BÁO CÁO BÀI TẬP

**Môn học: Bảo mật Web và Ứng dụng**

**Kỳ báo cáo: Buổi 04 (Session 02)**

**Tên chủ đề: Pentesting Android Applications**

*GV: Nghi Hoàng Khoa*

*Ngày báo cáo: 22/05/2023*

**Nhóm: mét xi**

**1.THÔNG TIN CHUNG:**

Lớp: NT213.N21.ATCL

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Trương Văn Rồng | 20521831 | 20521831@gm.uit.edu.vn |
| 2 | Nguyễn Tấn Tài | 20521859 | 20521859@gm.uit.edu.vn |

**2.NỘI DUNG THỰC HIỆN:[[1]](#footnote-0)**

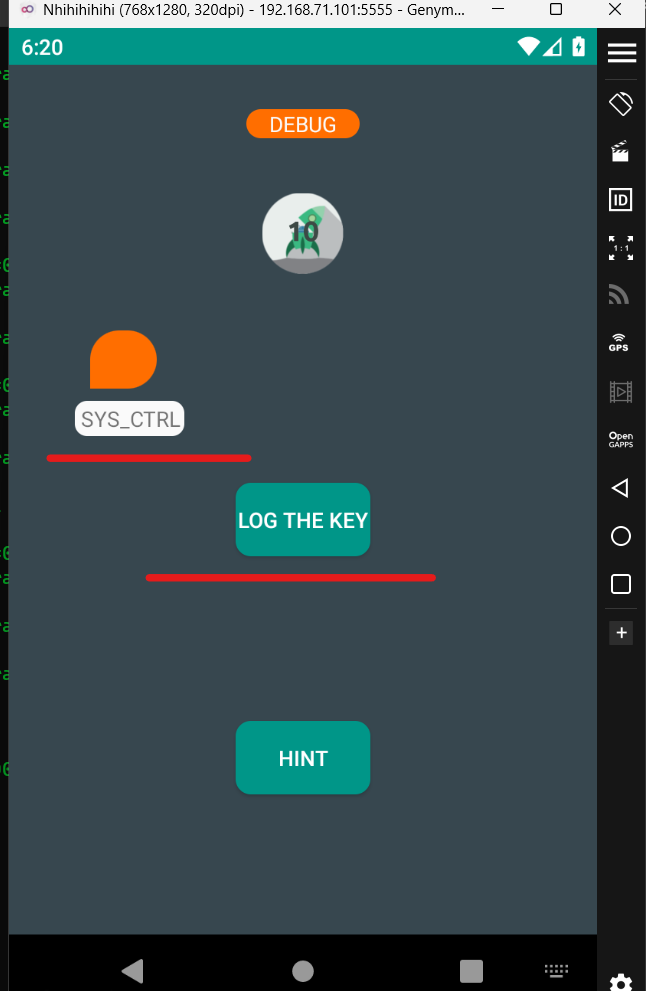
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** | **Người đóng góp** |
| 1 | EVABSv5 | **100%** | Rồng |
| 2 | Droid | **100%** | Tài |

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

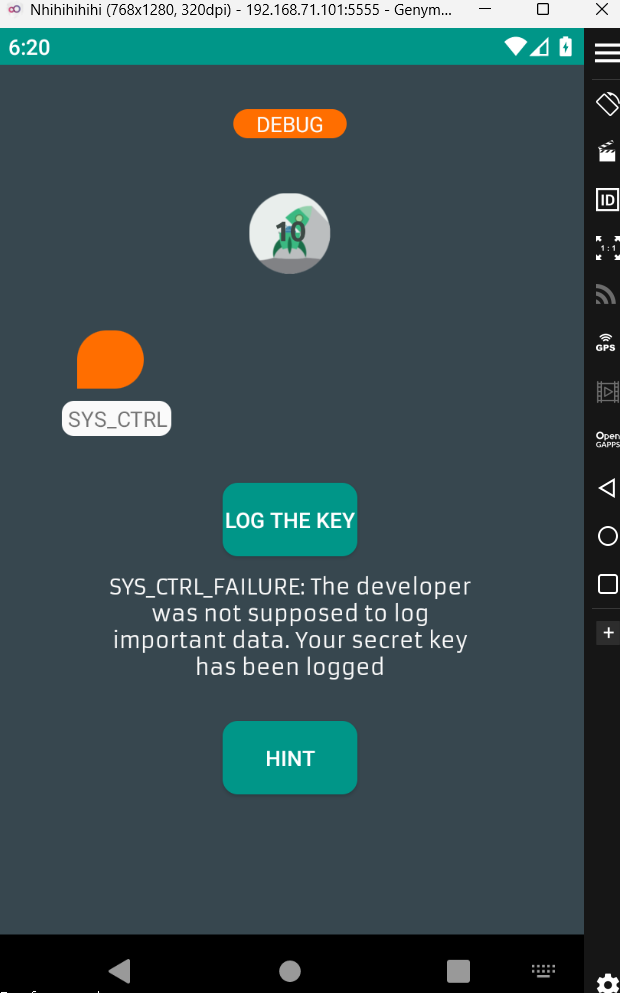
BÁO CÁO CHI TIẾT

* **EVABSv5.apk**

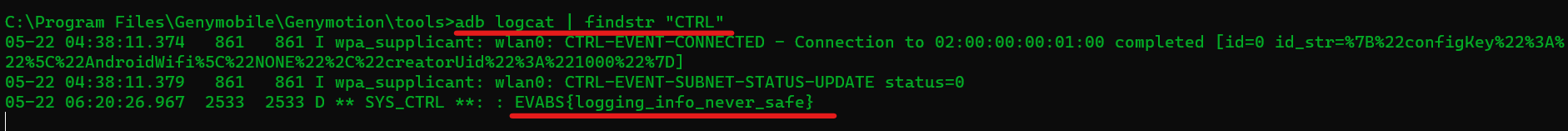
**Level 1 - Debug Me**



* Như hint khi ta ấn button sẽ ghi lại log

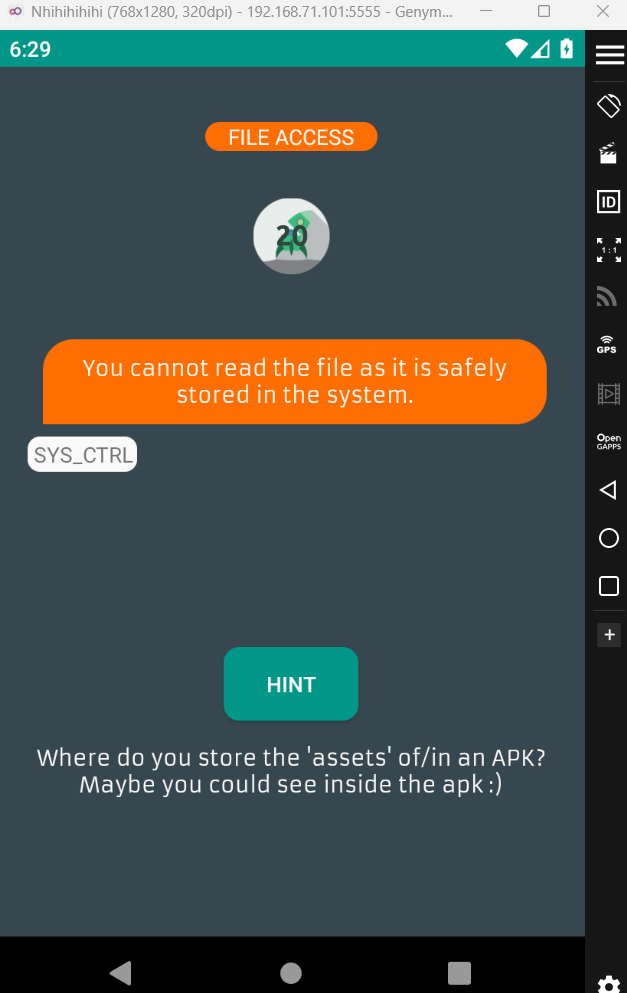


* Lúc này đơn giản ta chỉ cần đọc log bằng lệnh logcat và tìm string tương ứng

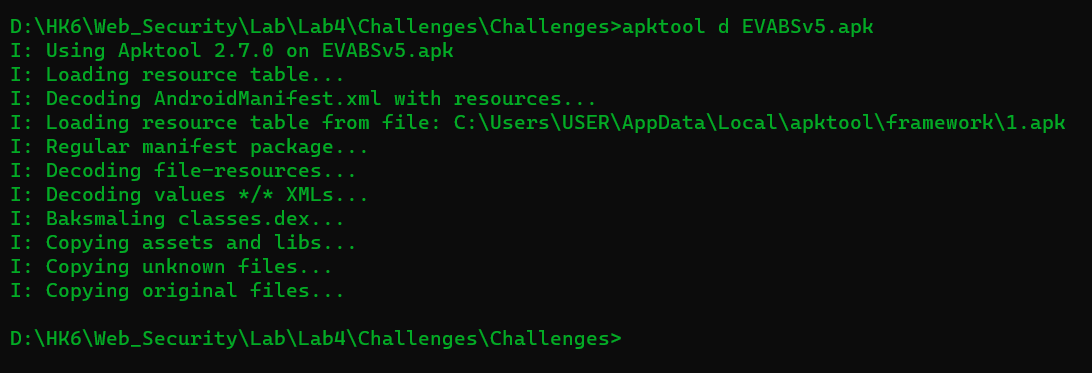


* EVABS{logging\_info\_never\_safe}

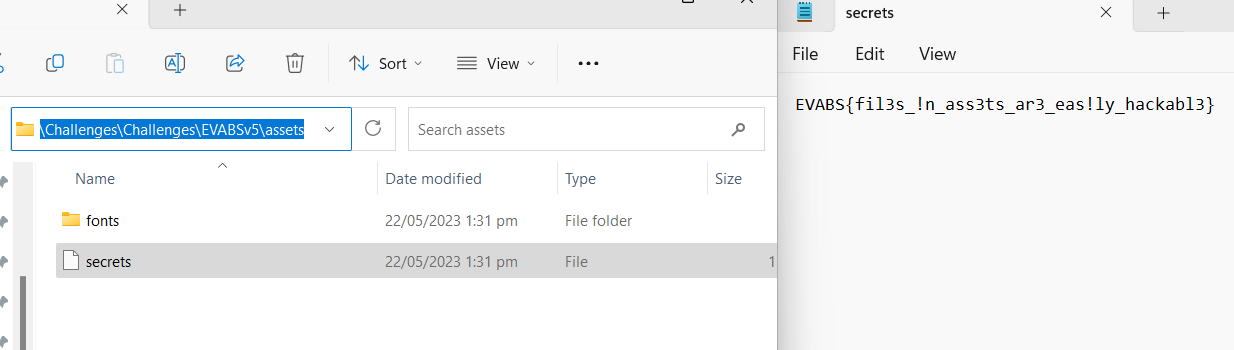
**Level 2 - File Access**



* Ở trong lab này ta được hint rằng tìm trong thư mục assests
* Đơn giản ta decompile file APK nhằm lấy thông resources như AndroidManifest.xml

