BÁO CÁO THỰC HÀNH

**Môn học: Bảo mật web và ứng dụng**

**Session 4: Pentesting Android**

**Applications**

*GVHD: Ngô Đức Hoàng Sơn*

**Nhóm: 08**

1. **THÔNG TIN CHUNG:**

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

Lớp: NT213.P12.ANTT

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | MSSV | Email |
| 1 | Hồ Vỉ Khánh | 22520633 | 22520633@gm.uit.edu.vn |

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

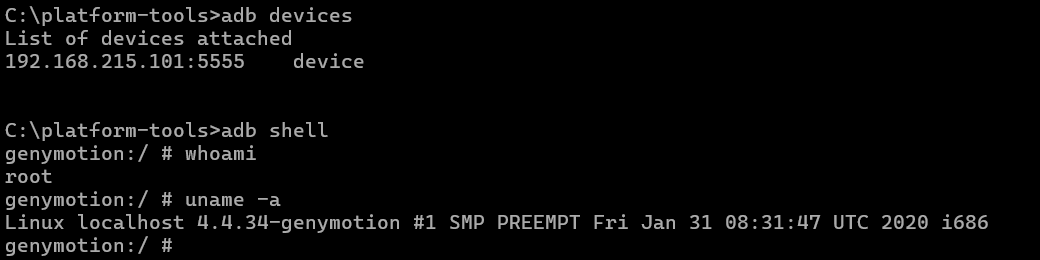
|  |  |  |
| --- | --- | --- |
| STT | Nội dung | Tình trạng |
| 1 | Yêu cầu 1 | 100% |
| 2 | Yêu cầu 2 | 100% |
| 3 | Yêu cầu 3 | 100% |
| 4 | Yêu cầu 4 | 100% |
| 5 | Yêu cầu 5 | 100% |
| 6 | Yêu cầu 6 | 100% |
| 7 | Yêu cầu 7 | 100% |
| Điểm tự đánh giá 10/10 | | |

