

CLOUD AND NETWORK SECURITY

CHARLES GITHINJI

CS-CNS07-24006

Week 2: Assignment 3

Hack the Box: Intro to Network Traffic Analysis

INTRODUCTIONS

The goal of this lab is to understand network traffic analysis. I will explore the fundamental concepts of network traffic analysis, learn the basics of Tcpdump and Wireshark, and how to use various filters in these tools. Through hands-on experience with Tcpdump and Wireshark, I will gain the skills necessary to perform effective network traffic analysis. The specific objectives of this lab are;

This lab helps me learn how to implement network traffic analysis using Wireshark and Tcpdump.

1. Grasp the principles behind network analysis
2. Understand the key features of tcpdump

3. Learn to use Wireshark
4. Apply filters in Wireshark for more detailed analysis

Part 1: Network Traffic Analysis Introduction

Network Traffic Analysis (NTA) can be described as the act of examining network traffic to characterize common ports and protocols utilized, establish a baseline for our environment, monitor and respond to threats, and ensure the greatest possible insight into our organization's network.

Required Skills and Knowledge

1. TCP/IP Stack & OSI Model
2. Basic Network Concepts
3. Common Ports and Protocols
4. Concepts of IP Packets and the Sublayers
5. Protocol Transport Encapsulation

Environment and Equipment

Common Traffic Analysis Tools

Tool	Description
------	-------------

Tcpdump- *tcpdump* is a command-line utility that, with the aid of *LibPcap*, captures and interprets network traffic from a network interface or capture file.

Tshark- *TShark* is a network packet analyzer much like *TCPDump*. It will capture packets from a live network or read and decode from a file. It is the command-line variant of Wireshark.

Wireshark- Wireshark is a graphical network traffic analyzer. It captures and decodes frames off the wire and allows for an in-depth look into the environment. It can run many different dissectors against the traffic to characterize the protocols and applications and provide insight into what is happening.

NGrep- NGrep is a pattern-matching tool built to serve a similar function as *grep* for Linux distributions. The big difference is that it works with network traffic packets. NGrep understands how to read live traffic or traffic from a PCAP file and utilize regex expressions and BPF syntax. This tool shines best when used to debug traffic from protocols like HTTP and FTP.

Tcpick- *tcpick* is a command-line packet sniffer that specializes in tracking and reassembling TCP streams. The functionality to read a stream and reassemble it back to a file with *tcpick* is excellent.

Network Taps- Taps (Gigamon, Niagra-taps) are devices capable of taking copies of network traffic and sending them to another place for analysis. These can be in-line or out of band. They can actively capture and analyze the traffic directly or passively by putting the original packet back on the wire as if nothing had changed.

Networking Span Ports- Span Ports are a way to copy frames from layer two or three networking devices during egress or ingress processing and send them to a collection point. Often a port is mirrored to send those copies to a log server.

Elastic Stack- The Elastic Stack is a culmination of tools that can take data from many sources, ingest the data, and visualize it, to enable searching and analysis of it.

SIEMS- SIEMS (such as Splunk) are a central point in which data is analyzed and visualized. Alerting, forensic analysis, and day-to-day checks against the traffic are all use cases for a SIEM.

Performing Network Traffic Analysis

NTA Workflow



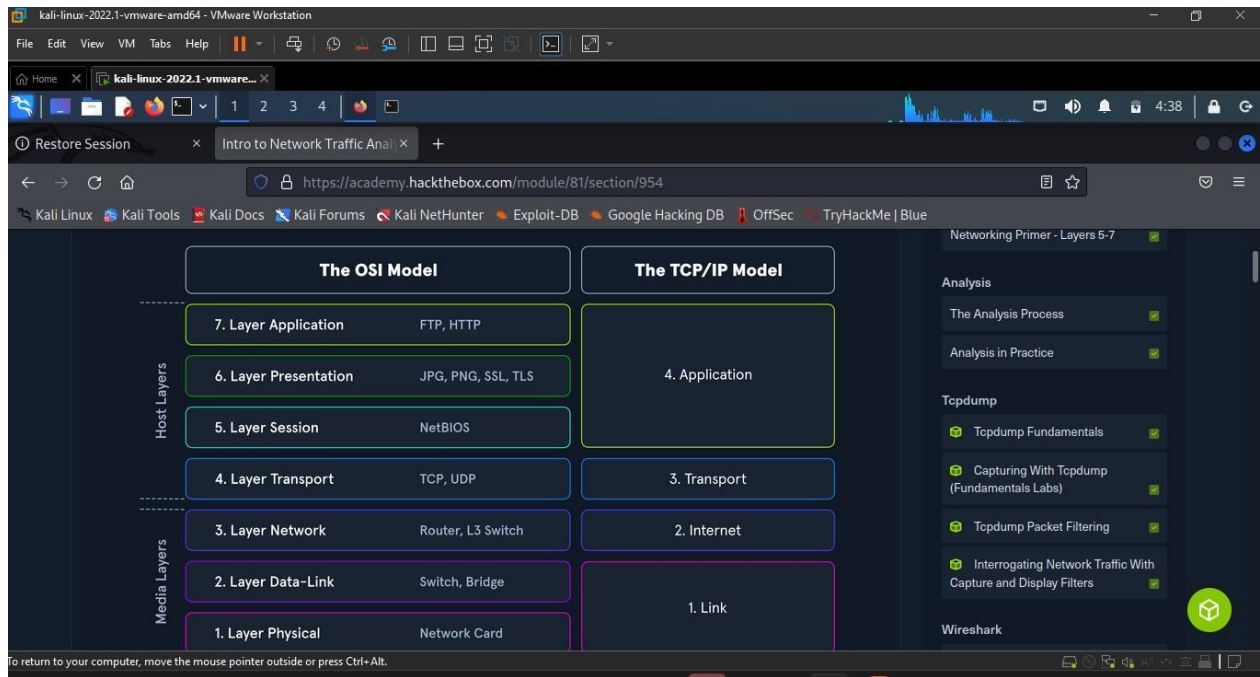
Traffic analysis, or NTA, is a method used to study and understand network traffic. It involves examining and analyzing the flow of data packets in a network to identify patterns, trends, and anomalies. NTA is not an exact science because it can be influenced by various factors, such as the goals of the analysis (whether it's detecting network errors or malicious actions) and the level of visibility into the network.

Networking Primer - Layers 1-4

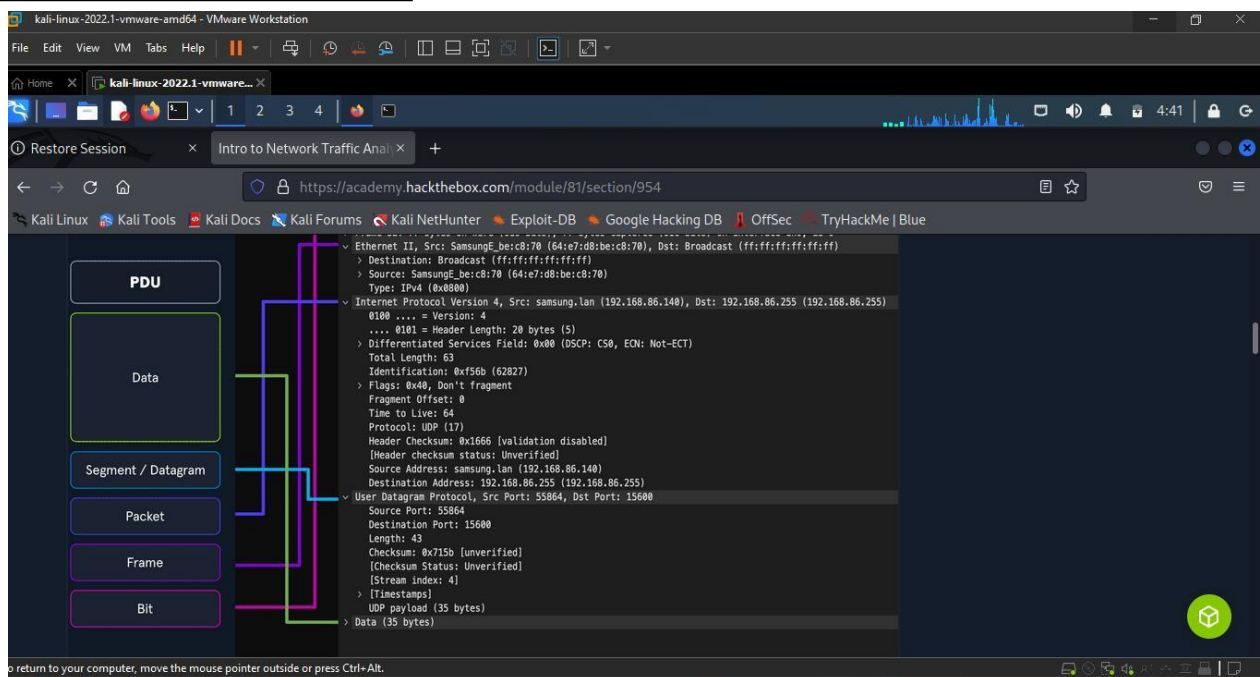
OSI / TCP-IP Models

The image above gives a great view of the Open Systems Interconnect (OSI) model and the Transmission Control Protocol - Internet Protocol (TCP-IP) model side by side.

PDU Example



PDU Packet Breakdown

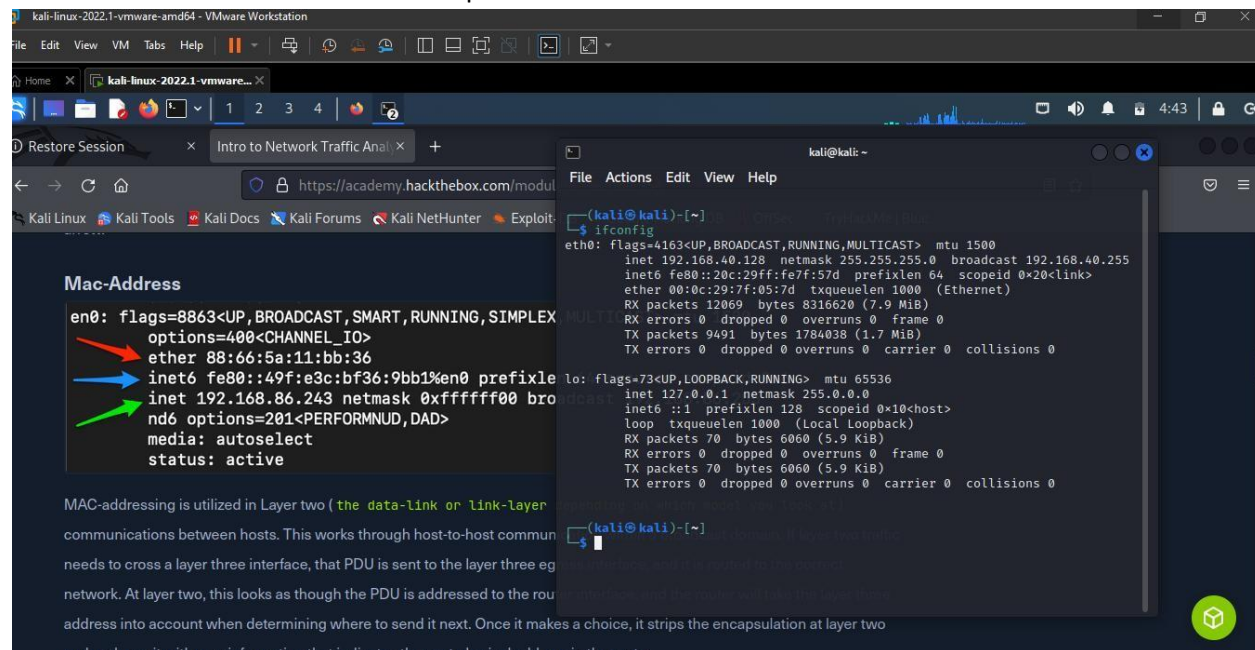


The image above shows the makeup of a PDU side by side with a packet breakout from Wireshark's Packet Details pane.

Addressing Mechanisms

MAC-Addressing

Each logical or physical interface attached to a host has a Media Access Control (MAC) address. This address is a 48-bit six octet address represented in hexadecimal format.



IP Addressing

The Internet Protocol (IP) was developed to deliver data from one host to another across network boundaries. IP is responsible for routing packets, the encapsulation of data, and fragmentation and reassembly of datagrams when they reach the destination host.

