Course: Cloud and Network Security – C2 – 2025.

Student: Luke Mbogo

Student No: cs-cns09-25076

Tuesday, May 30, 2025.

Week 2 Assignment 2:

Network Traffic Analysis (HTB).

Contents

Introduction 4

Network Traffic Analysis 4

Networking Primer - Layers 1-4. 5

OSI/TCP-IP Models. 6

OSI / TCP-IP Model differences 6

PDU Breakdown. 7

MAC-Addressing 8

IPv4 & IPv6 Addresses. 9

TCP/UDP and Transport Mechanisms. 9

HTB Answers. 12

Networking Primer - Layers 5-7 13

HTTP and HTTPS. 13

FTP 13

SMB 14

HTB Answers. 14

The Analysis Process. 15

Analysis Dependencies 15

Analysis in Practice 16

Descriptive Analysis 16

Diagnostic Analysis 16

Predictive Analysis 17

Prescriptive Analysis 17

Tcpdump Fundamentals 18

Switch commands. 19

Packet Analysis. 19

HTB Answers. 20

Fundamentals Lab. 21

Tcpdump Packet Filtering 21

Task. 22
HTB Answers. 23
Interrogating Network Traffic With Capture and Display Filters 25
HTB Answers. 27
Analysis with Wireshark 27
HTB Answers 29
Packet Inception, Dissecting Network Traffic With Wireshark. 30
Task. 30
Wireshark Lab Report 31
FTP Analysis 32
Lab Report Summary. 33
HTB Answers. 34
Guided Lab: Traffic Analysis Workflow 35
HTB Answers. 37
Decrypting RDP connections 38
HTB Answers. 39
Conclusion 39
HTB Badge. 40

Introduction

Network Traffic Analysis

In this first module on Network Traffic Analysis (NTA), I learned the fundamental role it plays in cybersecurity—mainly in identifying threats, understanding normal network behavior, and supporting incident response. I now understand that by analyzing traffic patterns and using the right tools, I can detect unusual or malicious activity more effectively. What stood out to me was how attackers can use legitimate-looking tools and credentials to stay hidden, making NTA a vital skill for uncovering these threats early. This module gave me a solid foundation in both the theory and tools needed to start analyzing real network traffic.

Key Take-aways.

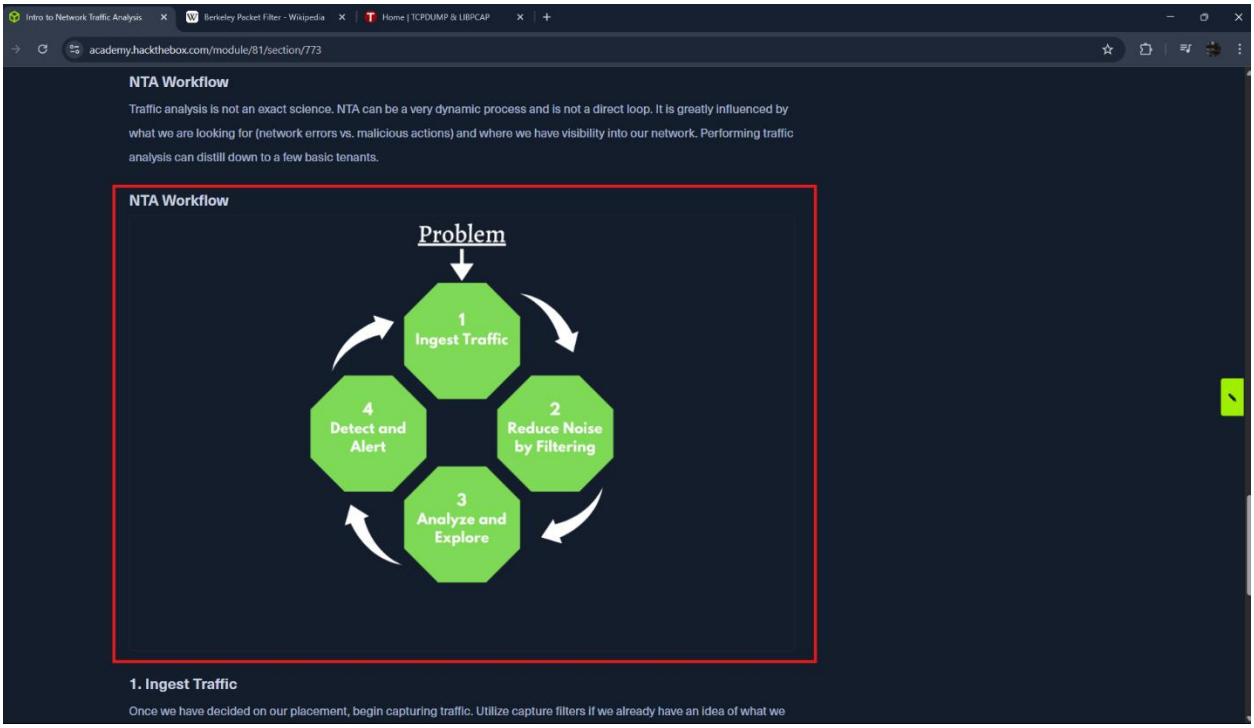
- I learned that Network Traffic Analysis helps create a baseline for normal network behavior and is key in spotting irregular activities like malware, port scans, and protocol misuse.
- I now have a clearer understanding of the OSI and TCP/IP models, how packets flow through networks, and how different protocols operate. This knowledge helps me better interpret the traffic I'm analyzing.
- I was introduced to powerful tools like Wireshark, tcpdump, TShark, NGrep, and tcpick. I now know what each tool is best suited for—whether it's live capture, reassembly of TCP streams, or pattern matching.
- I learned how devices like network taps and SPAN ports help in capturing traffic, especially in segmented or VLAN-based networks.
- One useful skill I picked up was using BPF (Berkeley Packet Filter) syntax to reduce noise and focus on relevant traffic during analysis.
- The four-step workflow—Ingest, Filter, Analyze, and Detect—gave me a structured approach to handle traffic analysis efficiently. I also understood that "Fix and Monitor" is a necessary follow-up step once action is taken.

This module helped me build a strong base in traffic analysis and made me more confident in using tools and techniques to investigate network activity. It also gave me a clearer picture of what to look for when hunting for threats in a real environment. While studying traffic capture and analysis, I came across **egress** and **ingress** processing:

- **Ingress processing** refers to how **incoming network traffic** is handled **as it enters** a device, like a switch, firewall, or endpoint. This includes filtering, routing decisions, and logging.

- **Egress processing** is about how **outgoing traffic** is managed **as it leaves** a device or network segment. This might include encryption, tagging, or applying firewall rules before data is sent out.

NTA Workflow Diagram.



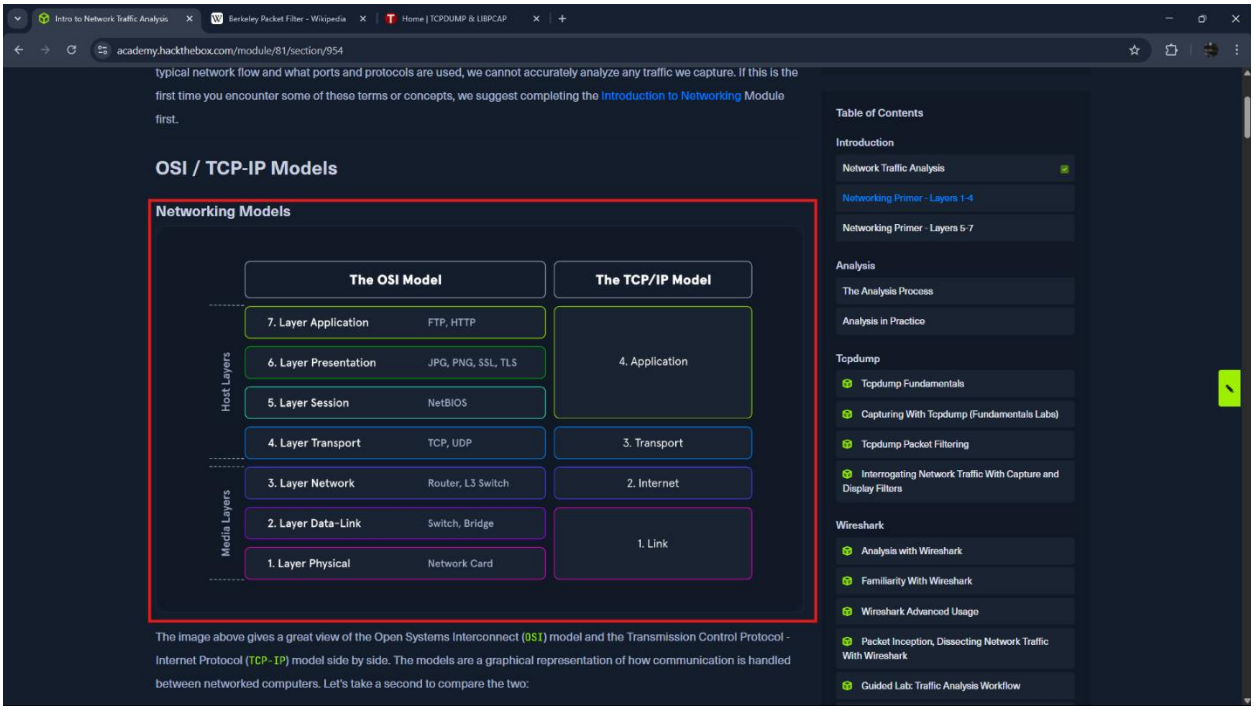
Networking Primer - Layers 1-4.

This section refreshed my understanding of the fundamental networking concepts, especially the first four layers of the OSI and TCP/IP models. These layers are essential because they define how data moves across the network—from the physical cables and signals (Layer 1) up to the transport layer (Layer 4), which manages communication between devices.

I learned that without a solid grasp of these layers and how common protocols operate within them, it would be impossible to make sense of captured network traffic. Knowing which ports and protocols are typically used at these layers helps me recognize normal network behavior and spot suspicious activities.

Understanding these models gives me the foundation to dissect traffic captures more effectively and supports my ability to identify potential security issues. The image below gives a great view of the Open Systems Interconnect (OSI) model and the Transmission Control Protocol - Internet Protocol (TCP-IP) model side by side

OSI/TCP-IP Models.



OSI / TCP-IP Model differences

The difference between the **TCP/IP model** and the **OSI model** — they are two different **networking models** used to understand how data travels in a network.

Key Differences:

Aspect	**OSI Model**	**TCP/IP Model**
Full Name	Open Systems Interconnection	Transmission Control Protocol/Internet Protocol
Developed by	ISO (International Standard Org)	U.S. Department of Defense
Layers	7 Layers	4 Layers (some say 5)
Use Case	Theoretical reference	Practical/Real-world protocol stack (used on the Internet)
Commonality	Used mainly for teaching and standardization	Used in actual network communication (e.g., Internet)

OSI Model - 7 Layers:

1. Application
2. Presentation
3. Session
4. Transport
5. Network
6. Data Link
7. Physical

TCP/IP Model - 4 Layers:

1. Application
2. Transport
3. Internet
4. Network Access (or Link)

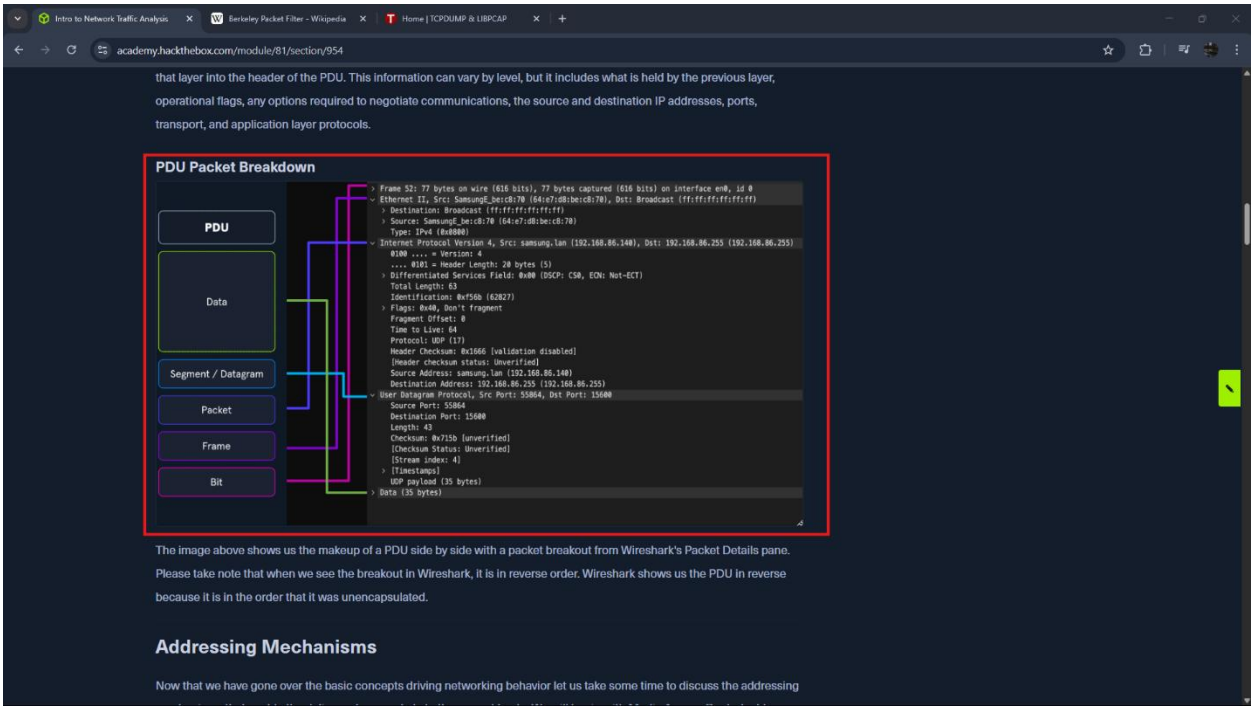
PDU Breakdown.

I learned that a Protocol Data Unit (PDU) is basically a packaged chunk of data that gets sent over the network. As data moves down through the different layers of the network stack, each layer **wraps the data from the previous layer inside a new "bubble"**, a process called **encapsulation**. Each layer adds its own header with important info like addresses, ports, flags, and protocol details.

When looking at a packet in Wireshark, the order shown is actually the reverse of how the data was encapsulated — it shows the packet as it is being **unwrapped** from the bottom layer up. This means Wireshark starts with the data closest to the application and moves outward toward the physical layer.

Understanding encapsulation is key because it helps me see what info is added at each step and how the data is structured, which is crucial when analyzing packets for anomalies or threats.

PDU Packet Breakdown.



MAC-Addressing

I learned that every network interface, whether physical or logical, has a unique Media Access Control (MAC) address. This is a 48-bit identifier shown in hexadecimal, used specifically at **Layer 2** (the data-link layer) to enable communication between devices on the same local network or broadcast domain.

MAC addresses are essential because they ensure that packets get delivered to the right device within that local network segment. When traffic needs to cross from one network to another (Layer 3), the packet is sent to the router's interface using its MAC address. The router then looks at the Layer 3 IP address to decide where to send the packet next. During this routing, the Layer 2 encapsulation (including the MAC address) is stripped off and replaced with the new MAC address for the next hop in the path.

This process helped me understand how local delivery and routing work together through different layers and how MAC addresses support device identification on a local network.

I learned that the Internet Protocol (IP) is what enables data to travel from one host to another across different networks. IP handles the routing of packets, breaking down data into smaller fragments if needed, and then reassembling them at the destination.

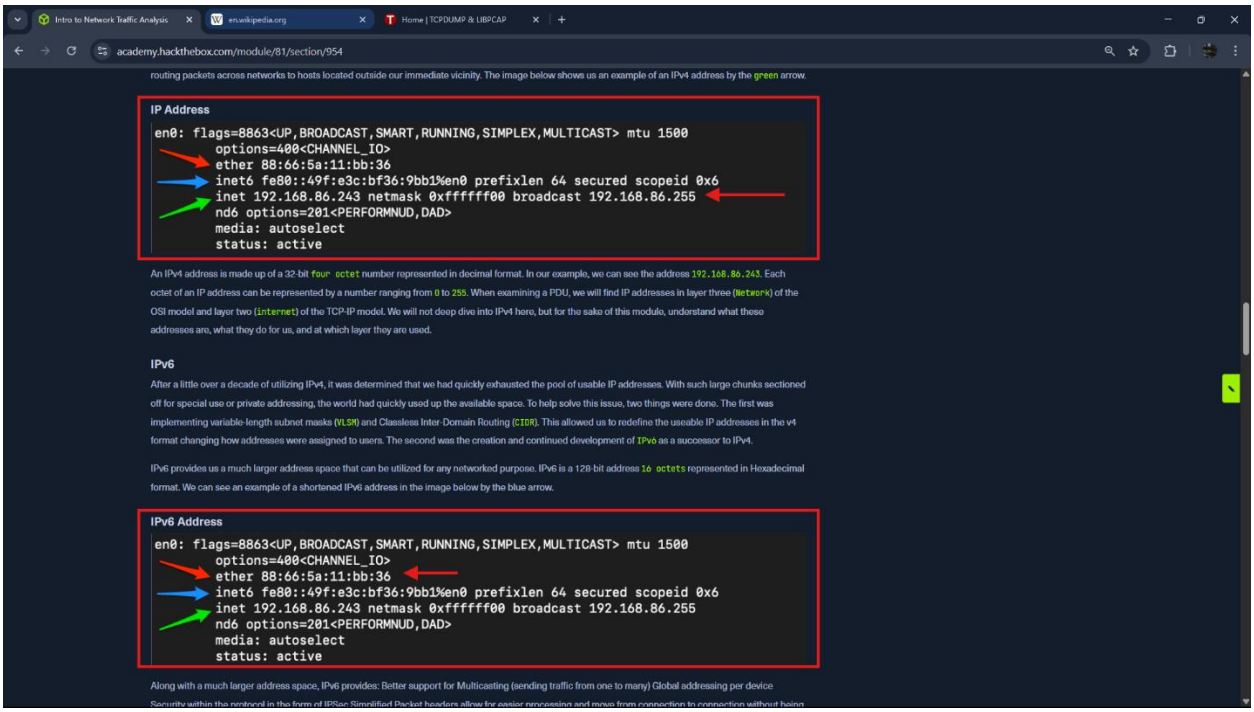
IP itself is **connectionless**, meaning it doesn't guarantee that data will reach its destination or arrive in order. For reliability, IP depends on upper-layer protocols like TCP.

