black hat ASIA 2021
May 6-7, 2021
BRIEFINGS

(Un)protected Broadcasts in Android 9 and 10

Dr. Ryan Johnson - Kryptowire

Dr. Mohamed Elsabagh - Kryptowire

Dr. Angelos Stavrou - Kryptowire

Agenda

Intents

Protected Broadcasts

(Un)protected Broadcast Vulnerability

Notable Instances of the Vulnerability

Resolution

Disclosure

Conclusions

Who we are

kryptowire

Kryptowire was jump-started by Defense Advanced Research Projects Agency (DARPA) in late 2011 and R&D supported by Department of Homeland Security Science & Technology (DHS S&T) and National Institute of Standards and Technology (NIST)

Enterprise Mobile Security: Software Assurance, Developer Integration & Mobile Device Management (MDM), Threat Feed, & Security Analytics

Ryan Johnson

Mohamed Elsabagh

Angelos Stavrou

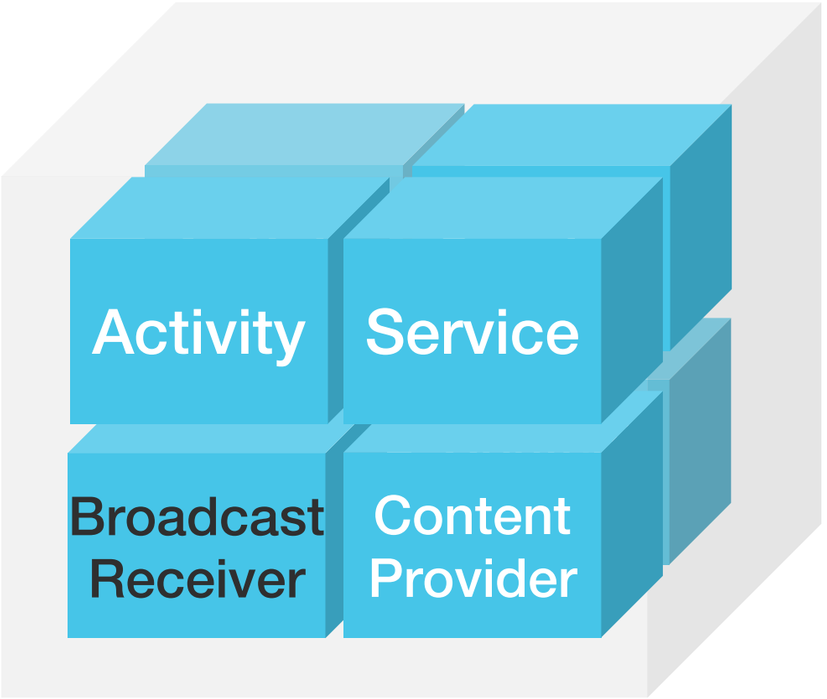
App Components

Android apps are composed of app components

Can be started independently and perform dedicated tasks

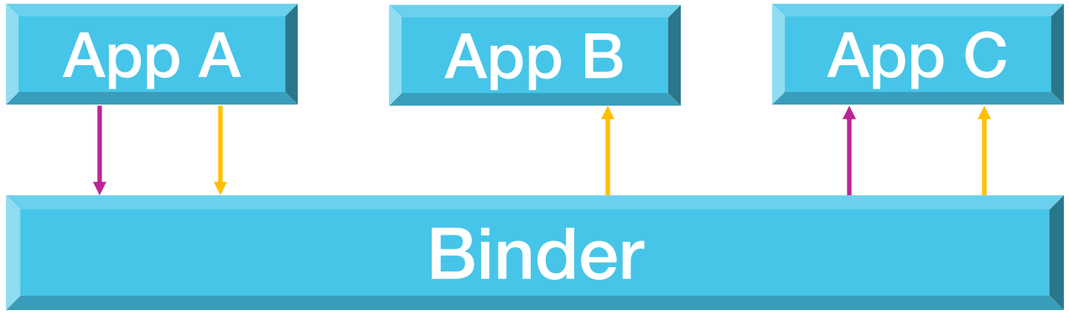
Declared in an app's `AndroidManifest.xml` file

```
<receiver android:name=".NfcBootCompletedReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```



Intents

IPC messages sent within and between apps



Explicit Intents specify an exact destination app component receiver, whereas *implicit Intents* do not and rely solely on actions to determine the receiver(s)

Implicit

