Contents

I. Tools/Software used:

1. MOBSF:

Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis. MobSF supports mobile app binaries (APK, XAPK, IPA & APPX) along with zipped source code and provides REST APIs for seamless integration with your CI/CD or DevSecOps pipeline. The Dynamic Analyzer helps you to perform runtime security assessment and interactive instrumented testing

2. Jadx-Gui:

it's a Dex to Java decompiler.

Command line and GUI tools for producing Java source code from Android Dex and Apk files.

3. Ghidra:

Ghidra is a free and open-source reverse engineering tool developed by the National Security Agency of the United States. The binaries were released at RSA Conference in March 2019; the sources were published one month later on GitHub. Ghidra is seen by many security researchers as a competitor to IDA Pro.

4. Androwarn:

Androwarn is a tool whose main aim is to detect and warn the user about potential malicious behaviors developed by an Android application.

5. ApkTool:

A tool for reverse engineering 3rd party, closed, binary Android apps. It can decode resources to nearly original form and rebuild them after making some modifications. It also makes working with an app easier because of the project like file structure and automation of some repetitive tasks like building apk, etc. It is **NOT** intended for piracy and other non-legal uses. It could be used for localizing, adding some features or support for custom platforms, analyzing applications and much more.

6. Genymotion:

Genymotion Desktop is an Android emulator which includes a complete set of sensors and features in order to interact with a virtual Android environment. With Genymotion Desktop, you can test your Android applications on a wide range of virtual devices for development, test and demonstration purposes.

7. ApkSigner:

The apksigner tool, available in revision 24.0.3 and higher of the Android SDK Build Tools, lets you sign APKs and confirm that an APK's signature will be verified successfully on all versions of the Android platform supported by that APK

8. AAPT:

aapt stands for Android Asset Packaging Tool and is included in the tools/ directory of the SDK. This tool allows you to view, create, and update Zip-compatible archives (zip, jar, apk). It can also compile resources into binary assets.

9. Jarsigner:

Signs and verifies Java Archive (JAR) files.

10. Adb:

The Android Debug Bridge is a programming tool used for the debugging of Android-based devices. The daemon on the Android device connects with the server on the host PC over USB or TCP, which connects to the client that is used by the end-user over TCP.

11. Burp suite:

Burp Suite is an integrated platform/graphical tool for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.

12. OWASP:

The Open Web Application Security Project is an online community that produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security. The OWASP provides free and open resources. It is led by a non-profit called The OWASP Foundation

13. Mitmproxy:

mitmproxy is your Swiss-army knife for debugging, testing, privacy measurements, and penetration testing. It can be used to intercept, inspect, modify and replay web traffic such as HTTP/1, HTTP/2, Web Sockets, or any other SSL/TLS-protected protocols

14. Frida:

Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers.

15. Radare2:

Radare2 is a complete framework for reverse-engineering and analyzing binaries; composed of a set of small utilities that can be used together or independently from the command line

II. Application to analyze:

FacilePay

FacilePay - Payment for Stripe makes it easy for tradespeople and other small business owners to collect credit/debit card payment right on the spot with their smartphone!

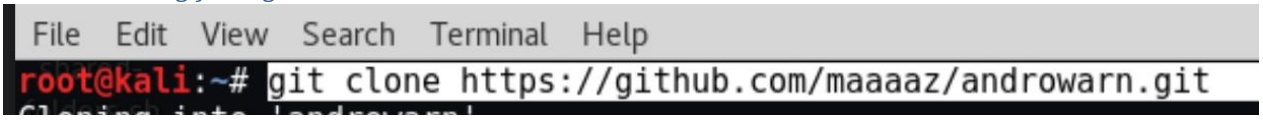
Website that contains information about the app:

https://www.facilepay.ca/

III. Static analysis:

1. Installation of androwarn:

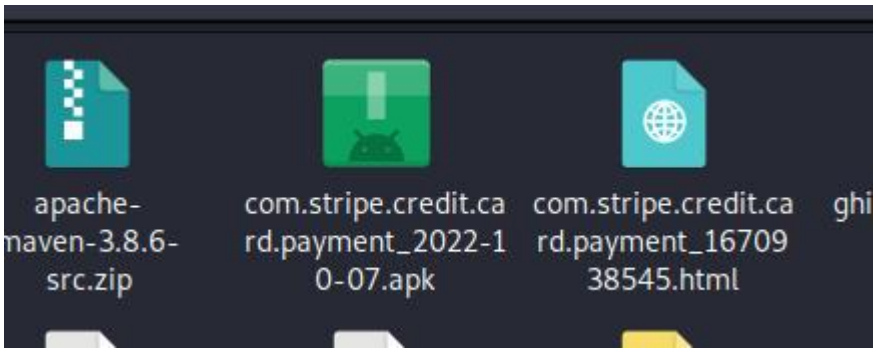
i. Cloning from git:



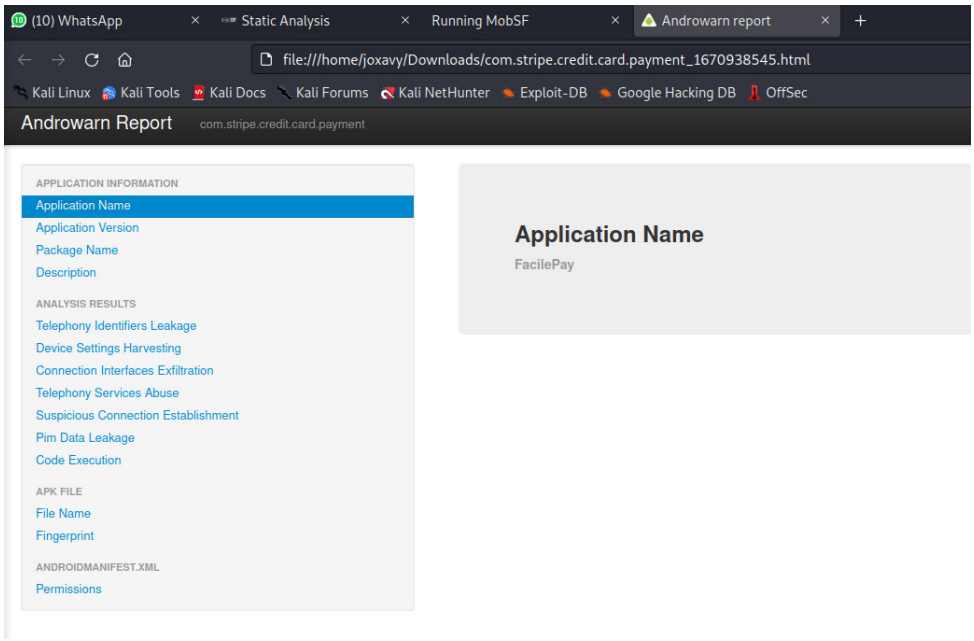
ii. Using it to analyze the apk:

```
┌──(joxavy㉿kali)-[~/Downloads]
└─$ androwarn -i com.stripe.credit.card.payment_2022-10-07.apk
[+] Androwarn version 1.6

[+] Loading the APK file ...
Requested API level 30 is larger than maximum we have, returning API level 28 instead.
[+] Analysis successfully completed and HTML file report available '/home/joxavy/Downloads/com.stripe.credit.card
.payment_1670938545.html'

┌──(joxavy㉿kali)-[~/Downloads]
└─$
```

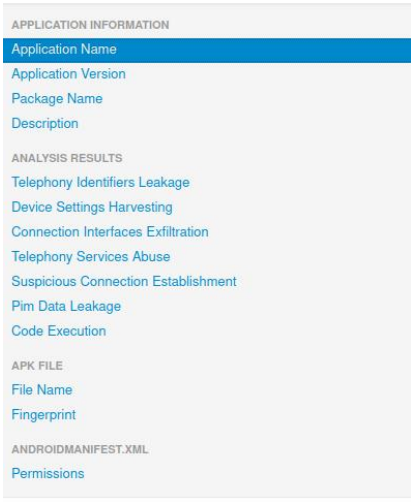
The output of the analysis is saved inside an HTML file:



when clicked it displays the following result:



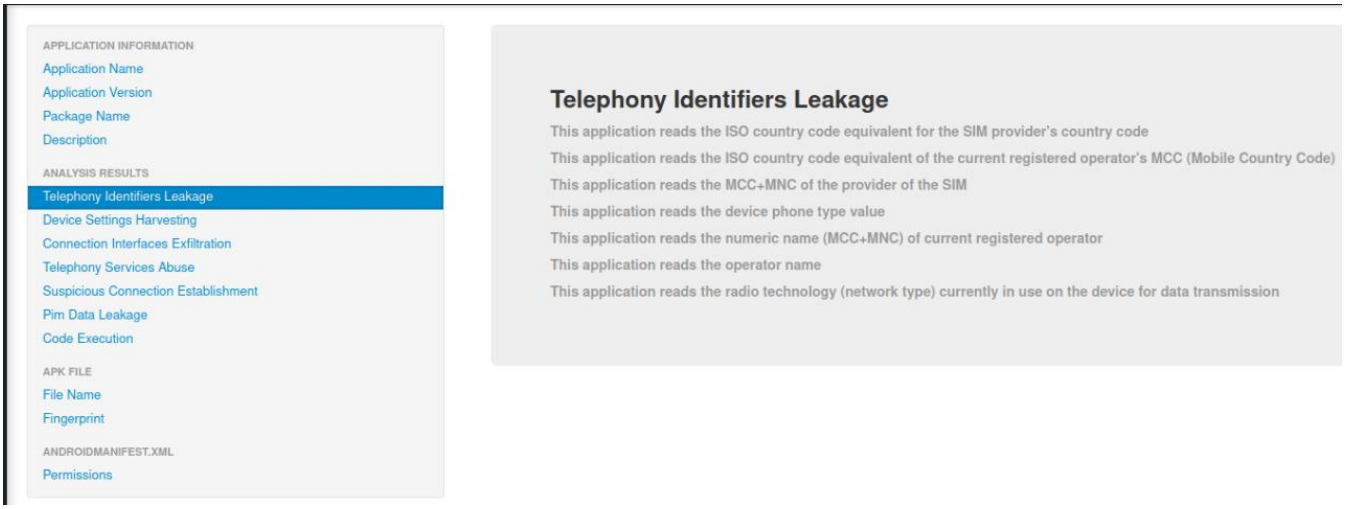
As we can see, this tool displays multiple information about the apk given, including the following details:



We will now check each result one by one.

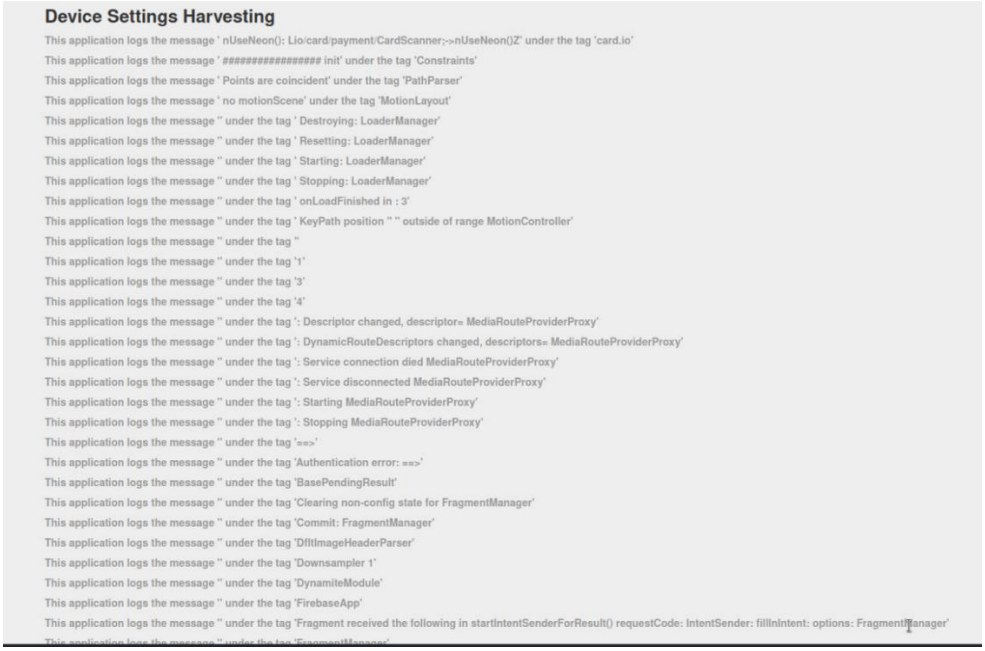
Telephony identifiers leakage:

Which are physical slots indexes or IDs: unique indexes referring to a physical SIM slot



This application leaks the ISO country code, operator name, etc.

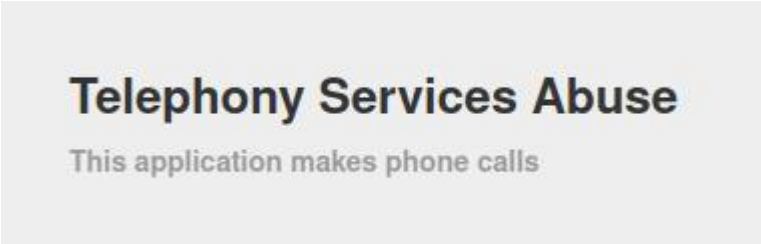
Device settings harvesting:



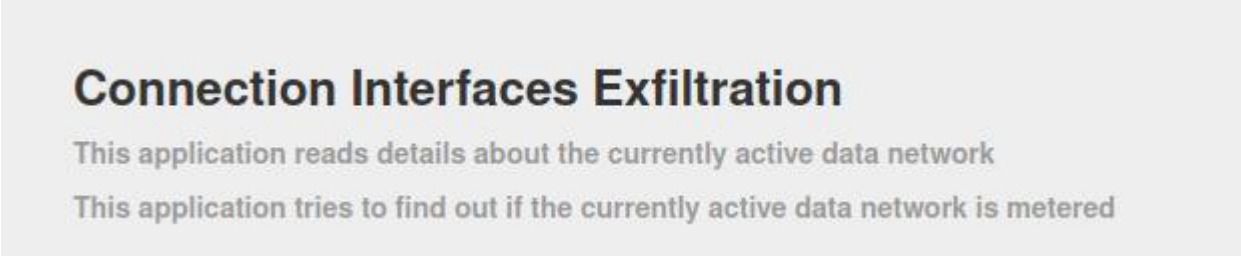
It logs a lot of info.

Telephony services Abuse:

It can place phone calls.



Connection Interfaces Exfiltration:



Suspicious connection establishment:

Suspicious Connection Establishment

This application opens a Socket and connects it to the remote address ' returned no addresses for ; port is out of range' on the 'N/A' port

This application opens a Socket and connects it to the remote address '' on the 'N/A' port

This application opens a Socket and connects it to the remote address 'Ljava/lang/StringBuilder;->toString()Ljava/lang/String;' on the 'N/A' port

This application opens a Socket and connects it to the remote address 'Ljava/net/Proxy;->type()Ljava/net/Proxy$Type;' on the 'N/A' port

This application opens a Socket and connects it to the remote address 'timeout' on the 'N/A' port

Pim data leakage:

Pim Data Leakage

This application accesses data stored in the clipboard

Code execution:

Code Execution

This application loads a native library

This application loads a native library: 'androidndkgif'

Fingerprint:

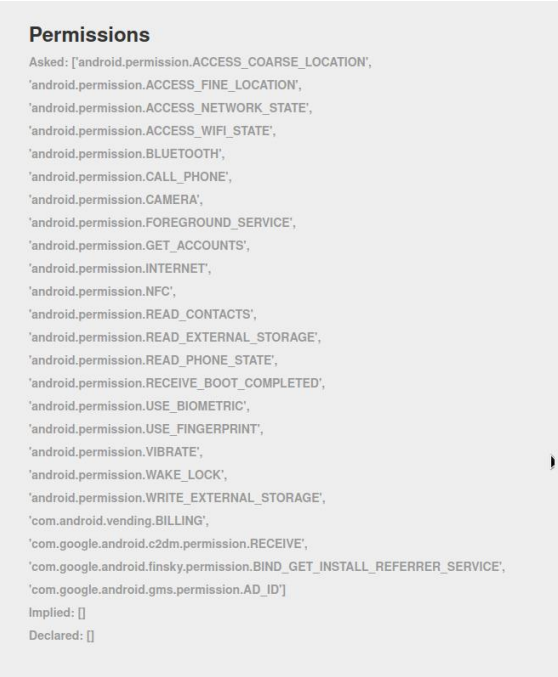
Fingerprint

MD5: 6eea256a97972a541f42ece2b727c51d

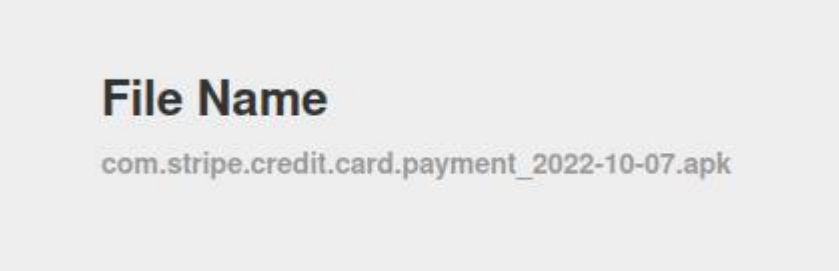
SHA-1: 54ec3a59e2cd5f46f042bc8d1ced47beaa124291

SHA-256: f7d57fde88ad404a1480d21c519c37d2d218e41f255066cf1293e7f3b6c4c6da

Permissions:



Filename:



All of this can be checked by clicking on the following file:

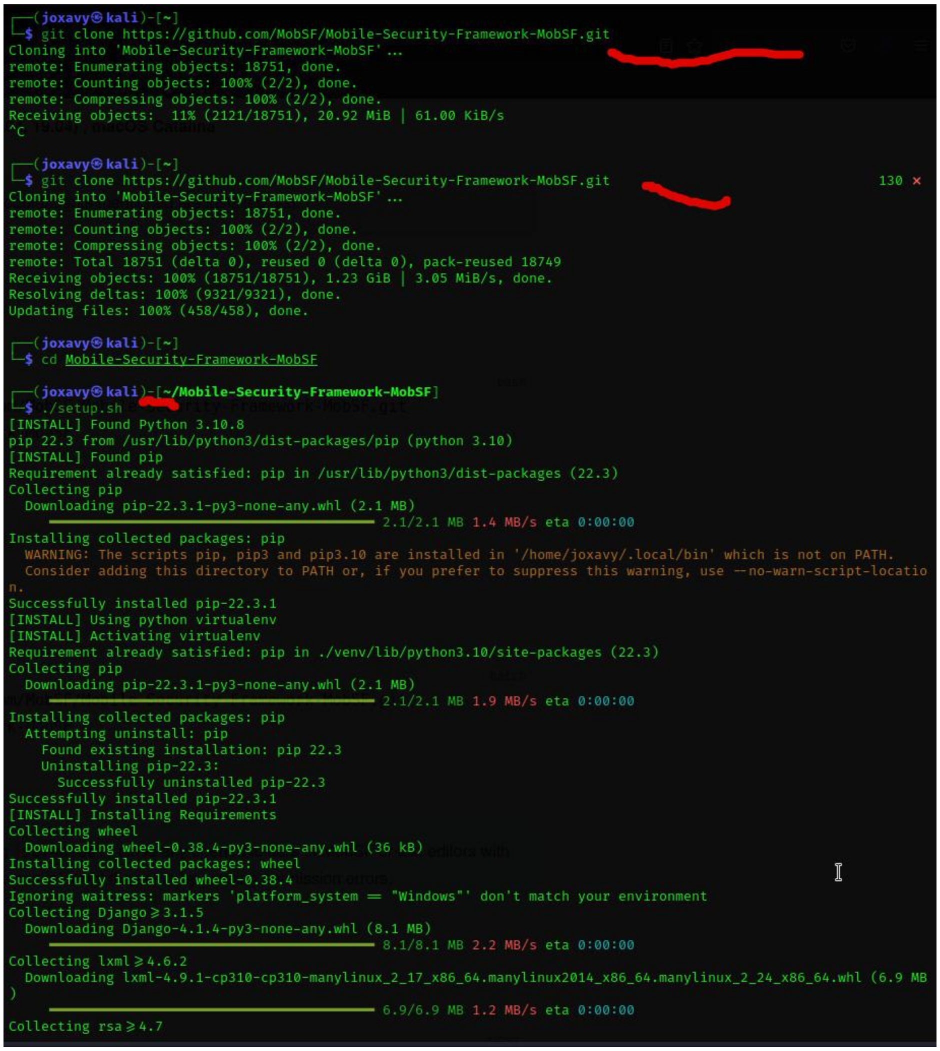
com.stripe.credit.card.payment_1670938545.html

2. Installation of MOBSF:

a) Getting started:

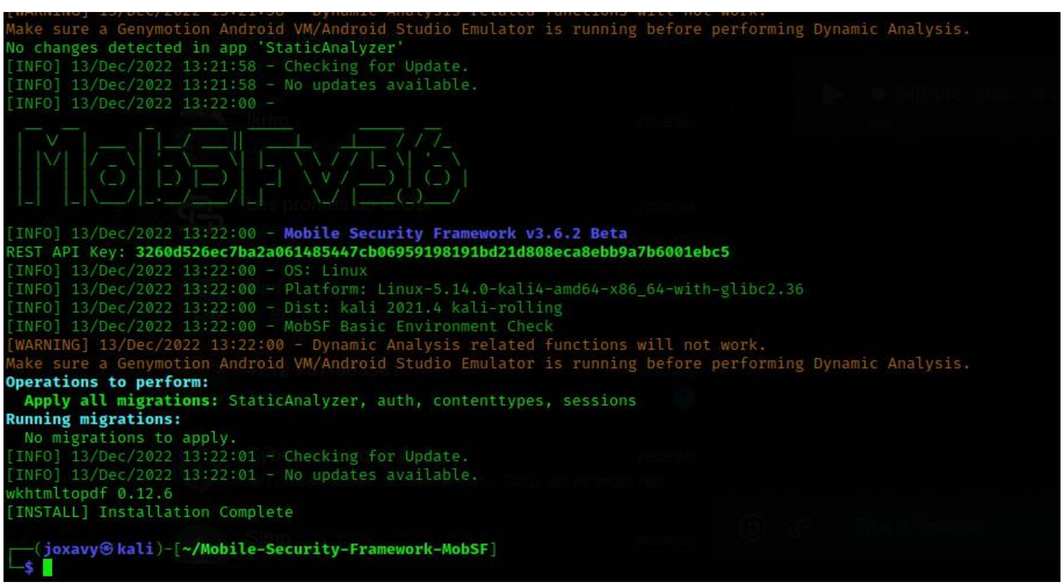
Cloning it from GitHub

Starting it up with ./setup.sh



```
┌──(joxavy㉿kali)-[~]
└─$ git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 18751, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
Receiving objects:  11% (2121/18751), 20.92 MiB | 61.00 KiB/s
^C

┌──(joxavy㉿kali)-[~]
└─$ git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git        130 ✕
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 18751, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 18751 (delta 0), reused 0 (delta 0), pack-reused 18749
Receiving objects: 100% (18751/18751), 1.23 GiB | 3.05 MiB/s, done.
Resolving deltas: 100% (9321/9321), done.
Updating files: 100% (458/458), done.

┌──(joxavy㉿kali)-[~]
└─$ cd Mobile-Security-Framework-MobSF

┌──(joxavy㉿kali)-[~/Mobile-Security-Framework-MobSF]
└─$ ./setup.sh
[INSTALL] Found Python 3.10.8
pip 22.3 from /usr/lib/python3/dist-packages/pip (python 3.10)
[INSTALL] Found pip
Requirement already satisfied: pip in /usr/lib/python3/dist-packages (22.3)
Collecting pip
  Downloading pip-22.3.1-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 1.4 MB/s eta 0:00:00
Installing collected packages: pip
  WARNING: The scripts pip, pip3 and pip3.10 are installed in '/home/joxavy/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-locatio
n.
Successfully installed pip-22.3.1
[INSTALL] Using python virtualenv
[INSTALL] Activating virtualenv
Requirement already satisfied: pip in ./venv/lib/python3.10/site-packages (22.3)
Collecting pip
  Downloading pip-22.3.1-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 1.9 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3
    Uninstalling pip-22.3:
      Successfully uninstalled pip-22.3
Successfully installed pip-22.3.1
[INSTALL] Installing Requirements
Collecting wheel
  Downloading wheel-0.38.4-py3-none-any.whl (36 kB)
Installing collected packages: wheel
Successfully installed wheel-0.38.4
Ignoring waitress: markers 'platform_system == "Windows"' don't match your environment
Collecting Django>=3.1.5
  Downloading Django-4.1.4-py3-none-any.whl (8.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.1/8.1 MB 2.2 MB/s eta 0:00:00
Collecting lxml>=4.6.2
  Downloading lxml-4.9.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl (6.9 MB
)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 1.2 MB/s eta 0:00:00
Collecting rsa>=4.7
```

