

Wireshark

What is wireshark?

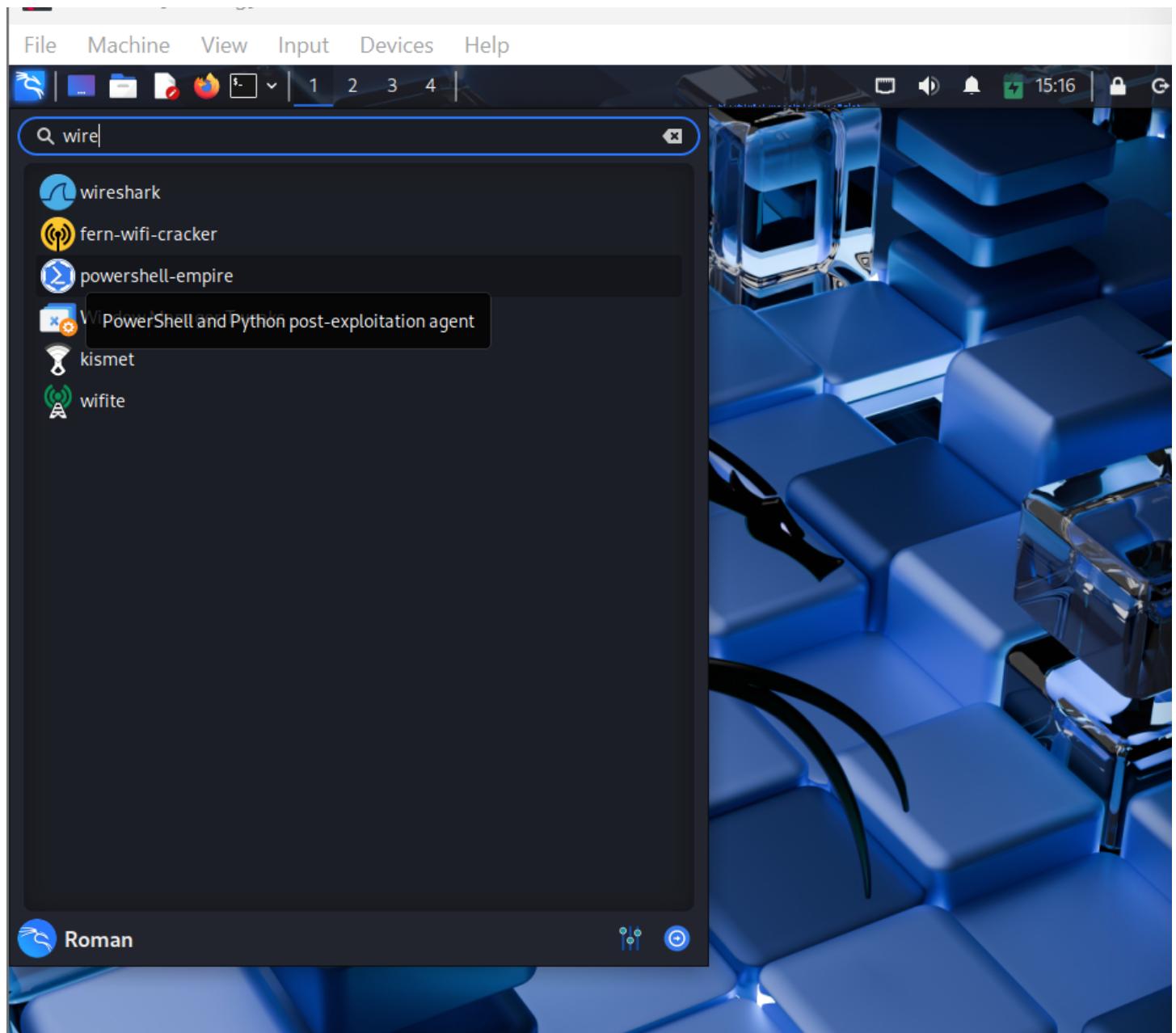
Wireshark is a **free and open-source network protocol analyzer** used for **network troubleshooting, analysis, software development, and cybersecurity monitoring**. It captures packets in real time and displays them in a human-readable format, allowing users to inspect network traffic at various levels.

Wireshark is considered an essential tool for **network engineers, penetration testers, cybersecurity analysts, and IT professionals**.

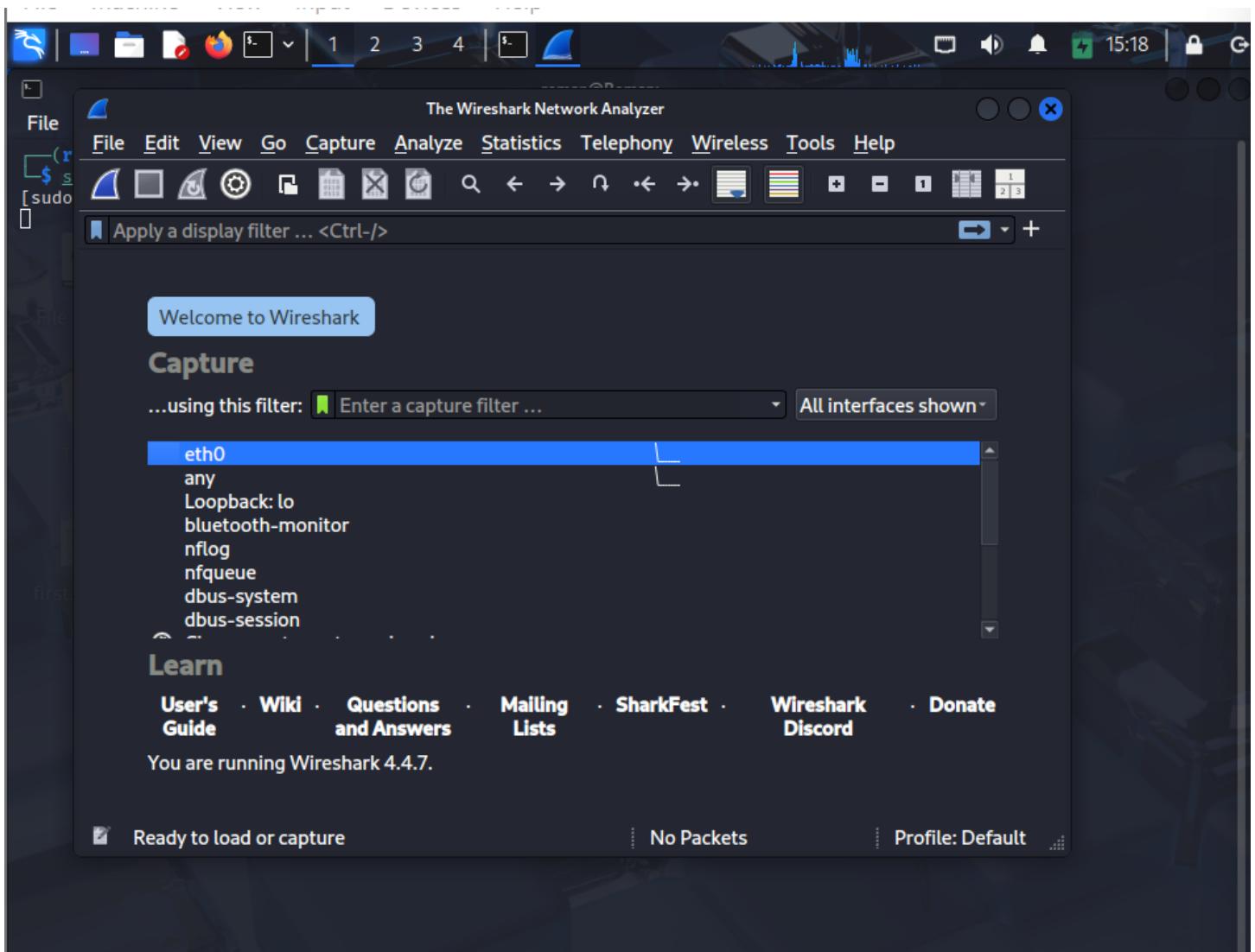
Wireshark Features

- Captures live network traffic in real time
- Supports deep inspection of over 2,000 network protocols
- Allows filtering of captured traffic using powerful display filters
- Highlights packets using customizable color rules for better visibility
- Reconstructs complete TCP/HTTP conversations using the “Follow Stream” feature
- Supports multiple platforms: Windows, Linux, macOS, and Unix
- Enables marking and commenting on packets for easy reference
- Provides export options in various formats such as .pcap, .txt, and .csv
- Includes VoIP analysis tools with call flow diagrams and audio playback
- Supports decryption of SSL/TLS and WPA2 traffic (when keys are available)
- Offers command-line tools like tshark for automated or remote analysis
- Allows plugin support and custom dissectors for protocol extension
- Displays detailed statistics: protocol hierarchy, endpoint conversations, I/O graphs
- Offers customizable layouts and filter profiles for different analysis needs

Step1: Open the Wireshark Application

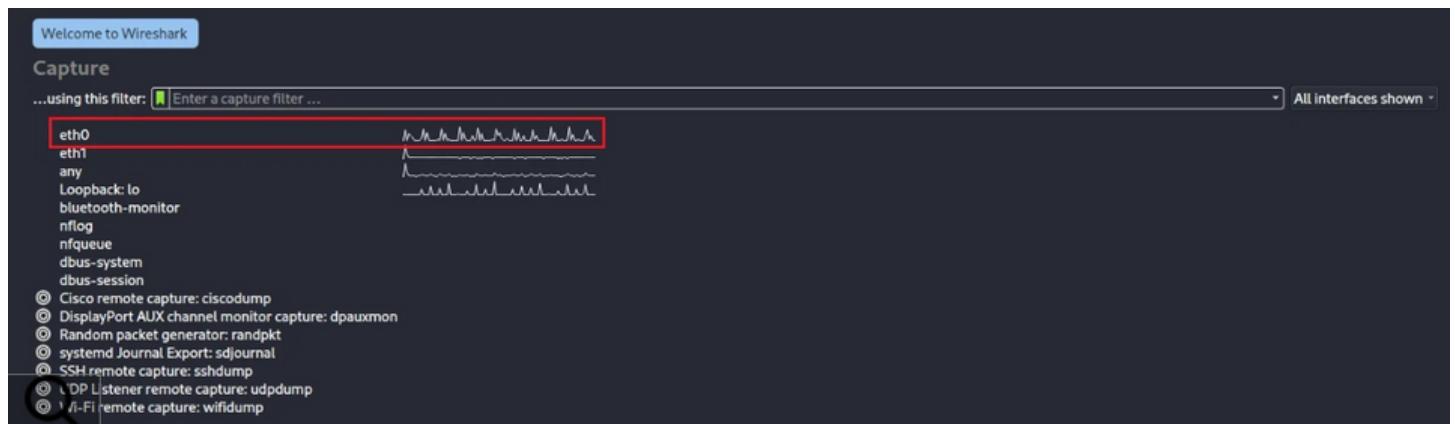


Or you can simply write on terminal wireshark and it will open the wireshark Application



