**Lab 2 Report Template**

**Name: Caleb Lemmons**

**ISU ID: 568911383**

**Date: 11/04/24**

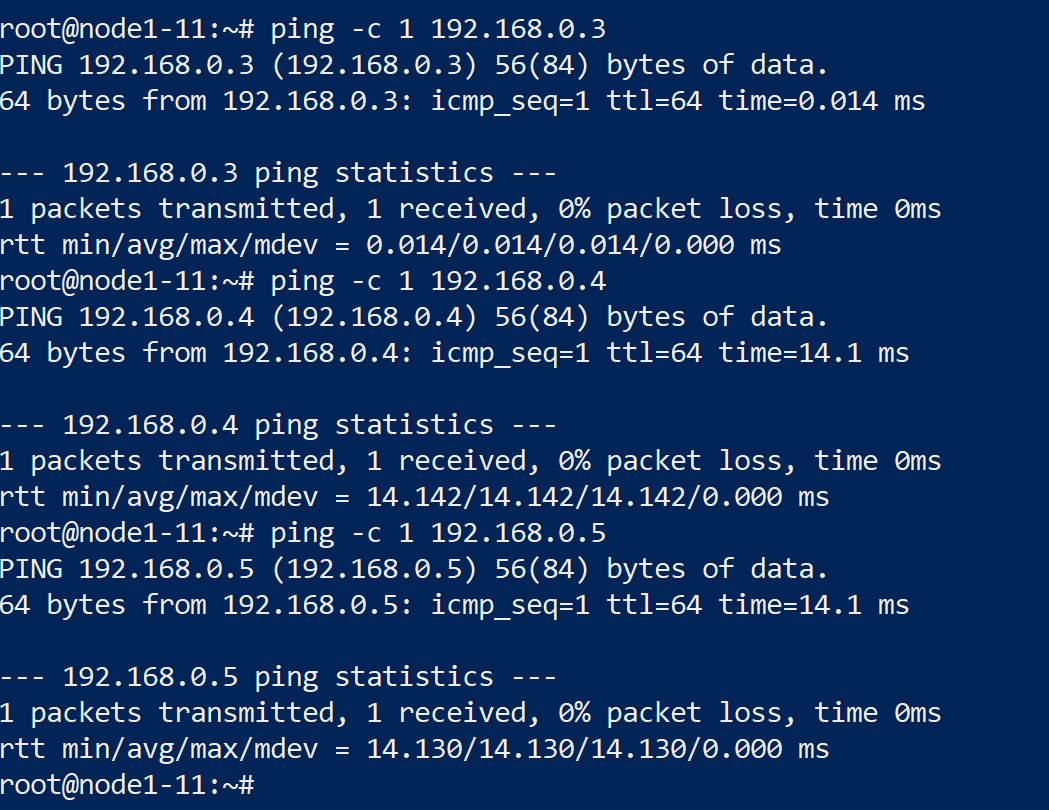
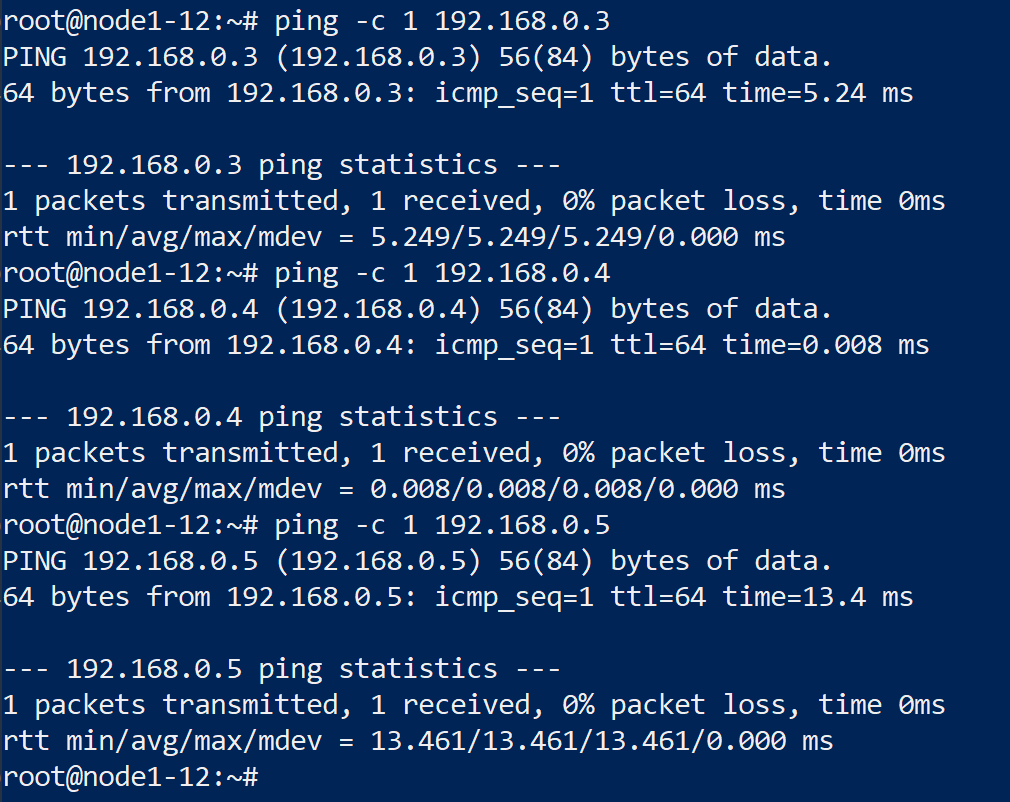
1. **Overview**

This lab uses the ORBIT testbed to allow us to demonstrate a Man-in-the-Middle (On-Path) attack in a controlled environment. The objective of the lab was to understand how Mallory is able to intercept communication between two people as well as analyze and compare the different types of information Mallory is able to gain through 4 methods of connection (ftp, sftp, telnet, & ssh). I first must set up the wireless network and the multiple nodes. Then I will be able to configure different servers for communication and execute an ARP spoofing attack as Mallory. This of course includes the capturing and analyzing of network traffic with Ettercap

