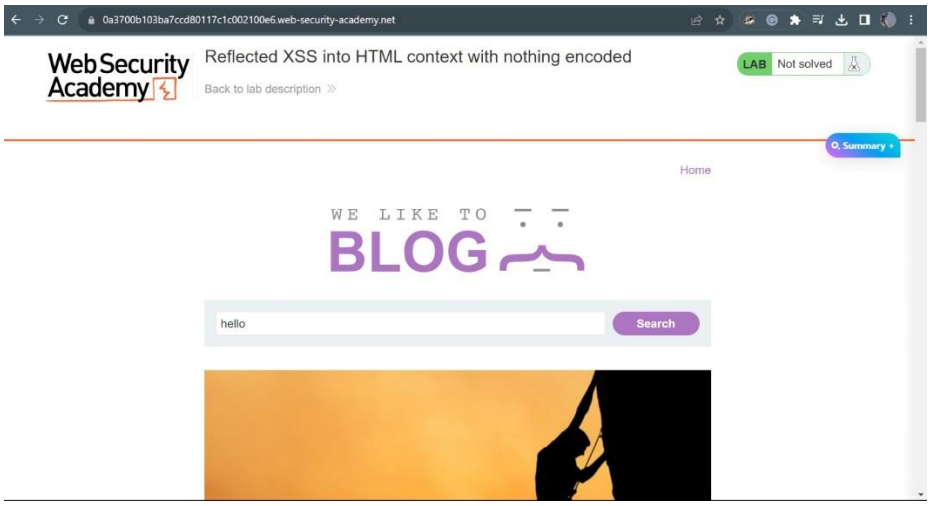
Port-Swiggers labs

1. **Reflected XSS into HTML context with nothing encoded: -**

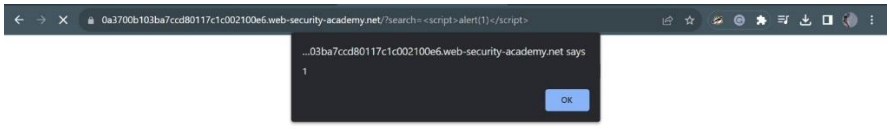
In this lab, we will be performing a reflected XSS attack on the search function.

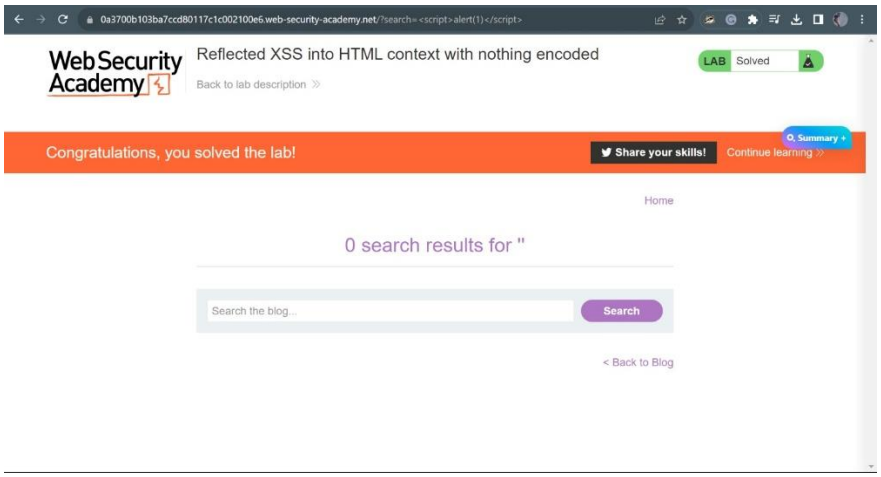


In the search bar we can input a simple alert function from javascript:

<script>alert(“hello”)</script>

When you hit search, a pop-up alert will show your your “hello” message. This means your attack was successful.





We can edit directly in the query string. Below I changed the “hello” into “hello”:/?search=<script>alert(“hello”)<%2Fscript>

Once you hit enter, the pop-up message will be changed.

we have solved this simple lab. Click ok and you will be at the result page.

