



Sharif University of Technology
Electrical Engineering Complex

Data Networks Project

Melika Rajabi - 99101608

July 7, 2024

Part 2: FTP Protocol Understanding

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

There are lots of protocols used for transferring files. Some of them are FTPS, SFTP, SCP, HTTP, HTTPS, WebDAV, WebDAVS, TFTP, AS2, OFTP, and AFTP. Here I'm going to explain some of them which I think are more important.

- **FTPS:** When the FTP protocol was initially drafted, security was not a concern. Since then, many things have changed, and sending data over any public network without encryption is considered very risky and, in some cases, prohibited. In order to address this issue, a set of security extensions to create a more secure FTP protocol was proposed to protect both the control and data channel connections either concurrently or independently using secure sockets layer (SSL) and transport layer security (TLS). This encryption can be done manually, but is time-consuming and error-prone, so organizations use FTPS servers/FTPS clients to provide data security. The following table represents a comparison between FTPS and FTP:

FTPS	FTP
Encryptions happens on both command and data channels via either implicit SSL or explicit SSL	Unencrypted information exchange in both command and data channels
Communication is human-readable	Communication is human-readable
Firewall port for server allow inbound connections on port 21 and/or 990, 989	Firewall port for server allow inbound connections on port 21
Firewall port for client allow outbound connections to port 21 and passive port range defined by server	Firewall port for client allow outbound connections to port 21 and passive port range defined by server

- **SFTP**: SSH file transfer protocol (SFTP) is based on the SSH (Secure Shell) protocol, which is best known for providing secure access to shell accounts on remote servers.

Unlike FTP, SFTP does not use separate command and data connections. Both data and commands are transferred in specially formatted packets via a single connection. All data sent between the client and server is encrypted using an agreed-upon encryption cipher. SFTP sessions can also be further protected through the use of public and private keys, which offer an alternative form of authentication known as public key authentication. This can be used as an alternative to or in conjunction with the traditional form of authentication of usernames and passwords. It's also more firewall-friendly. The following table represents a comparison between SFTP and FTP:

SFTP	FTP
All information exchange between the FTP server and client are encrypted via SSH protocol. SFTP can also encrypts the session	Unencrypted information exchange in both command and data channels
Communication is not human-readable as it's in a binary format	Communication is human readable
Firewall port for server allow inbound connections on port 22	Firewall port for server allow inbound connections on port 21
Firewall port for client allow outbound connections to port 22	Firewall port for client allow outbound connections to port 21 and passive port range defined by server

- **SCP**: Leveraging the SSH protocol, SCP (Secure Copy Protocol) transfers files via encrypted IP-based data tunnels, ensuring both speed and security. It does this by moving files between local hosts and remote hosts (or two remote hosts).

It lacks file management capabilities. It offers little support for resume file transfers. It's built for file transfers only. Thus you can't create directories or directory listings or delete files. Depending on the specific data type, it's far more limited in scope. FTP can be faster than SCP when dealing with large files or multiple files because FTP uses separate channels for commands and data, which reduces the amount of overhead per file. However, the difference in speed may not be noticeable unless you're

transferring very large files or a considerable number of files. FTP, especially in its active mode, can be problematic with firewalls, as it requires opening several ports, leading to potential security vulnerabilities. SCP, due to its use of SSH, needs just one port (typically port 22), making it easier to manage with firewalls.

- **HTTP**: HTTP stands for HyperText Transfer Protocol. It is an internet standard that allows the process of transfer of web pages over the internet. It also defines how the web browser will respond to any web request. HTTP is the backbone of WWW. It uses Transmission Control Protocol (TCP). The following table represents a comparison between HTTP and FTP:

HTTP	FTP
Faster	Slower
Efficient in transferring small files	Efficient in transferring large files
Uses one way communication system	Uses two way communication system
Only supports the data connection	Supports both data connection and control connection
Can use both types of Persistent and Non-persistent TCP connection	Uses a Persistent TCP connection for the Control connection and a Non-persistent TCP Connection for Data Connection
The files transferred to the computer over the internet are not saved to the memory	The files transferred to the computer over the internet are saved to the memory
Doesn't require authentication	Requires authentication
Less prone to firewall issue	Prone to firewall issues
Stateless	Maintains states

HTTP	FTP
Used to provide the web pages to the web browser from the webserver	Used to upload or download files between client and server
The set of rules that how web pages are transferred on different computers over the internet	The set of rules that permit the downloading and uploading the files on the computer over the internet
Uses Transmission Control Protocol and runs on TCP port 80	Uses Transmission Control Protocol and runs on TCP port 20 and TCP port 21
The URL starts with HTTP	The URL starts with FTP

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

FTP which is based on TCP is among the most commonly used protocols for transferring files.

Both TCP and UDP are applicable in the transport layer. No matter whether you use TCP or UDP, data is broken into packets when being sent to the receiving computer.

TCP (Transmission Control Protocol) transfers data in a carefully controlled sequence of packets. As each packet is received at the destination, an acknowledgment is sent back to the sender. If the sender does not receive the acknowledgment in a certain period of time, it simply sends the packet again. To protect the sequence, further packets cannot be sent until the missing package has been successfully transmitted and an acknowledgment received.

With poor quality networks, it is possible for the original packet to be received intact but the acknowledgment packet to be either corrupted or lost in transmission. This would cause the whole packet to be resent unnecessarily, having an impact on transmission times. In addition, not all bandwidth is used up early in the cycle, and the rate of transmission is dictated by how much data gets through until failures start to occur.

