How to Use Wireshark to Capture Network Traffic (2024)

Whether an ethical hacker or SOC analyst, you must understand how to use Wireshark to capture network traffic. This powerful open-source network protocol analyzer can turn bytes on the wire into readable packets you can analyze.

Wireshark can troubleshoot networks, extract sensitive information from network packets, and identify cyber attacks. This comprehensive guide will show you how to use this powerful tool, when it should be used, and how to capture and analyze network traffic. You will also see how to use Wireshark in the real world to perform common red and blue team activities.

Let's get our keyboards ready and start capturing network traffic with Wireshark!

What Is Wireshark?

Wireshark is an open-source network protocol analyzer widely used to troubleshoot network issues, analyze network packets, and perform security auditing. However, its true power comes from its unmatched support for over 3000 network protocols. This lets the tool transform bytes on a network into visual network packets that you can efficiently analyze using its graphical interface and various features.

Wireshark features:

- **Packet capture**: Wireshark can capture traffic using your network interface.
- **Protocol analysis**: Wireshark supports the decoding and analyzing of over 3000 network protocols so you can understand their structure and content.
- **Packet filtering**: Wireshark includes powerful display and capture filters that filter network traffic.
- **Network packet reconstruction**: Wireshark can reconstruct network packets to display application-level protocols so you can see web pages, images, or other application data.
- **Network traffic statistics and visualizations**: Wireshark provides statistical data and visualizations of network traffic.
- **Customization**: You add functionality or customization to Wireshark through its plugins.

Analyzing network traffic is crucial in cyber security. It allows you to perform blue team activities (such as detecting a cyber attack, analyzing malware, and collecting forensic evidence) and red team tasks (like stealing sensitive information, extracting files, and testing your malware's network communications).

Wireshark lets you do all these things and more! Let's find out how.

Wireshark can be used for packet capture or as a packet sniffer. Packet capture is when you record the network packets traveling across the network to and from your machine. It is commonly used to troubleshoot network problems or perform blue team activities.

Sniffing involves capturing network packets traveling across the network that are not intended for your machine. You can set your capture interface to promiscuous mode and use Wireshark to perform sniffing.

When Should Wireshark Be Used?

Wireshark should be used whenever you need to analyze network packets and understand the network communication happening across a network.

However, its most powerful feature is its ability to perform network protocol analysis. The tool supports many network protocols and can transform network packets into human-readable data. Using Wireshark's built-in features, you can perform statistical analysis on this data, follow TCP streams, read sensitive information, extract full files, and much more.

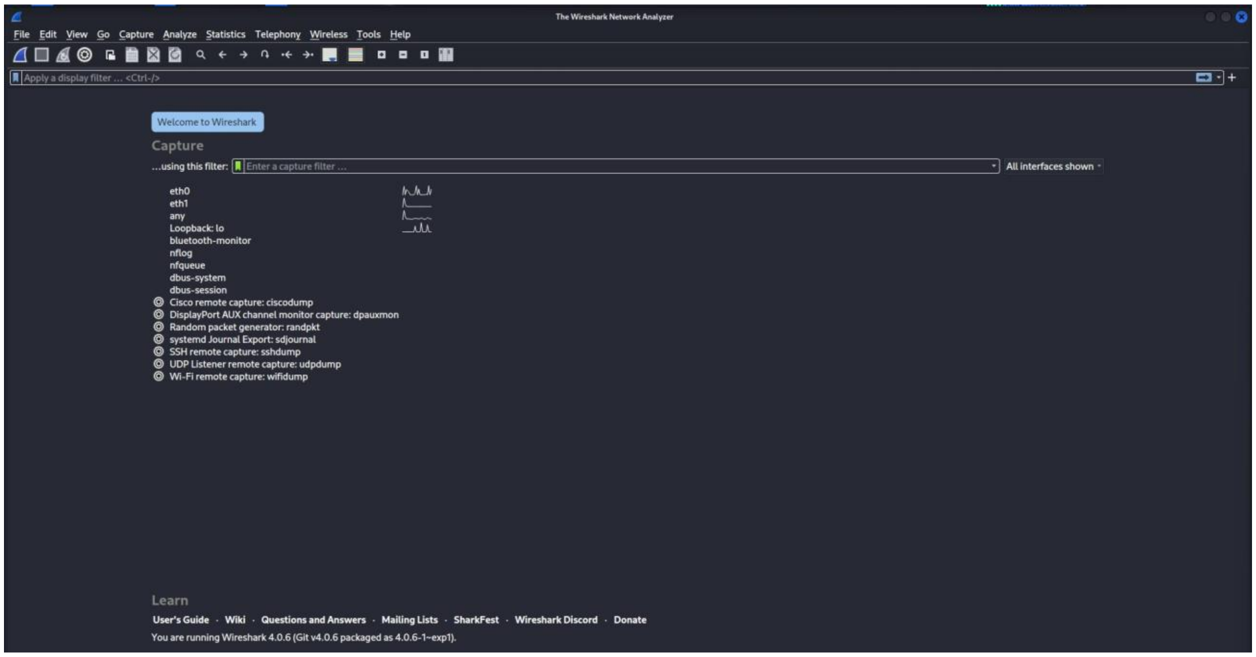
Capturing Network Traffic

To use Wireshark, you can capture network traffic or upload a packet capture (pcap) file. Let's look at how to capture network traffic.

Wireharsk comes pre-installed on Kali Linux, so to capture network traffic in Wireshark, you only need to perform three steps:

Step 1: Open the Wireshark application

First, open the Wireshark application by clicking on the icon. Wireshark's welcome screen will greet you.

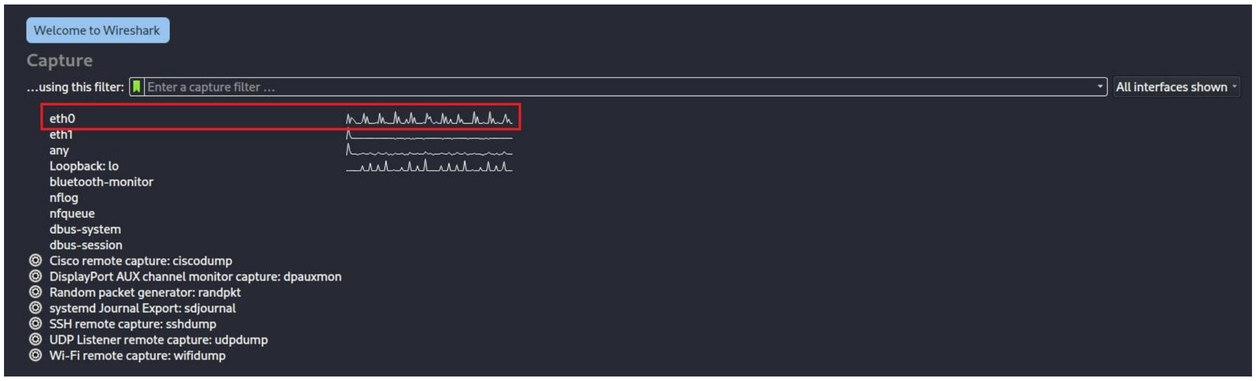

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate

```
eth0
eth1
any
Loopback: lo
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
Wi-Fi remote capture: wifidump
```

You are running Wireshark 4.0.6 (Git v4.0.6 packaged as 4.0.6-1~exp1).

Step 2: Select a network interface to capture traffic on

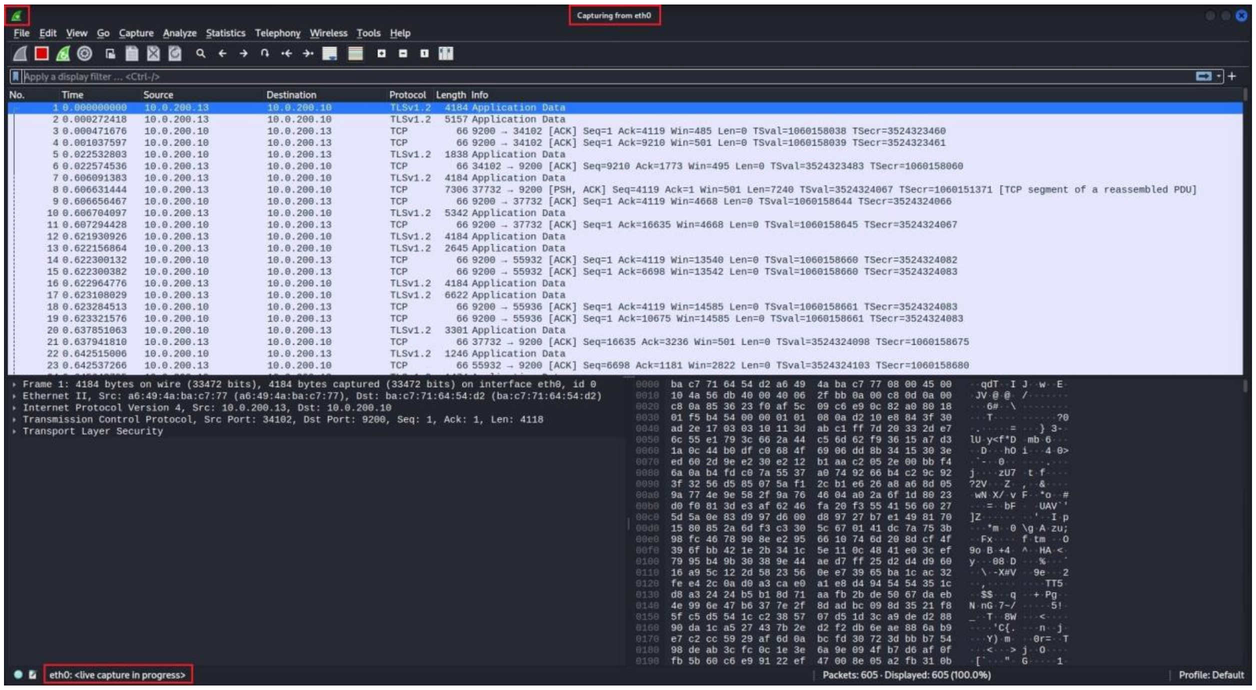
From the welcome screen, select which of your network interfaces you want to capture traffic on. The line to the right of the interface indicates the network traffic flow passing through that interface.



```
eth0
eth1
any
Loopback: lo
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
Wi-Fi remote capture: wifidump
```

Step 3: Stopping the packet capture

Double-clicking on the capture interface you want to capture traffic to start capturing packets.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
eth0: <live capture in progress>
Profile: Default

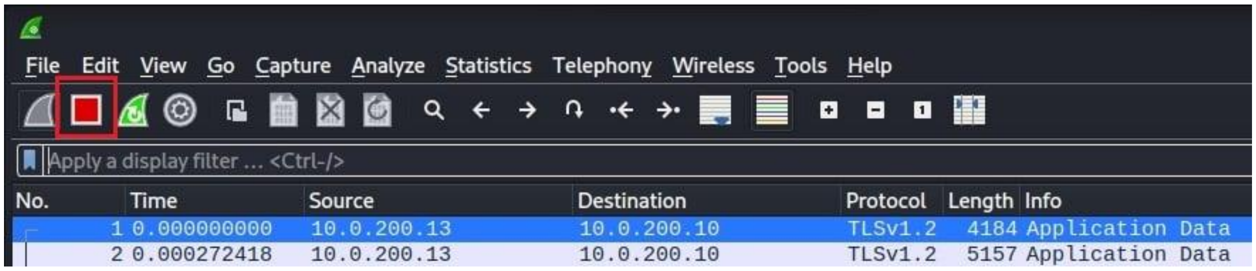
```
No.   Time         Source        Destination   Protocol Length Info
    1 0.000000000  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
    2 0.000272418  10.0.200.13   10.0.200.10   TLSv1.2  5157 Application Data
    3 0.000471676  10.0.200.10   10.0.200.13   TCP        66 9200 → 34102 [ACK] Seq=1 Ack=4119 Win=485 Len=0 TSval=1060158038 TSecr=3524323460
    4 0.001037597  10.0.200.10   10.0.200.13   TCP        66 9200 → 34102 [ACK] Seq=1 Ack=9210 Win=501 Len=0 TSval=1060158039 TSecr=3524323461
    5 0.022532803  10.0.200.10   10.0.200.13   TLSv1.2  1838 Application Data
    6 0.022574536  10.0.200.13   10.0.200.10   TCP        66 34102 → 9200 [ACK] Seq=9210 Ack=1773 Win=495 Len=0 TSval=3524323483 TSecr=1060158060
    7 0.606091383  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
    8 0.606631444  10.0.200.13   10.0.200.10   TCP      7306 37732 → 9200 [PSH, ACK] Seq=4119 Ack=1 Win=501 Len=7240 TSval=3524324067 TSecr=1060151371 [TCP segment of a reassembled PDU]
    9 0.606656467  10.0.200.10   10.0.200.13   TCP        66 9200 → 37732 [ACK] Seq=1 Ack=4119 Win=4668 Len=0 TSval=1060158644 TSecr=3524324066
   10 0.606704097  10.0.200.13   10.0.200.10   TLSv1.2  5342 Application Data
   11 0.607294428  10.0.200.10   10.0.200.13   TCP        66 9200 → 37732 [ACK] Seq=1 Ack=16635 Win=4668 Len=0 TSval=1060158645 TSecr=3524324067
   12 0.621930926  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
   13 0.622156864  10.0.200.13   10.0.200.10   TLSv1.2  2645 Application Data
   14 0.622300132  10.0.200.10   10.0.200.13   TCP        66 9200 → 55932 [ACK] Seq=1 Ack=4119 Win=13540 Len=0 TSval=1060158660 TSecr=3524324082
   15 0.622300382  10.0.200.10   10.0.200.13   TCP        66 9200 → 55932 [ACK] Seq=1 Ack=6698 Win=13542 Len=0 TSval=1060158660 TSecr=3524324083
   16 0.622964776  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
   17 0.623108029  10.0.200.13   10.0.200.10   TLSv1.2  6622 Application Data
   18 0.623284513  10.0.200.10   10.0.200.13   TCP        66 9200 → 55936 [ACK] Seq=1 Ack=4119 Win=14585 Len=0 TSval=1060158661 TSecr=3524324083
   19 0.623321576  10.0.200.10   10.0.200.13   TCP        66 9200 → 55936 [ACK] Seq=1 Ack=10675 Win=14585 Len=0 TSval=1060158661 TSecr=3524324083
   20 0.637851063  10.0.200.10   10.0.200.13   TLSv1.2  3301 Application Data
   21 0.637941810  10.0.200.13   10.0.200.10   TCP        66 37732 → 9200 [ACK] Seq=16635 Ack=3236 Win=501 Len=0 TSval=3524324098 TSecr=1060158675
   22 0.642515006  10.0.200.10   10.0.200.13   TLSv1.2  1246 Application Data
   23 0.642537266  10.0.200.13   10.0.200.10   TCP        66 55932 → 9200 [ACK] Seq=6698 Ack=1181 Win=2822 Len=0 TSval=3524324103 TSecr=1060158680
```

```
› Frame 1: 4184 bytes on wire (33472 bits), 4184 bytes captured (33472 bits) on interface eth0, id 0
› Ethernet II, Src: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77), Dst: ba:c7:71:64:54:d2 (ba:c7:71:64:54:d2)
› Internet Protocol Version 4, Src: 10.0.200.13, Dst: 10.0.200.10
› Transmission Control Protocol, Src Port: 34102, Dst Port: 9200, Seq: 1, Ack: 1, Len: 4118
› Transport Layer Security
```

```
0000  ba c7 71 64 54 d2 a6 49  4a ba c7 77 08 00 45 00   ··qdT··I J··w··E·
0010  10 4a 56 db 40 00 40 06  2f bb 0a 00 c8 0d 0a 00   ·JV·@·@· /·······
0020  c8 0a 85 36 23 f0 af 5c  09 c6 e9 0c 82 a0 80 18   ···6#··\ ········
0030  01 f5 b4 54 00 00 01 01  08 0a d2 10 e8 84 3f 30   ···T···· ······?0
0040  ad 2e 17 03 03 10 11 3d  ab c1 ff 7d 20 33 2d e7   ·.·····= ···} 3-·
0050  6c 55 e1 79 3c 66 2a 44  c5 6d 62 f9 36 15 a7 d3   lU·y<f*D ·mb·6···
0060  1a 0c 44 b0 df c0 68 4f  69 06 dd 8b 34 15 30 3e   ··D···hO i···4·0>
0070  ed 60 2d 9e e2 30 e2 12  b1 aa c2 05 2e 00 bb f4   ·`-··0·· ····.···
0080  6a 0a b4 fd c0 7a 55 37  a0 74 92 66 b4 c2 9c 92   j····zU7 ·t·f····
0090  3f 32 56 d5 85 07 5a f1  2c b1 e6 26 a8 a6 8d 05   ?2V···Z· ,··&····
00a0  9a 77 4e 9e 58 2f 9a 76  46 04 a0 2a 6f 1d 80 23   ·wN·X/·v F··*o··#
00b0  d0 f0 81 3d e3 af 62 46  fa 20 f3 55 41 56 60 27   ···=··bF · ·UAV`'
00c0  5d 5a 0e 83 d9 97 d6 00  d8 97 27 b7 e1 49 81 70   ]Z······ ··'··I·p
00d0  15 80 85 2a 6d f3 c3 30  5c 67 01 41 dc 7a 75 3b   ···*m··0 \g·A·zu;
00e0  98 fc 46 78 90 8e e2 95  66 10 74 6d 20 8d cf 4f   ··Fx···· f·tm ··O
00f0  39 6f bb 42 1e 2b 34 1c  5e 11 0c 48 41 e0 3c ef   9o·B·+4· ^··HA·<·
0100  79 95 b4 9b 30 38 9e 44  ae d7 ff 25 d2 d4 d9 60   y···08·D ···%···`
0110  16 a9 5c 12 2d 58 23 56  0e e7 39 65 ba 1c ac 32   ··\·-X#V ··9e···2
0120  fe e4 2c 0a d0 a3 ca e0  a1 e8 d4 94 54 54 35 1c   ··,····· ····TT5·
0130  d8 a3 24 24 b5 b1 8d 71  aa fb 2b de 50 67 da eb   ··$$···q ··+·Pg··
0140  4e 99 6e 47 b6 37 7e 2f  8d ad bc 09 8d 35 21 f8   N·nG·7~/ ·····5!·
0150  5f c5 d5 54 1c c2 38 57  07 d5 1d 3c a9 de d2 88   _··T··8W ···<····
0160  90 da 1c a5 27 43 7b 2e  d2 f2 db 6e ae 88 6a b9   ····'C{. ···n··j·
0170  e7 c2 cc 59 29 af 6d 0a  bc fd 30 72 3d bb b7 54   ···Y)·m· ··0r=··T
0180  98 de ab 3c fc 0c 1e 3e  6a 9e 09 4f b7 d6 af 0f   ···<···> j··O····
0190  fb 5b 60 c6 e9 91 22 ef  47 00 8e 05 a2 fb 31 0b   ·[`···". G·····1·
```

