LAB REPORT

LAB TITLE: Analyzing Data Link and Network Layer Traffic with Wireshark

Name: Gabriel Miracle Nwauche

PSU ID: GMN5192@psu.edu

Course: IST 220

Instructor: Nick Giacobe

Date: 2/16/2023

General Context:

Resiliency was truly shown in this week's lab excursion. Within the last 4 layers of the OSI layers, the user is tasked with the Transport and Application layers which is said to be the “microscope” of the Data Link, Network Layer and Transport Layer. To begin the lab, the Wireshark application is opened which begins capturing the Network Traffic which receives thousands of packets throughout the process. In the early stages the lab prompted the user to inspect the File Transfer packets first and examine the informative contents inside the packet details view. This would be a typical occurrence when dealing with Network traffic though, so let's take it to the other side of the lab, but we will inevitably come back to the packets.

A difficult part of this lab was the actual transferring of the target file across the specified systems in the network. What made it so difficult was the “nitpicking” through the file directories and subdirectories to articulate where the file actually was (Text file). Once the lab guided the user through those steps and the transfer was successful, the user had to switch networks to navigate a command prompt. This prompt succeeded, however, this would lead to the inevitable return to the packet dissector in Wireshark. This was after an issue that happened in the Command prompt window that sent a reset packet to the network, derailing the login process and frustrating the user. After finding the packet and resolving the solution through TCP streams, the lab continued to flow smoothly. Once another mishap was reached, this portion of the lab gave me enough knowledge on the topic to solve it myself, like a person in the real world would!

Technical Context

This lab required diligence and true understanding of the connection between the Application layer and the Transport layer protocols. Usually during the Wireshark capture of the sent and received packets, we would be focusing on the ICMP packets; but since we were focusing on a different layer of the OSI, we ignored the protocol that the ping command usually uses. Now we examine the TFTP traffic in the network, but this would apply to the user manually transferring a file from that server. Then when it comes to the FTP packets we look at the client (vWorkstations) and server (TargetWindows01) but one of the skills you can become knowledgeable of is recognizing specific transfers of specific packet forms. For example, the user would need to know you can only use this understanding of client-server correlation if the systems are used in internal networks. The capture on *Figure 17* gives a good representation of how those systems are connected.

The most important skill of this lab is one that involves putting the pieces together to make a bigger picture. This is much of what *fragmentation* of the packets do. First the user would have to know the background of why these packets are split up in the first place. This is because the protocols we were previously viewing just conveys the data, not *communicate.* The first example would be the connection between the TLS and TCP packets of the data. The TLS is

a higher protocol than the TCP, but the TLS does still need to rely on the TCP packets for data release. This would call a TLS stream which is initiated with a Client Hello: a version of the TLS being used and becomes available to cipher suites. Then onto the Server Hello: the certificate for the client to validate with the cipher suite, this would identify the server and initiate the Key exchange. Session keys are created for both the client and server, encrypting all further communication for security precaution (this process connects with the TCP Handshake). With that being said, learning how to piece together fragmented packets can put together encrypted information and display it as what it is supposed to be. The Raw data looks so overwhelming but when placed in a file can make for a better picture!

SECTION 1: HANDS ON DEMONSTRATION, PART 1

Configure the Wireshark and Generate Network Traffic

28. Make a screen capture showing the successful tftp file transfer message in the Command Prompt.

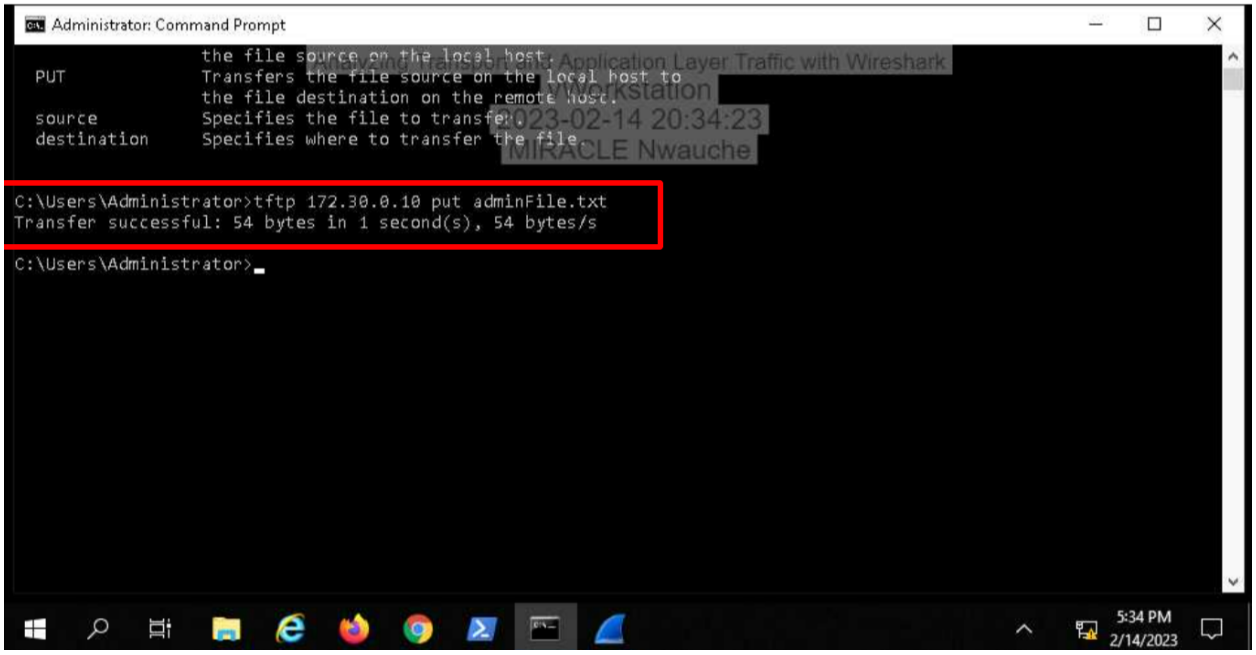

Figure 1, Section 1, Part 1: *TFTP file transfer*

There was a successful tftp file added onto the vWorkstation device, after typing the command prompt *tftp 172.30.0.10 put adminFile.txt* the message on the screen shows the prompt was successful.

32. **Make a screen capture** showing the **Tftpd64 Server log**.