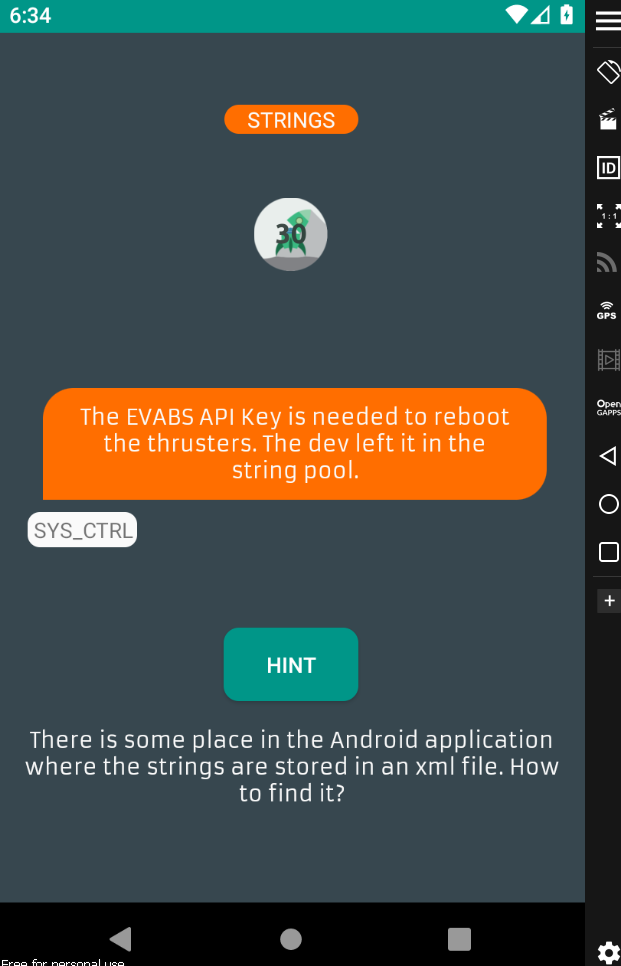


* Đến đây ta tìm assets và flag thôi

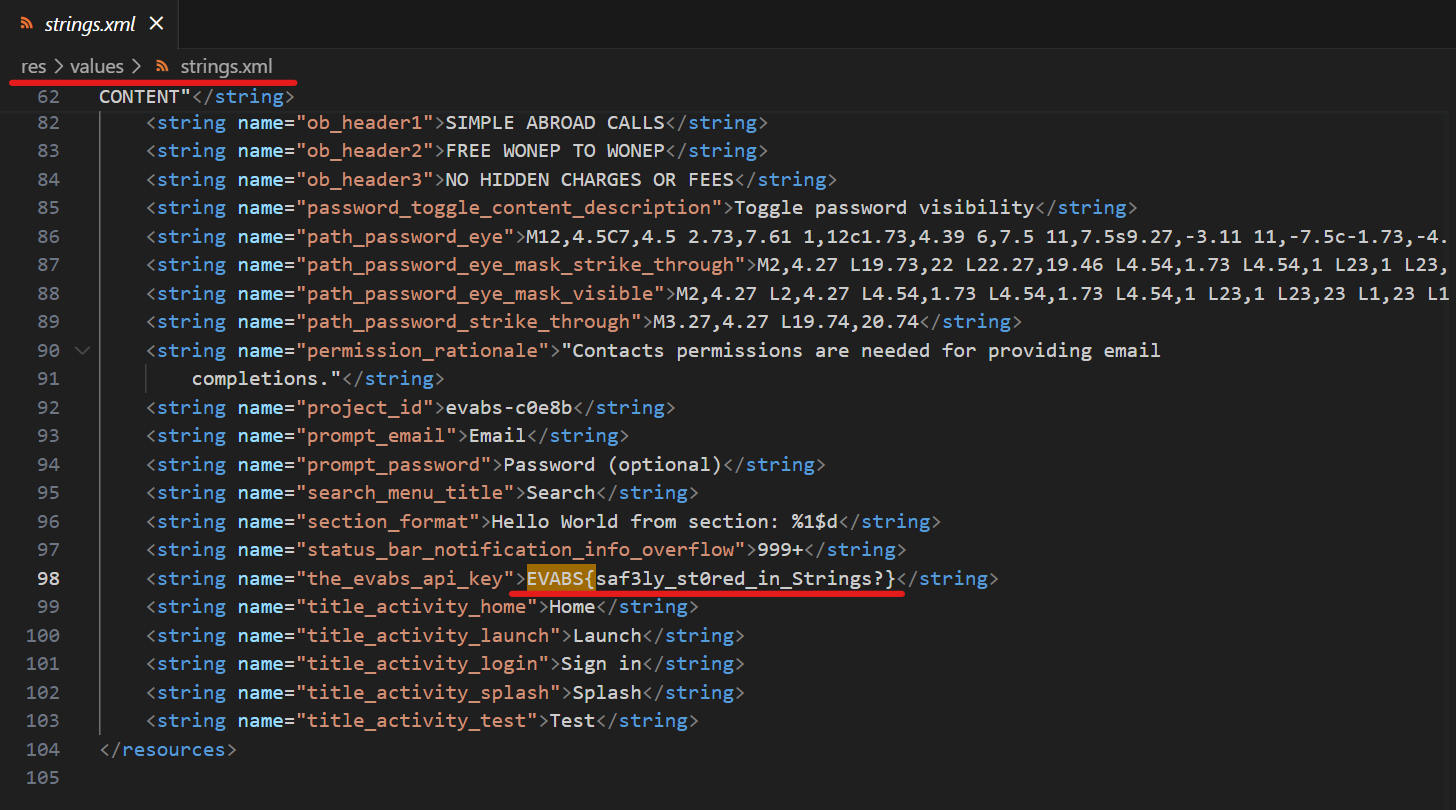


* EVABS{fil3s\_!n\_ass3ts\_ar3\_eas!ly\_hackabl3}

**Level 3: Strings**

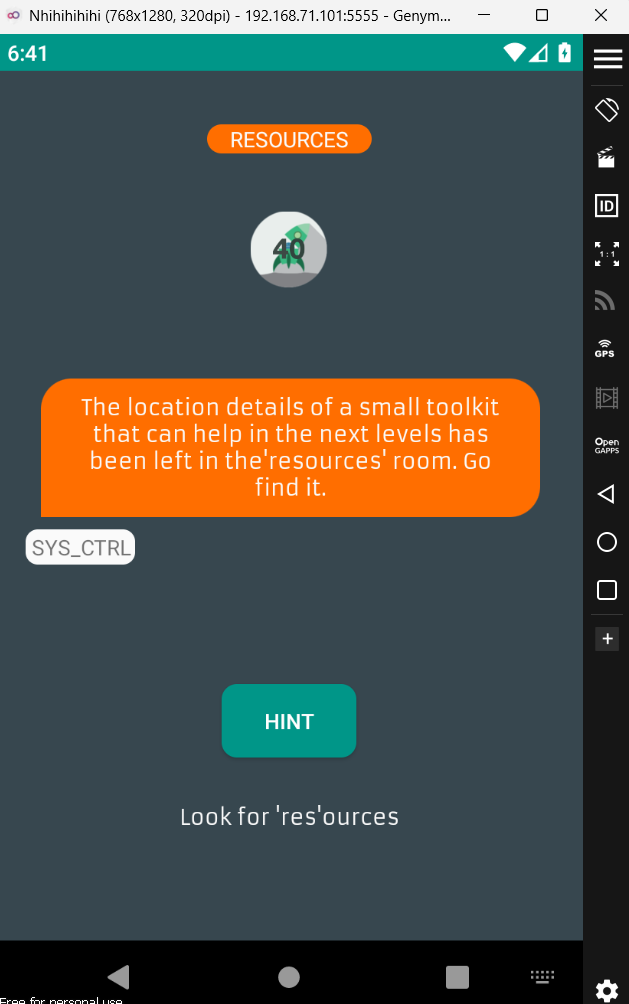


* Tương tự chalenge ta tìm trong các file strings như hint
* Search ngay có flag

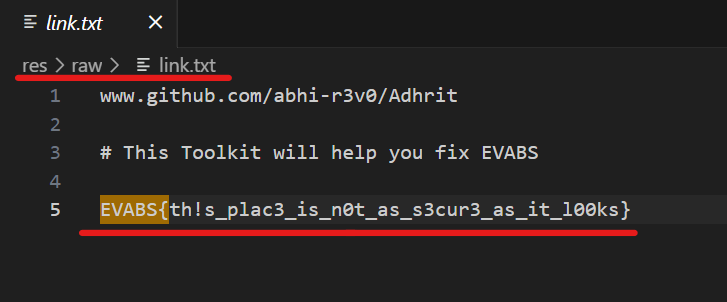


* EVABS{saf3ly\_st0red\_in\_Strings?}

**Level 4: Resource**

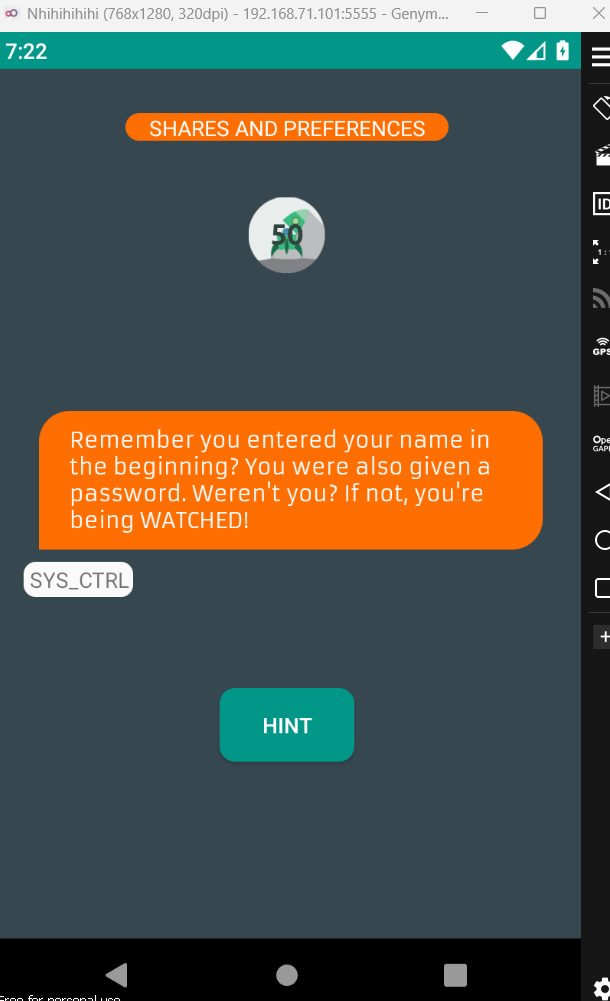


* Lab này ta cần tìm ở thư mục resource
* Thấy ngay ở link.txt

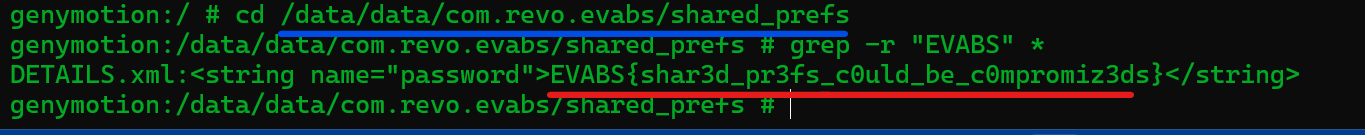


* EVABS{th!s\_plac3\_is\_n0t\_as\_s3cur3\_as\_it\_l00ks}

**Level 5: Shares and Preferences**



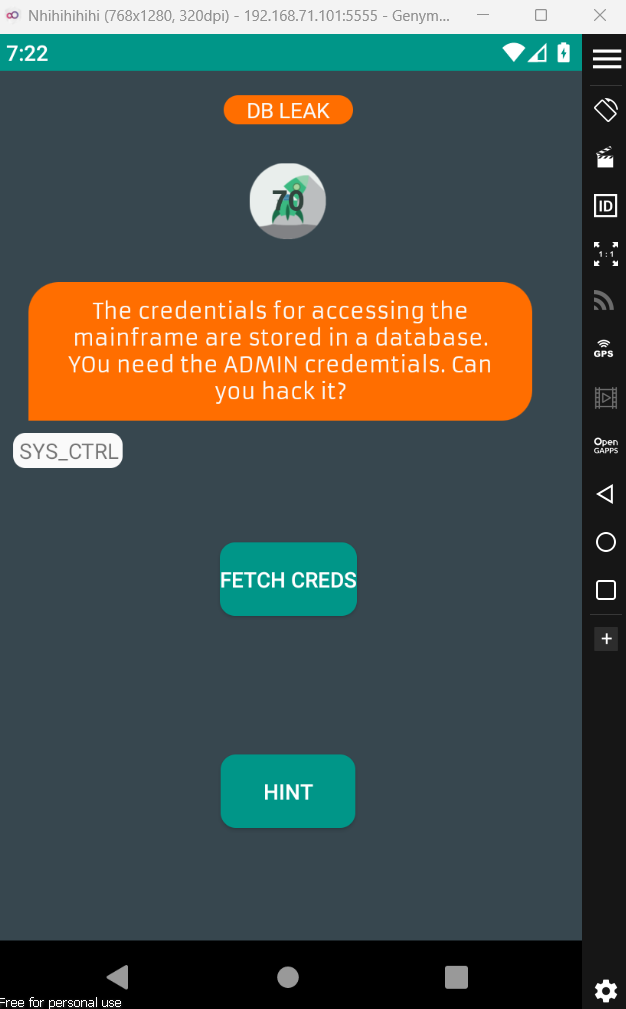
* SharedPreferences là một cơ chế lưu trữ dữ liệu đơn giản và dễ sử dụng trong Android. Nó cho phép bạn lưu trữ và truy xuất dữ liệu trong các cặp giá trị (key-value).
* Di chuyển đến thư mục chứa các file SharedPreferences **com.revo.evabs**
* Đồng thời tìm kiếm chuỗi **EVABS** trong các file xml



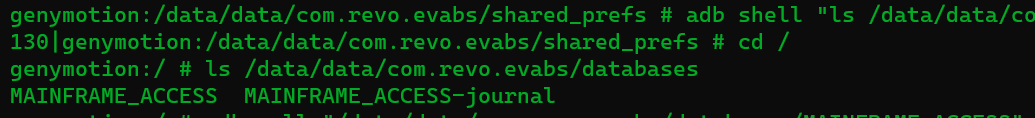
* EVABS{shar3d\_pr3fs\_c0uld\_be\_c0mpromiz3ds}

**Level 6 - DB leak**

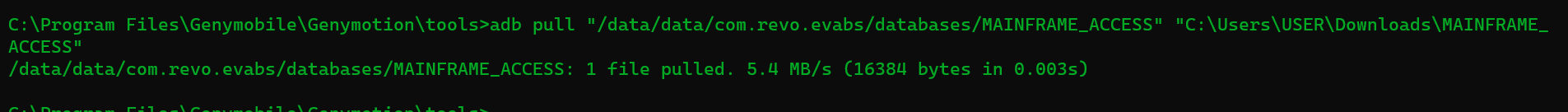
* Ở trong môi trường Android, các ứng dụng thường cần lưu trữ dữ liệu trên thiết bị. Một trong những cách phổ biến để lưu trữ dữ liệu là sử dụng cơ sở dữ liệu SQLite. Thường được lưu tại /data/data/<package-name>/databases



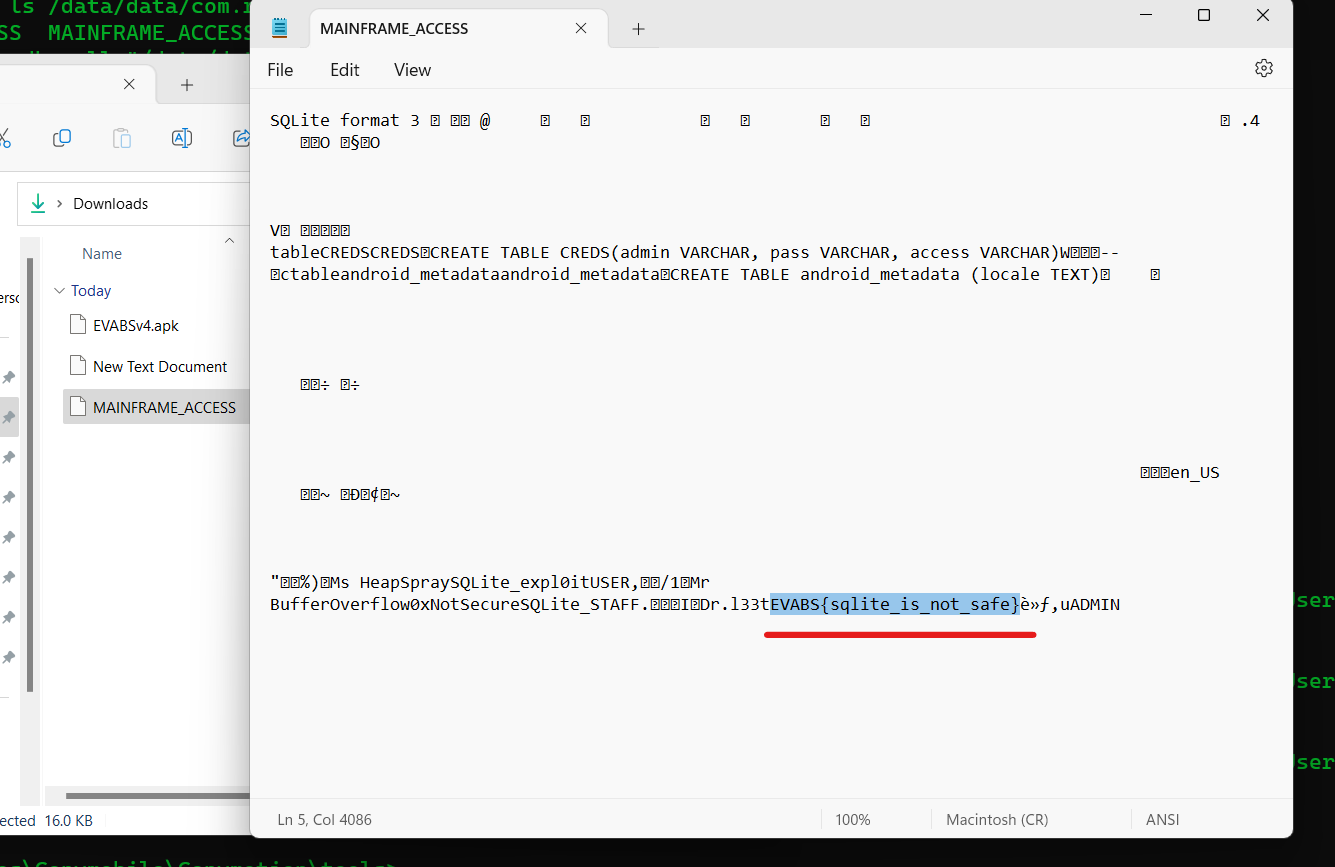
* Ta tìm thử các database hiện có và thử từng cái