When UDP (User Datagram Protocol) is used, the packets are sent "blind".

The transfer continues regardless of whether data is being successfully received or not. This potential loss may result in a corrupted file – in the case of a streamed video this could be some missing frames or out of sync audio, but generally it will require a file to be resent in its entirety. The lack of guarantee makes the transfer fast, but unless combined with rigorous error checking, it is often unsuitable for data transfers.

The best results for fast file transfer are gained from a combination of both UDP and TCP. UDP is used to transfer the data, whilst TCP is used to ensure that any packets lost are resent. An agent at the receiving station is used to reconstruct the data after the transfer.

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

One notable disadvantage is that FTP operates over unencrypted channels by default, which poses security risks, particularly when transmitting sensitive or confidential information. While secure FTP variants address this concern to some extent, configuring and maintaining secure FTP implementations can be complex and require additional resources and expertise.

Additionally, it lacks built-in support for file synchronization and version control. Unlike more advanced file transfer protocols or cloud-based solutions, FTP does not inherently provide mechanisms for managing file revisions, tracking changes, or synchronizing files across multiple devices or locations.

Moreover, FTP's reliance on a client-server architecture may present scalability challenges for organizations with large-scale file transfer requirements or distributed networks. Managing multiple FTP servers and clients across different locations can become cumbersome and inefficient, necessitating careful planning and resource allocation.

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

In active mode, the client connects from a random unprivileged port ($N > 1024$) to the FTP server's command port, port 21. Then, the client

starts listening to port $N + 1$ and sends the command "PORT $N + 1$ " from its command port to the FTP server's command port. The server then sends an ACK back to the client's command port. The server will then connect back to the client's specified data port from its local data port, which is port 20. Finally, the client sends an ACK back.

The main problem with active mode actually falls on the client side. The FTP client doesn't make the actual connection to the data port of the server. It simply tells the server what port it is listening on and the server connects back to the specified port on the client. From the client side firewall this appears to be an outside system initiating a connection to an internal client. Something that is usually blocked.

In order to resolve the issue of the server initiating the connection to the client a different method for FTP connections was developed. This was known as passive mode.

In passive mode FTP the client initiates both connections to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. When opening an FTP connection, the client opens two random unprivileged ports locally ($N > 1024$ and $N + 1$). The first port contacts the server on port 21, then the client will issue the "PASV" command. The result of this is that the server then opens a random unprivileged port ($P > 1024$) and sends the PORT P command back to the client. The client then initiates the connection from port $N + 1$ to port P on the server to transfer data. Finally, the server sends back an ACK.

Also it's good to mention that FTP clients are often using NAT behind firewalls; and if a movable device, such as a laptop, will have an ever-changing external IP address. This means that the IP address sent to the server, using the "PORT" command in active mode will have to be re-configured each time the external IP address changes. However, in passive mode the configuration responsibility is placed on the server-side, which is less likely to be changing or mobile.

**2.5 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.**

There are different ways an FTP server and client software can conduct a file transfer using FTP:

- Anonymous FTP: This is the most basic form of FTP. It provides support for data transfers without encrypting data or using a username and password. It's most commonly used for download of material that is allowed for unrestricted distribution. It works on port 21.
- Password-protected FTP: This is also a basic FTP service, but it requires the use of a username and password, though the service might not be encrypted or secure. It also works on port 21.

By default, FTP does not encrypt traffic, and individuals can capture packets to read usernames, passwords and other data. Therefore other enhanced versions of FTP such as FTPS and SFTP are available. I have explained these protocols in the previous sections but here I am going to talk about them from the point of security.

- **SFTP**: SFTP combines the secure authentication and encryption features of SSH with file transfer functionality. It allows users to securely upload, download, and manage files on remote servers using an encrypted connection.

It uses encryption to protect data during transit. If that data is stolen or listened to during a transfer, the hacker would still need to break the encryption to read it. Because of this, SFTP is suitable for applications that call for data security.