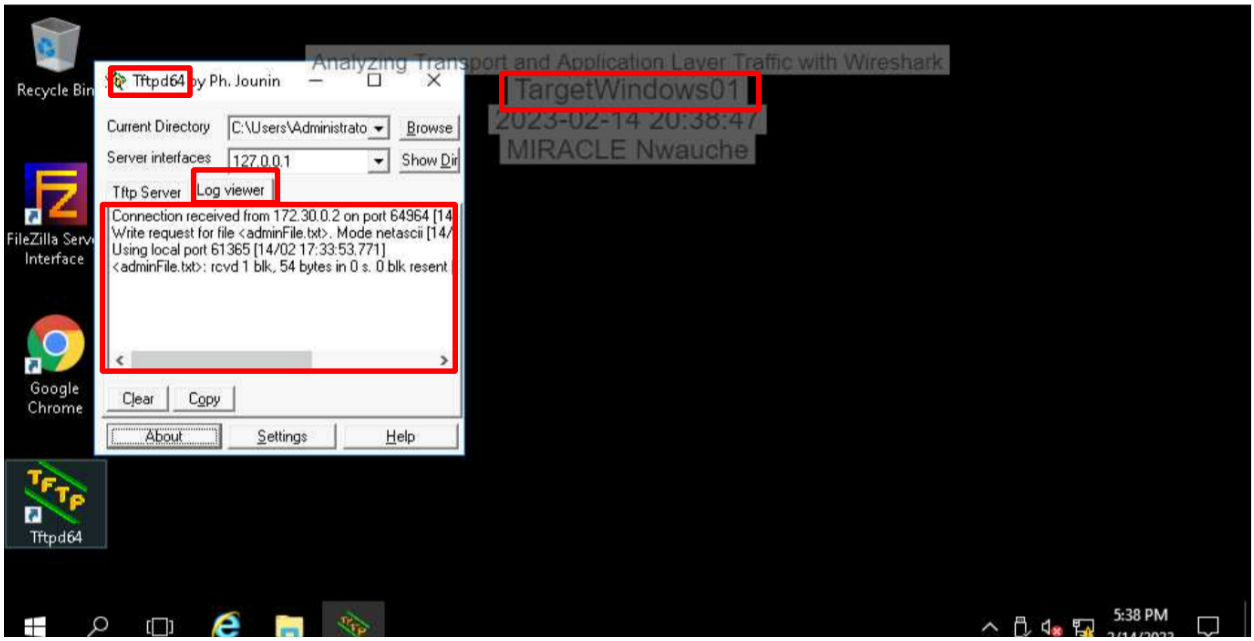

Figure 2, Section 1, Part 1: *TFTP server log*

In the TargetWindows01 VM machine, after opening the tftpd64 application the directory pops up on the screen entailing the connection received from the vWorkstation IP address request for the admin file. Tftp is usually only used in internal networks between Clients and Servers.

45. **Make a screen capture** showing the **successful SFTP file transfer**.

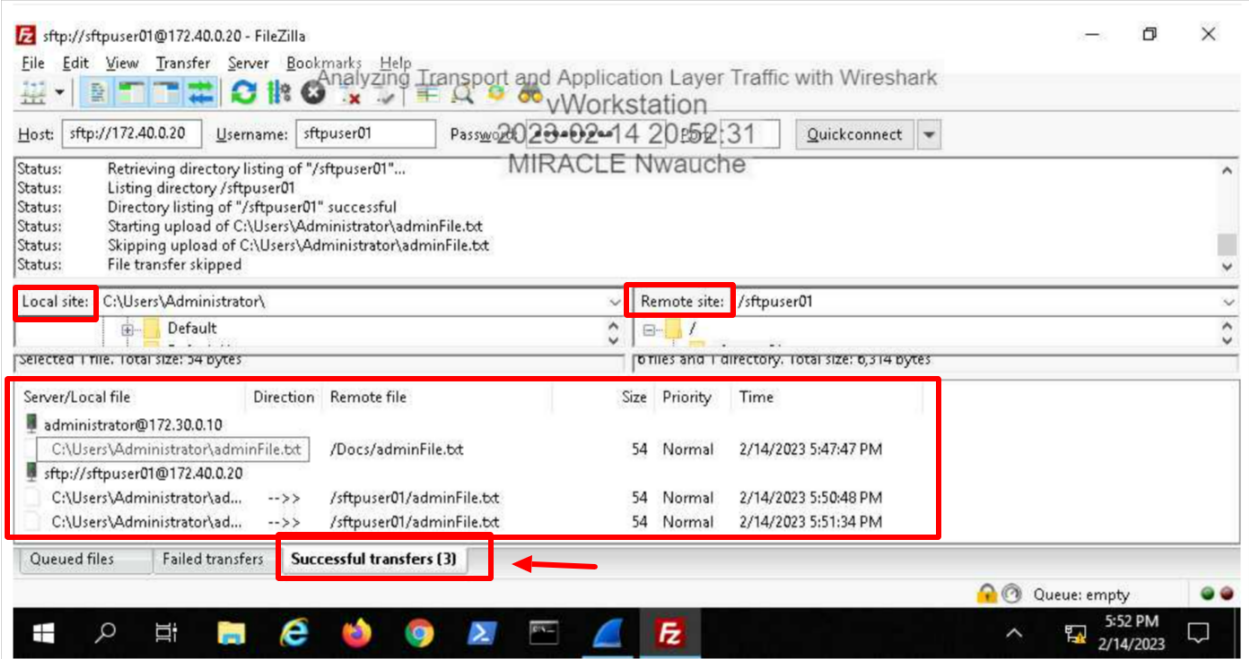

Figure 3, Section 1, Part 1: *Successful SFTP file transfer*

Successful SFTP file transfer, once we targeted the Remote and Local files and uploaded the file, we are gifted with 3 successful file transfers ready for our disposal. This shows a SFTP traffic Transfer between vWorkstation and TargetLinux.

Part 2: Perform Protocol Analysis while using Wireshark

5. **Make a screen capture** showing the ***Last Login:*** **information in the Packet Details pane**.

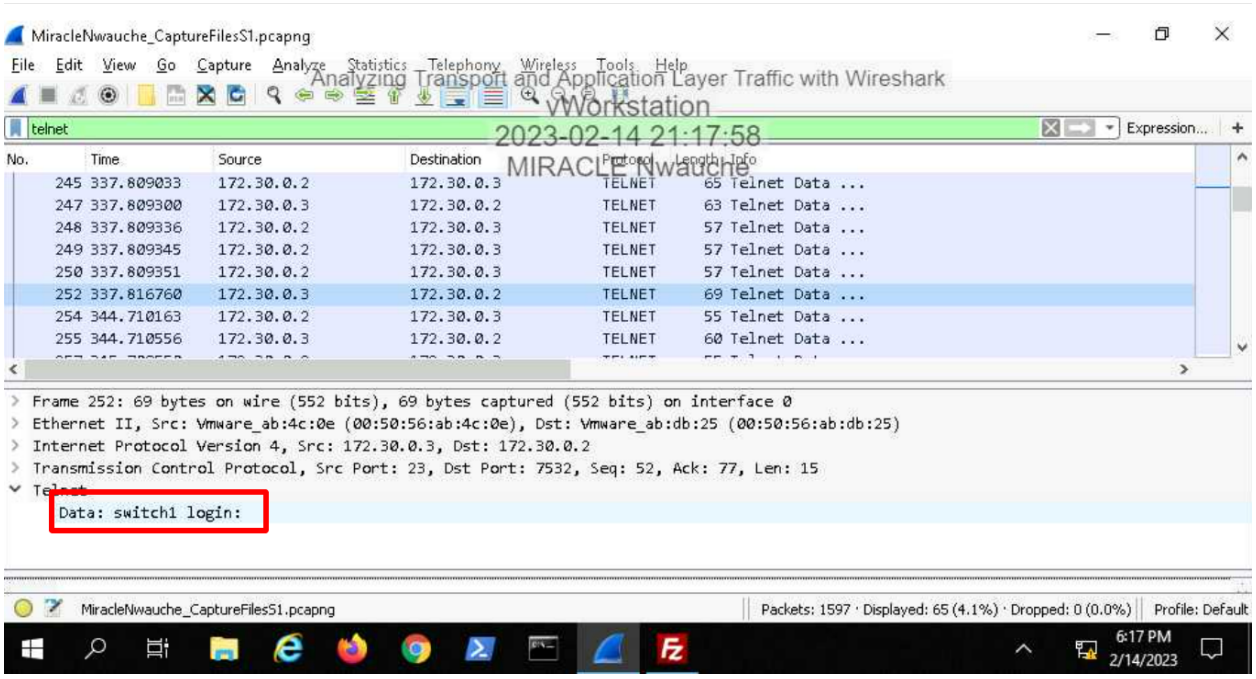

Figure 4, Section 1, Part 2: *Last Login: Information*

After scrolling through Telnet we find in the packet details view the *switch1 login:* As you scroll through more of the Telnet packets you will find the password to the corresponding packets. This information will be useful later in the lab because some packets can be disguised as other packets and cause errors at the password portion of the login.

11. **Make a screen capture** showing the **SSHv2 encryption and mac selections for the SSH connection**.

