

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: CYSE 230, 2021 Spring

Laboratory Exercise 2 – Stored Cross Site Scripting

1. Overview

In this lab, students will use the Cyber Range: Cyber Basics (2018) environment to learn how to execute and exploit stored cross-site scripting on an intentionally vulnerable web application.

2. Resources required

This exercise requires a Kali Linux VM running in the Cyber Range.

3. Initial Setup

For this exercise, you will log in to your Cyber Range account and select the Cyber Basics (2018) environment, then click “start” to start your environment and “join” to get to your Linux desktop login. Log in using these credentials:

Username: **student**
Password: **student**

4. Tasks

[Note to instructors: Some previously learned techniques are not explained again in this lesson, nor are there screenshots of the results. This is an attempt to scaffold the learning experience.]

Task 1: DVWA Stored XSS with Low Security

- Open a browser and navigate to dvwa.example.com
- Log in with admin/password


If you have any issues with the DVWA database, you can always reset it to the defaults by clicking on the **Setup / Reset DB** button on the left-hand side and then clicking the **Create / Reset Database** button at the bottom of the page. If you are unable to login, you can still reset the database by navigating to <http://dvwa.example.com/setup.php> and clicking on the **Create / Reset Database** button. This will reset the username/password to admin/password. The main two glitches on the DVWA site you could run into and may need to reset the database are:

1. the inability to log in, and
2. the inability to change the Security Level below impossible.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

DVWA Security

PHP Info

About

Logout

Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset.**
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

Setup Check

Operating system: ***nix**
Backend database: **MySQL**
PHP version: **5.5.9-1ubuntu4.21**

Web Server SERVER_NAME: **dvwa.example.com**

PHP function display_errors: **Disabled**
PHP function safe_mode: **Disabled**
PHP function allow_url_include: **Enabled**
PHP function allow_url_fopen: **Enabled**
PHP function magic_quotes_gpc: **Disabled**
PHP module gd: **Installed**
PHP module mysql: **Installed**
PHP module pdo_mysql: **Installed**

MySQL username: **root**
MySQL password: ***blank***
MySQL database: **dvwa**
MySQL host: **127.0.0.1**

reCAPTCHA key: **6LdK7xITAAzzAAJQTfL7fu6I-0aPI8KHHeAT_yJg**

[User: root] Writable folder /var/www/html/hackable/uploads/: **Yes**
[User: root] Writable file /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: **Yes**

Status in red, indicate there will be an issue when trying to complete some modules.

Create / Reset Database

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX

dvwa.example.com/setup.php

Setup DVWA

Instructions

About

Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset.**
You can also use this to reset the administrator credentials ("admin // password") at any stage.

Setup Check

Operating system: ***nix**
Backend database: **MySQL**
PHP version: **5.5.9-1ubuntu4.21**

Web Server SERVER_NAME: **dvwa.example.com**

PHP function display_errors: **Disabled**
PHP function safe_mode: **Disabled**
PHP function allow_url_include: **Enabled**
PHP function allow_url_fopen: **Enabled**
PHP function magic_quotes_gpc: **Disabled**
PHP module gd: **Installed**
PHP module mysql: **Installed**
PHP module pdo_mysql: **Installed**

MySQL username: **root**
MySQL password: ***blank***
MySQL database: **dvwa**
MySQL host: **127.0.0.1**

reCAPTCHA key: **6LdK7xiTAAzAAJQTIL7tu6l-0aPI8KHHeAT_yJg**

[User: root] Writable folder /var/www/html/hackable/uploads/: **Yes**
[User: root] Writable file /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: **Yes**

Status in red, indicate there will be an issue when trying to complete some modules.

Create / Reset Database

For this task we need to log in to the DVWA and set the security level to low. (**Reminder:** login credentials are: admin/password.) Then click the XSS (Stored) button on the left column.

XSS (DOM)

XSS (Reflected)

XSS (Stored)

DVWA Security

PHP Info

About

Low ▼ Submit

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security tool that monitors and filters any user supplied input against DVWA to serve as a live example of how Web Application some cases how WAFs can be circumvented.