Finished installation:

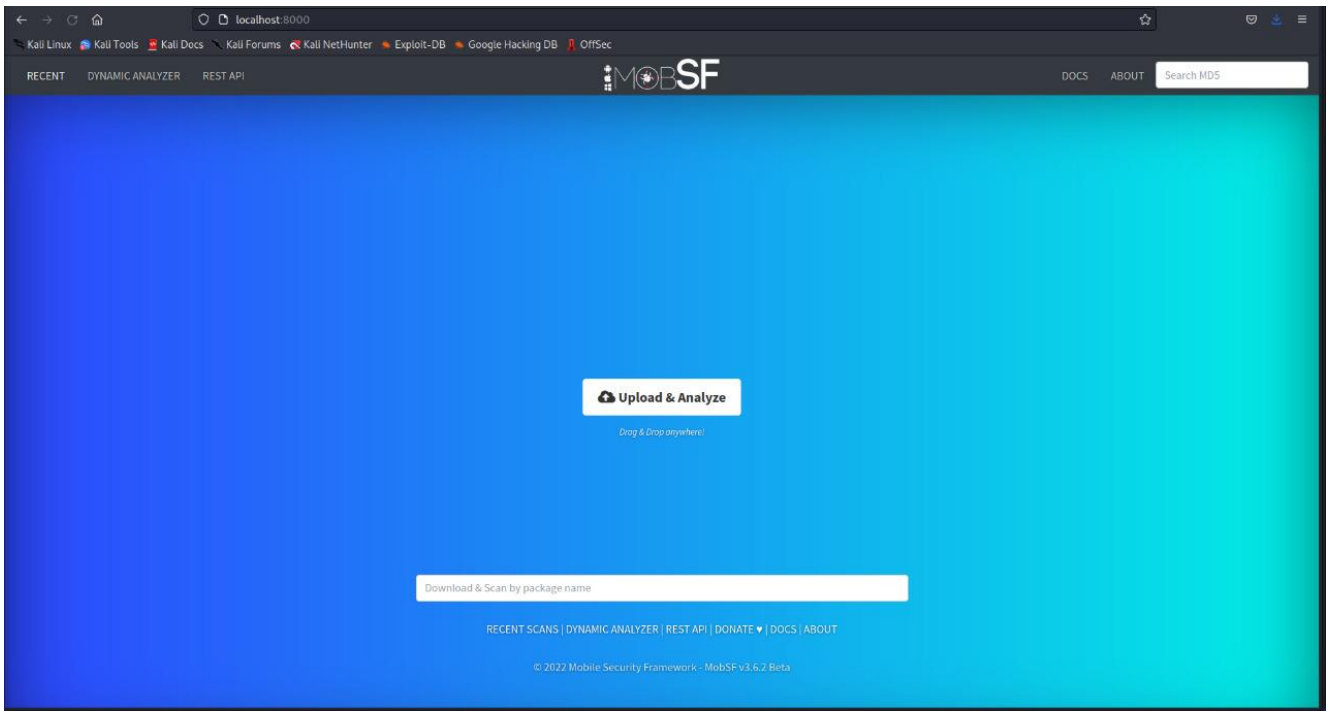


```
Make sure a Genymotion Android VM/Android Studio Emulator is running before performing Dynamic Analysis.
No changes detected in app 'StaticAnalyzer'
[INFO] 13/Dec/2022 13:21:58 - Checking for Update.
[INFO] 13/Dec/2022 13:21:58 - No updates available.
[INFO] 13/Dec/2022 13:22:00 -

[INFO] 13/Dec/2022 13:22:00 - Mobile Security Framework v3.6.2 Beta
REST API Key: 3260d526ec7ba2a061485447cb06959198191bd21d808eca8ebb9a7b6001ebc5
[INFO] 13/Dec/2022 13:22:00 - OS: Linux
[INFO] 13/Dec/2022 13:22:00 - Platform: Linux-5.14.0-kali4-amd64-x86_64-with-glibc2.36
[INFO] 13/Dec/2022 13:22:00 - Dist: kali 2021.4 kali-rolling
[INFO] 13/Dec/2022 13:22:00 - MobSF Basic Environment Check
[WARNING] 13/Dec/2022 13:22:00 - Dynamic Analysis related functions will not work.
Make sure a Genymotion Android VM/Android Studio Emulator is running before performing Dynamic Analysis.
Operations to perform:
  Apply all migrations: StaticAnalyzer, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.
[INFO] 13/Dec/2022 13:22:01 - Checking for Update.
[INFO] 13/Dec/2022 13:22:01 - No updates available.
wkhtmltopdf 0.12.6
[INSTALL] Installation Complete

┌──(joxavy㉿kali)-[~/Mobile-Security-Framework-MobSF]
└─$
```

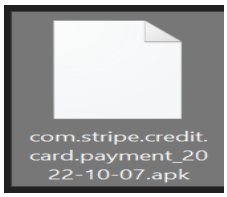
b) Running the tool:

Launching the UI interface with localhost:8000 on any web browser:

We get the following interface:

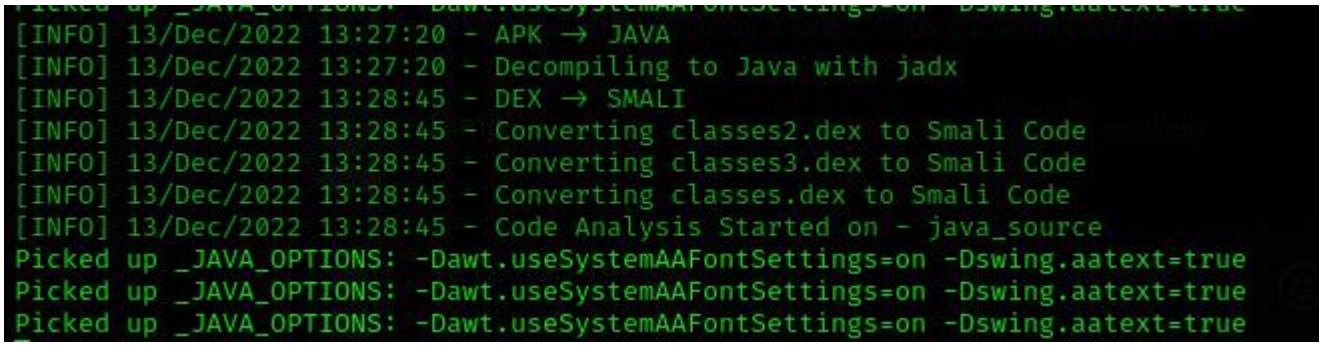


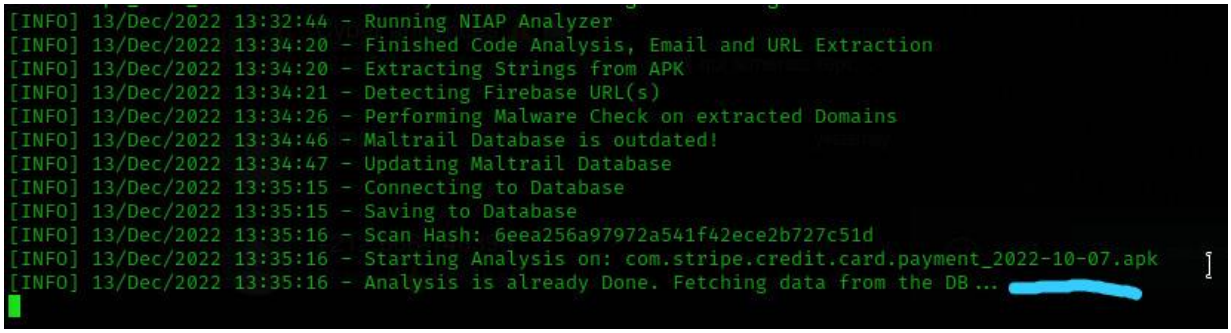
After dropping the apk, that is, the following apk:



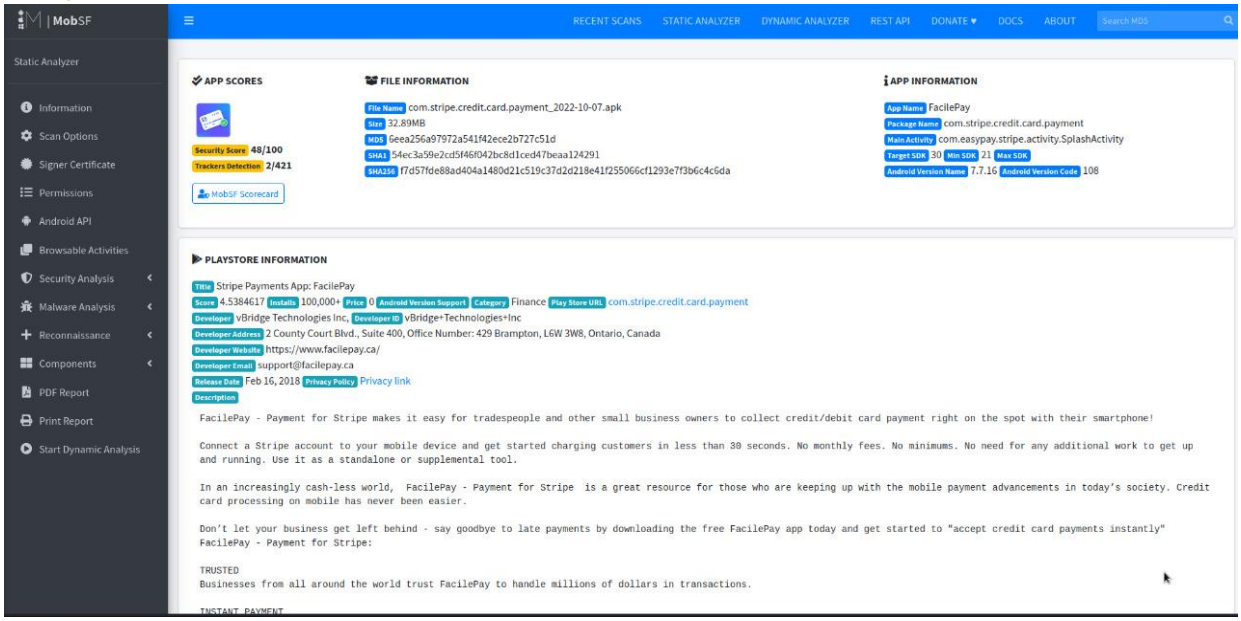
We wait a few minutes then get the following result:

As we can see on the command line while we wait, the apk file is being translated to java, dex to smali, etc.





c) Result in the Ui interface:



First thing we get is general information about the app, the encryption methods, package name, android version, etc.

On the second half, we get play store information, app title, installs, release date, etc.

d) Signer Certificate:

Checking the apk signature:

As we can see, the algorithm used to sign the apk is “**rsassa_pcks1v15**” which is a signature algorithm is the most widely used digital signature scheme in practice. Its two main strengths are its extreme simplicity, which makes it very easy to implement, and that verification of signatures is significantly faster than for DSA or ECDSA.

The hash algorithm used is: SHA256

SHA-256, which stands for secure hash algorithm 256, is a cryptographic hashing algorithm (or function) that's used for message, file, and data integrity verification

e) Application Permissions:

APPLICATION PERMISSIONS

Search:

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.ACCESS_COARSE_LOCATION	dangerous	coarse (network-based) location	Access coarse location sources, such as the mobile network database, to determine an approximate phone location, where available. Malicious applications can use this to determine approximately where you are.
android.permission.ACCESS_FINE_LOCATION	dangerous	fine (GPS) location	Access fine location sources, such as the Global Positioning System on the phone, where available. Malicious applications can use this to determine where you are and may consume additional battery power.
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.ACCESS_WIFI_STATE	normal	view Wi-Fi status	Allows an application to view the information about the status of Wi-Fi.
android.permission.BLUETOOTH	normal	create Bluetooth connections	Allows applications to connect to paired bluetooth devices.
android.permission.CALL_PHONE	dangerous	directly call phone numbers	Allows the application to call phone numbers without your intervention. Malicious applications may cause unexpected calls on your phone bill. Note that this does not allow the application to call emergency numbers.
android.permission.CAMERA	dangerous	take pictures and videos	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.
android.permission.FOREGROUND_SERVICE	normal		Allows a regular application to use Service.startForeground.
android.permission.GET_ACCOUNTS	dangerous	list accounts	Allows access to the list of accounts in the Accounts Service.
android.permission.INTERNET	normal	full Internet access	Allows an application to create network sockets.

Showing 1 to 10 of 24 entries

Previous 1 2 3 Next

PERMISSION	STATUS	INFO	DESCRI
android.permission.NFC	normal	control Near-Field Communication	Allows a readers.
android.permission.READ_CONTACTS	dangerous	read contact data	Allows a applicat
android.permission.READ_EXTERNAL_STORAGE	dangerous	read external storage contents	Allows a
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity	Allows t permiss active, t
android.permission.RECEIVE_BOOT_COMPLETED	normal	automatically start at boot	Allows a longer t running
android.permission.USE_BIOMETRIC	normal		Allows a
android.permission.USE_FINGERPRINT	normal	allow use of fingerprint	This cor
android.permission.VIBRATE	normal	control vibrator	Allows t
android.permission.WAKE_LOCK	normal	prevent phone from sleeping	Allows a
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete external storage contents	Allows a

PERMISSION	STATUS	IN
com.android.vending.BILLING	unknown	U
com.google.android.c2dm.permission.RECEIVE	signature	C2
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	unknown	U
com.google.android.gms.permission.AD_ID	unknown	U

As we can see in this screen, the tool shows the different permissions included in the app, some of which are dangerous permissions such as the one that gives access to the coarse location, because once it is allowed, it will let the app determine the phone location, even if its approximate, and also the camera permission that allows the application to take pictures and videos, that the hackers can later use against you if it's a malicious app.

f) Android API:

API	FILES
Android Notifications	com/easypay/stripe/notification/a.java
Base64 Decode	com/airbnb/lottie/t/b.java com/bumptech/glide/load/n/e.java com/easypay/stripe/util/g.java com/stripe/android/stripe3ds2/security/PublicKeyFactory.java com/stripe/android/stripe3ds2/transactions/ChallengeResponseData.java j/b/b/a/i/a0/j/r0.java k/c.java
Base64 Encode	com/easypay/stripe/util/g.java h/i/l/d.java j/b/b/a/i/a0/j/r0.java j/b/b/a/i/p.java
Certificate Handling	io/grpc/k1/e.java io/grpc/k1/h.java io/grpc/k1/m.java
Crypto	com/easypay/stripe/activity/LoginFingerPrintActivity.java com/easypay/stripe/activity/WebConnectActivity.java com/easypay/stripe/m/a.java com/easypay/stripe/n/a.java com/easypay/stripe/setup/b/f2.java com/easypay/stripe/util/g.java com/easypay/stripe/util/r.java com/nimbusds/jose/jwk/k.java com/nimbusds/jose/u/a.java

It uses base4 decoding, that Is used to encode binary data as printable text. This allows you to transport binary over protocols or mediums that cannot handle binary data formats and require simple text.

```
public Bitmap a(String str) {
    g gVar = this.d.get(str);
    if (gVar == null) {
        return null;
    }
    Bitmap a = gVar.a();
    if (a != null) {
        return a;
    }
    com.airbnb.lottie.b bVar = this.c;
    if (bVar != null) {
        Bitmap a2 = bVar.a(gVar);
        if (a2 != null) {
            c(str, a2);
        }
        return a2;
    }
    String b = gVar.b();
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inScaled = true;
    options.inDensity = 160;
    if (b.startsWith("data:") && b.indexOf("base64,") > 0) {
        try {
            byte[] decode = Base64.decode(b.substring(b.indexOf(44) + 1), 0);
            Bitmap decodeByteArray = BitmapFactory.decodeByteArray(decode, 0, decode.length, options);
            c(str, decodeByteArray);
            return decodeByteArray;
        } catch (IllegalArgumentException e2) {
            d.c("data URL did not have correct base64 format.", e2);
            return null;
        }
    }
    try {
        if (!TextUtils.isEmpty(this.b)) {
            AssetManager assets = this.a.getAssets();
            Bitmap k2 = h.k(BitmapFactory.decodeStream(assets.open(this.b + b), null, options), gVar.e(), gVar.c());
            c(str, k2);
            return k2;
        }
        throw new IllegalStateException("You must set an images folder before loading an image. Set it with LottieComposition#setImagesFolder or LottieDrawable#setImagesFolder");
    } catch (IOException e3) {
        d.c("Unable to open asset.", e3);
```

As we can see here in this class, the base64 is used.

g) Classes that use encryption methods:

Crypto	com/easypay/stripe/activity/LoginFingerPrintActivity.java
	com/easypay/stripe/activity/WebConnectActivity.java
	com/easypay/stripe/m/a.java
	com/easypay/stripe/n/a.java
	com/easypay/stripe/setup/b/f2.java
	com/easypay/stripe/util/g.java
	com/easypay/stripe/util/r.java
	com/nimbusds/jose/jwk/k.java
	com/nimbusds/jose/u/a.java
	com/nimbusds/jose/u/b.java
	com/nimbusds/jose/u/d.java

For example, the LoginFingerPrintActivity:

LoginFingerPrintActivity.java

```
1.  package com.easypay.stripe.activity;
2.
3.  import android.app.KeyguardManager;
4.  import android.hardware.fingerprint.FingerprintManager;
5.  import android.os.Bundle;
6.  import android.security.keystore.KeyGenParameterSpec;
7.  import android.security.keystore.KeyPermanentlyInvalidatedException;
8.  import com.stripe.credit.card.payment.R;
9.  import com.stripe.databinding.ActivityLoginFingerPrintBinding;
10. import java.io.IOException;
11. import java.security.InvalidAlgorithmParameterException;
12. import java.security.InvalidKeyException;
13. import java.security.KeyStore;
14. import java.security.KeyStoreException;
15. import java.security.NoSuchAlgorithmException;
16. import java.security.NoSuchProviderException;
17. import java.security.UnrecoverableKeyException;
18. import java.security.cert.CertificateException;
19. import javax.crypto.Cipher;
20. import javax.crypto.KeyGenerator;
21. import javax.crypto.NoSuchPaddingException;
22. import javax.crypto.SecretKey;
23. /* loaded from: classes.dex */
24. public class LoginFingerPrintActivity extends com.easypay.stripe.g {
25.
26.     /* renamed from: j  reason: collision with root package name */
27.     private KeyStore f1931j;
28.
29.     /* renamed from: k  reason: collision with root package name */
30.     private Cipher f1932k;
31.
32.     public boolean H() {
33.         try {
34.             this.f1932k = Cipher.getInstance("AES/CBC/PKCS7Padding");
35.             try {
36.                 this.f1931j.load(null);
37.                 this.f1932k.init(1, (SecretKey) this.f1931j.getKey("EasyPay", null));
38.                 return true;
39.             } catch (KeyPermanentlyInvalidatedException unused) {
40.                 return false;
41.             } catch (IOException e) {
42.                 e = e;
```

The methods crypto. cipher, crypto. keyGenerator, crypto. SecretKey, are used.

```
public class LoginFingerPrintActivity extends com.easypay.stripe.g {

    /* renamed from: j  reason: collision with root package name */
    private KeyStore f1931j;

    /* renamed from: k  reason: collision with root package name */
    private Cipher f1932k;

    public boolean H() {
        try {
            this.f1932k = Cipher.getInstance("AES/CBC/PKCS7Padding");
            try {
                this.f1931j.load(null);
                this.f1932k.init(1, (SecretKey) this.f1931j.getKey("EasyPay", null));
                return true;
            } catch (KeyPermanentlyInvalidatedException unused) {
                return false;
            } catch (IOException e) {
                e = e;
                throw new RuntimeException("Failed to init Cipher", e);
            } catch (InvalidKeyException e2) {
                e = e2;
                throw new RuntimeException("Failed to init Cipher", e);
            } catch (KeyStoreException e3) {
                e = e3;
                throw new RuntimeException("Failed to init Cipher", e);
            } catch (NoSuchAlgorithmException e4) {
                e = e4;
                throw new RuntimeException("Failed to init Cipher", e);
            } catch (UnrecoverableKeyException e5) {
                e = e5;
                throw new RuntimeException("Failed to init Cipher", e);
            } catch (CertificateException e6) {
                e = e6;
                throw new RuntimeException("Failed to init Cipher", e);
            }
        } catch (NoSuchAlgorithmException | NoSuchPaddingException e7) {
            throw new RuntimeException("Failed to get Cipher", e7);
        }
    }

    protected void I() {
```

They called cipher named “f1932k” and used the encryption methods AES/CBC/PCKS7PADDING to encrypt it, using the try-catch statement that consists of a try block followed by one or more catch clauses, which specify handlers for different exceptions. When an exception is thrown such as KeyPermanentlyinvalidated exception, nosuchalgorithm exception, etc. and the common language runtime (CLR) looks for the catch statement that handles this exception.

m.java

```
1.  package io.grpc.k1;
2.
3.  import java.net.Socket;
4.  import java.util.Arrays;
5.  import java.util.Collections;
6.  import java.util.List;
7.  import javax.net.ssl.HostnameVerifier;
8.  import javax.net.ssl.SSLPeerUnverifiedException;
9.  import javax.net.ssl.SSLSocket;
10. import javax.net.ssl.SSLSocketFactory;
11. /* compiled from: OkHttpTlsUpgrader.java */
12. /* loaded from: classes3.dex */
13. final class m {
14.     static final List<io.grpc.k1.r.g> a = Collections.unmodifiableList(Arrays.asList(io.grpc.k1.r.g.HTTP_2));
15.
16.     static String a(String str) {
17.         return (str.startsWith("[") && str.endsWith("]")) ? str.substring(1, str.length() - 1) : str;
18.     }
19.
20.     public static SSLSocket b(SSLSocketFactory sSLSocketFactory, HostnameVerifier hostnameVerifier, Socket socket, String str, int i2, io.grpc.k1.r.b bVar) {
21.         com.google.common.base.n.o(sSLSocketFactory, "sslSocketFactory");
22.         com.google.common.base.n.o(socket, "socket");
23.         com.google.common.base.n.o(bVar, "spec");
24.         SSLSocket sSLSocket = (SSLSocket) sSLSocketFactory.createSocket(socket, str, i2, true);
25.         bVar.c(sSLSocket, false);
26.         String h2 = j.e().h(sSLSocket, str, bVar.f() ? a : null);
27.         boolean contains = a.contains(io.grpc.k1.r.g.b(h2));
28.         com.google.common.base.n.w(contains, "Only " + a + " are supported, but negotiated protocol is %s", h2);
29.         if (hostnameVerifier == null) {
30.             hostnameVerifier = io.grpc.k1.r.d.a;
31.         }
32.         if (hostnameVerifier.verify(a(str), sSLSocket.getSession())) {
33.             return sSLSocket;
34.         }
35.         throw new SSLPeerUnverifiedException("Cannot verify hostname: " + str);
36.     }
37. }
```

A different class is using another security method that is SSLSocketFactory, it acts as a factory for creating secure sockets.

This class is an abstract subclass of javax. net. SocketFactory.

Secure socket factories encapsulate the details of creating and initially configuring secure sockets.