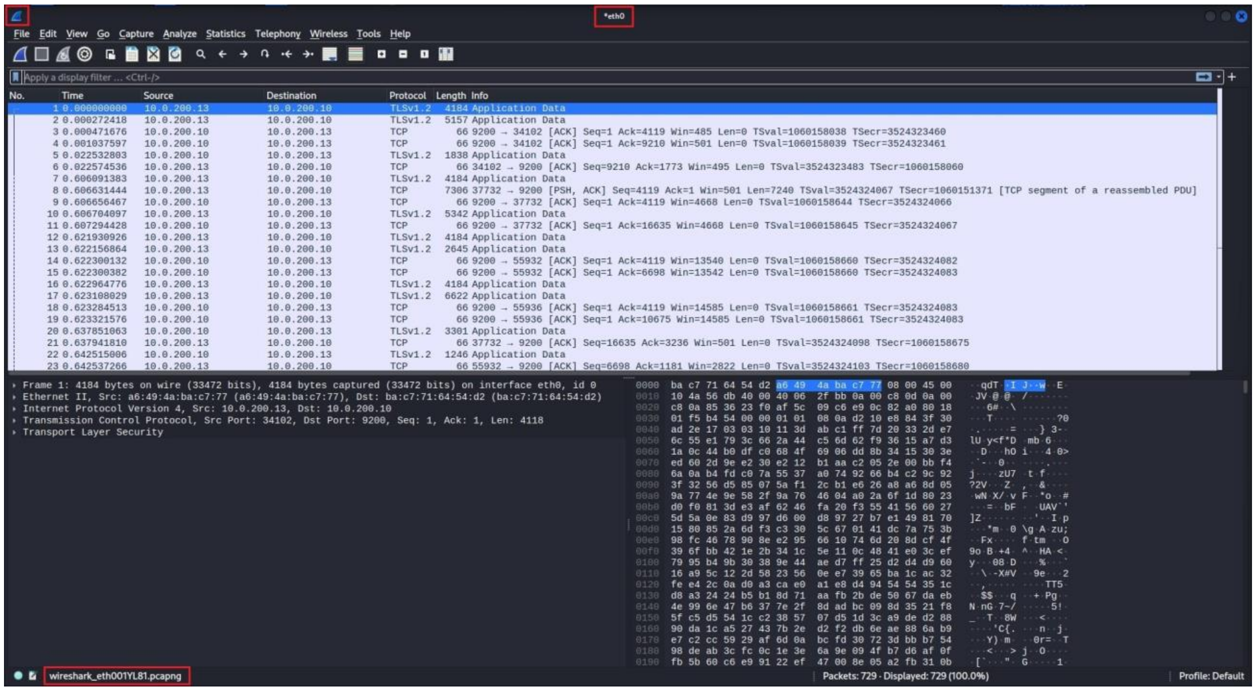
Packets: 605 · Displayed: 605 (100.0%)

To stop capture packets, click on the red Stop Capture button.


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

```
No.   Time          Source         Destination    Protocol  Length Info
    1 0.000000000   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
    2 0.000272418   10.0.200.13    10.0.200.10    TLSv1.2   5157 Application Data
```

This will save the packet capture into a temporary file and allow you to perform your analysis.



*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

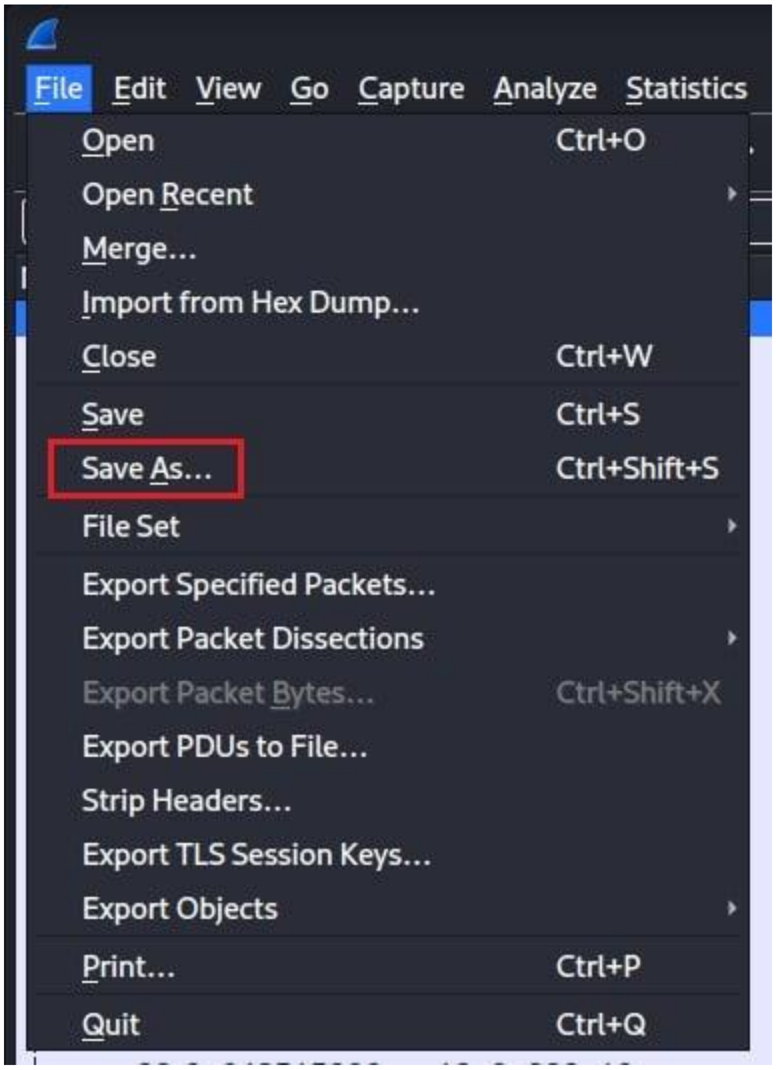
```
No.   Time         Source        Destination   Protocol Length Info
    1 0.000000000  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
    2 0.000272418  10.0.200.13   10.0.200.10   TLSv1.2  5157 Application Data
    3 0.000471676  10.0.200.10   10.0.200.13   TCP        66 9200 → 34102 [ACK] Seq=1 Ack=4119 Win=485 Len=0 TSval=1060158038 TSecr=3524323460
    4 0.001037597  10.0.200.10   10.0.200.13   TCP        66 9200 → 34102 [ACK] Seq=1 Ack=9210 Win=501 Len=0 TSval=1060158039 TSecr=3524323461
    5 0.022532803  10.0.200.10   10.0.200.13   TLSv1.2  1838 Application Data
    6 0.022574536  10.0.200.13   10.0.200.10   TCP        66 34102 → 9200 [ACK] Seq=9210 Ack=1773 Win=495 Len=0 TSval=3524323483 TSecr=1060158060
    7 0.606091383  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
    8 0.606631444  10.0.200.13   10.0.200.10   TCP      7306 37732 → 9200 [PSH, ACK] Seq=4119 Ack=1 Win=501 Len=7240 TSval=3524324067 TSecr=1060151371 [TCP segment of a reassembled PDU]
    9 0.606656467  10.0.200.10   10.0.200.13   TCP        66 9200 → 37732 [ACK] Seq=1 Ack=4119 Win=4668 Len=0 TSval=1060158644 TSecr=3524324066
   10 0.606704097  10.0.200.13   10.0.200.10   TLSv1.2  5342 Application Data
   11 0.607294428  10.0.200.10   10.0.200.13   TCP        66 9200 → 37732 [ACK] Seq=1 Ack=16635 Win=4668 Len=0 TSval=1060158645 TSecr=3524324067
   12 0.621930926  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
   13 0.622156864  10.0.200.13   10.0.200.10   TLSv1.2  2645 Application Data
   14 0.622300132  10.0.200.10   10.0.200.13   TCP        66 9200 → 55932 [ACK] Seq=1 Ack=4119 Win=13540 Len=0 TSval=1060158660 TSecr=3524324082
   15 0.622300382  10.0.200.10   10.0.200.13   TCP        66 9200 → 55932 [ACK] Seq=1 Ack=6698 Win=13542 Len=0 TSval=1060158660 TSecr=3524324083
   16 0.622964776  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
   17 0.623108029  10.0.200.13   10.0.200.10   TLSv1.2  6622 Application Data
   18 0.623284513  10.0.200.10   10.0.200.13   TCP        66 9200 → 55936 [ACK] Seq=1 Ack=4119 Win=14585 Len=0 TSval=1060158661 TSecr=3524324083
   19 0.623321576  10.0.200.10   10.0.200.13   TCP        66 9200 → 55936 [ACK] Seq=1 Ack=10675 Win=14585 Len=0 TSval=1060158661 TSecr=3524324083
   20 0.637851063  10.0.200.10   10.0.200.13   TLSv1.2  3301 Application Data
   21 0.637941810  10.0.200.13   10.0.200.10   TCP        66 37732 → 9200 [ACK] Seq=16635 Ack=3236 Win=501 Len=0 TSval=3524324098 TSecr=1060158675
   22 0.642515006  10.0.200.10   10.0.200.13   TLSv1.2  1246 Application Data
   23 0.642537266  10.0.200.13   10.0.200.10   TCP        66 55932 → 9200 [ACK] Seq=6698 Ack=1181 Win=2822 Len=0 TSval=3524324103 TSecr=1060158680
```

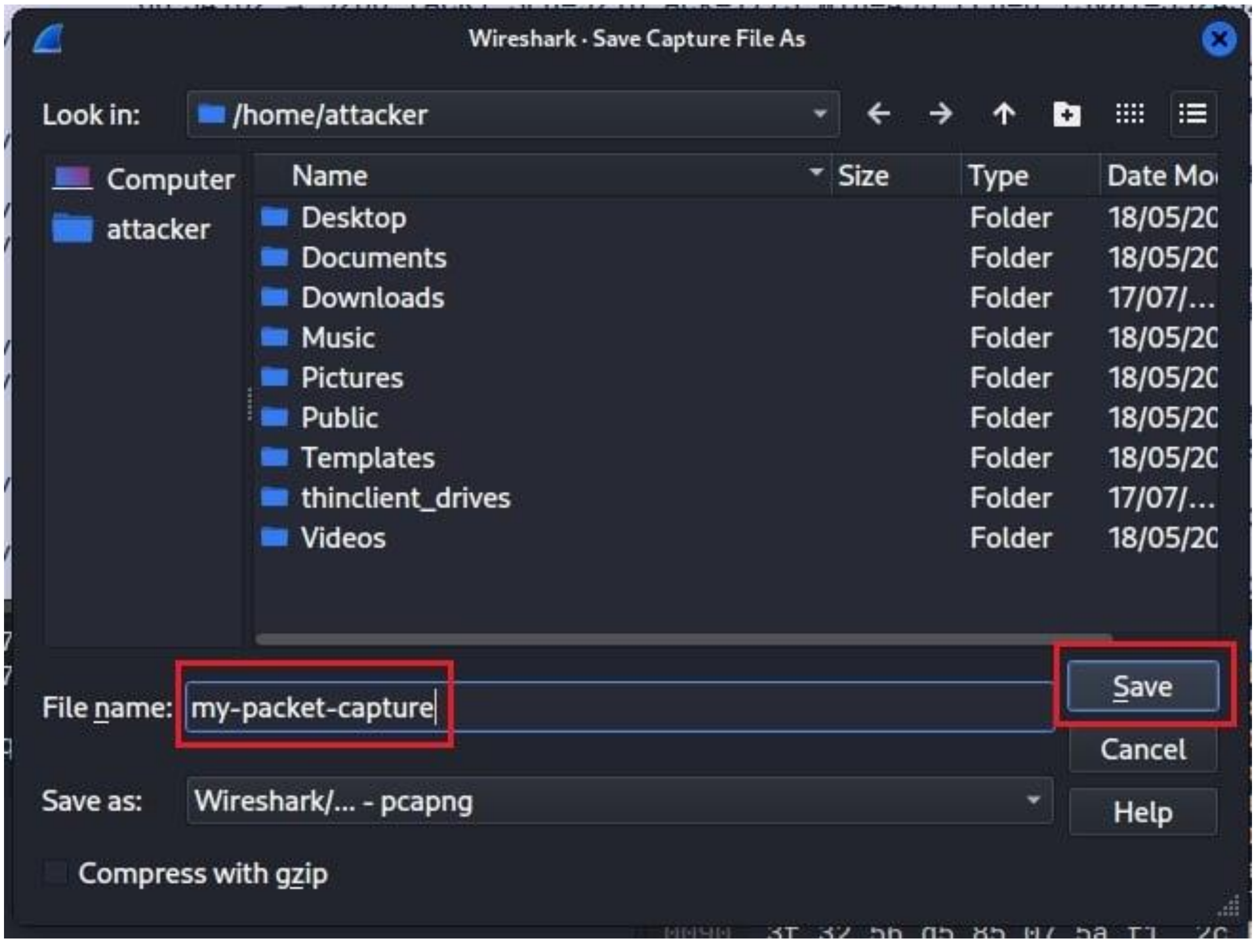
```
› Frame 1: 4184 bytes on wire (33472 bits), 4184 bytes captured (33472 bits) on interface eth0, id 0
› Ethernet II, Src: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77), Dst: ba:c7:71:64:54:d2 (ba:c7:71:64:54:d2)
› Internet Protocol Version 4, Src: 10.0.200.13, Dst: 10.0.200.10
› Transmission Control Protocol, Src Port: 34102, Dst Port: 9200, Seq: 1, Ack: 1, Len: 4118
› Transport Layer Security
```

```
0000  ba c7 71 64 54 d2 a6 49  4a ba c7 77 08 00 45 00
0010  10 4a 56 db 40 00 40 06  2f bb 0a 00 c8 0d 0a 00
0020  c8 0a 85 36 23 f0 af 5c  09 c6 e9 0c 82 a0 80 18
0030  01 f5 b4 54 00 00 01 01  08 0a d2 10 e8 84 3f 30
0040  ad 2e 17 03 03 10 11 3d  ab c1 ff 7d 20 33 2d e7
0050  6c 55 e1 79 3c 66 2a 44  c5 6d 62 f9 36 15 a7 d3
0060  1a 0c 44 b0 df c0 68 4f  69 06 dd 8b 34 15 30 3e
0070  ed 60 2d 9e e2 30 e2 12  b1 aa c2 05 2e 00 bb f4
0080  6a 0a b4 fd c0 7a 55 37  a0 74 92 66 b4 c2 9c 92
0090  3f 32 56 d5 85 07 5a f1  2c b1 e6 26 a8 a6 8d 05
00a0  9a 77 4e 9e 58 2f 9a 76  46 04 a0 2a 6f 1d 80 23
00b0  d0 f0 81 3d e3 af 62 46  fa 20 f3 55 41 56 60 27
00c0  5d 5a 0e 83 d9 97 d6 00  d8 97 27 b7 e1 49 81 70
00d0  15 80 85 2a 6d f3 c3 30  5c 67 01 41 dc 7a 75 3b
00e0  98 fc 46 78 90 8e e2 95  66 10 74 6d 20 8d cf 4f
00f0  39 6f bb 42 1e 2b 34 1c  5e 11 0c 48 41 e0 3c ef
0100  79 95 b4 9b 30 38 9e 44  ae d7 ff 25 d2 d4 d9 60
0110  16 a9 5c 12 2d 58 23 56  0e e7 39 65 ba 1c ac 32
0120  fe e4 2c 0a d0 a3 ca e0  a1 e8 d4 94 54 54 35 1c
0130  d8 a3 24 24 b5 b1 8d 71  aa fb 2b de 50 67 da eb
0140  4e 99 6e 47 b6 37 7e 2f  8d ad bc 09 8d 35 21 f8
0150  5f c5 d5 54 1c c2 38 57  07 d5 1d 3c a9 de d2 88
0160  90 da 1c a5 27 43 7b 2e  d2 f2 db 6e ae 88 6a b9
0170  e7 c2 cc 59 29 af 6d 0a  bc fd 30 72 3d bb b7 54
0180  98 de ab 3c fc 0c 1e 3e  6a 9e 09 4f b7 d6 af 0f
0190  fb 5b 60 c6 e9 91 22 ef  47 00 8e 05 a2 fb 31 0b
```

wireshark_eth001YL81.pcapng
Packets: 729 · Displayed: 729 (100.0%)
Profile: Default

To save your data into a specific file. Select File > Save As, then enter the location and name of the file under which you want to save this data.

