MultiPersonCalculator

SAMULI KÄPPI & AKI KANKAANPÄÄ

The Aim

- The aim of the program was to provide a way for two or more people to use the same calculator at the same time.
- Achieved by having a central server where people need to log in to with their credentials.
- While the program has no functional use-cases where it would be better than a calculator and any type of communication method, it is meant to be a concept to work securely while interracting with a backend server.

Program Structure

- The project contains both a backend Flask server, as well as a separate application.
- The user interacts with the server via HTTP requests using the application widgets. These requests are predetermined and non-modifiable by the user other than for login and registration details.
- The application gives the user a similar interface to any simple calculator visually, with buttons that change the backend status and updates it to the user.

Language and frameworks

- Python was used to create this project for its simplicity and ease of use.
- Flask was used for the backend server for the same reason that Python was used, it's easy and it just works.
- Argon2 was chosen over standard bcrypt since it's modern and has won a password hashing competition in 2015.
- JWT's were used to handle user session tokens.

Secure Programming

- Passwords hashed and salted using argon2id `ph.hash(password)` default parameters aim for 50ms hash time.
- Passwords are stored in a hashed form in case someone manages to gain access to the database.
- JSON web tokens were used to maintain session cookies with limited duration.

```
jwt.encode({"exp": datetime.datetime.now(tz=datetime.timezone.utc) + datetime.timedelta(minutes=10)}, key, algorithm="HS256")
```

The secret key for the JWT tokens was generated with Secrets library since it offered cryptographically secure pseudo random number generation. A 256-bit key is generated with it when the server starts.

Secure Programming

- All user input is being validated, even input that shouldn't be possible to be sent.
- HTTP is going over TLS to prevent passive attackers.
- TLS is done by using Flaks's adhoc method which generates certificate on the fly. The downside of this approach is that the user's can't validate that the certificate.
- User's are limited to the actions the buttons provide. They can bypass this by sending http requests to the server, but if the requests don't match the exact format they are rejected.

Testing

- Testing was done manually by interacting with the program and analyzing the code.
- Some testing was also done sending externally made requests to make sure the program would reject any outside interference.
- Since user inputs are rather limited pretty much everything was tested.