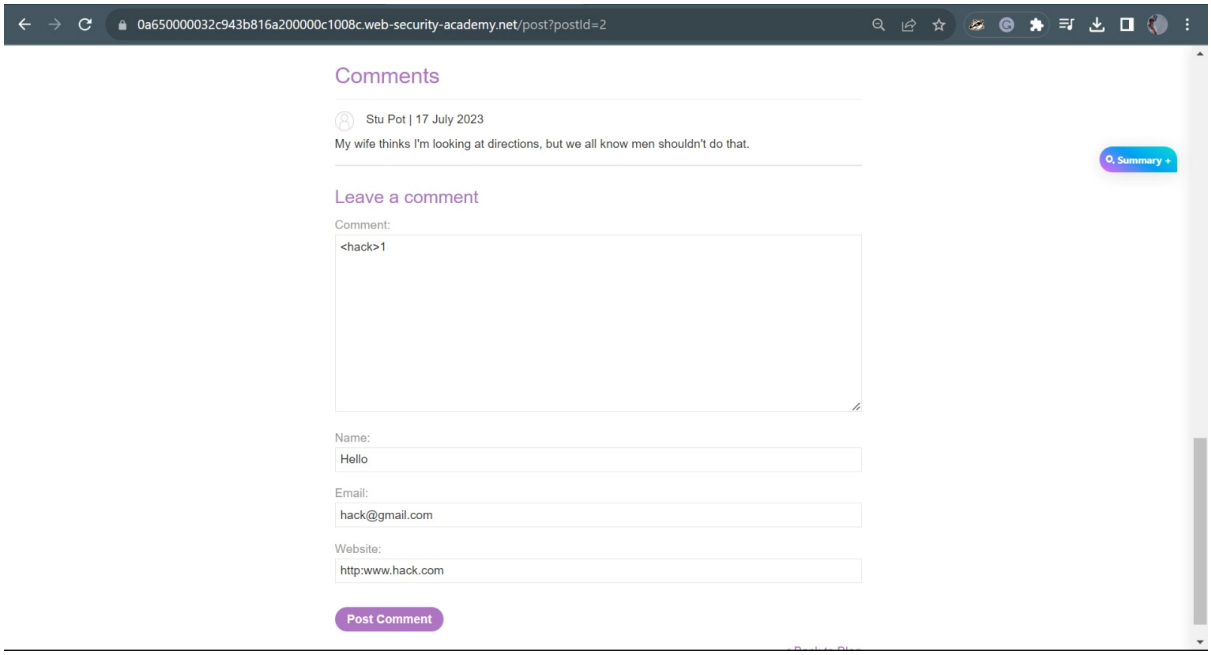
2. Stored XSS into HTML context with nothing encoded: -

In this lab, we will be inserting JavaScript code as a stored XSS attack.

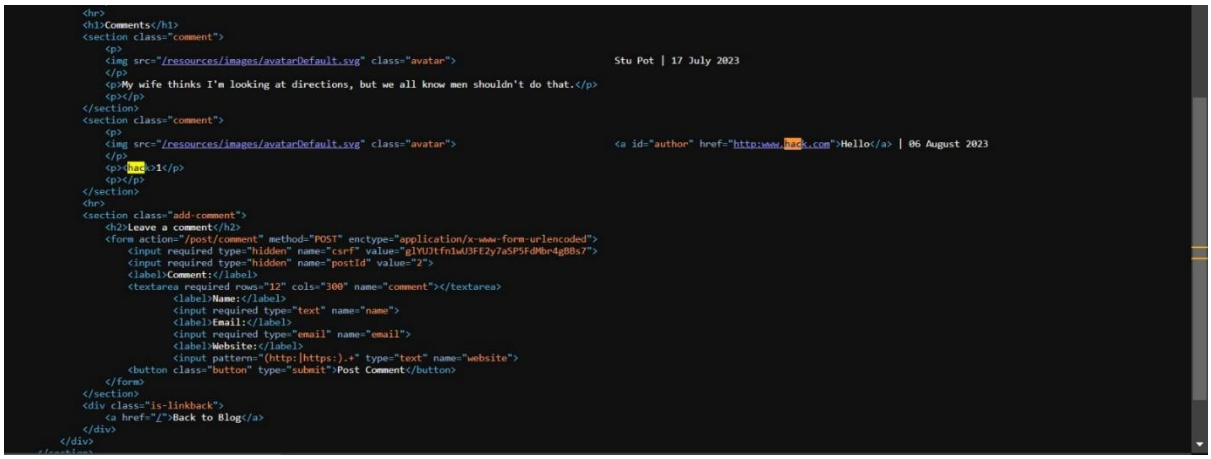
First, let's view one of the blog posts and scroll down to the comment section.

