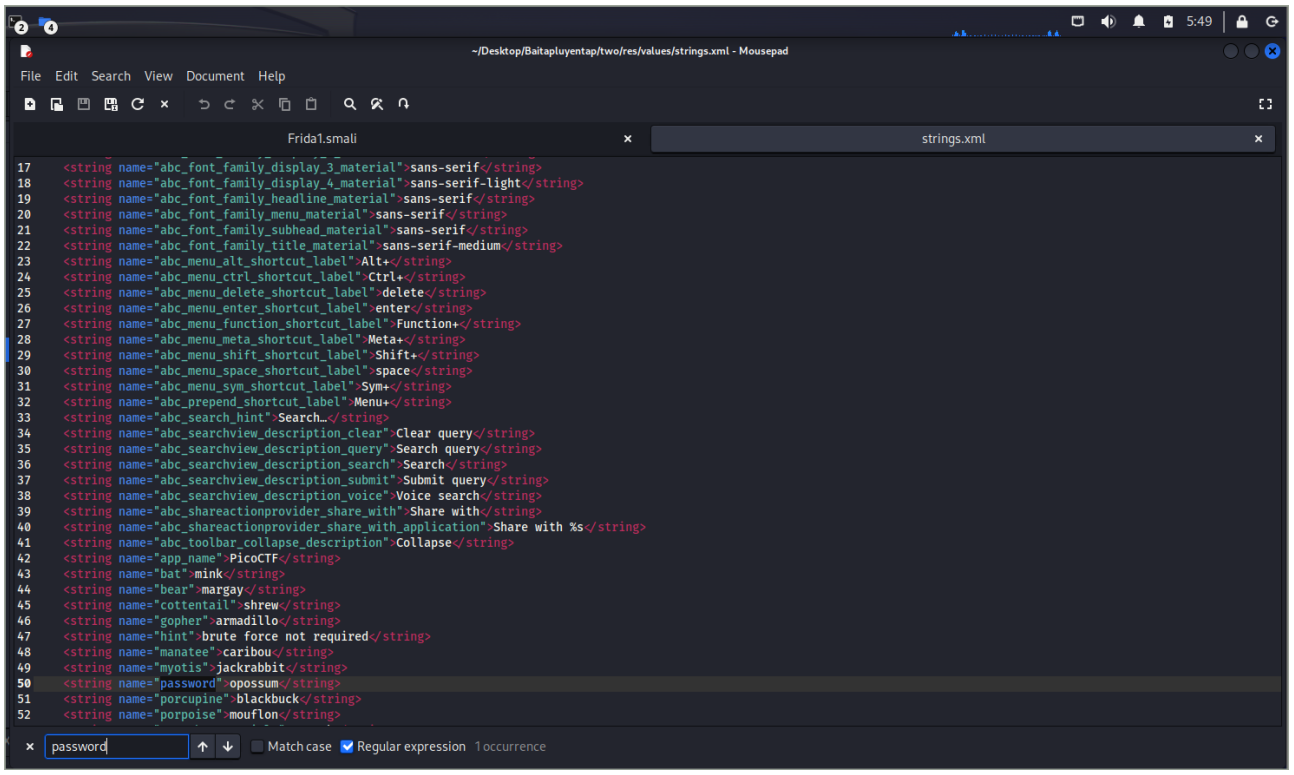
Hình 40: Chuỗi nhập vào không đúng

Dữ liệu liên quan tới các chuỗi sẽ được tìm thấy trong lớp R$string, chúng ta thấy nó sẽ là **password**:



