Sri Lanka Institute of Information Technology

B.Sc. (Hons) in Information Technology

Specializing in Cyber Security

Year 2, Semester 2

IE2062: Web Security

Week 1 Submission – PortSwigger Labs - XSS

IT23714120

A.M.M.G.K.P. Athawuda

Contents

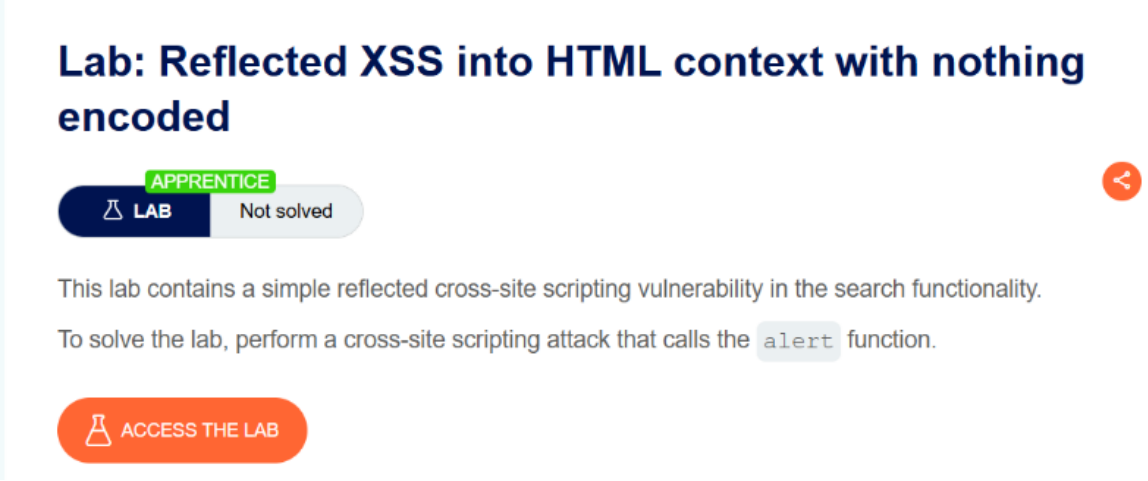
1 Introduction

XSS is a Cross-Site Scripting, which is a client-side code injection attack where malicious JavaScript is injected into a web page viewed by other users.

Main types of XSS are:

1. Reflected XSS
   Script is injected into a request and reflected immediately in the response.
2. Stored XSS
   Script is stored on the server and served to multiple users.
3. DOM- based XSS
   The injection occurs via client-side JavaScript without server-side modification.

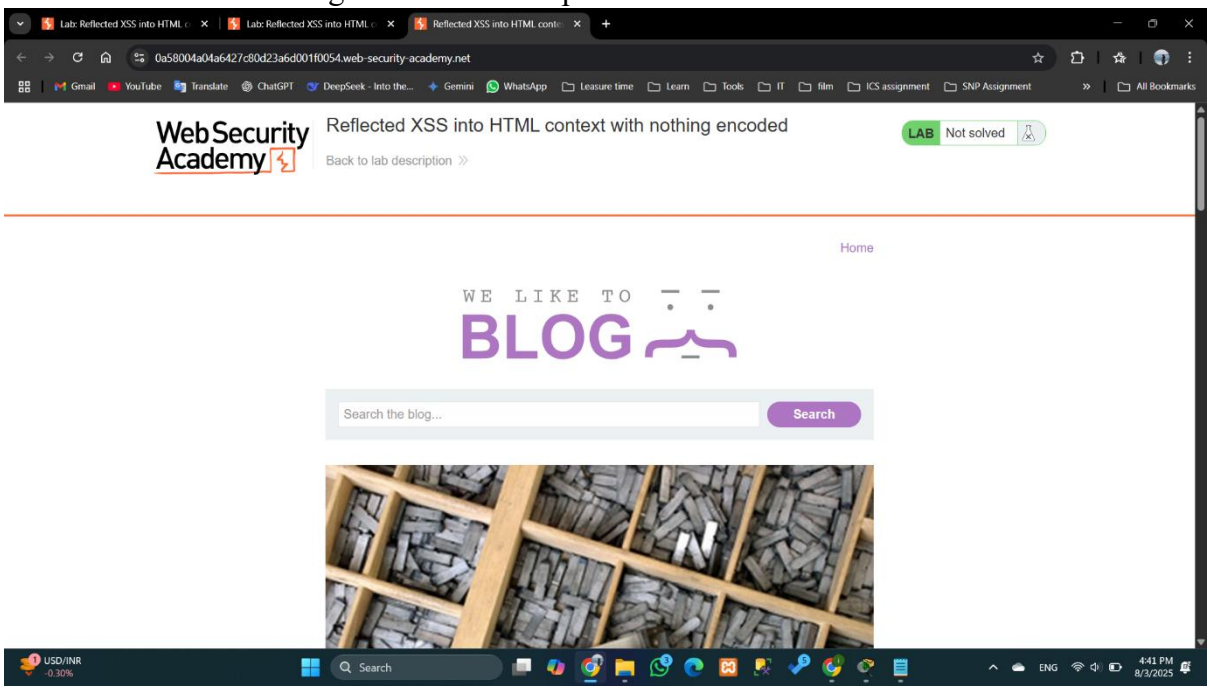
2 Lab 1: Reflected XSS into HTML context with nothing encoded



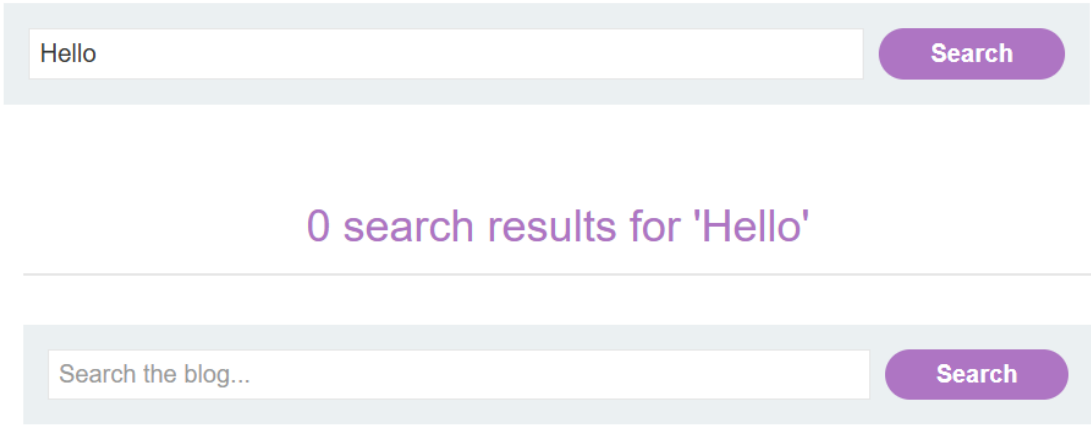
The instruction for the lab says that there is a vulnerability in search functionality. And to solve the lab it must perform a cross-site scripting attack that calls the 'alert' function.

