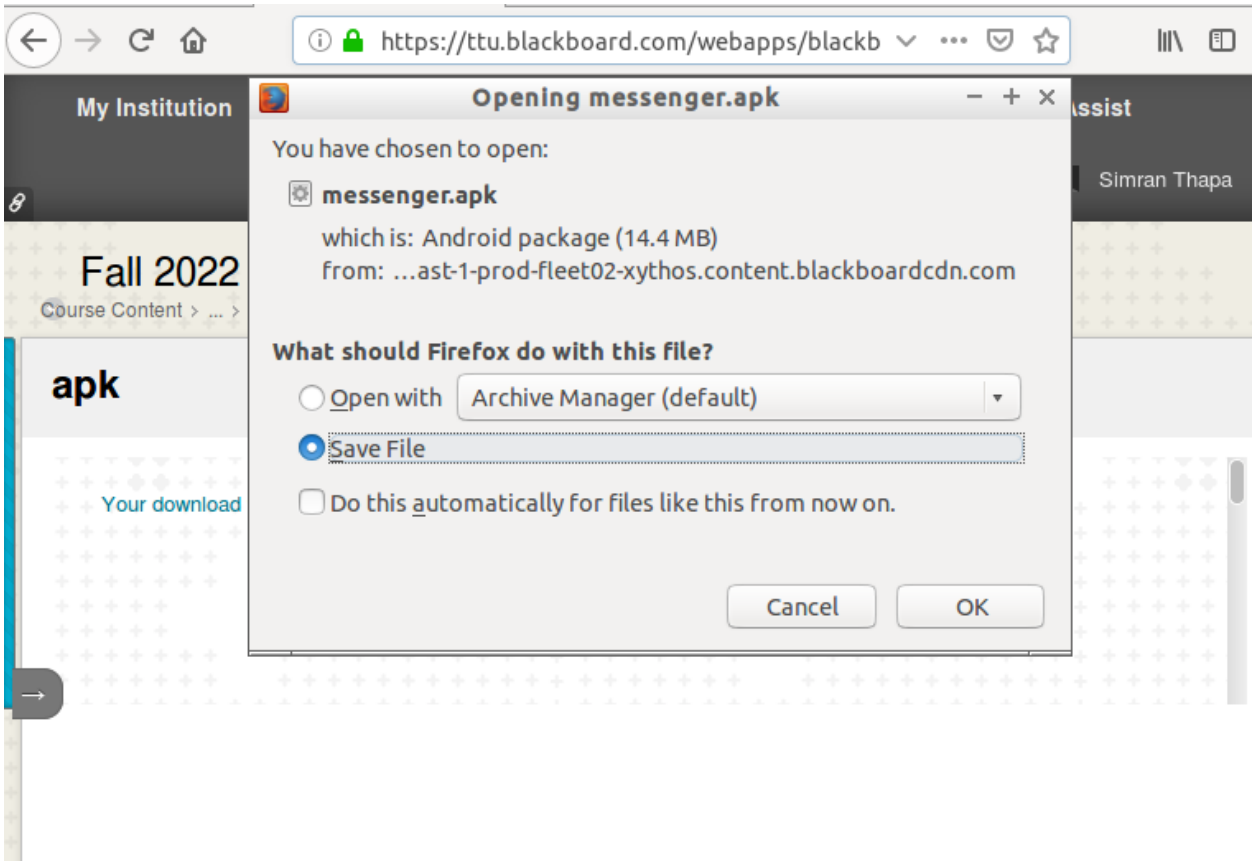
Malicious APK File Analysis No. 14

Steps and Screenshots

Steps:-

1) Download the malicious app from the blackboard.



2) unzip the messenger.apk to messenger_unzipped

```
santoku@santoku-VirtualBox:~/Downloads$ sudo unzip messenger.apk -d messenger_unzipped
```

```
  inflating: messenger_unzipped/assets/fbt/default/strings.bin
  inflating: messenger_unzipped/DebugProbesKt.bin
  inflating: messenger_unzipped/NOTICE
 extracting: messenger_unzipped/firebase-annotations.properties
 extracting: messenger_unzipped/firebase-common.properties
 extracting: messenger_unzipped/firebase-components.properties
  inflating: messenger_unzipped/firebase-iid-interop.properties
  inflating: messenger_unzipped/firebase-iid.properties
  inflating: messenger_unzipped/firebase-measurement-connector.properties
  inflating: messenger_unzipped/firebase-messaging.properties
  inflating: messenger_unzipped/play-services-ads-identifier.properties
  inflating: messenger_unzipped/play-services-analytics-impl.properties
  inflating: messenger_unzipped/play-services-analytics.properties
 extracting: messenger_unzipped/play-services-base.properties
 extracting: messenger_unzipped/play-services-basement.properties
  inflating: messenger_unzipped/play-services-location.properties
  inflating: messenger_unzipped/play-services-places-placereport.properties
  inflating: messenger_unzipped/play-services-stats.properties
  inflating: messenger_unzipped/play-services-tagmanager-v4-impl.properties
 extracting: messenger_unzipped/play-services-tasks.properties
 extracting: messenger_unzipped/okhttp3/internal/publicsuffix/publicsuffixes.gz

santoku@santoku-VirtualBox:~/Downloads$
```

