

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

----- ❦ ★ ❦ -----



BÁO CÁO KẾT QUẢ THỰC HÀNH
BẢO MẬT WEB VÀ ỨNG DỤNG
CHALLENGE 04 – PENTESTING ANDROID
APPLICATIONS

Giảng viên giảng dạy: Nghi Hoàng Khoa

Nhóm sinh viên thực hiện:

- | | |
|-----------------------|----------|
| 1. Phạm Khôi Nguyên | 18520114 |
| 2. Vũ Tiến Đạt | 18520264 |
| 2. Phan Thanh Hải | 18520705 |
| 4. Nguyễn Quang Huy | 18520846 |
| 3. Nguyễn Lý Đình Nhì | 18521205 |

TP. HỒ CHÍ MINH, 05/2021

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

----- ❦ ★ ❦ -----



BÁO CÁO KẾT QUẢ THỰC HÀNH

BẢO MẬT WEB VÀ ỨNG DỤNG

CHALLENGE 04 – PENTESTING ANDROID

APPLICATIONS

Giảng viên giảng dạy: Nghi Hoàng Khoa

Nhóm sinh viên thực hiện:

- | | |
|-----------------------|----------|
| 1. Phạm Khôi Nguyên | 18520114 |
| 2. Vũ Tiến Đạt | 18520264 |
| 2. Phan Thanh Hải | 18520705 |
| 4. Nguyễn Quang Huy | 18520846 |
| 3. Nguyễn Lý Đình Nhì | 18521205 |

TP. HỒ CHÍ MINH, 05/2021

MỤC LỤC

LỜI NÓI ĐẦU	1
CHƯƠNG 1. EVABS.....	2
1. Level 1: Debug Me	2
2. Level 2: File Access	2
3. Level 3: Strings.....	4
4. Level 4: Resources.....	5
5. Level 5: Shares and Prefs	6
6. Level 6: DB Leak	7
7. Level 7: Export	8
8. Level 8: Decode.....	9
9. Level 9: Smali Injection	11
10. Level 10: Interception.....	14
11. Level 11: Custom Access	15
12. Level 12: Instrument.....	18
CHƯƠNG 2. DROID.....	23
1. Nhật ký droid đã đi đâu. Bạn có thể tìm thấy tại: <code>one.apk</code>	23
2. Tìm kiếm và lấy flag. Bạn có thể tìm thấy tại: <code>two.apk</code>	25
3. Tìm kiếm và lấy flag. Bạn có thể tìm thấy tại: <code>three.apk</code>	29
4. Dịch ngược, vá lại tập tin và lấy cờ. Bạn có thể tìm thấy tại: <code>four.apk</code>	33
5. Dịch ngược, vá lại tập tin và lấy cờ. Bạn có thể tìm thấy tại: <code>five.apk</code>	37

LỜI NÓI ĐẦU

Đây là phần bài làm của nhóm cho bài tập challenge buổi 04.

Cảm ơn anh Nghi Hoàng Khoa trong thời gian vừa qua chịu khó trả lời những câu hỏi, những thắc mắc của nhóm.

CHƯƠNG 1. EVABS

1. Level 1: Debug Me

Giao diện khi ta vào level 1 (Debug Me) của app sẽ có dạng như sau:



Với thông báo này, ta biết được rằng secret key mà ta cần tìm đã được log lại, vì vậy lúc này ta chỉ cần đọc log của thiết bị thì sẽ có được giá trị cần tìm.

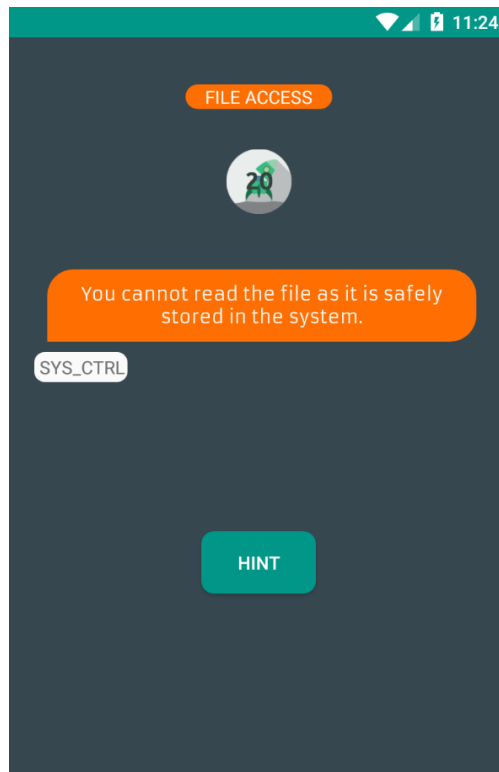
Ta vào command shell gõ lệnh `adb logcat`. Sau đó, ta vào app và nhấn nút **LOG THE KEY**. Ta quan sát thông tin hiển thị trên màn hình command shell sau khi nhấn nút:

```
05-08 11:18:42.335 163 163 I logd.reader: type=1400 audit(0.0:1343): avc: denied { read } for path="/dev/socket/[33491]"
dev="sockfs" ino=33491 scontext=u:r:logd:s0 tcontext=u:r:init:s0 tclass=unix_stream_socket permissive=1
05-08 11:18:42.335 163 163 I logd.reader: type=1400 audit(0.0:1344): avc: denied { setopt } for path="/dev/socket/lo
gdr" scontext=u:r:logd:s0 tcontext=u:r:init:s0 tclass=unix_stream_socket permissive=1
05-08 11:19:10.657 522 1250 D WifiCondControl: Scan result ready event
05-08 11:19:31.978 2348 2348 D ** SYS_CTRL **: : EVABS{logging_info_never_safe1}
05-08 11:19:31.975 2348 2348 I com.revo.evabs: type=1400 audit(0.0:1345): avc: denied { sendto } for path="/dev/socket
/logdw" scontext=u:r:untrusted_app:s0:c512,c768 tcontext=u:r:init:s0 tclass=unix_dgram_socket permissive=1
05-08 11:19:31.978 2348 2348 D ** SYS_CTRL **: : EVABS{logging_info_never_safe1}
05-08 11:20:00.020 522 538 E memtrack: Couldn't load memtrack module
05-08 11:20:00.020 522 538 W android.os.Debug: failed to get memory consumption info: -1
```

Vậy flag cần tìm của bài này là: `EVABS{logging_info_never_safe1}`.


2. Level 2: File Access

Giao diện khi ta vào level 2 (File Access) của app sẽ có dạng như sau:

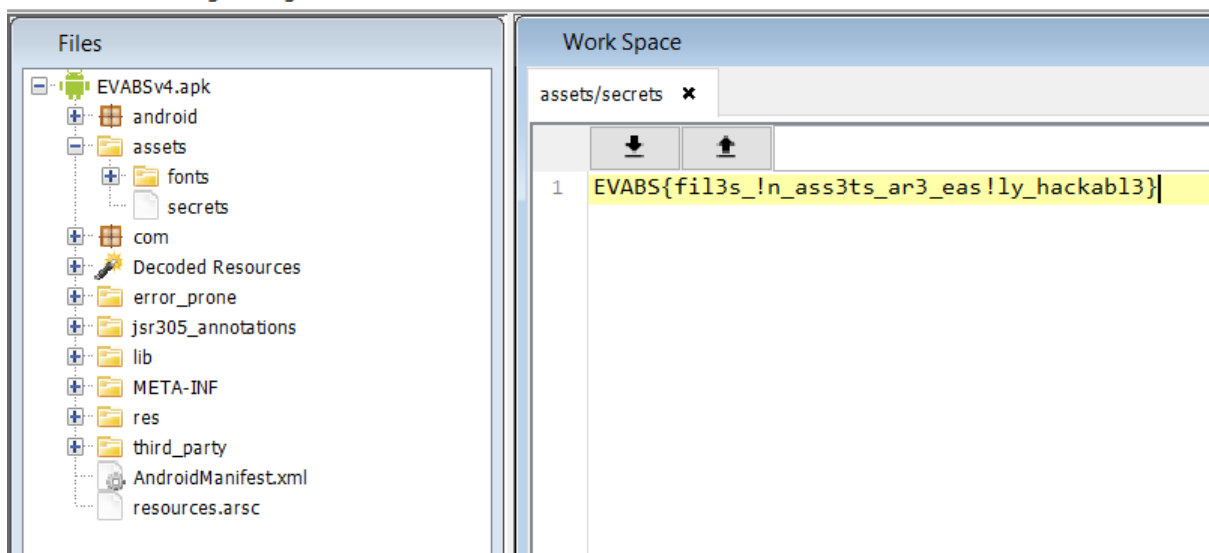


Với dòng miêu tả trên, ta thấy rằng flag của bài này có thể được đặt tại nơi lưu trữ tài nguyên của một tập tin apk. Theo cấu trúc của tập tin apk thì một trong những nơi lưu trữ tài nguyên là tại thư mục **asset**, để mở thư mục này trong tập tin apk ta sẽ sử dụng Bytecode Viewer.

Ta truy cập vào đường dẫn **assets/secrets** và phát hiện được flag của bài này:

 Bytecode Viewer 2.9.22 - <https://bytecodeviewer.com> | <https://the.bytecode.club> - @Konloch

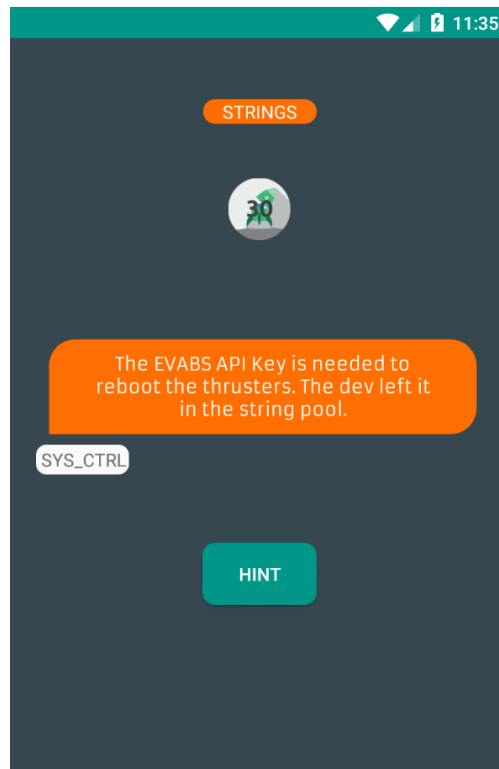
File View Settings Plugins



Vậy flag cần tìm của bài này là: **EVABS{fil3s_!n_ass3ts_ar3_eas!ly_hackabl3}**.

3. Level 3: Strings

Giao diện khi ta vào level 3 (Strings) của app sẽ có dạng như sau:



Với dòng miêu tả ở trên, ta biết được flag cần tìm là 1 API KEY và được lập trình viên lưu như một chuỗi kí tự, bên cạnh đó ta biết được rằng các chuỗi sử dụng trong ứng dụng apk được lưu tại tập tin `string.xml`, để có được tập tin này ta sẽ dịch ngược ứng dụng bằng apktool:

```
C:\Users\Win 10>apktool d EVABSV4.apk
I: Using Apktool 2.5.0 on EVABSV4.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Win 10\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Sau đó, ta truy cập vào đường dẫn `EVABSV4\res\values\strings.xml`. Ta tìm kiếm chuỗi EVABS trong tập tin `strings.xml` cho đến khi tìm thấy được flag.