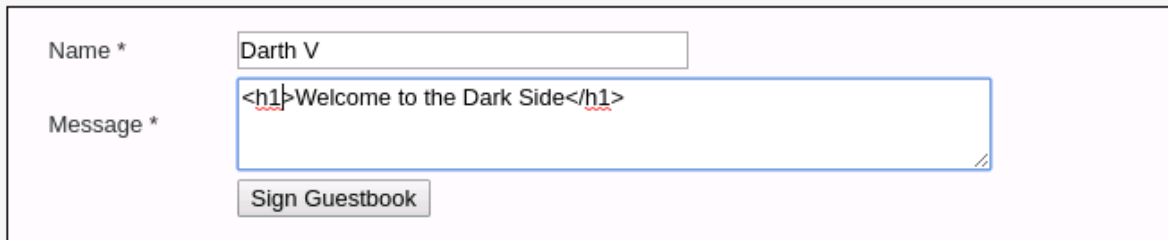
- In the form, type a name in the name submission box. Then type a message wrapped in an html h1 tag: `<h1>message here</h1>` and then click on the **Sign Guestbook** button.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

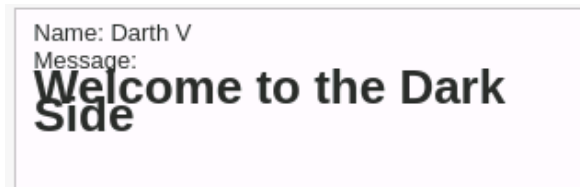
Term: (Fall, Spring, Summer, Winter) 20XX

Vulnerability: Stored Cross Site Scripting (XSS)



Name *

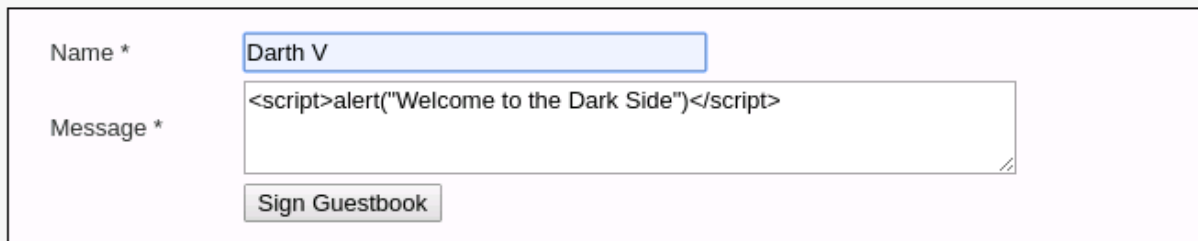
Message *



Name: Darth V
Message:
Welcome to the Dark Side

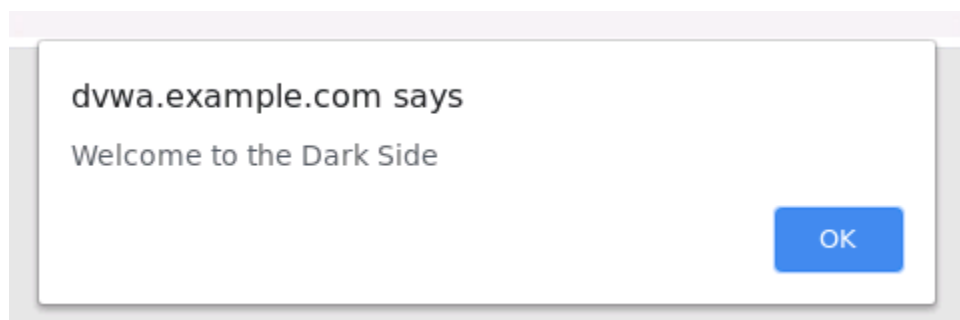
Notice that the `<h1>` tag has changed our message to be a heading. This means we can inject HTML. Let's try a script tag next. Type the following: `<script>alert("Welcome to the Dark Side")</script>` and click on the **Sign Guestbook** button.

