Task name: Cross Site Scripting (XSS)

Name: Aathish M

XSS, or Cross-Site Scripting

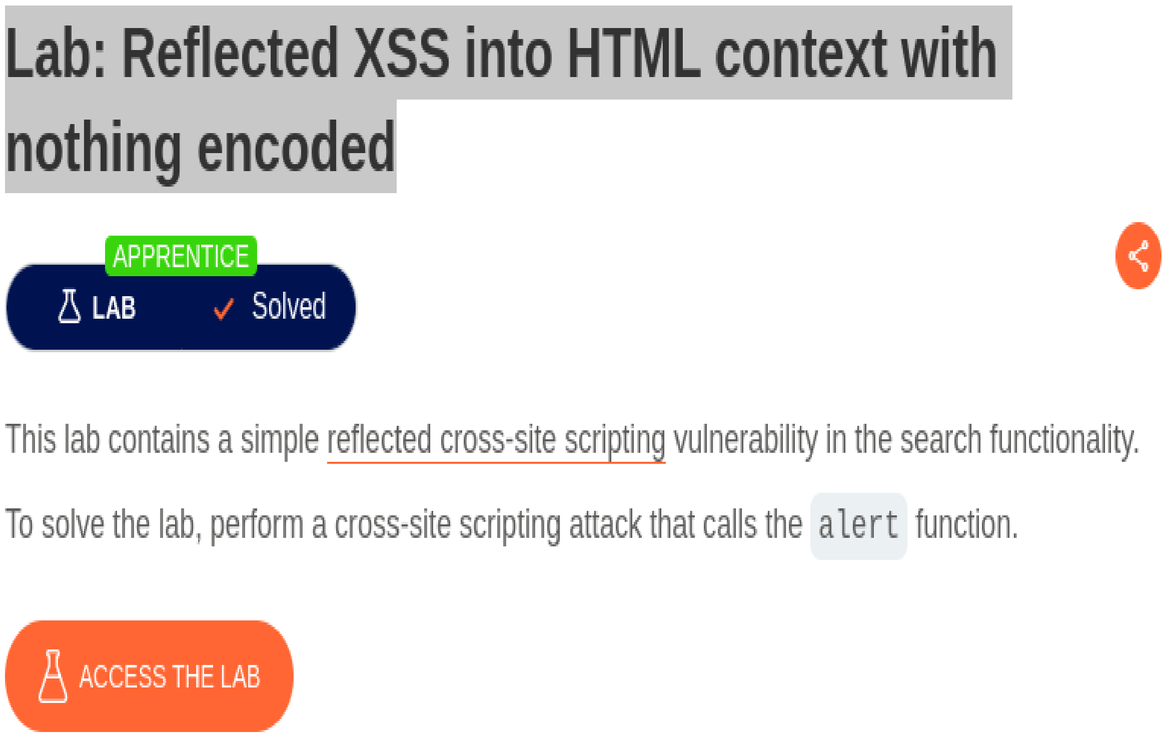
A security vulnerability in web applications. It occurs when an attacker injects malicious code (usually JavaScript) into a website, which then runs in other users' browsers. This can allow the attacker to steal data, impersonate users, or manipulate the website's appearance and functionality.

Lab 1 : Reflected XSS into HTML context with nothing encoded

Severity: Critical
Steps To Reproduce:

The lab contains Reflected XSS

This is a type of XSS attack where malicious code is included in a link or form submission. When a user clicks the link or submits the form, the malicious code is immediately reflected back to the user's browser and executed. This often happens when user input is not properly sanitized before being displayed on a webpage.

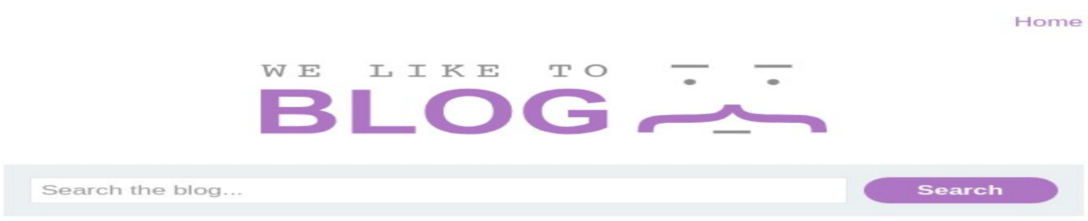


Lab: Reflected XSS into HTML context with nothing encoded

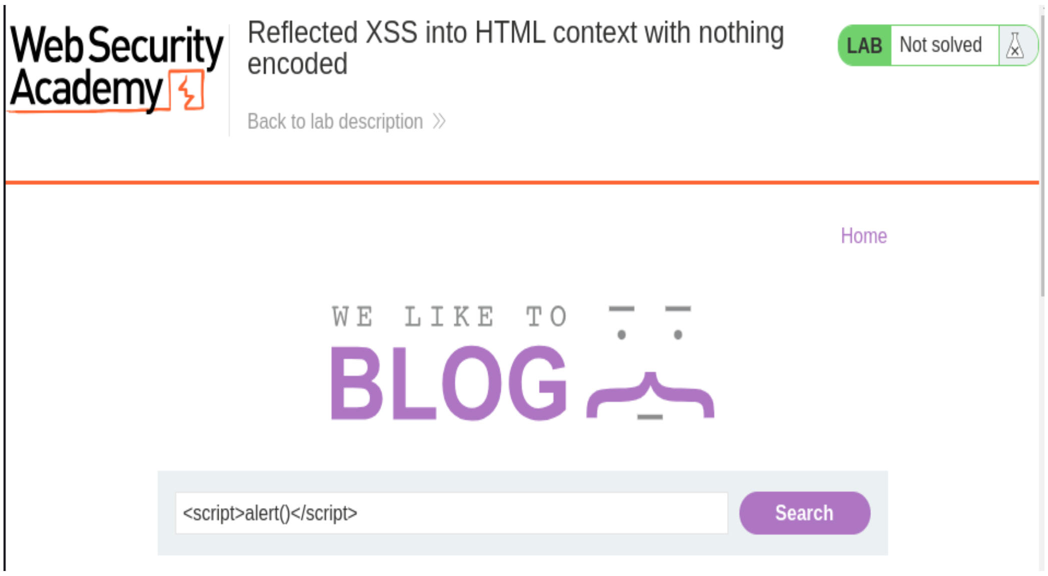
This lab contains a simple reflected cross-site scripting vulnerability in the search functionality.

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

- open the portswigger site and navigate to this page and access the lab

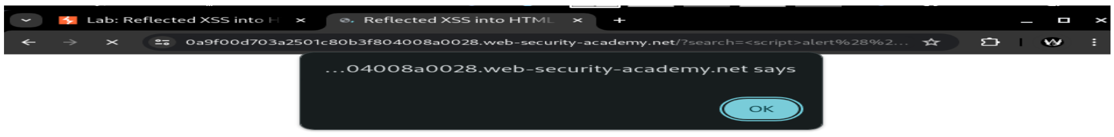


- Identify the field or section where user can give input like comment or search bars



<script>alert()</script>

- Enter the payload "<script>alert()</script>" into the search field and Submit the payload by clicking the search button to test for XSS



0a9f00d703a2501c80b3f804008a0028.web-security-academy.net/?search=<script>alert%28%2...

