Roja Atashkar
DN : Prof. Pakravan's Data Network
July 7, 2024

Data Network Project(FTP)

Question 1. Client and Server Implementation (65 points)
The code is given in the folder Part 1.

Question 2. FTP Protocol Understanding (15 points)

1. Investigate other protocols that are used for file transfer and compare them with FTP.

FTP (File Transfer Protocol) is a well-established protocol for file transfer, but it has limitations. Here's a comparison of FTP with other common protocols:
Secure vs. Insecure:
FTP: Inherently insecure, transmits data and passwords in plain text. Consider FTPS (FTP with SSL/TLS) or SFTP for secure alternatives.
SFTP (SSH File Transfer Protocol): Built on SSH (Secure Shell) for secure file transfer. Encrypts data and uses secure authentication.
HTTPS (Hypertext Transfer Protocol Secure): Secure version of HTTP, often used for web downloads. Encrypts data transmission.
Ease of Use:
FTP: Relatively easy to use, basic FTP clients are built into many operating systems.
SFTP: Requires an SFTP client application, may be less familiar to some users.
HTTP/HTTPS: Most web browsers can handle file downloads over HTTPS, very user-friendly.
Features:
FTP: Offers basic file transfer functionality.
SFTP: Supports secure file transfer, some clients offer features like directory listings and remote file management.
HTTP/HTTPS: Primarily for downloading files from web servers, limited file management features. Other Options:
WebDAV (Web Distributed Authoring and Versioning): Enables collaborative editing and versioning of files on a web server.
TFTP (Trivial File Transfer Protocol): Simple protocol for transferring small files, often used for network booting.
Choosing the Right Protocol:
Security is paramount: Use SFTP or FTPS for sensitive data transfers.
Simplicity is key: FTP or HTTP might be suitable for casual file sharing.
Collaboration needed: Consider WebDAV for shared editing of files.

2. What is the commonly used transport protocol for file transfers? Is it possible to use UDP as the transport layer protocol?

The most commonly used transport protocol for file transfers is Transmission Control Protocol (TCP). TCP is a reliable, connection-oriented protocol. This means it establishes a connection between the sender and receiver, verifies data arrives correctly through error checking and retransmission, and guarantees the order of data packets. This reliability is crucial for ensuring complete and accurate file transfers, especially for large files.
UDP (User Datagram Protocol) can be used for file transfers, but it's less common due to its limitations:
Unreliable: UDP doesn't guarantee delivery, error checking, or order of packets. This can lead to missing

data or corrupted files.
Connectionless: UDP doesn't establish a connection, making it less suitable for complex file transfers.
However, UDP can be useful in specific scenarios:
Streaming media: Since order isn't critical for real-time media like audio or video, UDP can handle basic data transfer efficiently.
Simple file transfers: For very small, non-critical files where missing a bit isn't a big deal (e.g., configuration files), UDP might be acceptable.
In summary, TCP is the go-to transport protocol for reliable file transfers due to its connection-oriented and error-checking nature. UDP has niche applications but is generally not preferred for file transfer due to its unreliability.

3. What are the drawbacks of FTP that have made it fairly obsolete on the modern web?

FTP, while a well-established protocol, has several drawbacks that make it less than ideal for the modern web, especially when security is a concern. Here are the main reasons FTP is considered somewhat obsolete:

1. Lack of Security: Standard FTP transmits data and passwords in plain text. This means anyone snooping on the network connection can easily intercept sensitive information. This is a major security risk, especially for transferring confidential data.

2. Limited Authentication: Basic FTP relies on username and password for authentication, which is susceptible to brute-force attacks or credential theft.

3. No Encryption: Data transferred via standard FTP is not encrypted, making it vulnerable to eavesdropping and tampering.

4. Outdated Features: FTP lacks features like automatic file transfer resumption after interruptions or robust progress tracking, which can be frustrating for users.

5. Difficulty with Compliance: Many regulations and compliance standards require data transfers to be encrypted. Standard FTP doesn't meet these requirements.

Modern Alternatives: Secure alternatives like FTPS (FTP with SSL/TLS) and SFTP (SSH File Transfer Protocol) address these shortcomings by offering encryption and stronger authentication methods. Additionally, protocols like HTTPS (secure web downloads) and WebDAV (collaborative file editing) provide secure and feature-rich options for specific needs.

While FTP servers might still exist for legacy purposes, secure alternatives are generally preferred for modern web file transfers due to the increased emphasis on data security and robust features.

4. What are the disadvantages of using active mode FTP? How the passive mode can handle these problems?

Active mode FTP has a few disadvantages that make passive mode the preferred option for most file transfers. Here's a breakdown of the issues and how passive mode solves them:
Disadvantages of Active Mode FTP:
Firewall Issues: In active mode, the server initiates a connection back to the client on a random high port.

This is often blocked by firewalls on the client side, as they typically only allow outgoing connections and view incoming connections from unknown ports with suspicion. This can prevent the data transfer from even starting.

NAT (Network Address Translation) Issues: If the client is behind a Network Address Translation (NAT) device like a router, the server might use the private IP address of the client instead of the public IP for the return connection. This mismatch prevents the data transfer from reaching the client.

Client Configuration: Active mode might require the client to configure firewalls to allow incoming connections on random high ports, which can be a security risk and adds complexity for users.

How Passive Mode FTP Handles These Problems:

Client Initiates Connections: In passive mode, the client initiates both control and data connections. This means the client can choose a port number and ensure its firewall allows connections on that port. This eliminates firewall issues on the client side.

Server Listens for Connection: The server listens on a randomly chosen high port for the data connection from the client. Since the server is expecting this connection, it won't be blocked by firewalls on the server side.

No Client Configuration Needed: Passive mode doesn't require special client configuration for firewalls, making it more user-friendly.

In summary, passive mode FTP solves the firewall and NAT traversal issues that plague active mode by shifting the responsibility of initiating the data connection to the client, allowing for a smoother and more secure file transfer process.

5. In FTP, how is data transfer security guaranteed? Is there even a default security measure for FTP? Investigate SFTP, FTPS, and FTP over SSH protocols in terms of their security

Standard FTP, unfortunately, offers very little in terms of built-in security. Here's a breakdown:

No Encryption: Data transferred via FTP is transmitted in plain text. This means anyone snooping on the network can see usernames, passwords, and the actual file contents.

Weak Authentication: Basic FTP relies solely on username and password for authentication, which is vulnerable to brute-force attacks or credential theft.

Alternatives with Enhanced Security:

Since standard FTP lacks security features, several secure alternatives have emerged:

SFTP (SSH File Transfer Protocol):

Leverages SSH (Secure Shell) for secure file transfer.
Encrypts all data in transit, protecting it from eavesdropping.
Uses strong public key cryptography for authentication, eliminating the need to transmit passwords in plain text.

Considered one of the most secure options for file transfer.
FTPS (FTP with Secure Sockets Layer/Transport Layer Security):

Combines the traditional FTP file transfer commands with SSL/TLS encryption.
Encrypts data during transfer, similar to SFTP.
Authentication methods can vary depending on the implementation, but some might still rely on usernames and passwords (less secure).
FTP over SSH (also known as SCP):

Not a separate protocol, but a combination of tools using SSH for secure transfer. Uses SSH for secure connection and authentication.
Relies on basic FTP commands for file transfer functionality.
Offers a secure alternative to standard FTP but might require command-line familiarity. Security Comparison:

Strongest Security: SFTP takes the lead with its combination of robust encryption and public key authentication.
Improved Security: FTPS offers encryption, but authentication methods might vary in strength.
Choose implementations with strong password protocols or public key options.
Limited Security: FTP over SSH provides a secure connection but relies on basic FTP commands, so the security depends on the chosen authentication method within the FTP layer.
Choosing the Secure Option:

For the most secure file transfers, SFTP is the preferred choice due to its comprehensive encryption and strong authentication methods. FTPS can be a good alternative if SFTP compatibility is an issue, but ensure it uses strong authentication methods. When using FTP over SSH, make sure the chosen authentication method within the FTP layer is secure.

6.With Wireshark, you can observe network packets traversing any network interface. Use Wireshark to investigate packets while you are downloading a file from your FTP server. Note that you should run FTP on the loopback interface. What is the maximum size of a TCP packet containing data? Use the following command on Linux-based machines to create an arbitrary large file.