Https/https connections used in other classes included in the app:

HTTP Connection	com/airbnb/lottie/v/c.java com/bumptech/glide/load/m/j.java com/stripe/android/stripe3ds2/transaction/StripeHttpClient$doGetRequest$2.java com/stripe/android/stripe3ds2/transaction/StripeHttpClient.java com/stripe/net/HttpURLConnectionClient.java com/stripe/net/LiveStripeResponseGetter.java j/b/b/b/a/a/d.java
HTTPS Connection	com/stripe/android/networking/ConnectionFactory.java com/stripe/android/networking/StripeConnection.java com/stripe/android/stripe3ds2/observability/DefaultErrorReporter.java
Inter Process Communication	com/bumptech/glide/m/e.java com/easypay/stripe/activity/AddCustomerActivity.java com/easypay/stripe/activity/AddDiscountActivity.java com/easypay/stripe/activity/AddInvoiceItemActivity.java com/easypay/stripe/activity/AddTaxActivity.java

h) Browsable activity:

BROWSABLE ACTIVITIES

ACTIVITY	INTENT
com.easypay.stripe.activity.MainActivity	**Schemes:** http://, **Hosts:** payment_intent_return,
com.easypay.stripe.activity.RecurringPaymentCreateActivity	**Schemes:** http://, **Hosts:** payment_intent_return,
com.easypay.stripe.activity.SplashActivity	**Schemes:** facilepay://, **Hosts:** navigation,
com.stripe.android.payments.StripeBrowserLauncherActivity	**Schemes:** stripesdk://, **Hosts:** payment_return_url, **Paths:** /com.stripe.credit.card.payment,

Browsable activities are required in order for the intent filter to be accessible from a web browser. Without it, clicking a link in a browser cannot resolve to the app.

i) Certificate analysis:

CERTIFICATE ANALYSIS

Search:

TITLE	SEVERITY	DESCRIPTION
Application vulnerable to Janus Vulnerability	warning	Application is signed with v1 signature scheme, making it vulnerable to Janus vulnerability on Android 5.0-8.0, if signed only with v1 signature scheme. Applications running on Android 5.0-7.0 signed with v1, and v2/v3 scheme is also vulnerable.
Signed Application	info	Application is signed with a code signing certificate

Showing 1 to 2 of 2 entries

Previous 1 Next

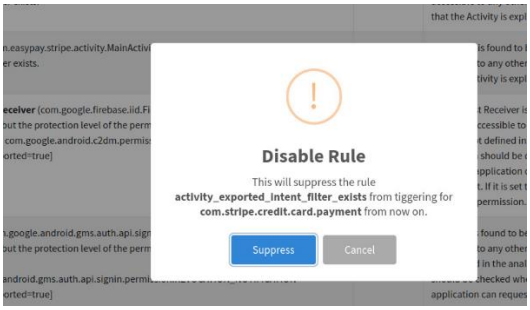
Application is signed with v1 signature scheme which makes its vulnerable to Janus vulnerability which comes from the possibility to add extra bytes to APK files and to DEX files.

j) Activity analysis:

NO	ISSUE	SEVERITY	DESCRIPTION	OPTIONS
1	Clear text traffic is Enabled For App [android:usesCleartextTraffic=true]	high	The app intends to use cleartext network traffic, such as cleartext HTTP, FTP stacks, DownloadManager, and MediaPlayer. The default value for apps that target API level 27 or lower is "true". Apps that target API level 28 or higher default to "false". The key reason for avoiding cleartext traffic is the lack of confidentiality, authenticity, and protections against tampering; a network attacker can eavesdrop on transmitted data and also modify it without being detected.	
2	**Activity** (com.easypay.stripe.activity.RecurringPaymentCreateActivity) is not Protected. An intent-filter exists.	warning	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Activity is explicitly exported.	
3	**Activity** (com.easypay.stripe.activity.MainActivity) is not Protected. An intent-filter exists.	warning	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Activity is explicitly exported.	
4	**Broadcast Receiver** (com.google.firebase.iid.FirebaseInstanceIdReceiver) is Protected by a permission, but the protection level of the permission should be checked. **Permission:** com.google.android.c2dm.permission.SEND [android:exported=true]	warning	A Broadcast Receiver is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. It is protected by a permission which is not defined in the analysed application. As a result, the protection level of the permission should be checked where it is defined. If it is set to normal or dangerous, a malicious application can request and obtain the permission and interact with the component. If it is set to signature, only applications signed with the same certificate can obtain the permission.	
5	**Service** (com.google.android.gms.auth.api.signin.RevocationBoundService) is Protected by a permission, but the protection level of the permission should be checked. **Permission:** com.google.android.gms.auth.api.signin.permission.REVOCATION_NOTIFICATION [android:exported=true]	warning	A Service is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. It is protected by a permission which is not defined in the analysed application. As a result, the protection level of the permission should be checked where it is defined. If it is set to normal or dangerous, a malicious application can request and obtain the permission and interact with the component. If it	

Application uses cleartext network traffic for Http, ftp stacks which poses danger to the app, as well as having many activities not completely protected, therefore many warnings are shown.

As we can see here, there is an option that lets you see what rule needs to be deleted from the activity in order to eliminate this threat.



When you click on it, you can disable that rule and fix it without owning the source code and editing directly from android studio.

k) Code analysis:

</> CODE ANALYSIS

Search:

NO	ISSUE	SEVERITY	STANDARDS	FILES	OPTIONS
1	The App logs information. Sensitive information should never be logged.	info	**CWE:** CWE-532: Insertion of Sensitive Information into Log File **OWASP MASVS:** MSTG-STORAGE-3	com/airbnb/lottie/LottieAnimationView.java com/airbnb/lottie/x/c.java com/bumptech/glide/b.java com/bumptech/glide/l/d.java com/bumptech/glide/l/e.java com/bumptech/glide/load/engine/GlideException.java com/bumptech/glide/load/engine/a0/e.java com/bumptech/glide/load/engine/a0/i.java com/bumptech/glide/load/engine/b0/a.java com/bumptech/glide/load/engine/b0/b.java com/bumptech/glide/load/engine/h.java com/bumptech/glide/load/engine/i.java com/bumptech/glide/load/engine/k.java com/bumptech/glide/load/engine/y.java	

The application logs important information in many classes, as shown in the files part.

As we can see in this class:

```
public void p(String str, String str2) {
    o(com.airbnb.lottie.w.k0.c.o(r.p.d(r.p.k(new ByteArrayInputStream(str.getBytes()))), str2);
}

public void setAnimation(int i2) {
    this.f1400h = i2;
    this.f1399g = null;
    setCompositionTask(e.k(getContext(), i2));
}

@Deprecated
public void setAnimationFromJson(String str) {
    p(str, null);
}

public void setAnimationFromUrl(String str) {
    setCompositionTask(e.m(getContext(), str));
}

public void setComposition(d dVar) {
    if (com.airbnb.lottie.c.a) {
        String str = t;
        Log.v(str, "Set Composition \n" + dVar);
    }
    this.e.setCallback(this);
    this.f1407q = dVar;
    boolean H = this.e.H(dVar);
    i();
    if (getDrawable() != this.e || H) {
        setImageDrawable(null);
        setImageDrawable(this.e);
        onVisibilityChanged(this, getVisibility());
        requestLayout();
        for (j jVar : this.f1405n) {
            jVar.a(dVar);
        }
    }
}
```

It logs the composition value here.

c.java

```
1.  package com.airbnb.lottie.x;
2.
3.  import android.util.Log;
4.  import com.airbnb.lottie.i;
5.  import java.util.HashSet;
6.  import java.util.Set;
7.  /* compiled from: LogcatLogger.java */
8.  /* loaded from: classes.dex */
9.  public class c implements i {
10.     private static final Set<String> a = new HashSet();
11.
12.     @Override // com.airbnb.lottie.i
13.     public void a(String str, Throwable th) {
14.         if (a.contains(str)) {
15.             return;
16.         }
17.         Log.w("LOTTIE", str, th);
18.         a.add(str);
19.     }
20.
21.     public void b(String str, Throwable th) {
22.         if (com.airbnb.lottie.c.a) {
23.             Log.d("LOTTIE", str, th);
24.         }
25.     }
26.
27.     @Override // com.airbnb.lottie.i
28.     public void debug(String str) {
29.         b(str, null);
30.     }
31.
32.     @Override // com.airbnb.lottie.i
33.     public void warning(String str) {
34.         a(str, null);
35.     }
36. }
```

It also logs more information about Lottie in here.

2	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	warning	**CWE:** CWE-312: Cleartext Storage of Sensitive Information **OWASP Top 10:** M9: Reverse Engineering **OWASP MASVS:** MSTG-STORAGE-14	com/bumptech/glide/load/engine/d.java com/bumptech/glide/load/engine/p.java com/bumptech/glide/load/engine/w.java com/bumptech/glide/load/g.java com/stripe/android/EphemeralKey.java com/stripe/android/PaymentConfiguration.java com/stripe/android/auth/PaymentBrowserAuthContract.java com/stripe/android/googlepaylauncher /GooglePayLauncherContract.java com/stripe/android/googlepaylauncher /GooglePayPaymentMethodLauncherContract.java com/stripe/android/googlepaylauncher /GooglePayPaymentMethodLauncherViewModel.java

The application code also contains hardcoded sensitive information that aren't supposed to be hardcoded but soft coded instead, inside its classes.

Hard coding sensitive information exposes that information to attackers. The severity of this rule can vary depending on the kind of information that is disclosed. Frequently, the information disclosed is password or key information, which can lead to remote exploitation.

d.java

```java
package com.bumptech.glide.load.engine;

import java.security.MessageDigest;
/* compiled from: DataCacheKey.java */
/* loaded from: classes.dex */
final class d implements com.bumptech.glide.load.f {
    private final com.bumptech.glide.load.f b;
    private final com.bumptech.glide.load.f c;

    /* JADX INFO: Access modifiers changed from: package-private */
    public d(com.bumptech.glide.load.f fVar, com.bumptech.glide.load.f fVar2) {
        this.b = fVar;
        this.c = fVar2;
    }

    @Override // com.bumptech.glide.load.f
    public void a(MessageDigest messageDigest) {
        this.b.a(messageDigest);
        this.c.a(messageDigest);
    }

    @Override // com.bumptech.glide.load.f
    public boolean equals(Object obj) {
        if (obj instanceof d) {
            d dVar = (d) obj;
            return this.b.equals(dVar.b) && this.c.equals(dVar.c);
        }
        return false;
    }

    @Override // com.bumptech.glide.load.f
    public int hashCode() {
        return (this.b.hashCode() * 31) + this.c.hashCode();
    }

    public String toString() {
        return "DataCacheKey{sourceKey=" + this.b + ", signature=" + this.c + '}';
    }
}
```

This MessageDigest class provides applications the functionality of a message digest algorithm, such as SHA-1 or SHA-256. Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value.

This class returns the value of this message digest class into the result, therefore exposing, as seen in the picture.

On the other hand, it also exposes information about the glide functionality, Glide supports fetching, decoding, and displaying video stills, images, and animated GIFs. Glide includes a flexible API that allows developers to plug in to almost any network stack.

It is implemented into the build Gradle file by adding the following dependencies:

```
repositories {
  google()
  mavenCentral()
}

dependencies {
  implementation 'com.github.bumptech.glide:glide:4.14.2'
  annotationProcessor 'com.github.bumptech.glide:compiler:4.14.2'
}
```

As we can see here, it also exposes the hardcoded information about the Ephemeral Key as shown below:

It is imported from stripe android sdk:

EphemeralKey.java

```
1.  package com.stripe.android;
2.
3.  import android.os.Parcel;
4.  import android.os.Parcelable;
5.  import com.stripe.android.model.StripeModel;
6.  import com.stripe.android.stripe3ds2.transactions.MessageExtension;
7.  import kotlin.h0.d.r;
8.  /* compiled from: EphemeralKey.kt */
9.  /* loaded from: classes2.dex */
10. public final class EphemeralKey implements StripeModel {
11.     private final long created;
12.     private final long expires;
13.     private final String id;
14.     private final boolean isLiveMode;
15.     private final String objectId;
16.     private final String objectType;
17.     private final String secret;
18.     private final String type;
19.     public static final Parcelable.Creator<EphemeralKey> CREATOR = new Creator();
20.     public static final int $stable = 8;
21.
```

Which is makes it quick and easy to build an excellent payment experience in your Android app. They provide powerful and customizable UI elements that can be used out-of-the-box to collect the users' payment details. they also expose the low-level APIs that power those UIs so that you can build fully

custom experiences.

3	App can read/write to External Storage. Any App can read data written to External Storage.	warning	**CWE:** CWE-276: Incorrect Default Permissions **OWASP Top 10:** M2: Insecure Data Storage **OWASP MASVS:** MSTG-STORAGE-2	com/github/mikephil/charting/charts/Chart.java com/github/mikephil/charting/utils/FileUtils.java

The application can also read/write to external storage, as seen here:

```
    }

    public boolean saveToPath(String str, String str2) {
        Bitmap chartBitmap = getChartBitmap();
        try {
            FileOutputStream fileOutputStream = new FileOutputStream(Environment.getExternalStorageDirectory().getPath() + str2 + "/" + str + ".png");
            chartBitmap.compress(Bitmap.CompressFormat.PNG, 40, fileOutputStream);
            fileOutputStream.close();
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
```

A file output stream is an output stream for writing data to a File, in this case, that file is written into external storage with the “getExternalStorageDirectory” function.

```
    public static List<Entry> loadEntriesFromFile(String str) {
        File file = new File(Environment.getExternalStorageDirectory(), str);
        ArrayList arrayList = new ArrayList();
        try {
            BufferedReader bufferedReader = new BufferedReader(new FileReader(file));
            while (true) {
                String readLine = bufferedReader.readLine();
                if (readLine == null) {
                    break;
                }
                String[] split = readLine.split("#");
                if (split.length <= 2) {
                    arrayList.add(new Entry(Float.parseFloat(split[0]), Integer.parseInt(
                } else {
                    int length = split.length - 1;
                    float[] fArr = new float[length];
                    for (int i2 = 0; i2 < length; i2++) {
                        fArr[i2] = Float.parseFloat(split[i2]);
                    }
                    arrayList.add(new BarEntry(Integer.parseInt(split[split.length - 1]),
                }
            }
        } catch (IOException e) {
            Log.e(LOG, e.toString());
        }
        return arrayList;
    }

    public static void saveToSdCard(List<Entry> list, String str) {
        File file = new File(Environment.getExternalStorageDirectory(), str);
        if (!file.exists()) {
            try {
                file.createNewFile();
            } catch (IOException e) {
                Log.e(LOG, e.toString());
            }
        }
        try {
```

JellyBeanApiCompatibility.java

```
package com.j256.ormlite.android.compat;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.CancellationSignal;
import com.j256.ormlite.android.compat.ApiCompatibility;
/* loaded from: classes2.dex */
public class JellyBeanApiCompatibility extends BasicApiCompatibility {

    /* loaded from: classes2.dex */
    protected static class JellyBeanCancellationHook implements ApiCompatibility.CancellationHook {
        private final CancellationSignal cancellationSignal = new CancellationSignal();

        @Override // com.j256.ormlite.android.compat.ApiCompatibility.CancellationHook
        public void cancel() {
            this.cancellationSignal.cancel();
        }
    }

    @Override // com.j256.ormlite.android.compat.BasicApiCompatibility, com.j256.ormlite.android.compat.ApiCompatibility
    public ApiCompatibility.CancellationHook createCancellationHook() {
        return new JellyBeanCancellationHook();
    }

    @Override // com.j256.ormlite.android.compat.BasicApiCompatibility, com.j256.ormlite.android.compat.ApiCompatibility
    public Cursor rawQuery(SQLiteDatabase sQLiteDatabase, String str, String[] strArr, ApiCompatibility.CancellationHook cancellationHook) {
        return cancellationHook == null ? sQLiteDatabase.rawQuery(str, strArr) : sQLiteDatabase.rawQuery(str, strArr, ((JellyBeanCancellationHook) cancellationHook).cancellationSignal);
    }
}
```

They also write information from this class to external storage.

4	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	warning	**CWE:** CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') **OWASP Top 10:** M7: Client Code Quality	com/j256/ormlite/android/AndroidCompiledStatement.java com/j256/ormlite/android/AndroidDatabaseConnection.java com/j256/ormlite/android/compat/ApiCompatibility.java com/j256/ormlite/android/compat/BasicApiCompatibility.java com/j256/ormlite/android/compat/JellyBeanApiCompatibility.java h/v/a/g/a.java j/b/b/a/i/a0/j/a0.java j/b/b/a/i/a0/j/r0.java j/b/b/a/i/a0/j/t0.java

Application uses SQL database which is not trusted.

Example of usage in some of the classes’ code:

the application uses insecure random number generator which can cause danger to the functionalities it is used with, especially if it's for encrypting personal data of users.

```
48. import java.util.Iterator;
49. import java.util.Locale;
50. import java.util.Random;
51. import java.util.TimeZone;
52. import java.util.regex.Pattern;
53. import javax.crypto.IllegalBlockSizeException;
54. import javax.crypto.NoSuchPaddingException;
55. import org.json.JSONException;
56. import org.json.JSONObject;
57. /* compiled from: Util.java */
58. /* loaded from: classes.dex */
59. public class r {
60.     public static p b;
```

5	The App uses an insecure Random Number Generator.	warning	**CWE:** CWE-330: Use of Ins Random Values **OWASP Top 10:** M5: Insu Cryptography **OWASP MASVS:** MSTG-C

As seen below, it imports the util.random functionality:

```
7.      }
8.
9.      public static int q(Context context) {
0.          int[] intArray = context.getResources().getIntArray(R.array.androidcolors);
1.          int nextInt = new Random().nextInt(intArray.length);
2.          if (nextInt >= intArray.length) {
3.              nextInt++;
4.          }
5.          return intArray[nextInt];
6.      }
7.
```

AndroidCompiledStatement.java

```
1.  package com.j256.ormlite.android;
2.
3.  import android.database.Cursor;
4.  import android.database.SQLException;
5.  import android.database.sqlite.SQLiteDatabase;
6.  import android.database.sqlite.SQLiteStatement;
7.  import com.j256.ormlite.android.compat.ApiCompatibility;
8.  import com.j256.ormlite.android.compat.ApiCompatibilityUtils;
9.  import com.j256.ormlite.dao.ObjectCache;
10. import com.j256.ormlite.field.SqlType;
11. import com.j256.ormlite.logger.Logger;
```

Public class random function: An instance of this class is used to generate a stream of pseudorandom numbers. The class uses a 48-bit seed, which is modified using a linear congruential formula.

If two instances of Random are created with the same seed, and the same sequence of method calls is made for each, they will generate and return identical sequences of numbers. Which means it doesn't provide semantic security or confidential security, it is very much advised to use a different method.

The application uses certificate pinning to detect Man in The Middle attacks

6	This App uses SSL certificate pinning to detect or prevent MITM attacks in secure communication channel.	secure	**OWASP MASVS:** MSTG-NETWORK-4	com/easypay/stripe/n/a.java

What is certificate pinning?

okhttp3

Class CertificatePinner

java.lang.Object
okhttp3.CertificatePinner

```
public final class CertificatePinner
extends Object
```

Constrains which certificates are trusted. Pinning certificates defends against attacks on certificate authorities. It also prevents connections through man-in-the-middle certificate authorities either known or unknown to the application's user.

This class currently pins a certificate's Subject Public Key Info as described on Adam Langley's Weblog. Pins are either base64 SHA-256 hashes as in HTTP Public Key Pinning (HPKP) or SHA-1 base64 hashes as in Chromium's static certificates.

As shown inside the class, this is how it's used:

```
t(300L, TimeUnit.SECONDS).certificatePinner(builder.add("www.facilepayforstripe.ca", "sha256/" + str).build()).addInterceptor(httpLoggingInterceptor).addI
```

They use the SHA256 algorithm, which is very secure, and almost impossible to be cracked because it takes 12,700,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000 years to be cracked;

There will also be around 36^64 / 2^256 or 34,600,000,000,000,000,000,000 collisions found. Note that the possible combinations of the string are greater than the number of possible hashes.

7	App creates temp file. Sensitive information should never be written into a temp file.	warning	**CWE:** CWE-276: Incorrect Default Permissions **OWASP Top 10:** M2: Insecure Data Storage **OWASP MASVS:** MSTG-STORAGE-2	h/s/b.java

The application creates temporary file that contain sensitive information in the class called h/s/b.java

```
    private static void b(Closeable closeable) {
        try {
            closeable.close();
        } catch (IOException e) {
            Log.w("MultiDex", "Failed to close resource", e);
        }
    }

    private static void c(ZipFile zipFile, ZipEntry zipEntry, File file, String str) {
        InputStream inputStream = zipFile.getInputStream(zipEntry);
        File createTempFile = File.createTempFile("tmp-" + str, ".zip", file.getParentFile());
        Log.i("MultiDex", "Extracting " + createTempFile.getPath());
        try {
            ZipOutputStream zipOutputStream = new ZipOutputStream(new BufferedOutputStream(new FileOutputStream(createTempFile)));
            ZipEntry zipEntry2 = new ZipEntry("classes.dex");
            zipEntry2.setTime(zipEntry.getTime());
            zipOutputStream.putNextEntry(zipEntry2);
            byte[] bArr = new byte[Http2.INITIAL_MAX_FRAME_SIZE];
```

The application uses bad encryption modes that are vulnerable to attacks, such as the PKCS5/PKCS7 paddings.

8	The App uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is vulnerable to padding oracle attacks.	high	**CWE:** CWE-649: Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-3	com/easypay/stripe/activity/LoginFingerPrintActivity.java com/easypay/stripe/util/g.java

As shown previously in the class that is responsible for encrypting the fingerprint:

```
public boolean H() {
    try {
        this.f1932k = Cipher.getInstance("AES/CBC/PKCS7Padding");
        try {
            this.f1931j.load(null);
            this.f1932k.init(1, (SecretKey) this.f1931j.getKey("EasyPay", null));
            return true;
        } catch (KeyPermanentlyInvalidatedException unused) {
            return false;
        } catch (IOException e) {
            e = e;
            throw new RuntimeException("Failed to init Cipher", e);
        } catch (InvalidKeyException e2) {
```

As seen here, the finger print has been encrypted with it.

```
/* JADX INFO: Access modifiers changed from: protected */
@Override // com.easypay.stripe.g, androidx.fragment.app.d, androidx.activity.ComponentActivity, androidx.core.app.g, android.app.Activity
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    ActivityLoginFingerPrintBinding activityLoginFingerPrintBinding = (ActivityLoginFingerPrintBinding) androidx.databinding.g.g(this, R.layout.activity_login_finger_print);
    KeyguardManager keyguardManager = (KeyguardManager) getSystemService("keyguard");
    FingerprintManager fingerprintManager = (FingerprintManager) getSystemService("fingerprint");
    if (!fingerprintManager.isHardwareDetected()) {
        activityLoginFingerPrintBinding.tvError.setText(getString(R.string.msg_no_fingerprint_sensor));
    } else if (androidx.core.content.a.a(this, "android.permission.USE_FINGERPRINT") != 0) {
        activityLoginFingerPrintBinding.tvError.setText(getString(R.string.msg_fingetprint_permission_not_enable));
    } else if (!fingerprintManager.hasEnrolledFingerprints()) {
        activityLoginFingerPrintBinding.tvError.setText(getString(R.string.msg_register_atlist_one_fingerprint));
    } else if (!keyguardManager.isKeyguardSecure()) {
        activityLoginFingerPrintBinding.tvError.setText(getString(R.string.msg_lock_screen_security_not_enable_in_setting));
    } else {
        I();
        if (H()) {
            new com.easypay.stripe.util.j(this).a(fingerprintManager, new FingerprintManager.CryptoObject(this.f1932k));
        }
    }
```

Also used here:

```
/* loaded from: classes.dex */
public class g {
    Cipher a = Cipher.getInstance("AES/CBC/PKCS5Padding");
    byte[] b = new byte[32];
    byte[] c = new byte[16];

    /* JADX INFO: Access modifiers changed from: private */
    /* compiled from: CriptoHelper.java */
    /* loaded from: classes.dex */
    public enum a {
        ENCRYPT,
        DECRYPT
```

Application copies sensitive information to clipboard which can put it at risk of being used by other applications for malicious reasons.

