Security analysis of TRIMM runner app group 42

Nowadays, security is a very sensitive topic and also one of the biggest issues in the software industry.

We wanted to be careful when approaching this matter and as it is often said in the security lectures you do not have to invent new ways use proven techniques. So we followed OWASP's guidelines closely.

First of all throughout the whole project we used prepared statements which is the best way to prevent SQL injection attacks. So that an intruder cannot comment or create a tautology while trying to log in as another user. Or even try to access and delete the whole database.

Cross-Site Scripting was prevented by input sanitation on the frontend. We used a ready javascript library which was recommended by OWASP with all the edge cases covered. So now an intruder cannot embed malicious scripts or URLs because all the input in our web application is sanitized before being sent to the server or executed by the browser. So we prevented reflected and stored XSS.Also this library prevents from Command injection attacks even though we do not use cmd commands in our application maybe in the future an extension could require so and the application will be safe.

Passwords are being hashed and then stored in the database using PBKDF2withHmacSha256 which as key derivation function combined with SHA256 is one of the best ways to prevent Rainbow attacks, table lookups because the password is iterated 60 000 times with this algorithm which makes computationally expensive to brute force the passwords. Adding more to password we decided to use salt which is every time randomly generated using CPRNG so that we shrink the chance of having the same hash for the same plain text passwords. We also have set the key length to 256.

Changing of password is also secured by using third party identifier like e.g. Gmail. We send a safe token(CPRNG) to the email of the user who wants to change passwords of course the intruder cannot use theirs email for that because emails are stored in the database and checked. Also for sending the emails because we do not want someone to "listen" we open a TLS channel to the mail server to send the token.

We also verify the email when registering it is mandatory to verify your email in order to complete our security "wall". Again we open a TLS channel to the mail server to send CPRNG token. This way also no one can abuse your email by registering you in our web application without your consent.

We implemented also session management. Whenever a user logs in new CPRNG token is generated with an expiry time of 1 hour. First it is hard to brute force the session token and use it to query information about the user. And it uniquely identifies the user without using easy to break usernames and etc.