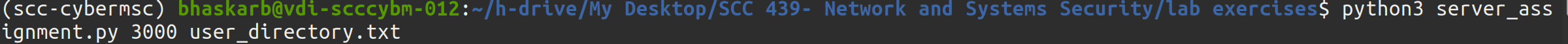
**Chat App execution instructions and test results**

**Execution:**

1. There are three files: server\_assignment.py, client\_assignment.py, user\_directory.txt and
2. server\_assignmnet.py has the functionalities required by server program to perform mutual chap validation and securely exchanging (send and receive) messages using encryption
3. client\_assignment.py has the functionalities required by server program to perform mutual chap validation and securely exchanging (send and receive) messages using encryption
4. user\_directory.txt has the list of users available to contact with the details of username, port no, ip address and password.
5. To execute server\_assignmnet.py use below command:

python3 server\_assignment.py ‘port\_no’ ‘user\_directry’ (input port no and user directory file name)

eg: python3 server\_assignment.py 3000 user\_directory.txt



1. After server is running. Please do the following steps accordingly:

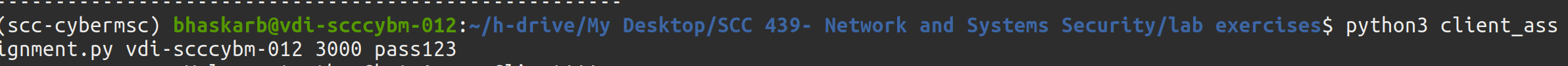
Receive a message: Start the client so that server gets an incoming connection.

Send a message: input ‘hi’ in the terminal so that the server knows an incoming message is given for it to send.

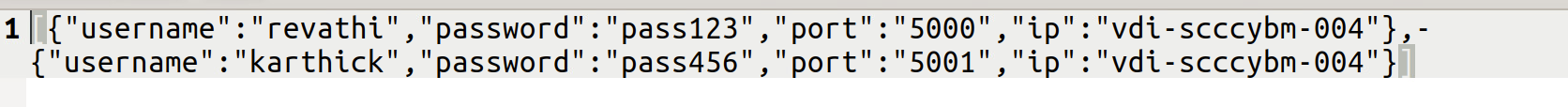
1. To execute client\_assignmnet.py use below command:

python3 client\_assignment.py ‘ip address’ ‘port\_no’ ‘password of the user’

eg: python3 client\_assignment.py vdi-scccybm-012 3000 pass123



1. User\_directory.txt file contains the list of users in json serialized format.



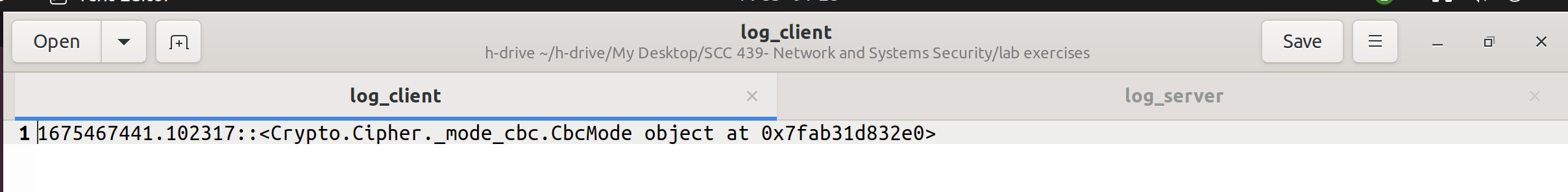
1. The inputs are logged in a secured manner by performing HMAC on the message

log\_server.txt

Graphical user interface, text, application, website

Description automatically generated

log\_client.txt



1. Please use same port number to execute both client and server because I was getting connection refused error when using different ports. I tried debugging but couldn't find.