10	This App copies data to clipboard. Sensitive data should not be copied to clipboard as other applications can access it.	Info	**OWASP MASVS:** MSTG-STORAGE-10	com/easypay/stripe/activity/PaymentLinkSendActivity.java

As shown below:

```
}

/* JADX INFO: Access modifiers changed from: private */
public void P() {
    ((ClipboardManager) getSystemService("clipboard")).setPrimaryClip(ClipData.newPlainText(AnnotatedPrivateKey.LABEL, this.f1962k));
    com.easypay.stripe.util.r.k0(this, getString(R.string.lbl_copy_to_clipboard_msg));
}

private void Q(String str, final ImageView imageView) {
    try {
```

l) Shared library binary analysis:

We get the libraries that are used in the application with all related information:

SHARED LIBRARY BINARY ANALYSIS

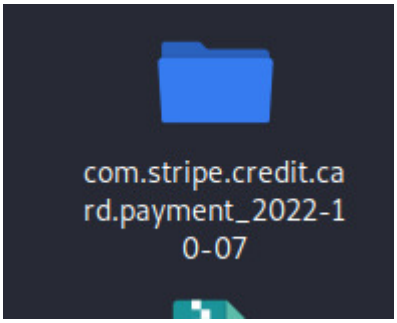
Search:

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
1	lib/x86_64 /libcardioRecognizer_tegra2.so	**True** info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	**True** info This shared object has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	**None** info The shared object does not have run-time search path or RPATH set.	**None** info The shared object does not have RUNPATH set.	**False** warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	**True** info Symbols are stripped.
2	lib/x86_64/libcardioRecognizer.so	**True** info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	**True** info This shared object has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	**None** info The shared object does not have run-time search path or RPATH set.	**None** info The shared object does not have RUNPATH set.	**False** warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	**True** info Symbols are stripped.
3	lib/x86_64/libcardioDecider.so	**True** info The shared object has NX	**True** info This shared object has a stack canary	**None** info The shared	**None** info The shared	**False** warning The shared object does not have any fortified	**True** info Symbols are

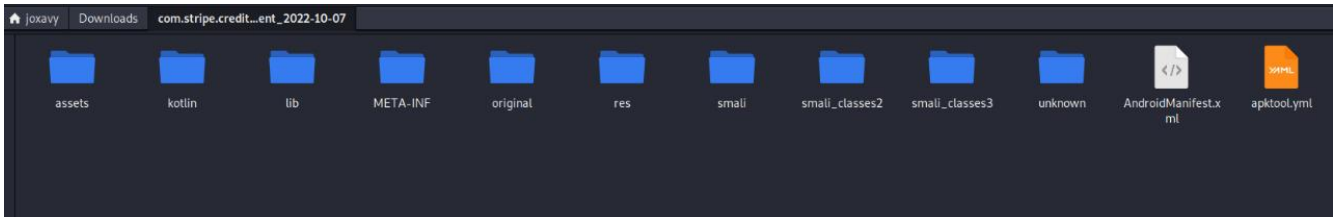
We use the apktool in order to decompress the .apk file and get its components:

```
┌──(joxavy㉿kali)-[~/Downloads]
└─$ apktool d com.stripe.credit.card.payment_2022-10-07.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1-dirty on com.stripe.credit.card.payment_2022-10-07.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/joxavy/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Baksmaling classes2.dex...
I: Baksmaling classes3.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
I: Copying META-INF/services directory

┌──(joxavy㉿kali)-[~/Downloads]
└─$
```

A folder holding the same name as the apk file, gets created:

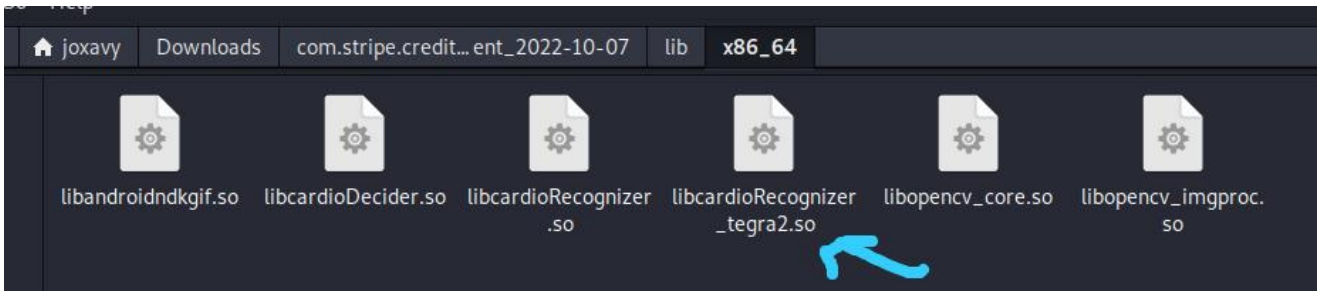


We check the components of this folder, we see:



After entering the lib folder, we find different folders that are issued for distinct architectures like x86, Arm,etc.

The file we are interested in is located inside the x86 architecture's folder:

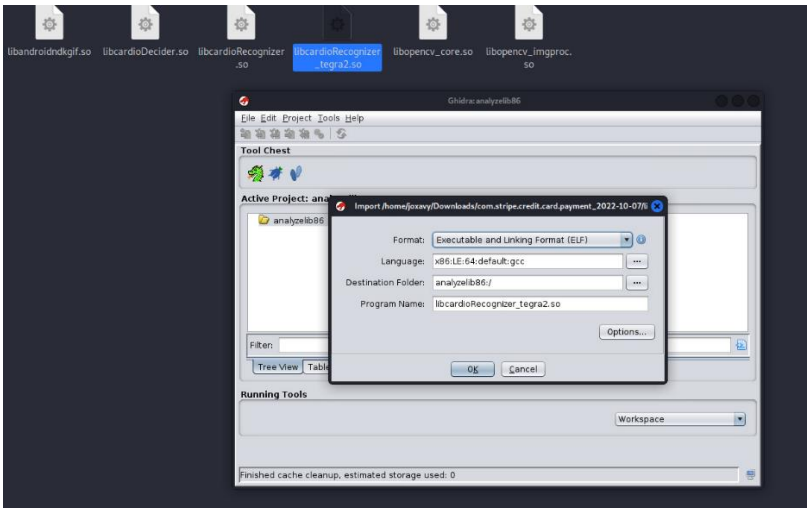


That is the first file that appeared in analysis table:

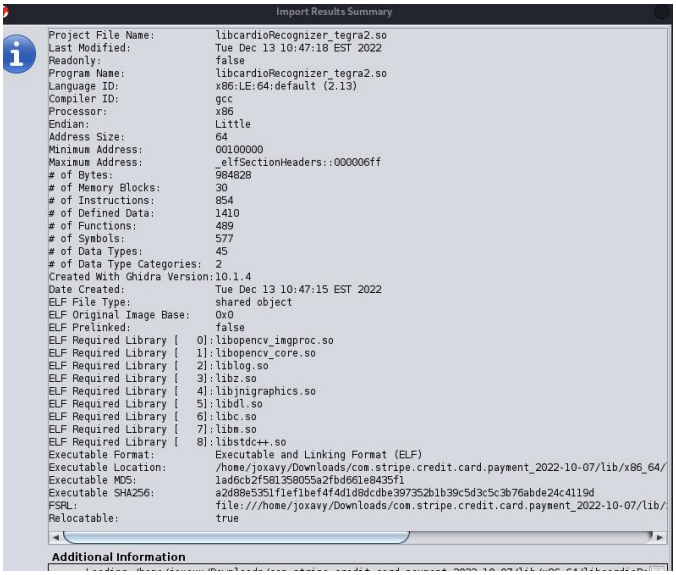
NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
1	lib/x86_64 /libcardioRecognizer_tegra2.so	**True** info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	**True** info This shared object has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	**None** info The shared object does not have run-time search path or RPATH set.	**None** info The shared object does not have RUNPATH set.	**False** warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	**True** info Symbols are stripped.

We are going to check it for fortified functions as the warning states it doesn't have any.

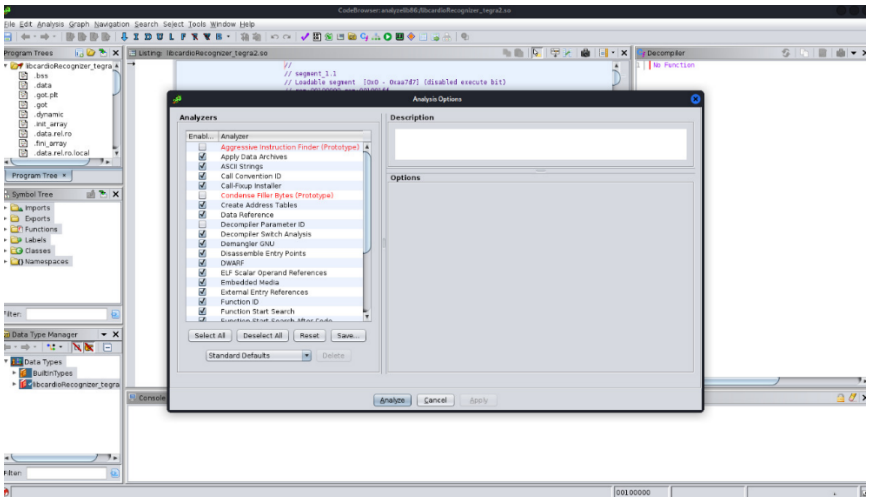
First step we will take, is that we will open the .so file with Ghidra:



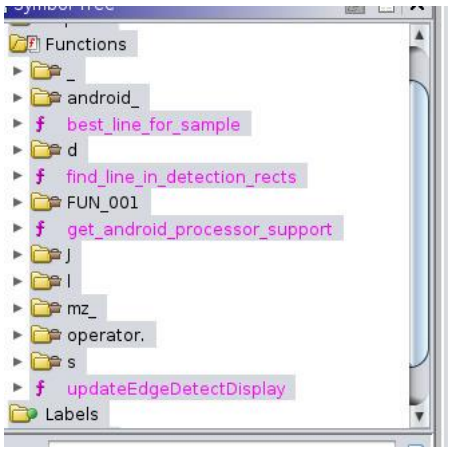
We get general information about the file:



Analyze the file:



After the analysis is done, we check the functions' part:

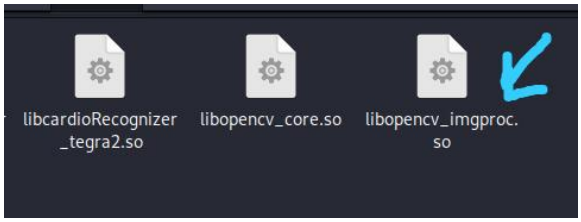


As we can see we can’t find the fortified functions.

I will then proceed to check the second file of the table that is “libopencv_imgproc.so” in order to detect

if it contains stack canary values.

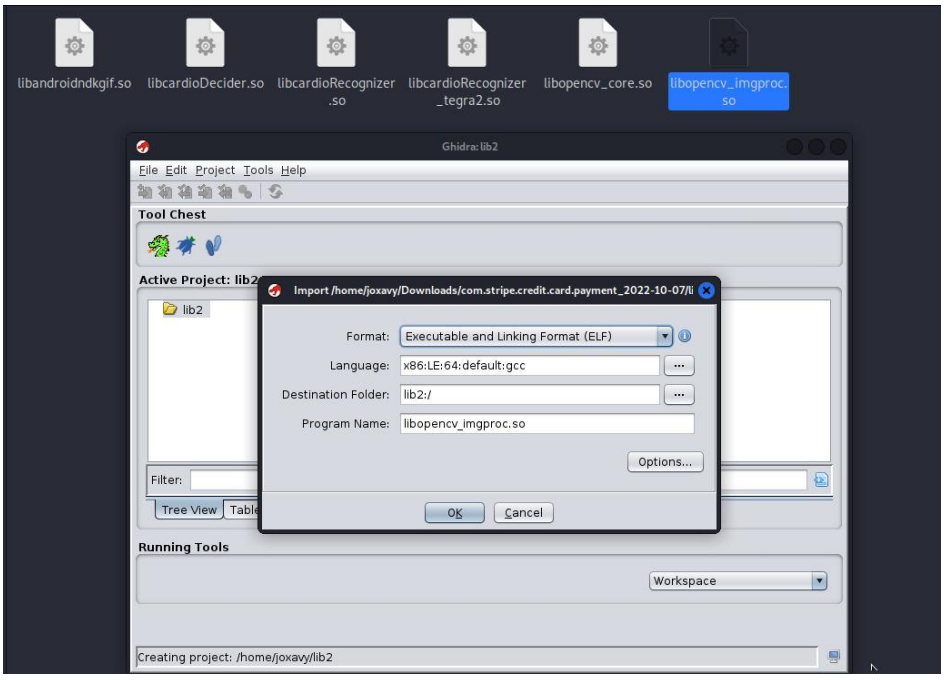
What are stack canaries? They are a secret value placed on the stack which changes every time the program is started. Prior to a function return, the stack canary is checked and if it appears to be modified, the program exits immediately.



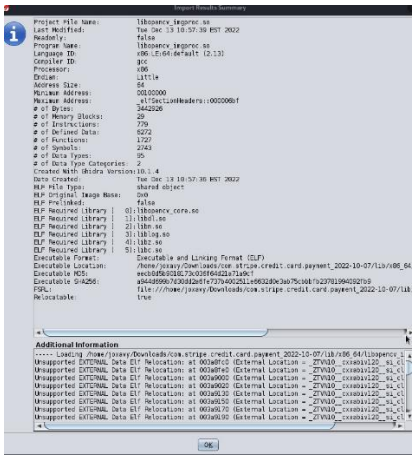
As stated in the warning

5	lib/x86_64/libopencv_imgproc.so	**True** info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	**False** high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	**None** info The shared object does not have run-time search path or RPATH set.	**None** info The shared object does not have RUNPATH set.	**False** warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	**True** info Symbols are stripped.

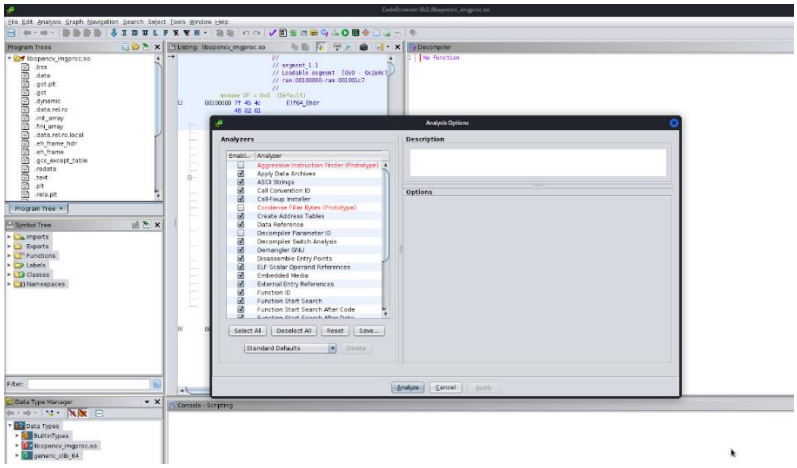
We open it with Ghidra:



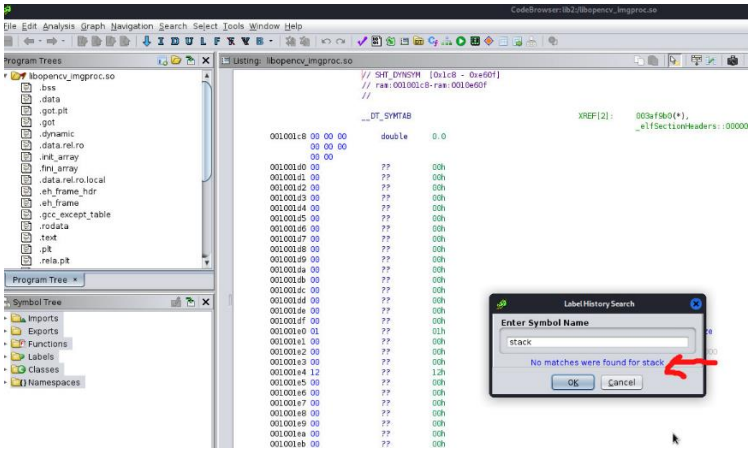
The general information about this library Is shown as follows:



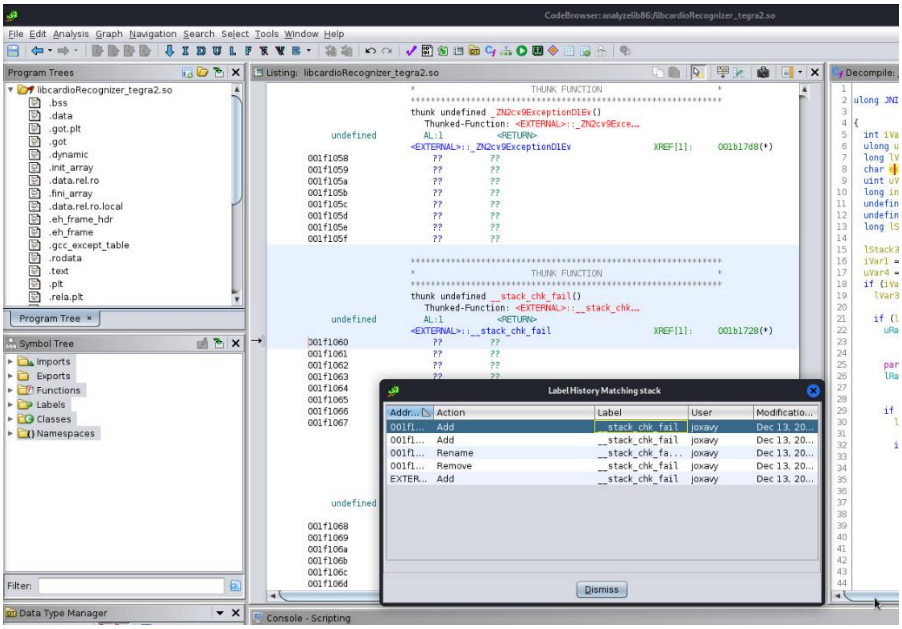
We analyze it:



After searching for it, we see that there is no match.

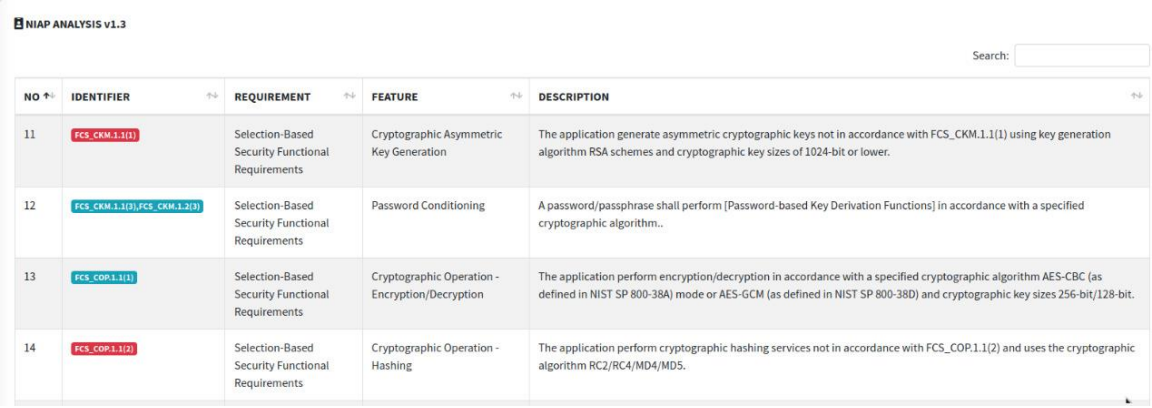


I will then compare it with another library to see the difference, here is the result after analyzing “libcardReconginzer_tegra2.so” library:



As I search for the stack canaries, I get the output above.

m) Niap analysis:



NO	IDENTIFIER	REQUIREMENT	FEATURE	DESCRIPTION
11	FCS_CKM.1.1(1)	Selection-Based Security Functional Requirements	Cryptographic Asymmetric Key Generation	The application generate asymmetric cryptographic keys not in accordance with FCS_CKM.1.1(1) using key generation algorithm RSA schemes and cryptographic key sizes of 1024-bit or lower.
12	FCS_CKM.1.1(3),FCS_CKM.1.2(3)	Selection-Based Security Functional Requirements	Password Conditioning	A password/passphrase shall perform [Password-based Key Derivation Functions] in accordance with a specified cryptographic algorithm..
13	FCS_COP.1.1(1)	Selection-Based Security Functional Requirements	Cryptographic Operation - Encryption/Decryption	The application perform encryption/decryption in accordance with a specified cryptographic algorithm AES-CBC (as defined in NIST SP 800-38A) mode or AES-GCM (as defined in NIST SP 800-38D) and cryptographic key sizes 256-bit/128-bit.
14	FCS_COP.1.1(2)	Selection-Based Security Functional Requirements	Cryptographic Operation - Hashing	The application perform cryptographic hashing services not in accordance with FCS_COP.1.1(2) and uses the cryptographic algorithm RC2/RC4/MD4/MD5.

What are the NIAP “Requirements for Vetting Mobile Applications”?

NIAP is a collaboration between the National Institute for Standards and Technology (NIST) and the National Security Agency (NSA).

NIAP oversees the Common Criteria evaluation and certification guidelines used to evaluate whether commercial IT products meet security standards for government deployments.

The NIAP mobile app vetting requirements grew out of the Protection Profile for Application Software. The profile explains the security standards against which an application will be assessed before it’s approved for deployment within a government environment.

Why is there a warning?

To understand it, first, we need to explain what FCS_CKM1.1 is.

FCS_CKM.1, Cryptographic key generation, is a component that requires the cryptographic key sizes and method used to generate cryptographic keys to be specified, this can be in accordance with an assigned standard. It should be used to specify the cryptographic key sizes and the method (e.g. algorithm) used to generate the cryptographic keys.

Only one instance of the component is needed for the same method and multiple key sizes. The key size could be common or different for the various entities, and could be either the input to or the output from the method.

The application uses an asymmetric key that is not in accordance with the FCS_CKM1.1 key generation method.

14	FCS_COP.1.1(2)	Selection-Based Security Functional Requirements	Cryptographic Operation - Hashing	The application perform cryptographic hashing services not in accordance with FCS_COP.1.1(2) and uses the cryptographic algorithm RC2/RC4/MD4/MD5.

As detected in the problem, the application also uses a cryptographic hashing service that isn't in accordance with the FCS_COP.1.1 and instead uses the algorithms RC2/RC4/MD4/MD5.

Here is a list of the algorithms that are actually supposed to be used with this operation:

FCS_COP.1(2) Cryptographic Operation - Hashing (Refined)

FCS_COP.1.1(2)

The **OS** shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm ***[selection:***

- *SHA-1,*
- *SHA-256,*
- *SHA-384,*
- *SHA-512]*

and message digest sizes [selection:

- ***160 bits,***
- ***256 bits,***
- ***384 bits,***
- ***512 bits]***

n) File analysis:

FILE ANALYSIS

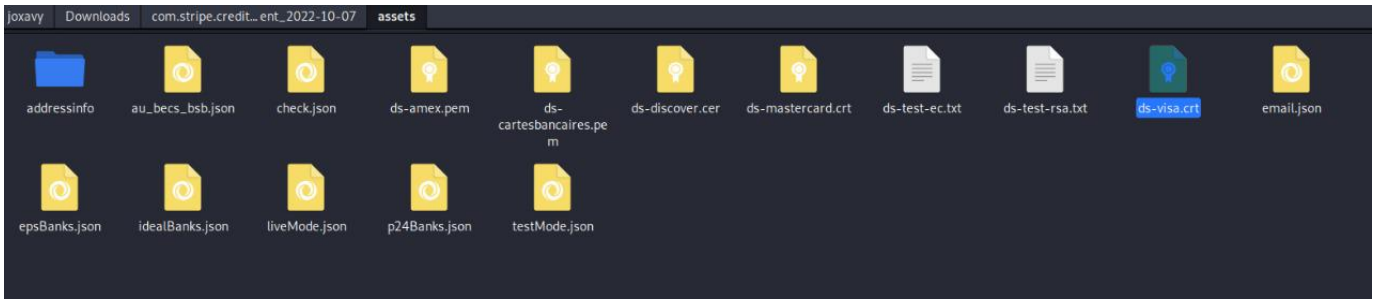
Se

NO	ISSUE	FILES
1	Certificate/Key files hardcoded inside the app.	META-INF/services/io.grpc.ManagedChannelProvider META-INF/services/java.security.Provider assets/ds-amex.pem assets/ds-cartesbancaires.pem assets/ds-discover.cer assets/ds-mastercard.crt assets/ds-visa.crt

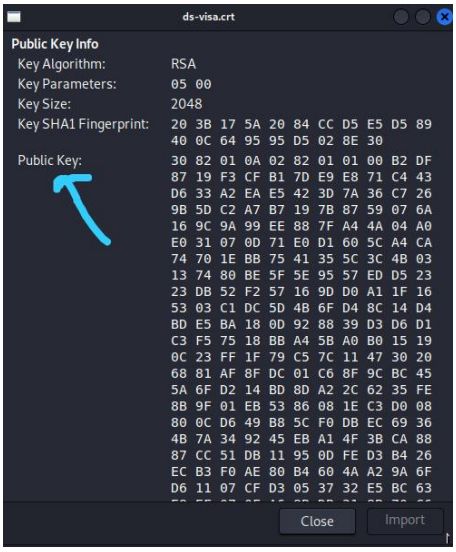
Showing 1 to 1 of 1 entries

It displays the files where the certificate/key files are hardcoded.

