

## 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.

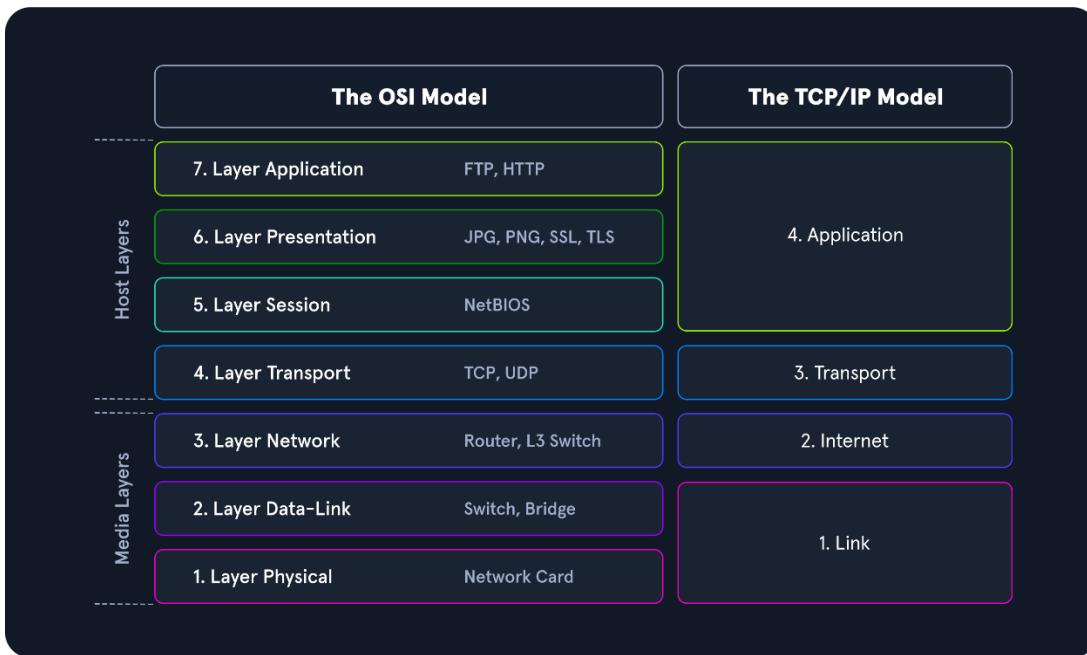
To add to the above, NTA is particularly useful for detecting and responding to attacks that leverage legitimate credentials and tools. This is because NTA can monitor all of the traffic on the network, including traffic that is generated by legitimate users and applications. By monitoring all of the 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

https://academy.hackthebox.com/module/81/section/954

JOB BOARDS CODE SNIPPETS DEVOPS OPENSOURCE SPECIALIZATION AWS RESTART HNGX INTERNSHIP CYBER SHUJAA

Questions

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

+ 0 🎉 How many layers does the OSI model have?

7

Submit Hint

- 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

+ 0 🌟 How many layers are there in the TCP/IP model?

4

Submit Hint

+ 0 🌟 True or False: Routers operate at layer 2 of the OSI model?

False

Submit Hint

+ 0 🌟 What addressing mechanism is used at the Link Layer of the TCP/IP model?

MAC-address

Submit Hint

- 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**

+ 0 🌟 At what layer of the OSI model is a PDU encapsulated into a packet? ( the number )

3

Submit Hint

+ 0 🌟 What addressing mechanism utilizes a 32-bit address?

IPv4

Submit Hint

+ 0 🌟 What Transport layer protocol is connection oriented?

TCP

Submit Hint

- 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**

+ 0 🌟 What Transport Layer protocol is considered unreliable?  
UDP  
Submit Hint

+ 0 🌟 TCP's three-way handshake consists of 3 packets: 1.Syn, 2.Syn & ACK, 3. \_? What is the final packet of the handshake?  
ACK  
Submit Hint

◀ Previous Next ▶ Mark Complete & Next

## 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.

- What is the default operational mode method used by FTP?  
**active**
- FTP utilizes what two ports for command and data transfer? (separate the two numbers with a space)  
**20 and 21**
- Does SMB utilize TCP or UDP as its transport layer protocol?  
**TCP**

Questions

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

+ 0 What is the default operational mode method used by FTP?

active

Submit Hint

- 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**

+ 0 SMB has moved to using what TCP port?

445

Submit Hint

+ 0 Hypertext Transfer Protocol uses what well known TCP port number?

80

Submit Hint

+ 0 What HTTP method is used to request information and content from the webserver?

GET

Submit Hint

- 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
<b>Permission</b>	Passive/ Active	<b>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.</b>
<b>Mirrored Port</b>	Passive	<b>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.</b>

<b>Capture Tool</b>	<b>Passive/ Active</b>	<b>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.</b>
<b>In-line Placement</b>	<b>Active</b>	<b>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.</b>
<b>Network Tap or Host With Multiple NIC's</b>	<b>Active</b>	<b>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</b>
<b>Storage and Processing Power</b>	<b>Passive/ Active</b>	<b>You will need plenty of storage space and processing power for traffic capture off a tap.</b>

## 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

+ 0 Utilizing the output shown in question-1.png, who is the server in this communication? (IP Address)

**174.143.213.184**

**Submit** **question-1.zip** **Hint**

+ 0 Were absolute or relative sequence numbers used during the capture? (see question-1.zip to answer)

**relative**

**Submit** **question-1.zip** **Hint**

+ 0 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.

**Integrated Terminal**

- 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 -nrr HTTP.cap`

**174.143.213.184**

**It is the destination IP address**

```
$ tcpdump -nri HTTP.cap
reading file 'HTTP.cap', link-type EN10MB (Ethernet), snapshot length 65535
15:45:13.266621 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [S], seq 287613953, win 5840, options [mss 1460,sackOK,Ts val 835172936 ecr 2216538,nop,wscale 7], length 0
15:45:13.313726 IP 192.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1460.57678, ack 287613954, 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 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1135, ack 1, win 46, options [nop,nop,Ts val 2216543 ecr 835172936], length 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
15:45:13.361100 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1440.57678, ack 135, win 108, options [nop,nop,Ts val 835172948 ecr 2216543], length 0
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 835172948 ecr 2216543], length 0
15:45:13.363606 IP 192.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1449.57678, ack 135, win 108, options [nop,nop,Ts val 835172948 ecr 2216543], length 0
15:45:13.363610 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 2907, win 91, options [nop,nop,Ts val 2216548 ecr 835172948], length 0
15:45:13.366844 IP 192.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 4345, win 114, options [nop,nop,Ts val 2216548 ecr 835172948], length 0
15:45:13.366844 IP 192.143.213.184.80 > 192.168.1.140.57678: 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 [.], ack 4345, win 108, 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 [.], ack 4345, win 108, options [nop,nop,Ts val 2216548 ecr 835172948], length 0
15:45:13.413003 IP 192.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1440.57678, ack 135, win 108, options [nop,nop,Ts val 2216548 ecr 835172948], length 0
15:45:13.413003 IP 192.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1440.57678, ack 135, win 108, options [nop,nop,Ts val 2216548 ecr 835172948], 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 2216548 ecr 835172948], length 0
15:45:13.414013 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 8689, win 192, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.416301 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 10137, win 204, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
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 835172948], length 0
15:45:13.416424 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 10137:1585, ack 135, win 108, options [nop,nop,Ts val 2216548 ecr 835172948], length 0
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 835172948], length 0
15:45:13.416432 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], seq 11585, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.416432 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], seq 11585, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.416432 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], seq 11585, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.458467 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 130314481, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.458467 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 14481, win 272, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.458467 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], seq 14481.57678, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.461293 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 14481.5929, win 299, options [nop,nop,Ts val 2216558 ecr 835172948], length 0
15:45:13.461302 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 15929, win 318, options [nop,nop,Ts val 2216558 ecr 835172948], 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 2216553 ecr 835172948], length 0
15:45:13.463422 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 17377, ack 318, options [nop,nop,Ts val 2216558 ecr 835172948], 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 2216553 ecr 835172948], length 0
15:45:13.463544 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 18825, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.464163 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 18825, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
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 835172948], length 0
15:45:13.464171 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], seq 20273, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
15:45:13.466749 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 20273:1721, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], length 0
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 835172948], length 0
15:45:13.466773 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 21721:22046, ack 135, win 108, options [nop,nop,Ts val 2216553 ecr 835172948], 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 835172948], length 0
15:45:13.467401 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], seq 135, ack 22046, win 408, options [nop,nop,Ts val 2216558 ecr 835172948], length 0
15:45:13.513631 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 22046, ack 136, win 108, options [nop,nop,Ts val 2216558 ecr 835172948], length 0
15:45:13.513656 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], ack 22047, win 408, options [nop,nop,Ts val 2216558 ecr 835172948], length 0
```

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

**relative**

```
[root@kali ~]# tcpdump -rrr HTTP.cap
reading from file HTTP.cap, link-type EN10MB (Ethernet), snapshot length 65535
15:45:13.313726 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [S], seq 3344080264, ack 2387613954, win 5840, options [mss 1460,sackOK,TS val 835172936 ecr 2216538,nop,wscale 7], length 0
15:45:13.313726 IP 192.168.1.140.57678 > 174.143.213.184.80: Flags [.], 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 [.], seq 1135, 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 2216548], length 0
15:45:13.361089 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1136, ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216548], length 0
15:45:13.361089 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1140, ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216548], length 0
15:45:13.363606 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 14917897, ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216543], length 1448: HTTP
15:45:13.363606 IP 174.143.213.184.80 > 192.168.1.140.57678: 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 289774345, ack 135, win 108, options [nop,nop,TS val 835172948 ecr 2216543], length 1448: HTTP
15:45:13.366844 IP 174.143.213.184.80 > 192.168.1.140.57678: 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 434515793, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.411084 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 5793, win 137, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.413082 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 5793172961, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.413082 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 1441, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 0
15:45:13.414005 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 724118689, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.414013 IP 174.143.213.184.80 > 192.168.1.140.57678: 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 868910137, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.416309 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 10137, win 204, options [nop,nop,TS val 216553 ecr 835172961], length 0
15:45:13.416424 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1013711585, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.416432 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 11585, win 227, options [nop,nop,TS val 2216553 ecr 835172961], length 0
15:45:13.416536 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1158511033, ack 135, win 108, options [nop,nop,TS val 835172961 ecr 2216548], length 1448: HTTP
15:45:13.416536 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 1158511033, win 227, options [nop,nop,TS val 835172961 ecr 2216548], length 0
15:45:13.458467 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1303114481, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.458467 IP 174.143.213.184.80 > 192.168.1.140.57678: 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 [.], seq 1448115929, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.461302 IP 174.143.213.184.80 > 192.168.1.140.57678: 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 1592917377, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.463438 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1737711882, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.463544 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1737711882, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.464165 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1882120340, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 1448: HTTP
15:45:13.464165 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 1882120340, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 0
15:45:13.464171 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 20273121721, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 0
15:45:13.464171 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 20273121721, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 0
15:45:13.466749 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 20273121721, ack 135, win 108, options [nop,nop,TS val 835172973 ecr 2216553], length 0
15:45:13.466757 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 21721122046, ack 135, win 108, options [nop,nop,TS val 835172974 ecr 2216553], length 325: HTTP
15:45:13.466776 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], ack 22046, win 408, options [nop,nop,TS val 2216558 ecr 835172974], length 0
15:45:13.513631 IP 174.143.213.184.80 > 192.168.1.140.57678: Flags [.], seq 22046, ack 136, win 108, options [nop,nop,TS val 835172986 ecr 2216558], length 0
15:45:13.513656 IP 174.143.213.184.80 > 192.168.1.140.57678: 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**

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.

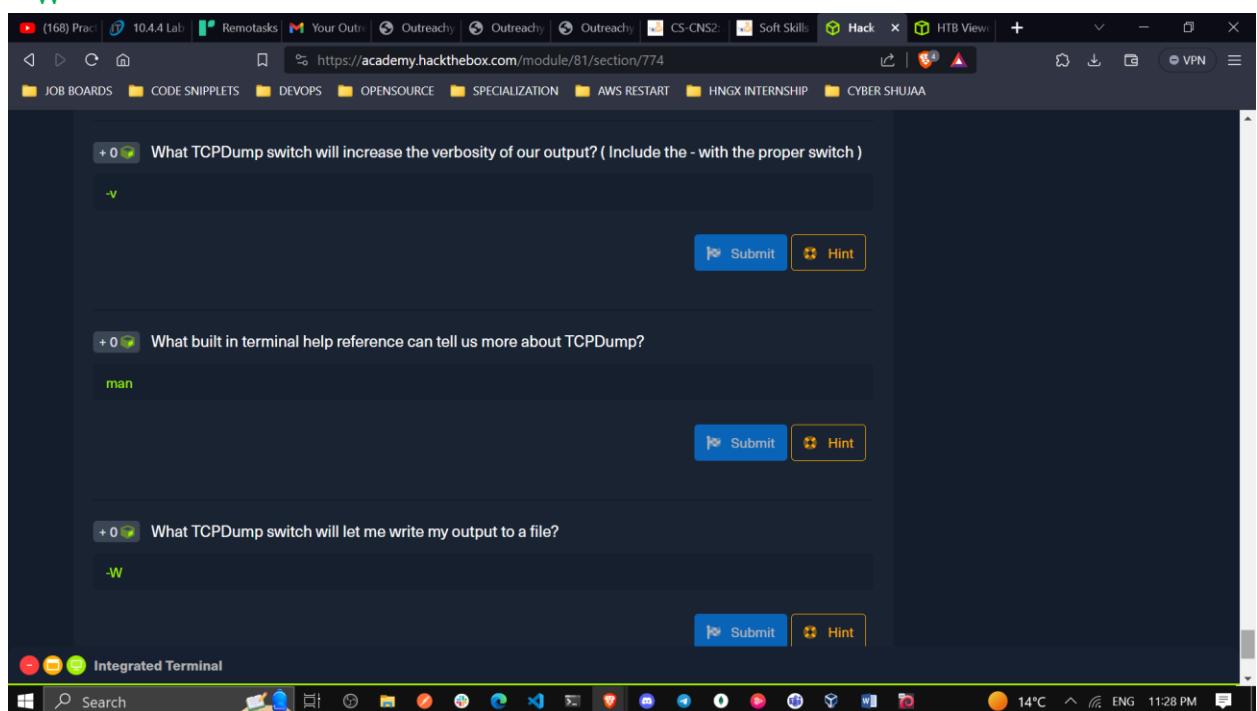
**-nvXc 100**

+ 0 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**

+ 0

- 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.**

