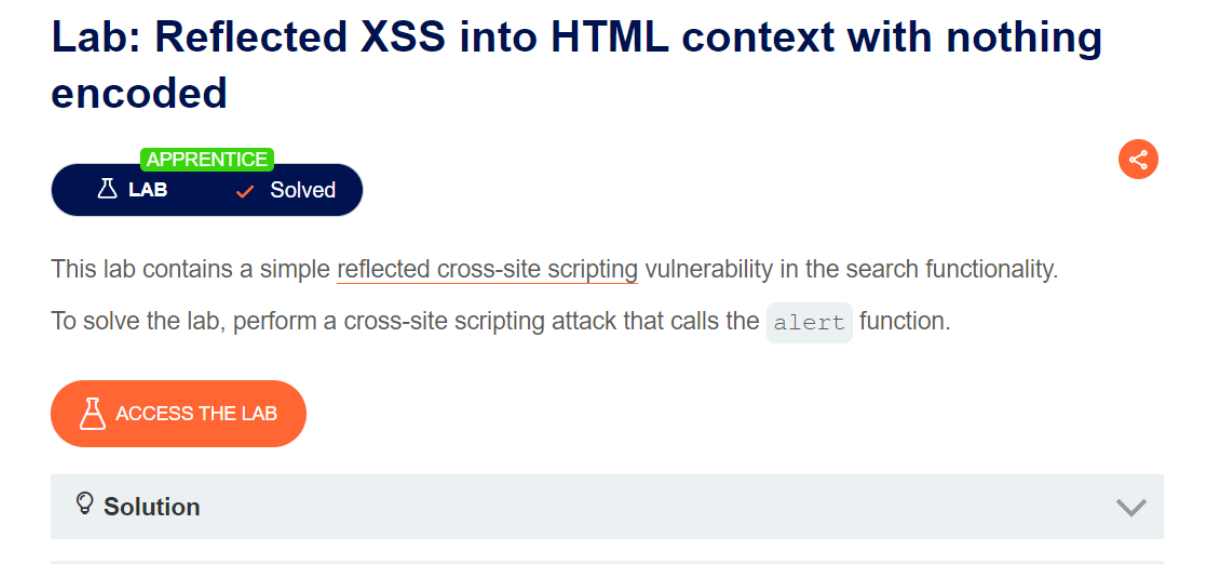
PORT SWIGGER

Cross Site Script (xss)

Cross-Site Scripting (XSS) *is a web security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can execute in the victim's browser, potentially stealing sensitive information, hijacking user sessions, or spreading malware. XSS attacks exploit improper validation of user input and can occur in forms, URLs, or cookies.*

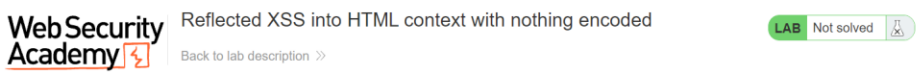
Lab 1: Reflected XSS into HTML context with nothing encoded



This is the first lab of Xss in portswigger. The description explains there is a xss vulnerability in the search functionality of the lab. To solve it we have to perform reflected xss attack that calls the alert function.

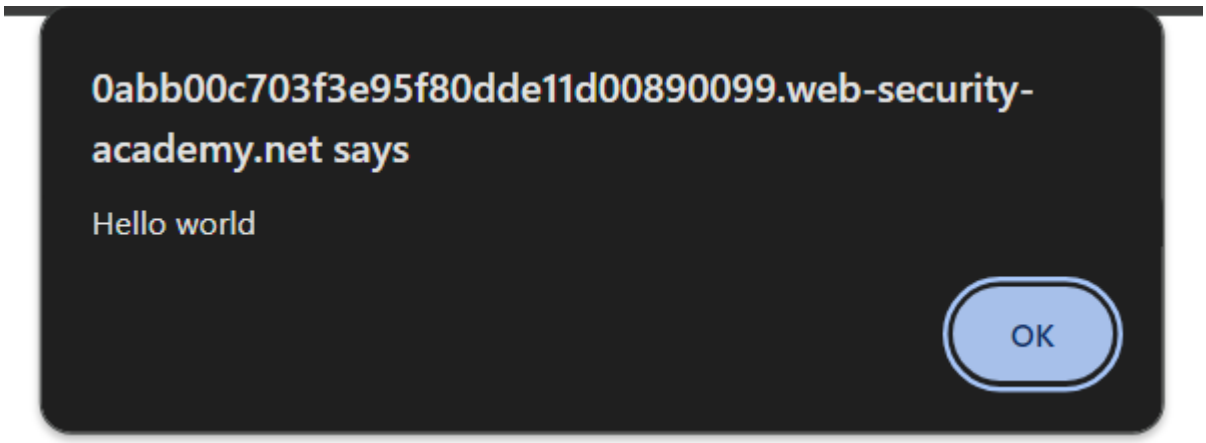
Reflected XSS *is a type of cross-site scripting attack where malicious scripts are immediately reflected off a web server and executed in the victim's browser. This occurs when user-supplied input (like from a form or URL parameter) is included in the response without proper validation or encoding. The attacker typically tricks the victim into clicking a crafted link that sends the malicious script to the server, which then reflects it back in the response. The script runs in the victim's browser, potentially allowing the attacker to steal sensitive information or perform actions on behalf of the victim.*

Let’s access the lab,



There is a search functionality in the website, let’s input any simple javascript code on the search bar.

<script>alert("Hello world")</script>



After entering this code on the search bar and hitting search a pop up will show the “Hello world” on the screen.

