Cross Site Scripting

Refer for theory: https://portswigger.net/web-security/cross-site-scripting

Vulnerability Labs(Apprentice):

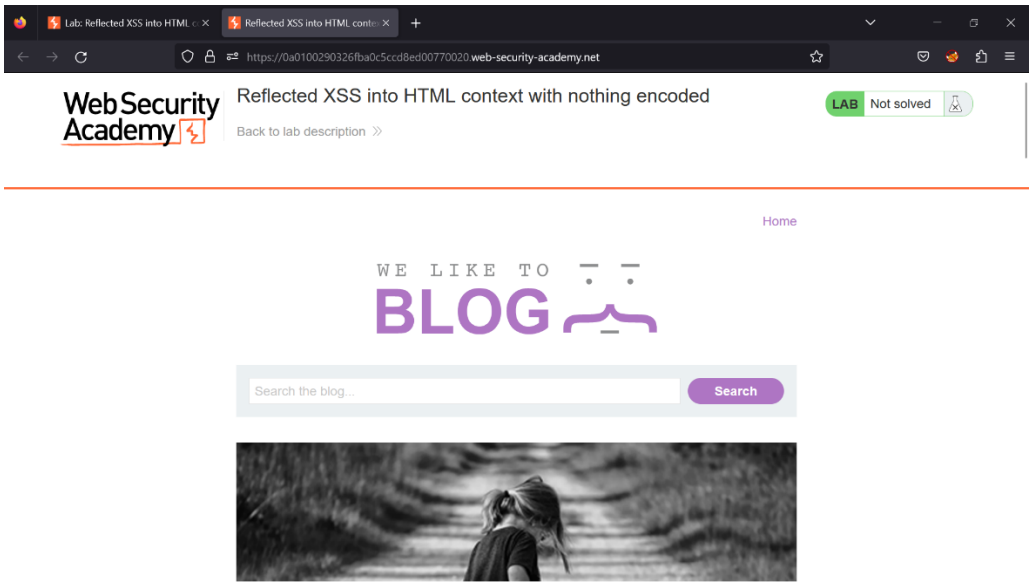
1. Reflected XSS into HTML context with nothing encoded.

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.

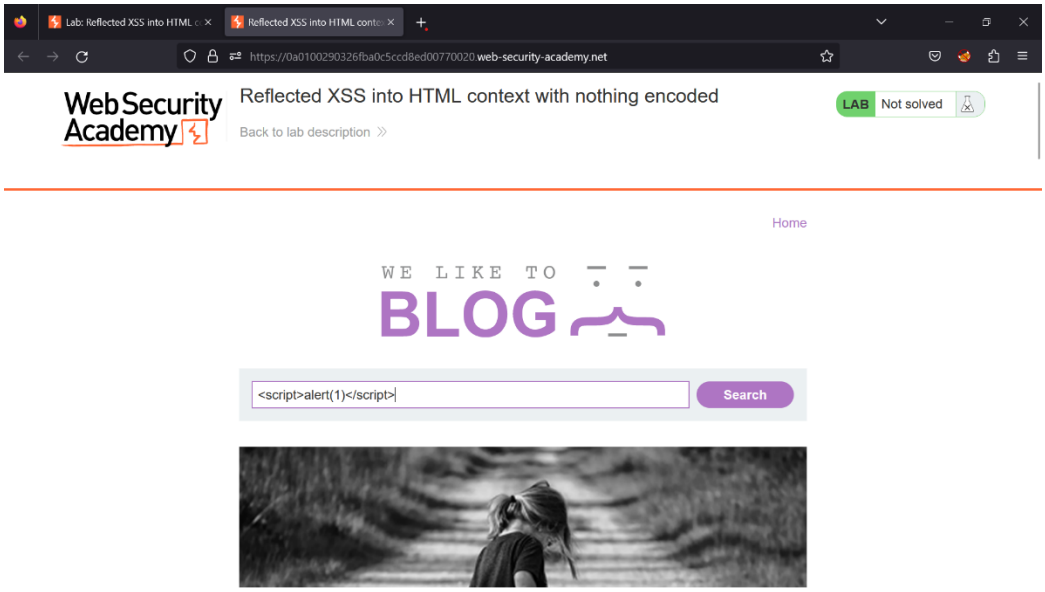
Solution:

- The programming language to do XSS on the site will be JavaScript.
- When the lab is accessed, a site related to blogs will appear.
- It will have a text field where the user can search for certain blogs.