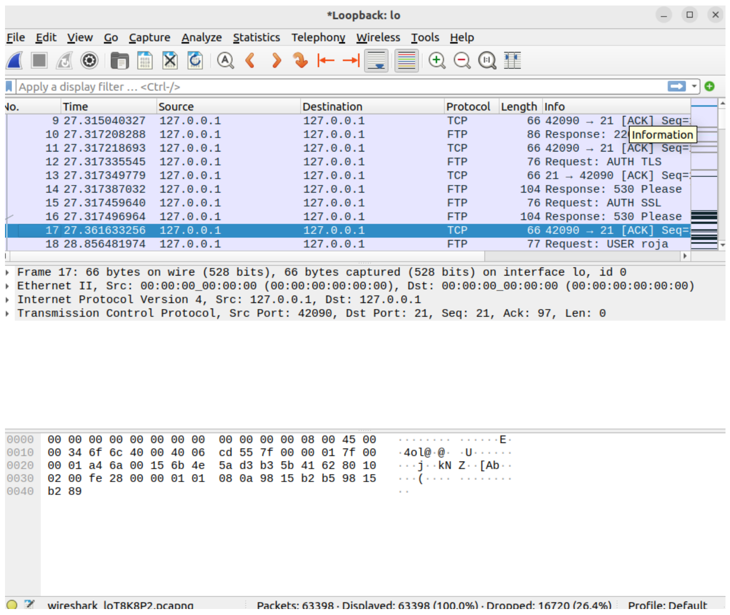
I used vsftpd to start an FTP server on Ubuntu and FileZilla to download the file.txt file.
The theoretical maximum size of a TCP packet is 64KB (65,535 bytes). However, the actual maximum data size will be limited by the Maximum Transmission Unit (MTU) of the network interface. Common Ethernet MTUs are typically 1500 bytes.

Wireshark Analysis :

Wireshark will display captured packets. By filtering for the FTP protocol and examining the TCP segment size, you'll observe that the data portion of the packets won't be close to 64KB. Instead, it is around 66 bytes

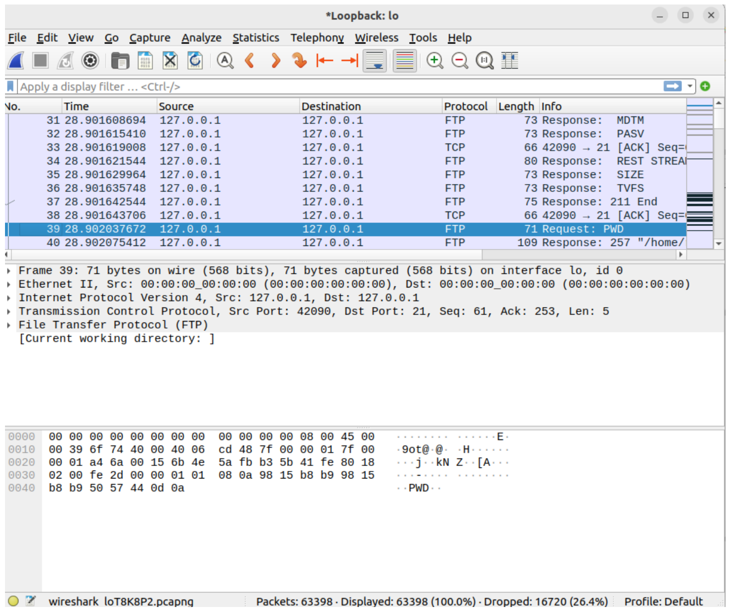
Conclusion:

While TCP can theoretically handle data packets up to 64KB, real-world network limitations like MTU restrict the actual data size in a packet. Wireshark helps visualize this by showing the actual TCP segment size during the FTP download, which will be constrained by the network's MTU.


*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Information

```
No.  Time          Source     Destination  Protocol  Length  Info
   9 27.315040327  127.0.0.1  127.0.0.1    TCP        66 42090 → 21 [ACK] Seq=
  10 27.317208288  127.0.0.1  127.0.0.1    FTP        86 Response: 22
  11 27.317218693  127.0.0.1  127.0.0.1    TCP        66 42090 → 21 [ACK] Seq=
  12 27.317335545  127.0.0.1  127.0.0.1    FTP        76 Request: AUTH TLS
  13 27.317349779  127.0.0.1  127.0.0.1    TCP        66 21 → 42090 [ACK] Seq=
  14 27.317387032  127.0.0.1  127.0.0.1    FTP       104 Response: 530 Please
  15 27.317459640  127.0.0.1  127.0.0.1    FTP        76 Request: AUTH SSL
  16 27.317496964  127.0.0.1  127.0.0.1    FTP       104 Response: 530 Please
  17 27.361633256  127.0.0.1  127.0.0.1    TCP        66 42090 → 21 [ACK] Seq=
  18 28.856481974  127.0.0.1  127.0.0.1    FTP        77 Request: USER roja

Frame 17: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 42090, Dst Port: 21, Seq: 21, Ack: 97, Len: 0

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 34 6f 6c 40 00 40 06  cd 55 7f 00 00 01 7f 00   ·4ol@·@· ·U······
0020  00 01 a4 6a 00 15 6b 4e  5a d3 b3 5b 41 62 80 10   ···j··kN Z··[Ab··
0030  02 00 fe 28 00 00 01 01  08 0a 98 15 b2 b5 98 15   ···(···· ········
0040  b2 89                                              ··

wireshark_loT8K8P2.pcapng    Packets: 63398 · Displayed: 63398 (100.0%) · Dropped: 16720 (26.4%)    Profile: Default
```

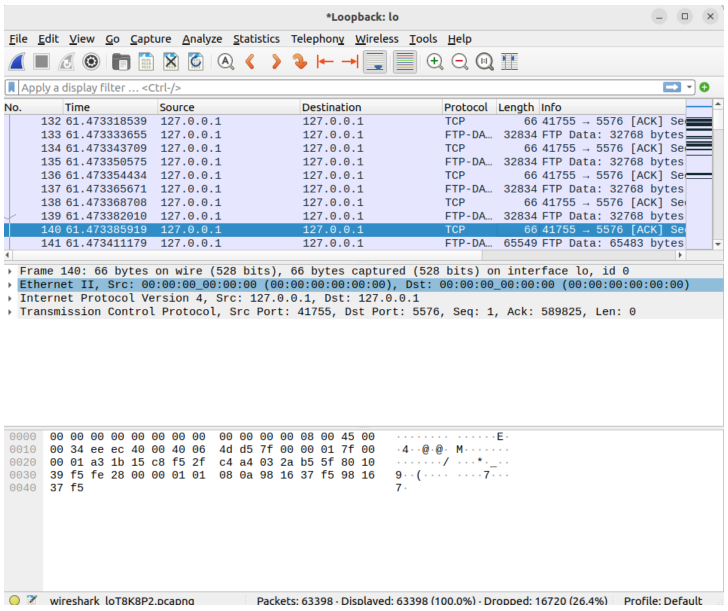
FIGURE 1. Wireshark captures


*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time          Source     Destination  Protocol  Length  Info
  31 28.901608694  127.0.0.1  127.0.0.1    FTP        73 Response:  MDTM
  32 28.901615410  127.0.0.1  127.0.0.1    FTP        73 Response:  PASV
  33 28.901619008  127.0.0.1  127.0.0.1    TCP        66 42090 → 21 [ACK] Seq=
  34 28.901621544  127.0.0.1  127.0.0.1    FTP        80 Response:  REST STREA
  35 28.901629964  127.0.0.1  127.0.0.1    FTP        73 Response:  SIZE
  36 28.901635748  127.0.0.1  127.0.0.1    FTP        73 Response:  TVFS
  37 28.901642544  127.0.0.1  127.0.0.1    FTP        75 Response: 211 End
  38 28.901643706  127.0.0.1  127.0.0.1    TCP        66 42090 → 21 [ACK] Seq=
  39 28.902037672  127.0.0.1  127.0.0.1    FTP        71 Request: PWD
  40 28.902075412  127.0.0.1  127.0.0.1    FTP       109 Response: 257 "/home/

