Analyze the network Packet

Wireshark : s an open-source network protocol analyzer. It uses a graphical user interface (GUI), which makes it easier to visualize network communications for packet analysis purposes. Wireshark has many features to explore that are beyond the scope of this course. You'll focus on how to use basic filtering to isolate network packets so that you can find what you need.

Display filters

Wireshark's display filters let you apply filters to packet capture files. This is helpful when you are inspecting packet captures with large volumes of information. Display filters will help you find specific information that's most relevant to your investigation. You can filter packets based on information such as protocols, IP addresses, ports, and virtually any other property found in a packet. Here, you'll focus on display filtering syntax and filtering for protocols, IP addresses, and ports.

Comparison operators

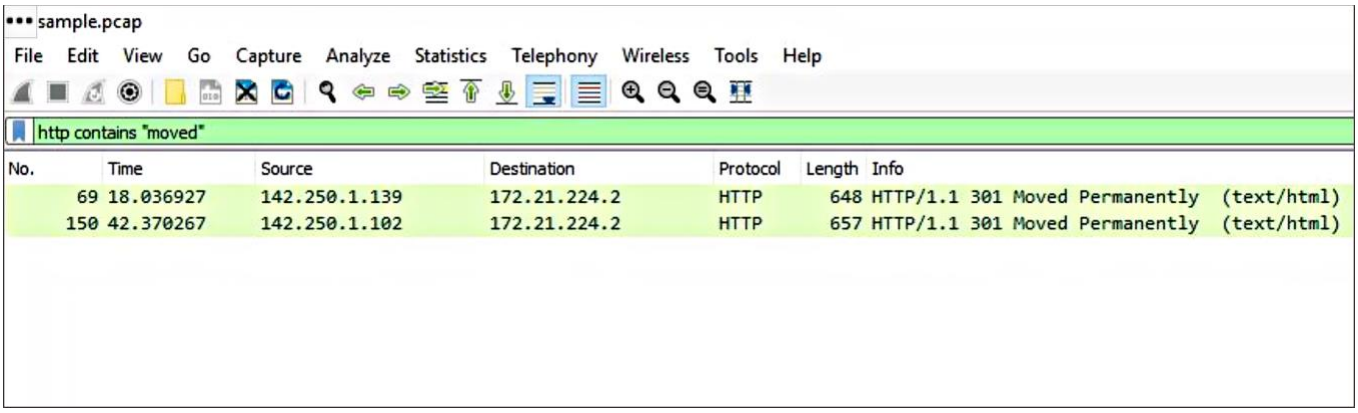
You can use different comparison operators to locate specific header fields and values. Comparison operators can be expressed using either abbreviations or symbols. For example, this filter using the == equal symbol in this filter ip.src == 8.8.8.8 is identical to using the eq abbreviation in this filter ip.src eq 8.8.8.8.

This table summarizes the different types of comparison operators you can use for display filtering.

Operator type	Symbol	Abbreviation
Equal	==	eq
Not equal	!=	ne
Greater than	>	gt
Less than	<	lt
Greater than or equal to	>=	ge
Less than or equal to	<=	le

Contains operator

The contains operator is used to filter packets that contain an exact match of a string of text. Here is an example of a filter that displays all HTTP streams that match the keyword "moved".