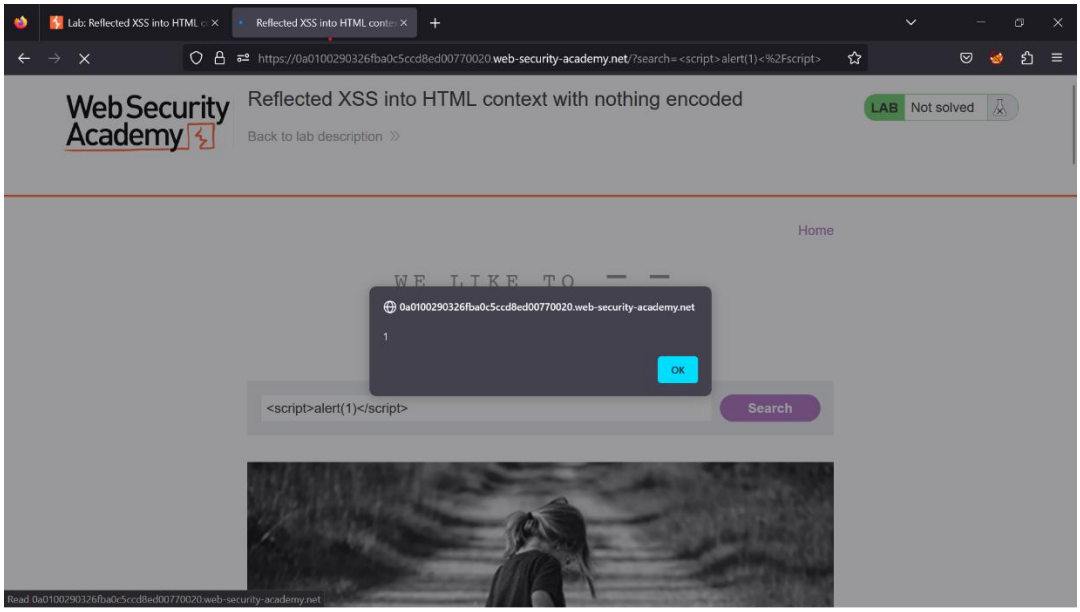


- The code snippet will be put in the search bar.
- Type the following script:

<script>alert(1)</script>



- Once entered, the script will be executed and a alert box will be displayed.



- Click ok and the lab should be solved.

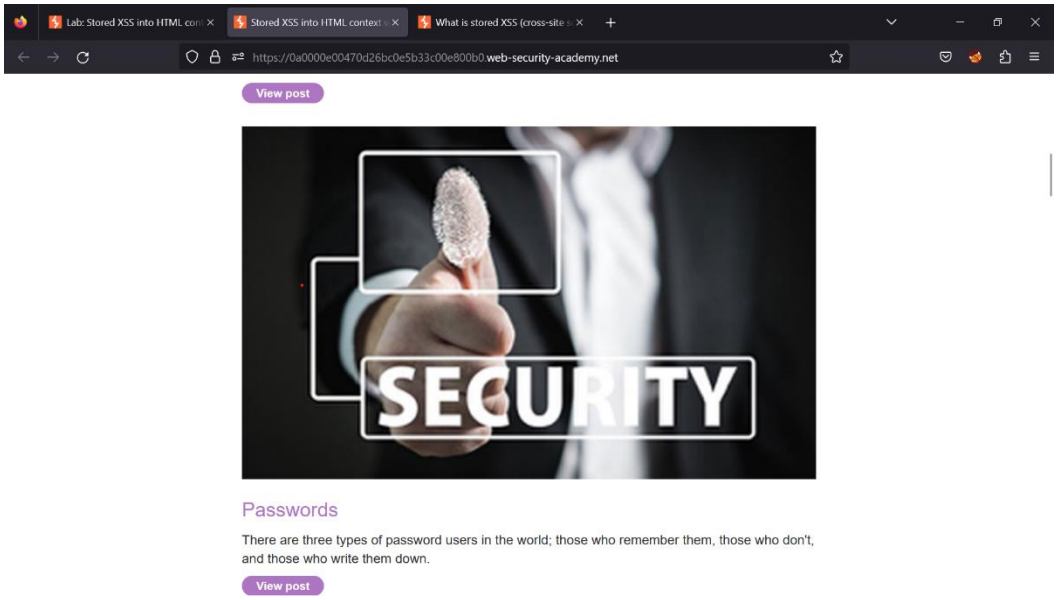
2. Stored XSS into HTML context with nothing encoded.

