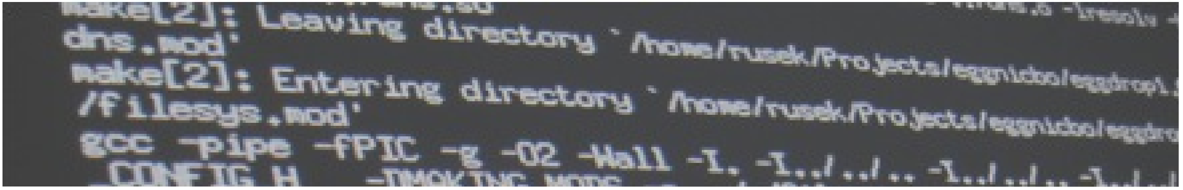


```
make[2]: Leaving directory `/home/rusek/Projects/eggnicbo/eggdrop1.
dns.mod'
make[2]: Entering directory `/home/rusek/Projects/eggnicbo/eggdrop1.
/filesys.mod'
gcc -pipe -fPIC -g -O2 -Wall -I. -I../../.. -I../../..
  CONFIG_H    -DMAKING_MODS
```

Active 802.11 fingerprinting

Sergey Bratus
Cory Cornelius, Daniel Peebles,
Axel Hansen

Dartmouth College
INSTITUTE FOR SECURITY
TECHNOLOGY STUDIES
Cyber Security and Trust Research & Development
http://www.ISTS.dartmouth.edu

Motivation

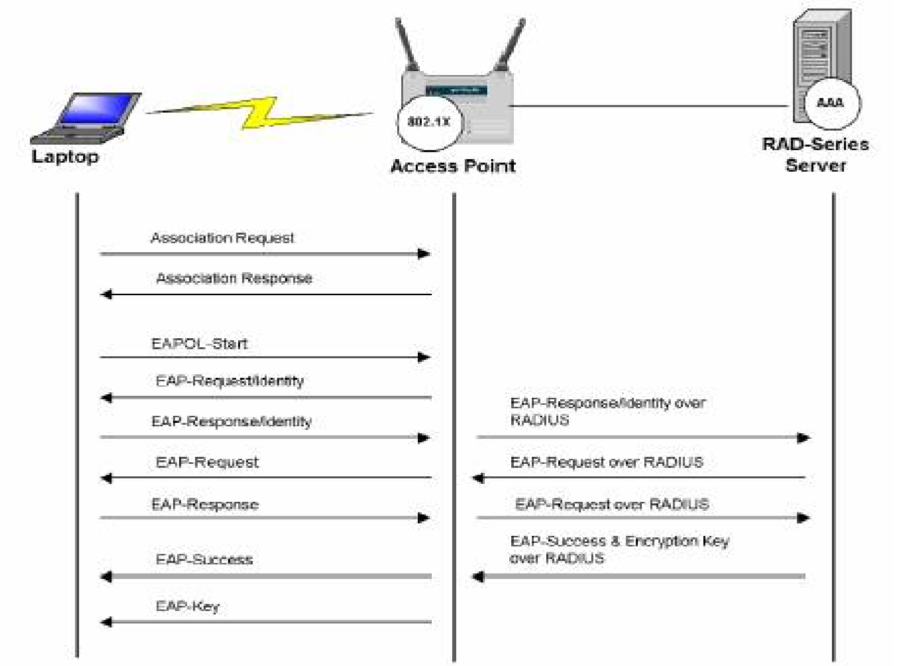
Can a client station trust an AP?

Is this AP one of a trusted group, or evil faker?

Why yes, just exchange some crypto with it, and verify the AP knows the right secrets.

Problem solved, right?

Not exactly: are all these exchanges ***bug-free***?



The problem

Initially, an AP is just a MAC address
(and other easily faked info)

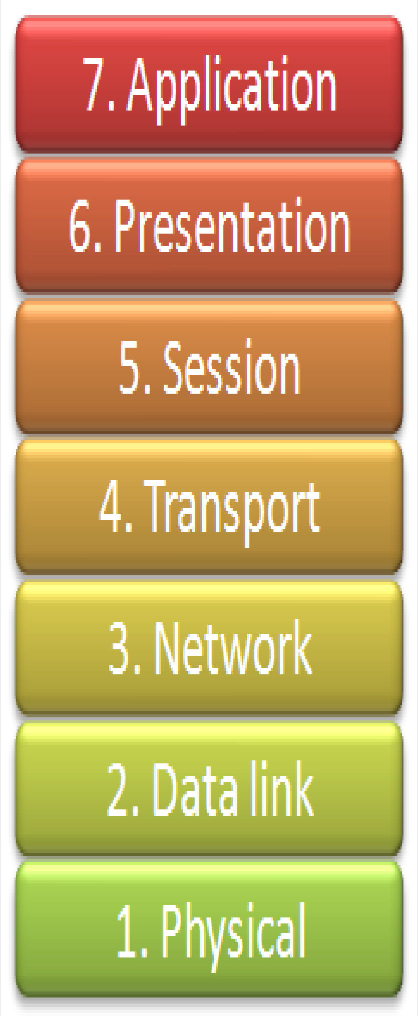
That's all we know.

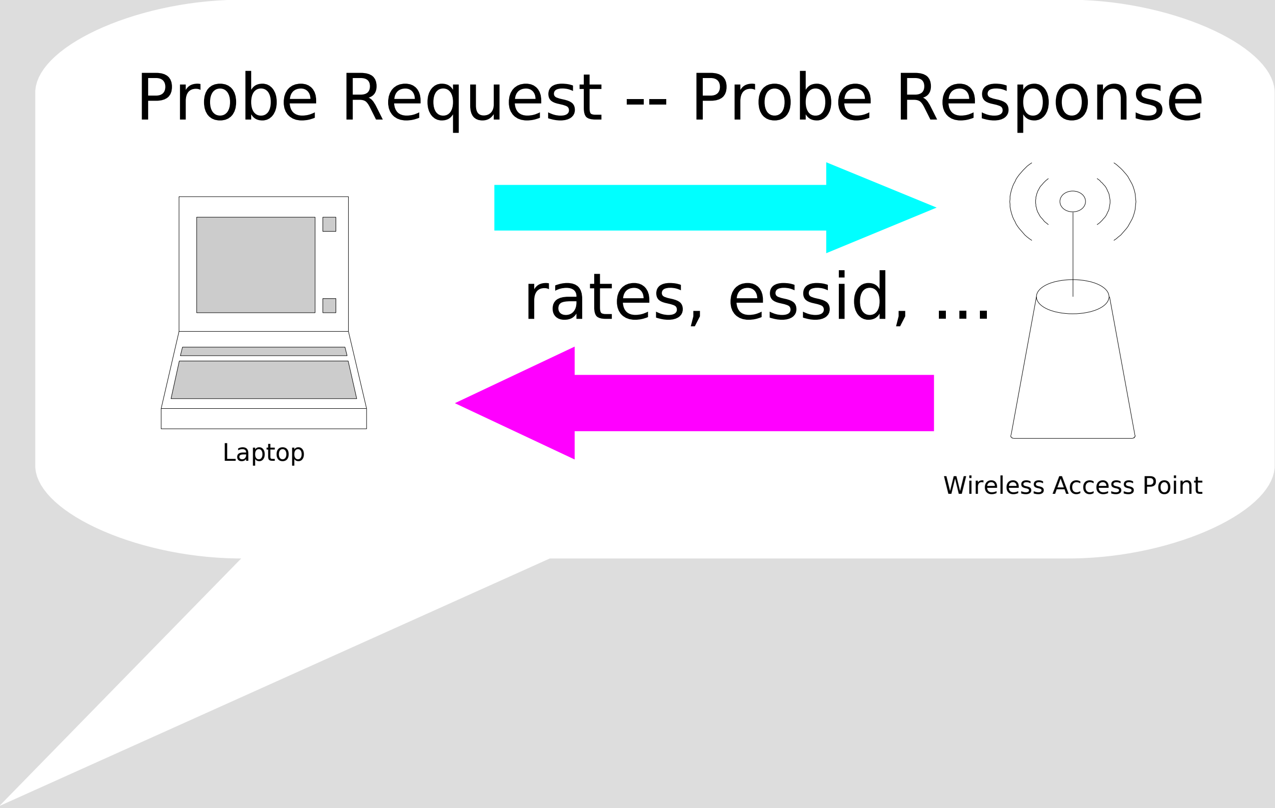
Trust me!

