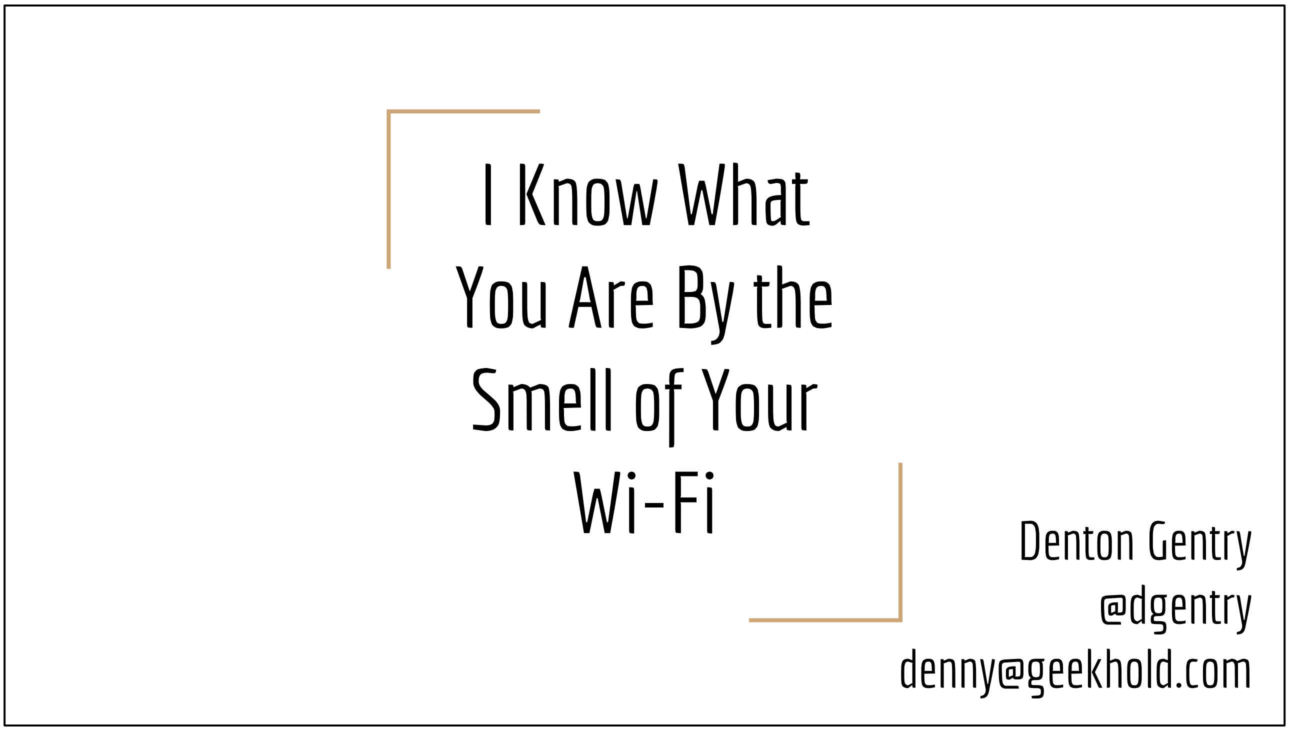


# I Know What You Are By the Smell of Your Wi-Fi

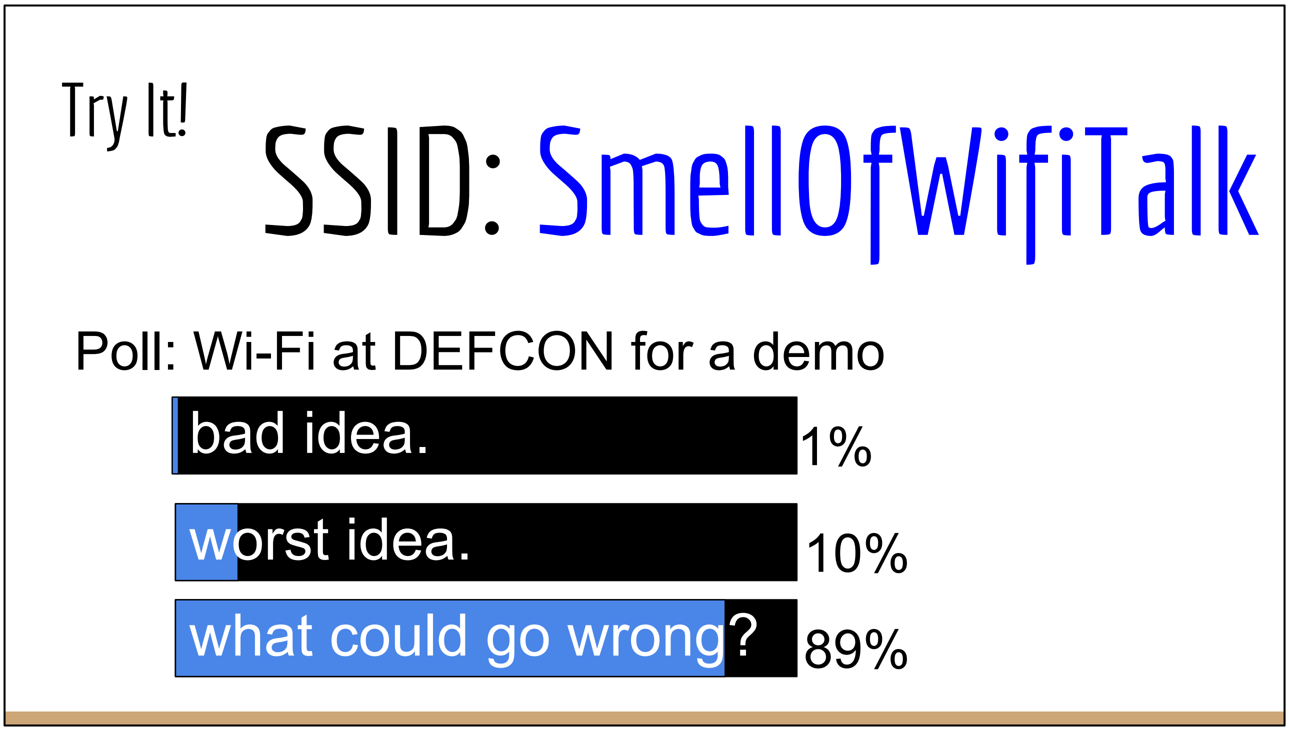
Denton Gentry
@dgentry
denny@geekhold.com

We’re here today to talk about a mechanism which identifies the type of device connecting to a Wi-Fi network. It can be quite specific: it can tell the difference between an iPhone 5 and an iPhone 5s, between a Samsung Galaxy S7 and an S8, between a Withings scale and a Nest Thermostat.

Classically, this kind of client detection would be called “fingerprinting” like the OS fingerprinting mechanisms in nmap. However, in modern usage the term “fingerprinting” has evolved to mean identification of specific users, such as browser fingerprinting to identify a specific individual.

As the mechanism discussed here identifies the species of the device, not the unique individual, we refer to it as Wi-Fi Taxonomy.

0:50



Try It!

## SSID: SmellOfWifiTalk

Poll: Wi-Fi at DEFCON for a demo

| | |
|---|---|
| bad idea. | 1% |
| worst idea. | 10% |
| what could go wrong? | 89% |

You can try it! Joining SmellOfWifiTalk will print the system's estimation of what your device is, for however long a rogue Wi-Fi network lasts at DEFCON. It will be printed in that terminal window up on the screen.