Solution:

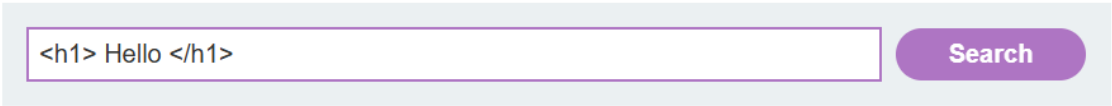
1. Click Access the lab and go to the new site provided.

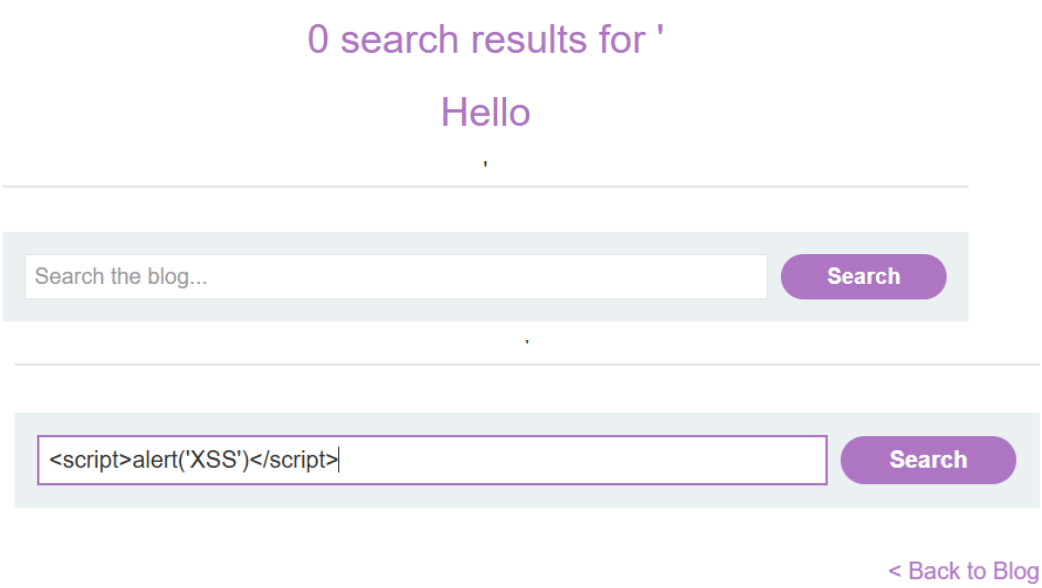


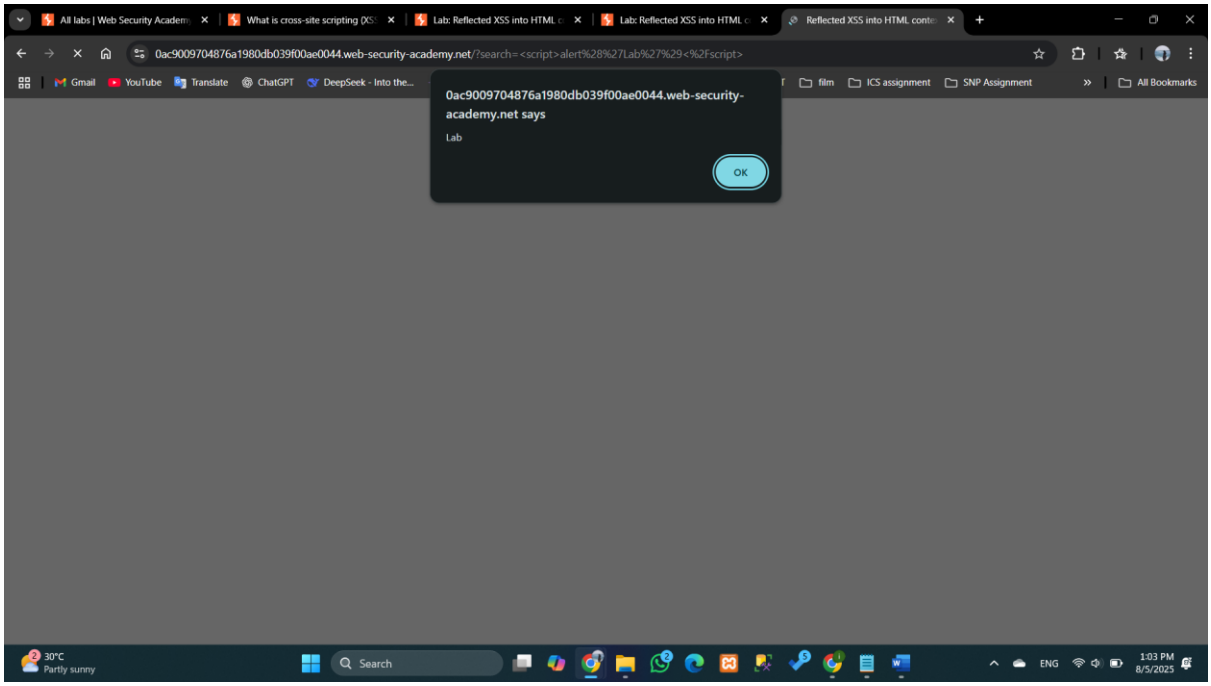
2. In the search box or in the URL type something to how that text appeared.



3. Type various kinds of inputs in the search box to see what happened. When type a html tag or js code it rendered the output.







4. When JavaScript code entered an alert box popped up. It executes Js input without any encoding.

- It demonstrates that the website is vulnerable to reflected XSS attacks as it

Lab: Reflected XSS into HTML context with nothing encoded

APPRENTICE

LAB Solved

3 Lab 2: Stored XSS into HTML context with nothing encoded

Instruction:

Lab: Stored XSS into HTML context with nothing encoded

APPRENTICE

LAB Solved

This lab contains a stored cross-site scripting vulnerability in the comment functionality.