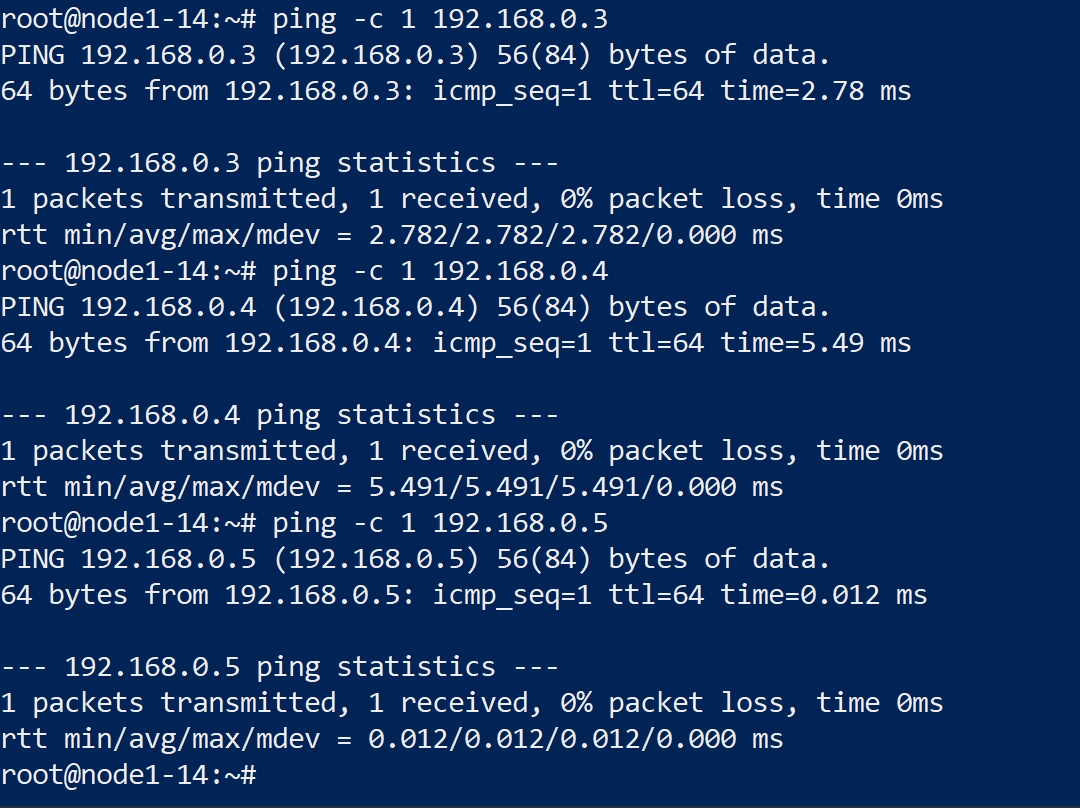
1. **Lab steps and observations**

* **Testbed & nodes were set up successfully**

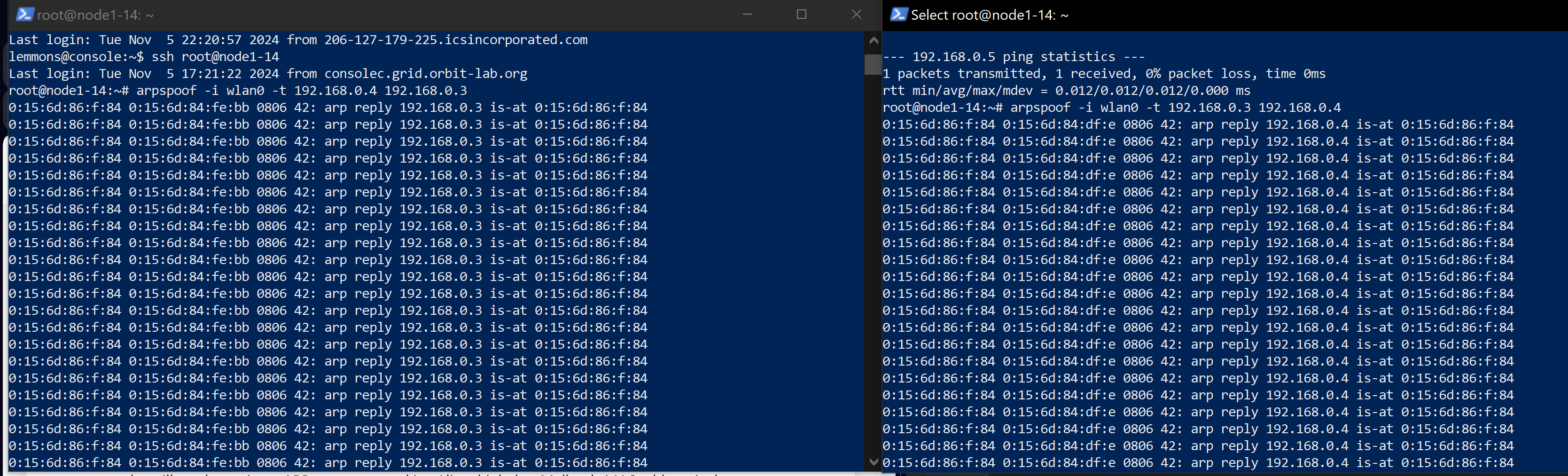
*Alice* *Bob*

*Mallory*

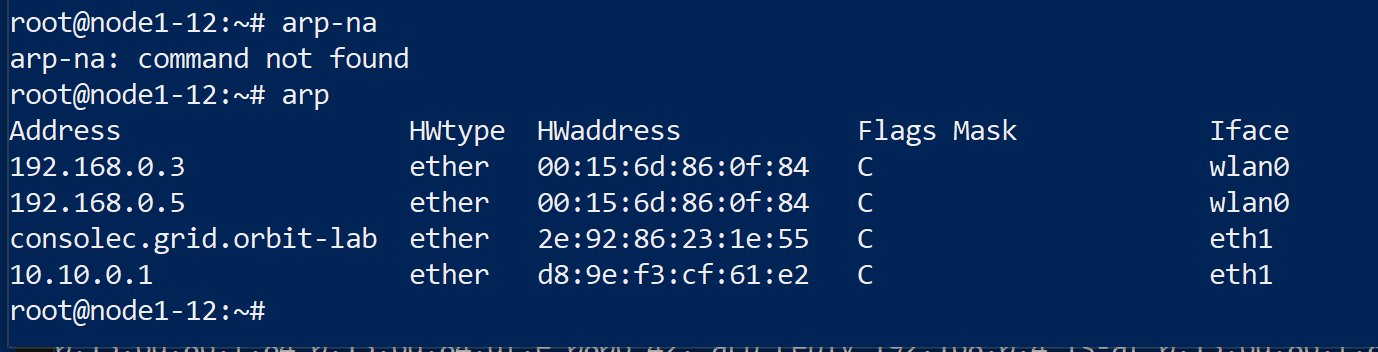


* **Output on the terminal indicates that Mallory is sending gratuitous ARPs:**

****

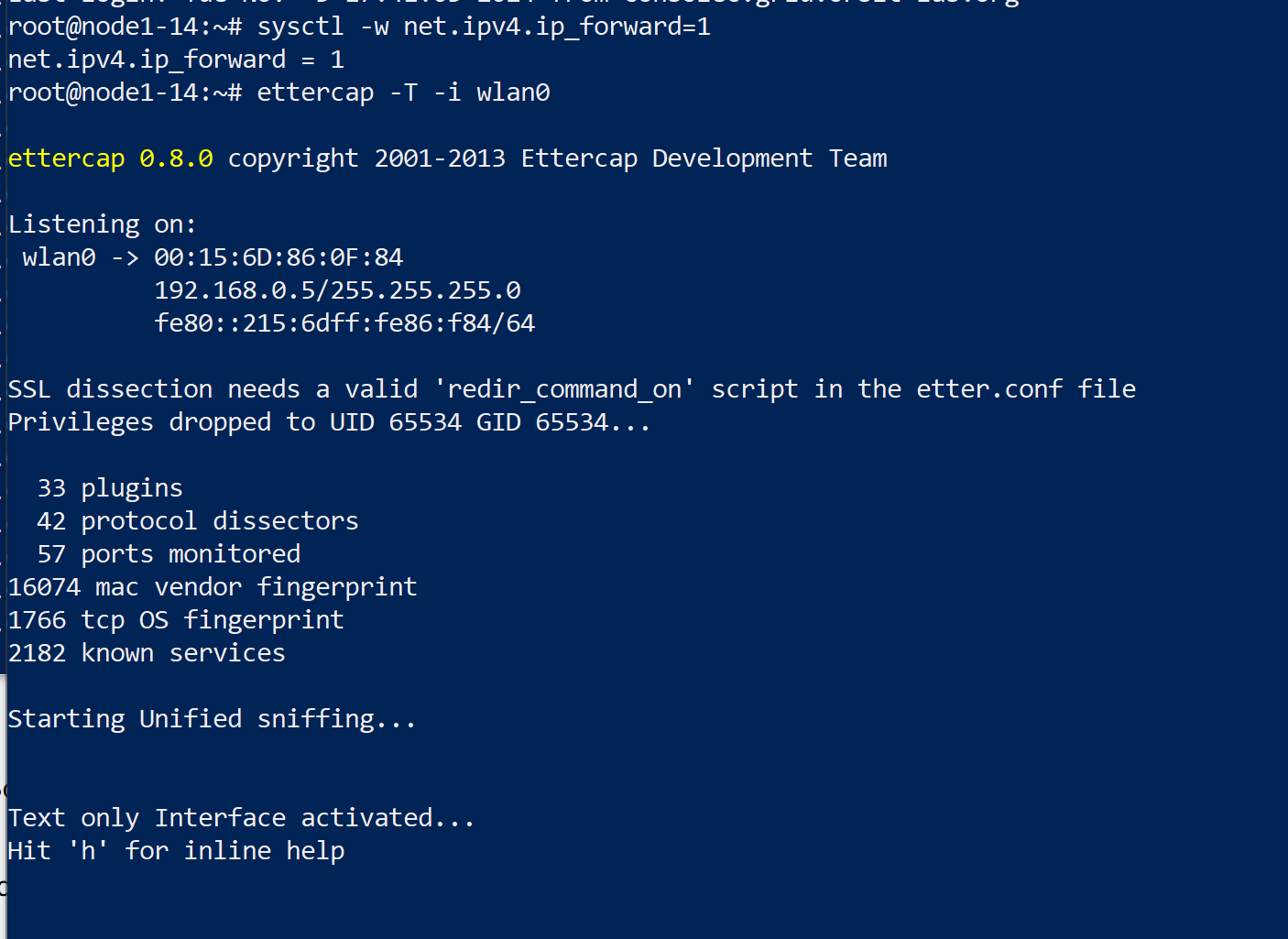
*The screenshot above shows Mallory successfully sending fake ARP messages to redirect traffic between Alice and Bob through herself. This can be confirmed by the MAC addresses.*

