Packet Analysis

INFO-6081 – Monitoring & Incident Response

FANSHAWE

Learning Outcomes

- Packet Analysis
- Tcpdump
- Berkley Packet Filter
- Examining Full Content Data
- Dumpcap
- Tshark
- Wireshark
- Display filters
- Wireshark Statistics

Packet Analysis

- Packet analysis tools read network traffic from an interface or a saved capture (trace) file and present the information to an analyst
- This information is used to interpret what is occurring on the network
- Packet analysis tools are available for either the command line, or graphical interfaces

Tcpdump

- Tcpdump is a command line traffic analyzer that is available on SO
- It is often used to read the data created by netsniff-ng, and stored in **/nsm/sensor_data/*HOSTNAME-INTERFACE*/dailylogs**
- Tcpdump is a protocol analyzer, as it interpret and decode the protocols used at various layers of the OSI model

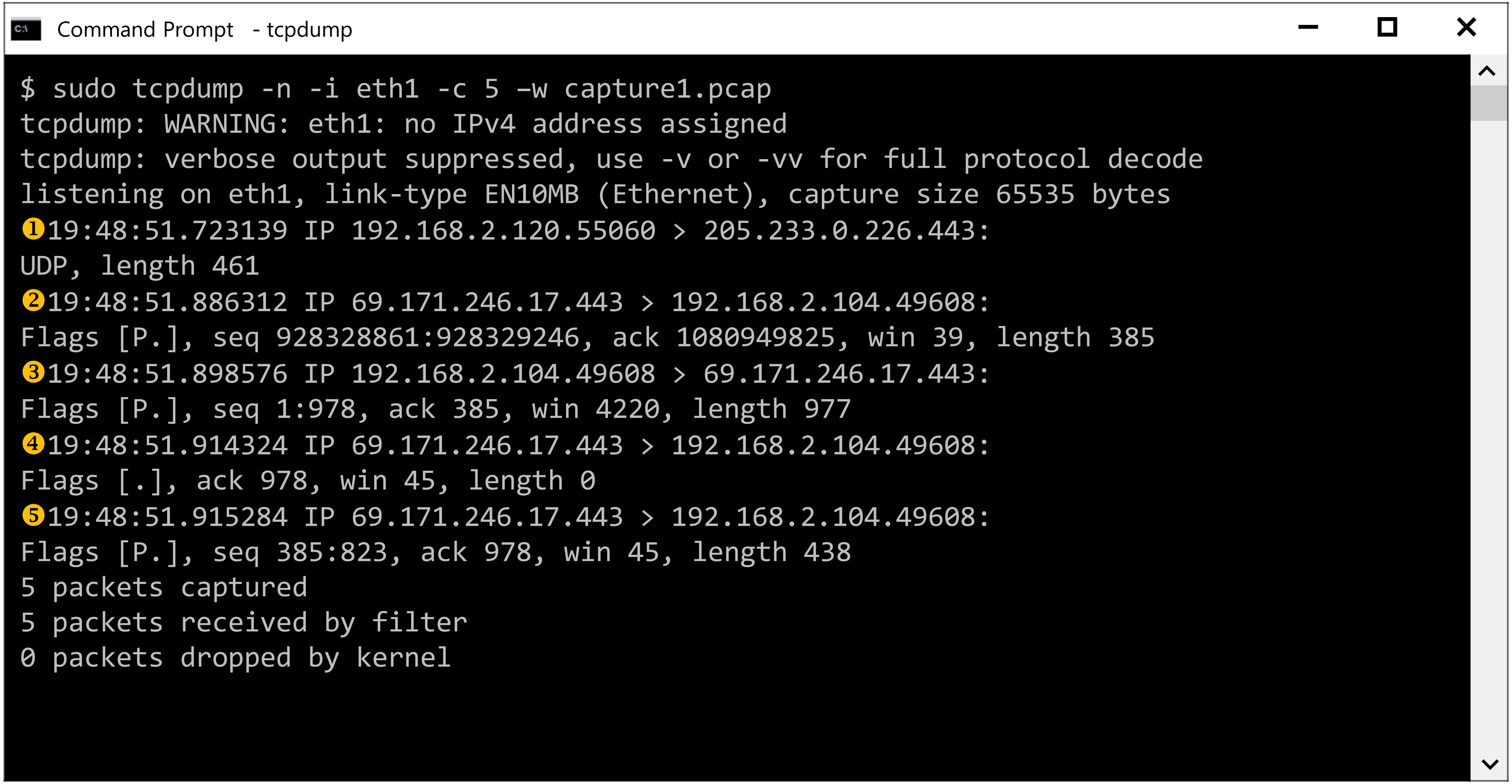
Tcpdump – Operation

- The following command displays live traffic in real time:

```
tcpdump –n –i <interface> -s <snaplen> -c <count> -w <filename>
```

- The command is modified by the options:
  - **-n** – Do not resolve IP addresses to hostnames
  - **-i** – Specify an interface to monitor
  - **-s** – Specify how many bytes to capture from each packet
    - **-s 0 –** captures the entire packet
  - **-c** – Set the number of packets to capture
  - **-w** – Write the captured data to disk

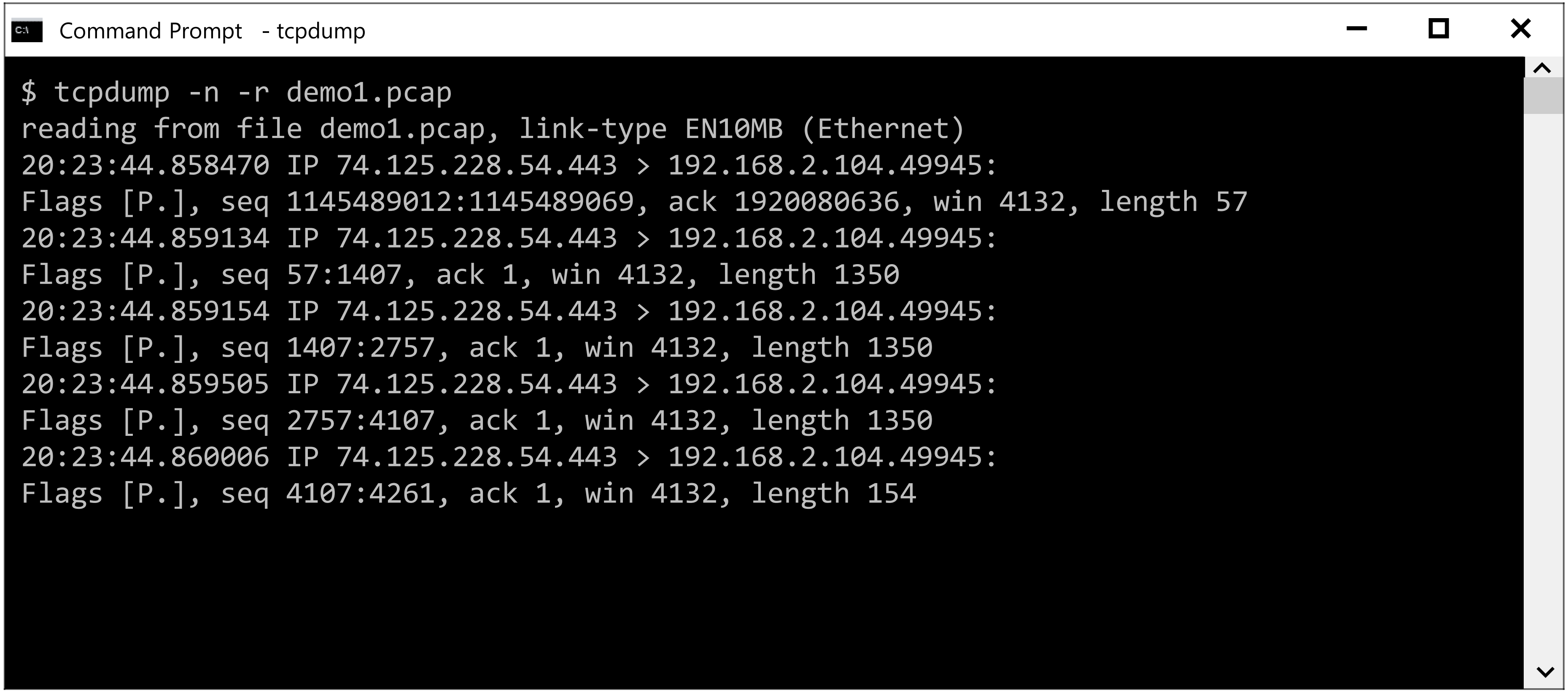
Tcpdump – Example



```
$ sudo tcpdump -n -i eth1 -c 5 -w capture1.pcap
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
❶19:48:51.723139 IP 192.168.2.120.55060 > 205.233.0.226.443:
UDP, length 461
❷19:48:51.886312 IP 69.171.246.17.443 > 192.168.2.104.49608:
Flags [P.], seq 928328861:928329246, ack 1080949825, win 39, length 385
❸19:48:51.898576 IP 192.168.2.104.49608 > 69.171.246.17.443:
Flags [P.], seq 1:978, ack 385, win 4220, length 977
❹19:48:51.914324 IP 69.171.246.17.443 > 192.168.2.104.49608:
Flags [.], ack 978, win 45, length 0
❺19:48:51.915284 IP 69.171.246.17.443 > 192.168.2.104.49608:
Flags [P.], seq 385:823, ack 978, win 45, length 438
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Tcpdump – Operation

- Tcpdump can also be used to read capture (trace) files:

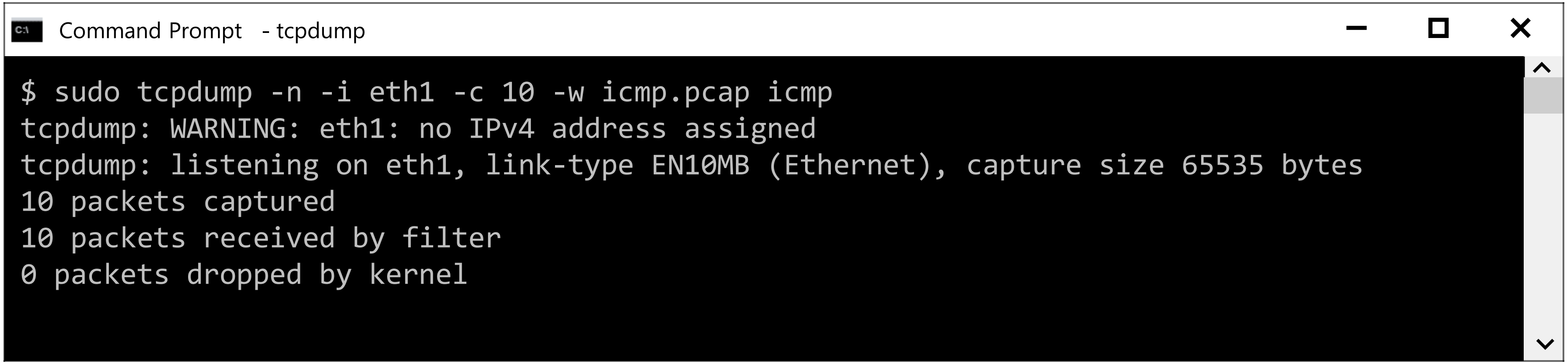


```
$ tcpdump -n -r demo1.pcap
reading from file demo1.pcap, link-type EN10MB (Ethernet)
20:23:44.858470 IP 74.125.228.54.443 > 192.168.2.104.49945:
Flags [P.], seq 1145489012:1145489069, ack 1920080636, win 4132, length 57
20:23:44.859134 IP 74.125.228.54.443 > 192.168.2.104.49945:
Flags [P.], seq 57:1407, ack 1, win 4132, length 1350
20:23:44.859154 IP 74.125.228.54.443 > 192.168.2.104.49945:
Flags [P.], seq 1407:2757, ack 1, win 4132, length 1350
20:23:44.859505 IP 74.125.228.54.443 > 192.168.2.104.49945:
Flags [P.], seq 2757:4107, ack 1, win 4132, length 1350
20:23:44.860006 IP 74.125.228.54.443 > 192.168.2.104.49945:
Flags [P.], seq 4107:4261, ack 1, win 4132, length 154
```

Tcpdump – Operation

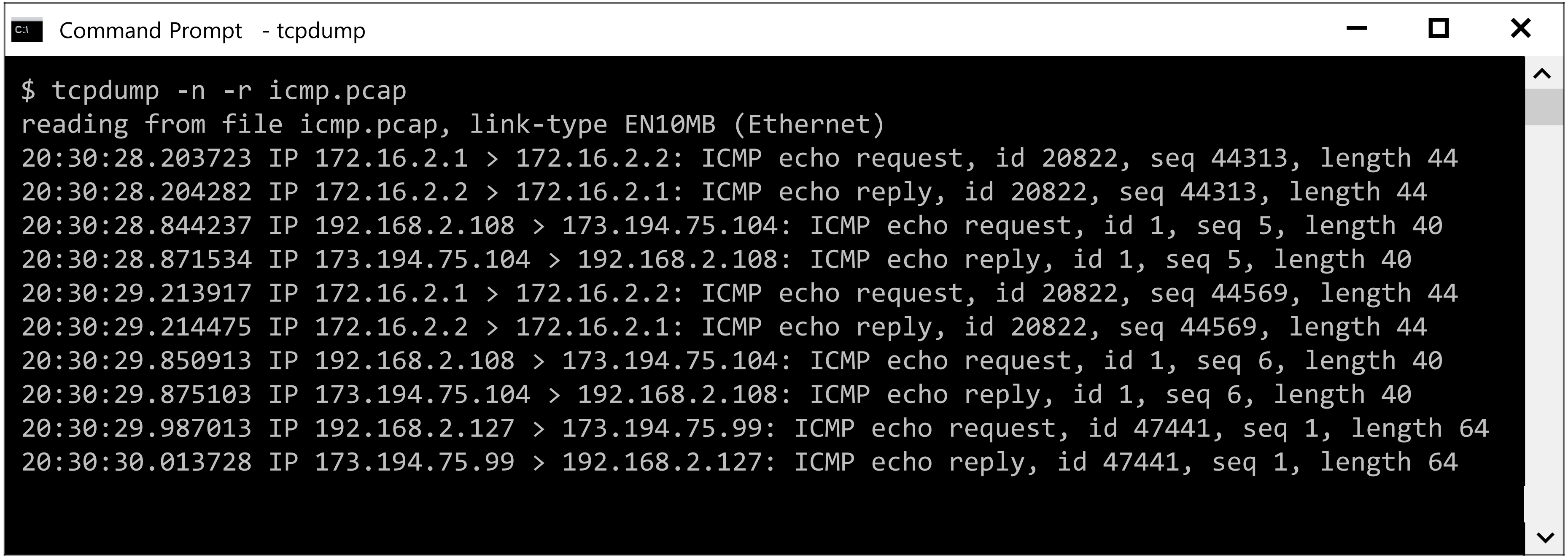
- To reduce the amount of data, Tcpdump can apply filters in the Berkley Packet Filter (BPF) format to the command

- BPF filters include protocol, direction, and type
  - To capture only ICMP traffic:



```
$ sudo tcpdump -n -i eth1 -c 10 -w icmp.pcap icmp
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