We will check one of these files by accessing their location on the decompressed apk folder:

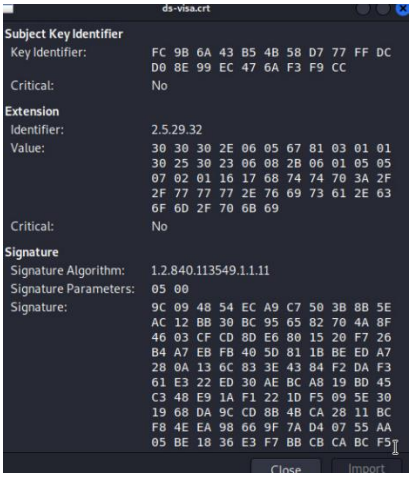


Such as the ds-visa.crt file:

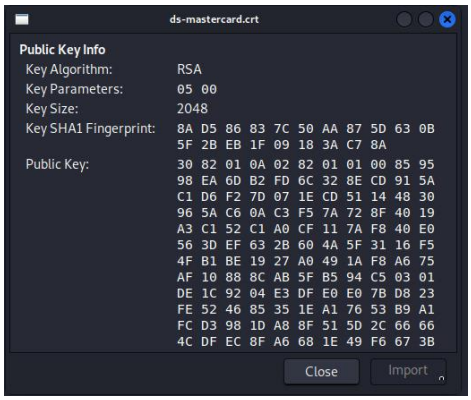


After clicking it, we see the public key used to encrypt this file as shown above.

There is also information about the signature:



We can also check the other ds-mastercard.crt file:

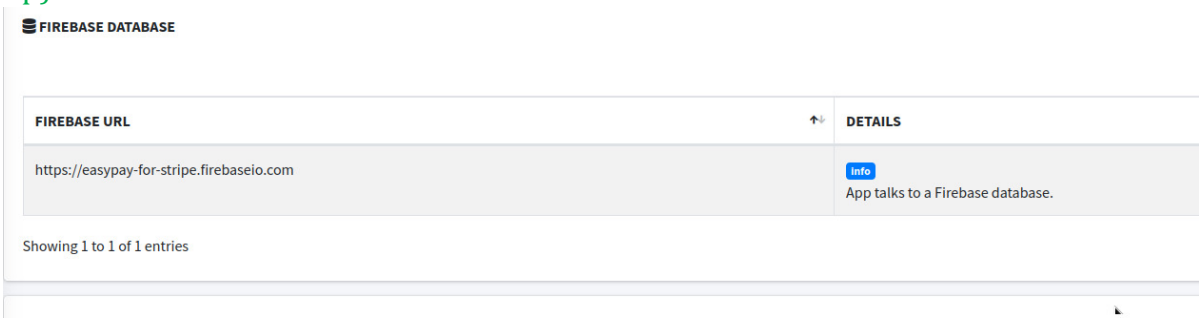


The public key for it is also shown, even the SHA1 fingerprint.

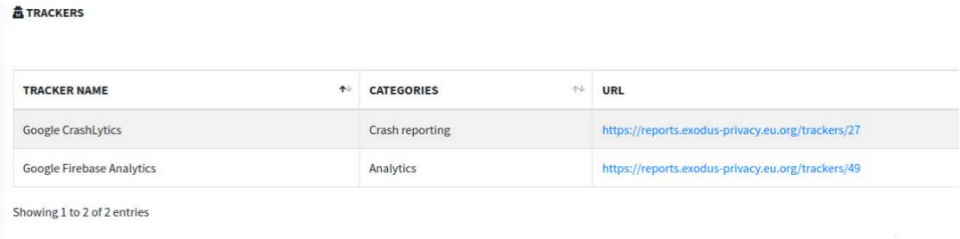
o) Server locations:

The tool also provides the server locations which are connected to the app.

p) Firebase database:



q) Trackers:



As we can see here, it shows which trackers are tracking users' information.

r) Activities:

We can see all the different activities that are used in the app:

ACTIVITIES

com.easypay.stripe.activity.ChangePassword
com.easypay.stripe.activity.ForgotPassword
com.easypay.stripe.activity.LoginActivity
com.easypay.stripe.activity.SignUpActivity
com.easypay.stripe.activity.RecurringPaymentCreateActivity
com.easypay.stripe.activity.RecurringChargeDetailFromCustomer
com.easypay.stripe.activity.RecurringPaymentDetailActivity
com.easypay.stripe.activity.WebViewLoadActivity
com.easypay.stripe.activity.QrCodeGenerateActivity
com.easypay.stripe.activity.PaymentLinkSendActivity
com.easypay.stripe.activity.PaymentLinkActivity
com.easypay.stripe.activity.PaidPdfLoadActivity
com.easypay.stripe.activity.CustomerListActivity
com.easypay.stripe.activity.EditInvoiceItemActivity
com.easypay.stripe.activity.NewInvoiceActivity
com.easypay.stripe.activity.InvoiceItemList
com.easypay.stripe.activity.AddInvoiceItemActivity
com.easypay.stripe.activity.EditDiscountActivity
com.easypay.stripe.activity.EditTaxActivity
com.easypay.stripe.activity.AddDiscountActivity
com.easypay.stripe.activity.DiscountListActivity
com.easypay.stripe.activity.AddTaxActivity
com.easypay.stripe.activity.TaxRateListActivity
com.easypay.stripe.activity.InvoiceSendActivity
com.easypay.stripe.activity.InvoiceViewActivity
com.easypay.stripe.activity.SplashActivity
com.easypay.stripe.activity.MainActivity
com.easypay.stripe.setup.SelectLanguageActivity
com.easypay.stripe.activity.WebConnectActivity

s) Services:

We can also see the services used:

SERVICES

com.easypay.stripe.notification.MyFirebaseMessagingService
com.easypay.stripe.services.CallingApiService
com.easypay.stripe.services.AccountSyncApiService
com.google.firebase.components.ComponentDiscoveryService
com.google.android.gms.cast.framework.media.MediaNotificationService
com.google.android.gms.cast.framework.ReconnectionService
com.google.firebase.messaging.FirebaseMessagingService
com.google.android.gms.auth.api.signin.RevocationBoundService
com.google.android.gms.measurement.AppMeasurementService
com.google.android.gms.measurement.AppMeasurementJobService
androidx.work.impl.background.systemalarm.SystemAlarmService
androidx.work.impl.background.systemjob.SystemJobService
androidx.work.impl.foreground.SystemForegroundService
com.google.android.datatransport.runtime.backends.TransportBackendDiscovery
com.google.android.datatransport.runtime.scheduling.jobscheduling.JobInfoSchedulerService
androidx.room.MultiInstanceInvalidationService

t) The receivers and the providers:

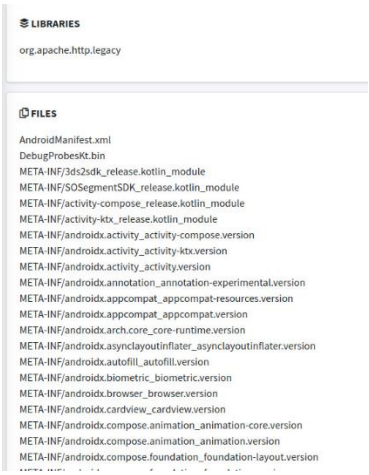
RECEIVERS

com.google.android.gms.cast.framework.media.MediaIntentReceiver
com.google.firebase.iid.FirebaseInstanceIdReceiver
com.google.android.gms.measurement.AppMeasurementReceiver
androidx.work.impl.utils.ForceStopRunnable$BroadcastReceiver
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryChargingProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryNotLowProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$StorageNotLowProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$NetworkStateProxy
androidx.work.impl.background.systemalarm.RescheduleReceiver
androidx.work.impl.background.systemalarm.ConstraintProxyUpdateReceiver
androidx.work.impl.diagnostics.DiagnosticsReceiver
com.google.android.datatransport.runtime.scheduling.jobscheduling.AlarmManagerSchedulerBroadcastReceiver
androidx.profileinstaller.ProfileInstallReceiver

PROVIDERS

androidx.core.content.FileProvider
com.google.firebase.perf.provider.FirebasePerfProvider
com.google.firebase.provider.FirebaseInitProvider
androidx.work.impl.WorkManagerInitializer
androidx.startup.InitializationProvider

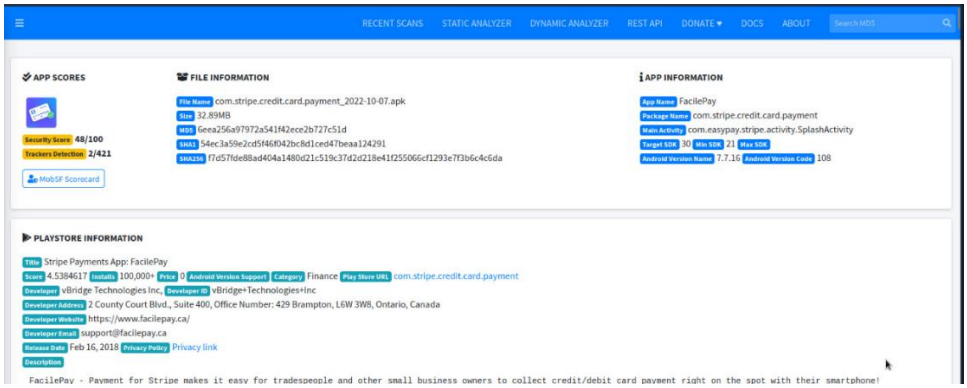
u) The libraries and files:



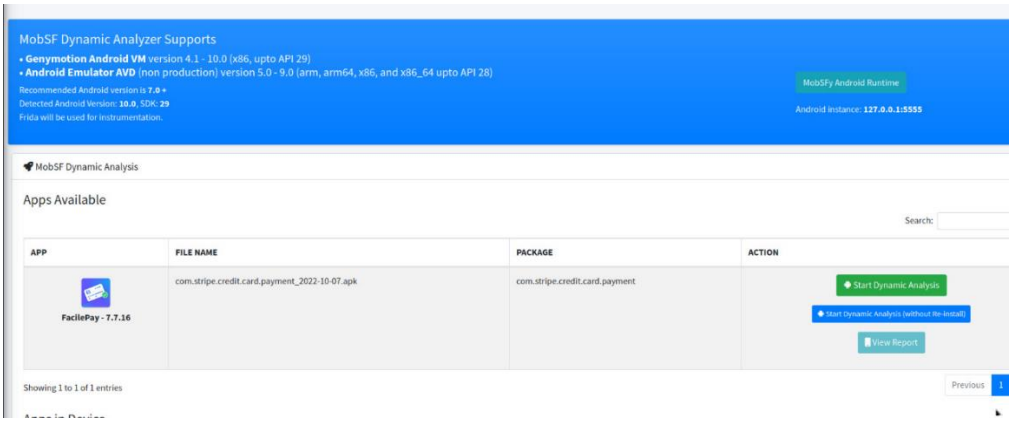
IV. Dynamic analysis:

1. Dynamic analysis with MobSF:

We need to select the dynamic analyzer:

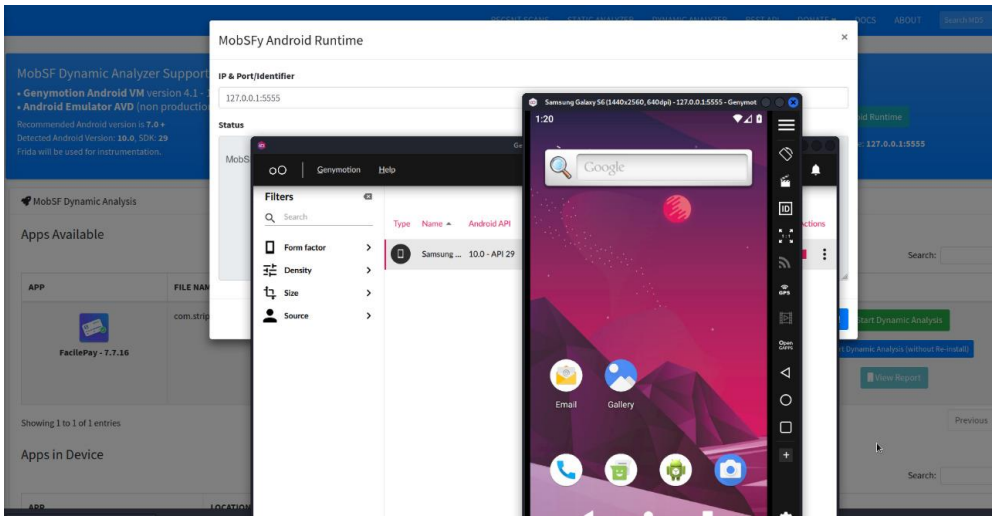


And start analyzing the app by clicking on the start analysis button.

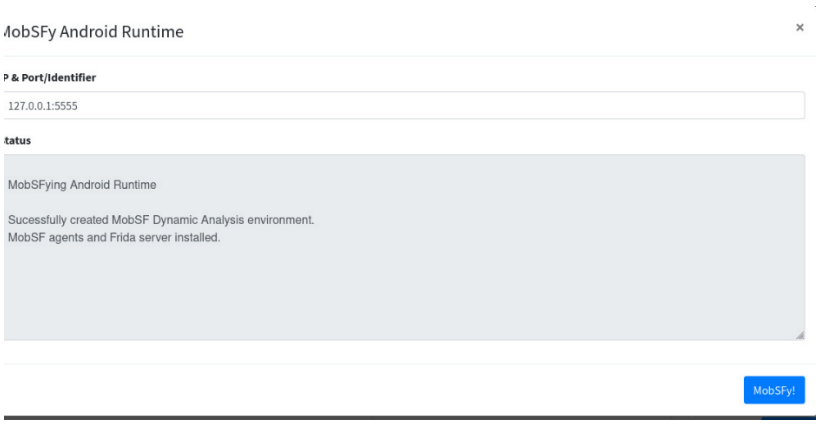


Which requires Genymotion to be already running and connected, after installing it, we launch the emulator and connect it to the dynamic analyzer of MOBSF.

It automatically detects the emulator and connects to it via the Ip address 127.0.0.1 and the port 5555

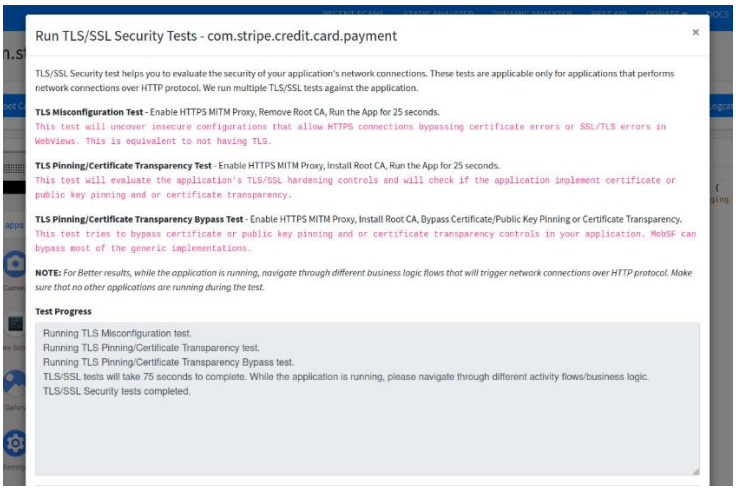


As seen this result:



The environment has been successfully created.

After connecting it, we can start by running TLS/SSL security tests and see if the tool can bypass them:



After running them, we get the following result:

TESTS	RESULT
TLS Misconfiguration Test	✅
TLS Pinning/Certificate Transparency Test	❌
TLS Pinning/Certificate Transparency Bypass Test	❌
Cleartext Traffic Test	✅

Which means that the tool was able to bypass both TLS misconfiguration test and cleartext traffic test.

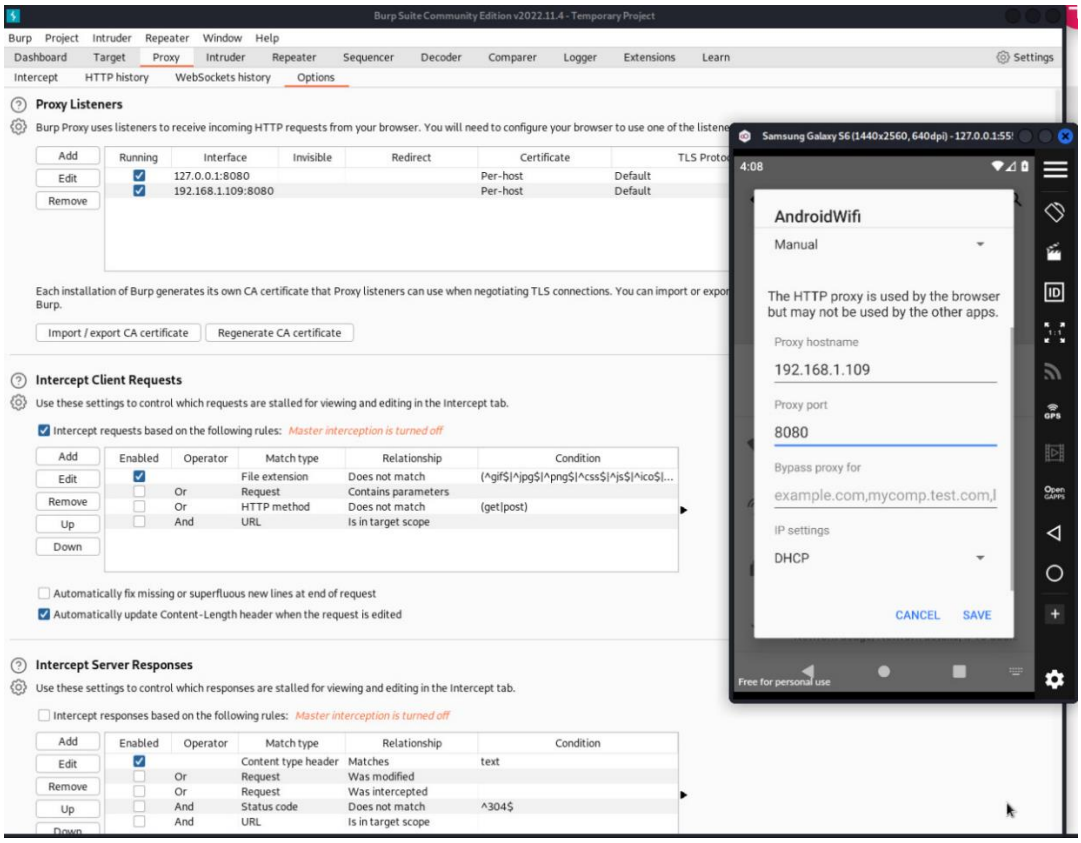
2. Dynamic analysis with burp suite:

After running the genymotion emulator, we will download the apk file into it.

And install burp suite tool:

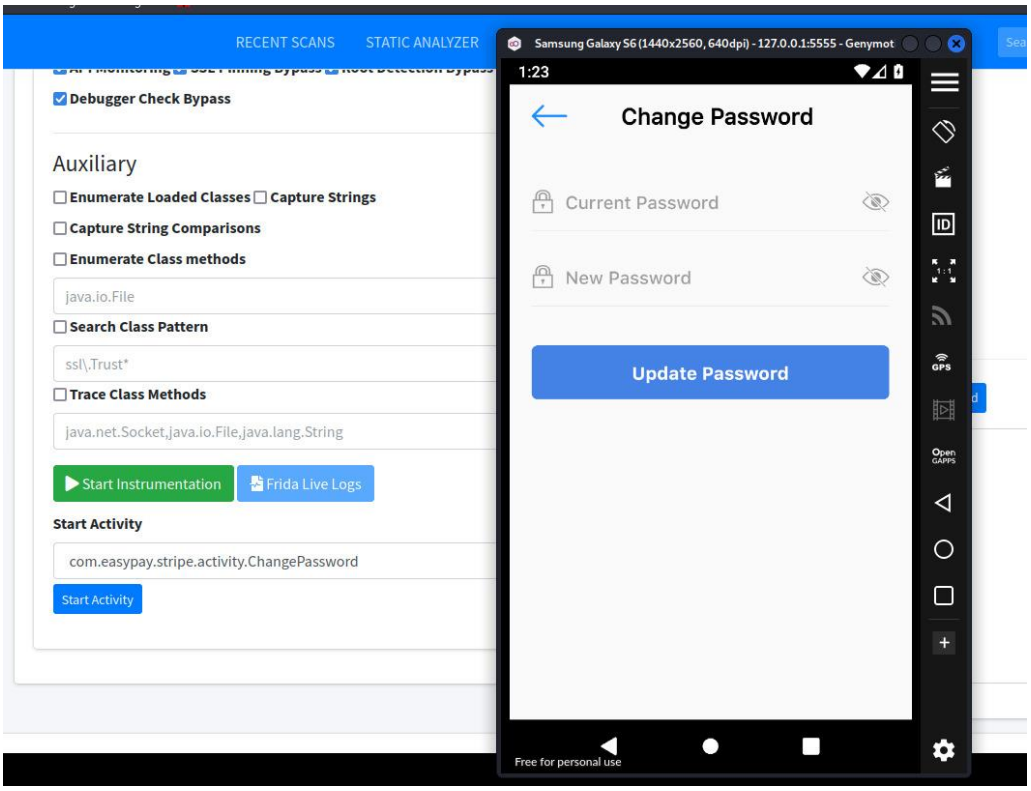
```
┌──(joxavy㉿kali)-[~]
└─$ sudo apt install burpsuite
[sudo] password for joxavy:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  android-libboringssl faraday-client libarmadillo10 libcharls2 libgdal29 libgeos3.10.1 libicu71:i386
  libigdgmm11 libodbc1 libodbccr2 libpython3.9-dev libqhull8.0 libtbb2 libvpx6 odbcinst odbcinst1debian2
  python-mpltoolkits.basemap-data python3-deprecation python3-llvmlite python3-pyproj python3-pyshp python3.9
  python3.9-dev python3.9-minimal
Use 'sudo apt autoremove' to remove them.
The following packages will be upgraded:
  burpsuite
1 upgraded, 0 newly installed, 0 to remove and 1420 not upgraded.
Need to get 230 MB of archives.
After this operation, 42.3 MB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 burpsuite amd64 2022.11.4-0kali1 [230 MB]
Ign:1 http://kali.download/kali kali-rolling/main amd64 burpsuite amd64 2022.11.4-0kali1
Get:1 http://kali.download/kali kali-rolling/main amd64 burpsuite amd64 2022.11.4-0kali1 [230 MB]
Fetched 39.0 MB in 7min 31s (86.3 kB/s)
(Reading database ... 307006 files and directories currently installed.)
Preparing to unpack .../burpsuite_2022.11.4-0kali1_amd64.deb ...
Unpacking burpsuite (2022.11.4-0kali1) over (2021.10.2-0kali3) ...
Setting up burpsuite (2022.11.4-0kali1) ...
Processing triggers for kali-menu (2021.4.2) ...
Scanning processes ...
Scanning processor microcode ...
Scanning linux images ...
```

And then, set up a proxy inside burpsuite, by going to proxy, options and add, give it the following ip address: 192.168.1.109 with the port number:8080

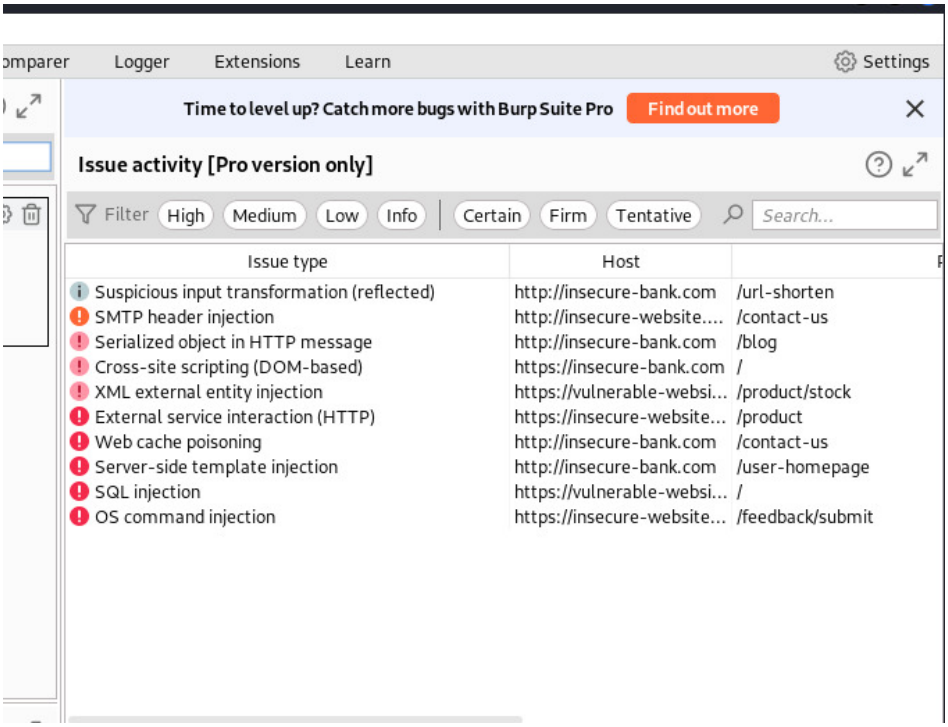


Do the same thing inside the emulator, in order to connect it to the burpsuite proxy.

