

# PORT SWIGGER

## Cross Site Script (xss)

**Cross-Site Scripting (XSS)** is a web security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can execute in the victim's browser, potentially stealing sensitive information, hijacking user sessions, or spreading malware. XSS attacks exploit improper validation of user input and can occur in forms, URLs, or cookies.

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

#### Lab: Reflected XSS into HTML context with nothing encoded

APPRENTICE  
LAB ✓ Solved



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.

ACCESS THE LAB

💡 Solution



This is the first lab of XSS in portswigger. The description explains there is a XSS vulnerability in the search functionality of the lab. To solve it we have to perform reflected XSS attack that calls the alert function.

**Reflected XSS** is a type of cross-site scripting attack where malicious scripts are immediately reflected off a web server and executed in the victim's browser. This occurs when user-supplied input (like from a form or URL parameter) is included in the response without proper validation or encoding. The attacker typically tricks the victim into clicking a crafted link that sends the malicious script to the server, which then reflects it back in the response. The script runs in the victim's browser, potentially allowing the attacker to steal sensitive information or perform actions on behalf of the victim.

Let's access the lab,

**WebSecurity Academy** Reflected XSS into HTML context with nothing encoded  
[Back to lab description >>](#)

Home



There is a search functionality in the website, let's input any simple javascript code on the search bar.

```
<script>alert("Hello world")</script>
```

0abb00c703f3e95f80dde11d00890099.web-security-academy.net says

Hello world

OK

After entering this code on the search bar and hitting search a pop up will show the “Hello world” on the screen.

Notice the end of the URL:

```
/?search=<script>alert("Hello+World")<%2Fscript>
```

After the question mark (?) is where the query string starts.

So we completed this lab 😊

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

## 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

⌚ Solution



⌚ Community solutions



This is the second lab of XSS in portswigger. The description explains there is a XSS vulnerability in the comment functionality of the lab. To solve it we have to perform stored XSS attack that calls the alert function.

**Stored XSS**, also known as **persistent XSS**, is a type of cross-site scripting attack where the malicious script is permanently stored on a target server, such as in a database, comment field, or message board. When other users access the affected page, the malicious script is served to their browsers and executed without any interaction from the attacker. This type of XSS can be more damaging than reflected XSS because it affects multiple users who visit the page where the malicious script is stored, leading to widespread exposure and potential data theft or session hijacking.

So let's submit a comment on the blog.

This is the lab,

**Web Security Academy** Stored XSS into HTML context with nothing encoded  
[Back to lab description >](#)

Home

WE LIKE TO BLOG



Let's explore the lab and open a blog, after that submit a simple xss script on the comment section.

<script>alert(1)</script>

I thank you for introducing me to a brand-new feeling. Enjoying being on the internet.

Mike Groin | 23 September 2024  
My mum said I can't go over until I've tidied my room.

**Leave a comment**

Comment:  
<script>alert(1)</script>

Name:  
fake name

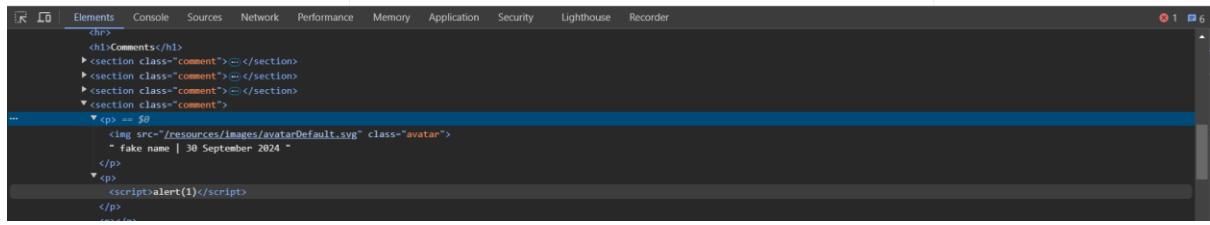
Email:  
fakemail@gmail.com

Website:

**Post Comment**

After submitting the lab, just refresh the page and you will get a pop message of 1. This means that you have successfully executed the attack.