```
[eu-academy-2] [10.10.14.130] [htb-ac-785849@htb-ivnqqv8y7z] (~)
└── [★]$ which tcpdump
/usr/bin/tcpdump
[eu-academy-2] [10.10.14.130] [htb-ac-785849@htb-ivnqqv8y7z] (~)
└── [★]$
```

## Task #2: Start a capture

```
[eu-academy-2] [10.10.14.130] [htb-ac-785849@htb-ivnqqv8y7z] (~)
└── [★]$ which tcpdump
/usr/bin/tcpdump
[eu-academy-2] [10.10.14.130] [htb-ac-785849@htb-ivnqqv8y7z] (~)
└── [★]$ tcpdump -D
1.eth0 [Up, Running, Connected]
2.eth1 [Up, Running, Connected]
3.tun0 [Up, Running, Connected]
4.any (Pseudo-device that captures on all interfaces) [Up, Running]
5.lo [Up, Running, Loopback]
6.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
7.nflog (Linux netfilter log (NFLOG) interface) [none]
8.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
9 dbus-system (D-Bus system bus) [none]
10 dbus-session (D-Bus session bus) [none]
[eu-academy-2] [10.10.14.130] [htb-ac-785849@htb-ivnqqv8y7z] (~)
└── [★]$
```

```

05:28:57.876762 IP proxy-uk.hbt-cloud.com.52158 > htbs://academy.hackthebox.com/module/81/section/786: Flags [P.], seq 3552:3584, ack 457506, win 3523, options [nop,nop,TS val 2641639693 ecr 1572845737], length 32: HTTP
05:28:57.877548 IP htbs://ivnqqv8y7r.hbt-cloud.com.http > proxy-uk.hbt-cloud.com.52158: Flags [P.], seq 457506:460170, ack 3584, win 503, options [nop,nop,TS val 1.1572845755 ecr 2641639693], length 2664: HTTP
05:28:57.877799 IP proxy-uk.hbt-cloud.com.52158 > htbs://ivnqqv8y7r.hbt-cloud.com.http: Flags [., ack 460170, win 3523, options [nop,nop,TS val 2641639695 ecr 1572845755]], length 0
05:28:57.87845781 IP htbs://ivnqqv8y7r.hbt-cloud.com.ssh > proxy-uk.hbt-cloud.com.44240: Flags [P.], seq 256192:257988, ack 4789, win 2574, options [nop,nop,TS val 1.1572845781 ecr 2641639617], length 1796
05:28:57.903469 IP htbs://ivnqqv8y7r.hbt-cloud.com.ssh > proxy-uk.hbt-cloud.com.44240: Flags [P.], seq 257988:259888, ack 4789, win 2574, options [nop,nop,TS val 1.1572845781 ecr 2641639617], length 1900
05:28:57.903516 IP htbs://ivnqqv8y7r.hbt-cloud.com.ssh > proxy-uk.hbt-cloud.com.44240: Flags [P.], seq 259888:260132, ack 4789, win 2574, options [nop,nop,TS val 1.1572845781 ecr 2641639617], length 244
05:28:57.903558 IP htbs://ivnqqv8y7r.hbt-cloud.com.ssh > proxy-uk.hbt-cloud.com.44240: Flags [P.], seq 260132:260552, ack 4789, win 2574, options [nop,nop,TS val 1.1572845781 ecr 2641639617], length 420
05:28:57.903594 IP htbs://ivnqqv8y7r.hbt-cloud.com.ssh > proxy-uk.hbt-cloud.com.44240: Flags [P.], seq 260552:260972, ack 4789, win 2574, options [nop,nop,TS val 1.1572845781 ecr 2641639617], length 420
05:28:57.903890 IP proxy-uk.hbt-cloud.com.44240 > htbs://ivnqqv8y7r.hbt-cloud.com.ssh: Flags [., ack 257988, win 10223, options [nop,nop,TS val 2641639721 ecr 1572845781]], length 0
05:28:57.903890 IP proxy-uk.hbt-cloud.com.44240 > htbs://ivnqqv8y7r.hbt-cloud.com.ssh: Flags [P.], seq 4789:4825, ack 257988, win 10223, options [nop,nop,TS val 2641639721 ecr 1572845781]], length 36
05:28:57.903934 IP proxy-uk.hbt-cloud.com.44240 > htbs://ivnqqv8y7r.hbt-cloud.com.ssh: Flags [., ack 259888, win 10223, options [nop,nop,TS val 2641639721 ecr 1572845781]], length 0
05:28:57.903944 IP proxy-uk.hbt-cloud.com.44240 > htbs://ivnqqv8y7r.hbt-cloud.com.ssh: Flags [P.], seq 4825:4861, ack 260972, win 10223, options [nop,nop,TS val 2641639721 ecr 1572845781]], length 36
05:28:57.903955 IP htbs://ivnqqv8y7r.hbt-cloud.com.ssh > proxy-uk.hbt-cloud.com.44240: Flags [., ack 4861, win 2574, options [nop,nop,TS val 1572845781 ecr 2641639721]], length 0
^C
1235 packets captured
1272 packets received by filter
0 packets dropped by kernel
[eu-academy-2]-[10.10.14.130]-[htb-ac-785849@htb-ivnqqv8y7r]-[~]
[★]$
```

### Task #3: Utilize Basic Capture Filters

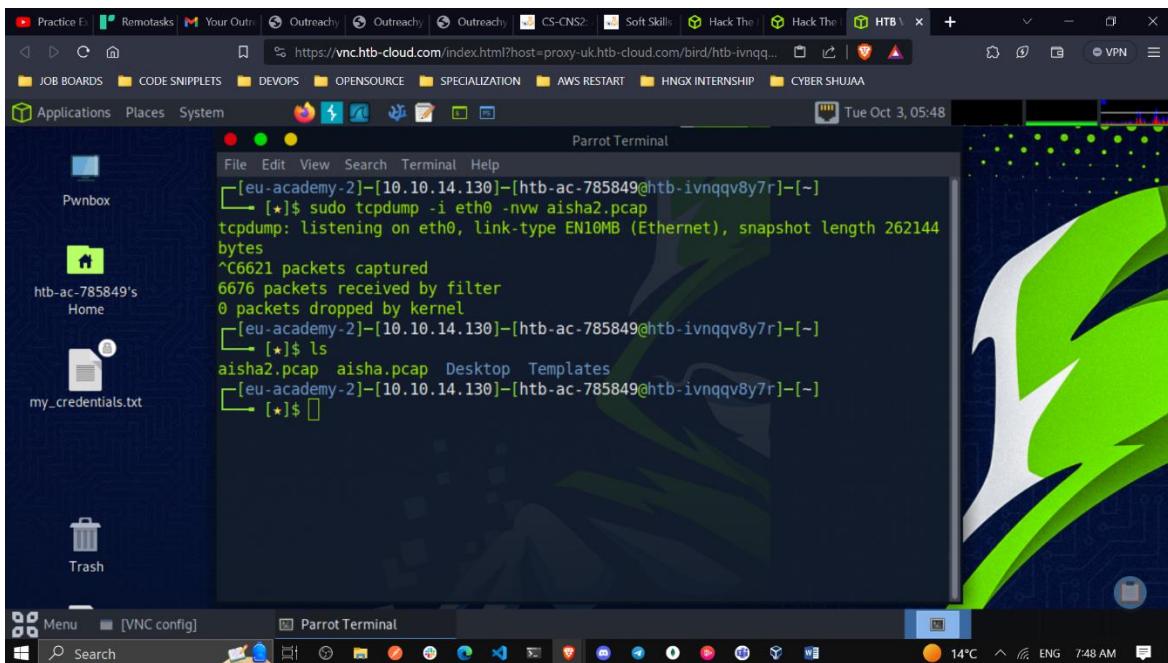
`sudo tcpdump -i eth0 -vx`