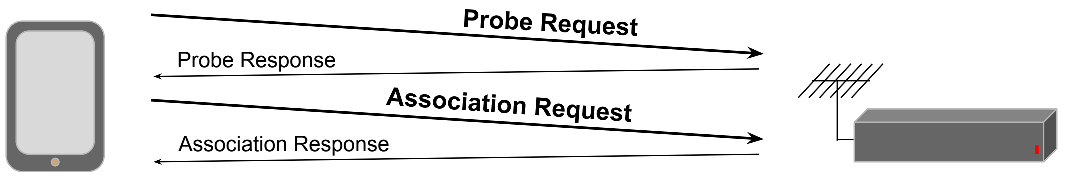
0:30

*Try It!* *SSID: SmellOfWifiTalk*

# MAC Sublayer Management Entity (MLME)

**Probe Request:** Asks nearby APs to respond.

**Association Request:** join the Wi-Fi network



The mechanism works by examining Wi-Fi Management frames, called MLME frames. These frames are used to join, leave, and configure the Wi-Fi network. They are not TCP/IP packets, they are not routable and they do not leave the Wi-Fi network.

We'll focus on two specific frames:

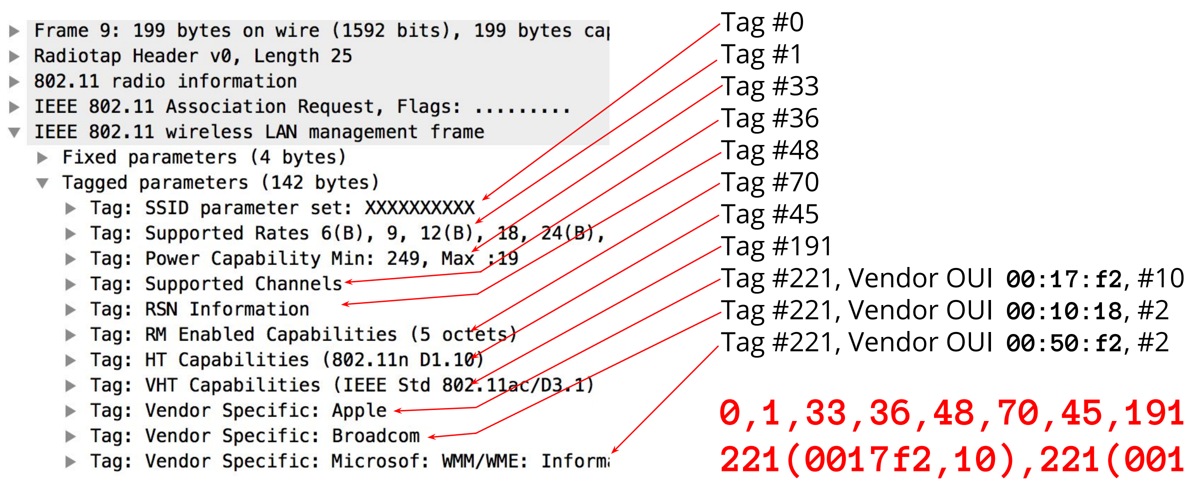
1. Probe Request, where a client can ask all nearby APs or one specific AP to respond. The client includes information about itself and its capabilities in the request, and the AP can respond with its own capabilities in the response.
2. We'll also look at the Association Request, when a client joins a Wi-Fi network. The client includes many of the same capabilities as in the Probe Request, plus a few more.

There are a bunch of other types of frames, like the Authentication frame or Action frames to adjust various parameters, but the Taxonomy mechanism we're talking about today relies on these two.

0:50

*Try It!  SSID: SmellOfWifiTalk*

## Signature: Information Elements



```
▶ Frame 9: 199 bytes on wire (1592 bits), 199 bytes ca
▶ Radiotap Header v0, Length 25
▶ 802.11 radio information
▶ IEEE 802.11 Association Request, Flags: .........
▼ IEEE 802.11 wireless LAN management frame
  ▶ Fixed parameters (4 bytes)
  ▼ Tagged parameters (142 bytes)
    ▶ Tag: SSID parameter set: XXXXXXXXXX
    ▶ Tag: Supported Rates 6(B), 9, 12(B), 18, 24(B),
    ▶ Tag: Power Capability Min: 249, Max :19
    ▶ Tag: Supported Channels
    ▶ Tag: RSN Information
    ▶ Tag: RM Enabled Capabilities (5 octets)
    ▶ Tag: HT Capabilities (802.11n D1.10)
    ▶ Tag: VHT Capabilities (IEEE Std 802.11ac/D3.1)
    ▶ Tag: Vendor Specific: Apple
    ▶ Tag: Vendor Specific: Broadcom
    ▶ Tag: Vendor Specific: Microsof: WMM/WME: Inform
```

Tag #0
Tag #1
Tag #33
Tag #36
Tag #48
Tag #70
Tag #45
Tag #191
Tag #221, Vendor OUI 00:17:f2, #10
Tag #221, Vendor OUI 00:10:18, #2
Tag #221, Vendor OUI 00:50:f2, #2

```
0,1,33,36,48,70,45,191,
221(0017f2,10),221(0010
18,2),221(0050f2,2)
```

Information Elements are Type-Length-Value tuples packed one after the other in the management frame. They are all optional, though in practice a few are universal because Wi-Fi can't really work without them. Each Wi-Fi standard has added more Information Elements. In 802.11b days there were very few. 802.11g added a few, 802.11n and ac added a bunch more, and so on.

In addition to the standard elements, there is a mechanism for vendors to define their own. Vendor extensions are type 221 with an identifier for the vendor called the Organizationally Unique Identifier or OUI, and finally by a subtype so the vendor can define multiple extensions. Because the Length provides enough information to skip over the IE, any Wi-Fi device can interoperate whether it understands the vendor extensions or not. It can skip over those it doesn't implement.

This is the Association Request from an iPhone 7+, as broken out by Wireshark. The Association Request includes the SSID the client wants to join, information about its supported rates and channels, about its power levels and radio management capabilities, plus three vendor extensions from Microsoft, Broadcom, and Apple.

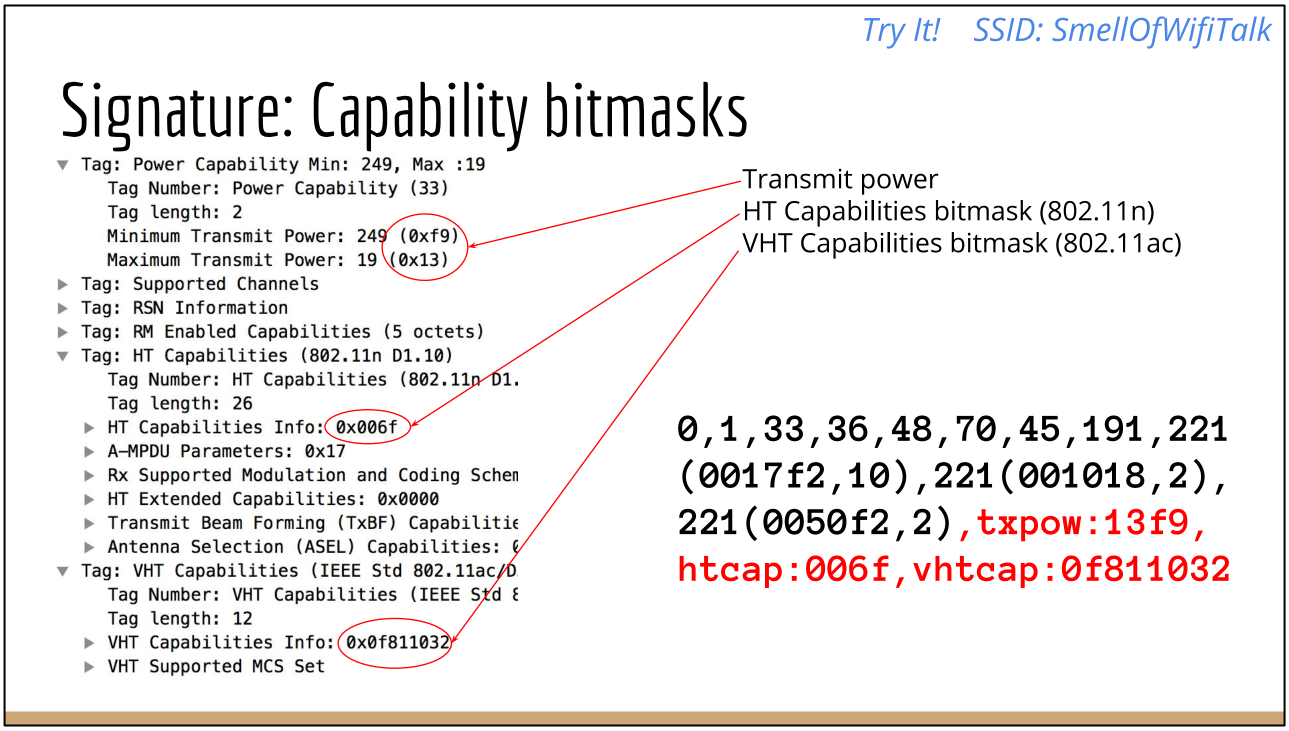
A few of the vendor extensions are extremely widespread. The Microsoft extension shown above is for prioritization, and it's widely present even on devices not running any kind of Windows OS. The Broadcom vendor extension is also very widespread, owing to how common Broadcom chipsets are. The Apple extension shown here was added in iOS 10.2 on all iOS devices.