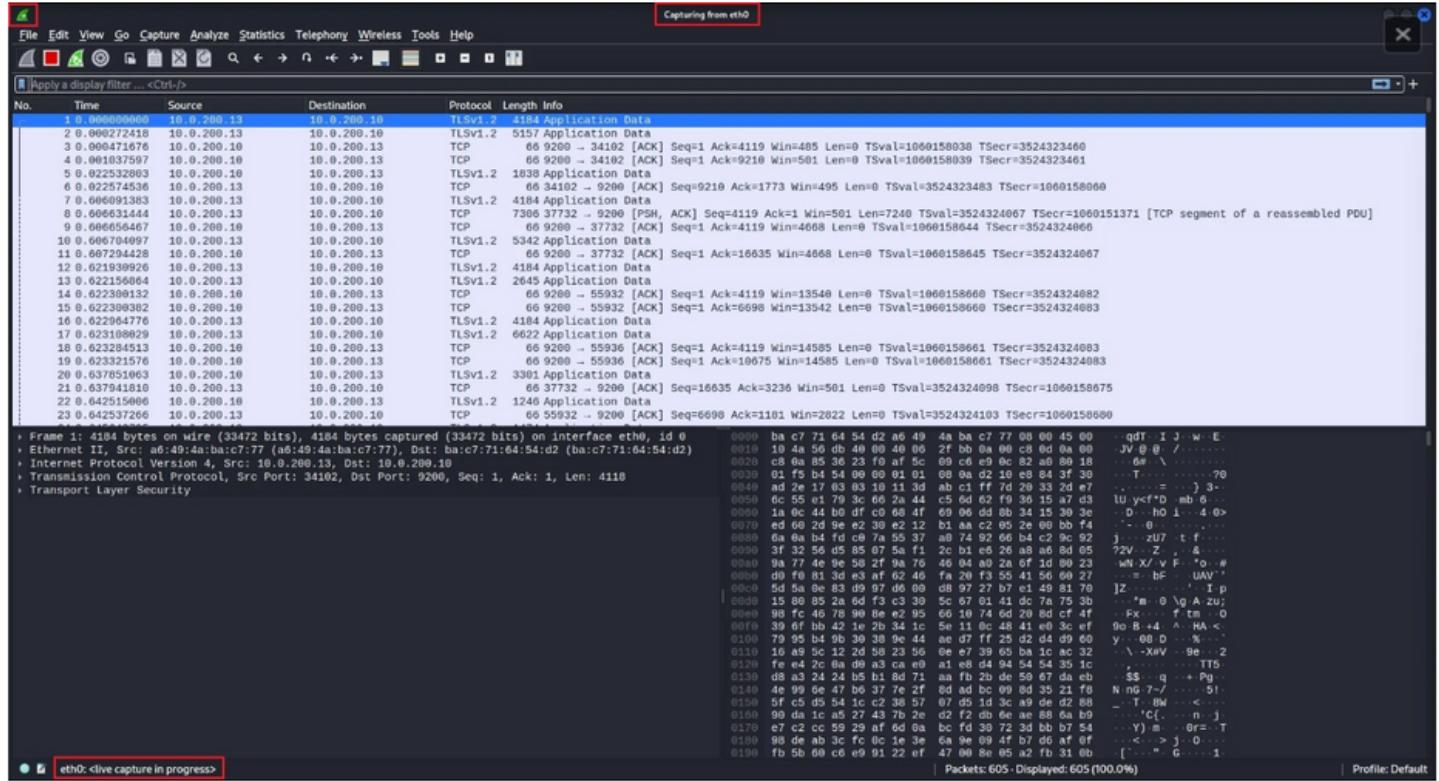
Step2: Select a networking interface to capture Traffic on:

- **Wi-Fi** – Captures traffic from your wireless internet connection
- **Ethernet** – Captures traffic from a wired network (LAN cable)
- **Loopback (lo)** – Captures internal traffic within your own computer (e.g., localhost)
- **VMware/VirtualBox** – Captures traffic from virtual machines or virtual networks
- **Bluetooth** – Captures Bluetooth-related network activity
-

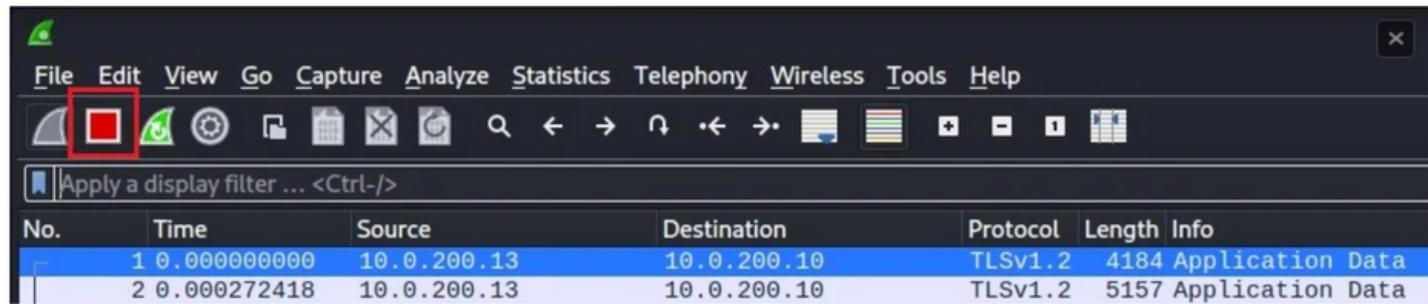


Step3: Stopping the Packet Capture

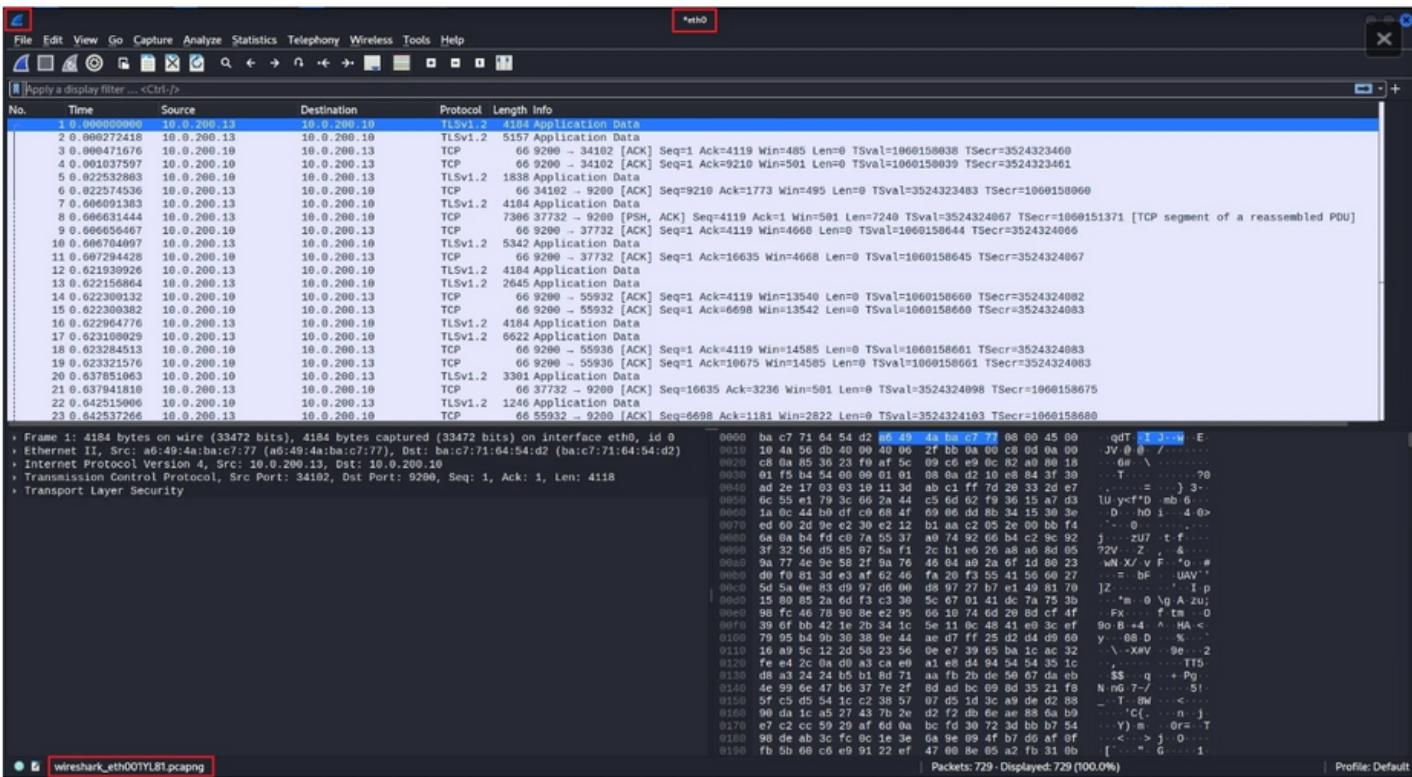
Double-Clicking on the capture interface you want to Capture Traffic to start Capturing Packet



