BÁO CÁO THỰC HÀNH

Môn học: Pháp chứng kỹ thuật số

Lab 5: Mobile Forensics

GVHD: Đoàn Minh Trung

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT522.N11.ATCL.1

STT	Họ và tên	MSSV	Email
1	Vũ Hoàng Thạch Thiết	20521957	20521957@gm.uit.edu.vn
2	Hoàng Thanh Lâm	20521513	20521513@gm.uit.edu.vn
3	Lê Viết Tài Mẫn	20521593	20521593@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Bài tập 1	100%
2	Bài tập 2	100%
3	Bài tập 3	100%
4	Bài tập 4	80%

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

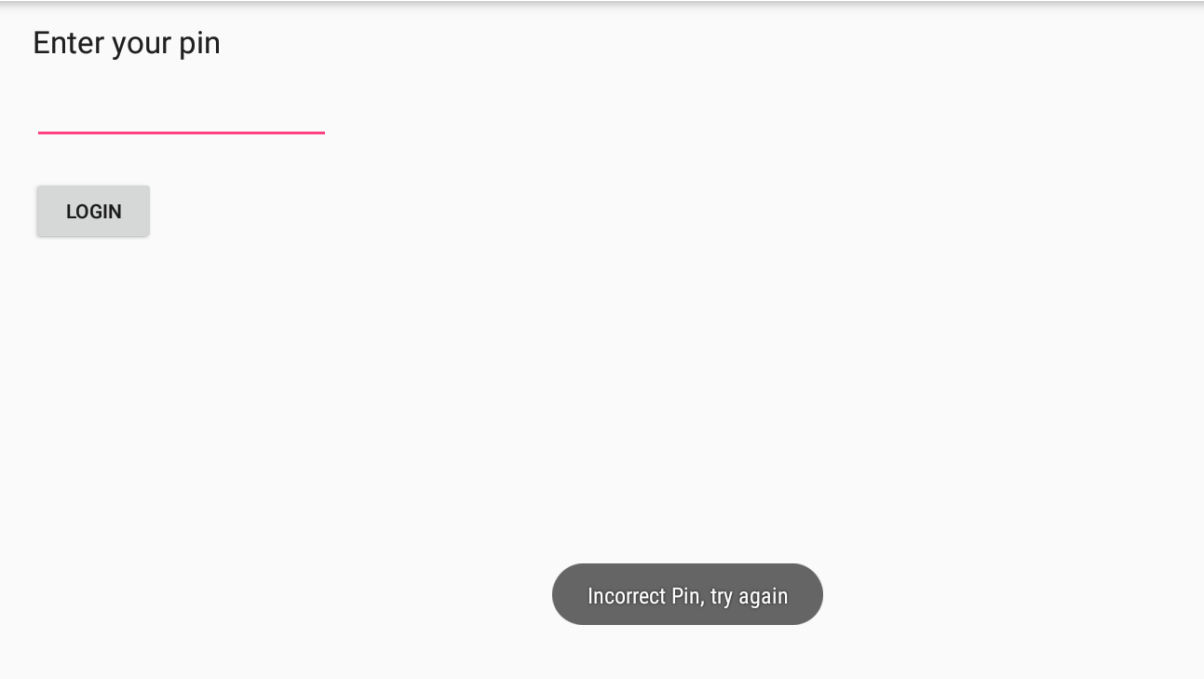
[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

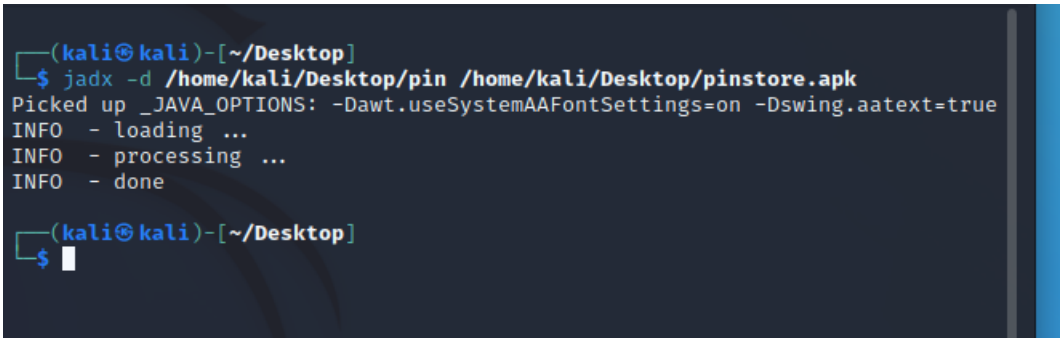
1. Kịch bản 01. Thực hiện phân tích ứng dụng Android

- 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 (decompile) trên mã nguồn Android để phân tích.

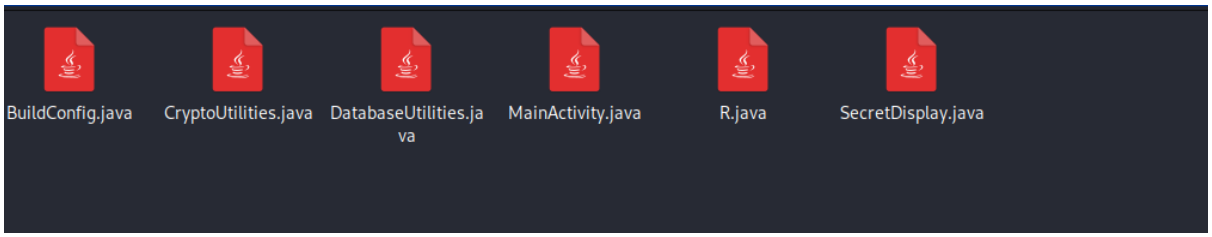
Đáp án:



- Khi cài đặt thì nó yêu cầu ta nhập pin
- Nếu sau thì in ra "Incorrect Pin, try again"
- Ta dùng công cụ jadx để decompile file apk



- Trong đường dẫn "/sources/pinlock/ctf/pinlock/com/pinstore/" ta thấy có các file



- Thường trong MainActitivy sẽ là chứa các Activity chính của chương trình

```
public void onClick(View view) {
    String enteredPin = MainActivity.this.pinEditText.getText().toString();
    String pinFromDB = null;
    String hashOfEnteredPin = null;
    try {
        DatabaseUtilities dbUtil = new DatabaseUtilities(MainActivity.this.getApplicationContext());
        pinFromDB = dbUtil.fetchPin();
    } catch (IOException e) {
        e.printStackTrace();
    }
    try {
        hashOfEnteredPin = CryptoUtilities.getHash(enteredPin);
    } catch (UnsupportedEncodingException e2) {
        e2.printStackTrace();
    } catch (NoSuchAlgorithmException e3) {
        e3.printStackTrace();
    }
    if (pinFromDB.equalsIgnoreCase(hashOfEnteredPin)) {
        Intent intent = new Intent(MainActivity.this, SecretDisplay.class);
        intent.putExtra("pin", enteredPin);
        MainActivity.this.startActivity(intent);
        return;
    }
    MainActivity.this.pinEditText.setText("");
    Toast.makeText(MainActivity.this, "Incorrect Pin, try again", 1).show();
}
```

- Đoạn code trong MainActivity
- Ta có thể thấy

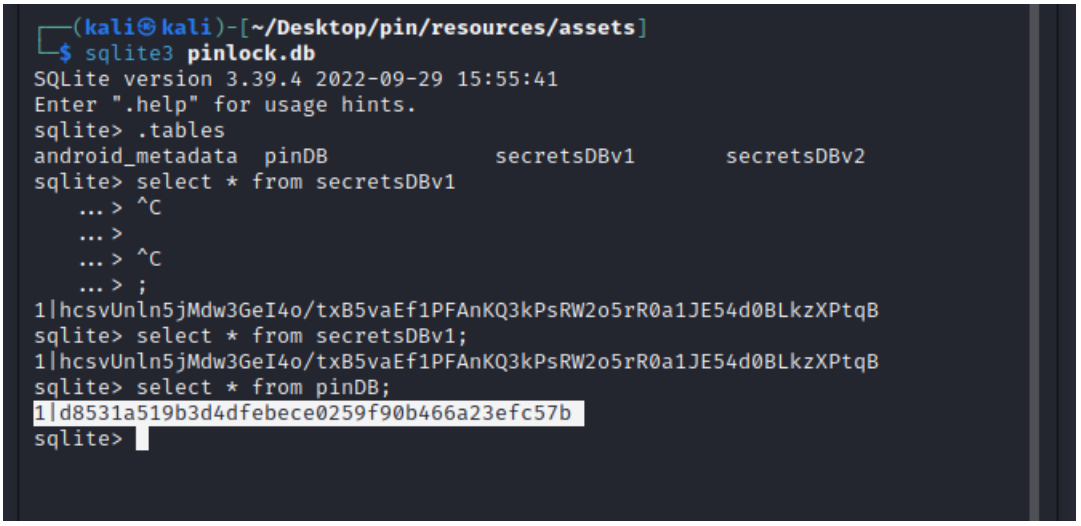
  + Chương trình khởi tạo biến để lấy thông tin pin ta nhập vào sau đó khởi tạo biến pinFromDB là null và biến hashOfEnteredPin là null

  + Ta có thể đoán là nó sẽ lấy mã pin người dùng nhập và mã PIN từ database. Sau đó nó sử dụng lớp CryptoUilitiese để băm mã PIN nhập vào và so sánh với mã PIN từ database

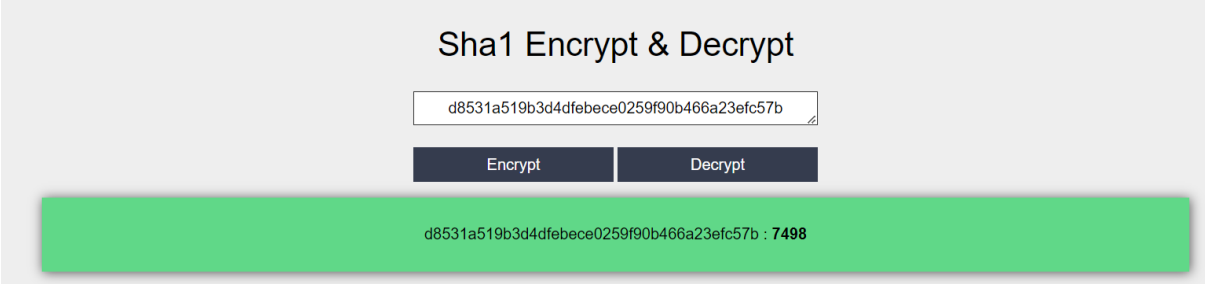
  + Nếu 2 mã PIN khớp thì nó sẽ chuyển qua màn hình SecretDisplay còn nếu không khớp thì sẽ có thông báo được hiển thị
- Vậy ta chỉ cần vào xem database của nó để xem mã hash là được

```
public static String getHash(String instance) throws NoSuchAlgorithmException, UnsupportedEncodingException {
    final byte[] bytes = instance.getBytes();
    instance = null;
    while (true) {
        try {
            instance = (String)MessageDigest.getInstance("SHA-1");
            ((MessageDigest)instance).update(bytes, 0, bytes.length);
            return getHex(((MessageDigest)instance).digest());
        }
        catch (final NoSuchAlgorithmException ex) {
            continue;
        }
        break;
    }
}
```