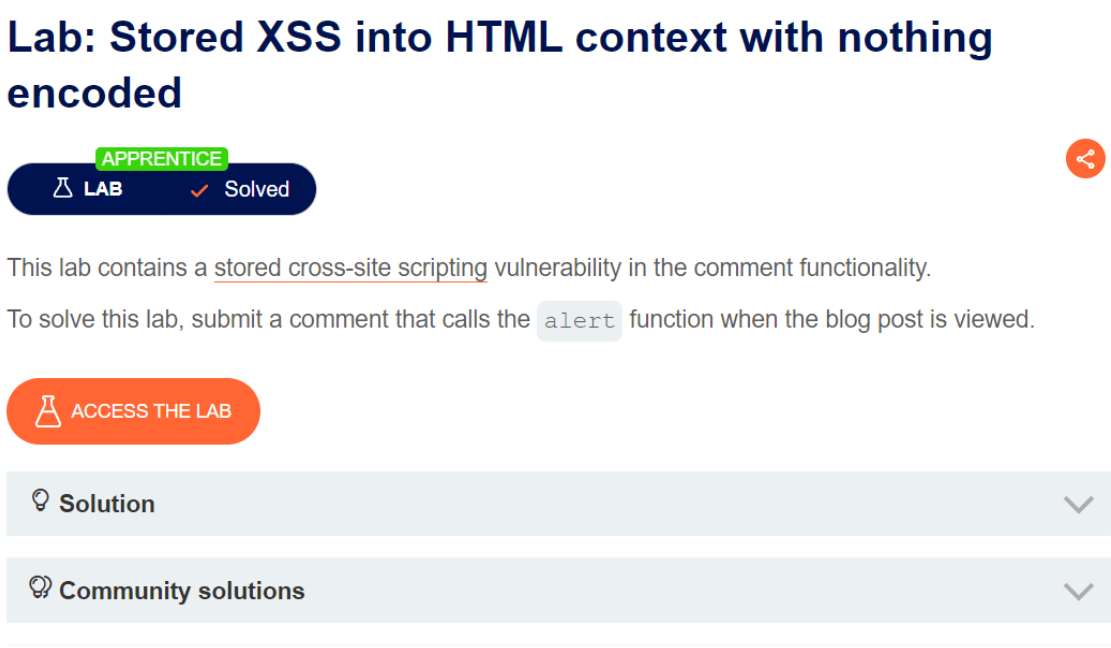
Notice the end of the URL:

/?search=<script>alert(“Hello+World”)<%2Fscript>

After the question mark (?) is where the query string starts.

So we completed this lab 😊

Lab 2: Stored XSS into HTML context with nothing encoded

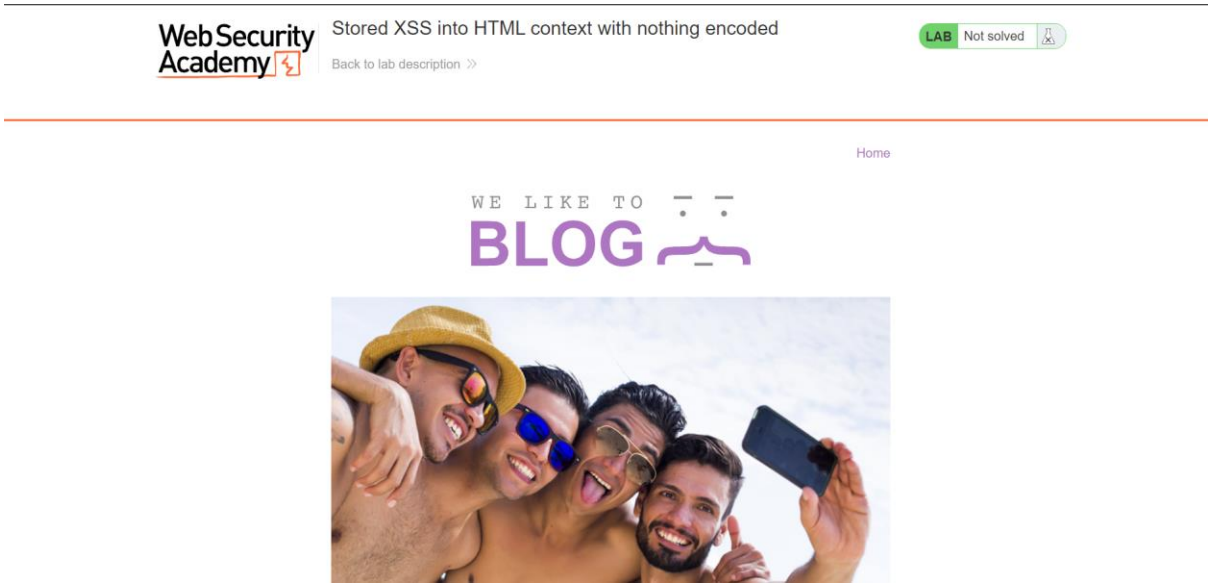


This is the second lab of Xss in portswigger. The description explains there is a xss vulnerability in the comment functionality of the lab. To solve it we have to perform stored xss attack that calls the alert function.

Stored XSS, also known as **persistent XSS**, is a type of cross-site scripting attack where the malicious script is permanently stored on a target server, such as in a database, comment field, or message board. When other users access the affected page, the malicious script is served to their browsers and executed without any interaction from the attacker. This type of XSS can be more damaging than reflected XSS because it affects multiple users who visit the page where the malicious script is stored, leading to widespread exposure and potential data theft or session hijacking.