```
Intent intent = new Intent("android.intent.action.BOOT_COMPLETED");
sendBroadcast(intent);
```

Explicit

```
Intent intent = new Intent("android.intent.action.BOOT_COMPLETED");
intent.setClassName("com.android.nfc", "com.android.nfc.NfcBootCompletedReceiver");
sendBroadcast(intent);
```

Protected Broadcasts

Prevents unauthorized entities from sending system broadcast Intents with specific actions

- Commonly used by the Android Framework and system apps

```
<protected-broadcast android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
<protected-broadcast android:name="android.intent.action.BOOT_COMPLETED"/>
<protected-broadcast android:name="android.intent.action.LOCALE_CHANGED"/>
```

Can generally be received by any process that registers for a protected broadcast action, although the sender can require that the receiver possess a permission in order to receive it

```
<receiver android:name=".NfcBootCompletedReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

Who can send protected broadcasts?

```
final boolean isCallerSystem;
switch (UserHandle.getAppId(callingUid)) {
    case ROOT_UID:
    case SYSTEM_UID:
    case PHONE_UID:
    case BLUETOOTH_UID:
    case NFC_UID:
    case SE_UID:
        isCallerSystem = true;
        break;
    default:
        isCallerSystem = (callerApp != null) && callerApp.persistent;
        break;
}
// First line security check before anything else: stop non-system apps from
// sending protected broadcasts.
if (!isCallerSystem) {
    if (isProtectedBroadcast) {
        String msg = "Permission Denial: not allowed to send broadcast "
                + action + " from pid="
                + callingPid + ", uid=" + callingUid;
        Slog.w(TAG, msg);
        throw new SecurityException(msg);
    } ...
```

UID checks

System processes with specific UIDs and system apps with the `android:persistent` attribute set to `true` in their `AndroidManifest.xml` file

Only system apps can be persistent

`SecurityException` is thrown when the caller is not part of the system (i.e., `isCallerSystem` is `false`)

Android 9 AOSP snippet from ActivtiyManagerService.java

Not all system apps are created equal

Android apps have an APK file with a path on the file system

```
$ adb shell pm list package -f
/system/priv-app/SettingsGoogle/SettingsGoogle.apk=com.android.settings
/system/app/EasterEgg/EasterEgg.apk=com.android.egg
/vendor/app/TimeService/TimeService.apk=com.qualcomm.timeservice
...
```

Apps contained in a `priv-app` directory are considered "privileged"

- Can be explicitly granted permissions without user involvement through `xml` files

Which app directories can ...?

Android Version	*Can* Register Protected Broadcasts	*Cannot* Register Protect Broadcasts
11	/system/framework, /system/app, /system/priv-app, /vendor/app, /vendor/priv-app, /vendor/overlay, /odm/app, /odm/priv-app, /odm/overlay, /oem/app, /oem/priv-app, /oem/overlay, /product/app, /product/priv-app, /product/overlay /system_ext/app, /system_ext/priv-app, & /system_ext/overlay	/data/app
10	/system/framework, /system/priv-app, /vendor/priv-app, /odm/priv-app, /product/priv-app, & /product_services/priv-app	/data/app, /system/app, /vendor/app, /odm/app, /oem/app, /product/app, /product_services/app, /vendor/overlay, /product_services/overlay, /product/overlay, /odm/overlay, & /oem/overlay
9	/system/framework, /system/priv-app, /vendor/priv-app, /odm/priv-app, & /product/priv-app	/data/app, /system/app, /vendor/app, /odm/app, /oem/app, & /product/app, /vendor/overlay, & /product/overlay
8	/system/framework, /system/app, /system/priv-app, /vendor/app, /oem/app, & /vendor/overlay	/data/app