- To perform crypto authentication of AP, driver must parse complex data structures
- Complex data from untrusted source?

  -- *Is this such a good idea?*

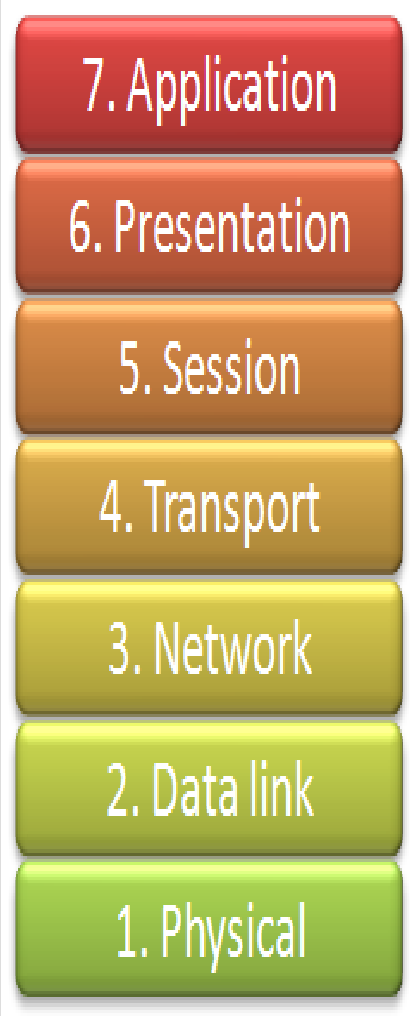
Say it ain't so



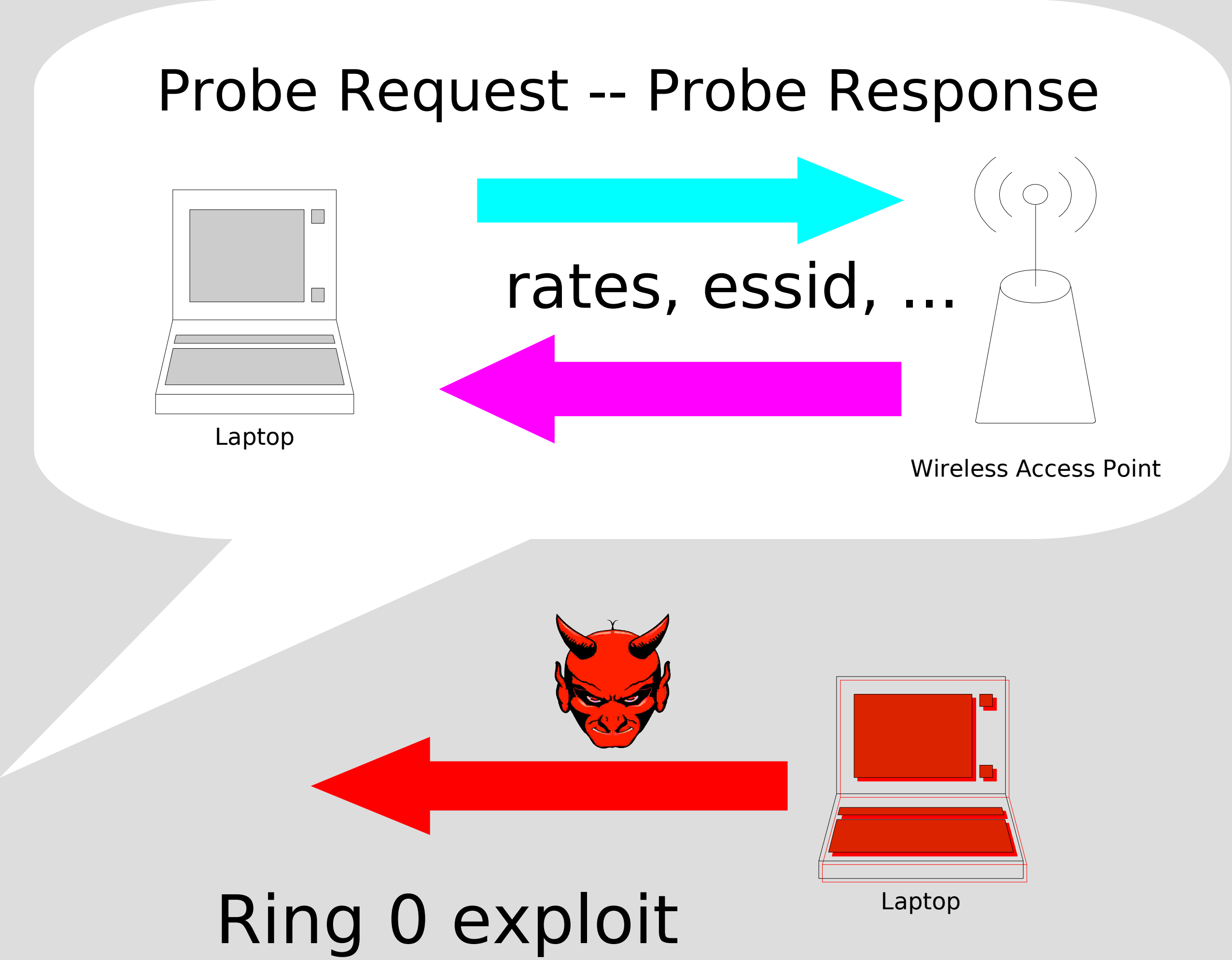

Probe Request -- Probe Response

rates, essid, ...