This lab contains a stored cross-site scripting vulnerability in the comment functionality.

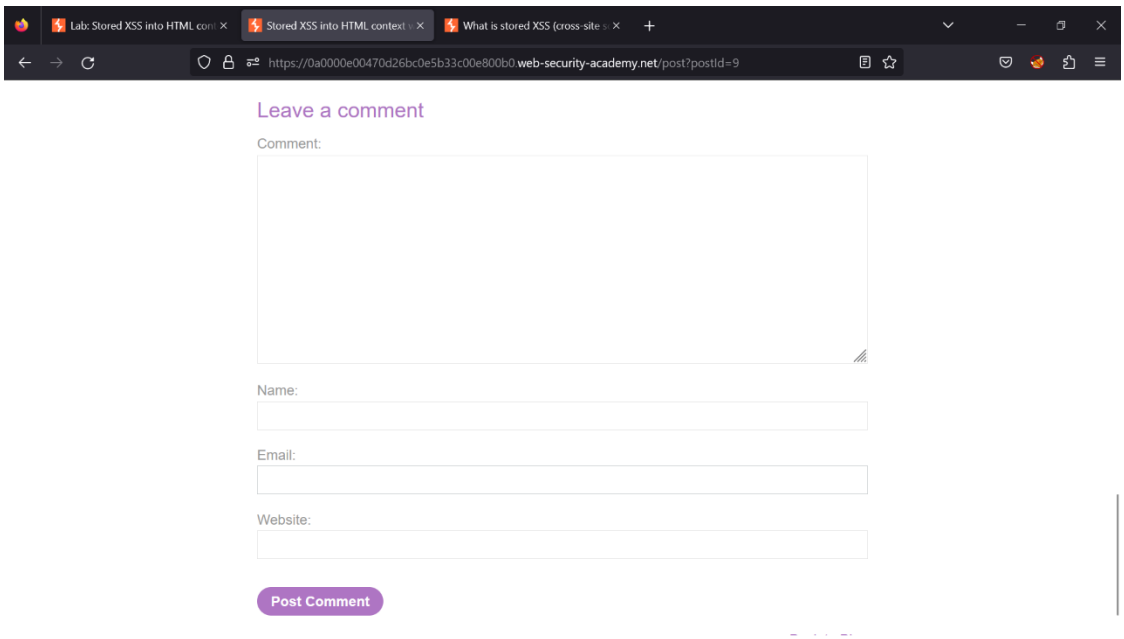
To solve this lab, submit a comment that calls the alert function when the blog post is viewed.

Solution:

- Stored cross-site scripting (also known as second-order or persistent XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.
- Once the lab is accessed a site related to blogs will apper.