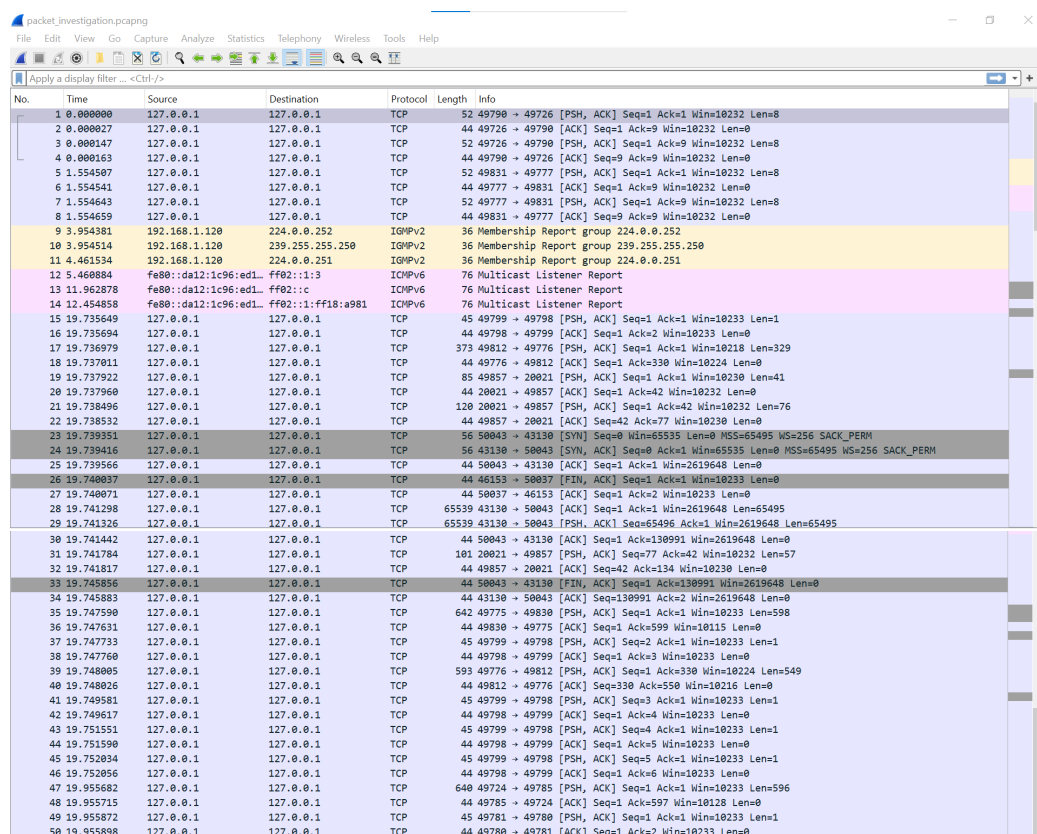
It supports a process of tunneling, where data is transferred through an encrypted "tunnel". Unlike direct transfers, it can protect all data transmitted, including authentication credentials. SSH tunneling also allows users to use multiple applications within an encrypted tunnel.

- **FTPS**: The security problem was addressed by combining the plaintext, unencrypted FTP with Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols. A resulting combination would then carry the file transfer functionality inside an encrypted SSL/TLS tunnel that provides end-to-end encryption and transport security.

- **FTP over SSH:** This enables the secure transmission of files across the web. It works by first establishing an SSH tunnel between the client and the server. Once an SSH tunnel has been established, it is used by the File Transfer Protocol (FTP) to send data back and forth. In this scenario, the FTP handles the transfer of information while the SSH tunnel ensures that the data is transmitted in a secure fashion. Nowadays, FTP over SSH is not seeing widespread use as it is difficult to set up. The tunnel is set up between the FTP client and the FTP server.