There are two main versions of IP today:

- **IPv4** — The most common version currently used, which assigns addresses like the familiar 192.168.x.x format.
- **IPv6** — Designed as the successor to IPv4, with a much larger address space to accommodate the growing number of devices online.

Understanding IPv4 addressing is crucial because it's the core way networks route packets to hosts beyond the local network, ensuring communication between different devices globally.

IPv4 & IPv6 Addresses.



TCP/UDP and Transport Mechanisms.

At the **Transport Layer**, I learned that this is where decisions are made on how data is sent and received. It's like the control center that manages how application data gets broken down, sent out, and then reassembled properly when received.

The two main protocols used here are:

TCP (Transmission Control Protocol)

- **Connection-oriented** – it sets up a connection first using a **3-way handshake**.

- **Reliable** – it tracks every packet using **sequence and acknowledgment numbers**.
- **Used for** things like SSH or file transfers, where data integrity is critical.
- **Slower**, but safer – it won't let partial data through if something goes wrong.

UDP (User Datagram Protocol)

- **Connectionless** – no handshake, it just sends the data.
- **Fast but unreliable** – doesn't care if the packet arrives or not.
- **Used for** video streaming or DNS, where speed is more important than perfect delivery.
- **No acknowledgments** – it's a fire-and-forget method.

The key difference is that **TCP ensures the data gets there correctly**, while **UDP just tries to get it there fast**. A real-world comparison that helped me understand this:

- TCP is like sending registered mail – the recipient signs for it.
- UDP is like dropping flyers at every door – no guarantees, but it's quick.

*TCP Three-way Handshake***.**

The **Transmission Control Protocol (TCP)** establishes reliable connections between hosts through a process called the **three-way handshake**. This process ensures both parties are synchronized and ready to exchange data accurately. It begins when the **client sends a TCP packet** with the **SYN flag** set. This packet contains a sequence number and other options like window size or maximum segment size. It's the client's way of saying, "I want to start a session and here's how I want to communicate."

Next, the **server replies with a SYN-ACK packet**. This serves two purposes: it acknowledges the client's SYN and also includes the server's own SYN request, along with any modified options it prefers. This back-and-forth ensures both sides agree on the communication parameters.

Finally, the **client sends an ACK packet** to confirm the server's SYN. At this point, the connection is established, and the two devices can begin exchanging data. For example, if a user requests a webpage, the HTTP request will follow right after this handshake.

- SYN - Client initiates
- SYN-ACK - Server responds
- ACK - Client confirms

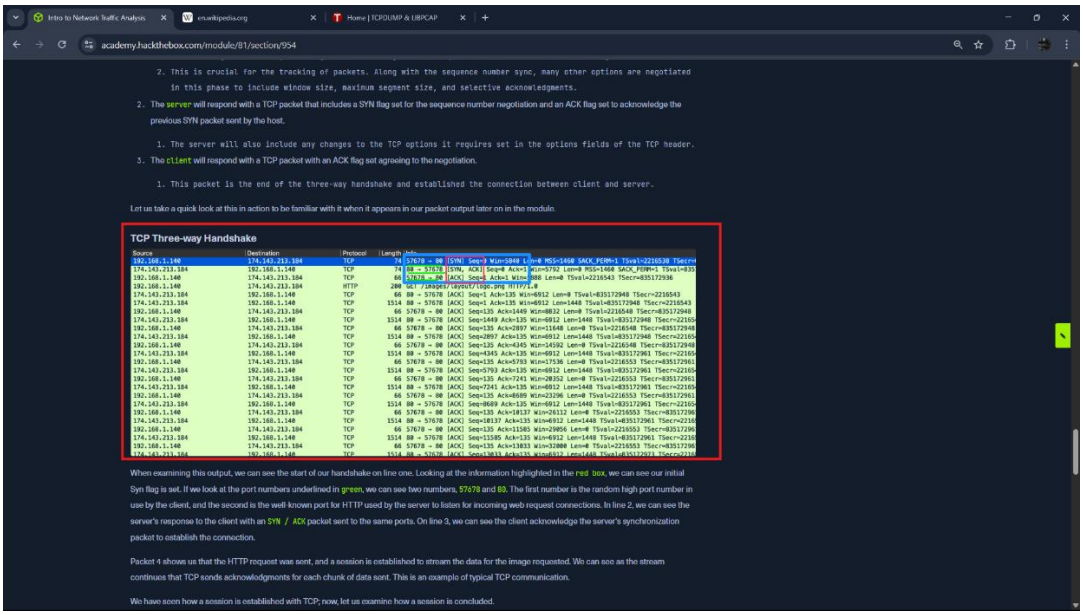
When the session is over, **TCP uses the FIN flag** to terminate it gracefully. The host that wants to end the session sends a **FIN, ACK** packet. The other party responds with its own **FIN, ACK**, and

finally, the original sender replies with an **ACK** to confirm closure. This four-packet exchange ensures that all data has been sent and received properly before the connection is closed.

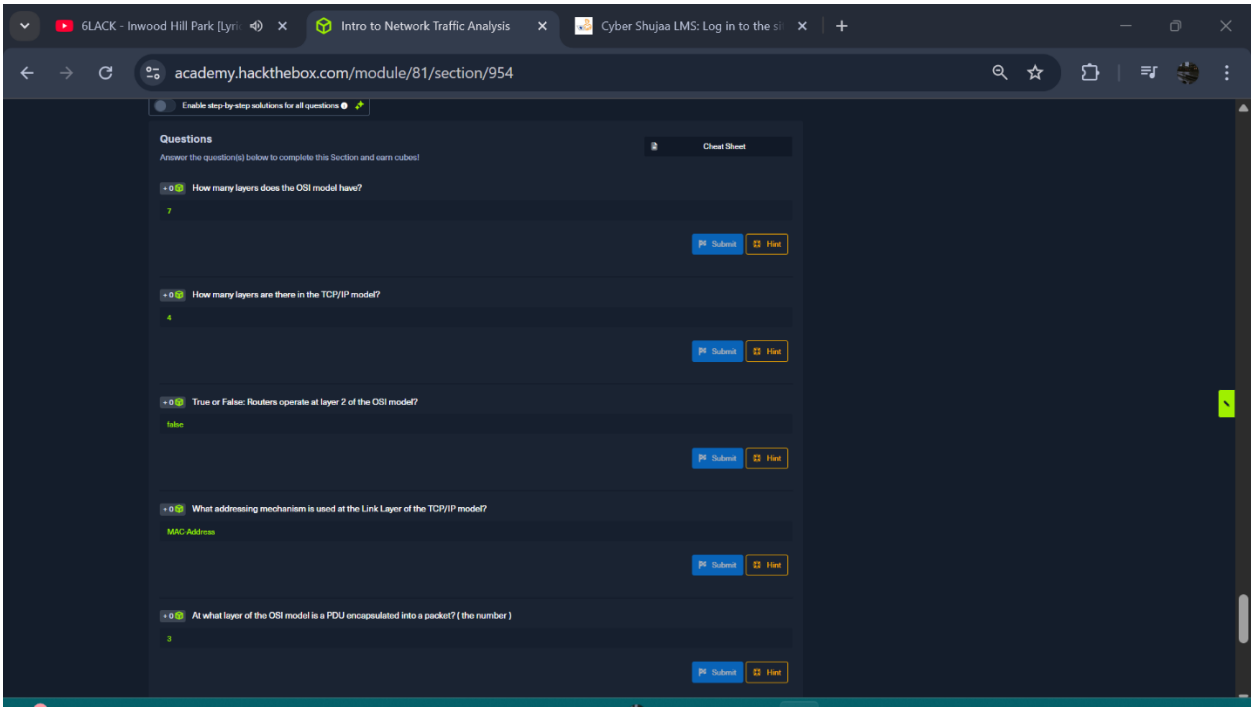
- FIN, ACK
- FIN, ACK
- ACK

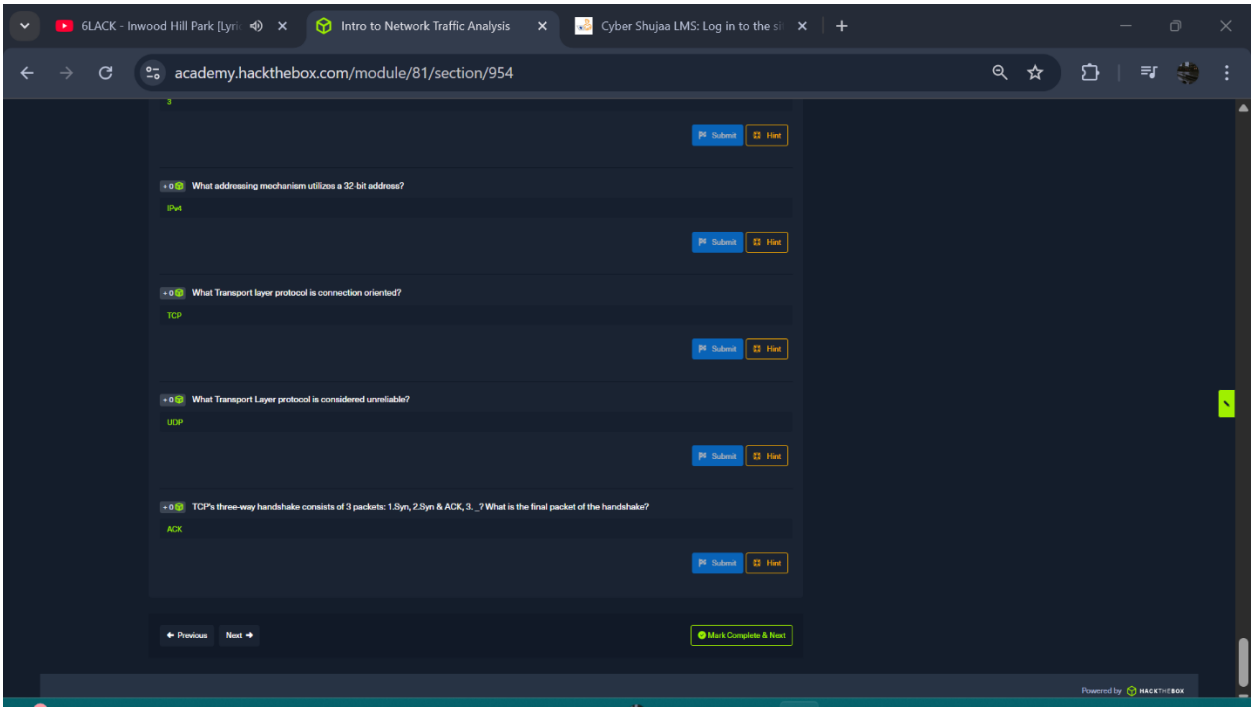
This structured approach helps maintain order and reliability in network communications, especially when data integrity is critical.

TCP Three-way Handshake.



HTB Answers.


6LACK - Inwood Hill Park [Lyri
Intro to Network Traffic Analysis
Cyber Shujaa LMS: Log in to the si
academy.hackthebox.com/module/81/section/954
Enable step-by-step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
How many layers does the OSI model have?
7
Submit
Hint
How many layers are there in the TCP/IP model?
4
Submit
Hint
True or False: Routers operate at layer 2 of the OSI model?
false
Submit
Hint
What addressing mechanism is used at the Link Layer of the TCP/IP model?
MAC Address
Submit
Hint
At what layer of the OSI model is a PDU encapsulated into a packet? (the number)
3
Submit
Hint


6LACK - Inwood Hill Park [Lyri
Intro to Network Traffic Analysis
Cyber Shujaa LMS: Log in to the si
academy.hackthebox.com/module/81/section/954
3
Submit
Hint
What addressing mechanism utilizes a 32-bit address?
IPv4
Submit
Hint
What Transport layer protocol is connection oriented?
TCP
Submit
Hint
What Transport Layer protocol is considered unreliable?
UDP
Submit
Hint
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
Powered by HACKTHEBOX

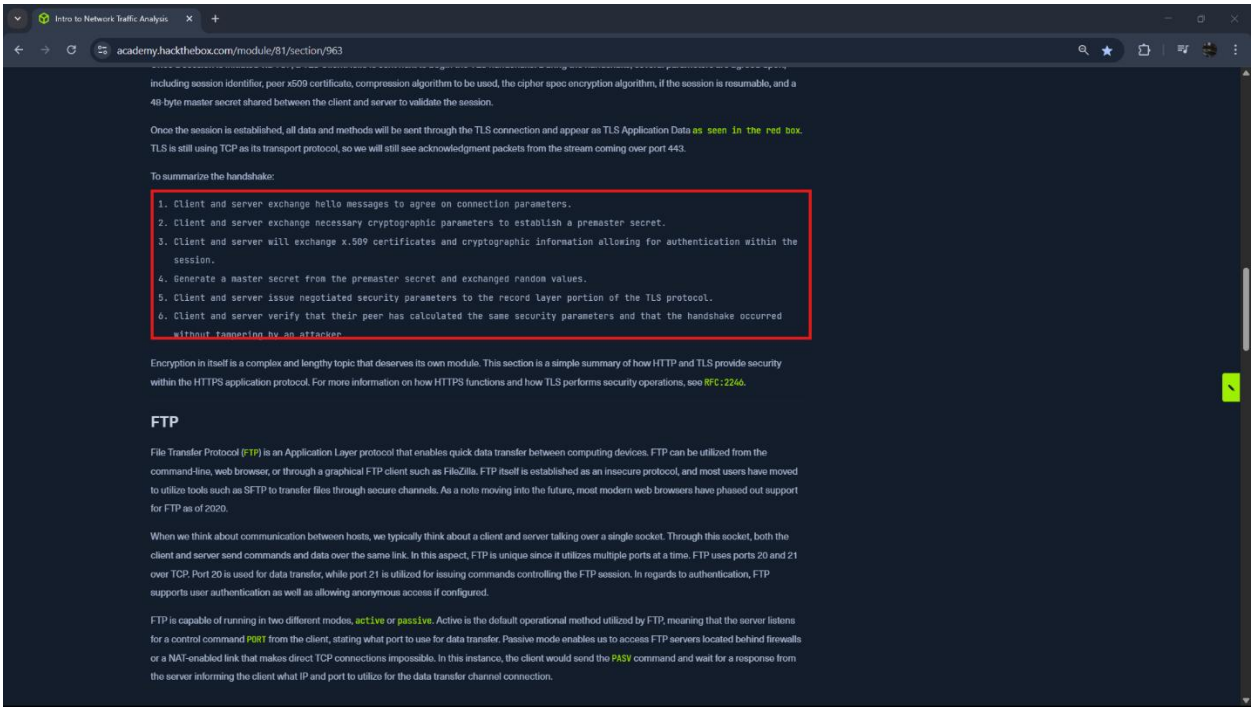
Networking Primer- Layers 5-7

HTTP and HTTPS.

Layers 5 to 7 of the OSI model handle how applications interact over networks. HTTP (Hypertext Transfer Protocol) is a stateless protocol used to request and transfer data like webpages and images. It works over TCP, typically on port 80 or 8000, and uses methods such as GET (to retrieve data), POST (to submit data), PUT (to update or create resources), and DELETE (to remove resources). GET and HEAD are required methods, while the rest are optional depending on the server.

HTTPS is the secure version of HTTP, running over port 443. It uses TLS (Transport Layer Security) to encrypt communications between a client and server, protecting against eavesdropping and attacks. The TLS handshake involves both sides exchanging cryptographic parameters and certificates, agreeing on how to secure the session, and establishing a shared master secret. After that, all data is encrypted, even though TCP is still used as the transport layer protocol.

HTTPS handshake.



FTP

I learned that FTP (File Transfer Protocol) is an Application Layer protocol used for transferring files between devices. While it's fast and accessible via tools like FileZilla or the command line,

it's not secure—most modern users prefer **SFTP** for encrypted transfers. I also noted that modern browsers stopped supporting FTP in 2020.

What makes FTP unique is that it uses **two ports (20 for data and 21 for commands)** and supports both **active and passive modes**. In active mode, the client tells the server where to send data, while in passive mode, the server tells the client which port to use—useful when behind firewalls. I also studied common FTP commands like:

- USER / PASS – for login
- PORT / PASV – to switch modes
- LIST, CWD, PWD, RETR, QUIT – for file navigation and transfers

Seeing a live FTP session helped me understand how command and data flows occur between two IPs.

