Week2: **Assignment 3:- HTB Academy: Introduction to Network Traffic Analysis**

Report by: **Aisha Khalifan, cs-cns04-23014**

Introduction

Log in to your Hack The Box academy account
on https://academy.hackthebox.com/course/preview/intro-to-network-traffic-analysis

In the Modules section, select the **"Intro to Network Traffic Analysis (Tier 0)" or use the following link: https://academy.hackthebox.com/module/details/81.** This module will explore principles of network traffic analysis and practically demonstrate the use of traffic analysis tools such as Wireshark and tcpdump.

Network Traffic Analysis (NTA) is the process of examining network traffic to better understand how your network is used and to identify potential threats. NTA can help security specialists to:

- Identify common ports and protocols: NTA can help you to identify the ports and protocols that are most commonly used in your network. This information can be used to establish a baseline for normal network behavior and to detect anomalies that may indicate a security threat.
- Establish a baseline: NTA can be used to establish a baseline for normal network traffic. This baseline can then be used to detect anomalies that may indicate a security threat.
- Detect and respond to threats: NTA can be used to detect a wide range of security threats, including malware, botnets, and denial-of-service attacks. Once a threat has been detected, NTA can be used to investigate the threat and to take steps to mitigate it.
- Gain visibility: NTA can provide security specialists with visibility into all of the traffic on their network. This visibility can be used to identify potential threats and to troubleshoot network problems.

NTA is an important tool for security specialists because it can help them to protect their networks from a wide range of threats.

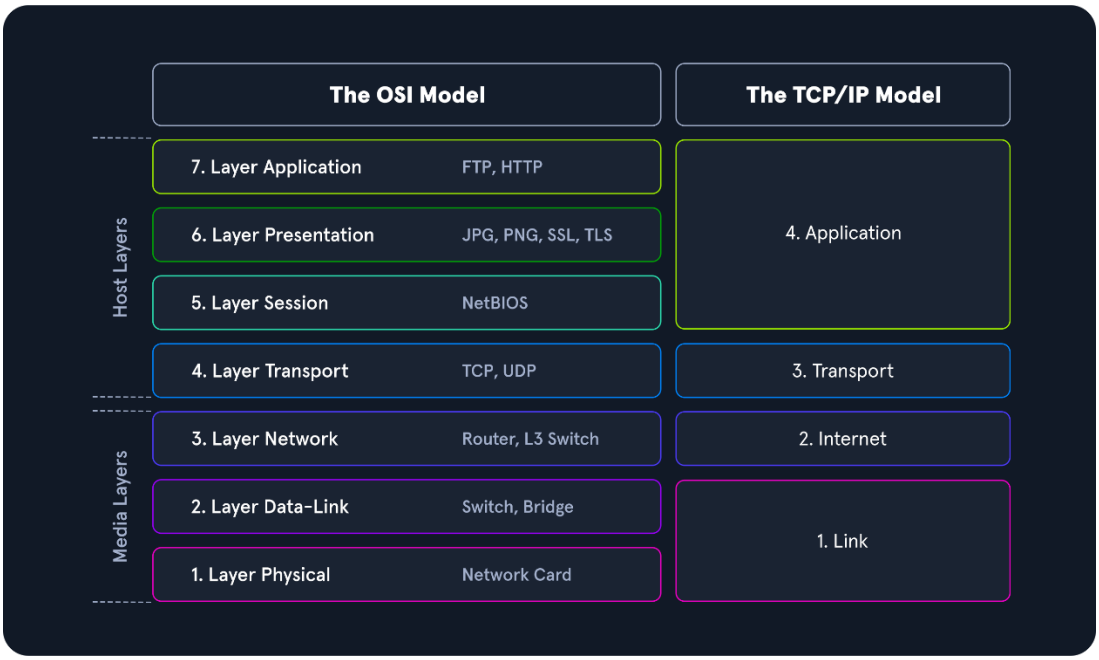
To add to the above, NTA is particularly useful for detecting and responding to attacks that leverage legitimate credentials and tools. This is because NTA can monitor all of the traffic on the network, including traffic that is generated by legitimate users and applications. By monitoring all of the traffic, NTA can identify anomalies that may indicate a malicious attack, even if the attacker is using legitimate credentials and tools.

Overall, NTA is a powerful tool that can help security specialists to protect their networks from a wide range of threats, both old and new.

Answers to questions

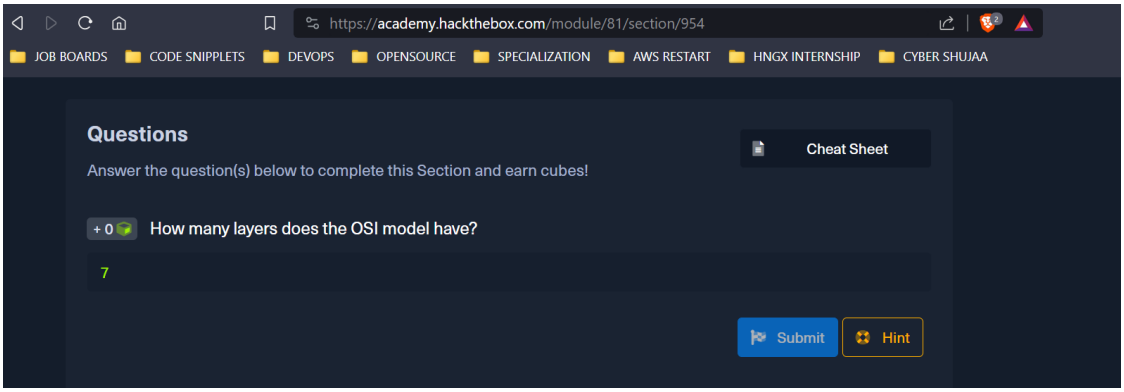
Introduction

Used the following reference to answer my part A and part B



A. Networking Primer - Layers 1-4

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

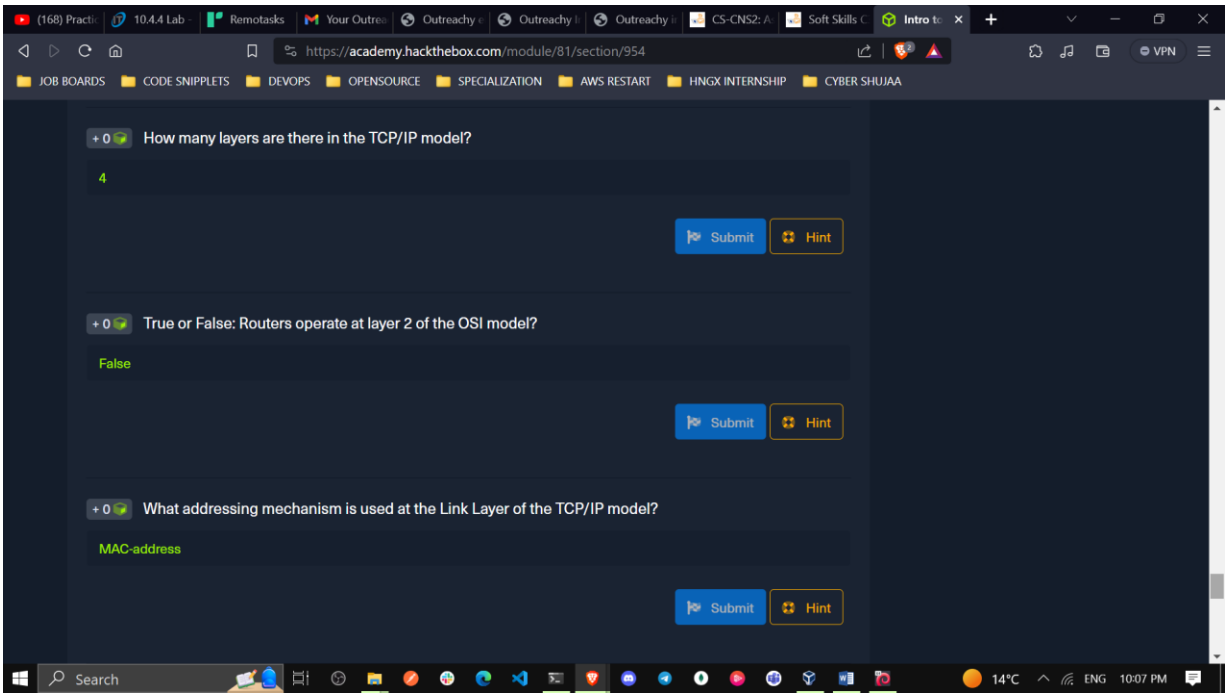


b. How many layers are there in the TCP/IP model?
4

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

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

MAC-address



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

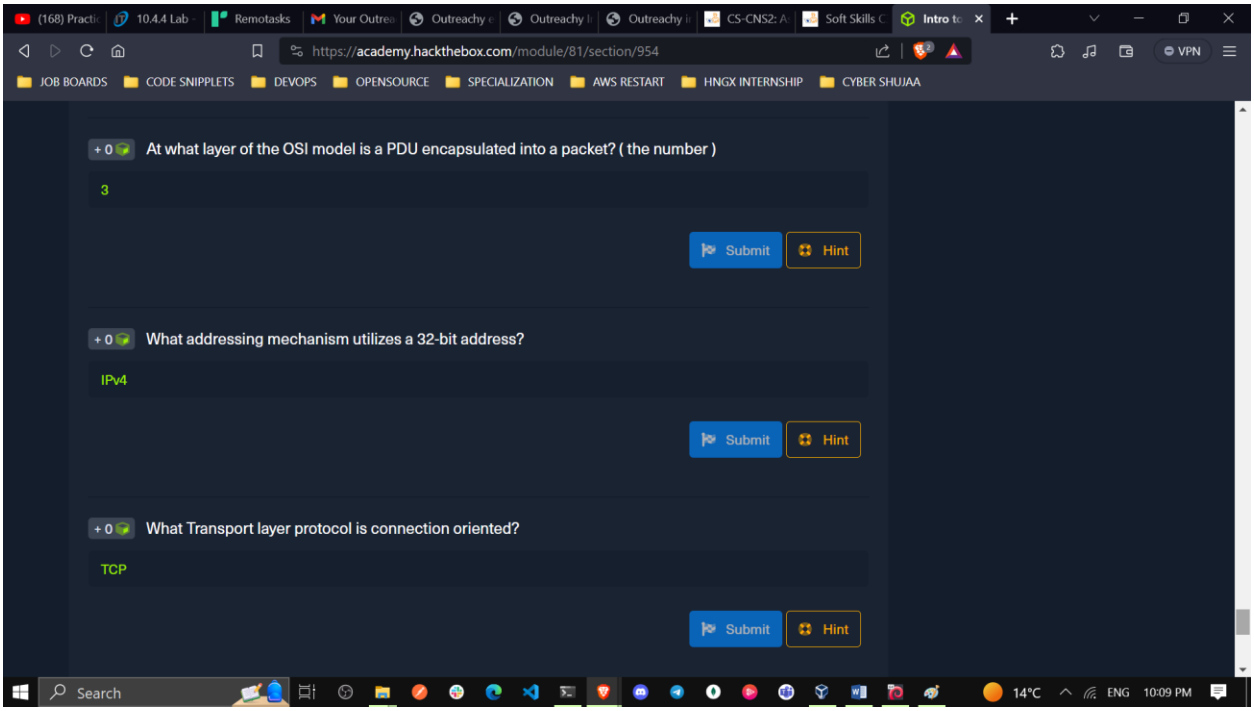
3

f. What addressing mechanism utilizes a 32-bit address?

IPv4

g. What Transport layer protocol is connection oriented?

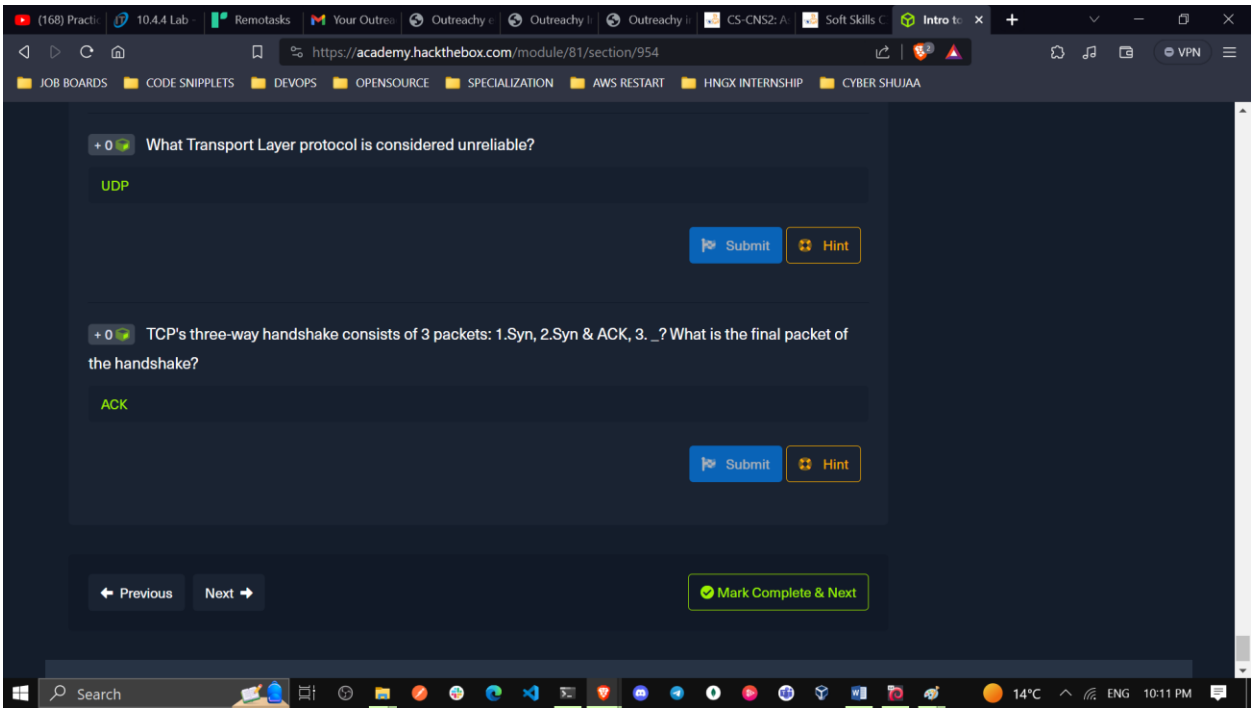
TCP



h. What Transport Layer protocol is considered unreliable?

UDP

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



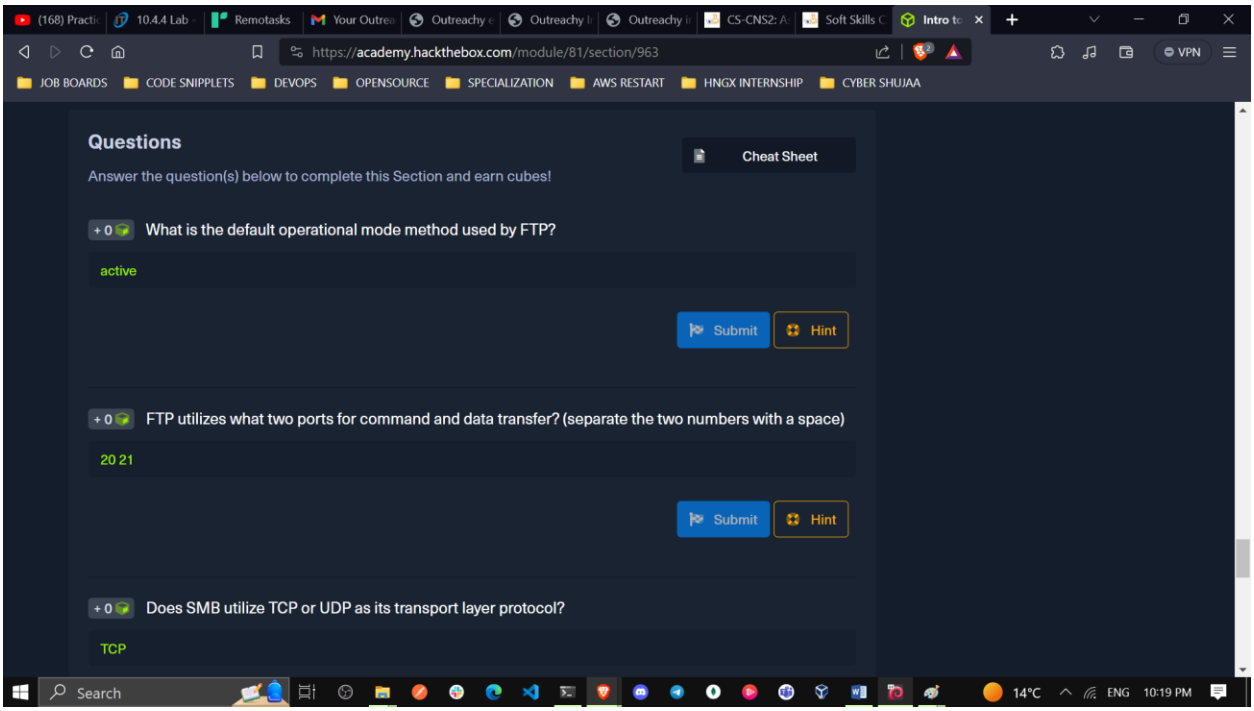
B. Networking Primer - Layers 5-7

It takes many different applications and services to maintain a network connection and ensure that data can be transferred between hosts.

