IMF Business School

Hoja de respuestas

Módulo	Seguridad en smartphones
Nombre y apellidos	Olha Biedash
Fecha entrega	01.03.2019

The applications analyzed

The applications we selected for the analysis are two mobile banking applications, developed in different countries and by different vendors:

InsecureBankV2

We selected this application in tutorials purpose to be able to practice new skills and tools having a "tutorial line" in form of existing documentation for this project. In our paper we use comparative analysis of results provided by different static analysis tools, which is unique comparing to what can be found in Internet. The project itself is a decent tool to train a novice ad to better understand the use of static analysis tools.

de.number26.android.apk

No26 is the Europe's first completely mobile bank, which is founded in Germany and gaining more popularity among users. The main peculiarity of mobile bank is absence of physical bank branches. Bank offers rich mobile application functionality to satisfy most day-to-day financial activities and with easy and fast onboarding process. The outstanding feature of No26 bank is that the mobile application development approach is applied on financial branch. Meaning, that bank service has "freemium" basic functionality and "premium" functionality, making the use of financial operations more flexible and financial products friendlier and more familiar to end users.

Reasons for the analysis

Any financial technology present online for a big number of users is a cheesy piece for cyber criminals. But in the present reality performing online transactions from web as from mobile applications has become an inevitable part of people's everyday life. And of course the mobile banking is becoming more important due to the tremendous transformation that mobile phones are no longer the ordinary communication devices, but a continue of personality. The world's leading financial institutions are adopting "Mobile First" strategies to leverage the game-changing platform that has revolutionized banking and become the customer's channel of choice. But while mobile presents enticing business opportunities, it also stretches the boundaries of the threat landscape, dramatically expanding the attack surface and becoming an increasing threat to the mobile banking revolution.

Applications are a favorite target. In fact, research suggests that 80% of successful breaches target the application layer. And with the explosive growth of the mobile channel and user demand for anytime/anywhere access to mobile services, mobile apps are stretching the boundaries of security, and putting them squarely in the crosshairs of malicious attacks.

Consider that 60% of mobile malware specifically targets financial information on mobile devices, and that 95% of the tested apps has at least one vulnerability.

In this work we will conduct static analysis of two bank applications.
The goal for the InsecureBankV2 application is to get more practice on understanding vulnerability analysis.

The goal for No26 Bank application is to verify the level of secure development in the real life. Another goal of this work is to study most popular static code analyzer tools for mobile applications and compare their output. We will try to evaluate if any of the selected applications have any of the most critical vulnerabilities according to the QWASP Mobile Top 10 classification 2016.

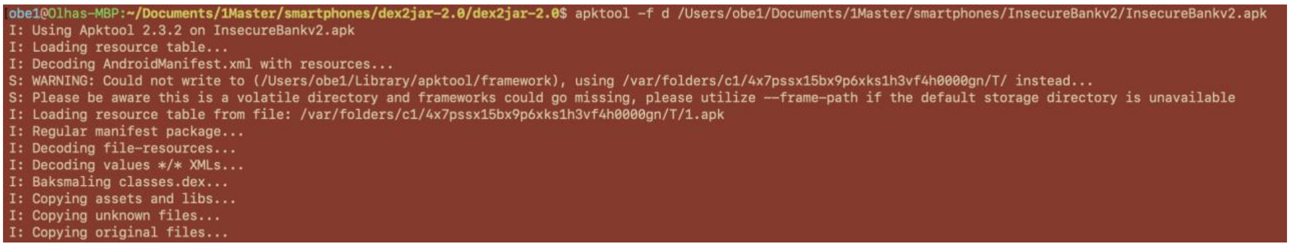
For the analysis we will use the following toolset:

- For obtaining source code: apktool, dex2jar, JD-GUI;
- For vulnerabilities check: MobSF and drozer.

Analysis: InsecureBankv2 // add link

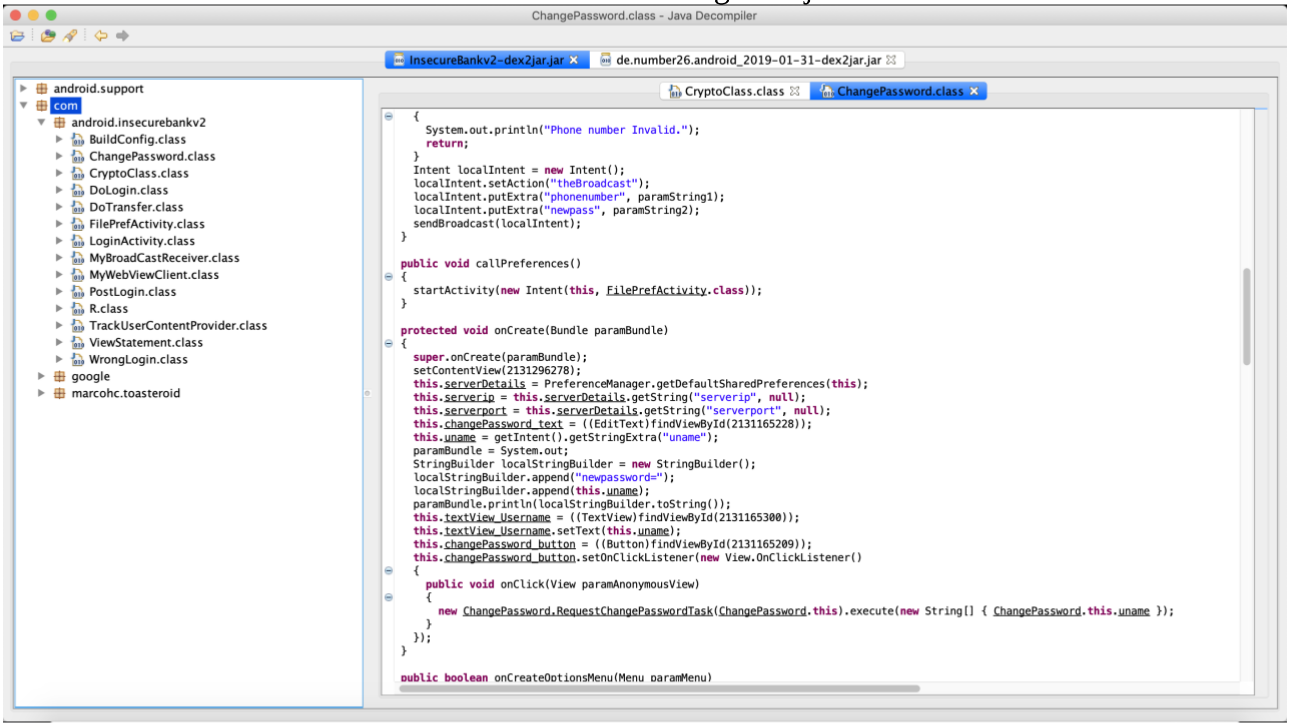
Preparation

First, we used apktool to decompile apk file to see the manifest file.



```
obe1@Olhas-MBP:~/Documents/1Master/smartphones/dex2jar-2.0/dex2jar-2.0$ apktool -f d /Users/obe1/Documents/1Master/smartphones/InsecureBankv2/InsecureBankv2.apk
I: Using Apktool 2.3.2 on InsecureBankv2.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (/Users/obe1/Library/apktool/framework), using /var/folders/c1/4x7pssx15bx9p6xks1h3vf4h0000gn/T/ instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the default storage directory is unavailable
I: Loading resource table from file: /var/folders/c1/4x7pssx15bx9p6xks1h3vf4h0000gn/T/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

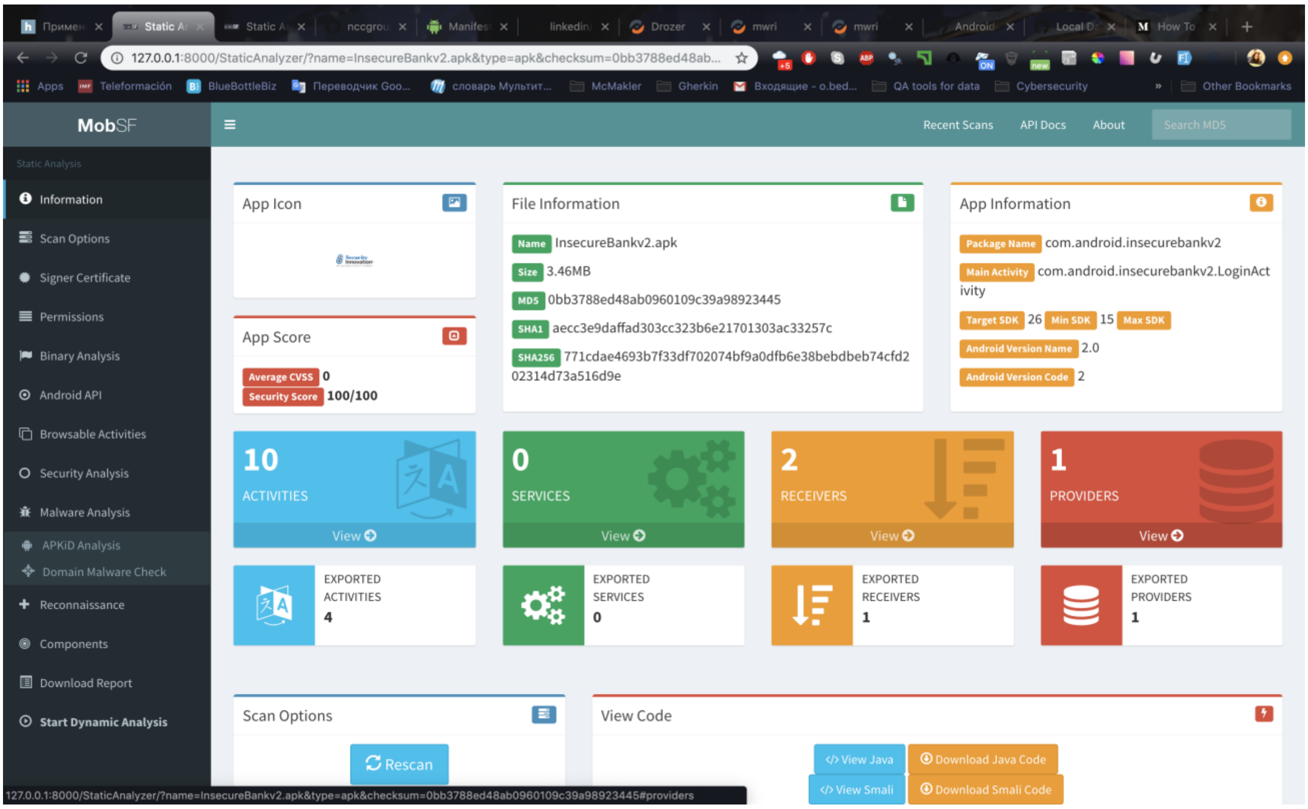
Then we converted dex classes into Java classes using dex2jar tool:



```
    {
      System.out.println("Phone number Invalid.");
      return;
    }
    Intent localIntent = new Intent();
    localIntent.setAction("theBroadcast");
    localIntent.putExtra("phonenumber", paramString1);
    localIntent.putExtra("newpass", paramString2);
    sendBroadcast(localIntent);
  }

  public void callPreferences()
  {
    startActivity(new Intent(this, FilePrefActivity.class));
  }

  protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2131296278);
    this.serverDetails = PreferenceManager.getDefaultSharedPreferences(this);
    this.serverip = this.serverDetails.getString("serverip", null);
    this.serverport = this.serverDetails.getString("serverport", null);
    this.changePassword_text = ((EditText)findViewById(2131165228));
    this.uname = getIntent().getStringExtra("uname");
    paramBundle = System.out;
    StringBuilder localStringBuilder = new StringBuilder();
    localStringBuilder.append("newpassword=");
    localStringBuilder.append(this.uname);
    paramBundle.println(localStringBuilder.toString());
    this.textView_Username = ((TextView)findViewById(2131165300));
    this.textView_Username.setText(this.uname);
    this.changePassword_button = ((Button)findViewById(2131165209));
    this.changePassword_button.setOnClickListener(new View.OnClickListener()
    {
      public void onClick(View paramAnonymousView)
      {
        new ChangePassword.RequestChangePasswordTask(ChangePassword.this).execute(new String[] { ChangePassword.this.uname });
      }
    });
  }

  public boolean onCreateOptionsMenu(Menu paramMenu)
```

As you can see, the code is not obfuscated. (**Mobile Top 10 2016-M9-Reverse Engineering)**

MobSF report analysis



127.0.0.1:8000/StaticAnalyzer/?name=InsecureBankv2.apk&type=apk&checksum=0bb3788ed48ab...

App Icon

App Score

Average CVSS 0
Security Score 100/100

File Information

Name InsecureBankv2.apk
Size 3.46MB
MD5 0bb3788ed48ab0960109c39a98923445
SHA1 aecc3e9daffad303cc323b6e21701303ac33257c
SHA256 771cdae4693b7f33df702074bf9a0dfb6e38bebdbeb74cfd202314d73a516d9e

App Information

Package Name com.android.insecurebankv2
Main Activity com.android.insecurebankv2.LoginActivity
Target SDK 26 Min SDK 15 Max SDK
Android Version Name 2.0
Android Version Code 2

10 ACTIVITIES View
0 SERVICES View
2 RECEIVERS View
1 PROVIDERS View

EXPORTED ACTIVITIES 4
EXPORTED SERVICES 0
EXPORTED RECEIVERS 1
EXPORTED PROVIDERS 1

Scan Options

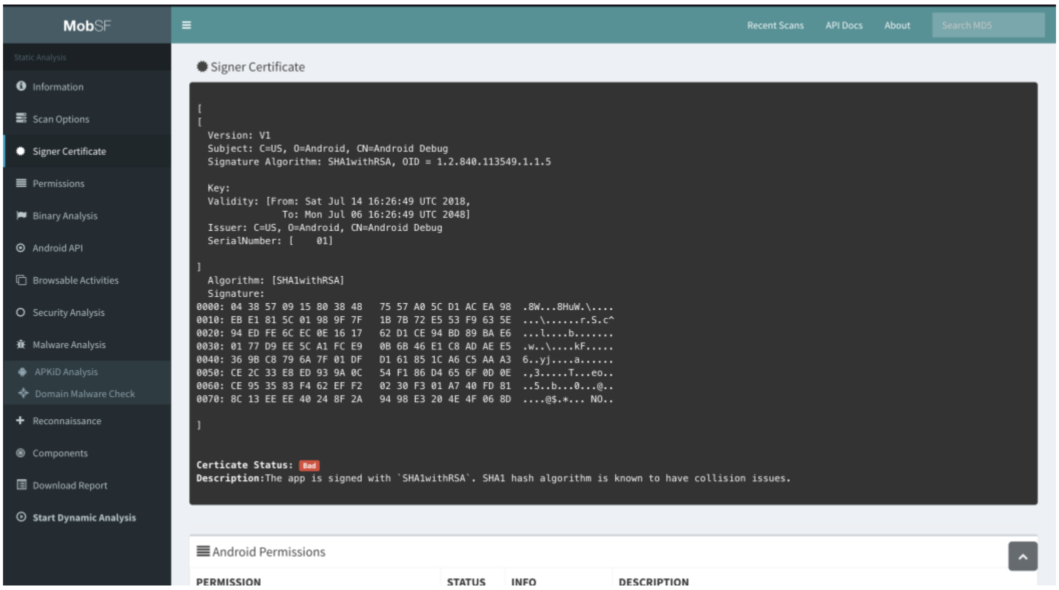
Rescan

View Code

View Java
Download Java Code
View Smali
Download Smali Code

127.0.0.1:8000/StaticAnalyzer/?name=InsecureBankv2.apk&type=apk&checksum=0bb3788ed48ab0960109c39a98923445#providers

MobSF tool starts app analysis with static analysis report gives a useful dashboard at the beginning of the report. It shows the main information about the file and about the application itself. What is interesting – the state of an application code is measured by a Security Score, which is very informative for further analysis. In our case, InsecureBankV2 application received 100 points of 100 possible. This already a proof that this application is poorly developed from the security perspective.



Signer Certificate

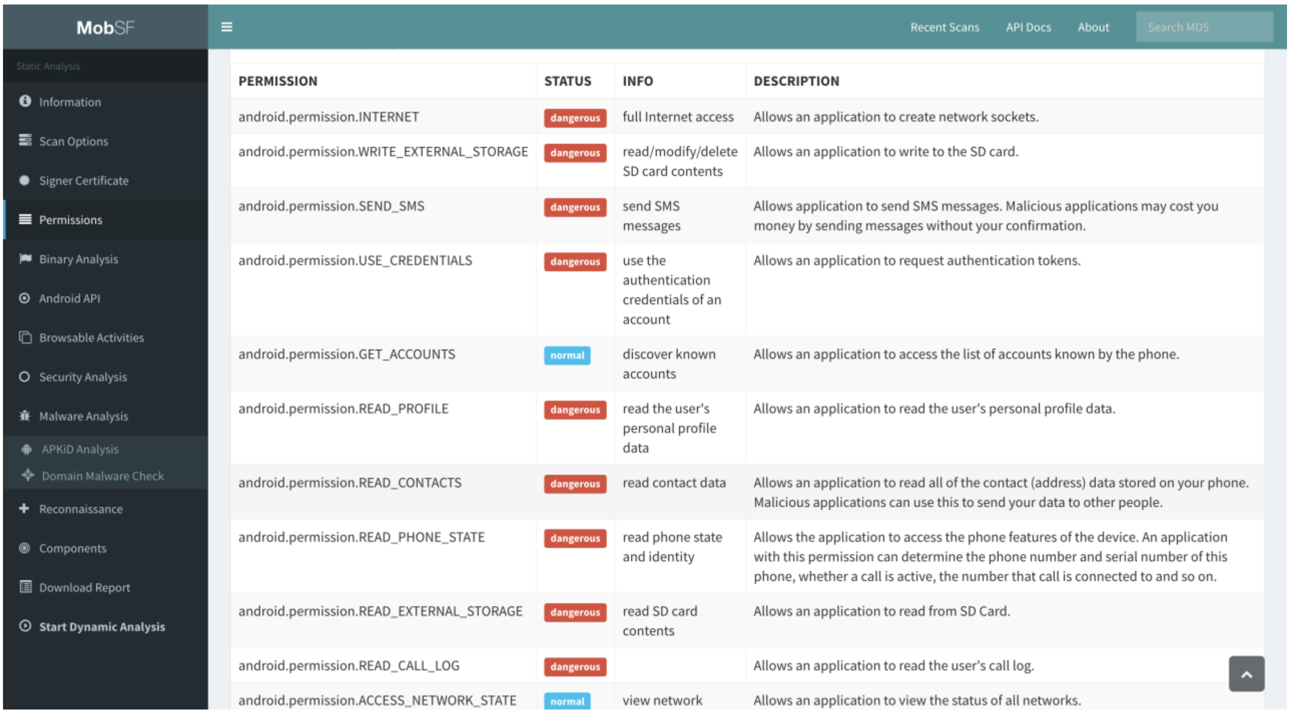
```
[
[
  Version: V1
  Subject: C=US, O=Android, CN=Android Debug
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:
  Validity: [From: Sat Jul 14 16:26:49 UTC 2018,
               To: Mon Jul 06 16:26:49 UTC 2048]
  Issuer: C=US, O=Android, CN=Android Debug
  SerialNumber: [    01]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 04 38 57 09 15 80 38 48   75 57 A0 5C D1 AC EA 98  .8W...8HuW.\....
0010: EB E1 81 5C 01 98 9F 7F   1B 7B 72 E5 53 F9 63 5E  ...\......r.S.c^
0020: 94 ED FE 6C EC 0E 16 17   62 D1 CE 94 BD 89 BA E6  ...l....b.......
0030: 01 77 D9 EE 5C A1 FC E9   0B 6B 46 E1 C8 AD AE E5  .w..\....kF.....
0040: 36 9B C8 79 6A 7F 01 DF   D1 61 85 1C A6 C5 AA A3  6..yj....a......
0050: CE 2C 33 E8 ED 93 9A 0C   54 F1 86 D4 65 6F 0D 0E  .,3.....T...eo..
0060: CE 95 35 83 F4 62 EF F2   02 30 F3 01 A7 40 FD 81  ..5..b...0...@..
0070: 8C 13 EE EE 40 24 8F 2A   94 98 E3 20 4E 4F 06 8D  ....@$.*... NO..

]


