

Wi-Fi Monitoring & Kismet

Mike Kershaw
@KismetWireless
Sharkfest 2019

Intro

- Wi-Fi sniffing has been around since the late 1990s
- Still something we need to do now...
- More and more “last-mile” is going to wireless
- More and more sensors, control networks, etc are going to wireless
- Offices are increasingly using Wi-Fi instead of running cable
- BYOD (Bring Your Own Device) is huge
- Plenty of security problems need monitoring

Get off my lawn

- Kismet is over 18 years old now
- I used to joke it was old enough to drive. Now it's old enough to buy cigarettes and vote.
- Undergone several significant rewrites over that period
- Most recent major rewrite in the last few years adds all new capabilities, user interfaces, etc
- More on this later though...

Why do we need something special?

- Why do we even need another tool just to monitor Wi-Fi
- There's already so many that monitor packets
- Maybe have heard of one or two
- Rhymes with "Tire Bark"
- I heard there's some sort of conference about it?

Wi-Fi is a unicorn

- Truly *shared medium*. Anywhere signal goes, it impacts something
- Not just shared media with your network, but shared with everyone near you
- Multiple networks overlap bandwidth and channel access
- *Isn't Ethernet*. Your OS might act like it is. It isn't.
- Remember the OSI model? You're suddenly really going to care about layer 1 and 2 more than you ever did before.
- Knowing a network is there is *not* knowing what's going on with the network
- Knowing what's impacting your network is *not* simple!

Discovering Wi-Fi networks

- Several techniques can be used to discover Wi-Fi
- Scanning mode: looks for advertising networks; can't see clients, but does a good job showing what access points are out there. This doesn't have anything to do with packet capture, just listing advertising SSIDs
- Access-point assisted: found in some enterprise APs, can list channel availability, number of clients, and so on, but not packets.
- But to *really* know what's going on we need to see the packets. This is Sharkfest, after all!

Getting the Wi-Fi packets

- You can open a generic Wi-Fi interface in Wireshark (or tcpdump, or whatever)
- You'll get packets
- But...

Lies, and screenshots of also lies

- ▶ Frame 31: 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits) on interf
- ▼ Ethernet II, Src: IntelCor_23:54:7e (a0:a4:c5:23:54:7e), Dst: Armorlin_9d:cc:00 (
 - ▶ Destination: Armorlin_9d:cc:00 (00:18:7d:9d:cc:00)
 - ▶ Source: IntelCor_23:54:7e (a0:a4:c5:23:54:7e)
 - Type: IPv4 (0x0800)
- ▶ Internet Protocol Version 4, Src: 172.27.4.208, Dst: 34.194.201.2
- ▶ Transmission Control Protocol, Src Port: 37516, Dst Port: 443, Seq: 1, Ack: 1, Le
- ▶ Transport Layer Security

802.3 / Ethernet II frames

- The Wi-Fi card tells the OS it's an Ethernet device because everything understands Ethernet
- *Data* packets on Wi-Fi can hold the same *content* as Ethernet, so it makes sense to present them to the OS that way
- But you'll *only* see data, and you'll *only* see data for *your* connection (and broadcast, I guess. Depending on your driver.)
- There's a lot more to Wi-Fi than just data!

Actually capturing Wi-Fi packets

- ▶ Frame 1: 332 bytes on wire (2656 bits), 332 bytes captured (2656 bits)
- ▶ Radiotap Header v0, Length 25
- ▶ 802.11 radio information
- ▼ IEEE 802.11 Beacon frame, Flags:C
 - Type/Subtype: Beacon frame (0x0008)
 - ▶ Frame Control Field: 0x8000
 - .000 0000 0000 0000 = Duration: 0 microseconds
 - Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
 - Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
 - Transmitter address: [REDACTED] (88:eb:22:a8)
 - Source address: [REDACTED] (88:eb:22:a8)
 - BSS Id: [REDACTED] (88:eb:22:a8)
 - 0000 = Fragment number: 0
 - 1111 1101 0110 = Sequence number: 4054
 - Frame check sequence: 0x3fe92434 [unverified]
 - [FCS Status: Unverified]
 - ▶ IEEE 802.11 wireless LAN

Getting to Wi-Fi

- For wired NICs there is promisc mode, which turns off the MAC address filter and returns all packets seen
- For wireless, you need a whole other mode of operation
- Enter... monitor mode
- The operating system doesn't understand raw 802.11, but Wireshark does!
- Monitor mode isn't part of the spec, but lots of drivers support it... more or less.

Getting monitor mode

- Now the fun part...
- We need an operating system that supports monitor mode
- We need a Wi-Fi card with firmware that supports monitor mode
- We need drivers that support monitor mode
- ...
- That's a lot of things we need!
- This means that the *specific chipset* matters; not just the brand and model. Manufacturers change the internal chipsets all the time without mentioning it. Awesome.
- We'll dive into per-OS specifics later on...

Monitor mode problems

- Monitor mode isn't really a defined mode
- It's not part of IEEE802.11 or Wi-Fi standards
- Requires drivers to work, firmware to work, requires knowledge of the card internals
- Unfortunate amount of variance even when it does work
- Common failure modes are reporting no packets, not reporting data packets, or reporting corrupt, bogus, or otherwise mangled packets
- Getting into monitor mode is easier now than 10 years ago, but still a number of ways it can fail
- Kismet tries to solve the edge cases automatically as much as possible

Bogus packets

- Wi-Fi packets have a checksum (FCS) to validate them
- You'd think this would solve the problems with bogus packets...
- Some drivers don't report checksums... just... at all...
- Some drivers report only packet validity with no checksum - and then do it wrong
- Some report a checksum that they calculate themselves, defeating the whole purpose

Sniffing 11B (and G, and A)

- “Traditional” Wi-Fi
- Simpler radio encodings
- One antenna transmitting - even if your AP has two, only one is transmitting
- If you’re in range, you’re probably going to see the packets
- Having some assurance you’ve seen every packet on the channel is pretty easy

Modern Wi-Fi (N, AC, AX)

- MIMO - Multiple In, Multiple Out
- Really means: Many antennas do things at the same time
- More bandwidth...
- But it also builds off reflections and multipath (bounced and out-of-phase signals that normally distort reception)
- Suddenly *physical location* becomes part of the signal
- You get bounces from walls, etc, and get great signal...
- A sniffer 10 feet away (or outside the building) might see *nothing* from your data packets

Beamforming and phased array

- Next-level Sci-Fi stuff
- Multiple omnidirectional antennas - antennas that send signal in all directions
- Triggering each antenna slightly out of phase with the other antennas lets you build a signal that *acts like it's directional*
- AP can “beam” a signal directly at a user, making that user get a *stronger* signal
- Great for Wi-Fi
- Terrible for sniffing!
- AP has a directional signal pointed at a user; if you're not in that direction, you may see no data at all

Newer cards, fewer cards, less drivers

- Of course, sniffing newer protocols requires a newer card!
- You might see the presence of a *network* (thanks to backwards compatibility)
- ... but there's no chance of seeing 11n data with an ABG card
- Or 11ac with an 11n card...
- Newer tech costs more
- Fewer companies make it
- Cheap cards not available
- There's a handful of 11ac cards that work...
- And as far as I know, a *single* 11ax card that can do monitor right now (the newest Intel, with the 5.2 kernel)

Options for capturing

- So what are the actual options for capturing?
- Depends on OS...

OSX

- Capture with the built-in Airport card works
- Nothing else does, at least in the OSS world
- In theory *possible* to do USB capture, but would require a lot of code which hasn't been written

Windows

- AirPCAP from CACE - but approaching EOL
- Some commercial products
- In theory USB is possible, but again, no code yet
- WSL cannot do USB. WSL2 *might*? It's not yet clear, but being based on Hyper-V it's unlikely.