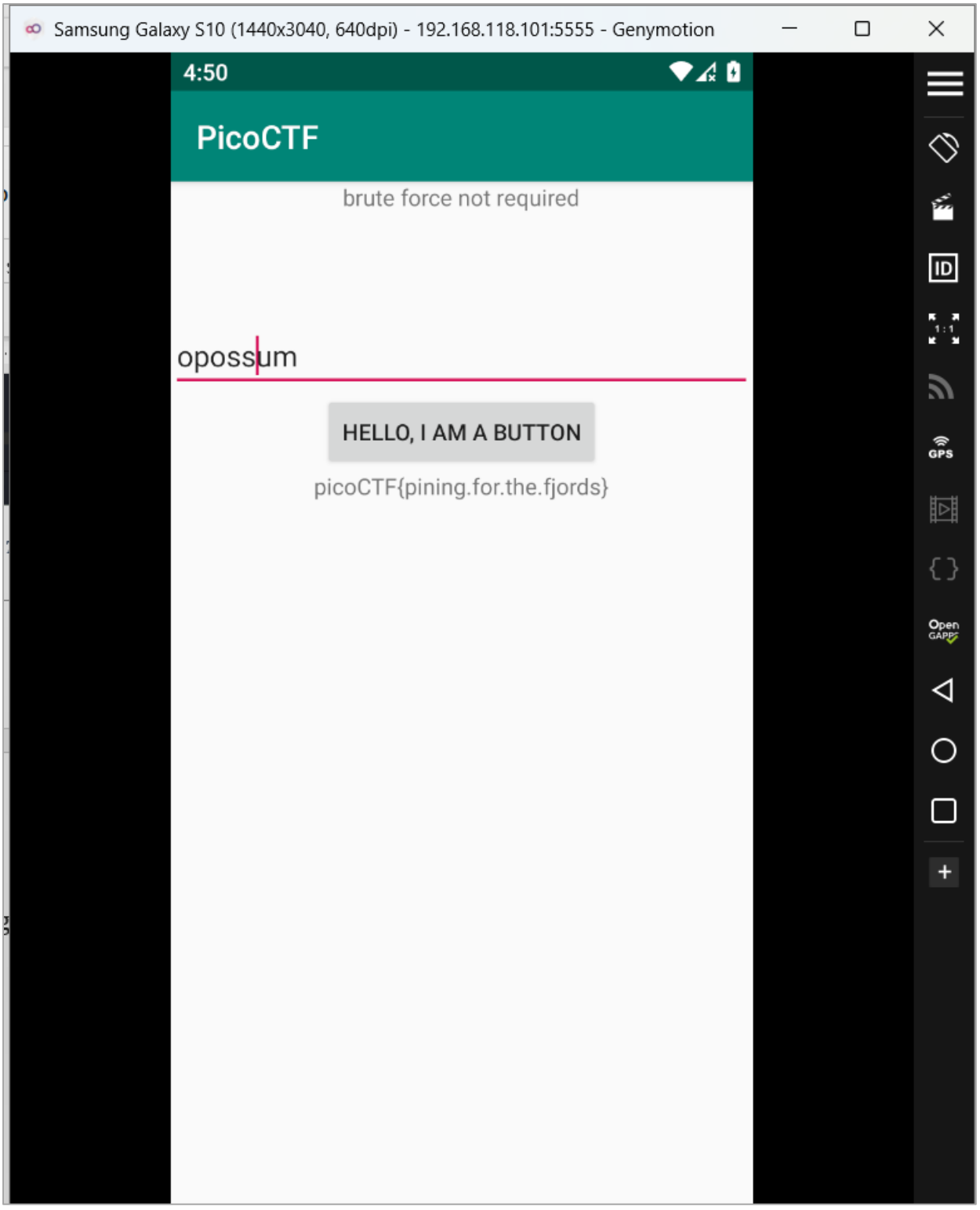
Hình 41: Password được gán với 2131427375

Tiếp đó, chúng ta sẽ tìm giá trị chuỗi được gán cho password trong tệp res/values/string.xml:



Hình 42: Tìm thấy chuỗi cần nhập vào

Chuỗi chúng ta sẽ nhập vào ứng dụng là opossum:



Hình 43: Tìm thấy flag

picoCTF{pining.for.the.fjords}

2.3. three.apk

Nội dung code getFlag trong level 3:

Hình 44: Hàm getFlag

Chúng ta cần thực hiện tính toán các giá trị j, m, I, k để từ đó đưa vào kết quả lấy các chuỗi với vị trí tương ứng:

```
return paramString.equals("".concat(arrayOfString[k]).concat(".").concat(arrayOfString[m]).concat(".").
concat(arrayOfString[j]).concat(".").concat(arrayOfString[k + j - m]).concat(".").concat(arrayOfString[3]).
concat(".").concat(arrayOfString[i])) ? sesame(paramString) : "NOPE";
```

Qua tính toán, chúng ta được như sau:

j = 0

m = 1

i = 1 + 1 - 0 = 2

k = 3 + 2 = 5

k+j-m = 5 + 0 - 1 = 4

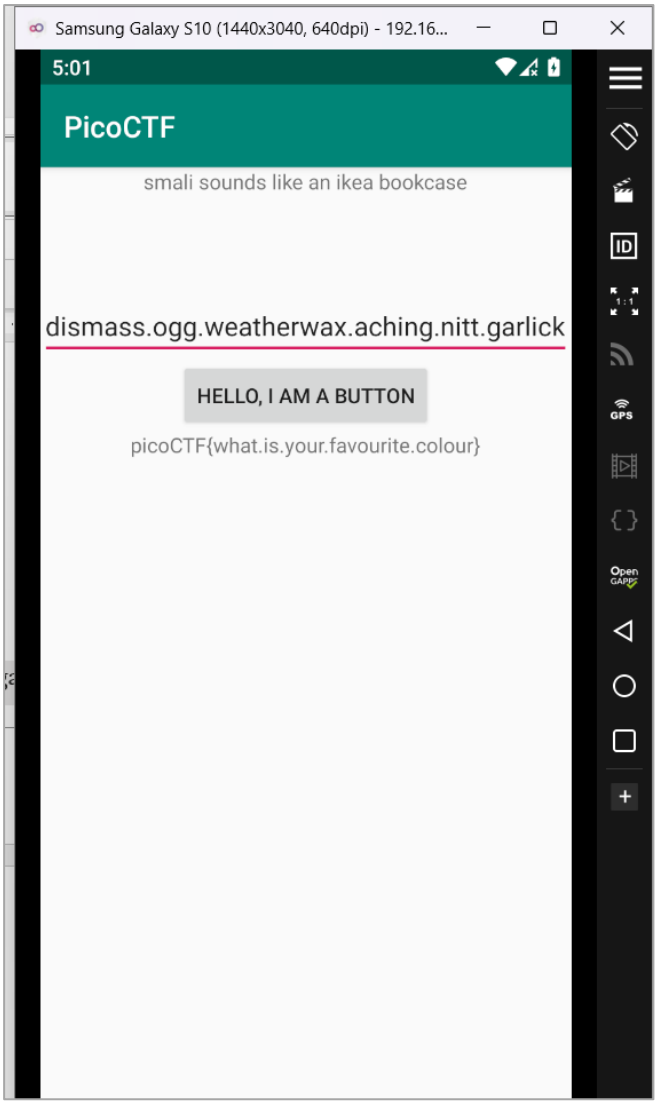
Thứ tự các giá trị được gọi tới trong dòng code trên sẽ là

