Design and Development of Secure Software
Marco Vieira, mvieira@dei.uc.pt
Nuno Antunes, nmsa@dei.uc.pt

(2022/11/07: v1: Complete assignment description. Minor changes are possible)

Assignment #2: Secure Coding and Vulnerability Detection

Due: Dec. 12th @ 23:59h (soft deadline)

OBJECTIVE:

Understand the relevance and applicability of secure coding practices in the context of Web applications, and the relevance and applicability of testing and static analysis in the detection of vulnerabilities in the Web context.

READINGS:

- OWASP, "OWASP Secure Coding Practices – Quick Reference Guide v2.0", The OWASP Foundation, 2010.
- David H. Hovemeyer, William W. Pugh "*FindBugs™ Manual*", University of Maryland, 2012.
- N. Antunes and M. Vieira, "Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 269–283, 2015.

PREPARATION:

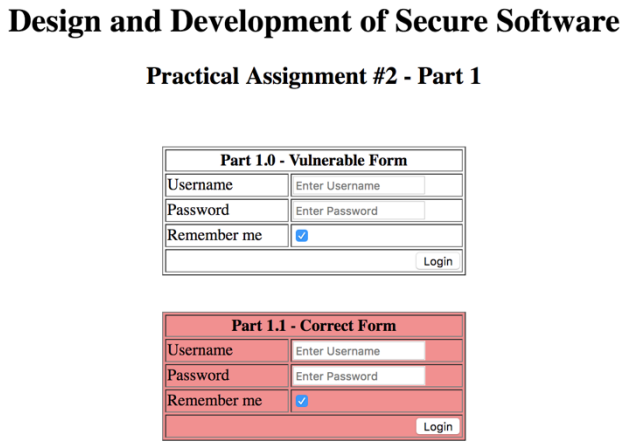
Before starting the exercise, you should read carefully this assignment and understand what tasks are to be conducted and the goals to be achieved. This will allow you to better select not only the programming environment and language that you will use, but also to the vulnerability detection tools that you will use on the code you will develop. It is of your best interest to make sure that the tools you decide to use are applicable to the type of applications you are developing.

Although the functionalities of the assignment are described as a web application, you can opt to develop them in the form of a Microservice if you are more familiarized or prefer to explore those technologies.

EXERCISE 1:

The goal of this exercise is to implement three simple functionalities of a Web application, including two versions for each functionality: one correctly implemented from a security perspective (i.e. applying good security practices), and the other having software vulnerabilities (SQL Injection, XSS, CSRF etc.). The functionalities to be implemented are as follows.

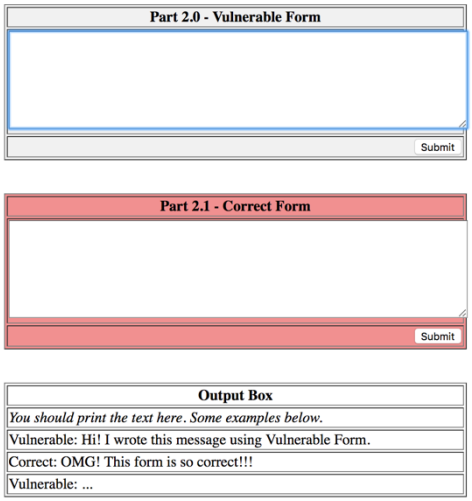
Part 1.1: The first functionality consists of a typical authentication form, based on a username and a password. The picture on the right shows the intended interface. The form in the top should be vulnerable, while the one in the bottom should be secure (you may decide to implement the functionality in different Web pages or in the same page). The output should include a message of success or failure.
You are responsible to select the scheme of authentication and to register the users according to the best practices to manage passwords. *If you create a new interface for that, you are not required to implement the vulnerable form.*



The two functionalities below should become available only after the authentication.

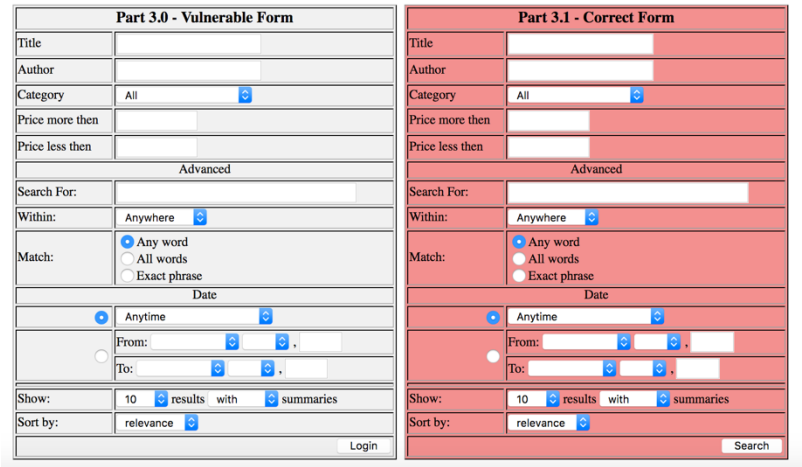
Part 1.2: The second functionality is a simple form where the user may introduce some text that is sent to the server to be stored in the database and then displayed back in the bottom of the page (as shown in the next figure). As before, the form in the top should be vulnerable, while the one in the bottom should be secure (again, the two may be implemented in different Web pages or in the same one), and the functionality is available only after a successful authentication.