...04008a0028.web-security-academy.net says

- After submitting the payload a alert popup shows up and we can confirm the payloads is injected successful.

Mitigation for xss:

To fix this issue the Input given by user should be sanitized (filtered) so not does not allow any user to input malicious scripts or payloads

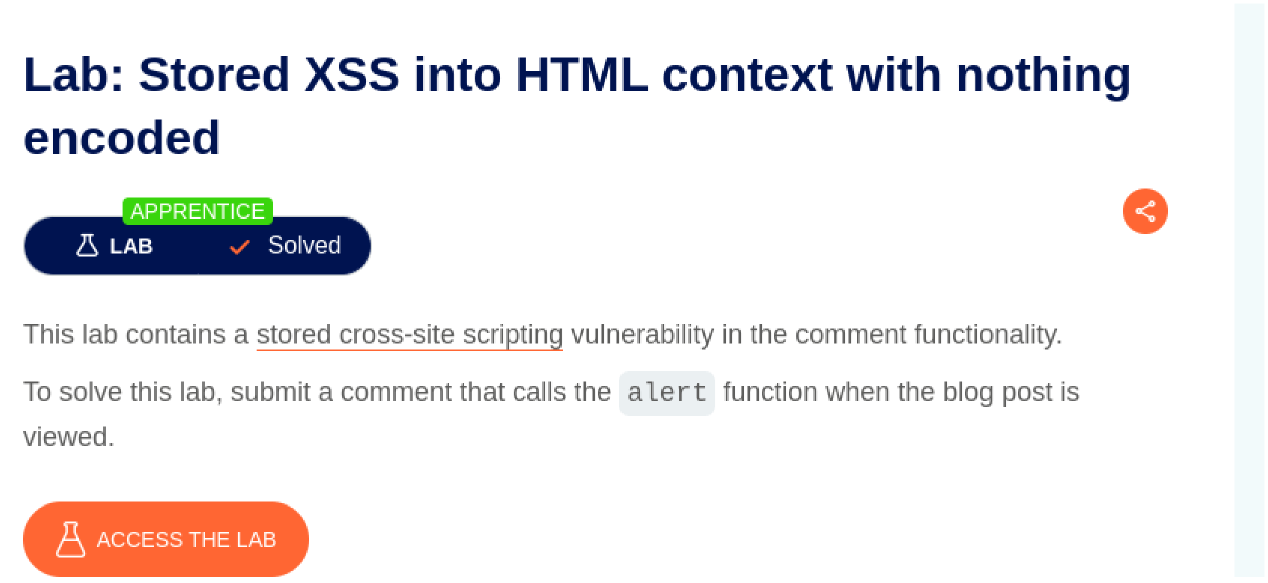
Stored XSS

A type of cross-site scripting attack where the malicious script is permanently stored on the target server. Attacker injects malicious code into a website's database .Code is retrieved and displayed to other users . Users' browsers execute the malicious script .Attacker can steal data or perform actions as the user .

Lab 2: Stored XSS into HTML context with nothing encoded

Severity: Critical
Steps To Reproduce:

The lab contains Stored XSS , which is more dangerous than reflected XSS because it affects all visitors to the compromised page, not just those who click a specific link.


Lab: Stored XSS into HTML context with nothing encoded

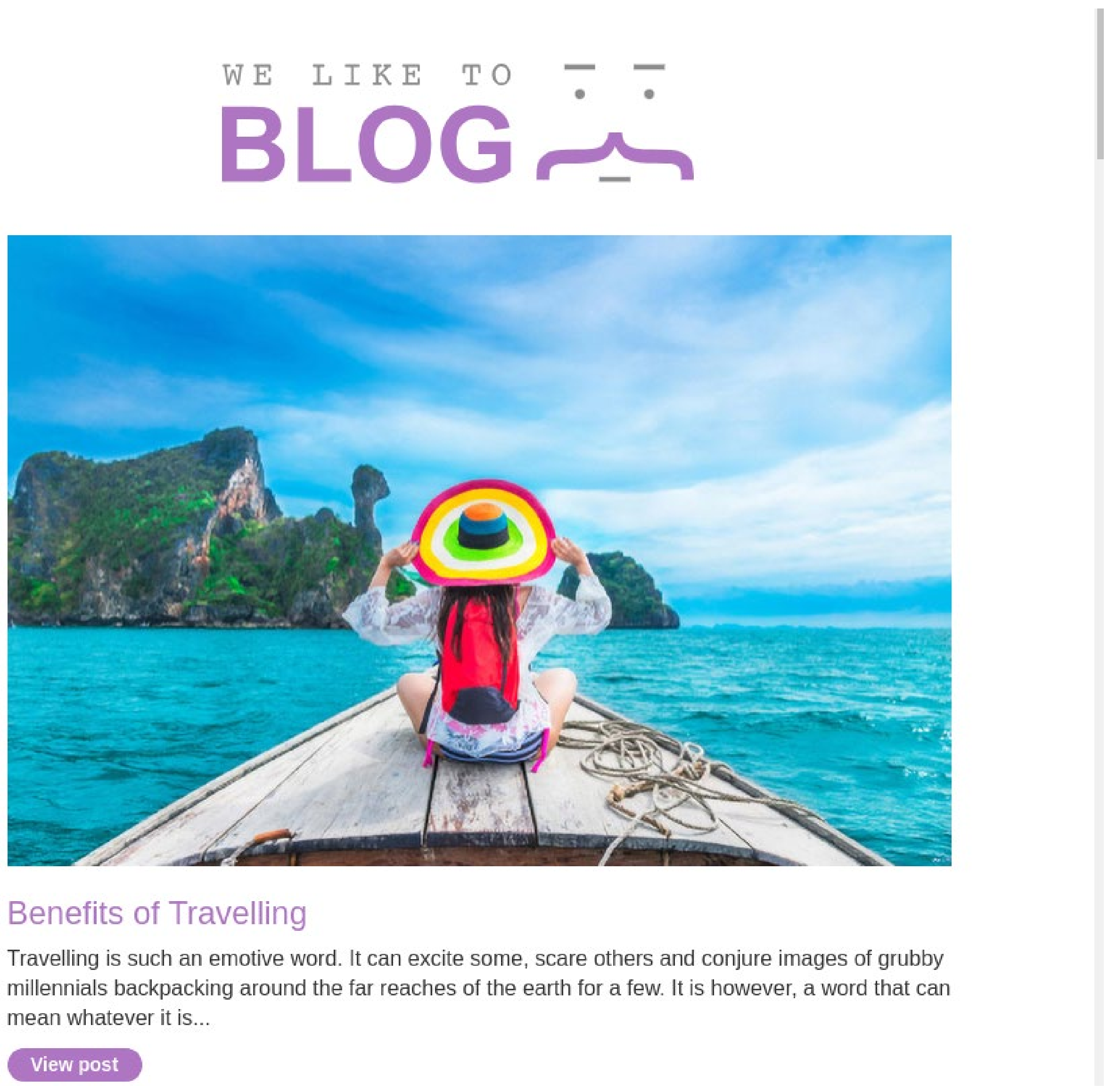
APPRENTICE
LAB Solved

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.

