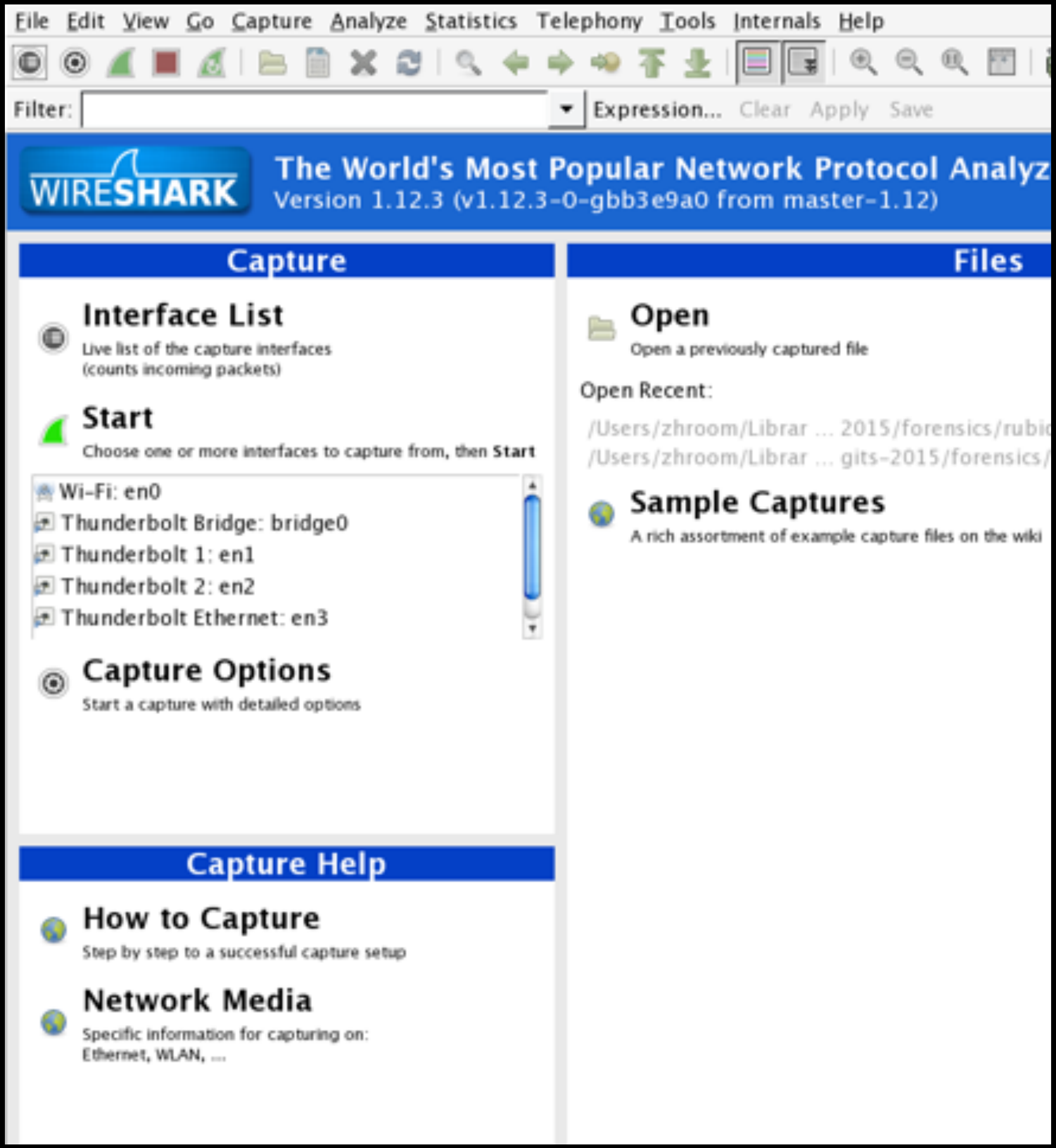
Learning Wireshark through CTF

Chad Rikansrud
@zhroom_42

Why Wireshark?

- GUI Application (simple initial learning curve)
- Intuitive for novice, full-featured for expert
- Essential for forensics, securing networks, pentesters
- CTF swiss army knife for network challenges
- Watching real-time captures


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
Clear
Apply
Save
WIRESHARK
Capture
Files
Capture Help

The World's Most Popular Network Protocol Analyz
Version 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)

Interface List
Live list of the capture interfaces
(counts incoming packets)

Start
Choose one or more interfaces to capture from, then **Start**

Wi-Fi: en0
Thunderbolt Bridge: bridge0
Thunderbolt 1: en1
Thunderbolt 2: en2
Thunderbolt Ethernet: en3

Capture Options
Start a capture with detailed options

How to Capture
Step by step to a successful capture setup

Network Media
Specific information for capturing on:
Ethernet, WLAN, ...

Open
Open a previously captured file

Open Recent:
/Users/zhroom/Librar ... 2015/forensics/rubic
/Users/zhroom/Librar ... gits-2015/forensics/

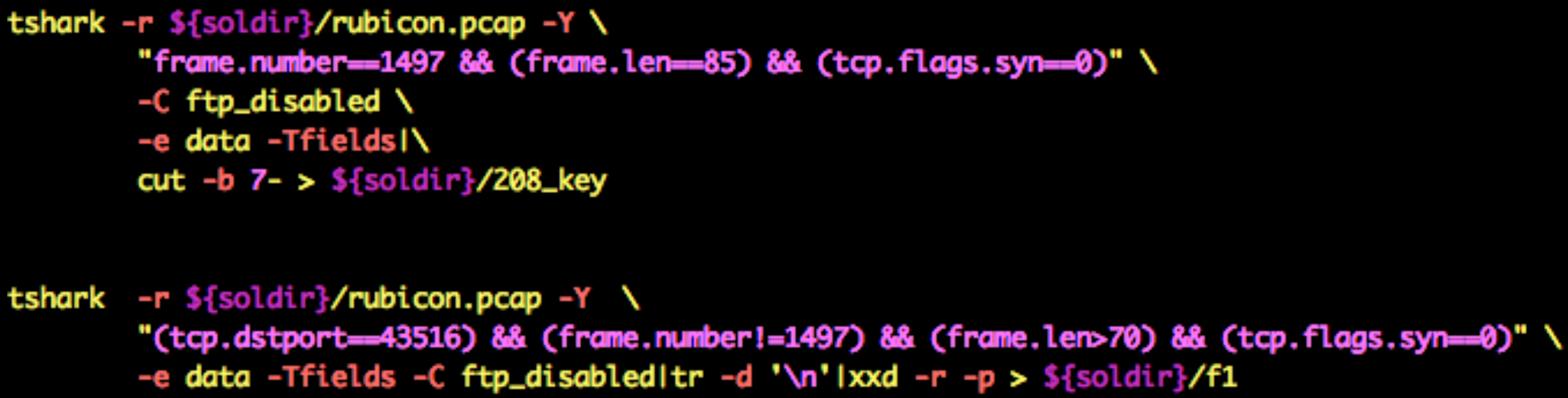
Sample Captures
A rich assortment of example capture files on the wiki

What is it great for?

- Statistical analysis (big picture) overview of network traffic
- Deep inspection of packet bytes
- Script-able and inherent search / filter functions
- Reduction of large PCAPs into more manageable chunks
- Decryption of SSL/TLS & WPA (wireless) traffic

When to use a different tool?

- Mass carving (or carving outside simple TCP stream)
- Carving via bash / python scripts (use tshark / tcpdump)
- Real time captures over days or on headless systems



```
tshark -r ${soldir}/rubicon.pcap -Y \
        "frame.number==1497 && (frame.len==85) && (tcp.flags.syn==0)" \
        -C ftp_disabled \
        -e data -Tfields|\
        cut -b 7- > ${soldir}/208_key


tshark  -r ${soldir}/rubicon.pcap -Y  \
        "(tcp.dstport==43516) && (frame.number!=1497) && (frame.len>70) && (tcp.flags.syn==0)" \
        -e data -Tfields -C ftp_disabled|tr -d '\n'|xxd -r -p > ${soldir}/f1
```

