Lab Homework 3: Filter Input, Escape Output

Security. Without the proper precautionary measures, it's very easy for some malicious person to *hack* your web site. This lab will help you develop your *filter input, escape output* skills so that you can protect the web sites that you develop.

1. Learning Objectives

- Practice critiquing an existing website's design.
- Design and code forms for user experience and security.
- Practice the *filter input, escape output* approach for securing yours site's data input and output.
- Develop your *look-up reference documentation* skills in order figure out how to use PHP functions like `filter_input()` or `filter_var()`.
- Practice debugging using the PHP server console.

2. Deadline

Lab Homework	Deadline	Slip Days	Credit	Solution
All Parts	Sun 2/16, 11:59pm	Max: 2 days	20 points (completion)	Provided

3. Instructions

1. Clone your lab repository.

   Clone the following URL:
   `git@github.coecis.cornell.edu:info2300-2020sp/YOUR_GITHUB_USERNAME-lab03.git`

2. **Sign the attendance sheet.**

   It is your responsibility to sign the attendance sheet before you leave. If it wasn't handed to you, **ask to sign it**. No signature on the attendance sheet will result in an immediate 0 for this lab's attendance grade. Forgetting to sign the attendance sheet will not be considered a valid excuse to have your lab attendance corrected.

3. Work together.

   Work with your peers to complete this lab. Talk aloud, discuss, troubleshoot problems. **You are encouraged to work together so long as you do your own work and you don't give away answers.**

4. Submit.

   When you're finished, follow the instructions in **submit.md** to submit your assignment.

4. Warning & Frequently Asked Questions

In this lab you will focus on *filtering input, and escaping output*. Due to this focus, the lab's code is implemented without sticky values, form validation, or corrective feedback. This is **not** considered best practice. We have remove sticky values, form validation, and corrective feedback to help you focus on practicing *filtering input and escaping output*. You are still expected to follow best practice in other assignments.

- Do I need to make the insurance form sticky?

  If the write-up doesn't say anything about that, then you can assume that you are not required to do it.

  TL;DR: No!

- Do I need to implement corrective feedback?

  The lab doesn't ask you to do that.

  TL;DR: No!

- I am suppose to implement form validation?

  No. See the question above.

- My forms submits even when the data is wrong. Do I have to fix this?

  No. There is no form validation or feedback in this lab. When you click the submit button, it should show the confirmation *regardless* of whether the form was completed correctly.

  TL;DR: No!

- But there are blank values in the confirmation page if I don't enter any data. Is that okay?

  Yes. The confirmation page should echo the *filtered* values as submitted by the user. Some of these may be blank.

- When I open insurance.php all I see is a blank page in my web browser.

  Please read the lab write-up. This is intentional.

  TL;DR: Debug the code, fix the syntax error, reload the page in the web browser.

- Do I need to *filter input, escape output*, validate form data (server-side), and provide corrective feedback for Project 1.

  Yes!

Part I: Design Critique

Your client, 2300 Insurance, has contracted with you to help them improve their insurance enrollment application. After some internal research they've discovered that their current enrollment process is costing them extra money in lost sales and in additional call center support as a result of the poor user experience. To reduce their overhead costs, they've asked you to help them fix their current enrollment process by addressing the usability and security issues.

From their internal research they've discovered two major issues that need to be addressed:

1. Their current enrollment form has received many complaints from customers that their form is frustrating to use.
2. The current form permits malicious users to send dubious input to the server. The company's security team is concerned they are not doing enough to protect their customers and their systems from attack.

Objective: Your task is to improve the application process for the customers as well as address the security concerns.

