

Wireshark 101 Labs

Portfolio Samples

Will Kittredge

TABLE OF CONTENTS

Lab 2: Capture and Classify Your Own Background Traffic	4
Lab 3: Open a Network Capture File	10
Chapter 0 Challenge.....	10
Lab 4: Add HTTP Host Field as a Column.....	13
Lab 5: Set Key Wireshark Preferences	13
Lab 6: Create new Profile Based on the Default Profile.....	14
Lab 7: Import a DNS/HTTP Errors Profile.....	14
Lab 8: Spot Path and Server Latency Problems.....	15
Chapter 1 Challenge.....	15
Lab 9: Capture to File Sets.....	18
Lab 10: Use Ring Buffer to Conserve Drive Space.....	18
Lab 11: Capture only Traffic to or from your IP Address	18
Lab 13: Create, Save, and Apply a DNS Capture Filter	19
Chapter 2 Challenge.....	19
Lab 14: Use Auto-Complete to Find Traffic to a Specific HTTP Server	23
Lab 16: Filter on HTTP Traffic the Right Way	23
Lab 17: Filter on Traffic to or from Online Backup Subnets	24
Lab 18: Filter on DNS Name Errors or HTTP 404 Responses.....	24
Lab 19: Detect background File Transfers on Startup	24
Lab 20: Locate TCP Connection Attempts to a Client	25
Lab 21: Filter to Locate a Set of Key Words in a Trace File.....	25
Lab 22: Filter with Wildcards between Words	25
Lab 23: Import Display Filters into a Profile.....	25
Lab 24: Create and Import HTTP Filer Exp Buttons.....	26
Chapter 3 Challenge.....	26
Lab 25: Add a Column to Display Coloring Rules in Use	29
Lab 26: Build a Coloring Rule.....	29
Lab 27: Create Temporary Conversation Coloring Rules	29
Lab 28: Use the Intelligent Scrollbar to Quickly Find Problems.....	30

Lab 29: Export a Single TCP Conversation.....	30
Lab 30: Export a List of HTTP Host Field Values from a Trace File	30
Chapter 4 Challenge.....	31
Lab 31: Filter on Most Active TCP Conversation.....	33
Lab 32: Set up GeoIP to Map Targets Globally	33
Lab 33: Detect Suspicious Protocols or Applications.....	34
Lab 34: Compare Traffic to/from a Subnet to Other Traffic	35
Lab 35: Identify an Overloaded Client	35
Lab 36: Detect and Graph File Transfer Problems	36
Chapter 5 Challenge.....	36
Lab 37: Use Reassembly to Find a Web Site's Hidden HTTP Message	39
Lab 38: Extract a File from an FTP Transfer.....	40
Lab 39: Carve Out a HTTP Object from a Web Browsing Session	41
Chapter 6 Challenge.....	41
Lab 40: Read Analysis Notes in a Malicious Redirection Trace File.....	43
Lab 41: Export Malicious Redirection Packet Comments.....	44
Chapter 7 Challenge.....	44
Lab 42: Split a File and Work with Filtered File Sets	46
Lab 43: Merge a Set of Files using a Wildcard	46
Lab 44: Use Tshark to Capture to File Sets with an Autostop Condition.....	46
Lab 45: Use Tshark to Extract HTTP GET Requests	47
Lab 46: Use Tshark to Extract HTTP Host Names and IP Addresses	47
Chapter 8 Challenge.....	48
References.....	49

LAB 2: CAPTURE AND CLASSIFY YOUR OWN BACKGROUND TRAFFIC

i. Insert a screenshot of active interfaces and answer the following:

Interface	Traffic	Link-layer Header	Promis	Snaplen	Buffer (M	Monit	Capture Filter
Local Area Connection* 9		Ethernet	<input checked="" type="checkbox"/>	default	2	—	
Local Area Connection* 8		Ethernet	<input checked="" type="checkbox"/>	default	2	—	
Local Area Connection* 7		Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> vEthernet (WSL (Hyper-V firewall))	—	Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> vEthernet (Default Switch)	—	Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> Bluetooth Network Connection 0		Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> VMware Network Adapter VMnet8	—	Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> VMware Network Adapter VMnet1	—	Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> Local Area Connection* 12		Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> Local Area Connection* 11		Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> Wi-Fi 0		Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> Ethernet 0	~~~~~	Ethernet	<input checked="" type="checkbox"/>	default	2	—	
> Adapter for loopback traffic capture	~~~~~	BSD loopback	<input checked="" type="checkbox"/>	default	2	—	
> Ethernet 1		Ethernet	<input checked="" type="checkbox"/>	default	2	—	

a. How many different types of interfaces do you have? Explain what they all are.

There are 14 listed interfaces. Ethernet 0 and 1 are physical interfaces that are built into my motherboard (the board also has built-in Wi-Fi and Bluetooth interfaces, but these are disabled in BIOS due to some problems I'm working through with Linux dual booting). The listed Wi-Fi 0 and Bluetooth 0 interfaces are also physical. I have a PCIe Wi-Fi/Bluetooth adapter with firmware that plays nice with Linux slotted into my motherboard. I believe that the rest of the interfaces are virtual. The loopback interface is for local communication (the device communicating with itself). Both VMware interfaces were created by VMware Workstation and are used for VM networking (VMware, 2019). Similarly, the vEthernet interfaces are used by Hyper-V – one appears to be for my Ubuntu WSL environment (Harwood et al., 2021). I honestly wasn't sure about the Local Area Connection interfaces, and I'm still not completely satisfied with the answer I found. However, I believe that these are also virtual interfaces. According to a StackExchange answer, Windows automatically creates these for its own purposes and sets them as hidden (Mittal, 2014).

b. Provide a screenshot of one interface's detailed characteristics.

▼ Ethernet 0	Ethernet	checkbox	default	2	—
Addresses: 192.168.1.100, fe80::ff45:f9e0:ae95:f293, fdfe:7474:5607:c4a:98ee:a7f:d2af:cd32, fdfe:7474:5607:c4a:c3b9:7fb0:2f40:cdb2					

I was unable to locate any additional views that show more detailed information about a specific interface in Wireshark. Hovering the mouse cursor over an interface or expanding the arrow does show associated addresses though.

ii. Insert screenshots that show:

a. Only DNS (port 53) traffic filtered.

udp.port == 53 tcp.port == 53						
No.	Time	Source	Destination	Protocol	Length	Info
628	194.713312	192.168.1.100	192.168.1.1	DNS	91	Standard query 0xf8c9 A settings-win.da
629	194.713373	192.168.1.100	192.168.1.1	DNS	91	Standard query 0xc7aa AAAA settings-wi
630	194.729717	192.168.1.100	192.168.1.1	DNS	91	Standard query 0xc7aa AAAA settings-wi
631	194.729725	192.168.1.100	192.168.1.1	DNS	91	Standard query 0xf8c9 A settings-win.da
632	194.742597	192.168.1.1	192.168.1.100	DNS	222	Standard query response 0xf8c9 A setti
633	194.742727	192.168.1.1	192.168.1.100	DNS	273	Standard query response 0xc7aa AAAA se

b. A DNS packet that initiated a DNS request.

udp.port == 53 tcp.port == 53						
No.	Time	Source	Destination	Protocol	Length	Info
628	194.713312	192.168.1.100	192.168.1.1	DNS	91	Standard query 0xf8c9 A settings-win.data.microsoft.com
▶ Frame 628: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface \Device\NPF_{E03B19B5-CDB4-410E-B9B9-764429E76425}, id 0						
▶ Ethernet II, Src: ASUSTekC_54:bf:d4 (a8:5e:45:54:bf:d4), Dst: Netgear_a4:de:68 (14:59:c0:a4:de:68)						
▶ Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.1						
▶ User Datagram Protocol, Src Port: 56983, Dst Port: 53						
▶ Domain Name System (query)						
Transaction ID: 0xf8c9						
Flags: 0x0100 Standard query						
Questions: 1						
Answer RRs: 0						
Authority RRs: 0						
Additional RRs: 0						
▶ Queries						
▶ settings-win.data.microsoft.com: type A, class IN						
Name: settings-win.data.microsoft.com						
[Name Length: 31]						
[Label Count: 4]						
Type: A (Host Address) (1)						
Class: IN (0x0001)						
[Response In: 632]						

iii. Insert a screenshot that shows only HTTP (port 80) traffic filtered.

http http2 http3						
No.	Time	Source	Destination	Protocol	Length	Info
706	11.467490	192.168.1.100	192.229.211.108	OCSP	515	Request
719	11.492318	192.229.211.108	192.168.1.100	OCSP	599	Response
969	11.648678	192.168.1.100	192.229.211.108	OCSP	515	Request
1247	11.677297	192.229.211.108	192.168.1.100	OCSP	599	Response
5346	12.356671	192.168.1.100	13.227.38.91	OCSP	524	Request
5927	12.458793	13.227.38.91	192.168.1.100	OCSP	998	Response

> Frame 706: 515 bytes on wire (4120 bits), 515 bytes captured (4120 bits) on interface
> Ethernet II, Src: ASUSTekC_54:bf:d4 (a8:5e:45:54:bf:d4), Dst: Netgear_a4:de:68 (14:59:0:
> Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.229.211.108
> Transmission Control Protocol, Src Port: 50276, Dst Port: 80, Seq: 1, Ack: 1, Len: 461
> Hypertext Transfer Protocol
> Online Certificate Status Protocol

My original 5-minute background capture did not include any HTTP traffic, I started a new capture and browsed to a webpage to generate the traffic in the screenshot above.

iv. Insert a screenshot that shows the source and destination port of a packet and answer the following:

udp.port == 53 tcp.port == 53			
No.	Time	Source	Destination
629	194.713373	192.168.1.100	192.168.1.1
> Frame 629: 91 bytes on wire (728 bits), 91 bytes captured			
> Ethernet II, Src: ASUSTekC_54:bf:d4 (a8:5e:45:54:bf:d4), Dst: 00:0c:29:14:00:00 (laptop) [eth0]			
> Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.1 [eth0]			
> User Datagram Protocol, Src Port: 64835, Dst Port: 53			
Source Port: 64835			
Destination Port: 53			
Length: 57			
Checksum: 0x8400 [unverified]			
[Checksum Status: Unverified]			
[Stream index: 28]			
[Timestamps]			
UDP payload (49 bytes)			
> Domain Name System (query)			

a. Can you tell the direction of a packet by the source and destination port? How?

I think that it is *possible* to determine the direction of a packet based on its source and destination port because of reserved and ephemeral ports, but I'm not certain if it is *always possible* (I'm leaning towards no). If a packet's source port is some well-known port number like 80 or 443 and the destination port is some larger port number in the ephemeral range, it stands to reason that the packet is likely outbound from a web server/inbound to a client. The opposite (outbound from a client/inbound to a web server) would likely be true in this example when the source and destination ports are swapped (Rosenberg, 2003; Johnson, 2019). It seems like this logic could break down if the client forces a well-known port number to be used instead of an ephemeral port number (is this possible? I don't see why not), or potentially in some other edge cases that are unknown to me.

v. Insert an example screenshot and answer the following:

> Frame 495: 491 bytes on wire (3928 bits), 491 bytes captured (3928 bits) on interface
> Ethernet II, Src: ASUSTekC_54:bf:d4 (a8:5e:45:54:bf:d4), Dst: Netgear_a4:de:68 (14:59:)
> Internet Protocol Version 4, Src: 192.168.1.100, Dst: 131.94.130.45
> Transmission Control Protocol, Src Port: 56840, Dst Port: 80, Seq: 1, Ack: 1, Len: 437
> Hypertext Transfer Protocol
0000 14 59 c0 a4 de 68 a8 5e 45 54 bf d4 08 00 45 00 .Y...h.^ ET...E.
0010 01 dd b8 02 40 00 80 06 00 00 c0 a8 01 64 83 5e@....d.^
0020 82 2d de 08 00 50 59 37 5a c3 a5 8f d2 1f 50 18 ..PY7 Z....P.
0030 04 02 c9 67 00 00 47 45 54 20 2f 7e 65 73 6a 2f ...g..GE T /~esj/
0040 63 67 73 34 32 38 35 2f 63 6c 61 73 73 31 33 2e cgs4285/ class13.
0050 68 74 6d 6c 20 48 54 54 50 2f 31 2e 31 0d 0a 48 html HTT P/1.1..H
0060 6f 73 74 3a 20 75 73 65 72 73 2e 63 73 2e 66 69 ost: use rs.cs.fi
0070 75 2e 65 64 75 0d 0a 55 73 65 72 2d 41 67 65 6e u.edu..U ser-Agen
0080 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 t: Mozil la/5.0 (
0090 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e 30 3b Windows NT 10.0;
00a0 20 57 69 6e 36 34 3b 20 78 36 34 3b 20 72 76 3a Win64; x64; rv:
00b0 31 33 30 2e 30 29 20 47 65 63 6b 6f 2f 32 30 31 130.0) G ecko/201
00c0 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 31 33 00101 Fi refox/13
00d0 30 2e 30 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 0.0..Acc ept: tex
00e0 74 2f 68 74 6d 6c 2c 61 70 70 6c 69 63 61 74 69 t/html,applicati
00f0 6f 6e 2f 78 68 74 6d 6c 2b 78 6d 6c 2c 61 70 70 on/xhtml+xml,app
0100 6c 69 63 61 74 69 6f 6e 2f 78 6d 6c 3b 71 3d 30 lication /xml;q=0
0110 2e 39 2c 69 6d 61 67 65 2f 61 76 69 66 2c 69 6d .9,image /avif,image
0120 61 67 65 2f 77 65 62 70 2c 69 6d 61 67 65 2f 70 /webp,image/p
0130 6e 67 2c 69 6d 61 67 65 2f 73 76 67 2b 78 6d 6c ng,image /svg+xml
0140 2c 2a 2f 2a 3b 71 3d 30 2e 38 0d 0a 41 63 63 65 ,/*;q=0 .8..Acc
0150 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e 2d pt-Langu age: en-
0160 55 53 2c 65 6e 3b 71 3d 30 2e 35 0d 0a 41 63 63 US,en;q=0.5..Acc
0170 65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a ept-Encoding: gz
0180 69 70 2c 20 64 65 66 6c 61 74 65 0d 0a 44 4e 54 ip, deflate,DNT
0190 3a 20 31 0d 0a 53 65 63 2d 47 50 43 3a 20 31 0d : 1..Sec -GPC: 1..
01a0 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 Connect ion: kee
01b0 70 2d 61 6c 69 76 65 0d 0a 55 70 67 72 61 64 65 p-alive..Upgrade
01c0 2d 49 6e 73 65 63 75 72 65 2d 52 65 71 75 65 73 -Insecu re-Request
01d0 74 73 3a 20 31 0d 0a 50 72 69 6f 72 69 74 79 3a ts: 1..Priority: u=0, i... .
01e0 20 75 3d 30 2c 20 69 0d 0a 0d 0a

a. Where does the actual “packet” PDU begin and end in a trace file?

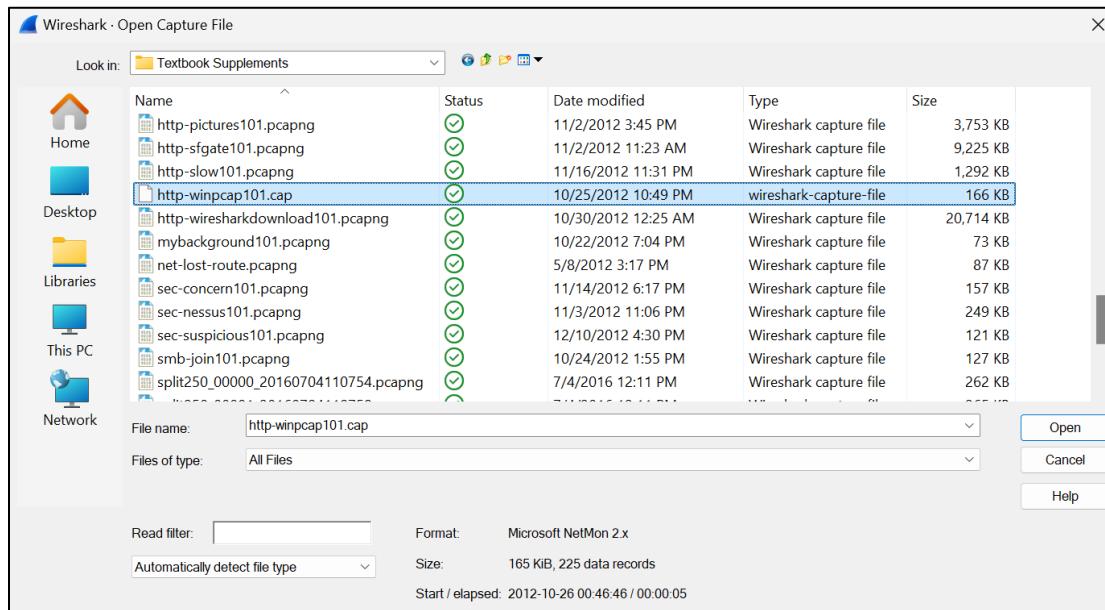
I was uncertain if this question was referring to how packets are actually stored in the .pcapng file format, or if it meant where the “data” actually begins/ends on the individual packet level as seen in Wireshark. In the context of part B of this question, I interpreted it as the latter. The “packet” PDU label is used loosely, but technically refers to protocol data units at the network layer as we’ve learned. Each “layer” adds a header with the required/necessary information. In the screenshot above, the IPv4 header is selected; this marks the beginning of the layer 3 packet, which itself contains a layer 4 segment/datagram.

b. Explain the difference between the PDUs of packet, frame, and a segment.