Frame 39: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 42090, Dst Port: 21, Seq: 61, Ack: 253, Len: 5
File Transfer Protocol (FTP)
 [Current working directory: ]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 39 6f 74 40 00 40 06  cd 48 7f 00 00 01 7f 00   ·9ot@·@· ·H······
0020  00 01 a4 6a 00 15 6b 4e  5a fb b3 5b 41 fe 80 18   ···j··kN Z··[A···
0030  02 00 fe 2d 00 00 01 01  08 0a 98 15 b8 b9 98 15   ···-···· ········
0040  b8 b9 50 57 44 0d 0a                               ··PWD··

wireshark_loT8K8P2.pcapng    Packets: 63398 · Displayed: 63398 (100.0%) · Dropped: 16720 (26.4%)    Profile: Default
```

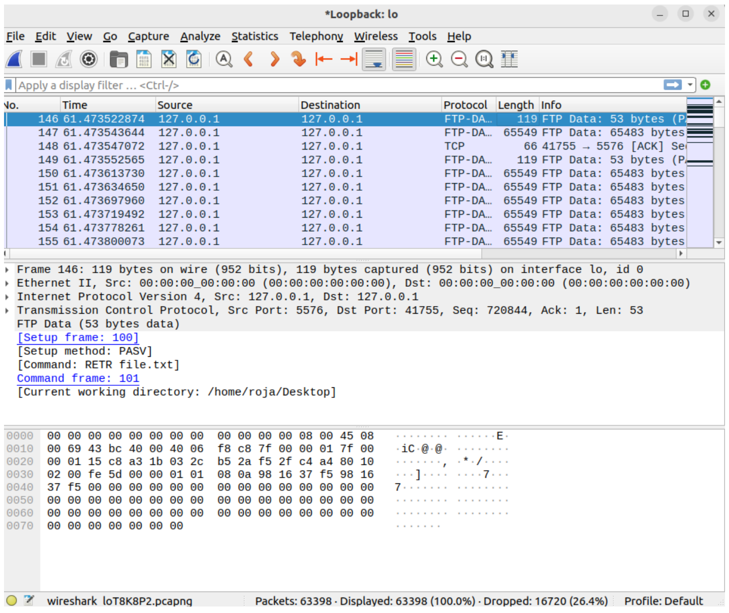
FIGURE 2. Wireshark captures

7. Did you know that Sharif University hosts an FTP server where you can download useful content like engineering programs, drivers, and so on? To visit it, first, you need to set up your Sharif VPN to access it when you are not on campus. Then, refer to this website and you can download tools you might need. While downloading a file from Sharif FTP, run Wireshark and observe the received packets. Are the captured packets similar to the previous part? If not, explain the difference. The data portion of the packets is around 1500 bytes which is different from part 6 which was around 66 bytes.


*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time           Source      Destination  Protocol  Length Info
  132 61.473318539   127.0.0.1   127.0.0.1    TCP          66 41755 → 5576 [ACK] Se
  133 61.473333655   127.0.0.1   127.0.0.1    FTP-DA…   32834 FTP Data: 32768 bytes
  134 61.473343709   127.0.0.1   127.0.0.1    TCP          66 41755 → 5576 [ACK] Se
  135 61.473350575   127.0.0.1   127.0.0.1    FTP-DA…   32834 FTP Data: 32768 bytes
  136 61.473354434   127.0.0.1   127.0.0.1    TCP          66 41755 → 5576 [ACK] Se
  137 61.473365671   127.0.0.1   127.0.0.1    FTP-DA…   32834 FTP Data: 32768 bytes
  138 61.473368708   127.0.0.1   127.0.0.1    TCP          66 41755 → 5576 [ACK] Se
  139 61.473382010   127.0.0.1   127.0.0.1    FTP-DA…   32834 FTP Data: 32768 bytes
  140 61.473385919   127.0.0.1   127.0.0.1    TCP          66 41755 → 5576 [ACK] Se
  141 61.473411179   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes

› Frame 140: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface lo, id 0
› Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
› Transmission Control Protocol, Src Port: 41755, Dst Port: 5576, Seq: 1, Ack: 589825, Len: 0

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 34 ee ec 40 00 40 06  4d d5 7f 00 00 01 7f 00   ·4··@·@· M·······
0020  00 01 a3 1b 15 c8 f5 2f  c4 a4 03 2a b5 5f 80 10   ·······/ ···*·_··
0030  39 f5 fe 28 00 00 01 01  08 0a 98 16 37 f5 98 16   9··(···· ····7···
0040  37 f5                                              7·

wireshark_loT8K8P2.pcapng    Packets: 63398 · Displayed: 63398 (100.0%) · Dropped: 16720 (26.4%)    Profile: Default
```

Figure 3. Wireshark captures

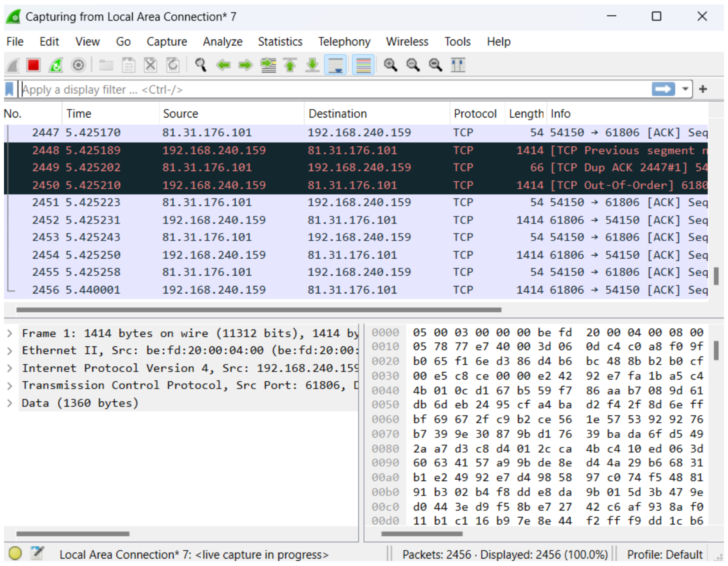

*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time           Source      Destination  Protocol  Length Info
  146 61.473522874   127.0.0.1   127.0.0.1    FTP-DA…     119 FTP Data: 53 bytes (P
  147 61.473543644   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes
  148 61.473547072   127.0.0.1   127.0.0.1    TCP          66 41755 → 5576 [ACK] Se
  149 61.473552565   127.0.0.1   127.0.0.1    FTP-DA…     119 FTP Data: 53 bytes (P
  150 61.473613730   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes
  151 61.473634650   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes
  152 61.473697960   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes
  153 61.473719492   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes
  154 61.473778261   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes
  155 61.473800073   127.0.0.1   127.0.0.1    FTP-DA…   65549 FTP Data: 65483 bytes

› Frame 146: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface lo, id 0
› Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
› Transmission Control Protocol, Src Port: 5576, Dst Port: 41755, Seq: 720844, Ack: 1, Len: 53
  FTP Data (53 bytes data)
  [Setup frame: 100]
  [Setup method: PASV]
  [Command: RETR file.txt]
  Command frame: 101
  [Current working directory: /home/roja/Desktop]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 08   ········ ······E·
0010  00 69 43 bc 40 00 40 06  f8 c8 7f 00 00 01 7f 00   ·iC·@·@· ········
0020  00 01 15 c8 a3 1b 03 2c  b5 2a f5 2f c4 a4 80 10   ·······, ·*·/····
0030  02 00 fe 5d 00 00 01 01  08 0a 98 16 37 f5 98 16   ···]···· ····7···
0040  37 f5 00 00 00 00 00 00  00 00 00 00 00 00 00 00   7······· ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00                               ·······

wireshark_loT8K8P2.pcapng    Packets: 63398 · Displayed: 63398 (100.0%) · Dropped: 16720 (26.4%)    Profile: Default
```

Figure 4. Wireshark captures

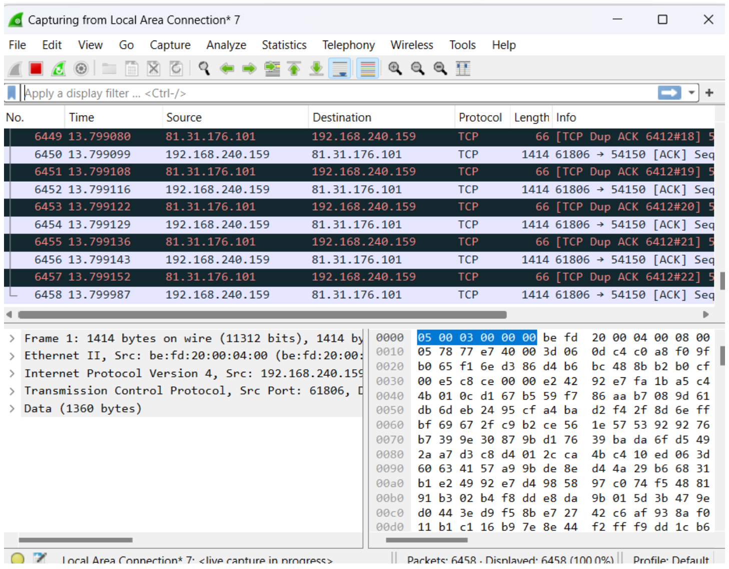
Question 3. Part 3: Setting Up a Local FTP Server on Ubuntu (20 points)

Part A: Install and Configure FTP Server


Capturing from Local Area Connection* 7
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.    Time      Source           Destination      Protocol Length Info
  2447 5.425170  81.31.176.101    192.168.240.159  TCP        54 54150 → 61806 [ACK] Seq
  2448 5.425189  192.168.240.159  81.31.176.101    TCP      1414 [TCP Previous segment n
  2449 5.425202  81.31.176.101    192.168.240.159  TCP        66 [TCP Dup ACK 2447#1] 54
  2450 5.425210  192.168.240.159  81.31.176.101    TCP      1414 [TCP Out-Of-Order] 6180
  2451 5.425223  81.31.176.101    192.168.240.159  TCP        54 54150 → 61806 [ACK] Seq
  2452 5.425231  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq
  2453 5.425243  81.31.176.101    192.168.240.159  TCP        54 54150 → 61806 [ACK] Seq
  2454 5.425250  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq
  2455 5.425258  81.31.176.101    192.168.240.159  TCP        54 54150 → 61806 [ACK] Seq
  2456 5.440001  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq

> Frame 1: 1414 bytes on wire (11312 bits), 1414 by
> Ethernet II, Src: be:fd:20:00:04:00 (be:fd:20:00:
> Internet Protocol Version 4, Src: 192.168.240.159
> Transmission Control Protocol, Src Port: 61806, D
> Data (1360 bytes)

0000  05 00 03 00 00 00 be fd  20 00 04 00 08 00
0010  05 78 77 e7 40 00 3d 06  0d c4 c0 a8 f0 9f
0020  b0 65 f1 6e d3 86 d4 b6  bc 48 8b b2 b0 cf
0030  00 e5 c8 ce 00 00 e2 42  92 e7 fa 1b a5 c4
0040  4b 01 0c d1 67 b5 59 f7  86 aa b7 08 9d 61
0050  db 6d eb 24 95 cf a4 ba  d2 f4 2f 8d 6e ff
0060  bf 69 67 2f c9 b2 ce 56  1e 57 53 92 92 76
0070  b7 39 9e 30 87 9b d1 76  39 ba da 6f d5 49
0080  2a a7 d3 c8 d4 01 2c ca  4b c4 10 ed 06 3d
0090  60 63 41 57 a9 9b de 8e  d4 4a 29 b6 68 31
00a0  b1 e2 49 92 e7 d4 98 58  97 c0 74 f5 48 81
00b0  91 b3 02 b4 f8 dd e8 da  9b 01 5d 3b 47 9e
00c0  d0 44 3e d9 f5 8b e7 27  42 c6 af 93 8a f0
00d0  11 b1 c1 16 b9 7e 8e 44  f2 ff f9 dd 1c b6

Local Area Connection* 7: <live capture in progress> || Packets: 2456 · Displayed: 2456 (100.0%) || Profile: Default
```

