

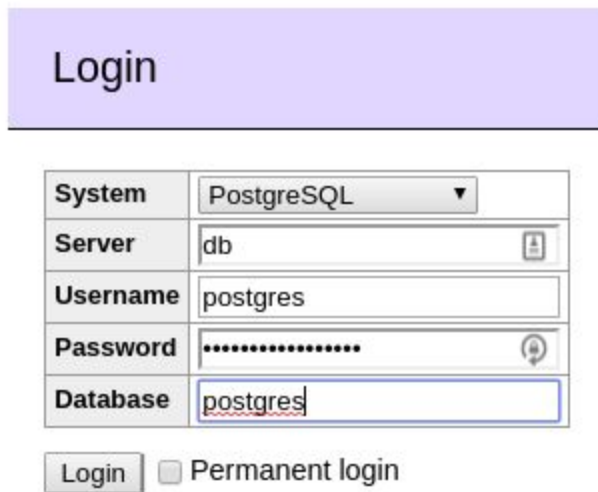
PROG38263 - Assignment 2

Fixing and Insecure PHP Blog Application - 7.5%

Description

For this assignment you will work alone or with a classmate to identify and exploit vulnerabilities in an insecure blog application. After that you will add and test security controls to mitigate or prevent the vulnerabilities from being exploited.

The web application's technology stack is PHP, Nginx and PostgreSQL. The application is provided via Github and can be built/launched with `docker-compose build` & `docker-compose up`. For convenience, there is also a container provided for exploring the database. It can be accessed by navigating to the application in a web browser on port 8080. Use the credentials as shown in figure 1 to login to the database.



The screenshot shows a web interface for logging into a database. At the top is a purple box with the word "Login". Below it is a form with the following fields: "System" (a dropdown menu set to "PostgreSQL"), "Server" (a text input with "db"), "Username" (a text input with "postgres"), "Password" (a text input with masked characters "....." and a toggle icon), and "Database" (a text input with "postgres"). At the bottom of the form are a "Login" button and a checkbox labeled "Permanent login".

Figure 1 - Database Web Login

Vulnerabilities

The application has the following known vulnerabilities that you must find and exploit.

1. Cross-Site Scripting (XSS)
2. SQL Injection
3. Broken Access Control (i.e. you can do things without authenticating that you should be able to do)
4. Missing role-based access control enforcement and management (i.e. Blog authors should only be able to delete their posts. Admins can delete anyone's posts. There is currently no mechanism for changing or managing the roles for users. Actually, there is no mechanism for even creating or managing users without issuing manual SQL against the database).
5. Insecure password handling and storage.
6. CSRF
7. The entire website, including the login page, is served over plaintext HTTP.
8. The web application has no logging except for the default logs generated by Nginx.

9. The website only uses single-factor authentication.

Objectives

1. For the vulnerabilities described above, find and exploit at least one instance of that vulnerability.
2. Provide proof of your exploit through screenshots or tool output.
3. Write a sentence describing the severity and impact of the vulnerability.
4. Implement the most appropriate security control for that vulnerability.

Evaluation & Grading

- 2 marks for each vulnerability: 1 mark for identifying/exploiting and providing proof. 1 mark for the description of the severity and potential impact of the vulnerability.
- 2 marks for each successfully implemented and tested security control: 1 mark for the control implementation, 1 mark for testing the control (tests should be sufficiently robust).
- 2 marks for using proper version control practices (i.e. git on either Github, Bitbucket, Gitlab, etc) to regularly commit and save your work. If you're working with a partner your changes must be merged regularly into the master branch.
- Total: 20 marks.

Bonus Mark!

- Implement database caching using memcached. 1 mark.
- Implement persistent session management using redis. 1 mark.
- Allow Articles to be written using either some simple HTML formatting tags (like <h1>, , etc) or Markdown but still prevent the use of malicious JavaScript in articles. 1 mark.

Deliverable

Upload a complete copy of your source code to the SLATE dropbox. In addition, add your instructor as a contributor to the git project so they can review the commits and commit messages.

Due Date

See the dropbox on SLATE for due date details.