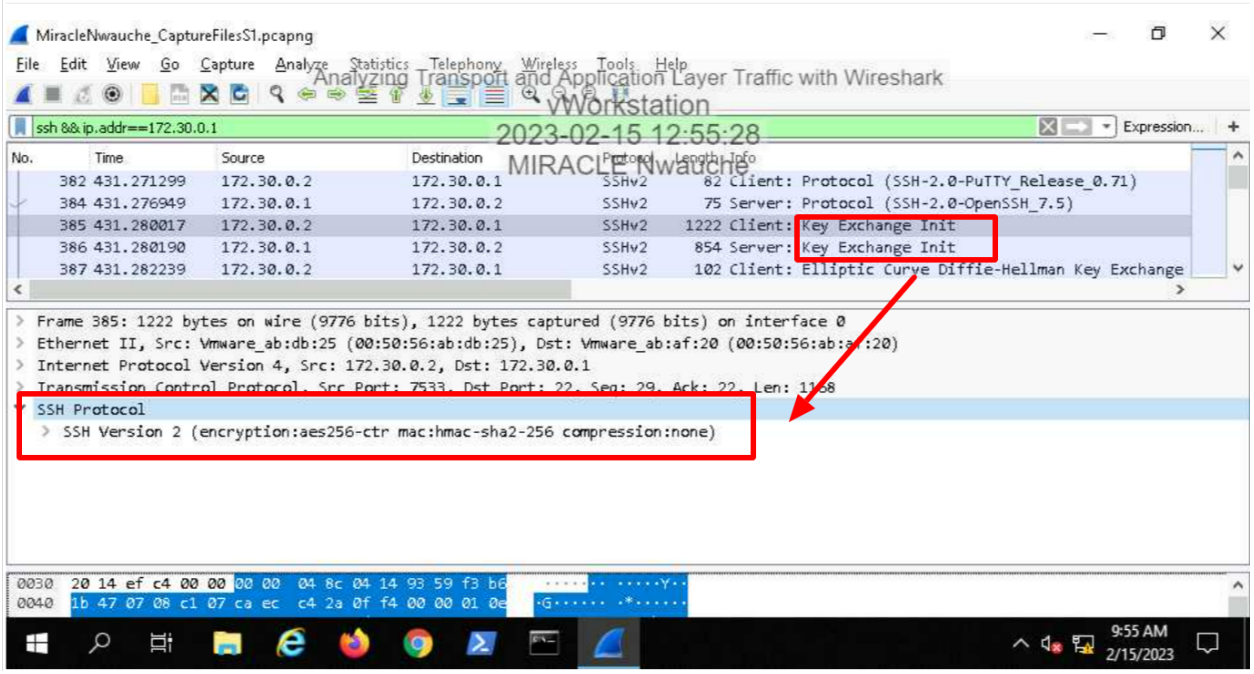

Figure 5, Section 1, Part 2: *SSHv2 encryption and mac selections*

This picture shows one of the key exchange packets and their contents, taking a closer look at the SSH protocol observing the MAC for encryption. In this case (SSH version) the MAC actually refers to Message Authentication Code rather than the Media Access Control.

16. **Make a screen capture** showing the **highlighted (encrypted) data in the Packet Bytes pane**.

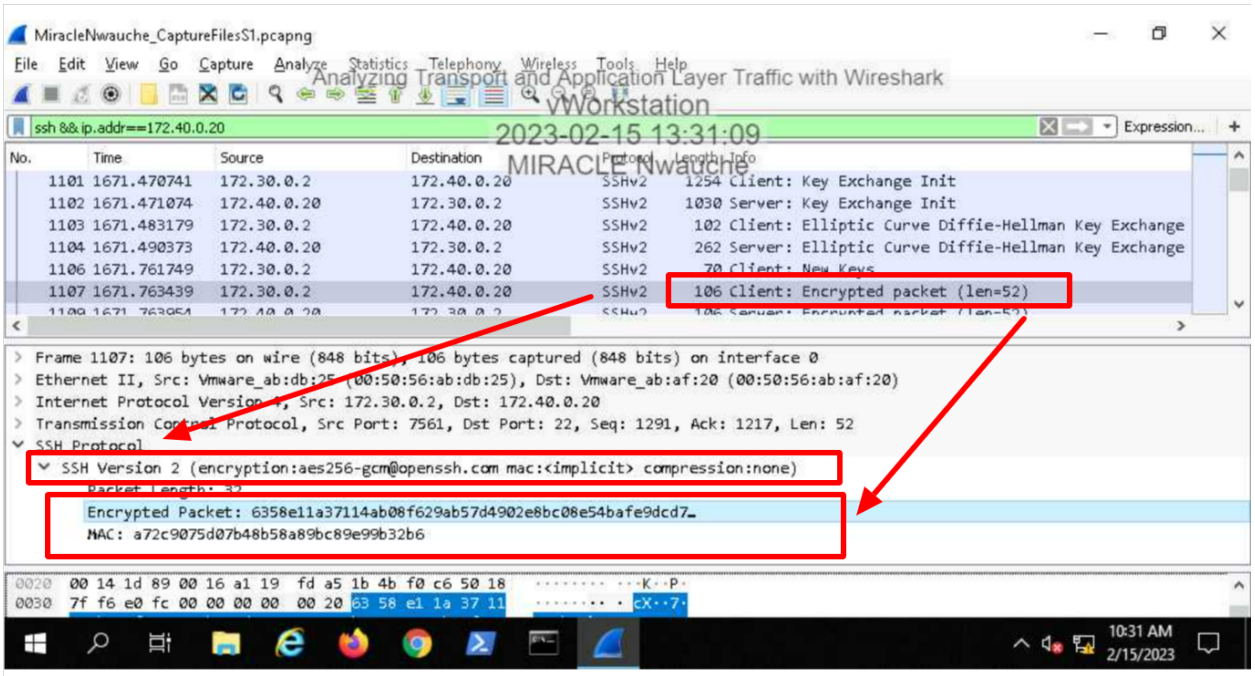

Figure 6, Section 1, Part 2: *Encrypted data*

For the filtered list in Wireshark, the user is prompted to find the first encrypted packer in the packet pane list. After selecting the Encrypted packet pane. More information is displayed about the encryption. Information is contained in the packet's encrypted payloads. HEX/Ascii data corresponds to the information below.

20. **Make a screen capture** showing the **Destination Port used for the initial TFTP transfer request**.