As mentioned in part A, “frame,” “packet,” and “segment” (or “datagram”) technically refer to PDUs at layers 2, 3, and 4 respectively. A frame is the PDU at the data link layer. A frame contains a packet, which itself contains a segment/datagram. A packet is the PDU at the network layer and contains a segment/datagram, and so on.

LAB 3: OPEN A NETWORK CAPTURE FILE

i. Provide a screenshot of completed step 2.

**CHAPTER 0 CHALLENGE**

0-1. How many packets are in this trace file?

Packets: 20 · Displayed: 20 (100.0%)

According to the information displayed at the bottom of the screen in Wireshark, there are 20 packets in the trace file.

0-2. What IP hosts are making a TCP connection in frames 1, 2, and 3?

No.	Time	Source	Destination	Protocol
1	0.000000	192.168.1.108	50.19.229.205	TCP
2	0.092419	50.19.229.205	192.168.1.108	TCP
3	0.092521	192.168.1.108	50.19.229.205	TCP

In frames 1, 2, and 3, IP hosts **192.168.1.108** and **50.19.229.205** are making a TCP connection.

0-3. What HTTP command is sent in frame 4?

```

> Frame 4: 1384 bytes on wire (11072 bits), 1384 bytes captured (11072 bits) on interface unknown, id 0
> Ethernet II, Src: HonHaiPr_68:74:f6 (90:4c:e5:68:74:f6), Dst: Cisco-Li_d9:94:c0 (00:1d:7e:d9:94:c0)
> Internet Protocol Version 4, Src: 192.168.1.108, Dst: 50.19.229.205
> Transmission Control Protocol, Src Port: 60139, Dst Port: 80, Seq: 1, Ack: 1, Len: 1330
< Hypertext Transfer Protocol
  > GET /Tracking/V3/Instream/Impression/?start|2873|72147|75904|9028|26105|undefined|1338|3379|807|BBE

```

In frame 4, the HTTP command is **GET**.

0-4. What is the length of the largest frame in this trace file?

No.	Time	Source	Destination	Protocol	Length
15	11.175009	192.168.1.108	50.19.229.205	HTTP	1428
12	3.675382	192.168.1.108	50.19.229.205	HTTP	1428
4	0.094027	192.168.1.108	50.19.229.205	HTTP	1384

Sorting the “Length” column from largest to smallest shows two frames tied for largest at 1,428 bytes.

0-5. What protocols are seen in the Protocol column?

No.	Time	Source	Destination	Protocol
1	0.000000	192.168.1.108	50.19.229.205	TCP
2	0.092419	50.19.229.205	192.168.1.108	TCP
3	0.092521	192.168.1.108	50.19.229.205	TCP
4	0.094027	192.168.1.108	50.19.229.205	HTTP

The only protocols seen in this file are TCP and HTTP.

0-6. What responses are sent by the HTTP server?

ip.dst == 192.168.1.108						
No.	Time	Source	Destination	Protocol	Length	Info
2	0.092419	50.19.229.205	192.168.1.108	TCP	66	80 → 60139 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
5	0.196174	50.19.229.205	192.168.1.108	TCP	54	80 → 60139 [ACK] Seq=1 Ack=1331 Win=8960 Len=0
6	0.204299	50.19.229.205	192.168.1.108	HTTP	607	HTTP/1.1 302 Found
8	0.326624	50.19.229.205	192.168.1.108	HTTP	607	HTTP/1.1 302 Found
10	0.674498	50.19.229.205	192.168.1.108	HTTP	607	[TCP Spurious Retransmission] HTTP/1.1 302 Found
13	3.776869	50.19.229.205	192.168.1.108	HTTP	607	HTTP/1.1 302 Found
16	11.283854	50.19.229.205	192.168.1.108	HTTP	607	HTTP/1.1 302 Found
18	70.319743	50.19.229.205	192.168.1.108	TCP	54	80 → 60139 [FIN, ACK] Seq=2213 Ack=5405 Win=17664

The HTTP server responds with **302 Found**.

0-7. Is there any IPv6 traffic in this trace file?

No.	Time	Source	Destination
1	0.000000	192.168.1.108	50.19.229.205
2	0.092419	50.19.229.205	192.168.1.108
3	0.092521	192.168.1.108	50.19.229.205
4	0.094027	192.168.1.108	50.19.229.205
5	0.196174	50.19.229.205	192.168.1.108
6	0.204299	50.19.229.205	192.168.1.108
7	0.221142	192.168.1.108	50.19.229.205
8	0.326624	50.19.229.205	192.168.1.108
9	0.600950	192.168.1.108	50.19.229.205
10	0.674498	50.19.229.205	192.168.1.108
11	0.674551	192.168.1.108	50.19.229.205
12	3.675382	192.168.1.108	50.19.229.205
13	3.776869	50.19.229.205	192.168.1.108
14	3.975053	192.168.1.108	50.19.229.205
15	11.175009	192.168.1.108	50.19.229.205
16	11.283854	50.19.229.205	192.168.1.108
17	11.478274	192.168.1.108	50.19.229.205
18	70.319743	50.19.229.205	192.168.1.108
19	70.319865	192.168.1.108	50.19.229.205
20	74.757892	192.168.1.108	50.19.229.205

No, it appears that all the source and destination addresses are IPv4.

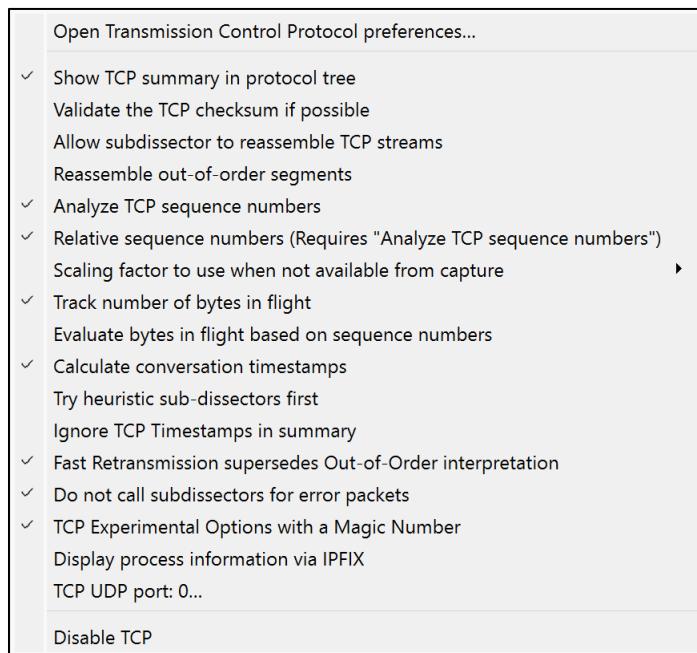
LAB 4: ADD HTTP HOST FIELD AS A COLUMN

i. Provide a screenshot of completed step 6.

No.	Time	Source	Destination	Protocol	Length	Host
15	8.974846	24.6.173.220	199.181.132.249	HTTP	342	www.disney.com
5723	14.380454	24.6.173.220	68.71.216.36	HTTP	1791	weblogger01.data...
5941	14.550770	24.6.173.220	66.235.138.59	HTTP	1952	w88.go.com
5730	14.381594	24.6.173.220	66.235.138.59	HTTP	1579	w88.go.com
1859	12.099824	24.6.173.220	68.71.209.50	HTTP	379	tredir.go.com
3456	13.588623	24.6.173.220	199.181.131.249	HTTP	338	search.disney.com
4876	14.117914	24.6.173.220	74.217.240.83	HTTP	431	pix04.revsci.net
3445	13.525109	24.6.173.220	74.217.240.83	HTTP	335	js.revsci.net
	32 11.415512	24.6.173.220	199.181.132.249	HTTP	338	disney.com

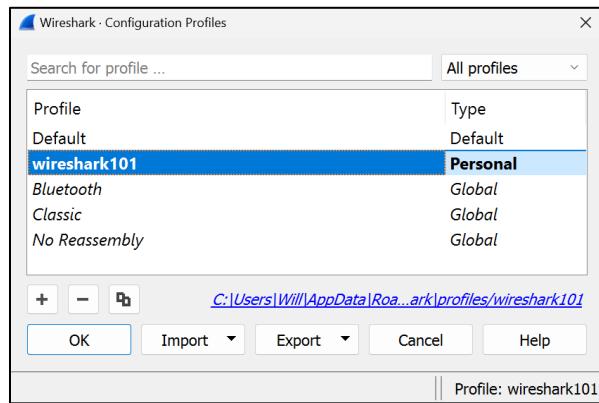
LAB 5: SET KEY WIRESHARK PREFERENCES

i. Follow all the steps in the lab and provide a complete screenshot of step 9.



LAB 6: CREATE NEW PROFILE BASED ON THE DEFAULT PROFILE

i. Make a profile labeled wireshark101 and provide a screenshot.

**LAB 7: IMPORT A DNS/HTTP ERRORS PROFILE**

i. Provide a screenshot of completed step 6.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.113	192.168.0.1	DNS	72	Standard query 0xa570 A www.
2	0.999676	192.168.0.113	192.168.0.1	DNS	72	Standard query 0xa570 A www.
3	1.999677	192.168.0.113	192.168.0.1	DNS	72	Standard query 0xa570 A www.
4	3.999680	192.168.0.113	192.168.0.1	DNS	72	Standard query 0xa570 A www.
5	5.095538	192.168.0.1	192.168.0.113	DNS	72	Standard query response 0xa5
6	5.095541	68.87.76.183	192.168.0.113	DNS	72	Standard query response 0xa5
7	6.047552	192.168.0.1	192.168.0.113	DNS	72	Standard query response 0xa5
8	6.047555	68.87.78.136	192.168.0.113	DNS	72	Standard query response 0xa5
9	21.825414	192.168.0.113	192.168.0.1	DNS	76	Standard query 0x8b73 A d.ge
10	21.837542	192.168.0.1	192.168.0.113	DNS	92	Standard query response 0x8b

LAB 8: SPOT PATH AND SERVER LATENCY PROBLEMS

i. Provide a screenshot of completed step 7.

No.	Time	TCP Delta	Source	Destination	Protocol
354	118.195308	118.195308000	24.6.173.220	69.4.231.53	TCP
210	29.006113	41.640641000	69.4.231.53	24.6.173.220	HTTP
34	0.006965	36.357656000	69.4.231.53	24.6.173.220	HTTP
16	18.096205	18.096205000	24.6.173.220	69.4.231.53	TCP
30	0.015479	18.052142000	69.4.231.53	24.6.173.220	HTTP
23	17.965049	17.965049000	69.4.231.53	24.6.173.220	HTTP
204	0.512248	14.907886000	69.4.231.53	24.6.173.220	TCP
202	1.115240	14.812617000	69.4.231.53	24.6.173.220	TCP
1098	14.745399	14.745399000	69.4.231.53	24.6.173.220	TCP

CHAPTER 1 CHALLENGE

1-1. In which frame number does the client request the default webpage (“/”)

No.	Time	Source	Destination	Protocol	Info
13	0.001319	24.6.169.43	24.6.173.220	HTTP	GET / HTTP/1.1
14	0.030997	24.6.173.220	24.6.169.43	HTTP	HTTP/1.1 200 OK
15	0.000510	24.6.173.220	24.6.169.43	HTTP	Continuation
17	0.000118	24.6.173.220	24.6.169.43	HTTP	Continuation
18	0.039374	24.6.169.43	24.6.173.220	HTTP	GET /style.css HTTP/1.1

Frame 13.

1-2. What response code does the server send in frame 17?

No.	Time	Source	Destination	Protocol	Info
14	0.030997	24.6.173.220	24.6.169.43	HTTP	HTTP/1.1 200 OK
15	0.000510	24.6.173.220	24.6.169.43	HTTP	Continuation
17	0.000118	24.6.173.220	24.6.169.43	HTTP	Continuation
18	0.039374	24.6.169.43	24.6.173.220	HTTP	GET /style.css HTTP/1.1

The response code is **200 OK**.

1-3. What is the largest TCP delta value seen in this trace file?

No.	Time	TCP Delta	Source	Destination	Protocol
285	0.285165	15.438012000	24.6.169.43	24.6.173.220	HTTP
286	0.001057	15.406091000	24.6.169.43	24.6.173.220	HTTP
287	0.000002	15.296079000	24.6.169.43	24.6.173.220	HTTP
279	2.688634	15.139514000	24.6.169.43	24.6.173.220	HTTP
264	4.865139	11.988774000	24.6.169.43	24.6.173.220	HTTP

Using the TCP Delta column that we set up in lab 8, frame 285 has the largest delta value at 15.438012000.

1-4. How many SYN packets arrived after at least a 1 second Delay

tcp.flags.syn == 1						
No.	Time	TCP Delta	Source	Destination	Protocol	Info
3	6.006083	6.006083000	24.6.169.43	24.6.173.220	TCP	[TCP Retransmission] 63286 → 87 [SYN]
6	5.999911	5.999911000	24.6.169.43	24.6.173.220	TCP	[TCP Retransmission] 63287 → 87 [SYN]
2	3.000825	3.000825000	24.6.169.43	24.6.173.220	TCP	[TCP Retransmission] 63286 → 87 [SYN]
5	2.995490	2.995490000	24.6.169.43	24.6.173.220	TCP	[TCP Retransmission] 63287 → 87 [SYN]
21	0.000737	0.000737000	24.6.173.220	24.6.169.43	TCP	87 → 63290 [SYN, ACK] Seq=0 Ack=1 Win=
27	0.000115	0.000450000	24.6.173.220	24.6.169.43	TCP	87 → 63293 [SYN, ACK] Seq=0 Ack=1 Win=

Still in the same view (sorted largest to smallest by TCP delta value), I applied `tcp.flags.syn == 1` as a display filter. It looks like four SYN packets arrived after at least a 1 second delay.

1-5. (*not in book*) Research and provide a chart of 5 valid HTTP/1.1 Status Codes. In your own words, include the code number, definition, and a brief example of when one might see it. Cite all sources in APA format.