1. The Enrollment Form

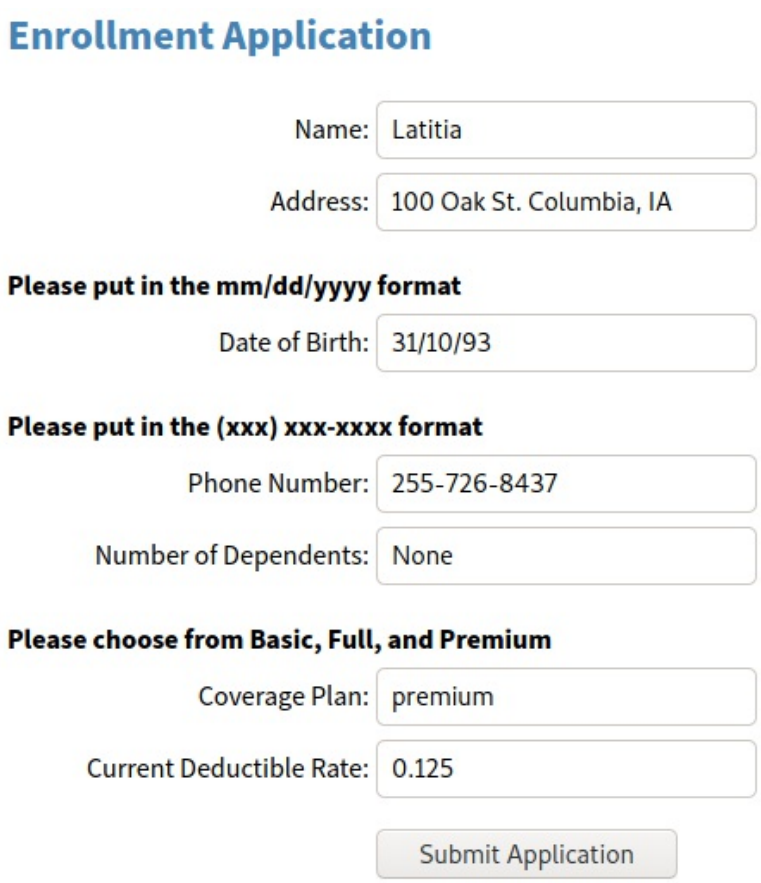
Below is a sample of a typical customer:

Letitia Wright
October 31, 1993
(255) 726-8437

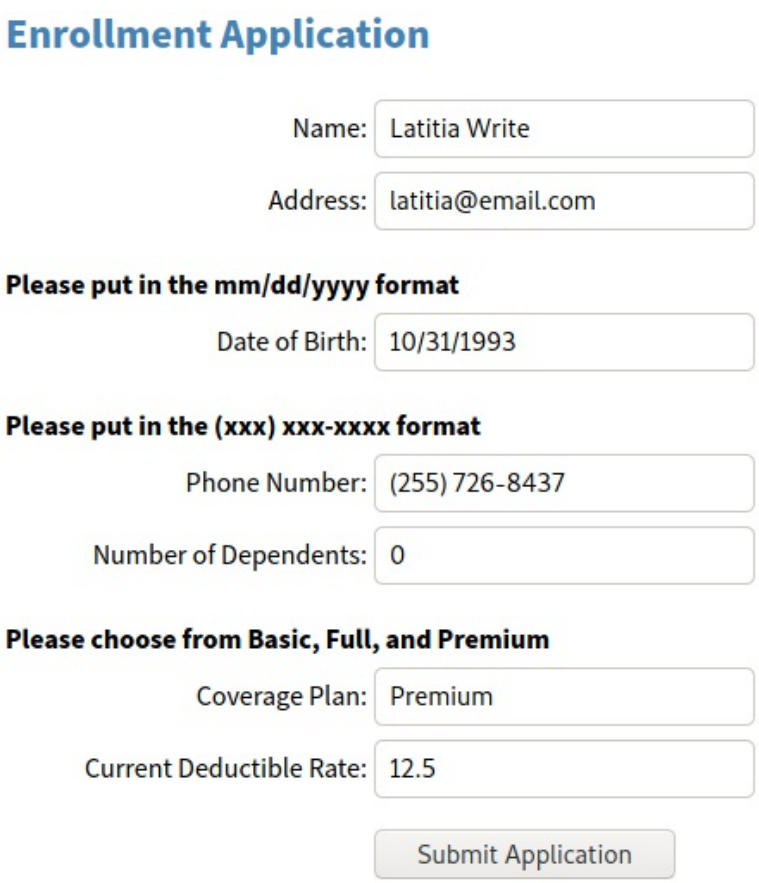
No Dependents
Looking for Premium Coverage
Current Deductable: 12.5%

(data faked; image source: http://guru.bafta.org/60-seconds-with-letitia-wright)

Here's an **incomplete/incorrect** application:



Here is an example of the **correctly** completed application:



Next, view insurance.php in VS Code. **Do not try and view insurance.php in your web browser!** Look over the application's code to identify possible causes for the issues described by the client. For example, take a look at the input types.

