

Malicious APK File Analysis

No. 15

Stage 2

Your job is to investigate the content of a given malicious Android app. Using the tools given in the above and below link and the ones presented in the class to learn about analysis of apk file, your job is to analyze the given app and reveal the secret code along with the description of what the malicious part is trying to do:

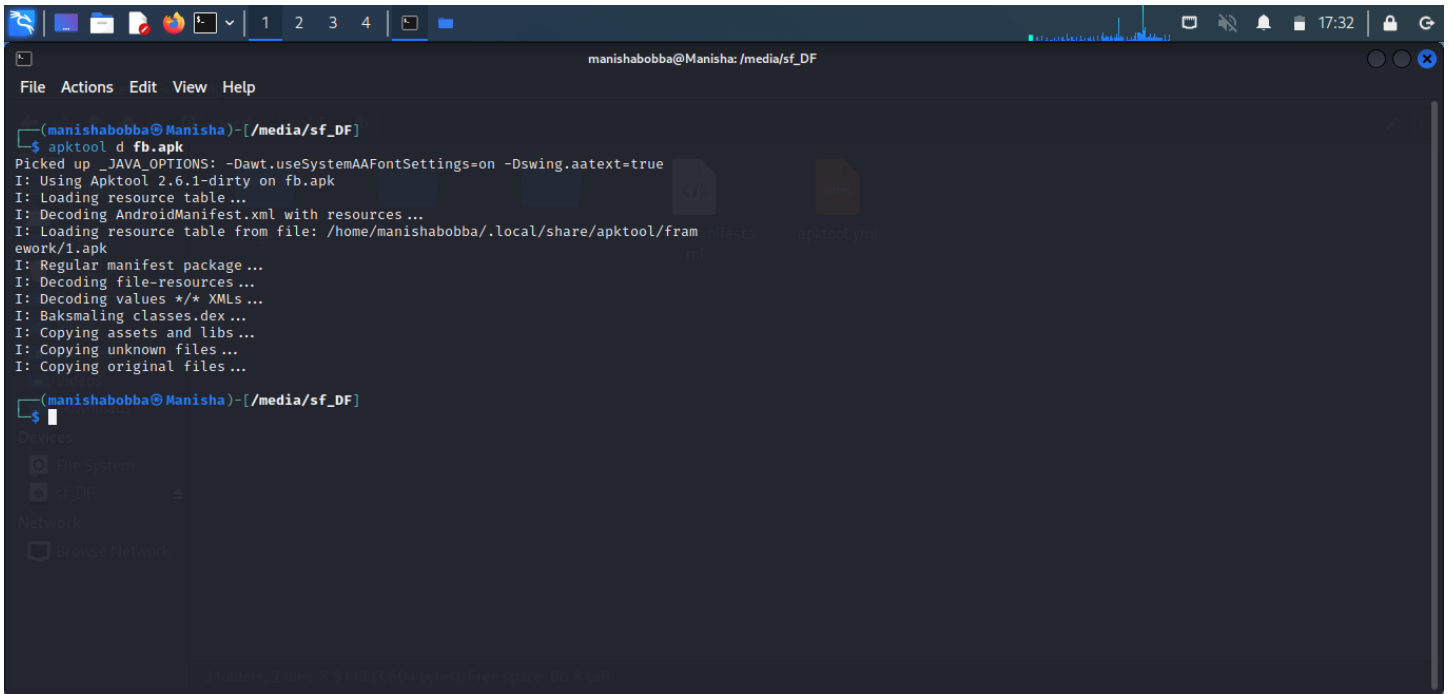
<https://github.com/asiamina/A-Course-on-DigitalForensics/tree/master/Hands%20On%20Experience/Android%20Forensics>

1. Analyze the given app and report what the malicious part is trying to achieve. Submit a word document along with some snapshots.
2. What is the secret code? Submit a report step-by-step explaining on how you have done the process and which tools are used and how.

Deliverable: A document showing your analysis of the android app file along with some snapshots and steps taken and revealing secret code in the end.

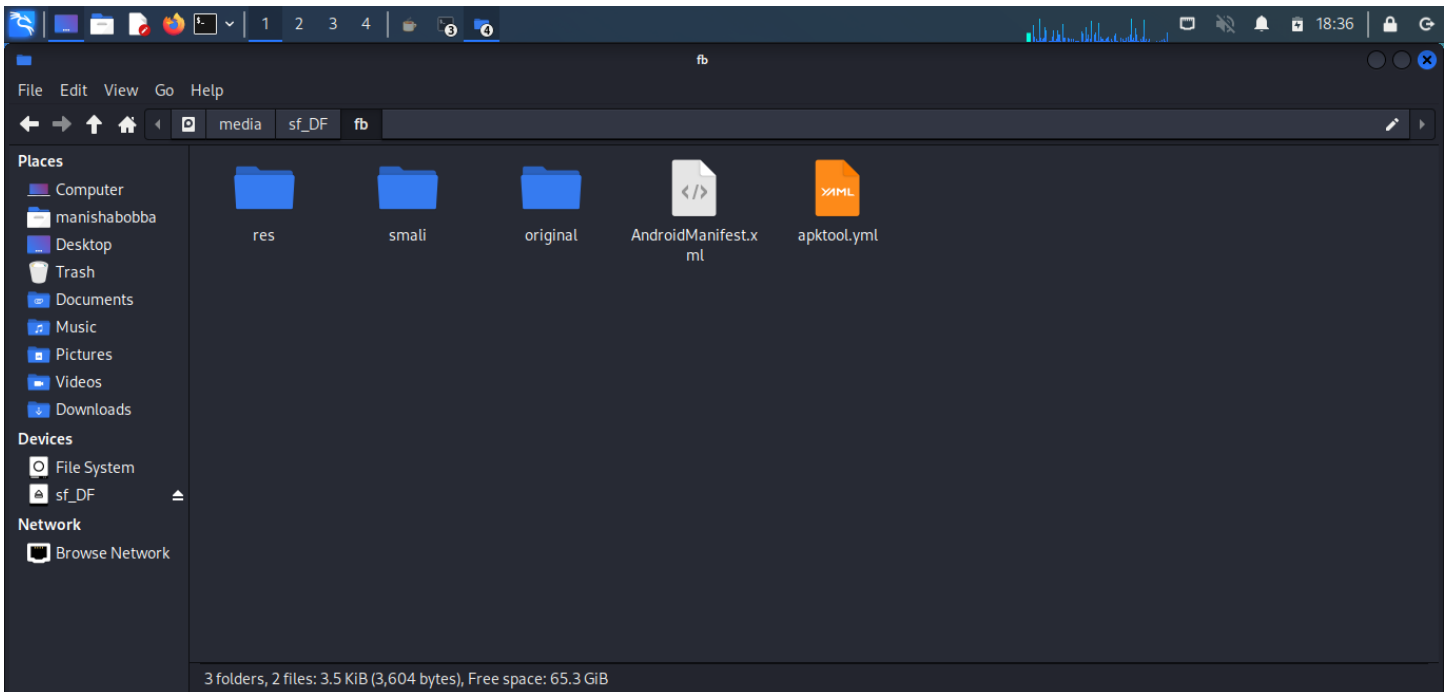
Team given to analyze – Team 15

We used APK tool to decompile the APK.

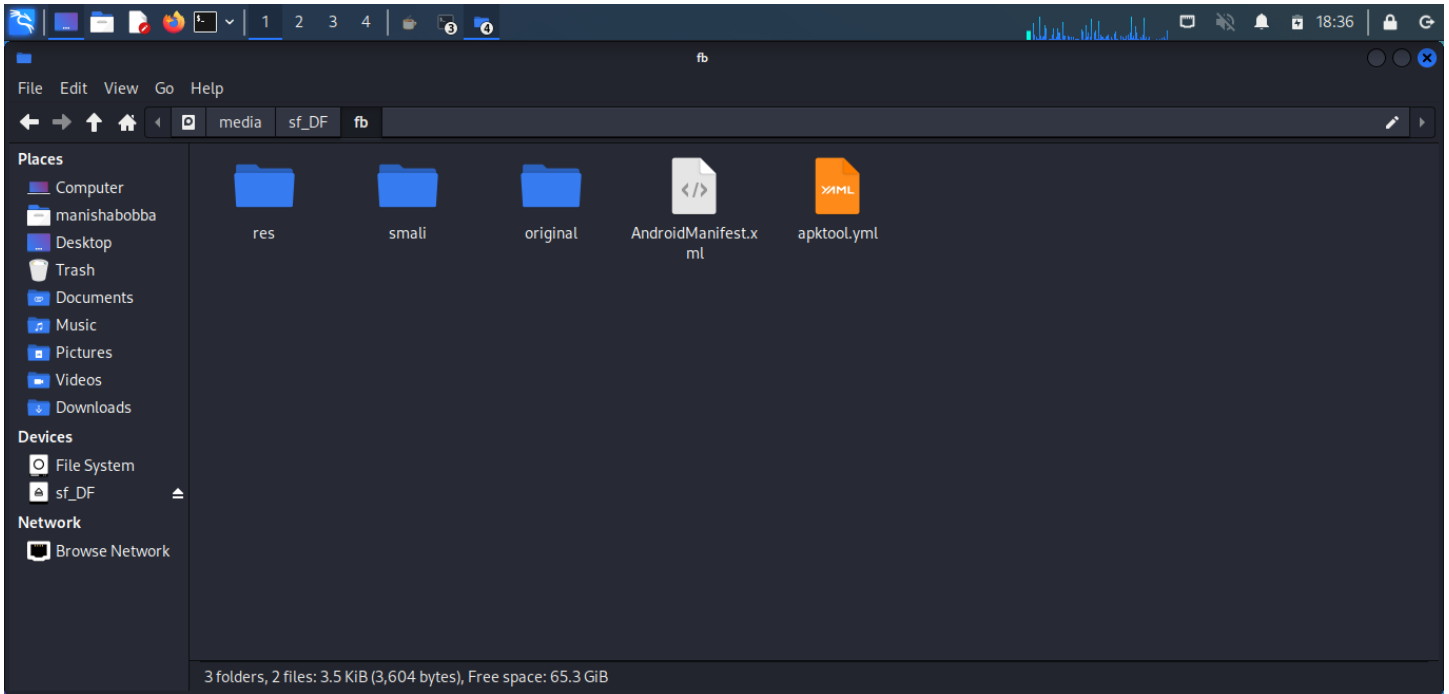


```
(manishabobba@Manisha)-[/media/sf_DF]
$ apktool d fb.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1-dirty on fb.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/manishabobba/.local/share/apktool/framework/1.apk
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...
```

