

Wireshark Advanced Usage

In this section, we will cover some advanced usage with Wireshark. The project developers have included many different capabilities ranging from tracking TCP conversations to cracking wireless credentials. The inclusion of many different plugins makes Wireshark one of the best traffic analysis tools.

Plugins

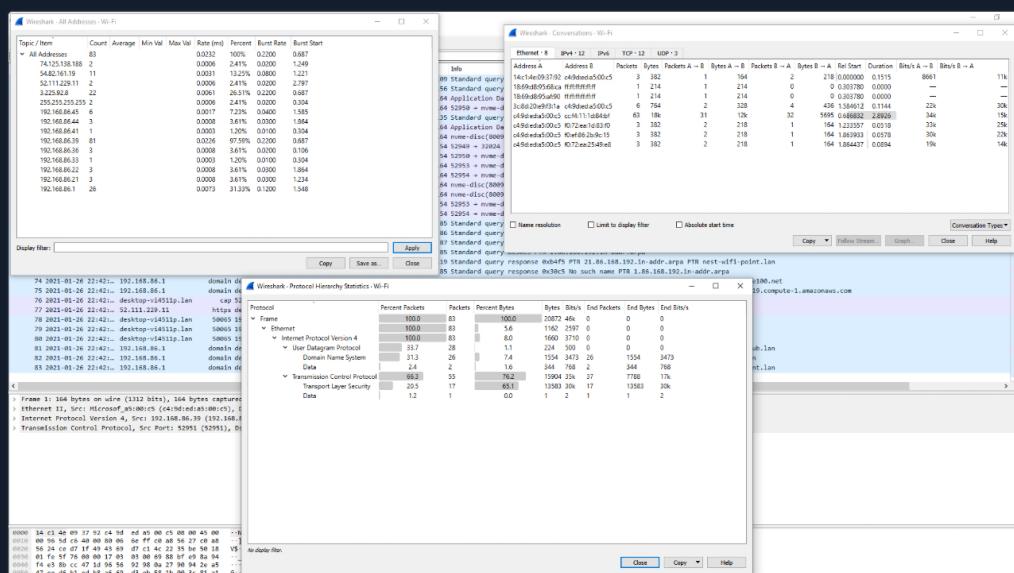
The analyze and statistics radials provide a plethora of plugins to run against the capture. In this section, we will work through a couple of them. We would cover all of which Wireshark offers, but sadly, it is simply not achievable in an introductory module. I urge everyone to experiment and play as we go through this journey.

The Statistics and Analyze Tabs

The Statistics and Analyze tabs can provide us with great insight into the data we are examining. From these points, we can utilize many of the baked-in plugins Wireshark has to offer.

The plugins here can give us detailed reports about the network traffic being utilized. It can show us everything from the top talkers in our environment to specific conversations and even breakdown by IP and protocol.

Statistics Tab



Analyze

From the Analyze tab, we can utilize plugins that allow us to do things such as following TCP streams, filter on conversation types, prepare new packet filters and examine the expert info Wireshark generates about the traffic. Below are a few examples of how to use these plugins.

Analyze Tab

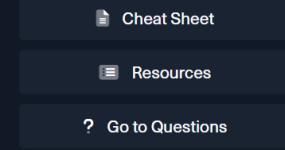
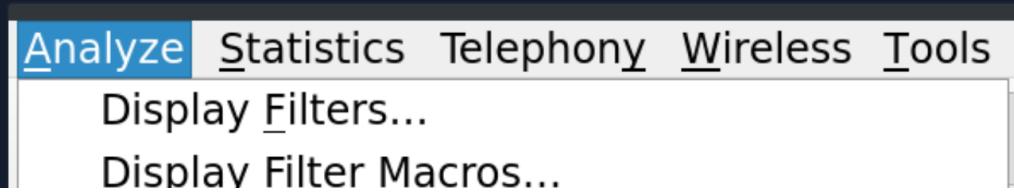


Table of Contents

Introduction

- Network Traffic Analysis
- Networking Primer - Layers 1-4
- Networking Primer - Layers 5-7

Analysis

- The Analysis Process
- Analysis in Practice

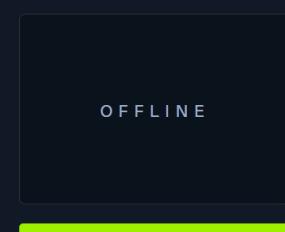
Tcpdump

- Tcpdump Fundamentals
- Capturing With Tcpdump (Fundamentals Labs)
- Tcpdump Packet Filtering
- Interrogating Network Traffic With Capture and Display Filters

Wireshark

- Analysis with Wireshark
- Familiarity With Wireshark
- Wireshark Advanced Usage
- Packet Inception, Dissecting Network Traffic With Wireshark
- Guided Lab: Traffic Analysis Workflow
- Decrypting RDP connections

My Workstation



Display Filter Expression...

