# Android – Teabot Banker Malware

# Threat Report

Date: 03-06-2021

K. Narahari

**Overview:**

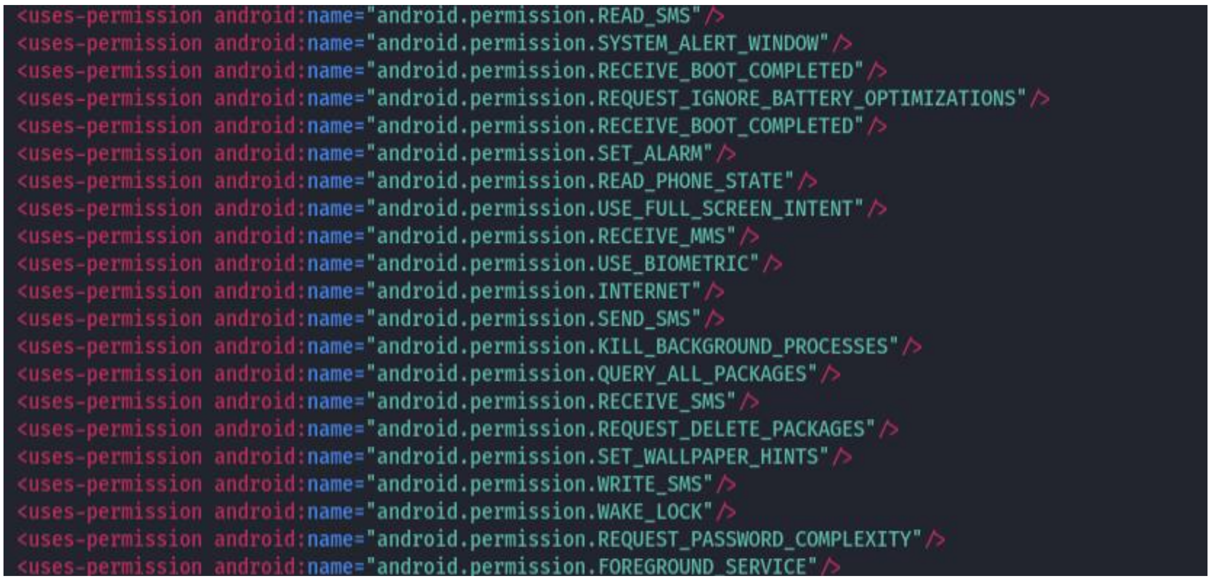
Attackers are now targeting popular Android applications to inject malicious malware and releasing it on the third-party platforms and torrents to steal data from the victim devices for their profit.

In our honeypot, we found a malicious Teabot Banking Trojan which was embedded on the Android application of "DHL", a popular international courier service. Teabot is the latest malware that aims to steal the login and credit card details, steals SMS and it can get complete control of the victim device from the attacker server and perform malicious application aiming majorly at banking applications and services.

**File Hash:** **bd0ee78cded55ff30a37c670cfa66236**

**Technical Analysis:**

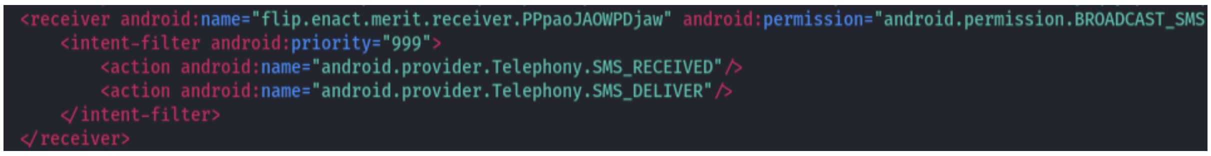
We have downloaded the malware sample and reverse-engineered the apk, then performed static analysis and went through the manifest.xml file which is the primary file to begin the analysis.



```
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<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.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.SET_ALARM"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.USE_FULL_SCREEN_INTENT"/>
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
<uses-permission android:name="android.permission.USE_BIOMETRIC"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>
<uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES"/>
<uses-permission android:name="android.permission.SET_WALLPAPER_HINTS"/>
<uses-permission android:name="android.permission.WRITE_SMS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.REQUEST_PASSWORD_COMPLEXITY"/>
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
```

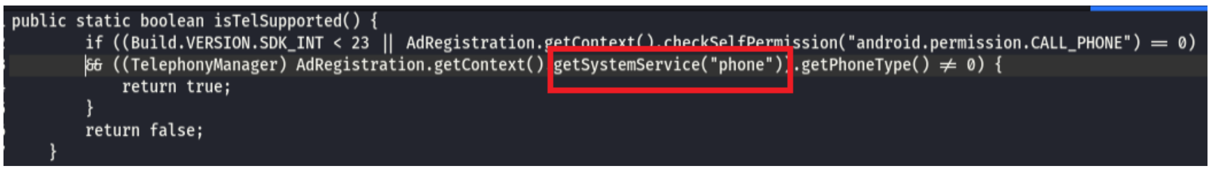
From the permission list used by this apk, we can observe there are plenty of permissions given which are dangerous and used by the malware to perform illegal and malicious tasks.