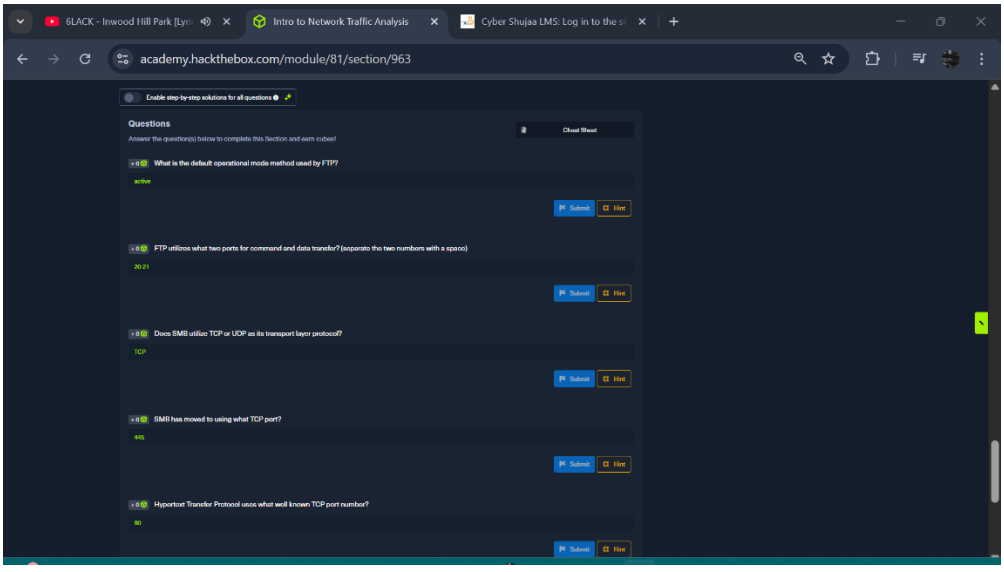
SMB

I also learned about **Server Message Block (SMB)**, a protocol mainly used in Windows environments to **share files, printers, and other resources** across networks. It requires user authentication and works over **port 445 (TCP)** or older methods like NetBIOS (ports 137–139).

SMB performs typical TCP functions like handshakes, but I learned that **repeated login failures in packet captures can indicate a brute-force attempt or unauthorized access**. This is often how attackers move laterally through networks using stolen credentials.

Overall, SMB is helpful for users but also a **common target for attackers**, making it important to monitor and analyze its traffic for unusual behavior.

HTB Answers.



The Analysis Process.

Network Traffic Analysis (NTA) is a flexible, repeatable process used to inspect network data for unusual or malicious activity. Its goal is to break down traffic into understandable pieces, spot anything out of the ordinary—like unauthorized remote access attempts—and compare current traffic to a baseline of normal network behavior.

NTA is crucial because it gives us visibility into what's happening on the network, helping admins and defenders catch issues early. By monitoring traffic over time, we can establish a baseline that makes spotting changes easier. Advanced setups combine NTA with tools like IDS/IPS, firewalls, and logging platforms (Splunk, ELK) to automate alerts on suspicious activity. Still, manual checks are important since attackers often find ways around automated defenses. The human eye remains key to detecting threats.

Besides security, NTA helps with daily troubleshooting, like identifying where connection problems lie or verifying that network protocols are working correctly.

Analysis Dependencies

Traffic capture can be **passive** (copying data without interfering) or **active** (capturing data inline with the traffic flow). Each method has different requirements:

- **Passive capture** needs permission, a mirrored port, and a capture tool. It listens silently without altering traffic.
- **Active capture** also needs permission and a capture tool but requires inline placement, like a network tap or a host with multiple NICs, because it directly intercepts the traffic.

Knowing what normal traffic looks like on your network is critical. Without a baseline, analyzing captured traffic is overwhelming—you'd have to inspect every conversation to separate normal from suspicious activity.

For example, if several hosts show high latency and strange files appear, a capture might reveal unusual connections (like two user PCs communicating over ports normally used for web or SMB traffic). This behavior would stand out against the baseline, quickly flagging a potential breach and allowing a fast response.

Practice Analysis.

Network traffic analysis is a flexible and dynamic process that varies depending on the specific network problem and the visibility you have into the network. It's not an exact science but can

be approached systematically by breaking it down into key types of analysis and following a workflow to guide your investigation.

Analysis in Practice

Descriptive Analysis

Descriptive analysis is the foundation of any effective data or network investigation. It involves summarizing and characterizing the data set by examining individual features to understand what's happening. This step is crucial because it helps identify any errors or anomalies in data collection, as well as outliers that might indicate suspicious or unusual activity. Example:

- The target is multiple hosts potentially downloading malicious files from the domain bad.example.com.
- The time window is the last 48 hours plus an additional 2 hours forward, to cover possible delayed activity.
- Supporting details include specific suspicious filenames such as superbad.exe and new-crypto-miner.exe.
- The network segment under consideration is 192.168.100.0/24.
- The protocols used during these downloads were HTTP and FTP.

By framing these parameters, descriptive analysis helps set clear boundaries for the investigation and ensures focus on relevant data. This approach forms the basis of a systematic workflow: determining the issue, what to look for, when it happened, and where to search.

Diagnostic Analysis

Diagnostic analysis aims to uncover the root cause of suspicious network behavior. The process begins by capturing live traffic from the 192.168.100.0/24 network and retrieving historical PCAP or NetFlow data from the SIEM. The captured traffic is then filtered to isolate relevant HTTP and FTP activity, especially those involving suspicious files like superbad.exe and new-crypto-miner.exe. Key steps include:

- **Filtering out baseline traffic** to focus on unusual file transfers and GET requests.
- **Inspecting FTP and HTTP traffic** to identify and reconstruct file transfers.
- **Detecting lateral movement**, which may indicate internal spreading of the malicious payload.

By focusing only on the critical components of traffic, this analysis validates whether a compromise has occurred and explains how. Ultimately, it helps determine the true nature of the incident and informs the appropriate response.

Predictive Analysis

Predictive analysis focuses on using both historical and current network data to anticipate future incidents. It builds on the findings from descriptive and diagnostic analyses to identify patterns, trends, or anomalies that may indicate upcoming threats or deviations from normal behavior.

A critical part of this process is thorough documentation. Taking detailed notes during the investigation helps track when traffic was captured, which hosts were suspicious, and what conversations involved the malicious files. These notes should include timeframes, packet numbers, and a clear summary of findings. This ensures that all insights are well-organized and ready for decision-makers to review.

Once data is documented, it should be compared against baseline traffic patterns and known threat signatures (such as those from malware or hacking tools). This comparison helps predict if and how the issue might spread or reoccur.

In essence, predictive analysis is about using what we've discovered to forecast potential risks. It enables us to provide actionable insight and inform strategic responses that help prevent future compromises.

Prescriptive Analysis

Prescriptive analysis is the final and most action-oriented phase of the data investigation process. Its purpose is to use insights derived from previous analysis stages—descriptive, diagnostic, and predictive—to determine the best steps to eliminate a problem and prevent it from recurring.

At this stage, your findings should drive decisive action. If multiple hosts were found downloading a malicious file from bad.example.com, you now recommend actions like isolating affected machines, blocking communication with the malicious domain, or implementing stricter firewall rules. These actions aren't guesses—they're based on evidence uncovered during the investigation.

Key Components of an Effective Analysis

1. Know Your Environment: Maintain updated asset inventories and network maps
2. Placement Is Key: Capture traffic **as close to the source as possible**
3. Persistence: Stay vigilant — even low-frequency communication (e.g., once per day) may be part of an attack (like C2).

Analysis Approach and Practical Tips

- **Start with Common Protocols**: HTTP/S, FTP, Email, TCP/UDP.

- **Filter Out Irrelevant Data**: Focus only on what's related to the incident.
- **Examine Internal Communications**: Hosts shouldn't normally talk to each other often — peer-to-peer traffic can be a red flag.
- **Review Remote Access Protocols**: SSH, RDP, Telnet. Cross-check with your org's security policy.
- **Look for Patterns**: Repetitive behaviors (e.g., same time every day) may indicate malware or C2 activity.
- **Spot Unique Events**:
    - ✓ Unusual port bindings
    - ✓ New/different User-Agent strings
    - ✓ Changes in host access behavior
- **Use a Second Set of Eyes**: Collaboration can help catch what you might miss.

Toolset Beyond Wireshark & Tcpdump

- **Snort**: Intrusion detection.
- **Security Onion**: Full-stack visibility.
- **Firewalls & SIEMs**: Add context and enrich data for better decisions.

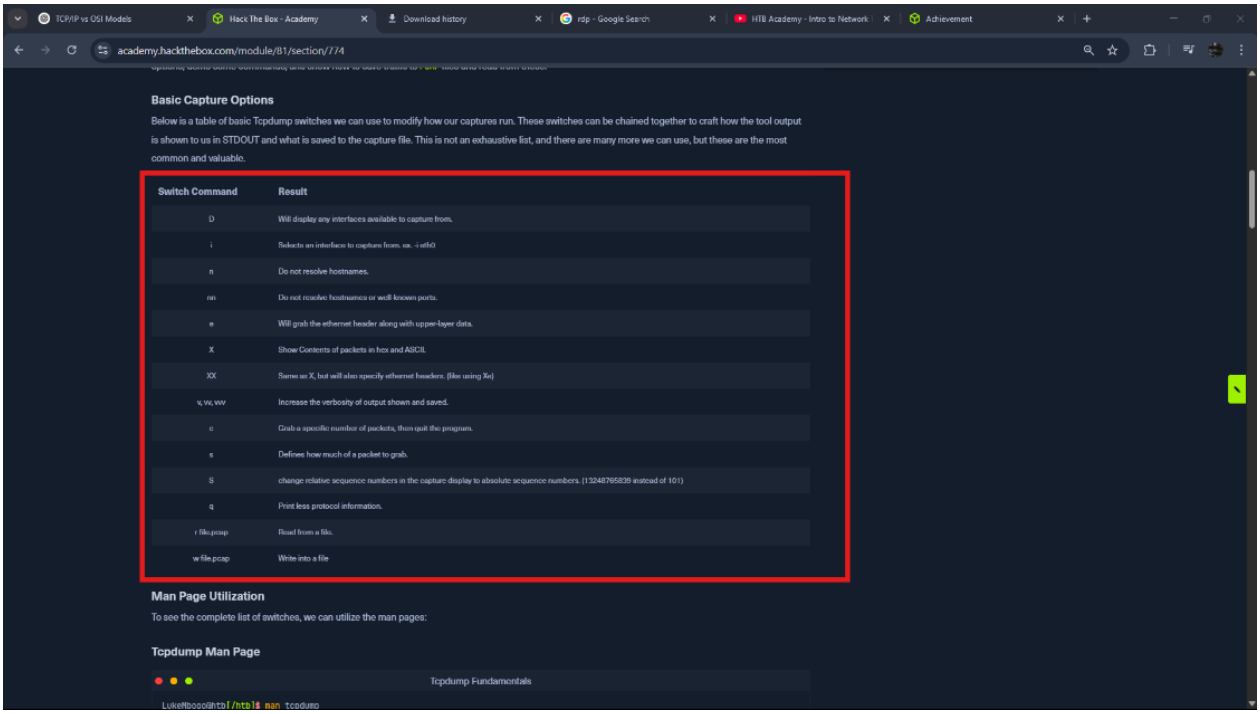
Tcpdump Fundamentals.

During this learning session, I explored **Tcpdump**, a powerful command-line packet sniffer used to capture and analyze network traffic. It works directly through the terminal without requiring a graphical interface, which makes it ideal for remote use over SSH. Although Tcpdump can feel intimidating at first due to its wide range of filters and options, understanding the core functionalities makes it much easier to work with. Tcpdump operates using the **pcap** or **libpcap** libraries and relies on **promiscuous mode** to monitor all traffic on a network interface—not just the packets meant for the local host. This makes it a great tool for deep packet inspection in local area networks.

The tool is native to most Unix-like systems (like Linux, AIX, BSD, and Solaris), and often comes pre-installed. Running it typically requires **root privileges**, so using sudo is essential. On my system, I confirmed that Tcpdump was installed using the which tcpdump command and verified its version using sudo tcpdump --version. My current version was 4.9.3, using libpcap 1.9.1. I also learned that while Windows once had a Tcpdump equivalent called **WinDump**, its support has ended. A modern workaround is to use the Windows Subsystem for Linux (WSL)

with a distribution like Ubuntu or Parrot OS to run Tcpdump and other Linux-native tools right within Windows.

Switch commands.



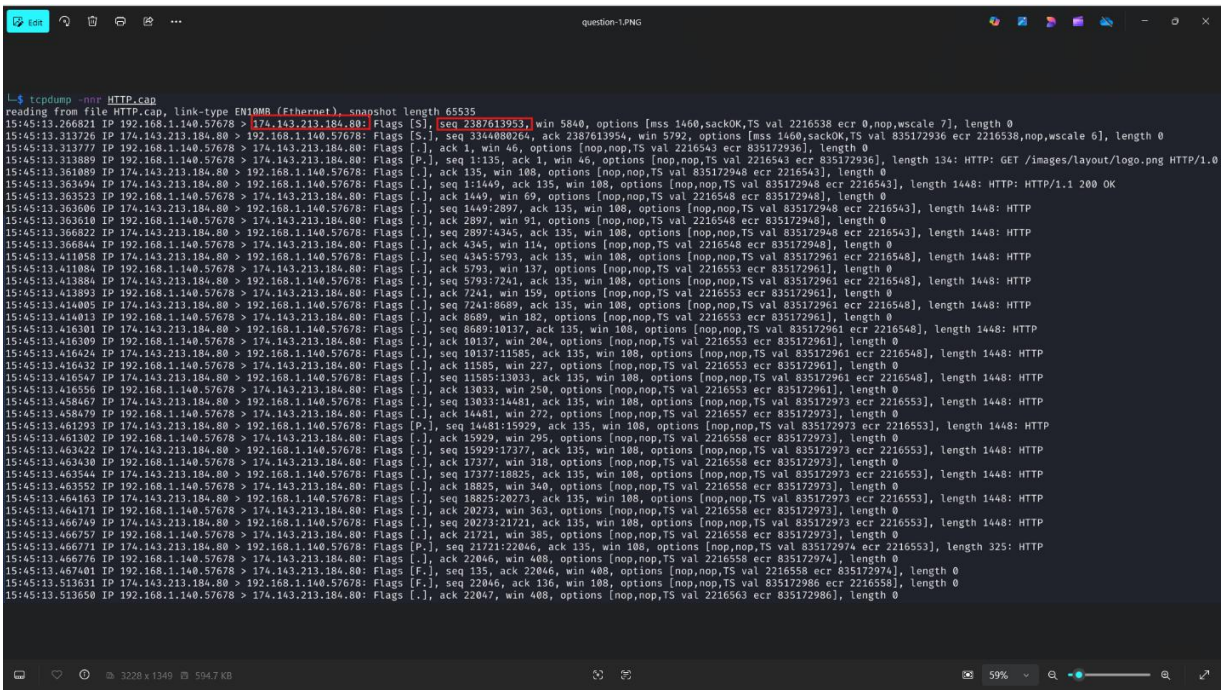
Packet Analysis.

As part of the exercise, I was provided with a PNG file question-1.png containing a screenshot of captured network traffic. I closely examined the packets and headers shown in the image to identify the server and analyze the TCP sequence numbering.

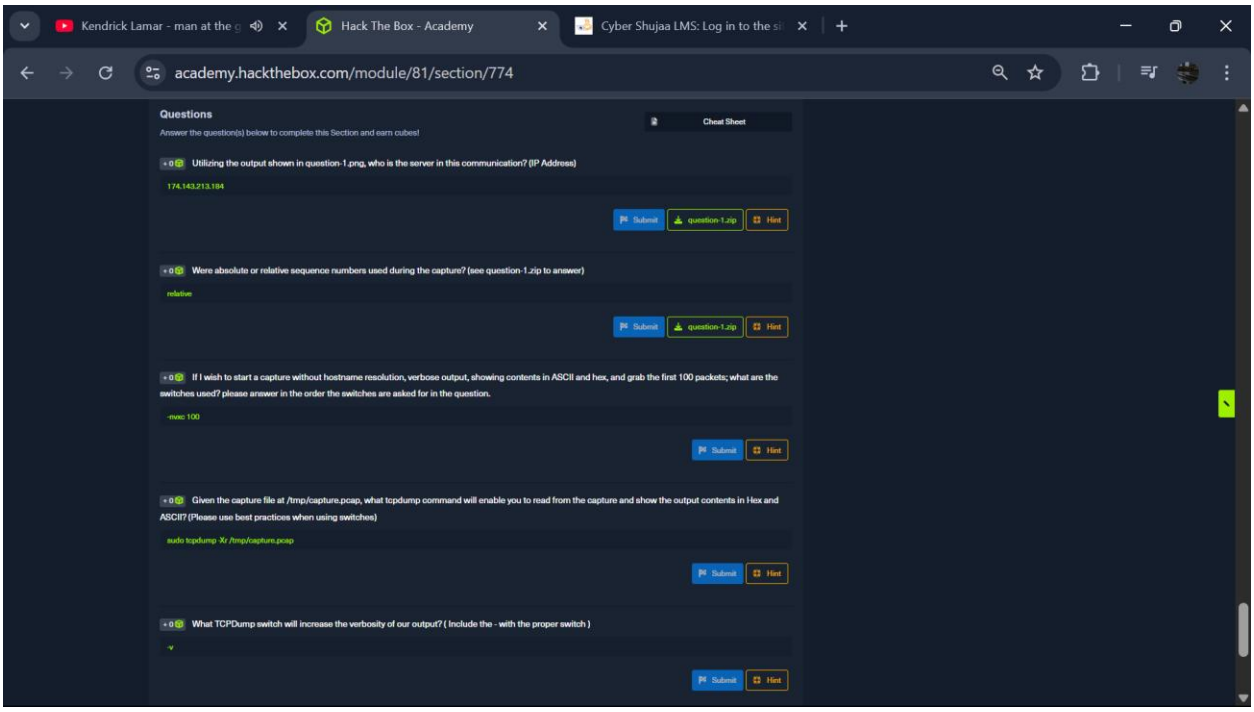
From the IP address information shown, I determined that **174.143.213.184** was the **server** in the communication. This conclusion was based on observing the **direction of the TCP handshake and data flow**, where this IP responded to connection initiation, a behavior typical of a server.