**Testing**

**CASE -1: Server having incoming connection (server receiving and client sending)**

**Server:**

1. Starting the server connection by proving port no and user directory file name
2. The server connection is created, and it is in listening mode waiting for either an incoming message or incoming connection
3. Now, start the client so that the server has an incoming connection to connect.
4. Client sends the hello message with username to start the chap authentication.
5. Once the chap is successful connection is established.
6. The server waits for the message from client.
7. Once it receives the message, it performs the integrity and confidentiality checks by performing decryption, comparing crc and hmac values
8. Once message is confirmed to be not tampered, ack message.
9. Server terminates the connection with client.

**Text

Description automatically generated**

**A picture containing graphical user interface

Description automatically generated**

**Client:**

1. Starting the connection by proving ip, port and password of the user (Provide the client password so that the server can authenticate and connect)
2. Password of the user trying to communicate is stored in the directory on server side so that when the user provides password from client it is authenticated.
3. Enter the username with respective to the password entered to perform mutual authentication.
4. After authentication is done, client is asked for the message to send the server.
5. After message is sent, the client connection is closed.

**Text

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

**CASE -2: Server having inputs from user selected from directory (server sending and client receiving)**

**Server:**

1. The server is in listening mode after the above test case. To indicate it has an incoming input type ’hi’ in terminal.
2. Server is asking user is asked for input either to load from directory or to manually provide the inputs.
3. Here, select 1 so it reads from directory. To that server asks for username.
4. Enter the username to get password from user directory for authentication.
5. Once the chap is successful connection is established.
6. The server asks for the message to send to client.
7. Server generates the pdu with the input message to maintain integrity and confidentiality when transferred over network by performing decryption, comparing crc and hmac values
8. Once the ack message is received from client, server terminates the connection with client.

**Text

Description automatically generated**

**Client:**

1. After the username is given as input in server, start the connection by proving ip, port and password of the user (Provide the client password so that the server can authenticate and connect)
2. Password of the user trying to communicate is stored in the directory on server side so that when the user provides password from client it is authenticated.
3. Enter the username with respective to the password entered to perform mutual authentication.
4. After authentication is done, client receives a message.
5. Client performs the integrity and confidentiality checks by performing decryption, comparing crc and hmac values. Once message is confirmed to be not tampered, ack message.
6. After message is sent, the client connection is closed.

**Text

Description automatically generated**

**CASE- 3: Server having inputs given by user (server sending and client receiving)**

**Server**

1. As mentioned in the previous test cases, ‘hi’ is entered in terminal and the user is asked for input.
2. Since option 2 is selected, user is asked to provide manual input.
3. Once the values are entered, they are compared with dictionary and if they match, a connection is established
4. Now, client needs to be started.
5. Chap authentication is started and if it is successful, user is asked for the message to send the client.
6. Once the message is sent and received ack message, the client connection is closed.

**Text

Description automatically generated**

**Text

Description automatically generated**

**Client**

1. After the username is given as input in server, start the connection by proving ip, port and password of the user (Provide the client password so that the server can authenticate and connect)
2. Password of the user trying to communicate is stored in the directory on server side so that when the user provides password from client it is authenticated.
3. Enter the username with respective to the password entered to perform mutual authentication.
4. After authentication is done, client receives a message.
5. Client performs the integrity and confidentiality checks by performing decryption, comparing crc and hmac values. Once message is confirmed to be not tampered, ack message.
6. After message is sent, the client connection is closed.

**Text

Description automatically generated**

**CASE -4: Server having incorrect inputs (server sending client receiving)**

**Server**

1. The server is in listening mode after the above test case. To indicate it has an incoming input type ‘hi’ in terminal.
2. Server is asking user is asked for input either to load from directory or to manually provide the inputs.
3. Here, select 2 so that user enter the inputs.
4. The values provided as input are checked with directory if it does not match the connection is not established.
5. Here the highlighted port no is wrong when compared to directory (entered val-50001 directory val- 5001)

**Text

Description automatically generated**



**CASE -5: Client connecting with wrong password (server receiving and client sending)**

**Server**

At the server end when the client tries to connect with wrong credentials, an exception is thrown.

Text

Description automatically generated

**Client**

If trying to establish a connection with wrong username and password, the authentication fails, and connection is closed

**Text

Description automatically generated**

**CASE – 6:** if wrong input is given for username in server side Text

Description automatically generated