ENPM809W - Introduction to Secure Coding

Lab - 9 – Attack Lab – Poor Error Handling Logging

Author: Syed Mohammad Ibrahim
UID: iamibi
Email: iamibi@umd.edu

Phase 1: Unsafe Code

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.

http://localhost:52251/Content/Unsafe.aspx

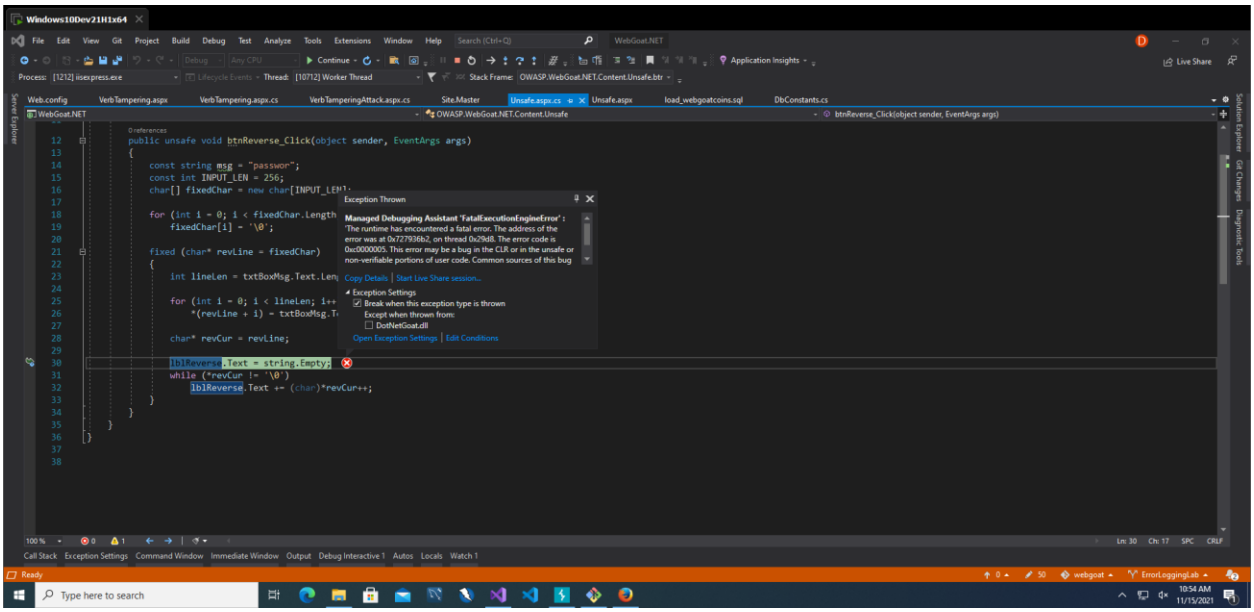
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters, please provide the bytes provided to the input in Hexadecimal format. For example: 9ABCD01234.

“A” X 5000 provided in the text box.

c) Describe the output result.

The IIS server crashed with an exception caused by buffer write overflow.

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.

CWE-ID(s):
- CWE-787: Out-of-bounds Write
- CWE-248: Uncaught Exception

Filename: Unsafe.aspx.cs
Line Number: 30

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

The given function is marked unsafe which in turn allows the usage of pointers. Because the pointers are manually managed by programmer, they can potentially read or write to memory locations that are out of bounds. They may end up allocating memory which is not freed for a longer time as Garbage Collector doesn't runs asynchronously and doesn't have the idea of allocations made by pointers. The function doesn't handle the exception which leads to crashing of application.

Phase 2: HTTP VERB Tampering

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.

