BÁO CÁO THỰC HÀNH

**Môn học: Bảo mật web và ứng dụng**

**Kỳ báo cáo: Buổi 04 (Session 04)**

**Tên chủ đề: Pentest Android Application**

*GVHD: Ngô Khánh Khoa*

*Ngày báo cáo: 7/05/2024*

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT213.O21.ANTN

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Phạm Ngọc Thơ | 21522641 | 21522641@gm.uit.edu.vn |
| 2 | Hà Thị Thu Hiền | 21522056 | 21522056@gm.uit.edu.vn |

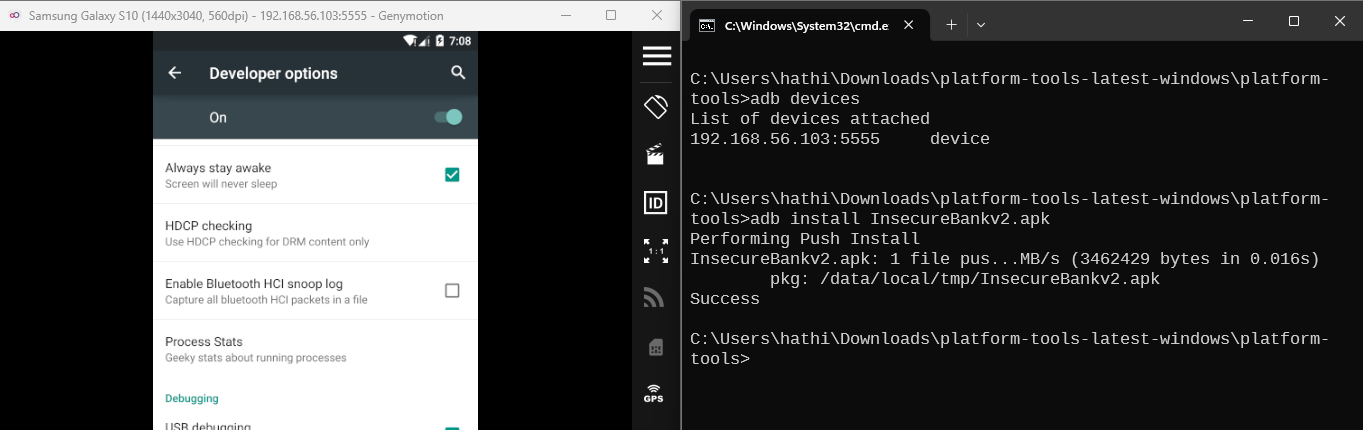
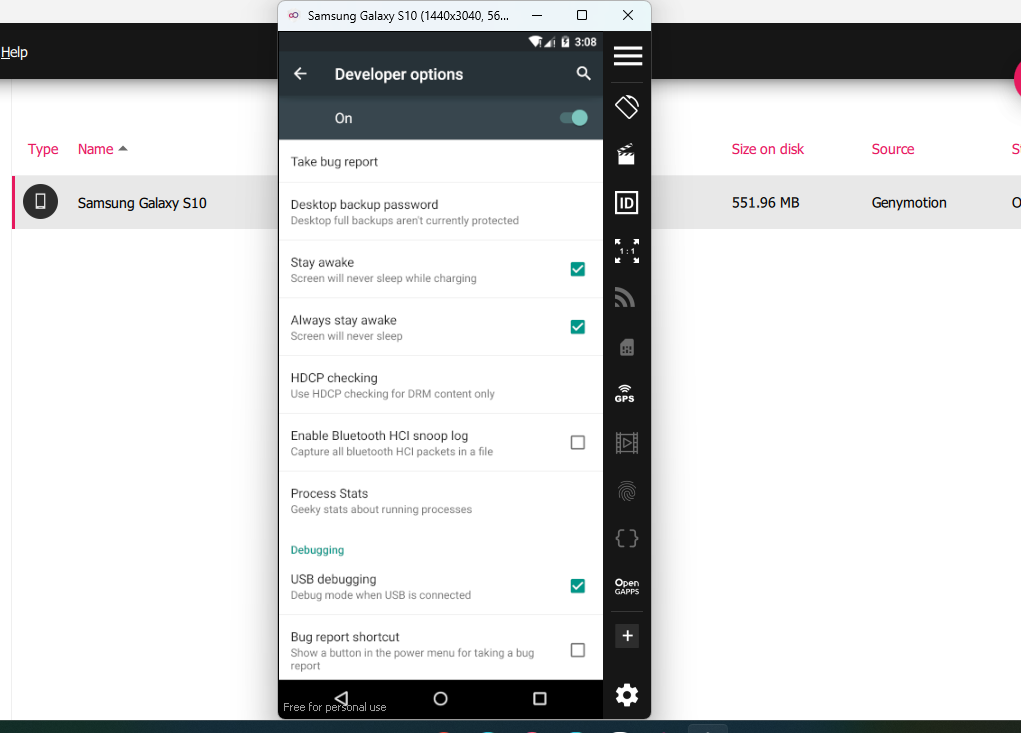
1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

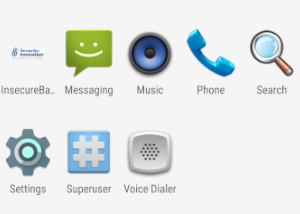
|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** |
| 1 | Kịch bản 01 | 100% |
| 2 | Kịch bản 02 | 100% |
| 3 | Kịch bản 03 | 100% |
| 4 | Kịch bản 04 | 100% |
| 5 | Kịch bản 05 | 100% |
| 6 | Kịch bản 06 | 100% |
| 7 | Kịch bản 07 | 100% |
| 8 | Challenge 1 EVABSv5 | Done 1-10 |
| 9 | Challenge 2 PicoCTF | Done 1 |

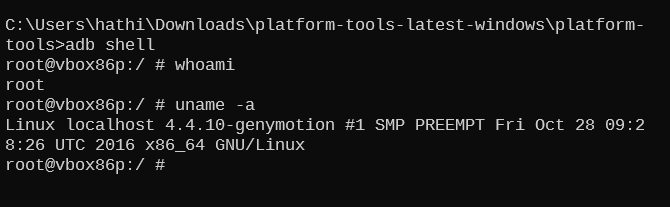
**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

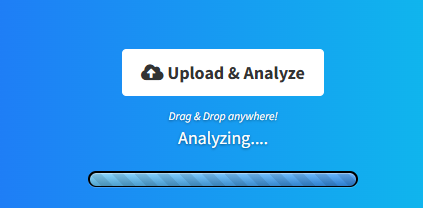
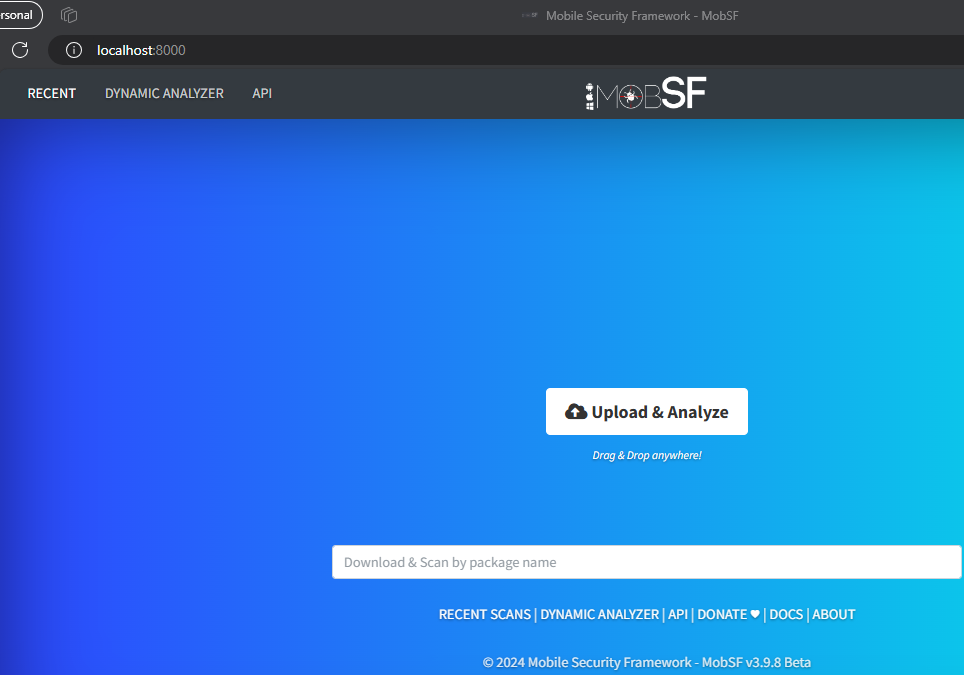
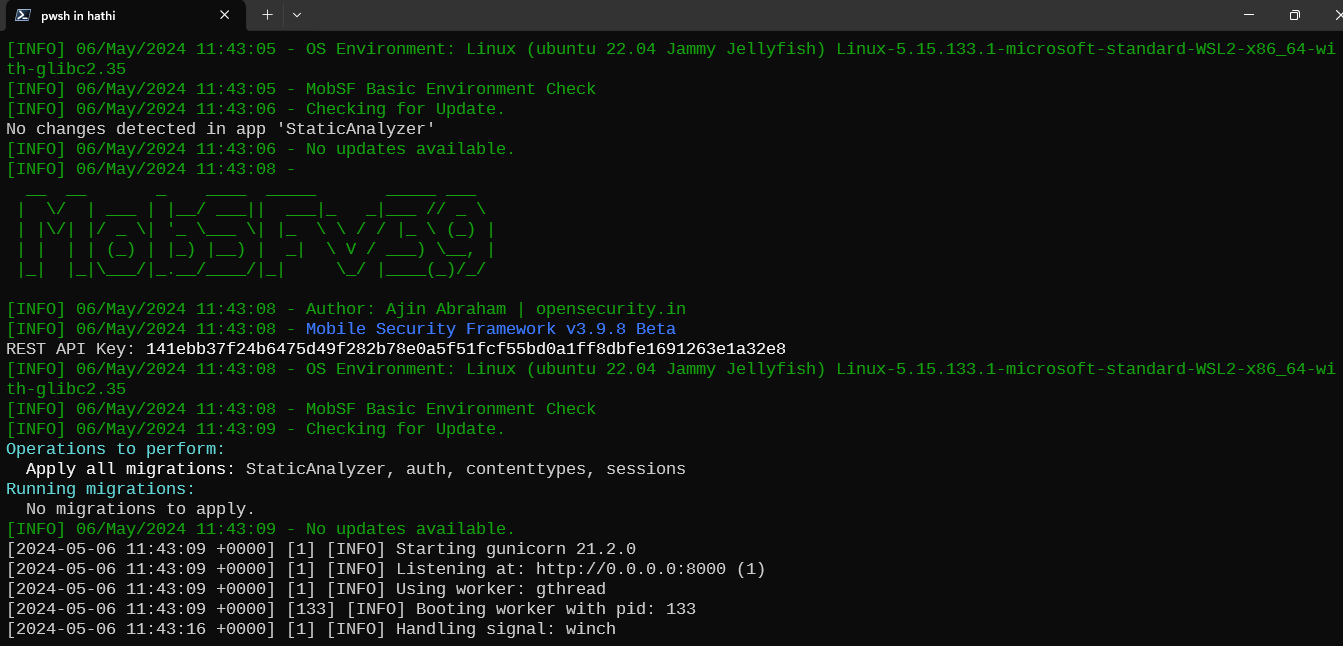
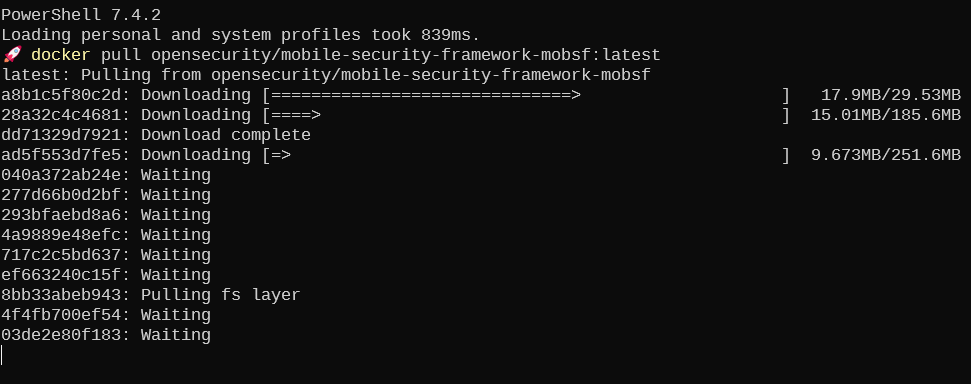
BÁO CÁO CHI TIẾT

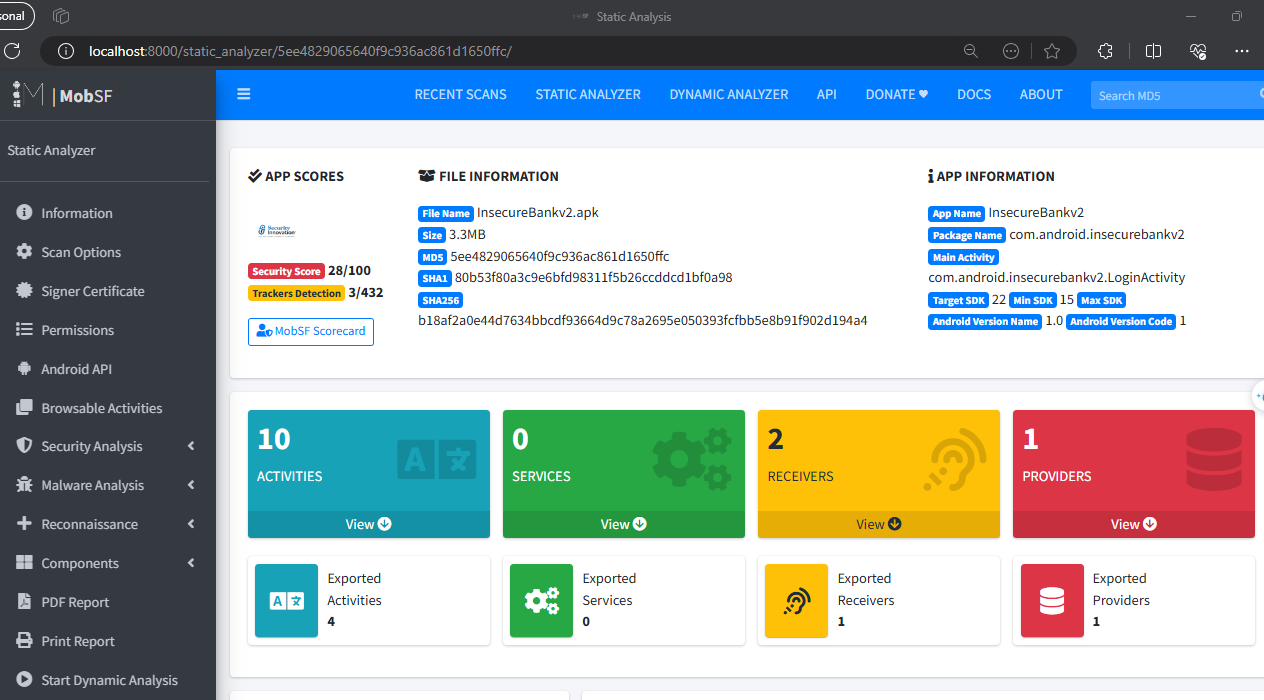
## **Yêu cầu 1: Phân tích và chỉ ra điểm bất thường của đoạn code trên?**