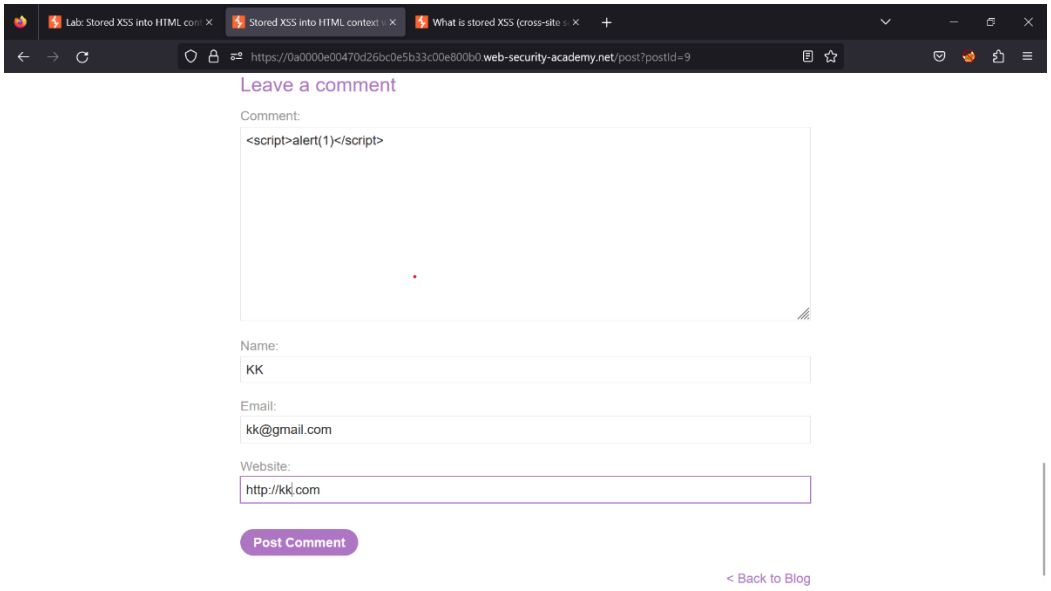


- Click on view post of any blog and go to the comment section of it.

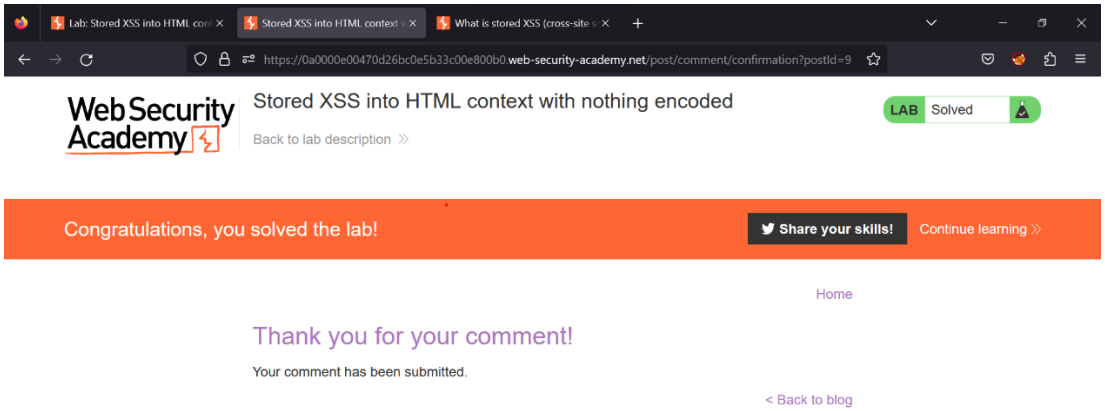


- Enter the script in the comment section. Do not leave any field empty as it's a requirement from the browser side.
- The script is:

<script>alert(1)</script>



- Once the comment is posted the necessary changes will occur in the browser and the lab will be solved.