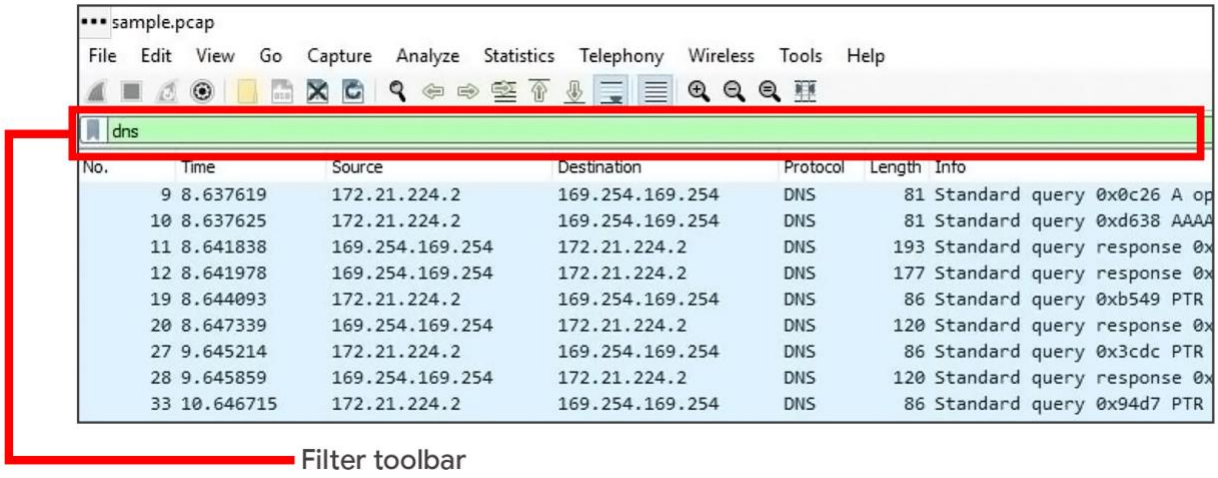


Matches operator

The matches operator is used to filter packets based on the regular expression (regex) that's specified. Regular expression is a sequence of characters that forms a pattern. You'll explore more about regular expressions later in this program.

Filter toolbar

You can apply filters to a packet capture using Wireshark's filter toolbar. In this example, dns is the applied filter, which means Wireshark will only display packets containing the DNS protocol.



Pro tip: Wireshark uses different colors to represent protocols. You can customize colors and create your own filters.

Filter for protocols

Protocol filtering is one of the simplest ways you can use display filters. You can simply enter the name of the protocol to filter. For example, to filter for DNS packets simply type dns in the filter toolbar. Here is a list of some protocols you can filter for:

- dns
- http
- ftp
- ssh
- arp
- telnet
- icmp

Filter for an IP address

You can use display filters to locate packets with a specific IP address.

For example, if you would like to filter packets that contain a specific IP address use ip.addr, followed by a space, the equal == comparison operator, and the IP address. Here is an example of a display filter that filters for the IP address 172.21.224.2:

ip.addr == 172.21.224.2

To filter for packets originating from a specific source IP address, you can use the ip.src filter. Here is an example that looks for the 10.10.10.10 source IP address:

ip.src == 10.10.10.10

To filter for packets delivered to a specific destination IP address, you can use the ip.dst filter. Here is an example that searches for the 4.4.4.4 destination IP address:

ip.dst == 4.4.4.4

Filter for a MAC address

You can also filter packets according to the Media Access Control (MAC) address. As a refresher, a MAC address is a unique alphanumeric identifier that is assigned to each physical device on a network.

Here's an example:

eth.addr == 00:70:f4:23:18:c4

Filter for ports

Port filtering is used to filter packets based on port numbers. This is helpful when you want to isolate specific types of traffic. DNS traffic uses TCP or UDP port 53 so this will list traffic related to DNS queries and responses only.

For example, if you would like to filter for a UDP port:

udp.port == 53

Likewise, you can filter for TCP ports as well:

tcp.port == 25

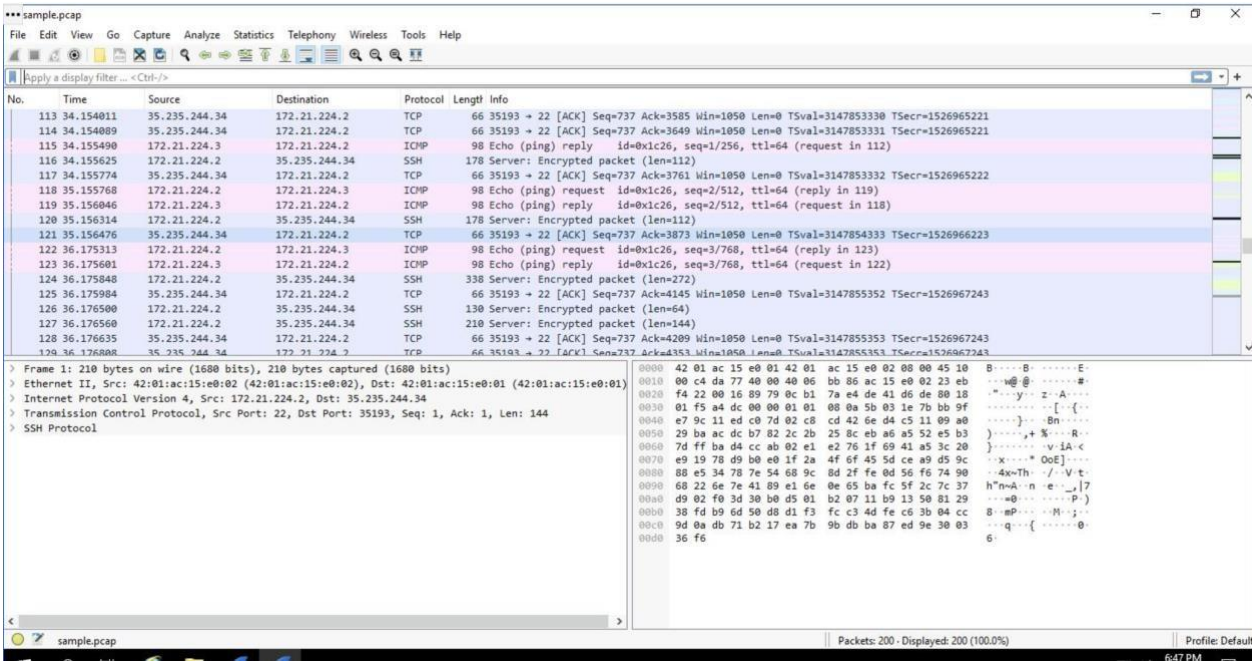
Now , we are going to analyze the packet using Wireshark for small.pcap file