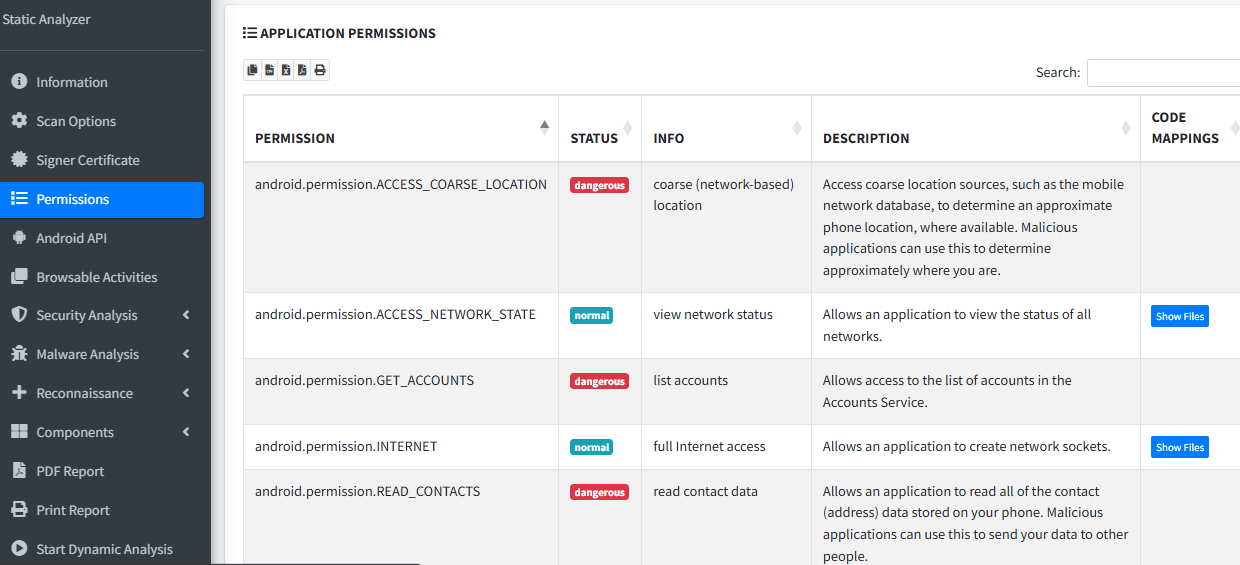


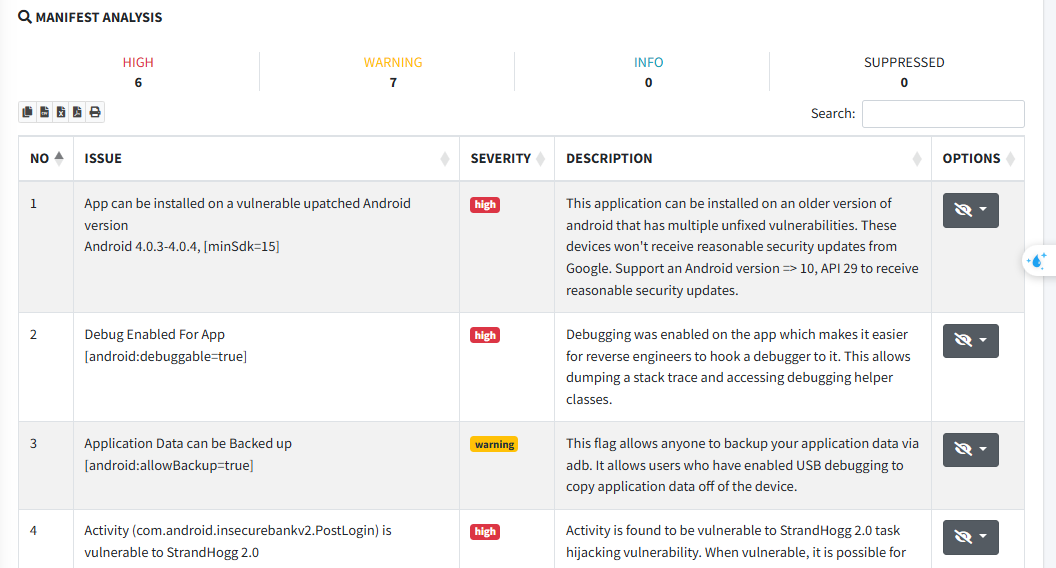


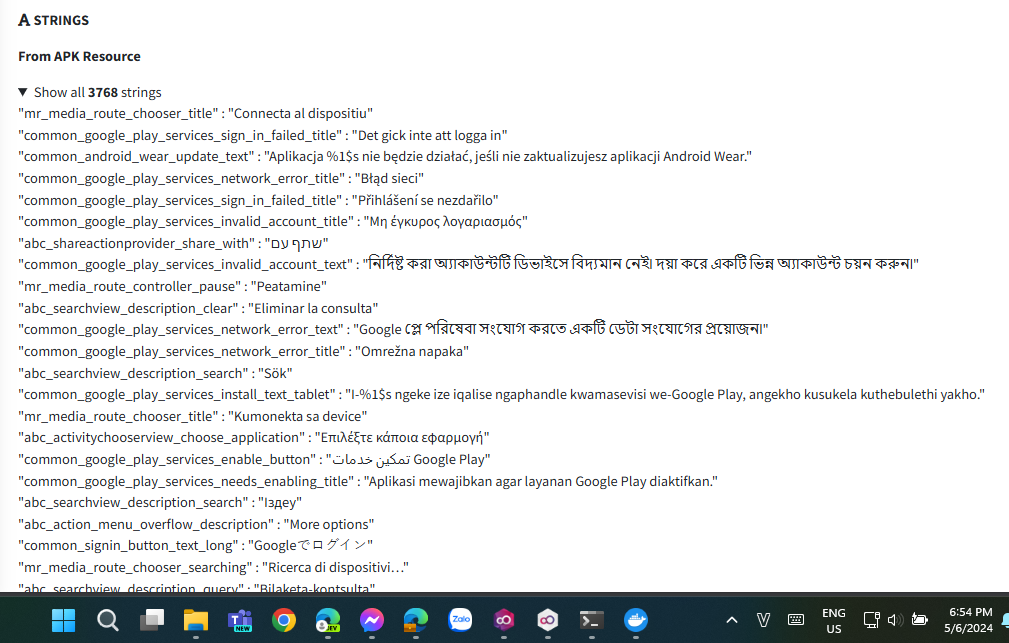


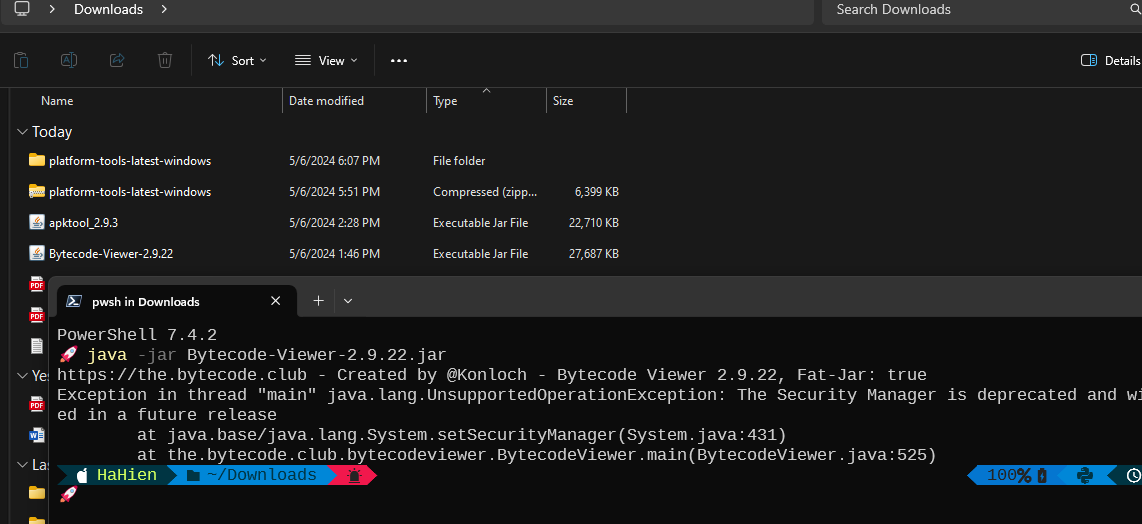


- Kiểm tra cấu hình phân quyền

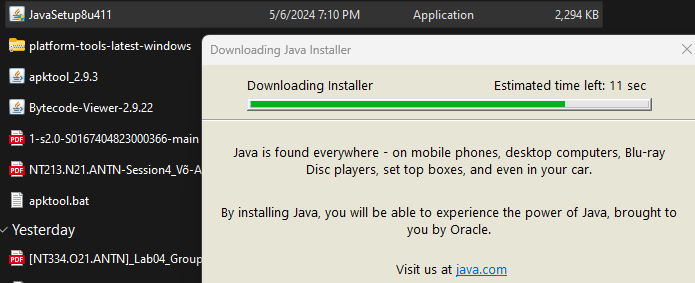
- Kiểm tra cấu hình trong tập tin Manifest

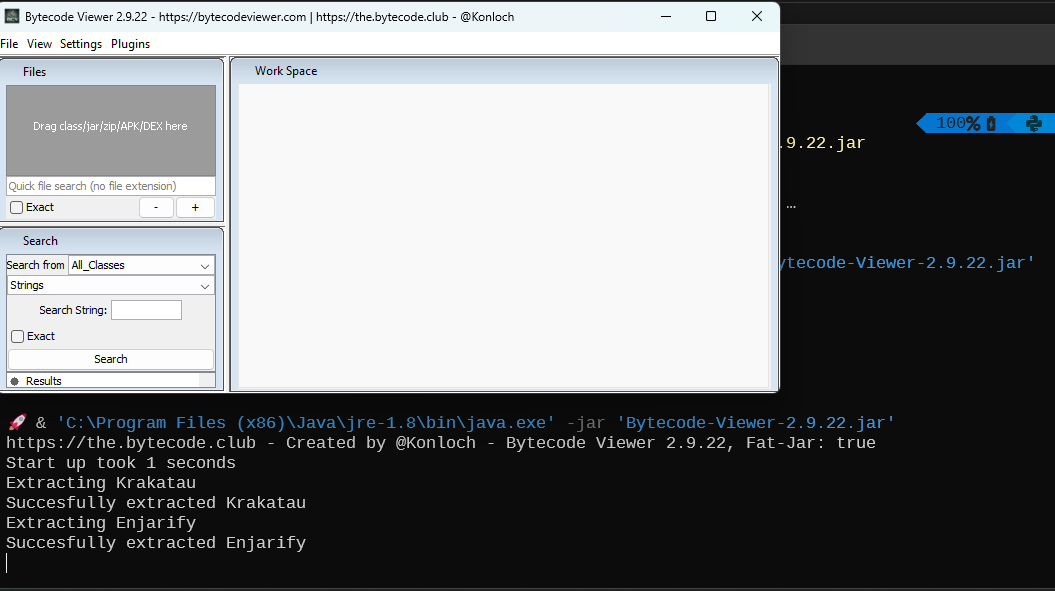
- MobSF cung cấp thông cần thiết để đánh giá độ bảo mật của chương trình

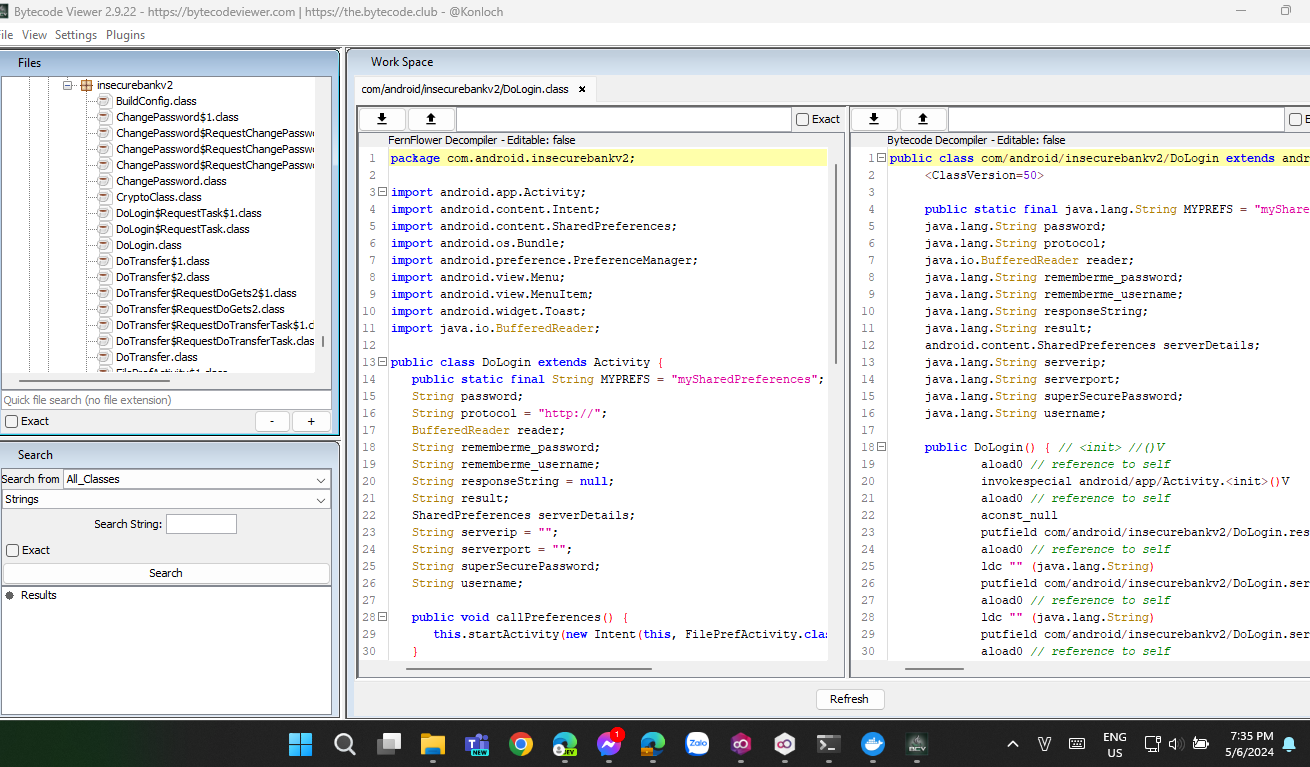


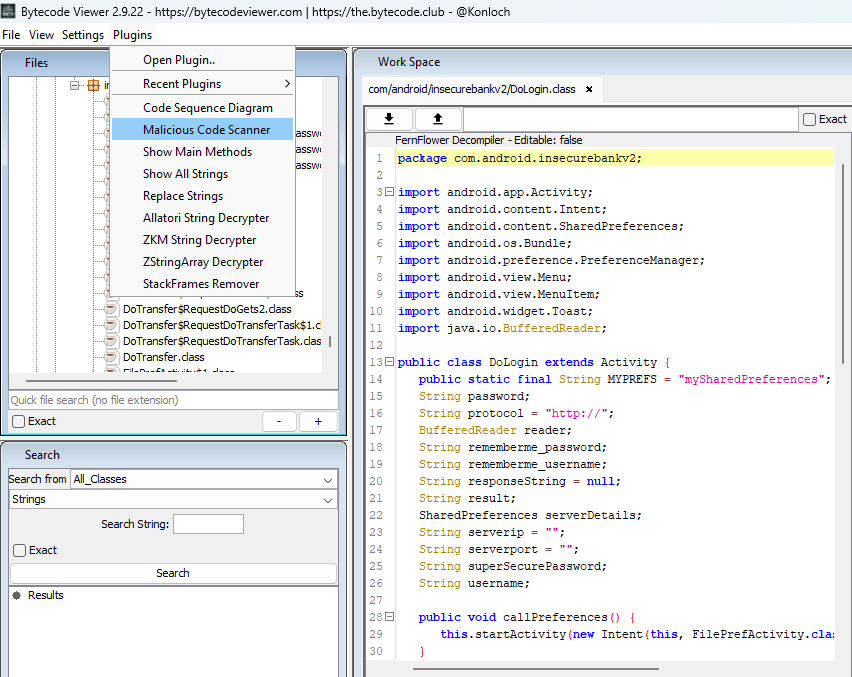
- Trong trường hợp này, chương trình đang cố gắng sử dụng Security Manager, một tính năng của Java được sử dụng để thiết lập các quy tắc bảo mật cho ứng dụng Java. Tuy nhiên, từ Java 9 trở đi, Security Manager đã bị đánh dấu là không được khuyến khích sử dụng và sẽ bị gỡ bỏ trong các phiên bản Java tương lai.