After decompiling the APK file we can see the folders below.

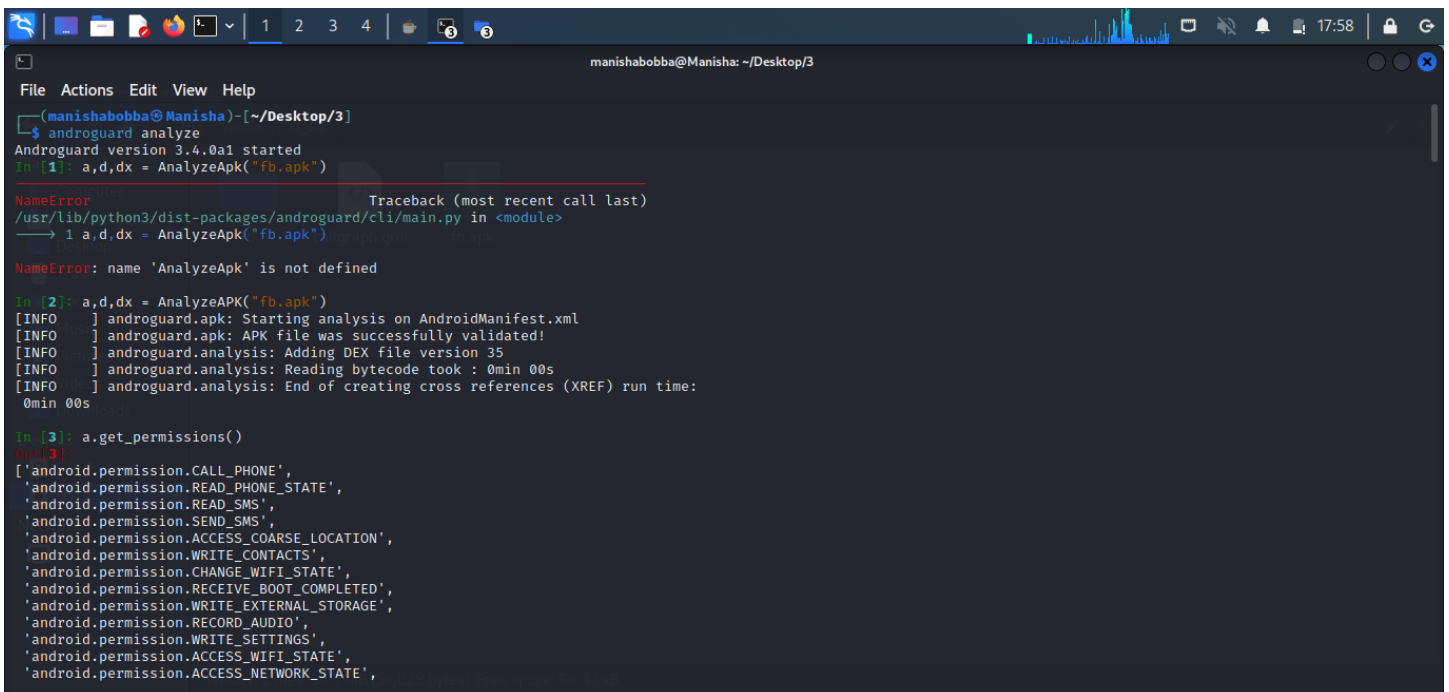


We searched the secret code everywhere in the file but couldn't find it. It looks like Team 15 has not embedded any secret code in the file.



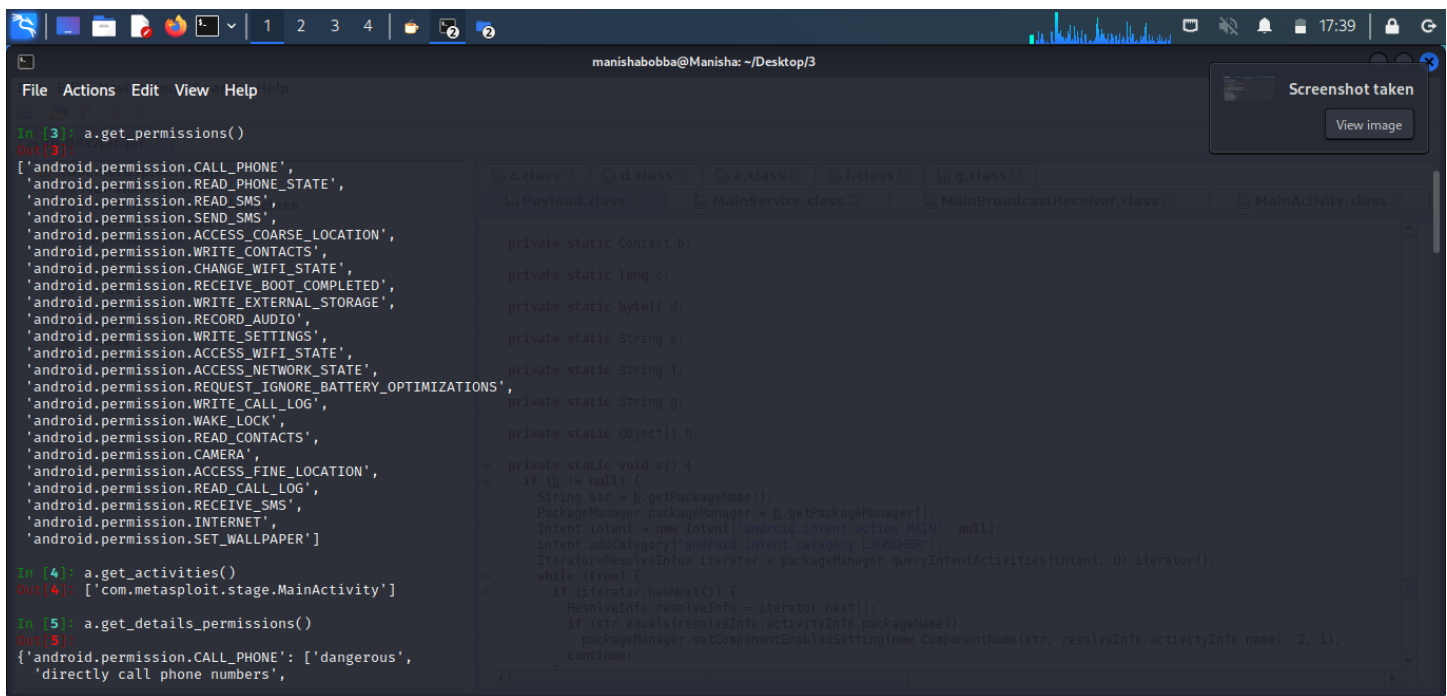
Analysis of the APK File

We used androguard to do the static analysis of the APK file.



These are the permissions that app needs to get, in order to use.

We used ***a.get_permissions()*** method to get all permissions that the APK uses.



