# CS305 Assignment 1

## Introduction

Telnet is a network protocol primarily used to establish text-based, bidirectional communication between a local computer and a remote server. It operates over a TCP/IP connection, allowing users to remotely control servers via command-line interface. Telnet is often used to manage devices, servers, or network equipment, but due to its lack of encryption, it poses security risks and has largely been replaced by encrypted protocols like SSH.

In this assignment, we need to implement a Telnet service with simplified NTLM authentication. You are required to set up both the server and client, enabling basic functions such as registration, login, and calculation.

## Backgrounds

In this project, our system is mainly composed of two parts: Telnet service and NTLM authentication.

### Telnet Service

The Telnet service consists of two components: the server and the client. When the server starts the Telnet service, it listens for requests on a designated port. When a client wants to access the server's Telnet service, it first sends a request to the server's listening port to establish a TCP connection. Once the connection is established, the client sends its login information (username and password) to the server for authentication. After the authentication is successful, the server begins to provide services to the client. At this point, the client can send commands to the server for execution, and the server will return the results.

## NTLM Authentication

The NTLM authentication process is carried out through a three-way handshake. First, the client initiates an authentication request, and the server generates a random challenge message and sends it to the client. After receiving the challenge, the client encrypts the challenge message using the hashed value of the user's password, generating a response message, which is then sent back to the server. The server uses the stored hash of the user's password to encrypt the same challenge, and then compares the result with the client's response. If they match, the authentication is successful, and the client is granted access; if they don't match, the authentication fails. This mechanism prevents the transmission of plaintext passwords over the network, enhancing security.

However, in this project, we do not need to implement the full NTLM authentication, only a simplified version of it. The specific details of the requirements will be explained in the **Tasks** section below.

# System Design

---

This project consists of two main parts: the server and the client. The server continuously listens on a specified port for TCP connection requests from the client. Once a request is received, the server establishes a TCP connection with the client. After the connection is established, the client is in an unlogged-in state by default, at which point the user can perform registration or login operations. Upon successful login, the user can perform simple calculations via the server.

## Client

The client mainly needs to implement the following functionalities:

1. Establish a TCP connection with the server.
2. Send commands entered from the console to the server in plaintext.
3. Receive responses from the server and display them on the console.
4. Encrypt passwords before sending any password-related messages to the server.

## Server

The server mainly needs to implement the following functionalities:

1. Listen on a port to establish TCP connections.

2. Receive and process commands sent by the client. Before processing, the server needs to check the type of command and determine if the command format is valid. For valid commands, the server processes and returns the result; for invalid commands, it returns an error message.
3. Implement user registration management to perform login authentication.

In this system, the client only needs to transmit the commands entered from the console to the server in plaintext, and the server is responsible for validating the legality and type of the commands. However, if you plan to implement a simple NTLM authentication, you need to encrypt the password when sending any password-related information from the client.

## Tasks

---

### Task 1: Connection Establishment (20 points)

In this task, you need to establish a TCP connection between the client and the server. This task can be divided into two parts:

1. The client should be able to send a request to a specified IP address and port in order to establish a TCP connection. (5 points)
2. The server needs to listen for connection requests on a specified port and respond to the client's connection request to establish a TCP connection. A server can simultaneously hold sessions with multiple clients.(10 points)
3. The server should be able to receive user message (saved to variable `receive_data`) for commands processing, the user command records should be saved into a txt file. Connection close detection should also be realized here (if connection is closed, the `main_loop` function will return `False, None`). (5 points)

### Task 2: User Authentication (25 points)

In this task, you need to implement the user authentication module for the server, which mainly includes the following aspects:

1. The server should be able to load the existing registration information stored locally upon startup. (5 points)
2. The client should be able to register new users on the server. After successful registration, the server will synchronize the newly registered user to the local records. The format for the client registration command is: `register $(username) $(password)`. (10 points)

3. The client can log in to the server using an already registered user by entering the username and password. The format for the client login command is: `login $(username) $(password)`. After the server receives a login request from the client, it will compare the provided credentials with all the registered information. If a matching username and password are found, the login is successful. (10 points)

## Task 3: NTLM Authentication (25 points)

In this task, you need to implement a simplified NTLM authentication, which mainly involves the following steps:

1. When the client sends the user's password to login, the password must be encrypted. MD5 encryption is used for this, and the `ntlm_hash_func` function in `function.py` implements this encryption. During login, the username is still sent in plaintext, but the password needs to be encrypted with MD5 before being sent. Similarly, any password-related information (such as the password during registration or when changing the password) is no longer transmitted in plaintext but as an MD5-encrypted string. Therefore, the user passwords stored on the server are no longer plaintext but are the MD5-encrypted passwords. You need to implement the `server_message_encrypt` function in `function.py`. (5 points)
2. After the server receives the login information from the client, it first compares the registration information (which was already implemented in the previous task). If the comparison is successful, the server will send back a challenge message to the client. The challenge message consists of 8 random bytes, and they do not need to be encrypted during transmission. You need to implement the `generate_challenge` function in `function.py`. (5 points)
3. Before completing the tasks below, you need to first implement an HMAC-SHA256 encryption function. It is implemented by `calculate_response` function in `function.py`. This step can be implemented using the `hmac.new()` function from the `hmac` library. The usage is as follows: `hmac.new(key, msg=None, digestmod=None)`, where the parameter `key` is the encryption key (type: `bytes`), which is the MD5-encrypted password; the parameter `msg` is the message to be encrypted (type: `bytes`), which is the challenge; and the parameter `digestmod` is the encryption method, here using `hashlib.sha256`. (5 points)
4. After receiving the challenge from the server, the client will use HMAC-SHA256 to encrypt its MD5-encrypted password together with the challenge (You need to use the `calculate_response` function in `function.py`), and then send the result back to the server. You need to implement the `server_response` function in `function.py`. (5 points)
5. After the server receives the client's challenge response, it will perform the same encryption using the MD5-encrypted password stored for that user and the challenge (You need to use the `calculate_response` function in `function.py`). It will compare the results of both encryptions, and if they match, the authentication is considered successful, and the user is logged in. (5 points)

## Task 4: Command Processing (25 points)

In this task, you need to implement command processing for users after logging in through Telnet. We need to handle the following commands:

1. **Addition**: The format is `sum $(number1) $(number2) ...`. You can pass more than two numbers for cumulative addition. (4 points)
2. **Multiplication**: The format is `multiply $(number1) $(number2) ...`. You can pass more than two numbers for cumulative multiplication. (4 points)
3. **Subtraction**: The format is `subtract $(number1) $(number2)`. Only two numbers can be passed for subtraction. (3 points)
4. **Division**: The format is `divide $(number1) $(number2)`. Only two numbers can be passed for division. (3 points)
5. **Change password**: The format is `changepwd $(new_password)`. (3 points)
6. **Help**: When entering "?" or "help", display help information for all commands. (3 points)
7. **Disconnect**: When entering "exit", log out and disconnect. When the connection is disconnected, the client program will stop running as well.(3 points)
8. **Logout**: When entering "logout", log out of the session. After logging out, the connection between you and the server is not disconnected. You can still log in to the server again as a different user and perform computations.(2 points)

You need to implement the `login_cmds` function in `function.py`.

## Task 5: Exception Handling (5 points)

In this task, you do not need to implement any new features, but you need to improve the code from the previous tasks to handle exceptions caused by incorrect input or connection interruptions. These exceptions include but are not limited to:

1. Accessing an invalid IP or port.
2. Attempting to register a user that is already registered.
3. Attempting to log in again after already being logged in.
4. Too many or too few command parameters, or the command does not exist.
5. Extra spaces following a command.

# Submission

When submitting, you need to submit a report along with all the code, packaged into a .zip archive named `sid_Telnet.zip`, where `sid` is your student ID.

## Report

In the report, you need to present the following content:

1. Screenshots of the command line from both the client and server after successful execution. These results should cover the tasks you have completed to demonstrate your work.
2. Communication packets captured between the client and server using Wireshark. If both your server and client are running locally, you can capture the packets through the "Adapter for loopback traffic capture" interface.
3. If you added additional functions or modified the structure and parameters of existing functions, you need to provide a detailed explanation of the implementation and usage.

## Code

Submit all three files: `client.py`, `server.py`, and `functions.py`. If you added any additional code files, please include them as well and provide an explanation in the report.

# Notice

---

1. This assignment will be graded using scripts, so **please try not to modify the parameters and return values of the existing functions in the template code**. If you must modify the structure of the functions in the template code, **please be sure to explain in detail in your report what changes you made and how you modified it**. Also, **explain how the modified functions should be called**. Failure to do so may result in serious grading errors.
2. To facilitate script-based grading, **please make sure to submit and name your files according to the specified format**. Failure to do so may result in serious grading errors.
3. The report does not count toward your grade, but if possible, please submit it. If you made code modifications as mentioned in point 1, be sure to provide a detailed report to assist with grading.