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

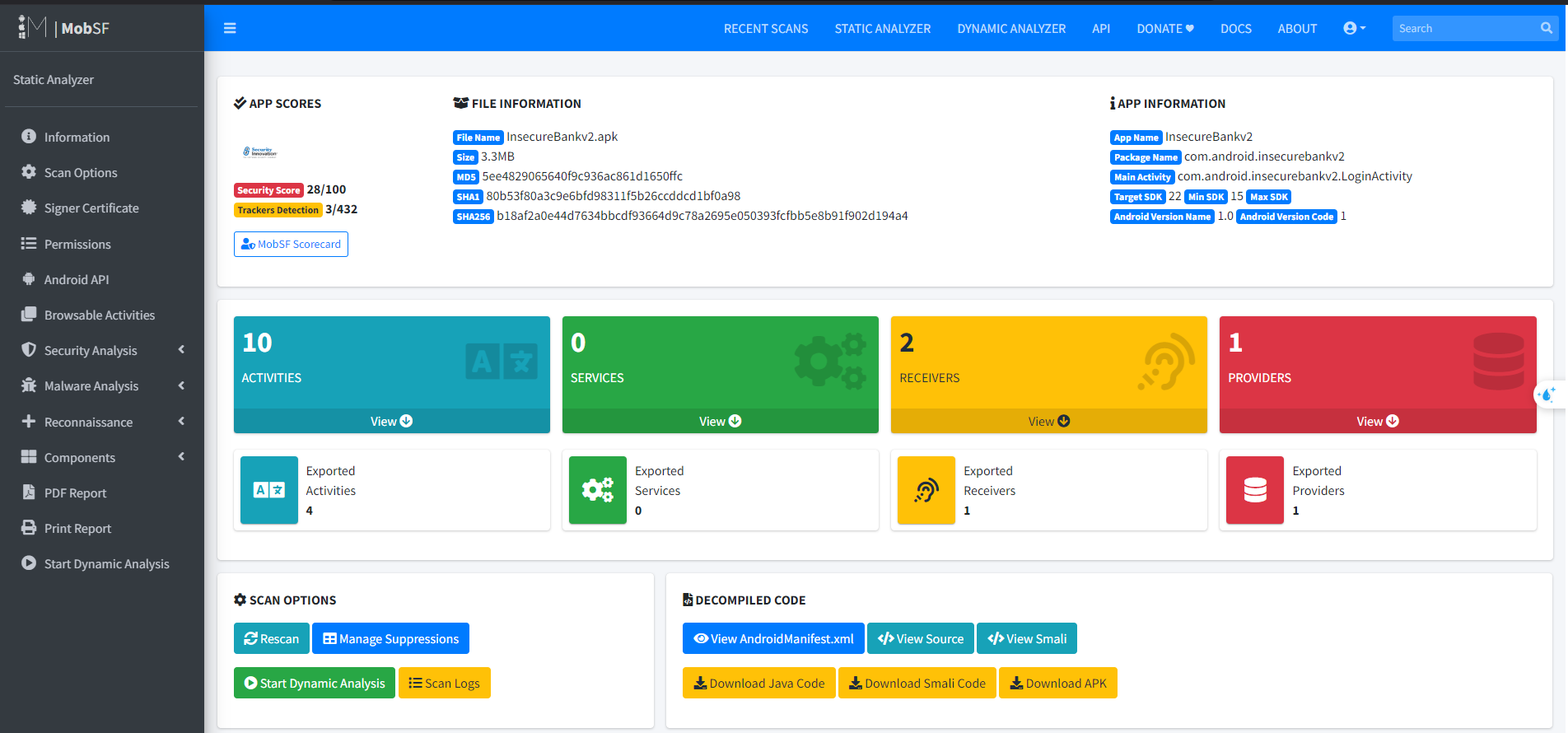
BÁO CÁO CHI TIẾT

# Thực hành

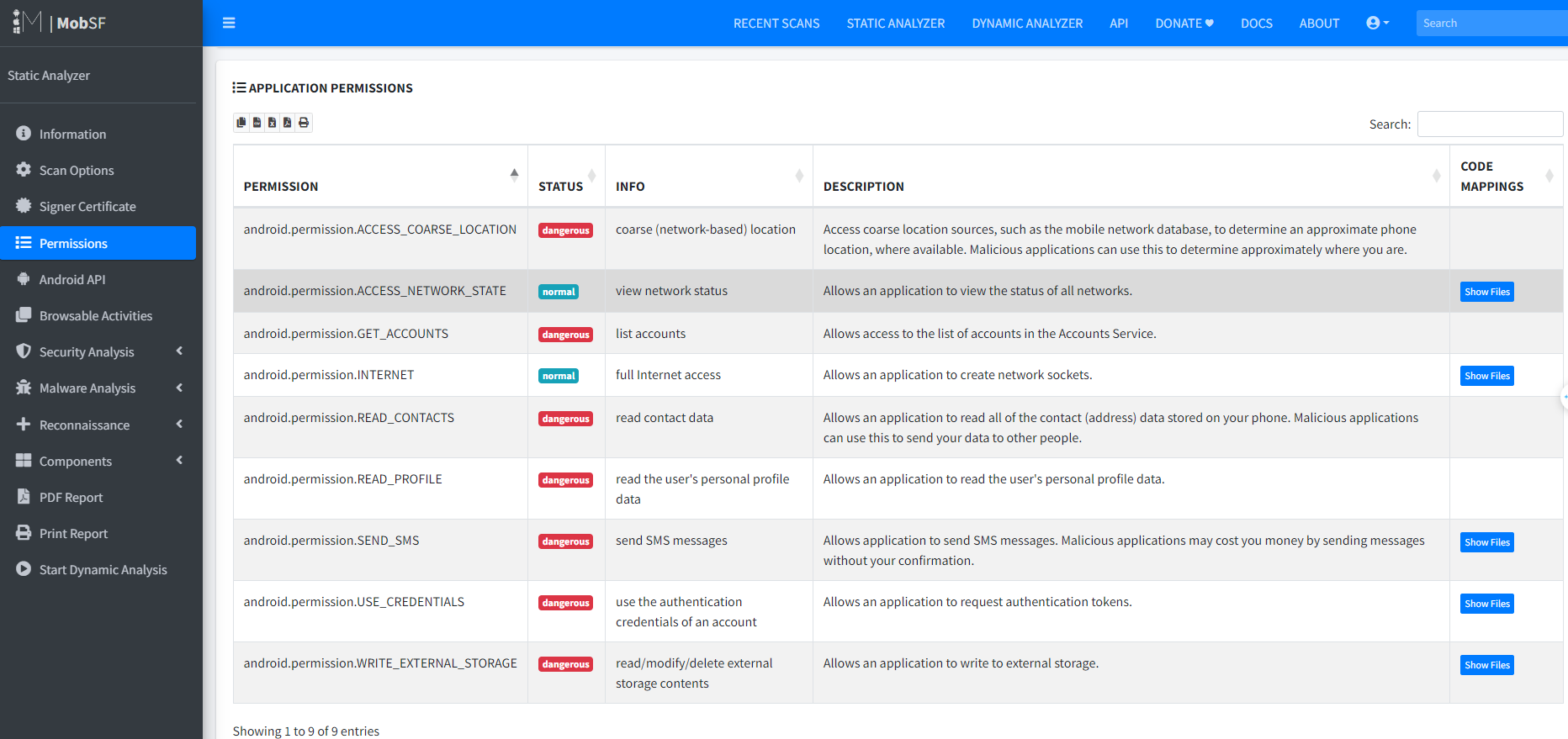
- Cài đặt Android Debug Bridge (ADB)



- Tải lên hoặc kéo thảtập tin InsecureBankv2.apk MobSF để tiếnh hành phân tích tĩnh

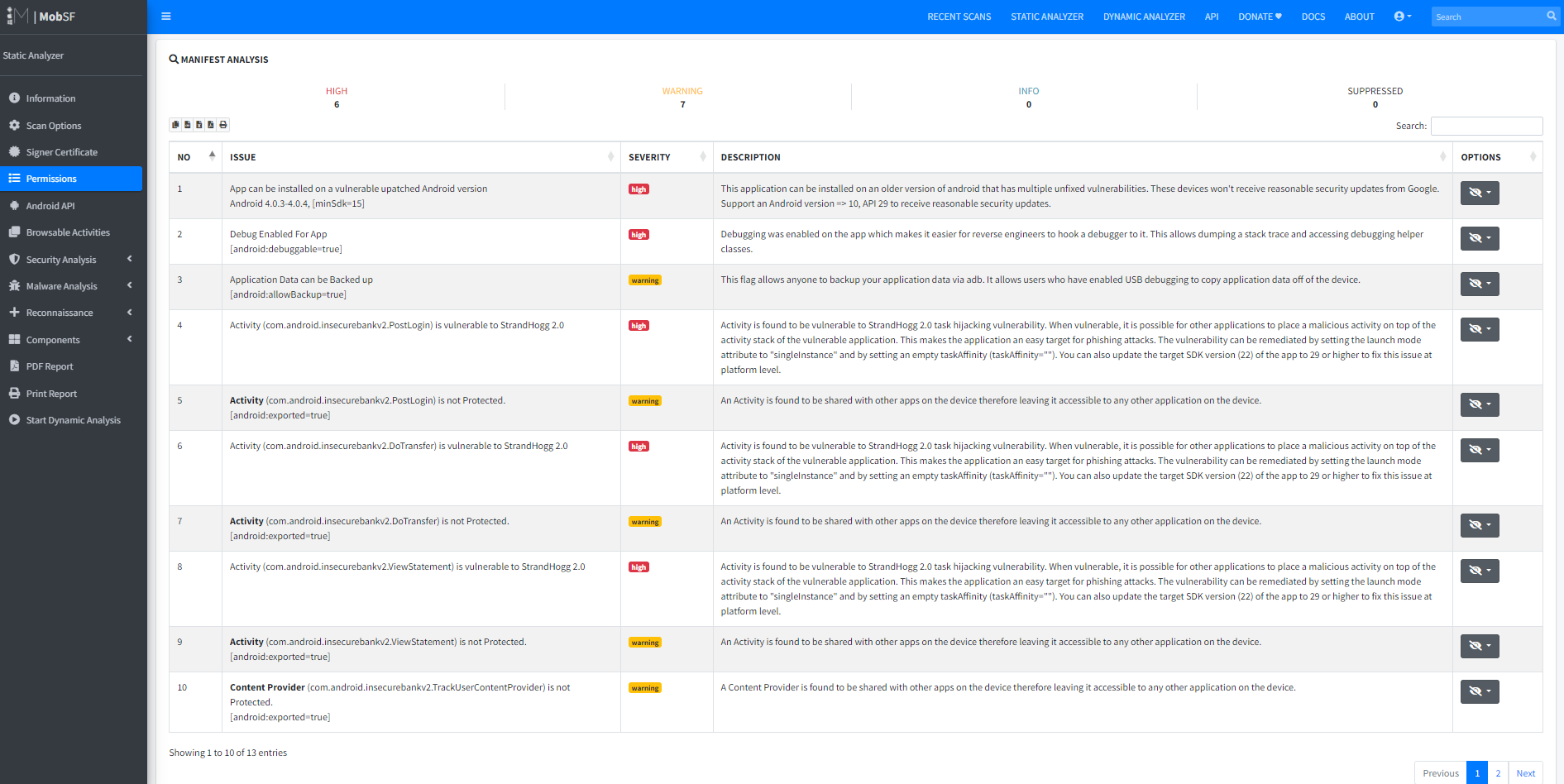


- Kiểm tra các cấu hình phân quyền



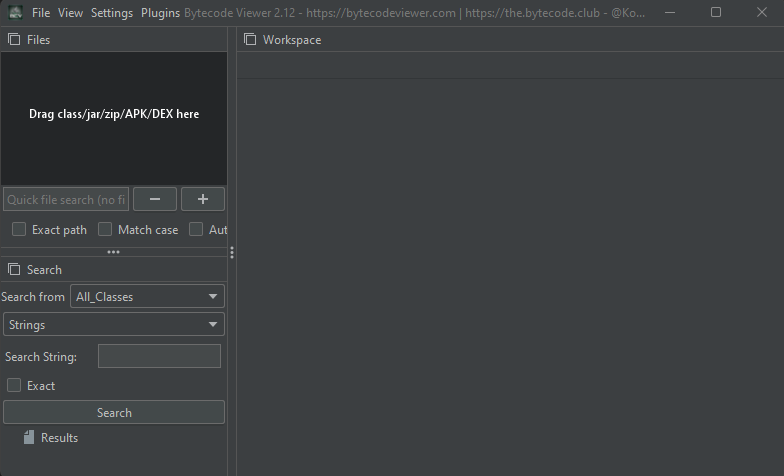
- Đầu tiên, dựa vào chương trình phân tích code, ta thấy phát hiện một số vấn đề như SEND\_SMS, USE\_CREDENTIAL,…