"READ_SMS, WRITE_SMS, SEND_SMS, RECEIVE_SMS" is used to steal the OTP, verification codes, message body (hiding, forwarding, intercepting). Using the broadcast receiver, they are broadcasting the SMS and this malicious app will have listeners to capture and it will be sent to the attacker.

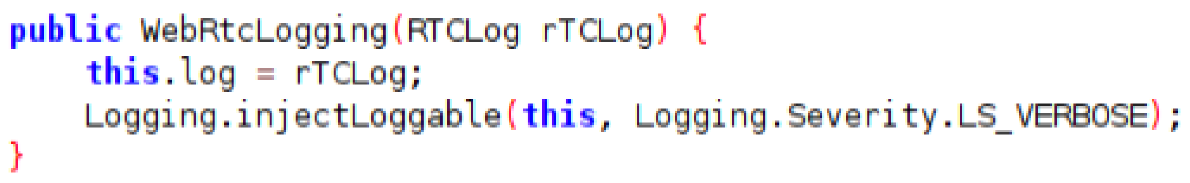


```
<receiver android:name="flip.enact.merit.receiver.PPpaoJAOWPDjaw" android:permission="android.permission.BROADCAST_SMS
    <intent-filter android:priority="999">
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        <action android:name="android.provider.Telephony.SMS_DELIVER"/>
    </intent-filter>
</receiver>
```

Telephony manager is responsible for SIM-based calls and also VOIP-based calls and they use WebRTC based API to transmit the calls and SMS related data.

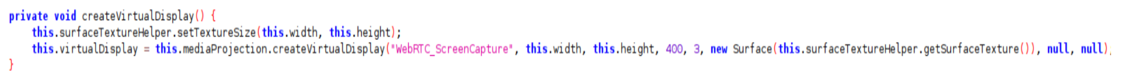


```
public static boolean isTelSupported() {
        if ((Build.VERSION.SDK_INT < 23 || AdRegistration.getContext().checkSelfPermission("android.permission.CALL_PHONE") == 0)
        && ((TelephonyManager) AdRegistration.getContext().getSystemService("phone")).getPhoneType() ≠ 0) {
            return true;
        }
        return false;
    }
```



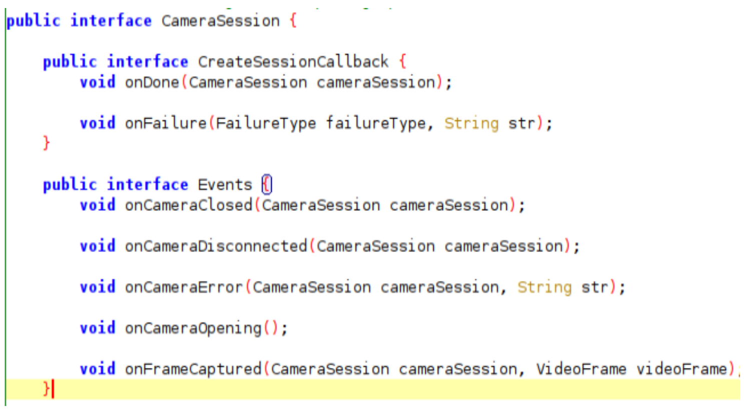
```
public WebRtcLogging(RTCLog rTCLog) {
    this.log = rTCLog;
    Logging.injectLoggable(this, Logging.Severity.LS_VERBOSE);
}
```

This malware even creates a virtual display to the attacker here using the WebRTC. The attacker remotely duplicates the display and monitors all the activity performed by the victim. They also use the same technique to launch fake pages, fake screens due to the malicious code embedded in this apk.



