Peer Review - Yuxuan Xie

Manual Code Review

The code is well-documented, providing clear instructions on how to set up and run the server, explaining its communication processes, and offering some guidance on how to extend its functionality. Overall, the documentation is thorough—good work.

In the README, vulnerabilities are already listed, which is helpful for transparency, though it might reduce the impact of discovering them during review. Nonetheless, I will detail a few key vulnerabilities here.

- The code in lines 33 to 57 of index.html contains potential vulnerabilities.
- The chat.html file also has known vulnerabilities.

Starting with index.html, the portion of the code referenced deals with the admin room, which relies on localStorage to set the username and room. This approach allows an authenticated user to access the admin room without any additional security checks. It may be beneficial to implement proper authentication mechanisms using sessions to differentiate admin access from regular user rooms. Without such safeguards, the admin room could be indistinguishable from other rooms. However, if I'm misunderstanding the intent, it's possible that there's no actual data leakage here.

Unfortunately, I was unable to fully test chat.html due to a non-functional message-sending feature when clicking the "send" button.

In chat.html, two external scripts are linked:

```
    <script src="/socket.io/socket.io.js"></script>
    <script src="/js/chat.js"></script>
```

While the chat functionality seems to rely on these scripts, the socket.io.js file was missing from my local copy of the code, which made it impossible to test socket-based communication. This missing file could have been removed for submission purposes or may have been inadvertently excluded.

Upon reviewing chat.js, I identified several potential vulnerabilities. Specifically, the script frequently uses functions such as .innerHTML and insertAdjacentHTML, which are prone to cross-site scripting (XSS) attacks if user input is not properly sanitized.

After testing, I confirmed that both the room name and username fields are vulnerable to stored XSS. By injecting malicious HTML or JavaScript into these fields, it is possible to execute arbitrary code on other users' machines. For example, using the following payload demonstrates the vulnerability:

```

<img src="invalid_link" onerror="alert();" >

```

This allows an attacker to execute JavaScript in the context of any chat room.

The other notable issue is there is a distinc lack of moderating the file types that are uploaded this is a serious issue as malicious code could be uploaded to the website then exploited.

Static Analysis

The static analysis tool I used was HCL's Appscan Codesweep, this went over all the JS files and looked for vulnerable code.

In chat.js:

1 x innerHTML 6 x insertAdjacentHTML

In index.js there were two flaws relevant to the app sessions, since you didn't seem to use this its not entirely relevant but I'll add it anyway.

```
A) The cookie should always be secure
B) The plain text secret for the express session is set to the default
value.
```

```
// Using session middleware
app.use(session({
    secret: 'your-secret-key', // Use a secure key
    resave: false,
    saveUninitialized: false,
    cookie: { secure: false }  // If you are using https, set it to true
}));
```