Not so much wireshark

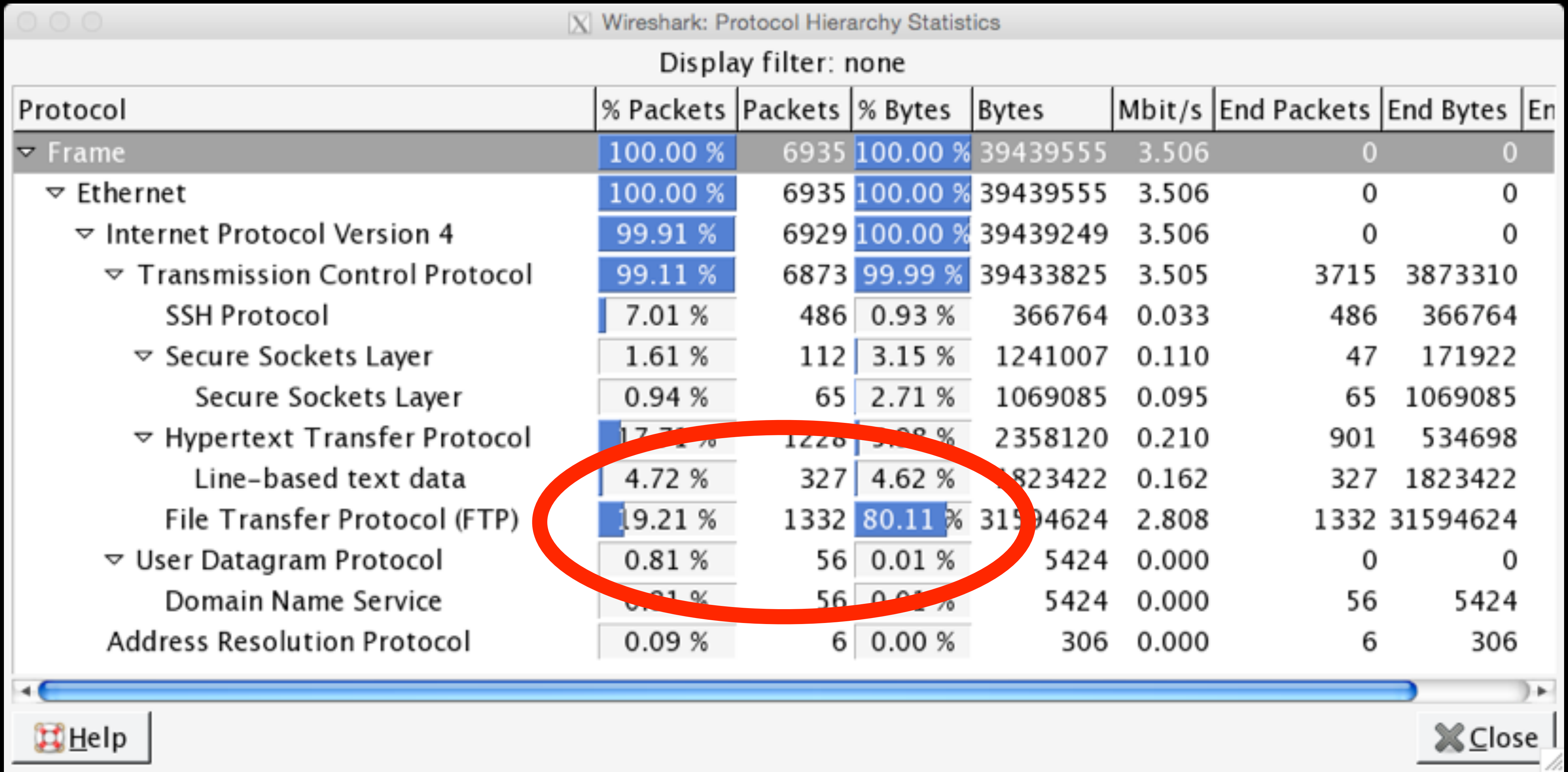
Better tools for the job

Steps to solve

- Analysis
- Carving
- Review / Repeat / Rework
- Automate (optional? no!)
- Win!

Analysis Phase

- Use a repeatable process every time you encounter a new PCAP, something like:
    - Check statistics (Protocols?)
    - Check endpoints (Noisy endpoints?)
    - Check conversations (Who is talking to each other?)
    - Download objects (What is out there?)
    - Filter, tweak, rule out easy protocols -> carve

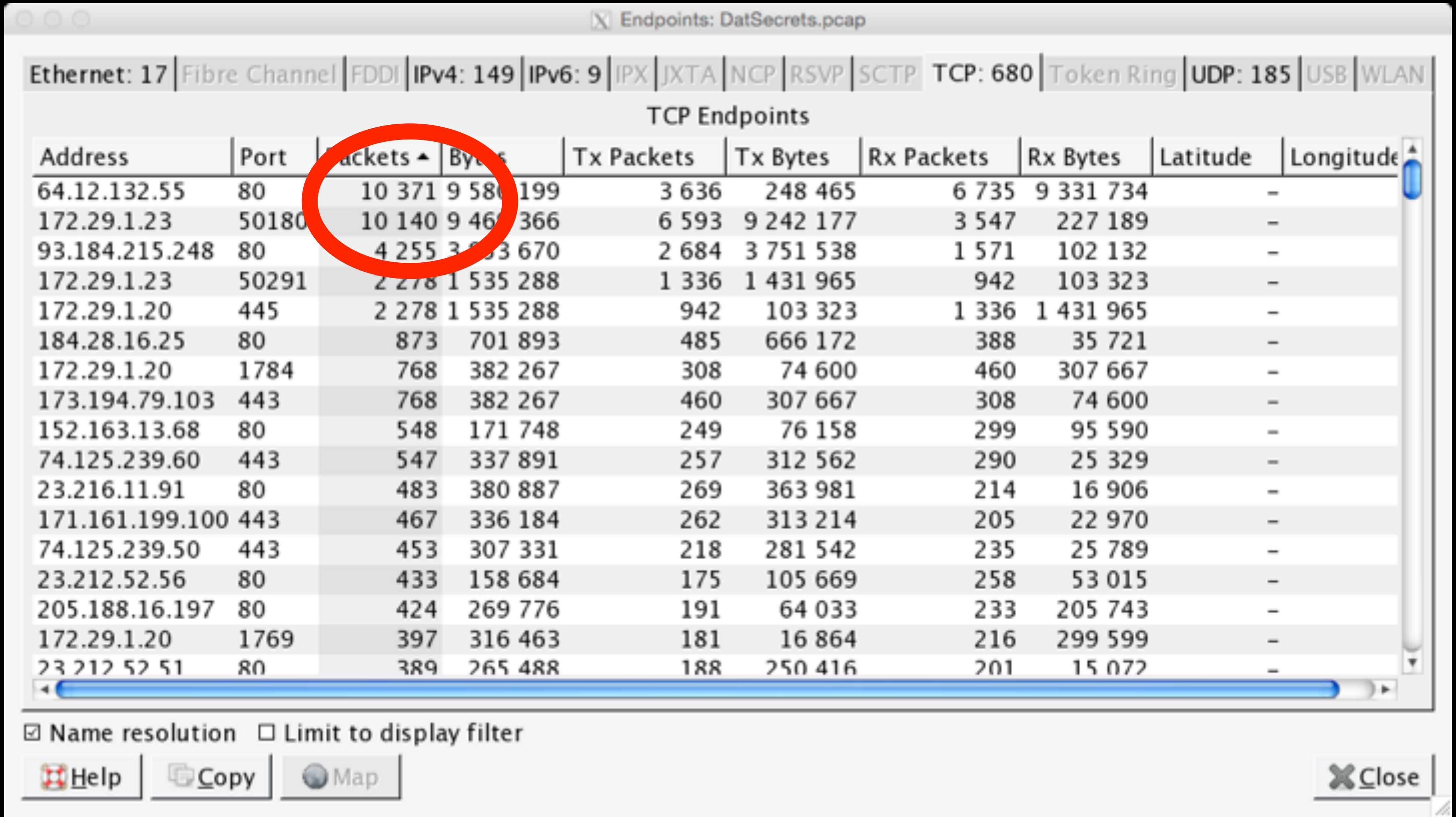


Display filter: none

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	En
Frame	100.00 %	6935	100.00 %	39439555	3.506	0	0	
Ethernet	100.00 %	6935	100.00 %	39439555	3.506	0	0	
Internet Protocol Version 4	99.91 %	6929	100.00 %	39439249	3.506	0	0	
Transmission Control Protocol	99.11 %	6873	99.99 %	39433825	3.505	3715	3873310	
SSH Protocol	7.01 %	486	0.93 %	366764	0.033	486	366764	
Secure Sockets Layer	1.61 %	112	3.15 %	1241007	0.110	47	171922	
Secure Sockets Layer	0.94 %	65	2.71 %	1069085	0.095	65	1069085	
Hypertext Transfer Protocol	17.71 %	1228	5.98 %	2358120	0.210	901	534698	
Line-based text data	4.72 %	327	4.62 %	1823422	0.162	327	1823422	
File Transfer Protocol (FTP)	19.21 %	1332	80.11 %	31594624	2.808	1332	31594624	
User Datagram Protocol	0.81 %	56	0.01 %	5424	0.000	0	0	
Domain Name Service	0.81 %	56	0.01 %	5424	0.000	56	5424	
Address Resolution Protocol	0.09 %	6	0.00 %	306	0.000	6	306	

Statistics (Heavy protocols?)

