



TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

WEB SECURITY



Content

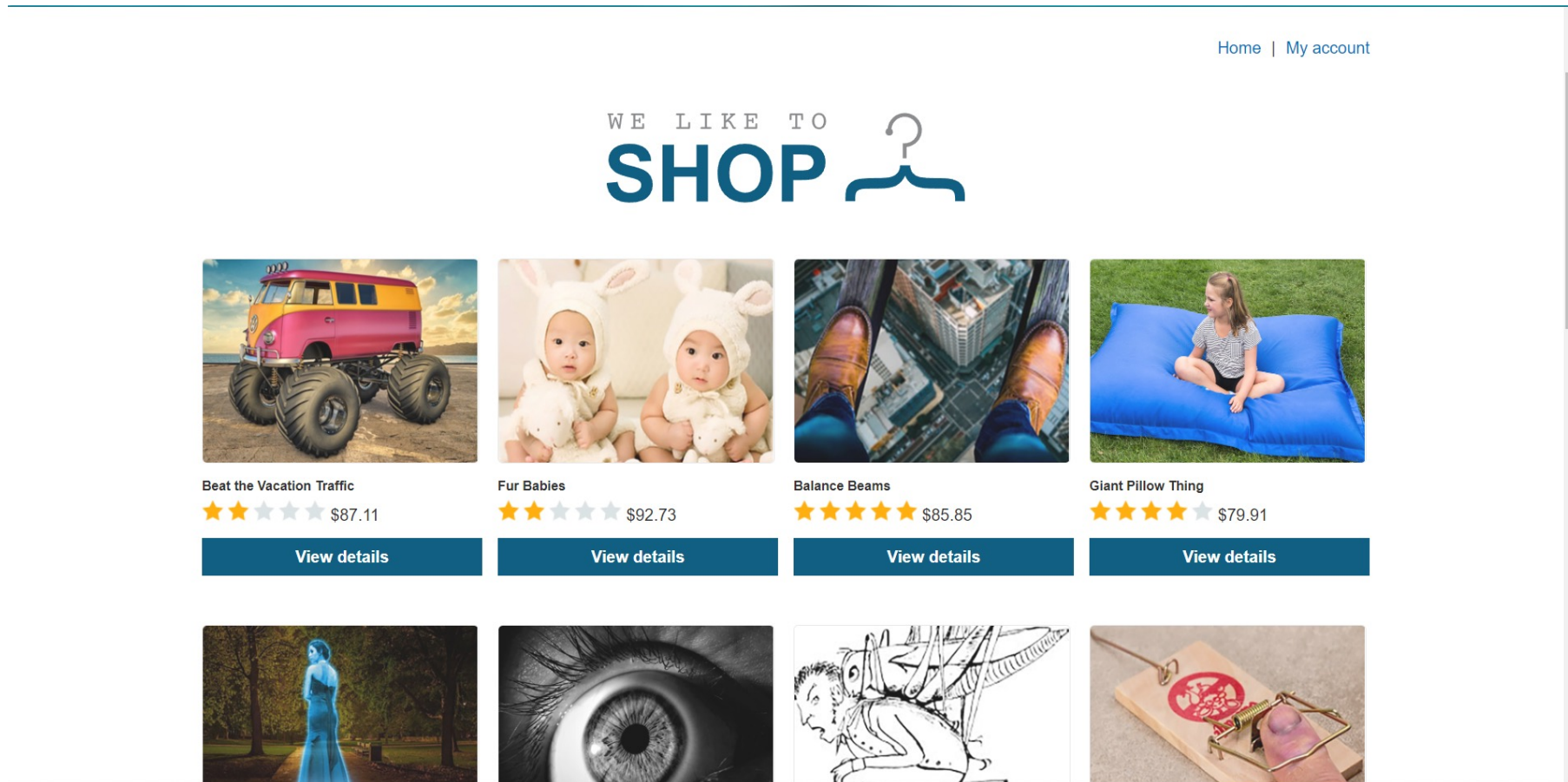
1. SQL Injection
2. XSS

Content

1. SQL Injection
2. XSS

SQL Injection

Eg: Perform an SQL injection attack that logs in to the application as the administrator user.



