BÁO CÁO THỰC HÀNH

**Môn học: Pháp chứng kỹ thuật số**

**Lab 5: Mobile Forensic**

*GVHD: Đoàn Minh Trung*

1. **THÔNG TIN CHUNG:**

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

Lớp: NT334.O21.ATTN

|  |  |  |  |
| --- | --- | --- | --- |
| **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 0 | 100% |
| 2 | Kịch bản 1 | 100% |
| 3 | Kịch bản 2 | 100% |
| 4 | Kịch bản 3 | 100% |
| 5 | Kịch bản 4 |  |

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

BÁO CÁO CHI TIẾT

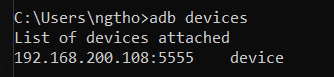
1. **Kịch bản 0. Thực hiện lại các yêu cầu ở phần hướng dẫn.**

**Yêu cầu sinh viên chọn một ứng dụng và phân tích trên thiết bị di động Android của mình.**

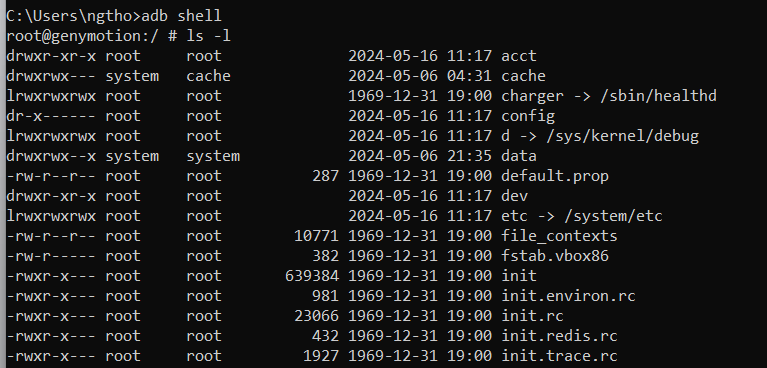
* Trên Genymotion, em có thiết lập sẵn một thiết bị như sau:



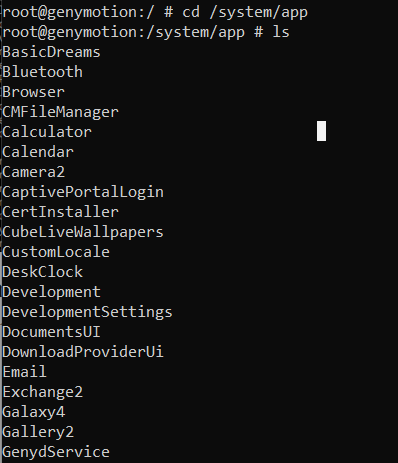
* Sử dụng adb để truy cập thiết bị. Kiểm tra thiết bị đã kết nối bằng cách mở cmd và gõ lệnh:



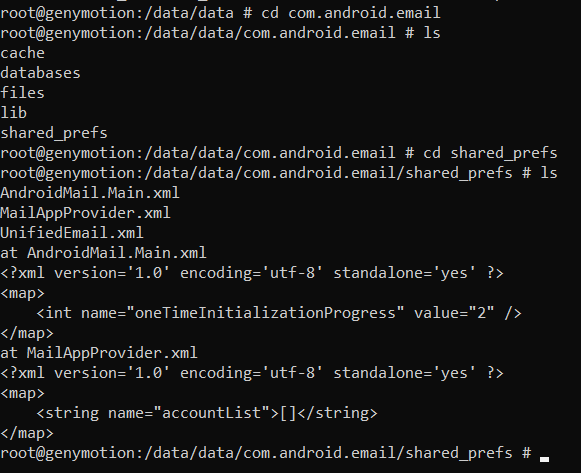
* Vào shell bằng lệnh *adb shell*, liệt kê các file/thư mục bằng *ls –l:*



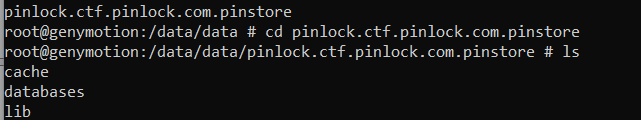
* Trích xuất dữ liệu trên thiết bị Android:
* Xem danh sách các ứng dụng hệ thống/ các ứng dụng cài sẵn:



* Xem dữ liệu các ứng dụng, nằm trọng thư mục /data/data. Để xem cần có quyền root trên thiết bị:
* Đối với từng ứng dụng được liệt kê bên trên, để xếm các giá trị tham chiếu quan trọng thì tìm trọng thư mục của nó và được lưu dưới dạng tập tin .xml: */data/data/<package\_name>/shared\_prefs.* Ví dụ xem thông tin ứng dụng email:



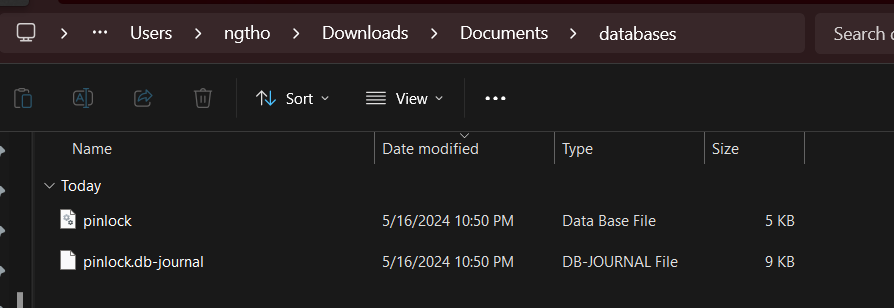
* Xem dữ liệu cục bộ của ứng dụng. Ví dụ xem thông tin ứng dụng em đã cài là *pinstore.apk*:



* Tải dữ liệu từ thiết bị di động về máy tính điều tra:



Kết quả:

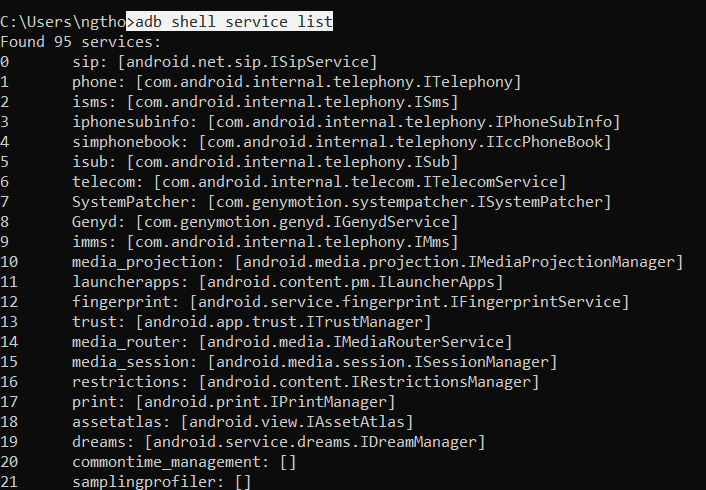


* Xác định ý nghĩa của câu lệnh sau:

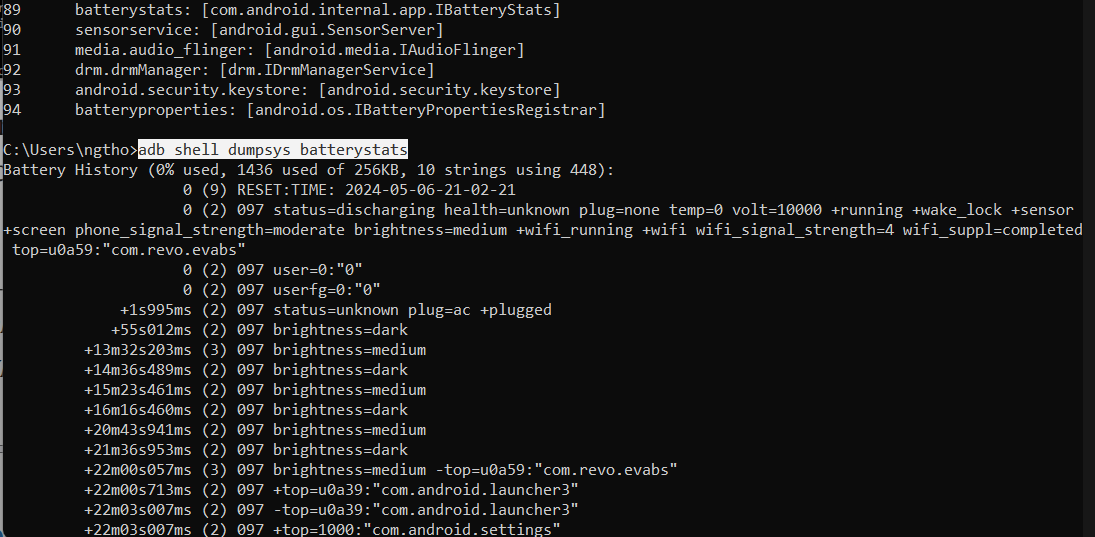


Câu lệnh này sao chép file được lưu trữ ở bất kỳ đâu trên điện thoại và lưu file đó vào một thư mục cụ thể trên máy tính thông qua công cụ Android Debug Bridge.

