A Tale of Three Brothers: Three Android Privacy Bugs

(CVE-2018-9489 / CVE-2018-9581 / CVE-2018-15835)

@nightwatchcyber

November 9th, 2018

Table of Contents

About Me

- I was a software developer most of my career and security bug bounty hunter on the side
- Currently work in application security full time **but I'm here personally, not on behalf of my employer**
- Have presented before at BSides Philly / DE / DC
- Was involved in some early anti-spam work:
  - Co-chaired IRTF's Anti Spam Research Group (ASRG)
  - Involved in IETF pre-standards work for SPF/DKIM
  - Created protocol for exchanging spam reports (MARF / RFC 5965)
- Helping with the "security.txt" proposal
- Also did some non-security standards work:
  - RFCs 4180 (CSV files) and 6922 (SQL MIME type)
  - Participated in W3C's CSV for the Web group

Some of my past CVEs

Assigned in 2018

- CVE-2018-6019 – Samsung Display Solutions app
- CVE-2018-0237 – Cisco AMP for Endpoints (MacOS)

Assigned in 2017

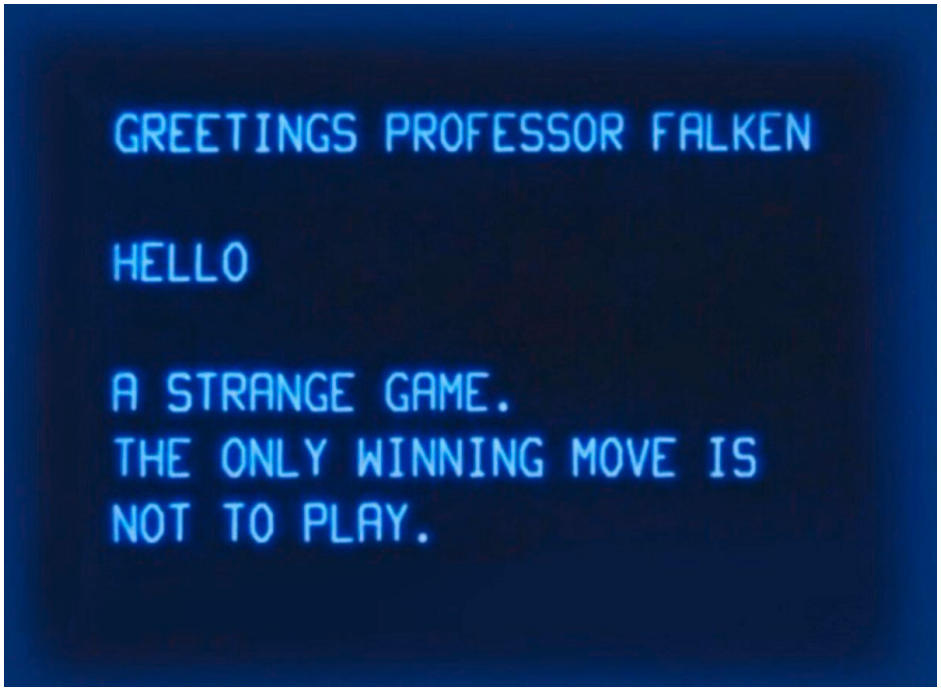
- CVE-2017-16905 – DuoLingo's TinyCards Android app
- CVE-2017-15882 – Private Internet Access Android app
- CVE-2017-15397 – Google's Chrome OS
- CVE-2017-14582 – Zoho 24x7 Poller for Android
- CVE-2017-13243 – Google's Android OS
- CVE-2017-11706 – Boozt Android app
- CVE-2017-9977 – AVG AntiVirus for MacOS
- CVE-2017-9245 – Google's News/Weather Android app
- CVE-2017-9045 – Google's I/O 2017 Android app
- CVE-2017-8878 – ASUS Routers
- CVE-2017-8877 – ASUS Routers
- CVE-2017-8769 – Facebook's WhatsApp app
- CVE-2017-5892 – ASUS Routers
- CVE-2017-5891 – ASUS Routers
- CVE-2017-5082 – Google's Chrome for Android

Assigned in 2016

- CVE-2016-6936 – Adobe's AIR SDK and Compiler
- CVE-2016-6723 – Google's Android OS
- CVE-2016-5672 – Intel's Crosswalk toolkit
- CVE-2016-5348 – Google's Android OS
- CVE-2016-5341 – Google's Android OS

DISCLAIMER!!!

Don’t do anything without talking to a (good) lawyer first!



Overview of Some Android Features:

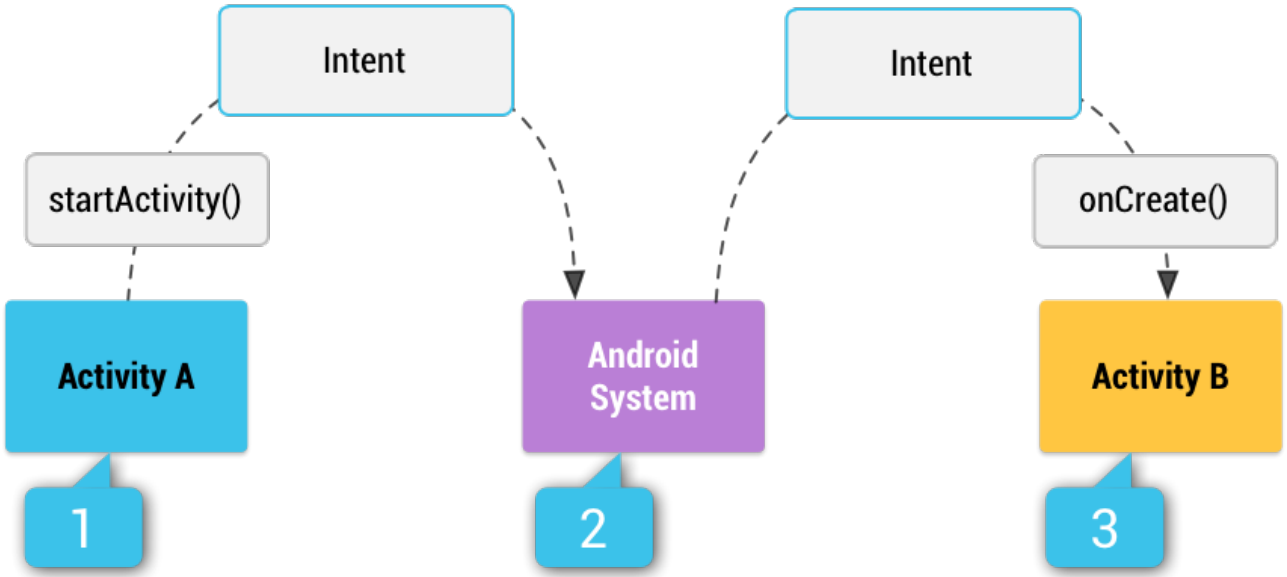
Intents and Broadcasts

Application Permissions

Intents and Broadcasts

- Applications on Android are sandboxed
- The OS does provide a means for events to be sent between app components, or between apps
- This is done by using “Intents”
- An “Intent” is a message that gets sent to other apps; can open screens or just carry data
- Can be restricted to specific receivers but developers often fail to do that
- If private data is included, other apps can sniff it
- Since Android 5.0, Local Broadcast Manager is included for Intent usage within the same app – it emulate broadcasts; apps often won’t use it :)

Intents and Broadcasts - Example




```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

(Code/photo from Android's official documentation)

Application Permissions

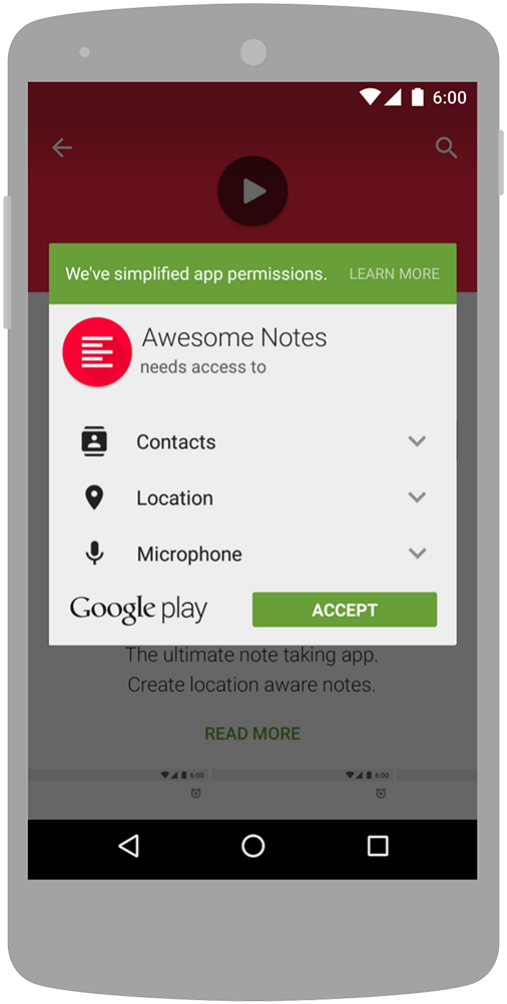
- A permissions structure exists for apps in Android
- The purpose is to protect privacy – required before either before sensitive data or system features are accessed by an app
- Permissions are requested via a manifest, which is an XML file (“AndroidManifest.xml”) inside the APK
- Permissions are handled differently depending on OS version, permission type, etc.
- Some are requested during install, some when the app runs for the first time, and some every time
- Some sensitive data or features can only be accessed by the OS or system apps (like Gplay)
- **Manifest permissions don’t affect intents!!!**

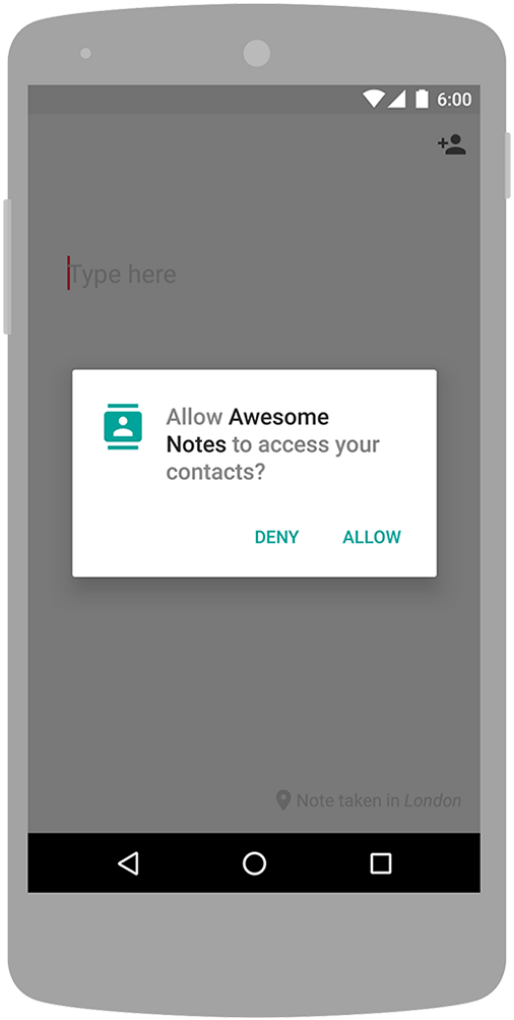
Application Permissions - Examples

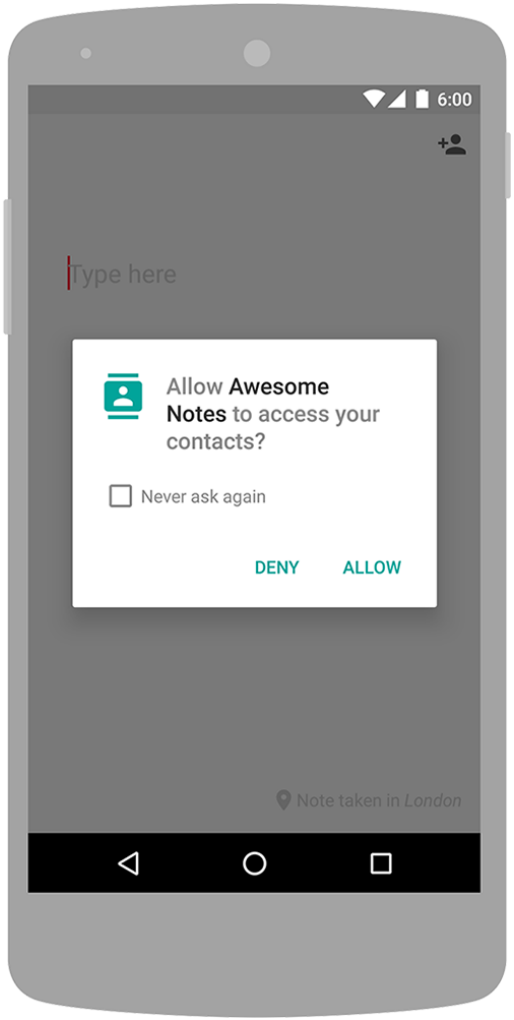
```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.snazzyapp">

