Peer Review - Yuvraj Singh Anand

Manual Code Review

Key Observations:

- **Mock Functions:** The inclusion of mock functions is understandable for testing purposes, but it is critical to differentiate them from production code or to ensure they are adequately commented to avoid confusion.

- **Lack of Encryption and Signing:** The absence of encryption in the current implementation of the messaging functionality is a significant security flaw. For a messaging app, encryption and message signing are fundamental to ensuring confidentiality, integrity, and authenticity. This must be addressed urgently.

- **Client Disconnect & Leave Functionality:** The current implementation handles client disconnection with a leave message, which is potentially unsafe. This opens the server to port flooding attacks, as malicious actors could keep connections open indefinitely by abusing this functionality. The absence of proper timeout or disconnection handling could result in resource exhaustion, eventually leading to server crashes. I ran a simple test which, after a few hours, generated 400 new connections. Without appropriate measures, this could lead to a denial of service (DoS) over time.

Code Example

```
import websockets
import asyncio
from contextlib import suppress

async def connect_to_server():
    uri = "ws://localhost:8765"
    while True:
        try:
            async with websockets.connect(uri) as websocket:
                await websocket.send("Hello, Server!")
                response = websocket.recv()
                print(f"Received: {response}")
        except Exception as e:
            print(f"Error: {e}")

with suppress(Exception):
    asyncio.get_event_loop().run_until_complete(connect_to_server())
```

It also appears active client connections to close as there must be a limit to how many active ports websocket can handle

```
lloyd@lloydlaptop:~/Desktop/SP2024/PeerReview/Yuvraj Singh Anand$ python3
client1-1.py
Please enter a username: aegizz
'Hello' message was sent to server
> /private
Please enter the recipient's username to send a private chat: aegizz
Now enter your message to aegizz: test
Private chat to aegizz was sent...
Private chat from aegizz to (you): test
> Error: no close frame received or sent
```

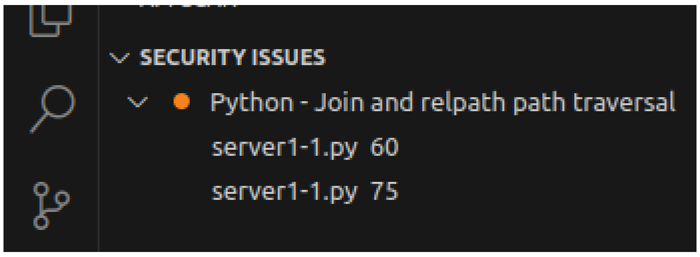
Observations:

- Running the script above quickly spawns multiple connections without proper handling of disconnection or resource cleanup. After manually terminating the attack, approximately 13,000 used ports were released. This confirms the potential for a port exhaustion attack, especially if sustained over a longer period.
- File Upload Vulnerability: Another major issue is the lack of file type validation when handling uploads. This opens up the possibility of uploading malicious files, which could be exploited to attack the web server. File type validation, combined with strong input sanitization, is essential to protect against potential threats like directory traversal or remote code execution attacks.

Static Code Analysis

The static code analysis was straightforward, with minimal issues identified. However, HCL's AppScan CodeSweep flagged two critical issues in the server files.

- Input Sanitization on File Upload: The code responsible for handling file uploads does not sanitize the input properly, allowing users to potentially manipulate file paths and upload files outside the designated directory. This poses a significant security risk, as it could lead to arbitrary file writes or access to sensitive areas of the file system.


SECURITY ISSUES
Python - Join and relpath path traversal
server1-1.py 60
server1-1.py 75

Vulnerable Code

```
    file_path = os.path.join(UPLOAD_DIR, filename)
```

In this snippet, the filename is concatenated directly without sanitization, which makes it vulnerable to path traversal attacks. An attacker could use URL-encoded characters (e.g., ../) to navigate outside the upload directory, potentially overwriting critical system files or accessing restricted areas.

Suggested Fix:

To mitigate this issue, ensure that the uploaded file's name is sanitized, and file uploads are restricted to specific file types and directories.

Conclusion

While the core functionality of the application appears to work as intended, there are several critical security concerns that must be addressed. Encryption should be added to protect data integrity and confidentiality, and the handling of client disconnections must be improved to avoid resource exhaustion. Additionally, proper file upload validation and input sanitization should be implemented to prevent exploitation.