- Cài java 8 để chạy xem sao

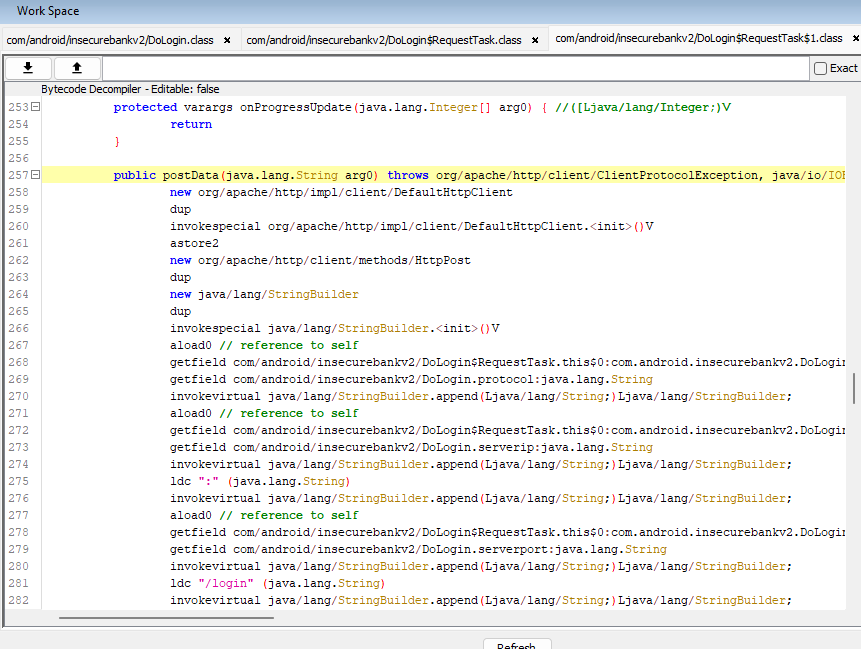




- Sử dụng Malicious Code Scanner ở mục Plugins để quét các đoạn code có vẻ nguy hiểm

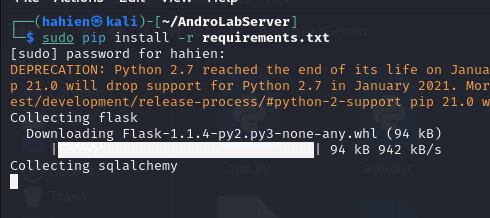


* Ví dụ: Nhìn thấy đoạn code lạ ở đường dẫn: com/android/insecurebankv2/DoLogin$RequestTask.class

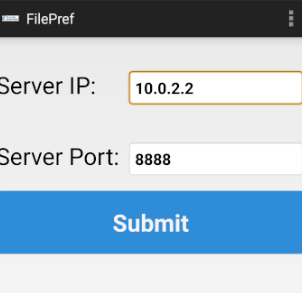
- Các điểm bất thường trong đoạn code trên là:

1. **Sử dụng HttpClient cũ (Deprecated):** Mã sử dụng **org.apache.http.impl.client.DefaultHttpClient** và **org.apache.http.client.methods.HttpPost**, nhưng các lớp này đã bị loại bỏ trong Android API level 23. Thay vào đó, từ Android API level 23 trở lên, bạn nên sử dụng **HttpURLConnection** hoặc thư viện HTTP hiện đại như OkHttp.
2. **Sử dụng AES mà không có Initialization Vector (IV):** Trong hàm **saveCreds**, có vẻ như mã hóa AES được sử dụng để lưu trữ mật khẩu, nhưng không có Initialization Vector (IV) nào được sử dụng. Điều này có thể làm giảm độ an toàn của việc mã hóa.
3. **Sử dụng Base64 mã hóa không an toàn:** Trong hàm **saveCreds**, mật khẩu được mã hóa bằng Base64 trước khi lưu vào SharedPreferences. Base64 không phải là một phương pháp mã hóa an toàn cho mật khẩu, vì nó chỉ là một cách chuyển đổi dữ liệu sang định dạng có thể đọc được mà không cung cấp bất kỳ cơ chế bảo vệ nào.
4. **Cách tiếp cận bảo mật không đầy đủ:** Dữ liệu nhạy cảm như tên người dùng và mật khẩu được lưu trữ trong SharedPreferences mà không có bất kỳ biện pháp bảo mật nào khác ngoài mã hóa cơ bản.
5. **Không kiểm tra chứng chỉ TLS/SSL:** Nếu ứng dụng giao tiếp qua mạng, việc này có thể tạo ra một lỗ hổng bảo mật nếu không kiểm tra chứng chỉ TLS/SSL.
6. **Sử dụng API không an toàn:** Có vẻ như API được sử dụng để gửi thông tin đăng nhập là không an toàn, có thể dẫn đến các lỗ hổng bảo mật như SQL Injection hoặc Cross-Site Scripting (XSS).
7. **Sử dụng AsyncTask:** AsyncTask được sử dụng trong mã, đây là một cách lỗi thời để thực hiện các tác vụ nền. Nó có thể dẫn đến các vấn đề liên quan đến vòng đời của hoạt động và tiêu thụ tài nguyên không kiểm soát được.
8. **Cung cấp quyền truy cập không cần thiết:** Mã yêu cầu quyền truy cập không cần thiết hoặc quyền truy cập lớn hơn cần thiết, điều này có thể gây ra các vấn đề về quyền riêng tư và bảo mật.
9. **Không kiểm tra đầu vào người dùng:** Có vẻ như không có kiểm tra hoặc xác thực nào được thực hiện trên dữ liệu đầu vào từ người dùng, điều này có thể tạo ra các lỗ hổng bảo mật như XSS hoặc SQL Injection.
10. **Logging Sensitive Information:** Có một số lời ghi log được thực hiện với thông tin nhạy cảm như tên người dùng và mật khẩu, điều này có thể gây ra các vấn đề về bảo mật nếu log này không được xử lý đúng cách.

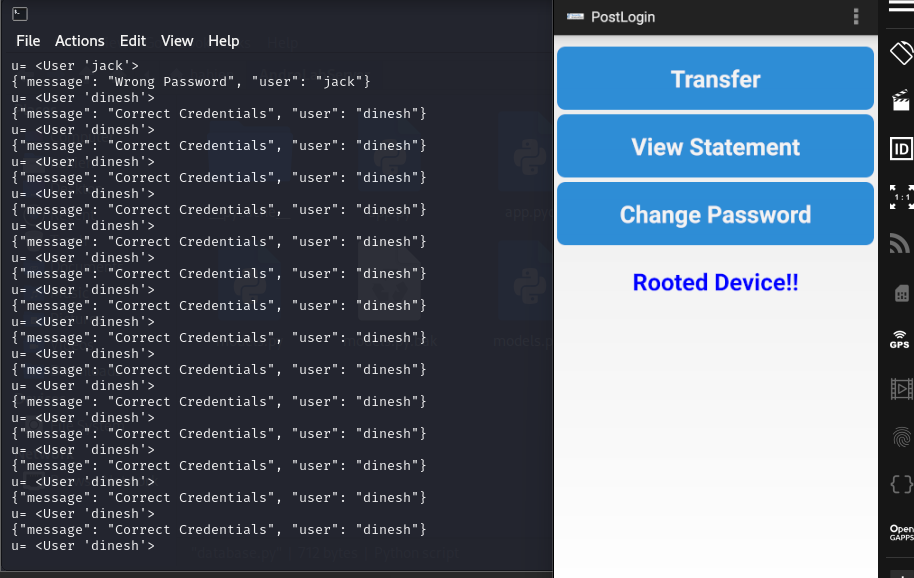
## **Yêu cầu 2: Chỉ ra rằng dữ liệu lưu trữ có an toàn hay không?**



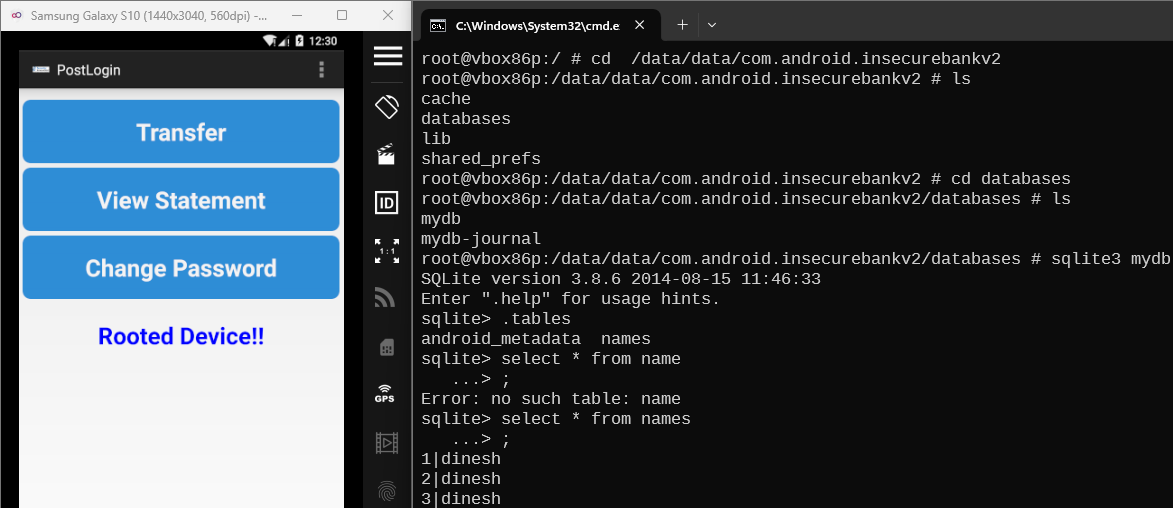
* Chạy theo lệnh trong file readme, xoá convert\_unicode=True trong database.py là chạy thành công server lên port 8888.
* Chỉnh sửa thông tin thành địa chỉ IP máy server và port tại đây rồi submit



* Cuối cùng, điền username và password của Dinesh như đã được cung cấp trong đề, và login thành công.

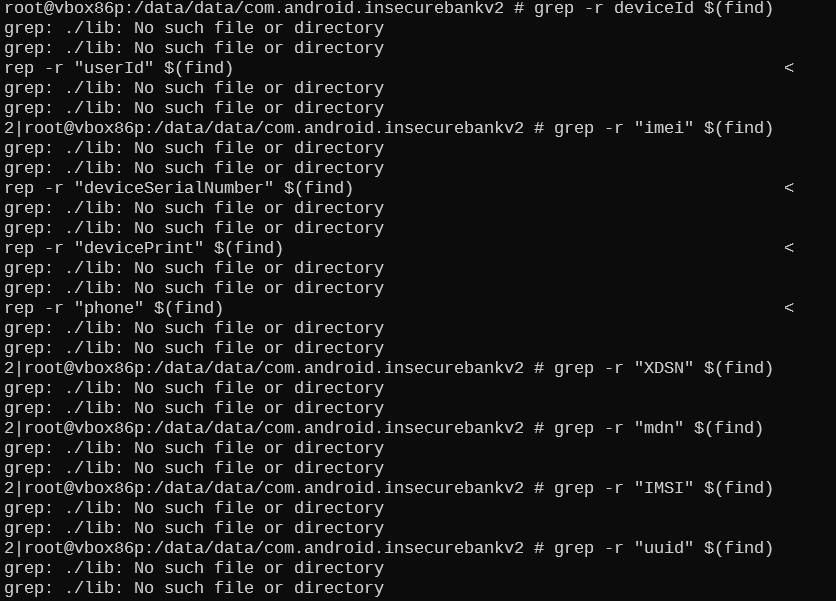


* Gọi adb shell để tương tác với máy điện thoại, xem trong database có gì



* Việc lưu trữ dữ liệu dưới dạng plaintext không được mã hoá ==> dữ liệu không an toàn.

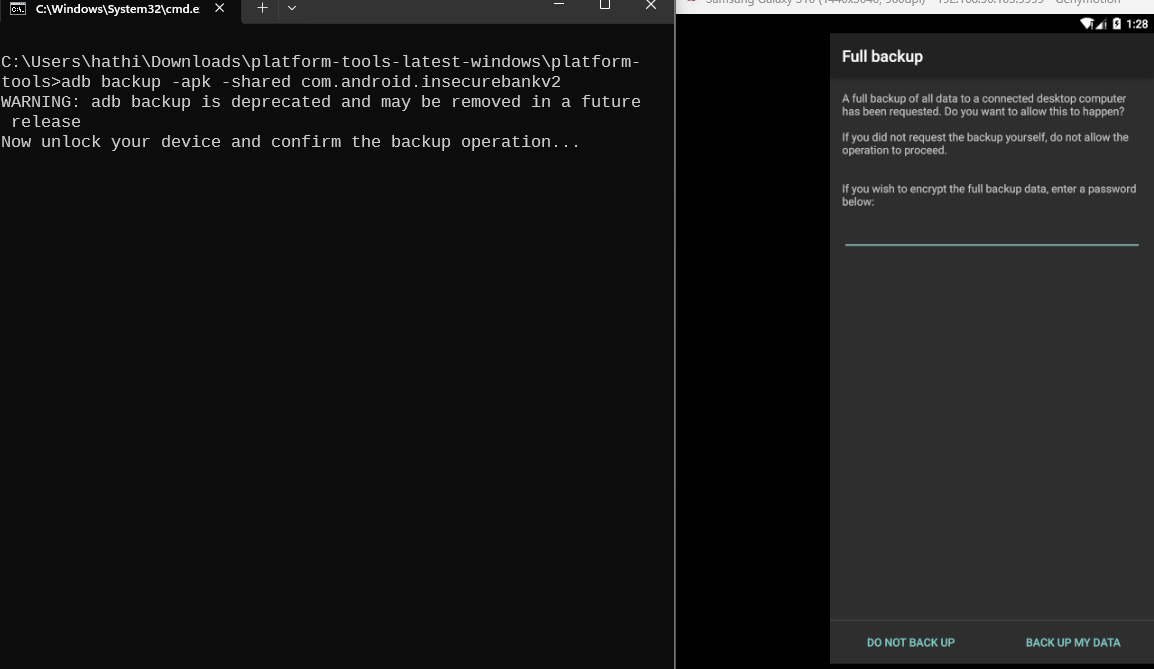
## **Yêu cầu 3: Kiểm tra xem thông tin nhạy cảm có lưu lại trên thiết bị hay không? Một số từ khoá: deviceId, userId, imei, deviceSerialNumber, devicePrint, phone, XDSN, mdn, IMSI, uuid…**

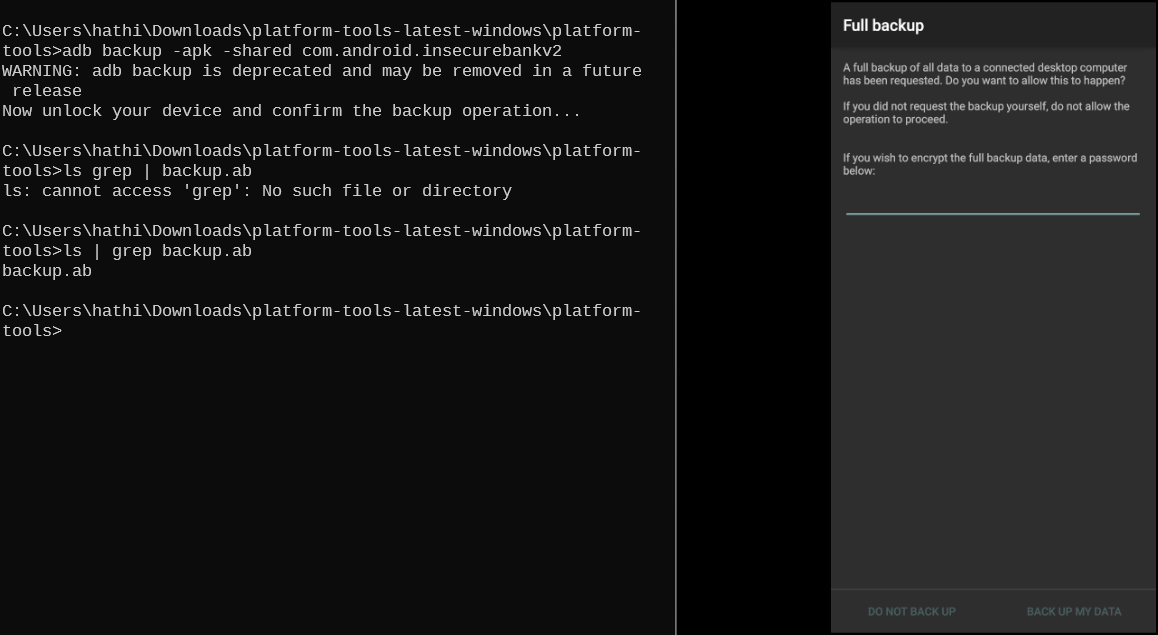


* Dựa vào đầu ra của câu lệnh grep, không có dấu hiệu của thông tin nhạy cảm được tìm thấy trong các tệp tin trên thiết bị. Cụ thể, không có kết quả nào được trả về cho các từ khóa nhạy cảm như deviceId, userId, imei, deviceSerialNumber, devicePrint, phone, XDSN, mdn, IMSI, uuid. Điều này cho thấy rằng thông tin nhạy cảm này có thể không được lưu trữ trong các tệp tin hoặc thư mục mà grep đã tìm kiếm.

