

## Week 2 Assignment 2

Course: [Cloud and Network Security - C1-2026](#)

Student Name: [Bussllus Bertrand](#)

Student Number: [CS-CNS11-26004](#)

Wednesday, January 28, 2026

Week two Assignment two:

Class exercise: Introduction to Network Traffic Analysis

## Week 2 Assignment 2

### Table of Contents

Introduction .....	3
Module Questions & Answers .....	3
Networking Primer - Layers 1-4.....	3
Networking Primer - Layers 5-7.....	8
Tcpdump Fundamentals .....	12
Fundamentals Lab .....	15
Tcpdump Packet Filtering.....	20
Interrogating Network Traffic With Capture and Display Filters .....	21
Analysis with Wireshark.....	22
Wireshark Advanced Usage .....	25
Packet Inception, Dissecting Network Traffic With Wireshark .....	28
Guided Lab: Traffic Analysis Workflow .....	29
Decrypting RDP connections.....	30
Verification of Completion.....	31
Shareable Link .....	32
Completion Screenshot .....	32
Conclusion & Reflection .....	32

## Introduction

This report details the successful completion of the "Intro to Network Traffic Analysis module from Hack the box Academy. Network traffic analysis is a cornerstone of modern cybersecurity, providing the foundational skills necessary to detect malicious activity, respond to security incidents, and diagnose network health issues. By examining the data that flows across a network, security professionals can uncover hidden threats, understand attack methodologies, and verify the integrity of their digital infrastructure.

The core objective of this module was to establish a strong theoretical and practical understanding of network traffic analysis. This involved exploring the fundamentals of network protocols and gaining hands-on experience with industry-standard tools. The exercise focused heavily on the practical application of Wireshark for graphical packet analysis and tcpdump for command-line traffic capture.

The following sections provide a detailed analysis of the module's exercises, demonstrating the process used to solve each challenge.

## Module Questions & Answers

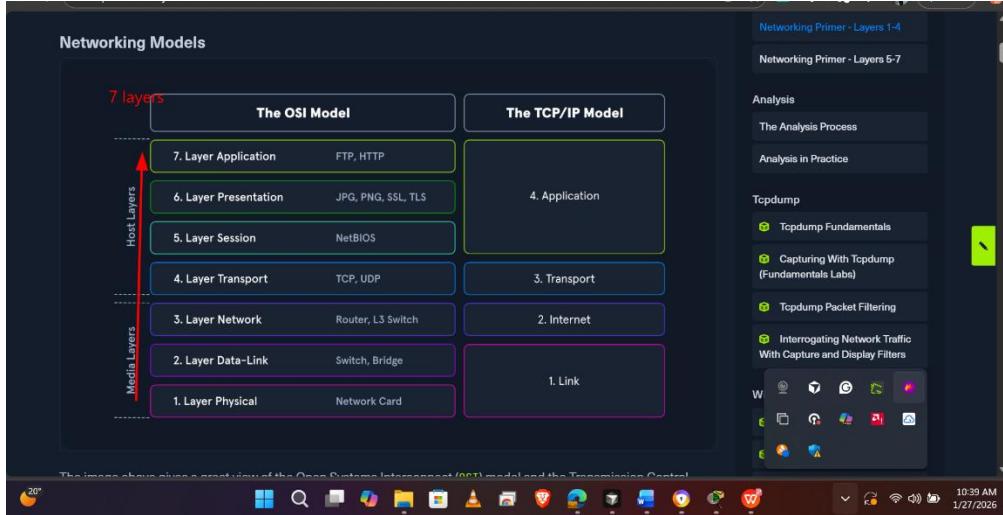
This section provides a systematic walkthrough of the questions presented in the module and the answers, together with supporting evidence.

### Networking Primer - Layers 1-4

**How many layers does the OSI model have?**

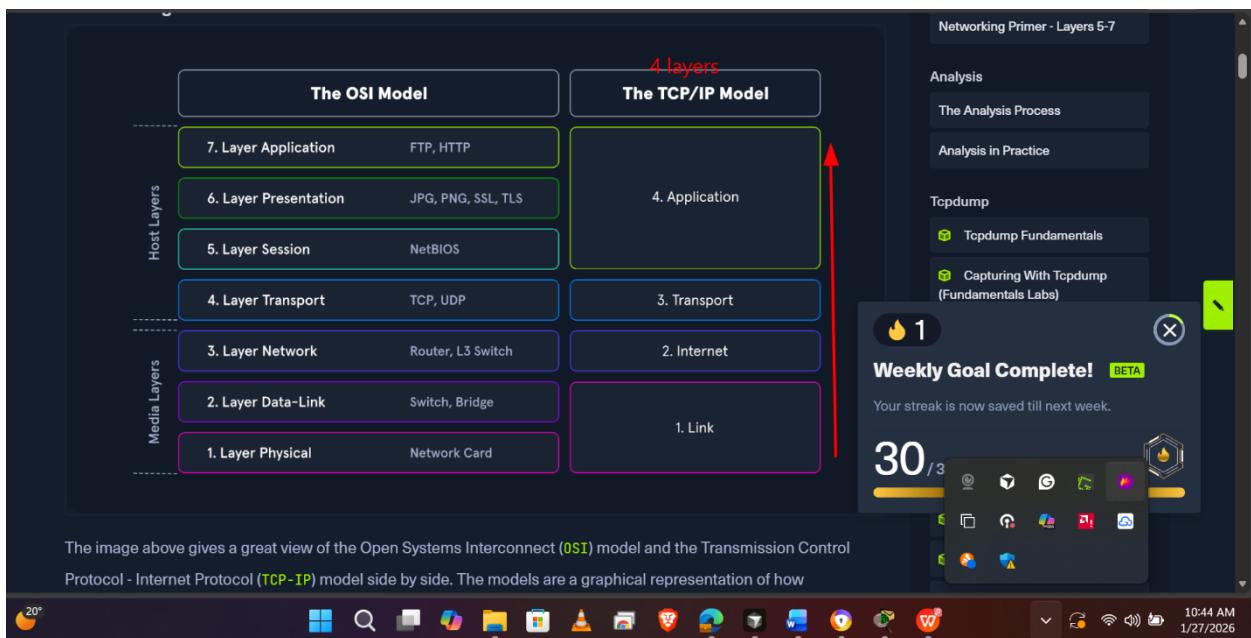
Answer: 7

## Week 2 Assignment 2



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

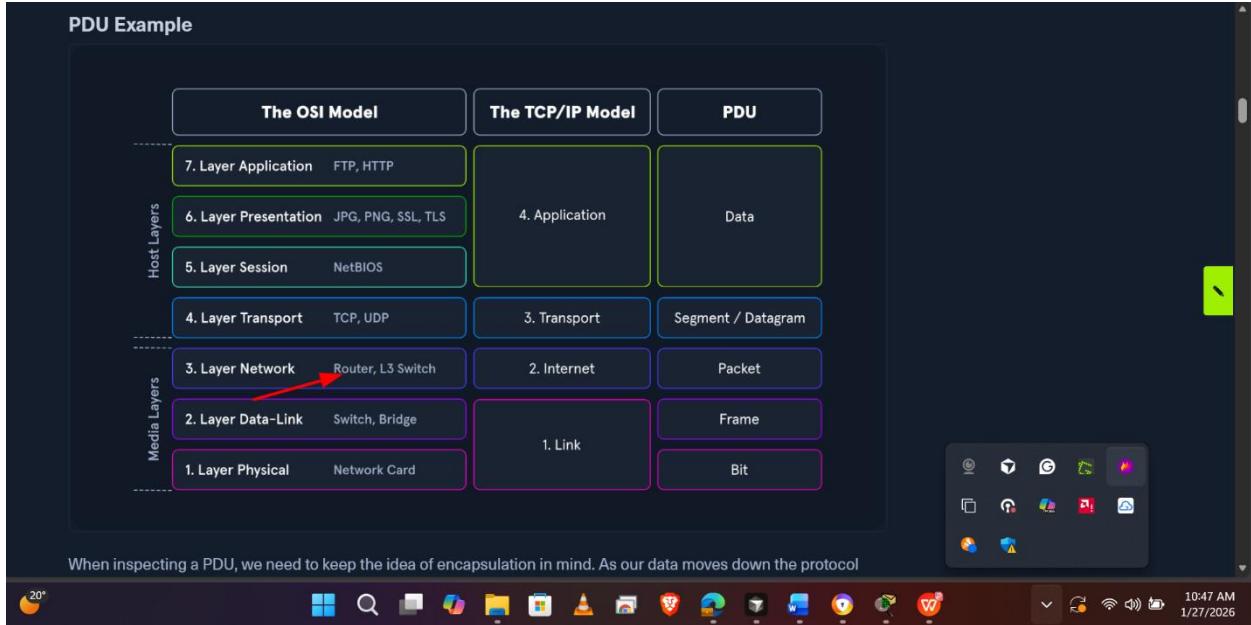
Answer: 4



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

Answer: False

## Week 2 Assignment 2



What addressing mechanism is used at the Link Layer of the TCP/IP model?

Answer: Mac-address

**Mac-Address**

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=400<CHANNEL_IO>
    ether 88:6:5:a:11:bb:36
    inet6 fe80::49fe:e3c:bf36:9bb1%en0 prefixlen 64 secured scopeid 0x6
    inet 192.168.86.243 netmask 0xffffffff broadcast 192.168.86.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```

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 egress interface, and it is routed to the correct network. At layer two, this looks as though the PDU is addressed to the router interface, and the router will take the layer three address into account when determining where to send it next. Once it makes a choice, it strips the encapsulation at layer two and replaces it with new information that indicates the next physical address in the route.

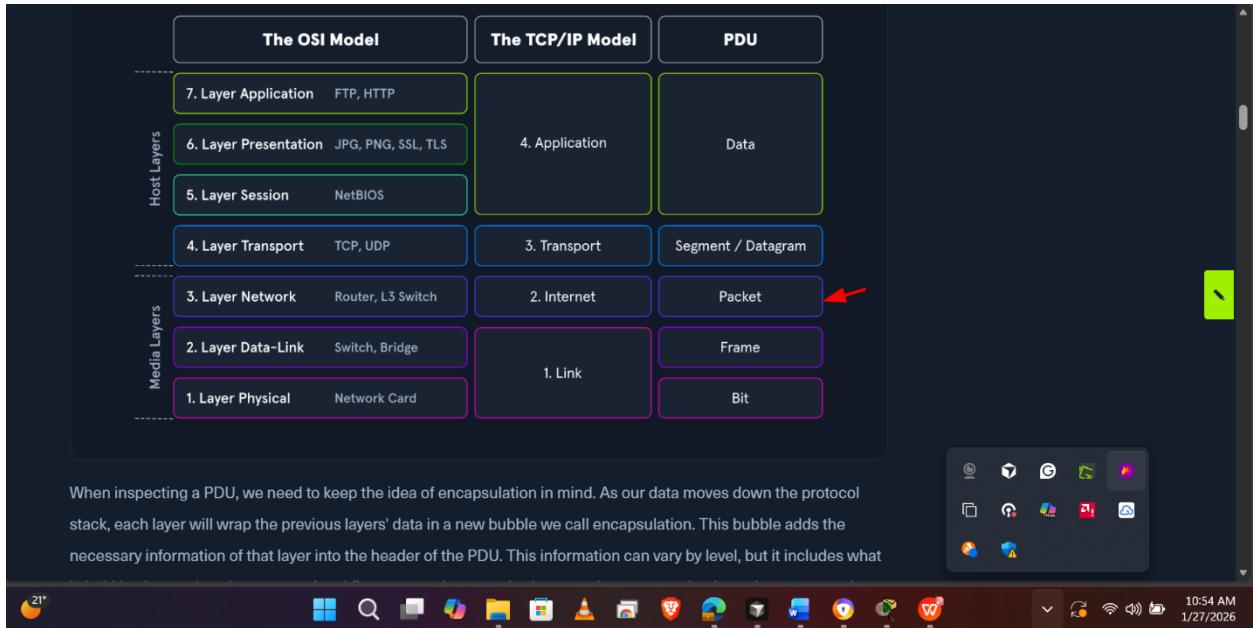
### IP Addressing

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. By nature, IP is a connectionless protocol that provides no assurances that data will

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

Answer: 3

## Week 2 Assignment 2



**What addressing mechanism utilizes a 32-bit address?**

Answer: ipv4

IPv4 addressing is the core method of routing packets across networks to hosts located outside our immediate vicinity. The image below shows us an example of an IPv4 address by the **green** arrow.

```
IP Address

en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=400<CHANNEL_IO>
    ether 88:66:5a:11:bb:36
    inet6 fe80::49f:3c:bf36:9bb1%en0 prefixlen 64 secured scopeid 0x6
        inet 192.168.86.243 netmask 0xffffffff broadcast 192.168.86.255
            nd6 options=201<PERFORMNUD,DAD>
            media: autoselect
            status: active
```

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.