Apply as Column

Ctrl+Shift+I

Apply as Filter

Prepare a Filter

Conversation Filter

Enabled Protocols...

Ctrl+Shift+E

Decode As...

Reload Lua Plugins

Ctrl+Shift+L

SCTP

Follow

Show Packet Bytes...

Ctrl+Shift+O

Expert Information

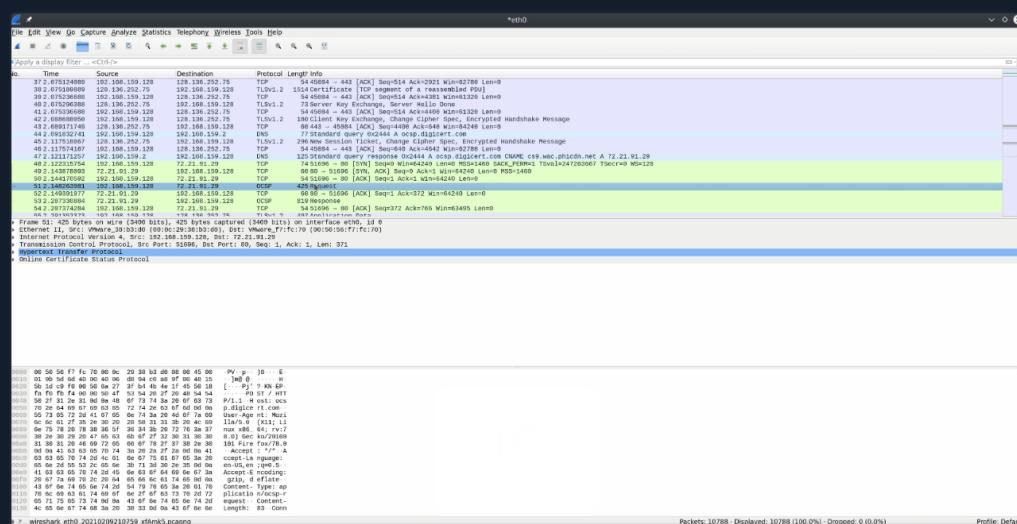
Following TCP Streams

Wireshark can stitch TCP packets back together to recreate the entire stream in a readable format. This ability also allows us to pull data ([images](#), [files](#), [etc.](#)) out of the capture. This works for almost any protocol that utilizes TCP as a transport mechanism.

To utilize this feature:

- right-click on a packet from the stream we wish to recreate.
- select follow → TCP
- this will open a new window with the stream stitched back together. From here, we can see the entire conversation.

Follow A Stream Via GUI



Alternatively, we can utilize the filter `tcp.stream eq #` to find and track conversations captured in the pcap file.

Filter For A Specific TCP Stream

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.100.18.5	10.100.16.1	TCP	62	3756 → 23 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
2	0.001244	10.100.16.1	10.100.18.5	TCP	60	23 → 3756 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460

3	0.001263	10.100.18.5	10.100.16.1	TCP	54 3756 → 23 [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	0.003143	10.100.16.1	10.100.18.5	TELNET	60 Telnet Data ...
5	0.003201	10.100.18.5	10.100.16.1	TELNET	57 Telnet Data ...
6	0.004161	10.100.16.1	10.100.18.5	TELNET	60 Telnet Data ...
7	0.150539	10.100.18.5	10.100.16.1	TCP	54 3756 → 23 [ACK] Seq=4 Ack=6 Win=65530 Len=0
8	0.151553	10.100.16.1	10.100.18.5	TELNET	64 Telnet Data ...
9	0.351722	10.100.18.5	10.100.16.1	TCP	54 3756 → 23 [ACK] Seq=4 Ack=16 Win=65520 Len=0
10	1.285886	10.100.18.5	10.100.16.1	TELNET	55 Telnet Data ...
11	1.287056	10.100.16.1	10.100.18.5	TCP	60 23 → 3756 [ACK] Seq=16 Ack=5 Win=8192 Len=0

Notice that the first three packets in the image above have a full TCP handshake. Following those packets, we can see the stream transferring data. We have cleared anything not related out of view by utilizing the filter, and we now can see the conversation in order.