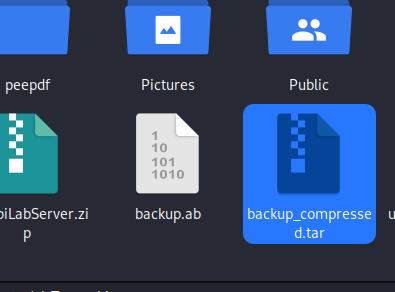
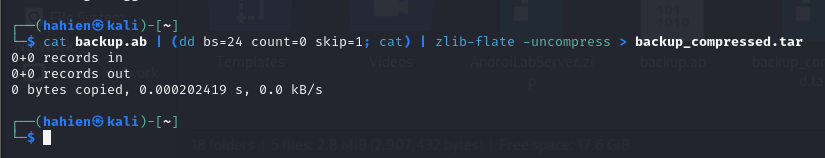
1. **Yêu cầu 4: Theo bạn thư mục sao lưu chứa thông tin nào cần mã hoá, chỉ ra.**

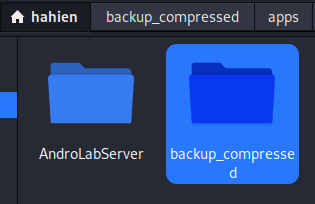
* Đăng nhập với tài khoản người dùng bình thường (dinesh/Dinesh@123$ hoặc jack/Jack@123$)
* Thực thi lệnh sau để sao lưu
* Tại giao diện điện thoại tiến hành cho phép backup



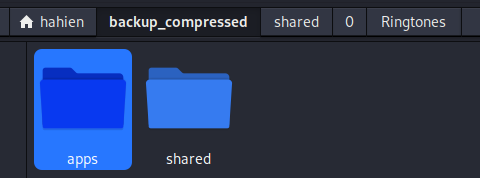


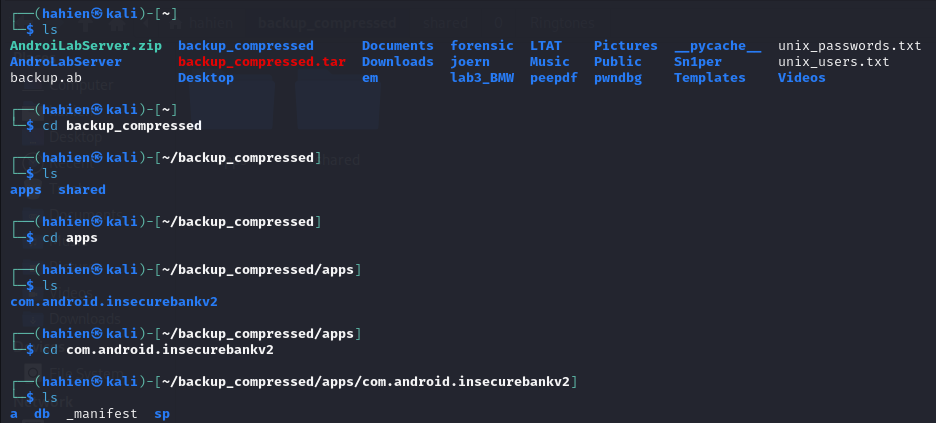
* Copy qua máy ảo kali
* Chuyển đổi tập tin sao lưu qua định dạng có thể đọc được (cài gói qpdf)





* Xem qua thì thư mục share rỗng



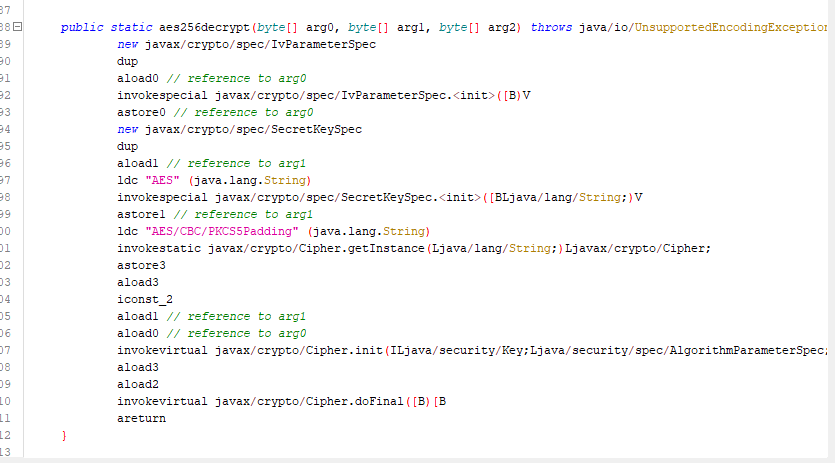
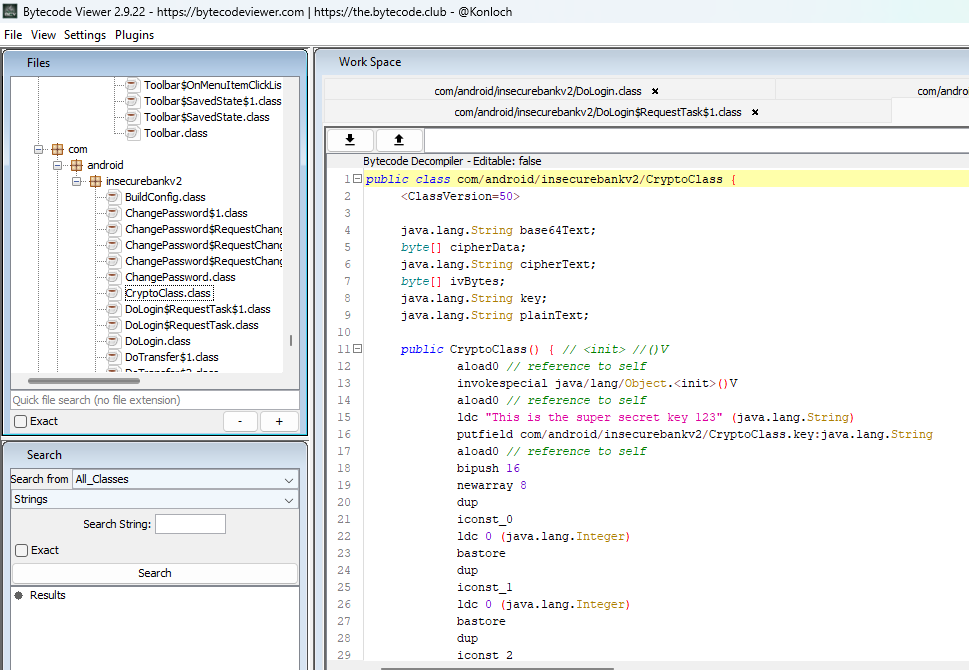


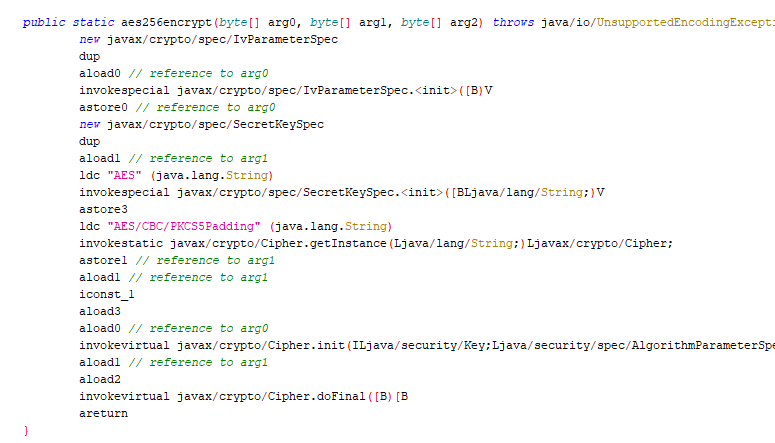


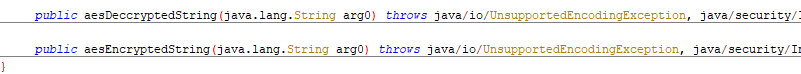
Vậy thư mục cần được mã hoá là thư mục db (database), để bảo về thông tin người dùng như chúng ta đã nói ở câu 2.

1. **Yêu cầu 5: Viết chương trình giải mã đoạn dữ liệu mã hoá (python3 chẳng hạn…)**

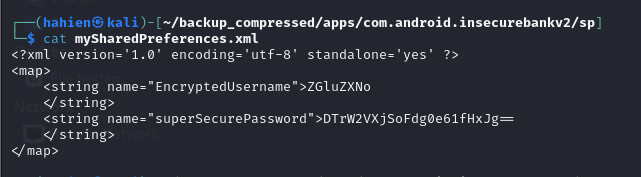
* Tại đường dẫn com/android/insecurebankv2/CryptoClass.class, chứa các hàm mã hoá và giả mã.
* Tiếp tục tìm kiếm thông tin liên quan đến cơ chế mã hoá thì ta thấy được mã hoá đang sử dụng là aes cbc, với key là This is the super secret key 123 và iv như hình



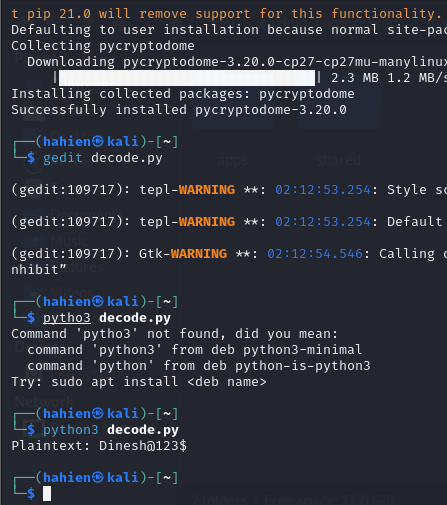
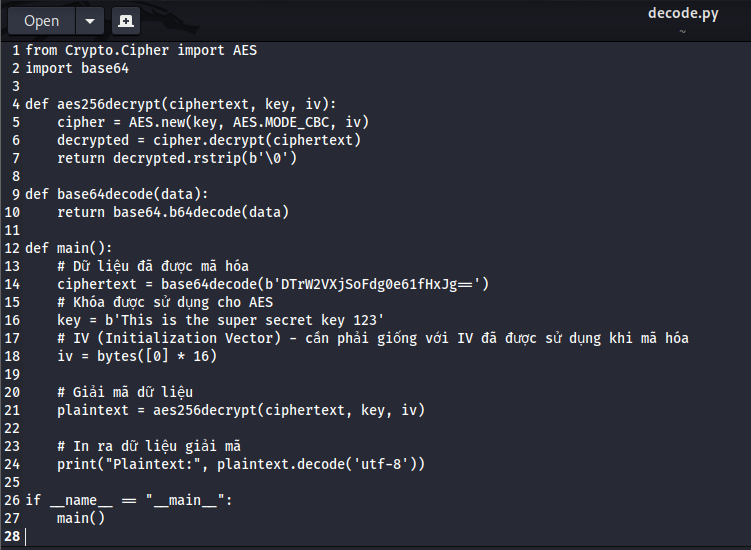




* Ta cũng thấy được các thông tin mã hoá



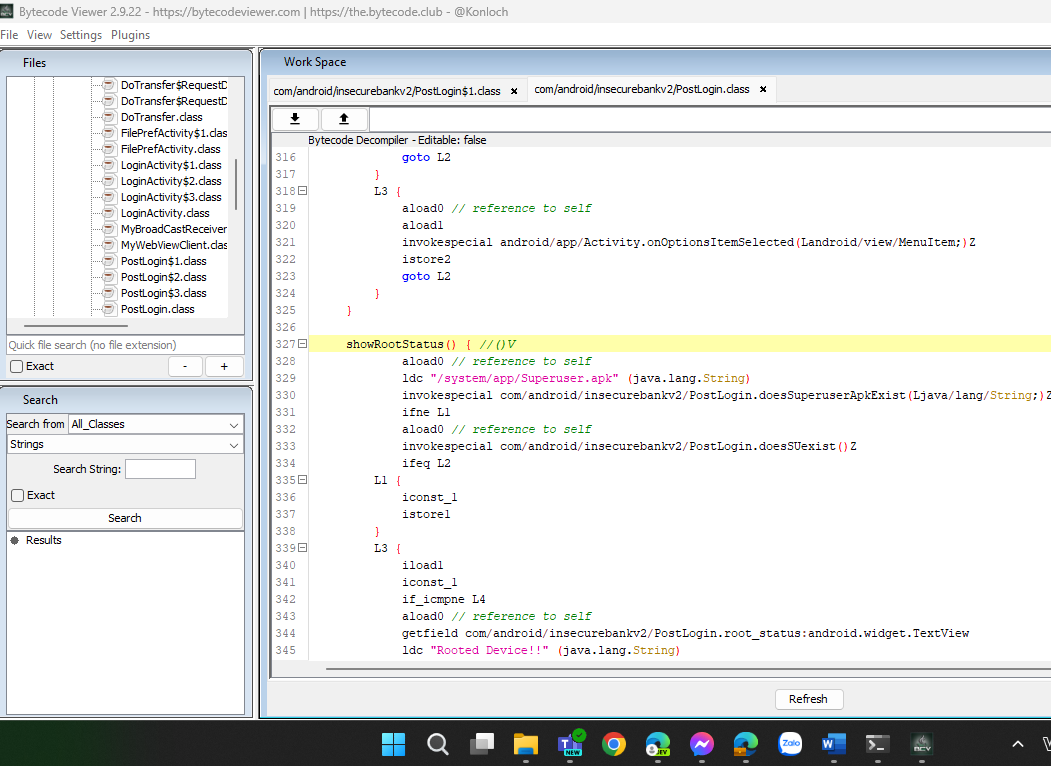
* Viết code để decypt

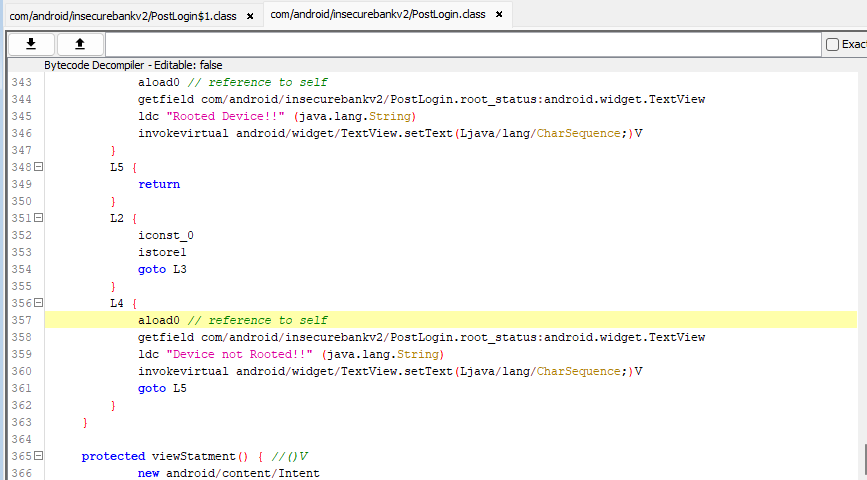


* Dữ liệu mã hoá là mật khẩu của tài khoản dinesh: **Dinesh@123$**

1. **Yêu cầu 6: Sinh viên điều chỉnh mã nguồn ứng dụng sao cho luôn hiển thị trạng thái “Rooted Device!!” với bất kỳ trạng thái nào của thiết bị**