3) checking the messenger_unzipped folder

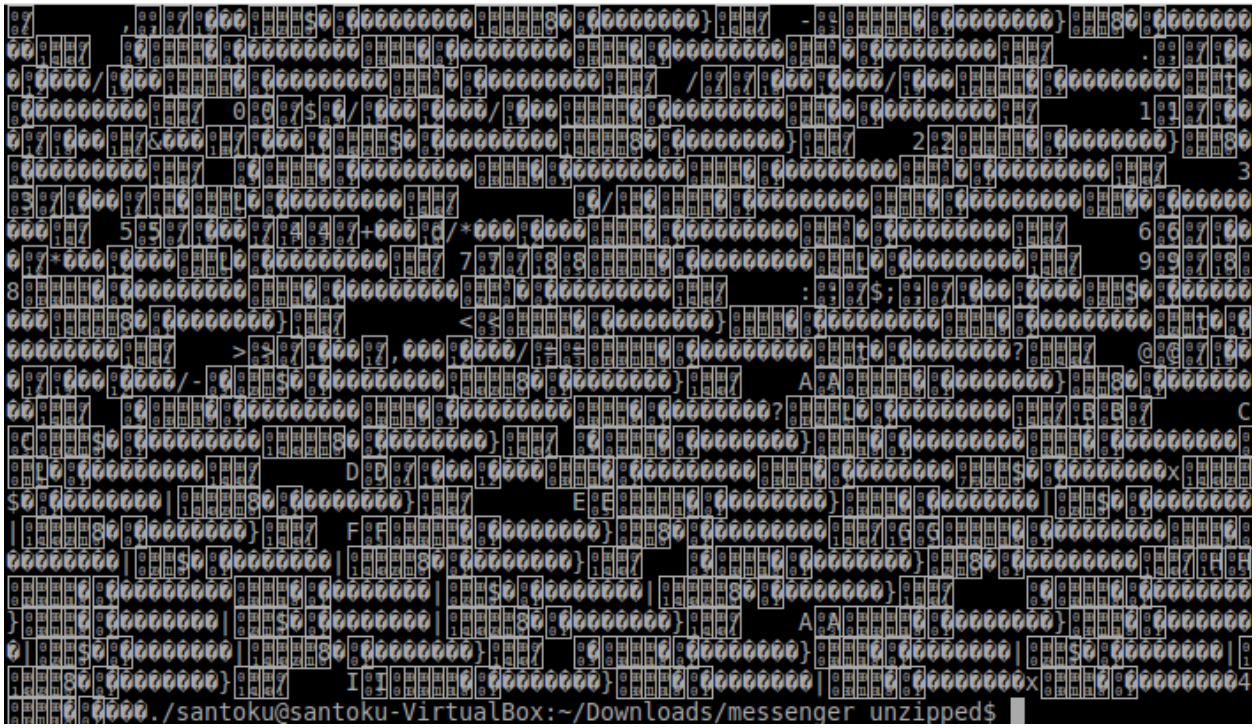
A general inspection of the messenger_unzipped folder shows that there are different files and folders. There are .properties file for firebase and play services.

```
santoku@santoku-VirtualBox:~/Downloads$ cd messenger_unzipped/
santoku@santoku-VirtualBox:~/Downloads/messenger_unzipped$ ls
AndroidManifest.xml                       NOTICE
assets                                    okhttp3
classes.dex                               play-services-ads-identifier.properties
DebugProbesKt.bin                         play-services-analytics-impl.properties
firebase-annotations.properties           play-services-analytics.properties
firebase-common.properties                play-services-basement.properties
firebase-components.properties            play-services-base.properties
firebase-iid-interop.properties           play-services-location.properties
firebase-iid.properties                   play-services-places-placereport.properties
firebase-measurement-connector.properties play-services-stats.properties
firebase-messaging.properties             play-services-tagmanager-v4-impl.properties
kotlin                                    play-services-tasks.properties
lib                                       res
META-INF                                  resources.arsc
santoku@santoku-VirtualBox:~/Downloads/messenger_unzipped$
```

4) Deobfuscation of the sample

Now to check if file is obfuscated or not we can check the AndroidManifest.xml file.

```
santoku@santoku-VirtualBox:~/Downloads/messenger_unzipped$ cat AndroidManifest.xml
```



As we can see that the .xml file is in encoded form and we cannot read it. Hence the file is obfuscated. Now to deobfuscate it we can use apktool .

```
┌──(simran㉿kali)-[~/Downloads]
└─$ apktool d messenger.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1 on messenger.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/simran/.local/share/apktool/framework/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...

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

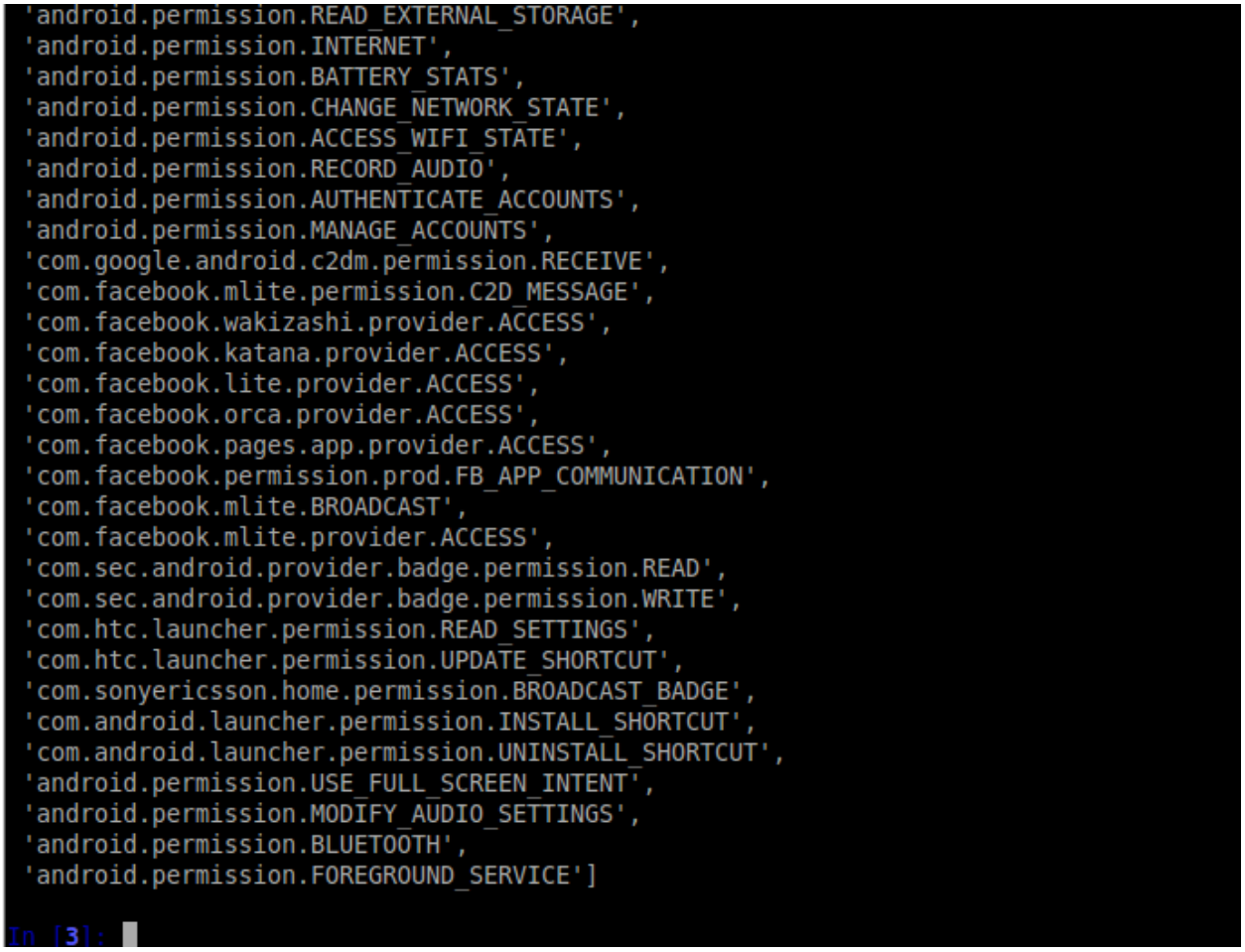
5) Permissions the app needs

We can use androlyze.py to see the permissions the app needs.