Statistics -> Protocol Hierarchy



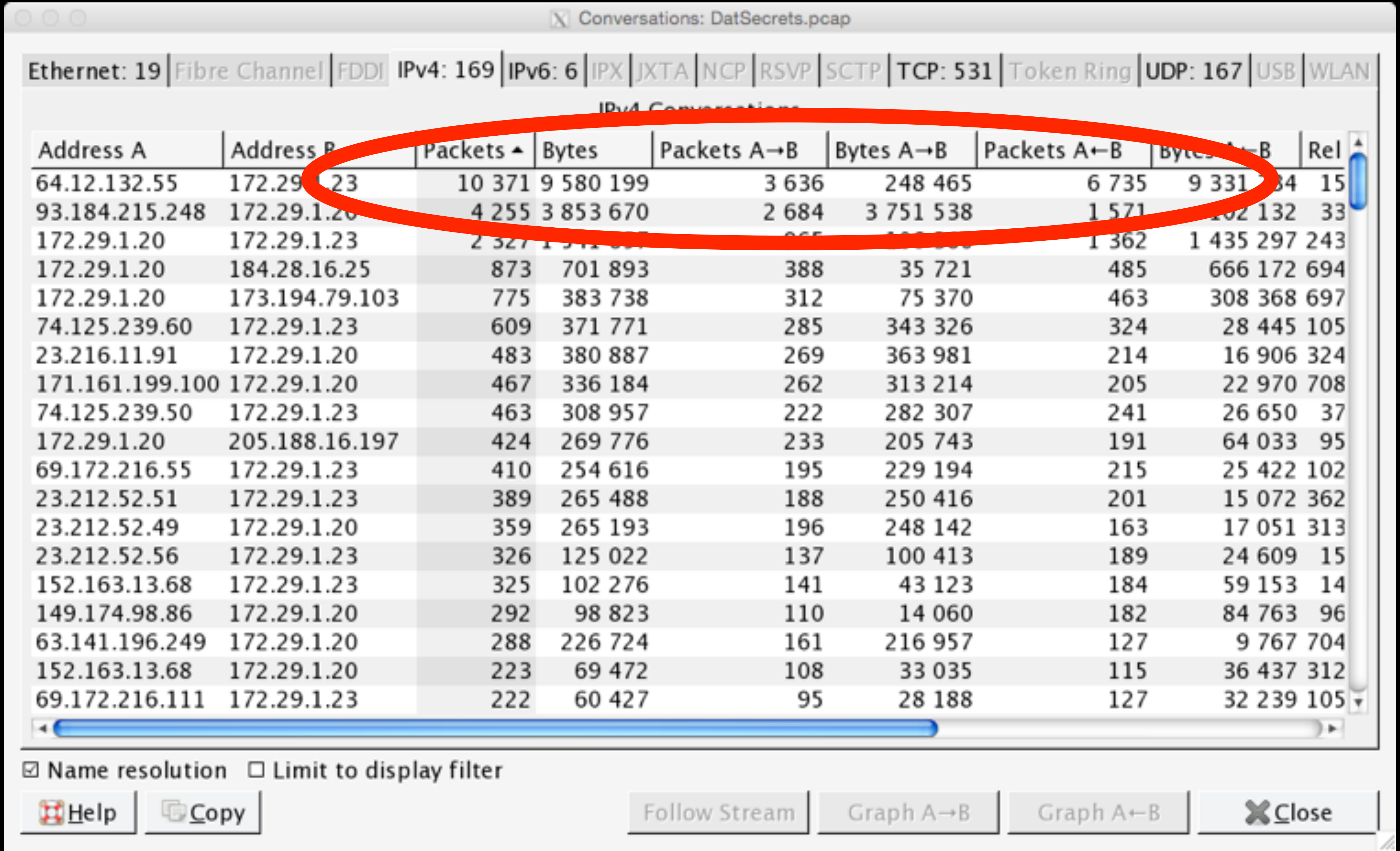
TCP Endpoints

Address	Port	Packets ▲	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Latitude	Longitude
64.12.132.55	80	10 371	9 580 199	3 636	248 465	6 735	9 331 734	–	
172.29.1.23	50180	10 140	9 469 366	6 593	9 242 177	3 547	227 189	–	
93.184.215.248	80	4 255	3 853 670	2 684	3 751 538	1 571	102 132	–	
172.29.1.23	50291	2 278	1 535 288	1 336	1 431 965	942	103 323	–	
172.29.1.20	445	2 278	1 535 288	942	103 323	1 336	1 431 965	–	
184.28.16.25	80	873	701 893	485	666 172	388	35 721	–	
172.29.1.20	1784	768	382 267	308	74 600	460	307 667	–	
173.194.79.103	443	768	382 267	460	307 667	308	74 600	–	
152.163.13.68	80	548	171 748	249	76 158	299	95 590	–	
74.125.239.60	443	547	337 891	257	312 562	290	25 329	–	
23.216.11.91	80	483	380 887	269	363 981	214	16 906	–	
171.161.199.100	443	467	336 184	262	313 214	205	22 970	–	
74.125.239.50	443	453	307 331	218	281 542	235	25 789	–	
23.212.52.56	80	433	158 684	175	105 669	258	53 015	–	
205.188.16.197	80	424	269 776	191	64 033	233	205 743	–	
172.29.1.20	1769	397	316 463	181	16 864	216	299 599	–	
23.212.52.51	80	389	265 488	188	250 416	201	15 072	–	

☑ Name resolution ☐ Limit to display filter

Endpoints (Where is the noise?)

Statistics -> Endpoints



Conversations: DatSecrets.pcap

Ethernet: 19 | Fibre Channel | FDDI | IPv4: 169 | IPv6: 6 | IPX | JXTA | NCP | RSVP | SCTP | TCP: 531 | Token Ring | UDP: 167 | USB | WLAN