ACCESS THE LAB

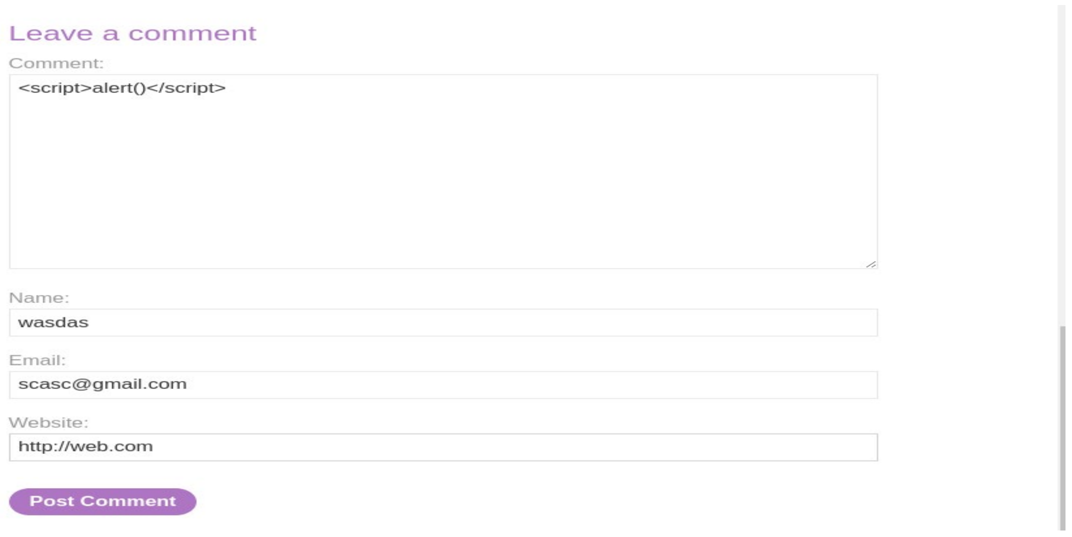
- Go to the port swigger site and access the lab using the given link


WE LIKE TO BLOG

Benefits of Travelling

Travelling is such an emotive word. It can excite some, scare others and conjure images of grubby millennials backpacking around the far reaches of the earth for a few. It is however, a word that can mean whatever it is...

- The page has many post so i got into one and found comment section in it


Leave a comment

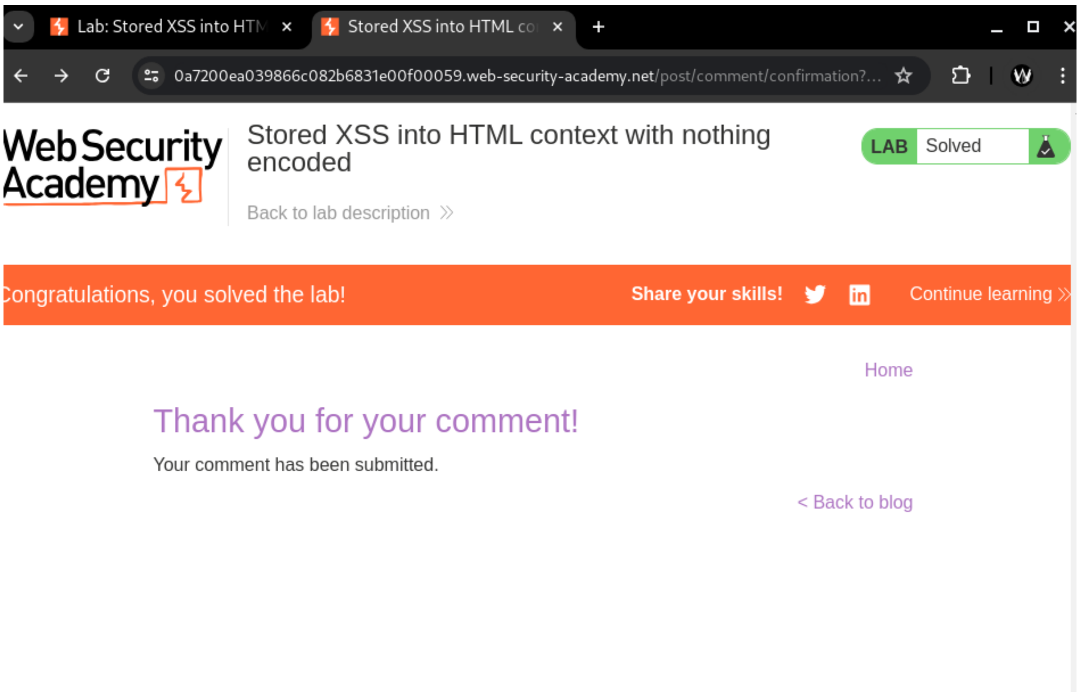
Comment:
```
<script>alert()</script>
```

Name: wasdas

Email: scasc@gmail.com

Website: http://web.com

- I used the payload <script>alert()</script> and posted a comment to check if it is vulnerable


Lab: Stored XSS into HTM
Stored XSS into HTML co
0a7200ea039866c082b6831e00f00059.web-security-academy.net/post/comment/confirmation?...
Stored XSS into HTML context with nothing encoded
LAB Solved
Back to lab description
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home
Thank you for your comment!
Your comment has been submitted.
< Back to blog

- After submitting the comment I clicked back to blog button and received the alert popup this proved that there is a stored xss vulnerability and the input from user is not sanitized/filtered

  Mitigation:
  Sanitize input, encode output, use CSP headers, and implement proper security controls to prevent malicious script storage and execution.