2.6 Wireshark

Using Wireshark I have observed the network when using "get" command on a 1GB arbitrary file. The captured packets are as follows. Data is transferred from packet number 15 until packet number 50:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	52	49790 → 49726 [PSH, ACK] Seq=1 Ack=1 Win=10232 Len=8
2	0.000027	127.0.0.1	127.0.0.1	TCP	44	49726 → 49790 [ACK] Seq=1 Ack=9 Win=10232 Len=0
3	0.000147	127.0.0.1	127.0.0.1	TCP	52	49726 → 49790 [PSH, ACK] Seq=1 Ack=9 Win=10232 Len=8
4	0.000163	127.0.0.1	127.0.0.1	TCP	44	49790 → 49726 [ACK] Seq=9 Ack=9 Win=10232 Len=0
5	1.554507	127.0.0.1	127.0.0.1	TCP	52	49831 → 49777 [PSH, ACK] Seq=1 Ack=1 Win=10232 Len=8
6	1.554541	127.0.0.1	127.0.0.1	TCP	44	49777 → 49831 [ACK] Seq=1 Ack=9 Win=10232 Len=0
7	1.554643	127.0.0.1	127.0.0.1	TCP	52	49777 → 49831 [PSH, ACK] Seq=1 Ack=9 Win=10232 Len=8
8	1.554659	127.0.0.1	127.0.0.1	TCP	44	49831 → 49777 [ACK] Seq=9 Ack=9 Win=10232 Len=0
9	3.954381	192.168.1.120	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
10	3.954514	192.168.1.120	239.255.255.250	IGMPv2	36	Membership Report group 239.255.255.250
11	4.461534	192.168.1.120	224.0.0.251	IGMPv2	36	Membership Report group 224.0.0.251
12	5.460884	fe80::da12:1c96:ed1::ff02::1:3	ff02::1:3	ICMPv6	76	Multicast Listener Report
13	11.962878	fe80::da12:1c96:ed1::ff02::c	ff02::c	ICMPv6	76	Multicast Listener Report
14	12.454858	fe80::da12:1c96:ed1::ff02::1:ff18:a981	ff02::1:ff18:a981	ICMPv6	76	Multicast Listener Report
15	19.735649	127.0.0.1	127.0.0.1	TCP	45	49799 → 49798 [PSH, ACK] Seq=1 Ack=1 Win=10233 Len=1
16	19.735694	127.0.0.1	127.0.0.1	TCP	44	49798 → 49799 [ACK] Seq=1 Ack=2 Win=10233 Len=0
17	19.736079	127.0.0.1	127.0.0.1	TCP	373	49812 → 49776 [PSH, ACK] Seq=1 Ack=1 Win=10218 Len=329
18	19.737811	127.0.0.1	127.0.0.1	TCP	44	49776 → 49812 [ACK] Seq=1 Ack=330 Win=10224 Len=0
19	19.737922	127.0.0.1	127.0.0.1	TCP	85	49857 → 20021 [PSH, ACK] Seq=1 Ack=1 Win=10230 Len=41
20	19.737960	127.0.0.1	127.0.0.1	TCP	44	20021 → 49857 [ACK] Seq=1 Ack=42 Win=10232 Len=0
21	19.738496	127.0.0.1	127.0.0.1	TCP	120	20021 → 49857 [PSH, ACK] Seq=1 Ack=42 Win=10232 Len=76
22	19.738532	127.0.0.1	127.0.0.1	TCP	44	49857 → 20021 [ACK] Seq=42 Ack=77 Win=10230 Len=0
23	19.739351	127.0.0.1	127.0.0.1	TCP	56	50043 → 43130 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
24	19.739416	127.0.0.1	127.0.0.1	TCP	56	43130 → 50043 [SYN, ACK] Seq=1 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
25	19.739566	127.0.0.1	127.0.0.1	TCP	44	50043 → 43130 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
26	19.740037	127.0.0.1	127.0.0.1	TCP	44	46153 → 50037 [FIN, ACK] Seq=1 Ack=1 Win=10233 Len=0
27	19.740071	127.0.0.1	127.0.0.1	TCP	44	50037 → 46153 [ACK] Seq=1 Ack=2 Win=10233 Len=0
28	19.741298	127.0.0.1	127.0.0.1	TCP	65539	43130 → 50043 [ACK] Seq=1 Ack=1 Win=2619648 Len=65495
29	19.741326	127.0.0.1	127.0.0.1	TCP	65539	43130 → 50043 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=65495
30	19.741442	127.0.0.1	127.0.0.1	TCP	44	50043 → 43130 [ACK] Seq=1 Ack=130991 Win=2619648 Len=0
31	19.741784	127.0.0.1	127.0.0.1	TCP	181	20021 → 49857 [PSH, ACK] Seq=77 Ack=42 Win=10232 Len=57
32	19.741817	127.0.0.1	127.0.0.1	TCP	44	49857 → 20021 [ACK] Seq=42 Ack=134 Win=10230 Len=0
33	19.745856	127.0.0.1	127.0.0.1	TCP	44	50043 → 43130 [FIN, ACK] Seq=1 Ack=130991 Win=2619648 Len=0
34	19.745883	127.0.0.1	127.0.0.1	TCP	44	43130 → 50043 [ACK] Seq=130991 Ack=2 Win=2619648 Len=0
35	19.747590	127.0.0.1	127.0.0.1	TCP	642	49775 → 49830 [PSH, ACK] Seq=1 Ack=1 Win=10233 Len=598
36	19.747631	127.0.0.1	127.0.0.1	TCP	44	49830 → 49775 [ACK] Seq=1 Ack=599 Win=10115 Len=0
37	19.747733	127.0.0.1	127.0.0.1	TCP	45	49799 → 49798 [PSH, ACK] Seq=2 Ack=1 Win=10233 Len=1
38	19.747760	127.0.0.1	127.0.0.1	TCP	44	49798 → 49799 [ACK] Seq=1 Ack=3 Win=10233 Len=0
39	19.748005	127.0.0.1	127.0.0.1	TCP	593	49776 → 49812 [PSH, ACK] Seq=1 Ack=330 Win=10224 Len=549
40	19.748026	127.0.0.1	127.0.0.1	TCP	44	49812 → 49776 [ACK] Seq=330 Ack=550 Win=10216 Len=0
41	19.748581	127.0.0.1	127.0.0.1	TCP	45	49799 → 49798 [PSH, ACK] Seq=3 Ack=1 Win=10233 Len=1
42	19.749617	127.0.0.1	127.0.0.1	TCP	44	49798 → 49799 [ACK] Seq=1 Ack=4 Win=10233 Len=0
43	19.751551	127.0.0.1	127.0.0.1	TCP	45	49799 → 49798 [PSH, ACK] Seq=4 Ack=1 Win=10233 Len=1
44	19.751590	127.0.0.1	127.0.0.1	TCP	44	49798 → 49799 [ACK] Seq=1 Ack=5 Win=10233 Len=0
45	19.752034	127.0.0.1	127.0.0.1	TCP	45	49799 → 49798 [PSH, ACK] Seq=5 Ack=1 Win=10233 Len=1
46	19.752056	127.0.0.1	127.0.0.1	TCP	44	49798 → 49799 [ACK] Seq=1 Ack=6 Win=10233 Len=0
47	19.955682	127.0.0.1	127.0.0.1	TCP	640	49724 → 49785 [PSH, ACK] Seq=1 Ack=1 Win=10233 Len=596
48	19.955715	127.0.0.1	127.0.0.1	TCP	44	49785 → 49724 [ACK] Seq=1 Ack=597 Win=10128 Len=0
49	19.955872	127.0.0.1	127.0.0.1	TCP	45	49781 → 49780 [PSH, ACK] Seq=1 Ack=1 Win=10233 Len=1
50	19.955898	127.0.0.1	127.0.0.1	TCP	44	49780 → 49781 [ACK] Seq=1 Ack=2 Win=10233 Len=0