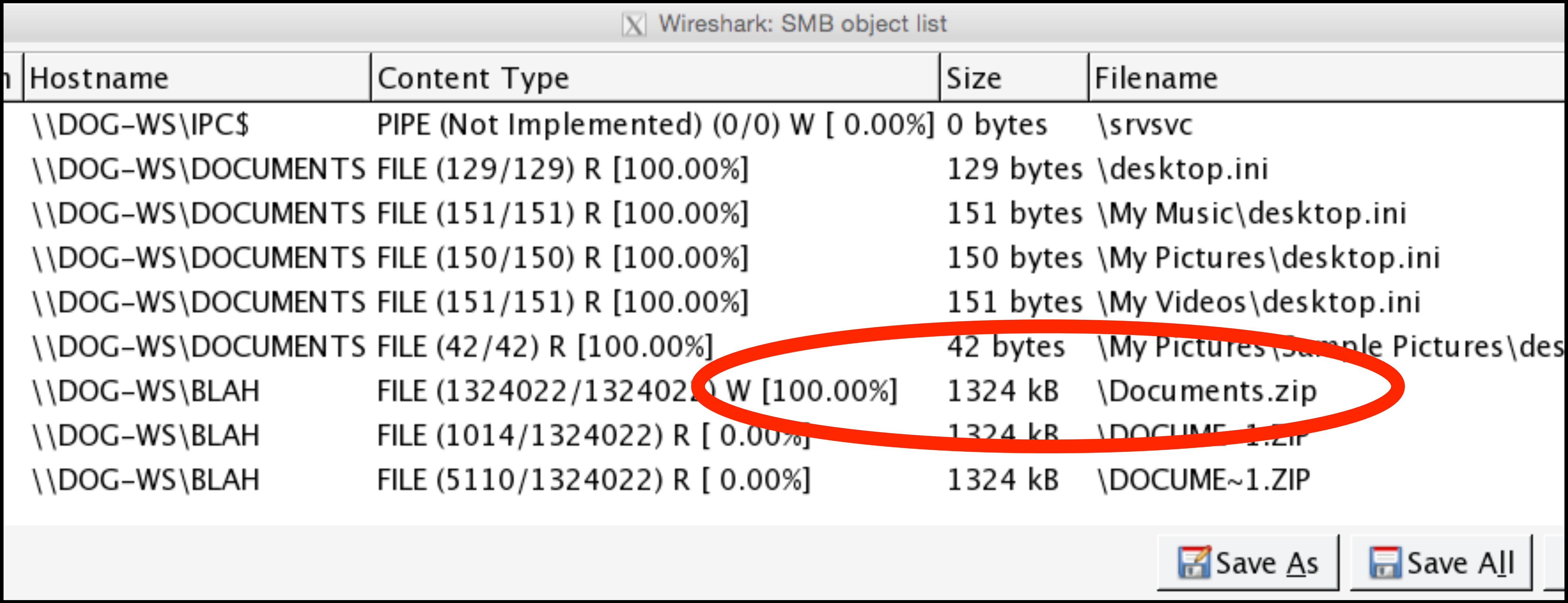
IPv4 Conversations

Address A	Address B	Packets ▲	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B	Rel
64.12.132.55	172.29.1.23	10 371	9 580 199	3 636	248 465	6 735	9 331 [illegible]34	15
93.184.215.248	172.29.1.20	4 255	3 853 670	2 684	3 751 538	1 571	102 132	33
172.29.1.20	172.29.1.23	2 327	[illegible]	[illegible]	[illegible]	1 362	1 435 297	243
172.29.1.20	184.28.16.25	873	701 893	388	35 721	485	666 172	694
172.29.1.20	173.194.79.103	775	383 738	312	75 370	463	308 368	697
74.125.239.60	172.29.1.23	609	371 771	285	343 326	324	28 445	105
23.216.11.91	172.29.1.20	483	380 887	269	363 981	214	16 906	324
171.161.199.100	172.29.1.20	467	336 184	262	313 214	205	22 970	708
74.125.239.50	172.29.1.23	463	308 957	222	282 307	241	26 650	37
172.29.1.20	205.188.16.197	424	269 776	233	205 743	191	64 033	95
69.172.216.55	172.29.1.23	410	254 616	195	229 194	215	25 422	102
23.212.52.51	172.29.1.23	389	265 488	188	250 416	201	15 072	362
23.212.52.49	172.29.1.20	359	265 193	196	248 142	163	17 051	313
23.212.52.56	172.29.1.23	326	125 022	137	100 413	189	24 609	15
152.163.13.68	172.29.1.23	325	102 276	141	43 123	184	59 153	14
149.174.98.86	172.29.1.20	292	98 823	110	14 060	182	84 763	96
63.141.196.249	172.29.1.20	288	226 724	161	216 957	127	9 767	704
152.163.13.68	172.29.1.20	223	69 472	108	33 035	115	36 437	312
69.172.216.111	172.29.1.23	222	60 427	95	28 188	127	32 239	105

☑ Name resolution ☐ Limit to display filter

Conversations - Who’s talking to each other?

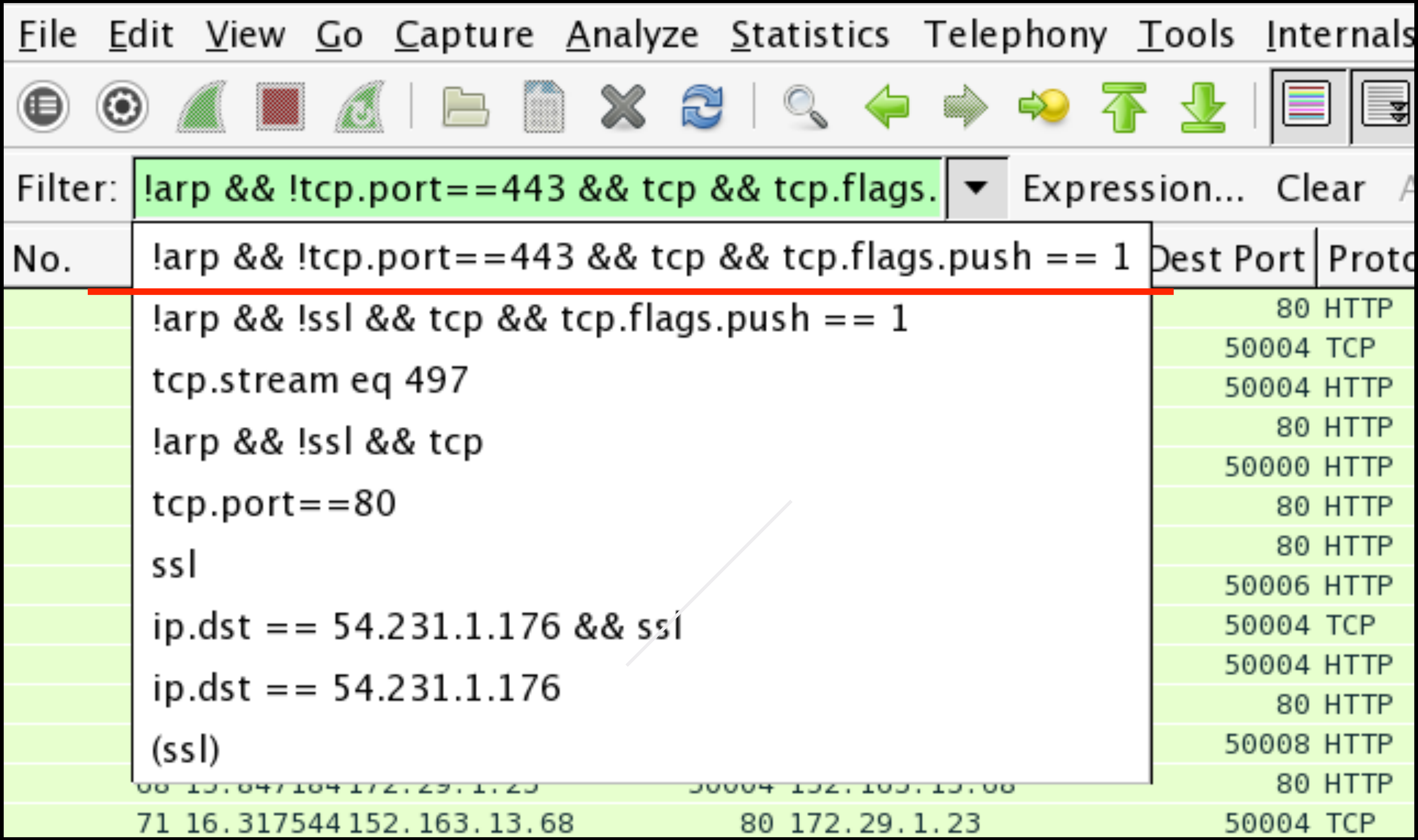
Statistics -> Conversations



Hostname	Content Type	Size	Filename
\\DOG-WS\IPC$	PIPE (Not Implemented) (0/0) W [0.00%]	0 bytes	\srvsvc
\\DOG-WS\DOCUMENTS	FILE (129/129) R [100.00%]	129 bytes	\desktop.ini
\\DOG-WS\DOCUMENTS	FILE (151/151) R [100.00%]	151 bytes	\My Music\desktop.ini
\\DOG-WS\DOCUMENTS	FILE (150/150) R [100.00%]	150 bytes	\My Pictures\desktop.ini
\\DOG-WS\DOCUMENTS	FILE (151/151) R [100.00%]	151 bytes	\My Videos\desktop.ini
\\DOG-WS\DOCUMENTS	FILE (42/42) R [100.00%]	42 bytes	\My Pictures\Sample Pictures\des
\\DOG-WS\BLAH	FILE (1324022/1324022) W [100.00%]	1324 kB	\Documents.zip
\\DOG-WS\BLAH	FILE (1014/1324022) R [0.00%]	1324 kB	\DOCUME~1.ZIP
\\DOG-WS\BLAH	FILE (5110/1324022) R [0.00%]	1324 kB	\DOCUME~1.ZIP

Download objects - Easy win!

File -> Export Objects -> HTTP or DICOM or SMB



```
!arp && !tcp.port==443 && tcp && tcp.flags.
```

```
!arp && !tcp.port==443 && tcp && tcp.flags.push == 1
!arp && !ssl && tcp && tcp.flags.push == 1
tcp.stream eq 497
!arp && !ssl && tcp
tcp.port==80
ssl
ip.dst == 54.231.1.176 && ssl
ip.dst == 54.231.1.176
(ssl)
```