Say it ain't so



7. Application
6. Presentation
5. Session
4. Transport
3. Network
2. Data link
1. Physical

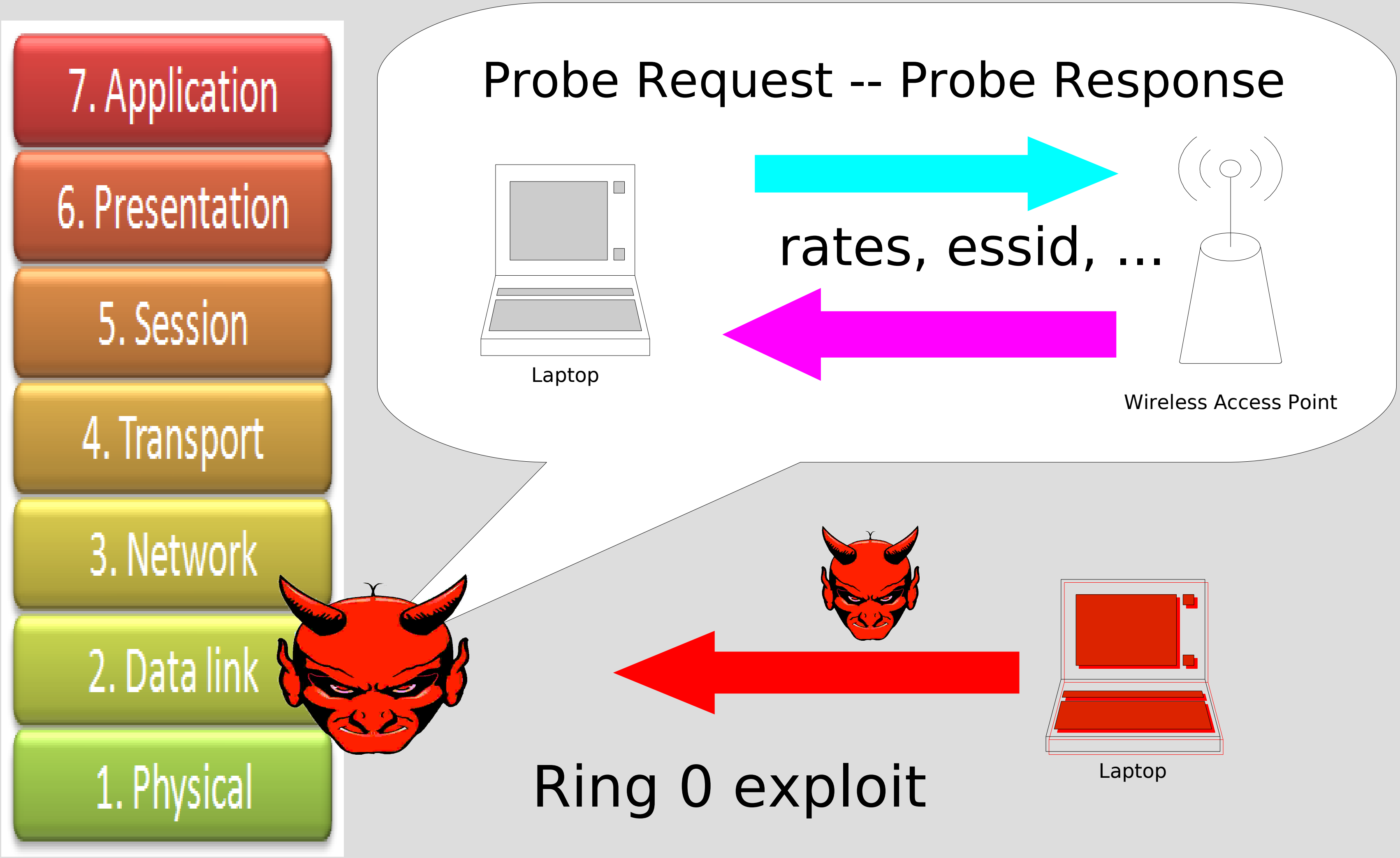


Probe Request -- Probe Response

rates, essid, ...

Ring 0 exploit

Say it ain't so



Ring 0 exploit

AP vs. clients

Early 802.11: AP = castle, must fight off barbarians (unauthorized clients)

Reality: can peasants = clients find the right castle?

- *Dai Zovi, Macaulay:* Karma
- *Shmoo:* “Badass tackle...”
- *Simple Nomad:* “Friendly skies...”
- *Cache & Maynor:* “Hijackng a MacBook in 60 seconds”
- Month of kernel bugs (Nov '06)

Fingerprint it!

Fingerprint the AP before trying to authenticate and associate with it:

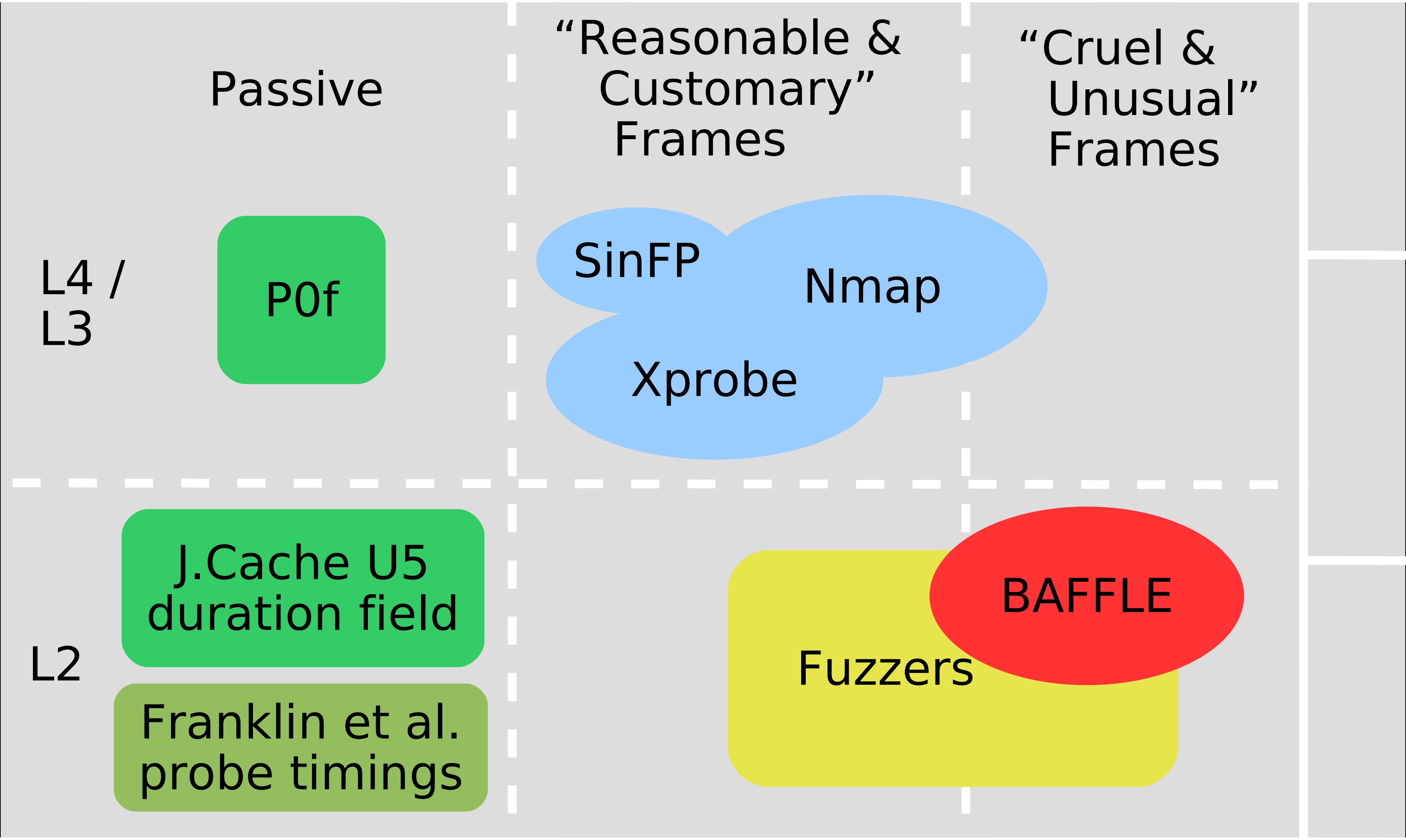
- limit the kinds of accepted data

Must be simple & cheap (no RF spectrum analysis, Fourier transforms, etc.)