a. What is the default operational mode method used by FTP?
active

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

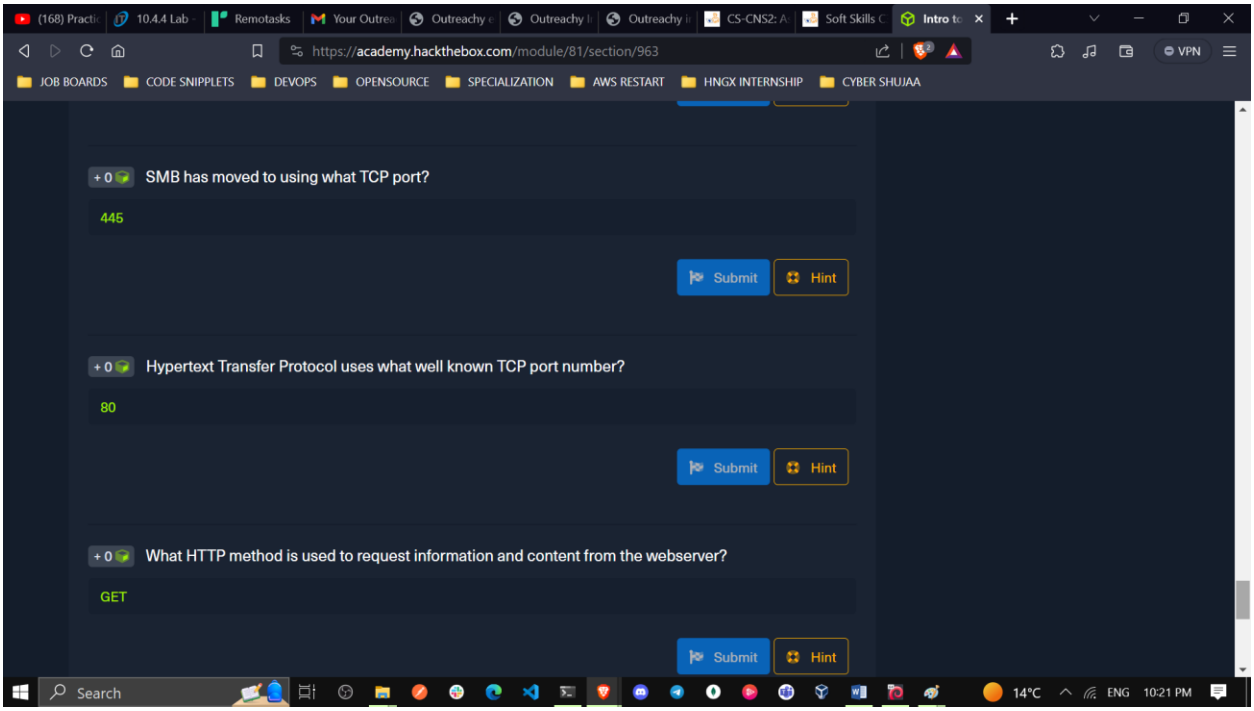
c. Does SMB utilize TCP or UDP as its transport layer protocol?
TCP



d. SMB has moved to using what TCP port?
445

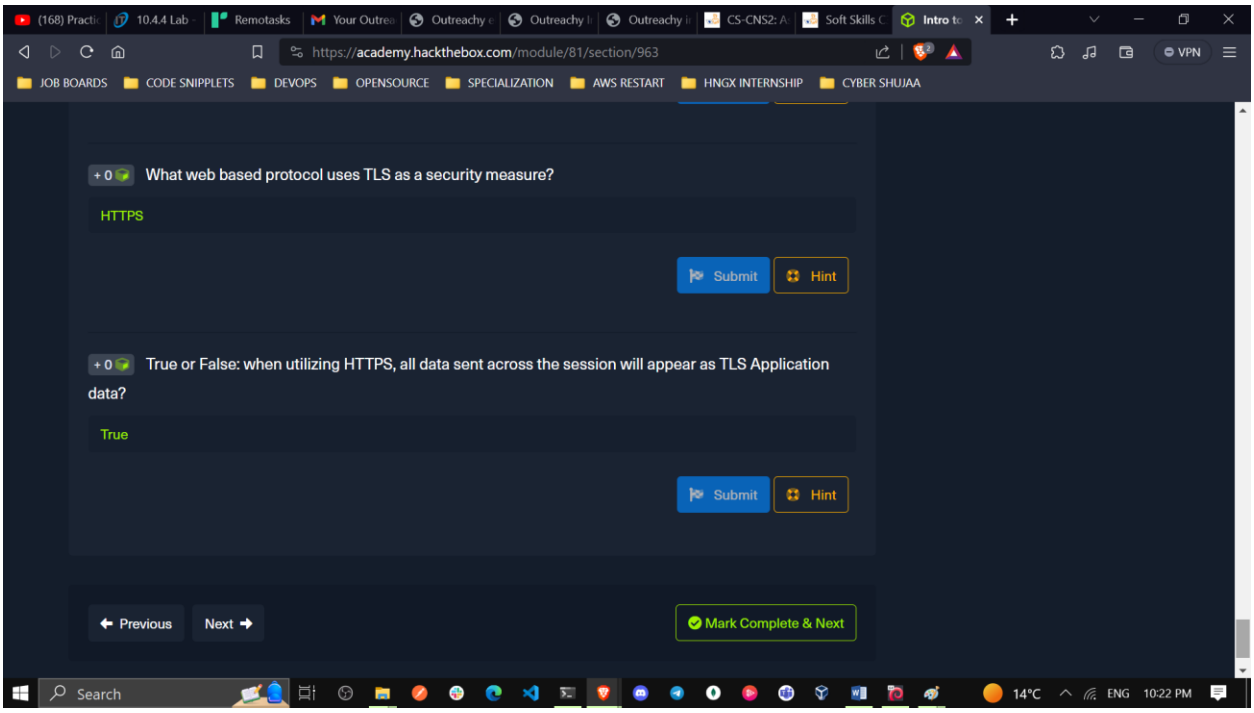
e. Hypertext Transfer Protocol uses what well known TCP port number?
80

f. What HTTP method is used to request information and content from the webserver?
GET



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

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



C. The Analysis Process

Network Traffic Analysis is a vital and dynamic process, adaptable based on available tools, organizational permissions, and network visibility. Its objective is to establish a repeatable analysis process. This involves dissecting network data, identifying irregularities that could indicate malicious activity, and understanding traffic trends against a baseline.

Traffic analysis offers essential insights for both proactive defense and daily operations troubleshooting. It can be performed actively or passively, depending on permissions and tools available, with key dependencies including permissions, capture tools, in-line placement, network tap or multiple NICs, and adequate storage and processing power.

Understanding daily traffic patterns through a baseline is crucial for efficient analysis and anomaly detection. This analysis is pivotal in swiftly identifying and mitigating potential network breaches

Traffic Capture Dependencies

They can be of two types: **Passive and active as shown in the table below:**

Dependencies	Type	Description
Permission	**Passive/ Active**	**Always ask for written permission from the right authority before capturing data, as it could be against the rules or laws in some organizations, especially in sensitive sectors like healthcare or banking. Stay legal and ethical, even if you consider yourself a hacker.**
Mirrored Port	**Passive**	**To capture data effectively, configure a switch or router interface to copy data to a specific port while enabling promiscuous mode on your NIC. This allows inspection of traffic not usually visible on other links.**

Capture Tool	**Passive/ Active**	**To process traffic, use tools like Wireshark on a capable computer. Be cautious as filtering large PCAP files can strain system resources. Ensure the host has sufficient power.**
In-line Placement	**Active**	**Placing a Tap in-line requires a topology change for the network you are working in. The source and destination hosts will not notice a difference in the traffic, but for the sake of routing and switching, it will be an invisible next hop the traffic passes through on its way to the destination.**
Network Tap or Host With Multiple NIC's	**Active**	**A computer with two NIC's, or a device such as a Network Tap is required to allow the data we are inspecting to flow still**
Storage and Processing Power	**Passive/ Active**	**You will need plenty of storage space and processing power for traffic capture off a tap.**

D. Analysis in Practice

In this section we go through the components of a network analysis

Workflow for Traffic Analysis:

a. Descriptive Analysis:

1. Issue identification and scope definition.

b. Diagnostic Analysis:

2. Capture network traffic and filter components.
3. Understanding captured network traffic.

c. Predictive Analysis:

4. Note-taking and mind mapping of results.
5. Summary of analysis for decision-making.

d. Prescriptive Analysis:

6. Actions and solutions based on the workflow.

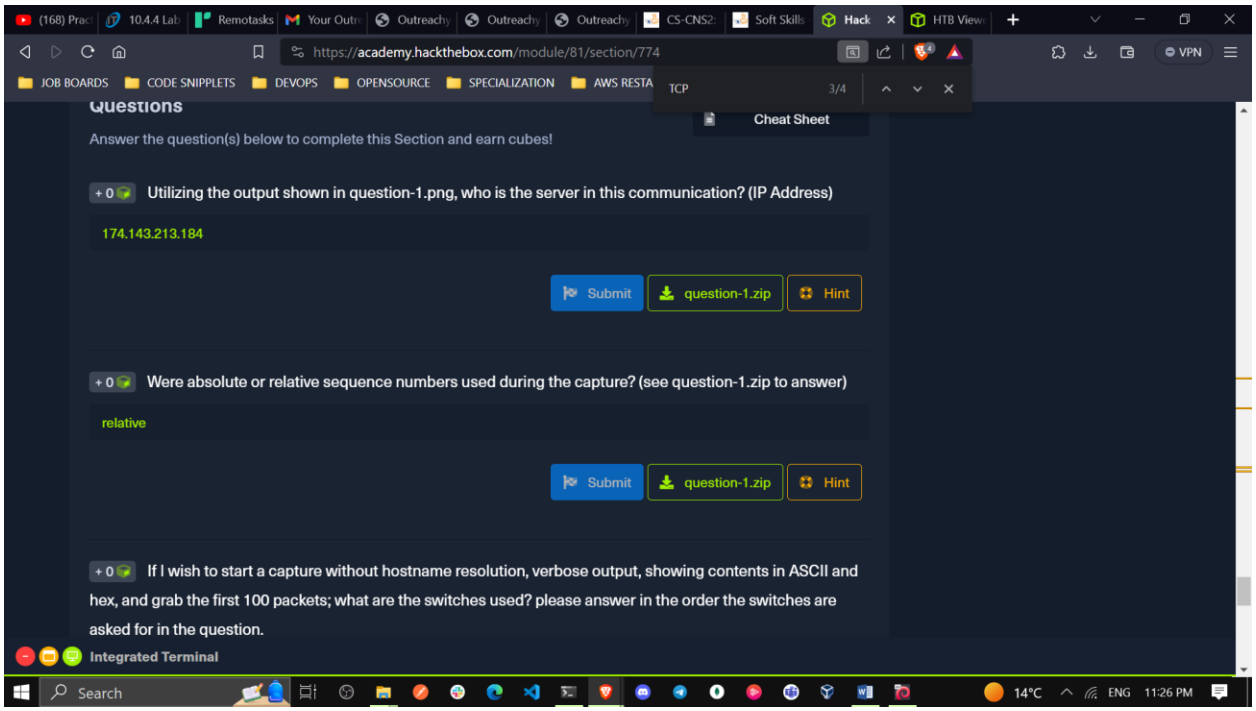
Key Components of Effective Analysis:

i. Know your environment: Asset inventory and network maps.
ii. Placement is key: Ideal tool placement for capturing traffic.
iii. Persistence: Continuous drive to identify and solve issues.

Analysis Approach:

Start with standard protocols and progress to specific ones.
Look for patterns and unusual events in network traffic.
Don't hesitate to seek assistance for thorough analysis.

E. Tcpdump Fundamentals



i. Utilizing the output shown in question-1.png, who is the server in this communication? (IP Address) unzip question-1.zip open question-1.PNG
They used this command: tcpdump –nnr HTTP.cap
174.143.213.184
It is the destination IP address

```
└─$ tcpdump -nnr 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 1:135, 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 1:1449, 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 2897:4345, 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
15:45:13.414005 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
15:45:13.464171 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
15:45:13.466771 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
15:45:13.466776 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
15:45:13.467401 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [F.], seq 135, 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 22046, ack 136, win 108, options [nop,nop,TS val 835172986 ecr 2216558], 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
```

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

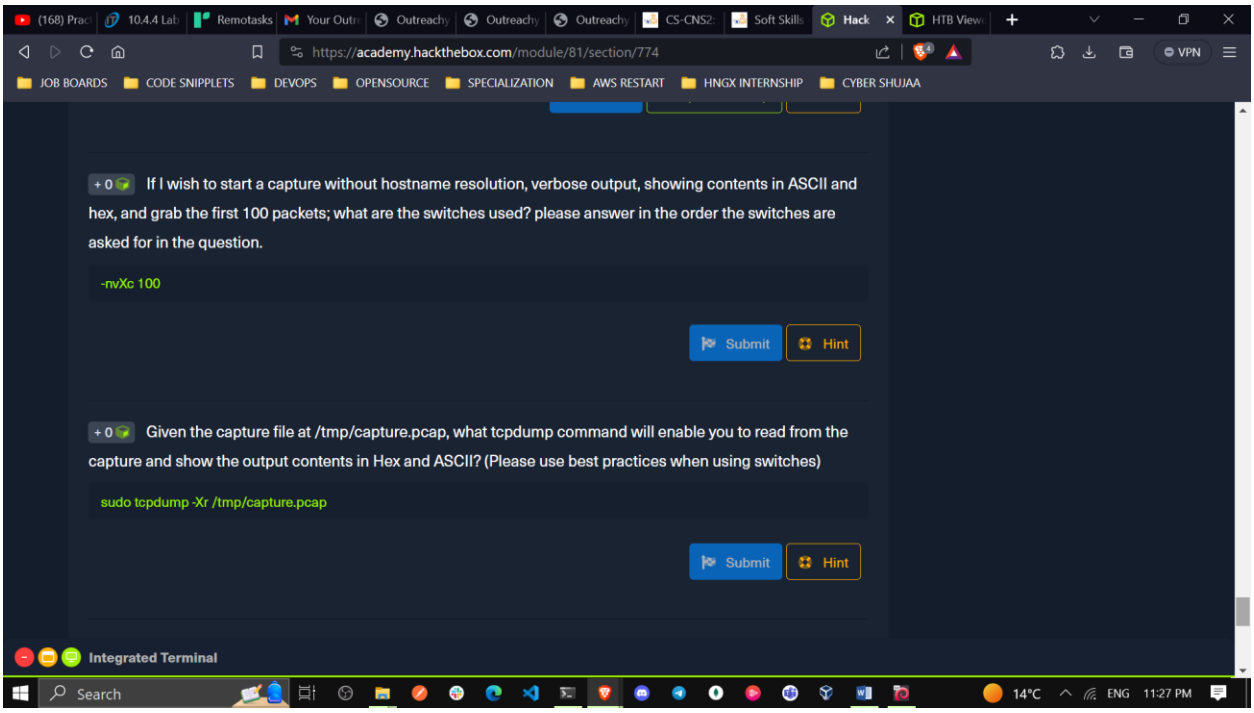
```
┌──(kali㉿kali)-[~]
└─$ tcpdump -nnr HTTP.cap
reading from file HTTP.cap, link-type EN10MB (Ethernet), snapshot length 65535
15:45:13.266821 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [S], seq 2387613953, win 5840, options [mss 1460,sackOK,TS val 2216538 ecr 0,nop,wscale 7], length 0
15:45:13.313726 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 1:135, 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 1:1449, 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 2897:4345, 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
15:45:13.414005 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
15:45:13.461382 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
15:45:13.464171 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
15:45:13.466771 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
15:45:13.466776 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
15:45:13.467401 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [F.], seq 135, 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 22046, ack 136, win 108, options [nop,nop,TS val 835172986 ecr 2216558], 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
```

iii. 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

iv. 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



v. What TCPDump switch will increase the **verbosity** of our output? (Include the — with the proper switch)

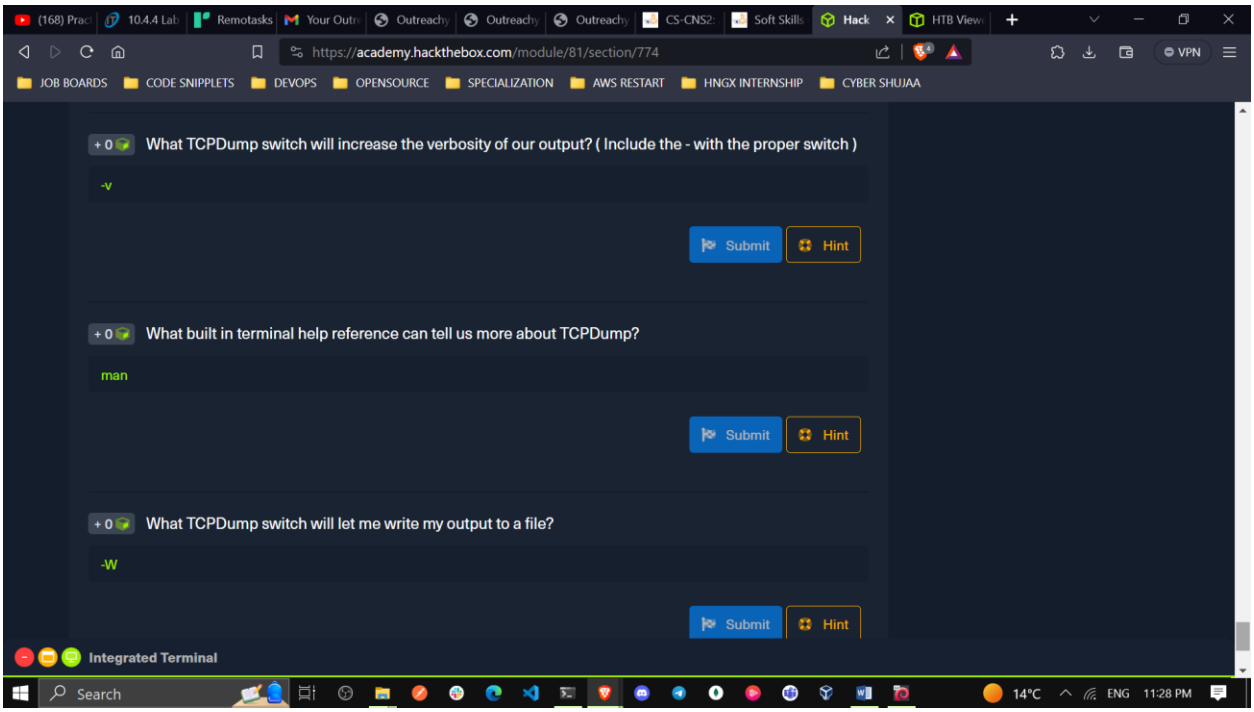
-v

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

man

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

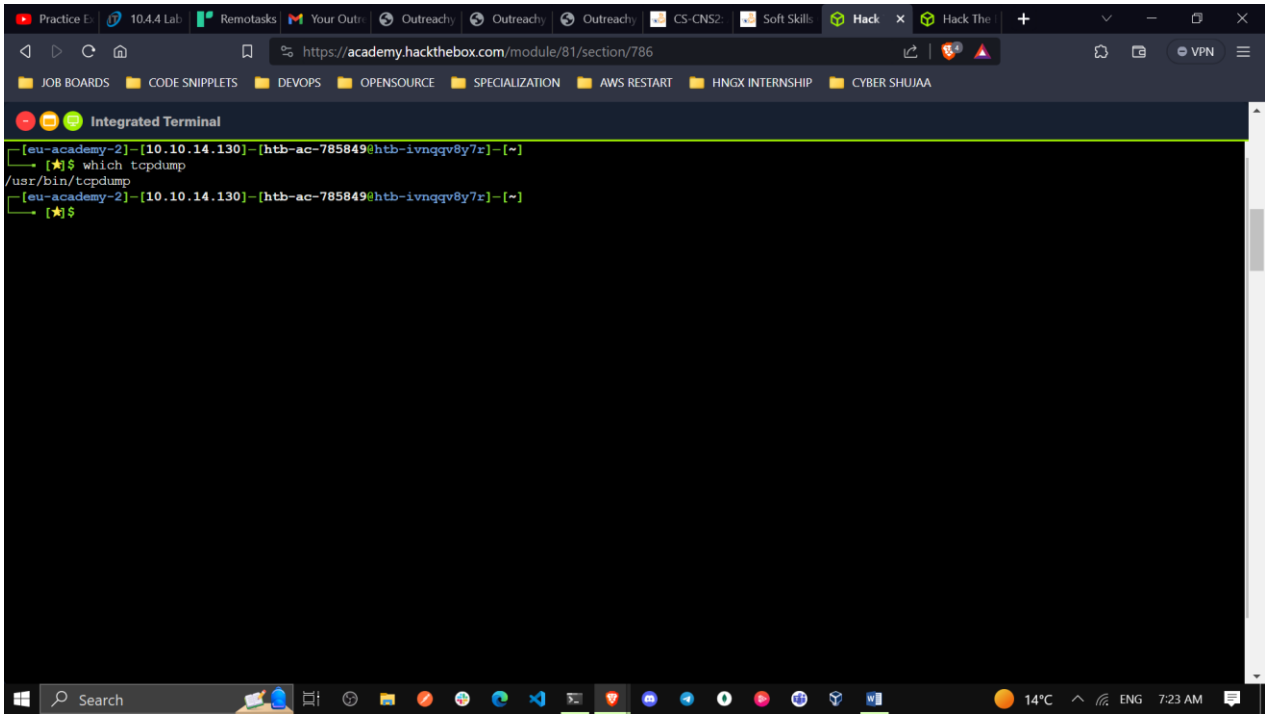
—W



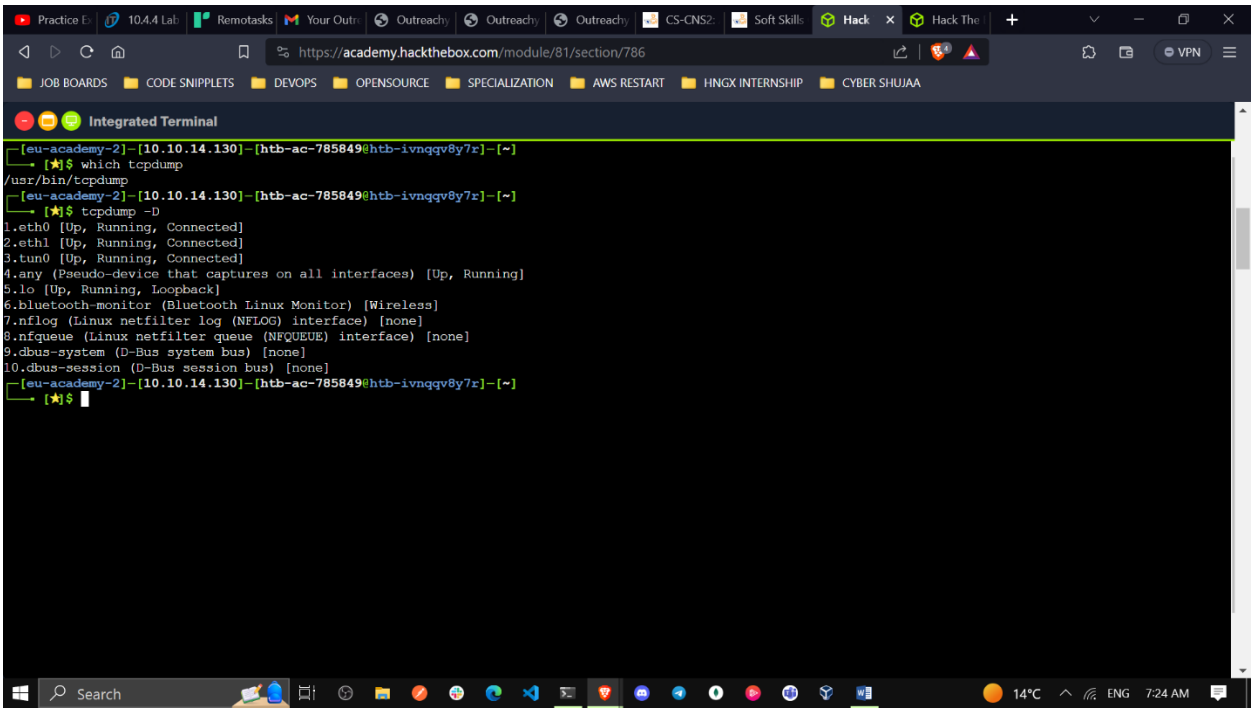
F. Capturing with Tcpdump(Fundamentals Labs)