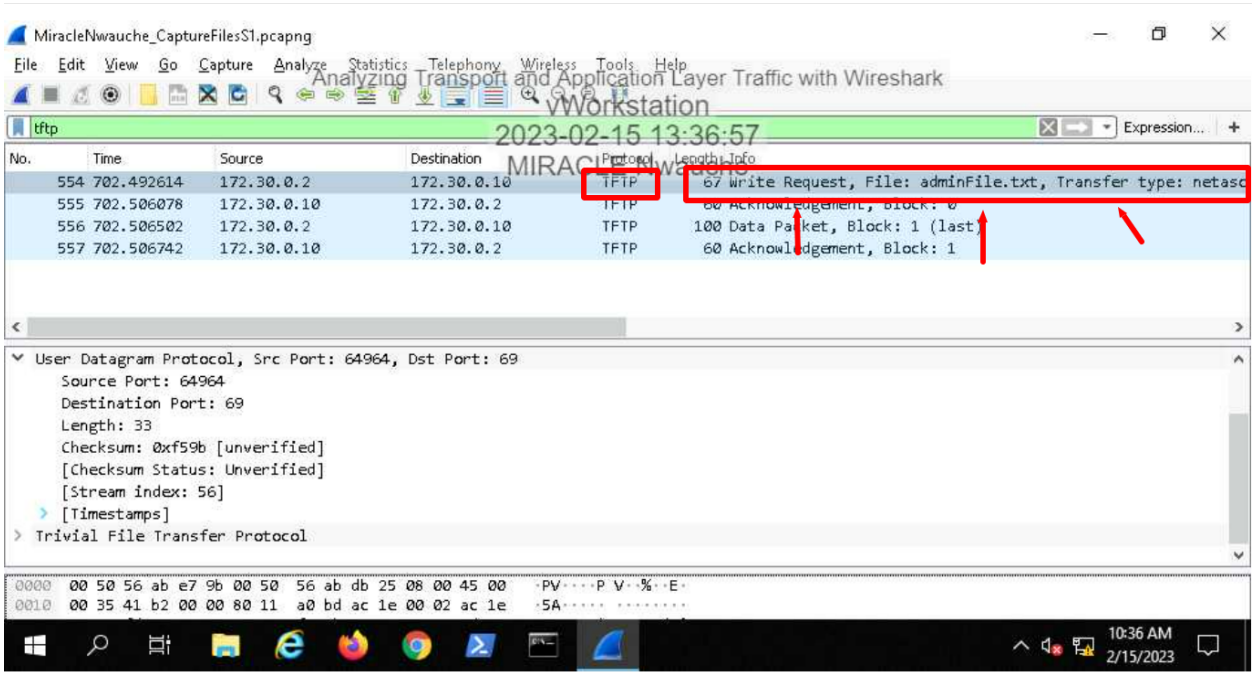

Figure 7, Section 1, Part 2: *TFTP transfer request*

The diagram of the TFTP protocol shows only a few packets captured and transferred from the internal network. The User Datagram Protocol (UDP) shows the destination of the selected Port. TFTP uses UDP as a connectionless Transport Layer Protocol, this does not guarantee the delivery like the TCP does though.

25. **Make a screen capture** showing the **passive port specified by the FTP server in the Packet Details pane**.

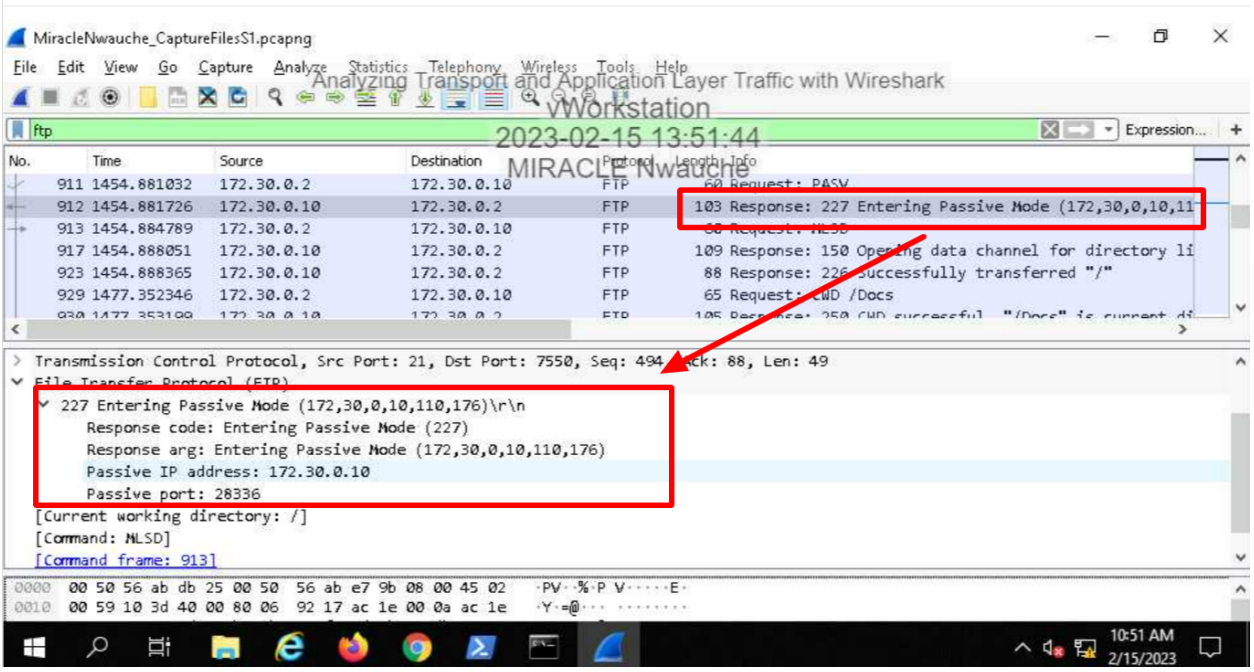

Figure 8, Section 1, Part 2: *Passive Port (FTP)*

After filtering the FTP packets out in Wireshark we can see a new Feature in the packet details pane view. FTP (File Transfer Protocol). This shows us the response code classifying the mode as Active or Passive. Typically the destination port should match the Passive Port Value.

29. **Make a screen capture** showing the **Destination Port field value in the Packet Details pane**.

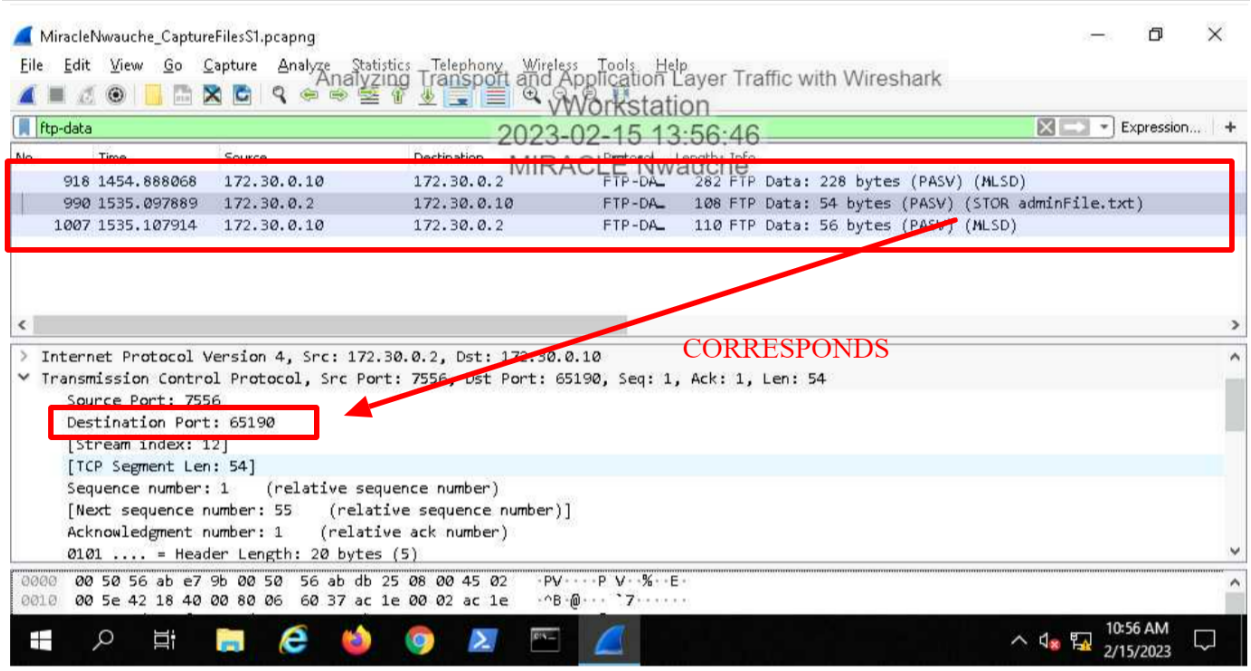

Figure 9, Section 1, Part 2: *Destination Port field value*

Destination port field value in the packet details pane. Something peculiar I noticed was in the Info section of the selected packet. It shows the *adminFile.txt* transferred from the TargetWindow01 to vWorkstation: There were successful files transferred over.