1) Double-click on small.pcap to open it with wireshark as wireshark has been already installed

Columns available:

No. : The index number of the packet in this packet capture file
Time: The timestamp of the packet
Source: The source IP address
Destination: The destination IP address
Protocol: The protocol contained in the packet
Length: The total length of the packet
Info: Some infomation about the data in the packet (the payload) as interpreted by Wireshark

1) As soon as we open the file , the Wireshark page looks in the below manner

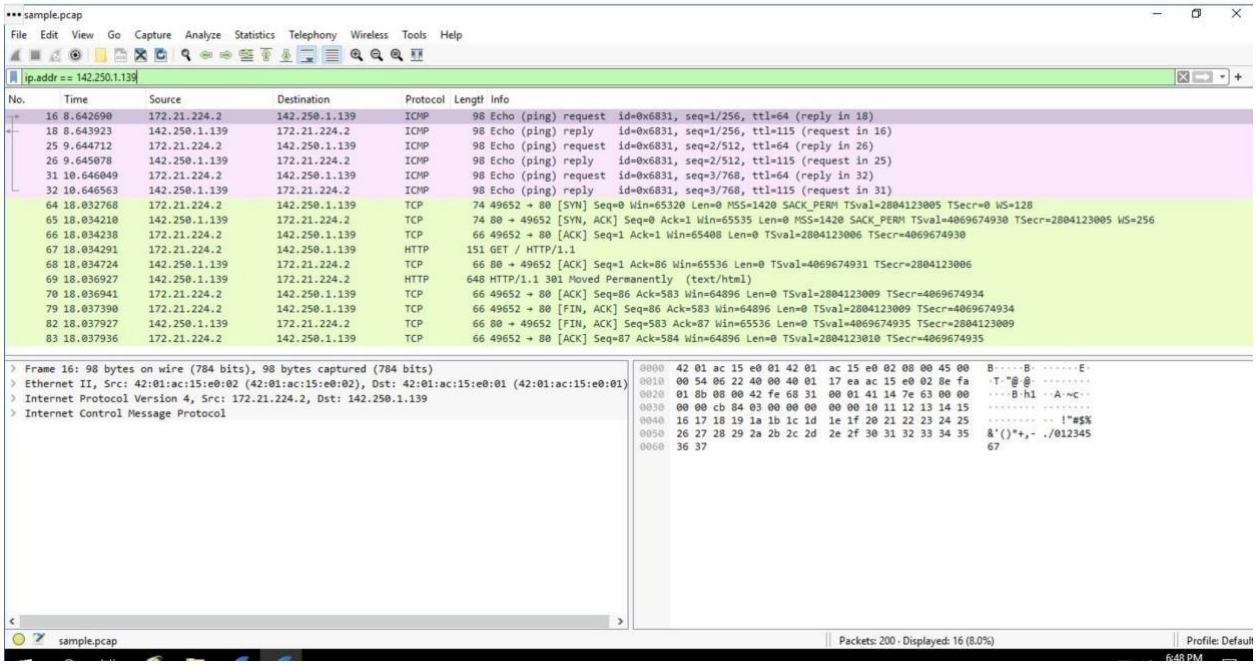


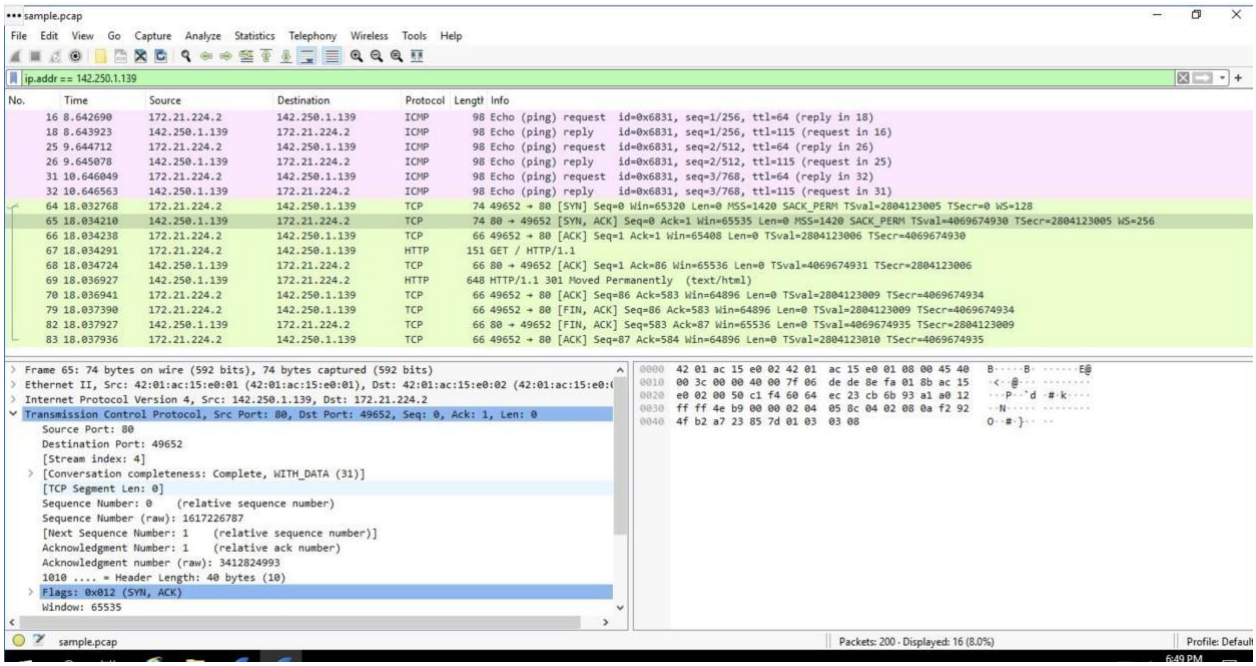
It displays all the data based on colored format which will enable us to differentiate based on the protocols. we can also apply filters to fetch the data easily. On every packet when we check

for subsets Eg: frame , ethernet, Internet protocol, Transmission control protocol/ UDP protocol, it will provide us with all the details to detect the deviations.

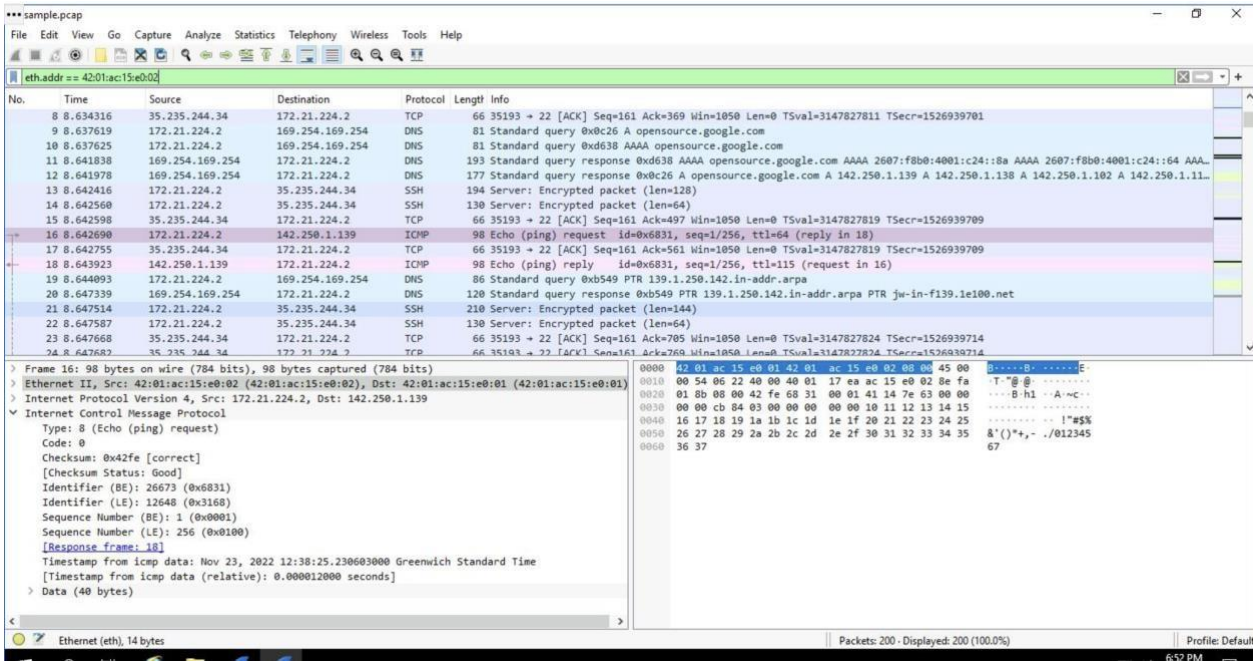
Below are some filters which we use on a regular basis.

