

# WRITE UP – MOBILE PENTEST

Nama: Cindy

NIM: 2602107000

## 1. Competitor Mail v2 APK.zip

### a. Rooted device and emulator detection -> bypassed detection

- Exploitable status: **EXPLOITABLE**
- Tools: JADX-Gui
- Information of vulnerability and exploit

Saya melakukan analisis source code terlebih dahulu menggunakan JADX-Gui. Seperti biasa yang saya temukan bahwa root check biasanya terdapat pada MainActivity.

```
19     public void onCreate(Bundle bundle) {  
20         super.onCreate(bundle);  
21         setContentView(R.layout.activity_main);  
23         this.rootCheckService = new RootCheckService(this);  
24         this.emulatorCheckService = new EmulatorCheckService();  
25     }  
  
    /* JADX INFO: Access modifiers changed from: protected */  
    @Override // androidx.appcompat.app.AppCompatActivity, androidx.fragment.app.FragmentActivity, android.app.Activity  
28     public void onStart() {  
29         super.onStart();  
34         startActivity(new Intent(this, LoginActivity.class));  
36         finish();  
    }
```

Pada line 19 terdapat class rootCheckService() dan line 24 terdapat class EmulatorCheckService() yang dimana kedua class tersebut seharusnya untuk mendeteksi device root dan pemakaian emulator saat mengakses aplikasi. Namun, implementasi kedua class ini hanya inisialisasi saja dan tidak diterapkan dalam MainActivity. Sebagai akibatnya, deteksi emulator dan deteksi root tidak berfungsi dengan baik dalam aplikasi ini. Dari hal tersebut, mengakibatkan aplikasi HakuBank dapat diakses melalui perangkat yang sudah di root dan emulator.

### b. Send E-mail message -> duplicate each sent e-mail message and send to [user@email.com](mailto:user@email.com)

- Exploitable status: **EXPLOITABLE**
- Tools:
  - ☐ BurpSuite
  - ☐ Android Visual Studio
  - ☐ Frida
- Information of vulnerability and exploit

Agar CodeShare frida dapat dijalankan, maka hal pertama yang dilakukan adalah dengan meng-execute frida-server.

```
emu64xa:/data/local/tmp # ./frida-server-16.3.3-android-x86_64
```

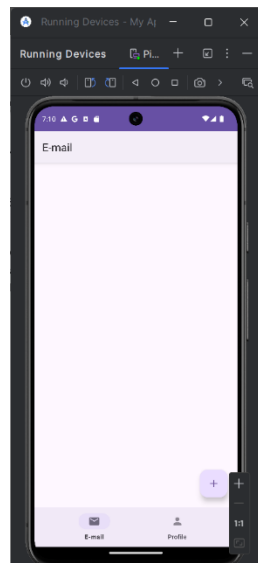
Setelah itu, untuk mengetahui nama aplikasi Competitor Mail saya menjalankan command `frida-ps -U -a` yang nantinya nama aplikasi tersebut dapat di execute CodeShare frida.

```
C:\Users\cindy>frida-ps -U -a
-----
PID   Name                Identifier
-----
11766 Chrome              com.android.chrome
9859  Competitor Mail     com.climawan.comp6844001_uas.competitormail
10895 Files              com.google.android.documentsui
1624  Google              com.google.android.googlequicksearchbox
1624  Google              com.google.android.googlequicksearchbox
3069  Google Play Store   com.android.vending
11636 Magisk              com.topjohnwu.magisk
1671  Messages            com.google.android.apps.messaging
1100  SIM Toolkit          com.android.stk
1101  Settings            com.android.settings
```

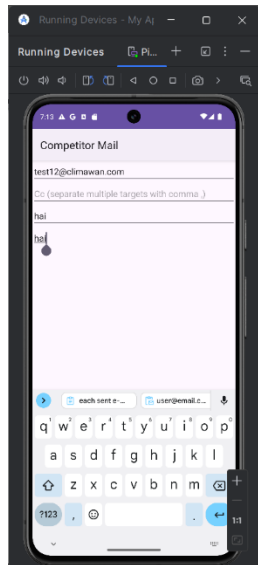
Setelah mendapatkan nama aplikasi Competitor Mail, maka dapat menjalankan CodeShare frida dengan menjalankan command **frida --codeshare**

**KaiserBloo/ssl-and-root-bypass -f**

**com.climawan.comp6844001\_uas.competitormail -U**. Tujuannya menjalankan command tersebut adalah agar dapat mengintercept aplikasi target yang dimana aplikasi tersebut terdapat SSL Pinning. Kegunaan SSL Pinning adalah untuk mempersulit penyerang untuk meretas aplikasi dan menghindari pihak ke-3.