After that, we will launch an activity:

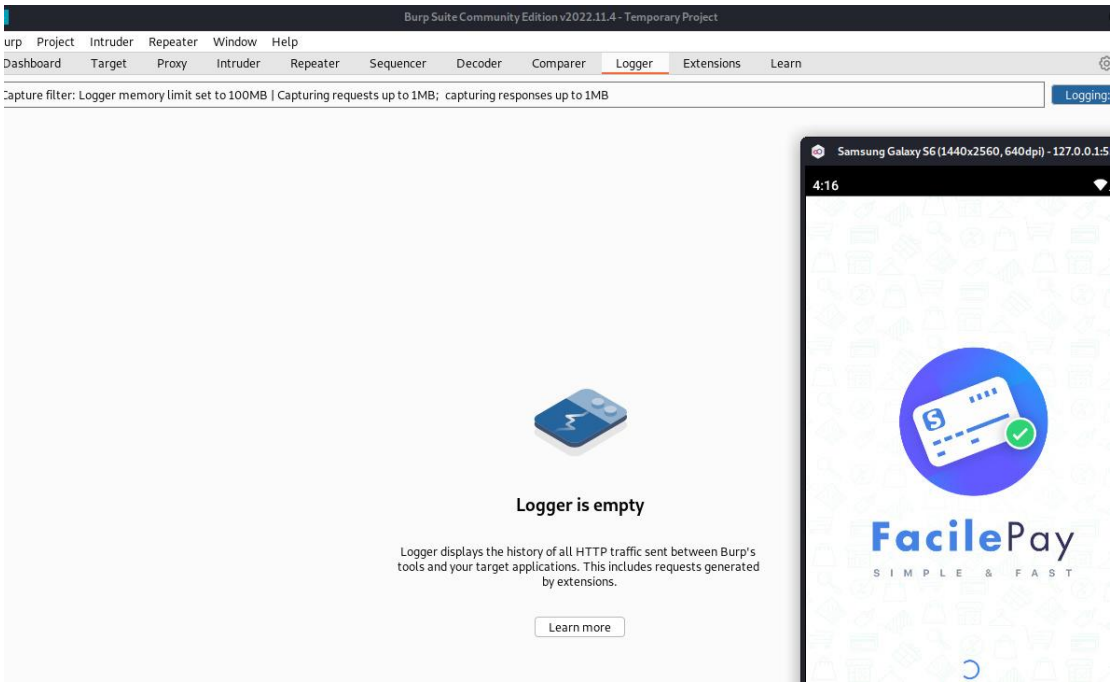


Now each issue inside that activity will be shown on burpsuite:

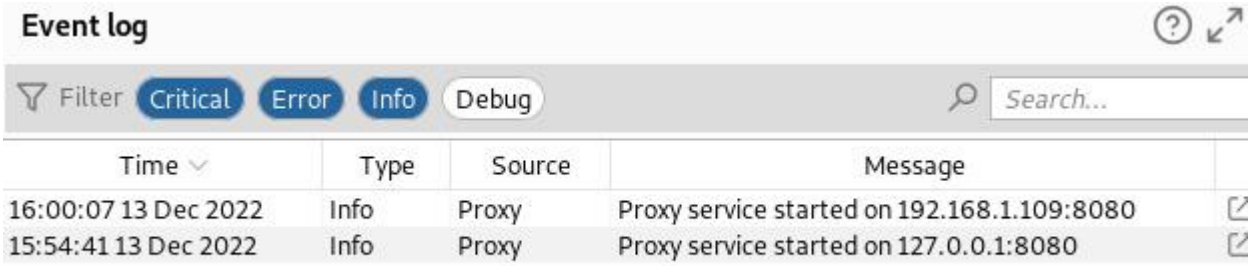


it detects all of the issues.

And it also saves the cache data of the different sites used by the activity inside the cache memory:



It shows inside its event log, information regarding the proxy and the device it is running on.



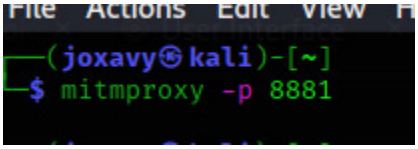
3. Dynamic analysis with MITMproxy:

Mitmproxy starts as a regular HTTP proxy by default and listens on `http://localhost:8080`.

First things first, we will launch the mobile emulator without connecting it to the proxy and check the mitm.it site to see if the device is connected or not.

The result shows that the device's traffic is not passing through the proxy.

I will now check the command line interface of the mitmproxy by typing the following command:

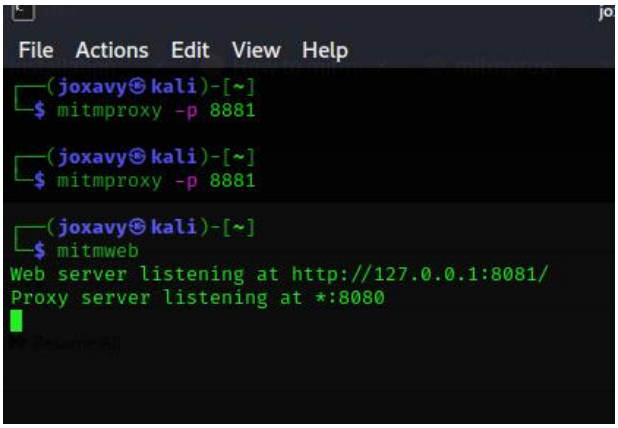


The result:

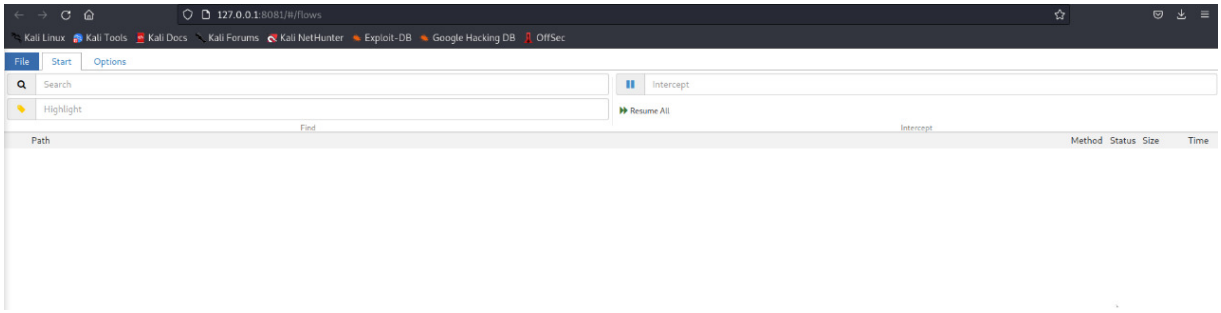
As you can see, no traffic is being registered.

Same thing appears on the GUI interface of the proxy:

Running it:

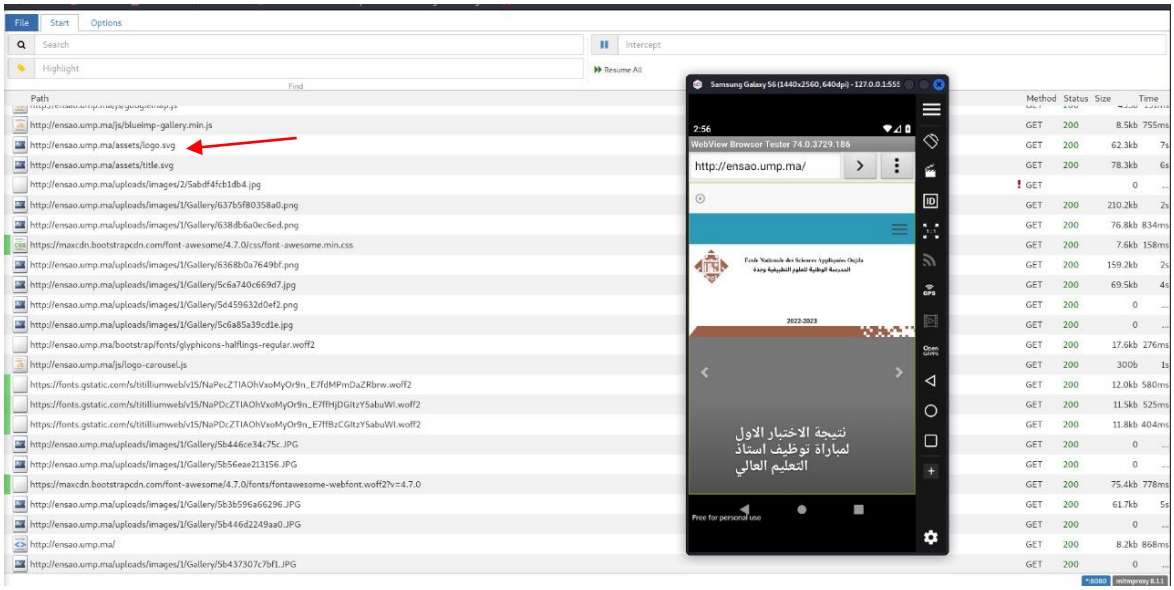


Checking the interface on Browser:



I will then configure the proxy on the mobile emulator by using my device's Ip address as server name and the port number 8080 that the proxy server is listening to as a port.

Then I will navigate through the phone to different websites as seen below:

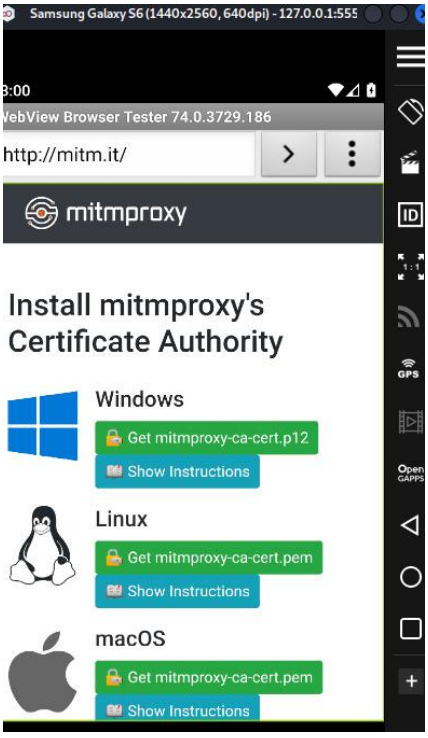


As you can see above, the traffic is being caught on the mitmproxy's interface.

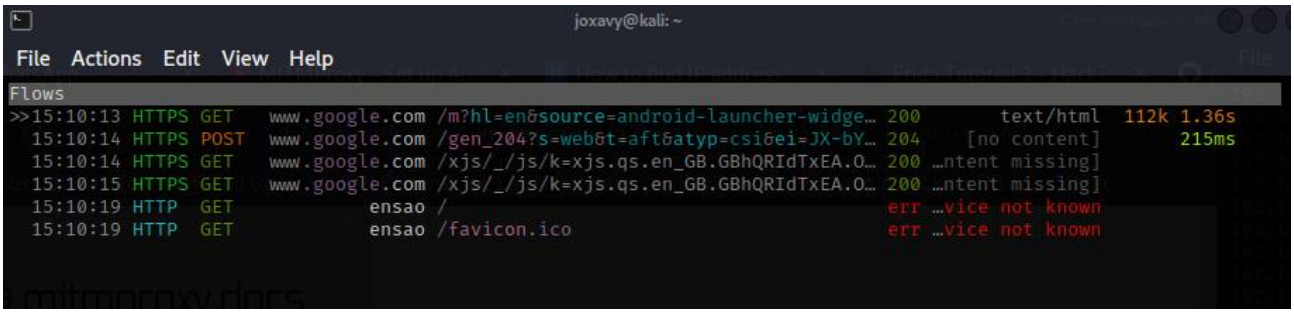
They're also being caught under the command:

```
┌──(joxavy㉿kali)-[~]
└─$ mitmweb
Web server listening at http://127.0.0.1:8081/
Proxy server listening at *:8080
192.168.1.109:44722: client connect
192.168.1.109:44722: client disconnect
192.168.1.109:44724: client connect
192.168.1.109:44724: client disconnect
192.168.1.109:44726: client connect
192.168.1.109:44726: client disconnect
192.168.1.109:44728: client connect
192.168.1.109:44728: server connect www.google.com:443 (142.250.200.196:443)
192.168.1.109:44730: client connect
192.168.1.109:44730: server connect fonts.gstatic.com:443 (142.250.201.3:443)
192.168.1.109:44732: client connect
192.168.1.109:44734: client connect
192.168.1.109:44736: client connect
192.168.1.109:44734: server connect i.ytimg.com:443 (172.217.19.150:443)
192.168.1.109:44736: server connect play.google.com:443 (142.250.200.206:443)
192.168.1.109:44738: client connect
192.168.1.109:44740: client connect
192.168.1.109:44742: client connect
192.168.1.109:44744: client connect
192.168.1.109:44746: client connect
192.168.1.109:44748: client connect
192.168.1.109:44732: server connect www.gstatic.com:443 (172.217.21.3:443)
192.168.1.109:44742: server connect i.ytimg.com:443 (172.217.19.54:443)
192.168.1.109:44750: client connect
192.168.1.109:44738: server connect www.gstatic.com:443 (172.217.21.3:443)
192.168.1.109:44750: server connect play.google.com:443 (142.250.200.206:443)
192.168.1.109:44740: server connect www.gstatic.com:443 (172.217.21.3:443)
192.168.1.109:44752: client connect
192.168.1.109:44752: error establishing server connection: [Errno -5] No address associated with hostname
192.168.1.109:44752: client disconnect
192.168.1.109:44756: client connect
192.168.1.109:44758: client connect
192.168.1.109:44758: error establishing server connection: [Errno -2] Name or service not known
192.168.1.109:44758: client disconnect
192.168.1.109:44756: client disconnect
192.168.1.109:44756: error establishing server connection: client disconnected
192.168.1.109:44760: client connect
```

Checking the same website again that is mitm.it to see if the device is connected:



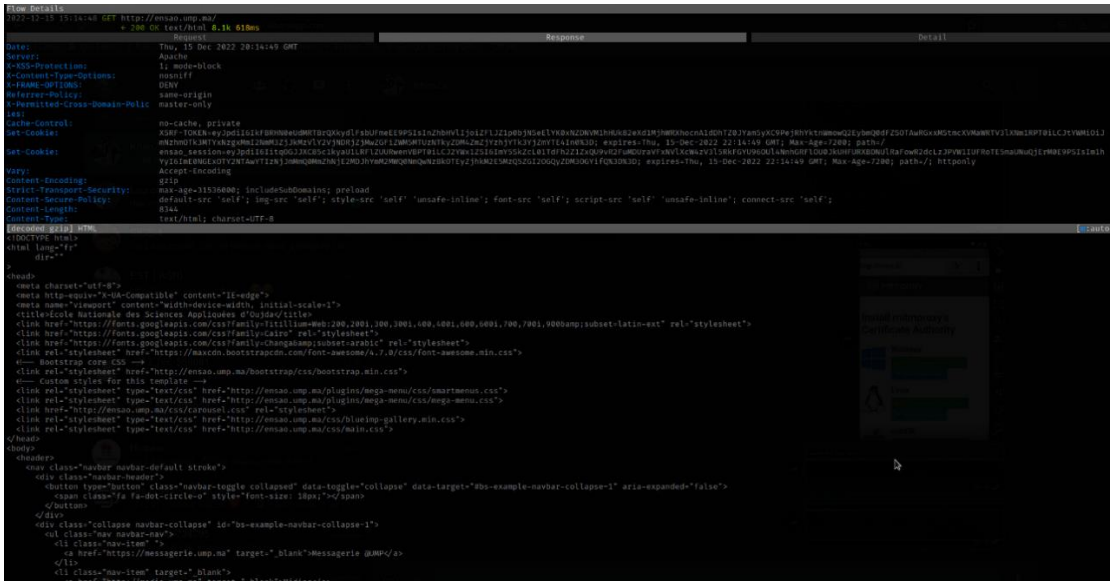
Information is also displayed in the command line interface of mitmproxy:



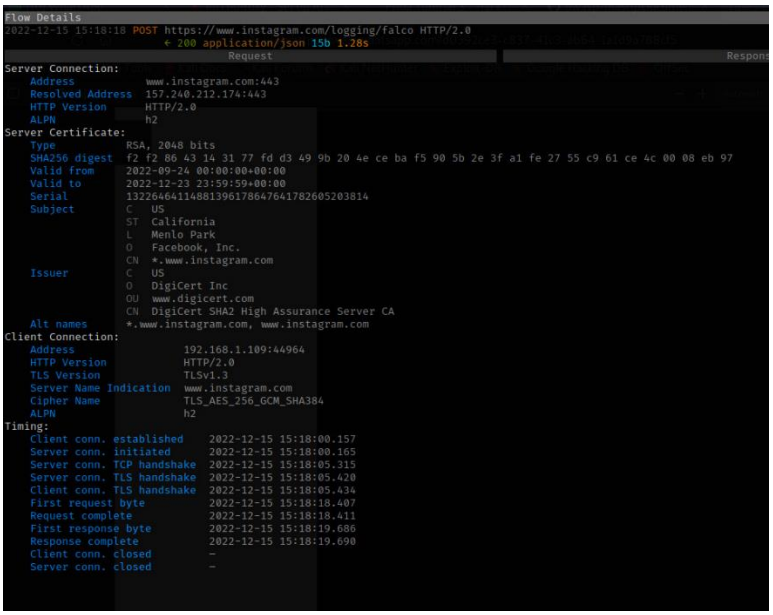
By clicking on any link, we get diverse information:

Request info:

Response info:

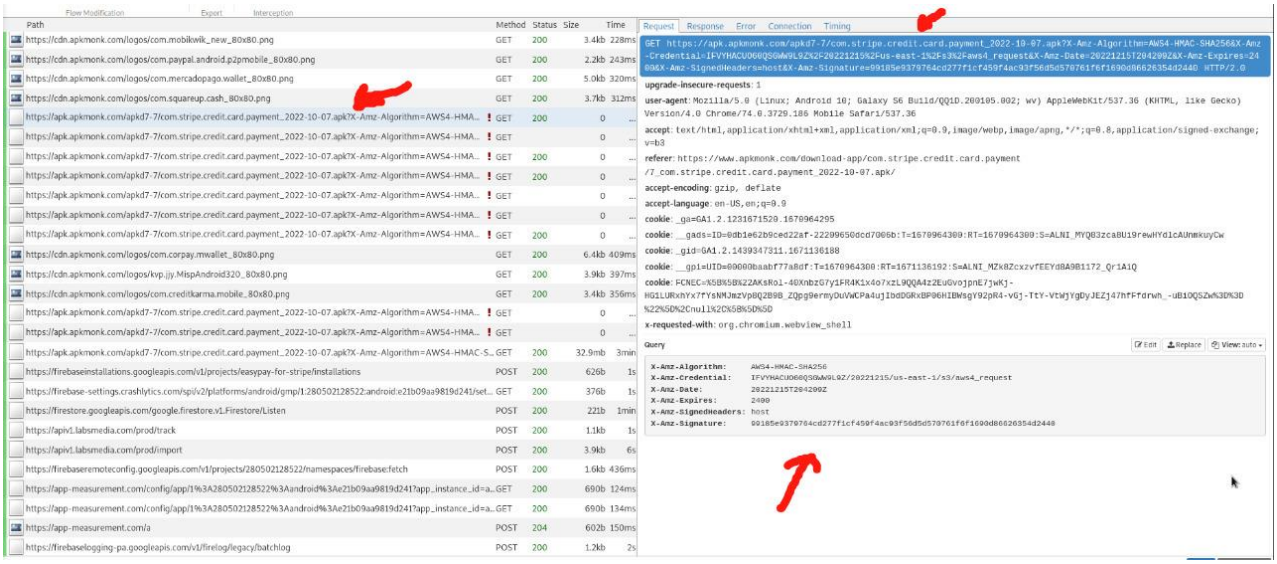


Detail info:



After connecting my mobile device (not the emulator) to the mitmproxy, and trying to navigate through websites, I get the following result on the proxy command line interface:

After launching the app on the emulator, the information also gets displayed in the proxy:



It includes different information about it including the algorithm used, the certificate date, the expiry date, etc.

4. Dynamic analysis with Frida:

Frida is a free and open-source dynamic code instrumentation toolkit that lets you execute snippets of JavaScript into native apps on Android and iOS (as well as on other platforms).