Certicate Status: Bad
Description:The app is signed with `SHA1withRSA`. SHA1 hash algorithm is known to have collision issues.
```

Android Permissions

PERMISSION	STATUS	INFO	DESCRIPTION

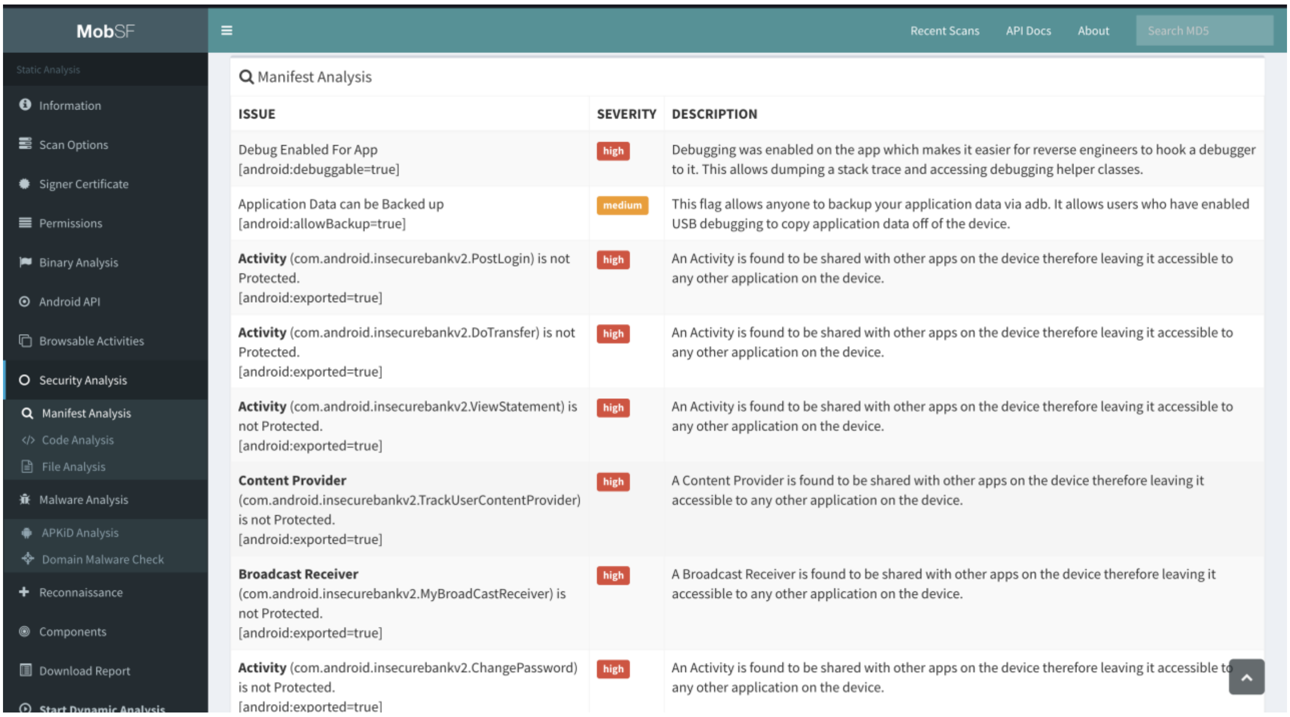
Signer Certificate section shows the certificate technology applied to sign the application. If the certificate algorithm is weak, the attacker may decipher the key and replace the apk file with his malicious application.
In our case the demo application was signed with SHA1withRSA algorithm which is defined as Bad by MobSF. (**Mobile Top 10 2016-M5-Insufficient Cryptography**)

As for the Permissions used by application, MobSF defines many of them as dangerous. But this is mostly from the point of view of the malicious application, defining if it may damage the system or get to the sensitive information. And if we imply that the owner of the application is not the original one, we should look closer to what the application does and why it demands such permissions.



PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.INTERNET	dangerous	full Internet access	Allows an application to create network sockets.
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete SD card contents	Allows an application to write to the SD card.
android.permission.SEND_SMS	dangerous	send SMS messages	Allows application to send SMS messages. Malicious applications may cost you money by sending messages without your confirmation.
android.permission.USE_CREDENTIALS	dangerous	use the authentication credentials of an account	Allows an application to request authentication tokens.
android.permission.GET_ACCOUNTS	normal	discover known accounts	Allows an application to access the list of accounts known by the phone.
android.permission.READ_PROFILE	dangerous	read the user's personal profile data	Allows an application to read the user's personal profile data.
android.permission.READ_CONTACTS	dangerous	read contact data	Allows an application to read all of the contact (address) data stored on your phone. Malicious applications can use this to send your data to other people.
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
android.permission.READ_EXTERNAL_STORAGE	dangerous	read SD card contents	Allows an application to read from SD Card.
android.permission.READ_CALL_LOG	dangerous		Allows an application to read the user's call log.
android.permission.ACCESS_NETWORK_STATE	normal	view network	Allows an application to view the status of all networks.

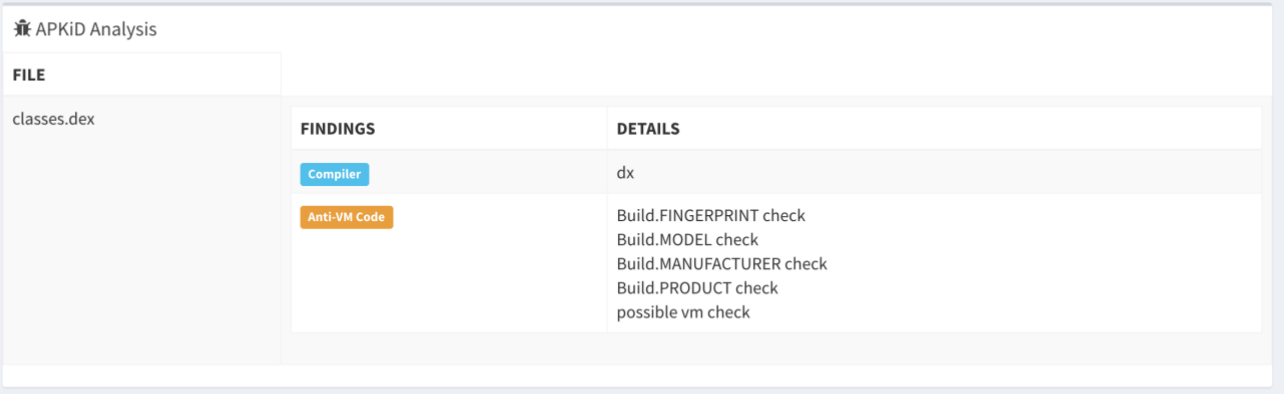
Manifest Analysis section shows details about exported activities. Yet, there is no functionality to launch activities from UI at once, unlike drozer. Yet, the quick overview with some annotation might give some clues. In our case, we analyzed activities more closely using drozer.



Manifest Analysis

ISSUE	SEVERITY	DESCRIPTION
Debug Enabled For App [android:debuggable=true]	high	Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.
Application Data can be Backed up [android:allowBackup=true]	medium	This flag allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.
Activity (com.android.insecurebankv2.PostLogin) is not Protected. [android:exported=true]	high	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.
Activity (com.android.insecurebankv2.DoTransfer) is not Protected. [android:exported=true]	high	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.
Activity (com.android.insecurebankv2.ViewStatement) is not Protected. [android:exported=true]	high	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.
Content Provider (com.android.insecurebankv2.TrackUserContentProvider) is not Protected. [android:exported=true]	high	A Content Provider is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.
Broadcast Receiver (com.android.insecurebankv2.MyBroadCastReceiver) is not Protected. [android:exported=true]	high	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.
Activity (com.android.insecurebankv2.ChangePassword) is not Protected. [android:exported=true]	high	An Activity is found to be shared with other apps on the device therefore leaving it accessible t any other application on the device.

APKiD Analysis section shows which compilers are used and what anti virtual machine code is used.



APKiD Analysis

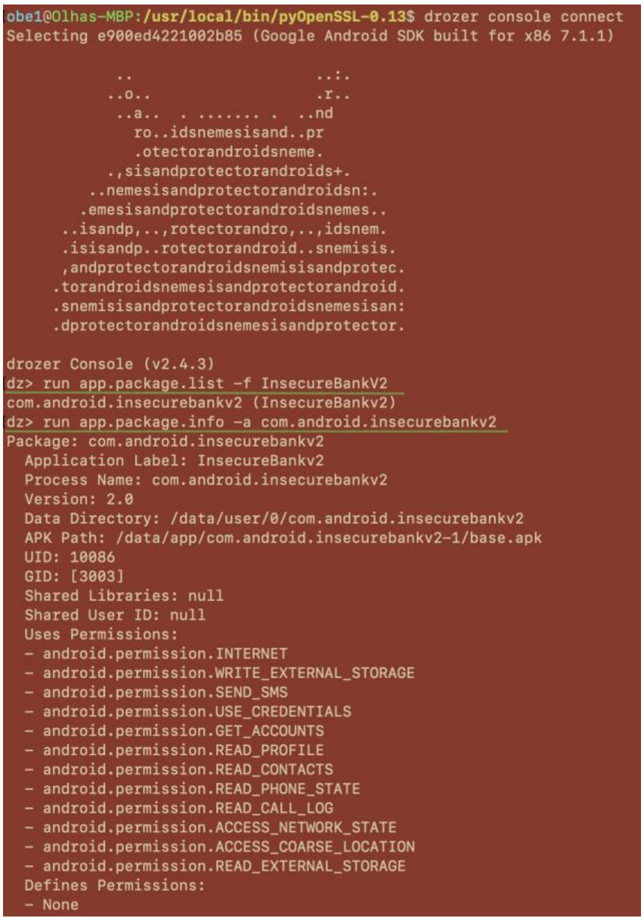
FILE	FINDINGS	DETAILS
classes.dex	Compiler	dx
	Anti-VM Code	Build.FINGERPRINT check Build.MODEL check Build.MANUFACTURER check Build.PRODUCT check possible vm check

Further code analysis can be conducted following the path, described in [4]. Yet, at the moment we omit excessive detalization, leaving space for further analysis.
In our case we can say that InsecureBankV2 has Anti-VM code, yet comparing to No26 Bank application, where are three different Anti-VM techniques applied. The mechanism of attack is also sufficiently described in article [4]. (**Mobile Top 10 2016-M1-Improper Platform Usage**).

The full MobSF report of InsecureBankV2 is attached.

Drozer report analysis

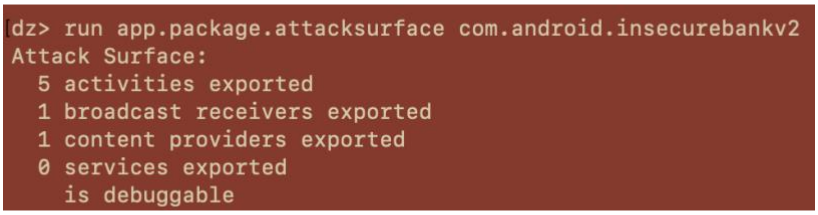
Drozer is a powerful tull for vulnerability analysis of the applications.
We set up the connection between Drozer client and the Console app and launched scanning of InsecureBankV2 application.



```
obe1@Olhas-MBP:/usr/local/bin/pyOpenSSL-0.13$ drozer console connect
Selecting e900ed4221002b85 (Google Android SDK built for x86 7.1.1)

            ..                    ..:.
           ..o..                  .r..
            ..a..  . ....... .  ..nd
              ro..idsnemesisand..pr
              .otectorandroidsneme.
           .,sisandprotectorandroids+.
         ..nemesisandprotectorandroidsn:.
        .emesisandprotectorandroidsnemes..
      ..isandp,..,rotectorandro,..,idsnem.
      .isisandp..rotectorandroid..snemisis.
      ,andprotectorandroidsnemisisandprotec.
     .torandroidsnemesisandprotectorandroid.
     .snemisisandprotectorandroidsnemesisan:
     .dprotectorandroidsnemesisandprotector.

drozer Console (v2.4.3)
dz> run app.package.list -f InsecureBankV2
com.android.insecurebankv2 (InsecureBankv2)
dz> run app.package.info -a com.android.insecurebankv2
Package: com.android.insecurebankv2
  Application Label: InsecureBankv2
  Process Name: com.android.insecurebankv2
  Version: 2.0
  Data Directory: /data/user/0/com.android.insecurebankv2
  APK Path: /data/app/com.android.insecurebankv2-1/base.apk
  UID: 10086
  GID: [3003]
  Shared Libraries: null
  Shared User ID: null
  Uses Permissions:
  - android.permission.INTERNET
  - android.permission.WRITE_EXTERNAL_STORAGE
  - android.permission.SEND_SMS
  - android.permission.USE_CREDENTIALS
  - android.permission.GET_ACCOUNTS
  - android.permission.READ_PROFILE
  - android.permission.READ_CONTACTS
  - android.permission.READ_PHONE_STATE
  - android.permission.READ_CALL_LOG
  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.ACCESS_COARSE_LOCATION
  - android.permission.READ_EXTERNAL_STORAGE
  Defines Permissions:
  - None
```

First, we launched *app.package.info* command to find out the basic info about application package and the list of permissions it uses and defines.

Now we define the attack vectors for the application by running command *app.package.attacksurface*



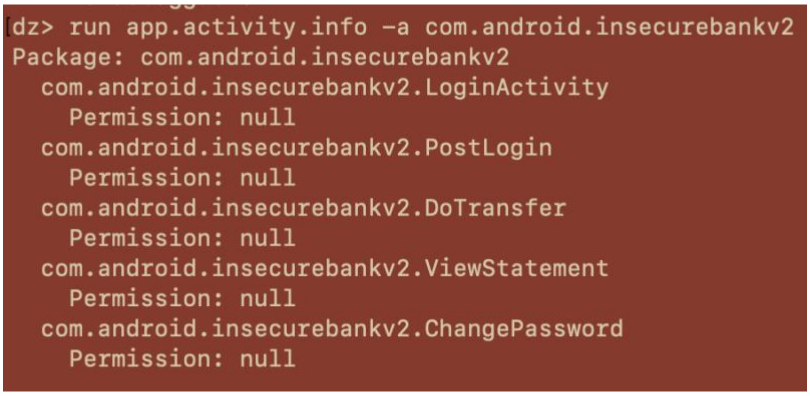
```
dz> run app.package.attacksurface com.android.insecurebankv2
Attack Surface:
  5 activities exported
  1 broadcast receivers exported
  1 content providers exported
  0 services exported
    is debuggable
```

Drozer console using attack surface exposes **Mobile Top 10 2016-M1-Improper Platform Usage** category threats that are a number of potential vectors as a result of exposed API call. The app 'exports' (makes accessible to other apps) a number of activities (screens used by the app), content providers (database objects) and services (background workers).

Five exported activities relate to

We also note that the service is debuggable, which means that we can attach a debugger to the process, using adb, and step through the code. (**Mobile Top 10 2016-M10-Extraneous Functionality**)

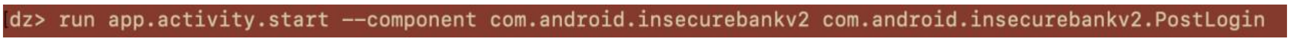
Launching activities:



```
dz> run app.activity.info -a com.android.insecurebankv2
Package: com.android.insecurebankv2
  com.android.insecurebankv2.LoginActivity
    Permission: null
  com.android.insecurebankv2.PostLogin
    Permission: null
  com.android.insecurebankv2.DoTransfer
    Permission: null
  com.android.insecurebankv2.ViewStatement
    Permission: null
  com.android.insecurebankv2.ChangePassword
    Permission: null
```

- LoginActivity
  Expected activity, as the login screen is the first activity user sees in application.

- PostLogin

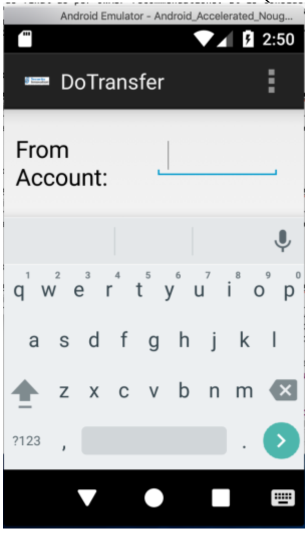
  