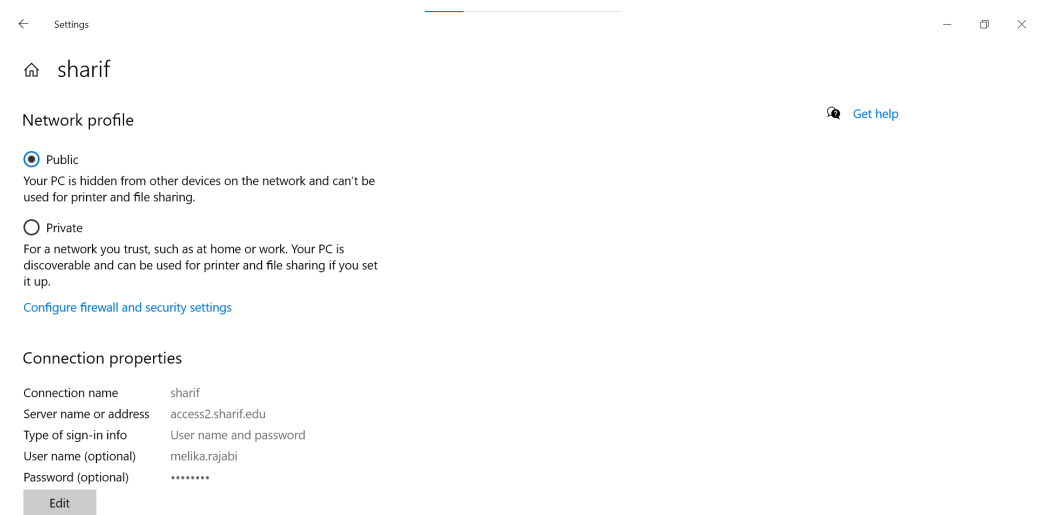
The biggest TCP packet containing data had 65539 bytes. The following image is representing one of the biggest packets:

```
> Frame 28: 65539 bytes on wire (524312 bits), 65539 bytes captured (524312 bits) on interface \Device\NPF_{Loopback, id 0}
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 43130, Dst Port: 50043, Seq: 1, Ack: 1, Len: 65495
  Data (65495 bytes)
    Data [truncated]: 0d8821ab27a7a67a1fd6ae06fdb2fd144a7e0355fd068c5e6ccc918a19c37ea2e46462f719c6c4c4b93addc983f93f4b617443fc20bbcf51662
    [Length: 65495]
```

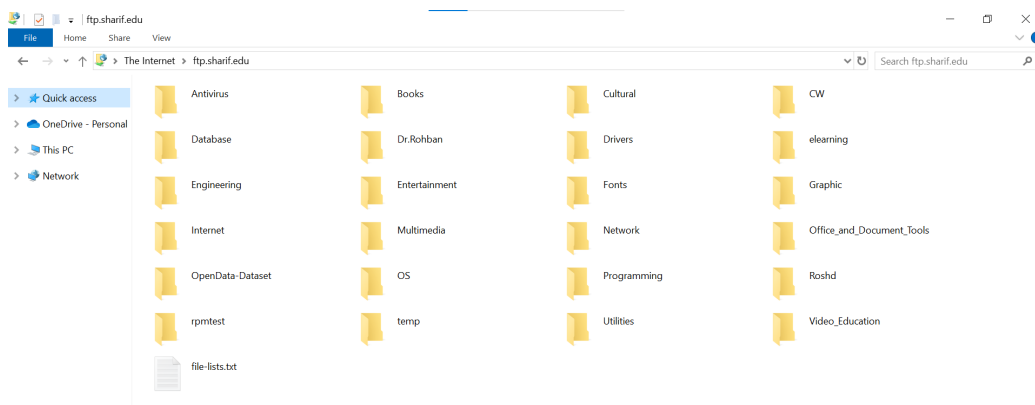
Moreover, by doing a research we realize that the maximum size of a TCP packet is truly $65535 \approx 64K$ bytes which is restricted by the Maximum Transmission Unit (MTU) of network resources. MTU is the maximum size of the data transfer limit set by hardware in a network. The packet size should never exceed MTU.

2.7 Sharif FTP Server

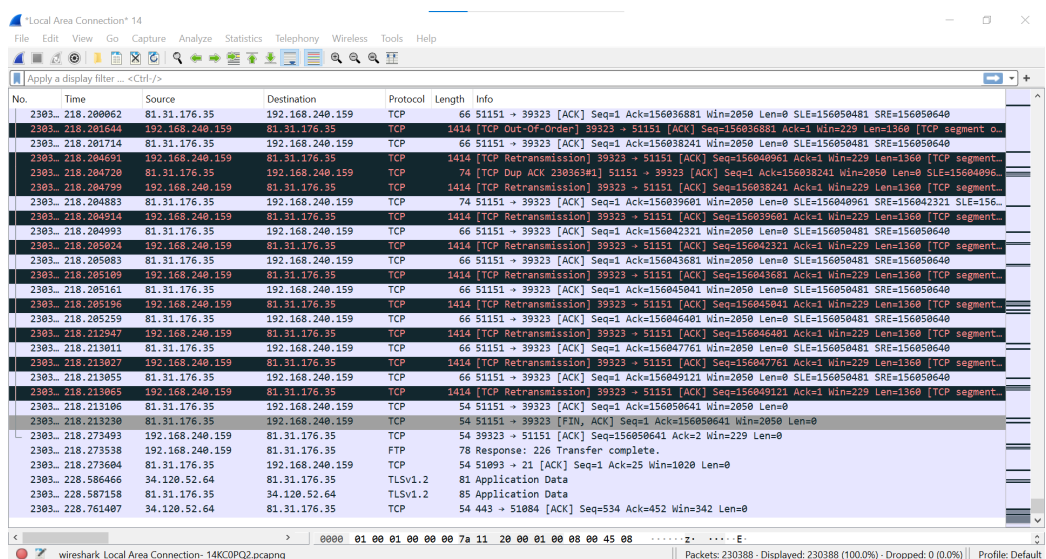
First of all, to connect to the university's network remotely, we have to set up a VPN as follows:



Now, by entering `"/ftp://ftp.sharif.edu/"` in the file explorer, we are able to download (copy) files from the server to our own computer.