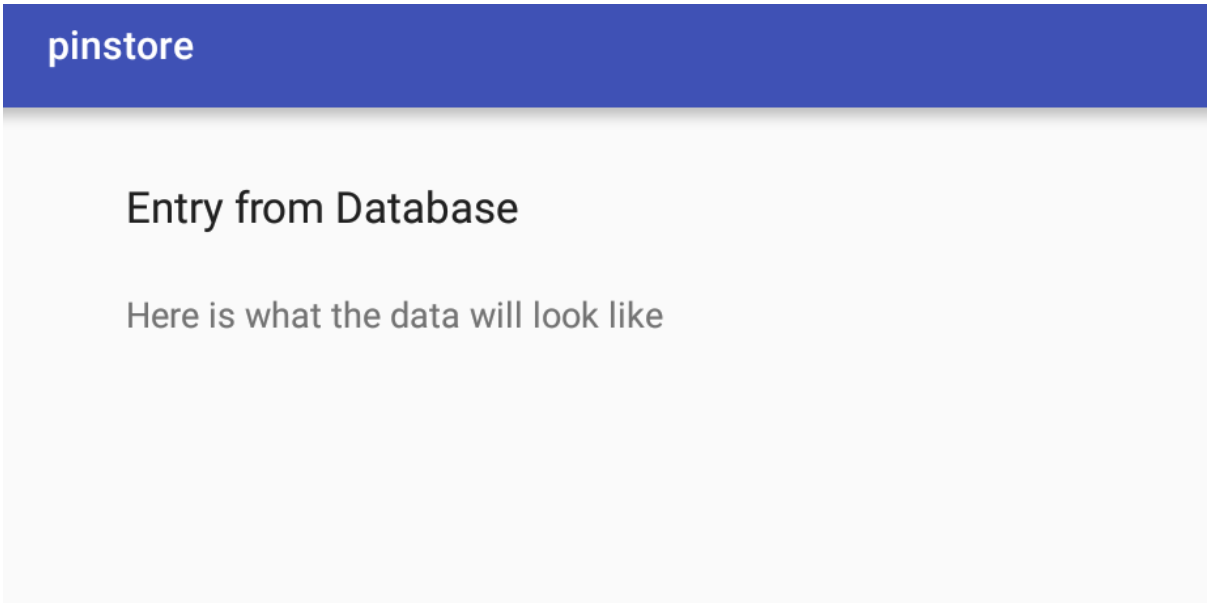
- Trong CryptoUtilities.Class ta thấy nó sử dụng SHA-1 để hash mã PIN của ta
- Khi decompile các file và xem trong các thư mục ta thấy trong thư mục "/resources/assets/" có 1 file tên là pinlock.db có vẻ như đây sẽ là file database của ta



- Dùng sqlite3 để xem các tables trong databse
- Xem qua các tables thì tables pinDB có vẻ là khớp với SHA-1 của ta



- Đưa vào các công cụ online để decrypt thì ta lấy được PIN là "7498"



- Kết quả khi ta nhập vào mã PIN đúng
- Nhưng có vẻ là chưa ra được flag của ta

```java
package pinlock.ctf.pinlock.com.pinstore;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;
/* loaded from: classes.dex */
public class SecretDisplay extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityDonut, android.app.Activity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_secret_display);
        Context context = getApplicationContext();
        TextView tv = (TextView) findViewById(R.id.secretTextView);
        String pin = getIntent().getStringExtra("pin");
        try {
            DatabaseUtilities dbUtils = new DatabaseUtilities(getApplicationContext());
            CryptoUtilities cryptoUtils = new CryptoUtilities("v1", pin);
            tv.setText(cryptoUtils.decrypt(dbUtils.fetchSecret()));
        } catch (Exception e) {
            Log.e("Pinlock", "exception", e);
        }
        Toast.makeText(context, pin, 1);
    }
}
```

- Ta xem code của 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 mãm bằng cách truyền phiên bản “v1” và “pin” vào
- Sau đó nó sử dụng hàm fetchSecret() để lấy chuỗi bí mật từ database sau đó chuỗi này được decrypt và kết quả giải mã được sẽ hiển thị lên giao diện

```java
    public String fetchSecret() throws IOException {
        openDB();
        Cursor cursor = this.db.rawQuery("SELECT entry FROM secretsDBv1", null);
        String secret = "";
        if (cursor.moveToFirst()) {
            secret = cursor.getString(0);
        }
        Log.d("secret", secret);
        cursor.close();
        return secret;
    }

    public String fetchPin() throws IOException {
        openDB();
        Cursor cursor = this.db.rawQuery("SELECT pin FROM pinDB", null);
        String pin = "";
        if (cursor.moveToFirst()) {
            pin = cursor.getString(0);
        }
        cursor.close();
        return pin;
    }
```

- Đây là hàm fetchSecret() nhưng ta nhập nó lại không phải là flag

```
    public SecretKeySpec getKey(String version) throws Exception {
        if (version.equalsIgnoreCase("v1")) {
            Log.d("Version", version);
            byte[] keyBytes = "t0ps3kr3tk3y".getBytes("UTF-8");
            MessageDigest md = MessageDigest.getInstance("SHA-1");
            SecretKeySpec keySpec = new SecretKeySpec(Arrays.copyOf(md.digest(keyBytes), 16), "AES");
            return keySpec;
        }
        Log.d("Version", version);
        byte[] salt = "SampleSalt".getBytes();
        char[] pinArray = this.pin.toCharArray();
        SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        KeySpec ks = new PBEKeySpec(pinArray, salt, 1000, 128);
        SecretKey secretKey = secretKeyFactory.generateSecret(ks);
        SecretKeySpec keySpec2 = new SecretKeySpec(secretKey.getEncoded(), "AES");
        return keySpec2;
    }
```

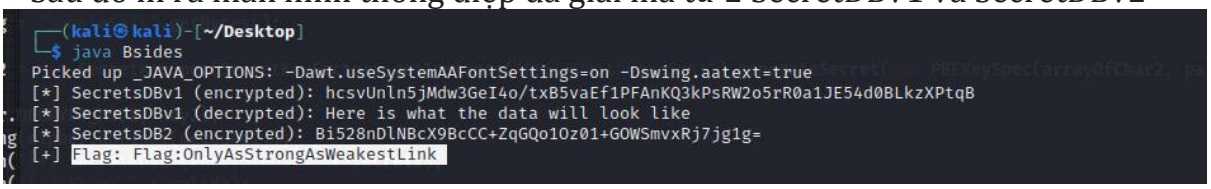
- Ta xem hàm getkey trong CyptoUtilities class
- Phương thức getKey(String version) trong lớp CryptoUtilities được sử dụng để tạo SecretKeySpec dựa trên phiên bản và pin.
- Nếu phiên bản là "v1", phương thức tạo một SecretKeySpec từ một chuỗi cố định "t0ps3kr3tk3y" và sử dụng thuật toán mã hóa AES.
- Nếu phiên bản không phải "v1", phương thức sử dụng thuật toán PBKDF2 và HMAC-SHA1 để tạo một SecretKey từ pin và salt. Sau đó, tạo một SecretKeySpec từ khóa đã được mã hóa và sử dụng thuật toán mã hóa 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.

  ➔ Ý tưởng là ta sẽ sử dụng pin và chuỗi secret từ secretsDBv2 là sẽ có flag của ta

```
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.util.Arrays;
import java.util.Base64;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

public class Bsides {

    public static void main(String[] args) {
        String pin = "7498";                                                                    // PinDB - d8531a519b3d4dfebece0259f90b466a23efc57b (SHA-1)
        String secret1 = "hcsvUnln5jMdw3GeI4o/txB5vaEf1PFAnKQ3kPsRW2o5rR0a1JE54d0BLkzXPtqB";    // SecretsDBv1
        String secret2 = "Bi528nDlNBcX9BcCC+ZqGQo1Oz01+GOWSmvxRj7jg1g=";                       // SecretsDBv2
        try {
            byte[] decoded1 = Base64.getDecoder().decode(secret1);
            byte[] decoded2 = Base64.getDecoder().decode(secret2);
            Cipher cipher;
            cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            SecretKeySpec key1;
            key1 = new SecretKeySpec(Arrays.copyOf(MessageDigest.getInstance("SHA-1").digest("t0ps3kr3tk3y".getBytes()), 16), "AES");
            cipher.init(Cipher.DECRYPT_MODE, key1);
            String salida = new String(cipher.doFinal(decoded1), "UTF-8");
            System.out.println("[*] SecretsDBv1 (encrypted): " + secret1);
            System.out.println("[*] SecretsDBv1 (decrypted): " + salida);

            char[] arrayOfChar2 = pin.toCharArray();
            byte[] paramString = "SampleSalt".getBytes();

            SecretKeySpec key2 = new SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret(new PBEKeySpec(arrayOfChar2, paramString, 1000, 128)).getEncoded(), "AES");

            cipher.init(Cipher.DECRYPT_MODE, key2);
            salida = new String(cipher.doFinal(decoded2));
            System.out.println("[*] SecretsDB2 (encrypted): " + secret2);
            System.out.println("[+] Flag: " + salida);

        } catch (UnsupportedEncodingException | NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | IllegalBlockSizeException | BadPaddingException | InvalidKeySpecException e) {
            e.printStackTrace();
        }
    }
}
```

- Đây là code của ta
- Đoạn code như sau:

+ Khai báo chuỗi pin "7498" đây là giá trị mã hóa SHA của mã pin có sẵn trong pinDB
+ Khai báo 2 chuỗi secret1 và secret2 lấy được từ database chứa thông diệp mã hóa
+ Sử dụng các hàm mã hóa và giải mã AES với các SecretKeySpec của từng phiên bản database mã hóa trong hàm getKey()
+ Sau đó in ra màn hình thông điệp đã giải mã từ 2 SecretDBv1 và SecretDBv2



