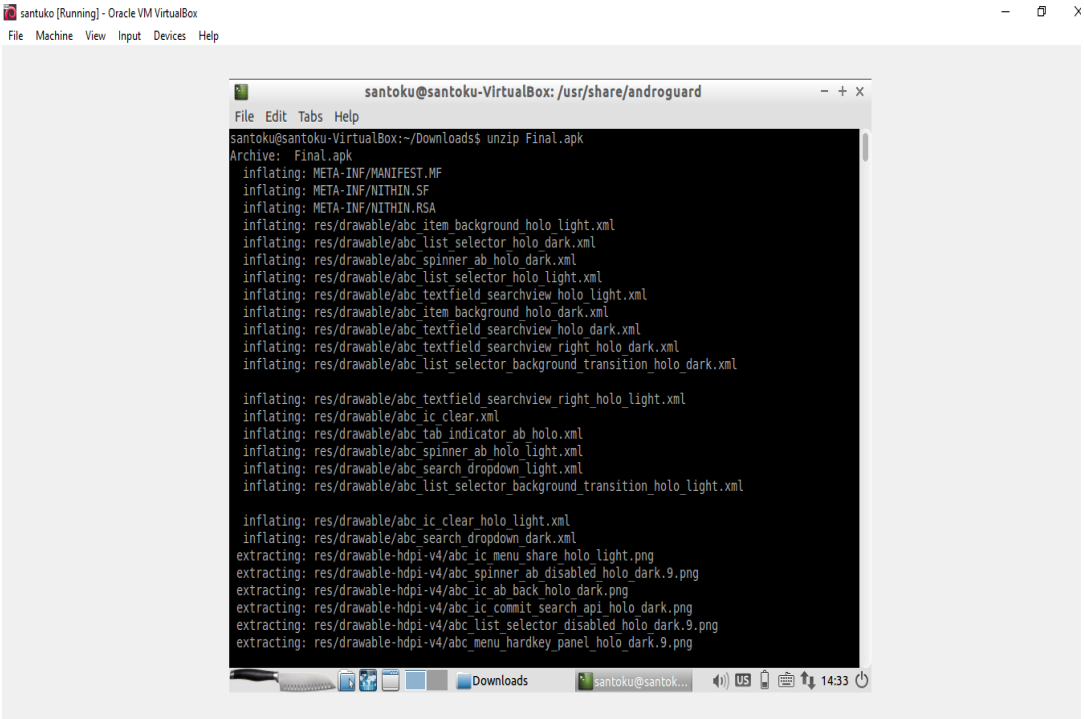
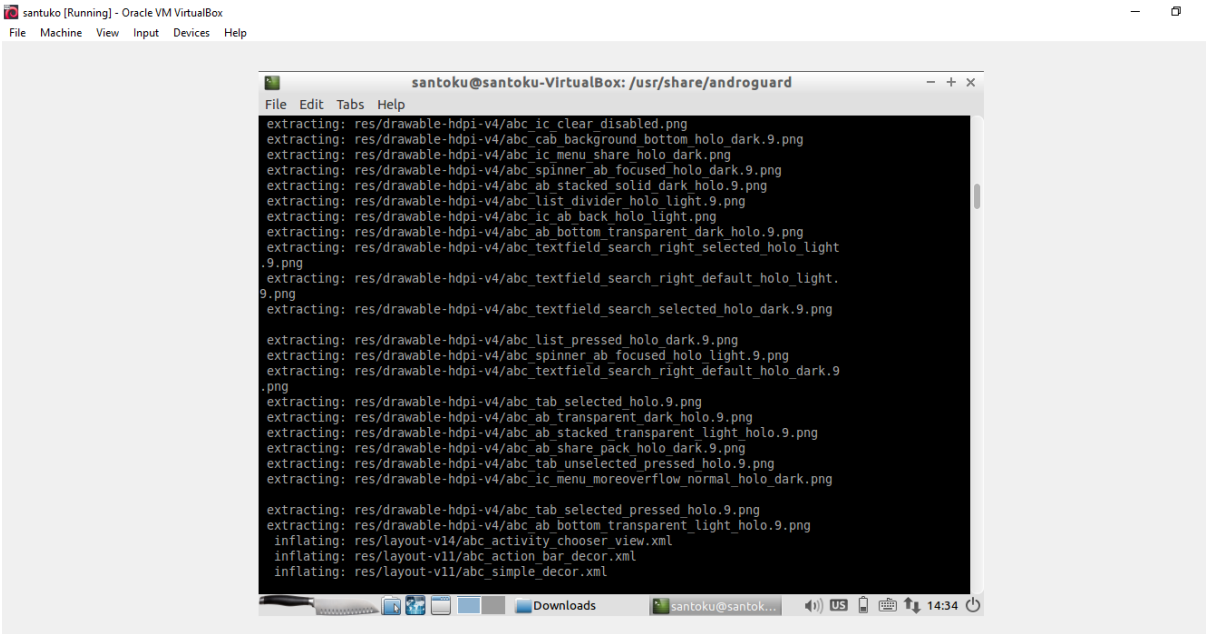
Malicious APK File Analysis
No. 13