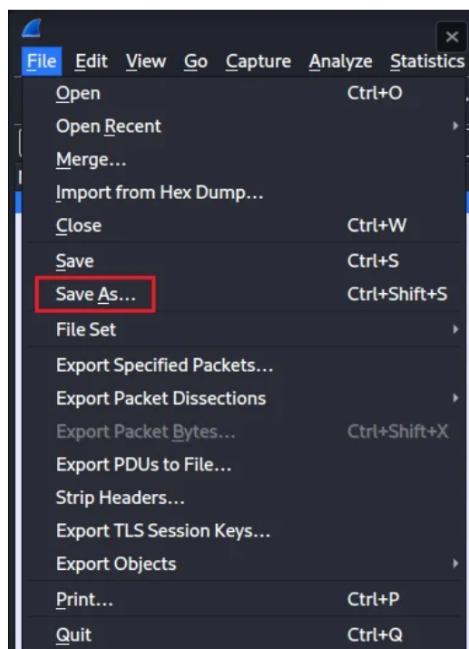
To Stop Capture Packets, Click on red Stop Capture Button

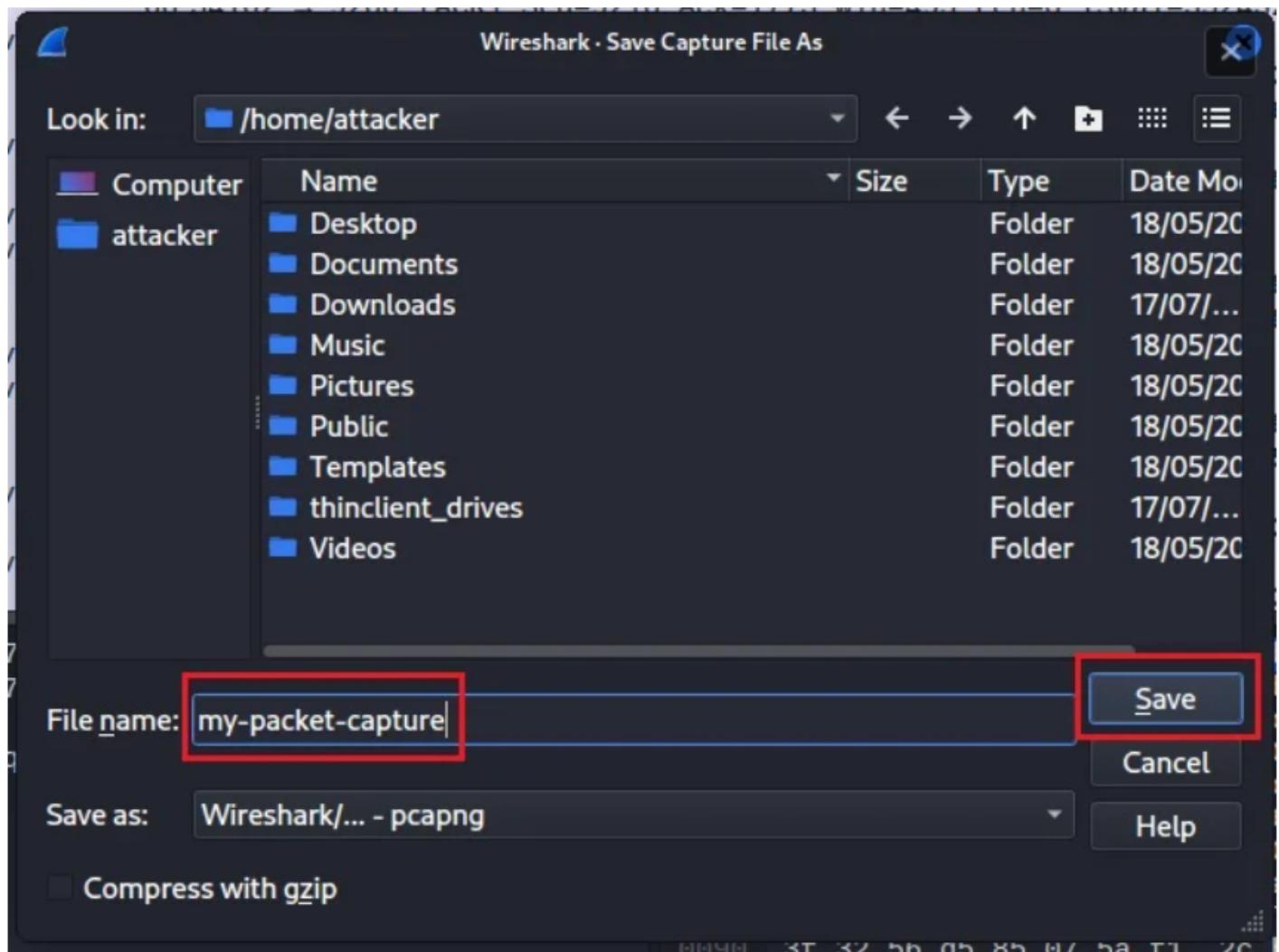


This will Capture into temporary file and allow you to perform your analysis

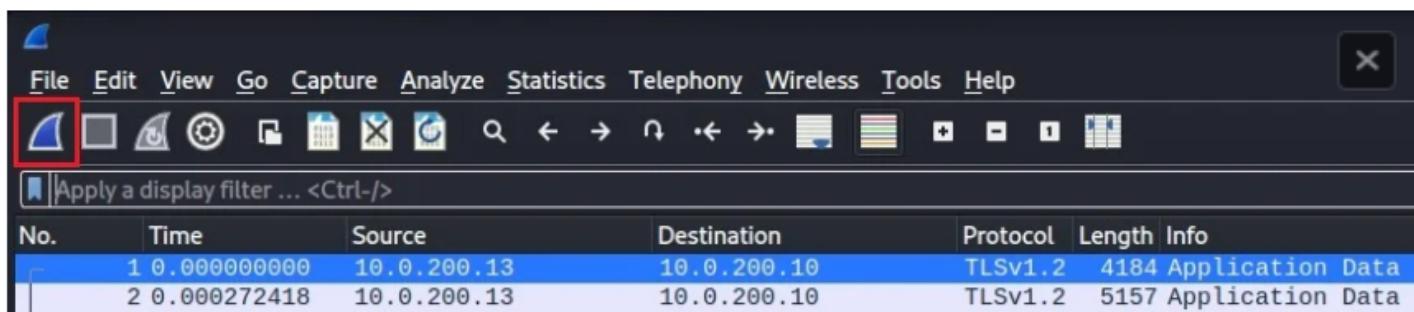


to save Your data into a specific file. Select File > Save As, then enter the location and name of the file under which you want to save this data





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



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.

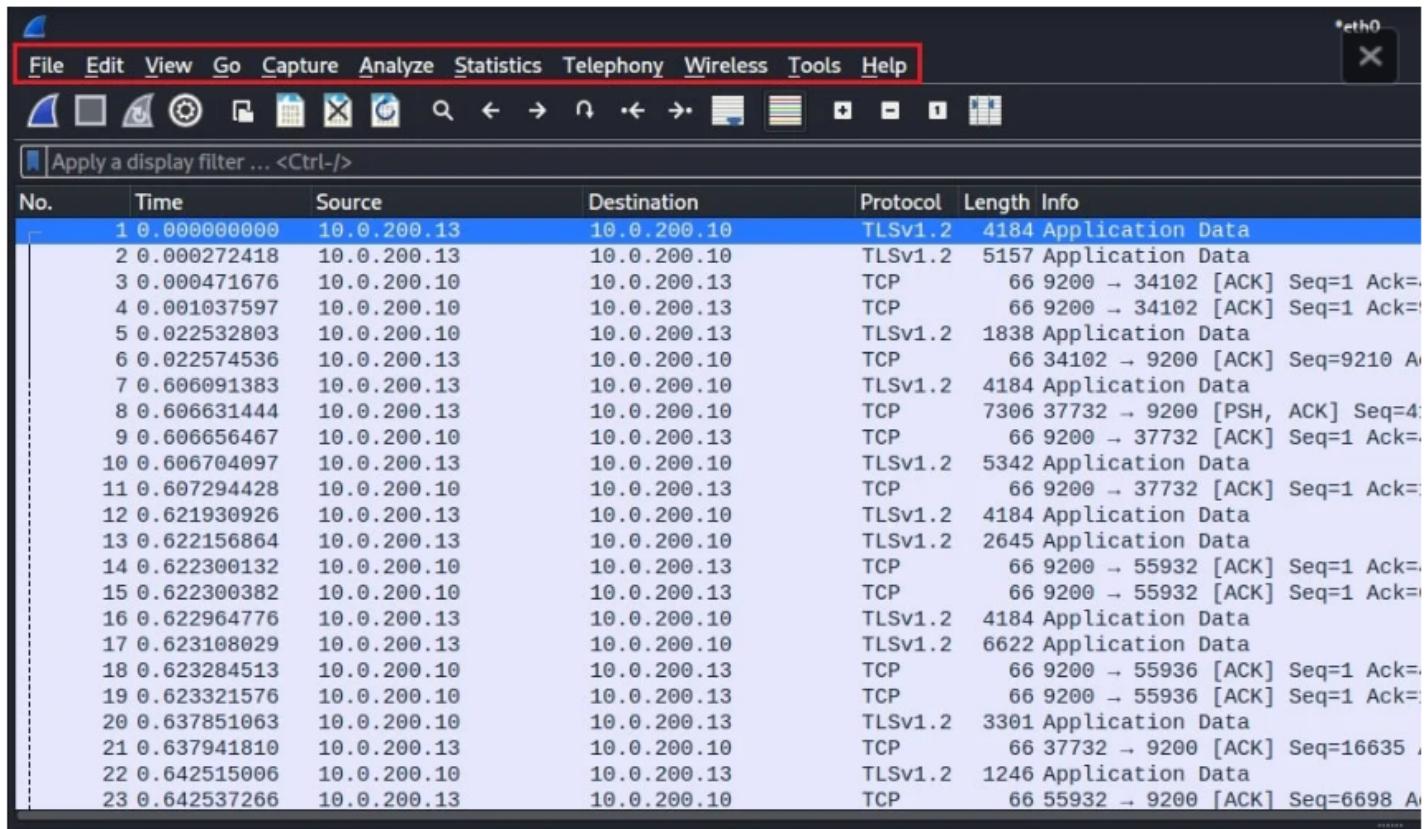
However, you usually want to capture all the network traffic on an interface and use Wireshark's **display filters** to filter a saved packet capture. This means you won't miss any relevant network traffic that the