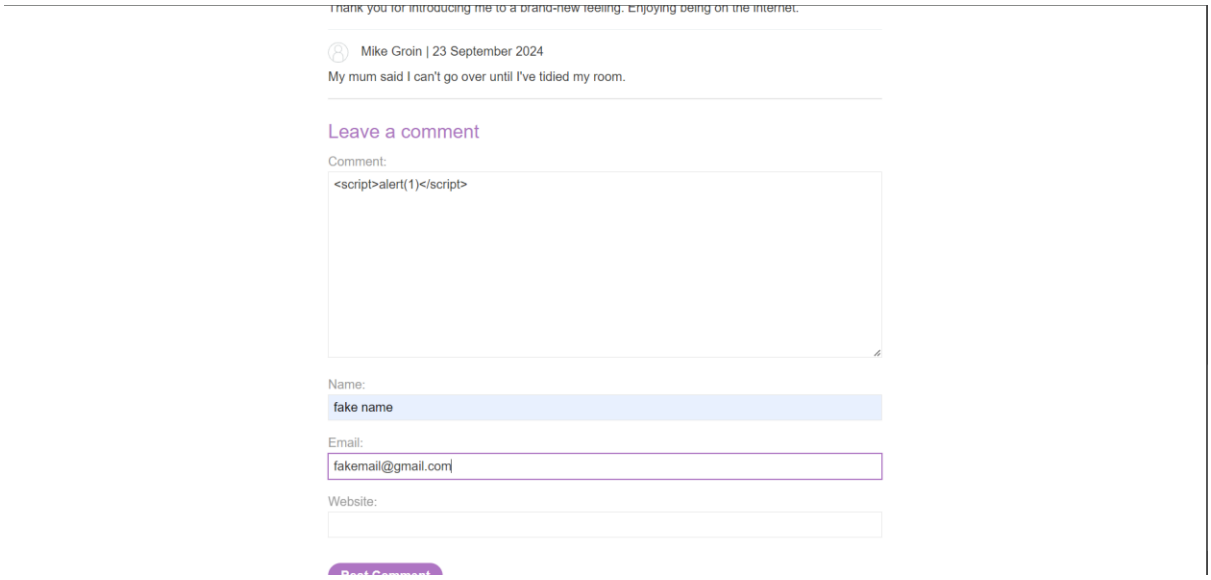
So let’s submit a comment on the blog.

This is the lab,



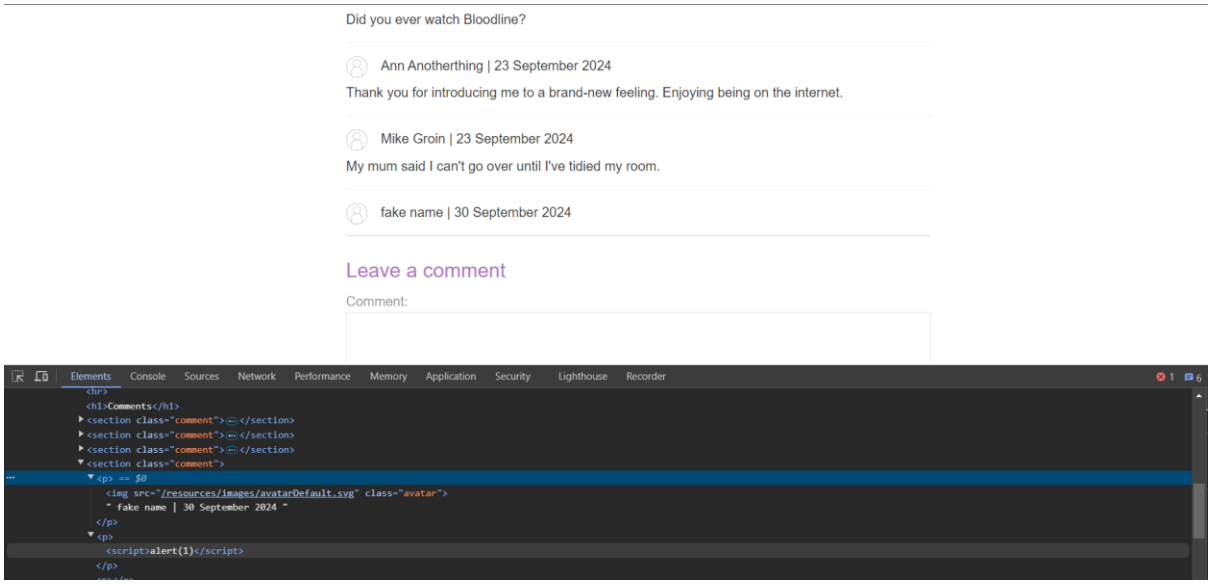
Let’s explore the lab and open a blog, after that submit a simple xss script on the comment section.

<script>alert(1)</script>



After submitting the lab, just refresh the page and you will get a pop message of 1. This means that you have successfully executed the attack.