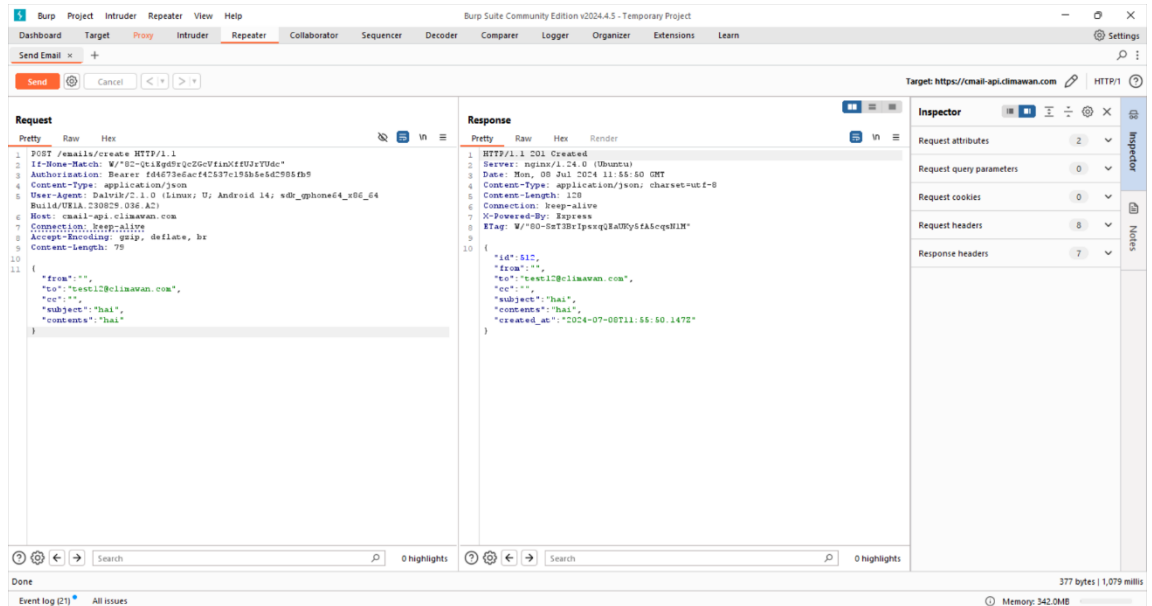


Pada gambar di atas merupakan tampilan awal dari aplikasi Competitor Mail yang dimana saya akan mencoba mencari kerentanan dari mengirimkan pesan melalui email.



Saya mencoba mengirimkan email kepada email [test12@climawan.com](mailto:test12@climawan.com) yang berisikan pesan “hai”. Namun sebelum mengirimkan pesan ke email tersebut, saya mengintercept terlebih dahulu melalui BurpSuite.

Lalu setelah BurpSuite jalan, saya mendapati response seperti di bawah ini melalui BurpSuite.



Setelah mendapat record dari BurpSuite, terdapat kata “from:” yang belum diisi. Hal ini dapat memanipulasi identitas pengirim yang sebenarnya. Selain itu, saat email terkirim tidak ada autentikasi yang dimana autentikasi tersebut harusnya ada dan hanya berlaku dalam 1x send email yang berbentuk token. Sehingga penyerang tidak melakukan spam email / mengirim email yang berulang-ulang dalam waktu yang sama.

c. Usage Shared Preference

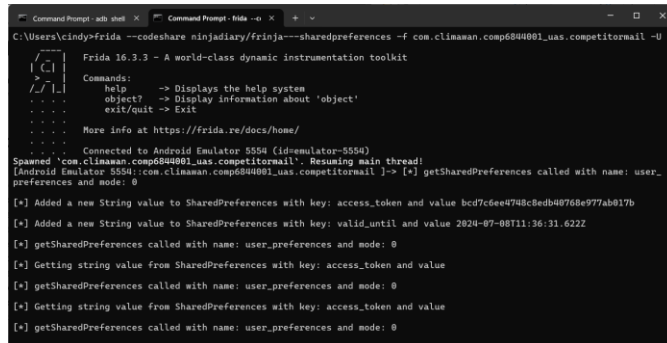
- Exploitable status: **EXPLOITABLE**
- Tools: Frida
- Information of vulnerability and exploit