IPv6

After a little over a decade of utilizing IPv4, it was determined that we had quickly exhausted the pool of usable IP addresses. With such large chunks sectioned off for special use or private addressing, the world had quickly used up

**What Transport layer protocol is connection oriented?**

Answer: TCP

## Week 2 Assignment 2

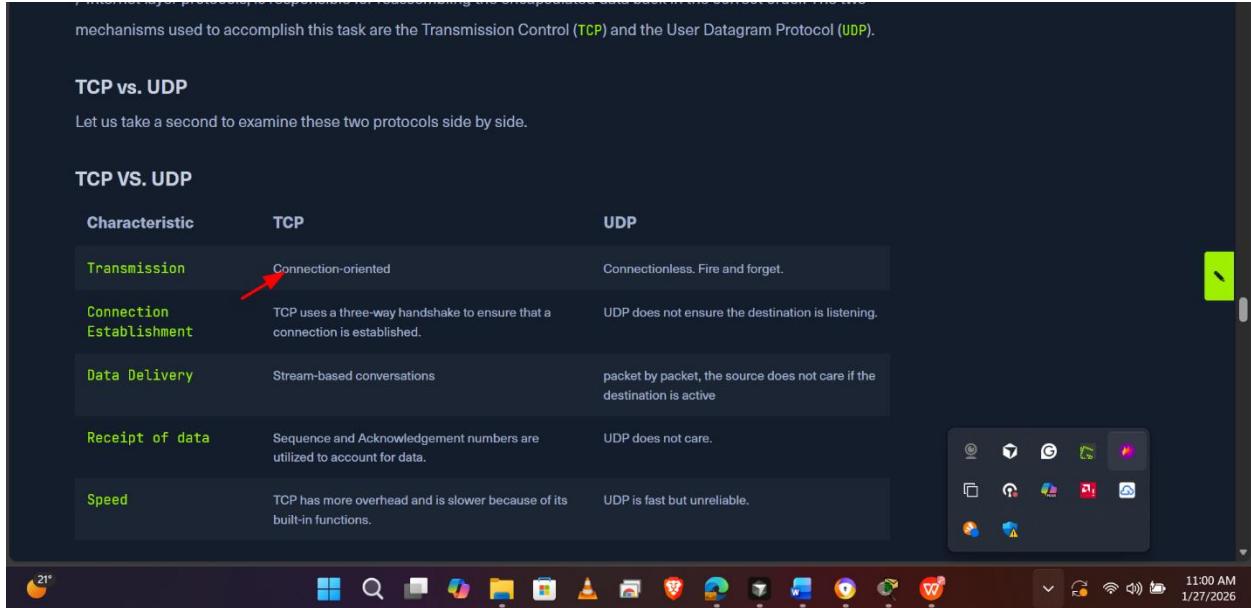
mechanisms used to accomplish this task are the Transmission Control ([TCP](#)) and the User Datagram Protocol ([UDP](#)).

**TCP vs. UDP**

Let us take a second to examine these two protocols side by side.

**TCP VS. UDP**

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.



### What Transport Layer protocol is considered unreliable?

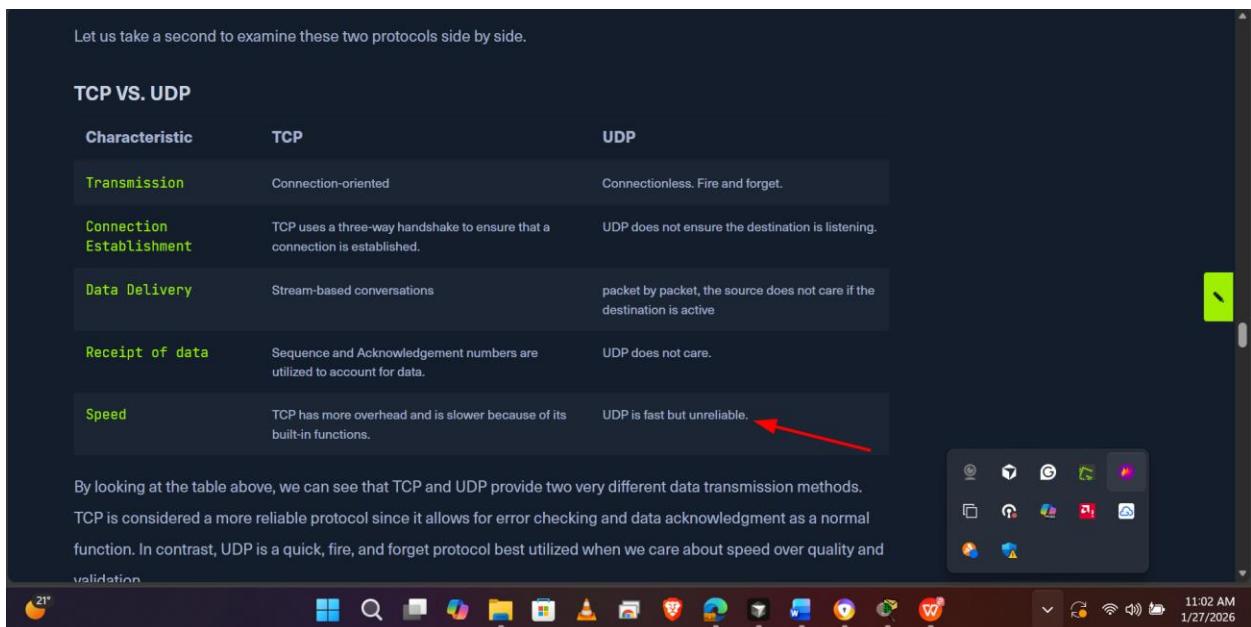
Answer: UDP

Let us take a second to examine these two protocols side by side.

**TCP VS. UDP**

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.

By looking at the table above, we can see that TCP and UDP provide two very different data transmission methods. TCP is considered a more reliable protocol since it allows for error checking and data acknowledgment as a normal function. In contrast, UDP is a quick, fire, and forget protocol best utilized when we care about speed over quality and validation.



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

Answer: ACK

## Week 2 Assignment 2

The terminal window displays the following text:

1. This is a synchronization packet. It will only be set in the first packet from host and server and enables establishing a session by allowing both ends to agree on a sequence number to start communicating with.
2. This is crucial for the tracking of packets. Along with the sequence number sync, many other options are negotiated in this phase to include window size, maximum segment size, and selective acknowledgments.
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.
  1. The server will also include any changes to the TCP options it requires set in the options fields of the TCP header.
  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.

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



## Networking Primer - Layers 5-7

### What is the default operational mode method used by FTP?

Answer: Active

The terminal window displays the following text:

File Transfer Protocol (**FTP**) is an Application Layer protocol that enables quick data transfer between computing devices. FTP can be utilized from the command-line, web browser, or through a graphical FTP client such as FileZilla. FTP itself is established as an insecure protocol, and most users have moved to utilize tools such as SFTP to transfer files through secure channels. As a note moving into the future, most modern web browsers have phased out support for FTP as of 2020.

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.

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.

### FTP Command & Response Examples



### FTP utilizes what two ports for command and data transfer? (separate the two numbers with a space)

Answer: 20 21

## Week 2 Assignment 2

File Transfer Protocol (FTP) is an Application Layer protocol that enables quick data transfer between computing devices. FTP can be utilized from the command-line, web browser, or through a graphical FTP client such as FileZilla. FTP itself is established as an insecure protocol, and most users have moved to utilize tools such as SFTP to transfer files through secure channels. As a note moving into the future, most modern web browsers have phased out support for FTP as of 2020.

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.

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.

### FTP Command & Response Examples



## Does SMB utilize TCP or UDP as its transport layer protocol?

Answer: TCP

This is not an exhaustive list of the possible FTP control commands that could be seen. These can vary based on the FTP application or shell in use. For more information on FTP, see [RFC:959](#).

### SMB

Server Message Block (SMB) is a protocol most widely seen in Windows enterprise environments that enables sharing resources between hosts over common networking architectures. SMB is a connection-oriented protocol that requires user authentication from the host to the resource to ensure the user has correct permissions to use that 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.

As a user, SMB provides us easy and convenient access to resources like printers, shared drives, authentication servers, and more. For this reason, SMB is very attractive to potential attackers as well.

Like any other application that uses TCP as its transport mechanism, it will perform standard functions like the three-way handshake and acknowledging received packets. Let us take a second to look at some SMB traffic to familiarize ourselves.



## SMB has moved to using what TCP port?

Answer: 445

## Week 2 Assignment 2

This is not an exhaustive list of the possible FTP control commands that could be seen. These can vary based on the FTP application or shell in use. For more information on FTP, see [RFC:959](#).

### SMB

Server Message Block ([SMB](#)) is a protocol most widely seen in Windows enterprise environments that enables sharing resources between hosts over common networking architectures. SMB is a connection-oriented protocol that requires user authentication from the host to the resource to ensure the user has correct permissions to use that 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.

As a user, SMB provides us easy and convenient access to resources like printers, shared drives, authentication servers, and more. For this reason, SMB is very attractive to potential attackers as well.

Like any other application that uses TCP as its transport mechanism, it will perform standard functions like the three-way handshake and acknowledging received packets. Let us take a second to look at some SMB traffic to familiarize ourselves.



### Hypertext Transfer Protocol uses what well known TCP port number?

Answer: 80

our applications. It takes many different applications and services to maintain a network connection and ensure that data can be transferred between hosts. This section will outline just a vital few.

### HTTP

Hypertext Transfer Protocol ([HTTP](#)) is a stateless Application Layer protocol that has been in use since 1990. HTTP enables the transfer of data in clear text between a client and server over TCP. The client would send an HTTP request to the server, asking for a resource. A session is established, and the server responds with the requested 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.

#### HTTP Methods

To perform operations such as fetching webpages, requesting items for download, or posting your most recent tweet all require the use of specific methods. These methods define the actions taken when requesting a URI. Methods:

Method	Description
HEAD	<a href="#">required</a> is a safe method that requests a response from the server similar to a Get request except that the message body is not included. It is a great way to acquire more information about the server and its operational status.



### What HTTP method is used to request information and content from the webserver?

Answer: GET

## Week 2 Assignment 2

The screenshot shows a Windows desktop environment. On the left, a browser window displays a page titled "HTTP Methods" with a table of methods and their descriptions. A red arrow points from the text "from the server" in the GET description to the Wireshark interface on the right. On the right, the Wireshark application is open, showing a list of network protocols and a packet list pane at the bottom. The taskbar at the bottom of the screen shows various pinned icons.

Method	Description
HEAD	<b>required</b> is a safe method that requests a response from the server similar to a Get request except that the message body is not included. It is a great way to acquire more information about the server and its operational status.
GET	<b>required</b> Get is the most common method used. It requests information and content from the server. For example, GET <a href="http://10.1.1.1/Webserver/index.html">http://10.1.1.1/Webserver/index.html</a> requests the index.html page from the server based on our supplied URI.
POST	<b>optional</b> Post is a way to submit information to a server based on the fields in the request. For example, submitting a message to a Facebook post or website forum is a POST action. The actual action taken can vary based on the server, and we should pay attention to the response codes sent back to validate the action.
PUT	<b>optional</b> Put will take the data appended to the message and place it under the requested URI. If an item does not exist there already, it will create one with the supplied data. If an object already exists, the new PUT will be considered the most up-to-date, and the object will be modified to match. The easiest way to visualize the differences between PUT and POST is to think of it like this; PUT will create or update an object at the URI supplied, while POST will create child entities at the provided URL. The action taken can be compared with the difference between creating a new file vs. writing comments about that file on the same page.
DELETE	<b>optional</b> Delete does as the name implies. It will remove the object at the given URI.
TRACE	<b>optional</b> Allows for remote server diagnosis. The remote server will echo the same request that was sent in its response if the TRACE method is enabled.
OPTIONS	<b>optional</b> The Options method can gather information on the supported HTTP methods the server recognizes. This way, we can determine the requirements for interacting with a specific resource or server without actually requesting data or objects

### What web based protocol uses TLS as a security measure?

Answer: HTTPS

The screenshot shows a Windows desktop environment. On the left, a browser window displays a page with text explaining that HTTPS is a modification of HTTP that uses Transport Layer Security (TLS) for encryption. A red arrow points from the word "TLS" in the text to the Wireshark interface on the right. On the right, the Wireshark application is open, showing a packet list pane with several network packets. The taskbar at the bottom of the screen shows various pinned icons.

For more information on HTTP as a protocol or how it operates, see [RFC:2616](#).

### 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 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.

Even though it is HTTP at its base, HTTPS utilizes ports 443 and 8443 instead of the standard port 80. This is a simple way for the client to signal the server that it wishes to establish a secure connection. Let's look at an output of HTTPS traffic and discern how a [TLS handshake](#) functions for a minute.

### TLS Handshake Via HTTPS

Source	Destination	Protocol	Length	Info
192.168.86.243	194.20.55.68	TCP	78	68201 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSeq=2199353520 TSecr=0
194.20.55.68	192.168.86.243	TCP	66	443 → 68201 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1480 SACK_PERM=1 WS=18..
192.168.86.243	194.20.55.68	TCP	54	68201 → 443 [ACK] Seq=1 Ack=1 Win=702144 Len=0

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

Answer: True

## Week 2 Assignment 2

way for the client to signal the server that it wishes to establish a secure connection. Let's look at an output of Wireshark traffic and discern how a TLS handshake functions for a minute.