- Trong đó SecretDBv2 là flag của ta

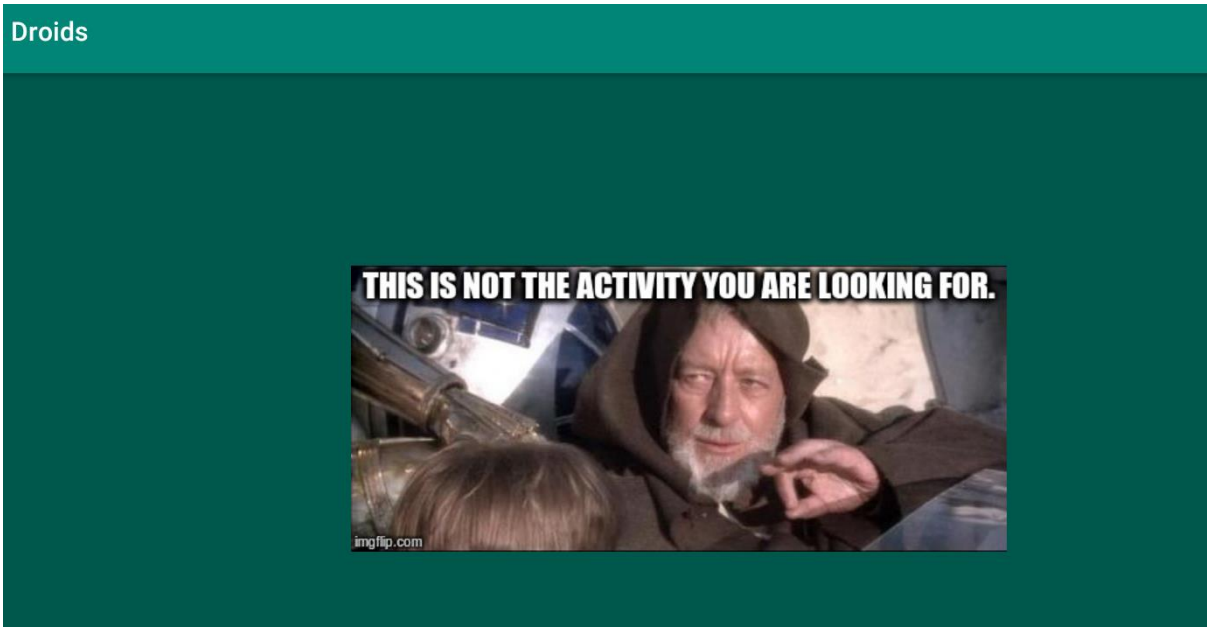
➔ Flag: OnlyAsStrongAsWeakestLink

2. 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{....}

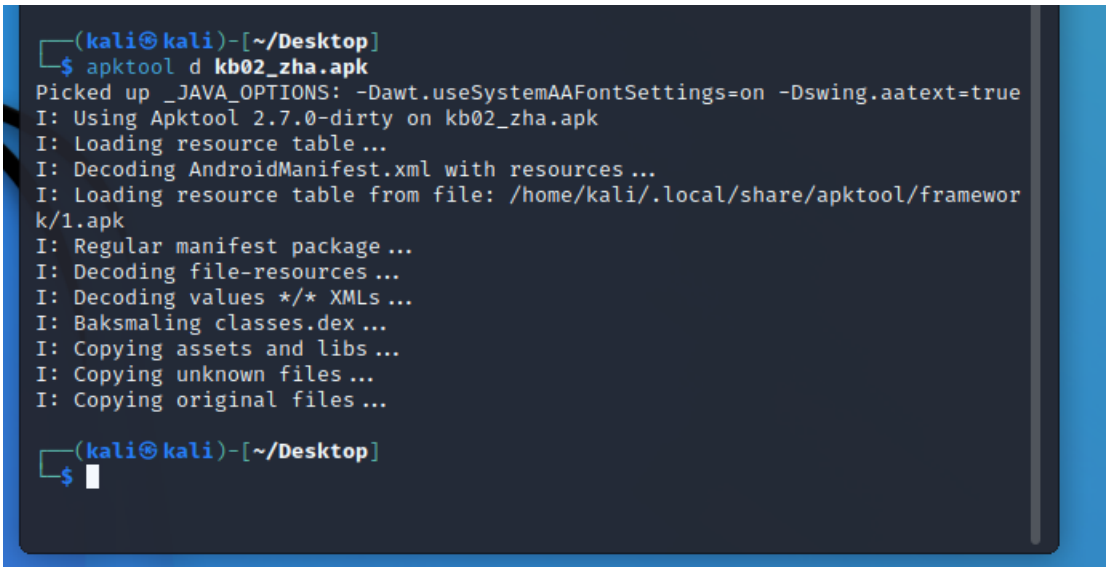
Đáp án:

Gợi ý:

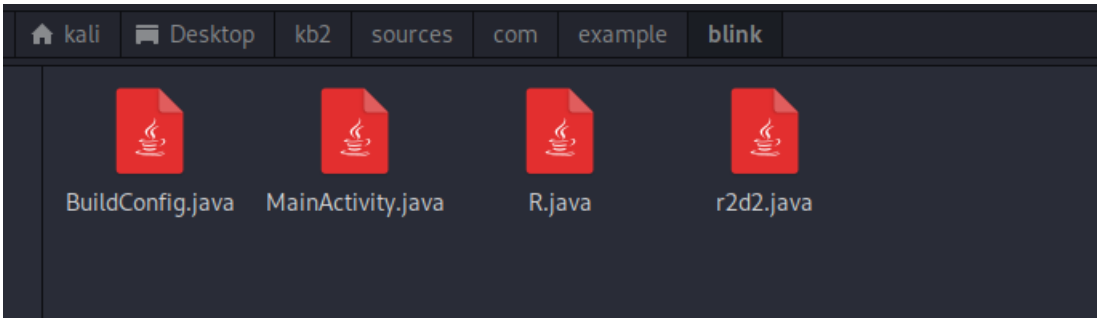


- Khi cài đặt vào truy cập vào ứng dụng ta được như hình trên

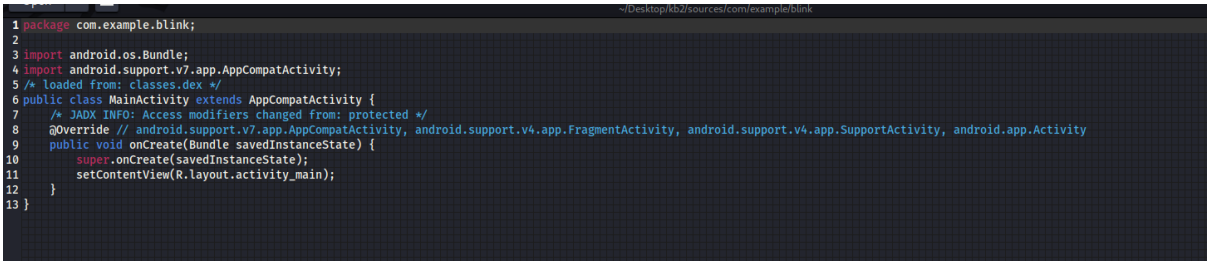
- Ta cũng dùng công cụ apktool để decompile file apk để có thể xem được file smali the gợi ý



- Ta cũng dùng công cụ jadx để có thể xem java code của chương trình



- Ta thấy được 4 file



- Xem MainActivity thì có vẻ nó không phải là luồng hoạt động chính của chương trình
- Vậy còn file r2d2.java

```java
package com.example.blink;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Base64;
import android.widget.ImageView;
/* loaded from: classes.dex */
public class r2d2 extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.SupportActi
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_r2d2);
        ImageView image = (ImageView) findViewById(R.id.imageView);
        String imageString = " 
2wBDAQMDAwQDBAgEBAgQCwkLEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBD/wgARCAGEAmwDAREAAhEBAxEB/8QAHQABAAICAwEBAA
aAAwDAQACEAMQAAABqoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
adPd9LFqHUu061t7Wjehsxj6qxGnocfq1xgROAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAD96Uy+WyrC8zztv87f5YtaqM0zhITSSSu0eFurMD9FwoG9zheZeYgAAAAAAAAAAAAAA
```

```java
sqKQCoHnpE+CRiTNi7WUiUDcRK3COkXYQYVAjyPI5oJciJggAJHmXJm5AuSFTjotPbwkrgDcREo0dBEhF5nrLXNW8joiRQCxu8ta3QoFCEe4pF4WZdWbJLQc
Y1tQUQRAgi0GwOdJxdkI14WbwPpVBGXqV1ABIJMBV4fWsW85hyCTcpisESFCRAAlAlaJL0Uc20SgRkDKLF3VSgLSRxmJKgT88gYOu8YZqW5nW8GQwBCUyA9P
Q==".split(",")[1];
        byte[] imageBytes = Base64.decode(imageString, 0);
        Bitmap decodedImage = BitmapFactory.decodeByteArray(imageBytes, 0, imageBytes.length);
        image.setImageBitmap(decodedImage);
    }
```