Section 2: Applied Learning, Part 1: Configure Wireshark and Generate Network Traffic

7. **Make a screen capture** showing the **successfully executed netcat command**.

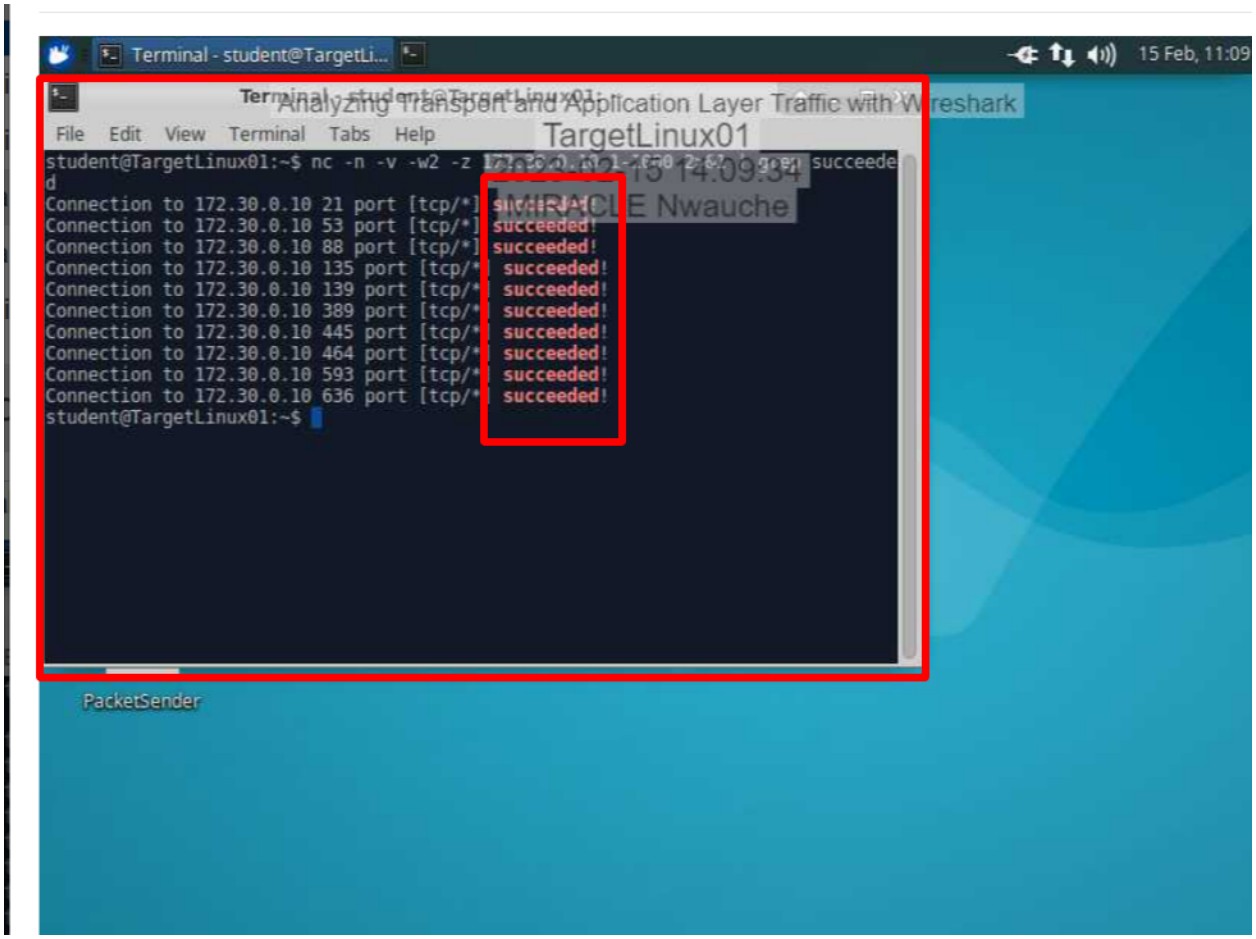

Figure 10, Section 2, Part 1: *Netcat*

After the executed command from Linux the row of connected ports are displayed after the port number there is a message telling the user TCP protocols succeeded. Command used was *-n -v -w2 -z 2>&1 | grep succeeded.*

20. **Make a screen capture** showing the **successful transfer in the Command Prompt output**.

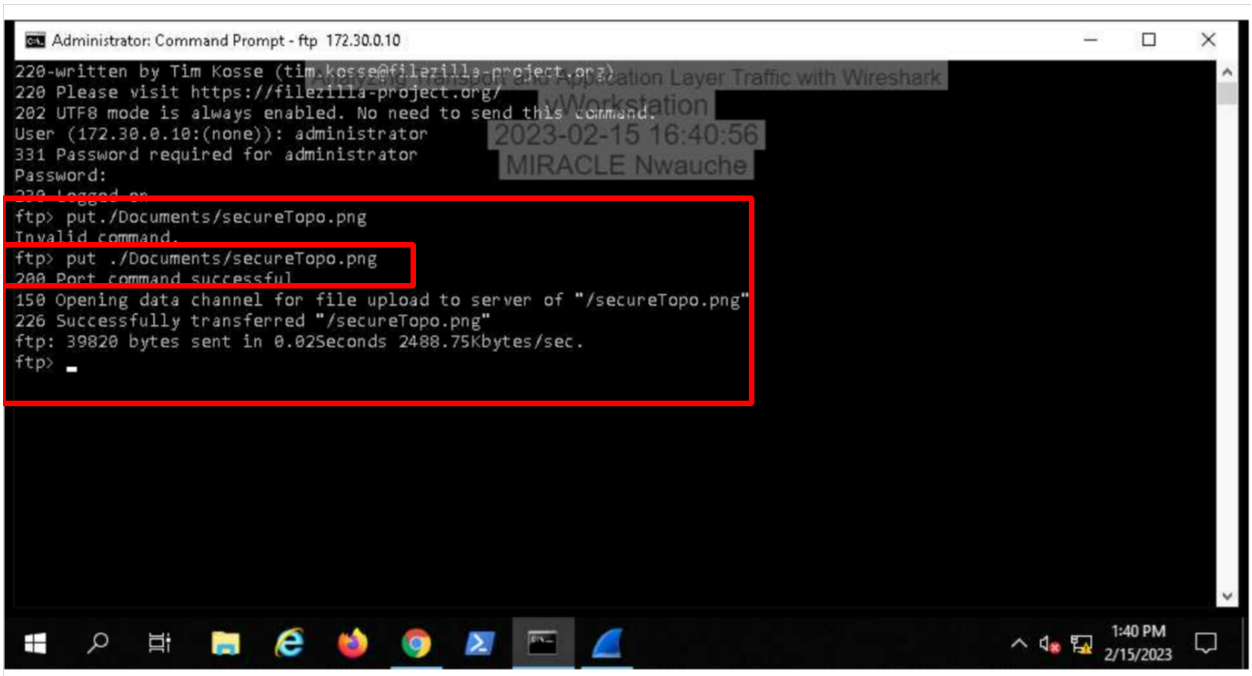

Figure 11, Section 2, Part 1: In the moment, trying to execute this command took a longer time than necessary. Look at the part of the Command prompt where I input the FTP code: even though those commands look different, before this I typed the same exact code as the one that connected the port and it kept saying “Login information failed!” After this portion of the lab I figured out it was the reset packets that were causing this issue.

Section 2, Part 2: Performing protocol Analysis Using Wireshark

5. **Make a screen capture** showing the **TCP flags set in the Packet Details pane for the first RST packet**.