To solve this lab, submit a comment that calls the `alert` function when the blog post is viewed.

Solution:

1. There are some blogs on the site and to go to comment section by clicking 'view post'.

No Silly Names, Please

We hear about it all the time, the unusual names people have given their children. I say unusual to be polite because, to be honest, some of them are just downright ridiculous. Have these parents no idea of the pressure...

View post

Comments

Dean N'Mean | 24 July 2025

Sorry I haven't been in touch mother, I keep getting your email wrong. Hope this reaches you.

Bart Gallery | 01 August 2025

Don't give up your day job.

Leave a comment

Comment:

Name:

Email:

Website:

Post Comment

< Back to Blog

2. Then post a comment in plain text and use HTML to see what happened.

Leave a comment

Comment:

Hello

Name:

Kavi

Email:

abcd@gmail.com

Website:

Post Comment

Thank you for your comment!

Your comment has been submitted.

< Back to blog

Kavi | 05 August 2025

Hello

Html:

Leave a comment

Comment:

<h3> Hello </h3>

Name:

Kavi

Email:

abcd@gmail.com

Website:

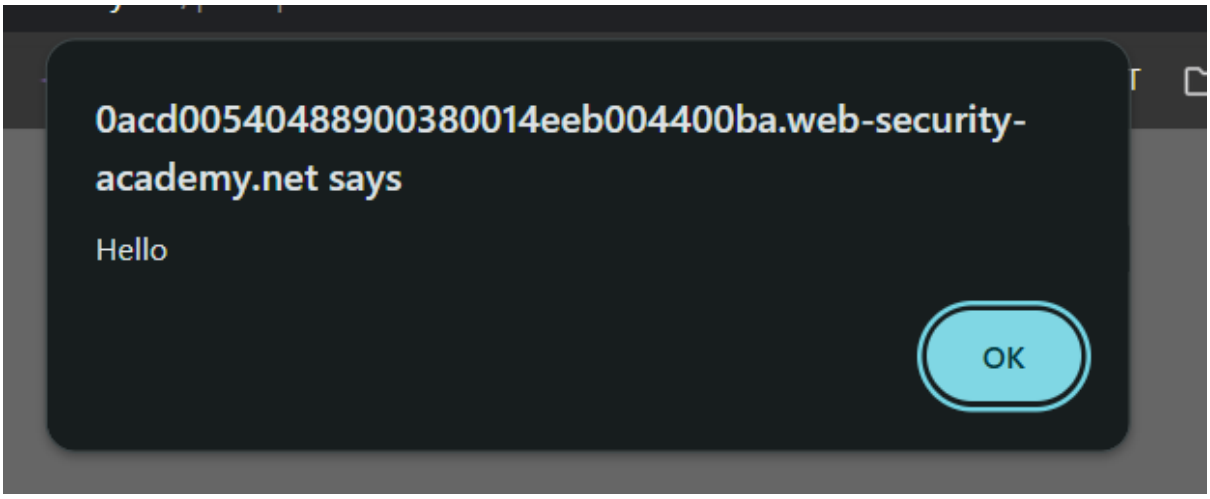
Kavi | 05 August 2025

Hello

3. Enter a JavaScript payload:

Comment:

<script> alert('Hello') </script>



4. Alert executes. It means the site is vulnerable to all users who view the comment, and the output is not encoded when displaying comments.

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

Instruction:

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

APPRENTICE

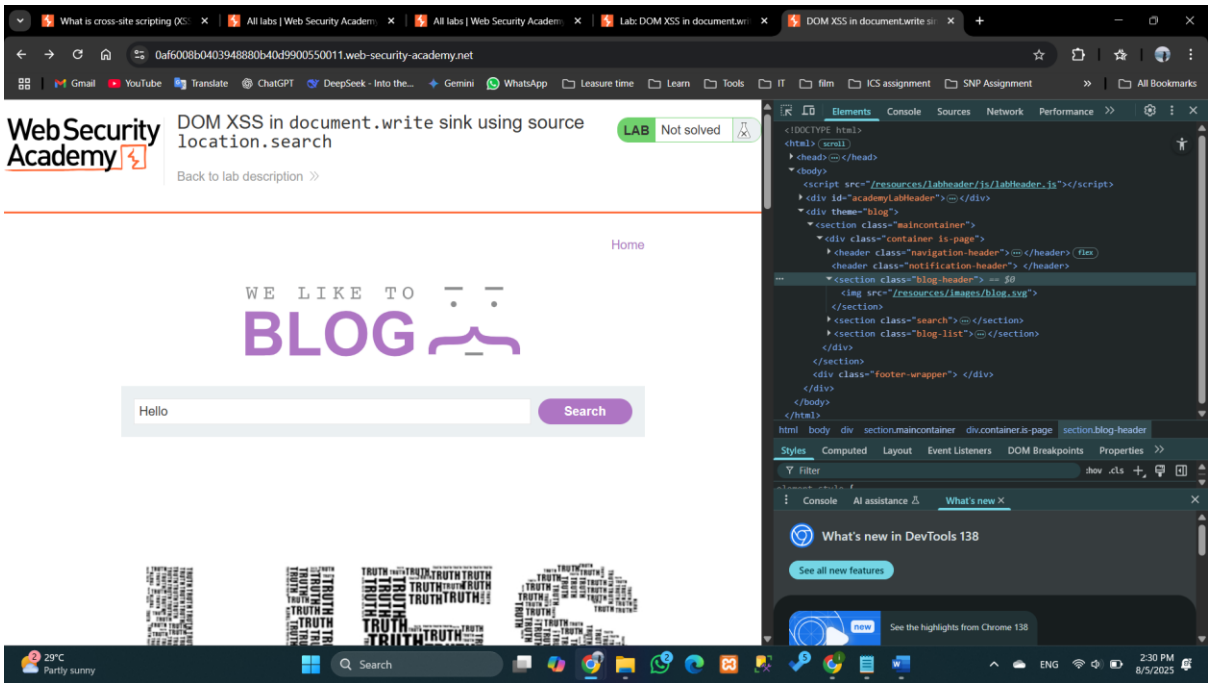
LAB Solved

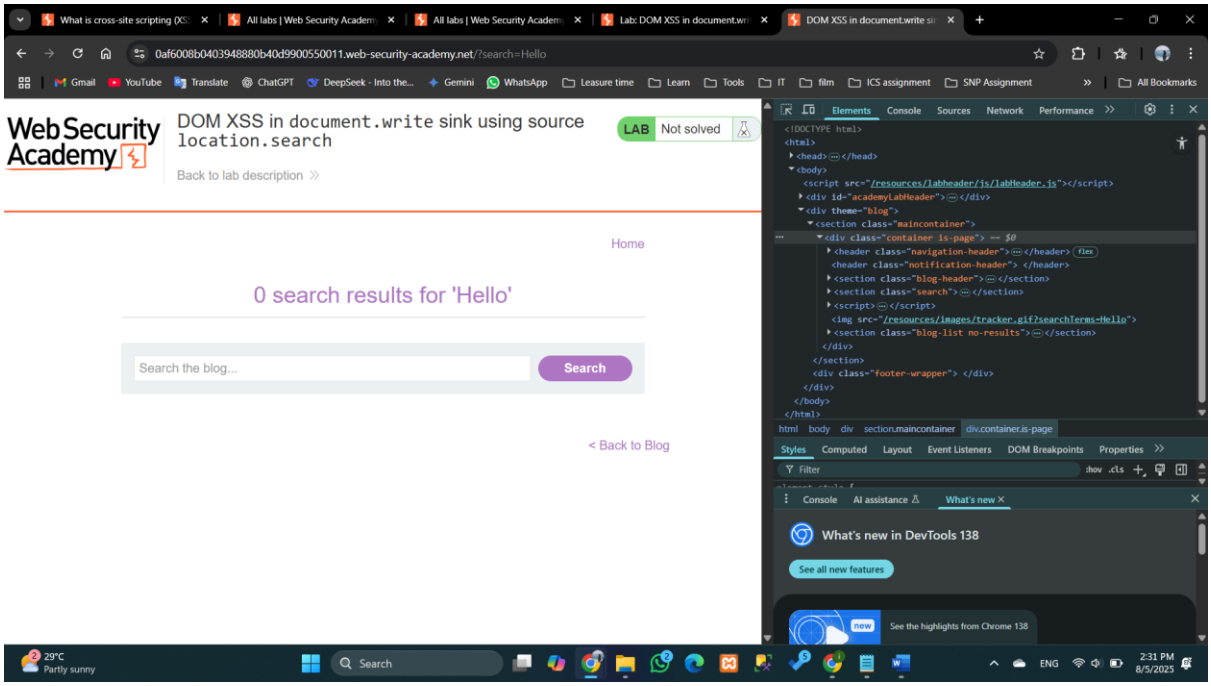
This lab contains a DOM-based cross-site scripting vulnerability in the search query tracking functionality. It uses the JavaScript `document.write` function, which writes data out to the page. The `document.write` function is called with data from `location.search`, which you can control using the website URL.

To solve this lab, perform a cross-site scripting attack that calls the `alert` function.

Solution:

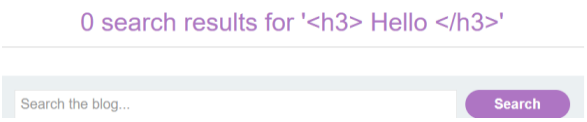
1. Access the lab and open the ‘inspect’ the page.
2. Enter a test string like ‘Hello’ and see what happens in the code.

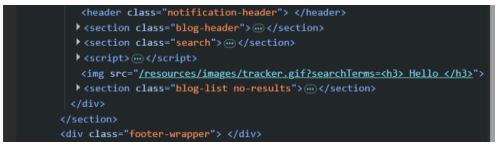




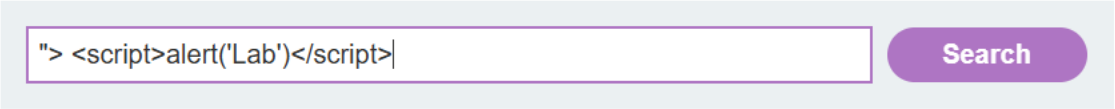
3. The string that entered appeared in ‘src’ attribute ‘<img>’ tag.

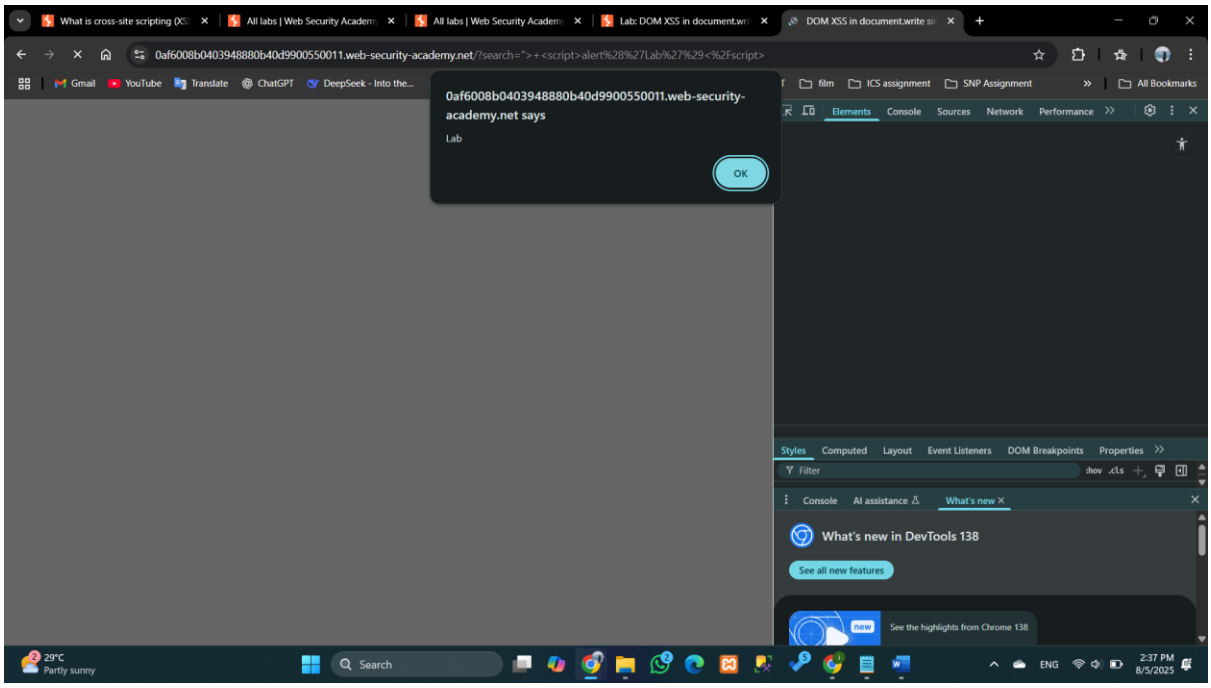
```
▸ <script> ⋯ </script>
  <img src="/resources/images/tracker.gif?searchTerms=Hello">
▸ <section class="blog-list no-results"> ⋯ </section>
```