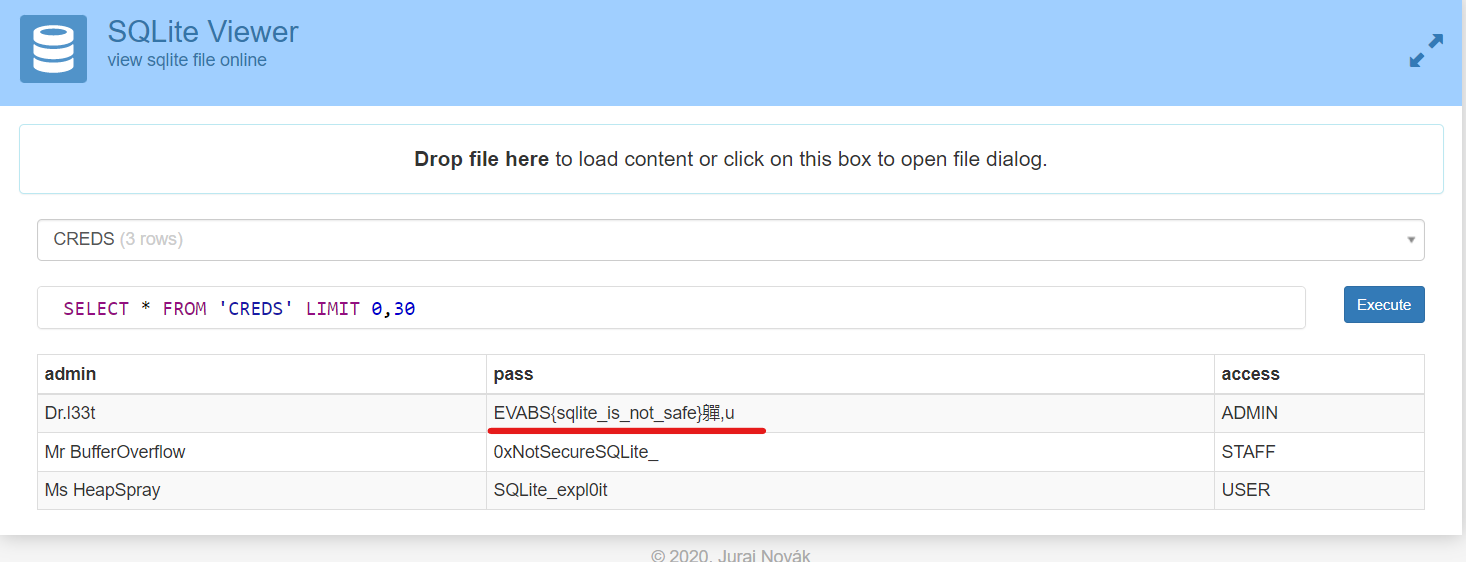
* Pull về máy thật để đọc



* Có thể dễ dàng thấy flag ở file note pad

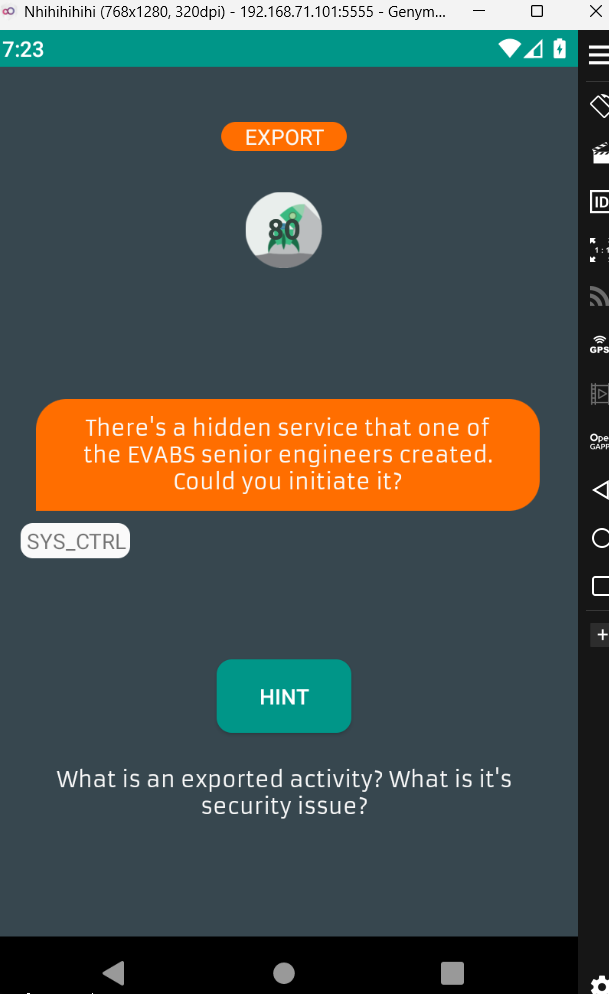


* Rõ hơn có thể dùng tool online để xem sqlite

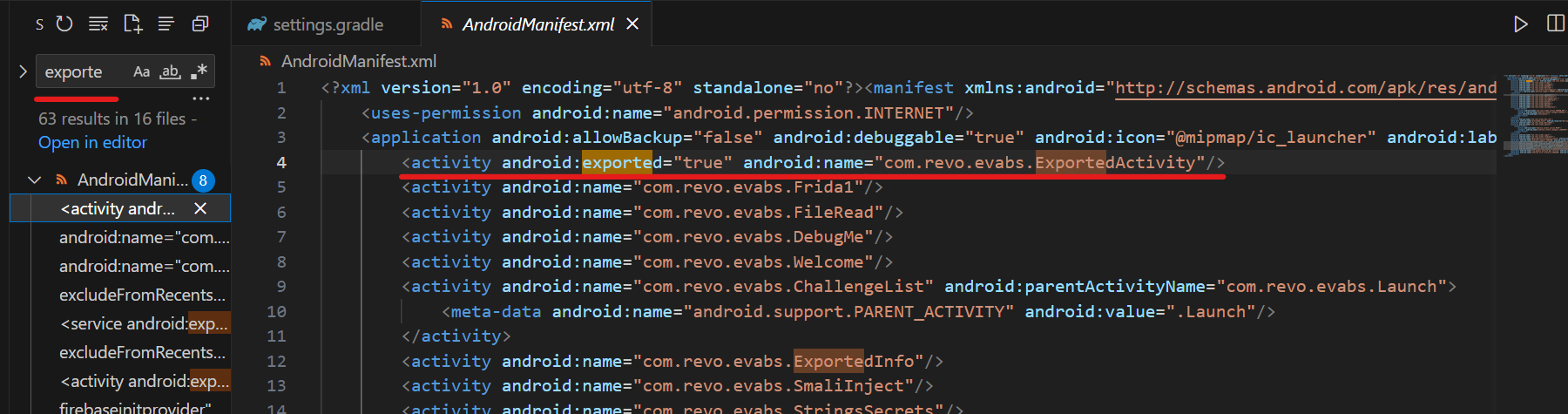


* EVABS{sqlite\_is\_not\_safe}

**Level 7 - Export**



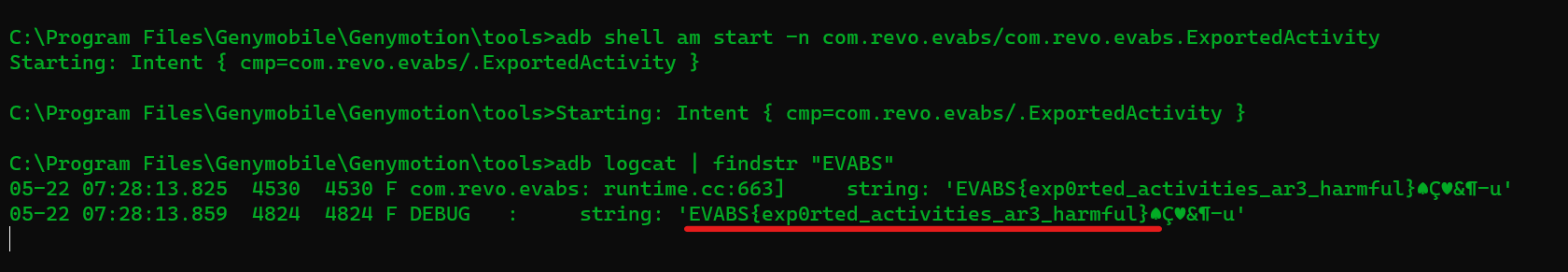
* Activity trong Android là một thành phần cốt lõi trong ứng dụng, nơi diễn ra các hoạt động tương tác với người dùng
* Sau khi kiểm tra hàm main ta thấy có activityandroid:exported="true"



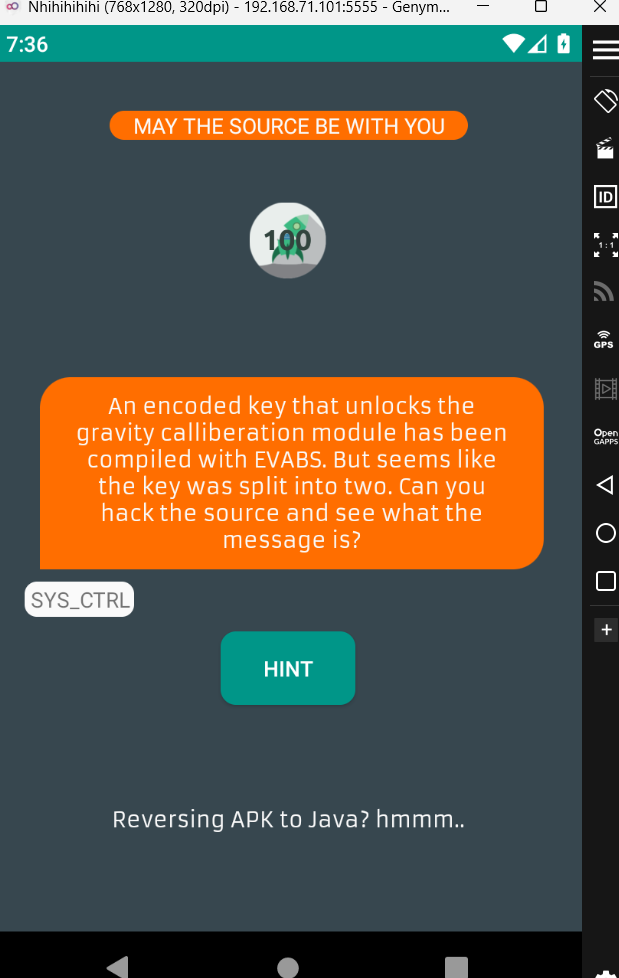
* Ta có thể trigger lại activity này bằng command sau:

adb shell am start -n [package name]/[package name].[exported activity]