Figure 5. Wireshark captures sharif ftp


Capturing from Local Area Connection* 7
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.    Time       Source           Destination      Protocol Length Info
  6449 13.799080  81.31.176.101    192.168.240.159  TCP        66 [TCP Dup ACK 6412#18] 5
  6450 13.799099  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq
  6451 13.799108  81.31.176.101    192.168.240.159  TCP        66 [TCP Dup ACK 6412#19] 5
  6452 13.799116  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq
  6453 13.799122  81.31.176.101    192.168.240.159  TCP        66 [TCP Dup ACK 6412#20] 5
  6454 13.799129  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq
  6455 13.799136  81.31.176.101    192.168.240.159  TCP        66 [TCP Dup ACK 6412#21] 5
  6456 13.799143  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq
  6457 13.799152  81.31.176.101    192.168.240.159  TCP        66 [TCP Dup ACK 6412#22] 5
  6458 13.799987  192.168.240.159  81.31.176.101    TCP      1414 61806 → 54150 [ACK] Seq

> Frame 1: 1414 bytes on wire (11312 bits), 1414 by
> Ethernet II, Src: be:fd:20:00:04:00 (be:fd:20:00:
> Internet Protocol Version 4, Src: 192.168.240.159
> Transmission Control Protocol, Src Port: 61806, D
> Data (1360 bytes)

0000  05 00 03 00 00 00 be fd  20 00 04 00 08 00
0010  05 78 77 e7 40 00 3d 06  0d c4 c0 a8 f0 9f
0020  b0 65 f1 6e d3 86 d4 b6  bc 48 8b b2 b0 cf
0030  00 e5 c8 ce 00 00 e2 42  92 e7 fa 1b a5 c4
0040  4b 01 0c d1 67 b5 59 f7  86 aa b7 08 9d 61
0050  db 6d eb 24 95 cf a4 ba  d2 f4 2f 8d 6e ff
0060  bf 69 67 2f c9 b2 ce 56  1e 57 53 92 92 76
0070  b7 39 9e 30 87 9b d1 76  39 ba da 6f d5 49
0080  2a a7 d3 c8 d4 01 2c ca  4b c4 10 ed 06 3d
0090  60 63 41 57 a9 9b de 8e  d4 4a 29 b6 68 31
00a0  b1 e2 49 92 e7 d4 98 58  97 c0 74 f5 48 81
00b0  91 b3 02 b4 f8 dd e8 da  9b 01 5d 3b 47 9e
00c0  d0 44 3e d9 f5 8b e7 27  42 c6 af 93 8a f0
00d0  11 b1 c1 16 b9 7e 8e 44  f2 ff f9 dd 1c b6

Local Area Connection* 7: <live capture in progress> || Packets: 6458 · Displayed: 6458 (100.0%) || Profile: Default
```

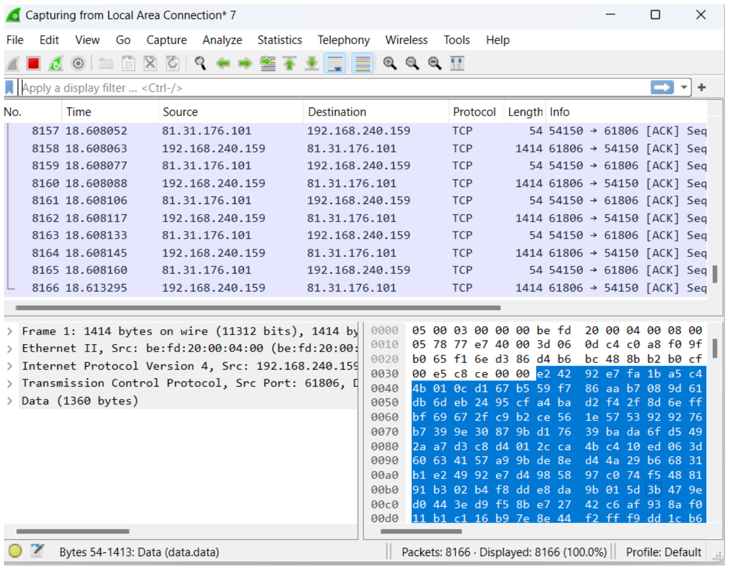
Figure 6. Wireshark captures sharif ftp

Part B: Configure Firewall

Question Why is it important to configure firewall rules for FTP traffic?

Firewalls are essential security tools that act as gatekeepers, controlling incoming and outgoing traffic on your system. Configuring firewall rules for FTP traffic is crucial for several reasons:

Reduced Attack Surface: Without firewall rules, any device on the internet can potentially attempt to connect to your FTP server. This creates a larger attack surface, making your system more vulnerable to malicious attacks. By restricting access through firewall rules, you limit the points of entry for potential threats.
Mitigating Unauthorized Access: Firewalls provide a layer of control over who can access your FTP server. By allowing only specific connections (based on IP address or other criteria), you can prevent unauthorized users from accessing sensitive data or manipulating files.
Improved Security for Exploits: FTP servers, like any software, can have vulnerabilities. Malicious actors might try to exploit these vulnerabilities to gain unauthorized access to your system. Restricting access


Capturing from Local Area Connection* 7
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

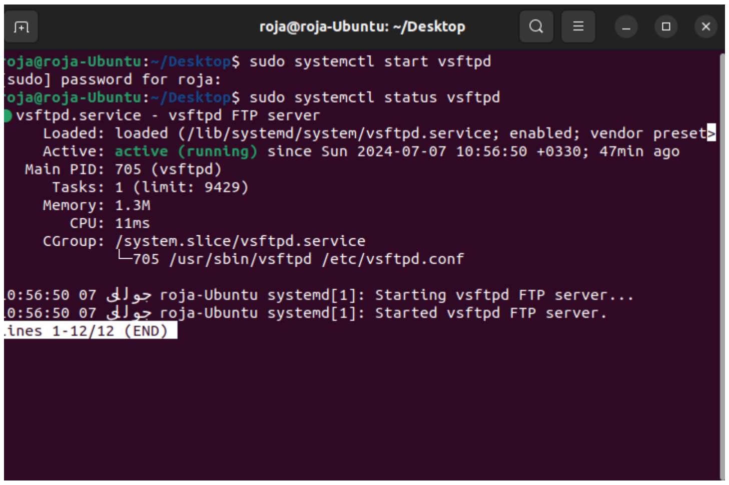
```
No.    Time       Source            Destination       Protocol  Length Info
  8157 18.608052  81.31.176.101     192.168.240.159   TCP         54 54150 → 61806 [ACK] Seq
  8158 18.608063  192.168.240.159   81.31.176.101     TCP       1414 61806 → 54150 [ACK] Seq
  8159 18.608077  81.31.176.101     192.168.240.159   TCP         54 54150 → 61806 [ACK] Seq
  8160 18.608088  192.168.240.159   81.31.176.101     TCP       1414 61806 → 54150 [ACK] Seq
  8161 18.608106  81.31.176.101     192.168.240.159   TCP         54 54150 → 61806 [ACK] Seq
  8162 18.608117  192.168.240.159   81.31.176.101     TCP       1414 61806 → 54150 [ACK] Seq
  8163 18.608133  81.31.176.101     192.168.240.159   TCP         54 54150 → 61806 [ACK] Seq
  8164 18.608145  192.168.240.159   81.31.176.101     TCP       1414 61806 → 54150 [ACK] Seq
  8165 18.608160  81.31.176.101     192.168.240.159   TCP         54 54150 → 61806 [ACK] Seq
  8166 18.613295  192.168.240.159   81.31.176.101     TCP       1414 61806 → 54150 [ACK] Seq
```

