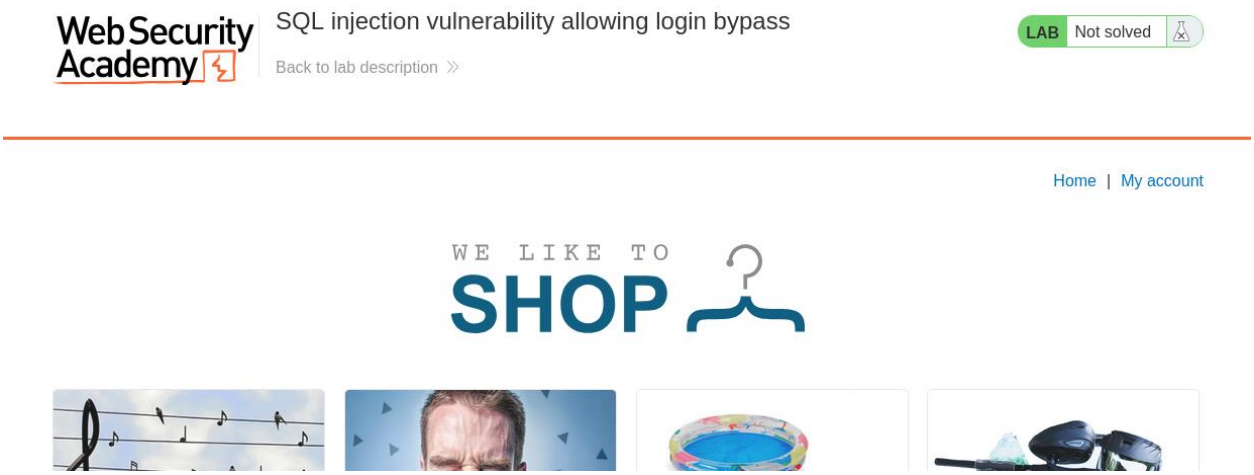
Day 4

Task 1: SQL Injection

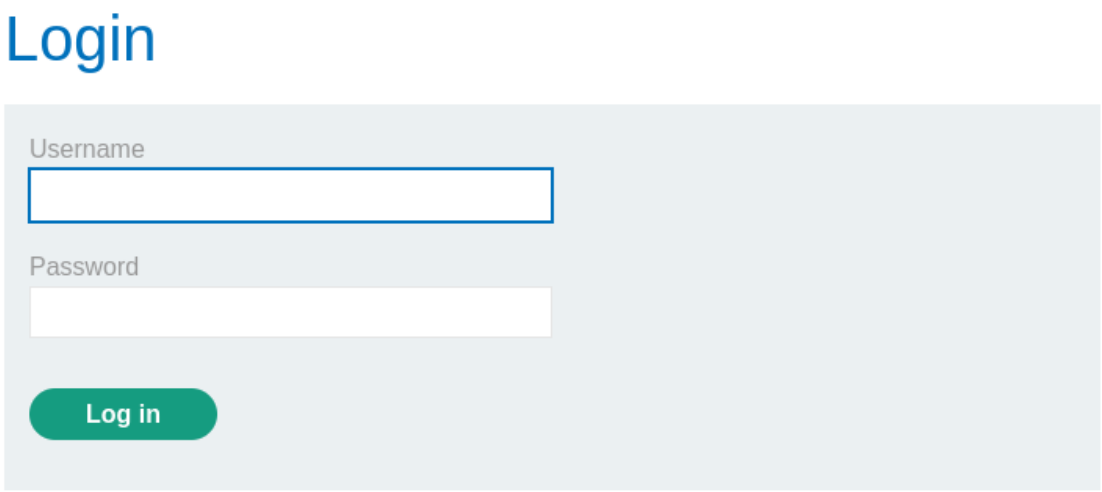
Lab 1: SQL injection vulnerability allowing login bypass

This lab contains a SQL injection vulnerability in the login function. To solve the lab, we perform a SQL injection attack that logs in to the application as the administrator user.

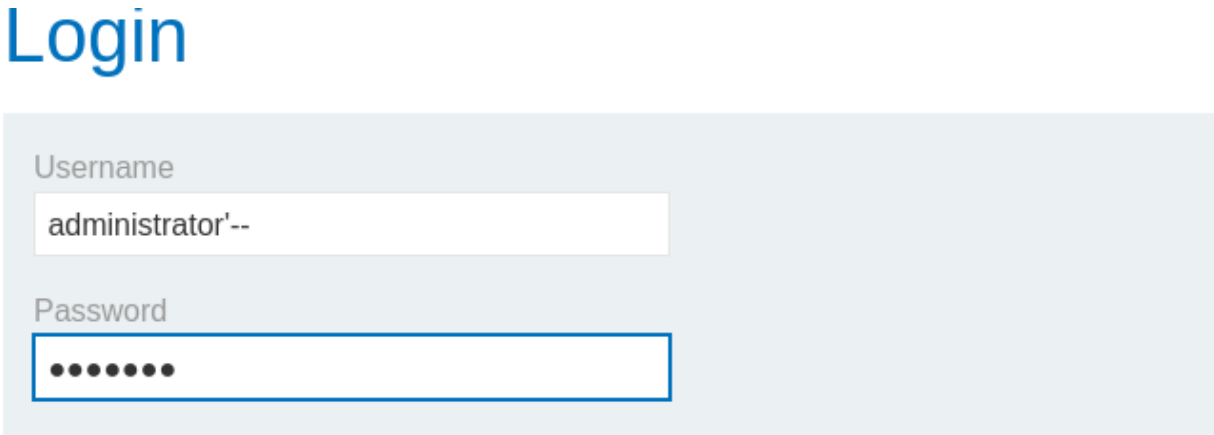
- First I open the website and click on my account



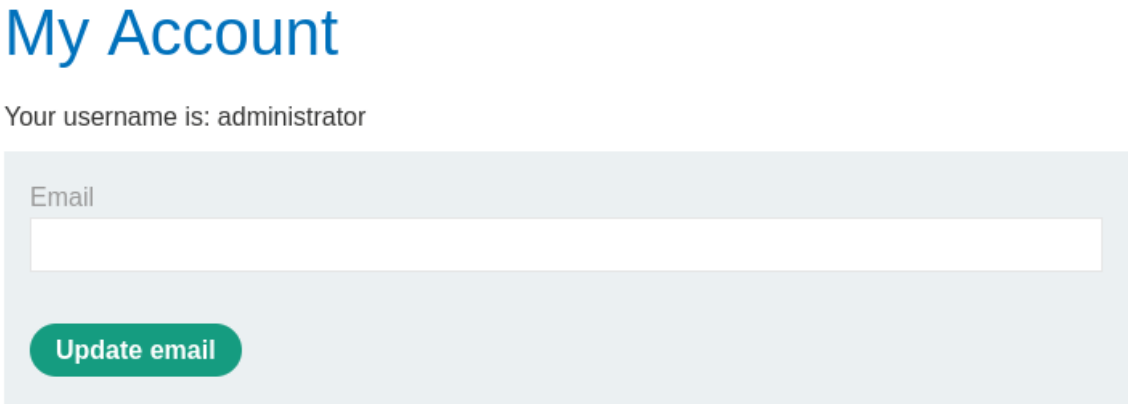
- And now it shows me the login portal and that's where I'll do my attack



