ECE-458 (Spring 2020) – Project 2
Password Safe Web Application
(Designed by Sean Kauffman)
Due on July 23, 2020 (11:59pm)

1 Instructions

Your solutions i.e., codes and the report should be submitted through LEARN and Crowdmark by the specified due date. All files should be included in one zip or tar.gz file. Your report should be a plain text file in the top level of the archive. You should also submit your report to Crowdmark. The assignment requires that you only modify two of the source files from the boilerplate: **student.js** and **student.php**. These files must also be included in the top level of the archive. IF OTHER SOURCE FILES ARE INCLUDED THEY WILL BE IGNORED AND REPLACED WITH FILES FROM THE BOILERPLATE.

- alice-a2.tar.gz:
  - report.txt
  - student.js
  - student.php

You are allowed to work individually or in groups of two with your classmates to discuss ideas and solutions. Your group should be registered in Crowdmark in order to be marked correctly. The submitted work MUST BE STRICTLY YOUR OWN WRITING. This policy also applies to looking up ideas on the Internet or textbooks. Moreover, any collaboration MUST be acknowledged in the submission (explicitly name the persons that helped you or that you helped, persons that shared ideas with you or that you shared ideas with them; explicitly name online sources or textbooks where you found ideas that helped you write your solution, etc.).

The standard advice on what to do about collaborations to avoid getting in trouble is: you may discuss and share ideas with classmates, but then, put aside any annotations and DO NOT LOOK at them when writing your own solution (this ensures that your writing is original and distinct). Also, NEVER share any of your written solutions or your written code with anyone; preferably, DO NOT ALLOW ANYONE to look at your written solutions (keep in mind that if they have a good memory and write the exact text in your solutions and submit it, you will be equally liable in the academic dishonesty incident). It is assumed that you are familiar with the University policy on academic integrity.

2 ECE 458 Password Safe Web Application

In this project, you will design and implement the security aspects of a password storage/retrieval web application. For each user, passwords for different sites are stored and protected by a "master password" (i.e., that is required for the user to log in).

The system must feature client-side encryption so that it is resilient to database theft (or even unauthorized access by a system administrator, law enforcement subpoenas requesting

access to a user or users passwords, etc.). The user has their "master password" that is used for logging into the system, and that same master password is used to encrypt the information stored in the server's database. Log in details such as site names and site users do not need to be encrypted with the master password, only the site passwords must be.

You should consider all forms of remote attack discussed in the course. Although the application uses a browser-based front end, you should not assume attacks can only originate from the browser. You may assume that the attacker does not control the web server, but you should assume that an attacker may gain access to your source files and database.

The system consists of the following pages which you will implement:

(1) A *sign up* page — The user chooses a username (which must be unique) and a password, and provides a valid email and their full name. The application may assume that the email address is valid as long as it is formatted correctly. The application also does not need to support user management, such as password update and recovery.

(2) A *login* page — The user submits a username and password; upon successful authentication, the system initiates a session and displays the dashboard with a personalized welcome message.

(3) An *add password* page — Once logged in, users can add passwords that they want to store to the safe. The page takes three parameters: the site or app for which this password applies, the username (optional), and the password (notice that the client needs to encrypt the password before sending so that the server will *never* have access to any of the passwords). *Only the password field needs to be encrypted!*

This same page may be used for data other than passwords like security questions and answers (e.g., the user may enter, in the "site" field: *Bank of Waterloo – security question – favourite movie*, leave username field empty, and the answer to that question may be given as the password field).

(4) A *retrieve password* page — The user selects the site from a list, and the application looks up the entry and transmits the encrypted password to the browser. Once at the browser, the client must decrypt it before displaying it.

3 Implementation

Much of the web application has been implemented for you. You are not allowed to modify the HTML or the database schema, and significant boilerplate has been provided for both the client JavaScript and the server PHP. You will be responsible for designing and writing the missing parts.

You are not allowed to use any external libraries, such as CryptoJS, jQuery, or a database ORM. This means you will need to use provided PRNG, hash, ecryption, and decryption functions in the boilerplate. This also means you must write your own SQL to interact with database. You should assume that the web browser used to access the site is **Firefox 75 on Linux**. You should assume the version of PHP used for the server is **PHP 7.4.5**. For hashing you should use SHA2-256 and for encryption you should use AES-256 in CBC mode. These are the algorithms used in the provided functions.