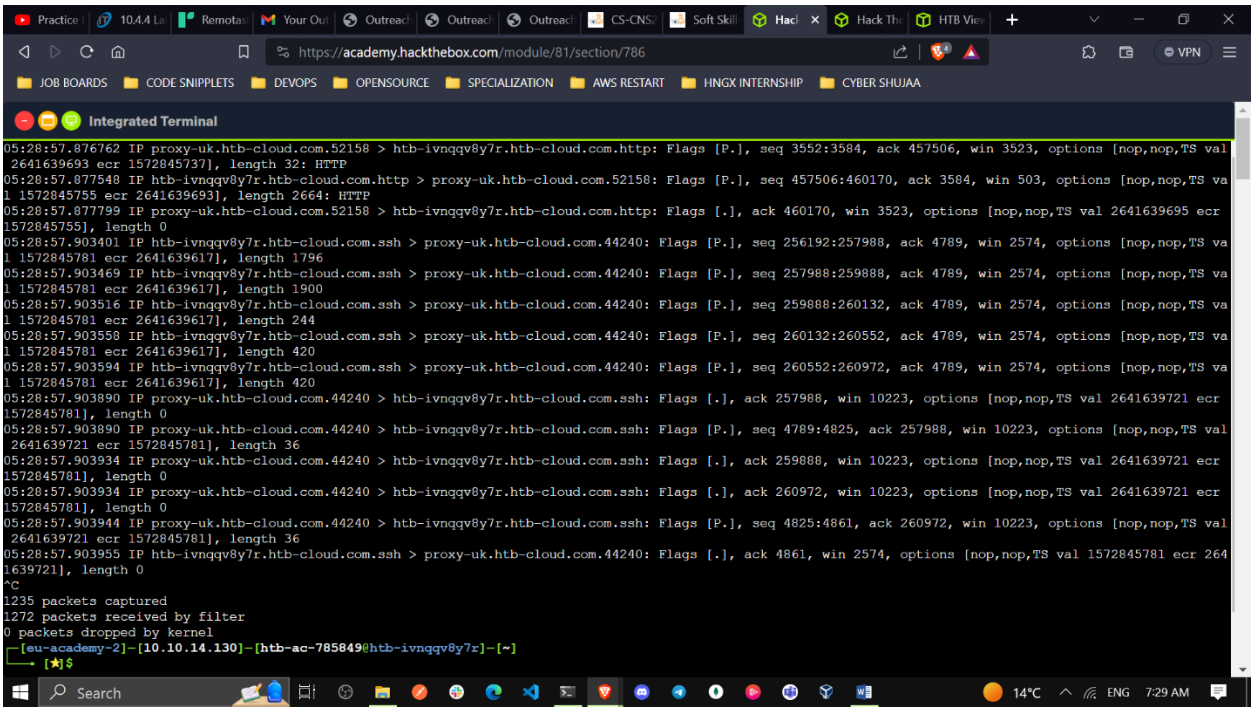
Tasks

Task #1:Validate Tcpdump is installed on our machine.



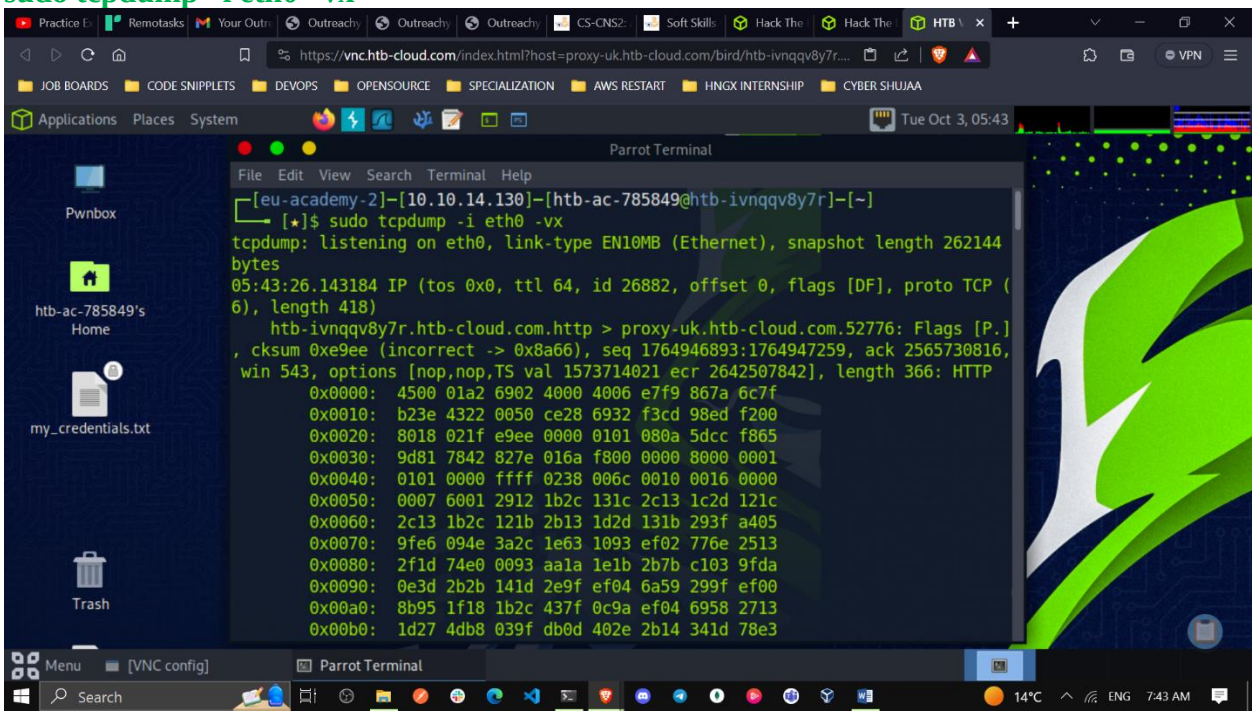
Task #2: Start a capture





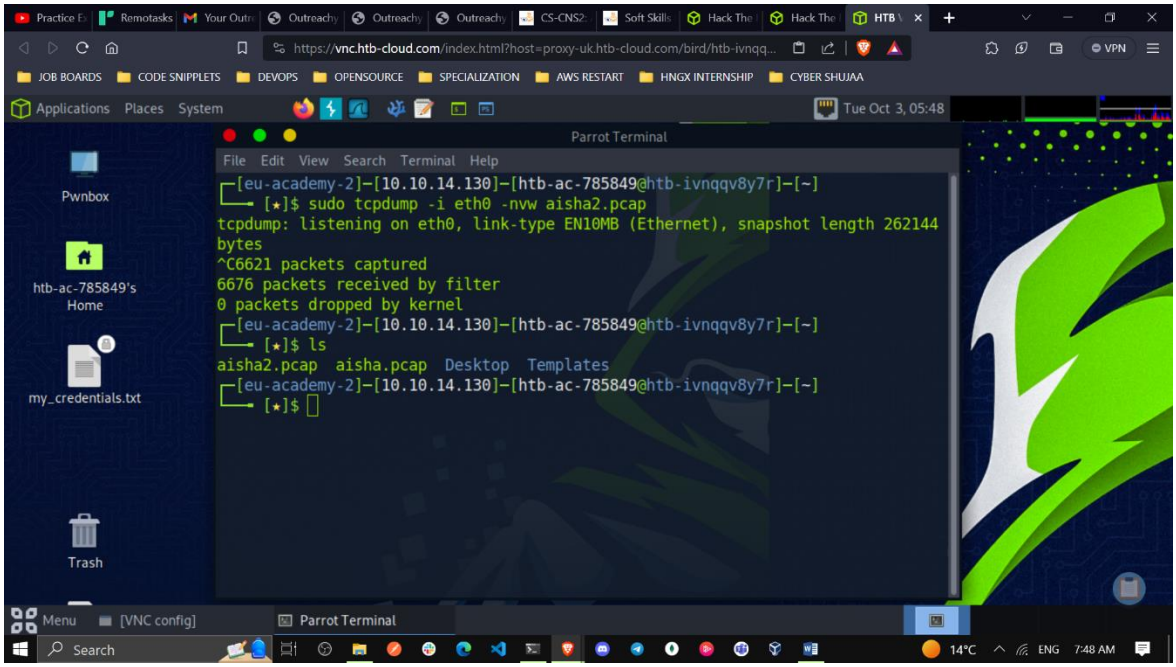
Task #3: Utilize Basic Capture Filters

sudo tcpdump –i eth0 –vx

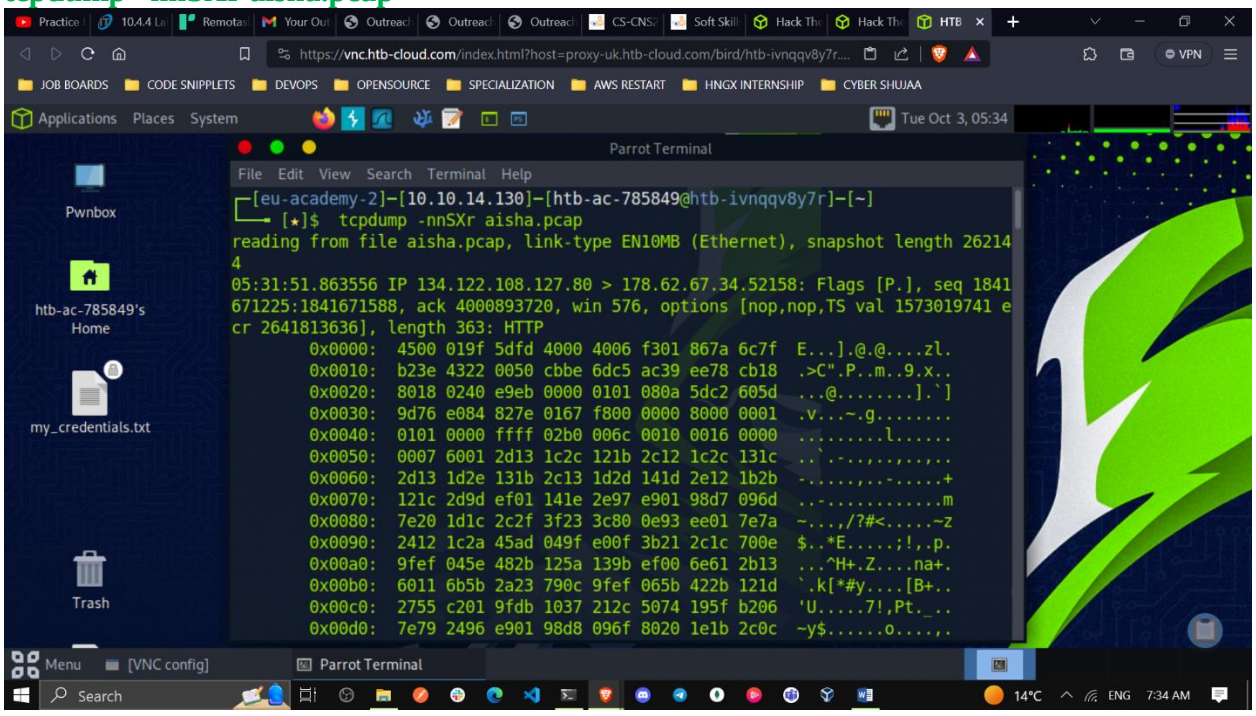


Task #4: Save a Capture to a .PCAP file.

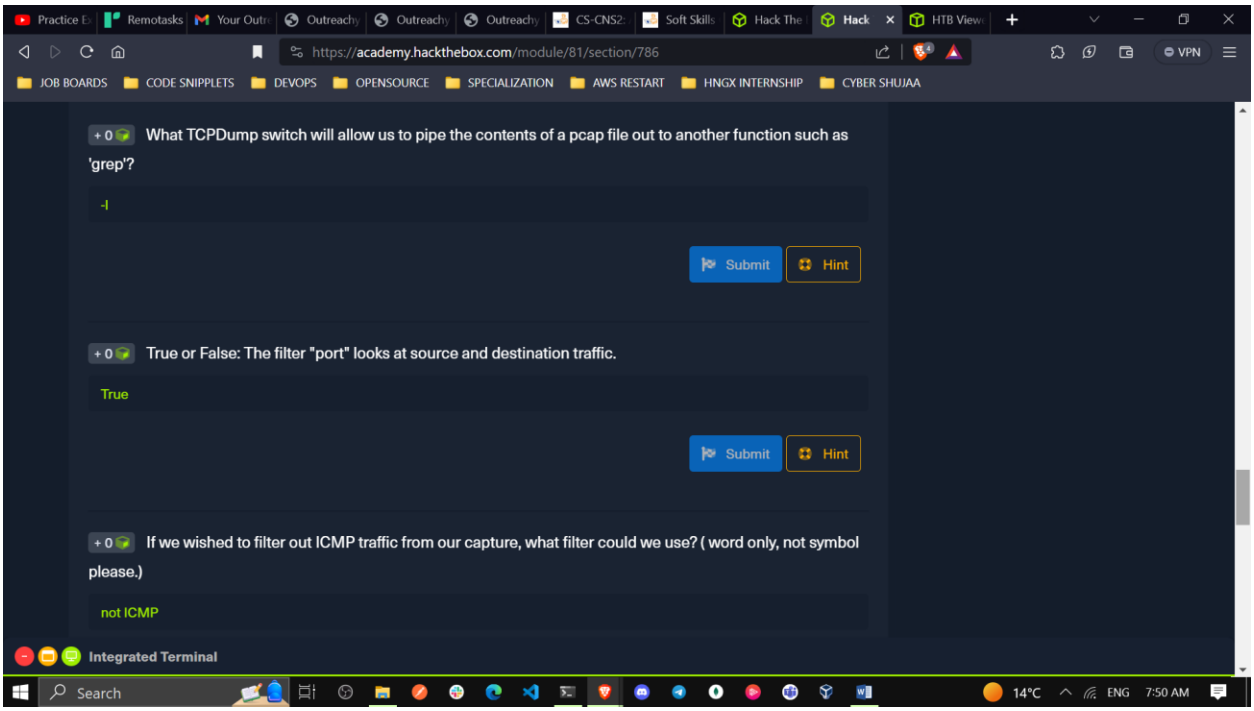
sudo tcpdump –i eth0 –nvw aisha.pcap



Task #5: Read the Capture from a .PCAP file.
tcpdump –nnSXr aisha.pcap



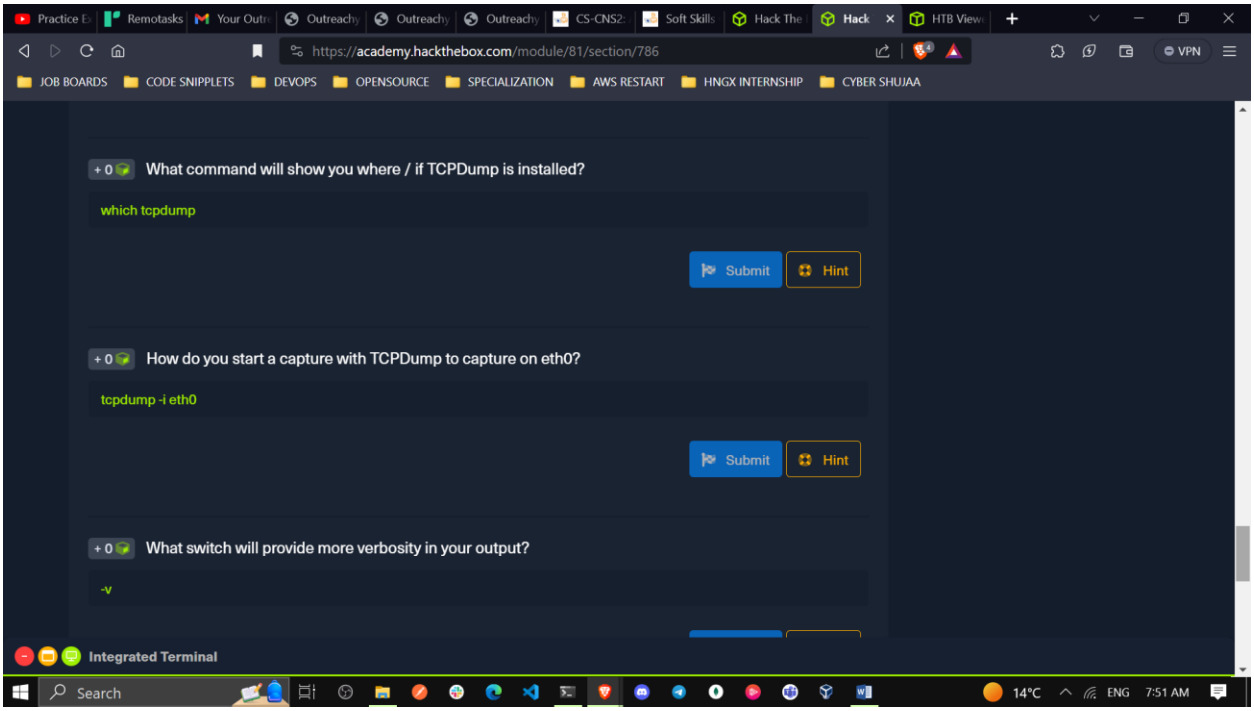
i. What TCPDump switch will allow us to **pipe the contents** of a pcap file out to another function such as ‘grep’? **-l**
ii. True or False: The filter “port” looks at source and destination traffic. **True**
iii. If i wished to **filter out ICMP traffic** from out capture, what filter could we use? (word only, not symbol please.) **not icmp**



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

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

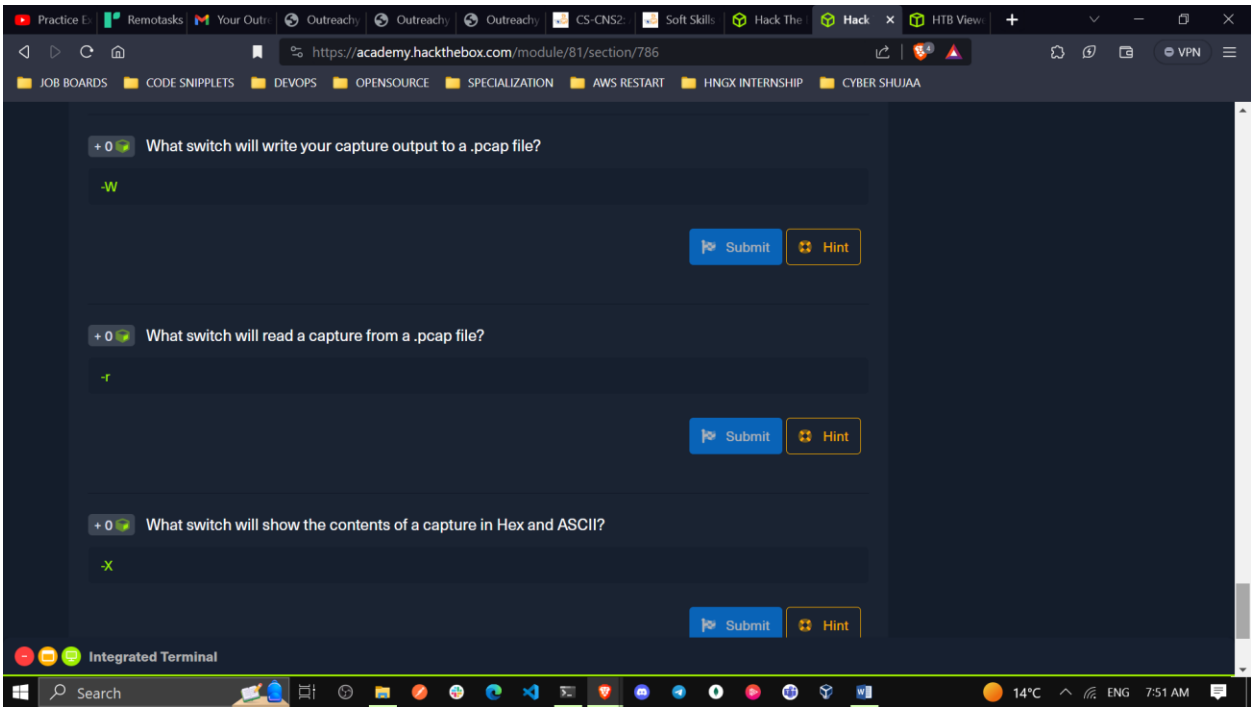
vi. What switch will provide **more verbosity** in your output? **–v**



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

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

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



G. Tcpdump Packet Filtering

i. What filter will allow me to see traffic coming from or **destined to the host** with an **ip of 10.10.20.**1?