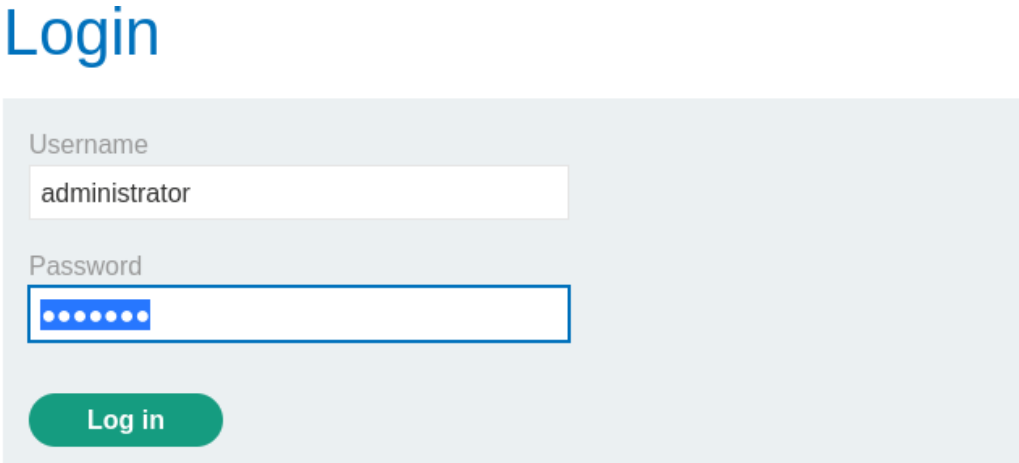
- Since it asked me to log in to the administrator account I filled in the parameter in this way to bypass the check in and log me into the account without checking the password as I filled it randomly



- As you can see here the attack was successful and I was able to gain access



- Another way to the attack is to try to log in normally and don't worry about the password as we will intercept the request using burp



- After we intercept it we will modify the username parameter as following then forward it

```
Te: trailers
Connection: keep-alive

csrf=vukr4ZtIvBa0zCrvXKKt5AvMKEUBrjpy&username=administrator'--&password=3132132
```