So when we inspect the comment section we can still see our script



The screenshot shows a web page with a comment section. The comments are:

- Ann Anotherthing | 23 September 2024  
Thank you for introducing me to a brand-new feeling. Enjoying being on the internet.
- Mike Groin | 23 September 2024  
My mum said I can't go over until I've tidied my room.
- fake name | 30 September 2024

Below the comments is a "Leave a comment" form with a "Comment:" input field.

In the browser's developer tools (Elements tab), the DOM structure for the comments is visible. A specific comment from "fake name" is selected, showing its internal HTML structure:

```
<hr>
<h1>Comments</h1>
<section class="comment"></section>
<section class="comment"></section>
<section class="comment"></section>
<section class="comment">
  <p> == $0
    
    "Fake name | 30 September 2024"
  </p>
<p>
  <script>alert(1)</script>
</p>
</section>
```

So we completed this lab 😊

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

**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.

[ACCESS THE LAB](#)

[Solution](#)

This lab contains a DOM-based XSS vulnerability, this is 3<sup>rd</sup> lab of XSS in portswigger. This lab uses a javascript document.write function, which writes data out to the page. The document.write function is called with data from location.search. We have to solve this lab by performing xss that calls the alert function.

Let's see the lab.

WebSecurity Academy [DOM XSS in document.write sink using source location.search](#) Back to lab description >

WE LIKE TO BLOG

Search the blog... [Search](#)



There is a search bar available in the website, let's type any strings on the search bar and inspect the page.

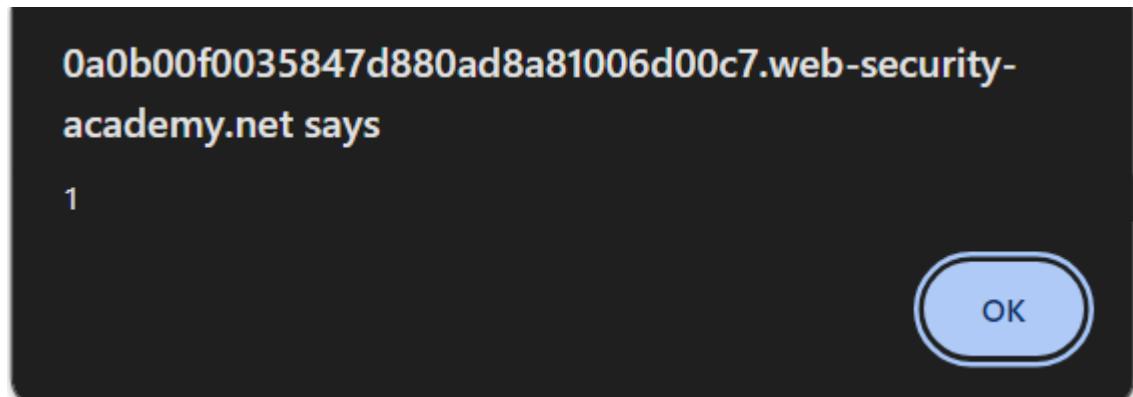
The screenshot shows a web browser window for the 'Web Security Academy' lab titled 'DOM XSS in document.write sink using source location.search'. The page displays '0 search results for 'abc123''. Below the search bar, the browser's developer tools are open, specifically the 'Elements' tab. The DOM tree shows the search results for 'abc123' and highlights an tag with the src attribute set to 'resources/images/tracker.gif?searchItems=abc123'. The browser's address bar shows the URL '0a0b00f0035847d880ad8a81006d00c7.web-security-academy.net'.

After submitting a random string I inspected the page and found that the `<img>` `</img>` attribute containing the random string I searched.

So let's break out the img attribute,

Here is the script,

```
"><svg onload=alert(1)>
```



Now we will get a pop of 1, so the Lab is finished

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

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

APPRENTICE

LAB

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



🧪 ACCESS THE LAB

💡 Solution



This lab contains a **reflected xss** vulnerability. It describes that this vulnerability presents on the search functionality of the blog. It is HTML encoded.

Let's access the lab.

Web Security Academy

Reflected XSS into attribute with angle brackets HTML-encoded

LAB Not solved



[Back to lab description >](#)

Home

WE LIKE TO BLOG

Search the blog...