SQL Injection

Step 1: Open Devtools → Network to intercept http request, then try login with username is **administrator** and password is anything.

👉 We can see the browser send a http request with payload have 2 fields that we just enters.



SQL injection vulnerability allowing login bypass

[Back to lab description >>](#)

LAB Not solved

[Home](#) | [My account](#)

Login

Invalid username or password.

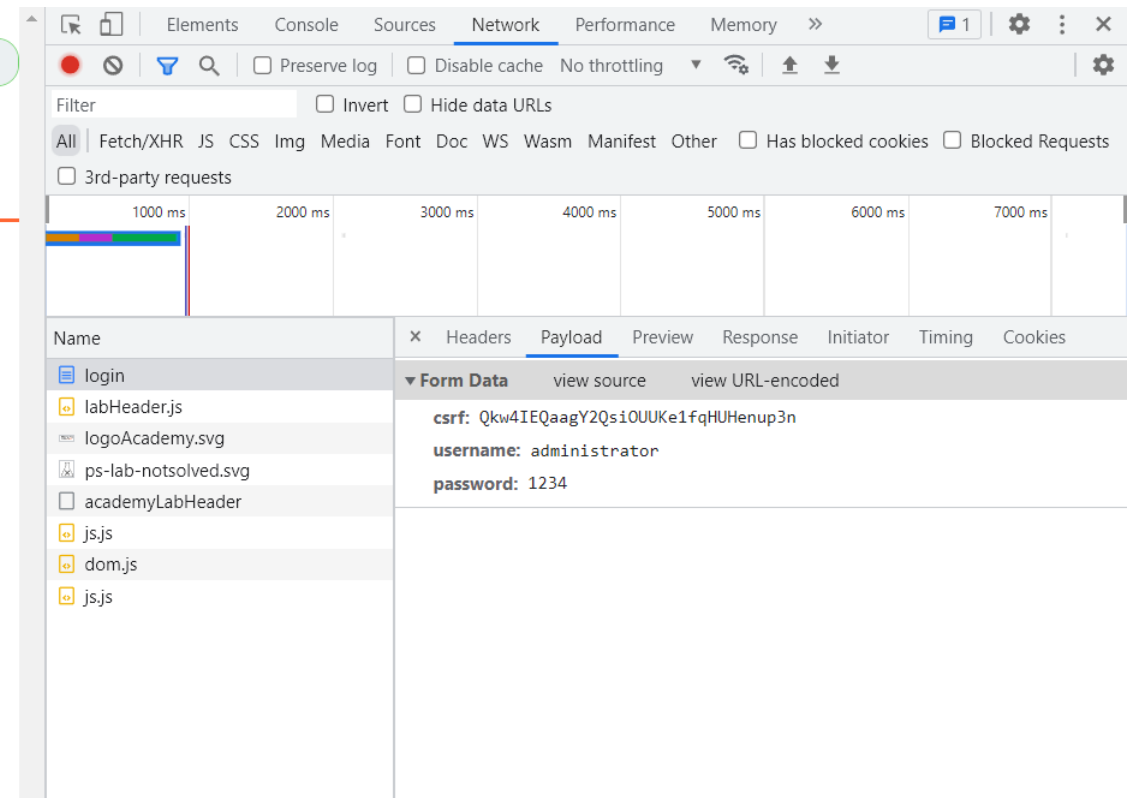
Username

administrator

Password


....

Log in



SQL Injection

Step 2: From http request, we can guess that SQL query will be like:



```
1 SELECT * FROM USER WHERE username='administrator' AND password='1234'
```

Step 3: So if we enter username with value `administrator' --` the query will be like:



```
1 SELECT * FROM USER WHERE username='administrator' -- 'and password='1234'
```



👉 In SQL, after “--” is comment so the query will select the account with username is `administrator` without any password

SQL Injection

Step 4: We can test our predictions:

Web Security Academy 

SQL injection vulnerability allowing login bypass

LAB Solved 

Back to lab description >>

Congratulations, you solved the lab!