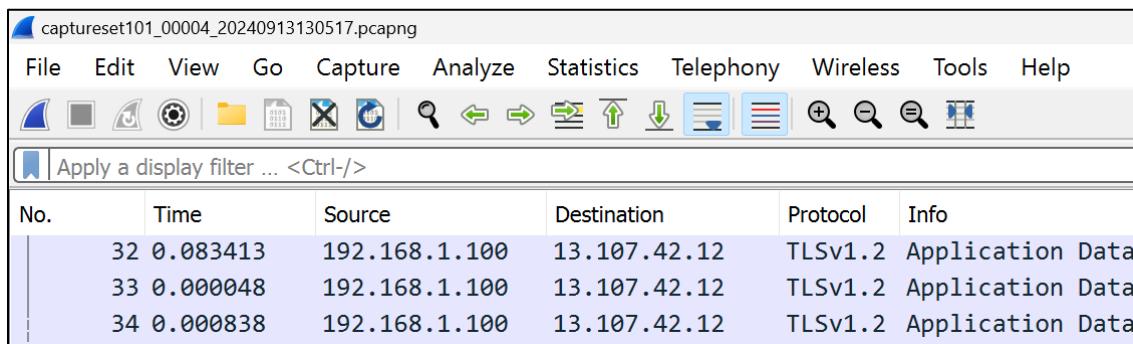
HTTP Status Codes		
Code	Category	Description
1xx	Informational	For informational purposes only, a request was received and is processing
2xx	Success	Request was processed successfully
3xx	Redirection	More action is needed before request can process successfully
4xx	Client Error	Client-side error; the request contains a syntax mistake or cannot be processed
5xx	Server Error	Server-side error; the request is good but cannot be processed

Code	Description
201 Created	The request was successfully processed and a new resource was created (e.g., new user account)
404 Not Found	Server could not find requested resource (e.g., client requests a page that has moved URLs without any redirection)
301 Moved Permanently	The resource has been moved to a new URL (e.g., a client requested www.website.com/info , but it is now www.website.com/about)
502 Bad Gateway	A proxy/gateway server got an invalid response from upstream (e.g., a networking misconfiguration causes a communication issue)
403 Forbidden	The request is good but is not fulfilled (e.g., wrong credentials are provided or user has insufficient permissions)

The table above is based on [RFC9110 \(sections 15.1 – 15.6\)](#), the [IANA HTTP status code registry](#), and [Mozilla documentation \(Fielding et al., 2022; IANA, 2022; Mozilla Developer Network, 2024\)](#).

LAB 9: CAPTURE TO FILE SETS

i. Follow all the steps in the lab and provide a complete screenshot of step 7.

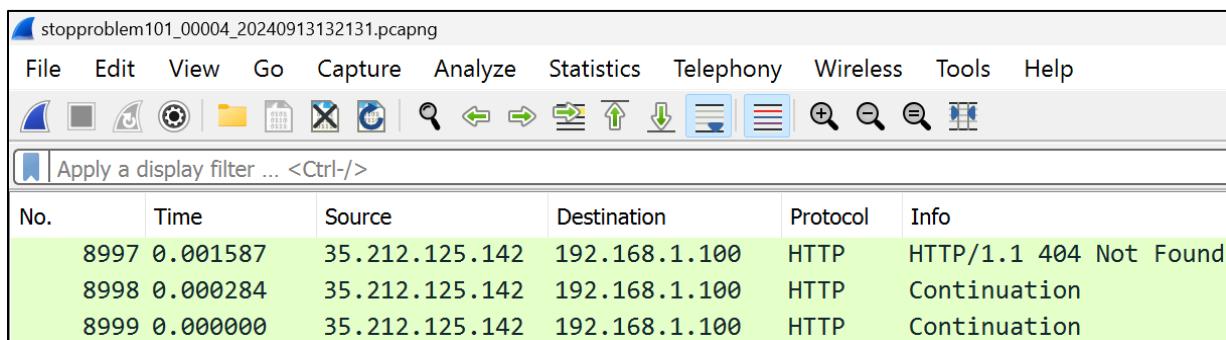


captureset101_00004_20240913130517.pcapng

No.	Time	Source	Destination	Protocol	Info
32	0.083413	192.168.1.100	13.107.42.12	TLSv1.2	Application Data
33	0.000048	192.168.1.100	13.107.42.12	TLSv1.2	Application Data
34	0.000838	192.168.1.100	13.107.42.12	TLSv1.2	Application Data

LAB 10: USE RING BUFFER TO CONSERVE DRIVE SPACE

i. Follow all the steps in the lab and provide a complete screenshot of step 10.

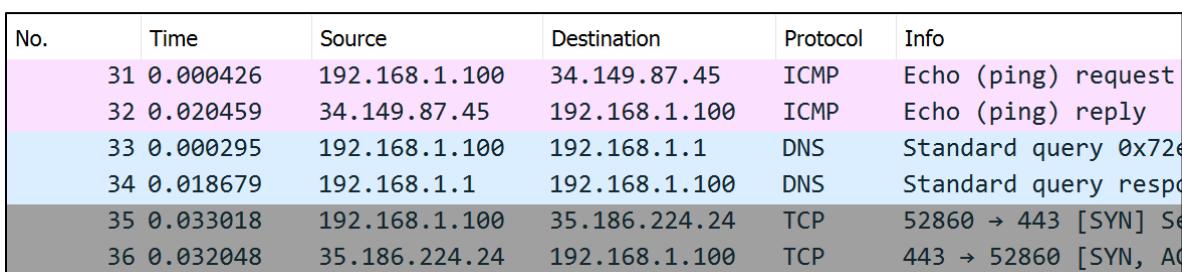


stopproblem101_00004_20240913132131.pcapng

No.	Time	Source	Destination	Protocol	Info
8997	0.001587	35.212.125.142	192.168.1.100	HTTP	HTTP/1.1 404 Not Found
8998	0.000284	35.212.125.142	192.168.1.100	HTTP	Continuation
8999	0.000000	35.212.125.142	192.168.1.100	HTTP	Continuation

LAB 11: CAPTURE ONLY TRAFFIC TO OR FROM YOUR IP ADDRESS

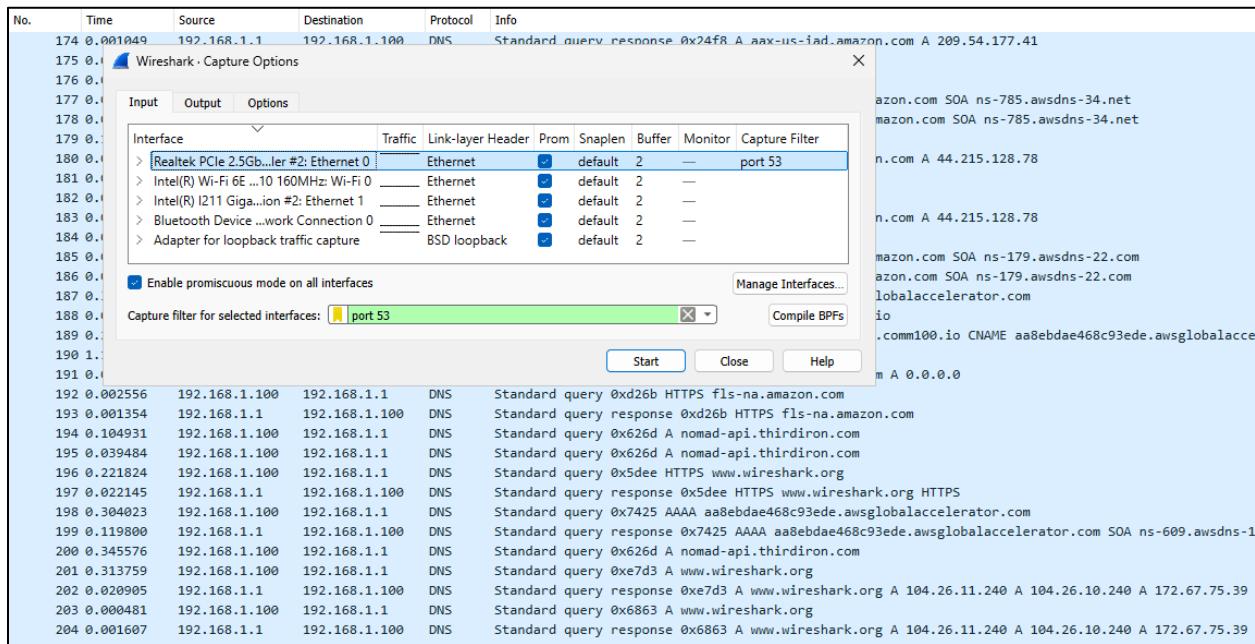
i. Follow all the steps in the lab and provide a complete screenshot of step 6



No.	Time	Source	Destination	Protocol	Info
31	0.000426	192.168.1.100	34.149.87.45	ICMP	Echo (ping) request
32	0.020459	34.149.87.45	192.168.1.100	ICMP	Echo (ping) reply
33	0.000295	192.168.1.100	192.168.1.1	DNS	Standard query 0x72e
34	0.018679	192.168.1.1	192.168.1.100	DNS	Standard query resp
35	0.033018	192.168.1.100	35.186.224.24	TCP	52860 → 443 [SYN] Se
36	0.032048	35.186.224.24	192.168.1.100	TCP	443 → 52860 [SYN, AC

LAB 13: CREATE, SAVE, AND APPLY A DNS CAPTURE FILTER

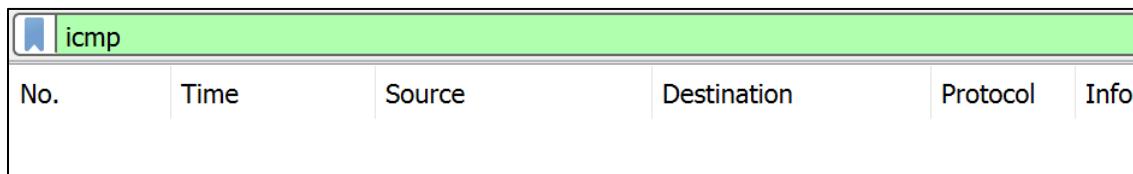
i. Follow all the steps in the lab and provide a complete screenshot of step 10.



The screenshot shows the Wireshark interface with a list of captured DNS packets. A capture filter is applied to port 53. The interface list shows several network interfaces, including Realtek PCIe 2.5Gb...ler #2: Ethernet 0, Intel(R) Wi-Fi 6E ...10 160MHz: Wi-Fi 0, Intel(R) I211 Gigabit...ion #2: Ethernet 1, Bluetooth Device ...work Connection 0, and Adapter for loopback traffic capture. The 'Enable promiscuous mode on all interfaces' checkbox is checked. The 'Capture filter for selected interfaces' field contains 'port 53'. The list of captured packets shows various DNS queries and responses, such as 'Standard query 0xd26b HTTPS fls-na.amazon.com' and 'Standard query response 0xd26b HTTPS fls-na.amazon.com'.

CHAPTER 2 CHALLENGE

2-1. Did you capture any ICMP traffic?



The screenshot shows the Wireshark interface with a list of captured packets. An ICMP capture filter is applied. The table header includes columns for No., Time, Source, Destination, Protocol, and Info. The list of captured packets is empty, indicating no ICMP traffic was captured.

No ICMP packets were captured.

2-2. What protocols are listed for your browsing session to www.ChapelleU.com? (redirects to www.chappell-university.com)

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.100	18.236.36.28	TCP	54512 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.001897	18.236.36.28	192.168.1.100	TCP	80 → 54512 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1
3	0.000040	192.168.1.100	18.236.36.28	TCP	54512 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
4	0.000085	192.168.1.100	18.236.36.28	HTTP	GET / HTTP/1.1

I couldn't connect to an HTTP version of www.chappell-university.com (only HTTPS), so I browsed to www.testingmcafesites.com instead to generate this traffic.

The TCP and HTTP protocols are listed for the browsing session.

2-3. How many ICMP packets did you capture?

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.100	34.149.87.45	ICMP	Echo (ping) request id=0x0001, seq=25/6400, ttl=128 (reply in 2)
2	0.020650	34.149.87.45	192.168.1.100	ICMP	Echo (ping) reply id=0x0001, seq=25/6400, ttl=56 (request in 1)
3	0.988892	192.168.1.100	34.149.87.45	ICMP	Echo (ping) request id=0x0001, seq=26/6656, ttl=128 (reply in 4)
4	0.019450	34.149.87.45	192.168.1.100	ICMP	Echo (ping) reply id=0x0001, seq=26/6656, ttl=56 (request in 3)
5	0.995769	192.168.1.100	34.149.87.45	ICMP	Echo (ping) request id=0x0001, seq=27/6912, ttl=128 (reply in 6)
6	0.019963	34.149.87.45	192.168.1.100	ICMP	Echo (ping) reply id=0x0001, seq=27/6912, ttl=56 (request in 5)
7	0.981919	192.168.1.100	34.149.87.45	ICMP	Echo (ping) request id=0x0001, seq=28/7168, ttl=128 (reply in 8)
8	0.018964	34.149.87.45	192.168.1.100	ICMP	Echo (ping) reply id=0x0001, seq=28/7168, ttl=56 (request in 7)

8 ICMP packets were captured (4 requests and 4 replies).

2-4. What ICMP Type and Code numbers are listed in your Trace File?

▼ Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0

▼ Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0

The requests show Type 8/Code 0, and the replies show Type 0/Code 0.

2-5. (not in book) Research ICMP Types and Codes. Provide a concise explanation for each of the following Type 3 Code conditions; what does it mean exactly? Explain the specifics of why one would get this response.

Type	Code	
3	0	Network unreachable
	1	Host unreachable
	3	Port unreachable
	5	Source route failed
	7	Destination host unknown
	10	Communication with destination host is administratively prohibited
	12	Destination host unreachable for type of service

a. Type 3/Code 0 -- Network unreachable

The destination network is unreachable – no packets could be routed there. This could happen when a router tries to forward a packet to the correct network, but then finds that there isn’t any path in its routing table or it’s too far away to reach (MartinGarcia & Lyon, n.d., ICMP Codes Section; Osterloh, 2002, p. 4).

b. Type 3/Code 1 -- Host unreachable

The host’s network is reachable, but the actual host is not. This could happen when a router tries to deliver a packet to a disconnected host. The router then checks to see if it has a MAC address for the host on the local network, attempts to find the MAC address using ARP, receives no reply, and replies with Host Unreachable (MartinGarcia & Lyon, n.d., ICMP Codes Section; Osterloh, 2002, p. 4).

c. Type 3/Code 3 -- Port unreachable

The network and the host are reachable, but the specific port on the host is not. A packet gets routed and delivered to a host, but the destination port is closed/invalid, or the process isn’t running (MartinGarcia & Lyon, n.d., ICMP Codes Section; Osterloh, 2002, p. 4).

- d. Type 3/Code 5 -- Source route failed

Source routing is a kind of routing where the path that a packet takes is specified before it is sent (Red Hat, n.d.). The Source Route Failed response occurs when the destination can't be reached by the source route (MartinGarcia & Lyon, n.d., ICMP Codes Section; Osterloh, 2002, p. 4).

- e. Type 3/Code 7 -- Destination host unknown

The destination host is not known by the router on its network. A packet is received by a router, but the address is bad, or the destination is unknown. It sends this message in that event (MartinGarcia & Lyon, n.d., ICMP Codes Section; Osterloh, 2002, p. 4).

- f. Type 3/Code 10 -- Communication with destination host is administratively prohibited

The destination network and host are reachable, but the last hop router is denying communication with the destination host – it is not allowing traffic to pass (MartinGarcia & Lyon, n.d., ICMP Codes Section; Osterloh, 2002, p. 4).

- g. Type 3/Code 12 -- Destination host unreachable for type of service

The destination network is reachable, but the destination host is not because the specified type of service (TOS) in the IP header (second byte) isn't available. The host cannot provide that service (MartinGarcia & Lyon, n.d., ICMP Codes Section; Osterloh, 2002, p. 4; Thomas, 2014).

