COSC310 - Assessment 4: TCP socket server/client v2

By Chris Cody - ID:220209093

TCP socket server/client is a basic TCP socket server and client written in Python 3. The client and server implements my new and improved protocol for COSC310 Assessment 4. The client and server are capable of performing basic key/value store operations over an encrypted TCP socket connection.

Protocol description

For details of the protocol, see the protocol description document (protocol.md).

Security features

For details of the security issues with v1 of this protocol, how they were mitigated and the security features of v2, see the security features document (report.md).

Test user accounts

The following user accounts can be used for testing:

Client id: abc Password: 123

Client id: admin Password: 7(a6fm^YnfPC<$5Y

Installation

Ensure you have the latest version of Python 3 installed on your machine. This should already be installed on turing but if you plan to run the client or server on another machine, Python 3 can be downloaded from https://www.python.org/downloads/ (https://www.python.org/downloads/)

This script requires several core python libraries which should all be installed by default when python is initially installed.

Running the server and client

Using the shell script

Server

1. Open a terminal window and navigate to the directory containing the startserver.sh file.
2. Make sure the shell script is executable by running the following command. This is only needed the first time you run the script.

```
chmod +x startServer.sh
```

3. Run the shell script by using the command

```
./startServer.sh
```

5. The server should now be running and listening for connections on the default port 4242.

Client

1. Open a terminal window and navigate to the directory containing the `startClient.sh` file.
2. Make sure the shell script is executable by running the following command. This is only needed the first time you run the script.

```
chmod +x startClient.sh
```

3. Run the shell script by using the command

```
./startClient.sh
```

4. The client should now be running and attempt to connect to the server on 127.0.0.1:4242.
5. The test client login details can be used for testing: Client id: abc Password: 123

Using the python scripts

Server

1. Open a terminal window and navigate to the directory containing the `socketServer.py` file.
2. Run the command:

```
python socketServer.py
```

3. The server should now be running and listening for connections on the default port 4242.

Client

1. Open a terminal window and navigate to the directory containing the `socketClient.py` file.
2. Run the command:

```
python socketClient.py
```

3. The client should now be running and attempt to connect to the server on 127.0.0.1:4242.
4. The test client login details can be used for testing: Client id: abc Password: 123

Options for the server and client

The examples below show the use of command line options with the shell script, but the python scripts can be used in the same way.

Server

The port to listen for client connections can be specified by passing the port number on startup. For example, to listen on port 1234, run the command:

```
./startServer.sh 1234
```

If a port number is not specified, the server will listen on the default port 4242.

Client

The server address and port can be specified by passing the address and port number on startup. For example, to connect to a server using 192.168.1.1:1234, run the command:

```
./startClient.sh 192.168.1.1 1234
```

If arguments are supplied, the client will attempt to connect to the server on the default address and port 127.0.0.1:4242. If only one argument is supplied, the client will use this as the IP address/hostname and connect to the default port 4242.