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 comprised of four components

The Menu Bar



1. File

- Open, save, close capture files
- Export packets in various formats
- Capture file properties
- Print selected packets

2. Edit

- Find packets (by string, hex, or display filter)
- Mark, unmark, or ignore packets
- Configuration profiles and preferences

3. View

- Customize layout, packet bytes, and colorization
- Show/hide interface elements
- Zoom in/out on packet list or data

4. Go

- Navigate through packets: next, previous, last marked, etc.
- Useful for quickly jumping between points of interest

5. Capture

- Start, stop, or restart a live capture
- Select input interfaces
- Set capture filters and options

6. Analyze

- Apply or disable display filters
- Follow TCP/HTTP streams
- Decode as specific protocol
- Enable/disable protocol dissectors

7. Statistics

- View summaries like protocol hierarchy, endpoints, conversations, IO graphs
- Analyze flow of traffic and packet types

8. Telephony

- Analyze VoIP, RTP, SIP calls
- Useful for voice traffic inspection

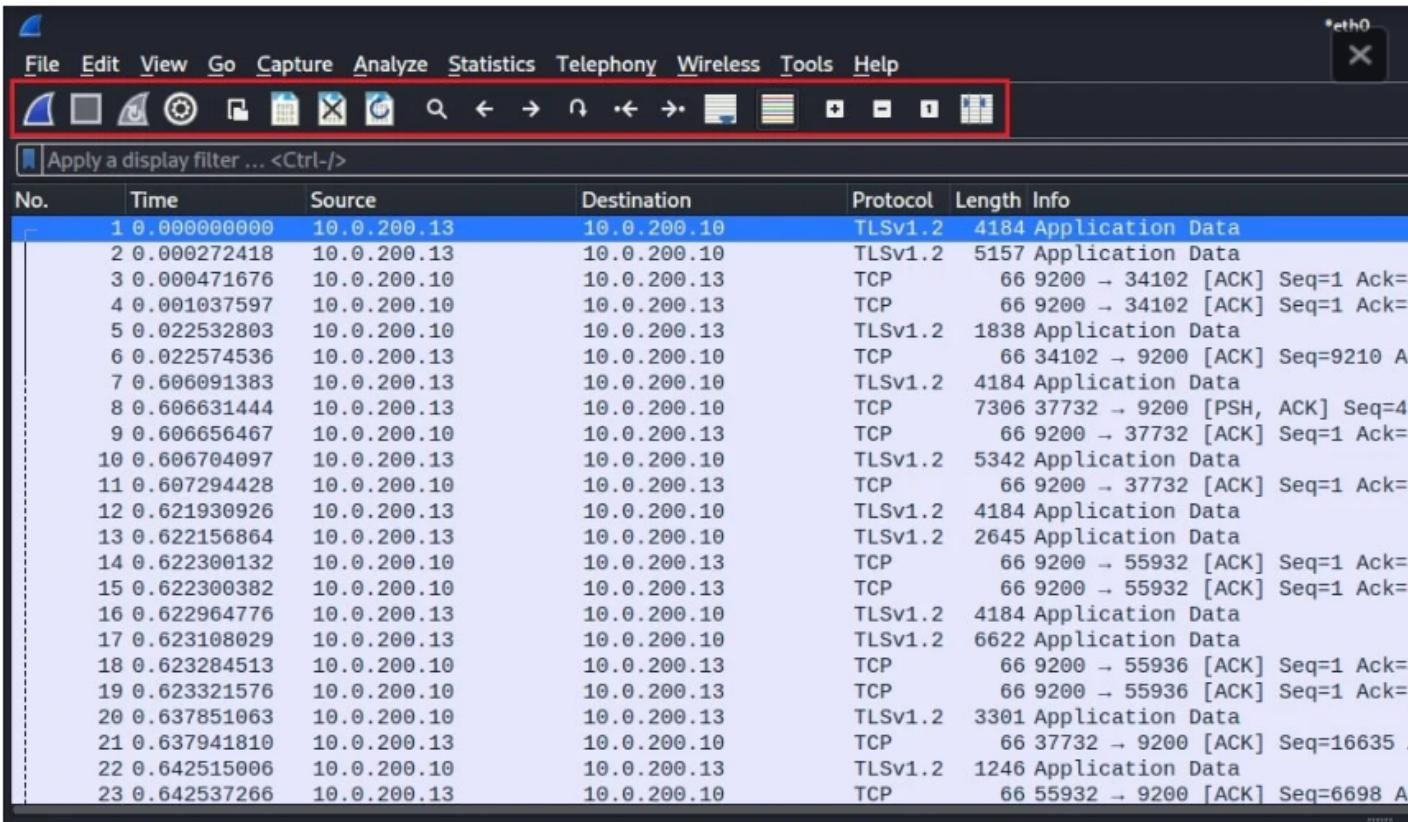
9. Tools

- Access additional tools and external plugins

10. Help

- Access Wireshark documentation
- About Wireshark, plugin info, and keyboard shortcuts

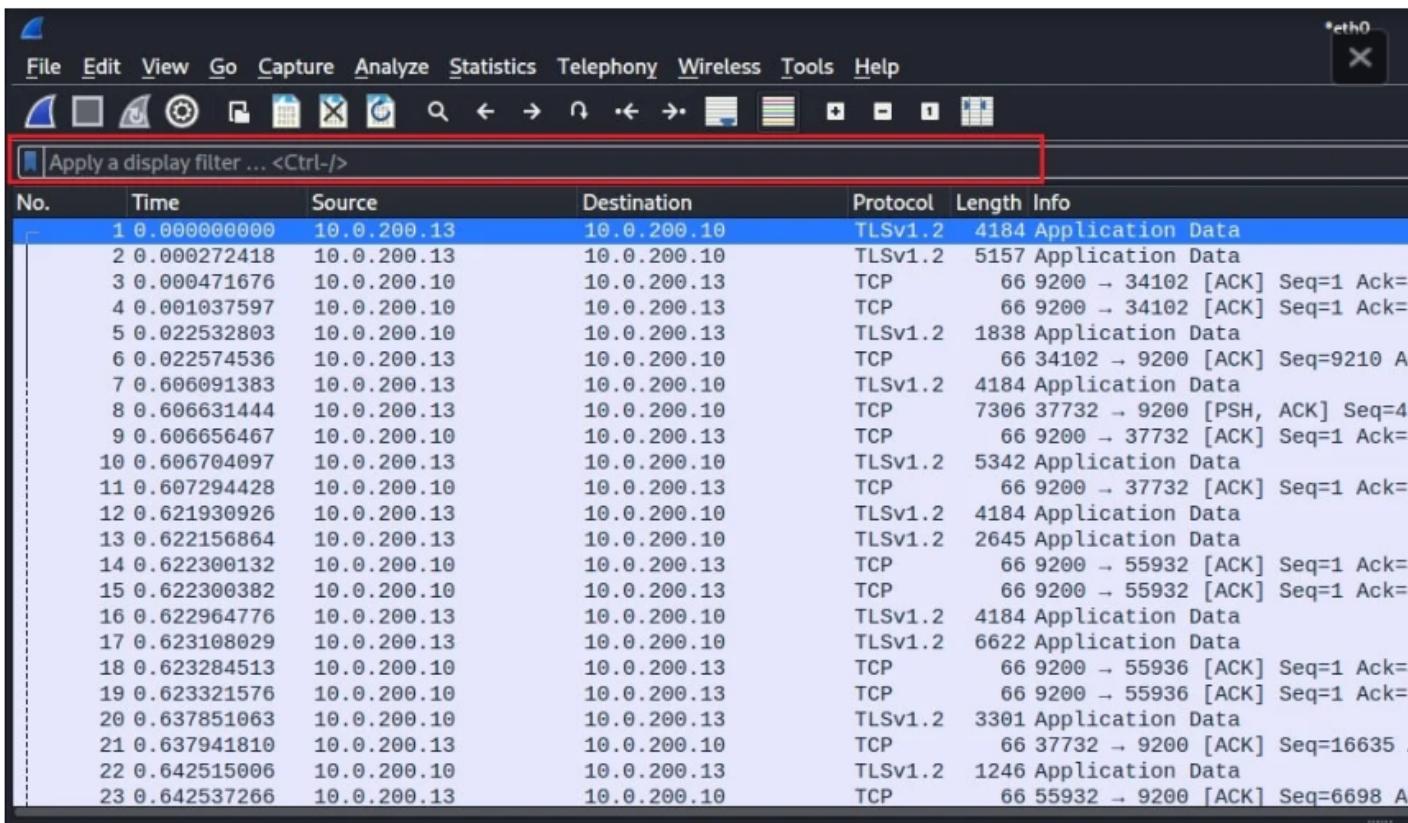
The main Toolbar



| 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 A |
| 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



| 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 A |
| 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

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------------|-------------|-------------|----------|--------|--------------------------------|
| 1 | 0.000000000000 | 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 | 0.0022532803 | 10.0.200.10 | 10.0.200.13 | TLSv1.2 | 1838 | Application Data |
| 6 | 0.0022574536 | 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 | 0.6066631444 | 10.0.200.13 | 10.0.200.10 | TCP | 7306 | 37732 → 9200 [PSH, ACK] Seq=1 |
| 9 | 0.6066656467 | 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 A |
| 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