```
> Frame 1: 1414 bytes on wire (11312 bits), 1414 by
> Ethernet II, Src: be:fd:20:00:04:00 (be:fd:20:00:
> Internet Protocol Version 4, Src: 192.168.240.15
> Transmission Control Protocol, Src Port: 61806, D
> Data (1360 bytes)
```

```
0000  05 00 03 00 00 00 be fd  20 00 04 00 08 00
0010  05 78 77 e7 40 00 3d 06  0d c4 c0 a8 f0 9f
0020  b0 65 f1 6e d3 86 d4 b6  bc 48 8b b2 b0 cf
0030  00 e5 c8 ce 00 00 e2 42  92 e7 fa 1b a5 c4
0040  4b 01 0c d1 67 b5 59 f7  86 aa b7 08 9d 61
0050  db 6d eb 24 95 cf a4 ba  d2 f4 2f 8d 6e ff
0060  bf 69 67 2f c9 b2 ce 56  1e 57 53 92 92 76
0070  b7 39 9e 30 87 9b d1 76  39 ba da 6f d5 49
0080  2a a7 d3 c8 d4 01 2c ca  4b c4 10 ed 06 3d
0090  60 63 41 57 a9 9b de 8e  d4 4a 29 b6 68 31
00a0  b1 e2 49 92 e7 d4 98 58  97 c0 74 f5 48 81
00b0  91 b3 02 b4 f8 dd e8 da  9b 01 5d 3b 47 9e
00c0  d0 44 3e d9 f5 8b e7 27  42 c6 af 93 8a f0
00d0  11 b1 c1 16 b9 7e 8e 44  f2 ff f9 dd 1c b6
```

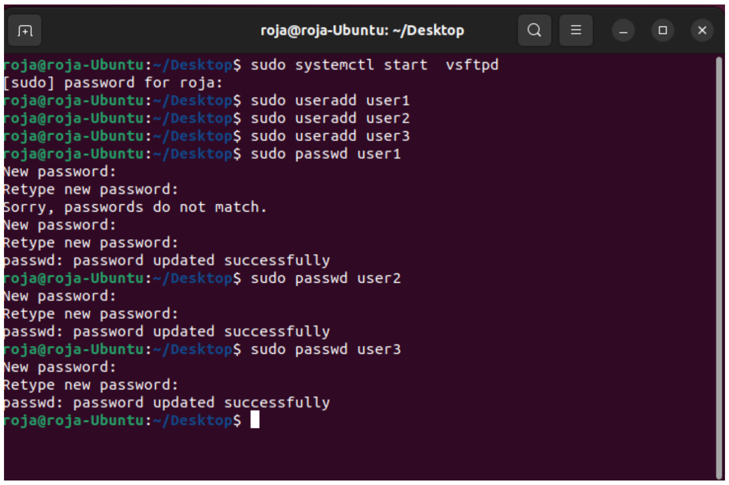
```
Bytes 54-1413: Data (data.data)    Packets: 8166 · Displayed: 8166 (100.0%)    Profile: Default
```

Figure 7. Wireshark captures sharif ftp



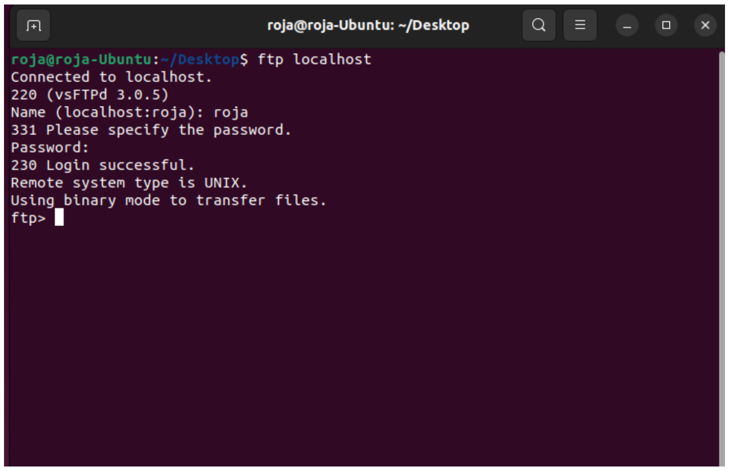
```
oja@roja-Ubuntu:~/Desktop$ sudo systemctl start vsftpd
sudo] password for roja:
oja@roja-Ubuntu:~/Desktop$ sudo systemctl status vsftpd
 vsftpd.service - vsftpd FTP server
     Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset>
     Active: active (running) since Sun 2024-07-07 10:56:50 +0330; 47min ago
   Main PID: 705 (vsftpd)
      Tasks: 1 (limit: 9429)
     Memory: 1.3M
        CPU: 11ms
     CGroup: /system.slice/vsftpd.service
             └─705 /usr/sbin/vsftpd /etc/vsftpd.conf

0:56:50 07 جولای roja-Ubuntu systemd[1]: Starting vsftpd FTP server...
0:56:50 07 جولای roja-Ubuntu systemd[1]: Started vsftpd FTP server.
ines 1-12/12 (END)
```

Figure 8. vsftpd



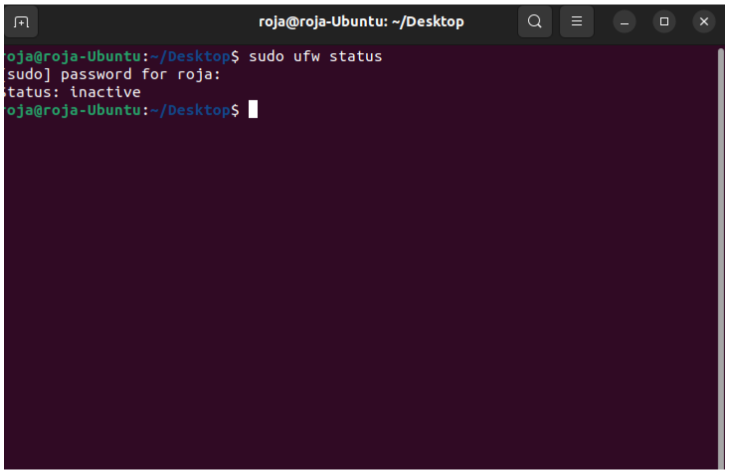
```
roja@roja-Ubuntu:~/Desktop$ sudo systemctl start  vsftpd
[sudo] password for roja:
roja@roja-Ubuntu:~/Desktop$ sudo useradd user1
roja@roja-Ubuntu:~/Desktop$ sudo useradd user2
roja@roja-Ubuntu:~/Desktop$ sudo useradd user3
roja@roja-Ubuntu:~/Desktop$ sudo passwd user1
New password:
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: password updated successfully
roja@roja-Ubuntu:~/Desktop$ sudo passwd user2
New password:
Retype new password:
passwd: password updated successfully
roja@roja-Ubuntu:~/Desktop$ sudo passwd user3
New password:
Retype new password:
passwd: password updated successfully
roja@roja-Ubuntu:~/Desktop$ █
```

Figure 9. add three users



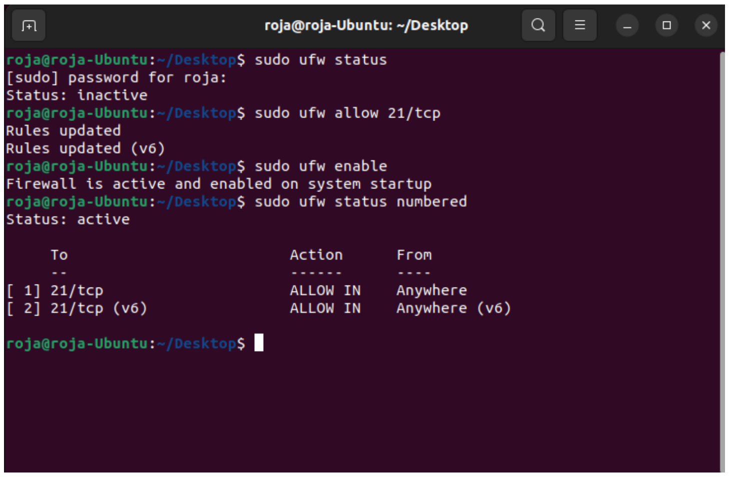
```
roja@roja-Ubuntu:~/Desktop$ ftp localhost
Connected to localhost.
220 (vsFTPd 3.0.5)
Name (localhost:roja): roja
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Figure 10. connect to localhost



```
oja@roja-Ubuntu:~/Desktop$ sudo ufw status
sudo] password for roja:
tatus: inactive
oja@roja-Ubuntu:~/Desktop$
```

Figure 11. firewall status



```
roja@roja-Ubuntu:~/Desktop$ sudo ufw status
[sudo] password for roja:
Status: inactive
roja@roja-Ubuntu:~/Desktop$ sudo ufw allow 21/tcp
Rules updated
Rules updated (v6)
roja@roja-Ubuntu:~/Desktop$ sudo ufw enable
Firewall is active and enabled on system startup
roja@roja-Ubuntu:~/Desktop$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 21/tcp                     ALLOW IN    Anywhere
[ 2] 21/tcp (v6)                ALLOW IN    Anywhere (v6)

roja@roja-Ubuntu:~/Desktop$
```

Figure 12. config firewall to allow FTP traffic

through firewalls makes it harder for attackers to exploit such vulnerabilities.