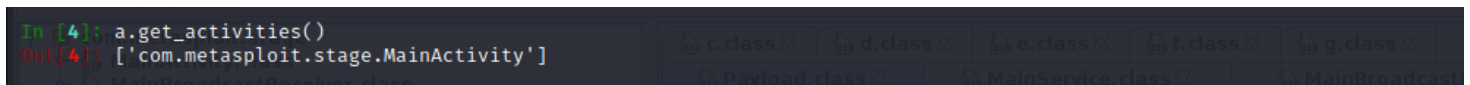
```
In [3]: a.get_permissions()
Out[3]:
['android.permission.CALL_PHONE',
'android.permission.READ_PHONE_STATE',
'android.permission.READ_SMS',
'android.permission.SEND_SMS',
'android.permission.ACCESS_COARSE_LOCATION',
'android.permission.WRITE_CONTACTS',
'android.permission.CHANGE_WIFI_STATE',
'android.permission.RECEIVE_BOOT_COMPLETED',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

In [4]: a.get_activities()
Out[4]: ['com.metasploit.stage.MainActivity']

In [5]: a.get_details_permissions()
Out[5]:
{'android.permission.CALL_PHONE': ['dangerous',
'directly call phone numbers',
```

These are the activities that are listed in the APK file.

get_activities() is the method used to list the activities of the APK file.



```
In [4]: a.get_activities()
Out[4]: ['com.metasploit.stage.MainActivity']
```

The screenshot shows a Kali Linux desktop environment. At the top, a Windows taskbar is visible with various application icons. The main window is a terminal titled 'manishabobba@Manisha: ~/Desktop/3'. The terminal output shows a Metasploit Meterpreter session where the user has loaded a module and is inspecting the permissions of a remote application. The output of 'a.get_details_permissions()' lists several permissions, including 'android.permission.CALL_PHONE', 'android.permission.READ_PHONE_STATE', 'android.permission.READ_SMS', 'android.permission.SEND_SMS', and 'android.permission.ACCESS_COARSE_LOCATION', each with a brief description of what they allow the app to do. A 'Screenshot taken' notification is visible in the top right corner of the terminal window.

```
File Actions Edit View Help

'android.permission.INTERNET',
'android.permission.SET_WALLPAPER']

In [4]: a.get_activities()
Out[4]: ['com.metasploit.stage.MainActivity']

In [5]: a.get_details_permissions()
Out[5]:
{'android.permission.CALL_PHONE': ['dangerous',
'directly call phone numbers',
'Allows the app to call phone numbers\n      without your intervention. This ma
y result in unexpected charges or calls.\n      Note that this doesn't allow the
app to call emergency numbers.\n      Malicious apps may cost you money by making
calls without your\n      confirmation.'],
'android.permission.READ_PHONE_STATE': ['dangerous',
'read phone status and identity',
'Allows the app to access the phone\n      features of the device. This permis
sion allows the app to determine the\n      phone number and device IDs, whether
a call is active, and the remote number\n      connected by a call.'],
'android.permission.READ_SMS': ['dangerous',
'read your text messages (SMS or MMS)',
'Allows the app to read SMS\n      messages stored on your phone or SIM card. T
his allows the app to read all\n      SMS messages, regardless of content or conf
identiality.'],
'android.permission.SEND_SMS': ['dangerous',
'send SMS messages',
'Allows the app to send SMS messages.\n      This may result in unexpected charg
es. Malicious apps may cost you money by\n      sending messages without your conf
irmation.'],
'android.permission.ACCESS_COARSE_LOCATION': ['dangerous',
'approximate location\n      (network-based)',
'Allows the app to get your\n      approximate location. This location is deriv
ed by location services using\n      network location sources such as cell towers
and Wi-Fi. These location\n      services must be turned on and available to you']}
```

The screenshot shows the Android Studio IDE with a Java file named `MainService.java` open. The code defines a service that manages Wi-Fi connections and network status. It includes permissions for accessing network state, ignoring battery optimizations, requesting ignore battery optimizations, reading call logs, waking the device, reading contacts, and accessing fine location. The code also includes methods for starting and stopping the service, and for querying the network status.

```
'view Wi-Fi connections',
'Allows the app to view information\n      about Wi-Fi networking, such as whet
her Wi-Fi is enabled and name of\n      connected Wi-Fi devices.'],
android.permission.ACCESS_NETWORK_STATE': ['normal',
'view network connections',
'Allows the app to view\n      information about network connections such as wh
ich networks exist and are\n      connected.'],
'android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS': ['normal',
'Unknown permission from android reference',
'Unknown permission from android reference'],
'android.permission.WRITE_CALL_LOG': ['dangerous',
'write call log',
'Allows the app to modify your phone's call log, including data about incoming
and outgoing calls.\n      Malicious apps may use this to erase or modify your
call log.'],
'android.permission.WAKE_LOCK': ['normal',
'prevent phone from sleeping',
'Allows the app to prevent the phone from going to sleep.'],
'android.permission.READ_CONTACTS': ['dangerous',
'read your contacts',
'Allows the app to\n      read data about your contacts stored on your phone, i
ncluding the\n      frequency with which you've called, emailed, or communicated
in other ways\n      with specific individuals. This permission allows apps to sa
ve your\n      contact data, and malicious apps may share contact data without yo
ur\n      knowledge.'],
'android.permission.CAMERA': ['dangerous',
'take pictures and videos',
'Allows the app to take pictures and videos\n      with the camera. This permi
ssion allows the app to use the camera at any\n      time without your confirmati
on.'],
'android.permission.ACCESS_FINE_LOCATION': ['dangerous',
'precise location (GPS and\n      network-based)',
'Allows the app to get your\n      precise location using the Global Positionin
g System (GPS) or network\n      location sources such as cell towers and Wi-Fi.
```

```
'approximate location\n    (network-based)',\n'Allows the app to get your\n    approximate location. This location is deriv\n    ed by location services using\n    network location sources such as cell towers\n    and Wi-Fi. These location\n    services must be turned on and available to you\n    r device for the app to\n    use them. Apps may use this to determine approxima\n    tely where you\n    are.'],\n'android.permission.WRITE_CONTACTS': ['dangerous',\n    'modify your contacts',\n    'Allows the app to\n    modify the data about your contacts stored on your phon\n    e, including the\n    frequency with which you've called, emailed, or communicate\n    d in other ways\n    with specific contacts. This permission allows apps to delet\n    e contact\n    data.'],\n'android.permission.CHANGE_WIFI_STATE': ['dangerous',\n    'connect and disconnect from Wi-Fi',\n    'Allows the app to connect to and\n    disconnect from Wi-Fi access points an\n    d to make changes to device\n    configuration for Wi-Fi networks.'],\n'android.permission.RECEIVE_BOOT_COMPLETED': ['normal',\n    'run at startup',\n    'Allows the app to\n    have itself started as soon as the system has finis\n    hed booting.\n    This can make it take longer to start the phone and allow t\n    he\n    app to slow down the overall phone by always running.'],\n'android.permission.WRITE_EXTERNAL_STORAGE': ['dangerous',\n    'modify or delete the contents of your SD card',\n    'Allows the app to write to the SD card.'],\n'android.permission.RECORD_AUDIO': ['dangerous', 'record audio'],\n'android.permission.WRITE_SETTINGS': ['normal',\n    'modify system settings',\n    'Allows the app to modify the\n    system's settings data. Malicious apps m\n    ay corrupt your system's\n    configuration.'],\n'android.permission.ACCESS_WIFI_STATE': ['normal',\n    'view Wi-Fi connections',\n    'Allows the app to view information\n    about Wi-Fi networking, such as whet\n    her Wi-Fi is enabled and name of\n    connected Wi-Fi devices.'],\n'android.permission.ACCESS_NETWORK_STATE': ['normal',\n    'view network connections',\n    'Allows the app to view information\n    about network connections, such as whet\n    her the device is connected to the internet.'],\n'android.permission.READ_CALL_LOG': ['dangerous',\n    'read call log',\n    'Allows the app to read\n    your phone's call log, including data about inco\n    ming and outgoing calls.\n    This permission allows apps to save your call log\n    data, and malicious apps\n    may share call log data without your knowledge.'],\n'android.permission.RECEIVE_SMS': ['dangerous',\n    'receive text messages (SMS)',\n    'Allows the app to receive and process SMS\n    messages. This means the app\n    could monitor or delete messages sent to your\n    device without showing them\n    to you.'],\n'android.permission.INTERNET': ['dangerous',\n    'full network access',\n    'Allows the app to create\n    network sockets and use custom network protocol\n    s. The browser and other\n    applications provide means to send data to the int\n    ernet, so this\n    permission is not required to send data to the internet.'],\n'android.permission.SET_WALLPAPER': ['normal',\n    'set wallpaper',\n    'Allows the app to set the system wallpaper.'],\n\nprivate static byte[] d;\nprivate static String a;\nprivate static String g;\nprivate static void a() {\n    if (b != null) {\n        String str = b.getPackageName();\n        PackageManager packageManager = b.getPackageManager();\n        Intent intent = new Intent("android.intent.action.MAIN", null);\n        intent.addCategory("android.intent.category.LAUNCHER");\n        Iterator<ResolveInfo> iterator = packageManager.queryIntentActivities(intent, 0).iterator();\n        while (true) {\n            if (iterator.hasNext()) {\n                ResolveInfo resolveInfo = iterator.next();\n                if (str.equals(resolveInfo.activityInfo.packageName)) {\n                    packageManager.setComponentEnabledSetting(new ComponentName(str, resolveInfo.activityInfo.name), 2, 1);\n                    continue;\n                }\n            }\n        }\n    }\n}
```

```
'Allows the app to take pictures and videos\n    with the camera. This permi\n    ssion allows the app to use the camera at any\n    time without your confirmati\n    on.'],\n'android.permission.ACCESS_FINE_LOCATION': ['dangerous',\n    'precise location (GPS and\n    network-based)',\n    'Allows the app to get your\n    precise location using the Global Positionin\n    g System (GPS) or network\n    location sources such as cell towers and Wi-Fi.\n    These location services\n    must be turned on and available to your device for\n    the app to use them.\n    Apps may use this to determine where you are, and ma\n    y consume additional\n    battery power.'],\n'android.permission.READ_CALL_LOG': ['dangerous',\n    'read call log',\n    'Allows the app to read\n    your phone's call log, including data about inco\n    ming and outgoing calls.\n    This permission allows apps to save your call log\n    data, and malicious apps\n    may share call log data without your knowledge.'],\n'android.permission.RECEIVE_SMS': ['dangerous',\n    'receive text messages (SMS)',\n    'Allows the app to receive and process SMS\n    messages. This means the app\n    could monitor or delete messages sent to your\n    device without showing them\n    to you.'],\n'android.permission.INTERNET': ['dangerous',\n    'full network access',\n    'Allows the app to create\n    network sockets and use custom network protocol\n    s. The browser and other\n    applications provide means to send data to the int\n    ernet, so this\n    permission is not required to send data to the internet.'],\n'android.permission.SET_WALLPAPER': ['normal',\n    'set wallpaper',\n    'Allows the app to set the system wallpaper.'],\n\nprivate static long c;\nprivate static byte[] d;\nprivate static String a;\nprivate static String g;\nprivate static void a() {\n    if (b != null) {\n        String str = b.getPackageName();\n        PackageManager packageManager = b.getPackageManager();\n        Intent intent = new Intent("android.intent.action.MAIN", null);\n        intent.addCategory("android.intent.category.LAUNCHER");\n        Iterator<ResolveInfo> iterator = packageManager.queryIntentActivities(intent, 0).iterator();\n        while (true) {\n            if (iterator.hasNext()) {\n                ResolveInfo resolveInfo = iterator.next();\n                if (str.equals(resolveInfo.activityInfo.packageName)) {\n                    packageManager.setComponentEnabledSetting(new ComponentName(str, resolveInfo.activityInfo.name), 2, 1);\n                    continue;\n                }\n            }\n        }\n    }\n}
```