LAB 14: USE AUTO-COMPLETE TO FIND TRAFFIC TO A SPECIFIC HTTP SERVER

i. Follow all the steps in the lab and provide a complete screenshot of step 5.

http.host contains "hearst"						
No.	Time	Source	Destination	Protocol	Host	Info
159	0.000000	24.6.173.220	208.93.137.180	HTTP	aps.hearstnlp.com	GET /Scripts/loadAds.js HTTP/1.1
388	0.127133	24.6.173.220	208.93.137.180	HTTP	aps.hearstnlp.com	GET /Scripts/loadAdsMain.js HTTP/1.1
406	0.029183	24.6.173.220	208.93.137.180	HTTP	aps.hearstnlp.com	GET /SRO/GetJS?url=www.sfgate.com/feedb
458	0.163355	24.6.173.220	208.93.137.180	HTTP	aps.hearstnlp.com	GET /Scripts/initDefineAds.js HTTP/1.1
586	0.554234	24.6.173.220	216.155.207.26	HTTP	cm.npc-hearst.overture.com	GET /js_1_0/?config=2130893885&type=new
1071	0.346887	24.6.173.220	23.23.99.162	HTTP	hearst.jump-time.net	GET /sfgate.gif?url=http%3A//www.sfgate
10055	66.874309	24.6.173.220	208.93.137.180	HTTP	aps.hearstnlp.com	GET /SRO/GetJS?url=www.sfgate.com/%3Fco
10067	0.664242	24.6.173.220	208.93.137.180	HTTP	aps.hearstnlp.com	GET /SRO/GetJS?url=extras.sfgate.com/sf
10250	3.045941	24.6.173.220	216.155.207.26	HTTP	cm.npc-hearst.overture.com	GET /js_1_0/?config=2130893885&type=new
10332	0.270298	24.6.173.220	23.23.99.162	HTTP	hearst.jump-time.net	GET /sfgate.gif?url=http%3A//www.sfgate

ii. Follow all the steps in the lab and provide a complete screenshot of step 6.

http.request.method=="POST"						
No.	Time	Source	Destination	Protocol	Host	Info
859	0.000000	24.6.173.220	199.7.57.72	OCSP	ocsp.verisign.com	Request
864	0.000510	24.6.173.220	199.7.57.72	OCSP	ocsp.verisign.com	Request
865	0.000430	24.6.173.220	199.7.57.72	OCSP	ocsp.verisign.com	Request
897	0.013423	24.6.173.220	199.7.57.72	OCSP	ocsp.verisign.com	Request
898	0.000324	24.6.173.220	199.7.57.72	OCSP	ocsp.verisign.com	Request
2043	2.645699	24.6.173.220	67.192.92.227	HTTP	ad.auditude.com	POST /adserver?u=97df6f8f08d8730261d4b44204353b4c&u=69832e95d26ae65e6
3418	4.189073	24.6.173.220	208.81.191.110	HTTP	www.meebo.com	POST /cmd/cx HTTP/1.1 (application/x-www-form-urlencoded)
3419	0.000360	24.6.173.220	208.81.191.110	HTTP	www.meebo.com	POST /cmd/tc HTTP/1.1 (application/x-www-form-urlencoded)
3476	0.245204	24.6.173.220	208.81.191.110	HTTP	www.meebo.com	POST /cmd/getrotate HTTP/1.1 (application/x-www-form-urlencoded)
10022	59.080083	24.6.173.220	208.93.137.180	HTTP	extras.sfgate.com	POST /sfgate/modules/formHandlers/sfgSupportMailHandler.php HTTP/1.1
10406	5.266161	24.6.173.220	208.81.191.110	HTTP	www.meebo.com	POST /cmd/cx HTTP/1.1 (application/x-www-form-urlencoded)
10578	0.510405	24.6.173.220	67.192.92.227	HTTP	ad.auditude.com	POST /adserver?u=97df6f8f08d8730261d4b44204353b4c&u=69832e95d26ae65e6

LAB 16: FILTER ON HTTP TRAFFIC THE RIGHT WAY

i. Follow all the steps in the lab and provide a complete screenshot of step 3.

tcp.port == 80						
No.	Time	Source	Destination	Protocol	Info	
12	0.000000	24.6.173.220	199.181.132.249	TCP	35518 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460	
13	0.034726	199.181.132.249	24.6.173.220	TCP	80 → 35518 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0	
14	0.000075	24.6.173.220	199.181.132.249	TCP	35518 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0	
15	0.000370	24.6.173.220	199.181.132.249	HTTP	GET / HTTP/1.1	
16	0.032641	199.181.132.249	24.6.173.220	HTTP	HTTP/1.1 301 Moved Permanently (text/html)	
21	0.108487	24.6.173.220	199.181.132.249	TCP	35519 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460	
22	0.091631	24.6.173.220	199.181.132.249	TCP	35518 → 80 [ACK] Seq=289 Ack=461 Win=65240 Len=0	

> Frame 12: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{6E79FEC0-FF

> Ethernet II, Src: HewlettP_a7:bf:a3 (d4:85:64:a7:bf:a3), Dst: Cadant_31:bb:c1 (00:01:5c:31:bb:c1)

> Internet Protocol Version 4, Src: 24.6.173.220, Dst: 199.181.132.249

> Transmission Control Protocol, Src Port: 35518, Dst Port: 80, Seq: 0, Len: 0

HTTP http-disney101.pcapng | Packets: 6143 · Displayed: 5917 (96.3%) | Profile: wireshark101

Screenshot edited to include packet display count

LAB 17: FILTER ON TRAFFIC TO OR FROM ONLINE BACKUP SUBNETS

i. Follow all the steps in the lab and provide a complete screenshot of step 3.

ip.addr == 216.115.74.0/24					
No.	Time	Source	Destination	Protocol	Info
118	0.000000	24.6.173.220	216.115.74.235	TCP	1145 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
119	0.031742	216.115.74.235	24.6.173.220	TCP	80 → 1145 [SYN, ACK] Seq=0 Ack=1 Win=3900 Len=0 MSS=1300 WS=1 SACK_PERM=1
120	0.000331	24.6.173.220	216.115.74.235	TCP	1145 → 80 [ACK] Seq=1 Ack=1 Win=66300 Len=0
121	0.000576	24.6.173.220	216.115.74.235	HTTP	GET /php/updateMetric.php?product_key=MABPEME000-6E2P-2ACJ-3KP3-JF0E-009F&
122	0.037863	216.115.74.235	24.6.173.220	HTTP	HTTP/1.1 200 OK (text/html)
123	0.003173	24.6.173.220	216.115.74.235	TCP	1145 → 80 [RST, ACK] Seq=227 Ack=581 Win=0 Len=0
131	2.218194	24.6.173.220	216.115.74.235	TCP	1146 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
132	0.031846	216.115.74.235	24.6.173.220	TCP	80 → 1146 [SYN, ACK] Seq=0 Ack=1 Win=3900 Len=0 MSS=1300 WS=1 SACK_PERM=1
133	0.000395	24.6.173.220	216.115.74.235	TCP	1146 → 80 [ACK] Seq=1 Ack=1 Win=66300 Len=0

LAB 18: FILTER ON DNS NAME ERRORS OR HTTP 404 RESPONSES

i. Follow all the steps in the lab and provide a complete screenshot of step 2

dns.flags.rcode == 3					

ii. Follow all the steps in the lab and provide a complete screenshot of step 4.

(dns.flags.rcode == 3) (http.response.code == 404)					
No.	Time	Source	Destination	Protocol	Info
9	0.000000	198.66.239.146	24.6.173.220	HTTP	HTTP/1.1 404 Not Found (text/html)
18	9.902010	75.75.75.75	24.6.173.220	DNS	Standard query response 0x8e30 No such name
27	15.749441	198.66.239.146	24.6.173.220	HTTP	HTTP/1.1 404 Not Found (text/html)

LAB 19: DETECT BACKGROUND FILE TRANSFERS ON STARTUP

i. Follow all the steps in the lab and provide a complete screenshot of step 4.

No.	Time	Source	Destination	Protocol	Info
309	0.009279	2001:558:6045:...	2001:558:feed:...	DNS	Standard query 0xb240 A dl-client709.dropbox.com
310	0.012170	2001:558:feed:...	2001:558:6045:...	DNS	Standard query response 0xb240 A dl-client709.dropbox.com A 50.17.223.168
311	0.000784	24.6.169.43	50.17.223.168	TCP	54693 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
312	0.029152	108.160.161.163	24.6.169.43	TCP	80 → 54690 [ACK] Seq=179 Ack=397 Win=16896 Len=0
313	0.070578	50.17.223.168	24.6.169.43	TCP	443 → 54693 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=128
314	0.000784	24.6.169.43	50.17.223.168	TCP	54693 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0

LAB 20: LOCATE TCP CONNECTION ATTEMPTS TO A CLIENT

i. Follow all the steps in the lab and provide a complete screenshot of step 3.

tcp.flags == 0x002 && ip.dst == 24.6.0.0/16					
No.	Time	Source	Destination	Protocol	Info
352	0.000000	121.125.72.180	24.6.169.43	TCP	57003 → 8880 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
353	0.256469	121.125.72.180	24.6.173.220	TCP	57003 → 8880 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
535	53.885510	24.6.169.43	24.6.173.220	TCP	54708 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
537	2.999402	24.6.169.43	24.6.173.220	TCP	[TCP Retransmission] 54708 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
551	6.007251	24.6.169.43	24.6.173.220	TCP	[TCP Retransmission] 54708 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1

LAB 21: FILTER TO LOCATE A SET OF KEY WORDS IN A TRACE FILE

i. Follow all the steps in the lab and provide a complete screenshot of step 3.

frame matches "(?i)(sombrero football)"					
No.	Time	Source	Destination	Protocol	Info
1563	0.000000	24.6.173.220	184.28.78.185	HTTP	GET /file_thumbview_approve/16268884/1/stock-photo-16268884-
3418	39.792172	24.6.173.220	184.28.78.185	HTTP	GET /file_thumbview_approve/21968700/1/stock-photo-21968700-
3740	2.938768	24.6.173.220	184.28.78.185	HTTP	GET /file_thumbview_approve/21968700/2/stock-photo-21968700-

LAB 22: FILTER WITH WILDCARDS BETWEEN WORDS

i. Follow all the steps in the lab and provide a complete screenshot of step 3.

http.request.uri matches "baby.{1,20}smiling"					
No.	Time	Source	Destination	Protocol	Info
404	0.000000	24.6.173.220	184.28.78.185	HTTP	GET /file_thumbview_approve/10195917/1/stock-video-10195917-baby-on-belly-smiling.jpg HTTP/1.1
427	0.042738	24.6.173.220	184.28.78.185	HTTP	GET /file_thumbview_approve/16072653/1/stock-photo-16072653-mom-and-baby-smiling.jpg HTTP/1.1
749	3.539544	24.6.173.220	184.28.78.185	HTTP	GET /file_thumbview_approve/16072653/2/stock-photo-16072653-mom-and-baby-smiling.jpg HTTP/1.1

LAB 23: IMPORT DISPLAY FILTERS INTO A PROFILE

i. Follow all the steps in the lab and provide a complete screenshot of step 3.

```
# This file is automatically generated, DO NOT MODIFY.
~
```

My file was initially empty (besides a comment)

LAB 24: CREATE AND IMPORT HTTP FILER EXP BUTTONS

i. Follow all the steps in the lab and provide a complete screenshot of step 7.

TCPFlags UDP>1 TCP>.5 iRTT>.150 HTTP>1 DNS>.1 DNSErr HTTPErr SMBErr GET|POST

CHAPTER 3 CHALLENGE

3-1. How many frames travel to or from **80.78.246.209**?

ip.addr == 80.78.246.209					
No.	Time	Source	Destination	Protocol	Info
485	0.000000	24.6.181.160	80.78.246.209	TCP	1270 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
486	0.000813	24.6.181.160	80.78.246.209	TCP	1271 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
487	0.212933	80.78.246.209	24.6.181.160	TCP	80 → 1271 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=128
488	0.000865	24.6.181.160	80.78.246.209	TCP	1271 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
489	0.000953	80.78.246.209	24.6.181.160	TCP	80 → 1270 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=128
490	0.000763	24.6.181.160	80.78.246.209	HTTP	GET / HTTP/1.1

Flags (12 bits) (tcp.flags), 2 bytes Packets: 519 · Displayed: 32 (6.2%) Profile: wireshark101

Screenshot edited to include packet display count

I used the display filter `ip.addr == 80.78.246.209` to find that 32 frames travelled to or from that IP address.

3-2. How many DNS packets are in this trace file?

dns					
No.	Time	Source	Destination	Protocol	Info
396	0.000000	2001:558:6045:...	2001:558:feed:...	DNS	Standard query 0xf7a5 A hackers.ru
397	0.187676	2001:558:feed:...	2001:558:6045:...	DNS	Standard query response 0xf7a5 A hackers.ru SOA ns1.inforography.ru
398	0.001206	2001:558:6045:...	2001:558:feed:...	DNS	Standard query 0xcba7 AAAA hackers.ru
399	0.215063	2001:558:feed:...	2001:558:6045:...	DNS	Standard query response 0xcba7 AAAA hackers.ru SOA ns1.inforography.ru
480	9.049289	2001:558:6045:...	2001:558:feed:...	DNS	Standard query 0xed4a A www.webhackers.ru
482	0.268204	2001:558:feed:...	2001:558:6045:...	DNS	Standard query response 0xed4a A www.webhackers.ru A 80.78.246.209
483	0.002437	2001:558:6045:...	2001:558:feed:...	DNS	Standard query 0x1bc1 AAAA www.webhackers.ru
484	0.265009	2001:558:feed:...	2001:558:6045:...	DNS	Standard query response 0x1bc1 AAAA www.webhackers.ru SOA dns1.yandex.ru

I used a `dns` display filter to find that there are 8 DNS packets in the file.

3-3. How many frames have the TCP SYN bit set to **1**?

tcp.flags.syn == 1					
No.	Time	Source	Destination	Protocol	Info
1	0.000000	2001:558:6045:...	2001:4860:4001...	TCP	1194 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1440 WS=4 SACK_PERM=1
2	0.000961	2001:558:6045:...	2001:4860:4001...	TCP	1195 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1440 WS=4 SACK_PERM=1
3	0.014608	2001:4860:4001...	2001:558:6045:...	TCP	80 → 1195 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1410 SACK_PERM=1 WS=64
6	0.001548	2001:4860:4001...	2001:558:6045:...	TCP	80 → 1194 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1410 SACK_PERM=1 WS=64
466	65.255117	2001:558:6045:...	2001:4860:4001...	TCP	1268 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1440 WS=4 SACK_PERM=1
467	0.027720	2001:4860:4001...	2001:558:6045:...	TCP	80 → 1268 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1410 SACK_PERM=1 WS=64
485	6.428263	24.6.181.160	80.78.246.209	TCP	1270 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
486	0.000813	24.6.181.160	80.78.246.209	TCP	1271 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
487	0.212933	80.78.246.209	24.6.181.160	TCP	80 → 1271 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=128
489	0.001818	80.78.246.209	24.6.181.160	TCP	80 → 1270 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=128
501	0.774140	24.6.181.160	80.78.246.209	TCP	1272 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
503	0.212099	80.78.246.209	24.6.181.160	TCP	80 → 1272 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=128

Using the `tcp.flags.syn == 1` display filter, there are 12 frames with the SYN bit set.

3-4. How many frames contain a string **set-cookie** in upper or lower case?

frame matches "(?i)(set-cookie)"					
No.	Time	Source	Destination	Protocol	Info
9	0.000000	2001:4860:4001...	2001:558:6045:...	HTTP	HTTP/1.1 200 OK [BoundErrorUnreassembled Packet]
471	65.258567	2001:4860:4001...	2001:558:6045:...	HTTP	HTTP/1.1 200 OK (GIF89a)
475	2.997750	2001:4860:4001...	2001:558:6045:...	TCP	[TCP Retransmission] 80 → 1268 [PSH, ACK] Seq=1 A

3 frames contain **set-cookie** in upper or lower case. In lab 21, we used `frame matches "(?i)(sombrero|football)"` to find regex matches for frames containing the string “sombrero” or “football” (case insensitive). I used the same filter edited to match “set-cookie” instead.