So when we inspect the comment section we can still see our script



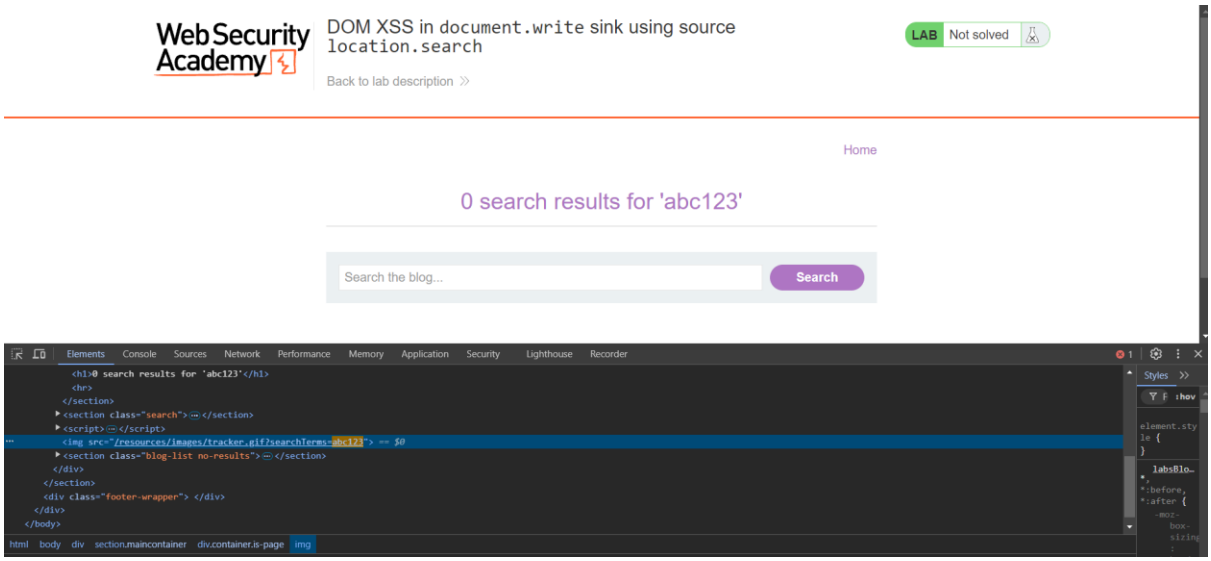
So we completed this lab 😊

Lab 3: DOM XSS in document.write sink using source location.search

This lab contains a DOM-based XSS vulnerability, this is 3rd lab of XSS in portswigger. This lab uses a javascript document.write function, which writes data out to the page. The document.write function is called with data from location.search. We have to solve this lab by performing xss that calls the alert function.

Let's see the lab.

There is a search bar available in the website, let’s type any strings on the search bar and inspect the page.

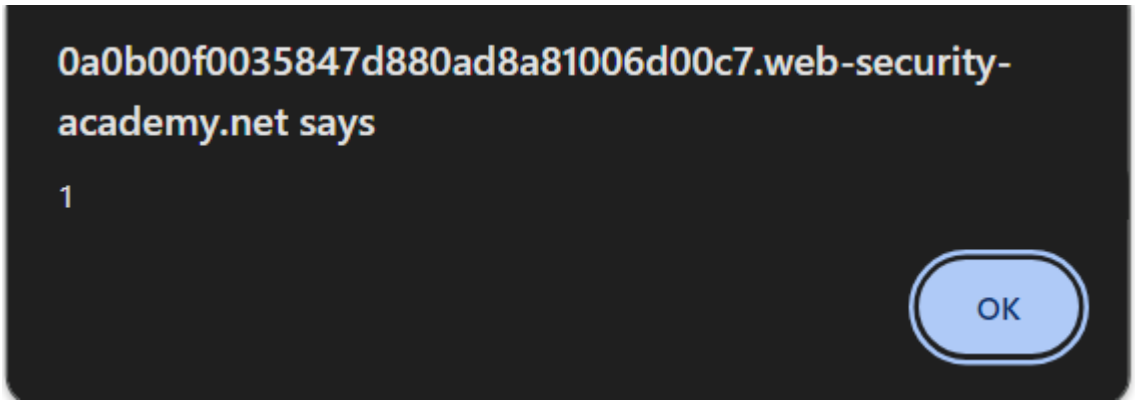


After submitting a random string I inspected the page and found that the <img> < /img> attribute containing the random string I searched.