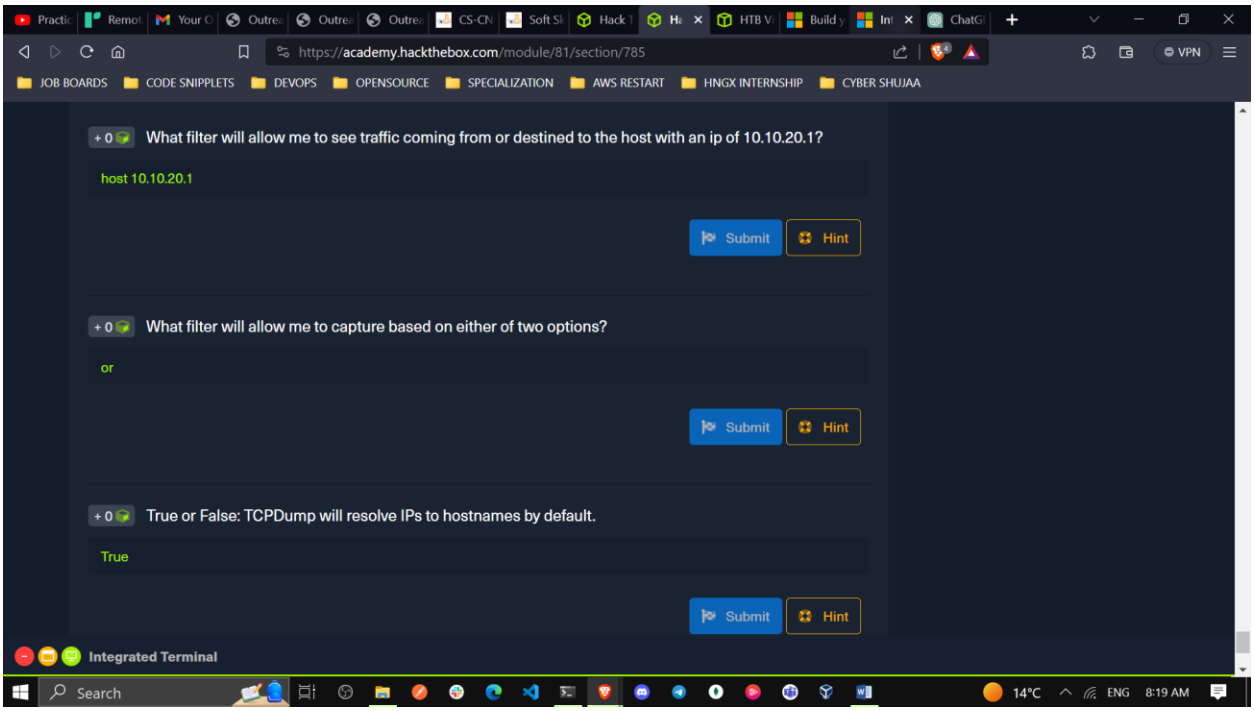
host 10.10.20.1

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

or

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

True



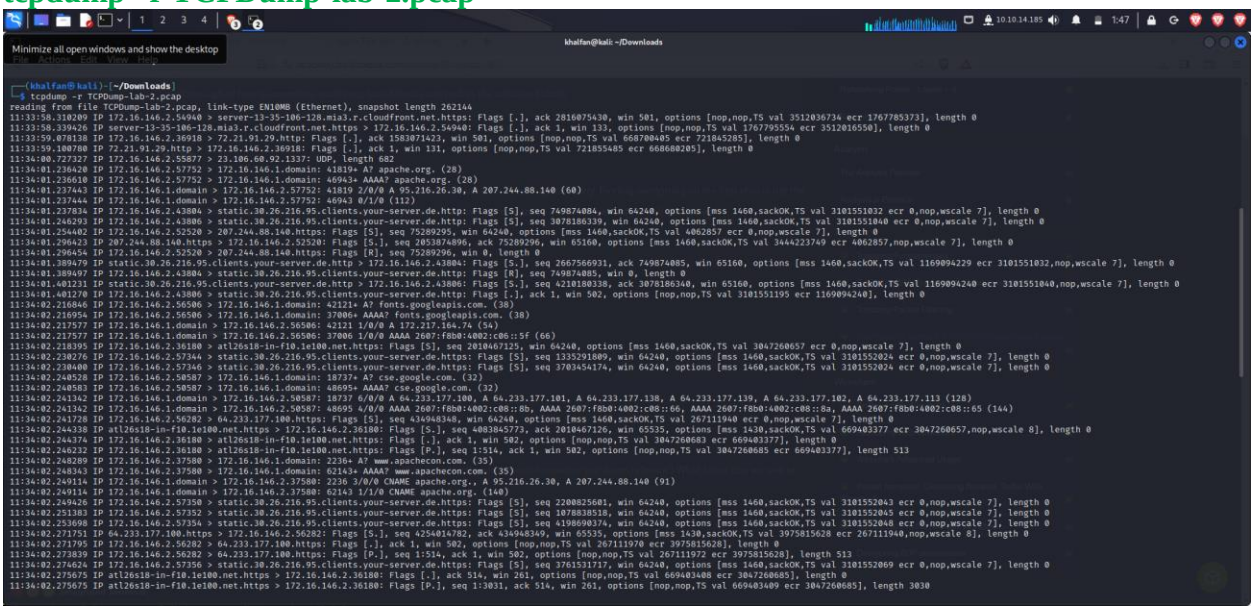
H. Interrogating Network Traffic with Capture and Display Filters

Tasks

Utilizing TCPDump-lab-2.zip in the optional resources, perform the lab to the best of your ability. Finding everything on the first shot is not the goal.

Task1: **Read a capture from a file without filters implemented**

tcpdump –r TCPDump-lab-2.pcap



Task2: Identify the type of traffic seen.
Common protocols: TCP and UDP protocols. HTTP, HTTPS,DNS

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

- **The .https in the above packet means its port 443**
- **The source port is 54940 (IP 172.16.146.2, port 54940).**
- **The destination port is https, which typically refers to port 443 for HTTPS (HTTP over SSL/TLS).**

The following is a snippet of my output above

PROTOCOLS USED	PORTS	WHERE USED
TCP	54940	Used in communication between IP `172.16.146.2 and server-13-35-106-128.mia3.r.cloudfront.net for HTTPS.
TCP	36918	Used in communication between IP 172.16.146.2 and 72.21.91.29 for HTTP.
UDP	1337	Used in communication between IP 172.16.146.2 and 23.106.60.92 with UDP.
TCP	43804	Used in communication between IP 172.16.146.2 and static.30.26.216.95.clients.your-server.de for HTTP.
TCP	43806	Used in communication between IP 172.16.146.2 and static.30.26.216.95.clients.your-server.de for HTTP.
TCP	52520	Used in communication between IP 172.16.146.2 and 207.244.88.140 for HTTPS.
TCP	50587	Used in communication between IP 172.16.146.2 and 172.16.146.1 for DNS.

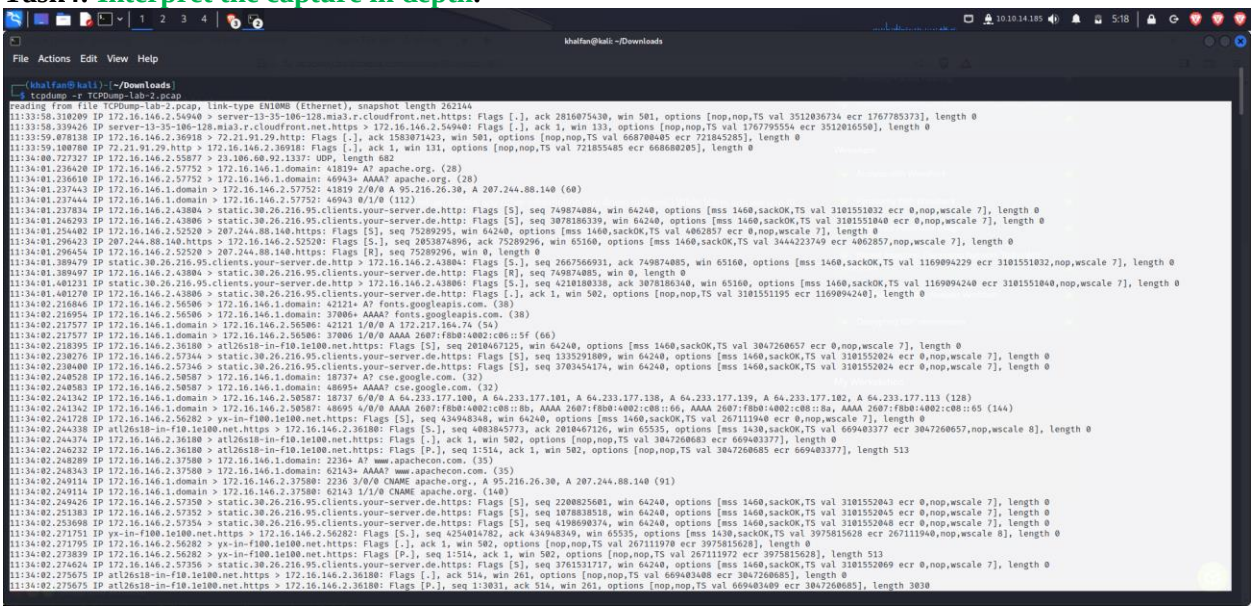
Task3: Identify conversations.

CONVERSATION	SOURCE IP/PORT	PORT	DESTINATION	PORT
1	172.16.146.2	54940	server-13-35-106-128.mia3.r.cloudfront.net	443(HTTPS)
2	server-13-35-106-128.mia3.r.cloudfront.net	443 (HTTPS)	172.16.146.2	54940

3	172.16.146.2	36918	72.21.91.29	80 (HTTP)
4	72.21.91.29	80(HTTP)	172.16.146.2	36918
5	172.16.146.2	55877	23.106.60.92	1337(UDP)
6	172.16.146.2	57752	172.16.146.1	53 (DNS)
7	172.16.146.1	53 (DNS)	172.16.146.2	57752
8	172.16.146.2	43804	static.30.26.216.95.clients.your-server.de	80(HTTP)
9	static.30.26.216.95.clients.your-server.de	80(HTTP)	172.16.146.2	43804

The above is just part of my network capture

Task4: Interpret the capture in depth.



What is the timestamp of the first established conversation in the pcap file? 11.33.58

What is the IP address/s of apache.org from the DNS server responses?

11:34:01.236420 IP 172.16.146.2.57752 > 172.16.146.1.domain: 41819+ A? apache.org. (28)

1:34:01.236420 - DNS query (A record) from IP 172.16.146.2 to DNS server for "apache.org".

11:34:01.236610 - DNS query (AAAA record) from IP 172.16.146.2 to DNS server for "apache.org".

What protocol is being utilized in that first conversation? (name/#)

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

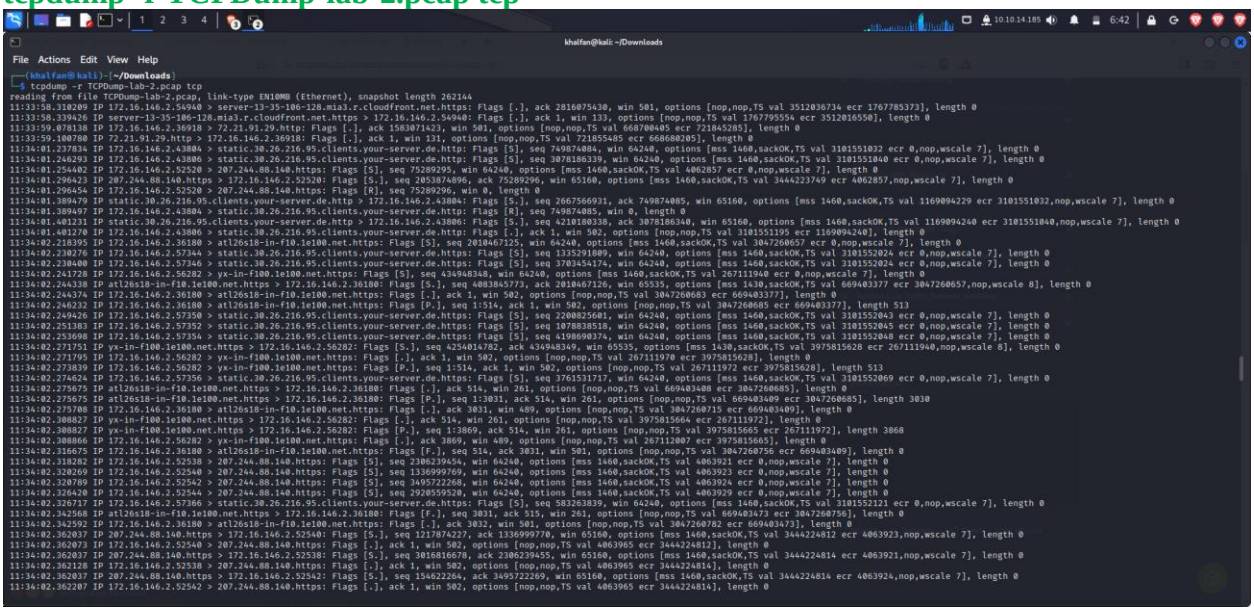
In this conversation the packet is sent from IP 172.16.146.2 on port 54940 to "server-13-35-106-128.mia3.r.cloudfront.net" on port 443 (HTTPS), is utilizing the TCP (Transmission Control Protocol) protocol. This is evident from the presence of TCP flags in the packet, specifically the "[.]" which indicates an acknowledgment (ACK) in TCP.

Task5: Filter out traffic.

For this task I have filtered in the previous tasks.

Task6: Filter for TCP traffic.

tcpdump -r TCPDump-lab-2.pcap tcp



- **Lines starting with "11:33:58.310209" and similar are individual packets captured at specific timestamps.**
- **Each line shows details about a TCP packet, including source and destination IP addresses, ports, TCP flags, and additional options.**
- **For example, the first line:**

  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

indicates a TCP packet sent from IP 172.16.146.2 on port 54940 to the server "server-13-35-106-128.mia3.r.cloudfront.net" on port 443 (HTTPS). The packet is an acknowledgment (ack) with certain TCP flags and options.

The tcp filter ensures that only TCP traffic is displayed.

Task7: What can you determine about the server in the first conversation.

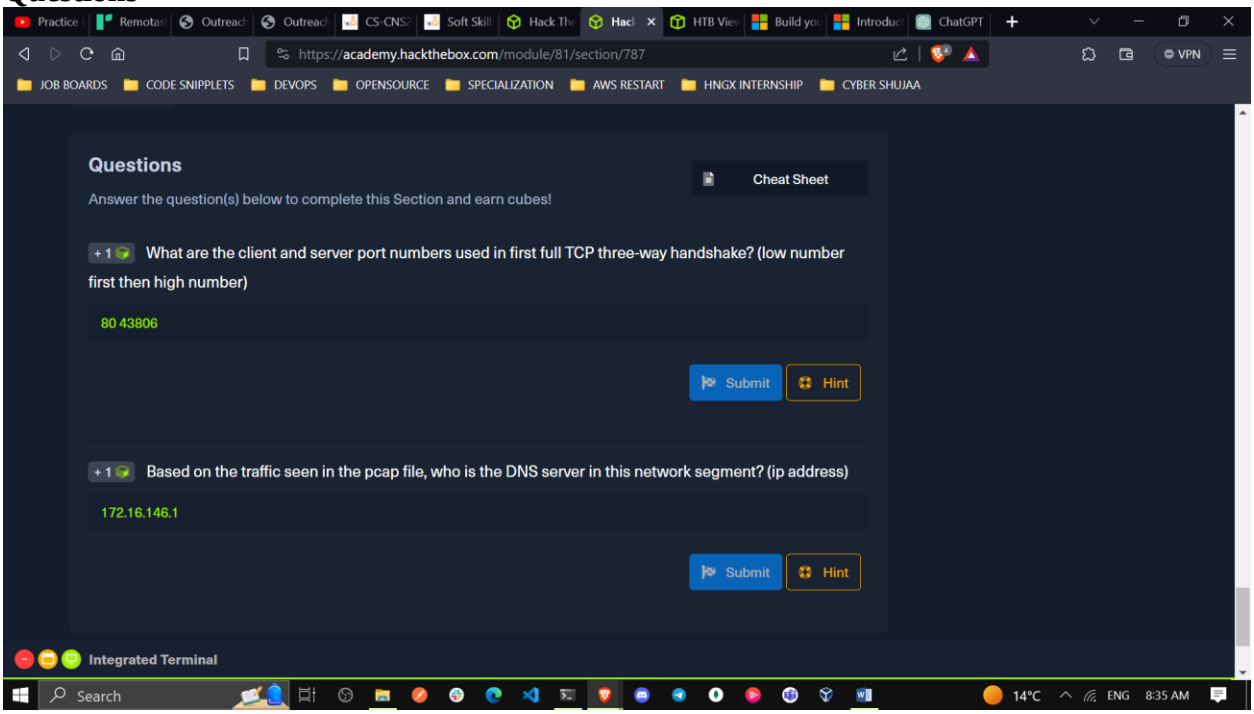
In the first conversation, the client (source) at IP address 172.16.146.2 is communicating with the server at the address "server-13-35-106-128.mia3.r.cloudfront.net" over HTTPS (port 443). The communication involves TCP flags and options.

Source IP: 172.16.146.2 and Source Port: 54940

Destination IP: server-13-35-106-128.mia3.r.cloudfront.net and Destination Port: 443 (HTTPS)

The client is sending an acknowledgment (ACK) to the server, indicating successful receipt of data or a previous message from the server. The server is responding with an acknowledgment (ACK) as well.