3-5. How many frames contain the TCP delta time greater than 1 second?

tcp.time_delta > 1						
No.	Time	TCP Delta	Source	Destination	Protocol	Info
369	0.000000	6.926872000	2001:558:6045::	2001:4860:4001...	HTTP	GET /search?hl=en&rls=com.microsoft:en-us;IE-Address&
389	0.526390	6.878651000	2001:558:6045::	2001:4860:4001...	HTTP	GET /csi?v=3&s=web&action=&ei=ESRBUI0H5DjiwLL2oC4BA&
392	2.261963	2.620065000	2001:558:6045::	2001:4860:4001...	HTTP	GET /url?sa=t&rct=j&q=metasploit&source=web&cd=1&ved=
400	53.111153	55.021905000	2001:558:6045::	2001:4860:4001...	HTTP	GET /search?q=hackers.ru&sourceid=ie7&rls=com.microso
472	1.366705	54.230623000	2001:558:6045::	2001:4860:4001...	HTTP	GET /csi?v=3&s=web&action=&e=17259,28290,28663,37102,
475	2.430248	2.997750000	2001:4860:4001...	2001:558:6045...	TCP	[TCP Retransmission] 80 → 1268 [PSH, ACK] Seq=1 Ack=7
477	2.680229	5.958008000	2001:558:6045::	2001:4860:4001...	HTTP	GET /url?sa=t&rct=j&q=hackers.ru&source=web&cd=1&ved=
509	16.510075	14.645575000	24.6.181.160	80.78.246.209	TCP	1270 → 80 [RST, ACK] Seq=322 Ack=4121 Win=0 Len=0
510	0.000746	15.160264000	24.6.181.160	80.78.246.209	TCP	1271 → 80 [RST, ACK] Seq=446 Ack=903 Win=0 Len=0
511	0.002305	16.213878000	2001:558:6045::	2001:4860:4001...	TCP	1194 → 80 [RST, ACK] Seq=5811 Ack=106956 Win=0 Len=0
512	0.000839	21.371147000	2001:558:6045::	2001:4860:4001...	TCP	1195 → 80 [RST, ACK] Seq=5183 Ack=238514 Win=0 Len=0
513	0.003172	18.998414000	2001:558:6045::	2001:4860:4001...	TCP	1268 → 80 [RST, ACK] Seq=721 Ack=494 Win=0 Len=0
514	50.601829	64.799323000	80.78.246.209	24.6.181.160	TCP	80 → 1272 [FIN, ACK] Seq=728 Ack=208 Win=6912 Len=0
515	1.303469	1.303469000	80.78.246.209	24.6.181.160	TCP	[TCP Retransmission] 80 → 1272 [FIN, ACK] Seq=728 Ack
516	2.556541	2.556541000	80.78.246.209	24.6.181.160	TCP	[TCP Retransmission] 80 → 1272 [FIN, ACK] Seq=728 Ack
517	5.248543	5.248543000	80.78.246.209	24.6.181.160	TCP	[TCP Retransmission] 80 → 1272 [FIN, ACK] Seq=728 Ack
518	10.285664	10.285664000	80.78.246.209	24.6.181.160	TCP	[TCP Retransmission] 80 → 1272 [FIN, ACK] Seq=728 Ack
519	20.512819	20.512819000	80.78.246.209	24.6.181.160	TCP	[TCP Retransmission] 80 → 1272 [FIN, ACK] Seq=728 Ack

Screenshot edited to include packet display count

18 frames have a TCP delta time greater than 1 second. In a previous lab (from a different week), we applied `tcp.time_delta` as a column. I re-enabled the column, then prepared a display filter using the context menus in Wireshark. I edited the filter to be `tcp.time_delta > 1` and applied it.

LAB 25: ADD A COLUMN TO DISPLAY COLORING RULES IN USE

i. Follow all the steps in the lab and provide a screenshot of step 4.

No.	Time	Source	Destination	Protocol	Coloring Rule Name	Info
472	0.000004	66.109.241.50	24.6.173.220	HTTP	HTTP	Continuation
473	0.000176	24.6.173.220	66.109.241.50	TCP	HTTP	10623 → 80 [ACK] Seq=316 Ack=11041 Win=66240 Len=0
474	0.000778	66.109.241.50	24.6.173.220	HTTP	HTTP	Continuation
475	0.008319	66.109.241.50	24.6.173.220	TCP	Bad TCP	[TCP Dup ACK 410#1] 80 → 10623 [ACK] Seq=12421 Ack=316 Win=65220 Len=0
476	0.027249	24.6.173.220	75.75.75.75	DNS	UDP	Standard query 0x7394 A partner.googleadservices.com

LAB 26: BUILD A COLORING RULE

i. Follow all the steps in the lab and provide a screenshot of step 5.

No.	Time	Source	Destination	Protocol	Info
11	0.000473	10.234.125.254	10.121.70.151	FTP	Request: PASS merlin
12	0.000804	10.121.70.151	10.234.125.254	TCP	21 → 2221 [ACK] Seq=1 Ack=1 Win=49152 Len=0
13	0.007684	10.121.70.151	10.234.125.254	FTP	Response: 530 Login incorrect.
14	0.001176	10.234.125.254	10.121.70.151	TCP	2220 → 21 [FIN, ACK] Seq=1 Ack=23 Win=17447 Len=0
15	0.000839	10.121.70.151	10.234.125.254	TCP	21 → 2224 [ACK] Seq=1 Ack=1 Win=49152 Len=0
16	0.007129	10.121.70.151	10.234.125.254	FTP	Response: 331 Password required for admin.
17	0.001306	10.234.125.254	10.121.70.151	FTP	Request: PASS mercury

LAB 27: CREATE TEMPORARY CONVERSATION COLORING RULES

i. Follow all the steps in the lab and provide a screenshot of step 5.

No.	Time	Source	Destination	Protocol	Info
138	0.045044	210.72.21.11	24.6.173.220	HTTP	Continuation
139	0.002220	210.72.21.11	24.6.173.220	HTTP	Continuation
140	0.000125	24.6.173.220	210.72.21.11	TCP	61601 → 80 [ACK] Seq=268 Ack=4742 Win=65700 Len=0
141	0.082919	209.177.86.18	24.6.173.220	HTTP	Continuation

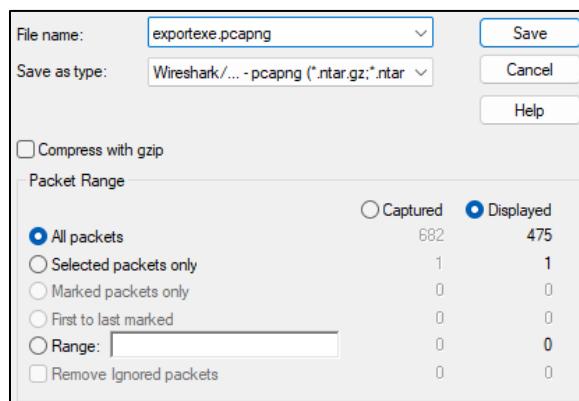
LAB 28: USE THE INTELLIGENT SCROLLBAR TO QUICKLY FIND PROBLEMS

i. Follow all the steps in the lab and provide a screenshot of step 4.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	161.58.73.170	12.234.12.108	HTTP	HTTP/1.0 401 Authorization Required (text/html)
2	0.000083	161.58.73.170	12.234.12.108	TCP	80 → 1124 [FIN, ACK] Seq=1306 Ack=1 Win=49152 Len=0
3	0.000038	12.234.12.108	161.58.73.170	TCP	1124 → 80 [ACK] Seq=1 Ack=1307 Win=63207 Len=0
4	10.536317	12.234.12.108	161.58.73.170	TCP	1124 → 80 [FIN, ACK] Seq=1 Ack=1307 Win=63207 Len=0
5	0.000629	12.234.12.108	161.58.73.170	TCP	1125 → 80 [SYN] Seq=0 Win=0 MSS=1460 SACK_PERM=1
6	0.096437	161.58.73.170	12.234.12.108	TCP	80 → 1124 [ACK] Seq=1307 Ack=2 Win=49152 Len=0
7	2.869444	12.234.12.108	161.58.73.170	TCP	[TCP Retransmission] 1125 → 80 [SYN] Seq=0 Win=64512 Len=0 MSS=1460 SACK_PERM=1
8	6.008476	12.234.12.108	161.58.73.170	TCP	[TCP Retransmission] 1125 → 80 [SYN] Seq=0 Win=64512 Len=0 MSS=1460 SACK_PERM=1
9	0.156745	161.58.73.170	12.234.12.108	TCP	80 → 1125 [SYN, ACK] Seq=0 Ack=1 Win=49152 Len=0 MSS=1460 SACK_PERM=1
10	0.000079	12.234.12.108	161.58.73.170	TCP	1125 → 80 [ACK] Seq=1 Ack=1 Win=64512 Len=0
11	0.000291	12.234.12.108	161.58.73.170	HTTP	GET /stats HTTP/1.1
12	0.087260	161.58.73.170	12.234.12.108	TCP	80 → 1125 [ACK] Seq=1 Ack=382 Win=49152 Len=0

LAB 29: EXPORT A SINGLE TCP CONVERSATION

i. Follow all the steps in the lab and provide a screenshot of step 4.

**LAB 30: EXPORT A LIST OF HTTP HOST FIELD VALUES FROM A TRACE FILE**

i. Follow all the steps in the lab and provide a screenshot of step 7.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	No.	Time	Source	Destination	Host	Protocol	Info							
2	8	0.24.6.173.220	208.48.81.133	www.freewebsites.com.au	HTTP	GET /HTTP/1.1								
3	11	0.100123	24.6.173.220	208.48.81.133	www.freewebsites.com.au	HTTP	GET /unique/track.js?referrer=about%3Abank	HTTP/1.1						
4	14	0.042474	24.6.173.220	208.48.81.133	www.freewebsites.com.au	HTTP	GET /?pagesection=body	HTTP/1.1						
5	18	0.021377	24.6.173.220	208.48.81.133	www.freewebsites.com.au	HTTP	GET /?pagesection=forsale	HTTP/1.1						
6	25	0.061105	24.6.173.220	208.48.81.133	www.freewebsites.com.au	HTTP	GET /common/fabulousdomains/skins/fab/images/banner/fabulous_sale_bottom.gif	HTTP/1.1						
7	26	0.000379	24.6.173.220	208.48.81.133	www.freewebsites.com.au	HTTP	GET /common/fabulousdomains/skins/fab/images/banner/sale_buynow.png	HTTP/1.1						
8	35	0.052922	24.6.173.220	208.48.81.133	www.freewebsites.com.au	HTTP	GET /common/fabulousdomains/skins/fab/images/banner/sale_bg.png	HTTP/1.1						

CHAPTER 4 CHALLENGE

4-1. What coloring rule does frame 170 match?

```

Frame 170: 1210 bytes on wire (9680 bits), 1210 bytes captured (9680 bits) on interface unknown, id 0
  > Interface id: 0 (unknown)
  Encapsulation type: Ethernet (1)
  Arrival Time: Aug 31, 2012 16:51:38.466332000 Eastern Daylight Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1346446298.466332000 seconds
  [Time delta from previous captured frame: 0.005430000 seconds]
  [Time delta from previous displayed frame: 0.005430000 seconds]
  [Time since reference or first frame: 0.532897000 seconds]
  Frame Number: 170
  Frame Length: 1210 bytes (9680 bits)
  Capture Length: 1210 bytes (9680 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp:http:data-text-lines]
  [Coloring Rule Name: Bad TCP]
  [Coloring Rule String: tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive && !tcp.analysis.keep_alive_ack]

```

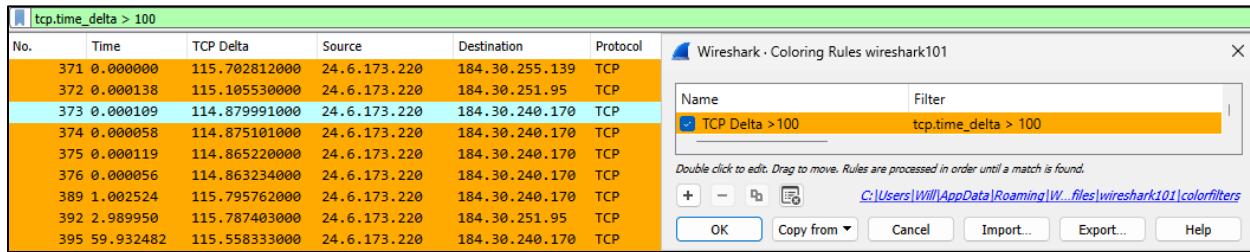
Frame 170 matches the Bad TCP coloring rule.

4-2. Temporarily color TCP stream 5 with a light blue background and apply a filter on this traffic. How many packets match your filter?

tcp.stream == 5					
No.	Time	Source	Destination	Protocol	Info
20	0.000000	24.6.173.220	184.30.240.170	TCP	29360 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
42	0.031452	184.30.240.170	24.6.173.220	TCP	80 → 29360 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=4
44	0.000122	24.6.173.220	184.30.240.170	TCP	29360 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
46	0.000693	24.6.173.220	184.30.240.170	HTTP	GET /swa/c/sitecopy_setup_L3.css HTTP/1.1
81	0.037860	184.30.240.170	24.6.173.220	TCP	80 → 29360 [ACK] Seq=1 Ack=597 Win=15792 Len=0
83	0.011778	184.30.240.170	24.6.173.220	HTTP	HTTP/1.1 200 OK (text/css)
166	0.194656	24.6.173.220	184.30.240.170	TCP	29360 → 80 [ACK] Seq=597 Ack=617 Win=65084 Len=0
298	1.304084	24.6.173.220	184.30.240.170	HTTP	GET /web/fw/j/mtagconfig-2011-08.js HTTP/1.1
322	0.027597	184.30.240.170	24.6.173.220	HTTP	HTTP/1.1 200 OK (application/x-javascript)
330	0.200456	24.6.173.220	184.30.240.170	TCP	29360 → 80 [ACK] Seq=1785 Ack=1956 Win=65700 Len=0
373	114.879991	24.6.173.220	184.30.240.170	TCP	29360 → 80 [FIN, ACK] Seq=1785 Ack=1956 Win=65700 Len=0
381	0.030611	184.30.240.170	24.6.173.220	TCP	80 → 29360 [FIN, ACK] Seq=1956 Ack=1786 Win=18168 Len=0
384	0.000195	24.6.173.220	184.30.240.170	TCP	29360 → 80 [ACK] Seq=1786 Ack=1957 Win=65700 Len=0

13 packets match the `tcp.stream == 5` display filter.

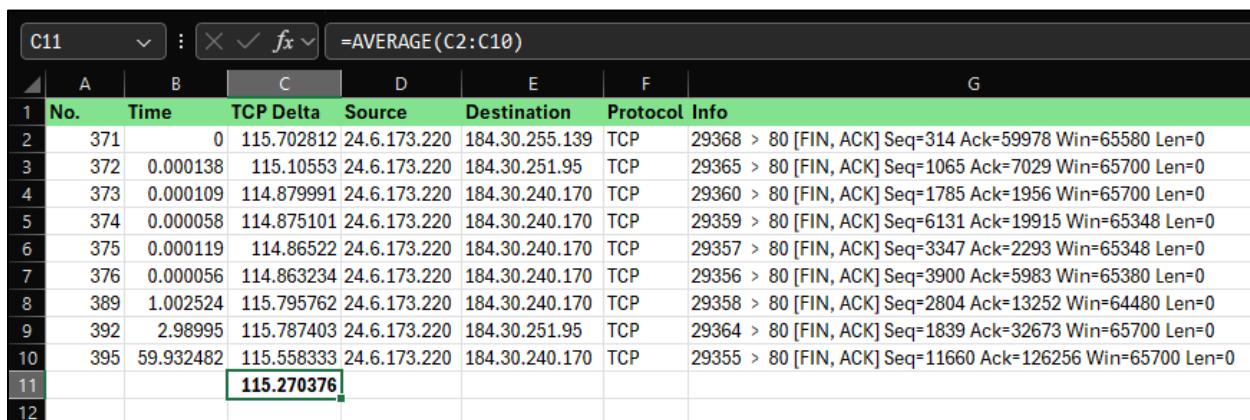
4-3. Create and apply a coloring rule for TCP delta delays greater than 100 seconds. How many frames match this coloring rule?



The Wireshark interface is shown with a list of TCP frames. The first 9 frames (371 to 379) have a yellow background, indicating they match the coloring rule. The 10th frame (395) has a white background. To the right, the 'Coloring Rules' dialog is open, showing a single rule named 'TCP Delta > 100' with the filter 'tcp.time_delta > 100'.

There are 9 matches for this rule. In frame 373, the background color is overridden by the colorized conversation from the previous question.

4-4. Export this filtered TCP delta information in CSV format. Using Excel, what is the average TCP delta time? Screenshot your Excel worksheet.