IPv4

The most common addressing mechanism most are familiar with is the Internet Protocol address version 4 (IPv4). An IPv4 address is made up of a 32-bit four octet number represented in decimal format. In my example, we can see the address 172.16.60.69.

```

Command Prompt

Connection-specific DNS Suffix . : 
Description . . . . . : VMware Virtual Ethernet Adapter for VMnet8
Physical Address. . . . . : 00-50-56-C0-00-08
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . : fe80::f975:e312:2344:4b77%33(Preferred)
IPv4 Address. . . . . : 192.168.40.1(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Tuesday, October 1, 2024 11:12:09 AM
Lease Expires . . . . . : Tuesday, October 1, 2024 12:02:45 PM
Default Gateway . . . . . : 
DHCP Server . . . . . : 192.168.40.254
DHCPv6 IAID . . . . . : 1157648470
DHCPv6 Client DUID. . . . . : 00-01-00-01-2B-FA-34-A2-C8-5B-76-F3-2C-1B
Primary WINS Server . . . . . : 192.168.40.2
NetBIOS over Tcpip. . . . . : Enabled

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : mmu.ac.ke
Description . . . . . : Intel(R) Dual Band Wireless-AC 8260
Physical Address. . . . . : 3A-23-17-13-25-82
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . : fe80::de8d:92e:f82b:651f%19(Preferred)
IPv4 Address. . . . . : 172.16.60.148(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Tuesday, October 1, 2024 11:41:29 AM
Lease Expires . . . . . : Tuesday, October 1, 2024 12:41:34 PM

```

IPv6

IPv6 provides us a much larger address space that can be utilized for any networked purpose. IPv6 is a 128-bit address 16 octets represented in Hexadecimal format.

IPv6 Addressing Types

Type	Description
Unicast	Addresses for a single interface.
Anycast	Addresses for multiple interfaces, where only one of them receives the packet.
Multicast	Addresses for multiple interfaces, where all of them receive the same packet.
Broadcast	Does not exist and is realized with multicast addresses.

When thinking about each address type, it is helpful to remember that Unicast traffic is host to host, while Multicast is one to many, and Anycast is one to many in a group where only one will answer the packet. (think load balancing).

Even with its current state providing many advantages over IPv4, the adoption of IPv6 has been slow to catch on.

Adoption of IPv6

TCP / UDP, Transport Mechanisms

Characteristic	TCP	UDP
----------------	-----	-----

Transmission	Connection-oriented	Connectionless. Fire and forget.
Connection Establishment	TCP uses a three-way handshake to ensure that a connection is established.	UDP does not ensure the destination is listening.
Data Delivery	Stream-based conversations	packet by packet, the source does not care if the destination is active
Receipt of data	Sequence and Acknowledgement numbers are utilized to account for data.	UDP does not care
Speed	TCP has more overhead and is slower because of its built-in functions.	UDP is fast but unreliable.

TCP Three-way Handshake

1. The client sends a packet with the SYN flag set to on along with other negotiable options in the TCP header.
2. The server will respond with a TCP packet that includes a SYN flag set for the sequence number negotiation and an ACK flag set to acknowledge the previous SYN packet sent by the host.
3. The client will respond with a TCP packet with an ACK flag set agreeing to the negotiation.

Let us take a quick look at this in action to be familiar with it when it appears in our packet output later on in the module.

TCP Three-way Handshake

Source	Destination	Protocol	Length	Info
192.168.1.140	174.143.213.184	TCP	74	57678 → 80 [SYN] Seq=0 Win=5848 Len=0 MSS=1460 SACK_PERM=1 TSval=2216538 TSecr=
174.143.213.184	192.168.1.140	TCP	74	80 ← 57678 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=835
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=1 Ack=1 Win=888 Len=0 TSval=2216543 TSecr=835172936
192.168.1.140	174.143.213.184	HTTP	280	GET /images/layout/logo.png HTTP/1.0
174.143.213.184	192.168.1.140	TCP	66	80 ← 57678 [ACK] Seq=1 Ack=135 Win=6912 Len=0 TSval=835172948 TSecr=2216543
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=1 Ack=135 Win=6912 Len=1448 TSval=835172948 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=1449 Win=8832 Len=0 TSval=2216548 TSecr=835172948
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=135 Ack=135 Win=6912 Len=1448 TSval=835172948 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=2897 Win=11648 Len=0 TSval=2216548 TSecr=835172948
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=2897 Ack=135 Win=6912 Len=1448 TSval=835172948 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=4345 Win=14592 Len=0 TSval=2216548 TSecr=835172948
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=4345 Ack=135 Win=6912 Len=1448 TSval=835172961 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=5793 Win=17536 Len=0 TSval=2216553 TSecr=835172961
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=5793 Ack=135 Win=6912 Len=1448 TSval=835172961 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=7241 Win=26352 Len=0 TSval=2216553 TSecr=835172961
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=7241 Ack=135 Win=6912 Len=1448 TSval=835172961 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=8689 Win=23296 Len=0 TSval=2216553 TSecr=835172961
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=8689 Ack=135 Win=6912 Len=1448 TSval=835172961 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=18137 Win=26112 Len=0 TSval=2216553 TSecr=835172961
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=18137 Ack=135 Win=6912 Len=1448 TSval=835172961 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=11585 Win=29056 Len=0 TSval=2216553 TSecr=835172961
174.143.213.184	192.168.1.140	TCP	1514	80 ← 57678 [ACK] Seq=11585 Ack=135 Win=6912 Len=1448 TSval=835172961 TSecr=2216543
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=13033 Win=32000 Len=0 TSval=2216553 TSecr=835172961

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

TCP Session Teardown

Another flag we will see with TCP is the FIN flag.

1. **FIN, ACK**
2. **FIN, ACK,**
3. **ACK**

We have seen how a session is established with TCP; now, let us examine how a session is concluded.

TCP Session Teardown

Source	Destination	Protocol	Length	Info
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=8689 Win=25296 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=8689 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=10137 Win=26112 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=10137 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=11585 Win=29056 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=11585 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=13033 Win=32000 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=13033 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=14481 Win=34816 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [PSH, ACK] Seq=14481 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=15929 Win=37760 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=15929 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=17377 Win=40704 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=17377 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=18825 Win=43520 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=18825 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=20273 Win=46464 Len=0
174.143.213.184	192.168.1.140	TCP	1514	80 → 57678 [ACK] Seq=20273 Ack=135 Win=6912 Len=1448
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=21721 Win=49280 Len=0
174.143.213.184	192.168.1.140	HTTP	391	HTTP/1.1 200 OK (PNG)
192.168.1.140	174.143.213.184	TCP	66	57678 → 80 [ACK] Seq=135 Ack=22046 Win=52224 Len=0

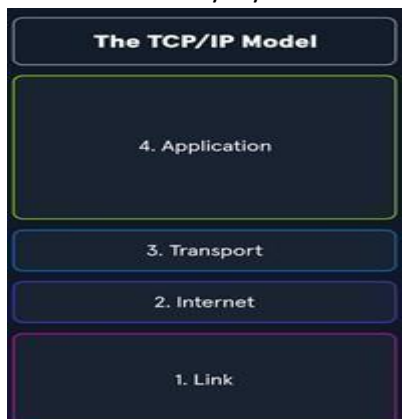
to return to your computer, move the mouse pointer outside or press Ctrl+Alt

Questions

1. How many layers does the OSI model have? 7



2. How many layers are there in the TCP/IP model? 4



3. True or False: Routers operate at layer 2 of the OSI model? **False**

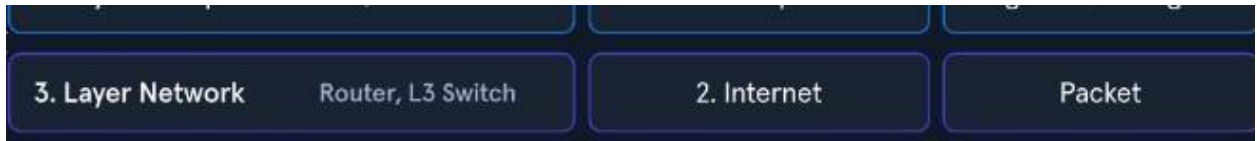
3. Layer Network

Router, L3 Switch

4. What addressing mechanism is used at the Link Layer of the TCP/IP model? **MAC-address**

MAC-addressing is utilized in Layer two (the data-link or link-layer depending on which model you look at) communications between hosts. This works through host-to-host communication within a broadcast domain. If layer two traffic needs to cross a layer three interface, that PDU is sent to the layer three across interface, and it is

5. At what layer of the OSI model is a PDU encapsulated into a packet? (the number) **3**



6. What addressing mechanism utilizes a 32-bit address? **IPv4**

An IPv4 address is made up of a 32-bit four octet number represented in decimal format. In our example, we can see the address 192.168.86.243. Each octet of an IP address can be represented by a number ranging from 0 to 255. When examining a PDU, we will find IP addresses in layer three (Network) of the OSI model and layer two (internet) of the TCP-IP model. We will not deep dive into IPv4 here, but for the sake of this module, understand what these addresses are, what they do for us, and at which layer they are used.

7. What Transport layer protocol is connection oriented? **TCP**

8. What Transport Layer protocol is considered unreliable? **UDP**

TCP VS. UDP		
Characteristic	TCP	UDP
Transmission	Connection-oriented	Connectionless. Fire and forget.

9. TCP's three-way handshake consists of 3 packets: 1.Syn, 2.Syn & ACK, 3. _? What is the final packet of the handshake? **ACK**

3. The client will respond with a TCP packet with an ACK flag set agreeing to the negotiation.

1. This packet is the end of the three-way handshake and established the connection between client and server.

Networking Prime – Layers 5-7

Questions

1. What is the default operational mode method used by FTP? **Active**

FTP is capable of running in two different modes, **active** or **passive**. Active is the default operational method utilized by FTP, meaning that the server listens for a control command **PORT** from the client, stating what port to use for data transfer. Passive mode enables us to access FTP servers located behind firewalls or a NAT-enabled link that makes direct TCP connections impossible. In this instance, the client would send the **PASV** command and wait for a response from the server informing the client what IP and port to utilize for the data transfer channel connection.

2. FTP utilizes what two ports for command and data transfer? (separate the two numbers with a space)
20 21

When we think about communication between hosts, we typically think about a client and server talking over a single socket. Through this socket, both the client and server send commands and data over the same link. In this aspect, FTP is unique since it utilizes multiple ports at a time. FTP uses ports 20 and 21 over TCP. Port 20 is used for data transfer, while port 21 is utilized for issuing commands controlling the FTP session. In regards to authentication, FTP supports user authentication as well as allowing anonymous access if configured.

3. Does SMB utilize TCP or UDP as its transport layer protocol? **TCP**

resource or perform actions. In the past, SMB utilized NetBIOS as its transport mechanism over UDP ports 137 and 138. Since modern changes, SMB now supports direct TCP transport over port 445, NetBIOS over TCP port 139, and even the QUIC protocol.

4. SMB has moved to using what TCP port? **445**

resource or perform actions. In the past, SMB utilized NetBIOS as its transport mechanism over UDP ports 137 and 138. Since modern changes, SMB now supports direct TCP transport over port 445, NetBIOS over TCP port 139, and even the QUIC protocol.

5. Hypertext Transfer Protocol uses what well known TCP port number? **80**

media (HTML, images, hyperlinks, video). HTTP utilizes ports 80 or 8000 over TCP during normal operations. In exceptional circumstances, it can be modified to use alternate ports, or even at times, UDP.

6. What HTTP method is used to request information and content from the webserver? **Get**

GET **required** Get is the most common method used. It requests information and content from the server. For example, **GET http://10.1.1.1/Webserver/index.html** requests the index.html page from the server based on our supplied URI.

7. What web based protocol uses TLS as a security measure? **HTTPS**

HTTPS

HTTP Secure (**HTTPS**) is a modification of the HTTP protocol designed to utilize Transport Layer Security (**TLS**) or Secure Sockets Layer (**SSL**) with older applications for data security. TLS is utilized as an encryption mechanism to secure the communications between a client and a server. TLS can wrap regular HTTP traffic within TLS, which

8. True or False: when utilizing HTTPS, all data sent across the session will appear as TLS Application data?

True

HTTP Secure (**HTTPS**) is a modification of the HTTP protocol designed to utilize Transport Layer Security (**TLS**) or Secure Sockets Layer (**SSL**) with older applications for data security. TLS is utilized as an encryption mechanism to secure the communications between a client and a server. TLS can wrap regular HTTP traffic within TLS, which means that we can encrypt our entire conversation, not just the data sent or requested. Before the TLS mechanism was in place, we were vulnerable to Man-in-the-middle attacks and other types of reconnaissance or hijacking, meaning anyone in the same LAN as the client or server could view the web traffic if they were listening on the wire. We can now have security implemented in the browser enabling everyone to encrypt their web habits, search requests, sessions or data transfers, bank transactions, and much more.