```

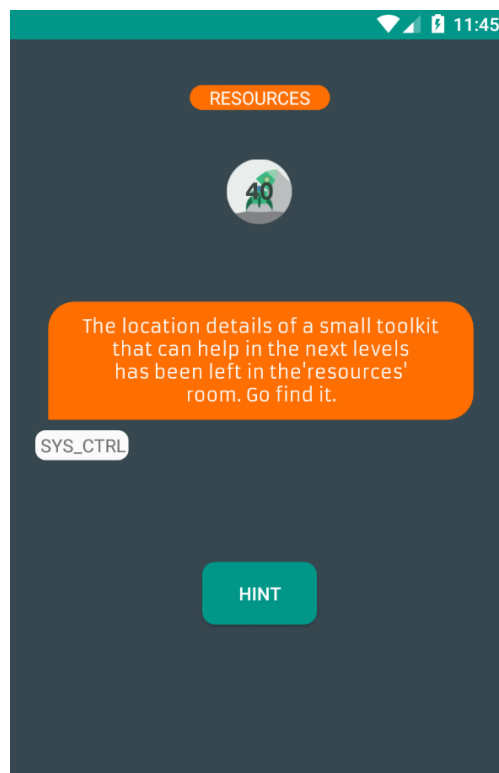
<string name="path_password_strike_through">M3.27,4.27 L19.74,20.74</string>
<string name="permission_rationale">"Contacts permissions are needed for providin
  completions."</string>
<string name="project_id">evabs-c0e8b</string>
<string name="prompt_email">Email</string>
<string name="prompt_password">Password (optional)</string>
<string name="search_menu_title">Search</string>
<string name="section_format">Hello World from section: %1$d</string>
<string name="status_bar_notification_info_overflow">999+</string>
<string name="the_evabs_api_key">EVABS{saf3ly_st0red_in_Strings?}</string>
<string name="title_activity_home">Home</string>
<string name="title_activity_launch">Launch</string>
<string name="title_activity_login">Sign in</string>
<string name="title_activity_splash">Splash</string>
<string name="title_activity_test">Test</string>
</resources>

```

Vậy flag cần tìm của bài này là: EVABS{saf3ly_st0red_in_Strings?}.

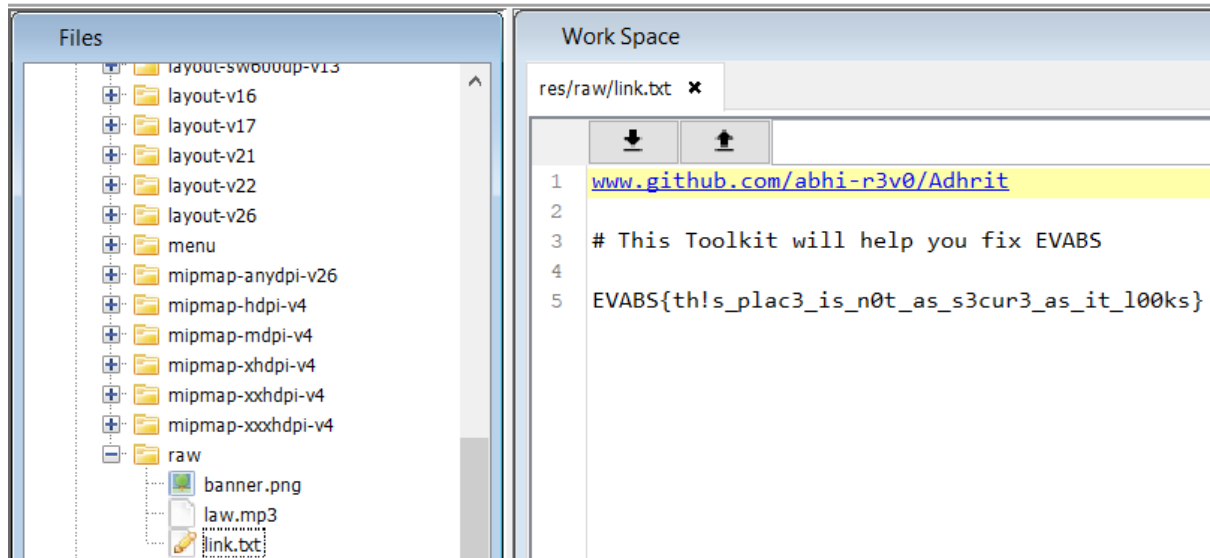
4. Level 4: Resources

Giao diện khi ta vào level 4 (Resources) của app sẽ có dạng như sau:



Dựa vào mô tả trên, ta biết được giá trị flag của bài này sẽ nằm đâu đó trong thư mục res (resources) của tập tin apk. Do đó, ta sẽ sử dụng Bytecode Viewer để mở tập tin EVABSV4.apk và tìm kiếm trong đó. Kết quả là ta tìm thấy flag trong đường dẫn res/raw/link.txt.

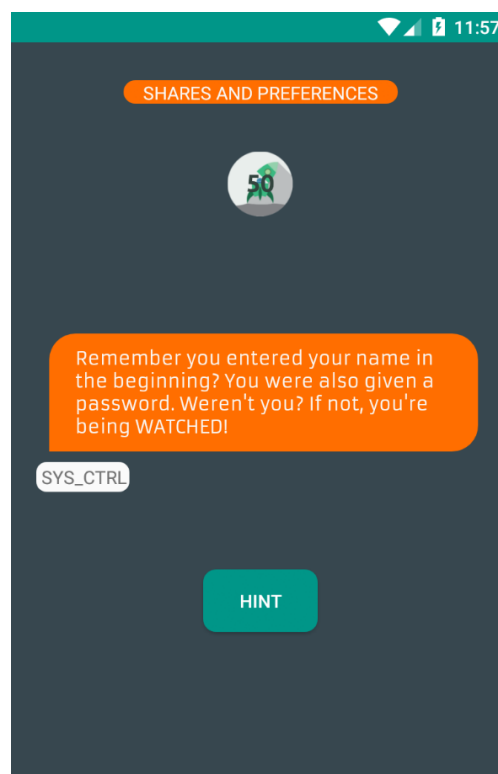
File View Settings Plugins



Vậy flag cần tìm của bài này là: `EVABS{th!s_plac3_is_n0t_as_s3cur3_as_it_l00ks}`.

5. Level 5: Shares and Prefs

Giao diện khi ta vào level 5 (Shares and Prefs) của app sẽ có dạng như sau:



Data được lưu trữ ở dạng key-value trong Shared Preferences được xây dựng sẵn trong Android. Ta thực hiện vào Shared Preferences để xem thông tin. Shared Preferences

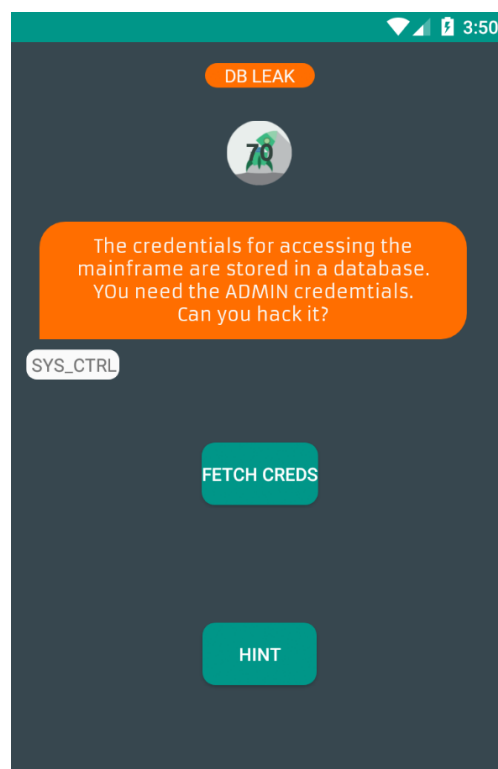
được lưu tại đường dẫn `data/data/<YOUR_APP_ID>/shared_prefs/<SHARED_PREF_NAME>.xml`.

```
PS D:\Virtual Machine\platform-tools> .\adb shell
vbox86p:/ # cd /data/data/com.revo.evabs/shared_prefs
vbox86p:/data/data/com.revo.evabs/shared_prefs # ls
DETAILS.xml  PREFERENCE.xml
vbox86p:/data/data/com.revo.evabs/shared_prefs # cat PREFERENCE.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <boolean name="isFirstRun" value="false" />
</map>
vbox86p:/data/data/com.revo.evabs/shared_prefs # cat DETAILS.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="password">EVABS{shar3d_pr3fs_c0uld_be_c0mpromiz3ds}</string>
  <string name="username">Anonymous</string>
</map>
```

Có 2 tập tin `.xml`, xem lần lượt từng tập tin thì ta thấy file `DETAILS.xml` có chứa flag. Vậy flag cần tìm của bài này là: `EVABS{shar3d_pr3fs_c0uld_be_c0mpromiz3ds}`.

6. Level 6: DB Leak

Giao diện khi ta vào level 6 (DB Leak) của app sẽ có dạng như sau:



Dựa vào mô tả trên, trước tiên, ta sẽ tìm kiếm thử nơi lưu cơ sở dữ liệu của hệ thống.

```
vbox86p:/data/data/com.revo.evabs # ls
cache  code_cache  databases  lib  shared_prefs
```

Như vậy, cơ sở dữ liệu của hệ thống nằm trong đường dẫn `/data/data/com.revo.evabs/databases`.

```
vbox86p:/data/data/com.revo.evabs/databases # ls
MAINFRAME_ACCESS  MAINFRAME_ACCESS-journal
```

Hiện tại có 2 cơ sở dữ liệu trong hệ thống. Ta lần lượt kiểm tra từng cơ sở dữ liệu, trước tiên là `MAINFRAME_ACCESS`. Biết rằng hệ thống đang sử dụng SQLite, ta sử dụng lệnh `sqlite3 MAINFRAME_ACCESS` để vào SQLite shell. Ta kiểm tra thử xem cơ sở dữ liệu `MAINFRAME_ACCESS` có những bảng nào và kiểm tra dữ liệu của mỗi bảng sử dụng lệnh `select * from <tên_bảng>;`

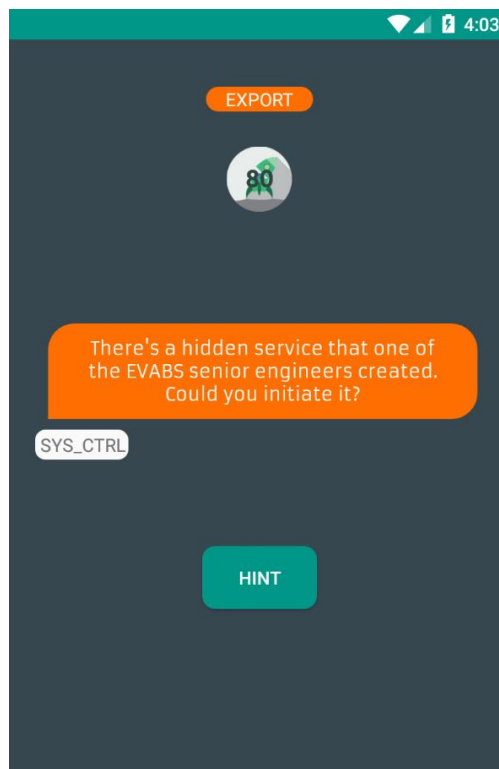
```
vbox86p:/data/data/com.revo.evabs/databases # sqlite3 MAINFRAME_ACCESS
SQLite version 3.18.2 2017-07-21 07:56:09
Enter ".help" for usage hints.
sqlite> .tables
CREDS          android_metadata
sqlite> select * from CREDS;
Dr.133t|EVABS{sqlite_is_not_safe}|E|ADMIN
Mr BufferOverflow|0xNotSecureSQLite_|STAFF
Ms HeapSpray|SQLite_exploit|USER
```

Trong bảng `CREDS` của cơ sở dữ liệu `MAINFRAME_ACCESS` có chứa flag cần tìm.

Vậy flag cần tìm của bài này là: `EVABS{sqlite_is_not_safe}`.

7. Level 7: Export

Giao diện khi ta vào level 7 (Export) của app sẽ có dạng như sau:



Dựa vào mô tả trên, ta sử dụng apktool để đọc tập tin apk và decompile tập `AndroidManifest.xml` về tập tin `.xml` nguyên bản.

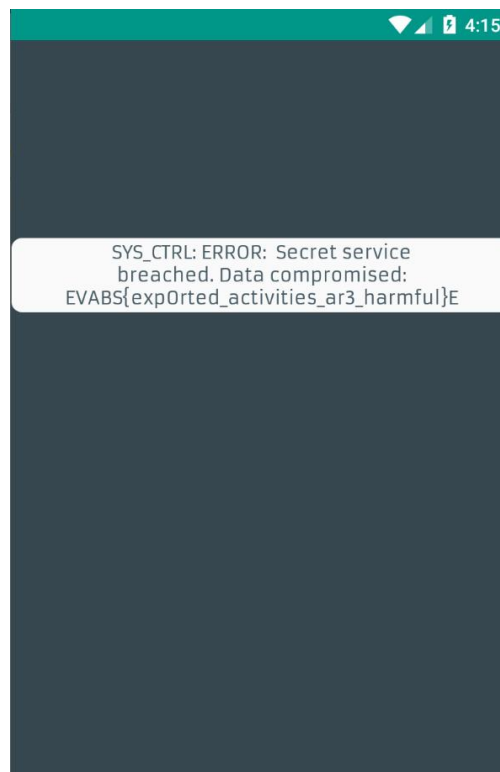
```
1 <?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res,
2 <uses-permission android:name="android.permission.INTERNET"/>
3 <application android:allowBackup="false" android:debuggable="true" android:icon="@mipmap/ic_launcher" android
4 <activity android:exported="true" android:name="com.revo.evabs.ExportedActivity"/>
5 <activity android:name="com.revo.evabs.Frida1"/>
6 <activity android:name="com.revo.evabs.FileRead"/>
7 <activity android:name="com.revo.evabs.DebugMe"/>
8 <activity android:name="com.revo.evabs.Welcome"/>
9 <activity android:name="com.revo.evabs.ChallengeList" android:parentActivityName="com.revo.evabs.Launch">
```

Ta để ý dòng 4, thuộc tính `exported` được set thành `true`, tức là activity này có thể được gọi bởi ứng dụng khác.

Ta thực thi lệnh activity tại dòng 4 trong shell sử dụng lệnh `adb shell`.