* **Attempt at running ‘arp-na’ on Bob**



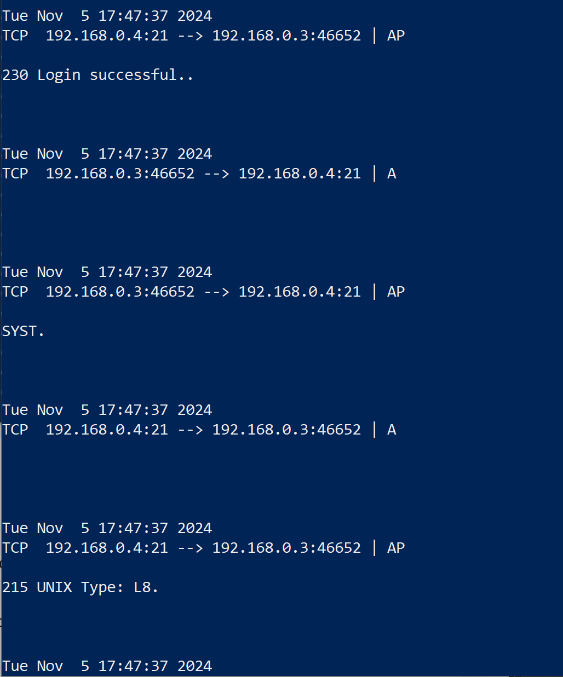
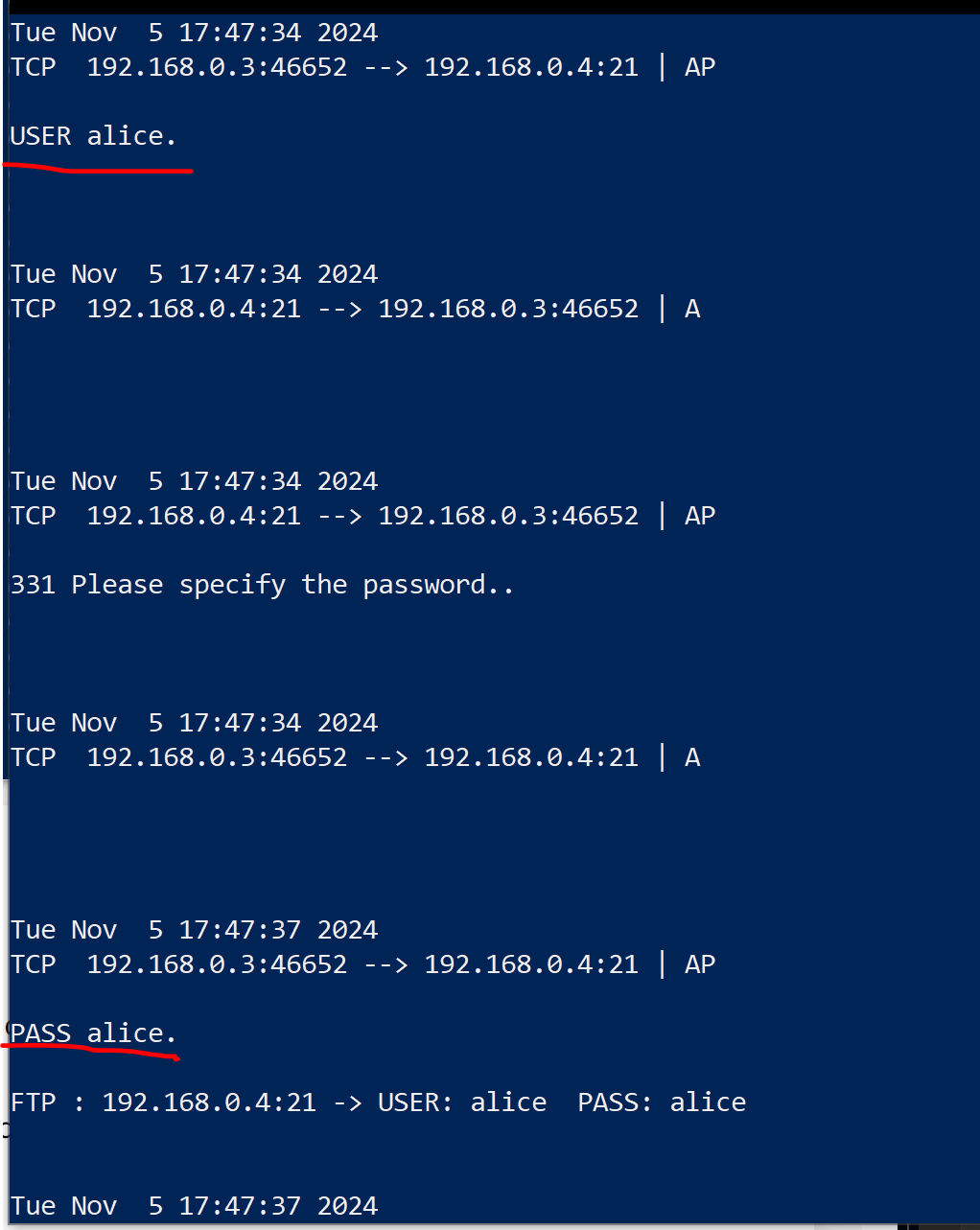
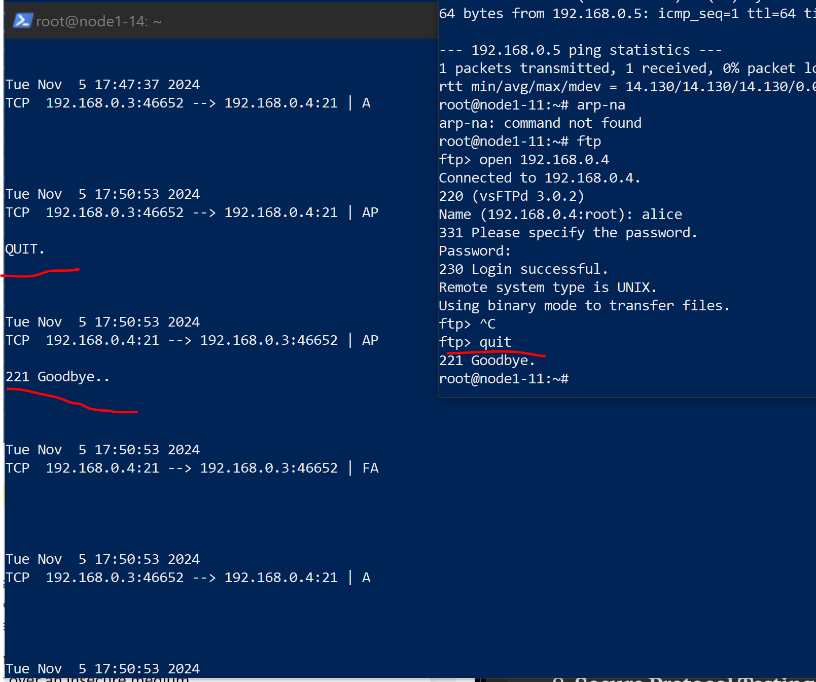
*The ‘arp-na’ command did not appear to be available on this machine. Assuming we were checking the ARP tables, I chose a different method. The screenshot shows the MAC address mapping for Alice is mapped though Mallory’s hardware, which indicate if the ARP poisoning was successful. If it was not successful, then it would show the legitimate MAC addresses in the ‘HWAddress’ field.*

