SEC

Lab #2 - Authentication

- This lab will be graded.
- The quality of your code will be graded.
- Your submission has to be in Rust.
- Test your code as much as you can but we will **not** put the focus on testing.
- We provide you with a template so that you can focus on the essential but you are allowed to modify it.
- You have to answer some questions in a **report** (but only these questions). This report will contribute to the **course grade**.

1 Two-factor Authentication

The goal of this lab is to implement a two-factor authentication mechanism (2FA). We will use the following two factors:

- A password
- A Yubikey (`https://www.yubico.com/`)

You have to implement the following features:

1. A registration process asking for a password, an email address, and a Yubikey. The email is going to be used as a username and to reset the password.
2. A login process asking for an email address, a password, and a Yubikey (when 2FA is enabled).
3. A password reset process sending a token by email.
4. A possibility to disable the 2FA for a user.

2 Indications

2.1 General

- The focus of this lab is the security part.
- You can modify the templates as you wish.
- Do not forget to verify users' inputs.
- We do **not** look at the network security and assume that the communication channel is secure. Do not use this template in production!

2.2 Authentication

- Regarding **password authentication**, we will use **strong authentication** using a challenge-response protocol. This means that we do **not** send the password to authenticate. Instead, we receive a **challenge**, which is a random value (e.g. 128 bits) and we have to compute a **response** using a MAC algorithm and a key. In this lab, we are going to use the `HMAC-SHA256` algorithm. The **key** for the MAC will be derived from the password using a hash function of your choice.[1]
- For the second factor, using the Yubikey, we will use the `PIV` authentication (`https://developers.yubico.com/PIV/`, `https://docs.rs/yubikey/0.5.0/yubikey/piv/index.html`). Regarding the signature algorithm, we will use the `ECDSA` signature algorithm over the elliptic curve P-256. You do **not** need to understand the advantages and problems of these various algorithms in this class.

2.3 Communications

- Use the `send()` and `receive()` function of `Connection` to send any serializable (with `serde`) type between the client and the server.
- Do not forget to add `#[derive(Serialize, Deserialize, Debug)]` to enable serialization through `serde`.

2.4 Mailer

Sending emails from an application is not trivial. You can for example use a Gmail address set up with an `application password`: `https://support.google.com/mail/answer/185833`.

2.5 Useful crates

- `read_input`: to obtain easily users inputs.
- `yubikey`: To interact with the Yubikeys on the client side. If needed, install the YubiKey Manager CLI `https://developers.yubico.com/yubikey-manager/`.

[1] In reality, we do not use a hash function but a `key derivation function` (KDF), but this is out of the scope of this course.

- `ecdsa` and `p256`: to verify the signatures on the **server side**, `p256` is only used to setup the `ecdsa` crate with the corresponding elliptic curve.
- `hmac`: For the HMAC authentication function.
- `lettre` (**v0.10**): To send emails from the server.

3 Output examples

```
<< Please register yourself >>
- Email: test@test.test
- Password: test
No Yubikey detected: Please enter one and press [Enter] to continue...

[[ Authentication success ]]


<< Please authenticate yourself >>
- Email: test@test.test
- Password: test
PIN: 123456
Pin: [OK]

[[ Authentication success ]]
```

4 Report

Please answer to the following questions in a report.

1. What are the advantages of a challenge-response authentication compared to a weak authentication protocol?
2. In your application, when do you require the user to input its Yubikey? Justify.
3. What can you do to handle Yubilkey losses?
4. An attacker recovered the challenge and the associated response (in the password authentication). How does this allow an attacker to perform a bruteforce attack? What did you implement to make this attack hard?
5. For sending the email, what are the advantages of using an application password?
6. In the Yubikey, what is the purpose of the management key, the pin and the puk?