  ```
  dz> run app.activity.start --component com.android.insecurebankv2 com.android.insecurebankv2.PostLogin
  ```

  Calling this activity allowed to bypass login procedure. This can be a potential issue of the data leakage. (**Mobile Top 10 2016-M2-Insecure Data Storage, Mobile Top 10 2016-M6-Insecure Authorization**)

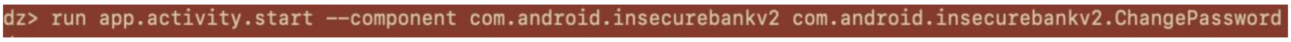
- DoTransfer

  Calling this activity opens transaction creation process. Activity is available to an unauthorized user. (**Mobile Top 10 2016-M1-Improper Platform Usage**)

  From Account:

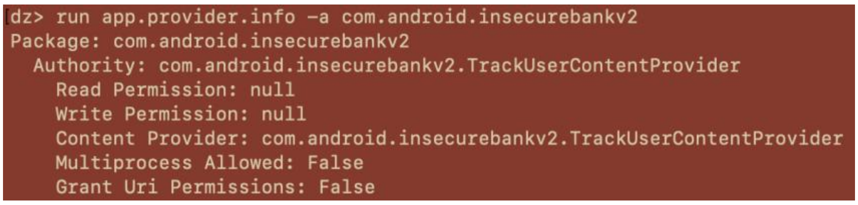
- ChangePassword

  ```
  dz> run app.activity.start --component com.android.insecurebankv2 com.android.insecurebankv2.ChangePassword
  ```

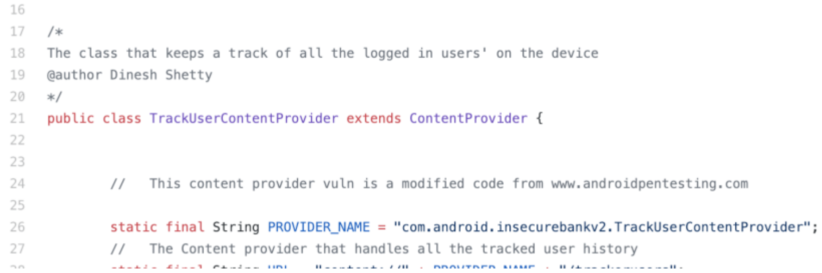
> “The screen you are seeing will allow to change password without authenticating to the application. In this specific case, another person should have had an access to the device to be able to change password. However, there are cases when parameters can be passed to the activities being launched, and those activities would operate on the given parameters. It is important to keep that in mind when evaluating real-world applications (looking into the source code of exported activities would be warranted to determine whether it reads any parameters from an intent that was used to launch it).” [1] (**Mobile Top 10 2016-M4-Insecure Authentication**)

Analyzing content providers



```
dz> run app.provider.info -a com.android.insecurebankv2
Package: com.android.insecurebankv2
  Authority: com.android.insecurebankv2.TrackUserContentProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.android.insecurebankv2.TrackUserContentProvider
    Multiprocess Allowed: False
    Grant Uri Permissions: False
```

Drozer shows one exported content provider TrackUserContentProvider. We simply search for the content provider in the source code to see what is does exactly. This is what we find:

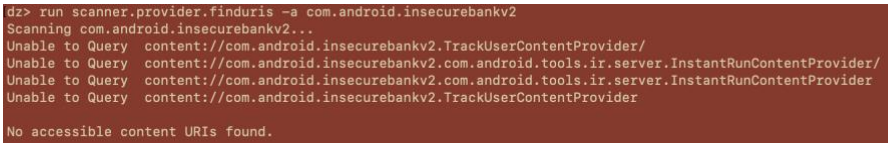


```

/*
The class that keeps a track of all the logged in users' on the device
@author Dinesh Shetty
*/
public class TrackUserContentProvider extends ContentProvider {


        //   This content provider vuln is a modified code from www.androidpentesting.com

        static final String PROVIDER_NAME = "com.android.insecurebankv2.TrackUserContentProvider";
        //   The Content provider that handles all the tracked user history
```

It means that this content provider, which does not require any permission to interact with it, stores the information about all logged in users. This is a serious vulnerability refers to Mobile Top 10 2016-M2-Insecure Data Storage.

Database-backed Content Providers

We also tried to search for accessible content URIs, but the scan reported no results.

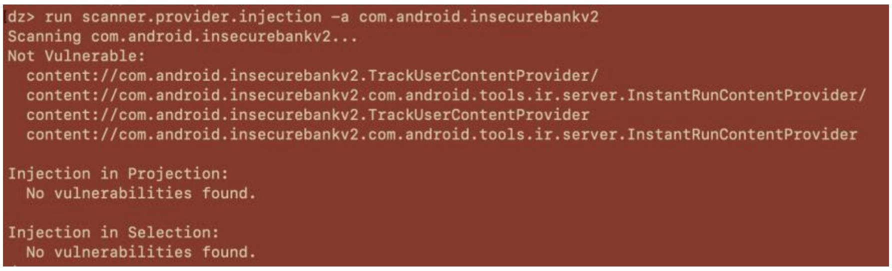


```
dz> run scanner.provider.finduris -a com.android.insecurebankv2
Scanning com.android.insecurebankv2...
Unable to Query  content://com.android.insecurebankv2.TrackUserContentProvider/
Unable to Query  content://com.android.insecurebankv2.com.android.tools.ir.server.InstantRunContentProvider/
Unable to Query  content://com.android.insecurebankv2.com.android.tools.ir.server.InstantRunContentProvider
Unable to Query  content://com.android.insecurebankv2.TrackUserContentProvider

No accessible content URIs found.
```

For that we could not obtain any data, but we will come back to this topic later, as it needs investigation.

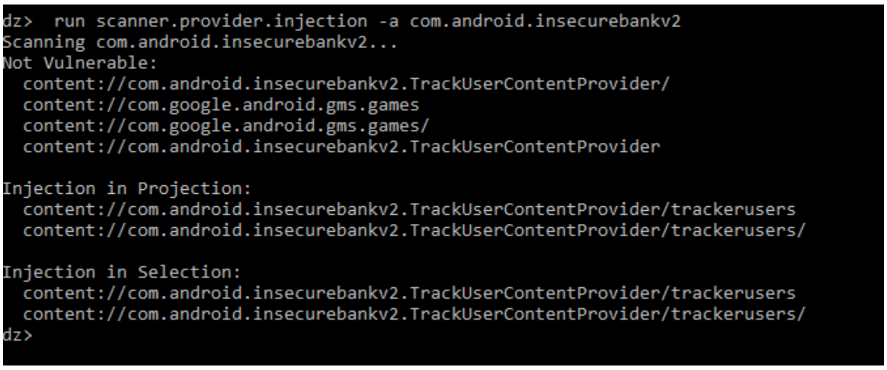
Content Provider Vulnerabilities

From the first attempt, drozer showed that no sql injection vulnerabilities were found.



```
dz> run scanner.provider.injection -a com.android.insecurebankv2
Scanning com.android.insecurebankv2...
Not Vulnerable:
  content://com.android.insecurebankv2.TrackUserContentProvider/
  content://com.android.insecurebankv2.com.android.tools.ir.server.InstantRunContentProvider/
  content://com.android.insecurebankv2.TrackUserContentProvider
  content://com.android.insecurebankv2.com.android.tools.ir.server.InstantRunContentProvider

Injection in Projection:
  No vulnerabilities found.

Injection in Selection:
  No vulnerabilities found.
```

But we made another attempt: we had to fix the backend of the application to run Python server and repeated the test.



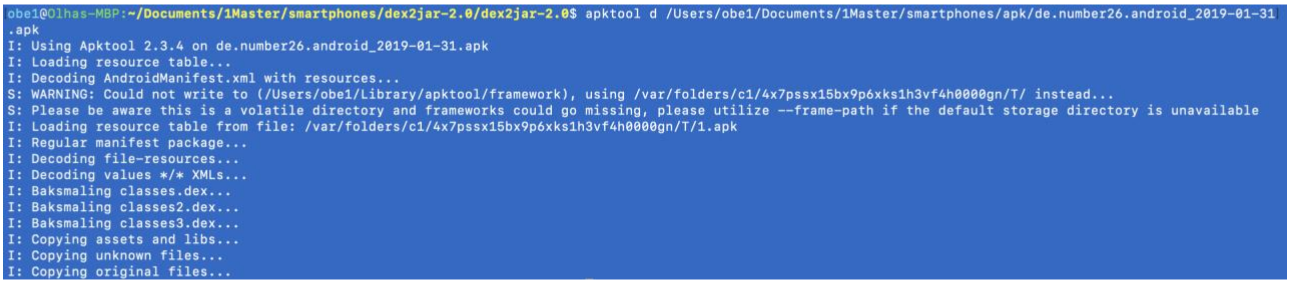
```
dz>  run scanner.provider.injection -a com.android.insecurebankv2
Scanning com.android.insecurebankv2...
Not Vulnerable:
  content://com.android.insecurebankv2.TrackUserContentProvider/
  content://com.google.android.gms.games
  content://com.google.android.gms.games/
  content://com.android.insecurebankv2.TrackUserContentProvider

Injection in Projection:
  content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers
  content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers/

Injection in Selection:
  content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers
  content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers/
dz>
```

This time we can see four vulnerable requests (**Mobile Top 10 2016-M2-Insecure Data Storage**).

Analysis: de.number26.android_2019-01-31.apk

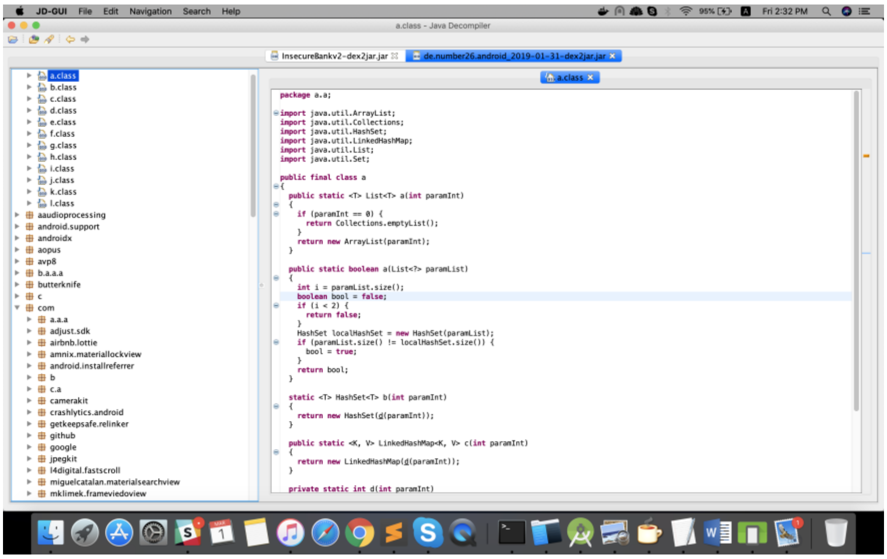
Preparation

We used apktool to **de.number26.android_2019-01-31.apk:**



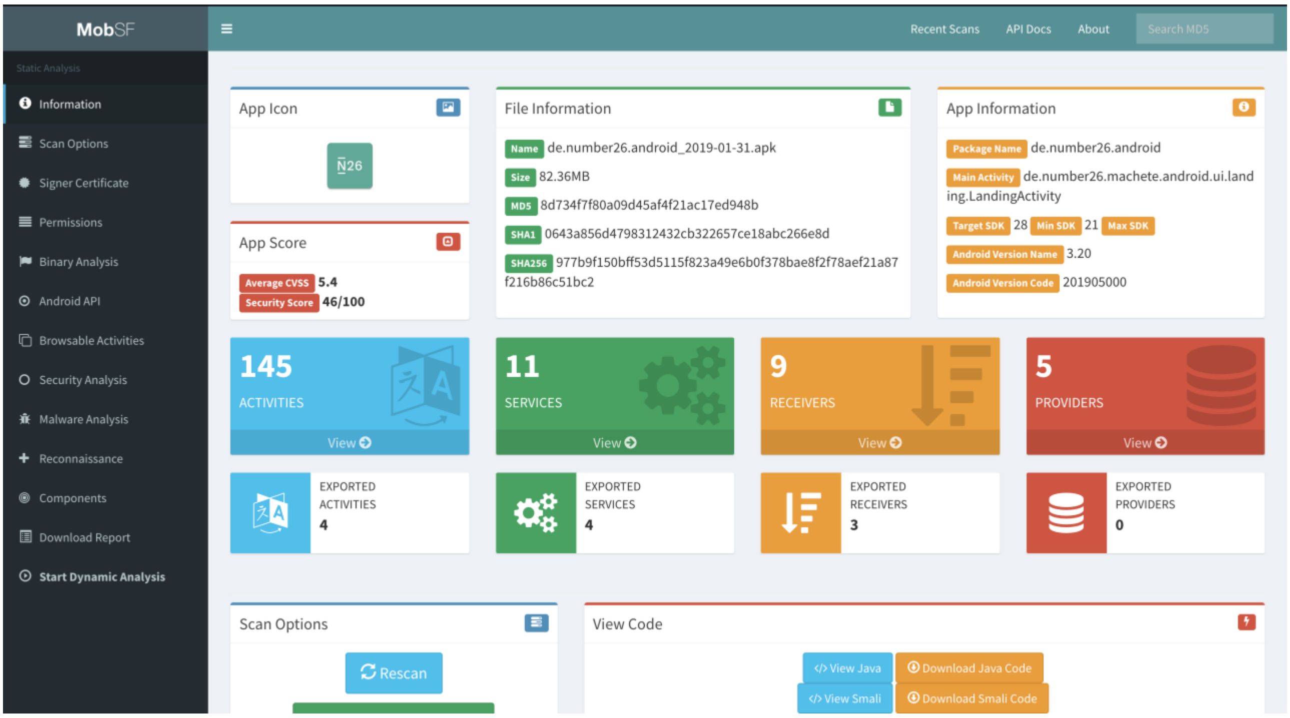
```
obe1@Olhas-MBP:~/Documents/1Master/smartphones/dex2jar-2.0/dex2jar-2.0$ apktool d /Users/obe1/Documents/1Master/smartphones/apk/de.number26.android_2019-01-31
.apk
I: Using Apktool 2.3.4 on de.number26.android_2019-01-31.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (/Users/obe1/Library/apktool/framework), using /var/folders/c1/4x7pssx15bx9p6xks1h3vf4h0000gn/T/ instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the default storage directory is unavailable
I: Loading resource table from file: /var/folders/c1/4x7pssx15bx9p6xks1h3vf4h0000gn/T/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...
```

We used dex2jar tool to convert dex classes into Java classes. The conversion showed that the source code of the application is obfuscated. It makes the further static analysis of the code exceptionally hard and it might be less time consuming to continue with dynamic analysis instead.