**TLS Handshake Via HTTPS**

Source	Destination	Protocol	Length	Info
192.168.86.243	184.28.55.68	TCP	78	68201 -> 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1400 WS=64 TStamp=2199335320 TSecr.=0
184.28.55.68	192.168.86.243	TCP	66	443 -> 68201 [SYN, ACK] Seq=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=0
192.168.86.243	184.28.55.68	TCP	54	68201 -> 443 [ACK] Seq=1 Win=262144 Len=0
184.28.55.68	192.168.86.243	TCP	54	443 -> 68201 [ACK] Seq=1 Win=262144 Len=0
184.28.55.68	192.168.86.243	TLSv1.3	60	ClientHello
184.28.55.68	192.168.86.243	TLSv1.3	54	443 -> 68201 [ACK] Seq=1 Win=67584 Len=0
192.168.86.243	184.28.55.68	TLSv1.3	266	Server Hello, Change Cipher Spec, Application Data
192.168.86.243	184.28.55.68	TLSv1.3	54	68201 -> 443 [ACK] Seq=554 Ack=213 Win=261888 Len=0
192.168.86.243	184.28.55.68	TLSv1.3	118	Change Cipher Spec, Application Data
192.168.86.243	184.28.55.68	TLSv1.3	146	Application Data
192.168.86.243	184.28.55.68	TLSv1.3	179	Application Data
192.168.86.243	184.28.55.68	TLSv1.3	187	Application Data
192.168.86.243	184.28.55.68	TLSv1.3	60	68201 -> 443 [ACK] Seq=313 Ack=618 Win=67584 Len=0
192.168.86.243	184.28.55.68	TLSv1.3	68	443 -> 68201 [ACK] Seq=213 Ack=718 Win=67584 Len=0
192.168.86.243	184.28.55.68	TLSv1.3	575	Application Data, Application Data
192.168.86.243	184.28.55.68	TCP	54	68201 -> 443 [ACK] Seq=1979 Ack=734 Win=261568 Len=0
192.168.86.243	184.28.55.68	TLSv1.3	85	Application Data
184.28.55.68	192.168.86.243	TCP	69	443 -> 68201 [ACK] Seq=734 Ack=1926 Win=69632 Len=0
184.28.55.68	192.168.86.243	TCP	68	68201 -> 443 [ACK] Seq=734 Ack=1979 Win=69632 Len=0
184.28.55.68	192.168.86.243	TCP	68	443 -> 68201 [ACK] Seq=2810 Win=69632 Len=0
184.28.55.68	192.168.86.243	TLSv1.3	1124	Application Data
184.28.55.68	192.168.86.243	TLSv1.3	1445	Application Data
184.28.55.68	192.168.86.243	TLSv1.3	1445	Application Data
184.28.55.68	192.168.86.243	TLSv1.3	1445	Application Data
192.168.86.243	184.28.55.68	TCP	54	68201 -> 443 [ACK] Seq=2010 Ack=5077 Win=256896 Len=0

In the first few packets, we can see that the client establishes a session to the server using port 443 boxed in blue. This signals the server that it wishes to use HTTPS as the application communication protocol.

Once a session is initiated via TCP, a TLS ClientHello is sent next to begin the TLS handshake. During the handshake, several parameters are agreed upon, including session identifier, peer x509 certificate, compression algorithm to be used, etc.

## Tcpdump Fundamentals

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

Answer: 174.143.213.184

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

Answer: Relative

```

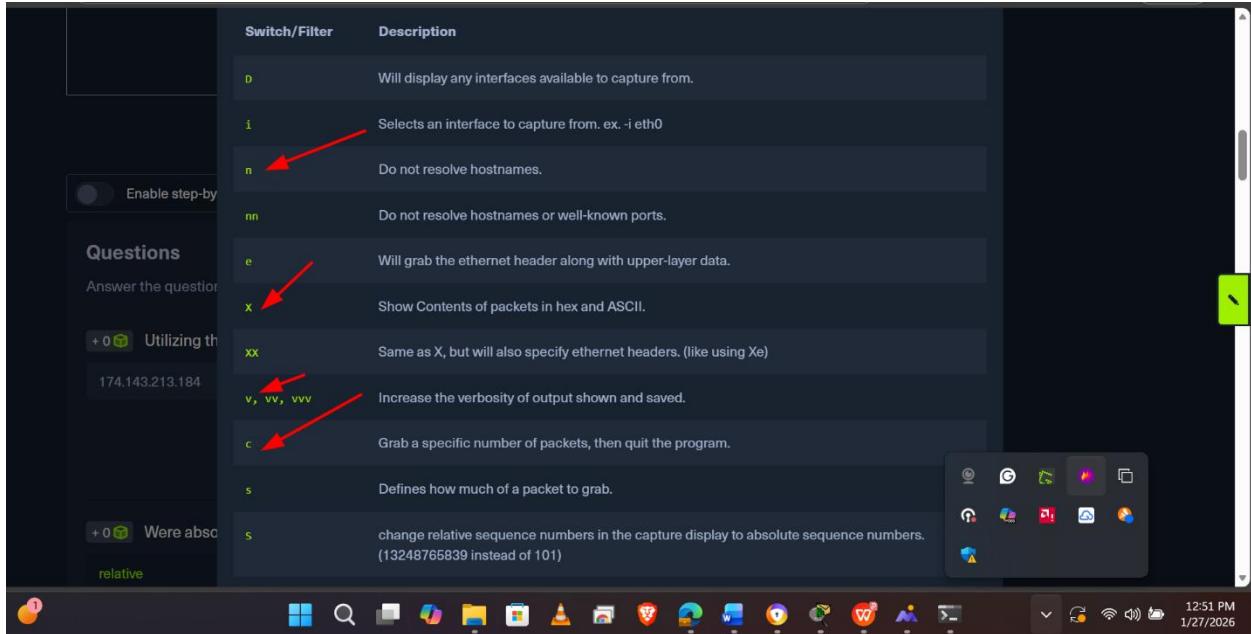
HTTP.cap
file HTTP.cap, link-type EN10MB (Ethernet), snapshot length 65535
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [S], seq 2387613953, win 5840, options [mss 1460,sackOK,TSAckOK,TSval 2216538 ecn 0,nop,wscale 7], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [S.], seq 334080264, ack 2387613953, win 5792, options [mss 1460,sackOK,TSval 835172936 ecn 2216538,nop,wscale 6], length 0
# IP 192.168.1.140.57678: Flags [.], ack 1, win 46, options [nop,nop,TSval 2216543 ecr 835172936], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 1, win 46, options [nop,nop,TSval 2216543 ecr 835172936], length 134: HTTP: GET /images/layout/logo.png HTTP/1.1
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 135, win 108, options [nop,nop,TSval 835172948 ecr 2216543], length 0
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 135, win 108, options [nop,nop,TSval 2216543 ecr 835172948], length 1448: HTTP: HTTP/1.1 200 OK
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 449, win 69, options [nop,nop,TSval 2216548 ecr 835172948], length 0
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 449, win 69, options [nop,nop,TSval 2216548 ecr 835172948], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 449:>2897, ack 135, win 108, options [nop,nop,TSval 835172948 ecr 2216543], length 1448: HTTP
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], seq 2897:>4345, ack 135, win 108, options [nop,nop,TSval 835172948 ecr 2216543], length 1448: HTTP
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 4345, win 114, options [nop,nop,TSval 2216548 ecr 835172948], length 0
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], seq 4345:>5793, ack 135, win 108, options [nop,nop,TSval 835172961 ecr 2216548], length 1448: HTTP
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 5793, win 137, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 0
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 135, win 108, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 1448: HTTP
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 135, win 108, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 0
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 135, win 108, options [nop,nop,TSval 835172961 ecr 2216548], length 1448: HTTP
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 689:>10137, ack 135, win 108, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 1448: HTTP
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 10137, win 204, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 11585, win 227, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 0
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 11585, win 227, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 1448: HTTP
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], ack 11585, win 250, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 1448: HTTP
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 135, win 108, options [nop,nop,TSval 2216553 ecr 835172961 ecr 2216548], length 1448: HTTP
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 2032:>1481, ack 135, win 108, options [nop,nop,TSval 835172973 ecr 2216553], length 1448: HTTP
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 4481, win 272, options [nop,nop,TSval 2216557 ecr 835172973], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 34481:>19929, ack 135, win 108, options [nop,nop,TSval 835172973 ecr 2216553], length 1448: HTTP
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 19929, win 295, options [nop,nop,TSval 2216558 ecr 835172973], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 19929:>17377, ack 135, win 108, options [nop,nop,TSval 835172973 ecr 2216553], length 1448: HTTP
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 17377, win 318, options [nop,nop,TSval 2216558 ecr 835172973], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 17377:>18825, ack 135, win 108, options [nop,nop,TSval 835172973 ecr 2216553], length 1448: HTTP
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 18825, win 340, options [nop,nop,TSval 2216558 ecr 835172973 ecr 2216553], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 20823:>20733, ack 135, win 108, options [nop,nop,TSval 835172973 ecr 2216553], length 144
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 20733, win 363, options [nop,nop,TSval 2216558 ecr 835172973], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 20733:>21721, ack 135, win 108, options [nop,nop,TSval 835172973 ecr 2216553], length 144
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 21721, win 385, options [nop,nop,TSval 2216558 ecr 835172973], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 21721:>22046, ack 135, win 108, options [nop,nop,TSval 835172973 ecr 2216553], length 32
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 22046, win 408, options [nop,nop,TSval 2216558 ecr 835172974], length 0
# IP 174.143.213.184.80 -> 192.168.1.140.57678: Flags [.], seq 22046:>22047, ack 136, win 108, options [nop,nop,TSval 835172980 ecr 2216558], length 0
# IP 192.168.1.140.57678 -> 174.143.213.184.80: Flags [.], ack 22047, win 408, options [nop,nop,TSval 2216563 ecr 835172986], length 0

```

## Week 2 Assignment 2

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.

Answer: -nvXc 100



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)

Answer: sudo tcpdump -Xr /tmp/capture.pcap

## Week 2 Assignment 2

```
i      Selects an interface to capture from. ex. -i eth0  
n      Do not resolve hostnames.  
nn     Do not resolve hostnames or well-known ports.  
e      Will grab the ethernet header along with upper-layer data.  
x      Show Contents of packets in hex and ASCII.  
xx    Same as X, but will also specify ethernet headers. (like using Xe)  
v, vv, vvv Increase the verbosity of output shown and saved.  
c      Grab a specific number of packets, then quit the program.  
s      Defines how much of a packet to grab.  
S      change relative sequence numbers in the capture display to absolute sequence numbers.  
(13248765839 instead of 101)  
q      Print less protocol information.  
r file.pcap Read from a file.
```

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

Answer: -v

```
nn     Do not resolve hostnames or well-known ports.  
e      Will grab the ethernet header along with upper-layer data.  
x      Show Contents of packets in hex and ASCII.  
xx    Same as X, but will also specify ethernet headers. (like using Xe)  
v, vv, vvv Increase the verbosity of output shown and saved.  
c      Grab a specific number of packets, then quit the program.  
s      Defines how much of a packet to grab.  
S      change relative sequence numbers in the capture display to absolute sequence numbers.  
(13248765839 instead of 101)  
q      Print less protocol information.  
r file.pcap Read from a file.  
w file.pcap Write into a file  
host   Host will filter visible traffic to show anything involving the designated host. Bi-directional
```

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

Answer: man

## Week 2 Assignment 2

The terminal window displays the following content:

**Man Page Utilization**

To see the complete list of switches, we can utilize the man pages:

**Tcpdump Man Page**

```
[!bash!]$ man tcpdump
```

Here are some examples of basic Tcpdump switch usage along with descriptions of what is happening:

**Listing Available Interfaces**

```
[!bash!]$ sudo tcpdump -D
```

Output listing available interfaces:

```
1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth0 (Bluetooth adapter number 0) [Wireless, Association status unknown]
5.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
6.nflog (Linux netfilter log (NFLOG) interface) [none]
7.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
8.dbus-system (D-Bus system bus) [none]
9.rhui-session (R-Rus session bus) [none]
```

The terminal window has a red arrow pointing to the "man" command in the first line of output.

**What TCPDump switch will let me write my output to a file?**

Answer: -w

The terminal window displays the Tcpdump man page with the "-w" switch highlighted by a red arrow.

**Questions**

Answer the question

+ 0 Utilizing the host 174.143.213.184 + 0 Were absconed

v, vv, vvv Increase the verbosity of output shown and saved.

c Grab a specific number of packets, then quit the program.

s Defines how much of a packet to grab.

S change relative sequence numbers in the capture display to absolute sequence numbers. (13248765939 instead of 101)

q Print less protocol information.

r file.pcap Read from a file.

w file.pcap Write into a file

host Host will filter visible traffic to show anything involving the designated host. Bi-directional

src / dest src and dest are modifiers. We can use them to designate a source or destination host or port.

net net will show us any traffic sourcing from or destined to the network designated. It uses / notation.

proto will filter for a specific protocol type. (ether, TCP, UDP, and ICMP as examples)

port port is bi-directional. It will show any traffic with the specified port as the source or destination.

Fundamentals Lab

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

Answer: -l

## Week 2 Assignment 2

The screenshot shows a Windows desktop environment. On the left, there's a 'Questions' sidebar with sections for 'What TCP!' and 'True or False'. The main area displays various TCPdump command examples with their descriptions. A red arrow points to the command `tcpdump -i (int) -w file.pcap -l \| grep 'string'`. At the bottom, a title 'Tcpdump Common Switches and Filters' is centered above a toolbar with various icons.