Usage Shared Preference adalah tujuannya untuk mendapatkan cache yang berupa access token dari login yang telah tersimpan pada aplikasi, namun access token hanya berlaku jika user yang sebenarnya tidak logout karena setiap kali user logout dan login, access token akan berubah.

Setelah itu saya mencoba menjalankan CodeShare frida yang dimana dapat menampilkan cache yang tersimpan pada aplikasi.

Command yang digunakan adalah **frida --codeshare ninjadiary/frinja---sharedpreferences -f com.climawan.comp6844001\_uas.competitormail -U**.

Setelah command tersebut dijalankan, hasilnya seperti pada di bawah ini.



```
C:\Users\cindy>frida --codeshare ninjadiary/frinja---sharedpreferences -f com.climawan.comp6844001_uas.competitormail -U
Frida 16.3.3 - 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 Android Emulator 5554 (id:emulator-5554)
[*] Spawned 'com.climawan.comp6844001_uas.competitormail'. Resuming main thread!
[Android Emulator 5554: com.climawan.comp6844001_uas.competitormail ]-> [*] getSharedPreferences called with name: user_preferences and mode: 0

[*] Added a new String value to SharedPreferences with key: access_token and value bcd7c6ee4748c8edb40768e977ab017b
[*] Added a new String value to SharedPreferences with key: valid_until and value 2024-07-08T11:36:31.622Z
[*] getSharedPreferences called with name: user_preferences and mode: 0
[*] Getting string value from SharedPreferences with key: access_token and value
[*] getSharedPreferences called with name: user_preferences and mode: 0
[*] Getting string value from SharedPreferences with key: access_token and value
[*] getSharedPreferences called with name: user_preferences and mode: 0
```

Dapat disimpulkan dari screenshot tersebut adalah access\_token yang tersimpan pada aplikasi adalah

```
[*] Added a new String value to SharedPreferences with key: access_token and value bcd7c6ee4748c8edb40768e977ab017b
```

Kerentanan ini dapat di exploit menggunakan BurpSuite pada HTTP Header dengan menambahkan Authorization: \*access token\*. Hal ini dapat masuk ke akun orang lain tanpa meminta username dan password selagi user yang sebenarnya tidak melakukan logout dan access token tersebut tidak terenkripsi.

d. Inject a forged intent

- Exploitable content: **NOT EXPLOITABLE**
- Tools: JADX-Gui
- Information of vulnerability and exploit