Vulnerability: Stored Cross Site Scripting (XSS)



Name *

Message *



dvwa.example.com says
Welcome to the Dark Side

OK

Click on the **OK** button. From here, it seems logical to grab the session cookie. Type `<script>alert(document.cookie)</script>` and click on the **Sign Guestbook** button. You should receive both prompts as they are stored on the page.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

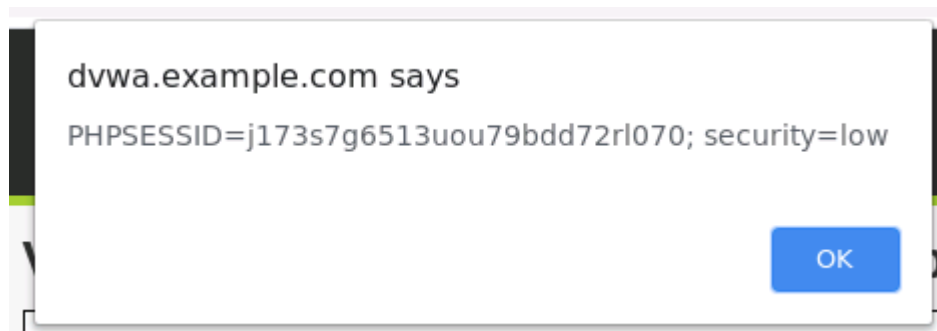
Term: (Fall, Spring, Summer, Winter) 20XX

Vulnerability: Stored Cross Site Scripting (XSS)



Name *

Message *



To be sure the XSS is stored on the page, click on any other DWVA page and then click on the XSS (Stored) page again. Both of the XSS prompts should appear one at a time after clicking OK each time you see a pop up. If an attacker can get Stored XSS on a site, they can potentially gather all session cookies and even credentials by setting up a JavaScript keylogger.

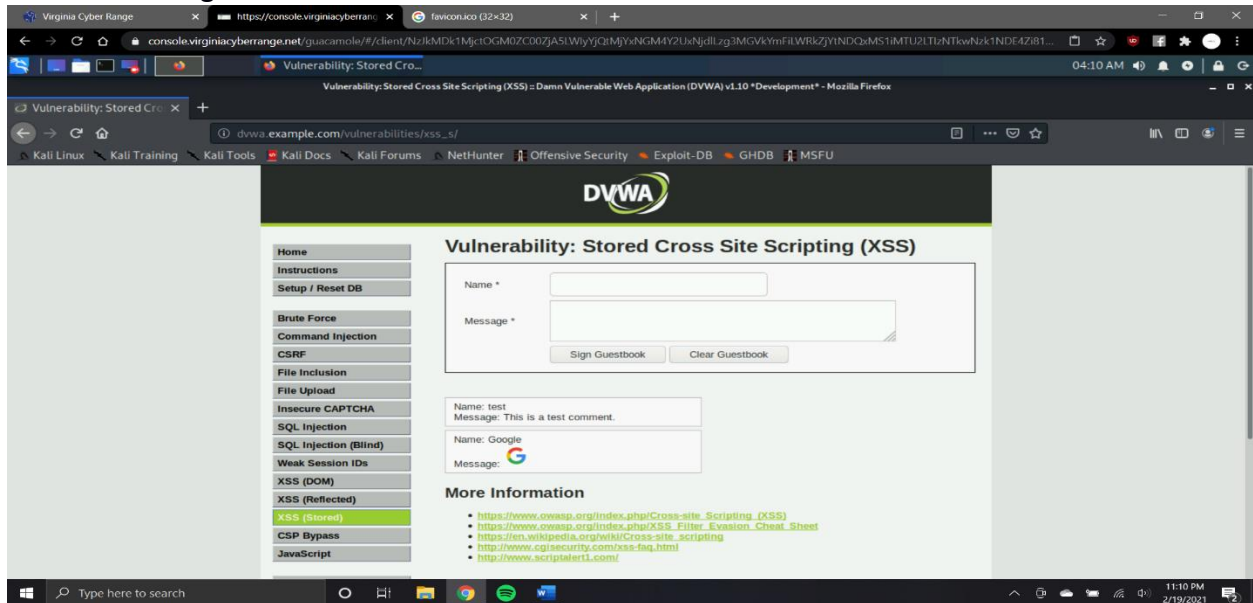
Question 1: do some research online regarding basic JavaScript syntax. And then submit another message to display the image (<https://www.google.com/favicon.ico>) on the Stored XSS webpage. Explain what you have done to make it happen, and provide a screen copy of the webpage showing you have successfully inserted the image into the page.

