Introduction

In this project we are going to develop and deploy a web application to our server, this application should support user accounts. One should be able to register a new account and login to their own account, after logging in the user should be greeted by a welcome message including their own username. This project is built upon the pre-code that was given in the assignment. The application should inherently protect the users from network-based replay attacks, snooping and their passwords from dictionary attacks. This will be done by establishing a https connection with the user, ensuring that all the data being transferred is encrypted and can only be read by receiving a counter part in the connection.

Background Theory

Certificate:

A website security certificate is a digital stamp of approval given by a trusted third-party known as a certificate authority (CA). A website security certificate is also known as an HTTPS certificate. One can see if their connection is certified by looking at the “lock” to the left of the URL.

Session cookies:

Typically, a cookie is used to tell if two requests come from the same browser keeping a user logged in. The cookie is a file containing an identifier (a string of letters and numbers) that a website server sends to a browser for temporary use during a limited timeframe.

Your design and implementation

This web application uses a secure https connection between the user and server, by doing this we ensure all the data is encrypted and signed. Security is needed to protect against cyber-attacks that focus on acquiring sensitive information or defending against illegitimate authentication attempts. It will ensure the information integrity and privacy is upheld. When establishing a secure and trusted connection with the user a certificate is needed, we use “Let’s Encrypt” as a CA (certificate authority).

When the server receives the password of a new user it will hash the password before storing it, when the user then tries to login to the application the integrity and privacy will be upheld. When verifying the incoming password against the stored hashed password, the application will only check if the hash value is the same before approving the authentication.

Preventing others from entering another's account through entering a list of commonly used passwords is hard, but to prevent the user from having a password that's simple to crack we set some restrictions. The restrictions used are:

- The length of the password needs to be between 8 and 30
- Must contain a capital letter
- Needs to include a number (0-9) and a symbol

After a user is authenticated, their username is stored in a session variable that can be accessed through the rest of the user's interactions on the web site. This lets us know which user is logged in and will remain there until the user exits the browser.

The different libraries we used were Flask, pythons re (*regular expression) and Bcrypt.* Flask was used for storing cookies through sessions, handling redirects and rendering of templates, and request was used to get user input from html. Regular expression was used to search the password of a registering user, to make sure the password met our requirements. Bcrypt handled the password hashing and check sum, to make sure that the passwords were stored safely.

Evaluation / Discussions

Registration:

The user has no restrictions on their username but if they do not meet the requirements for making a password, they are met with an error message. The same happens when the passwords typed in do not math each other. One will be redirected to the login page if it's a success. The passwords are also successfully hashed as one can see by opening the json file.

Login:

After entering your username and password the server will search through the user data and find your user, if it's unable to find a username and password that match you are met with an error message. If successful one is redirected to the logged-in page.

Logged-in:

This page will display the user's username along with a welcome message. This was made possible through session cookies.

I was unable to reach one of the requirements given, and that was to deploy it to the server. I have uploaded everything, and it is able to run on the server but if I disconnect or want to do something else, I must stop hosting.

Conclusion

The web application was developed and can withstand network-based replay attacks, snooping and dictionary attacks through hashing and https connection. One can register as a new user and login, when logged in your username will be displayed, this was done with Flask session.