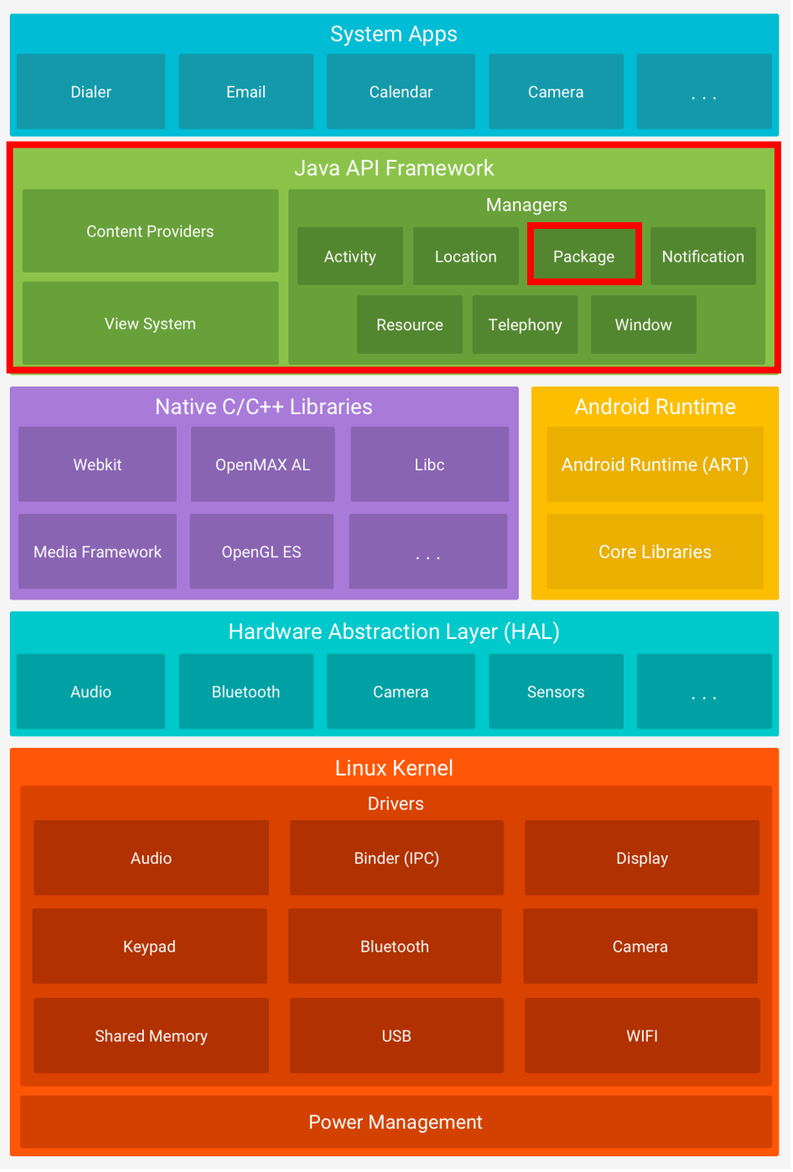
PackageManagerService

Back-end service that provides information about installed apps via the Android Framework APIs

`PackageManager` → `IPackageManager` → `PackageManagerService`

The system uses `PackageManagerService` to scan the partitions on system startup for apps and parses their manifests to determine installed apps and configure their broadcast permissions

- `/system`, `/vendor`, `/product`, `/odm`, `/oem`, `...`



PackageManagerService (Android 10)

```
// Collect privileged system packages.
final File privilegedAppDir = newFile(Environment.getRootDirectory(), "priv-app");
    scanDirTracedLI(privilegedAppDir,
        mDefParseFlags
        | PackageParser.PARSE_IS_SYSTEM_DIR,
        scanFlags
        | SCAN_AS_SYSTEM
        | SCAN_AS_PRIVILEGED,
        0);

// Collect ordinary system packages.
final File systemAppDir = new File(Environment.getRootDirectory(), "app");
    scanDirTracedLI(systemAppDir,
        mDefParseFlags
        | PackageParser.PARSE_IS_SYSTEM_DIR,
        scanFlags
        | SCAN_AS_SYSTEM,
        0);
```

Apps in the `/system/priv-app` directory are scanned with the `SCAN_AS_SYSTEM` and `SCAN_AS_PRIVILEGED` flags

