Fix Lab 10: Secure Session Management

Date Due: Tuesday, November 23, 2021 by 11:59 PM

Introduction

In this lab assignment, you will fix all the issues you uncovered in the last attack lab. This assumes you have completed both Lab 0: Lab Setup Guide and Lab 9: Poor Error Handling and Logging. For this lab you will need the ability to run the WebGoat.NET application, as well as debug it using Visual Studio 2019 Community Edition. You will also need to have completed the Git setup outlined in Lab 0 and the ability to push code changes to the remote repository on https://code.umd.edu.

For each of the following questions on the WebGoat.NET application, you will need to refer to your answer from Attack Lab 5, and fix the weaknesses found in the source code. Along with this, you will submit a writeup containing the following:

Question Number.

a) Provide the URL of the WebGoat.NET application page where you are exercising the question. **Do not** include any query parameters or any other special characters in the answer.
b) For the given CWE-ID, Filename, Line Number of the weakness identified in the previous Attack Lab, describe how you intend to fix the weakness. This can be as detailed as possible to explain how it will address the weakness.
c) Describe why your intended fix will address the weakness (either directly or indirectly) and whether to your knowledge it will prevent future attacks along the lines of the attack vector used in the previous lab. This can be as detailed as possible to explain why it will address the weakness.
d) List all the paths with filenames you are changing to implement the fix for the weakness.
e) **IMPORTANT:** Implement the fix for the question under a branch with the following naming convention:
    a. Lab<Lab #>_Phase<phase #>. For example, you can run:
        i. `git checkout main`
        ii. `git checkout -b Lab2_Phase1 # This is for Lab 2 Phase 1 fix`
    b. Now you implement your fix and commit the changes locally:
        i. `git add .`
        ii. `git commit -m "Implemented Lab 2 Phase 1." # Commit with a message`
    c. Repeat earlier step as many times to get your fix working. Commit and push the newly created branch to the remote repository using:
        i. `git push origin Lab2_Phase1`
    d. **Identify** and **submit the commit id** (the long hexadecimal alphanumeric string from `git log --pretty=oneline`) as part of line item 'e' for the given question number in your writeup.
    e. Read https://git-scm.com/book/en/v2/Git-Basics-Viewing-the-Commit-History for more information.

Please NOTE: For the questions below, the word 'discuss' below refers to question items b) and c) from the list at the top of this lab handout and the word 'implement' below refers to question item e).

Important Note: For this lab, you will be implementing fixes based off of the '**ErrorLoggingLab**' **branch** from the instructor's WebGoat.NET repository on code.umd.edu. Please install **log4net** package version **2.0.5** from NuGet Package manager after downloading the branch from the instructor's repository.

Phase 1: Error Handling in Unsafe Blocks

In the attack lab, you saw that unsafe inputs run in unsafe code blocks could cause runtime exceptions and errors to occur in the application.

1. Fix the issue you found in the attack lab for the 'Unsafe Blocks' exercise in the WebGoat .NET application.
2. Answer the given questions at the top of this handout and remember to commit and push your code under the Lab10_Phase1 branch.

Phase 2: Preventing HTTP Verb Tampering

In the attack lab, you saw that certain pages that were either hidden or not accessible to users could still be used to tamper with state in the application.

1. Assuming you wanted to retain the functionality of both the VerbTampering.aspx page that lets say displays messages to the user via the VerbTamperingAttack.aspx page which is used internally by the webserver administrators, how would you go about fixing this issue?
2. Answer the given questions at the top of this handout and remember to commit and push your code under the Lab10_Phase2 branch.

Phase 3: Fixing Error Handling

In the attack lab, you found that the file validation was being done at the cost of a resource leak and thereby performing poor error handling.

1. Now implement a fix for the resource leak and handle the error correctly.
2. Answer the given questions at the top of this handout and remember to commit and push your code under the Lab10_Phase3 branch.

Phase 4: Fixing Sensitive Logging Issues

In the attack lab, you found some sensitive information was being logged in the WebGoat .NET application.

1. Now implement a fix by not including that sensitive information you found was being leaked in the log files! But wait a minute, your Security Operations Center has told you they need to be able to audit the actions being performed in the WebGoat .NET application and still need a way to do that. How would you go about fixing the issues you found?
2. Answer the given questions at the top of this handout and remember to commit and push your code under the Lab10_Phase3 branch.

Submission Criteria

Please submit a writeup with:

1. Your name, UMD email ID and Lab Number.
2. The answers provided in the format given at the top of this lab handout.

Please only submit a **Word document** or a **PDF**. This lab contains code submission with branches for each phase. Please ensure the commits submitted are accessible by the auto grader.