THE ANALYSIS PROCESS

Network Traffic Analysis (NTA) is a dynamic process aimed at examining network data to detect anomalies and potential threats. It involves breaking down traffic into understandable chunks, identifying deviations from normal patterns, and flagging suspicious activities such as unauthorized remote access (e.g., via RDP, SSH, or Telnet). The analysis helps set a baseline for typical network behavior, making it easier to spot unusual trends.

NTA is crucial for both defense and daily operations, providing visibility into network usage, top-talking hosts, and internal communications. It aids in troubleshooting, detecting issues, and ensuring protocols function

correctly. Tools like IDS/IPS, firewalls, and logging systems (e.g., Splunk or ELK Stack) enhance NTA by quickly alerting on known attacks, but human oversight remains essential, as attackers constantly find ways to bypass automated defenses. Combining automated tools with manual checks ensures more effective monitoring and protection.

Part 2: Tcpdump Fundamentals

1. Utilizing the output shown in question-1.png, who is the server in this communication? (IP Address)

174.143.213.184

```
l-$ tcpdump -nnP HTTP.cap
reading from file HTTP.cap, link-type EN10MB (Ethernet), snapshot length 65535
15:45:13.266821 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [S], seq 2387613953, win 5840, options [mss 1460,sackOK,TS val 2216538 ecr 0,nop,wscale 7], length 0
15:45:13.313726 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [S.], seq 3344080264, ack 2387613954, win 5792, options [mss 1460,sackOK,TS val 835172936 ecr 2216538,nop,wscale 6], length 0
15:45:13.313777 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 1, win 46, options [nop,nop,TS val 2216543 ecr 835172936], length 0
15:45:13.313889 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [P.], seq 11135, ack 1, win 46, options [nop,nop,TS val 2216543 ecr 835172936], length 134: HTTP: GET /images/layout/logo.png HTTP/1.0
15:45:13.361089 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216543], length 0
15:45:13.363494 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 11449, ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216543], length 1448: HTTP: HTTP/1.1 200 OK
15:45:13.363523 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 1449, win 69, options [nop,nop,TS val 2216548 ecr 835172948], length 0
15:45:13.363606 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1449:2897, ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216543], length 1448: HTTP
15:45:13.363610 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 2897, win 91, options [nop,nop,TS val 2216548 ecr 835172948], length 0
15:45:13.366822 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 2897:4245, ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216543], length 1448: HTTP
15:45:13.366844 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 4345, win 114, options [nop,nop,TS val 2216548 ecr 835172948], length 0
15:45:13.411058 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 4345:5793, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.411084 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 5793, win 137, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.413884 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 5793:7241, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.413893 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 7241, win 159, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.414068 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 7241:8689, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.414013 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 8689, win 182, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.416301 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 8689:10137, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.416309 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 10137, win 204, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.416424 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 10137:11585, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.416432 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 11585, win 227, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.416547 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 11585:13033, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.416556 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 13033, win 250, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.458467 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 13033:14481, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.458479 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 14481, win 272, options [nop,nop,TS val 2216557 ecr 835172973], length 0
15:45:13.461293 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [P.], seq 14481:15929, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.461302 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 15929, win 295, options [nop,nop,TS val 2216558 ecr 835172973], length 0
15:45:13.463422 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 15929:17377, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.463430 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 17377, win 318, options [nop,nop,TS val 2216558 ecr 835172973], length 0
15:45:13.463544 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 17377:18825, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.463552 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 18825, win 340, options [nop,nop,TS val 2216558 ecr 835172973], length 0
15:45:13.464163 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 18825:20273, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.464173 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 20273, win 363, options [nop,nop,TS val 2216558 ecr 835172973], length 0
15:45:13.466749 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 20273:21721, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.466757 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 21721, win 385, options [nop,nop,TS val 2216558 ecr 835172973], length 0
15:45:13.466776 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [P.], seq 21721:22046, ack 135, win 108, options [nop,nop,TS val 835172974 ecr 2216553], length 325: HTTP
15:45:13.467401 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 22046, win 408, options [nop,nop,TS val 2216558 ecr 835172974], length 0
15:45:13.513631 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [F.], seq 135, ack 22046, win 408, options [nop,nop,TS val 2216558 ecr 835172986], length 0
15:45:13.513650 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 22047, win 408, options [nop,nop,TS val 2216563 ecr 835172986], length 0
```

2. Were absolute or relative sequence numbers used during the capture? (see question-1.zip to answer)

relative


```
L$ tcpdump -nnr H
reading from file H
15:45:13.266821 IP
15:45:13.313726 IP
15:45:13.313777 IP
15:45:13.313889 IP
15:45:13.361089 IP
15:45:13.363494 IP
15:45:13.363523 IP
15:45:13.363606 IP
15:45:13.363610 IP
15:45:13.366822 IP
15:45:13.366844 IP
15:45:13.411058 IP
15:45:13.411084 IP
15:45:13.413884 IP
15:45:13.413893 IP
15:45:13.414005 IP
15:45:13.414013 IP
15:45:13.416301 IP
15:45:13.416309 IP
15:45:13.416424 IP
15:45:13.416432 IP
15:45:13.416547 IP
15:45:13.416556 IP
15:45:13.458467 IP
15:45:13.458479 IP
15:45:13.461293 IP
15:45:13.461302 IP
15:45:13.463422 IP
15:45:13.463430 IP
15:45:13.463544 IP
15:45:13.463552 IP
15:45:13.464163 IP
15:45:13.464171 IP
15:45:13.466749 IP
15:45:13.466757 IP
15:45:13.466771 IP
15:45:13.466776 IP
15:45:13.467401 IP
15:45:13.513631 IP
15:45:13.513650 IP
```

3. If I wish to start a capture without hostname resolution, verbose output, showing contents in ASCII and hex, and grab the first 100 packets; what are the switches used? Please answer in the order the switches are asked for in the question. **-nvXc 100**

```
kali-linux-2022.1-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2022.1-vmware...
kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ sudo tcpdump -nvXc 100
[sudo] password for kali:
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 2621
44 bytes
06:13:38.334493 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.
40.2 tell 192.168.40.1, length 46
    0x0000: 0001 0800 0604 0001 0050 56c0 0008 c0a8 .....PV.....
    0x0010: 2801 0000 0000 0000 c0a8 2802 0000 0000 .....(.....
    0x0020: 0000 0000 0000 0000 0000 0000 0000 .....
06:13:41.424662 IP (tos 0x0, ttl 128, id 19060, offset 0, flags [none], proto
TCP (6), length 75)
    52.48.38.99.443 > 192.168.40.128.56574: Flags [P.], cksum 0x2c86 (correct
), seq 1257950312:1257950347, ack 242977983, win 64240, length 35
    0x0000: 4500 004b 4a74 0000 8006 ac7d 3430 2663 E..KJt.....}408c
    0x0010: c0a8 2880 01bb dcf6 4afa cc68 0e7b 8cbf ..(.....J..h.{..
    0x0020: 5018 faf0 2c86 0000 1703 0300 1e0e 4533 P... ,.....E3
    0x0030: 0f59 2af4 c2cf b695 84ec b9ee 139f 0fad .Y*.....
    0x0040: bb61 f170 d153 1c01 30d9 55 .a.p.S..0.U
06:13:41.424699 IP (tos 0x0, ttl 64, id 49890, offset 0, flags [DF], proto TC
P (6), length 40)
    192.168.40.128.56574 > 52.48.38.99.443: Flags [.], cksum 0x43d6 (incorec
t -> 0xe561), ack 35, win 62780, length 0
    0x0000: 4500 0028 c2e2 4000 4006 3432 c0a8 2880 E..(..@.@.42..(
    0x0010: 3430 2663 dcf6 01bb 0e7b 8cbf 4afa cc8b 408c.....{..J...
    0x0020: 5010 f53c 43d6 0000 P..<C...
06:13:42.305246 IP (tos 0x0, ttl 64, id 40702, offset 0, flags [DF], proto TC
P (6), length 278)
    192.168.40.128.44372 > 34.237.73.95.443: Flags [P.], cksum 0x567d (incorr
ect -> 0x73d7), seq 1208174516:1208174754, ack 797402798, win 62780, length 2
To return to your computer, move the mouse pointer outside or press Ctrl+Alt.
```

4. Given the capture file at /tmp/capture.pcap, what tcpdump command will enable you to read from the capture and show the output contents in Hex and ASCII? (Please use best practices when using switches)
- sudo tcpdump -Xr /tmp/capture.pcap**

```
(kali@kali)-[~]
$ sudo tcpdump -i eth0 -Xr /capture.pcap
reading from file /capture.pcap, link-type EN10MB (Ethernet), snapshot length 262144
16:14:43.844915 IP ec2-34-237-73-95.compute-1.amazonaws.com.https > 192.168.29.128.57998: Flags [P.], seq 1186325643:1186325912, ack 30
length 269
0x0000: 4500 0135 2664 0000 8006 c8ea 22ed 495f E..56d.....".I_
0x0010: c0a8 1d80 01bb e28e 46b5 e48b b717 fd19 .....F.....
0x0020: 5018 faf0 d0cd 0000 1703 0301 0803 9764 P.....d
0x0030: 57fa dd86 e61a cf23 5800 6baa 1f0b 5a21 W.....#X.k...Z!
0x0040: 2501 feae 825f 7ebb 3152 ca27 be22 435f %.....~.1R.'"C_
0x0050: d6b9 d800 fb1e 93c1 16d8 2958 8f23 74ac .....X.#t.
0x0060: 7aae 2819 d420 60d7 354c 6af5 5acc 655e z.(...`.5Lj.Z.e^
0x0070: 7216 8e6d 9f12 45cc 46ac 3bd0 667a 602d r...m..E.F.;.fz~-
0x0080: c89e 936a 1a8c a885 e511 cb00 9d59 e73c ...j.....Y.<
0x0090: 4423 1e77 16b2 63ec 5b58 a708 ce36 6219 D#.w..c.[X...6b.
0x00a0: e6e0 4b75 3328 f838 7031 cde9 a923 ce64 ..Ku3(.8p1...#.d
0x00b0: f9fe 27be 8d8e 5200 68d7 003c 64d4 d12f ..'...R.h..<d../
0x00c0: f197 fe56 3eed 11bb a152 2428 775a 357e ...V>...R$(wZ5~
0x00d0: 988b 5e71 cc4a 5661 c66a b4bb 3206 2fdb ..^q,JVa.j..2./
0x00e0: c1d0 a1fd 3b29 07a5 4cab d49e 852f cbfd .....;)..L....//
```

5. What TCPDump switch will increase the verbosity of our output? (Include the - with the proper switch)

-v

```
v,v,w,vw      Increase the verbosity of output shown and saved.
```

6. What built in terminal help reference can tell us more about TCPDump? Man

```
(kali@kali)-[~]
$ man tcpdump
(kali@kali)-[~]
$
```

7. What TCPDump switch will let me write my output to a file? -w

```
w file.pcap      Write into a file
```

Fundamentals Lab

1. What TCPDump switch will allow us to pipe the contents of a pcap file out to another function such as 'grep'? -l

```
(kali@kali)-[~]
$ sudo tcpdump host 192.168.29.128 -l | grep '*compute*'
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
```

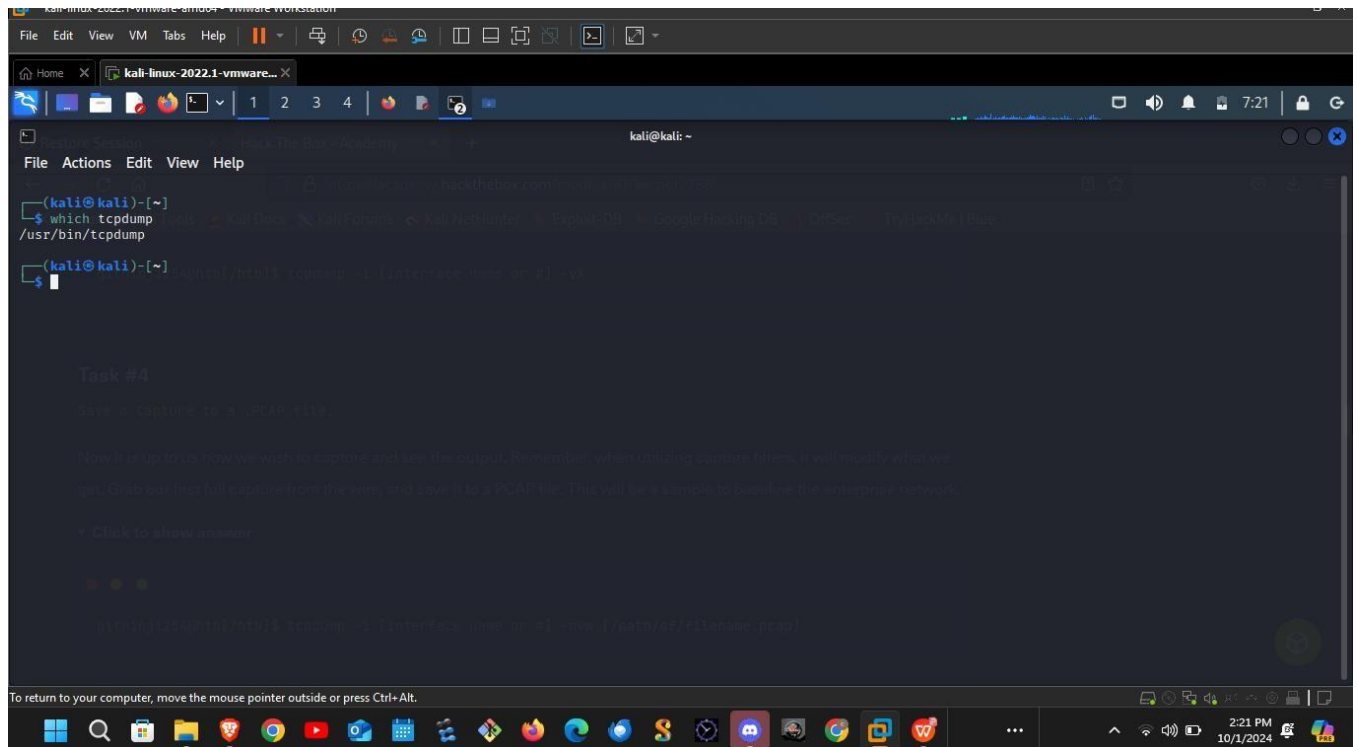
2. True or False: The filter "port" looks at source and destination traffic. **True**

```
port port is bi-directional. It will show any traffic with the specified port as the source or destination
portrange portrange allows us to specify ranges of ports (0-1024)
```