Tcpdump – Example

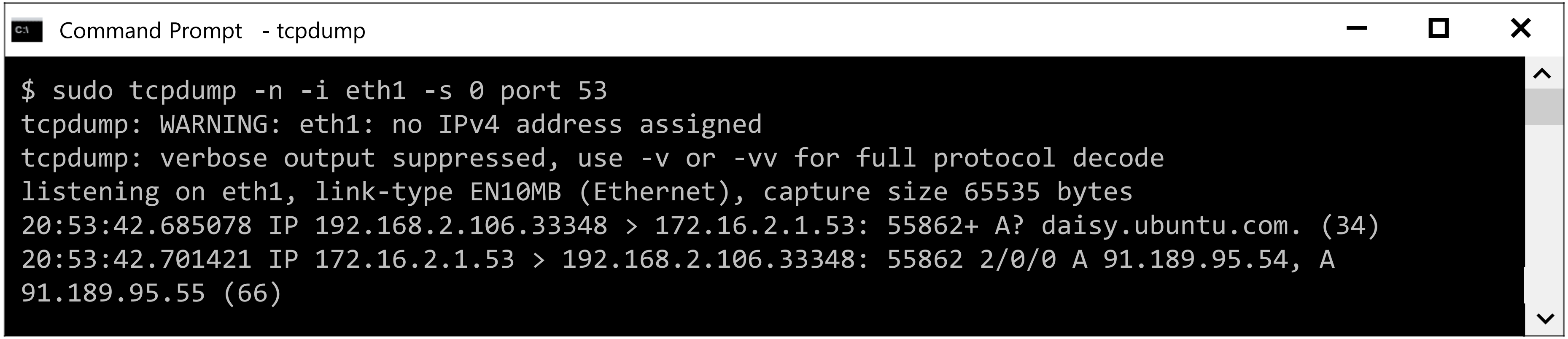


```
$ tcpdump -n -r icmp.pcap
reading from file icmp.pcap, link-type EN10MB (Ethernet)
20:30:28.203723 IP 172.16.2.1 > 172.16.2.2: ICMP echo request, id 20822, seq 44313, length 44
20:30:28.204282 IP 172.16.2.2 > 172.16.2.1: ICMP echo reply, id 20822, seq 44313, length 44
20:30:28.844237 IP 192.168.2.108 > 173.194.75.104: ICMP echo request, id 1, seq 5, length 40
20:30:28.871534 IP 173.194.75.104 > 192.168.2.108: ICMP echo reply, id 1, seq 5, length 40
20:30:29.213917 IP 172.16.2.1 > 172.16.2.2: ICMP echo request, id 20822, seq 44569, length 44
20:30:29.214475 IP 172.16.2.2 > 172.16.2.1: ICMP echo reply, id 20822, seq 44569, length 44
20:30:29.850913 IP 192.168.2.108 > 173.194.75.104: ICMP echo request, id 1, seq 6, length 40
20:30:29.875103 IP 173.194.75.104 > 192.168.2.108: ICMP echo reply, id 1, seq 6, length 40
20:30:29.987013 IP 192.168.2.127 > 173.194.75.99: ICMP echo request, id 47441, seq 1, length 64
20:30:30.013728 IP 173.194.75.99 > 192.168.2.127: ICMP echo reply, id 47441, seq 1, length 64
```

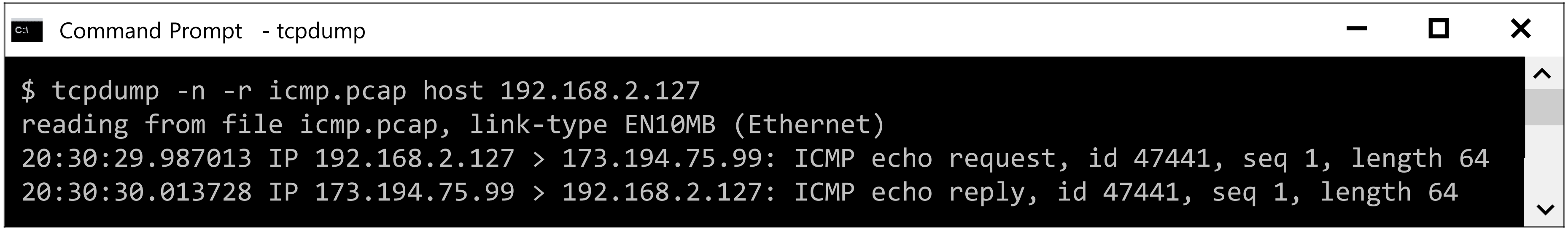
BPF Options

- host xxx.xxx.xxx.xxx (only traffic involving a single host)
- net xxx.xxx.xxx.xxx (only traffic involving a specific network)
- port xx (only traffic involving TCP/UDP port xx)
- src (only traffic from the source)
- dst (only traffic to the destination)
- ether (only Ethernet protocols)
- tcp (only TCP traffic)
- ip (only IP traffic)
- ip6 (only IPv6 traffic)
- arp (only ARP traffic)

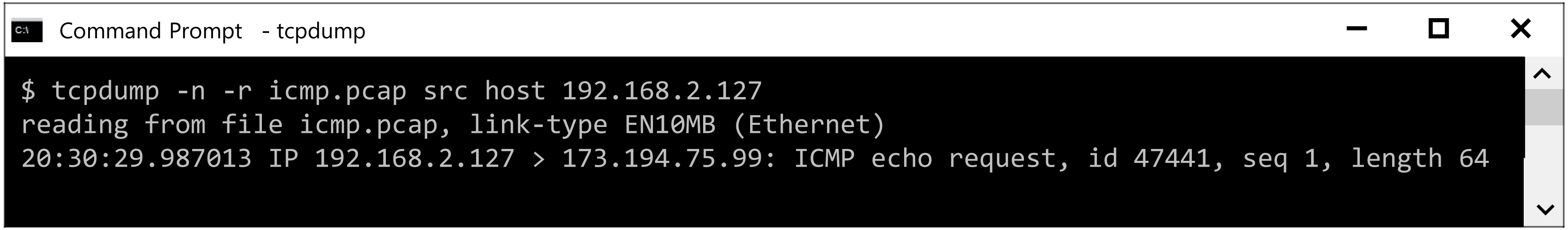
Tcpdump – Example



```
$ sudo tcpdump -n -i eth1 -s 0 port 53
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
20:53:42.685078 IP 192.168.2.106.33348 > 172.16.2.1.53: 55862+ A? daisy.ubuntu.com. (34)
20:53:42.701421 IP 172.16.2.1.53 > 192.168.2.106.33348: 55862 2/0/0 A 91.189.95.54, A
91.189.95.55 (66)
```

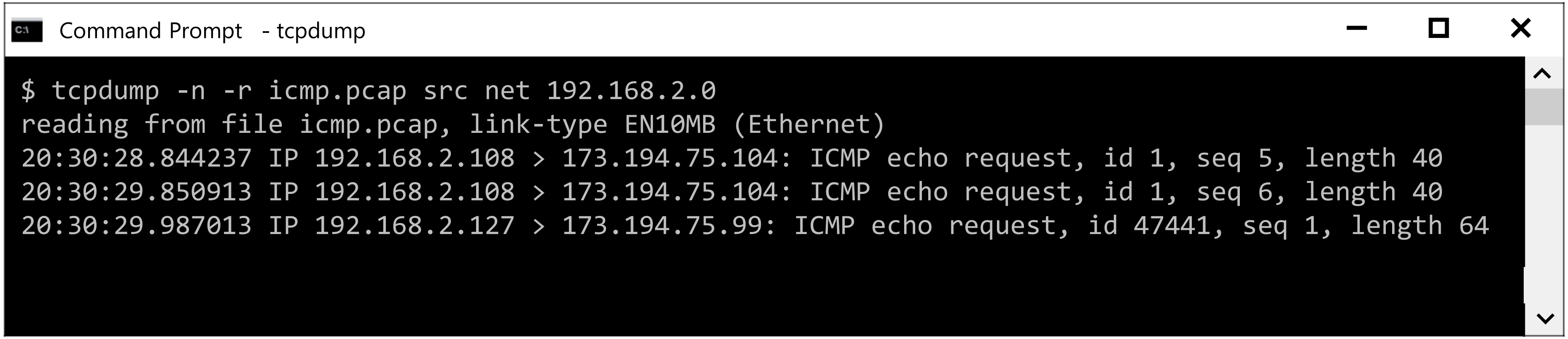


```
$ tcpdump -n -r icmp.pcap host 192.168.2.127
reading from file icmp.pcap, link-type EN10MB (Ethernet)
20:30:29.987013 IP 192.168.2.127 > 173.194.75.99: ICMP echo request, id 47441, seq 1, length 64
20:30:30.013728 IP 173.194.75.99 > 192.168.2.127: ICMP echo reply, id 47441, seq 1, length 64
```



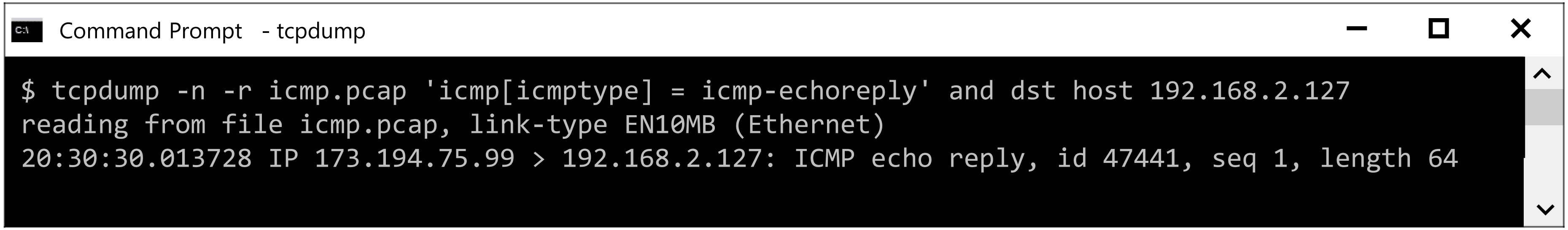
```
$ tcpdump -n -r icmp.pcap src host 192.168.2.127
reading from file icmp.pcap, link-type EN10MB (Ethernet)
20:30:29.987013 IP 192.168.2.127 > 173.194.75.99: ICMP echo request, id 47441, seq 1, length 64
```

Tcpdump – Example



```
$ tcpdump -n -r icmp.pcap src net 192.168.2.0
reading from file icmp.pcap, link-type EN10MB (Ethernet)
20:30:28.844237 IP 192.168.2.108 > 173.194.75.104: ICMP echo request, id 1, seq 5, length 40
20:30:29.850913 IP 192.168.2.108 > 173.194.75.104: ICMP echo request, id 1, seq 6, length 40
20:30:29.987013 IP 192.168.2.127 > 173.194.75.99: ICMP echo request, id 47441, seq 1, length 64
```

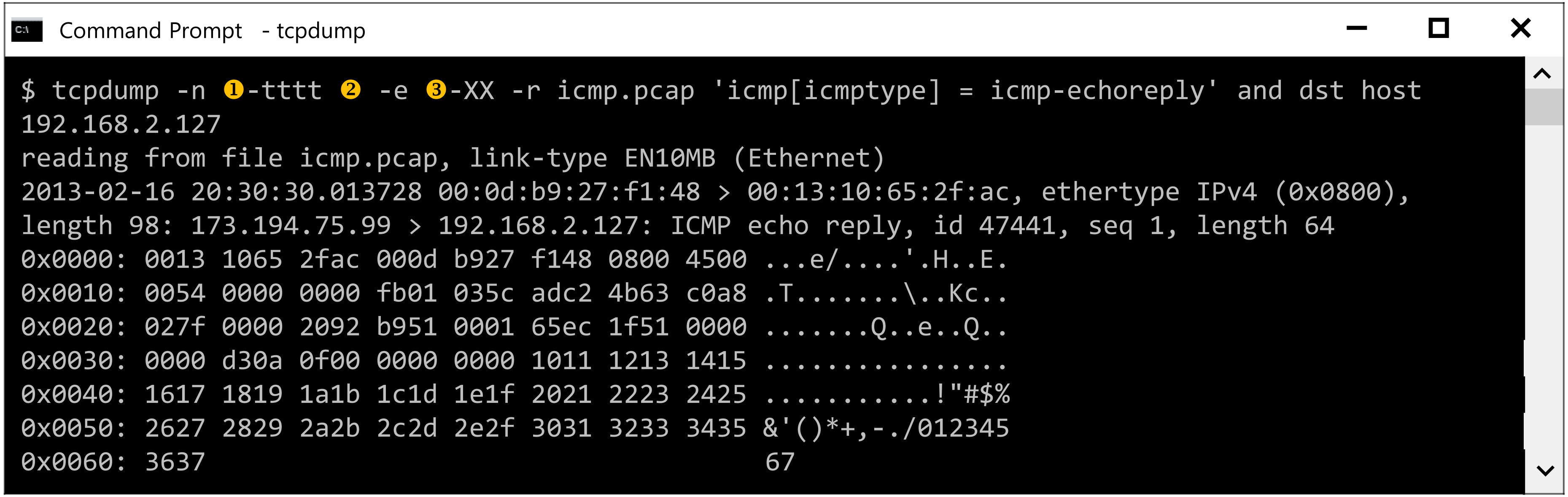
- Many protocols offer BPF primitives that allow viewing of specific aspects of traffic only



```
$ tcpdump -n -r icmp.pcap 'icmp[icmptype] = icmp-echoreply' and dst host 192.168.2.127
reading from file icmp.pcap, link-type EN10MB (Ethernet)
20:30:30.013728 IP 173.194.75.99 > 192.168.2.127: ICMP echo reply, id 47441, seq 1, length 64
```

Tcpdump – Details

- In addition to displaying layer 3 headers, Tcpdump can extract and display additional information about captures
  - The following output shows ❶ timestamps as YYYY-MM-DD HH:MM:SS.milliseconds, ❷ layer 2 headers, and ❸ hex and ASCII content

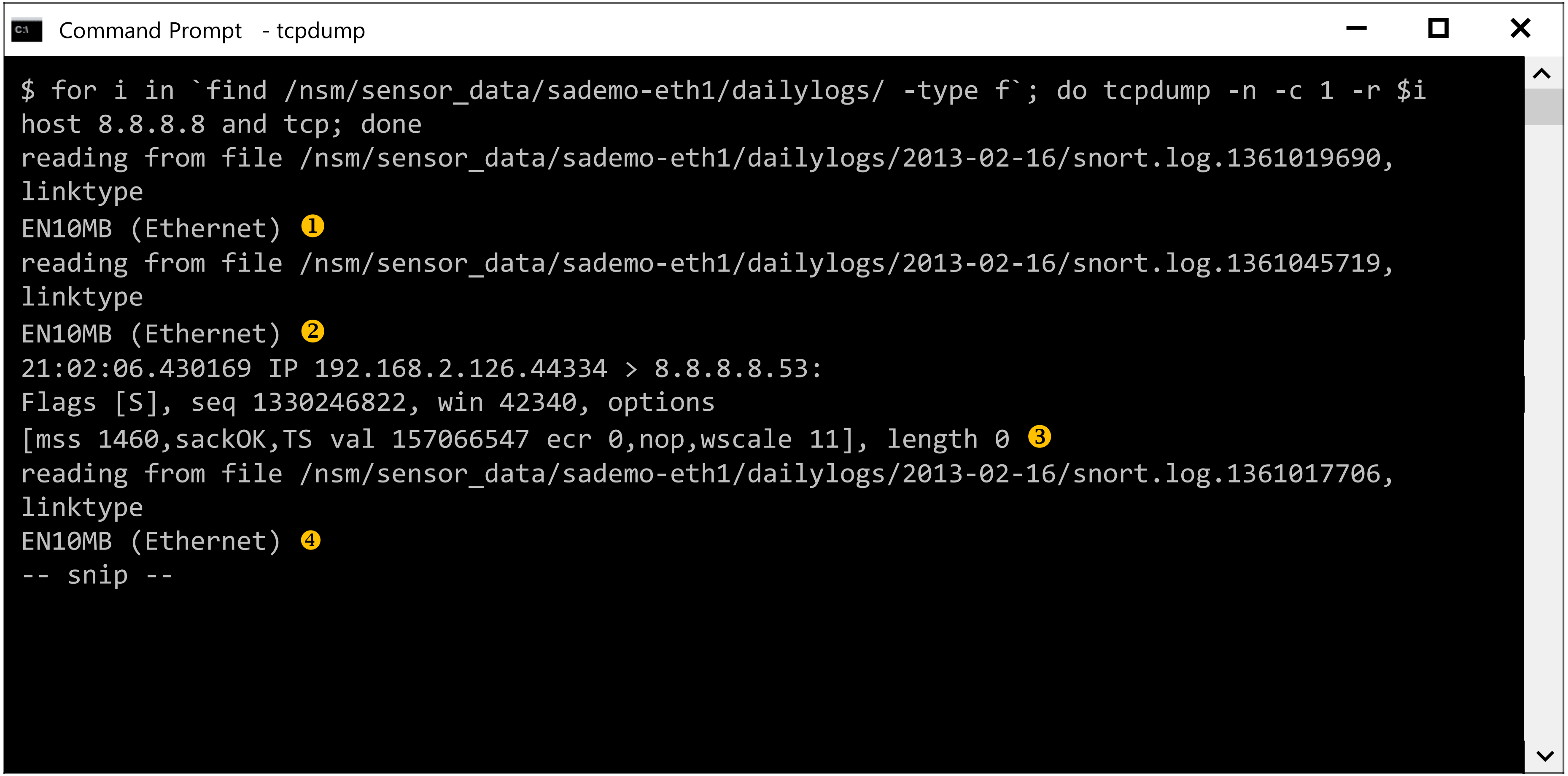


```
$ tcpdump -n ❶-tttt ❷ -e ❸-XX -r icmp.pcap 'icmp[icmptype] = icmp-echoreply' and dst host
192.168.2.127
reading from file icmp.pcap, link-type EN10MB (Ethernet)
2013-02-16 20:30:30.013728 00:0d:b9:27:f1:48 > 00:13:10:65:2f:ac, ethertype IPv4 (0x0800),
length 98: 173.194.75.99 > 192.168.2.127: ICMP echo reply, id 47441, seq 1, length 64
0x0000: 0013 1065 2fac 000d b927 f148 0800 4500 ...e/....'.H..E.
0x0010: 0054 0000 0000 fb01 035c adc2 4b63 c0a8 .T.......\..Kc..
0x0020: 027f 0000 2092 b951 0001 65ec 1f51 0000 .......Q..e..Q..
0x0030: 0000 d30a 0f00 0000 0000 1011 1213 1415 ................
0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425 ...........!"#$%
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 &'()*+,-./012345
0x0060: 3637                                    67
```

