ENPM809W - Introduction to Secure Coding

Lab - 1 - Attack Lab

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

Phase - 1, Part - 1: SQL Error Messages

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/SQLInjectionDiscovery.aspx

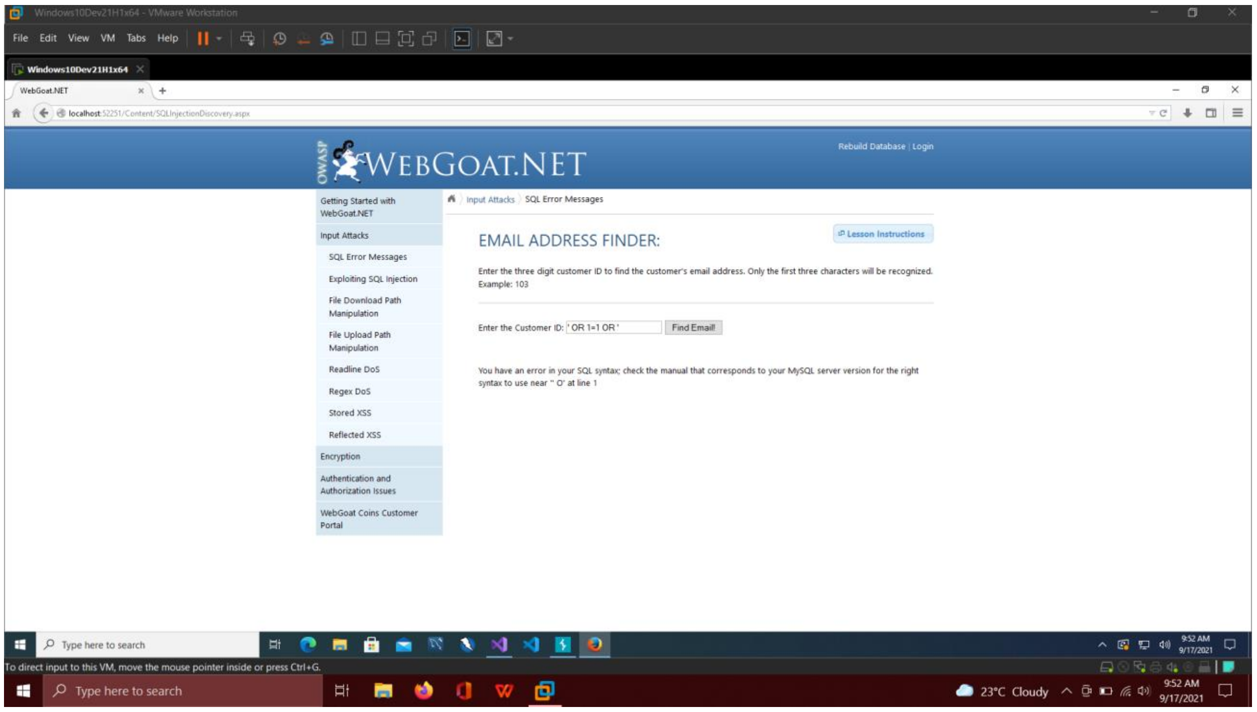
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each.If the input is non-printable characters,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

   The provided input was ' OR 1=1 OR ' which is a SQL query to always return true when passed in the **where** clause of SQL.

c) Describe the output result.

   The output result was "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' O' at line 1" which is an SQL syntax error message which demonstrates that the application is not sanitizing the inputs from the text field and is directly substituting them to the SQL query to get the result.

d) Provide a screenshot of the output.


Windows10Dev21H1x64 - VMware Workstation
File Edit View VM Tabs Help
WebGoat.NET
localhost:52251/Content/SQLInjectionDiscovery.aspx
WEBGOAT.NET
Rebuild Database | Login
Getting Started with WebGoat.NET
Input Attacks
SQL Error Messages
Exploiting SQL Injection
File Download Path Manipulation
File Upload Path Manipulation
Readline DoS
Regex DoS
Stored XSS
Reflected XSS
Encryption
Authentication and Authorization Issues
WebGoat Coins Customer Portal
Input Attacks › SQL Error Messages
Lesson Instructions
Find Email!
Type here to search
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
23°C Cloudy
9:52 AM 9/17/2021

EMAIL ADDRESS FINDER:

Enter the three digit customer ID to find the customer's email address. Only the first three characters will be recognized. Example: 103

Enter the Customer ID: ' OR 1=1 OR '

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' O' at line 1

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

CWE-89: Improper Neutralization of Special Elements used in SQL Command (‘SQL Injection’)
Filename: MySqlDbProvider.cs
Line number: 539

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

The vulnerability is that the input text is not being tested and sanitized before passing the value to the SQL constructed query. This causes the SQL to substitute the value as is, and the attacker can leverage this vulnerability to gain access to all the credentials stored in the database by crafting a query that will return other user’s information.