```
PS D:\Virtual Machine\platform-tools> .\adb shell am start -n com.revo.evabs/com.revo.evabs.ExportedActivity
Starting: Intent { cmp=com.revo.evabs/.ExportedActivity }
```

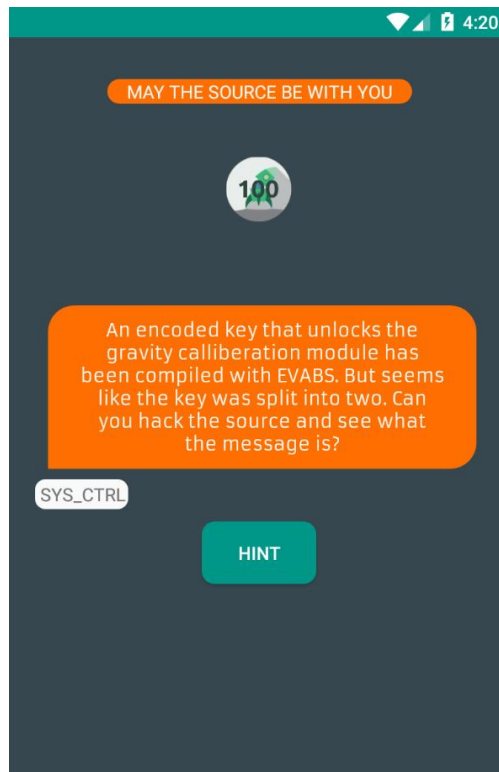
Sau đó, giao diện điện thoại ảo xuất hiện dòng chữ sau:



Vậy flag cần tìm của bài này là: `EVABS{exp0rted_activities_ar3_harmful}`.

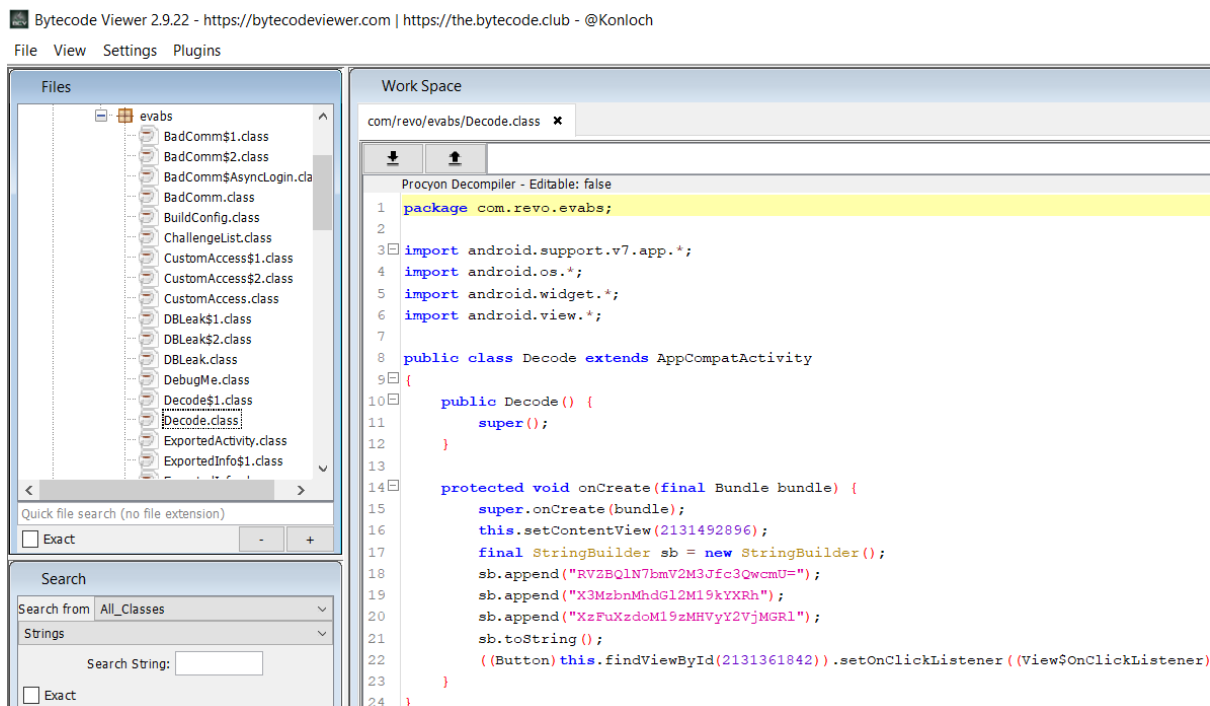
8. Level 8: Decode

Giao diện khi ta vào level 8 (Decode) của app sẽ có dạng như sau:



Ta sử dụng ByteCode Viewer để phân tích tập tin EVABSV4.apk.

Ta mở tập tin Decode.class và tiến hành xem xét.



Ở đây ta thấy các chuỗi được hardcoded, thử decode dạng base64 chuỗi (nhóm vào trang web <https://www.base64decode.org/> để decode chuỗi sang từ dạng base64 sang utf-8).

Decode from Base64 format


Simply enter your data then push the decode button.


```
RVZBQIN7bmV2M3Jfc3QwcmU=  
X3MzbnMhdGl2M19kYXRh  
XzFuXzdoM19zMHVyY2VjMGRl
```

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☒ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

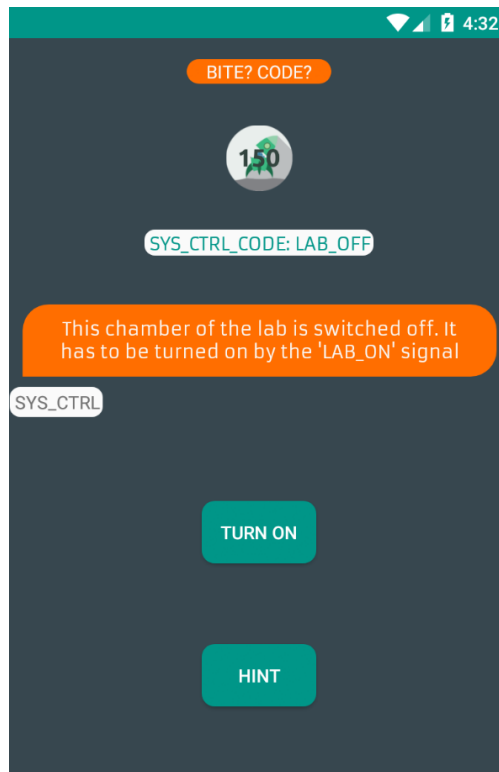
 < **DECODE** > Decodes your data into the area below.

```
EVABS{nev3r_st0re  
_s3ns!tiv3_data  
_1n_7h3_s0urcec0de}
```

Vậy flag cần tìm của bài này là: EVABS{nev3r_st0re_s3ns!tiv3_data_1n_7h3_s0urcec0de}.

9. Level 9: Smali Injection

Giao diện khi ta vào level 9 (Smali Injection) của app sẽ có dạng như sau:



Ta sử dụng ByteCode Viewer để phân tích tập tin EVABSV4.apk.

Ta mở tập tin SmaliInject.class và tiến hành xem xét.

```

1 package com.revo.evabs;
2
3 import android.support.v7.app.*;
4 import android.os.*;
5 import android.widget.*;
6 import android.view.*;
7
8 public class SmaliInject extends AppCompatActivity
9 {
10     String SIGNAL;
11
12     static {
13         System.loadLibrary("native-lib");
14     }
15
16     public SmaliInject() {
17         super();
18         this.SIGNAL = "LAB_OFF";
19     }
20
21     protected void onCreate(final Bundle bundle) {
22         super.onCreate(bundle);
23         this setContentView(2131492906);
24         final Button button = (Button) this.findViewById(2131361846);
25         final Button button2 = (Button) this.findViewById(2131361845);
26         final TextView textView = (TextView) this.findViewById(2131362096);
27         final TextView textView2 = (TextView) this.findViewById(2131362104);
28         final TextView textView3 = (TextView) this.findViewById(2131362095);
29         final TextView textView4 = (TextView) this.findViewById(2131362087);
30         button2.setOnClickListener((View.OnClickListener) new SmaliInject.SmaliInject$1(this, textView2));
31         button.setOnClickListener((View.OnClickListener) new SmaliInject.SmaliInject$2(this, textView3, textView, textView4));
32     }
33
34     public native String stringFromSmali();
35 }

```

Tiếp theo, ta quan sát tập tin SmaliInject\$2.class.

```

20 public void onClick(final View view) {
21     final String stringFromSmali = this.this$0.stringFromSmali();
22     if (this.this$0.SIGNAL.equals("LAB_ON")) {
23         this.val$tvlaboff.setText((CharSequence)"SYS_CTRL_CODE: LAB_ON");
24         this.val$labstat.setText((CharSequence)"SYS_CTRL: ACCESS_GRANTED. LAB UNLOCKED");
25         final TextView val$tvflag = this.val$tvflag;
26         final StringBuilder sb = new StringBuilder();
27         sb.append("EVABS{");
28         sb.append(stringFromSmali);
29         sb.append("}");
30         val$tvflag.setText((CharSequence)sb.toString());
31     }
32     else {
33         this.val$tvlaboff.setText((CharSequence)"SYS_CTRL_CODE: LAB_OFF");
34         this.val$labstat.setText((CharSequence)"SYS_CTRL: ACCESS_DENIED");
35     }
36 }

```

Như ta thấy, nếu biến `SIGNAL = "LAB_ON"` thì chương trình sẽ in ra flag. Nhưng biến `SIGNAL` đang được đặt mặc định là `"LAB_OFF"` và không có lệnh nào thay đổi giá trị của biến này cả. Vì thế chúng ta cần sửa code smali. Ta dịch ngược tập tin apk bằng apktool và mở tập tin `SmaliInject.class` lên. Sau đó, ta sửa `LAB_OFF` thành `LAB_ON`.

SmaliInject.smali ●

C: > Users > Win 10 > EVABSV4 > smali > com > revo > evabs > SmaliInject.smali

```

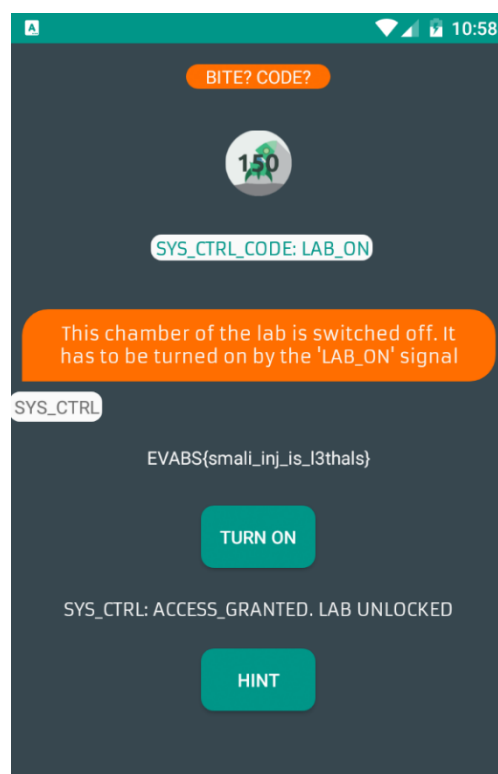
23 .method public constructor <init>()V
24     .locals 1
25
26     .line 11
27     invoke-direct {p0}, Landroid/support/v7/app/CompatActivity;-><init>()V
28
29     .line 13
30     const-string v0, "LAB_ON"
31
32     iput-object v0, p0, Lcom/revo/evabs/SmaliInject;->SIGNAL:Ljava/lang/String;
33
34   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

```

Tiếp theo, ta vá lại tập tin mới `AVABSV4_1.apk`.