Examining Full Content Data with Tcpdump

- Tcpdump is often used to examine the contents of saved traces
- Combining Tcpdump with common Linux commands and scripting methods, you can search logs for specific traffic patterns and types, using BPF filter to hone your output
- The following output shows:
  1. The first file does not contain any traffic matching the BFP filter
  2. The second file contains matching traffic
  3. The traffic is a TCP SYN request
  4. The third file does not contain any matching traffic

Examining Full Content Data with Tcpdump



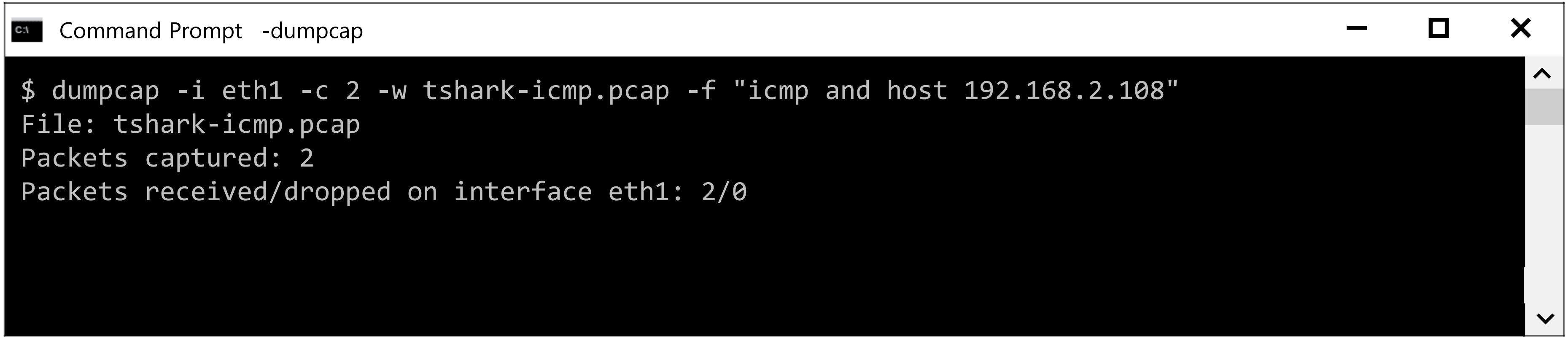
```
$ for i in `find /nsm/sensor_data/sademo-eth1/dailylogs/ -type f`; do tcpdump -n -c 1 -r $i
host 8.8.8.8 and tcp; done
reading from file /nsm/sensor_data/sademo-eth1/dailylogs/2013-02-16/snort.log.1361019690,
linktype
EN10MB (Ethernet) ❶
reading from file /nsm/sensor_data/sademo-eth1/dailylogs/2013-02-16/snort.log.1361045719,
linktype
EN10MB (Ethernet) ❷
21:02:06.430169 IP 192.168.2.126.44334 > 8.8.8.8.53:
Flags [S], seq 1330246822, win 42340, options
[mss 1460,sackOK,TS val 157066547 ecr 0,nop,wscale 11], length 0 ❸
reading from file /nsm/sensor_data/sademo-eth1/dailylogs/2013-02-16/snort.log.1361017706,
linktype
EN10MB (Ethernet) ❹
-- snip --
```

Dumpcap and Tshark

- Dumpcap and Tshark are tools shipped with Wireshark
  - Dumpcap is a traffic collection tool
  - Tshark is a protocol analyzer that can understand hundreds of protocols and apply filters using human friendly syntax
- Dumpcap uses the same BPF syntax as Tcpdump
- The power of Tshark (and Wireshark) are thanks to the protocol dissectors that translate the actions of protocols into easy to read headers

Dumpcap Example

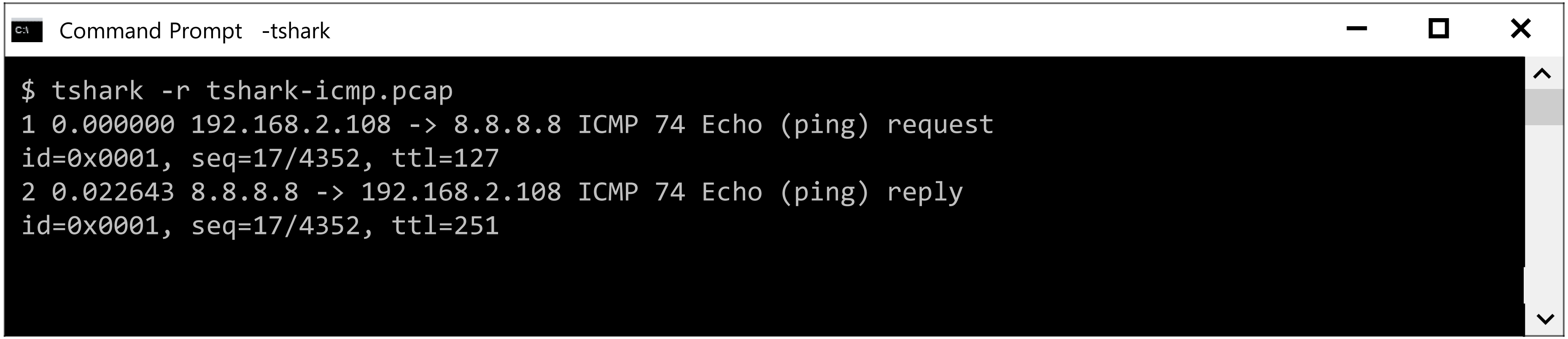
- The following output shows a dumpcap capture:
  - **-i** – Listening on the interface **eth1**
  - **-c** – Limiting the capture to **two** packets
  - **-w** – Saving to the file **tshark-icmp.pcap**
  - **-f** – Applying the BPF filter "**icmp and host 192.168.2.108**" to the data stream
    - Notice that the **-s** option to specify a maximum packet length is not required



```
$ dumpcap -i eth1 -c 2 -w tshark-icmp.pcap -f "icmp and host 192.168.2.108"
File: tshark-icmp.pcap
Packets captured: 2
Packets received/dropped on interface eth1: 2/0
```

Tshark Example

- Once the trace file has been created in dumpcap, you can analyze it with Tshark:

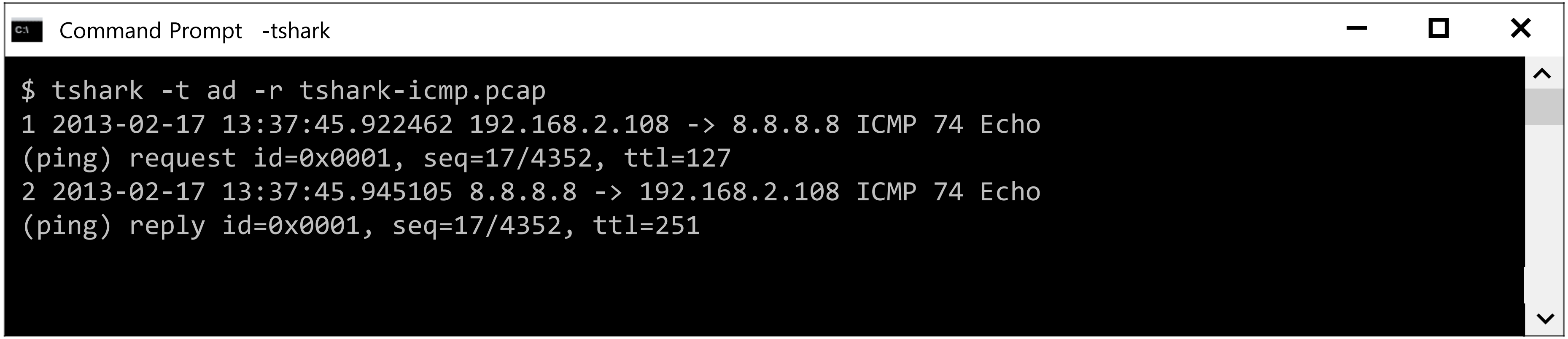


```
$ tshark -r tshark-icmp.pcap
1 0.000000 192.168.2.108 -> 8.8.8.8 ICMP 74 Echo (ping) request
id=0x0001, seq=17/4352, ttl=127
2 0.022643 8.8.8.8 -> 192.168.2.108 ICMP 74 Echo (ping) reply
id=0x0001, seq=17/4352, ttl=251
```

- The output is dissimilar to Tcpdump, but should be familiar from Wireshark

Tshark Example

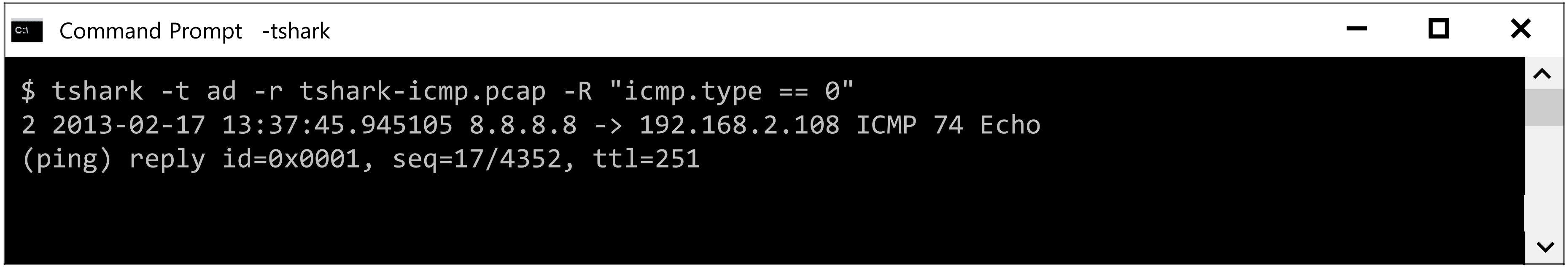
- By default, Tshark displays the time relative to the start of the capture
- To modify the display, use the -t (time) option, with the ad (absolute, date) parameters to display time that is easier to correlate with other actions



```
$ tshark -t ad -r tshark-icmp.pcap
1 2013-02-17 13:37:45.922462 192.168.2.108 -> 8.8.8.8 ICMP 74 Echo
(ping) request id=0x0001, seq=17/4352, ttl=127
2 2013-02-17 13:37:45.945105 8.8.8.8 -> 192.168.2.108 ICMP 74 Echo
(ping) reply id=0x0001, seq=17/4352, ttl=251
```

Tshark – Display Filters

- Tshark can use display filters to limit the amount of information displayed on the screen, without affecting the capture
- The following example limits the view to show only ICMP echo replies

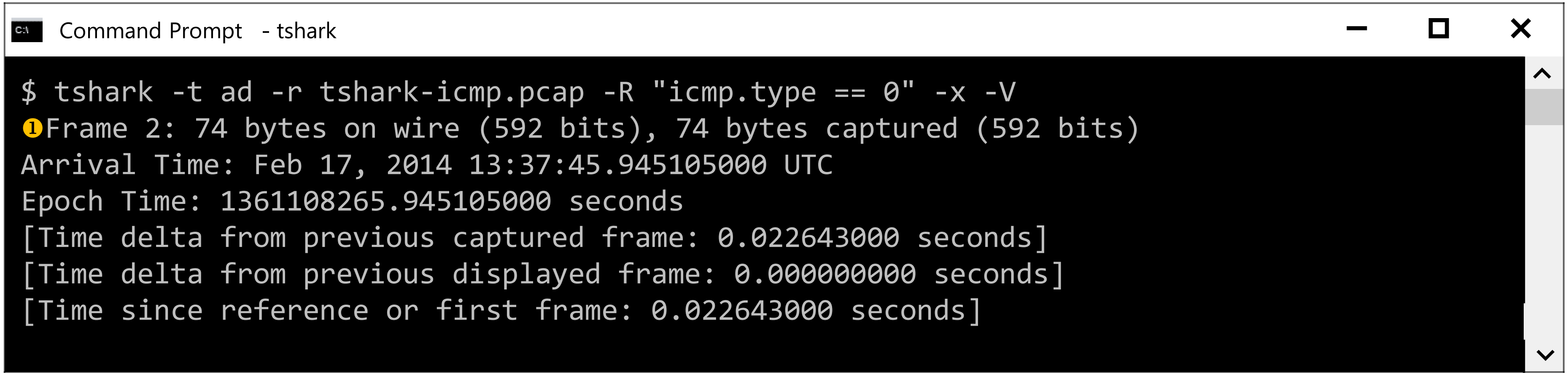


```
$ tshark -t ad -r tshark-icmp.pcap -R "icmp.type == 0"
2 2013-02-17 13:37:45.945105 8.8.8.8 -> 192.168.2.108 ICMP 74 Echo
(ping) reply id=0x0001, seq=17/4352, ttl=251
```

- There are thousands of display filter options available for Tshark, a repository of available filters can be found at: https://www.wireshark.org/docs/dfref/