Filtering Based on IP Address:

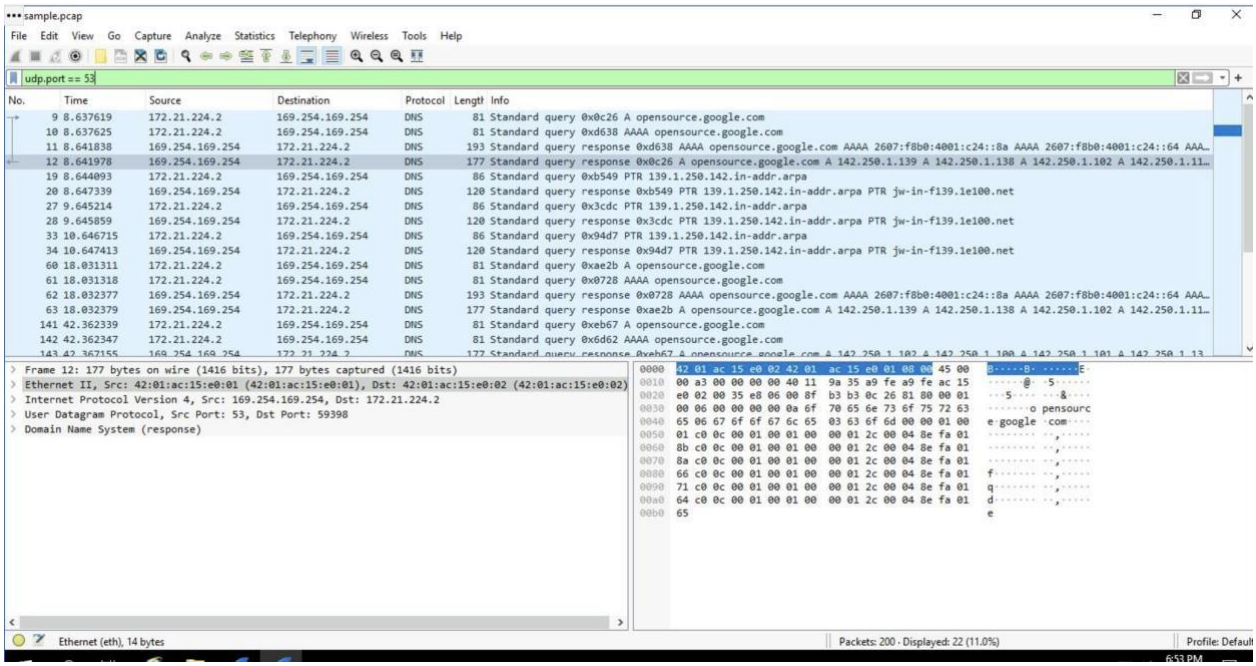




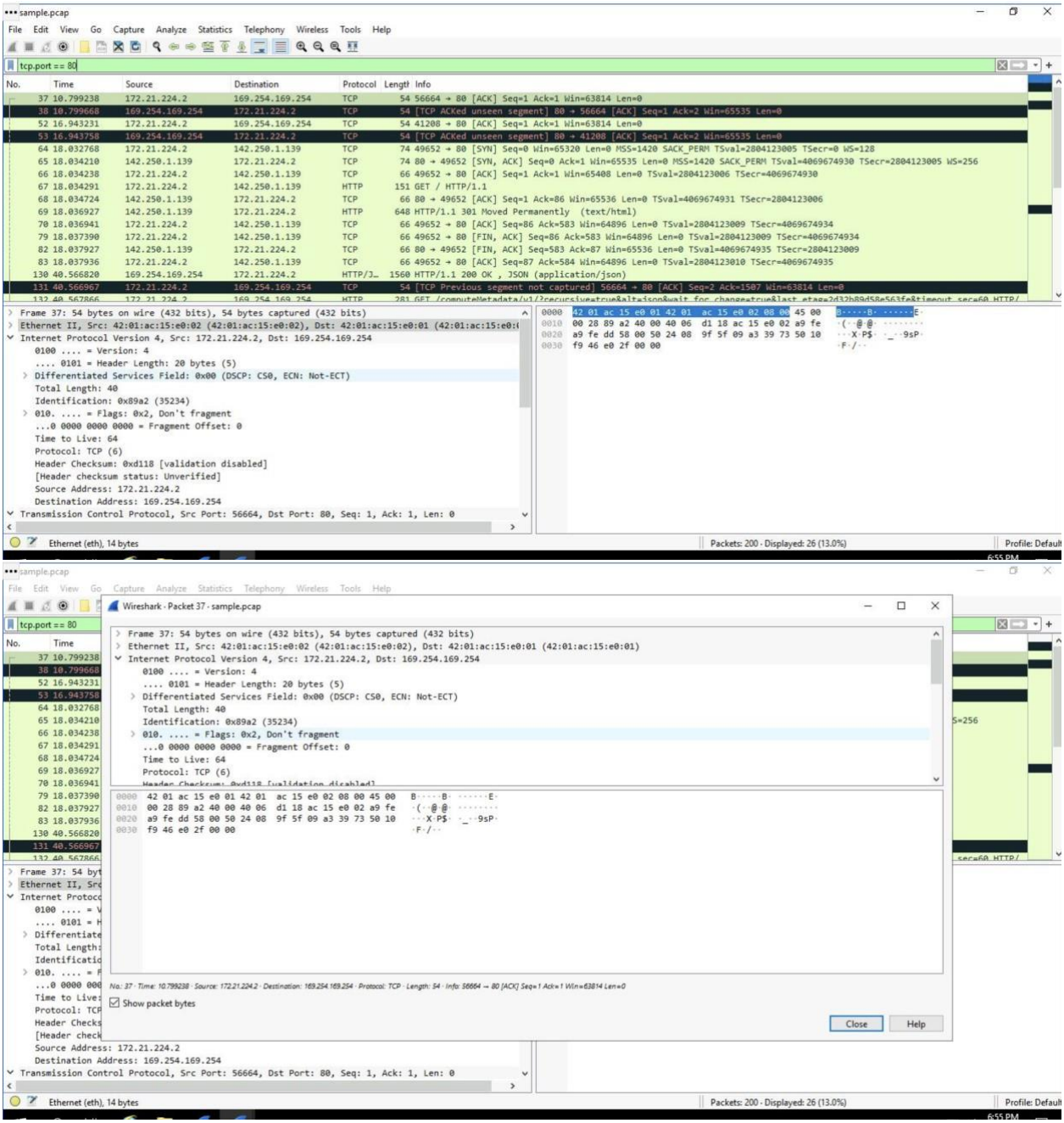
Filters to search based on MAC address:



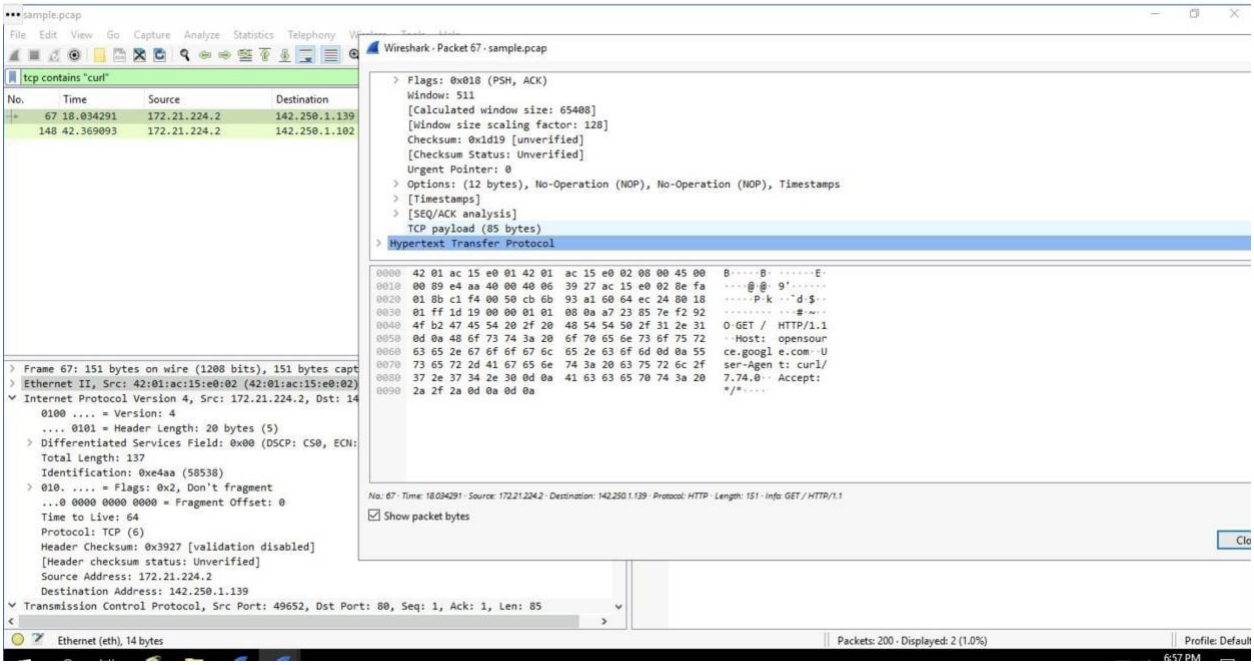
Filters to search for DNS protocol



Filter to search based on TCP protocol for HTTP requests:


sample.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 80
No. Time Source Destination Protocol Length Info
37 10.799238 172.21.224.2 169.254.169.254 TCP 54 56664 → 80 [ACK] Seq=1 Ack=1 Win=63814 Len=0
38 10.799668 169.254.169.254 172.21.224.2 TCP 54 [TCP ACKed unseen segment] 80 → 56664 [ACK] Seq=1 Ack=2 Win=65535 Len=0
52 16.943231 172.21.224.2 169.254.169.254 TCP 54 41208 → 80 [ACK] Seq=1 Ack=1 Win=63814 Len=0
53 16.943758 169.254.169.254 172.21.224.2 TCP 54 [TCP ACKed unseen segment] 80 → 41208 [ACK] Seq=1 Ack=2 Win=65535 Len=0
64 18.032768 172.21.224.2 142.250.1.139 TCP 74 49652 → 80 [SYN] Seq=0 Win=65320 Len=0 MSS=1420 SACK_PERM TSval=2804123005 TSecr=0 WS=128
65 18.034210 142.250.1.139 172.21.224.2 TCP 74 80 → 49652 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1420 SACK_PERM TSval=4069674930 TSecr=2804123005 WS=256
66 18.034238 172.21.224.2 142.250.1.139 TCP 66 49652 → 80 [ACK] Seq=1 Ack=1 Win=65408 Len=0 TSval=2804123006 TSecr=4069674930
67 18.034291 172.21.224.2 142.250.1.139 HTTP 151 GET / HTTP/1.1
68 18.034724 142.250.1.139 172.21.224.2 TCP 66 80 → 49652 [ACK] Seq=1 Ack=86 Win=65536 Len=0 TSval=4069674931 TSecr=2804123006
69 18.036927 142.250.1.139 172.21.224.2 HTTP 648 HTTP/1.1 301 Moved Permanently (text/html)
70 18.036941 172.21.224.2 142.250.1.139 TCP 66 49652 → 80 [ACK] Seq=86 Ack=583 Win=64896 Len=0 TSval=2804123009 TSecr=4069674934
79 18.037390 172.21.224.2 142.250.1.139 TCP 66 49652 → 80 [FIN, ACK] Seq=86 Ack=583 Win=64896 Len=0 TSval=2804123009 TSecr=4069674934
82 18.037927 142.250.1.139 172.21.224.2 TCP 66 80 → 49652 [FIN, ACK] Seq=583 Ack=87 Win=65536 Len=0 TSval=4069674935 TSecr=2804123009
83 18.037936 172.21.224.2 142.250.1.139 TCP 66 49652 → 80 [ACK] Seq=87 Ack=584 Win=64896 Len=0 TSval=2804123010 TSecr=4069674935
130 40.566820 169.254.169.254 172.21.224.2 HTTP/J… 1560 HTTP/1.1 200 OK , JSON (application/json)
131 40.566967 172.21.224.2 169.254.169.254 TCP 54 [TCP Previous segment not captured] 56664 → 80 [ACK] Seq=2 Ack=1507 Win=63814 Len=0
Frame 37: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 42:01:ac:15:e0:01 (42:01:ac:15:e0:01)
Internet Protocol Version 4, Src: 172.21.224.2, Dst: 169.254.169.254
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 40
Identification: 0x89a2 (35234)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: TCP (6)
Header Checksum: 0xd118 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.21.224.2
Destination Address: 169.254.169.254
Transmission Control Protocol, Src Port: 56664, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
0000 42 01 ac 15 e0 01 42 01 ac 15 e0 02 08 00 45 00
0010 00 28 89 a2 40 00 40 06 d1 18 ac 15 e0 02 a9 fe
0020 a9 fe dd 58 00 50 24 08 9f 5f 09 a3 39 73 50 10
0030 f9 46 e0 2f 00 00
Ethernet (eth), 14 bytes
Packets: 200 · Displayed: 26 (13.0%)
Profile: Default
6:55 PM