- Kiểm tra cấu hình trong tập tin Manifest

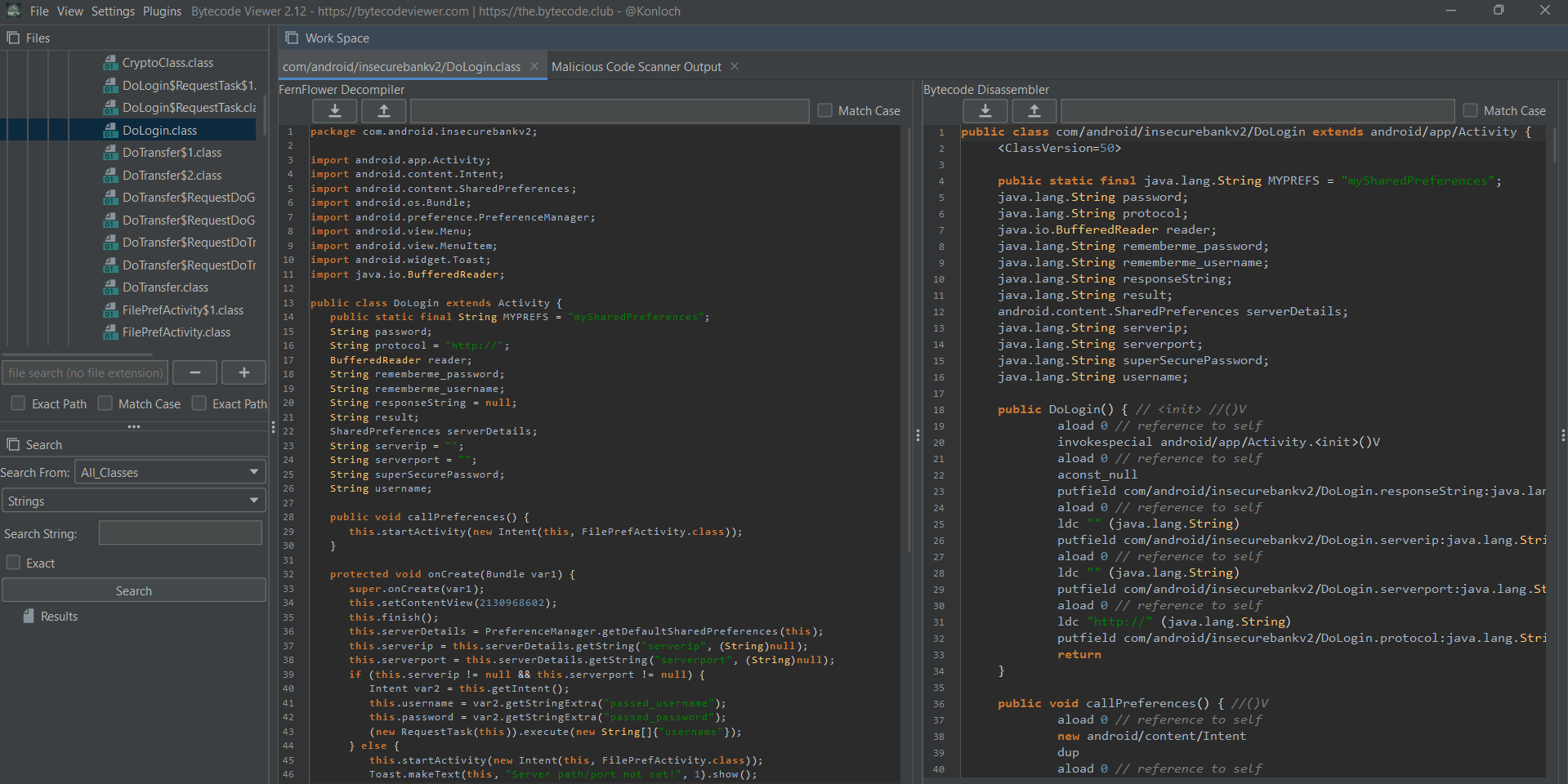


- Một trong các bước quan trong trong phân tích tĩnh là kiểm tra code. Để làm được việc này dùng ByteCode Viewer để xem code nhanh nha trong tập tin apk. Tải tập tin Bytecode-Viewer-2.9.12.jar từ <https://github.com/Konloch/bytecode-viewer/releases> và mở chương trình bằng dòng lệnh sau:

**java -jar Bytecode-Viewer-2.9.12.jar**

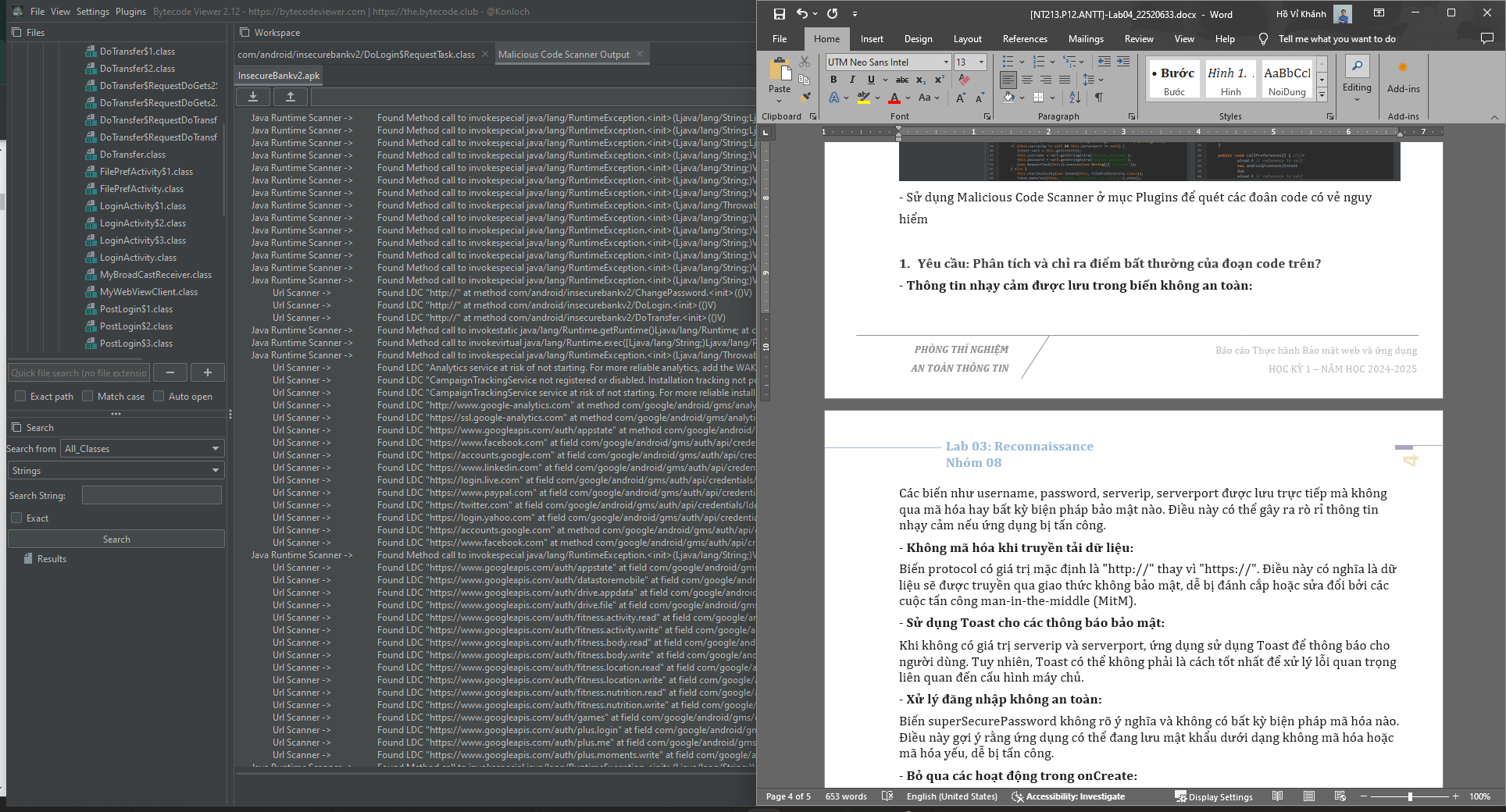
****

- Kéo thả tập tin apk, ByteCode Viewer sẽ phân tích và hiển thị code



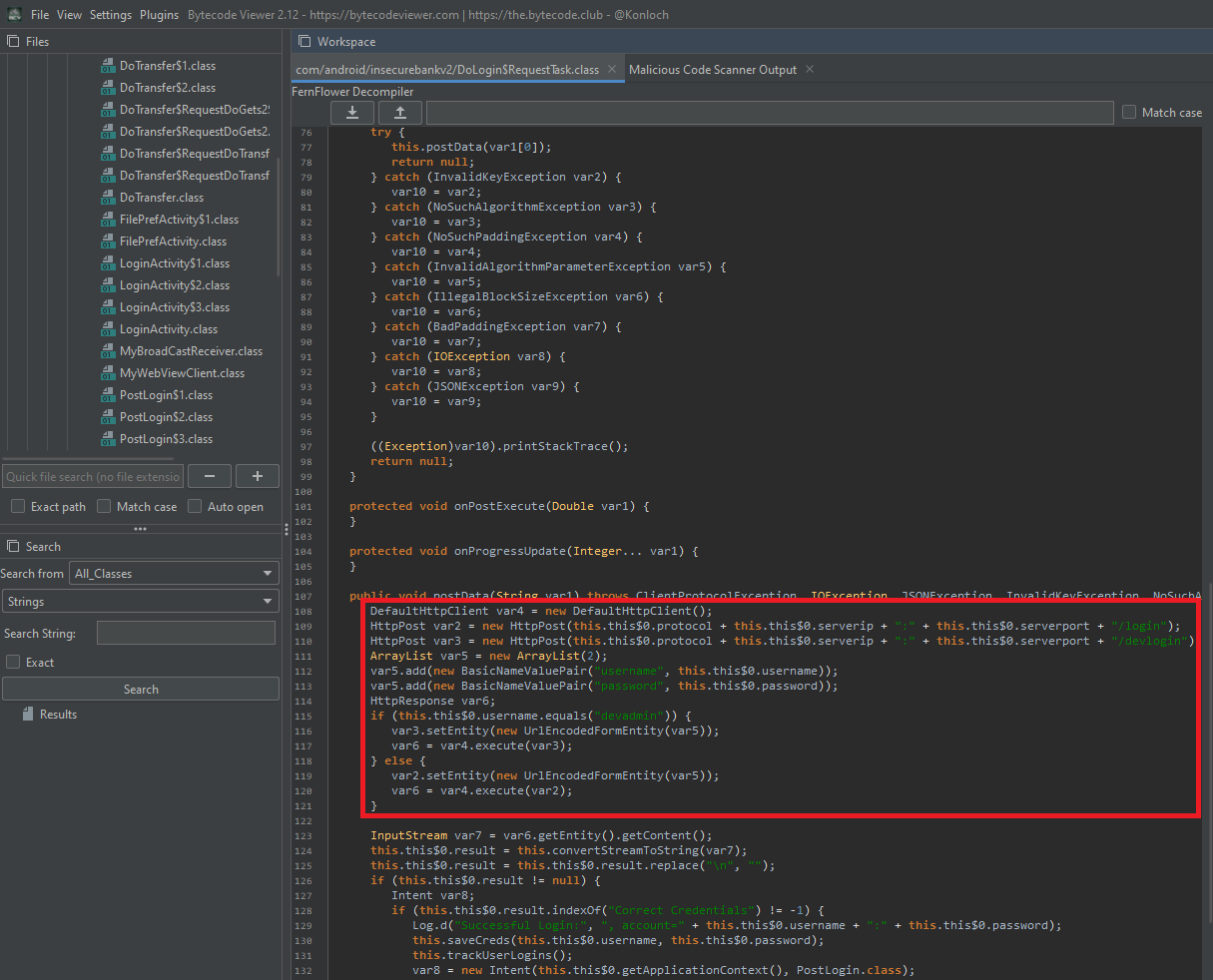
- Sử dụng Malicious Code Scanner ở mục Plugins để quét các đoân code có vẻ nguy

hiểm



**Ví dụ:** Nhìn thấy đoạn code lạ ở đường dẫn:

com/android/insecurebankv2/DoLogin$RequestTask.class



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

***Ta quan sát dòng code từ dòng 117 đến dòng 123:***

- Ta thấy chỉ cần nhập thông tin username là devadmin là đã tự động đăng nhập

được, không quan tâm đến mật khẩu.

**Có một số điểm bất thường trong đoạn mã này:**

- **Thông tin nhạy cảm được lưu trong biến không an toàn:**

Các biến như username, password, serverip, serverport được lưu trực tiếp mà không qua mã hóa hay bất kỳ biện pháp bảo mật nào. Điều này có thể gây ra rò rỉ thông tin nhạy cảm nếu ứng dụng bị tấn công.

- **Không mã hóa khi truyền tải dữ liệu:**

Biến protocol có giá trị mặc định là "http://" thay vì "https://". Điều này có nghĩa là dữ liệu sẽ được truyền qua giao thức không bảo mật, dễ bị đánh cắp hoặc sửa đổi bởi các cuộc tấn công man-in-the-middle (MitM).

- **Sử dụng Toast cho các thông báo bảo mật:**

Khi không có giá trị serverip và serverport, ứng dụng sử dụng Toast để thông báo cho người dùng. Tuy nhiên, Toast có thể không phải là cách tốt nhất để xử lý lỗi quan trọng liên quan đến cấu hình máy chủ.

- **Xử lý đăng nhập không an toàn:**

Biến superSecurePassword không rõ ý nghĩa và không có bất kỳ biện pháp mã hóa nào. Điều này gợi ý rằng ứng dụng có thể đang lưu mật khẩu dưới dạng không mã hóa hoặc mã hóa yếu, dễ bị tấn công.

- **Bỏ qua các hoạt động trong onCreate:**

Gọi finish() trong phương thức onCreate() sẽ khiến Activity kết thúc ngay sau khi được tạo, điều này làm cho DoLogin hoạt động không bình thường. Có thể dẫn đến lỗi không mong muốn khi hoạt động này không được hoàn tất đúng cách.

- **Tham số cứng trong lệnh gọi RequestTask:**

Việc sử dụng new String[]{"username"} mà không kiểm tra xem username và password có giá trị hợp lệ không có thể dẫn đến các lỗi khi gửi yêu cầu không đầy đủ.