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-------------|-------------|----------|--------|------------------------------------------------------------------|
| 301 | 05.063822468 | 10.0.200.13 | 10.0.200.3 | ICMP | 98 | Echo (ping) request id=0x7f01, seq=1/256, ttl=64 (reply in 302) |
| 302 | 05.063313842 | 10.0.200.3 | 10.0.200.13 | ICMP | 98 | Echo (ping) reply id=0x7f01, seq=1/256, ttl=128 (request in 301) |
| 314 | 06.080684234 | 10.0.200.13 | 10.0.200.3 | ICMP | 98 | Echo (ping) request id=0x7f01, seq=2/512, ttl=64 (reply in 315) |
| 315 | 06.080733358 | 10.0.200.3 | 10.0.200.13 | ICMP | 98 | Echo (ping) reply id=0x7f01, seq=2/512, ttl=128 (request in 314) |
| 316 | 07.104623148 | 10.0.200.13 | 10.0.200.3 | ICMP | 98 | Echo (ping) request id=0x7f01, seq=3/768, ttl=64 (reply in 317) |
| 317 | 07.104663867 | 10.0.200.3 | 10.0.200.13 | ICMP | 98 | Echo (ping) reply id=0x7f01, 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:b4:c7:77 (a6:49:4a:b4:c7:77), Dst: 4a:05:86:b5:c7:7d (4a:05:86:b5:c7:7d)
> Internet Protocol Version 4, Src: 10.0.200.13, Dst: 10.0.200.3
> Internet Control Message Protocol

