COSC310 - Assessment 4: TCP socket server/client v2 Protocol Description

By Chris Cody - ID:220209093

Initiating connection and encryption

All socket communications between the client and server must be encrypted using a TLS/SSL socket.

Protocol

Each message in the protocol consists of a string of ASCII characters followed by the line feed character (ASCII code 10, often represented as \n in programming languages). All messages from the client and server are case sensitive and must be smaller than the RECEIVE_BUFFER_SIZE, which is currently 4096 bits. The only exception is the hashed client password, sent as a fixed number of bits (256) and no training \n.

CONNECT

A client begins by establishing a TLS/SSL connection to the server and sending a `CONNECT` message. This consists of the string `CONNECT`, followed by a client id. The client id which consists of a string of any ASCII characters, terminated by a new line. The next message from the client is the hashed password for the client, sent as 256 bits with no trailing newline. The password is hashed using 100000 iterations of the sha256 algorithm and salted. The salt `b'\xed\x12\x92\xc6\x86\xb4\x8a\xbf\x10\xb3bd\x1c/m\xca'` should be used for testing but change this to a secret value in any production system.

On receiving a `CONNECT` message, a server first checks if there is an existing session using the client id indicated in the `CONNECT` message. If there is, the server responds with a message consisting of `CONNECT: ERROR`, as only one client can `CONNECT` to a session at a time. Otherwise, the server starts a new session for the client, responding `CONNECT: OK` if this is completed successfully, or `CONNECT: ERROR` otherwise.

If a client program receives a `CONNECT: ERROR` message, it should display an appropriate error message and exit.

PUT

Once a session has been established, the client can store data in the session by sending `PUT KEY`, where `KEY` is a string of ASCII characters other than the line feed character. The next message from the client is a string of ASCII characters other than the newline character, which is the data to associate with the key.

On receiving a `PUT` message, the server waits for the associated value to be sent, and then stores it under the given `KEY`. If the given `KEY` already exists in the store, its value is overwritten. Otherwise, the new key and associated value are added to the store. When this is completed successfully, the server responds `PUT: OK`. If this cannot be completed for any reason, the server responds `PUT: ERROR`.

GET

Once a session has been established, the client can request the data associated with a given key by sending `GET KEY`, where `KEY` is a string of ASCII characters other than the line feed character.

On receiving a `GET` message, the server retrieves the value associated with the given `KEY` and calculates the CRC-32 checksum of the value. The server returns a `GET VALUE` message to the client, where `VALUE` is a string of ASCII characters other than the line feed character. The server then immediately sends a second message containing an integer representation of the checksum. If the server is not able to retrieve the key or calculate the checksum for any reason (e.g., the given `KEY` does not exist in the store), the server responds `GET: ERROR` instead.

DELETE

Once a session has been established, the client can request to delete data associated with a given key by sending `DELETE KEY`, where `KEY` is a string of ASCII characters other than the line feed character.

On receiving a `DELETE` message, the server deletes the `KEY` and associated value from its store, if it exists, and responds `DELETE: OK`. If, for any reason, the key and associated value remain in the store, the server must respond `DELETE: ERROR`.

DISCONNECT

Once a session with a server has been established, the client can destroy the session by sending `DISCONNECT`.

The server removes all traces of the session from its system and responds `DISCONNECT: OK`.

Other messages

Other than the initial sharing of public keys and the above listed commands, any other message sent to or from the client is considered an error, and should result in the receiving party dropping the connection. In such a case, the server should remove all data associated with the client from its system.