    <uses-permission
android:name="android.permission.SEND_SMS"/>

    <application ...>
        ...
    </application>
</manifest>
```

(Code from Android's official documentation)

Application Permissions - Examples





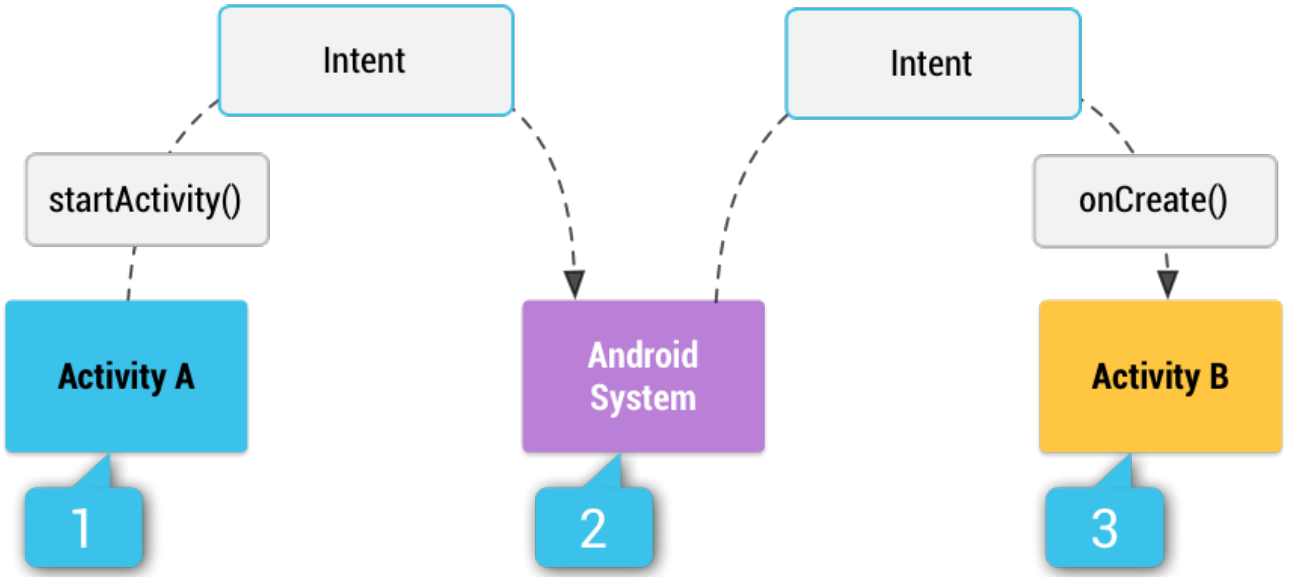


(Images from Android's official documentation)

What is the Root Cause for These Three Bugs?

(Public disclosure begins here)

Remember Intents?




```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

(Code/photo from Android's official documentation)

Root Cause

- Just like apps can broadcast Intents, so can the operating system itself
- Some of these are very useful – like letting apps know when the screen turns on, when the phone disconnects / reconnects to the Internet, when the phone goes to sleep, etc.
- Same security issues apply – by default, every app on the device can listen to Intents
- If sensitive data is carried in them, apps can sniff it
- Even if specific Android APIs require permissions, they don’t apply to Intents