* **Packet Forwarding enabled/Ettercap started**

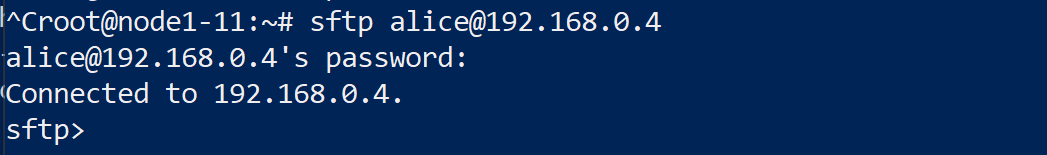
****

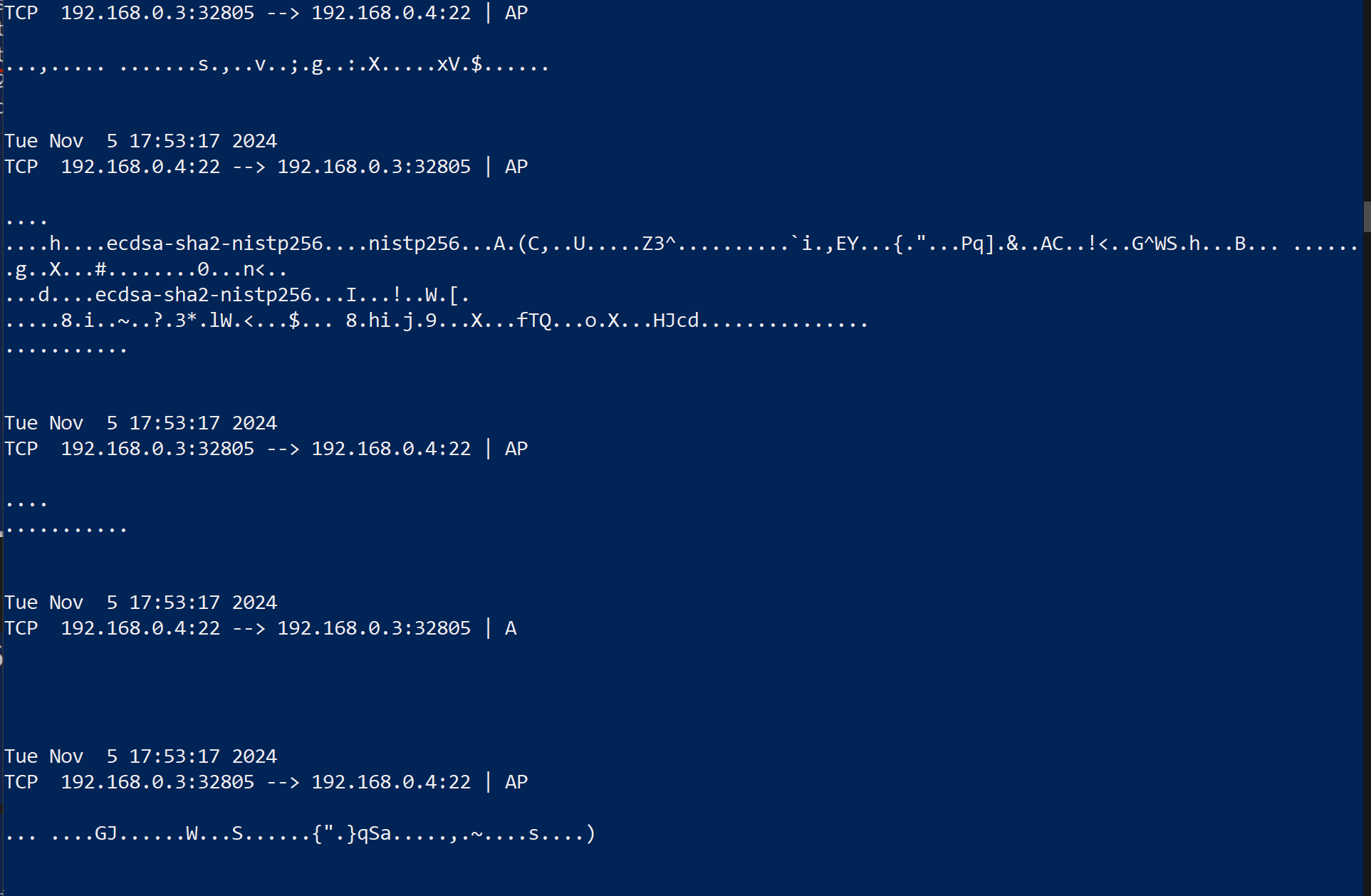
* **FTP Ettercap Results/Analysis**

A blue screen with white text

Description automatically generated ** **

The FTP command sequences and responses are fully visible, showing how FTP transmits all data without encryption.The Ettercap results clearly capture the credentials in plaintext over FTP. The username and password ‘alice:alice’ is fully visible.