http://localhost:52251/Content/VerbTampering.aspx

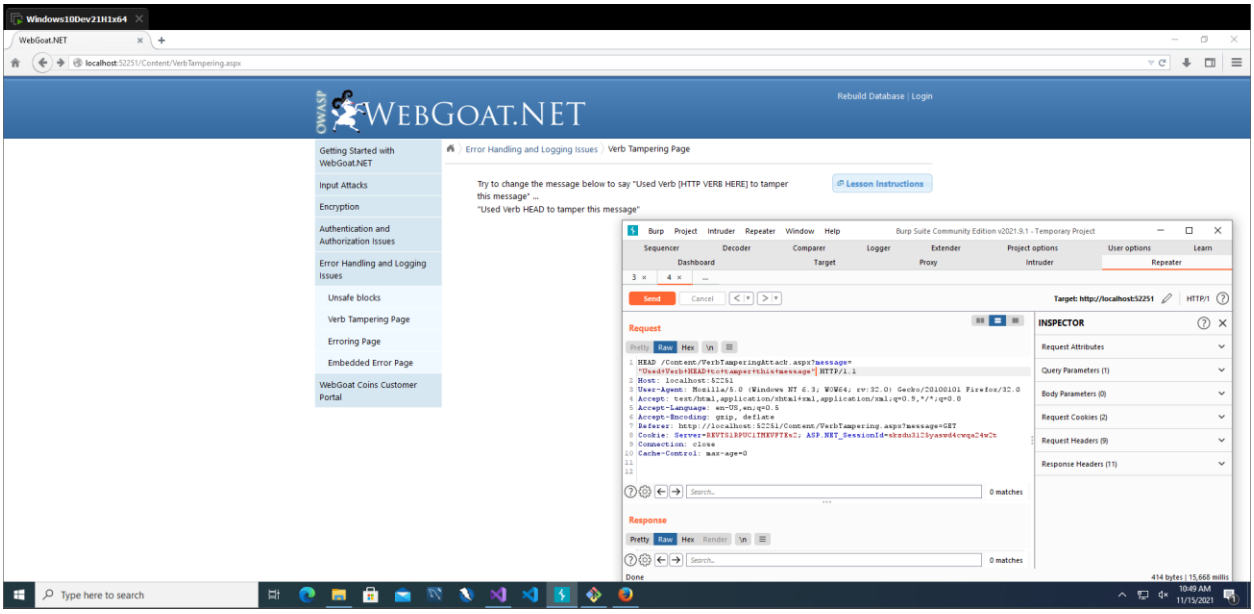
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters, please provide the bytes provided to the input in Hexadecimal format. For example: 9ABCD01234.

The following changes were made to the request captured by the Burp Suite
HEAD
/Content/VerbTamperingAttack.aspx?message="Used+Verb+HEAD+to+tamper+this+message"

c) Describe the output result.

The output was that the VerbTampering.aspx page "message" changed to "Used Verb HEAD to tamper this message" after refreshing.

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.

CWE-ID(s):
- CWE-650: Trusting HTTP Permission Methods on the Server Side

Filename: VerbTamperingAttack.aspx.cs
Line Number: 14

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

The server doesn't perform proper verb and request validation which cause the message to be changed by any other valid verb passed.

Phase 3: Poor Error Handling

Part 1

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.

http://localhost:52251/Content/Error.aspx

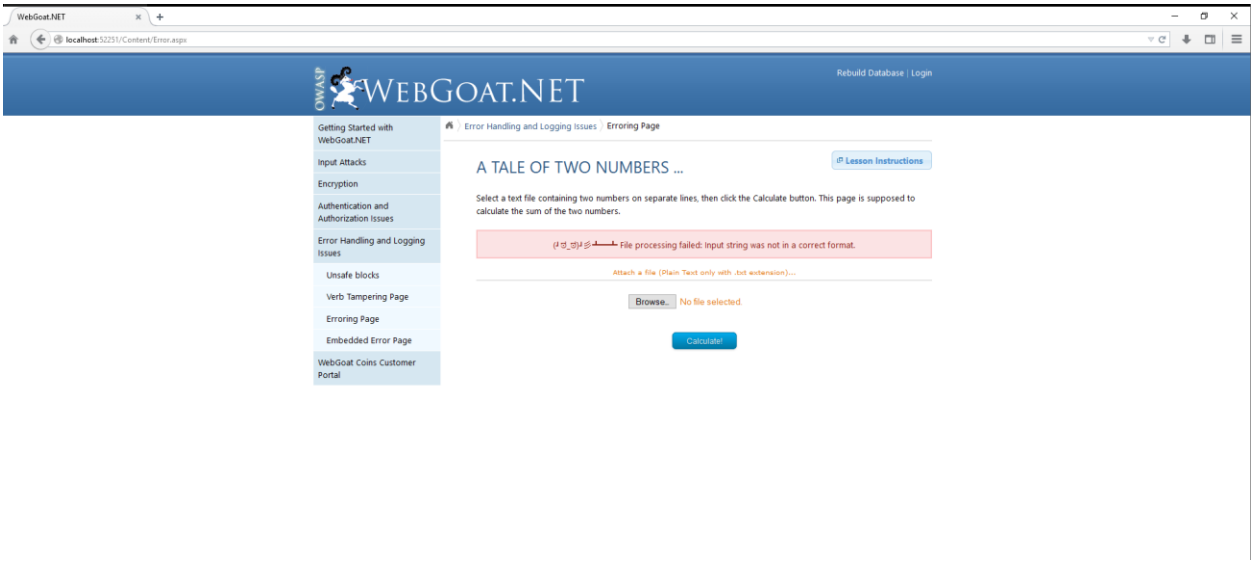
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters, please provide the bytes provided to the input in Hexadecimal format. For example: 9ABCD01234.