Step-1: Unzipping the apk file.

After downloading the apk file, we need to unzip it in order to get the actual malicious file. The command to unzip the file is shown in the below image. The file name is Final.apk.

There is no password for the file.





Step-2: Analyzing the apk

Santoku has a large set of tools that can be used in the analysis of an apk file. One of such tools is Androguard.

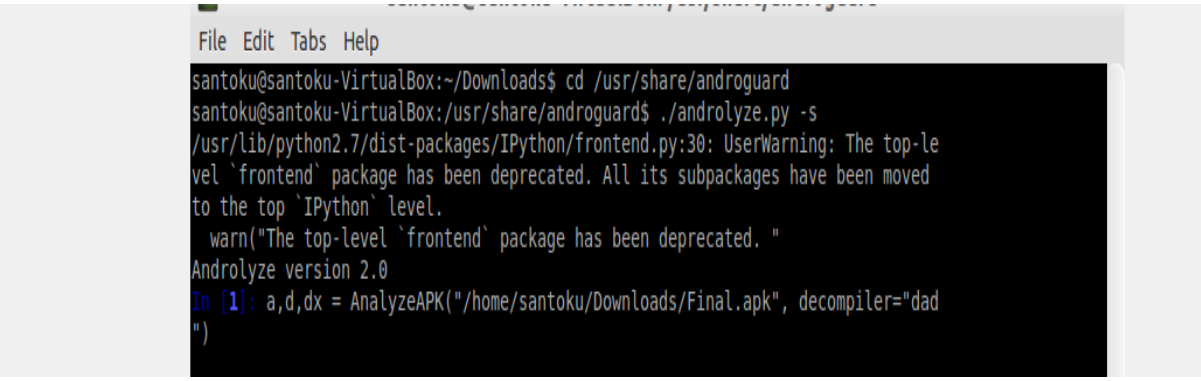
Androguard offers an interactive python shell to interact with the API through various commands.