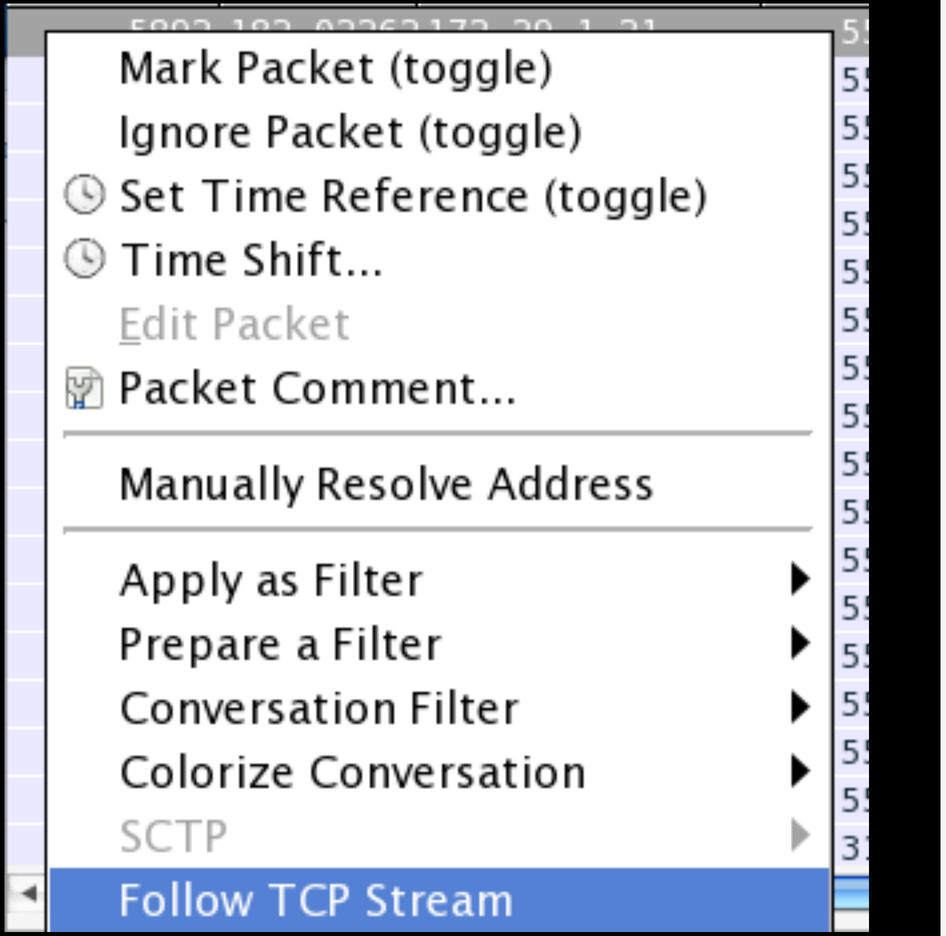
```
80 HTTP
50004 TCP
50004 HTTP
80 HTTP
50000 HTTP
80 HTTP
80 HTTP
50006 HTTP
50004 TCP
50004 HTTP
80 HTTP
50008 HTTP
80 HTTP
71 16.317544 152.163.13.68        80 172.29.1.23        50004 TCP
```

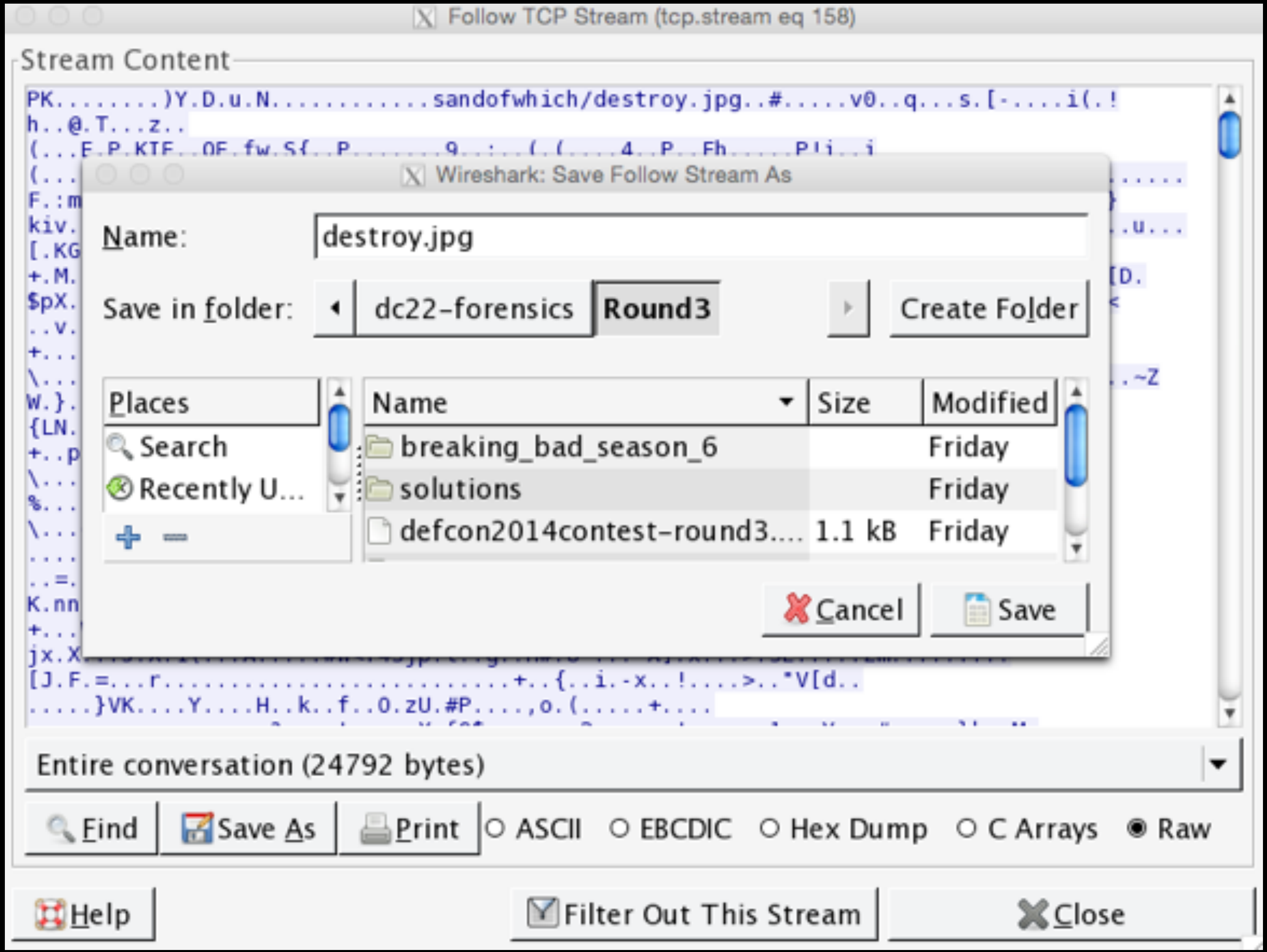
Tweak, filter and eliminate

Know your display filter syntax!
(note* **not** the same as libpcap syntax)

Carving Phase

- Easy
  - wireshark export object
  - wireshark save as binary
- Harder
  - tshark / tcpdump extract
- Hardest
  - Combination of multiple tools
  - Custom coding (know scripting lang + CLI utils!)





```
PK.......)Y.D.u.N...........sandofwhich/destroy.jpg..#.....v0..q...s.[-.....i(.!
h..@.T...z..
[J.F.=...r........................+..{..i.-x..!....>..*V[d..
.....}VK....Y....H..k..f..0.zU.#P....,o.(.....+....
```

Name: destroy.jpg

Save in folder: dc22-forensics Round3

Name	Size	Modified
breaking_bad_season_6		Friday
solutions		Friday
defcon2014contest-round3....	1.1 kB	Friday

Entire conversation (24792 bytes)

○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

TCP Stream saving binary

Simple extract of files from known protocols (or protocols used correctly)