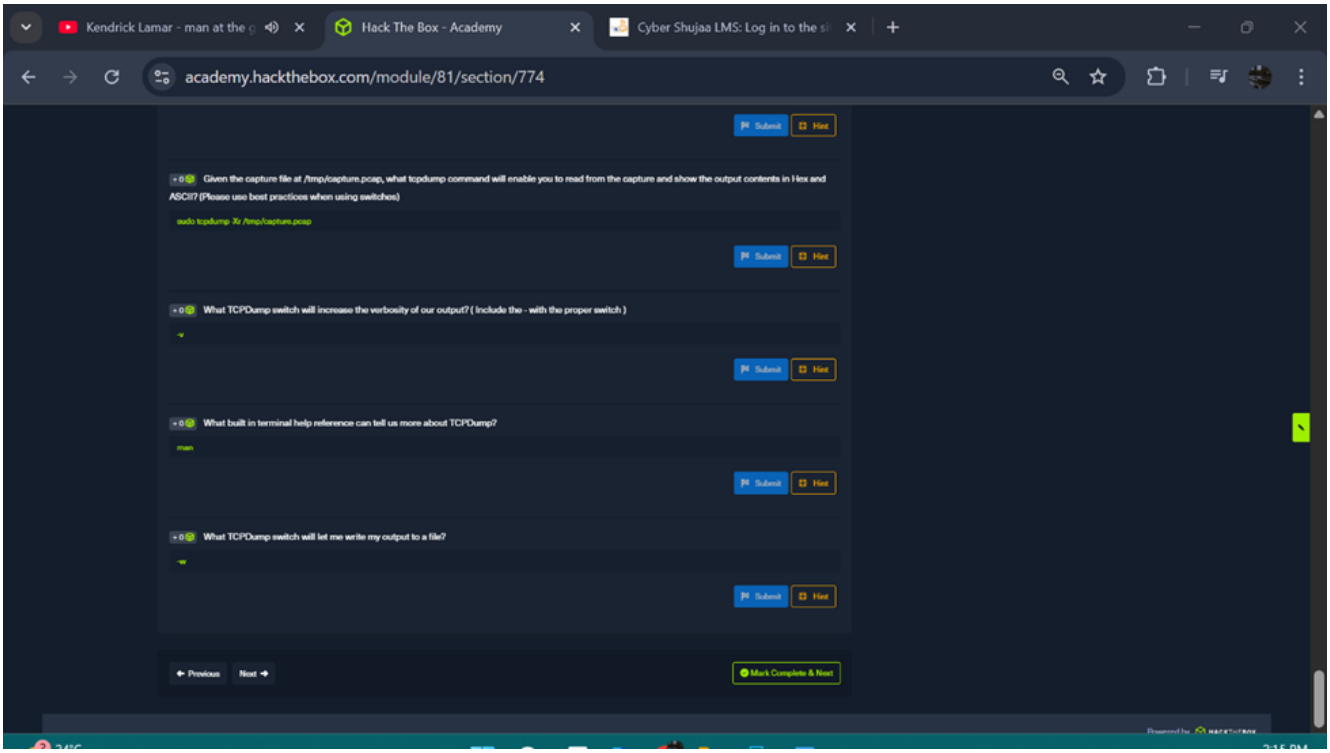
Additionally, I noted that the **sequence numbers were presented in a simplified format**, indicating that **relative sequence numbers** were being used. This is common in Wireshark-style displays where sequence numbers are adjusted to start at zero for easier interpretation during analysis.

This task helped reinforce my ability to interpret packet data visually and understand key aspects of TCP communication like identifying roles and evaluating sequence number formats.


question-1.PNG
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

HTB Answers.


Kendrick Lamar - man at the g
Hack The Box - Academy
Cyber Shujaa LMS: Log in to the si
academy.hackthebox.com/module/81/section/774
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
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
Were absolute or relative sequence numbers used during the capture? (see question-1.zip to answer)
relative
Submit
question-1.zip
Hint
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
Submit
Hint
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
Submit
Hint
What TCPDump switch will increase the verbosity of our output? (Include the - with the proper switch)
-v
Submit
Hint



Fundamentals Lab.

This lab helped me get hands-on experience with **tcpdump** while becoming more comfortable navigating and using tools in the terminal. I practiced core skills like capturing traffic, reading from and writing to .pcap files, and using essential tcpdump switches.

The goal was to simulate a real-world task as a new network administrator—capturing traffic within the local broadcast domain to validate that our setup works. I ensured the necessary tools were installed and confirmed I could successfully capture and analyze packets.

This kind of traffic analysis is crucial for identifying how hosts communicate, spotting anomalies, and even detecting backdoors or breaches using filters and patterns. Overall, the lab laid a strong foundation for deeper network traffic analysis.

Tcpdump Packet Filtering**.**

Using advanced filtering in tcpdump helps minimize captured traffic and reduce disk usage, making analysis faster and more focused.

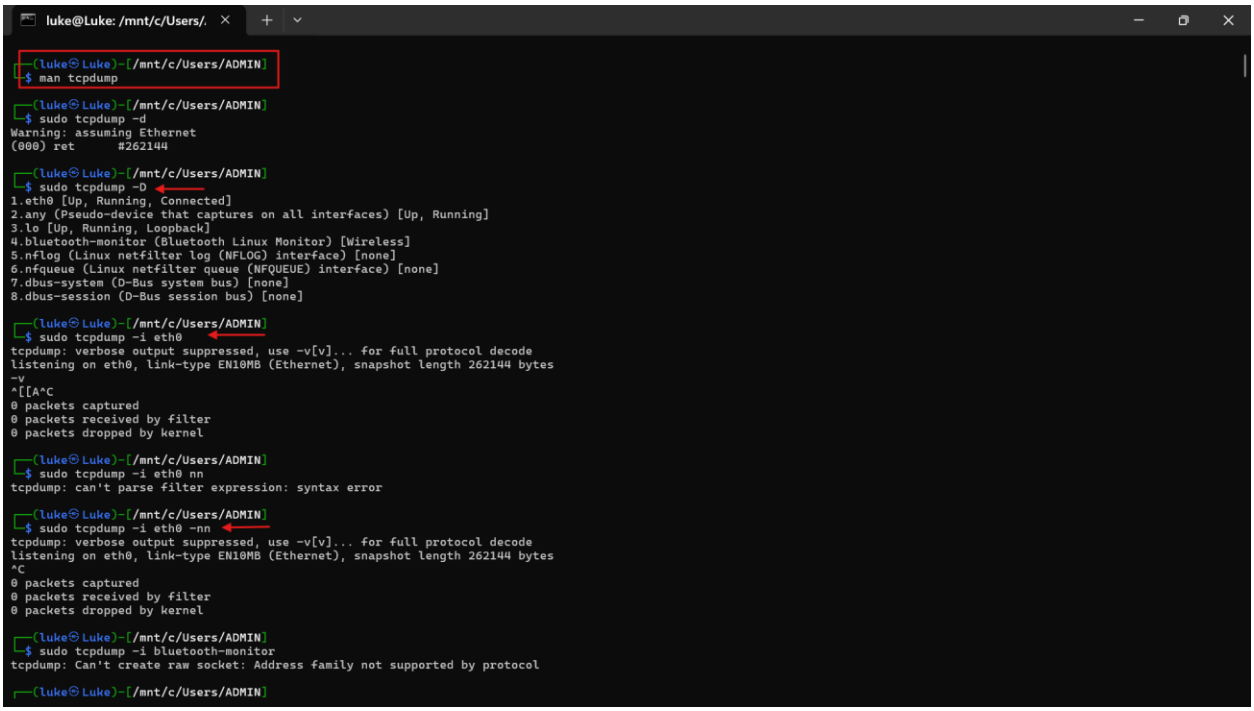
Common Filters.

Filter	Purpose
host	Filters packets where the specified IP is either source or destination.
src / dst	Designate specific source or destination IP, network, or port.
net	Filters packets involving a specific network (use CIDR, e.g. /24).
proto	Filters by protocol (e.g., tcp, udp, icmp, ether).
port	Filters packets by source or destination port.
portrange	Specify a range of ports, e.g., 0-1024.
less / greater	Filters by packet size (e.g., greater 128).
and / &&	Combine multiple filters (both conditions must match).
or	Match if **either** condition is true.
not	Exclude packets matching a condition.

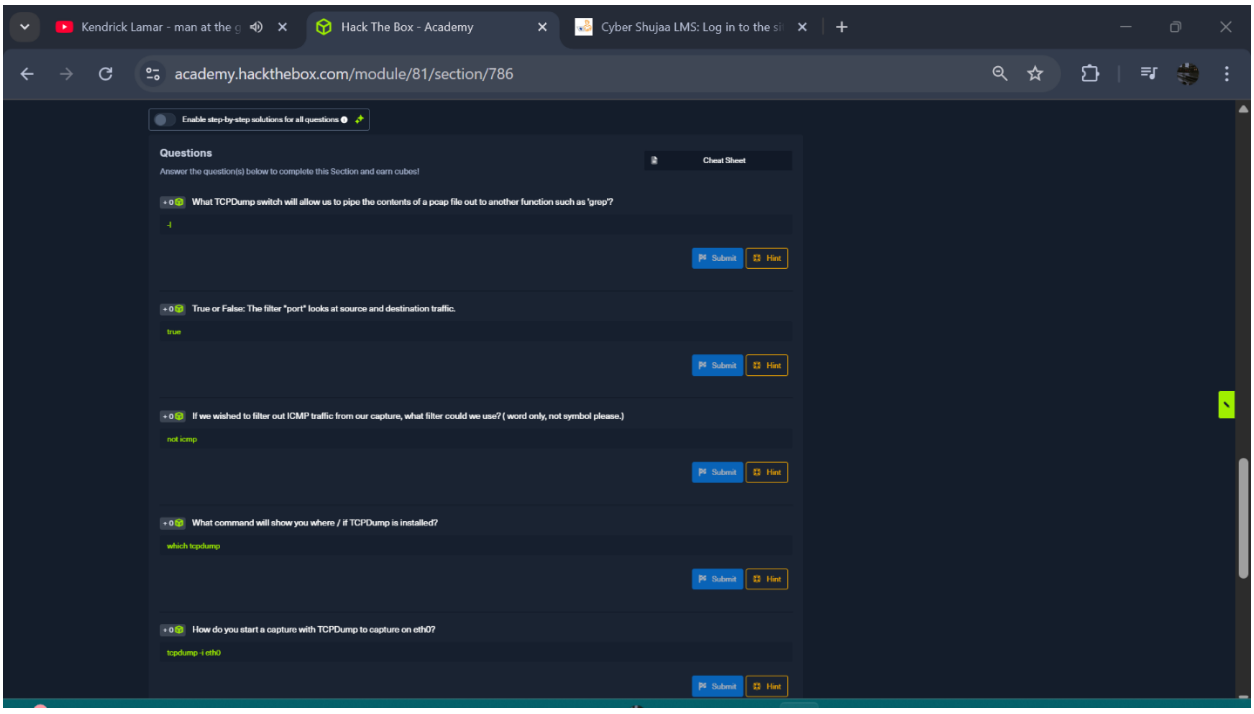
Task.

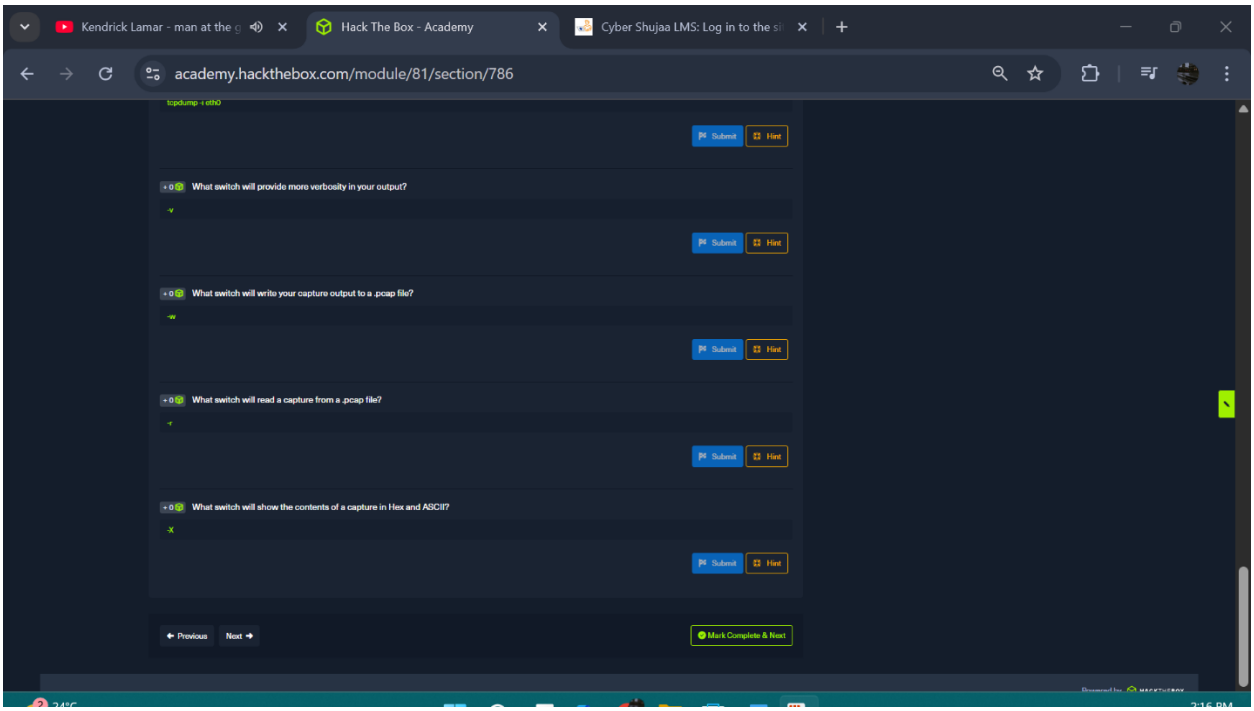
In this section, I learned how to use filters in network traffic analysis tools to focus on specific data. For example, to view traffic coming from or going to a particular IP address, such as 10.10.20.1, the filter host 10.10.20.1 can be used. This allows me to isolate communication involving that host easily. Additionally, I discovered that filters can combine multiple conditions using logical operators like or, which lets me capture traffic that meets either one of two options. This is useful for broadening or narrowing the scope of traffic I want to analyze.

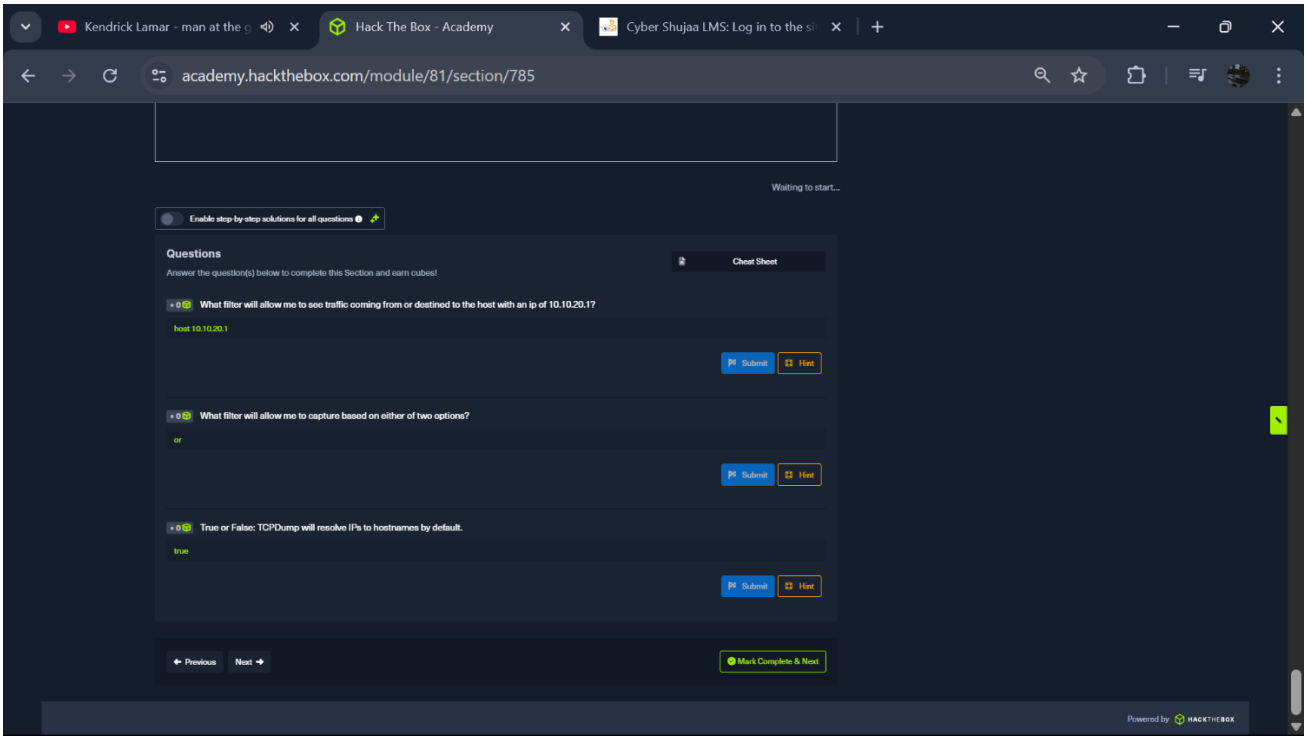
Lastly, I confirmed that the TCPDump tool resolves IP addresses to hostnames by default, which can make reading captures more intuitive by showing domain names instead of numeric IPs. Overall, these filtering techniques are essential for efficiently analyzing network data and troubleshooting network issues.


luke@Luke: /mnt/c/Users/.
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]
└─$ man tcpdump
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]
└─$ sudo tcpdump -d
Warning: assuming Ethernet
(000) ret #262144
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]
└─$ sudo tcpdump -D
1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]
└─$ sudo tcpdump -i eth0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
-v
^[[A^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]
└─$ sudo tcpdump -i eth0 nn
tcpdump: can't parse filter expression: syntax error
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]
└─$ sudo tcpdump -i eth0 -nn
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]
└─$ sudo tcpdump -i bluetooth-monitor
tcpdump: Can't create raw socket: Address family not supported by protocol
┌──(luke㉿Luke)-[/mnt/c/Users/ADMIN]

HTB Answers.