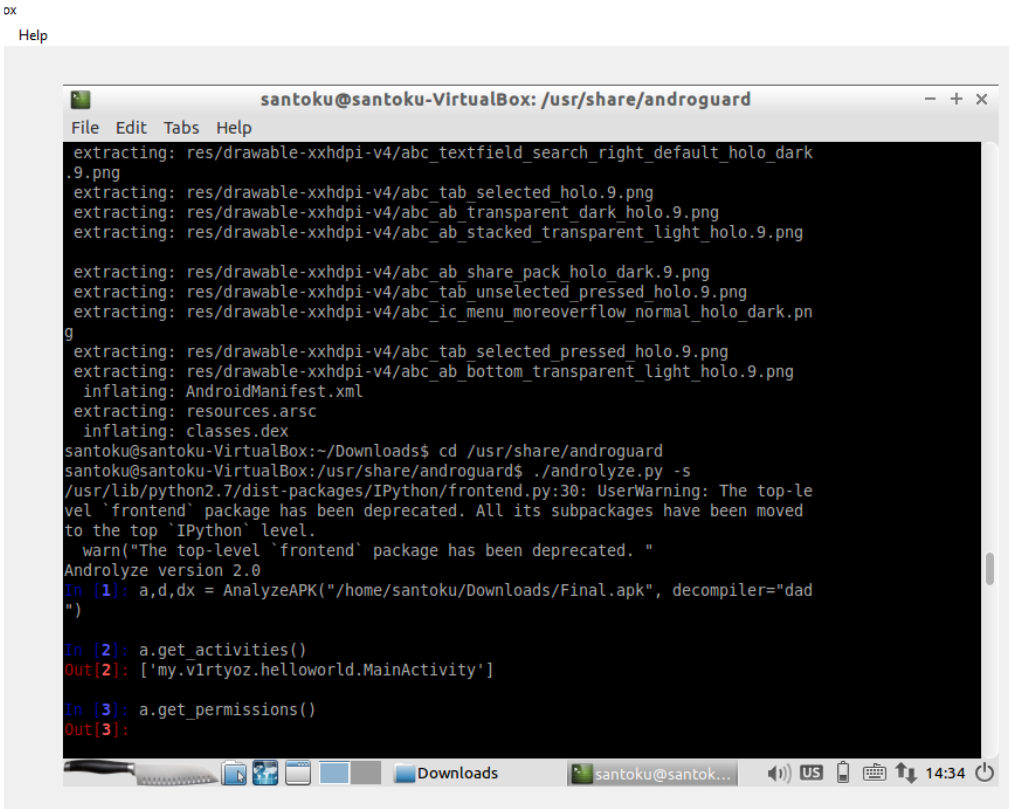
Using androlyze.py

(a) To run androlyze ipython shell, first navigate to androguard directory located in /usr/share directory as follows: $ cd /usr/share/Androguard

(b) Next, we will use the androguard's androlyze tool in the interactive python shell. To begin, type the following command in the terminal: $ ./androlyze.py -s

(c) Type the following command to decompile the apk using the default dad compiler:





(d) Let's start by getting the list of the APK's activities by typing a.activities(): It appears that the APK is using some sort of payment apis in addition to offering chat facilities.

(e) APK permissions can be examined as below:

```
santoku@santoku-VirtualBox: /usr/share/androguard
File  Edit  Tabs  Help
  'record audio',
  'Allows application to access the audio record path.'],
 'android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS': ['normal',
  'Unknown permission from android reference',
  'Unknown permission from android reference'],
 '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.SET_WALLPAPER': ['normal',
  'set wallpaper',
  'Allows the application to set the system wallpaper.'],
 '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 pho
ne. Malicious 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 applica
tions can corrupt your system's configuration."]}
```

```
  'full Internet access',
  'Allows an application to create network sockets.'],
 'android.permission.READ_CALL_LOG': ['dangerous',
  "read the user's call log.",
  "Allows an application to read the user's call log."],
 'android.permission.READ_CONTACTS': ['dangerous',
  'read contact data',
  'Allows an application to read all of the contact (address) data stored on you
r 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 applica
tion with this permission can determine the phone number and serial number of th
is phone, whether a call is active, the number that call is connected to and so
on.'],
 'android.permission.READ_SMS': ['dangerous',
  'read SMS or MMS',
  'Allows application to read SMS messages stored on your phone or SIM card. Mal
icious applications may read your confidential messages.'],
 'android.permission.RECEIVE_BOOT_COMPLETED': ['normal',
  'automatically start at boot',
  'Allows an application to start itself as soon as the system has finished boot
ing. This can make it take longer to start the phone and allow the application t
o slow down the overall phone by always running.'],
 'android.permission.RECEIVE_SMS': ['dangerous',
  'receive SMS',
  'Allows application to receive and process SMS messages. Malicious application
s may monitor your messages or delete them without showing them to you.'],
 'android.permission.RECORD_AUDIO': ['dangerous',
```