Apps in the `/system/app` directory are scanned with the `SCAN_AS_SYSTEM` flag

Apps that were not scanned with the `SCAN_AS_PRIVILEGED` flag (i.e., apps in an `app` directory) have their protected broadcast declarations ignored

```
if ((scanFlags & SCAN_AS_PRIVILEGED) == 0) {
    // clear protected broadcasts
    pkg.protectedBroadcasts = null;
    // ignore export request for single user receivers
    if (pkg.receivers != null) {
        for (int i = pkg.receivers.size() - 1; i >= 0; --i) {
            final PackageParser.Activity receiver = pkg.receivers.get(i);
            if ((receiver.info.flags & ActivityInfo.FLAG_SINGLE_USER) != 0) {
                receiver.info.exported = false;
            }
        }
    }
...
}
```

(Un)protected broadcast vulnerability

System apps that are not present in a `priv-app` directory on Android 9 and 10 will offer no protection for the protected broadcast actions the app declares, allowing any app to send them

- CVE-2020-0391 - CVSS 3.X Base Score = 7.3

① `Clutch.apk` declares protected broadcast(s) in app manifest

`/system/app/Clutch/Clutch.apk`

`system_server`

② `system_server` parses protected broadcasts in all app manifests but ignores any from an app not contained in a `priv-app` directory

③ `Third-party App` sends (un)protected broadcast Intents with actions that apps in an `app` directory that are offered no protection at runtime

`Third-party App`

④ `Clutch.apk` apps process (un)protected broadcast Intents sent from third-party app resulting in privilege escalation since it appears to be from an authorized sender

priv-app vs. app directories

Third-party app sends a broadcast Intent that is declared as protected → `am broadcast -a android.perfdump.action.EXT_EXEC_SHELL ...`

Declared in

```
<protected-broadcast
android:name="android.perfdump
.action.EXT_EXEC_SHELL"/>
```


```
Security exception: Permission Denial:
not allowed to send broadcast
android.perfdump.action.EXT_EXEC_SHELL
from pid=13064, uid=10282
```

```
Broadcasting: Intent { act=
android.perfdump.action.EXT_EXEC_SHELL
flg=0x400000 (has extras) }
Broadcast completed: result=0
```

Declared in

Threat Model

Attack Surface: Exported app components that register for (un)protected broadcast actions where the extent depends on the apps present on the device and their file system locations

Attack Vector: Broadcasting intents with (un)protected broadcast actions

Attack Requirements: Local app on the device that can broadcast Intents

Attack Goal: Privilege escalation due to sending spoofed broadcast intents that appears to be from the system when they are really from a third-party app

Android Versions Affected

When reported in May 2020

Android 9

Android 10

Android 11
Developer
Preview 3

Google Pixel 4 (Un)protected Broadcasts

Google Pixel 4 Android 10 build contains 3 apps with (un)protected broadcasts

- `google/flame/flame:10/QQ2A.200405.005/6254899:user/release-keys`

Package Name	(Un)protected Broadcasts	App Path on Device
com.qualcomm.qti.uceShimService	4	/product/app/uceShimService/uceShimService.apk
com.google.SSRestartDetector	2	/product/app/SSRestartDetector/SSRestartDetector.apk
com.android.service.ims.presence	4	/system/app/PresencePolling/PresencePolling.apk

PresencePolling app Overview

Pre-installed app with package name of `com.android.service.ims.presence` that facilitates Rich Communication Services (RCS)

- Path: `/system/app/PresencePolling/PresencePolling.apk`
- Executes with shared UID: `android.uid.phone`

IP Multimedia Subsystem (IMS) external project hosted on android.googlesource.com and is present on Google Pixel 3 and Google Pixel 4 devices

Cause file corruption of the internal database that “mirrors” the device’s official contacts provider

Perfdump app Overview

Pre-installed app with a package name of `com.qualcomm.qti.perfdump` that profiles processes using Linux `perf` tools

- App path: `/system/app/Perfdump/Perfdump.apk`
- Executes with shared UID: `android.uid.system`