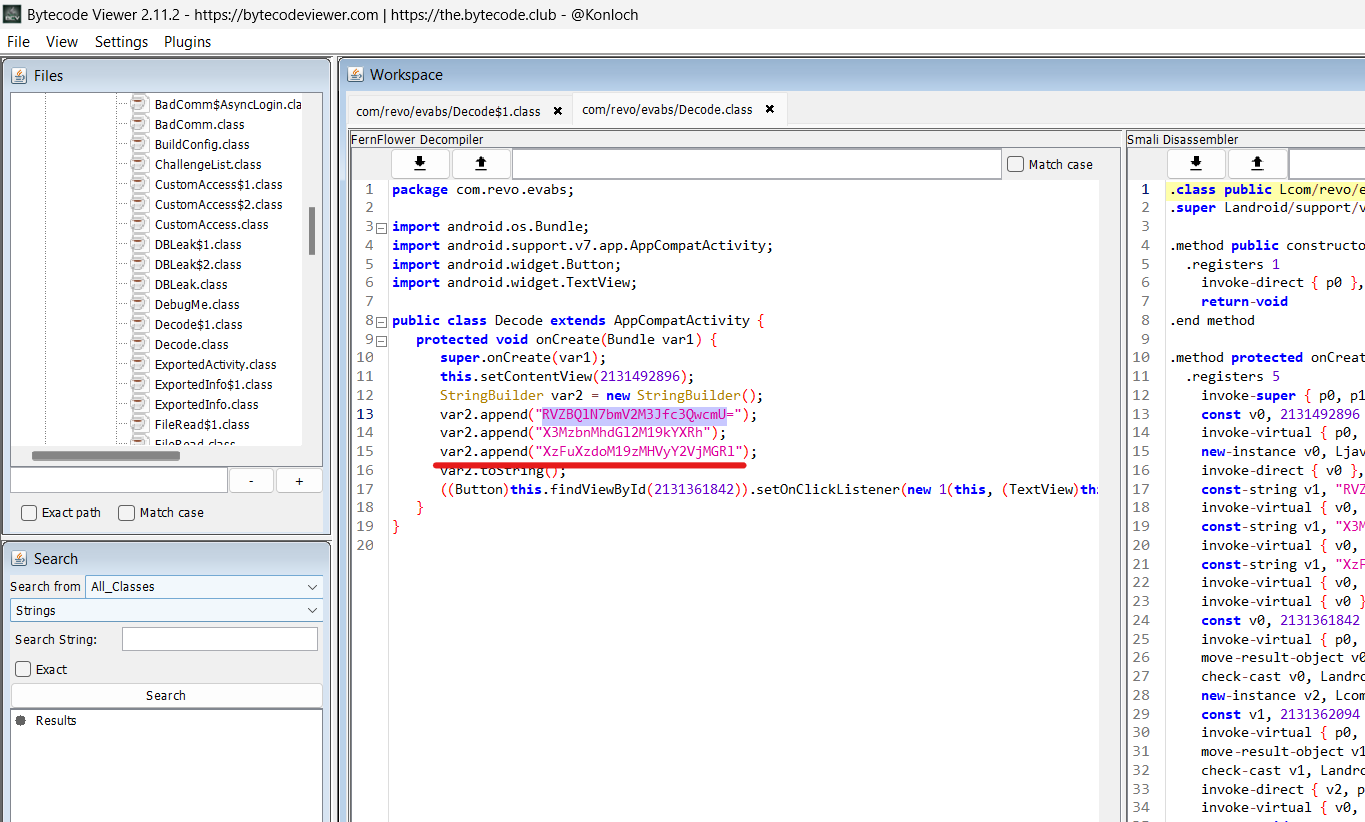
* Cuối cùng check log và tìm flag



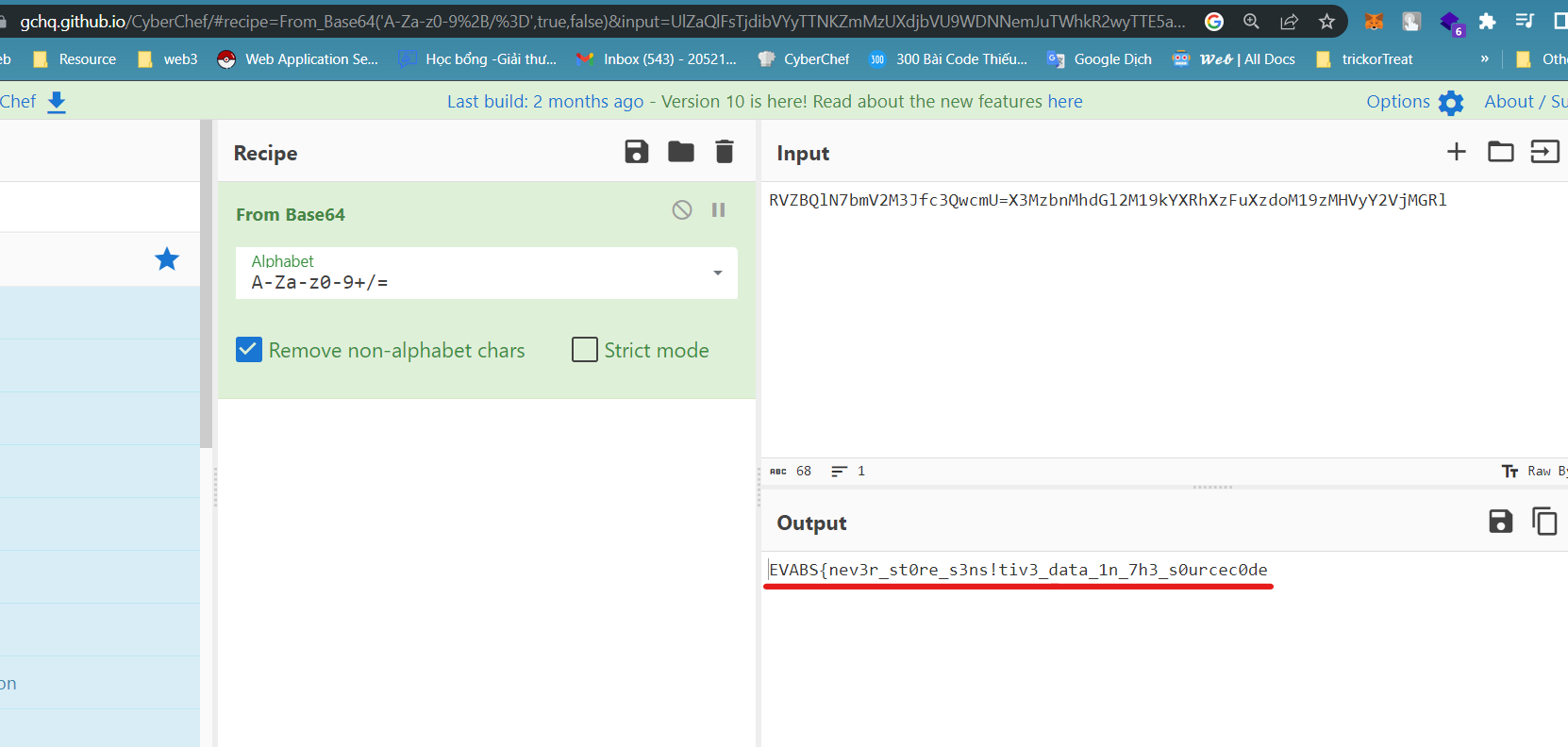
* EVABS{exp0rted\_activities\_ar3\_harmful}
* **Level 8 - Decode**



* Đọc hint ta nghĩ ngay đến reverse để đọc các class, ta decompile, ở Decode.class:

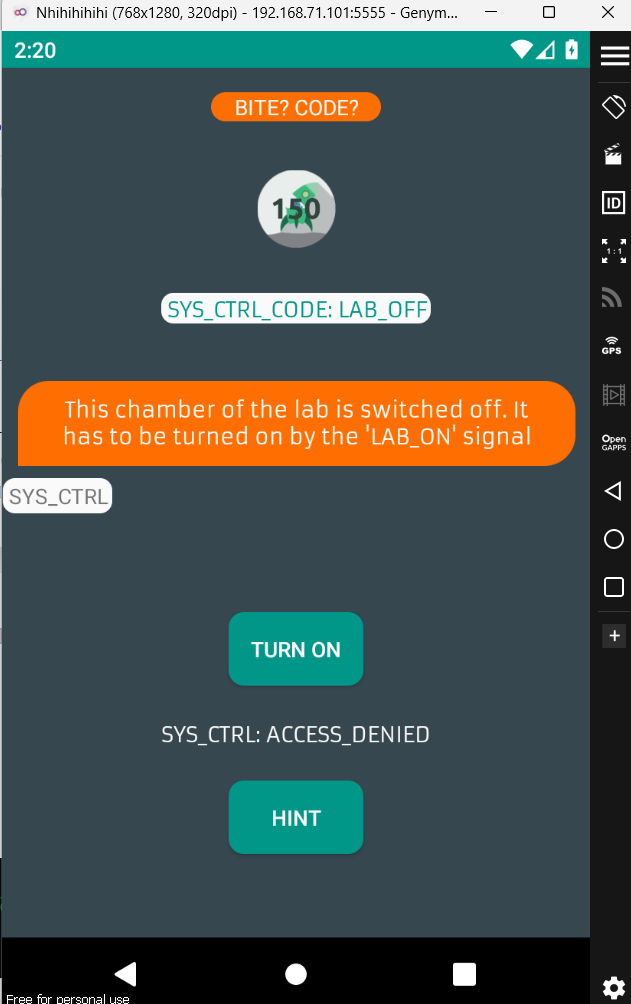


* Thấy rõ base64 và append các string lại để decode

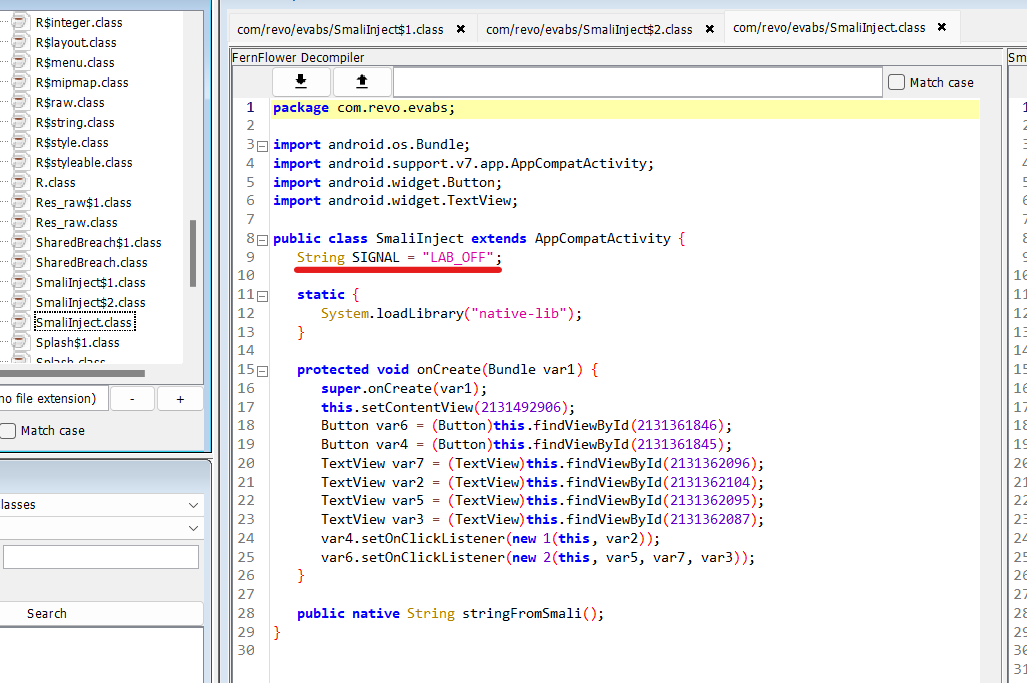


* EVABS{nev3r\_st0re\_s3ns!tiv3\_data\_1n\_7h3\_s0urcec0de}

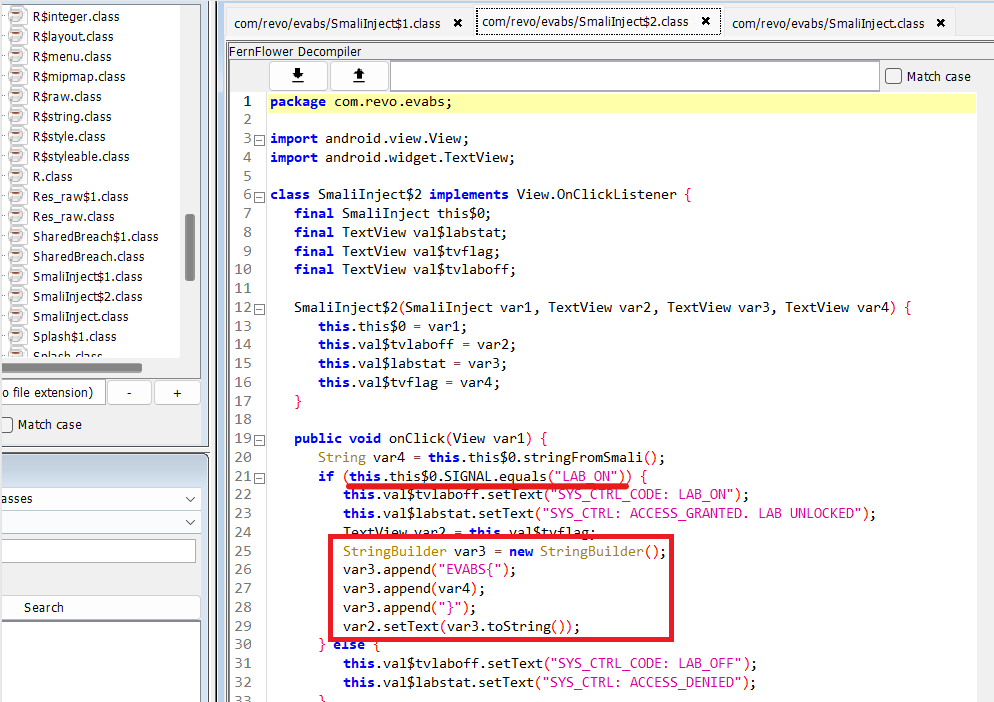
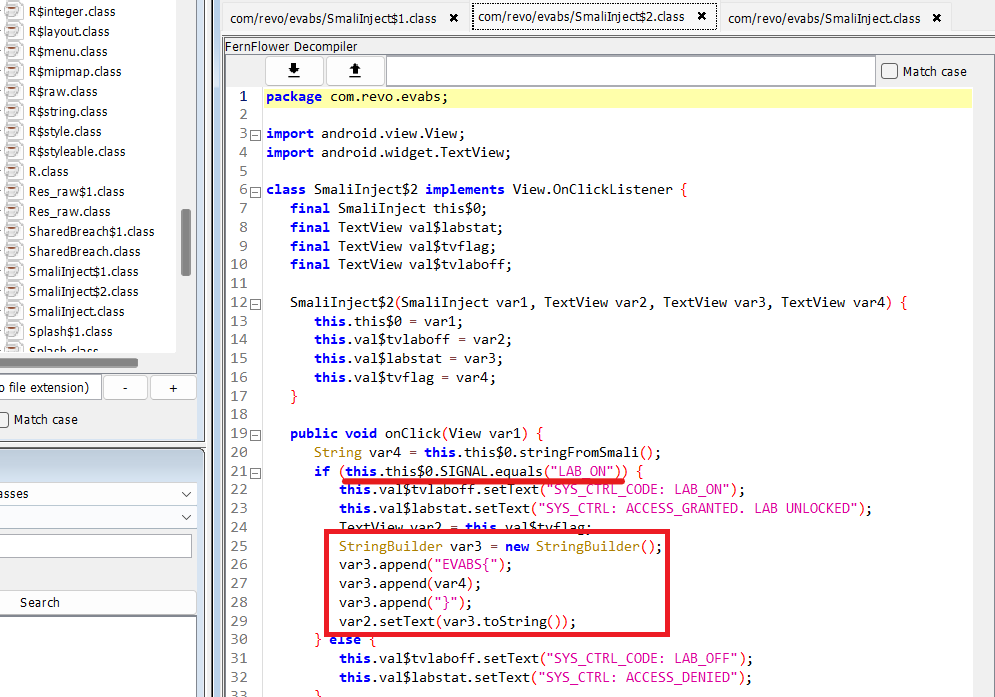
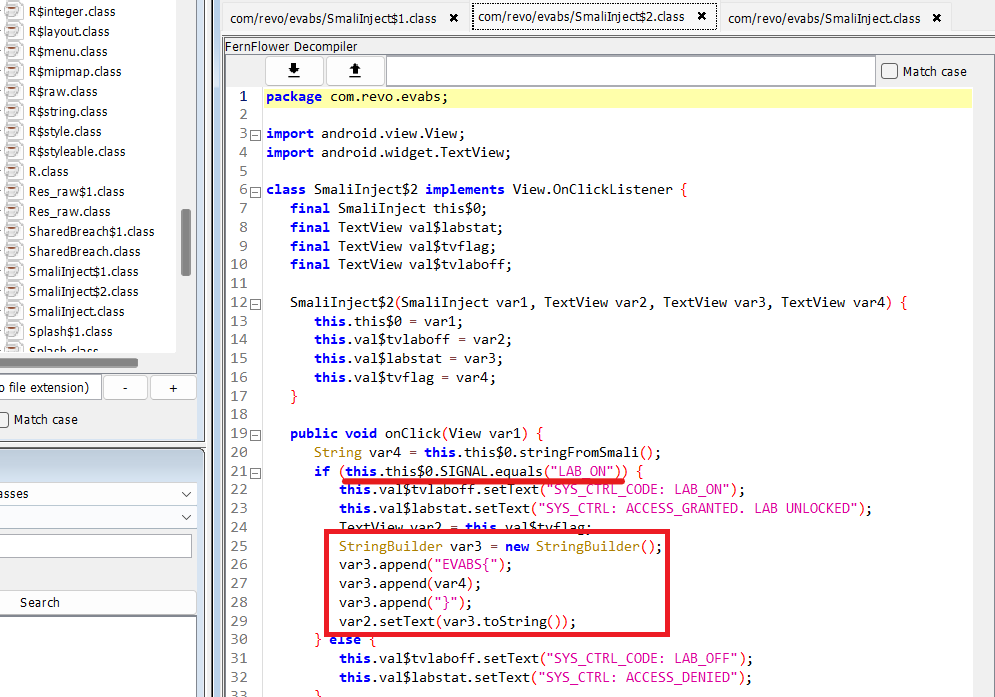
**Level 9 - Smali injection**