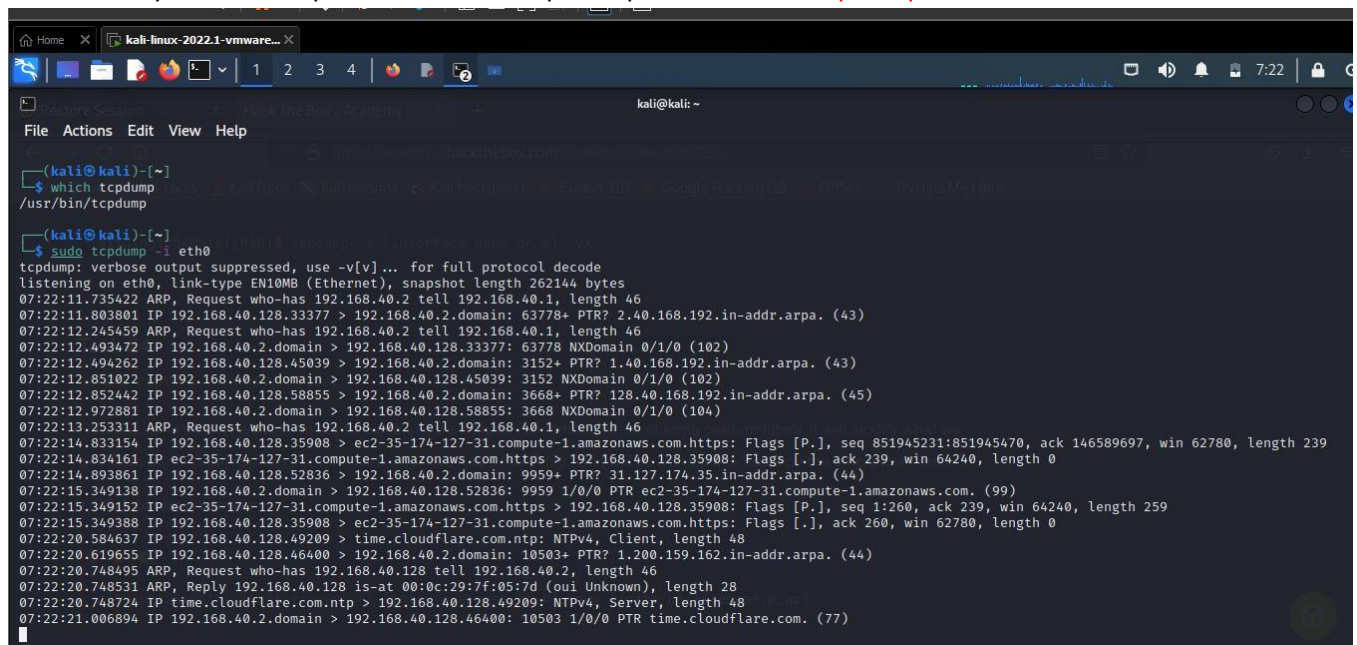
3. If we wished to filter out ICMP traffic from our capture, what filter could we use? (word only, not symbol please.) **not ICMP**

```
(kali@kali)-[~]
$ sudo tcpdump -i eth0 -Xr /capture.pcap not icmp
[sudo] password for kali:
reading from file /capture.pcap, link-type EN10MB (Ethernet), snapshot length 262144
16:14:43.844915 IP ec2-34-237-73-95.compute-1.amazonaws.com.https > 192.168.29.128.57998: Flags [P.], seq 1186325643:1186325912, ack 30
length 269
0x0000: 4500 0135 2664 0000 8006 c8ea 22ed 495f E..56d.....".I_
0x0010: c0a8 1d80 01bb e28e 46b5 e48b b717 fd19 .....F.....
0x0020: 5018 faf0 d0cd 0000 1703 0301 0803 9764 P.....d
0x0030: 57fa dd86 e61a cf23 5800 6baa 1f0b 5a21 W.....#X.k...Z!
0x0040: 2501 feae 825f 7ebb 3152 ca27 be22 435f %....~.1R.'"C_
0x0050: d6b9 d800 fb1e 93c1 16d8 2958 8f23 74ac .....X.#t.
0x0060: 7aae 2819 d420 60d7 354c 6af5 5acc 655e z.( ...".5Lj.Z.e^
0x0070: 7216 8e6d 9f12 45cc 46ac 3bd0 667a 602d r..m..E.F.;.fz'-
0x0080: c89e 936a 1a8c a885 e511 cb00 9d59 e73c ...j.....Y.<
0x0090: 4423 1e77 16b2 63ec 5b58 a708 ce36 6219 D#.w..c.[X...6b.
0x00a0: e6e0 4b75 3328 f838 7031 cde9 a923 ce64 ..Ku3(.8p1...#.d
0x00b0: f9fe 27be 8d8e 5200 68d7 003c 64d4 d12f ..'...R.h..<d../
0x00c0: f197 fe56 3eed 11bb a152 2428 775a 357e ...V>....R$(wZ5-
0x00d0: 988b 5e71 cc4a 5661 c66a b4bb 3206 2fdb ..^q.JVa.j..2./.
```

4. What command will show you where / if TCPDump is installed? **which tcpdump**



5. How do you start a capture with TCPDump to capture on eth0? **Tcpdump -i eth0**

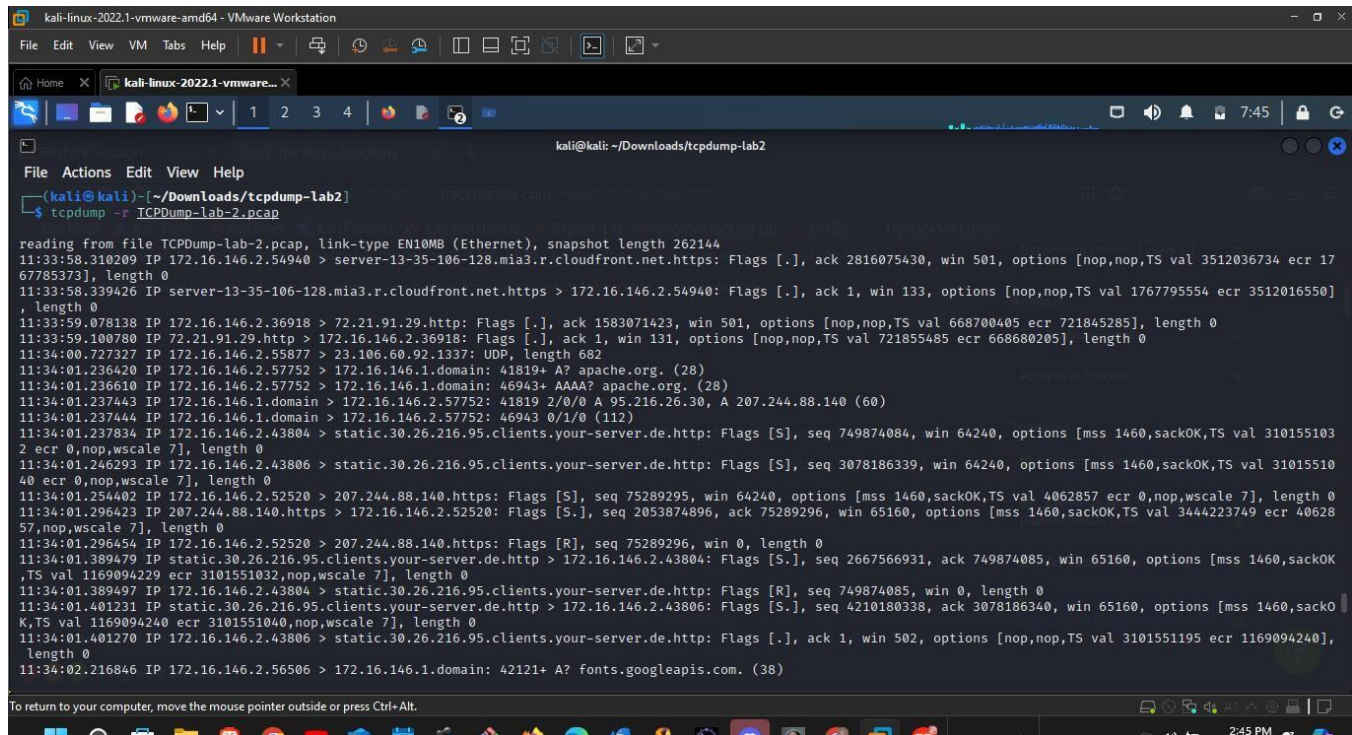


6. What switch will provide more verbosity in your output? **-v**

```
(kali@kali)-[~]
└─$ sudo tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:52:29.584682 IP (tos 0x0, ttl 64, id 5685, offset 0, flags [DF], proto TCP (6), length 288)
    192.168.29.128.57998 > ec2-34-237-73-95.compute-1.amazonaws.com.https: Flags [P.], cksum 0x4b87 (incorrect -> 0x8fff), seq 30718392,
    86365268, win 65535, length 248
16:52:29.586580 IP (tos 0x0, ttl 128, id 11178, offset 0, flags [none], proto TCP (6), length 40)
    ec2-34-237-73-95.compute-1.amazonaws.com.https > 192.168.29.128.57998: Flags [.], cksum 0x7bcb (correct), ack 248, win 64240, lengt
16:52:29.630310 IP (tos 0x0, ttl 64, id 63933, offset 0, flags [DF], proto UDP (17), length 71)
    192.168.29.128.43092 > 192.168.29.2.domain: 1905+ PTR? 95.73.237.34.in-addr.arpa. (43)
16:52:29.665880 IP (tos 0x0, ttl 128, id 11179, offset 0, flags [none], proto UDP (17), length 125)
    192.168.29.2.domain > 192.168.29.128.43092: 1905 1/0/0 95.73.237.34.in-addr.arpa. PTR ec2-34-237-73-95.compute-1.amazonaws.com. (97
16:52:29.667425 IP (tos 0x0, ttl 64, id 5107, offset 0, flags [DF], proto UDP (17), length 73)
    192.168.29.128.57998 > 192.168.29.2.domain: 1905+ PTR? 95.73.237.34.in-addr.arpa. (43)
```