* Tìm hiểu ADB Dumpsys (không đòi hỏi phải có quyền root), để xem các dịch vụ đang chạy trên máy, thực hiện:

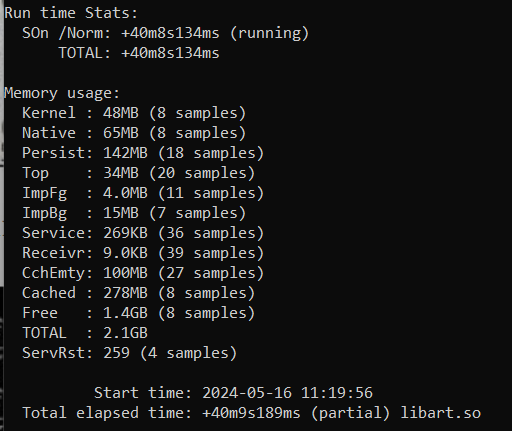


* Dùng dumpsys để xem thông tin các dịch vụ liên quan hệ thống:

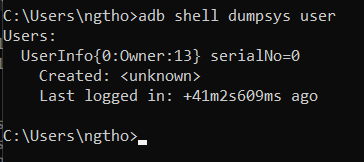


* Thử các lệnh dumpsys khác và ghi nhận kết quả:

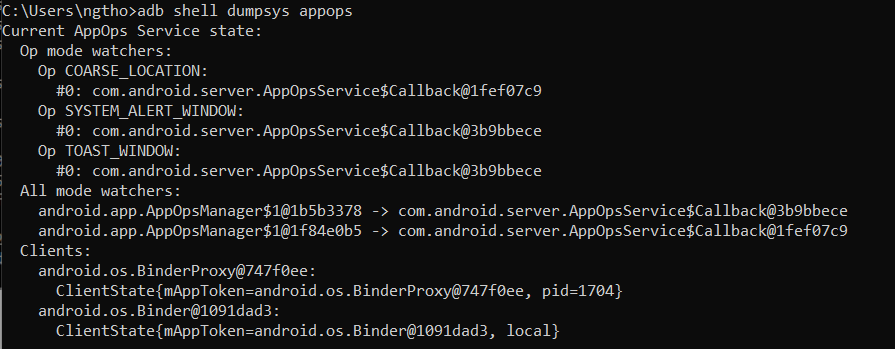
**Dumpsys procstats**: tình trạng sử dụng bộ xử lý của các ứng dụng đang chạy:

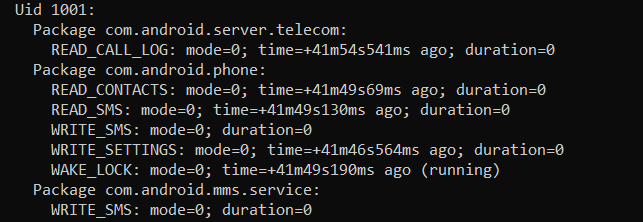


**Dumpsys user**: hiển thị thông tin người dùng đang sử dụng thiết bị:



**Dumpsys App Ops**: thông tin về quyền hạn có thể truy cập bởi các ứng dụng:



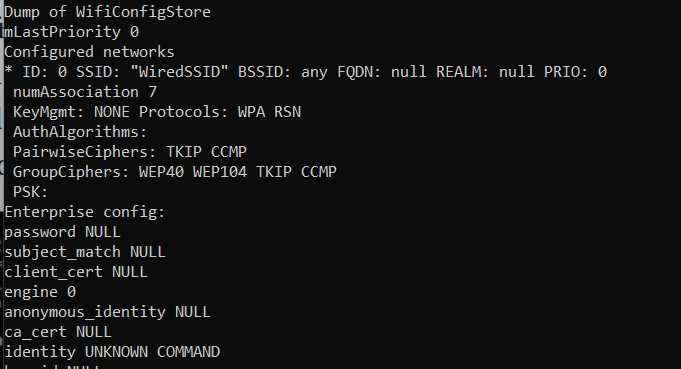


Cách thời điểm điều tra 41 phút 54 giây 541 mili giây, người dùng sử dụng ứng dụng Điện thọại (Phone) trên thiết bị của mình để đọc nhật ký cuộc gọi với quyền READ\_CALL\_LOG.

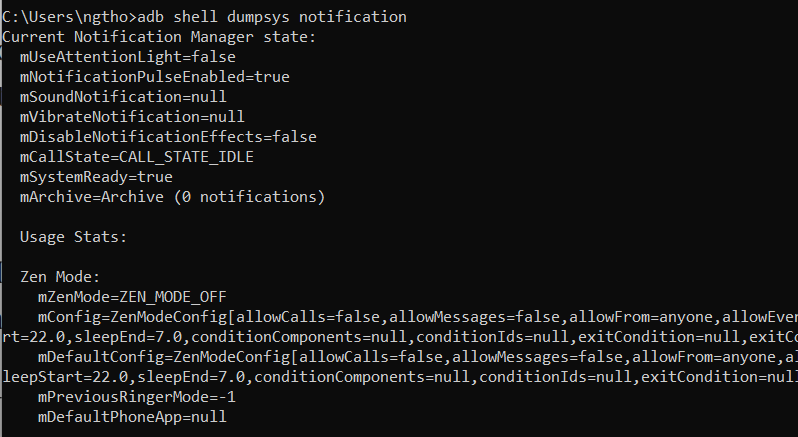
Sau đó, đọc danh bạ điện thoại READ\_CONTACTS. Dùng quyền READ\_SMS để đọc tin nhắn SMS, sau đó viết tin nhắn SMS.

Cuối cùng và giữ màn hình sáng bằng WAKE\_LOCK.

**Dumpsys Wi-Fi:** hiển thị danh sách các SSID mà thiết bị đã kết nối tới. Điều này hữu ích trọng các trường hợp muốn xếm người dùng này có thể đã tới những địa điểm nào:



**Dumpsys Notification**: hiển thị các thông tin về các thông báọ đang được kích hoạt trên thiết bị:



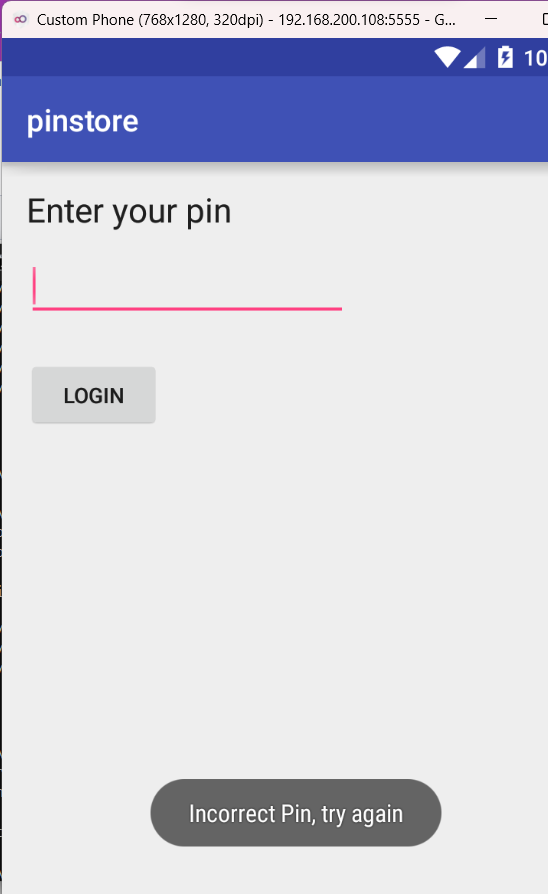
Nếu không chỉ định dịch vụ cụ thể, dumpsys sẽ trích xuất tất cả các dịch vụ đang chạy, ghi vào tập tin đầu ra.