Search 0 highlights

Ready

- Now you can see that the attack was successful as well

Pretty Raw Hex Render

```
HTTP/2 302 Found
Location: /my-account?id=administrator
Set-Cookie: session=
e4L4rdM8WR2IPR56b70Nme3Ke00DkKaK; Secure;
HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Lab 2: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

This lab contains a SQL injection vulnerability in the product category filter. When the user selects a category, the application carries out a SQL query like the following:

“SELECT * FROM products WHERE category = 'Gifts' AND released = 1”

To solve the lab, we perform a SQL injection attack that causes the application to display one or more unreleased products.

- First we get to the website and click on any of the categories “in this case GIFTS”

https://0a2900b2042283218004df6900570019.web-security-academy.net/filter?category=Gifts

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Web Security Academy hidden data

Back to lab home Back to lab description

Home

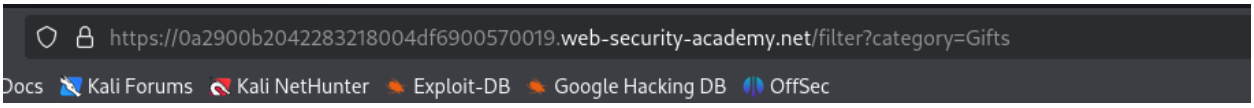
WE LIKE TO SHOP

Gifts

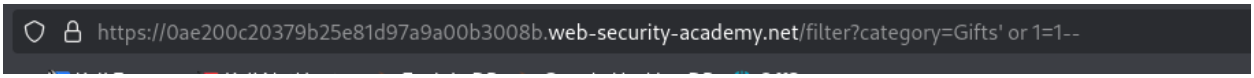
Refine your search:

All Accessories Food & Drink Gifts Lifestyle

- If you check the URL you can see the category parameter that we want to modify



- Once we modify the URL as following we'll be able to show the hidden category



- Once you finish and press enter you will find out that the attack was successful

Gifts' or 1=1--

- Note: you can do the same steps using burp by intercepting then modifying the category parameter in the URL as we did earlier and then forward the request. You will get the same output.

sql injection mitigation:

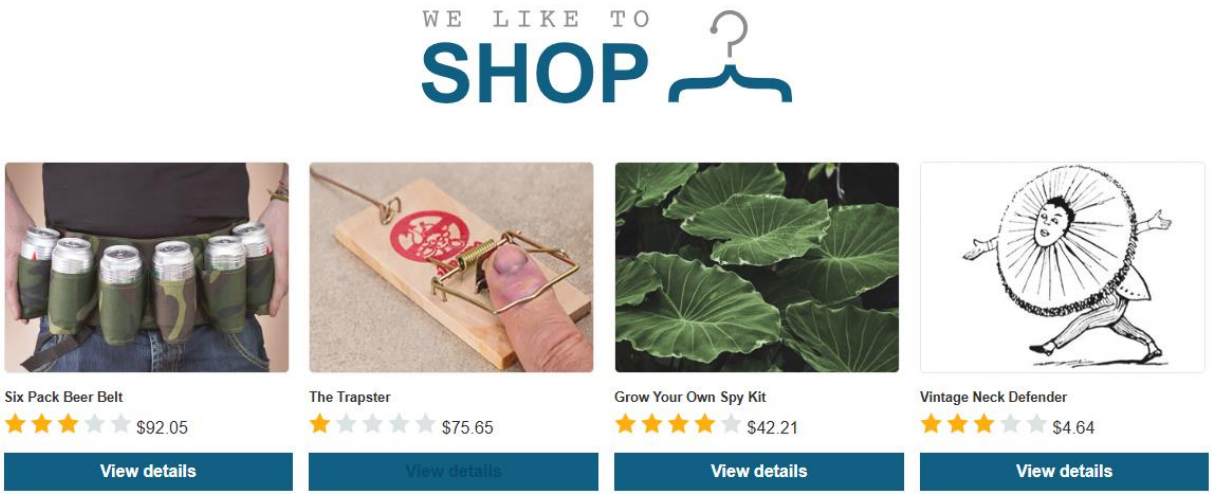
1. **use parameterized queries** – instead of inserting user input directly into SQL statements, use placeholders (like ?) to ensure inputs are treated as data, not code.
2. **implement stored procedures** – encapsulate SQL code within the database using stored procedures, which can accept parameters and reduce direct interaction with SQL statements.
3. **validate user inputs** – ensure that all user-provided data matches expected formats (e.g., numbers, specific strings) before processing.
4. **limit database permissions** – assign only necessary privileges to database users, following the principle of least privilege to minimize potential damage from an injection attack.
5. **use a web application firewall (WAF)** – deploy a WAF to monitor and filter out malicious traffic patterns, adding an extra layer of defense against SQL injection attempts.

Task 2: Path Traversal

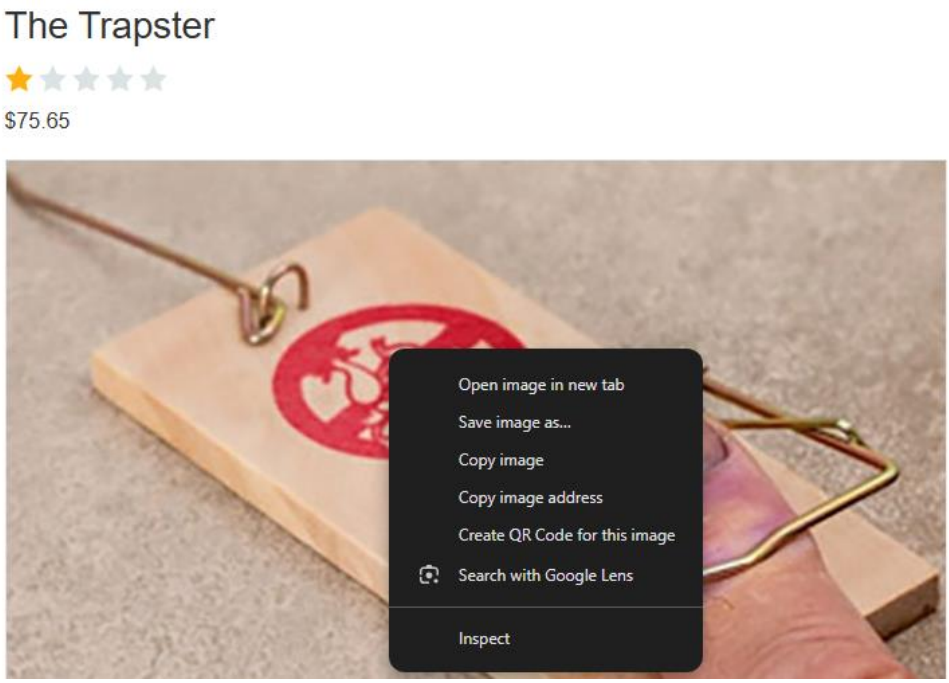
Lab 1: File path traversal, simple case

This lab contains a path traversal vulnerability in the display of product images. To solve the lab, we retrieve the contents of the /etc/passwd file.

- First we start by opening the lab and then we click on one of the products details



- Then after a right click on the image we open it in a new tab



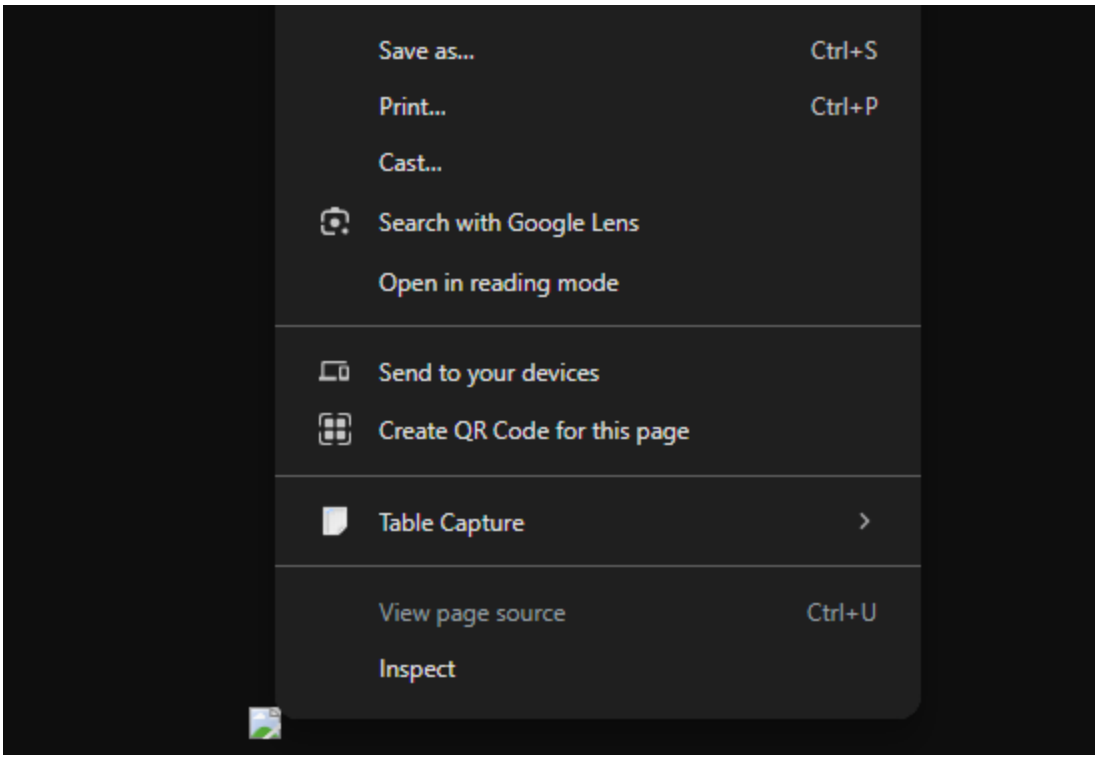
- As we can see in the url there's a parameter for the filename and that will be our starting point

0a3000a004ed129d803430a0009e00d2.web-security-academy.net/image?filename=41.jpg

- We will modify the parameter to locate the folder passwd and I first tried with the absolute path then I put "../" once before it and kept trying till I reached the file the path that contains the file

0a3000a004ed129d803430a0009e00d2.web-security-academy.net/image?filename=../../../etc/passwd

- After you reach the path you will find the folder shown as a broken image just save it and open it with a text editor



- After you open it with the text editor you will be able to see the content of the wanted file

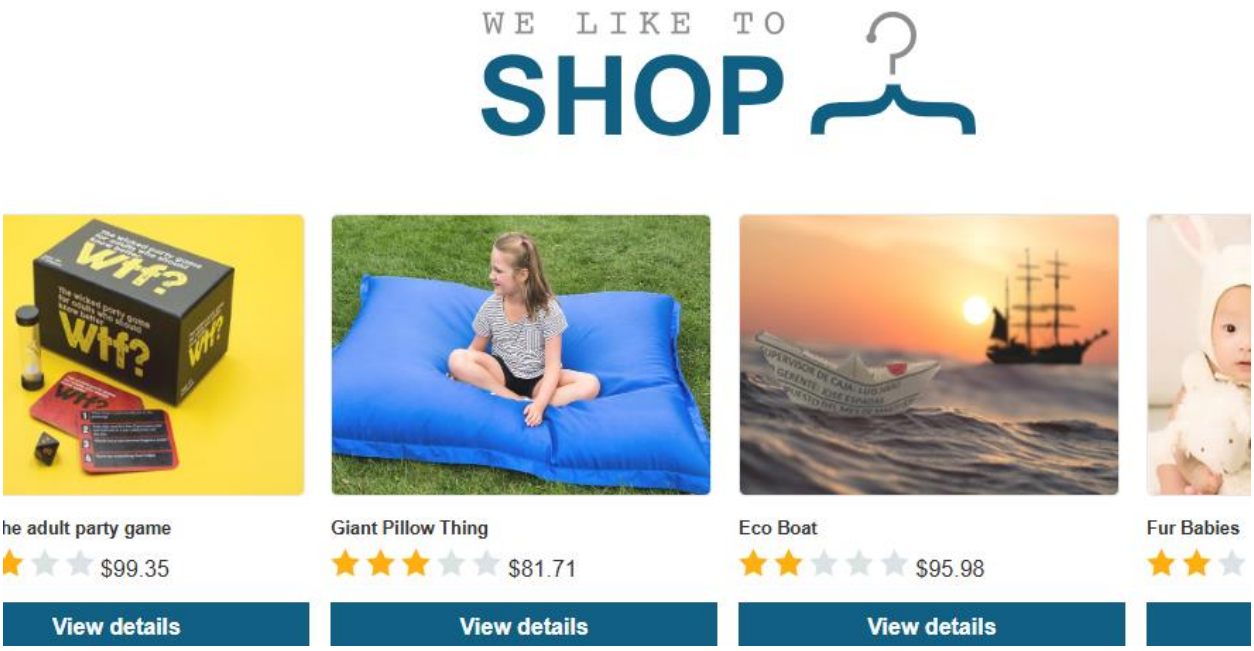
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

- Note: this could also be done on the burp by intercepting the request then modifying the parameter and then forwarding the request and you will be able to view the folder

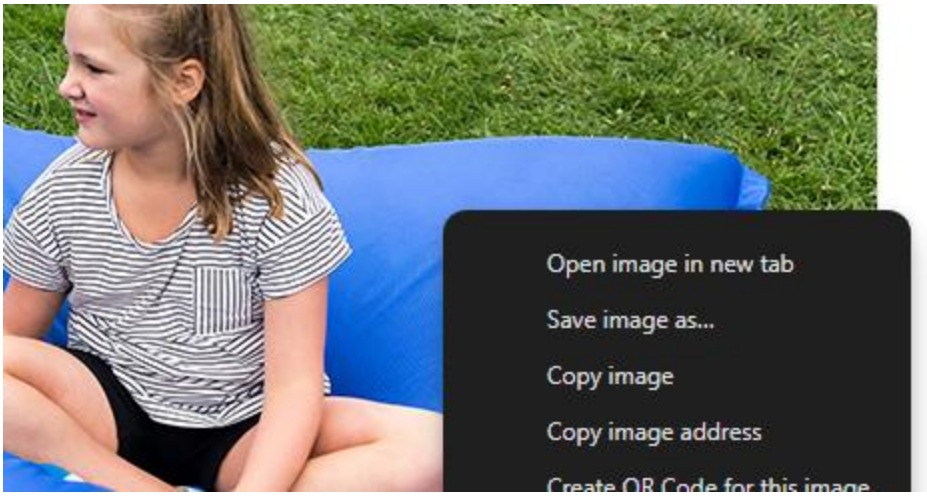
Lab 2: File path traversal, traversal sequences blocked with absolute path bypass

This lab contains a path traversal vulnerability in the display of product images. The application blocks traversal sequences but treats the supplied filename as being relative to a default working directory. To solve the lab, we retrieve the contents of the /etc/passwd file.

- We will do similar steps like the previous lab and check one of the products



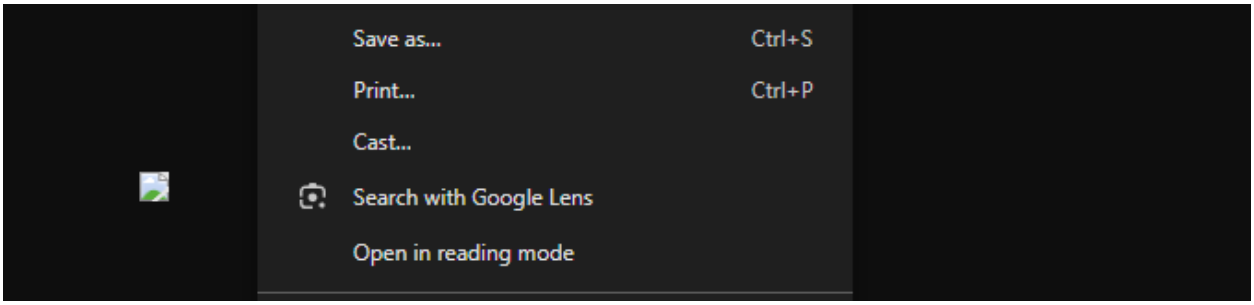
- Then we open the image in a new tab to be able get access to the URL