So let’s break out the img attribute,

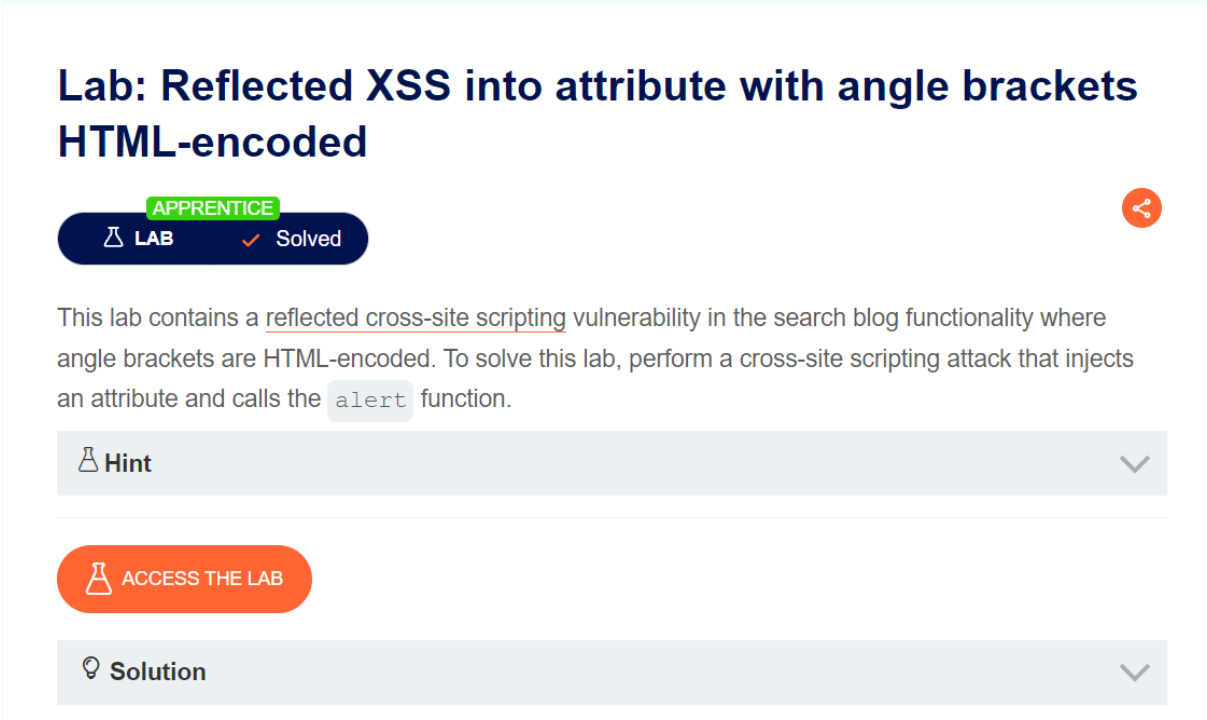
Here is the script,

"><svg onload=alert(1)>



Now we will get a pop of 1, so the Lab is finished

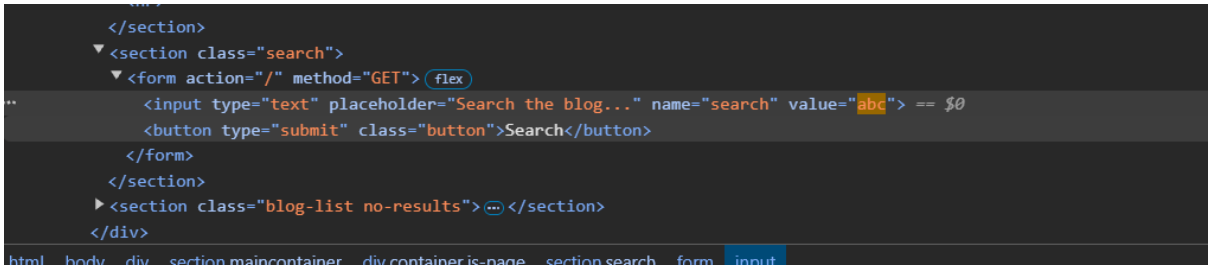
Lab 4: Reflected XSS into attribute with angle brackets HTML-encoded



This lab contains a **reflected xss** vulnerability. It describes that this vulnerability presents on the search functionality of the blog. It is HTML encoded.

Let's access the lab.

This is the lab, let's search something on the search bar and inspect the page.

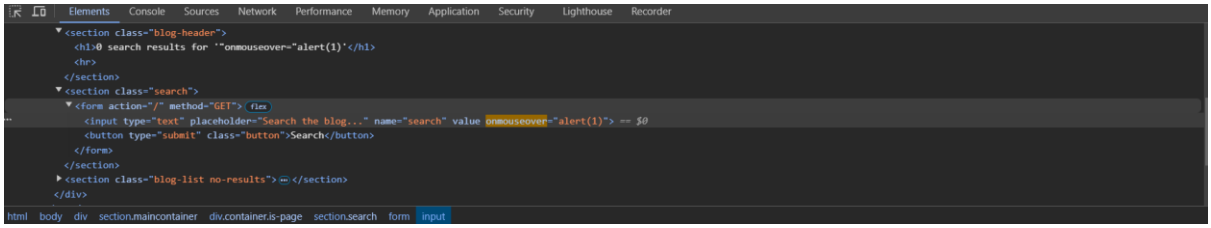


Here we can see the input I searched is **abc** and it is quoted so let's use a event attribute and inject event handler, **"onmouseover="**

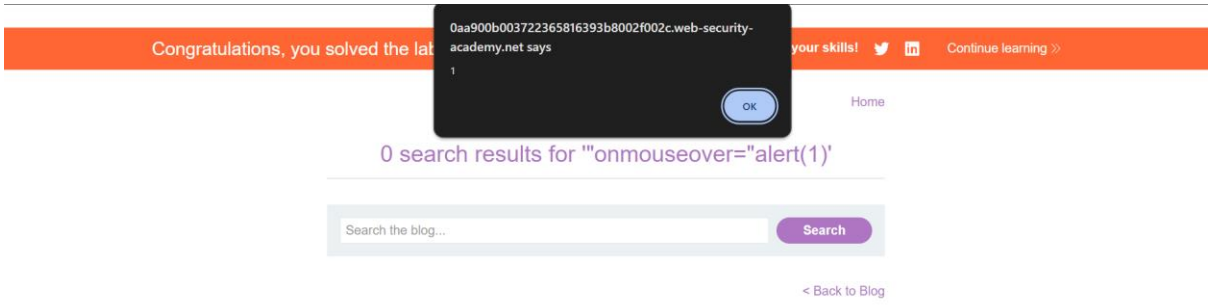
"The onmouseover event occurs when the mouse pointer enters an element"

"onmouseover="alert(1)

Let's see.

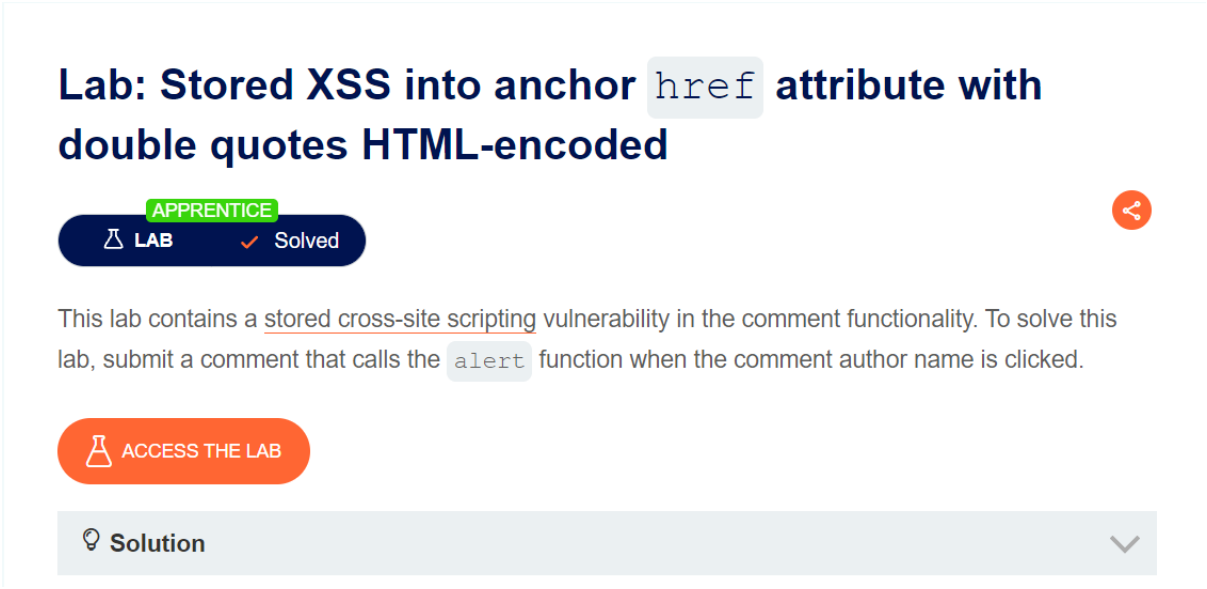


Here you can see that onmouseover event is reflected outside the encoded, hence the lab is completed u can see an alert "1" as coming as pop up.