The text that I found was :

```

```

and this successfully displayed the given image in the stored XSS webpage. The “` was sufficient to get the answer because it fit the message box.



Task 2: DVWA Stored XSS on Medium Security

Reset the DB as shown in the first page. After reset, go to the XSS(stored) page and verify all previous messages that you have posted have been removed.

On the sidebar menu, click on the DVWA Security button and use the dropdown menu to change the security settings to Medium. Be sure to click the **Submit** button. Sometime you may have to do this more than once for it to work. Check the bottom of the page to be sure the settings have changed to medium.



[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX

Logout

Username: admin
Security Level: medium
PHPIDS: disabled

You can enable PHPIDS across this site for the duration of
this course. [\[Enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

Try to use the previous technique by directly inserting a script, i.e., putting `<script>alert("Welcome to the Dark Side")</script>` in the message box, you should find the script will not be executed any longer.

Navigate to the Stored XSS page and click OK on the alerts you created in the first task. If we look at the source by clicking on the **View Source** button located on the bottom right of the page, we can see there are a few changes to the security settings. The name submission box has been filtered by replacing the string `<script>` with double quotations.

Vulnerability: Stored Cross Site Scripting

Name *

Message *

Sign Guestbook

Name: test
Message: This is a test comment.

Name: Darth V
Message:

Name: Darth V
Message:

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting
- https://www.owasp.org/index.php/XSS_Filter_Evasion
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptriot.com/>

Stored XSS Source

```
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string(
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ?
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string(
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ?
    $name = htmlspecialchars( $name );

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '
```

In order to still exploit this, we need to use an exploit that does not require `<script>`. Let's try to place a script in to the name submission box.

- Close out of viewing the source, if you have not already done so.
- In the name submission box, type: `<body onload=alert("medium")>`

7

© 2020 Virginia Cyber Range. Created by R. Eric Kiser. [\[CC BY-NC-SA 4.0\]](#)

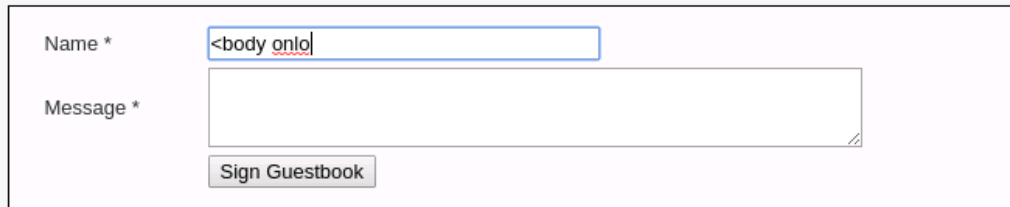
MR
KISER

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX

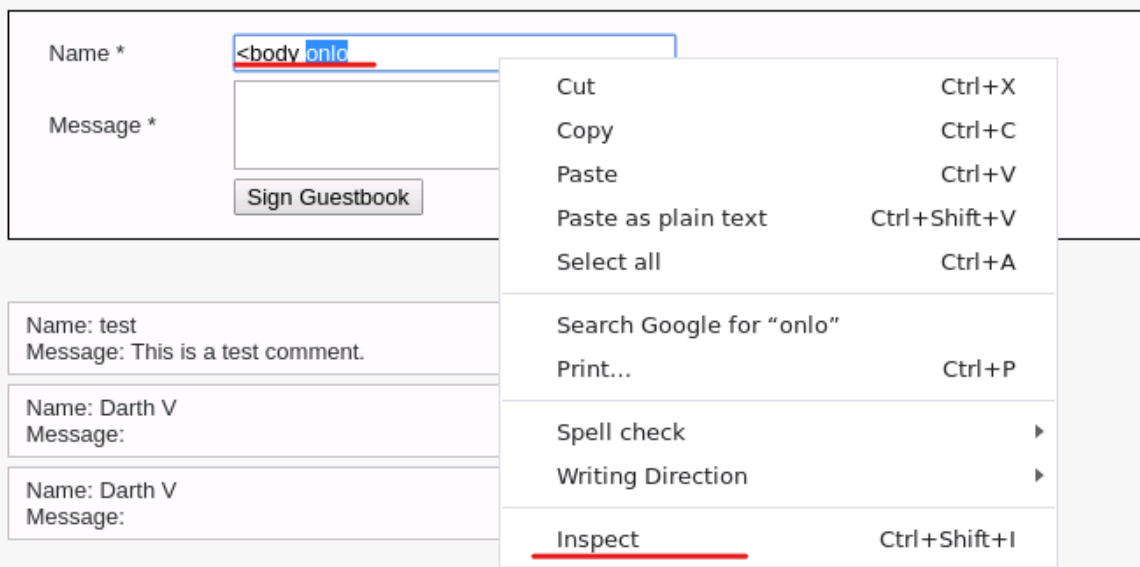
Vulnerability: Stored Cross Site Scripting (XSS)