* Xem PostLogin





* Code trên có nhiệm vụ xử lý các hoạt động sau khi người dùng đăng nhập thành công vào hệ thống. Đoạn mã này bao gồm các phương thức để kiểm tra xem thiết bị có root hay không và hiển thị trạng thái này cho người dùng.
* Để điều chỉnh mã nguồn để luôn hiển thị trạng thái "Rooted Device!!" với bất kỳ trạng thái nào của thiết bị, bạn có thể sửa đổi phương thức showRootStatus() như sau:

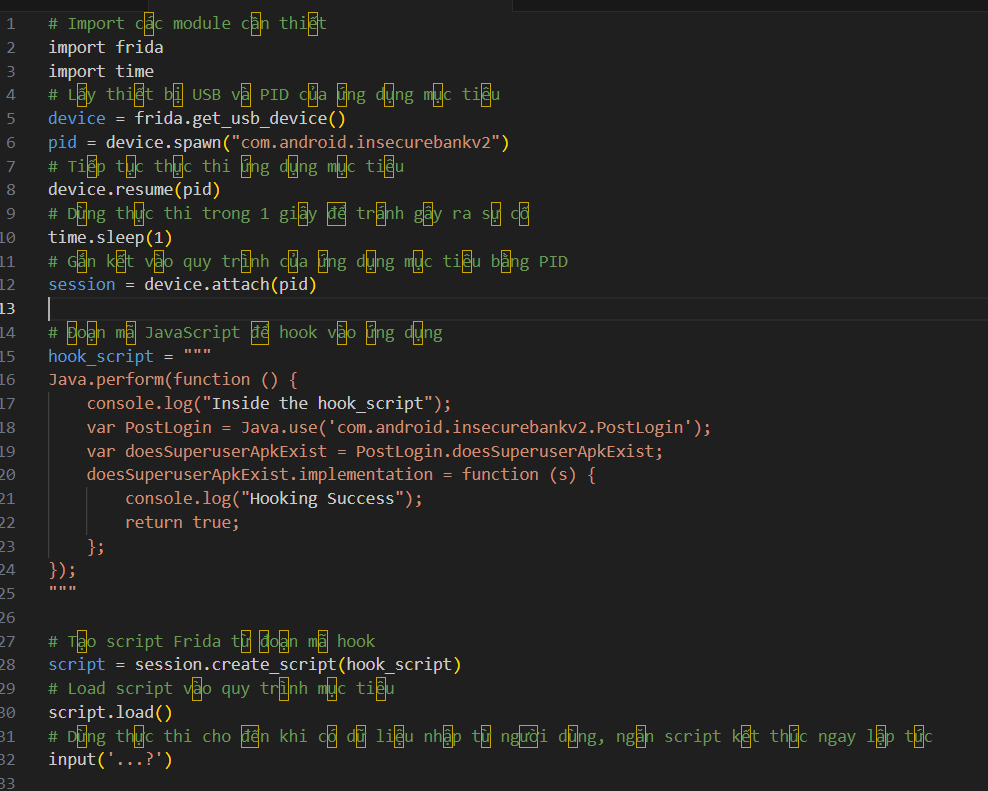


* Đoạn mã được điều chỉnh chỉ khác biệt nhỏ với đoạn mã gốc. Cụ thể:
  + Thay đổi trong phương thức showRootStatus():
    - Trong phương thức showRootStatus() đã được thay đổi để luôn đặt giá trị của biến istore1 thành 1, sau đó hiển thị thông điệp "Rooted Device!!" bằng cách gọi setText() trên root\_status. Điều này đảm bảo rằng mã sẽ luôn hiển thị "Rooted Device!!" cho người dùng.
    - Xóa điều kiện kiểm tra trạng thái root trong phương thức onCreateOptionsMenu() và onOptionsItemSelected():
* Trong các phương thức này, không còn có các điều kiện kiểm tra trạng thái root nữa. Các phương thức này không liên quan đến việc hiển thị trạng thái root, vì vậy việc loại bỏ các điều kiện này không ảnh hưởng đến hiệu suất hoặc chức năng của ứng dụng.
* Những thay đổi này đảm bảo rằng mã sẽ luôn hiển thị "Rooted Device!!" cho người dùng mà không quan tâm đến trạng thái thực tế của thiết bị.



1. **Yêu cầu 7: Hoàn thiện đoạn code trên và demo.**

* Code:

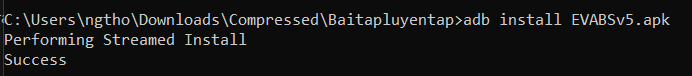


* Giải thích:
* Hook hàm*doesSuperuserApkexist()* để làm cho giá trị trả về của nó bằng true. Sử dụng *Java.use()* để ghi đè nội dung của phương thức này và trả về rooted trong mọi trường hợp.
* Phương thức *doesSuperuserApkexist*() có nhiệm vụ kiểm tra xem thiết bị Android có root hay không. Bằng cách ghi đè nó với lệnh *return true*, dù thiết bị có root hay không, phương thức này sẽ luôn trả về *true*. Khi điều này xảy ra, ứng dụng sẽ hiển thị thông báo "is Rooted".

1. **Challenge 1: Hoàn thành 12 levels EVABSv4:**

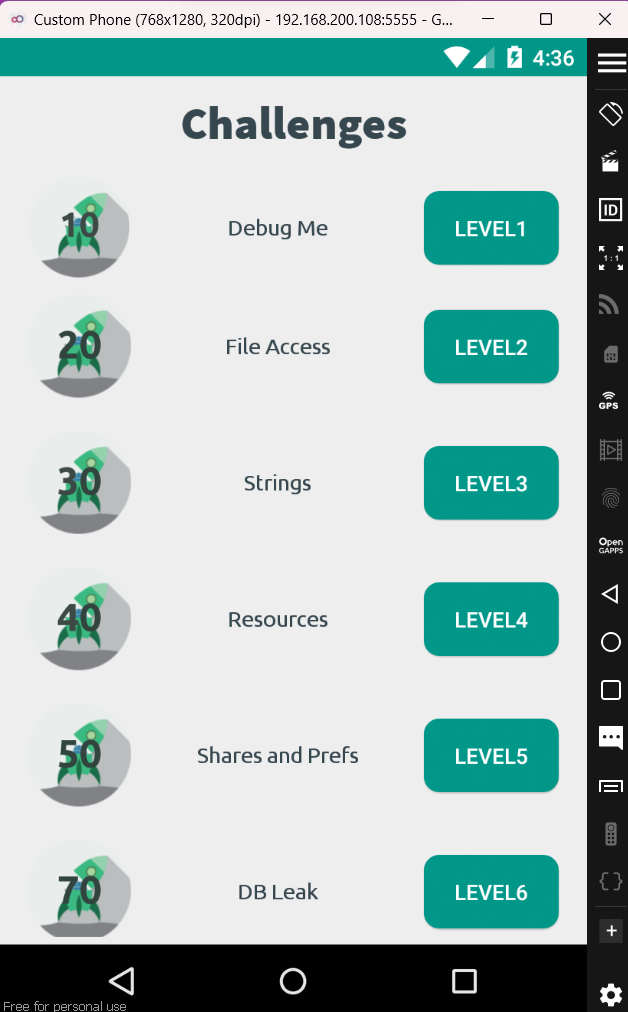
Link challenge: <https://github.com/abhi-r3v0/EVABS>

Sau khi được cung cấp file apk, em cần chạy ứng dụng trên thiết bị Android hoặc thiết bị giả lập. Set up môi trường và điện thoại ảo với Genymotion:



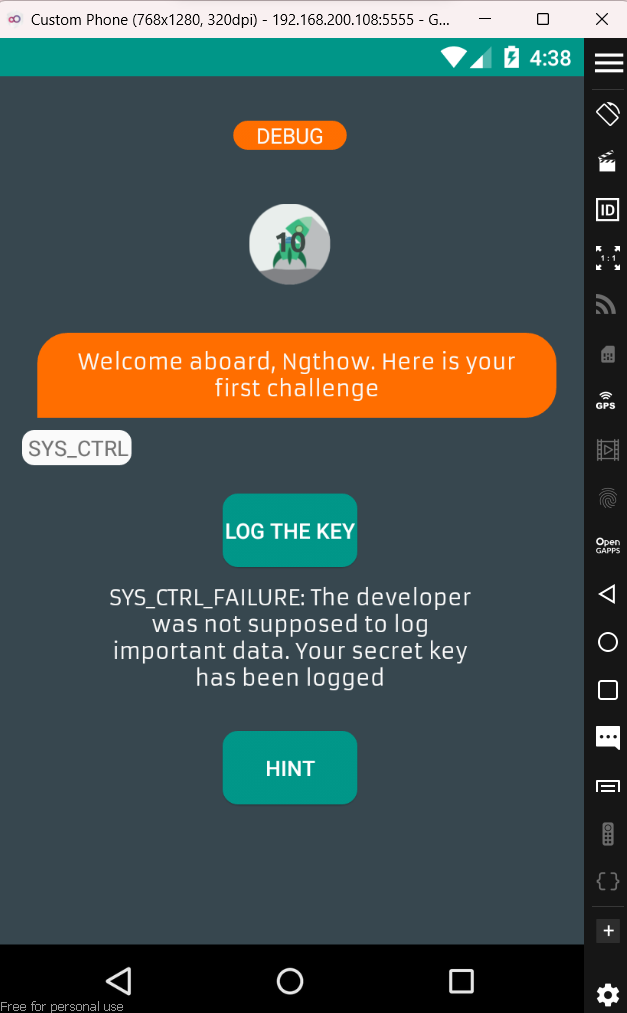
Cài đặt thành công:

Giao diện có 2 phần, một bên là list challenges, còn lại là *Flag check*. Sau khi nhấn chọn *Challenges*, tất cả challenge sẽ hiện ra:



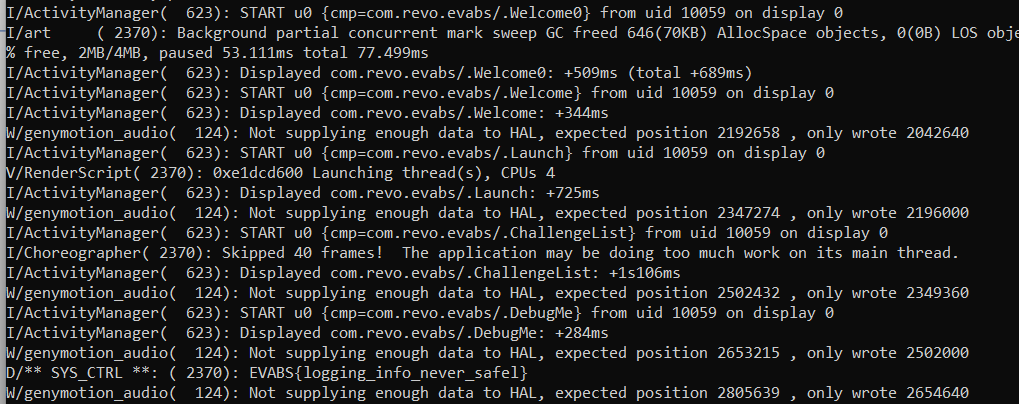
1. **Challenge 1: Debug me:**

* Quan sát thử thách: Ở level này, theo hint, em cần kiểm tra log của hệ thống bằng ADB.

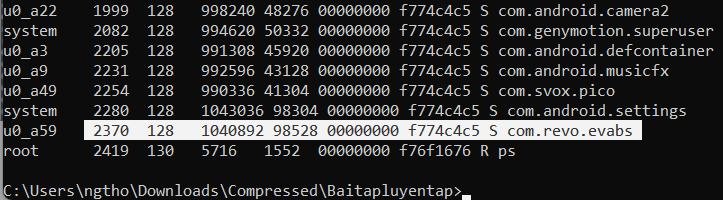


Sau khi research, em biết được *Logcat* là một công cụ dòng lệnh giúp kết xuất nhật ký thông báo của hệ thống, bao gồm các thông báo mà bạn đã viết từ ứng dụng của mình bằng lớp *Log*. Và cú pháp thường dùng để chạy *logcat* là thông qua shell *adb (android debug bridge)*. Do đó câu lệnh đơn giản nhất nên thử là ***adb logcat***

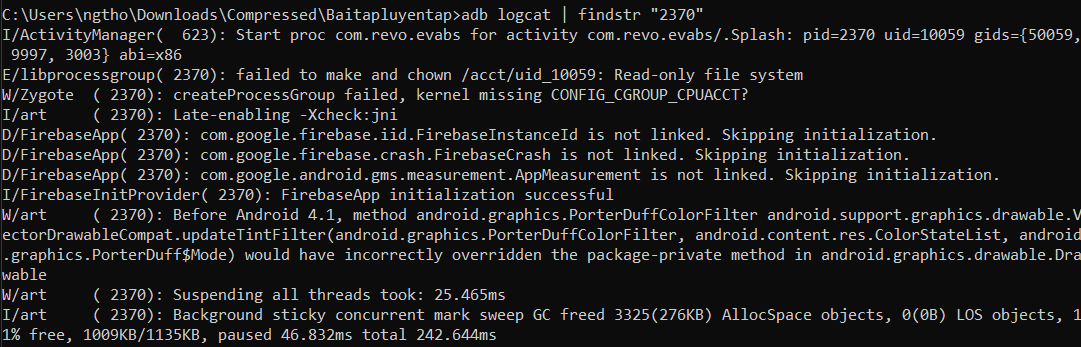
* Thực hiện:
* Chạy lệnh ***adb logcat***, được kết quả tuy nhiên rất nhiều nên khó quan sát:



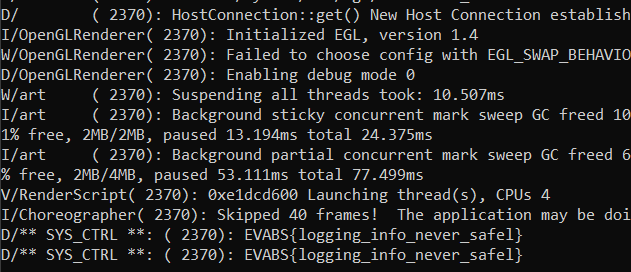
* Dùng lệnh ***adb shell ps*** để tìm PID của ứng dụng, từ đó tìm kiếm log dựa trên PID sẽ nhanh hơn:



* Tìm kiếm tiến trình với PID=2370:



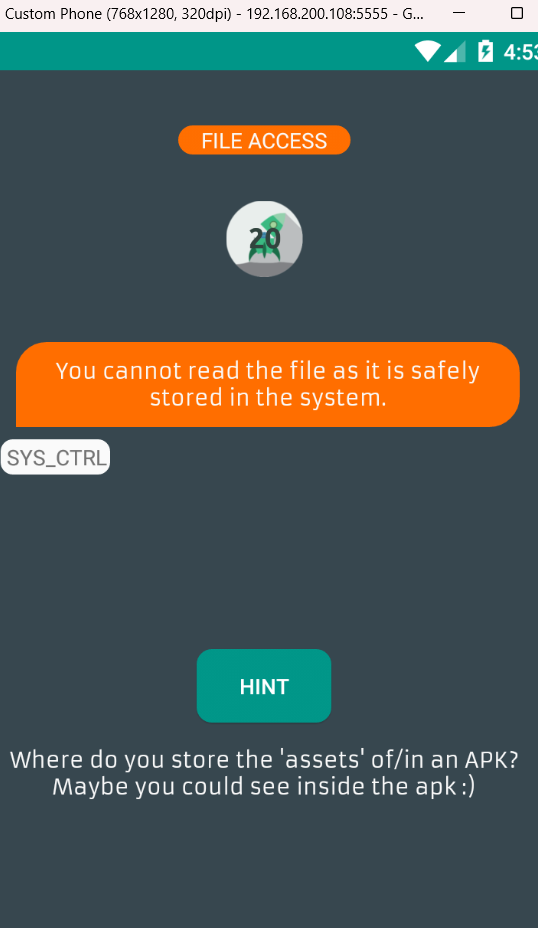
* Quay lại điện thoại ảo, nhấn nút “Log the key”, flag sẽ hiển thị trên terminal:



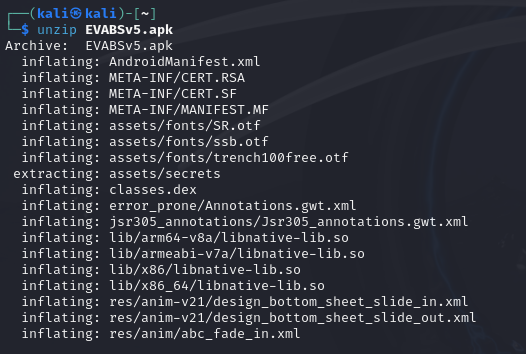
Flag: **EVABS{logging\_info\_never\_safel}**

1. **Challenge 2: File access:**

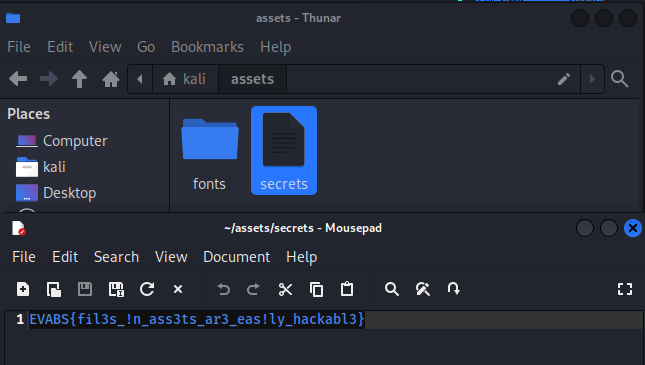
* Challenge:



* Thực hiện:
* Hint có đề cập đến thư mục “assests” trong file APK, đây là một thư mục lưu những tài nguyên không được biên dịch như hình ảnh, âm thanh,... Gợi ý cũng chỉ ra file assets có thể được tìm thấy trong file apk. Do đó, em sẽ unzip file apk trước:



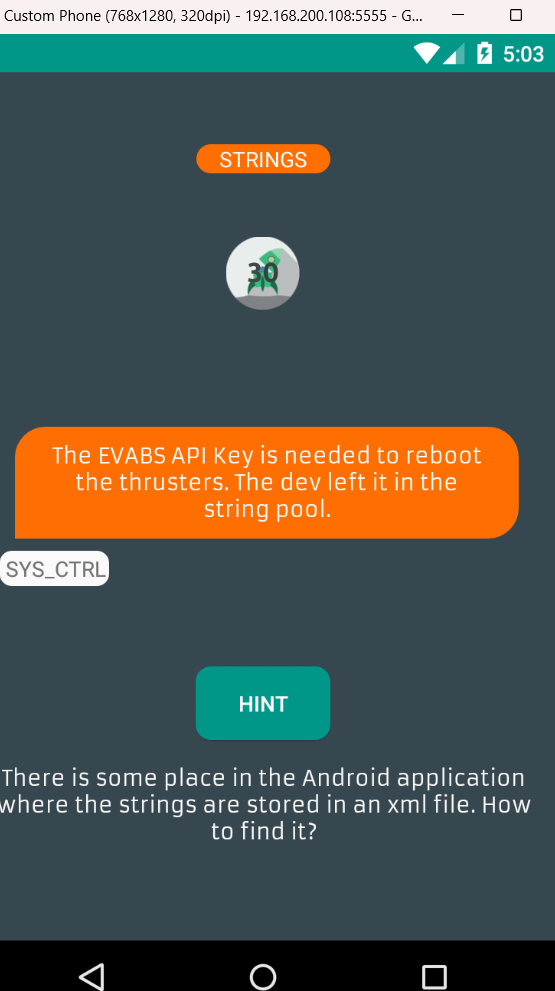
* Trong thư mục *assets* có file *secrets*, mở file tìm được flag:



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

1. **Challenge 3: Strings**

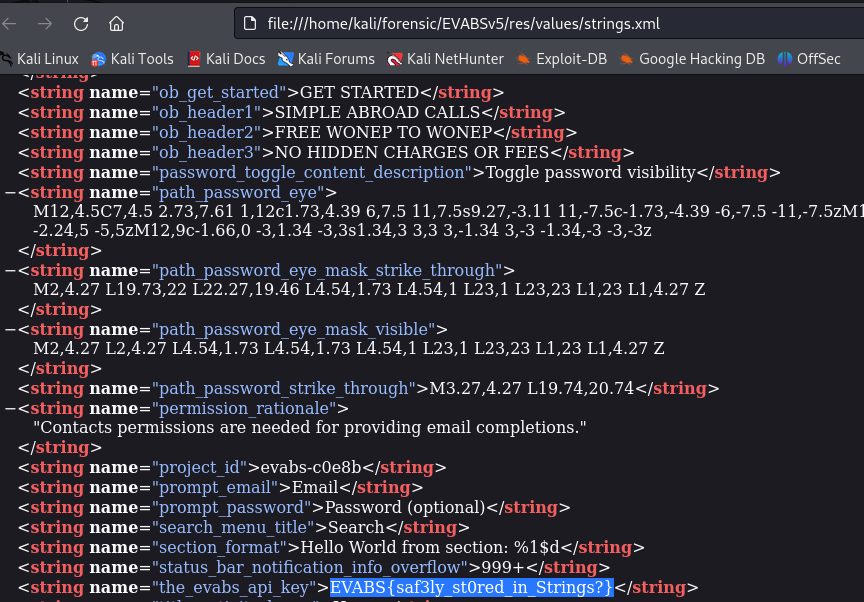
* Challenge:



* Thực hiện:
* Challengs đề cập đến strings. Thông thường trong source các ứng dụng android, strings sẽ được lưu trong file *strings.xml* tại thư mục *res/values/*. Để xem được nội dung trong file này, trước hết, cần dịch ngược mã nguồn (decompile) bằng apktool:

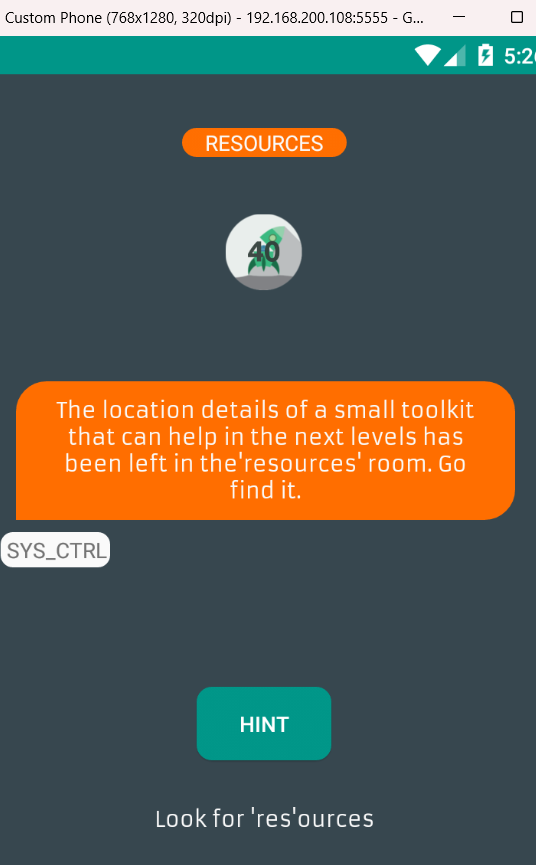


* Vào thư mục *res/values/* để xem file *strings.xml*:

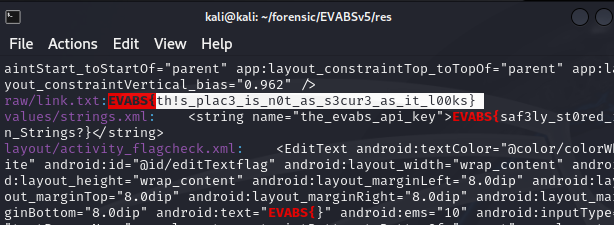
Flag: **EVABS{saf3ly\_st0red\_in\_Strings?}**

1. **Challenge 4: Resource:**

* Challenge:



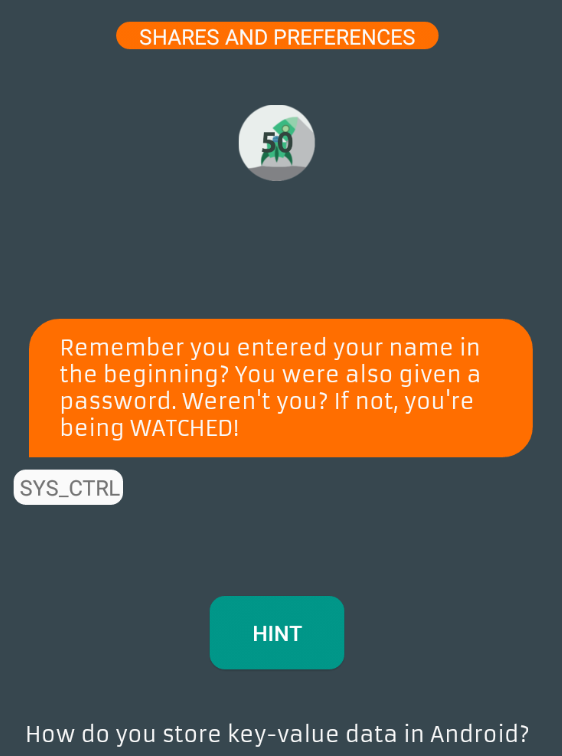
* Thực hiện:
* Sử dụng kết quả decompile của bài trước để giải quyết bài này. Tìm trong thư mục *res* nội dung “EVABS{“ với câu lệnh grep –r “EVABS{”. Được kết quả:



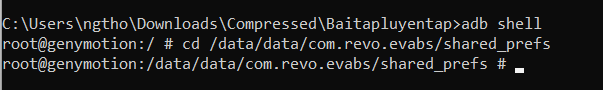
Flag: **EVABS{th!s\_plac3\_is\_n0t\_as\_s3cur3\_as\_it\_l00ks}**

1. **Challenge 5: Shares and Prefs**

* Challenge:



* Thực hiện:
* Hint chỉ ra vấn đề trong challenge này là Shared Preferences. Shared Preferences là 1 cách lưu trữ những dữ liệu dạng nguyên thủy (bool , int , float , double , String) trong Android dưới dạng file XML với những cặp key/value. Để làm việc với Shared Preferences, Android SDK có cung cấp 2 class là SharedPreferences và SharePreferences. Thông thường, chúng sẽ được lưu tại */data/data/<package\_name>/shared\_prefs/\*.xml.*
* Sử dụng adb shell để truy cập vào máy và thư mục trên:

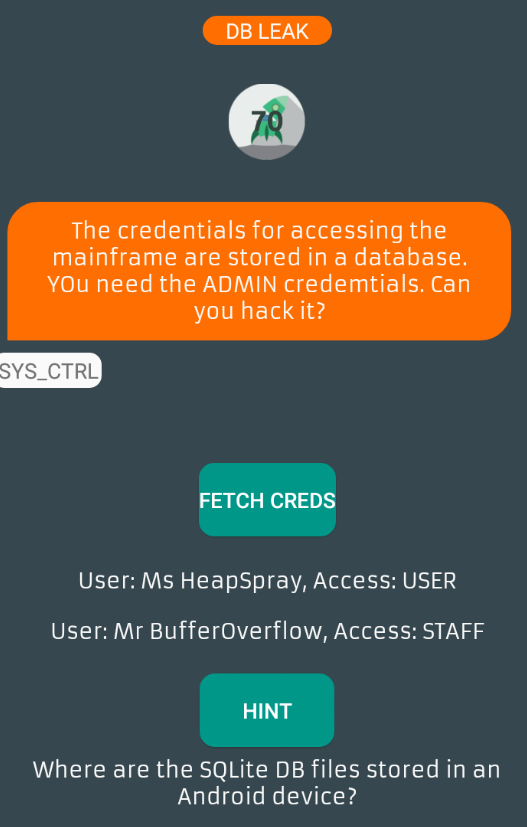


* Dùng lệnh *findstr.exe /s /i "EVABS{" \**, tìm được flag là:

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

1. **Challenge 6: DB Leak**

* Challenge:



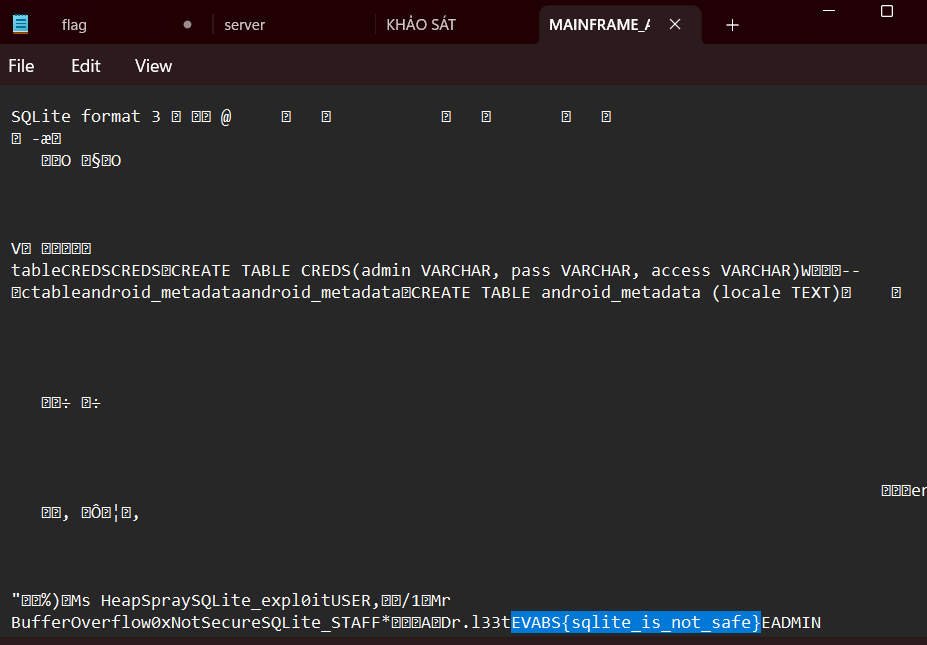
* Thực hiện:
* Thư mục mà cơ sở dữ liệu SQLite được lưu trữ là */data/data/<package\_name>/databases/*, trong đó <package\_name> là ID gói của ứng dụng.
* Chạy thử lệnh sau để kiểm tra các db có trong ứng dụng:



* Pull db về để xem được nội dung trong db:

*adb pull "/data/data/com.revo.evabs/databases/MAINFRAME\_ACCESS"*

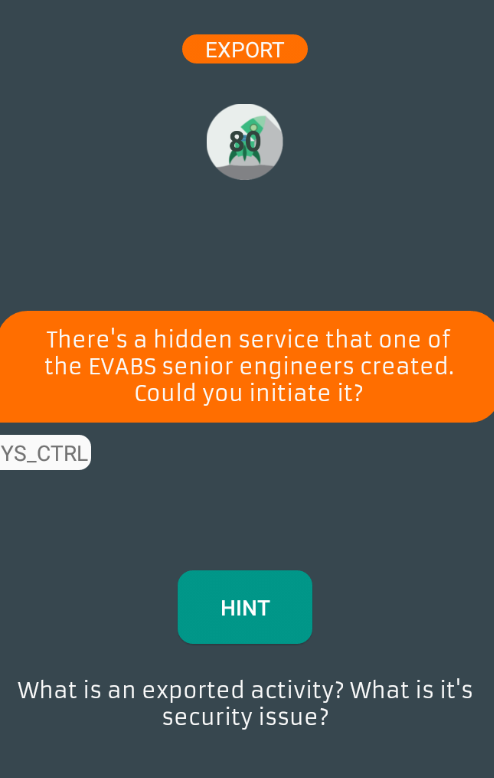
* Mở file MAINFRAM\_ACCESS trên notepad, em tìm được flag:



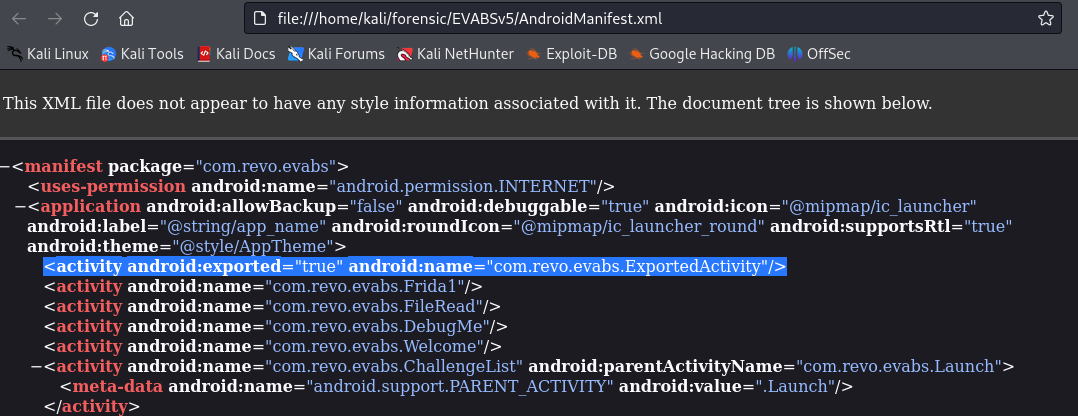
Flag: **EVABS{sqlite\_is\_not\_safe}**

1. **Challenge 7: Export**

* Challenge:

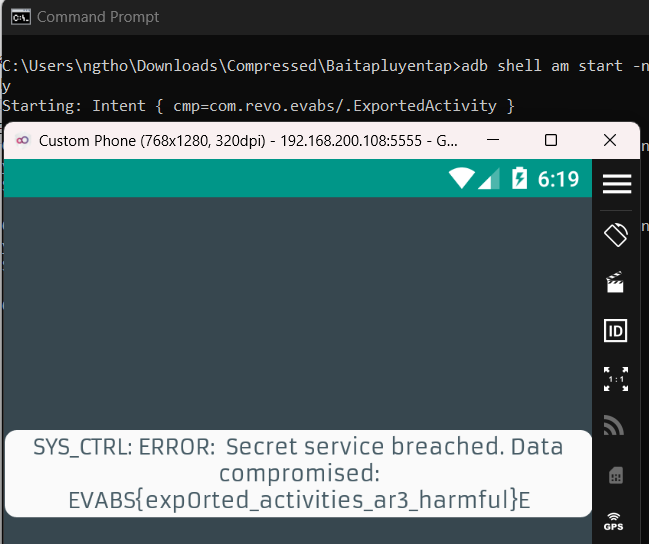


* Thực hiện:
* Một hoạt động (activity) được gọi là "exported" khi nó được khai báo trong tệp manifest của ứng dụng với thuộc tính android:exported="true". Thuộc tính này cho phép các thành phần khác từ các ứng dụng hoặc từ hệ thống Android truy cập vào hoạt động đó mà không cần quyền xác nhận. Khi một hoạt động được đánh dấu là "exported", nó trở nên công khai và có thể được khởi chạy bởi các thành phần bên ngoài của ứng dụng hoặc từ các ứng dụng khác trên cùng một thiết bị.
* Trước hết, em sẽ kiểm tra file *AndroidMainfest.xml,* chú ý các thông tin liên quan đến *exported*:



* Khi sử dụng adb để kích hoạt một hoạt động đã được exported, ta sẽ cần sử dụng lệnh *adb shell am start* cùng với tên gói của ứng dụng và tên của hoạt động muốn kích hoạt:

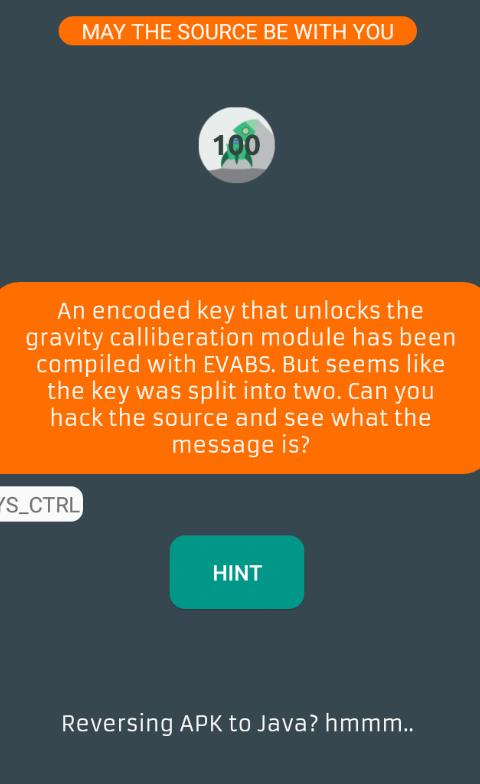
*adb shell am start -n com.revo.evabs/com.revo.evabs.ExportedActivity*



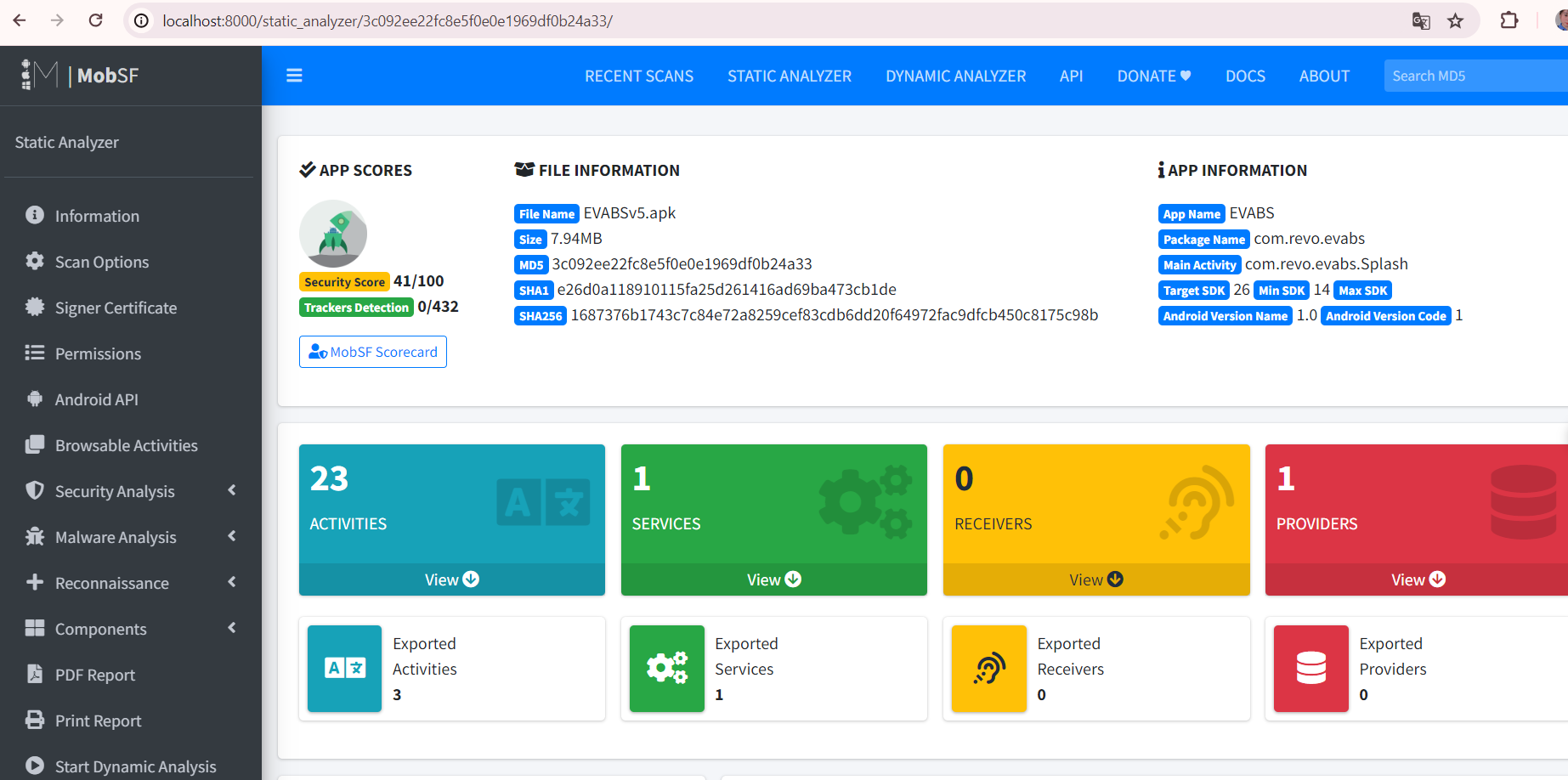
Flag: **EVABS{exp0rted\_activities\_ar3\_harmful}**

1. **Challenge 8: Decode**

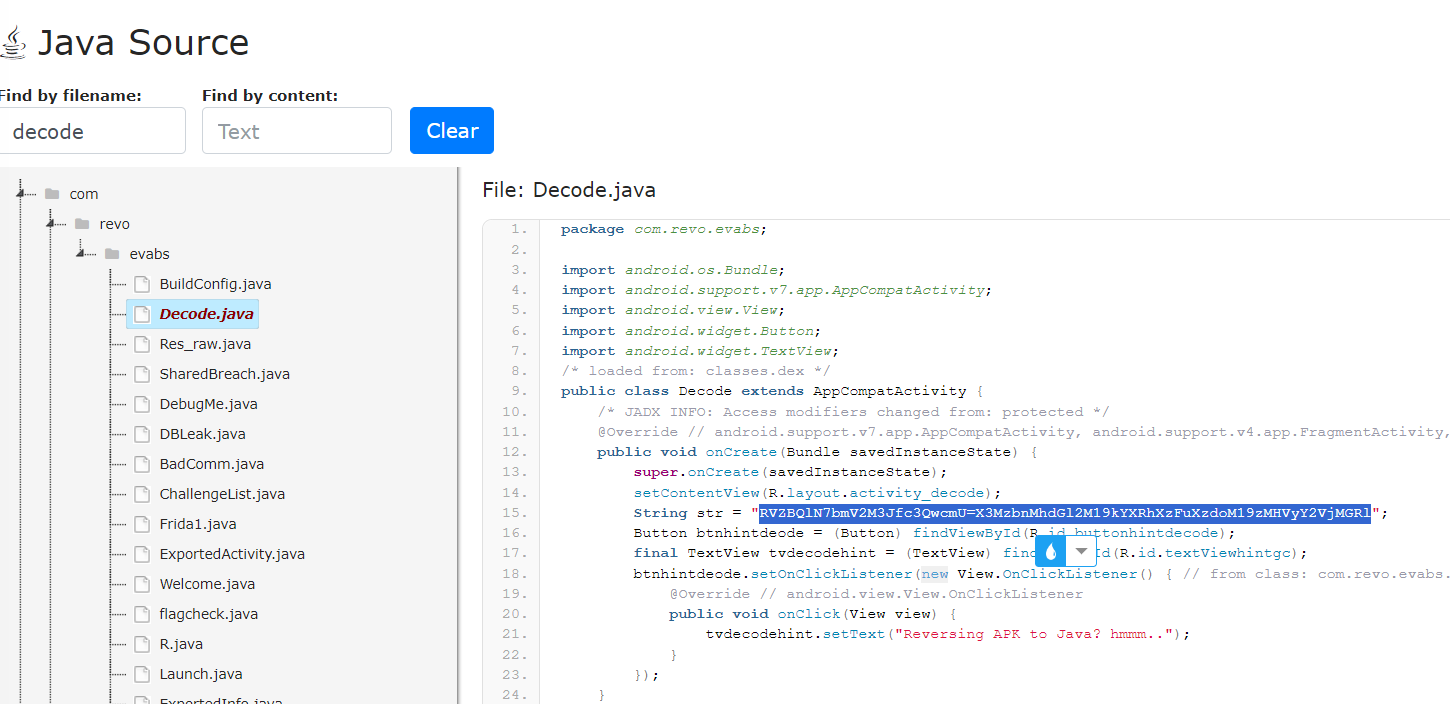
* Challenge:



* Thực hiện:
* Sử dụng MoBSF để phân tích, decompile và chuyển đổi thành java:



* Mở file *decode.java*, thấy được chuỗi đã mã hóa base64 như sau:



* Sử dụng CyberChef để decode:

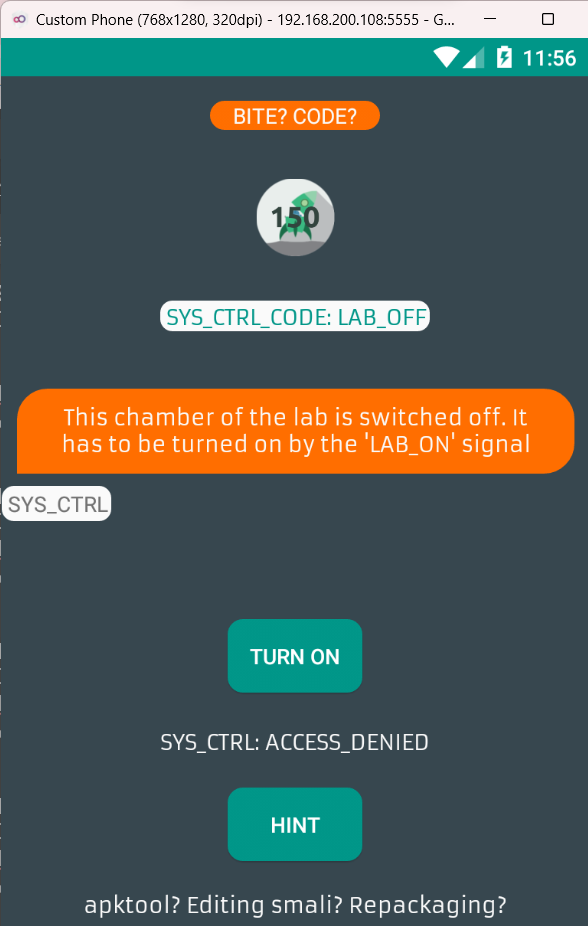


* Vậy flag là:

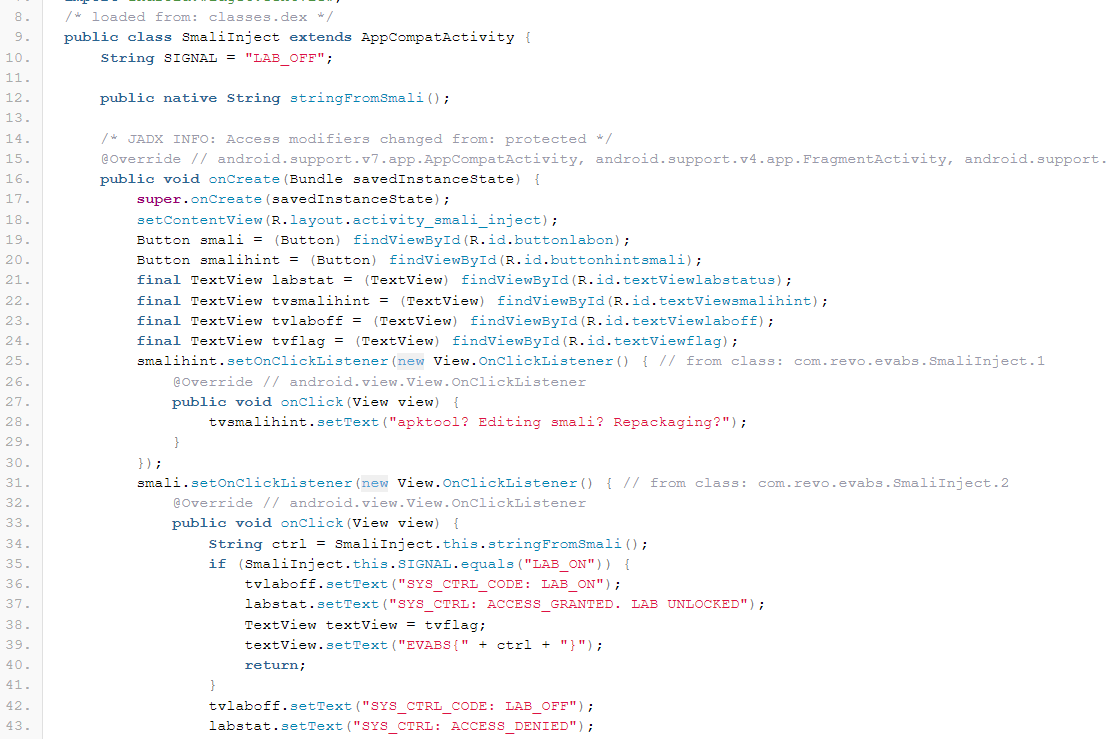
**EVABS{nev3r\_st0re\_s3ns!tiv3\_data\_1n\_7h3\_s0urcec0de}**