Command	Description
<code>tcpdump -i (int) -w file.pcap</code>	Runs a capture on the specified interface and writes the output to a file.
<code>tcpdump -r file.pcap</code>	TCPDump will read the output from a specified file.
<code>tcpdump -r/-w file.pcap -l \  grep 'string'</code>	TCPDump will utilize the capture traffic from a live capture or a file and set stdio as line-buffered. We can then utilize pipe ( ) to send that output to other tools such as grep to look for strings or specific patterns.
<code>tcpdump -i (int) host (ip)</code>	TCPDump will start a capture on the interface specified at (int) and will only capture traffic originating from or destined to the IP address or hostname specified after host.
<code>tcpdump -i (int) port (#)</code>	Will filter the capture for anything sourcing from or destined to port (#) and discard the rest.
<code>tcpdump -i (int) proto (#)</code>	Will filter the capture for any protocol traffic matching the (#). For example, (6) would filter for any TCP traffic and discard the rest.
<code>tcpdump -i (int) (proto name)</code>	Will utilize a protocols common name to filter the traffic captured. TCP/UDP/ICMP as examples.

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

Answer: True

This screenshot is similar to the one above, showing the same 'Questions' sidebar and TCPdump command examples. A red arrow points to the 'port' filter example, which is described as bi-directional and capable of filtering traffic by source or destination port. The interface and system tray are identical to the first screenshot.

Filter	Description
<code>w file.pcap</code>	Write into a file
<code>host</code>	Host will filter visible traffic to show anything involving the designated host. Bi-directional
<code>src / dest</code>	<code>src</code> and <code>dest</code> are modifiers. We can use them to designate a source or destination host or port.
<code>net</code>	<code>net</code> will show us any traffic sourcing from or destined to the network designated. It uses / notation.
<code>proto</code>	will filter for a specific protocol type. (ether, TCP, UDP, and ICMP as examples)
<code>port</code>	<code>port</code> is bi-directional. It will show any traffic with the specified port as the source or destination.
<code>portrange</code>	<code>Portrange</code> allows us to specify a range of ports. (0-1024)
<code>less / greater "</code> <code>&lt; &gt;</code>	<code>less</code> and <code>greater</code> can be used to look for a packet or protocol option of a specific size.
<code>and / &amp;&amp;</code>	<code>and</code> <code>&amp;&amp;</code> can be used to concatenate two different filters together. for example, src host AND port.
<code>or</code>	<code>or</code> <code>or</code> allows for a match on either of two conditions. It does not have to meet both. It can be t
<code>not</code>	<code>not</code> is a modifier saying anything but x. For example, not UDP.

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

Answer: not ICMP

## Week 2 Assignment 2

The screenshot shows a terminal window with a list of TCPDUMP filters and a TShark help menu.

**TCPDUMP Filters:**

- net
- proto
- port
- portrange
- less / greater "
- and / &&
- or
- not

A red arrow points to the "not" filter entry.

**TShark Help:**

Command	Description
tshark -h	Prints the help menu.

**What command will show you where / if TCPDump is installed?**

Answer: which tcpdump

The screenshot shows a terminal window displaying the output of the "which" command, which shows the path to the tcpdump executable.

```
busslus@htb[/htb]$ which tcpdump
```

A red arrow points to the terminal output.

**Tasks:**

**Task #1**

Validate Tcpdump is installed on our machine.

Before we can get started, ensure we have tcpdump installed. What command do we use to determine if tcpdump is installed on Linux?

▼ Click to show answer

To determine if we have tcpdump installed, we can utilize the command in Linux or hit the Windows key and start typing tcpdump on Windows.

**Task #2**

Start a capture.

Once we know tcpdump is installed, we are ready to start our first capture. If we are unsure of what interface we

**Wireshark:**

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

**How do you start a capture with TCPDump to capture on eth0?**

Answer: tcpdump -i eth0

## Week 2 Assignment 2

```
Target IP == 10.129.43.4
Username == htb-student
Password == HTB._@cademy_stdnt!
```

### Tcpdump

Command	Description
<code>tcpdump --version</code>	Prints the tcpdump and libpcap version strings then exits.
<code>tcpdump -h</code>	Prints the help and usage information.
<code>tcpdump -D</code>	Prints a list of usable network interfaces from which tcpdump can capture.
<code>tcpdump -i &lt;interface name or #&gt;</code>	Executes tcpdump and utilizes the interface specified to capture on.
<code>tcpdump -i &lt;int&gt; -w &lt;file.pcap</code>	Runs a capture on the specified interface and writes the output to a file.
<code>tcpdump -r &lt;file.pcap</code>	TCPDump will read the output from a specified file.
<code>tcpdump -r/-w &lt;file.pcap&gt; -l \&gt;  grep 'string'</code>	TCPDump will utilize the capture traffic from a live capture or a file and set stdout as line-buffered. We can then utilize pipe ( ) to send that output to other tools such as grep to look for strings or specific patterns.

What switch will provide more verbosity in your output?

Answer: -v

```
n          Do not resolve hostnames.
nn         Do not resolve hostnames or well-known ports.
e          Will grab the ethernet header along with upper-layer data.
x          Show Contents of packets in hex and ASCII.
xx         Same as X, but will also specify ethernet headers. (like using Xe)
v, vvv     Increase the verbosity of output shown and saved.
c          Grab a specific number of packets, then quit the program.
s          Defines how much of a packet to grab.
          change relative sequence numbers in the capture display to absolute sequence numbers.
          (13248765839 instead of 101)
q          Print less protocol information.
r <file.pcap>      Read from a file.
w <file.pcap>      Write into a file
```

What switch will write your capture output to a .pcap file?

Answer: -w

## Week 2 Assignment 2

The screenshot shows the Wireshark interface with a list of command-line options. A red arrow points to the 'w file.pcap' option, which is described as 'Write into a file'.

Option	Description
v, vv, vvv	Increase the verbosity of output shown and saved.
c	Grab a specific number of packets, then quit the program.
s	Defines how much of a packet to grab.
S	change relative sequence numbers in the capture display to absolute sequence numbers. (13248765839 instead of 101)
q	Print less protocol information.
r file.pcap	Read from a file.
w file.pcap	Write into a file
host	Host will filter visible traffic to show anything involving the designated host. Bi-directional
src / dest	src and dest are modifiers. We can use them to designate a source or destination host or port.
net	net will show us any traffic sourcing from or destined to the network designated. It uses / notation.
proto	will filter for a specific protocol type. (ether, TCP, UDP, and ICMP as examples)
port	port is bi-directional. It will show any traffic with the specified port as the source or destination.

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

Answer: -r

The screenshot shows the Wireshark interface with a list of command-line options. A red arrow points to the 'r file.pcap' option, which is described as 'Read from a file'.

Option	Description
c	Grab a specific number of packets, then quit the program.
s	Defines how much of a packet to grab.
S	change relative sequence numbers in the capture display to absolute sequence numbers. (13248765839 instead of 101)
q	Print less protocol information.
r file.pcap	Read from a file.
w file.pcap	Write into a file
host	Host will filter visible traffic to show anything involving the designated host. Bi-directional
src / dest	src and dest are modifiers. We can use them to designate a source or destination host or port.
net	net will show us any traffic sourcing from or destined to the network designated. It uses / notation.
proto	will filter for a specific protocol type. (ether, TCP, UDP, and ICMP as examples)
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 a range of ports. (0-1024)

**What switch will show the contents of a capture in Hex and ASCII?**

Answer: -X

## Week 2 Assignment 2

Tcpdump Common Switches and Filters

Switch/Filter	Description
<code>d</code>	Will display any interfaces available to capture from.
<code>i</code>	Selects an interface to capture from. ex. -i eth0
<code>n</code>	Do not resolve hostnames.
<code>nn</code>	Do not resolve hostnames or well-known ports.
<code>e</code>	Will grab the ethernet header along with upper-layer data.
<code>x</code>	Show Contents of packets in hex and ASCII. <span style="color:red">→</span>
<code>xx</code>	Same as X, but will also specify ethernet headers. (like using Xe)
<code>v, vv, vvv</code>	Increase the verbosity of output shown and saved.
<code>c</code>	Grab a specific number of packets, then quit the program.
<code>s</code>	Defines how much of a packet to grab.
<code>S</code>	change relative sequence numbers in the capture display to absolute sequence numbers.

## Tcpdump Packet Filtering

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

Answer: host 10.10.20.1

and / &&      `and &&` can be used to concatenate two different filters together. for example, src host AND port.

or      `or` allows for a match on either of two conditions. It does not have to meet both. It can be tricky.

not      `not` is a modifier saying anything but x. For example, not UDP.

With these filters, we can filter the network traffic on most properties to facilitate the analysis. Let us look at some examples of these filters and how they look when we use them. When using the `host` filter, whatever IP we input will be checked for in the source or destination IP field. This can be seen in the output below.

**Host Filter** →

Tcpdump Packet Filtering

```
busslus@htb[~/htb]$ ### Syntax: host [IP]
busslus@htb[~/htb]$ sudo tcpdump -i eth0 host 172.16.146.2

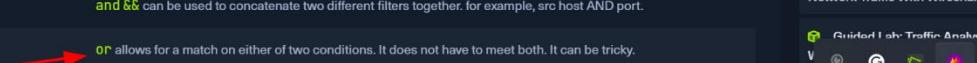
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:50:53.072536 IP 172.16.146.2.48738 > ec2-52-31-199-148.eu-west-1.compute.amazonaws.com.https: Flags [P.], ack 92, win 69, length 0
14:50:53.108740 IP 172.16.146.2.55606 > 172.67.1.1.https: Flags [P.], seq 4227143181:4227143273, ack 92, win 69, length 0
14:50:53.173084 IP 172.67.1.1.https > 172.16.146.2.55606: Flags [.], ack 92, win 69, length 0
14:50:53.175017 IP 172.16.146.2.35744 > 172.16.146.1.domain: 55991+ PTR? 148.199.31.52.in-addr.arpa. 55991 1/0/0 PTR ec2-52-31-199-148.eu
14:50:53.175714 IP 172.16.146.1.domain > 172.16.146.2.35744: 55991 1/0/0 PTR ec2-52-31-199-148.eu
```

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

Answer: or

## Week 2 Assignment 2

## Helpful TCPDump Filters

Filter	Result
host	<b>host</b> will filter visible traffic to show anything involving the designated host. Bi-directional
src / dest	<b>src</b> and <b>dest</b> are modifiers. We can use them to designate a source or destination host or port.
net	<b>net</b> will show us any traffic sourcing from or destined to the network designated. It uses / notation.
proto	will filter for a specific protocol type. (ether, TCP, UDP, and ICMP as examples)
port	<b>port</b> is bi-directional. It will show any traffic with the specified port as the source or destination.
portrange	<b>portrange</b> allows us to specify a range of ports. (0-1024)
less / greater "< >"	<b>less</b> and <b>greater</b> can be used to look for a packet or protocol option of a specific size.
and / &&	<b>and &amp;&amp;</b> can be used to concatenate two different filters together. for example, src host AND port.
or	<b>or</b> allows for a match on either of two conditions. It does not have to meet both. It can be tricky. 
not	<b>not</b> is a modifier saying anything but x. For example, not UDP.

With these filters, we can filter the network traffic on most properties to facilitate the analysis. Let us look at some

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

### Wireshark

- Analysis with Wireshark
- Familiarity With Wireshark
- Wireshark Advanced Usage
- Packet Inception, Dissecting Network Traffic With Wireshark

- Network Minicourse
- Wireshark
- Sniffing with Wireshark
- Protocol Analysis with Wireshark
- Wireshark Advanced Usage
- Wireshark Plugins
- Wireshark Scripts
- Wireshark Configuration
- Wireshark Troubleshooting

- Wireshark Icons
- Wireshark Plugins
- Wireshark Scripts
- Wireshark Configuration
- Wireshark Troubleshooting



1:51 PM  
1/27/2026

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

Answer: true

```
busslus@htb:[/htb]$ ### Syntax: not! [requirement]
busslus@htb:[/htb]$ sudo tcpdump -r sus.pcap not icmp

14:54:03.879882 ARP, Request who-has 172.16.146.1 tell 172.16.146.2, length 28
14:54:03.880266 ARP, Reply 172.16.146.1 is-at 8a:66:5a:11:8d:64 (oui Unknown), length 46
14:54:16.541657 IP 172.16.146.2.55592 > ec2-52-211-164-46.eu-west-1.compute.amazonaws.com.https: 
14:54:16.568659 IP 172.16.146.2.53329 > 172.16.146.1.domain: 24866+ A? app.hackthebox.eu. (35)
14:54:16.616032 IP 172.16.146.1.domain > 172.16.146.2.53329: 24866 3/0/0 A 172.67.1.1, A 104.20.6
14:54:16.616396 IP 172.16.146.2.56204 > 172.67.1.1.https: Flags [S], seq 2697802378, win 64240, o
14:54:16.637895 IP 172.67.1.1.https > 172.16.146.2.56204: Flags [S.], seq 752000032, ack 26978023
14:54:16.637937 IP 172.16.146.2.56204 > 172.67.1.1.https: Flags [.], ack 1, win 502, length 0
14:54:16.644551 IP 172.16.146.2.56204 > 172.67.1.1.https: Flags [P.], seq 1:514, ack 1, win 502,
14:54:16.667236 IP 172.67.1.1.https > 172.16.146.2.56204: Flags [.], ack 514, win 66, length 0
14:54:16.668307 IP 172.67.1.1.https > 172.16.146.2.56204: Flags [P.], seq 1:2766, ack 514, win 66
14:54:16.668319 IP 172.16.146.2.56204 > 172.67.1.1.https: Flags [.], ack 2766, win 496, length 0
14:54:16.670536 IP ec2-52-211-164-46.eu-west-1.compute.amazonaws.com.https > 172.16.146.2.55592:
14:54:16.670559 IP 172.16.146.2.55592 > ec2-52-211-164-46.eu-west-1.compute.amazonaws.com.https:
```