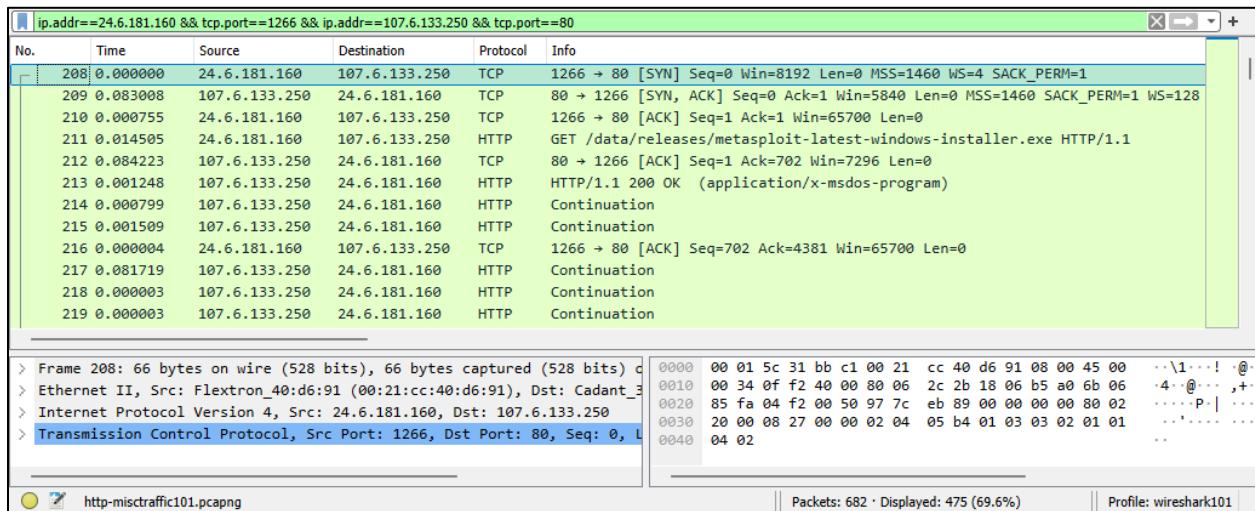


An Excel spreadsheet is shown with data from the Wireshark capture. The columns are labeled: No., Time, TCP Delta, Source, Destination, Protocol, and Info. The 'TCP Delta' column is highlighted in green. The cell C11 contains the formula '=AVERAGE(C2:C10)'. The value 115.270376 is highlighted in green in cell C11.

Using Excel, I found that the average TCP delta time in the export is 115.270376 seconds.

LAB 31: FILTER ON MOST ACTIVE TCP CONVERSATION

i. Follow all the steps in the lab and provide a screenshot of step 5.

**LAB 32: SET UP GEOIP TO MAP TARGETS GLOBALLY**

i. Follow all the steps in the lab and provide a screenshot of step 4.

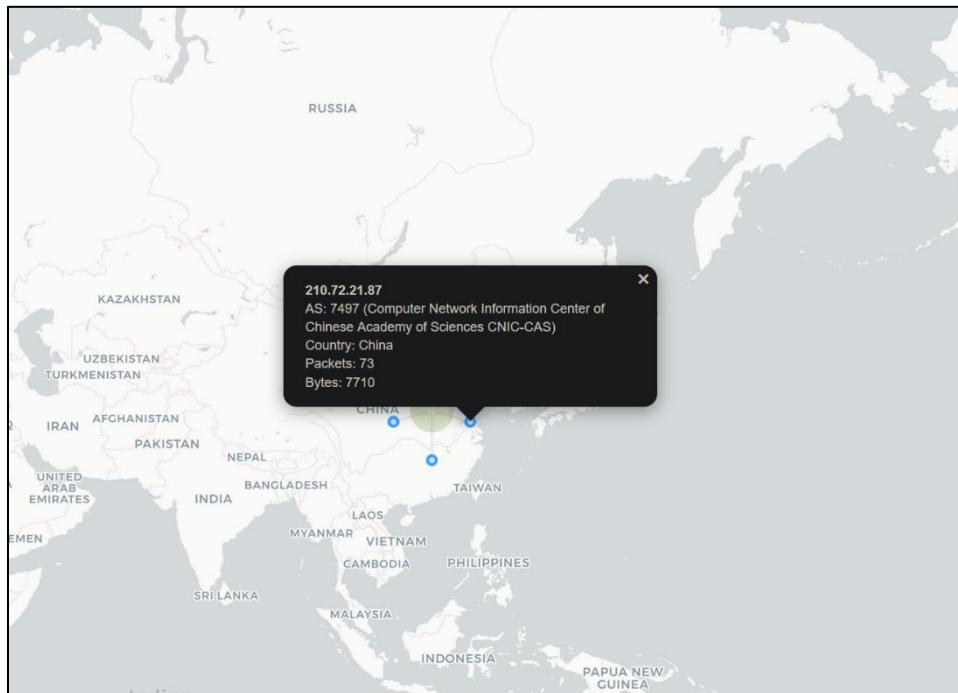
Endpoints - http-browse101c.pcapng										
	Ethernet · 2	IPv4 · 11	IPv6	TCP · 80	UDP · 68					
Address	Address	Address	Address	Address	Address	Address	Address	Address	Address	Address
24.6.173.220	1,668	799 k		742	97 k	926	701 k	United States	Livermore	7922
50.23.252.178	63	52 k		42	50 k	21	1932	United States	Dallas	—
75.75.75.75	152	20 k		76	14 k	76	5915	United States	Roseville	7922
123.125.115.126	82	14 k		33	9223	49	4944	China	Beijing	4808
173.194.79.121	10	2024		4	1366	6	658	United States	—	15169
202.96.25.95	72	9940		30	6780	42	3160	China	Beijing	4808
209.177.86.18	982	655 k		611	589 k	371	65 k	United States	—	21859
210.72.21.11	64	10 k		28	7191	36	3455	China	—	7497
210.72.21.12	99	19 k		42	13 k	57	5468	China	—	7497
210.72.21.42	71	7391		29	4145	42	3246	China	—	7497

Name resolution Limit to display filter Endpoint Types▼

Copy Map Close Help

ii. Follow the steps and provide a screenshot of step 5 (any plot point).

Note: one may have to change permission of the file to open (`sudo chmod 775`).



LAB 33: DETECT SUSPICIOUS PROTOCOLS OR APPLICATIONS

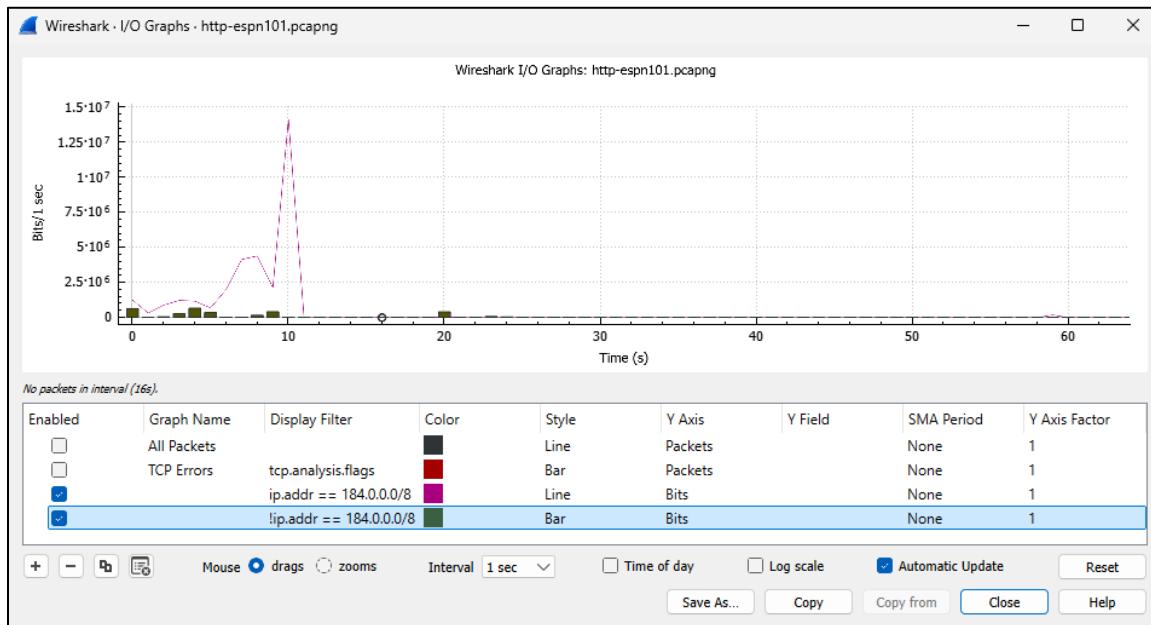
i. Follow all the steps in the lab and provide a screenshot of step 3.

No.	Time	Source	Destination	Protocol	Info
566	0.000000	24.6.173.220	67.220.66.111	IRC	Request (CAP)
569	0.028091	67.220.66.111	24.6.173.220	IRC	Response (NOTICE) (NOTICE)
570	0.000129	24.6.173.220	67.220.66.111	IRC	Request (NICK) (USER)
579	0.078260	67.220.66.111	24.6.173.220	IRC	Response (NOTICE)
581	0.073585	67.220.66.111	24.6.173.220	IRC	Response (NOTICE) (CAP)
583	0.006958	24.6.173.220	67.220.66.111	IRC	Request (CAP)
584	0.038451	67.220.66.111	24.6.173.220	IRC	Response (CAP)
585	0.000348	24.6.173.220	67.220.66.111	IRC	Request (CAP)
586	0.026163	67.220.66.111	24.6.173.220	IRC	Response (PING)
587	0.000459	24.6.173.220	67.220.66.111	IRC	Request (PONG)
588	0.027775	67.220.66.111	24.6.173.220	IRC	Response (001) (002) (003) (004) (005) (006) (042) (251) (252)
589	0.001404	67.220.66.111	24.6.173.220	IRC	Response (ne) (254) (255) (265) (266) (250) (375) (372) (372)

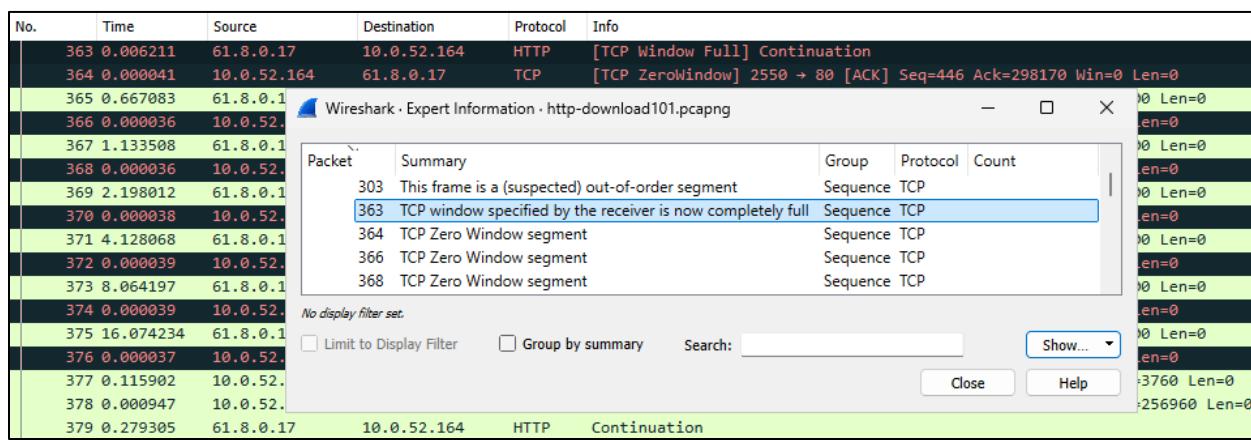
> Frame 588: 1078 bytes on wire (8624 bits), 1078 bytes captured (8624 bits) on interface unknown, id 0
 > Ethernet II, Src: Cadant_31:bb:c1 (00:01:5c:31:bb:c1), Dst: HewlettP_a7:bf:a3 (d4:85:64:a7:bf:a3)
 > Internet Protocol Version 4, Src: 67.220.66.111, Dst: 24.6.173.220
 > Transmission Control Protocol, Src Port: 6667, Dst Port: 30209, Seq: 360, Ack: 90, Len: 1024
 > Internet Relay Chat
 > Response: :bartholomew.2600.net 001 mregion :Welcome to the 2600net Internet Relay Chat Network mregion
 > Response: :bartholomew.2600.net 002 mregion :Your host is bartholomew.2600.net[67.220.66.111/6667], running version hybrid-7.2.2 D²Gabcdfgiklnorsuvwxyz biklmnopstveIh bklov
 > Response: :bartholomew.2600.net 004 mregion bartholomew.2600.net 005 mregion CALLEDID CASEMAPPING=rfc1459 DEAF=D KICKLEN=160 MODES=4 NICKLEN=25 CHANNELLEN=50 CHANNELNAME=bartholomew.2600.net 006 mregion CHANLIMIT=25 CHANNELNAME=bartholomew.2600.net 007 mregion CHANMODES=eIb,k,l,imnpst AWAYLEN=160 KNOCK EU
 > Response: :bartholomew.2600.net 042 mregion 3B0AADHT :your unique ID
 > Response: :bartholomew.2600.net 251 mregion :There are 8 users and 462 invisible on 7 servers
 > Response: :bartholomew.2600.net 252 mregion 15 :IRC Operators onli

LAB 34: COMPARE TRAFFIC TO/FROM A SUBNET TO OTHER TRAFFIC

i. Follow all the steps in the lab and provide a screenshot of step 5.

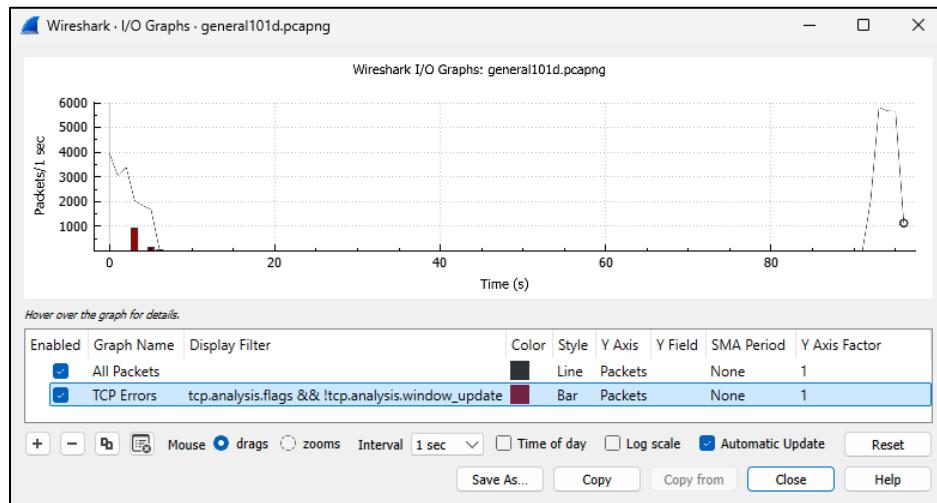
**LAB 35: IDENTIFY AN OVERLOADED CLIENT**

i. Follow all the steps in the lab and provide a screenshot of step 4.

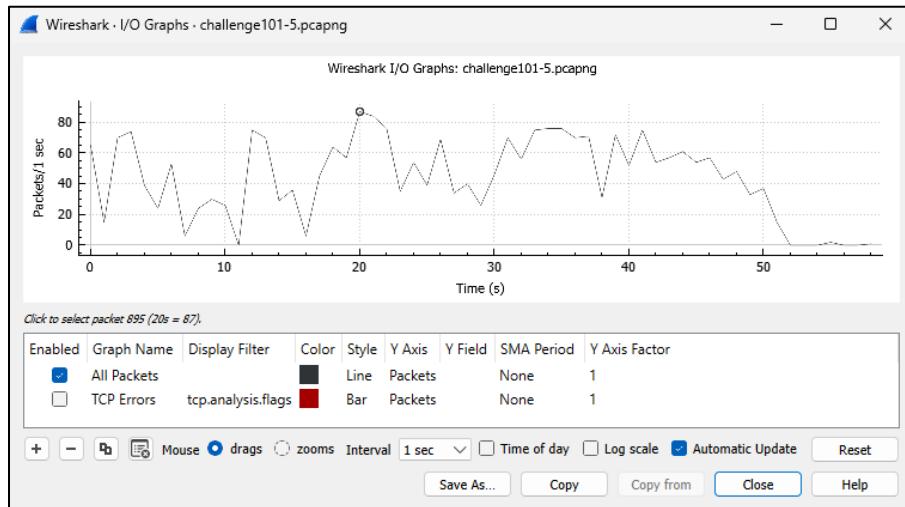


LAB 36: DETECT AND GRAPH FILE TRANSFER PROBLEMS

i. Follow all the steps in the lab and provide a screenshot of step 5.