Linux

- Best chance of it working
- Many chipsets work:
 - Intel
 - Atheros (most)
 - Mediatek (new kernels)
 - Raspberry Pi built-in (with kernel patches)
 - Often others
- Small / embedded systems can be very good for feeding packets to a full-fledged system running Windows or OSX
- Often available as LiveCD/LiveUSB

LiveCD (well, LiveUSB)

- Multiple live distributions of Linux meant to run off a CD/DVD, or more recently, USB
- Kali is the most common; also Pentoo and recently, Parrot
- Typically the simplest way to get going
 - May not have the latest versions of the tools you need
 - May not be simple to compile new tools in
- Many have a 'persistent storage' file on USB to let you save changes between reboots
- May require disabling secure boot & other security options in your BIOS

Virtual machines

- Generally discouraged because there are a *lot* of edge cases
- USB devices may work with USB passthrough
- I've had good luck with VMWare on Windows. Lots of problems reported with VMWare on OSX
- Hyper-V *cannot* do USB
- Often the community will immediately refuse to help if you're running in a VM
- *May* be possible to do PCI passthrough on Linux KVM, but at that point you're already well into making custom stuff

Remote capture

- Kismet supports remote capture over TCP
- Lets you take a cheap Linux system and use it to funnel packets to your 'real' system
- Any device that can capture packets, to any system capable of running the Kismet server
- Transparent - fully controllable as if it were a local source
- Raspberry Pi, OpenWRT routers, and others are good candidates for remote capture
- Good option for feeding packets to Windows/WSL, OSX based Kismet servers

When you don't need monitor

- You don't need monitor mode when:
- You're analyzing traffic from a program to a server using your laptop
- You're looking at broadcast/multicast stuff coming to your system
- You're looking at data packets leaving your system
- You're looking at data traffic network-wide tapping the wired side

When you DO need monitor

- You absolutely need monitor mode to:
- See Wi-Fi level non-data packets
- See access points
- See Wi-Fi level DoS or spoofing attacks
- Find hidden APs
- Attack WEP or WPA

So let's talk about Kismet

- Designed to do a handful of things
- Manage Wi-Fi cards
 - Deal with different driver quirks and get them into monitor mode
 - Keep the OS from breaking our config
 - Channel hopping
- Enumerate networks and devices and their relationships
- More “network” and “device” centric than packet centric
- Can also generate a packet feed for deep packet analysis with a tool like Wireshark
- Handle a bunch of different types of radios, including more than just Wi-Fi

Old Kismet

- Kismet has been around since 2001
- You're probably used to it looking like this, if you've used it before..

Kismet Sort View Windows

Name	BSSID	T C	Ch	Freq	Pkts	Size	Bcr%	Sig	Clnt	Manuf	Qty	Seen By
TRENDnet	00:14:D1:5F:97:12	A 0	1	2417	1	0B	---	---	1	TrendwareI	---	wlan0
QQF93	00:1F:90:F2:CD:C2	A W	1	2412	1	0B	---	---	1	ActiontecE	US	wlan0
landscapers	00:14:BF:07:2F:84	A N	6	2437	2	0B	10%	-86	1	Cisco-Link	---	wlan0
linksys_SES_45997	00:16:B6:1B:E4:FF	A 0	6	2447	2	0B	---	---	1	Cisco-Link	---	wlan0
linksys	00:1A:70:D9:BC:13	A N	6	2437	2	0B	---	---	1	Cisco-Link	---	wlan0
MPA41	00:1F:90:E6:E0:84	A W	11	2462	3	0B	---	---	1	ActiontecE	---	wlan0
TFS	00:09:5B:D7:9D:B2	A N	---	2462	4	0B	---	---	1	Netgear	---	wlan0
Autogroup Probe	00:13:E8:92:3F:CB	P N	---	----	5	0B	---	0	1	IntelCorpo	---	wlan0
meskas	00:18:01:F5:65:E1	A 0	11	2462	7	0B	10%	-87	1	ActiontecE	US	wlan0
6SI03	00:1F:90:FA:F4:C8	A W	---	2412	8	0B	---	---	1	ActiontecE	---	wlan0
Xu Chen	00:18:01:F9:70:F0	A N	6	2442	9	0B	0%	-75	1	ActiontecE	US	wlan0
7J4R0	00:1F:90:E6:04:F1	A W	11	2462	14	0B	---	-70	1	ActiontecE	---	wlan0
TK421	00:18:01:FE:68:77	A 0	6	2437	14	0B	---	-82	1	ActiontecE	---	wlan0
Elina-PC-Wireless	00:24:B2:0E:E6:E2	A 0	11	2462	14	0B	0%	-31	1	Netgear	---	wlan0
Pickles	00:1F:33:F3:C5:4A	A 0	2	2422	17	0B	---	---	1	Netgear	---	wlan0
38c8	00:16:CB:07:6D:77	A W	6	2447	38	0B	---	-76	1	MonHaiPrec	---	wlan0
MAC	Crypt	Freq	Pkts	Size	Manuf	DHCP	Host	DHCP	OS			
00:13:10:35:59:CB	0	2462	624	0B	Cisco-Link	---	---	---	---			
00:11:24:A4:6F:B3	6	2452	6	708B	AppleCompu	---	---	---	---			
00:13:10:35:59:C9	5	2452	5	1K	Cisco-Link	---	---	---	---			
00:17:AB:3D:25:98	4	2452	4	626B	Nintendo	---	---	---	---			
00:13:E8:92:3F:CB	8	----	8	1K	IntelCorpo	---	---	---	---			

DRD1812

Networks
17

Packets
787

Pkt/Sec
10

Elapsed
00:01.05

wlan0
9

No GPS info (GPS not connected)

INFO: Detected new managed network "landscapers", BSSID 00:14:BF:07:2F:84, encryption no, channel 6, 54.00 mbit

ERROR: No update from GPSD in 15 seconds or more, attempting to reconnect

ERROR: Could not connect to the spectools server localhost:30569

INFO: Detected new managed network "QQF93", BSSID 00:1F:90:F2:CD:C2, encryption yes, channel 1, 54.00 mbit

ERROR: No update from GPSD in 15 seconds or more, attempting to reconnect

After a significant rewrite...

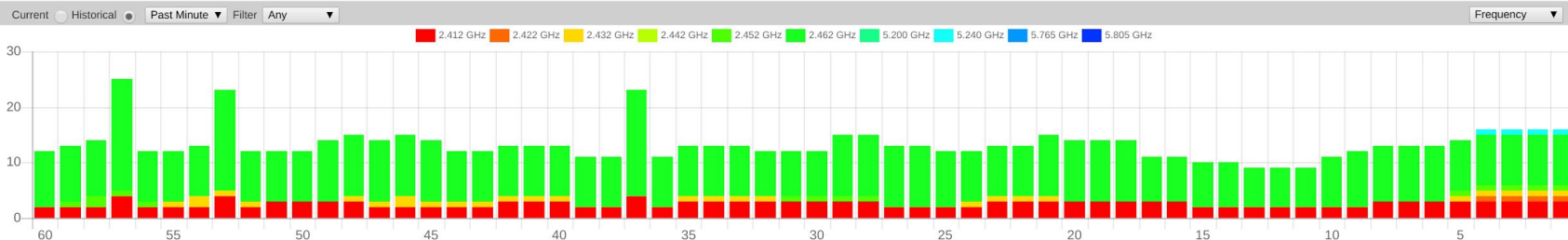
- Kismet now has a new web-based UI
- Records queryable over HTTP and exportable over JSON
- New logging system w/ single-file logging
- Live pcap feeds over HTTP
- Seamless remote capture

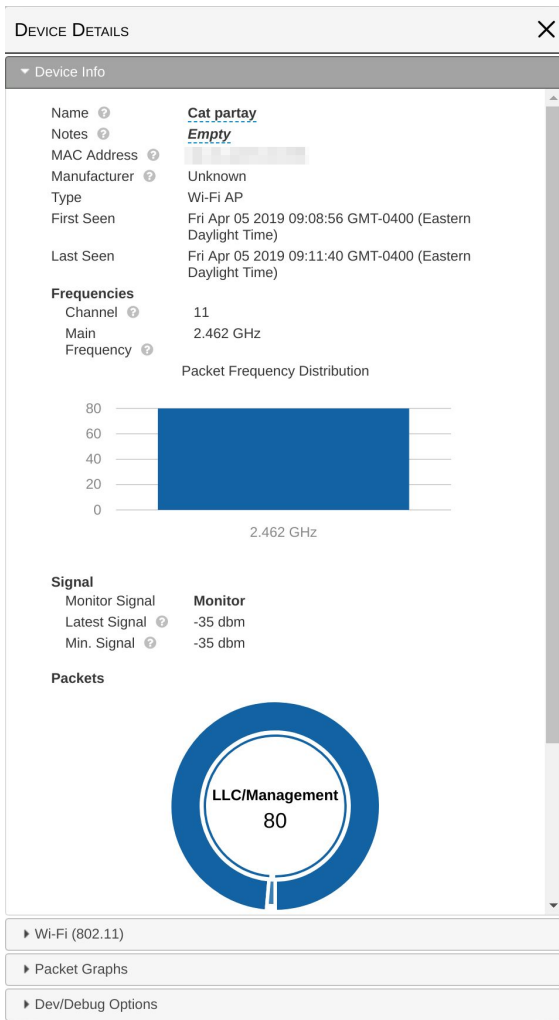
Search:

Name	Type	Phy	Crypto	Signal	Channel	Last Seen	Data	Packets	Clients	BSSID
UESC-N	Wi-Fi AP	IEEE802.11	WPA-CCMP	-35	11	Apr 05 2019 09:09:59	1.43 KB		5	
	Wi-Fi AP	IEEE802.11	WPA-CCMP	-84	11	Apr 05 2019 09:09:32	0 B		0	
	Wi-Fi AP	IEEE802.11	WPA-TKIP	-35	1	Apr 05 2019 09:10:00	0 B		0	
	Wi-Fi AP	IEEE802.11	WPA-CCMP	-64	6	Apr 05 2019 09:09:49	0 B		0	
	Wi-Fi AP	IEEE802.11	WPA-CCMP	-34	11	Apr 05 2019 09:09:59	0 B		0	
	Wi-Fi AP	IEEE802.11	Open	-49	1	Apr 05 2019 09:10:00	0 B		0	
Cat partay	Wi-Fi AP	IEEE802.11	WPA-CCMP	-35	11	Apr 05 2019 09:09:59	0 B		0	
	Wi-Fi AP	IEEE802.11	WPA-CCMP	-74	11	Apr 05 2019 09:09:59	0 B		0	
UESC	Wi-Fi AP	IEEE802.11	WPA-CCMP	-38	11	Apr 05 2019 09:09:59	22.95 KB		4	
UESC-N	Wi-Fi AP	IEEE802.11	WPA-CCMP	-74	11	Apr 05 2019 09:09:59	0 B		0	
Knappster	Wi-Fi AP	IEEE802.11	WPA-CCMP	-81	1	Apr 05 2019 09:09:56	0 B		0	
UESC	Wi-Fi AP	IEEE802.11	WPA-CCMP	-75	11	Apr 05 2019 09:09:59	0 B		0	
	Wi-Fi Bridged	IEEE802.11	n/a	-35	11	Apr 05 2019 09:09:12	740 B		0	
	Wi-Fi Bridged	IEEE802.11	n/a	-38	11	Apr 05 2019 09:09:59	22.96 KB		0	

Showing 1 to 7 of 20 entries

Messages Channels





Signal

Monitor Signal

Monitor

Latest Signal ⓘ

Min. Signal ⓘ

-35 dbm

-35 dbm

Packets

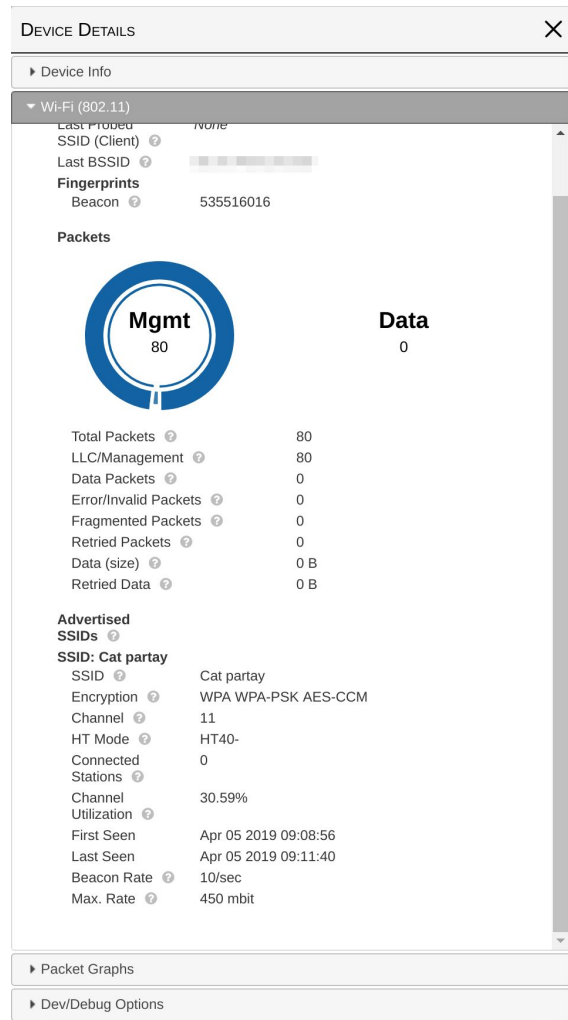
LLC/Management

80

► Wi-Fi (802.11)

► Packet Graphs

► Dev/Debug Options



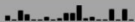
► Packet Graphs

► Dev/Debug Options

▶ btud0



▼ alfa0



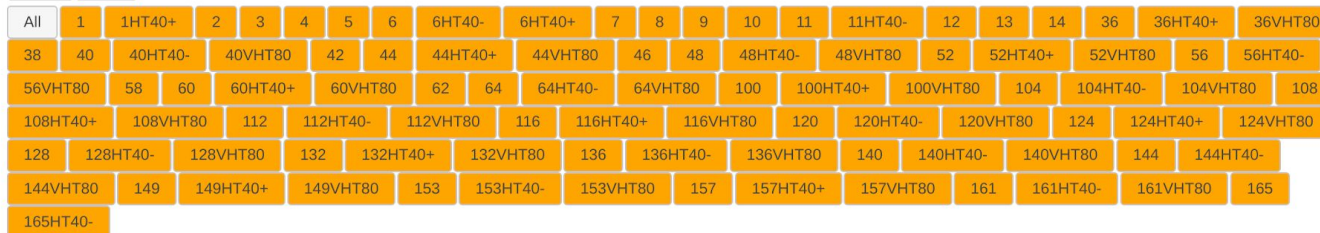
Interface wx00c0ca906a90
Hardware 8812au
UUID 5FE308BD-0000-0000-0000-00C0CA906A90
Packets 11296910
Retry on Error Kismet will try to re-open this source if an error occurs
Active

Paused Running

Channel Options

Lock Hop (Hopping at 5/second)

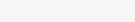
Channels



▶ rtl433



▶ intel



▶ tetra02-wlan1



▶ tetra01-wlan0



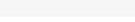
▶ tetra01-wlan1



▶ tetra02-wlan0



▶ rtl433 mediawhore



Kismet vs Wireshark

- Kismet and Wireshark do different things, and *absolutely* you'll want to use both for Wi-Fi
- Wireshark is packet oriented
- Kismet is device oriented

Traditional packet capture

- Plug into the network you're capturing from
- *Or*, enable a span port or tap on the network you're capturing from
- Look at the packets.
- Maybe you need to switch VLANs, at worst


Wi-Fi packet capture

- If you're just trying to monitor a single known user, tap into the wired side...
- If you only need to care about *data flow* not *wireless behavior*, tap into the wired side
- Otherwise, no single source of data
- What channel is the AP the user is connected to on?
- Is it user-to-user traffic? Does it even leave the AP?
- Is there another AP right next to you that's flooding the airwaves with its own traffic?
- Did someone bring an AP from home in and plug it into the network?
- Is someone sending non-data packets to try to disrupt your network?

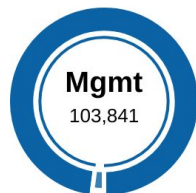
What Kismet does with packets

- Kismet looks at the contents of the packets and decodes them (like Wireshark)
- But then Kismet sorts them into access points, devices, and so on
- Tracks interactions between devices, such as joining a network, leaving, advertising a new network, changing encryption, etc
- Can include geolocation (GPS) data with packet capture
- Massages different packet types into one coherent record
- Offers insight and the ability to drill down into network and device details
- Offers filtered packet streams to feed into Wireshark!

How Kismet sees a network

Last Beacons
SSID (AP) ? UESC
Last Probed
SSID (Client) ? None
Last BSSID ? 
Fingerprints
Beacon ? 2428581901

Packets



Total Packets ?	104110
LLC/Management ?	103841
Data Packets ?	269
Error/Invalid Packets ?	0
Fragmented Packets ?	0
Retried Packets ?	38
Data (size) ?	36.78 KB
Retried Data ?	0 B

WPA Key Exchange ?

Handshake
Packets 7
Handshake
PCAP  Download Pcap File

Advertised

SSIDs ?

SSID: UESC

SSID ?	UESC
Encryption ?	WPA2 WPA2-PSK AES-CCM
MFP ?	Unavailable
Channel ?	1
HT Mode ?	HT20
Connected Stations ?	0
Channel Utilization ?	13.73%
First Seen	May 31 2019 15:22:47
Last Seen	Jun 01 2019 14:09:20
Beacon Rate ?	10/sec
Max. Rate ?	216.7 mbit

Associated

Clients ?