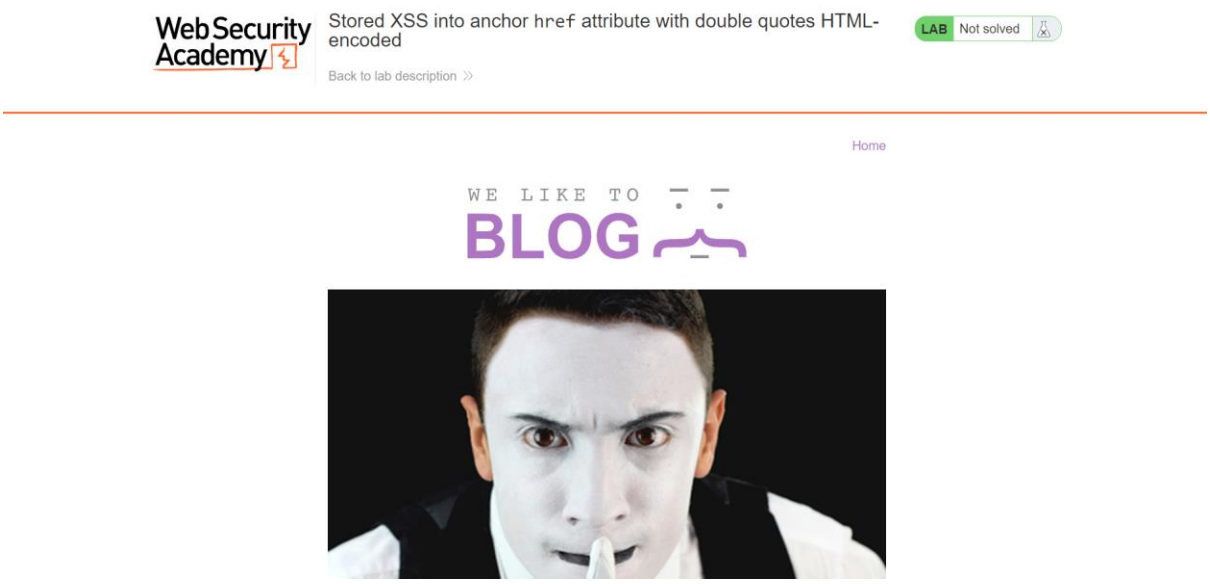
We did completed the lab 😊

Lab 5: Stored XSS into anchor href attribute with double quotes HTML-encoded

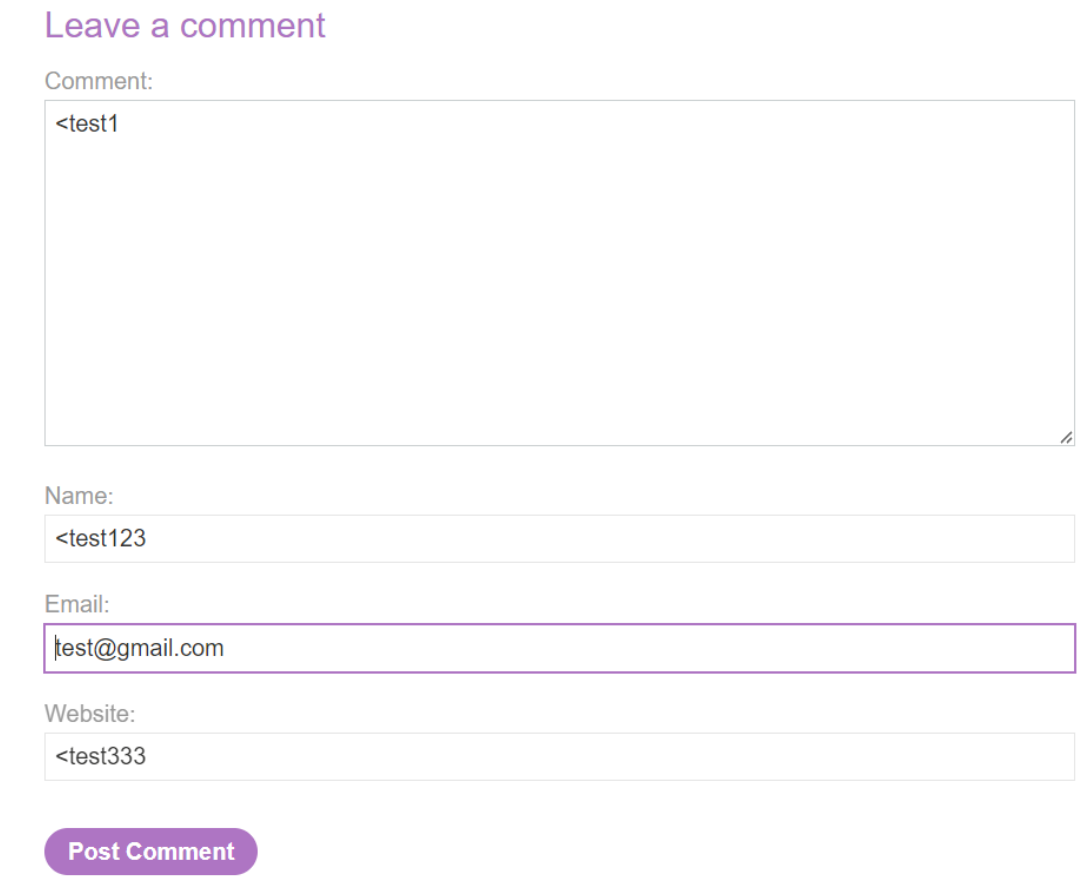


This is a **Stored xss** vulnerability lab in portswigger. From the description we can understand the vulnerability is on the comment functionality. To solve this,we need to call an alert function when the comment authors name is clicked.