The signature lists the tag numbers of the IEs present in the frame, in the order they appear, as a text string of decimal numbers. For vendor extensions it additionally includes the OUI of the vendor and the subtype. For this first part of the signature, we end up with the text string shown in red.

This part of the signature is most strongly influenced by the OS of the client device, where the client Wi-Fi stack is implemented.

It is next most strongly influenced by the Wi-Fi chipset, both in terms of the standards it supports and for any vendor extensions implemented by that vendor in their driver.

2:10



*Try It!* *SSID: SmellOfWifiTalk*

## Signature: Capability bitmasks

```
▼ Tag: Power Capability Min: 249, Max :19
    Tag Number: Power Capability (33)
    Tag length: 2
    Minimum Transmit Power: 249 (0xf9)
    Maximum Transmit Power: 19 (0x13)
▶ Tag: Supported Channels
▶ Tag: RSN Information
▶ Tag: RM Enabled Capabilities (5 octets)
▼ Tag: HT Capabilities (802.11n D1.10)
    Tag Number: HT Capabilities (802.11n D1.
    Tag length: 26
  ▶ HT Capabilities Info: 0x006f
  ▶ A-MPDU Parameters: 0x17
  ▶ Rx Supported Modulation and Coding Schen
  ▶ HT Extended Capabilities: 0x0000
  ▶ Transmit Beam Forming (TxBF) Capabilitie
  ▶ Antenna Selection (ASEL) Capabilities: 0
▼ Tag: VHT Capabilities (IEEE Std 802.11ac/D
    Tag Number: VHT Capabilities (IEEE Std 8
    Tag length: 12
  ▶ VHT Capabilities Info: 0x0f811032
  ▶ VHT Supported MCS Set
```

Transmit power
HT Capabilities bitmask (802.11n)
VHT Capabilities bitmask (802.11ac)

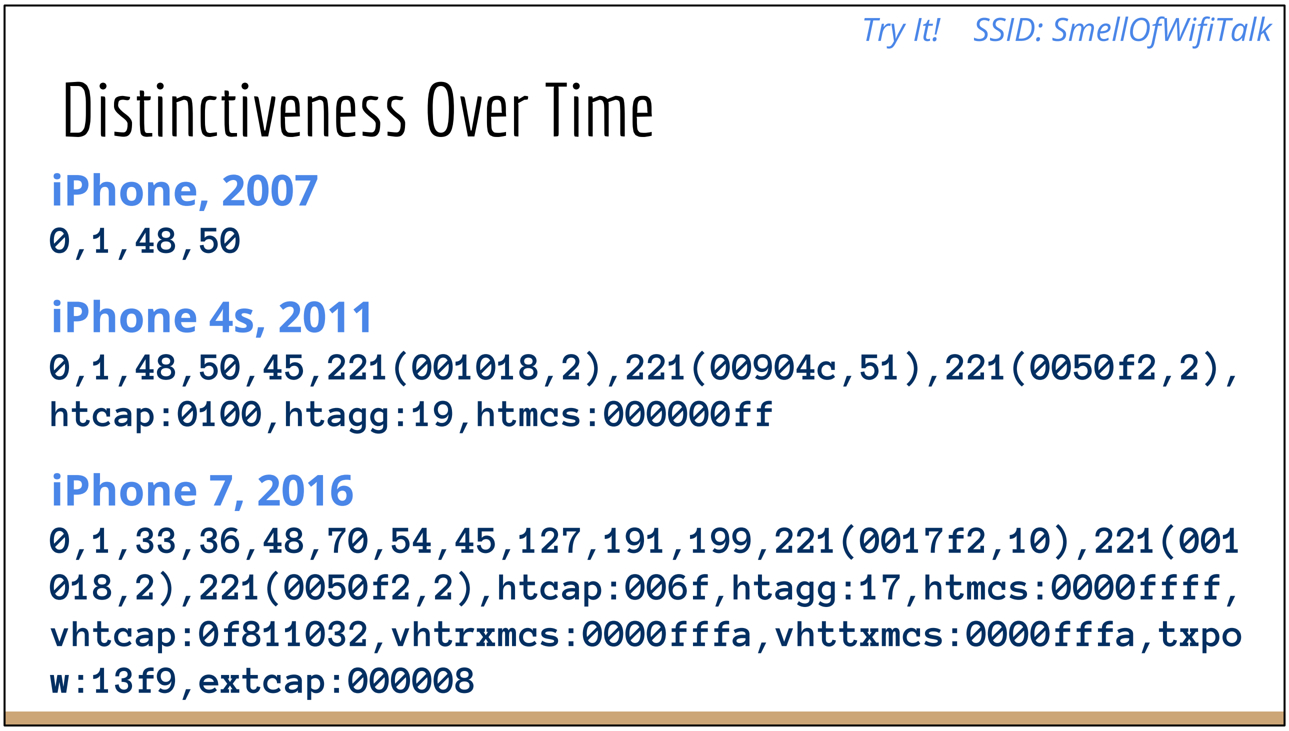
```
0,1,33,36,48,70,45,191,221
(0017f2,10),221(001018,2),
221(0050f2,2),txpow:13f9,
htcap:006f,vhtcap:0f811032
```

In addition to the tag numbers, a few of the Information Elements contain capability bitmasks or other information which is useful in identifying the device. For example:

- 802.11n defines 16 bits of optional capabilities, 802.11ac defines another 32 bits. This is most strongly influenced by the chipset, and the subset of the standard implemented by the ASIC.
- The TX Power IE depends strongly on the board design, in how the antennas are laid out. Two devices built by the same manufacturer, using the same software and the same Wi-Fi chipset will often have different TX Power values.
- The number of antennas is encoded in the dot11n and ac capabilities, and is also indicative of a board design.
- There is an Extended Capabilities bitmask containing even more optional elements. It is most strongly influenced by the driver and WPA supplicant software.

A number of the capability bitmasks are appended to the signature to further differentiate it, as shown in red here.