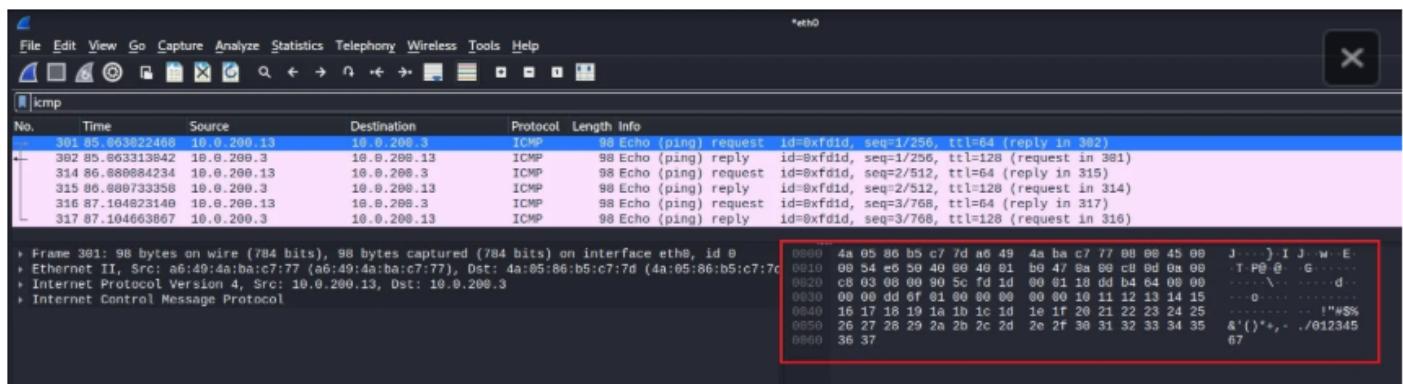
```

0000 4a 95 86 b5 c7 7d a6 49 4a ba c7 77 88 00 45 00 J... ] I J H E
0010 00 b4 e6 50 40 00 40 01 b0 47 8a 00 c8 0d 08 00 T P@ @ G ...
0020 c8 03 08 00 00 5c fd 1d 00 01 18 dd b4 64 08 00 . . \ . d ...
0030 00 00 dd 0f 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 . . . . 1%"$%
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



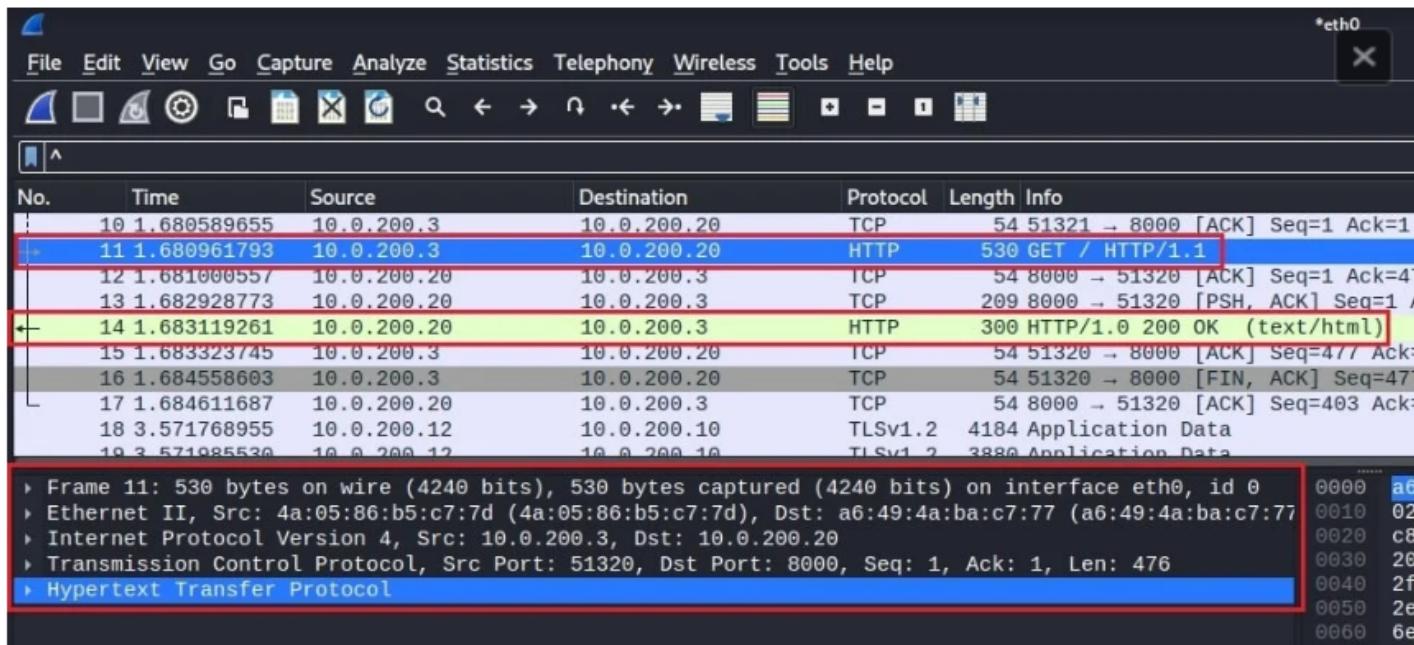
To the right of the packet details pane is the packet bytes pane. This shows the corresponding bytes Wireshark extracted from the wire to reconstruct the network packet data. These bytes are shown in hexadecimal format and are highlighted when you select information within the packet details pane. This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

Analysing 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

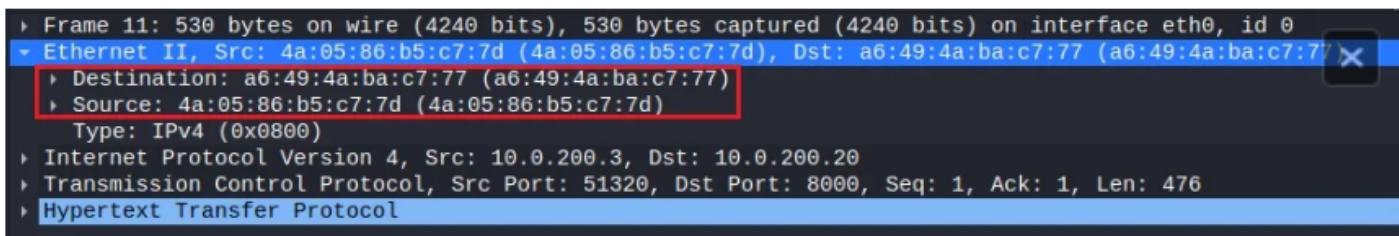
To examine a packet, select it within the packet list pane. This will populate the packet details pane with that packet's network information.



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).



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: 0xa622 (43622)
    > 010. .... = Flags: 0x2, Don't fragment
        ...0 0000 0000 0000 = Fragment Offset: 0
        Time to Live: 128
        Protocol: TCP (6)
        Header Checksum: 0xa75 [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).

The screenshot shows the Wireshark interface with the following details:

- File Bar:** File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help.
- Toolbar:** Contains icons for file operations, search, and zoom.
- Display Filter Bar:** Shows the filter "icmp".
- Packet List:** A table showing the following columns: No., Time, Source, Destination, Protocol, Length, Info, and a hex dump column on the far right. The "Protocol" column is highlighted with a red border.
- Bottom Status Bar:** Shows the details of the selected packet (Frame 301) and the number of bytes (0000 to 0060).

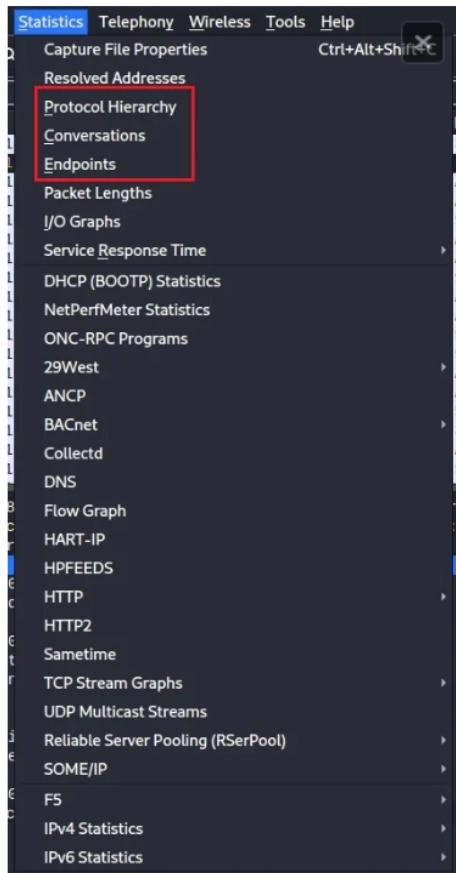
| No. | Time | Source | Destination | Protocol | Length | Info | |
|-----|--------------|-------------|-------------|----------|--------|----------------------------|------|
| 301 | 85.063022468 | 10.0.200.13 | 10.0.200.3 | ICMP | 98 | Echo (ping) request id=0xf | 0000 |
| 302 | 85.063313042 | 10.0.200.3 | 10.0.200.13 | ICMP | 98 | Echo (ping) reply id=0xf | 0010 |
| 314 | 86.080084234 | 10.0.200.13 | 10.0.200.3 | ICMP | 98 | Echo (ping) request id=0xf | 0020 |
| 315 | 86.080733358 | 10.0.200.3 | 10.0.200.13 | ICMP | 98 | Echo (ping) reply id=0xf | 0030 |
| 316 | 87.104023140 | 10.0.200.13 | 10.0.200.3 | ICMP | 98 | Echo (ping) request id=0xf | 0040 |
| 317 | 87.104663867 | 10.0.200.3 | 10.0.200.13 | ICMP | 98 | Echo (ping) reply id=0xf | 0050 |
| | | | | | | | 0060 |

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.

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.



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.

Wireshark - Protocol Hierarchy Statistics - eth0

| 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.

Close Copy Help

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

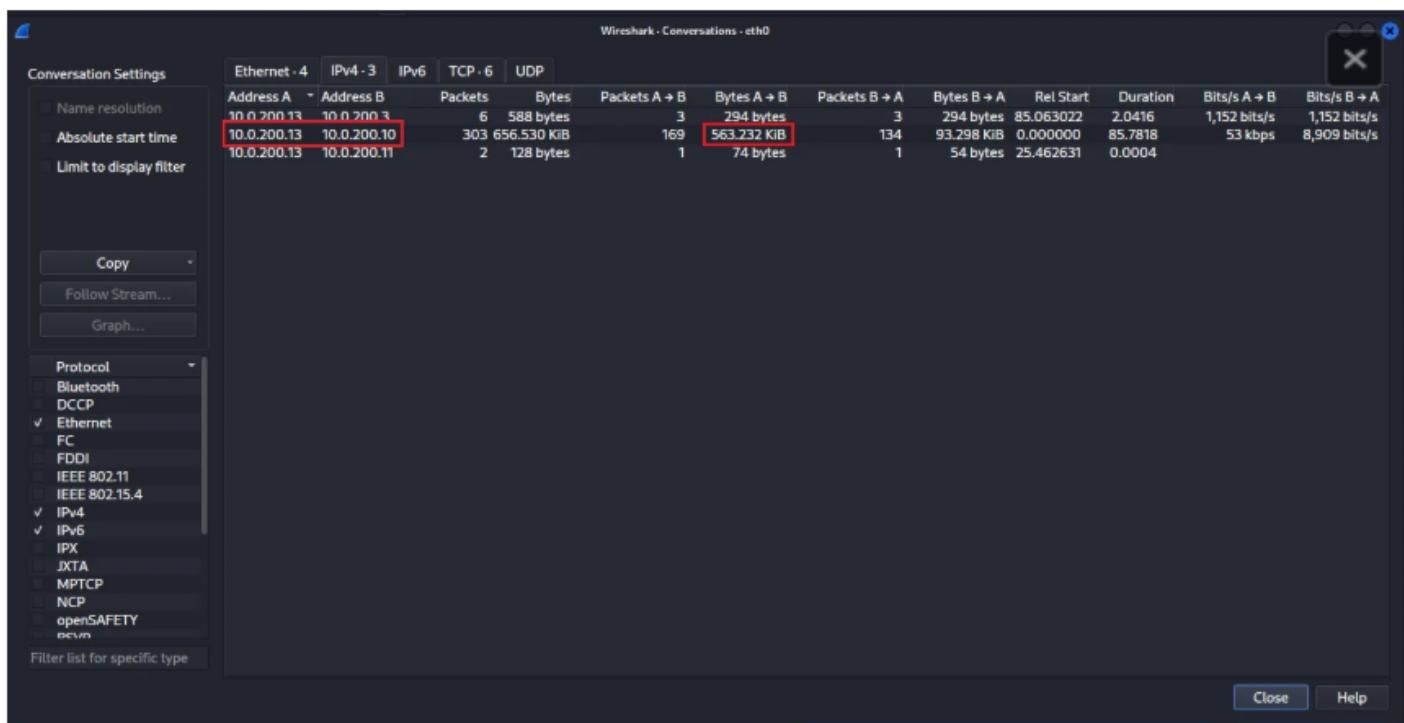
| Endpoint Settings | | Ethernet · 5 | IPv4 · 4 | IPv6 | TCP · 8 | UDP | | | | | | |
|--------------------------|-------------------------|--------------|-----------|------------|------------|------------|------------|------------|---------|------|-----------|-----------------|
| | | Address | _packets_ | Bytes | Tx Packets | Tx Bytes | Rx Packets | Rx Bytes | Country | City | AS Number | AS Organization |
| <input type="checkbox"/> | Name resolution | 10.0.200.3 | 6 | 588 bytes | 3 | 294 bytes | 3 | 294 bytes | | | | |
| <input type="checkbox"/> | Limit to display filter | 10.0.200.10 | 303 | 656.530 kB | 134 | 93.298 kB | 169 | 563.232 kB | | | | |
| <input type="checkbox"/> | | 10.0.200.11 | 2 | 128 bytes | 1 | 54 bytes | 1 | 74 bytes | | | | |
| <input type="checkbox"/> | | 10.0.200.13 | 311 | 657.229 kB | 173 | 563.592 kB | 138 | 93.638 kB | | | | |

Copy

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

Close Help

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.



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.

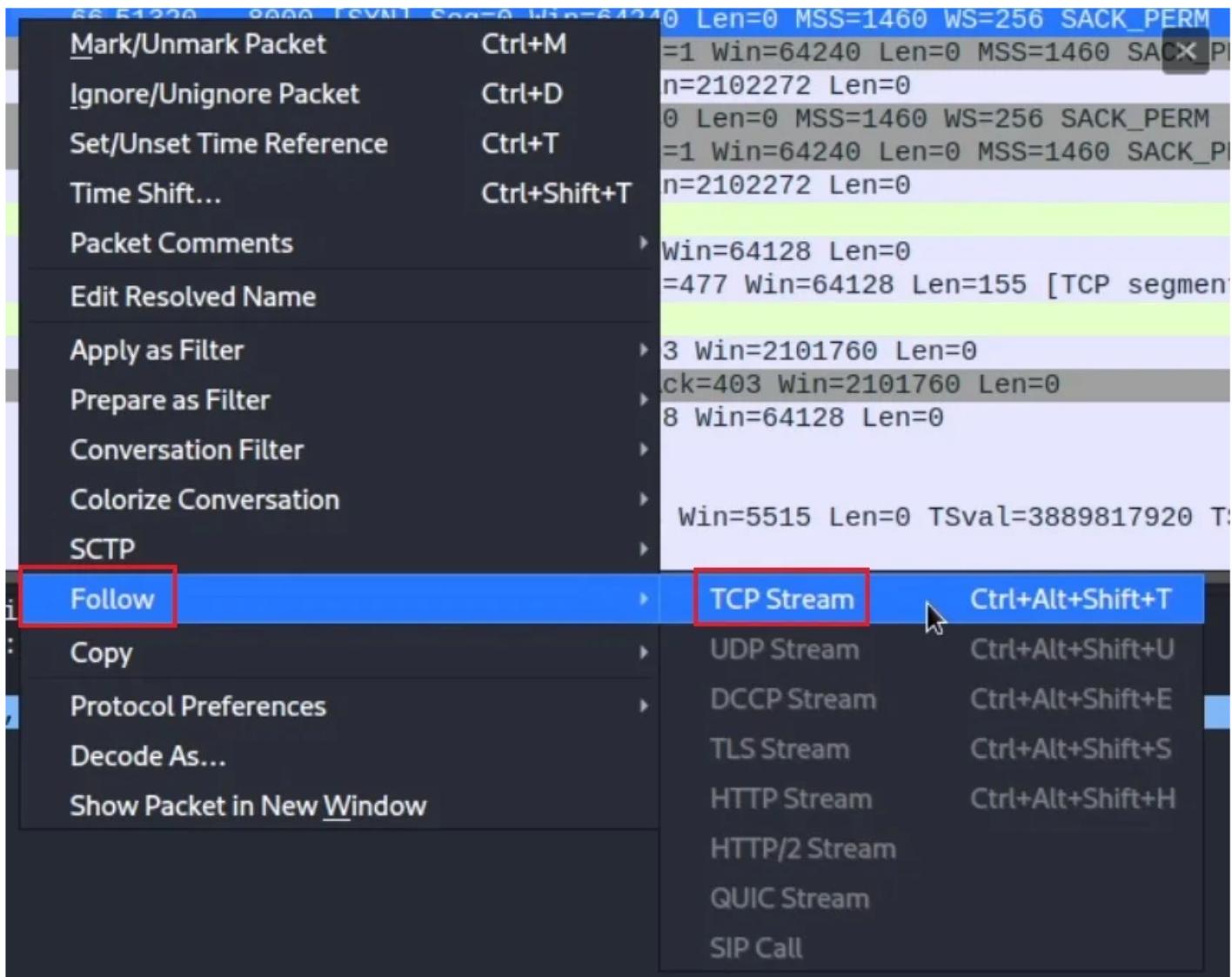
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

*eth0

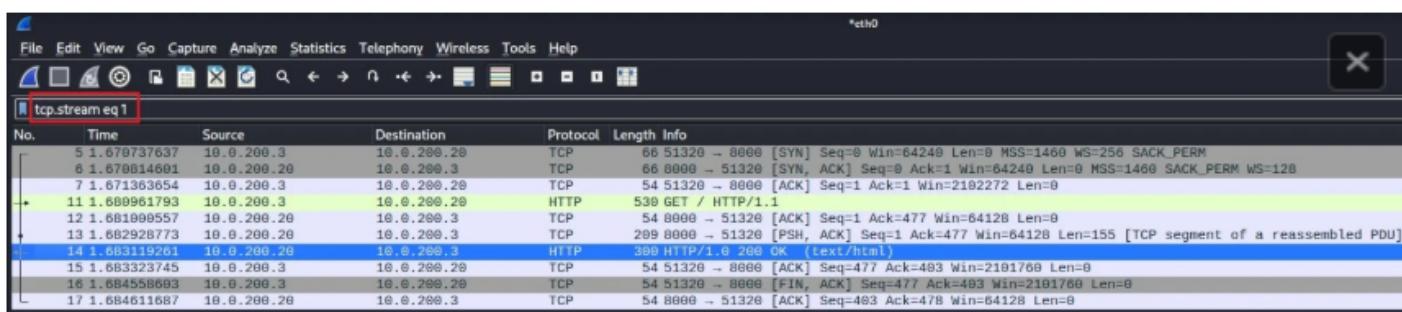
Current filter: icmp

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-------------|-------------|----------|--------|------------------------------------------------------------------------|
| 1 | 0.000000000 | 10.0.200.12 | 10.0.200.10 | TLSv1.2 | 1640 | Application Data |
| 2 | 0.000456868 | 10.0.200.10 | 10.0.200.12 | TCP | 66 | 9208 - 60008 [ACK] Seq=1 Ack=1575 Win=1780 Len=0 TSval=3889814348 TS |
| 3 | 0.006893708 | 10.0.200.10 | 10.0.200.12 | TLSv1.2 | 489 | Application Data |
| 4 | 0.008931362 | 10.0.200.12 | 10.0.200.10 | TCP | 66 | 60008 - 9208 [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 - 0000 [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 | 8008 - 51320 [SYN, ACK] Seq=1 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM |
| 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=64248 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_PERM |
| 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 | 538 | 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 | 1.684558693 | 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 | 3.571965530 | 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 | 9208 - 60008 [ACK] Seq=1 Ack=7933 Win=5515 Len=0 TSval=3889817920 TS |
| 21 | 3.5805372120 | 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.



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



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.

The screenshot shows a Wireshark window with a red border around the main content area. At the top, it says "Wireshark - Follow TCP Stream (tcp.stream eq 1) - eth0". The main pane displays a network conversation between a client and a server. The client's GET request includes headers for User-Agent (Mozilla/5.0), Accept (text/html, application/xhtml+xml, application/xml, image/webp, image/apng, */*, application/signed-exchange; v=b3; q=0.7), Accept-Encoding (gzip, deflate), and Accept-Language (en-GB, en; q=0.9, en-US; q=0.8). The server responds with an HTTP/1.0 200 OK status, providing a directory listing. The listing includes a single item: a link to "stationx-logo.png". The bottom of the window shows the status bar with "1 client pkt, 2 server pkts, 1 turn.", a dropdown for "Entire conversation (877 bytes)", a "Show data as ASCII" button, and a "Stream 1" button.