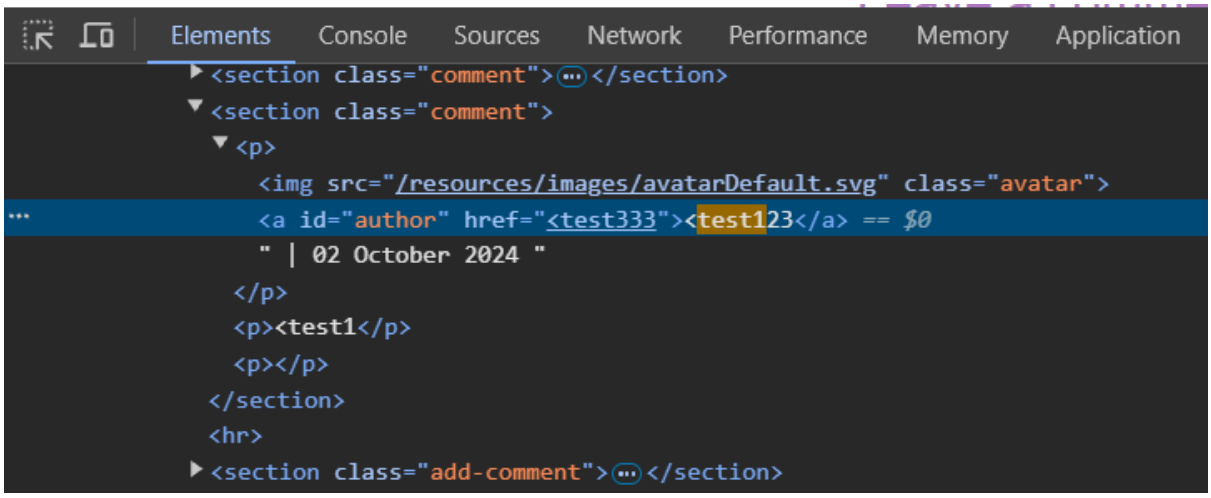
Let's access the lab,



Let's see if the comment section is vulnerable to xss, use this command on the comment section,



After submitting inspect the page after going back to blog.



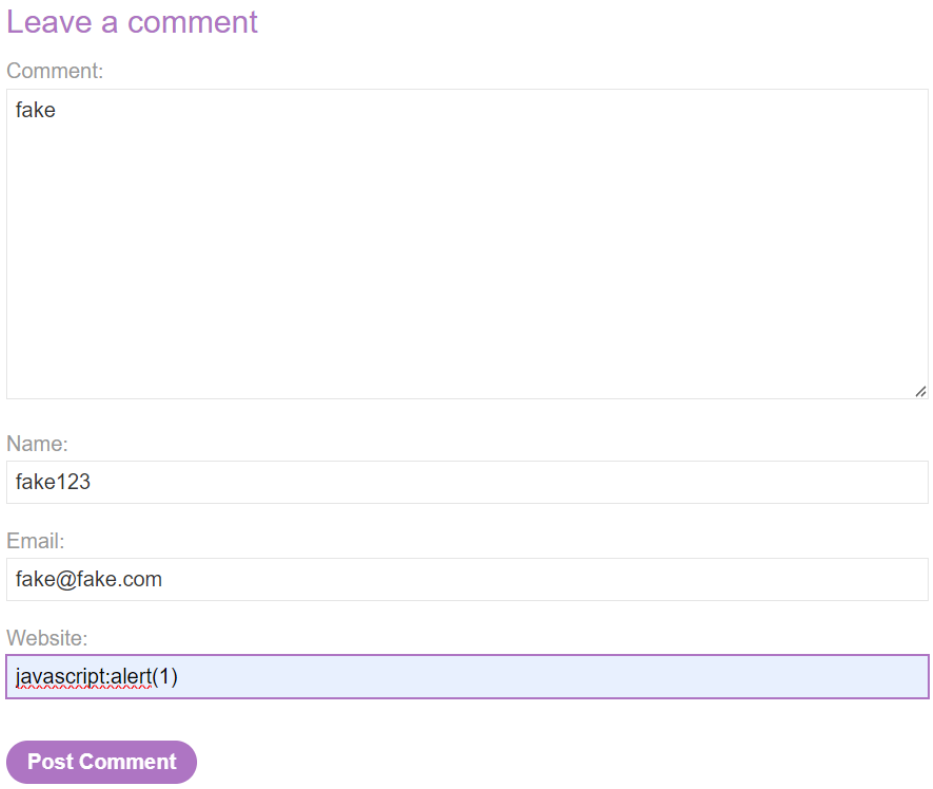
Here we can see the **test333 & test123** is covered on a **href** attribute**.**

“The href attribute specifies the URL of the page the link goes to”