```

[eu-academy-2]-[10.10.14.130]-[htb-ac-785849@htb-ivnqqv8y7r]-[~]
[★]$ sudo tcpdump -i eth0 -vx
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
05:43:26.143184 IP (tos 0x0, ttl 64, id 26882, offset 0, flags [DF], proto TCP (6), length 418)
    htbs://ivnqqv8y7r.hbt-cloud.com.http > proxy-uk.hbt-cloud.com.52776: Flags [P.], cksum 0xe9ee (incorrect -> 0x8a66), seq 1764946893:1764947259, ack 2565730816, win 543, options [nop,nop,TS val 1573714021 ecr 2642507842], length 366: HTTP
        0x0000: 4500 01a2 6902 4000 4006 e7f9 867a 6c7f
        0x0010: b23e 4322 0050 ce28 6932 f3cd 98ed f200
        0x0020: 8018 021f e9ee 0000 0101 080a 5dcc f865
        0x0030: 9d81 7842 827e 016a f800 0000 8000 0001
        0x0040: 0101 0000 ffff 0238 006c 0010 0016 0000
        0x0050: 0007 0001 2912 1b2c 131c 2c13 lc2d 121c
        0x0060: 2c13 1b2c 121b 2b13 1d2d 131b 293f a405
        0x0070: 9fe6 094e 3a2c 1e63 1093 ef02 776e 2513
        0x0080: 2f1d 74e0 0093 aala 1e1b 2b7b c103 9fd4
        0x0090: 0e3d 2b2b 141d 2e9f ef04 6a59 299f ef00
        0x00a0: 8b95 1f18 1b2c 437f 0c9a ef04 6958 2713
        0x00b0: 1d27 4db8 039f db0d 402e 2b14 341d 78e3
```

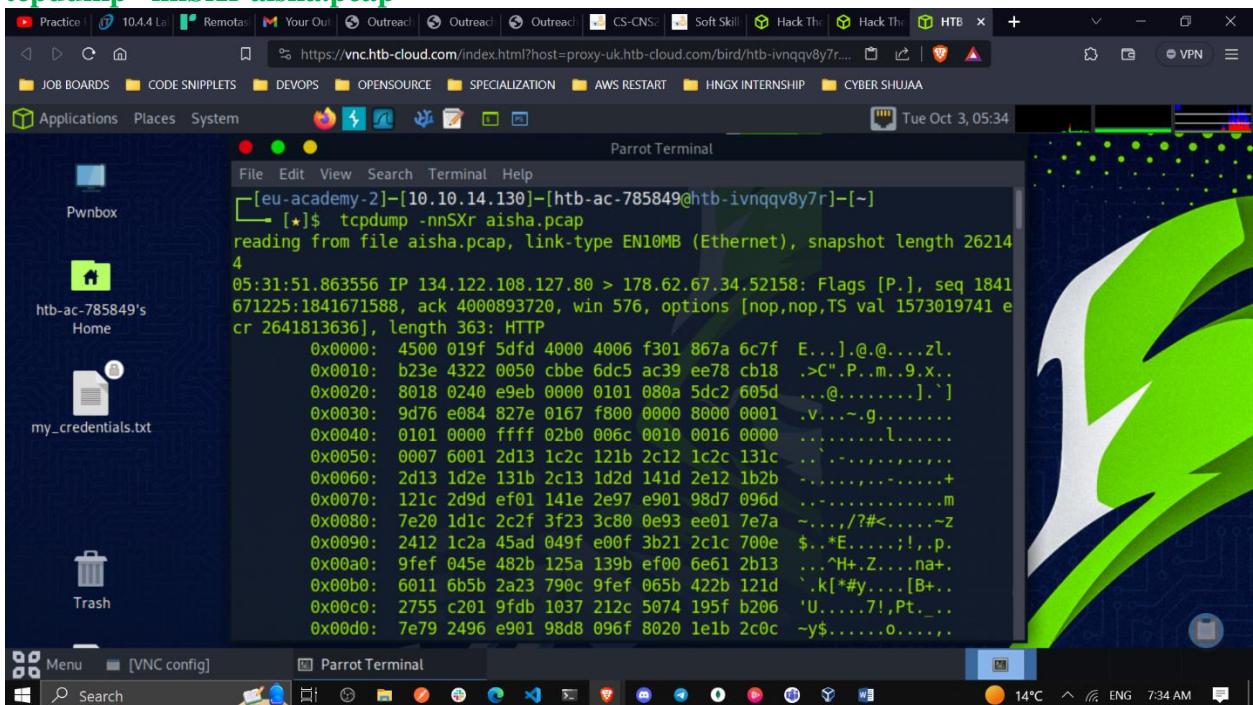
### Task #4: Save a Capture to a .PCAP file.

`sudo tcpdump -i eth0 -nvw aisha.pcap`

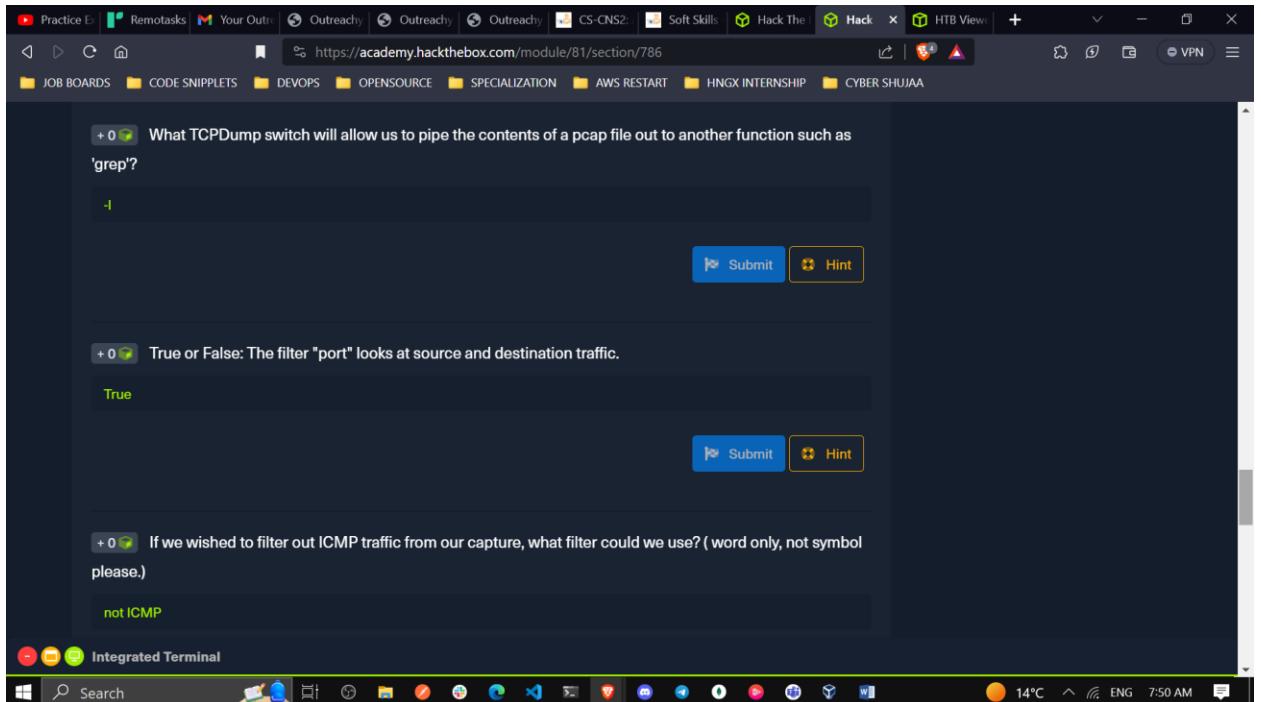


### Task #5: Read the Capture from a .PCAP file.

`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**

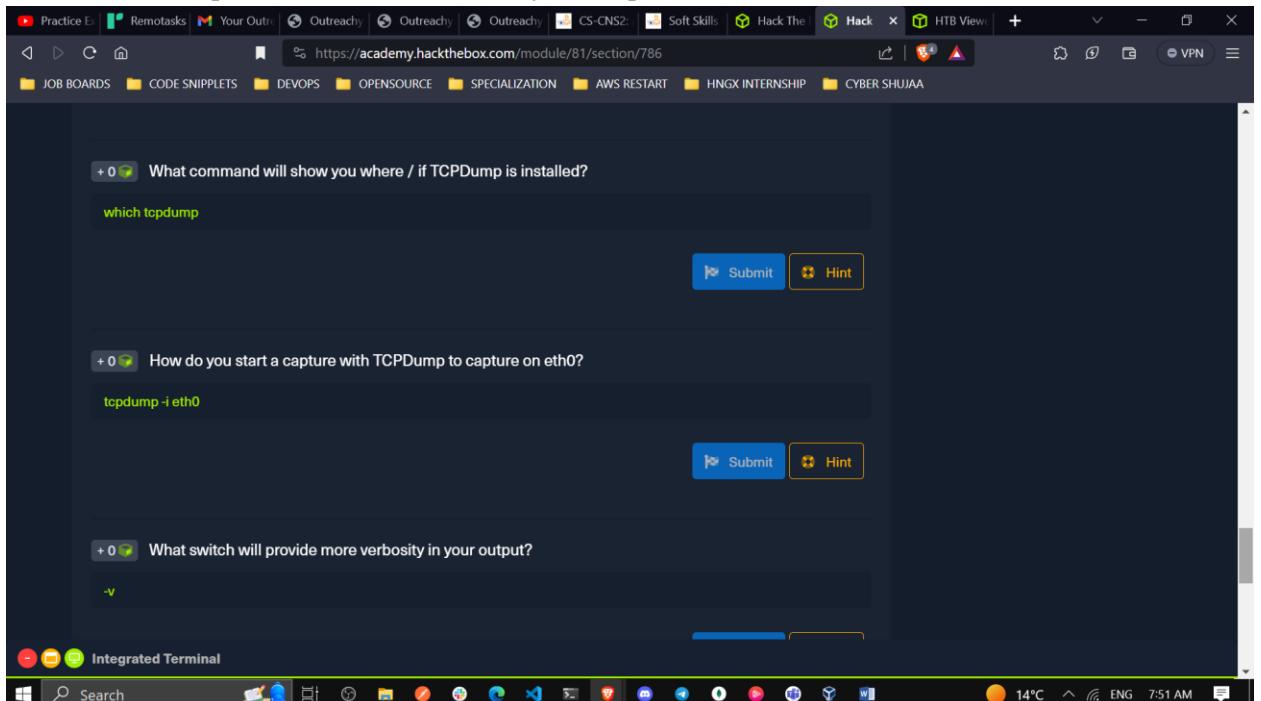


+ 0 What TCPDump switch will allow us to pipe the contents of a pcap file out to another function such as 'grep'?  
-|

+ 0 True or False: The filter "port" looks at source and destination traffic.  
True

+ 0 If we wished to filter out ICMP traffic from our 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**

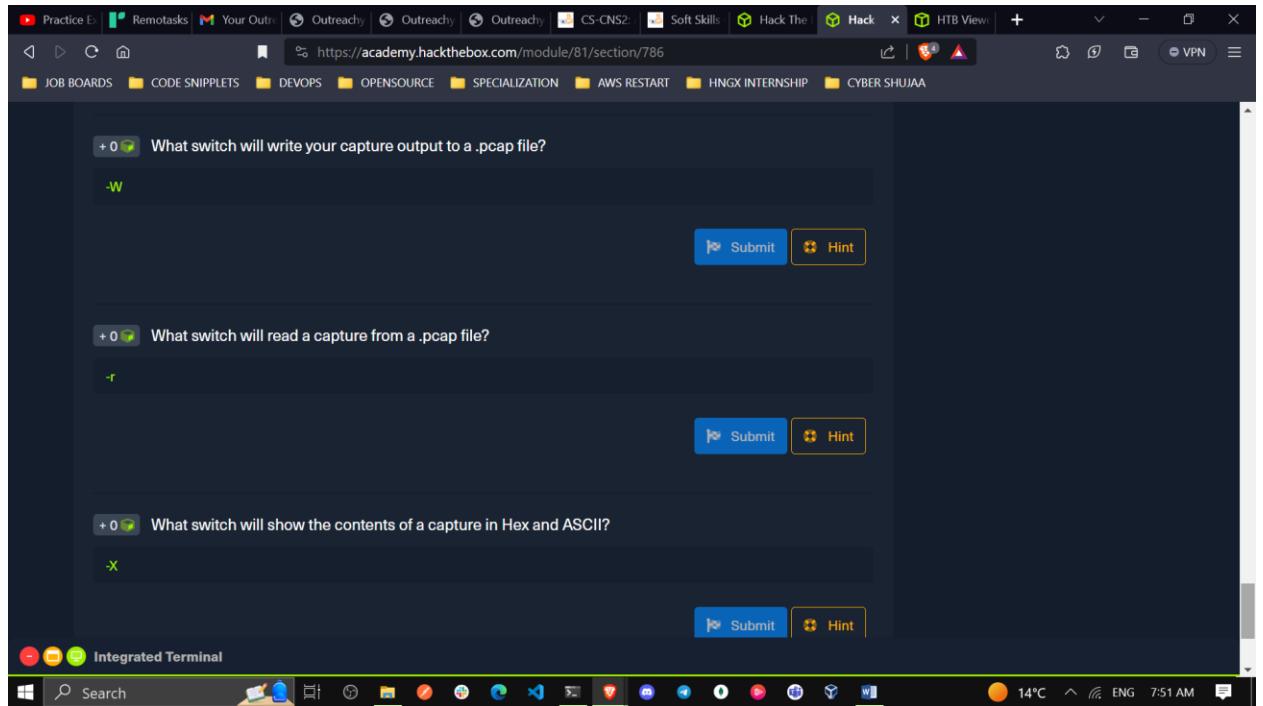


+ 0 What command will show you where / if TCPDump is installed?  
which tcpdump

+ 0 How do you start a capture with TCPDump to capture on eth0?  
tcpdump -i eth0

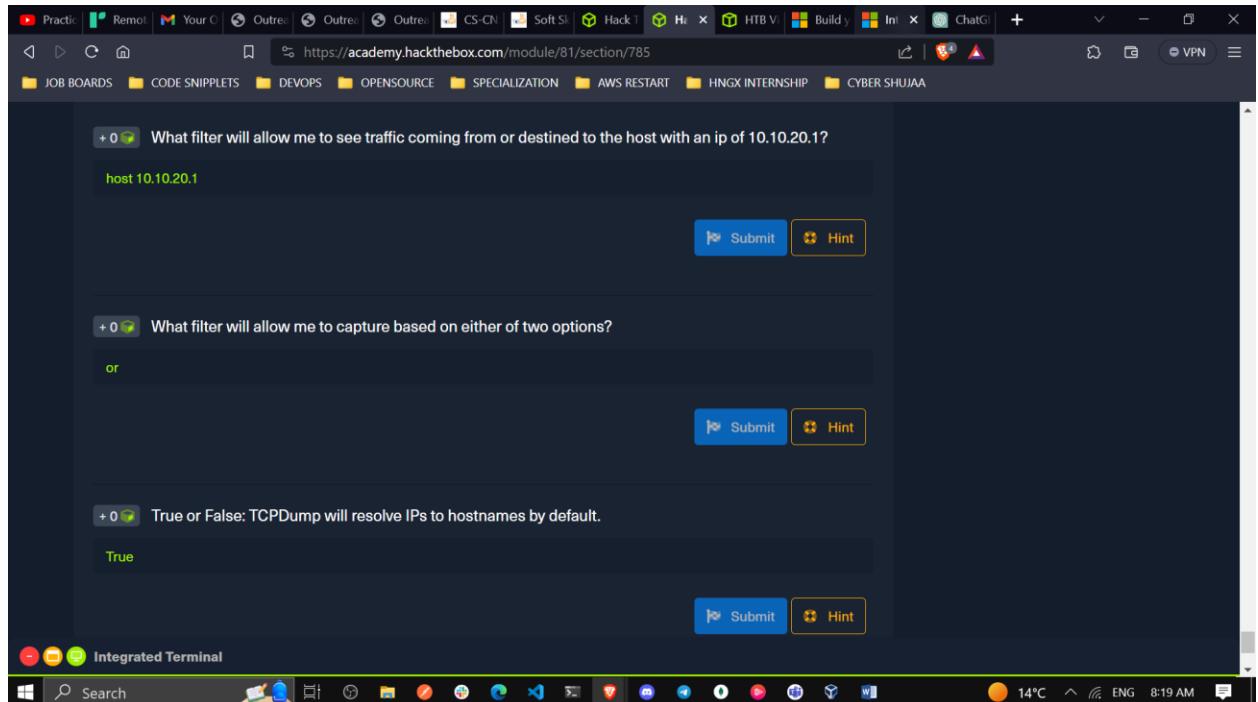
+ 0 What switch will provide more verbosity in your output?  
-v

- 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

Questions

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

+ 1 🎁 What are the client and server port numbers used in first full TCP three-way handshake? (low number first then high number)

80 43806

+ 1 🎁 Based on the traffic seen in the pcap file, who is the DNS server in this network segment? (ip address)

172.16.146.1

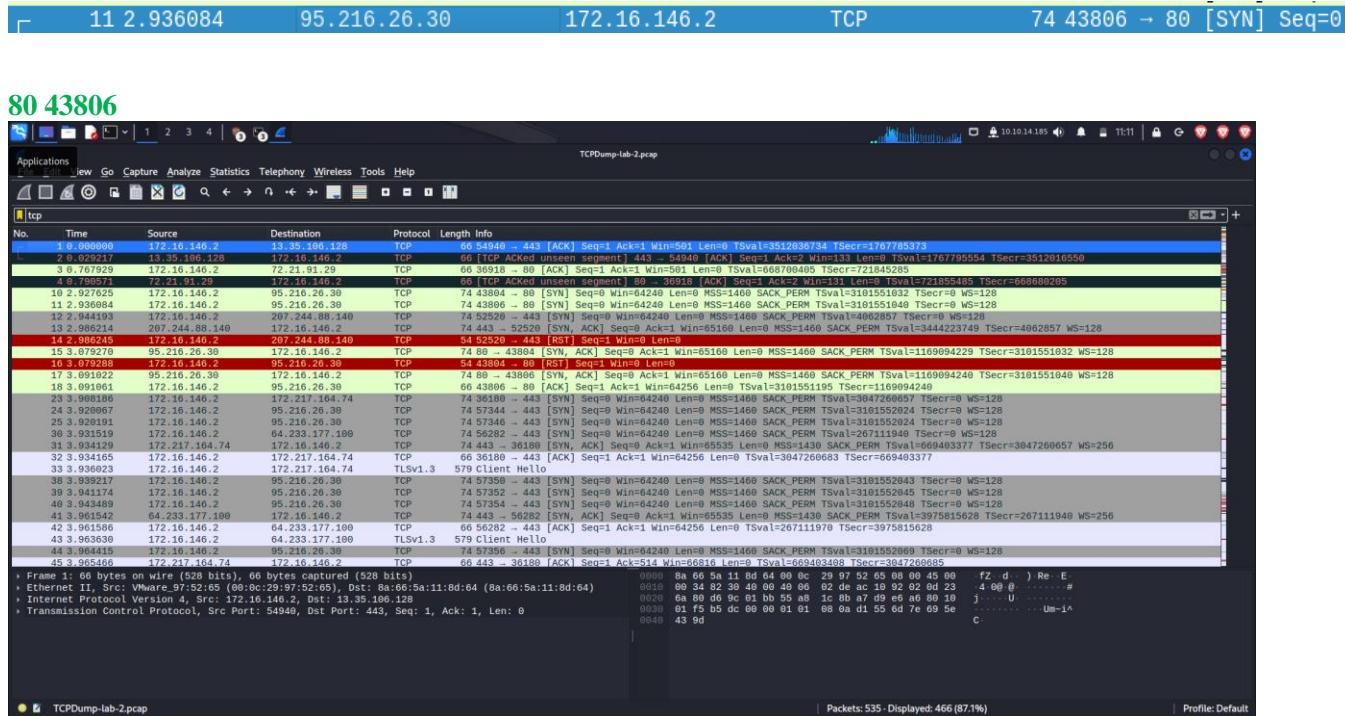
Integrated Terminal

- 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
```