```
C:\Users\Win 10>apktool b EVABsv4 -o EVABsv4_1.apk
I: Using Apktool 2.5.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
```

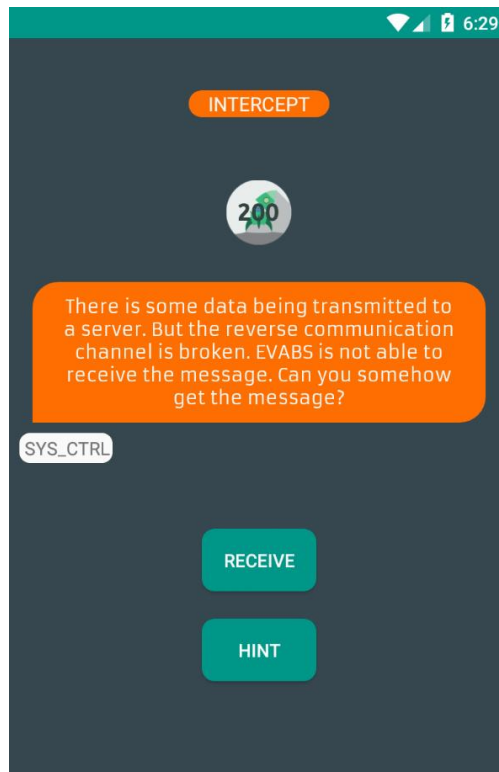
Cuối cùng, ta vào app lại và nhấn nút **TURN ON**. Kết quả là ta sẽ thấy xuất hiện một chuỗi chứa flag cần tìm phía trên nút **TURN ON**.



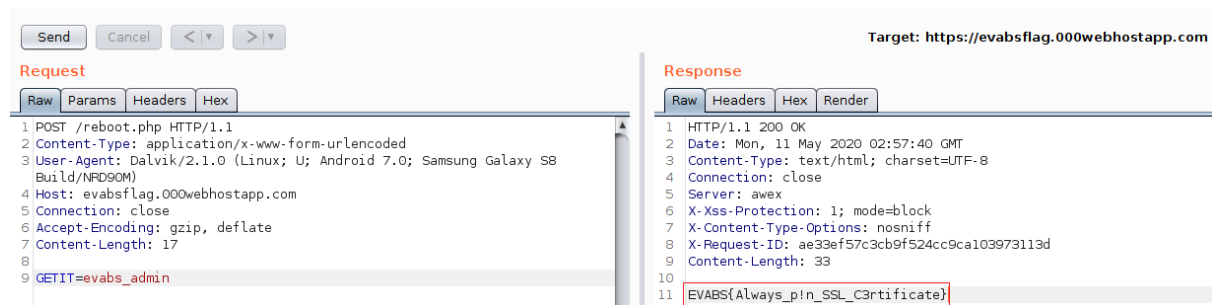
Vậy flag cần tìm của bài này là: `EVABS{smali _inj_is _l3thals}`.

10. Level 10: Interception

Giao diện khi ta vào level 10 (Interception) của app sẽ có dạng như sau:



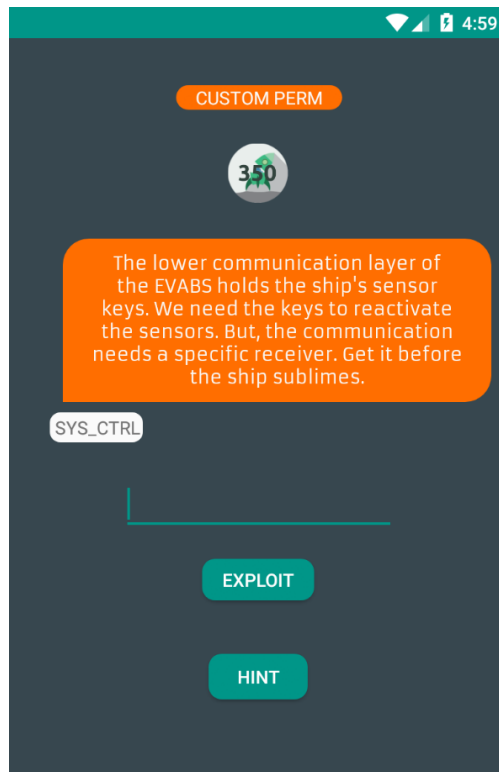
Ta dùng Burp Suite bắt request khi ta nhấn nút **RECEIVE**. Sau đó ta **Send to Request...** và bấm **Send**. Trong phần Response sẽ chứa thông tin của flag.



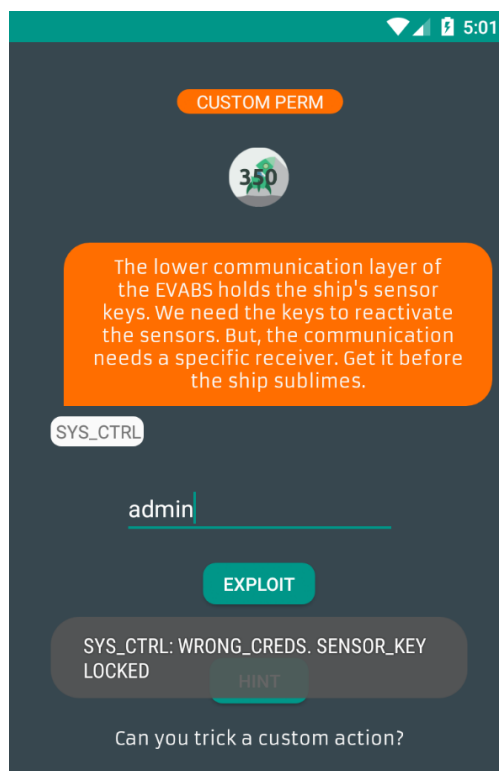
Vậy flag cần tìm của bài này là: EVABS{Always_p!n_SSL_C3rtificate}.

11. Level 11: Custom Access

Giao diện khi ta vào level 11 (Custom Access) của app sẽ có dạng như sau:



Ta nhập thử một chuỗi bất kì (ví dụ là `admin`) và nhấn nút **EXPLOIT**. Kết quả cho ta biết ta đã nhập sai chuỗi:



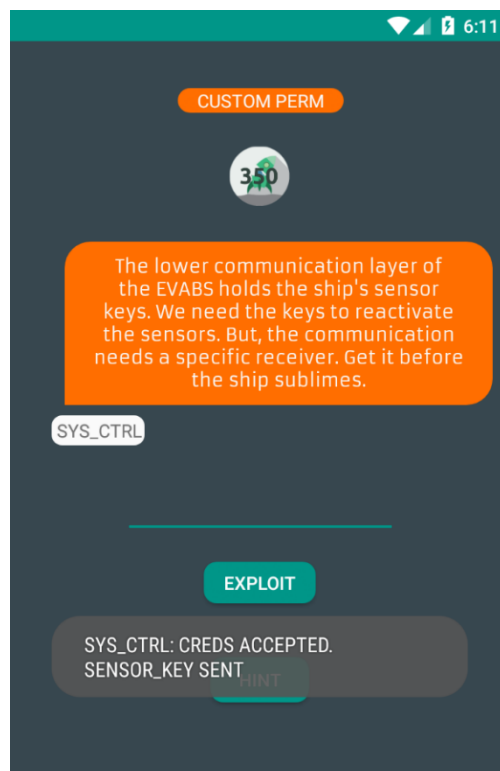
Ta sử dụng ByteCode Viewer để phân tích tập tin `EVABSV4.apk`.
Ta mở tập tin `CustomAccess.class` và tiến hành xem xét.

```

22 private void GetSensorKey() {
23     if (new String(new char[] { 'c', 'u', 's', 't', '0', 'm', '_', 'p', '3', 'r', 'm' }).equals(((EditText)th
24         Toast.makeText((Context)this, (CharSequence)"SYS_CTRL: CREDs ACCEPTED. SENSOR_KEY SENT", 1).show();
25         final Intent intent = new Intent("com.revo.evabs.action.SENSOR_KEY");
26         final StringBuilder sb = new StringBuilder();
27         sb.append("EVABS(");
28         sb.append(this.stringFromJNI());
29         sb.append(")");
30         intent.putExtra("android.intent.extra.TEXT", sb.toString());
31         intent.setType("text/plain");
32         this.startActivity(intent);
33     }
34     else {
35         Toast.makeText((Context)this, (CharSequence)"SYS_CTRL: WRONG_CREDs. SENSOR_KEY LOCKED", 1).show();
36     }
37 }

```

Ở bài này, ta phải tìm đúng input mới được, xem code ở trên là ta thấy ngay input đúng là `cust0m_p3rm`. Ta nhập chuỗi này vào trong app và nhấn nút **EXPLOIT** thu được kết quả như hình sau:



Sau khi nhập đúng input thì flag đã được truyền vào intent `com.revo.evabs.action.SENSOR_KEY` bằng hàm `putExtra()`.

Ta sẽ sử dụng hook với frida để sửa hàm `putExtra()` cho nó in ra flag. Ta tạo một tập tin `hook.py` có nội dung như sau:

```

1  import frida
2  import sys
3
4  def onMessage(message, data):
5      |   print(message)
6
7  package = "com.revo.evabs"
8
9  hook_script="""
10 Java.perform
11 (
12     function()
13     {
14         var intent = Java.use("android.content.Intent");
15         intent.putExtra.overload("java.lang.String", "java.lang.String").implementation = function(var_1, var_2)
16         {
17             |   send("[+] Flag: " + var_2);
18         };
19     }
20 );
21 """
22
23 process = frida.get_usb_device().attach(package)
24 script = process.create_script(hook_script)
25 script.on("message", onMessage)
26 print("+ Hooking", package)
27 script.load()
28 sys.stdin.read()

```

Sau đó, ta chạy tập tin `hook.py` sau khi đã chạy `frida server`. Qua bên app ta nhập lại chuỗi `cust0m_p3rm` và nhấn nút **EXPLOIT**. Phía bên màn hình shell sẽ xuất hiện những thông tin như sau:

```

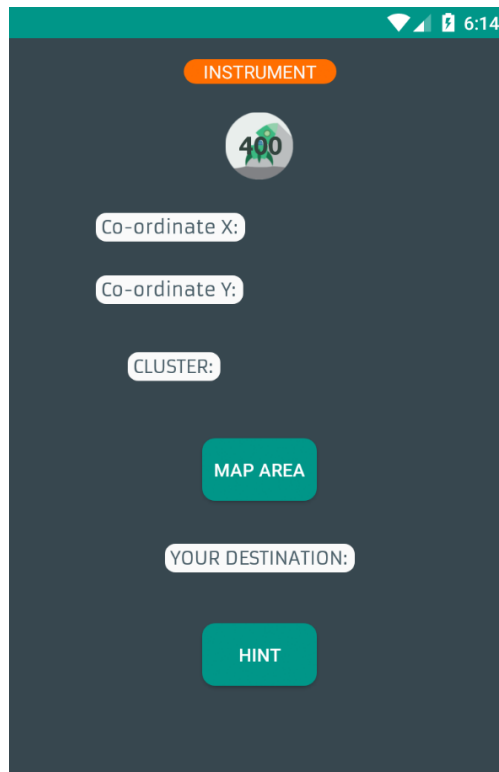
D:\Virtual Machine\platform-tools>python hook.py
+ Hooking com.revo.evabs
{'type': 'send', 'payload': '[+] Flag: EVABS{always_verify_packag3sa}'}
{'type': 'error', 'description': 'Error: Implementation for putExtra expected return value compatible with android.conte
nt.Intent', 'stack': 'Error: Implementation for putExtra expected return value compatible with android.content.Intent\n
    at ne (frida/node_modules/frida-java-bridge/lib/class-factory.js:614)\n    at <anonymous> (frida/node_modules/frida-j
ava-bridge/lib/class-factory.js:592)', 'fileName': 'frida/node_modules/frida-java-bridge/lib/class-factory.js', 'lineNum
ber': 614, 'columnNumber': 1}

```

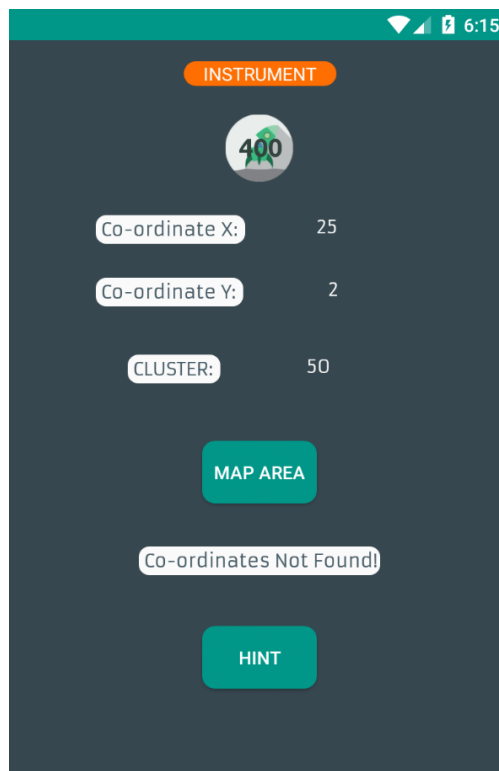
Vậy flag cần tìm của bài này là: `EVABS{always_verify_packag3sa}`.

12. Level 12: Instrument

Giao diện khi ta vào level 12 (Instrument) của app sẽ có dạng như sau:



Ta nhấn thử nút **MAP AREA**.



Ta sử dụng ByteCode Viewer để phân tích tập tin EVABSv4.apk.

Ta mở tập tin Frida1.class và tiến hành xem xét.

