Mandatory Assignment Part 03 – Review

Project in mind: *https://github.com/WillemSch/login-server*

For this part, you'll get access to another project. Examine the documentation and the code and try running the application. Write a short security analysis.

- *Do you understand the overall design, based on the documentation and available source code?*

Yes! I read the entire documentation and it was easily readable, made sense and I tested the application while reading it incase there was an error or something he missed. The code is split into three different areas of interest. The app.py which originally was unstructured and hard to read has been stripped of almost all code and now solely operates as a flask-starting point. The rest of the code (old and new) is split into either controllers to handle requests and the services to execute DB queries and commands.

- *Do you see any potential improvements that could be made in design / coding style / best practices? I.e., to make the source code clearer and make it easier to spot bugs.*

I think there could be better/more use of code-commenting. A lot of the python files contain little to no commenting at all and it's easier for someone reviewing the code (me) if there are comments explaining difficult code.

- *Based on the threat model described by the author(s) (from part 2B), do you see any vulnerabilities in the application?*

From what I can see, the author has not described any threat model, but has mentioned vulnerabilities.
To build on the existing points regarding vulnerabilities:
An attacker would most likely be after confidential information in the state that the application is currently in. It's a messaging app.

The author has fixed the issues regarding impersonation. (The user can no longer choose who the message is sent by, as well as inputs have been sanitized to avoid SQL injections etc...)

A current vulnerability I see is due to weak user credentials (I was able to sign up an account with the password "123" which would almost instantly crack either to a brute force attack, or one from a keyword-list; A user would be able to steal login information and their messages if that user used a weak password.

This isn't necessarily the author's fault, but it can be mitigated/prevented by adding a requirement to the password (e.g min. 8 characters, 1 special character).

- *Do you see any security risks that the author(s) haven't considered? (Have a look at the OWASP Top Ten list and "Unit 2: Attack Vectors" in Security for Software Engineers)*

OWASP Top Ten:

<u>Identification and Authentication Failures:</u>

The app in its current state permits brute forcing and/or automated attacks. There is no timeout/captcha implementation as well as a password could be anything (Even "123") Therefore an attacker could just send requests automated until it guesses correctly.

<u>Security Logging and Monitoring Failures</u>

As far as I can see, there is no logging system, and as mentioned on OWASP: "Without logging and monitoring, breaches cannot be detected."

These are the two points from OSWASP that I don't believe are considered.

Regarding XSS and SQL injections, these are not possible, as the author mentioned. In the beginning, they were however, and you could inject malicious scripts and SQL queries into inputs. This has been fixed.

- *You may find it useful to try some security testing tools. Also, think back to the techniques you used in the exercise on XSS attacks – what sort of damage could you do by tricking a logged-in user into click on a link?*

You could trick the user into clicking on a link by making it look safe/an extension of the current site. From then on, you could prompt the user to hand over confidential information, make the user download malicious software, trick the user act in ways you choose.