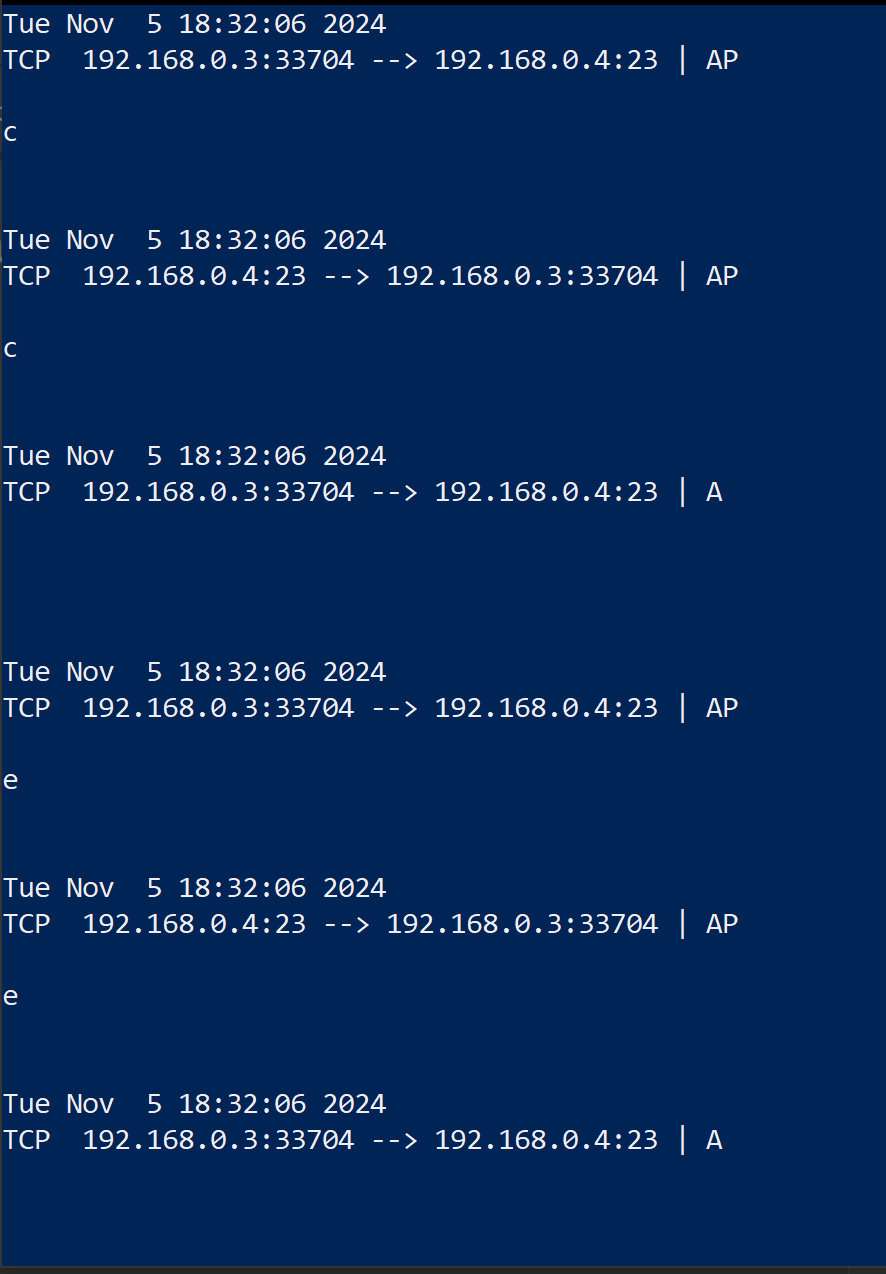
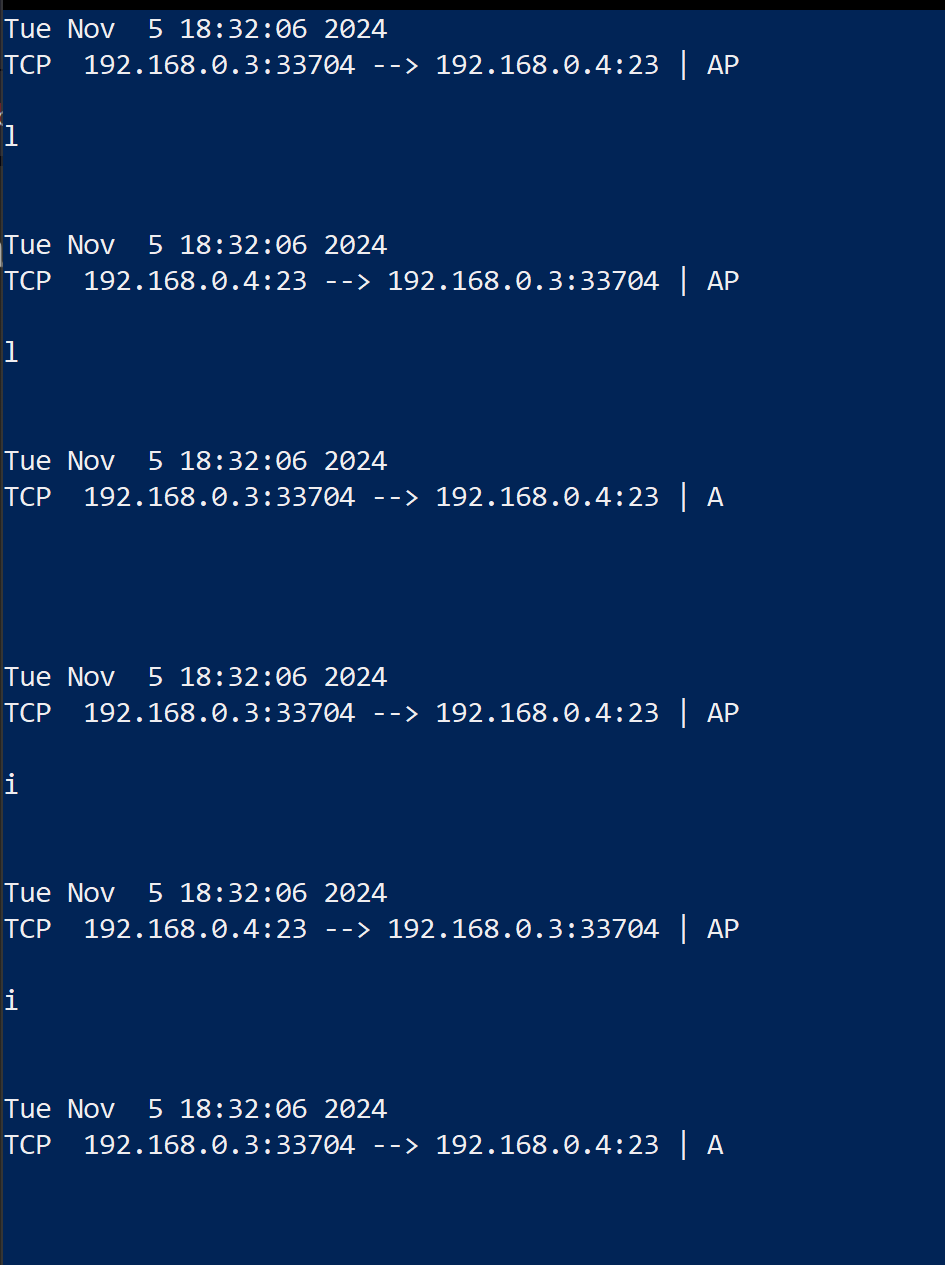
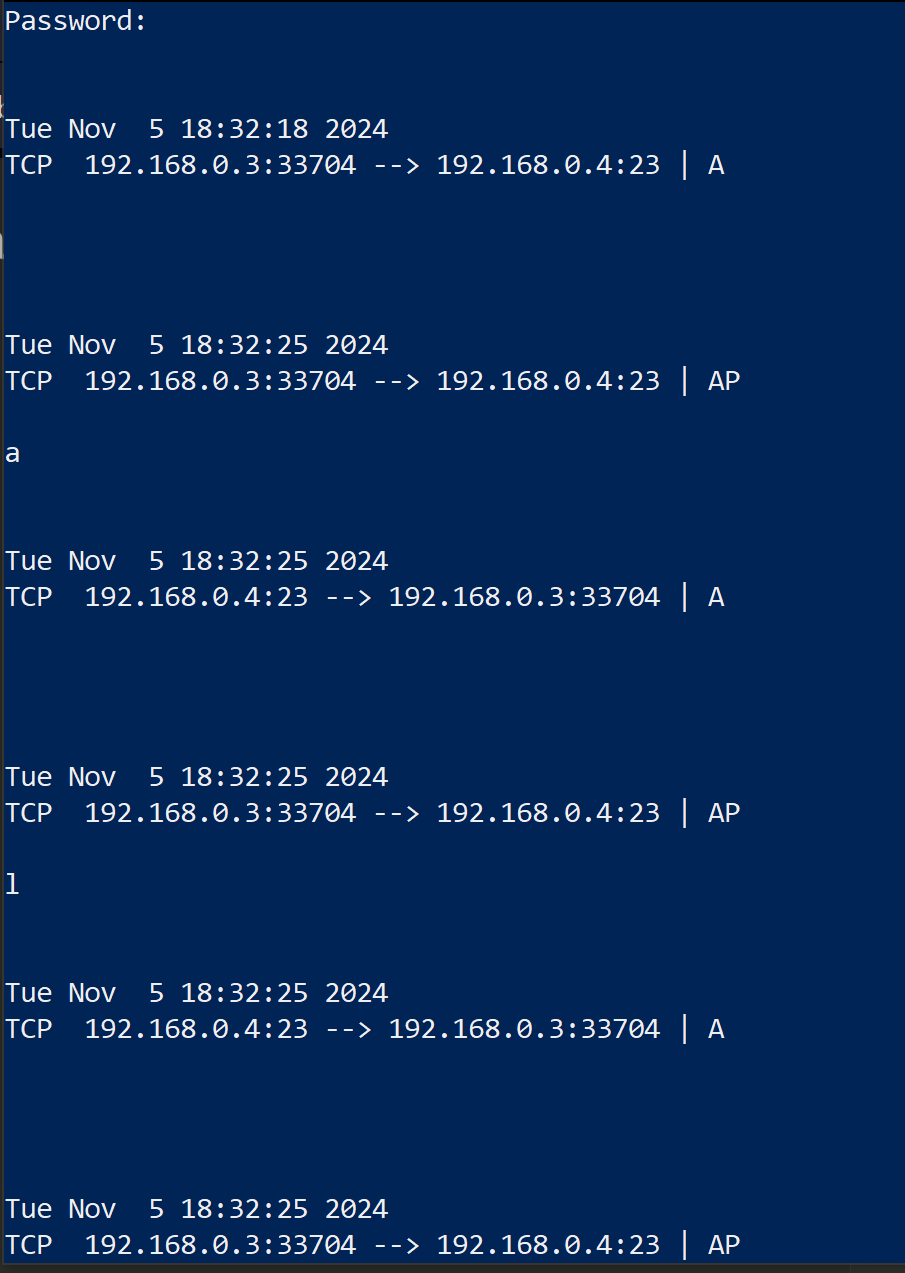