- Biến imageString được lưu dưới dạng base64 và chứa dữ liệu hình ảnh
- Sau đó có hàm được dùng để decode
- Ta sẽ viết chương trình sử dụng lớp Base64 của java để giải mã base64 sau đó lưu nó vào dưới dạng hình ảnh

```java
import java.util.Base64;
import java.io.FileOutputStream;
import java.io.IOException;

public class Base64Decoder {
    public static void main(String[] args) {
        String base64String = "
2wBDAQMDAwQDBAgEBAgQCwkLEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBD/wgARCAGEAmwDAREAAhEBAxEB/8QAHQABAAICAwEBAAAAAAAAAAAAAAcIAQYCBAUJA//EABwBAQABBQEBAAAAAAAAAA
aAAwDAQACEAMQAAABqoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
adPd9LFqHUu061t7Wjehsxj6qxGnocfq1xgROAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAD96Uy+WyrC8zztv87f5YtaqM0zhITSSSu0eFurMD9FwoG9zheZeYgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAM1Mw3DWXrX8l2EgeLy+VMqQJIAwnKATgzMa
wXc8qJQAAAAAAAC46Wws106hr4A9/idWuMUsAAAAAAAAAAAAAAAAAAAAAAAAA5FteT7KauZ7DkAAAAAAAcai/TGft8SrXVdbqOfRxAAPVJDNeNJMAA5G/
HqkdHngAHZPVPIPwAAAAAAAAAAAAAABzmb68B2+1+YyyAAAAAABxPI3lisnW9bCvrsfgp4xIAG6l9jZz8SqhV8wDJcAsecjSCgxroBIxdw3A0opAaCAAAAAAAAAAAAAAdmmfon837z99bdAAAAACWJcciiGeiYdYuma3w8yMUgAALqFgADpHzSP
m7fc8KoAAAAAK2q+kxKtdcwIu9DjcZnAgAABeYm4A/M+aZrYJGPoaZBgrcU9Bk+nB7YBqx81TAAAAAAAAAAAAAAPUsVfQ35u3jDucgAAAABLT/V4lZOr4EW+hsflXSoYSAABMRek7AIRKOHEH7l/
iTTJ55QEjoGS5xYQyYK3FPjAAAAAAAAAAAAAAP3ifop83b3sa24AAAABxMzKuF2iMfa4laOn4Gibm1xpAAAZJOJgNaIEOgAD1CwR6pChH4APRLOkgEblZzogAAAAAAAAAAAAAHNN8uCbjbvLZRAAAHE5A4nJJA4y/
PMohnomBW7o+JredaxAYgAAAAAAAAAAAAAAAAAAAAAAAAAAORbHlOznDmOeOIOQAAAAAONUpo6W0ogHpWBX73+L5d63ikAAAAAAAAAAAAAAAAAAAAAAAAAAAAOVUS75bKuJwXb4przEAAAAAAAADEsS8LfWK5dU10JeuxevcjjTIAAAAAAAAAAAA
+JzbR8X2vKyAAAAAAAAA4gxeiM/cYdWOta7Usy2AAAAAAAAAAAAAAAAAAAAAAAAAAO3TN4OI7fePF5QAAAAAAAAAHGp5+3sUW79p9XzaAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAByluusvXa4Lt/Q1FzlEgAAAAAAAADFccL1NSu4auF/
WYfGJAAAAAAAAAAAAAAAAAAAAAAAAAAGZCVPPX7hcN2vc1t4AAAAAAAAcciitnYtbXP3WDgwkAAAAAAAAAAAAAAAAAAAAAAAZlKHnr9veJ7P09Xk5tsHJA4gHIHE5AA41PyzKK0dd1tfPc4XEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGTMtq19y1
x8QxAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAZP0pS15vJsTzXO3jy1/Ni5kUwgElUqYxcp17fW4A6RhQZ7PC6V2MAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAzLnE7pq7speRv7/56/s+mu+riVjqXqNY3VrR99ZjL1eNHO8o/
CujESAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMgzMggcYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACx5Ywi808mQEdEikakFlrjTzfzQTxD9
CATsE7lKy6RQUu0eIVNLUn7E4FfCw5UA3U3Q3c65VEscSKR2R+Tmaido8A2U3E+axbYhcgYAAAAAAAAAAAAAAAAAAH0IO8bWfM4+iBAB2CbD51FqDRSyBoJT8tkTibeUmLsGiEdG2nlE1lZiRjbiNDkTcQsTARqaEWQPmsXpPnWAAAAAAAAAAAA
U847p6ZqJtx0TXDcTaCxh89gAAAAAAAAACWzYT1DzzTjaTXyJwAAAAAAAAAAADYzXAAAAAAAAAAAAAesdU6YAAAAAAAAAAAO6DJ0QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAf//EADIQAAEDBAAEBAUDBAMAAAAAAAAAYDBAUAAQIHETZAUBAVF
2gAIAQEAAQUC/wDODhXw3r4cq+HKvhvXD+DY4ZZ3igmflrsNS42prroYbUiNQDevLI2vKo6lYGFXpyAi7mn+pmClSuuiCOsu3Xb5/wAATTzVzgdaychUQJwkNb2pCFjJVOe1ZnhT2Pdx63fYAakiBwPhkTA4cPemICMnESgGfwOfew8JckGTJi1
QZY454mmvvgq+PDvISFqzyqCCTVHozsG42v/
bu4qOLkUizZt2DbpL241sEQ8rX7q2bqulxSASH4npiSShWcev8v5vdNXQFl3FulUUTSTJdlt21PpB3Ir90bI5uV4WLRhozpJ4sioBMiM5Sez7trSI+vnejdPGzFEn2ZmpZdws6U7vqthdvB9ERncXB2nCWUnluP3MI19JK46xL88ZUOIYe322te
Y09FpNx/wAXTZJ23mGfl8p4BkanLk20OOPjumKSsj4gUJhCDfjseDwmBm/56yM/cfenSSMgESQ7k5y978fa1G6SVEvHLK2NiJwm7n/AA165SaGP2bpcpWivGHUwWifEmUwRHutjP3H3p0WiiDAgAJaGrLG+N/
Z1uVYjsskqmtj4bHL0YWLvfjfwSUzSUCi1uSRXg7eN2SBuS5E0146oJ030X47bJ027HrUsssFEFMVUfe/NEAHETVpwPl4HP2ONDR/ODdsN2JfBL7ilHWDp24er/
YwkXsY4jNzPkU1t2Y3TJDOZJcvsZPnUe4gNxJ2R9Vw+iDcPzEXTtw9X621Cbmzsc6BRPBXEh1pHv6l4CThl+9aqf8Az4XonbJq+RIdX2vT1g7YK941tLeXT/SSsJGTKRDrN6ypVBRDPuzdbNutBSqU1F9LOCkRPYEABKw9ZY3xv3XV8/
Zq76a9uNEIHEzVp8eejrruiCyjdUSIEiCJ6Z++bxrSfmXE3J91GCFcdkWD5tJNOkyythY/
LfNnHdwwwWHl2zpB2j0XG1qPDn5lce8iJm7HlI+RZyjboFFMEcDTYN3NXv3uEIJGBcDxzEzmPvTRLEwKROayBAp33FTLDKA2LLxdQ5nBTVrXtf2fxUmQREPhP7Rcr05duHav8AtlwqKL56Huw20tamuzBpekC8bcV57DV57DUsUD7enOxhhvT7b
wC19faybzbPAGEE0yzUkU6aavDRwhhvS4Er0uBKggQUelfpcCVsMGFYIXCRNQumM9XBDZmBAQnNCnpaC0U6iifL9ZhYyQj2ygoYHxzHV4LfE6GGjMwH9UDcWg5Ag50nsLX/APSmQEBCc0J7KC4KGQT1YD4pwY2Jy+w/
S4EoqD4pgf8ApcCVtAOGx2E1aGwBFGelwJW0hiMG5MCAhOaE9piEcNOG+sQZRAMCxiWmDnVkUxiO0jiOLceL3TjDaVPCshF5wWeOZAc2qVT44vp+SezD8xfu4sZlTkpm2Wjbf59yuFcynVvIuy5J/GL1qDG2A3uTlFP9N0lZXdFaYdOHTXaidsw
F+ZkE7J4Q0NpnPNUuoijrrnlbs5c1C1sgHYusMnu72nxMNW8i7QifNRBj/dkVZXxK7KeYA3aYT9m2hJ5Qp764SNSz/KVlArlLeP+xoz8yEe0lWWygeChozU4tIj8fs6RSkjHVvIs8MMCJZ26RZNtOZXUF9ycop/pq/
8zUHnrkOSItqvSKF1byLwtx3Bd3YQ0rzVLuvo2irRNZyi5+ZMbs5cAW1mgazffM2Ht1rZwG6t5FWSTcIoJWQRLOaQOQZzgYUg80KZdogr2yhD1o2f7N9OQqiRBBqQhXKW8f8AY0Z+TN46jxdqdTK8rFzMUawpFDKj83q3kXaZHNDjGXMCWeS0×f
TWytbc9apGoKfan4YLRAjq3kV3IfTS5PF2mh/S3NWxHF2gjjlbLGBc/UFe7OXIttZlGNRTNsZG7X6sR1byK0lOJTRZzTqsYbw8Fsc+YFKPaNdTLaXFJQQaShJJSTOIYyLu8hIBfKW8f9jRn5PuTa0bnf4twJ2wMdW8i7v/
AGqtKzTTFqVDTcri+OOGLsvY32nhnipiJiDQST3FOtW0Fq3kXYch5VJ0IRXk+1docijLq70d144u7m9pNvrWtIG8grsVyji5baxwySCZqU8p29RZzTG8W4D2mLmJOFcp7eMsE5snnSLKmexTKPaThPOEl4MonRqn2wzCSZ1pyaaR80XAMITqxZ6
hX//xAAmEQABBAEDBAIDAQAAAAAAAABAAIDEQQSMVAhMDJAEBMUYJBR/9oACAEDAQE/
Af6VXaa0lNxyUMQoYoQxmr6gvrCOO0o4gTsWk6AhaSFf6DFj2hjgLSAtSpdfmvksBT8cFSYyrnmQlRwhV3aTogVNCRzkMOpCOkB3×1TowVPBp25m1DFZTG6R6ZFrIgN9OYhh1JjAFfqEAqaHTyzRZUEdes4LIeCEeVxY7KrT6pkpS5VIyF3KgWo
YPTLlkTV05kqGYsUcwKq+/VIvpTT3st+apMk0qLJV96TIpST2t+dtMmITMhqEgKBtX8Wr+LWy+wBHIClyCdkST+g9U2YhNyiEMtqGS0r8hq/IajlNCOW1OyU6YlFx/SLKs/wBNf//
EADURAAEDAwMCAwUHBAMAAAAAAAEAAgMEBREGITESUDJAQRMUIjBCFRYjM1NhcRBRYJAgJEP/2gAIAQIBAT8B/wBcOVkLIWR/gwPVynubHvlVd/oKIfE7JVXruPiAKbXNbJsFNqavmPjX23X/AKi+26/9RN1DXxnxqn1tXx+J2VSa9HE4VJqegq
qwmywvGYzlbnhZK5785wYMuV01fSUbSIdyrjqmsrzhpwFLMXeuV/KwFj/AIfCiB6KMgcnCob/AFVA78NytWt4pWhtVsVFVR1Iy1EY73whsrndKe2s63ndXjVktyJj4anPLuVn5mSrXqKptr9zsrLqCmubR1nBWfRqcOjvPBV/
1FHbGY+pXG5z17+qUpzsj57n9fKp6t9McsK03qptTiGY7oO6viPd+FnC1DqFlsiLB4lV1z62UvkPk4pHQn2jVpbVHvX/
AF51jPHdhyr1dY7fF1HlXO5Pr5C56JB48nlQTmB3VGtL337Ri6H8hHussop2F7uFqm9G4TlrePKlo9EGrTdDXmcPpxso3P8AZhr+e66zu4pofdG8lEl2×58rT0MlY/
```

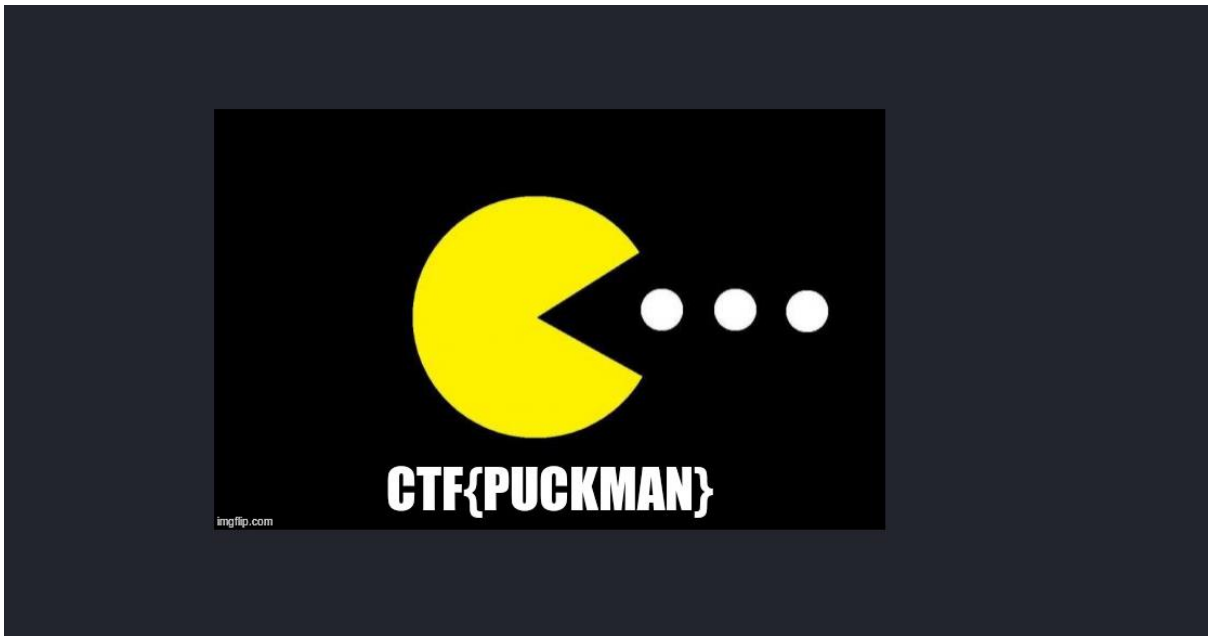
```
        // Tách phần dữ liệu base64 từ chuỗi
        String[] parts = base64String.split(",");
        String base64Data = parts[1];

        // Giải mã base64 thành dữ liệu bytes
        byte[] imageData = Base64.getDecoder().decode(base64Data);

        // Ghi dữ liệu hình ảnh vào tệp
        try (FileOutputStream fos = new FileOutputStream("image.jpg")) {
            fos.write(imageData);
            System.out.println("Đã giải mã và lưu hình ảnh thành công.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- Phần base64String ta chỉ việc đưa hết chuỗi cần giải mã vào là được
- Lưu file trên dưới tên Base64Decoder.java trùng tên với tên class

```
┌──(kali㉿kali)-[~/Desktop]
└─$ javac Base64Decoder.java
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ java Base64Decoder
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Đã giải mã và lưu hình ảnh thành công.