```
santoku@santoku-VirtualBox:~/Downloads$ /usr/share/androguard/androlyze.py -s
/usr/lib/python2.7/dist-packages/IPython/frontend.py:30: UserWarning: The top-le
vel `frontend` package has been deprecated. All its subpackages have been moved
to the top `IPython` level.
  warn("The top-level `frontend` package has been deprecated. "
Androlyze version 2.0
In [1]: a,d,dx = AnalyzeAPK("messenger.apk",decompiler="dad")

In [2]:
```

Run a.get_permissions() to get all permissions

```
In [1]: a,d,dx = AnalyzeAPK("messenger.apk",decompiler="dad")

In [2]: a.get_permissions()
Out[2]:
['android.permission.ACCESS_FINE_LOCATION',
 'android.permission.WRITE_SETTINGS',
 'android.permission.SEND_SMS',
 'android.permission.CHANGE_WIFI_STATE',
 'android.permission.WRITE_CALL_LOG',
 'android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS',
 'android.permission.RECEIVE_SMS',
 'android.permission.SET_WALLPAPER',
 'android.permission.ACCESS_COARSE_LOCATION',
 'android.permission.WRITE_CONTACTS',
 'android.permission.READ_SMS',
 'android.permission.READ_CALL_LOG',
 'android.permission.CALL_PHONE',
 'android.permission.READ_CONTACTS',
 'android.permission.READ_PROFILE',
 'android.permission.READ_PHONE_STATE',
 'android.permission.READ_PHONE_NUMBERS',
 'android.permission.RECEIVE_BOOT_COMPLETED',
 'android.permission.ACCESS_NETWORK_STATE',
 'android.permission.WRITE_EXTERNAL_STORAGE',
 'android.permission.VIBRATE',
 'android.permission.GET_ACCOUNTS',
 'android.permission.WAKE_LOCK',
 'android.permission.CAMERA',
 'android.permission.READ_EXTERNAL_STORAGE',
 'android.permission.INTERNET',
 'android.permission.BATTERY_STATS',
```



Likewise run a.get_details_permissions() to get more details of the permissions.

```
In [3]: a.get_details_permissions()
Out[3]:
{'android.permission.ACCESS_COARSE_LOCATION': ['dangerous',
  'coarse (network-based) location',
  'Access coarse location sources, such as the mobile network database, to determine a
n approximate phone location, where available. Malicious applications can use this to
determine approximately where you are.'],
 'android.permission.ACCESS_FINE_LOCATION': ['dangerous',
  'fine (GPS) location',
  'Access fine location sources, such as the Global Positioning System on the phone, w
here available. Malicious applications can use this to determine where you are and may
 consume additional battery power.'],
 'android.permission.ACCESS_NETWORK_STATE': ['normal',
  'view network status',
  'Allows an application to view the status of all networks.'],
 'android.permission.ACCESS_WIFI_STATE': ['normal',
  'view Wi-Fi status',
  'Allows an application to view the information about the status of Wi-Fi.'],
 'android.permission.AUTHENTICATE_ACCOUNTS': ['dangerous',
  'act as an account authenticator',
  'Allows an application to use the account authenticator capabilities of the Account
Manager, including creating accounts as well as obtaining and setting their passwords.
'],
 'android.permission.BATTERY_STATS': ['dangerous',
  'modify battery statistics',
  'Allows the modification of collected battery statistics. Not for use by normal appl
ications.'],
 'android.permission.BLUETOOTH': ['dangerous',
```

```
 'android.permission.SET_WALLPAPER': ['normal',
  'set wallpaper',
  'Allows the application to set the system wallpaper.'],
 'android.permission.USE_FULL_SCREEN_INTENT': ['normal',
  'Unknown permission from android reference',
  'Unknown permission from android reference'],
 'android.permission.VIBRATE': ['normal',
  'control vibrator',
  'Allows the application to control the vibrator.'],
 'android.permission.WAKE_LOCK': ['normal',
  'prevent phone from sleeping',
  'Allows an application to prevent the phone from going to sleep.'],
 'android.permission.WRITE_CALL_LOG': ['dangerous',
  "write (but not read) the user's contacts data.",
  "Allows an application to write (but not read) the user's contacts data."],
 'android.permission.WRITE_CONTACTS': ['dangerous',
  'write contact data',
  'Allows an application to modify the contact (address) data stored on your phone. Ma
licious applications can use this to erase or modify your contact data.'],
 'android.permission.WRITE_EXTERNAL_STORAGE': ['dangerous',
  'modify/delete SD card contents',
  'Allows an application to write to the SD card.'],
 'android.permission.WRITE_SETTINGS': ['normal',
  'modify global system settings',
  "Allows an application to modify the system's settings data. Malicious applications
can corrupt your system's configuration."],
 'com.android.launcher.permission.INSTALL_SHORTCUT': ['normal',
  'Unknown permission from android reference',
  'Unknown permission from android reference'],
 'com.android.launcher.permission.UNINSTALL_SHORTCUT': ['normal',
  'Unknown permission from android reference',
```

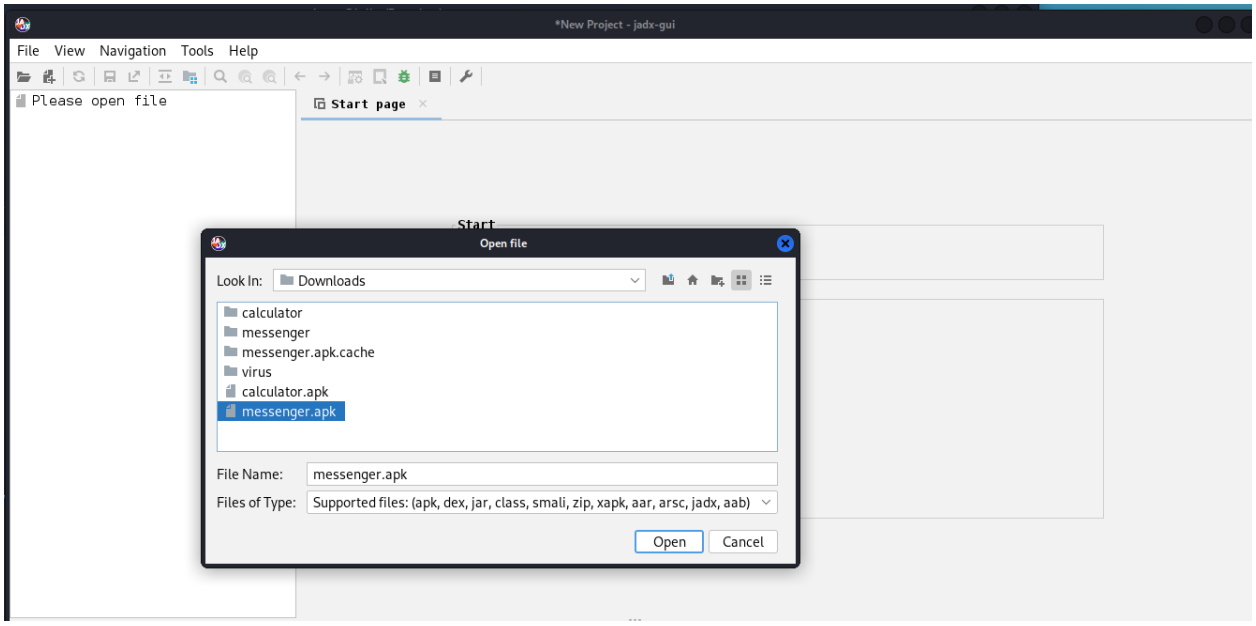
6) activities listed inside apk

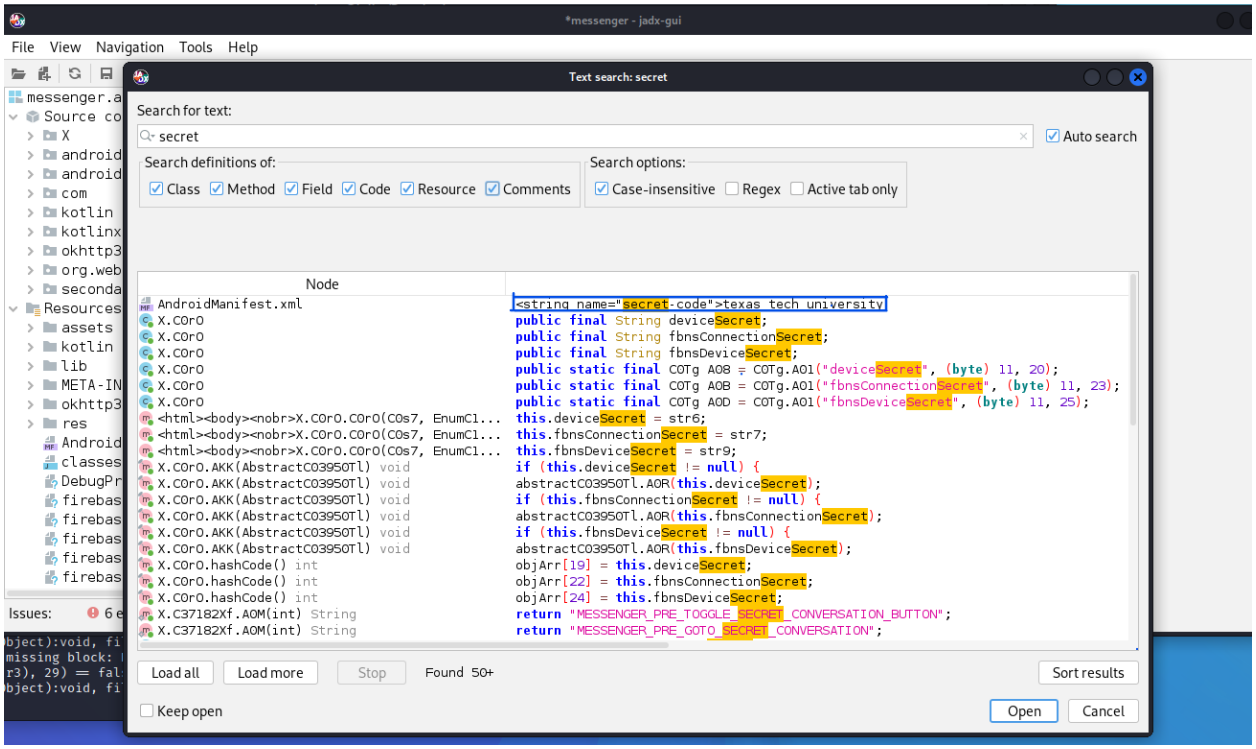
We can use a.get_activites() to get all the activities of the app.