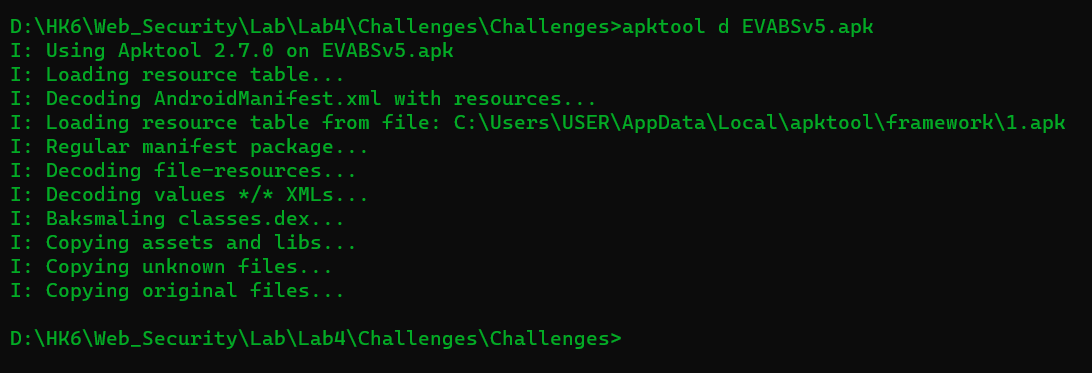
* Ở lab này ta có hint Small inject, ta tìm ở các class SmallInject và thấy define SIGNAL **LAB\_OFF**



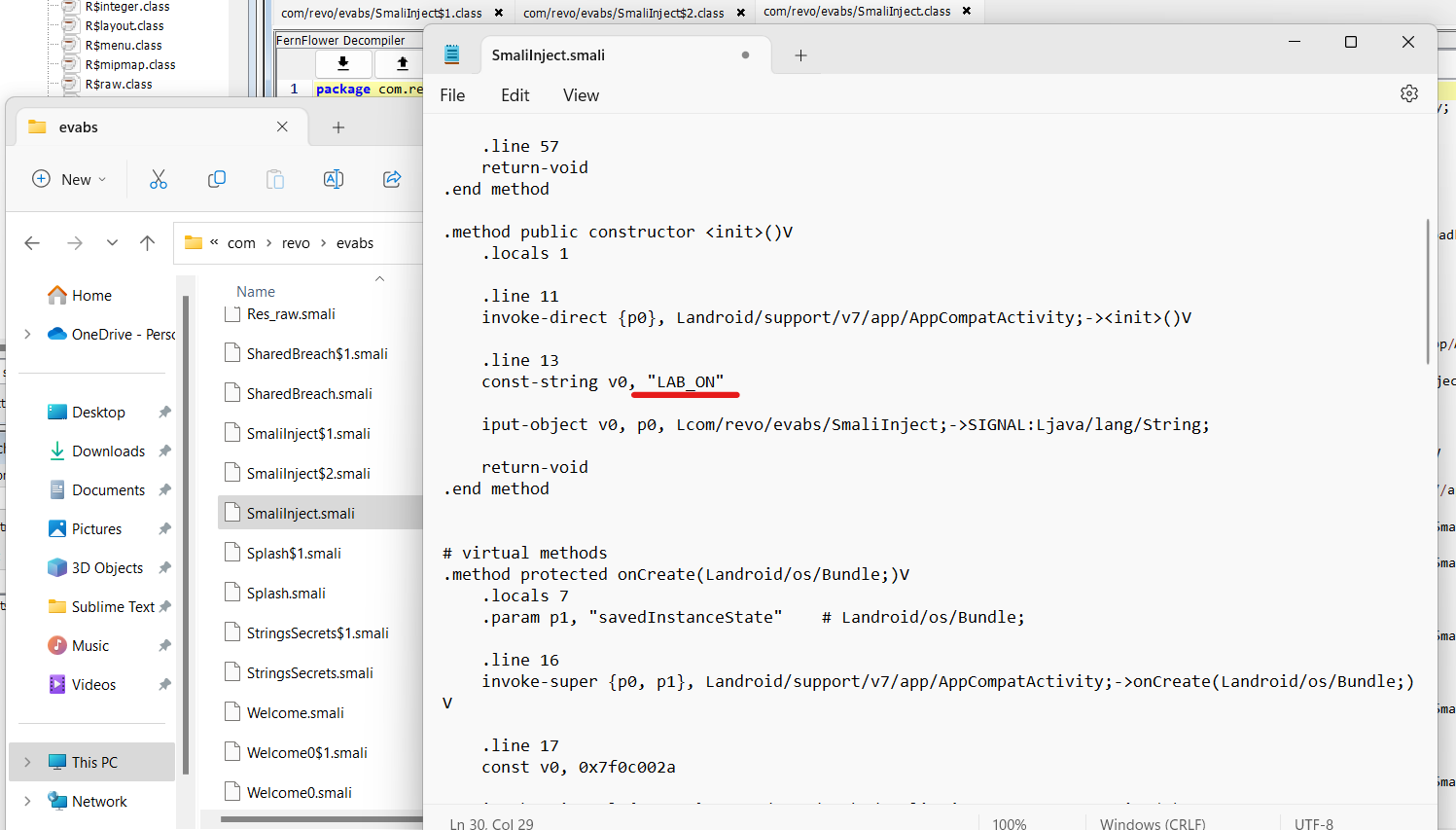
* Khi phân tích ta có thể thấy khi SIGNAL này có giá trị **LAB\_ON** sẽ hiện thị ra flag