```
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

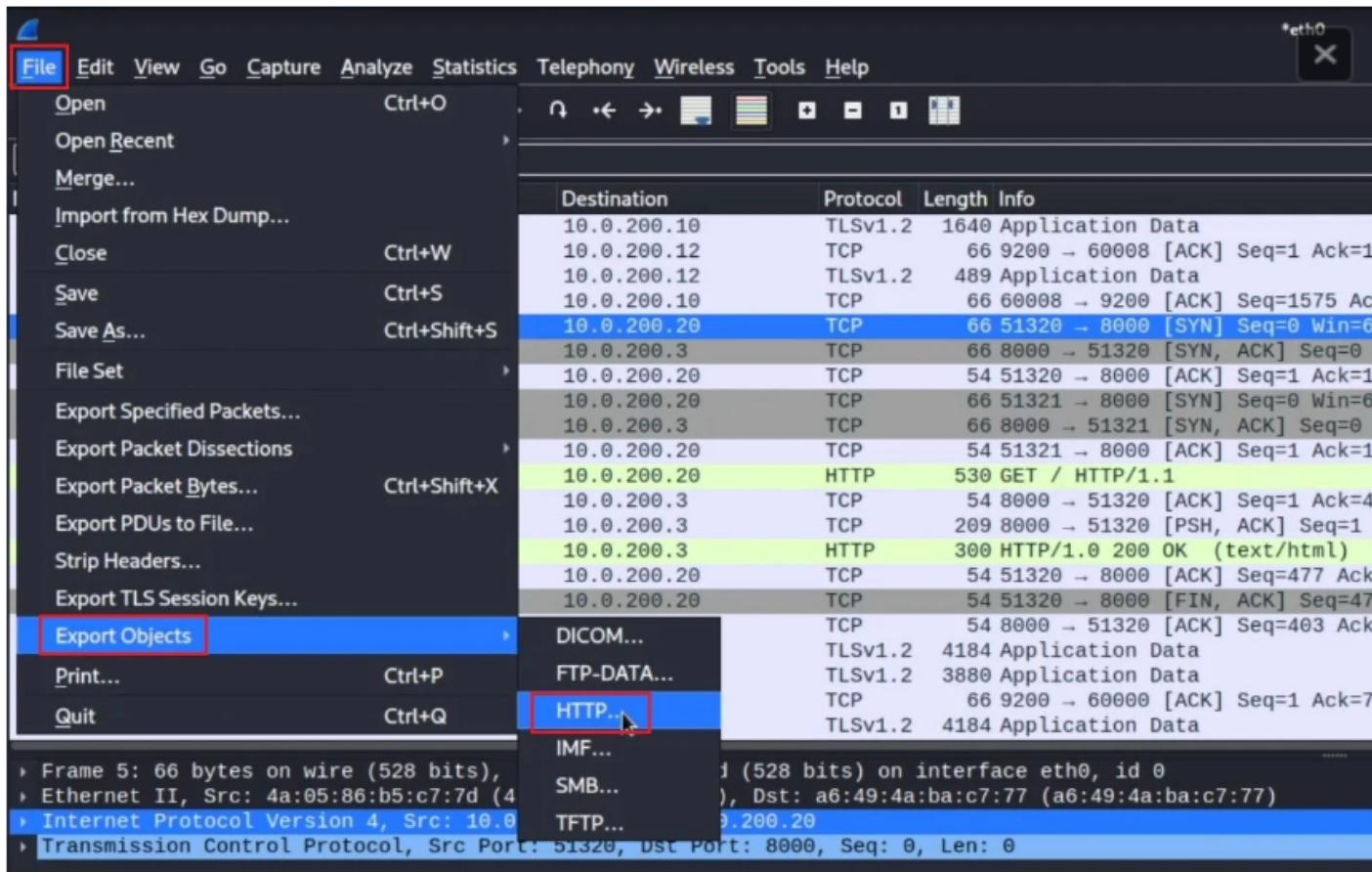
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="stationx-logo.png">stationx-logo.png</a></li>
</ul>
<hr>
</body>
</html>