Pada kode di bawah dicek **android:exported="false"** pada komponen Activity yang dimana menandakan bahwa komponen tersebut tidak dapat diakses oleh aplikasi lain di luar dari aplikasi yang sama. Hal ini penting untuk menjaga keamanan aplikasi karena mengurangi risiko serangan dari penyerang eksternal yang mencoba untuk memanipulasi komponen yang terbuka.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:compileSdkVersion="34" android:compileSdkVersionCodename="14">
  <uses-sdk android:minSdkVersion="33" android:targetSdkVersion="34"/>
  <uses-permission android:name="android.permission.INTERNET"/>
  <permission android:name="com.climawan.comp6844001.us.competitorwall.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION" android:protectionLevel="signature"/>
  <uses-permission android:name="com.climawan.comp6844001.us.competitorwall.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/>
  <application android:theme="@style/Theme.CompetitorWallV2" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:allowBackup="true" android:supportRtl="true" and
  <activity android:name="com.climawan.comp6844001.us.competitorwall.auth.password.UpdatePasswordActivity" android:exported="false"/>
  <activity android:label="@string/title_activity_home" android:name="com.climawan.comp6844001.us.competitorwall.home.HomeActivity" android:exported="false"/>
  <activity android:name="com.climawan.comp6844001.us.competitorwall.MainActivity" android:exported="true">
    <intent-filter>
      <action android:name="android.intent.action.MAIN"/>
      <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
  </activity>
  <activity android:name="com.climawan.comp6844001.us.competitorwall.auth.LoginActivity" android:exported="false"/>
  <activity android:name="com.climawan.comp6844001.us.competitorwall.auth.Register.RegisterActivity" android:exported="false"/>
  <activity android:name="com.climawan.comp6844001.us.competitorwall.email.EmailCreateActivity" android:exported="false"/>
  <activity android:name="com.climawan.comp6844001.us.competitorwall.email.EmailShowActivity" android:exported="false"/>
  <provider android:name="androidx.startup.InitializationProvider" android:exported="false" android:authorities="com.climawan.comp6844001.us.competitorwall.androidx-startup">
    <meta-data android:name="androidx.startup.InitializationProvider" android:value="androidx.startup"/>
    <meta-data android:name="androidx.lifecycle.ProcessLifecycleInitializer" android:value="androidx.startup"/>
    <meta-data android:name="androidx.profileinstaller.ProfileInstallerInitializer" android:value="androidx.startup"/>
  </provider>
  <uses-library android:name="androidx.window.extensions" android:required="false"/>
  <uses-library android:name="androidx.window.idlecase" android:required="false"/>
  <receiver android:name="androidx.profileinstaller.ProfileInstallerReceiver" android:permission="android.permission.DUMP" android:enabled="true" android:exported="true" android:directBo
    <intent-filter>
      <action android:name="androidx.profileinstaller.action.INSTALL_PROFILE"/>
    </intent-filter>
    <intent-filter>
      <action android:name="androidx.profileinstaller.action.SKIP_FILE"/>
    </intent-filter>
    <intent-filter>
      <action android:name="androidx.profileinstaller.action.SAVE_PROFILE"/>
    </intent-filter>
    <intent-filter>
      <action android:name="androidx.profileinstaller.action.BENCHMARK_OPERATION"/>
    </intent-filter>
  </receiver>
</application>
</manifest>

```

## 2. HakuBank Source Code

### a. Rooted device detection

```

1 package com.climawan.comp6844001.pertemuan5.hakubank.services;
2 import androidx.appcompat.app.AppCompatActivity;
3
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.util.Log;
7 import java.io.File;
8
9 3 usages new *
10 public class RootDetectorService {
11
12     1 usage new *
13     public boolean isDeviceRooted() {
14         String[] paths = { "/system/app/Superuser.apk", "/sbin/su", "/system/bin/su", "/system/xbin/su" };
15         for (String path : paths) {
16             if (new File(path).exists()) return true;
17         }
18         return false;
19     }
20 }

```

**Loc:**

app\src\main\java\com\climawan\comp6844001\pertemuan5\hakubank\services\RootDetectorService.java

Pada gambar di atas, saya menambahkan file baru pada folder service yang Bernama “RootDetectorService”. Pada file tersebut saya membuat class RootDetectorService yang memiliki function isDeviceRooted(). Kegunaan dari function tersebut adalah untuk memeriksa apakah device telah di root. Biasanya perangkat yang diroot menggunakan kata kunci “su” yang artinya Super User. Jika hal tersebut ditemukan, maka function akan mengembalikan nilai true (perangkat terdeteksi root) dan mengembalikan false jika tidak terdeteksi root.

```
package com.climawan.comp6844001.pertemuan5.hakubank.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Toast;

import com.climawan.comp6844001.pertemuan5.hakubank.R;
import com.climawan.comp6844001.pertemuan5.hakubank.services.RootDetectorService;

import Christopher L.*

public class MainActivity extends AppCompatActivity {

    Christopher L.*

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RootDetectorService rootDetectorService = new RootDetectorService();
        if (rootDetectorService.isDeviceRooted()) {
            Toast.makeText(this, "Root Detected!", Toast.LENGTH_LONG).show();
            finish();
        }
    }

    Christopher L.*

    @Override
    protected void onStart() {
        super.onStart();
    }
}
```

## Loc:

app/src/main/java/com/climawan/comp6844001/pertemuan5/hakubank/activities/MainActivity.java

Pada file MainActivity, saya menambahkan beberapa kode yang dimana ketika activity dimulai maka hal yang dilakukan pertama kali adalah menginisialisasi tampilan menggunakan layout main\_activity. Kemudian membuat objek dari class RootDetectedService dan memanggil function isDeviceRooted() untuk mengecek apakah device di root. Jika terdeteksi adanya root, maka terdapat muncul pop-up “Root Detected!” dan otomatis langsung keluar dari aplikasi tersebut (finish()) agar aplikasi tidak terus berjalan.

## b. Emulator detection