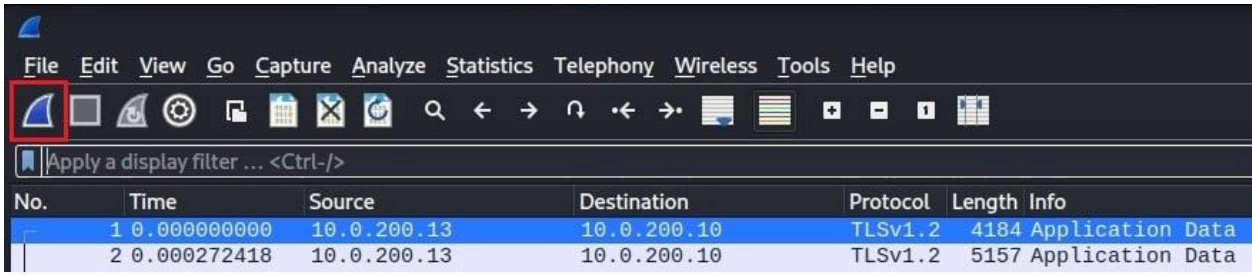

File
Edit
View
Go
Capture
Analyze
Statistics
Open Ctrl+O
Open Recent
Merge…
Import from Hex Dump…
Close Ctrl+W
Save Ctrl+S
Save As… Ctrl+Shift+S
File Set
Export Specified Packets…
Export Packet Dissections
Export Packet Bytes… Ctrl+Shift+X
Export PDUs to File…
Strip Headers…
Export TLS Session Keys…
Export Objects
Print… Ctrl+P
Quit Ctrl+Q



Name	Size	Type	Date Mo
Desktop		Folder	18/05/20
Documents		Folder	18/05/20
Downloads		Folder	17/07/...
Music		Folder	18/05/20
Pictures		Folder	18/05/20
Public		Folder	18/05/20
Templates		Folder	18/05/20
thinclient_drives		Folder	17/07/...
Videos		Folder	18/05/20

File name: my-packet-capture
Save as: Wireshark/... - pcapng
Compress with gzip

To start a new packet capture, select the Start Capture button. Any unsaved capture data will be automatically deleted.



```
No.  Time          Source        Destination   Protocol  Length Info
   1 0.000000000   10.0.200.13   10.0.200.10   TLSv1.2   4184 Application Data
   2 0.000272418   10.0.200.13   10.0.200.10   TLSv1.2   5157 Application Data
```

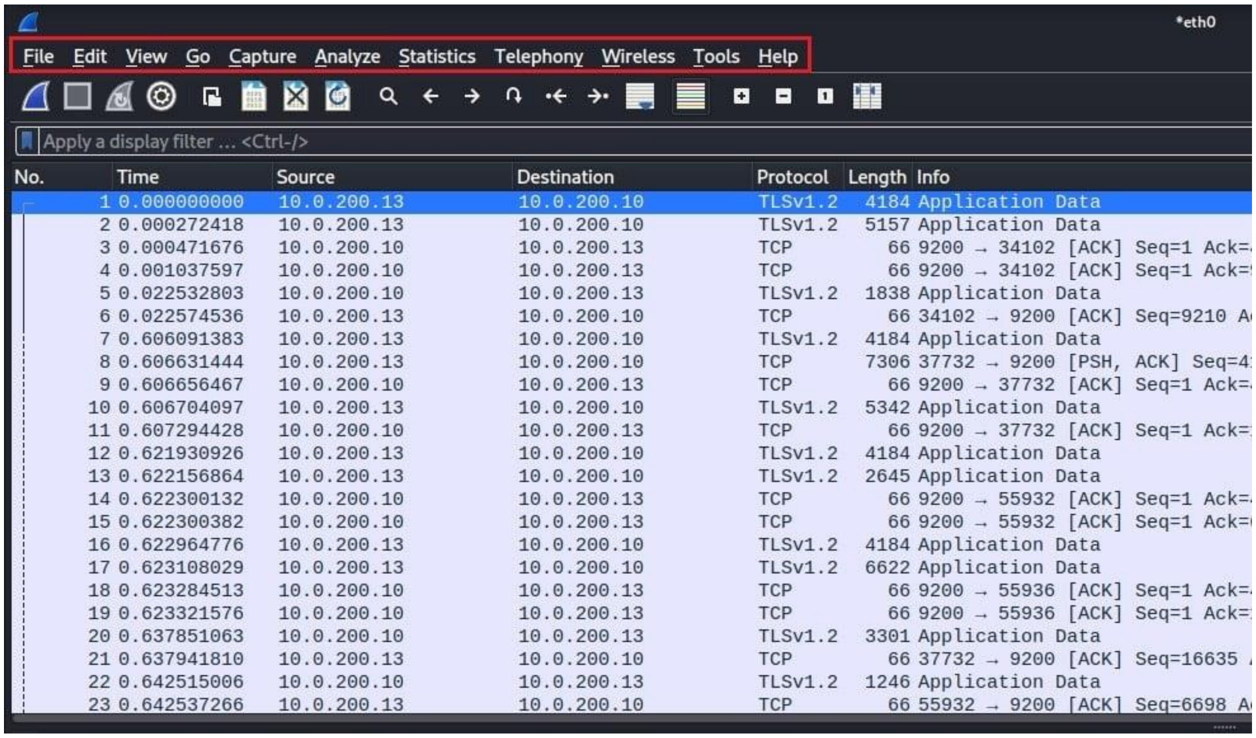
When capturing network traffic with Wireshark, you can use the tool's **capture filters** to limit the network packets that should be captured by the tool using specific criteria a packet must meet. They help you narrow the scope of the network traffic captured to only what is relevant to you and reduce Wireshark's processing overhead.

However, you usually want to capture all the network traffic on an interface and use Wireshark's **display filters** to filter a saved packet capture. This means you won't miss any relevant network traffic that the capture filter may have excluded.

Understanding the Interface

Once you have captured network traffic or imported a saved packet capture file, you will be greeted by Wireshark's default interface. This interface is compromised of four components.

The menu bar

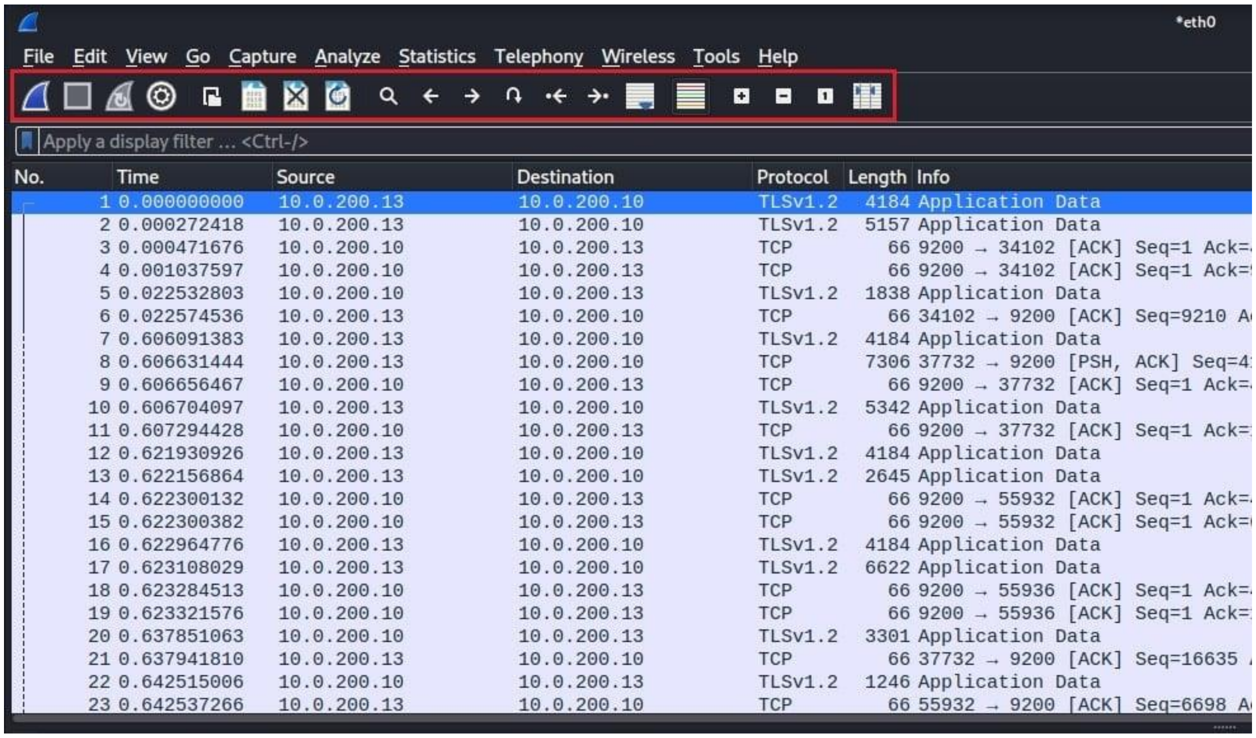

*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time          Source         Destination    Protocol Length Info
    1 0.000000000   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
    2 0.000272418   10.0.200.13    10.0.200.10    TLSv1.2   5157 Application Data
    3 0.000471676   10.0.200.10    10.0.200.13    TCP         66 9200 → 34102 [ACK] Seq=1 Ack=
    4 0.001037597   10.0.200.10    10.0.200.13    TCP         66 9200 → 34102 [ACK] Seq=1 Ack=
    5 0.022532803   10.0.200.10    10.0.200.13    TLSv1.2   1838 Application Data
    6 0.022574536   10.0.200.13    10.0.200.10    TCP         66 34102 → 9200 [ACK] Seq=9210 A
    7 0.606091383   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
    8 0.606631444   10.0.200.13    10.0.200.10    TCP       7306 37732 → 9200 [PSH, ACK] Seq=4
    9 0.606656467   10.0.200.10    10.0.200.13    TCP         66 9200 → 37732 [ACK] Seq=1 Ack=
   10 0.606704097   10.0.200.13    10.0.200.10    TLSv1.2   5342 Application Data
   11 0.607294428   10.0.200.10    10.0.200.13    TCP         66 9200 → 37732 [ACK] Seq=1 Ack=
   12 0.621930926   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
   13 0.622156864   10.0.200.13    10.0.200.10    TLSv1.2   2645 Application Data
   14 0.622300132   10.0.200.10    10.0.200.13    TCP         66 9200 → 55932 [ACK] Seq=1 Ack=
   15 0.622300382   10.0.200.10    10.0.200.13    TCP         66 9200 → 55932 [ACK] Seq=1 Ack=
   16 0.622964776   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
   17 0.623108029   10.0.200.13    10.0.200.10    TLSv1.2   6622 Application Data
   18 0.623284513   10.0.200.10    10.0.200.13    TCP         66 9200 → 55936 [ACK] Seq=1 Ack=
   19 0.623321576   10.0.200.10    10.0.200.13    TCP         66 9200 → 55936 [ACK] Seq=1 Ack=
   20 0.637851063   10.0.200.10    10.0.200.13    TLSv1.2   3301 Application Data
   21 0.637941810   10.0.200.13    10.0.200.10    TCP         66 37732 → 9200 [ACK] Seq=16635
   22 0.642515006   10.0.200.10    10.0.200.13    TLSv1.2   1246 Application Data
   23 0.642537266   10.0.200.13    10.0.200.10    TCP         66 55932 → 9200 [ACK] Seq=6698 A
```

Wireshark's menu bar is located at the top of its interface. It consists of several menus you can use to take advantage of Wireshark's powerful features:

- **File**: To can open and save files, as well as export packets, TLS session keys, and objects.
- **Edit**: To modify captured packets.
- **View**: To customize Wireshark's interface.
- **Go**: For packet navigation.
- **Capture**: To control starting, stopping, and configuring packet captures.
- **Analyze**: Wireshark's tools for analyzing network packets.
- **Statistics**: For applying statistical analysis to network traffic.
- **Telephony**: For analyzing Voice over IP (VoIP) traffic.
- **Wireless**: For analyzing Bluetooth traffic.
- **Tools**: Additional custom tools written in Lua.
- **Help**: Links to Wireshark's documentation.

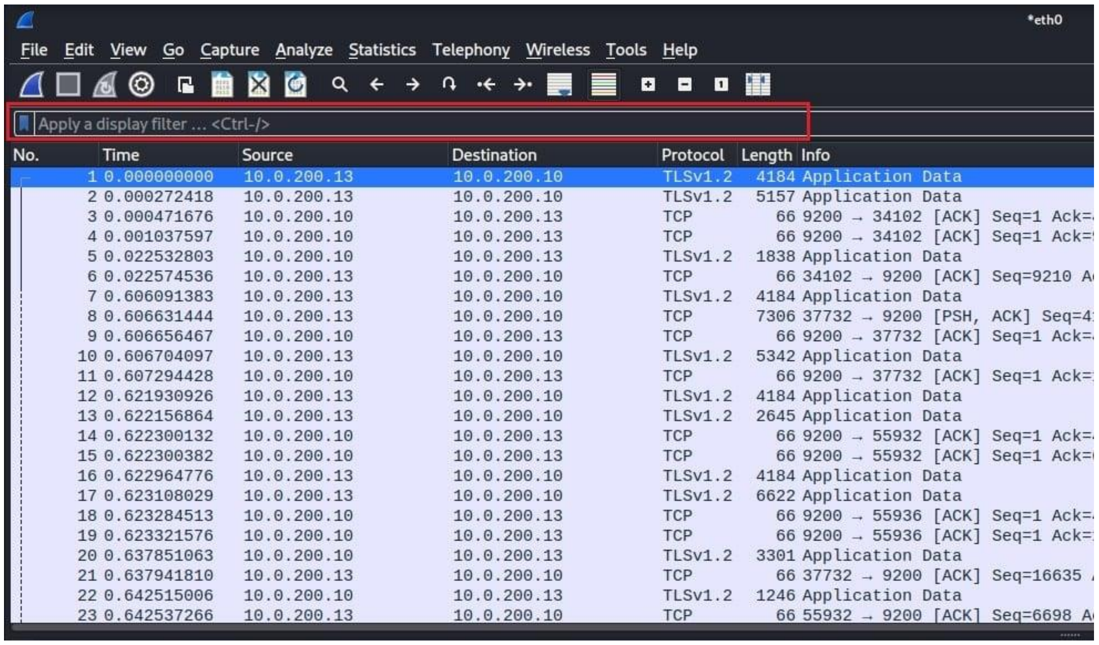
The main toolbar


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time          Source         Destination    Protocol  Length Info
    1 0.000000000   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
    2 0.000272418   10.0.200.13    10.0.200.10    TLSv1.2   5157 Application Data
    3 0.000471676   10.0.200.10    10.0.200.13    TCP         66 9200 → 34102 [ACK] Seq=1 Ack=
    4 0.001037597   10.0.200.10    10.0.200.13    TCP         66 9200 → 34102 [ACK] Seq=1 Ack=
    5 0.022532803   10.0.200.10    10.0.200.13    TLSv1.2   1838 Application Data
    6 0.022574536   10.0.200.13    10.0.200.10    TCP         66 34102 → 9200 [ACK] Seq=9210 A
    7 0.606091383   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
    8 0.606631444   10.0.200.13    10.0.200.10    TCP       7306 37732 → 9200 [PSH, ACK] Seq=4
    9 0.606656467   10.0.200.10    10.0.200.13    TCP         66 9200 → 37732 [ACK] Seq=1 Ack=
   10 0.606704097   10.0.200.13    10.0.200.10    TLSv1.2   5342 Application Data
   11 0.607294428   10.0.200.10    10.0.200.13    TCP         66 9200 → 37732 [ACK] Seq=1 Ack=
   12 0.621930926   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
   13 0.622156864   10.0.200.13    10.0.200.10    TLSv1.2   2645 Application Data
   14 0.622300132   10.0.200.10    10.0.200.13    TCP         66 9200 → 55932 [ACK] Seq=1 Ack=
   15 0.622300382   10.0.200.10    10.0.200.13    TCP         66 9200 → 55932 [ACK] Seq=1 Ack=
   16 0.622964776   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
   17 0.623108029   10.0.200.13    10.0.200.10    TLSv1.2   6622 Application Data
   18 0.623284513   10.0.200.10    10.0.200.13    TCP         66 9200 → 55936 [ACK] Seq=1 Ack=
   19 0.623321576   10.0.200.10    10.0.200.13    TCP         66 9200 → 55936 [ACK] Seq=1 Ack=
   20 0.637851063   10.0.200.10    10.0.200.13    TLSv1.2   3301 Application Data
   21 0.637941810   10.0.200.13    10.0.200.10    TCP         66 37732 → 9200 [ACK] Seq=16635
   22 0.642515006   10.0.200.10    10.0.200.13    TLSv1.2   1246 Application Data
   23 0.642537266   10.0.200.13    10.0.200.10    TCP         66 55932 → 9200 [ACK] Seq=6698 A
```

Below the menu bar is Wireshark's main toolbar. These shortcut buttons let you start/stop your packet capture, adjust your capture options, open/close capture files, search for packets, and resize interface panes.