By observing the network while copying a file using Wireshark, we result in the following capture:



There were some slight differences between the packets in the previous part and this part. In this part, packets were smaller and actually had the same size. Whereas in the previous part, packets were from different size orders. This is because in our local FTP server, we haven't set any rules beyond the socket library on the size of the packets. However, there may be some restrictions and limitations on the size of the packets in the university's FTP server. This makes the error correction easier and decreases the error rate.

Part 3: Setting Up a Local FTP Server on Ubuntu

Part A: Install and Configure FTP Server

For this part, I have used the following commands in the terminal in Ubuntu:

To install the required package:

```
sudo apt install vsftpd
```

To launch the service and enable it to automatically start at boot:

```
sudo systemctl start vsftpd  
sudo systemctl enable vsftpd
```

```
melikarajabi@melikarajabi:~/Desktop$ sudo systemctl start vsftpd  
melikarajabi@melikarajabi:~/Desktop$ sudo systemctl enable vsftpd  
Synchronizing state of vsftpd.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable vsftpd
```

To Create a backup copy of the default configuration file:

```
sudo cp /etc/vsftpd.conf /etc/vsftpd.conf_default
```

To add users' usernames and passwords:

```
sudo useradd -m [username]  
sudo passwd [username]  
[password]
```

```
melikarajabi@melikarajabi:/etc$ sudo useradd -m sara  
melikarajabi@melikarajabi:/etc$ sudo passwd sara  
New password:  
BAD PASSWORD: The password is shorter than 8 characters  
Retype new password:  
passwd: password updated successfully
```

To make the firewall open ports 20 and 21 for FTP traffic:

```
sudo ufw allow 20/tcp  
sudo ufw allow 21/tcp
```

```
melikarajabi@melikarajabi:/etc$ sudo ufw allow 20/tcp
Rules updated
Rules updated (v6)
melikarajabi@melikarajabi:/etc$ sudo ufw allow 21/tcp
Rules updated
Rules updated (v6)
```

To connect to the FTP server:

```
sudo ftp [system name]
```

```
melikarajabi@melikarajabi:/etc$ sudo ftp melikarajabi
Connected to melikarajabi.
220 (vsFTPd 3.0.5)
Name (melikarajabi:melikarajabi): sara
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

After entering the username and password defined earlier, the server is ready to get command.

Part B: Configure Firewall

To check the firewall status:

```
sudo ufw status
```

```
melikarajabi@melikarajabi:~$ sudo ufw status
Status: active
```

To	Action	From
--	-----	----
20/tcp	ALLOW	Anywhere
21/tcp	ALLOW	Anywhere
20/tcp (v6)	ALLOW	Anywhere (v6)
21/tcp (v6)	ALLOW	Anywhere (v6)

As explained in the previous section, we have configured firewall to allow FTP traffic, ports 20 and 21.

Question 1: Why is it important to configure firewall rules for FTP traffic?

Firewall rules limit access to authorized FTP connections, reducing the risk of unauthorized file transfers or server breaches. But configuring it for FTP is more important than other protocols. FTP is a messy protocol because it requires multiple connections, sometimes in both directions. Random high-numbered ports are used for data transfer. A strict firewall might block these connections by default as they are non-standard ports. Firewall rules can be set to allow a range of ports to accommodate these connections.

Question 2: What are the specific ports used for FTP, and how do you open them in the firewall?

Ports 20 and 21 are used for FTP. To open them in the firewall we can use "sudo ufw allow [number of port]" command as explained earlier.

Part C: Change Default Directory

As explained in Part A, we make a copy of the configuration file "vsftpd.conf" using "cp" command.

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

This is because usually many times you might need to modify a config, include, or template file. Having a back up makes us able to revert to our saved copy if made a mistake.

To create a new directory:

```
sudo mkdir /srv/ftp/[new_location]
```

To change the default directory where the ftp user will be placed upon login:

```
sudo usermod -d /srv/ftp/[new_location] ftp
```

To restart the vsftpd service to apply the changes:

```
sudo systemctl restart vsftpd.service
```

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

By default, FTP users often land in their home directory. This could potentially give them access to sensitive files or mess up system configurations if their home directory isn't properly secured. Changing the default to a specific directory restricts where they can go and what they can see. Moreover you can set up permissions on specific directories, so even if an FTP user can navigate around a bit, they can only see and modify files they're authorized to access.

Also if you have multiple FTP users for different purposes, it can be helpful to have them all start in a directory relevant to their task. This makes things clearer and avoids confusion.

I have created a directory named "media" and added an image file to it. The client could access the file:

```
ftp> ls
229 Entering Extended Passive Mode (|||26181|)
150 Here comes the directory listing.
-rw-rw-r-- 1 1000 1000 4244 Jul 07 09:53 image.jpeg
226 Directory send OK.
ftp> █
```

Part D: Securing FTP

Encrypt FTP Traffic

One method to secure our FTP server is to encrypt the traffic. This is done through FTPS (FTP over SSL (Secure Socket Layer)).

For this to work, users need to be set up with a shell account on the FTP server. This adds a layer of secure encryption to the FTP traffic.

To create a new certificate with openssl:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem
```

Then provide the required information when prompted, or keep the default configuration by pressing Enter.