1. **Kịch bản 01. Thực hiện phân tích ứng dụng Android**

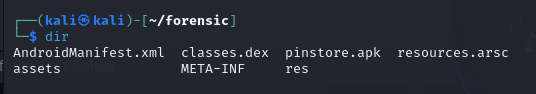
* Mô tả: Phân tích ứng dụng Android, tìm mã PIN trong ứng dụng để tìm flag.
* Tài nguyên thực hiện: pinstore.zip
* Yêu cầu – Gợi ý: Sử dụng các công cụ dịch ngược (de) trên mã nguồn Andrọid để phân tích.

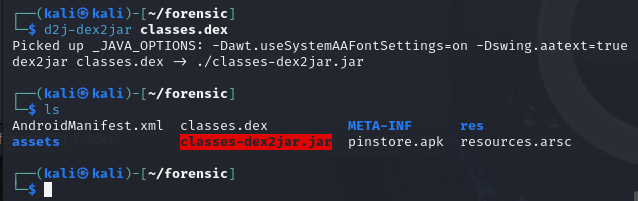
*Đáp án:*

* Cài đặt và mở ứng dụng. App yêu cầu nhập mã PIN, do đó ta cần tìm mã PIN hợp lệ:

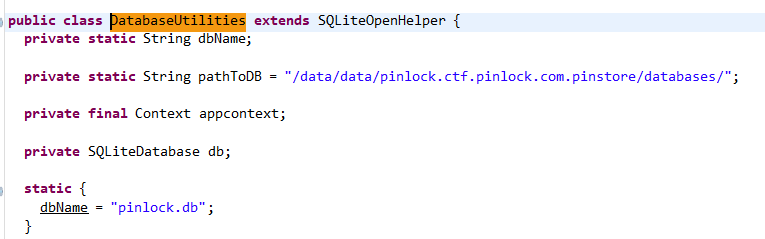


* Giải nén tập tin .zip, và giải nén tập tin .apk. Sau khi thu được tập tin classes.dex thì chuyển đổi thành tập tin JAR:

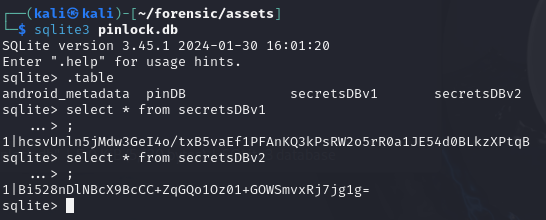




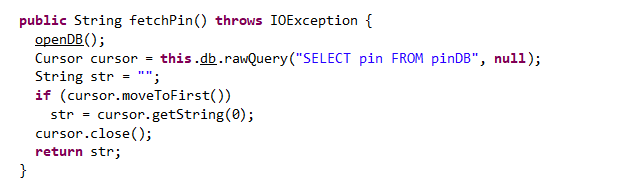
* Dùng JD-GUI để mở tập tin *.jar*, quan sát:



Một CSDL được tạo ra có tên là *pinlock.db*. Lớp này quản lý CSDL SQLite, để xem được database, thực hiện:



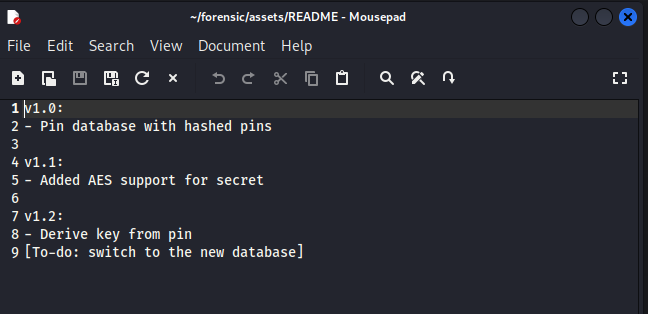
Mã PIN được lưu trong *pinDB:*



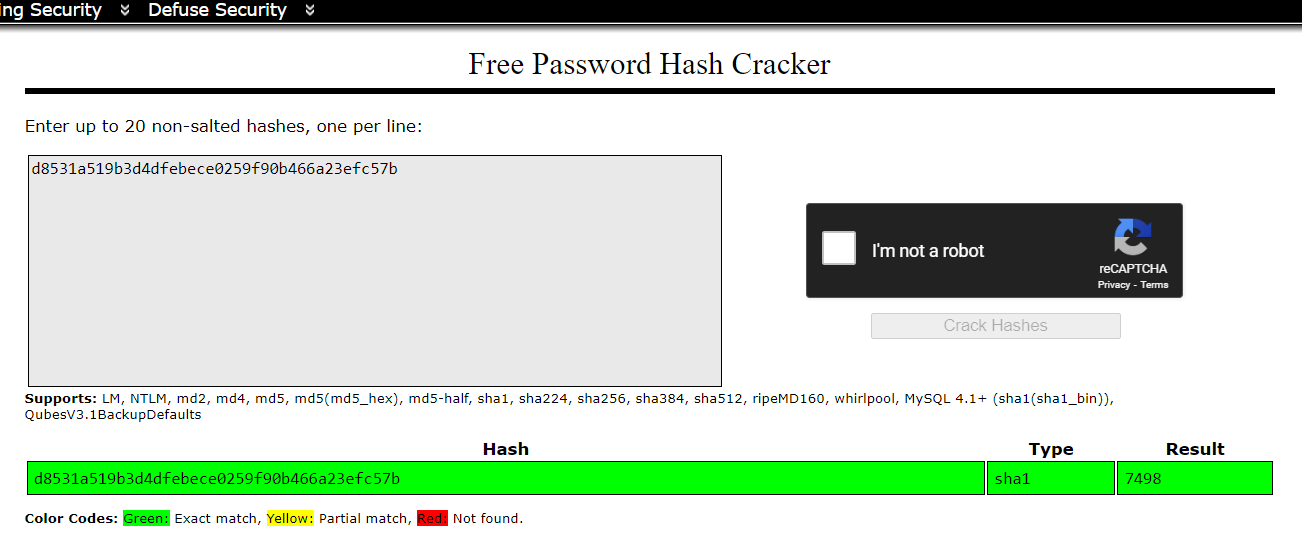
nên em sẽ xem thông tin được lưu trong bảng này:

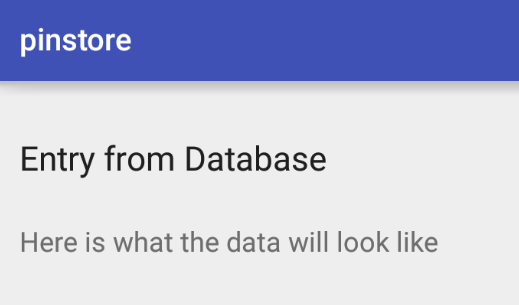


* Chú ý đến tập tin README trong thư mục assets:

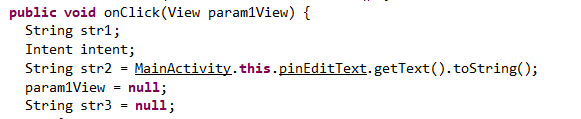


Ở đây chỉ ra rằng PIN đã bị hash, em dùng CrackStation để decrypt. Tìm được PIN là 7498, nhập vào app, hiện thông báo thôi chứ chẳng có flag:

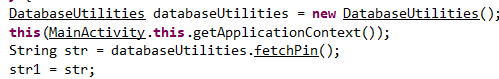




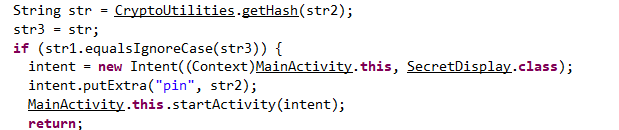
* Quay lại phân tích flow chương trình. Quan sát MainActivity.class, lấy mã PIN từ EditText nơi người dùng nhập vào:



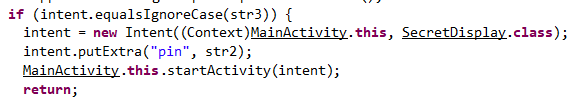
Khởi tạo DatabaseUtilities để truy cập cơ sở dữ liệu. Gọi phương thức fetchPin() để lấy mã PIN lưu trữ. Lưu PIN và str1:



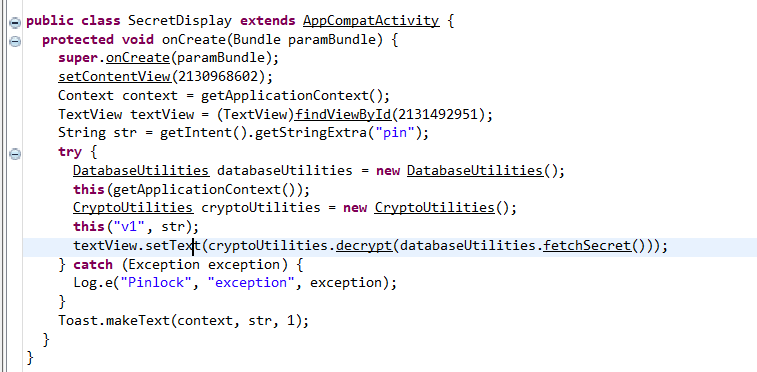
Băm mã PIN người dùng nhập vào bằng phương thức CryptoUtilities.getHash(). So sánh mã PIN băm (str3) với mã PIN lưu trữ (str1). Nếu khớp, tạo Intent để chuyển sang SecretDisplay Activity và truyền mã PIN qua Intent:



Nếu mã PIN đúng, sẽ chuyển sang SecretDisplay. Nhưng đúng cũng không hiện gì hết:

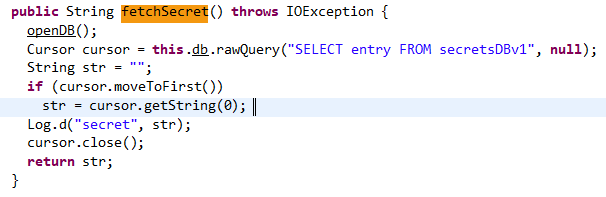


* Xem SecretDisplay:

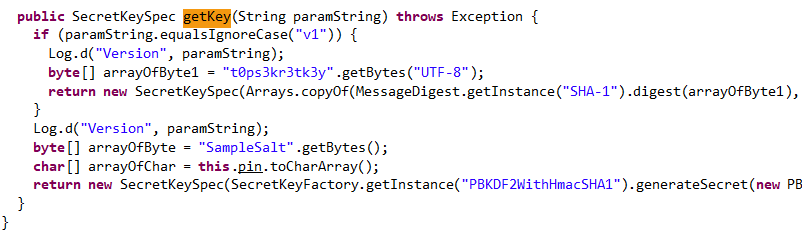


Ta thấy sau khi ta nhập mã pin vào thì nó sẽ lấy giá trị đó và dùng CryptoUtilities để thực hiện mã hóa và giải mã bằng cách truyền phiên bản “v1” và “pin” vào.

Sau đó nó sử dụng hàm fetchSecret() để lấy chuỗi bí mật từ database, chuỗi này được decrypt và kết quả giải mã được sẽ hiển thị. Nội dung fetchSecret():



* Xem hàm getkey trong CyptoUtilities class:

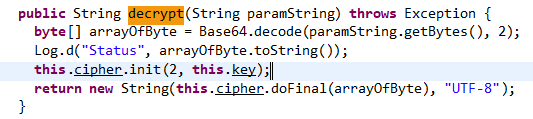


Kết hợp với gợi ý ở README, ta có thể thấy:

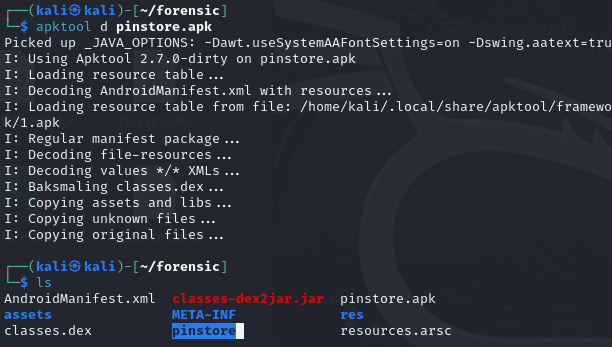
v1: Sử dụng chuỗi cố định "t0ps3kr3tk3y" và băm SHA-1 để tạo khóa AES.

Ở version khác v1: Sử dụng thuật toán PBKDF2 với HMAC-SHA1, mã PIN và salt "SampleSalt" để tạo khóa AES.

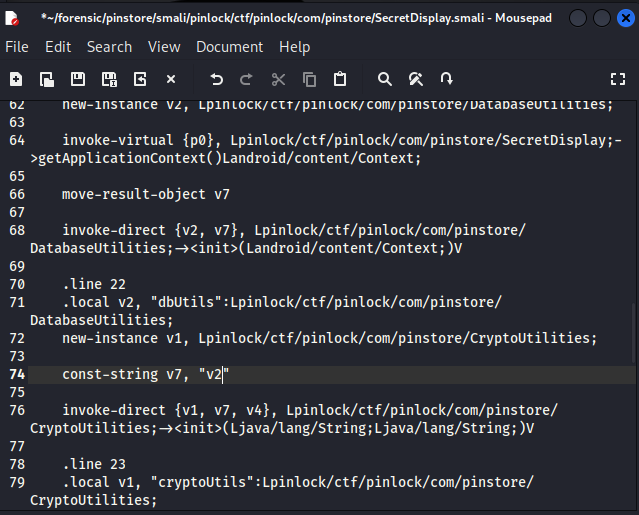
Kết quả là SecretKeySpec tạo ra dựa trên phiên bản và PIN được sử dụng cho việc mã hóa và giải mã dữ liệu.



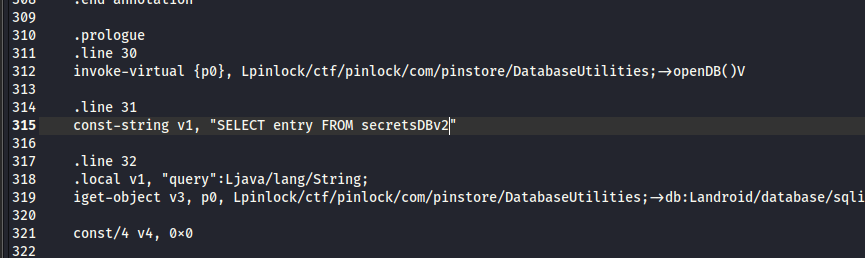
* Dùng apktool để decompile file apk, em muốn xem các file .smali và chỉnh sửa để tạo ra phiên bản apk khác:



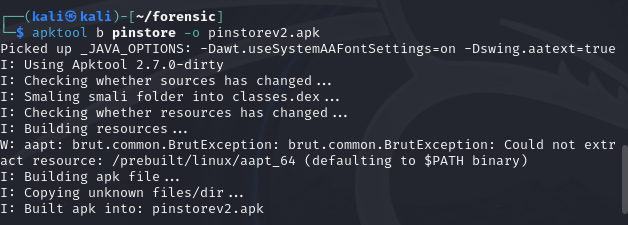
* Chỉnh dòng 74 trong file SecretDisplay.smali từ “v1” thành “v2”:

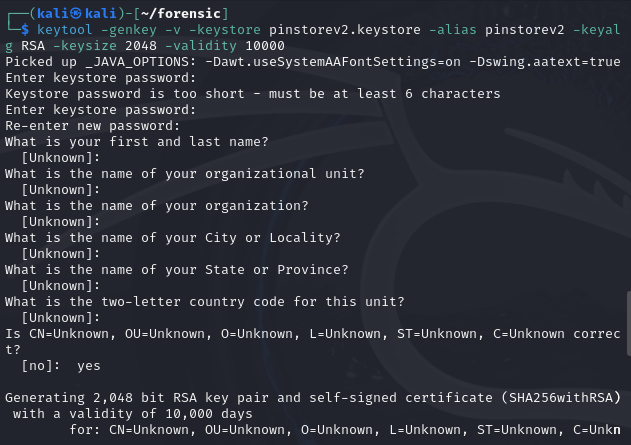


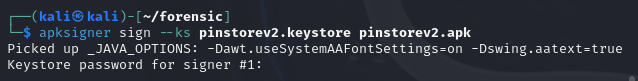
* Chỉnh dòng 315 trong file DatabaseUtilities.smali, chọn từ bảng secretsDBv2 thay vì secretsDBv1:



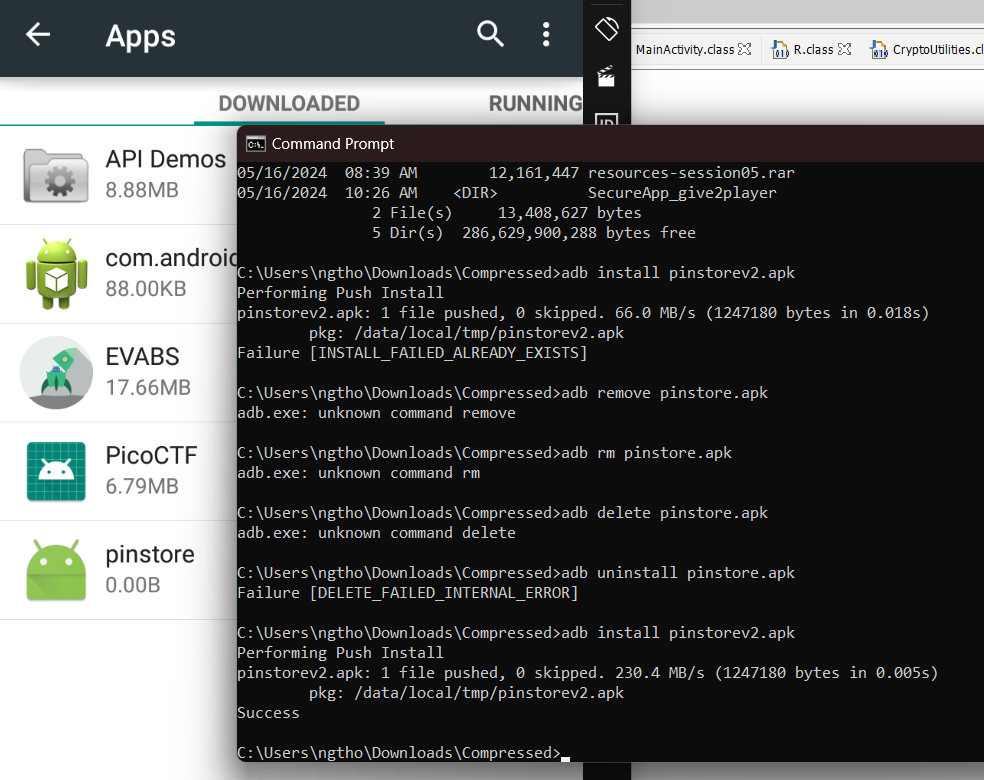
* Thực hiện ký và tạo file apk mới để cập nhật các chỉnh sửa trong smali:



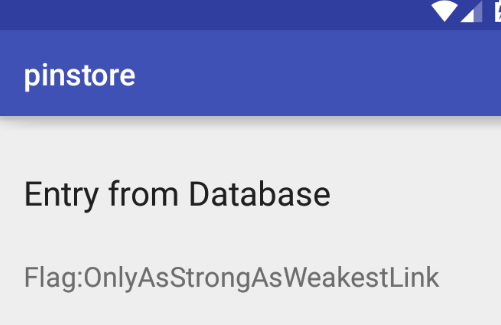




* Gỡ app cũ để cài lại app vừa được chỉnh sửa: sử dụng công cụ apktool để build lại APK từ mã nguồn của ứng dụng Android đã được giải nén trước đó, tạo chữ ký và ký file:



* Thực hiện mở app, nhập lại mã PIN:



1. **Kịch bản 02. Thực hiện phân tích tập tin ứng dụng thu được.**

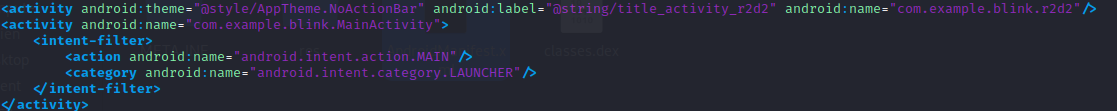
* Mô tả: Ứng dụng kb02 cần được phân tích thành mã smali để tìm flag.
* Tài nguyên thực hiện: kb02\_zha.apk.
* Yêu cầu – Gợi ý: sử dụng công cụ APKTool/ JADX/ dex2jar/ jdgui/Android Studio, flag có dạng CTF{….}

- Decomlie APK





- Xem activity

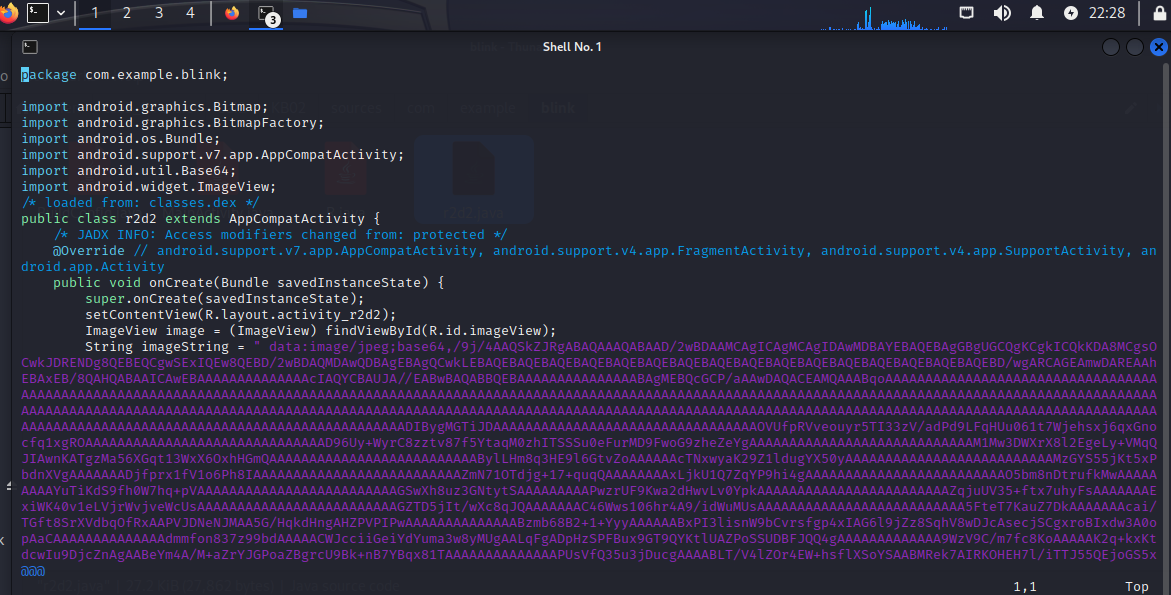
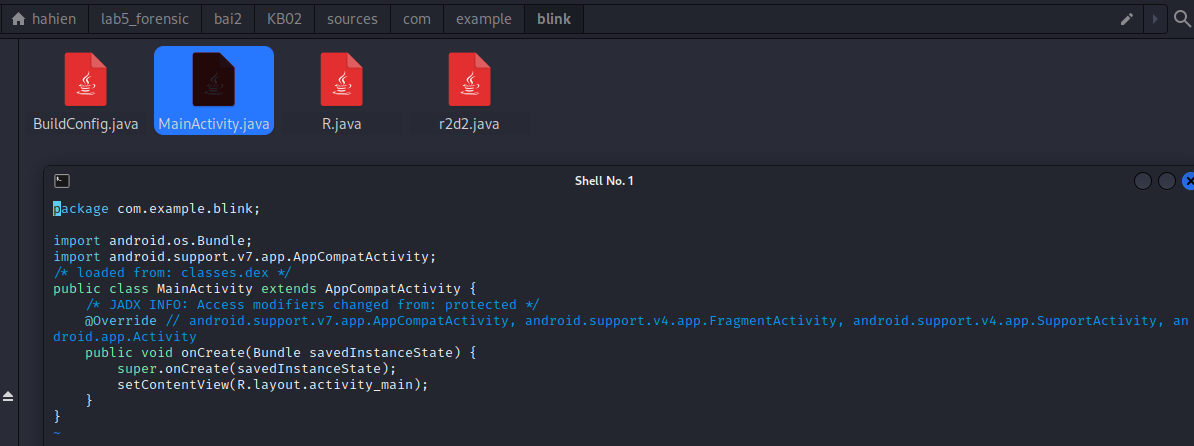
- Activity chính của ứng dụng là com.example.blink.MainActivity, đây là Activity sẽ được khởi chạy khi người dùng mở ứng dụng từ launcher.