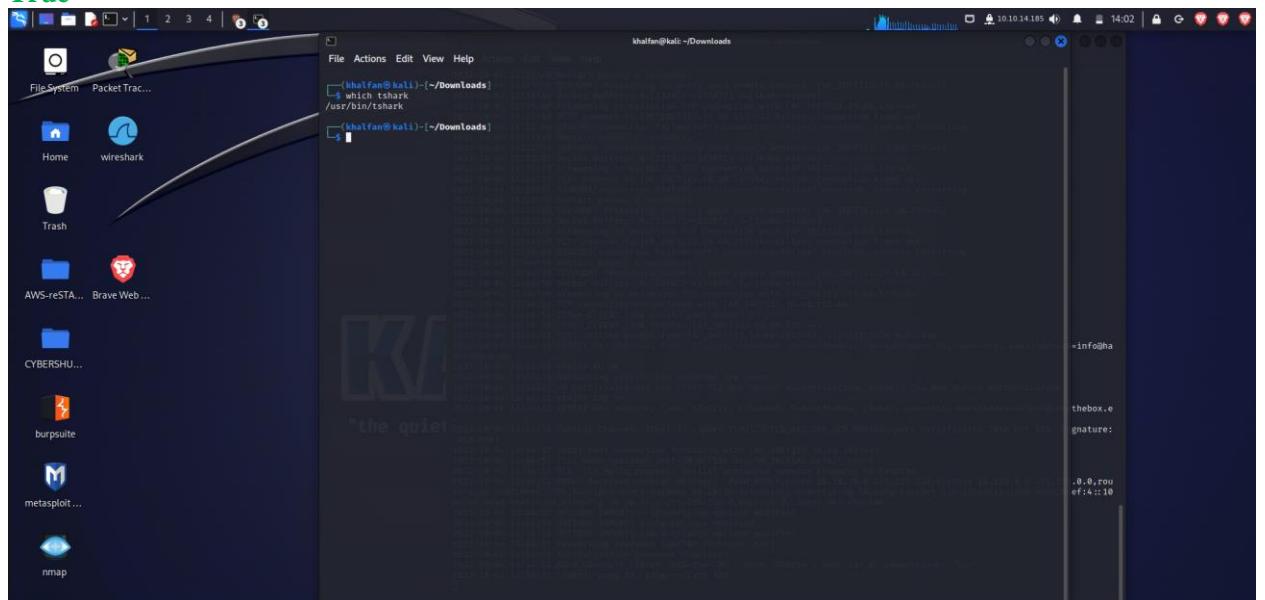
- ii. Based on the traffic seen in the pcap file, who is the DNS server in this network segment? (ip address) Download: [zeek zeek -C -r TCPDump-lab-2.pcap](#) then [cat dns.log](#)

**172.16.146.2**

## I. Analysis with Wireshark

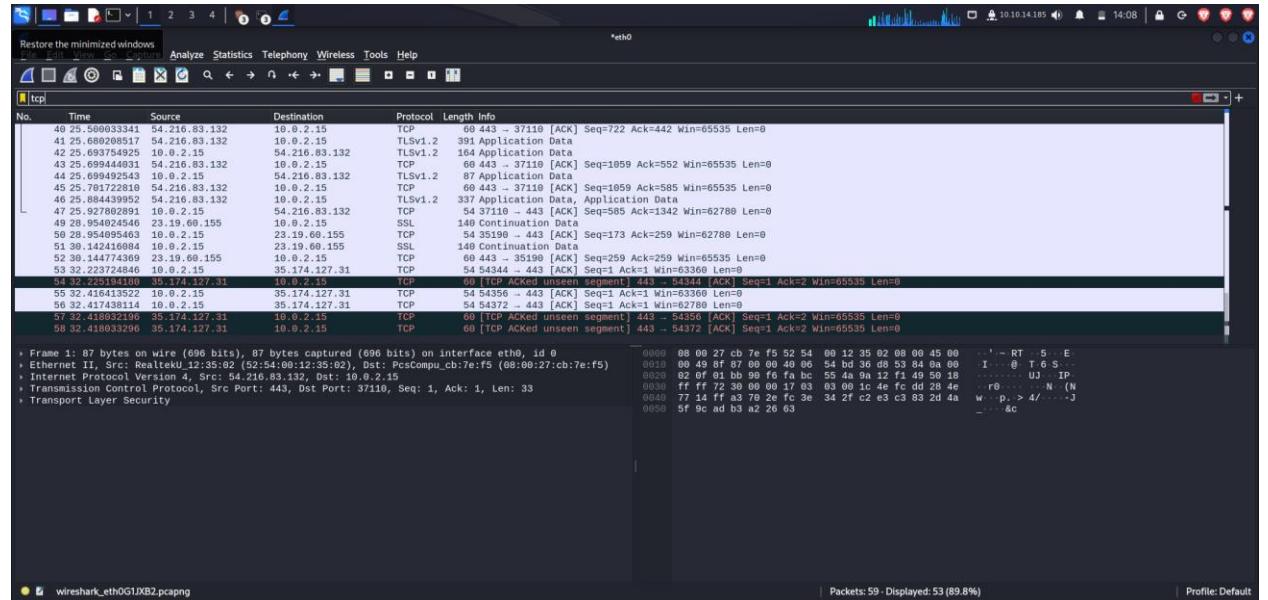
- i. True or False: Wireshark can run on both Windows and Linux.