Kendrick Lamar - man at the g
Hack The Box - Academy
Cyber Shujaa LMS: Log in to the si
academy.hackthebox.com/module/81/section/786
Enable step-by-step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
What TCPDump switch will allow us to pipe the contents of a pcap file out to another function such as 'grep'?
-l
Submit
Hint
True or False: The filter "port" looks at source and destination traffic.
true
Submit
Hint
If we wished to filter out ICMP traffic from our capture, what filter could we use? (word only, not symbol please.)
not icmp
Submit
Hint
What command will show you where / if TCPDump is installed?
which tcpdump
Submit
Hint
How do you start a capture with TCPDump to capture on eth0?
tcpdump -i eth0
Submit
Hint

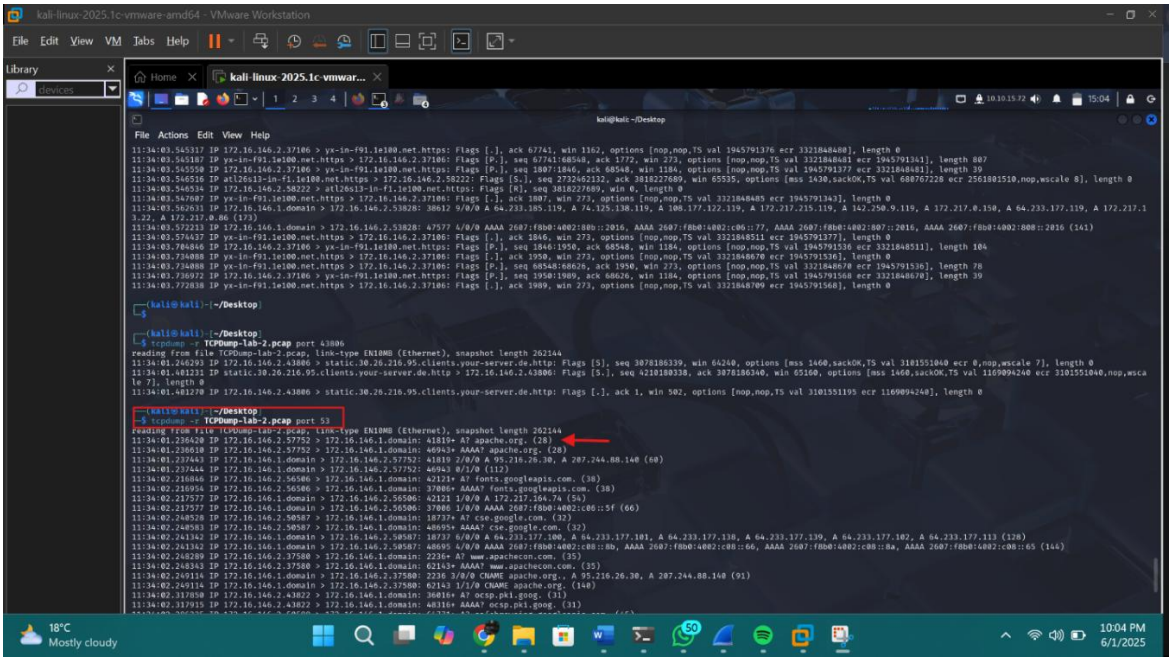

Kendrick Lamar - man at the g
Hack The Box - Academy
Cyber Shujaa LMS: Log in to the si
academy.hackthebox.com/module/81/section/786
tcpdump -i eth0
Submit
Hint
What switch will provide more verbosity in your output?
-v
What switch will write your capture output to a .pcap file?
-w
What switch will read a capture from a .pcap file?
-r
What switch will show the contents of a capture in Hex and ASCII?
-X
Previous
Next
Mark Complete & Next


Kendrick Lamar - man at the g
Hack The Box - Academy
Cyber Shujaa LMS: Log in to the si
academy.hackthebox.com/module/81/section/785
Waiting to start...
Enable step-by-step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
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
Submit
Hint
What filter will allow me to capture based on either of two options?
or
True or False: TCPDump will resolve IPs to hostnames by default.
true
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX

Interrogating Network Traffic With Capture and Display Filters

In this lab, I gained practical experience interrogating network traffic using TCPDump and applying packet filters to analyze a captured .PCAP file. The main goal was to understand how to isolate and interpret different types of network communications, focusing on DNS and HTTP/S servers in the local network.

I started by reading the capture file without any filters to get an overall view of the traffic. This initial step showed various protocols and ports in use, primarily DNS (port 53), HTTP (port 80), and HTTPS (port 443). Recognizing these common protocols helped me target filters more effectively.

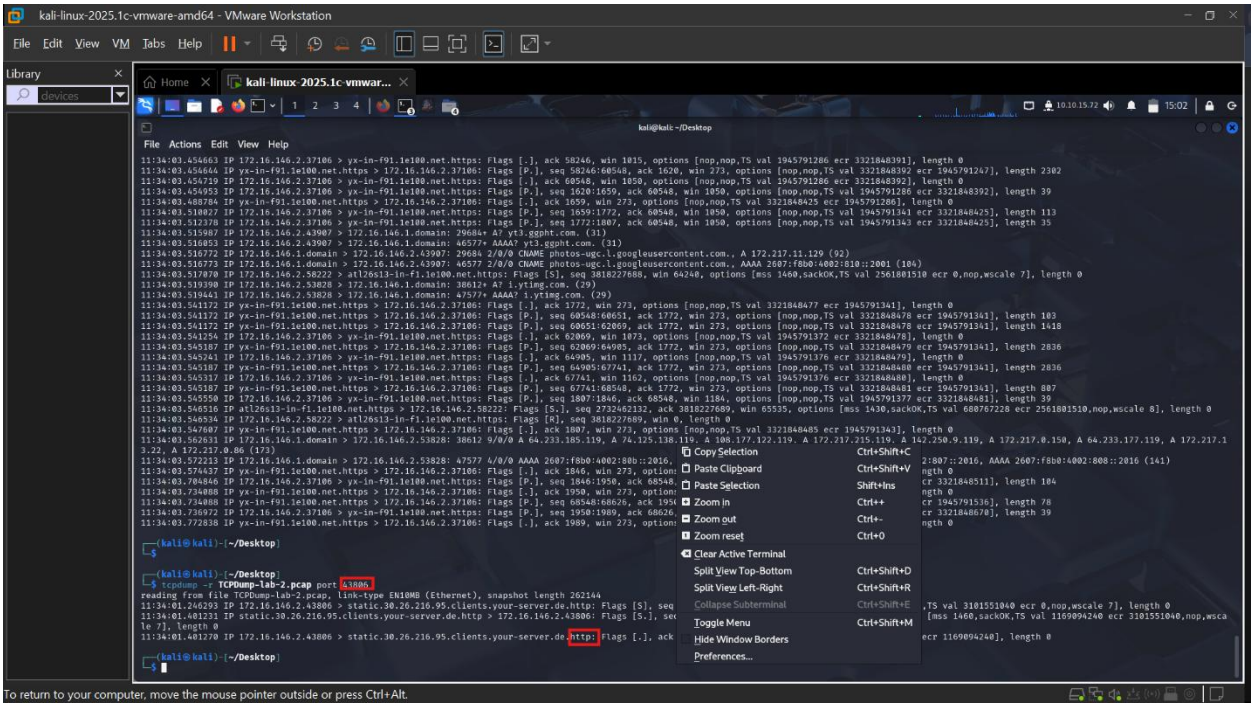


Next, I identified conversations between hosts and servers by observing connection patterns. Servers typically responded on well-known ports, while clients used random high ports. I learned to analyze TCP three-way handshakes and how enabling absolute sequence numbers can clarify conversation tracking.

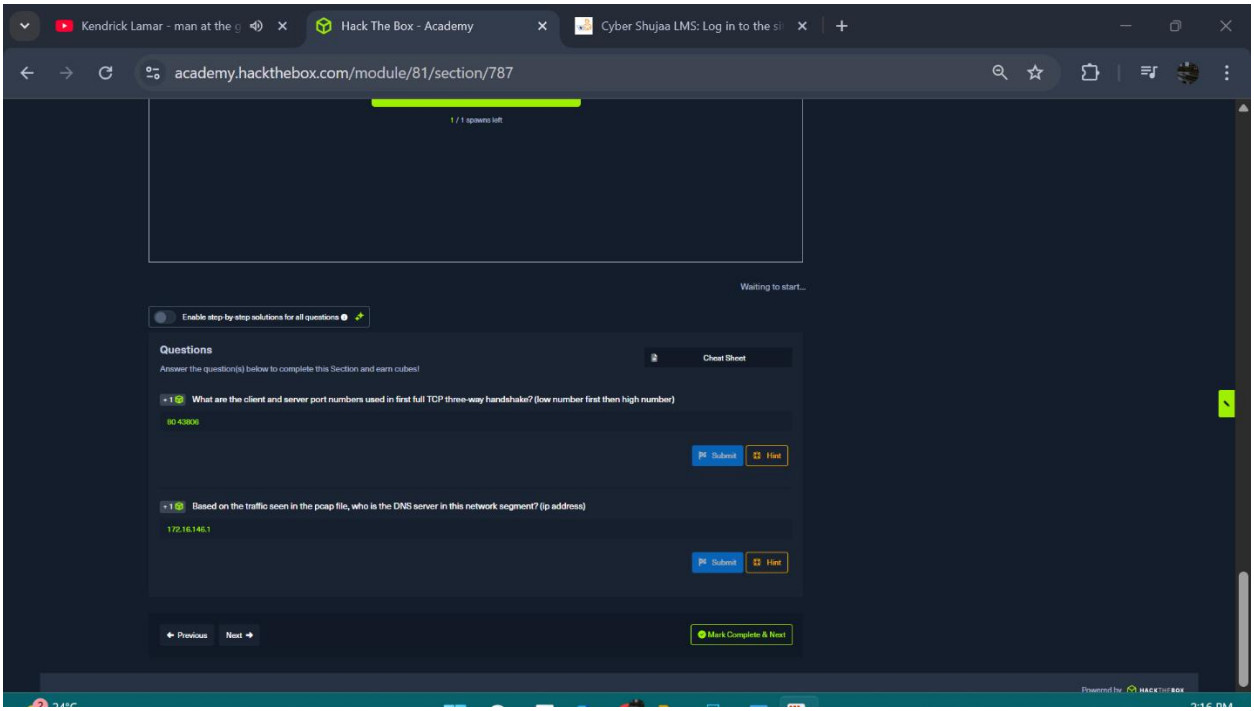
Diving deeper, I examined timestamps of conversations, located IP addresses from DNS responses, and confirmed protocols used in initial connections. Using filters, I isolated DNS traffic and discovered which DNS servers handled requests, what domain names were queried, and what DNS record types appeared. I learned that A records provide IP addresses corresponding to domain names. Filtering for TCP traffic on ports 80 and 443 allowed me to

focus on webserver activity. I identified common HTTP request methods and typical response codes, gaining insights into the nature of web traffic on the network.

Lastly, I attempted to analyze the first webserver conversation at the application level by examining HTTP response content in ASCII and Hex formats. Although somewhat challenging with TCPDump alone, this step revealed useful details about the webserver and the services it was running. Overall, this lab expanded my understanding of network traffic analysis using filters and hands-on packet inspection. It reinforced the importance of methodical filtering to uncover relevant network behaviors and equipped me with skills to explore network data more confidently. I look forward to applying these techniques to capture and analyze traffic in other environments, including my home network.



HTB Answers.



Analysis with Wireshark

Wireshark is a powerful, free, and open-source network traffic analyzer similar to tcpdump but with a graphical user interface (GUI). It supports multiple platforms and can capture live data from various interfaces such as WiFi, USB, and Bluetooth, saving this data in different formats. What makes Wireshark stand out is its ability to perform deep packet inspection and provide detailed analysis across hundreds of protocols, offering insights that other tools might miss.

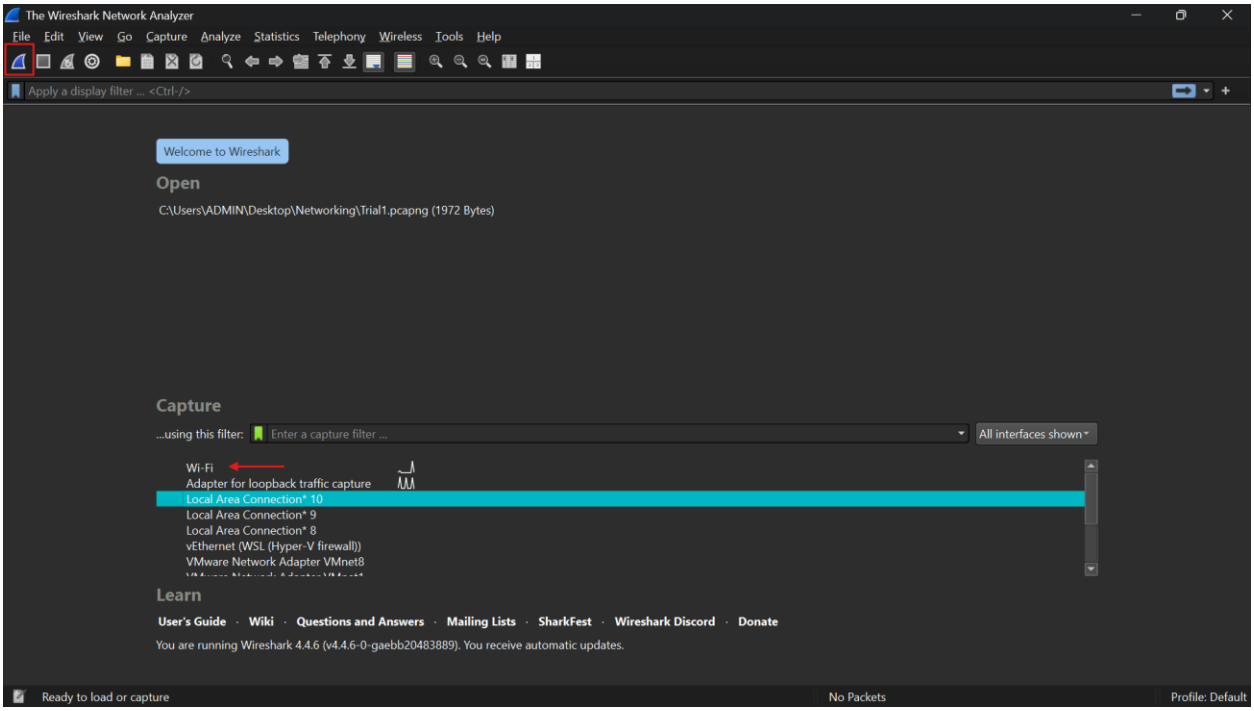
While Wireshark is best known for its GUI, it can also be used from the command line, which is helpful when working on systems without a graphical environment. This flexibility makes it a versatile tool for network analysis across different operating systems.

Wireshark supports a wide range of network types including Ethernet, 802.11 wireless, Bluetooth, and more. It also includes decryption capabilities for protocols like IPsec, SSL/TLS, and WPA2, enhancing its usefulness for security analysis.

To use Wireshark effectively, the system requirements vary by platform. On Windows, it needs the Universal C Runtime, a modern processor, sufficient RAM and disk space, and a supported network card. On Linux and UNIX-like systems, Wireshark runs smoothly with similar hardware needs, and prebuilt packages are available for easy installation.

Overall, learning about Wireshark highlighted how its advanced features and broad support make it an essential tool for detailed network traffic capture and analysis, whether through GUI or command line.

Wireshark GUI.



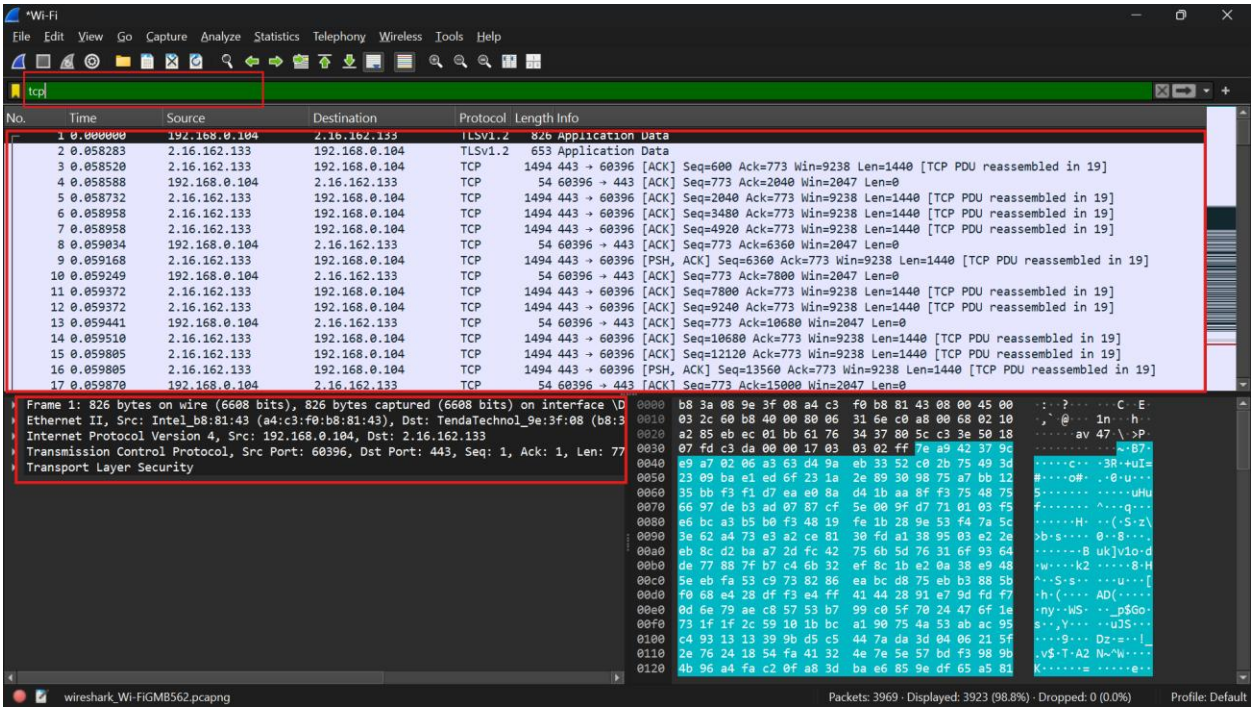
After working with packet capture from the command line, I got the chance to explore Wireshark's graphical interface. This tool gives a much more visual and detailed look into packet data, and understanding its layout made it easier for me to analyze network traffic.