The below listed activities are coming under **dangerous** part of the code.

1. Directly call phone numbers
2. Read phone status and identity
3. Read your text messages (SMS or MMS)
4. Send SMS messages
5. Approximate location
6. Allows app to modify your call log
7. Read your contacts
8. Take pictures and videos
9. Allows the app to get your precise location or GPRS

10. Modify your contacts
 11. Connect and disconnect from Wi-Fi
 12. Modify or delete the contents of your SD card
 13. Receive text messages
 14. Full network access
-
-

The below listed activities comes under the **normal** part of the code.

1. View network connections
2. Prevent phone from sleeping
3. Run at startup
4. View Wi-Fi connections
5. Modify system settings
6. Allows the app to take pictures and videos

get_package() is used to get the names of the packages in the APK.

```
In [7]: a.get_package()  
Out[7]: 'com.metasploit.stage'
```

get_androidversion_code() is used to get the version code of the APK file that is included in build.gradle or AndroidManifest.xml

```
In [8]: a.get_androidversion_code()  
Out[8]: '1'
```

get_androidversion_name() is used to get the version name of the APK file that is included in build.gradle or AndroidManifest.xml

```
In [9]: a.get_androidversion_name()  
Out[9]: '1.0'
```

get_min_sdk_version() is used to get the minimum Android SDK version that this package expects to be runnable as specified in the manifest.

```
In [10]: a.get_min_sdk_version()  
Out[10]: '10'
```

get_max_sdk_version() is used to get the maximum Android SDK version that this package expects to be runnable as specified in the manifest.

```
In [11]: a.get_max_sdk_version()
```

get_target_sdk_version() is used to get the API Level on which the application is designed to run.

The android system will prevent the user from installing the application if the system's API Level is lower than the values specified in this attribute.

```
In [12]: a.get_target_sdk_version()  
Out[12]: '17'
```

get_files() is used to get the classes.

```
In [13]: a.get_files()  
Out[13]:  
['AndroidManifest.xml',  
 'resources.arsc',  
 'classes.dex',  
 'META-INF/',  
 'META-INF/MANIFEST.MF',  
 'META-INF/SIGNFILE.SF',  
 'META-INF/SIGNFILE.RSA']
```

get_app_icon() is used to get the name of the icon. Here we got nothing as there is no icon present in the APK file.

```
File Actions Edit View Help  
In [14]: a.get_app_icon()
```

get_app_name() is used to get the name of the application.

```
In [15]: a.get_app_name()  
Out[15]: 'MainActivity'
```


get_declared_permissions() is used to get the declared permissions. We got nothing as there are no declared permissions.

Return type – list of strings

```
In [16]: a.get_declared_permissions()  
Out[16]: []
```

get_declared_permissions_details() is used to get the list of declared permissions.

Return type – dictionary

```
In [17]: a.get_declared_permissions_details()  
Out[17]: {}
```

get_dex_names() returns the name of all dex files found in the APK. This method only accounts for “official” dex files in the root directory of apk named classes.dex

Return type – list of strings

```
In [19]: a.get_dex_names()  
Out[19]: <filter at 0x7f1269eebc40>
```

get_receivers() returns the android : name attributes of all receivers.

Return type - list of strings

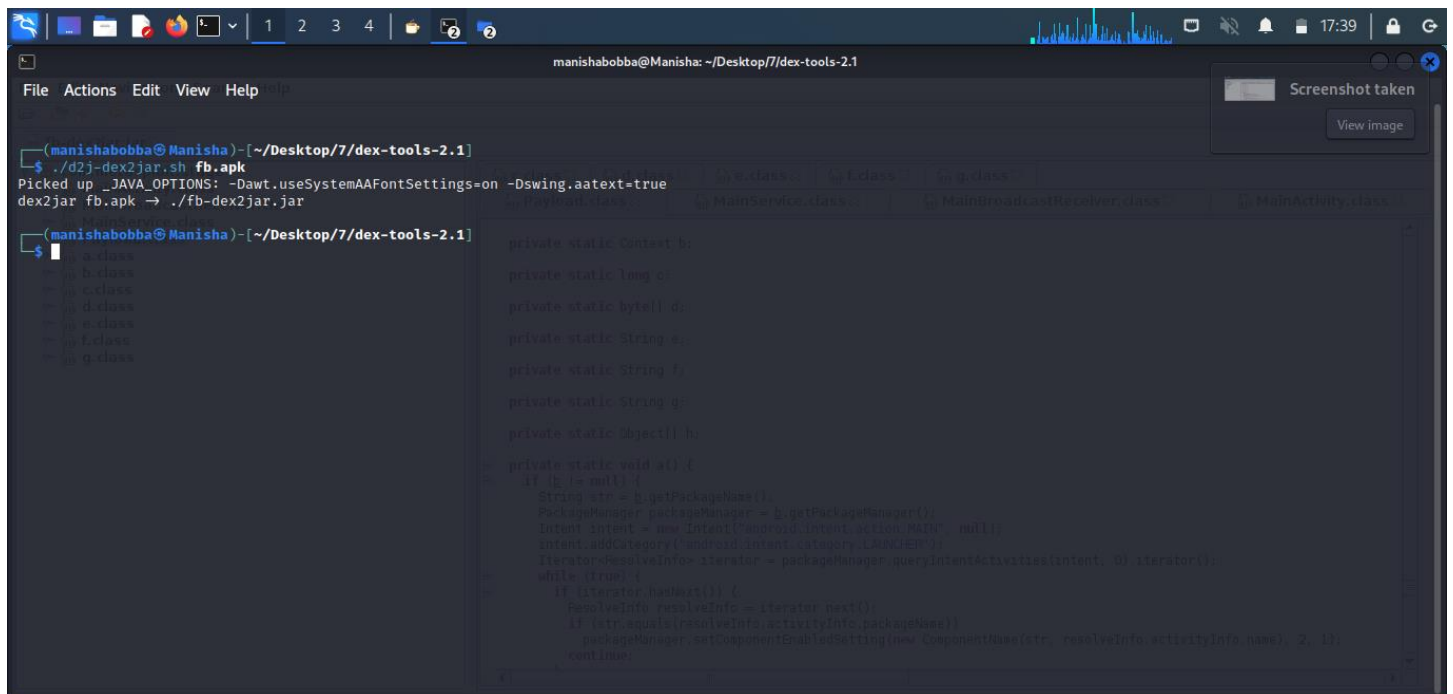
```
In [20]: a.get_receivers()  
Out[20]: ['com.metasploit.stage.MainBroadcastReceiver']
```

get_services() returns the android : name attributes of all services.

