THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Thapar Institute of Engineering and Technology, Patiala

Computer Science and Engineering
Network Programming Laboratory (UCS413)
Assignment:1

1. Familiarity with Lab environment and understanding Client-Server model, Unix basic commands, socket programming headerfiles, and elementary socket system calls.

   (a) Understanding and using of commands like ifconfig, netstat, ping, arp, telnet, ftp, finger, traceroute, whois etc.

   (b) Socket header files contain data definitions, structures, constants, macros, and options used by socket subroutines. An application program must include the appropriate header file to make use of structures or other information a particular socket subroutine requires.

   Commonly used socket header files are:

   - **/usr/include/netinet/in.h**: Defines Internet constants and structures.
   - **/usr/include/netdb.h** Contains data definitions for socket subroutines.
   - **/usr/include/sys/socket.h** Contains data definitions and socket structures.
   - **/usr/include/sys/types.h** Contains data type definitions.
   - **/usr/include/arpa.h** Contains definitions for internet operations.
   - **/usr/include/sys/errno.h** Defines the errno values that are returned by drivers and other kernel-level code.

   (c) Elementary socket system calls:

   - **socket() System Call:** Creates an end point for communication and returns a descriptor:
     int socket (int AddressFamily, int Type, int Protocol);
   - **Bind( ) System call:** Binds a name to a socket. The bind subroutine assigns a Name parameter to an unnamed socket. It assigns a local protocol address to a socket.
     int bind (int sockfd, struct sockaddr *myaddr, int addrlen);
   - **connect() System call:** The connect function is used by a TCP client to establish a connection with a TCP server.
     int connect(int sockfd, struct sockaddr *servaddr, int addrlen);
   - **listen() System call:** This system call is used by a connection-oriented server to indicate that it is willing to receive connections.
     int listen (int sockfd, int backlog);
   - **accept() System call:** The actual connection from some client process is waited for by having the server execute the accept system call.
     int accept (int sockfd, struct sockaddr *cliaddr, int *addrlen);
   - **send( ), sendto( ), recv( ) and recvfrom( ) system calls:** These system calls are similar to the standard read and write functions.
   - **close( ) system call:** The normal Unix close function is also used to close a socket and terminate a TCP connection.
     int close (int sockfd);

How To Install an FTP Server On Ubuntu with vsftpd

Step 1: Update System Packages

Start by updating your repositories – enter the following in a terminal window:

```
sudo apt update
```

Step 2: Install vsftpd Server on Ubuntu

A common open-source FTP utility used in Ubuntu is **vsftpd**. It is recommended for its ease of use.

1. To install **vsftpd**, enter the command:

```
sudo apt install vsftpd
```

Step 3: Backup Configuration Files

Before making any changes, make sure to back up your configuration files.

1. Create a backup copy of the default configuration file by entering the following:

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

Step 4: Create FTP User

Create a new FTP user with the following commands:

```
sudo useradd -m testuser

sudo passwd testuser
```

The system should ask you to create a password for the new **testuser** account.

Step 5: Configure Firewall to Allow FTP Traffic

If you are using UFW that comes standard with Ubuntu, it will block FTP traffic by default. Enter the following commands to open **Ports 20** and **21** for FTP traffic:

```
sudo ufw allow 20/tcp

sudo ufw allow 21/tcp
```

Step 6: Connect to Ubuntu FTP Server

Connect to the FTP server with the following command:

```
sudo ftp ubuntu-ftp
```

Replace ***ubuntu-ftp*** with the name of your system (taken from the command line).

Log in using the **testuser** account and password you just set. You should now be successfully logged in to your FTP server.

Configuring and Securing Ubuntu vsftpd Server

Change Default Directory

By default, the FTP server uses the **/srv/ftp** directory as the default directory. You can change this by creating a new directory and changing the FTP user home directory.