5.1.0.4.3.2

Tương ứng với:

```
dismass.ogg.weatherwax.aching.nitt.garlick
```

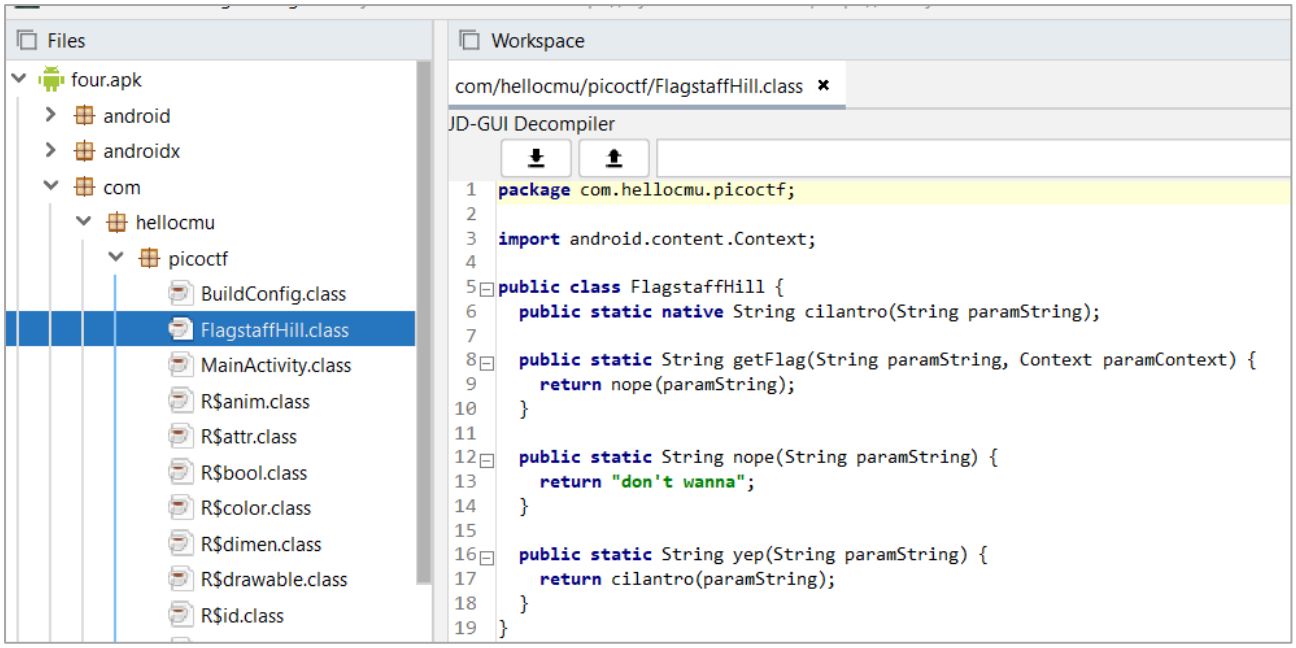
Nhập chuỗi trên vào ứng dụng, chúng ta có flag:



picoCTF{what.is.your.favourite.colour}

2.4. four.apk

Nội dung hàm getFlag của level 4:



Hình 45: Nội dung hàm getFlag

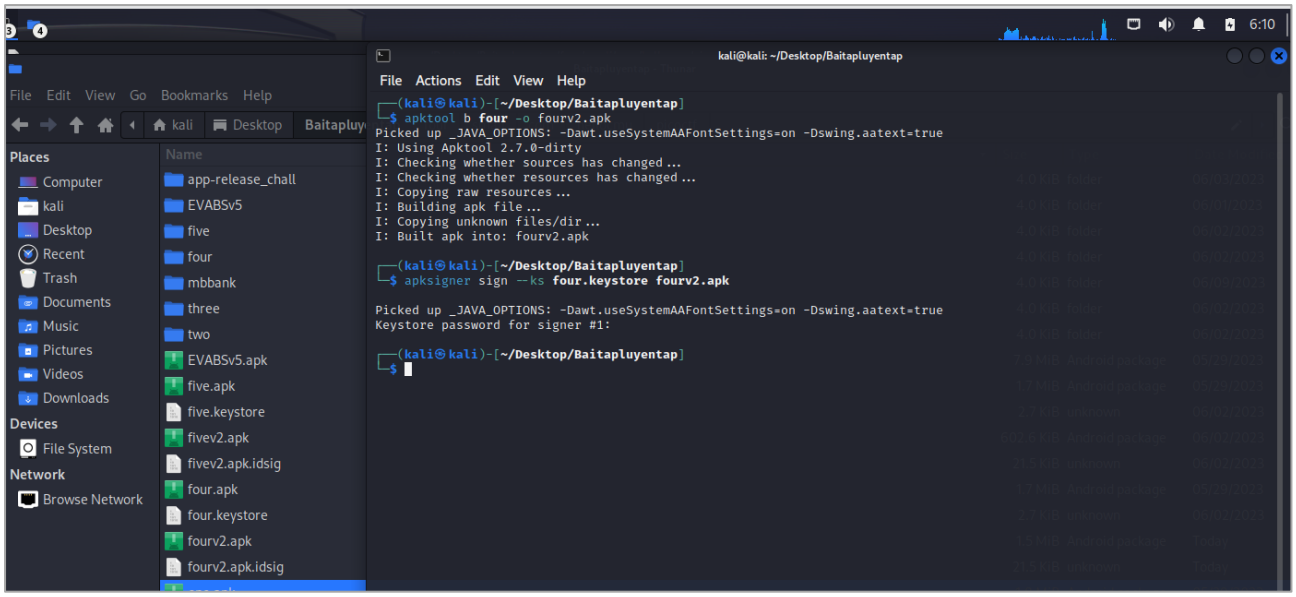
Ta thấy hàm getFlag luôn được return về hàm nope(), vậy nên chúng ta sẽ thử thay đổi để nó gọi tới hàm yep:

```
19 .method public static getFlag(Ljava/lang/String;Landroid/content/Context;)Ljava/lang/String;
20     .locals 1
21     .param p0, "input"    # Ljava/lang/String;
22     .param p1, "ctx"    # Landroid/content/Context;
23
24     .line 19
25     invoke-static {p0}, Lcom/hellocmu/picoctf/FlagstaffHill;→nope(Ljava/lang/String;)Ljava/lang/String;
26
27     move-result-object v0
28
29     .line 20
30     .local v0, "flag":Ljava/lang/String;
31     return-object v0
32 .end method
```

```
19 .method public static getFlag(Ljava/lang/String;Landroid/content/Context;)Ljava/lang/String;
20     .locals 1
21     .param p0, "input"    # Ljava/lang/String;
22     .param p1, "ctx"    # Landroid/content/Context;
23
24     .line 19
25     invoke-static {p0}, Lcom/hellocmu/picoctf/FlagstaffHill;→yep(Ljava/lang/String;)Ljava/lang/String;
26
27     move-result-object v0
28
29     .line 20
30     .local v0, "flag":Ljava/lang/String;
31     return-object v0
32 .end method
```

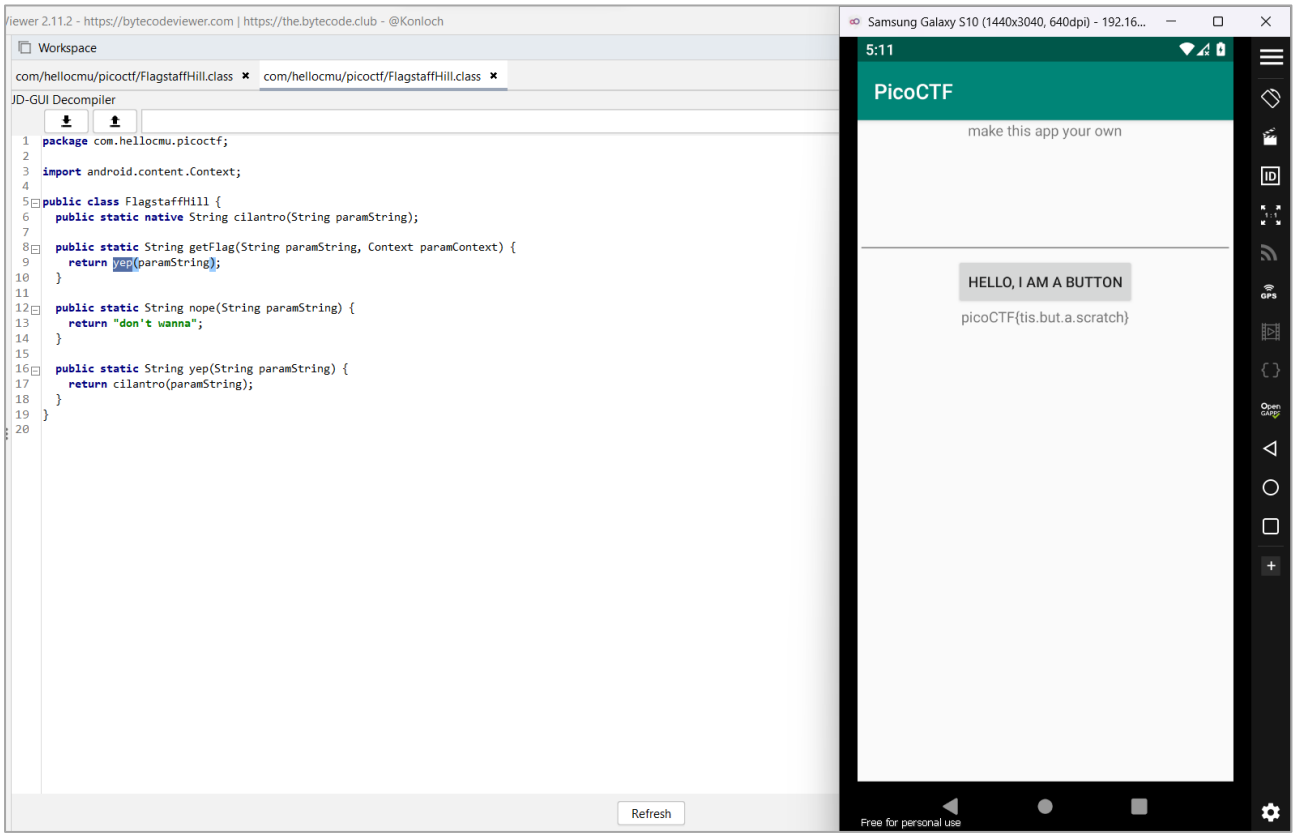
Hình 46: Thay đổi code trong hàm getFlag