Return type - list of strings

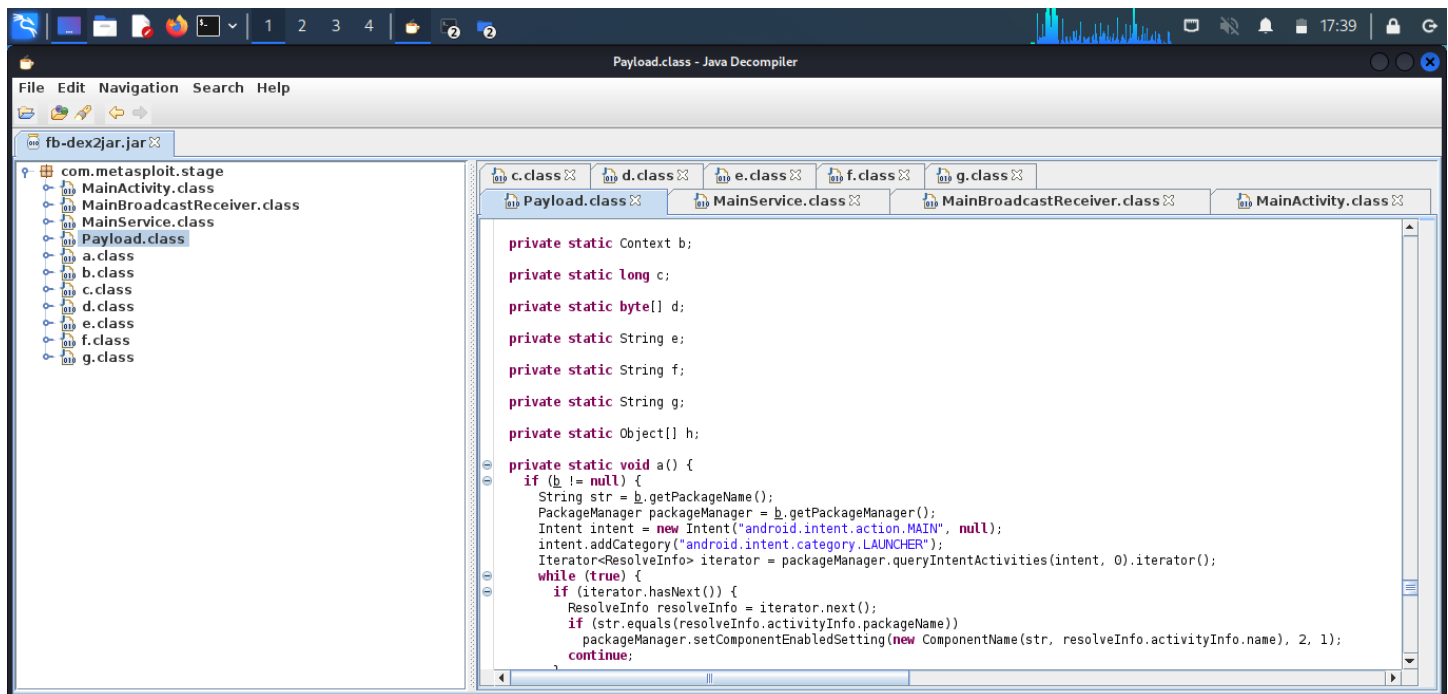
```
In [21]: a.get_services()  
Out[21]: ['com.metasploit.stage.MainService']
```

Return type – list of bytes



We see that the APK is converted to jar file.

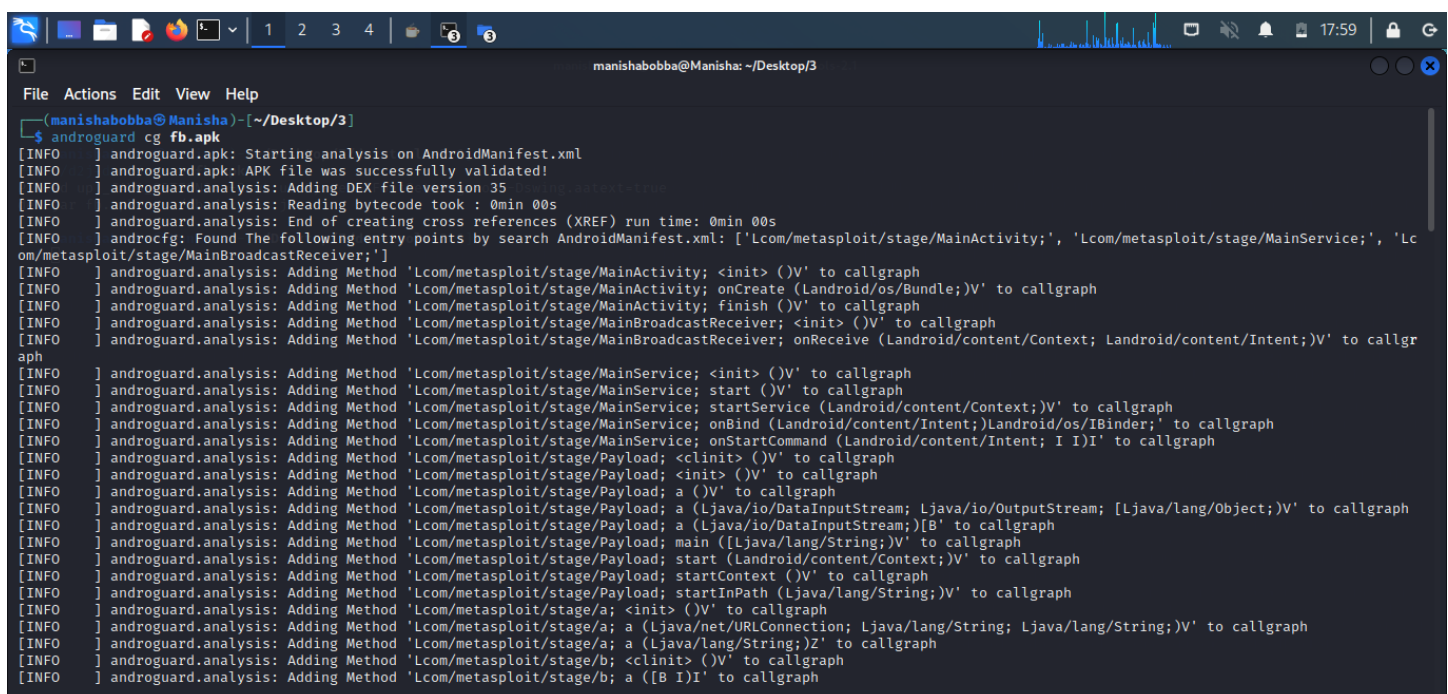
When we open the converted jar file using JD-GUI we can see all the classes and packages of the APK file as below.



The command androguard cg fb.apk is used to create call graph from APK. The

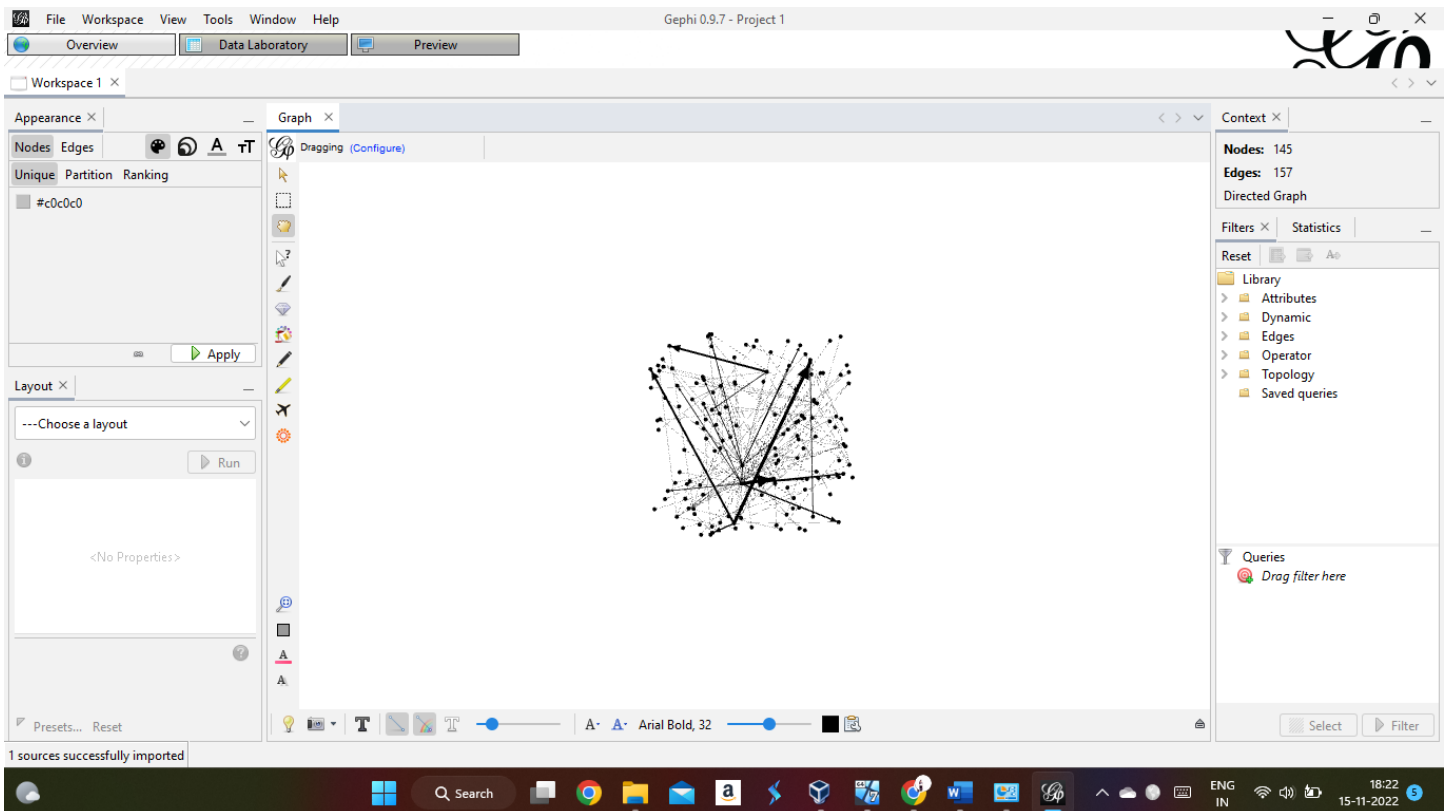
call graph is constructed from the analysis object and then converted into a networkx MultiDiGraph.