Follow IP stack fingerprinting ideas:

- unusual and non-standard header field combinations – but in link layer (L2)

Where we fit in



Passive

"Reasonable & Customary" Frames

"Cruel & Unusual" Frames

L4 / L3

P0f

SinFP

Nmap

Xprobe

L2

J.Cache U5 duration field

Franklin et al. probe timings

BAFFLE

Fuzzers

TCP/IP fingerprinting

L3, need an L2 connection

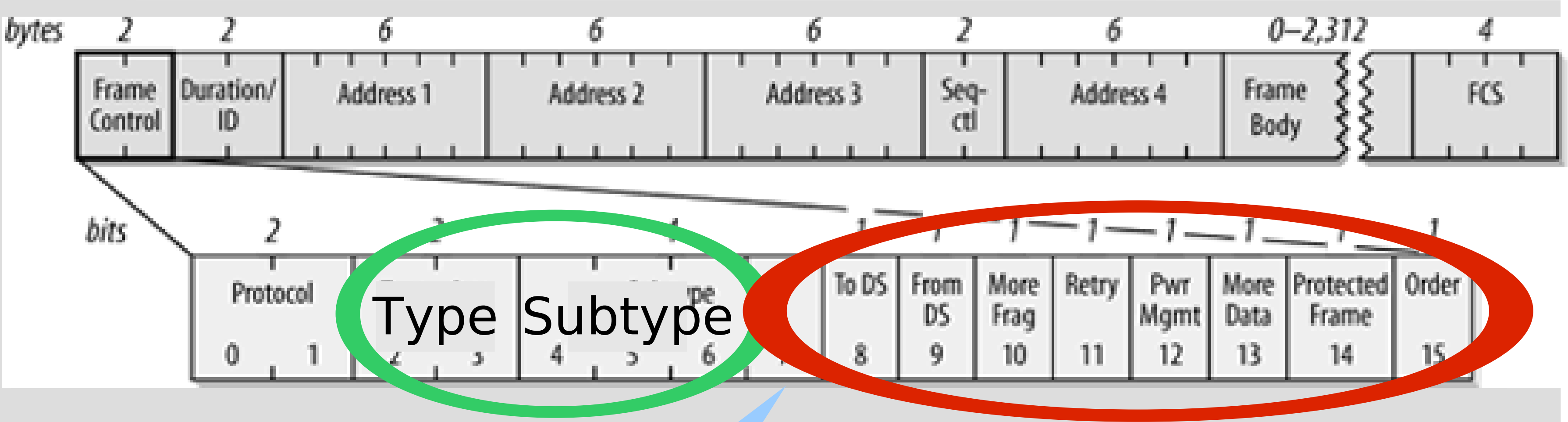
- **Nmap** (1998-2006, ...)
- **Xprobe** (2001, 2005, ...)
- **P0f** (2000, 2006)
- **SinFP** (2005)
- Timing-related: *Ping RTT* (2003), *Clock Skew* (2005)

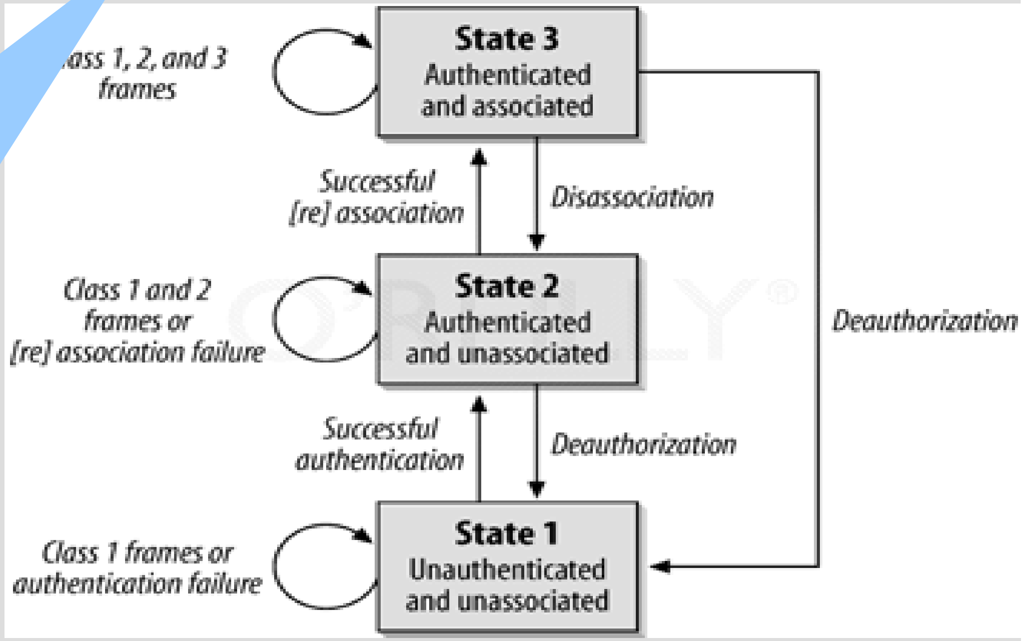
- Scrubbers: **Norm**, **Bro** (2000-01)
- **Honeyd**, **Morph** (2004-)
- ... ?