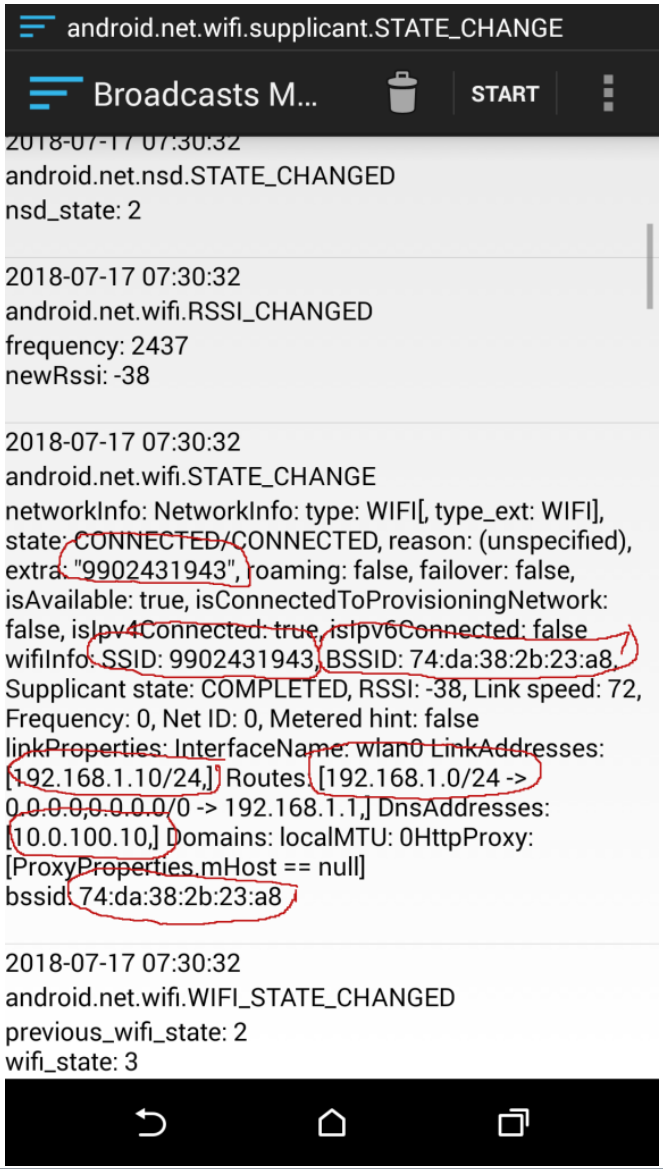
Root Cause

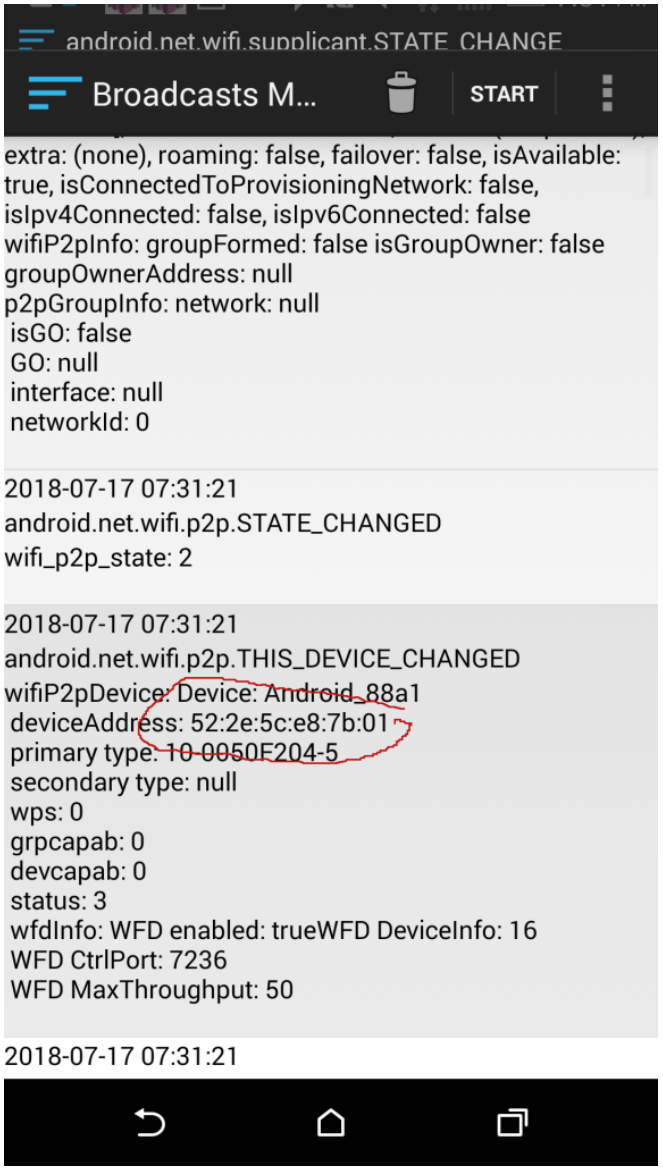
- The root cause of these three bugs is that Android OS is broadcasting sensitive data inside Intents, system-wide, on a regular basis
- For each of these, the data would or should normally be restricted by permissions
- These features date back years, some perhaps to Android 1.0
- It is trivial for apps to see and capture this data, no special permissions needed
- All of these are privacy-related

Exploiting via an app

- There are several apps available that can show Intents on a device, “Internal Broadcasts Monitor” by Vilius Kraujutis is one of them
  - Install Link and Source Code
- Just install, tap “Start” and watch the Intents fly by
- You may be able to see some of this data in the device logs via ADB
- This is how we discovered these – we were playing around with Intent monitoring during a pentest of an app and saw the OS generated Intents

Exploiting via an app - Examples





Exploiting via code

```
public class MainActivity extends Activity {
@Override
public void onCreate(Bundle state) {
    IntentFilter filter = new IntentFilter();

    filter.addAction(
    android.net.wifi.WifiManager.NETWORK_STATE_CHANGED_ACTION);

    filter.addAction(
android.net.wifi.WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);

    registerReceiver(receiver, filter);
}

BroadcastReceiver receiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
    Log.d(intent.toString());
    ….
}
};
```

Bug #1 - Battery Info

CVE-2018-15835

Not disclosed before

W3C Battery API Privacy

- Around 2014-2015, major browsers added a Battery Status API based on a W3C proposal
- The intention was to allow websites to switch to an energy saving mode as needed
- Some researchers (Lukasz Olejnik, and others) found privacy issues that can be exploited to track users, and were in fact exploited by websites in the wild
- **Surprise!**
- The API was changed or removed by most browsers

W3C Battery API Privacy

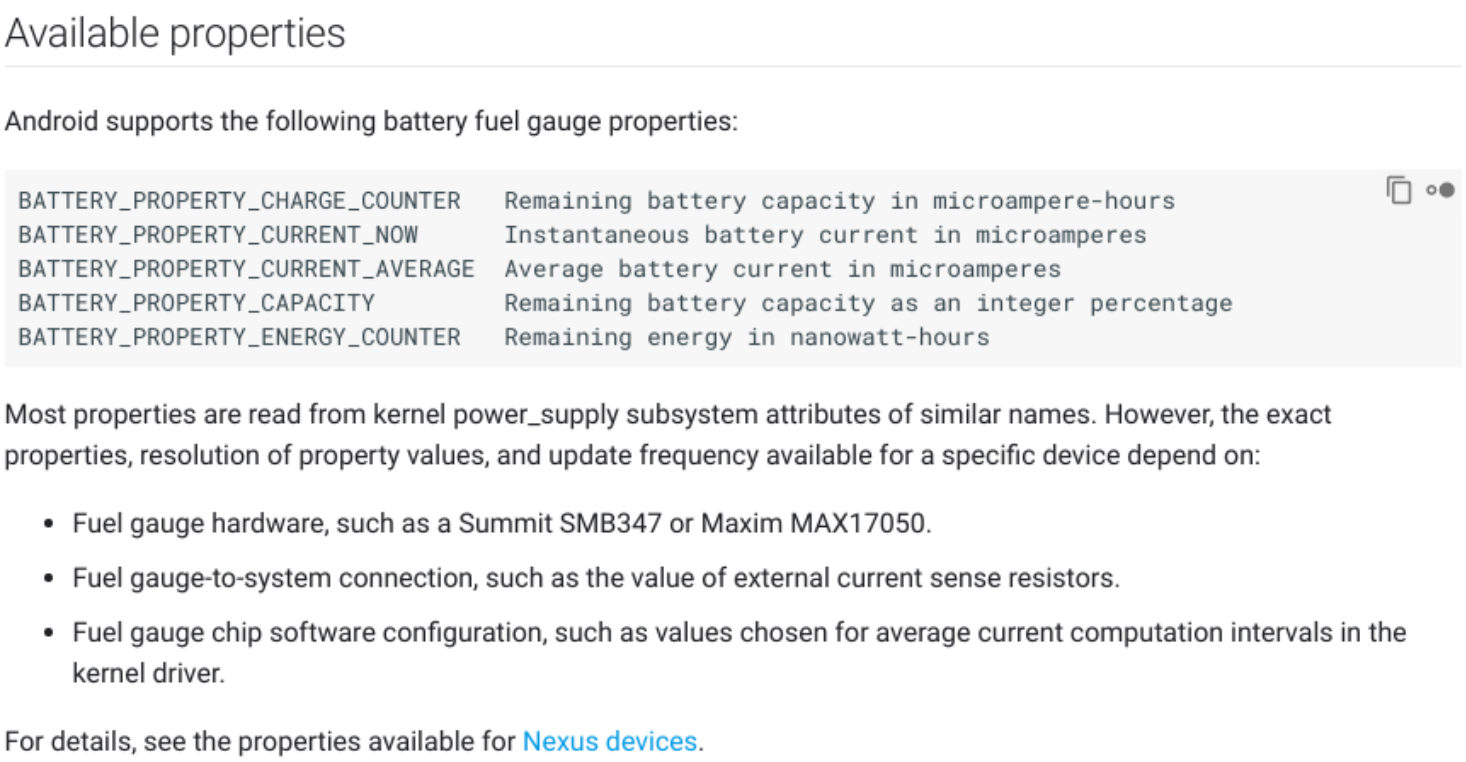
- The original paper describes privacy issues based on a single value (battery level) that is derived from a bunch of Linux UPower variables (voltage, battery capacity, etc).
- Issue with high-precision battery levels
- Can be used to fingerprint and track users across sites, and re-spawning within a short interval based on frequency of discharge and capacity
- Same research team looked at other sensors