Vulnerable versions when path is `/<partition>/app/Perfdump/Perfdump.apk`

- Version code: 8, Version Name: 3.0.1
- Version code: 7, Version Name: 2.1.1

Command injection vulnerability due to a `protected-broadcast` not being protected at runtime

Perfdump app Manifest

```
<protected-broadcast android:name="android.perfdump.action.START_ERROR"/>
<protected-broadcast android:name="android.perfdump.action.DUMP_FINISH"/>
<protected-broadcast android:name="android.perfdump.action.CLEAR_FINISH"/>
<protected-broadcast android:name="android.perfdump.action.EXT_START_TRACE"/>
<protected-broadcast android:name="android.perfdump.action.EXT_DUMP_TRACE"/>
<protected-broadcast android:name="android.perfdump.action.EXT_EXEC_SHELL"/>
<protected-broadcast android:name="android.perfdump.action.EXT_FEEDBACK"/>
```

Protected broadcasts that the Perfdump app declares

```
<receiver android:name=".StaticReceiver">
    <intent-filter>
        <action android:name="android.perfdump.action.EXT_START_TRACE"/>
        <action android:name="android.perfdump.action.EXT_DUMP_TRACE"/>
        <action android:name="android.perfdump.action.EXT_EXEC_SHELL"/>
        <action android:name="android.intent.action.DEVICE_STORAGE_LOW"/>
        <action android:name="android.intent.action.DEVICE_STORAGE_OK"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.provider.Telephony.SECRET_CODE"/>
        <data android:host="73733867" android:scheme="android_secret_code"/>
    </intent-filter>
</receiver>
```

Perfump app component that registers for protected broadcast actions that it declares

Perfdump app Vulnerability Details

Command injection vulnerability - CVE-2020-11164 – CVSS 3.X Base Score = 7.8

- Executes arbitrary commands as `system` using `sh -c <command_to_execute>`

```
Intent intent = new Intent("android.perfdump.action.EXT_EXEC_SHELL");
intent.setClassName("com.qualcomm.qti.perfdump", "com.qualcomm.qti.perfdump.StaticReceiver");
intent.putExtra("callerPackageName", "com.test");
intent.putExtra("shellCommand", <command_to_execute>);
sendBroadcast(intent);
```

Qualcomm advisory ranked the vulnerability as *high severity* and listed the affected chipsets:

- Agatti, APQ8096AU, APQ8098, Bitra, Kamorta, MSM8909W, MSM8917, MSM8940, Nicobar, QCA6390, QCM2150, QCS605, Rennell, SA6155P, SA8155P, Saipan, SDA660, SDM429W, SDM450, SDM630, SDM636, SDM660, SDM670, SDM710, SM6150, SM7150, SM8150, SM8250, SXR1130, SXR2130

Perfdump app Affected Devices Sample

Vendor	Model	Product Name	Android Version	App Version Code	App Version Name
Sony	Xperia 1	802SO	9	8	3.0.1
Nokia	7 Plus	B2N_sprout	9	7	2.1.1
Fairphone	Fairphone 3	FP3	9	8	3.0.1
Meizu	Note 9	meizunote9	9	7	2.1.1
Meizu	16Xs	meizu16Xs	9	8	3.0.1
Xiaomi	Poco F1	beryllium	9	7	2.1.1
Xiaomi	Mi 9	cepheus	9	7	2.1.1
Xiaomi	Mi 8	dipper	9	7	2.1.1
Xiaomi	Mi 8 Pro	equuleus	9	7	2.1.1
Xiaomi	Mi Max 3	nitrogen	9	7	2.1.1
Xiaomi	Mi Mix 3	perseus	9	7	2.1.1

QMMI app QMMI Overview

Pre-installed app with package name of `com.qualcomm.qti.qmmi` that allows the user to test various hardware capabilities

- App path: `/system/app/Qmmi/Qmmi.apk`
- Executes with shared UID: `android.uid.system`