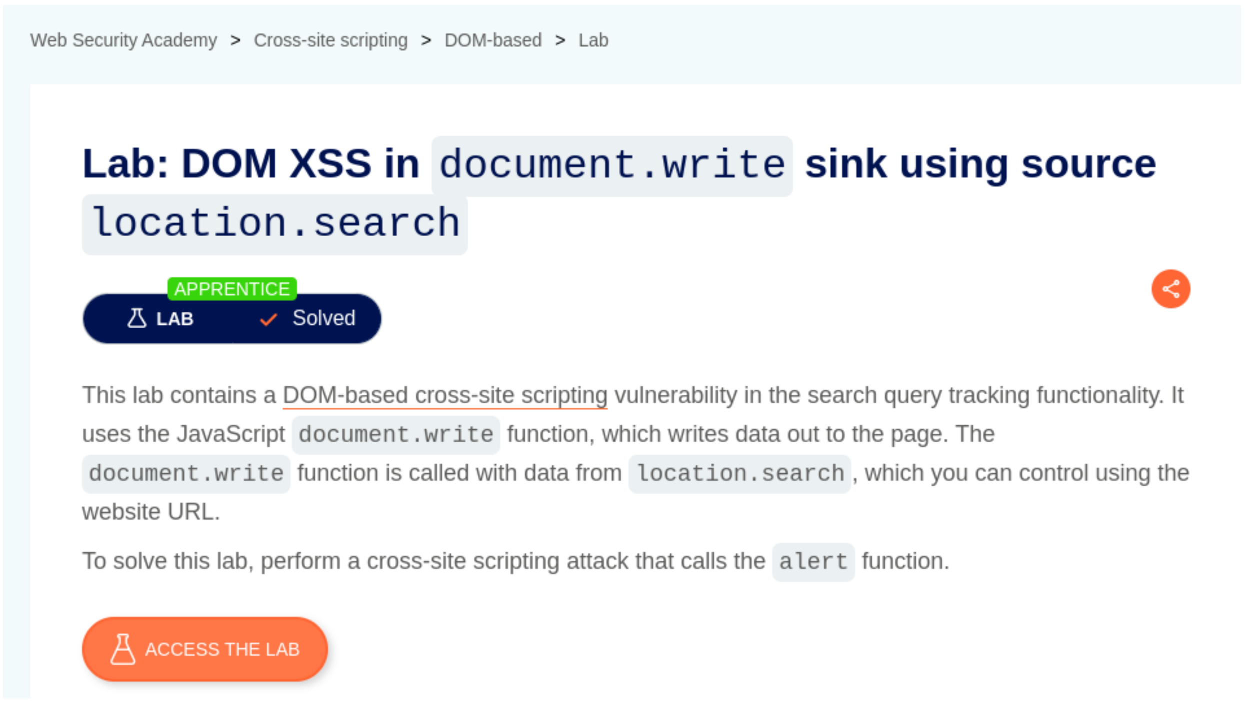
DOM XSS

DOM-based XSS occurs when client-side scripts manipulate the Document Object Model (DOM) using untrusted data, typically from user input or URL parameters. The malicious script executes in the user's browser, modifying page content or behavior without server involvement. This type of XSS is harder to detect as the vulnerability exists in client-side code rather than server responses.

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

Severity: High
Steps To Reproduce:

This lab contains DOM-based XSS, which occurs when malicious script modifies the Document Object Model (DOM) in the victim's browser, without server interaction

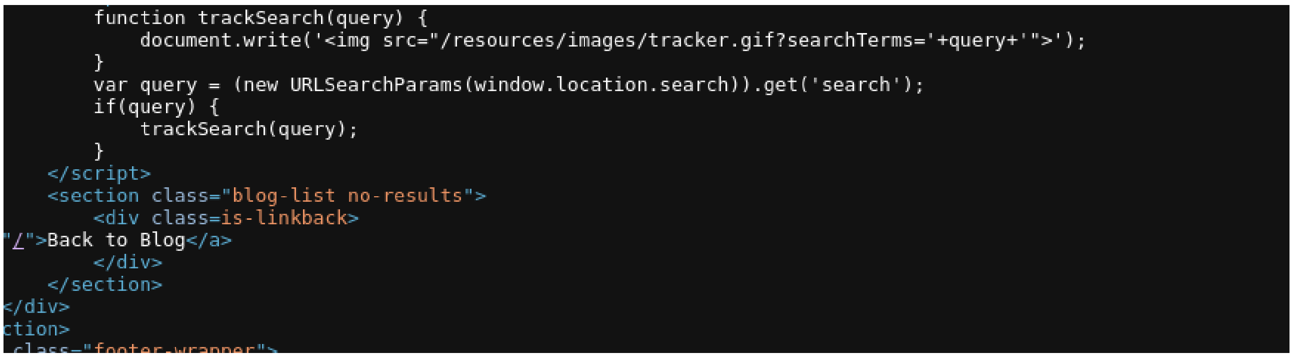

Web Security Academy > Cross-site scripting > DOM-based > Lab

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.

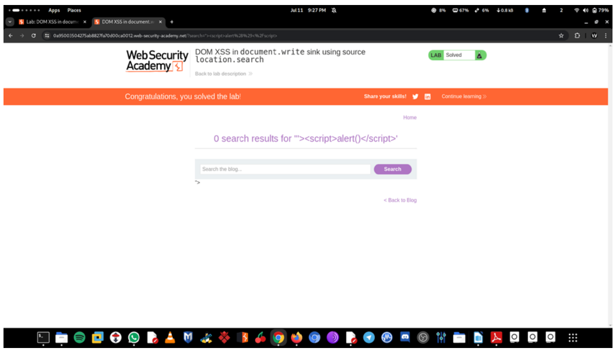
To solve this lab, perform a cross-site scripting attack that calls the `alert` function.

- To navigate inside portswigger and access the lab using the given link



```
        function trackSearch(query) {
            document.write('<img src="/resources/images/tracker.gif?searchTerms='+query+'">');
        }
        var query = (new URLSearchParams(window.location.search)).get('search');
        if(query) {
            trackSearch(query);
        }
    </script>
    <section class="blog-list no-results">
        <div class=is-linkback>
"/">Back to Blog</a>
        </div>
    </section>
</div>
ction>
 class="footer-wrapper">
```

- By viewing the page found that there is a search filed for user input after viewing the source found that I can inject payload into the query field



```
0a95003504275ab8827fa70d00ca0012.web-security-academy.net/?search="><script>alert%28%29<%2Fscript>
```

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

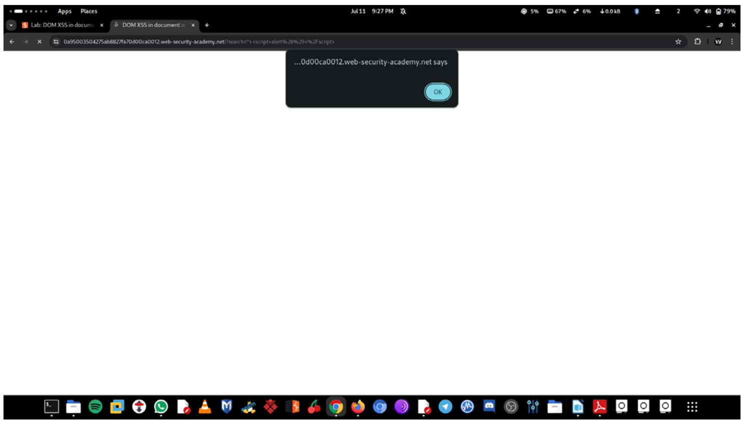
LAB Solved

Congratulations, you solved the lab!