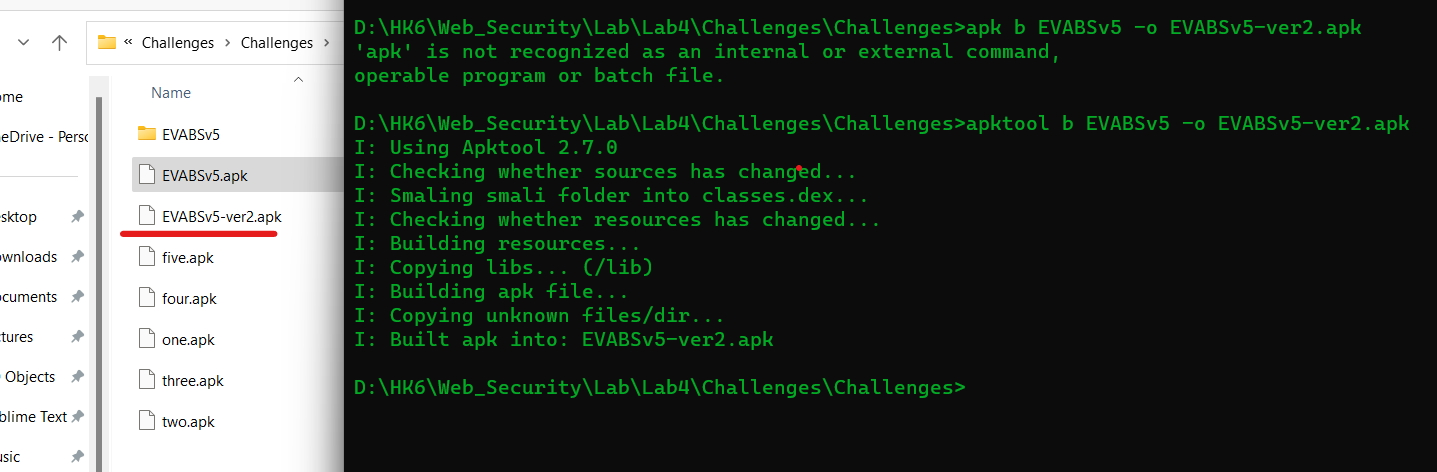
* Đến đây ta decompile



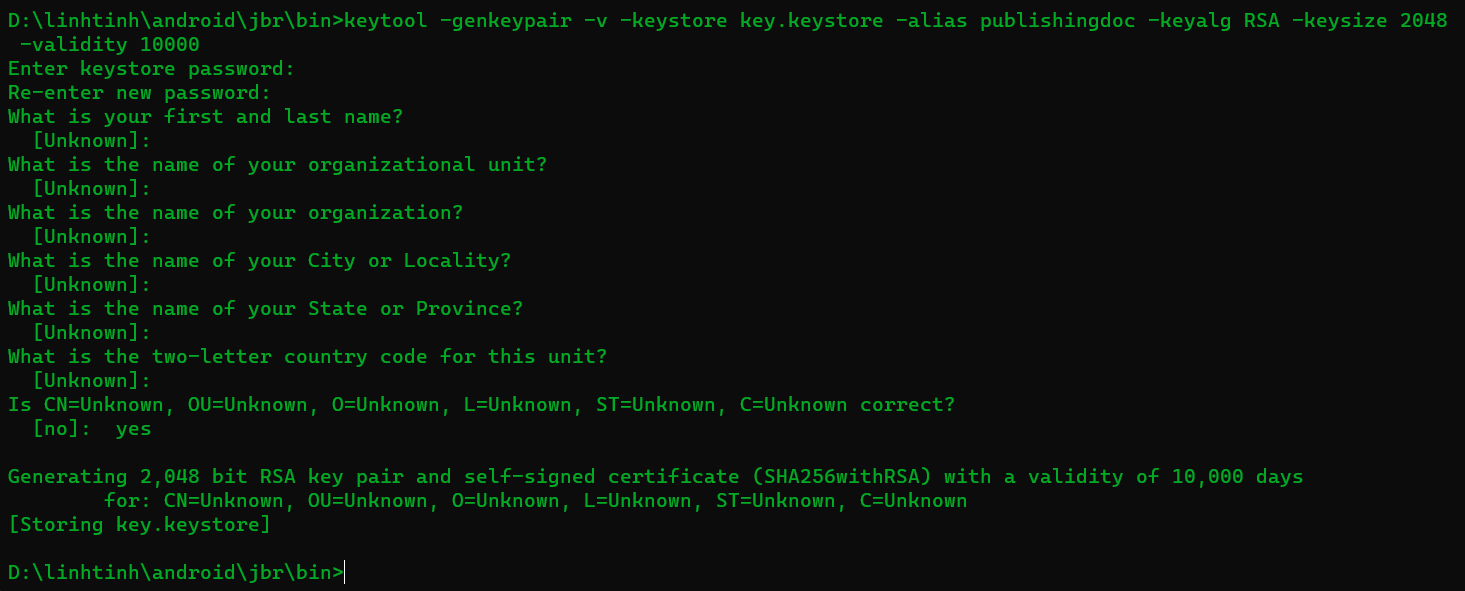
* Sửa SIGNAL có giá trị **LAB\_ON**



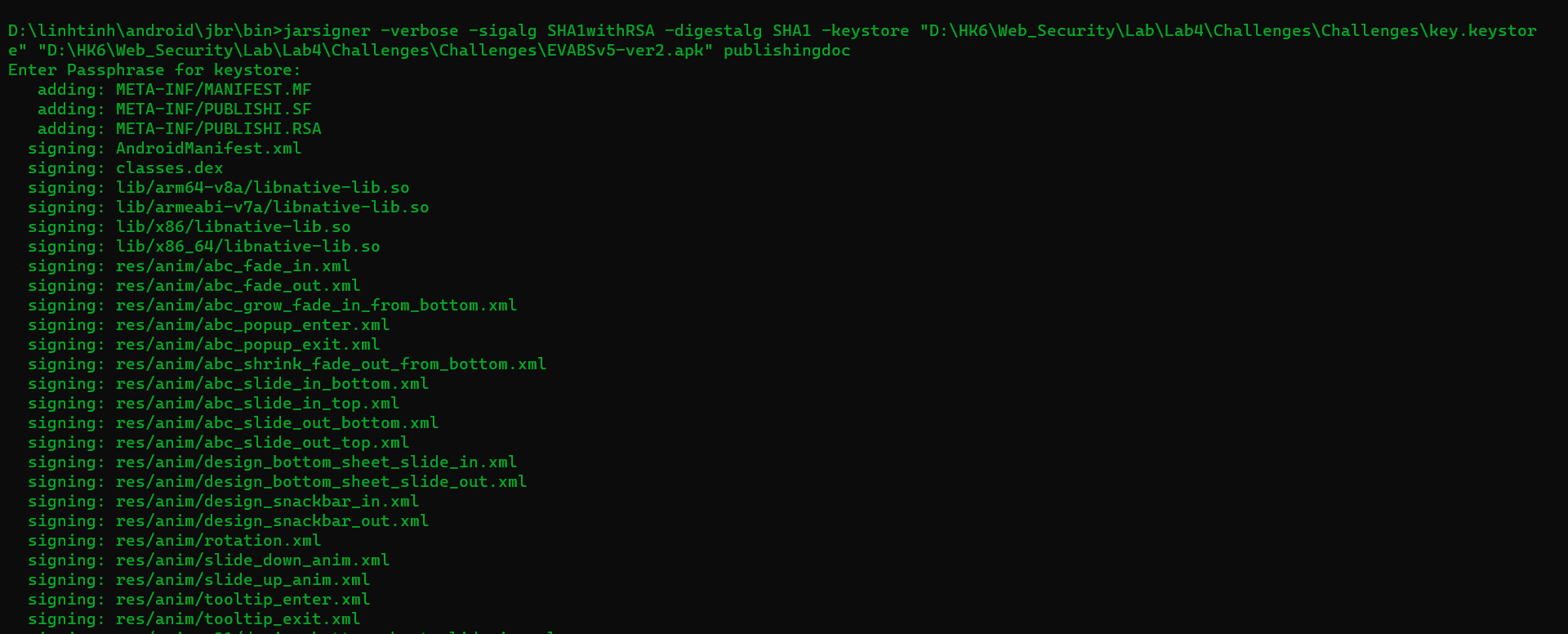
* Đến đây ta build lại

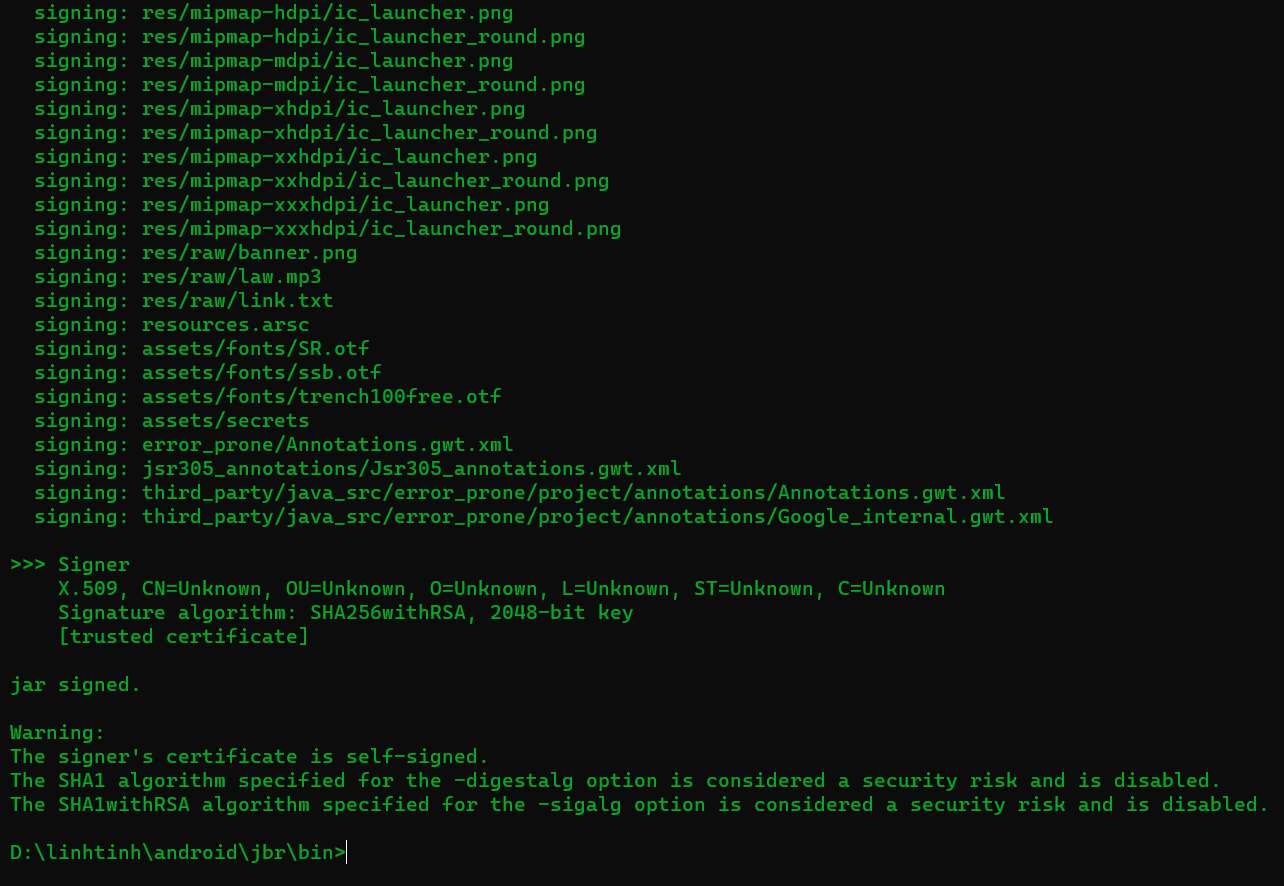


* Tạo chữ kí

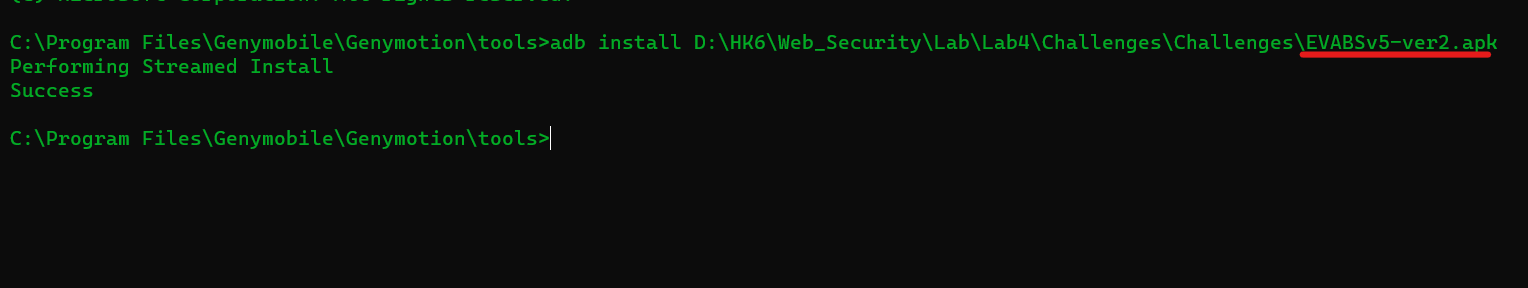


* Kí lại file apk mới vừa tạo

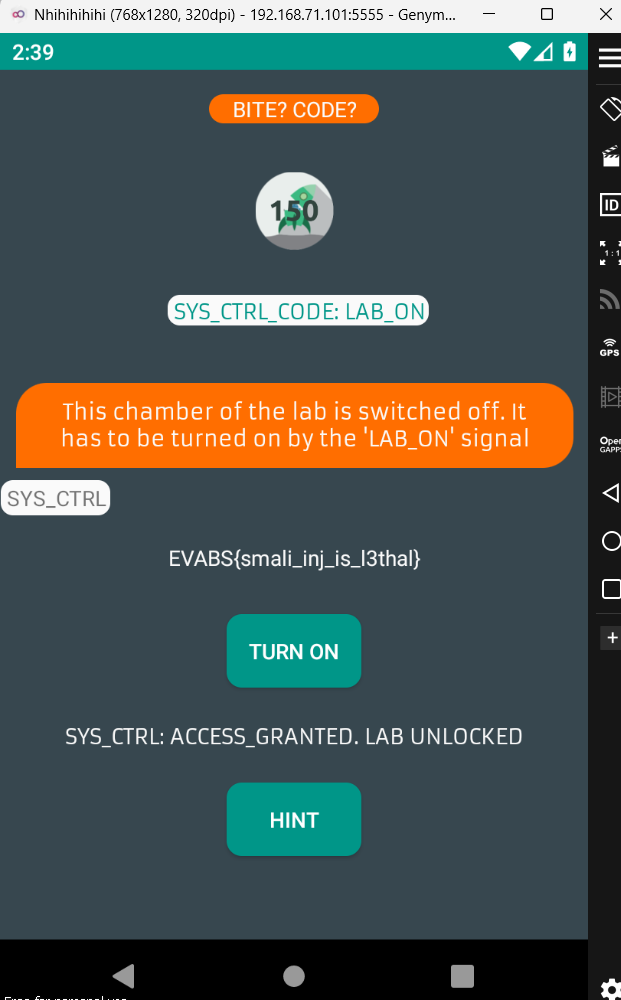




* Cuối cùng ta cài đặt lại file

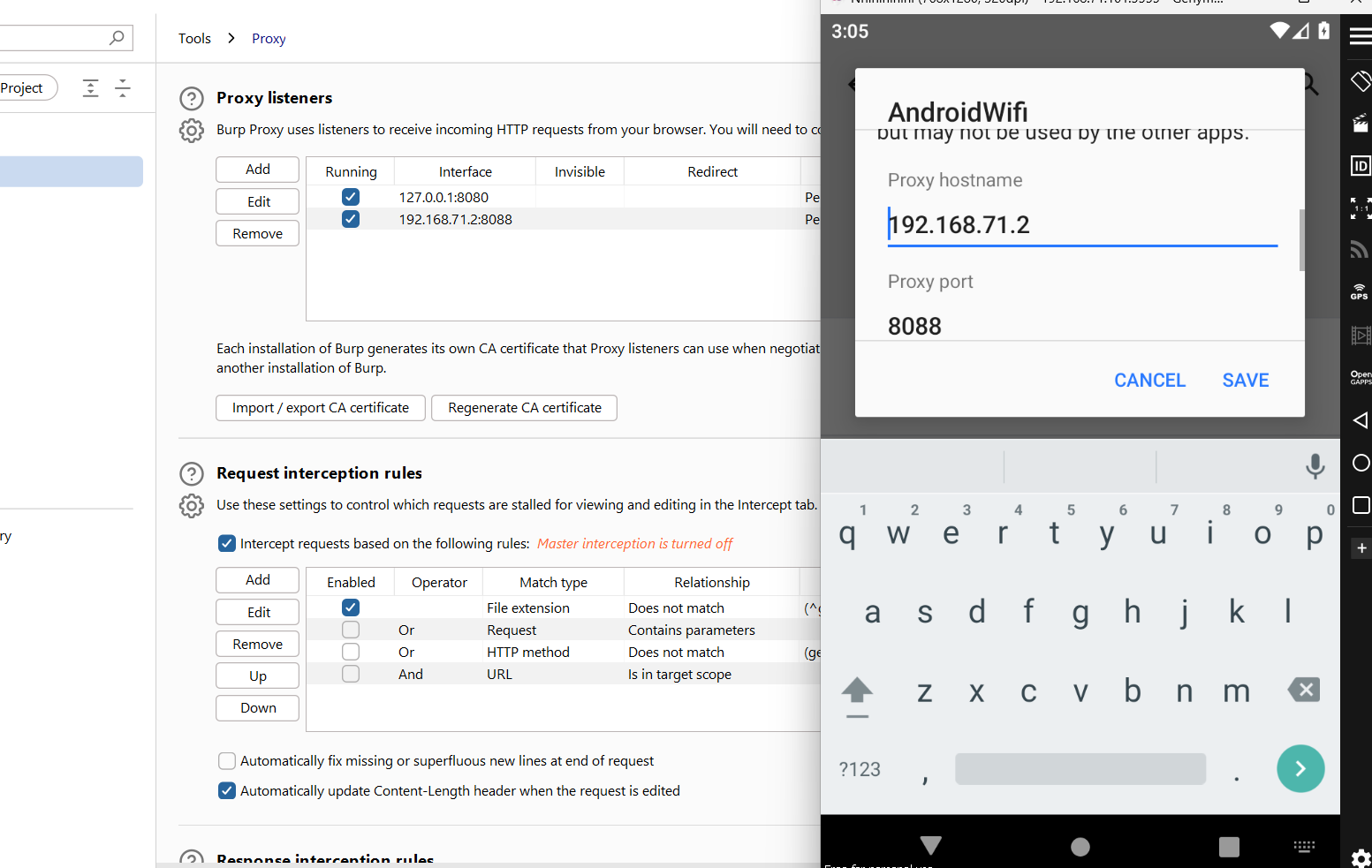


* Vào lại **TURN ON** và có flag

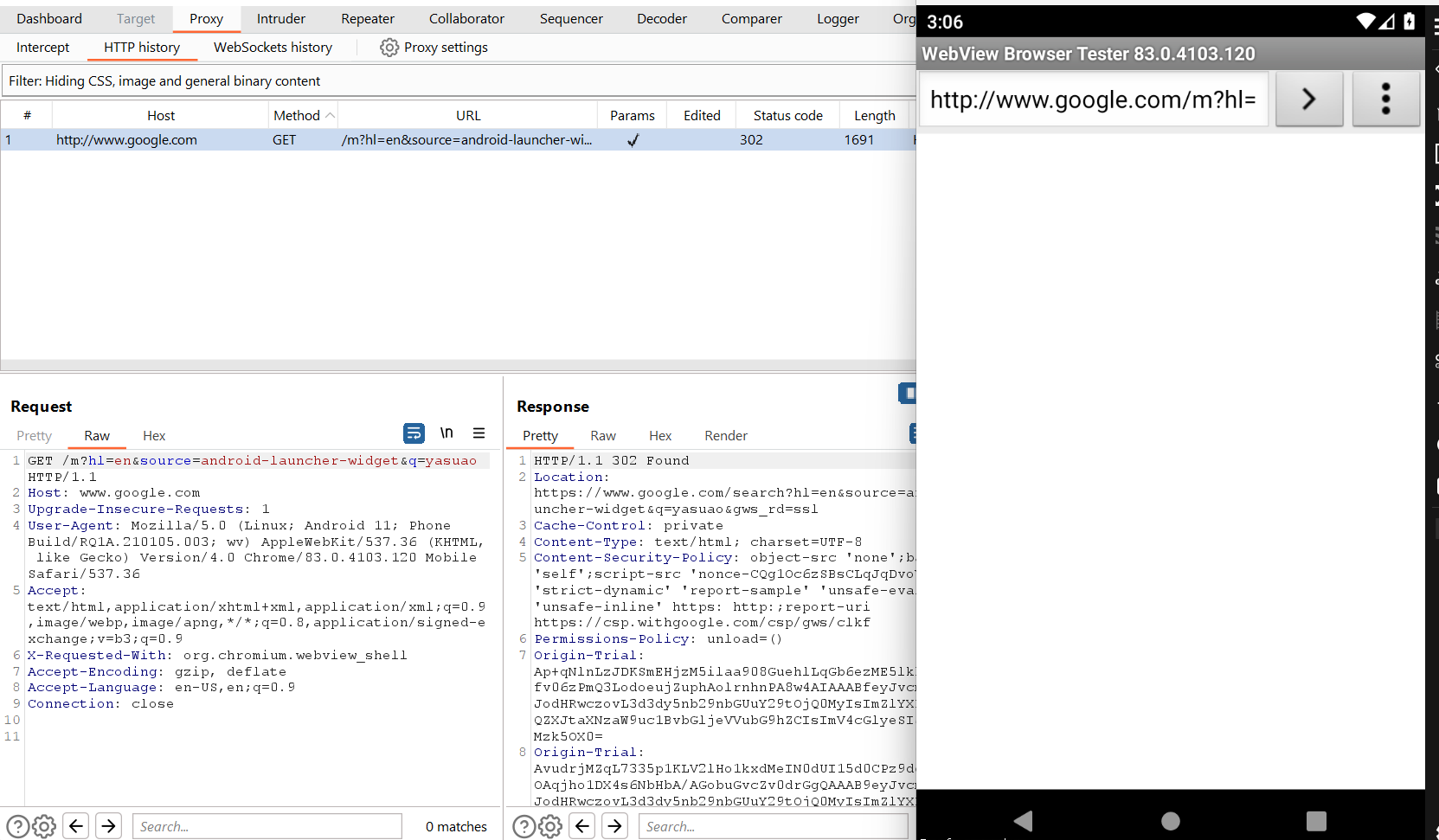


**Level 10: Intercept**

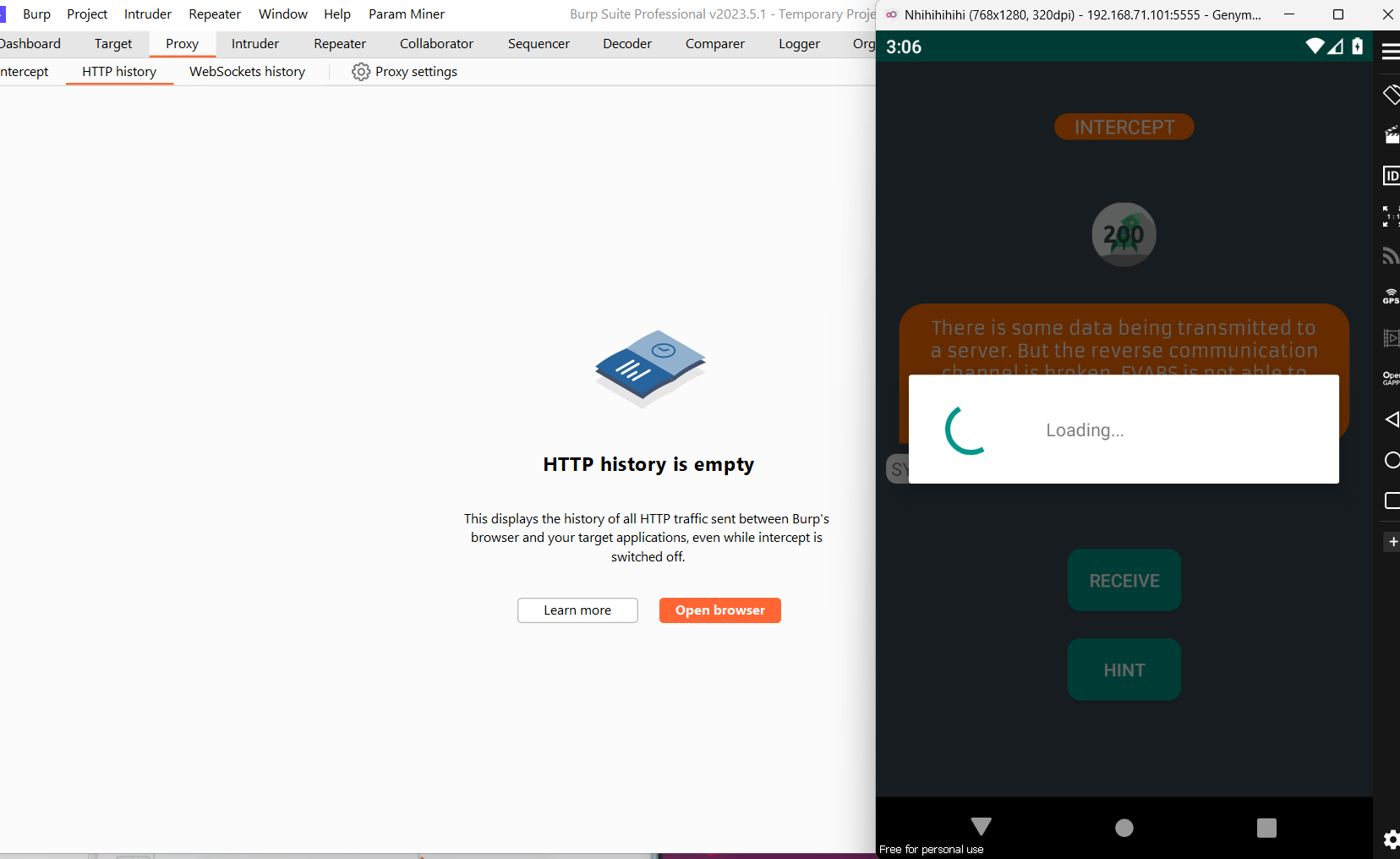