A few compromises must be made due to our desire to avoid running full-featured web servers. One such compromise is that the application will not use SSL/TLS, while this would be a requirement in a real setting. Another compromise is that there is no layer of security on the database because we are using SQLite3 and it does not support any permissions apart from filesystem permissions.

(i) Boilerplate

The provided boilerplate is a functioning web application, but it doesn't do anything. You will need to implement certain functions to complete the assignment. There are two files that must be modified: **student.js** and **student.php**. You are encouraged to explore the boilerplate code to make sure you understand how it works before proceeding with your implementation. However, **students are not allowed to modify other files in the project**.

While the boilerplate does implement any of the functionality of the application, it does do all of the network communication that is needed. A good idea to get started on your implementation is to start the server (`./start.sh`) and load the application in your browser (`http://localhost:8000/`). Open the developer console and look at the network connections while also watching the shell where the start script was run. Try clicking around the site and filling in form fields.

Files in the boilerplate:

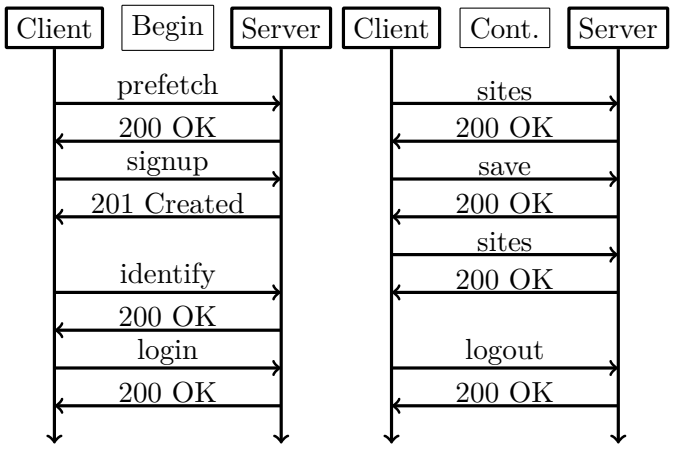
- *api.php* – Contains APIs that you must use to complete the assignment. **Look in this file, but don't modify it!**
- *client.js* – Contains functionality and APIs for the client portion of the application. **Look in this file, but don't modify it!**
- *dropdb.sql* – A SQL script to drop all the tables and indexes in the database. To use this script, run `sqlite3 passwordsafe.db < dropdb.sql`.
- *favicon.ico* – A terrible favicon file. This is what the browser will display in the tab when you visit the site.
- *index.html* – The HTML for the application. **This file must not be modified.** You can load this file in your web browser by first running *start.sh* and then visiting `http://localhost:8000/`.
- *initdb.sql* – A SQL script to create all the tables and indexes in the database. To use this script, run `sqlite3 passwordsafe.db < initdb.sql`. **Look in this file, but don't modify it!**
- *passwordsafe.db* – The sqlite3 database used by the application. This file should exist and have the tables initialized by the *initdb.sql* script before the application will work.
- *server.php* – Contains functionality for the server portion of the application. **This file must not be modified.**

- *start.sh* – A shell script that will start the PHP server in the current directory on port 8000. Run it as `./start.sh` and then direct your web browser to `http://localhost:8000/`.
- *styles.css* – The stylesheets for index.html. You can play around with this if you want things to look prettier during your testing.
- *student.js* – Contains the client functions that you must implement to complete the assignment. **This file must be modified!**
- *student.php* – Contains the server functions that you must implement to complete the assignment. **This file must be modified!**

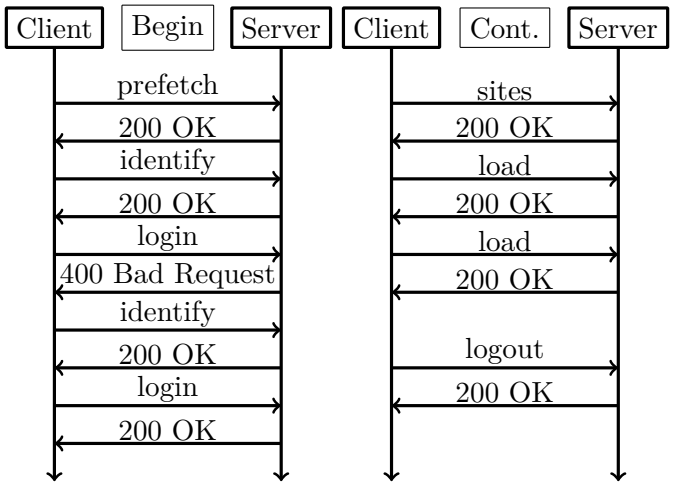
(ii) Workflows

The site is implemented as a single-page-application, which means that there is only one HTML file that is never reloaded during a web session. Instead of loading new HTML files, content on the site is updated by JavaScript that communicates with a server via HTTP requests with JSON (`https://json.org/`) bodies.

