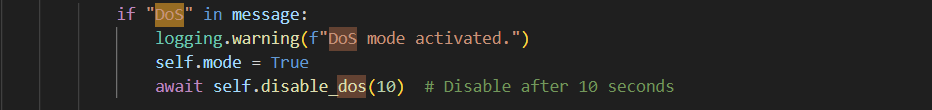
**vulnerability**

1. **DoS (Denial of Service) Vulnerability**



**Issue**:

I think in this case, any user can easily trigger a server's DoS mode by sending a message containing a DoS, causing the server to stop functioning normally for that period of time. So, this may be a backdoor that would allow a malicious user to easily attack the server.

**Fix Suggestion:**

This feature should be removed, or at the very least should be permission controlled. Only authenticated administrator users should be able to activate or deactivate this mode.

文本

中度可信度描述已自动生成

1. **Signature verification logic issues**

手机屏幕截图

描述已自动生成

**Issue:**

The signature verification logic should ensure that the signature of any message is valid. Then, if the signature verification fails, then the message should not be processed. Additionally, JSON data may be formatted and serialised slightly differently in different environments, which may cause signature verification to fail.

**Fix Recommendation:**

Ensure that signature verification is strictly enforced in all message processing, especially in the handle\_signed\_data function.

文本

描述已自动生成

1. **Problems in the file upload function**

文本

描述已自动生成

**Issue:**

The file upload feature may allow users to overwrite files on the server via a file path traversal attack. This is due to the fact that uploaded filenames are saved directly to the server directory, which may still be a risk despite some handling.

**Fix Recommendation:**

For uploaded filenames, all potentially dangerous characters should be removed or replaced, such as . / and other path traversal symbols.

**vantage**

1. **Asynchronous architecture design**

**Advantages**: asyncio is used throughout the system to handle concurrent tasks, including client-server communication and user input processing. You make use of async def and await, so programs can handle multiple tasks efficiently and also avoid blocking operations. I think this is very beneficial for improving program responsiveness, especially in highly concurrent web applications.

**Specific performance:**

In server.py, async def listen() and async def handle\_message() implement non-blocking message listening and processing.

In client.py, the connect\_to\_server() and send\_message() functions implement asynchronous message sending and receiving, ensuring that the client is not blocked (server) by a single operation when communicating with the server (client).

**2. Modular design**

**Advantages:** The structure of your code is very modular, great, and the different functions are clearly divided into separate modules that are easy to maintain and extend. For example, communication between server, client, encryption and neighbouring servers is clearly separated. This design allows each module to focus on its own functionality, enhancing the maintainability and extensibility of the code.

**Specific performance:**

server.py focuses on server startup, connecting, message receiving and sending (server).

client.py focuses on client-server connectivity, message processing, and cryptographic operations (client).

neighbourhood.py handles interactions between neighbourhood servers, allowing the system to scale to multiple servers (neighbourhood).