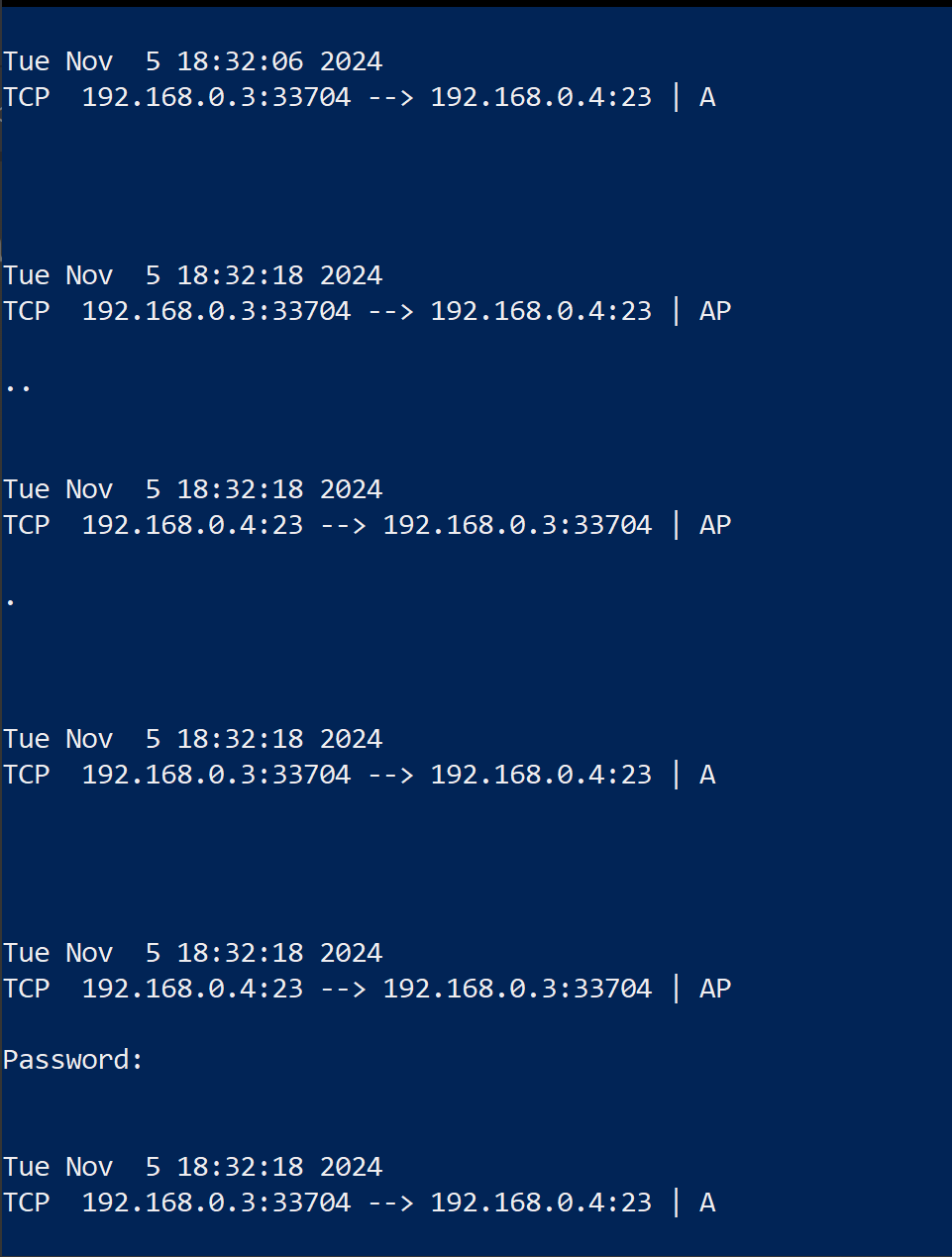
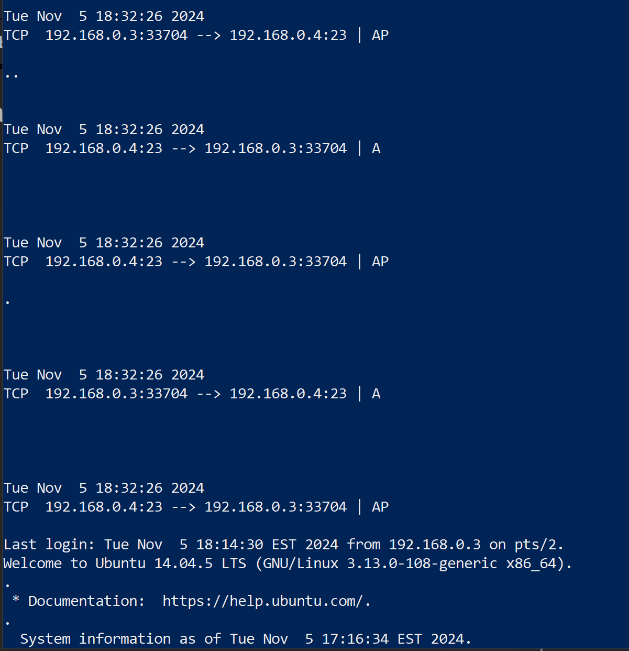