W3C Battery API Privacy - References

- **“The Leaking Battery” (2015);** by Łukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz;
- **“Online tracking: A 1-million-site measurement and analysis” (2016);** by Steven Englehardt and Arvind Narayanan
- **“Battery Status Not Included: Assessing Privacy in Web Standards” (2017);** Łukasz Olejnik, Steven Englehardt, Arvind Narayanan; see also this blog post
- Additional academic research exists as well

The bug

- Android exposes battery information via Intents (“BATTERY_CHANGED”) and APIs (BatteryManager)
- No special permissions are required (but perhaps should be?)
- Information includes the following (from official docs):



Available properties

Android supports the following battery fuel gauge properties:

```
BATTERY_PROPERTY_CHARGE_COUNTER   Remaining battery capacity in microampere-hours
BATTERY_PROPERTY_CURRENT_NOW      Instantaneous battery current in microamperes
BATTERY_PROPERTY_CURRENT_AVERAGE  Average battery current in microamperes
BATTERY_PROPERTY_CAPACITY         Remaining battery capacity as an integer percentage
BATTERY_PROPERTY_ENERGY_COUNTER   Remaining energy in nanowatt-hours
```

Most properties are read from kernel power_supply subsystem attributes of similar names. However, the exact properties, resolution of property values, and update frequency available for a specific device depend on:

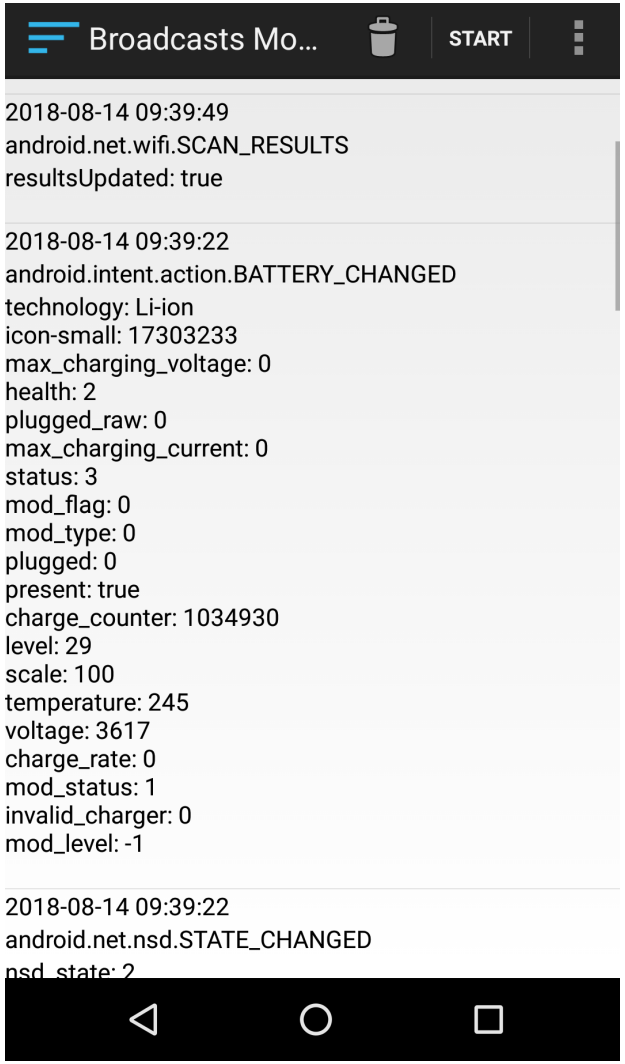
- Fuel gauge hardware, such as a Summit SMB347 or Maxim MAX17050.
- Fuel gauge-to-system connection, such as the value of external current sense resistors.
- Fuel gauge chip software configuration, such as values chosen for average current computation intervals in the kernel driver.

For details, see the properties available for Nexus devices.