Client 10:98:C3:9E:BF:54

Client Info	View Client Details
Name	 4
Type	Wi-Fi Client
Manufacturer	Murata Manufacturing Co. Ltd
First Connected	Jun 01 2019 10:05:08
Last Connected	Jun 01 2019 10:05:08
Data	0 B
Retried Data	0 B

Data over time

- Data seen over time - total number of packets from each device
- Decoding beacons - Wi-Fi beacons contain a *ton* of information about the network, and can contain info about the channel, number of clients, and more
- Collecting handshakes - tools like Aircrack can use the handshake to derive the network password
- Client behavior over time when they join and leave different APs
- Trend-based security alerts when attributes change

How Wireshark sees the network

No.	Time	Source	Destination	Protocol	Length	Info
10.000000		Ubiquiti	(... Ubiquiti	802.11	70	802.11 Block Ack, Flags=.....C
20.000092		Ubiquiti	(... Ubiquiti	802.11	70	802.11 Block Ack, Flags=.....C
30.000074		Ubiquiti	(... Ubiquiti	802.11	70	802.11 Block Ack, Flags=.....C
40.000108		9a:8a:	Broadcas	802.11	262	Beacon frame, SN=2599, FN=0, Flags=.....C, BI=100, SSID=Wildcard (Broadcast)
50.000115		Ubiquiti	Broadcas	802.11	261	Beacon frame, SN=3059, FN=0, Flags=.....C, BI=100, SSID=UESC
60.000179		9a:8a:	Broadcas	802.11	283	Beacon frame, SN=409, FN=0, Flags=.....C, BI=100, SSID=Wildcard (Broadcast)
70.000190		8a:8a:	Broadcas	802.11	283	Beacon frame, SN=662, FN=0, Flags=.....C, BI=100, SSID=Wildcard (Broadcast)
80.000151		7a:8a:	Broadcas	802.11	263	Beacon frame, SN=2961, FN=0, Flags=.....C, BI=100, SSID=UESC-N
90.000205			NestLabs	802.11	52	Acknowledgement, Flags=.....C
100.000201		NestLabs	Ubiquiti	802.11	1547	QoS Data, SN=3906, FN=0, Flags=.p....TC
110.000249			NestLabs	802.11	52	Acknowledgement, Flags=.....C
120.000144		Ubiquiti	Broadcas	802.11	261	Beacon frame, SN=3060, FN=0, Flags=.....C, BI=100, SSID=UESC
130.000251		NestLabs	Ubiquiti	802.11	1547	QoS Data, SN=3908, FN=0, Flags=.p....TC
140.000778		NestLabs	Ubiquiti	802.11	1547	QoS Data, SN=3909, FN=0, Flags=.p..R..TC
150.000162		Sercon	Broadcas	802.11	319	Beacon frame, SN=3671, FN=0, Flags=.....C, BI=100, SSID=mios_45035085
160.000207		Ubiquiti	Broadcas	802.11	282	Beacon frame, SN=2176, FN=0, Flags=.....C, BI=100, SSID=UESC
170.000254			NestLabs	802.11	52	Acknowledgement, Flags=.....C
180.000788		NestLabs	Ubiquiti	802.11	1546	QoS Data, SN=3910, FN=0, Flags=.p....TC
190.001033		Ubiquiti	(... NestLabs	802.11	70	802.11 Block Ack, Flags=.....C
200.000491		NestLabs	Ubiquiti	802.11	1547	QoS Data, SN=3909, FN=0, Flags=.p....TC
210.000246		NestLabs	Ubiquiti	802.11	1547	QoS Data, SN=3907, FN=0, Flags=.p....TC
220.000171		8a:8a:	Broadcas	802.11	262	Beacon frame, SN=2701, FN=0, Flags=.....C, BI=100, SSID=Wildcard (Broadcast)
230.001028		NestLabs	Ubiquiti	802.11	1546	QoS Data, SN=3911, FN=0, Flags=.p..R..TC
240.001246		NestLabs	Ubiquiti	802.11	1546	QoS Data, SN=3914, FN=0, Flags=.p....TC
250.000785			NestLabs	802.11	52	Acknowledgement, Flags=.....C
260.001227		NestLabs	Ubiquiti	802.11	1558	QoS Data, SN=3913, FN=0, Flags=.p....TC
270.001229		Ubiquiti	(... NestLabs	802.11	70	802.11 Block Ack, Flags=.....C
280.000806		Ubiquiti	(... NestLabs	802.11	70	802.11 Block Ack, Flags=.....C

Packet-oriented

- All the same info, but presented very differently
- Not obvious immediately what clients are associated to the network
- Being packet-oriented, Wireshark is excellent at digging into specifics within individual packets, which Kismet isn't as good for

Wireless packet types

- Ethernet has one type of packet - a packet with data in it
- Wi-Fi has 3 main types of packets
- And something like 45 sub-types spread between them

Management packets

- Define the network (beacons)
- Control access to the network (probe requests and probe responses)
- Typically contain nested lists of data which contain all the options (SSID, encryption, nearby-radio reports, anything the standard or some company thinks need to get shoved in there)

Control packets

- Extremely short packets, often with only one MAC address
- Used for collision detection and avoidance
- Used to reserve the channel for transmit
- Used to confirm data received correctly

Data packets

- Most like an Ethernet packet
- Holds the data contents - snap header, etc
- May be encrypted with WPA, etc

You need all 3

- You need all 3 to know what's going on
- Without capturing management packets you can't know network capabilities, SSIDs, etc
- Without management you can't see most denial of service attacks
- Control packets control retransmits, and can reveal hidden nodes
- Data packets contain the network payloads
- You'll see all of them in Wireshark when looking at pcaps

Channel hopping

“You can see all of the channels, some of the time, or some of the channels, all of the time, but you can’t see all the channels all the time.”

-- PT Barnum

“Don’t trust every quote you read on the Internet”

-- Abraham Lincoln

Why channels matter

- Wi-Fi breaks the available frequency ranges up into channels
- Some channels overlap, some don't
- On 2.4GHz, channels are 22MHz (or 20MHz) wide, but only 5MHz apart
- Sniffing on one channel gets you packets from adjacent channels
- Typically 1, 6, and 11 are the only channels that are considered non-overlapping on 2.4GHz. It gets far worse with 802.11n and 40Mhz channels!
- On 5GHz there are many more, non-overlapping channels
- There's also 40MHz channels for 802.11n
- There's also 80MHz channels for 802.11ac
- There's also 160MHz channels for 802.11ac

Channel challenges

- A radio can only tune to a single channel - they're designed to exclude other channels as much as possible
- A radio can usually only tune to a single channel mode - 20MHz, 40MHz, 80MHz; it may not see data on other modes
- To see what's happening across all channels, we need to keep changing the channel
- It's a trade-off: You can try to see everything happening on one channel, or you can see a little of what is happening on many channels
- Like trying to listen to all the FM radio channels, you can listen to one, or you can seek through them all and hear 5 seconds at a time
- More channels = less time per channel

Channels, channels, everywhere

- 802.11b: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
 - 802.11g: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
-
- 11 channels (or 14 in some parts of the world)
 - Channels are 22 (or 20) MHz wide and 5MHz apart, significant overlap
 - You want to cover all channels, but will often see traffic from 3 channels away

802.11a

- 36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144, 149, 153, 157, 161, 165, 169, 173, 177, 181
- 38 more channels, combined with the 11 at 2.4GHz
- Still 20MHz wide, but now further apart - notice the gaps?
- Better for signal, but worse for sniffing; channel overlap doesn't really help us.

802.11n

- 1, 1HT40+, 2, 3, 4, 5, 6, 6HT40-, 6HT40+, 7, 8, 9, 10, 11, 11HT40-, 36, 36HT40+, 40, 40HT40-, 44, 44HT40+, 48, 48HT40-, 52, 52HT40+, 56, 56HT40-, 60, 60HT40+, 64, 64HT40-, 100, 100HT40+, 104, 104HT40-, 108, 108HT40+, 112, 112HT40-, 116, 116HT40+, 120, 120HT40-, 124, 124HT40+, 128, 128HT40-, 132, 132HT40+, 136, 136HT40-, 140, 140HT40-, 149, 149HT40+, 153, 153HT40-, 157, 157HT40+, 161, 161HT40-, 165, 165HT40-

How did we get 64 channels?

- Added 40MHz wide channels
- Above and below some channels (HT40+ and HT40-)
- Now we need to look at channel 1... but also 1HT40+. And 6, but also 6HT40-. And 6HT40+.

802.11ac

- Wait for it...