Demos

Test / RRR Phase

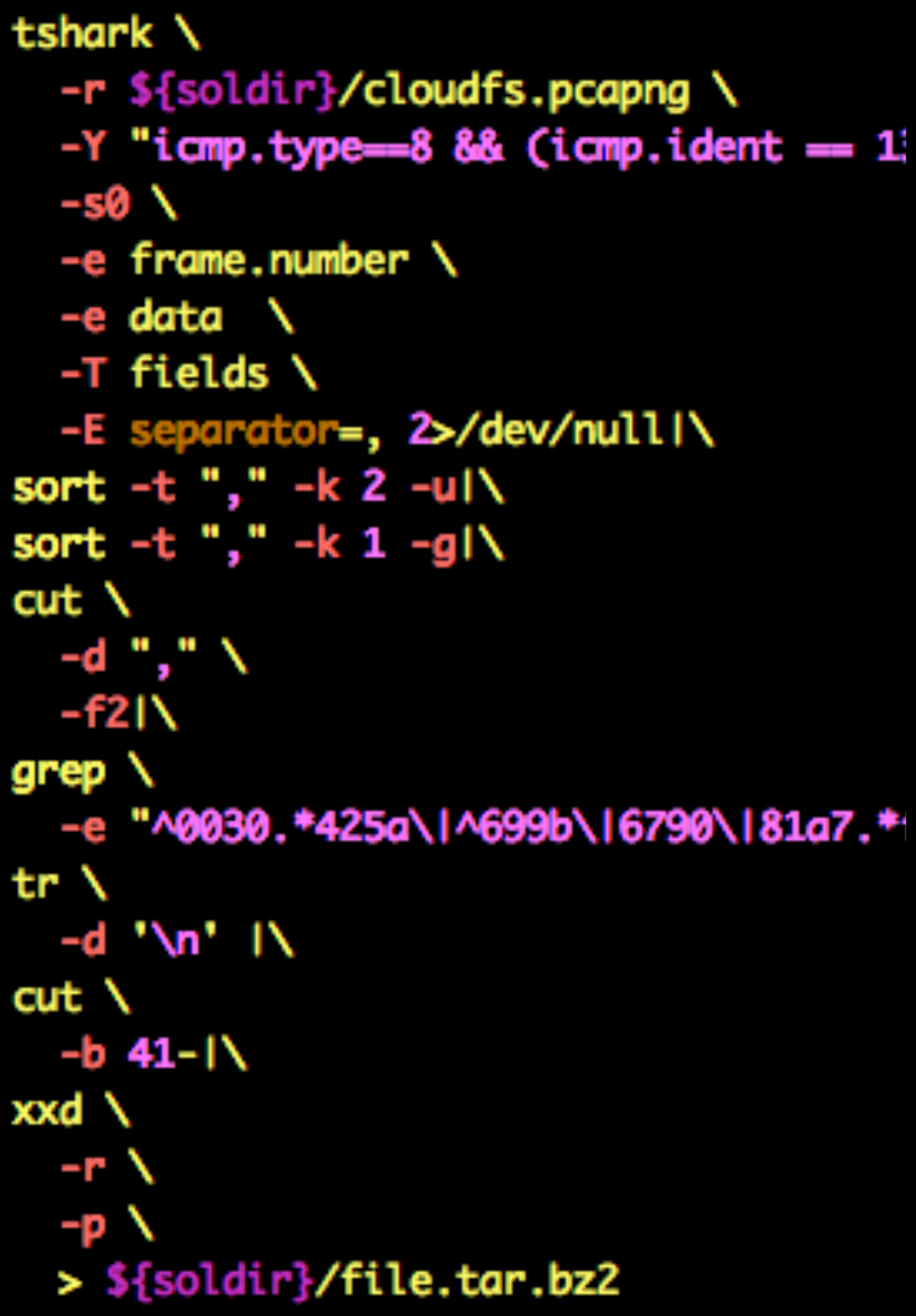
- Did we get an obvious win?
- Does it match the spirit/letter of what was asked?
- Possible red herrings (they’d NEVER do that!)
- Penalty for checking?

Automate Phase

- Why go through the extra work to automate?
  - After the party, when the CTF is over - you will not remember.
  - Reuse Reuse Ruse
  - Demo to your friends and local DC group!

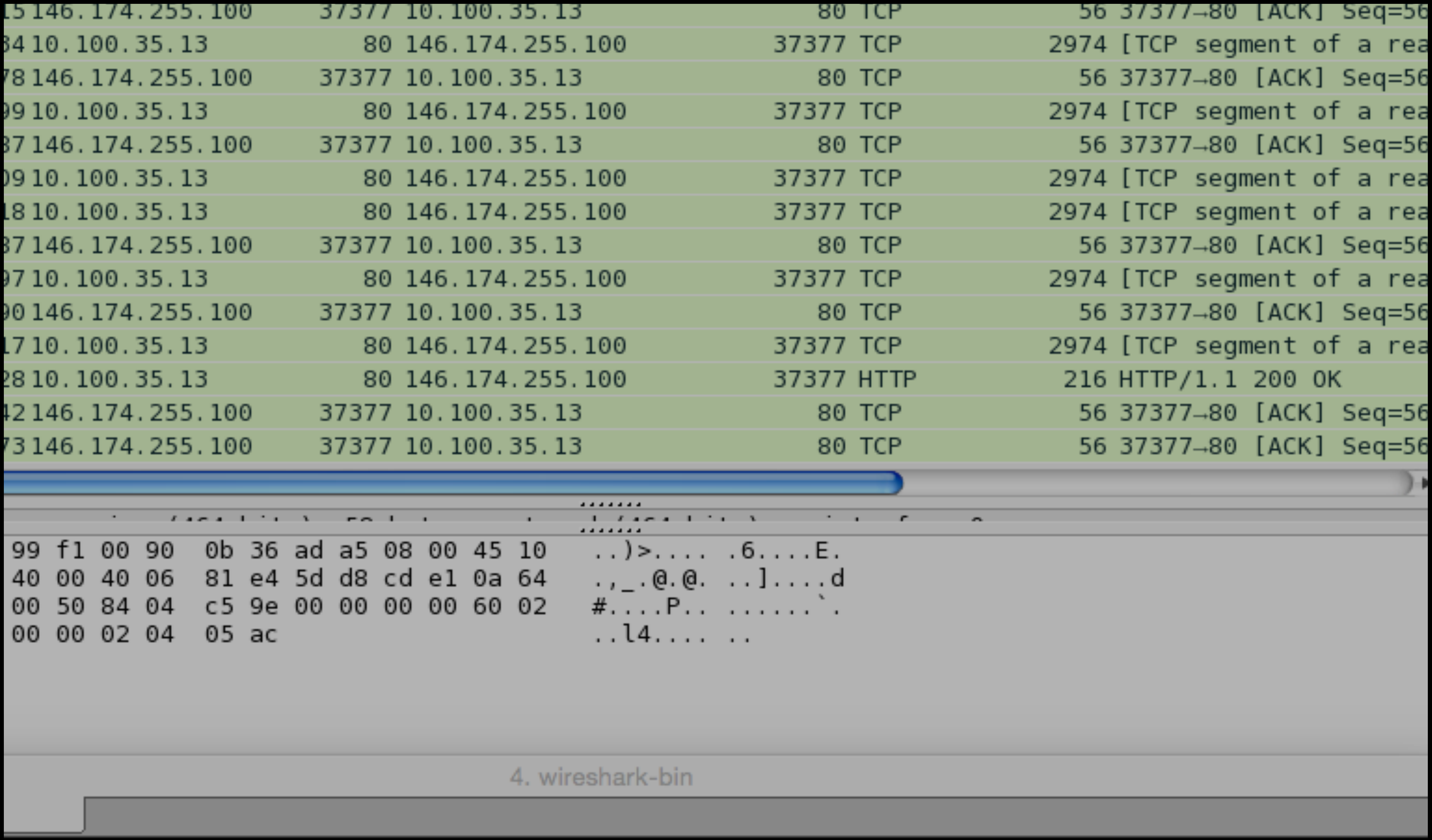
Script example

This file carves out a bz2 binary from the pcap, which uses a home-rolled protocol



```
tshark \
  -r ${soldir}/cloudfs.pcapng \
  -Y "icmp.type==8 && (icmp.ident == 1
  -s0 \
  -e frame.number \
  -e data  \
  -T fields \
  -E separator=, 2>/dev/null|\
sort -t "," -k 2 -u|\
sort -t "," -k 1 -g|\
cut \
  -d "," \
  -f2|\
grep \
  -e "^0030.*425a\|^699b\|6790\|81a7.*
tr \
  -d '\n' |\
cut \
  -b 41-|\
xxd \
  -r \
  -p \
  > ${soldir}/file.tar.bz2
```

Tips, Tricks, Appendix



```
15 146.174.255.100   37377 10.100.35.13         80 TCP    56 37377→80 [ACK] Seq=56
34 10.100.35.13         80 146.174.255.100   37377 TCP  2974 [TCP segment of a rea
78 146.174.255.100   37377 10.100.35.13         80 TCP    56 37377→80 [ACK] Seq=56
99 10.100.35.13         80 146.174.255.100   37377 TCP  2974 [TCP segment of a rea
37 146.174.255.100   37377 10.100.35.13         80 TCP    56 37377→80 [ACK] Seq=56
09 10.100.35.13         80 146.174.255.100   37377 TCP  2974 [TCP segment of a rea
18 10.100.35.13         80 146.174.255.100   37377 TCP  2974 [TCP segment of a rea
37 146.174.255.100   37377 10.100.35.13         80 TCP    56 37377→80 [ACK] Seq=56
97 10.100.35.13         80 146.174.255.100   37377 TCP  2974 [TCP segment of a rea
90 146.174.255.100   37377 10.100.35.13         80 TCP    56 37377→80 [ACK] Seq=56
17 10.100.35.13         80 146.174.255.100   37377 TCP  2974 [TCP segment of a rea
28 10.100.35.13         80 146.174.255.100   37377 HTTP  216 HTTP/1.1 200 OK
42 146.174.255.100   37377 10.100.35.13         80 TCP    56 37377→80 [ACK] Seq=56
73 146.174.255.100   37377 10.100.35.13         80 TCP    56 37377→80 [ACK] Seq=56
```

```
99 f1 00 90  0b 36 ad a5 08 00 45 10   ..)>.... .6....E.
40 00 40 06  81 e4 5d d8 cd e1 0a 64   .,_.@.@. ..]....d
00 50 84 04  c5 9e 00 00 00 00 60 02   #....P.. ......`.
00 00 02 04  05 ac                     ..l4.... ..
```

```
ssh udesktop "tcpdump -i internal.v3500 -s0 -w - 'not port 22'"|wireshark  -B 5 -s0 -i- -k >/dev/null 2>&1
```