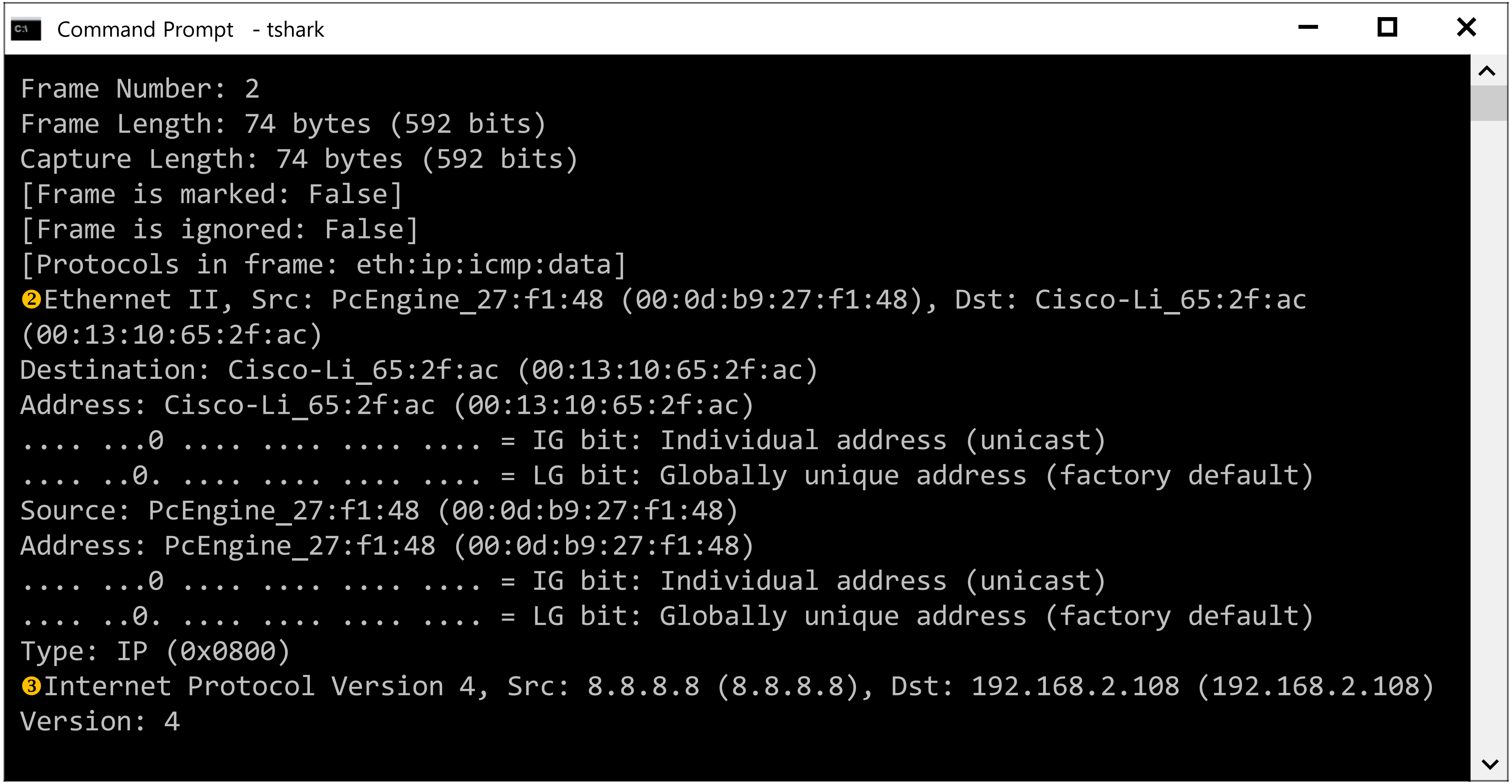
Tshark Example

- Tshark reveals its depth of knowledge when the **-v** option is used
- Additionally, adding **-x** adds hex and ASCII output to the display



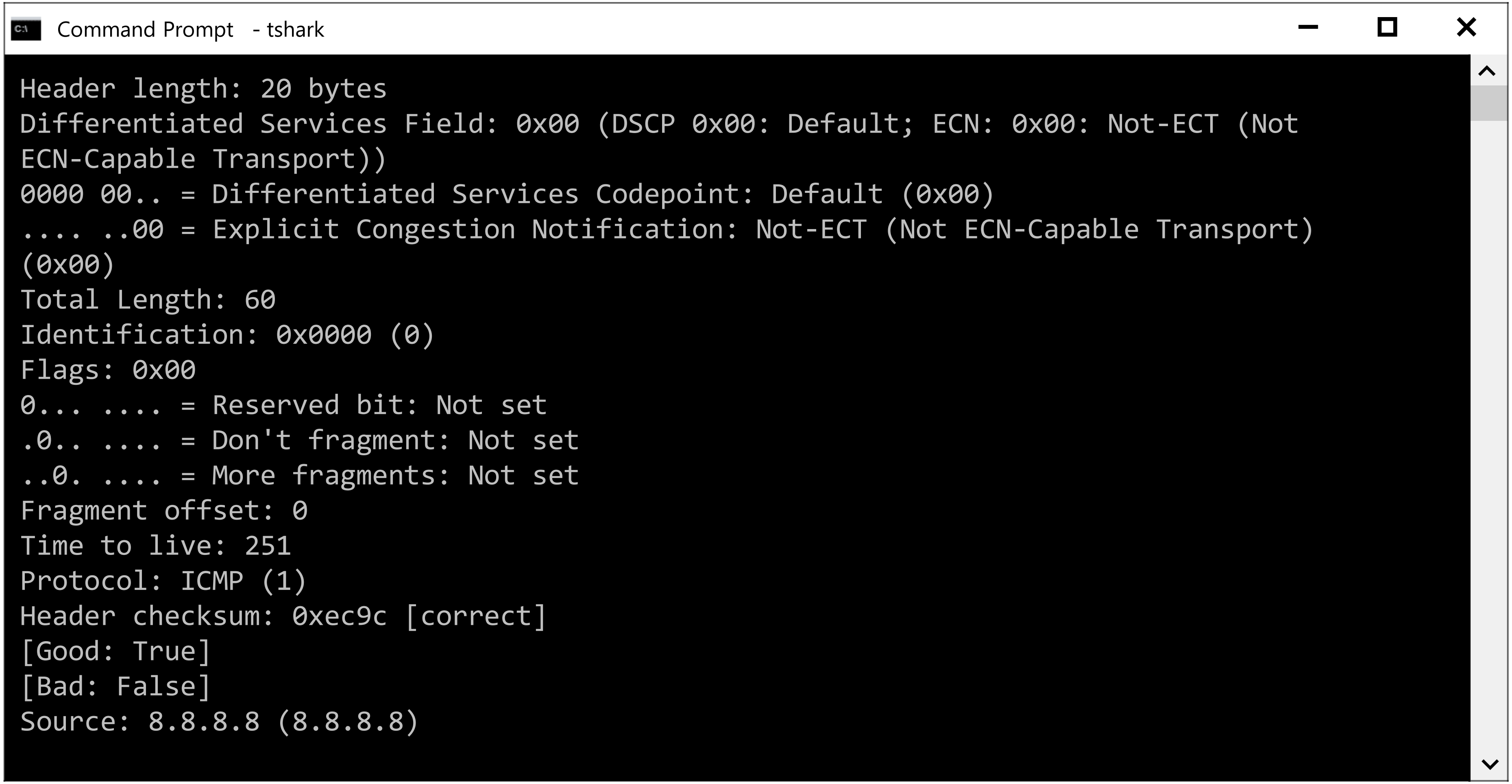
```
$ tshark -t ad -r tshark-icmp.pcap -R "icmp.type == 0" -x -V
❶Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
Arrival Time: Feb 17, 2014 13:37:45.945105000 UTC
Epoch Time: 1361108265.945105000 seconds
[Time delta from previous captured frame: 0.022643000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.022643000 seconds]
```

Tshark Example



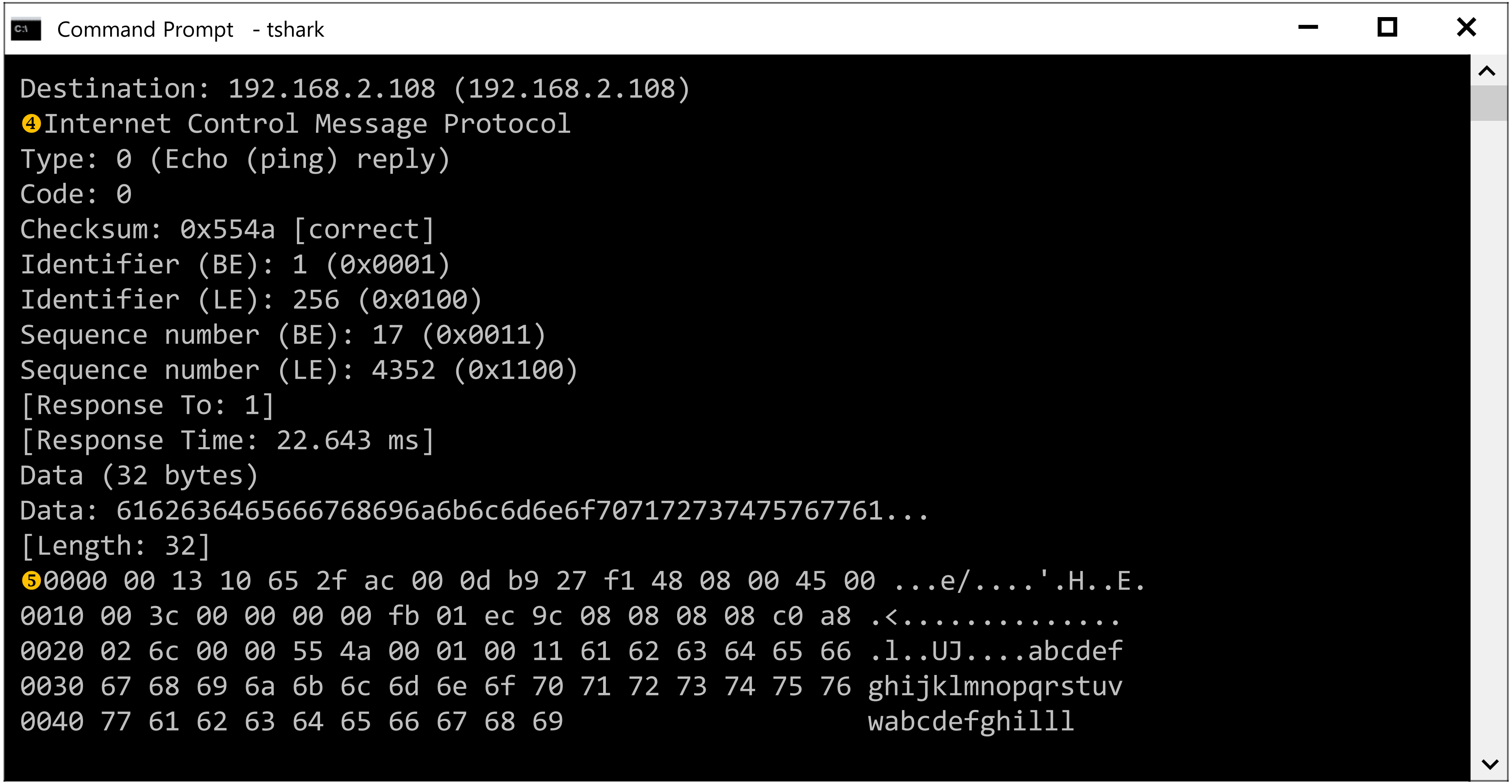
```
Frame Number: 2
Frame Length: 74 bytes (592 bits)
Capture Length: 74 bytes (592 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ip:icmp:data]
❷Ethernet II, Src: PcEngine_27:f1:48 (00:0d:b9:27:f1:48), Dst: Cisco-Li_65:2f:ac
(00:13:10:65:2f:ac)
Destination: Cisco-Li_65:2f:ac (00:13:10:65:2f:ac)
Address: Cisco-Li_65:2f:ac (00:13:10:65:2f:ac)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
Source: PcEngine_27:f1:48 (00:0d:b9:27:f1:48)
Address: PcEngine_27:f1:48 (00:0d:b9:27:f1:48)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
Type: IP (0x0800)
❸Internet Protocol Version 4, Src: 8.8.8.8 (8.8.8.8), Dst: 192.168.2.108 (192.168.2.108)
Version: 4
```

Tshark Example



```
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not
ECN-Capable Transport))
0000 00.. = Differentiated Services Codepoint: Default (0x00)
.... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport)
(0x00)
Total Length: 60
Identification: 0x0000 (0)
Flags: 0x00
0... .... = Reserved bit: Not set
.0.. .... = Don't fragment: Not set
..0. .... = More fragments: Not set
Fragment offset: 0
Time to live: 251
Protocol: ICMP (1)
Header checksum: 0xec9c [correct]
[Good: True]
[Bad: False]
Source: 8.8.8.8 (8.8.8.8)
```

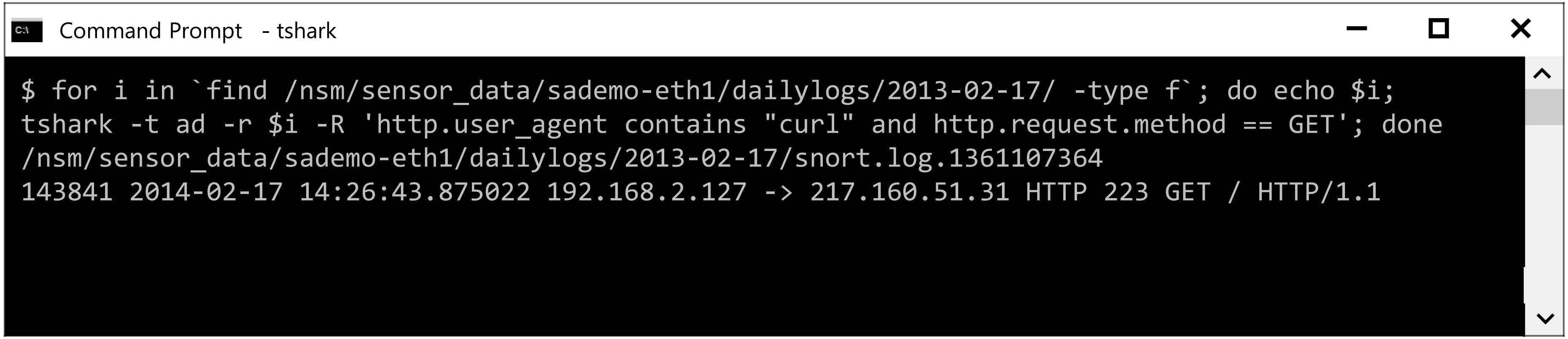
Tshark Example



```
Destination: 192.168.2.108 (192.168.2.108)
❹Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x554a [correct]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence number (BE): 17 (0x0011)
Sequence number (LE): 4352 (0x1100)
[Response To: 1]
[Response Time: 22.643 ms]
Data (32 bytes)
Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
[Length: 32]
❺0000 00 13 10 65 2f ac 00 0d b9 27 f1 48 08 00 45 00  ...e/....'.H..E.
0010 00 3c 00 00 00 00 fb 01 ec 9c 08 08 08 08 c0 a8 .<..............
0020 02 6c 00 00 55 4a 00 01 00 11 61 62 63 64 65 66 .l..UJ....abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmnopqrstuv
0040 77 61 62 63 64 65 66 67 68 69                   wabcdefghilll
```

Examining Full Content Data with Tshark

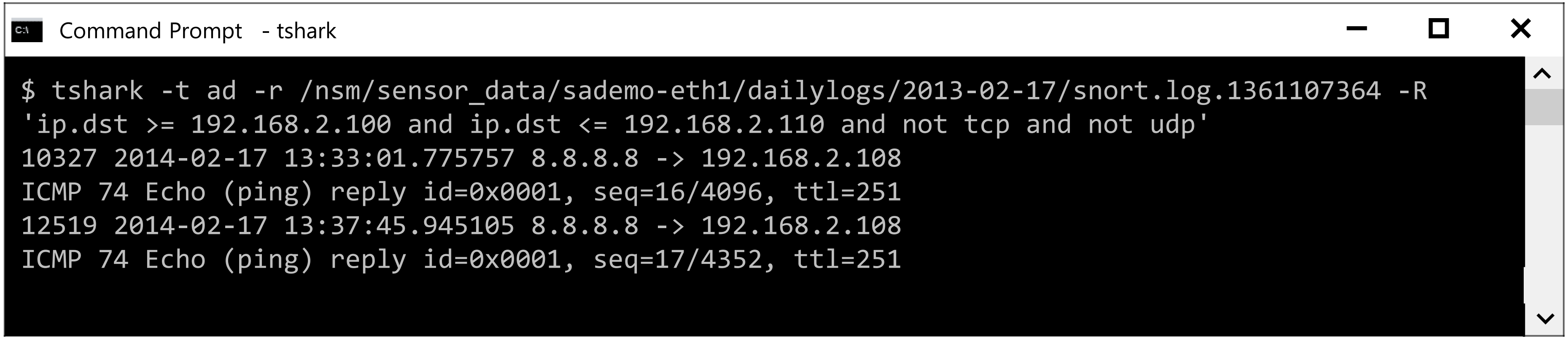
- Similarly to Tcpdump, Tshark can be combined with common Linux commands and scripting to deliver powerful searches through full content data
- One additional advantage of Tshark is the number of available display filters and the flexible configuration options
- To use a loop to search a directory for an HTTP GET request generated by Curl and list the relevant trace file:



```
$ for i in `find /nsm/sensor_data/sademo-eth1/dailylogs/2013-02-17/ -type f`; do echo $i;
tshark -t ad -r $i -R 'http.user_agent contains "curl" and http.request.method == GET'; done
/nsm/sensor_data/sademo-eth1/dailylogs/2013-02-17/snort.log.1361107364
143841 2014-02-17 14:26:43.875022 192.168.2.127 -> 217.160.51.31 HTTP 223 GET / HTTP/1.1
```