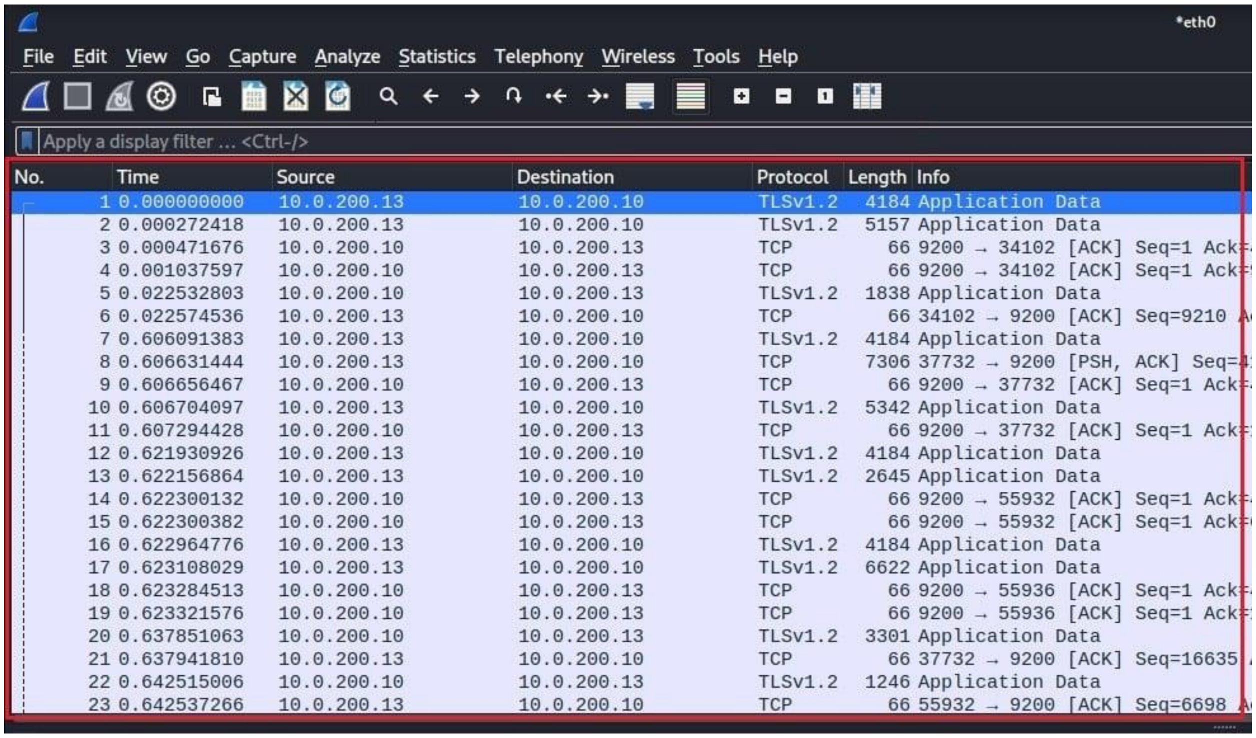
The display filter bar


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time          Source         Destination    Protocol  Length Info
    1 0.000000000   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
    2 0.000272418   10.0.200.13    10.0.200.10    TLSv1.2   5157 Application Data
    3 0.000471676   10.0.200.10    10.0.200.13    TCP         66 9200 → 34102 [ACK] Seq=1 Ack=
    4 0.001037597   10.0.200.10    10.0.200.13    TCP         66 9200 → 34102 [ACK] Seq=1 Ack=
    5 0.022532803   10.0.200.10    10.0.200.13    TLSv1.2   1838 Application Data
    6 0.022574536   10.0.200.13    10.0.200.10    TCP         66 34102 → 9200 [ACK] Seq=9210 A
    7 0.606091383   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
    8 0.606631444   10.0.200.13    10.0.200.10    TCP       7306 37732 → 9200 [PSH, ACK] Seq=4
    9 0.606656467   10.0.200.10    10.0.200.13    TCP         66 9200 → 37732 [ACK] Seq=1 Ack=
   10 0.606704097   10.0.200.13    10.0.200.10    TLSv1.2   5342 Application Data
   11 0.607294428   10.0.200.10    10.0.200.13    TCP         66 9200 → 37732 [ACK] Seq=1 Ack=
   12 0.621930926   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
   13 0.622156864   10.0.200.13    10.0.200.10    TLSv1.2   2645 Application Data
   14 0.622300132   10.0.200.10    10.0.200.13    TCP         66 9200 → 55932 [ACK] Seq=1 Ack=
   15 0.622300382   10.0.200.10    10.0.200.13    TCP         66 9200 → 55932 [ACK] Seq=1 Ack=
   16 0.622964776   10.0.200.13    10.0.200.10    TLSv1.2   4184 Application Data
   17 0.623108029   10.0.200.13    10.0.200.10    TLSv1.2   6622 Application Data
   18 0.623284513   10.0.200.10    10.0.200.13    TCP         66 9200 → 55936 [ACK] Seq=1 Ack=
   19 0.623321576   10.0.200.10    10.0.200.13    TCP         66 9200 → 55936 [ACK] Seq=1 Ack=
   20 0.637851063   10.0.200.10    10.0.200.13    TLSv1.2   3301 Application Data
   21 0.637941810   10.0.200.13    10.0.200.10    TCP         66 37732 → 9200 [ACK] Seq=16635
   22 0.642515006   10.0.200.10    10.0.200.13    TLSv1.2   1246 Application Data
   23 0.642537266   10.0.200.13    10.0.200.10    TCP         66 55932 → 9200 [ACK] Seq=6698 A
```

Next down is Wireshark's display filter bar. This allows you to filter the network traffic you have captured based on protocol, source/destination IP address, port number, MAC address, packet type (e.g., ICMP, ARP, etc.), packet length, time range, and packet content. You will see this in action later.

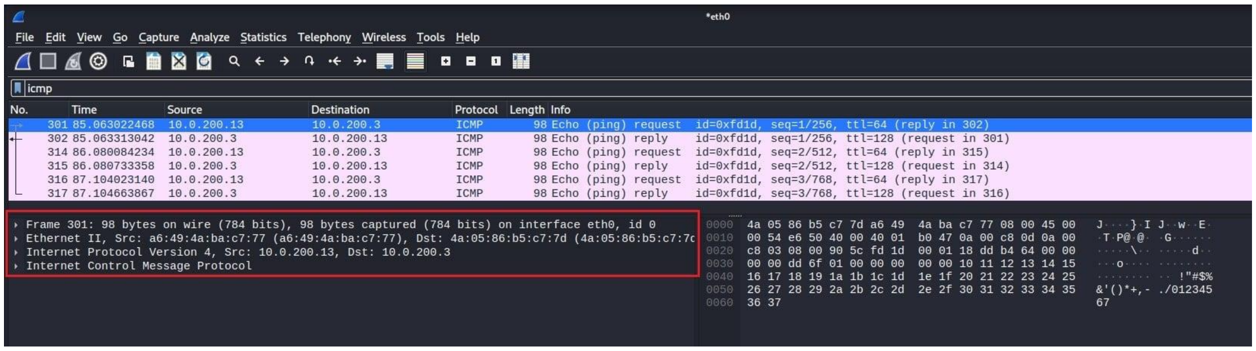
The packet list pane


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time         Source        Destination   Protocol Length Info
    1 0.000000000  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
    2 0.000272418  10.0.200.13   10.0.200.10   TLSv1.2  5157 Application Data
    3 0.000471676  10.0.200.10   10.0.200.13   TCP        66 9200 → 34102 [ACK] Seq=1 Ack
    4 0.001037597  10.0.200.10   10.0.200.13   TCP        66 9200 → 34102 [ACK] Seq=1 Ack
    5 0.022532803  10.0.200.10   10.0.200.13   TLSv1.2  1838 Application Data
    6 0.022574536  10.0.200.13   10.0.200.10   TCP        66 34102 → 9200 [ACK] Seq=9210 A
    7 0.606091383  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
    8 0.606631444  10.0.200.13   10.0.200.10   TCP      7306 37732 → 9200 [PSH, ACK] Seq=
    9 0.606656467  10.0.200.10   10.0.200.13   TCP        66 9200 → 37732 [ACK] Seq=1 Ack
   10 0.606704097  10.0.200.13   10.0.200.10   TLSv1.2  5342 Application Data
   11 0.607294428  10.0.200.10   10.0.200.13   TCP        66 9200 → 37732 [ACK] Seq=1 Ack
   12 0.621930926  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
   13 0.622156864  10.0.200.13   10.0.200.10   TLSv1.2  2645 Application Data
   14 0.622300132  10.0.200.10   10.0.200.13   TCP        66 9200 → 55932 [ACK] Seq=1 Ack
   15 0.622300382  10.0.200.10   10.0.200.13   TCP        66 9200 → 55932 [ACK] Seq=1 Ack
   16 0.622964776  10.0.200.13   10.0.200.10   TLSv1.2  4184 Application Data
   17 0.623108029  10.0.200.13   10.0.200.10   TLSv1.2  6622 Application Data
   18 0.623284513  10.0.200.10   10.0.200.13   TCP        66 9200 → 55936 [ACK] Seq=1 Ack
   19 0.623321576  10.0.200.10   10.0.200.13   TCP        66 9200 → 55936 [ACK] Seq=1 Ack
   20 0.637851063  10.0.200.10   10.0.200.13   TLSv1.2  3301 Application Data
   21 0.637941810  10.0.200.13   10.0.200.10   TCP        66 37732 → 9200 [ACK] Seq=16635
   22 0.642515006  10.0.200.10   10.0.200.13   TLSv1.2  1246 Application Data
   23 0.642537266  10.0.200.13   10.0.200.10   TCP        66 55932 → 9200 [ACK] Seq=6698 A
```

This is the main area of Wireshark's interface. It summarizes the key details of each packet captured, which you can scroll through or sort using the various columns. You can also right-click on an individual packet to perform various actions, such as marking a packet, applying filters, or following a packet stream.

The packet details pane


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

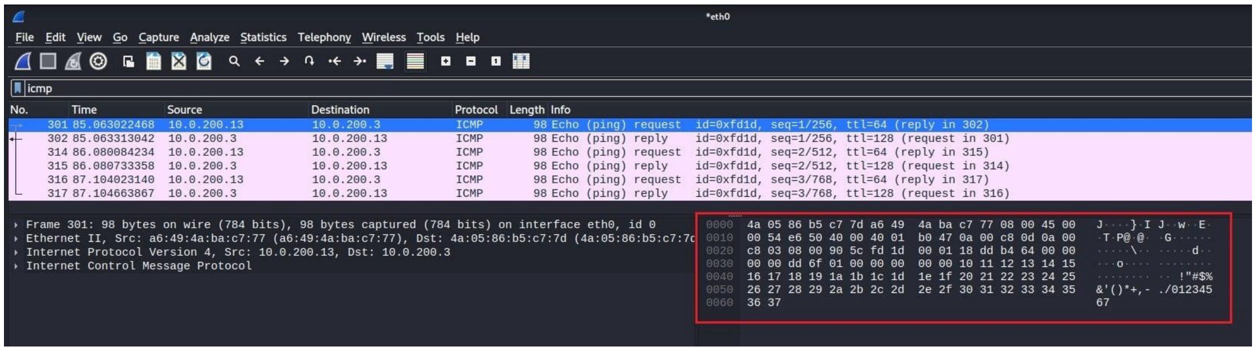
```
icmp
No.   Time          Source        Destination   Protocol Length Info
  301 85.063022468  10.0.200.13   10.0.200.3    ICMP       98 Echo (ping) request  id=0xfd1d, seq=1/256, ttl=64 (reply in 302)
  302 85.063313042  10.0.200.3    10.0.200.13   ICMP       98 Echo (ping) reply    id=0xfd1d, seq=1/256, ttl=128 (request in 301)
  314 86.080084234  10.0.200.13   10.0.200.3    ICMP       98 Echo (ping) request  id=0xfd1d, seq=2/512, ttl=64 (reply in 315)
  315 86.080733358  10.0.200.3    10.0.200.13   ICMP       98 Echo (ping) reply    id=0xfd1d, seq=2/512, ttl=128 (request in 314)
  316 87.104023140  10.0.200.13   10.0.200.3    ICMP       98 Echo (ping) request  id=0xfd1d, seq=3/768, ttl=64 (reply in 317)
  317 87.104663867  10.0.200.3    10.0.200.13   ICMP       98 Echo (ping) reply    id=0xfd1d, seq=3/768, ttl=128 (request in 316)

› Frame 301: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
› Ethernet II, Src: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77), Dst: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7c
› Internet Protocol Version 4, Src: 10.0.200.13, Dst: 10.0.200.3
› Internet Control Message Protocol

0000  4a 05 86 b5 c7 7d a6 49  4a ba c7 77 08 00 45 00   J····}·I J··w··E·
0010  00 54 e6 50 40 00 40 01  b0 47 0a 00 c8 0d 0a 00   ·T·P@·@· ·G······
0020  c8 03 08 00 90 5c fd 1d  00 01 18 dd b4 64 00 00   ·····\·· ·····d··
0030  00 00 dd 6f 01 00 00 00  00 00 10 11 12 13 14 15   ···o···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

Once you select a packet, the packet detail pane will populate. This includes the specific network packet details broken down based on TCP/IP network layer. You will see how to inspect individual packets later.

The packet bytes pane


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp

```
No.   Time          Source         Destination    Protocol Length Info
  301 85.063022468  10.0.200.13    10.0.200.3     ICMP       98 Echo (ping) request  id=0xfd1d, seq=1/256, ttl=64 (reply in 302)
  302 85.063313042  10.0.200.3     10.0.200.13    ICMP       98 Echo (ping) reply    id=0xfd1d, seq=1/256, ttl=128 (request in 301)
  314 86.080084234  10.0.200.13    10.0.200.3     ICMP       98 Echo (ping) request  id=0xfd1d, seq=2/512, ttl=64 (reply in 315)
  315 86.080733358  10.0.200.3     10.0.200.13    ICMP       98 Echo (ping) reply    id=0xfd1d, seq=2/512, ttl=128 (request in 314)
  316 87.104023140  10.0.200.13    10.0.200.3     ICMP       98 Echo (ping) request  id=0xfd1d, seq=3/768, ttl=64 (reply in 317)
  317 87.104663867  10.0.200.3     10.0.200.13    ICMP       98 Echo (ping) reply    id=0xfd1d, seq=3/768, ttl=128 (request in 316)
```

```
› Frame 301: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
› Ethernet II, Src: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77), Dst: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7c
› Internet Protocol Version 4, Src: 10.0.200.13, Dst: 10.0.200.3
› Internet Control Message Protocol
```

```
0000  4a 05 86 b5 c7 7d a6 49  4a ba c7 77 08 00 45 00   J····}·I J··w··E·
0010  00 54 e6 50 40 00 40 01  b0 47 0a 00 c8 0d 0a 00   ·T·P@·@·  ·G······
0020  c8 03 08 00 90 5c fd 1d  00 01 18 dd b4 64 00 00   ·····\··  ·····d··
0030  00 00 dd 6f 01 00 00 00  00 00 10 11 12 13 14 15   ···o····  ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········  ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,-  ./012345
0060  36 37                                              67
```

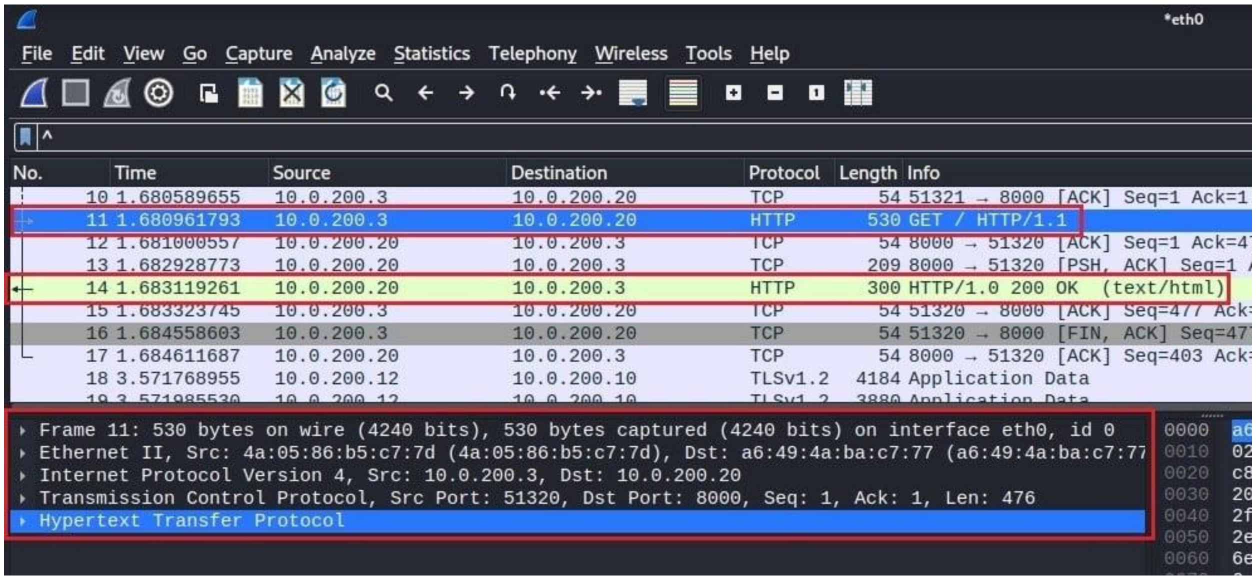
To the right of the packet details pane is the packet bytes pane. This shows the corresponding bytes Wireshark extracted from the wire to reconstruct the network packet data. These bytes are shown in hexadecimal format and are highlighted when you select information within the packet details pane. This pane is often used for troubleshooting when Wireshark cannot parse network data.

Analyzing Captured Traffic

You can take several actions in Wireshark to analyze the network traffic you have captured. The first of these is examining the individual packets captured.

Examining Packets

To examine a packet, select it within the packet list pane. This will populate the packet details pane with that packet's network information.


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
No.   Time          Source         Destination    Protocol Length Info
   10 1.680589655   10.0.200.3     10.0.200.20    TCP        54 51321 → 8000 [ACK] Seq=1 Ack=1
   11 1.680961793   10.0.200.3     10.0.200.20    HTTP      530 GET / HTTP/1.1
   12 1.681000557   10.0.200.20    10.0.200.3     TCP        54 8000 → 51320 [ACK] Seq=1 Ack=4
   13 1.682928773   10.0.200.20    10.0.200.3     TCP       209 8000 → 51320 [PSH, ACK] Seq=1 A
   14 1.683119261   10.0.200.20    10.0.200.3     HTTP      300 HTTP/1.0 200 OK  (text/html)
   15 1.683323745   10.0.200.3     10.0.200.20    TCP        54 51320 → 8000 [ACK] Seq=477 Ack=
   16 1.684558603   10.0.200.3     10.0.200.20    TCP        54 51320 → 8000 [FIN, ACK] Seq=477
   17 1.684611687   10.0.200.20    10.0.200.3     TCP        54 8000 → 51320 [ACK] Seq=403 Ack=
   18 3.571768955   10.0.200.12    10.0.200.10    TLSv1.2  4184 Application Data
   19 3.571985530   10.0.200.12    10.0.200.10    TLSv1.2  3880 Application Data
```

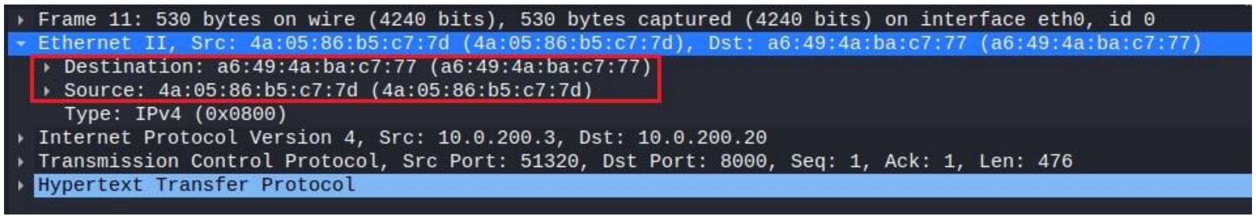
```
› Frame 11: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
› Ethernet II, Src: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7d), Dst: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77
› Internet Protocol Version 4, Src: 10.0.200.3, Dst: 10.0.200.20
› Transmission Control Protocol, Src Port: 51320, Dst Port: 8000, Seq: 1, Ack: 1, Len: 476
› Hypertext Transfer Protocol
```

```
0000  a6
0010  02
0020  c8
0030  20
0040  2f
0050  2e
0060  6e
```

Here packet number 11 has been selected. You see a right-facing arrow indicating a request in Wireshark's packet list pane and a left-facing arrow indicating a response several packets down (packet 14). This is a classic HTTP request and response network conversation.

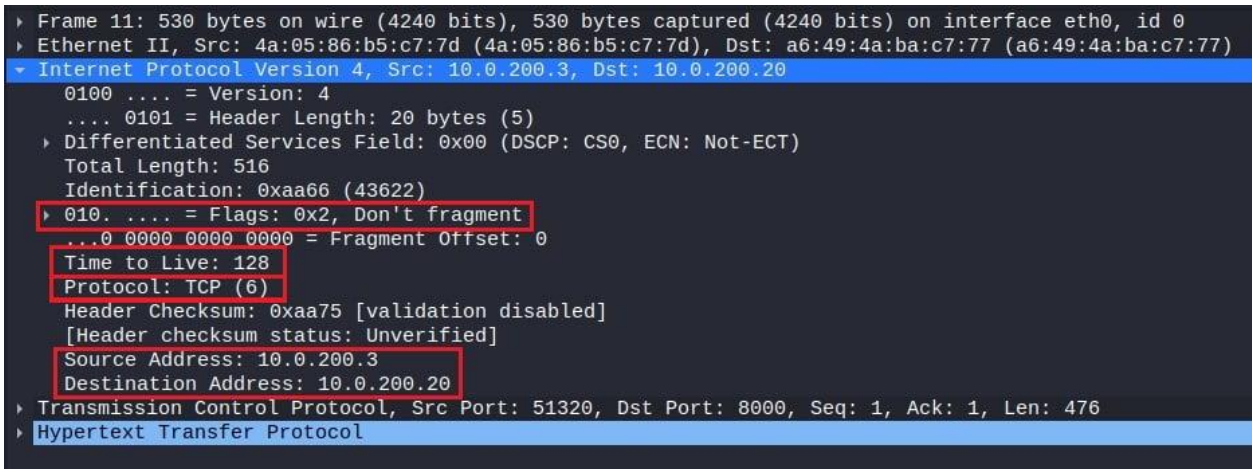
You will also see the packet details pane populated with network information about packet number 11, including information at the link, Internet, transport, and TCP/IP stack application layer.

To start, you can analyze the network information at the **link layer**. This includes the source and destination MAC addresses of the two communicating devices and the type of Internet protocol used (IPv4 or IPv6).