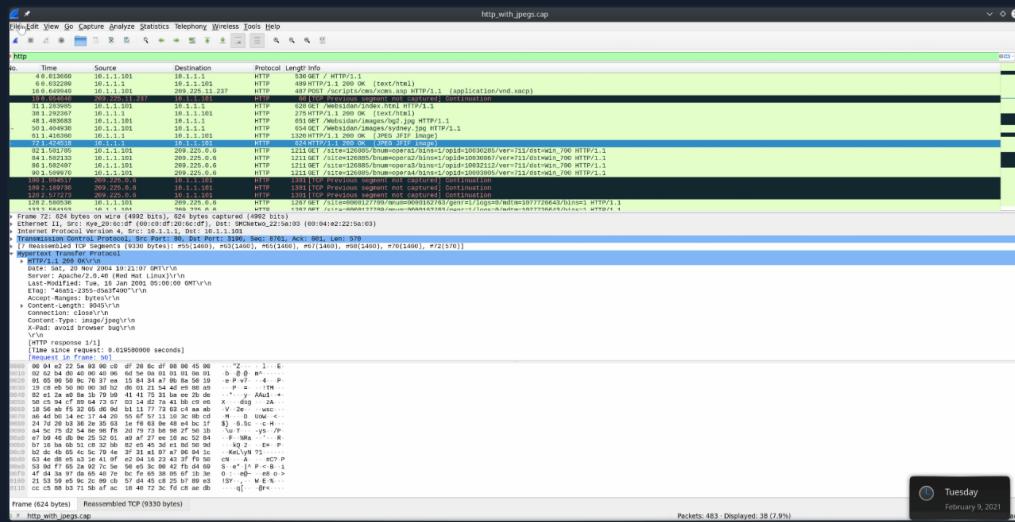
Extracting Data and Files From a Capture

Wireshark can recover many different types of data from streams. It requires you to have captured the entire conversation. Otherwise, this ability will fail to put an incomplete datagram back together. If we want a more in-depth understanding of how this capability works, check out the Networking 101 Module or research TCP/IP fragmentation.

To extract files from a stream:

- stop your capture.
- Select the File radial → Export → , then select the protocol format to extract from.
- (DICOM, HTTP, SMB, etc.)

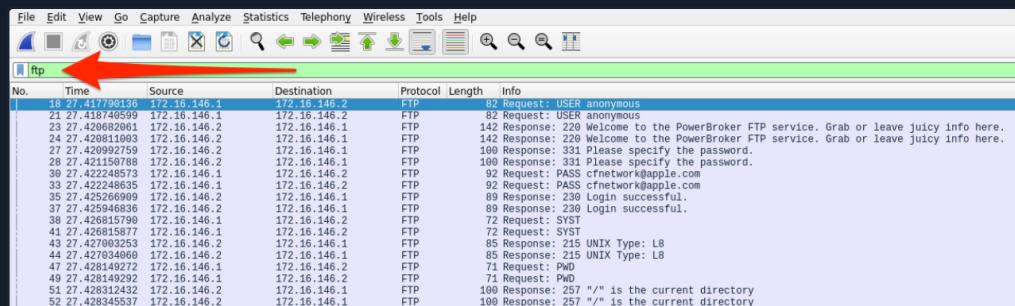
Extract Files From The GUI



Another exciting way to grab data out of the pcap file comes from FTP. The File Transfer Protocol moves data between a server and host to pull it out of the raw bytes and reconstruct the file. (image, text documents, etc.) FTP utilizes TCP as its transport protocol and uses ports **20 & 21** to function. TCP port 20 is used to transfer data between the server and host, while port 21 is used as the FTP control port. Any commands such as login, listing files, and issuing download or uploads happen over this port. To do so, we need to look at the different **FTP** display filters in Wireshark. A complete list of these can be found [here](#). For now, we will look at three:

- **ftp** - Will display anything about the FTP protocol.
- We can utilize this to get a feel for what hosts/servers are transferring data over FTP.