**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

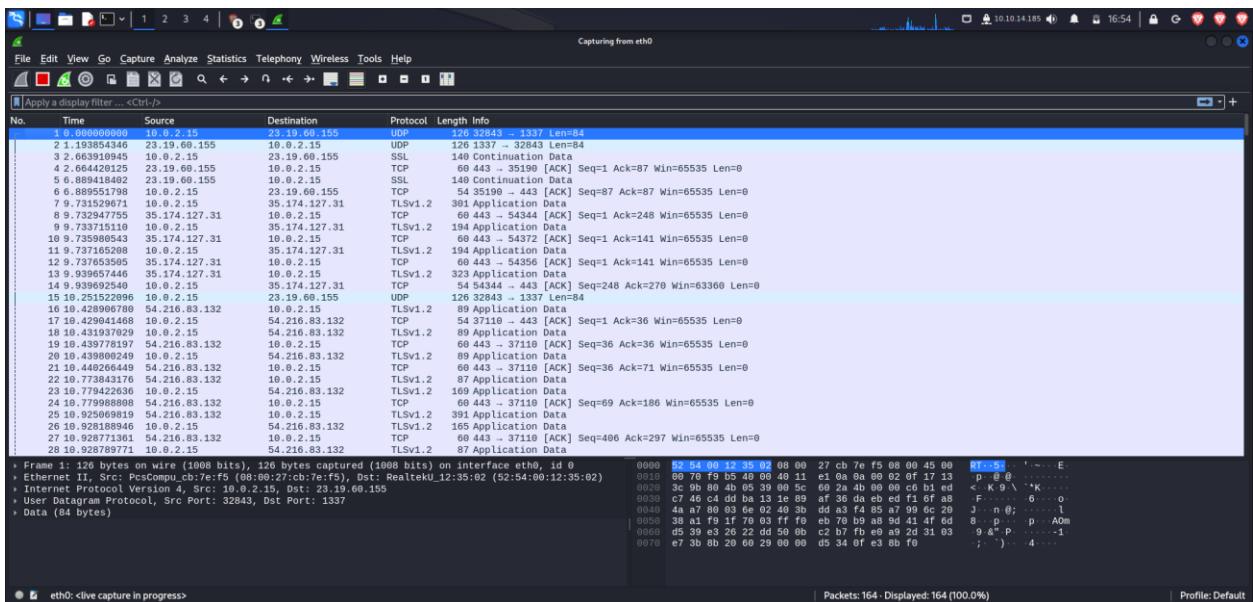
#### Task #1

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



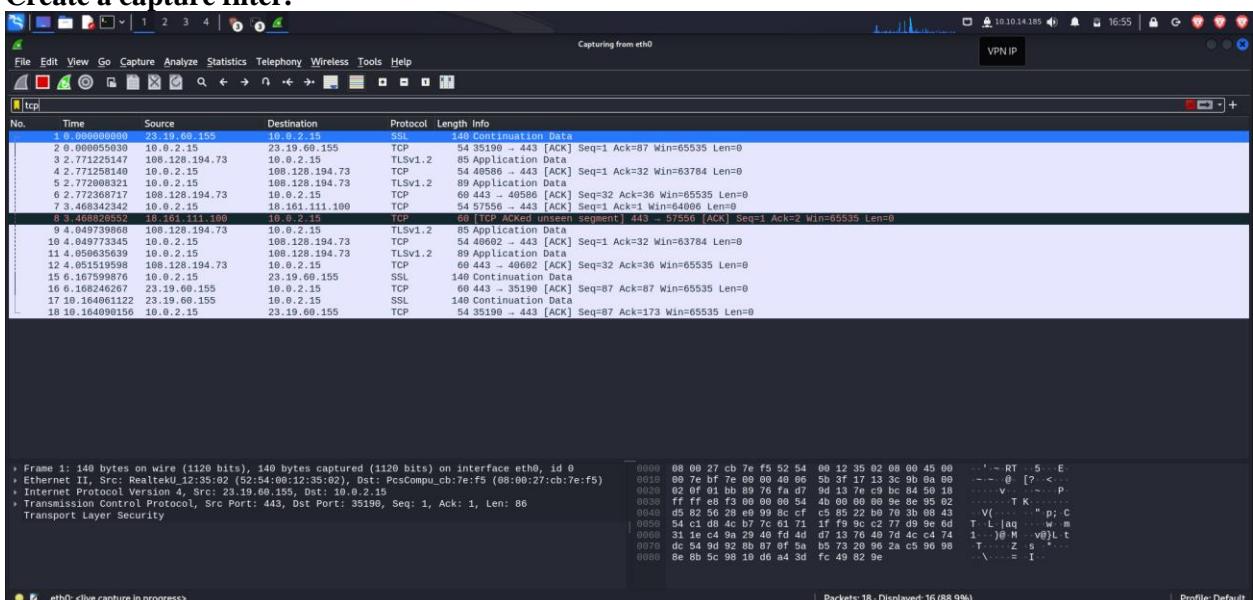
#### Task #2

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



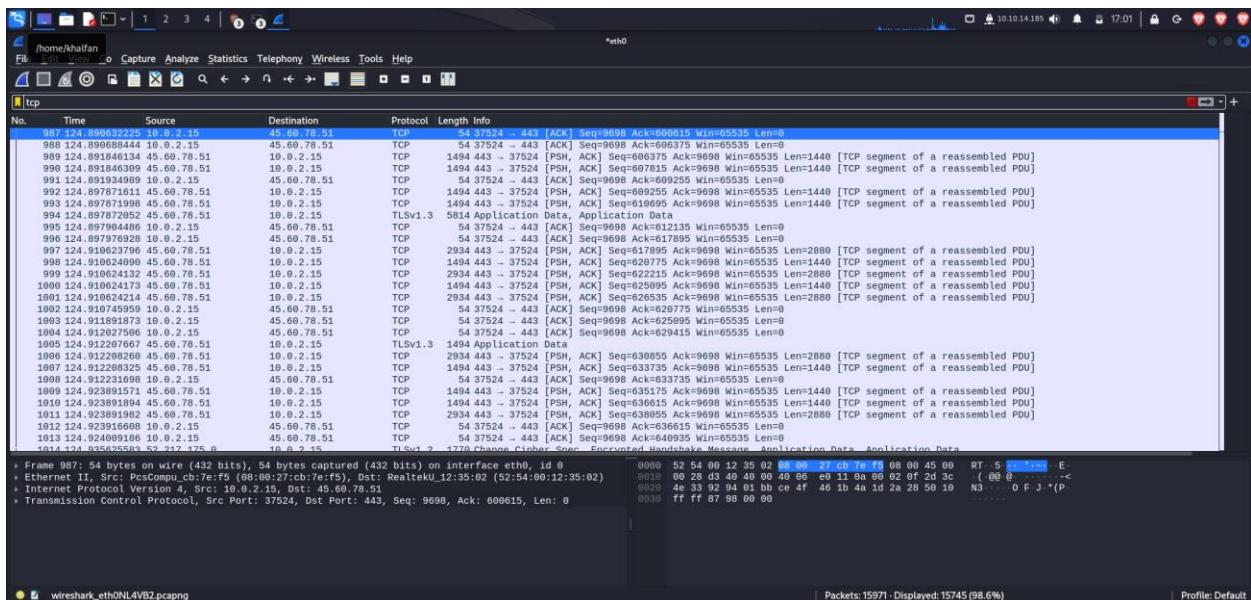
### Task #3

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

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

**Statistics**

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

**Analyze**

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

**tcp**

- 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

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

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

← Previous      Next →       Mark Complete & Next

- 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**

The screenshot shows a browser window for 'Hack The Box - Academy' with the URL [academy.hackthebox.com/module/81/section/789](https://academy.hackthebox.com/module/81/section/789). The page displays a question about a Transformer Leader, asking for the filename of the image. Below the question is a file download link for 'Rise-Up.jpg'. To the right, a 'Wireshark - Export - HTTP object list' window is open, showing a list of captured HTTP requests. The entry for 'Rise-Up.jpg' is highlighted, showing its details: Hostname: 10.10.20.129, Content-Type: image/jpeg, Size: 89 kB, and Filename: http\_with\_ljeps.cap.

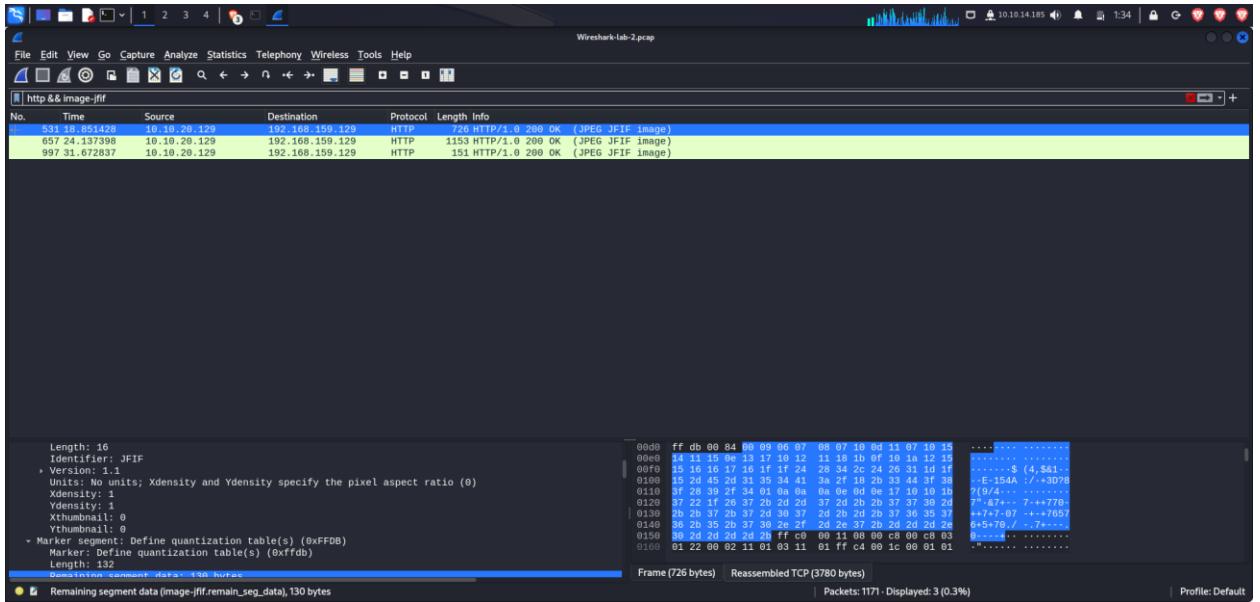
  

The screenshot shows the Wireshark interface with a capture file named 'Wireshark-lab-2.pcap'. A search filter 'http & image-jfif' is applied. The packet list shows three entries related to the challenge:

- No. 531: Src: 10.10.20.129, Dest: 192.168.159.129, HTTP, Length: 151, Info: 151 HTTP/1.0 200 OK (JPEG JFIF image)
- No. 657: Src: 10.10.20.129, Dest: 192.168.159.129, HTTP, Length: 1153, Info: 1153 HTTP/1.0 200 OK (JPEG JFIF image)
- No. 997: Src: 10.10.20.129, Dest: 192.168.159.129, HTTP, Length: 151, Info: 151 HTTP/1.0 200 OK (JPEG JFIF image)

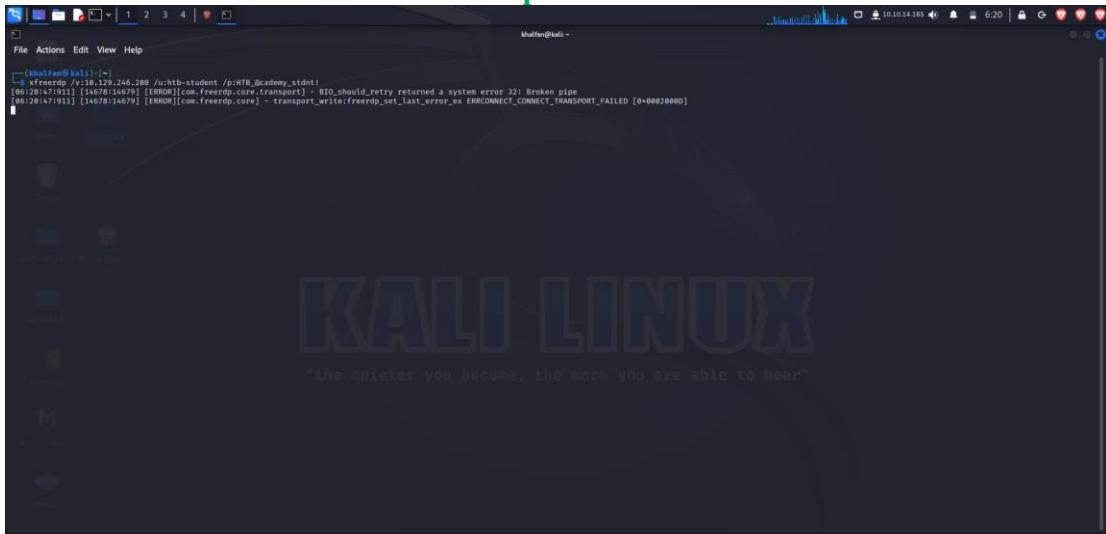
The details and bytes panes show the structure of the JPEG files captured by Wireshark.

- What was the filename of the image that contained a certain Transformer Leader? (name.filetype)  
**Rise-Up.jpg**
- Which employee is suspected of performing potentially malicious actions in the live environment?  
**Bob**

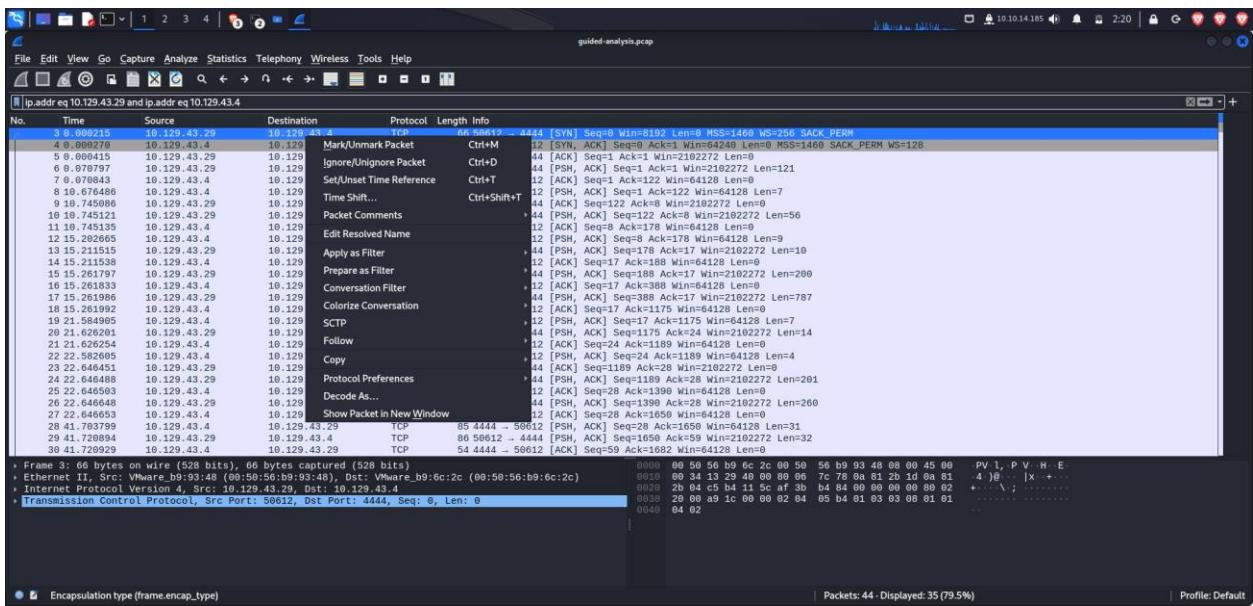


## M. Guided Lab: Traffic Analysis Workflow Tasks

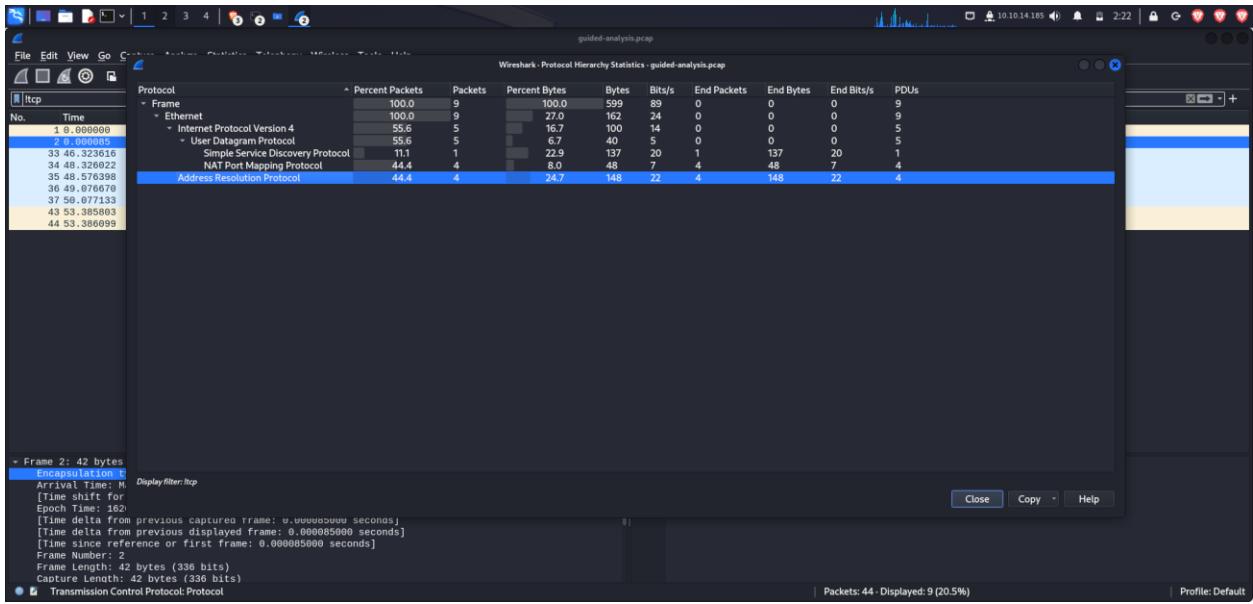
### Task 1: Connect to the live host for capture.