The bug

- More information is exposed via this API than what the web battery API did - same privacy issues apply here
- In our limited testing, we were to re-identify devices within a short time based on their charging information
- Affects Android 5.0 or later, including forks
- More research is needed

Android Battery API Example




```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle state) {
        IntentFilter filter = new
IntentFilter();
filter.addAction(Intent.ACTION_BATTERY_CHANGED)
;

        registerReceiver(receiver, filter);
    }

    BroadcastReceiver receiver = new
BroadcastReceiver() {
        @Override
        public void onReceive(Context context,
Intent intent) {
                Log.d(intent.toString());
                ….
        }
    };
```

Vendor Response

- The bug was responsibly disclosed to the vendor in March 2018
- Vendor assessed the bug and set the severity as “NSBC” = “Not Security Bulletin-Class”
- “It was rated as not being a security vulnerability that would meet the severity bar for inclusion in an Android security bulletin.”
- No fix is planned or known at this time
- CVE-2018-15835 was assigned for tracking

Summary and Implications

- Any Android application can capture/monitor detailed battery information via Intents or the API without extra permissions **(but perhaps should require permissions?)**
- Affects versions of **Android 5.0 and later** including **forks** such as Kindle's FireOS
- Tracked under **CVE-2018-15835**, disclosed publicly here for the first time
- This can be used to **fingerprint** a particular device and **track users** across apps (untested)
- Can potentially be used to **re-spawn sessions** within a short time (confirmed via limited testing)
- **No fix or workaround is available right now**
- We don't know if this is being used "in the wild"

Bug #2 – RSSI Levels

CVE-2018-9581

Not disclosed before

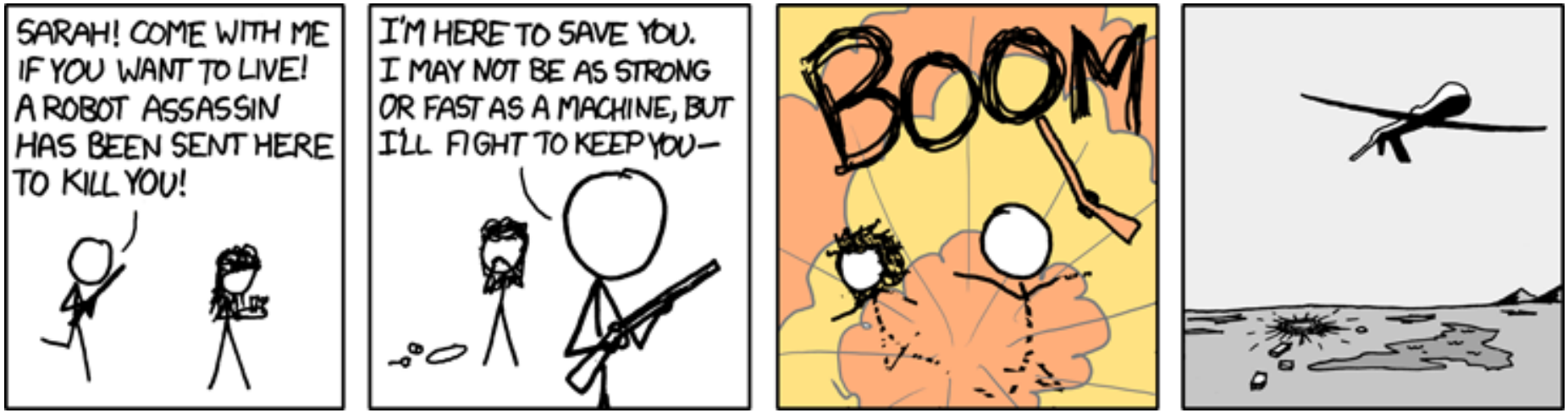
What is RSSI in regards to WiFi?

- RSSI or “Received Signal Strength Indicator” is a measure of how powerful a signal is on the client in relation to the access point
- As per IEEE standards, this is not a direct measurement like dbM, but a translated one
- RSSI can be on a scale from 0 to 255 but each chipset does it’s own thing
- Also used in Bluetooth and cellular connections, but differently

RSSI and GeoLocation

- RSSI can be used for indoor geolocation based on the access point since signal strength varies depending on the rooms and walls, **but isn’t always accurate**
- Also called indoor positioning, limited to small areas, not global like GPS
- 802.11mc (WiFi RTT) can also do this in Android 9
- **BUT, accessing the RTT API in Android 9, OR the normal Android WiFi API versions requires special permissions**

What Can You Do with Indoor Positioning? - Probably



(xkcd)

What Can You Do with Indoor Positioning? - More Likely

(xkcd)

What Can You Do with Indoor Positioning? - But Maybe this?

Adversarial WiFi Sensing

Yanzi Zhu[†], Zhujun Xiao[‡], Yuxin Chen[‡], Zhijing Li[†], Max Liu[‡],
Ben Y. Zhao[‡] and Haitao Zheng[‡]

[†]University of California, Santa Barbara [‡]University of Chicago

{yanzi, zhijing}@cs.ucsb.edu {zhujunxiao, yxchen, maxliu, ravenben, htzheng}@cs.uchicago.edu

The conclusion in this paper (emphasis added):

… our work brings up an inconvenient truth about wireless transmissions. While greatly improving our everyday life, they also unknowingly reveal information about ourselves and our actions. ***By designing a simple and powerful attack, we show that bad actors outside of a building can secretly track user presence and movement inside the building*** *by just passively listening to ambient WiFi transmissions (even if they are encrypted) ...*

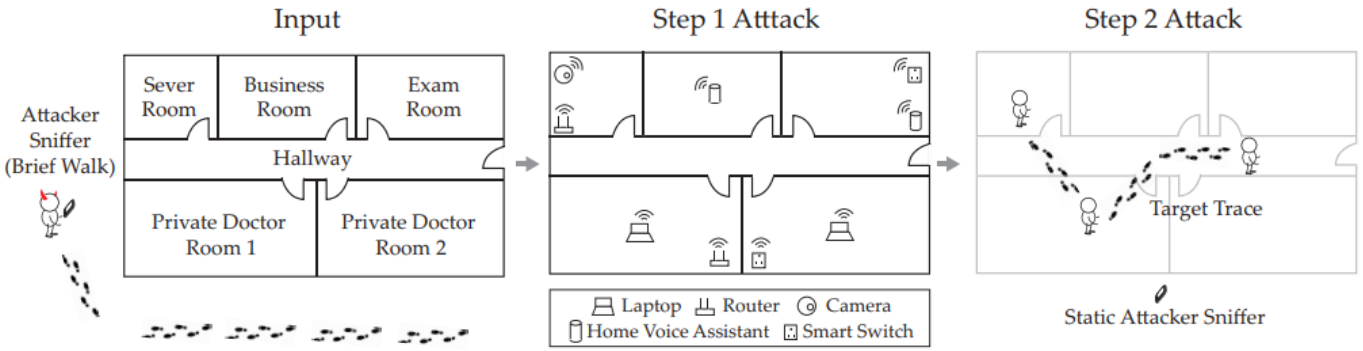


Figure 1: Illustration of our attack scenarios in a doctor's office.