Question What are the specific ports used for FTP, and how do you open them in the firewall?

The standard port used for FTP traffic is 21 (TCP). This is where the communication between the FTP client and server takes place. To allow incoming connections for FTP traffic, you need to create a firewall rule that specifies this port and protocol.
Here's how you open the FTP port in the firewall using UFW on Ubuntu:

Recommended Approach - Allow Specific Port:

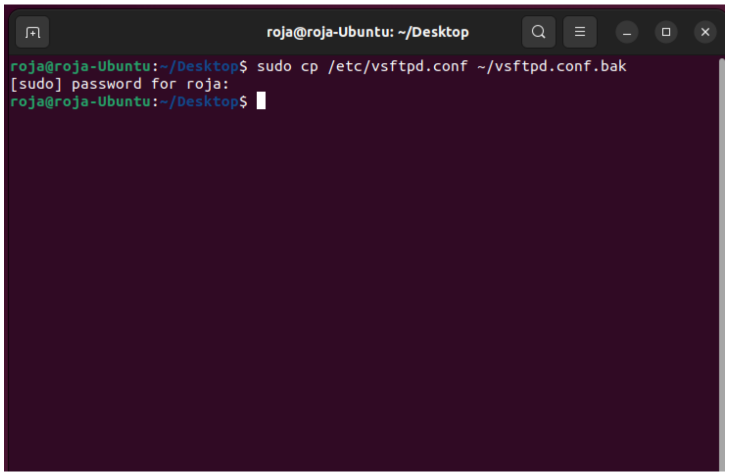
sudo ufw allow 21/tcp

This allows only authorized connections on port 21 specifically for the TCP protocol used by FTP. This is the most secure approach as it minimizes unnecessary access.
Not Recommended Approach - Allow All Traffic:

sudo ufw allow OpenSSH

While it's technically possible to open all ports (using allow OpenSSH), it's highly discouraged. This approach removes any firewall restrictions, exposing your system to all types of traffic on all ports.

Part C: Change Default Directory



```
roja@roja-Ubuntu:~/Desktop$ sudo cp /etc/vsftpd.conf ~/vsftpd.conf.bak
[sudo] password for roja:
roja@roja-Ubuntu:~/Desktop$
```

Figure 13. back up vsftpd configuration files

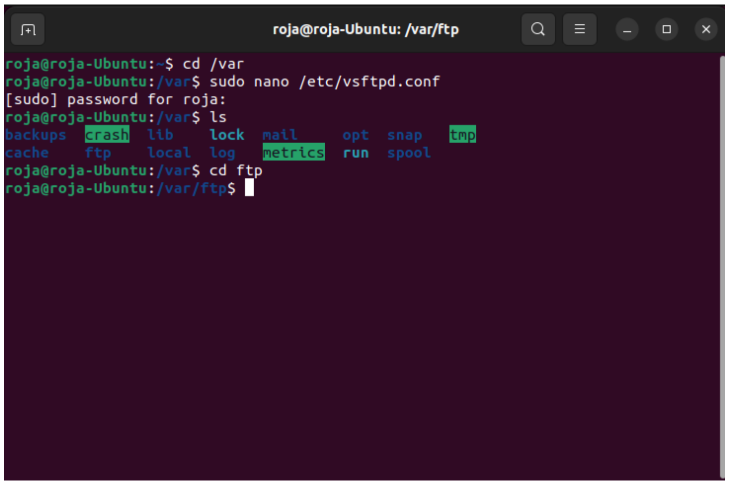
Question Why is it important to back up configuration files before editing them?

Safety Net: In case something goes wrong during editing, you can restore the original configuration from the backup. This ensures you can revert to a working state if needed.

Version Control: If you make multiple configuration changes, having backups allows you to track changes and revert to specific versions if necessary.

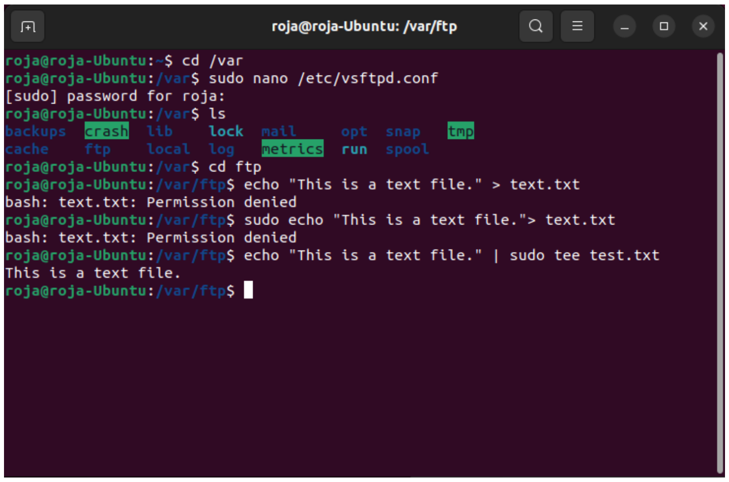
Question Why is it beneficial to change the default directory for FTP users?

Improved Security: The default directory for FTP users is often their home directory. This might contain sensitive files you don't want users to access. By changing the default directory, you can restrict access to a specific location intended for FTP purposes. Organization: It helps segregate files used for FTP from other user files, making management and access control easier.



```
roja@roja-Ubuntu: /var/ftp
roja@roja-Ubuntu:~$ cd /var
roja@roja-Ubuntu:/var$ sudo nano /etc/vsftpd.conf
[sudo] password for roja:
roja@roja-Ubuntu:/var$ ls
backups  crash  lib    lock  mail     opt  snap   tmp
cache    ftp    local  log   metrics  run  spool
roja@roja-Ubuntu:/var$ cd ftp
roja@roja-Ubuntu:/var/ftp$ 
```

FIGURE 14. Modify the vsftpd configuration



```
roja@roja-Ubuntu: /var/ftp
roja@roja-Ubuntu:~$ cd /var
roja@roja-Ubuntu:/var$ sudo nano /etc/vsftpd.conf
[sudo] password for roja:
roja@roja-Ubuntu:/var$ ls
backups  crash  lib    lock  mail     opt  snap   tmp
cache    ftp    local  log   metrics  run  spool
roja@roja-Ubuntu:/var$ cd ftp
roja@roja-Ubuntu:/var/ftp$ echo "This is a text file." > text.txt
bash: text.txt: Permission denied
roja@roja-Ubuntu:/var/ftp$ sudo echo "This is a text file."> text.txt
bash: text.txt: Permission denied
roja@roja-Ubuntu:/var/ftp$ echo "This is a text file." | sudo tee test.txt
This is a text file.
roja@roja-Ubuntu:/var/ftp$ 
```

FIGURE 15. Create a new file in the new directory

Part D: Securing FTP

Part 1

Question Why is it important to encrypt FTP traffic?

It's important to encrypt FTP traffic to ensure the security and privacy of data being transmitted over the network. Without encryption, any data, including usernames, passwords, and actual file contents, can be intercepted and viewed by unauthorized parties.

Question What are the benefits of using SSL/TLS encryption for FTP?