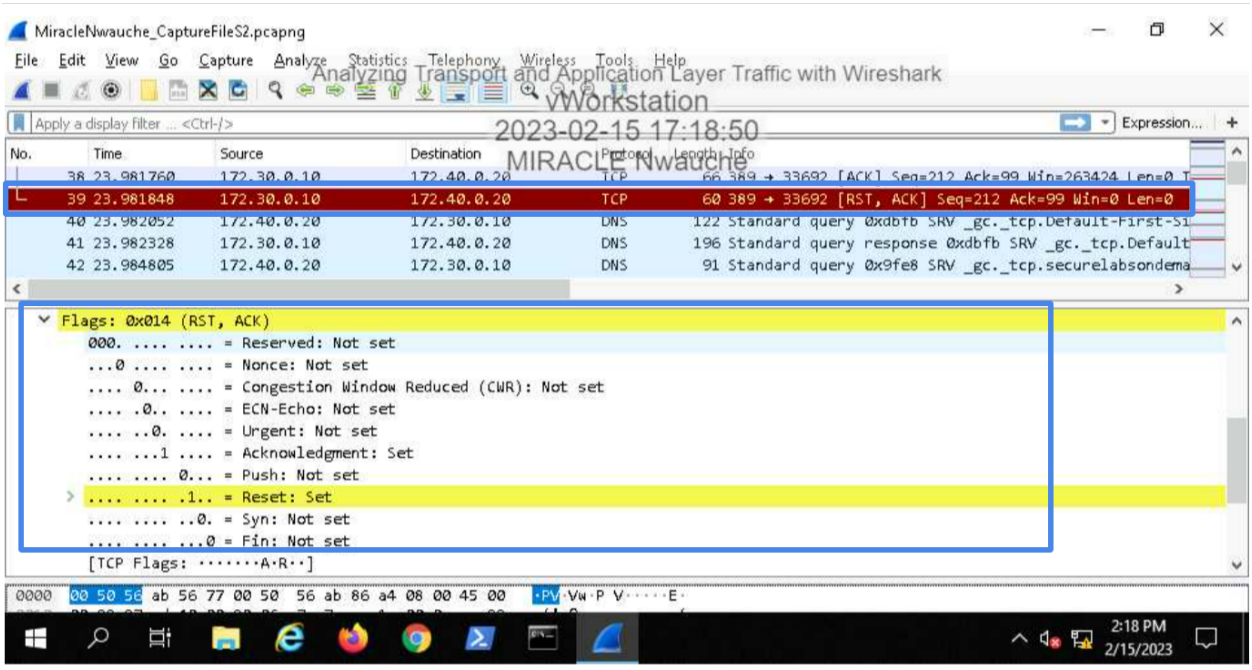

Figure 12, Section 2, Part 2: *TCP flags in RST packet*

The red packet is shown in the packet list where the RST packet takes place in the stream. The flags are shown in the packet view list 0 for “On” and 1 for “Off”. Reset flag for RST packets sent as a TCP header typically seen from an aborted connection. They appear as a forged connection packet which warrants the name TCP reset attacks.

10. **Make a screen capture** showing the **FIN and ACK flags set in the Packet Details View**.

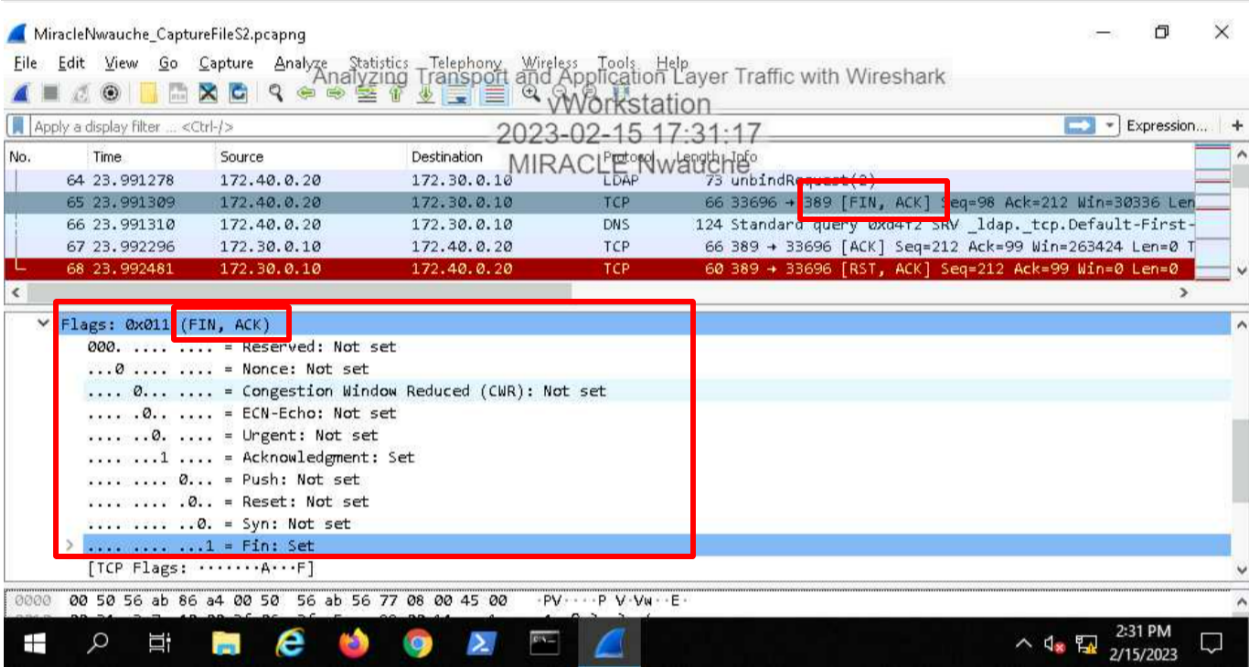

Figure 13, Section 2, Part 2: *FIN/ACK packets*

The FIN,ACK packet from the packet list view shows the flags with the open and closed ports. Although having a FIN packet with ACK is not necessary, it is just used for another form of clarification with the packet that it was sent and received successfully.

16. **Make a screen capture** showing the **highlighted Encrypted Application Data in the Packet Bytes pane**.