1:00



*Try It!* *SSID: SmellOfWifiTalk*

## Distinctiveness Over Time

**iPhone, 2007**

```
0,1,48,50
```

**iPhone 4s, 2011**

```
0,1,48,50,45,221(001018,2),221(00904c,51),221(0050f2,2),
htcap:0100,htagg:19,htmcs:000000ff
```

**iPhone 7, 2016**

```
0,1,33,36,48,70,54,45,127,191,199,221(0017f2,10),221(001
018,2),221(0050f2,2),htcap:006f,htagg:17,htmcs:0000ffff,
vhtcap:0f811032,vhtrxmcs:0000fffa,vhttxmcs:0000fffa,txpo
w:13f9,extcap:000008
```

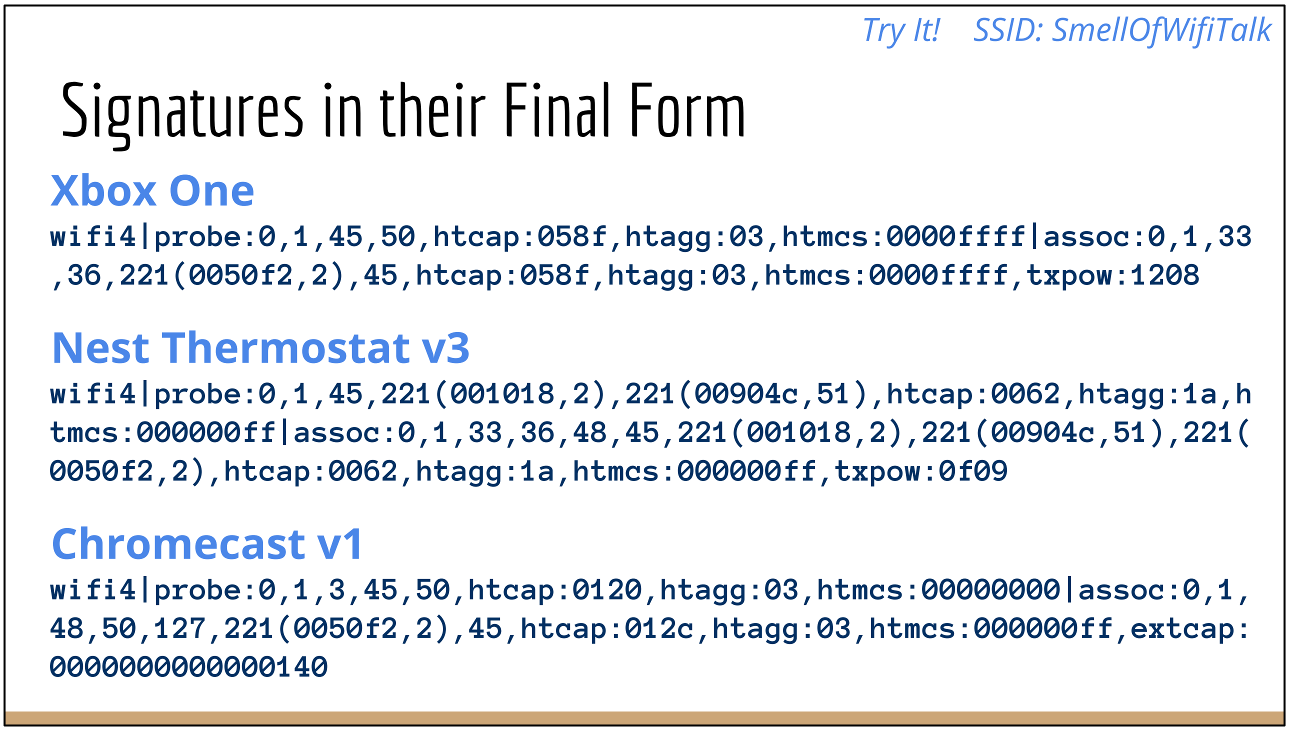
Looking at the signature as we've discussed it so far, it has become more distinctive over time. This shows the Associate Request portion of the signatures for three devices.

The first is from an original iPhone, which is a dot11g device. The Taxonomy mechanism really wouldn't have worked in that timeframe, there was almost no differentiation in the contents of MLME frames.

iPhone 4s is an dot11n device, introduced about 4 years later. It adds a number of options to its management frames.

iPhone 7 is from about 5 years after that, and is an dot11ac device. It added even more.

0:40



*Try It!*   *SSID: SmellOfWifiTalk*

## Signatures in their Final Form

### Xbox One

```
wifi4|probe:0,1,45,50,htcap:058f,htagg:03,htmcs:0000ffff|assoc:0,1,33
,36,221(0050f2,2),45,htcap:058f,htagg:03,htmcs:0000ffff,txpow:1208
```

### Nest Thermostat v3

```
wifi4|probe:0,1,45,221(001018,2),221(00904c,51),htcap:0062,htagg:1a,h
tmcs:000000ff|assoc:0,1,33,36,48,45,221(001018,2),221(00904c,51),221(
0050f2,2),htcap:0062,htagg:1a,htmcs:000000ff,txpow:0f09
```

### Chromecast v1

```
wifi4|probe:0,1,3,45,50,htcap:0120,htagg:03,htmcs:00000000|assoc:0,1,
48,50,127,221(0050f2,2),45,htcap:012c,htagg:03,htmcs:000000ff,extcap:
0000000000000140
```

The full signature contains the list of IEs and the various bitmasks from each of the Probe Request and the Associate Request, separated by a pipe. The whole thing is prefaced by “wifi4” as this is the fourth iteration on the signature format. Including the prefix allowed the wifi, wifi2, and wifi3 signatures to remain in the database while updating.

We shall speak no more of the earlier three.

0:25

*Try It!*    *SSID: SmellOfWifiTalk*

## Mobile Only!

Taxonomy identifies the Wi-Fi circuitry, device driver, and OS.

- Works for highly integrated devices: mobile and IOT.
- With a Wi-Fi card in a laptop… it identifies the card.

The Wi-Fi taxonomy signature is influenced by the client OS, by its Wi-Fi chipset, and its board layout. The current database of signatures identifies the most common Wi-Fi devices: overwhelmingly phones nowadays. We have signatures for most widely sold phones and tablet devices of the last few years, and a selection of other types of devices like:

- Media streaming devices from Google, Apple, Roku, Amazon, et cetera
- Internet of Things devices from Nest, Honeywell, Withings, and so forth

For larger devices like laptops and desktops which use a separate Wi-Fi card: the mechanism identifies the card. We had signatures from some laptop and desktop devices, but it was kindof ridiculous.

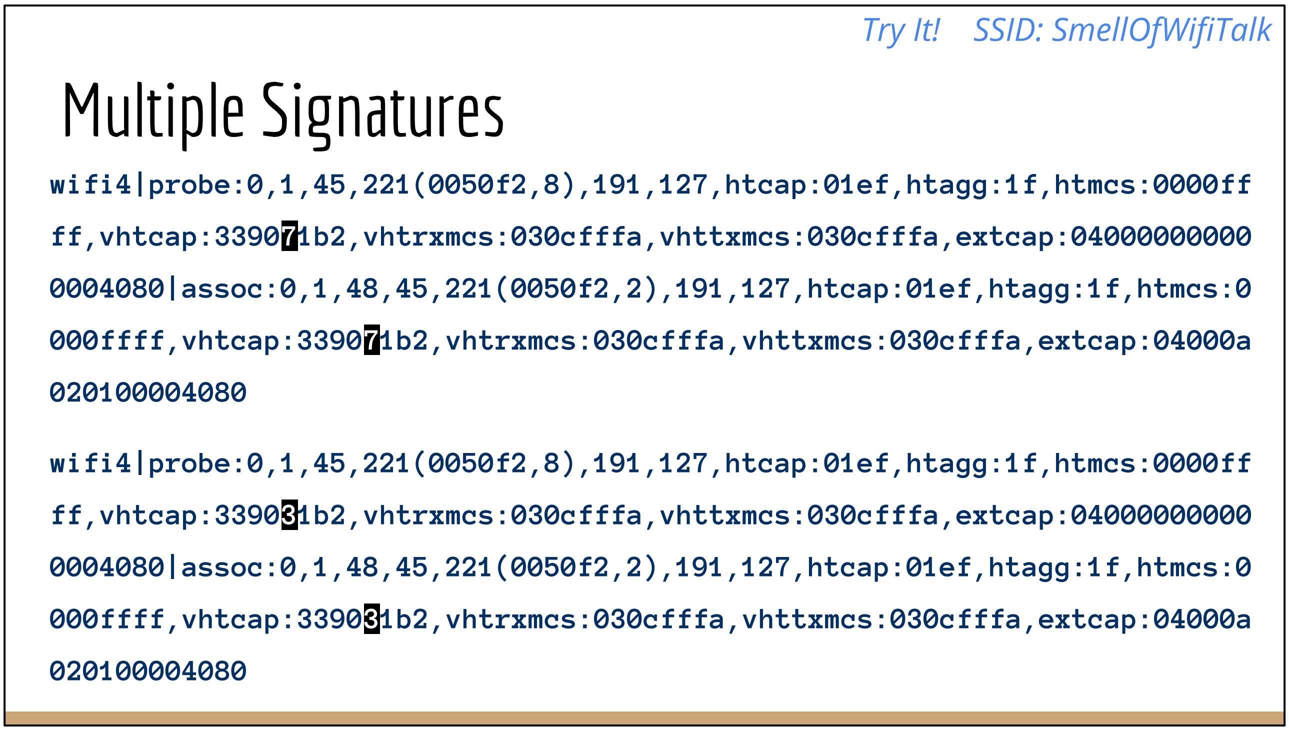
- One model of Apple Airport Extreme card could be a MacBook, or iMac, or Mac Pro. That's pretty much the entire generation of machines, we couldn't distinguish them using this mechanism.
- Intel Centrino chipsets as used in Windows laptops are even less distinctive. It could be almost anything.