Examining Full Content Data with Tshark

- Tshark can also assist in finding traffic from a range of IP addresses

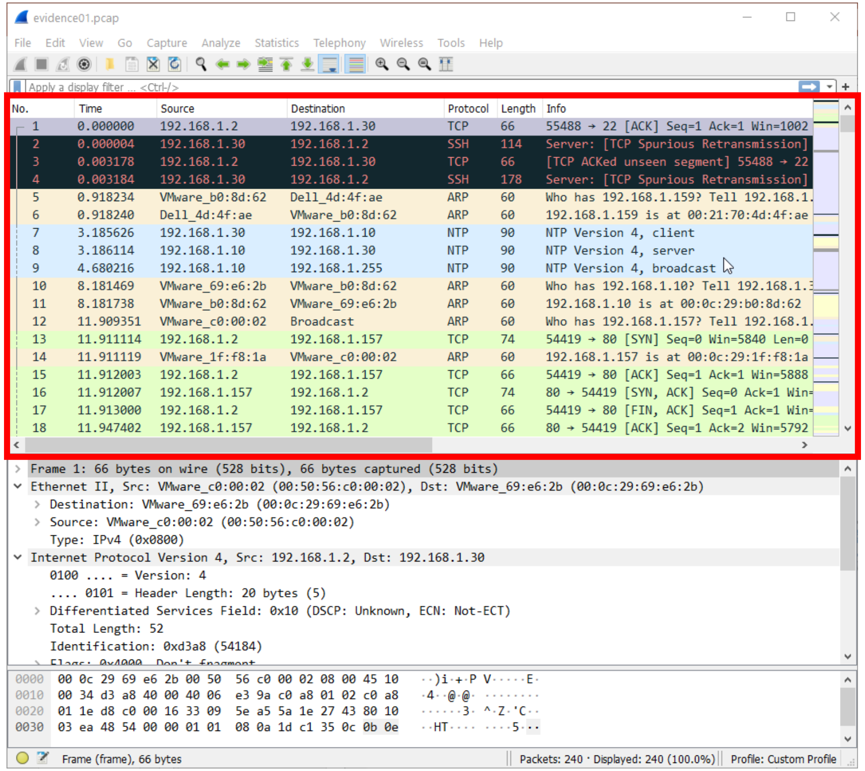


```
$ tshark -t ad -r /nsm/sensor_data/sademo-eth1/dailylogs/2013-02-17/snort.log.1361107364 -R
'ip.dst >= 192.168.2.100 and ip.dst <= 192.168.2.110 and not tcp and not udp'
10327 2014-02-17 13:33:01.775757 8.8.8.8 -> 192.168.2.108
ICMP 74 Echo (ping) reply id=0x0001, seq=16/4096, ttl=251
12519 2014-02-17 13:37:45.945105 8.8.8.8 -> 192.168.2.108
ICMP 74 Echo (ping) reply id=0x0001, seq=17/4352, ttl=251
```

Wireshark

- Wireshark is the go-to network protocol analyzer for GUI environments
- Like CLI-based tools, Wireskark can display packet headers, content and analyze the details of communications
- Unlike CLI-based tools, Wireshark can also generate reports and graphs that incorporate visual elements to better represent data
- Wireshark is used interactively, with data related to packets updated in real time

Wireshark Interface – Packet List Pane


evidence01.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time       Source           Destination      Protocol Length Info
  1  0.000000   192.168.1.2      192.168.1.30     TCP      66     55488 → 22 [ACK] Seq=1 Ack=1 Win=1002
  2  0.000004   192.168.1.30     192.168.1.2      SSH      114    Server: [TCP Spurious Retransmission]
  3  0.003178   192.168.1.2      192.168.1.30     TCP      66     [TCP ACKed unseen segment] 55488 → 22
  4  0.003184   192.168.1.30     192.168.1.2      SSH      178    Server: [TCP Spurious Retransmission]
  5  0.918234   VMware_b0:8d:62  Dell_4d:4f:ae    ARP      60     Who has 192.168.1.159? Tell 192.168.1.
  6  0.918240   Dell_4d:4f:ae    VMware_b0:8d:62  ARP      60     192.168.1.159 is at 00:21:70:4d:4f:ae
  7  3.185626   192.168.1.30     192.168.1.10     NTP      90     NTP Version 4, client
  8  3.186114   192.168.1.10     192.168.1.30     NTP      90     NTP Version 4, server
  9  4.680216   192.168.1.10     192.168.1.255    NTP      90     NTP Version 4, broadcast
 10  8.181469   VMware_69:e6:2b  VMware_b0:8d:62  ARP      60     Who has 192.168.1.10? Tell 192.168.1.3
 11  8.181738   VMware_b0:8d:62  VMware_69:e6:2b  ARP      60     192.168.1.10 is at 00:0c:29:b0:8d:62
 12  11.909351  VMware_c0:00:02  Broadcast        ARP      60     Who has 192.168.1.157? Tell 192.168.1.
 13  11.911114  192.168.1.2      192.168.1.157    TCP      74     54419 → 80 [SYN] Seq=0 Win=5840 Len=0
 14  11.911119  VMware_1f:f8:1a  VMware_c0:00:02  ARP      60     192.168.1.157 is at 00:0c:29:1f:f8:1a
 15  11.912003  192.168.1.2      192.168.1.157    TCP      66     54419 → 80 [ACK] Seq=1 Ack=1 Win=5888
 16  11.912007  192.168.1.157    192.168.1.2      TCP      74     80 → 54419 [SYN, ACK] Seq=0 Ack=1 Win=
 17  11.913000  192.168.1.2      192.168.1.157    TCP      66     54419 → 80 [FIN, ACK] Seq=1 Ack=1 Win=
 18  11.947402  192.168.1.157    192.168.1.2      TCP      66     80 → 54419 [ACK] Seq=1 Ack=2 Win=5792
```

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
v Ethernet II, Src: VMware_c0:00:02 (00:50:56:c0:00:02), Dst: VMware_69:e6:2b (00:0c:29:69:e6:2b)
  > Destination: VMware_69:e6:2b (00:0c:29:69:e6:2b)
  > Source: VMware_c0:00:02 (00:50:56:c0:00:02)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.30
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xd3a8 (54184)
  > Flags: 0x4000, Don't fragment
```

```
0000  00 0c 29 69 e6 2b 00 50  56 c0 00 02 08 00 45 10   ··)i·+·P V·····E·
0010  00 34 d3 a8 40 00 40 06  e3 9a c0 a8 01 02 c0 a8   ·4··@·@· ········
0020  01 1e d8 c0 00 16 33 09  5e a5 5a 1e 27 43 80 10   ······3· ^·Z·'C··
0030  03 ea 48 54 00 00 01 01  08 0a 1d c1 35 0c 0b 0e   ··HT···· ····5···
```

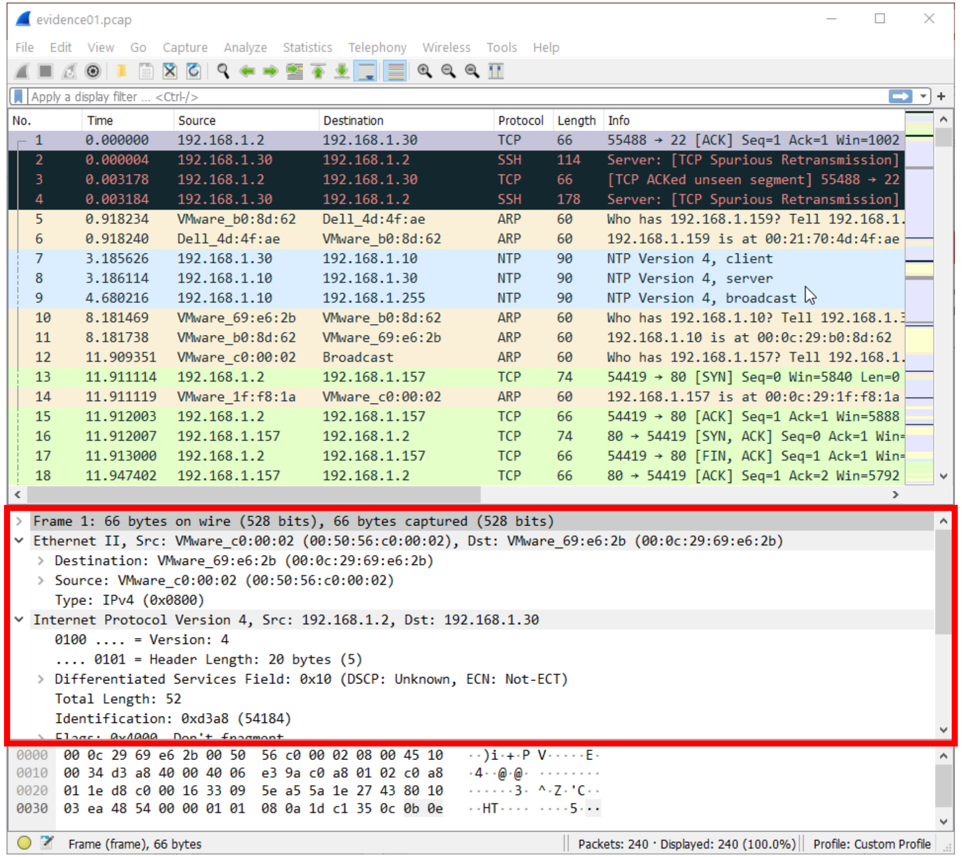
Frame (frame), 66 bytes | Packets: 240 · Displayed: 240 (100.0%) | Profile: Custom Profile

Symbol	Description
┌	First packet in a conversation.
│	Part of the selected conversation.
┆	*Not* part of the selected conversation.
└	Last packet in a conversation.
→	Request
←	Response
✓	The selected packet acknowledges this packet.
✓✓	The selected packet is a duplicate acknowledgement of this packet.
●	The selected packet is related to this packet in some other way, e.g. as part of reassembly.

Lists all the packets captured including:

- The timestamp of the packet
- Source and destination IP addresses
- Protocol
- Packet length
- Packet info summary

Wireshark Interface – Packet Details Pane


evidence01.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.2	192.168.1.30	TCP	66	55488 → 22 [ACK] Seq=1 Ack=1 Win=1002
2	0.000004	192.168.1.30	192.168.1.2	SSH	114	Server: [TCP Spurious Retransmission]
3	0.003178	192.168.1.2	192.168.1.30	TCP	66	[TCP ACKed unseen segment] 55488 → 22
4	0.003184	192.168.1.30	192.168.1.2	SSH	178	Server: [TCP Spurious Retransmission]
5	0.918234	VMware_b0:8d:62	Dell_4d:4f:ae	ARP	60	Who has 192.168.1.159? Tell 192.168.1.
6	0.918240	Dell_4d:4f:ae	VMware_b0:8d:62	ARP	60	192.168.1.159 is at 00:21:70:4d:4f:ae
7	3.185626	192.168.1.30	192.168.1.10	NTP	90	NTP Version 4, client
8	3.186114	192.168.1.10	192.168.1.30	NTP	90	NTP Version 4, server
9	4.680216	192.168.1.10	192.168.1.255	NTP	90	NTP Version 4, broadcast
10	8.181469	VMware_69:e6:2b	VMware_b0:8d:62	ARP	60	Who has 192.168.1.10? Tell 192.168.1.3
11	8.181738	VMware_b0:8d:62	VMware_69:e6:2b	ARP	60	192.168.1.10 is at 00:0c:29:b0:8d:62
12	11.909351	VMware_c0:00:02	Broadcast	ARP	60	Who has 192.168.1.157? Tell 192.168.1.
13	11.911114	192.168.1.2	192.168.1.157	TCP	74	54419 → 80 [SYN] Seq=0 Win=5840 Len=0
14	11.911119	VMware_1f:f8:1a	VMware_c0:00:02	ARP	60	192.168.1.157 is at 00:0c:29:1f:f8:1a
15	11.912003	192.168.1.2	192.168.1.157	TCP	66	54419 → 80 [ACK] Seq=1 Ack=1 Win=5888
16	11.912007	192.168.1.157	192.168.1.2	TCP	74	80 → 54419 [SYN, ACK] Seq=0 Ack=1 Win=
17	11.913000	192.168.1.2	192.168.1.157	TCP	66	54419 → 80 [FIN, ACK] Seq=1 Ack=1 Win=
18	11.947402	192.168.1.157	192.168.1.2	TCP	66	80 → 54419 [ACK] Seq=1 Ack=2 Win=5792