The GUI is made up of three main panes:

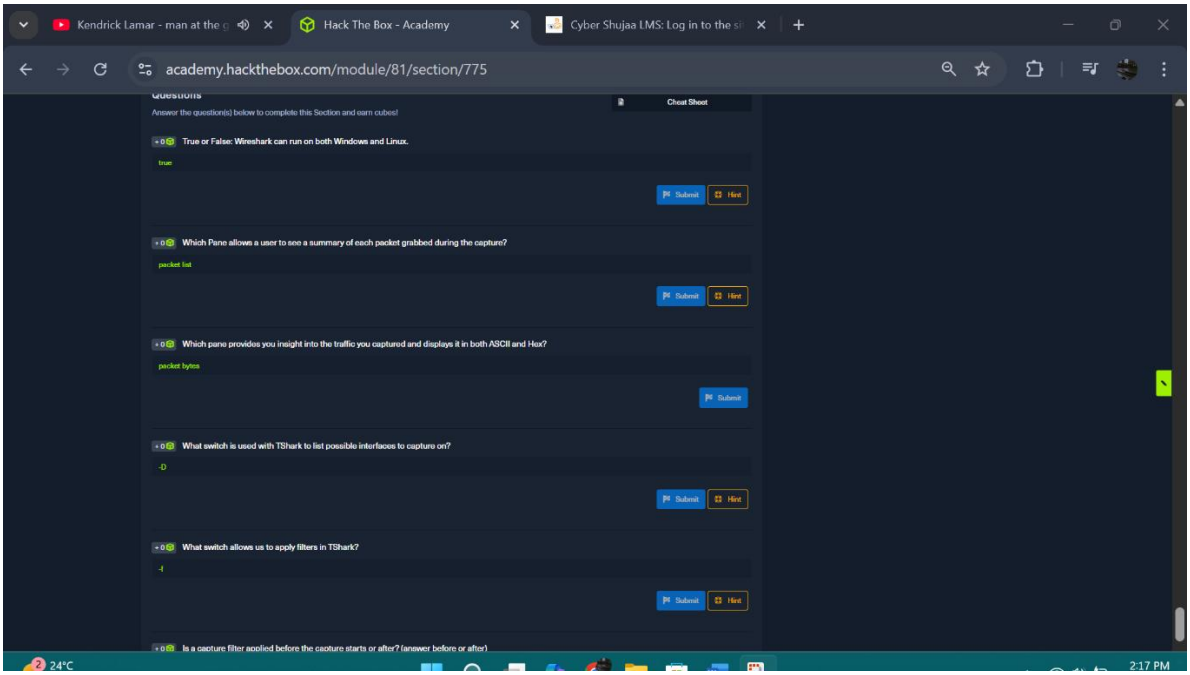
1. **Packet List** – This is the top section that displays a summary of each captured packet. I learned how to view key details like the packet number, timestamp, source and destination IPs, protocol used (like TCP or DNS), and additional information based on the protocol type. It's also possible to customize the columns depending on what details I want to focus on.

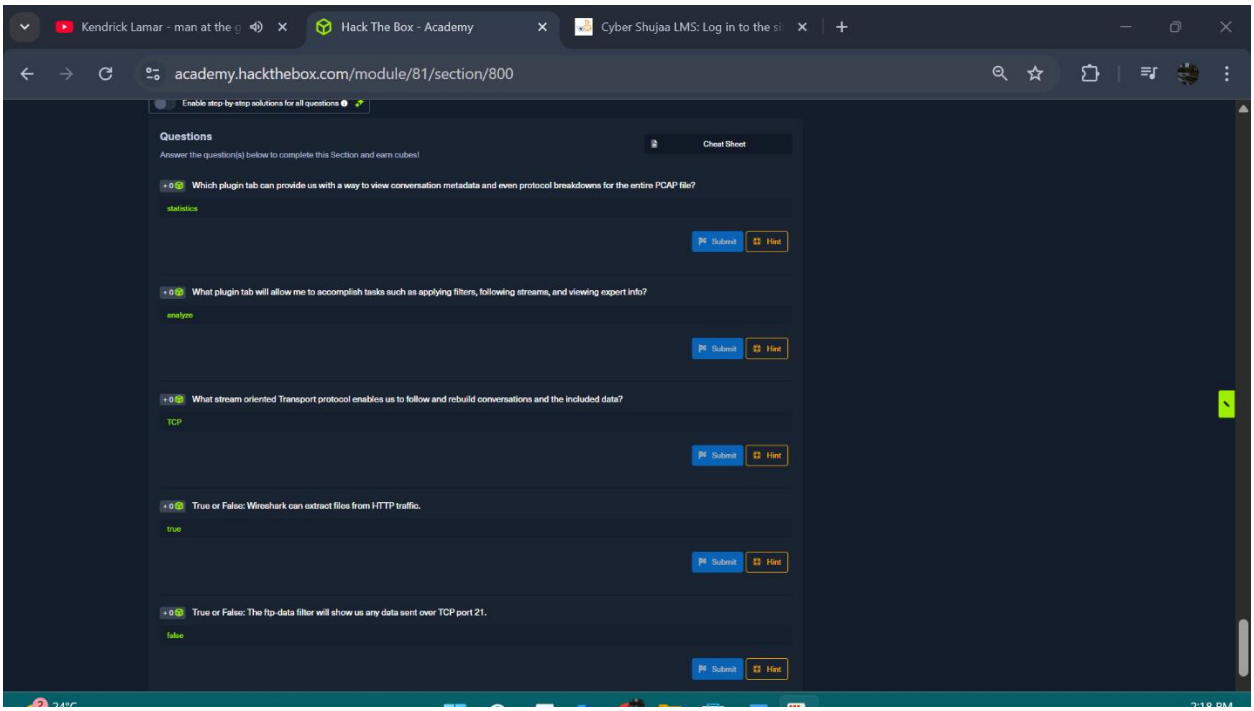
2. **Packet Details** – This middle section breaks down the packet contents layer by layer, following the OSI model. What stood out to me is that Wireshark presents the data from lower layers (like Ethernet) at the top, and higher layers (like HTTP) at the bottom. This view helps drill deeper into each protocol within a packet.

3. **Packet Bytes** – The bottom pane shows the raw data of the selected packet in both hexadecimal and ASCII formats. It's a useful way to verify that the decoded information in the details pane matches the actual data being transmitted. I found it especially helpful that when you select a field in the details pane, it gets highlighted in this view, making it easier to track exactly where that data is in the raw output.



HTB Answers.





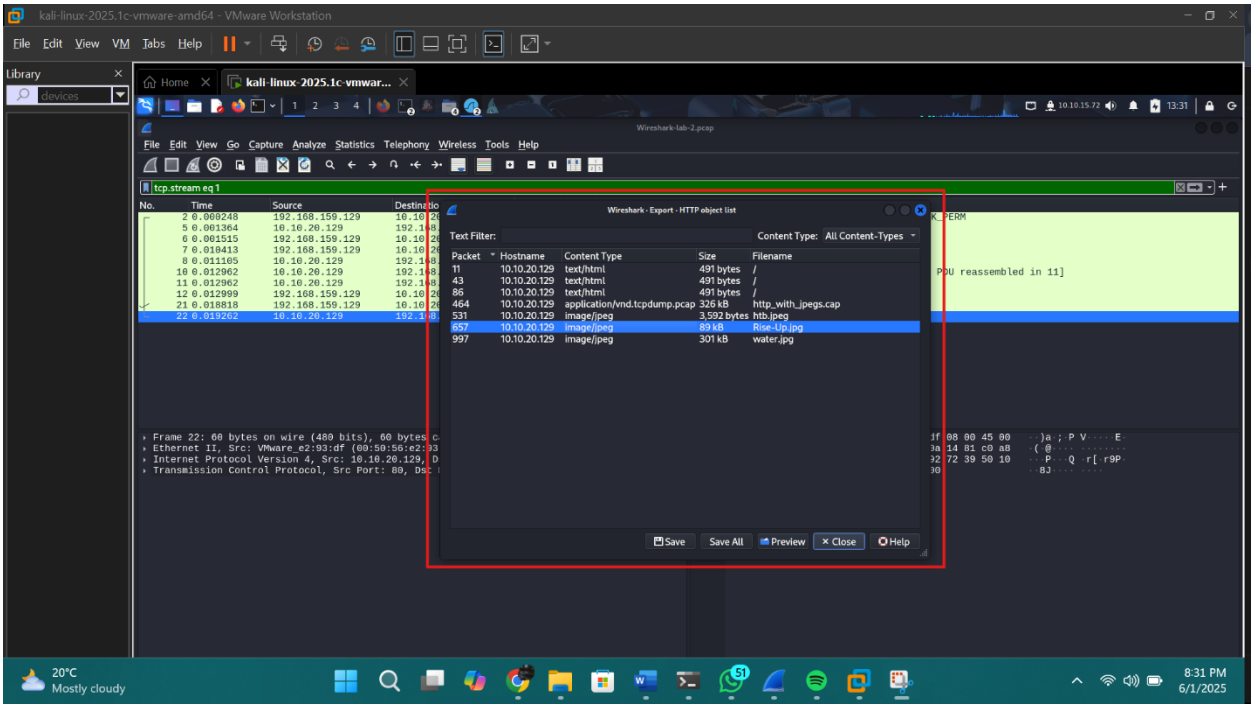
Packet Inception, Dissecting Network Traffic With Wireshark.

In this lab, I explored how to analyze captured HTTP traffic using Wireshark and extract embedded files, specifically images, from a .pcap file. The exercise was centered on investigating a pre-captured file (Wireshark-lab-2.pcap) and identifying any transferred images that might be used as evidence in a security context.

Task.

I began by opening the .pcap file in Wireshark. It contained about 1171 packets, but only a small portion—less than 20—were HTTP-related. To narrow the focus, I applied an HTTP display filter (http) which immediately helped isolate relevant web traffic from all the noise. Through the filtered view, I could clearly identify several GET requests and 200 OK responses. These responses confirmed that certain resources were successfully transferred from a server to a client. I selected one of the 200 OK packets and used the **Follow TCP Stream** option to view the complete data exchange in that specific communication. This helped verify that a file transfer did occur.

To determine if the transferred data included any images, I changed the display filter to http && image-jfif, which specifically targeted JPEG image files embedded in HTTP traffic. This led me to a few packets indicating the presence of image data in JPEG format. Finally, I used Wireshark's **Export Objects** feature to extract these images from the HTTP stream (File → Export Objects → HTTP). This allowed me to save the JPEG files locally for further analysis. These images could then be examined more closely to see if they contained any hidden or suspicious content.

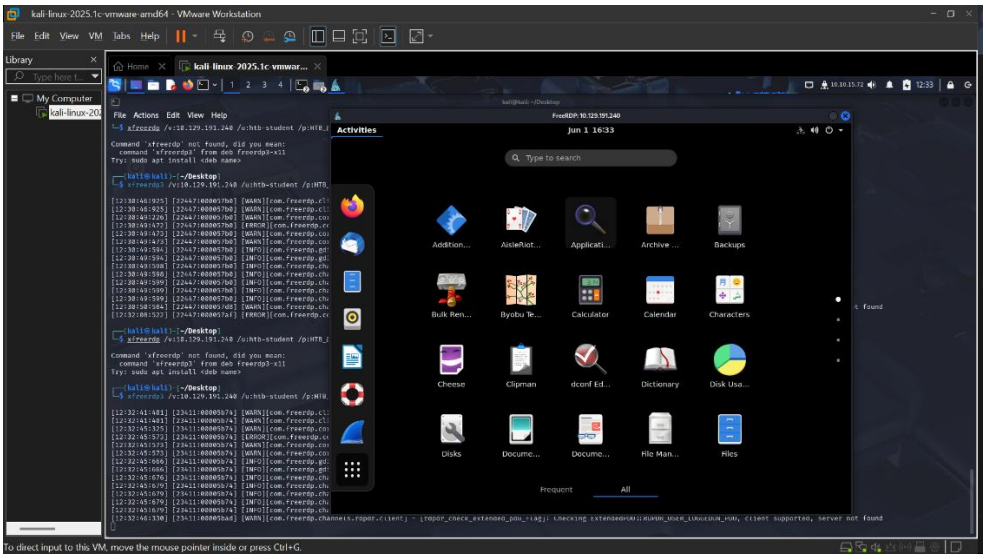


Wireshark Lab Report

In this part of the lab, I learned how to remotely access a virtual machine using **XfreeRDP** and capture live traffic using Wireshark. I accessed the lab machine by connecting via the FreeRDP client with the provided credentials and target IP. Once inside the lab environment, I opened Wireshark and started capturing on interface **ENS224**, letting it run for a few minutes to gather enough data for analysis.

Before diving into specific tasks, I took time to manually explore the .pcap file created during the capture. I applied what I had previously learned—checking conversations, following streams, and identifying protocols. This helped me better understand how machines on the network were communicating and which services were in use.

From my observation, there was notable communication between two IP addresses: **172.16.10.2** and **172.16.10.20**. I could see both **FTP and HTTP traffic** flowing between them.

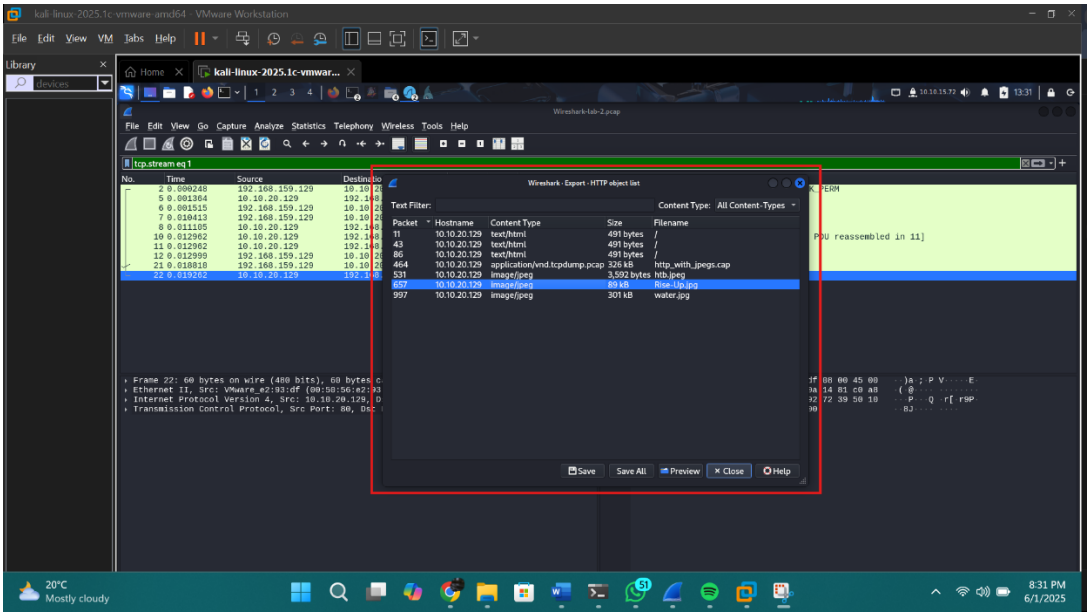


FTP Analysis

Using the display filter ftp, I narrowed down the packets to FTP-specific communication. I then applied the filter ftp.request.command to inspect the FTP commands being issued. This helped me see the sequence of login attempts and file transactions.