Questions

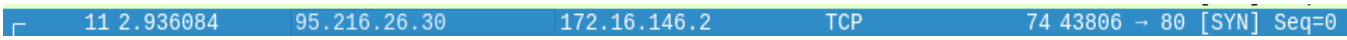


i. What are the **client** and **server port** numbers used in first full TCP three-way handshake? (low number first then high number)

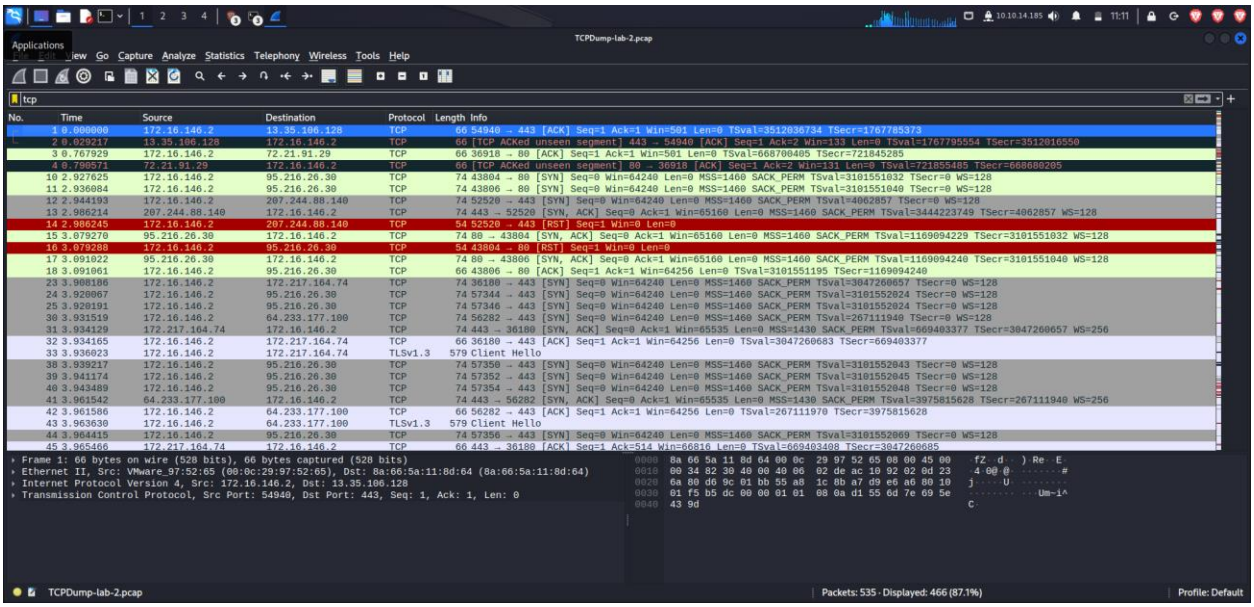
Instead of using wireshark, I used my terminal by checking it like this:

tshark -r TCPDump-lab-2.pcap -Y "tcp"

FILTER: tcp.port == 80



80 43806



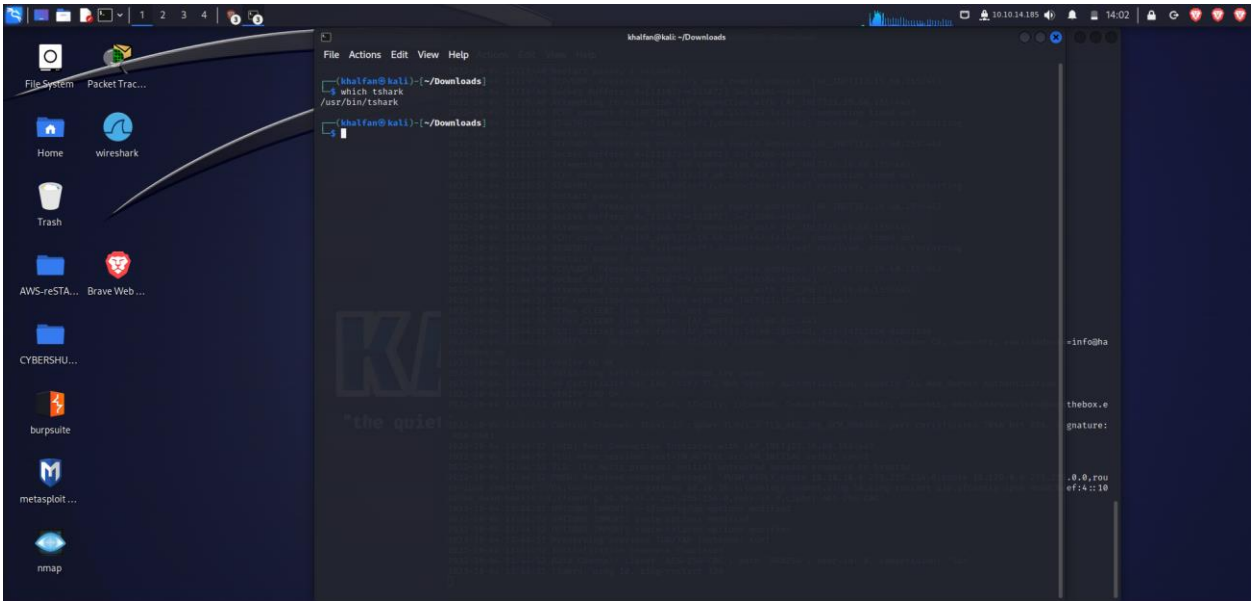
ii. Based on the traffic seen in the pcap file, who is the **DNS** server in this network segment? (ip address) Download: zeek zeek -C -r TCPDump-lab-2.pcap then cat dns.log

172.16.146.2

I. Analysis with Wireshark

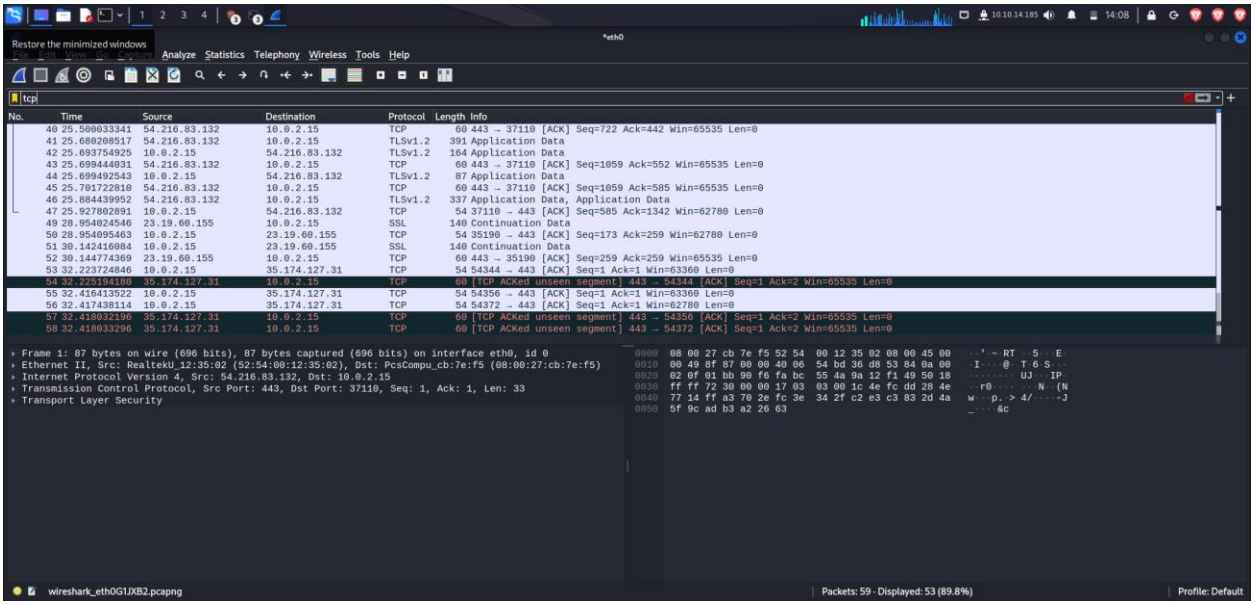
i. True or False: Wireshark can run on both Windows and Linux.

True



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

Packet List



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

Packet Bytes

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

-D

v. What switch allows us to apply filters in TShark?

-f

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

before

J. Familiarity With Wireshark

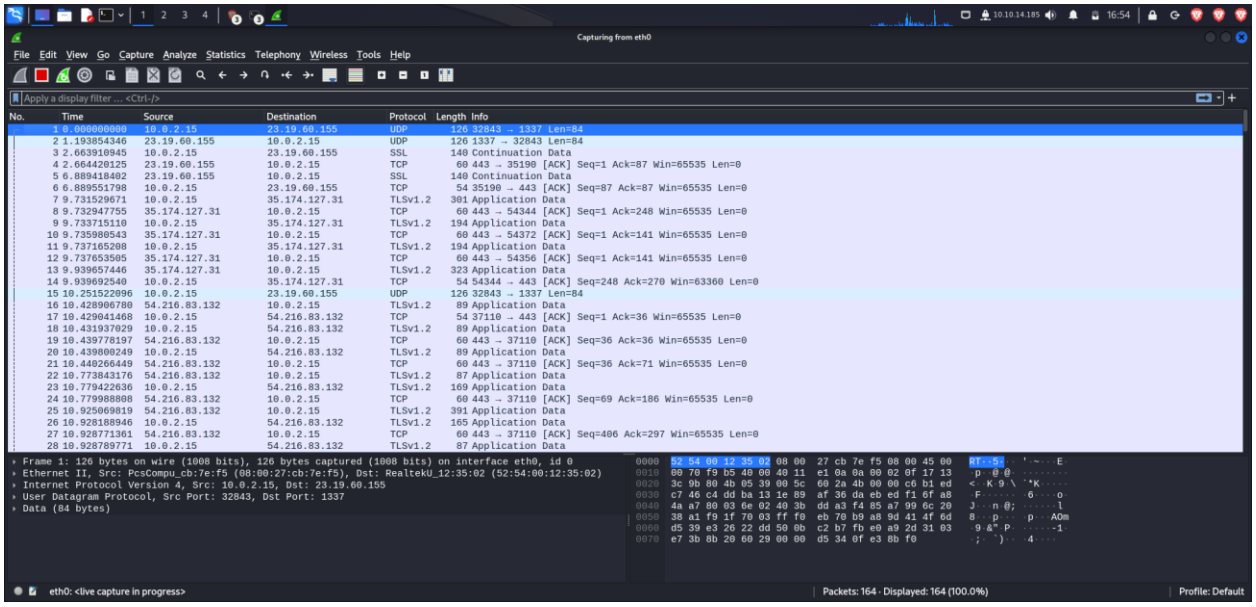
Tasks

Task #1

Validate Wireshark is installed, then open Wireshark and familiarize yourself with the GUI windows and toolbars.

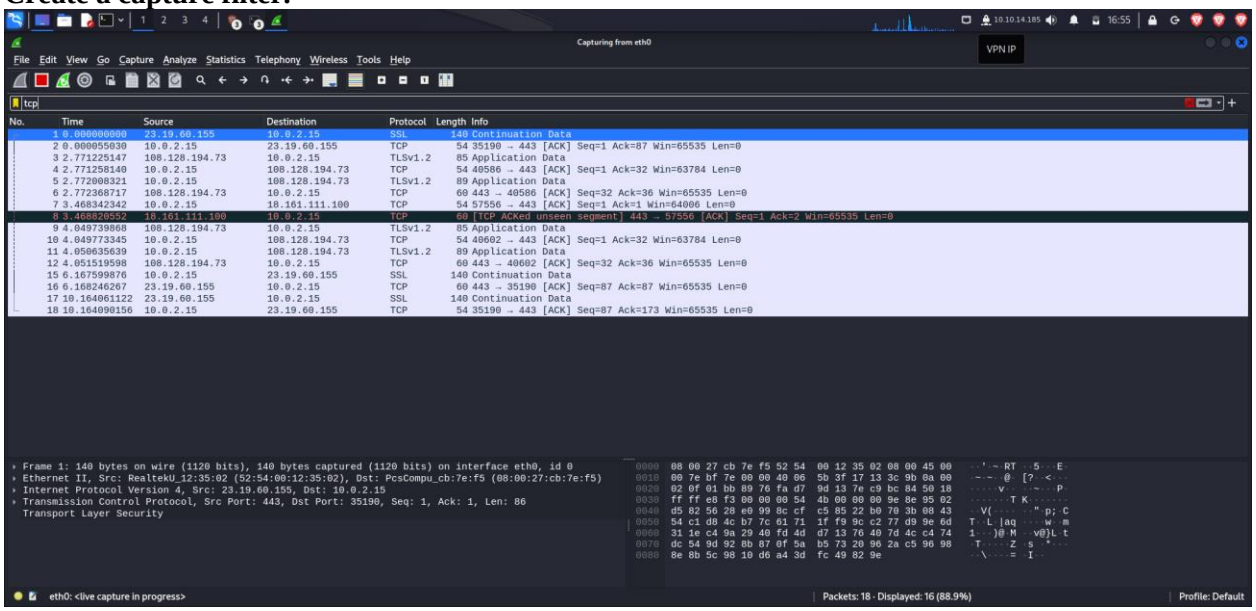
Task #2

Select an interface to run a capture on and create a capture filter to show only traffic to and from your host IP.



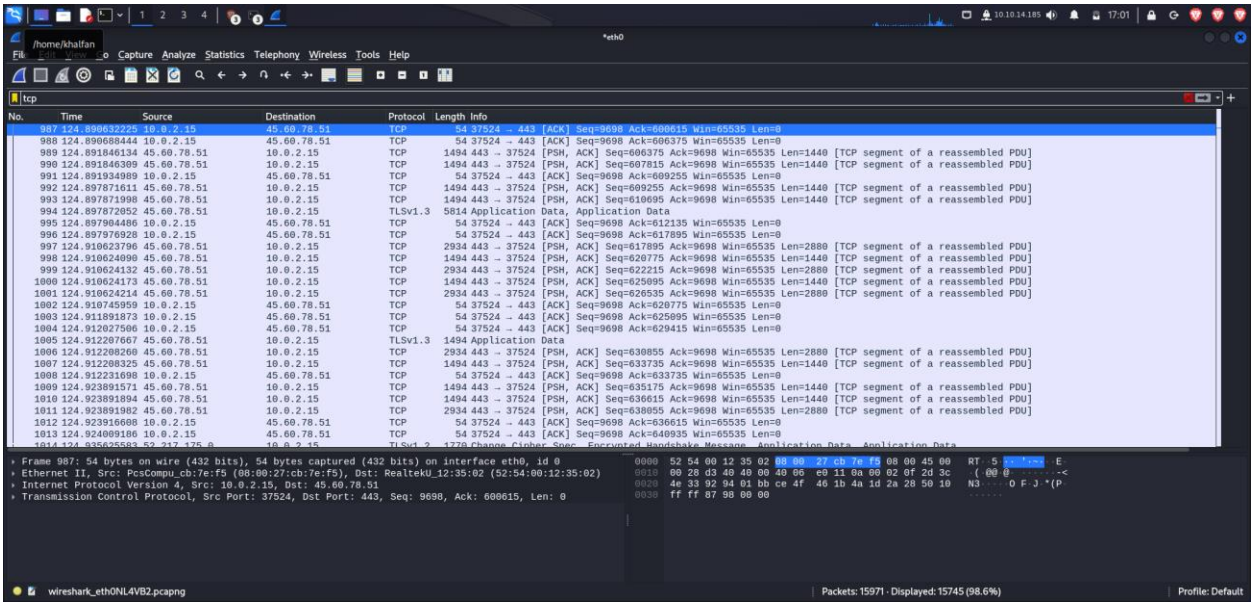
Task #3

Create a capture filter.



Task #4

Navigate to a webpage to generate some traffic.



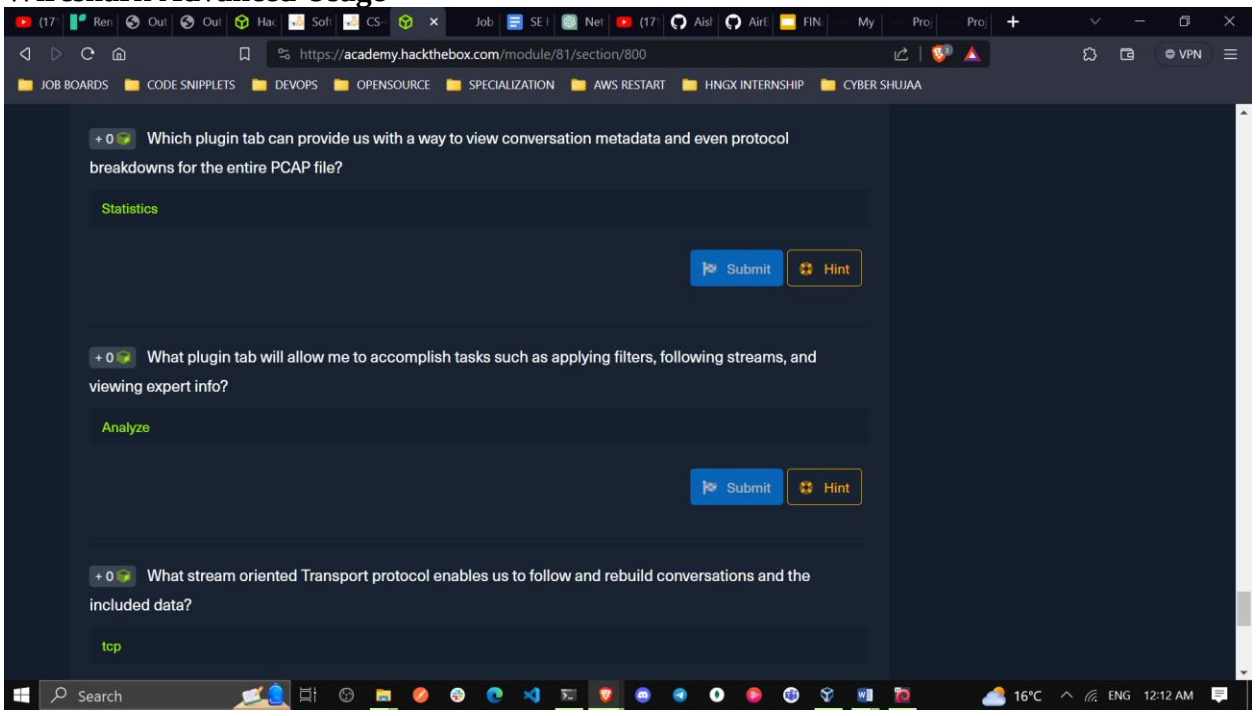
Task #5

Use the capture results to answer the following questions.

What application-level protocols are displayed in the results?

Can we discern anything in clear text? What was it?