As you can see the name submission box is only allowing 10 characters. We may be able to modify this using the developer tools.

- Right click on the Name submission box and click on Inspect.

Vulnerability: Stored Cross Site Scripting (XSS)



```
▼ <tr>
  <td width="100">Name *</td>
  ▼ <td>
    <input name="txtName" type="text" size="30" maxlength="10">
  </td>
</tr>
```

Take notice of the **maxlength="10"**. Double click on the 10 and change it to 100. Since the check for this is on the client side and not the server side we are able to bypass the limitation. Let's try our exploit again.

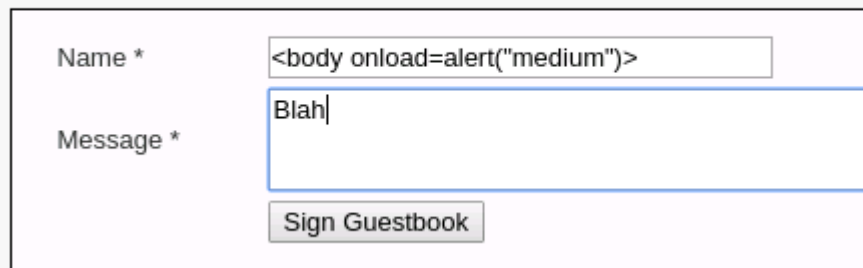
- In the name submission box, type `<body onload=alert("medium")>` and in the message submission box type any message you would like. Finally, press the **Sign Guestbook** button.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX

Vulnerability: Stored Cross Site Sc



Name *

Message *

Once again, we can test to see if the XSS is stored by navigating to any other page then back to the XSS (Stored) page.

dvwa.example.com/vulnerabilities/xss_s/

dvwa.example.com says
medium

OK

Question 2: Do a research online, find a second way, other than the <body> tag given in the example, that can also get the script executed. Explain what you have find, show that you have successfully override the length limit, and show that you have successfully have the script execute and the “medium” message popped out (you will at least have some verbal explanation and 2 screenshots).