```
In [4]: a.get_activities()
Out[4]:
['com.facebook.mlite.coreui.view.MainActivity',
 'com.facebook.mlite.sso.view.LoginActivity',
 'com.facebook.mlite.threadview.view.ThreadViewActivity',
 'com.facebook.mlite.messagerequests.view.MessageRequestsActivity',
 'com.facebook.mlite.messagerequests.view.FilteredRequestsActivity',
 'com.facebook.mlite.share.view.ShareActivity',
 'com.facebook.mlite.composer.view.ComposerActivity',
 'com.facebook.mlite.mediaview.view.MediaViewActivity',
 'com.facebook.mlite.update.view.ApkUpdateActivity',
 'com.facebook.mlite.bugreporter.view.BugReporterActivity',
 'com.facebook.mlite.zero.optin.MLiteZeroOptinInterstitial',
 'com.facebook.mlite.accounts.view.AccountsActivity',
 'com.facebook.mlite.gdpr.view.GdprConsentActivity',
 'com.facebook.mlite.gdpr.view.GdprControlCenterActivity',
 'com.facebook.mlite.intenthandling.IntentHandlerActivity',
 'com.facebook.mlite.intenthandling.SecureIntentHandlerActivity',
 'com.facebook.mlite.privacyflowtrigger.view.PrivacyFlowTriggerActivity',
 'com.facebook.mlite.rtc.view.CallActivity',
 'com.facebook.mlite.lowdisk.view.LowDiskSpaceActivity',
 'com.facebook.mlite.util.app.ProcessRestart',
 'com.facebook.mlite.notify.action.MLiteNotificationThreadMuteDialogActivity',
 'com.facebook.mlite.nux.lib.implementation.NuxActivity',
 'com.google.android.gms.common.api.GoogleApiActivity',
 'com.facebook.mlite.composer.view.CallComposerActivity',
 'com.facebook.mlite.frx.web.view.FrxReportActivity',
 'com.facebook.mlite.threadview.view.ParticipantsActivity',
 'com.facebook.mlite.presence.pref.view.PresencePreferenceActivity',
 'com.facebook.mlite.story.viewer.StoryViewerActivity',
 'com.facebook.mlite.threadcustomization.view.NicknamesActivity',
 'com.facebook.mlite.camera.view.CameraActivity',
 'com.facebook.mlite.policies.view.thirdpartynotices.ThirdPartyNoticesActivity']

In [5]:
```

7) Secret Code

We can use jadx-gui tool to analyze apk and search the apk file.



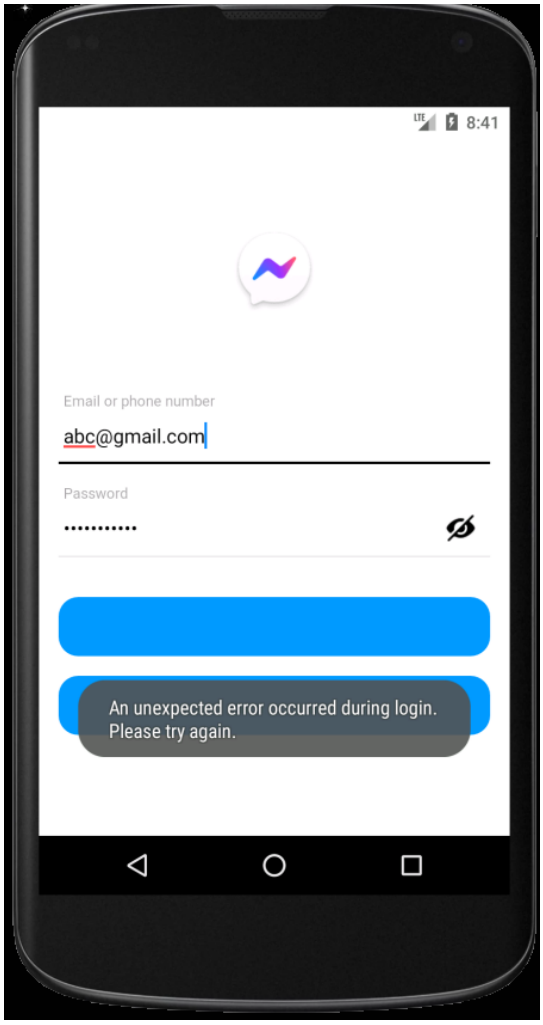


Here the secret code is **texas tech university** and has been stored in AndroidManifest.xml file.

8) What the normal part of the code does?

Now to check what the normal part of the apk does we install the apk in the android emulator.

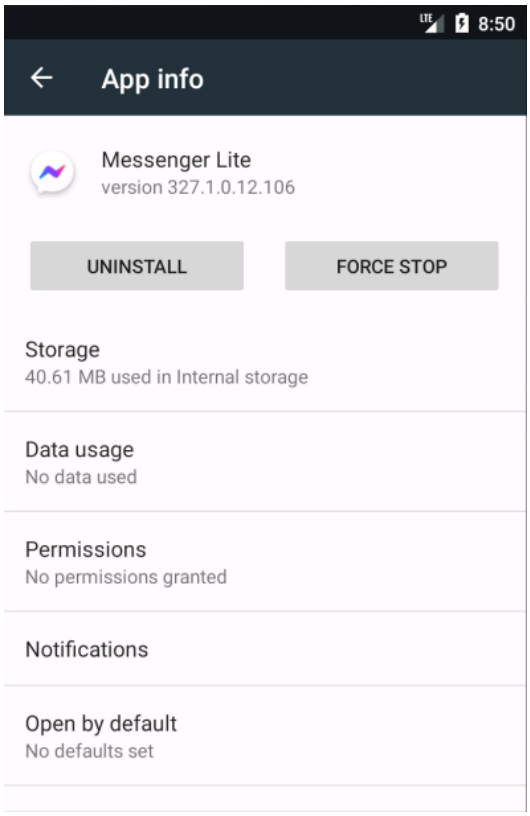
```
PS C:\Users\Manish Wagle\Downloads> adb devices
List of devices attached
emulator-5554   device

