Name: Bo Chen

E-mail: [bchen80@gatech.edu](mailto:bchen80@gatech.edu)

Class: CS 3251 - Computer Networks I –B

Assignment: Sockets Programming Assignment 1

Platform: Python

Description of assignment:

Developing two different versions of basic online authentication server, one is using TCP protocol; the other one is UDP protocol. The service is based on a shared secret "password" and a challenge response protocol. The client interface will allow users to provide a username and password that will be used to determine whether or not they are granted access. The client command should be called "letmein" and it should read the username and password from the command line. The command should print either "Welcome to our service." or "User authorization failed." as output.

Descriptions of all files:

**server-tcp.py**: TCP server side source code

**letmetin-tcp.py**: TCP client side source code

**server-udp.py**: UDP server side source code

**letmein-udp.py**: UDP client side source code

**sample.txt:**

Instruction for compiling and running:

**Running a server:**

1. Open one terminal window in the computer
2. Use cd command to the folder which contains all the files
3. Command for setting up a TCP server: python server-tcp.py 8591

Command for setting up a UDP server: python server-udp.py 8591

**Running a client after set up a server:**

1. Open a new terminal
2. Use cd command to the folder which contains all the files
3. Command for TCP client to test user authentication:

python letmein-tcp.py 127.0.0.1:8591 username password

Command for UDP client to test user authentication:

python letmein-udp.py 127.0.0.1:8591 username password

The username and password in the above command lines need to replace to my data sample in the code, you can choice any one set of these three:

*Username* *Password*

userone passwordone

usertwo passwordtwo

userthree passwordthree

Exception:

For general cases, input command line with the right username and password will return successful authentication, but when an input incorrect format command, the system will execute the “sys. exit ()” command, and return an exception message: “Invalid input argument, please enter a correct format command.” Then the user have to reenter a new correct format command, there is also command example will be given

.

Description of application protocol:

**TCP:**

1. Setting up the TCP server socket, binding to a particular port for incoming connection requests from clients
2. On the client side, creating a client socket, initiating the TCP connection, connecting to the server IP.
3. On the server side, server socket creates a connection socket to accept client connection.
4. Then client uses a client socket to send the “Authentication Request” to the server socket.
5. On the server side, server socket reads a request from client socket, server socket will determine if that request is “Authentication Request”, then sends back a random 64-bit string to the client socket.
6. Client socket receives that particular random string from server, using a special hashing function such as MD5, hashing the username + password + random string to get a unique hashcode, and send a string contains username + hashcode to the server.
7. The connection socket on the server side, receives the message contains username and hashcode, connection socket uses the username to find the corresponding password in the database. Doing a same hashing function use the username + password + random string to get a new hash code, comparing it with the hashcode received from client. If two hashcode are identical, that means the user information is right, and sent back a successful log in confirmation to client socket, otherwise sent an “authentication failed” message to the client. Connection socket closes.
8. Client socket receives confirmation message from the server, and output the message to the user. Client socket closes.

**UDP:**

1. Setting up a UDP server socket, binding to a particular port
2. On the client side, creating a client socket. Then creates an “Authentication Request” message, send to a particular address and port to the server socket.
3. Server socket receives the request from client socket, and confirm that is an “Authentication Request”, generating a random 64-bit string, send back to the client socket.
4. Client socket receives the random string, and use MD5 to get a unique hashcode (username + password + random string), send a message containing the username and hashcode back to the server socket.
5. Server socket receives the message from client sockets, and uses the username receives from the client to find the corresponding password in the server database, and uses same MD5 hashing function to get a new hashcode (username + password + random string). Comparing the new hashcode with the one from the client side. Then sending back the confirmation result to the client socket.
6. Client socket receives the confirmation result, and notify the users. Client socket closes.

Bugs:

1. Handling the most invalid input argument from user, if the argument is invalid with incorrect format, the system will be halted and tells user to input correct command. There are still some edge cases, and the system will throw the exception error messages.
2. About the message lost situation, I only implement a timeout function in the letmein-udp.py, when the server doesn’t respond to client or the server isn’t open, the client will keep trying resend message to server.
3. If there is a sleep function in the server to delay the sending message to client, a “can not find the substring” error will occur.