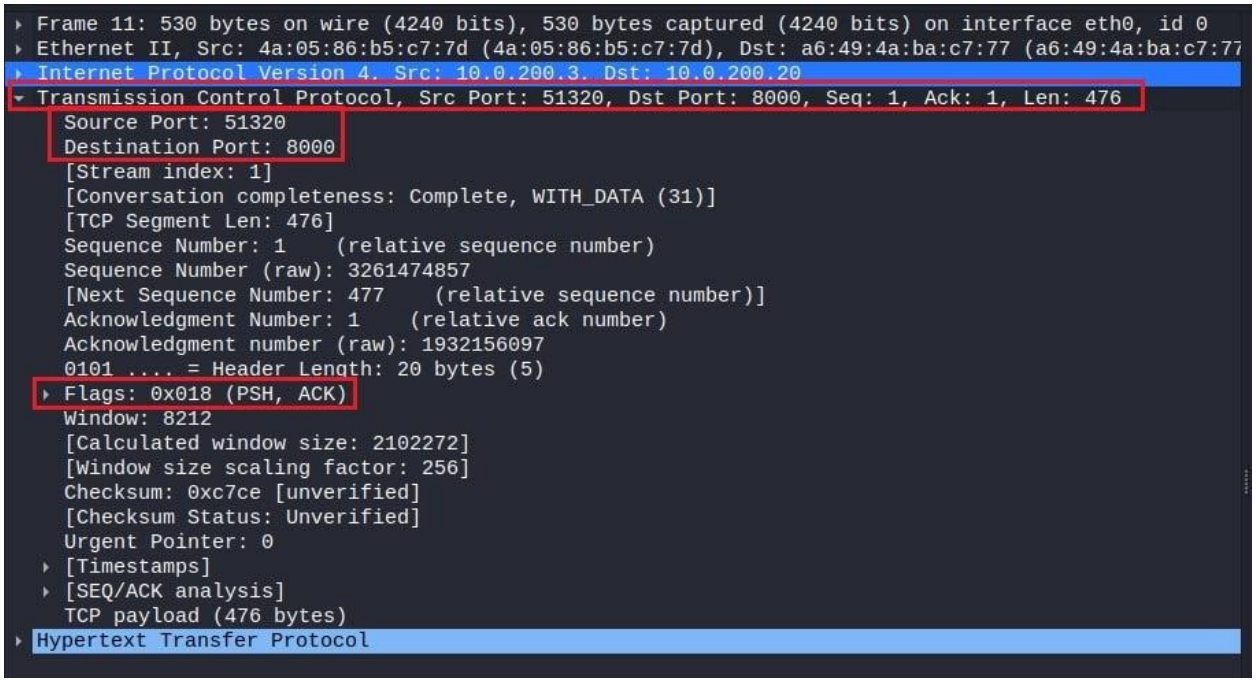
```
▸ Frame 11: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
▾ Ethernet II, Src: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7d), Dst: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77)
  ▸ Destination: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77)
  ▸ Source: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7d)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.200.3, Dst: 10.0.200.20
▸ Transmission Control Protocol, Src Port: 51320, Dst Port: 8000, Seq: 1, Ack: 1, Len: 476
▸ Hypertext Transfer Protocol
```

You can then move on to the information included within the **Internet layer**. This example contains information specific to the IPv4 protocol, such as IP flags, the source and destination IP address, Time to Live (TTL), the transport protocol encapsulated within this packet (e.g., TCP), and other header information.



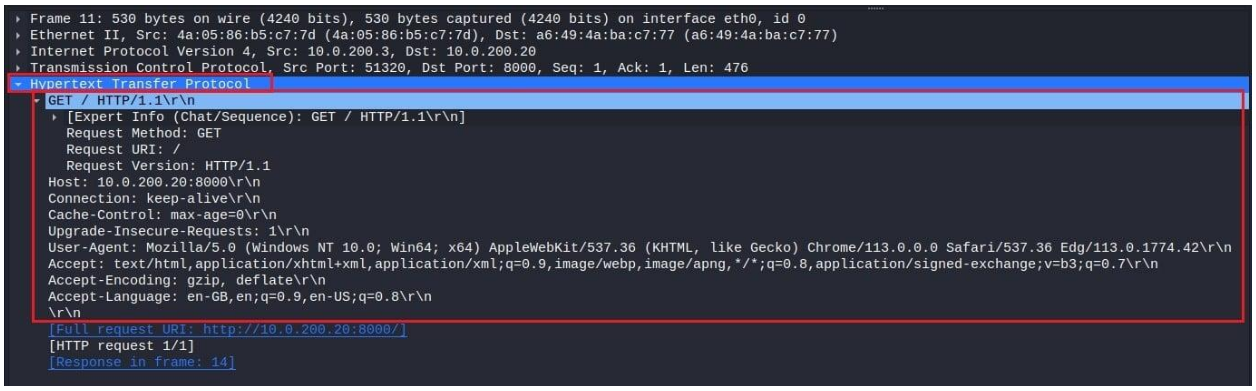
```
▸ Frame 11: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
▸ Ethernet II, Src: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7d), Dst: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77)
▾ Internet Protocol Version 4, Src: 10.0.200.3, Dst: 10.0.200.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 516
    Identification: 0xaa66 (43622)
  ▸ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0xaa75 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.200.3
    Destination Address: 10.0.200.20
▸ Transmission Control Protocol, Src Port: 51320, Dst Port: 8000, Seq: 1, Ack: 1, Len: 476
▸ Hypertext Transfer Protocol
```

Following the IP layer is the **transport layer**. This network protocol is responsible for the end-to-end data delivery between hosts and will be either UDP or TCP. It includes the source and destination port of the segment, TCP-specific header flags, the size of the encapsulated application message, and other header information.



```
Frame 11: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
Ethernet II, Src: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7d), Dst: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77
Internet Protocol Version 4, Src: 10.0.200.3, Dst: 10.0.200.20
Transmission Control Protocol, Src Port: 51320, Dst Port: 8000, Seq: 1, Ack: 1, Len: 476
  Source Port: 51320
  Destination Port: 8000
  [Stream index: 1]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 476]
  Sequence Number: 1    (relative sequence number)
  Sequence Number (raw): 3261474857
  [Next Sequence Number: 477    (relative sequence number)]
  Acknowledgment Number: 1    (relative ack number)
  Acknowledgment number (raw): 1932156097
  0101 .... = Header Length: 20 bytes (5)
  Flags: 0x018 (PSH, ACK)
  Window: 8212
  [Calculated window size: 2102272]
  [Window size scaling factor: 256]
  Checksum: 0xc7ce [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
  [SEQ/ACK analysis]
  TCP payload (476 bytes)
Hypertext Transfer Protocol
```

Finally, the packet details pane will show you network information about the packet's application protocol (if an application protocol is used). In this case, HTTP is the application protocol, so you can see information about the request method, request URI, and other common HTTP headers. This will be where you find most of the network information you want to analyze.

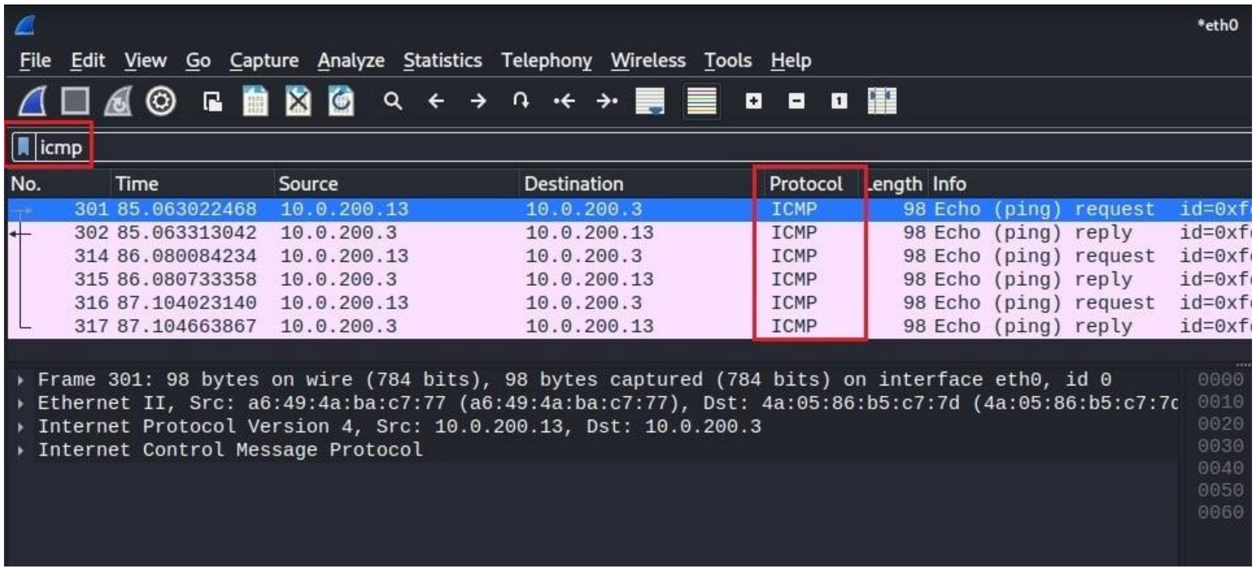


```
Frame 11: 530 bytes on wire (4240 bits), 530 bytes captured (4240 bits) on interface eth0, id 0
Ethernet II, Src: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7d), Dst: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77)
Internet Protocol Version 4, Src: 10.0.200.3, Dst: 10.0.200.20
Transmission Control Protocol, Src Port: 51320, Dst Port: 8000, Seq: 1, Ack: 1, Len: 476
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
  Host: 10.0.200.20:8000\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/113.0.1774.42\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-GB,en;q=0.9,en-US;q=0.8\r\n
  \r\n
  [Full request URI: http://10.0.200.20:8000/]
  [HTTP request 1/1]
  [Response in frame: 14]
```

Refining your view with display filters

You must sift through thousands of network packets when analyzing traffic. To make this task more efficient, Wireshark has built-in display filters that you can use to narrow down the packets displayed in its packet list pane.

To use these filters, enter them into Wireshark's display filter bar (below the main toolbar).

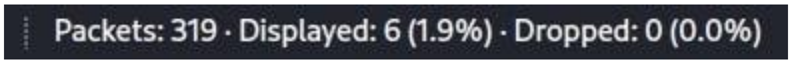


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp

```
No.  Time          Source        Destination   Protocol Length Info
301  85.063022468  10.0.200.13   10.0.200.3    ICMP     98 Echo (ping) request  id=0xf
302  85.063313042  10.0.200.3    10.0.200.13   ICMP     98 Echo (ping) reply    id=0xf
314  86.080084234  10.0.200.13   10.0.200.3    ICMP     98 Echo (ping) request  id=0xf
315  86.080733358  10.0.200.3    10.0.200.13   ICMP     98 Echo (ping) reply    id=0xf
316  87.104023140  10.0.200.13   10.0.200.3    ICMP     98 Echo (ping) request  id=0xf
317  87.104663867  10.0.200.3    10.0.200.13   ICMP     98 Echo (ping) reply    id=0xf

› Frame 301: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
› Ethernet II, Src: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77), Dst: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7c
› Internet Protocol Version 4, Src: 10.0.200.13, Dst: 10.0.200.3
› Internet Control Message Protocol
```

In this screenshot, the filter icmp has been used. This filter will only display packets that use the Internet Control Message Protocol (ICMP) as their network-layer protocol. Now you can analyze all relevant packets without searching through thousands of irrelevant ones. Wireshark will show you how many packets have been filtered out by your display filter at the bottom right of its interface.



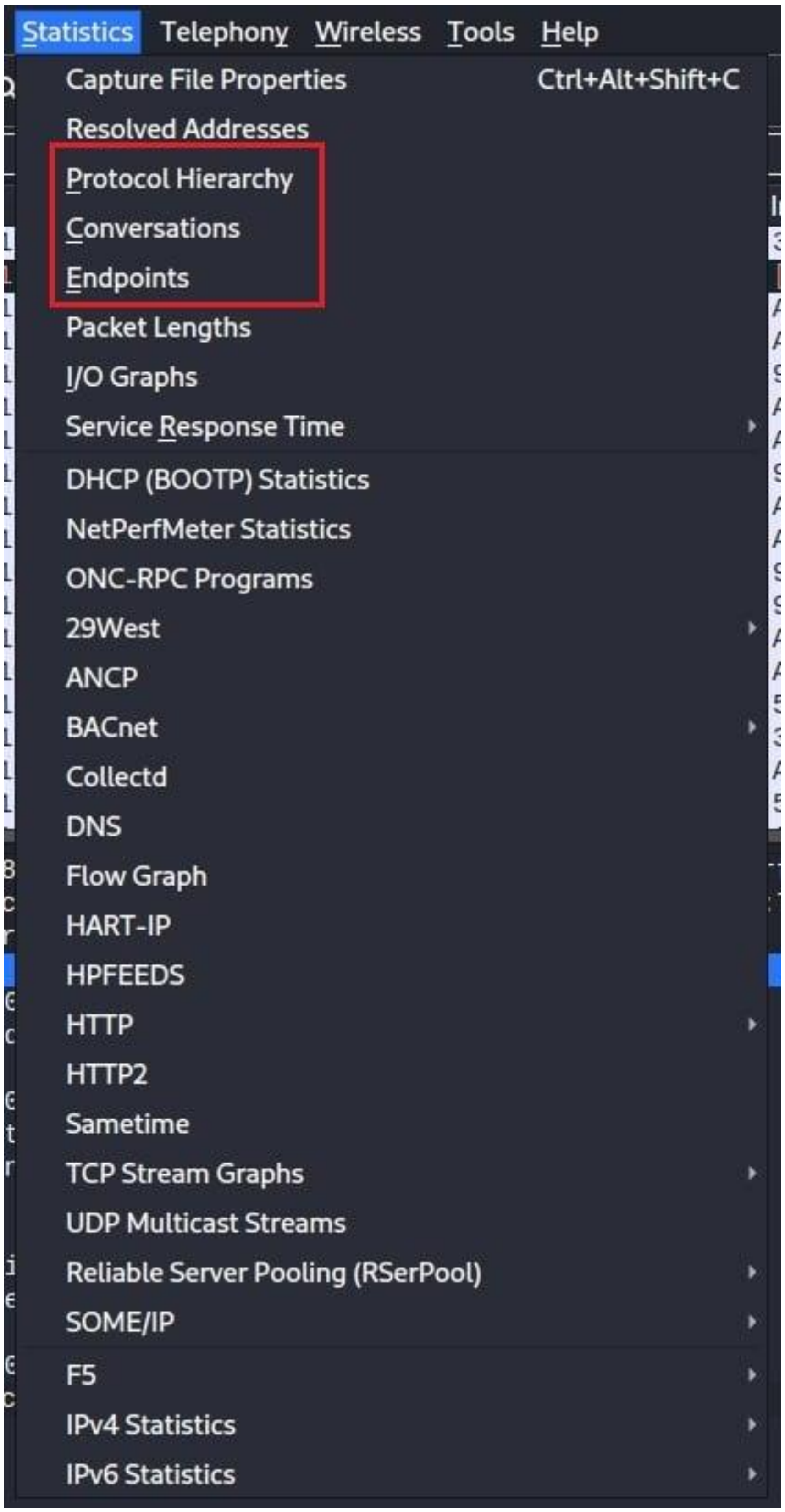
Packets: 319 · Displayed: 6 (1.9%) · Dropped: 0 (0.0%)

You should use display filters to refine your view and speed up your network analysisThis will show you how to filter by IP address, port number, protocol, and more!

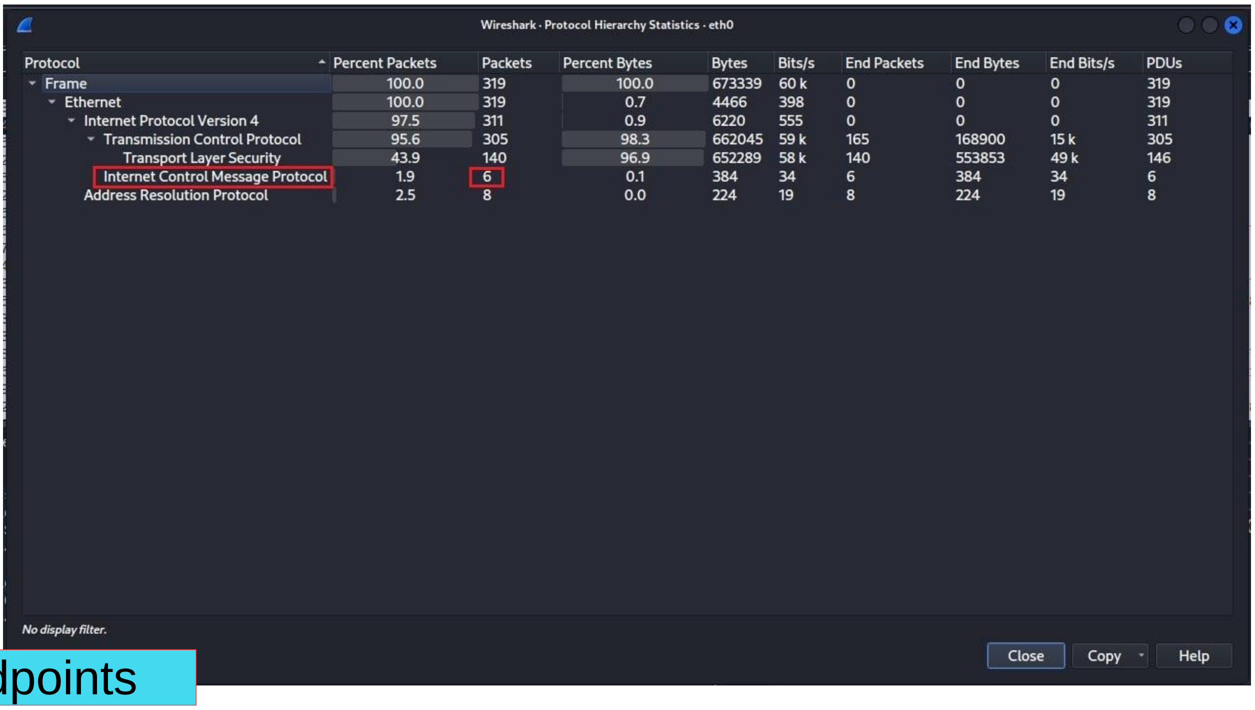
Performing statistical analysis

Aside from analyzing individual network packets, Wireshark also has a powerful statistical analysis feature that lets you quickly summarise your network traffic.

Select the Statistic menu from Wireshark's main menu bar. This will provide options for showing summary information about the protocols being used, the endpoints communicating, and the network conversations between those endpoints.


Statistics
Telephony
Wireless
Tools
Help
Capture File Properties
Ctrl+Alt+Shift+C
Resolved Addresses
Protocol Hierarchy
Conversations
Endpoints
Packet Lengths
I/O Graphs
Service Response Time
DHCP (BOOTP) Statistics
NetPerfMeter Statistics
ONC-RPC Programs
29West
ANCP
BACnet
Collectd
DNS
Flow Graph
HART-IP
HPFEEDS
HTTP
HTTP2
Sametime
TCP Stream Graphs
UDP Multicast Streams
Reliable Server Pooling (RSerPool)
SOME/IP
F5
IPv4 Statistics
IPv6 Statistics

Select the Protocol Hierarchy option to discover the protocols used within your captured network traffic. This will generate a window summarizing the network protocols used at each TCP/IP stack layer.