PS C:\Users\Manish Wagle\Downloads> adb install messenger.apk
Performing Streamed Install
Success
PS C:\Users\Manish Wagle\Downloads>
```

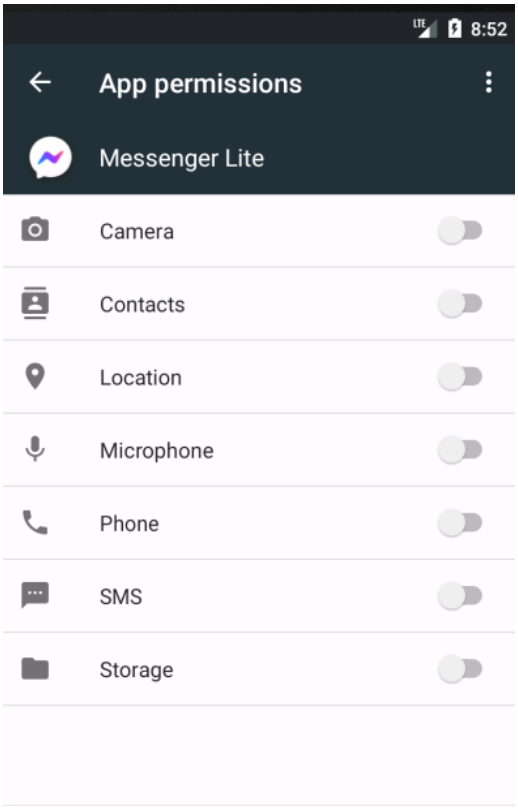


The apk file given did not install in newer android versions. To install the apk we had to create emulator for android 7 and install the apk.

As we can see from screenshot above, it seems like the apk is for facebook messenger lite app. Since we donot know what the malicious part of the apk does, we didnot enter any personal detail there.