┌──(kali㉿kali)-[~/Desktop]
└─$
```

- Sử dụng 2 câu lệnh trên để compile và chạy

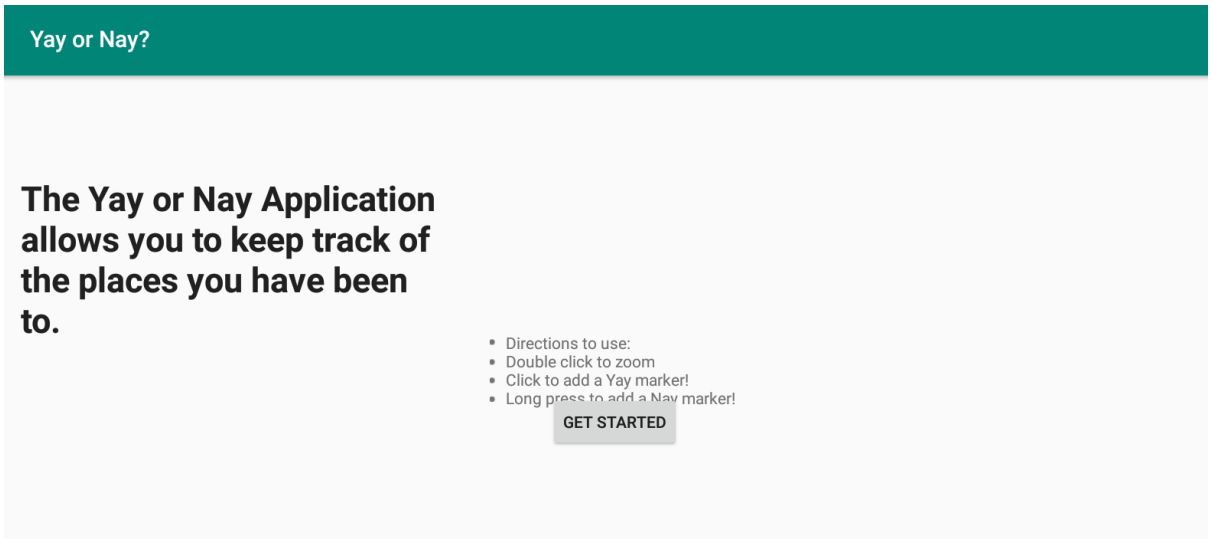


- Ta có được file hình ảnh
  ➔ Flag: CTF{PUCKMAN}

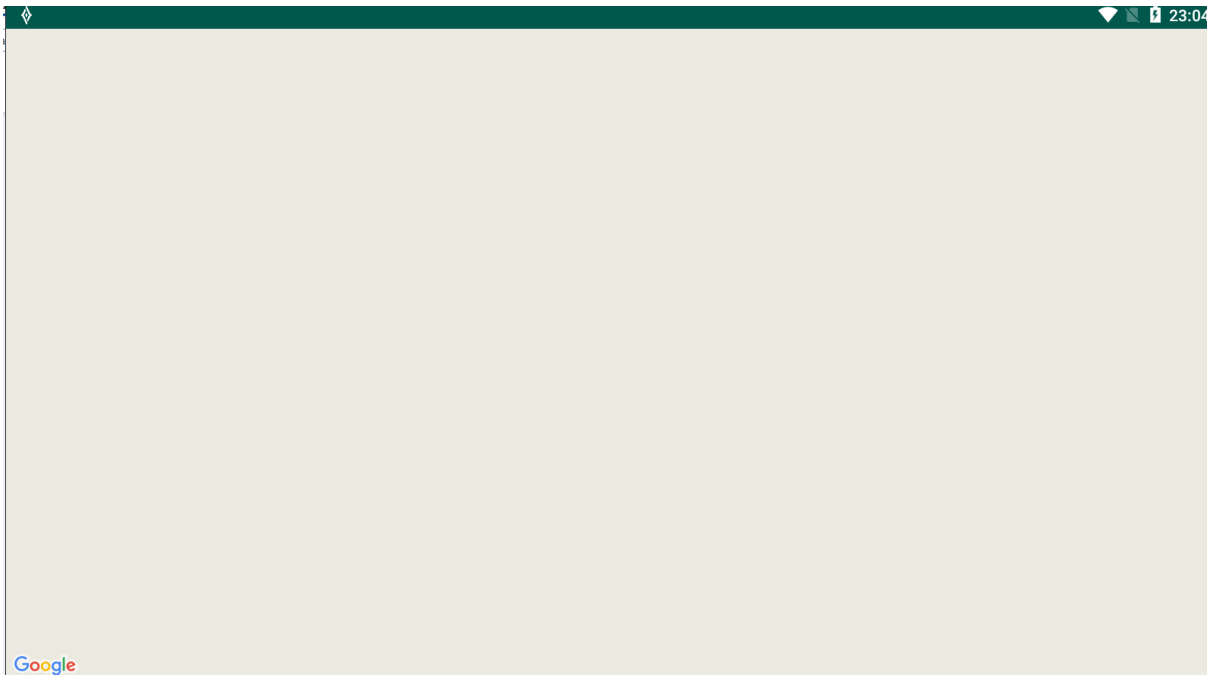
3. Kịch bản 03. Thực hiện phân tích tập tin ứng dụng thu được.

- 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:



- Ta có giao diện ứng dụng như trên



- Khi ấn get start thì không thấy gì hết có vẻ như API đã cũ

```
┌──(kali㉿kali)-[~/Desktop/kb03_yon/assets]
└─$ sqlite3 Location.db
SQLite version 3.39.4 2022-09-29 15:55:41
Enter ".help" for usage hints.
sqlite> .tables
android_metadata  locations
sqlite> select * from locations
   ...> ;
02/03/2019|37.7842927|-122.4053593|120.0
02/03/2019|37.7838412|-122.4041845|0.0
02/07/2019|37.7863323436302|-122.42828886956|120.0
02/07/2019|37.7851367932719|-122.402353584766|120.0
02/07/2019|37.782343920755|-122.404699847102|0.0
02/07/2019|37.7819822174966|-122.401994504035|0.0
02/07/2019|37.7673615647781|-122.467592954636|120.0
02/07/2019|37.7740060120854|-122.428736798465|120.0
02/07/2019|37.7738507141208|-122.428377382457|0.0
02/07/2019|37.7732178582912|-122.428442761302|0.0
02/07/2019|37.7720162653569|-122.426562532783|0.0
02/07/2019|37.7730111454145|-122.422018200159|120.0
02/06/2019|37.7737341079262|-122.417571097612|120.0
02/06/2019|37.7068383881913|-122.344666644931|120.0
02/06/2019|37.7087049563015|-122.443877533078|0.0
02/06/2019|37.6842005525645|-122.443419881165|0.0
02/06/2019|37.6847434257611|-122.441491037607|0.0
02/08/2019|37.7703669470161|-122.421883132294|0.0
02/08/2019|37.7692135433168|-122.421840216949|0.0
02/08/2019|37.7679922727318|-122.421840216949|0.0
02/08/2019|37.7703330236347|-122.419651534393|0.0
02/08/2019|37.7691456954801|-122.419651534393|0.0
```

- Theo hint của bài thì ta xem database của app khi đã reverse file apk ta được file Locations.db
- Ta dùng sqlite3 để xem
- Theo ta thấy thì nó sẽ có 4 cột
  + Cột đầu tiên là ngày tháng năm
  + Cột thứ 2,3 là tọa độ của điểm
  + Cột cuối là giá trị màu là 120.0 hoặc 0.0

```
/* JADX INFO: Access modifiers changed from: package-private */
public Location(Date date, double latitude, double longitude, float color) {
    this.date = date;
    this.latitude = latitude;
    this.longitude = longitude;
    this.color = color;
}
```

- Thông tin 1 điểm ta thấy được trong class Location