91 channels

1, 1HT40+, 2, 3, 4, 5, 6, 6HT40-, 6HT40+, 7, 8, 9, 10, 11, 11HT40-, 12, 13, 36, 36HT40+, 36VHT80, 40, 40HT40-, 40VHT80, 44, 44HT40+, 44VHT80, 48, 48HT40-, 48VHT80, 52, 52HT40+, 52VHT80, 56, 56HT40-, 56VHT80, 60, 60HT40+, 60VHT80, 64, 64HT40-, 64VHT80, 100, 100HT40+, 100VHT80, 104, 104HT40-, 104VHT80, 108, 108HT40+, 108VHT80, 112, 112HT40-, 112VHT80, 116, 116HT40+, 116VHT80, 120, 120HT40-, 120VHT80, 124, 124HT40+, 124VHT80, 128, 128HT40-, 128VHT80, 132, 132HT40+, 132VHT80, 136, 136HT40-, 136VHT80, 140, 140HT40-, 140VHT80, 144, 144HT40-, 144VHT80, 149, 149HT40+, 149VHT80, 153, 153HT40-, 153VHT80, 157, 157HT40+, 157VHT80, 161, 161HT40-, 161VHT80, 165, 165HT40-

80MHz channels

- Now we have 80MHz wide channels, plus 40MHz, plus 20MHz.
- 80MHz channels can have many 'control' channels with the same data channel... but we still need to look at each individually.
- 36, 36HT40+, 36VHT80, 40, 40HT40-, 40VHT80, 44, 44HT40+, 44VHT80, 48, 48HT40-, 48VHT80, 52, 52HT40+, 52VHT80

Just gets worse

- 802.11AC Wave2 adds 160MHz channels and 80+80 (dual 80MHz not next to each other)
- 802.11AX adds more wide channels
- Some devices use custom 5MHz and 10MHz

More & faster

- So we have to change channel, quickly, to see what's going on
 - We can do two things:
1. Hop faster; but at some point it doesn't make sense to channel hop quicker. Kismet defaults to 5 channels a second, and that can bog down some radios and operating systems
 2. Add more radios. This adds more power requirements, more cost, and more antennas



Kismet & multiple cards

- Kismet can run just about as many Wi-Fi cards as your system can support
- ... And even more over the network, using embedded sensors
- Practical limits come into play; After 5 or 10 cards the kernel can start having a Bad Time, and an even worse time depending on the card type
- Pushed network-attached cards over 50 without much trouble
- Still needs more RAM and processor power on the server to handle the packet loads

Smarter hopping

- Kismet tries to maximize channel hopping by two main ways:
 1. Scrambling the channels. Channels are shuffled so that Kismet won't hop to two overlapping channels in sequence
 2. Offsetting the channel list between cards so that two cards won't be on overlapping channels at the same time.
- As you add more cards, Kismet will automatically offset it so you cover as many unique channels at once as possible

Crazy Kismet setups

- Massive number of radios for massive channel coverage
- Wi-Fi Cactus uses 25 Hak5 Wi-Fi Pineapple Tetra units
- Each runs OpenWRT and has 2 802.11abgn radios
- 50 total radios
- Network remote capture to an Intel i7 NUC
- Reported ~515,000 devices in a single session at Defcon
- Filled a 2TB SSD every 2 hours



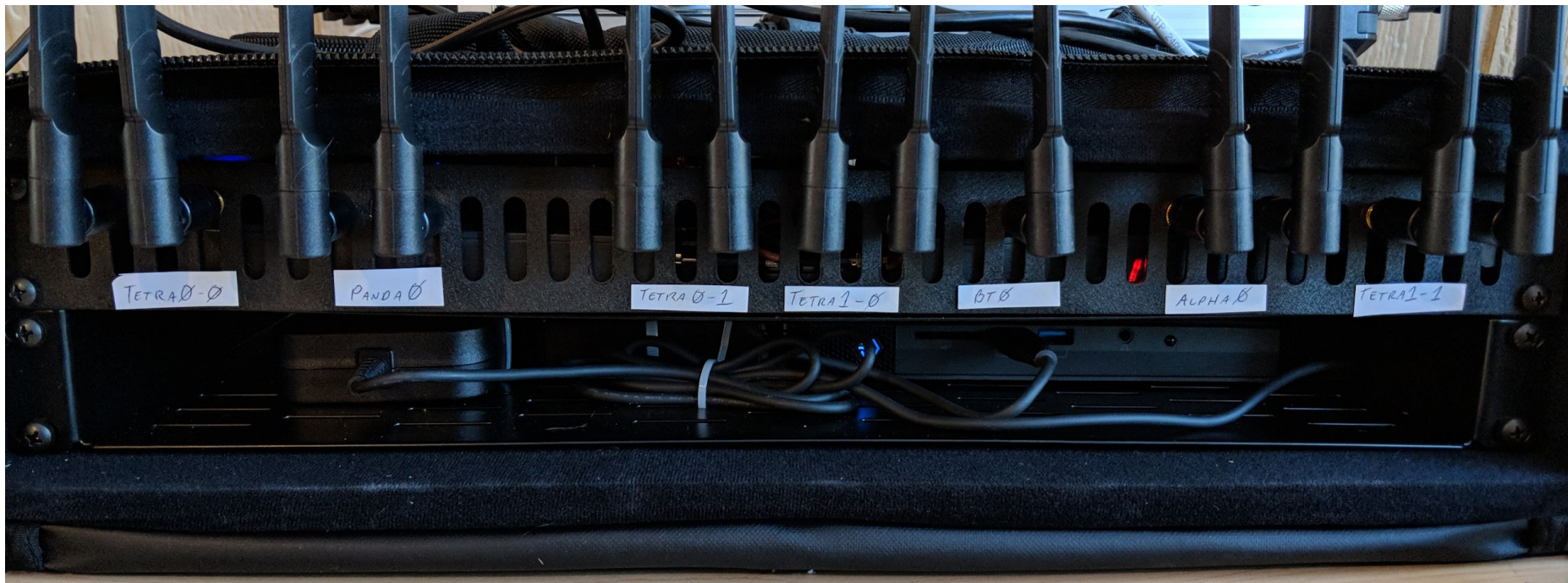
Mobile monitoring



Car NUC

- Intel i5 NUC
- Fanless industrial case - one giant heatsink
- 802.11AC, Bluetooth, RTL433-SDR, and 900MHz zigbee

Test crate



Test crate

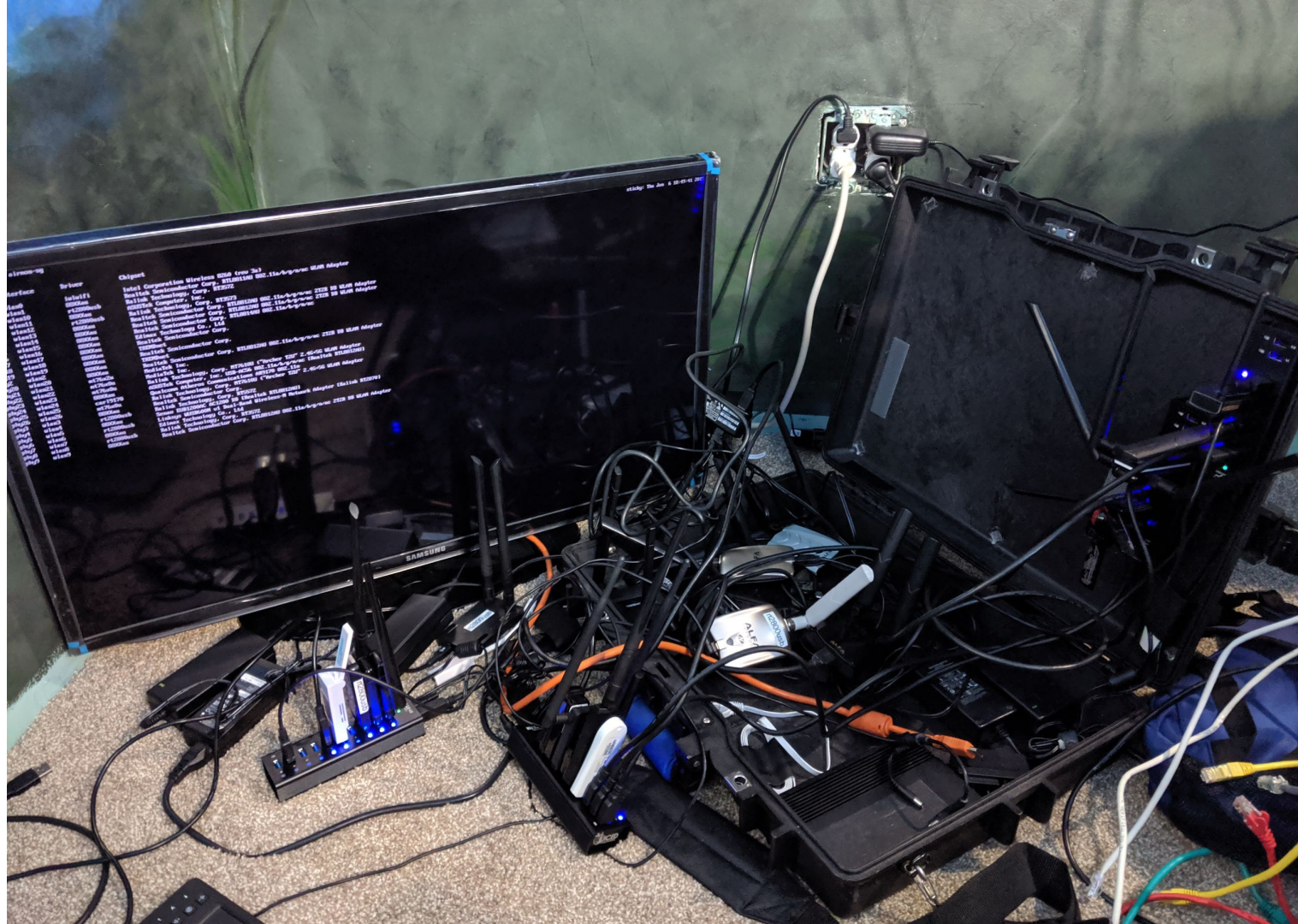
- 2U soft-case portable rack
- 2x Pineapple Tetra (4 radios)
- RTL433 radio
- Assorted USB radios
- Intel i7 NUC
- Testbed and conference demo kit
- Too heavy and obnoxious to fly with