Phase - 1, Part - 2: SQL Injection

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/SQLInjection.aspx

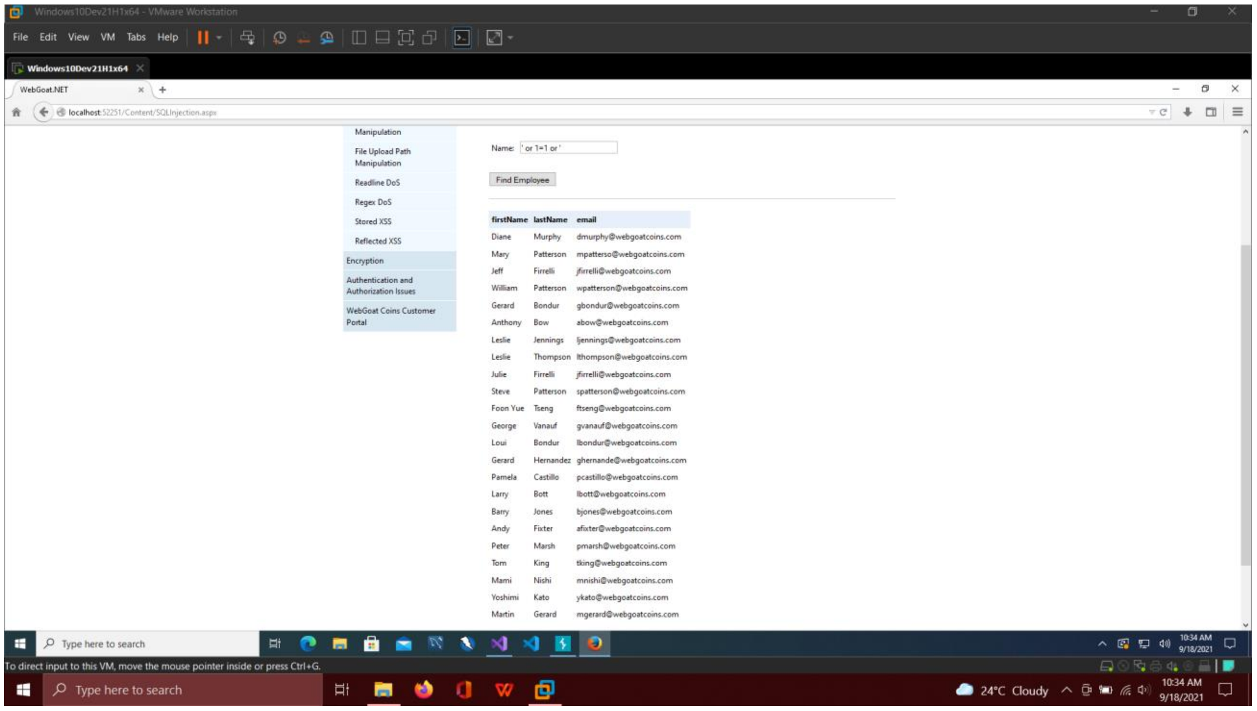
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters,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

The provided input was ‘ or 1=1 or ’ which makes the regular SQL query to always return true.

c) Describe the output result.

The output result was the list of all the users present in the database with three columns, that is firstName, lastName and email.

d) Provide a screenshot of the output.


Windows10Dev21H1x64 - VMware Workstation
File Edit View VM Tabs Help
Windows10Dev21H1x64
WebGoat.NET
localhost:52251/Content/SQLInjection.aspx
Manipulation
File Upload Path Manipulation
Readline DoS
Regex DoS
Stored XSS
Reflected XSS
Encryption
Authentication and Authorization Issues
WebGoat Coins Customer Portal
Find Employee
Type here to search
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
24°C Cloudy
10:34 AM
9/18/2021

Name: ' or 1=1 or '

firstName	lastName	email
Diane	Murphy	dmurphy@webgoatcoins.com
Mary	Patterson	mpatterso@webgoatcoins.com
Jeff	Firrelli	jfirrelli@webgoatcoins.com
William	Patterson	wpatterson@webgoatcoins.com
Gerard	Bondur	gbondur@webgoatcoins.com
Anthony	Bow	abow@webgoatcoins.com
Leslie	Jennings	ljennings@webgoatcoins.com
Leslie	Thompson	lthompson@webgoatcoins.com
Julie	Firrelli	jfirrelli@webgoatcoins.com
Steve	Patterson	spatterson@webgoatcoins.com
Foon Yue	Tseng	ftseng@webgoatcoins.com
George	Vanauf	gvanauf@webgoatcoins.com
Loui	Bondur	lbondur@webgoatcoins.com
Gerard	Hernandez	ghernande@webgoatcoins.com
Pamela	Castillo	pcastillo@webgoatcoins.com
Larry	Bott	lbott@webgoatcoins.com
Barry	Jones	bjones@webgoatcoins.com
Andy	Fixter	afixter@webgoatcoins.com
Peter	Marsh	pmarsh@webgoatcoins.com
Tom	King	tking@webgoatcoins.com
Mami	Nishi	mnishi@webgoatcoins.com
Yoshimi	Kato	ykato@webgoatcoins.com
Martin	Gerard	mgerard@webgoatcoins.com

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

CWE-89: Improper Neutralization of Special Elements used in SQL Command (‘SQL Injection’)
Filename: MySqlDbProvider.cs
Line number: 517

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

The vulnerability was that the input was not being sanitized while it was being substituted to the SQL query. This led to a random SQL query that can be passed as an input, causing the query to return the information of all the users from the database.