```
28
29    @Override // com.google.android.gms.maps.OnMapReadyCallback
30    public void onMapReady(GoogleMap googleMap) {
31        this.mMap = googleMap;
32        this.mMap.setOnMapLongClickListener(this);
33        this.mMap.setOnMapClickListener(this);
34        try {
35            DatabaseUtils dbUtil = new DatabaseUtils(getApplicationContext());
36            ArrayList<Location> locations = dbUtil.fetchLocations();
37            Iterator<Location> it = locations.iterator();
38            while (it.hasNext()) {
39                Location location = it.next();
40                LatLng temp = new LatLng(location.latitude, location.longitude);
41                float color = 120.0f;
42                String label = "Yay!";
43                if (location.color == 0.0d) {
44                    color = 0.0f;
45                    label = "Nay!";
46                }
47                this.mMap.addMarker(new MarkerOptions().position(temp).title(label).icon(BitmapDescriptorFactory.defaultMarker(color)));
48            }
49            LatLng bSidesSF = new LatLng(37.7842927d, -122.4037178d);
50            this.mMap.moveCamera(CameraUpdateFactory.newLatLng(bSidesSF));
51            this.mMap.animateCamera(CameraUpdateFactory.zoomTo(10.0f));
52        } catch (IOException e) {
53            e.printStackTrace();
54        }
55    }
56
```

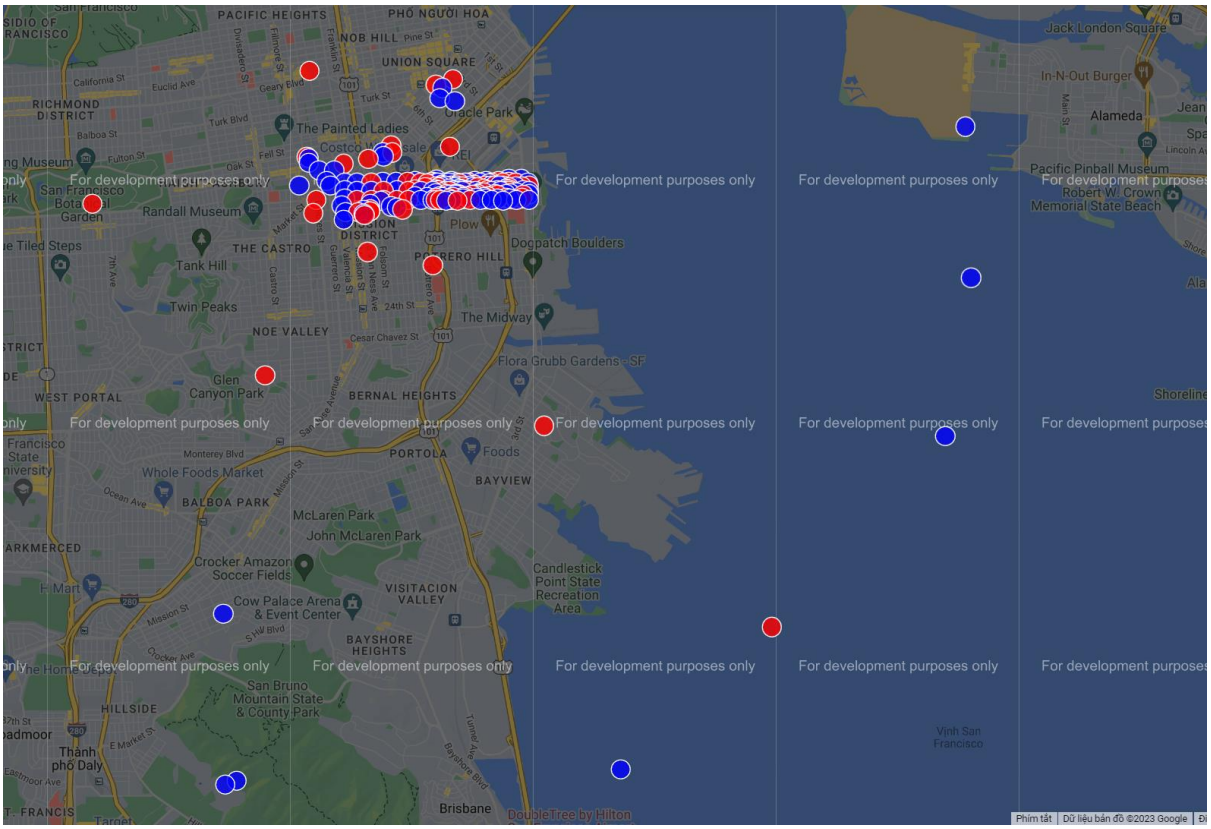
- Trong class MapsActivity thì ta thấy hàm này mỗi khi map được load lên thì sẽ đồng thời load tất cả các điểm đánh dấu cũ từ trong cơ sở dữ liệu ra
- Và thông tin của 1 điểm được định nghĩa bằng class Location như trên
- Do ứng dụng API khá cũ nên ta sẽ tạo tài khoản google development console để lấy Javasript maps API và tự load những điểm này lên

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Google Maps Example</title>
5      <script src="https://maps.googleapis.com/maps/api/js?key=KHÓA_API_CỦA_BẠN"></script>
6    </head>
7    <body>
8      <div id="map" style="height: 400px; width: 100%;"></div>
9
10     <script>
11       function initMap() {
12         var map = new google.maps.Map(document.getElementById('map'), {
13           zoom: 10,
14           center: { lat: 37.7749, lng: -122.4194 },
15         });
16
17         var locations = [
18           [37.7842927, -122.4053593, 'red'],
19           [37.7838412, -122.4041845, 'blue'],
20           // Thêm nhiều địa điểm khác nếu cần
21         ];
22
23         for (var i = 0; i < locations.length; i++) {
24           var location = locations[i];
25           var marker = new google.maps.Marker({
26             position: { lat: location[0], lng: location[1] },
27             map: map,
28             icon: {
29               path: google.maps.SymbolPath.CIRCLE,
30               fillColor: location[2],
31               fillOpacity: 1,
32               strokeWeight: 0,
33               scale: 10,
34             },
35           });
36         }
37       }
38
39       // Gọi initMap khi trang đã tải xong
40       google.maps.event.addDomListener(window, 'load', initMap);
41     </script>
```

```
38
39        // Gọi initMap khi trang đã tải xong
40        google.maps.event.addDomListener(window, 'load', initMap);
41      </script>
42    </body>
43  </html>
44
```

- Đây là code để load các điểm trong database lên
- ở đây ta thay 120.0 là red còn 0.0 là blue

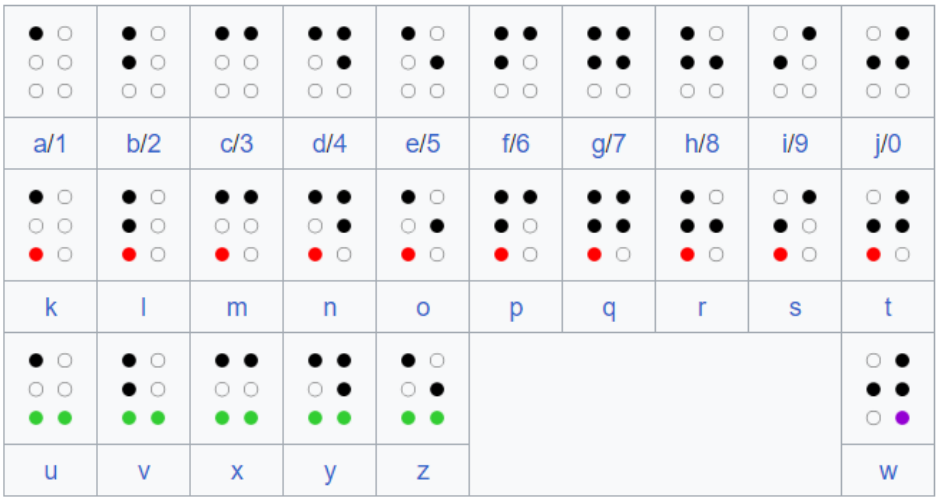
- Cần thay API trong đoạn code của mình bằng API ta lấy được từ google development console



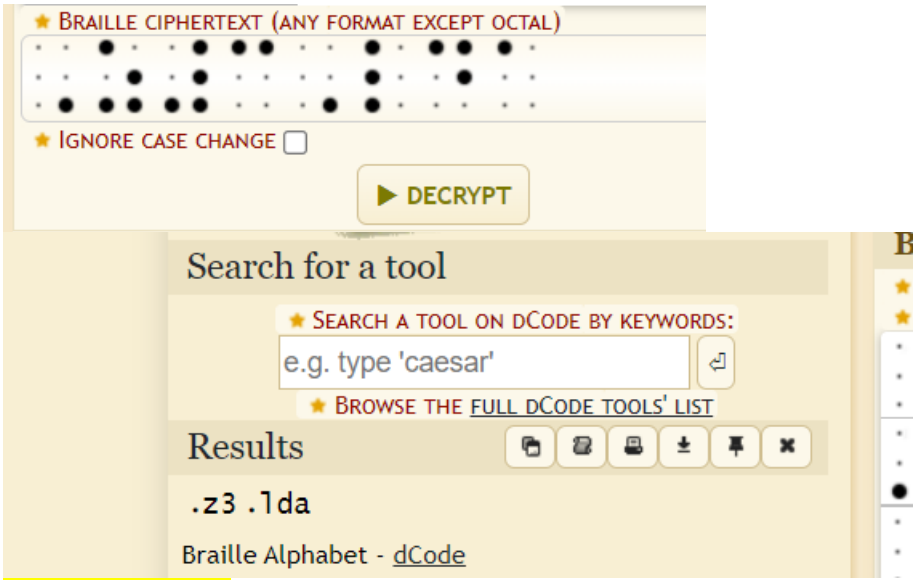
- Ở đây em dùng trang web codepen.io để viết code html để chạy
- Khi ta load các điểm lên thì ta được các điểm như trên
- Tới đây ta vẫn chưa sử dụng tới dữ kiện cột ngày trong database
- Khi lọc các điểm theo ngày thì ngày "02/08/2019" cho ta các điểm khá là thú vị

- Đây có vẻ như là "Braille"

Derivation (colored dots) of the 26 braille letters of the basic Latin alphabet from the 10 numeric digits (black dots)



- Ta có thể coi điểm màu đỏ là những chấm đen còn màu xanh là chấm trắng
- Sử dụng link

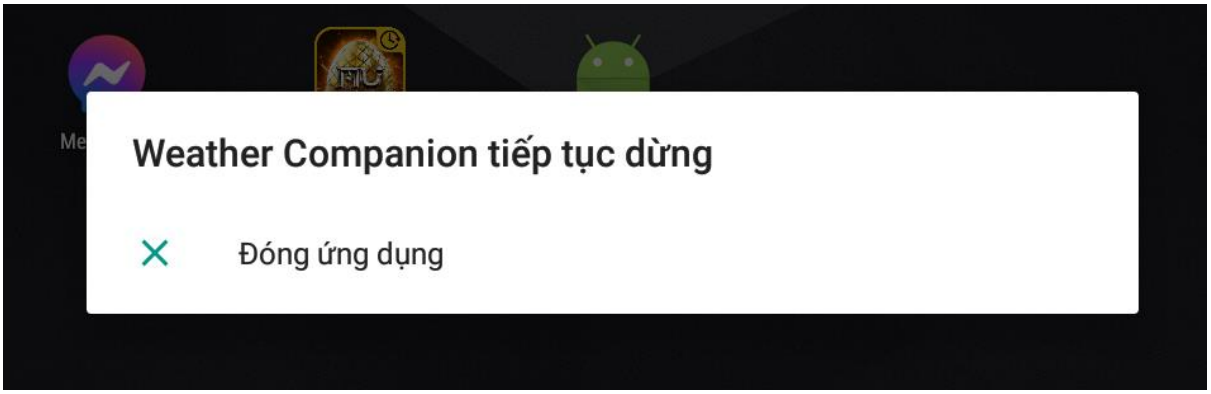


➔ Flag: Z3LDA

4. 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 Android đang chạy của ứng dụng. Sử dụng một số công cụ decompile apk như Jadx để phân tích code ứng dụng. Flag có định dạng CTF{...}

Đáp án:



- Khi cài đặt ứng dụng thì nó bị crash
- Ta xem thử file AndroidManifest.xml của nó xem sao

```
<manifest android:versionCode="1" android:versionName="1.0" package="com.example.myapplica
  <uses-sdk android:minSdkVersion="26" android:targetSdkVersion="27"/>
```

- Sau khi decompile thì ta thấy phiên bản sdk mà hệ thống tương thích là 26 hoặc 27 nghĩa là android 8 và andoid 8.1

```
ne" android:name="com.example.myapplication.MainActivity">
```

- Chương trình có activity là com.example.myapplication.MainActivity

```
a.java
1 package com.example.myapplication;
2
3 import android.os.Bundle;
4 import android.support.v7.app.c;
5 import android.support.v7.widget.Toolbar;
6 import java.io.IOException;
7 /* loaded from: classes.dex */
8 public class MainActivity extends c {
9     @Override // android.support.v7.app.c, android.support.v4.app.h, android.support.v4.app.z, android.app.Activity
10    public void onCreate(Bundle bundle) {
11        super.onCreate(bundle);
12        setContentView(R.layout.activity_main);
13        c().a((Toolbar) findViewById(R.id.toolbar));
14        try {
15            new a(this, getApplicationContext()).execute(new Void[0]);
16        } catch (IOException e) {
17            e.printStackTrace();
18        }
19    }
20 }
```