0 search results for '"><script>alert()</script>'

">

- Used okay”> and tried to close the existing tag and continued with inserting the payload "><script>alert()</script>



```
0a95003504275ab8827fa70d00ca0012.web-security-academy.net/?search="><script>alert%28%29<%2Fscript>
```

...0d00ca0012.web-security-academy.net says

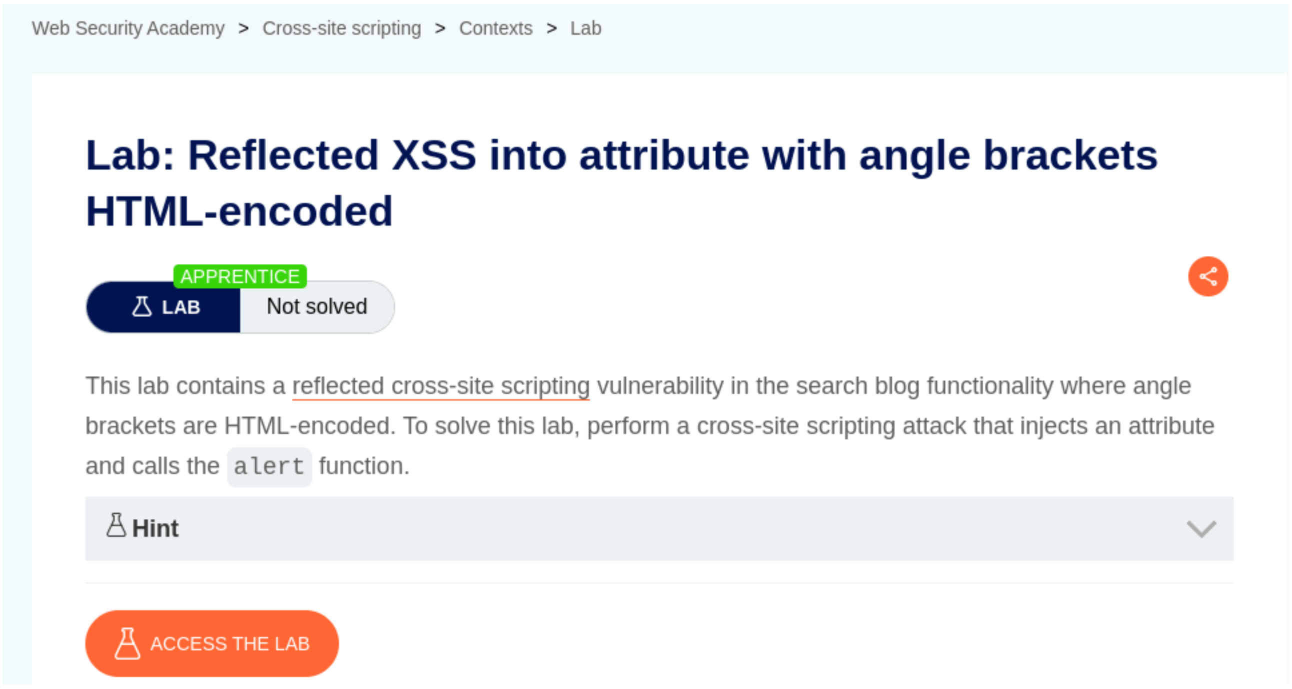
- A alert poped up confirming the Vulnerability

Mitigation:
Sanitize and validate all inputs, especially those affecting the DOM, to prevent injection of untrusted data.

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

Severity: High

Steps To Reproduce:



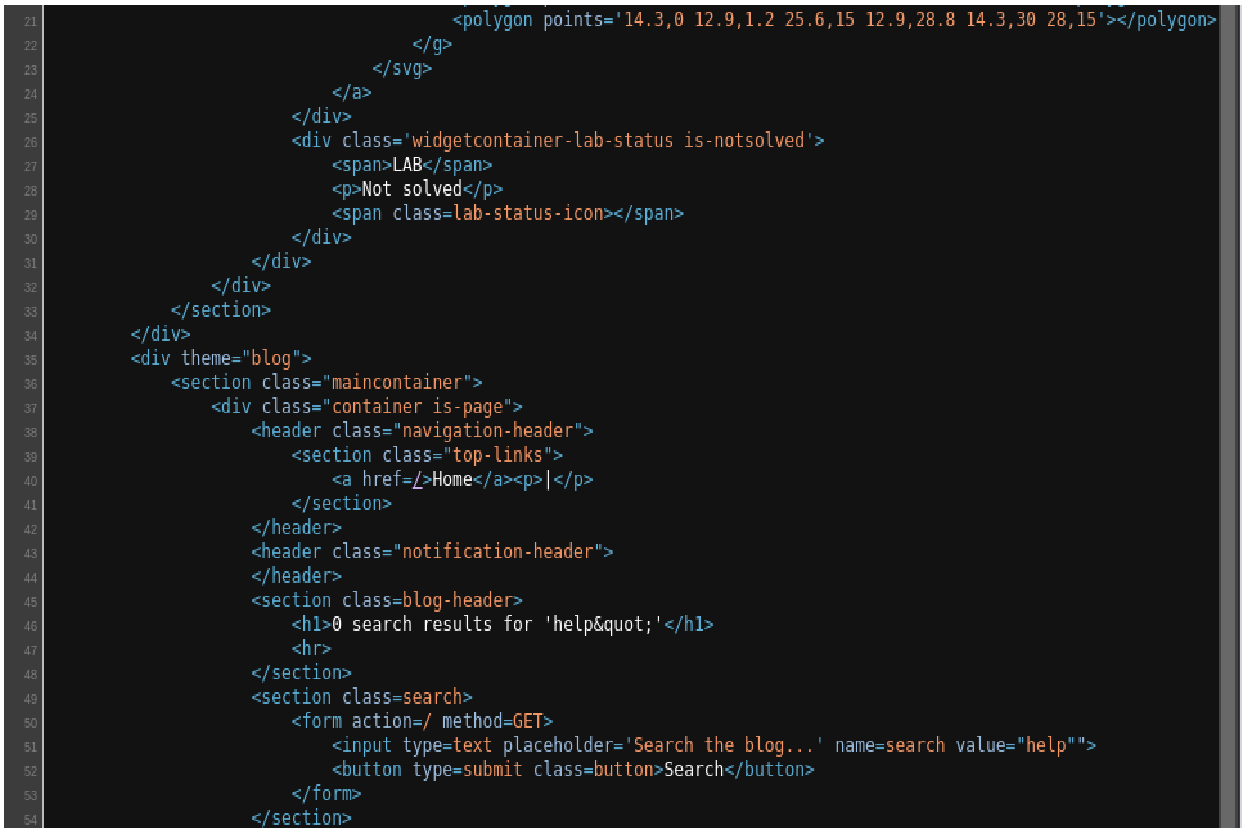
Web Security Academy > Cross-site scripting > Contexts > Lab