Practical Assignment #2 - Part 2



Part 1.3: Finally, the last functionality consists of an advanced search form, where the user may fill different fields to define the search that will be done. The search should consider the information provided by the user and give correct answers. In practice, the output is the list of books that satisfy the search conditions.

Practical Assignment #2 - Part 3



Note that the final solution should include a database (see resources below) and a Web server. The user interacts with the system via a Web application. The types of vulnerabilities in the vulnerable versions are of your decision, but diversity is valued.

EXERCISE 2:

The goal of this exercise is to use testing and static analysis to detected vulnerabilities in the code of the applications you developed. The activities proposed are divided in three parts.

Part 2.1: Select the at least two vulnerability detection tools to use: at least one based on testing (e.g. a security scanner such as Acunetix WVS), and at least one tool based on static code analysis (e.g. SpotBugs). Make sure that the selected tools are adequate for the type of applications you will be developing.

Part 2.2: Use the selected tools to detect vulnerabilities in all the web pages developed during Exercise 1. In some cases, the testing tool is not able to test the code because the input parameters have very specific information domains, therefore you should try to help the tool in generating better tests by providing this information in the scan configuration (e.g. use numerical values in price fields, etc.). As a complement, you should also analyze the code and identify one vulnerability not reported by the testing tool. For that vulnerability, provide a ready to use exploit (if possible, including proof of concept code).

Part 2.3: Analyze the vulnerabilities reported and classify them in terms of true positives and false positives for the used tools. Understand the reported false positives and the vulnerabilities that were left undetected, and explain the reasons behind these incorrect classifications. Discuss and compare the tools, with special focus on the strengths and limitations.

Optional: Using more than two tools in your assessment, if done correctly, is valued.

RESOURCES:

The following resources are provided to support the development:

- A docker container with a PostgreSQL database with 3 tables, one for each of the exercises
- HTML prototypes of the interfaces
- Simple demos for java and php on how to use the PostgreSQL database, already integrated into docker containers that you can reuse.
- GitHub repository with folders for Java, Python, php, and NodeJs development, already with configured projects to ease the process.
    - Available at: ddss-a2 (we will create a fork for you using Github Classroom)

      It is mandatory to use git for your assignment.

PROGRAMING LANGUAGE / FRAMEWORK:

You may choose the programing language and/or framework to be used to develop the work, considering the factors described in the assignment.

DELIVER:

The source code developed, and a short report of the work conducted. The report should explain the work done to allow an adequate assessment of the work developed. The recommendation is to organize the report following the steps proposed for each part. The source code should be maintained in the github.com repository provided. Good sources management are essential for code quality and security!

The source code shall include means to automatize build and deployment, such as a Docker container or a tool like ant, maven, phing, or equivalent (depending on the used language). It is particularly relevant to list the good practices used in each of the correct implementations and list the vulnerabilities that exist in each of the vulnerable versions. For the listed vulnerabilities, include a description of an exploit and, when possible, a proof of concept code.

Your report should also include the reasoning behind the selection of the vulnerability detection tools. It should describe the process and the results of the activities proposed. For the exploit developed in Part 2.2, should describe the vulnerability targeted by the exploit, and explaining that same exploit. The document should list all the vulnerabilities reported by the tools, dividing them in terms of true positives and false positives. Finally, it should also present and explain the analysis of the Part 2.3.

GROUPS:

The assignment is prepared to be executed by 1 student in 40 hours. However, it is possible to develop the exercise in groups of two, with the following changes to the description:

Exercise 1:

- Considering the GitHub repository in use:
  - (a) Each student should make his own commits. **This is mandatory!** This will help assessing the contribution of each one;
  - (b) The final submission should be one pull request to the main branch. **This is mandatory!**
- You should add one extra functionality that allows you to insert data in one of the tables on the database at your choice.
  - (c) Like the other functionalities, you should implement correct and vulnerable versions.
  - (d) Besides other vulnerabilities that you may choose, the vulnerable version should contribute to the existence of a second order vulnerability. Develop the exploit for this vulnerability in your deliverables.

Exercise 2:

- Select 2 additional tools, of any type, to use in the vulnerability detection process.
- Include the results of these tools in the analysis, discussing and comparing their results also;
- Provide one more ready-to-use exploit (two in total).

CHALLENGE:

Enhance your correct version of the login with a two-factor authentication using, for instance, time-based one-time password, email or sms token.

This part is optional, and it is valued as extra points up to 10% of the assignment's grade. This challenge is valid for both singles and groups.

QUESTIONS:

If you have questions about the scope of the work or any other aspect, please talk with the Professors of the course. You can do so face-to-face, by email, or using Skype.