- Đây là MainActivity

- Đọc sơ qua thì đoạn mã cố gắng thực thi 1 tác vụ bằng cách sử dụng lớp “a”, truyền hoạt động “this” và ngữ cảnh “getApplicationContext()” làm đối số
- Lớp này có thể là 1 lớp thực hiện tùy chỉnh của 1 tác vụ bất đồng bộ AsyncTask để thực hiện các hoạt động nền
- Sau đó dòng execute để khởi động thực thi tác vụ
- Từ đây ta xem code của a.class

```
{
    private final MainActivity a;
    private final Context b;

    public a(final MainActivity a, final Context b) {
        super();
        this.a = a;
        this.b = b;
    }

    private String a() {
        final String s = "";
        final k j = k.j();
        if (((i)j).h == null) {
            ((i)j).h = ((i)j).g.a((i)j);
        }
        final g g = (g)((i)j).h;
        final String s2 = null;
        String string = s;
        String s3 = s2;
        try {
            string = s;
            s3 = s2;
            final Utils utils = new Utils(this.b);
            string = s;
            s3 = s2;
            final String a = Utils.a("LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCk1JSUV2QUlCQURBTkJna3Foa2lHOXcwQkFRUUZBQVNDQktZd2dnU2lBZ0VBQW9JQkFRQ2J
            string = s;
            s3 = s2;
            string = s;
            s3 = s2;
            final String str = new String(utils.dks());
            string = s;
            s3 = s2;
            string = s;
            s3 = s2;
            final String str2 = new String(utils.ss("CnoM+J76ft/1blqCH4j4+/Oh/EHG+iWbhzfijs5a7XHKINC8ach6YLbhAyUm9+mI\nGuGVAklGeeN5InzSubRcrL8BSo
            string = s;
            s3 = s2;
            final String a2 = Utils.a("M5VXSRDXNBVGCZDWOBUWCWJYG52FAK3BOIYU2VSYORSVGMLRINEW2YSHMZREO6KLJV3SWNLPKEXWY5KINRMHGOLWKFTU053ILFGVCWAKNJ
            string = s;
            s3 = s2;
```

- đoạn mã trên được sử dụng để xây dựng một địa chỉ URL. Trong đó, đối tượng a kiểu StringBuilder được sử dụng để ghép nối một chuỗi JSON. Sau đó, chuỗi JSON này được kết hợp với các giá trị a2 và a3 ở phía trên để tạo thành một địa chỉ URL hoàn chỉnh.

```
        final InputStreamReader in = new InputStreamReader(httpURLConnection.getInputStream());
        string = s;
        s3 = string13;
        final BufferedReader bufferedReader = new BufferedReader(in);
        string = s;
        s3 = string13;
        string = s;
        s3 = string13;
        final StringBuffer sb12 = new StringBuffer();
        string = s;
        String line;
        while (true) {
            s3 = string13;
            line = bufferedReader.readLine();
            if (line == null) {
                break;
            }
            string = line;
            s3 = string13;
            string = line;
            s3 = string13;
            final StringBuilder sb13 = new StringBuilder();
            string = line;
            s3 = string13;
            sb13.append(line);
            string = line;
            s3 = string13;
            sb13.append("\n");
            string = line;
            s3 = string13;
            sb12.append(sb13.toString());
            string = line;
        }
        string = line;
        s3 = string13;
        string = sb12.toString();
        s3 = string13;
    }
    catch (final IOException ex) {
        ex.printStackTrace();
```

- Tạo một đối tượng InputStreamReader từ httpURLConnection.getInputStream(). Đối tượng này được sử dụng để đọc dữ liệu từ kết nối mạng.
- Khởi tạo một đối tượng BufferedReader từ InputStreamReader để đọc dữ liệu theo từng dòng.
- Tạo một đối tượng StringBuffer (được khởi tạo như final StringBuffer sb12 = new StringBuffer()) để lưu trữ dữ liệu đọc được từ kết nối mạng.
- Sử dụng vòng lặp while để đọc từng dòng dữ liệu từ bufferedReader.readLine() và lưu vào sb12.
- Khi đọc xong tất cả các dòng, chuyển nội dung của sb12 thành một chuỗi bằng cách sử dụng sb12.toString().
- Trong trường hợp xảy ra lỗi IOException, ngoại lệ được in ra màn hình với ex.printStackTrace().
- Nếu s3 (biến không được định nghĩa trong đoạn mã hiện tại) không null, in ra màn hình thông báo "Background URL in progress" bằng Log.d("Background url", "Background URL in progress").
- Trả về chuỗi kết quả là string.

  ➔ Đoạn mã trên thực hiện việc kết nối mạng, đọc dữ liệu từ kết nối và trả về chuỗi kết quả để sử dụng trong quá trình xử lý tiếp theo

```
    protected final void onPostExecute(final Object o) {
        final String s = (String)o;
        Log.d("Complete reponse", s);
        try {
            final JSONObject jsonObject = new JSONObject(s);
            final String string = ((JSONObject)jsonObject.get("display_location")).getString("city");
            final JSONObject jsonObject2 = (JSONObject)jsonObject.get("current_weather");
            final String string2 = jsonObject2.getString("temperature");
            final String string3 = jsonObject2.getString("precipitation");
            final String string4 = jsonObject2.getString("humidity");
            final String string5 = jsonObject2.getString("wind");
            ((TextView)this.a.findViewById(2131230765)).setText((CharSequence)string);
            ((TextView)this.a.findViewById(2131230837)).setText((CharSequence)string3);
            ((TextView)this.a.findViewById(2131230802)).setText((CharSequence)string4);
            ((TextView)this.a.findViewById(2131230912)).setText((CharSequence)string5);
            ((TextView)this.a.findViewById(2131230886)).setText((CharSequence)string2);
        }
        catch (final JSONException ex) {
            ex.printStackTrace();
        }
    }
}
```

- Sau đó lấy dữ liệu phản hồi và hiển thị lên ứng dụng
- Vậy JSON ở đây là gì?
- Ta xem trong file Utils

```
public static String a(String str, int i) {
    byte[] bytes;
    if (i == 0) {
        bytes = Base64.getDecoder().decode(str);
    } else {
        org.apache.a.a.a.a aVar = new org.apache.a.a.a.a();
```

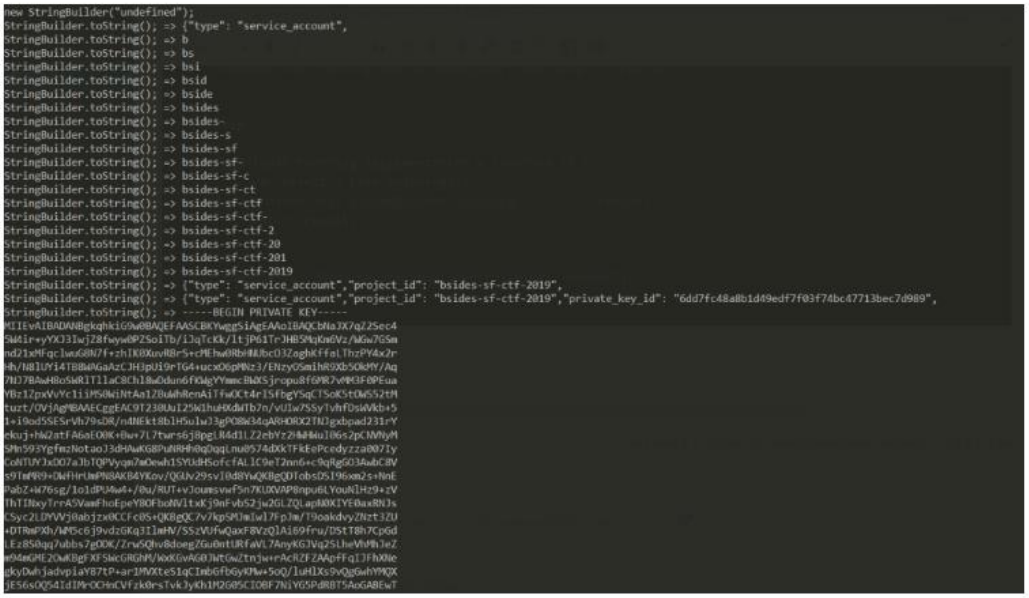
```
public static String a(int[] iArr) {
    String str = "";
    for (int i = 0; i < iArr.length; i++) {
        str = str + ((char) iArr[i]);
    }
    return str;
}
```

```
/* JADX INFO: Access modifiers changed from: package-private */
public native byte[] dks();

/* JADX INFO: Access modifiers changed from: package-private */
public native long gci();

/* JADX INFO: Access modifiers changed from: package-private */
public native byte[] ss(String str, int i);
}
```

- Có thể thấy rằng chúng đang giải mã Base64, int thành char, BigInteger thành Hex và các chức năng của ba lớp gốc
- Để rõ hơn ta sẽ sử dụng frida để hook

```
Java.perform(function(){

    // Step - 1

    var array_list = Java.use("java.util.ArrayList");
    var ApiClient = Java.use('com.android.org.conscrypt.TrustManagerImpl');

    ApiClient.checkTrustedRecursive.implementation = function(a1, a2, a3, a4, a5, a6) {
        var k = array_list.$new();
        return k;
    }


    // Step - 2

    console.log("Hooking Java");

    const StringBuilder = Java.use('java.lang.StringBuilder');

    StringBuilder.$init.overload('java.lang.String').implementation = function (arg) {
            var partial = "";
            var result = this.$init(arg);
            console.log('new StringBuilder("' + result + '");')
            return result;
    }

    console.log("Hooking new StringBuilder(java.lang.String)");


    // Step - 3

    StringBuilder.toString.implementation = function () {
            var result = this.toString();
            console.log('StringBuilder.toString(); => ' + result)
            return result;
    }

    console.log("Hooking StringBuilder.toString() hooked");

}, 0);
```

- Đoạn mã trên mô tả các bước sau:
  + Vô hiệu hóa cơ chế SELinux trên thiết bị.
  + Chạy Frida server trên thiết bị.
  + Viết một script hook bằng Frida, sử dụng đoạn mã mẫu từ liên kết tham khảo được cung cấp.
  + Phần đầu tiên của script hook đóng vai trò là một phương pháp để bypass SSL Pinning trong quá trình hook ứng dụng Android. Tuy nhiên, trong quá trình kiểm tra thực tế, đã xác định rằng việc xóa phần này vẫn cho phép hook thành công. Liên kết tham khảo cung cấp các phương pháp bypass SSL Pinning.
  + Phần thứ hai của script hook tập trung vào việc hook phương thức toString() trong lớp String của Java. Điều này cho phép trình bày hoàn hảo khả năng hook tất cả mọi thứ trong Frida. Lý do hook phương thức này là vì có một phép toán toString() trên URL trong đoạn mã trước đó, giúp hiển thị nội dung đã được mã hóa và xử lý bằng các phương pháp khác nhau.
  + Phần thứ ba của script hook in ra kết quả của phương thức toString() mà đã được hook.



