# Lab-11-2: Cross Site Scripting (XSS)

In today’s lab, we are following on from the injection attacks we performed last class. However, we are looking at totally different type of injection called Cross Site Scripting (or XSS).

**CHECK YOU HAVE THE FOLLOWING VIRTUAL MACHINES BEFORE STARTING:**

1. **Kali Linux**
2. **Ubuntu-WebApps**

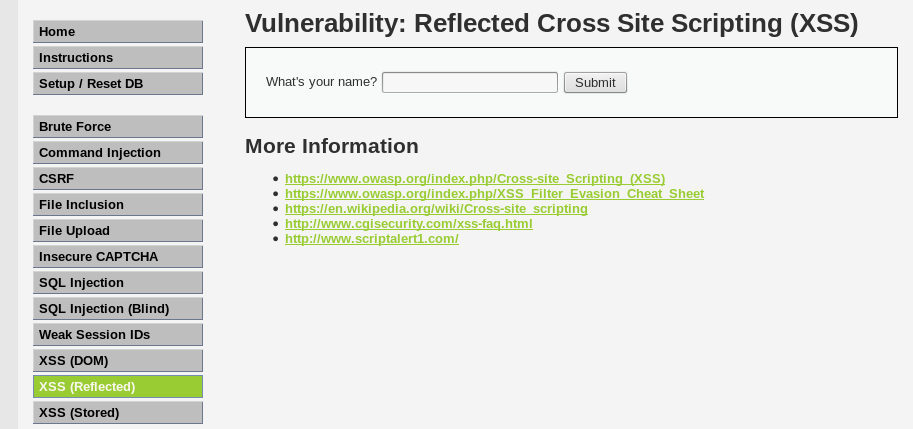
Run the following script if you don’t: L:\Virtual Machines\IN618\Lab-11-2.bat

**Exercise-11-2-1: XSS Reflection (low)**

Like the exercise in lab-11-1, we be using DVWA to practice XSS attacks and mitigation.

Let’s get started:

1. Start Ubuntu-WebApps
2. Start Kali-Linux
3. In Kali-Linux, open Firefox
4. Navigate to: <http://192.168.19.123/DVWA/>
5. Login with the “admin” and “password”
6. Select the XSS (Reflected) exercise
7. You should see an exercise like the image below
8. Ready!



As the other exercises, we have done – check the “View Help” button for information on the exercise. When ready, close the help dialog box.

The input box is asking you to enter you name. Try entering your name, and view the output.

We should check out the source code to see what the website is doing behind the scenes:

To help you out, the code is below:

<?php

// Is there any input?

if( array\_key\_exists( "name", $\_GET ) && $\_GET[ 'name' ] != NULL ) {

// Feedback for end user

echo '<pre>Hello ' . $\_GET[ 'name' ] . '</pre>';

}

?>

**Q1.** What type of HTTP request is the PHP script performing?

GET

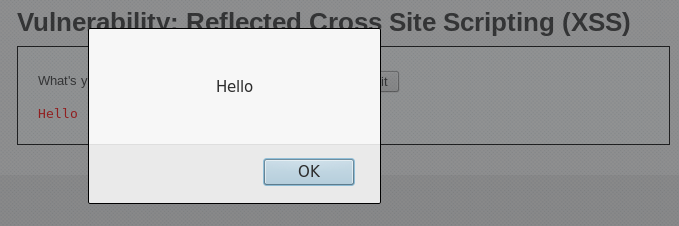
**Q2.** Is the PHP performing any checks of the data supplied by the user?

It is performing 2 checks on the input

To get started, here is a script we could inject to get an alert box to popup and say hello:

<script>alert(“HELLO”)</script>

Try writing the code in into the web application input box (copying and pasting may not work). Hopefully, you will get a dialog popup similar to the one below:



**Congratulations! You have performed an XSS attack… but it doesn’t do much!**

During the lecture, we talked about different XSS attacks. Specifically, we talked about stealing a cookie from a user and gave an example of displaying a cookie using an alert. Go back to the slides from today and try to find some JavaScript that we could use to generate an alert and display our own cookie.

**Q3.** Document the JavaScript code you used to get the website cookie.

<script>alert(document.cookie)</script

Hmmmm, that was not very malicious. Maybe we should try something else to get a real idea about the power of XSS.

We are going to change the header of the webpage. We could just as easily change the images, layout or style. Before we enter the script below, we should make sure we understand each part.