```
package com.climawan.comp6844001.pertemuan5.hakubank.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Toast;

import com.climawan.comp6844001.pertemuan5.hakubank.R;
import com.climawan.comp6844001.pertemuan5.hakubank.services.RootDetectorService;

import Christopher L.*

public class MainActivity extends AppCompatActivity {

    Christopher L.*

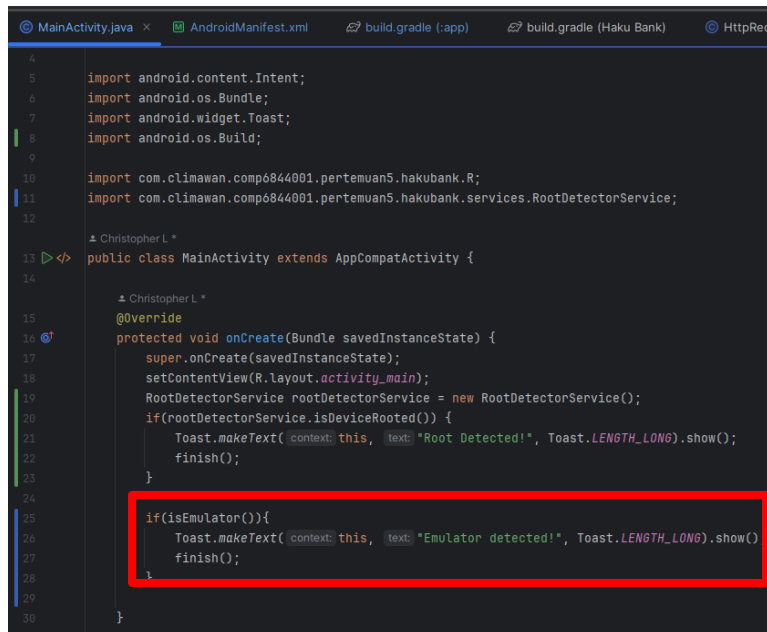
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RootDetectorService rootDetectorService = new RootDetectorService();
        if (rootDetectorService.isDeviceRooted()) {
            Toast.makeText(this, "Root Detected!", Toast.LENGTH_LONG).show();
            finish();
        }
    }

