Laboratory assignment 3:
Web Application Security

For all the assignments in this course, you are expected to work home and only book a lab slot if you are stuck and require a TA's assistance or you are required to demonstrate your solution.

1 Purpose

In this assignment, you will investigate the impact of two of the top ten most critical web application security risks. Unfortunately, there are lots of websites that are vulnerable to those kinds of security issues. This lab will help you learn how to develop more secure web applications.

2 Reporting

To pass this assignment, you need to work through the exercises and write down your answers to the questions. When you are finished, call for your supervisor to discuss your answers.

3 Preparation at home

Read the material as specified bellow. Then, read the rest of this assignment before you start your work.

3.1 Reading

- Section 11.2, Handling Program Input, in course book.
- The OWASP Top 10 Report: https://owasp.org/Top10/

3.2 Optional Reading

- SQL Tutorial; Section 3, 5–7: http://www.sqlcourse.com
- OWASP ZAP – Getting Started: https://www.zaproxy.org/getting-started/
- Sqlmap user's manual https://github.com/sqlmapproject/sqlmap/wiki

- Web Security Academy by PortSwigger: https://portswigger.net/web-security
- Web application hacker's handbook by Dafydd Stuttard and Marcus Pinto (Available online at Chalmers' Library).

4 Introduction

In this assignment, you will investigate a web application against two of the top ten most popular security vulnerabilities, namely, SQL Injection, and Cross Site Scripting. You are going to use the web application at http://localhost:55173, which basically provides you a system to login, post comments, purchase products, etc. This application is vulnerable to the two attacks mentioned earlier. You need to work through the tasks and exploit those vulnerabilities and try to breach the system. Moreover, you are asked to fix the source code for the web application to remove these vulnerabilities, such as the SQL-Injection vulnerability.

This assignment is performed in with a terminal with npm installed. You will use Firefox and/or Chrome as web-browsers. There is also the possibility to use one instance of Firefox in normal mode and one in private mode. To start Firefox in private mode go to File → New private window or press Control+Shift+P. Alternatively run `firefox.exe --new-instance --ProfileManager` to start Firefox with a new profile..

4.1 Preparing your system

You will need to do some preparation of the system for running this lab. The web application needs to be copied to a directory where you have write permission.

4.1.1 Copying the Web Application

All files needed to perform this lab are stored on Canvas. Please download the source code for the web application, named `cs_lab3.zip` on Canvas. Before you start, you need to unzip the file to a destination you can write like %TMP%.

The files must be placed in a place where you and IIS can write to ensure that the web application works.

4.2 Running the Web Application

When the terminal is open, change your directory to the one containing the lab files you earlier unzipped. To run, write `npm install` to install the dependencies for the project. Then, write `npm run dev` to launch the project. You should be able to open http://localhost:55173 at this point, it might take a bit of time to launch.

In Task 1, you will login to the web application. After you log in, you can do several activities such as viewing the comments that have been posted earlier, posting comments, and uploading images. You need to investigate where and how you can breach the system with the vulnerabilities mentioned, and get control of the application. Afterwards, you are asked to fix the code so that it is impossible to perform SQL Injection against the web application.

5 Exercises

You will now look at two problems with this web application. First, you will investigate SQL injection against the application. Then, you will look at a cross-site scripting problem. In the last part, you will also see some problems with the cookies.

5.1 Exercise 1: SQL Injection

The number-one security risk, among the top ten according to OWASP, are injection vulnerabilities. In the web application, you are now going to investigate the problem of SQL injection.

One of the critical steps in a web application is the process of authentication. Usually all account information is stored in a database, and when the user authenticates, a query is sent to the database for validating the username and password. This web application is also using a database as back-end storage.

Now, please look at the problems around SQL injection in the OWASP Top 10 Report. Then, look at the following tasks.

Make sure the web application is running when performing these tasks.

Task 1: Unauthorized Login
In this task, you will look at the problem of *unauthorized access* to the web application. First, start *Firefox* and browse to the web application at `http://localhost:55173`. This login page is vulnerable to SQL injection. Now, keeping in mind the issues described about SQL injection (in OWASP Top 10), try to login to the web application. What did you write to login? What is the problem? Why can you circumvent the authentication system?

Hint: A comment in SQL is written as "--".

Task 2: Unauthorized Modification

Based on the fact that you can login to the system, what else can be done to the database? Can you create a new account without having administrative privileges? If you could, how was this accomplished?

Hint: What pages on the website has user input?
Hint: What can the `INSERT` SQL statement be used for?

As you see, the problem of SQL injection (together with a misconfigured webserver) can lead to a multitude of security-related problems. Some suggestions on how to protect the web application against SQL injections are given in the OWASP Top 10 Report.

Task 3: Preventing SQL Injection
Using the information you gained in the previous tasks, you should now find the corresponding SQL statement in the web application and update the code so that this type of threats is eliminated. What is making this SQL code vulnerable? How would you fix this vulnerability?

5.2 Exercise 2: Cross-Site Scripting

Cross-site scripting is the number seven problem in the Top 10 OWASP list. In this exercise, we will demonstrate the problems around cross-site scripting, and the potential unwanted outcomes of such an attack.

In the web application, a cross-site scripting vulnerability has been introduced into the comment form. We are now going to demonstrate the vulnerability with this comment form.

Task 4: Demonstrating Cross-Site Scripting
This demonstration is shown by having one "attacker" and one "normal user".

Now, make sure that the web application is running. While progressing through this task, please have the following questions in mind:

- Can you suggest any security improvements?
- How can these types of vulnerabilities be prevented?

Then, do the following steps:

Step	"attacker"	"normal user"
1	Start `Firefox` and browse to the first page at `http://localhost:55173`.	
2	Open up another terminal and run `python -m http.server 55555`.	
3	The vulnerability is located in the comment form. Press the comments link at the navigation bar. Then, paste the attack code, which you find in the file `Demo\Attack.txt` in the project, into the comment form.	
4		Now, Start the `Chrome` web browser and *login* to the web application at `http://localhost:55173`.

Step	"attacker"	"normal user"
5		Go to the page "comments". Your browser has now been infected by the cross-site script. Check the terminal running the python server. What did you find? What is the vulnerability in the web application that could make this possible? (Make sure you do not logout here.)
6	The attacker now has access to the same `cookie` in python. Copy the data to your clipboard. Open the cookie manager in `Firefox` by `Ctrl+Shift+i` and then going to storage. Add the cookie by, filling in the form. The name should be `sessionid` and in the value, you should paste the authentication id you copied. Also, make sure that the hostname is just `localhost` without any port number, and that the session box is ticked. Now browse to `http://localhost:55173`. What happened? What access do you now have?	
7	Now, we are going to upload a file. Click on the Upload File, and upload the file `+page.svelte` to the path `../../src/routes/backdoor/+page` and +page.server.ts to the path `../../src/routes/backdoor/+page.server`, which you will also find under the `demo` directory in the project.	
8	Now, browse to `http://localhost:55173/backdoor`. Try to run a command, e.g., `dir` or `ls`. What happened?	
9		Log out from the web application in Chrome.

Note that you do not necessarily need to use two browsers. You could use one browser (Firefox or Google Chrome) and open one window in normal *mode and one in* incognito *mode.*

Based on this demonstration, can you suggest any security improvements? How can these types of vulnerabilities be prevented?

6 Approval

It is now time to call the TA and discuss your answers.