```
package a.a;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Set;

public final class a
{
  public static <T> List<T> a(int paramInt)
  {
    if (paramInt == 0) {
      return Collections.emptyList();
    }
    return new ArrayList(paramInt);
  }

  public static boolean a(List<?> paramList)
  {
    int i = paramList.size();
    boolean bool = false;
    if (i < 2) {
      return false;
    }
    HashSet localHashSet = new HashSet(paramList);
    if (paramList.size() != localHashSet.size()) {
      bool = true;
    }
    return bool;
  }

  static <T> HashSet<T> b(int paramInt)
  {
    return new HashSet(d(paramInt));
  }

  public static <K, V> LinkedHashMap<K, V> c(int paramInt)
  {
    return new LinkedHashMap(d(paramInt));
  }

  private static int d(int paramInt)
```

MobSF report analysis



File Information

Name de.number26.android_2019-01-31.apk

Size 82.36MB

MD5 8d734f7f80a09d45af4f21ac17ed948b

SHA1 0643a856d4798312432cb322657ce18abc266e8d

SHA256 977b9f150bff53d5115f823a49e6b0f378bae8f2f78aef21a87f216b86c51bc2

App Information

Package Name de.number26.android

Main Activity de.number26.machete.android.ui.landing.LandingActivity

Target SDK 28 Min SDK 21 Max SDK

Android Version Name 3.20

Android Version Code 201905000

App Score

Average CVSS 5.4

Security Score 46/100

145 ACTIVITIES

11 SERVICES

9 RECEIVERS

5 PROVIDERS

EXPORTED ACTIVITIES 4

EXPORTED SERVICES 4

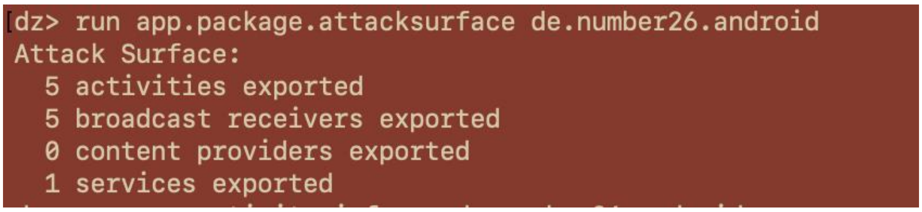
EXPORTED RECEIVERS 3

EXPORTED PROVIDERS 0

Scan Options

View Code

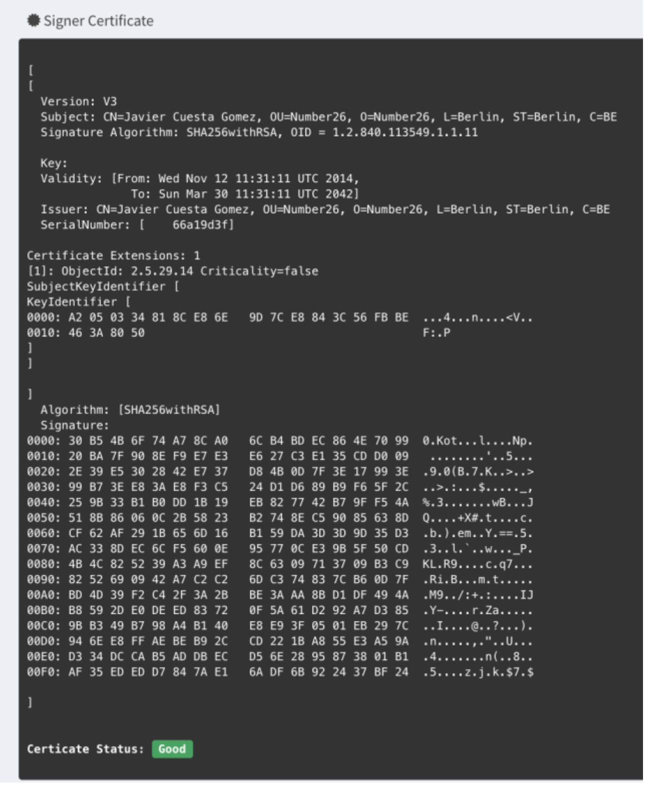
The dashboard shows that the application has 145 activities, 11 services, 9 receivers and 5 providers. The potentially vulnerable are 4 activities, 4 services, 3 receivers. It is interesting to notice the difference between the attach surface of this current apk file and the one, we tried to scan earlier (release of mid Dec 2018):



```
dz> run app.package.attacksurface de.number26.android
Attack Surface:
  5 activities exported
  5 broadcast receivers exported
  0 content providers exported
  1 services exported
```

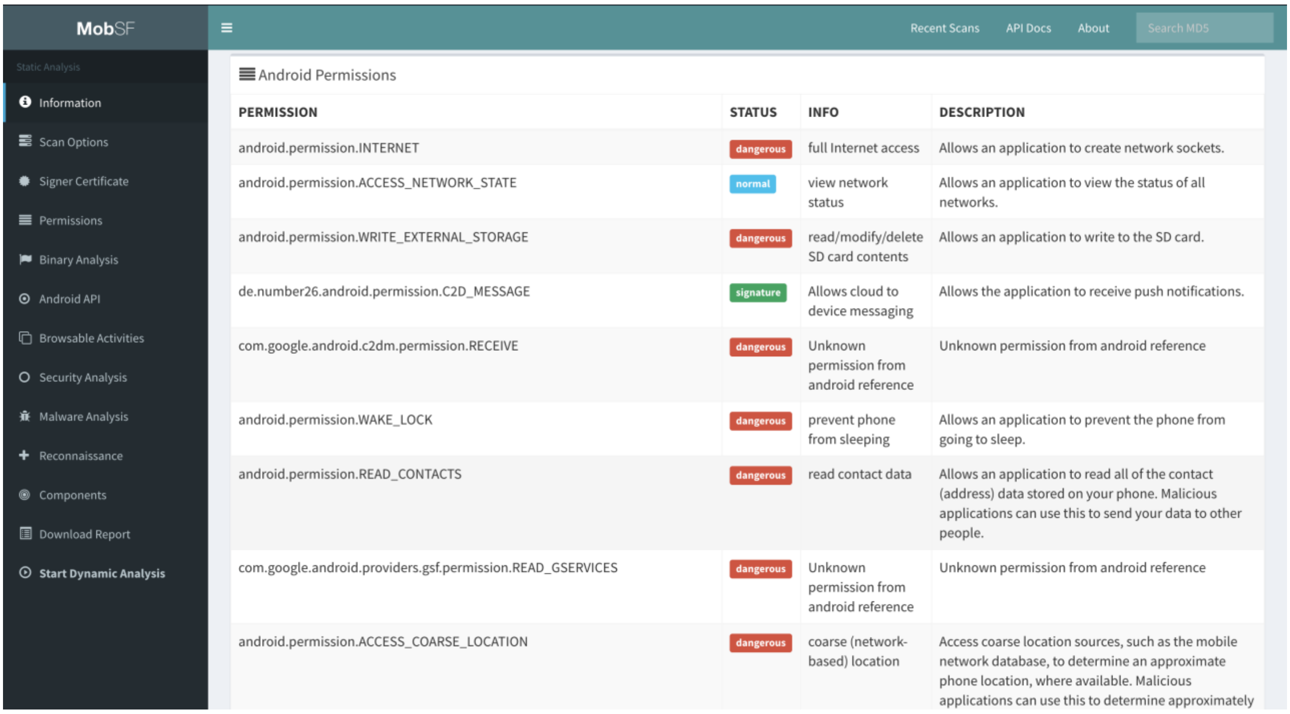
It shows that the number of exported services has grown, yet the rate of exported activities and receivers has improved.

The security score according to MobSF is 46 points out of 100 possible, which is a sign of a securely developed application.



```
[
[
  Version: V3
  Subject: CN=Javier Cuesta Gomez, OU=Number26, O=Number26, L=Berlin, ST=Berlin, C=BE
  Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

  Key:
  Validity: [From: Wed Nov 12 11:31:11 UTC 2014,
               To: Sun Mar 30 11:31:11 UTC 2042]
  Issuer: CN=Javier Cuesta Gomez, OU=Number26, O=Number26, L=Berlin, ST=Berlin, C=BE
  SerialNumber: [    66a19d3f]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: A2 05 03 34 81 8C E8 6E   9D 7C E8 84 3C 56 FB BE  ...4...n....<V..
0010: 46 3A 80 50                                        F:.P
]
]


]
  Algorithm: [SHA256withRSA]
  Signature:
0000: 30 B5 4B 6F 74 A7 8C A0   6C B4 BD EC 86 4E 70 99  0.Kot...l....Np.
0010: 20 BA 7F 90 8E F9 E7 E3   E6 27 C3 E1 35 CD D0 09   ........'..5...
0020: 2E 39 E5 30 28 42 E7 37   D8 4B 0D 7F 3E 17 99 3E  .9.0(B.7.K..>..>
0030: 99 B7 3E E8 3A E8 F3 C5   24 D1 D6 89 B9 F6 5F 2C  ..>.:...$....._,
0040: 25 9B 33 B1 B0 DD 1B 19   EB 82 77 42 B7 9F F5 4A  %.3.......wB...J
0050: 51 8B 86 06 0C 2B 58 23   B2 74 8E C5 90 85 63 8D  Q....+X#.t....c.
0060: CF 62 AF 29 1B 65 6D 16   B1 59 DA 3D 3D 9D 35 D3  .b.).em..Y.==.5.
0070: AC 33 8D EC 6C F5 60 0E   95 77 0C E3 9B 5F 50 CD  .3..l.`..w..._P.
0080: 4B 4C 82 52 39 A3 A9 EF   8C 63 09 71 37 09 B3 C9  KL.R9....c.q7...
0090: 82 52 69 09 42 A7 C2 C2   6D C3 74 83 7C B6 0D 7F  .Ri.B...m.t.....
00A0: BD 4D 39 F2 C4 2F 3A 2B   BE 3A AA 8B D1 DF 49 4A  .M9../:+.:....IJ
00B0: B8 59 2D E0 DE ED 83 72   0F 5A 61 D2 92 A7 D3 85  .Y-....r.Za.....
00C0: 9B B3 49 B7 98 A4 B1 40   E8 E9 3F 05 01 EB 29 7C  ..I....@..?...).
00D0: 94 6E E8 FF AE BE B9 2C   CD 22 1B A8 55 E3 A5 9A  .n.....,."..U...
00E0: D3 34 DC CA B5 AD DB EC   D5 6E 28 95 87 38 01 B1  .4.......n(..8..
00F0: AF 35 ED ED D7 84 7A E1   6A DF 6B 92 24 37 BF 24  .5....z.j.k.$7.$