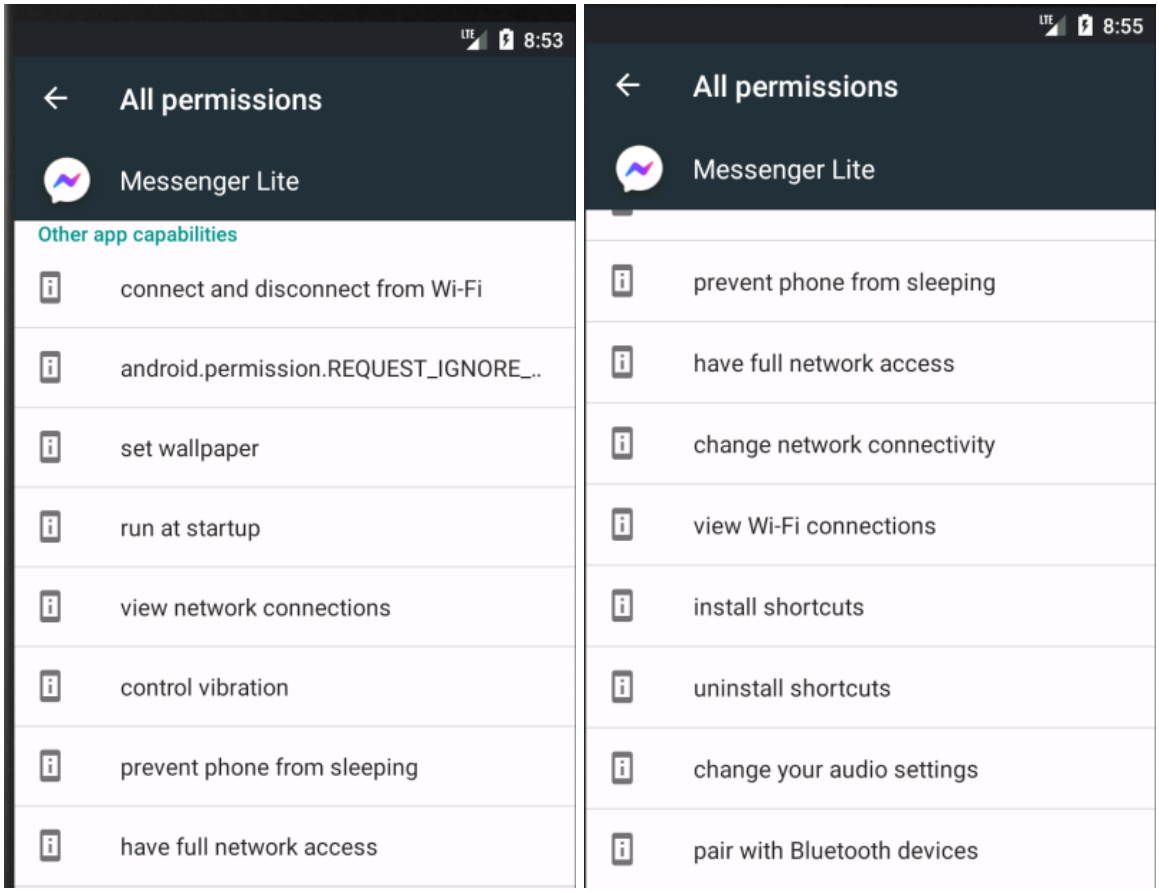
Below are screenshots of detail of the apk from app settings and list of permission the app ask for.



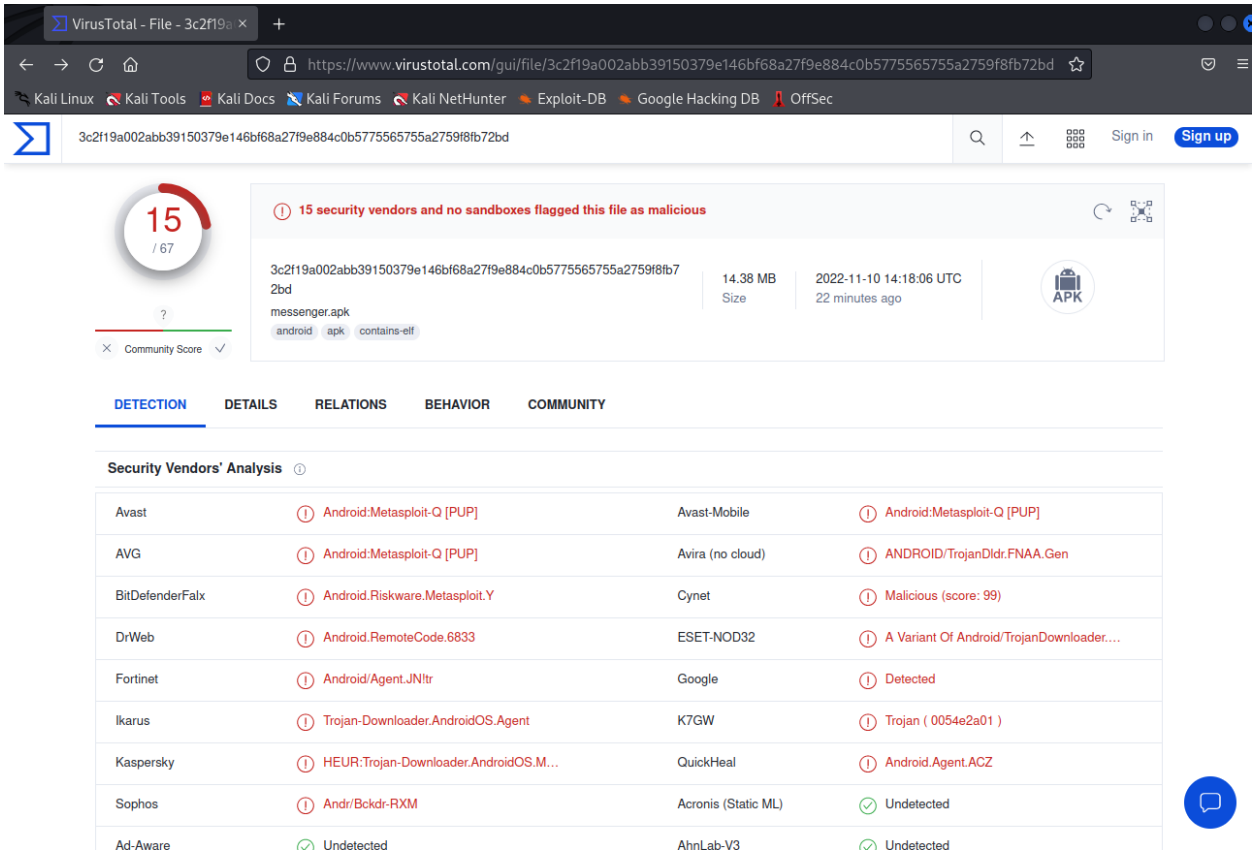


9) What the malicous part of the code does?

Now if we check all the permissions the app needs and it's other capabilities we can see that there are some scary things the app can do.



Uploading the malicious app onto the virustotal and analyzing, we can find out that it contains trojans and metasploits.