* Ta thiết lập điện thoại và burp



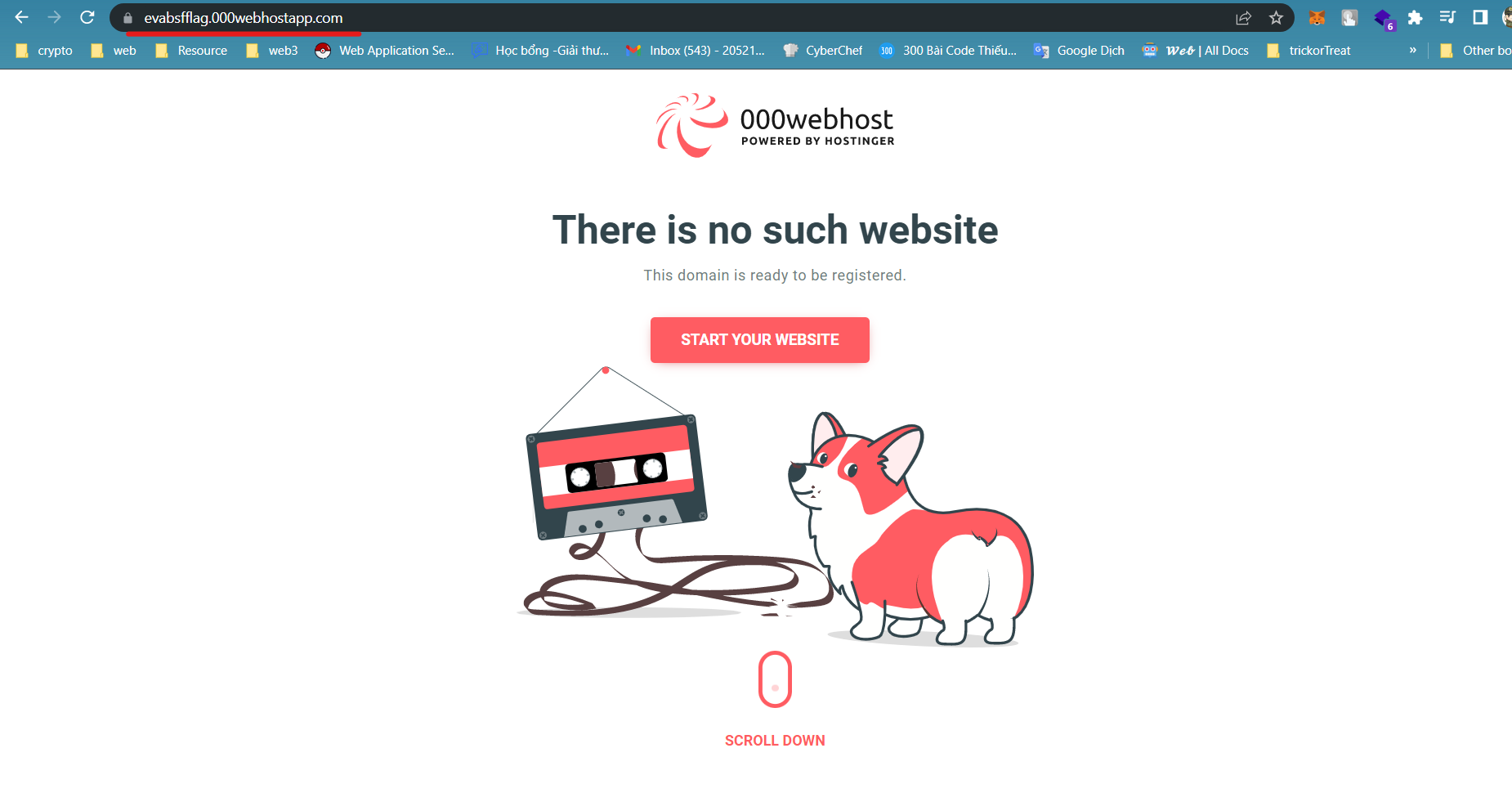
* Có thể thấy đã intercept gói tin thành công khi máy truy cập internet



* Tuy nhiên khi truy cập vào lab ta lại không thể nào bắt được gói tin nào



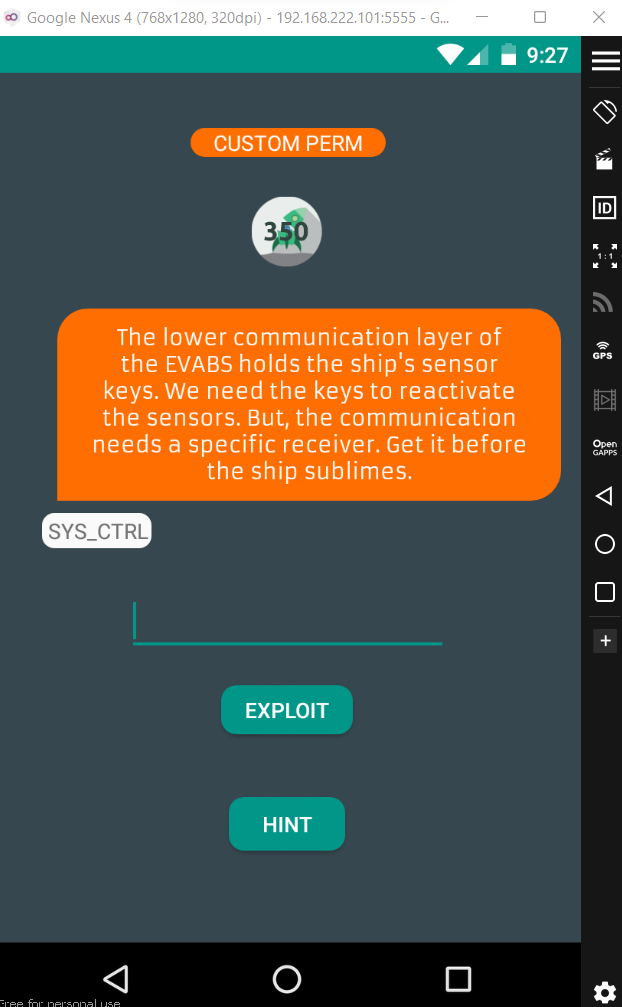
* Sau khi tìm hiểu mọi cách để fix có vẻ trang web host cho lab này đã hết hạn nên không truy cập được nữa



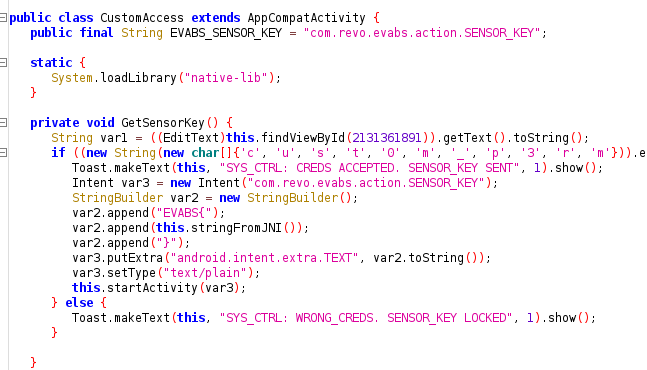
* Thời điểm làm challenge này web đã sập, khi đọc qua writeup cũ có thể phân tích đơn giản ý tưởng là ta intercept được gói tin và sửa đổi là xong

**Level 11: Custom PERM**

* Level này yêu cầu ta nhập input vào.