FTP Dissector



57 27.429456139	172.16.146.1	172.16.146.2	FTP	74 Request: TYPE I
59 27.430965808	172.16.146.1	172.16.146.1	FTP	97 Response: 200 Switching to Binary mode.
60 27.430965829	172.16.146.1	172.16.146.1	FTP	97 Response: 200 Switching to Binary mode.
63 27.430755408	172.16.146.1	172.16.146.2	FTP	73 Request: CWD /
65 27.430755428	172.16.146.1	172.16.146.2	FTP	73 Request: CWD /
66 27.430894435	172.16.146.2	172.16.146.1	FTP	103 Response: 250 Directory successfully changed.
67 27.430965742	172.16.146.2	172.16.146.1	FTP	103 Response: 250 Directory successfully changed.
70 27.430965727	172.16.146.1	172.16.146.2	FTP	92 Request: PORT 172.16.146.1,194,99
72 27.430965745	172.16.146.1	172.16.146.2	FTP	78 Request: PASV
73 27.432275691	172.16.146.2	172.16.146.1	FTP	117 Response: 200 PORT command successful. Consider using PASV.
74 27.432275691	172.16.146.2	172.16.146.1	FTP	445 Response: 200 PORT command successful. Consider using PASV.

Frame 18: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface eth0, id 0
Ethernet II, Src: VMare_97:52:65 (00:0c:29:97:52:65), Dst: 172.16.146.1 (0a:66:5a:11:8d:64)
Internet Protocol Version 4, Src: 172.16.146.1, Dst: 172.16.146.2
Transmission Control Protocol, Src Port: 49761, Dst Port: 21, Seq: 1, Ack: 1, Len: 16
File Transfer Protocol (FTP)
[Current working directory:]

0000 00 0c 29 97 52 65 8a 66 5a 11 8d 64 08 00 45 02 ..) Re-f
0010 00 44 00 00 00 00 40 06 fe 8d ac 19 92 01 ac 19 ..D ..@>
0020 92 02 c2 61 00 15 00 3e f8 70 52 86 8c 1e 00 18 ..a ..>
0030 08 0a bc e7 00 00 01 01 08 0a 69 f3 59 cc 19 8c ..
0040 2b 9c 55 53 45 52 20 61 66 6f 66 79 60 6f 75 73 +USER a
0050 0d 0a

- **ftp.request.command** - Will show any commands sent across the ftp-control channel (port 21)

- We can look for information like usernames and passwords with this filter. It can also show us filenames for anything requested.

FTP-Request-Command Filter

No.	Time	Source	Destination	Protocol	Length	Info
18 27.41799136	172.16.146.1	172.16.146.2	FTP	82 Request: USER anonymous		
21 27.418740599	172.16.146.1	172.16.146.2	FTP	82 Request: USER anonymous		
38 27.422485732	172.16.146.1	172.16.146.2	FTP	92 Request: PASS cfnetwork@apple.com		
39 27.426815790	172.16.146.1	172.16.146.2	FTP	72 Request: SYST		
38 27.426815790	172.16.146.1	172.16.146.2	FTP	72 Request: SYST		
41 27.426815877	172.16.146.1	172.16.146.2	FTP	72 Request: SYST		
47 27.428149272	172.16.146.1	172.16.146.2	FTP	71 Request: PWD		
49 27.428149272	172.16.146.1	172.16.146.2	FTP	71 Request: PWD		
53 27.428149272	172.16.146.1	172.16.146.2	FTP	71 Request: PWD		
57 27.429456139	172.16.146.1	172.16.146.2	FTP	74 Request: TYPE I		
63 27.430755408	172.16.146.1	172.16.146.2	FTP	73 Request: CWD /		
65 27.430755428	172.16.146.1	172.16.146.2	FTP	73 Request: CWD /		
70 27.430965742	172.16.146.1	172.16.146.2	FTP	92 Request: PORT 172.16.146.1,194,99		
72 27.430972745	172.16.146.1	172.16.146.2	FTP	72 Request: LIST		
76 27.433216926	172.16.146.1	172.16.146.2	FTP	72 Request: LIST		
88 27.434387518	172.16.146.1	172.16.146.2	FTP	72 Request: LIST		
96 27.434387518	172.16.146.1	172.16.146.2	FTP	82 Request: USER anonymous		
102 27.452173569	172.16.146.1	172.16.146.2	FTP	92 Request: PASS cfnetwork@apple.com		
106 27.452173732	172.16.146.1	172.16.146.2	FTP	72 Request: SYST		
118 27.454089169	172.16.146.1	172.16.146.2	FTP	71 Request: PWD		
114 27.459047994	172.16.146.1	172.16.146.2	FTP	74 Request: TYPE I		
118 27.459047994	172.16.146.1	172.16.146.2	FTP	73 Request: CWD /		
122 27.459512151	172.16.146.1	172.16.146.2	FTP	72 Request: PASV		
125 27.459819188	172.16.146.1	172.16.146.2	FTP	72 Request: LIST		
145 27.646844533	172.16.146.1	172.16.146.2	FTP	82 Request: USER anonymous		
151 27.650087045	172.16.146.1	172.16.146.2	FTP	92 Request: PASS cfnetwork@apple.com		
155 27.654571242	172.16.146.1	172.16.146.2	FTP	72 Request: SYST		
159 27.659875950	172.16.146.1	172.16.146.2	FTP	71 Request: PWD		
163 27.657518680	172.16.146.1	172.16.146.2	FTP	74 Request: TYPE I		
118 27.41799136	172.16.146.1	172.16.146.2	FTP	74 Request: TYPE I		
200 00 0c 29 97 52 65 8a 66 5a 11 8d 64 08 00 45 02 ..) Re-f						
201 00 44 00 00 00 00 40 06 fe 8d ac 19 92 01 ac 19 ..D ..@>						
202 92 02 c2 61 00 15 00 3e f8 70 52 86 8c 1e 00 18 ..a ..>						
203 08 0a bc e7 00 00 01 01 08 0a 69 f3 59 cc 19 8c ..						
204 2b 9c 55 53 45 52 20 61 66 6f 66 79 60 6f 75 73 +USER a						
205 0d 0a						