In the next step open the "vsftpd.conf" file and change the "ssl_enable=NO" to "ssl_enable=YES".

Then, add the following lines to the configuration file:

```
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_ciphers=HIGH
pasv_min_port=40000
pasv_max_port=50000
```

Now restart the service to apply the changes:

```
sudo systemctl restart vsftpd.service
```

Question 1: Why is it important to encrypt FTP traffic?

FTP traffic itself, by default, is unencrypted. This means any data transferred, including usernames, passwords, and the files themselves, can be intercepted by someone snooping on the network. Encrypting FTP traffic addresses these security concerns.

Encryption scrambles the data being transferred using a secret key. This makes it unreadable to anyone who intercepts it without the key, protecting sensitive information like usernames, passwords, and the content of the files themselves.

Also it provides a layer of authentication, verifying that you're connecting to the intended server and not a malicious imposter.

Furthermore, encryption helps ensure that the data hasn't been tampered with during transfer. With encryption, any changes to the data will be detected, preventing unauthorized modifications or data corruption.

Question 2: What are the benefits of using SSL/TLS encryption for FTP?

There are several significant benefits to using SSL/TLS encryption for FTP, which creates a secure version known as FTPS:

Enhanced Confidentiality: The primary advantage is that SSL/TLS encrypts all data transferred during an FTP session. This makes it unreadable to anyone eavesdropping on the network, protecting sensitive information from unauthorized access.

Improved Integrity: SSL/TLS ensures data integrity by detecting any modifications during transfer. Any tampering with the data will be flagged, preventing corrupted files or compromised information from reaching their destination.

Stronger Authentication: Some FTPS implementations offer an additional layer of authentication. This verifies that you're connecting to the legitimate FTP server and not a malicious imposter trying to steal your credentials.

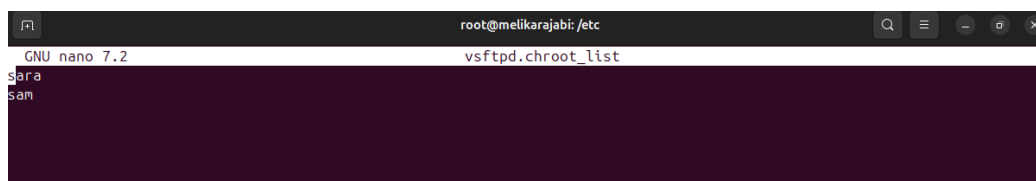
Limit User Access

Another method of securing vsftpd is to limit users to their home directory.

First of all, open the configuration file and uncomment the following line:
`chroot_local_user=YES`

In the next step, we create the user list file. This enables vsftpd to deny or allow access to the listed users. The listed users will have full access to anywhere on the server. If a user is not listed, they are restricted to their specified home directory.

Open the "vsftpd.chroot_list" file and add one user per line to create the list file.



```
root@melikarajabi: /etc
GNU nano 7.2 vsftpd.chroot_list
sara
sam
```

Then, open the "vsftpd.conf" file and uncomment the following lines:

```
chroot_local_user=YES
chroot_list_file=/etc/vsftpd.chroot_list
```

```

chroot_local_user=YES
#
# You may specify an explicit list of local users to chroot to
# directory. If chroot_local_user is YES, then this list overrides
# the default list of users to NOT chroot().
# (Warning! chroot'ing can be very dangerous. Only use this
# if the user does not have write access to the directory, and
# if you have the user's home directory in the chroot.)
chroot_local_user=YES
#chroot_list_enable=YES
# (default follows)
chroot_list_file=/etc/vsftpd.chroot_list
#

```

Finally restart the vsftpd service as explained previously.

Note that we can also create the list of blocked users from FTP access which is stored in the `/etc/ftpusers` file. To add blocked users, edit this file and add one user per line.

Question 1: Why do we need to secure FTP servers?

FTP servers, by default, are like sending postcards. They transmit information, including usernames, passwords, and the data itself, openly across the internet. This lack of security makes them vulnerable to a number of threats, which is why securing FTP servers is crucial.

Question 2: What are the methods to limit user access on an FTP server?

- Account Creation: Implement a system for creating user accounts with unique and strong passwords. Avoid default accounts or easily guessable credentials.

User Roles: Assign user roles with specific permissions. This allows you to grant access only to the directories and functions each user needs, restricting their ability to navigate or modify unauthorized areas.

Home Directory Jail: Configure users to start in a specific directory (their jail) upon login. This restricts their ability to wander the server filesystem.

and potentially access sensitive areas.

Permissions Management: Utilize file system permissions (like `chmod`) to control access to specific directories and files. Set permissions so that only authorized users can read, write, or execute files within a directory.

Anonymous Access: Disable anonymous login if you don't require it. This eliminates the possibility of unauthorized users gaining access to your server.

IP Address Restrictions: Configure the FTP server to only allow connections from specific IP addresses. This can be helpful if you know which users will be legitimately accessing the server.

FTPS Encryption: Implement FTPS, which uses SSL/TLS encryption to protect all data transferred during an FTP session.

Part 4: Bandwidth and Transfer Rate Control

Question : Why is bandwidth control important in network applications?

Bandwidth control is crucial for maintaining smooth operation in network applications. Imagine a highway with a limited number of lanes. Without control, too many cars (data packets) trying to use the highway at once would lead to traffic jams (network congestion). Bandwidth control acts like traffic management for your network, ensuring everything runs efficiently.