Phase - 2, Part - 1: File Download Path Manipulation

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/PathManipulation.aspx

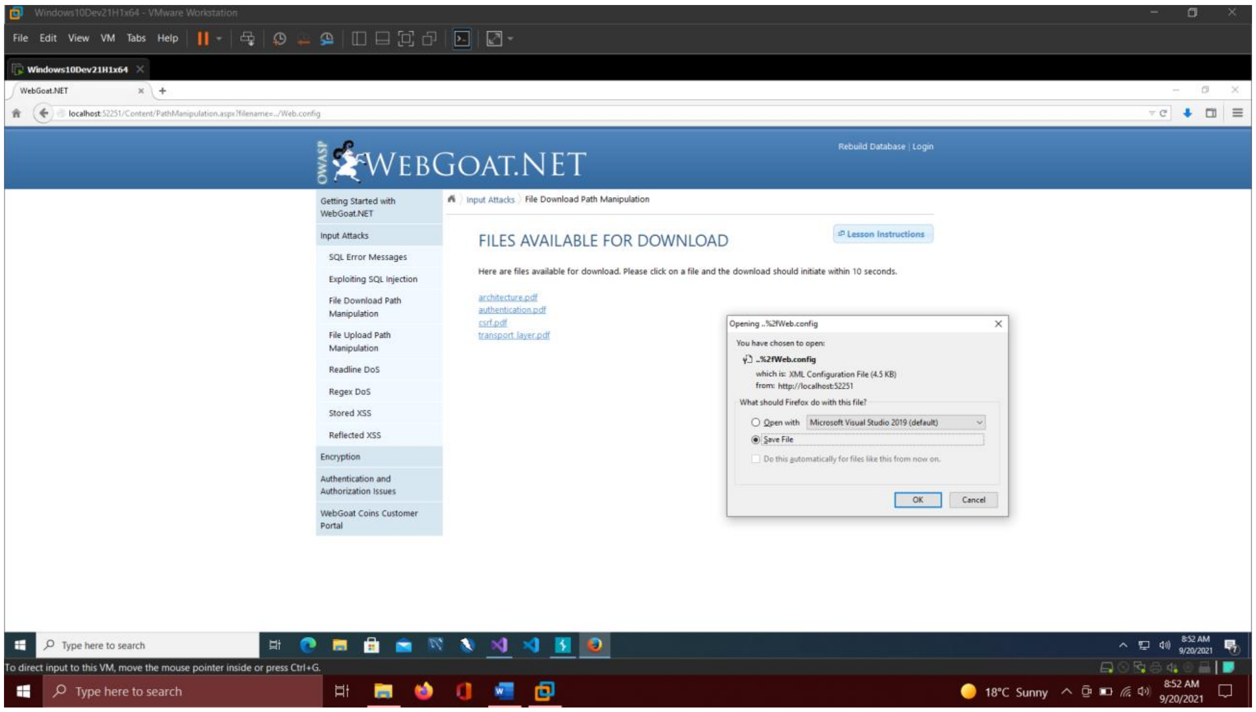
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters,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

The provided input was
http://localhost:52251/Content/PathManipulation.aspx?filename=../Web.config
Where the parameter 'filename' was passed in as a path to the web.config file.

c) Describe the output result.

The output was the web.config file which was now accessible and can be downloaded locally.

d) Provide a screenshot of the output.



localhost:52251/Content/PathManipulation.aspx?filename=../Web.config

FILES AVAILABLE FOR DOWNLOAD

Here are files available for download. Please click on a file and the download should initiate within 10 seconds.

architecture.pdf
authentication.pdf
csrf.pdf
transport layer.pdf

Opening ..%2fWeb.config

You have chosen to open:

..%2fWeb.config

which is: XML Configuration File (4.5 KB)
from: http://localhost:52251

What should Firefox do with this file?

Open with Microsoft Visual Studio 2019 (default) [✓ Save File]

Do this automatically for files like this from now on.

Windows10Dev21H1x64 - VMware Workstation
File Edit View VM Tabs Help
WebGoat.NET
WEBGOAT.NET
Rebuild Database | Login
Input Attacks > File Download Path Manipulation
Lesson Instructions
Getting Started with WebGoat.NET
Input Attacks
SQL Error Messages
Exploiting SQL Injection
File Download Path Manipulation
File Upload Path Manipulation
Readline DoS
Regex DoS
Stored XSS
Reflected XSS
Encryption
Authentication and Authorization Issues
WebGoat Coins Customer Portal
OK
Cancel
Type here to search
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
18°C Sunny
8:52 AM
9/20/2021

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

CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
Filename: PathManipulation.aspx.cs
Line number: 25

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

Since the input from the URL is being substituted to the query parameter "?filename=" directly without any checks being made, an attacker can input a relative path which can get them access to confidential or critical files.

Phase - 2, Part - 2: File Upload Path Manipulation

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/UploadPathManipulation.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.