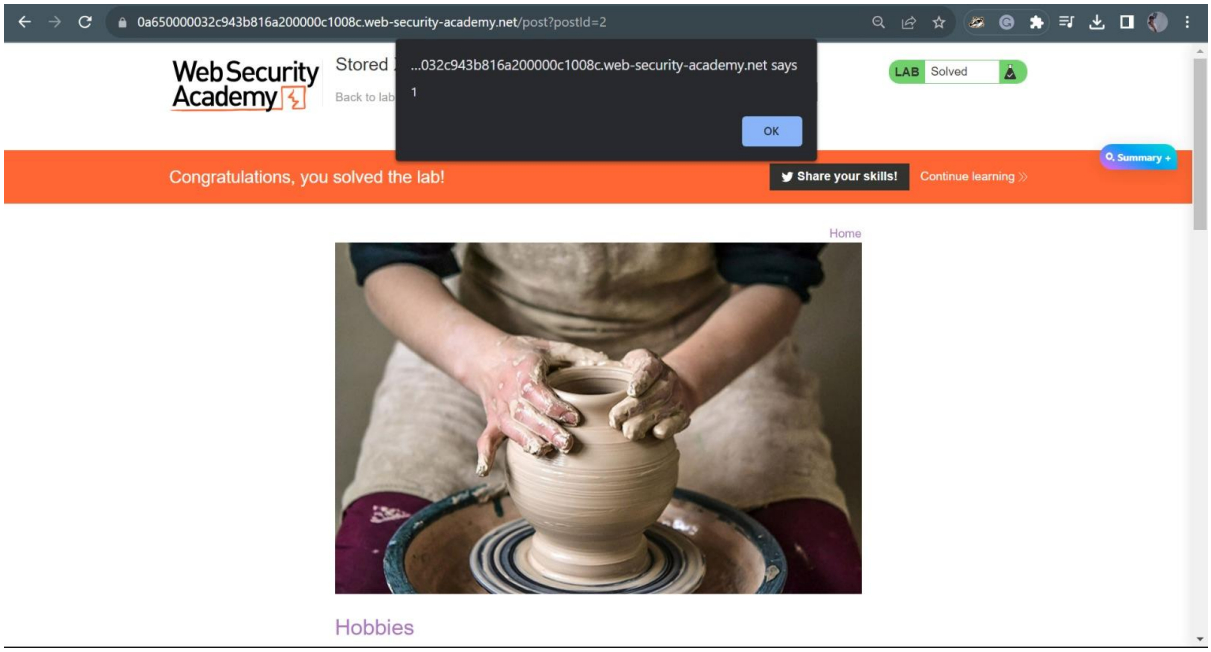
To perform a stored XSS attack, we need to post the comment with the malicious code so that it will be stored in their database. So the next time anyone visiting this page, their web browser will render it while executing the malicious code.

Go ahead and post your comment.



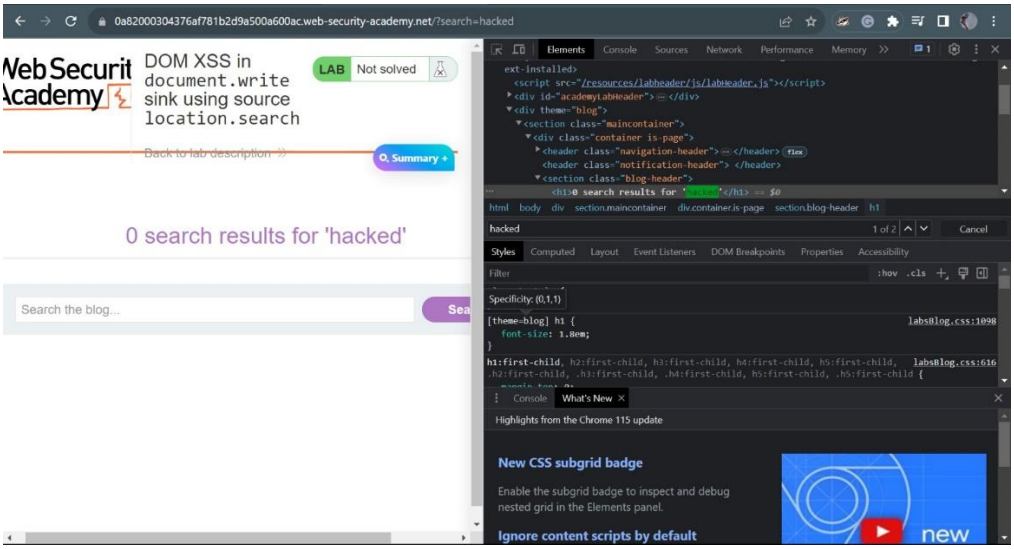
Here we can see that when we return to the same blog post. The pop-up alert with the “hello” message appears. This means that you have successfully executed the attack.





