SEC
Lab #3

- This lab will be graded.
- The quality of your code will be graded.
- Your submission has to be in Rust.
- Exceptionally, we will not ask you to provide tests for this lab.
- We provide you with a working template so that you can focus on the important parts of the lab.

1 Description

The objective of this lab is to analyze an application and fix the flaws that are present. Like the previous lab, we provide a client and a server application, but this time they are fully implemented. The applications communicate through a TLS channel that you have to configure yourself.
The server stores a database of users (username, password, phone number, role) and interacts with the client application. While there is no initial user authentication to interact with the server, some actions require a logged-in user with a specific role (HR/standard user).
The following actions are available:

1. **Show users**: Display the users information (username, phone number, role)
2. **Change my phone number**: Allows a user to change his own phone number
3. **Change someone's phone number**: Allows a user with the HR role to change anyone's phone number
4. **Add user**: Allows a user with the HR role to add a new user
5. **Login**: Login to the application with a username/password
6. **Logout**: Logout from the application

2 Indications

We provide below a non-exhaustive list of tasks to improve the application.

2.1 TLS

In order to start the application, you first have to generate a private key and a certificate for the server. The fastest way to get up and running is to generate a self-signed **ECDSA** certificate, but you can also create a complete PKI chain if you wish. When generating the certificate, you will have to specify the certificate subject (CN). You can either use `localhost` or a different name (but you will need to update your hosts file). The private key has to be in the PKCS8 format. Once this is done, you can update/verify the configuration of the server and the client.

Useful links:

- `https://cloud.google.com/iot/docs/how-tos/credentials/keys`
- `https://wiki.mozilla.org/Security/Server_Side_TLS`

2.2 Input validation

The server does not verify the inputs provided by the client application. On the server-side, there is no need to implement a complex input validation process; just validate the inputs and return an error to the client if it fails, no need to request the inputs again.

2.3 Logging

You have to implement a proper logging policy for the application. You can use the **simplelog** or **log4rs** crates to help you do this. There is no need to add logging to the client application.

2.4 Access controls

The access controls are currently split into multiple functions which makes it hard to maintain. Try to improve the access controls with what you saw in the courses. When you start the server for the first time, the users below are created automatically and stored in `db.ron`. You are free to change it.

Username	Password	Role
default_user	default_pass	Standard user
default_hr	default_pass	HR

3 Report

Include a short report that highlights the main changes to the application, and eventually other comments.