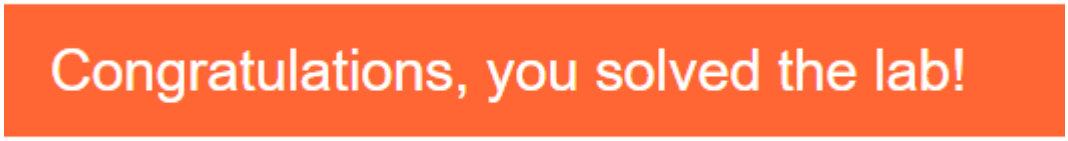




4. This indicates the application dynamically writes search terms into the DOM using *'document.write'*.
5. To break down that html attribute and enter JavaScript enter the below payload:
   (" closes the src attribute and > closes img tagged)







5 Lab 4: DOM XSS in innerHTML sink using source location.search

Instruction:

Lab: DOM XSS in `innerHTML` sink using source `location.search`

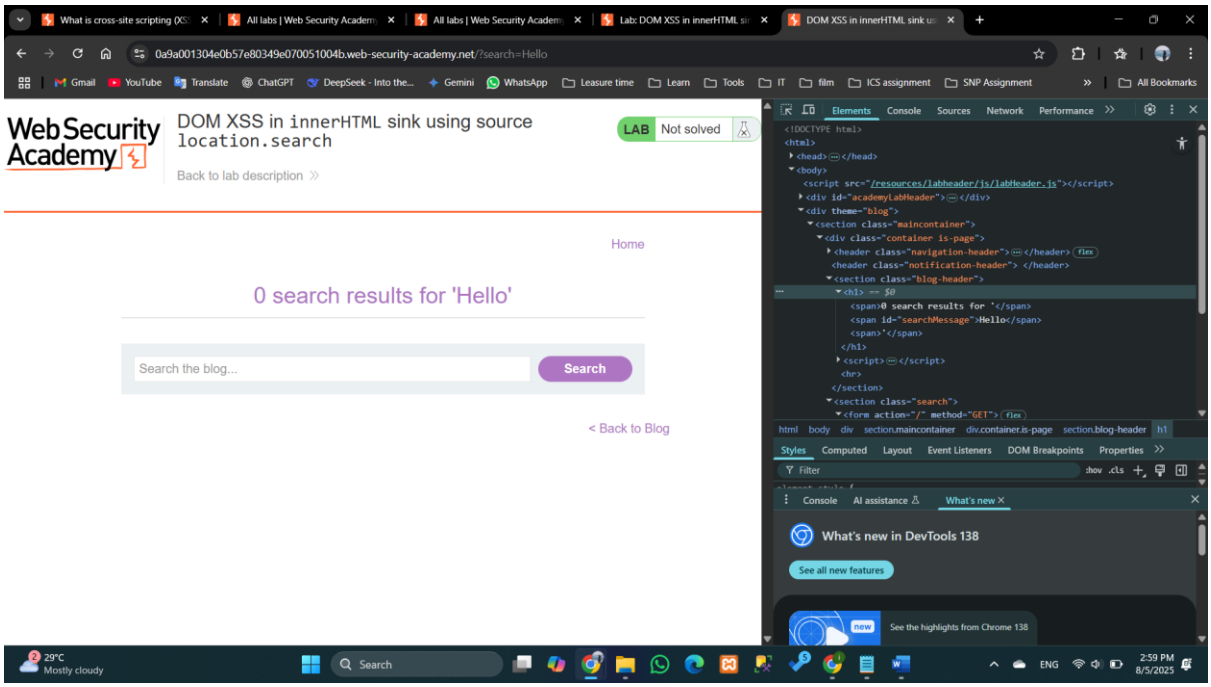
APPRENTICE

LAB Solved

This lab contains a DOM-based cross-site scripting vulnerability in the search blog functionality. It uses an `innerHTML` assignment, which changes the HTML contents of a `div` element, using data from `location.search`.

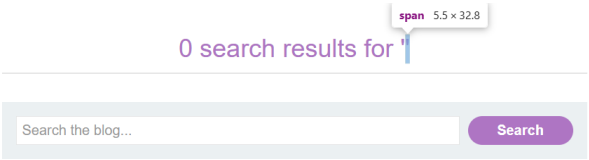
To solve this lab, perform a cross-site scripting attack that calls the `alert` function.

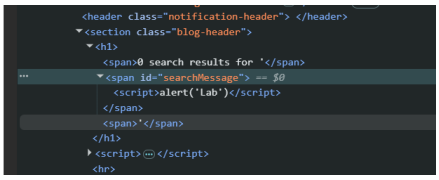
Solution:



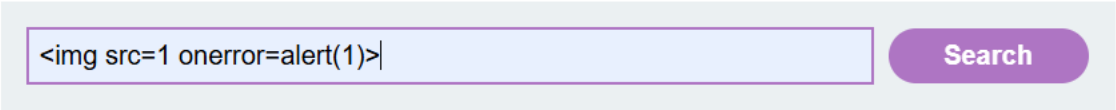
Search <h3> Hello </h3>

<script>alert('Lab') </script>

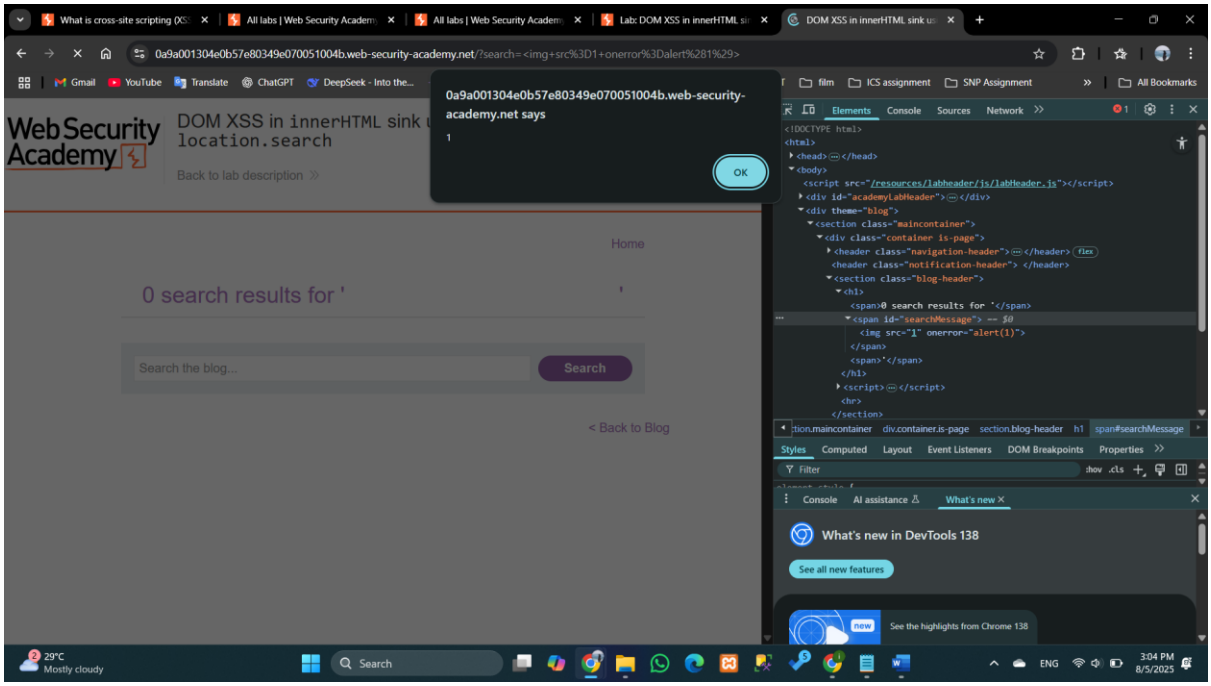




Enter the payload:



[to force an error: - src(1); invalid source , an onerror event handler containing the malicious JavaScript]



Popup is appearing. The page is vulnerable.

6 Lab 5: DOM XSS in jQuery anchor href attribute sink using location.search source

Instruction:

Lab: DOM XSS in jQuery anchor `href` attribute sink using `location.search` source

APPRENTICE

LAB | Not solved

This lab contains a DOM-based cross-site scripting vulnerability in the submit feedback page. It uses the jQuery library's `$` selector function to find an anchor element, and changes its `href` attribute using data from `location.search`.

To solve this lab, make the "back" link alert `document.cookie`.

ACCESS THE LAB

Solution:

1. Navigate to submit feedback page.
2. The URL:

   0ae4003103d023a082960bce00f7005d.web-security-academy.net/feedback?returnPath=/

   Inspect the page:

   ```
   <div class="is-linkback"> == $0
    ::before
    <a id="backLink" href="/">Back</a>
   /div>
   ```

3. Modify the URL to:

   https://0ae4003103d023a082960bce00f7005d.web-security-academy.net/feedback?returnPath=/test123

   ```
   <div class="is-linkback"> == $0
     ::before
     <a id="backLink" href="/test123">Back</a>
   </div>
   ```

4. 'Return path' parameter controls the back links destination, and it directly updates the href tag.
5. Since href accepts JavaScript; enter the payload:

   https://0ae4003103d023a082960bce00f7005d.web-security-academy.net/feedback?returnPath=javascript:alert(document.cookie)

```
::before
<a id="backLink" href="javascript:alert(document.cookie)">Back
</a> == $0
</div>
```

Congratulations, you solved the lab!

7 Lab 6: DOM XSS in jQuery selector sink using a hashchange event

Instruction:

Lab: DOM XSS in jQuery selector sink using a hashchange event

APPRENTICE

LAB Not solved

This lab contains a DOM-based cross-site scripting vulnerability on the home page. It uses jQuery's `$()` selector function to auto-scroll to a given post, whose title is passed via the `location.hash` property.

To solve the lab, deliver an exploit to the victim that calls the `print()` function in their browser.

ACCESS THE LAB

Solution:

1. Access the lab and see where the jQuery selector exists.

```
<script>
$(window).on('hashchange', function(){
var post = $('section.blog-list h2:contains(' +
decodeURIComponent(window.location.hash.slice(1)) + ')');
if (post) post.get(0).scrollIntoView();
});
</script>
```

2. go to the exploit server and paste the payload.

Craft a response

URL: https://exploit-0a0100da04f9f0d08104e3f9013e0007.exploit-server.net/exploit

HTTPS

File:

/exploit

Head:

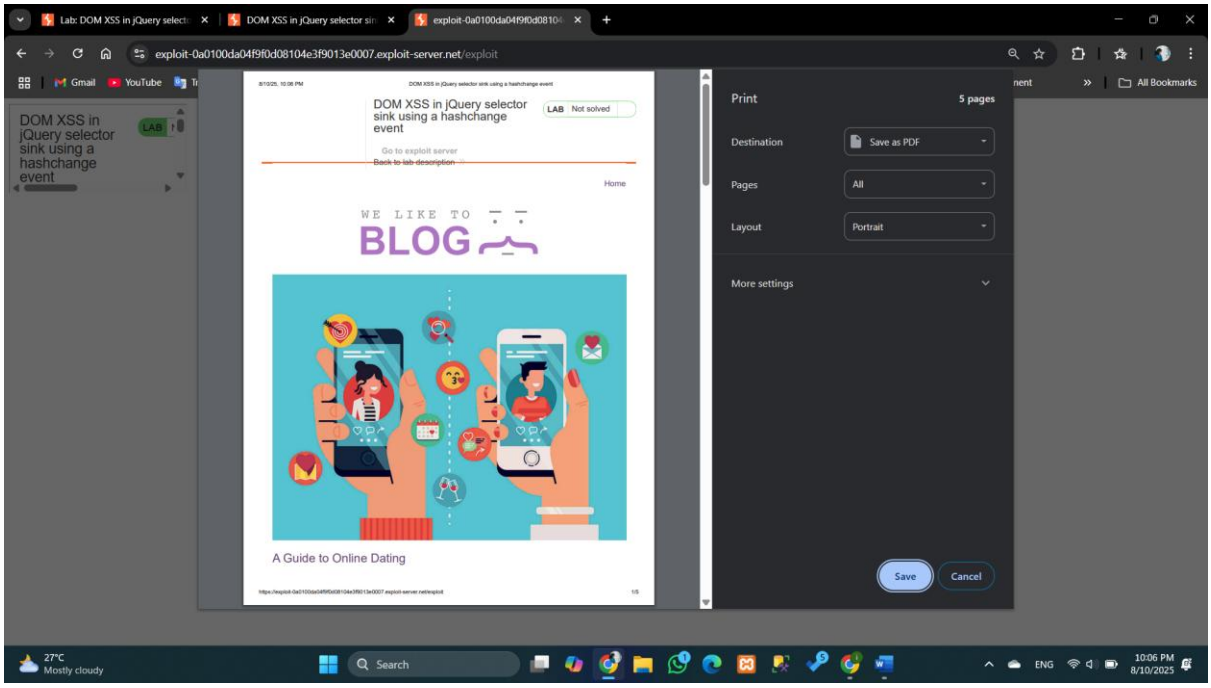
```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<iframe src="https://0ad200cb043ef03d8156e45b004e00cd.web-security-academy.net/#" onload="this.src+='<img src=x onerror=print()>'"></iframe>
```