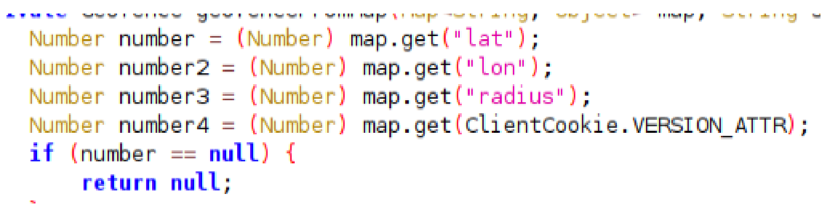
```
private void createVirtualDisplay() {
    this.surfaceTextureHelper.setTextureSize(this.width, this.height);
    this.virtualDisplay = this.mediaProjection.createVirtualDisplay("WebRTC_ScreenCapture", this.width, this.height, 400, 3, new Surface(this.surfaceTextureHelper.getSurfaceTexture()), null, null),
}
```

Attacker can even take control of the camera on the victim device and manipulate the stream, open camera, close camera, and record camera.



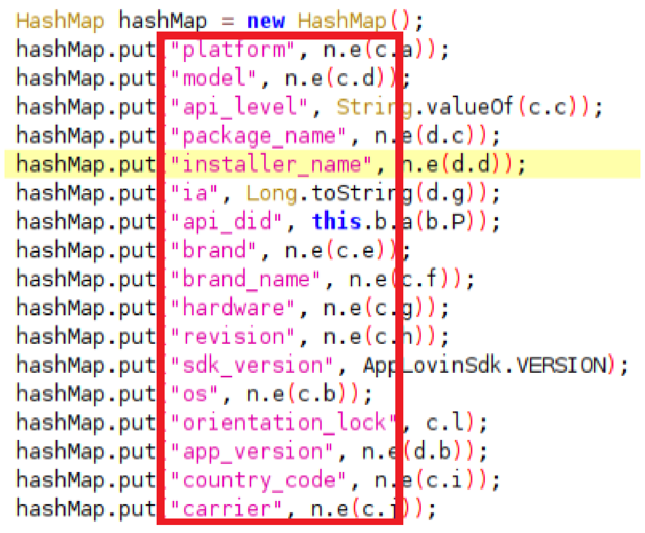
```
public interface CameraSession {

    public interface CreateSessionCallback {
        void onDone(CameraSession cameraSession);

        void onFailure(FailureType failureType, String str);
    }

    public interface Events {
        void onCameraClosed(CameraSession cameraSession);

        void onCameraDisconnected(CameraSession cameraSession);

        void onCameraError(CameraSession cameraSession, String str);

        void onCameraOpening();

        void onFrameCaptured(CameraSession cameraSession, VideoFrame videoFrame),
    }
```

The attacker steals the exact geolocation using the geofence function and monitors the location of the victim device.



```
    Number number = (Number) map.get("lat");
    Number number2 = (Number) map.get("lon");
    Number number3 = (Number) map.get("radius");
    Number number4 = (Number) map.get(ClientCookie.VERSION_ATTR);
    if (number == null) {
        return null;
```

The attacker is stealing the complete device information which includes device type, device model, API level of the device. Also, the packages and apps installed on the device along with the source from where they were installed. He even enumerates the hardware information, and type of operating system the handset is using along with country code and other info which can be seen in the below screenshot, and sending it to the attacker's server.



```
HashMap hashMap = new HashMap();
hashMap.put("platform", n.e(c.a));
hashMap.put("model", n.e(c.d));
hashMap.put("api_level", String.valueOf(c.c));
hashMap.put("package_name", n.e(d.c));
hashMap.put("installer_name", n.e(d.d));
hashMap.put("ia", Long.toString(d.g));
hashMap.put("api_did", this.b.a(b.P));
hashMap.put("brand", n.e(c.e));
hashMap.put("brand_name", n.e(c.f));
hashMap.put("hardware", n.e(c.g));
hashMap.put("revision", n.e(c.h));
hashMap.put("sdk_version", AppLovinSdk.VERSION);
hashMap.put("os", n.e(c.b));
hashMap.put("orientation_lock", c.l);
hashMap.put("app_version", n.e(d.b));
hashMap.put("country_code", n.e(c.i));
hashMap.put("carrier", n.e(c.j));
```

**IOCS:**

| |
|---|
| https://c.amazon-adssystem.com/ |
| http:// https://grs/dbankcloud.com/ |

**MITRE Techniques:**

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

| Execution (TA0041) |
|---|
| Deliver Malicious App via Other Means (T1476) |

**Subex Secure Protection**

Subex Secure detects the android sample as “SS_Gen_Andro_TeabotBankerTrojan_A”.

**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, analysed, 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.