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
v Ethernet II, Src: VMware_c0:00:02 (00:50:56:c0:00:02), Dst: VMware_69:e6:2b (00:0c:29:69:e6:2b)
  > Destination: VMware_69:e6:2b (00:0c:29:69:e6:2b)
  > Source: VMware_c0:00:02 (00:50:56:c0:00:02)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.30
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xd3a8 (54184)
  > Flags: 0x4000, Don't fragment
```

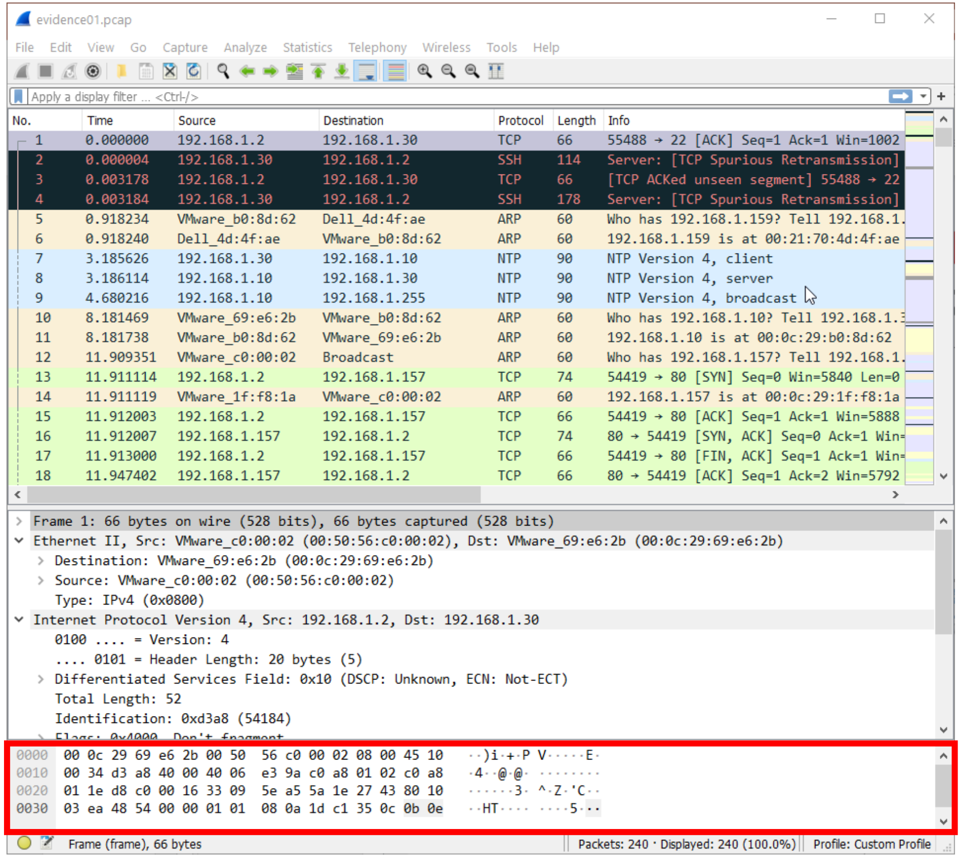
```
0000  00 0c 29 69 e6 2b 00 50  56 c0 00 02 08 00 45 10   ··)i·+·P V·····E·
0010  00 34 d3 a8 40 00 40 06  e3 9a c0 a8 01 02 c0 a8   ·4··@·@· ········
0020  01 1e d8 c0 00 16 33 09  5e a5 5a 1e 27 43 80 10   ······3· ^·Z·'C··
0030  03 ea 48 54 00 00 01 01  08 0a 1d c1 35 0c 0b 0e   ··HT···· ····5···
```

Frame (frame), 66 bytes | Packets: 240 · Displayed: 240 (100.0%) | Profile: Custom Profile

Lists all the packets captured including:

- Time difference between packets received
- Source and destination IP addresses
- Protocol
- Packet length
- Protocol fields

Wireshark Interface – Packet Bytes Pane



evidence01.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.2	192.168.1.30	TCP	66	55488 → 22 [ACK] Seq=1 Ack=1 Win=1002
2	0.000004	192.168.1.30	192.168.1.2	SSH	114	Server: [TCP Spurious Retransmission]
3	0.003178	192.168.1.2	192.168.1.30	TCP	66	[TCP ACKed unseen segment] 55488 → 22
4	0.003184	192.168.1.30	192.168.1.2	SSH	178	Server: [TCP Spurious Retransmission]
5	0.918234	VMware_b0:8d:62	Dell_4d:4f:ae	ARP	60	Who has 192.168.1.159? Tell 192.168.1.
6	0.918240	Dell_4d:4f:ae	VMware_b0:8d:62	ARP	60	192.168.1.159 is at 00:21:70:4d:4f:ae
7	3.185626	192.168.1.30	192.168.1.10	NTP	90	NTP Version 4, client
8	3.186114	192.168.1.10	192.168.1.30	NTP	90	NTP Version 4, server
9	4.680216	192.168.1.10	192.168.1.255	NTP	90	NTP Version 4, broadcast
10	8.181469	VMware_69:e6:2b	VMware_b0:8d:62	ARP	60	Who has 192.168.1.10? Tell 192.168.1.3
11	8.181738	VMware_b0:8d:62	VMware_69:e6:2b	ARP	60	192.168.1.10 is at 00:0c:29:b0:8d:62
12	11.909351	VMware_c0:00:02	Broadcast	ARP	60	Who has 192.168.1.157? Tell 192.168.1.
13	11.911114	192.168.1.2	192.168.1.157	TCP	74	54419 → 80 [SYN] Seq=0 Win=5840 Len=0
14	11.911119	VMware_1f:f8:1a	VMware_c0:00:02	ARP	60	192.168.1.157 is at 00:0c:29:1f:f8:1a
15	11.912003	192.168.1.2	192.168.1.157	TCP	66	54419 → 80 [ACK] Seq=1 Ack=1 Win=5888
16	11.912007	192.168.1.157	192.168.1.2	TCP	74	80 → 54419 [SYN, ACK] Seq=0 Ack=1 Win=
17	11.913000	192.168.1.2	192.168.1.157	TCP	66	54419 → 80 [FIN, ACK] Seq=1 Ack=1 Win=
18	11.947402	192.168.1.157	192.168.1.2	TCP	66	80 → 54419 [ACK] Seq=1 Ack=2 Win=5792

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
v Ethernet II, Src: VMware_c0:00:02 (00:50:56:c0:00:02), Dst: VMware_69:e6:2b (00:0c:29:69:e6:2b)
  > Destination: VMware_69:e6:2b (00:0c:29:69:e6:2b)
  > Source: VMware_c0:00:02 (00:50:56:c0:00:02)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.30
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xd3a8 (54184)
  > Flags: 0x4000, Don't fragment
```

```
0000  00 0c 29 69 e6 2b 00 50  56 c0 00 02 08 00 45 10   ··)i·+·P V·····E·
0010  00 34 d3 a8 40 00 40 06  e3 9a c0 a8 01 02 c0 a8   ·4··@·@· ········
0020  01 1e d8 c0 00 16 33 09  5e a5 5a 1e 27 43 80 10   ······3· ^·Z·'C··
0030  03 ea 48 54 00 00 01 01  08 0a 1d c1 35 0c 0b 0e   ··HT···· ····5···
```

Frame (frame), 66 bytes | Packets: 240 · Displayed: 240 (100.0%) | Profile: Custom Profile

Lists information inside the selected packet including:

- Incremental count or offset from the start of the selected packet
- Packet displayed in hexadecimal
- Packet displayed in ASCII

Wireshark – Display filters

- Syntax is based on expressions returning true or false
- Using with Boolean logic operators to create expressions

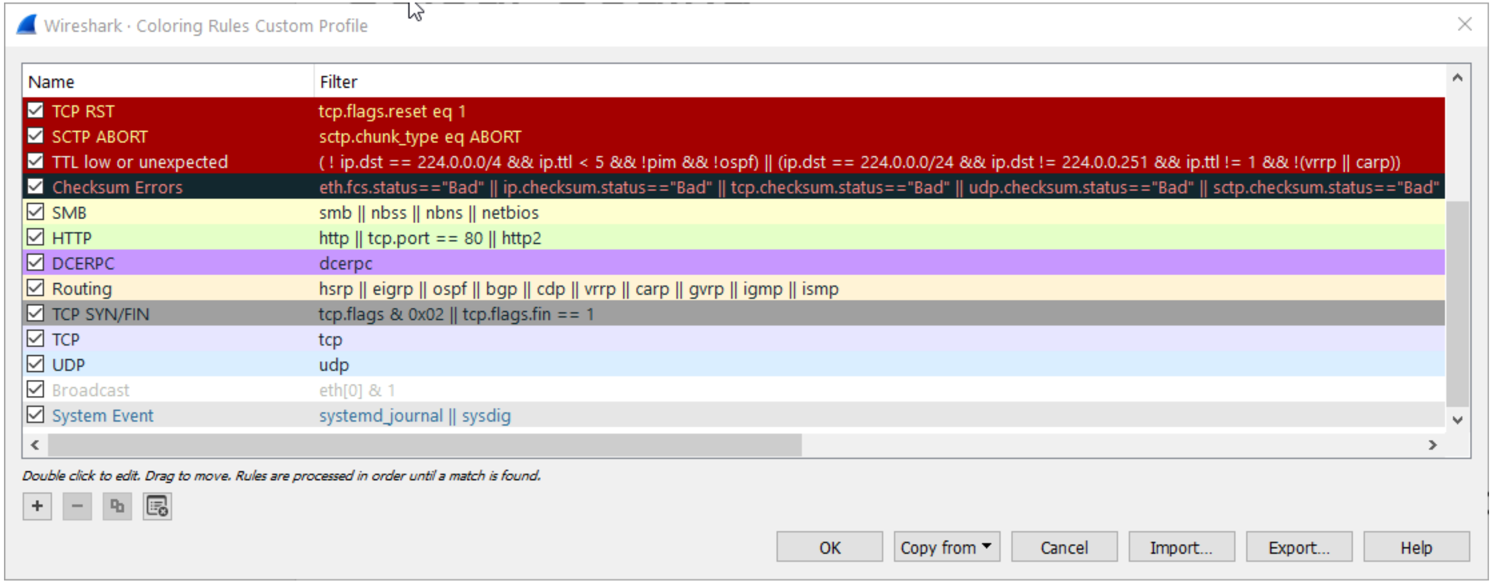
English	C-like	Description and Example
eq	==	Equal `ip.src eq 10.0.0.5`
ne	!=	Not equal `ip.dst != 10.0.0.20`
gt	>	Greater than `frame.len ge 10`
lt	<	Less than `frame.len < 128`
ge	>=	Greater than or equal to `frame.len ge 0x100`
le	<=	Less than or equal to `frame.len <= 0x20`
contains		Protocol, field or slice contains a value `sip.To contains “a1762”`
matches	~	Protocol or text field match Pearl regular expression `http.host matches “acme\.(org|com|net)”`
bitwise_and	&	Compare bit field value `tcp.flags & 0x02`

Wireshark – Display filters – Operators

- Using display filters requires matching operators against variables in the packet
- Expressions are combined using logical operators

English	C-like	Description and Example
and	&&	Logical AND - Returns true if both expressions are true
or	||	Logical OR – Returns true if one or both expressions are true
xor	^^	Exclusive OR – Returns true if only one of both expressions is true
not	!	Logical NOT – Negates the following expression
	[…]	Slice operator – With this operator a slice (substring) of the string can be accessed. dns.resp.name[1..4] accesses the first four characters of the DNS responses name
	( )	Groups expressions together

Wireshark – Colour Coding


Wireshark · Coloring Rules Custom Profile

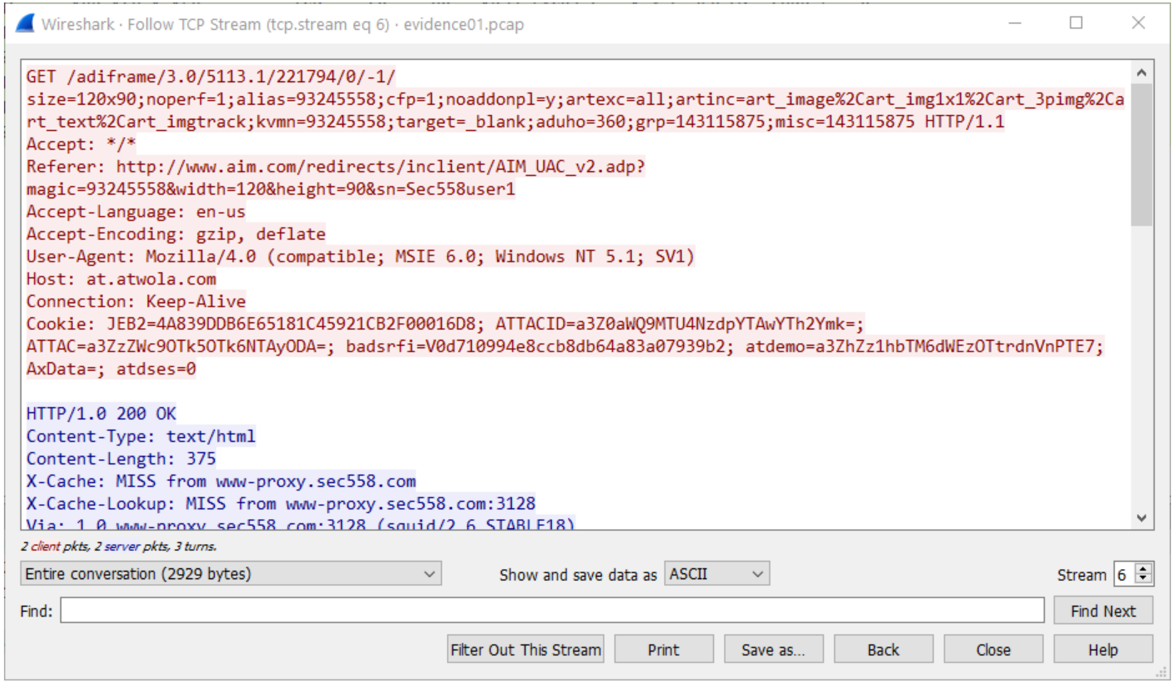
Name	Filter
☑ TCP RST	tcp.flags.reset eq 1
☑ SCTP ABORT	sctp.chunk_type eq ABORT
☑ TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) || (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp || carp))
☑ Checksum Errors	eth.fcs.status=="Bad" || ip.checksum.status=="Bad" || tcp.checksum.status=="Bad" || udp.checksum.status=="Bad" || sctp.checksum.status=="Bad"
☑ SMB	smb || nbss || nbns || netbios
☑ HTTP	http || tcp.port == 80 || http2
☑ DCERPC	dcerpc
☑ Routing	hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || igmp || ismp
☑ TCP SYN/FIN	tcp.flags & 0x02 || tcp.flags.fin == 1
☑ TCP	tcp
☑ UDP	udp
☑ Broadcast	eth[0] & 1
☑ System Event	systemd_journal || sysdig

Double click to edit. Drag to move. Rules are processed in order until a match is found.

- The packet list pane uses colours to differentiate traffic types. By default,
    - light purple is TCP traffic
    - light blue is UDP traffic
    - black identifies packets with errors

Wireshark – Viewing Streams

- To view the full conversation by protocol between the source and destination, right-click a packet and select Follow > TCP Stream.
- Closing the TCP stream window will highlight conversation in the Packet List Pane



Wireshark · Follow TCP Stream (tcp.stream eq 6) · evidence01.pcap

```
GET /adiframe/3.0/5113.1/221794/0/-1/
size=120x90;noperf=1;alias=93245558;cfp=1;noaddonpl=y;artexc=all;artinc=art_image%2Cart_img1x1%2Cart_3pimg%2Ca
rt_text%2Cart_imgtrack;kvmn=93245558;target=_blank;aduho=360;grp=143115875;misc=143115875 HTTP/1.1
Accept: */*
Referer: http://www.aim.com/redirects/inclient/AIM_UAC_v2.adp?
magic=93245558&width=120&height=90&sn=Sec558user1
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: at.atwola.com
Connection: Keep-Alive
Cookie: JEB2=4A839DDB6E65181C45921CB2F00016D8; ATTACID=a3Z0aWQ9MTU4NzdpYTAwYTh2Ymk=;
ATTAC=a3ZzZWc9OTk5OTk6NTAyODA=; badsrfi=V0d710994e8ccb8db64a83a07939b2; atdemo=a3ZhZz1hbTM6dWEzOTtrdnVnPTE7;
AxData=; atdses=0

HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 375
X-Cache: MISS from www-proxy.sec558.com
X-Cache-Lookup: MISS from www-proxy.sec558.com:3128
Via: 1.0 www-proxy.sec558.com:3128 (squid/2.6.STABLE18)
```

2 client pkts, 2 server pkts, 3 turns.