lp

santoku@santoku-VirtualBox: /usr/share/androguard

File Edit Tabs Help

```
 'android.permission.RECEIVE_SMS']

In [4]: a.get_details_permissions()
Out[4]:
{'android.permission.ACCESS_COARSE_LOCATION': ['dangerous',
  'coarse (network-based) location',
  'Access coarse location sources, such as the mobile network database, to deter
mine an approximate phone location, where available. Malicious applications can
use this to determine approximately where you are.'],
 'android.permission.ACCESS_FINE_LOCATION': ['dangerous',
  'fine (GPS) location',
  'Access fine location sources, such as the Global Positioning System on the ph
one, where available. Malicious applications can use this to determine where you
 are and may consume additional battery power.'],
 'android.permission.ACCESS_NETWORK_STATE': ['normal',
  'view network status',
  'Allows an application to view the status of all networks.'],
 'android.permission.ACCESS_WIFI_STATE': ['normal',
  'view Wi-Fi status',
  'Allows an application to view the information about the status of Wi-Fi.'],
 'android.permission.CALL_PHONE': ['dangerous',
  'directly call phone numbers',
  'Allows an application to initiate a phone call without going through the Dial
er user interface for the user to confirm the call being placed. '],
 'android.permission.CAMERA': ['dangerous',
  'take pictures and videos',
  'Allows application to take pictures and videos with the camera. This allows t
he application to collect images that the camera is seeing at any time.'],
 'android.permission.CHANGE_WIFI_STATE': ['dangerous',
  'change Wi-Fi status',
```

Downloads santoku@santok... 14:36

```
File  Edit  Tabs  Help
Out[2]: ['my.v1rtyoz.helloworld.MainActivity']

In [3]: a.get_permissions()
Out[3]:
['android.permission.ACCESS_NETWORK_STATE',
 'android.permission.INTERNET',
 'android.permission.ACCESS_WIFI_STATE',
 'android.permission.READ_CONTACTS',
 'android.permission.RECEIVE_BOOT_COMPLETED',
 'android.permission.ACCESS_FINE_LOCATION',
 'android.permission.SEND_SMS',
 'android.permission.WRITE_EXTERNAL_STORAGE',
 'android.permission.READ_SMS',
 'android.permission.WRITE_CONTACTS',
 'android.permission.READ_CALL_LOG',
 'android.permission.CHANGE_WIFI_STATE',
 'android.permission.WRITE_CALL_LOG',
 'android.permission.CALL_PHONE',
 'android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS',
 'android.permission.CAMERA',
 'android.permission.WRITE_SETTINGS',
 'android.permission.SET_WALLPAPER',
 'android.permission.WAKE_LOCK',
 'android.permission.ACCESS_COARSE_LOCATION',
 'android.permission.RECORD_AUDIO',
 'android.permission.READ_PHONE_STATE',
 'android.permission.RECEIVE_SMS']
```

(f) service is a general entry point for keeping an app running in the background. The APK's services can be obtained using the following command:

```
  '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 applica
tions can corrupt your system's configuration."]}

In [5]: a.get_services()
Out[5]: ['my.v1rtyoz.helloworld.cwgvc.Agzrj']
```

(g) We can also check what Android version the APK is compatible with:

```
In [7]: a.get_androidversion_code()
Out[7]: u'1'

In [8]: a.get_androidversion_name()
Out[8]: u'1.0'

In [9]: a.get_min_sdk_version()
Out[9]: u'8'

In [10]: a.get_max_sdk_version()

In [11]: a.get_target_sdk_version()
Out[11]: u'19'
```

(h) We can check where the app signature is located. This displays that the Signature/KEY file is located in the META-INF folder of the APK.

```
In [12]: a.get_signature_name()
Out[12]: u'META-INF/NITHIN.RSA'
```

Malware analysis using Androguard on kali linux:

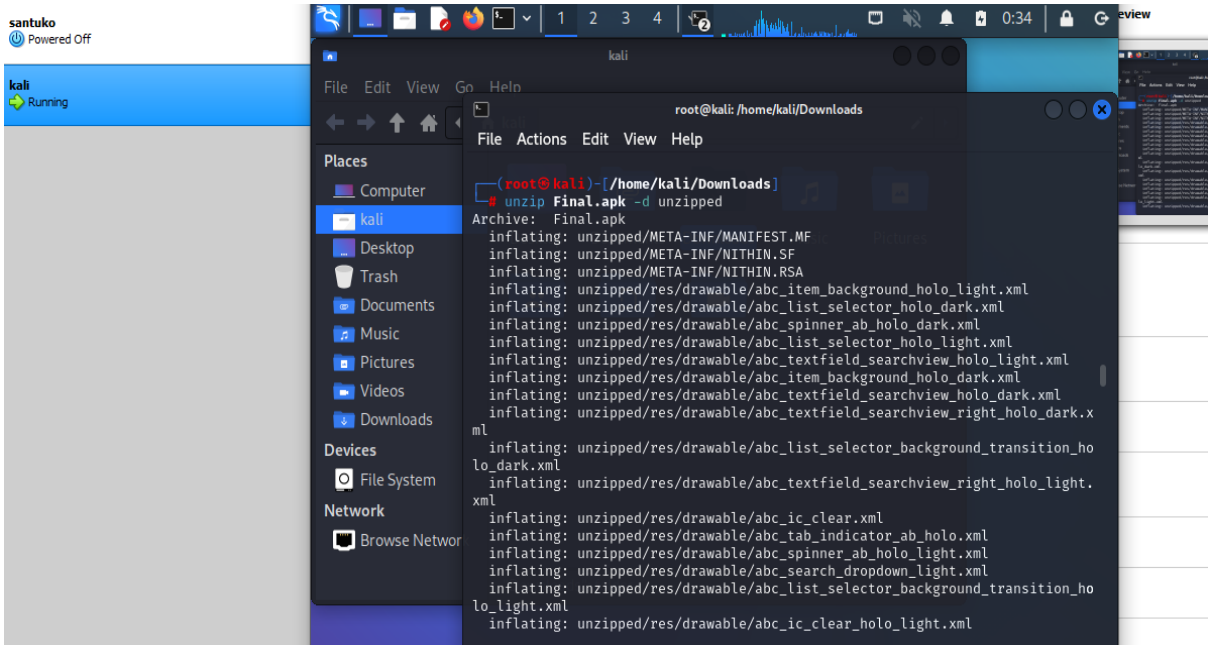
(a) To get the app name, we will use a.get_app_name() command as follows:

```
In [25]: a.get_app_name()
Out[25]: '少年西游决'

In [26]:
```