- In the URL we can modify the filename parameter with the path but this time with absolute one

- After typing the absolute path you will be able to reach the file directly

- Now it shows the file as broken image, save it and then open it with text editor



- Now you can view the file content as requested

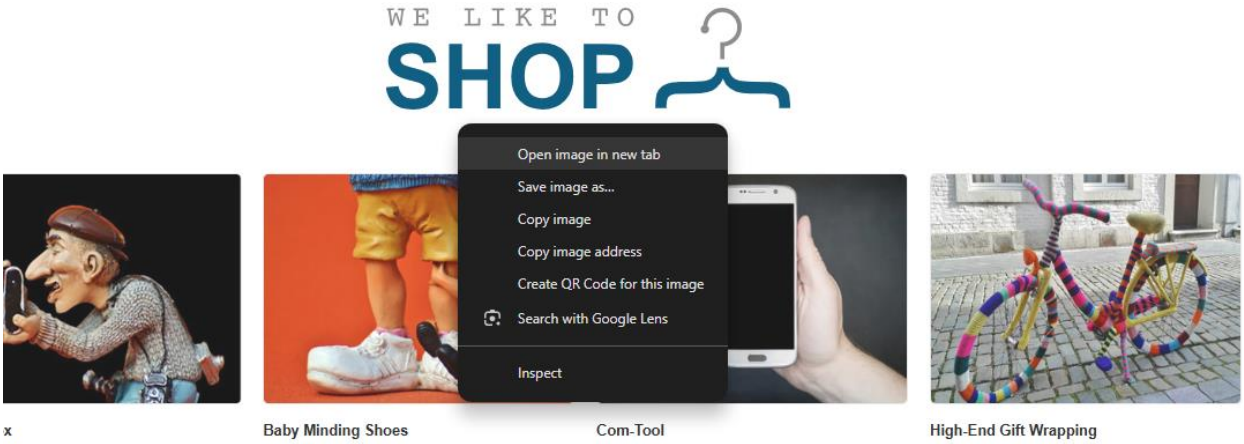
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

- Note: this could also be done on the burp by intercepting the request then modifying the parameter and then forwarding the request and you will be able to view the folder

Lab 3: File path traversal, traversal sequences stripped non-recursively

This lab contains a path traversal vulnerability in the display of product images. The application strips path traversal sequences from the user-supplied filename before using it. To solve the lab, we retrieve the contents of the /etc/passwd file.

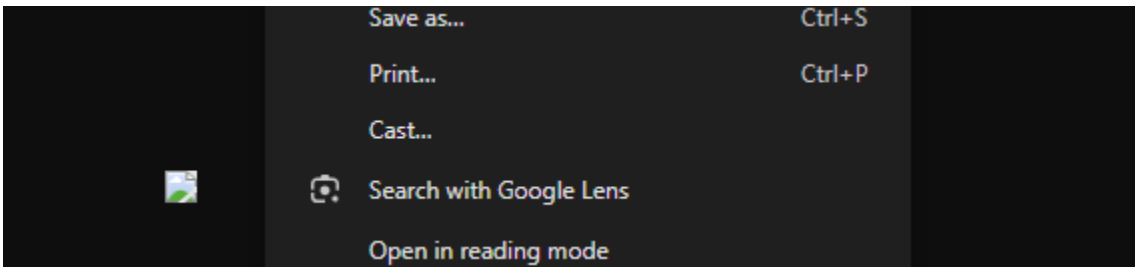
- First as the previous labs I will open the image of one of the products to check the URL



- We will modify the filename parameters as we did but this time is different as the system checks for the path traversal

- Luckily on this website it doesn't check on the path traversal recursively so all I needed to do was to put the traversal twice so when it finds it and remove it once the rest will stay leaving us with a normal path traversal '../../../etc/passwd'

- Now it shows the file as broken image, save it and then open it with text editor



- Now you can view the file content as requested

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

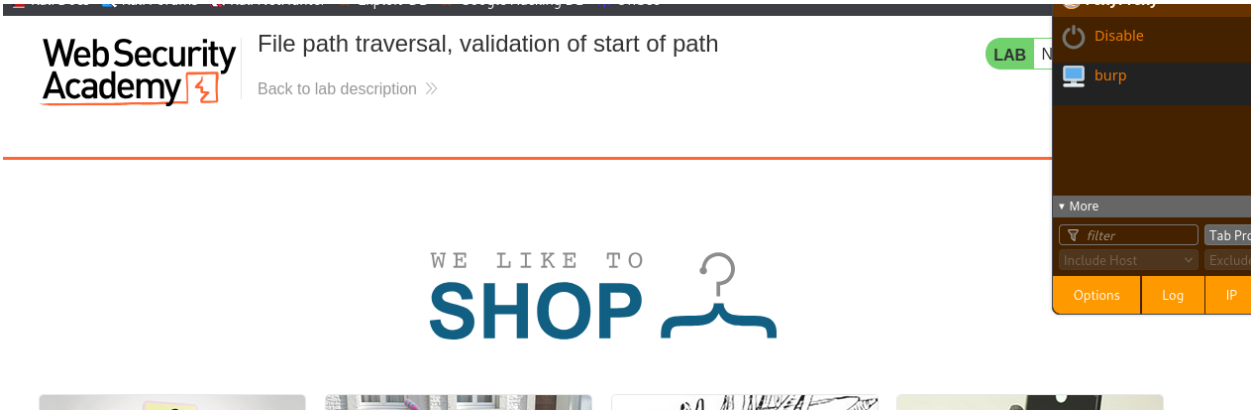
- Note: this could also be done on the burp by intercepting the request then modifying the parameter and then forwarding the request and you will be able to view the folder

Lab 4: File path traversal, validation of start of path

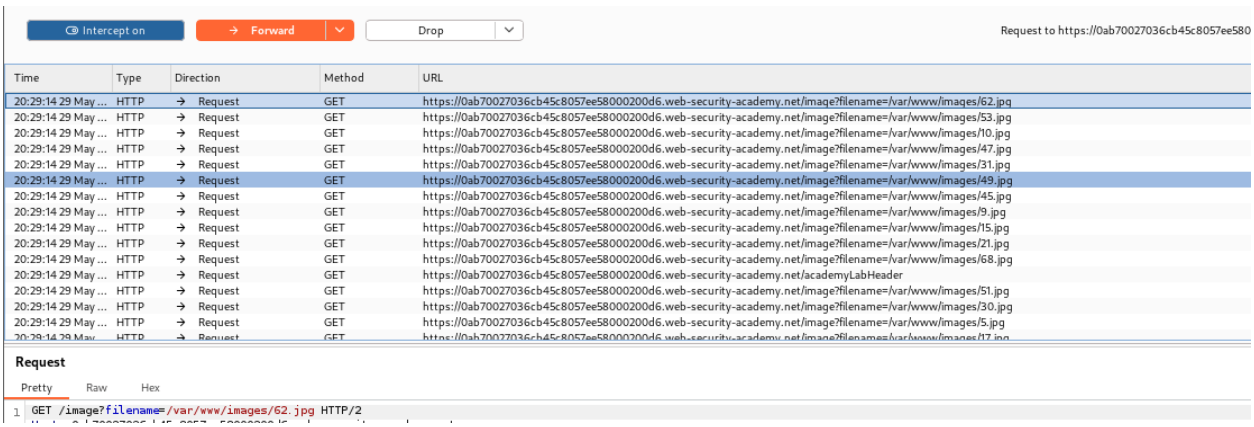
This lab contains a path traversal vulnerability in the display of product images. The application transmits the full file path via a request parameter, and validates that the supplied path starts with the expected folder. To solve the lab, we retrieve the contents of the /etc/passwd file.

- Same steps like the previous but this time we will use burp instead of editing the URL

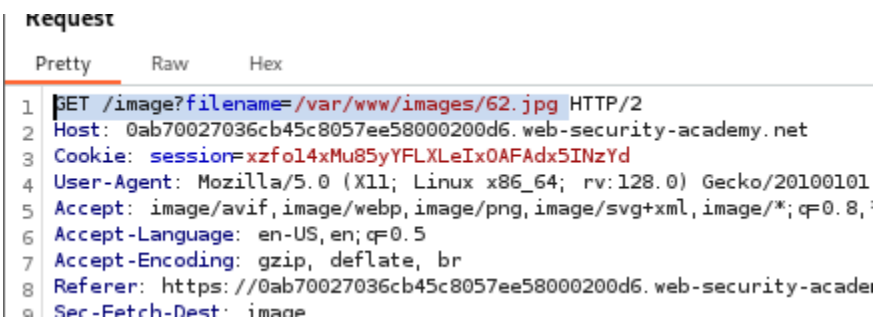
- I first opened the website then turned on the foxyproxy extension and the interception on burp



- Burp will show you multiple requests to the location of the picture so we send one of them to the repeater

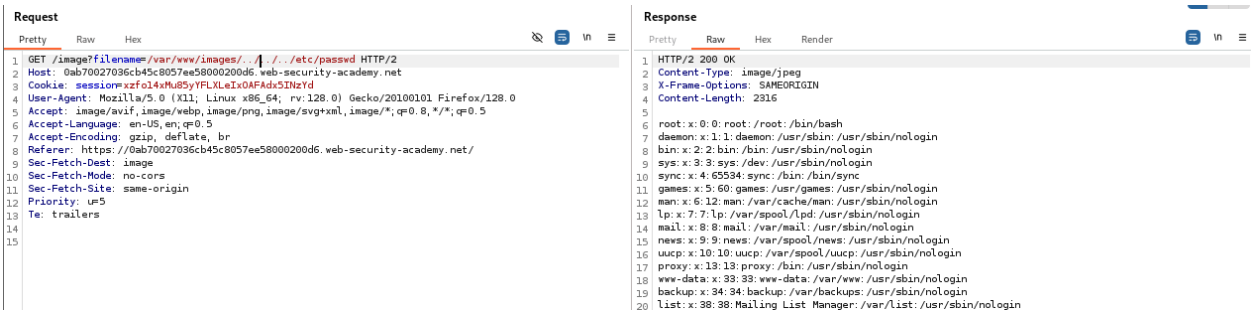


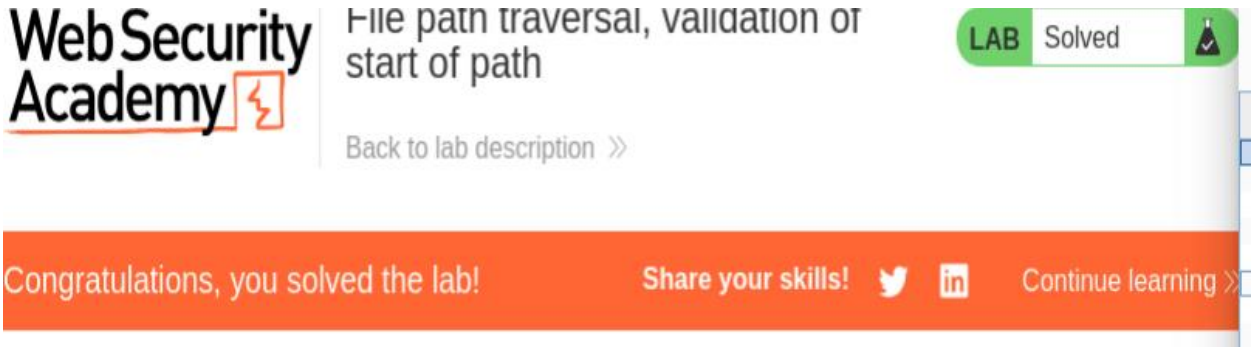
- When we examine the request we find out that the path is in the absolute form which gives us the chance to try the absolute path as well for the passwd file



- first I gave a try with a traversal path but it didn't work so I kept trying to find the right one

- after a couple of tries I was able to find the correct path and when I forwarded the request I was able to reach the file as requested





path traversal mitigation:

1. **avoid user input in file paths** – don't let users specify file names or paths directly. if needed, use predefined options or indexes instead.
2. **validate and sanitize inputs** – check that user inputs don't contain dangerous patterns like ../ or %2e%2e%2f. use allow lists to permit only safe inputs.
3. **normalize and verify paths** – convert file paths to their absolute form and ensure they reside within the intended directory. functions like realpath() in PHP or Path.Combine() in .NET can help.
4. **implement least privilege** – ensure the application has only the necessary permissions to access files, reducing potential damage if an attack occurs.
5. **use chroot jails or virtual directories** – restrict the application's access to a specific portion of the file system, preventing it from navigating to sensitive areas.

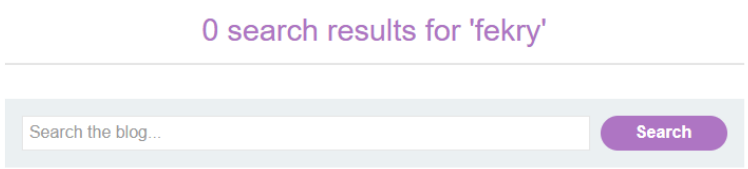
Task 3: XSS

Lab1: Reflected XSS into HTML context with nothing encoded

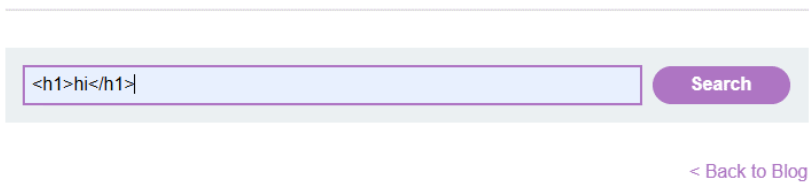
This lab contains a simple reflected cross-site scripting vulnerability in the search functionality.

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

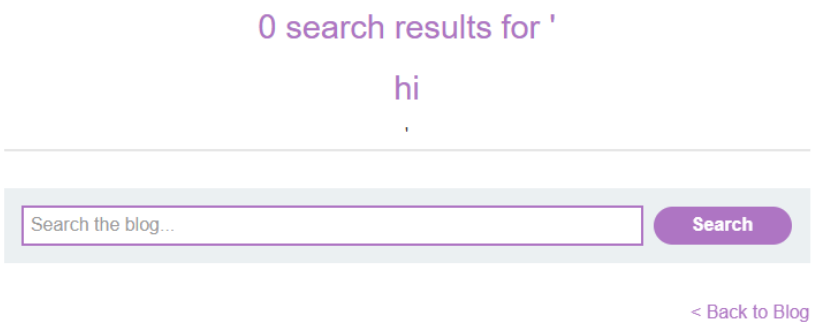
- First I tried to do any type of search and obviously the output was 0 match



- Then I tried some html injection by just adding a header like that



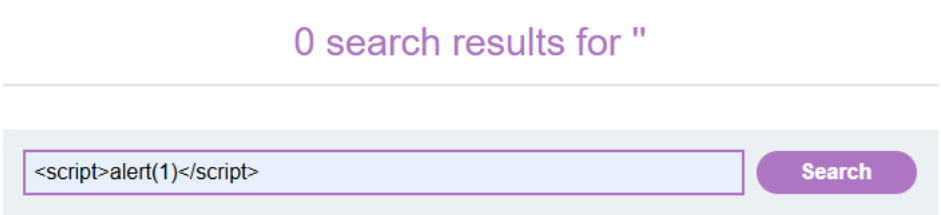
- And the result was surprising as the website didn't stop the injection and the command was reflected on screen



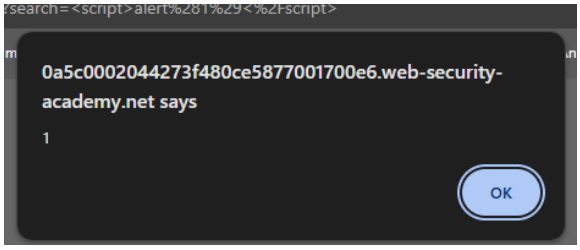
- When I clicked on inspect I was able to find the html command injected in the main code

