Fix Lab 6: Secure Authentication and Authorization

Date Due: Tuesday, October 26, 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 5: Poor Authentication and Authorization. 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 1, 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).

Phase 1: Adding Timeouts to Login

In the attack lab, you saw how attackers were able to brute force the login page. You will setup a timeout to the login page to prevent brute force attacks.

1. The login page for the WebGoat Coins Customer Portal allows for multiple logins without a timeout.
2. Implement a simple fix for this ONLY by adding a timeout for a user that tries to login more than three (3) times. This can be done in a variety of ways. However, the following is one simple approach (which can still be abused, but is better than nothing):
    i. Create a login log table that logs the email id used for login, and the timestamp for the last unsuccessful login.
    ii. In the Login page, modify the code to first add an entry into this log table whenever a password does not verify.
    iii. Next modify the Login page to first retrieve the three latest records by email id and to check whether three consecutive attempts have been made within a one-hour period using the TimeSpan class. Fail silently if the condition is true, otherwise proceed with the normal login operation.
3. Answer the given questions at the top of this handout and remember to commit and push your code under the Lab6_Phase1 branch.

Phase 2: Setting up TLS to prevent eavesdropping

In the attack lab, you saw that the credentials were being sent in the clear and were able to be easily intercepted.

1. The login page for the WebGoat Coins Customer Portal and all its pages do not use SSL.
2. Implement a fix for this ONLY adding TLS/SSL to the portal and configure it correctly. For a guide on how to do this, please see https://dotnetcodr.com/2015/09/18/how-to-enable-ssl-for-a-net-project-in-visual-studio/.
3. Answer the given questions at the top of this handout and remember to commit and push your code under the Lab6_Phase2 branch.

Phase 3: Secure Password Reset

In the attack lab, the forgot password page was insecurely implemented.

1. The forgot password page in Content folder needs to be fixed in order to implement that functionality securely. The actual and full process for doing that in ASP.NET is outlined here:

https://docs.microsoft.com/en-us/aspnet/identity/overview/features-api/account-confirmation-and-password-recovery-with-aspnet-identity. This typically involves:

i. Confirming the email-based identity of a user's account by sending a confirmation email using MailMessage class during user account creation.
ii. When the forgot password functionality is invoked by the user, it checks a field in the database on whether the user has confirmed their email-based identity. If not, it silently fails.
iii. The forgot password page also does not display the information to the user/attacker on whether the email-based identity even exists within the software system.
iv. It also makes use of a time-bound token (which expires for example in 3 hours) to ensure that the password can only be reset within a specific timeframe from when the user requests the password reset. The token prevents DoS attempts which use multiple password reset requests for the same account. A crafted email using the token in a URL is then sent to the email address for the one specified on the Forgot Password page. The account is not locked out at this point.
v. The account lockout feature is only recommended if it is implemented in conjunction with Two-Factor authentication, which is also implemented in the user's account. This will prevent someone malicious from locking out a normal user.

However, the above solution uses ASP .NET Identity framework along with Entity framework (optionally) and can be cumbersome to retrofit an old application like WebGoat.NET.

2. For this phase, implement a secure Forgot Password functionality using the concepts from the earlier example of using the Identity framework, but without using the Identity framework itself.
    i. We are going to assume that all the emails in the WebGoat .NET application are already verified emails.
    ii. Change the two-step password recovery into a single step process that just takes the email and the answer to the security question.
    iii. Fail silently if the email does not exist in the database. Otherwise, if the answer to the security question matches, do the following steps 2.iv – 2.viii.
    iv. Modify ChangePassword.aspx and its associated code file to accept a URL parameter called 'token'. To learn how to access a URL parameter, see https://docs.microsoft.com/en-us/previous-versions/aspnet/6c3yckfw%28v%3dvs.100%29.
    v. On the Forgot Password page:
        a) Use the https://github.com/kspearrin/Otp.NET library to generate a Timed One-Time Password code with the current UTC time and.
        b) Send an email with a crafted URL to an email address (use your own email for now instead of the user's email).
        c) To send email see:
            1. https://www.c-sharpcorner.com/UploadFile/2a6dc5/how-to-send-a-email-using-Asp-Net-C-Sharp/
        d) To store your email id and password, do not store it in the code. Instead use the Secret Manager in ASP.NET Core (and **do not check in** the secrets or the secrets file in your submission):

1. https://docs.microsoft.com/en-us/aspnet/core/security/app-secrets?view=aspnetcore-5.0&tabs=windows

e)The crafted URL will point to: http://localhost:52251/Content/ChangePassword.aspx?token=, where the token you generated is appended to the end.

f) Make sure to choose a cryptographically strong secret when generating the TOTP. You may choose to implement the TOTP related operations for this lab as part of its own class.

vi. Copy the ChangePassword.aspx file from *WebGoatCoins* folder to the *Content* folder.

vii. Modify the ChangePassword.aspx file to take the accepted 'token' parameter and use the TOTP verification function to check whether the token is valid or not. You can use the VerificationWindow function to specify the number of "time steps" that are acceptable from when the token was generated up to when the token is being verified. And only when the token is valid, you allow the user to change the password for the given email.

viii. When you get the email, you can copy and paste the link into WebGoat .NET and check whether you are able to reset the password for the given email account.

3. Answer the given questions at the top of this handout and remember to commit and push your code under the Lab6_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.