ENPM809W – Introduction Secure Software Engineering

Fix Lab 8: Secure Session Management

Date Due: Tuesday, November 9, 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 7: Poor Session Management – Session attacks. 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 7, 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).

NOTE: You will be using the same '**SessionLab**' branch of WebGoat .NET to answer all these questions.

Phase 1: Preventing Session Fixation

As part of the attack lab, you completely ruined the reputation of the auction site and so they worked out a plea deal where you fix their WebGoat .NET site in order to avoid being attacked in the same way in the future in exchange for your freedom.

1. You decide to first prevent the session fixation problem. You discover it has already been fixed to a degree – namely the Logout feature clears the Session. Is there anything additional that needs to be done? If so, please implement the fixes. Otherwise, proceed to the next question.
2. Please answer the questions at the top of this handout to describe the fix (if any) and why it addresses the session fixation problem.

Phase 2: Secure Session based Authorization

Now that you have fixed the session fixation problem, you find out that the attacks you used permitted users to snoop on Customer Orders of other users even without logging in as the customer to whom the orders belonged.

1. Fix the authorization issue where the session information of the logged in user is used to authorize the customer orders that will be displayed including the list of orders.
2. Please answer the questions at the top of this handout to describe the fix (if any) and why it addresses the authorization problem.

Phase 3: Preventing Cross-Site Request Forgeries (CSRF)

You keep the best attack to fix for last, and are now trying to figure out how you can really fix cross-site request forgery issues. This auction site is really terribly designed because it has pages called StoredXSS.aspx and ReflectedXSS.aspx, which are clearly vulnerable to cross-site scripting attacks. But you realize, those pages are not the root of the problem. People should be allowed to post all kinds of cool scripted comments. Who remembers the 90's animated gif style comments that allowed you to load cool fonts and render your comment in those cool fonts …? Ah! so nostalgic.

Unfortunately, you are also not able to use the latest ASP .NET Core or MVC since you have to implement the fix for the existing WebGoat .NET application. You decide to implement the Synchronizer Token pattern in order to prevent the session based forgeries as outlined in an article by James Jardine (https://web.archive.org/web/20200707191912/http://software-security.sans.org/developer-how-to/developer-guide-csrf). A pdf of this page has also been posted to Week 10 modules section.

1. Implement the fix as outlined in the article.
2. Does the fix invalidate the attack you performed in the previous attack lab? Please answer the questions at the top of this handout to describe the fix (if any) and why it addresses the authorization problem.

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.