

Protocol Specification

Transport Layer

- **Protocol:** TCP
- **Default Port:** 1337
- **Encoding:** UTF-8
- **Message Format:** Newline-terminated (`\n`) text messages

Protocol Flow

1. Connection Phase

- Client connects to server
- Server sends: "Welcome! Please log in.\n"

2. Authentication Phase

Step 1: Username

- User types: "User: <username>\n"
- Server validates format
- If invalid: server sends "error: invalid input\n" and closes connection
- If valid: server stores username and expects password

Step 2: Password

- User types: "Password: <password>\n"
- Server validates format and checks credentials
- If invalid format: server sends "error: invalid input\n" and closes connection
- If wrong credentials: server sends "Failed to login\n" and resets to username step
- If correct: server sends "Hi <username>, good to see you.\n" and proceeds to command phase

3. Command Phase

After successful login, user can send these following commands:

a) Parentheses Check

- User types: "parentheses: <expression>\n"
- Expression must contain only (and) characters
- Server response: "the parentheses are balanced: yes\n" or "the parentheses are balanced: no\n"
- Error: "error: invalid input\n" (if expression contains invalid characters)

b) LCM (Least Common Multiple)

- User types: "lcm: <num1> <num2>\n"
- Server response: "the lcm is: <result>\n"
- Error: "error: invalid input\n" (if not exactly 2 integers)

c) Caesar Cipher

- User types: "caesar: <text> <shift>\n"
- Text must contain only letters (A-Z, a-z) and spaces
- Server response: "the ciphertext is: <encrypted_text>\n"
- Error: "error: invalid input\n" (if invalid characters or format)

d) Quit

- User types: "quit\n"
- Server closes connection (no response)

e) Invalid Command

- Any other command format
- Server response: "error: invalid input\n"
- Connection remains open

Welcome! Please log in.

User: alice

Password: secret123

Hi alice, good to see you.

parentheses: (()())

the parentheses are balanced: yes

lcm: 12 18

Server: the lcm is: 36

caesar: hello world 3

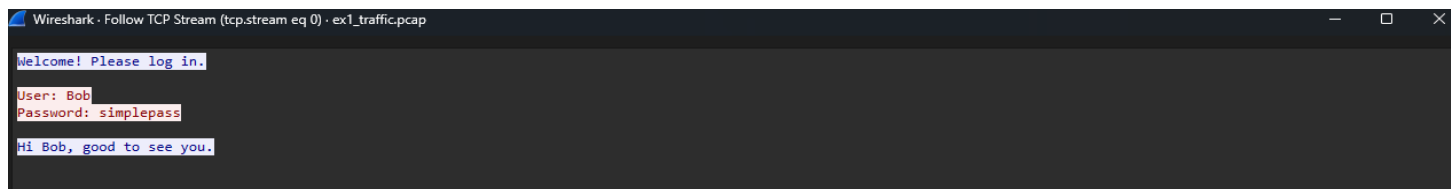
the ciphertext is: koor zruog

quit

[Connection closed]

Blue represents Server's output

Yellow represents Client's input.



In generally: The screenshot shows the application protocol flow. The server sends a welcome message (Blue), the client sends the username 'Bob' (Red), and the server validates it

More specifically:

	Info	Length	Protocol	Destination	Source	Time	.No
Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM [SYN]	1337 → 64985 56		TCP	127.0.0.1	127.0.0.1	9.892546	3
Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM [SYN, ACK]	64985 → 1337 56		TCP	127.0.0.1	127.0.0.1	9.892611	4
Seq=1 Ack=1 Win=65280 Len=0 [ACK]	1337 → 64985 44		TCP	127.0.0.1	127.0.0.1	9.892641	5
Seq=1 Ack=1 Win=65280 Len=24 [PSH, ACK]	64985 → 1337 68		TCP	127.0.0.1	127.0.0.1	9.892816	6
Seq=1 Ack=25 Win=65280 Len=0 [ACK]	1337 → 64985 44		TCP	127.0.0.1	127.0.0.1	9.892844	7
Seq=1 Ack=25 Win=65280 Len=31 [PSH, ACK]	1337 → 64985 75		TCP	127.0.0.1	127.0.0.1	36.184195	16
Seq=25 Ack=32 Win=65280 Len=0 [ACK]	64985 → 1337 44		TCP	127.0.0.1	127.0.0.1	36.184231	17
Seq=25 Ack=32 Win=65280 Len=25 [PSH, ACK]	64985 → 1337 69		TCP	127.0.0.1	127.0.0.1	36.184337	18
Seq=32 Ack=50 Win=65280 Len=0 [ACK]	1337 → 64985 44		TCP	127.0.0.1	127.0.0.1	36.184363	19

Lines 1-3- the packets are SYN, SYN-ACK, ACK and thus, these three packets show the TCP 3-Way Handshake establishing the connection between the client and server.

Line 4 (PSH, ACK packet)- means the server pushes (PSH) data to the client (the 'Welcome' message)

Line 5 (ACK packet): The client acknowledges receiving the message.

Line 6 (PSH, ACK packet)- the client pushes (PSH) data to the server (his username and password)

Line 7 (ACK packet): The server acknowledges receiving the message.

Line 8 (PSH, ACK packet)- the server sends the "... good to see you" message.

Line 9 (ACK packet): The client acknowledges receiving the message.