To change the FTP home directory, enter the following:

```
sudo mkdir /srv/ftp/new_location

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

Restart the **vsftpd** service to apply the changes:

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

Now, you can put any files you want to share via FTP into the **/srv/ftp** folder (if you left it as the default), or the **/srv/ftp/*new_location*/** directory (if you changed it).

Authenticate FTP Users

If you want to let authenticated users upload files, edit the **vsftpd.conf** file by entering the following:

```
sudo nano /etc/vsftpd.conf
```

Find the entry labeled ***write_enable=NO***, and change the value to "**YES**."

```
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
```

Save the file, exit, then restart the FTP service with the following:

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

This allows the user to make changes inside their home directory.

Note: To learn more about using FTP, refer to our in-depth guide on Linux ftp command.

Securing FTP

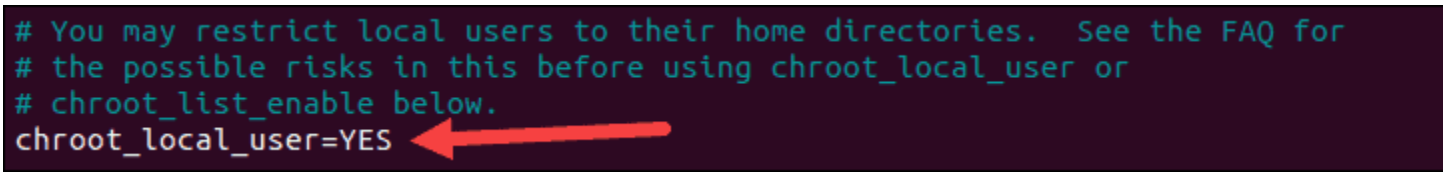
Numerous exploits take advantage of unsecured FTP servers. In response, there are several configuration options in **vsftpd.conf** that can help secure your FTP server.

Limit User Access

One method is to limit users to their home directory. Open **vsftpd.conf** in an editor and uncomment the following command:

```
chroot_local_user=YES
```

This is an example of the file in **nano**:



Create a User List File

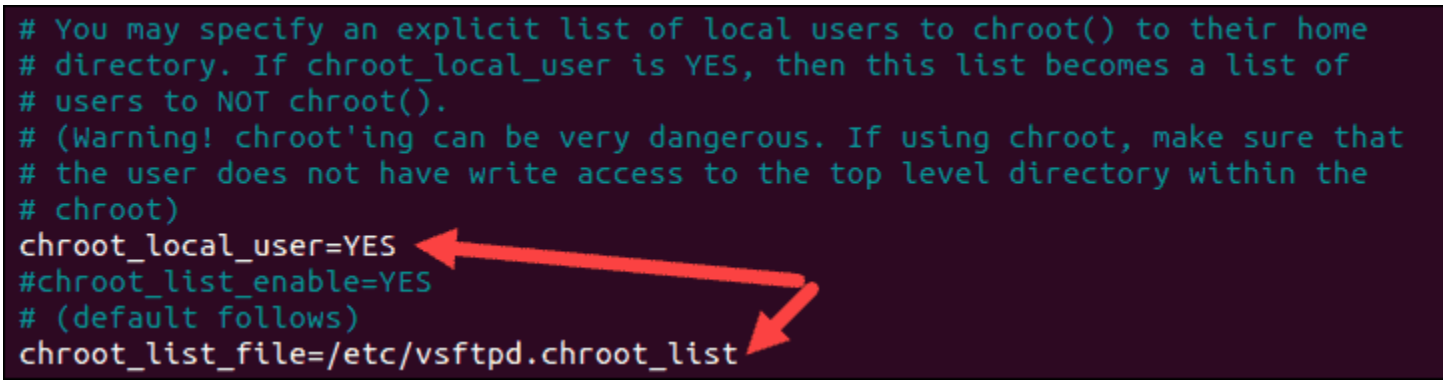
To create a list file, edit **/etc/vsftpd.chroot_list**, and add one user per line.

Instruct your FTP server to limit this list of users to their own home directories by editing **vsftpd.conf**:

```
chroot_local_user=YES

chroot_list_file=/etc/vsftpd.chroot_list
```

The image ilustrates the edits that were made:



Restart the **vsftpd** service:

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

By default, the list of blocked users from FTP access is stored in **/etc/ftpusers**. To add blocked users, edit this file and add one user per line.

Client side Commands to connect

To Connect to VSFTPD

ftp server-ip-address

for example
tp 192.168.0.101

it will ask to enter our FTP username

Below are **some of the most common FTP commands** that we can use:

- cd - Change directory on remote machine.
- lcd - Change directory on local machine.
- ls - View the names of the files and directories in the current remote directory.
- mkdir - Create a new directory within the remote directory.
- pwd - Print the current working directory on the remote machine.
- delete - Delete a file in the current remote directory.
- rmdir- Remove a directory in the current remote directory.
- get - Copies a file from the remote server to the local machine.
- mget - Allows you to copy multiple files from the remote server to the local machine.
- put - Copies a file from the local machine to the remote machine.
- mput - Copies a file from the local machine to the remote machine.

How to Install Telnet Server

By default, the Telnet server package is available in the Ubuntu 20.04 default repository. You can install it by just running the following command:

```
sudo apt install telnetd -y
```

Once the installation is completed, you can check the status of Telnet service using the following command:

```
sudo systemctl status inetd
```

Output:

```
? inetd.service - Internet superserver
   Loaded: loaded (/lib/systemd/system/inetd.service; enabled; v
endor preset: enabled)
   Active: active (running) since Mon 2021-04-29 10:24:05 UTC; 3
8s ago
     Docs: man:inetd(8)
 Main PID: 2883 (inetd)
    Tasks: 1 (limit: 1114)
   CGroup: /system.slice/inetd.service
           ??2883 /usr/sbin/inetd


Apr 29 10:24:05 ubuntu2004 systemd[1]: Starting Internet superse
rver...
Apr 29 10:24:05 ubuntu2004 systemd[1]: Started Internet superser
ver.
Test Telnet Connection from Remote System
Telnet server is now installed and listening on port 23. It's ti
me to connect the Telnet server from the remote system.
```

Now, log in to another Ubuntu system and run the following command:

```
telnet 192.168.0.100
```

You will be asked to enter your username and password. After successful authentication, you should see the following output:

```
Trying 192.168.0.100...
Connected to 192.168.0.100.
Escape character is '^]'.
Ubuntu 20.04 LTS
ubuntu2004 login: hitesh
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-72-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 29 Apr 2021 09:16:14 AM UTC

  System load:            0.83
  Usage of /:             14.7% of 39.12GB
  Memory usage:           30%
  Swap usage:             0%
  Processes:              163
  Users logged in:        0
```

```
IPv4 address for ens33: 192.168.0.100
IPv6 address for ens33: 2003:e1:bf4b:8b00:20c:29ff:fef5:ee3c
```

You can now run any command on the Telnet server using Telnet.

Install OpenSSH

Install it by running the following command in your terminal:

```
sudo apt-get install openssh-server -y
```

After the installation is done, you'll have SSH enabled on your Ubuntu desktop.

Configure SSH

Now that you've installed SSH, you can configure it. Like changing the default port (recommended for security reasons), disabling "root" user login etc.

For now, we'll just update our default SSH port (which is 22). First, open up the ssh configuration file by running the following command:

```
sudo nano /etc/ssh/sshd_config
```

If you don't have nano installed (it's a text editor), run this command:

```
sudo apt-get install nano -y
```

Once you open the file, find and change the following line from:

```
# Port 22
```

to

```
Port 1337
```

Use a different port number, whichever one you want to.

Once you are done, save and close the file with `Ctrl + W`, then `Y` and hit `Enter/Return`

Before restarting SSH you need to configure your firewall to allow the port you provided before. If you're using UFW, just run:

```
sudo ufw allow 1337
```

You need to check with your internet provider and your modem/router(s) if you need to allow the new port.

Now, restart SSH for the changes to take effect. Run the following command:

```
sudo service ssh restart
```

And that's it. You are done.

Now you can use SSH to log into your server. Just open up Terminal and run:

```
ssh username@ip -p1337
```

To log into your server via SSH, right from your Ubuntu desktop terminal. Of course, change 'username', 'ip' and the port number you're using on your Ubuntu server

Also you can use putty or other SSH Client programs to connect