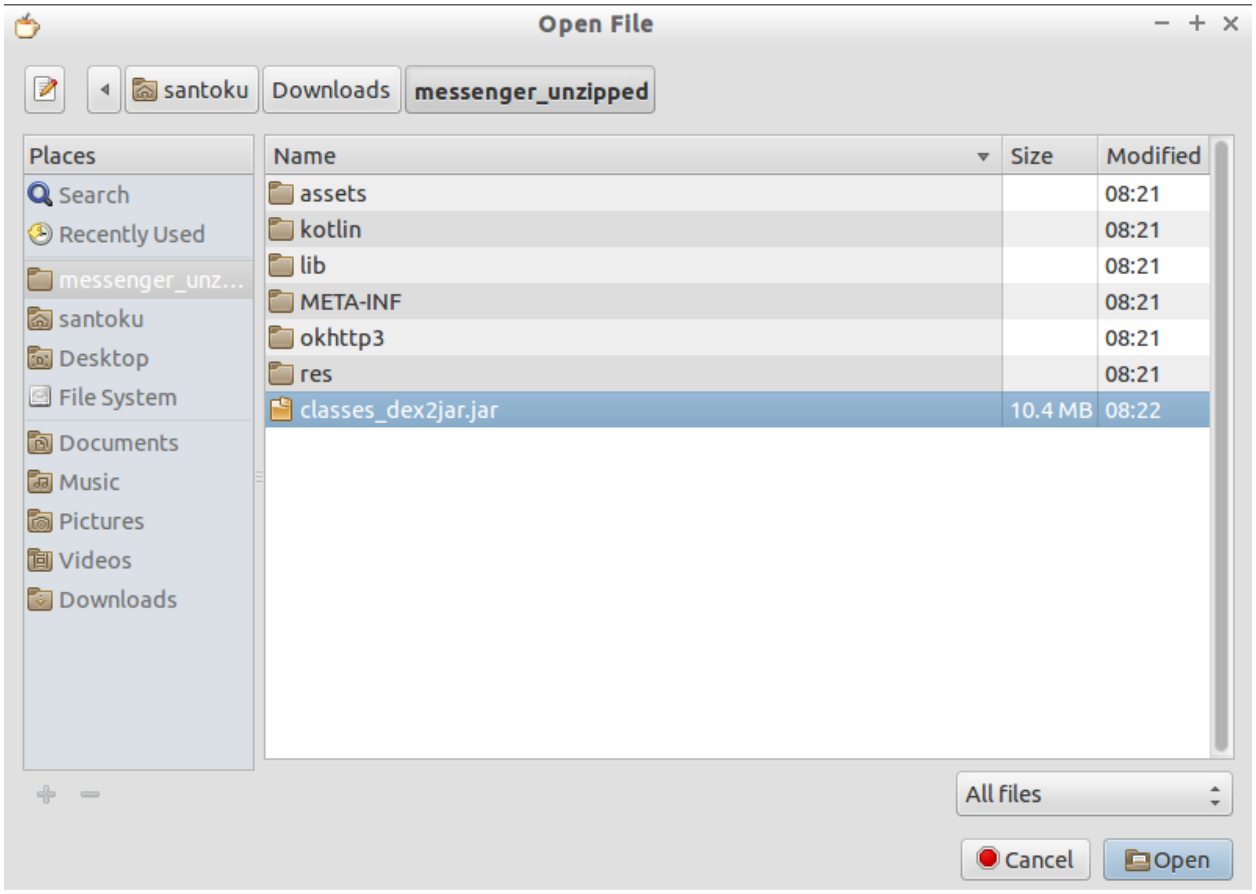


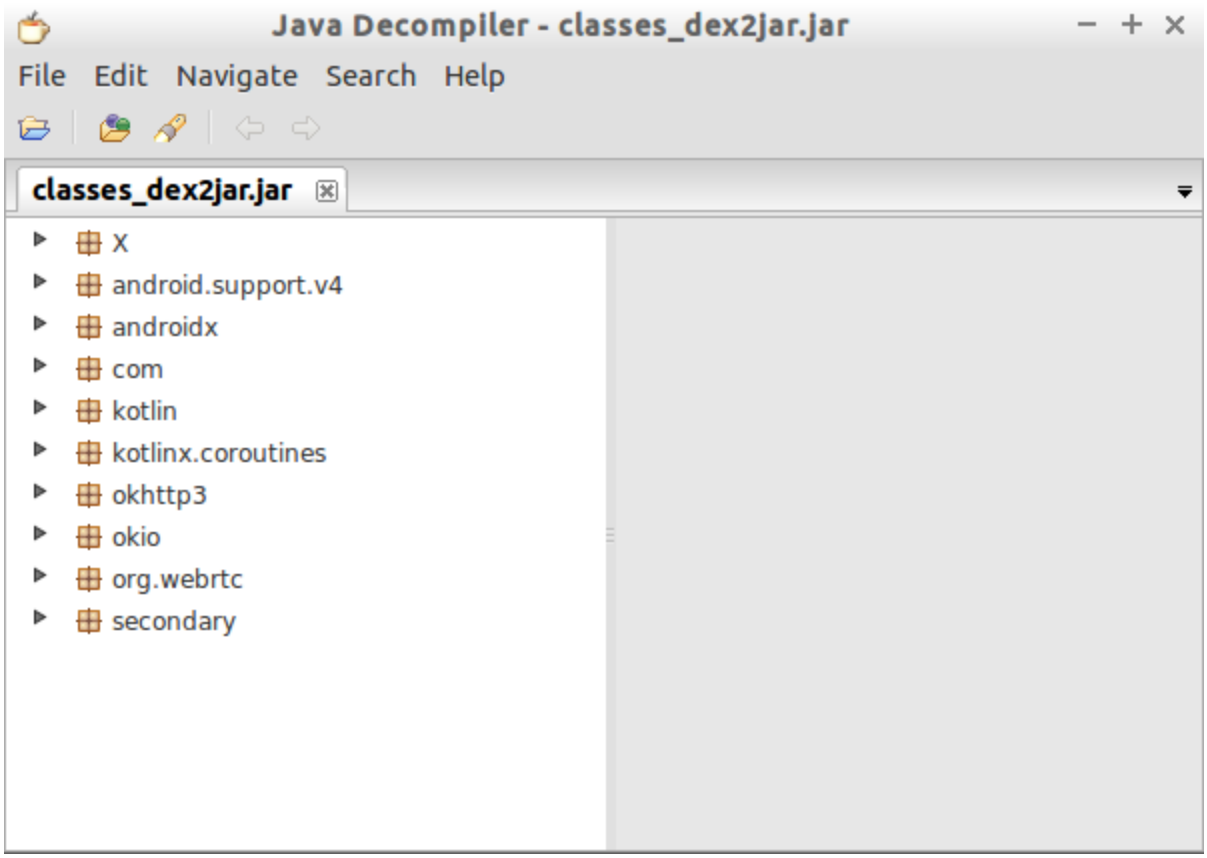
It seems metasploit has been used in this app along with some trojans attached to it.

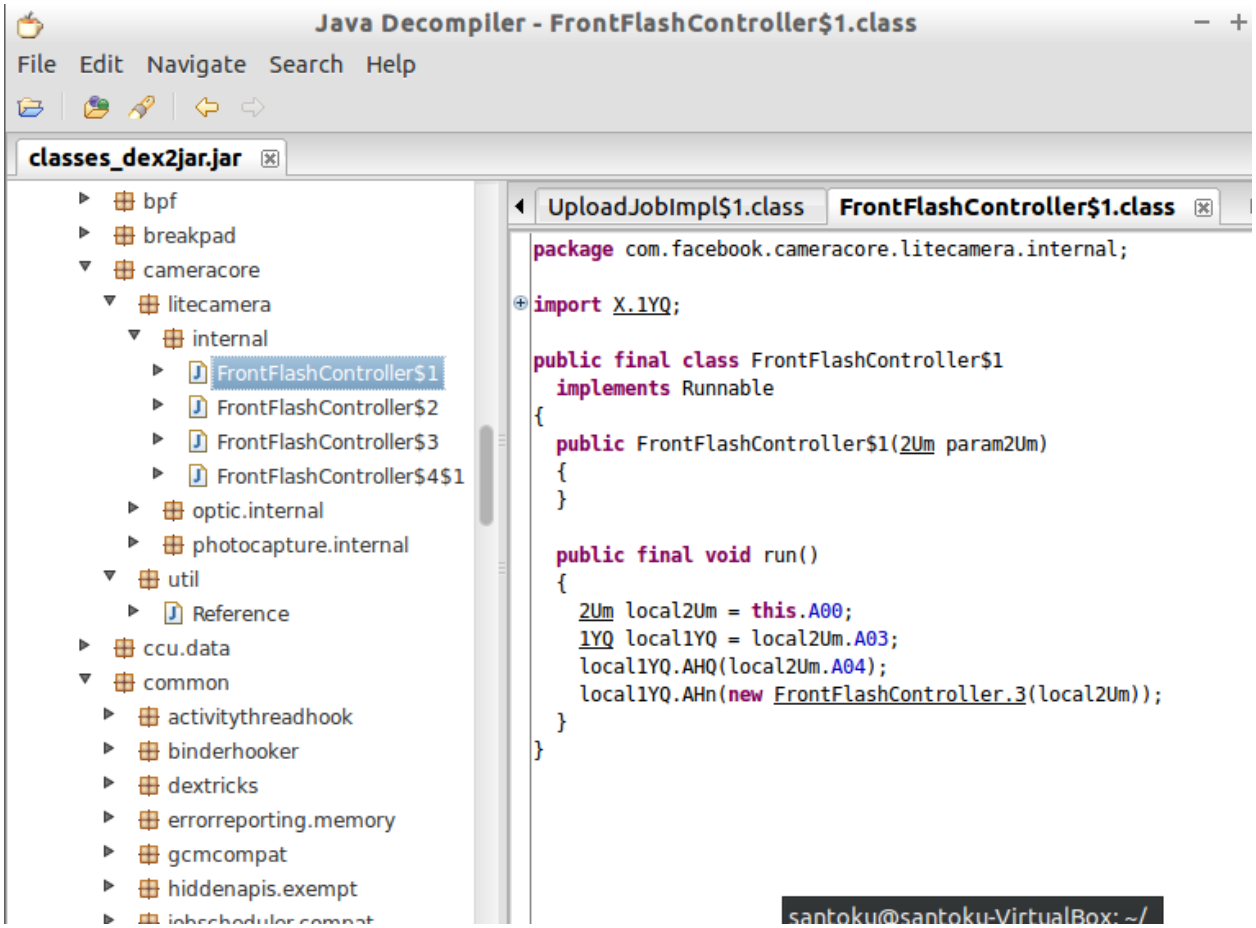
10) dissected Java code using JD-GUI

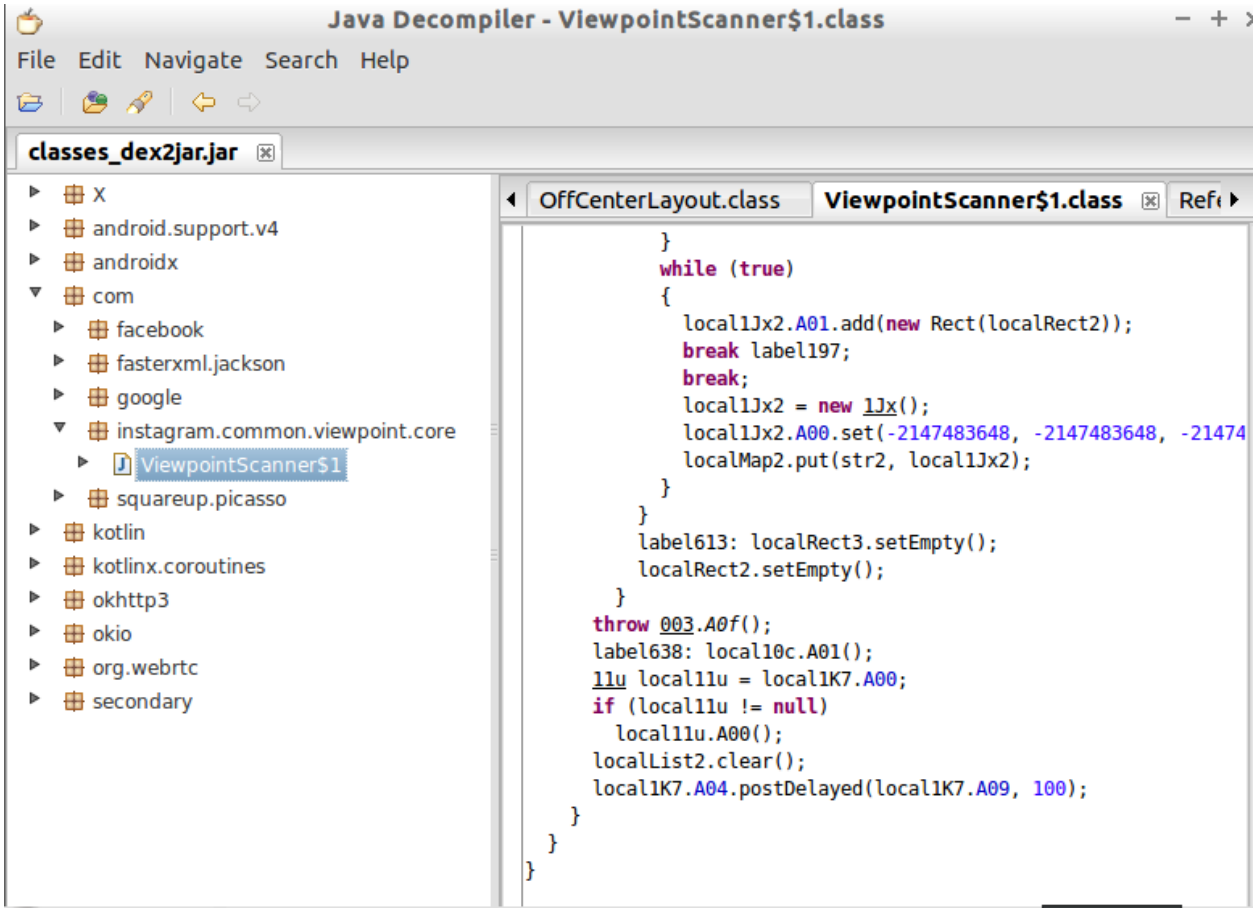
We can run command dex2jar classes.dex inside the virus_unzipped folder to convert the dex file to jar file. The jar file will then be open in JD-GUI to further analyzed.