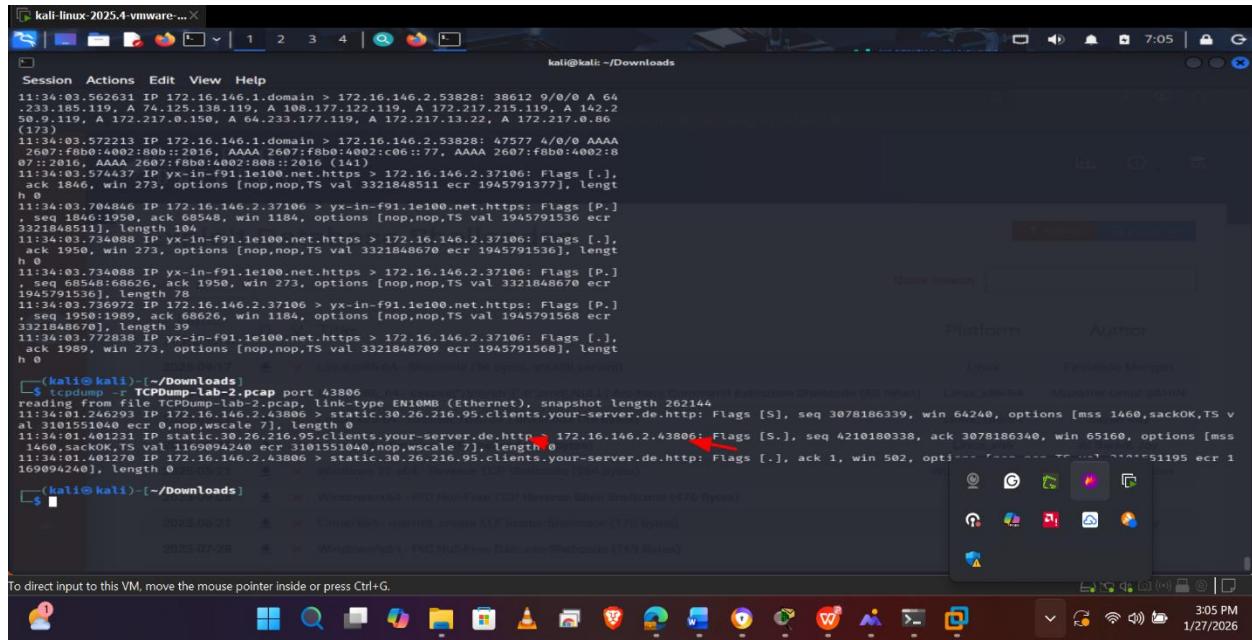
It looks much different now. We only see some ARP traffic, and then we see some HTTPS traffic we did not get to before. This is because we negated any ICMP traffic from being displayed using `not icmp`.

## Interrogating Network Traffic With Capture and Display Filters

**What are the client and server port numbers used in first full TCP three-way handshake? (low number first then high number)**

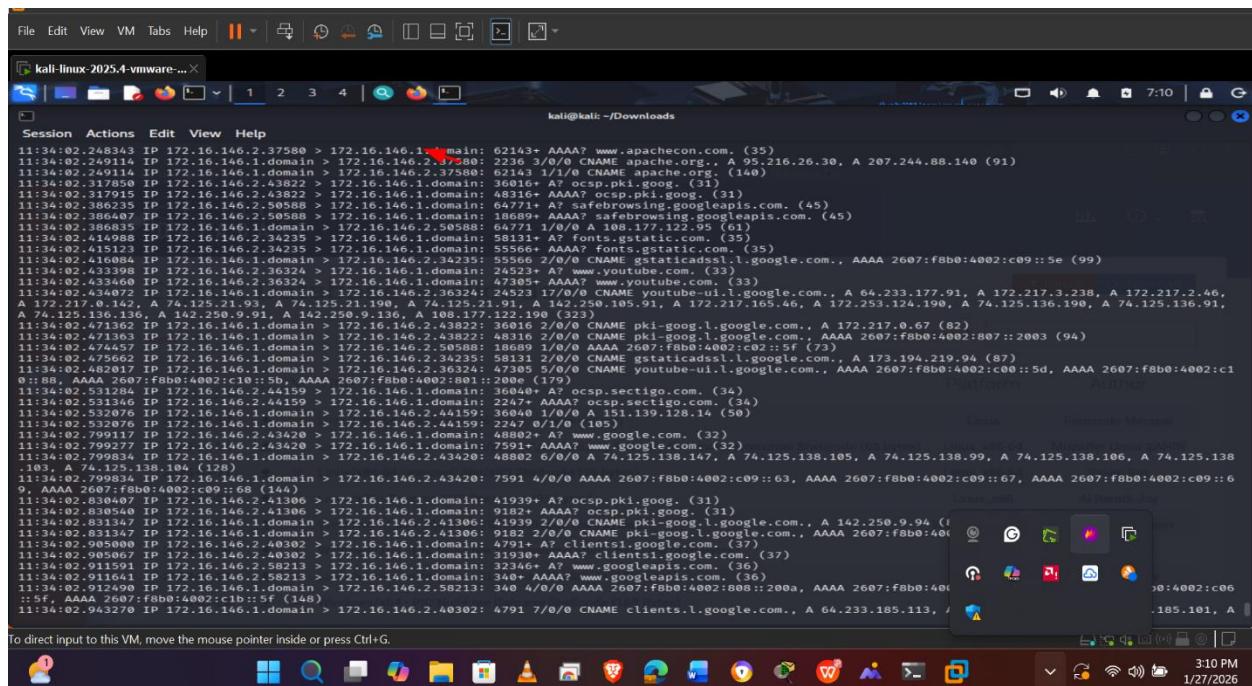
Answer: 80 43806

## Week 2 Assignment 2



Based on the traffic seen in the pcap file, who is the DNS server in this network segment? (ip address)

Answer: 172.16.146.1

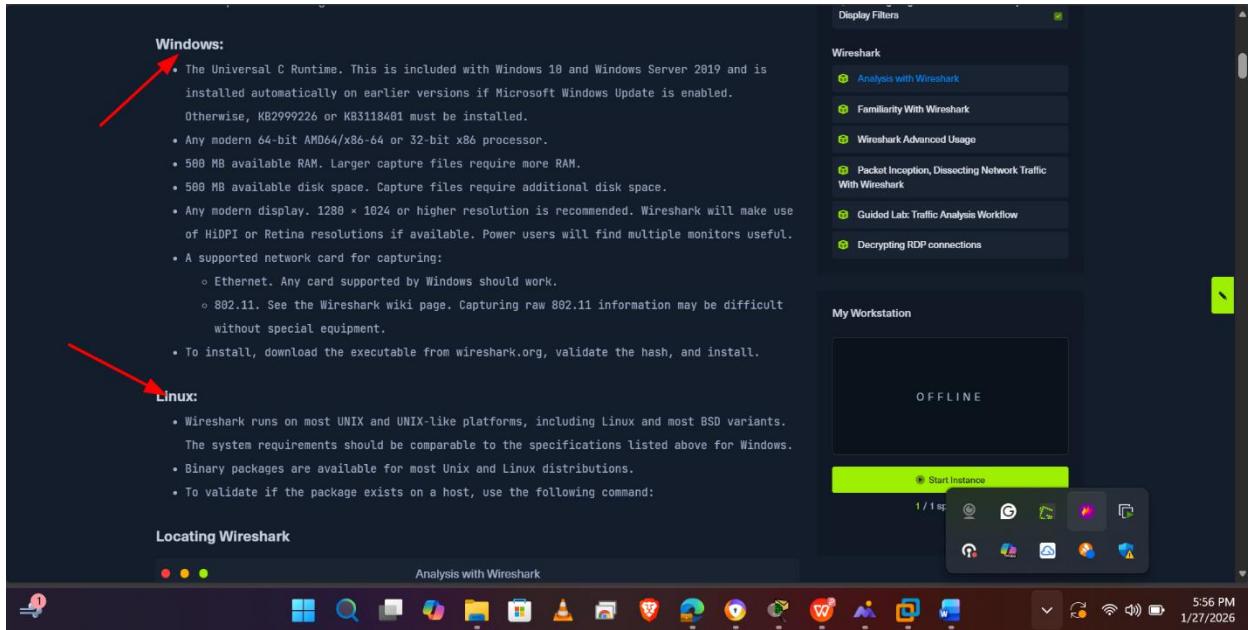


Analysis with Wireshark

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

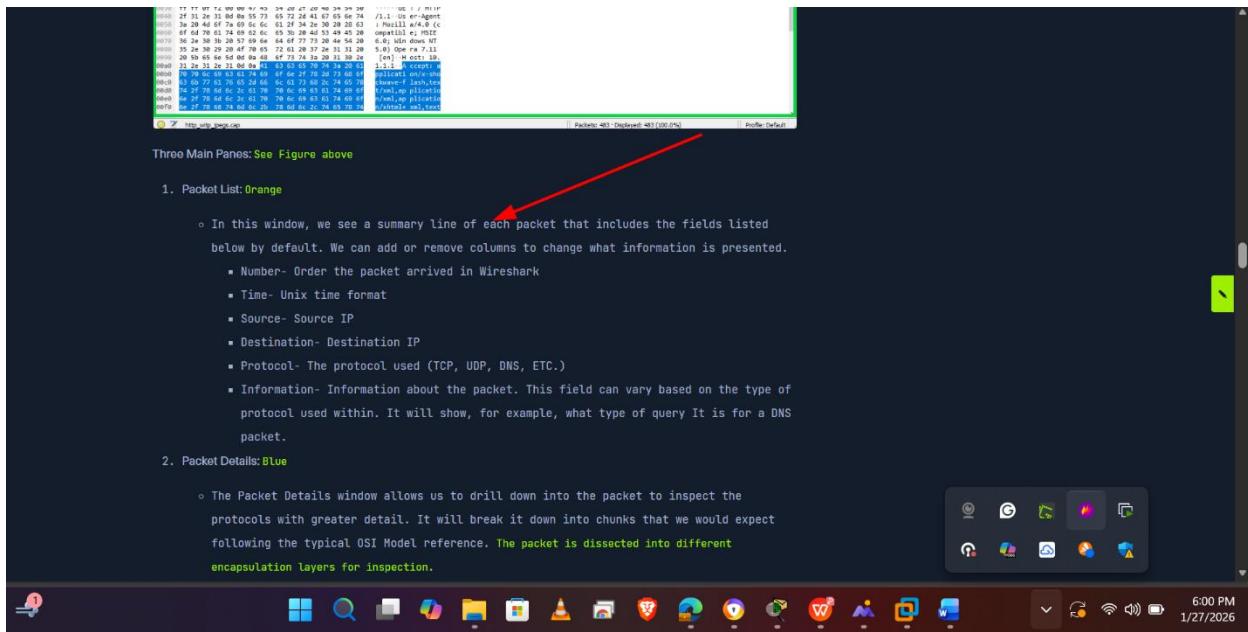
Answer: True

## Week 2 Assignment 2



Which Pane allows a user to see a summary of each packet grabbed during the capture?