- Có một Activity khác được định nghĩa là com.example.blink.r2d2. Điều này có thể là một gợi ý về nơi để tìm kiếm thông tin quan trọng hoặc flag.

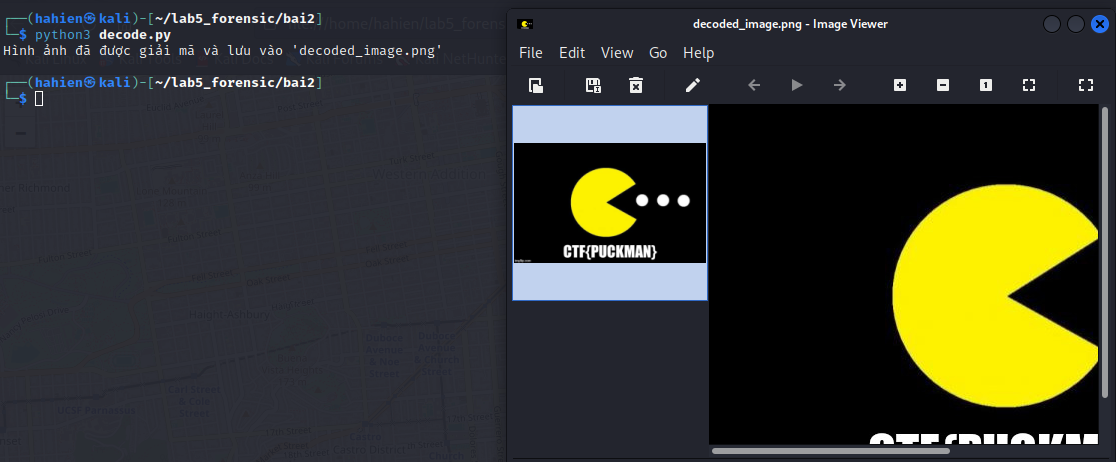


- Ứng dụng được đánh dấu là debuggable, nghĩa là nó có thể chạy trong chế độ gỡ lỗi ==> có thể làm lộ thông tin nhạy cảm hoặc các điểm yếu bảo mật.

- Kiểm tra nội dung của MainActivity và r2d2:



- Tại đây ta thấy một đoạn mã base64 đáng nghi ngờ, decode xem sao



FLAG: CTF{PUCKMAN}

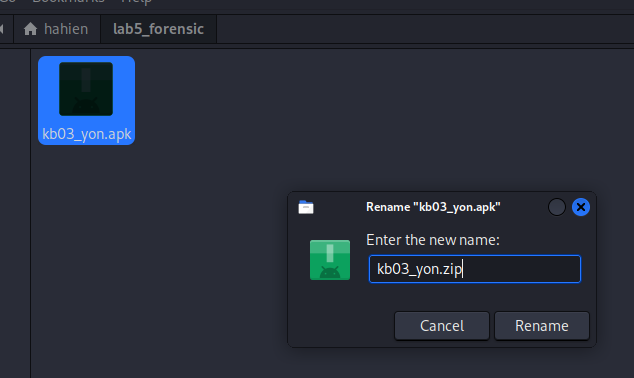
1. **Kịch bản 03: Kịch bản 03. Thực hiện phân tích tập tin ứng dụng thu được.**

* Mô tả: Một ứng dụng có tính năng ghi nhớ các địa điểm mà người dùng

muốn hay không muốn tham quan chỉ bằng dấu tick đơn giản trên bản

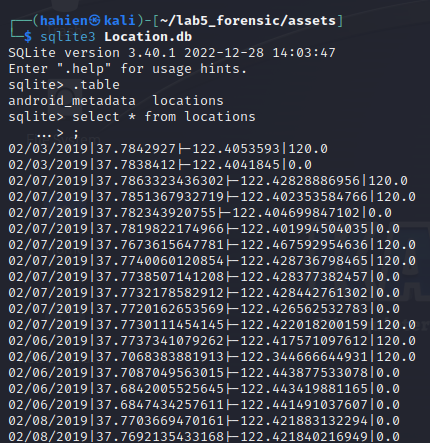
đồ. Tìm flag.

* Tài nguyên: kb03\_yon.apk
* Yêu cầu – Gợi ý: Decompile, chú ý CSDL của ứng dụng.

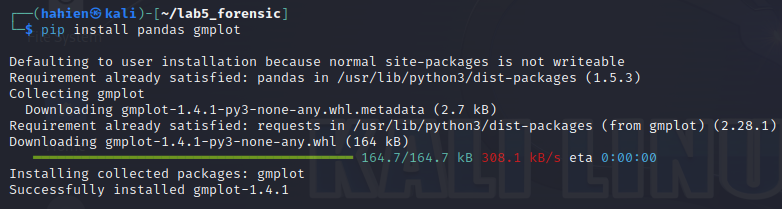




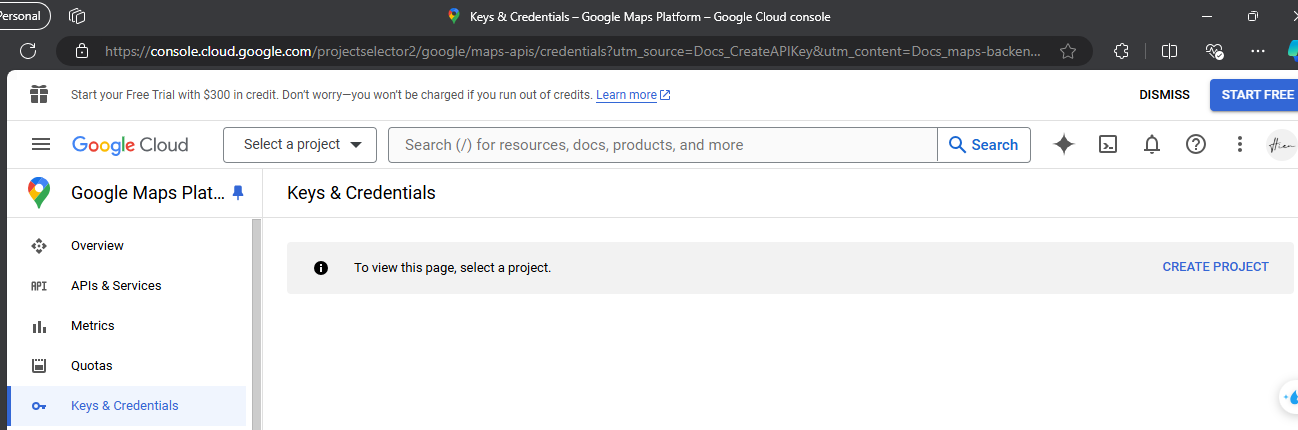
* Gợi ý là chú ý đến CSDL, vì vậy ta cần file assets và mở file Locations.db lên và xem thông tin bên trong thì ta thấy được thông tin ngày tháng, toạ độ X, toạ độ Y và màu 120 và 0