We generated callgraph.gml file which can be opened using Gephi tool.



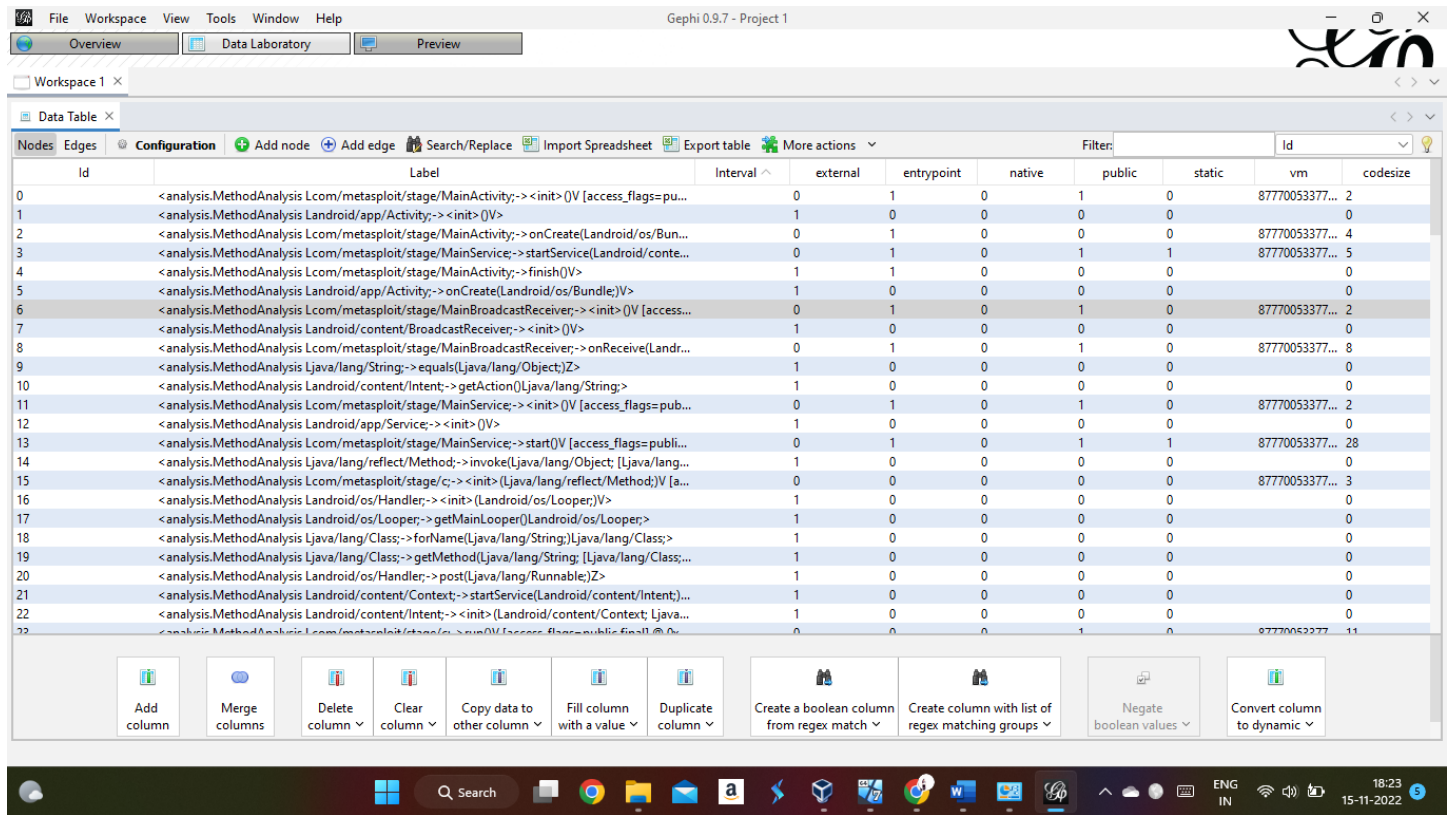
```
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ServerSocket; <init> (I)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ServerSocket; accept ()Ljava/net/Socket;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ServerSocket; close ()V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/Socket; getInputStream ()Ljava/io/InputStream;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/Socket; getOutputStream ()Ljava/io/OutputStream;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/Socket; <init> (Ljava/lang/String; I)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/io/DataOutputStream; <init> (Ljava/io/OutputStream;)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/lang/Thread; sleep (J)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/lang/Thread; <init> ()V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/URL; <init> (Ljava/lang/String;)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/URLConnection; openConnection ()Ljava/net/URLConnection;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/URLConnection; getInputStream ()Ljava/io/InputStream;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/URLConnection; addRequestProperty (Ljava/lang/String; Ljava/lang/String;)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/io/ByteArrayOutputStream; <init> ()V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/util/LinkedList; <init> ()V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/util/concurrent/TimeUnit; toMillis (J)J' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ssl/SSLContext; getInstance (Ljava/lang/String;)Ljava/net/ssl/SSLContext;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ssl/SSLContext; init ([Ljava/net/ssl/KeyManager; [Ljava/net/ssl/TrustManager; Ljava/security/SecureRandom;
)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ssl/SSLContext; getSocketFactory ()Ljava/net/ssl/SSLSocketFactory;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/SecureRandom; <init> ()V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ssl/HttpsURLConnection; setSSLSocketFactory (Ljava/net/ssl/SSLSocketFactory;)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/net/ssl/HttpsURLConnection; setHostnameVerifier (Ljava/net/ssl/HostnameVerifier;)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/cert/CertificateException; <init> ()V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/cert/CertificateException; <init> (Ljava/lang/String;)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/cert/CertificateException; <init> (Ljava/lang/Throwable;)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/MessageDigest; getInstance (Ljava/lang/String;)Ljava/security/MessageDigest;' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/MessageDigest; update ([B)V' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/MessageDigest; digest ()[B' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/security/cert/X509Certificate; getEncoded ()[B' to callgraph
[INFO] ] androguard.analysis: Adding Method 'Ljava/util/Arrays; equals ([B [B)Z' to callgraph
```

We can see the graph as below in Gephi from the file generated from the above steps.



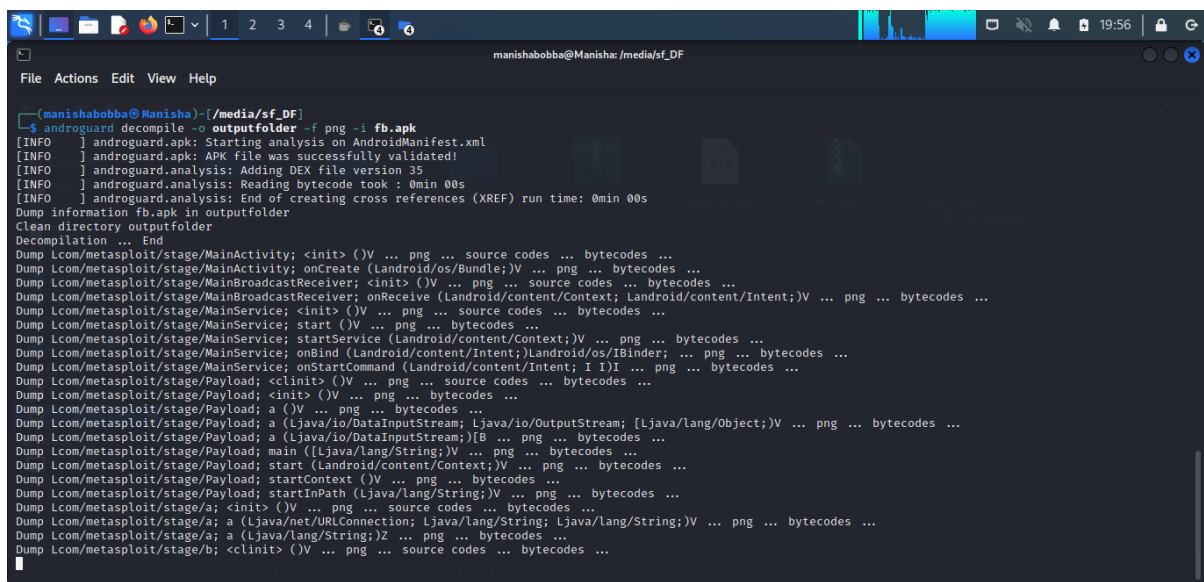
These are all the nodes and edges that show the attributes of every node or edge as table columns of the APK file in Gephi tool.

Here we are showing the methods, label and node ID's.