Highly portable



Intel Compute Stick

- Intel CS125
- Quad-core Intel Atom
- 2GB RAM
- 802.11AC and Bluetooth
- Makes a nice portable system for site surveys
- Costs more than a rpi3 but is significantly more featureful (\$120)
- “Big” versions based on Intel M3 with 4 gig of ram available (\$300)



Scaling what you need

- Kismet has a lot of memory tuning knobs
- Memory used depends on the types of devices seen since Kismet tries to only allocate memory as needed
- Some rules of thumb:
- RPI3b+, 1GB RAM: 25,000 to 30,000 devices in a single session
- Intel i5, 16GB RAM: 300,000 devices in a single session

Per session

- Remember this is *per session*
- No reason smaller devices like a rpi3 can't be used with hourly session rotation, or similar
- Kismet has device timeout and log timeout options

Antennas

Antennas

- Antenna choice matters
- You'll want to pick *based on what you're trying to do*
- Always remember: antennas don't *add* power, they just *shape* the power you have
- Always going to be a tradeoff of reception in one area for reception in another

Antenna shapes

- Antennas usually come in one of two basic shapes
- Omnidirectional - radiate in all directions (hence, omni), either as a sphere or a torus (donut)
- Directional - focuses the power in one direction. Directional antennas are highly variable, typically from 180 degrees down to a few degrees

Reading antenna graphs

- All (reputable) antennas should come with a chart that shows their gain in various directions
- Typically shown as horizontal and vertical gain

Hazards of high gain

- You might think “more gain is more better!”
- It really, really depends on what you’re trying to do
- Increasing gain decreases coverage
- You can over-amplify: Too much gain, too close to an AP, is like someone screaming in your ear. It’s very loud, but may not make much sense
- You can’t get more signal, only shape it - so increasing gain takes signal away from something else
- All those cards on Amazon that come bundled with “18dB omni” antennas 2 feet tall...
- Where do you think the signal gain comes from?
- Using directional antennas blinds you to other networks

Universal gotchas

- Wi-Fi needs *antennas for Wi-Fi*
- SDR needs *antennas for the frequency you need*
- You can't just take a generic antenna for CB or something else and use it
- Must be the right frequency - 2.4GHz and 5.8GHz for Wi-Fi
- Must be 50 Ohm impedance (cable tv and CB use 75)
- Must be the right connector: Almost all Wi-Fi gear uses RP-SMA but check your manufacturer!

More than just Wi-Fi?

Other things Kismet can sniff

- If you have additional radios, Kismet can sniff more than Wi-Fi
- You'll need the right hardware though - Wi-Fi radios can only see Wi-Fi, you won't be able to see other protocols or frequencies

SDR vs dedicated

- A lot of these captures use a SDR, Software Defined Radio
- A dedicated radio (Wi-Fi, Bluetooth, etc) uses a chip designed for the protocol, and radio components designed for the protocol
- A SDR uses a generic receiver which can typically tune to many frequencies and passes raw data to the OS
- Upsides: Super flexible and can see an “infinite” number of protocols
- Downsides: Software is hard, radio is less discriminating, cost is high, power consumption is high

Cheap-as-dirt SDR

- The RTL-SDR uses a tuner designed for european digital terrestrial TV, DVB
- Someone figured out how to make it report raw samples
- It's not a *good* SDR
- But it's a *very, very cheap* SDR.
- How cheap?

This is one of the *expensive* ones



NooElec NESDR Smart Bundle - Premium RTL-SDR w/Aluminum Enclosure, 0.5PPM TCXO, SMA Input & 3 Antennas. RTL2832U & R820T2-Based Software Defined Radio.

by NooElec

★★★★☆ 217 customer reviews | 72 answered questions

Price: **\$29.95** ✓prime FREE One-Day

- Premium RTL-SDR bundle includes newly designed NESDR SMART in beautiful brushed aluminum enclosure, re-designed antenna base with 2m (6.5') RG-58 feed cable, an antenna masts. Proudly built by NooElec in the USA and Canada! Full 2-year product warranty
- A wide variety of improvements on other designs, including ultra-low phase noise 0.5PPM TCXO, RF-suitable voltage regulator, custom heatsink, 2 silicone pads and SMA connector
- Designed from the ground up to reduce USB port occlusion. Run multiple NESDR SMART side-by-side with any USB-compliant device, including tightly-spaced embedded computers like the Raspberry Pi
- SDR frequency capability approximately 25MHz-1700MHz. Frequency range can be extended down to 100kHz or lower with the Ham It Up, available on Amazon (Product B009LQT3G6)
- An 8pc SMA adapter set and carrying case is also available on Amazon (Product ID B073JT98RR)

[Compare with similar items](#)

New (2) from **\$29.95** ✓prime

[Report incorrect product information.](#)

Price: **\$29.95** ✓prime FREE One-Day

So cheap it's good

- It really is a crap radio
- But for \$20 or less to see a bunch of protocols...
- Kismet can use it for a number of things
- Or use a number of RTLSDRs for different things at the same time

Rtl433

- One of the other ISM radio bands is 433MHz
- A *lot* of little sensors live on it - weather stations, remote control light switches, wireless thermometers, and so on
- The tool **rtl_433** can decode them
- Kismet integrates rtl433 as a capture type

Weather station

DEVICE DETAILS



▶ Device Info

▼ RTL-433 (SDR)

Model	Fine Offset Electronics WH1080/WH3080 Weather Station
Device ID	89
Battery	OK
Thermometer	
Temperature	60.80° F
Humidity (%)	65%
Weather	
Wind Direction	0° (N)
Wind Gust	2.49 MPH
Rain	495



Ambient Weather F007T 8-Channel Wireless Thermometer for WS-07, WS-08, WS-09, WS-10 Weather Stations

by [Ambient Weather](#)



54 customer reviews | 11 answered questions

Price: **\$9.19** + \$7.52 shipping

Note: Not eligible for Amazon Prime.

- Supports Ambient Weather WS-07, WS-08, WS-09, WS-10 weather stations.
- The F007T features eight channel dip switches for multi-channel and degF/degC display.
- Includes Ambient Weather 1 year factory warranty.

[Compare with similar items](#)

New (1) from \$9.19 + \$7.52 shipping

 [Report incorrect product information.](#)



AcuRite 01024M Pro Weather Station with HD Display, Lightning Detector, Rain, Wind...



275

\$129.99 

[Shop now >](#)

DEVICE DETAILS



► Device Info

▼ RTL-433 (SDR)

Model	Ambient Weather F007TH Thermo-Hygrometer
-------	--

Device ID	84
-----------	----

Channel	4
---------	---

Battery	Low
---------	-----

Thermometer

Temperature	71.00° F
-------------	----------

Humidity (%)	14%
--------------	-----

ADSB

- Airplanes have a radio beacon which advertises the flight number, heading, speed, altitude, and location
- Called ADSB or Mode-S
- You guessed it, the RTLSDR can see it...
- Kismet plus airplanes? Why not!