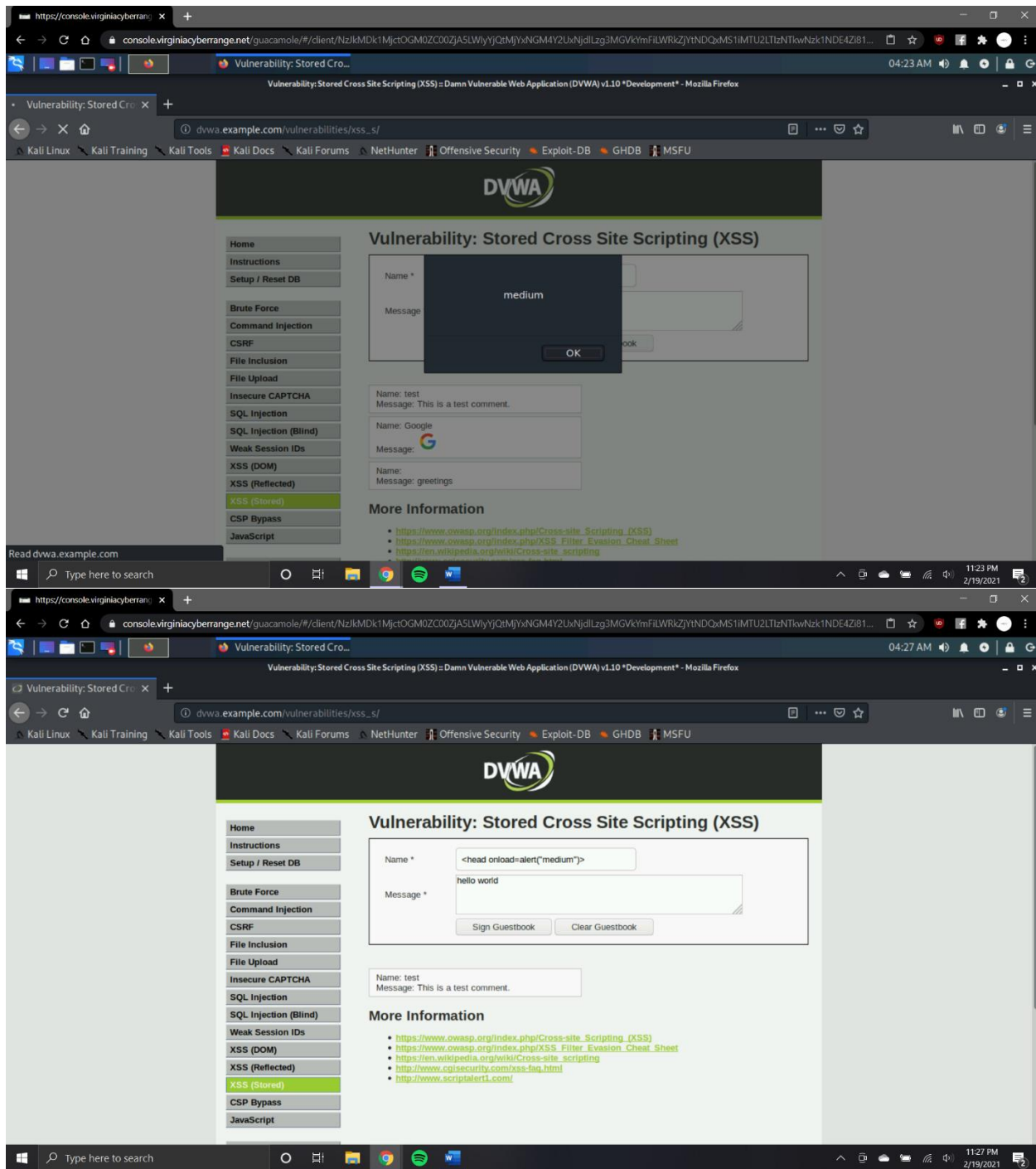
Hint: you can get some inspiration from here <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>

In my research I learned that scripts can be placed in both the <body> and <head> section of an HTML page. So in this example I used <head> instead of <body>. The examples of my overriding the length limit using Inspect Element and the popup “medium” message is shown below.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX



[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX

Question 3: Do a research online and find out the usage and syntax of the “document.location” attribute in JavaScript. And use this attribute to direct the DVWA webpage to <https://www.google.com>. In your response, you must clearly explain each step you have done to make this happen (such as what settings/configurations that you have changed, and what message you have entered, and where have you entered such message), and the final result of your redirection. Include all necessary screen shots to support your statement.

The Document.location attribute returns a location object, which contains information about the URL of the document and provides methods for changing that URL and loading another URL.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