- Chuỗi có định dạng JSON này là key được sử dụng để ủy quyền trong google cloud storage
- Và file JSON sẽ có định dạng như sau, khi đã sắp xếp lại

```
{
    "type": "service_account",
    "project_id": "bsides-sf-ctf-2019",
    "private_key_id": "6dd7fc48a8b1d49edf7f03f74bc47713bec7d989",
    "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCbNaJX7qZ2Sec4\nSW4ir+yYXJ3IwjZ8fwyw0PZSoiTb/iJqTcKk/ltjP61TrJHB5MqKm6Vz/\nWGw7GSm\nnd21xMFqclwuG8N7f+zhIK0XuvRBrS+cMEhw0RbHNUbcO3ZaghKffaLThzPY4x2r\nHh/N8lUYi4TB8WAGaAzCJH3pUi9rTG4+ucxO6pMNz3/ENzyOSmihR9Xb5OkMY/\nAq\nN7N7BAwH8oSWRlTllaC8Chl8wDdun6fKWgYYmmcBWXSjropu8f6MR7vM3F0PEua\nYBz1ZpxVvYc1iiMS0WlNtAa1ZBuWhRenAiTfwOCt4rISfbgYSqCTSoKStOWS52tM\ntuzt/OVjAgMBAAECggEAC9T230UuI25W1huHXdwTb7n/vUIw7SSyTvhfDsWVkb+5\n1+i9od5SESrVh79sDR/n4NEkt8blH5ulwJ3gPO8W34qARHORXZTN3gxbpad231rY\nekuj+hW2atFA6aEO0K+Bw+7L7twrs6j8pgLR4d1LZ2ebYz2HWHWuI06s2pCNVNyM\nSMn593YgfmzNotaoJ3dHAwKG8PuNRHh0qDqqLnu0S74dXkTFkEePcedyzza007Iy\nCoNTUYJxDO7aJbTQPVyqm7mOewh1SYUdHSofcfAL1C9eT2nn6+c9qRgGO3AwbC8V\ns9Im4R9+DWfHrUmPN8AKB4YKov/QGUv29svI0d8YwQKBgQDTobsDSI96xm2s+NnE\nPabZ+W76sg/1o1dPUWw+/0u/RUT+vJoumswwf5n7KUXVAP8npu6LYouNlHz9+zV\nThTINxyTrrA5VamFhoEpeY8OFboNVltxKj9nFvbS2jw2GLZQLapN0XIYE0axRNJs\nCSyc2LDYVVj0abjzx0CCFc0S+QK8gQC7v7kp5MJmIwl7FpJm/\nT9oakdvyZNzt3ZU\n+DTRmPXh/WM5c6j9vdzGKq3IlmHV/SSzVUfwQaxF8VzQlAi69fru/DStT8h7CpGd\nlEz8S0qq7ubbs7gOOK/ZrwSQhv8doegZGu0ntURfaVL7AnyKGJVq2SLheVhMhJeZ\nm94mGME2OwKBgFXFSWcGRGhM/\nWxKGvAG0JWtGwZtnjw+rAcRZFZAApfFqIJFhXNe\ngkyDwhjadvpiaY87tP+ar1MVXteS1qCImbGfbGyKMw+5oQ/\nluHlXs9vQgGwhYMQX\njES6sOQS4IdIMrOCHnCVfzk0rsTvkJyKh1M2G05CIOBF7NiYG5PdRBT5AoGABEwT\nFNrReDz9DpApsanCNcWY9PoMIe3lC3TS4Kk7l3yRNNNs3sHlt7NqXtjyTE+K83/\nU\nMa+PHdq0YSHCQWU35RhorD7TO922D37gFDY080ychBLM96VasQTMefJdDHSUN17l\nZrJDaluixpP7/b0qTlPB9J8uYjH2tlFW+FBAu9kCgYBch8VvyBzlGay5r07joRju\ncdnVfrGAUdZRLxA5hrIaKvKFy28CaaKV/\nlZNI9NmdZntj6aIQ0rzUWidOQsmj2o4\n7VcArl5UNurhx4uhg2GClcUIHY5YdExHfEujBu6uMQdZjBpX87uoaggC7jwJanQC\nlApTPc3478g/mho9S5eDMw==\n-----END PRIVATE KEY-----\n",
    "client_email": "weather-companion-service-acco@bsides-sf-ctf-2019.iam.gserviceaccount.com",
    "client_id": "116037946827001874660",
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
    "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/weather-companion-service-acco%40bsides-sf-ctf-2019.iam.gserviceaccount.com"
}
```

- Sau khi tìm hiểu cách cài đặt: Link
- thì ta sử dụng lệnh như sau

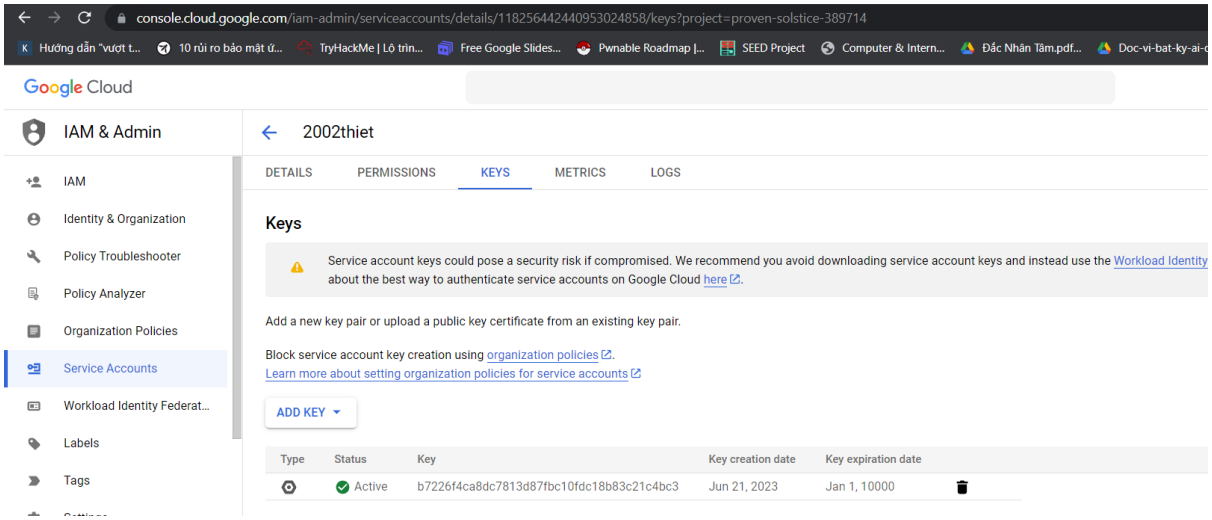
```
(base) thiet@thiet-virtual-machine:~/Desktop$ gcloud auth activate-service-accou
nt --key-file=key2.json
```

- Nhưng kết quả trả về

```
(base) thiet@thiet-virtual-machine:~/Desktop$ gcloud auth activate-service-accou
nt --key-file=key2.json
ERROR: (gcloud.auth.activate-service-account) There was a problem refreshing you
r current auth tokens: ('invalid_grant: Invalid grant: account not found', {'err
or': 'invalid_grant', 'error_description': 'Invalid grant: account not found'})
Please run:

  $ gcloud auth login

to obtain new credentials.

If you have already logged in with a different account, run:

  $ gcloud config set account ACCOUNT

to select an already authenticated account to use.
(base) thiet@thiet-virtual-machine:~/Desktop$
```

- Có vẻ như tài khoản này đã bị xóa và không còn tồn tại nữa

- Em sau đó đã tự tạo tài khoản để thử



- Tại đây đã tạo thành công và có key sau đó nó lưu dưới dạng file JSON và tải về máy


```
"type": "service_account",
"project_id": "proven-solstice-389714",
"private_key_id": "b7226f4ca8dc7813d87fbc10fdc18b83c21c4bc3",
"private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCg+soNPXFL0/
4\nJp7Tnr2aSnnCcSS1zfKLeLDQNsHZ3NzJki25efGVkgRGu5ERTtjAh8JDEgnQ17jo\nvhMiZ2Rq/TQ4YYn/
>vHoL0TbUfCvsfDAj41bf7kI0VJLq4R6K6Qmdf4Tp21Ihne\nGgOjXcN5z2qoi+gBjORQ9feuEffakUlZk1sPuIjVnejPHI0AQNwdJtBGhEubi+dL\nXebZz7p+k+ZtBHruhtayodej36INUFvLEbClr6KTOeLmK+CI
/zJyjUh\nCmgp8K2buHL00AJeVqJnXbCEcwo5r5cRt6aucAyk1NTixPcQdkw7XHNN7ru2Dqie\nBg9wTmrpg919WwDJ9DACT70BO7MT27f5XpdqM72iUwmVshNUzD4zb9GytGvI3hZA\nFIBtXp8IzS3xWB14xZaYm
=+1RssHrg/g6Co/Nb8fSL9cIvcdjiXANHQm9IA4sUmIu38l+yfcKa32XW\nUkkpc+7MVBhYgjt/Yra42O3jJFJ6zw/
j2b6c1x6swKBgQDgsDTlY7TAKH8BN2En\n82TVmVkC7E9IvoWl5DW6EZVYenvimeeOm288LaPeVEwAhILBx3MqjjBUqPY9CeCm\nJ7NgOXPnDtKapOUc4Q+9JK2qOW0lpBKEEM7ekcG9nISE3HeWmOmrWPCWk23JhR
iLGstfEMhc\nxSGhxmjkQDEJXeQa8aqZhu2C10N+QAQpRq62nHI7xsbFRVOWEodbXJ4zgeYGC0Xk\nZxa2iFbruzUsQWRC+NTp3yIgWgh0zNudOBThju6tw7yewgWZI8ATqfxxOMpWpCAe\nh1RsX4SifwKBgAkencc
J\n0YX7xD7t5PfQaQ48biw4ATYudXAcyoVz7w/kxROHwnyMXNBDQkRHDNVv50Iz2pyJ\nohvehqKPi8FowWHXawSNY5Z4/WfEgx++dkmgVlJFhlS6HdS0GTnJVY3pAoGBAK86\nKKp0QaSG3RXY0g66/
rQKMvNKYcFbabT3zHDcOrAORkNzqO/RpGuyrM6HY08c71L\nfheQl8JsQfKuLREvWon6jln/PFeB5aA0rQpubzx/
ılxf+xL9n4JY30tBu8v2JD3\nx1tHRthULJwwGVEZSqR+vmMdMpJb8yO3fvyatQjZAoGBAMjlXf5AbnuzIM8LApTl\nC5gHtLinI8hVT9Hw/
t0PcoNrw8UrFseRf9r4PCYg02p3mMIeJsXEwuGzxYfxAC\nkxGxDLo6czj6Wd3POT1FcFwyV9PJ5qjZ7yRd3izGN8EZBupOc+MHjJze2ZfYH3Yf\nogOPV642Yg+DbNV+yFTVDDHR\n-----END PRIVATE KEY--
"client_email": "id-002thiet@proven-solstice-389714.iam.gserviceaccount.com",
"client_id": "118256442440953024858",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",
"token_uri": "https://oauth2.googleapis.com/token",
"auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
"client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/id-002thiet%40proven-solstice-389714.iam.gserviceaccount.com",
"universe_domain": "googleapis.com"
```


- Đây là file JSON của ta

```
(base) thiet@thiet-virtual-machine:~/Desktop$ gcloud auth activate-service-accou
nt --key-file=key.json
Activated service account credentials for: [id-002thiet@proven-solstice-389714.i
am.gserviceaccount.com]
```

- Tại đây ta đã active được tài khoản
- Nhưng ở đây ta cần tài khoản của challenge CTF này mà hiện không còn nên cũng không thể làm gì hơn
- Đây là các bước nếu có tài khoản được cấp

```
$  gcloud auth activate-service-account --key-file=key.json
Activated service account credentials for: [weather-companion-service-acco@bsides-sf-ctf-2019.iam.gserviceaccount

$ gsutil ls -p bsides-sf-ctf-2019 gs://weather-companion
gs://weather-companion/flag.txt
gs://weather-companion/weather.json
```

```
$ gsutil -m cp -r -p bsides-sf-ctf-2019 gs://weather-companion ./
```

➔ Flag: CTF{buck3t_s3at5}

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