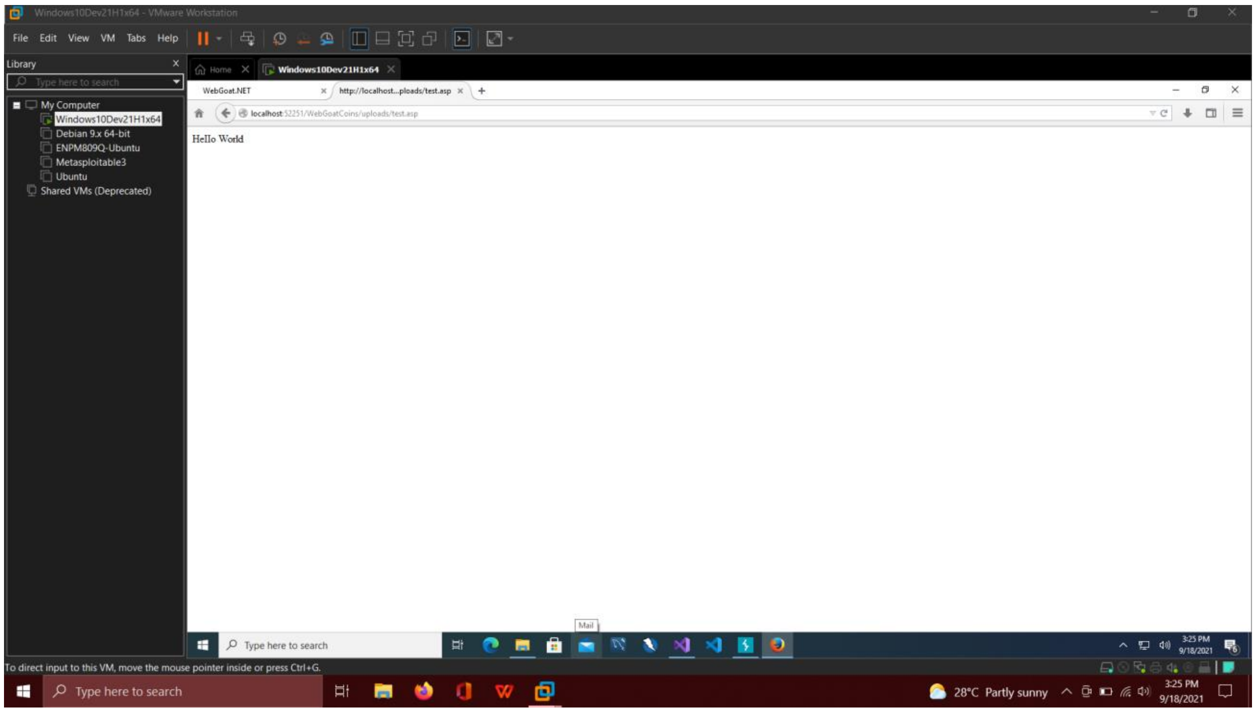
   The input was a **test.asp** file which was uploaded to the webgoat server with the following code:

```
<html xmlns="www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    <%Response. Write( "Hello World") %>

    </div>
    </form>
</body>
</html>
```

c) Describe the output result.

   The result of uploading the above content was that when we tried accessing the file with the path **localhost:52251/WebGoatCoins/uploads/test.asp**, it would execute successfully and print the “Hello World” text on the browser.

d) Provide a screenshot of the output.



localhost:52251/WebGoatCoins/uploads/test.asp

```
Hello World
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

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

CWE-434: Unrestricted Upload of a File with Dangerous Type
Filename: UploadPathManipulation.aspx.cs
Line number: 25

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

The vulnerability resides in the way the files are being uploaded without any content integrity checks being done either in the front-end or backend systems. The files should always be verified whether or not they are of dangerous types or not.

Phase - 2, Part - 3: Readline DOS

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/ReadlineDoS.aspx

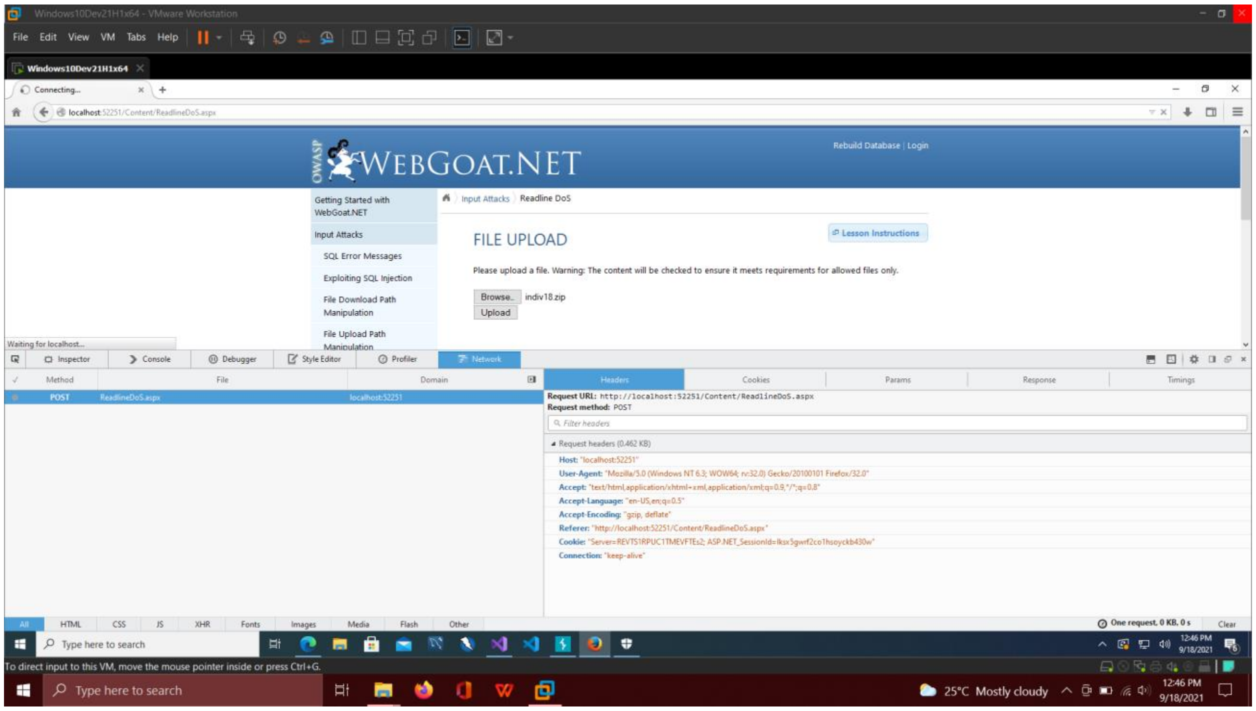
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters,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

Input was a large text file with the first 10 bytes being C00401224.

c) Describe the output result.

The result was that the webpage crashed while trying to read the large file.

d) Provide a screenshot of the output.



FILE UPLOAD

Please upload a file. Warning: The content will be checked to ensure it meets requirements for allowed files only.

indiv18.zip