First, let's check which emulators are connected:

```
┌──(joxavy㉿kali)-[~]
└─$ adb devices
List of devices attached
127.0.0.1:5555  device
```

Installing Frida and Frida tools:

```
┌──(joxavy㉿kali)-[~]
└─$ pip install frida
Defaulting to user installation because normal site-packages is not writeable
Collecting frida
  Downloading frida-16.0.8-cp37-abi3-manylinux_2_5_x86_64.manylinux1_x86_64.whl (19.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.2/19.2 MB 588.1 kB/s eta 0:00:00
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from frida) (65.5.0)
Installing collected packages: frida
Successfully installed frida-16.0.8

┌──(joxavy㉿kali)-[~]
└─$ pip install frida-tools
Defaulting to user installation because normal site-packages is not writeable
Collecting frida-tools
  Downloading frida-tools-12.0.4.tar.gz (177 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 177.6/177.6 kB 466.0 kB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: frida<17.0.0,⩾16.0.0 in ./.local/lib/python3.10/site-packages (from frida-tools) (
16.0.8)
Requirement already satisfied: pygments<3.0.0,⩾2.0.2 in /usr/lib/python3/dist-packages (from frida-tools) (2.7.1)
Requirement already satisfied: prompt-toolkit<4.0.0,⩾2.0.0 in ./.local/lib/python3.10/site-packages (from frida-t
ools) (3.0.19)
Requirement already satisfied: colorama<1.0.0,⩾0.2.7 in /usr/lib/python3/dist-packages (from frida-tools) (0.4.4)
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from frida<17.0.0,⩾16.0.0→frida-too
ls) (65.5.0)
Requirement already satisfied: wcwidth in /usr/lib/python3/dist-packages (from prompt-toolkit<4.0.0,⩾2.0.0→frida
-tools) (0.1.9)
Building wheels for collected packages: frida-tools
  Building wheel for frida-tools (pyproject.toml) ... done
  Created wheel for frida-tools: filename=frida_tools-12.0.4-py3-none-any.whl size=187228 sha256=2f6516192ab59042d
ab471b04947c2a0a70d3de69e24a944e7562f3010dc2b1a
  Stored in directory: /home/joxavy/.cache/pip/wheels/1b/5f/95/ff066ec78a422d8577494f72e0e5b0b06f7d442896b7bcfaf4
Successfully built frida-tools
Installing collected packages: frida-tools
  WARNING: The scripts frida, frida-apk, frida-compile, frida-create, frida-discover, frida-join, frida-kill, frid
a-ls, frida-ls-devices, frida-ps, frida-pull, frida-push, frida-rm and frida-trace are installed in '/home/joxavy/
.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location
.
Successfully installed frida-tools-12.0.4

┌──(joxavy㉿kali)-[~]
```

Entering the adb shell:

```
┌──(joxavy㉿kali)-[~]
└─$ adb shell
vbox86p:/ # cd /data
vbox86p:/data # ls
adb           app-private  drm        misc_ce     property                  system_ce   vendor_ce
anr           app-staging  gsi        misc_de     resource-cache            system_de   vendor_de
apex          backup       local      nfc         rollback                  tombstones
app           bootchart    lost+found ota         rollback-observer         unencrypted
app-asec      cache        media      ota_package server_configurable_flags user
app-ephemeral dalvik-cache mediadrm   per_boot    ss                        user_de
app-lib       data         misc       preloads    system                    vendor
```

Placing ourselves inside the sdcard/downloads in order to retrieve the compressed file for the Frida server and copying it into the new path “data/local/tmp”:

```
2|vbox86p:/ # cd sdcard
vbox86p:/sdcard # ls
Alarms Android DCIM Download Movies Music Notifications Pictures Podcasts Ringtones
vbox86p:/sdcard # cd Download
vbox86p:/sdcard/Download # ls
frida-server-16.0.8-android-x86.xz mitmproxy-9.0.1-linux.tar.gz
vbox86p:/sdcard/Download # tar -xf frida-server-16.0.8-android-x86.xz
tar: invalid tar format
1|vbox86p:/sdcard/Download # unxz frida-server-16.0.8-android-x86.xz
vbox86p:/sdcard/Download # ls
frida-server-16.0.8-android-x86 mitmproxy-9.0.1-linux.tar.gz
vbox86p:/sdcard/Download # mv frida-server-16.0.8-android-x86 /data/local/tmp
vbox86p:/sdcard/Download # cd /data/local/tmp
vbox86p:/data/local/tmp # ls
```

Renaming it and giving it execution permissions, then running it:

```
vbox86p:/data/local/tmp # mv frida-server-16.0.8-android-x86  frida-server
vbox86p:/data/local/tmp # ./frida-server
/system/bin/sh: ./frida-server: can't execute: Permission denied
126|vbox86p:/data/local/tmp # chmod 755 frida-server
vbox86p:/data/local/tmp # ./frida-server
```

With Frida-server running, you should now be able to get a list of running processes with the following command :

```
┌──(joxavy㉿kali)-[~]
└─$ frida-ps -Uai
 PID  Name                Identifier
4
3769  Amaze               com.amaze.filemanager
2619  FacilePay           com.stripe.credit.card.payment
3635  Files               com.android.documentsui
2459  Gallery             com.android.gallery3d
2195  Phone               com.android.dialer
3103  Search              com.android.quicksearchbox
2569  Settings            com.android.settings
3407  WebView Shell       org.chromium.webview_shell
   -  Calendar            com.android.calendar
   -  Camera              com.android.camera2
   -  Clock               com.android.deskclock
   -  Contacts            com.android.contacts
   -  Custom Locale       com.android.customlocale2
   -  Development Settings  com.android.development_settings
   -  Email               com.android.email
   -  Messaging           com.android.messaging
   -  Superuser           com.genymotion.superuser
```

This will show the names and identifiers of all apps, if they are currently running it will also show their PIDs. As w can see amaze has a PID while Clock doesn't.

The PID of my application (FacilePay)in this list is 2619(it can change)

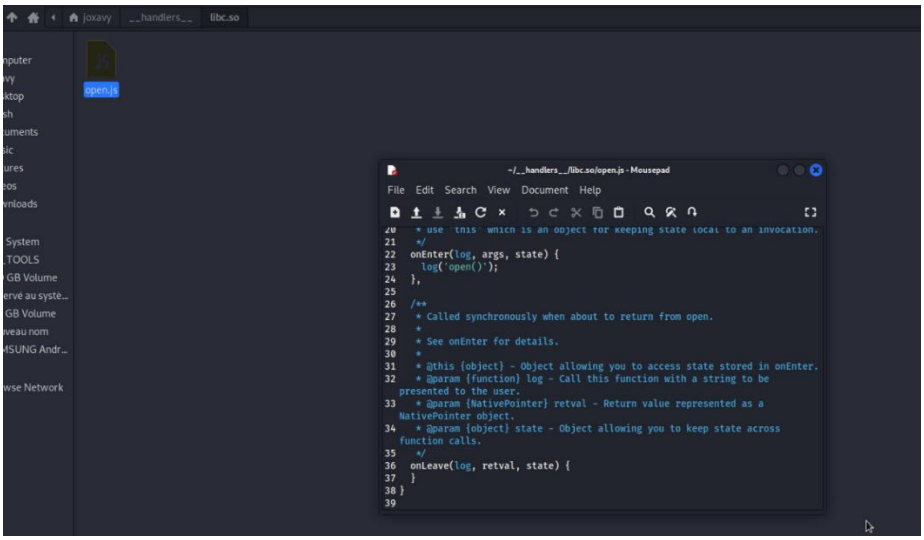
Running the following command with the PID of the app:

```
┌──(joxavy㉿kali)-[~]
└─$ frida-trace -U 2619 -i "open"
Instrumenting...
open: Auto-generated handler at "/home/joxavy/__handlers__/libc.so/open.js"
Started tracing 1 function. Press Ctrl+C to stop.
```

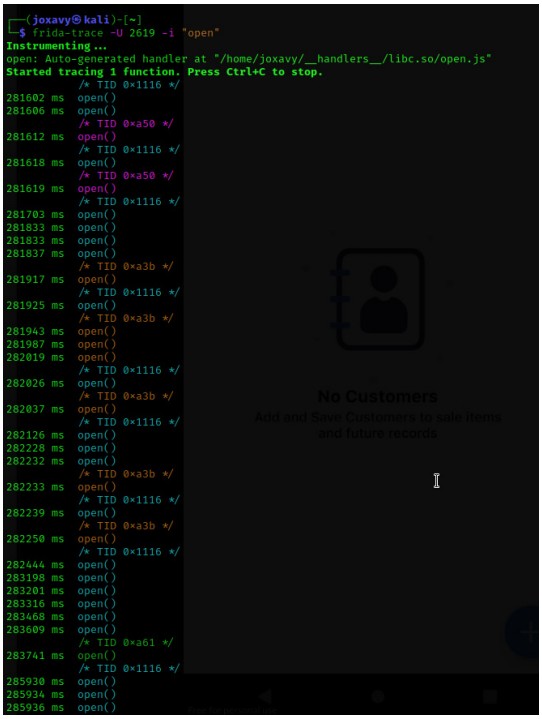
This generates a little JavaScript in _handlers_/libc.so/open.js, which Frida injects into the process.

The script traces all calls to the open function in libc.so.

Here is the script that has been generated:



The tracing result:



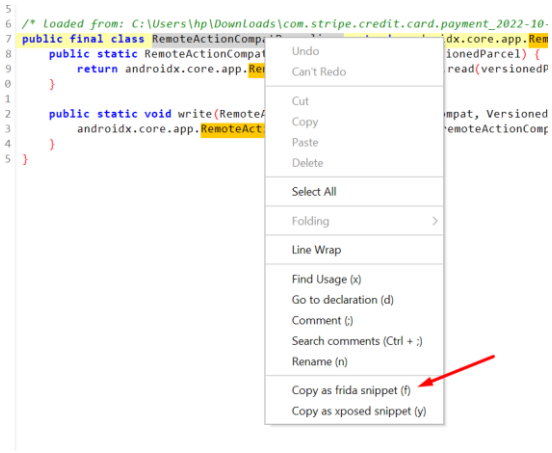
Use the Frida CLI tool (frida) to work with Frida interactively. It hooks into a process and gives you a command line interface to Frida's API.

```
┌──(root💀kali)-[/home/joxavy]
└─# frida -U 2619
    ____
   / _  |   Frida 16.0.8 - A world-class dynamic instrumentation toolkit
  | (_| |
   > _  |   Commands:
  /_/ |_|       help      → Displays the help system
  . . . .       object?   → Display information about 'object'
  . . . .       exit/quit → Exit
  . . . .
  . . . .   More info at https://frida.re/docs/home/
  . . . .
  . . . .   Connected to A50 (id=127.0.0.1:5555)

[A50::PID::2619 ]→
```

The JADX decompiler can generate Frida snippets through its graphical code browser.

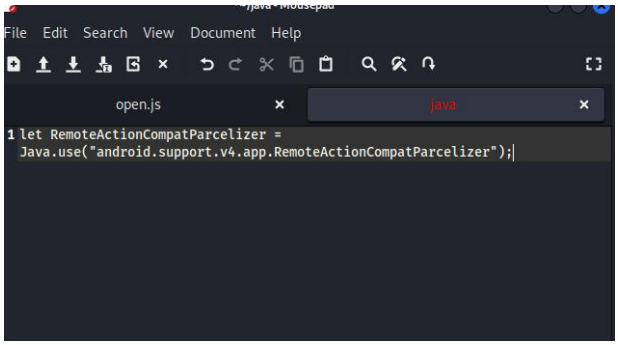
To use this feature, open the APK or DEX with jadx-gui, browse to the target method, right click the method name, and select "Copy as frida snippet (f)".

Copying the snippet:



The above steps place the following output in the pasteboard, which you can then paste in a JavaScript file and feed into frida -U -l.

Pasting the result into a JS file:

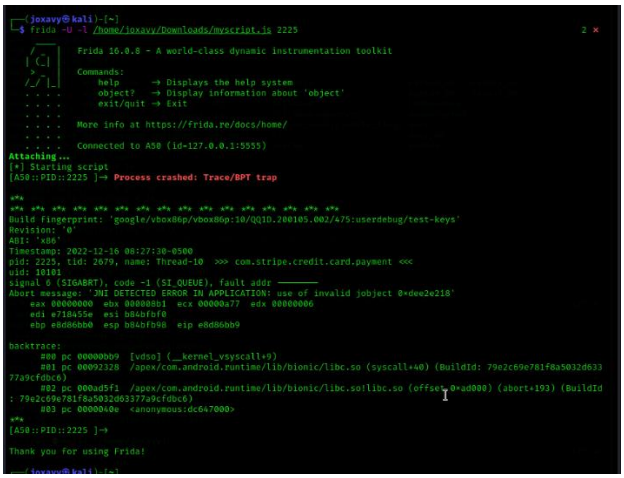


Feeding it into Frida:

Frida also lets you search for and work with instantiated objects that are on the heap. The following script searches for instances of android.view.View objects and calls their toString method. The result is printed to the console:

```
setImmediate(function() {
    console.log("[*] Starting script");
    Java.perform(function () {
        Java.choose("android.view.View", {
            "onMatch":function(instance){
                console.log("[*] Instance found: " + instance.toString());
            };    "onComplete":function() {
                console.log("[*] Finished heap search") }    }): }):}):
```

Feeding it into frida:



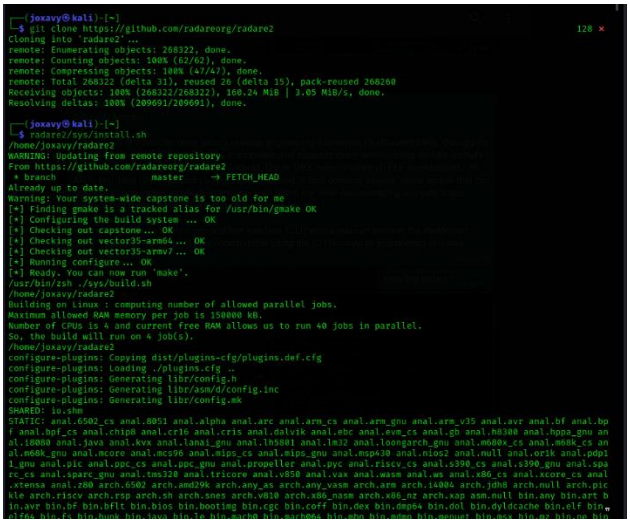
5. Dynamic analysis with radare2:

radare2 (r2) is a popular open-source reverse engineering framework for disassembling, debugging, patching and analyzing binaries that is scriptable and supports many architectures and file formats including Android and iOS apps.

For Android, Dalvik DEX (odex, multidex), ELF (executables, .so, ART) and Java (JNI and Java classes) are supported.

It also contains several useful scripts that can help you during mobile application analysis as it offers low level disassembling and safe static analysis that comes in handy when traditional tools fail.

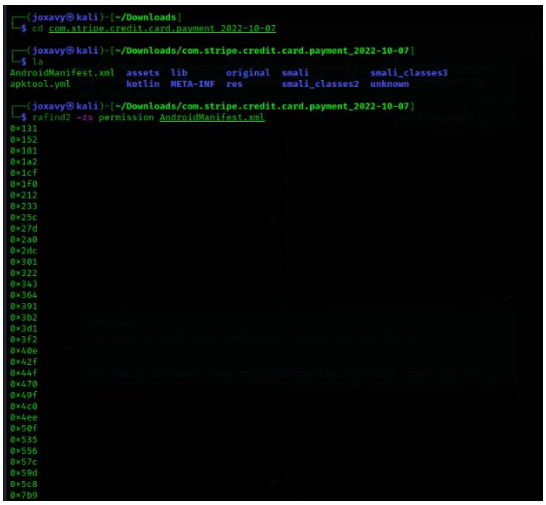
installing radare2:



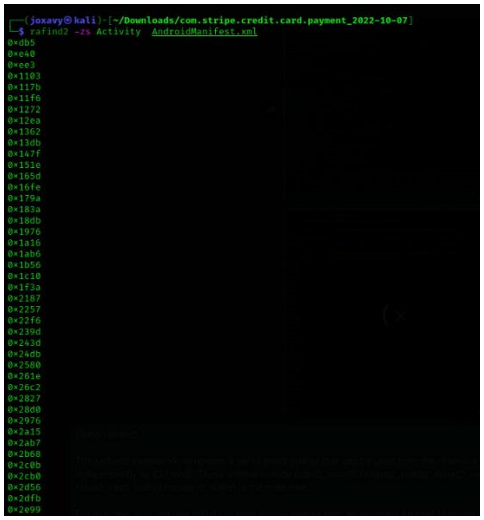
The radare2 framework comprises a set of small utilities that can be used from the r2 shell or independently as CLI tools. These utilities include rabin2, rasm2, rahash2, radiff2, rafind2, ragg2, rarun2, rax2, and of course r2, which is the main one.

For example, you can use rafind2 to read strings directly from an encoded Android Manifest (AndroidManifest.xml):

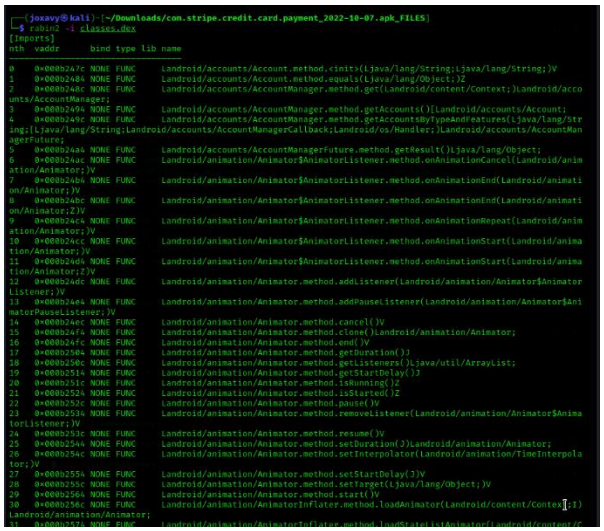
For permission:



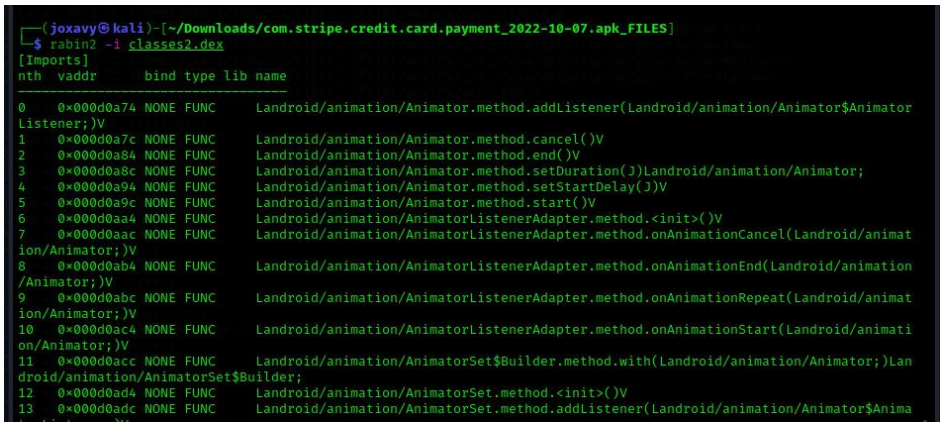
For activities:



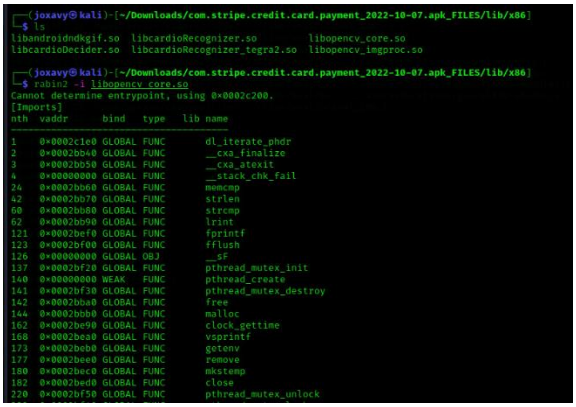
Using rabin2 to get information about a binary file for classes.dex :



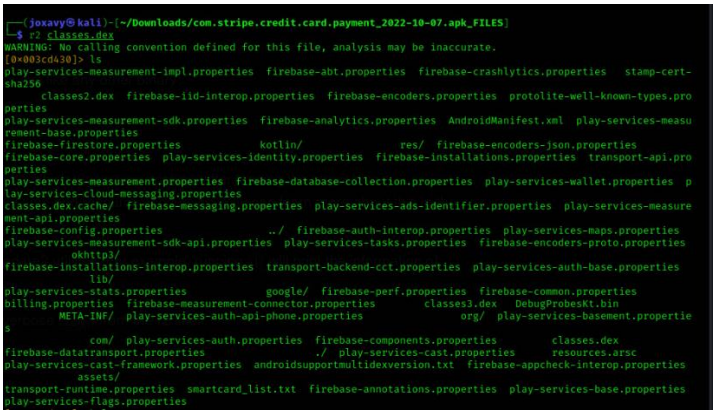
For classes2.dex:



For libopencv library:

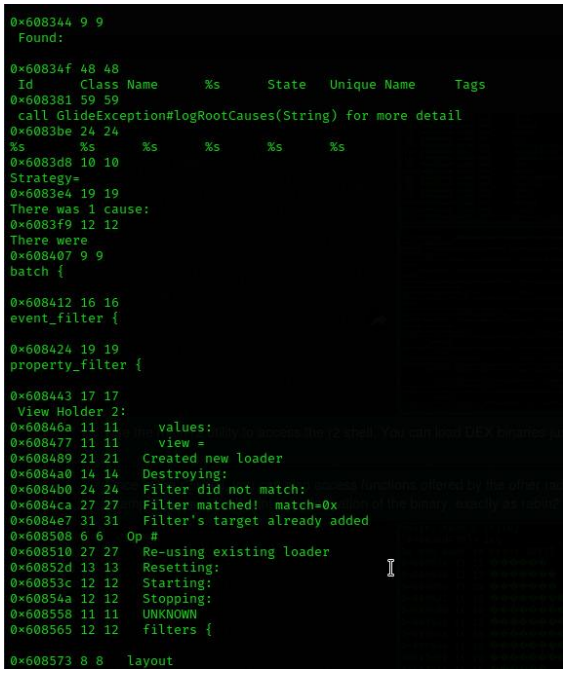


Use the main r2 utility to access the r2 shell. You can load DEX binaries just like any other binary:



Once in the r2 shell, you can also access functions offered by the other radare2 utilities.

To print all the strings, use rabin2 -Z or the command iz (or the less verbose izq) from the r2 shell.



