Project description

Learn Net is a server-client project that covers some of the networking concepts for learning/teaching proposes. In both sides some setting options are considered so the user can test the different methods. For example, a message can be sent either as a plain text or cipher text.

Table of Contents

Manual

Server

Run the software

Linux:

1. From Linux Terminal: ./server

2. Double click the server.desktop file that should be in the same directory where server file is located.

If you received the untrusted launch application message, simply click the Trust and Launch button.

Start listening

Set the port number and click on the listening. To set the access policy read the access limitation section.

Access limitation

Settings → availability

Options:

1. Everyone: Any client including our software without authentication can connect to the server.

2. Our clients: Only our authenticated users with our client software can connect to the server.

Show and hide the windows:

In the view tab by clicking on the options you can hide/show them.

Stop listening

Click on the stop button. If you stop the listening when there are connected clients, automatically all of them will be disconnected and their sockets will be closed.

Close the software

You can either close the software by clicking on the close button in menu tab or close button in the top right of the software. It is always better to first stop the listening before closing the software.

Client

Run the software

Linux:

1. From Linux Terminal: ./client

2. Double click the client.desktop file that should be in the same directory where client file is located.

If you received the untrusted launch application message, simply click the Trust and Launch button.

Connect to the server

Set the four required parameters and click the connect button. The security method of user name and password can be selected in the settings tab.

Disconnect from the server

Click on the disconnect button.

Send a message

Fill the message box and click the send button. The encryption algorithm can be set from settings tab in the encryption sub field.

Set the message encryption algorithm

Settings → Encryption

Set the message encryption key

After selecting the encryption algorithm, bellow the message box the encryption box appears. Select a key and click the set button.

Note: Encryption key for each encryption algorithm must be set separately.

Set authentication security

Settings → Authentication security

Close the software

You can either close the software by clicking on the close button in menu tab or close button in the top right of the software. It is always better to first disconnect before closing the software.

Test cases

Server

Start button

---

1. User starts the server without setting the port number.

Desired output:

-Action/s: Display proper message.

-Message:

The port number is required to start the server.

-Place of message:

Main box (connection messages).

---

2. User starts the server with an invalid port number. Invalid port number is either smaller than 2000 or bigger than 65535.

Desired output:

-Action/s: Display proper message.

-Message:

The port number is not valid. It must be a number between 2000 and 65535.

-Place of message:

Main box (connection messages).

---

3. User starts the server with a proper input.

Desired output:

-Action/s: 1. Display proper message, 2. Start listening.

-Message:

The server has started and we are listening for new connections.

-Place of message:

Main box (connection messages).

---

4. User starts the server when it is running.

Desired output:

-Action/s: Display proper message.

-Message:

The server is already running.

-Place of message:

Main box (connection messages).

---

5. User starts the server with a proper input but the server can not start listening. For example another software use the selected port number and therefore the server can not use it.

Desired output:

-Action/s: Display proper message.

-Message:

Listen() error.

-Place:

Main box (connection messages).

Stop button

---

1. User stops the server when it is working.

Desired output:

-Action/s: 1. Display proper message, 2. Stop listening.

-Message:

The server has stopped.

-Place of message:

Main box (connection messages).

---

2. User stops the server when it is not working.

Desired output:

-Action/s: Display proper message.

-Message:

The server is already stopped.

-Place of message:

Main box (connection messages).

Got a new client

---

1. A new client connects to the server.

Desired output:

-Action/s: 1. Display proper message, 2. Accept the new connection.

-Message:

We got a new connection from IP: <Client’s IP address> port: <Client’s port number>

-Place of message:

Main box (connection messages).

Lost a client

---

1. A client disconnects from the server.

Desired output:

-Action/s: 1. Display proper message, 2. Close the client’s socket.

-Message:

We lost a connection from IP: <Client’s IP address> port: <client’s port number>

-Place of message:

Main box (connection messages).

Receive a client’s message

---

1. The server receives a message from one of the clients.

Desired output:

-Action/s: 1. Display proper message, 2. Send the message to all of the clients.

-Message:

<Client’s user name> : <Client’s message>

-Place of message:

Chat box.

Client

Connect button

---

1. User press the connect button without all the required inputs.

Desired output:

-Action/s: Display proper message.

-Message:

There are four messages according to the user input:

a. The IP address is not valid.

b. The port number is not valid. It must be a number between 2000 and 65535.

c. The user name field must be set.

d. The user name can not be longer than 19 characters.

For example if the user insert a correct IP address and left the other fields, the b and c messages will be printed.

-Place of message:

Connection box.

---

2. The user press the connect button when it is already connected.

Desired output:

-Action/s: Display proper message.

-Message:

We are already connected.

-Place of message:

Connection box.

---

3. The user press the connect with proper input values.

Desired output:

-Action/s: 1. Display proper message, 2. connects to the server.

-Message: We have connected to IP: <Server’s IP address> port: <Server’s port number>

-Place of message:

Connection box.

---

4. The user press the connect with proper inputs in syntax but, the IP address and port number are not matched with the server.

Desired output:

-Action/s: Display proper message.

-Message: Because of one of the following reasons we failed to connect.

1. The IP address and port number are not matched with the server.

2. The server is not running.

-Place of message:

Connection box.

---

5. The user try to connect when the server is not running.

Desired output:

-Action/s: Display proper message.

-Message: Because of one of the following reasons we failed to connect.

1. The IP address and port number are not matched with the server.

2. The server is not running.

-Place of message:

Connection box.

Disconnect button

---

1. The user press the disconnect button when it is already disconnected.

Desired output:

-Action/s: Display proper message.

-Message:

We are already disconnected.

-Place of message:

Connection box.

---

2. The user press the disconnect button when it is connected and it can successfully disconnect.

Desired output:

-Action/s: 1. Display proper message, 2. Disconnect from the server.

-Message:

We have disconnected from IP: <Server's IP address> port: <Server's port number>

-Place of message:

Connection box.

---

3. The user press the disconnect button when it is connected but, for any technical reason it can not disconnect.

Desired output:

-Action/s: Display proper message.

-Message:

We failed to disconnect.

-Place of message:

Connection box.

Server closes the connection

---

1. When the client is connected and the server close the connection.

Desired output:

-Action/s: 1. Display proper message, 2. Close the socket.

-Message:

The server has closed our connection.

-Place of message:

Connection box.

Receive a message from the server (other clients messages)

---

1. Receive a message from one server which is the message of one of the clients.

Desired output:

-Action/s: 1. Display proper message.

-Message:

<Client’s user name> : <Client’s message>

-Place of message:

Chat box.

Send button

---

1. The user click the send button without any message.

Desired output:

-Action/s: 1. Display proper message.

-Message:

Empty message is NOT allowed.

-Place of message:

Chat box.

---

2. The user try to send a message which is longer than 80 characters.

Desired output:

-Action/s: 1. Display proper message.

-Message:

The message should NOT be longer that 80 characters.

-Place of message:

Chat box.

---

3. The user send a message which has less than 80 characters.

Desired output:

-Action/s: 1. Display proper message, 2. Send the message to the server.

-Message:

<Client’s user name> : <Client’s message>.

-Place of message:

Chat box.

---

4. The user clicks the send button when it is not connected to the server.

Desired output:

-Action/s: 1. Display proper message, 2. Send the message to the server.

-Message:

We are NOT connected. Pleas first connect and try again.

-Place of message:

Chat box.

Future development steps

1. Adding all the features to the document.

2. Adding at least one asymmetric encryption algorithms.

3. Adding special modes for the client like SNTP.