# INF-2310

# Assignment 4, Web Authentication


# Magnus Dahl-Hansen, April 2023


## Introductions

This report describes the implementation of a simple web-site that handles and manages log-in credentials to register and authenticate users. The application offers functionality for registering new users, logging in with an already registered users, displaying a welcome message to logged in users and prevent replay attacks, network snooping and dictionary attacks.

## Background Theory

Before reading this report the reader should be familiar with Flask, Hyper Text Transfer Protocol Secure, TSL/SSL and three different cyber attacks: replay attacks, network snooping and dictionary attacks.

This application is built using the Flask module which is a framework for building web applications in python. Flask is referred to as a micro framework designed to keep the creation of a web application scalable and simple.

Hyper Text Transfer Protocol Secure (HTTPS) is a version of HTTP enabling secure transfer of data over the web. HTTPS most commonly uses Secure Socket Layer (SSL) or Transport Security Layer (TSL) protocols to encrypt data being sent between a server and a client. This web-application makes use of SSL to encrypt data and is set up in the following way: First a secure connection is established between a client and a server. The server then provides the client with a certificate proving its authenticity. The two parts proceed to exchange public keys and use asymmetric encryption to agree upon a shared key to further use symmetric encryption to exchange data.

Replay attacks are a type of network attack where the attacker intercepts an ongoing connection and saves packets being sent between two parts. The attacker then uses the packets to resend them at a later time to impersonate a legitimate client or server. Replay attacks can be prevented in many ways but the most common ones is to make use of a nonce variable which can only be used once and a timestamp to prove the originality of a message. SSL makes use of both these techniques.

Network snooping is an attack where the attacker tries to pick up and examining data packets being sent to obtain private information such as passwords, credit card details and so on. Network snooping can be prevented by encrypting data so that even if a data packet was picked up by an attacker it can not be decrypted.

Dictionary attacks involves trying to guess passwords and other private information by systematically trying a large amount of commonly used passwords and other phrases. Software tools are commonly used to rapidly try inputting all the different combinations. To prevent against dictionary attacks one can implement an algorithm which requires a user to create strong password which are not easily guessed, such as requiring upper and lower case letters, numbers and special characters. Another solution is to limit the number of tries and to enable two-factor authentication.

## Design and implementation

When starting the web-application a check will be made to see if there exists a database for storing usernames and passwords. If no database is found an SQLite3 database will be created with the following attributes: ID, username and password.

The code is divided into five main functions where three of them will render HTML templates for the home-page, log-in page and register page. The remaining two handles the registration of a new user and log-in credentials.

By default the application will proceed to render a HTML home-page template telling the user to either log in or register as a new user. However, should a local dictionary indicate that a user is already logged in a home-page displaying a welcome message will be rendered instead by providing the “username” from the dictionary to the HTML home-page form.

When a user wants to register as a new user the register_post function is called. This function receives a request sent from the HTML-form which contains the users entered username, and two passwords which should be equal. The username request is used to check whether or not the username already exists. If so, the function responds with an error message telling the user that the username already exists and that a new one must be chosen. If the username is unique the criteria for the password is checked. The criteria for the password is set for it to be at least six characters long and must contain at least one big character and one integer. If these criteria is not met an error message is sent telling the criteria for the password. Should the username be unique, password criteria be satisfied and two equal passwords are provided then a SQLite3 query is performed which adds the username and password along with an unique ID to the database.

When a new user has been successfully created he will be able to log in. From the HTML log-in template a username and password is sent to the server and received in the login_post function. This function will use the username and password from the request to first check if the username exists in the database then see if the password matches the username in the database. Should all be correct the user is redirected to the HTML logged-in form and the username is added to a dictionary keeping track of logged in users. Should however the username or password be incorrect an error response is sent displaying whether the username or password is incorrect.

Including the five main functions a helper function has been made to check if a given username exists in the database. This is simply done by executing an SQLite3 query on the database and returning true or false.

TSL/SSL is implemented by telling the Flask application to run the app with SSL encryption and providing SSL_context to the app.run() function. To run SSL encryption a certificate and private key must be provided and have been obtained by CertBot. By providing these the application will run using HTTPS.

## Evaluation / Discussions

The implemented code is able to run a web application which returns specific endpoints based on the users input. These are: rendering a home page, register page and a log in page. The code is also able to successfully register a new user, log in with a registered username, displaying a welcome message to logged in users and fend against replay attack, dictionary attacks and network snooping.

The application makes use of SSL to encrypt data packets to fend against network snooping and uses nonce and timestamps to fend against replay attacks. To fend against dictionary attacks strong password requirements have been implemented and an attempt to implement cookies was made to limit a users log in attempts. However, unknown errors occurred when trying to implement the cookie functionality by using Flask's library. The errors have not been resolved.

The application is running HTTPS and the entered user credentials are encrypted when sent to the server. However, when opening the web browser you may be redirected to a page saying that the connection is not secure. This is because the certificate that has been created to run HTTPS is self made by using CertBot and is not signed by a legitimate authority and therefore not recognized as a legitimate source.

A last mention is that the application does not run on the course server. The application has been attempted to be copied to the server by using Secure Copy (SCP) and GiT. SCP is run by providing a file source to be copied and a destination source to move copy the files in the terminal. SCP did not allow this action because the password used to access the server was refused. Creating a GiT repository on the server and cloning the code to it was also attempted, but once again the server refused to accept the password used for GiT. A possible reason for this may be that two-factor authentication must be used to access this specific repository which causes difficulties when using a terminal.

## Conclusions

This report described how a simple web-site handling registration and log-in functionality was implemented. The implementation is able to successfully render the required HTML forms, register new users and logging in with existing users. The report also describes the different methods used to secure communication between the client and the server, how replay attacks, network snooping and dictionary attacks were accounted for and why the application is not running on the course server.