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

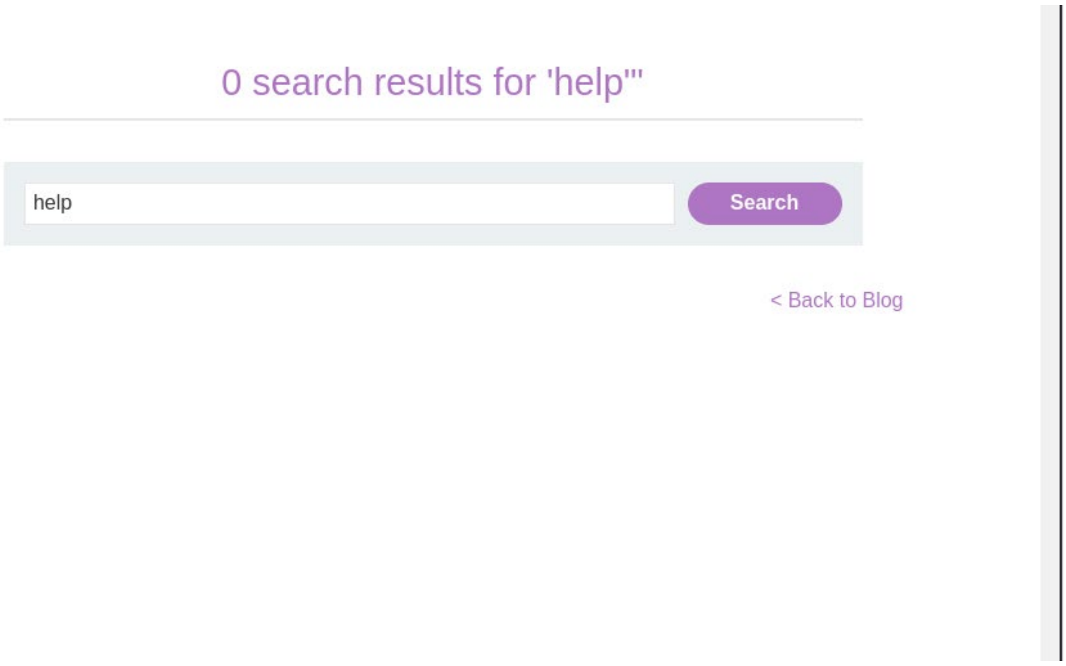
APPRENTICE LAB Not solved

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.

- To navigate to page and click the access lab button to use the lab and found that the angular brackets are encoded

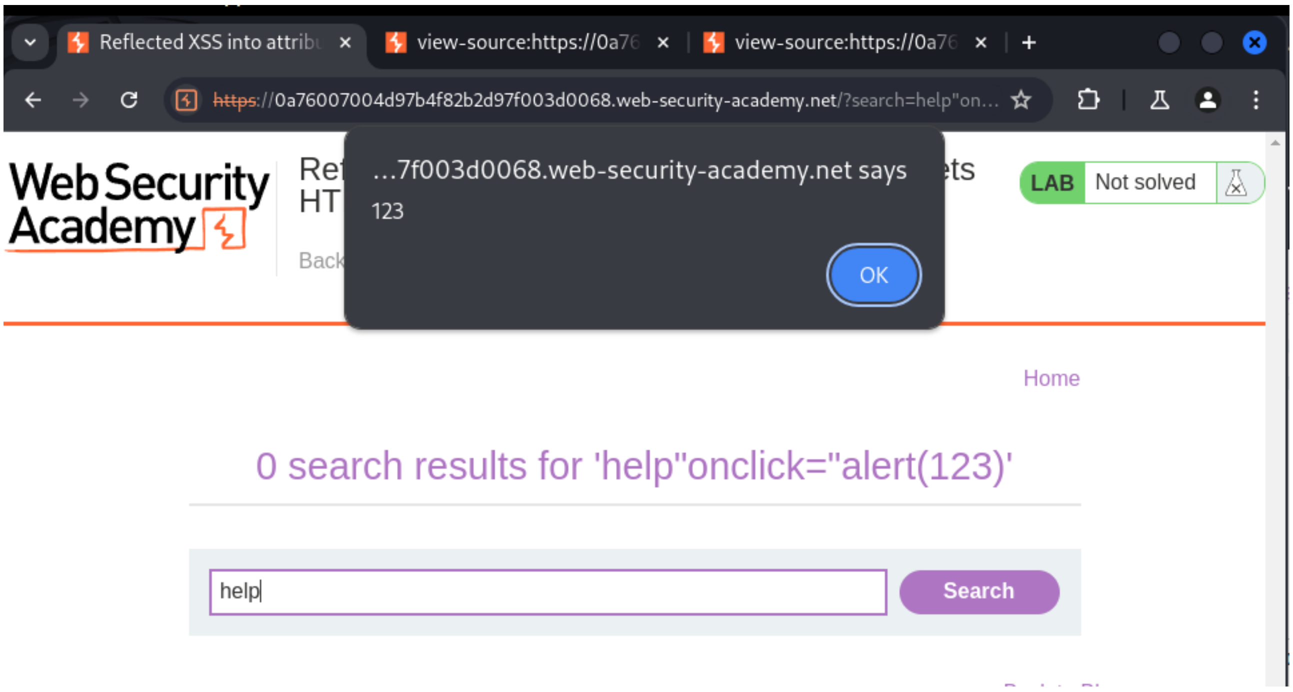


```
                                        <polygon points='14.3,0 12.9,1.2 25.6,15 12.9,28.8 14.3,30 28,15'></polygon>
                                    </g>
                                </svg>
                            </a>
                        </div>
                        <div class='widgetcontainer-lab-status is-notsolved'>
                            <span>LAB</span>
                            <p>Not solved</p>
                            <span class=lab-status-icon></span>
                        </div>
                    </div>
                </div>
            </section>
        </div>
        <div theme="blog">
            <section class="maincontainer">
                <div class="container is-page">
                    <header class="navigation-header">
                        <section class="top-links">
                            <a href=/>Home</a><p>|</p>
                        </section>
                    </header>
                    <header class="notification-header">
                    </header>
                    <section class=blog-header>
                        <h1>0 search results for 'help&quot;'</h1>
                        <hr>
                    </section>
                    <section class=search>
                        <form action=/ method=GET>
                            <input type=text placeholder='Search the blog...' name=search value="help"">
                            <button type=submit class=button>Search</button>
                        </form>
                    </section>
```



0 search results for 'help"'

- Found that double quotes is not encoded since it was not validated properly, so decided to used the payload inside the double quotes


Reflected XSS into attrib
view-source:https://0a76
https://0a76007004d97b4f82b2d97f003d0068.web-security-academy.net/?search=help"on...
WebSecurity Academy
LAB Not solved
Home
Search