BAFFLE

- Written in Ruby 1.8.2
- Ruby LORCON bindings from Metasploit
- Builds Pcap/BPF filters for 802.11 frames from Ruby objects
- Domain-specific language for tests, probes, and for matching responses

Bits and states

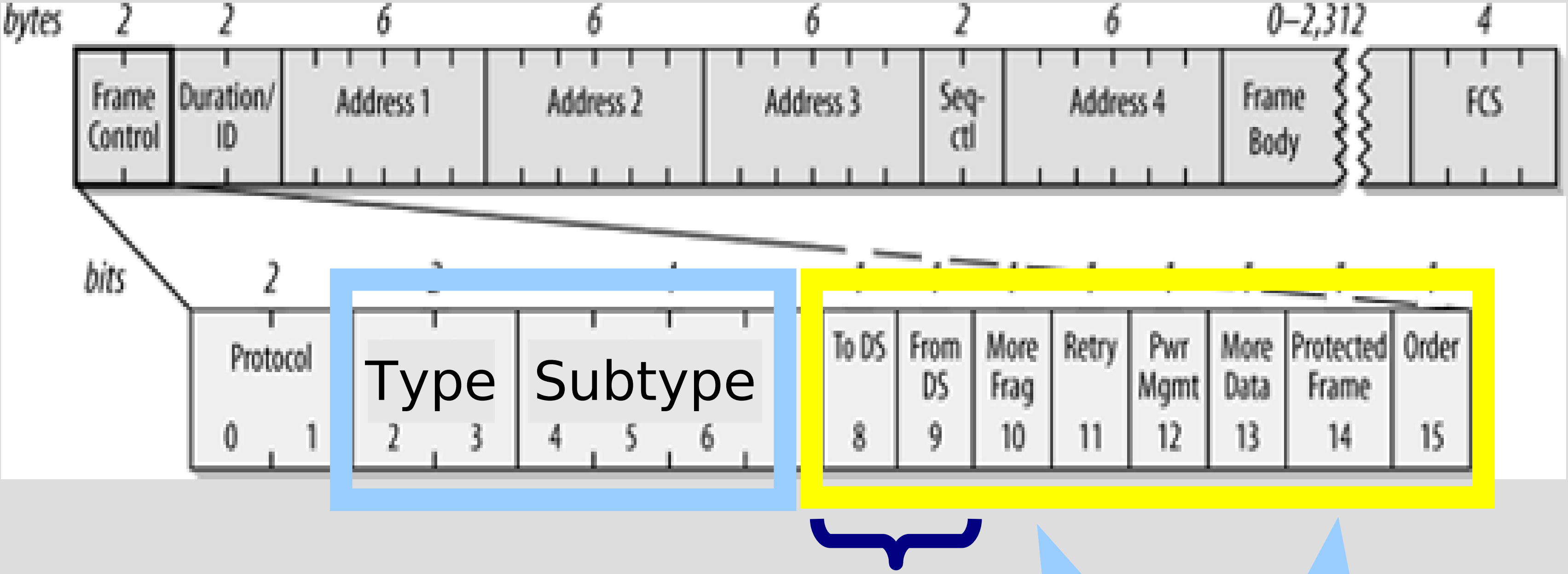



State 3
Authenticated and associated
ass 1, 2, and 3 frames
Successful [re] association
Disassociation
State 2
Authenticated and unassociated
Class 1 and 2 frames or [re] association failure
Successful authentication
Deauthorization
Deauthorization
State 1
Unauthenticated and unassociated
Class 1 frames or authentication failure

Not all flags make sense for all types & subtypes

Not all flags make sense for all states

802.11 fiddly bits

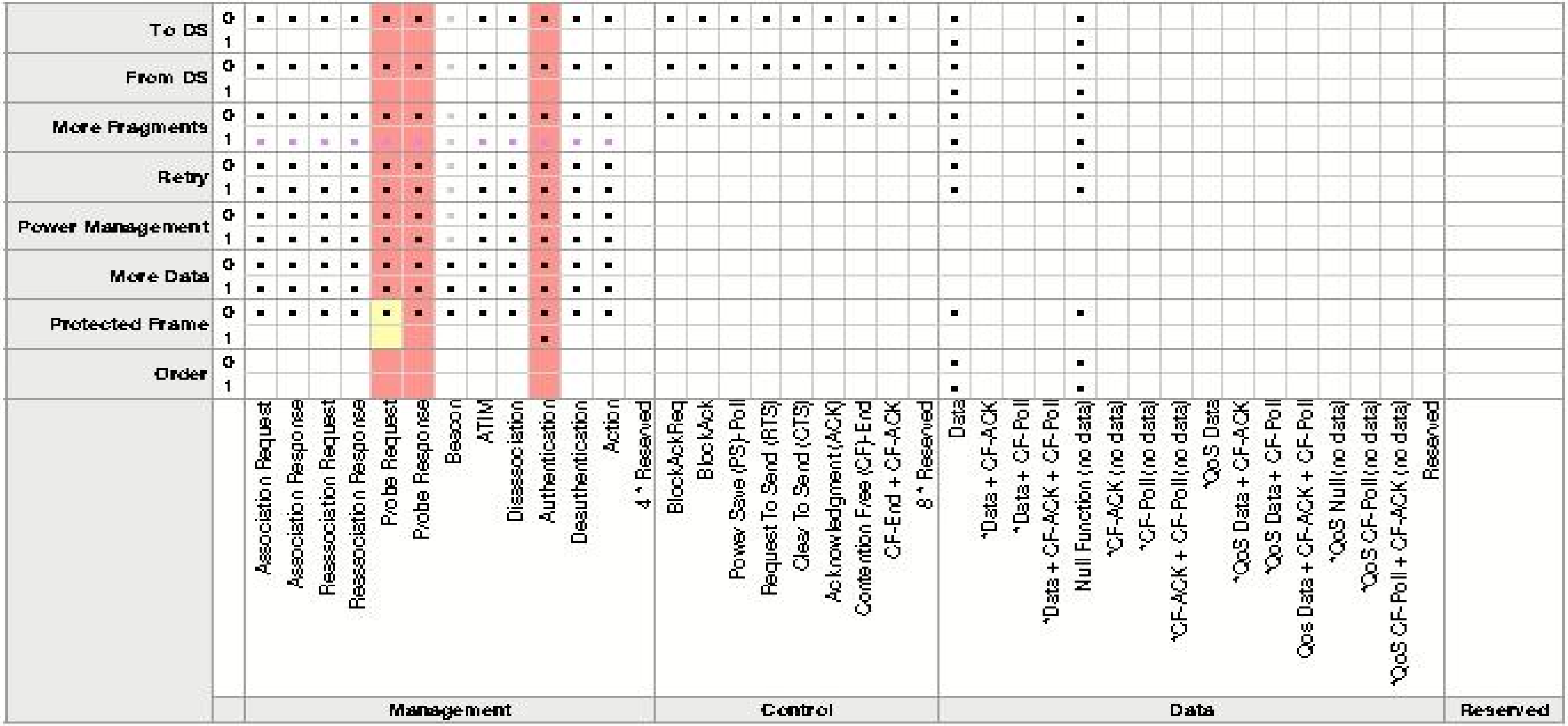


Only 0 makes sense on Mgmt & Ctrl frames

Unusual on Probes

Not for Mgmt frames

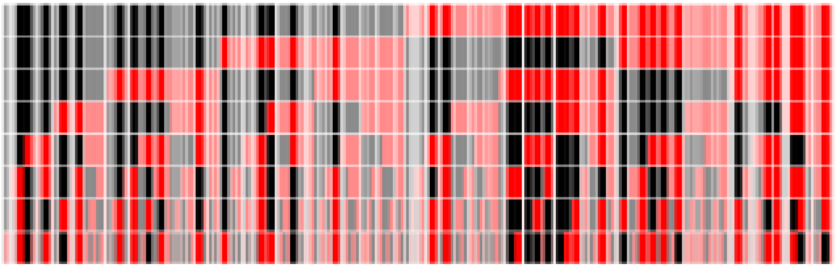
So many flags...