1. **Challenge 9: Smali injection**

* Challenge:

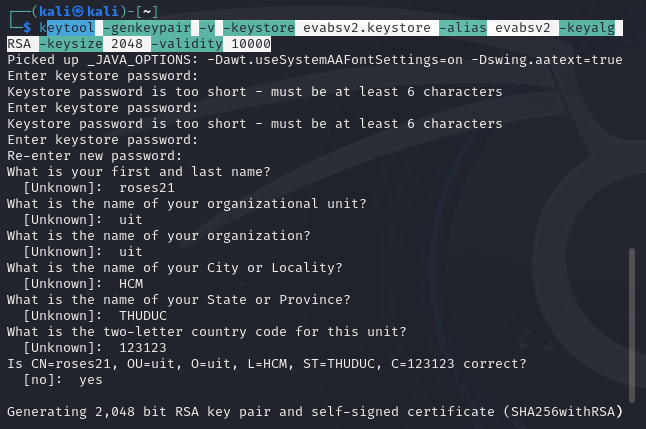


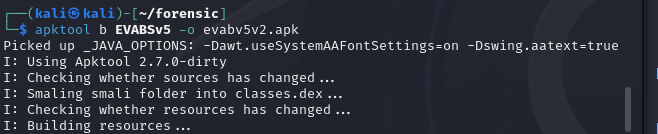
* Thực hiện:
* Dùng MoBSF để xem source và tải code, tìm được file *SmaliInject.java* liên quan đến smali injection:



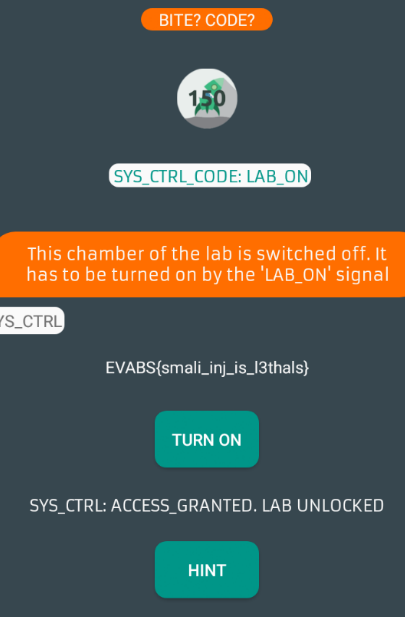
* Quan sát code trên, thấy có một chuỗi SIGNAL được khai báo và khởi tạo với giá trị “LAB\_OFF”, khi đó, sẽ không có hiện tượng gì xảy ra vì không có hàm làm thay đổi SIGNAL và hiện thông báo “ACCESS\_DENIED”. Nếu SEGNAL là “LAB\_ON” sẽ in ra flag. Nhiệm vụ của lab này là đổi giá trị của SIGNAL thành “LAB\_ON”.
* Đổi giá trị SIGNAL trong code smali.
* Tạo chữ ký:

*keytool -genkeypair -v -keystore evabsv2.keystore -alias evabsv2 -keyalg RSA -keysize 2048 -validity 10000*





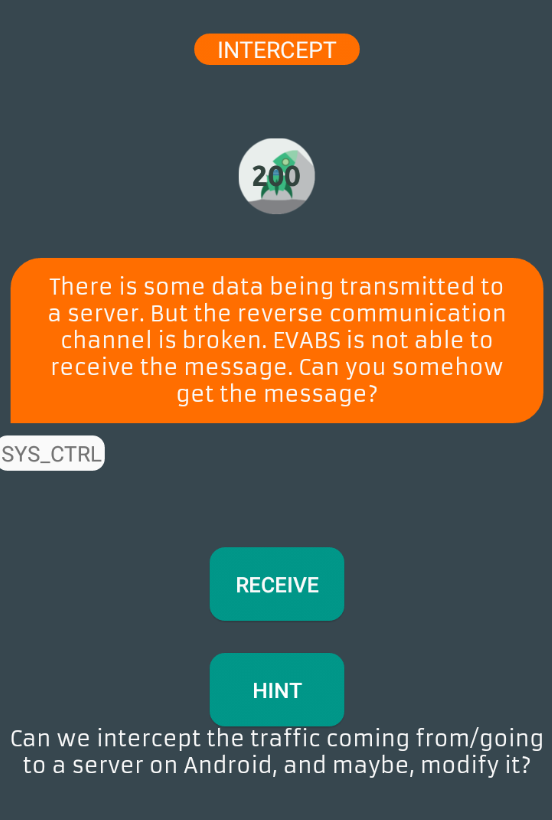
* Cài đặt apk mới lên điện thoại, nhấn “Turn on”:



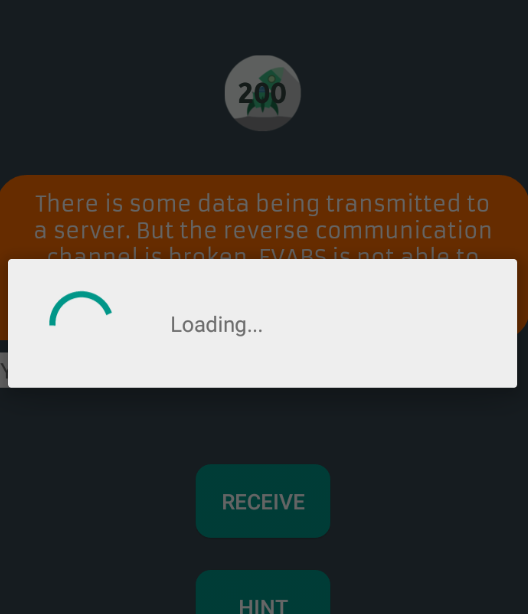
Flag: **EVABS{smali\_inj\_is\_I3thaIs}**

1. **Challenge 10: Interception**

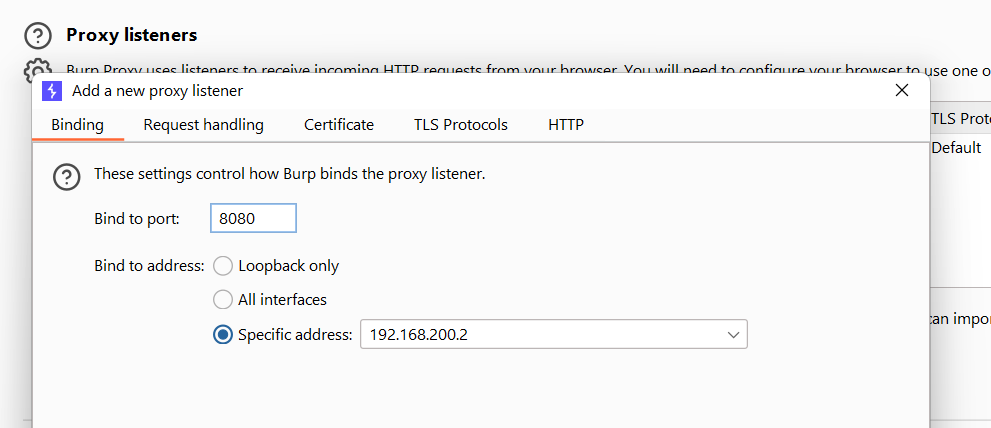
* Challenge:



* Thực hiện:
* Một tool phổ biến hỗ trợ việc intercept chính là Burp Suit. Khi nhấn nút “Receive”, không có gì hiển thị ra màn hình cả:



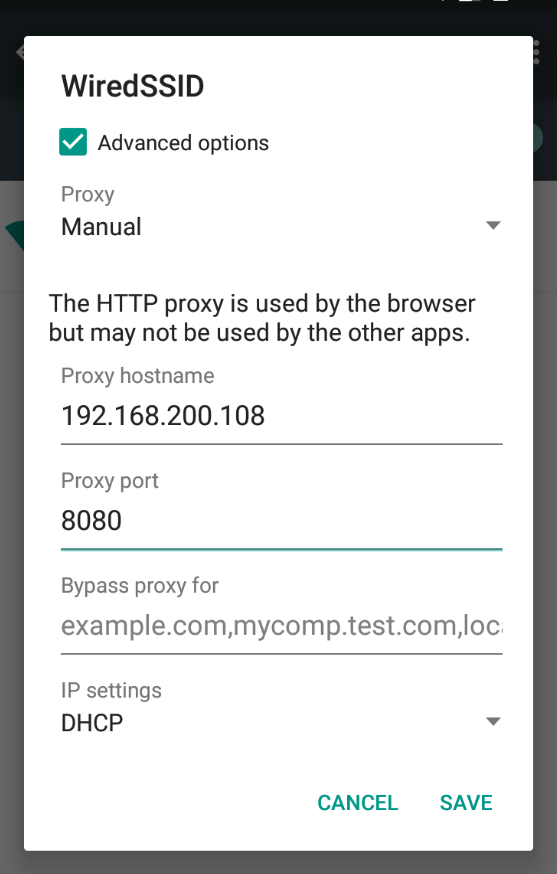
* Mở Burp Suit, cấu hình proxy như sau:



Chọn địa chỉ ip nằm trong vùng mạng của ip điện thoại ảo:



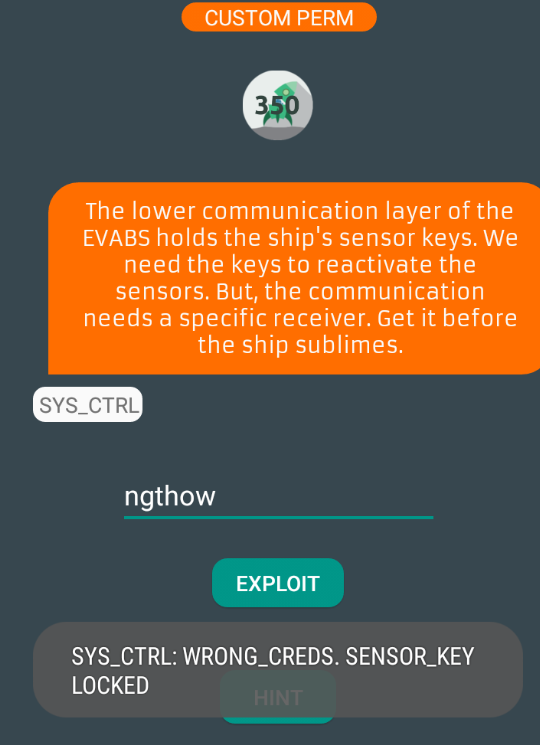
* Cấu hình trên điện thoại ảo để Burp Suit bắt được traffic:



* Loay hoay nhưng mà em vẫn chưa tìm được flag trong traffic ạ:<

1. **Challenge 11: Custom exploit**

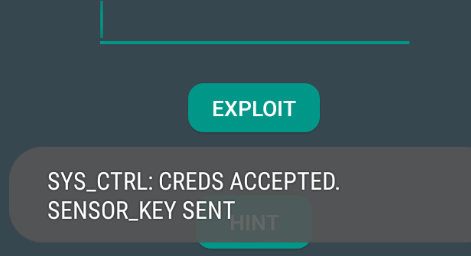
* Challenge:
* Thực hiện:
* Nhập input ngẫu nhiên thì bị báo sai:



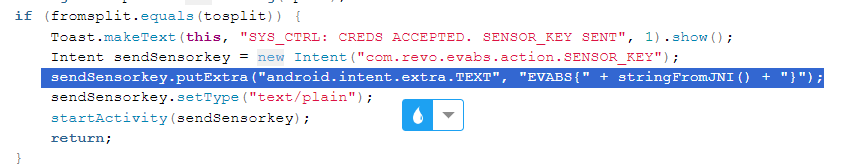
* Cần tìm input để nhập vào, quan sát file *CustomAccess.java*, input em tìm được là *cust0m\_p3rm*:



Kết quả đúng:



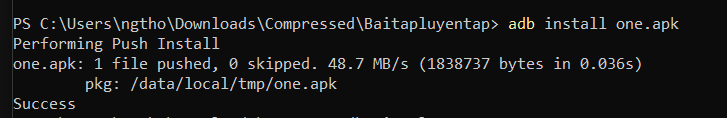
* Sau khi nhập đúng input thì flag sẽ được truyền vào intent *com.revo.evabs.action.SENSOR\_KEY* bằng hàm *putExtra()*:



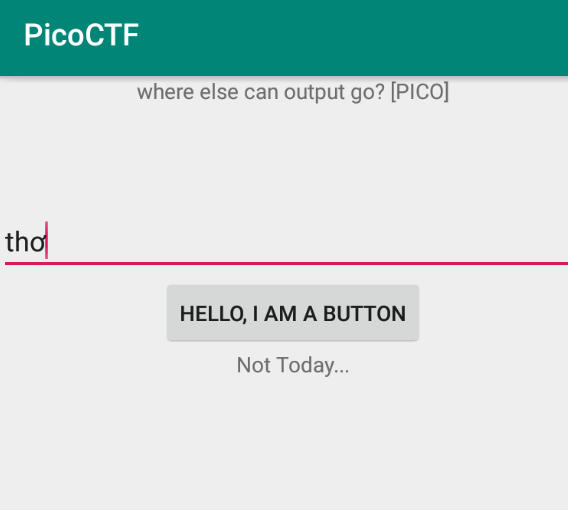
1. **Challenge 2:**

**1. PicoCTF 1:**

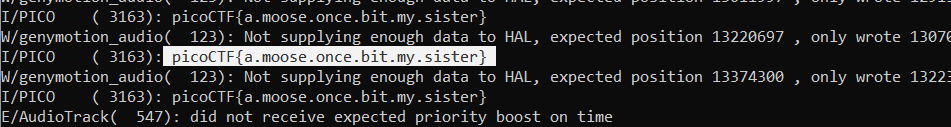
* Cài đặt file apk:



* Em nhập ngẫu nhiên input, màn hình hiển thị:



* Một nơi khác mà output sẽ xuất hiện là log, do đó em sẽ tìm trong log trước, thông qua adb:



* Flag: **picoCTF{a.moose.once.bit.my.sister}**

---

***Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này***

# **YÊU CẦU CHUNG**

* Sinh viên tìm hiểu và thực hành theo hướng dẫn.
* Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
* Sinh viên báo cáo kết quả thực hiện và nộp bài.

**Báo cáo:**

* File .DOCX và .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
* Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
* Đặt tên theo định dạng: [Mã lớp]-SessionX\_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành đã đăng ký với GVHD-TH).

*Ví dụ: [*NT101.K11.ATCL*]-Session1\_Group3.*

* Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
* Không đặt tên đúng định dạng – yêu cầu, sẽ **KHÔNG** chấm điểm bài Lab.
* Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá**: Sinh viên hiểu và tự thực hiện được bài thực hành. Khuyến khích:

* Chuẩn bị tốt.
* Có nội dung mở rộng, ứng dụng trong kịch bản phức tạp hơn, có đóng góp xây dựng bài thực hành.

*Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**

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