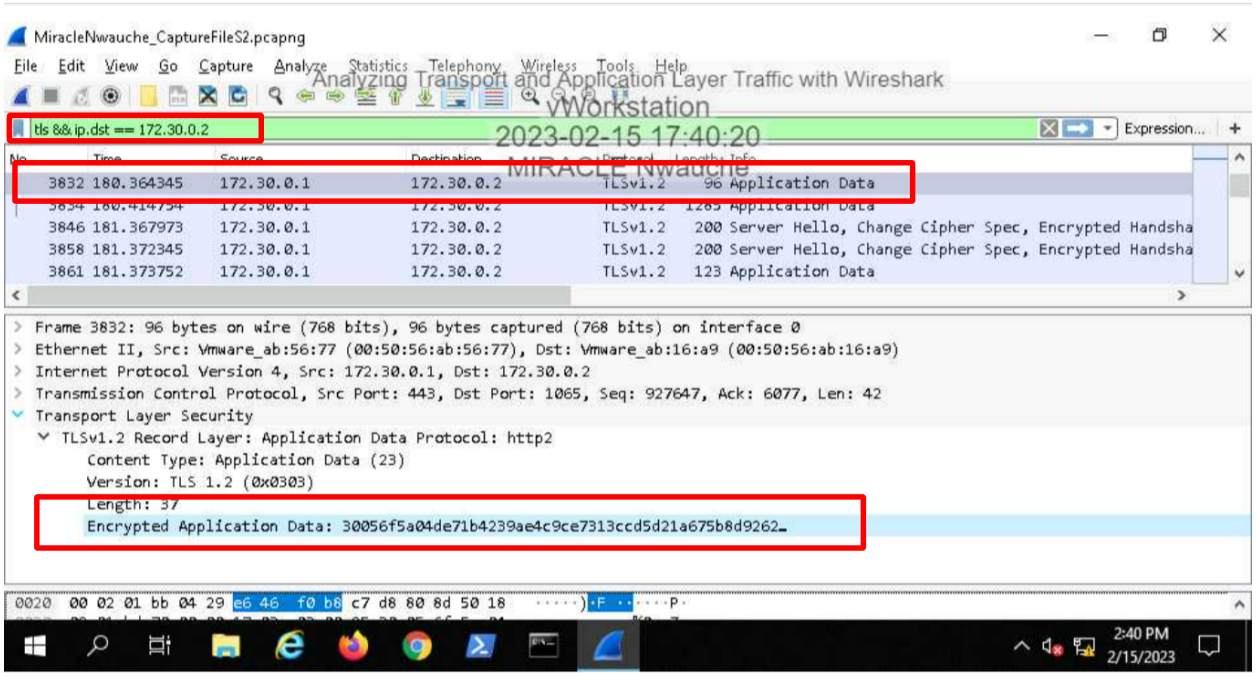

Figure 14, Section 2, Part 2: *Encrypted Application Data in the Packet Bytes pane*

After filtering the packet list to conform to the specific data that is TLS and the IP destination of our host network, we can now look for the packets that resemble Application Data. Within those packets you can see the encrypted data withheld in the information column. This is a Hexadecimal code that only a computer would be able to read but up to the user to decipher it!

22. **Make a screen capture** showing the **certificate details in the Packet Details pane**.

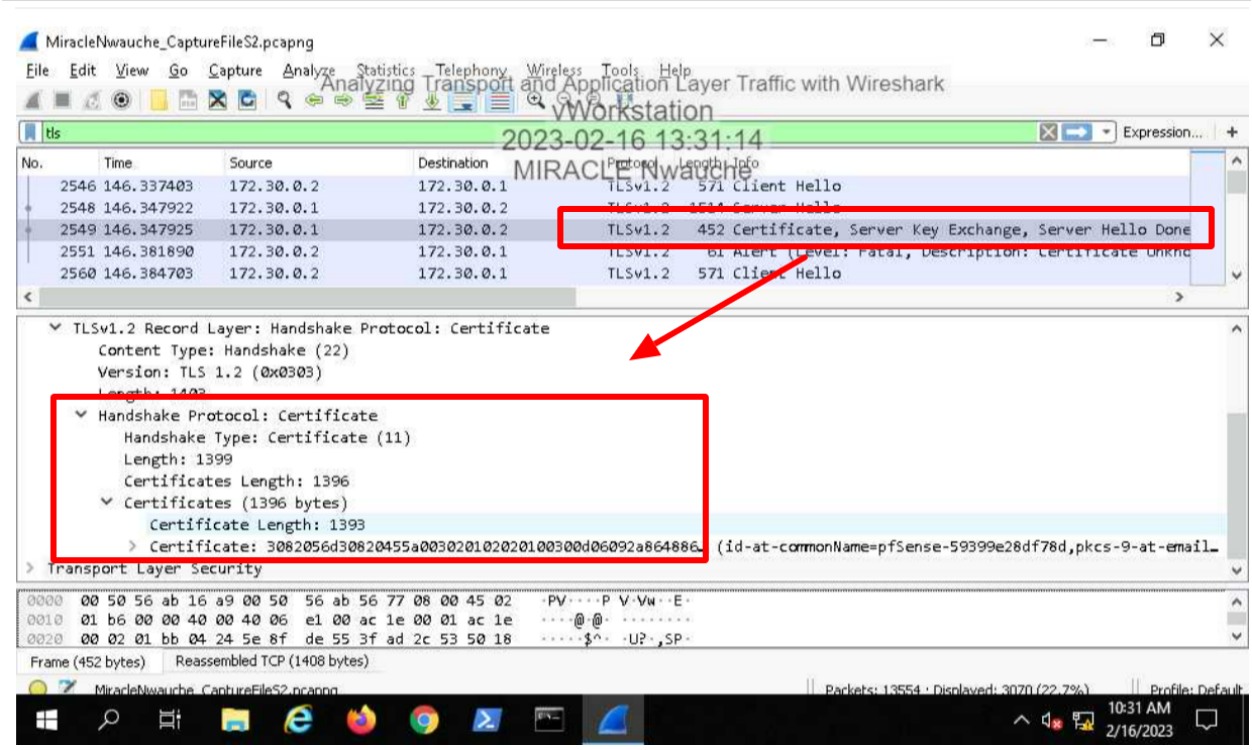

Figure 15, Section 2, Part 2: *certificate details*

The selected packet in the frame has a certificate to send to the client for authentication that is initiated by the handshake protocol. When the drop down list is examined, the length of the handshake and certificate (in bytes) are shown with an encryption code.

25. **Make a screen capture** showing the **complete set of data in the TCP Stream window**.

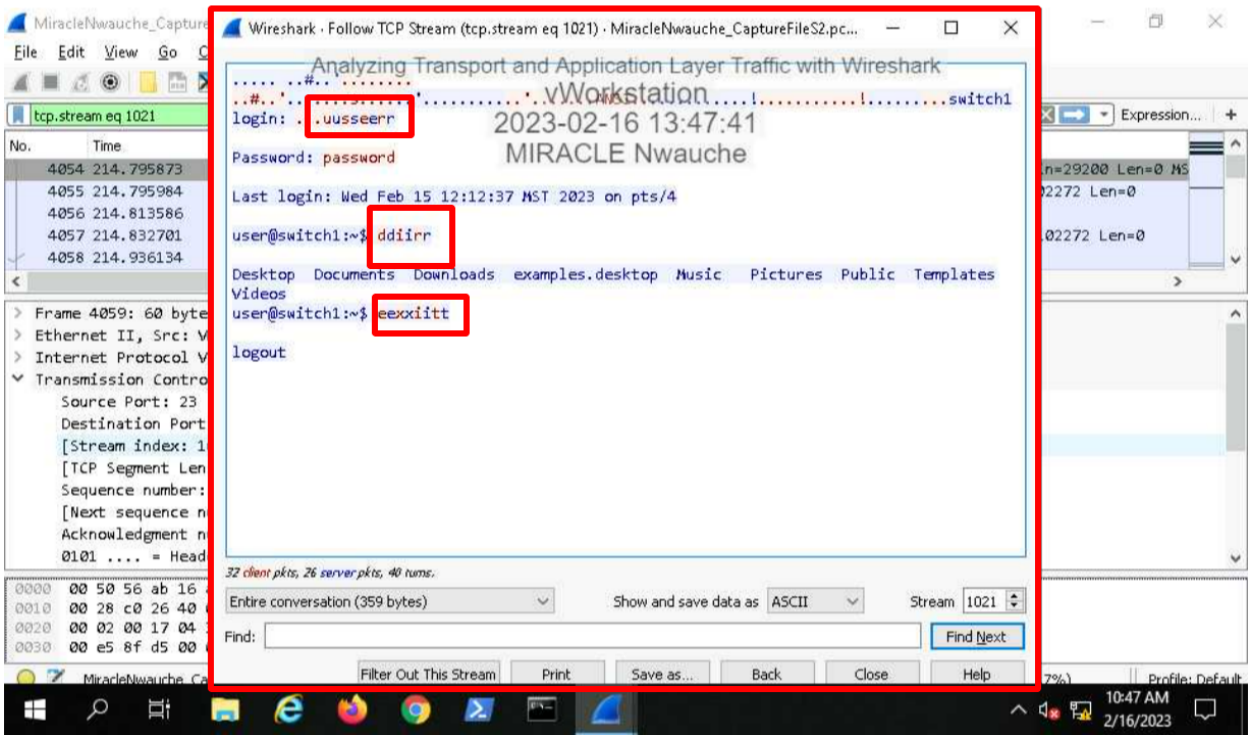

Figure 16, Section 2, Part 2: *TCP stream*

This capture of the following TCP stream shows the encrypted message of the TCP packet. The red characters represent the message from the client, inversely the blue shows a message from the server. The client sends a message first, followed by the echo of the server.

