Threat	Fixed IV in AES-CBC Encryption
Affected component	MessageU encryption module
Module details	AESWrapper.cpp (lines 46-54)
Vulnerability class	Cryptographic implementation weakness
Description	The application uses a fixed all-zero Initialization Vector (IV) for AES-CBC encryption. The IV is hard coded as a zero-filled array in the encryption and decryption functions. Using a fixed IV makes the encryption deterministic, meaning identical plaintext messages will always produce identical ciphertext.
Result	Loss of message confidentiality. An attacker monitoring encrypted communications can perform pattern analysis to identify when identical messages are sent. This enables chosen-plaintext attacks and ciphertext analysis that could lead to message content disclosure.
Prerequisites	An attacker would need to capture encrypted messages between users.
Business impact	could result in disclosure of sensitive communications.
Proposed remediation	Generate a cryptographically secure random IV for each encryption operation. The IV should be prepended to the ciphertext and sent along with it (IVs don't need to be kept secret, only random and unique per message). Modify the decryption function to extract and use the IV from the beginning of the received ciphertext.
Risk	Damage potential: 8 Reproducibility: 10 Exploitability: 6 Affected users: 10 Discoverability: 7 Overall: 8.2

Threat	Client ID spoofing and session impersonation
Affected component	MessageU authentication system
Module details	server.py (handle_client method, lines 60-123)
Vulnerability class	Authentication bypass
Description	The protocol relies solely on a Client ID (UUID) provided by the client in each request header without any verification that the sender is the legitimate owner of that ID. After registration, any request can be made by simply including a known Client ID in the request header. The server performs no validation to ensure the client is authorized to use that ID, such as digital signatures or challenge-response mechanisms.
Result	An attacker who obtains a valid Client ID can fully impersonate that user, access their messages, and send messages as them. This compromises the entire security model of the system.
Prerequisites	An attacker would need to discover a valid Client ID, which could be obtained through network sniffing, as the ID is transmitted in every request.
Business impact	Complete compromise of user accounts and their private communications. This undermines the confidentiality and integrity of all messages in the system.
Proposed remediation	Implement a proper authentication system using digital signatures where each request is signed with the client's private key. Implement a challenge-response mechanism to verify client identity. Add session tokens with proper validation and limited lifetimes. Consider adding TLS for transport layer security to prevent ID sniffing.
Risk	Damage potential: 10 Reproducibility: 10 Exploitability: 7 Affected users: 10 Discoverability: 6 Overall: 8.6

Threat	Message replay attacks
Affected component	MessageU message handling system
Module details	server.py (handle_send_message method, lines 172-198)
Vulnerability class	Message integrity and authenticity
Description	The protocol lacks any mechanism to prevent replay attacks. There are no timestamps, nonces (numbers used once), or sequence numbers included in message payloads. The server accepts and processes any correctly formatted message without verifying whether it has been seen before.
Result	An attacker can capture legitimate encrypted messages and resend them multiple times, causing the recipient to receive the same message repeatedly with no way to identify duplicates. This could have serious consequences in contexts where message freshness is critical (e.g., "Yes, transfer the money").
Prerequisites	An attacker would need to be able to capture network traffic and replay messages, which could be done through various network interception techniques.
Business impact	Loss of message integrity and potential for malicious actions through message replay, undermining the reliability and trustworthiness of the communication system.
Proposed remediation	Implement unique message identifiers, timestamps, or incrementing sequence numbers that are included in the encrypted message payload. Add a nonce system where each message contains a unique random value that can only be used once. Keep track of previously received message IDs on the client side to detect and reject duplicates. Add message expiration mechanisms to prevent old messages from being accepted after a certain time period.
Risk	Damage potential: 7 Reproducibility: 10 Exploitability: 8 Affected users: 10 Discoverability: 6 Overall: 8.2

Threat	Insufficient RSA key length
Affected component	MessageU cryptographic implementation
Module details	RSAWrapper.h (line 13) and RSAWrapper.cpp
Vulnerability class	Cryptographic strength weakness
Description	The application uses RSA keys with a length of only 1024 bits as defined in the RSAWrapper.h file. This key length is below current security standards and recommendations. Current best practices recommend a minimum of 2048 bits for RSA keys to provide adequate security margins against advances in computational power and cryptanalytic techniques.
Result	RSA keys with 1024 bits are vulnerable to factorization attacks with sufficient computational resources. If the private key is compromised through factorization, an attacker could decrypt all messages encrypted with the corresponding public key, completely breaking the confidentiality of communications.
Prerequisites	An attacker would need significant computational resources to factor a 1024-bit RSA key. While not trivial, this is increasingly within reach of well-resourced adversaries.
Business impact	Using cryptography that falls below current security standards undermines the long-term security of all communications in the system. Messages captured today could be decrypted in the future when factorization of 1024-bit keys becomes more feasible.
Proposed remediation	Increase the RSA key length to at least 2048 bits. ensuring the application can properly handle the larger key sizes throughout its workflow.
Risk	Damage potential: 8 Reproducibility: 5 Exploitability: 3 Affected users: 10 Discoverability: 9 Overall: 7.0