1 client pkt, 2 server pkts, 1 turn.
```

Extracting Files

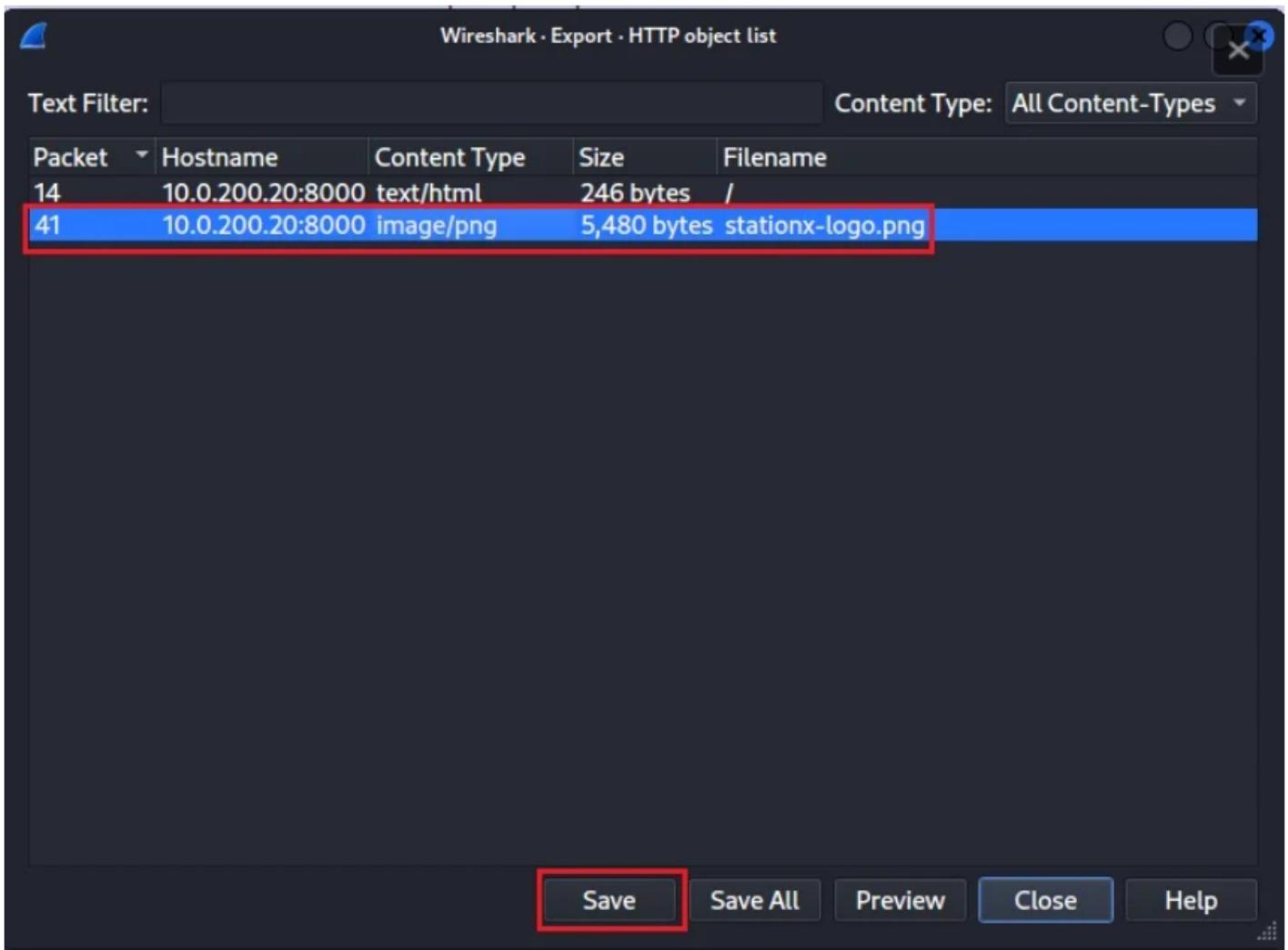
If you are on the red team, you often use Wireshark to extract sensitive information, such as usernames, passwords, and files. Wireshark can reconstruct files from the capture packets. This feature is incredibly useful as it allows you to extract various file types from network packets.

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.

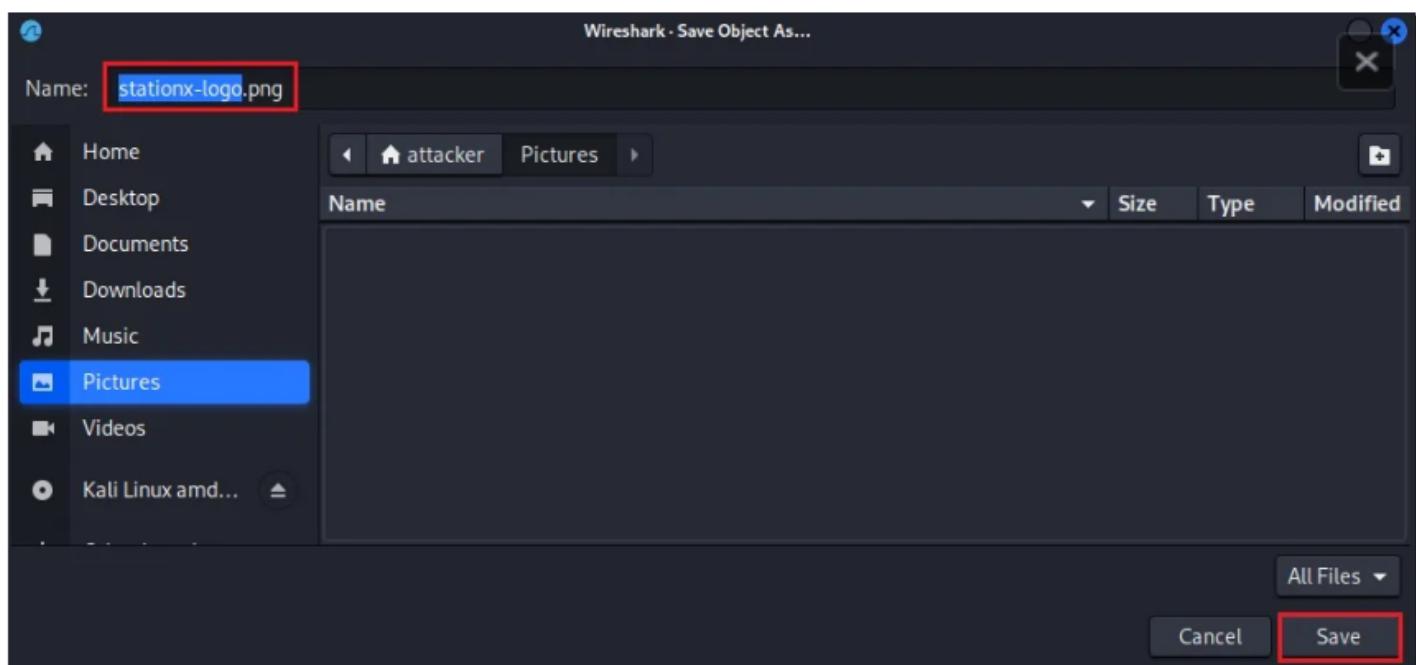


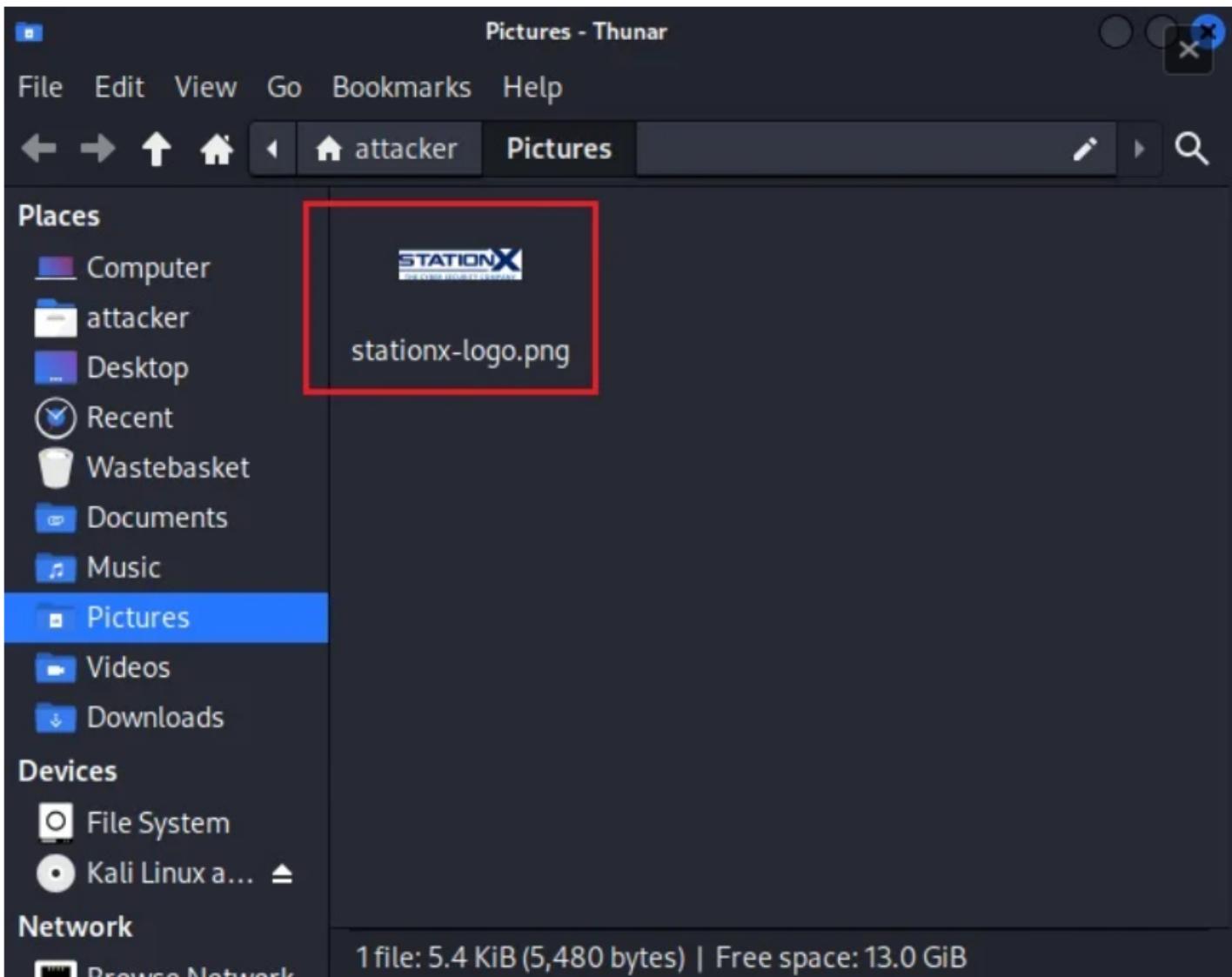
Selecting HTTP opens a new window populated by all the objects transferred using HTTP in the packet capture file. There are two objects; a static HTML page and a PNG image.

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



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





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.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

*eth0 X

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

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.

Wireshark - Follow TCP Stream (tcp.stream eq 7) - eth0

```

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.

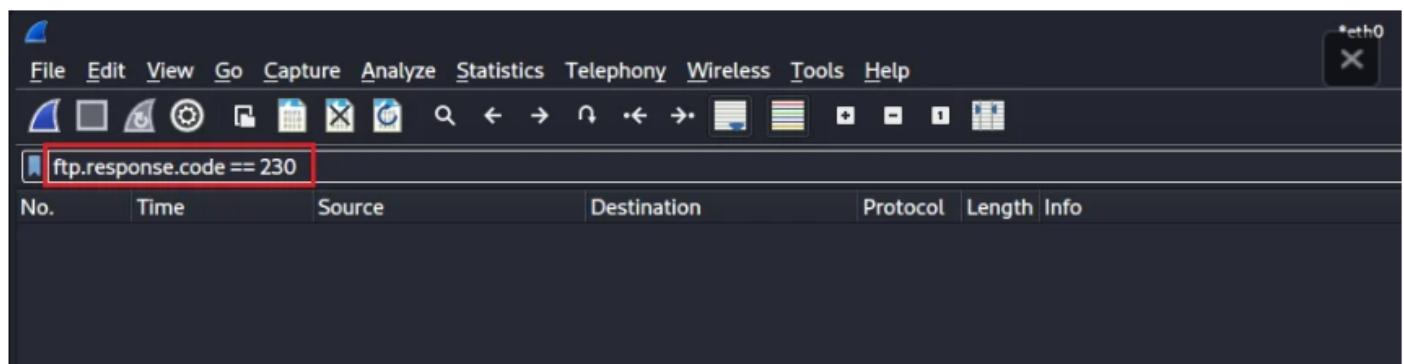
Entire conversation (108 bytes) Show data as ASCII Stream 7

Find: Find Next

Filter Out This Stream Print Save as... Back Close Help

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.”



No login attempts were successful!

Conclusion

Wireshark is a powerful open-source network protocol analyzer that turns bytes on the wire into network traffic you can analyze. Its simple-to-use interface provides an overview of your capture traffic in the list pane and specific information about each packet in the details pane.

You have seen how to examine network packets in granular detail, refine your view using Wireshark's display filters, and summarize network traffic with statistical analysis tools. These skills allow you to use Wireshark in the real world for following TCP streams to uncover conversations, extracting files, and identifying cyber attacks

Fun Part

You can find out what cyber security team you should join in the [Red Team vs Blue Team: Which Is the Best Choice for You?](#)