7. What switch will write your capture output to a .pcap file? **-w**

8. What switch will read a capture from a .pcap file? **-r**



```
kali@kali: ~/Downloads/tcpdump-lab2
└─$ tcpdump -r TCPDump-lab-2.pcap
reading from file TCPDump-lab-2.pcap, link-type EN10MB (Ethernet), snapshot length 262144
11:33:58.310209 IP 172.16.146.2.54940 > server-13-35-106-128.mia3.r.cloudfront.net.https: Flags [.], ack 2816075430, win 501, options [nop,nop,TS val 3512036734 ecr 176785373], length 0
11:33:58.339426 IP server-13-35-106-128.mia3.r.cloudfront.net.https > 172.16.146.2.54940: Flags [.], ack 1, win 133, options [nop,nop,TS val 1767795554 ecr 3512016550], length 0
11:33:59.078138 IP 172.16.146.2.36918 > 72.21.91.29.http: Flags [.], ack 1583071423, win 501, options [nop,nop,TS val 668700405 ecr 721845285], length 0
11:33:59.100780 IP 72.21.91.29.http > 172.16.146.2.36918: Flags [.], ack 1, win 131, options [nop,nop,TS val 721855485 ecr 668680205], length 0
11:34:00.727327 IP 172.16.146.2.55877 > 23.106.60.92.1337: UDP, length 682
11:34:01.236420 IP 172.16.146.2.57752 > 172.16.146.1.domain: 41819+ A? apache.org. (28)
11:34:01.236610 IP 172.16.146.2.57752 > 172.16.146.1.domain: 46943+ AAAA? apache.org. (28)
11:34:01.237443 IP 172.16.146.1.domain > 172.16.146.2.57752: 41819 2/0/0 A 95.216.26.30, A 207.244.88.140 (60)
11:34:01.237444 IP 172.16.146.1.domain > 172.16.146.2.57752: 46943 0/1/0 (112)
11:34:01.237834 IP 172.16.146.2.43804 > static.30.26.216.95.clients.your-server.de.http: Flags [S], seq 749874084, win 64240, options [mss 1460,sackOK,TS val 310155103 ecr 0,nop,wscale 7], length 0
11:34:01.246293 IP 172.16.146.2.43806 > static.30.26.216.95.clients.your-server.de.http: Flags [S], seq 3078186339, win 64240, options [mss 1460,sackOK,TS val 310155104 ecr 0,nop,wscale 7], length 0
11:34:01.254402 IP 172.16.146.2.52520 > 207.244.88.140.https: Flags [S], seq 75289295, win 64240, options [mss 1460,sackOK,TS val 4062857 ecr 0,nop,wscale 7], length 0
11:34:01.296423 IP 207.244.88.140.https > 172.16.146.2.52520: Flags [S.], seq 2053874896, ack 75289296, win 65160, options [mss 1460,sackOK,TS val 3444223749 ecr 4062857,nop,wscale 7], length 0
11:34:01.296454 IP 172.16.146.2.52520 > 207.244.88.140.https: Flags [R], seq 75289296, win 0, length 0
11:34:01.389479 IP static.30.26.216.95.clients.your-server.de.http > 172.16.146.2.43804: Flags [S.], seq 2667566931, ack 749874085, win 65160, options [mss 1460,sackOK,TS val 1169094229 ecr 3101551032,nop,wscale 7], length 0
11:34:01.389497 IP 172.16.146.2.43804 > static.30.26.216.95.clients.your-server.de.http: Flags [R], seq 749874085, win 0, length 0
11:34:01.401231 IP static.30.26.216.95.clients.your-server.de.http > 172.16.146.2.43806: Flags [S.], seq 4210180338, ack 3078186340, win 65160, options [mss 1460,sackOK,TS val 1169094240 ecr 3101551040,nop,wscale 7], length 0
11:34:01.401270 IP 172.16.146.2.43806 > static.30.26.216.95.clients.your-server.de.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 3101551195 ecr 1169094240], length 0
11:34:02.216846 IP 172.16.146.2.56506 > 172.16.146.1.domain: 42121+ A? fonts.googleapis.com. (38)
```

9. What switch will show the contents of a capture in Hex and ASCII? **-X**

Tcpdump Packet Filtering

1. What filter will allow me to see traffic coming from or destined to the host with an IP of 10.10.20.1? **host 10.10.20.1**

```
(kali@kali)~$ sudo tcpdump host 192.168.29.128
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:10:50.616973 IP 192.168.29.128.57998 > ec2-34-237-73-95.compute-1.amazonaws.com.https: Flags [P.], seq 3071857684:3071857932, ack 1186385030, win 65535, length 248
17:10:50.619051 IP ec2-34-237-73-95.compute-1.amazonaws.com.https > 192.168.29.128.57998: Flags [.], ack 248, win 64240, length 0
17:10:50.723137 IP 192.168.29.128.37053 > 192.168.29.2.domain: 59712+ PTR? 95.73.237.34.in-addr.arpa. (43)
17:10:50.774524 IP 192.168.29.2.domain > 192.168.29.128.37053: 59712 1/0/0 PTR ec2-34-237-73-95.compute-1.amazonaws.com. (97)
17:10:50.775956 IP 192.168.29.128.33133 > 192.168.29.2.domain: 18627+ PTR? 128.29.168.192.in-addr.arpa. (45)
17:10:50.808074 IP 192.168.29.2.domain > 192.168.29.128.33133: 18627 NXDomain 0/1/0 (104)
17:10:50.821629 IP 192.168.29.128.43314 > 192.168.29.2.domain: 42463+ PTR? 2.29.168.192.in-addr.arpa. (43)
17:10:50.858242 IP 192.168.29.2.domain > 192.168.29.128.43314: 42463 NXDomain 0/1/0 (102)
17:10:50.858260 IP ec2-34-237-73-95.compute-1.amazonaws.com.https > 192.168.29.128.57998: Flags [P.], seq 1:270, ack 248, win 64240, length 269
17:10:50.901800 IP 192.168.29.128.57998 > ec2-34-237-73-95.compute-1.amazonaws.com.https: Flags [.], ack 270, win 65535, length 0
17:10:52.193462 IP ec2-52-48-38-99.eu-west-1.compute.amazonaws.com.https > 192.168.29.128.43120: Flags [P.], seq 1831238494:1831238529, ack 702698541, win 64240, length 35
```

2. What filter will allow me to capture based on either of two options? **Or**

```
(kali@kali)~/Downloads$ sudo tcpdump -xr /home/kali/Downloads/extracting-objects-from-pcap-example-01.pcap tcp or host 162.50.180.107
reading from file /home/kali/Downloads/extracting-objects-from-pcap-example-01.pcap, link-type EN10MB (Ethernet), snapshot length 65535
16:37:54.670219 IP 10.6.27.102.49157 > a23-63-254-163.deploy.static.akamaitechnologies.com.http: Flags [S], seq 3513516593, win 8192, options [mss 14 wscale 8,nop,nop,sackOK], length 0
0x0000: 4500 0034 0061 4000 8006 bf14 0a06 1b66 E...4.a@.....f
0x0010: 173f fea3 c005 0050 d16c 0231 0000 0000 .?.....P.l.1....
0x0020: 8002 2000 7fce 0000 0204 05b4 0103 0308 .....
0x0030: 0101 0402 ....
16:37:54.712504 IP a23-63-254-163.deploy.static.akamaitechnologies.com.http > 10.6.27.102.49157: Flags [S.], seq 2439624788, ack 3513516594, win 64240, options [mss 1460], length 0
0x0000: 4500 002c 3c42 0000 8006 c33b 173f fea3 E...<B.....;? ..
0x0010: 0a06 1b66 0050 c005 9169 b854 d16c 0232 ...f.P...i.T.l.2
0x0020: 6012 faf0 8424 0000 0204 05b4 .....
16:37:54.712953 IP 10.6.27.102.49157 > a23-63-254-163.deploy.static.akamaitechnologies.com.http: Flags [.], ack 1, win 64240, length 0
0x0000: 4500 0028 0062 4000 8006 bf1f 0a06 1b66 E..(.b@.....f
0x0010: 173f fea3 c005 0050 d16c 0232 9169 b855 .?.....P.l.2.i.U
0x0020: 5010 faf0 9be1 0000 P.....
16:37:54.713411 IP 10.6.27.102.49157 > a23-63-254-163.deploy.static.akamaitechnologies.com.http: Flags [P.], seq 1:98, ack 1, win 64240, length 97: HTTP/1.1
0x0000: 4500 0089 0063 4000 8006 bebd 0a06 1b66 E....c@.....f
0x0010: 173f fea3 c005 0050 d16c 0232 9169 b855 .?.....P.l.2.i.U
0x0020: 5018 faf0 ec35 0000 4745 5420 2f6e 6373 P....5..GET./ncs
0x0030: 692e 7478 7420 4854 5450 2f31 2e31 0d0a i.txt.HTTP/1.1..
0x0040: 436f 6e6e 6563 7469 6f6e 3a20 436c 6f73 Connection:.Clos
0x0050: 650d 0a55 736f 722d 4167 656e 743a 204d e..User-Agent:.M
0x0060: 6963 726f 736f 6674 204e 4353 490d 0a48 icrosoft.NCSE..H
0x0070: 6f73 743a 2077 7777 2e6d 7366 746e 6373 ost:.www.msftncs
0x0080: 692e 636f 6d0d 0a0d 0a i.com....
16:37:54.713521 IP a23-63-254-163.deploy.static.akamaitechnologies.com.http > 10.6.27.102.49157: Flags [.], ack 98, win 64240, length 0
0x0000: 4500 0028 3c43 0000 8006 c33e 173f fea3 E..(<C.....>? ..
0x0010: 0a06 1b66 0050 c005 9169 b855 d16c 0293 ...f.P...i.U.l..
0x0020: 5010 faf0 9b80 0000 P.....
16:37:54.743959 IP a23-63-254-163.deploy.static.akamaitechnologies.com.http > 10.6.27.102.49157: Flags [FP.], seq 1:180, ack 98, win 64240, length 17: HTTP/1.1 200 OK
0x0000: 4500 00db 3c44 0000 8006 c28a 173f fea3 E...<D.....?..
```