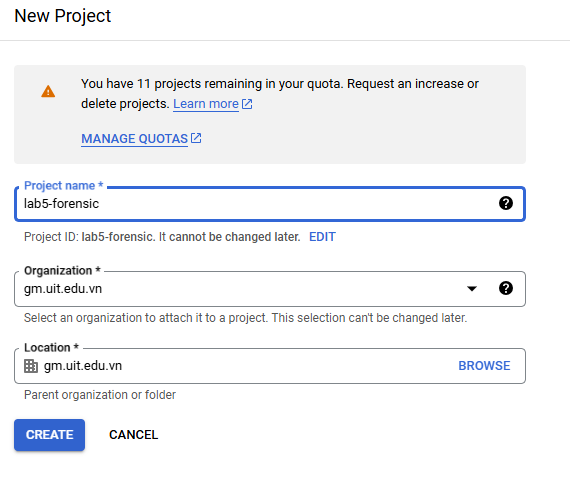


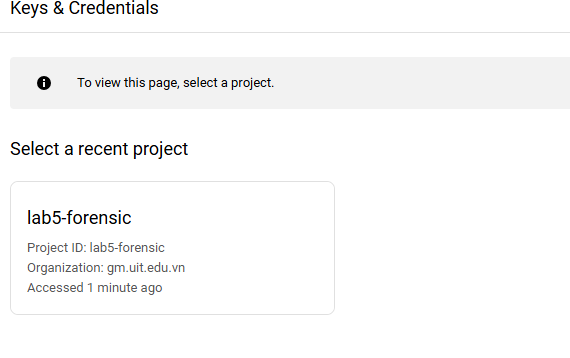
* Ta sẽ cần đọc dữ liệu từ tệp Location.db bằng Python và hiển thị các vị trí trên Google Maps bằng API của Google Maps.



* Lấy Key API







import sqlite3

import pandas as pd

from gmplot import gmplot

*# Bước 1: Kết nối tới cơ sở dữ liệu SQLite và đọc dữ liệu*

conn = sqlite3.connect('Location.db')

query = "SELECT \* FROM locations"

df = pd.read\_sql\_query(query, conn)

conn.close()

*# Bước 2: Phân tích dữ liệu thành các danh sách riêng biệt*

dates = df.iloc[:, 0].tolist()

latitudes = df.iloc[:, 1].astype(float).tolist()

longitudes = df.iloc[:, 2].astype(float).tolist()

altitudes = df.iloc[:, 3].astype(float).tolist()

*# Bước 3: Khởi tạo trình vẽ Google Maps*

*# Thay 'api\_key' bằng khóa API Google Maps*

gmap = gmplot.GoogleMapPlotter(latitudes[0], longitudes[0], 13, apikey='api\_key')

*# Bước 4: Vẽ các vị trí lên bản đồ*

gmap.scatter(latitudes, longitudes, color='red', size=40, marker=False)

*# Tùy chọn: Bạn có thể vẽ đường nối giữa các điểm*

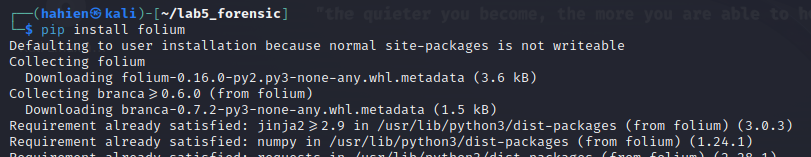
gmap.plot(latitudes, longitudes, 'blue', edge\_width=2.5)

*# Bước 5: Lưu bản đồ vào tệp HTML*

gmap.draw("map.html")

print("Bản đồ đã được tạo và lưu vào map.html")

* Vì tốn phí trên GG Map, nên em sẽ dùng leaflet



import sqlite3

import pandas as pd

import folium

*# Bước 1: Kết nối tới cơ sở dữ liệu SQLite và đọc dữ liệu*

conn = sqlite3.connect('Location.db')

*# Kiểm tra các bảng trong cơ sở dữ liệu*

cursor = conn.cursor()

cursor.execute("SELECT name FROM sqlite\_master WHERE type='table';")

tables = cursor.fetchall()

if ('locations',) in tables:

query = "SELECT \* FROM locations"

df = pd.read\_sql\_query(query, conn)

conn.close()

*# Bước 2: Phân tích dữ liệu thành các danh sách riêng biệt*

dates = df.iloc[:, 0].tolist()

latitudes = df.iloc[:, 1].astype(float).tolist()

longitudes = df.iloc[:, 2].astype(float).tolist()

altitudes = df.iloc[:, 3].astype(float).tolist()

*# Bước 3: Khởi tạo bản đồ Leaflet với folium*

*# Chọn tọa độ trung tâm bản đồ là tọa độ đầu tiên trong danh sách*

map\_center = [latitudes[0], longitudes[0]]

mymap = folium.Map(location=map\_center, zoom\_start=13)

*# Bước 4: Vẽ các vị trí lên bản đồ với màu sắc dựa trên độ cao*

for lat, lon, alt in zip(latitudes, longitudes, altitudes):

color = 'red' if alt == 0.0 else 'blue'

folium.CircleMarker(

location=[lat, lon],

radius=5,

color=color,

fill=True,

fill\_color=color

).add\_to(mymap)

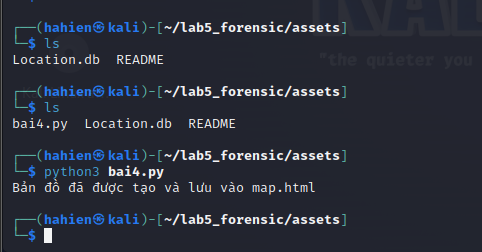
*# Bước 5: Lưu bản đồ vào tệp HTML*

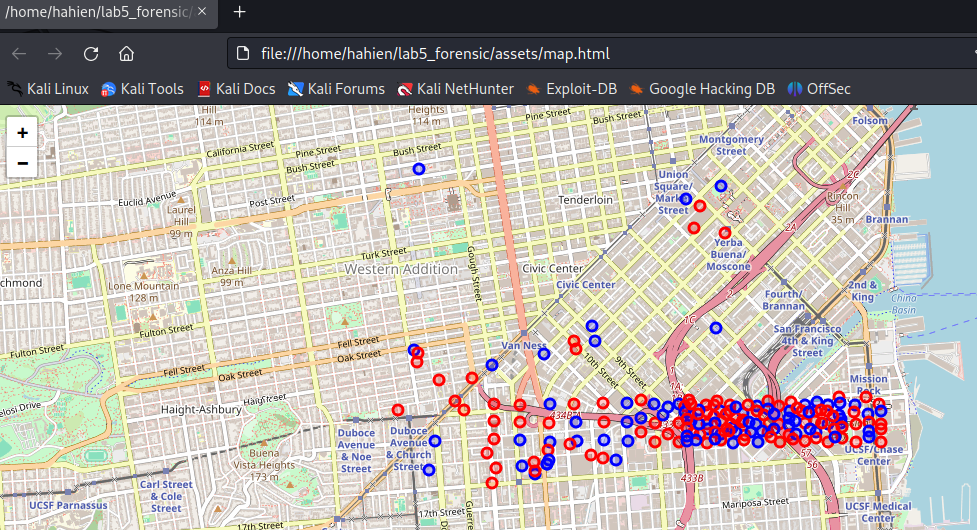
mymap.save("map.html")

print("Bản đồ đã được tạo và lưu vào map.html")

else:

print("Bảng 'locations' không tồn tại trong cơ sở dữ liệu.")