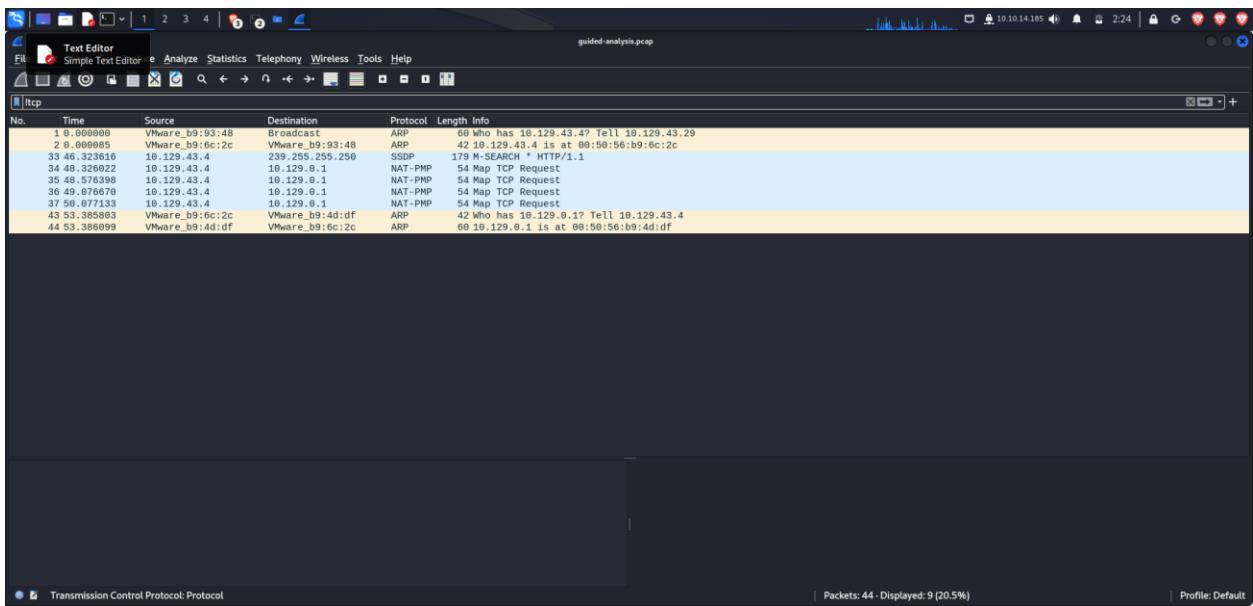
### Conversations



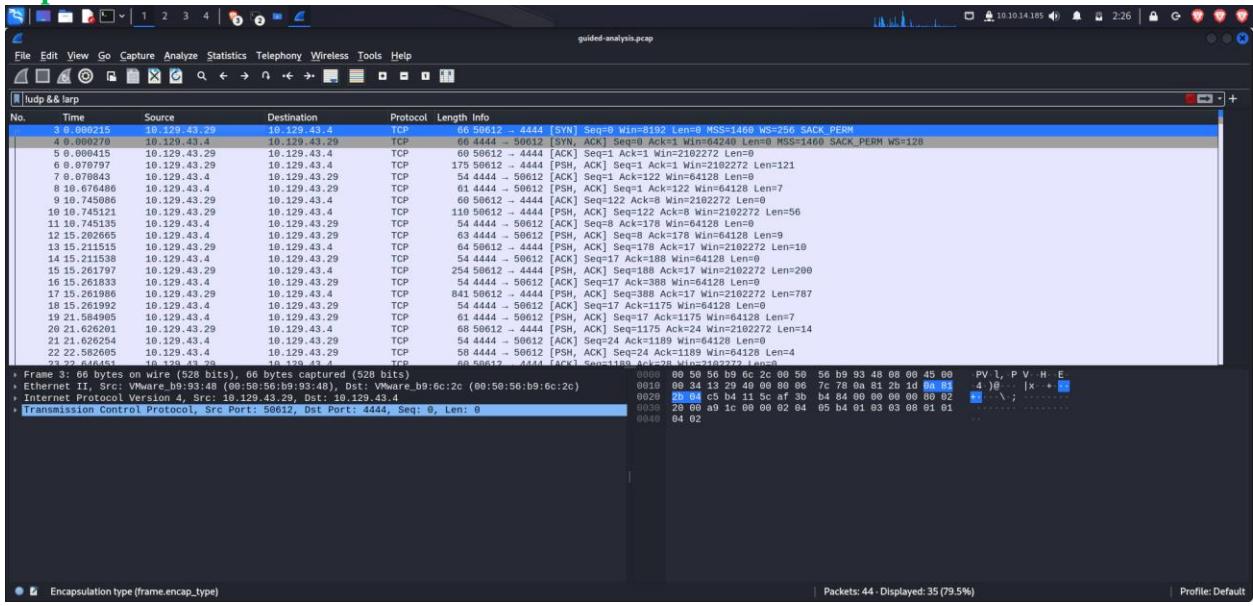
## Protocol Statistics



Udp



## Tcp



Follow tcp stream

Wireshark - Follow TCP Stream (tcp.stream eq 0) - guided-analysis.pcap

Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Information	
3	8.0086215	10.129.43.2	nta-rrdp-srv01.verb3n	RRDP		
4	8.009270	10.129.43.2	c:\Users\verb3n\Downloads>ipconfig	HTTP		
5	8.009415	10.129.43.2	ipconfig	HTTP		
6	8.009425	10.129.43.2	ipconfig	HTTP		
7	8.070843	10.129.43.2	Windows IP Configuration	HTTP		
8	10.676486	10.129.43.2	Windows IP Configuration	HTTP		
9	10.745086	10.129.43.2	Windows IP Configuration	HTTP		
10	10.745121	10.129.43.2	Ethernet adapter Ethernet0:	HTTP		
11	10.745135	10.129.43.2		HTTP		
12	15.262035	10.129.43.2	Connection-specific DNS Suffix . : .htb	HTTP		
13	15.262051	10.129.43.2	IPv6 Address . . . . . : dead:beef::fca1:e285:126d:3b73	HTTP		
14	15.262538	10.129.43.2	Temporary Pv6 Address . . . . . : fe80::fca1:e285:126d:3b73%4	HTTP		
15	15.263179	10.129.43.2	Link-local IPv6 Address . . . . . : fe80::fca1:e285:126d:3b73%4	HTTP		
16	15.261833	10.129.43.2	IPv4 Address . . . . . : 10.129.43.25	HTTP		
17	15.263196	10.129.43.2	Subnet Mask . . . . . : 255.255.0.0	HTTP		
18	15.261992	10.129.43.2	Default Gateway . . . . . : fe80::250:56ff:feb9:ddd%4	HTTP		
19	20.626303	10.129.43.2		HTTP		
20	21.626204	10.129.43.2	Tunnel adapter isatap..htb:	HTTP		
21	22.582665	10.129.43.2		HTTP		
22	22.646451	10.129.43.2	Media State . . . . . : Media disconnected	HTTP		
24	22.646488	10.129.43.2	Connection-specific DNS Suffix . : .htb	HTTP		
25	22.646503	10.129.43.2		HTTP		
26	22.646544	10.129.43.2	Tunnel adapter Teredo Tunneling Pseudo-Interface:	HTTP		
27	22.646550	10.129.43.2		HTTP		
28	41.783799	10.129.43.2	Connection-specific DNS Suffix . :	HTTP		
29	41.728984	10.129.43.2	IPv6 Address . . . . . : 2001:0:2851:782c:28d9:1692:e895:c3a3	HTTP		
30	41.728929	10.129.43.2	Link-local IPv6 Address . . . . . : fe80::28d9:1692:e895:c3a3%2	HTTP		
31	41.783461	10.129.43.2	Default Gateway . . . . . :	HTTP		
32	41.783497	10.129.43.2		HTTP		
38	51.248566	10.129.43.2	c:\Users\verb3n\Downloads>cd c:\	HTTP		
-	Frame 10: 110 bytes on wire (886 bits), 110 bytes captured (886 bits) on interface Ethernet 0 Arrival Time: 10.202020000 seconds (1.202020000 us/byte) [Time shift for this packet: 0.000000000 seconds] Epoch Time: 1620678744.555556 [Time delta from previous cap: 0.000000000 seconds] [Time delta from previous disp: 0.000000000 seconds] [Time since reference or first cap: 0.000000000 seconds]	12 client pkts, 6 server pkts, 122 total	Entire conversation (0.914 bytes)	Show data as ASCII	Stream 0	Find Next
Find:	Filter Out This Stream	Print	Save as...	Back	Close	Help

**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**

```

Tunnel adapter Teredo Tunnelling Pseudo-Interface:
Connection-specific DNS Suffix . :
IPv6 Address . . . . . : 2861:0:2851:782c:28d9:1692:e895:c3a3
Link-local IPv6 Address . . . . . : fe80::28d9:1692:e895:c3a3%2
Default Gateway . . . . . :

c:\Users\mrbsn\Downloads>cd c:\

c:\>dir
dir
Volume in drive C has no label.
Volume Serial Number is EBC0-0EA6

Directory of c:\

07/10/2010 04:47 AM <DIR> PerLogs
05/19/2021 01:08 PM <DIR> Program Files
05/19/2021 01:10 PM <DIR> Program Files (x86)
05/10/2020 09:34 PM <DIR> Users
05/10/2022 12:46 PM <DIR> Windows
0 File(s) 0 bytes

5 Dir(s) 21,421,400,864 bytes free

c:\>net user hacker PassWrd1 /add
net user hacker PassWrd1 /add
The command completed successfully.

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

c:\>

```

Entire conversation (1,914 bytes) Show data as ASCII Stream 0 Find Next Find:

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Vmware b9:93:48	Broadcast	ARP	68	who has 10.129.43.4 tell me
2	0.000005	Vmware b9:93:48	10.129.43.4	ARP	68	is at 00:56:0f:b9:93:48
3	0.000215	10.129.43.4	10.129.43.4	TCP	66	58612 - 4444 [SYN] Seq=0 Win=1312 Len=0 MSS=1460 WS=256 SACK_PERM
4	0.000270	10.129.43.4	10.129.43.29	TCP	66	58612 - 58612 [SYN, ACK] Seq=1 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
5	0.000415	10.129.43.29	10.129.43.4	TCP	60	58612 - 4444 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
6	0.070797	10.129.43.29	10.129.43.4	TCP	175	58612 - 4444 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=121
7	0.070802	10.129.43.4	10.129.43.29	TCP	61	4444 - 58612 [ACK] Seq=122 Ack=122 Win=64128 Len=7
8	0.070806	10.129.43.4	10.129.43.29	TCP	60	58612 - 4444 [PSH, ACK] Seq=122 Ack=122 Win=2102272 Len=0
9	0.070808	10.129.43.29	10.129.43.4	TCP	119	58612 - 4444 [PSH, ACK] Seq=122 Ack=122 Win=2102272 Len=56
10	0.070812	10.129.43.29	10.129.43.4	TCP	54	4444 - 58612 [ACK] Seq=128 Ack=128 Win=64128 Len=0
11	0.070815	10.129.43.4	10.129.43.29	TCP	63	4444 - 58612 [PSH, ACK] Seq=8 Ack=178 Win=64128 Len=9
12	0.070816	10.129.43.4	10.129.43.29	TCP	64	4444 - 58612 [ACK] Seq=9 Ack=178 Win=64128 Len=10
13	0.070818	10.129.43.4	10.129.43.29	TCP	54	4444 - 58612 [ACK] Seq=10 Ack=178 Win=64128 Len=9
14	0.070819	10.129.43.4	10.129.43.29	TCP	254	58612 - 4444 [PSH, ACK] Seq=188 Ack=17 Win=2102272 Len=200
15	0.070821	10.129.43.4	10.129.43.29	TCP	54	4444 - 58612 [ACK] Seq=17 Ack=388 Win=64128 Len=0
16	0.070822	10.129.43.4	10.129.43.29	TCP	841	58612 - 4444 [PSH, ACK] Seq=388 Ack=17 Win=2102272 Len=787
17	0.070823	10.129.43.4	10.129.43.29	TCP	54	4444 - 58612 [ACK] Seq=17 Win=64128 Len=0
18	0.070824	10.129.43.4	10.129.43.29	TCP	61	4444 - 58612 [ACK] Seq=175 Ack=175 Win=64128 Len=7
19	0.070825	10.129.43.4	10.129.43.29	TCP	60	58612 - 4444 [PSH, ACK] Seq=175 Ack=175 Win=2102272 Len=14
20	0.070826	10.129.43.4	10.129.43.29	TCP	54	4444 - 58612 [ACK] Seq=24 Ack=1189 Win=64128 Len=0
21	0.070827	10.129.43.4	10.129.43.29	TCP	255	58612 - 4444 [PSH, ACK] Seq=1189 Ack=28 Win=2102272 Len=201
22	0.070828	10.129.43.4	10.129.43.29	TCP	54	4444 - 58612 [ACK] Seq=28 Ack=1189 Win=64128 Len=0
23	0.070829	10.129.43.4	10.129.43.29	TCP	314	58612 - 4444 [PSH, ACK] Seq=28 Ack=1189 Win=64128 Len=260
24	0.070830	10.129.43.4	10.129.43.29	TCP	54	4444 - 58612 [ACK] Seq=28 Ack=1650 Win=64128 Len=0
25	0.070831	10.129.43.4	10.129.43.29	TCP	85	4444 - 58612 [PSH, ACK] Seq=28 Ack=1650 Win=64128 Len=31

