Hacettepe University

Computer Engineering Department

BBM465 - Information Security Lab - 2024 Fall

Assignment 3

December 4, 2024

Group ID:
15

Students name:
Ece Nur Taş
Büşra Burhan

Students Number:
2210356039
2220356051

1 Client Side

1.1 Registration Page

The `register` function is responsible for handling user registration within the application. It ensures the validation and secure storage of user credentials while maintaining uniqueness and proper password management.

- **Username**: The desired username of the user. If a username has already been used during a previous registration, the new user is required to choose a different username. The username must only contain alphabetic characters and numbers; special characters are not allowed.

- **Password**: The password chosen by the user. It must match the validation password entered during registration. Additionally, the password must be at least 7 characters long.
  The entered password is hashed using `Crypto.Hash.SHA256` before being sent to the server, ensuring secure storage of user credentials.

- **HOTP Chain Generation**: During registration, an OTP chain is generated for the user in a recursive function `create_OTP`. The entered password is used as 'seed' value and it is the intial value for OTP generation function. The dictionary `dict` is populated with the OTP chain for the registered user. This dictionary stores all 100 OTPs generated during the registration process. The last generated OTP is sent to the server and stored in the variable `current_otp_dict[username]`. A dictionary is used because it enables the storage of unique OTP chains for each user, with new key-value pairs added for every registered user. See Section 2.2 for further details about how the OTP chain is used.

- The `create_database` function is invoked to securely store user data in database file. Refer to Section 2.1 for a detailed explanation of this function.

1.2 Login Page

The `login` function takes the username and password provided by the user. If the given username does not exist in the database, the user is redirected to the registration page.

- If the user exists, the `update_database` function on the server verifies whether the entered password matches the one used during the registration process. If the passwords match, the hashed $(n-1)$th OTP is checked against the nth OTP that was previously stored on the server. The OTP chain decreases in size each time the user logs in. Once all OTPs are depleted, a new OTP chain of length 100 is generated using the user's password, similar to the

process during registration. The last OTP in the chain is sent to the server, and the entire process is repeated. This mechanism ensures enhanced security for user logins. The details of this process are explained in Section 2.2.

1.3 Welcome Page

When user succesfully logs in, welcome page with the "Welcome, <username>" comes. To provide this, the part of the code in the welcome.html is updated as below:

```
<div class="welcome-container">
  <h1>Welcome, {{username}}!</h1>
  <p>You have successfully logged in. Enjoy your experience!</p>
</div>
```

2 Server Side

Server is a simulation of a web server. Some functions implemented at server side are used at client side.

2.1 Create Database File

When `create_database` function is invoked from client side, the information of a user which has just registered, stored in the database.txt file as in the format of username1;hashedpassword;OTPToken;Counter. Counter is 0 at the registration.

2.2 Update Database File

The `update_database` function plays a crucial role in securely managing user credentials and OTP chains. It is responsible for validating user login credentials and updating the server's database with the latest OTP after a successful login. By dynamically updating the OTP chain after each successful login, the function ensures that only authenticated users can log in, while maintaining the security of user credentials and OTPs.

The function starts by attempting to open the database file in append mode. If successful, it calculates the `counter`, which is derived from the length of the remaining OTP chain for the given `username`. The `new_otp` is extracted as the last OTP from the user's OTP chain.

- First, the function verifies if the provided `hashed_password` matches the stored hashed password for the `username` in the server. This comparison is made using the `get_hashed_password` function.

- If the passwords match, the function proceeds to verify if the hashed version of the `new_otp` matches the currently stored OTP in the server's `current_otp_dict` for the `username`. This step ensures that the login attempt is authenticated using the correct OTP from the chain.
- Upon successful validation, the function writes the updated user information to the database file as a format of username1;hashedpassword;OTPToken;Counter. Additionally, the `current_otp_dict` for the `username` is updated to the latest OTP.
- If any validation step fails, the the login attempt results with a new login page.

3 Database File Encryption

The `encrypt_line` and `decrypt_line` functions play a critical role in securing the database by encrypting not only passwords but the entire database file containing all user information.

encrypt_line Function: This function utilizes the RSA algorithm with the public key to encrypt a given line and returns the encrypted result. It is executed every time a line is written to the database file, ensuring that all data is stored in an encrypted format directly in the database.

decrypt_line Function: This function is used whenever data needs to be retrieved from the database, such as searching for a hashed password or checking if a user is registered. The function retrieves the encrypted line from the database and decrypts it using the private key, returning the unencrypted version. This decryption process does not alter the database, which always remains encrypted. Only a decrypted copy of the data is used as needed.

4 Notes

http://127.0.0.1:5000 is the local address.
http://127.0.0.1:5000//register
http://127.0.0.1:5000//login
http://127.0.0.1:5000//welcome