(Text/Images from "Adversarial WiFi Sensing"; Yanzi Zhu, et al; arXiv:1810.10109; used with author permission)

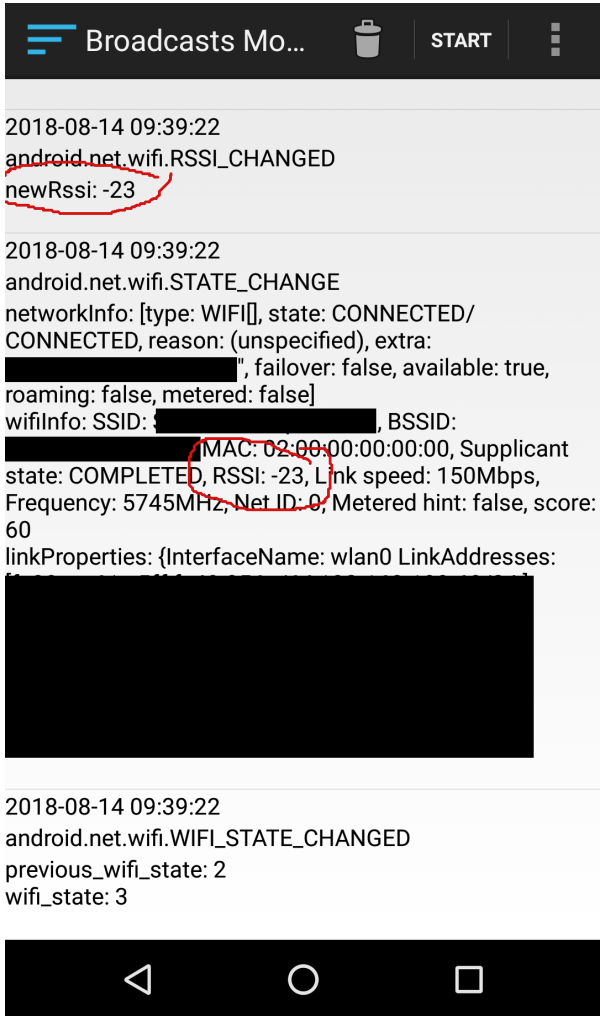
What Can You Do with Indoor Positioning?

- **You can (in theory) kill people** - Caleb Thompson gave several talks about his experience building such WiFi positioning system
- **HOWEVER** – what's more likely…
  …. is that indoor positioning can be used by places like malls to track shoppers
- We can imagine a retailer bundling such functionality in their apps and having that trigger when you walk into their store
- Recent research shows that you can track people moving indoors with greater accuracy than possible before

The bug

- Android exposes RSSI information via Intents (“STATE_CHANGE” and “RSSI_CHANGED”)
- STATE_CHANGE no longer exposes this in Android 9
- RSSI_CHANGED is still present in all versions of Android
- No special permissions are required
- To access the same information via the normal APIs (WiFi Manager) apps require special permissions
- Our testing confirmed that indoor positioning is possible (on a room level in a single building). Testing included multiple phones and OS versions, including forks
- RSSI numbers may not be consistent across phones

RSSI Examples




```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle state) {
        IntentFilter filter = new
IntentFilter();

filter.addAction(android.net.wifi.WifiManager.N
ETWORK_STATE_CHANGED_ACTION);
filter.addAction(android.net.wifi.WifiManager.R
SSI_CHANGED_ACTION);

        registerReceiver(receiver, filter);
    }

    BroadcastReceiver receiver = new
BroadcastReceiver() {
        @Override
        public void onReceive(Context context,
Intent intent) {
                Log.d(intent.toString());
                ….
        }
    };
```

Testing information

Testing for GeoLocation Within a Building

Our test used the following devices:

- Pixel 2, running Android 8.1.0, patch level July 2018
- Nexus 6P, running Android 8.1.0, patch level July 2018
- Moto G4, running Android 7.0, patch level April 2018
- Kindle Fire HD (8 gen), running Fire OS 5.6.10, which is forked from Android 5.1.1, updated April 2018
- Router used was ASUS RT-N56U running the latest firmware

(We included the Kindle Fire to show that forks of Android inherit this functionality)

Testing was done a multistory woodframe building with the following layout:

Room 1	[Router]	Room 2
Room 3		Room 4
	hallway	

Range of values collected during testing:

Room #	Pixel	Nexus	Moto G4	Kindle Fire
1	39 - 43	44	39 - 42	59 - 60
2	45 - 49	49 - 56	48 - 52	45 - 46
3	42 - 44	50	51 - 53	49 - 50
4	54 - 56	60 - 63	60 - 62	66

Vendor Response

- The bug was responsibly disclosed to the vendor March of 2018 as part of CVE-2018-9489; was split into a separate report in July 2018
- Vendor is still assessing the bug
- However, 90 days have passed since the separate report and we are disclosing it publicly
- No fix information is available, HOWEVER, one of the Intents (“STATE_CHANGED”) was fixed in Android 9 as part of CVE-2018-9489; still available in all lower versions; the other Intent (“RSSI_CHANGED”) is still present even in Android 9
- The vendor assigned CVE-2018-9581

Summary and Implications

- Any Android application can capture WiFi RSSI information without special permissions
- Affects all versions of Android
- CVE-2018-9581 assigned by the vendor, disclosed here for the first time
- Can be used for indoor positioning, confirmed via testing
- Partial fix exists as part of CVE-2018-9489; no additional fix information yet available
- We don’t know if this is being used “in the wild”

Bug #3 – MAC ID / WiFi Info

CVE-2018-9489

Disclosed originally in August 2018

WiFi APIs in Android

- Android has several APIs that can be used to retrieve information about the WiFi connection including the local IP address, WiFi network name, BSSID, signal band, etc.
- **BUT, accessing the WiFi API requires special permissions**
- Android doesn't recommend using hardware identifiers such as Android ID or IMEI
- Since Android 6.0, the MAC IDs of the device cannot be accessed via APIs – they always return "02:00:00:00:00:00"