3. True or False: TCPDump will resolve IPs to hostnames by default. **True**

Interrogating Network Traffic With Capture and Display Filters

```
kali@kali: ~/Downloads/tcpdump-lab2
tcpdump -r TCPDump-lab-2.pcap

reading from file TCPDump-lab-2.pcap, link-type EN10MB (Ethernet), snapshot length 262144
11:33:58.310209 IP 172.16.146.2.54940 > server-13-35-106-128.mia3.r.cloudfront.net.https: Flags [.], ack 2816075430, win 501, options [nop,nop,TS val 3512036734 ecr 1767785373], length 0
11:33:58.339426 IP server-13-35-106-128.mia3.r.cloudfront.net.https > 172.16.146.2.54940: Flags [.], ack 1, win 133, options [nop,nop,TS val 1767795554 ecr 3512016550], length 0
11:33:59.078138 IP 172.16.146.2.36918 > 72.21.91.29.http: Flags [.], ack 1583071423, win 501, options [nop,nop,TS val 668700405 ecr 721845285], length 0
11:33:59.100780 IP 72.21.91.29.http > 172.16.146.2.36918: Flags [.], ack 1, win 131, options [nop,nop,TS val 721855485 ecr 668680205], length 0
11:34:00.727327 IP 172.16.146.2.55877 > 23.106.60.92.1337: UDP, length 682
11:34:01.236420 IP 172.16.146.2.57752 > 172.16.146.1.domain: 41819+ A? apache.org. (28)
11:34:01.236610 IP 172.16.146.2.57752 > 172.16.146.1.domain: 46943+ AAAA? apache.org. (28)
11:34:01.237443 IP 172.16.146.1.domain > 172.16.146.2.57752: 41819 2/0/0 A 95.216.26.30, A 207.244.88.140 (60)
11:34:01.237444 IP 172.16.146.1.domain > 172.16.146.2.57752: 46943 0/1/0 (112)
11:34:01.237834 IP 172.16.146.2.43804 > static.30.26.216.95.clients.your-server.de.http: Flags [S], seq 749874084, win 64240, options [mss 1460,sackOK,TS val 3101551032 ecr 0,nop,wscale 7], length 0
11:34:01.246293 IP 172.16.146.2.43806 > static.30.26.216.95.clients.your-server.de.http: Flags [S], seq 3078186339, win 64240, options [mss 1460,sackOK,TS val 3101551040 ecr 0,nop,wscale 7], length 0
11:34:01.254402 IP 172.16.146.2.52520 > 207.244.88.140.https: Flags [S], seq 75289295, win 64240, options [mss 1460,sackOK,TS val 4062857 ecr 0,nop,wscale 7], length 0
11:34:01.296423 IP 207.244.88.140.https > 172.16.146.2.52520: Flags [S.], seq 2053874896, ack 75289296, win 65160, options [mss 1460,sackOK,TS val 3444223749 ecr 4062857,nop,wscale 7], length 0
11:34:01.296454 IP 172.16.146.2.52520 > 207.244.88.140.https: Flags [R], seq 75289296, win 0, length 0
11:34:01.389479 IP static.30.26.216.95.clients.your-server.de.http > 172.16.146.2.43804: Flags [S.], seq 2667566931, ack 749874085, win 65160, options [mss 1460,sackOK,TS val 1169094229 ecr 3101551032,nop,wscale 7], length 0
11:34:01.389497 IP 172.16.146.2.43804 > static.30.26.216.95.clients.your-server.de.http: Flags [R], seq 749874085, win 0, length 0
11:34:01.401231 IP static.30.26.216.95.clients.your-server.de.http > 172.16.146.2.43806: Flags [S.], seq 4210180338, ack 3078186340, win 65160, options [mss 1460,sackOK,TS val 1169094240 ecr 3101551040,nop,wscale 7], length 0
11:34:01.401270 IP 172.16.146.2.43806 > static.30.26.216.95.clients.your-server.de.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 3101551195 ecr 1169094240], length 0
11:34:02.216846 IP 172.16.146.2.56506 > 172.16.146.1.domain: 42121+ A? fonts.googleapis.com. (38)

To return to your computer, move the mouse pointer outside or press Ctrl+Alt.

kali@kali: ~/Downloads/tcpdump-lab2

}}, length 0
11:34:02.684107 IP yx-in-f100.1e100.net.https > 172.16.146.2.56302: Flags [P.], seq 1:3868, ack 514, win 261, options [nop,nop,TS val 3266376782 ecr 267112352], length 3867
11:34:02.684141 IP 172.16.146.2.56302 > yx-in-f100.1e100.net.https: Flags [.], ack 3868, win 489, options [nop,nop,TS val 267112382 ecr 3266376782], length 0
11:34:02.690789 IP 172.16.146.2.46918 > slc18s02-in-f3.1e100.net.http: Flags [P.], seq 379:756, ack 703, win 501, options [nop,nop,TS val 917146064 ecr 3427455185], length 377: HTTP: POST /gtsolcore HTTP/1.1
11:34:02.695787 IP 207.244.88.140.https > 172.16.146.2.52540: Flags [.], ack 594, win 506, options [nop,nop,TS val 3444225147 ecr 4064258], length 0
11:34:02.695787 IP 207.244.88.140.https > 172.16.146.2.52540: Flags [P.], seq 5209:5512, ack 594, win 506, options [nop,nop,TS val 3444225148 ecr 4064258], length 303
11:34:02.695812 IP 172.16.146.2.52540 > 207.244.88.140.https: Flags [.], ack 5512, win 501, options [nop,nop,TS val 4064299 ecr 3444225148], length 0
11:34:02.695787 IP 207.244.88.140.https > 172.16.146.2.52540: Flags [P.], seq 5512:5815, ack 594, win 506, options [nop,nop,TS val 3444225148 ecr 4064258], length 303
11:34:02.695845 IP 172.16.146.2.52540 > 207.244.88.140.https: Flags [.], ack 5815, win 499, options [nop,nop,TS val 4064299 ecr 3444225148], length 0
11:34:02.700287 IP 207.244.88.140.https > 172.16.146.2.52540: Flags [.], ack 913, win 504, options [nop,nop,TS val 3444225151 ecr 4064259], length 0
11:34:02.700287 IP 207.244.88.140.https > 172.16.146.2.52538: Flags [.], ack 594, win 506, options [nop,nop,TS val 3444225153 ecr 4064261], length 0
11:34:02.700287 IP 207.244.88.140.https > 172.16.146.2.52538: Flags [.], ack 917, win 504, options [nop,nop,TS val 3444225153 ecr 4064262], length 0
11:34:02.702755 IP 207.244.88.140.https > 172.16.146.2.52538: Flags [P.], seq 5209:5512, ack 917, win 504, options [nop,nop,TS val 3444225154 ecr 4064262], length 303
11:34:02.702777 IP 172.16.146.2.52538 > 207.244.88.140.https: Flags [.], ack 5512, win 501, options [nop,nop,TS val 4064306 ecr 3444225154], length 0
11:34:02.702755 IP 207.244.88.140.https > 172.16.146.2.52538: Flags [P.], seq 5512:5815, ack 917, win 504, options [nop,nop,TS val 3444225154 ecr 4064262], length 303
11:34:02.702827 IP 172.16.146.2.52538 > 207.244.88.140.https: Flags [.], ack 5815, win 499, options [nop,nop,TS val 4064306 ecr 3444225154], length 0
11:34:02.717255 IP slc18s02-in-f3.1e100.net.http > 172.16.146.2.46918: Flags [P.], seq 703:1404, ack 756, win 265, options [nop,nop,TS val 3427455369 ecr 917146064], length 701: HTTP: HTTP/1.1 200 OK
11:34:02.717313 IP 172.16.146.2.46918 > slc18s02-in-f3.1e100.net.http: Flags [.], ack 1404, win 501, options [nop,nop,TS val 917146090 ecr 3427455369], length 0
11:34:02.721072 IP 172.16.146.2.56302 > yx-in-f100.1e100.net.https: Flags [P.], seq 514:578, ack 3868, win 501, options [nop,nop,TS val 267112419 ecr 3266376782], length 64
11:34:02.721832 IP 172.16.146.2.56302 > yx-in-f100.1e100.net.https: Flags [P.], seq 578:748, ack 3868, win 501, options [nop,nop,TS val 267112420 ecr 3266376782], length 170
11:34:02.722340 IP 172.16.146.2.56302 > yx-in-f100.1e100.net.https: Flags [P.], seq 748:1111, ack 3868, win 501, options [nop,nop,TS val 267112420 ecr 3266376782], length 363
11:34:02.722816 IP 172.16.146.2.56302 > yx-in-f100.1e100.net.https: Flags [P.], seq 1111:1205, ack 3868, win 501, options [nop,nop,TS val 267112421 ecr 3266376782], length 94
11:34:02.750142 IP 172.16.146.2.37106 > yx-in-f91.1e100.net.https: Flags [P.], seq 1:514, ack 1, win 502, options [nop,nop,TS val 1945790581 ecr 3321847450], length 513

To return to your computer, move the mouse pointer outside or press Ctrl+Alt.
```

1. What are the client and server port numbers used in first full TCP three-way handshake? (low number first then high number) **80 43806**
2. Based on the traffic seen in the pcap file, who is the DNS server in this network segment? (ip address)

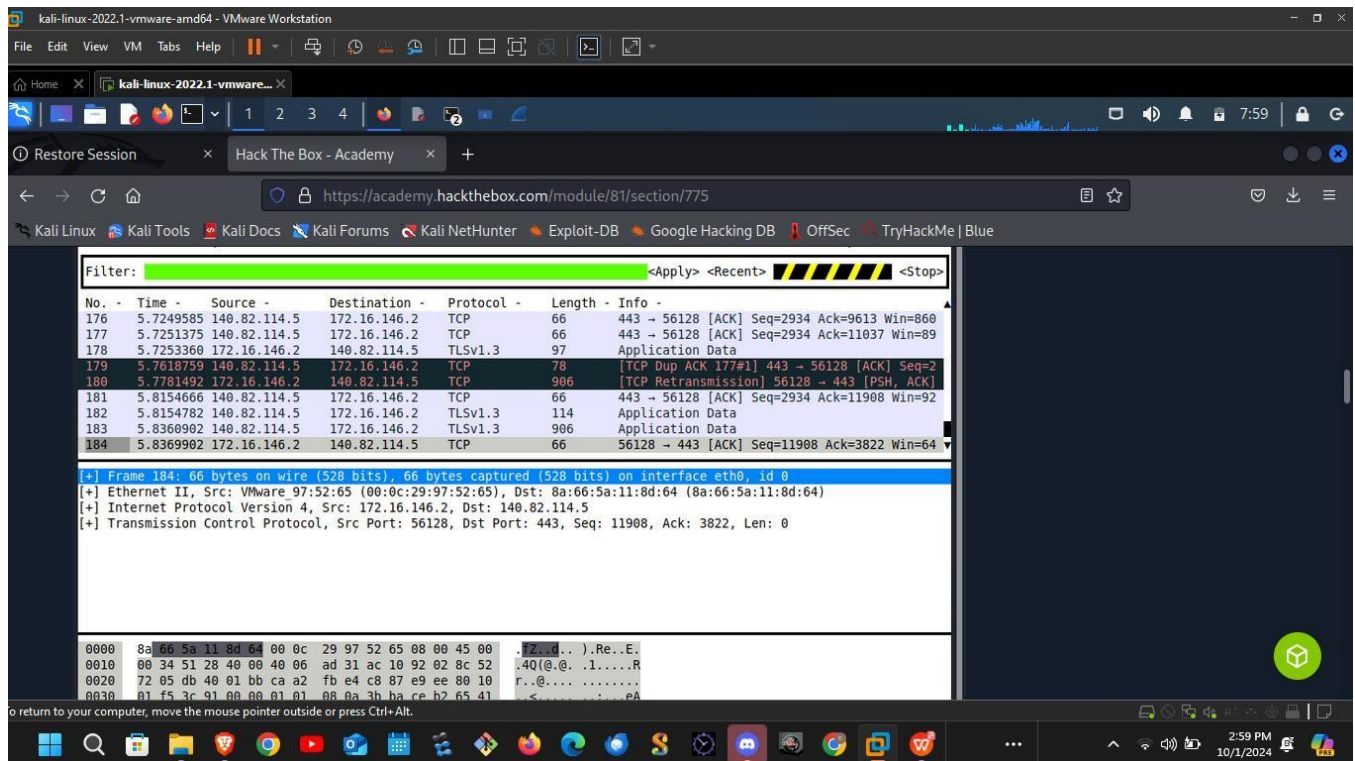
172.16.146.1

PART 3: Analysis with Wireshark

1. True or False: Wireshark can run on both Windows and Linux. **True**

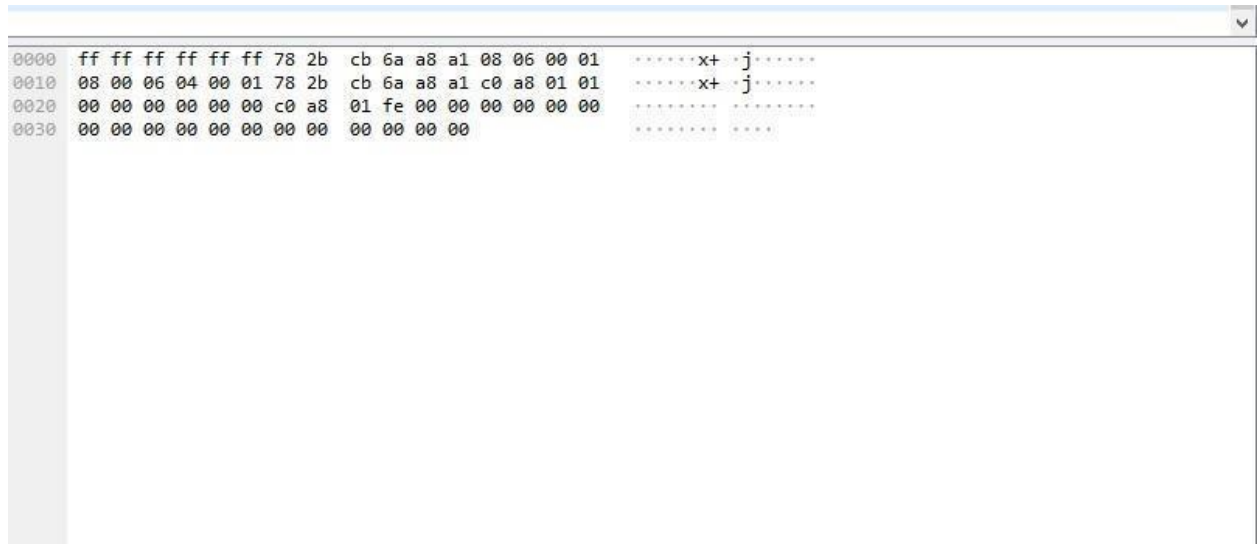
Wireshark is a free and open-source network traffic analyzer much like tcpdump but with a graphical interface. Wireshark is multi-platform and capable of capturing live data off many different interface types (to include WiFi, USB, and Bluetooth) and saving the traffic to several different formats. Wireshark allows the user to dive much deeper into the inspection of network packets than other tools. What makes Wireshark truly powerful is the analysis capability it provides, giving a detailed insight into the traffic.