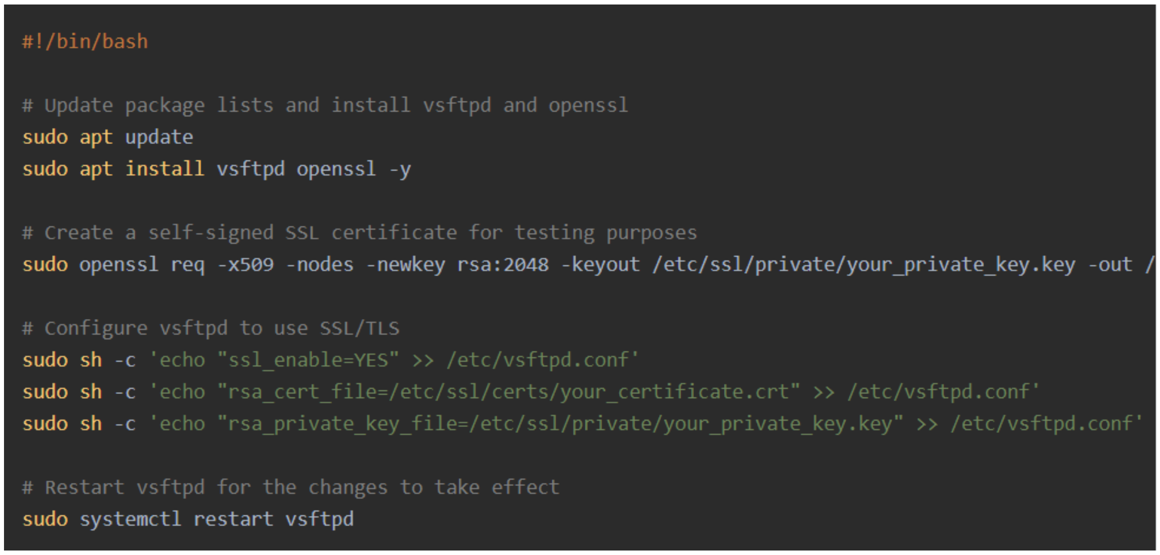
Benefits of using SSL/TLS encryption for FTP include:
Data Confidentiality: Encryption ensures that the data being transmitted is secure and cannot be read by unauthorized parties.
Data Integrity: SSL/TLS helps in ensuring that the data being transmitted is not modified by unauthorized parties.

Authentication: SSL/TLS provides a way for the client and server to authenticate each other, ensuring that the client is connecting to the intended server.

Configure the FTP server to use SSL/TLS encryption to secure the connection
Step 1: Install vsftpd (if not already installed) and OpenSSL:

Step 2: Create an SSL certificate for the FTP server. You can use a self-signed certificate for testing, or obtain a certificate from a trusted certificate authority (CA).

Step 3: Configure vsftpd to use SSL/TLS.
Step 4: Restart vsftpd for the changes to take effect:
QuestionWhy do we need to secure FTP servers?



```
#!/bin/bash

# Update package lists and install vsftpd and openssl
sudo apt update
sudo apt install vsftpd openssl -y

# Create a self-signed SSL certificate for testing purposes
sudo openssl req -x509 -nodes -newkey rsa:2048 -keyout /etc/ssl/private/your_private_key.key -out /

# Configure vsftpd to use SSL/TLS
sudo sh -c 'echo "ssl_enable=YES" >> /etc/vsftpd.conf'
sudo sh -c 'echo "rsa_cert_file=/etc/ssl/certs/your_certificate.crt" >> /etc/vsftpd.conf'
sudo sh -c 'echo "rsa_private_key_file=/etc/ssl/private/your_private_key.key" >> /etc/vsftpd.conf'

# Restart vsftpd for the changes to take effect
sudo systemctl restart vsftpd
```

Figure 16. Configure the FTP server

Securing FTP servers is crucial for several reasons:

Data Security: FTP servers often handle sensitive data, and securing them helps prevent unauthorized access and data breaches.

Compliance: Many industries and jurisdictions have regulations regarding data security and privacy. Securing FTP servers helps organizations comply with these regulations.

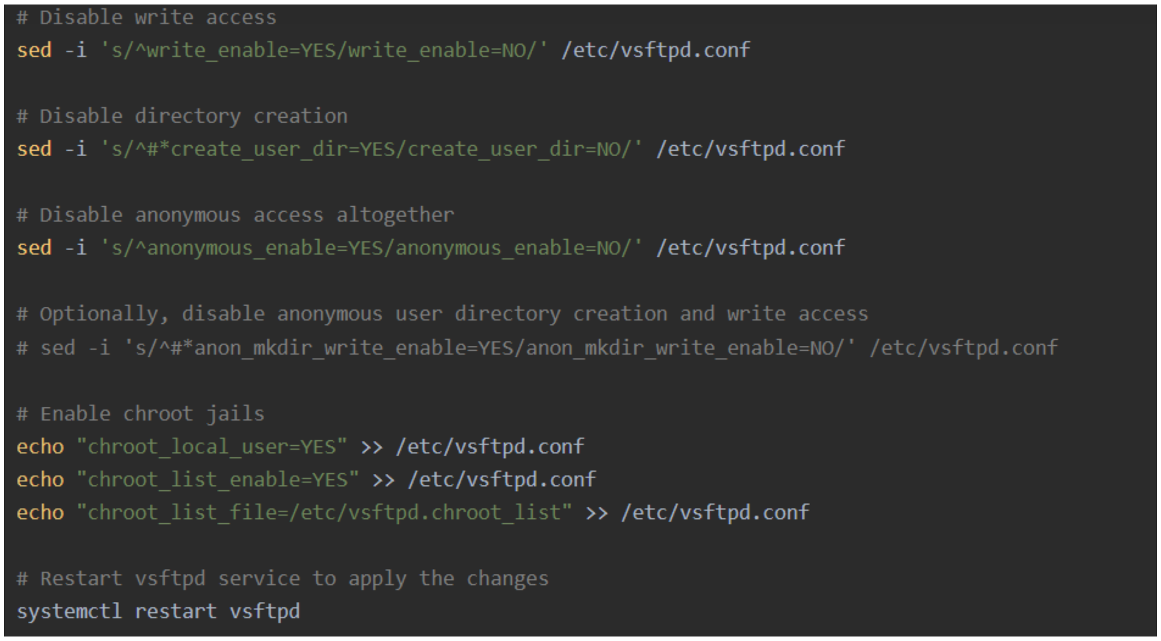
Protection Against Malware: Unsecured FTP servers can be exploited by malware and hackers to distribute malicious content. Securing the server helps prevent such misuse.

Part 2
vsftpd Configuration:

The vsftpd configuration file (/etc/vsftpd.conf) offers various options for limiting user access:

Write Access: Set write enable = NO to prevent users from uploading files.
Directory Creation: Set create user dir=NO to stop users from creating new directories within their FTP space.
Anonymous Access: Disable anonymous access altogether or use anon mkdir write enable=NO to restrict directory creation and write access for anonymous users.

Chroot Jails: Enable chroot local user with the new directory path to confine users to their designated directory, preventing access to other parts of the system.



```
# Disable write access
sed -i 's/^write_enable=YES/write_enable=NO/' /etc/vsftpd.conf

# Disable directory creation
sed -i 's/^#*create_user_dir=YES/create_user_dir=NO/' /etc/vsftpd.conf

# Disable anonymous access altogether
sed -i 's/^anonymous_enable=YES/anonymous_enable=NO/' /etc/vsftpd.conf

# Optionally, disable anonymous user directory creation and write access
# sed -i 's/^#*anon_mkdir_write_enable=YES/anon_mkdir_write_enable=NO/' /etc/vsftpd.conf

# Enable chroot jails
echo "chroot_local_user=YES" >> /etc/vsftpd.conf
echo "chroot_list_enable=YES" >> /etc/vsftpd.conf
echo "chroot_list_file=/etc/vsftpd.chroot_list" >> /etc/vsftpd.conf

# Restart vsftpd service to apply the changes
systemctl restart vsftpd
```

FIGURE 17. Configure the FTP server to be secure

Question What are the methods to limit user access on an FTP server?

There are several methods to limit user access on an FTP server:

User Isolation: Restrict each user to their home directory. This prevents users from navigating outside of their designated directory structure.

User Permissions: Leverage filesystem permissions to control which directories users can access and what operations they can perform within those directories. This can be achieved through the use of tools like chown, chmod, and chgrp.

Chroot Jail: Implement a chroot environment, which creates a virtualized environment that limits a user to a specific directory and its subdirectories. This prevents users from accessing other parts of the file system.

FTP User Groups: Utilize user groups to categorize users and assign permissions and access rights based on their group membership. This allows for granular control over user access based on their roles and responsibilities.

By implementing these measures, organizations can enforce the principle of least privilege, ensuring that users only have access to the resources necessary to perform their specific tasks, thus enhancing overall security.

Question 4. Bandwidth and Transfer Rate Control (15 points Bonus)
Question Why is bandwidth control important in network applications?
Bandwidth control is important in network applications for several reasons:

Resource Management: Bandwidth control allows network administrators to allocate network resources fairly among users or applications, ensuring no single user or application consumes excessive bandwidth at the expense of others.

Preventing Congestion: By controlling the rate at which data is transmitted, bandwidth control helps prevent network congestion, which can lead to degraded performance and delays for all users.