```
<h1>0 search results for '</h1>
<h1>hi</h1> == $0
"' "
<hr>
```

- Then I tried XSS payload injection in this way to activate the java command to just give an alert



- After I clicked on search after typing the command an alert popped up

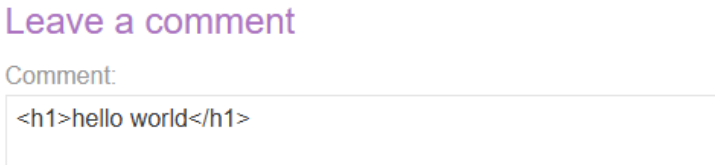


Lab2: Stored XSS into HTML context with nothing encoded

This lab contains a stored cross-site scripting vulnerability in the comment functionality. To solve this lab, we submit a comment that calls the alert function when the blog post is viewed.

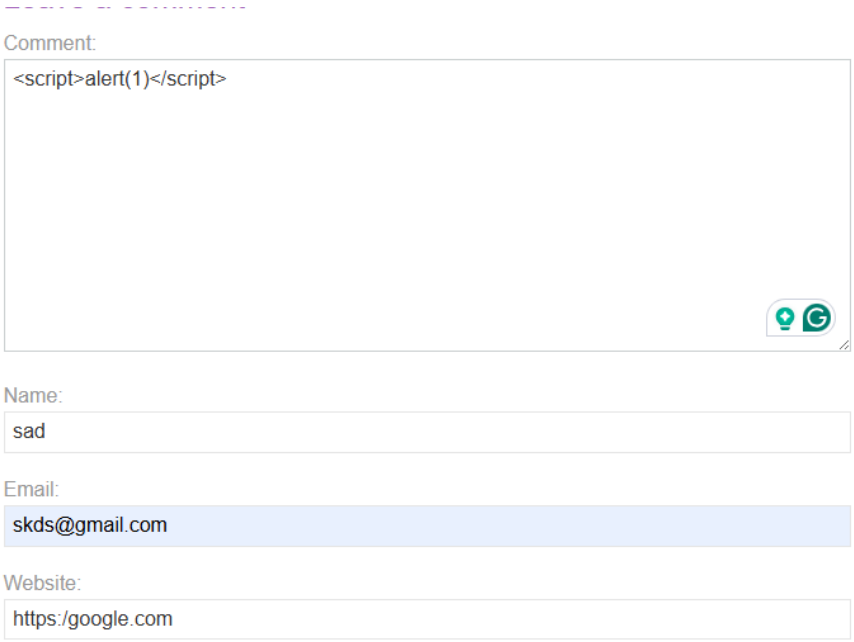
- Once we open the website it will take us to a blog and I will see what I can do with stored xss

- Once we open one of the topics we will be able to see the comment area. It accepts normal input so I tried first with the html injection as following to notice what will happen

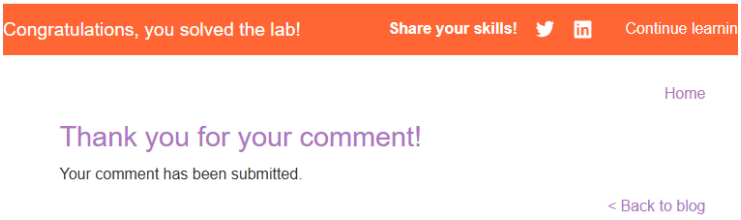


- And as expected it accepted the comment and stored it with showing it as a header not a comment

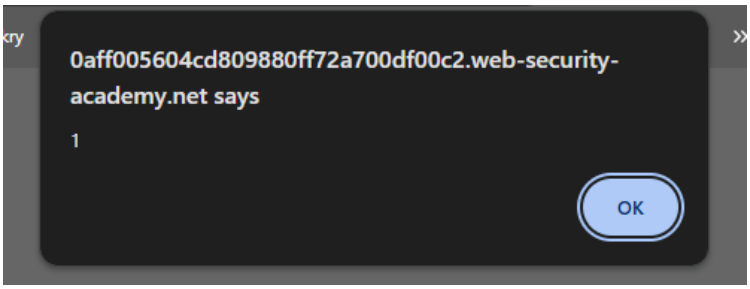
- Now after we verified that it's working with html injection it was time to start using the java script for the alert and after that I simply filled the rest with a bogus data



- And as expected it accepted the input but it won't show immediately the alert



- Once we go back to blog the alert will automatically pops up

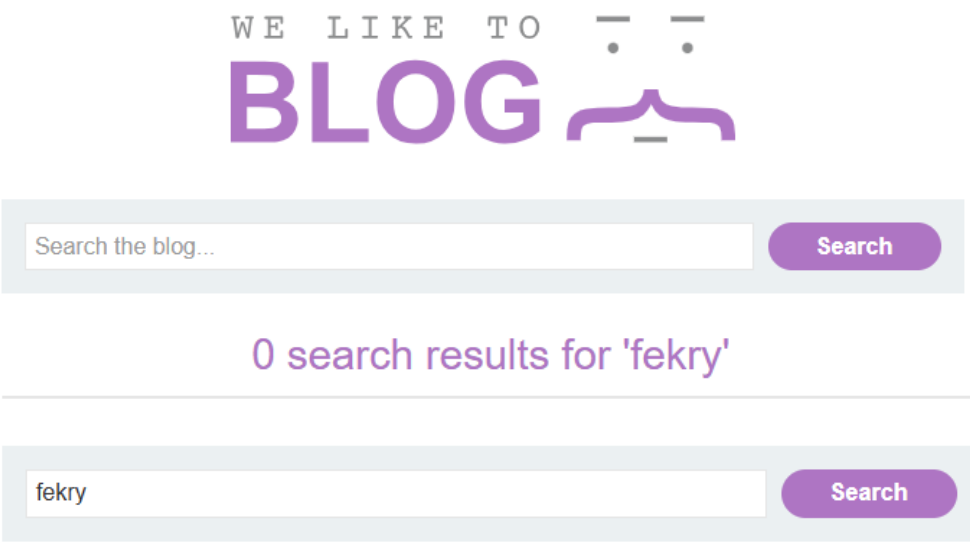


- Note: let's say I wanted to share the URL for the blog from the outside it will look normal but as we open the blog whenever we go to the same topic we posted our alert on, the alert will always pop up as it's already stored on the server

Lab3: 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.

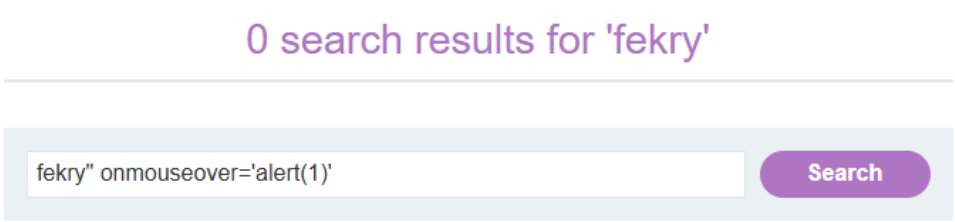
- Once we open the lab it will direct us to the blog where we want to attack so first I search with something normal to notice its behavior to understand how I will do my attack



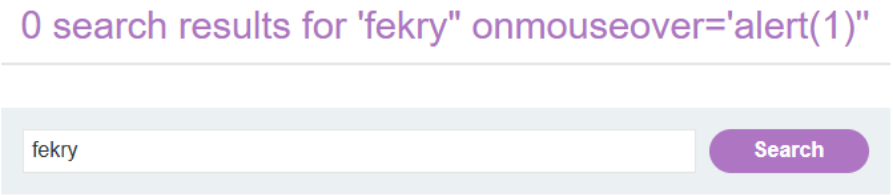
- As we can see the search term is shown in 2 different lines (in the header & the attribute value). What I'm trying to do is to break out of that attribute and raise an alarm whenever the mouse goes over the search bar