2. Which Pane allows a user to see a summary of each packet grabbed during the capture? **Packet List**



3. Which pane provides your insight into the traffic you captured and displays it in both ASCII and Hex?

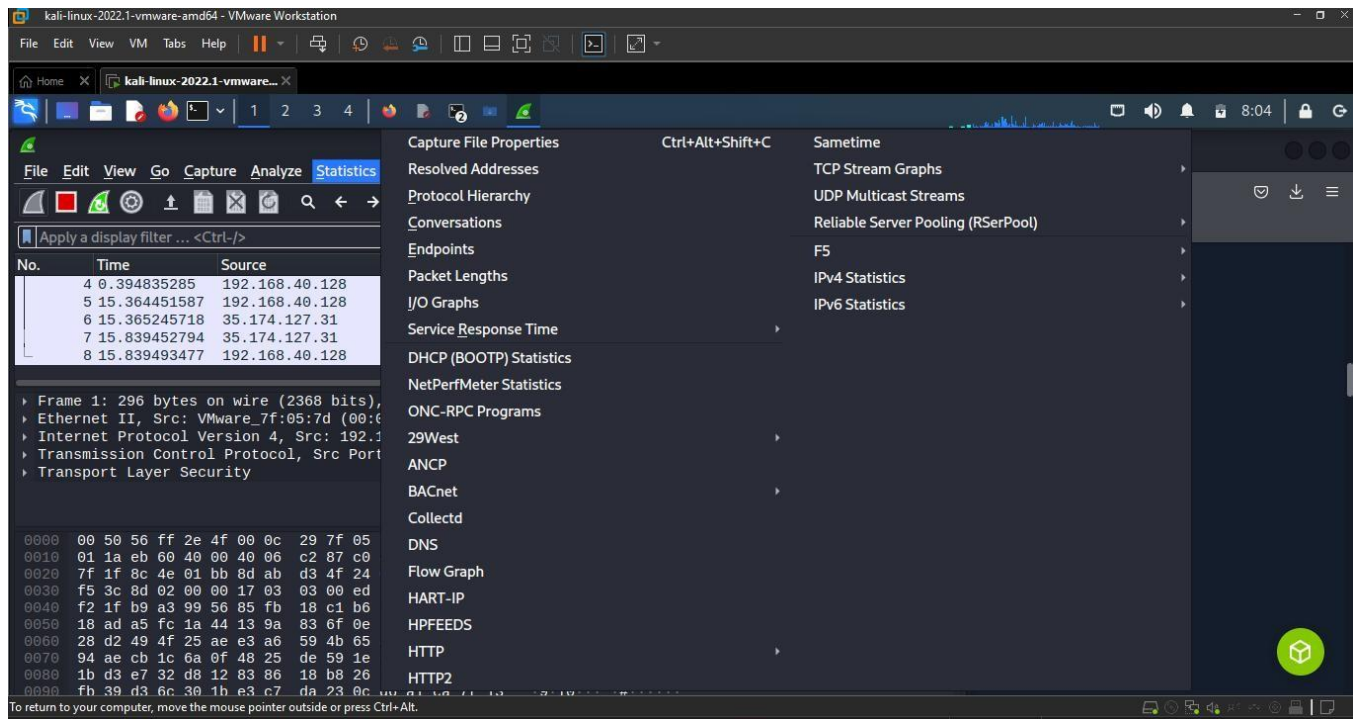
Packet Bytes



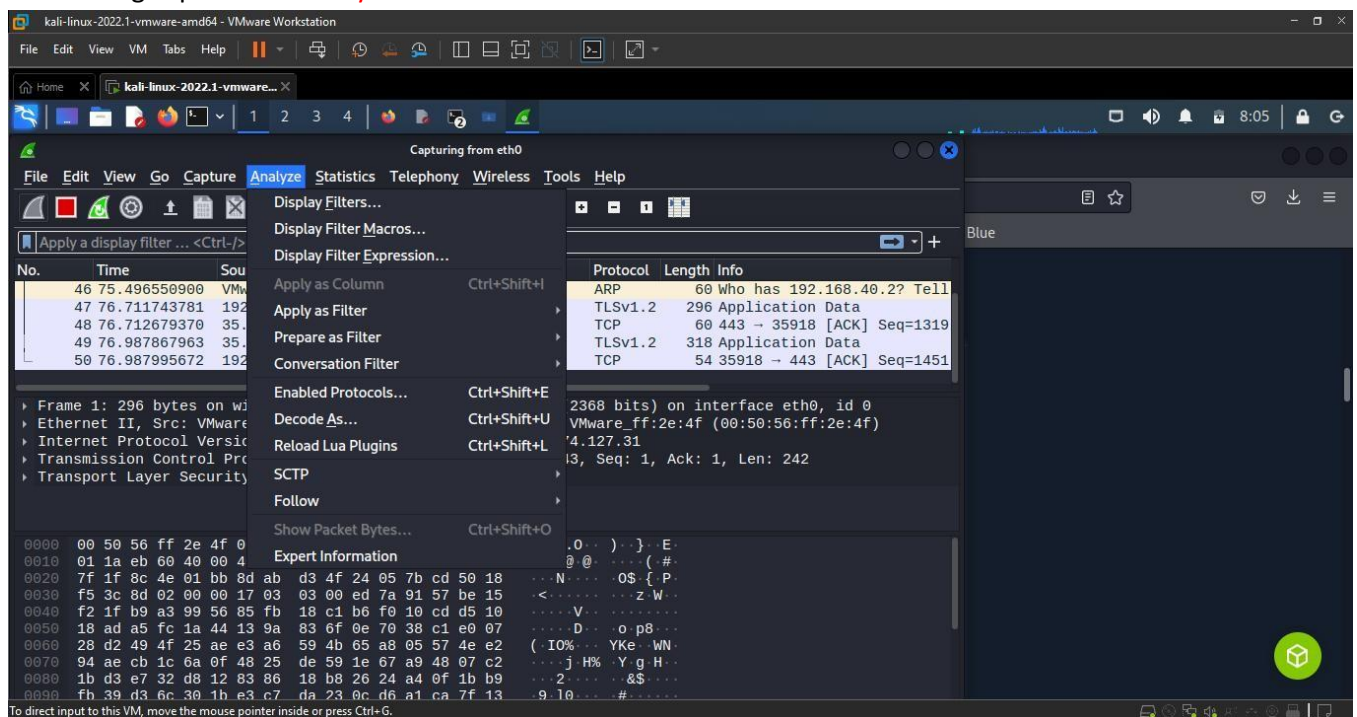
4. What switch is used with TShark to list possible interfaces to capture on? **-D**
5. What switch allows us to apply filters in TShark? **-f**
6. Is a capture filter applied before the capture starts or after? (answer before or after) **before**

Wireshark Advanced Usage

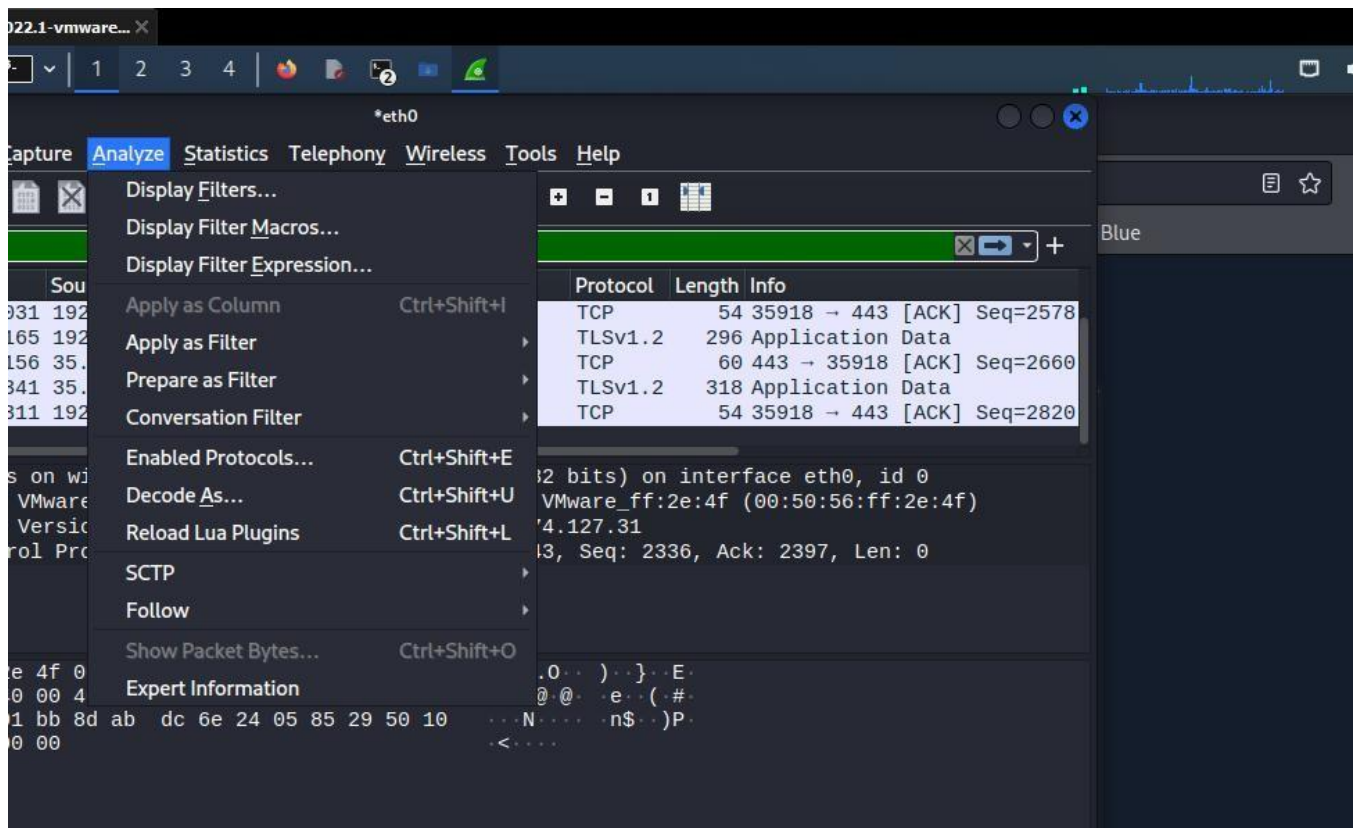
1. Which plugin tab can provide us with a way to view conversation metadata and even protocol breakdowns for the entire PCAP file? **Statistics**



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



- What stream oriented Transport protocol enables us to follow and rebuild conversations and the included data? **TCP**

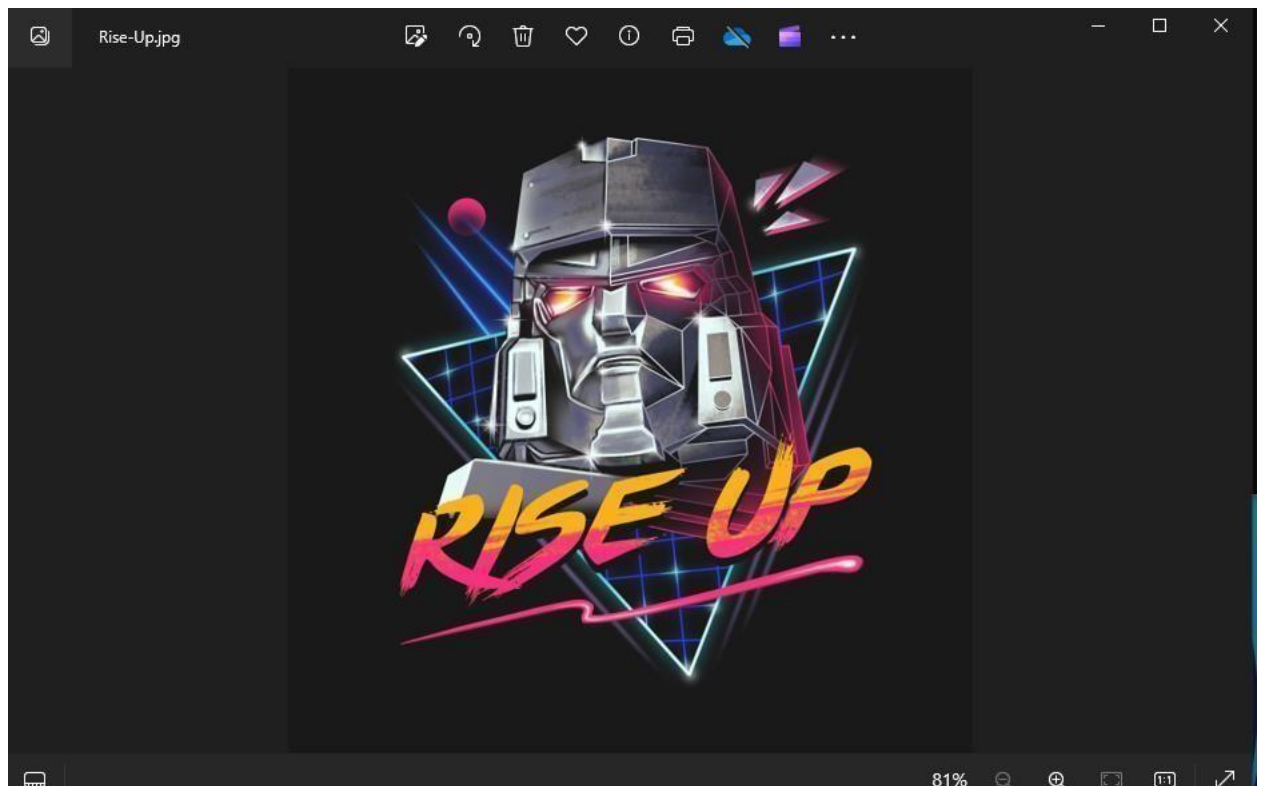


4. True or False: Wireshark can extract files from HTTP traffic. **True**
5. True or False: The ftp-data filter will show us any data sent over TCP port 21. **False**