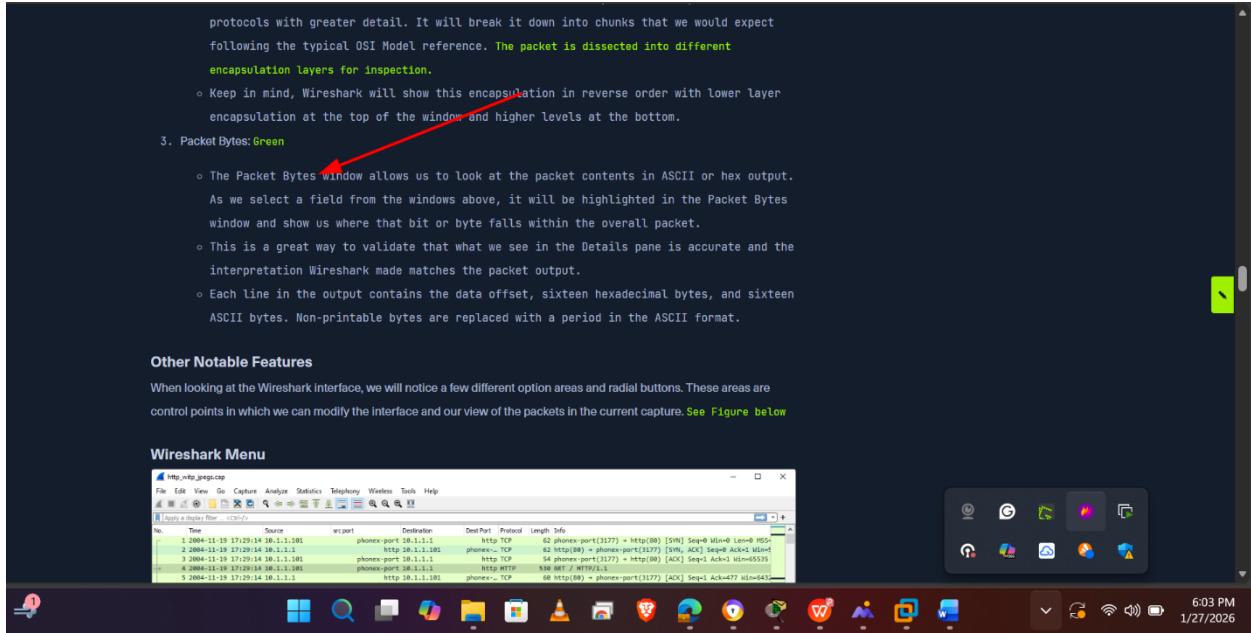
Answer: Packet List



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

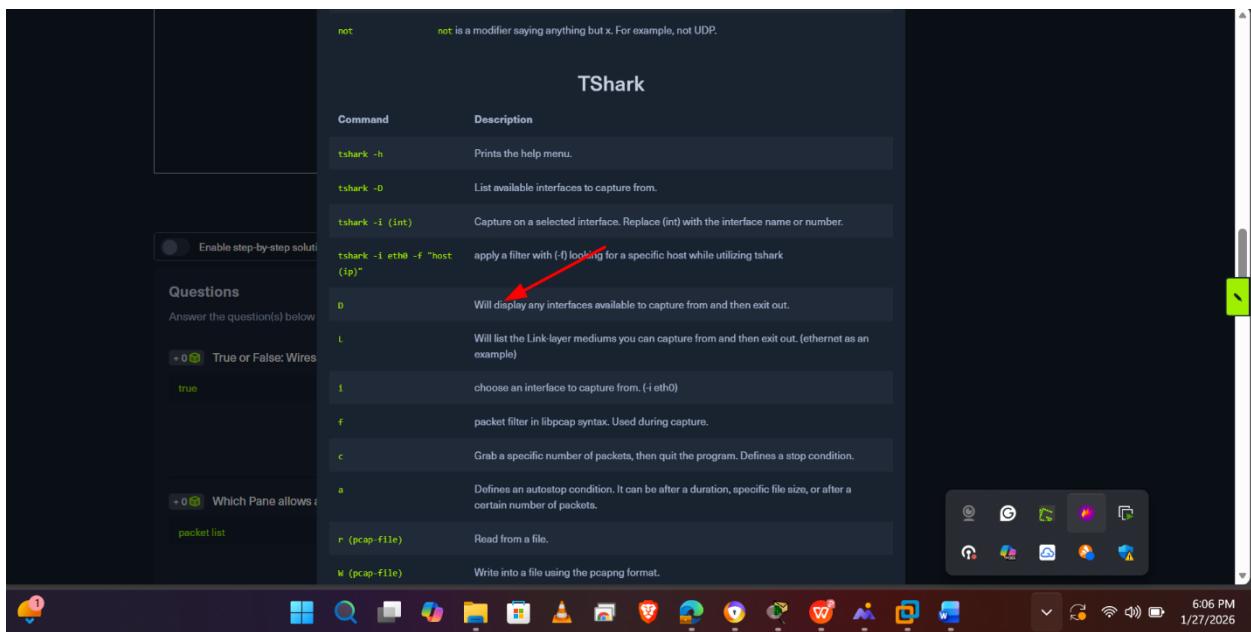
Answer: Packets byte

## Week 2 Assignment 2



What switch is used with TShark to list possible interfaces to capture on?

Answer: -D



What switch allows us to apply filters in TShark?

Answer: -f

## Week 2 Assignment 2

The screenshot shows the TShark application window. On the left, there's a sidebar with a 'Questions' section asking about capture filters. The main area displays the TShark help menu, which includes various command options like '-i', '-f', and '-c'. A red arrow points to the '-f' option, which is described as a 'packet filter in libpcap syntax. Used during capture.'

Command	Description
tshark -h	Prints the help menu.
tshark -o	List available interfaces to capture from.
tshark -i (int)	Capture on a selected interface. Replace (int) with the interface name or number.
tshark -l eth0 -f "host (ip)"	apply a filter with (-f) looking for a specific host while utilizing tshark
D	Will display any interfaces available to capture from and then exit out.
L	Will list the Link-layer mediums you can capture from and then exit out. (ethernet as an example)
i	choose an interface to capture from. (-i eth0)
f	packet filter in libpcap syntax. Used during capture. <span style="color:red">→</span>
c	Grab a specific number of packets, then quit the program. Defines a stop condition.
a	Defines an autostop condition. It can be after a duration, specific file size, or after a certain number of packets.
r (pcap-file)	Read from a file.
w (pcap-file)	Write into a file using the pcapng format.
p	Will print the packet summary while writing into a file (-W)

Is a capture filter applied before the capture starts or after? (answer before or after)

Answer: Before

The screenshot shows a terminal window with text explaining capture filters. A red arrow points to the first sentence: 'Capture Filters - are entered before the capture is started. These use BPF syntax like host 214.15.2.38 much in the same fashion as TCPDump. We have fewer filter options this way, and a capture filter will drop all other traffic not explicitly meeting the criteria set. This is a great way to trim down the data you write to disk when troubleshooting a connection, such as capturing the conversations between two hosts.'

Here is a table of common and helpful capture filters with a description of each:

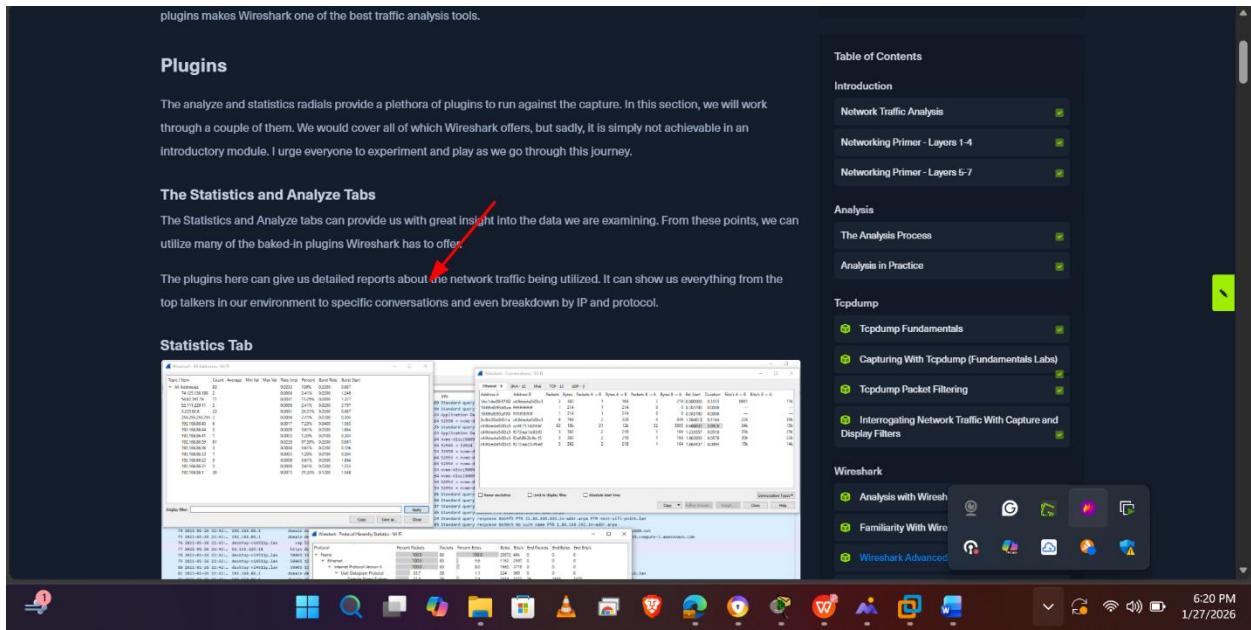
Capture Filters	Result
host x.x.x.x	Capture only traffic pertaining to a certain host
net x.x.x.x/24	Capture traffic to or from a specific network (using slash notation to specify the mask)
src/dst net x.x.x.x/24	Using src or dst net will only capture traffic sourcing from the specified network or destined to the target network
port #	will filter out all traffic except the port you specify
not port #	will capture everything except the port specified
port # and #	AND will concatenate your specified ports
portrange x-x	portrange will grab traffic from all ports within the range only