A text file containing the following lines:
abcd
efgh

c) Describe the output result.

The output was an error message displayed on the web browser. However, the file resource was not cleaned up as the exception occurred in the code.

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.

CWE-ID(s):
- CWE-460: Improper Cleanup on Thrown Exception

Filename: Error.aspx.cs
Line Number: 41

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

After the exception condition is rescued, the File is never deleted. This in turn can lead an attacker to perform a DOS attack on the server.

Part 2

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.

http://localhost:52251/Content/EmbeddedError.aspx

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.

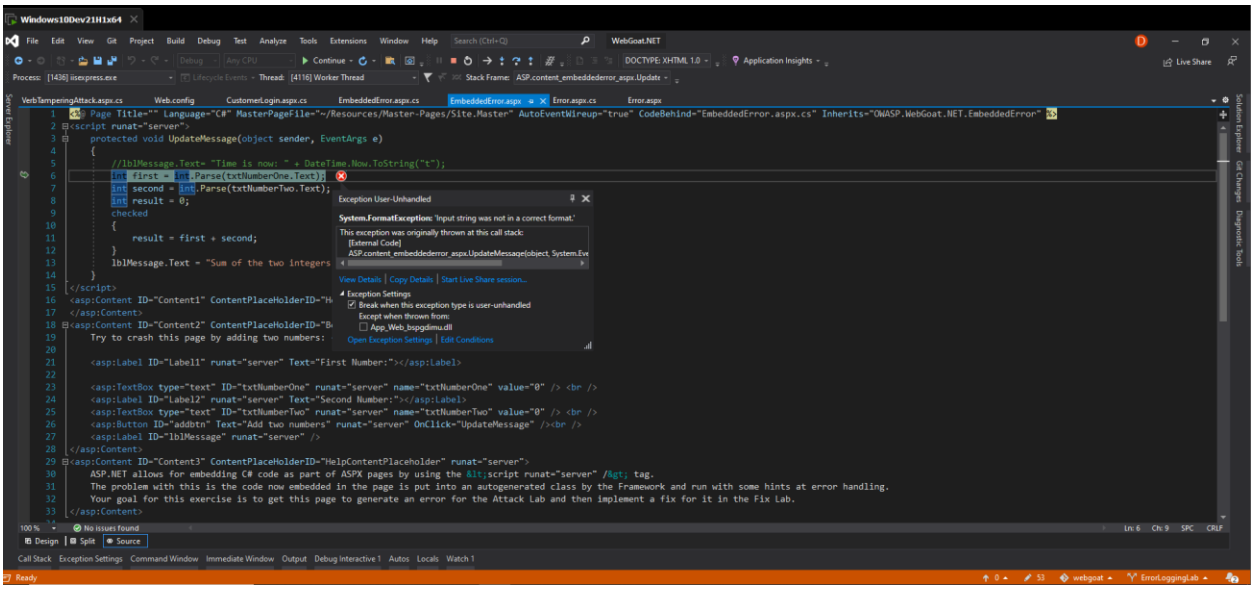
Textfield 1: 10.5

Textfield 2: 20.1

c) Describe the output result.

An exception was thrown causing the server page to crash.

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.

CWE-ID(s):
- CWE-248: Uncaught Exception

Filename: EmbeddedError.aspx
Line Number: 6

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

The code doesn't handle any value that is not an integer. That is characters or floating point numbers when passed makes the application crash.

Phase 4: Poor Logging Practices

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.

N/A

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.

Going through webgoat coins portal and logging in and logging out.

c) Describe the output result.
The logs were written for login with email id and password with no string sanitization being done.

d) Provide a screenshot of the output.

```
INFO  2021-11-15 11:56:25,930 27705ms NOTIFY               ButtonLogOn_Click  - User jerry@goatgoldstore.net attempted to log in with password password
INFO  2021-11-15 11:56:36,649 38424ms NOTIFY               btnLogout_Click    - User jerry@goatgoldstore.net performed a logout.
```

e) Provide the CWE-ID, Filename, Line Number of the weakness that is at the heart of the vulnerability.

CWE-ID(s):
- CWE-532: Insertion of Sensitive Information into Log File
- CWE-117: Improper Output Neutralization for Logs

Filename: CustomerLogin.aspx.cs
Line Number: 37

Filename: Logout.aspx.cs
Line Number: 26

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

The log file writes sensitive information in the logs like passwords and doesn't Neutralize the information that is being written in it. This can potentially lead an attacker to access log files and read/modify the contents present in it.