```
POST   ReadlineDoS.aspx   localhost:52251

Request URL: http://localhost:52251/Content/ReadlineDoS.aspx
Request method: POST
Request headers (0.462 KB)
Host: "localhost:52251"
User-Agent: "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0"
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
Accept-Language: "en-US,en;q=0.5"
Accept-Encoding: "gzip, deflate"
Referer: "http://localhost:52251/Content/ReadlineDoS.aspx"
Cookie: "Server=REVTS1RPUC1TMEVFTEs2; ASP.NET_SessionId=lksx5gwrf2co1hsoyckb430w"
Connection: "keep-alive"
```

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

CWE-434: Unrestricted Upload of File with Dangerous Type
Filename: ReadlineDoS.aspx.cs
Line number: 23

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

The vulnerability was that the file reading was being done using “Readline()” which tries to halt the program until it has completed reading the line from the buffer memory and reached a return carrier (\r) or new line (\n). Because of this, an attacker can push a file with a very big single line causing the full memory usage.

Phase - 3: Regex DOS

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/RegexDoS.aspx

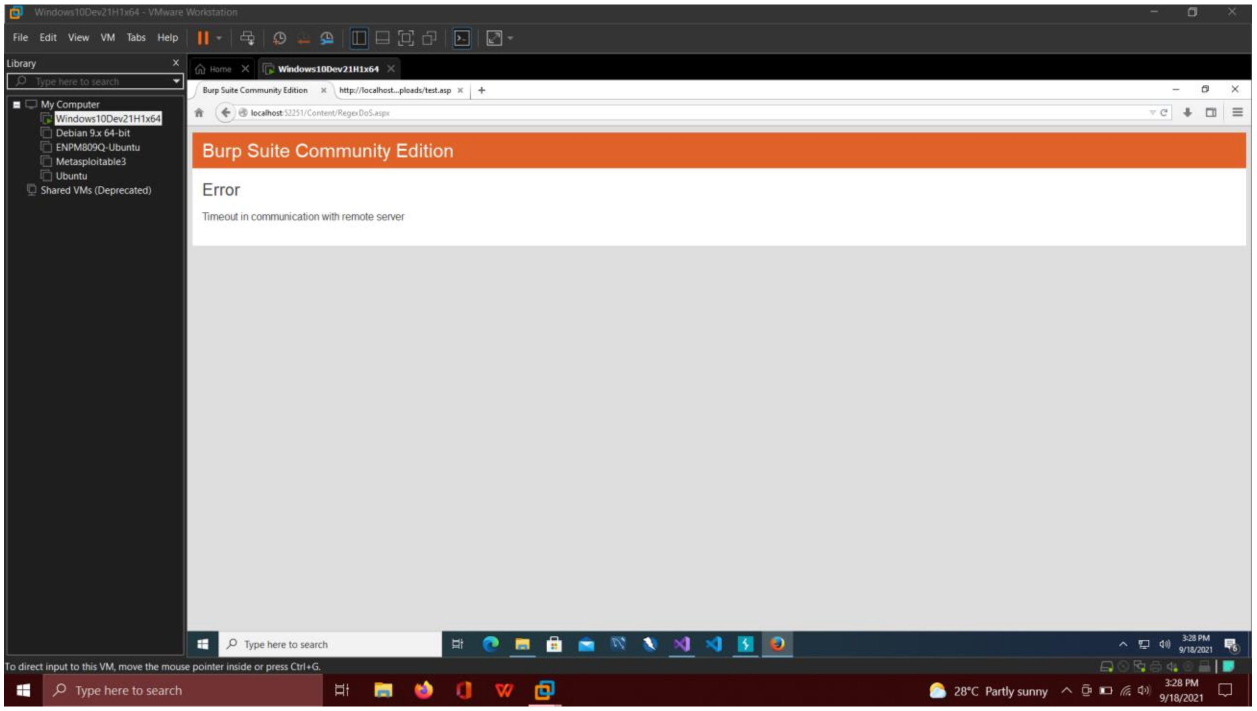
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters,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

The input was
“aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaa!” without the quotes.

c) Describe the output result.

The output was the page timed out because of trying to parse a large input regex.

d) Provide a screenshot of the output.



Burp Suite Community Edition

Error

Timeout in communication with remote server

Windows10Dev21H1x64 - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
Windows10Dev21H1x64
Debian 9.x 64-bit
ENPM809Q-Ubuntu
Metasploitable3
Ubuntu
Shared VMs (Deprecated)
localhost:52251/Content/RegexDoS.aspx
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
28°C Partly sunny
3:28 PM
9/18/2021

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

CWE-1333: Inefficient Regular Expression Complexity
Filename: RegexDoS.aspx.cs
Line number: 25

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

The vulnerability is that the input is not being sanitized to remove any kind of regular expression which in turn can assist the attacker to input a complex regex that can take a lot of time to compute. The input is directly substituted to the regex matcher.

Phase - 4, Part - 1: Stored XSS

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/StoredXSS.aspx

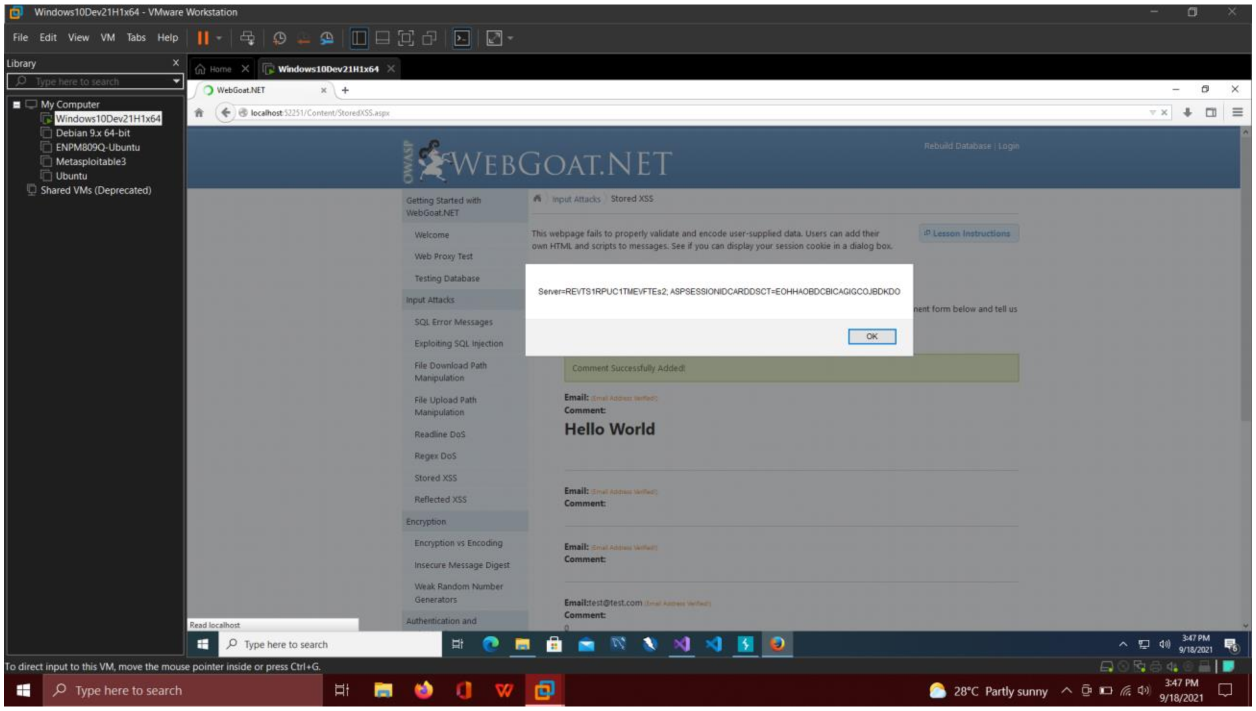
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters,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

The input was the following
Email: test@test.com
Comment: <script>alert(document.cookie);</script>

c) Describe the output result.

An alert was displayed with the session cookie values printed in it.

d) Provide a screenshot of the output.



Windows10Dev21H1x64 - VMware Workstation

localhost:52251/Content/StoredXSS.aspx

Input Attacks › Stored XSS

This webpage fails to properly validate and encode user-supplied data. Users can add their own HTML and scripts to messages. See if you can display your session cookie in a dialog box.