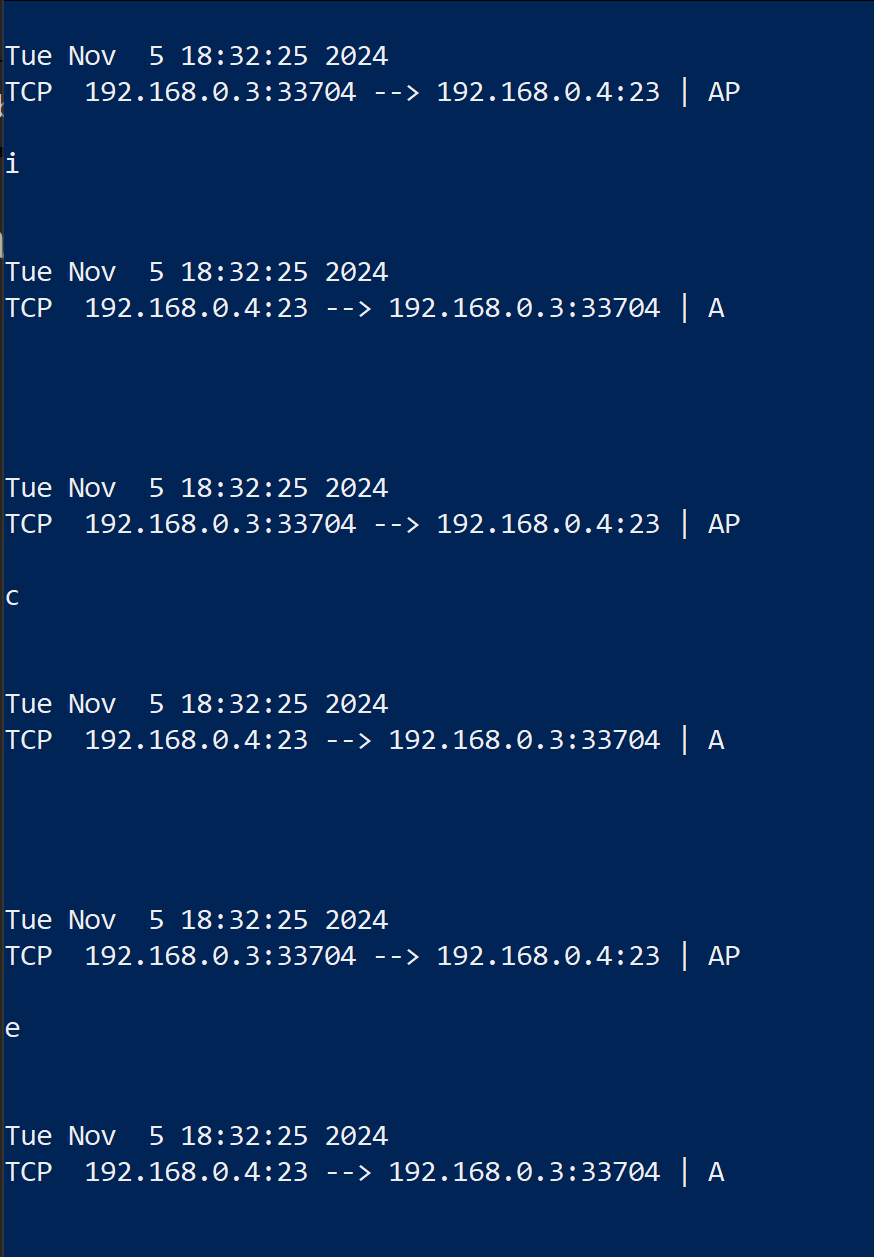
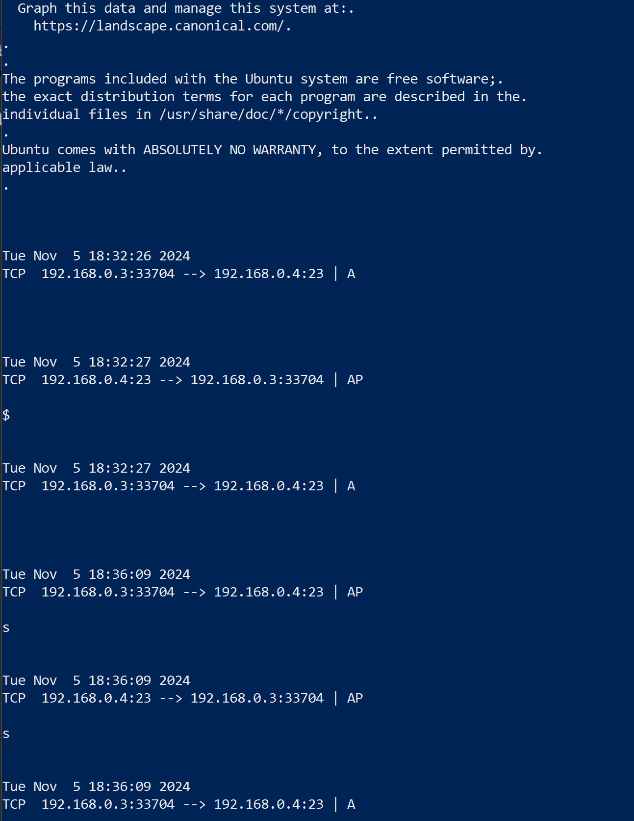
* **SFTP Ettercap Results/Analysis **