Sửa dòng 25 từ “nope” thành “yep”, sau đó rebuild và kí:



Hình 47: Repatch apk

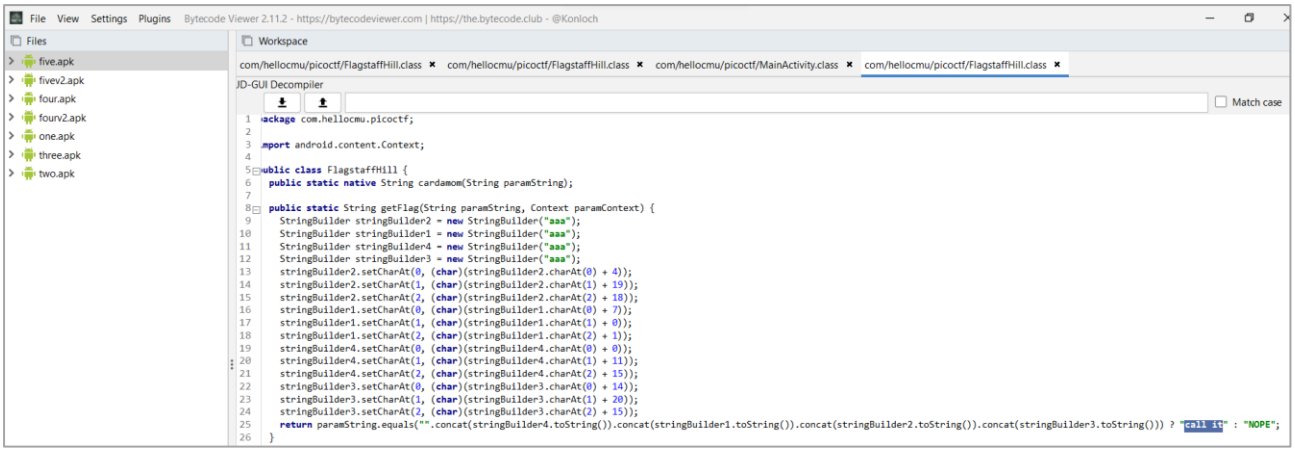
Kết quả chúng ta được:



Hình 48: Lấy được flag

picoCTF{tis.but.a.scratch}

2.5. five.apk



Hình 49: Nội dung hàm getFlag

Hàm getFlag của câu 5 cũng tương tự câu 3 là chúng ta cần tính toán, nhưng câu 5 khi tìm được đúng chuỗi nó chỉ trả về thông điệp là “call it”, vậy nên chúng ta cần thực hiện sửa code sao cho nó gọi tới hàm cardamom():

```
228     invoke-virtual {p0, v4}, Ljava/lang/String;→equals(Ljava/lang/Object;)Z
229
230     move-result v5
231
232     if-eqz v5, :cond_0
233
234     const-string v5, "call it"
235
236     return-object v5
237
```

```
229
230     move-result v5
231
232     if-eqz v5, :cond_0
233
234     invoke-static {p0}, Lcom/hellocmu/picoctf/FlagstaffHill;→sesame(Ljava/lang/String;)Ljava/lang/String;
235
236     move-result-object v5
237
238     return-object v5
239
```

Hình 50: Thay đổi gọi về cardamom của lệnh if

Thực hiện patch app và kí:

```
┌──(kali㉿kali)-[~/Desktop/Baitapluyentap]
└─$ apktool b five -o fivev2.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.7.0-dirty
I: Checking whether sources has changed ...
I: Smaling smali folder into classes.dex ...
I: Checking whether resources has changed ...
I: Copying raw resources ...
I: Building apk file ...
I: Copying unknown files/dir ...
I: Built apk into: fivev2.apk

┌──(kali㉿kali)-[~/Desktop/Baitapluyentap]
└─$ apksigner sign --ks five.keystore fivev2.apk

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Keystore password for signer #1:

┌──(kali㉿kali)-[~/Desktop/Baitapluyentap]
└─$
```

Hình 51: Repatch apk

Sau khi cài đặt lại ứng dụng, chúng ta tiếp tục với bước tính toán:

com/hellocmu/picoctf/FlagstaffHill.class × com/hellocmu/picoctf/FlagstaffHill.class × com/hellocmu/picoctf/FlagstaffHill.class ×

JD-GUI Decompiler

Match case

```
1  package com.hellocmu.picoctf;
2
3  import android.content.Context;
4
5  public class FlagstaffHill {
6    public static native String cardamom(String paramString);
7
8    public static String getFlag(String paramString, Context paramContext) {
9      StringBuilder stringBuilder1 = new StringBuilder("aaa");
10     StringBuilder stringBuilder2 = new StringBuilder("aaa");
11     StringBuilder stringBuilder3 = new StringBuilder("aaa");
12     StringBuilder stringBuilder4 = new StringBuilder("aaa");
13     stringBuilder1.setCharAt(0, (char)(stringBuilder1.charAt(0) + 4));
14     stringBuilder1.setCharAt(1, (char)(stringBuilder1.charAt(1) + 19));
15     stringBuilder1.setCharAt(2, (char)(stringBuilder1.charAt(2) + 18));
16     stringBuilder2.setCharAt(0, (char)(stringBuilder2.charAt(0) + 7));
17     stringBuilder2.setCharAt(1, (char)(stringBuilder2.charAt(1) + 0));
18     stringBuilder2.setCharAt(2, (char)(stringBuilder2.charAt(2) + 1));
19     stringBuilder3.setCharAt(0, (char)(stringBuilder3.charAt(0) + 0));
20     stringBuilder3.setCharAt(1, (char)(stringBuilder3.charAt(1) + 11));
21     stringBuilder3.setCharAt(2, (char)(stringBuilder3.charAt(2) + 15));
22     stringBuilder4.setCharAt(0, (char)(stringBuilder4.charAt(0) + 14));
23     stringBuilder4.setCharAt(1, (char)(stringBuilder4.charAt(1) + 20));
24     stringBuilder4.setCharAt(2, (char)(stringBuilder4.charAt(2) + 15));
25     return paramString.equals("".concat(stringBuilder3.toString()).concat(stringBuilder2.toString()).concat(stringBuilder1.toString()).concat(stringBuilder4.toString())) ? cardamom(paramStri
26   }
27 }
28
```