Name	Type	Phy	Crypto	Signal	Channel	Last Seen	Data	Packets
aae5a9	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:27:08	0 B	
a39089	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:27:05	0 B	
a860b7	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:27:01	0 B	
ac51b2	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:26:28	0 B	
ad9bc8	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:26:13	0 B	
aab100	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:25:59	0 B	
a033ae	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:25:58	0 B	
ac9ff2	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:25:57	0 B	
a695ea	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:25:57	0 B	
a5fd2d	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:25:34	0 B	
a1efed	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:25:30	0 B	
a89bab	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:25:03	0 B	
a2f948	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:24:59	0 B	
a64d2b	Airplane	RTLADSB	n/a	n/a	1.090 GHz	Jun 08 2019 11:24:25	0 B	


DEVICE DETAILS



▼ Device Info

Name  **ac51b2**
Notes  **Empty**
MAC Address  53:B2:BE:01:92:06
Manufacturer  RTLADSB
Type  Airplane
First Seen Sat Jun 08 2019 11:25:02 GMT-0400 (Eastern Daylight Time)
Last Seen Sat Jun 08 2019 11:27:49 GMT-0400 (Eastern Daylight Time)

Frequencies







Channel  None Advertised
Main Frequency  1.090 GHz

Packet Frequency Distribution



Packets



Total Packets  11
LLC/Management  0
Error/Invalid  0
Data  11
Encrypted  0
Filtered  0

► RTLADSB (SDR)

► Packet Graphs

► Dev/Debug Options

DEVICE DETAILS



► Device Info

▼ RTLADSB (SDR)

Model	ac51b2
Plane ICAO	ac51b2
REG ID	n893nn
MDL	b738
Aircraft Type	Boeing 737-823
Aircraft Operator	American Airlines
Callsign	
Speed	486
Heading	132.6

AMR

- Modern power and water meters use a protocol called AMR (Automatic Meter Reading)
- This lets the power company read your meter from the street
- Usually on the 900MHz ISM band
- RTL-SDR can see it!
- Kismet can collect it!

Name	Type	Phy	Crypto	Signal	Channel	Last Seen	Data	Packets
27255590	Power Meter	RTLAMR	n/a	n/a	912.600 MHz	Jun 08 2019 11:20:25	0 B	
29401451	Power Meter	RTLAMR	n/a	n/a	912.600 MHz	Jun 08 2019 11:20:20	0 B	
28356486	Power Meter	RTLAMR	n/a	n/a	912.600 MHz	Jun 08 2019 11:20:13	0 B	
19523852	Power Meter	RTLAMR	n/a	n/a	912.600 MHz	Jun 08 2019 11:20:05	0 B	
27147336	Power Meter	RTLAMR	n/a	n/a	912.600 MHz	Jun 08 2019 11:19:20	0 B	
28073800	Power Meter	RTLAMR	n/a	n/a	912.600 MHz	Jun 08 2019 11:18:56	0 B	

DEVICE DETAILS

X

▶ Device Info

▼ RTLAMR (SDR)

Model	29401451
Device ID	29401451
Current Reading	13015

Bluetooth

- Bluetooth can be a challenge to sniff
- BTLE is easier to find
- Kismet currently works with on-board Bluetooth cards in Linux to scan
- Scanning mode only right now
- Looks for advertising BT classic devices
- Looks for BTLE

Name	Type	Phy	Crypto	Signal	Channel	Last Seen	Data	Packets	Clients	BSSID
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:09	0 B		0	0
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:12	0 B		0	0
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:12	0 B		0	0
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:03	0 B		0	0
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:15	0 B		0	0
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:24:36	0 B		0	0
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:08	0 B		0	0
Tile	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:24:03	0 B		0	0
Linus iPhone	BR/EDR	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:08:22	0 B		0	0
LE_WH-1000XM3	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:20	0 B		0	0
LE-tommy n bose	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:22	0 B		0	0
LE-shut it	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:19	0 B		0	0
LE-reserved_K	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:21	0 B		0	0
LE-chilly	BTLE	Bluetooth	n/a	n/a	FHSS	Jun 09 2019 13:30:20	0 B		0	0

Zigbee

- Kismet can capture Zigbee using the Freaklabs cards
- More to come
- Right now Kismet captures, but doesn't dissect
- No device display, but packets go in the pcap for Wireshark to use

Mouse and Keyboard

- A common chipset used for mice and keyboards is the NordicRF 2.4 chip
- Sniffable using a range of hardware, including the CrazyPA
- Various devices are more (or less) vulnerable
- Some use NO encryption
- Kismet uses the MouseJack firmware and a CrazyPA or compatible USB device

ZWave

- ZWave home automation can be seen by the Yardstick1 and rfc4
- YS1 uses a cheap radio that can decode many formats
- ZWave decode in Kismet is *super* basic right now, more a proof of concept

Detecting shenanigans

WIDS

- IDS = Intrusion Detection System
- WIDS = Wireless Intrusion Detection System
- Aren't we clever in naming? I didn't come up with it.
- Kismet can do a number of WIDS-like functions
- There's also WIPS - Wireless Intrusion *Prevention* System...
- Kismet isn't that. It's also arguable how much prevention against some attacks you can do!

Wi-Fi attacks

- Surprisingly large attack surface
- Denial of service (DoS) attacks against the Wi-Fi protocol
- DoS against the physical media (RF jamming)
- Impersonation and spoofing attacks impersonating the wireless infrastructure
- Impersonation and cloning attacks impersonating wireless clients
- Exfiltration of data over Wi-Fi where there shouldn't be Wi-Fi
- Attacks against company infrastructure over Wi-Fi
- Attacks against the Wi-Fi drivers *directly* in clients and access points

Kismet alert modes

- Kismet has two main alert mechanisms
- *Trend* based alerts, where nothing is specifically *wrong*, but getting a lot of packets can be a problem (for instance, DoS attacks)
- *Fingerprint* based alerts, where a packet of that type should *never* occur, or where specific attacks against drivers or encryption can be identified
- User-configurable fingerprints like SSID matching

Kismet IDS detections

- Automatic learning & alerting on encryption, channel, and DHCP changes and conflicts. If the same AP appears on two channels, or suddenly advertises as “Open” when it was previously encrypted, Kismet will let you know. On Open networks, if a DHCP client changes OS or other fingerprints, Kismet will let you know
- Regex-based AP spoof detection; Kismet can be configured to detect variations of your SSID and alert on attempts to fool users
- Fingerprints of attacks against WPA, iPhone and Android Wi-Fi drivers, brute-force attempts against WEP and WPS, and more

Advanced fingerprinting

- Still being developed
- Instead of using only the MAC address and SSID text, fingerprint more attributes
- Beacons have a LOT of tags in them - Kismet now fingerprints the ones that shouldn't change
- Gives better insight into network spoofing - now tools have to accurately spoof *everything* about an AP
- Tag list used for fingerprint is configurable, too

The more data we have...

- The more we can look for discrepancies
- Being able to compare over time and see how multiple devices are claiming the same thing lets us look for oddities

Integrating with other tools

Making Kismet talk to other tools

- What good is a tool if it can't talk to other things?
- Kismet has several APIs for talking to other tools
- Old Kismet had a weird TCP protocol kind of based on IMAP
- New Kismet has HTTP, a REST-like interface, and outputs JSON
- New Kismet also has a runtime IPC protocol to extend functionality

Rest API

- Everything in the Kismet UI is generated by the REST-like API
- Kismet outputs standard JSON
- Any language that can talk HTTP and JSON can talk to Kismet
- Heavily documented - one of the big failings of the older Kismet code was a lack of docs
- Python module available in pip (kismet-rest)

REST API features

- Trivial to query devices, access points, etc
- 'Ekjson' streaming format (just like tshark) for incremental processing
- Windowed device viewing (efficient loading of target areas of very large numbers of devices)
- Streaming pcap and pcapng format for live packet export
- Alerts, messages, etc
- Mandatory documentation of fields and live retrieval of all fields

Kismet field descriptions

Name	ID	Type	Description
kismet.device.base.manuf	1	string/0	manufacturer name
kismet.stream.stream_id	2	double/10	Stream ID
kismet.stream.time	3	uint64_t/8	Start time of stream (second since epoch)
kismet.stream.name	4	string/0	Stream / Log name
kismet.stream.type	5	string/0	Stream / Log type
kismet.stream.path	6	string/0	Log path or stream remote client
kismet.stream.description	7	string/0	Stream / Log description
kismet.stream.packets	8	uint64_t/8	Number of packets (if known)
kismet.stream.size	9	uint64_t/8	Size of log, if known, in bytes
kismet.stream.max_packets	10	uint64_t/8	Maximum number of packets
kismet.stream.max_size	11	uint64_t/8	Maximum allowed size (bytes)
kismet.stream.paused	12	uint8_t/2	Stream processing paused
kismet.stream.stream	13	map[field, x]/14	Kismet data stream
kismet.messagebus.list	14	vector[x]/13	list of messages
kismet.messagebus.timestamp	15	uint64_t/8	message update timestamp
kismet.messagebus.message_string	16	string/0	Message content
kismet.messagebus.message_flags	17	int32_t/5	Message flags (per messagebus.h)
kismet.messagebus.message_time	18	uint64_t/8	Message time_t
kismet.messagebus.message	19	map[field, x]/827983394	Kismet message
kismet.datasource.driver.type	20	string/0	Type
kismet.datasource.driver.description	21	string/0	Description
kismet.datasource.driver.probe_capable	22	uint8_t/2	Datasource can automatically probe
kismet.datasource.driver.probe_ipc	23	uint8_t/2	Datasource requires IPC to probe
kismet.datasource.driver.list_capable	24	uint8_t/2	Datasource can list interfaces
kismet.datasource.driver.list_ipc	25	uint8_t/2	Datasource requires IPC to list interfaces
kismet.datasource.driver.local_capable	26	uint8_t/2	Datasource can support local interfaces
kismet.datasource.driver.local_ipc	27	uint8_t/2	Datasource requires IPC for local interfaces
kismet.datasource.driver.remote_capable	28	uint8_t/2	Datasource can support remote interfaces
kismet.datasource.driver.passive_capable	29	uint8_t/2	Datasource can support passive interface-less data
kismet.datasource.driver.tuning_capable	30	uint8_t/2	Datasource can control channels
kismet.datasource.tracker.driver	31	map[field, x]/1380255737	Datasource driver information
kismet.datasource.source_number	32	uint64_t/8	internal source number per Kismet instance