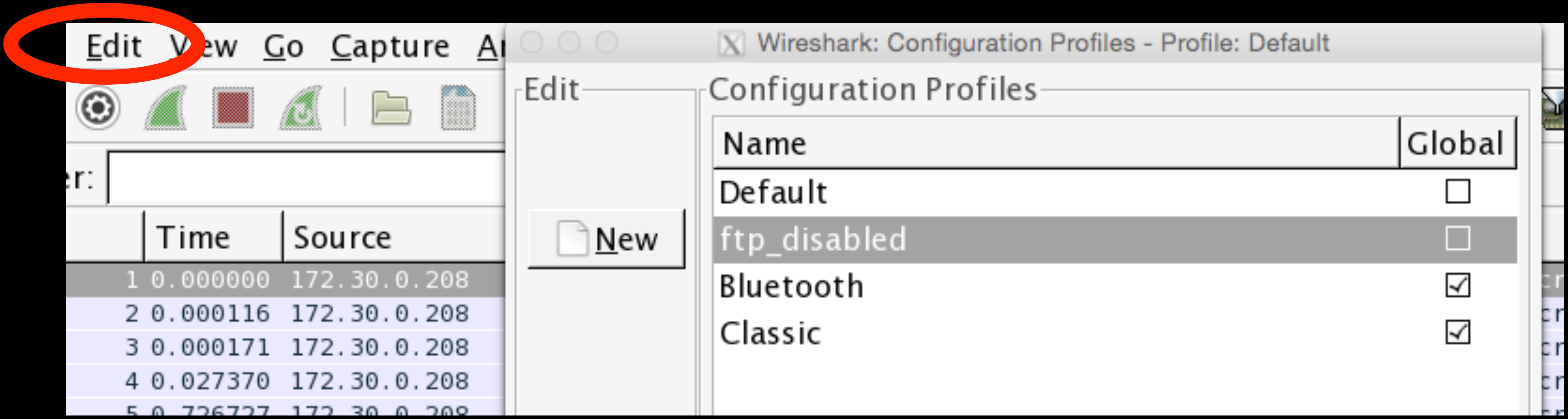
Remote Capture

Dump traffic from a remote host over ssh

SSL Decrypt

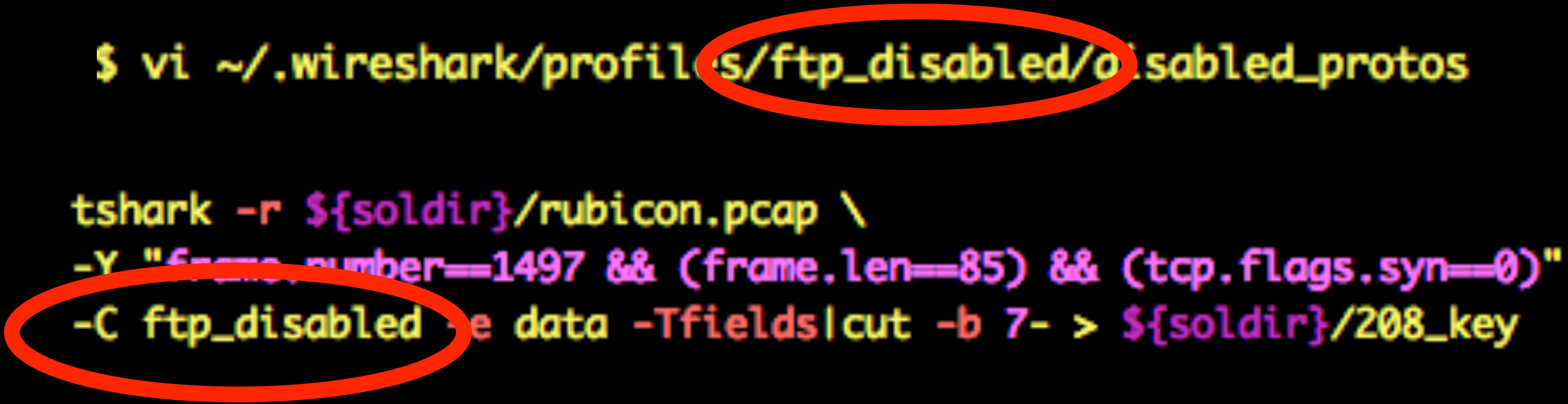
Demo

Custom Configurations



	Time	Source
1	0.000000	172.30.0.208
2	0.000116	172.30.0.208
3	0.000171	172.30.0.208
4	0.027370	172.30.0.208
5	0.726727	172.30.0.208

Name	Global
Default	□
ftp_disabled	□
Bluetooth	☑
Classic	☑



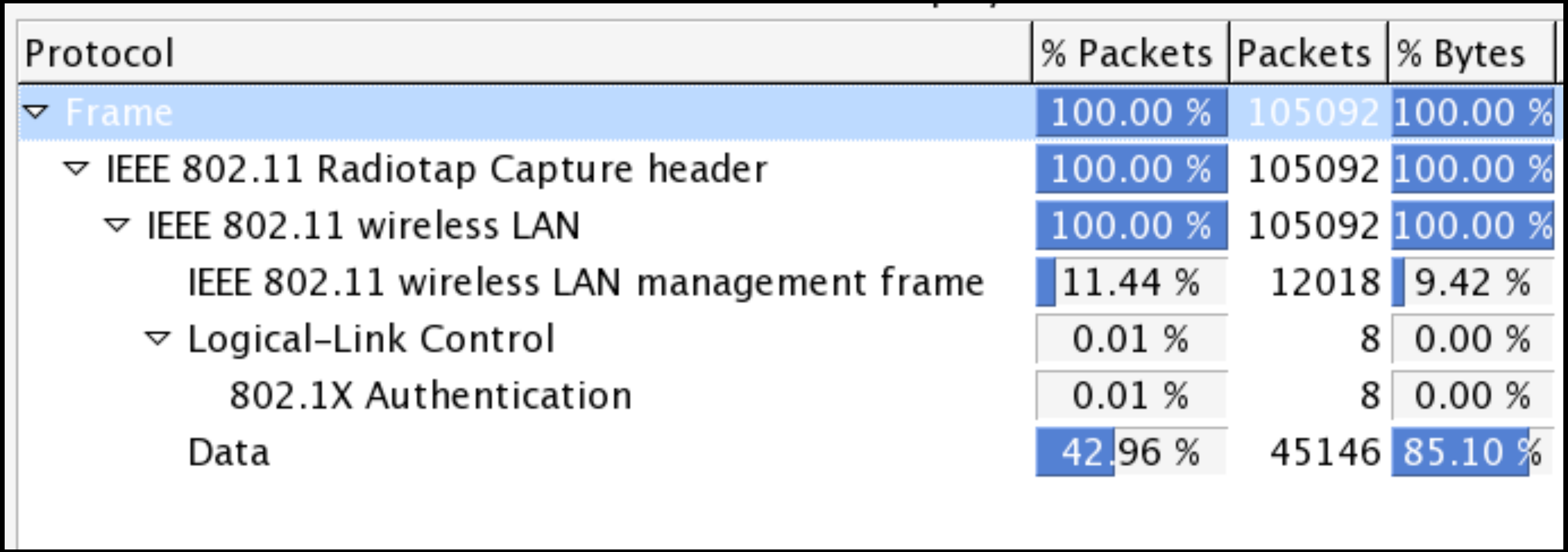
```
$ vi ~/.wireshark/profiles/ftp_disabled/disabled_protos
```

```
tshark -r ${soldir}/rubicon.pcap \
-Y "frame.number==1497 && (frame.len==85) && (tcp.flags.syn==0)"
-C ftp_disabled -e data -Tfields|cut -b 7- > ${soldir}/208_key
```

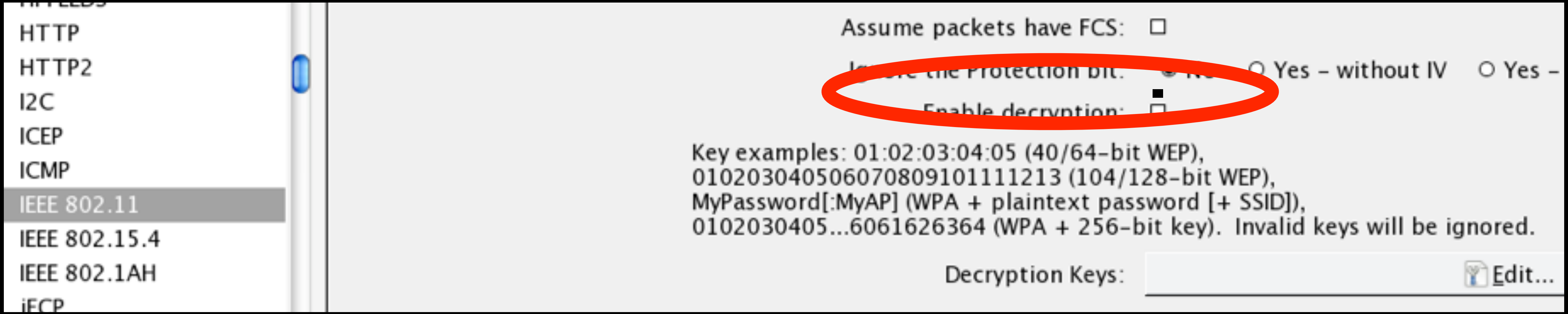
WPA Decrypt

- Capture Raw 802.11 traffic
- Know the WPA Shared key, passphrase, or WEP keys
- Wireshark does the hard work!