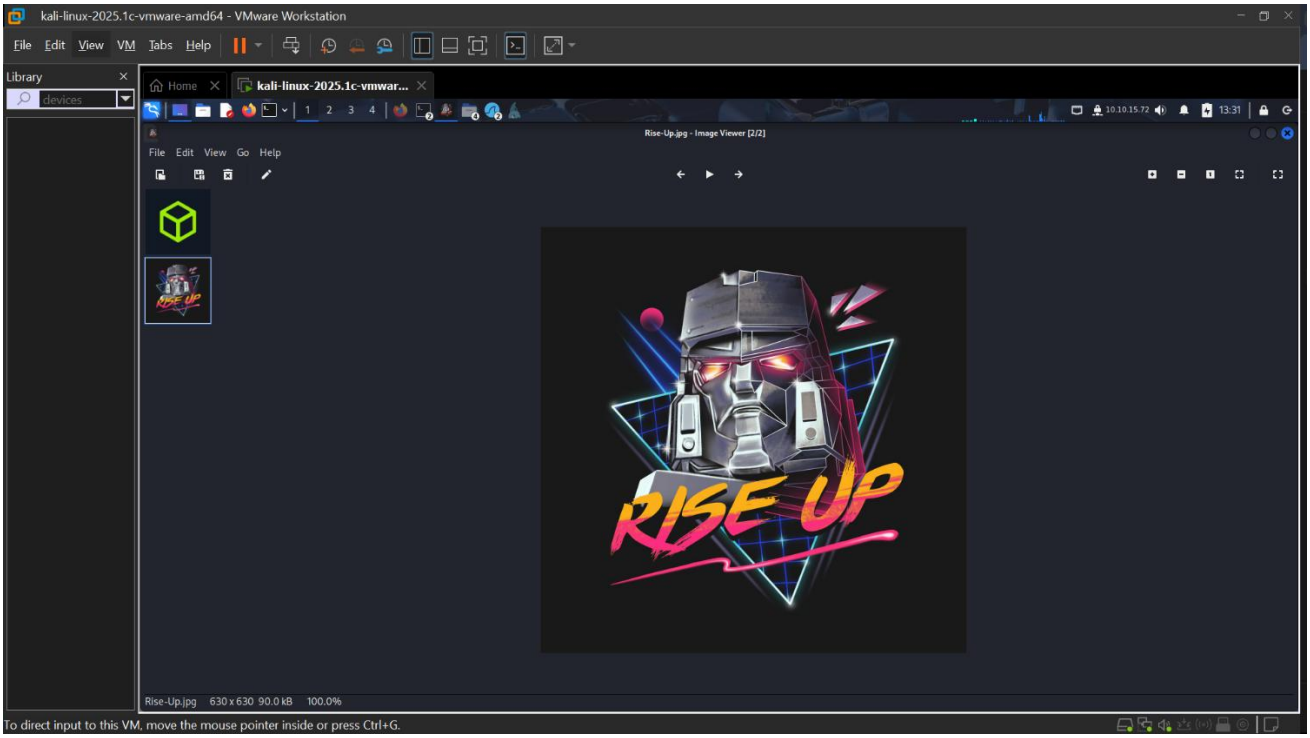
By analyzing the data, I could identify the **FTP server** and determine whether users were authenticated or connecting anonymously. In my case, I noticed that there was actual authentication involved, not just anonymous login.

To extract a transferred file, I located relevant ftp-data packets, followed the TCP stream of interest, and switched the view to **Raw format**. I then saved the contents using the original filename and verified the file type to confirm the extraction worked.



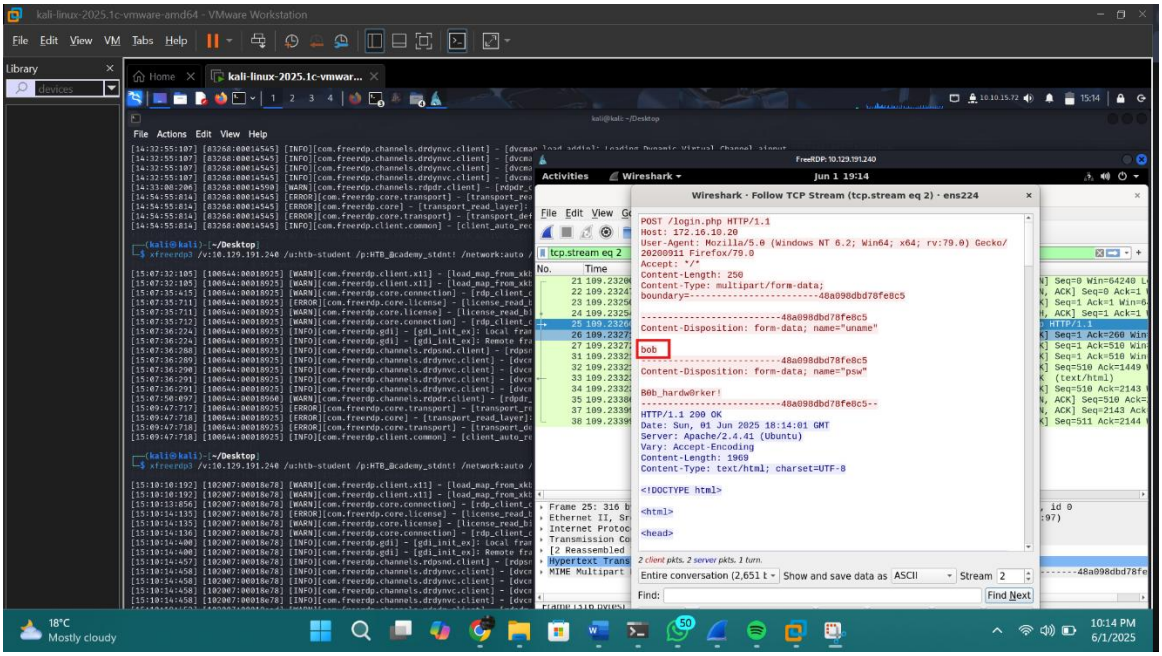
Lab Report Summary.

To begin, I opened a pre-captured file (Wireshark-lab-2.pcap) and applied the http filter to isolate HTTP packets. This helped reduce the clutter and focus on file transfers, specifically looking for images embedded in HTTP traffic. By inspecting the packets with a “200 OK” response and following their TCP streams, I was able to locate and extract a JPEG file. Using the “http && image-jfif” filter further narrowed down the results, and I successfully extracted an image named **Rise-Up.jpg**, which contained a reference to a certain Transformer leader.

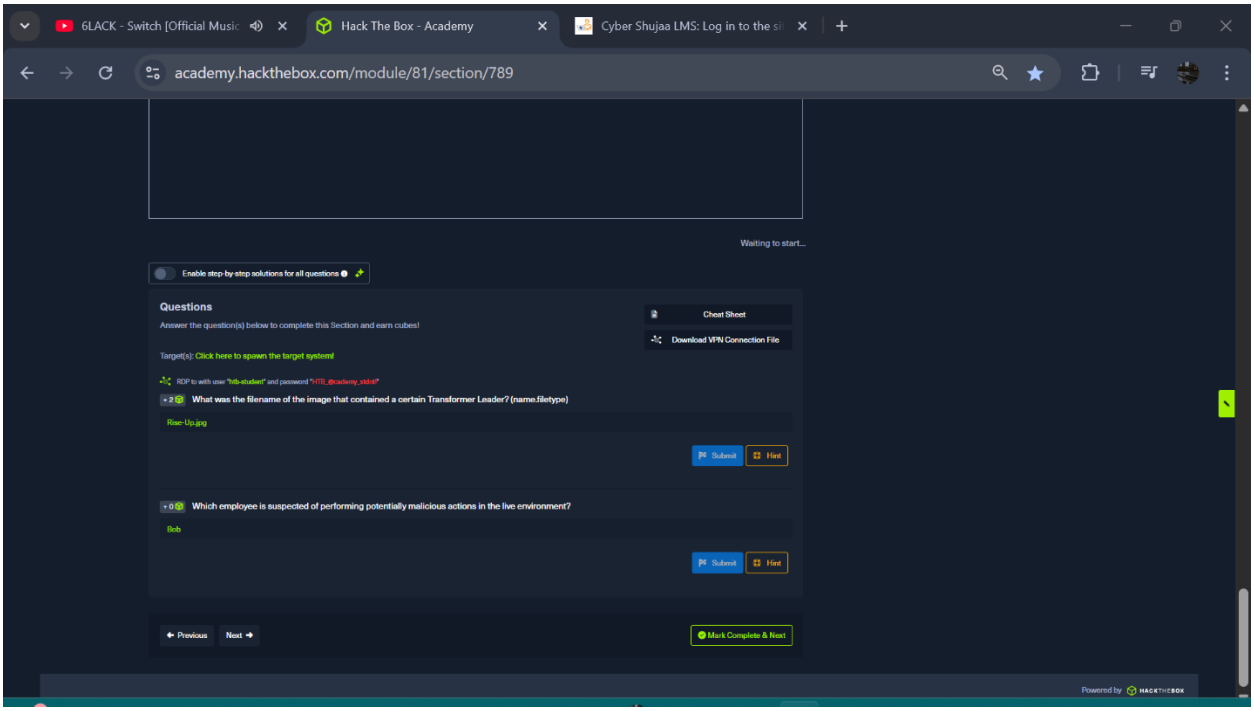


In the second part of the lab, I connected to the live environment using XFreeRDP, accessed the lab VM, and began a packet capture using interface ENS224. After letting it run for a few minutes, I began analyzing the traffic. I identified communication between two hosts (172.16.10.2 and 172.16.10.20) involving both HTTP and FTP protocols.

For FTP traffic, I filtered using ftp and further drilled down with ftp.request.command to review the interaction. I followed the TCP stream of the FTP data and saved one of the transferred files by exporting the raw content. The presence of FTP credentials and commands confirmed that a user had authenticated and transferred files, and further analysis suggested that **Bob** was the employee potentially involved in suspicious activity.


kali-linux-2025.1c-vmware-amd64 - VMware Workstation
Wireshark · Follow TCP Stream (tcp.stream eq 2) · ens224
POST /login.php HTTP/1.1
Host: 172.16.10.20
User-Agent: Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:79.0) Gecko/20200911 Firefox/79.0
Accept: */*
Content-Length: 256
Content-Type: multipart/form-data;
boundary=----------48a098dbd78fe8c5
Content-Disposition: form-data; name="uname"
bob
Content-Disposition: form-data; name="psw"
B0b_hardwOrker!
HTTP/1.1 200 OK
Date: Sun, 01 Jun 2025 18:14:01 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 1969
Content-Type: text/html; charset=UTF-8
<!DOCTYPE html>
<html>
<head>
18°C
Mostly cloudy
10:14 PM
6/1/2025

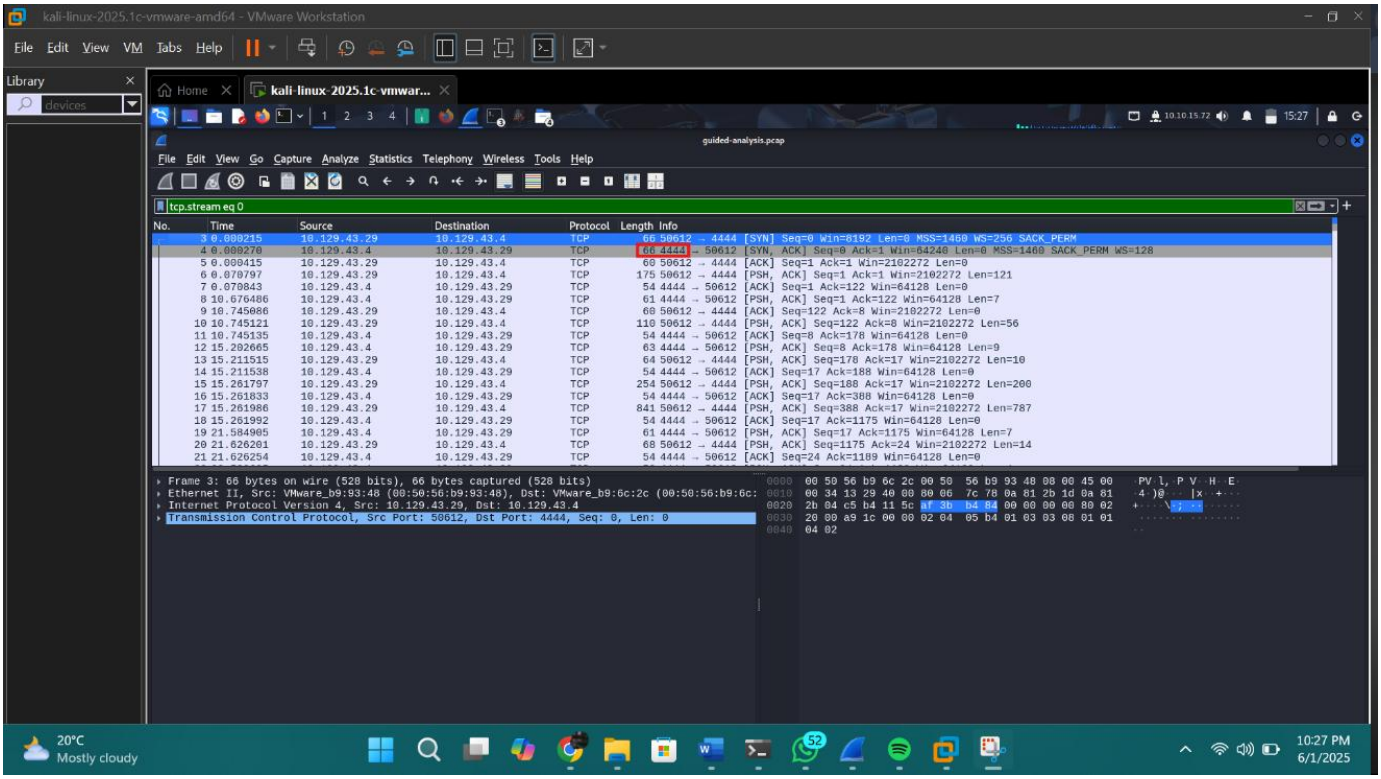
HTB Answers.


Hack The Box - Academy
academy.hackthebox.com/module/81/section/789
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
Download VPN Connection File
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
Submit
Hint
Previous
Next
Mark Complete & Next

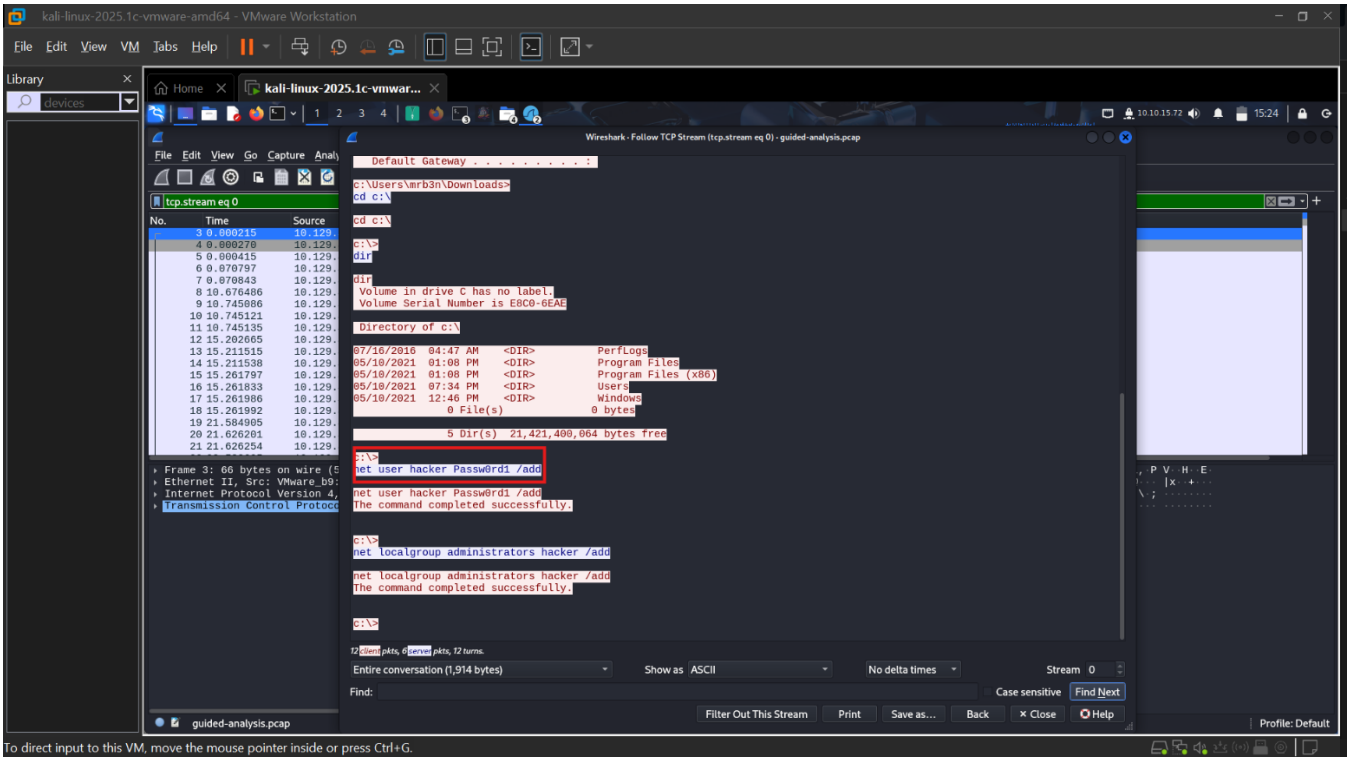
Guided Lab: Traffic Analysis Workflow

In this lab, I was tasked with investigating suspicious network activity linked to Bob's host (IP: 172.16.10.90). The issue was first noticed by one of the admins who observed strange connections coming from Bob's machine. My objective was to analyze this traffic using Wireshark and determine whether the activity was malicious.

To start, I connected to the Academy lab environment using FreeRDP and accessed the virtual machine where Wireshark was already set up. I began capturing live traffic on the ens224 interface and also examined a provided capture file named Guided-analysis.pcap, which contained historical data related to the issue. My scope focused on isolating traffic connected to Bob's host IP, especially within the last 48 hours. I applied appropriate filters in Wireshark to remove unrelated traffic and focused on the suspicious connections involving port 4444, which stood out as unusual.

