Attack Lab 5: Poor Authentication and Authorization

Date Due: Tuesday, October 19, 2021 by 11:59 PM

Introduction

This lab will demonstrate some of the common authentication and authorization weaknesses in web applications. This assumes you have completed Lab 0 which involves performing the Lab environment setup. For this lab, you will need to run the WebGoat.NET application which was performed as part of Lab 0: Phase 2.

WARNING: Do not attempt to use these techniques elsewhere. Only run them on the WebGoat.NET application either inside the virtual machine or the application running on your own machine. There are legal consequences to you if you use these elsewhere!

For each of the following on the WebGoat.NET application, please submit a single writeup containing all the phases and the answers to each of the questions provided in each phase. For each question, please look at the Lesson Instructions button on the given page unless specified otherwise. The answers should be provided in the following comma separated format:

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) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters, please provide the bytes provided to the input in Hexadecimal format. For example: 9ABCD01234.
c) Describe the output result.
d) Provide a screenshot of the output.
e) Provide the CWE-ID, Filename, Line Number of the weakness that is at the heart of the vulnerability.
f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

Note: The instructor may ask you apply some diffs to your codebase to update the Lab code for this Lab.

Phase 1: Burp Suite to intercept requests (15 points)

In Attack Lab 3, you were able to determine the password storage scheme was poor along with a weak cryptographic hash scheme being used. In Fix Lab 4, you fixed the password storage issue along with using a stronger cryptographic hash scheme. In this lab we look at authentication weaknesses where the authentication mechanism itself maybe implemented poorly.

If you have noticed, there is nothing preventing a user from trying to login 100, maybe 1000 or maybe even a million times! For this phase we will first get Burpsuite to intercept the Login request and determine the format used by the WebGoat.NET application to send the passwords to the web server.

You can watch the video here: https://www.youtube.com/watch?v=LSfnGi6Ah7M to determine how to configure, turn on and use 'Intercept' in BurpSuite. You can watch until timestamp 4:07 to understand the 'Intercept' setup. Now use one of the WebGoat.NET Coins Portal customer email and password that you cracked from Attack Lab 3 to intercept the Login using BurpSuite.

How is the WebGoat.NET application sending the login credentials? Are there any attempts to limit the number of logins? Now answer these using the questions given at the top of this lab handout.

Phase 2: Brute-force (20 points)

For this phase, a renowned international coin thief (you) has been able to get a hold of some of the email addresses of the participants in an auction that use the exact same WebGoat.NET application. You may have astutely already noticed there is no real way to register for this auction (in WebGoat). So, you devise a plan to try and find the passwords of some particular bidders by using brute-force. However, the auctioneers have updated the WebGoat.NET application to include your fixes from Fix Lab 4. The passwords are not really easy to crack this time (sadly for you). You are going to have to try and brute force to figure out the passwords being used.

For this phase, please pull **AuthLab** branch from the instructor's WebGoat repo. Then run the application in Firefox with BurpSuite and run 'Rebuild Database' clicking the button again.

Once you have applied the changes to the code and database, ensure everything works by using the WebGoat Coins Portal Login and trying to login as 'jerry@goatgoldstore.net' with password 'password' (without quotes). Now you will need to provide a dictionary to try passwords from. Remember the dictionary you used in Attack Lab 3 of passwords from SecLists (https://github.com/danielmiessler/SecLists/tree/master/Passwords)? The password for the auction customers below has been chosen from one of the lists (since the bidders you are targeting reuse passwords a lot and really like using the ones that Twitter has banned). You will use BurpSuite and its 'Intruder' or 'Repeater' features to implement the brute-forcing. You can watch the rest of the video provided in the earlier phase to understand how that works. The dictionary to use will be the twitter-banned list (https://github.com/danielmiessler/SecLists/blob/master/Passwords/twitter-banned.txt).

Your job will be to figure out the password for the following auction customers:

juri@gold4allagescom.net

karttunen@coinsoffinland.net

mory@osakacoinageco.net

Provide the answers to this question using items b), c), d) and f). For item b), it can be screenshots of how you setup BurpSuite to substitute the password for the brute-force login and how you configured it to stop when the login worked. For items c) and d), it can be what the password is for the three auction customer emails given above and how it succeeded.

Phase 3: Forgot Password (15 points)

For this phase, continuing your stratagem as the international coin thief, you wish to change three auction goer's passwords such that they are unable to login at the last minute close to when the auction ends and therefore miss the bids for the special coin you are targeting, thereby favoring someone (mostly from the previous phase) who has a low security facility to store the winning bid, from which you can steal the coin from. For this phase, you will take advantage of a poorly designed forgot password page to reset the password for the following email logins:

julie@golddepotinc.com

calaghan@australiangoldnetwork.net

semenov@kremlincollectables.com

Please go to the Authentication & Authorization section of the WebGoat.NET application and access the Forgot Password exercise and perform this for the above given email ids. Based on what you performed, please answer the questions given at the top of this lab handout.

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 as specified in each of the phases.

Please only submit a **Word document** or a **PDF**. There is no code submission for this lab.