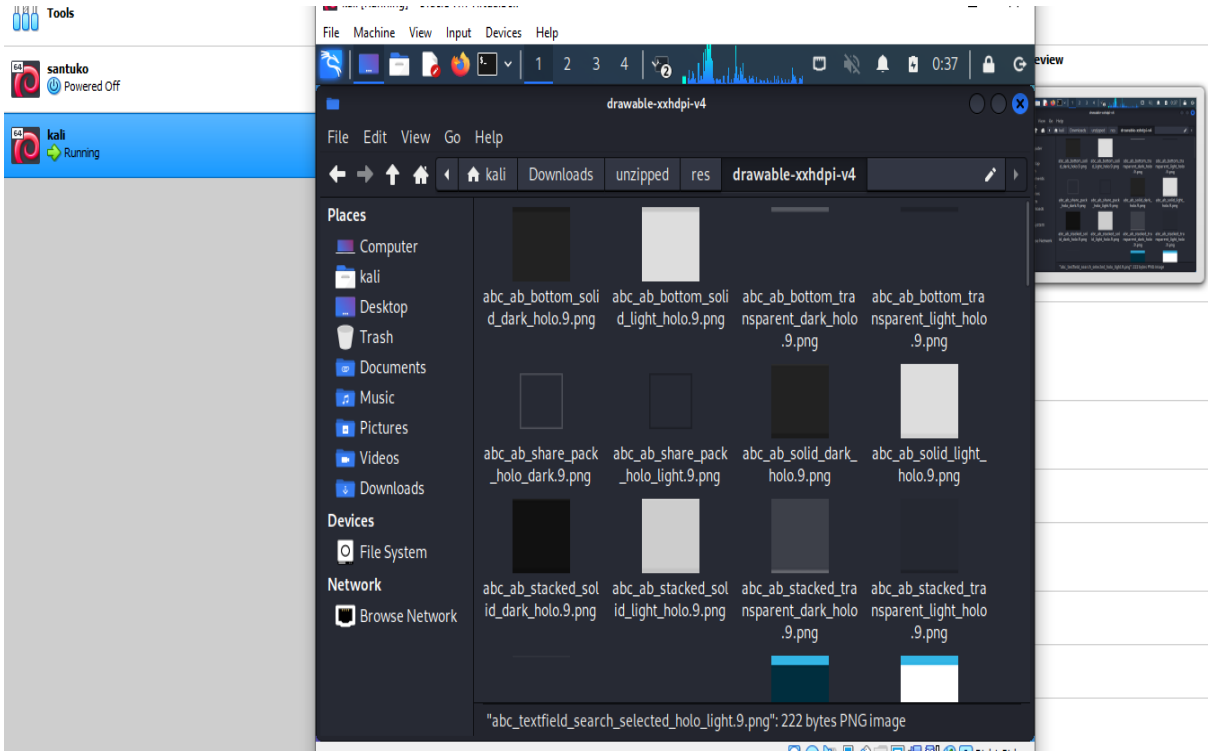
(b) Getting app logo:

```
In [25]: a.get_app_name()
Out[25]: '少年西游决'

In [26]: a.get_app_icon()
Out[26]: 'res/drawable-xxxhdpi/icon.png'

In [27]:
```

(c) unzip the apk to a folder called “Unzipped” as below



(d) The following is the unzipped directory. Let's navigate to "res/drawablexxxhdpi/icon.png" to see what the icon looks like:



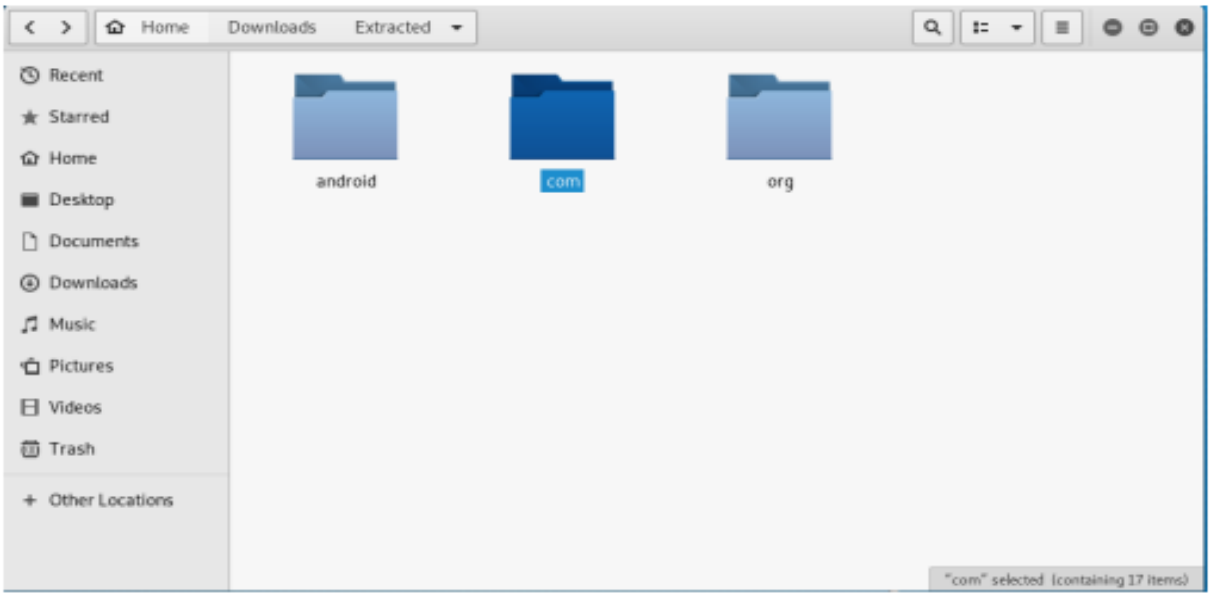
APK Decompilation using Androguard on Kali Linux:

Generating Control Flow Graphs

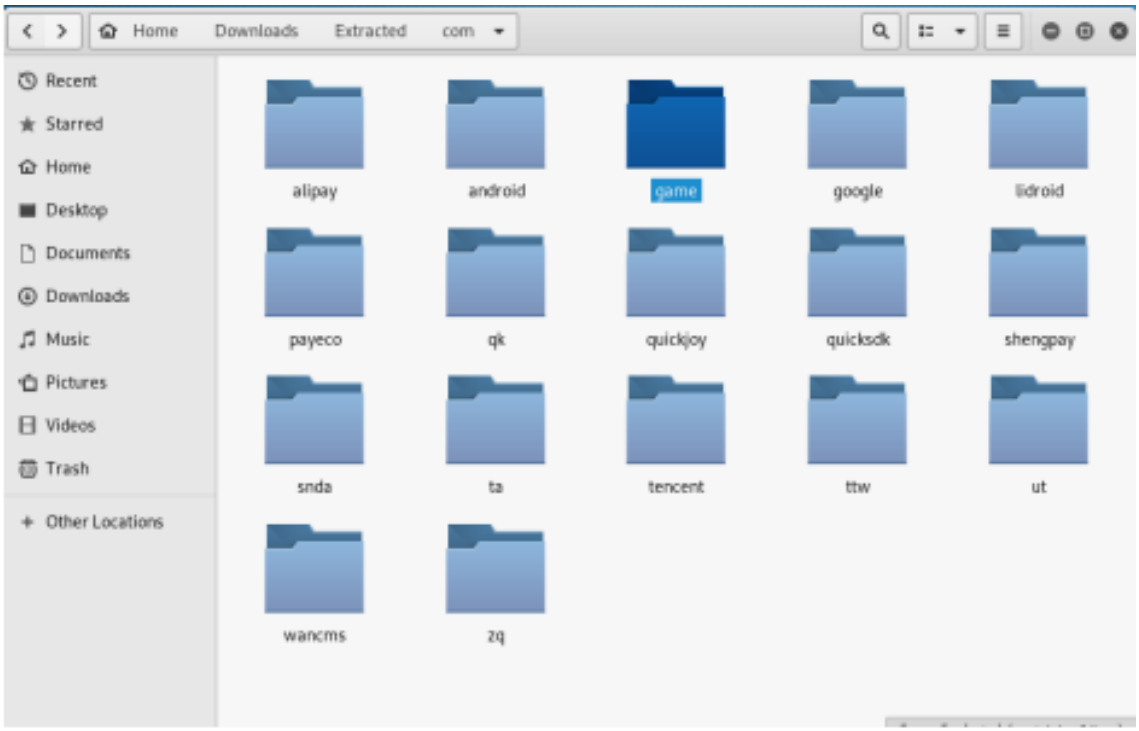
Androguard decompile command lets you extract and generate the control flow graphs for each class in the apk. We will use the png format.