 Share your skills!

Continue learning >>

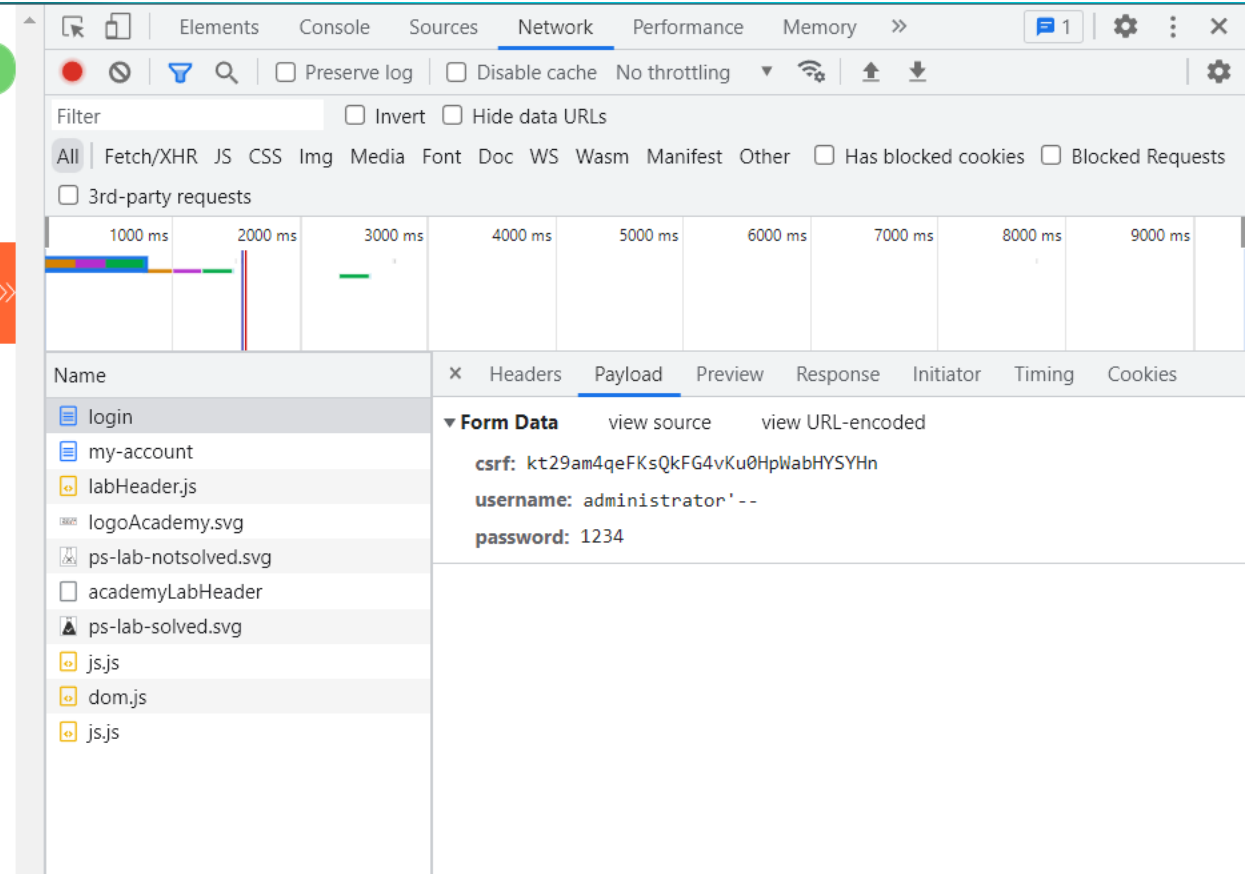
[Home](#) | [My account](#) | [Log out](#)

My Account

Your username is: administrator

Email

Update email



The screenshot shows the Network tab of a web browser. The filter is set to 'All'. The list of requests includes 'login', 'my-account', 'labHeader.js', 'logoAcademy.svg', 'ps-lab-notsolved.svg', 'academyLabHeader', 'ps-lab-solved.svg', 'js.js', 'dom.js', and 'js.js'. The 'login' request is selected, and the 'Payload' tab is active. The form data shows the following values:

Field	Value
csrf	kt29am4qeFKsQkFG4vKu0HpWabHYSYHn
username	administrator'--
password	1234

Content

1. SQL Injection
2. XSS

Eg: This sites contains a stored cross-site scripting vulnerability in the comment functionality. We will submit a comment that calls the alert function when the blog post is viewed.