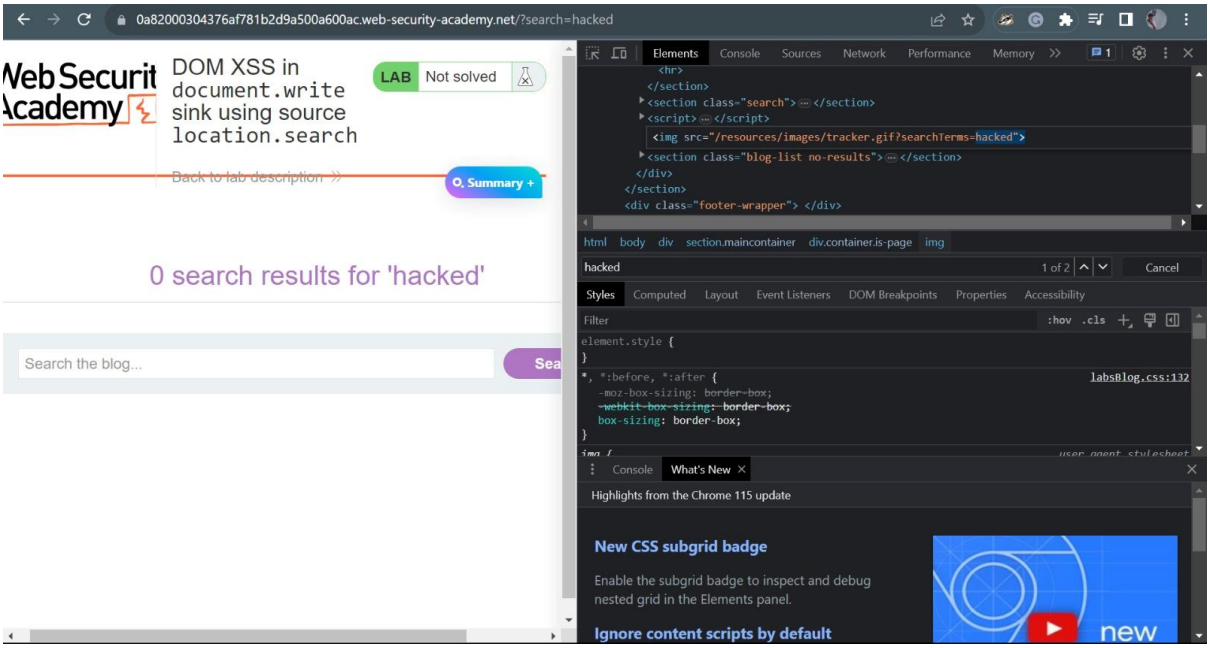
And with this we have solved this lab.