3. DOM XSS in document.write sink using source

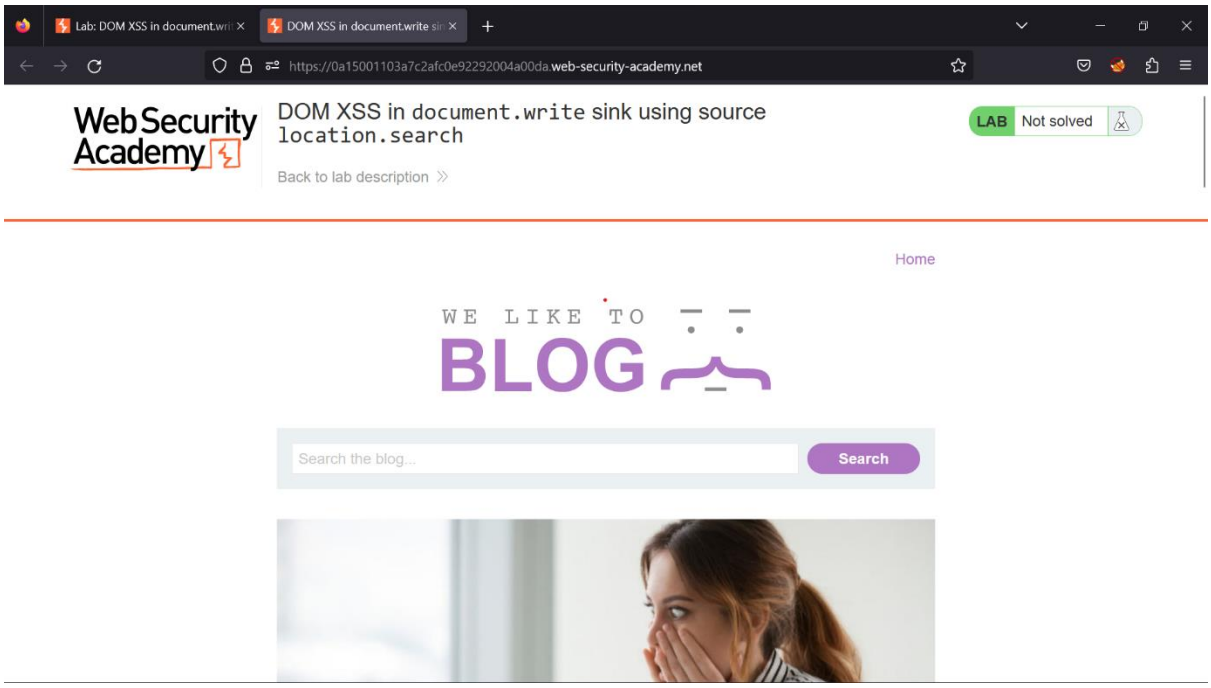
Refer for theory: https://portswigger.net/web-security/cross-site-scripting/dom-based

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.

Solution:

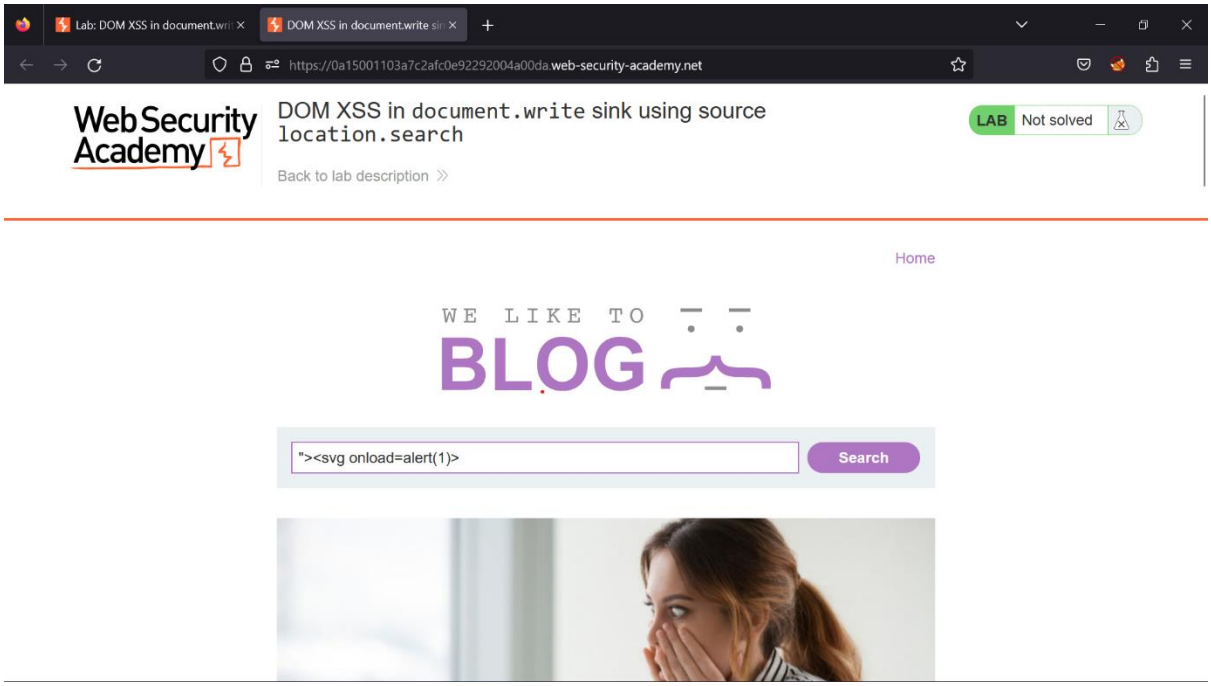
- The lab contains a website on blogging.



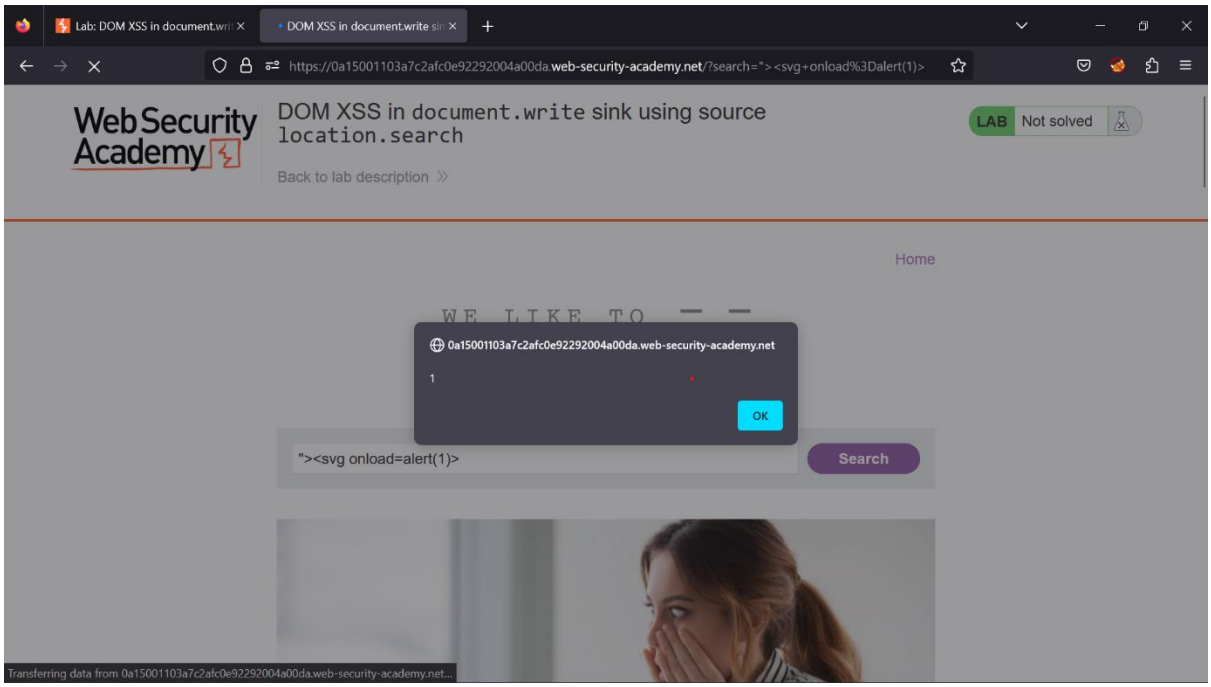
- It contains a search bar in which we will be doing the XSS attack.
- Type in the following command:

"><svg onload=alert(1)>

- The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.