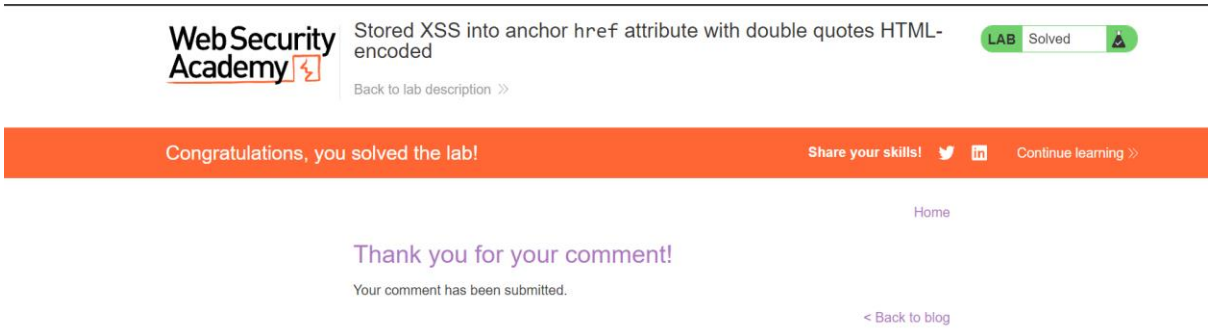
We can inject a **Javascirpt** to the code, which contain an anchor tag and inside the reflection which is occurring inside a href attribute. Try this script,

javascript:alert(123)

So return to the comment field and enter again the sections, and in the website column enter the **Javascript** code.



Enter the section like you see on the screenshot.



Yes we completed the lab 😊

Lab 6: Reflected XSS into a JavaScript string with angle brackets HTML encoded

Lab: Reflected XSS into a JavaScript string with angle brackets HTML encoded

APPRENTICE

LAB ✓ Solved

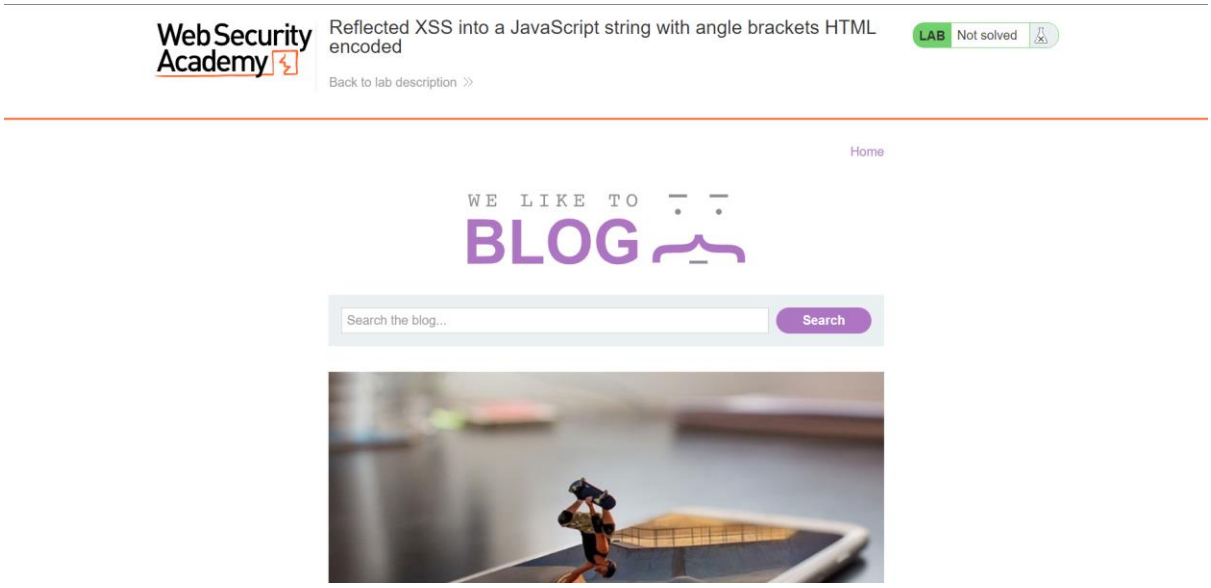
This lab contains a reflected cross-site scripting vulnerability in the search query tracking functionality where angle brackets are encoded. The reflection occurs inside a JavaScript string. To solve this lab, perform a cross-site scripting attack that breaks out of the JavaScript string and calls the `alert` function.

ACCESS THE LAB

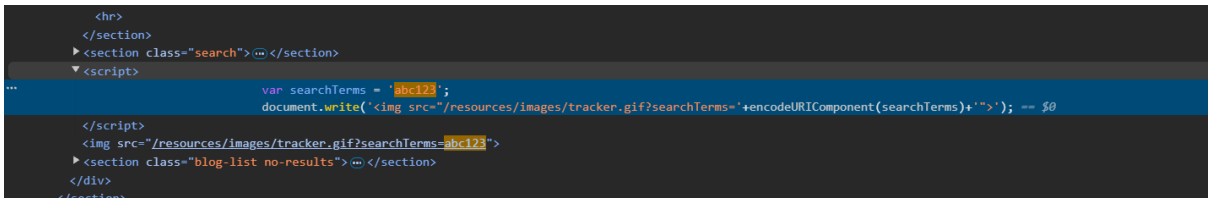
Solution

In this lab the description describes there is a **reflected xss** vulnerability in the search query of the lab. The reflection occurs inside a JavaScript string.

Let's see the lab,

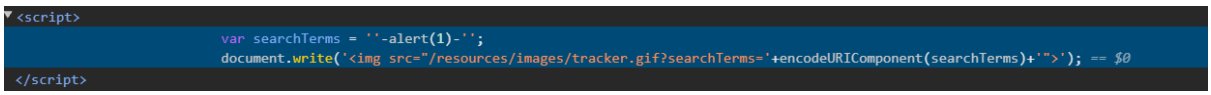


Type any random strings on the search bar, then inspect the page,



Here during the document.write process, all characters are encoded using encodeURIComponent, so XSS won't work from now on. So we needed to add the XSS payload in the place where '**abc123'** was placed.

So let's use this, **'-alert(1)-'**



Here the **'** turns into a character and other part into a comment. And here the problem is solved.

The lab is done 😊