# *Web Programming IV (420-H40-HR)*

# *Assignment 2*

Date assigned: Wednesday, February 28th

Date due: Tuesday, April 2nd @ 11:59 PM

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Advanced PHP All-in-One form for PHP with sessions, encryption, arrays, and file manipulation
* Secure and make robust a PHP web application
* OWASP ZAP – for details around authentication see: https://play.vidyard.com/iErtsKxpwKn4m8iRoovtH9

To do:

This assignment is based on your Assignment 1. Create your assignment 2 folder: *username*\_H40A02 and copy the contents of *username\_*H40A01 to the newly created folder. Include this document in the assignment folder.

Part A - Updating the Application

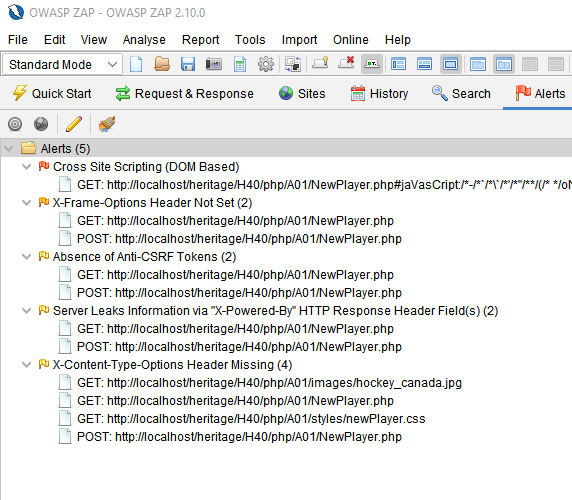
1. Clean up based on Assignment 1 feedback. Ensure the form properly caches values so that if an invalid value is found, the user does not have to re-enter good input—update as per any other feedback on Assignment 1. You must use an all-in-one form for this assignment. You can have a maximum of two separate all-in forms: one for the create user/login flow and then one for the retrieve/add/update/delete player flow, but you cannot have separate views for each function.
2. Add another layer of robustness: ensure you check that the file exists before accessing the file. Your application should be able to handle the file not existing. Either create the file or show an error page.
3. Refactor your project to be more modular and use OOP. You should at least have a class-player.php, a class-leader.php file, and whatever other classes/enums you think would improve your code.
4. Ensure your program is robust – it should be uncrashable and not have any errors or warnings! Use Postman to ensure you are not making any assumptions about data posted or queried for your assignment.

Part B – Add Login/Logout Functionality

1. Create new team leaders/users.
   1. Add a screen to your all-in-one form to create leaders.
   2. The only information you need from your leaders is their email address, name, and password.
   3. The following validation is required:
      1. For the name, follow the validation rules from assignment 1 for the first name. You do not need first and last names in this assignment for users, but you can if you want.
      2. Follow the email validation rules from assignment 1.
      3. The user’s email must be unique, a user cannot be created if the email is used for another user.
      4. Follow the password validation rules from assignment 1.
      5. Get the user to enter the password twice and ensure the passwords match.
      6. You do not need edit functionality for users.
   4. Store the leader data in a separate file called leader-data.txt in a similar format to the player data.
   5. Leader passwords are hashed into the data file.
2. Login/logout functionality.
   1. Ensure leader can only log in if their account already exists and they enter the correct password.
   2. The validation for email and password is no longer required since it has been moved to the create a leader flow – ensure those error messages are not present.
   3. On successful login, the main player screen (like in Assignment 1) is shown (List of Players). In the header of the web page include the name of the currently logged-in leader, the time logged in, and a logout link.
   4. The leader can only see the players that they added.
   5. The leader can only edit or delete players that they added.
   6. If a leader is not logged in, they only see the login or create user screen.
3. Implementation details/requirements
   1. The state of whether a leader is logged in/out is held in PHP sessions.

Part C – Securing your site

1. Run OWASP ZAP’s automated scan attack on your original Assignment 1. **Include the Alerts reports (like below). Ensure when you run it you have players created so that ZAP can test all functionality.**

Sample from Assignment 01: 

1. Using the Alert Reports and **the techniques learned in class**, plan out what changes you will make to your application.

|  |  |
| --- | --- |
| Vulnerability | Planned change |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Update your code as planned (if necessary), making it more secure. You must also defend against all the vulnerabilities taught to you.
2. Run OWASP ZAP against your more secure application. Include the Alerts below:

<answer here>

1. Explain any remaining alerts. Is it a concern that should be addressed? Why or why not?

<answer here>

**Marking Scheme**

|  |  |  |
| --- | --- | --- |
| **Part A/B - Advanced PHP** |  |  |
| Project Refactor |  |  |
| Using OOP |  | 10 |
| Checking file |  | 2 |
| Using all-in-one form |  | 10 |
| Login/Logout functionality |  |  |
| Create new user/store in file/associated to player |  | 10 |
| Login works as expected |  | 10 |
| Proper use of PHP hash |  | 7 |
| Password management |  | 7 |
|  |  |  |
| **Part C** |  |  |
| ZAP test results – A01 |  | 4 |
| Defense Plan |  | 6 |
| ZAP test results after the defense plan |  | 4 |
| Secure and Robust |  | 6 |
|  |  |  |
|  |  |  |
| **Assignment Organization** |  |  |
| Site Design |  | 10 |
| Code Design/Efficiency/standards |  | 10 |
|  |  |  |
| Organization - Files named correctly; meaningful variable, class, object, method names; code formatting; correct submission to Moodle. |  | 4 |

**To submit**

When you have completed the assignment, zip the files as YourUserName\_H40A02 (including this document) and save the zip file to Moodle.