So at this point, we don't even try and don't add signatures from laptops or desktops into the database. It tends to just result in confusion, and it isn't useful.

Additionally, there are a few classes of device which we choose not to gather signatures for:

- We only want to focus on common devices, things which a lot of people are likely to have. We use lists of top-selling electronics to target what we want to gather signatures for. If it is something unique or only in very low volume, we don't really want to add it to the database.
- We don't put in labels for things which might make someone uncomfortable if they see it in the UI from their router. That includes medical devices, devices of an adult nature, home incarceration monitoring, and so on.

1:50



Try It!    SSID: SmellOfWifiTalk

## Multiple Signatures

```
wifi4|probe:0,1,45,221(0050f2,8),191,127,htcap:01ef,htagg:1f,htmcs:0000ff
ff,vhtcap:339071b2,vhtrxmcs:030cfffa,vhttxmcs:030cfffa,extcap:04000000000
0004080|assoc:0,1,48,45,221(0050f2,2),191,127,htcap:01ef,htagg:1f,htmcs:0
000ffff,vhtcap:339071b2,vhtrxmcs:030cfffa,vhttxmcs:030cfffa,extcap:04000a
020100004080
```

```
wifi4|probe:0,1,45,221(0050f2,8),191,127,htcap:01ef,htagg:1f,htmcs:0000ff
ff,vhtcap:339031b2,vhtrxmcs:030cfffa,vhttxmcs:030cfffa,extcap:04000000000
0004080|assoc:0,1,48,45,221(0050f2,2),191,127,htcap:01ef,htagg:1f,htmcs:0
000ffff,vhtcap:339031b2,vhtrxmcs:030cfffa,vhttxmcs:030cfffa,extcap:04000a
020100004080
```

Many devices have been seen to emit more than one signature and so there is more than one entry for them in the database.

For devices which support both 2.4 and 5 GHz operation, the signatures are almost always distinct.

- there are Information Elements which are only defined for one or the other, like Extended Supported Rates on 2.4
- also the whole of do11ac is only defined for 5GHz.

So for devices which support both bands, we always capture signatures from each band.

However even in the same band, devices often have multiple signatures. They vary what they advertise based on local conditions like noise, or in what they saw from the AP.

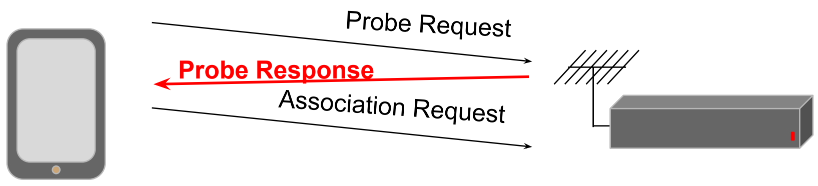
This example shows two signatures seen from a Google Pixel Phone. It varies its handling of beamforming, presumably based on the noise environment it sees.

0:50

*Try It!   SSID: SmellOfWifiTalk*

## Feedback loop with AP

```
wifi4|probe:0,1,45,
221(0050f2,8),191,1
27,htcap:016f,htagg
:1f,htmcs:000000ff,
vhtcap:33907132,vht
rxmcs:0186fffe,vhtt
xmcs:0186fffe,extca
p:04000000000000408
0|assoc:0,1,33,36,4
8,70,45,221(0050f2,
2),191,127,htcap:01
6f,...
```

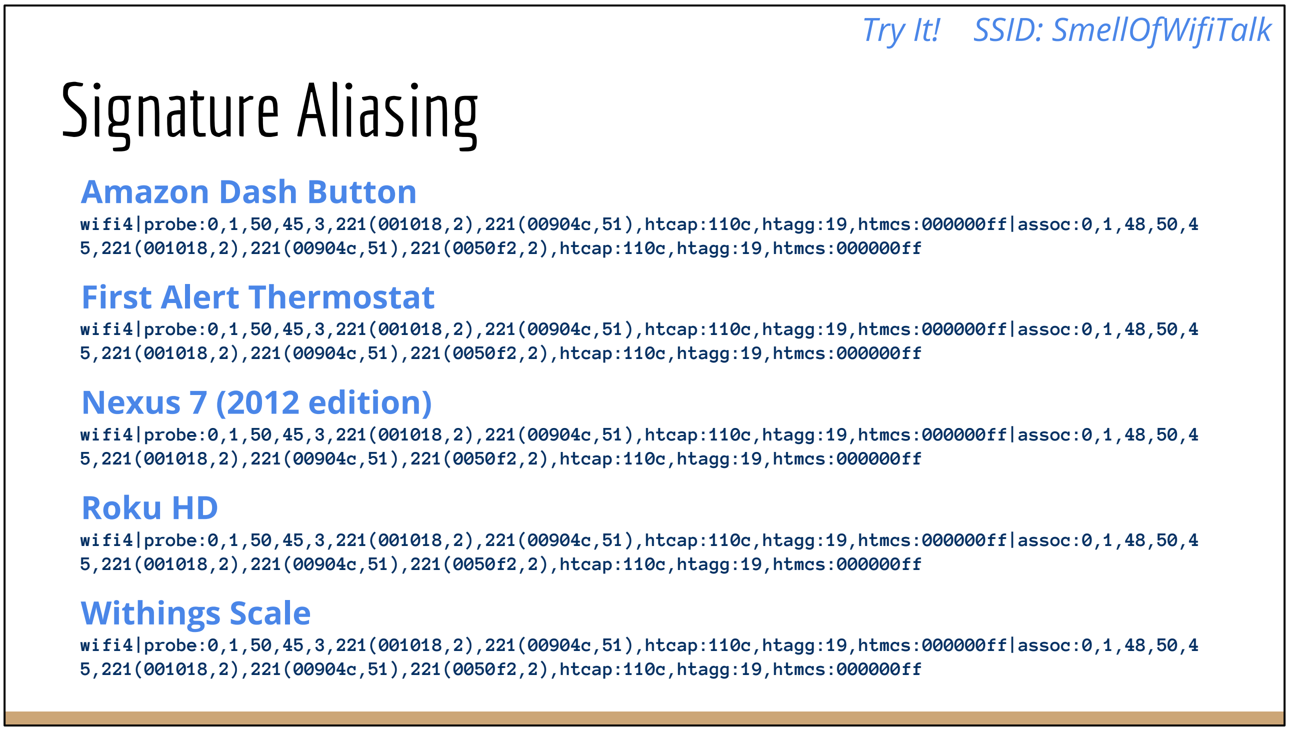


Clients can also behave differently depending on what the AP says in response to their Probe Request. For example:

- If the AP says it supports Radio Resource Management, most Apple and some Android devices will include a Spectrum Management IE in their Association Request (it’s #70).
- Another example: though dot11ac is only defined for 5 GHz, many vendors have a proprietary implementation handling 2.4GHz as well. Qualcomm includes the dot11ac information in the Probe Request, but will only include in the Association if it sees the right response from the AP.
  Broadcom also has a proprietary dot11ac implementation for 2.4GHz, but it works differently. Of course.