```

20 public Fridal() {
21     super();
22     this.a = 25;
23     this.b = 2;
24 }
25
26 public void onClick(final View view) {
27     final TextView textView = (TextView) this.findViewById(2131361996);
28     final TextView textView2 = (TextView) this.findViewById(2131362132);
29     final TextView textView3 = (TextView) this.findViewById(2131362134);
30     final TextView textView4 = (TextView) this.findViewById(2131362142);
31     textView2.setText((CharSequence)String.valueOf(this.a));
32     textView3.setText((CharSequence)String.valueOf(this.b));
33     this.x = this.a * this.b;
34     final int nextInt = new Random().nextInt(70);
35     textView4.setText((CharSequence)String.valueOf(this.x));
36     if (this.x > nextInt + 150) {
37         textView.setText((CharSequence)"VIBRAN IS RESDY TO FLY! YOU ARE GOING HOME!");
38         Log.d("CONGRATZ!", this.stringFromJNI());
39     }
40     else {
41         textView.setText((CharSequence)"Co-ordinates Not Found!");
42     }
43 }

```

Dựa theo code trên, biến x tương ứng với giá trị của cluster, nếu lớn hơn $\text{var5} + 150$ với var5 là một số nguyên ngẫu nhiên từ 0 đến 70 thì chúng ta sẽ thành công có được flag. Nhưng vì x là tích của $a=25$ và $b=2$, x cố định và bằng 50 trong khi var5 có giá trị ngẫu nhiên từ 0 đến 70 nên $150 \leq \text{var5} + 150 \leq 220$ nghĩa là x luôn bé hơn $\text{var5}+150$.

Ta sẽ sử dụng hook với frida để sửa hàm `nextInt(int)` cho trả về trực tiếp giá trị -125 . Khi đó $-125 + 150 < 50$.

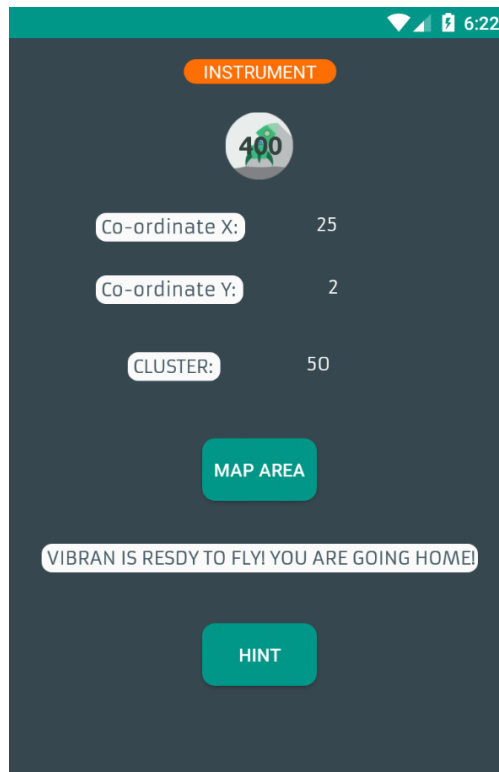
Ta tạo một tập tin `hook.py` có nội dung như sau:

```

1  import frida
2  import sys
3
4  def onMessage(message, data):
5      |   print(message)
6
7  package = "com.revo.evabs"
8
9  hook_script="""
10  Java.perform
11  (
12      function()
13      {
14          |   var random = Java.use("java.util.Random");
15          random.nextInt.overload("int").implementation = function(var_1)
16          {
17              |   return -125;
18          };
19      }
20  );
21  """
22
23  process = frida.get_usb_device().attach(package)
24  script = process.create_script(hook_script)
25  script.on("message", onMessage)
26  print("+ Hooking", package)
27  script.load()
28  sys.stdin.read()

```

Sau đó, ta chạy tập tin `hook.py` sau khi đã chạy `frida server`. Qua bên app ta nhấn lại nút **MAP AREA**.



Phía bên màn hình shell sẽ xuất hiện những thông tin như sau:

```
D:\Virtual Machine\platform-tools>python hook.py
+ Hooking com.revo.evabs
```

Ta vẫn chưa thấy flag. Có khả năng nó phản hồi trong log nên ta sử dụng lệnh `adb logcat` để hiển thị xem có thông tin của flag hay không.

```
pfs" ino=8089 scontext=u:r:init:s0 tcontext=u:object_r:fuse_device:s0 tclass=chr_file
05-08 18:22:46.284 498 498 I chatty : uid=0(root) /system/bin/batteryd identical 317 lines
05-08 18:22:46.284 498 498 W batteryd: type=1400 audit(0.0:9265): avc: granted { read } for path="/dev/fuse" dev="tm
pfs" ino=8089 scontext=u:r:init:s0 tcontext=u:object_r:fuse_device:s0 tclass=chr_file
05-08 18:22:47.900 1606 1606 D CONGRATZ!: EVABS{a_dynamic_h00k}E
05-08 18:22:48.276 498 498 W batteryd: type=1400 audit(0.0:9268): avc: granted { read } for path="/dev/fuse" dev="tm
pfs" ino=8089 scontext=u:r:init:s0 tcontext=u:object_r:fuse_device:s0 tclass=chr_file
05-08 18:22:58.280 498 498 I chatty : uid=0(root) /system/bin/batteryd identical 118 lines
05-08 18:22:58.284 498 498 W batteryd: type=1400 audit(0.0:9397): avc: granted { read } for path="/dev/fuse" dev="tm
pfs" ino=8089 scontext=u:r:init:s0 tcontext=u:object_r:fuse_device:s0 tclass=chr_file
05-08 18:23:00.002 464 478 E memtrack: Couldn't load memtrack module
```

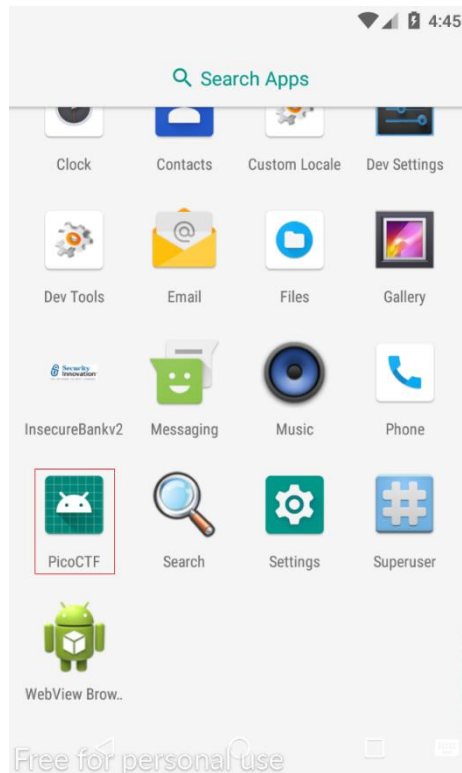
Vậy flag cần tìm của bài này là: `EVABS{a_dynamic_h00k}`.

CHƯƠNG 2. DROID

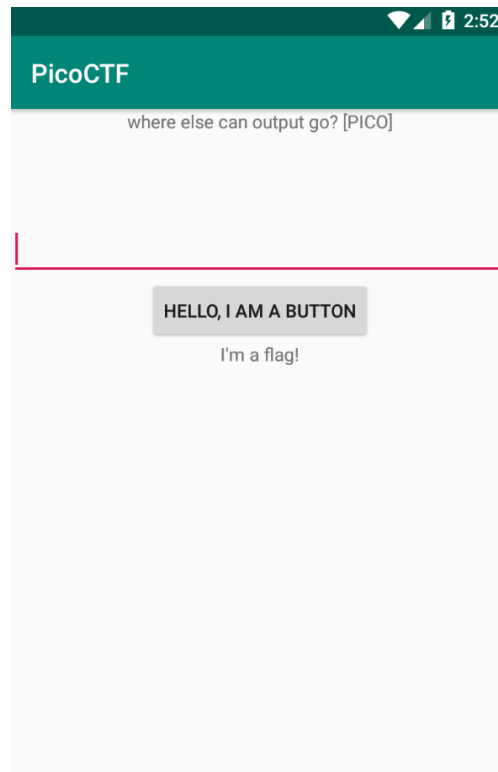
1. Nhật ký droid đã đi đâu. Bạn có thể tìm thấy tại: `one.apk`

Ta tiến hành cài đặt app `one.apk` sử dụng câu lệnh `adb install one.apk`.

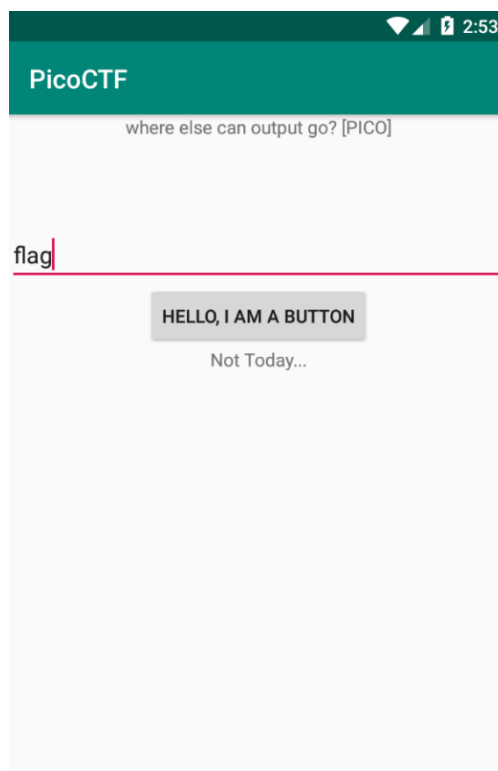
Sau đó, ta mở điện thoại ảo lên và mở app PicoCTF:



Giao diện của app PicoCTF sẽ như hình sau:



Ta nhập thử một giá trị bất kì (ví dụ là `flag`) và bấm nút **HELLO, I AM A BUTTON** thử.



Như ta thấy, dòng chữ `I'm a flag!` ở phía dưới nút bấm bị chuyển thành `Not Today...`

Giờ ta phải tìm xem flag được để ở đâu. Có nhiều tình huống để flag, nhưng đơn giản nhất là flag phản hồi trên log. Do đó, trước tiên, ta có thể xem thông tin droid log thông qua lệnh `logcat`.

Ta vào command shell gõ lệnh `adb logcat`.

Tiếp theo, quay trở lại giao diện trên trên điện thoại ảo và bấm nút **HELLO, I AM A BUTTON** một lần nữa. Kết quả về thông tin của droid sẽ xuất hiện như hình dưới. Và để ý thì chúng ta sẽ thấy thông tin của flag cần tìm.

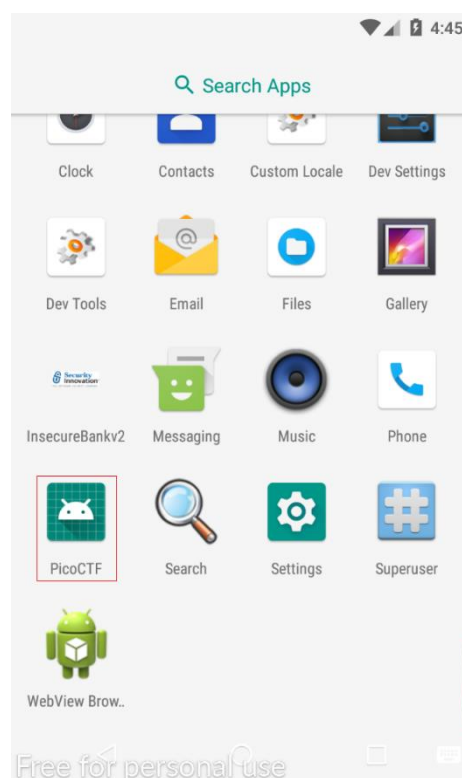
```
05-08 19:20:26.600 528 1334 D NetworkMonitor/NetworkAgentInfo [WIFI () - 100]: PROBE_FALLBACK http://www.google.com/100
05-08 19:20:26.601 528 1334 D NetworkMonitor/NetworkAgentInfo [WIFI () - 100]: PROBE_FALLBACK http://www.google.com/100
05-08 19:20:26.601 528 640 D ConnectivityService: NetworkAgentInfo [WIFI () - 100] validation failed
05-08 19:20:26.602 528 634 D WifiStateMachine: NETWORK_STATUS_UNWANTED_VALIDATION_FAILED
05-08 19:20:28.421 1776 1776 I PICO : picoCTF{a.moose.once.bit.my.sister}
05-08 19:20:32.329 528 553 I ActivityManager: Killing 1513:com.android.messaging/u0a65 (adj 906): empty #17
05-08 19:20:32.330 528 553 E memtrack: Couldn't load memtrack module
05-08 19:20:32.330 528 553 W android.os.Debug: failed to get memory consumption info: -1
```

Vậy flag cần tìm của bài này là: `picoCTF{a.moose.once.bit.my.sister}`.

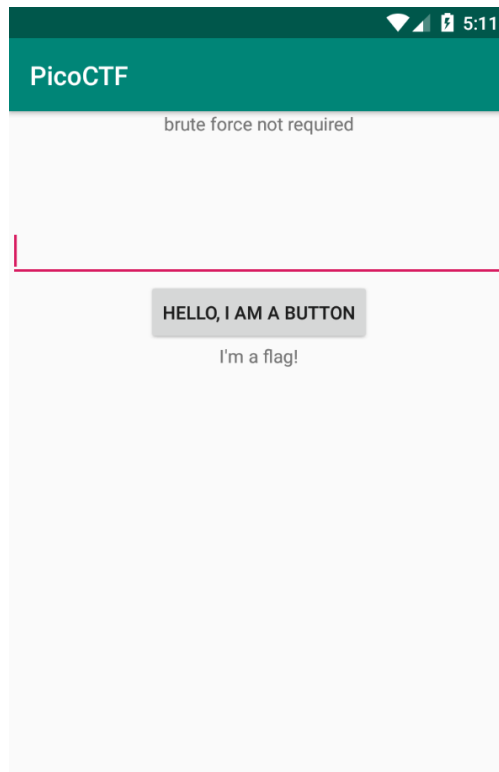
2. Tìm kiếm và lấy flag. Bạn có thể tìm thấy tại: `two.apk`

Ta tiến hành cài đặt app `two.apk` sử dụng câu lệnh `adb install two.apk`.

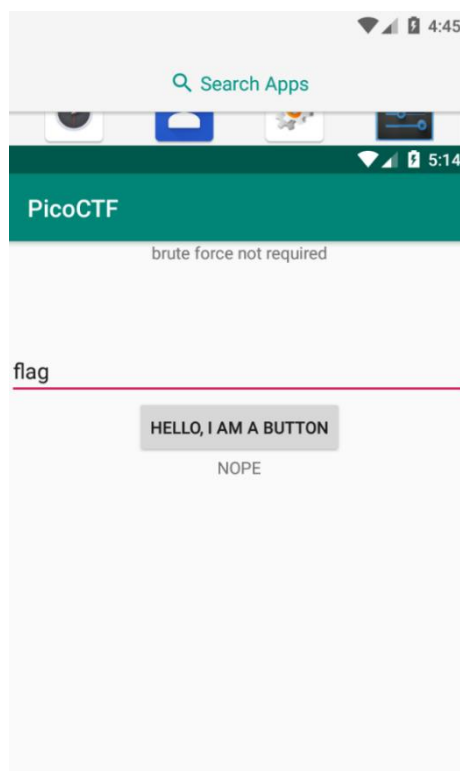
Sau đó, ta mở điện thoại ảo lên và mở app PicoCTF:



Giao diện của app PicoCTF sẽ như hình sau:



Ta nhập thử một giá trị bất kì (ví dụ là `flag`) và bấm nút **HELLO, I AM A BUTTON** thử.



Như ta thấy, dòng chữ `I'm a flag!` ở phía dưới nút bấm bị chuyển thành `NOPE`. Ta không thể sử dụng `logcat` như bài 2.1. để tìm flag được.

Giờ ta mở Bytecode Viewer lên và kéo thả tập tin `two.apk` vào. Sau đó, ta truy cập vào đường dẫn `com/helloctmu/picoctf/MainActivity.class`. Ta để ý dòng code từ dòng 21 đến dòng 24 trong tập tin `MainActivity.class`:

```
21 public void buttonClick(final View view) {
22     this.text_bottom.setText((CharSequence)FlagstaffHill.getFlag(this.text_input.getText().toString(), this.ctx));
23 }
24
```

Hàm `buttonClick()` sẽ thực hiện công việc thay đổi của dòng chữ bên dưới nút bấm khi ta tiến hành bấm nút thông qua hàm `getFlag()` trong tập tin `FlagstaffHill.class`.

Ta mở tập tin `FlagstaffHill.class` và xem code của hàm `getFlag()` từ dòng 13 đến dòng 18.

```
13 public static String getFlag(final String s, final Context context) {
14     if (s.equals(context.getString(2131427375))) {
15         return fenugreek(s);
16     }
17     return "NOPE";
18 }
```

Hàm `getFlag()` sẽ kiểm tra input nhập vào với giá trị của resource '2131427375'. Nếu đúng thì sẽ cho ta biết giá trị của flag.

Ta tiến hành dịch ngược ứng dụng sử dụng apktool. Trong shell, ta sử dụng lệnh `apktool d two.apk` để tiến hành dịch ngược ứng dụng.

```
C:\Users\Win 10>apktool d two.apk
I: Using Apktool 2.5.0 on two.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Win 10\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Sau đó, ta kiểm tra trong đường dẫn `res/values/public.xml` trước. Trong đây sẽ lưu các resource name và resource id sử dụng trong code. Quan sát ta thấy các resource id đều đang ở dạng hex, vì thế ta chuyển đổi resource id là 2131427375 sang dạng hex thành 7f0b002f.

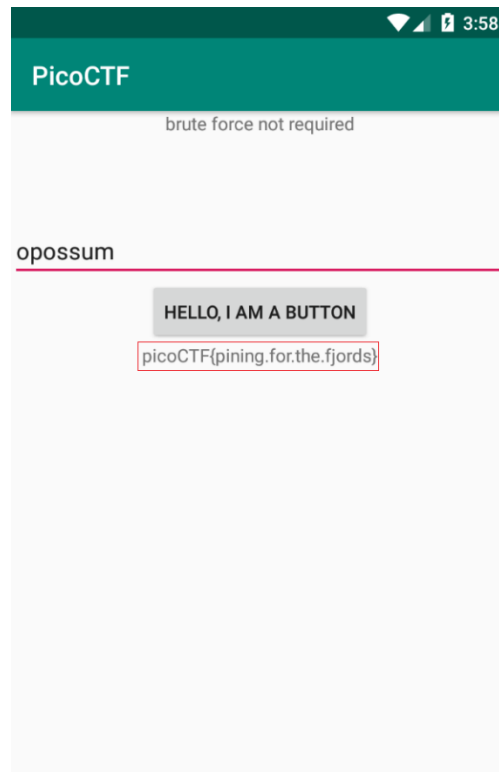
```
public.xml X
C: > Users > Win 10 > two > res > values > public.xml
888 <public type="string" name="contentall" id="0x7f0b002a" />
889 <public type="string" name="gopher" id="0x7f0b002b" />
890 <public type="string" name="hint" id="0x7f0b002c" />
891 <public type="string" name="manatee" id="0x7f0b002d" />
892 <public type="string" name="myotis" id="0x7f0b002e" />
893 <public type="string" name="password" id="0x7f0b002f" />
894 <public type="string" name="porcupine" id="0x7f0b0030" />
895 <public type="string" name="porpoise" id="0x7f0b0031" />
896 <public type="string" name="search_menu_title" id="0x7f0b0032" />
897 <public type="string" name="skunk" id="0x7f0b0033" />
898 <public type="string" name="status_bar_notification_info_overflow" id="0x7f0b0034" />
899 <public type="string" name="vole" id="0x7f0b0035" />
900 <public type="style" name="AlertDialog.AppCompat" id="0x7f0c0000" />
```

Ta thấy resource id trên tương ứng với resource name password.

Tiếp theo, ta truy cập vào thư mục mà ta đã dịch ngược ở trên, truy cập theo đường dẫn res/values/strings.xml và tìm resource name password. Ta để ý đến chuỗi opossum.

```
strings.xml X
C: > Users > Win 10 > two > res > values > strings.xml
48 <string name="manatee">caribou</string>
49 <string name="myotis">jackrabbit</string>
50 <string name="password">opossum</string>
51 <string name="porcupine">blackbuck</string>
52 <string name="porpoise">mouflon</string>
53 <string name="search_menu_title">Search</string>
54 <string name="skunk">elk</string>
55 <string name="status_bar_notification_info_overflow">999+</string>
56 <string name="vole">beaver</string>
57 </resources>
```

Ta thử quay lại app PicoCTF và nhập thử giá trị opossum vào và bấm nút. Thật may mắn làm sao khi ta nhận được kết quả có chứa flag như trong hình sau:

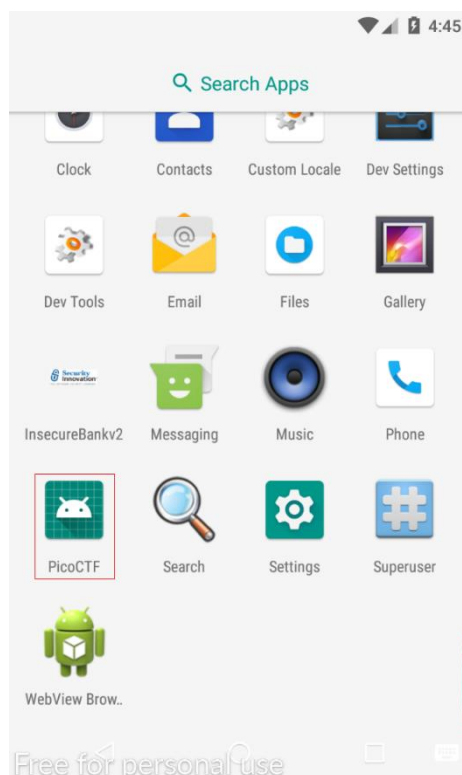


Vậy flag cần tìm của bài này là: `picoCTF{pining.for.the.fjords}`.

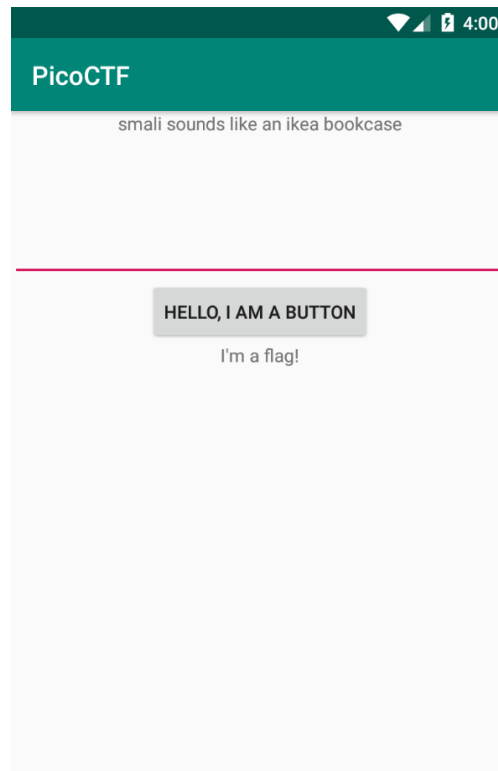
3. Tìm kiếm và lấy flag. Bạn có thể tìm thấy tại: `three.apk`

Ta tiến hành cài đặt app `three.apk` sử dụng câu lệnh `adb install three.apk`.

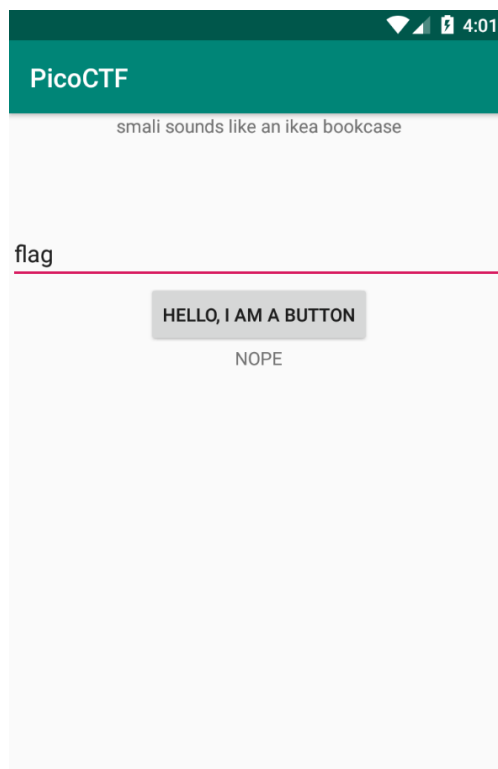
Sau đó, ta mở điện thoại ảo lên và mở app PicoCTF:



Giao diện của app PicoCTF sẽ như hình sau:



Ta nhập thử một giá trị bất kì (ví dụ là `flag`) và bấm nút **HELLO, I AM A BUTTON** thử.



Như ta thấy, dòng chữ `I'm a flag!` ở phía dưới nút bấm bị chuyển thành NOPE.

Giờ ta mở Bytecode Viewer lên và kéo thả tập tin `three.apk` vào. Sau đó, ta truy cập vào đường dẫn `com/helloctmu/picoctf/MainActivity.class`. Ta để ý dòng code từ dòng 21 đến dòng 24 trong tập tin `MainActivity.class`:

```
21 public void buttonClick(final View view) {  
22     this.text_bottom.setText((CharSequence)FlagstaffHill.getFlag(this.text_input.getText().toString(), this.ctx));  
23 }  
24
```

Hàm `buttonClick()` sẽ thực hiện công việc thay đổi của dòng chữ bên dưới nút bấm khi ta tiến hành bấm nút thông qua hàm `getFlag()` trong tập tin `FlagstaffHill.class`.