Step 1: We will try to submit a comment:

Comments

 Harry Legs | 05 December 2022

How long does it take to write stuff like this?

Leave a comment

Comment:

This post is awesome

Name:

ducnm

Email:

ducnm@gmail.com

Website:

Post Comment

Step 2: After submit, we reload the post and see our new comment, we can use Devtools → Elements to see our comment in raw HTML:

The image shows a web application interface on the left and a Chrome DevTools window on the right. The web application displays a comment by 'Harry Legs' from 05 December 2022, which says 'How long does it take to write stuff like this?'. Below it is a comment by 'ducnm' from 27 December 2022, which says 'This post is awesome'. A tooltip for the 'section.comment' element shows its dimensions (668 x 71.8) and various styles. The DevTools window shows the 'Elements' panel with the raw HTML of the comment. The comment's HTML is:

<p>This post is awesome</p> == \$0

. The 'Styles' panel shows the default user agent styles for the paragraph element. A blue arrow points from the comment's raw HTML in the Elements panel to a callout box on the right. The callout box contains a red, yellow, and green window control bar and the text: 1 <p>This post is awesome</p>

Step 3: We can guess that the database store comments in raw text, so if we comment a script tag, it will trigger when any user open the post, like this:



```
1 <script> /** bad stuff... </script>
```

XSS

Step 4: We will try our predictions to comment a script which will alert 1 when any user open the post:



Leave a comment

Comment:

`<script>alert(1)</script>`

Name:

hacker

Email:

evil@gmail.com

Website:

Post Comment

👉 After submit, when back to blog post, we will see an alert



XSS

With the Devtools, we can see raw HTML:

The image shows a Chrome DevTools interface with the 'Elements' panel on the left and a web page on the right. The 'Elements' panel displays the DOM tree of a comment section. A comment with the text 'This post is awesome' is selected, and its raw HTML is shown in the 'Elements' panel. A green box highlights the closing paragraph tag `</p>`. A blue arrow points from this box to a callout window on the right. The callout window shows the raw HTML of the comment form, which includes a text input field containing the injected script `<script> alert(1) </script>`. The form also includes fields for 'Name', 'Email', and 'Website', and a 'Post Comment' button. The web page on the right shows the rendered comment, which displays the text 'This post is awesome' and the user 'hacker | 27 December 2022'.

```
<section class="comment">
  <p>...</p>
  <p>This post is awesome</p>
  <p></p>
</section>
<section class="comment">
  <p> == $0
  
  " hacker | 27 December 2022 "
  </p>
  <p>
    <script>alert(1)</script>
  </p>
  <p></p>
  </section>
  <hr>
  <section class="add-comment">...</section>
  <div class="is-linkback">...</div>
</div>
</div>
</section>
</div>
<quillbot-extension-portal>...</quillbot-extension-portal>
</body>
<quillbot-extension-root>...</quillbot-extension-root>
</html>
```

1 `<p>`
2 `<script> alert(1) </script>`
3 `</p>`

Exercise

? Lab 01: Retrieve hidden data

This lab contains an SQL injection vulnerability in the product category filter. You must perform an SQL injection attack that causes the application to display details of all products in any category, both released and unreleased.

Exercise

? Lab 02: Determine the number of columns

This lab contains an SQL injection vulnerability in the product category filter. You must determine the number of columns returned by the query by performing an SQL injection UNION attack that returns an additional row containing null values.

Exercise

? Lab 03: 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. You must perform a cross-site scripting attack that injects an attribute and calls the alert function.

Excercise

? **Lab 04:** Stored XSS into anchor href attribute with double quotes HTML-encoded

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