- Once the search button is clicked, an alert box will appear and the lab will be solved.



The further labs are similar to the previous ones when it comes to exection. In the further part of this doc only the explaination of the commands will be done.

4. DOM XSS in innerHTML sink using source location.search

This lab contains a DOM-based cross-site scripting vulnerability in the search blog functionality. It uses an innerHTML assignment, which changes the HTML contents of a div element, using data from location.search.

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

Explanation:

- The Document Object Model (DOM) is a web browser's hierarchical representation of the elements on the page. Websites can use JavaScript to manipulate the nodes and objects of the DOM, as well as their properties.
- DOM manipulation in itself is not a problem. In fact, it is an integral part of how modern websites work. However, JavaScript that handles data insecurely can enable various attacks.
- DOM-based vulnerabilities arise when a website contains JavaScript that takes an attacker-controllable value, known as a source, and passes it into a dangerous function, known as a sink.

- To solve the lab you need to type in the following snippet:

```
<img src=1 onerror=alert(1)>
```

- The first two character "> is to escape the current html tag.
- When you reference <img src=x, this causes an error because the application is unable to find the resource x. This is intentionally done to make use of the onerror event handler.
- Prompt is similar to alert which acts as a proof of concept that the script ran.
- Try document.cookie and you should be able to see your current session cookie.

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

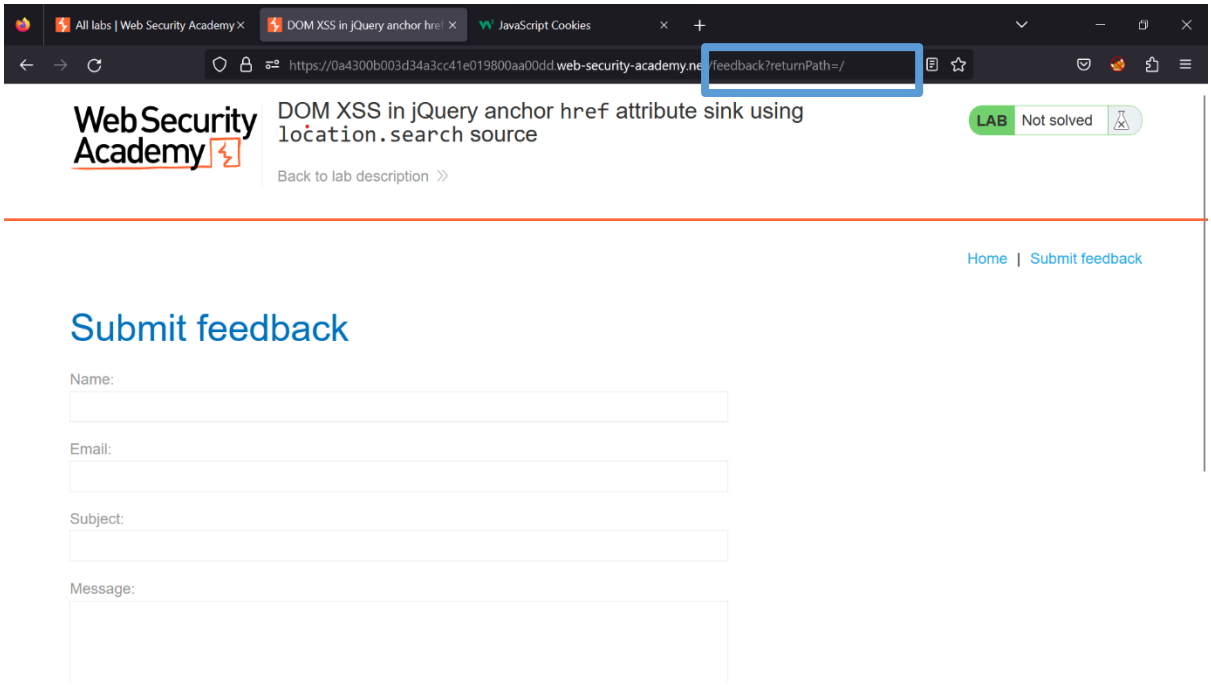
This lab contains a DOM-based cross-site scripting vulnerability in the submit feedback page. It uses the jQuery library's $ selector function to find an anchor element, and changes its href attribute using data from location.search.