sample.pcap
Wireshark · Packet 37 · sample.pcap
Frame 37: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 42:01:ac:15:e0:01 (42:01:ac:15:e0:01)
Internet Protocol Version 4, Src: 172.21.224.2, Dst: 169.254.169.254
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 40
Identification: 0x89a2 (35234)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: TCP (6)
0000 42 01 ac 15 e0 01 42 01 ac 15 e0 02 08 00 45 00
0010 00 28 89 a2 40 00 40 06 d1 18 ac 15 e0 02 a9 fe
0020 a9 fe dd 58 00 50 24 08 9f 5f 09 a3 39 73 50 10
0030 f9 46 e0 2f 00 00
No.: 37 · Time: 10.799238 · Source: 172.21.224.2 · Destination: 169.254.169.254 · Protocol: TCP · Length: 54 · Info: 56664 → 80 [ACK] Seq=1 Ack=1 Win=63814 Len=0
Show packet bytes
Close
Help
Source Address: 172.21.224.2
Destination Address: 169.254.169.254
Transmission Control Protocol, Src Port: 56664, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
Ethernet (eth), 14 bytes
Packets: 200 · Displayed: 26 (13.0%)
Profile: Default
6:55 PM


sample.pcap
File Edit View Go Capture Analyze Statistics Telephony
tcp contains "curl"
No. Time Source Destination
67 18.034291 172.21.224.2 142.250.1.139
148 42.369093 172.21.224.2 142.250.1.102
Frame 67: 151 bytes on wire (1208 bits), 151 bytes capt
Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02)
Internet Protocol Version 4, Src: 172.21.224.2, Dst: 14
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN:
Total Length: 137
Identification: 0xe4aa (58538)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: TCP (6)
Header Checksum: 0x3927 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.21.224.2
Destination Address: 142.250.1.139
Transmission Control Protocol, Src Port: 49652, Dst Port: 80, Seq: 1, Ack: 1, Len: 85
Wireshark · Packet 67 · sample.pcap
Flags: 0x018 (PSH, ACK)
Window: 511
[Calculated window size: 65408]
[Window size scaling factor: 128]
Checksum: 0x1d19 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[Timestamps]
[SEQ/ACK analysis]
TCP payload (85 bytes)
Hypertext Transfer Protocol
0000 42 01 ac 15 e0 01 42 01 ac 15 e0 02 08 00 45 00 B·····B· ·····E·
0010 00 89 e4 aa 40 00 40 06 39 27 ac 15 e0 02 8e fa ····@·@· 9'·····
0020 01 8b c1 f4 00 50 cb 6b 93 a1 60 64 ec 24 80 18 ·····P·k ··`d·$··
0030 01 ff 1d 19 00 00 01 01 08 0a a7 23 85 7e f2 92 ·····#·~··
0040 4f b2 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 O·GET / HTTP/1.1
0050 0d 0a 48 6f 73 74 3a 20 6f 70 65 6e 73 6f 75 72 ··Host: opensour
0060 63 65 2e 67 6f 6f 67 6c 65 2e 63 6f 6d 0d 0a 55 ce.googl e.com··U
0070 73 65 72 2d 41 67 65 6e 74 3a 20 63 75 72 6c 2f ser-Agen t: curl/
0080 37 2e 37 34 2e 30 0d 0a 41 63 63 65 70 74 3a 20 7.74.0·· Accept:
0090 2a 2f 2a 0d 0a 0d 0a */*····
No.: 67 · Time: 18.034291 · Source: 172.21.224.2 · Destination: 142.250.1.139 · Protocol: HTTP · Length: 151 · Info: GET / HTTP/1.1
Show packet bytes
Ethernet (eth), 14 bytes
Packets: 200 · Displayed: 2 (1.0%)
Profile: Default
6:57 PM