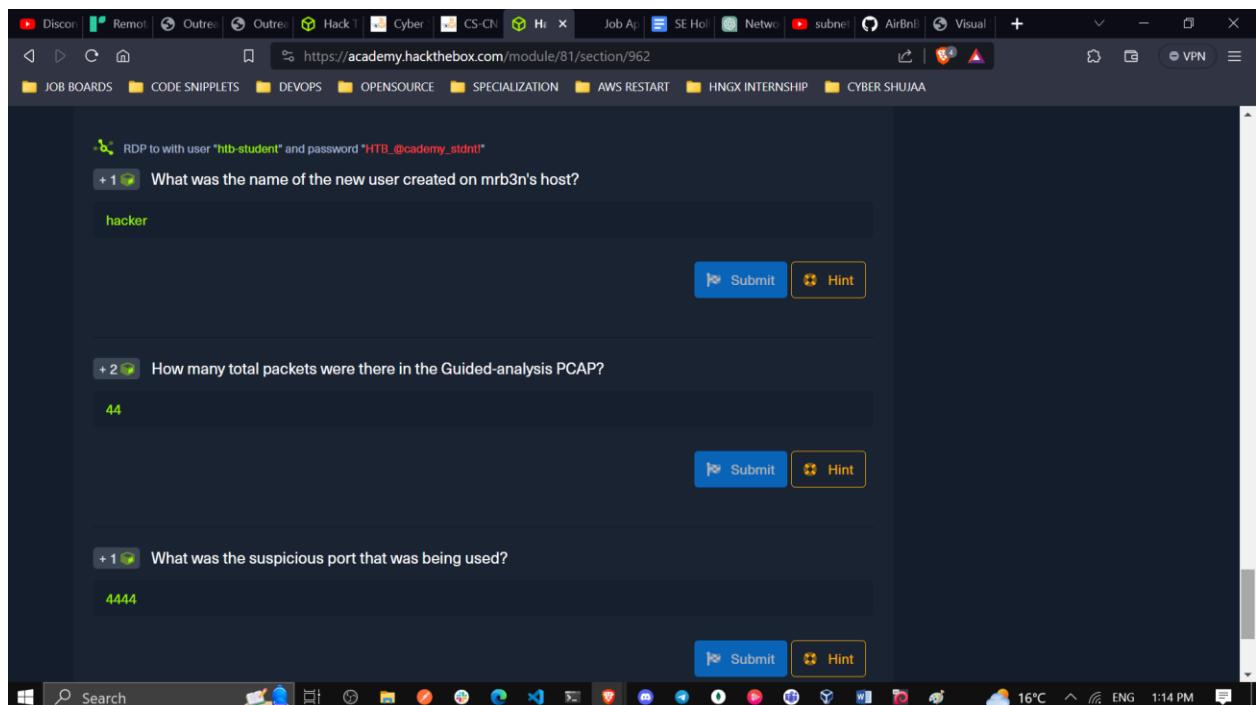
Frame 1: 00 bytes on wire (480 bits), 00 bytes captured (480 bits)
 Ethernet II, Src: VMware b9:93:48 (00:56:0f:b9:93:48), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 > Ethernet II, Src: VMware b9:93:48 (00:56:0f:b9:93:48), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 > Address Resolution Protocol (request)

0000 ff ff ff ff ff ff 00 00 56 0f b9 93 48 00 00 00 01
 0001 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0002 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0003 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Packets: 44 - Displayed: 44 (100.0%) | Profile: Default

44 located at the lower right side of the screen

iii. What was the suspicious port that was being used?  
4444



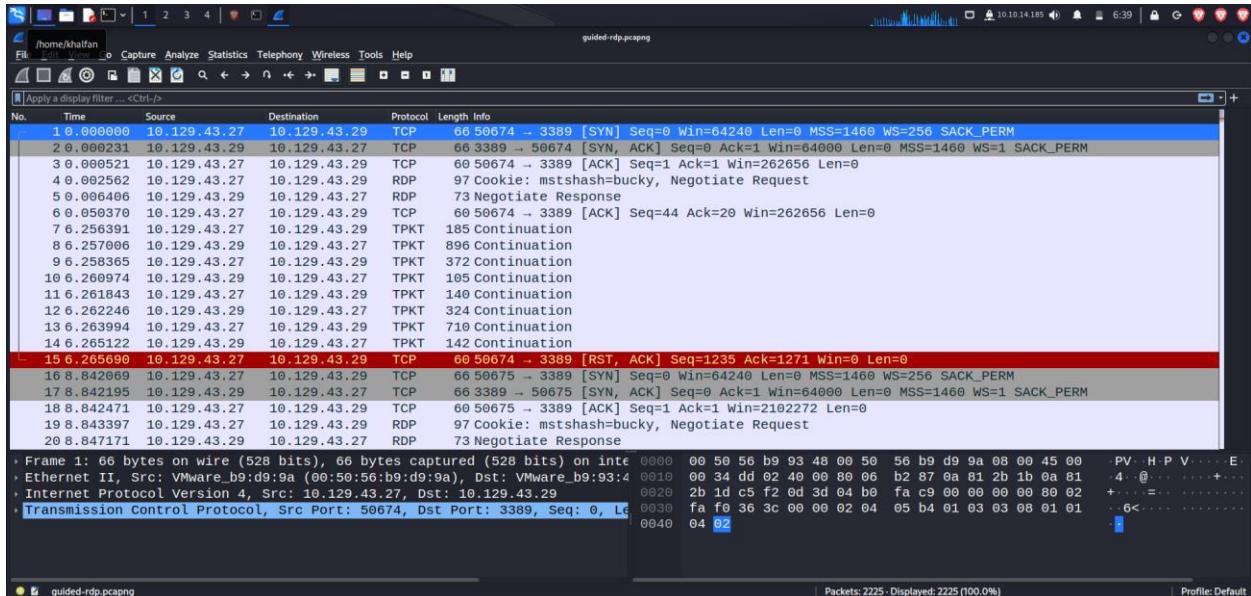
## N. Describing RDP connections

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

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

The terminal window shows the following analysis results:

File	Description
http_with_jpegs.cap	RDP analysis zip
i40e8837v1n1_Pocket_Tracer_Quiz.pka	Wireshark Lab-2.pcap
guided-analysis.zip	#Wireshark-Lab-2.zip
http.pcap	tcpdump-lab-2-answers.md
http_with_jpegs.pcap	TCPdump-lab-2.pcap
Packet_Tracer21_md4.signd.dll	#Walkthrough Answers.md
question-1.zip	Walkthrough Answers.md

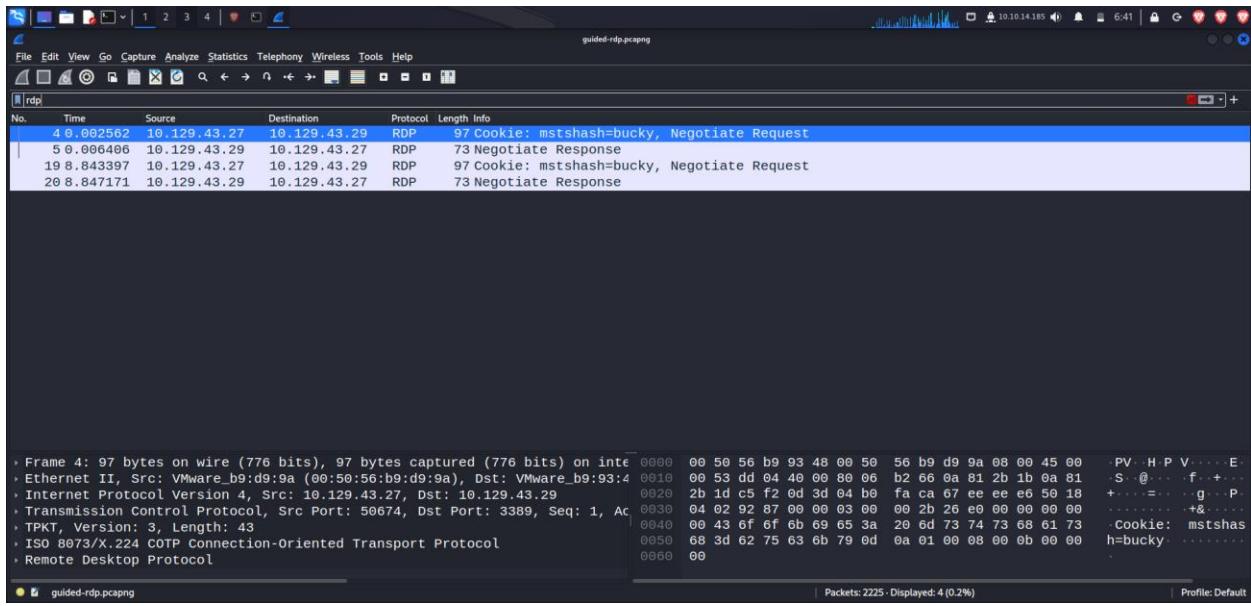


## Task #2

### Analyze the traffic included.

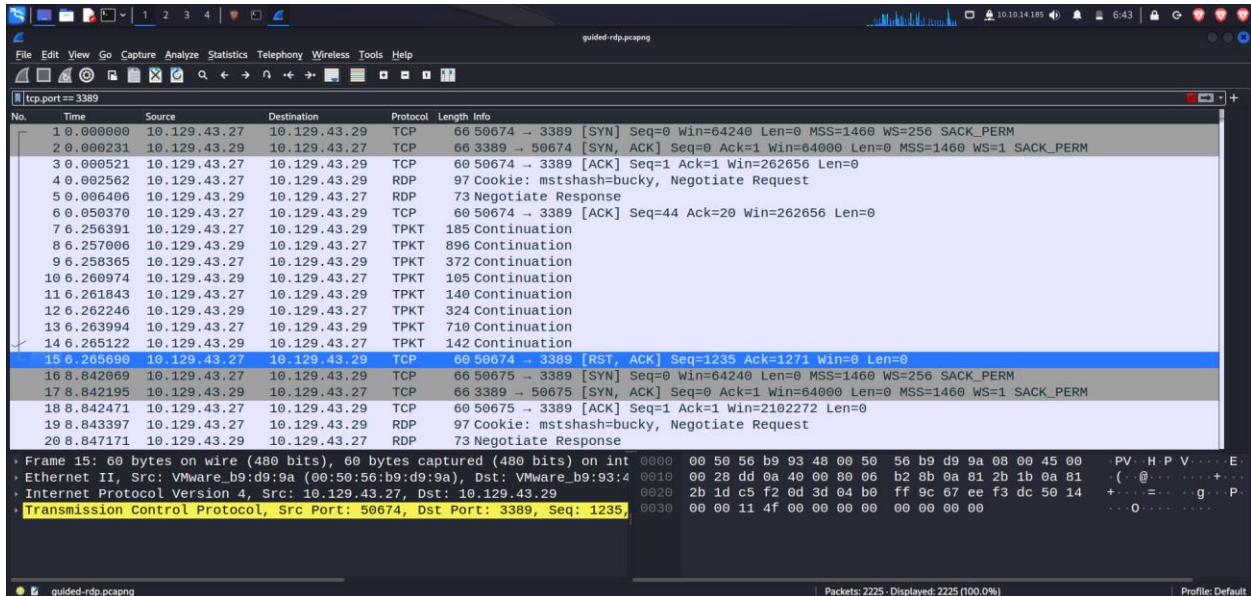
Take a minute to look at the traffic. Notice there is a lot of information here. We know our focus is on RDP, so let's take a second to filter on rdp and see what it returns.