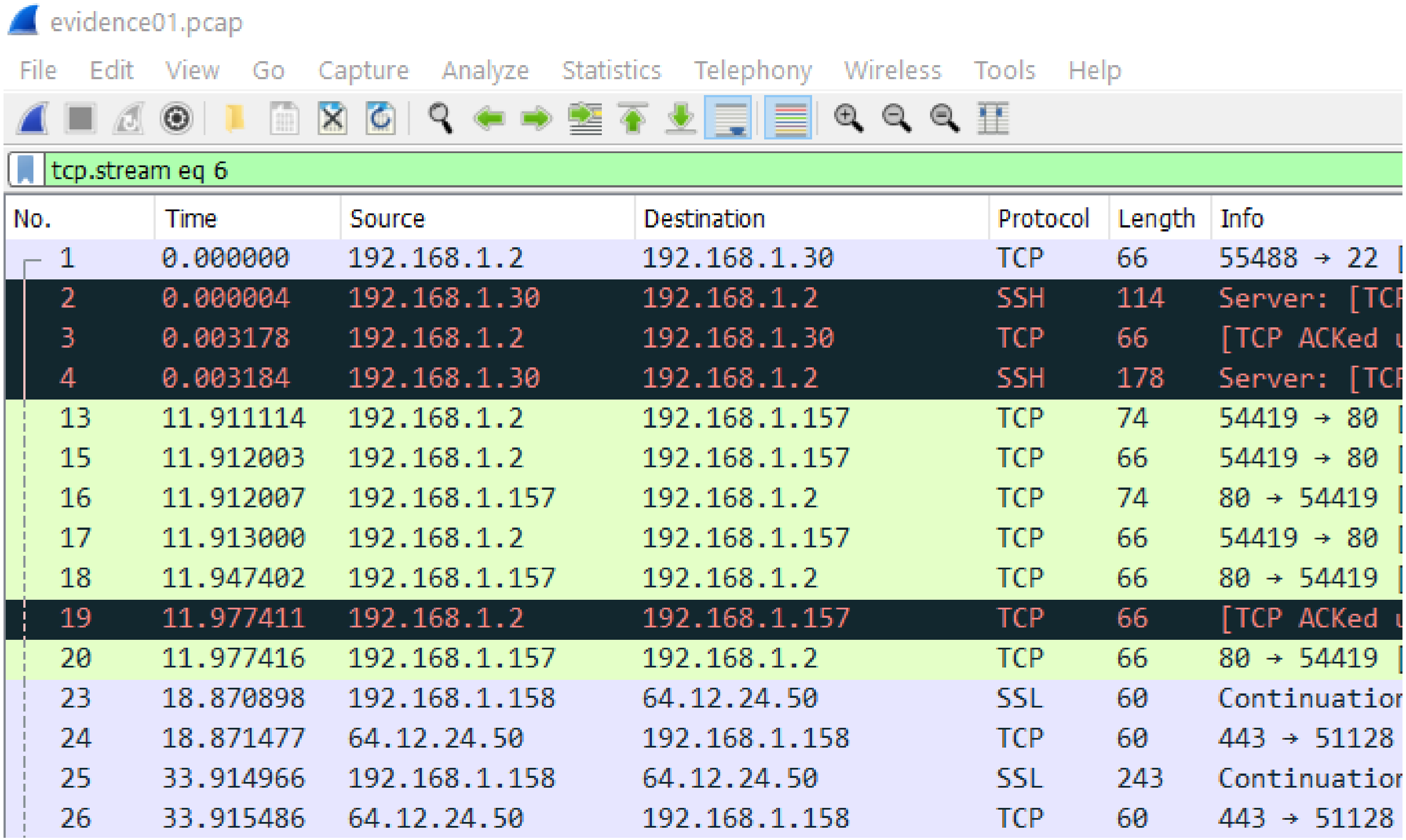
Entire conversation (2929 bytes)

Show and save data as: ASCII

Stream: 6

Find:

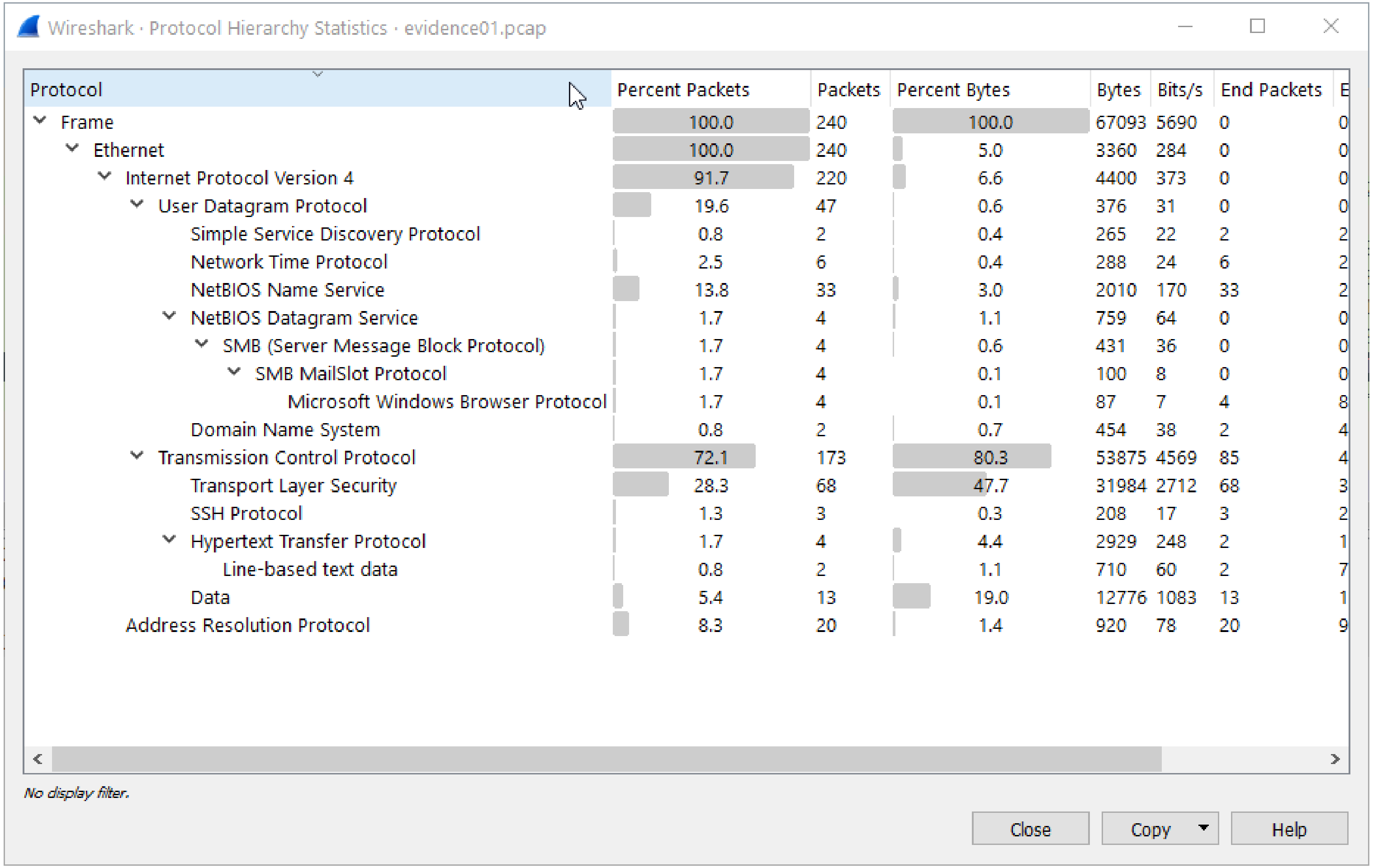
Wireshark – Viewing Streams



tcp.stream eq 6

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.2	192.168.1.30	TCP	66	55488 → 22 [
2	0.000004	192.168.1.30	192.168.1.2	SSH	114	Server: [TC
3	0.003178	192.168.1.2	192.168.1.30	TCP	66	[TCP ACKed
4	0.003184	192.168.1.30	192.168.1.2	SSH	178	Server: [TC
13	11.911114	192.168.1.2	192.168.1.157	TCP	74	54419 → 80 [
15	11.912003	192.168.1.2	192.168.1.157	TCP	66	54419 → 80 [
16	11.912007	192.168.1.157	192.168.1.2	TCP	74	80 → 54419 [
17	11.913000	192.168.1.2	192.168.1.157	TCP	66	54419 → 80 [
18	11.947402	192.168.1.157	192.168.1.2	TCP	66	80 → 54419 [
19	11.977411	192.168.1.2	192.168.1.157	TCP	66	[TCP ACKed
20	11.977416	192.168.1.157	192.168.1.2	TCP	66	80 → 54419 [
23	18.870898	192.168.1.158	64.12.24.50	SSL	60	Continuatio
24	18.871477	64.12.24.50	192.168.1.158	TCP	60	443 → 51128
25	33.914966	192.168.1.158	64.12.24.50	SSL	243	Continuatio
26	33.915486	64.12.24.50	192.168.1.158	TCP	60	443 → 51128

Wireshark Statistics – Protocol Hierarchy

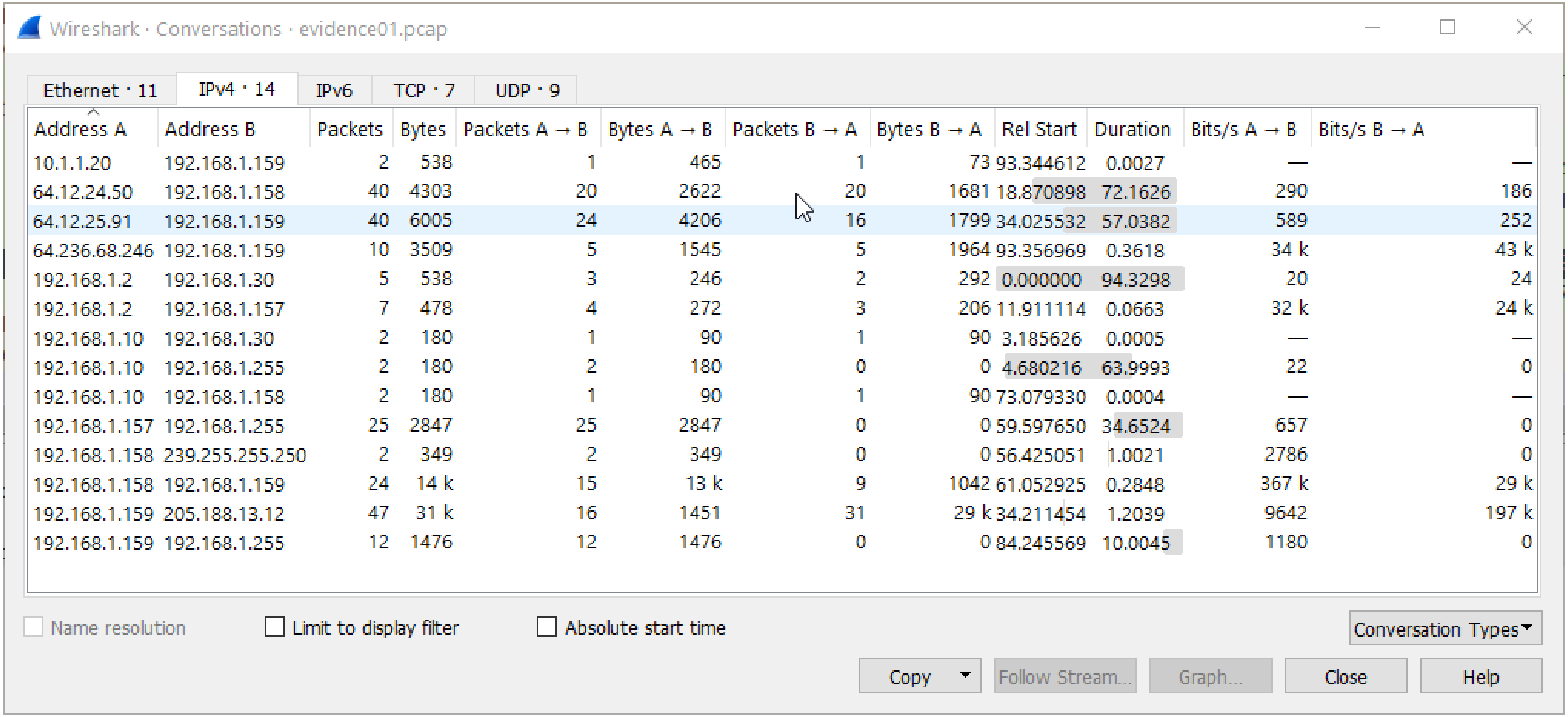


Wireshark · Protocol Hierarchy Statistics · evidence01.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	E
Frame	100.0	240	100.0	67093	5690	0	0
Ethernet	100.0	240	5.0	3360	284	0	0
Internet Protocol Version 4	91.7	220	6.6	4400	373	0	0
User Datagram Protocol	19.6	47	0.6	376	31	0	0
Simple Service Discovery Protocol	0.8	2	0.4	265	22	2	2
Network Time Protocol	2.5	6	0.4	288	24	6	2
NetBIOS Name Service	13.8	33	3.0	2010	170	33	2
NetBIOS Datagram Service	1.7	4	1.1	759	64	0	0
SMB (Server Message Block Protocol)	1.7	4	0.6	431	36	0	0
SMB MailSlot Protocol	1.7	4	0.1	100	8	0	0
Microsoft Windows Browser Protocol	1.7	4	0.1	87	7	4	8
Domain Name System	0.8	2	0.7	454	38	2	4
Transmission Control Protocol	72.1	173	80.3	53875	4569	85	4
Transport Layer Security	28.3	68	47.7	31984	2712	68	3
SSH Protocol	1.3	3	0.3	208	17	3	2
Hypertext Transfer Protocol	1.7	4	4.4	2929	248	2	1
Line-based text data	0.8	2	1.1	710	60	2	7
Data	5.4	13	19.0	12776	1083	13	1
Address Resolution Protocol	8.3	20	1.4	920	78	20	9

No display filter.

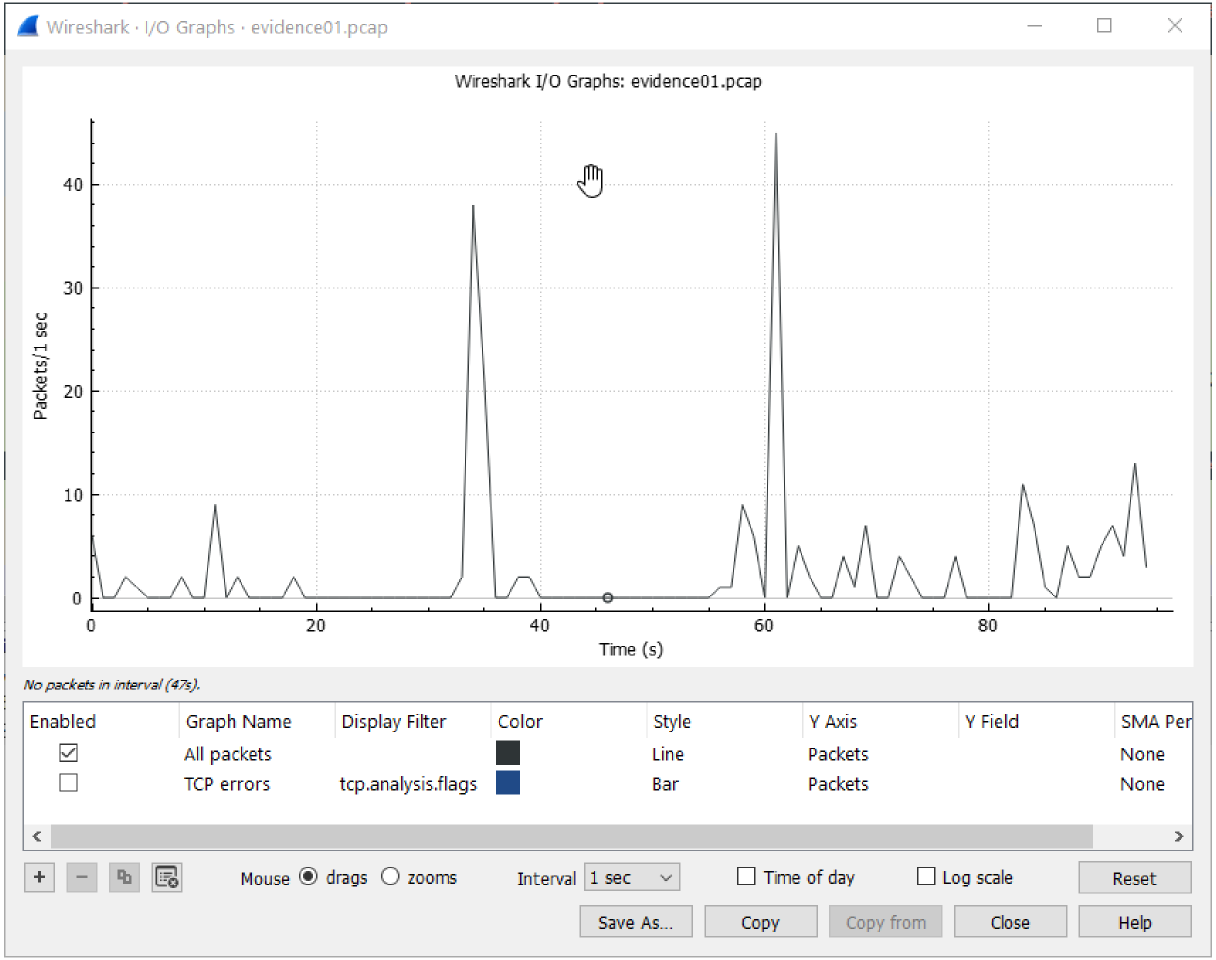
Wireshark Statistics – Conversation



Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.1.1.20	192.168.1.159	2	538	1	465	1	73	93.344612	0.0027	—	—
64.12.24.50	192.168.1.158	40	4303	20	2622	20	1681	18.870898	72.1626	290	186
64.12.25.91	192.168.1.159	40	6005	24	4206	16	1799	34.025532	57.0382	589	252
64.236.68.246	192.168.1.159	10	3509	5	1545	5	1964	93.356969	0.3618	34 k	43 k
192.168.1.2	192.168.1.30	5	538	3	246	2	292	0.000000	94.3298	20	24
192.168.1.2	192.168.1.157	7	478	4	272	3	206	11.911114	0.0663	32 k	24 k
192.168.1.10	192.168.1.30	2	180	1	90	1	90	3.185626	0.0005	—	—
192.168.1.10	192.168.1.255	2	180	2	180	0	0	4.680216	63.9993	22	0
192.168.1.10	192.168.1.158	2	180	1	90	1	90	73.079330	0.0004	—	—
192.168.1.157	192.168.1.255	25	2847	25	2847	0	0	59.597650	34.6524	657	0
192.168.1.158	239.255.255.250	2	349	2	349	0	0	56.425051	1.0021	2786	0
192.168.1.158	192.168.1.159	24	14 k	15	13 k	9	1042	61.052925	0.2848	367 k	29 k
192.168.1.159	205.188.13.12	47	31 k	16	1451	31	29 k	34.211454	1.2039	9642	197 k
192.168.1.159	192.168.1.255	12	1476	12	1476	0	0	84.245569	10.0045	1180	0