```
<section class=blog-header>
    <h1>0 search results for 'fekry'</h1>
    <hr>
</section>
<section class=search>
    <form action=/ method=GET>
        <input type=text placeholder='Search the blog...' name=search value="fekry">
        <button type=submit class=button>Search</button>
```

- type the same term along side the rest of the command as shown in search column to execute the attack and click on search



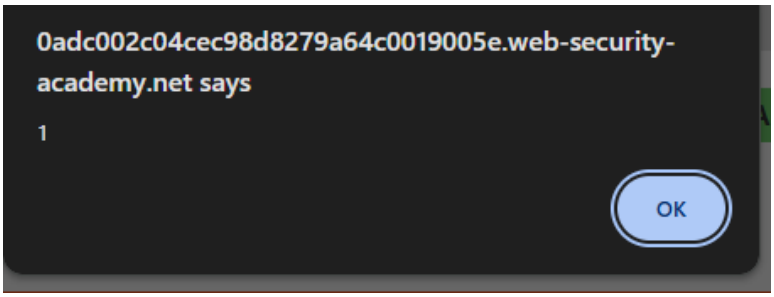
- from the outside it shows you the term and alert function in the header without any alarms popping up



- if we view the page source we will find that we were able to break out of the attribute and add our alarm function in a way where the page html accepted it