    Christopher L.*

    @Override
    protected void onStart() {
        super.onStart();
    }
}

private boolean isEmulator() {
    return (Build.FINGERPRINT.startsWith("generic")
        || Build.FINGERPRINT.startsWith("unknown")
        || Build.MODEL.contains("google_sdk")
        || Build.MODEL.contains("Emulator")
        || Build.MODEL.contains("Android SDK built for x86")
        || Build.MANUFACTURER.contains("Genymotion")
        || (Build.BRAND.startsWith("generic") && Build.DEVICE.startsWith("generic"))
        || "google_sdk".equals(Build.PRODUCT)
        || System.getProperty("ro.kernel.gemu") != null);
}
```



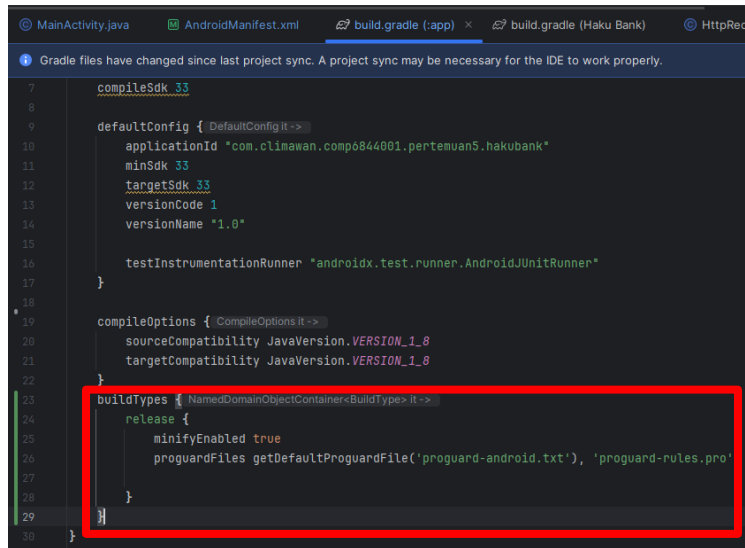
```
4
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.widget.Toast;
8 import android.os.Build;
9
10 import com.climawan.comp6844001.pertemuan5.hakubank.R;
11 import com.climawan.comp6844001.pertemuan5.hakubank.services.RootDetectorService;
12
13 Christopher L *
14 public class MainActivity extends AppCompatActivity {
15
16 Christopher L *
17 @Override
18 protected void onCreate(Bundle savedInstanceState) {
19     super.onCreate(savedInstanceState);
20     setContentView(R.layout.activity_main);
21     RootDetectorService rootDetectorService = new RootDetectorService();
22     if(rootDetectorService.isDeviceRooted()) {
23         Toast.makeText(context, this, "Root Detected!", Toast.LENGTH_LONG).show();
24         finish();
25     }
26
27     if(isEmulator()){
28         Toast.makeText(context, this, "Emulator detected!", Toast.LENGTH_LONG).show();
29         finish();
30     }
31 }
```

### Loc:

app/src/main/java/com/climawan/comp6844001/pertemuan5/hakubank/activities/MainActivity.java

Pada file MainActivity line 43, saya menambahkan beberapa kode yang isinya berupa list-list emulator yang akan di blacklist nantinya. Lalu saya menambahkan function isEmulator() yang merupakan jika nantinya ada user yang membuka HakuBank pada emulator sesuai list tadi, maka aplikasi langsung otomatis keluar dan menampilkan pesan error “Emulator Detected!”.

### c. Code obfuscation



```
7 compileSdk 33
8
9 defaultConfig { DefaultConfig it ->
10     applicationId "com.climawan.comp0844001.pertemuan5.hakubank"
11     minSdk 33
12     targetSdk 33
13     versionCode 1
14     versionName "1.0"
15
16     testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
17 }
18
19 compileOptions { CompileOptions it ->
20     sourceCompatibility JavaVersion.VERSION_1_8
21     targetCompatibility JavaVersion.VERSION_1_8
22 }
23 buildTypes { NamedDomainObjectContainer<BuildType> it ->
24     release {
25         minifyEnabled true
26         proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
27     }
28 }
29 }
```

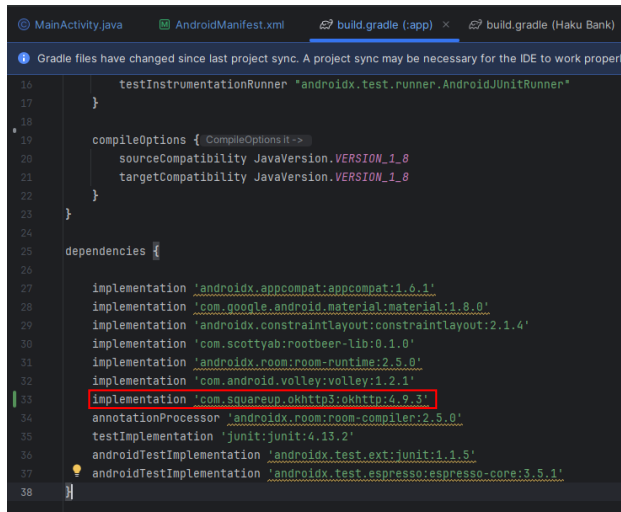
**Loc:** app/build.gradle

Saya menyisipkan beberapa code di build.gradle yang dimana tujuan adanya code tersebut adalah agar kode tidak dengan mudah dibaca oleh orang lain dan lebih sulit dibaca dan dipahami oleh manusia atau alat analisis seperti JADX-Gui serta terhindar dari reverse engineering. Sehingga aplikasi HakuBank jauh lebih baik dan tidak mudah terbaca source codenya.

### d. SSL pinning

Pembuatan suatu aplikasi, sangat penting untuk memasang SSL Pinning. Hal ini penting dilakukan karena SSL Pinning berguna untuk menjaga komunikasi 2 arah antara aplikasi dengan server sehingga terhindar dari Man in The Middle (MITM) atau menghindari adanya pihak ketiga yang ingin mencoba merusak/memanipulasi data. Biasanya tipe ssl pinning yang sering diterapkan adalah menggunakan OkHttp.



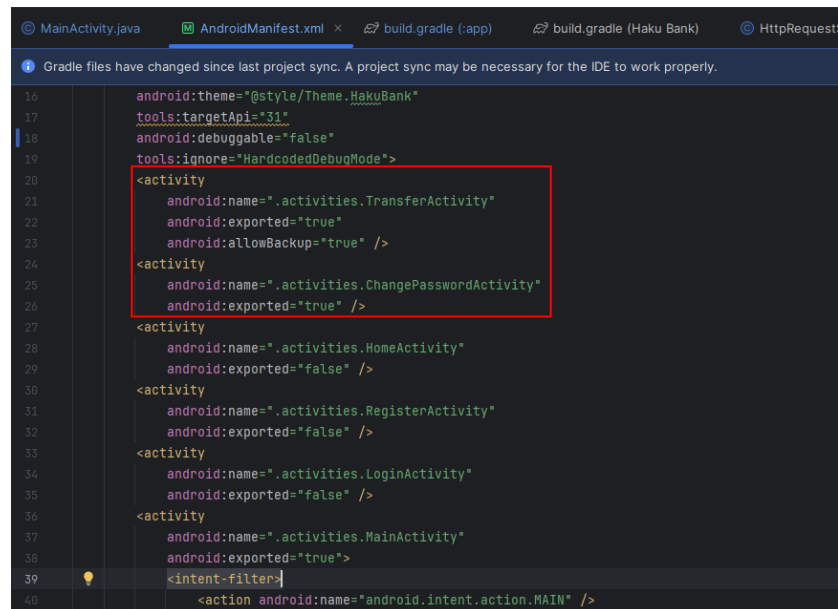