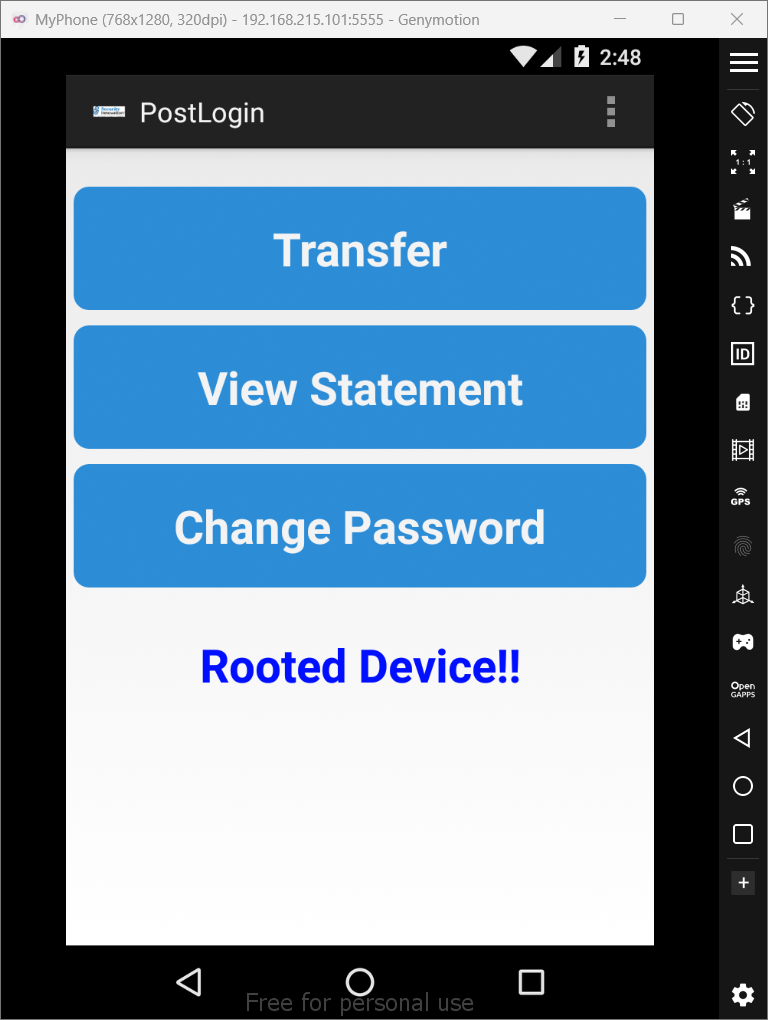
- **Lỗi mã hóa mã nguồn:**

Mã nguồn bao gồm các giá trị như 2130968602 hoặc 2131558557 không rõ ràng. Sẽ tốt hơn nếu sử dụng R.layout.layout\_name hoặc R.id.menu\_item\_name để dễ bảo trì mã nguồn.

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

- Đầu tiên, thử đăng nhập vào để chương trình lưu thông tin

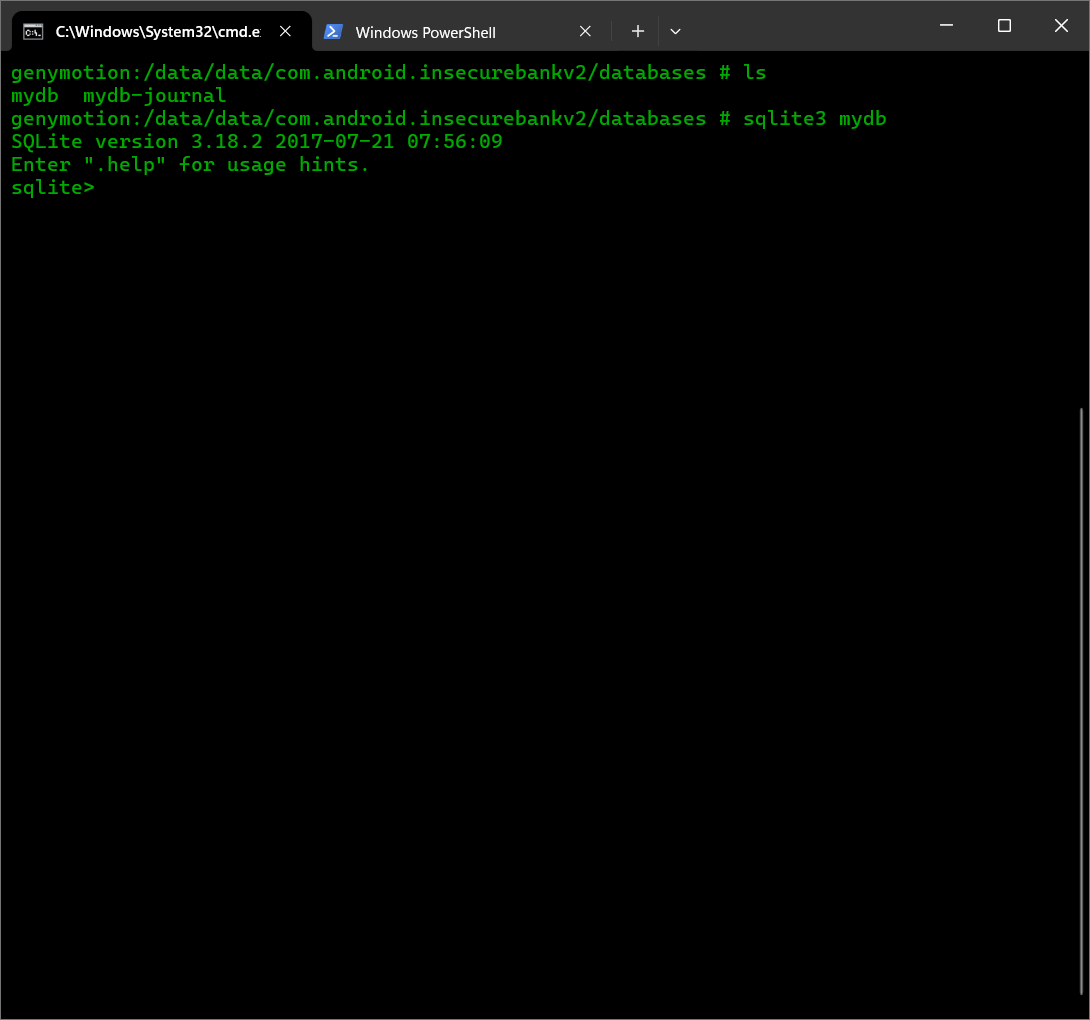
**Account:** dinesh:Dinesh@123$ hoặc jack:Jack@123$



- Từ máy chính, gõ **adb shell** để vào command line của máy ảo Android

- Tiếp đến vào command shell của điện thoại ảo di chuyển vào

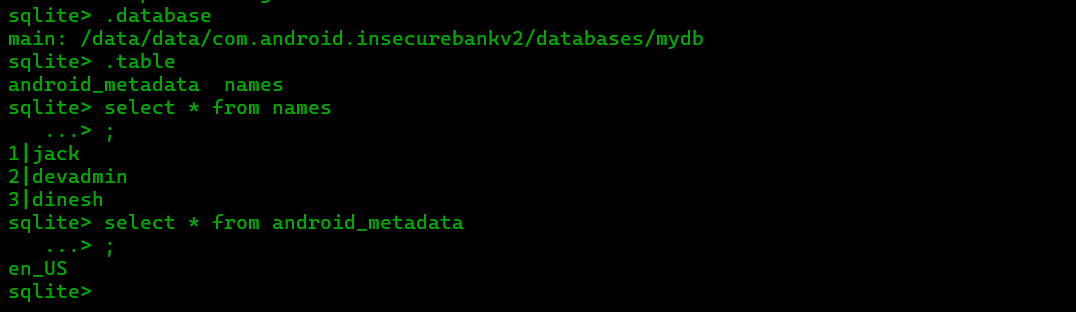
**/data/data/com.android.insecurebankv2/databases**