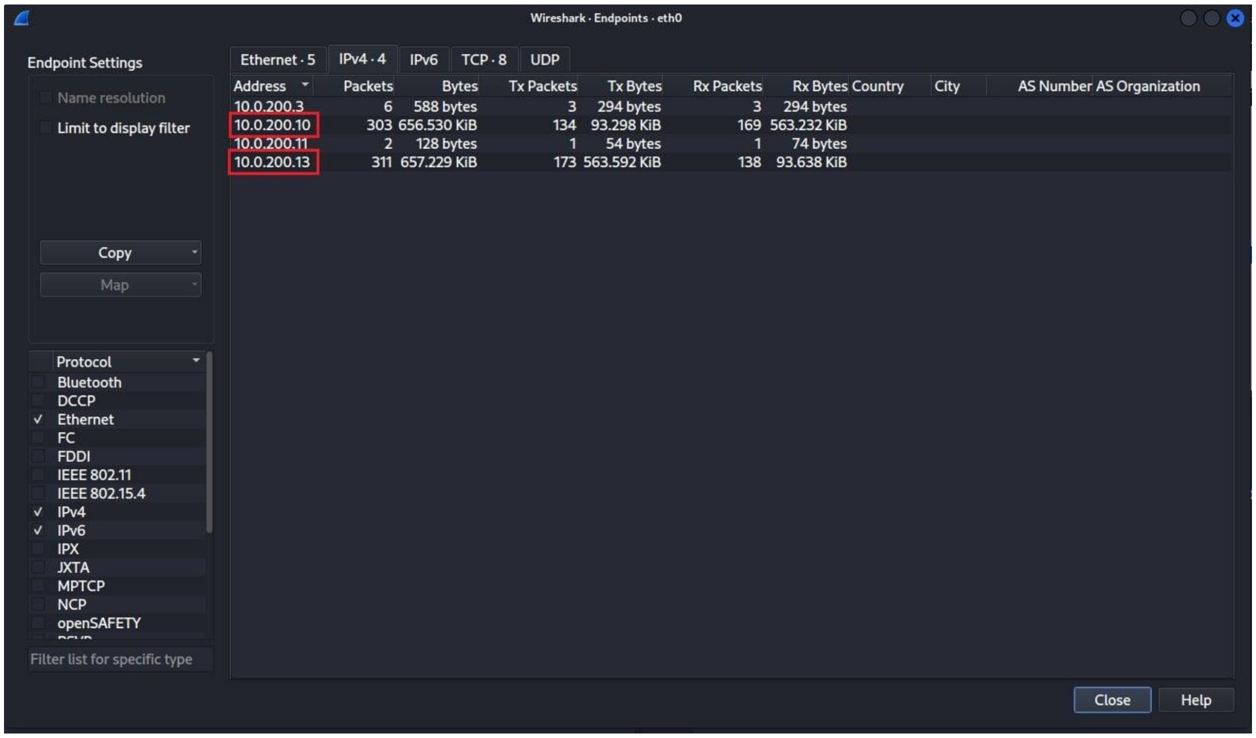
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	319	100.0	673339	60 k	0	0	0	319
Ethernet	100.0	319	0.7	4466	398	0	0	0	319
Internet Protocol Version 4	97.5	311	0.9	6220	555	0	0	0	311
Transmission Control Protocol	95.6	305	98.3	662045	59 k	165	168900	15 k	305
Transport Layer Security	43.9	140	96.9	652289	58 k	140	553853	49 k	146
Internet Control Message Protocol	1.9	6	0.1	384	34	6	384	34	6
Address Resolution Protocol	2.5	8	0.0	224	19	8	224	19	8

No display filter.

Endpoints

Select the Endpoints option to find out what endpoints are present in your network traffic. This will list all the unique endpoint devices communicating within the network packets you have captured based on Ethernet address, IP address, and TCP/UDP port.

In the example below, the IPv4 tab has been selected. This shows that four devices are communicating in this packet capture, with two being the primary exchangers of data.


Wireshark · Endpoints · eth0

Endpoint Settings

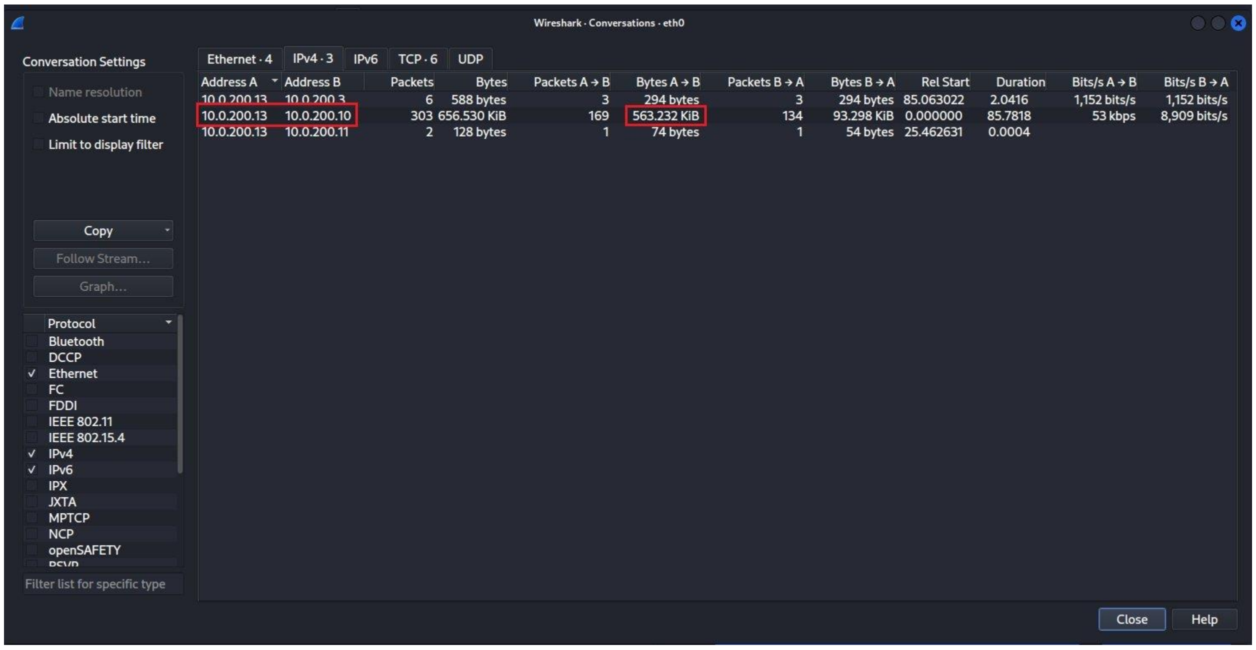
- Name resolution
- Limit to display filter

Ethernet · 5 | IPv4 · 4 | IPv6 | TCP · 8 | UDP

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	AS Number	AS Organization
10.0.200.3	6	588 bytes	3	294 bytes	3	294 bytes				
10.0.200.10	303	656.530 KiB	134	93.298 KiB	169	563.232 KiB				
10.0.200.11	2	128 bytes	1	54 bytes	1	74 bytes				
10.0.200.13	311	657.229 KiB	173	563.592 KiB	138	93.638 KiB				

Protocol: Bluetooth, DCCP, ✓ Ethernet, FC, FDDI, IEEE 802.11, IEEE 802.15.4, ✓ IPv4, ✓ IPv6, IPX, JXTA, MPTCP, NCP, openSAFETY

Select the Conversations option to delve into the network conversations within your packet capture. This will show you the network traffic traveling between endpoints. Here you can see a lot of data is traveling between 10.0.200.13 and 10.0.200.10, while very little is between 10.0.200.13 and 10.0.200.11.


Wireshark · Conversations · eth0

Conversation Settings

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

Ethernet · 4 | IPv4 · 3 | IPv6 | TCP · 6 | UDP

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
10.0.200.13	10.0.200.3	6	588 bytes	3	294 bytes	3	294 bytes	85.063022	2.0416	1,152 bits/s	1,152 bits/s
10.0.200.13	10.0.200.10	303	656.530 KiB	169	563.232 KiB	134	93.298 KiB	0.000000	85.7818	53 kbps	8,909 bits/s
10.0.200.13	10.0.200.11	2	128 bytes	1	74 bytes	1	54 bytes	25.462631	0.0004		

Protocol: Bluetooth, DCCP, ✓ Ethernet, FC, FDDI, IEEE 802.11, IEEE 802.15.4, ✓ IPv4, ✓ IPv6, IPX, JXTA, MPTCP, NCP, openSAFETY

You can use Wireshark’s statistical analysis feature to quickly identify outliers within your packet capture and refine your search using display filters to analyze these outliers.

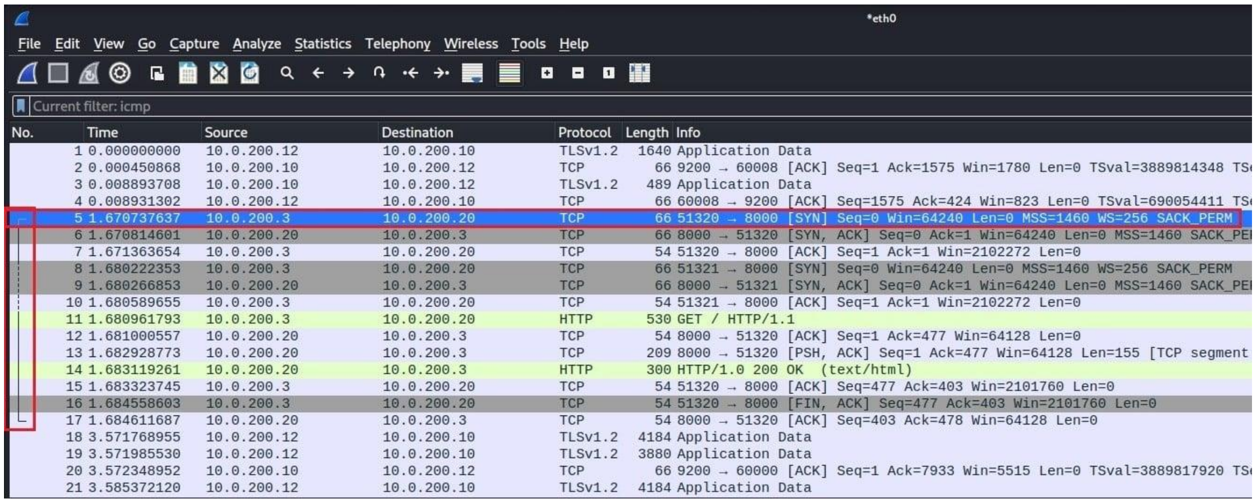
Techniques

Now that you have a basic understanding of capturing and analyzing traffic with Wireshark let's look at how you can use this powerful network protocol analyzer in the real world. The following are three use cases you will likely encounter whether you are on the red or blue team in cyber security.

Following TCP Streams

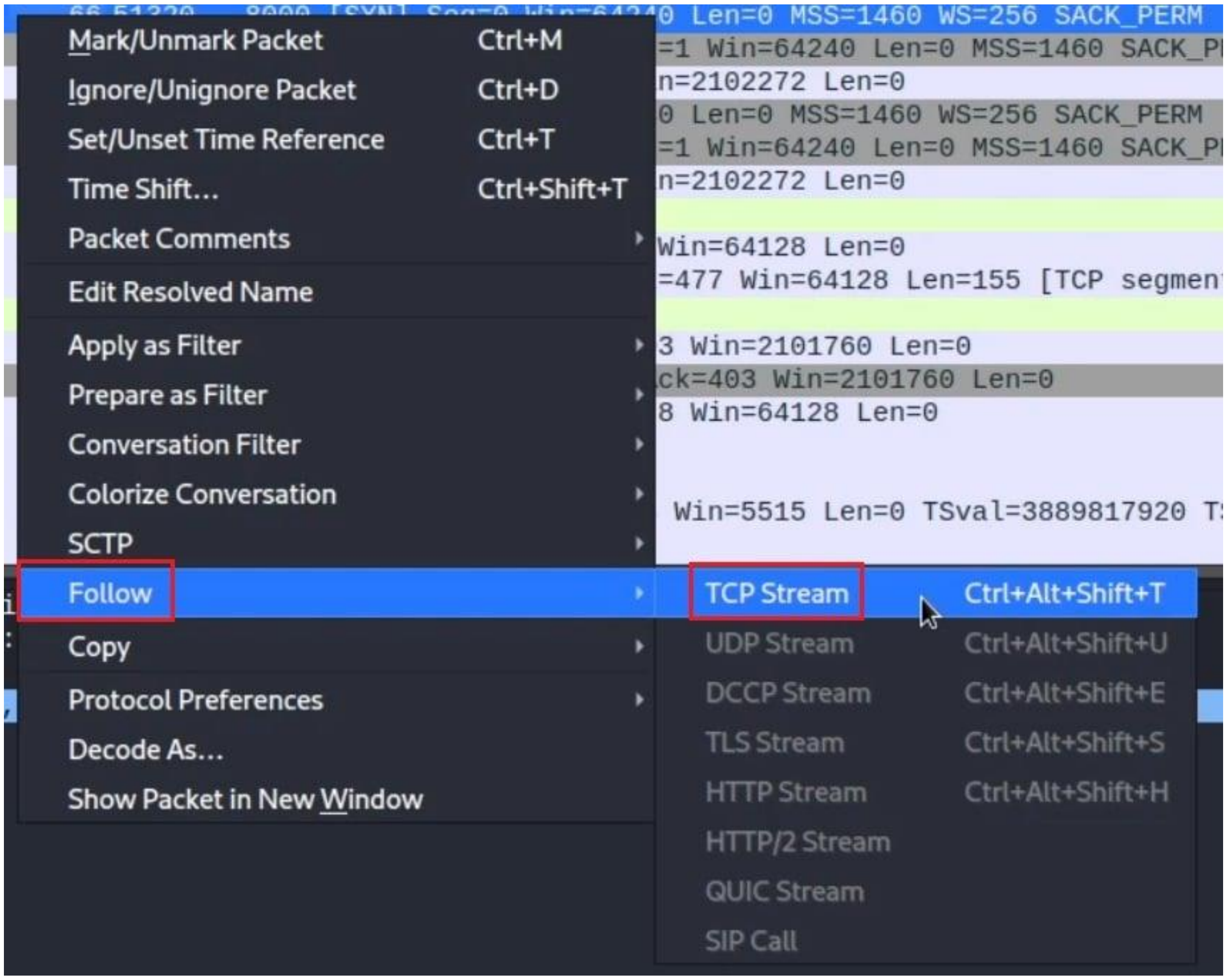
One of the most common use cases for Wireshark is following TCP streams. By following this stream, you can see the conversation between the two devices and the data they exchanged. This is useful for troubleshooting network issues or discovering hidden information.

To follow a TCP stream in Wireshark, right-click on a packet whose TCP stream you want to follow.


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Current filter: icmp

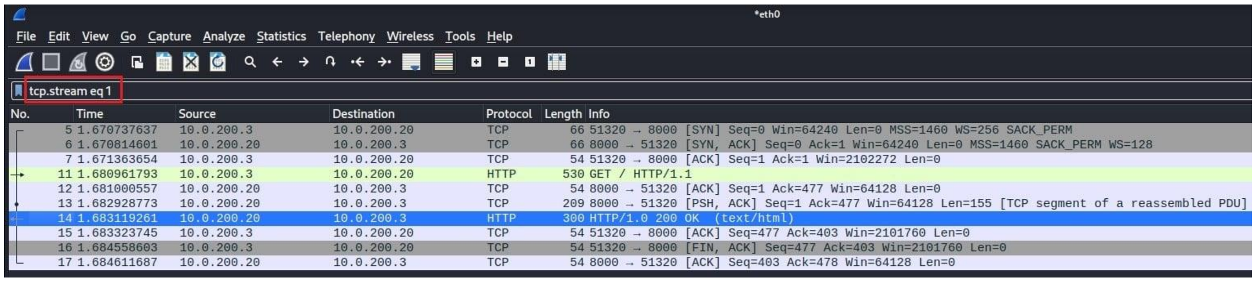
```
No.  Time         Source        Destination   Protocol Length Info
  1 0.000000000  10.0.200.12   10.0.200.10   TLSv1.2   1640 Application Data
  2 0.000450868  10.0.200.10   10.0.200.12   TCP         66 9200 → 60008 [ACK] Seq=1 Ack=1575 Win=1780 Len=0 TSval=3889814348 TS
  3 0.008893708  10.0.200.10   10.0.200.12   TLSv1.2    489 Application Data
  4 0.008931302  10.0.200.12   10.0.200.10   TCP         66 60008 → 9200 [ACK] Seq=1575 Ack=424 Win=823 Len=0 TSval=690054411 TS
  5 1.670737637  10.0.200.3    10.0.200.20   TCP         66 51320 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  6 1.670814601  10.0.200.20   10.0.200.3    TCP         66 8000 → 51320 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PE
  7 1.671363654  10.0.200.3    10.0.200.20   TCP         54 51320 → 8000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
  8 1.680222353  10.0.200.3    10.0.200.20   TCP         66 51321 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  9 1.680266853  10.0.200.20   10.0.200.3    TCP         66 8000 → 51321 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PE
 10 1.680589655  10.0.200.3    10.0.200.20   TCP         54 51321 → 8000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
 11 1.680961793  10.0.200.3    10.0.200.20   HTTP       530 GET / HTTP/1.1
 12 1.681000557  10.0.200.20   10.0.200.3    TCP         54 8000 → 51320 [ACK] Seq=1 Ack=477 Win=64128 Len=0
 13 1.682928773  10.0.200.20   10.0.200.3    TCP        209 8000 → 51320 [PSH, ACK] Seq=1 Ack=477 Win=64128 Len=155 [TCP segment
 14 1.683119261  10.0.200.20   10.0.200.3    HTTP       300 HTTP/1.0 200 OK  (text/html)
 15 1.683323745  10.0.200.3    10.0.200.20   TCP         54 51320 → 8000 [ACK] Seq=477 Ack=403 Win=2101760 Len=0
 16 1.684558603  10.0.200.3    10.0.200.20   TCP         54 51320 → 8000 [FIN, ACK] Seq=477 Ack=403 Win=2101760 Len=0
 17 1.684611687  10.0.200.20   10.0.200.3    TCP         54 8000 → 51320 [ACK] Seq=403 Ack=478 Win=64128 Len=0
 18 3.571768955  10.0.200.12   10.0.200.10   TLSv1.2   4184 Application Data
 19 3.571985530  10.0.200.12   10.0.200.10   TLSv1.2   3880 Application Data
 20 3.572348952  10.0.200.10   10.0.200.12   TCP         66 9200 → 60000 [ACK] Seq=1 Ack=7933 Win=5515 Len=0 TSval=3889817920 TS
 21 3.585372120  10.0.200.12   10.0.200.10   TLSv1.2   4184 Application Data
```

Here packet number 5 has been selected. It is a TCP connect request (TCP SYN flag) sent to port 8000 on machine 10.0.200.20 and is the start of the **TCP three-way handshake**. A drop-down arrow on the left of the packet list pane indicates the entire network conversation down to packet number 17. To follow this TCP stream, right-click on packet number 5 and select Follow > TCP Stream.



Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift… Ctrl+Shift+T
Packet Comments
Edit Resolved Name
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As…
Show Packet in New Window
TCP Stream Ctrl+Alt+Shift+T
UDP Stream Ctrl+Alt+Shift+U
DCCP Stream Ctrl+Alt+Shift+E
TLS Stream Ctrl+Alt+Shift+S
HTTP Stream Ctrl+Alt+Shift+H
HTTP/2 Stream
QUIC Stream
SIP Call

Wireshark will automatically apply a display filter that only shows packets from this TCP stream.

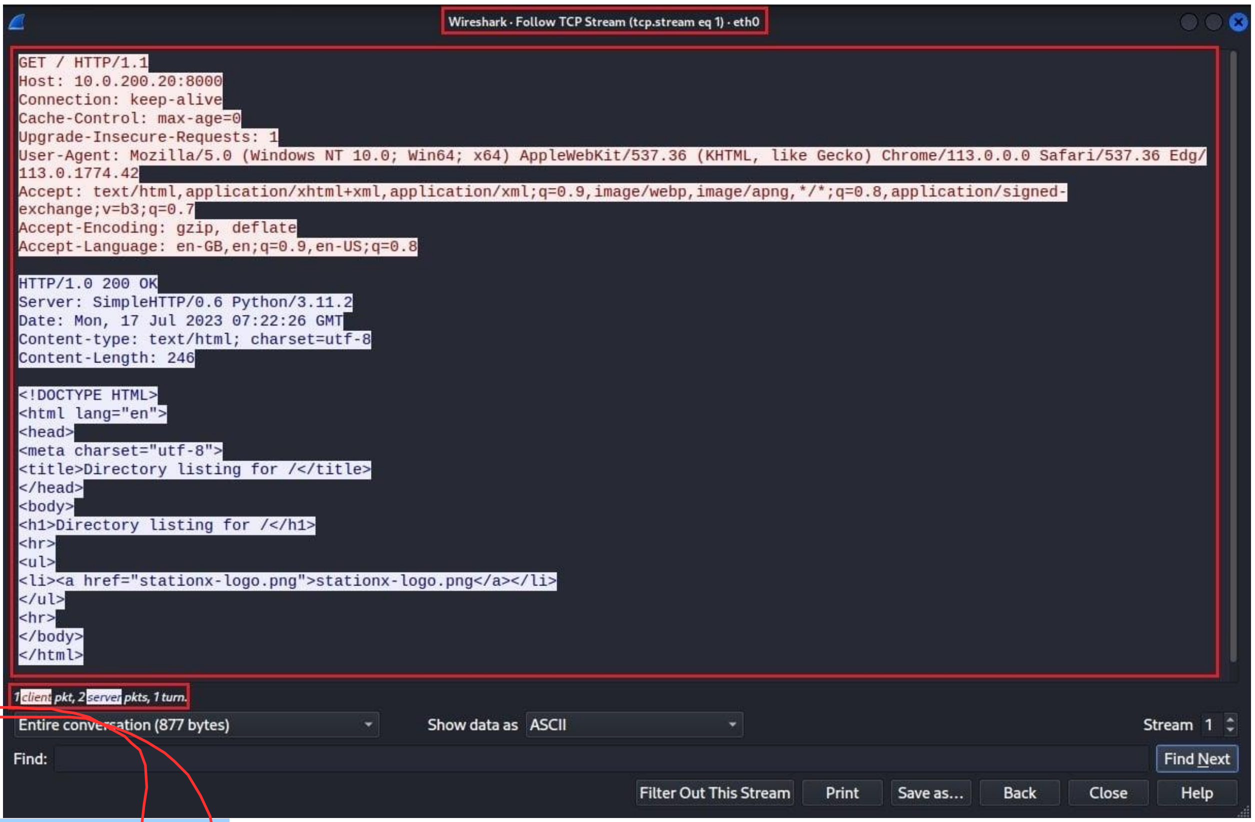


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.stream eq 1
```

```
No.  Time         Source       Destination  Protocol Length Info
  5 1.670737637  10.0.200.3   10.0.200.20  TCP        66 51320 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  6 1.670814601  10.0.200.20  10.0.200.3   TCP        66 8000 → 51320 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
  7 1.671363654  10.0.200.3   10.0.200.20  TCP        54 51320 → 8000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
 11 1.680961793  10.0.200.3   10.0.200.20  HTTP      530 GET / HTTP/1.1
 12 1.681000557  10.0.200.20  10.0.200.3   TCP        54 8000 → 51320 [ACK] Seq=1 Ack=477 Win=64128 Len=0
 13 1.682928773  10.0.200.20  10.0.200.3   TCP       209 8000 → 51320 [PSH, ACK] Seq=1 Ack=477 Win=64128 Len=155 [TCP segment of a reassembled PDU]
 14 1.683119261  10.0.200.20  10.0.200.3   HTTP      300 HTTP/1.0 200 OK  (text/html)
 15 1.683323745  10.0.200.3   10.0.200.20  TCP        54 51320 → 8000 [ACK] Seq=477 Ack=403 Win=2101760 Len=0
 16 1.684558603  10.0.200.3   10.0.200.20  TCP        54 51320 → 8000 [FIN, ACK] Seq=477 Ack=403 Win=2101760 Len=0
 17 1.684611687  10.0.200.20  10.0.200.3   TCP        54 8000 → 51320 [ACK] Seq=403 Ack=478 Win=64128 Len=0
```

Wireshark will also generate a summary view of the TCP stream in a new window. In this case, it highlights the HTTP messages sent between a client and server.



```
GET / HTTP/1.1
Host: 10.0.200.20:8000
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/
113.0.1774.42
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en;q=0.9,en-US;q=0.8

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.11.2
Date: Mon, 17 Jul 2023 07:22:26 GMT
Content-type: text/html; charset=utf-8
Content-Length: 246

<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="stationx-logo.png">stationx-logo.png</a></li>
</ul>
<hr>
</body>
</html>
```

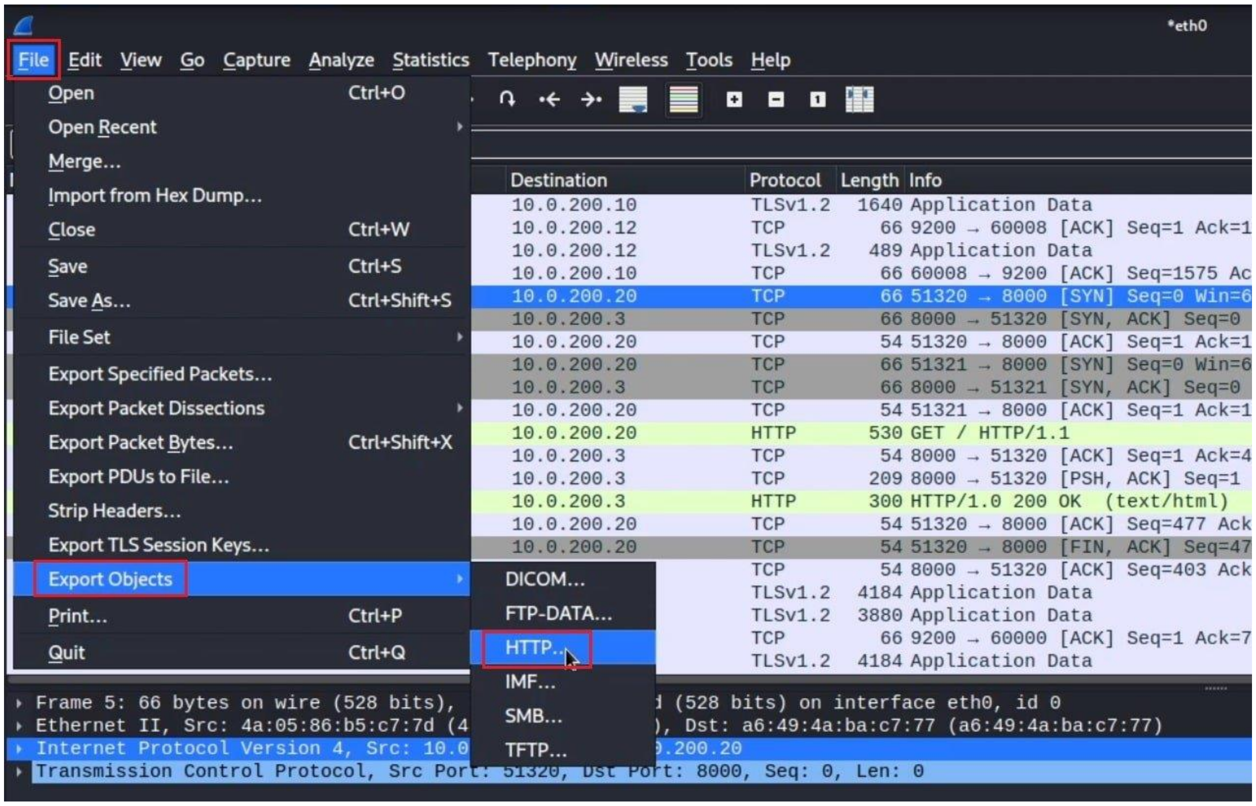
1 client pkt, 2 server pkts, 1 turn.

Entire conversation (877 bytes) Show data as ASCII Stream 1

Extracting Files

If you are on the red team, you often use Wireshark to extract sensitive information, such as usernames, passwords, and files. Wireshark can reconstruct files from the capture packets. This feature is incredibly useful as it allows you to extract various file types from network packets.

To extract files in Wireshark, select File > Export Objects. You can then select the object type you want to export based on the protocol used to transmit said object. In this case, an image was transferred using HTTP.

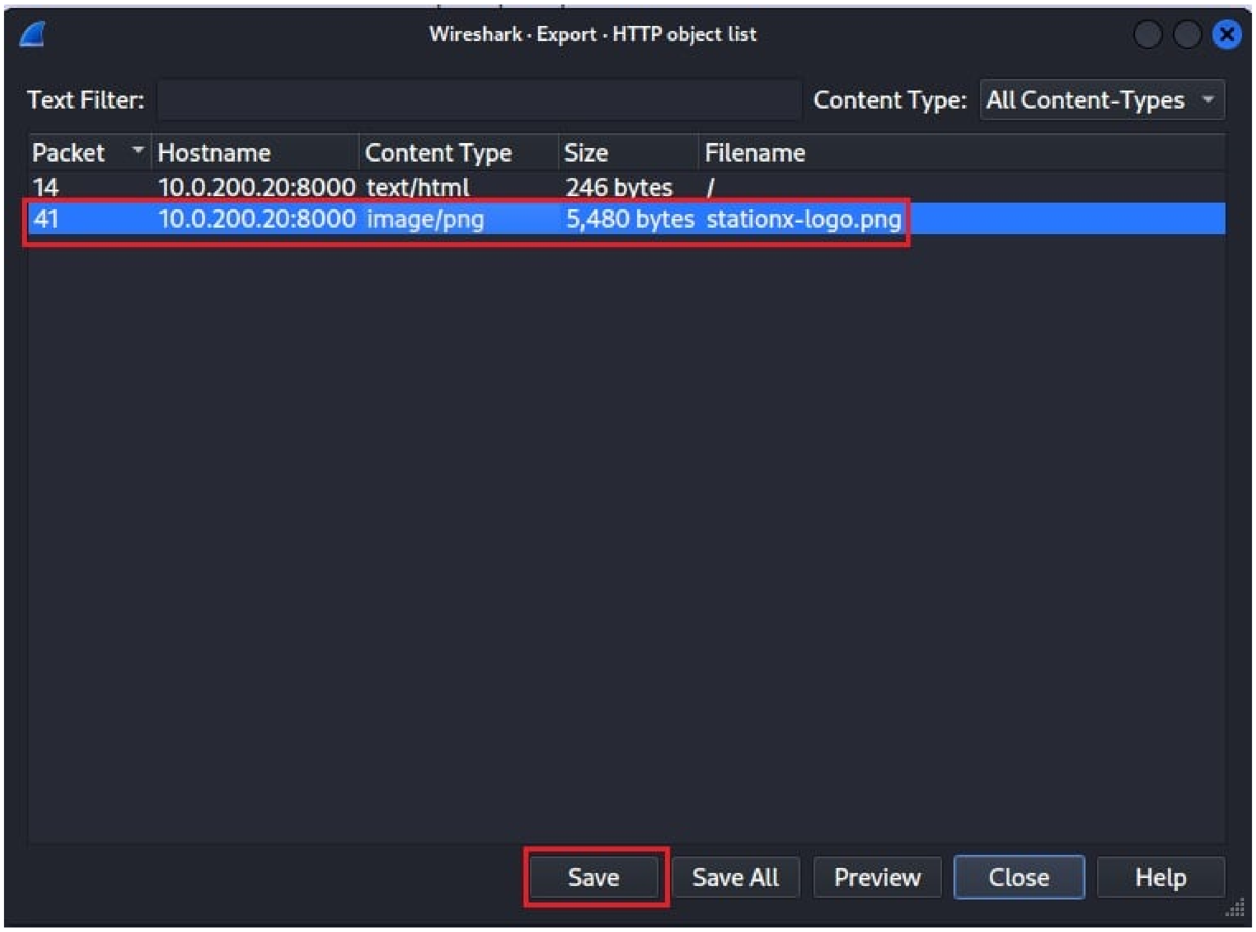


```
Destination     Protocol Length Info
10.0.200.10     TLSv1.2  1640 Application Data
10.0.200.12     TCP        66 9200 → 60008 [ACK] Seq=1 Ack=1
10.0.200.12     TLSv1.2   489 Application Data
10.0.200.10     TCP        66 60008 → 9200 [ACK] Seq=1575 Ac
10.0.200.20     TCP        66 51320 → 8000 [SYN] Seq=0 Win=6
10.0.200.3      TCP        66 8000 → 51320 [SYN, ACK] Seq=0
10.0.200.20     TCP        54 51320 → 8000 [ACK] Seq=1 Ack=1
10.0.200.20     TCP        66 51321 → 8000 [SYN] Seq=0 Win=6
10.0.200.3      TCP        66 8000 → 51321 [SYN, ACK] Seq=0
10.0.200.20     TCP        54 51321 → 8000 [ACK] Seq=1 Ack=1
10.0.200.20     HTTP      530 GET / HTTP/1.1
10.0.200.3      TCP        54 8000 → 51320 [ACK] Seq=1 Ack=4
10.0.200.3      TCP       209 8000 → 51320 [PSH, ACK] Seq=1
10.0.200.3      HTTP      300 HTTP/1.0 200 OK  (text/html)
10.0.200.20     TCP        54 51320 → 8000 [ACK] Seq=477 Ack
10.0.200.20     TCP        54 51320 → 8000 [FIN, ACK] Seq=47
                TCP        54 8000 → 51320 [ACK] Seq=403 Ack
                TLSv1.2  4184 Application Data
                TLSv1.2  3880 Application Data
                TCP        66 9200 → 60000 [ACK] Seq=1 Ack=7
                TLSv1.2  4184 Application Data