Store | View exploit | Deliver exploit to victim | Access log

Click 'view exploit'.



And then click 'Deliver exploit to victim'. Then the lab will be solved.

Congratulations, you solved the lab!

8 Lab 7: Reflected XSS into attribute with angle brackets HTML-encoded

Instruction:

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

APPRENTICE

LAB Not solved

This lab contains a reflected cross-site scripting vulnerability in the search blog functionality where angle brackets are HTML-encoded. To solve this lab, perform a cross-site scripting attack that injects an attribute and calls the `alert` function.

Hint

Just because you're able to trigger the `alert()` yourself doesn't mean that this will work on the victim. You may need to try injecting your proof-of-concept payload with a variety of different attributes before you find one that successfully executes in the victim's browser.

Solution:

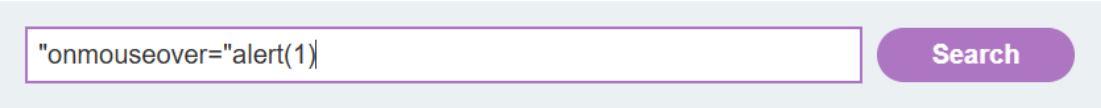
1. To see is the angle brackets are encoded in the search functionality search something in the search bar.

<Hello> Search

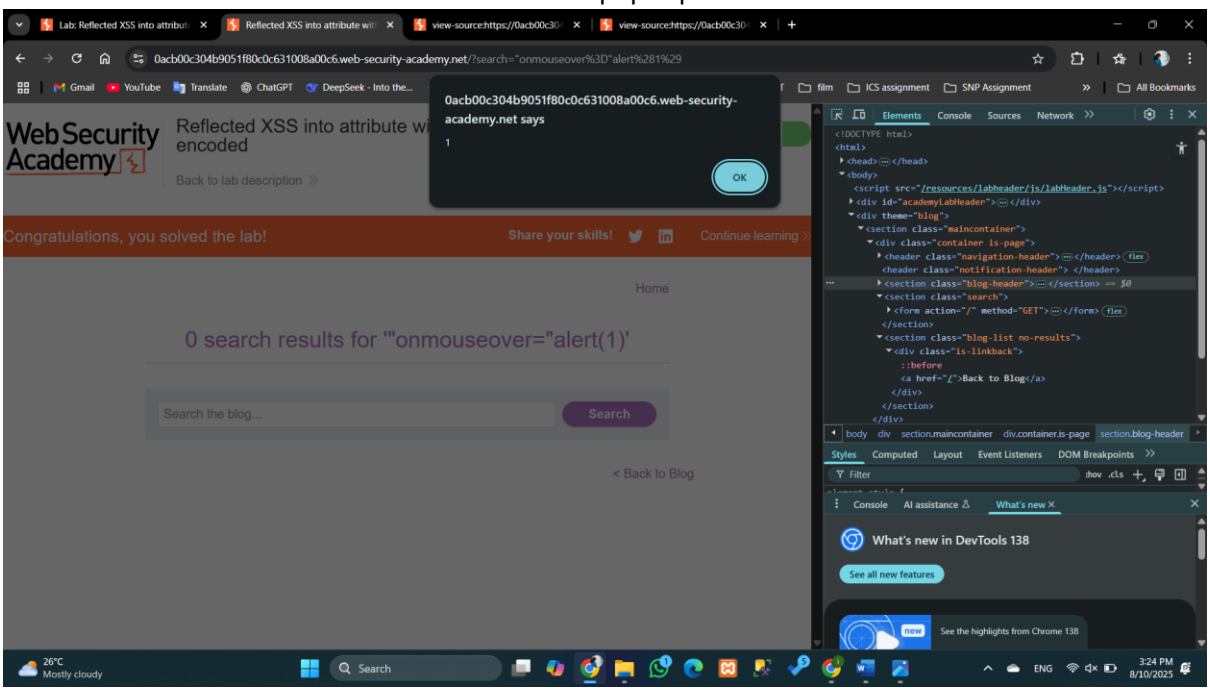
2. In the source code the brackets are encoded with < = &it and > = &gt.

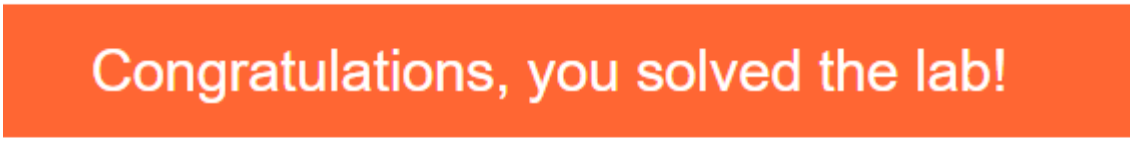
```
<section class=blog-header>
    <h1>0 search results for '&lt;Hello&gt;'</h1>
    <hr>
</section>
```

3. Insert a malicious payload:



4. When we move the mouse over the search bar. it pops up an alert.





9 Lab 8: Stored XSS into anchor *href* attribute with double quotes HTML-encoded

Instruction:

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

APPRENTICE

LAB | Not solved

This lab contains a stored cross-site scripting vulnerability in the comment functionality. To solve this lab, submit a comment that calls the `alert` function when the comment author name is clicked.