- Lọc điểm theo ngày để rõ hơn:

import sqlite3

import pandas as pd

import folium

*# Bước 1: Kết nối tới cơ sở dữ liệu SQLite và đọc dữ liệu*

conn = sqlite3.connect('Location.db')

*# Kiểm tra các bảng trong cơ sở dữ liệu*

cursor = conn.cursor()

cursor.execute("SELECT name FROM sqlite\_master WHERE type='table';")

tables = cursor.fetchall()

if ('locations',) in tables:

query = "SELECT \* FROM locations"

df = pd.read\_sql\_query(query, conn)

conn.close()

*# Bước 2: Phân tích dữ liệu thành các danh sách riêng biệt*

df['date'] = pd.to\_datetime(df.iloc[:, 0], format='%m/%d/%Y')

unique\_dates = df['date'].dt.date.unique()

for unique\_date in unique\_dates:

day\_df = df[df['date'].dt.date == unique\_date]

dates = day\_df.iloc[:, 0].tolist()

latitudes = day\_df.iloc[:, 1].astype(float).tolist()

longitudes = day\_df.iloc[:, 2].astype(float).tolist()

altitudes = day\_df.iloc[:, 3].astype(float).tolist()

*# Bước 3: Khởi tạo bản đồ Leaflet với folium*

*# Chọn tọa độ trung tâm bản đồ là tọa độ đầu tiên trong danh sách*

map\_center = [latitudes[0], longitudes[0]]

mymap = folium.Map(location=map\_center, zoom\_start=13)

*# Bước 4: Vẽ các vị trí lên bản đồ với màu sắc dựa trên độ cao*

for lat, lon, alt in zip(latitudes, longitudes, altitudes):

color = 'red' if alt == 0.0 else 'blue'

folium.CircleMarker(

location=[lat, lon],

radius=5,

color=color,

fill=True,

fill\_color=color

).add\_to(mymap)

*# Bước 5: Lưu bản đồ vào tệp HTML với tên tệp theo ngày*

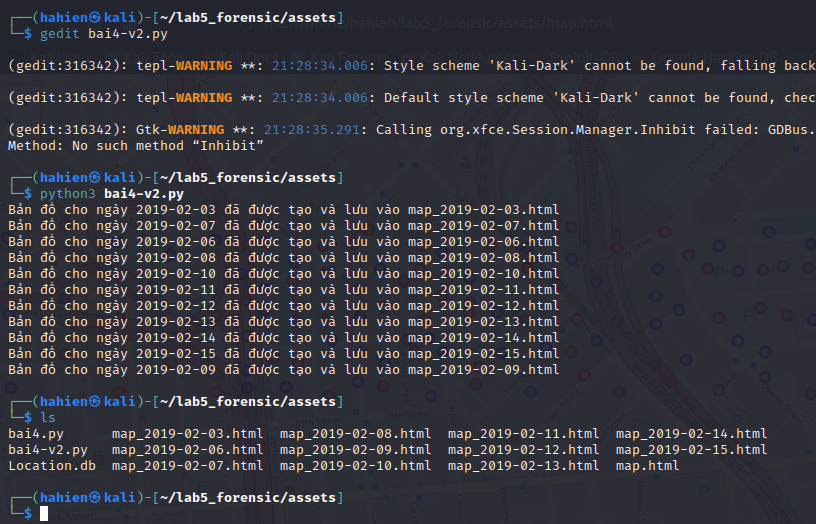
map\_filename = f"map\_{unique\_date}.html"

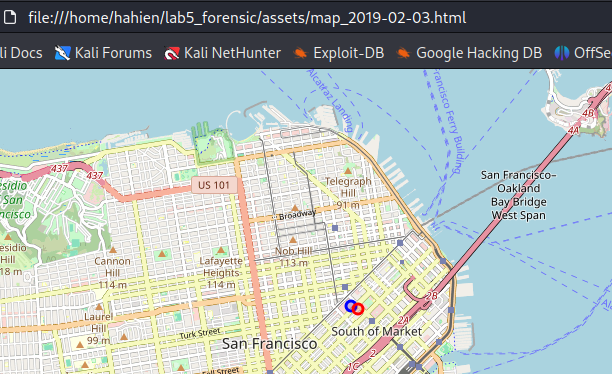
mymap.save(map\_filename)

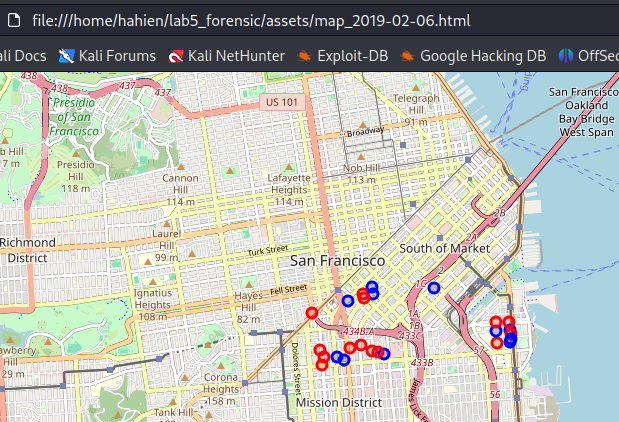
print(f"Bản đồ cho ngày {unique\_date} đã được tạo và lưu vào {map\_filename}")

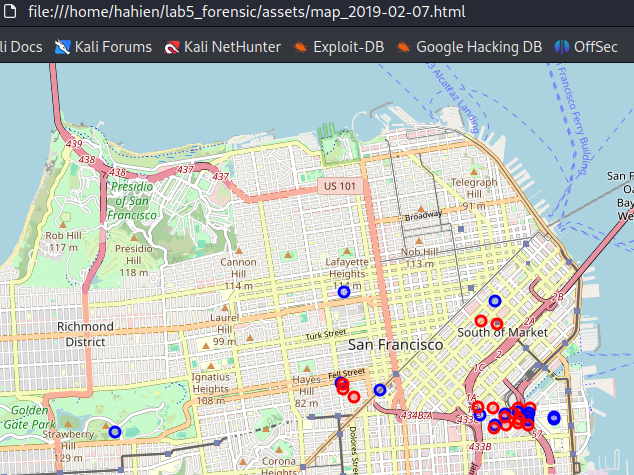
else:

print("Bảng 'locations' không tồn tại trong cơ sở dữ liệu.")

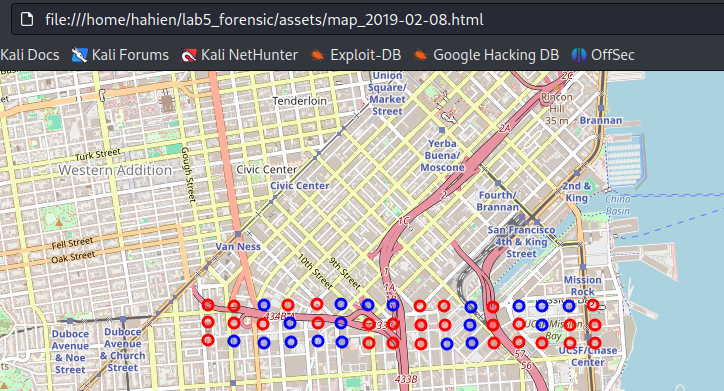




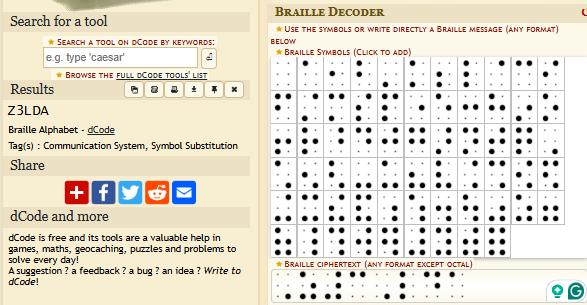




* Tiếp tục mở cái map khác thì vẫn không có gì đặc biệt, cho đến khi mở map ngàu 2/8/2019



* Nhìn ký hiệu trong hình vẽ được sắp xếp như mã Braille, thử decode ta có



FLAG: Z3LDA

1. **Kịch bản 04. Điều tra trên tập tin ứng dụng thu được.**

* Mô tả: Một ứng dụng thời tiết đơn giản có tính năng thu thập và hiển thị thông tin thời tiết.
* Tài nguyên: kb04\_tianqi.apk
* Yêu cầu – Gợi ý: Xác định phiên bản Andrọid đang chạy của ứng dụng. Sử dụng một số công cụ decomplie apk như Jadx để phân tích code ứng dụng. Flag có định dạng CTF{…}

---

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

# **YÊU CẦU CHUNG**

* Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, 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]-ExeX\_GroupY. (trong đó X là Thứ tự Bài tập, Y là mã số thứ tự nhóm trong danh sách mà GV phụ trách công bố).

*Ví dụ: [*NT101.K11.ANTT*]-Exe01\_Group03.*

* 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 nộp.
* Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá**:

* Hoàn thành tốt yêu cầu được giao.
* Có nội dung mở rộng, ứng dụng.

*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)