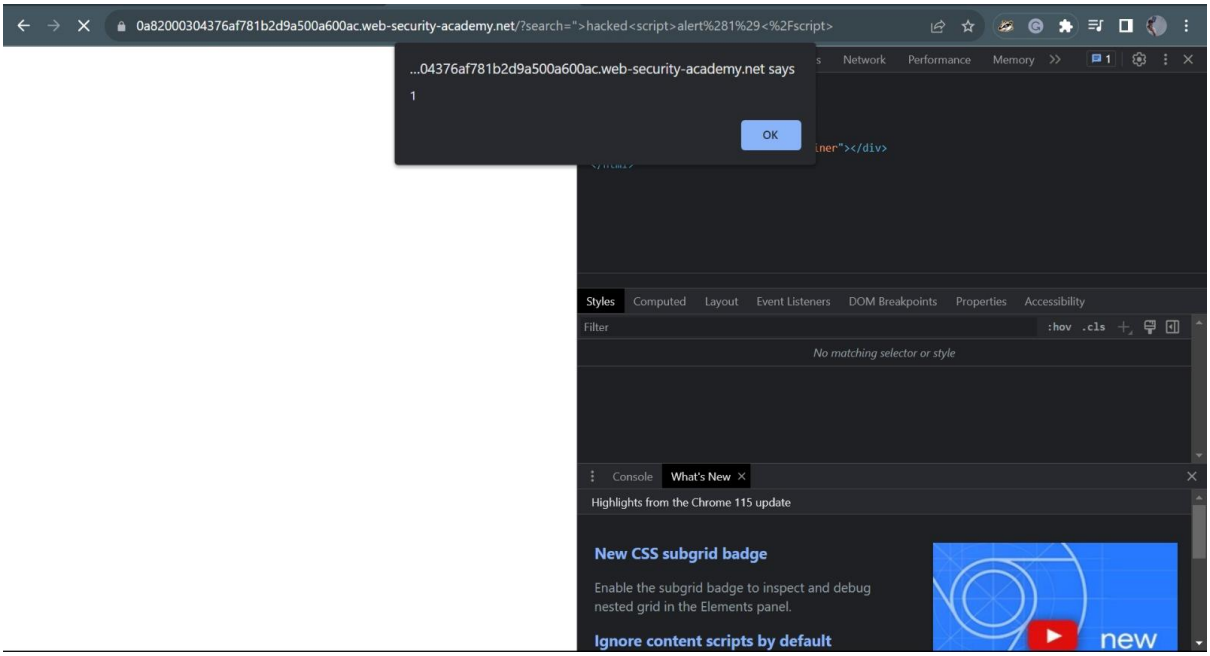
3. DOM XSS in `document. write-sink` using source `location.search`:-

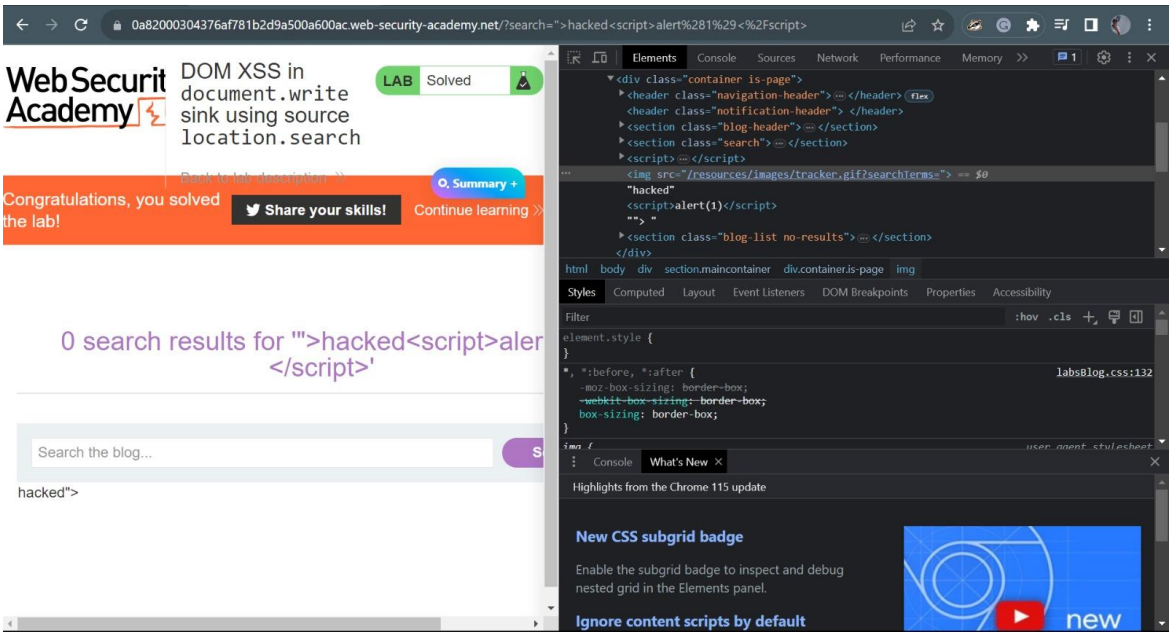


If we check the source code, we can see:
as document.write() is who writes the query as part of an img tag we can try to close the tag and insert an alert script.

Let's try then: `"><script>alert(hacked)</script>`







As we can see we've broken the tag, inserted out alert tag and solved the lab.