```
16 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
17 }
18
19 compileOptions {
20     sourceCompatibility JavaVersion.VERSION_1_8
21     targetCompatibility JavaVersion.VERSION_1_8
22 }
23
24 dependencies {
25     implementation 'androidx.appcompat:appcompat:1.6.1'
26     implementation 'com.google.android.material:material:1.8.0'
27     implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
28     implementation 'com.scottab:rootbeer-lib:0.1.0'
29     implementation 'androidx.room:room-runtime:2.5.0'
30     implementation 'com.android.volley:volley:1.2.1'
31     implementation 'com.squareup.okhttp3:okhttp:4.9.3'
32     annotationProcessor 'androidx.room:room-compiler:2.5.0'
33     testImplementation 'junit:junit:4.13.2'
34     androidTestImplementation 'androidx.test.ext:junit:1.1.5'
35     androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
36 }
```

**Loc:** app/build.gradle

Pada kode di atas line 33, saya menambahkan dependensi “implementation 'com.squareup.okhttp3:okhttp:4.9.3'” di file build.gradle. Dependensi tersebut merupakan penerapan pustaka HTTP untuk android dan java dengan cara yang efisien.

#### e. Intent injection prevention



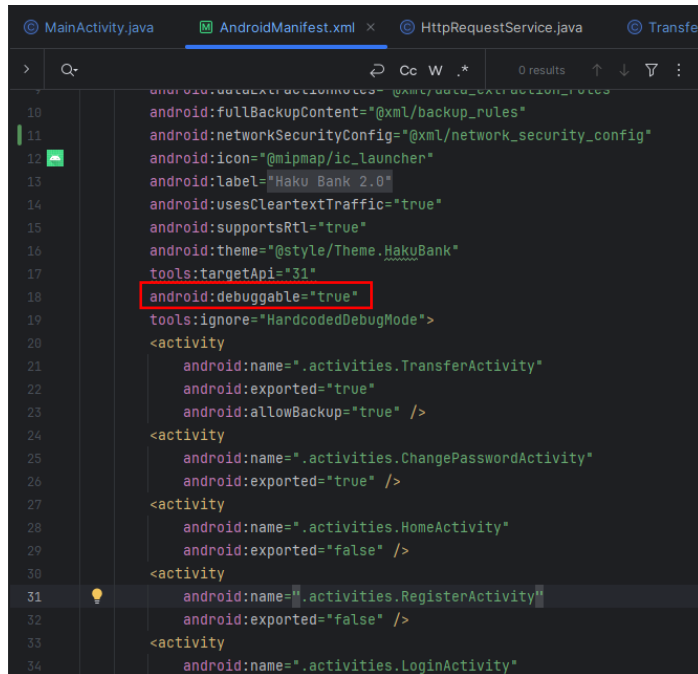
```
16 android:theme="@style/Theme_HakuBank"
17 tools:targetApi="31"
18 android:debuggable="false"
19 tools:ignore="HardcodedDebugMode">
20 <activity
21     android:name=".activities.TransferActivity"
22     android:exported="true"
23     android:allowBackup="true" />
24 <activity
25     android:name=".activities.ChangePasswordActivity"
26     android:exported="true" />
27 <activity
28     android:name=".activities.HomeActivity"
29     android:exported="false" />
30 <activity
31     android:name=".activities.RegisterActivity"
32     android:exported="false" />
33 <activity
34     android:name=".activities.LoginActivity"
35     android:exported="false" />
36 <activity
37     android:name=".activities.MainActivity"
38     android:exported="true">
39     <intent-filter>
40         <action android:name="android.intent.action.MAIN" />
```

**Loc:** app/src/main/AndroidManifest.xml

Code yang terdapat pada AndroidManifest terlihat bahwa ada 2 yang memiliki “android:exported=”true” yakni terdapat TransferActivity (line 21) dan ChangePasswordActivity (line 25). Lalu saya mengubah menjadi “android:exported=”false” seperti gambar di bawah ini.



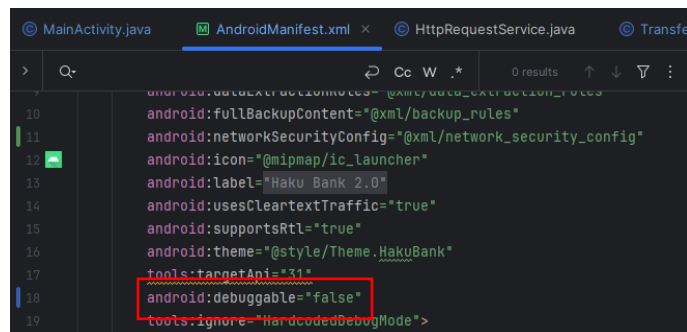
f. Disabling Android debugging feature