```
<form action=/ method=GET>
    <input type=text placeholder='Search the blog...' name=search value="fekry" onmouseover='alert(1)'">
    <button type=submit class=button>Search</button>
</form>
```

- all is left is just to move the mouse over the search bar and the alarm will automatically pop up



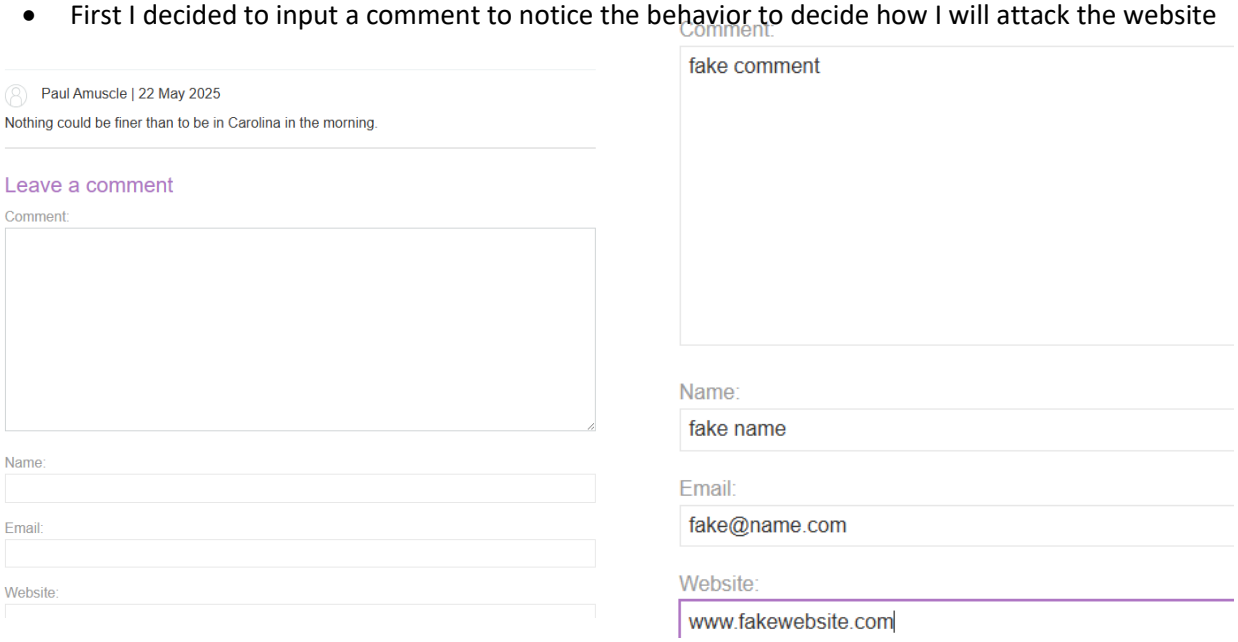
- As you can see the alert is already embedded into the URL so that means whenever the victim tries to open it the attack will take effect immediately

Lab4: Stored XSS into anchor href attribute with double quotes HTML-encoded

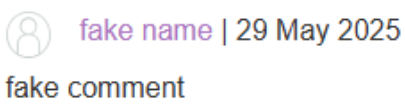
This lab contains a stored cross-site scripting vulnerability in the comment functionality. To solve this lab, submit a comment that calls the alert function when the comment author name is clicked.

- The website is a blog and we will open any of the topics then scroll down to the comments area

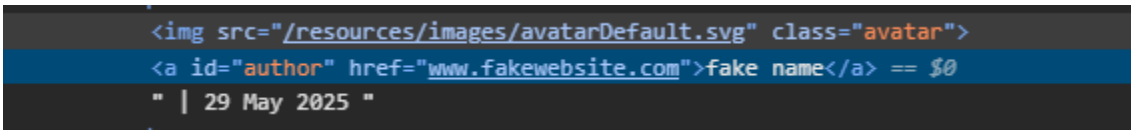
- First I decided to input a comment to notice the behavior to decide how I will attack the website



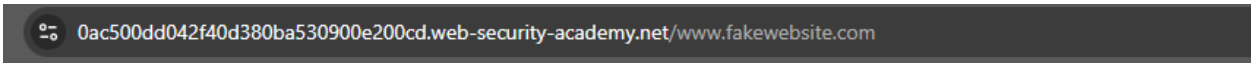
- We can see that the name is highlighted and if you click on it, it will direct you to the website I provided



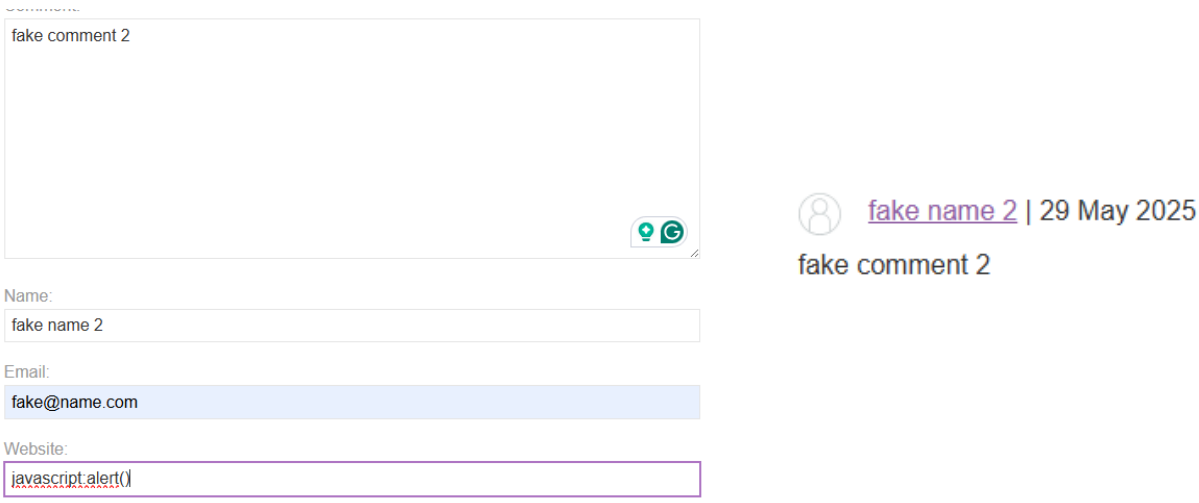
- And if we check the inspect tab we will find that the developer made the website is attached to the name as a hyperlink



- And when we open it we will see that it the website is appended to the blog website as well



- After understanding the concept of the website it is easier now to attach any alerts or attacks as a hyperlink to the name which I did in this case using java script alert function



- If we check the inspect we will find the alert function is attached to the name successfully