When capturing signatures for the database, we use three different APs to maximize the chances of capturing different signatures.

1:00



*Try It!* *SSID: SmellOfWifiTalk*

## Signature Aliasing

**Amazon Dash Button**

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff
```

**First Alert Thermostat**

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff
```

**Nexus 7 (2012 edition)**

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff
```

**Roku HD**

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff
```

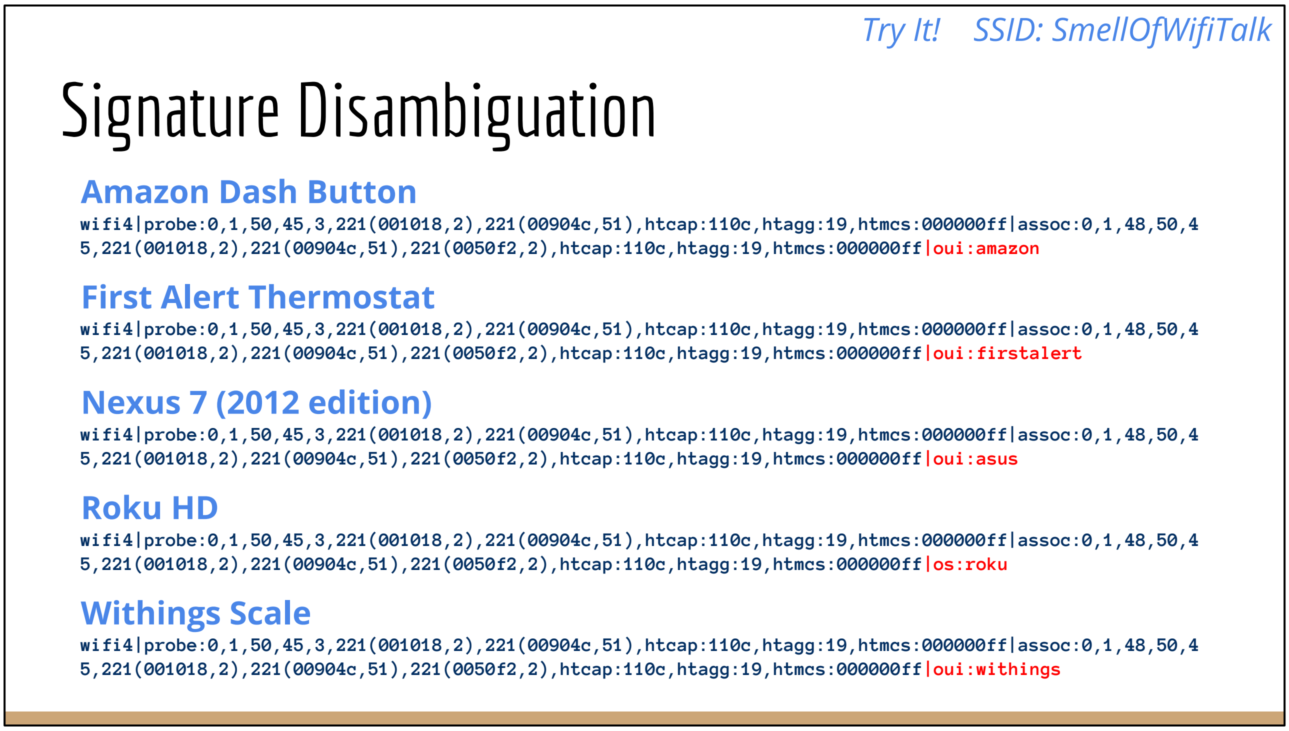
**Withings Scale**

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff
```

Nonetheless, sometimes we see the same signature from multiple devices.

These examples are all devices using the Broadcom 43362 chipset, running Linux, with the same driver, the same wpa_supplicant, and old enough that none of them provide a TX Power IE. The signatures are identical: an Amazon Dash Button, a First Alert Thermostat, a Nexus 7, Roku HD, and Withings Scale.

0:30



*Try It!* *SSID: SmellOfWifiTalk*

## Signature Disambiguation

### Amazon Dash Button

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff|oui:amazon
```

### First Alert Thermostat

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff|oui:firstalert
```

### Nexus 7 (2012 edition)

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff|oui:asus
```

### Roku HD

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff|os:roku
```

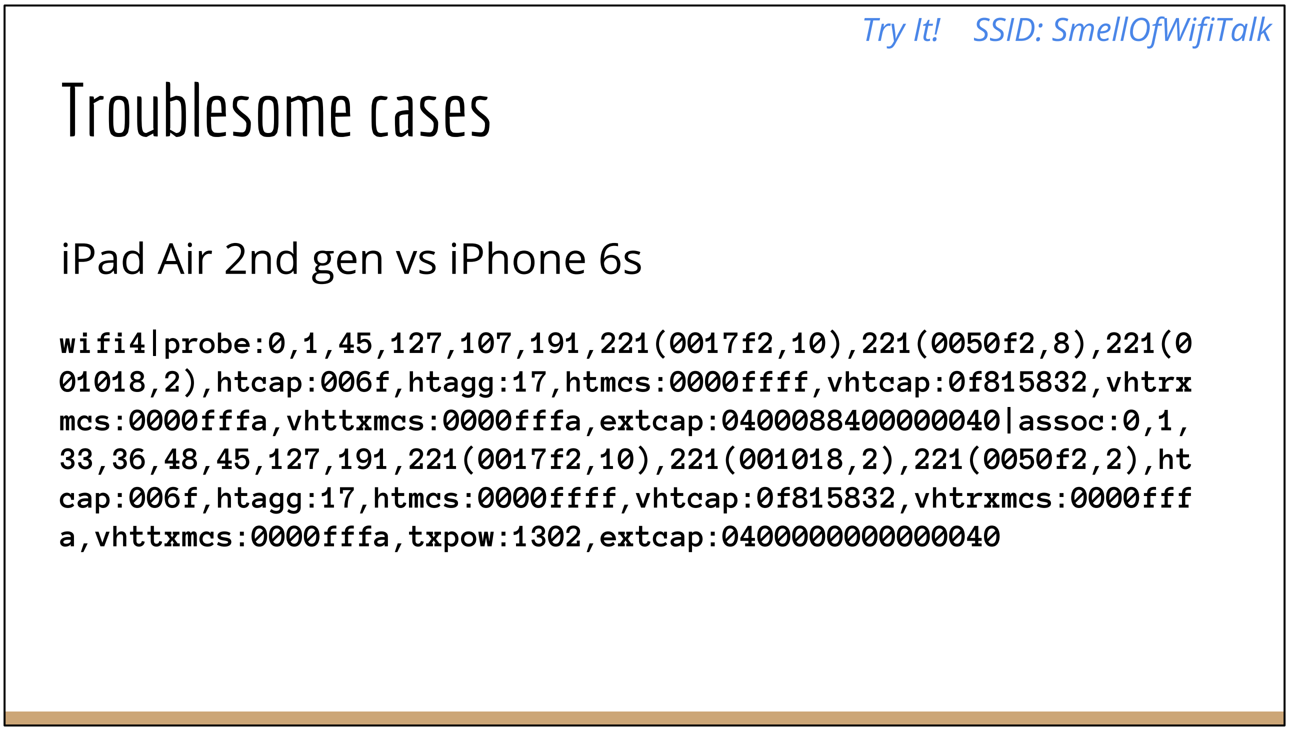
### Withings Scale