- Sử dụng lệnh sqlite3 mydb để tiến hành kiểm tra mydb. Trong database liệt kê tất cả

các table hiện có, ở đây ta thấy có 2 table là android\_metadata và names, ta sẽ cho hiển

thị toàn bộ các thông tin có trong 2 table đó



- Kết quả chỉ hiển thị các username ở dưới dạng plain text không có mã hóa

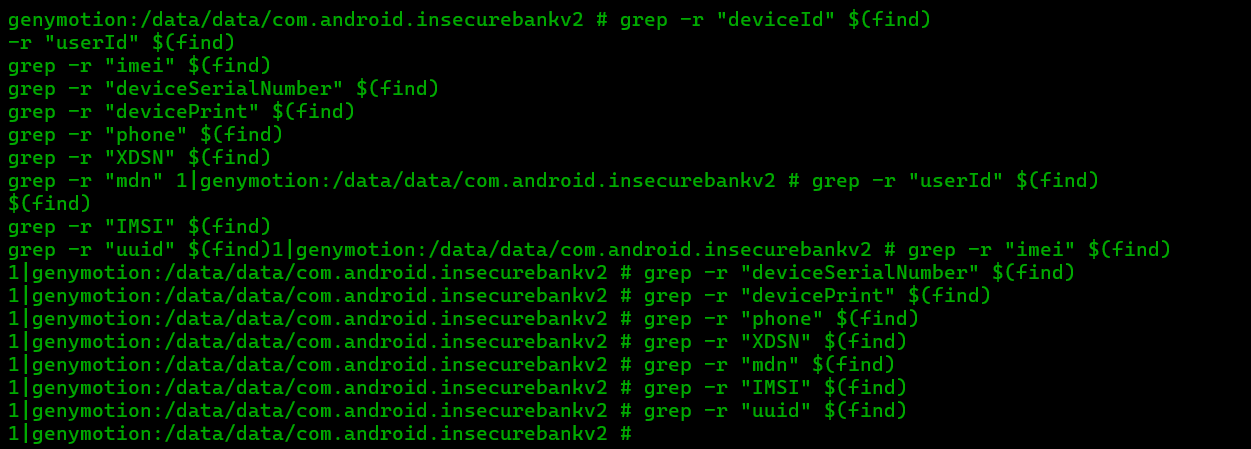
=> **Không an toàn**

## 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 …

- Vào adb shell đi đến /data/data/com.android.insecurebankv2

- Chạy lệnh:

**grep -r <string-to-find> $(find)**



- Trong phần này thì ta không tìm được thông tin nhạy cảm nào cả.

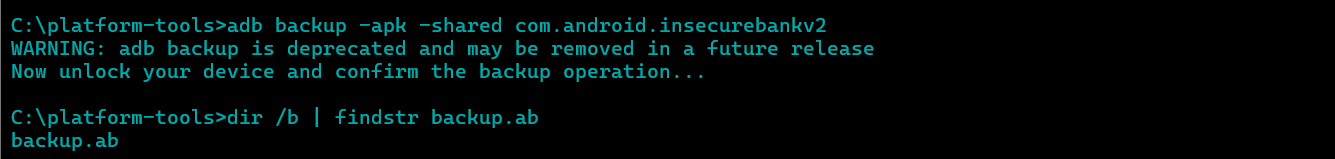
## Yêu cầu 4: Theo bạn thư mục sao lưu chứa thông tin nào cần mã hoá, chỉ ra.

- Đăng nhập với tài khoản người dùng bình thường (dinesh/Dinesh@123$ hoặc

jack/Jack@123$)

- Thực thi lệnh sau để sao lưu:

**adb backup -apk -shared com.android.insecurebankv2**



- Chuyển đổi tập tin sao lưu qua định dạng có thể đọc được (cài gói qpdf)

A computer screen shot of a computer

Description automatically generated

- Sau khi hoàn thành ta sẽ có file backup trên, thực hiện kiểm tra thì ta thấy có được một số thông tin được mã hoá

A computer screen with white text

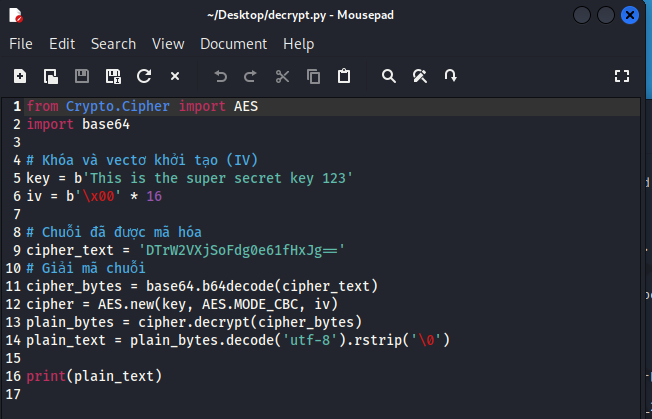
Description automatically generated

- Ta thấy thông tin của user đã mã hóa

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

- Đề bài cũng đã cung cấp cho ta một đoạn mã java dùng thuật toán AES-256 với modeC CBC và đệm PKCS5Padding để mã hóa kèm theo đó là key và iv dùng để giải mã

- Từ đó ta có thể viết một chương trình Python dùng để giải mã. Chạy chương trình trên one compiler, kết quả trả về là mật khẩu của user dinesh