Pcap over HTTP

- Grab live packets from Kismet - from anywhere you can reach the Kismet server...
- Filtered by capture source, tracked device, etc...
- As a pcap-ng with original capture source and all original headers!

```
dragorn@tabby:~$ curl -s http://kismet:kismetdemo@kismet.lan:2501/pcap/all_packets.pcapng > packets.pcapng
```

Proxy

- Oh yeah - and since Kismet is HTTP...
- You can run it through a proxy like nginx
- Proxy a directory into a Kismet server
- Add SSL via LetsEncrypt, etc

Kismetdb log

- Old Kismet logs were separate logs for GPS, pcap, and networks
- Using poorly-documented and often slightly malformed XML which nobody liked parsing
- New unified log format
- Holds packets, messages, non-packet data (like SDR records and such), GPS, device records; everything Kismet knows about
- Actually just a sqlite3 file
- Complex records stored as JSON blobs (a trick stolen from NoSQL databases and ELK)

Simple to process

- Kismetdb Python module in pip
- Any language which can talk sqlite3 and JSON can decode everything
- Trivial to convert to CSV, text, WIGLE, etc
- Tools included to turn kismetdb into PCAP for wireshark, other tools
- Normalized data for searching for specific logs

Historic live data!

- Having kismetdb logs gives us something else though...
- *Live access to historical data*
- Typically Kismet processes packets, logs them, and has no access to them again. Same for alerts and IDS events, and so on
- Now that we can query the kismetdb log, we provide REST API access to historic data

Packet slicing

- API lets us request a pcap file of packets, with filtering
- Want to grab all the packets within 5 minutes before and after an IDS event?
- All the packets with a signal stronger than -20dB?
- And so on... filter by time, capture source, signal, location boundaries, packet size...
- All queryable over JSON to get on-demand PCAP files

Handshake logging

- Since we can generate pcap and pcapng on demand...
- Kismet will also track WPA handshake packets
- Flags when a complete handshake has been found
- Download of a beacon+handshake for feeding to tools like aircrack-ng and hashcat
- Also captures handshakes with a PMKID for new attacks

WPA Key Exchange

Handshake
Packets

3

Handshake
PCAP

 **Download Pcap File**

Extending Kismet

- Kismet also has an IPC API (inter-process communication)
- Kismet packet capture and remote capture use the IPC channels
- External tools can also interact with the Kismet server to add features...
- Like adding new REST endpoints to the API, querying Kismet directly, and extending the Javascript and HTML UI

Which gets us...

- All sorts of options
- For instance, the ability to manage long-running processes (like testing WPA security during a pentest) and expose the status of them back to the user via Kismet
- Adding new protocols
- Automatically managing other external tools

Extending the UI

- The Kismet UI was designed to be extendable
- Datasources and plugins can add JS modules to be loaded automatically & add details for different decoders
- Or replace the entire UI with something else for specific purposes!

Mobile dashboard

- All new mobile-centric dashboard by ElKentaro
- Optimizes Kismet for small screens, tablets, etc
- Entire new UI written in HTML/JS and loaded as a plugin with no C++ code required

Home

System Details: Uptime: 13 minutes.

3/3

18

1

45

Active/Total Sources

Channels

Packets/sec

Total # of Devices

802.11 details

14

12

6

7

0

0

APs

Clients

Devices

Bridged

AdHoc

WDS

802.11 Channels

Current

Historical

Frequency

Devices per Channel

1.0

0.5

0

2.412 GHz

2.417 GHz

2.427 GHz

2.437 GHz

2.447 GHz

2.457 GHz

2.462 GHz

2.472 GHz

Bluetooth

6

0

BTLE Devices

BR/EDR Devices

Other

-

-

-

-

-

RTL433

Z-Wave

Mousejack

UAV

Unknown*

(*unkown=Emplyt kismet.device.basetype)

Home

Listing:Wi-Fi AP

1: aterm-dbe7f8-g

2: aterm-dbe7f8-gw

3: 2A:76:10:05:3B:3C

4: 2E:76:10:05:3B:3C

5: 28:76:10:05:3B:3C

6: HUMAX-807EE

7: Secretbase

8: 106F3FDB42FA

9: auhome_adRNSn-W

10: 106F3FD973C5

11: mFi_EA2EEC

12: rv440n-329775-2

13: aterm-c1f7b3-g

14: rv440n-329775-1

Future integration with Wireshark

- Soon will have Kismet tools for integrating directly with Wireshark, thanks to Wireshark's extendability
- Goals soon:
 - Open kismetdb logs directly in Wireshark
 - Access Kismet servers directly in Wireshark using extcap
 - Access Kismet remote capture nodes directly in Wireshark using extcap
- Kismet remote cap + Wireshark extcap will give us the ability to capture packets from any OS by adding a simple USB device or Ethernet device.

Stuff to do with Kismet

Wardriving and site survey

- Wardriving - mapping and finding networks and devices
- Aka “Site survey” if you want to be respectable
- Kismet can integrate with GPS to log location as well as signal data
- Survey facilities looking for out-of-place networks, improperly configured networks, etc

Site WIDS

- Multiple Kismet sensors using remote capture to a single, big server
- OR
- Multiple Kismet sensors running servers, collected over the Kismet REST API
- Monitor client movement, access point state, etc across an entire building (or campus)

Kismet Central

- New project to collate Kismet server data for site-wide WIDS
- Not quite ready for public release, but will be OSS
- Based on the Docker and Django stack:
 - Docker
 - Django
 - Celery task manager
 - Postgres
- Kismet has restrictions because it's designed to provide all the data from the runtime environment
- Kismet-Central is designed to collect data and operate on the Postgres database

Aggregating Kismet servers

- KC can take any number (for some large definition of “any”) Kismet servers
- Scrapes the runtime alert, server health, datasource health, and device lists
- Maintains maps, over time, of devices, Wi-Fi SSIDs, clients, etc
- Can alert across a whole deployment if an unknown device response for a SSID, can monitor clients across an entire deployment, etc
- Generates both a runtime UI and JSON data

Tracking SSIDs

- Able to aggregate SSIDs across all sensors
- Maps every BSSID beaconing or probing to the SSID record
- Allows drilling down to find clients across all BSSIDs in a SSID, regardless of Kismet server that saw it
- Can establish relationships of SSIDs on the same physical network, detect BSSIDs not advertising the correct encryption, and more

Tracking devices

- Devices can be APs or clients
- Can track devices across all sensors
- Know what physical Kismet servers saw a device
- Know *when* a sensor saw a device
- Track device behavior over time, what SSID it associated to, when, from where
- Drill down to the entire JSON record from each Kismet server, for each device

Combining log files

- Can also upload kismetdb logs
- Site survey logs, etc can be combined in the device list and SSID lists
- Can add detected SSID data and advertisements to the collection for non-real-time data

Getting Kismet

Source & packages

- <https://www.kismetwireless.net>
- Releases and latest code available in git to compile yourself
- Also packages for:
 - Ubuntu Xenial
 - Ubuntu Bionic
 - Ubuntu Cosmic
 - Ubuntu Dingo
 - Kali x86
 - Raspbian rpi3 and rpi0
 - Kali rpi3
 - Kali rpi0

Distros can be slow

- Distributions are often slow to pick up new versions
- Strongly recommend you get the latest code or packages
- Packages available as releases and nightlies!

Compiling modern C++ can be challenging

- Kismet is about 3.5MB when compiled
- Some files need 2.5GB of RAM
- Modern C++ can be very very hungry resolving compile-time things
- Suggest packages for systems like rpi3, etc

Let's taunt the demo gods

What's next

- Stability and performance
- Supporting more RF protocols
- Better BTLE/BT support, Zigbee support
- More distro packages
- More remote capture options to support OSX and Windows

Info

<https://www.kismetwireless.net>

<https://docs.kismetwireless.net>

@KismetWireless