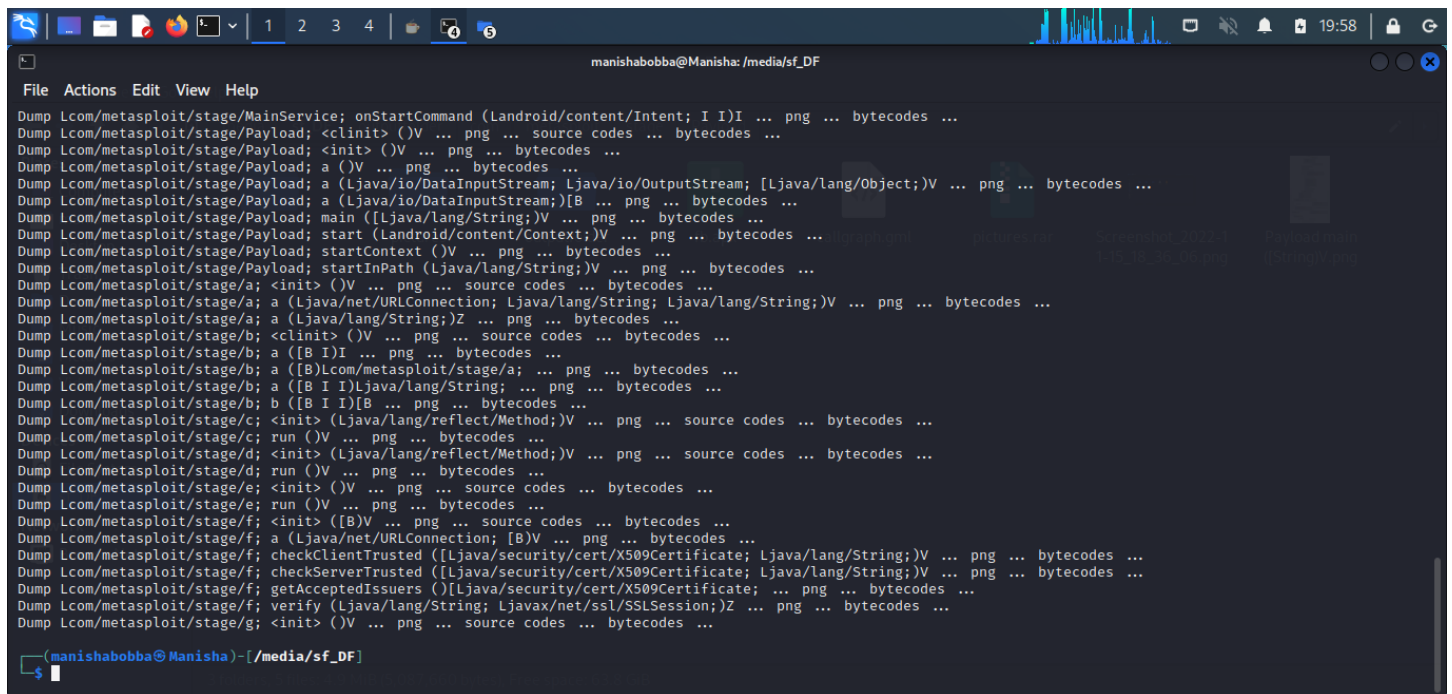
Id	Label	Interval	external	endpoint	native	public	static	vm	codesize
0	<analysis.MethodAnalysis Lcom/metasplit/stage/MainActivity;-> <init> ()V [access_flags=pu...	0	1	0	0	1	0	87770053377...	2
1	<analysis.MethodAnalysis Landroid/app/Activity;-> <init> ()V	1	0	0	0	0	0		0
2	<analysis.MethodAnalysis Lcom/metasplit/stage/MainActivity;-> onCreate(Landroid/os/Bun...	0	1	0	0	0	0	87770053377...	4
3	<analysis.MethodAnalysis Lcom/metasplit/stage/MainService;-> startService(Landroid/conte...	0	1	0	0	1	1	87770053377...	5
4	<analysis.MethodAnalysis Lcom/metasplit/stage/MainActivity;-> finish()V	1	1	0	0	0	0		0
5	<analysis.MethodAnalysis Landroid/app/Activity;-> onCreate(Landroid/os/Bundle;)V	1	0	0	0	0	0		0
6	<analysis.MethodAnalysis Lcom/metasplit/stage/MainBroadcastReceiver;-> <init> ()V [access...	0	1	0	0	1	0	87770053377...	2
7	<analysis.MethodAnalysis Landroid/content/BroadcastReceiver;-> <init> ()V	1	0	0	0	0	0		0
8	<analysis.MethodAnalysis Lcom/metasplit/stage/MainBroadcastReceiver;-> onReceive(Landr...	0	1	0	0	1	0	87770053377...	8
9	<analysis.MethodAnalysis Ljava/lang/String;-> equals(Ljava/lang/Object;)Z	1	0	0	0	0	0		0
10	<analysis.MethodAnalysis Landroid/content/Intent;-> getAction()Ljava/lang/String;	1	0	0	0	0	0		0
11	<analysis.MethodAnalysis Lcom/metasplit/stage/MainService;-> <init> ()V [access_flags=pub...	0	1	0	0	1	0	87770053377...	2
12	<analysis.MethodAnalysis Landroid/app/Service;-> <init> ()V	1	0	0	0	0	0		0
13	<analysis.MethodAnalysis Lcom/metasplit/stage/MainService;-> start()V [access_flags=publi...	0	1	0	0	1	1	87770053377...	28
14	<analysis.MethodAnalysis Ljava/lang/reflect/Method;-> invoke(Ljava/lang/Object; [Ljava/lang...	1	0	0	0	0	0		0
15	<analysis.MethodAnalysis Lcom/metasplit/stage/c;-> <init> (Ljava/lang/reflect/Method;)V [a...	0	0	0	0	0	0	87770053377...	3
16	<analysis.MethodAnalysis Landroid/os/Handler;-> <init> (Landroid/os/Looper;)V	1	0	0	0	0	0		0
17	<analysis.MethodAnalysis Landroid/os/Looper;-> getMainLooper()Landroid/os/Looper;	1	0	0	0	0	0		0
18	<analysis.MethodAnalysis Ljava/lang/Class;-> forName(Ljava/lang/String;)Ljava/lang/Class;	1	0	0	0	0	0		0
19	<analysis.MethodAnalysis Ljava/lang/Class;-> getMethod(Ljava/lang/String; [Ljava/lang/Class...	1	0	0	0	0	0		0
20	<analysis.MethodAnalysis Landroid/os/Handler;-> post(Ljava/lang/Runnable;)Z	1	0	0	0	0	0		0
21	<analysis.MethodAnalysis Landroid/content/Context;-> startService(Landroid/content/Intent)...	1	0	0	0	0	0		0
22	<analysis.MethodAnalysis Landroid/content/Intent;-> <init> (Landroid/content/Context; Ljava...	1	0	0	0	0	0		0
23	<analysis.MethodAnalysis Lcom/metasplit/stage/c;-> <init> ()V [access_flags=public,final]	0	0	0	0	1	0	87770053377...	11

The below command is used to generate the png image of the Control Flow Graphs of the classes of the APK file



```
(manishabobba@Manisha)~/media/sf_DF
$ androguard decompile -o outputfolder -f fb.apk
[INFO ] androguard.apk: Starting analysis on AndroidManifest.xml
[INFO ] androguard.apk: APK file was successfully validated!
[INFO ] androguard.analysis: Adding DEX file version 35
[INFO ] androguard.analysis: Reading bytecode took : 0min 00s
[INFO ] androguard.analysis: End of creating cross references (XREF) run time: 0min 00s
Dump information fb.apk in outputfolder
Clean directory outputfolder
Decompilation ... End
Dump Lcom/metasplit/stage/MainActivity; <init> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasplit/stage/MainActivity; onCreate(Landroid/os/Bundle;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/MainBroadcastReceiver; <init> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasplit/stage/MainBroadcastReceiver; onReceive(Landroid/content/Context; Landroid/content/Intent;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/MainService; <init> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasplit/stage/MainService; start()V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/MainService; startService(Landroid/content/Context;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/MainService; onBind(Landroid/content/Intent; Landroid/os/IBinder; ... png ... bytecodes ...
Dump Lcom/metasplit/stage/MainService; onStartCommand(Landroid/content/Intent; I I) ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; <init> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; <init> ()V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; a()V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; a(Ljava/io/DataInputStream; Ljava/io/OutputStream; [Ljava/lang/Object;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; a(Ljava/io/DataInputStream;)B ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; main([Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; start(Landroid/content/Context;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; startContext()V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/Payload; startInPath(Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/a; <init> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasplit/stage/a; a(Ljava/net/URLConnection; Ljava/lang/String; Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasplit/stage/a; a(Ljava/lang/String;)Z ... png ... bytecodes ...
Dump Lcom/metasplit/stage/b; <init> ()V ... png ... source codes ... bytecodes ...
```

We see that the generation of Control Flow Graph is success.