Optimizing Network Performance: By regulating data flow, bandwidth control prevents congestion and minimizes latency (delays). This ensures critical applications like video conferencing or VoIP calls have the resources they need to function smoothly.

Prioritizing Traffic: Network administrators can prioritize essential traffic over non-critical ones. For instance, video conferencing can be given higher priority compared to less urgent tasks like downloading large files.

Fairness and Efficiency: Bandwidth control prevents any single user or application from hogging all the bandwidth. This ensures everyone on the network gets a fair share and contributes to a more efficient overall network experience.

Security: Bandwidth control can be used to limit bandwidth for suspicious activity or certain types of traffic, potentially helping to mitigate security threats.

To observe the transmission rate, I have chunked the transmitted files into smaller packets (134217728 bits). Then, using the "time" library, I have measured the time spent to transmit each packet. Finally, by dividing the packet size to the duration, we result in the transmission rate. We can also find the transmission progress by dividing the length of sent packets to the total size of the file. Note that in the client, we have to receive the chunked file as well rather than the whole file.

The result for a pretty big file is as follows:

```
In [1]: runfile('C:/Users/Asus/Desktop/Part4/server/
server.py', wdir='C:/Users/Asus/Desktop/Part4/server')
bigfile 1073741824b is being transfered.
Transmission Rate = 1.7179869184 Gbps
Transmission Progress = 12.5 %
Transmission Rate = 4.294967296 Gbps
Transmission Progress = 25.0 %
Transmission Rate = 4.294967296 Gbps
Transmission Progress = 37.5 %
Transmission Rate = 4.294967296 Gbps
Transmission Progress = 50.0 %
Transmission Rate = 4.294967296 Gbps
Transmission Progress = 62.5 %
Transmission Rate = 2.8633115306666665 Gbps
Transmission Progress = 75.0 %
Transmission Rate = 4.294967296 Gbps
Transmission Progress = 87.5 %
Transmission Rate = 4.294967296 Gbps
Transmission Progress = 100.0 %

In [1]: runfile('C:/Users/Asus/Desktop/Part4/client/
client.py', wdir='C:/Users/Asus/Desktop/Part4/client')
get bigfile
150 OK to send data
226 Transfer complete
```

In the next step, I have set the maximum bandwidth limit to 3 *Gbps*. By using the "time" library, I have added an appropriate amount of delay so that the transmission rate stays below the maximum bandwidth. The results are as follows:

```

In [1]: runfile('C:/Users/Asus/Desktop/Part4/server/
server.py', wdir='C:/Users/Asus/Desktop/Part4/server')
bigfile 1073741824b is being transfered.
Transmission Rate = 1.7179869184 Gbps
Transmission Progress = 12.5 %
Transmission Rate = 3.0000000000000107 Gbps
Transmission Progress = 25.0 %
Transmission Rate = 2.8633115306666665 Gbps
Transmission Progress = 37.5 %
Transmission Rate = 3.0000000000000107 Gbps
Transmission Progress = 50.0 %
Transmission Rate = 2.8633115306666665 Gbps
Transmission Progress = 62.5 %
Transmission Rate = 3.0000000000000107 Gbps
Transmission Progress = 75.0 %
Transmission Rate = 3.0000000000000107 Gbps
Transmission Progress = 87.5 %
Transmission Rate = 3.0000000000000107 Gbps
Transmission Progress = 100.0 %

In [1]: runfile('C:/Users/Asus/Desktop/Part4/client/
client.py', wdir='C:/Users/Asus/Desktop/Part4/client')
get bigfile
150 OK to send data
226 Transfer complete

```

As we see, the transmission rate actually stays below 3 *Gbps*.

References

- [1] github.com/hadis98/FTP-Client-Server-Python-Socket-Programming
- [2] github.com/nikhilroxtomar/Multithreaded-File-Transfer-using-TCP-Socket-in-Python/blob/main/server.p
- [3] www.geeksforgeeks.org/difference-between-ftp-and-http/
- [4] thwack.solarwinds.com/resources/b/geek-speak/posts/what-is-the-difference-between-ftp-https-sftp
- [5] www.jscape.com/blog/understanding-key-differences-between-ftp-https-and-sftp
- [6] www.integrate.io/blog/the-complete-guide-to-ftp-https-sftp-and-scp/
- [7] www.jscape.com/blog/12-file-transfer-protocols-businesses
- [8] pro2col.com/tcp-udp-fast-file-transfer
- [9] essaygpt.hix.ai/essay/advantages-and-disadvantages-of-ftp-712f61
- [10] www.cosmos.esa.int/documents/772136/977578/psa_activeVsPassiveFtp.pdf
- [11] pro2col.com/blog/what-is-the-difference-between-active-and-passive-ftp
- [12] www.techtarget.com/searchnetworking/definition/File-Transfer-Protocol-FTP
- [13] kiteworks.com/risk-compliance-glossary/sftp
- [14] www.ssh.com/academy/ssh/ftp/https
- [15] www.awardspace.com/kb/what-is-ftp-over-ssh/
- [16] pinetools.com/random-file-generator
- [17] www.baeldung.com/cs/tcp-max-packet-size
- [18] ictc.sharif.edu
- [19] ictc.sharif.edu/
- [20] www.awardspace.com/kb/what-is-ftp-over-ssh/
- [21] www.inmotionhosting.com/support/website/backup-a-file-before-editing