Frame 18: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface eth0, id 0
Ethernet II, Src: VMare_97:52:65 (00:0c:29:97:52:65), Dst: 172.16.146.1 (0a:66:5a:11:8d:64)
Internet Protocol Version 4, Src: 172.16.146.1, Dst: 172.16.146.2
Transmission Control Protocol, Src Port: 49761, Dst Port: 21, Seq: 1, Ack: 1, Len: 16
File Transfer Protocol (FTP)
[Current working directory:]

0000 00 0c 29 97 52 65 8a 66 5a 11 8d 64 08 00 45 02 ..) Re-f
0010 00 44 00 00 00 00 40 06 fe 8d ac 19 92 01 ac 19 ..D ..@>
0020 92 02 c2 61 00 15 00 3e f8 70 52 86 8c 1e 00 18 ..a ..>
0030 08 0a bc e7 00 00 01 01 08 0a 69 f3 59 cc 19 8c ..
0040 2b 9c 55 53 45 52 20 61 66 6f 66 79 60 6f 75 73 +USER a
0050 0d 0a

- **ftp-data** - Will show any data transferred over the data channel (port 20)

- If we filter on a conversation and utilize **ftp-data**, we can capture anything sent during the conversation. We can reconstruct anything transferred by placing the raw data back into a new file and naming it appropriately.

FTP-Data Filter

No.	Time	Source	Destination	Protocol	Length	Info
85 27.435986082	172.16.146.2	172.16.146.1	FTP-DA-	235	FTP Data: 149 bytes (PASV) (LIST)	
130 27.462893749	172.16.146.2	172.16.146.1	FTP-DA-	235	FTP Data: 149 bytes (PASV) (LIST)	
181 27.663333916	172.16.146.2	172.16.146.1	FTP-DA-	112	FTP Data: 46 bytes (PASV) (RETR secrets.txt)	
241 37.399266312	172.16.146.2	172.16.146.1	FTP-DA-	138	FTP Data: 72 bytes (PASV) (RETR Shield-prototype-plans)	

Frame 181: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface eth0, id 0
Ethernet II, Src: VMare_97:52:65 (00:0c:29:97:52:65), Dst: 172.16.146.1 (0a:66:5a:11:8d:64)
Internet Protocol Version 4, Src: 172.16.146.2, Dst: 172.16.146.1
Transmission Control Protocol, Src Port: 53091, Dst Port: 49768, Seq: 1, Ack: 1, Len: 46
FTP Data (46 bytes)
File name: secrets.txt

```

[Setup method: PASV]
[Command: RETR secrets.txt]
[Command frame: 175]
[Current working directory: /]
Line-based text data (1 lines)
The Winter Soldier's arm has a reset button..\n

0000  8a 66 5a 11 8d 64 00 0c  29 97 52 65 08 00 45 0a  -fZ--d-
0010  00 62 7c cf 40 00 40 06  41 98 ac 10 92 02 ac 10  -b| @ @-
0020  92 01 cf 63 c2 68 f0 b7  1b b1 16 34 ba ec 88 18  --c h--
0030  01 fe 7c 79 00 00 01 01  08 04 19 8c 2c 93 40 d9  --ly--
0040  18 c6 54 68 65 20 57 69  6e 74 65 72 29 53 6f 6c  -The Wi nter
0050  64 69 65 72 27 73 20 61  72 6d 20 68 61 73 20 61  dier's a rm has
0060  20 72 65 73 65 74 20 62  75 74 74 6f 6e 2e 2e 0a  reset b

