Peer Review Report for Group 31 – Leo Nguyen

This peer review is dedicated to:

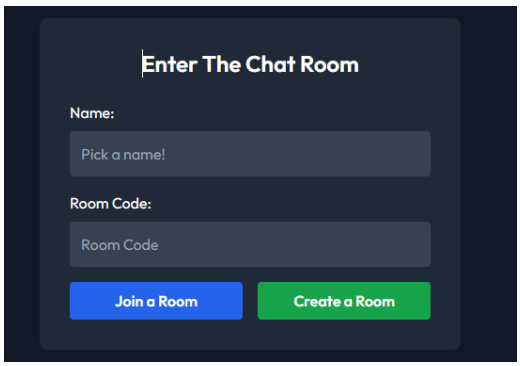
- Elliot Koh
- Bryan Van
- Leo Nguyen
- Nathaniel Cordero

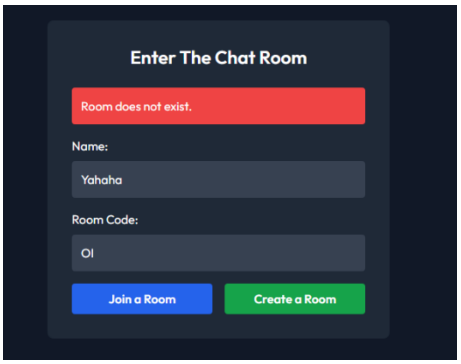
This review will attempt to create and join chat rooms, send public and private messages, and experiment with the implementation of features such as file transfer. The focus of this review is to identify deliberate backdoors and security vulnerabilities, while providing constructive feedback on the strengths of the system and areas for improvement.

Dynamic Analysis

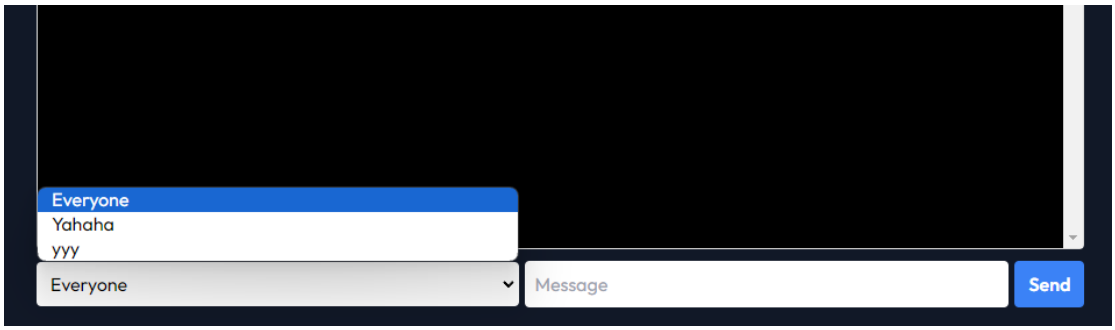
Functional Testing

In dynamic analysis, I ran the main.py script, and the application successfully started, as shown in the screenshots provided. Key features were tested, including joining rooms and sending both public and private (whisper) messages.

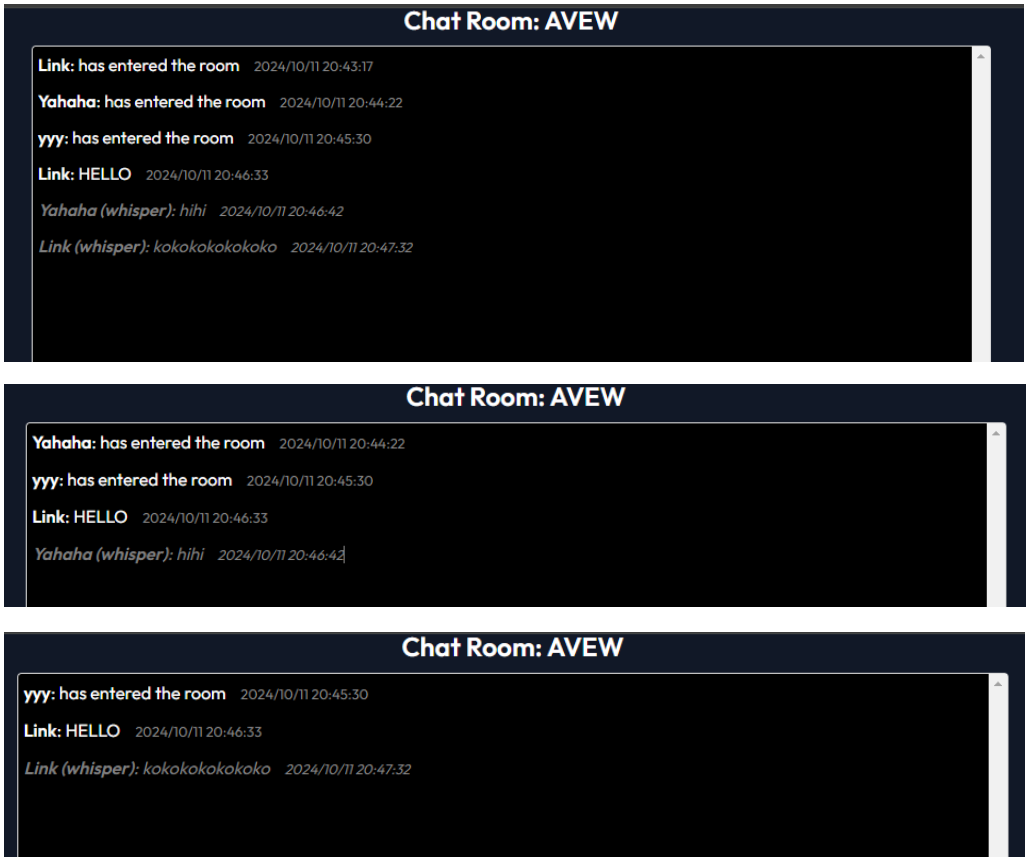




- ✧ Room Entry: Multiple users successfully joined the chat room, and messages were exchanged between users.
- ✧ Public and Whisper Messaging: Both public and private messaging functionalities worked as expected:



Below are the communication between 3 users in the same chat room:



Missing Functionalities

1. **File Transfer**:
    - While the chat functionality is solid, the application currently does not include a file-sharing feature. Adding file upload and download capabilities would improve collaboration among users, especially in scenarios where exchanging files is necessary.
2. **Message Encryption**:
    - No encryption mechanisms are implemented in the current version, leaving public and private messages unprotected. This exposes the system to risks such as man-in-the-middle attacks, where unauthorized users can intercept and read messages.
3. **Heartbeat Mechanism**:
    - The application lacks a heartbeat mechanism to check the connection status of clients. This can lead to scenarios where disconnected clients still appear connected, occupying server resources unnecessarily.

Static Code Analysis

The static analysis involved a thorough review of the code, focusing on identifying backdoors and other security vulnerabilities. While the implementation is strong in several areas, there are critical backdoors and flaws that require immediate attention.

Strengths

1. **Clean Code Structure**:
    - The separation between HTML templates (base.html, home.html, and room.html) and the backend logic (main.py) demonstrates a well-thought-out design, making the code easy to maintain and scale.
2. **User Interface Design**:
    - The chat interface is intuitive, with easy navigation for users to join rooms and exchange messages. The responsive design enhances the overall user experience, making it accessible on different devices.

Vulnerabilities and Backdoors

After reviewing the provided codebase, I did not find any deliberate backdoors that were intentionally planted. However, I did identify two critical areas where security flaws exist that could be exploited, which might unintentionally function as backdoors.
Backdoor 1 - Client-Side Authentication.

Backdoor 1 - Client-Side Authentication:

- The authentication process is handled on the client side using session data, which can be easily manipulated by anyone with access to the browser's developer tools. This presents a critical vulnerability, as users can modify session variables such as name and room, gaining unauthorized access to rooms or potentially sensitive areas of the system.

```
61  @app.route("/room")
62  def room():
63      room = session.get("room")
64      name = session.get("name")
65      if room is None or name is None:
66          return redirect(url_for("home"))
```

This approach relies on the session variables stored on the client side, which are not secure. It is recommended to handle all authentication and room access control on the server side, ensuring that session data is validated before granting access.

Backdoor 2 - Incomplete RSA Encryption Implementation:

- The code lacks input validation for critical fields such as usernames and chat messages. Without proper sanitization, this exposes the system to injection attacks, such as Cross-Site Scripting (XSS). Adding validation for all user inputs will reduce the risk of such vulnerabilities.

Additional Vulnerabilities

1. **Replay Attack Prevention**:
    - The system does not currently have any counter or timestamp mechanisms to prevent replay attacks. An attacker could capture and replay old messages, making the system vulnerable to such attacks. Implementing a message counter or

timestamp validation would help mitigate this risk.

2. **Insufficient Error Handling**:
    - Although basic error handling is present, it is insufficient for critical sections like cryptographic operations and message transmission. A more robust error-handling mechanism would make the system more resilient to failures.

Strengths

1. **Clean Code Structure**:
    - The code is well-organized, with a clear separation between the backend logic (main.py) and the HTML templates (base.html, home.html, and room.html). This makes the project easy to follow and maintain.
2. **User Interface Design**:
    - The chat interface is simple, intuitive, and responsive. Users can easily navigate between different rooms, and the design is visually appealing, enhancing the overall user experience.
3. **Smooth Functionality**:
    - The core chat functionality, including room joining and messaging (both public and private), works smoothly without issues. The system is responsive, and the chat interface performs well.

Areas for Improvement

The chat application is functional and intuitive, there are several areas where security can be improved, and additional features could be implemented.

1. **Add File Transfer**:
    - Introducing file-sharing functionality would make the application more versatile and practical for users who need to exchange files within a chat session.
2. **Implement End-to-End Encryption**:
    - To enhance security, I recommend implementing RSA for key exchange and AES for encrypting messages. This will ensure that messages are encrypted and secure during transmission, protecting users' privacy.
3. **Introduce Heartbeat Mechanism**:
    - A heartbeat mechanism would allow the server to detect and disconnect inactive clients, freeing up resources and ensuring that only active users remain connected.
4. **Move Authentication to the Server Side**:
    - Shifting the authentication process to the server will prevent users from manipulating session data in their browsers, ensuring that only authorized users can access certain rooms or functions.
5. **Enhance Input Validation**:
    - Adding input validation and sanitization for all user inputs will help prevent injection attacks such as XSS, protecting the integrity of the chat system.

Thank you for working hard on the chat application! If you would like further adjustments or additions, feel free to ask!