36. **Make a screen capture** showing the **reconstituted PNG file**.

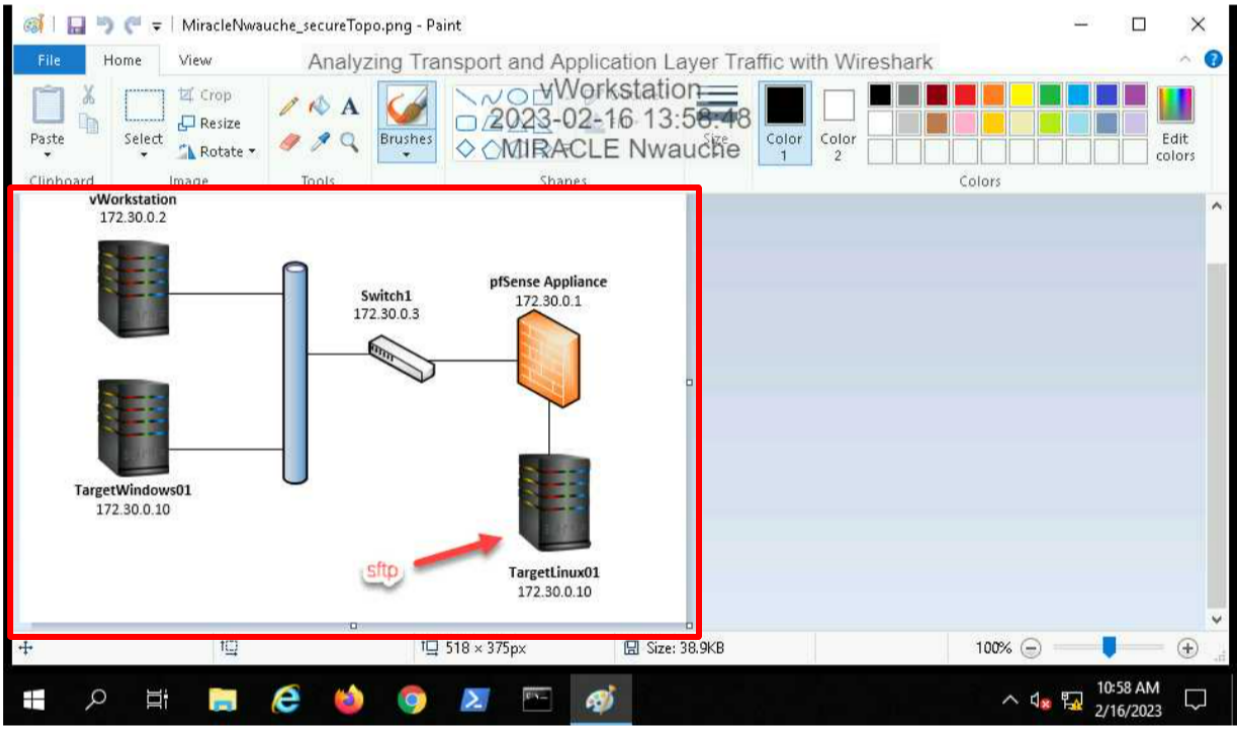

Figure 17, Section 2, Part 2: This picture is the PNG revamped version of the payload that was put back together from the fragmented packets. The picture displays the connection of the VM machines in the server network, they are both connected to switch1 and on the other side connected to the pfsense Appliance. The SFTP is directly linked with the Target Linux01 system.

SECTION 3, Part 1: Locate a Target Rar file Transfer in the Packet Capture

Record the file signature you used to find the RAR archive.

The file signature I used to find the RAR archive was the Hexadecimal value 52 61 72 21 1A 07 01 00

Record the name of the correct RAR archive file.

STOR Target_Payload2.rar

Part 2: Reassemble the RAR Archive from its Constituent Bytes

Make a screen capture showing the contents of the tar file.

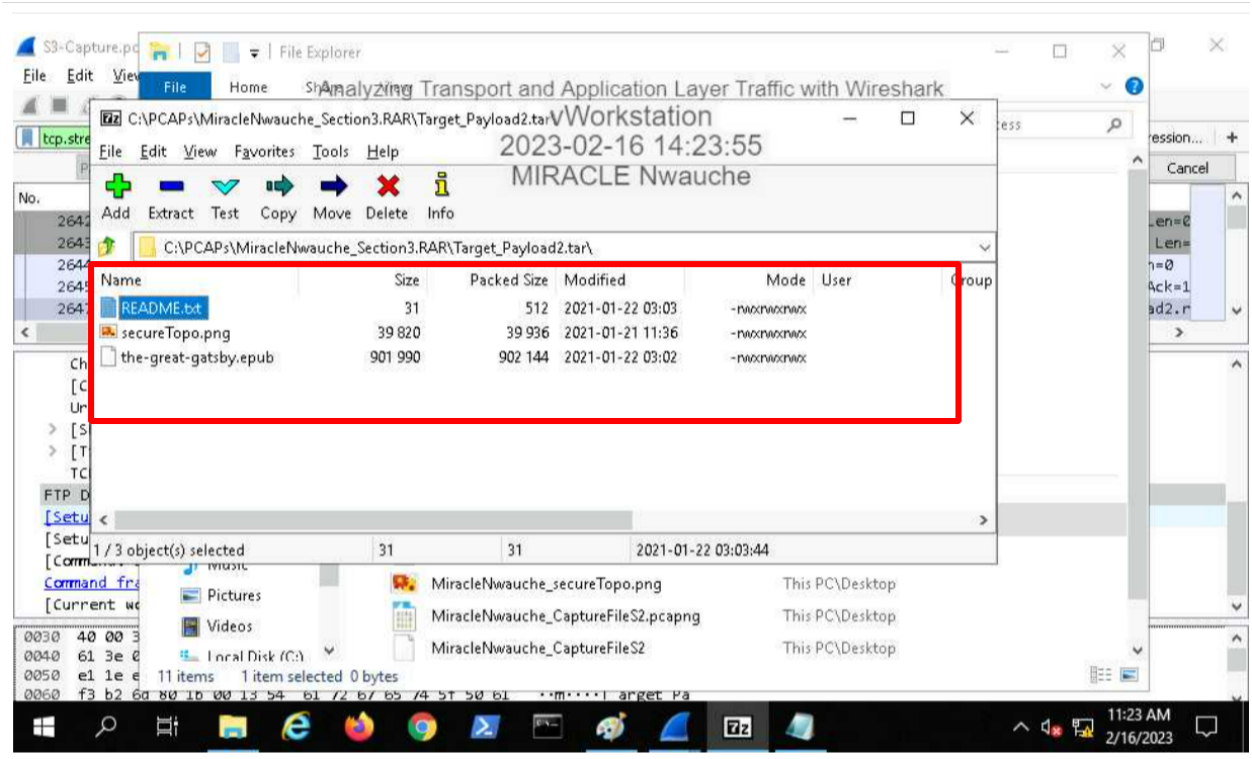

Figure 18, Section 3, Part 2: This capture shows the contents of the tar file saved to PCAPS after extracting the file as Raw data and saving it to the directory. Once opened, there are already files in the subdirectory with the README.txt file that contains the lost password.

Record the passphrase discovered in the **README.txt file**.

The code is {JBL-80802600-SaaS}