K. Wireshark Advanced Usage

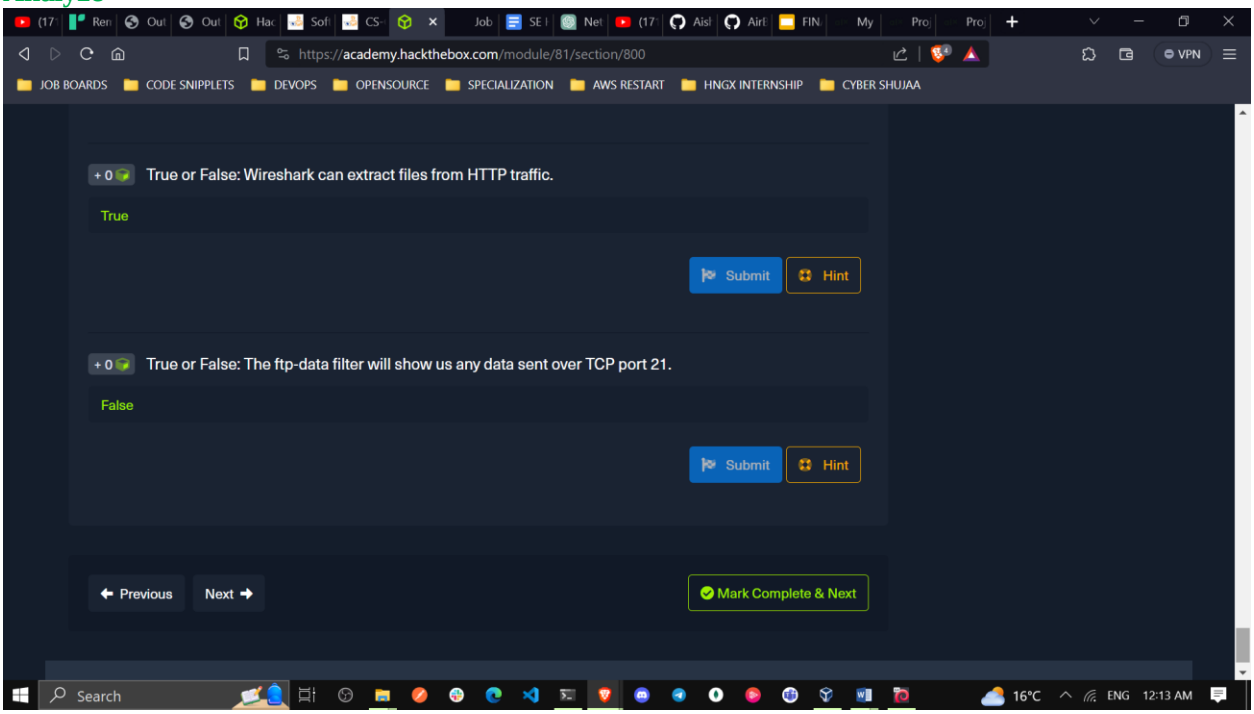


i. Which plugin tab can provide us with a way to view conversation metadata and even protocol breakdowns for the entire PCAP file?

Statistics

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

Analyze



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

tcp

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

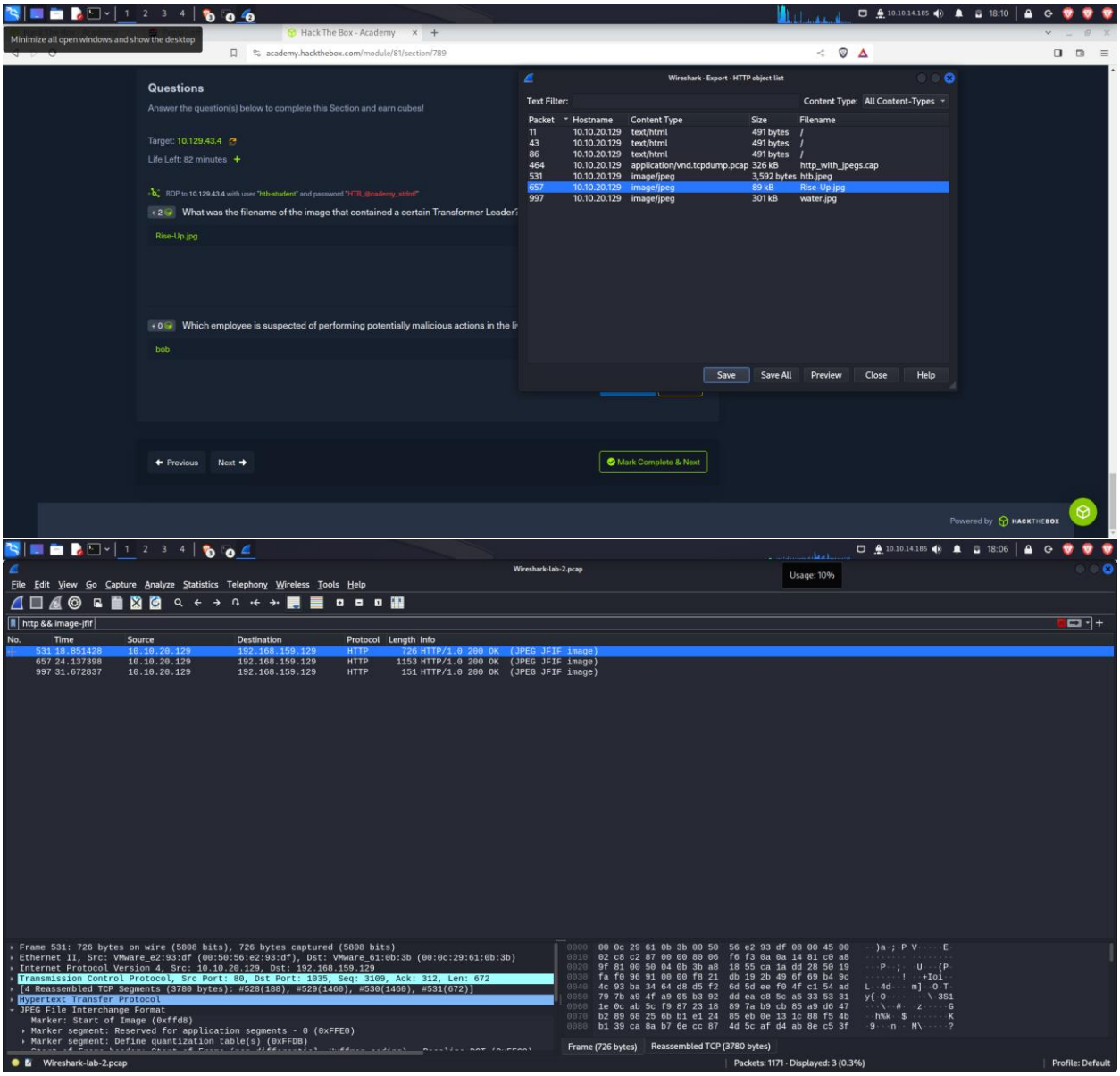
True

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

False

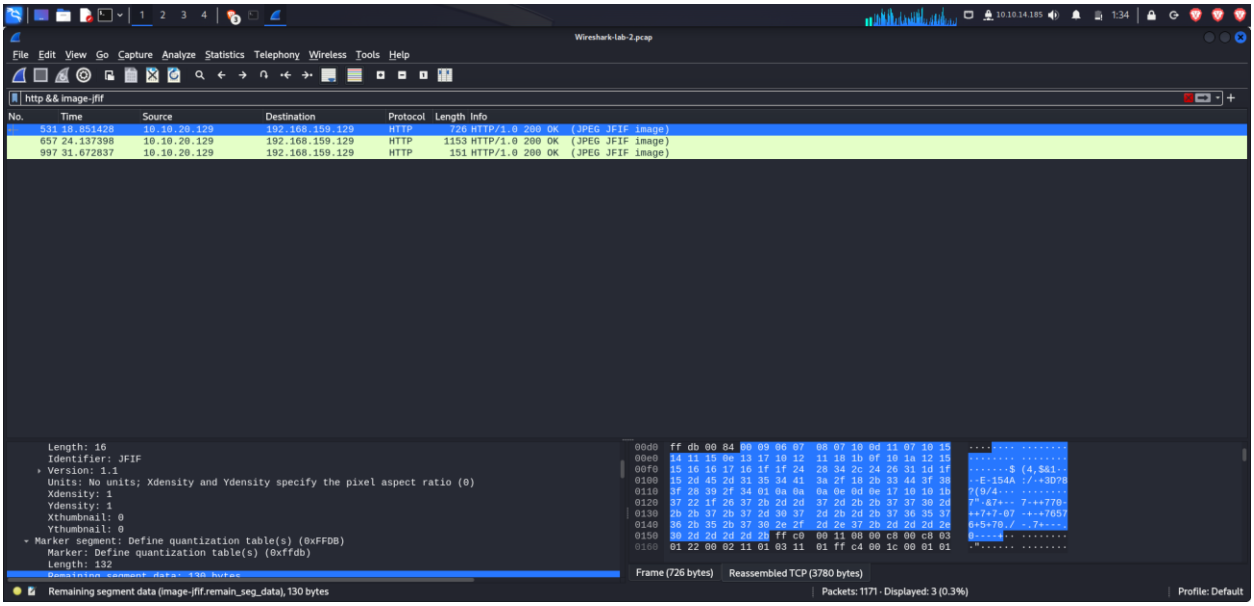
L. Packet Inception, Dissecting Network Traffic With Wireshark

unzip Wireshark-lab-2.zip



i. What was the filename of the image that contained a certain Transformer Leader? (name.filetype)
Rise-Up.jpg

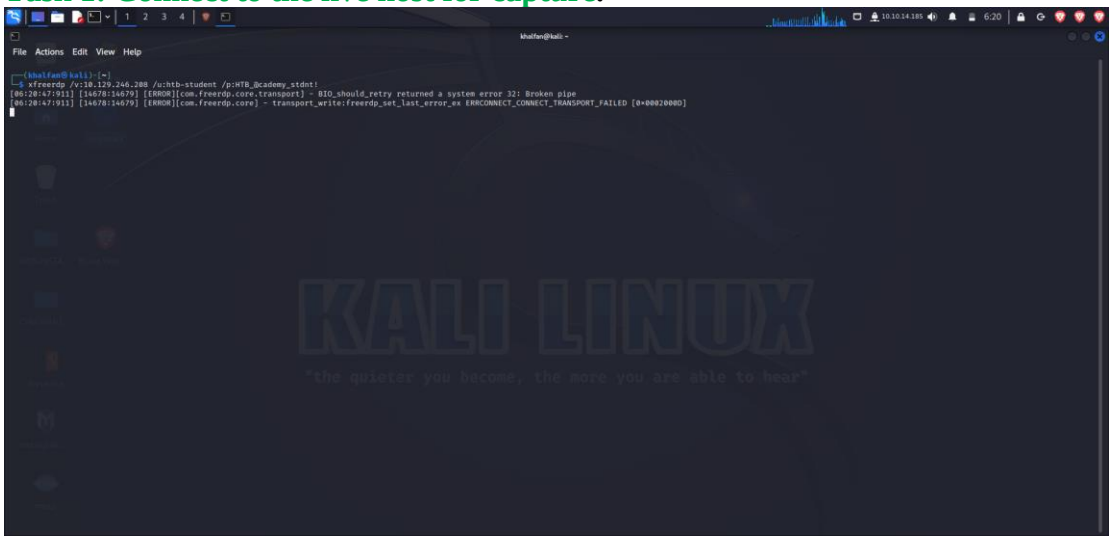
ii. Which employee is suspected of performing potentially malicious actions in the live environment?
Bob



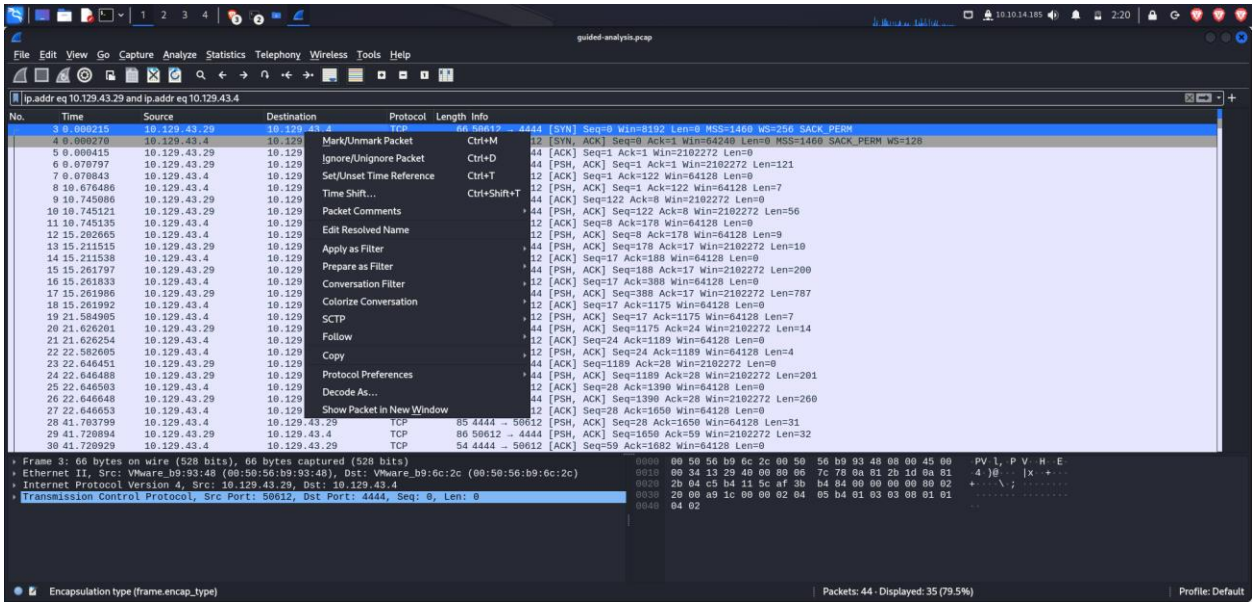
M. Guided Lab: Traffic Analysis Workflow

Tasks

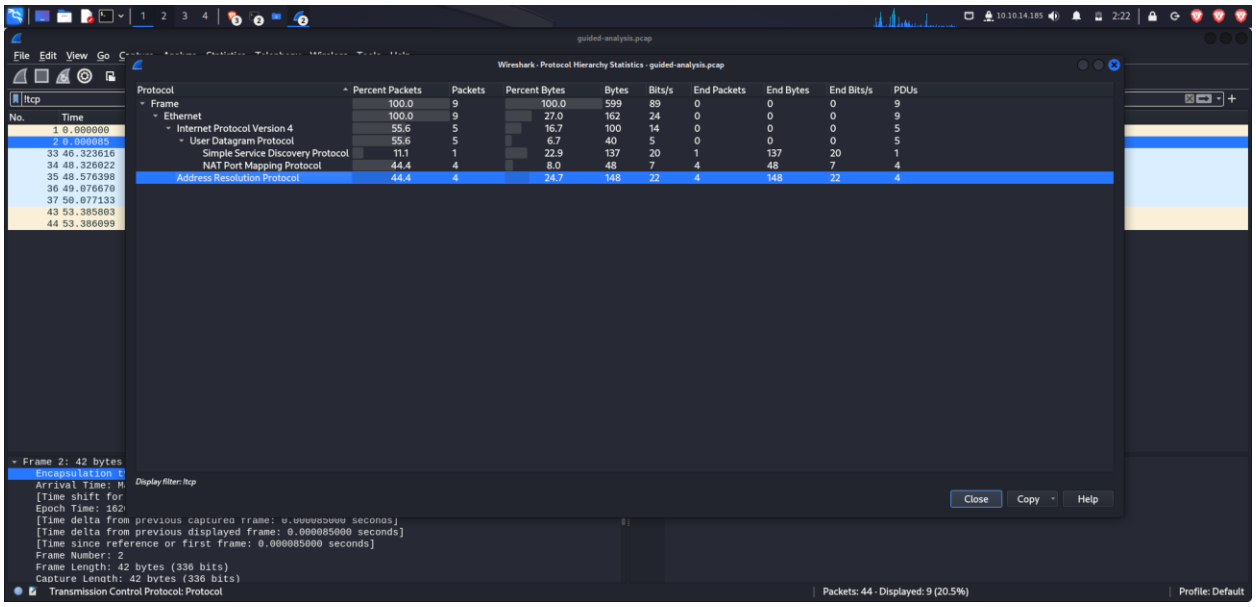
Task 1: Connect to the live host for capture.