Vulnerable version when path is `/<partition>/app/Qmmi/Qmmi.apk`

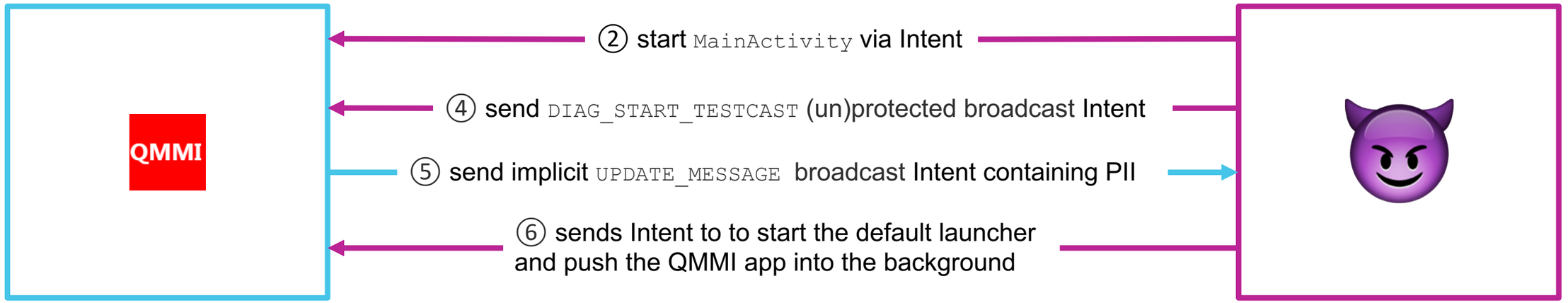
- Version code: 400, Version Name: 4.0

Programmatically obtain IMEI1, IMEI2 (if present), Wi-Fi MAC address, Bluetooth address, and serial number from a zero-permission app - CVE-2021-1929 (Currently reserved)

QMMI app Exploitation Workflow

⓪ QMMI app is located in an `app` directory and declares protected broadcasts

```
<protected-broadcast android:name="qualcomm.qti.qmmi.DIAG_START_TESTCAST_ACTION"/>
```



③ register for the action named `qualcomm.qti.qmmi.DIAG_START_TESTCAST`

① register for the action named `qualcomm.qti.qmmi.UPDATE_MESSAGE`

The (un)protected broadcast fix

https://android.googlesource.com/platform/frameworks/base/+/860fd4b6a2a4fe5d681bc07f2567fdc84f0d1580

- com/android/server/pm/PackageManagerService.java

```
if ((scanFlags & SCAN_AS_SYSTEM) != 0  {
    ...
  else
    // non system apps can't be flagged as core
    pkg.coreApp = false;
    // clear flags not applicable to regular apps
...
+   // clear protected broadcasts
+   pkg.protectedBroadcasts = null;
...
```

```
if ((scanFlags & SCAN_AS_PRIVILEGED) == 0) {
-   // clear protected broadcasts
-   pkg.protectedBroadcasts = null;
    // ignore export request for single user receivers
    if (pkg.receivers != null) {
        for  int i = pkg.receivers.size() - 1; i >= 0; --i) {
...
```

Backported fix to vulnerable devices that are still supported

(Un)protected broadcast disclosure timeline

5/08/2020: Initial disclosure to Android Security Team and affected vendors

6/08/2020: Submitted vulnerability report to Google's IssueTracker

6/09/2020: Submission acknowledged

6/15/2020: Google committed the fix - 860fd4b6a2a4fe5d681bc07f2567fdc84f0d1580

6/18/2020: Google finished their initial assessment and ranked the severity as "High"

8/21/2020: Google assigned CVE-2020-0391 for the vulnerability.

9/08/2020: Google changed the vulnerability status to "fixed" and provided bug bounty

Conclusions

Use defense-in-depth design principle by employing multiple forms of access control to guard app components that receive protected broadcasts

Providing explicit feedback to developers via a runtime warning for pre-installed apps may help identify these cases in the future

Ensure there is good communication when making significant changes to the system

Contact Info

Dr. Ryan Johnson
VP of Research

rjohnson@kryptowire.com

Dr. Mohamed Elsabagh
Director of Research

melsabagh@kryptowire.com

Dr. Angelos Stavrou
Chief Scientific Officer

astavrou@kryptowire.com

http://www.kryptowire.com