Most of the time you can append special options to your commands such as q to make the command less verbose or j to give the output in JSON format (use ~{} to prettify the JSON string):

```
┌──(joxavy㉿kali)-[~/Downloads/com.stripe.credit.card.payment_2022-10-07.apk_FILES]
└─$ r2 classes2.dex
WARNING: No calling convention defined for this file, analysis may be inaccurate.
[0×0028c5d4]> izj~{}
Do you want to print 339562 lines? (y/N)
[0×0028c5d4]> izj~{}
Do you want to print 339562 lines? (y/N) y
[
  {
    "vaddr": 4990526,
    "paddr": 4990526,
    "ordinal": 2,
    "size": 12,
    "length": 12,
    "section": "data",
    "type": "ascii",
    "string": "\u0000\u0001\u0000\u0000\u0001\u0001\u0001\u0000"
  },
  {
    "vaddr": 4990546,
    "paddr": 4990546,
    "ordinal": 3,
    "size": 12,
    "length": 12,
    "section": "data",
    "type": "ascii",
    "string": "\u0000\u0001\u0000\u0000\u0001\u0001\u0001\u0000"
  },
  {
    "vaddr": 4990565,
    "paddr": 4990565,
    "ordinal": 4,
    "size": 12,
    "length": 12,
    "section": "data",
    "type": "ascii",
    "string": "\u0000\u0001\u0000\u0000\u0001\u0001\u0001\u0000"
  },
  {
    "vaddr": 4990585,
    "paddr": 4990585,
    "ordinal": 5,
    "size": 12,
    "length": 12,
    "section": "data",
    "type": "ascii",
    "string": "\u0000\u0001\u0000\u0000\u0001\u0001\u0001\xc0"
  },
  {
    "vaddr": 4990604,
    "paddr": 4990604,
```

You can print the class names and their methods with the r2 command ic (information classes):

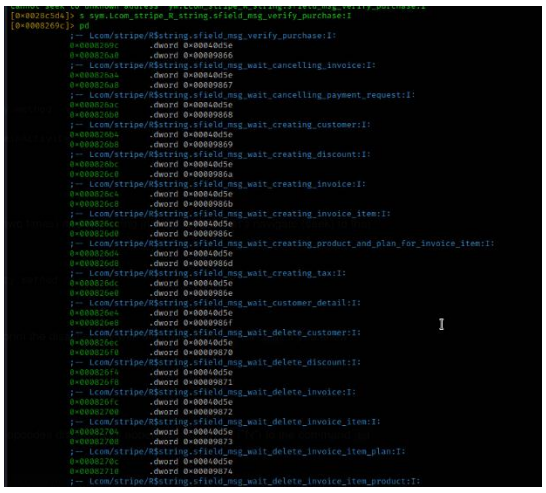
```
[0×0028c5d4]>
[0×0028c5d4]> ic
Do you want to print 58050 lines? (y/N) y
0×0013e13c [0×001759a4 - 0×00175a2e]    138 class 0 Lcom/google/android/material/floatingactionbutton/ExtendedFlo
atingActionButton$c :: Landroid/animation/AnimatorListenerAdapter;
0×001759a4 method 0 C    Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$c.method.
<init>(Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton;Lcom/google/android/materia
l/floatingactionbutton/f;Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$j;)V
0×001759c4 method 1 p    Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$c.method.
onAnimationCancel(Landroid/animation/Animator;)V
0×001759e8 method 2 p    Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$c.method.
onAnimationEnd(Landroid/animation/Animator;)V
0×00175a1c method 3 p    Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$c.method.
onAnimationStart(Landroid/animation/Animator;)V
0×0013e15c [0×00175a40 - 0×00175adc]    156 class 1 Lcom/google/android/material/floatingactionbutton/ExtendedFlo
atingActionButton$d :: Landroid/util/Property;
0×00175a88 method 0 C    Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$d.method.
<init>(Ljava/lang/Class;Ljava/lang/String;)V
0×00175a40 method 1 p    Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$d.method.
a(Landroid/view/View;)Ljava/lang/Float;
0×00175aa0 method 2 p    Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$d.method.
b(Landroid/view/View;Ljava/lang/Float;)V
0×00175a68 method 3 pbh  Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$d.method.
get(Ljava/lang/Object;)Ljava/lang/Object;
0×00175acc method 4 pbh  Lcom/google/android/material/floatingactionbutton/ExtendedFloatingActionButton$d.method.
```

You can print the imported methods with the r2 command ii (information imports):

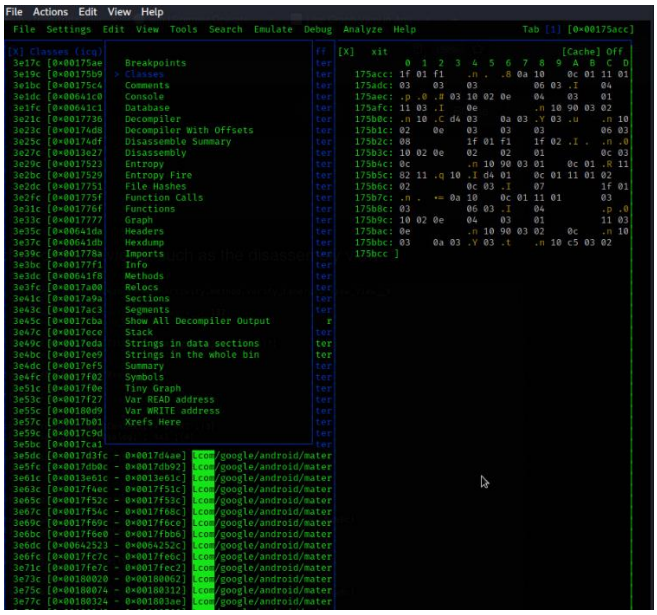
```
[0×0028c5d4]> ii
Do you want to print 4963 lines? (y/N) y
[Imports]
nth  vaddr      bind type lib name
―――――――――――――――――――――――――――――――
0    0×000d0a74 NONE FUNC     Landroid/animation/Animator.method.addListener(Landroid/animation/Animator$Animator
Listener;)V
1    0×000d0a7c NONE FUNC     Landroid/animation/Animator.method.cancel()V
2    0×000d0a84 NONE FUNC     Landroid/animation/Animator.method.end()V
3    0×000d0a8c NONE FUNC     Landroid/animation/Animator.method.setDuration(J)Landroid/animation/Animator;
4    0×000d0a94 NONE FUNC     Landroid/animation/Animator.method.setStartDelay(J)V
5    0×000d0a9c NONE FUNC     Landroid/animation/Animator.method.start()V
6    0×000d0aa4 NONE FUNC     Landroid/animation/AnimatorListenerAdapter.method.<init>()V
7    0×000d0aac NONE FUNC     Landroid/animation/AnimatorListenerAdapter.method.onAnimationCancel(Landroid/animat
ion/Animator;)V
8    0×000d0ab4 NONE FUNC     Landroid/animation/AnimatorListenerAdapter.method.onAnimationEnd(Landroid/animation
/Animator;)V
9    0×000d0abc NONE FUNC     Landroid/animation/AnimatorListenerAdapter.method.onAnimationRepeat(Landroid/animat
ion/Animator;)V
10   0×000d0ac4 NONE FUNC     Landroid/animation/AnimatorListenerAdapter.method.onAnimationStart(Landroid/animati
on/Animator;)V
11   0×000d0acc NONE FUNC     Landroid/animation/AnimatorSet$Builder.method.with(Landroid/animation/Animator;)Lan
droid/animation/AnimatorSet$Builder;
12   0×000d0ad4 NONE FUNC     Landroid/animation/AnimatorSet.method.<init>()V
13   0×000d0adc NONE FUNC     Landroid/animation/AnimatorSet.method.addListener(Landroid/animation/Animator$Anima
torListener;)V
14   0×000d0ae4 NONE FUNC     Landroid/animation/AnimatorSet.method.cancel()V
15   0×000d0aec NONE FUNC     Landroid/animation/AnimatorSet.method.end()V
16   0×000d0af4 NONE FUNC     Landroid/animation/AnimatorSet.method.isRunning()Z
17   0×000d0afc NONE FUNC     Landroid/animation/AnimatorSet.method.play(Landroid/animation/Animator;)Landroid/an
imation/AnimatorSet$Builder;
18   0×000d0b04 NONE FUNC     Landroid/animation/AnimatorSet.method.playSequentially([Landroid/animation/Animator
```

- A common approach when inspecting a binary is to search for something, navigate to it and visualize it in order to interpret the code.
- One of the ways to find something using radare2 is by filtering the output of specific commands, i.e. to grep them using ~ plus a keyword (~+ for case-insensitive).
- For example, we might know that the app is verifying something, we can inspect all radare2 flags and see where we find something related to "verify".
- When loading a file, radare2 tags everything it's able to find. These tagged names or references are called flags. You can access them via the command f.
- In this case we will grep the flags using the keyword "verify":

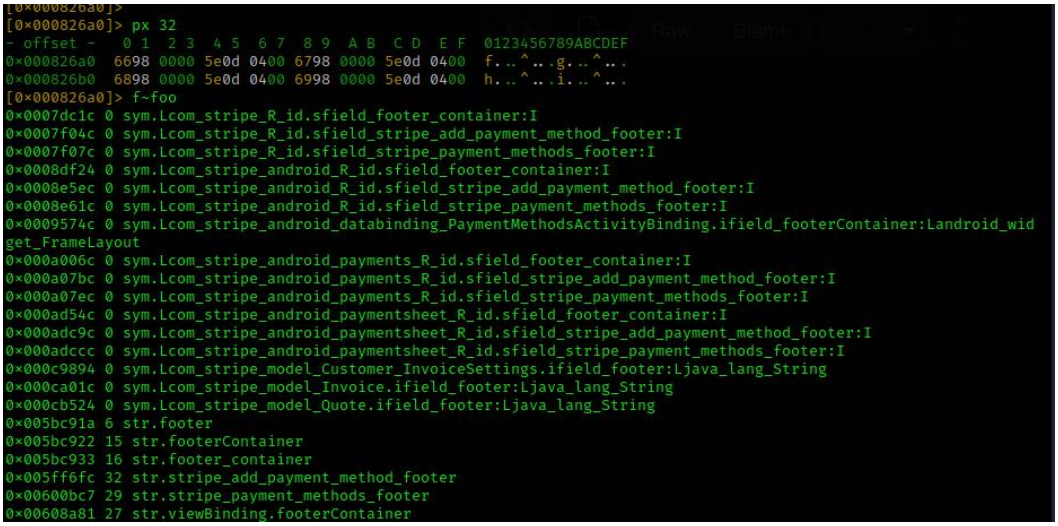
Let's navigate (seek) to the method “0x0008269c”by using its flag and print the disassembly with the command pd:



Instead of just printing the disassembly to the console you may want to enter the so-called Visual Mode by typing V:



print 32 byte hex dump current block and filter flags with ~grep (same as |grep):



V. Android Code Quality and Build Settings:

Going to test some of the android code quality methods.

1. Making Sure That the App is Properly Signed (MSTG-CODE-1):

i. Static Analysis:

Make sure that the release build has been signed via both the v1 and v2 schemes for Android 7.0 (API level 24) and above and via all the three schemes for Android 9 (API level 28) and above, and that the code-signing certificate in the APK belongs to the developer.

We will verify this by using the following tool:

Downloading it:

```
┌──(joxavy㉿kali)-[~/Downloads]
└─$ sudo apt-get -y install apksigner

[sudo] password for joxavy:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  faraday-client libarmadillo10 libcharls2 libgdal29 libgeos3.10.1 libicu71:i386 libig
  python3-deprecation python3-llvmlite python3-pyproj python3-pyshp python3.9 python3.
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libapksig-java
```

Using it to check for signature:

```
┌──(joxavy㉿kali)-[~/Downloads]
└─$ apksigner verify --verbose com.stripe.credit.card.payment_2022-10-07.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Verifies
Verified using v1 scheme (JAR signing): true
Verified using v2 scheme (APK Signature Scheme v2): true
Verified using v3 scheme (APK Signature Scheme v3): true
Verified using v4 scheme (APK Signature Scheme v4): false
Verified for SourceStamp: false
Number of signers: 1
WARNING: META-INF/3ds2sdk_release.kotlin_module not protected by signature. Unauthorized mo
WARNING: META-INF/SOSegmentSDK_release.kotlin_module not protected by signature. Unauthoriz
```

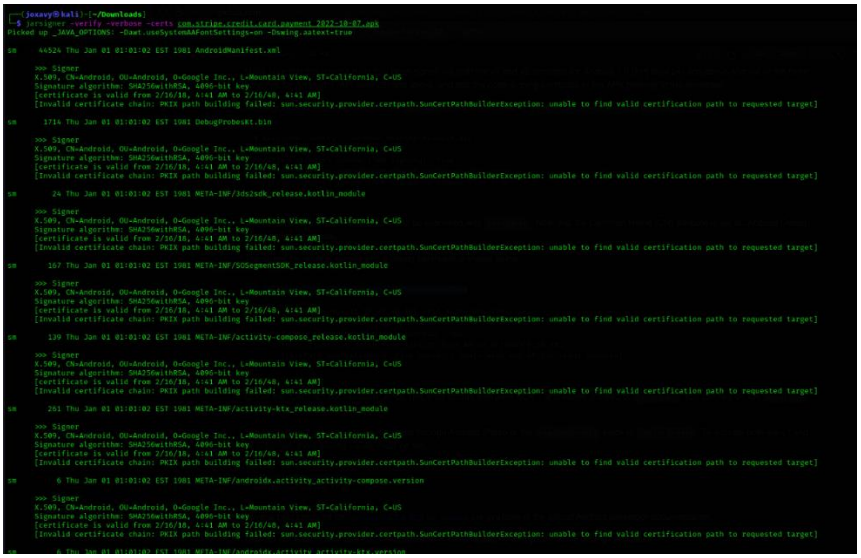
As seen in the result, the apk has been signed with v1, v2 and v3. But not v4.

It also displays the files which are not protected by signature:

```
WARNING: META-INF/3ds2sdk_release.kotlin_module not protected by signature. Unauthorized modifications to this JAR entry will not be detected. Delete or move the entry outside of META-INF/.
WARNING: META-INF/SOSegmentSDK_release.kotlin_module not protected by signature. Unauthorized modifications to this JAR entry will not be detected. Delete or move the entry outside of META-INF/.
WARNING: META-INF/activity-compose_release.kotlin_module not protected by signature. Unauthorized modifications to this JAR entry will not be detected. Delete or move the entry outside of META-INF/.
WARNING: META-INF/activity-ktx_release.kotlin_module not protected by signature. Unauthorized modifications to this JAR entry will not be detected. Delete or move the entry outside of META-INF/.
WARNING: META-INF/androidx.activity_activity-compose.version not protected by signature. Unauthorized modifications to this JAR entry will not be detected. Delete or move the entry outside of META-INF/.
WARNING: META-INF/androidx.activity_activity-ktx.version not protected by signature. Unauthorized modifications to this JAR entry will not be detected. Delete or move the entry outside of META-INF/.
```

The contents of the signing certificate can be examined with jarsigner. Note that the Common Name (CN) attribute is set to "Android Debug" in the debug certificate.

The output for an APK signed with a debug certificate is shown below:



ii. Dynamic Analysis:

Static analysis should be used to verify the APK signature.

2. Testing Whether the App is Debuggable (MSTG-CODE-2):

The android: debuggable attribute in the Application element that is defined in the Android manifest determines whether the app can be debugged or not.

i. Static Analysis:

Check AndroidManifest.xml to determine whether the android: debuggable attribute has been set and to find the attribute's value.

We open the AndroidManifest.xml file and check the <application> part:

We can see that it doesn't contain the android debuggable attribute.

You can use aapt tool from the Android SDK with the following command line to quickly check if the android: debuggable="true" directive is present, by using the following command:

If the command prints 1 then the directive is present, but in our case, it returns 0, as you can see 1 is not found. Therefore, the android debuggable directive is not present.

ii. Dynamic Analysis:

First, we need to determine the package name of the apk, by using this command:

The output gives that: com.stripe.credit.card.payement

Is the name of package, so we will use it.

adb can be used to determine whether an application is debuggable.

Use the following command:

```
┌──(joxavy㉿kali)-[~/Downloads]
└─$ adb shell dumpsys package com.stripe.credit.card.payment | grep -c "DEBUGGABLE"
0

┌──(joxavy㉿kali)-[~/Downloads]
└─$
```

If the command prints a number superior to zero then the application have the debug flag, if not, it doesn't. in our case, the command prints 0, which means it indeed doesn't contain the debug flag.

3. Testing for Debugging Code and Verbose Error Logging (MSTG-CODE-4):

Overview:

StrictMode is a developer tool for detecting violations, e.g., accidental disk or network access on the application's main thread. It can also be used to check for good coding practices, such as implementing performant code.

i. Static Analysis:

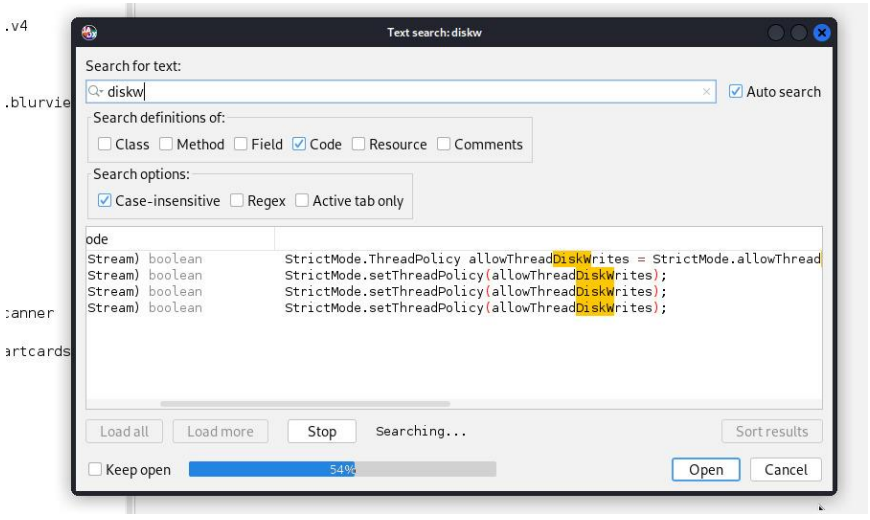
To determine whether StrictMode is enabled, you can look for the StrictMode.setThreadPolicy or StrictMode.setVmPolicy methods. Most likely, they will be in the onCreate method.

the detection methods for the thread policy are

- detectDiskWrites()
- detectDiskReads()
- detectNetwork()

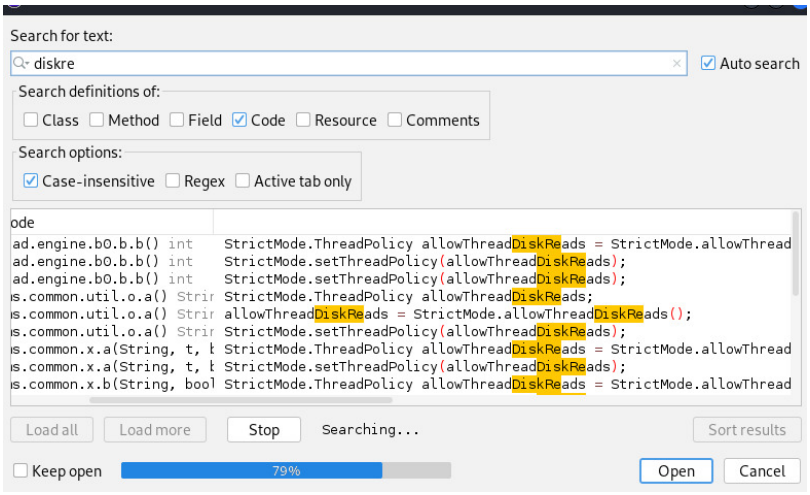
we will check if these methods exist by opening the apk inside the Jadx tool and searching for them one by one:

search for detectDiskWrites:

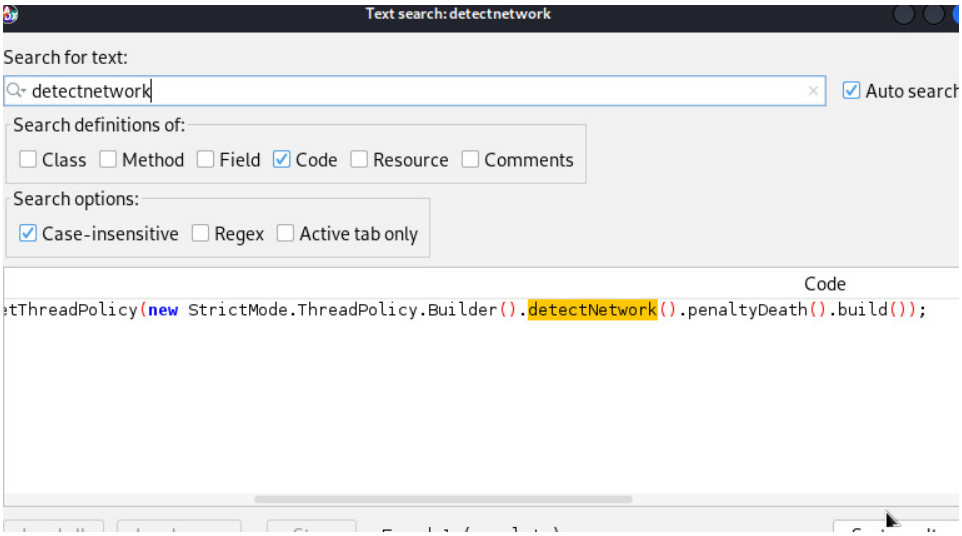


As we can see, it exists in our apk.

Search for detectDiskReads:



Search for DetectNetwrok:

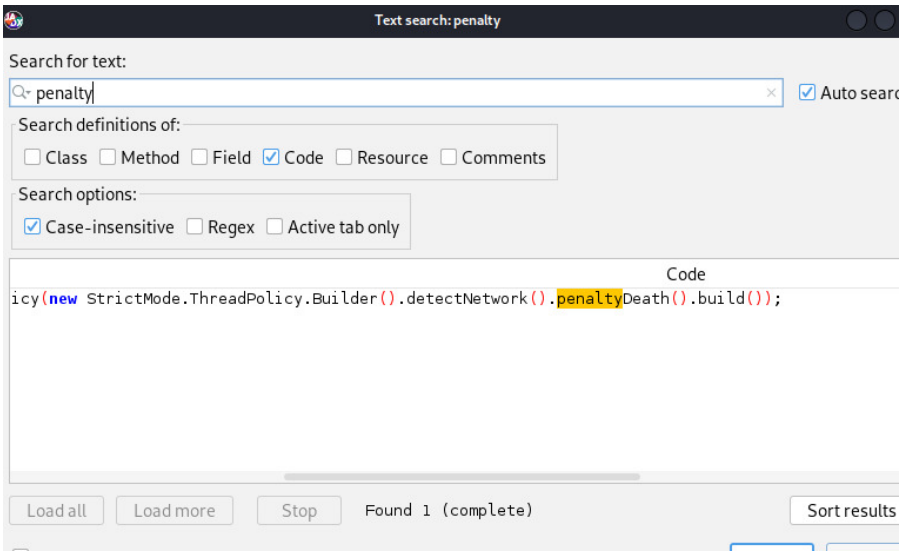


We conclude that the detection methods for the thread policies are indeed included in the app.

The penalties for thread policy violation are:

- penaltyLog() // Logs a message to LogCat
- penaltyDeath() // Crashes application, runs at the end of all enabled penalties
- penaltyDialog() // Shows a dialog

we will also check for the existence of the penalties with Jadx.



Only the penalty of death exists.

ii. Dynamic Analysis

There are several ways of detecting StrictMode; the best choice depends on how the policies' roles are implemented. They include

Logcat,

a warning dialog,

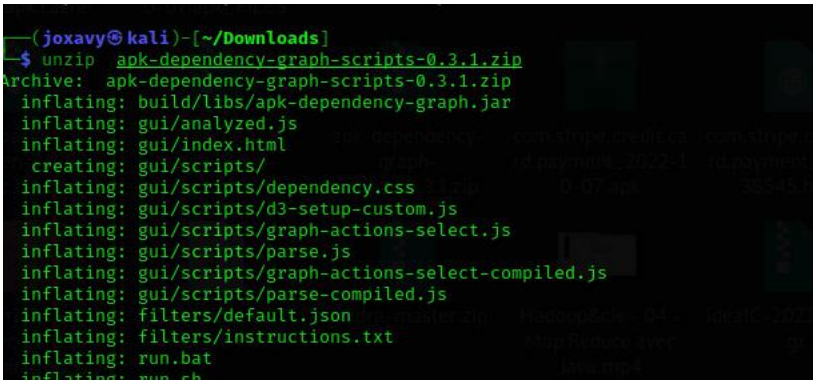
application crash.

VI. Graphing the apk dependencies:

Class dependency visualizer. Only apk file is needed.

Class coupling is one of the significant code metrics that shows how easy is to change, maintain and test the code. This tool helps to view whole picture of the project

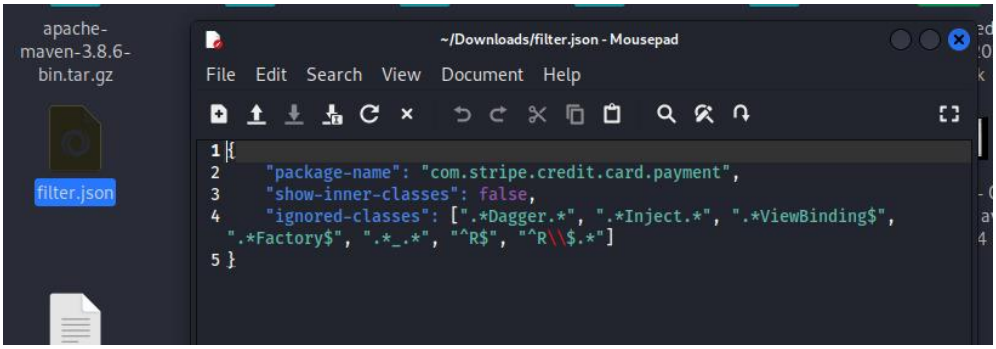
Unzipping the file:



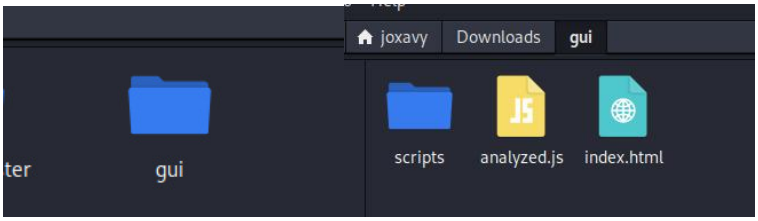
Running the tool:

```
┌──(root💀kali)-[/home/joxavy/Downloads]
└─# ./run.sh com.stripe.credit.card.payment_2022-10-07.apk filter.json
Baksmaling classes.dex ...
Baksmaling classes2.dex ...
Baksmaling classes3.dex ...
Analyzing dependencies ...
Success! Now open gui/index.html in your browser.
```

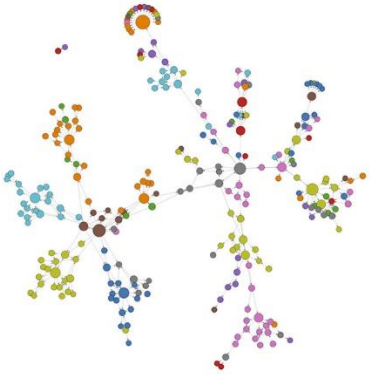
The filter.json file:



A file called index.html is created that contains the graph.



The output:



It’s a good architecture with low class coupling.