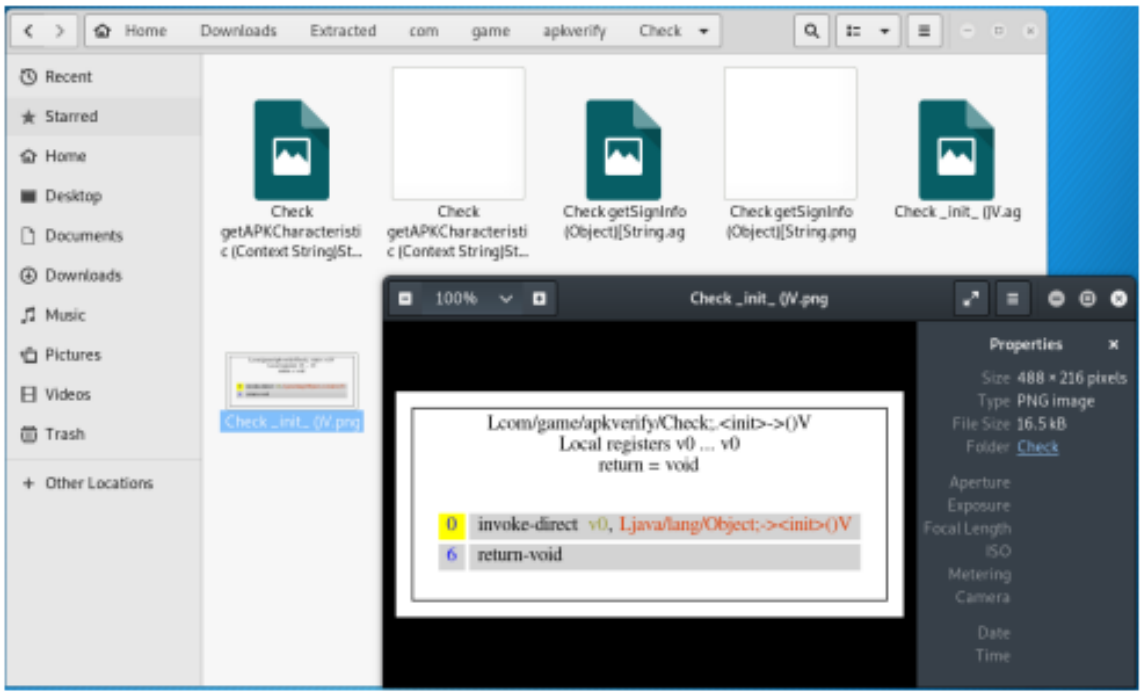
```
Usage: androguard decompile [OPTIONS] [FILE_]

  Decompile an APK and create Control Flow Graphs.

  Example:

      $ androguard resources.arsc

Options:
  -i, --input FILE       APK to parse (legacy option)
  -o, --output TEXT      output directory. If the output folder already
                         exsist, it will be overwritten!  [required]
  -f, --format TEXT      Additionally write control flow graphs for each
                         method, specify the format for example png, jpg, raw
                         (write dot file), ...
  -j, --jar              Use DEX2JAR to create a JAR file
  -l, --limit TEXT       Limit to certain methods only by regex (default:
                         '.*')
  -d, --decompiler TEXT  Use a different decompiler (default: DAD)
  --help                 Show this message and exit.
```

After decompilation, our Apk structure looks the following:



Exploring the game folder inside com, we can find a control flow graph as follows:





Generating Call Graph:

Androguard also has the ability to generate call graph using the cg command as follows:

```
Usage: androguard cg [OPTIONS] APK

  Create a call graph and export it into a graph format.

  The default is to create a file called callgraph.gml in the current
  directory!

  classnames are found in the type "Lfoo/bar/bla;".

  Example:

      $ androguard cg examples/tests/hello-world.apk

Options:
  -o, --output TEXT           Filename of the output file, the extension is
                              used to decide which format to use  [default:
                              callgraph.gml]
  -s, --show                  instead of saving the graph, print it with
                              mathplotlib (you might not see anything!)
  -v, --verbose               Print more output
  --classname TEXT            Regex to filter by classname  [default: .*]
  --methodname TEXT           Regex to filter by methodname  [default: .*]
  --descriptor TEXT           Regex to filter by descriptor  [default: .*]
  --accessflag TEXT           Regex to filter by accessflags  [default: .*]
  --no-isolated / --isolated  Do not store methods which has no xrefs
  --help                      Show this message and exit.
```

Secret code: No Secret code in this file