2. Design Critique & Refined Design

Objective: Form groups of about 4 people. Critique the form. Once you've identified the usability issues with the form, go to the board and design a new form (sketch).

Once all teams are done redesigning the form, as a class, share your redesign with the class.

Part II: Debugging & Client-Side Usability

Objective: Start the PHP and go to https://localhost:3000/insurance.php. You'll see a blank page. Your task is to figure out why this page is blank instead of showing rendered content.

Hint: Check the PHP server console in VS Code.

Objective: Once you've figured out the issue, correct the code so that the page loads in the browser successfully.

Objective: Using your team's redesigned enrollment form, fix the enrollment form. You will mostly likely need to change the type of the input for some of the fields in order to improve the form's client-side usability. You may also want to add or remove instructions in the form.

Part III: Filter Input

Objective: Next, you'll need to make sure you access the new HTTP parameters and filter them. You may do this using the request associative arrays (`$_GET[]` , `$_POST[]`) or the `filter_input()` method.

Note: Filtering for some of the fields (Date of Birth and Phone Number) may be very difficult based on what you currently know. If you can't figure it out, that's okay, just use `FILTER_SANITIZE_STRING` for now. You need to understand regular expressions to do this properly and we haven't covered those yet!

1. PHP Documentation

- Data Filtering: https://secure.php.net/manual/en/book.filter.php
  - `filter_var()` http://php.net/manual/en/function.filter-var.php

    ```
    $roses = $_POST['roses'];
    $roses = filter_var($roses, FILTER_VALIDATE_INT);
    ```

  - `filter_input()` http://php.net/manual/en/function.filter-input.php

    ```
    $roses = filter_input(INPUT_POST, 'roses', FILTER_VALIDATE_INT);
    ```

- Types of filters
  - Validate filters: http://php.net/manual/en/filter.filters.validate.php
  - Sanitize filters: http://php.net/manual/en/filter.filters.sanitize.php
- Variable handling Functions: http://php.net/manual/en/ref.var.php
- Regular Expressions: http://php.net/manual/en/function.preg-match.php

  Note: We do not expect you to know this or know how to properly use regular expressions. This may be very difficult, but if you want to try it... please go ahead!

2. Examples

```
// Sanitize: Strip HTML tags from client's first name.
$client_first_name = filter_input(INPUT_POST, "first_name", FILTER_SANITIZE_STRING);
```

```
// Transform: Convert dependents value to an integer.
$client_dependents = filter_input(INPUT_POST, "dependents", FILTER_VALIDATE_INT);
// Validate: Number of dependents must be greater than 0 (no negative values).
if ($client_dependents < 0) {
  $client_dependents = NULL;
}
```

Part IV: Escape Output

Objective: Now that you've probably filtered your input, it's time to make sure you *escape your output* for any *untrusted* values. You can escape your output for HTML by using `htmlspecialchars($name);`.

PHP Documentation: `htmlspecialchars()` : http://php.net/manual/en/function.htmlspecialchars.php

Part V: Test it!

Testing is really important. It means that you've checked that your program (or website) behaves the way you (and your client) expect. Testing is especially critical for ensuring that you've properly secured your website too. It is really easy to forget to secure a single parameter, but forgetting this one parameter may be enough to open your web site up for attacks!

Take the time to test that all inputs are filtered properly and all outputs are properly escaped:

- Try filling out the form as the profile we had given you earlier.
- Try injecting some code as demonstrated in Tuesday's lecture.

  Hint: Use the style tag to change the color to a bright red or yellow or make the font very large.

  ```
  <style>body { background-color: red; }</style>
  ```

Contributors

The following individuals made contributions to this assignment.

- Kyle Harms
- Sharon Jeong