****

The SFTP captures show obvious signs of data streams being encrypted. There seemed to be no clear text credentials visible, and at points, data appears as encrypted SSH packets. I couldn’t actually see the file transfer contents, which shows SFTP’s security advantages over FTP.

* **Telnet Ettercap Results/Analysis**

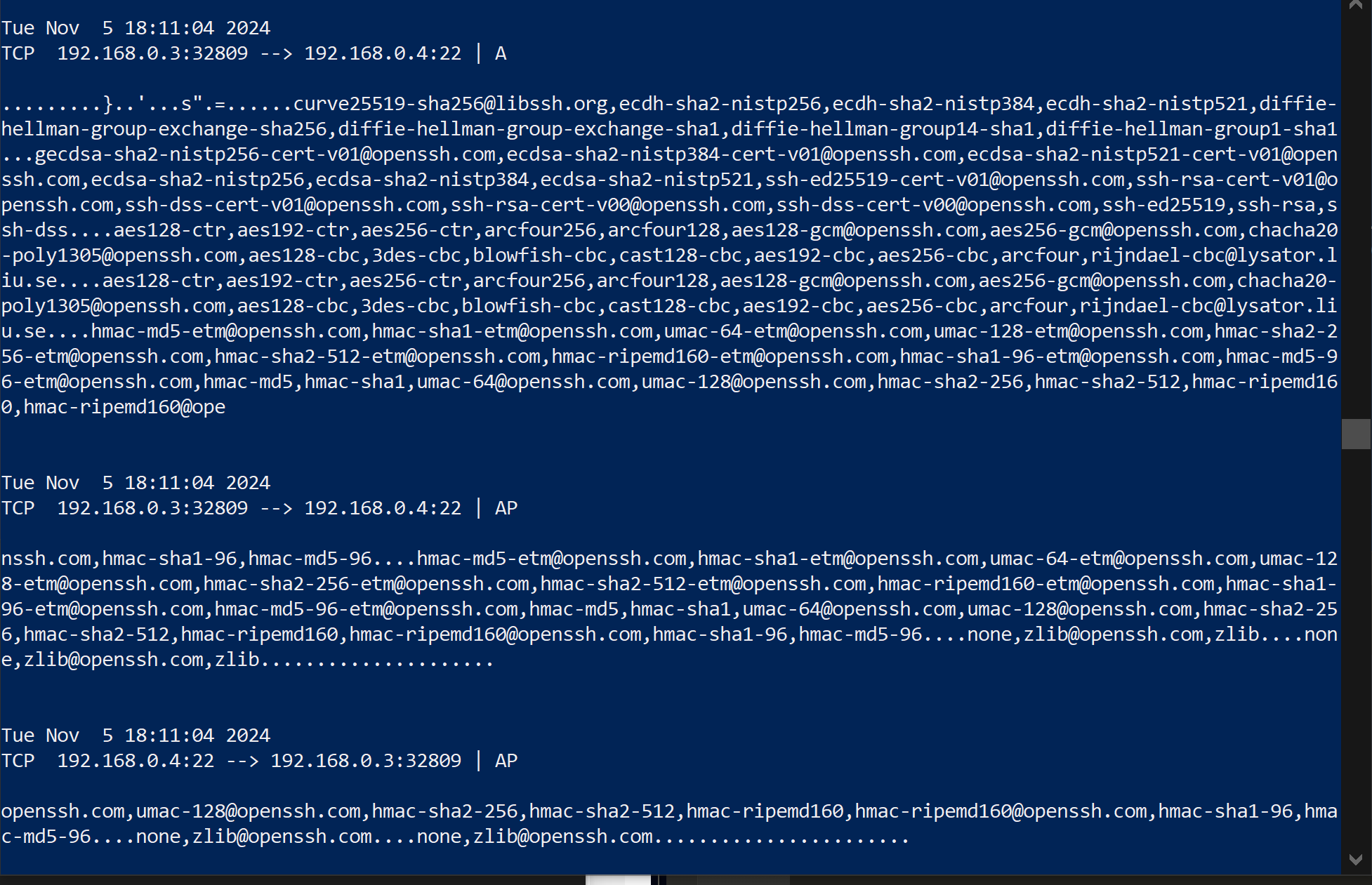
**A screenshot of a computer

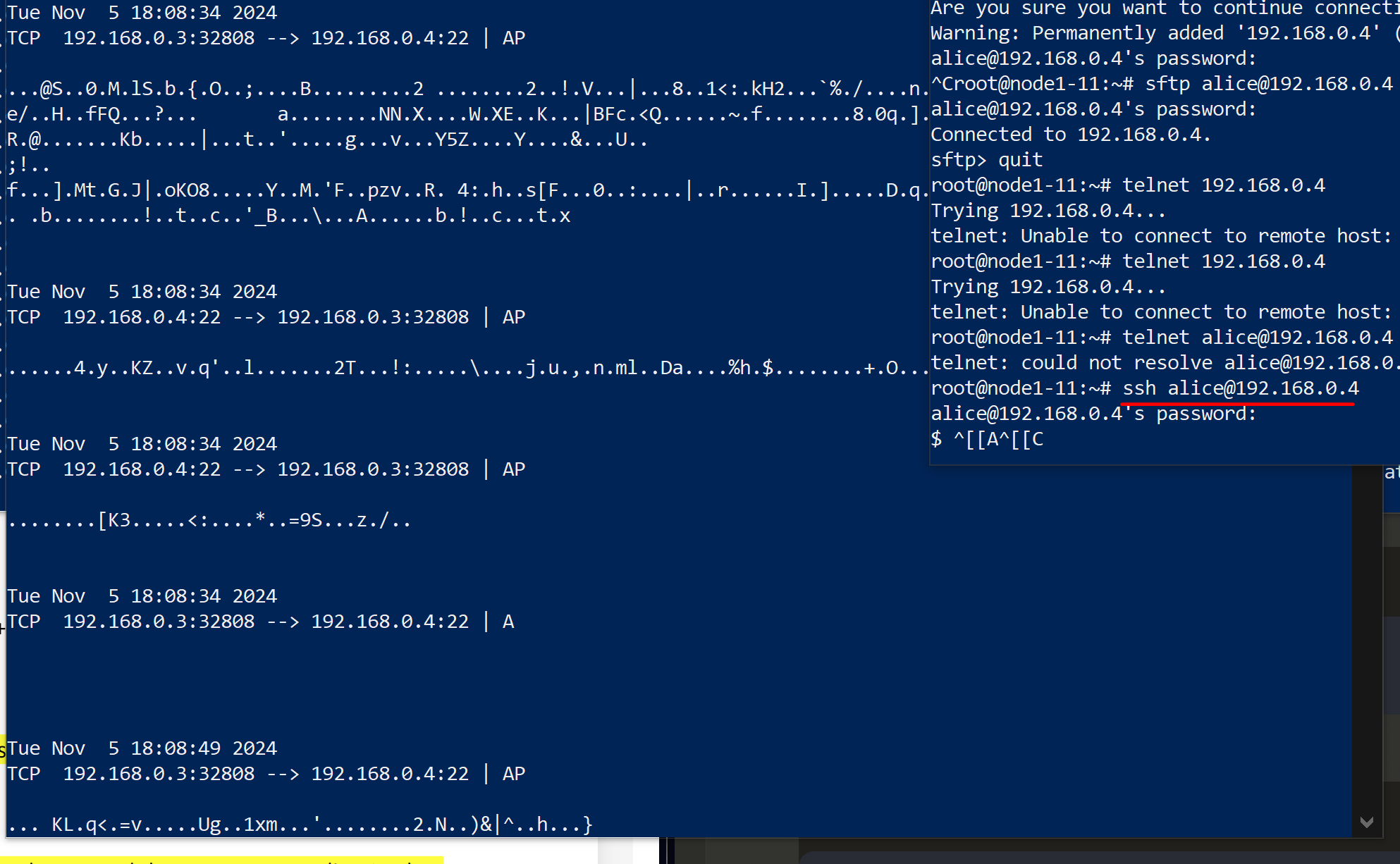
Description automatically generated    **

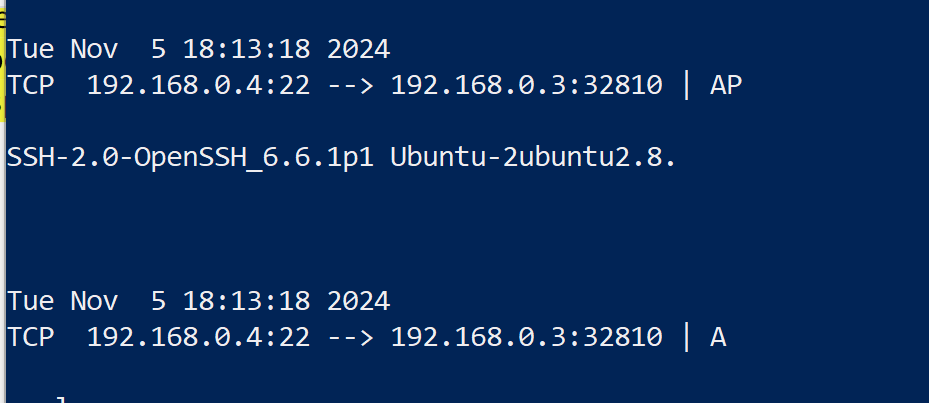


Similar to FTP, the telnet captures show log in credentials in plain text form twice for some reason. Telnet goes a step further by having all of the session content be visible. This means that every command line input and output is readable, which shows telnet’s lack of security and vulnerability to packet sniffing.

* **SSH Ettercap Results/Analysis**

****

****

****

The last set of captures happened to be the most secure. The SSH capture shows no visible signs of log in credentials or session content thanks to the use of encryption for data packets. The only visible piece of information is the SSH protocol negotiations and OpenSSH version.

* **Final Comparison**
* FTP: All traffic (log in credentials & file transfers) is visible in plaintext making it completely insecure.
* SFTP: Unlike FTP, traffic is encrypted through SSH tunneling, showing only encrypted packets (NO readable credentials or file data).
* Telnet: Similar to FTP, all session data including login credentials and command inputs are visible in plain text.
* SSH: Like SFTP, all traffic is encrypted, protecting credentials and session content from being intercepted.
* **Explain whether creds sent using this app can be captured by malicious users when using an insecure medium (Like a public WiFi hotspot, or unsecured WiFI network with no password)**
* FTP: Credentials can be easily captured as they are sent in plain text. On public WiFi, any malicious user can intercept these using packet sniffing tools
* SFTP: Credentials cannot be captured even on insecure networks due to SSH encryption
* Telnet: Credentials are extremely vulnerable to capture on public networks as they are transmitted without encryption
* SSH: Credentials remain secure even on public WiFi due to strong encryption, making them unreadable to attackers
* **Which file transfer application is more secure (FTP or SFTP)? Which remote login application is more secure (Telnet or SSH)? Explain…**
* SFTP is much more secure than FTP because of its use of encryption through SSH tunneling. While FTP sends everything in plain text, SFTP protects all traffic like credentials and file content from interception.
* SSH is significantly more secure than Telnet for similar reasons as above. Telnet sends all session data, including passwords in plain text, while SSH encrypts the entire session. This helps prevents from things like credential theft and session hijacking.

1. **Summary and conclusions**

Lab 2 demonstrated the security differences between encrypted and unencrypted protocols through hands on experience with MITM and ARP Spoofing attacks. The results showed that unencrypted protocols (FTP & Telnet) clearly exposed all data including credentials in plaintext, making them completely insecure on a public network. In contrast, SFTP and SSH successfully protected sensitive information with forms of encryption. Through the implementation of ARP poisoning and packet analysis on Ettercap, I was able to see firsthand why encrypted protocols are necessary for secure communications. This lab reinforced the importance of choosing secure protocols for protecting personal data in real-world situations.