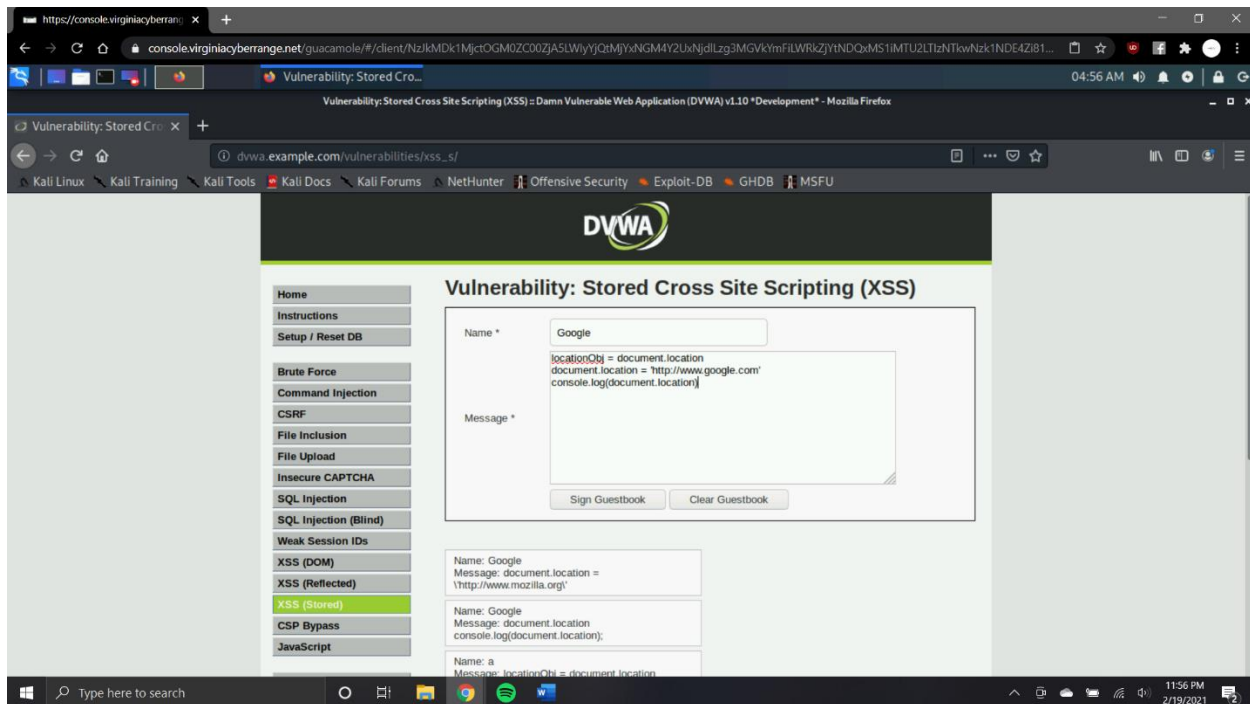
Term: (Fall, Spring, Summer, Winter) 20XX

The screenshot displays the 'Vulnerability: Stored Cross Site Scripting (XSS)' page of the Damn Vulnerable Web Application (DVWA). The page is accessed via a browser at the URL `https://console.virginiacyberrange.net/guacamola/#/client/NzJkMDk1MjctOGM0ZC00ZjA5LWlyYjQmYjYxNGM4Y2UxNjdlZG3MGVYmFILWRkZjYlNDQxMS1IMTU2LTZlNTkwNzk1NDE4ZiB1...`. The page features a sidebar with navigation links such as 'Home', 'Instructions', 'Setup / Reset DB', 'Brute Force', 'Command Injection', 'CSRF', 'File Inclusion', 'File Upload', 'Insecure CAPTCHA', 'SQL Injection', 'SQL Injection (Blind)', 'Weak Session IDs', 'XSS (DOM)', 'XSS (Reflected)', 'XSS (Stored)', 'CSP Bypass', 'JavaScript', 'DVWA Security', 'PHP Info', 'About', and 'Logout'. The main content area shows a form with 'Name' and 'Message' fields. The 'Name' field contains 'Google'. The 'Message' field contains a JavaScript payload: `locationObj = document.location; document.location = console.log(document.location);`. Below the form, there is a list of previous submissions and a 'More Information' section with links to OWASP and other resources. The browser's developer tools are open, showing the HTML structure of the page. The HTML structure includes a table with a row for the current submission and a row for the previous submission. The current submission's message is highlighted in blue. The developer tools also show the CSS styles for the page, including the 'margin' and 'border' properties.

[Any blue text should be replaced by instructor using material and font color changed to black.]

Course Title

Term: (Fall, Spring, Summer, Winter) 20XX



The research I did also stated that the syntax was supposed to start with setting a variable called `locationObj` and setting it equal to `document.location`. This was then followed by a line that set the attribute `document.location` equal to the URL `'https://www.google.com'`. The final line shows the printing of a location object to the console which is `console.log(document.location)`. To be honest, even when I searched up `document.location` syntax, there seemed to be less than 10 websites and/or articles on the topic and instead there were a high volume of resources for the object `window.location` which has similar usages and syntax.