☐ Name resolution ☐ Limit to display filter ☐ Absolute start time

Wireshark Statistics – IO Graph

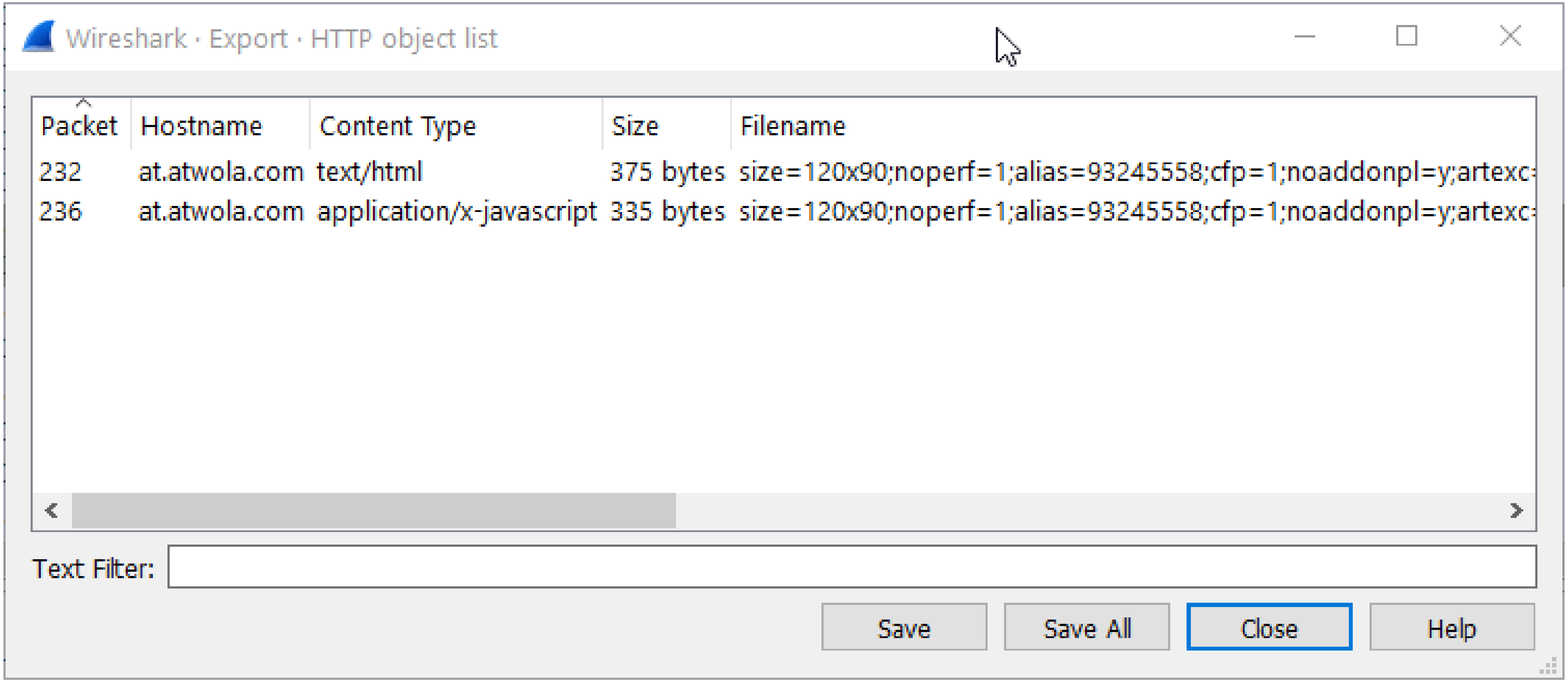


No packets in interval (47s).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Per
☑	All packets		■	Line	Packets		None
□	TCP errors	tcp.analysis.flags	🟦	Bar	Packets		None

Mouse ● drags ○ zooms Interval 1 sec □ Time of day □ Log scale

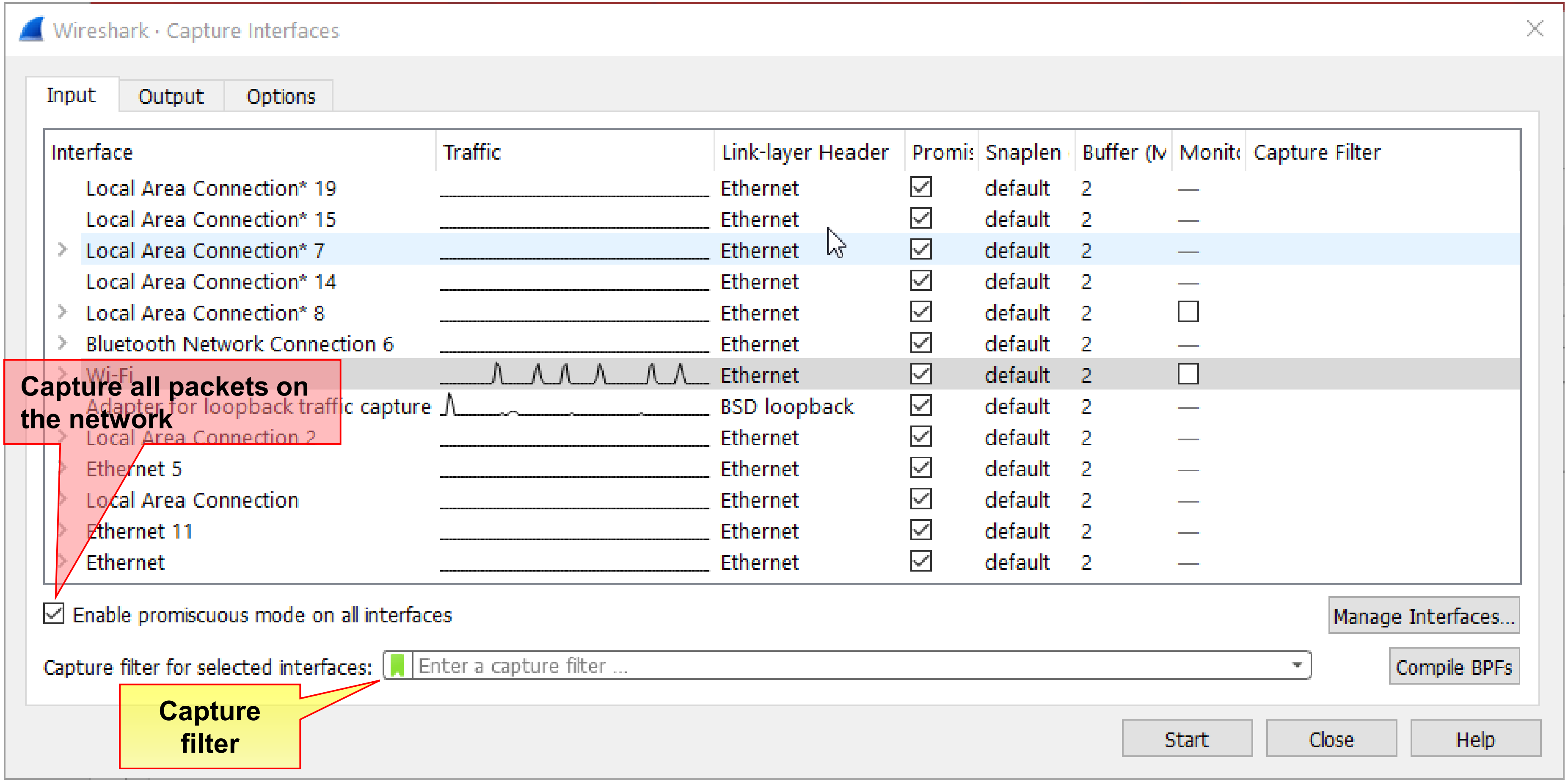
Wireshark – Export Objects



Packet	Hostname	Content Type	Size	Filename
232	at.atwola.com	text/html	375 bytes	size=120x90;noperf=1;alias=93245558;cfp=1;noaddonpl=y;artexc
236	at.atwola.com	application/x-javascript	335 bytes	size=120x90;noperf=1;alias=93245558;cfp=1;noaddonpl=y;artexc

Text Filter:

Wireshark – Capture Options



Wireshark · Capture Interfaces

Interface	Traffic	Link-layer Header	Promis	Snaplen	Buffer (M	Monit	Capture Filter
Local Area Connection* 19		Ethernet	☑	default	2	—	
Local Area Connection* 15		Ethernet	☑	default	2	—	
Local Area Connection* 7		Ethernet	☑	default	2	—	
Local Area Connection* 14		Ethernet	☑	default	2	—	
Local Area Connection* 8		Ethernet	☑	default	2	□	
Bluetooth Network Connection 6		Ethernet	☑	default	2	—	
Wi-Fi		Ethernet	☑	default	2	□	
Adapter for loopback traffic capture		BSD loopback	☑	default	2	—	
Local Area Connection 2		Ethernet	☑	default	2	—	
Ethernet 5		Ethernet	☑	default	2	—	
Local Area Connection		Ethernet	☑	default	2	—	
Ethernet 11		Ethernet	☑	default	2	—	
Ethernet		Ethernet	☑	default	2	—	

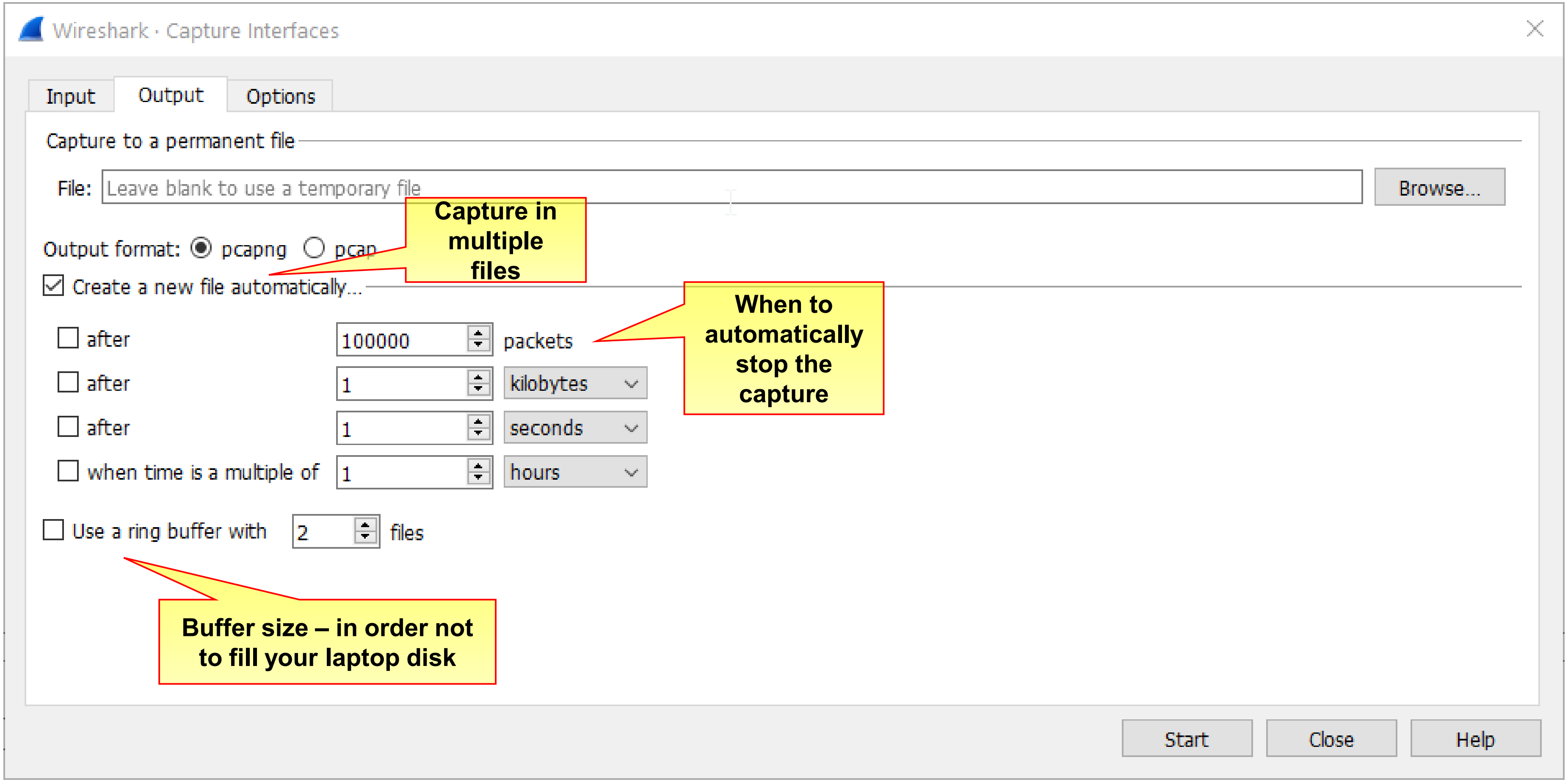
☑ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: Enter a capture filter ...

Capture all packets on the network

Capture filter

Wireshark – Capture Options



Wireshark · Capture Interfaces

Input | Output | Options

Capture to a permanent file

File: Leave blank to use a temporary file Browse...

Output format: ● pcapng ○ pcap

☑ Create a new file automatically...

□ after 100000 packets

□ after 1 kilobytes

□ after 1 seconds

□ when time is a multiple of 1 hours

□ Use a ring buffer with 2 files

Start Close Help

Capture in multiple files

When to automatically stop the capture

Buffer size – in order not to fill your laptop disk

Summary

- Packet analysis tools help analysts to identify traffic patterns present on a network
- Tcpdump can interpret and decode the protocols used at various layers of the OSI model
- Tcpdump and Dumpcap apply filters in the Berkley Packet Filter (BPF) format to reduce the amount of information an analyst must sift through
- Tshark utilizes human-friendly display filters instead of the BPF syntax
- Wireshark is a flexible tool used by analysts to understand network traffic
- Wireshark has access to thousands of display filters
- Wireshark can generate a variety of statistics

References

- Bejtlich, R. (2013). Chapter 6: Command Line Packet Analysis Tools. In The practice of network security monitoring understanding incident detection and response. San Francisco: No Starch Press.
- Bejtlich, R. (2013). Chapter 7: Graphical Packet Analysis Tools. In The practice of network security monitoring understanding incident detection and response. San Francisco: No Starch Press.
- (n.d.). Wireshark User’s Guide. Retrieved March 1, 2020, from https://www.wireshark.org/docs/wsug_html_chunked/