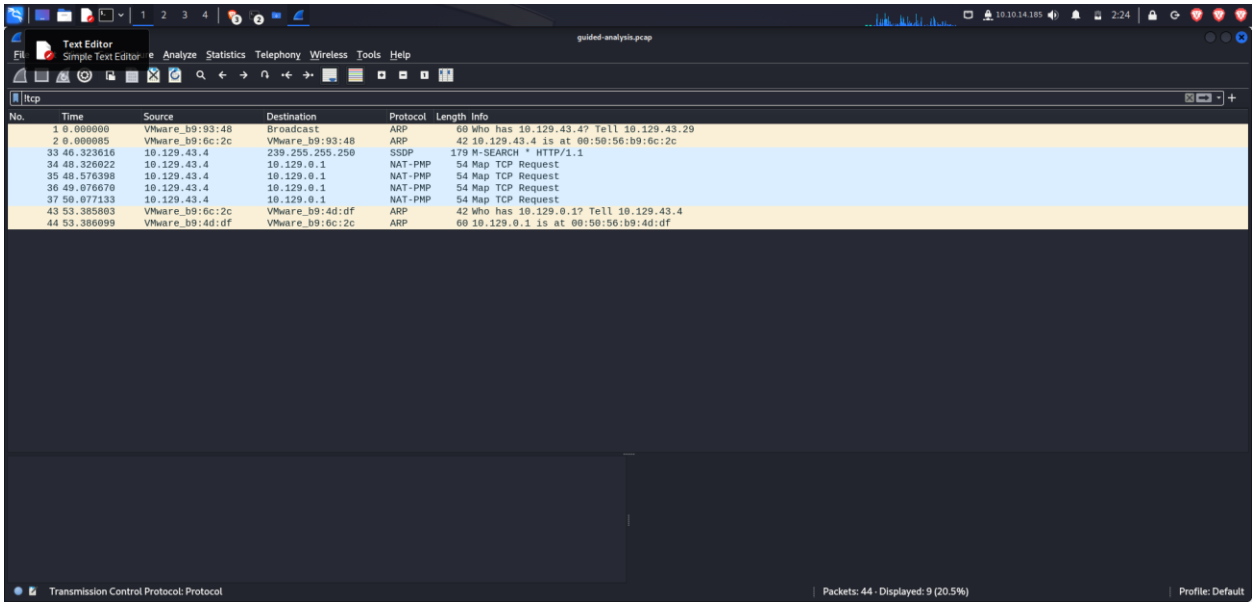
Conversations



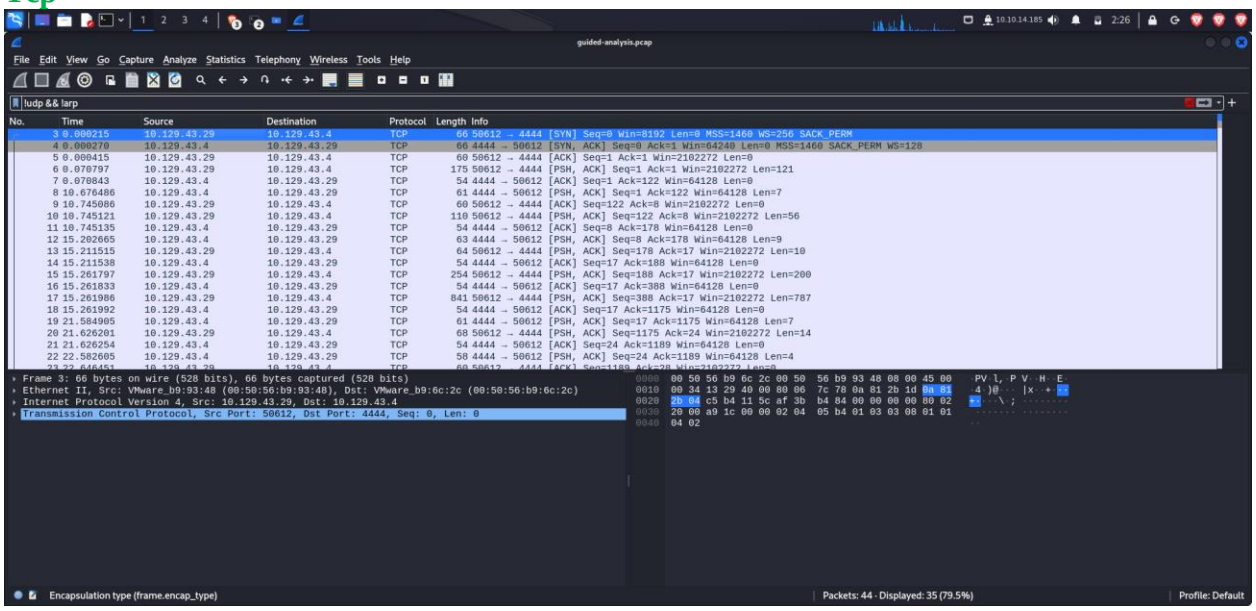
Protocol Statistics



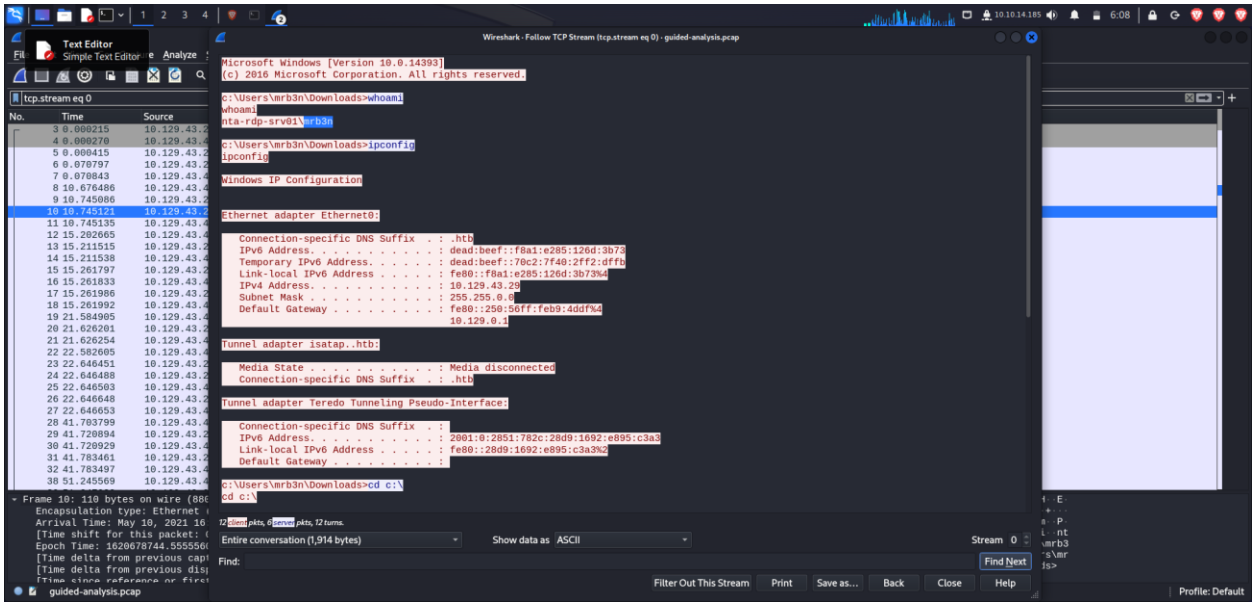
Udp



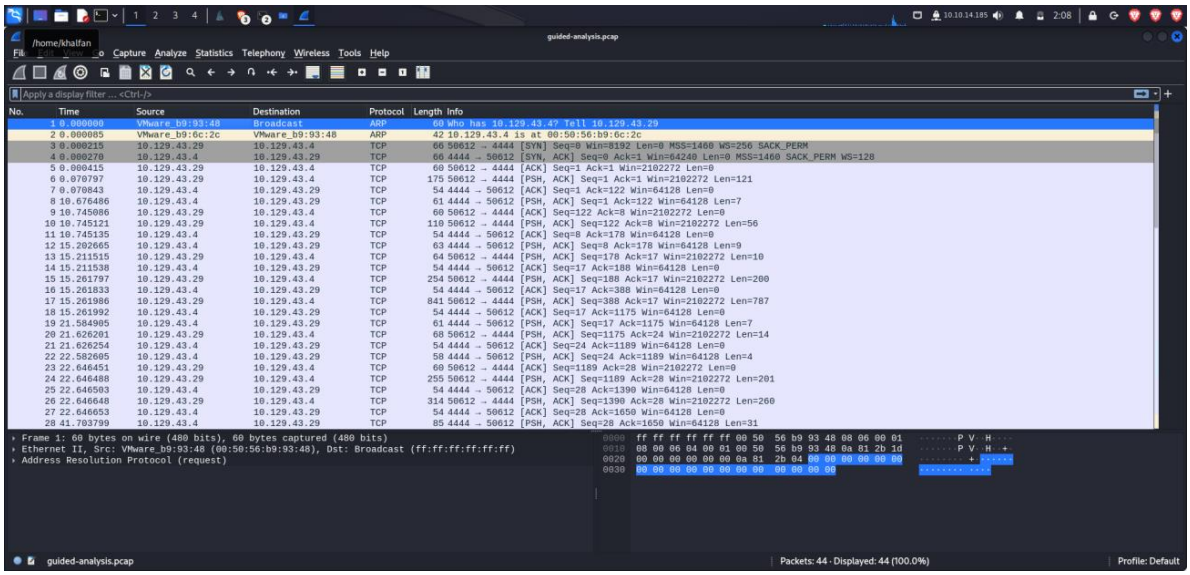
Tcp



Follow tcp stream

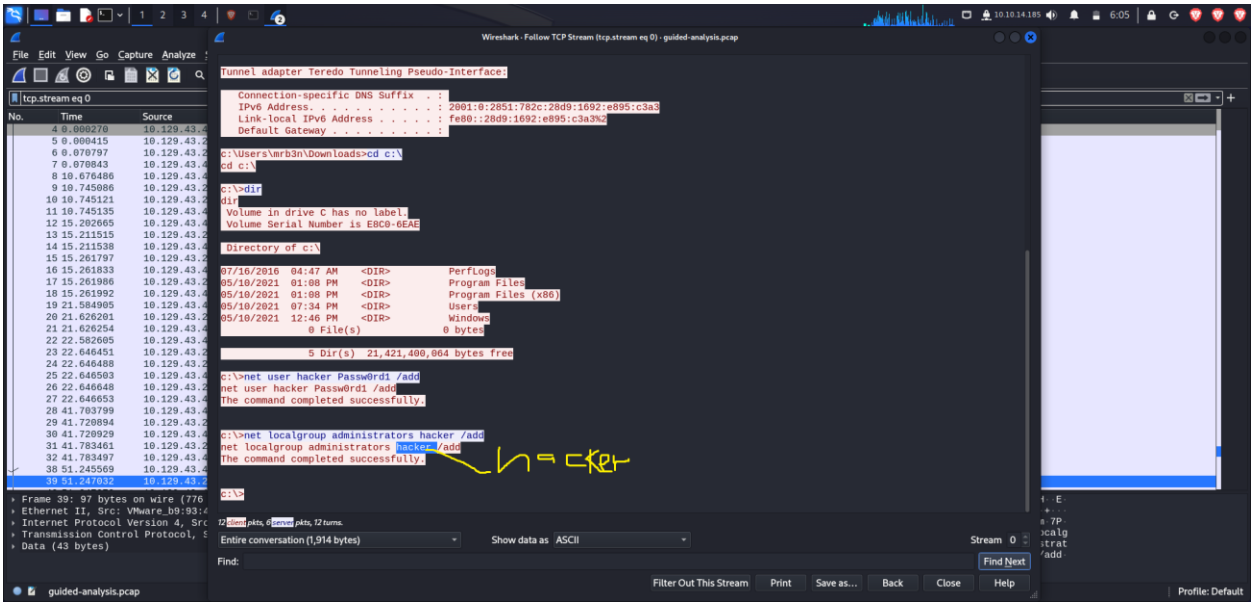


Someone made the account hacker and assigned it to the administrators group on this host

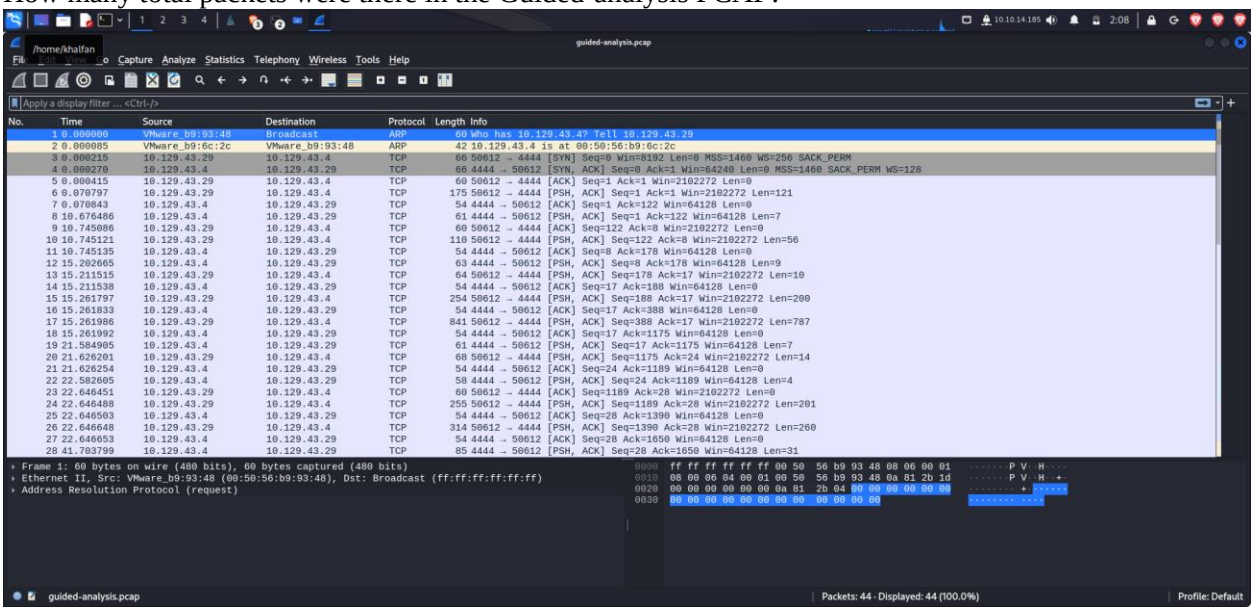


i. What was the name of the new user created on Mr. Ben's host?

Hacker



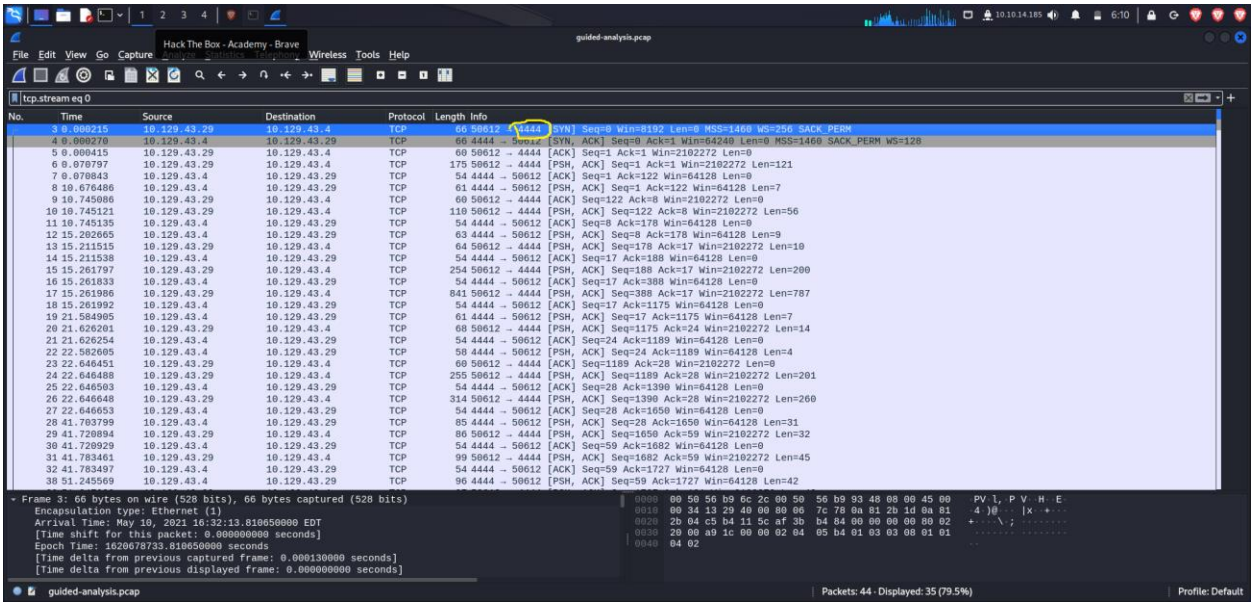
ii. How many total packets were there in the Guided-analysis PCAP?

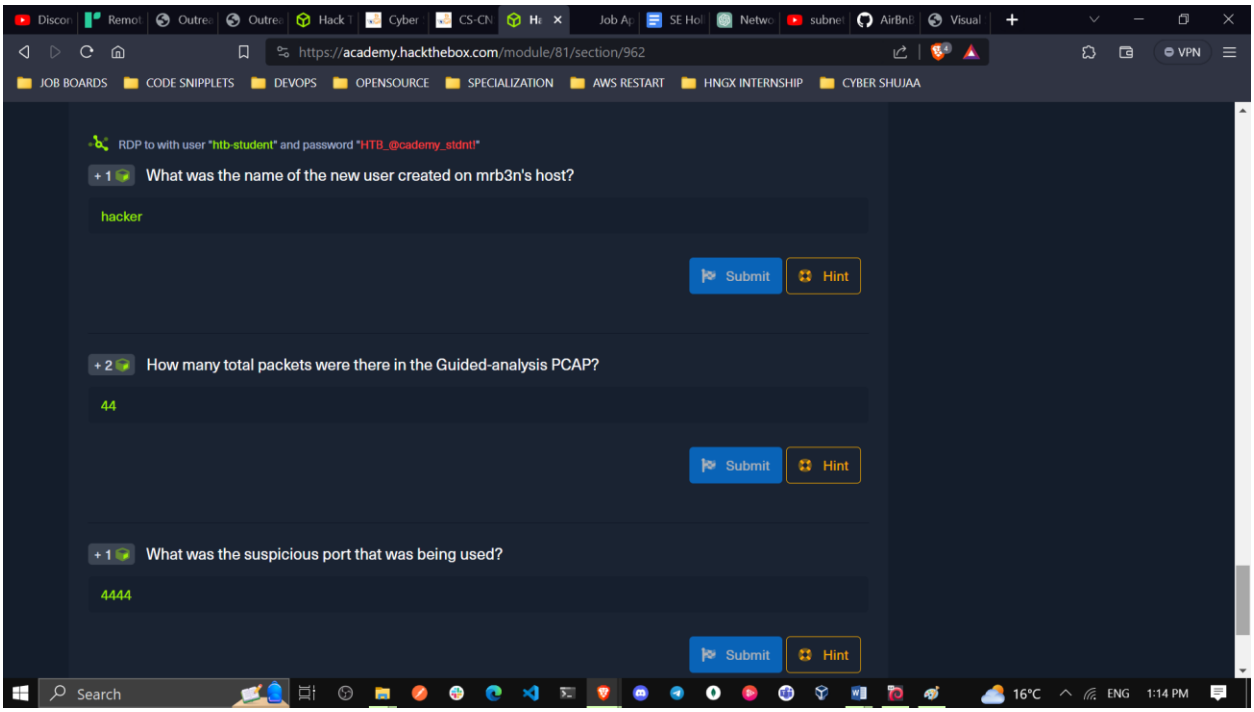


44 located at the lower right side of the screen

iii. What was the suspicious port that was being used?

4444

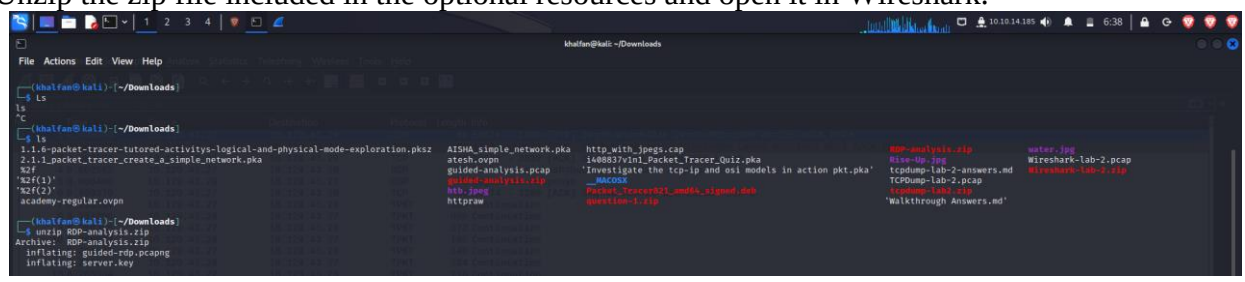


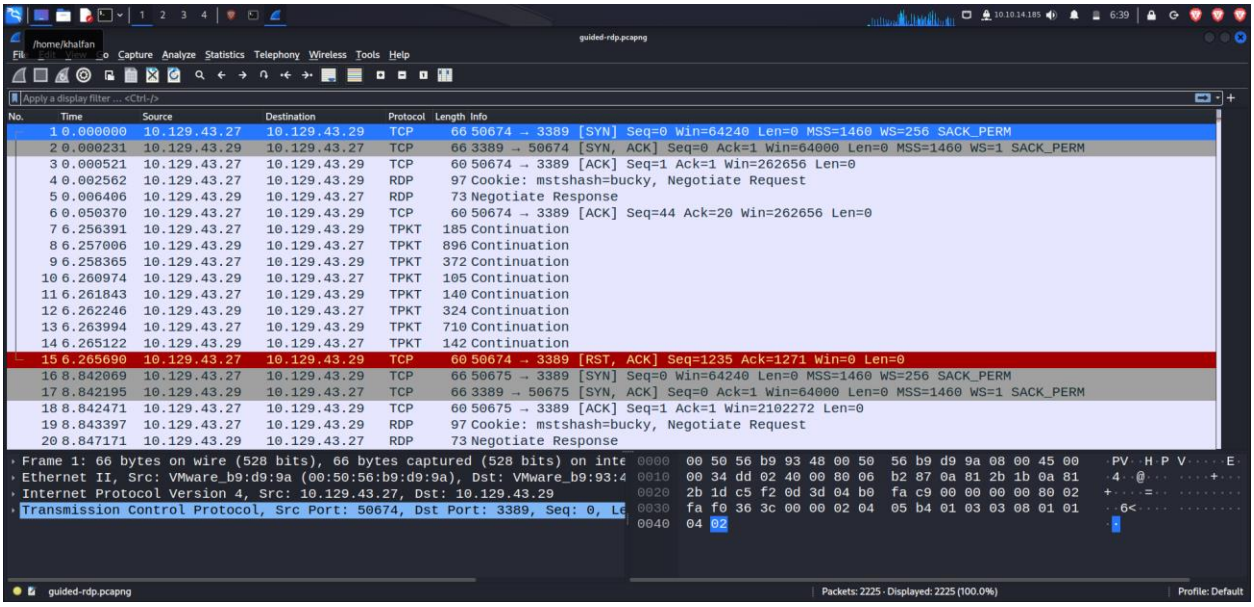


N. Descripting RDP connections

Task #1 Open the rdp.pcapng file in Wireshark.

Unzip the zip file included in the optional resources and open it in Wireshark.

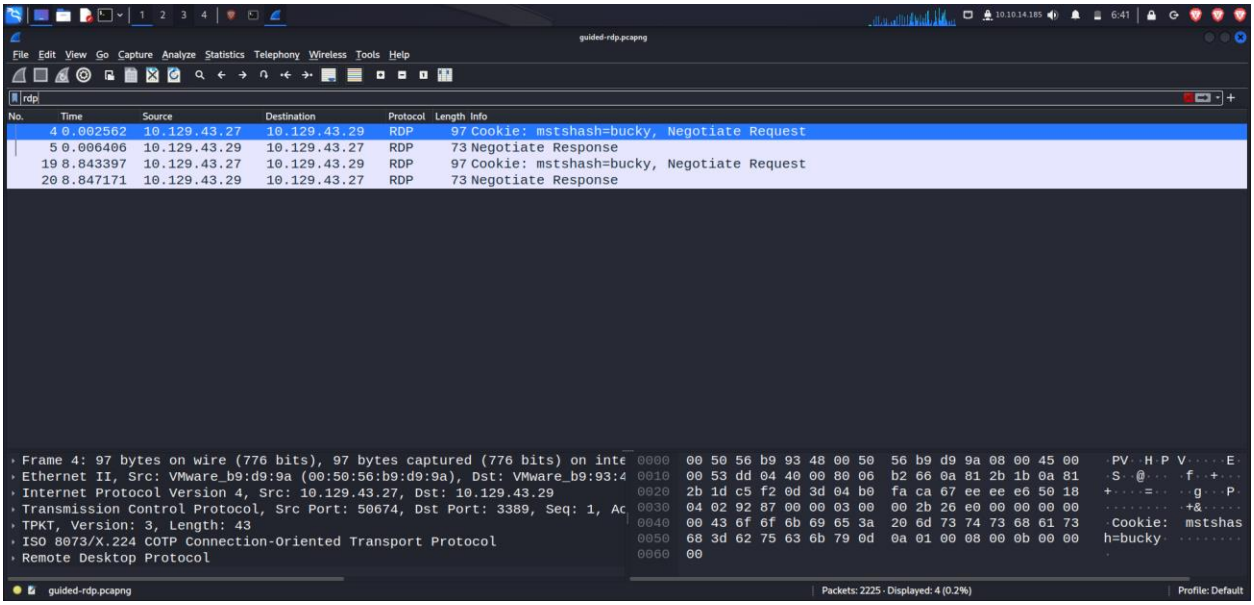




Task #2
Analyze the traffic included.