```
<img src="/resources/images/avatarDefault.svg" class="avatar">
<a id="author" href="javascript:alert()">fake name 2</a> == $0
" | 29 May 2025 "
```

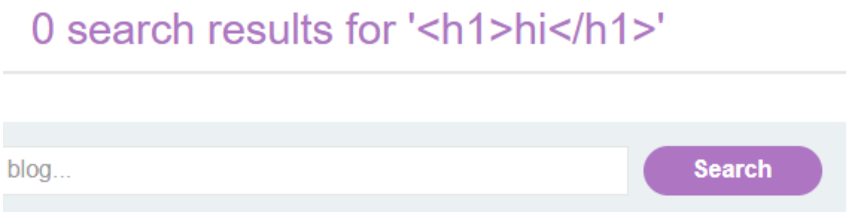
- Whenever we try to click on the name it will raise an alert and it will always be there unless the developer deletes it

Lab 5: 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.

- In this lab it will send me to the search bar as previous labs so I first tried the html injection

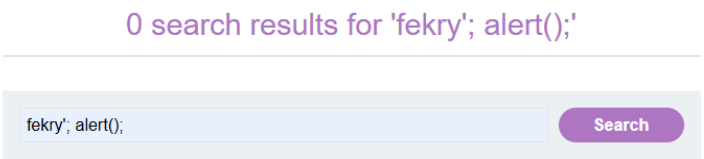
- I found out the developer protected himself from it and it consider it as a normal search term



- I tried then to search for a normal term and notice the behavior in the inspect window and it uses a java script as shown

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

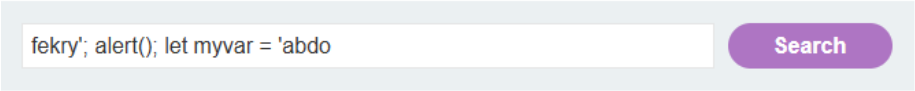
- Then I tried first using the search term and adding am alert function



- In the inspect window I noticed I was able to break out of the attribute but not fully

```
var searchTerms = 'fekry'; alert();';
document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
```

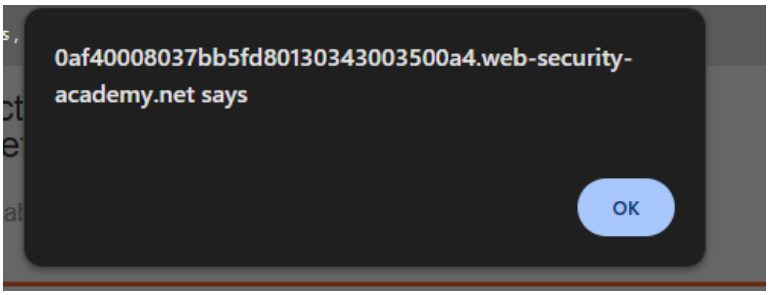
- The only viable solution for me is after the alert I add a new variable as follows



- What happened is we injected an alert between 2 variables so it was able to read it

```
<script> == $0
var searchTerms = 'fekry'; alert(); let myvar = 'abdo';
document.write('<img src="/resources/images/tracker.gif?searchTerms='+encodeURIComponent(searchTerms)+'">');
```

- You can see now that the alert is working successfully



- And if we check the URL we will find the attack is already embedded into it so whenever someone opens it will show an alert

xss mitigation:

1. **validate user input** – check that all user inputs match expected formats (e.g., alphanumeric characters) to prevent malicious scripts.
2. **encode output** – before displaying user input on web pages, convert special characters to HTML entities (e.g., < becomes <) to prevent browsers from interpreting them as code.
3. **implement content security policy (csp)** – set up CSP headers to restrict the sources from which scripts can be loaded, reducing the risk of executing malicious scripts.
4. **use http-only cookies** – set the HttpOnly flag on cookies to prevent client-side scripts from accessing them, protecting sensitive session information.