This diagram shows network communication flow for a session. In this visit, the client creates a new account, then uses the account to log in and create a new site, and finally logs out.



This diagram shows network communication flow for a session. In this visit, the client fails to log in, then successfully logs in and then loads two site passwords and logs out.



4 Deliverables

You will submit the modified student files as well as a short written report. It must be possible to run the application by copying the two submitted files into the boilerplate! All of the files must be in a single zip or tar.gz archive. The source files and the report should be at the top level of the archive. The report should be plain text. Your solutions should be submitted through both LEARN and Crowdmark by the specified due date.

- alice-a2.tar.gz:
  - report.txt
  - student.js
  - student.php

(i) Report

Please write a one-page report about the assignment explaining your design decisions. The report should be submitted to Crowdmark. This report should include the reasoning behind your choices as well as your opinions of possible deficiencies in the final work. Please make sure the report contains your name and student ID and is written in plain text. *A good idea is to write the design part of the report first, before you implement anything.*

(ii) Marking

Marking will focus primarily on the security aspects of the assignment. However, the application must also work! The best way to approach the assignment is to design the application with security in mind from the beginning. Follow the recommendations given in lectures and be careful with how encryption is used.

5 Frequently Asked Questions

(i) Should I be concerned with eavesdropping on the HTTP connection because it is unencrypted?

No. You should pretend that the HTTP connections are encrypted with SSL/TLS, even though they are not.

(ii) Should I use RSA to encrypt the log in password since it is sent over an unencrypted connection?

No. See Question (i). You should pretend the connection is encrypted.

(iii) How can I get the boilerplate to work under Ubuntu?

Install the following packages (using `sudo apt-get install` *packagename*):

- php
- php-sqlite3
- sqlite3

After those are installed you can run `start.sh` from a bash shell.

(iv) How can I get the boilerplate to work under Windows?

Use Windows Subsystem for Linux (WSL) to install Ubuntu:
`https://docs.microsoft.com/en-us/windows/wsl/install-win10`.
Then follow the instructions from Question (iii).

(v) Should I modify api.php?

No! You must not modify any source files except for student.js and student.php! If your code relies on modifications to other files, your application won't work when it is marked and you will receive a poor grade.

(vi) Can I modify other files, like server.php?

No. See Question (v). You must only modify student.js and student.php.

(vii) Can I use a method to connect to the database other than the passed PDO connection object?

No! You must use the passed PDO connection.

(viii) I'm worried that if the attacker obtains the database they will have enough information to log in. Is this a problem?

This is fine. The important thing is that an attacker who obtains the database will not have enough information to decrypt the secret data. Consider this: if the attacker has the entire database, what will they gain from logging into the site? Note: if there was a security layer on the database this would be more of a concern, since someone who accesses the *user* table might not be able to read the *user_safe* table.

(ix) How can I save the master password on the client when the user logs in?

You can just store it in a global JavaScript variable, or you can use sessionStorage if you want to get fancy. The web site is what is called a *single-page app*, so the JavaScript memory space persists during the life of the site visit.

(x) Should I send the log-in password in plaintext?

You should not give the server enough information to decrypt site passwords.

(xi) I'm unclear about the use of column X in table Y. What is it for?

The database schema is there so you don't have to write one yourself. It should be considered a tool to use to achieve your design. You should not try to reverse engineer the database schema to determine your design.

(xii) Where should I start

1. Load up the boilerplate and watch the network requests in your browser's developer console as you interact with the site. Compare to what you are seeing printed in the console where you ran `start.sh`. You can start to play around with this by adding more logging to see what is going on.

2. Read the documentation. The assignment has some but there is much more in the source files.

3. Use a pen and paper to write out what should happen for each action on the site. This should help you consider what data needs to be encrypted, hashed, or stored.

(xiii) Should my teammate and I submit one or two solutions?

You should submit only one solution to LEARN and Crowdmark including codes and report if you work in pairs. .

(xiv) Why PHP?

In short, because it is a popular and easy-to-use language for building web sites: `https://www.theregister.com/2020/06/08/25_years_of_php/`.