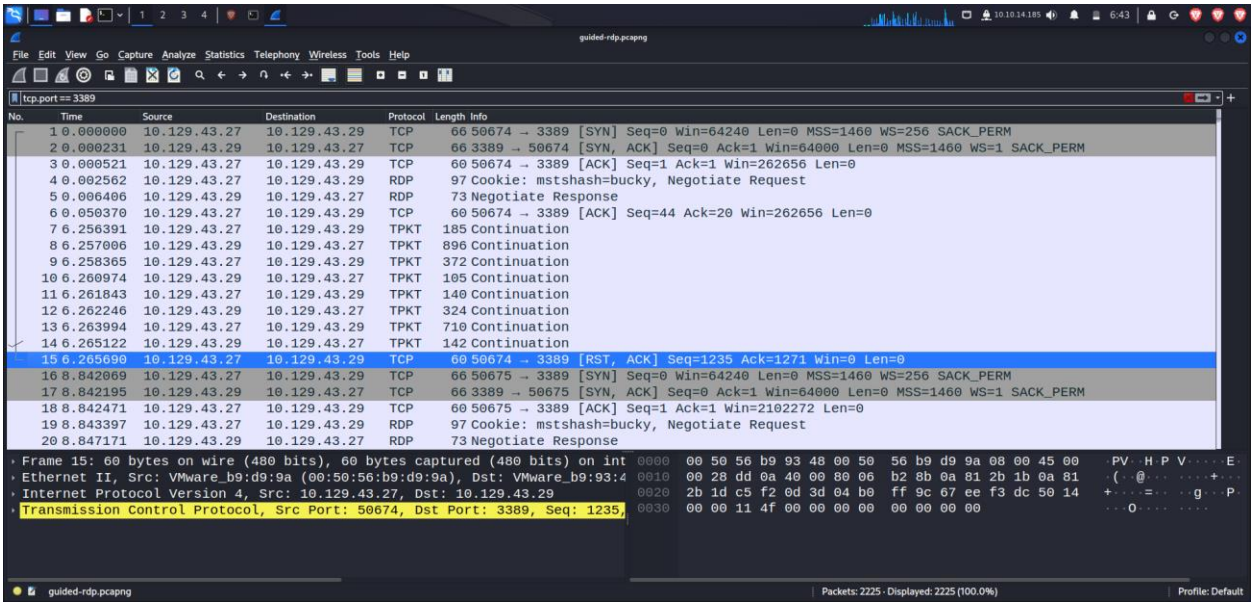
Take a minute to look at the traffic. Notice there is a lot of information here. We know our focus is on RDP, so let's take a second to filter on rdp and see what it returns.

RDP Filter



utilize the display filter tcp.port == 3389.

Filter For TCP Port 3389

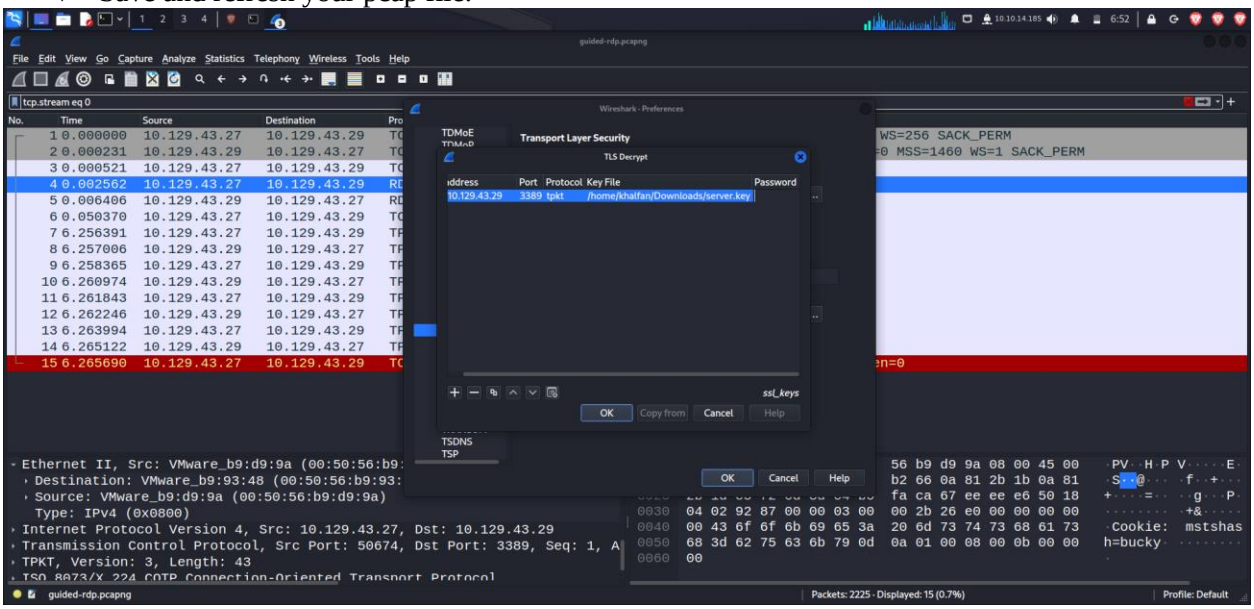


Task #3

Provide the RDP-key to Wireshark so it can decrypt the traffic.

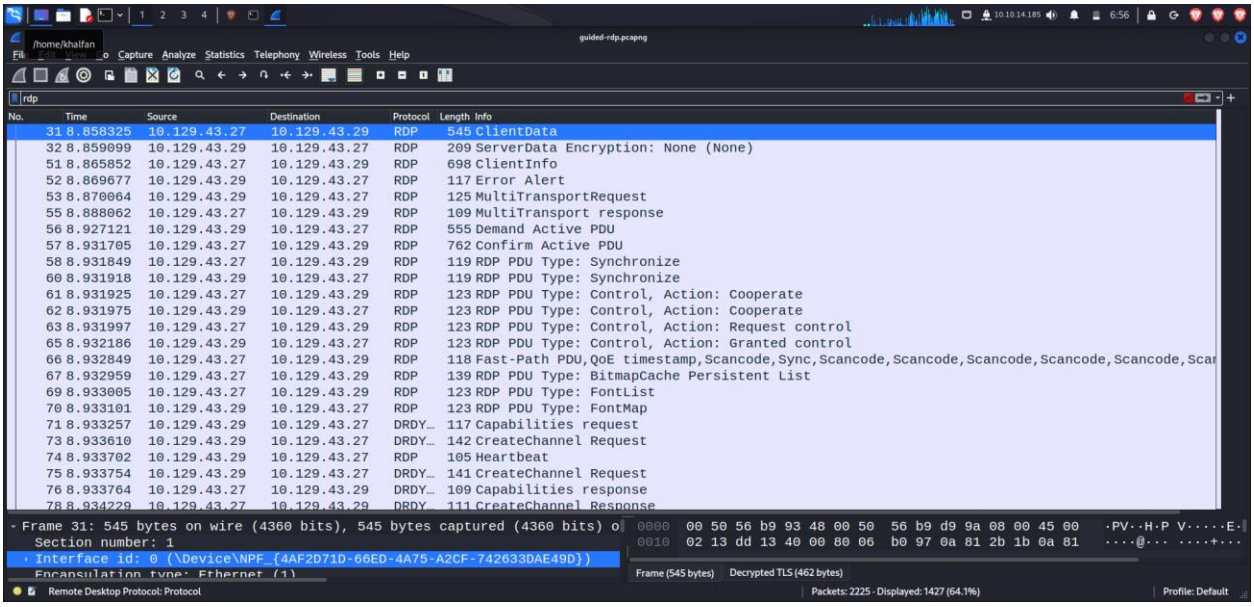
Steps

- Click the + to add a new key
- Type in the IP address of the RDP server 10.129.43.29
- Type in the port used 3389
- Protocol filed equals tpkt or blank.
- Browse to the server.key file and add it in the key file section.
- Save and refresh your pcap file.



When filtering once again on RDP, we should see some traffic in the display.

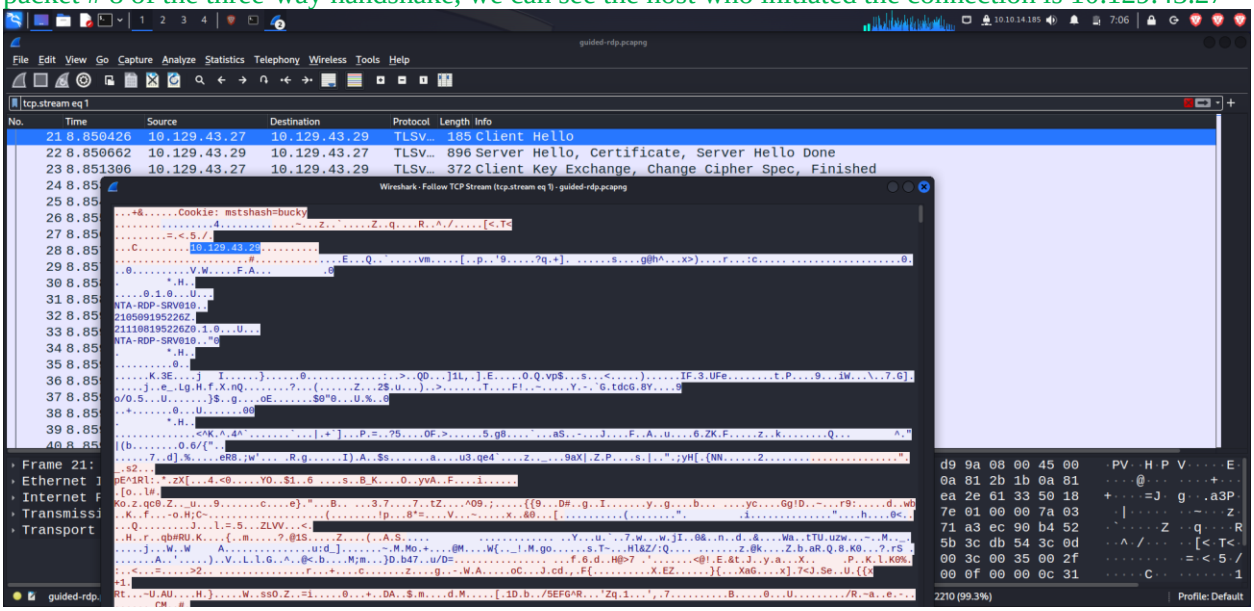
RDP In The Clear



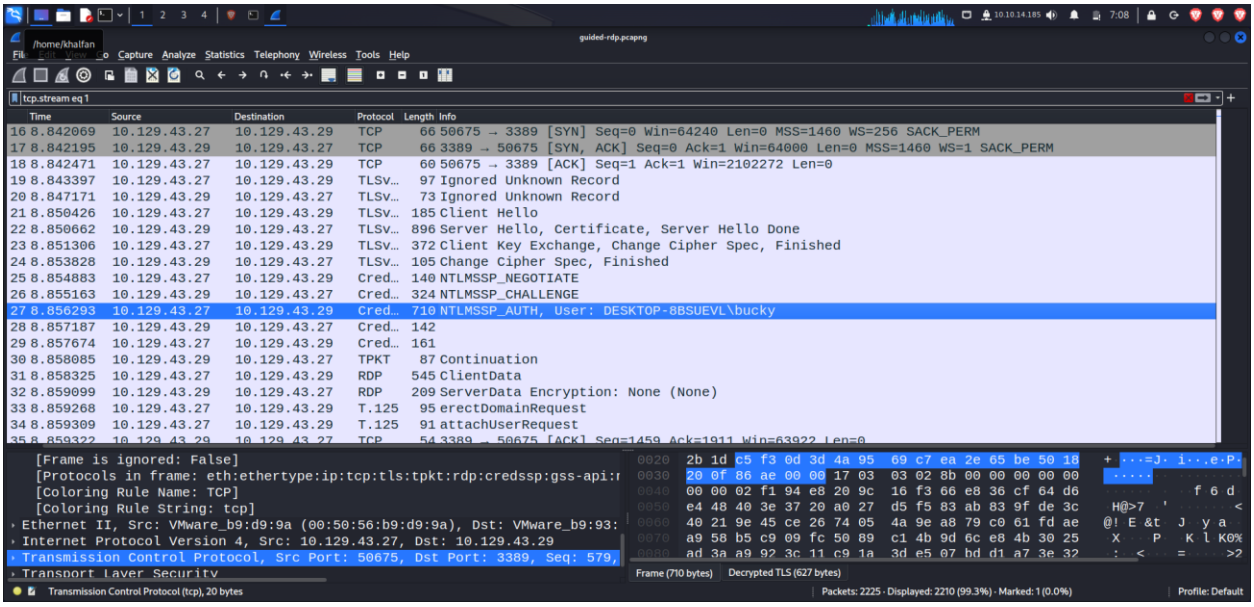
Perform Analysis of the Unencrypted Traffic

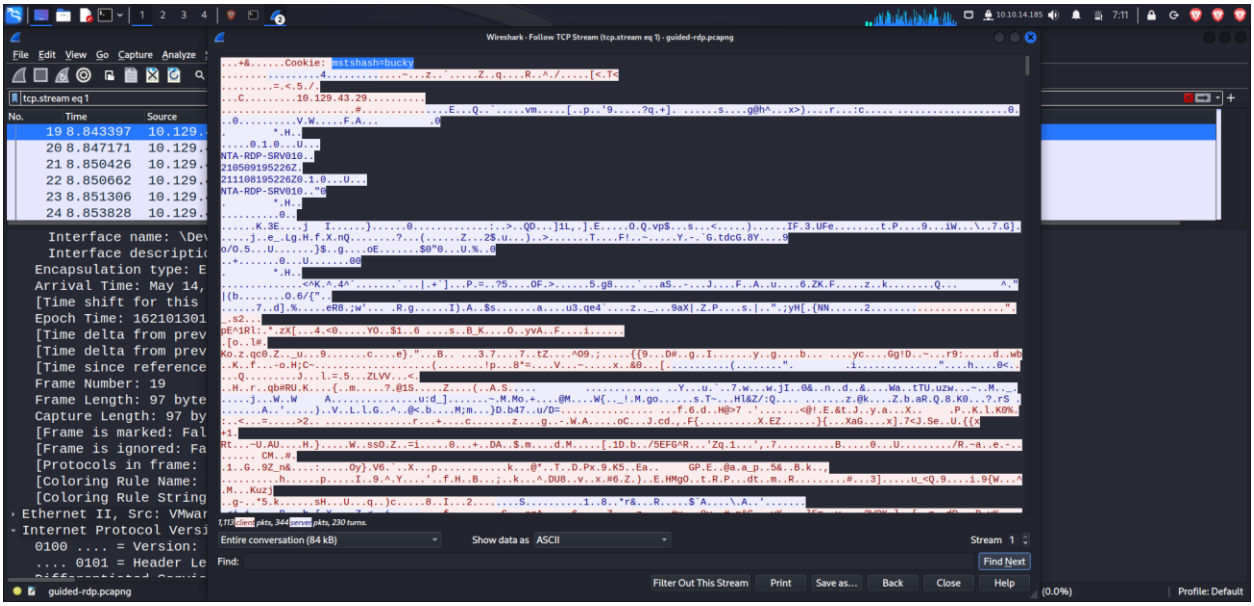
Now that we have broken RDP out of the TLS tunnel, what can we find? Perform the analysis steps

packet # 8 of the three-way handshake, we can see the host who initiated the connection is 10.129.43.27



i. What user account was used to initiate the RDP connection? **Bucky**

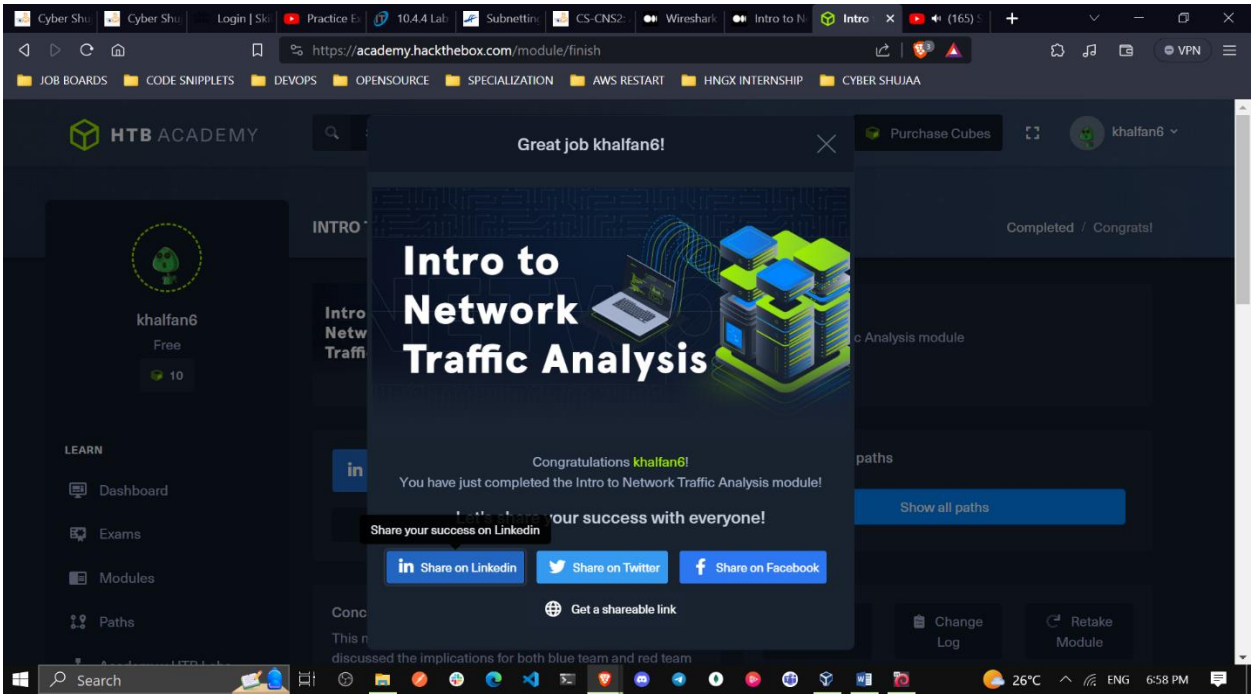




Provide a shareable link : https://academy.hackthebox.com/achievement/785849/81

Conclusion

This lab really impacted on my understanding of the network analysis and interpretation of network packets. Having repeated the lab like three teams helped in identifying all small details I had missed in the first round.


https://academy.hackthebox.com/module/finish
HTB ACADEMY
Purchase Cubes
khalfan6
Great job khalfan6!
Intro to Network Traffic Analysis
Congratulations khalfan6!
You have just completed the Intro to Network Traffic Analysis module!
Share your success on Linkedin
Share on Linkedin
Share on Twitter
Share on Facebook
Get a shareable link
Completed / Congrats!
LEARN
Dashboard
Exams
Modules
Paths
Show all paths
Change Log
Retake Module
Search
26°C
ENG
6:58 PM