ACCESS THE LAB

Solution:

Leave a comment

Comment:

hello

Name:

Kavi

Email:

abcd@gmail.com

Website:

Post Comment

```
<p>
<img src="/resources/images/avatarDefault.svg" class="avatar">
</p>
<p>hello</p>
<p></p>
```

Kavi | 10 August 2025

Leave a comment

Comment:

hi

Name:

Kavi

Email:

abcd@gmail.com

Website:

javascript:alert('hi')

Post Comment

Congratulations, you solved the lab!

Thank you for your comment!

Your comment has been submitted.

10 Lab 9: Reflected XSS into a JavaScript string with angle brackets HTML encoded

Instruction:

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

APPRENTICE

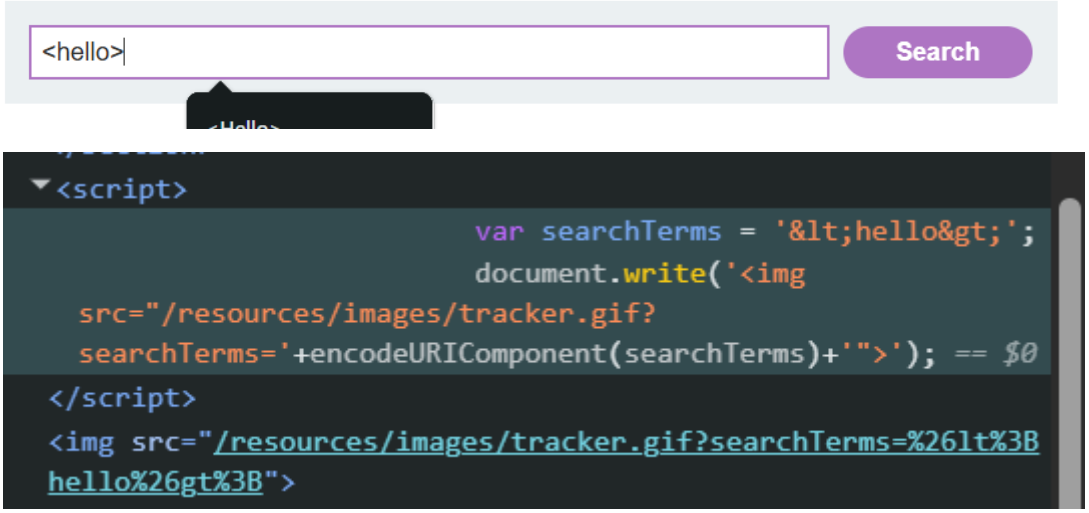
LAB | Not solved

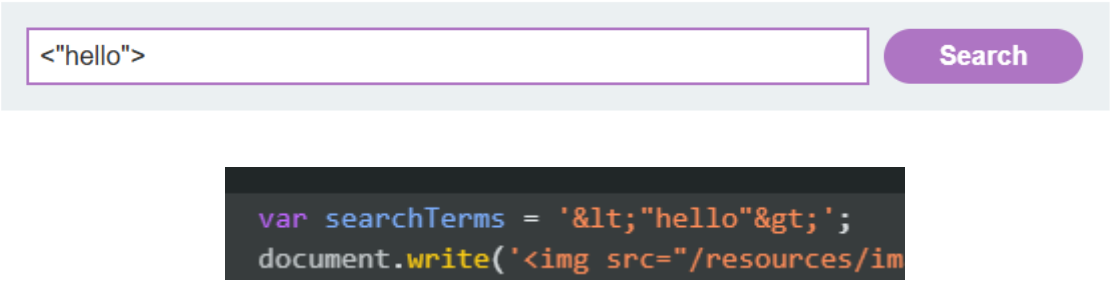
This lab contains a reflected cross-site scripting vulnerability in the search query tracking functionality where angle brackets are encoded. The reflection occurs inside a JavaScript string. To solve this lab, perform a cross-site scripting attack that breaks out of the JavaScript string and calls the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

ACCESS THE LAB

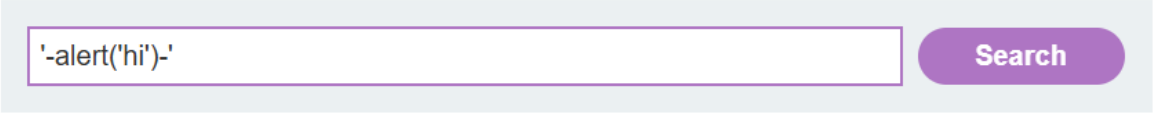
Solution:

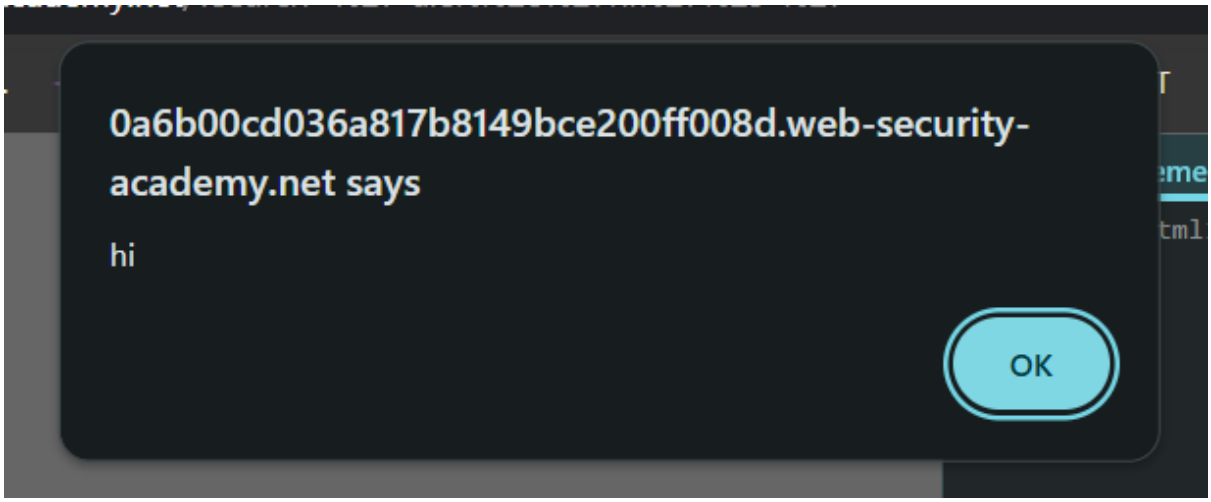
In this lab angle brackets (>, <) are html encoded as &gt and &lt. but the other symbols (/, “) are not encoded.





So we can use a single command to inject malicious code.





Pop up appeared.

Congratulations, you solved the lab!