Hình 52: Code sau khi đổi

stringBuilder1:

- Ký tự đầu tiên (index 0): 'a' + 4 = 'e'
- Ký tự thứ hai (index 1): 'a' + 19 = 't'
- Ký tự thứ ba (index 2): 'a' + 18 = 's'

stringBuilder2:

- Ký tự đầu tiên (index 0): 'a' + 7 = 'h'
- Ký tự thứ hai (index 1): 'a' + 0 = 'a'
- Ký tự thứ ba (index 2): 'a' + 1 = 'b'

stringBuilder3:

- Ký tự đầu tiên (index 0): a' + 0 = 'a'
- Ký tự thứ hai (index 1): 'a' + 11 = 'l'
- Ký tự thứ ba (index 2): 'a' + 15 = 'p'

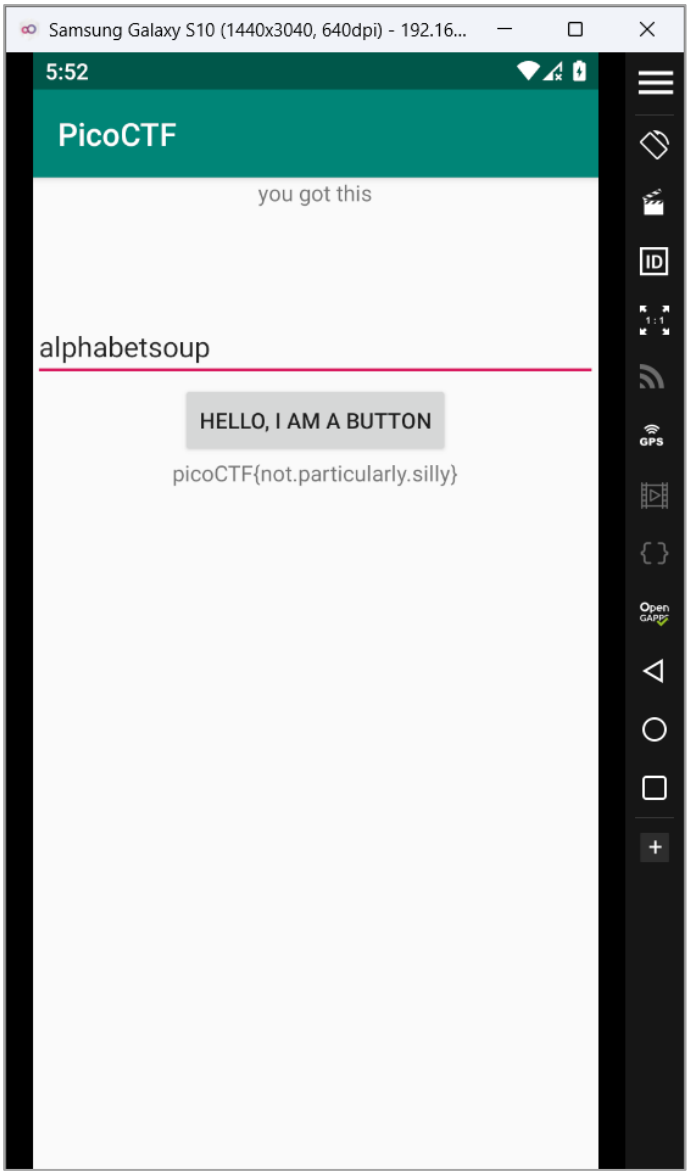
stringBuilder4:

- Ký tự đầu tiên (index 0): 'a' + 14 = 'o'
- Ký tự thứ hai (index 1): 'a' + 20 = 'u'
- Ký tự thứ ba (index 2): 'a' + 15 = 'p'

Nối các chuỗi trên theo thứ tự 3-2-1-4, chúng ta được keyword:

“**alphabetsoup**”

Nhập keyword và lấy flag:



Hình 53: Lấy được flag

picoCTF{not.particulary.silly}