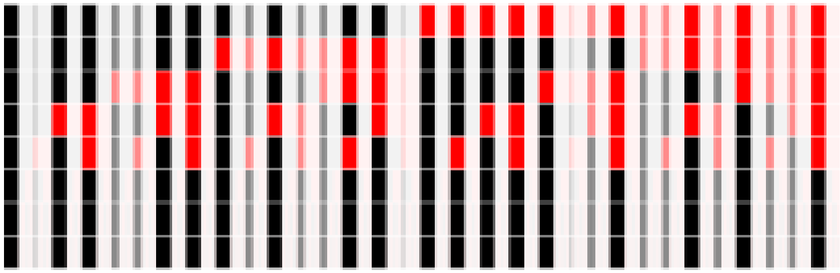
		Association Request	Association Response	Reassociation Request	Reassociation Response	Probe Request	Probe Response	Beacon	ATIM	Disassociation	Authentication	Deauthentication	Action	4 * Reserved	BlockAckReq	BlockAck	Power Save (PS)-Poll	Request To Send (RTS)	Clear To Send (CTS)	Acknowledgment (ACK)	Contention Free (CF)-End	CF-End + CF-ACK	8 * Reserved	Data	*Data + CF-ACK	*Data + CF-Poll	*Data + CF-ACK + CF-Poll	Null Function (no data)	*CF-ACK (no data)	*CF-Poll (no data)	*CF-ACK + CF-Poll (no data)	*QoS Data	*QoS Data + CF-ACK	*QoS Data + CF-Poll	Qos Data + CF-ACK + CF-Poll	*QoS Null (no data)	*QoS CF-Poll (no data)	*QoS CF-ACK + CF-Poll (no data)	Reserved	
To DS	0	■	■	■	■	🟥 ■	🟥 ■	▫	■	■	🟥 ■	■	■		■	■	■	■	■	■	■	■		■				■												
	1					🟥	🟥				🟥													■				■												
From DS	0	■	■	■	■	🟥 ■	🟥 ■	▫	■	■	🟥 ■	■	■		■	■	■	■	■	■	■	■		■				■												
	1					🟥	🟥				🟥													■				■												
More Fragments	0	■	■	■	■	🟥 ■	🟥 ■	▫	■	■	🟥 ■	■	■		■	■	■	■	■	■	■	■		■				■												
	1	🟪	🟪	🟪	🟪	🟥 🟪	🟥 🟪	▫	🟪	🟪	🟥 🟪	🟪	🟪											■				■												
Retry	0	■	■	■	■	🟥 ■	🟥 ■	▫	■	■	🟥 ■	■	■											■				■												
	1	■	■	■	■	🟥 ■	🟥 ■	▫	■	■	🟥 ■	■	■											■				■												
Power Management	0	■	■	■	■	🟥 ■	🟥 ■	▫	■	■	🟥 ■	■	■																											
	1	■	■	■	■	🟥 ■	🟥 ■	▫	■	■	🟥 ■	■	■																											
More Data	0	■	■	■	■	🟥 ■	🟥 ■	■	■	■	🟥 ■	■	■																											
	1	■	■	■	■	🟥 ■	🟥 ■	■	■	■	🟥 ■	■	■																											
Protected Frame	0	■	■	■	■	🟨 ■	🟥 ■	■	■	■	🟥 ■	■	■											■				■												
	1					🟨	🟥				🟥 ■																													
Order	0					🟥	🟥				🟥													■				■												
	1					🟥	🟥				🟥													■				■												
		Management													Control									Data																Reserved

Legend
- ■ Defined by IEEE 802.11 Specification
- ▫ In IEEE 802.11 Specification but purpose seems undefined
- 🟪 In IEEE 802.11 Specification but unlikely
- 🟥 Tested by BAFFLE
- 🟨 Tested by BAFFLE but of limited utility
- □ Not defined in IEEE 802.11 Specification
- * In IEEE 802.11 Specification but mostly unimplemented