## Wireshark Advanced Usage

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

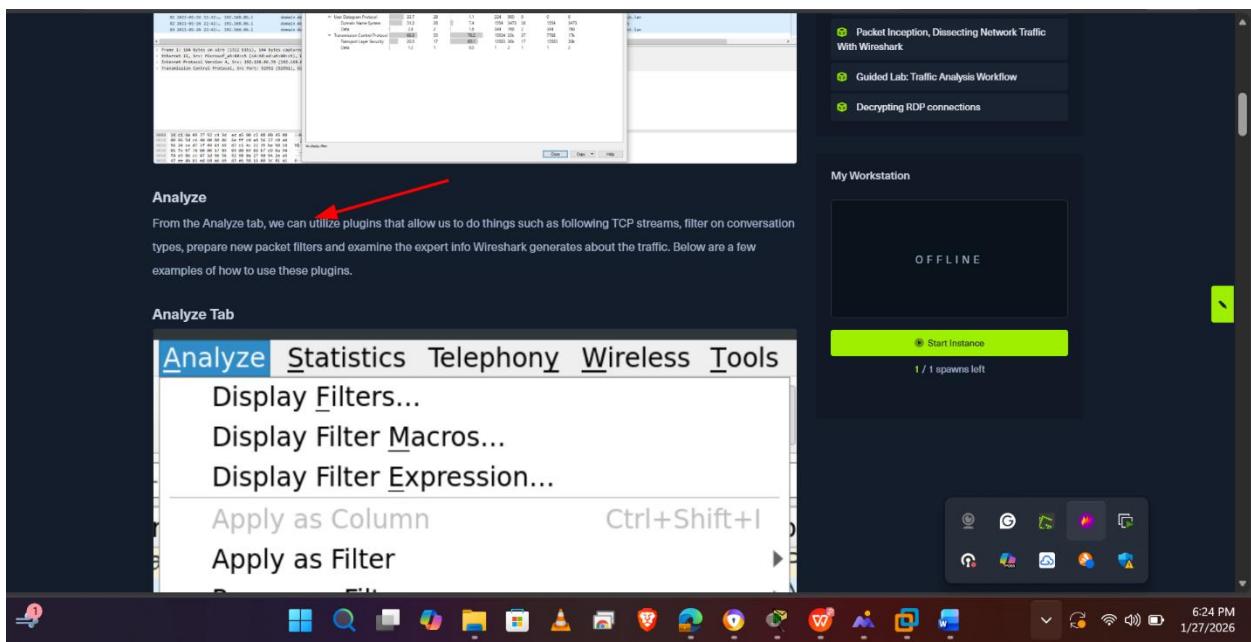
Answer: statistics

## Week 2 Assignment 2



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

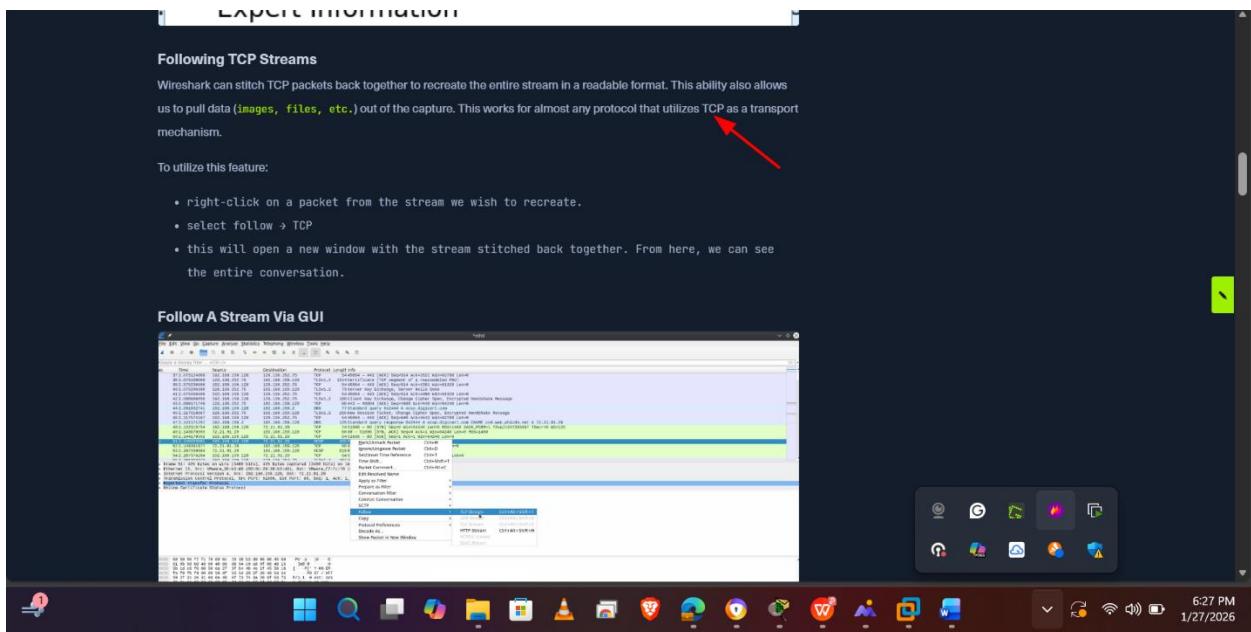
Answer: Analyze



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

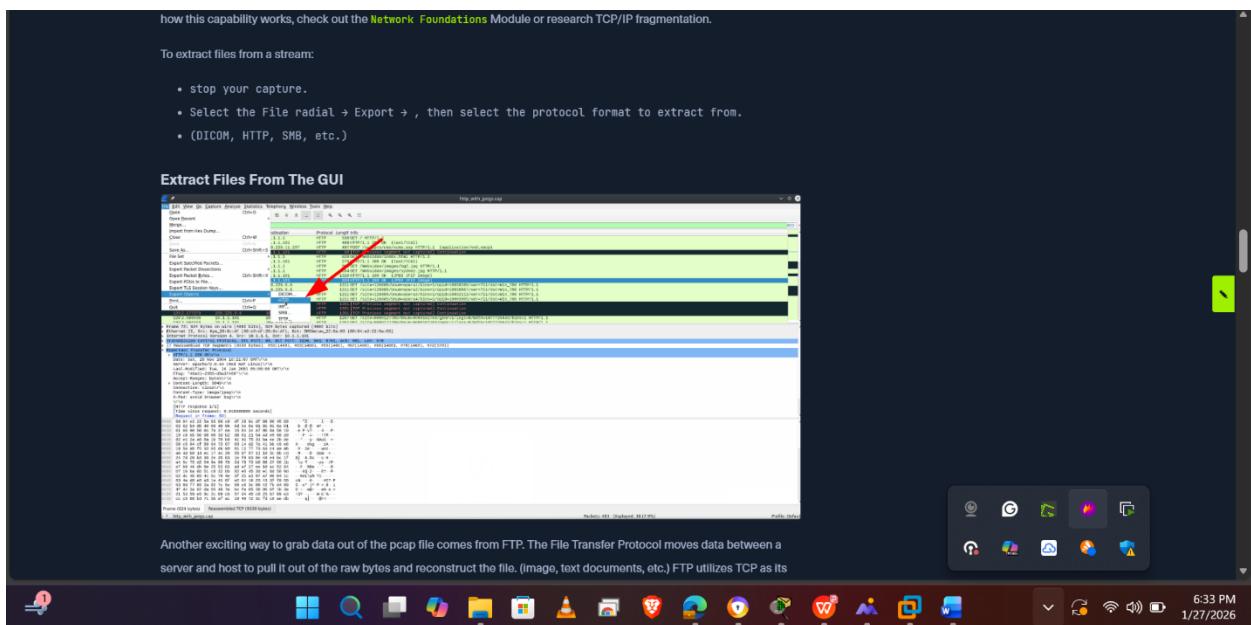
Answer: TCP

## Week 2 Assignment 2



**True or False: Wireshark can extract files from HTTP traffic.**

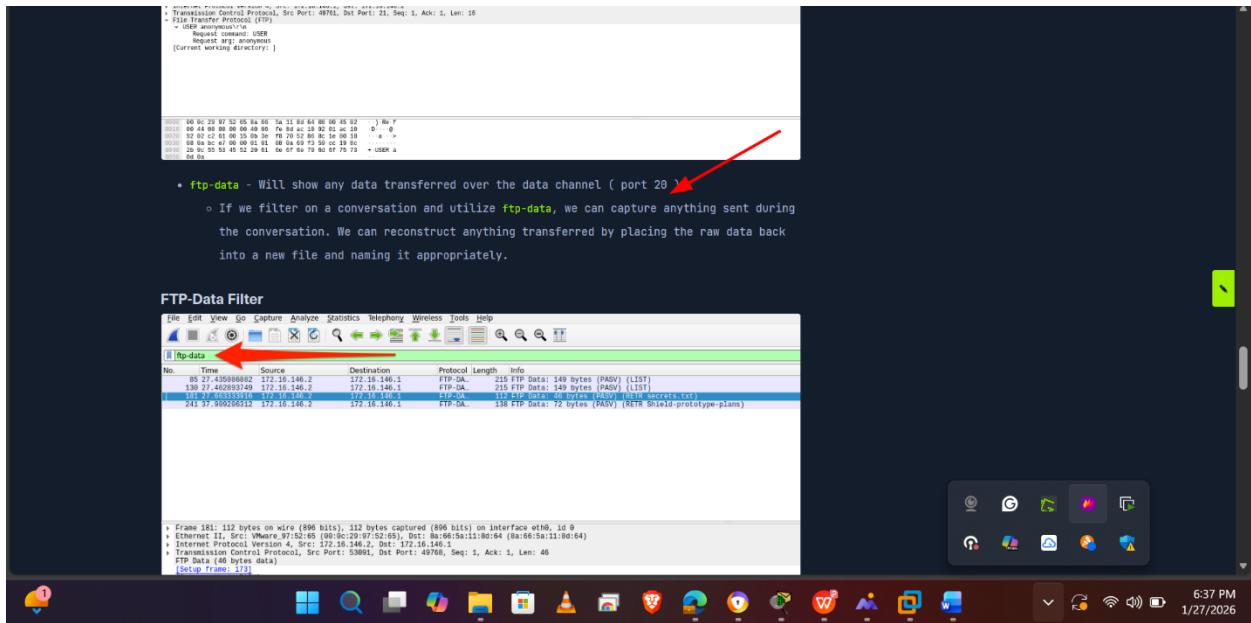
Answer: True



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

Answer: False

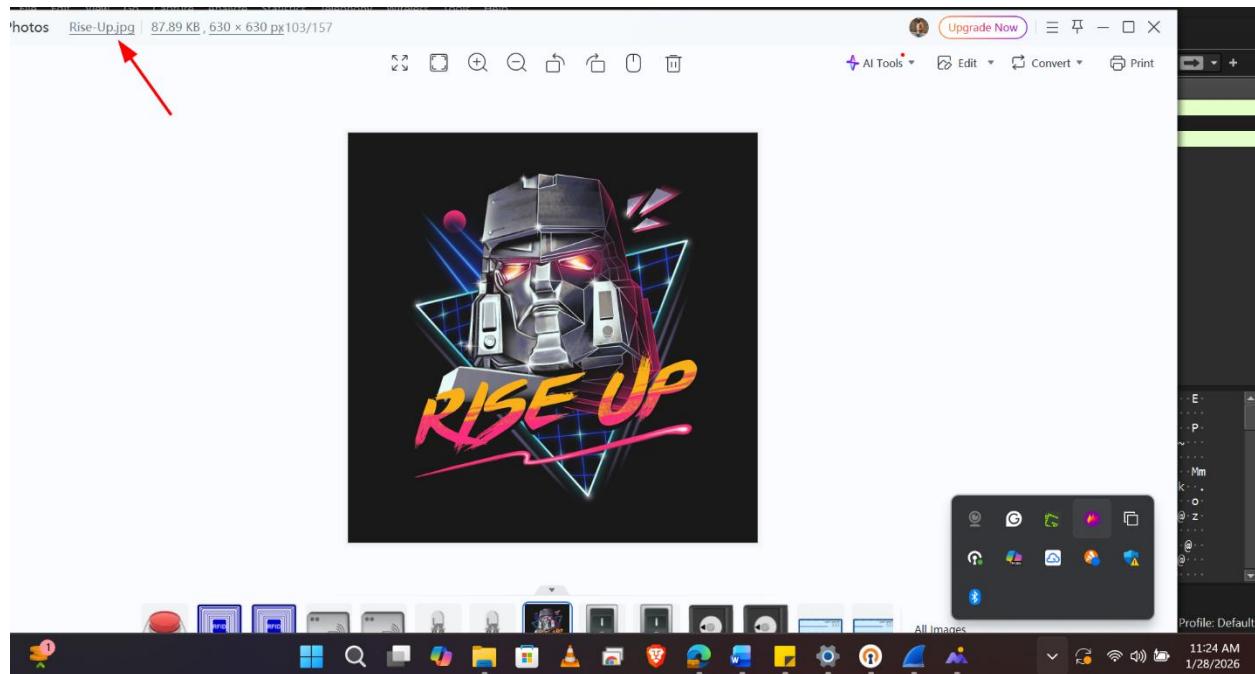
## Week 2 Assignment 2



### Packet Inception, Dissecting Network Traffic With Wireshark

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

Answer: Rise-up.jpg



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

## Week 2 Assignment 2

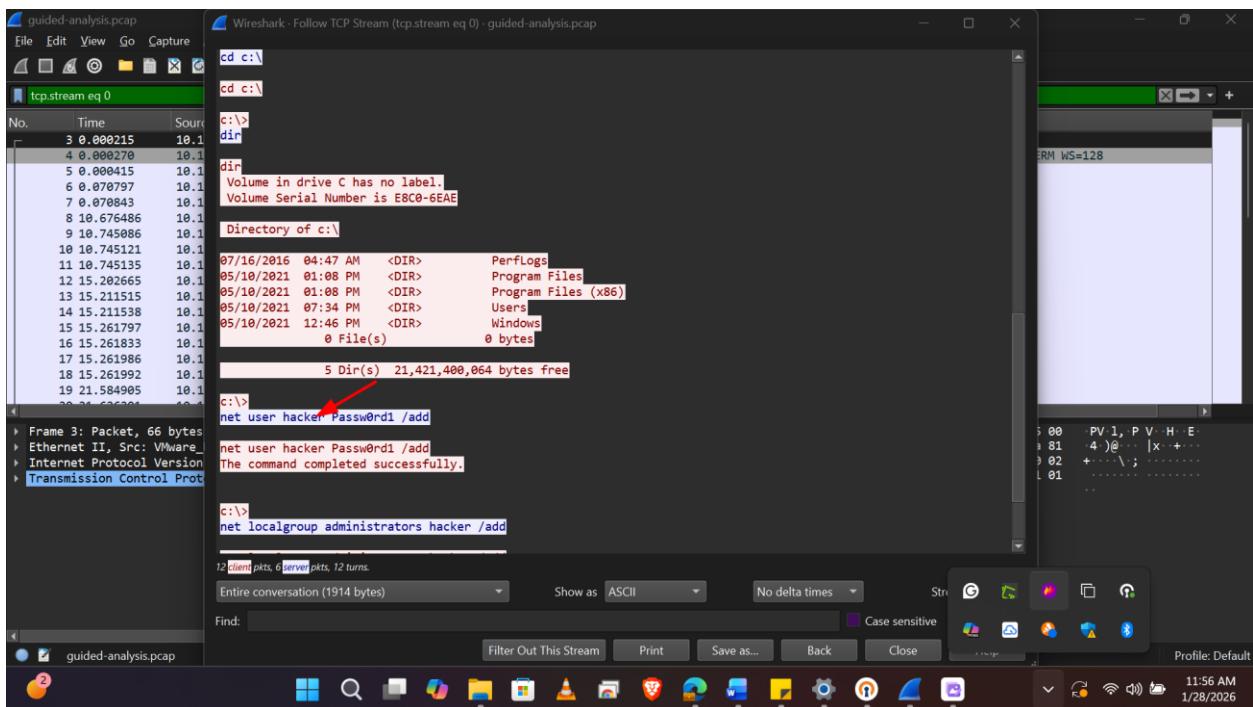
Answer: Bob



### Guided Lab: Traffic Analysis Workflow

What was the name of the new user created on mrb3n's host?

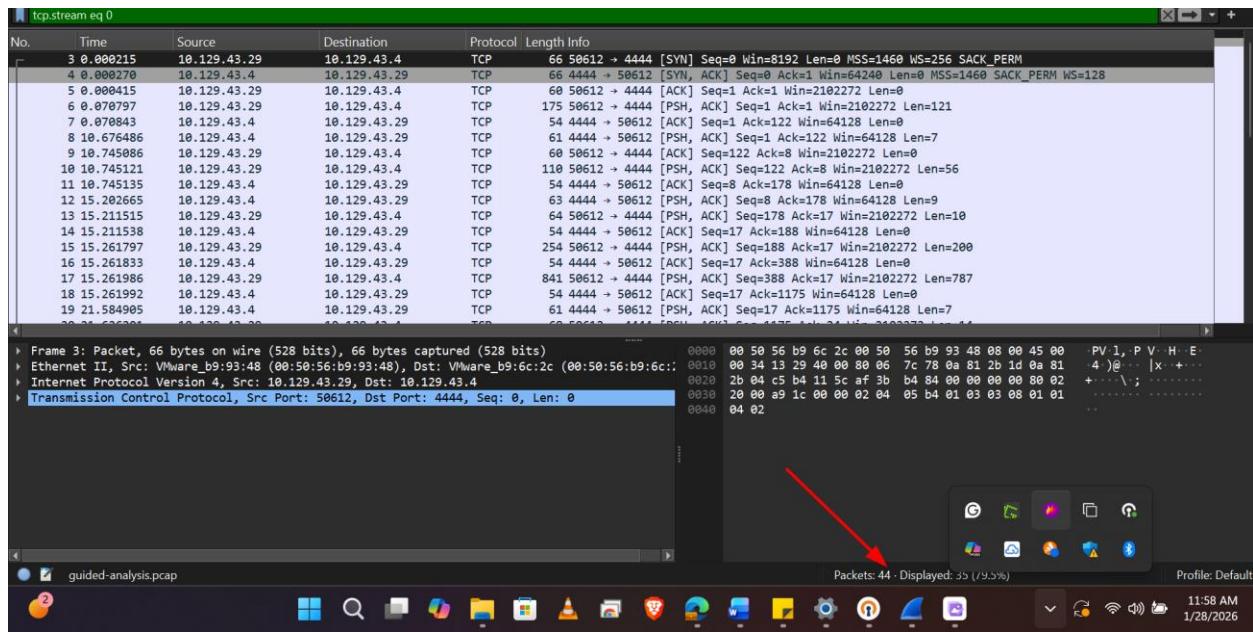
Answer: Hacker



How many total packets were there in the Guided-analysis PCAP?