Ta mở tập tin `FlagstaffHill.class` và xem code của hàm `getFlag()` từ dòng 11 đến dòng 21.

```
11 public static String getFlag(final String s, final Context context) {  
12     final String[] array = { "weatherwax", "ogg", "garlick", "nitt", "aching", "dismass" };  
13     final int n = 3 - 3;  
14     final int n2 = 3 / 3 + n;  
15     final int n3 = n2 + n2 - n;  
16     final int n4 = 3 + n3;  
17     if (s.equals("").concat(array[n4]).concat(".").concat(array[n2]).concat(".").concat(array[n]).c  
18         return sesame(s);  
19     }  
20     return "NOPE";  
21 }  
22  
23 public static native String sesame(final String p0);  
24 }
```

Hàm `getFlag()` sẽ kiểm tra input nhập vào với giá trị của chuỗi `""` sau khi concat một đồng tại dòng 17. Nếu đúng thì sẽ cho ta biết giá trị của flag. Do đó, ta sẽ chạy thử code trên và chỉnh sửa lại một ít để in ra giá trị của chuỗi `""` sau khi concat một đồng. Ta sử dụng online compiler Ideone để chạy code Java:

[edit](#) [fork](#) [download](#)[copy](#)

```
1. import java.util.*;
2.
3. public class Main
4. {
5.     public static String getFlag() {
6.         final String s = "";
7.         final String[] array = { "weatherwax", "ogg", "garlick", "nitt", "aching", "dismiss" };
8.         final int n = 3 - 3;
9.         final int n2 = 3 / 3 + n;
10.        final int n3 = n2 + n2 - n;
11.        final int n4 = 3 + n3;
12.        return "".concat(array[n4]).concat(".").concat(array[n2]).concat(".").concat(array[n]).concat(".").
concat(array[n4 + n - n2]).concat(".").concat(array[3]).concat(".").concat(array[n3]);
13.    }
14.
15.    public static void main(String []args) {
16.        System.out.println(getFlag());
17.    }
18. }
```

Success #stdin #stdout 0.06s 51000KB

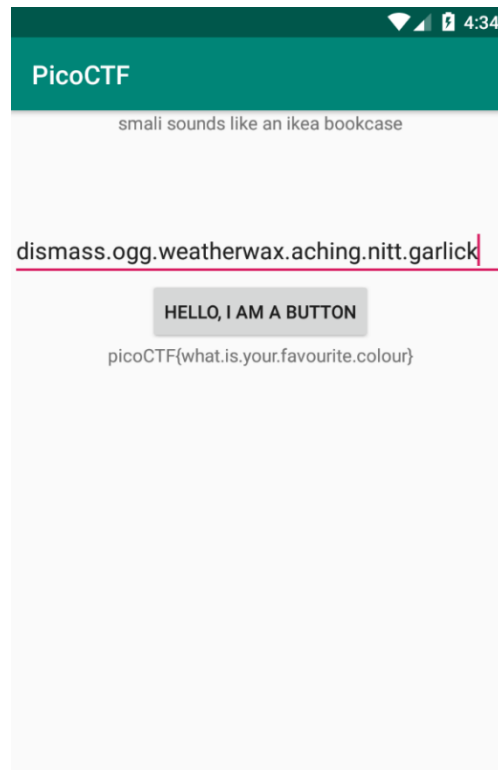
[comments \(0\)](#)[stdin](#)[copy](#)

Standard input is empty

[stdout](#)[copy](#)

dismiss.ogg.weatherwax.aching.nitt.garlick

Có được chuỗi này, ta quay lại app PicoCTF và nhập chuỗi này vào. Thật may mắn làm sao khi ta nhận được kết quả có chứa flag như trong hình sau:

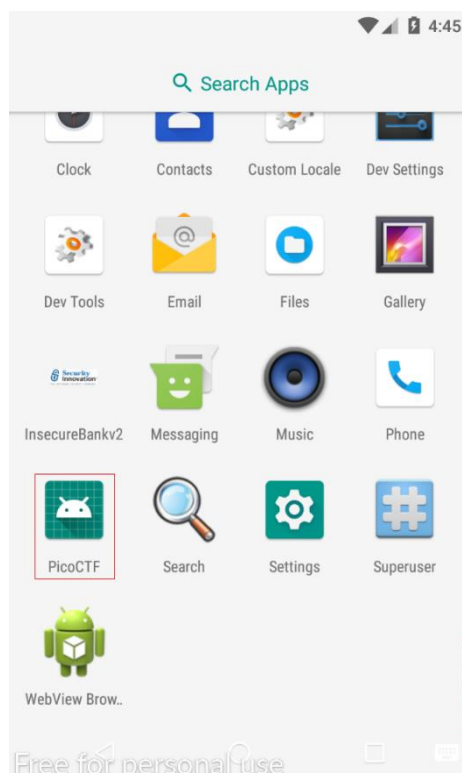


Vậy flag cần tìm của bài này là: `picoCTF{what.is.your.favourite.colour}`.

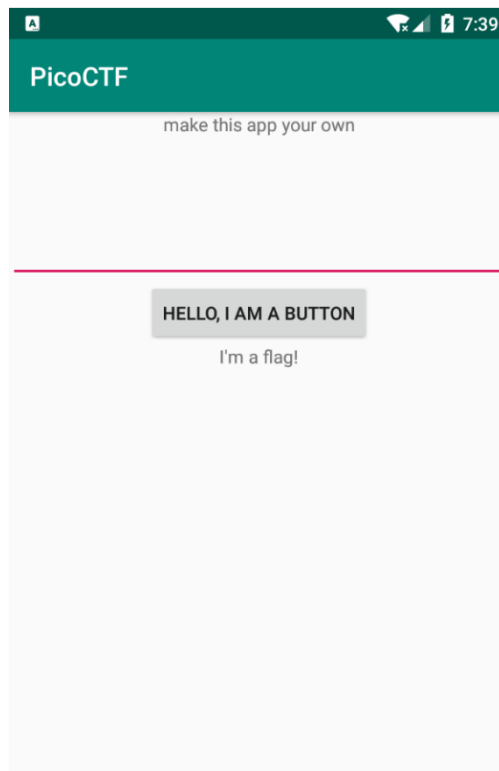
4. Dịch ngược, vá lại tập tin và lấy cờ. Bạn có thể tìm thấy tại: `four.apk`

Ta tiến hành cài đặt app `four.apk` sử dụng câu lệnh `adb install four.apk`.

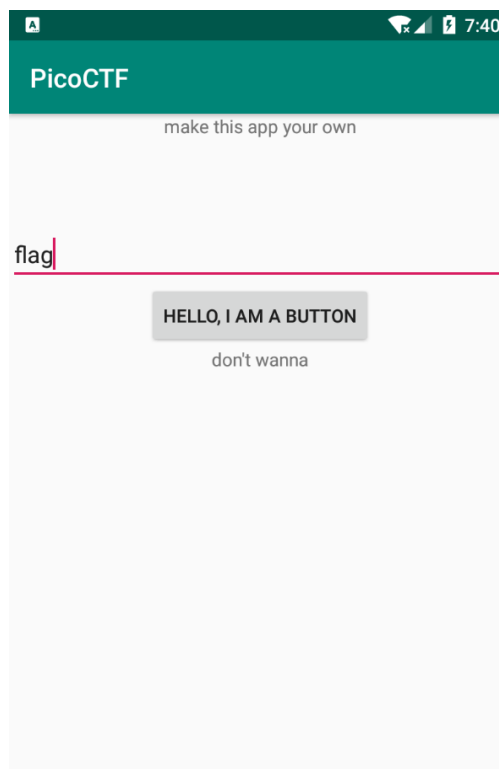
Sau đó, ta mở điện thoại ảo lên và mở app PicoCTF:



Giao diện của app PicoCTF sẽ như hình sau:



Ta nhập thử một giá trị bất kì (ví dụ là `flag`) và bấm nút **HELLO, I AM A BUTTON** thử.



Như ta thấy, dòng chữ `I'm a flag!` ở phía dưới nút bấm bị chuyển thành `don't wanna`.

Giờ ta mở Bytecode Viewer lên và kéo thả tập tin four.apk vào. Sau đó, ta truy cập vào đường dẫn com/helloctmu/picoctf/FlagstaffHill.class. Ta để ý dòng code từ dòng 11 đến dòng 23 trong tập tin FlagstaffHill.class:

```
11      public static native String cilantro(final String p0);
12
13      public static String getFlag(final String s, final Context context) {
14          return nope(s);
15      }
16
17      public static String nope(final String s) {
18          return "don't wanna";
19      }
20
21      public static String yep(final String s) {
22          return cilantro(s);
23      }
```

Hàm getFlag() chỉ trả về kết quả của hàm nope(). Mà cái chúng ta cần ở đây là hàm getFlag() phải return kết quả của hàm yep(). Để làm được điều đó chúng ta cần dịch ngược và vá lại tập tin sử dụng apktool. Trước tiên, ta tiến hành dịch ngược tập tin apk.

```
D:\Virtual Machine\platform-tools>apktool d four.apk
I: Using Apktool 2.5.0 on four.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Win 10\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Sau đó, ta vào lại tập tin FlagstaffHill.class trong đường dẫn smali\com\helloctmu\picoctf để chỉnh sửa lại hàm getFlag(), thay hàm return từ nope() thành yep() như hình dưới.

```

FlagstaffHill.smali X
D: > Virtual Machine > platform-tools > four > smali > com > hellocmu > picocft > FlagstaffHill.smali
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/picocft/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

```

Tiếp theo, ta vá lại tập tin cũng sử dụng apktool.

```

D:\Virtual Machine\platform-tools>apktool b four -o four_1.apk
I: Using Apktool 2.5.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...

```

Android yêu cầu các tập tin apk đều phải được ký bằng một chứng chỉ trước khi được phép cài đặt trên thiết bị. Sau khi chỉnh sửa, tập tin apk sẽ không còn toàn vẹn như ban đầu nên cần phải được ký lại. Ta tiến hành tạo keystore và dùng công cụ apksigner.

```

C:\Program Files\Java\jdk-16.0.1\bin>keytool -genkeypair -v -keystore key.keystore -alias publishingdoc -keyalg RSA -key
size 2048 -validity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: 123456
What is the name of your organizational unit?
[Unknown]: 123456
What is the name of your organization?
[Unknown]: 123456
What is the name of your City or Locality?
[Unknown]: 123456
What is the name of your State or Province?
[Unknown]: 123456
What is the two-letter country code for this unit?
[Unknown]: 123456
Is CN=123456, OU=123456, O=123456, L=123456, ST=123456, C=123456 correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
for: CN=123456, OU=123456, O=123456, L=123456, ST=123456, C=123456
[Storing key.keystore]

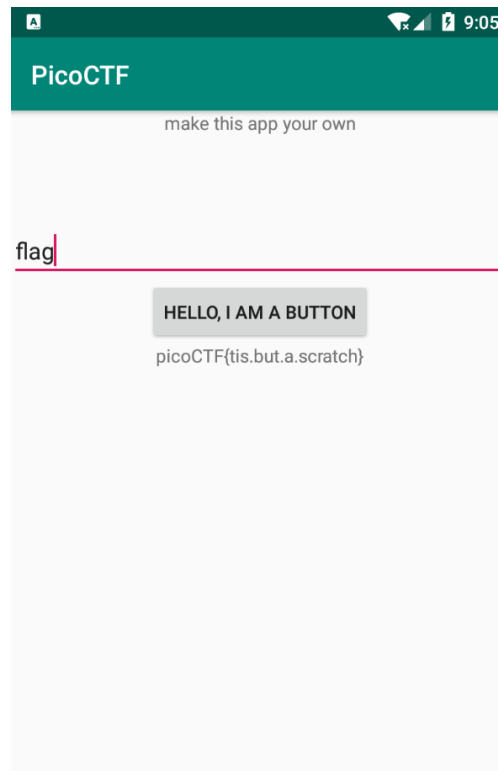
C:\Program Files\Java\jdk-16.0.1\bin>apksigner sign --ks key.keystore four_1.apk
Keystore password for signer #1:

```

Ta tiến hành xóa ứng dụng đã cài đặt trước đó và cài lại tập tin apk mới đã được chỉnh sửa.

```
PS D:\Virtual Machine\platform-tools> .\adb install four_1.apk
Performing Streamed Install
Success
```

Cuối cùng, ta vào app lại, nhập một chuỗi bất kì và nhấn nút **HELLO, I AM A BUTTON**. Kết quả là ta sẽ thấy xuất hiện một chuỗi chứa flag cần tìm phía dưới nút bấm.

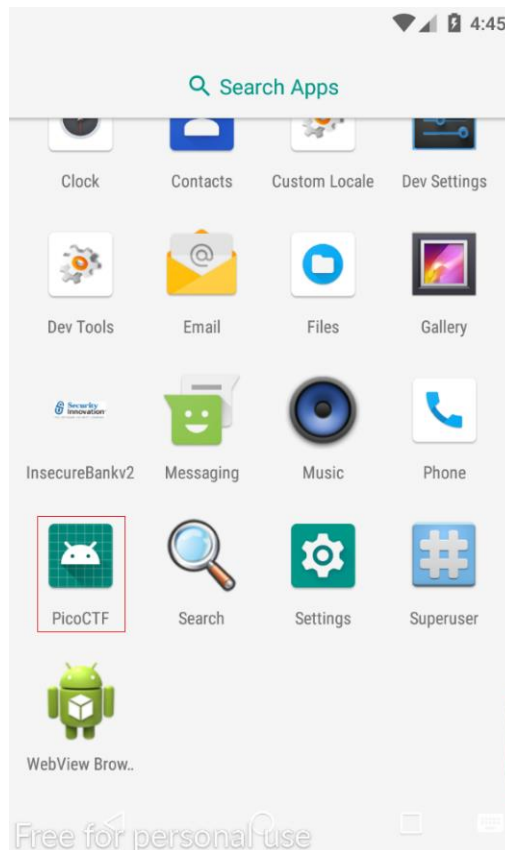


Vậy flag cần tìm của bài này là: `picoCTF{tis.but.a.scratch}`.

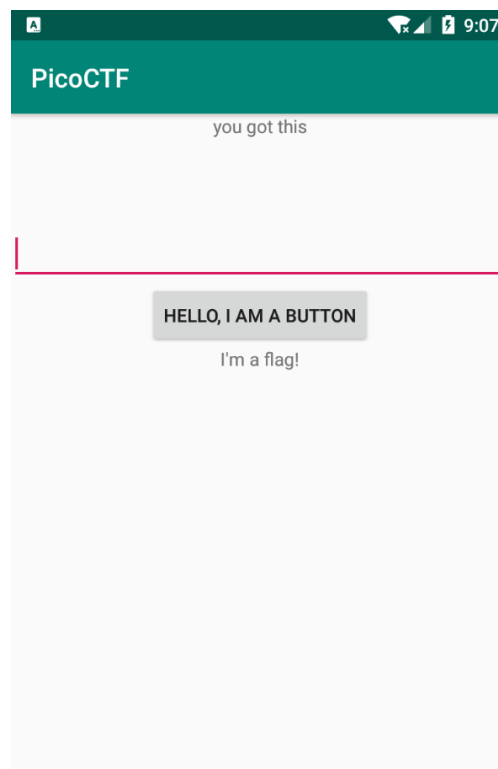
5. Dịch ngược, vá lại tập tin và lấy cờ. Bạn có thể tìm thấy tại: `five.apk`

Ta tiến hành cài đặt app `five.apk` sử dụng câu lệnh `adb install five.apk`.

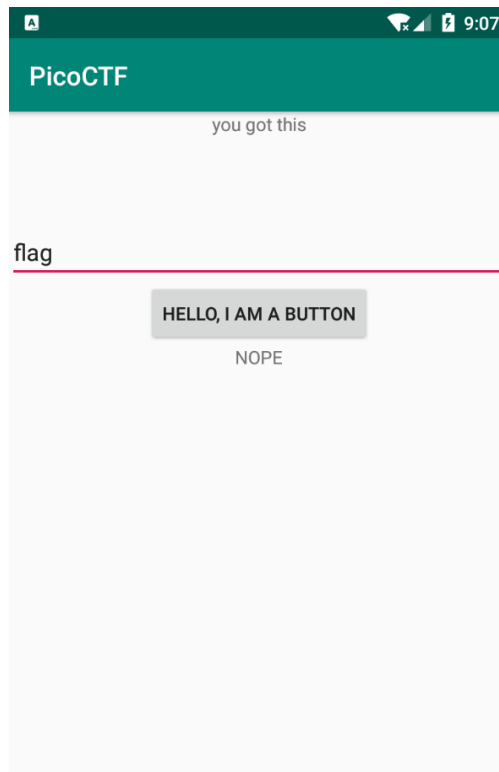
Sau đó, ta mở điện thoại ảo lên và mở app PicoCTF:



Giao diện của app PicoCTF sẽ như hình sau:



Ta nhập thử một giá trị bất kì (ví dụ là `flag`) và bấm nút **HELLO, I AM A BUTTON** thử.



Như ta thấy, dòng chữ I'm a flag! ở phía dưới nút bấm bị chuyển thành NOPE.

Giờ ta mở Bytecode Viewer lên và kéo thả tập tin five.apk vào. Sau đó, ta truy cập vào đường dẫn `com/helloctmu/picoctf/FlagstaffHill.class`. Ta để ý dòng code từ dòng 13 đến dòng 34 trong tập tin `FlagstaffHill.class`:

```

13 public static String getFlag(final String s, final Context context) {
14     final StringBuilder sb = new StringBuilder("aaa");
15     final StringBuilder sb2 = new StringBuilder("aaa");
16     final StringBuilder sb3 = new StringBuilder("aaa");
17     final StringBuilder sb4 = new StringBuilder("aaa");
18     sb.setCharAt(0, (char) (sb.charAt(0) + '\u0004'));
19     sb.setCharAt(1, (char) (sb.charAt(1) + '\u0013'));
20     sb.setCharAt(2, (char) (sb.charAt(2) + '\u0012'));
21     sb2.setCharAt(0, (char) (sb2.charAt(0) + '\u0007'));
22     sb2.setCharAt(1, (char) (sb2.charAt(1) + '\0'));
23     sb2.setCharAt(2, (char) (sb2.charAt(2) + '\u0001'));
24     sb3.setCharAt(0, (char) (sb3.charAt(0) + '\0'));
25     sb3.setCharAt(1, (char) (sb3.charAt(1) + '\u000b'));
26     sb3.setCharAt(2, (char) (sb3.charAt(2) + '\u000f'));
27     sb4.setCharAt(0, (char) (sb4.charAt(0) + '\u000e'));
28     sb4.setCharAt(1, (char) (sb4.charAt(1) + '\u0014'));
29     sb4.setCharAt(2, (char) (sb4.charAt(2) + '\u000f'));
30     if (s.equals("").concat(sb3.toString()).concat(sb2.toString()).concat(sb.toString()).concat(sb4.toString())) {
31         return "call it";
32     }
33     return "NOPE";
34 }

```


Hàm `getFlag()` sẽ kiểm tra input nhập vào với giá trị của chuỗi "" sau khi concat một đồng tại dòng 30. Nếu đúng thì trả về dòng chữ "call it". Do đó, ta sẽ chạy thử code trên và chỉnh sửa lại một ít để in ra giá trị của chuỗi "" sau khi concat một đồng. Ta sử dụng online compiler Ideone để chạy code Java:

```

6.  class Ideone
7.  {
8.      public static String getFlag() {
9.          final StringBuilder sb = new StringBuilder("aaa");
10.         final StringBuilder sb2 = new StringBuilder("aaa");
11.         final StringBuilder sb3 = new StringBuilder("aaa");
12.         final StringBuilder sb4 = new StringBuilder("aaa");
13.         sb.setCharAt(0, (char)(sb.charAt(0) + '\u0004'));
14.         sb.setCharAt(1, (char)(sb.charAt(1) + '\u0013'));
15.         sb.setCharAt(2, (char)(sb.charAt(2) + '\u0012'));
16.         sb2.setCharAt(0, (char)(sb2.charAt(0) + '\u0007'));
17.         sb2.setCharAt(1, (char)(sb2.charAt(1) + '\0'));
18.         sb2.setCharAt(2, (char)(sb2.charAt(2) + '\u0001'));
19.         sb3.setCharAt(0, (char)(sb3.charAt(0) + '\0'));
20.         sb3.setCharAt(1, (char)(sb3.charAt(1) + '\u000b'));
21.         sb3.setCharAt(2, (char)(sb3.charAt(2) + '\u000f'));
22.         sb4.setCharAt(0, (char)(sb4.charAt(0) + '\u000e'));
23.         sb4.setCharAt(1, (char)(sb4.charAt(1) + '\u0014'));
24.         sb4.setCharAt(2, (char)(sb4.charAt(2) + '\u000f'));
25.         return "".concat(sb3.toString()).concat(sb2.toString()).concat(sb.toString()).concat(sb4.toString
    ());
26.     }
27.     public static void main (String[] args)
28.     {
29.         System.out.println(getFlag());
30.     }
31. }

```

Success #stdin #stdout 0.08s 47024KB

 comments (0)

 stdin

 copy

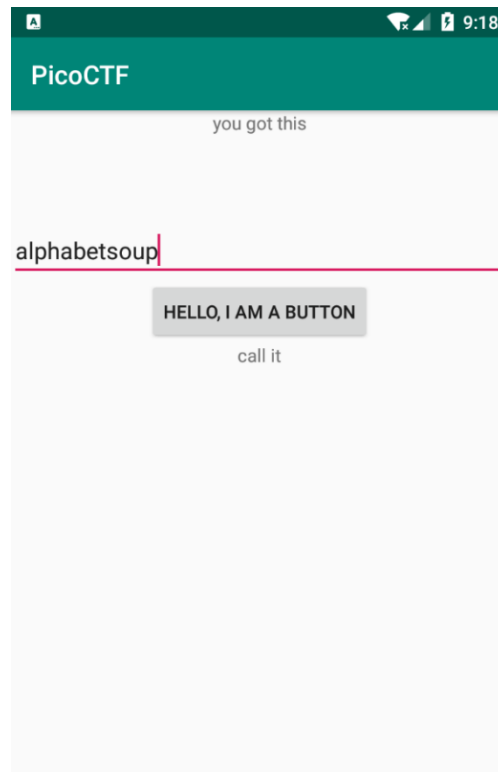
Standard input is empty

 stdout

 copy

alphabetsoup

Có được chuỗi này, ta quay lại app PicoCTF và nhập chuỗi alphabetsoup vào. Kết quả in ra dòng `call it` thay vì flag như trong hình sau:



Ta để ý thấy có một hàm `cardamom()` không được gọi, mặc dù hàm đó cũng nhận input giống hàm `getFlag()`. Do đó, ta thử sửa code smali để hàm `cardamom()` được gọi xem có tìm được flag hay không. Trước tiên, ta cần dịch ngược tập tin apk sử dụng apktool.

```
D:\Virtual Machine\platform-tools>apktool d five.apk
I: Using Apktool 2.5.0 on five.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Win 10\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Sau đó, ta truy cập vào đường dẫn `smali\com\hellocmu\picoctf` và sửa tập tin `FlagstaffHill.class`. Ta thay dòng

```
const-string v5, "call it"
```

thành

```
invoke-static                {p0},                Lcom/hellocmu/picoctf/FlagstaffHill;-
>cardamom(Ljava/lang/String;)Ljava/lang/String;
move-result-object v5
```

Khi gọi thẳng 1 hàm sẽ sử dụng `invoke-static`.

Dù gọi hàm kiểu gì thì sau đó cũng phải return kết quả của hàm hết. Để đưa kết quả return của hàm được gọi vào nơi gọi nó chúng ta sẽ sử dụng move-result-object.

Ta để ý đến lời gọi constructor init() ở ngay đầu tập tin sử dụng {p0} để nhận kết quả return từ hàm init() cho nên ta chọn biến {p0} trong file FlagstaffHill.smali để nhận kết quả return từ hàm.

```
FlagstaffHill.smali ×
D: > Virtual Machine > platform-tools > five > smali > com > hellocmu > picocft > FlagstaffHill.smali
229
230     move-result v5
231
232     if-eqz v5, :cond_0
233
234     invoke-static {p0}, Lcom/hellocmu/picocft/FlagstaffHill;->cardamom(Ljava/lang/String;)Ljava/lang/String;
235
236     move-result-object v5
237
238     return-object v5
239
240     .line 37
241     :cond_0
242     const-string v5, "NOPE"
243
244     return-object v5
245 .end method
```

Tiếp theo, ta vá lại tập tin cũng sử dụng apktool.

```
D:\Virtual Machine\platform-tools>apktool b five -o five_1.apk
I: Using Apktool 2.5.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
```

Android yêu cầu các tập tin apk đều phải được ký bằng một chứng chỉ trước khi được phép cài đặt trên thiết bị. Sau khi chỉnh sửa, tập tin apk sẽ không còn toàn vẹn như ban đầu nên cần phải được ký lại. Ta tiến hành tạo keystore và dùng công cụ apksigner. Ta có thể tận dụng keystore ở bài 2.4 mà không cần phải tạo keystore mới).

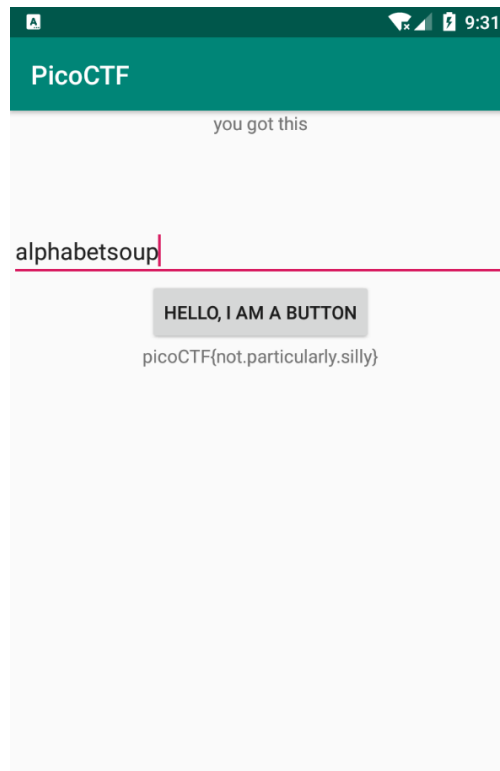
```
C:\Program Files\Java\jdk-16.0.1\bin>keytool -genkeypair -v -keystore key.keystore -alias publishingdoc -keyalg RSA -keysize 2048 -validity 10000
Enter keystore password:
keytool error: java.lang.Exception: Key pair not generated, alias <publishingdoc> already exists
java.lang.Exception: Key pair not generated, alias <publishingdoc> already exists
    at java.base/sun.security.tools.keytool.Main.doGenKeyPair(Main.java:1874)
    at java.base/sun.security.tools.keytool.Main.doCommands(Main.java:1152)
    at java.base/sun.security.tools.keytool.Main.run(Main.java:411)
    at java.base/sun.security.tools.keytool.Main.main(Main.java:404)

C:\Program Files\Java\jdk-16.0.1\bin>apksigner sign --ks key.keystore five_1.apk
Keystore password for signer #1:
```

Ta tiến hành xóa ứng dụng đã cài đặt trước đó và cài lại tập tin apk mới đã được chỉnh sửa.

```
PS D:\Virtual Machine\platform-tools> .\adb install five_1.apk
Performing Streamed Install
Success
```

Cuối cùng, ta vào app lại, nhập chuỗi alphabetsoup và nhấn nút **HELLO, I AM A BUTTON**. Kết quả là ta sẽ thấy xuất hiện một chuỗi chứa flag cần tìm phía dưới nút bấm.



Vậy flag cần tìm của bài này là: `picoCTF{not.particularly.silly}`.

Ta cũng có thể sửa code trong tập tin apk mà ta nhập gì (không cần phải nhập đúng chuỗi trên) vẫn trả về kết quả là flag của bài.

--- Hết ---