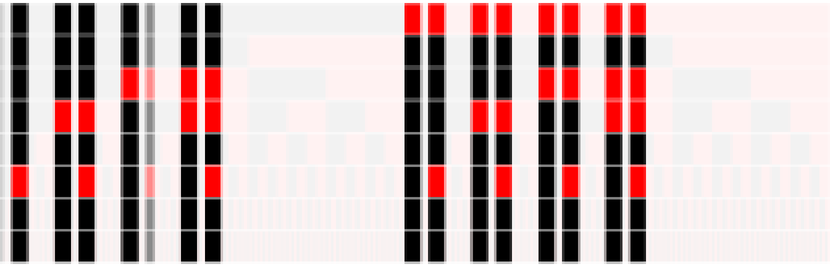
Probe Request tests



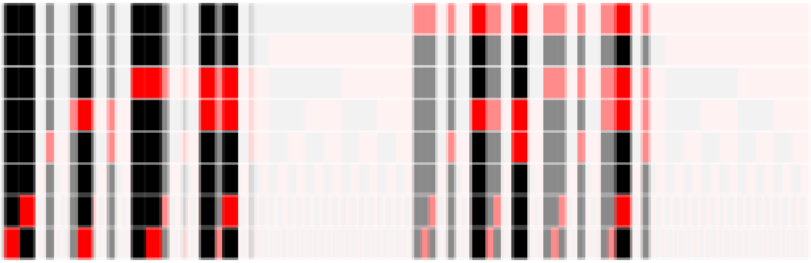
Cisco-Linksys WRT54g *ProbeFCTest*



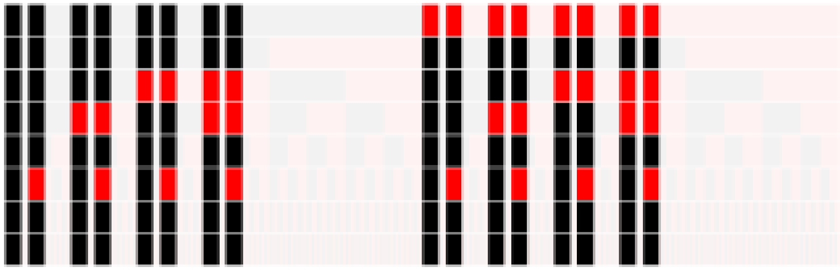
Extrasys WAP-257 *ProbeFCTest*



Madwifi-ng soft AP *ProbeFCTest*

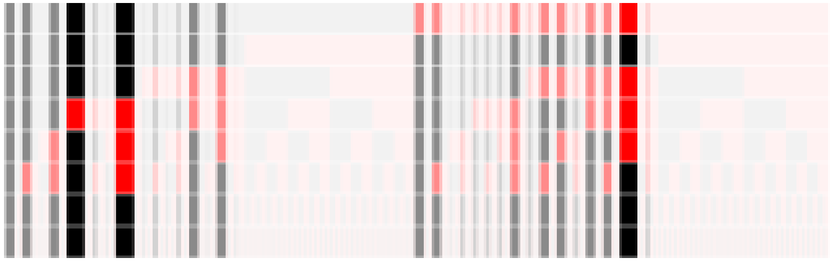


Hostap soft AP *ProbeFCTest*

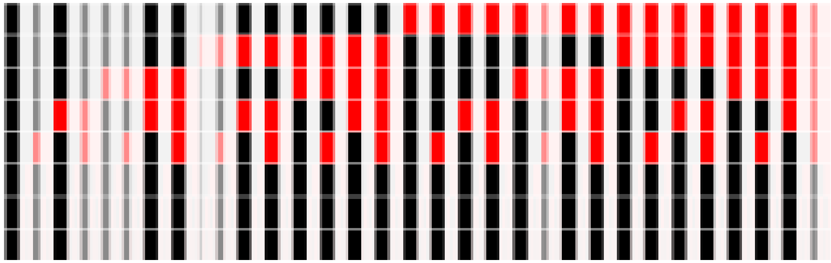


Aruba OpenWRT *ProbeFCTest*

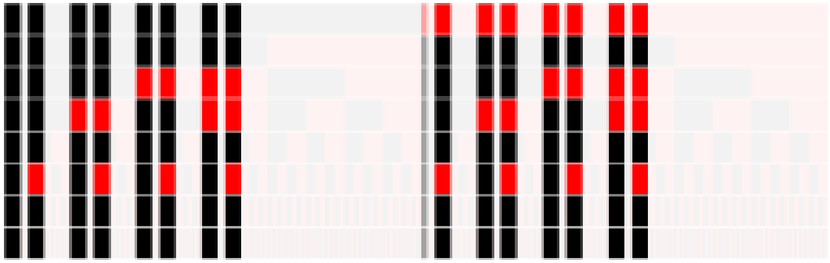
Auth Request tests



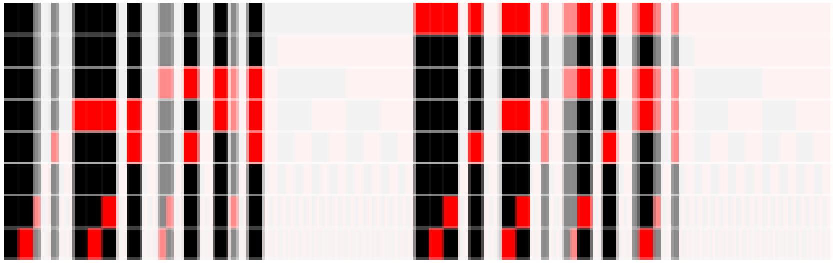
Cisco-Linksys WRT54g *AuthFCTest*



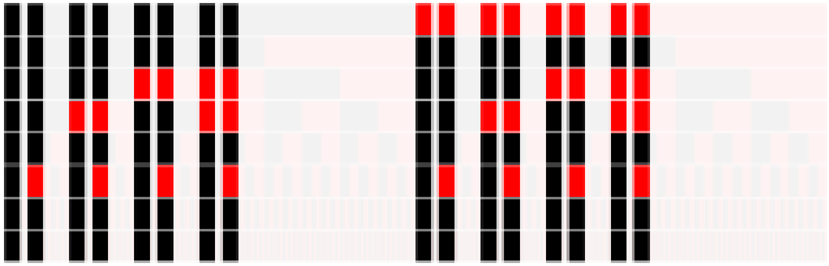
Extrasys WAP-257 *AuthFCTest*



Madwifi-ng soft AP *AuthFCTest*



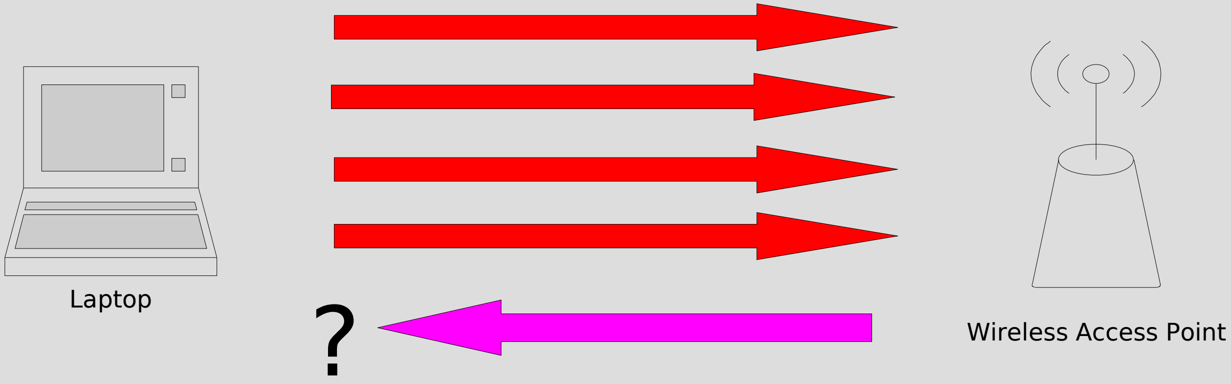
Hostap soft AP *AuthFCTest*



Aruba OpenWRT *AuthFCTest*

“Secret handshake”

- Send “gibberish” flag combinations in ProbeReq and AuthReq frames
- Watch for reactions (varying MACs helps):
- FromDS, ToDS, MoreFrags, MoreData on STA -> AP frames are all non-standard



Timing

TCP/IP L3

- Tony Capella (DC-11, '03): **Ping RTT**
  “Fashionably late – what your RTT tells ...”
- Kohno, Broido, Claffy ('05): **Clock Skew**
  “Remote physical device fingerprinting”
- Dan Kaminsky ('05): **IP frag time-outs**