]


Certicate Status: Good
```

Certificate Status of No26Bank is Good.



Android Permissions

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.INTERNET	dangerous	full Internet access	Allows an application to create network sockets.
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete SD card contents	Allows an application to write to the SD card.
de.number26.android.permission.C2D_MESSAGE	signature	Allows cloud to device messaging	Allows the application to receive push notifications.
com.google.android.c2dm.permission.RECEIVE	dangerous	Unknown permission from android reference	Unknown permission from android reference
android.permission.WAKE_LOCK	dangerous	prevent phone from sleeping	Allows an application to prevent the phone from going to sleep.
android.permission.READ_CONTACTS	dangerous	read contact data	Allows an application to read all of the contact (address) data stored on your phone. Malicious applications can use this to send your data to other people.
com.google.android.providers.gsf.permission.READ_GSERVICES	dangerous	Unknown permission from android reference	Unknown permission from android reference
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

Most of the Permissions of the application are defined as dangerous. Yet, looking closer on them and knowing the application functionality, most of them are righteous. Here are a list of potentially most vulnerable application permissions, through which another malicious application or a virus, may provoke the application. Yet, as the source code is obfuscated, we will not conduct the code analysis.

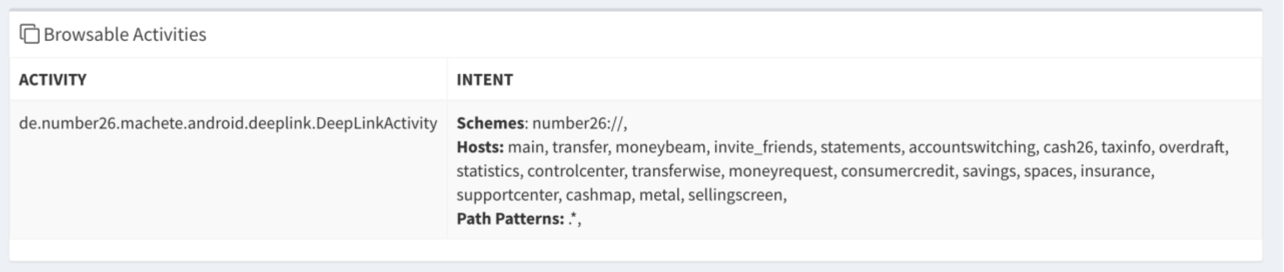
Permission **android.permission.WRITE_EXTERNAL_STORAGE** hints to investigate what exactly is stored into the local storage, if any sensitive information is logged and can be obtained. (**Mobile Top 10 2016-M2-Insecure Data Storage**)

Permission **android.permission.READ_EXTERNAL_STORAGE** can be potentially dangerous, if the input data is not verified. (possible **Mobile Top 10 2016-M7-Poor Code Quality**).

Permission **android.permission.USE_FINGERPRINT** is defined by MobSf as normal, yet, according to OWASP Top 10, **Mobile Top 10 2016-M4-Insecure Authentication** [5]:

> *"There are many different ways that a mobile app may suffer from insecure authentication: ... If the mobile app uses a feature like TouchID, it suffers from insecure authentication."*

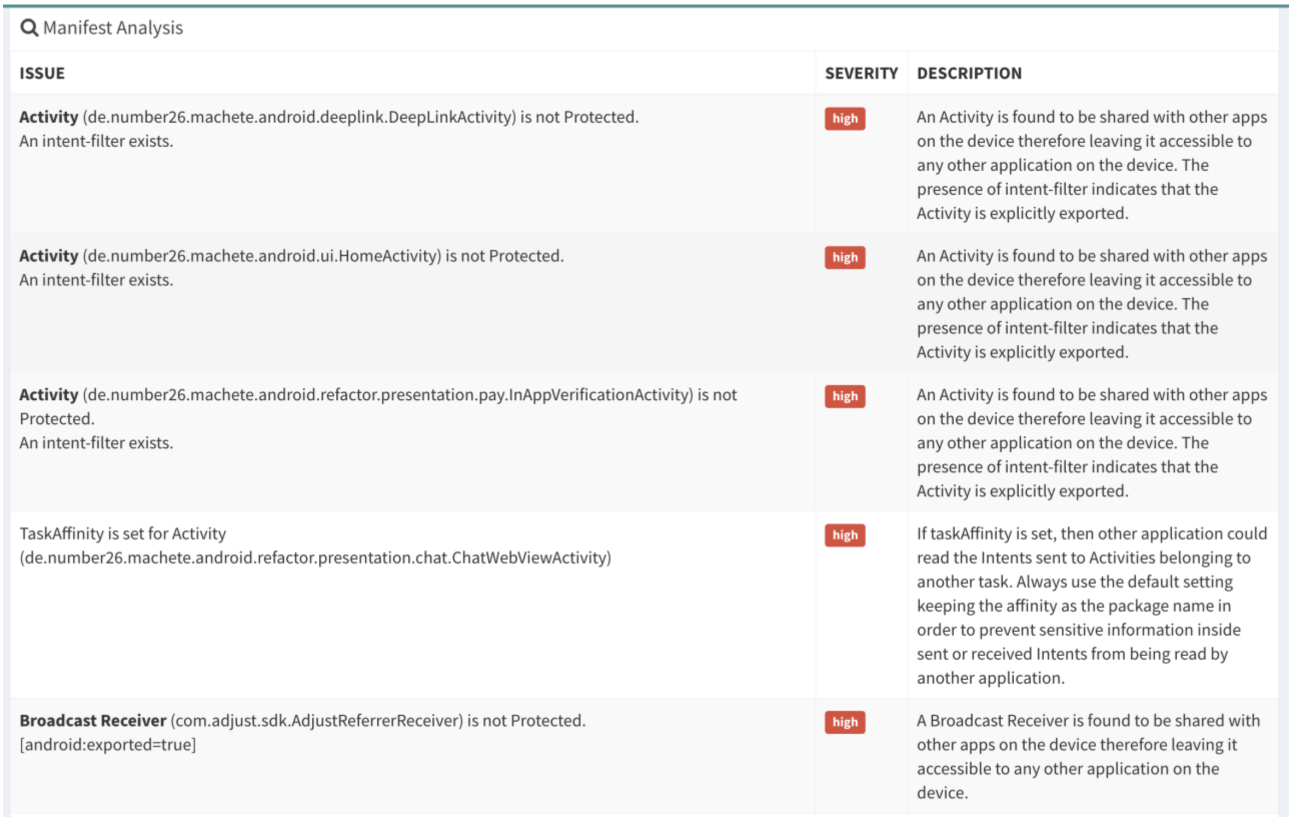
de.number26.android.permission.C2D_MESSAGE has a signature level. The mechanism, standing behind using this permission, should be tested, as the application can receive push messages. The data, read from message, should be verified before processing. (**Mobile Top 10 2016-M7-Poor Code Quality)**



Browsable Activities

ACTIVITY	INTENT
de.number26.machete.android.deeplink.DeepLinkActivity	**Schemes**: number26://, **Hosts:** main, transfer, moneybeam, invite_friends, statements, accountswitching, cash26, taxinfo, overdraft, statistics, controlcenter, transferwise, moneyrequest, consumercredit, savings, spaces, insurance, supportcenter, cashmap, metal, sellingscreen, **Path Patterns:** .*,

Browsable Activities section display de.number26.machete.android.deeplink.DeepLinkActivity.

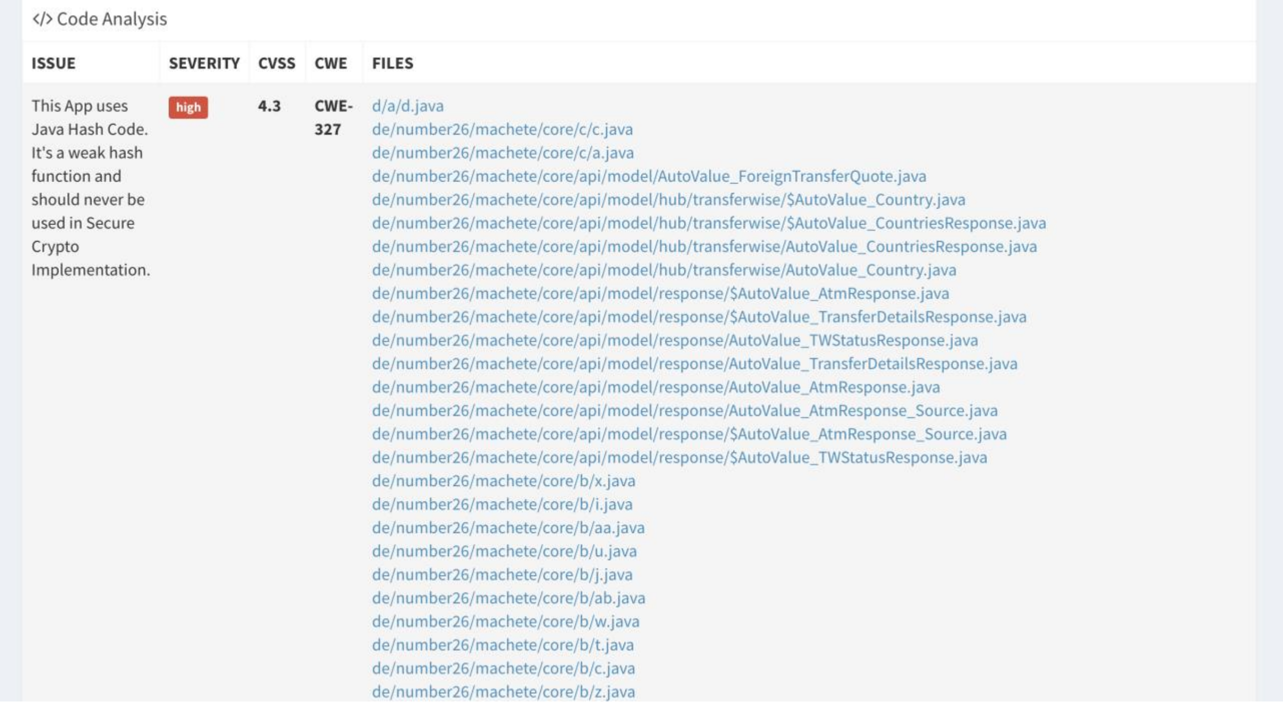


Manifest Analysis

ISSUE	SEVERITY	DESCRIPTION
Activity (de.number26.machete.android.deeplink.DeepLinkActivity) is not Protected. An intent-filter exists.	high	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.
Activity (de.number26.machete.android.ui.HomeActivity) is not Protected. An intent-filter exists.	high	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.
Activity (de.number26.machete.android.refactor.presentation.pay.InAppVerificationActivity) is not Protected. An intent-filter exists.	high	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.
TaskAffinity is set for Activity (de.number26.machete.android.refactor.presentation.chat.ChatWebViewActivity)	high	If taskAffinity is set, then other application could read the Intents sent to Activities belonging to another task. Always use the default setting keeping the affinity as the package name in order to prevent sensitive information inside sent or received Intents from being read by another application.
Broadcast Receiver (com.adjust.sdk.AdjustReferrerReceiver) is not Protected. [android:exported=true]	high	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.

Activity analysis shows potentially vulnerable activities, services and receivers. This knowledge will be used in analysis with drozer.