› Frame 5: 66 bytes on wire (528 bits),          d (528 bits) on interface eth0, id 0
› Ethernet II, Src: 4a:05:86:b5:c7:7d (4         ), Dst: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77)
› Internet Protocol Version 4, Src: 10.0         .200.20
› Transmission Control Protocol, Src Port: 51320, Dst Port: 8000, Seq: 0, Len: 0
```

Selecting HTTP opens a new window populated by all the objects transferred using HTTP in the packet capture file. There are two objects; a static HTML page and a PNG image.

To extract the image, select the packet (41) and click Save.

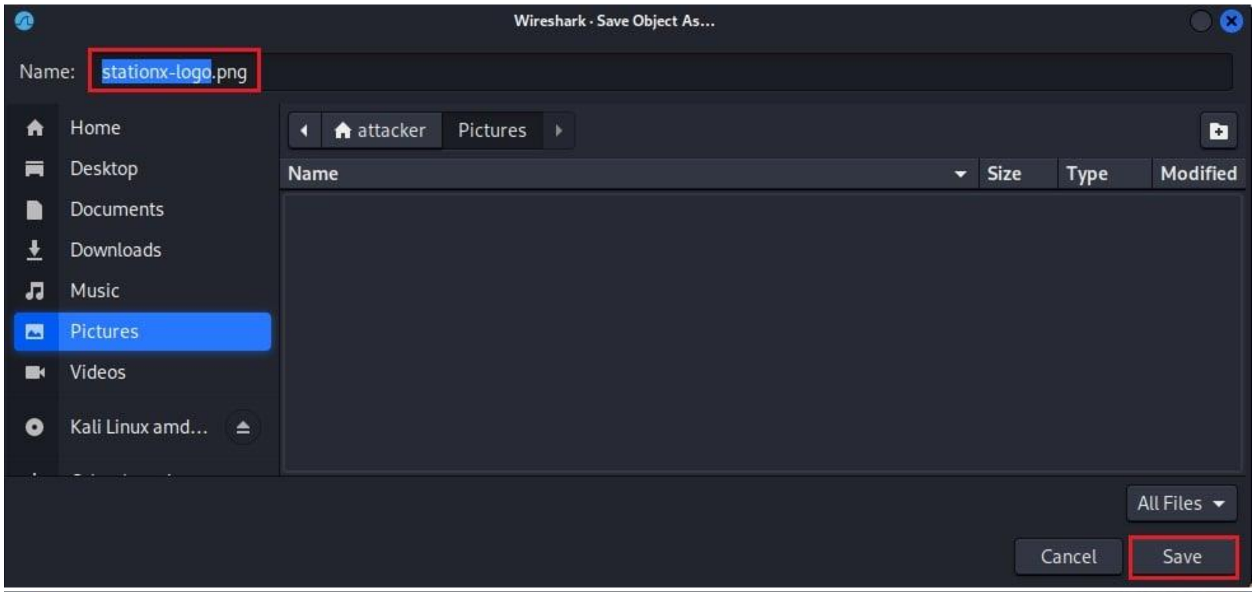

Wireshark · Export · HTTP object list

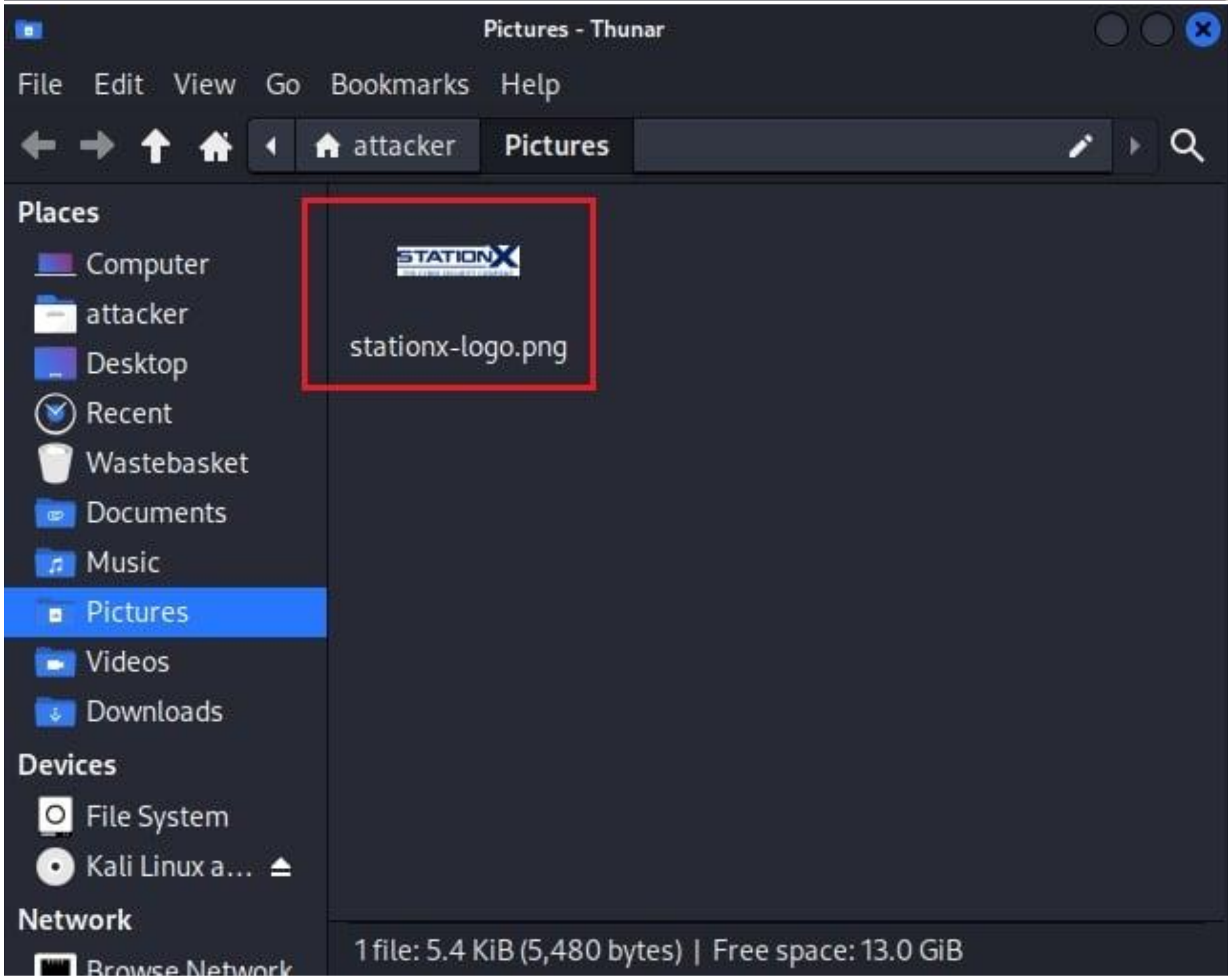
Text Filter: Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
14	10.0.200.20:8000	text/html	246 bytes	/
41	10.0.200.20:8000	image/png	5,480 bytes	stationx-logo.png

Now choose a location to save the image file to, and you can open the image in your favorite image viewer.

other binary file types.



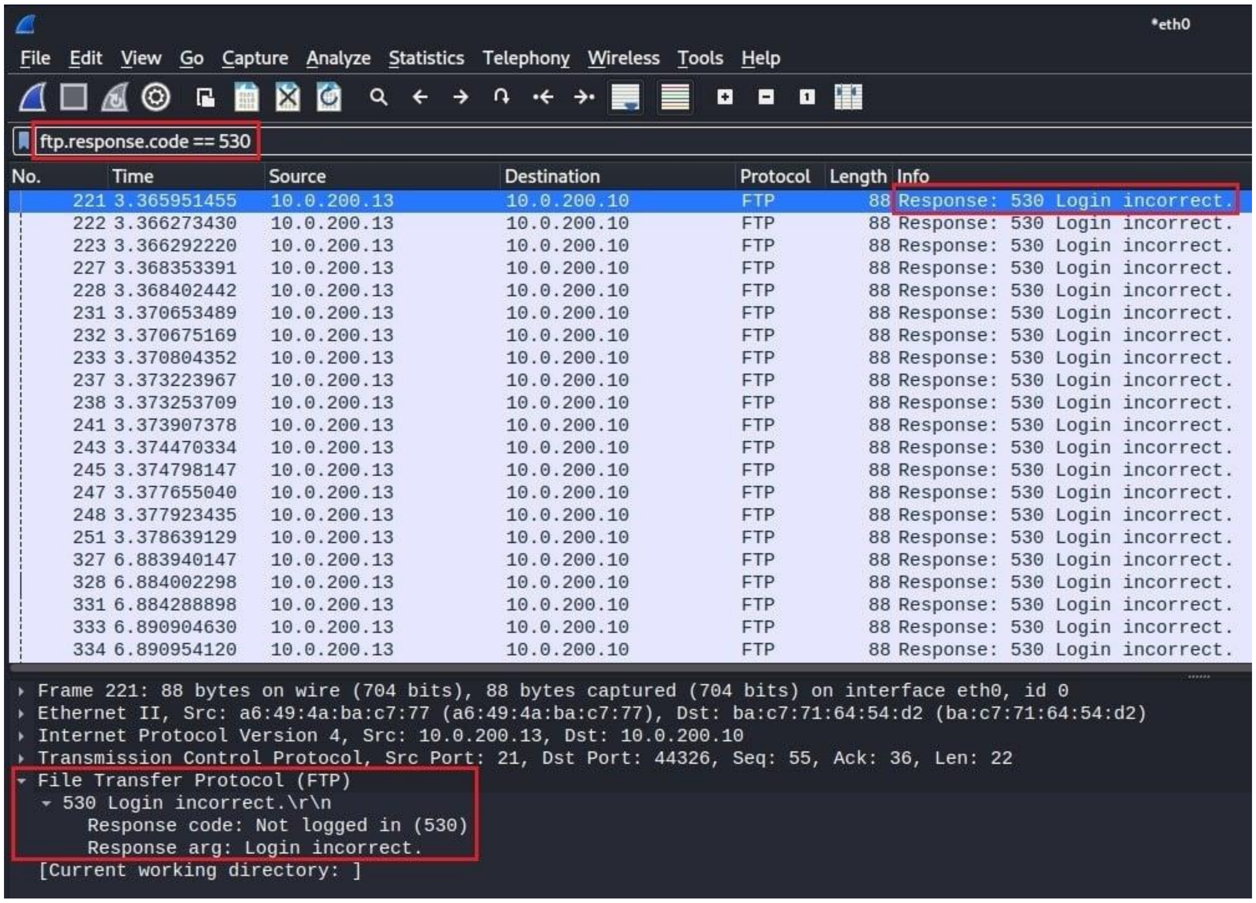


Using Wireshark, you can extract compressed files, executables, images, videos, email attachments, and other binary file types.

Identifying Attacks

If you are on the blue team, you will likely use Wireshark to identify cyber attacks by analyzing network traffic. This includes brute force attacks, port scans, and data exfiltration. Let's look at identifying an FTP brute force attack in Wireshark. The following packet capture was performed after running the **Hydra online cracking tool** against an FTP server.

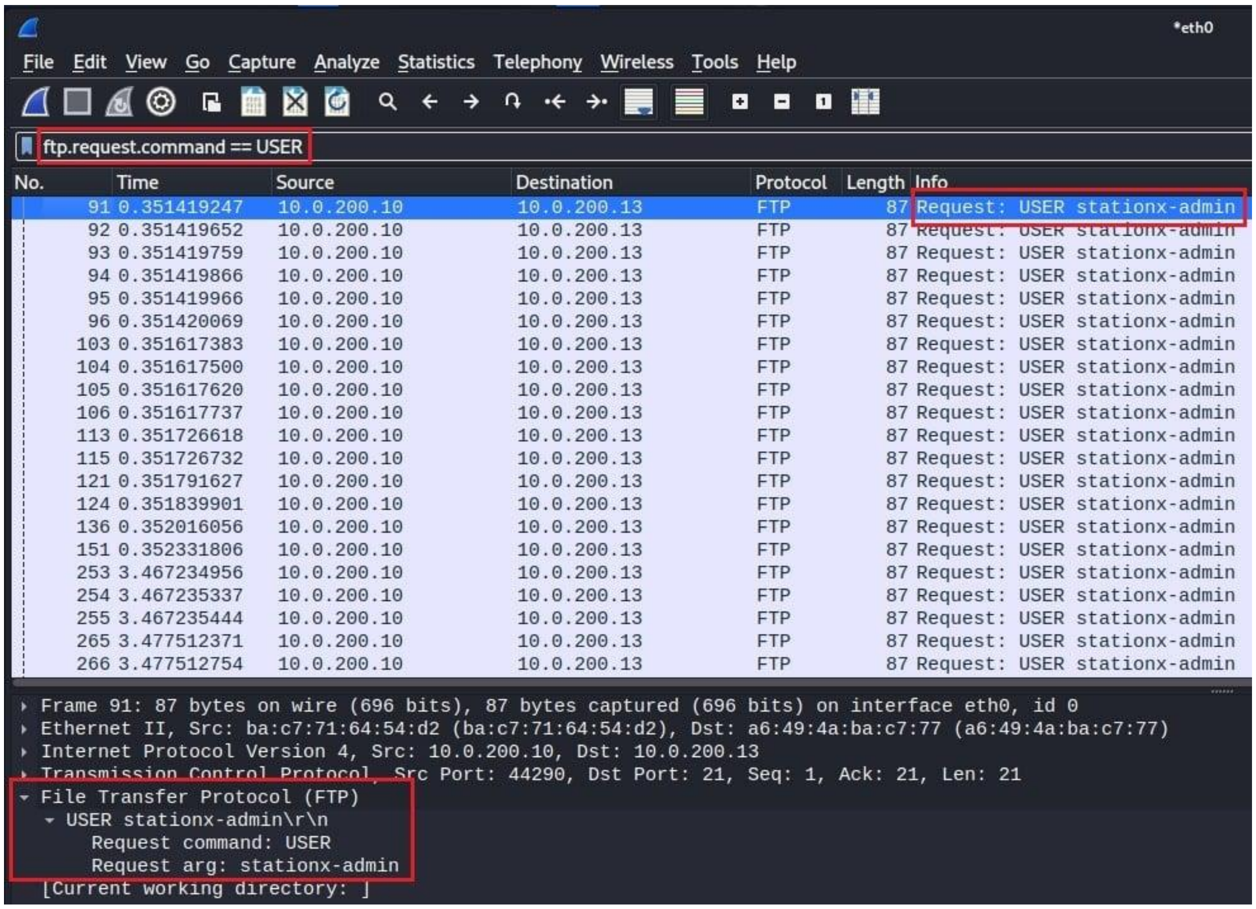
To see the FTP traffic, you can use the Wireshark display filter ftp.response.code == 530. This is the FTP response code for "Not logged in" and indicates an authentication failure. Seeing several of these errors with the same username but a different password indicates an FTP brute force attack.


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ftp.response.code == 530

```
No.    Time         Source        Destination   Protocol Length Info
   221 3.365951455  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   222 3.366273430  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   223 3.366292220  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   227 3.368353391  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   228 3.368402442  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   231 3.370653489  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   232 3.370675169  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   233 3.370804352  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   237 3.373223967  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   238 3.373253709  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   241 3.373907378  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   243 3.374470334  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   245 3.374798147  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   247 3.377655040  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   248 3.377923435  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   251 3.378639129  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   327 6.883940147  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   328 6.884002298  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   331 6.884288898  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   333 6.890904630  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.
   334 6.890954120  10.0.200.13   10.0.200.10   FTP         88 Response: 530 Login incorrect.

› Frame 221: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface eth0, id 0
› Ethernet II, Src: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77), Dst: ba:c7:71:64:54:d2 (ba:c7:71:64:54:d2)
› Internet Protocol Version 4, Src: 10.0.200.13, Dst: 10.0.200.10
› Transmission Control Protocol, Src Port: 21, Dst Port: 44326, Seq: 55, Ack: 36, Len: 22
˅ File Transfer Protocol (FTP)
  ˅ 530 Login incorrect.\r\n
      Response code: Not logged in (530)
      Response arg: Login incorrect.
  [Current working directory: ]
```

To find out if they belong to the same user, change the display filter to ftp.request.command == USER. This will show you all the FTP login requests where a username was specified.

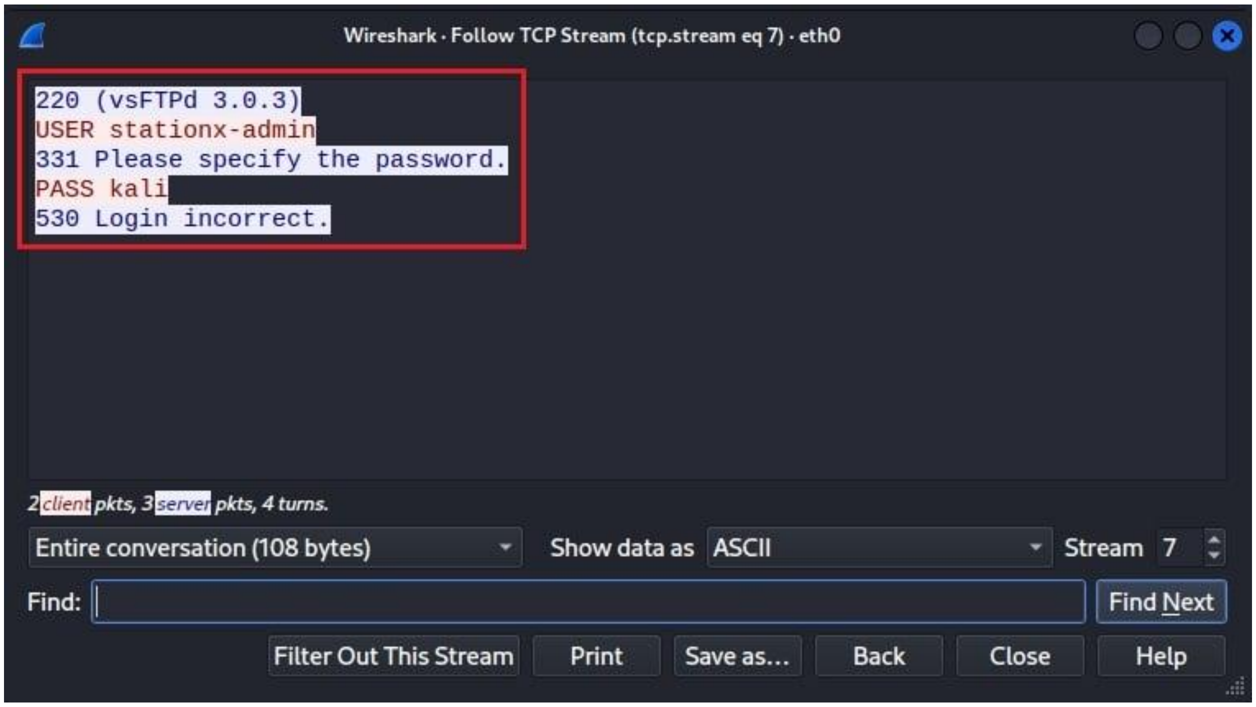

*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ftp.request.command == USER
```

```
No.  Time         Source       Destination  Protocol Length Info
  91 0.351419247  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
  92 0.351419652  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
  93 0.351419759  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
  94 0.351419866  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
  95 0.351419966  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
  96 0.351420069  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 103 0.351617383  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 104 0.351617500  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 105 0.351617620  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 106 0.351617737  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 113 0.351726618  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 115 0.351726732  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 121 0.351791627  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 124 0.351839901  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 136 0.352016056  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 151 0.352331806  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 253 3.467234956  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 254 3.467235337  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 255 3.467235444  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 265 3.477512371  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
 266 3.477512754  10.0.200.10  10.0.200.13  FTP      87 Request: USER stationx-admin
```

```
› Frame 91: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface eth0, id 0
› Ethernet II, Src: ba:c7:71:64:54:d2 (ba:c7:71:64:54:d2), Dst: a6:49:4a:ba:c7:77 (a6:49:4a:ba:c7:77)
› Internet Protocol Version 4, Src: 10.0.200.10, Dst: 10.0.200.13
› Transmission Control Protocol, Src Port: 44290, Dst Port: 21, Seq: 1, Ack: 21, Len: 21
▾ File Transfer Protocol (FTP)
  ▾ USER stationx-admin\r\n
      Request command: USER
      Request arg: stationx-admin
  [Current working directory: ]
```

Next, to determine if different passwords are being tried, right-click on a network packet, and select Follow > TCP Stream.



```
220 (vsFTPd 3.0.3)
USER stationx-admin
331 Please specify the password.
PASS kali
530 Login incorrect.
```

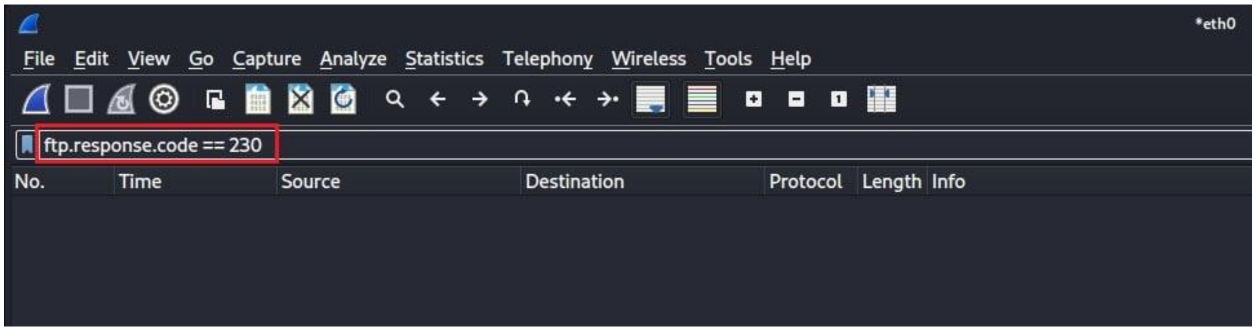
2 client pkts, 3 server pkts, 4 turns.

Entire conversation (108 bytes)   Show data as ASCII   Stream 7

Find:

Here the same user tries multiple passwords to log in, indicating a password spray attack against the user stationx-admin.

To determine if any attempts were successful, use the display filter ftp.response.code == 230 to look for “User logged in, proceed.”


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ftp.response.code == 230
No.  Time  Source  Destination  Protocol  Length  Info
```

No login attempts were successful!