802.11 L2

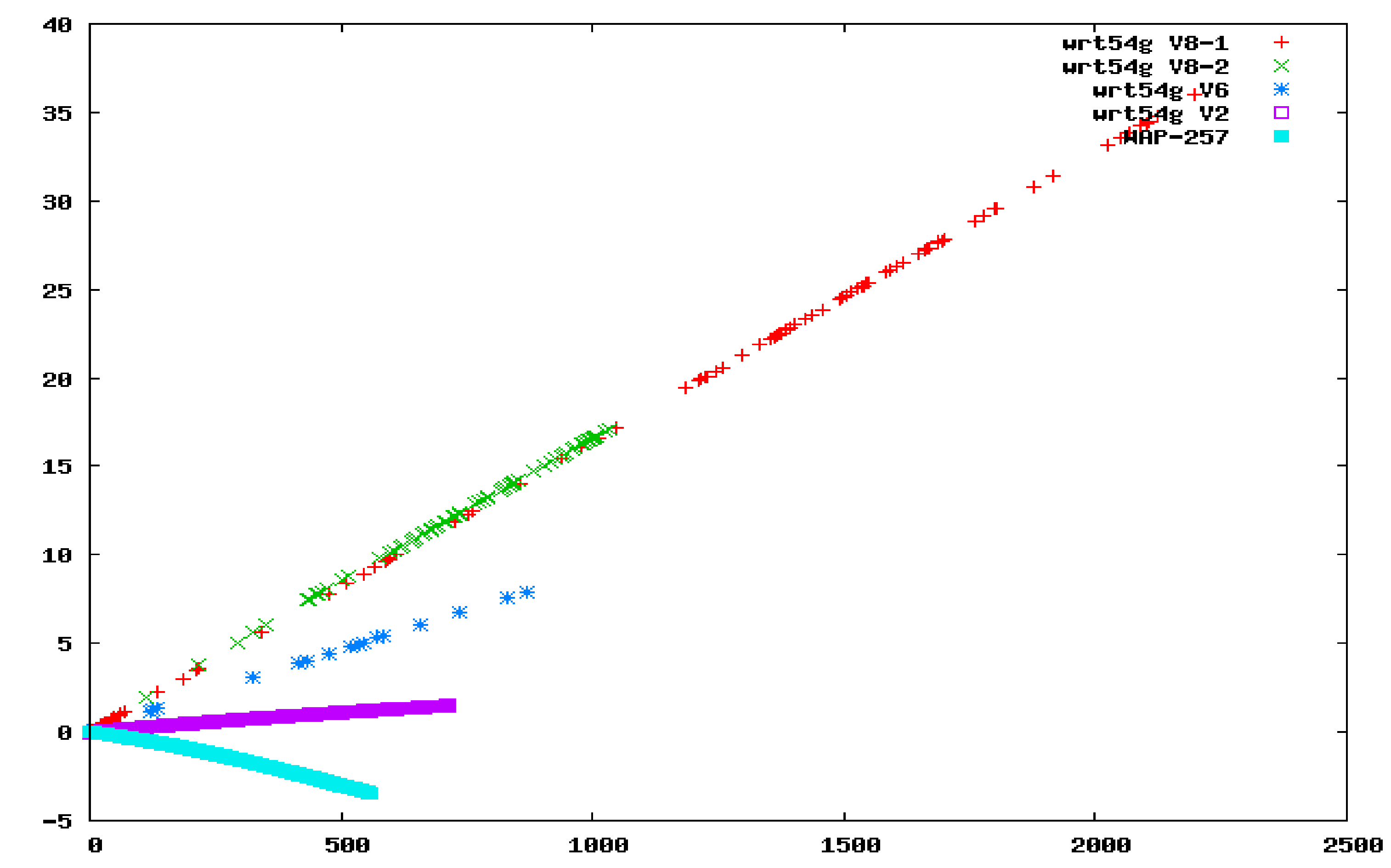
- Johnny Cache (Uninformed.org 5, '06):
  Statistical analysis of the **duration field**
- Franklin et al (USENIX Sec, '06): **Scanning**
  Time intervals between Probe Req frames

AP beacon clock skew

- Beacon frames contain AP clock's timestamp
- Each HW clock drift differently; **skew** is the derivative of the clock's offsets against another clock *(cf. Kohno, Broido, Claffy '05)*
- Issues:
  - AP clock's unique skew can be estimated reliably within 1-2 mins
  - Similar AP models have closer skews
  - Faking (e.g., with a laptop + Wi-Fi card in master mode) is hard enough

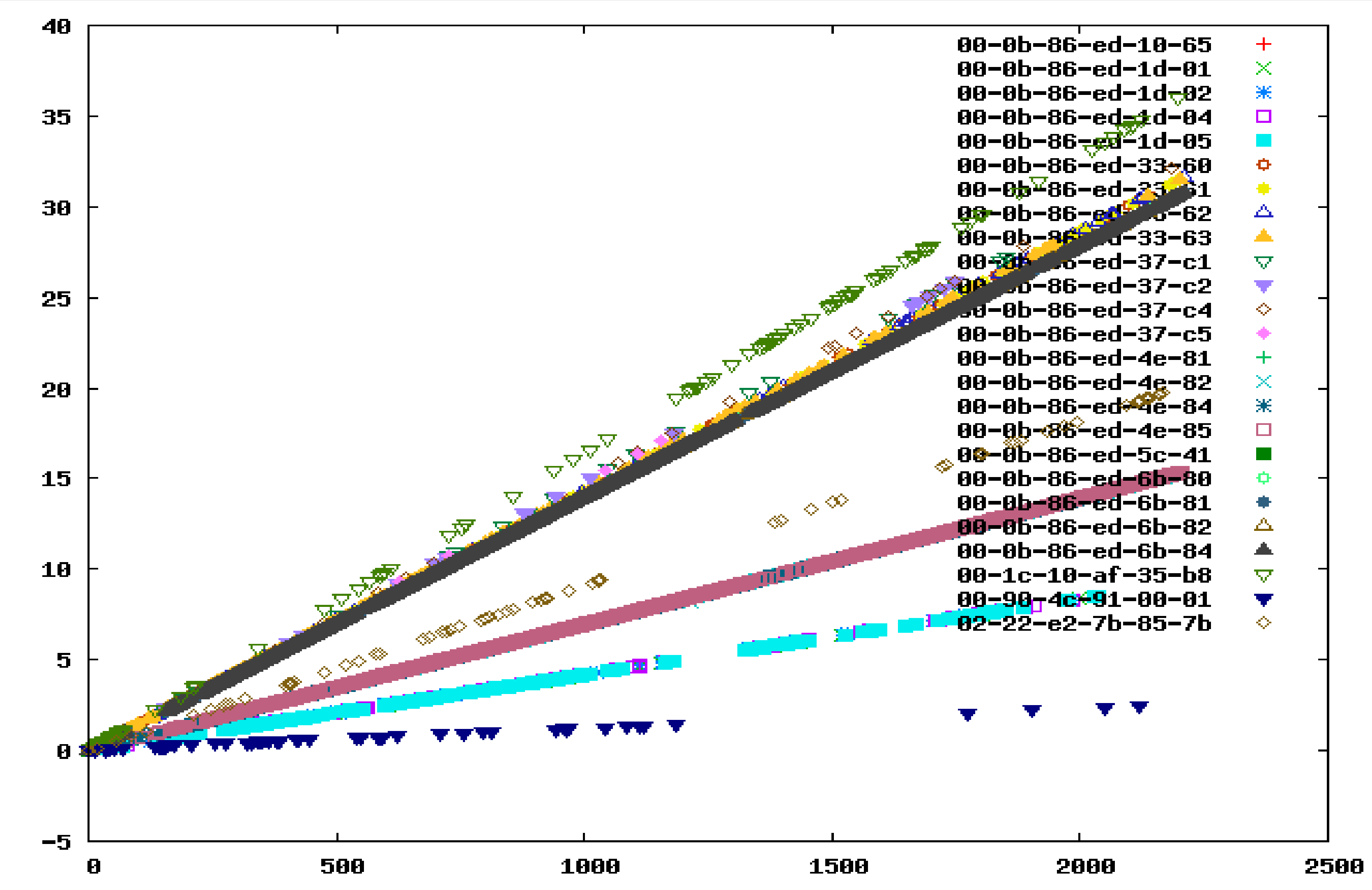
AP beacon clock skew

AP Time



Sensor Time

AP beacon clock skew



Source & Thanks

http://baffle.cs.dartmouth.edu/

- Johnny Cache for many inspirations
- Joshua Wright and Mike Kershaw for LORCON
- ToorCon & Uninformed.org
- Everyone else who helped (including authors of madwifi*, Metasploit, Ruby, Lapack and many other great tools)

Contact Information

Institute for Security Technology Studies
Dartmouth College
6211 Sudikoff Laboratory
Hanover, NH 03755

Phone: 603.646.0700
Fax: 603.646.1672

Email: info@ists.dartmouth.edu