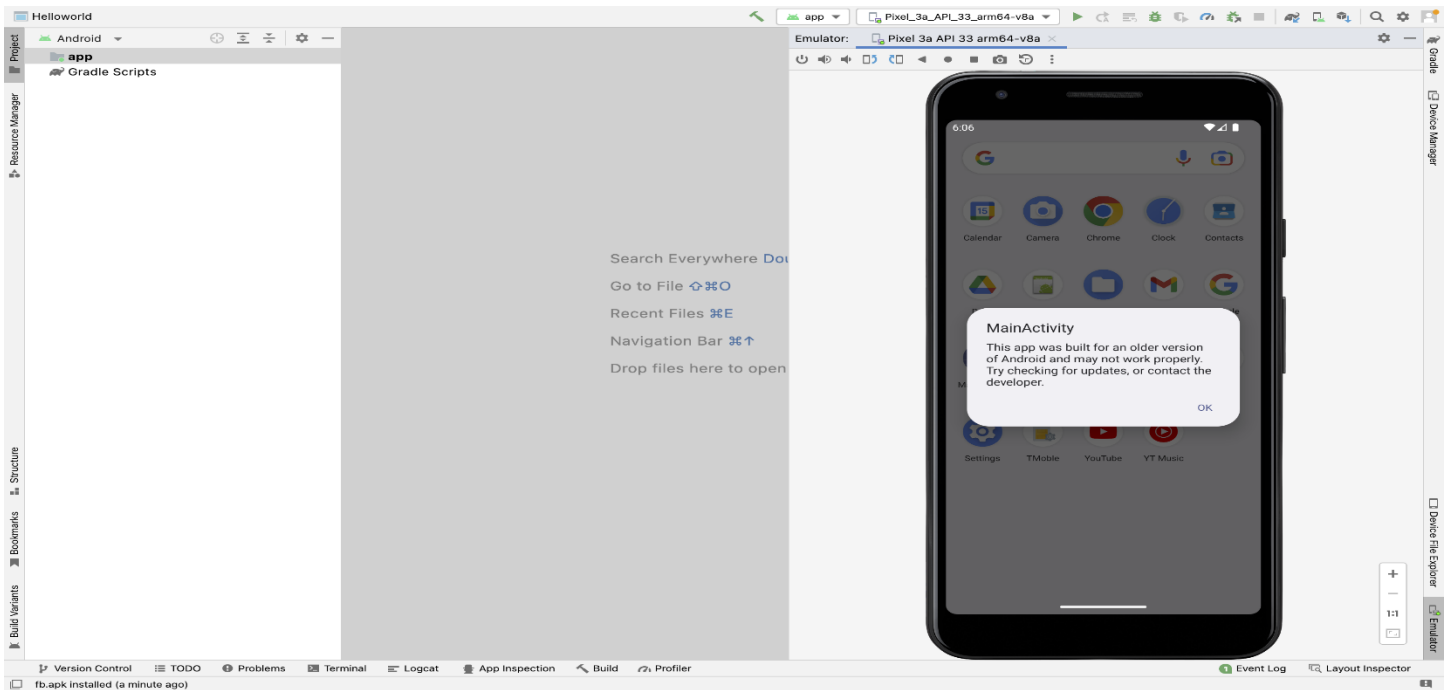
```
manishabobba@Manisha: /media/sf_DF
File Actions Edit View Help
Dump Lcom/metasploit/stage/MainService; onStartCommand (Landroid/content/Intent; I I)I ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; <clinit> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; <init> ()V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; a ()V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; a (Ljava/io/DataInputStream; Ljava/io/OutputStream; [Ljava/lang/Object;)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; a (Ljava/io/DataInputStream;) [B ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; main ([Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; start (Landroid/content/Context;)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; startContext ()V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/Payload; startInPath (Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/a; <init> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasploit/stage/a; a (Ljava/net/URLConnection; Ljava/lang/String; Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/a; a (Ljava/lang/String;)Z ... png ... bytecodes ...
Dump Lcom/metasploit/stage/b; <clinit> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasploit/stage/b; a ([B I)I ... png ... bytecodes ...
Dump Lcom/metasploit/stage/b; a ([B Lcom/metasploit/stage/a; ... png ... bytecodes ...
Dump Lcom/metasploit/stage/b; a ([B I I)Ljava/lang/String; ... png ... bytecodes ...
Dump Lcom/metasploit/stage/b; b ([B I I)[B ... png ... bytecodes ...
Dump Lcom/metasploit/stage/c; <init> (Ljava/lang/reflect/Method;)V ... png ... source codes ... bytecodes ...
Dump Lcom/metasploit/stage/c; run ()V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/d; <init> (Ljava/lang/reflect/Method;)V ... png ... source codes ... bytecodes ...
Dump Lcom/metasploit/stage/d; run ()V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/e; <init> ()V ... png ... source codes ... bytecodes ...
Dump Lcom/metasploit/stage/e; run ()V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/f; <init> ([B)V ... png ... source codes ... bytecodes ...
Dump Lcom/metasploit/stage/f; a (Ljava/net/URLConnection; [B)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/f; checkClientTrusted ([Ljava/security/cert/X509Certificate; Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/f; checkServerTrusted ([Ljava/security/cert/X509Certificate; Ljava/lang/String;)V ... png ... bytecodes ...
Dump Lcom/metasploit/stage/f; getAcceptedIssuers () [Ljava/security/cert/X509Certificate; ... png ... bytecodes ...
Dump Lcom/metasploit/stage/f; verify (Ljava/lang/String; Ljavax/net/ssl/SSLSession;)Z ... png ... bytecodes ...
Dump Lcom/metasploit/stage/g; <init> ()V ... png ... source codes ... bytecodes ...

(manishabobba@Manisha)~/media/sf_DF
```

[illegible]

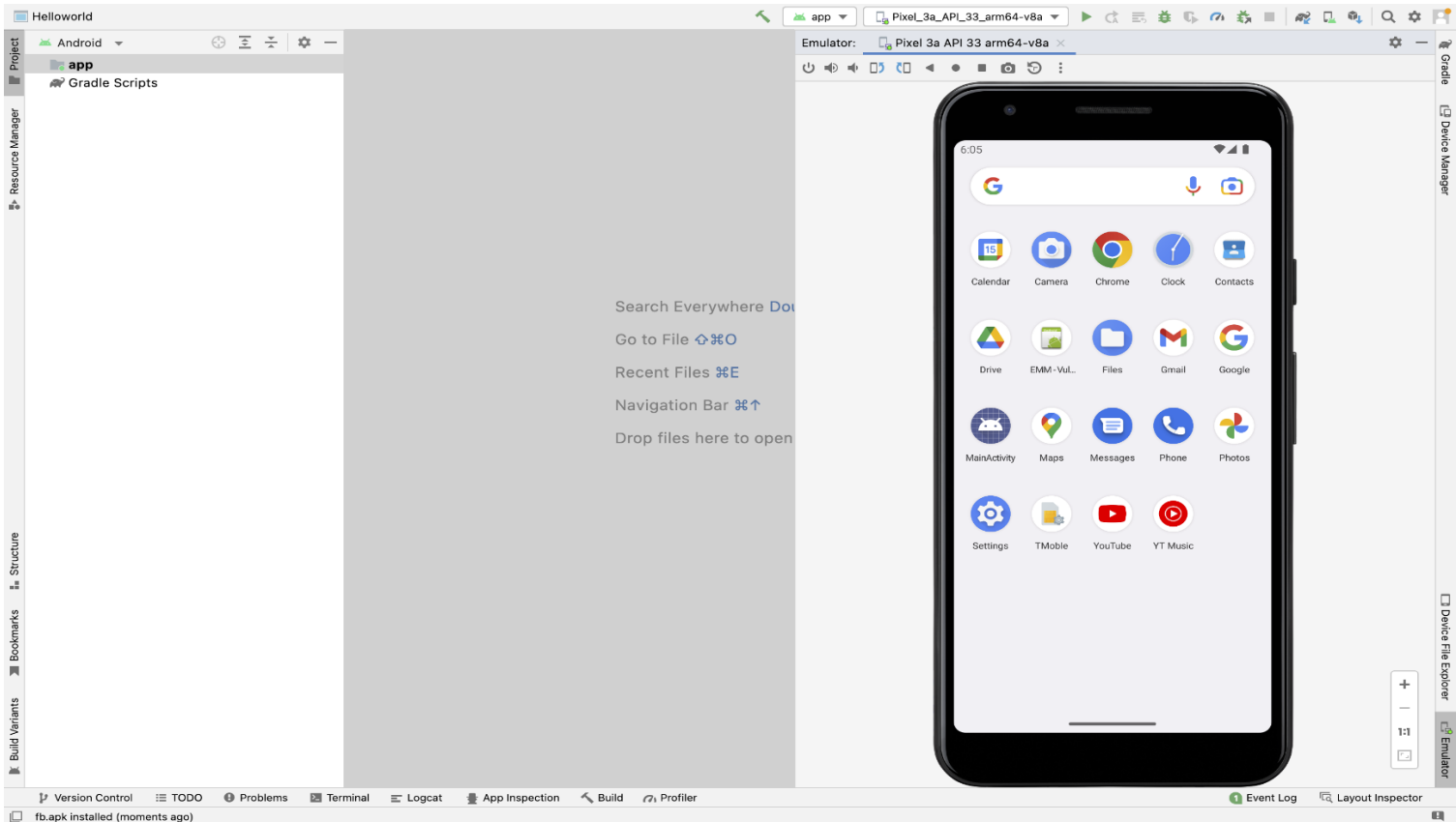
ANALYSIS USING ANDROID STUDIO:

We have tried to install the Malicious APK to the virtual device we observed it faced problems installing in the latest version of the android.



We have downgraded the Android version to 9 then we have successfully installed the application to the virtual device.

The installed APK file name is shown as Mainactivity.



How ever when we tried to open the application it asked for the permissions that the application can access when we grant the permission and click continue to proceed it should open the application how ever here here the application is not moving forward from the permissions page.

So we assume that the application is not installed.