```
10 android:fullBackupContent="@xml/backup_rules"
11 android:networkSecurityConfig="@xml/network_security_config"
12 android:icon="@mipmap/ic_launcher"
13 android:label="Haku Bank 2.0"
14 android:usesCleartextTraffic="true"
15 android:supportsRtl="true"
16 android:theme="@style/Theme.HakuBank"
17 tools:targetApi="31"
18 android:debuggable="true"
19 tools:ignore="HardcodedDebugMode">
20 <activity
21     android:name=".activities.TransferActivity"
22     android:exported="true"
23     android:allowBackup="true" />
24 <activity
25     android:name=".activities.ChangePasswordActivity"
26     android:exported="true" />
27 <activity
28     android:name=".activities.HomeActivity"
29     android:exported="false" />
30 <activity
31     android:name=".activities.RegisterActivity"
32     android:exported="false" />
33 <activity
34     android:name=".activities.LoginActivity"
```

Loc: app/src/main/AndroidManifest.xml

Pada gambar di atas terdapat, line 18 terdapat ‘**android:debuggable=”true”**’ yang dimana aplikasi HakuBank memberikan akses kepada siapa saja untuk melakukan debugging secara langsung. Hal ini memungkinkan aplikasi dapat di exploit melalui debugging. Maka dari itu, saya mengubah dari **true** menjadi **false** seperti di bawah ini.



```
10 android:fullBackupContent="@xml/backup_rules"
11 android:networkSecurityConfig="@xml/network_security_config"
12 android:icon="@mipmap/ic_launcher"
13 android:label="Haku Bank 2.0"
14 android:usesCleartextTraffic="true"
15 android:supportsRtl="true"
16 android:theme="@style/Theme.HakuBank"
17 tools:targetApi="31"
18 android:debuggable="false"
19 tools:ignore="HardcodedDebugMode">
```