Quality of Service (QoS): Bandwidth control is essential for implementing Quality of Service policies, ensuring that critical applications, such as real-time video or voice communications, receive the necessary bandwidth to operate smoothly.

Cost Control: For organizations with limited bandwidth or those operating in regions with data usage caps, bandwidth control helps manage costs by preventing excessive data usage.

Security: Bandwidth control can help prevent certain types of attacks, such as Distributed Denial of Service (DDoS) attacks, by limiting the rate of incoming traffic.

Implementation

```python
    def send_data(self, file_name):
        # Open the file for reading in binary mode
        file_size = os.path.getsize(file_name)
        with open(file_name, 'rb') as file:
            start_time = time.time()
            bytes_sent = 0

            while True:
                # Read data in chunks
                data = file.read(CHUNK_SIZE)  # Adjust chunk size as needed

                # Calculate transfer rate
                elapsed_time = time.time() - start_time
                transfer_rate = bytes_sent / elapsed_time if elapsed_time else 0

                # Introduce delay if transfer rate exceeds limit
                if transfer_rate > MAX_BANDWIDTH:
                    delay = (bytes_sent / MAX_BANDWIDTH) - elapsed_time
                    if delay > 0:
                        time.sleep(delay)

                # Send data (implement error handling for robustness)
                self.dataConn.sendall(data)

                # Update progress and bytes sent
                bytes_sent += len(data)
                progress = (bytes_sent / file_size) * 100
                print(f"Progress: {progress:.2f}%, Transfer Rate: {transfer_rate:.2f}
KB/s")

                # Check for end of file
                if not data:
                    break
            self.dataConn.sendall(b'\x00' * CHUNK_SIZE)

            print("File transfer complete.")
```

Listing 1. Updated send data function

```
    def send_data(self, file_name):
        # Open the file for reading in binary mode
        file_size = os.path.getsize(file_name)
        with open(file_name, 'rb') as file:
            start_time = time.time()
            bytes_sent = 0

            while True:
                # Read data in chunks
                data = file.read(CHUNK_SIZE)  # Adjust chunk size as needed

                # Calculate transfer rate
                elapsed_time = time.time() - start_time
                transfer_rate = bytes_sent / elapsed_time if elapsed_time else 0

                # Introduce delay if transfer rate exceeds limit
                if transfer_rate > MAX_BANDWIDTH:
                    delay = (bytes_sent / MAX_BANDWIDTH) - elapsed_time
                    if delay > 0:
                        time.sleep(delay)

                # Send data (implement error handling for robustness)
                self.dataConn.sendall(data)

                # Update progress and bytes sent
                bytes_sent += len(data)
                progress = (bytes_sent / file_size) * 100
                print(f"Progress: {progress:.2f}%, Transfer Rate: {transfer_rate:.2f}
   KB/s")

                # Check for end of file
                if not data:
                    break
            self.dataConn.sendall(b'\x00' * CHUNK_SIZE)

            print("File transfer complete.")
```

Listing 2. Updated send data function

```
    def get_data(self, file_name):
        with open(file_name, 'wb') as file:
            while True:
                # Receive data in chunks
                data = self.clientDataSock.recv(CHUNK_SIZE)
                if data == b'\x00' * CHUNK_SIZE:
                    break
                # Write received data to the file
                if data:
                    file.write(data)
```

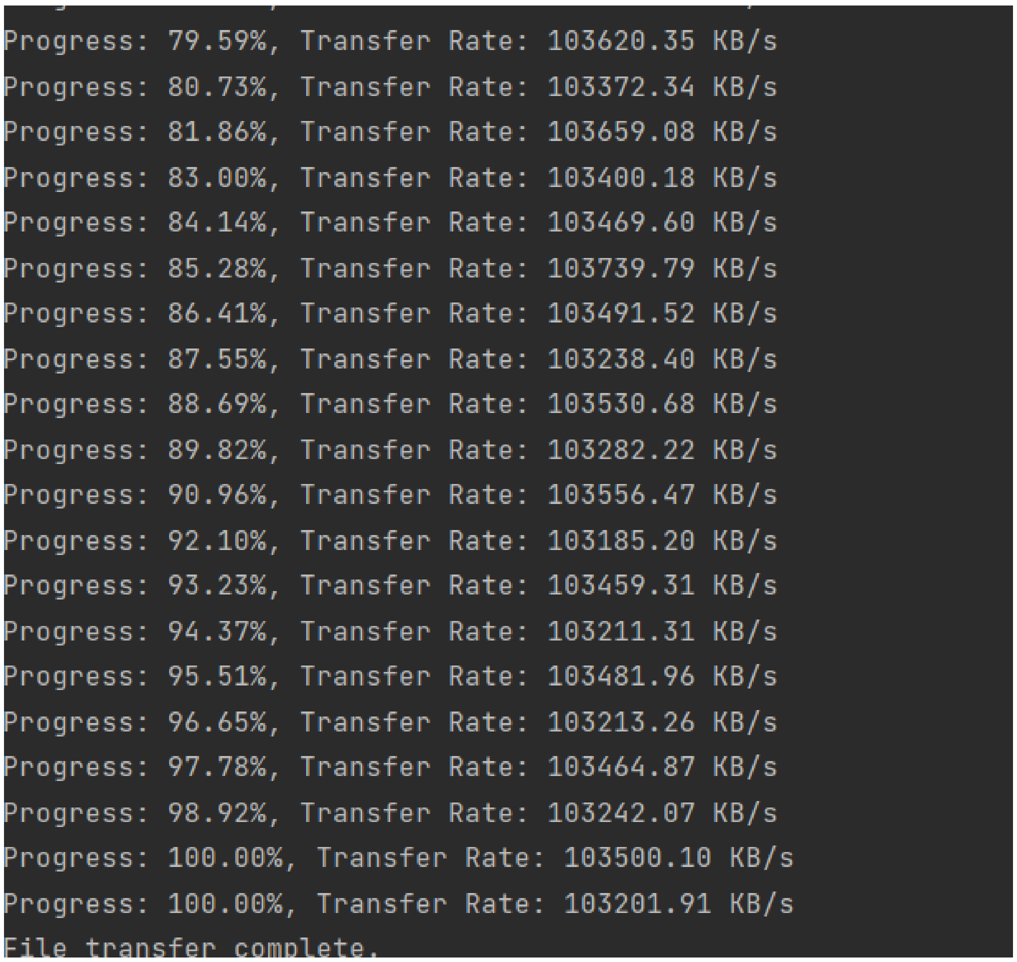
Listing 3. Updated get data function

Bandwidth Control:

The implemented approach assumes server-side throttling. The server controls the data transfer rate by introducing delays if it exceeds a predefined limit. The client doesn't need to be explicitly aware of this throttling mechanism.

End-of-Data Indication:
A custom end-of-data mechanism is implemented using the following steps:
Server-Side:
The server calculates the number of data chunks required to send the entire file based on a chosen chunk size.
The actual file data is then sent in chunks using a loop.
After sending all data chunks, a stream of zero bytes with the same size as the chunk size is sent to signal the end of data.
Client-Side:

The client receives the number of data chunks (end-of-data indicator) from the server.
It enters a loop, receiving data chunks the same number of times as the received indicator and writing them to the file.
After receiving all data chunks, the client receives another chunk of data. This should be a stream of zero bytes with the size of the chunk.



```
Progress: 79.59%, Transfer Rate: 103620.35 KB/s
Progress: 80.73%, Transfer Rate: 103372.34 KB/s
Progress: 81.86%, Transfer Rate: 103659.08 KB/s
Progress: 83.00%, Transfer Rate: 103400.18 KB/s
Progress: 84.14%, Transfer Rate: 103469.60 KB/s
Progress: 85.28%, Transfer Rate: 103739.79 KB/s
Progress: 86.41%, Transfer Rate: 103491.52 KB/s
Progress: 87.55%, Transfer Rate: 103238.40 KB/s
Progress: 88.69%, Transfer Rate: 103530.68 KB/s
Progress: 89.82%, Transfer Rate: 103282.22 KB/s
Progress: 90.96%, Transfer Rate: 103556.47 KB/s
Progress: 92.10%, Transfer Rate: 103185.20 KB/s
Progress: 93.23%, Transfer Rate: 103459.31 KB/s
Progress: 94.37%, Transfer Rate: 103211.31 KB/s
Progress: 95.51%, Transfer Rate: 103481.96 KB/s
Progress: 96.65%, Transfer Rate: 103213.26 KB/s
Progress: 97.78%, Transfer Rate: 103464.87 KB/s
Progress: 98.92%, Transfer Rate: 103242.07 KB/s
Progress: 100.00%, Transfer Rate: 103500.10 KB/s
Progress: 100.00%, Transfer Rate: 103201.91 KB/s
File transfer complete.
```

FIGURE 18. Output of get command