COSC540 - COMPUTER NETWORKS AND NETWORK SECURITY

Assessment – 4 Report

BIKASH NEUPANE
220245756
bneupan2@myune.edu.au

Table of Contents

Introduction 1

Identified Issues 1

1. Lack of Encryption 1
2. No User Verification 1
3. No Message Integrity Checks 1
4. Insecure User Credentials Storage 1
5. No Confidentiality and Authenticity 1
6. Lack of Read Receipts 1
7. No Secure Socket Layer or Transport Layer Security (SSL/TLS) 1

Mitigation Approaches 1

- Encryption of Communication (SSL/TLS) 1
- User Verification 2
- Message Integrity Checks 2
- Secure User Credentials Storage 2
- Confidentiality and Authenticity 2
- Read Receipts 2

Implementation Details 3

- Server Side (server.py) 3
- Client Side (client.py) 3

Unimplemented Mitigations 4

- Two-Factor Authentication (2FA) 4
- End-to-End Encryption (E2EE) 4
- Advanced Key Management 4
- User Online Status 4
- GUI Version for the client 4
- Going on adding more features 4

Conclusion 4

Secure Communication System

Introduction

This report describes the security, privacy, and trust issues identified in the initial version of the secure communication system. It also details the approaches taken to mitigate these issues, the implementation of these mitigations, and the limitations of the current solutions.

Identified Issues

The following security, privacy, and trust issues were identified in the initial version of the project:

1. **Lack of Encryption**: Communication between clients and the server was transmitted in plain text, making it vulnerable to eavesdropping.
2. **No User Verification**: There was no mechanism to verify the identity of users, leading to potential impersonation attacks.
3. **No Message Integrity Checks**: Messages could be altered in transit without detection.
4. **Insecure User Credentials Storage**: Username were stored in plaintext, making them vulnerable.
5. **No Confidentiality and Authenticity**: Messages could be intercepted and altered, and there was no guarantee that messages were sent by legitimate users.
6. **Lack of Read Receipts**: Users had no way of knowing if their messages were read by the recipients.
7. **No Secure Socket Layer or Transport Layer Security (SSL/TLS)**: Data transmission between clients and the server was not encrypted, making it vulnerable to interception and attacks.

Mitigation Approaches

To address the identified issues, the following mitigation strategies were implemented:

Encryption of Communication (SSL/TLS)

Issue: Lack of encryption made communication vulnerable to eavesdropping.
Solution: Implemented SSL/TLS to encrypt communication between clients and the server.
Implementation: Used Python's “SSL” module to create a secure socket layer for both the server and client.
Limitation: Requires valid SSL certificates and proper configuration.

User Verification

Issue: No user verification led to potential impersonation attacks.
Solution: Introduced a login mechanism with hashed passwords.
Implementation: Users must log in with a username and password. Passwords are hashed using SHA-256 before storage.
Limitation: Relies on the security of the hashing algorithm and the database.

Message Integrity Checks

Issue: Messages could be altered without detection.
Solution: Implemented hash-based message authentication codes (HMACs) to ensure message integrity.
Implementation: Each message includes an HMAC computed using a shared secret key.
Limitation: Requires secure management of the shared secret key.

Secure User Credentials Storage

Issue: Storing username or passwords in plaintext is insecure.
Solution: Stored passwords as hashes using a strong cryptographic hash function (SHA-256).
Implementation: Passwords are hashed before being stored in the database.
Limitation: Does not protect against brute force attacks if the database is compromised.

Confidentiality and Authenticity

Issue: Messages could be intercepted and altered, with no guarantee of authenticity.
Solution: Used HMACs to ensure message authenticity and SSL/TLS for confidentiality.
Implementation: Combined SSL/TLS encryption with HMACs for message integrity.
Limitation: Requires secure management of keys and proper configuration.

Read Receipts

Issue: Users had no way of knowing if their messages were read.
Solution: Implemented read receipts.
Implementation: The server updates the message status when a message is read and notifies the sender.
Limitation: Adds additional complexity to message handling.

Implementation Details

Server Side (server.py)

SSL/TLS Configuration:

- Created a secure context using ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
- Loaded the server's certificate and key using context.load_cert_chain(certfile="server.crt", keyfile="server.key")
- Wrapped the server socket with SSL using context.wrap_socket

User Verification and Password Hashing:

- Added SQLite database to store user credentials securely.
- Used SHA-256 to hash passwords before storing them in the database.

Message Handling:

- Implemented functions to handle user login, message sending, and reading messages.
- Added SQLite database to store messages securely.
- Used JSON to format responses for easy parsing by the client.

Client Side (client.py)

SSL/TLS Configuration:

- Created a secure context using ssl.create_default_context()
- Loaded the server's certificate using context.load_verify_locations('server.crt')
- Wrapped the client socket with SSL using context.wrap_socket

User Login and Password Hashing:

- Collected user credentials securely using *getpass*.*getpass* to avoid displaying passwords.
- Hashed passwords using SHA-256 before sending them to the server.

Message Handling:

- Implemented functions to compose messages, read messages, and view notifications.
- Used JSON to parse server responses for easy handling of structured data.

Unimplemented Mitigations

Two-Factor Authentication (2FA)

Reason: Due to time constraints and increased complexity, 2FA was not implemented.
Future Work: Plan to integrate 2FA using email or SMS for enhanced security.

End-to-End Encryption (E2EE)

Reason: Requires significant architectural changes and key management.
Future Work: Explore integrating E2EE to ensure messages are encrypted from sender to receiver.

Advanced Key Management

Reason: Current implementation uses basic key management for HMACs.
Future Work: Implement a more robust key management system, possibly using public key infrastructure (PKI).

User Online Status

Reason: Due to time constraints and increased complexity, tracking of users' online status is now currently done in server side only.
Future Work: Plan to implement features in client side to keep the track of users' online status.

GUI Version for the client

Reason: Due to time constraints and increased complexity, the client side still use CLI version.
Future Work: Plan to implement GUI version for the client to make it easier to use.

Going on adding more features

Reason: There may be more features to implement in GUI version to make it more efficient and robust.
Future Work: Plan to implement more features for secure communication system.

Conclusion

The updated secure communication system addresses many of the initial security, privacy, and trust issues. Implementing SSL/TLS, user verification, secure password storage, and message integrity checks significantly enhances the system's security. However, there are still areas for improvement, which are planned for future iterations. This report highlights the importance of ongoing security assessments and iterative improvements in developing secure systems.