Search



This is the lab, let's search something on the search bar and inspect the page.

```

        ...
    </section>
    ▼ <section class="search">
        ▼ <form action="/" method="GET"> flex
            ...
            <input type="text" placeholder="Search the blog..." name="search" value="abc" > == $0
            <button type="submit" class="button">Search</button>
        </form>
    </section>
    ▶ <section class="blog-list no-results"> ... </section>
</div>

```

html body div section.maincontainer div.container.is-page section.search form input

Here we can see the input I searched is **abc** and it is quoted so let's use a event attribute and inject event handler, “**onmouseover=“**

**“The onmouseover event occurs when the mouse pointer enters an element”**

**“onmouseover=“alert(1)”**

Let's see.

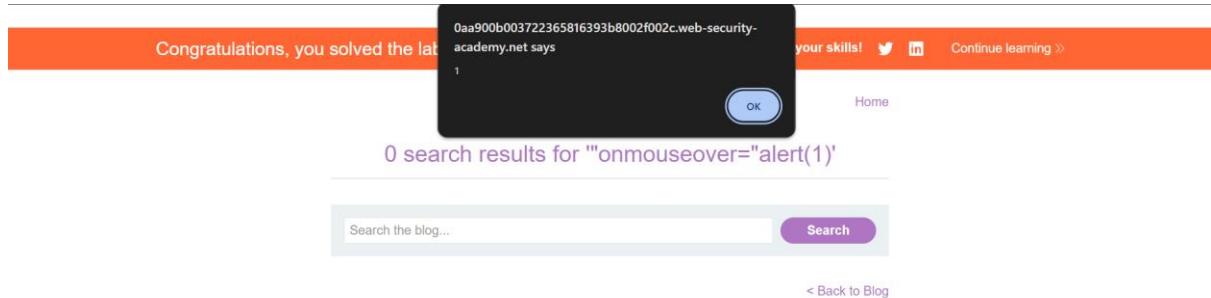
```

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder
    ...
    <section class="blog-header">
        <h1>0 search results for '"onmouseover="alert(1)"'</h1>
        <hr>
    </section>
    ▼ <section class="search">
        ▼ <form action="/" method="GET"> flex
            ...
            <input type="text" placeholder="Search the blog..." name="search" value "onmouseover="alert(1)"> == $0
            <button type="submit" class="button">Search</button>
        </form>
    </section>
    ▶ <section class="blog-list no-results"> ... </section>
</div>

```

html body div section.maincontainer div.container.is-page section.search form input

Here you can see that onmouseover event is reflected outside the encoded, hence the lab is completed u can see an alert “1” as coming as pop up.



We did completed the lab 😊

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

## Lab: Stored XSS into anchor href attribute with double quotes HTML-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 comment author name is clicked.



ACCESS THE LAB

Solution



This is a **Stored XSS** vulnerability lab in portswigger. From the description we can understand the vulnerability is on the comment functionality. To solve this, we need to call an alert function when the comment authors name is clicked.

Let's access the lab,

Web Security Academy

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

LAB Not solved

[Back to lab description >](#)

Home

WE LIKE TO BLOG



Let's see if the comment section is vulnerable to XSS, use this command on the comment section,

## Leave a comment

Comment:

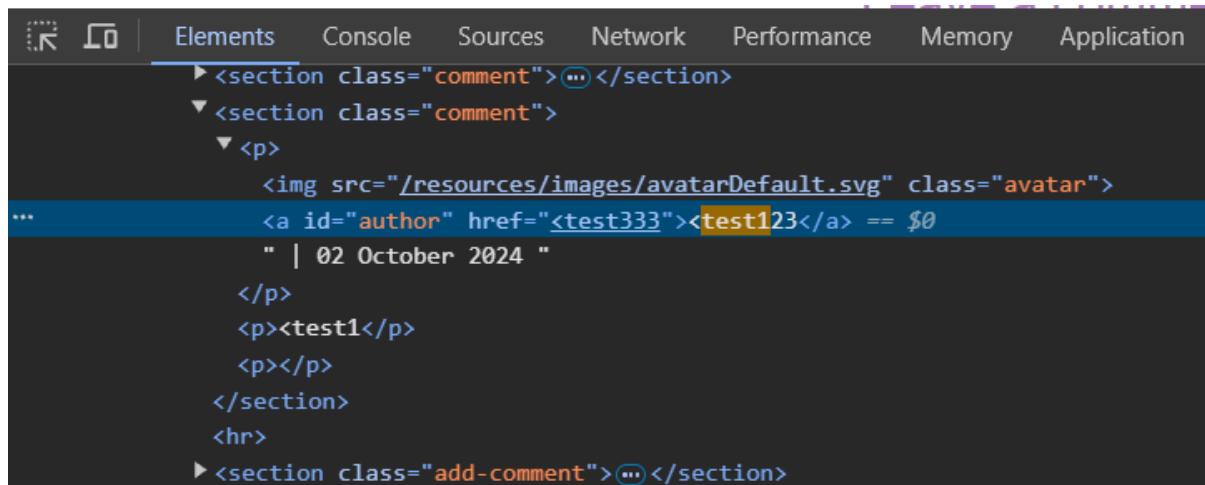
Name:

Email:

Website:

**Post Comment**

After submitting inspect the page after going back to blog.



Here we can see the **test333 & test123** is covered on a **href** attribute.

**“The href attribute specifies the URL of the page the link goes to”**

We can inject a **Javascript** to the code, which contain an anchor tag and inside the reflection which is occurring inside a href attribute. Try this script,

**javascript:alert(123)**

So return to the comment field and enter again the sections, and in the website column enter the **Javascript** code.

### Leave a comment

Comment:

fake

Name:

fake123

Email:

fake@fake.com

Website:

javascript:alert(1)

**Post Comment**

Enter the section like you see on the screenshot.

The screenshot shows a solved lab page from the Web Security Academy. At the top, it says "Stored XSS into anchor href attribute with double quotes HTML-encoded". Below that is a "Back to lab description" link. A prominent orange banner at the top states "Congratulations, you solved the lab!". To the right of the banner are links to "Share your skills!" (with icons for Twitter and LinkedIn) and "Continue learning >". At the bottom of the page, there's a "Home" link, a "Thank you for your comment!" message, and a note saying "Your comment has been submitted.". There's also a link "[< Back to blog](#)".

Yes we completed the lab 😊

# Lab 6: Reflected XSS into a JavaScript string with angle brackets HTML encoded

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

APPRENTICE  
LAB ✓ 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 the description describes there is a **reflected xss** vulnerability in the search query of the lab. The reflection occurs inside a JavaScript string.

Let's see the lab,

Web Security Academy

Reflected XSS into a JavaScript string with angle brackets HTML encoded

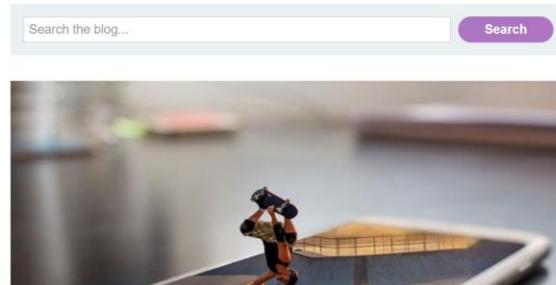
[Back to lab description >](#)

LAB Not solved



[Home](#)

WE LIKE TO BLOG



Type any random strings on the search bar, then inspect the page,

```
<hr>
</section>
► <section class="search">④ </section>
▼ <script>
...
    var searchTerms = 'abc123';
    document.write('');
    == $0
</script>

► <section class="blog-list no-results">④ </section>
</div>
</section>
```

Here during the document.write process, all characters are encoded using encodeURIComponent, so XSS won't work from now on. So we needed to add the XSS payload in the place where 'abc123' was placed.

So let's use this, '-alert(1)-'

```
▼ <script>
    var searchTerms = '--alert(1)--';
    document.write('');
    == $0
</script>
```

Here the ' turns into a character and other part into a comment. And here the problem is solved.

The screenshot shows a challenge page from WebSecurity Academy. At the top, it says "Reflected XSS into a JavaScript string with angle brackets HTML encoded". Below that, there's a green "Solved" button. The main content area has an orange banner at the top saying "Congratulations, you solved the lab!". Below the banner, it says "Share your skills! 🐦 LinkedIn Continue learning >". Underneath, there's a "Home" link and a search bar with the placeholder "Search the blog...". The search results section displays the message "0 search results for \"-alert(1)-\"".

The lab is done 😊