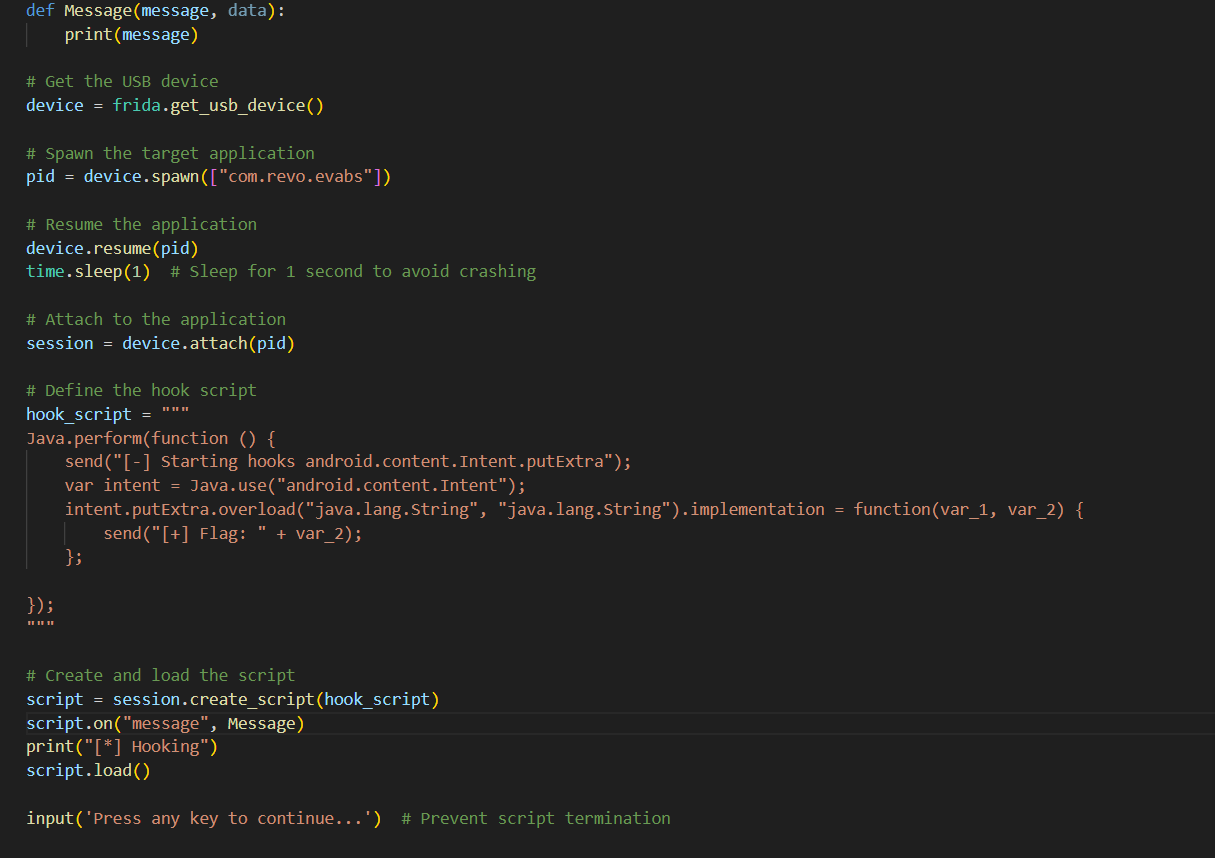


* Đọc source trên bytecode viewer:



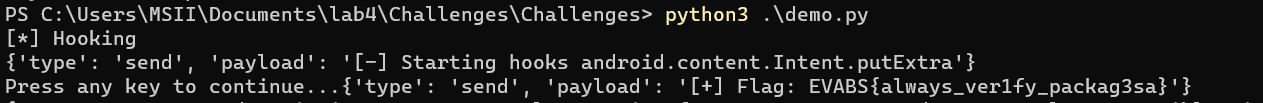
* Ta thấy input ở đây là chuỗi “cust0m\_p3rm”, nếu nhập đúng input thì flag sẽ được tạo ra vào truyền vào intent com.revo.evabs.action.SENSOR\_KEY bằng hàm putExtra().
* Ở đây ta sẽ suy nghĩ đến hướng hook hàm putExtra() bằng Frida để nó in ra flag cho mình.

Code frida:

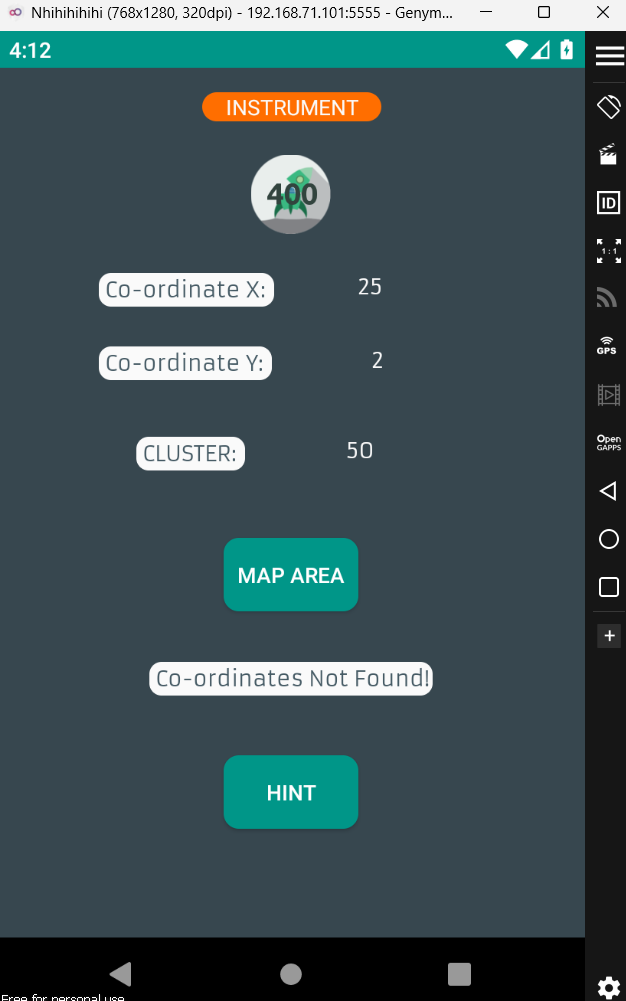


**Lưu ý**: Ở đây chúng ta phải tạo một hàm signal handler là Message. Hàm này sẽ xử lý các message gửi từ tiến trình đang chạy là “com.revo.evabs” đến shell của chúng ta và in ra data.  Khi hook\_script của chúng ta được load và hàm putExtra() được override thì khi tiến trình gọi đến hàm đó nó send biến var2 tức là Flag của bài.

* Chạy script:



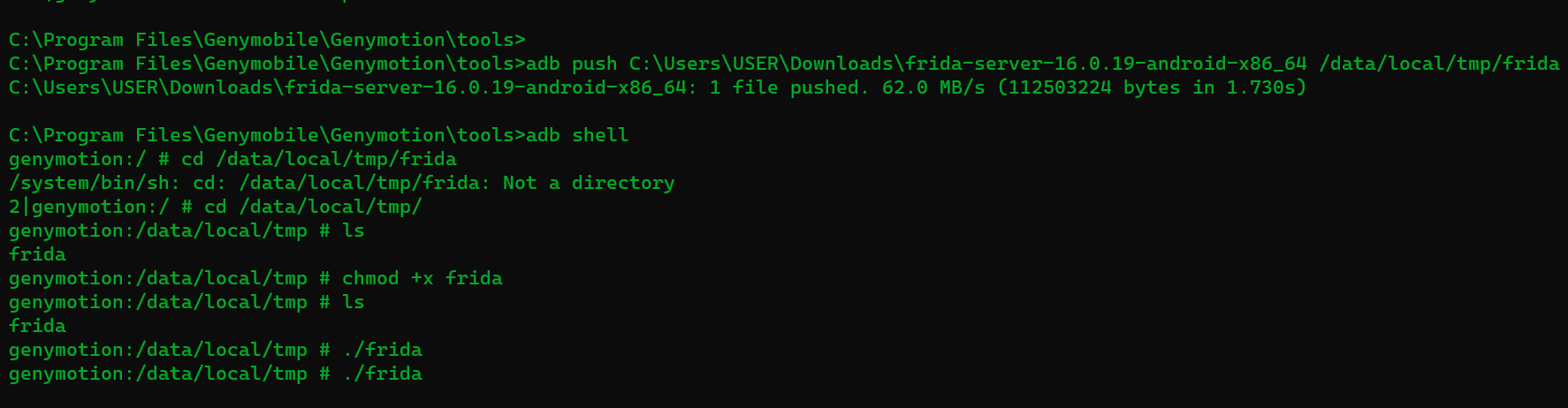
**Level 12 - Instrument**



* Khi đọc code challenge này ở Frida1.class



* Có thể phân tích cơ bản như sau chỉ cần giá trị của biến x tức a\*b lúc này lớn hơn var 2 (giá trị random từ 0 - 70) cộng 150 tuy nhiên giá trị a và b là cố định do đó x luôn bằng 50 do đó dù random cỡ nào cũng không thỏa điều kiện x hớn hơn 150+var2.
* Ý tưởng lúc này ta cần hook vào hàm random sao cho để điều kiện trên luôn đúng, cứ cho luôn giá trị random là -200 cho nóng
* Chạy frida ở server



* Viết đoạn code hook sau:

import frida

import time

# Get the USB device

device = frida.get\_usb\_device()

# Spawn the target application

pid = device.spawn(["com.revo.evabs"])

# Resume the application

device.resume(pid)

time.sleep(1)  # Sleep for 1 second to avoid crashing

# Attach to the application

session = device.attach(pid)

# Define the hook script

hook\_script = """

Java.perform(function () {

    send("[-] Starting hooks java.util.Random.nextInt");

    var random = Java.use("java.util.Random");

    random.nextInt.overload("int").implementation = function(var\_1) {

        return -200;

    };

});

"""

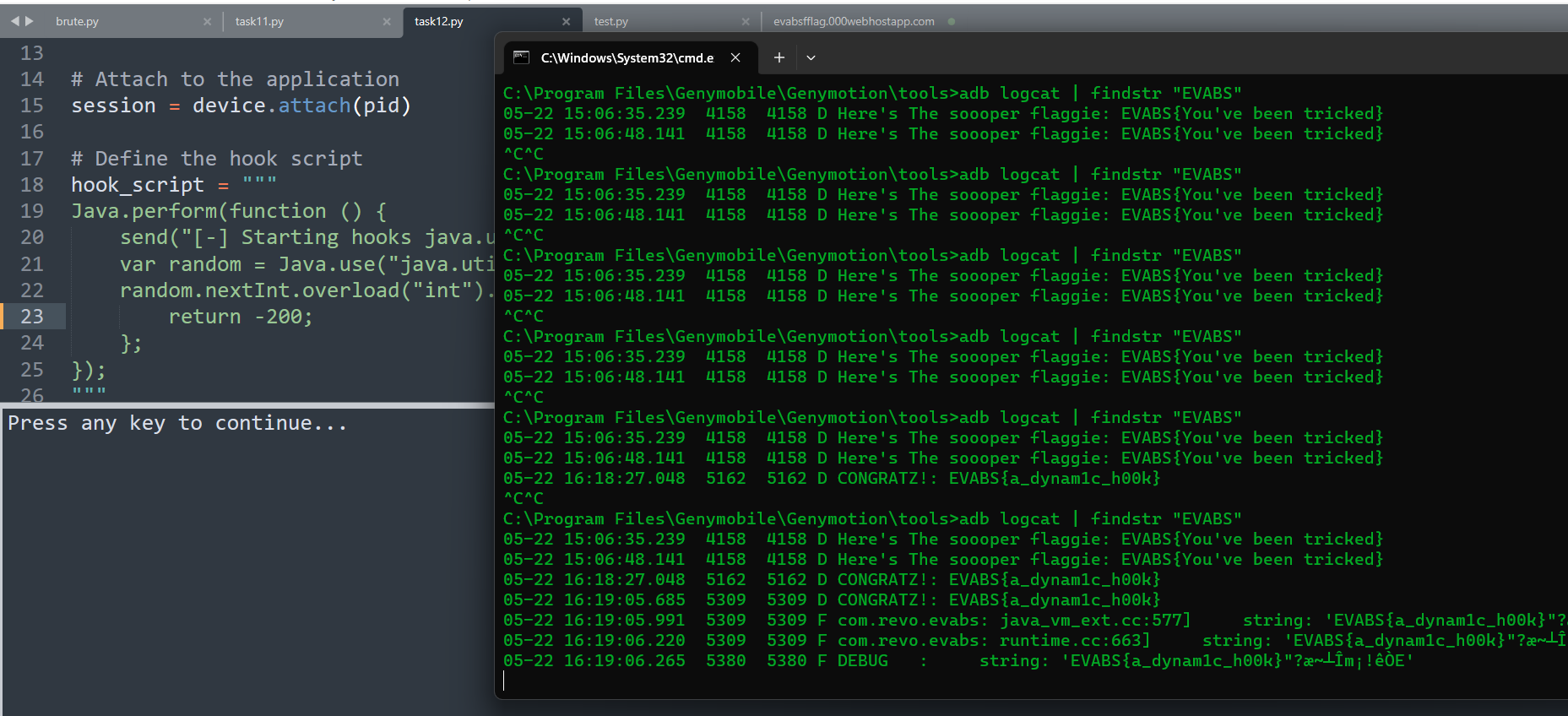
# Create and load the script

script = session.create\_script(hook\_script)

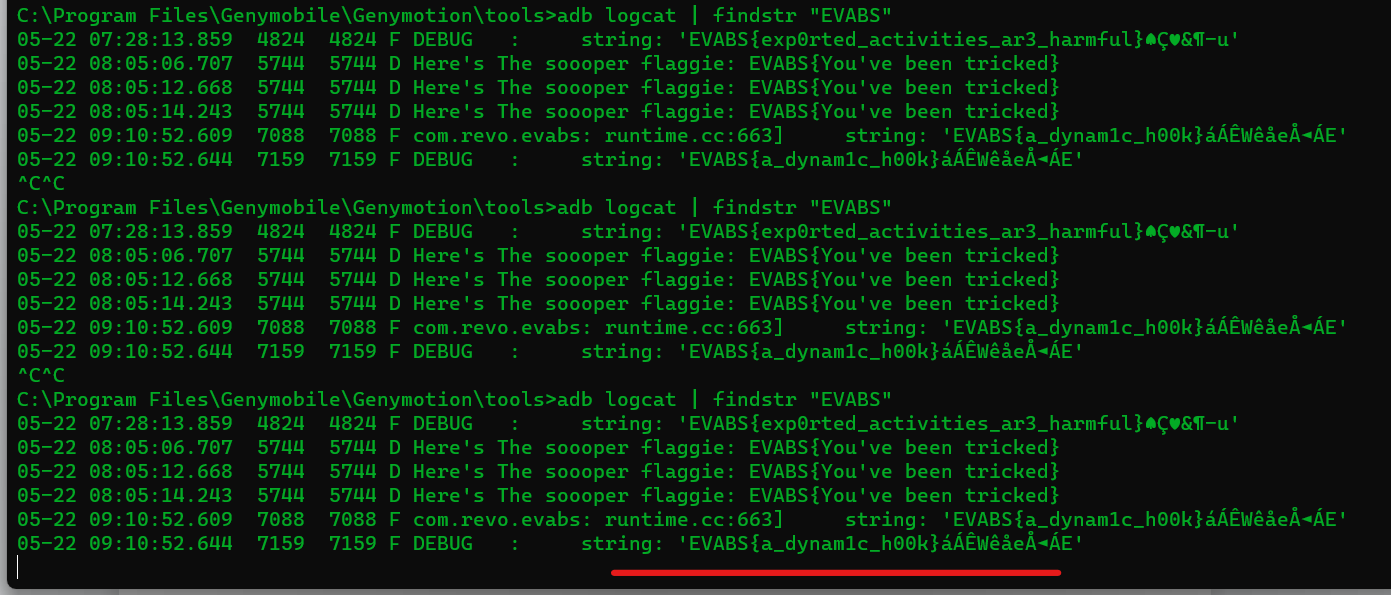
script.load()

input('Press any key to continue...')  # Prevent script termination

* Chạy ở máy thật



* Check ở log và thành công



* EVABS{a\_dynam1c\_h00k}

**HẾT**

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-0)