Code Analysis section will show us a long list of potential vulnerabilities. But keeping in mind that the code is obfuscated, this piece of report is of little use.

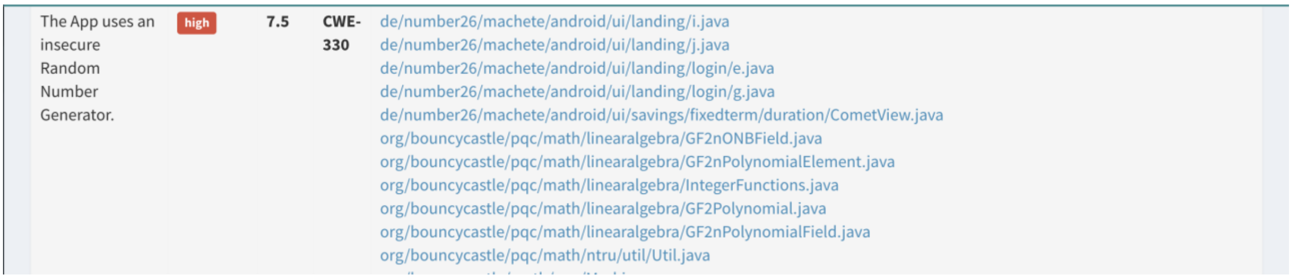


</> Code Analysis

ISSUE	SEVERITY	CVSS	CWE	FILES
This App uses Java Hash Code. It's a weak hash function and should never be used in Secure Crypto Implementation.	high	4.3	CWE-327	d/a/d.java de/number26/machete/core/c/c.java de/number26/machete/core/c/a.java de/number26/machete/core/api/model/AutoValue_ForeignTransferQuote.java de/number26/machete/core/api/model/hub/transferwise/$AutoValue_Country.java de/number26/machete/core/api/model/hub/transferwise/$AutoValue_CountriesResponse.java de/number26/machete/core/api/model/hub/transferwise/AutoValue_CountriesResponse.java de/number26/machete/core/api/model/hub/transferwise/AutoValue_Country.java de/number26/machete/core/api/model/response/$AutoValue_AtmResponse.java de/number26/machete/core/api/model/response/$AutoValue_TransferDetailsResponse.java de/number26/machete/core/api/model/response/AutoValue_TWStatusResponse.java de/number26/machete/core/api/model/response/AutoValue_TransferDetailsResponse.java de/number26/machete/core/api/model/response/AutoValue_AtmResponse.java de/number26/machete/core/api/model/response/AutoValue_AtmResponse_Source.java de/number26/machete/core/api/model/response/$AutoValue_AtmResponse_Source.java de/number26/machete/core/api/model/response/$AutoValue_TWStatusResponse.java de/number26/machete/core/b/x.java de/number26/machete/core/b/i.java de/number26/machete/core/b/aa.java de/number26/machete/core/b/u.java de/number26/machete/core/b/j.java de/number26/machete/core/b/ab.java de/number26/machete/core/b/w.java de/number26/machete/core/b/t.java de/number26/machete/core/b/c.java de/number26/machete/core/b/z.java

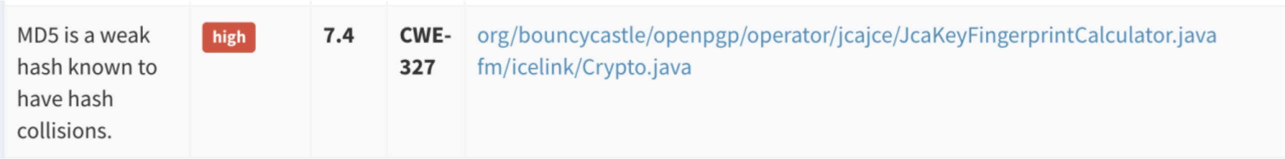
Yet, the first one found can be a sign of **Mobile Top 10 2016-M5-Insufficient Cryptography** vulnerability.

Another vulnerability is the quality of the random number generators, used to build hash sums, etc.



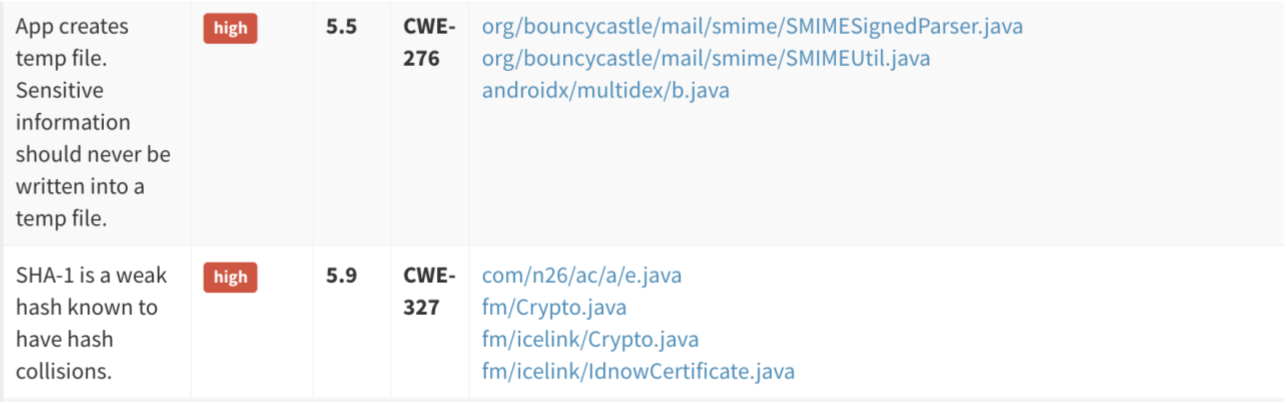
The App uses an insecure Random Number Generator.	high	7.5	CWE-330	de/number26/machete/android/ui/landing/i.java de/number26/machete/android/ui/landing/j.java de/number26/machete/android/ui/landing/login/e.java de/number26/machete/android/ui/landing/login/g.java de/number26/machete/android/ui/savings/fixedterm/duration/CometView.java org/bouncycastle/pqc/math/linearalgebra/GF2nONBField.java org/bouncycastle/pqc/math/linearalgebra/GF2nPolynomialElement.java org/bouncycastle/pqc/math/linearalgebra/IntegerFunctions.java org/bouncycastle/pqc/math/linearalgebra/GF2Polynomial.java org/bouncycastle/pqc/math/linearalgebra/GF2nPolynomialField.java org/bouncycastle/pqc/math/ntru/util/Util.java

Another **Mobile Top 10 2016-M5-Insufficient Cryptography** sign:



MD5 is a weak hash known to have hash collisions.	high	7.4	CWE-327	org/bouncycastle/openpgp/operator/jcajce/JcaKeyFingerprintCalculator.java fm/icelink/Crypto.java

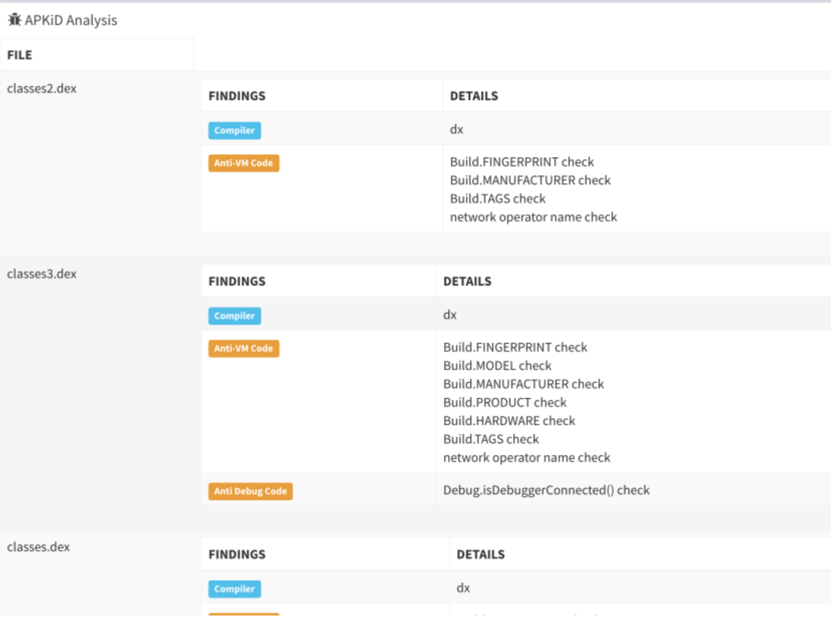
Application is storing data to external storage, so this might be a sensible point for Mobile Top 10 2016-M2-Insecure Data Storage.



App creates temp file. Sensitive information should never be written into a temp file.	high	5.5	CWE-276	org/bouncycastle/mail/smime/SMIMESignedParser.java org/bouncycastle/mail/smime/SMIMEUtil.java androidx/multidex/b.java
SHA-1 is a weak hash known to have hash collisions.	high	5.9	CWE-327	com/n26/ac/a/e.java fm/Crypto.java fm/icelink/Crypto.java fm/icelink/IdnowCertificate.java

Including the last one on the picture, we found 2 more vulnerabilities related to **Mobile Top 10 2016-M5-Insufficient Cryptography.**

APKiD Analysis shows that the application uses different anti VM code:



APKiD Analysis

FILE

classes2.dex

FINDINGS	DETAILS
Compiler	dx
Anti-VM Code	Build.FINGERPRINT check Build.MANUFACTURER check Build.TAGS check network operator name check

classes3.dex

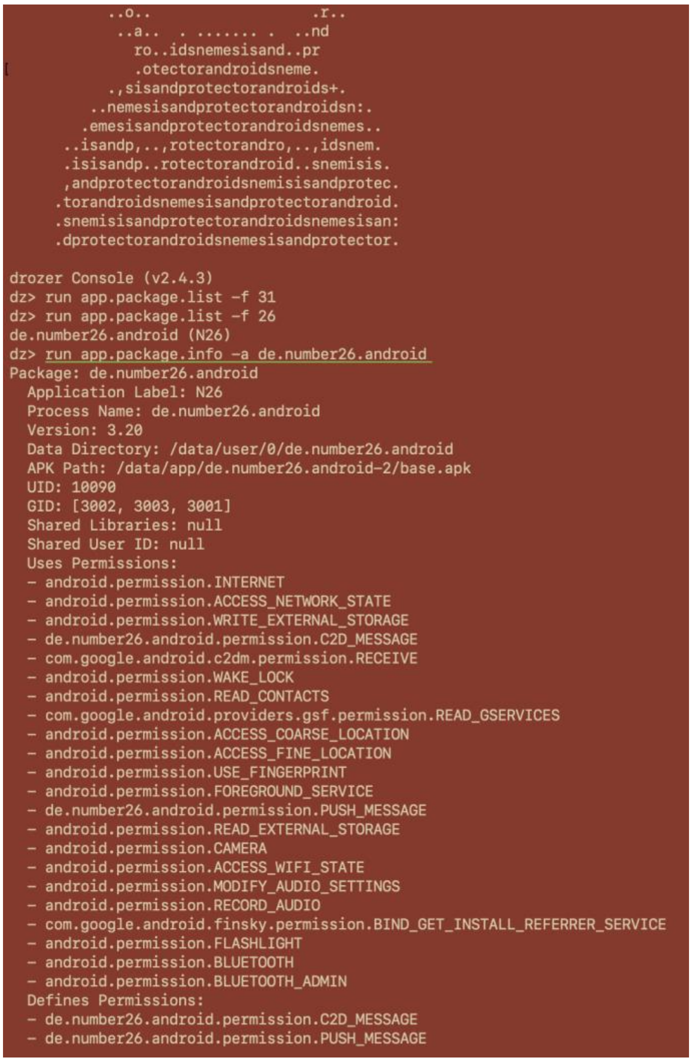
FINDINGS	DETAILS
Compiler	dx
Anti-VM Code	Build.FINGERPRINT check Build.MODEL check Build.MANUFACTURER check Build.PRODUCT check Build.HARDWARE check Build.TAGS check network operator name check
Anti Debug Code	Debug.isDebuggerConnected() check

classes.dex

FINDINGS	DETAILS
Compiler	dx

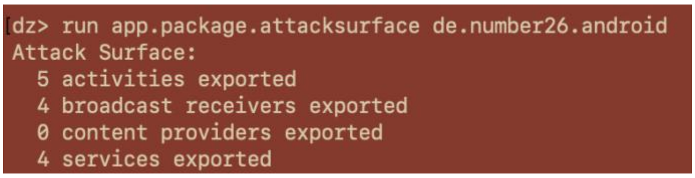
Drozer report analysis

For drozer analysis of N26 application we will start the same analysis path as for InsecureBankV2.