```
Server=REVTS1RPUC1TMEVFTEs2; ASPSESSIONIDCARDDSCT=EOHHAOBDCBICAGIGCOJBDKDO
```

Comment Successfully Added!

Email:
Comment:
Hello World

Email:
Comment:

Email:
Comment:

Email:test@test.com
Comment:

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

CWE-79: Improper Neutralization of Input During Web Page Generation (‘Cross-Site Scripting’)
Filename: StoredXSS.aspx.cs
Line number: 61

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

The vulnerability resides in the way the input is being treated. There is no sanitization happening on the comments text box which leads to an attacker passing in a javascript code script and getting the session information. This happens because the input is first stored and then loaded by the application as part of the HTML code to render the text.

Phase - 4, Part - 2: Reflected XSS

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/ReflectedXSS.aspx

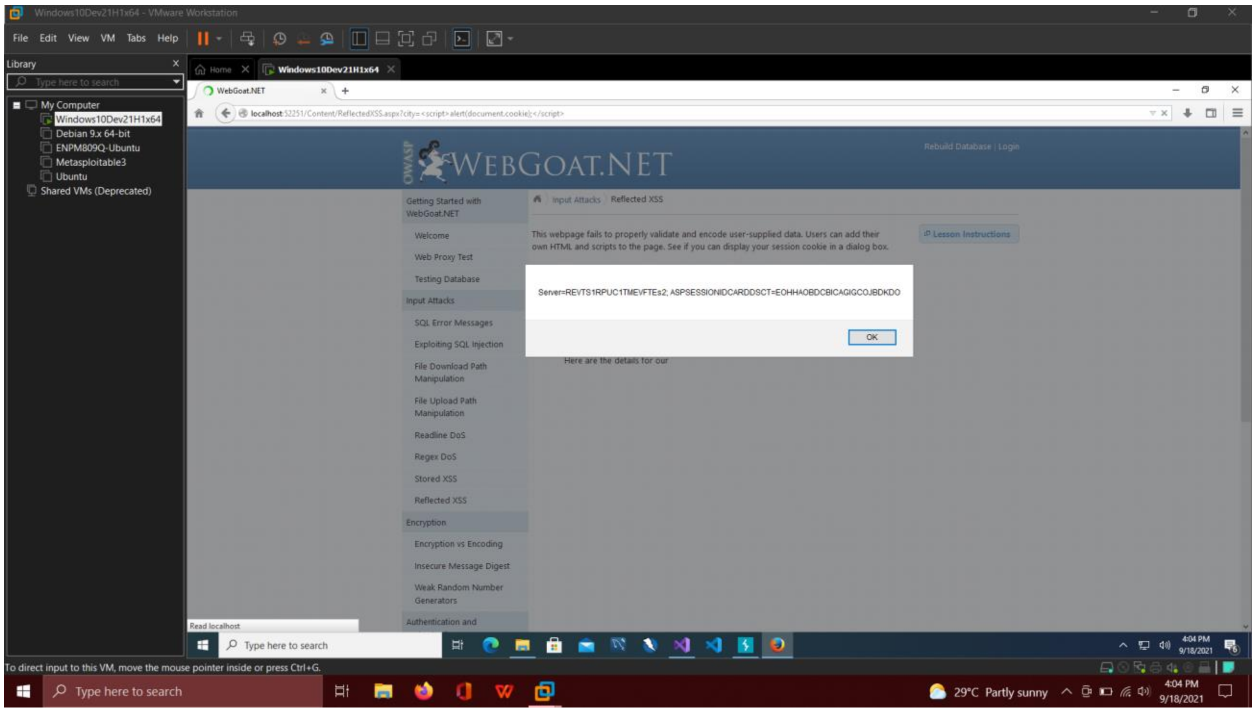
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters,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

   The input was a modified query parameter to the URL where we pass a javascript code snippet instead of an expected city value.
   http://localhost:52251/Content/ReflectedXSS.aspx?city=<script>alert(document.cookie);</script>

c) Describe the output result.

   The result is an alert pop-up which displays the session cookie values.

d) Provide a screenshot of the output.