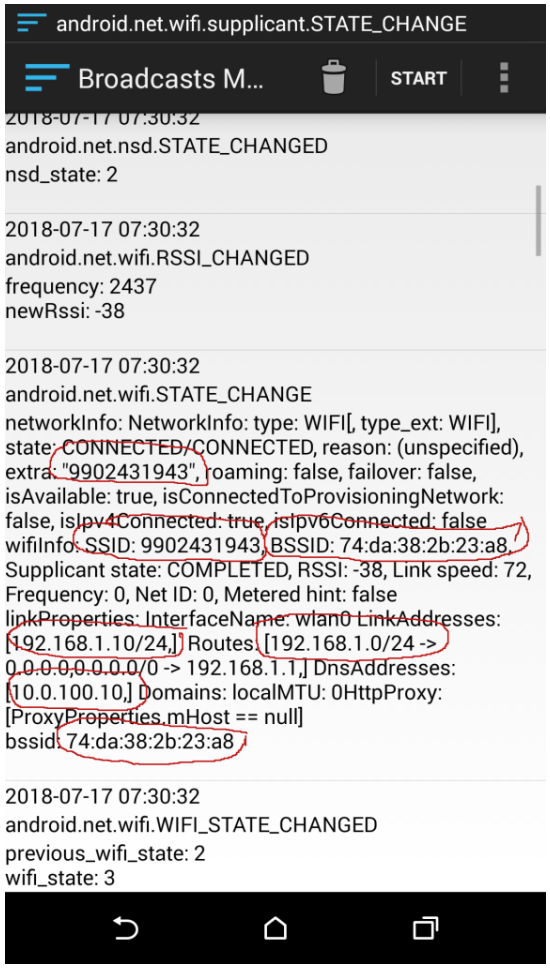
MAC IDs, Network Names and BSSIDs

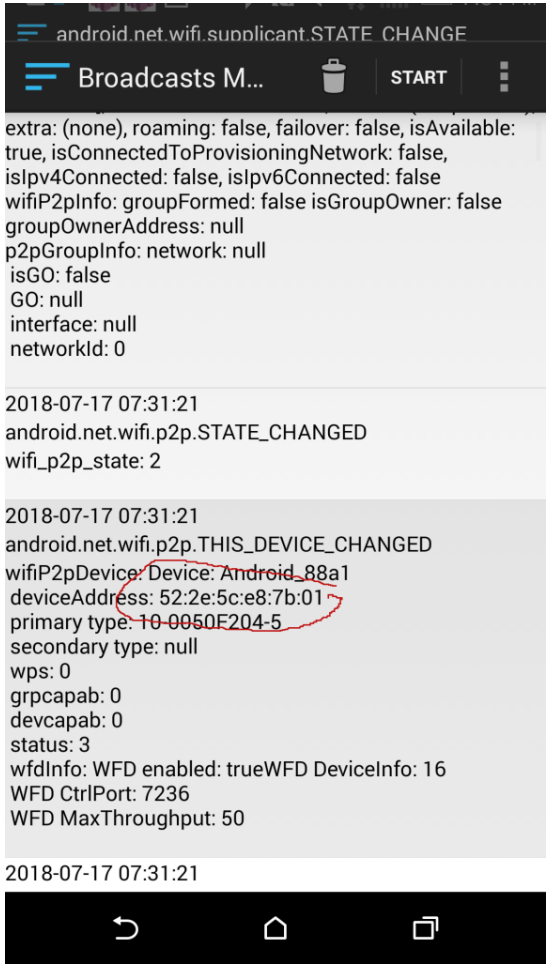
- **MAC IDs** are Ethernet identifiers assigned to hardware. Under normal circumstances they cannot be changed.
- In theory, they can be used to unmask the identity of the device owner via the supply chain; in practice it’s probably hard (Melissa virus story that didn’t happen).
- Most likely use is to uniquely identify devices
- Work has been done on randomizing MAC IDs during WiFi scans, but that doesn’t impact on-device use
- **BSSIDs** are hardware-derived identifiers for WiFi access points
- Can be used for rough geolocation, public and private databases (SkyHook) exist that map BSSIDs and network names to specific GPS coordinates

The bug

- Android exposes WiFi connection information including the MAC ID of the device, and BSSID of the router via Intents
- No special permissions are required
- On Android versions 6.0 and later, the correct MAC ID can be captured bypassing the privacy change in APIs
- However, on some Android versions one of the Intents hides the MAC ID, maybe related to the privacy change
- Can be used to uniquely identify and track devices
- BSSID information can be used for global geolocation
- There is other information including local IP address, gateway, signal band, DNS servers, etc.
- Testing confirms the issue across multiple phone models, Android versions and forks; all versions are believed to be affected

RSSI Examples






```
public class MainActivity extends
Activity {
    @Override
    public void onCreate(Bundle state)
{
        IntentFilter filter = new
IntentFilter();

filter.addAction(android.net.wifi.Wifi
Manager.NETWORK_STATE_CHANGED_ACTION);
filter.addAction(android.net.wifi.Wifi
Manager.RSSI_CHANGED_ACTION);

        registerReceiver(receiver,
filter);
    }

    BroadcastReceiver receiver = new
BroadcastReceiver() {
        @Override
        public void onReceive(Context
context, Intent intent) {

Log.d(intent.toString());
                ….
        }
    };
```

Vendor Response

- The bug was responsibly disclosed to the vendor in May 2018
- A fix was released as part of Android 9 in August 2018
- Public disclosure and our advisory was done in August 2018
- No fix is planned for lower versions of Android due to “breaking API changes”
- Tracked under CVE-2018-9489
- Unknown if being exploited “in the wild”

Summary / Q&A

- We discovered three privacy related bugs in Android OS, due to the use of Intents with sensitive data
- These allow exposure of information to on-device apps such as battery levels, WiFi signal strength (RSSI), device MAC ID, router BSSID, etc.
- Allow apps to fingerprint devices, track users, and geolocate devices (both locally and globally)
- These bugs bypass existing Android OS permissions and privacy changes
- Some have been fixed in Android 9, lower versions still affected
- Affects most if not all Android versions and devices are affected, including forks
- One bug has already been disclosed, we plan to publish advisories for the rest next week

Questions? Comments?

Email: research@nightwatchcybersecurity.com