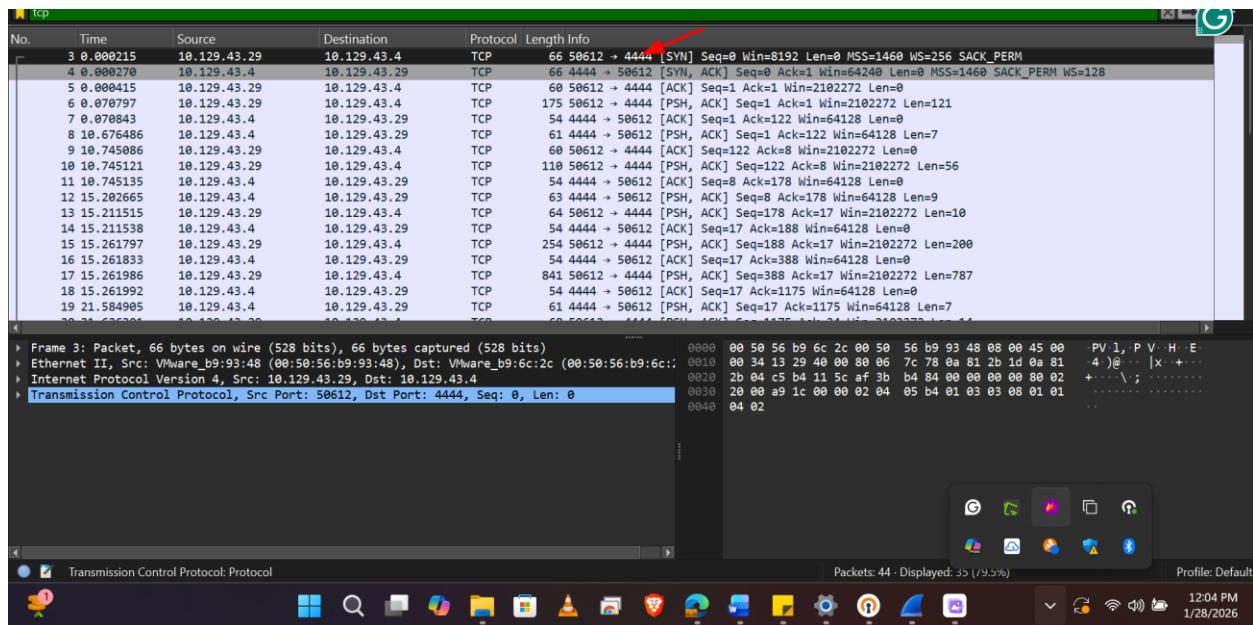
Answer: 44

## Week 2 Assignment 2



What was the suspicious port that was being used?

Answer: 4444

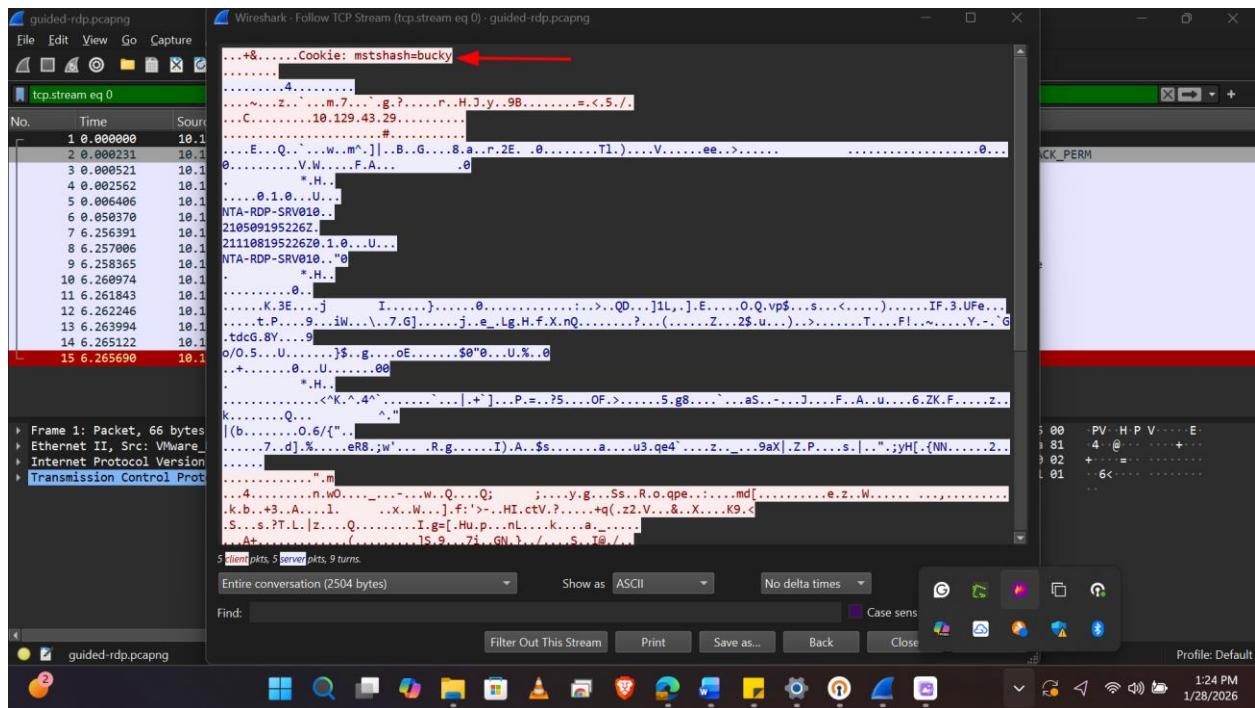


Decrypting RDP connections

What user account was used to initiate the RDP connection?

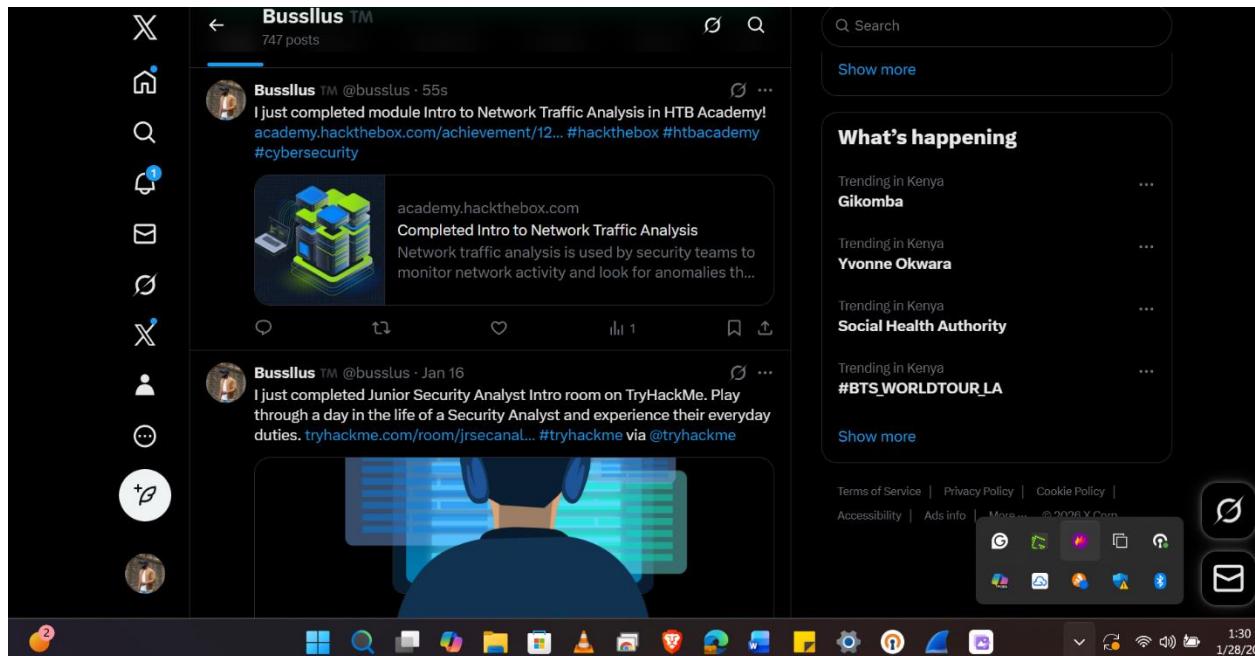
Answer: bucky

## Week 2 Assignment 2



## Verification of Completion

This completion has been shared on professional networks like Twitter to publicly document my commitment to continuous learning in cybersecurity. This section provides the official verification artifacts for the "Intro to Network Traffic Analysis" module.



## Week 2 Assignment 2

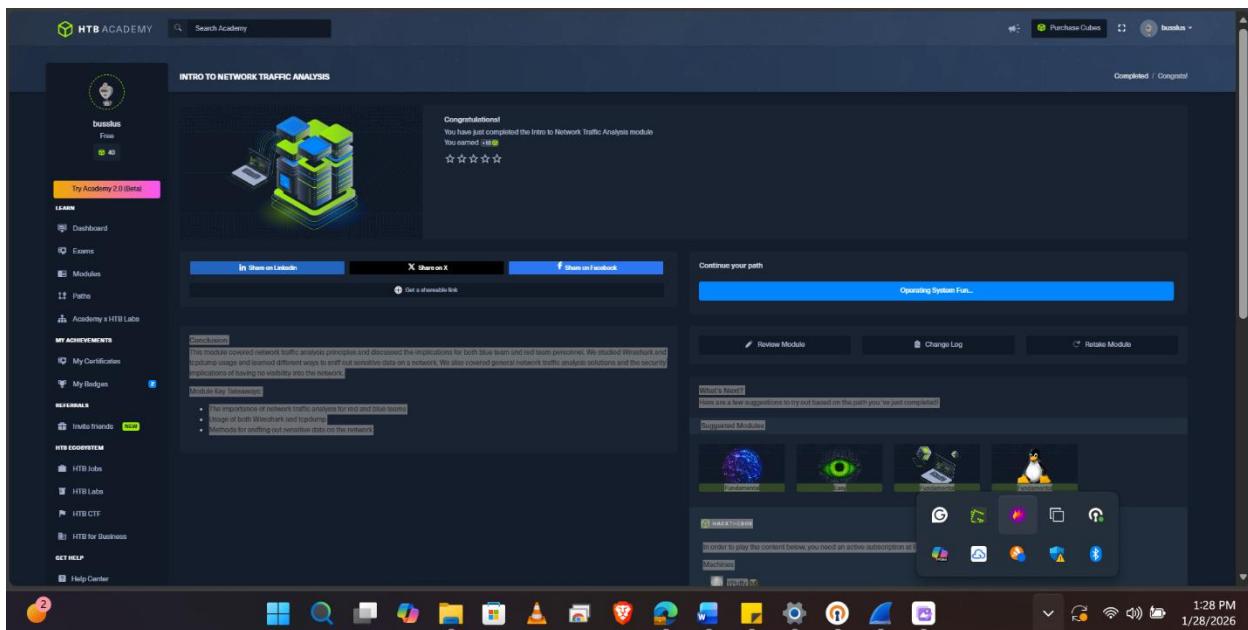
### Shareable Link

The following unique link, generated by Hack The Box Academy, provides official validation of the module's completion

<https://academy.hackthebox.com/achievement/badge/e6213437-fc33-11f0-9254-bea50ffe6cb4>

### Completion Screenshot

The screenshot below serves as the official confirmation that all sections and challenges within the module have been completed successfully.



### Conclusion & Reflection

This final section serves as a strategic reflection on the learning experience provided by the "Intro to Network Traffic Analysis" module. It synthesizes the key takeaways and evaluates the practical value of the skills acquired, framing them within the context of real-world cybersecurity operations.

The module provided a comprehensive and practical education in network traffic analysis. The most critical skills and concepts gained include:

- **The OSI Model in Practice:** The module provided a tangible link between the OSI model and packet analysis. For instance, identifying the attacker's IP in question 2.1 was a Layer 3 (Network) task focused on IP headers, while extracting credentials in 2.3 was a Layer 7

## Week 2 Assignment 2

(Application) task that required reassembling a TCP stream to read HTTP data. This practical application reinforces the model as a critical diagnostic framework.

- **Packet Header Analysis:** A significant portion of the learning involved dissecting the headers of various protocols (IP, TCP, UDP, HTTP). This skill is invaluable for understanding the context of a connection, identifying spoofed or malformed packets, and tracing the path of data across a network.
- **Effective Filtering in Wireshark:** Mastery of Wireshark's display filters is a force multiplier. The module provided hands-on practice moving from simple protocol filters (e.g., http) to complex, compound expressions (`ip.addr == X.X.X.X && tcp.port == 80`), which is essential for quickly isolating needles of evidence from a haystack of network data.
- **Command-Line Traffic Capture with tcpdump:** Learning tcpdump complements the graphical analysis of Wireshark. Its efficiency and scriptability make it indispensable for use on remote servers, in automated scripts, or in resource-constrained environments where a GUI is unavailable.

The skills acquired in this module are directly applicable to numerous real-world cybersecurity scenarios. For example, if a SOC alert flags a potential data exfiltration event, the tcpdump skills learned here would allow for immediate, targeted traffic capture on the suspect server, even without a GUI, to validate the threat.

This module has solidified my understanding of network communications and provided a robust foundation in the tools of the trade.