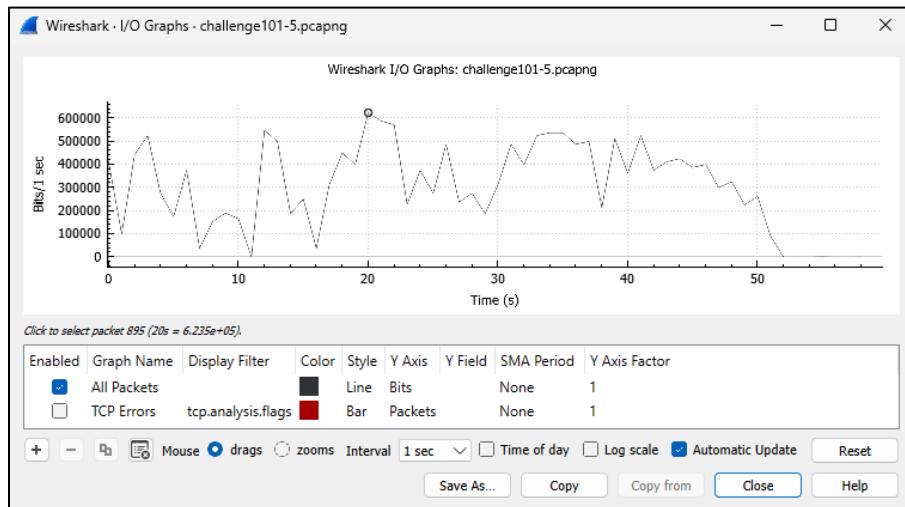
**CHAPTER 5 CHALLENGE**

5-1. Create an IO graph for this trace file. What is the highest packets-per-second value seen in this trace file?



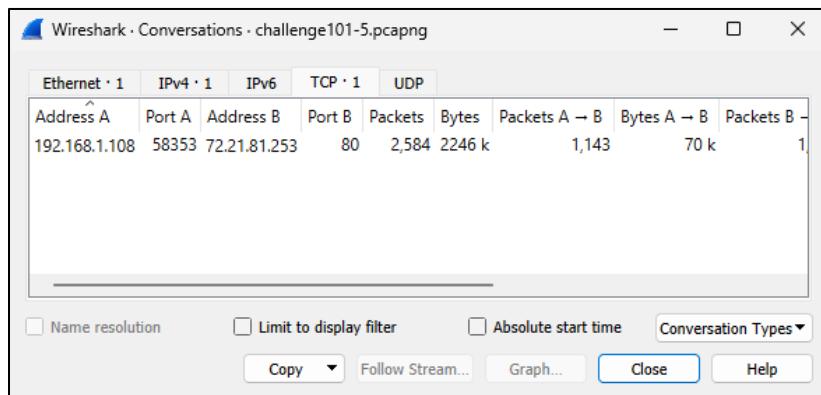
The highest packets-per-second value is about 87.

5-2. What is the highest bits-per-second value seen in this trace file?



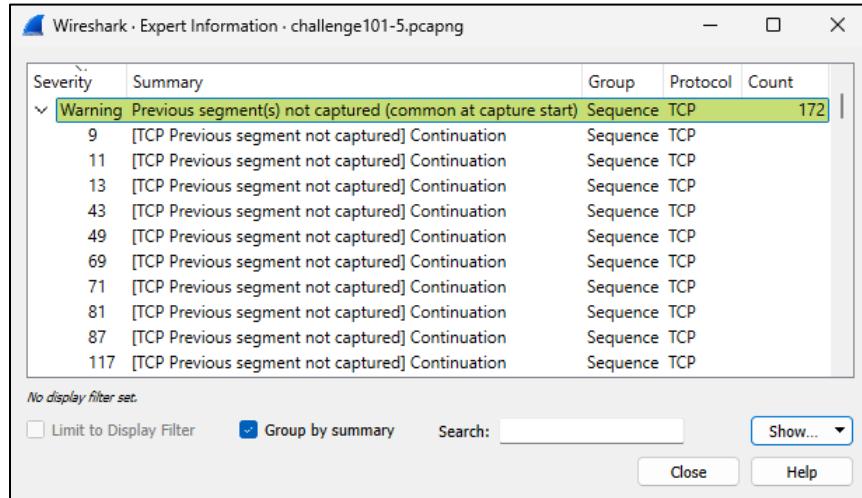
Changing the Y axis to Bits/s, the highest value is about $6.235e^5$ (623,500 bits-per-second).

5-3. How many TCP conversations are in this trace file?



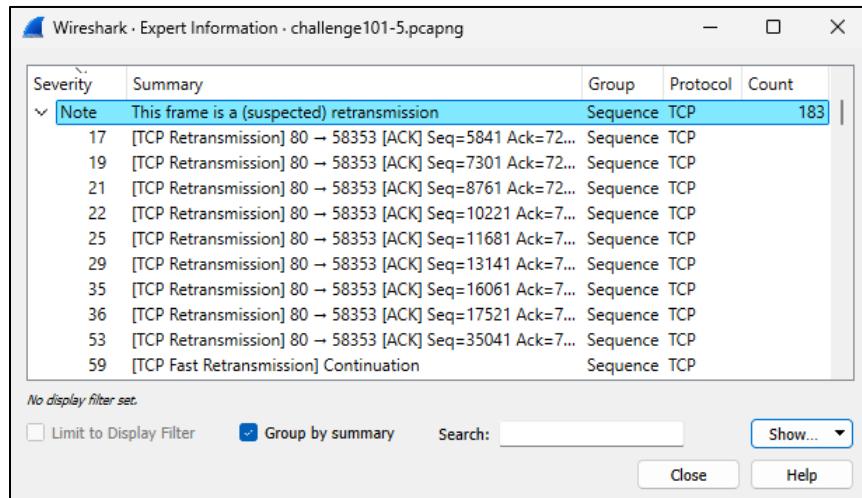
There is one TCP conversation in this trace file.

5-4. How many times has “Previous segment not captured” been detected in this trace file?



The “Previous segment(s) not captured” warning has been detected 172 times in this trace file.

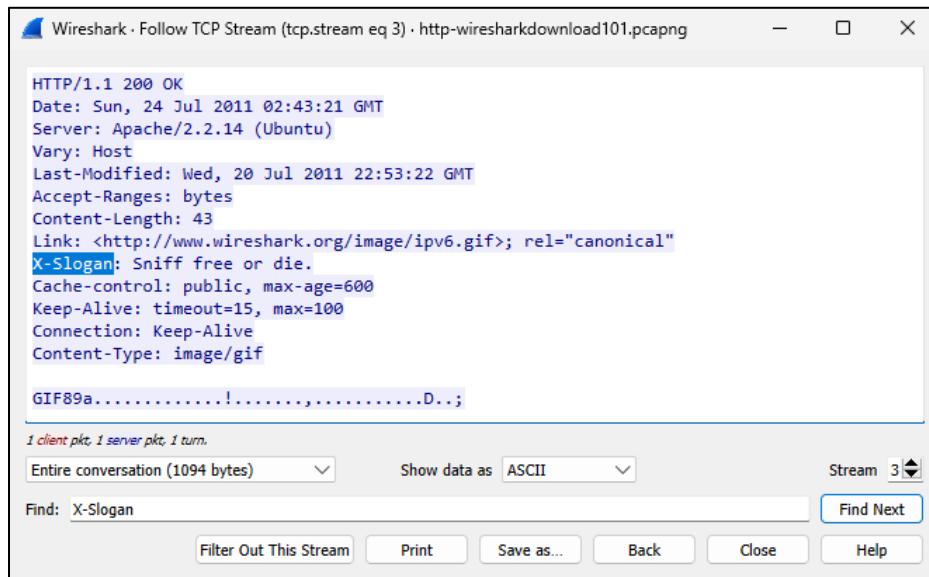
5-5. How many retransmissions and fast retransmissions are seen in this trace file?



There are 183 counts of the “(suspected) retransmission” note. After expanding the note, we can see that this number includes fast retransmissions.

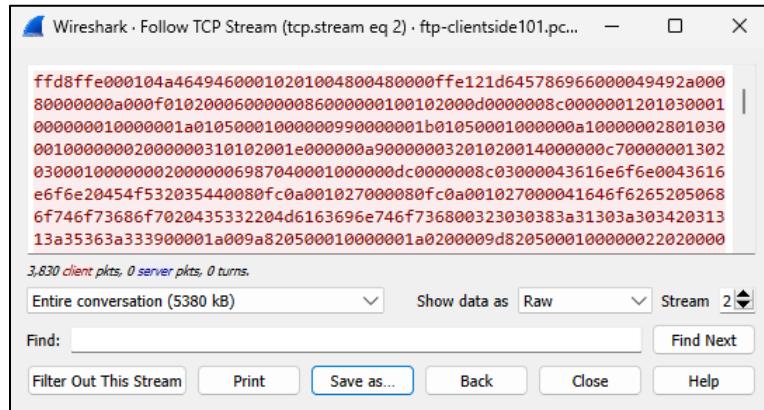
LAB 37: USE REASSEMBLY TO FIND A WEB SITE'S HIDDEN HTTP MESSAGE

i. Follow all the steps in the lab and provide a screenshot of step 5.



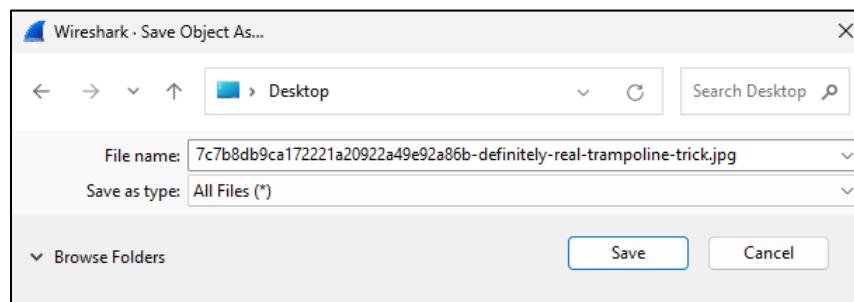
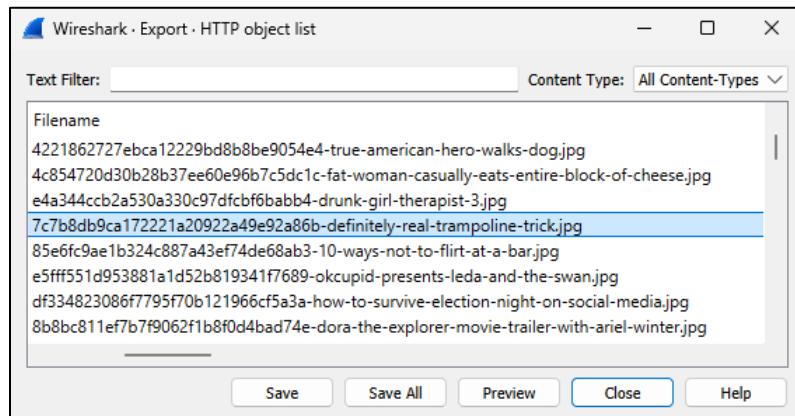
LAB 38: EXTRACT A FILE FROM AN FTP TRANSFER

i. Follow all the steps in the lab and provide screenshots of steps 7 and 9.

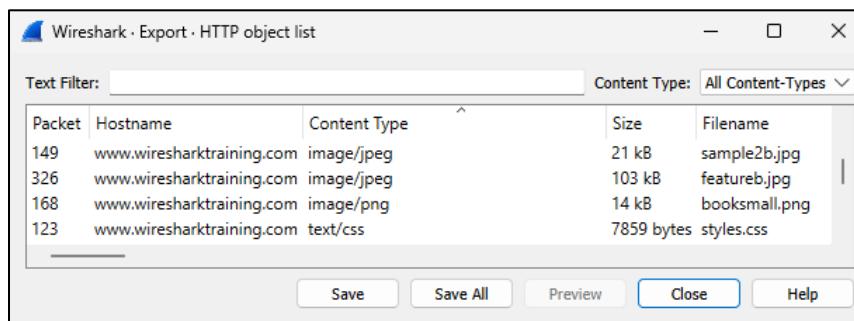


LAB 39: CARVE OUT A HTTP OBJECT FROM A WEB BROWSING SESSION

i. Follow all the steps in the lab and provide screenshots of steps 4 and 5.

**CHAPTER 6 CHALLENGE**

6-1. What two `.jpg` files can be exported from this trace file?



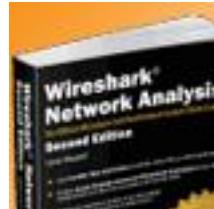
[**sample2b.jpg**](#) and [**featureb.jpg**](#) can be exported from the trace file.

6-2. On what HTTP server and in what directory does `next-active.png` reside?

```
> Frame 1214: 729 bytes on wire (5832 bits), 729 bytes captured (5832 bits) on interface \Device\NPF_{6E79FEC0-F
> Ethernet II, Src: Cadant_31:bb:c1 (00:01:5c:31:bb:c1), Dst: HewlettP_a7:bf:a3 (d4:85:64:a7:bf:a3)
> Internet Protocol Version 4, Src: 50.16.207.192, Dst: 24.6.173.220
> Transmission Control Protocol, Src Port: 80, Dst Port: 18459, Seq: 1, Ack: 756, Len: 675
< Hypertext Transfer Protocol
  > HTTP/1.1 301 Moved Permanently\r\n
    Date: Sun, 11 Nov 2012 04:00:44 GMT\r\n
    Server: Apache/2.2.23 (Amazon)\r\n
    Location: http://www.arbornetworks.com/modules/mod_arborslideshow/tmp/img/icon/slider/next-active.png\r\n
  > Content-Length: 383\r\n
  > Connection: close\r\n
  > Content-Type: text/html; charset=iso-8859-1\r\n
  > \r\n
  > [HTTP response 1/1]
  > [Time since request: 0.118313000 seconds]
  > [Request in frame: 1176]
  > [Request URI: http://arbornetworks.com/modules/mod_arborslideshow/tmp/img/icon/slider/next-active.png]
  > File Data: 383 bytes
  > Line-based text data: text/html (9 lines)
```

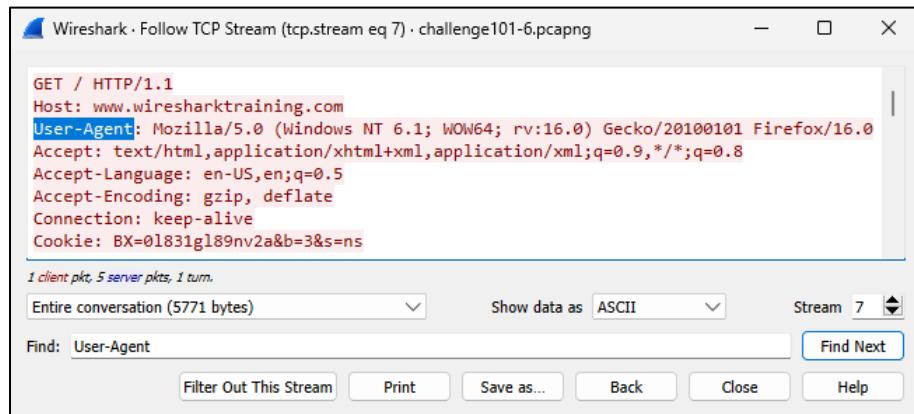
[www.arbornetworks.com](http://www.arbornetworks.com/modules/mod_arborslideshow/tmp/img/icon/slider/) in the [/modules/mod_arborslideshow/tmp/img/icon/slider/](http://www.arbornetworks.com/modules/mod_arborslideshow/tmp/img/icon/slider/) directory.

6-3. Export `booksmall.png` from this trace file. What is in the image?



[It's a low-resolution image of part of the book.](#)

6-4. Reassemble TCP stream 7. What type of browser is the client using in this stream?



Based on the value for **User-Agent**, a Firefox browser was used.

LAB 40: READ ANALYSIS NOTES IN A MALICIOUS REDIRECTION TRACE FILE

i. Follow all the steps in the lab and provide a complete screenshot of step 3.