```
           ..o..                  .r..
            ..a..  . ....... .  ..nd
              ro..idsnemesisand..pr
              .otectorandroidsneme.
           .,sisandprotectorandroids+.
         ..nemesisandprotectorandroidsn:.
        .emesisandprotectorandroidsnemes..
      ..isandp,..,rotectorandro,..,idsnem.
      .isisandp..rotectorandroid..snemisis.
      ,andprotectorandroidsnemisisandprotec.
     .torandroidsnemesisandprotectorandroid.
     .snemisisandprotectorandroidsnemesisan:
     .dprotectorandroidsnemesisandprotector.

drozer Console (v2.4.3)
dz> run app.package.list -f 31
dz> run app.package.list -f 26
de.number26.android (N26)
dz> run app.package.info -a de.number26.android
Package: de.number26.android
  Application Label: N26
  Process Name: de.number26.android
  Version: 3.20
  Data Directory: /data/user/0/de.number26.android
  APK Path: /data/app/de.number26.android-2/base.apk
  UID: 10090
  GID: [3002, 3003, 3001]
  Shared Libraries: null
  Shared User ID: null
  Uses Permissions:
  - android.permission.INTERNET
  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.WRITE_EXTERNAL_STORAGE
  - de.number26.android.permission.C2D_MESSAGE
  - com.google.android.c2dm.permission.RECEIVE
  - android.permission.WAKE_LOCK
  - android.permission.READ_CONTACTS
  - com.google.android.providers.gsf.permission.READ_GSERVICES
  - android.permission.ACCESS_COARSE_LOCATION
  - android.permission.ACCESS_FINE_LOCATION
  - android.permission.USE_FINGERPRINT
  - android.permission.FOREGROUND_SERVICE
  - de.number26.android.permission.PUSH_MESSAGE
  - android.permission.READ_EXTERNAL_STORAGE
  - android.permission.CAMERA
  - android.permission.ACCESS_WIFI_STATE
  - android.permission.MODIFY_AUDIO_SETTINGS
  - android.permission.RECORD_AUDIO
  - com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE
  - android.permission.FLASHLIGHT
  - android.permission.BLUETOOTH
  - android.permission.BLUETOOTH_ADMIN
  Defines Permissions:
  - de.number26.android.permission.C2D_MESSAGE
  - de.number26.android.permission.PUSH_MESSAGE
```

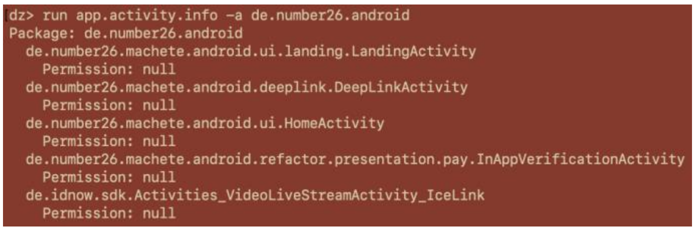
Define attack surface



```
dz> run app.package.attacksurface de.number26.android
Attack Surface:
  5 activities exported
  4 broadcast receivers exported
  0 content providers exported
  4 services exported
```

In this case, unlike InsecureBankV2, the application is not in debug mode. We would advise this kind of test to be included into prerelease verification, if not yet there.

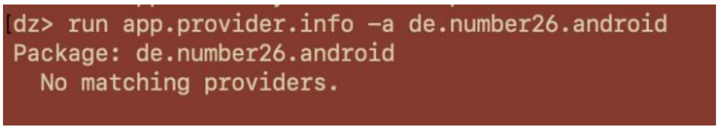
Launching activities:



```
dz> run app.activity.info -a de.number26.android
Package: de.number26.android
  de.number26.machete.android.ui.landing.LandingActivity
    Permission: null
  de.number26.machete.android.deeplink.DeepLinkActivity
    Permission: null
  de.number26.machete.android.ui.HomeActivity
    Permission: null
  de.number26.machete.android.refactor.presentation.pay.InAppVerificationActivity
    Permission: null
  de.idnow.sdk.Activities_VideoLiveStreamActivity_IceLink
    Permission: null
```

We tried to launch any activities, but the app either didn’t react on that, showing login screen, or the supplication stopped working. So at this stage we assume that the exported activities are not vulnerable. Yet, we think that another result may appear if this test is repeated with a test user logged in.

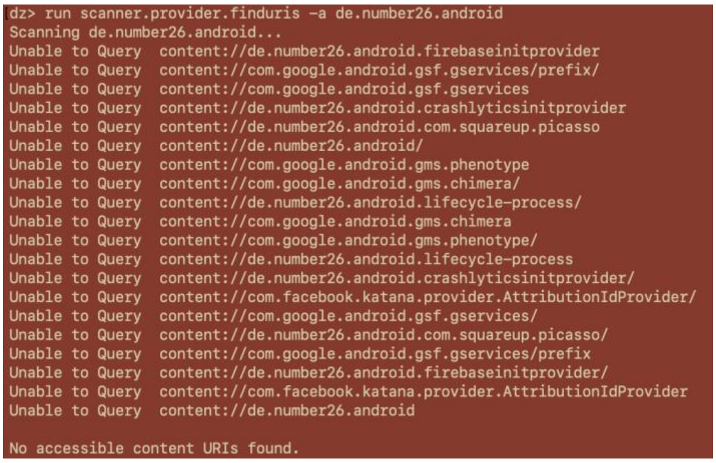
Analyzing content providers



```
[dz> run app.provider.info -a de.number26.android
Package: de.number26.android
  No matching providers.
```

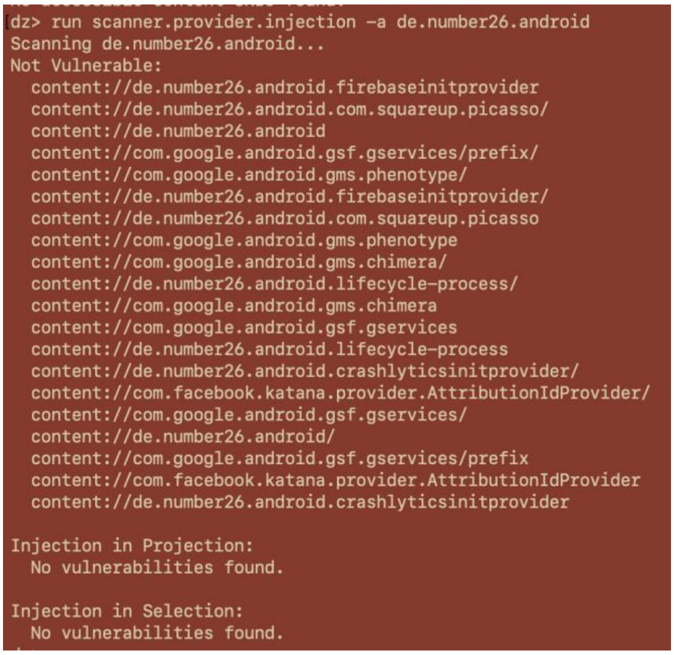
There are no vulnerable content providers in the application according to drozer.

Database-backed Content Providers



```
dz> run scanner.provider.finduris -a de.number26.android
Scanning de.number26.android...
Unable to Query  content://de.number26.android.firebaseinitprovider
Unable to Query  content://com.google.android.gsf.gservices/prefix/
Unable to Query  content://com.google.android.gsf.gservices
Unable to Query  content://de.number26.android.crashlyticsinitprovider
Unable to Query  content://de.number26.android.com.squareup.picasso
Unable to Query  content://de.number26.android/
Unable to Query  content://com.google.android.gms.phenotype
Unable to Query  content://com.google.android.gms.chimera/
Unable to Query  content://de.number26.android.lifecycle-process/
Unable to Query  content://com.google.android.gms.chimera
Unable to Query  content://com.google.android.gms.phenotype/
Unable to Query  content://de.number26.android.lifecycle-process
Unable to Query  content://de.number26.android.crashlyticsinitprovider/
Unable to Query  content://com.facebook.katana.provider.AttributionIdProvider/
Unable to Query  content://com.google.android.gsf.gservices/
Unable to Query  content://de.number26.android.com.squareup.picasso/
Unable to Query  content://com.google.android.gsf.gservices/prefix
Unable to Query  content://de.number26.android.firebaseinitprovider/
Unable to Query  content://com.facebook.katana.provider.AttributionIdProvider
Unable to Query  content://de.number26.android

No accessible content URIs found.
```

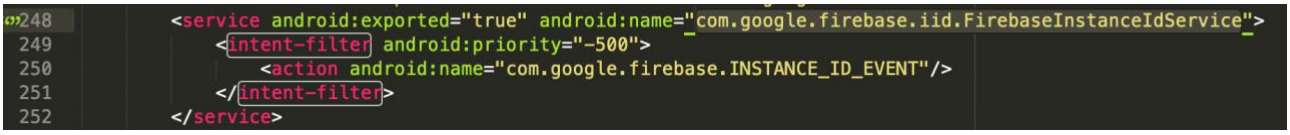
Content Provider Vulnerabilities



```
[dz> run scanner.provider.injection -a de.number26.android
Scanning de.number26.android...
Not Vulnerable:
  content://de.number26.android.firebaseinitprovider
  content://de.number26.android.com.squareup.picasso/
  content://de.number26.android
  content://com.google.android.gsf.gservices/prefix/
  content://com.google.android.gms.phenotype/
  content://de.number26.android.firebaseinitprovider/
  content://de.number26.android.com.squareup.picasso
  content://com.google.android.gms.phenotype
  content://com.google.android.gms.chimera/
  content://de.number26.android.lifecycle-process/
  content://com.google.android.gms.chimera
  content://com.google.android.gsf.gservices
  content://de.number26.android.lifecycle-process
  content://de.number26.android.crashlyticsinitprovider/
  content://com.facebook.katana.provider.AttributionIdProvider/
  content://com.google.android.gsf.gservices/
  content://de.number26.android/
  content://com.google.android.gsf.gservices/prefix
  content://com.facebook.katana.provider.AttributionIdProvider
  content://de.number26.android.crashlyticsinitprovider

Injection in Projection:
  No vulnerabilities found.

Injection in Selection:
  No vulnerabilities found.
```

According to what the drozer showed, we assume that the application is developed securely.

Manifest file analysis



```
<service android:exported="true" android:name="com.google.firebase.iid.FirebaseInstanceIdService">
    <intent-filter android:priority="-500">
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
```

FirebaseInstanceIdService class is deprecated.

Possible solution: https://medium.com/android-school/firebaseinstanceidservice-is-deprecated-50651f17a148

OWASP Mobile Top 10 classification

InsecureBankV2 application vulnerabilities found, classified according to OWASP Mobile Top 10:

- Mobile Top 10 2016-M1-Improper Platform Usage: 3;
- Mobile Top 10 2016-M2-Insecure Data Storage: 1;
- Mobile Top 10 2016-M4-Insecure Authentication: 1;
- Mobile Top 10 2016-M5-Insufficient Cryptography: 1;
- Mobile Top 10 2016-M6-Insecure Authorization: 1;
- Mobile Top 10 2016-M9-Reverse Engineering: 1;
- Mobile Top 10 2016-M10-Extraneous Functionality: 1.

No26Bank application vulnerabilities found, classified according to OWASP Mobile Top 10:

Mobile Top 10 2016-M2-Insecure Data Storage: 2;
Mobile Top 10 2016-M4-Insecure Authentication: 1;
Mobile Top 10 2016-M5-Insufficient Cryptography: 4;
Mobile Top 10 2016-M7-Poor Code Quality: 2,

Resource list:

1. https://oldbam.github.io/android/security/android-vulnerabilities-insecurebank-activities
2. https://medium.com/@ashrafrizvi3006/how-to-test-android-application-security-using-drozer-edc002c5dcac
3. https://github.com/rednaga/APKiD
4. https://www.cyberbit.com/blog/endpoint-security/anti-vm-and-anti-sandbox-explained/
5. https://www.owasp.org/index.php/Mobile_Top_10_2016-M4-Insecure_Authentication
6. https://blog.jayway.com/2009/09/24/the-browsable-category-revealed/