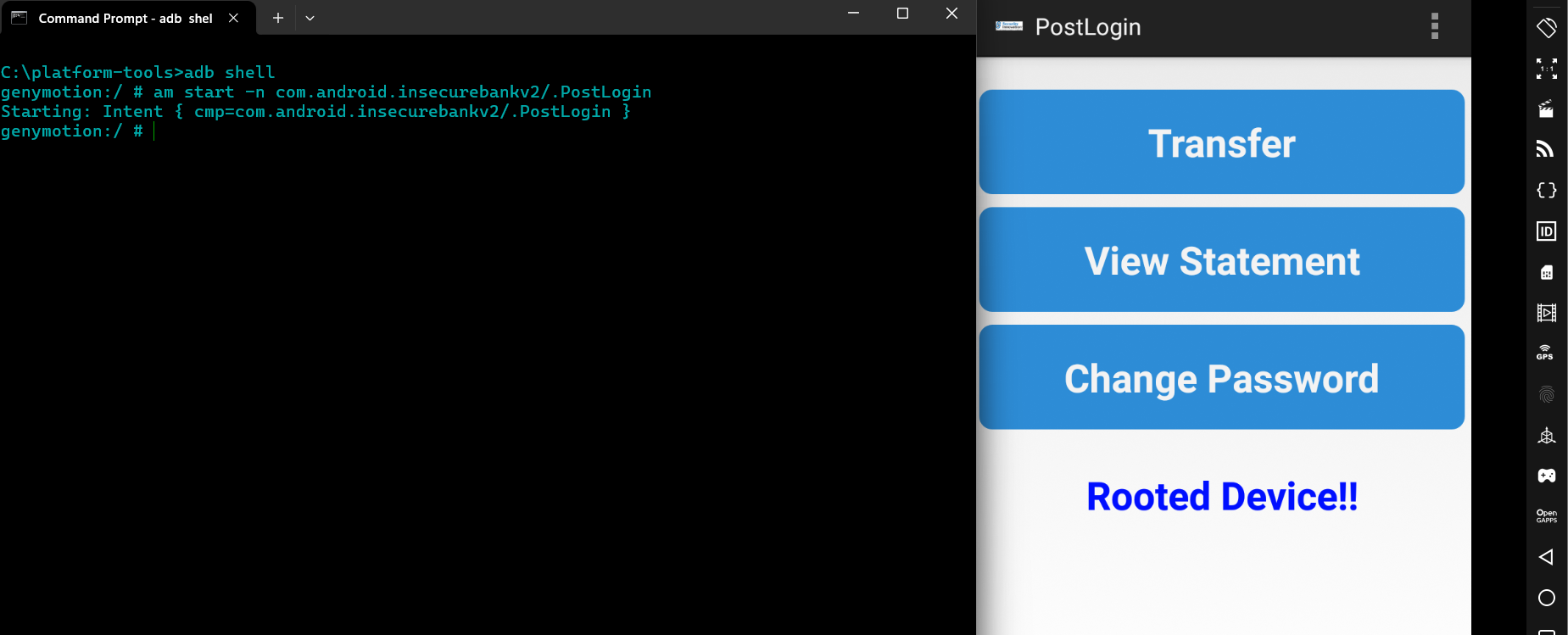


Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

- Ta thấy được thông tin bị mã hoá là mật khẩu của tài khoản Dinesh

**Activity Hijacking**

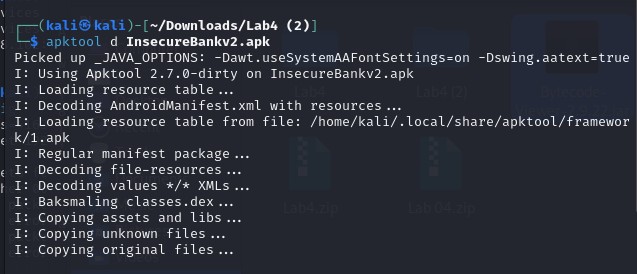
****

- Sau khi hijacking thì nó tự động nhảy sang tab PostLogin mà ta không cần đăng nhập

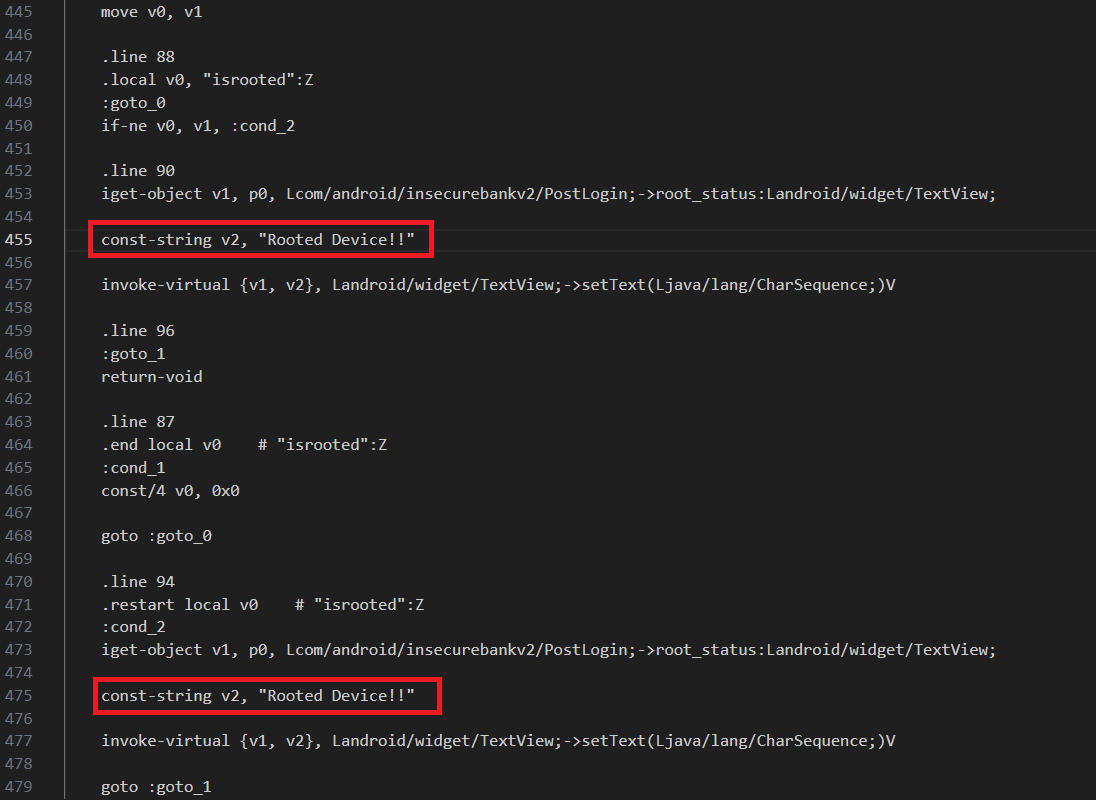
# Dịch ngược (Reverse Engineering)

## 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.

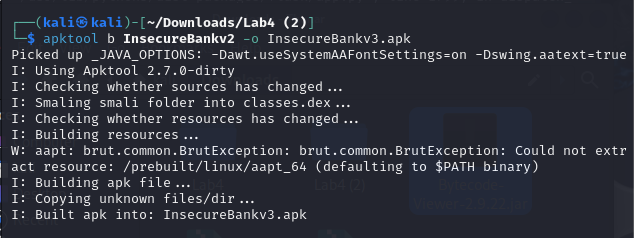
- Đầu tiên ta sẽ sử dụng apktool để dịch ngược chương trình



- Ta sẽ truy cập vào InsecureBankv2/smali/com/android/insecurebankv2/ PostLogin.smali để sửa đổi tập tin. Thay đổi nhánh else để in ra Rooted Device