Comment	Packet comments listed below.	Comment	Frame	19
1	This is the original search query for the "Peter Lik for sale" ...	Comment	Frame	
5	In this response, the server sends numerous thumbnail im...	Comment	Frame	
7	Now we clicked on the image load the expanded thumbna...	Comment	Frame	
12	We get the expanded image through Google - there are a ...	Comment	Frame	
14	We clicked on the web link associated with the expanded i...	Comment	Frame	
15	Here we begin connecting to www.artbrokerage.com at 66...	Comment	Frame	
18	We request an 850x600 size of a Peter Lik photo.	Comment	Frame	
21	Now we are making a request to www.ulisseide.org.	Comment	Frame	
23	This TCP connection is used to get the image file from arb...	Comment	Frame	
67	Here's the redirection to the malicious site. See the Locatio...	Comment	Frame	
68	We removed the DNS queries from the trace file - we must...	Comment	Frame	
75	Our malicious host is redirecting us to run a CGI script (in.c...	Comment	Frame	
79	And here we go... this is the ugly connection.	Comment	Frame	
84	Please oh please hit us over the head with a baseball bat! ...	Comment	Frame	
87	They're dropping a cookie on our drive and giving us a lin...	Comment	Frame	
96	Well that didn't go so well for them... our Symantec softwa...	Comment	Frame	
104	And another termination triggered by Symantec.	Comment	Frame	
117	Yes, Symantec is screaming with messages on our system...	Comment	Frame	
159	We're just returning to Google after a little sidetrack to the...	Comment	Frame	

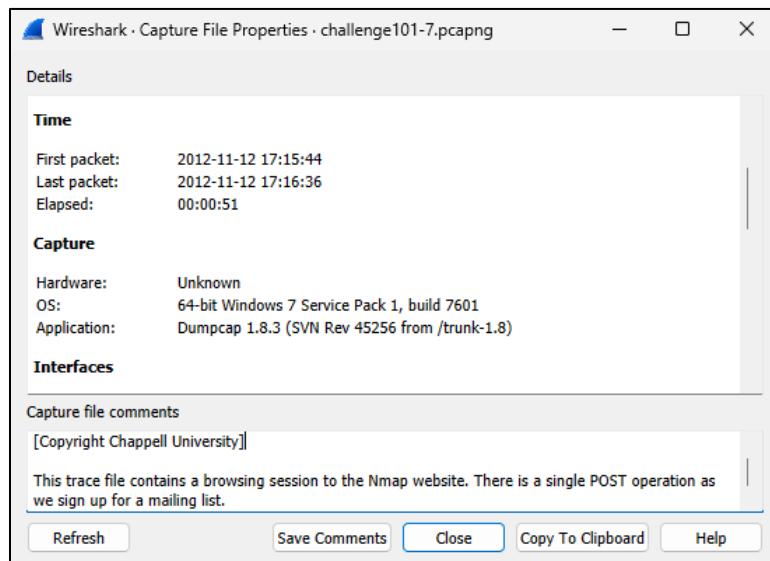
LAB 41: EXPORT MALICIOUS REDIRECTION PACKET COMMENTS

i. Follow all the steps in the lab and provide a screenshot of step 6.

No.	Time	Source	Destination	Host	Protocol	Coloring Rule Name	Comment	Info
1	0 24.6.173.220	74.125.224.84			HTTP		This is the original search query for the "Peter Lik for sale" images.	GET /sbd?q=peter+lik+for+sale&um=1&hl=en&client=firefox-a&sa=N&rls
2	0.062672 74.125.224.84	24.6.173.220			HTTP		In this response, the server sends numerous thumbnail images along with Continuation	
3	0.47305 24.6.173.220	74.125.224.84			HTTP		Now we clicked on the image link and the expanded thumbnail from Google. GET /imgres?imgurl=http://www.artbrokerage.com/artthumb/lkip_35911	
4	0.47305 24.6.173.220	74.125.224.84			HTTP		We get the expanded image through Google - there are a lot of web results.	GET /sbd?q=peter+lik+for+sale&um=1&hl=en&client=firefox-a&sa=N&rls
5	0.4843454 74.125.224.84	24.6.173.220			HTTP		We clicked on the web link associated with the expanded image. This is a continuation of the previous request.	GET /sbd?q=peter+lik+for+sale&um=1&hl=en&client=firefox-a&sa=N&rls
6	0.024838 24.6.173.220	77.93.251.49			TCP		Here we begin connecting to www.artbrokerage.com at 66.11.147.48. The 50319 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1	
7	0.002104 24.6.173.220	66.11.147.48			TCP		We request an 850x600 size of a Peter Lik photo.	GET /artthumb/lkip_35911_2_850x600/Peter_Lik_Beyond_Paradise.jpg HTTP/1.1
8	0.030955 24.6.173.220	66.11.147.48		www.artbrokerage.com	HTTP		Now we are making a request to www.ulisseide.org.	GET /stat/gthyu/index.php?>=peter+lik+inner+peace+for+sale HTTP/1.1
9	0.086025 24.6.173.220	77.93.251.49		www.ulisseide.org	HTTP		This TCP connection is used to get the image file from artbrokerage.com. HTTP/1.1 200 OK [BoundErrorUnreassembled Packet]	
10	0.161477 66.11.147.48	24.6.173.220			HTTP		Here's the redirection to the malicious site. See the Location line. We are HTTP/1.1 302 Found	
11	0.580651 77.93.251.49	24.6.173.220			HTTP		We removed the DNS queries from the trace file - we must have looked up 50319 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1	
12	0.02217 24.6.173.220	95.169.190.217			TCP		Our malicious host is redirecting us to run a CGI script (in.cgi). We'll have HTTP/1.1 302 Found	
13	0.382179 95.169.190.217	24.6.173.220			HTTP		And here we go... this is the ugly connection.	50320 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
14	0.003645 24.6.173.220	95.169.190.217			TCP		Please oh please hit us over the head with a baseball bat! We ask for the Continuation	
15	0.196512 24.6.173.220	95.169.190.217			HTTP		They're dropping a cookie on our drive and giving us a link to a .info site! HTTP/1.1 200 OK (text/html)	
16	0.210788 95.169.190.217	24.6.173.220			HTTP		Well that didn't go so well for them... our Symantec software terminated it.	50321 > 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	0.24408 24.6.173.220	78.41.203.19			TCP RST		And another termination triggered by Symantec.	50322 > 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
18	0.181295 24.6.173.220	78.41.203.19			TCP RST		Yes, Symantec is screaming with messages on our system...	50323 > 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	0.283123 24.6.173.220	78.41.203.19			TCP RST		We're just returning to Google after a little sidebar to the dark side...	GET /gen_204?&typ=&ct=background&cd=&ei=ejsdTsWPN4OmsQOf09W
20	12.732243 24.6.173.220	74.125.224.84	www.google.com		HTTP			

CHAPTER 7 CHALLENGE

7-1. What information is contained in the trace file annotation?



A comment describes what's happening in the trace file. Other information (such as the capture time, interface details, and file name/size) is also available under the Details section.

7-2. What packet comments are contained in this trace file?

The screenshot shows the 'Expert Information' dialog in Wireshark. The table lists various types of expert comments and their counts. The three comments related to this question are:

Severity	Summary	Group	Protocol	Count
> Warning	Malformed JFIF (JPEG) image			1
> Warning	Protocol LLMNR			2
> Warning	Sequence TCP			9
> Warning	Protocol DNS			4
> Note	Sequence TCP			33
> Chat	Sequence HTTP			144
> Chat	Sequence TCP			92
Comment	Comment Frame	Comment	Frame	3
	136 The Nmap page uses Google analytics to track visitor infor...	Comment	Frame	
	231 This 304 Not Modified response indicates that the client h...	Comment	Frame	
	738 This png file is located in the zenmap/images directory.	Comment	Frame	

There are three packet comments in the file (on frames 136, 231, and 738).

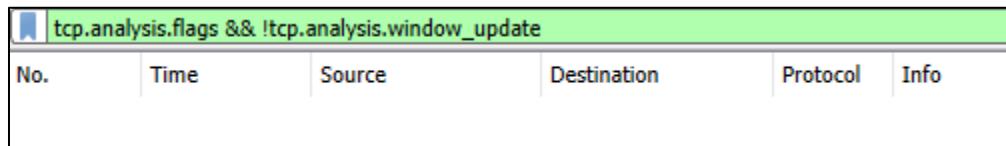
7-3. Add a comment to the POST message in this trace file. What packet did you alter?

The screenshot shows the Wireshark packet list with a comment dialog open over it. The comment dialog is titled 'Wireshark - Add Packet Comment' and contains the text 'POST packet (938).'. The packet list shows several TCP and HTTP packets, with packet 938 highlighted as the target for the comment.

The packet with the POST message was number 938.

LAB 42: SPLIT A FILE AND WORK WITH FILTERED FILE SETS

i. Follow all the steps in the lab and provide a screenshot of step 6.

**LAB 43: MERGE A SET OF FILES USING A WILDCARD**

i. Follow all the steps in the lab and provide a screenshot of step 3.

```
PS C:\Cyber\Traces\409> mergecap -w .\http-downloadaset.pcapng .\wk_7_WS_http-download-a20000*.*  
PS C:\Cyber\Traces\409> dir .\http-downloadaset.pcapng  
Directory: C:\Cyber\Traces\409  
  
Mode LastWriteTime Length Name  
---- ----- ----- ----  
-a--- 10/21/2024 12:46 PM 172113216 http-downloadaset.pcapng
```

LAB 44: USE TSHARK TO CAPTURE TO FILE SETS WITH AN AUTOSTOP CONDITION

i. Follow all the steps in the lab and provide a screenshot of step 5.

```
PS C:\Cyber\Traces\409> dir mytshark*.*  
Directory: C:\Cyber\Traces\409  
  
Mode LastWriteTime Length Name  
---- ----- ----- ----  
-a--- 10/21/2024 12:53 PM 4399160 mytshark_00001_20241021125230.pcapng  
-a--- 10/21/2024 12:53 PM 93924 mytshark_00002_20241021125301.pcapng  
-a--- 10/21/2024 12:54 PM 119432 mytshark_00003_20241021125331.pcapng  
-a--- 10/21/2024 12:54 PM 1379256 mytshark_00004_20241021125401.pcapng  
-a--- 10/21/2024 12:55 PM 624504 mytshark_00005_20241021125432.pcapng  
-a--- 10/21/2024 12:55 PM 51508 mytshark_00006_20241021125502.pcapng
```

LAB 45: USE TSHARK TO EXTRACT HTTP GET REQUESTS

i. Create an Excel export of `stats.txt` file and provide a sorted screenshot of results (part 8.6).

Protocol	Frames	Bytes
eth	1493	804985
ip	1469	803375
tcp	1095	694222
tls	597	631808
udp	374	109153
dns	212	25484
quic	151	80979
quic	24	23323
http	22	13383
data	19	1140
ocsp	16	11326
data	15	9861
data	5	1672
ssdp	4	848
data-text-lines	3	892
ipv6	3	350
udp	3	350
mdns	2	210
mdns	2	140
arp	2	120
snmp	1	140

I'm not sure if I understood the instructions correctly, but I used the data in the `stats.txt` file to create a table in Excel. The values are sorted by frames first and then by bytes (both from largest to smallest), but this kind of makes the hierarchy confusing at-a-glance. I looked for ways to sort the values while still retaining a hierarchy (like you can do in the hierarchy statistics window natively within Wireshark) but couldn't find a reasonable solution.

LAB 46: USE TSHARK TO EXTRACT HTTP HOST NAMES AND IP ADDRESSES

i. Follow all the steps in the lab and provide a screenshot of step 4.

http://hostaddrs.txt	
File	Edit
detectportal.firefox.com,34.107.221.82 detectportal.firefox.com,34.107.221.82 r10.o.lencr.org,104.123.153.168 r11.o.lencr.org,23.64.114.214 r11.o.lencr.org,23.64.114.214 detectportal.firefox.com,34.107.221.82 ocsp.digicert.com,192.229.211.108 r10.o.lencr.org,104.123.153.168 r10.o.lencr.org,104.123.153.168 r10.o.lencr.org,104.123.153.168 o.pki.goog,142.250.191.195	

CHAPTER 8 CHALLENGE

8-1. What Tshark parameters should you use to list active interfaces on your WS system?

```
PS C:\Cyber\Traces\409> tshark -D
1. \Device\NPF_{CE9E8B45-60A6-46B0-B523-7E3072D62905} (Local Area Connection* 9)
2. \Device\NPF_{2BAD2549-99CA-498D-8220-9202232639FC} (Local Area Connection* 8)
3. \Device\NPF_{EAE66B2D-B099-448F-8520-FC9300CFA55A} (Local Area Connection* 7)
4. \Device\NPF_{8E3FC44C-D55C-423D-BD17-5E508F5CF467} (vEthernet (WSL (Hyper-V firewall)))
5. \Device\NPF_{D10D470F-050E-4C77-9C4E-02D52597B5F0} (vEthernet (Default Switch))
6. \Device\NPF_{5F2CE87F-5C16-4A47-A31F-E87B87533201} (Bluetooth Network Connection 0)
7. \Device\NPF_{91F6BFCD-BC7E-45B7-B07D-53508C11FCB3} (VMware Network Adapter VMnet8)
8. \Device\NPF_{FC141F81-0CF6-4626-AB64-F3ED0F2B888B} (VMware Network Adapter VMnet1)
9. \Device\NPF_{153ADED8-F651-4BBB-BBF9-6A28FCC05897} (Local Area Connection* 12)
10. \Device\NPF_{7912AAA7-736F-48E4-9F15-B8A463CD2AF3} (Local Area Connection* 11)
11. \Device\NPF_{DA81636E-5594-48E4-9CFE-DB82B8396CC5} (Wi-Fi 0)
12. \Device\NPF_{E03B19B5-CDB4-410E-B9B9-764429E76425} (Ethernet 0)
13. \Device\NPF_Loopback (Adapter for loopback traffic capture)
14. \Device\NPF_{CEE937BE-027F-4458-B9CC-27C32E9C861C} (Ethernet 1)
```

You can use `tshark -D` to get a list of active interfaces.

8-2. Using Tshark to extract protocol hierarchy information, how many UDP frames are in `challenge101-8.pcapng`?

```
PS C:\Cyber\Traces\409> tshark -r .\challenge101-8.pcapng -qz io,phs | Select-String udp
                           udp
                           frames:62 bytes:8074
PS C:\Cyber\Traces\409>
```

There are 62 UDP frames in `challenge101-8.pcapng`.

8-3. Use Tshark to export all DNS Packets from `challenge101-8.pcapng` to a new trace file called `ch8dns.pcapng`. How many packets were exported?

```
PS C:\Cyber\Traces\409> tshark -r ".\challenge101-8.pcapng" -Y "dns" -w "ch8dns.pcapng"
PS C:\Cyber\Traces\409> capinfos .\ch8dns.pcapng | Select-String packets
Number of packets: 62
Average packet rate: 3 packets/s
Number of packets = 62
```

Adapting a command from lab 45 to this context, 62 DNS packets were exported from the `challenge101-8.pcapng` capture to `ch8dns.pcapng`.

References

Fielding, R., Nottingham, M., & Reschke, J. (2022). *HTTP semantics* (RFC No. 9110). RFC Editor. <https://doi.org/10.17487/RFC9110>

Hyper-V virtual switch. (2021, July 29). Microsoft Learn. <https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v-virtual-switch/hyper-v-virtual-switch>

IANA. (2022, June 8). *Hypertext transfer protocol (HTTP) status code registry*. Internet Assigned Numbers Authority. <https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml>

Johnson, E. (2019, May 23). *UDP and TCP protocols*. Knight Foundation School of Computing and Information Sciences. <https://users.cs.fiu.edu/~esj/cgs4285/class13.html>

MartinGarcia, L., & Lyon, G. (n.d.). *Nping reference guide*. Nmap Network Scanning. Retrieved September 17, 2024, from <https://nmap.org/book/nping-man.html>

Mittal, A. (2014, March 29). *Answer to “Why do I have 55 local area connections in ipconfig?”* [Online post]. Super User. <https://superuser.com/a/735043>

Mozilla Developer Network. (2024, July 25). *HTTP response status codes*. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

Osterloh, H. (2002, May 1). *Network layer/internet protocols*. InformIT. <https://www.informit.com/articles/article.aspx?p=26557&seqNum=5>

Red Hat. (n.d.). *Disable source routing*. Red Hat Product Documentation. Retrieved September 17, 2024, from https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-server_security-disable-source-routing#sect-Security_Guide-Server_Security-Disable-Source-Routing

Rosenberg, B. (2003, March 14). *TCP in a nutshell*. University of Miami Computer Science.

https://www.cs.miami.edu/home/burt/learning/Csc524.032/notes/tcp_nutshell.html

Thomas, M. (2014, February 27). *IPv4 packet format*. University of Cincinnati Homepages.

https://homepages.uc.edu/~thomam/Net1/Packet_Formats/ip.html

VMware. (2019, May 31). *Understanding virtual networking components*. VMware Docs Home.

<https://docs.vmware.com/en/VMware-Workstation-Pro/17/com.vmware.ws.using.doc/GUID-8FDE7881-C31F-487F-BEF3-B2107A21D0CE.html>