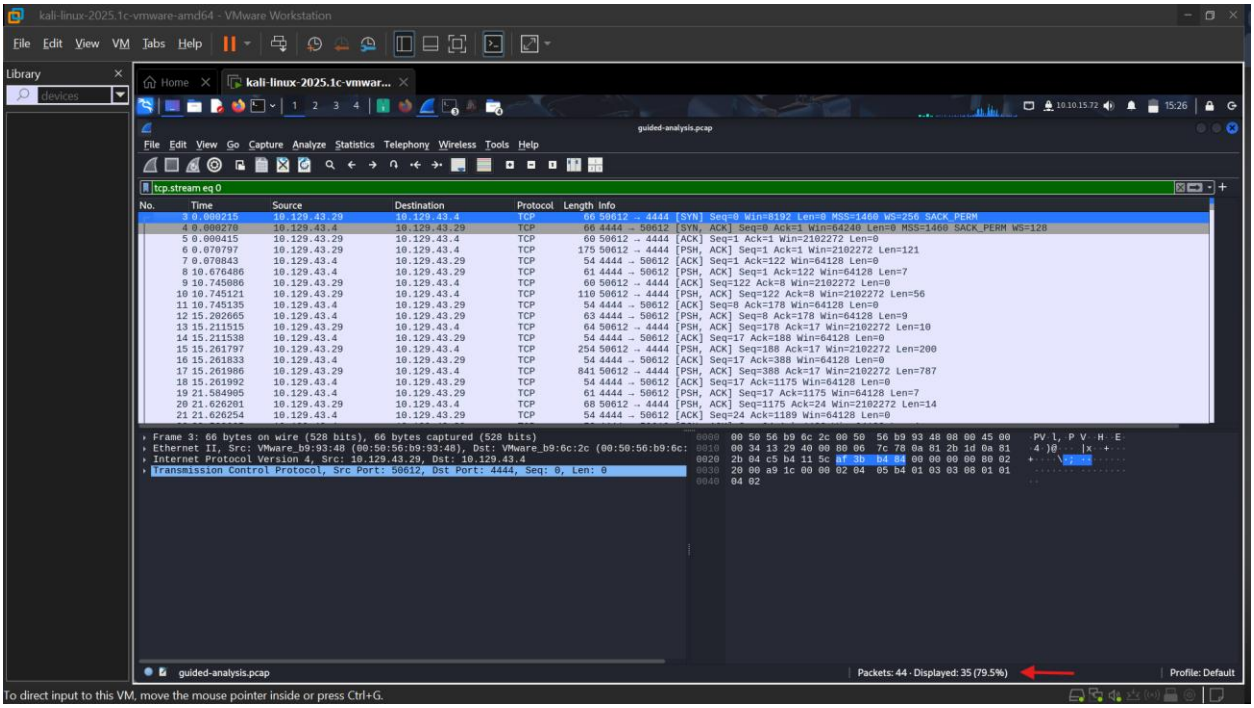


Through analysis of the conversations and data streams, I discovered that a new unauthorized user named **"hacker"** had been created on mrb3n's host, which strongly indicated potential privilege escalation or compromise.

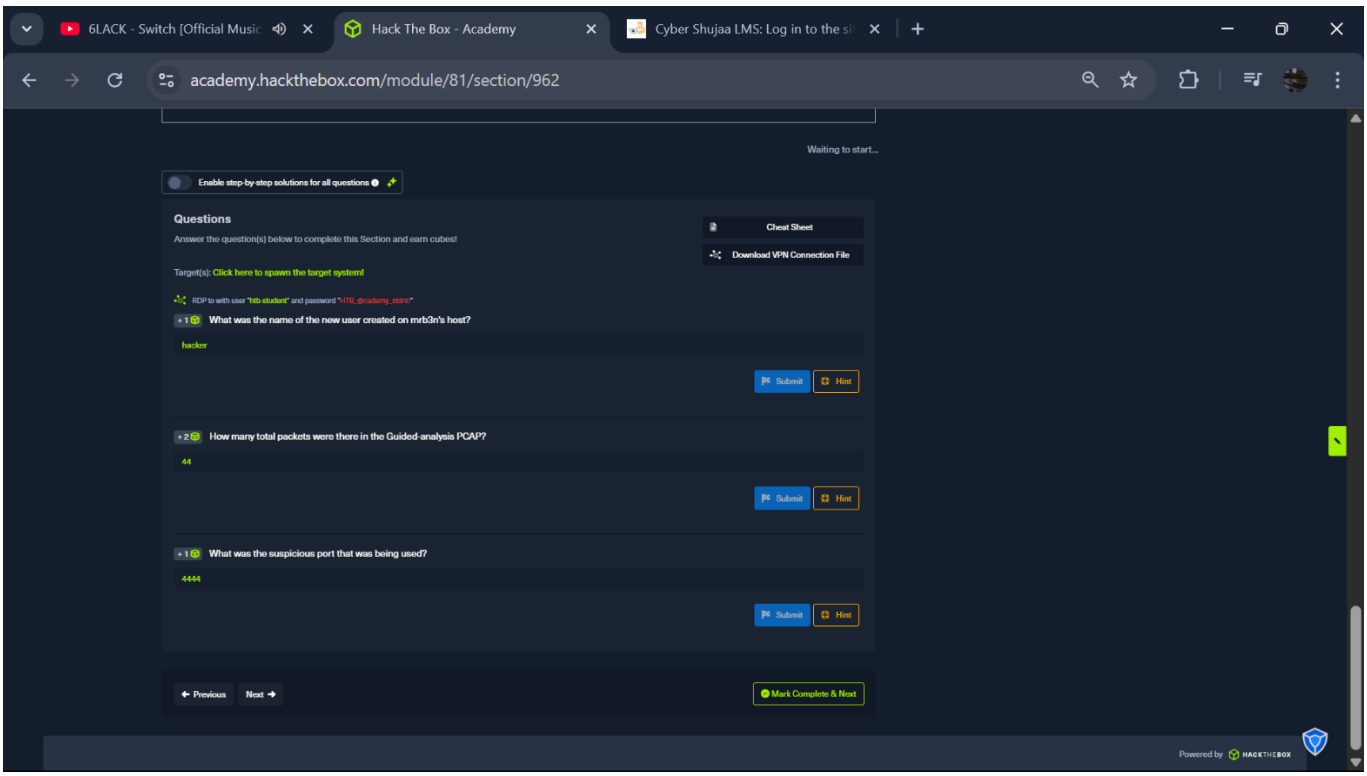


This lab helped me understand the importance of a structured approach when analyzing network traffic. From defining the scope, filtering relevant data, to identifying key indicators of compromise, the process allowed me to see how even a small anomaly can reveal deeper issues. I learned how to identify suspicious patterns such as unusual ports, strange traffic behavior, and unexpected user creation.

The experience reinforced the value of baseline traffic comparison, strong documentation during analysis, and the use of protocol-level investigation to uncover security concerns. This exercise gave me more confidence in using Wireshark and applying practical analysis workflows in real-world scenarios. I also identified that the total number of packets in the PCAP was 44.


kali-linux-2025.1c-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Home
kali-linux-2025.1c-vmwar...
guided-analysis.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 0
No. Time Source Destination Protocol Length Info
3 0.000215 10.129.43.29 10.129.43.4 TCP 66 50612 → 4444 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
4 0.000270 10.129.43.4 10.129.43.29 TCP 66 4444 → 50612 [SYN, ACK] Seq=0 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 50612 → 4444 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
6 0.070797 10.129.43.29 10.129.43.4 TCP 175 50612 → 4444 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=121
7 0.070843 10.129.43.4 10.129.43.29 TCP 54 4444 → 50612 [ACK] Seq=1 Ack=122 Win=64128 Len=0
8 10.676486 10.129.43.4 10.129.43.29 TCP 61 4444 → 50612 [PSH, ACK] Seq=1 Ack=122 Win=64128 Len=7
9 10.745086 10.129.43.29 10.129.43.4 TCP 60 50612 → 4444 [ACK] Seq=122 Ack=8 Win=2102272 Len=0
10 10.745121 10.129.43.29 10.129.43.4 TCP 110 50612 → 4444 [PSH, ACK] Seq=122 Ack=8 Win=2102272 Len=56
11 10.745135 10.129.43.4 10.129.43.29 TCP 54 4444 → 50612 [ACK] Seq=8 Ack=178 Win=64128 Len=0
12 15.202665 10.129.43.4 10.129.43.29 TCP 63 4444 → 50612 [PSH, ACK] Seq=8 Ack=178 Win=64128 Len=9
13 15.211515 10.129.43.29 10.129.43.4 TCP 64 50612 → 4444 [PSH, ACK] Seq=178 Ack=17 Win=2102272 Len=10
14 15.211538 10.129.43.4 10.129.43.29 TCP 54 4444 → 50612 [ACK] Seq=17 Ack=188 Win=64128 Len=0
15 15.261797 10.129.43.29 10.129.43.4 TCP 254 50612 → 4444 [PSH, ACK] Seq=188 Ack=17 Win=2102272 Len=200
16 15.261833 10.129.43.4 10.129.43.29 TCP 54 4444 → 50612 [ACK] Seq=17 Ack=388 Win=64128 Len=0
17 15.261986 10.129.43.29 10.129.43.4 TCP 841 50612 → 4444 [PSH, ACK] Seq=388 Ack=17 Win=2102272 Len=787
18 15.261992 10.129.43.4 10.129.43.29 TCP 54 4444 → 50612 [ACK] Seq=17 Ack=1175 Win=64128 Len=0
19 21.584905 10.129.43.4 10.129.43.29 TCP 61 4444 → 50612 [PSH, ACK] Seq=17 Ack=1175 Win=64128 Len=7
20 21.626201 10.129.43.29 10.129.43.4 TCP 68 50612 → 4444 [PSH, ACK] Seq=1175 Ack=24 Win=2102272 Len=14
21 21.626254 10.129.43.4 10.129.43.29 TCP 54 4444 → 50612 [ACK] Seq=24 Ack=1189 Win=64128 Len=0
Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: VMware_b9:93:48 (00:50:56:b9:93:48), Dst: VMware_b9:6c:2c (00:50:56:b9:6c:
Internet Protocol Version 4, Src: 10.129.43.29, Dst: 10.129.43.4
Transmission Control Protocol, Src Port: 50612, Dst Port: 4444, Seq: 0, Len: 0
guided-analysis.pcap
Packets: 44 · Displayed: 35 (79.5%)
Profile: Default
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

HTB Answers.


6LACK - Switch [Official Music
Hack The Box - Academy
Cyber Shujaa LMS: Log in to the si
academy.hackthebox.com/module/81/section/962
Waiting to start...
Enable step-by-step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
Download VPN Connection File
Target(s): Click here to spawn the target system!
RDP to with user "htb-student" and password "HTB_@cademy_stdnt!"
+1 What was the name of the new user created on mrb3n's host?
hacker
Submit
Hint
+2 How many total packets were there in the Guided analysis PCAP?
44
Submit
Hint
+1 What was the suspicious port that was being used?
4444
Submit
Hint
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX

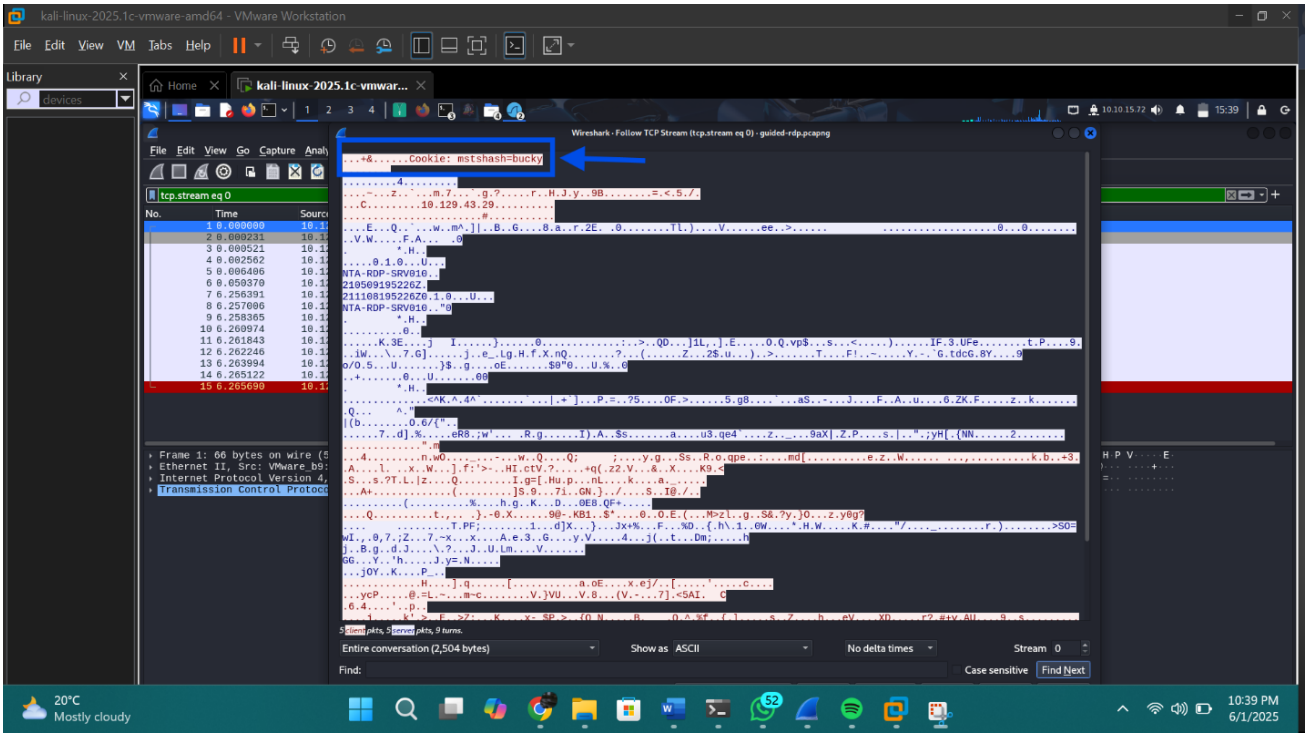
Decrypting RDP connections

The purpose of this lab was to explore the powerful capabilities of Wireshark, particularly its ability to decrypt Remote Desktop Protocol (RDP) traffic when the proper encryption key is available. This feature allows us to inspect what would normally be protected communication between hosts.

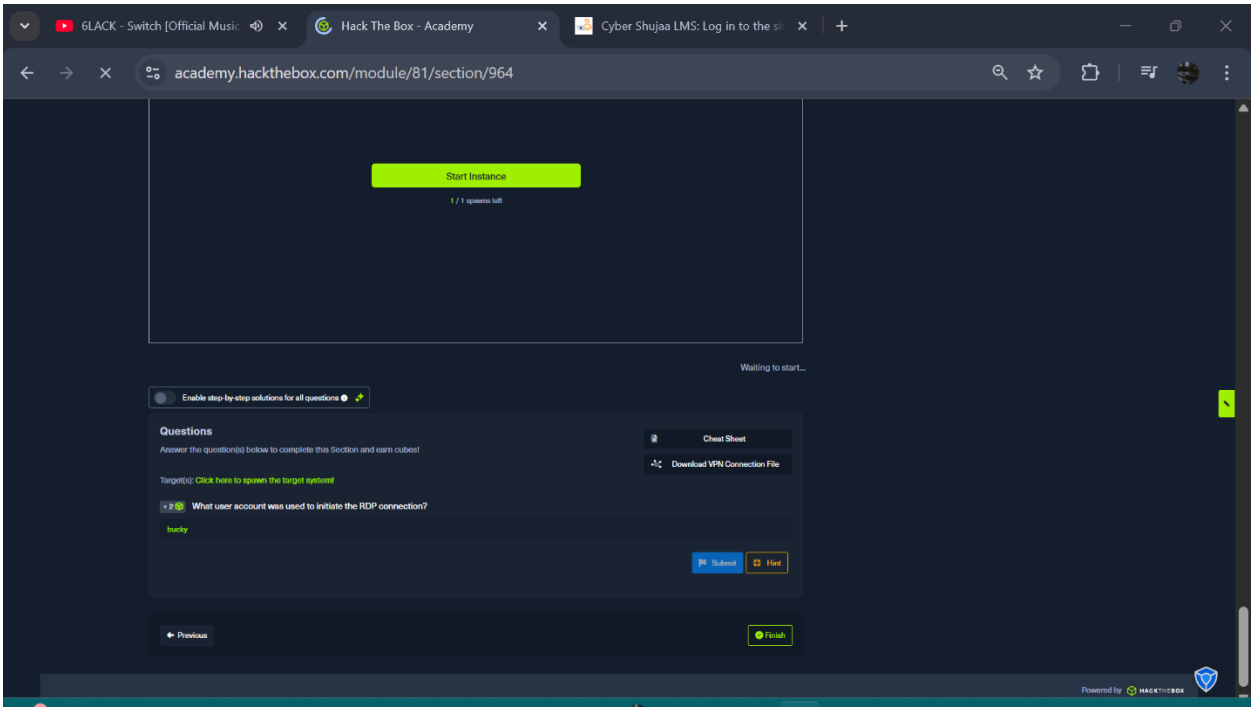
In this scenario, the Incident Response team had captured network traffic from Bob's machine showing RDP connections to another host on the network. While searching Bob's system for evidence, an RDP encryption key was discovered hidden inside a folder hive. This key enabled us to decrypt the captured RDP traffic. I began by opening the provided rdp.pcapng file in Wireshark. Given the large volume of data, I applied a filter specifically for RDP traffic to narrow down the packets relevant to our investigation.

Through the analysis, I identified that the RDP connection was initiated using the user account **"bucky"**. This helped confirm which credentials were in use during the suspicious remote access.

This exercise highlighted how, with the right decryption keys, Wireshark can reveal detailed session information from otherwise encrypted RDP traffic, which is critical for thorough forensic and security investigations. It emphasized the importance of key discovery and usage in decrypting network traffic to uncover potentially malicious activity.



HTB Answers.



Conclusion.

This challenge provided valuable hands-on experience in network traffic analysis using Wireshark and TCPdump within a controlled lab environment. Through practical tasks such as capturing live traffic, filtering protocols like FTP, HTTP, and RDP, and extracting files from packet captures, I developed a deeper understanding of how to investigate suspicious network behavior.

I learned how to apply targeted filters effectively to isolate relevant traffic, follow communication streams to uncover hidden data, and use decryption keys to analyze encrypted sessions. These skills are essential in real-world incident response scenarios where identifying malicious activity quickly and accurately is critical. Overall, this challenge reinforced the importance of methodical traffic analysis and how powerful tools like Wireshark and tcpdump can aid in uncovering security incidents, helping to protect networks by identifying potential threats early. The knowledge gained lays a strong foundation for further exploration and mastery in cybersecurity and network forensics.

HTB Badge.

HTB Shareable Link: https://academy.hackthebox.com/achievement/1920867/81