```
localhost:52251/Content/ReflectedXSS.aspx?city=<script>alert(document.cookie);</script>
```

This webpage fails to properly validate and encode user-supplied data. Users can add their own HTML and scripts to the page. See if you can display your session cookie in a dialog box.

```
Server=REVTS1RPUC1TMEVFTEs2; ASPSESSIONIDCARDDSCT=EOHHAOBDCBICAGIGCOJBDKDO
```

Here are the details for our

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

CWE-79: Improper Neutralization of Input During Web Page Generation (‘Cross-Site Scripting’)
Filename: ReflectedXSS.aspx.cs
Line number: 34

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

The vulnerability is that the input is being loaded in the context that it is a plain text but it is actually never verified and no kind of sanity checks are done. This leads an attacker to load a custom javascript as part of the URL query parameter which gets executed successfully on the browser window.

Phase - 5: WinSharpFuzz Fuzzing

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.

NA

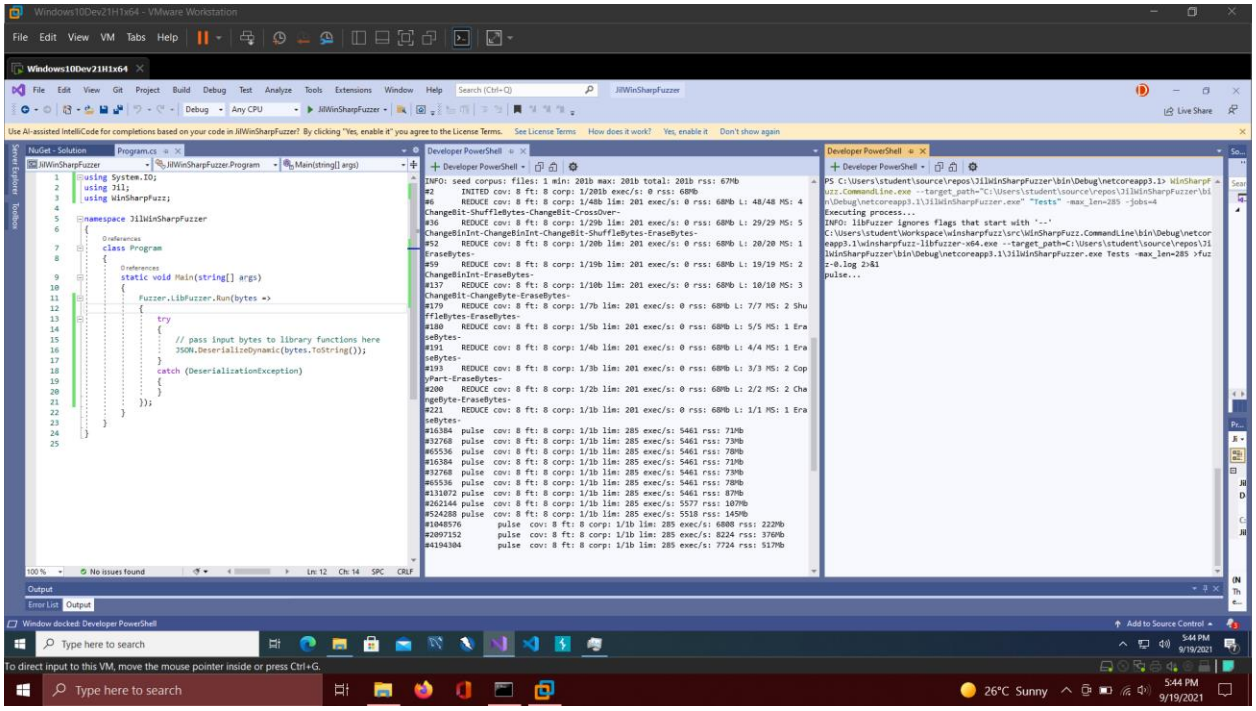
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters,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

