CSCI 311 Spring 2020: Lab 6

Learning Objectives:

- Identify and correct basic web security risks, including:
  - sql injection
  - Cross Site Scripting (XSS)
  - Lack of Validation
  - Lack of Sanitization

What to hand in:

- submit the following files to VIU Learn no later than March 14, 18:00:
  - Lab6PenTest.pdf
  - Zip Together fixed copy of:
    - BadSite (which should now be good(er), not bad ☺)
- ensure the appropriate files are correctly located on a server where they can be viewed/accessed by anyone (I will test your fixes on the Apache server)
- late submissions will be penalized 20% per day

All work must be individual.
Plagiarized work will result in a mark of 0. Further penalties may apply.

Marking Scheme:

- Errors found: up to 5 marks
- Documentation process: up to 5 marks
- Followed instructions: 2 marks

Instructions:

1. Set up the given (bad) web pages and database

Set up the Database

- open a connection to mysql on the command line
- use your lab database (USERNAME_labs)
- use the dbSetup.sql file to create and initialize the tables
- check that the following tables were created, and that they contain data:
  - books
  - authors
  - publishers
  - members

Set up the php pages and test them

- Copy the given BadSite directory, and its contents to your Lab6 directory.
- Fix the permissions of the directories and files in order to be able to serve the web pages. This includes the files in the bootstrap directory (which I've included for your convenience)
- Create and set up your dbinfo.inc file so that it is of the form:

```
<?php
     $servername = "wwwstu.csci.viu.ca";
```

```
$username = "username";//this is your mysql username
$password = "password";//your mysql password
$database = "username";//this is the name of your lab database,

?>
```

- Open Lab6.php in your browser, and try all the pages to ensure they are working correctly. Try adding values to the database. Try selecting an author and seeing what books they've written.

2. Document and fix five security vulnerabilities:

You will create a pen-test (penetration test) document which will be submitted with your lab. Undocumented fixes will not be counted. Each issue should be documented as follows:

For this step, you will write a formal document that identifies five ***distinct*** security vulnerabilities.

- Write a brief summary, including:
  - challenges you faced
  - what you learned
- For each vulnerability, your document will have the following format:
  - Name the **type** of security issue
  - Identify the specific **file(s) and line(s)** of code that cause the risk. Make sure these are the *original line numbers in the given code*. As you fix things, these lines will change, so it is important to document the process carefully.
  - Describe how to **fix** the problem
  - Describe the **steps** you used to exploit the vulnerability, and what happened after it was exploited.
  - **Note how you** the problem in the given code, and document exactly what was changed. To do this put comments in the code that indicate:
    - //start of fix for vulnerability XX
    - //end of fix for vulnerability XX
    - In some cases you may have to change code in multiple locations. Mark all such locations.

Format of pen-test documentation

Security Type:
- Name the vulnerability that was identified

Location:
- File name
  - List of line numbers involved
- File name
  - List of line numbers involved

Fix:
- Name the technique used to fix the vulnerability

Steps to identify:
- Detail what steps you took to *find* the vulnerability

Detailed steps to fix:
- Detail *all* steps you took to fix the vulnerability

Example:

Security Type:
- password stored as plaintext in database

Location:
- File abc.php
  - lines 33-35
  - lines 57-58
- File xyx.php
  - line 3

Fix:
- hash password before saving it to the database

Steps to identify:
- Looked through the php file that created the query string to store the database and saw that the value was not encrypted before being put in the mysql query

Detailed steps to fix:
- create a hash of the password using blowfish algorithm run 10 times
- use a pseudo random salt to prevent rainbow table attack
- store hashed password in db
- to validate password, hash password attempt with stored and hashed password
- In addition to lines identified above, the following functions were added to file abc.php:
  - password_check
    - input: plaintext password, hashed password
    - returns true if plaintext hashes to same hash
  - password_encrypt
    - ... and so forth

- **Hint**: Look through the code and try to identify where there are security problems. Remember to check the following places:
  - where user input is being written into a database
    - where users can input data that might be in an invalid format
    - where user's input may be in a valid format but may still be harmful
  - where database entries are used to generate or build html
  - where queries are formed
  - where users can intercept and modify data being sent to the database
  - where secure information (passwords etc) are stored in close proximity to safe information
- For full marks you must find **5 distinct security issues**. They must be distinct.
  - **For example, you may only count one Cross Site Scripting (XSS) vulnerability** even though there may be multiple vectors for this attack**.**
- **Note:** if you break the database (and you should try!!) you can use the given .sql file to reset the database. Just uncomment out the lines to drop the tables.