<script>document.getElementsByTagName('H1')[0].innerHTML="You have been hacked by an evil hacker"</script>

* <script></script>: The tags used to state we have JavaScript code
* Document.getElementsByTagName(‘H1’)[0]: Find the first <h1> HTML header
* .innerHTML=: Change the content of the <h1> header to a new string
* “You have been hacked by and evil hacker”: The new string we are injecting

If you are lost with anything above, ask a fellow student or you lecturer. All we are doing is finding the first HTML element that is a header (e.g., the first <h1> tag), and changing the string to be something else.

Enter the script in the user input box and watch the magic happen!

OPTIONAL: If you are interested and feel like a challenge: try to use Google to find something else you can change using an XSS injection. A couple of examples keyword searches are:

javascript change page title

javascript change h2 element

Remember, you can just change anything. Make sure the HTML element (e.g., <h1> tag, or <title> tag) exists is the page. Or find a way to create a new HTML element. The sky is the limit using XSS attack.

**Exercise-11-2-2: XSS Reflection (medium)**

We are going to switch to the medium exercise. We need a challenge today!

Navigate to the “DVWA Security” page using the link on the menu on the left side of the page.

Set the security level to “Medium” and click submit to save.

Navigate back to the “XSS Reflected” tutorial, this time in medium difficulty.

To save you some time, the updated PHP script is provided below:

<?php

// Is there any input?

if( array\_key\_exists( "name", $\_GET ) && $\_GET[ 'name' ] != NULL ) {

// Get input

$name = str\_replace( '<script>', '', $\_GET[ 'name' ] );

// Feedback for end user

echo "<pre>Hello ${name}</pre>";

}

?>

The code is a little different, it has included the following line as extra code:

$name = str\_replace( '<script>', '', $\_GET[ 'name' ] );

It looks like the web application developer has tried to stop XSS attacks. Specifically, the new code uses the PHP str\_replace function to check the data input by the user. If you want more information on this specific PHP function, check this tutorial: <https://www.w3schools.com/php/func_string_str_replace.asp>

**Q4.** What does this new code do? What happens if we include “<script>” in our input now? Feel free to use an example in you answer.

It will replace the ‘<script>’ tag with an empty string.

Get your brain thinking about how we could get around this new defence that the web application has. How could we still get our JavaScript code to execute?

If you need some help, have a look at the following tutorial:

<http://securenetworkmanagement.com/reflective-xss-dvwa-part-2/>

**Q5.** Enter a XSS attack to get around the medium level. Explain why it works.

<SCripT>alert(“Hello”)</sCrIpT>

This works because php str\_replace is case sensitive to match only ‘<script>’, by replacing the lower case letters with upper case it is easy to get around.

**Exercise-11-2-3: XSS Stored (low)**

We are going to switch to the low exercise – or stay on the medium if you want to. It is the same as the last exercise.

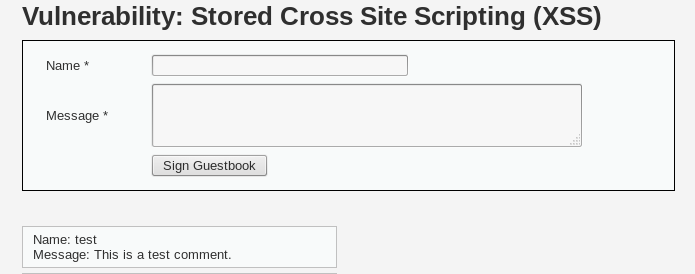
Navigate to the “DVWA Security” page using the link on the menu on the left side of the page.

Set the security level to “low” and click submit to save.

Navigate to the “XSS Stored” tutorial.

To save you some time, the PHP script for this exercise is the same as the low level for the reflected XSS exercise.

Below is an example of the web application you should see:



This is a simple web app that allows you to post comments. Try first by entering your name and a short message. Your information should appear in the comments below.

Now, we are going to change something of the web page and make it persistent (aka an XSS stored attack). Enter the following details:

Name: An elite hacker

Message: <script>document.title="HACKED"</script>

This should change the title of the webpage (the name you see in the web browser tab). Navigate to a couple of different exercises and then go back to the “XSS Stored” exercise. Does the modified title still appear?

**Q6.** Why does the title not revert to the original web page title for the exercise?

Beause the code message is stored in the database, so the XSS code is always run when the page is loaded.