```
{
 "menu":{
 "menu": {
   "id": 1,
 {
```

c) Describe the output result.

The fuzzer was trying to find weakness in the JSON library and printed the options it has already tried, in a log file.

d) Provide a screenshot of the output.



```
using System.IO;
using Jil;
using WinSharpFuzz;

namespace JilWinSharpFuzzer
{
    class Program
    {
        static void Main(string[] args)
        {
            Fuzzer.LibFuzzer.Run(bytes =>
            {
                try
                {
                    // pass input bytes to library functions here
                    JSON.DeserializeDynamic(bytes.ToString());
                }
                catch (DeserializationException)
                {
                }
            });
        }
    }
}
```

```
INFO: seed corpus: files: 1 min: 201b max: 201b total: 201b rss: 67Mb
#2      INITED cov: 8 ft: 8 corp: 1/201b exec/s: 0 rss: 68Mb
#6      REDUCE cov: 8 ft: 8 corp: 1/48b lim: 201 exec/s: 0 rss: 68Mb L: 48/48 MS: 4 ChangeBit-ShuffleBytes-ChangeBit-CrossOver-
#36     REDUCE cov: 8 ft: 8 corp: 1/29b lim: 201 exec/s: 0 rss: 68Mb L: 29/29 MS: 5 ChangeBinInt-ChangeBinInt-ChangeBit-ShuffleBytes-EraseBytes-
#52     REDUCE cov: 8 ft: 8 corp: 1/20b lim: 201 exec/s: 0 rss: 68Mb L: 20/20 MS: 1 EraseBytes-
#59     REDUCE cov: 8 ft: 8 corp: 1/19b lim: 201 exec/s: 0 rss: 68Mb L: 19/19 MS: 2 ChangeBinInt-EraseBytes-
#137    REDUCE cov: 8 ft: 8 corp: 1/10b lim: 201 exec/s: 0 rss: 68Mb L: 10/10 MS: 3 ChangeBit-ChangeByte-EraseBytes-
#179    REDUCE cov: 8 ft: 8 corp: 1/7b lim: 201 exec/s: 0 rss: 68Mb L: 7/7 MS: 2 ShuffleBytes-EraseBytes-
#180    REDUCE cov: 8 ft: 8 corp: 1/5b lim: 201 exec/s: 0 rss: 68Mb L: 5/5 MS: 1 EraseBytes-
#191    REDUCE cov: 8 ft: 8 corp: 1/4b lim: 201 exec/s: 0 rss: 68Mb L: 4/4 MS: 1 EraseBytes-
#193    REDUCE cov: 8 ft: 8 corp: 1/3b lim: 201 exec/s: 0 rss: 68Mb L: 3/3 MS: 2 CopyPart-EraseBytes-
#200    REDUCE cov: 8 ft: 8 corp: 1/2b lim: 201 exec/s: 0 rss: 68Mb L: 2/2 MS: 2 ChangeByte-EraseBytes-
#221    REDUCE cov: 8 ft: 8 corp: 1/1b lim: 201 exec/s: 0 rss: 68Mb L: 1/1 MS: 1 EraseBytes-
#16384  pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5461 rss: 71Mb
#32768  pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5461 rss: 73Mb
#65536  pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5461 rss: 78Mb
#16384  pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5461 rss: 71Mb
#32768  pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5461 rss: 73Mb
#65536  pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5461 rss: 78Mb
#131072 pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5461 rss: 87Mb
#262144 pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5577 rss: 107Mb
#524288 pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 5518 rss: 145Mb
#1048576        pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 6808 rss: 222Mb
#2097152        pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 8224 rss: 376Mb
#4194304        pulse  cov: 8 ft: 8 corp: 1/1b lim: 285 exec/s: 7724 rss: 517Mb
```

```
PS C:\Users\student\source\repos\JilWinSharpFuzzer\bin\Debug\netcoreapp3.1> WinSharpFuzz.CommandLine.exe --target_path="C:\Users\student\source\repos\JilWinSharpFuzzer\bin\Debug\netcoreapp3.1\JilWinSharpFuzzer.exe" "Tests" -max_len=285 -jobs=4
Executing process...
INFO: libFuzzer ignores flags that start with '--'
C:\Users\student\Workspace\winsharpfuzz\src\WinSharpFuzz.CommandLine\bin\Debug\netcoreapp3.1\winsharpfuzz-libfuzzer-x64.exe --target_path=C:\Users\student\source\repos\JilWinSharpFuzzer\bin\Debug\netcoreapp3.1\JilWinSharpFuzzer.exe Tests -max_len=285 >fuzz-0.log 2>&1
pulse...
```

From left to right: Code, Log file, Powershell running the fuzzer

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

NA

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

NA