### RDP Filter



**utilize the display filter `tcp.port == 3389`.**

**Filter For TCP Port 3389**

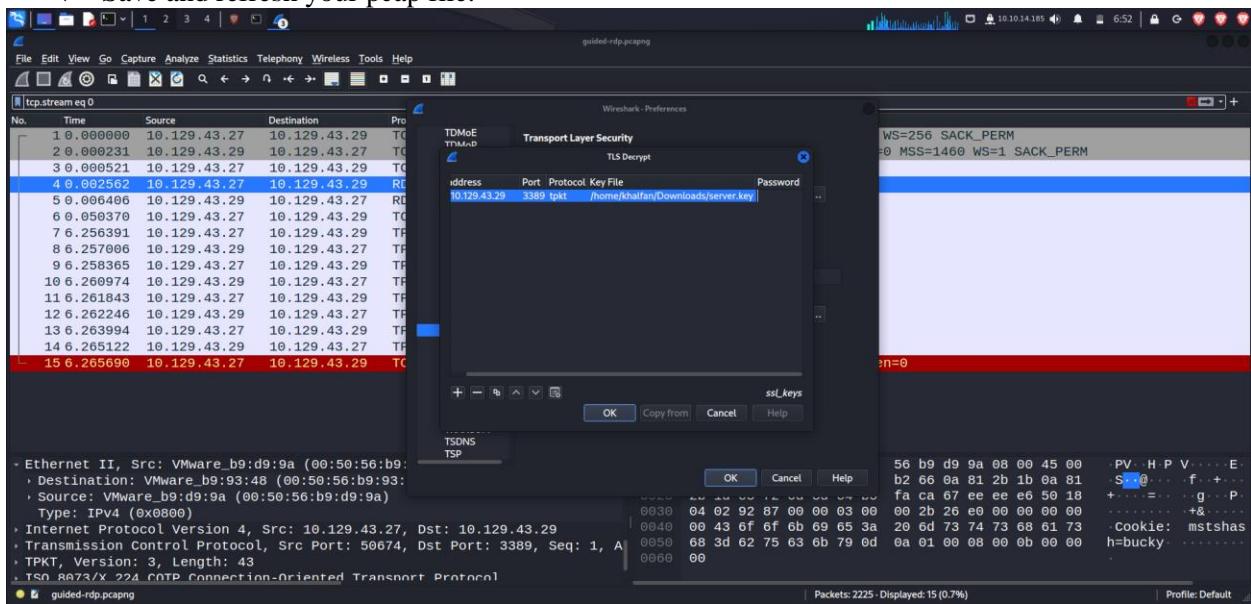


### Task #3

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

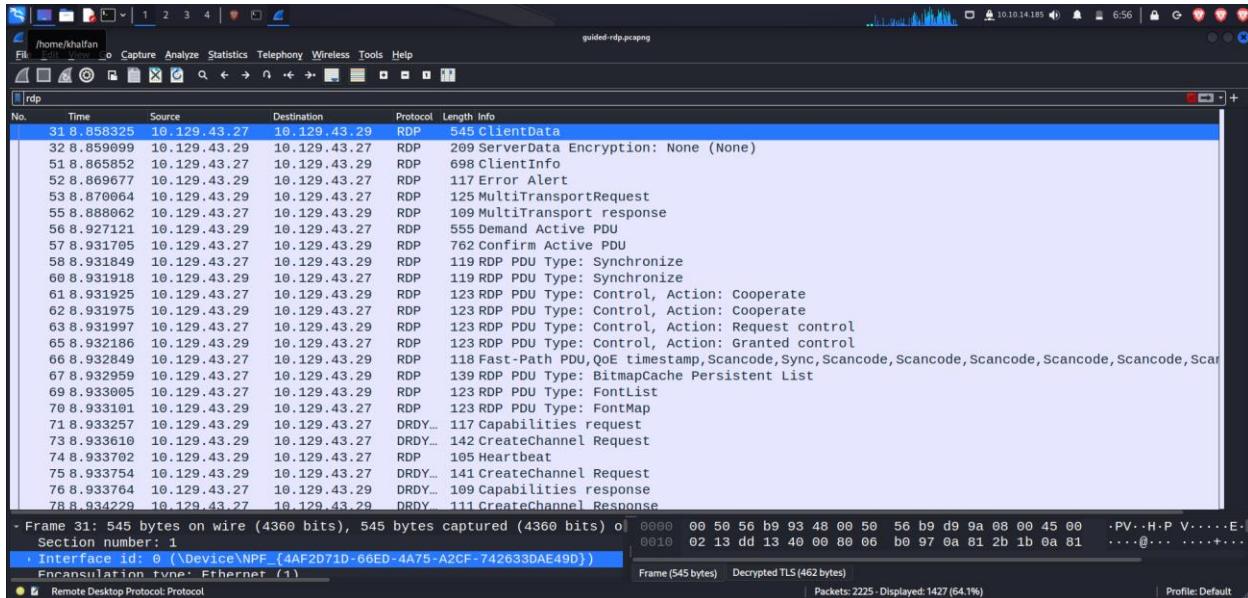
#### Steps

- ◆ 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 field equals tpkt or blank.
- ◆ Browse to the server.key file and add it in the key file section.
- ◆ Save and refresh your pcap file.



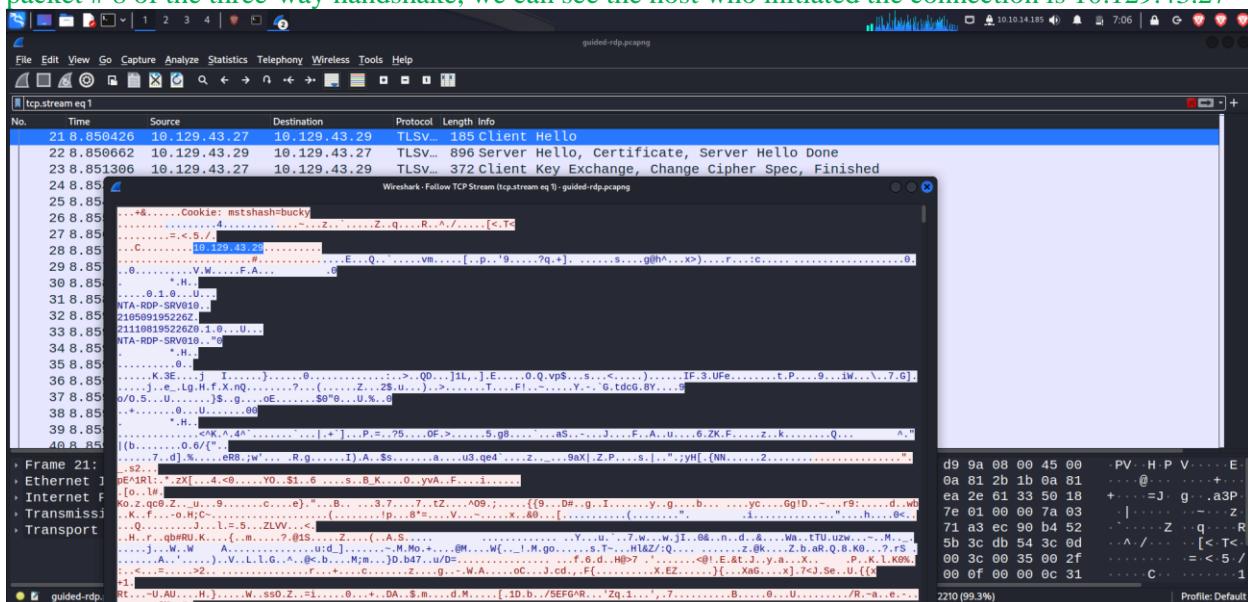
When filtering once again on RDP, we should see some traffic in the display.

### RDP In The Clear

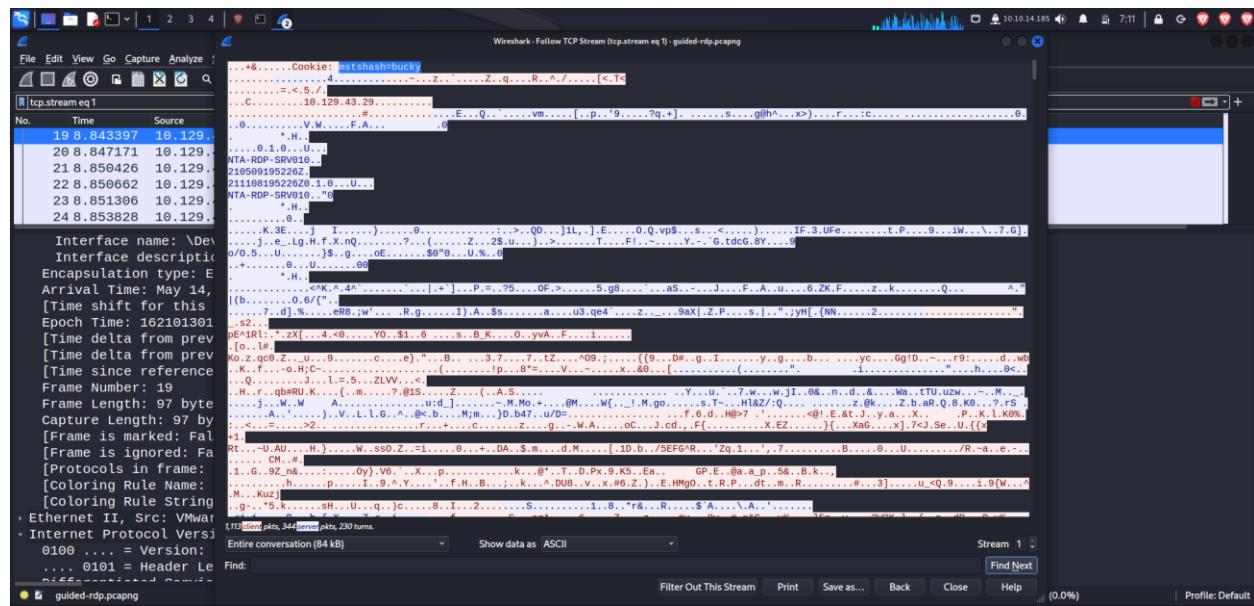
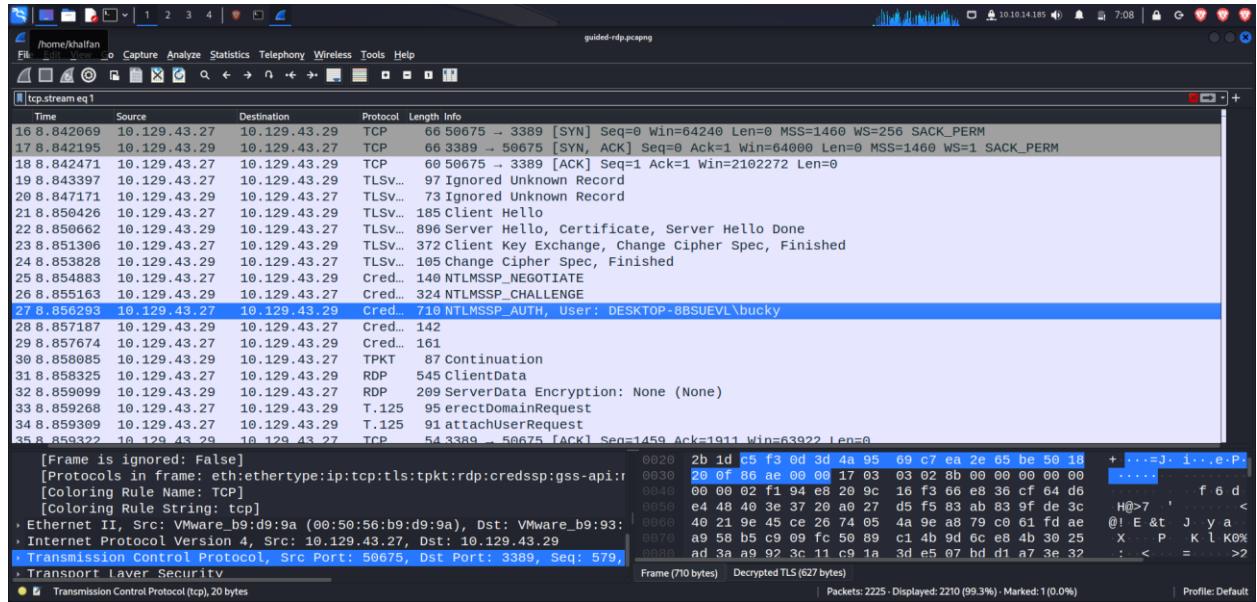


## Perform Analysis of the Unencrypted Traffic

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



- 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.

Cyber Sh... Cyber Sh... Login | Skip Practice E... 10.4.4 Lab Subnetting CS-CNS2 Wireshark Intro to N... Intro (165) + - ×

JOB BOARDS CODE SNIPPETS DEVOPS OPENSOURCE SPECIALIZATION AWS RESTART HNGX INTERNSHIP CYBER SHUJAA

https://academy.hackthebox.com/module/finish

HTB ACADEMY Great job khalfan6! Purchase Cubes khalfan6

INTRO TO NETWORK TRAFFIC ANALYSIS

Completed / Congrats!

LEARN

Dashboard Exams Modules Paths

in Share on LinkedIn Share on Twitter Share on Facebook

Congratulations khalfan6! You have just completed the Intro to Network Traffic Analysis module! Share your success with everyone!

Get a shareable link

Change Log Retake Module

Search 26°C ENG 6:58 PM

A screenshot of a computer monitor displaying the HTB Academy website. The main content area shows a completion message: "Great job khalfan6!" followed by "Completed / Congrats!". Below this, there's a large banner for the "Intro to Network Traffic Analysis" module, featuring a 3D illustration of a laptop and network components. On the left sidebar under "LEARN", the "Dashboard" option is selected. At the bottom of the page, there are social sharing buttons for LinkedIn, Twitter, and Facebook, along with a "Get a shareable link" button. The taskbar at the bottom of the screen shows various open applications and system status.