```
wifi4|probe:0,1,50,45,3,221(001018,2),221(00904c,51),htcap:110c,htagg:19,htmcs:000000ff|assoc:0,1,48,50,4
5,221(001018,2),221(00904c,51),221(0050f2,2),htcap:110c,htagg:19,htmcs:000000ff|oui:withings
```

In most cases like this we distinguish them using the top 24 bits of the MAC address, which is the Organizationally Unique Identifier. OUIs are assigned to a manufacturer. Adding the OUI as a qualifier can distinguish similar devices from different manufacturers which have the same signature.

We sometimes use information from DHCP: the options present in the request can identify the OS, as developed by Fingerbank (http://fingerbank.org) - and by the way, that earlier work on DHCP option signatures from Fingerbank inspired this mechanism for Wi-Fi. However using DHCP gets further and further from the Wi-Fi layer, so we try to be more sparing in using it. In particular, only the AP itself will be able to see the DHCP information after decryption, it cannot be used from a separate sniffer device on the Wi-Fi network.

0:45



*Try It!* *SSID: SmellOfWifiTalk*

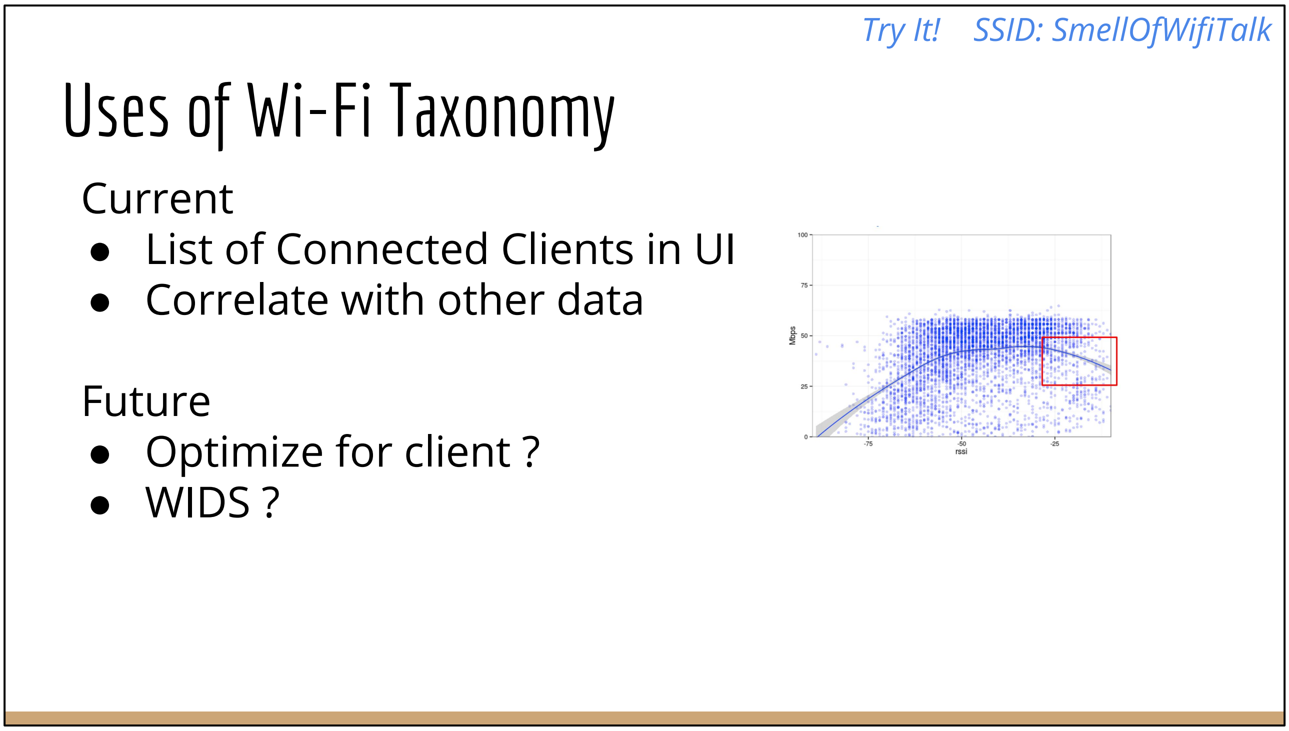
## Troublesome cases

iPad Air 2nd gen vs iPhone 6s

