Second question :

1.

Threat	Weak input validation on client username fields
Affected component	Client registration and message sending input (handle_request function)
Module details	client.cpp (lines around the handle_request and create_registration_packet functions)
Vulnerability class	Input validation bypass / injection potential
Description	The program accepts user input for usernames and recipient names without validation or length checking. The username is sent directly to the server as part of the binary protocol. Long or invalid characters (such as non-printable characters or oversized strings) can cause protocol corruption or overflow issues on the server side. An attacker could also send binary input through this mechanism.
Result	May allow denial of service, corrupted server state, or unexpected behavior, potentially leading to server crashes or vulnerability chaining.
Prerequisites	The client allows free text input for usernames or recipient fields without checks. The attacker has access to the client input field.
Business impact	Could lead to server instability, crashes, or protocol parsing errors. This could influence communication services or be exploited as part of a larger attack.
Proposed remediation	Validate input length on the client side before sending to the server (limit usernames to 16 ASCII characters). Block non-ASCII characters. Do not allow empty strings or overly long usernames. On the server side, reject malformed input.
Risk	Damage potential:7 Reproducibility: 10 Exploitability:7 Affected users: 7 Discoverability: 9 Overall: 7.8

2.

Threat	Local information leakage through unprotected my.info file
Affected component	Local file handling in the client (handle_response when saving user data)
Module details	client.cpp (lines around writing my.info)
Vulnerability class	Local file exposure / unencrypted sensitive data
Description	The client writes the username and client ID in plain text to a local file called my.info, without encryption or any file protection. This file is stored in a predictable location and can be read by any user or malware on the same system. It contains sensitive identifiers that could be used to impersonate the user or launch further attacks.
Result	Any local attacker or malicious process on the same machine can read my.info and obtain client IDs and usernames for impersonation or attacks on the server.
Prerequisites	The attacker has access to the client's filesystem (even without elevated permissions).
Business impact	User impersonation, unauthorized actions on the server, and potential compromise of user accounts.
Proposed remediation	Store my.info in a protected location (secure storage or with restricted permissions). Consider encrypting client ID data or using obfuscation.
Risk	Damage potential: 6 Reproducibility: 10 Exploitability: 7 Affected users: 7 Discoverability: 9 Overall: 7.8

3.

Threat	No verification of server identity (no server authentication)
Affected component	Client connection logic in connect_to_server()
Module details	network.cpp (function connect_to_server())
Vulnerability class	Lack of server authentication / Man-in-the-middle risk
Description	The client connects to any IP and port provided in server.info without verifying the server's identity. There is no verification that the server is trusted. An attacker on the same network or who takes over DNS could impersonate the server and receive client registration details, messages or other sensitive information.
Result	The client may send sensitive information to an attacker instead of the server.
Prerequisites	The attacker needs to trick the system to take control of connection between client and server (faking DNS for example)
Business impact	Identity theft, user data exploit, message interception, and unauthorized control over client-server communication.
Proposed remediation	Add SSL/TLS support, verify server certificates, and only connect to trusted servers.
Risk	Damage potential: 8 Reproducibility: 9 Exploitability: 7 Affected users: 7 Discoverability: 8 Overall: 7.8

4.

Threat	No authentication on request 604 – pulling messages
Affected component	handle_client() and process_request() functions for request code 604
Module details	message_handler.py (request 604 handling)
Vulnerability class	Missing authentication check
Description	When a client sends request code 604 the server only decodes the client ID from the header and retrieves messages. It does not verify if the client ID belongs to an authenticated or registered user. Any client could send a forged client ID to receive messages that don’t belong to them.
Result	Unauthenticated access to private messages for other users.
Prerequisites	Attacker knows or can guess another user's client ID (UUID).
Business impact	Confidential message leakage, severe privacy violation, and potential impersonation.
Proposed remediation	Before serving request 604, check if the client ID exists in user_storage using user_storage.get_user_by_id(). Only serve messages to verified clients.
Risk	Damage potential: 8 Reproducibility: 9 Exploitability: 7 Affected users: 7 Discoverability: 8 Overall: 7.8

5.

Threat	Fixed-size recv(1024) leads to potential data truncation or partial reads
Affected component	handle_client() function
Module details	message_handler.py (line where data = conn.recv(1024))
Vulnerability class	Incomplete or partial packet processing
Description	The server reads incoming client data with a single conn.recv(1024) call. TCP does not guarantee that the entire request will arrive in one chunk or be smaller than 1024 bytes. If the client sends a packet larger than 1024 bytes (or if TCP splits it), the server will only process part of the data, causing protocol errors or corrupted payload parsing.
Result	Incorrect or incomplete request handling, protocol decoding errors, and possibly server crashes or unintended behavior.
Prerequisites	The client sends a large payload (like a long username or large message) that exceeds 1024 bytes or arrives in multiple TCP packets.
Business impact	Potential denial of service or unreliable handling of legitimate client requests.
Proposed remediation	Use a loop to receive data until the full payload is read, based on the length from the decoded header.
Risk	Damage potential: 5 Reproducibility: 9 Exploitability: 6 Affected users: 6 Discoverability: 9 Overall: 7.0