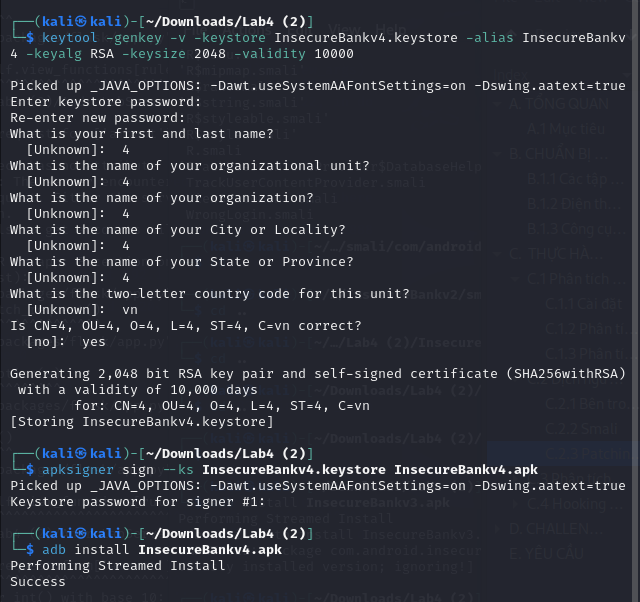


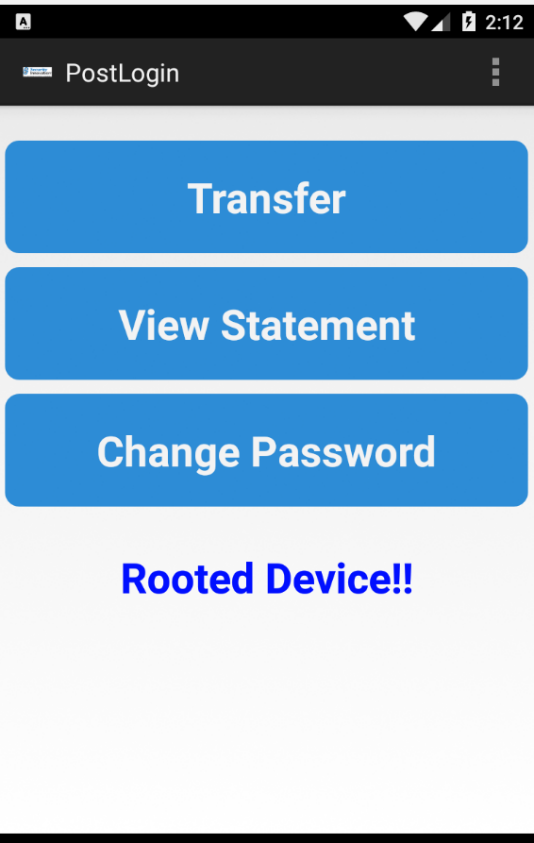
- Sử dụng apktool để vá lại file apk



- 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.

- Vì vậy ta cần tạo key và kí

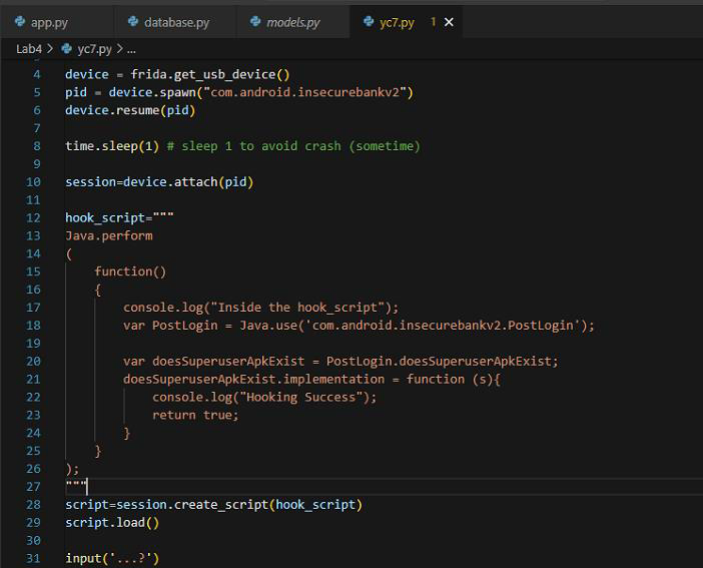




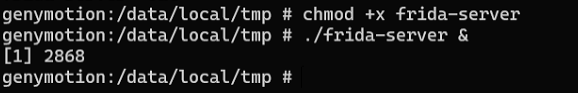
# Hooking với Frida

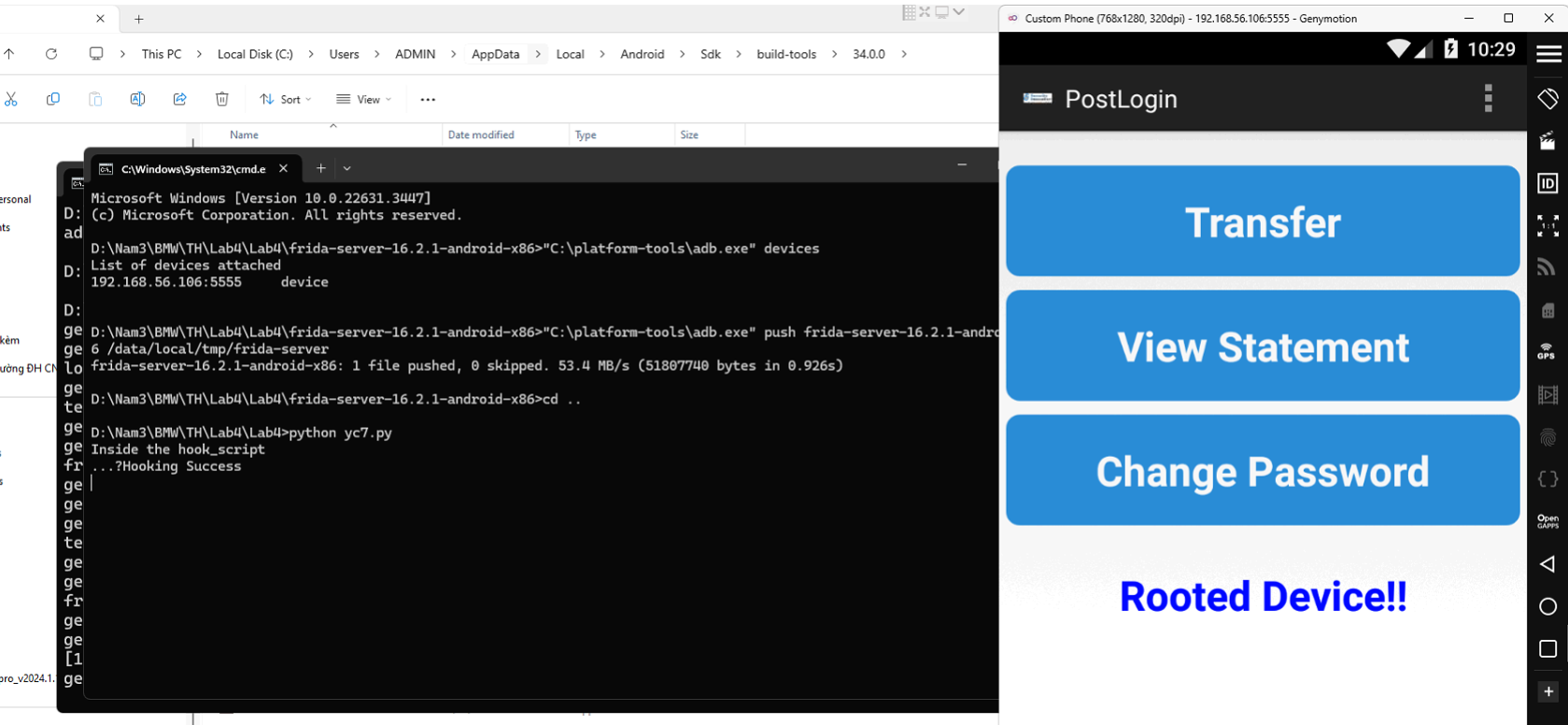
## Yêu cầu 7: Hoàn thiện đoạn code trên và demo.

- Đoạn code dưới dùng để hook method doesSuperuserApkExist trong class PostLogin của ứng dụng Android "InsecureBankv2". Khi hàm này được gọi, nó sẽ in ra "Hooking Success" trên console và trả về giá trị true.



- Thực hiện chạy frida-server trên điện thoại trước, sau đó tiến hành chạy file python để hook. Lúc này ứng dụng sẽ bắt ta đăng nhập, sau khi đăng nhập thành công sẽ hiển thị “Hooking Success”.





-- HẾT --

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