Attack Lab 9: Poor Error Handling and Logging

Date Due: Tuesday, November 16, 2021 by 11:59 PM

Introduction

This lab will demonstrate some of the common error handling and logging related 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 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.

Important Note: For this lab, you will be mostly performing exercises under 'Error Handling and Logging Issues' section of the WebGoat .NET application. Please pull 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: Unsafe Code

A lot of the managed or VM based languages (like Common Language Runtime based, Java Virtual Machine based, etc.) allow, for compatibility purposes, loosening restrictions on the compiler to allow writing code that is dangerous yet allows for maximum flexibility. In ASP.NET and .NET languages you are allowed to add 'unsafe' blocks of code by marking a function with the 'unsafe' keyword. This allows you to use pointers and more direct access to memory with memory-unsafe operations in .NET languages, which the language normally does not allow.

1. Do the 'Unsafe Blocks' exercise under the 'Error Handling and Logging Issues' section of the WebGoat .NET application and answer the questions given at the top of this lab handout. For this exercise, you are trying to come up with an input that will cause an exception or crash the ASP.NET application.

Phase 2: HTTP VERB Tampering

Many of the web servers or frameworks allow for authentication and authorization of HTTP verbs like GET, PUT, DELETE etc. for different pages. ASP .NET is no exception.

2. Now attempt the 'Verb Tampering' exercise under the 'Error Handling and Logging Issues' section of the WebGoat .NET application and answer the questions given at the top of this lab handout. For this exercise in the Attack Lab, **without logging in as any user,** the goal is to try to change the message on the page to say "Used Verb [HTTP VERB HERE] to tamper this message". Insert the **actual HTTP Verb** used to tamper the message instead of '[HTTP VERB HERE]'. Keep the following in mind:

- Use the Burp Suite Repeater feature to change/tamper the message below.
- Use the Repeater to make a request to **VerbTamperingAttack.aspx** (different from VerbTampering.aspx), which takes a query parameter called '**message**'.
- You will notice VerbTamperingAttack.aspx page is not in the WebGoat .NET menu and is "hidden". You will need to use Burp Suite to send the request to this page.
- The 'message' query parameter then changes what gets displayed in the message below.
- **Do not change** the Web.config file or any code file (You will see in this file that certain verbs for non-logged in users are not allowed).

Phase 3: Poor Error Handling

The .NET platform allows for good enough error or exception handling. The platform experts also backtracked on whether Exceptions should be derived from ApplicationException class or the base class Exception. The final word was to have it derive from Exception and have lots of shallow Exception classes instead of deep nested exception classes. However, despite these error handling capabilities, developers often don't handle them correctly or forget to clear up resources …

3. Now attempt the 'Erroring Page' exercise under the 'Error Handling and Logging Issues' section of the WebGoat .NET application and answer the questions given at the top of this lab handout with regards to what you just performed. This page implements a custom File Validation with error handling, however does it incorrectly leading to a resource leak.

ASP.NET allows for embedding C# code as part of ASPX pages by using the <script runat="server" /> tag. The problem with this is the code now embedded in the page is put into an autogenerated class by the Framework and run with some hints at error handling. For more information see https://docs.microsoft.com/en-us/troubleshoot/aspnet/inline-expressions and https://stackoverflow.com/a/28574590 (shows an example of the code generated by ASP.NET).

4. Now attempt the 'Embedded Error Page' exercise under the 'Error Handling and Logging Issues' section of the WebGoat .NET application. For this exercise in the Attack Lab, you will try to generate an exception by trying to add two numbers. Try to choose two numbers that will cause an exception. Assuming this calculator was a legitimate page/function of the WebGoat .NET application, now answer the questions specified at the top of this lab handout with regards to the handling of the error, not the Exception itself. For example, CWE-190 does not apply here since the number has not overflown and resulted in an integer that is then used by the application.

Phase 4: Poor Logging Practices

ASP.NET like many other languages and frameworks allows for logging. The WebGoat .NET does this by using the Log4Net library from Apache to do this.

5. For this exercise, identify all the places insecure logging is being performed in the WebGoat .NET application. Answer the questions at the top of this lab handout for each finding. You should a log file called log.txt in App_Data folder assuming you have installed log4net package version 2.0.5.

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.