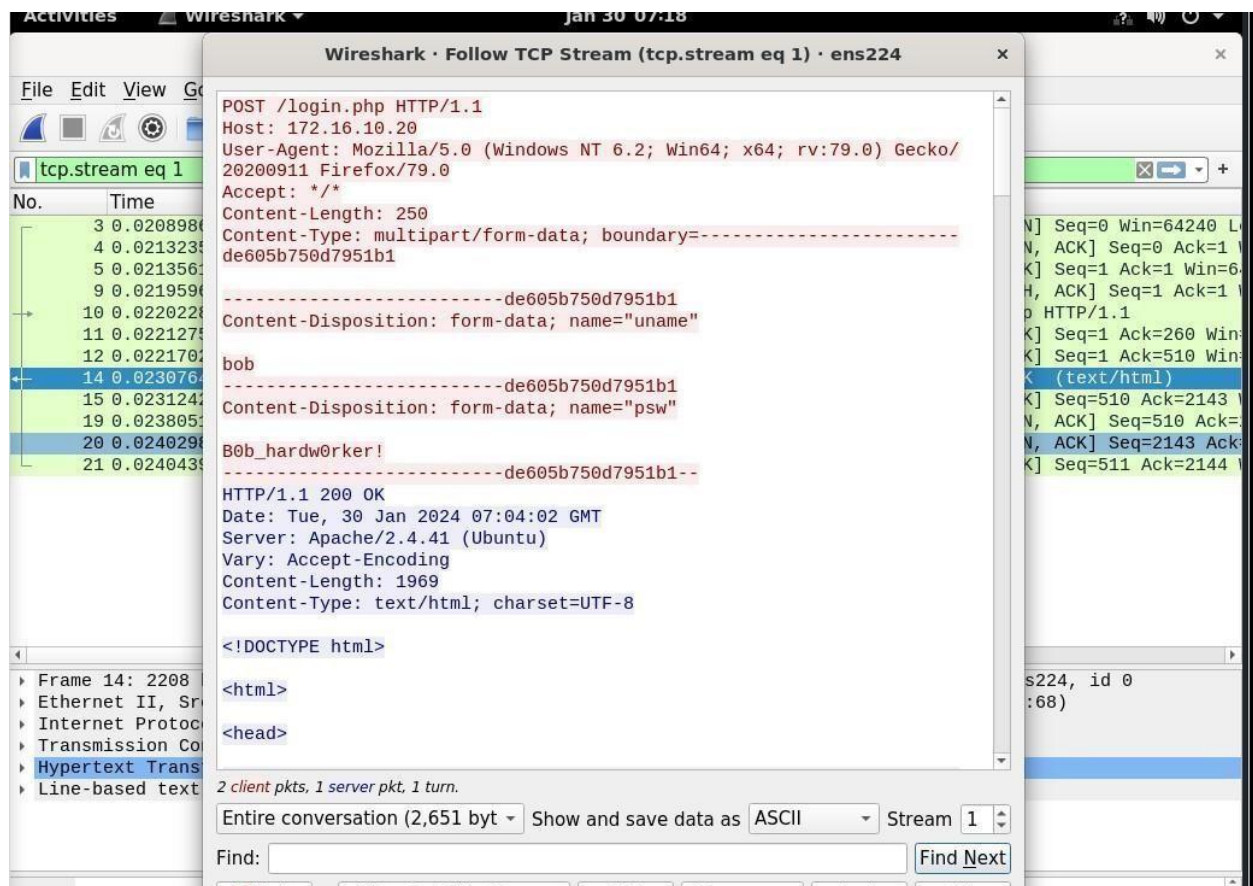
- **ftp-data** - Will show any data transferred over the data channel (port 20)
 - If we filter on a conversation and utilize **ftp-data**, we can capture anything sent during the conversation. We can reconstruct anything transferred by placing the raw data back into a new file and naming it appropriately.

Packet Inception, Dissecting Network Traffic with Wireshark

1. What was the filename of the image that contained a certain Transformer Leader? (name.filetype)



2. Which employee is suspected of performing potentially malicious actions in the live environment?



Guided Lab: Traffic Analysis Workflow

1. What was the name of the new user created on mrb3n's host? **Hacker**

```
c:\>net localgroup administrators hacker /add
net localgroup administrators hacker /add
The command completed successfully.
```

2. How many total packets were there in the Guided-analysis PCAP? **44**

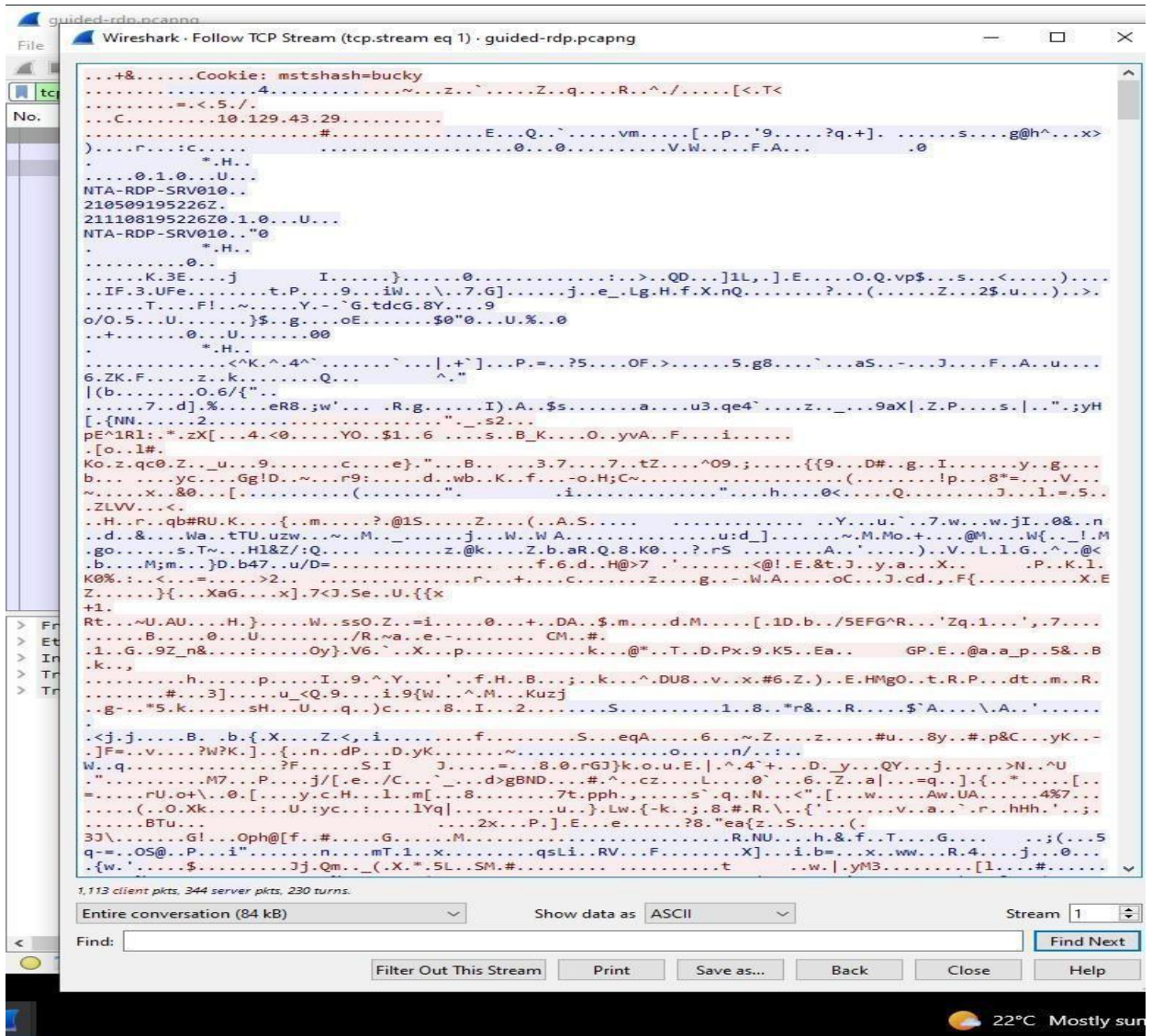
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
Frame	100.0	44	100.0	4445	666	0	0
Ethernet	100.0	44	13.9	616	92	0	0
Internet Protocol Version 4	90.9	40	18.0	800	119	0	0

3. What was the suspicious port that was being used? **4444**

No.	Time	Source	Destination	Protocol	src.p	dest.p	Length	Info
3	0.000215	10.129.43.29	10.129.43.4	TCP	506...	4444	66	50612 → 4444 [SYN] Seq=0 Wi
4	0.000270	10.129.43.4	10.129.43.29	TCP	4444	506...	66	4444 → 50612 [SYN, ACK] Seq
5	0.000415	10.129.43.29	10.129.43.4	TCP	506...	4444	60	50612 → 4444 [ACK] Seq=1 Ac
6	0.070797	10.129.43.29	10.129.43.4	TCP	506...	4444	175	50612 → 4444 [PSH, ACK] Seq
7	0.070843	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=1 Ac
8	10.676486	10.129.43.4	10.129.43.29	TCP	4444	506...	61	4444 → 50612 [PSH, ACK] Seq
9	10.745086	10.129.43.29	10.129.43.4	TCP	506...	4444	60	50612 → 4444 [ACK] Seq=122
10	10.745121	10.129.43.29	10.129.43.4	TCP	506...	4444	110	50612 → 4444 [PSH, ACK] Seq
11	10.745135	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=8 Ac
12	15.202665	10.129.43.4	10.129.43.29	TCP	4444	506...	63	4444 → 50612 [PSH, ACK] Seq
13	15.211515	10.129.43.29	10.129.43.4	TCP	506...	4444	64	50612 → 4444 [PSH, ACK] Seq
14	15.211538	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=17 A
15	15.261797	10.129.43.29	10.129.43.4	TCP	506...	4444	254	50612 → 4444 [PSH, ACK] Seq
16	15.261833	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=17 A
17	15.261986	10.129.43.29	10.129.43.4	TCP	506...	4444	841	50612 → 4444 [PSH, ACK] Seq
18	15.261992	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=17 A
19	21.584905	10.129.43.4	10.129.43.29	TCP	4444	506...	61	4444 → 50612 [PSH, ACK] Seq
20	21.626201	10.129.43.29	10.129.43.4	TCP	506...	4444	68	50612 → 4444 [PSH, ACK] Seq
21	21.626254	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=24 A
22	22.582605	10.129.43.4	10.129.43.29	TCP	4444	506...	58	4444 → 50612 [PSH, ACK] Seq
23	22.646451	10.129.43.29	10.129.43.4	TCP	506...	4444	60	50612 → 4444 [ACK] Seq=1189
24	22.646488	10.129.43.29	10.129.43.4	TCP	506...	4444	255	50612 → 4444 [PSH, ACK] Seq
25	22.646503	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=28 A
26	22.646648	10.129.43.29	10.129.43.4	TCP	506...	4444	314	50612 → 4444 [PSH, ACK] Seq
27	22.646653	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=28 A
28	41.703799	10.129.43.4	10.129.43.29	TCP	4444	506...	85	4444 → 50612 [PSH, ACK] Seq
29	41.720894	10.129.43.29	10.129.43.4	TCP	506...	4444	86	50612 → 4444 [PSH, ACK] Seq
30	41.720929	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=59 A
31	41.783461	10.129.43.29	10.129.43.4	TCP	506...	4444	99	50612 → 4444 [PSH, ACK] Seq
32	41.783497	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=59 A
38	51.245569	10.129.43.4	10.129.43.29	TCP	4444	506...	96	4444 → 50612 [PSH, ACK] Seq
39	51.247032	10.129.43.29	10.129.43.4	TCP	506...	4444	97	50612 → 4444 [PSH, ACK] Seq
40	51.247072	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=101
41	51.309247	10.129.43.29	10.129.43.4	TCP	506...	4444	99	50612 → 4444 [PSH, ACK] Seq
42	51.309270	10.129.43.4	10.129.43.29	TCP	4444	506...	54	4444 → 50612 [ACK] Seq=101

Decrypting RDP connections

1. What user account was used to initiate the RDP connection? **ucky**



<https://academy.hackthebox.com/achievement/868719/81>

CONCLUSION

In this lab, I learned basics of network traffic analysis, how to use tcpdump and Wireshark for implementing network traffic analysis.

As most command I had never interacted with they really brought to me a challenge. However, with the help of online research and study, I was able to overcome this.

