Lab 3 Part 2

Sanitisation and Validation

Overview

In this lab, we will look at:

- Sanitsation and validation in Node.js and Express with the **express-validator** and **express-sanitiser** modules

Tasks

Task 1: Prepare your environment

1. If you are working on your local machine (highly recommended!) you can continue to the next task.

2. If you are working directly on the virtual server (not recommended!) you will need to stop your previous application in order to complete this lab.

```
forever stopall
```

Task 2: Tag your work from Lab 3.1 (optional)

We will be continuing on from the work we did in Lab 2. You don't need to create a new directory or Git repository.

3. If you want, git tag your current version of the Berties Books application (see instructions in lab 3.1)

Task 3: Install validator module and import it

4. In order to do validation install the 'express-validator' module:

```
npm install express-validator
```

5. Import this module in your index.js file:

```
var validator = require ('express-validator');
```

Task 4: Validate email

Your register page should currently look something like this, with a field for the user to input their email address:

6. Add the following piece of code to your main.js file:

```
const { check, validationResult } = require('express-validator');
```

Please note, the above piece of code should be added at the beginning of your main.js so all the routes in main.js can access it.

Then update the first line of code in the registered' route, as follows:

```
    app.post('/registered', [check('email').isEmail()], function (req, res) {
        const errors = validationResult(req);
        if (!errors.isEmpty()) {
            res.redirect('./register'); }
        else {
        REST OF YOUR CODE
        }
```

As you can see above isEmail() is a function of the express-validator module that validates a form input as an email address.

In this way you are forcing the user to enter a correct email input, otherwise, you redirect the user back to the register page.

7. Run and test your web application:

- Enter a value for username and password and email.
- You should be redirected to this page again if the email is not a valid email input.

Task 5: Add more validation to your registration page

There are other functions available in the express-validator module such isEmpty(), isInt(), isLength() and more.

8. Read the documentation for the 'express-validator' module and add server-side validation check to make sure password length on the register page is at least 8 characters long.

9. Complete other server-side validations on your registration page that you think are appropriate.

Task 6: Extension: Add validation to other pages

10. Look through your other pages in your app and add other validations you feel would help.

Marks will be given for exhibiting a range of techniques over repeating the same technique.

11. Briefly document what validations you have done on what fields and forms, including the relevant code snippet for each.

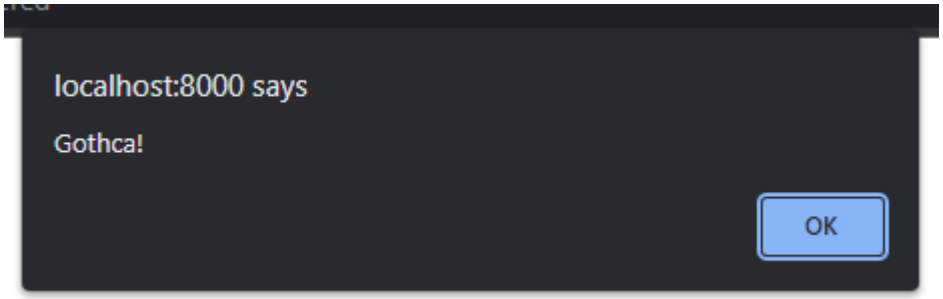
Task 7: Demonstrate Cross-Site Scripting

12. First, test your code to demonstrate that your code is vulnerable to Cross-Site Scripting (XSS) attacks.

Go to your register page and fill in the form, entering the following text into the First name field:

```
Henry <script>alert("Gothca!")</script>
```

Press Register. You will see that the hacker has caused javascript code to run on your browser:



Hello Henry Hacksmith you are now registered! We will send an email to you at hh@evil.comYour password is: catchme and your hashed password is: $2b$10$rz1zsFrlLZ3m9TbyKBjx2.kPHpZa3tIjHpOU68eMyx9UkbBwhvHzy

Note that this will only happen if you show the first name on your registered page.

13. How did this happen?! Check your request/response route and confirm you understand how.

Task 8: Add sanitisation to your code

14. In order to do sanitisation install the 'express-sanitizer' module:

```
npm install express-sanitizer
```

15. Then require it in your index.js file:

```
const expressSanitizer = require('express-sanitizer');
```

also, add the following line of code to your index.js file:

```
// Create an input sanitizer
app.use(expressSanitizer());
```

16. Find the places in your code where the first name is extracted from the request body. Wrap them in a call to req.sanitize:

```
req.sanitize(req.body.first)
```

17. Now check if the XSS attack from the previous task still works.

Research what this sanitize() function is doing to protect your site.

Task 9: Protect other fields on the register form from XSS attacks

18. In the previous task we protected the First Name field from XSS exploits. What other input fields on this form need protection?

Add sanitsation to the other input fields in the register form as required.

Task 10: Extension: Protect other fields in other forms from XSS attacks

19. Add sanitisation to other forms in your application as required.

20. Briefly document what sanitisations you have done on what fields and forms, including the relevant code snippet for each.

Task 11: Make your web application ready for submission

21. Push your code to GitLab and give Developers permission to me and the lab assistants (see the submission page for details)

22. If you have done the extension tasks, make sure you have documented each validation and sanitisation you have done. You will add this document to your submission on the VLE.

23. Make sure your code is nicely formatted and commented.

24. Make sure your web app looks neat. There is no need for fancy CSS styling, but the layout should not be messy, there should not be spelling mistakes, etc.

Task 12: Make your submission

After successful completion of all tasks in lab 3 part 1 and 2, your app is ready to be marked.
Remember to give permission to me and lab assistants to access your GitLab repo.

Then run your web application with **'forever'** and submit TWO URLs on the lab 3 assessment page:

1. URL of your git repo,
   example: https://gitlab.doc.gold.ac.uk/username/projectname
2. URL of your web server: www.doc.gold.ac.uk/usr/ID/

YOU WILL ONLY BE MARKED IF BOTH URLs ARE SUBMITTED and working.

The rubric for this lab submission is as follows:

• All pages link from/to home page		5 marks
• List books page access control	Lab 3.1 Task 5	5 marks
• Logout page	Lab 3.1 Task 6	5 marks
• Search page access control	Lab 3.1 Task 7	5 marks
• Add book page access control	Lab 3.1 Task 7	5 marks
• Bargain books page access control	Lab 3.1 Task 7	5 marks
• List users page access control	Lab 3.1 Task 7	5 marks
• Delete user page access control	Lab 3.1 Task 7	5 marks
• Validation of registration form	Lab 3.2 Task 4,5	10 marks
• Extension: validation of other forms	Lab 3.2 Task 6	15 marks
• Sanitisation of registration form	Lab 3.2 Task 8,9	10 marks
• Extension: Sanitisation of other forms	Lab 3.2 Task 10	15 marks
• Nice, tidy code with comments		5 marks
• Clean and tidy user interface		5 marks

END