WPA Capture Before



Protocol	% Packets	Packets	% Bytes
▽ Frame	100.00 %	105092	100.00 %
▽ IEEE 802.11 Radiotap Capture header	100.00 %	105092	100.00 %
▽ IEEE 802.11 wireless LAN	100.00 %	105092	100.00 %
IEEE 802.11 wireless LAN management frame	11.44 %	12018	9.42 %
▽ Logical-Link Control	0.01 %	8	0.00 %
802.1X Authentication	0.01 %	8	0.00 %
Data	42.96 %	45146	85.10 %



HTTP
HTTP2
I2C
ICEP
ICMP
IEEE 802.11
IEEE 802.15.4
IEEE 802.1AH
iECP

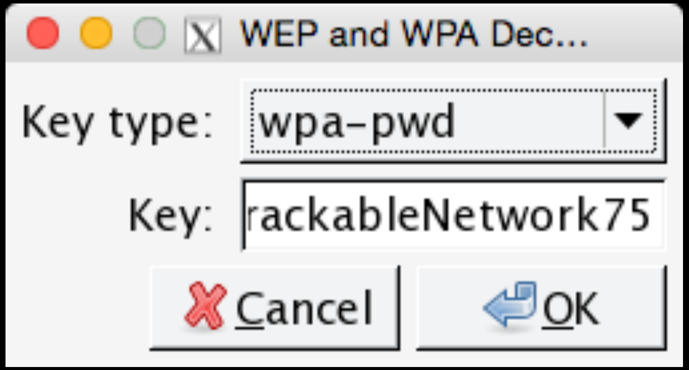
Assume packets have FCS: □

Ignore the Protection bit: ● No ○ Yes – without IV ○ Yes –

Enable decryption: □

Key examples: 01:02:03:04:05 (40/64-bit WEP),
0102030405060708091011111213 (104/128-bit WEP),
MyPassword[:MyAP] (WPA + plaintext password [+ SSID]),
0102030405...6061626364 (WPA + 256-bit key). Invalid keys will be ignored.

Decryption Keys: Edit...

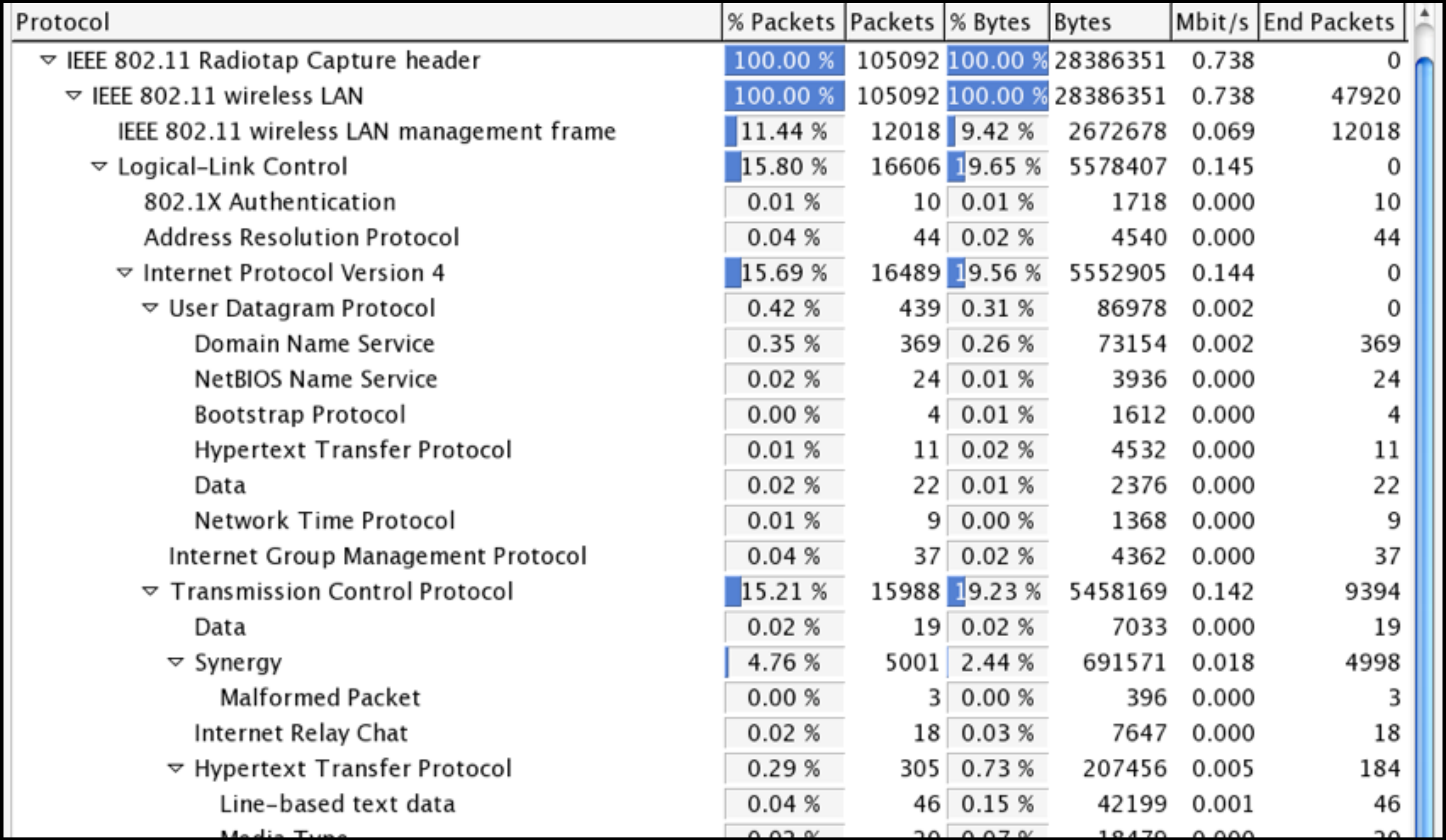


WEP and WPA Dec...

Key type: wpa-pwd

Key: rackableNetwork75

WPA Capture After!



Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets
▽ IEEE 802.11 Radiotap Capture header	100.00 %	105092	100.00 %	28386351	0.738	0
▽ IEEE 802.11 wireless LAN	100.00 %	105092	100.00 %	28386351	0.738	47920
IEEE 802.11 wireless LAN management frame	11.44 %	12018	9.42 %	2672678	0.069	12018
▽ Logical-Link Control	15.80 %	16606	19.65 %	5578407	0.145	0
802.1X Authentication	0.01 %	10	0.01 %	1718	0.000	10
Address Resolution Protocol	0.04 %	44	0.02 %	4540	0.000	44
▽ Internet Protocol Version 4	15.69 %	16489	19.56 %	5552905	0.144	0
▽ User Datagram Protocol	0.42 %	439	0.31 %	86978	0.002	0
Domain Name Service	0.35 %	369	0.26 %	73154	0.002	369
NetBIOS Name Service	0.02 %	24	0.01 %	3936	0.000	24
Bootstrap Protocol	0.00 %	4	0.01 %	1612	0.000	4
Hypertext Transfer Protocol	0.01 %	11	0.02 %	4532	0.000	11
Data	0.02 %	22	0.01 %	2376	0.000	22
Network Time Protocol	0.01 %	9	0.00 %	1368	0.000	9
Internet Group Management Protocol	0.04 %	37	0.02 %	4362	0.000	37
▽ Transmission Control Protocol	15.21 %	15988	19.23 %	5458169	0.142	9394
Data	0.02 %	19	0.02 %	7033	0.000	19
▽ Synergy	4.76 %	5001	2.44 %	691571	0.018	4998
Malformed Packet	0.00 %	3	0.00 %	396	0.000	3
Internet Relay Chat	0.02 %	18	0.03 %	7647	0.000	18
▽ Hypertext Transfer Protocol	0.29 %	305	0.73 %	207456	0.005	184
Line-based text data	0.04 %	46	0.15 %	42199	0.001	46
Media Type	0.02 %	20	0.07 %	18470	0.000	20

Questions?

Thank you!