```

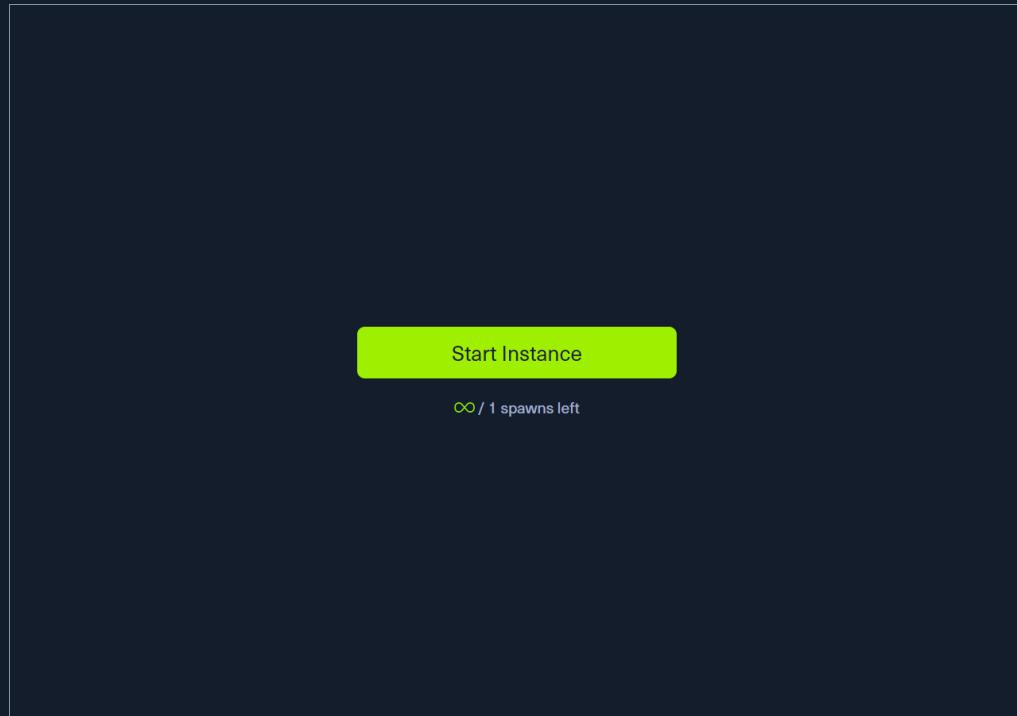
Since FTP utilizes TCP as its transport mechanism, we can utilize the `follow tcp stream` function we utilized earlier in the section to group any conversation we wish to explore. The basic steps to dissect FTP data from a pcap are as follows:

1. Identify any FTP traffic using the `ftp` display filter.
2. Look at the command controls sent between the server and hosts to determine if anything was transferred and who did so with the `ftp.request.command` filter.
3. Choose a file, then filter for `ftp-data`. Select a packet that corresponds with our file of interest and follow the TCP stream that correlates to it.
4. Once done, Change "Show and save data as" to "Raw" and save the content as the original file name.
5. Validate the extraction by checking the file type.


Connect to Pwnbox
 Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location: UK
 137ms

ⓘ Terminate Pwnbox to switch location



Waiting to start...

Questions

Answer the question(s) below to complete this Section and earn cubes!



Cheat Sheet

- + 0 📁 Which plugin tab can provide us with a way to view conversation metadata and even protocol breakdowns for the entire PCAP file?

Statistics



Submit



Hint

- + 0 📁 What plugin tab will allow me to accomplish tasks such as applying filters, following streams, and viewing expert info?

Analyze



Submit



Hint

- + 0 📁 What stream oriented Transport protocol enables us to follow and rebuild conversations and the included data?

TCP



Submit



Hint

- + 0 📁 True or False: Wireshark can extract files from HTTP traffic.

True



Submit



Hint

- + 0 📁 True or False: The ftp-data filter will show us any data sent over TCP port 21.

False



Submit



Hint

◀ Previous

Next ➔

Mark Complete & Next