...7f003d0068.web-security-academy.net says

123

OK

0 search results for 'help"onclick="alert(123)'

help

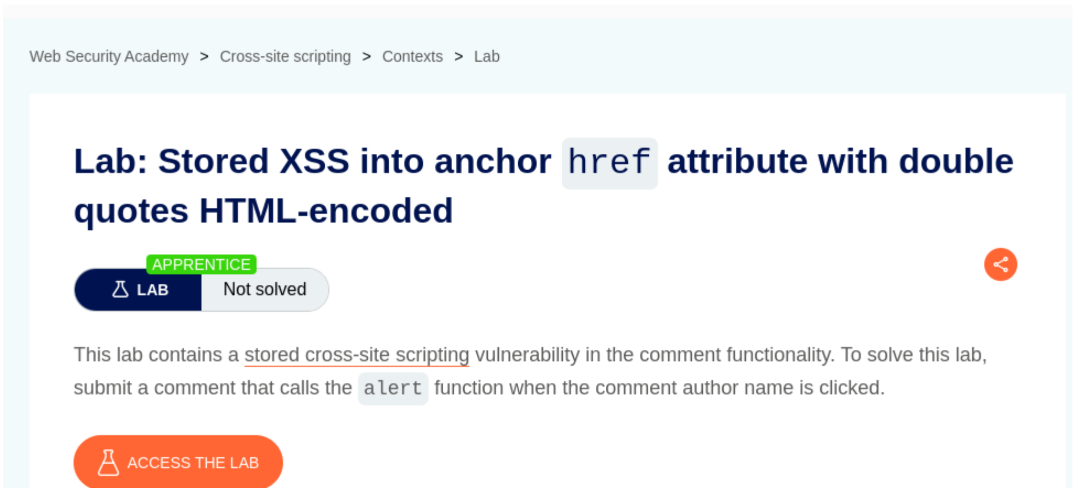
- used the payload help"onclick="alert(123) and got the popup when I clicked on the page, therefore confirming the vulnerability

  Mitigation:
  Sanitize and validate all inputs,encode user inputs and validate them on both client and server sides.

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

Severity: High

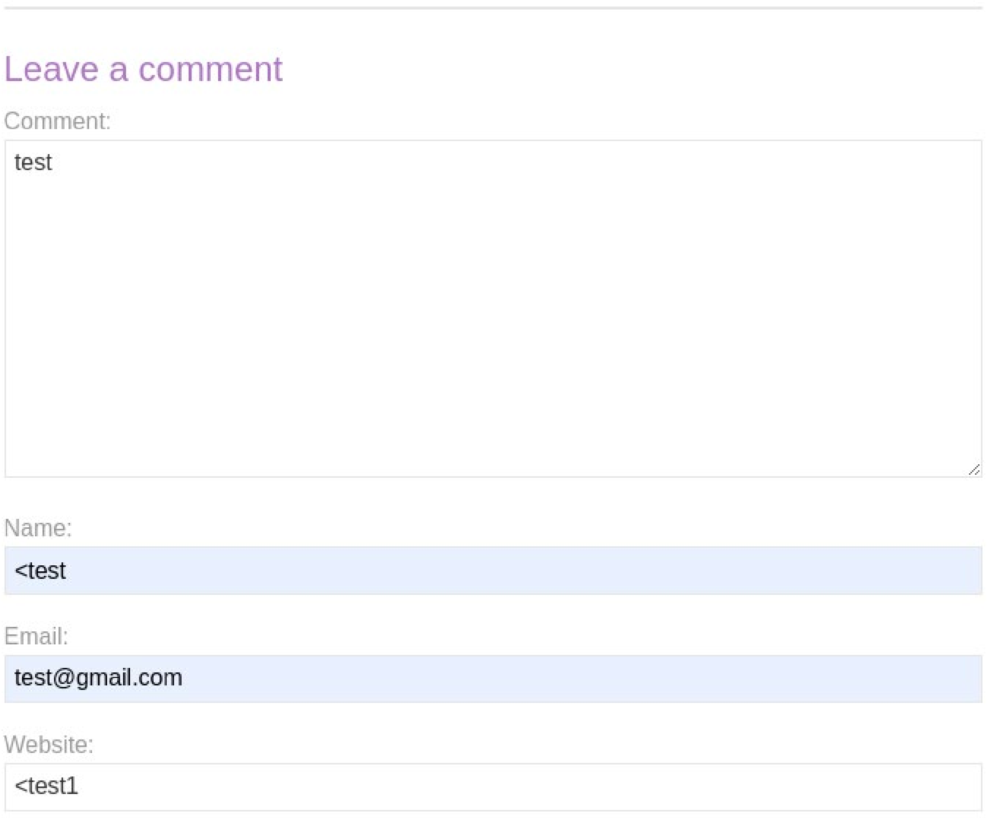
Steps To Reproduce:


Web Security Academy > Cross-site scripting > Contexts > Lab
APPRENTICE
LAB Not solved
ACCESS THE LAB

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.

- Navigate in the portswigger site and and access the lab using the given button



Leave a comment

Comment:

test

Name:

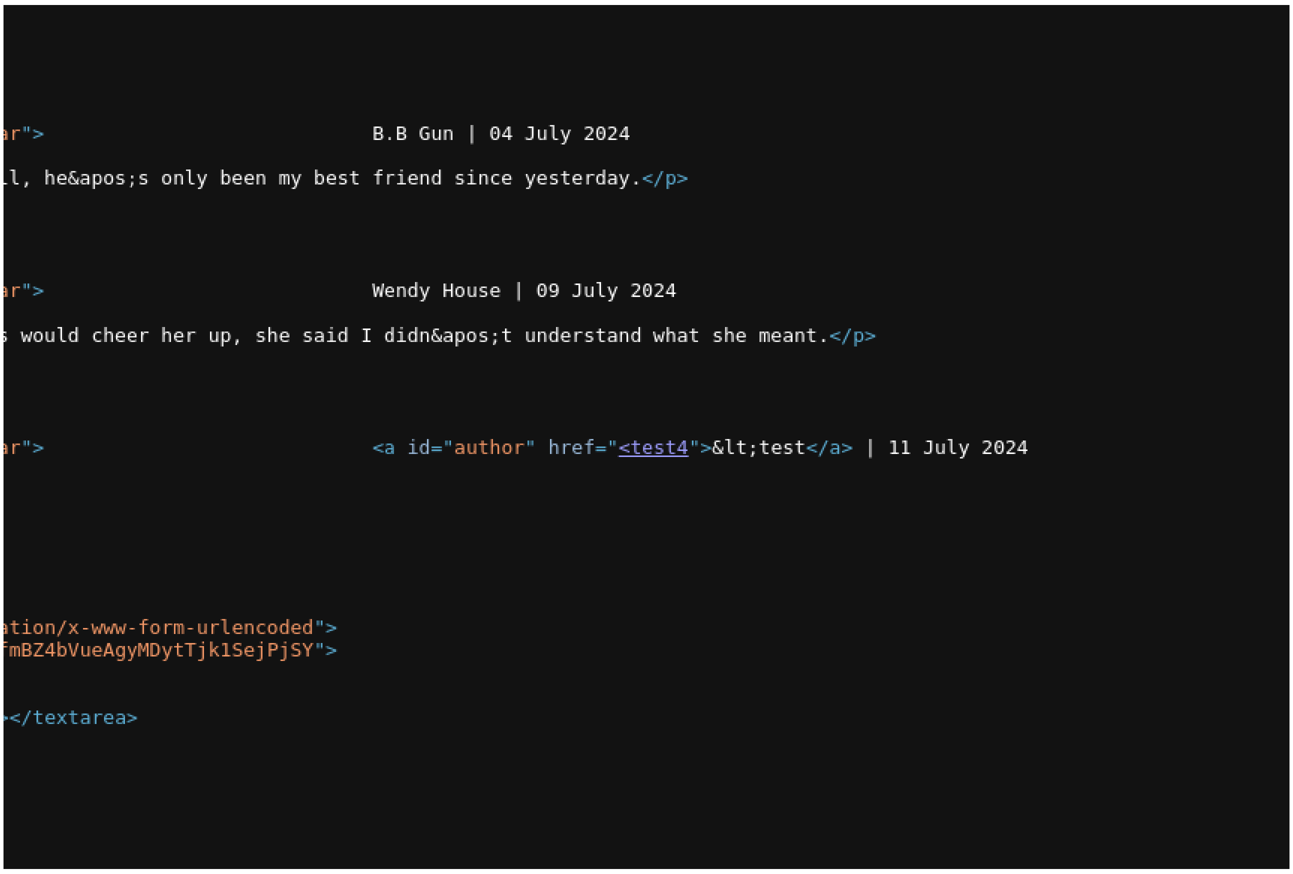
<test

Email:

test@gmail.com

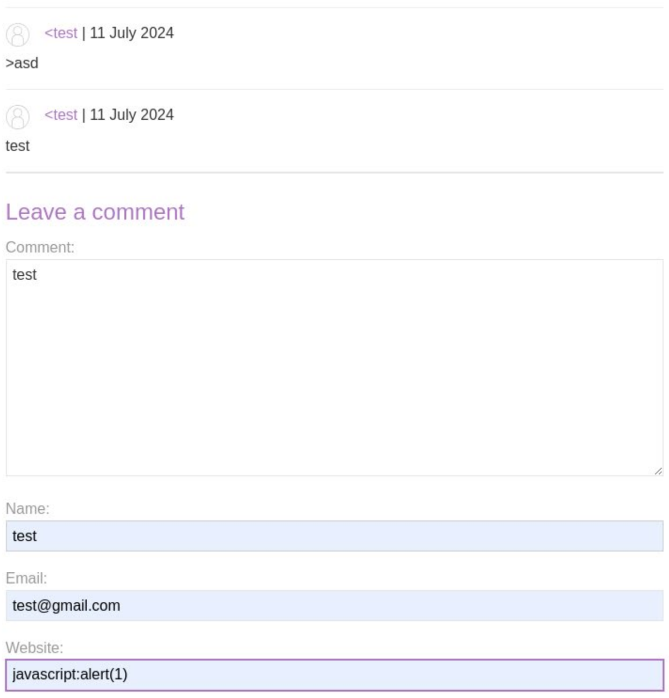
Website:

<test1



```
ar">                           B.B Gun | 04 July 2024

ll, he&apos;s only been my best friend since yesterday.</p>



ar">                           Wendy House | 09 July 2024

s would cheer her up, she said I didn&apos;t understand what she meant.</p>



ar">                           <a id="author" href="<test4">&lt;test</a> | 11 July 2024





ation/x-www-form-urlencoded">
fmBZ4bVueAgyMDytTjk1SejPjSY">


-</textarea>
```

- After analyzing with different input identified that the initial reflection is visible to be inside the anchor **href** attribute from viewing the source page



<test | 11 July 2024
>asd

<test | 11 July 2024
test

Leave a comment

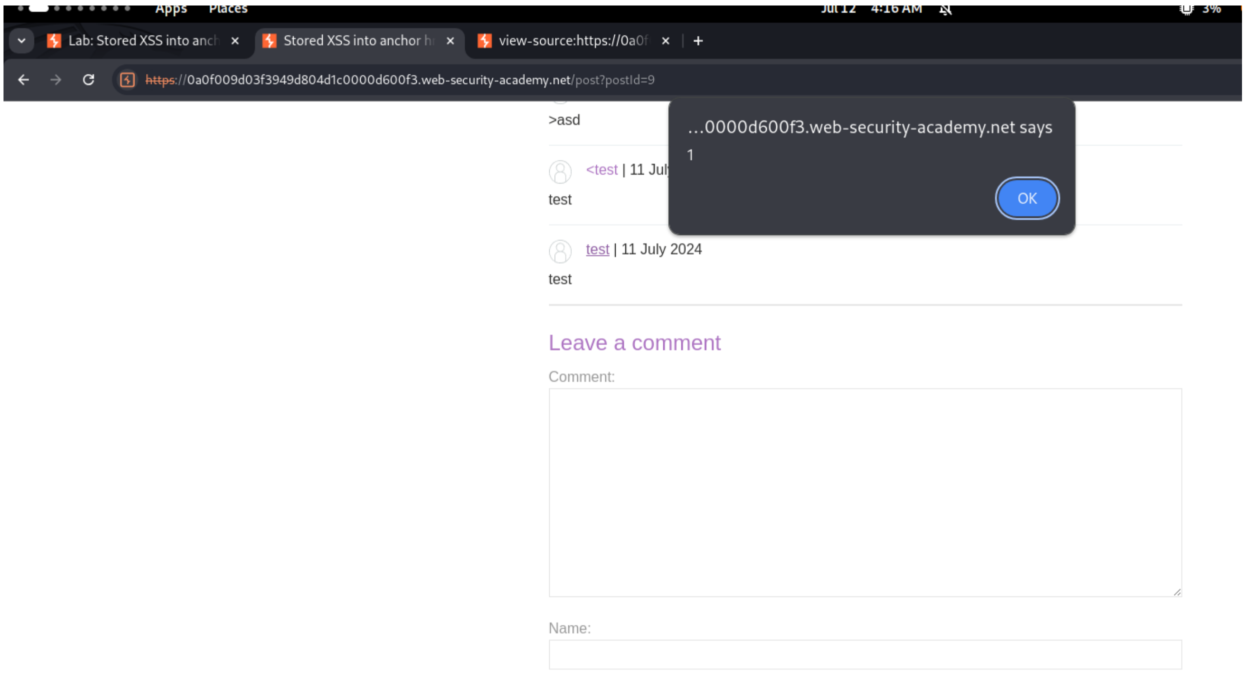
Comment:
test

Name:
test

Email:
test@gmail.com

Website:
javascript:alert(1)

- Injected the payload javascript:alert(1), which triggers an alert box.



https://0a0f009d03f3949d804d1c0000d600f3.web-security-academy.net/post?postId=9

>asd

...0000d600f3.web-security-academy.net says
1
OK

<test | 11 July
test

test | 11 July 2024
test

Leave a comment

Comment:

Name:

- By clicking the website link the alert popped up confirming the vulnerability
Mitigation:
To prevent XSS attacks in general, always sanitize and HTML-encode user inputs before rendering them in HTML context.