```
wifi4|probe:0,1,45,127,107,191,221(0017f2,10),221(0050f2,8),221(0
01018,2),htcap:006f,htagg:17,htmcs:0000ffff,vhtcap:0f815832,vhtrx
mcs:0000fffa,vhttxmcs:0000fffa,extcap:0400088400000040|assoc:0,1,
33,36,48,45,127,191,221(0017f2,10),221(001018,2),221(0050f2,2),ht
cap:006f,htagg:17,htmcs:0000ffff,vhtcap:0f815832,vhtrxmcs:0000fff
a,vhttxmcs:0000fffa,txpow:1302,extcap:0400000000000040
```

However there remain some cases which are still troublesome, mainly devices made by the same vendor, using the same software, the same Wi-Fi chipset, and at about the same time. For example, iPad Air 2nd generation and iPhone 6s have the same signature. We can try to use heuristics, like if the DHCP hostname contains iPad then maybe it’s an iPad. If nothing else though, we have to return both possibilities.

0:30



*Try It!* *SSID: SmellOfWifiTalk*

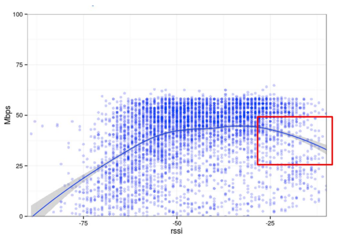
## Uses of Wi-Fi Taxonomy

Current

- List of Connected Clients in UI
- Correlate with other data

Future

- Optimize for client ?
- WIDS ?



This mechanism was originally developed as part of a Wi-Fi AP project. We intended to focus on identifying the Wi-Fi chipset the client was using. We thought if we could just know what chipset the client was using, we'd enable all kinds of very clever workarounds for bugs in that chipset and we would make Wi-Fi perfect… but it turns out that the kinds of bugs which can be easily worked around are mostly handled by the software in the client device. Who knew?

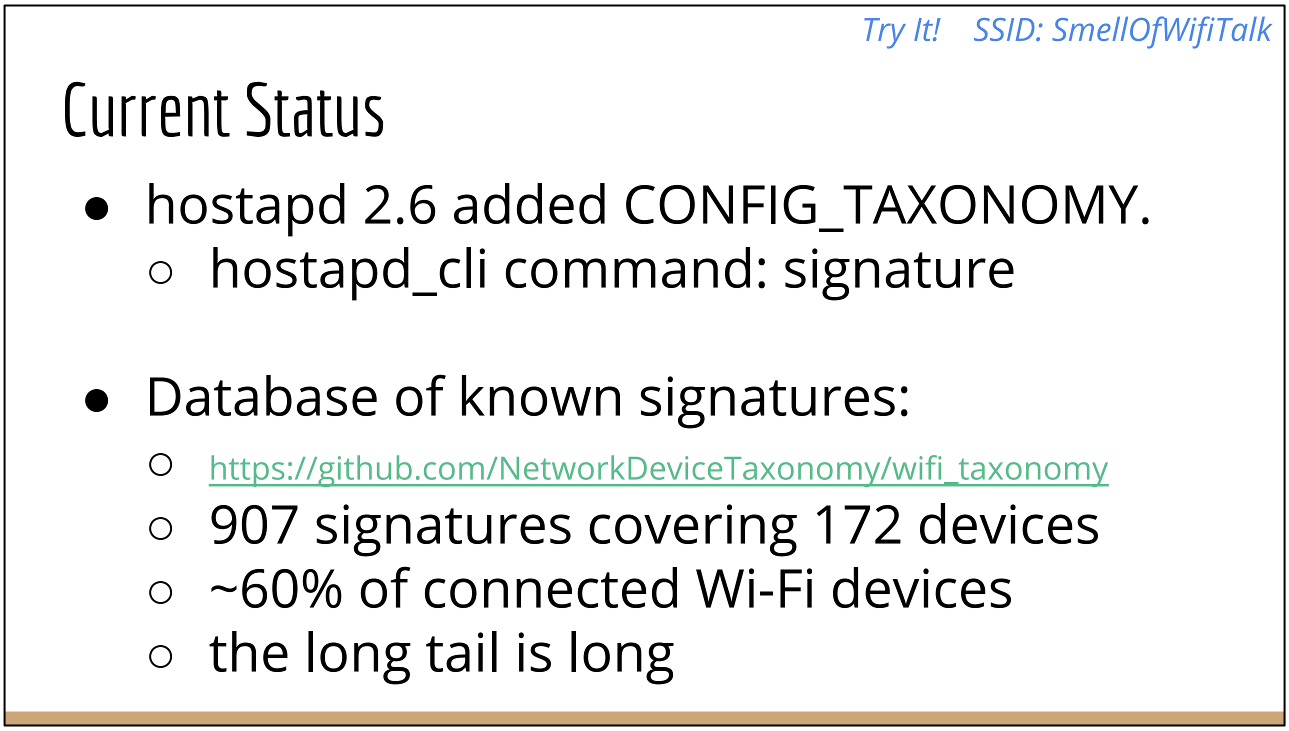
Instead, this information is currently used:

- in the UI where one can see a list of connected clients to also give an indication of what those clients are. If the client includes a useful hostname that is great, but if it doesn't include a hostname or it uses something like its serial number it is better to say what kind of device we think it is.
- We also use it to correlate with performance data to break it out by type of client device. My colleague Avery Pennarun gave a talk at Netdev 1.1, "Measuring wifi performance across all Google Fiber customers"
  This taxonomy mechanism was one of the topics. The graph on this page shows Wi-Fi throughput getting better and better as the client gets closer to the AP, until when it gets really close it drops sharply back down. That sort of behavior is only visible if the data can be broken out by the type of device. https://www.youtube.com/watch?v=yZcHbD84j5Y (the section about this mechanism starts at 16:09).

Possible uses in the future:

- Possibly of interest to this audience: Wireless Intrusion Detection Systems could make use of knowing more about the type of client connecting. Some of this stuff is rather difficult for malicious clients to spoof, for example pretending to be a Broadcom chipset if the attacking device actually uses Qualcomm or Marvell.
- We might also use it for optimizations based on the client, for example in packet reorder buffers. These keep retransmitted packets in order at the cost of introducing delay. Windows absolutely needs the packets to stay in order, it considers even minimal reordering as congestion and slows down. iOS and Linux are more tolerant of occasional reordering, and could get lower latency on average if we allowed occasional packets to arrive out of order.

2:00



*Try It!   SSID: SmellOfWifiTalk*

## Current Status

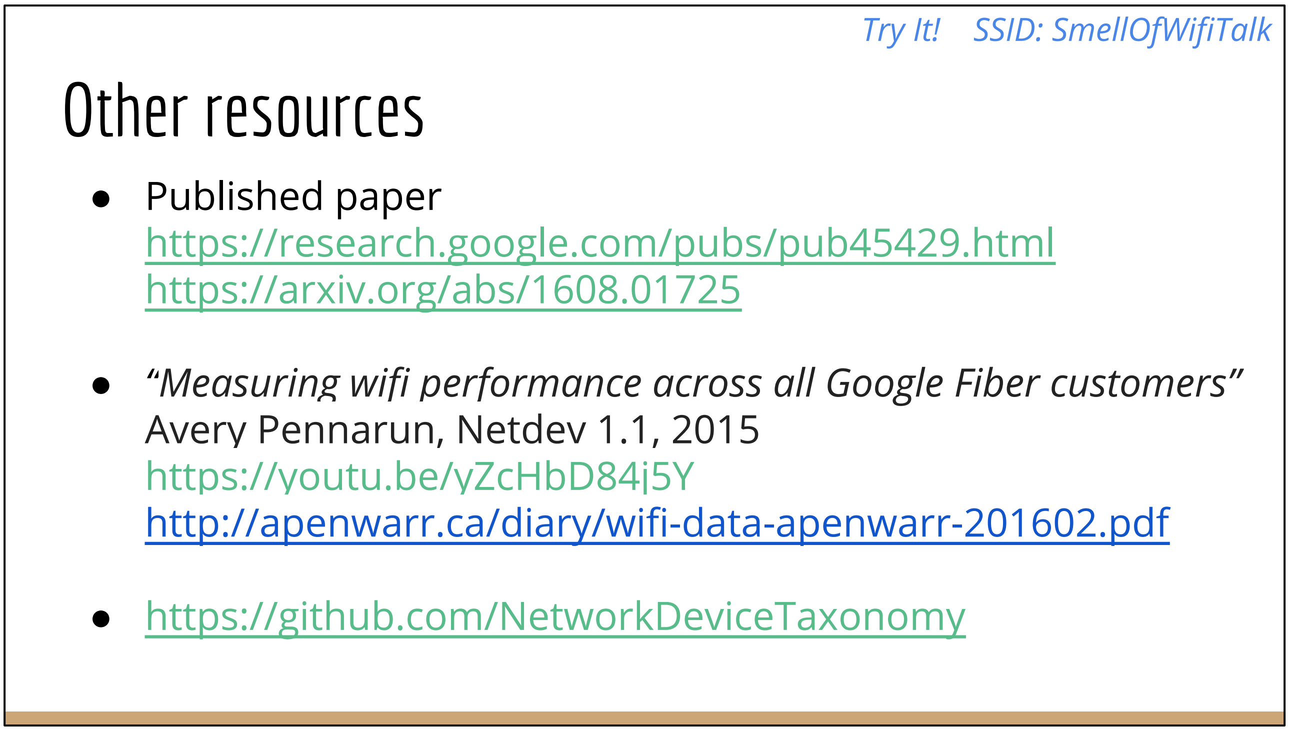
- hostapd 2.6 added CONFIG_TAXONOMY.
  - hostapd_cli command: signature
- Database of known signatures:
  - https://github.com/NetworkDeviceTaxonomy/wifi_taxonomy
  - 907 signatures covering 172 devices
  - ~60% of connected Wi-Fi devices
  - the long tail is long

The implementation to extract signatures for clients went into hostapd in August 2016, and is present in hostapd 2.6 and later.

The database of known signatures is released as open source code under an Apache license.
https://github.com/NetworkDeviceTaxonomy/wifi_taxonomy

It currently identifies about 60% of Wi-Fi clients across a broad swath of the market. The remaining 40% of devices are mostly laptops and desktops, with a long tail of other types of unidentified devices.

0:45



*Try It!    SSID: SmellOfWifiTalk*

## Other resources

- Published paper
  https://research.google.com/pubs/pub45429.html
  https://arxiv.org/abs/1608.01725

- *"Measuring wifi performance across all Google Fiber customers"*
  Avery Pennarun, Netdev 1.1, 2015
  https://youtu.be/yZcHbD84j5Y
  http://apenwarr.ca/diary/wifi-data-apenwarr-201602.pdf

- https://github.com/NetworkDeviceTaxonomy

We published a paper about the mechanism, which goes about a half level of detail deeper into the implementation.

Earlier I mentioned the talk at Netdev 1.1 by Avery Pennarun, another senior software engineer working on the project and co-developer of the mechanism. That talk described the overall environment where this mechanism was developed and how it was used in that environment.

The github site is intended to host other tools relating to client taxonomy and identification.

0:30

## The Way Forward

- Integration
- Better tools to gather signatures
- AP Taxonomy?

What comes next?

- So. Here is this thing. The signature mechanism is in hostapd, and the database of signatures is published as open source code, but it is only useful if integrated into other products and systems: Wi-Fi APs, or Wireless Intrusion Detection Systems, or other stuff.
- We need to develop better tools for gathering signatures. Right now it is pretty manually intensive.
- This talk has been all about how APs can identify clients, but running it in reverse would likely work as well. It could be run on a client to identify the type of AP it is connecting to. The list of IEs present in the AP's Beacon and Probe Response could be turned into a signature, and this would allow any other connectivity or performance checks to report more information about the AP.