```
santoku@santoku-VirtualBox:~/Downloads$ ls
messenger  messenger.apk  messenger_unzipped
santoku@santoku-VirtualBox:~/Downloads$ cd messenger_unzipped/
santoku@santoku-VirtualBox:~/Downloads/messenger_unzipped$ dex2jar classes.dex
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar classes.dex -> classes_dex2jar.jar
Done.
santoku@santoku-VirtualBox:~/Downloads/messenger_unzipped$
```


Open File
santoku
Downloads
messenger_unzipped
Places
Search
Recently Used
messenger_unz...
santoku
Desktop
File System
Documents
Music
Pictures
Videos
Downloads
Name
Size
Modified
assets
08:21
kotlin
08:21
lib
08:21
META-INF
08:21
okhttp3
08:21
res
08:21
classes_dex2jar.jar
10.4 MB
08:22
All files
Cancel
Open


Java Decompiler - classes_dex2jar.jar
File
Edit
Navigate
Search
Help
classes_dex2jar.jar
X
android.support.v4
androidx
com
kotlin
kotlinx.coroutines
okhttp3
okio
org.webrtc
secondary


Java Decompiler - FrontFlashController$1.class
File Edit Navigate Search Help
classes_dex2jar.jar
bpf
breakpad
cameracore
litecamera
internal
FrontFlashController$1
FrontFlashController$2
FrontFlashController$3
FrontFlashController$4$1
optic.internal
photocapture.internal
util
Reference
ccu.data
common
activitythreadhook
binderhooker
dextricks
errorreporting.memory
gcmcompat
hiddenapis.exempt
UploadJobImpl$1.class
FrontFlashController$1.class
package com.facebook.cameracore.litecamera.internal;
import X.1YQ;
public final class FrontFlashController$1
implements Runnable
{
public FrontFlashController$1(2Um param2Um)
{
}
public final void run()
{
2Um local2Um = this.A00;
1YQ local1YQ = local2Um.A03;
local1YQ.AHQ(local2Um.A04);
local1YQ.AHn(new FrontFlashController.3(local2Um));
}
}
santoku@santoku-VirtualBox: ~/


Java Decompiler - ViewpointScanner$1.class
File Edit Navigate Search Help
classes_dex2jar.jar
X
android.support.v4
androidx
com
facebook
fasterxml.jackson
google
instagram.common.viewpoint.core
ViewpointScanner$1
squareup.picasso
kotlin
kotlinx.coroutines
okhttp3
okio
org.webrtc
secondary
OffCenterLayout.class
ViewpointScanner$1.class
while (true)
{
local1Jx2.A01.add(new Rect(localRect2));
break label197;
break;
local1Jx2 = new 1Jx();
local1Jx2.A00.set(-2147483648, -2147483648, -21474
localMap2.put(str2, local1Jx2);
label613: localRect3.setEmpty();
localRect2.setEmpty();
throw 003.A0f();
label638: local10c.A01();
11u local11u = local1K7.A00;
if (local11u != null)
local11u.A00();
localList2.clear();
local1K7.A04.postDelayed(local1K7.A09, 100);