To solve this lab, make the "back" link alert document.cookie.

Explanation:

- To solve the lab you need to change returnPath parameter in the url to:

javascript:alert(document.cookie)



- Cookies are data, stored in small text files, on your computer. When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user. Cookies were invented to solve the problem "how to remember information about the user":
- When a user visits a web page, his/her name can be stored in a cookie. Next time the user visits the page, the cookie "remembers" his/her name.

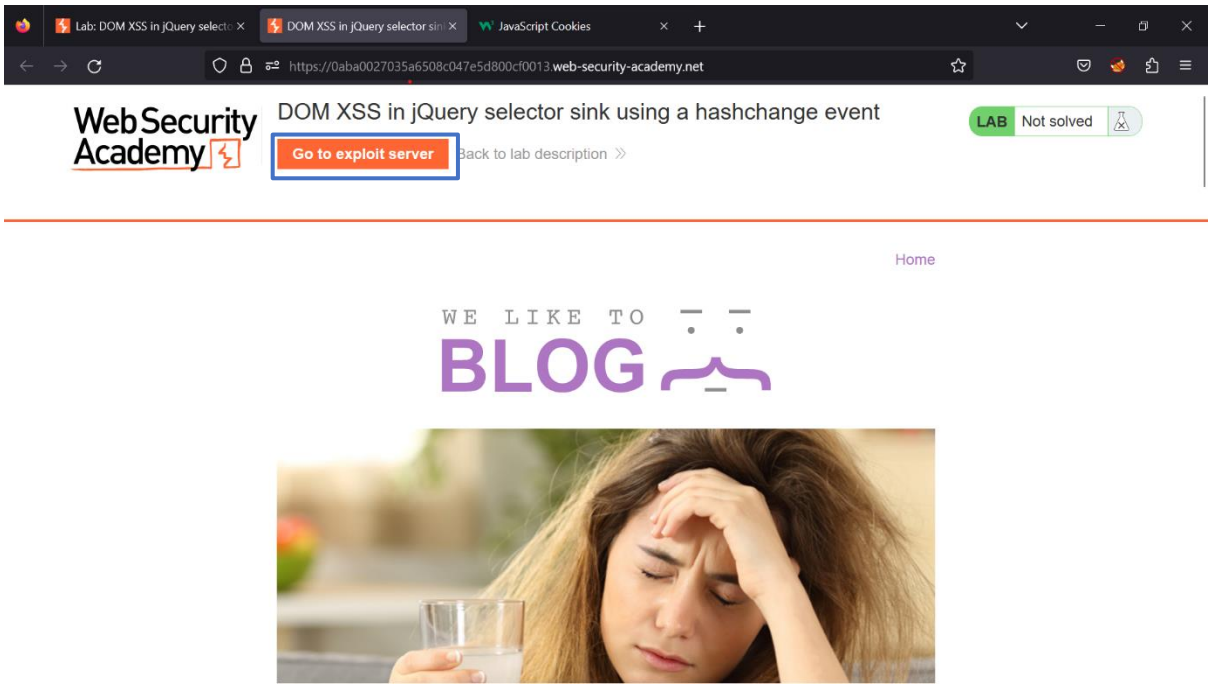
6. DOM XSS in jQuery selector sink using a hashchange event.

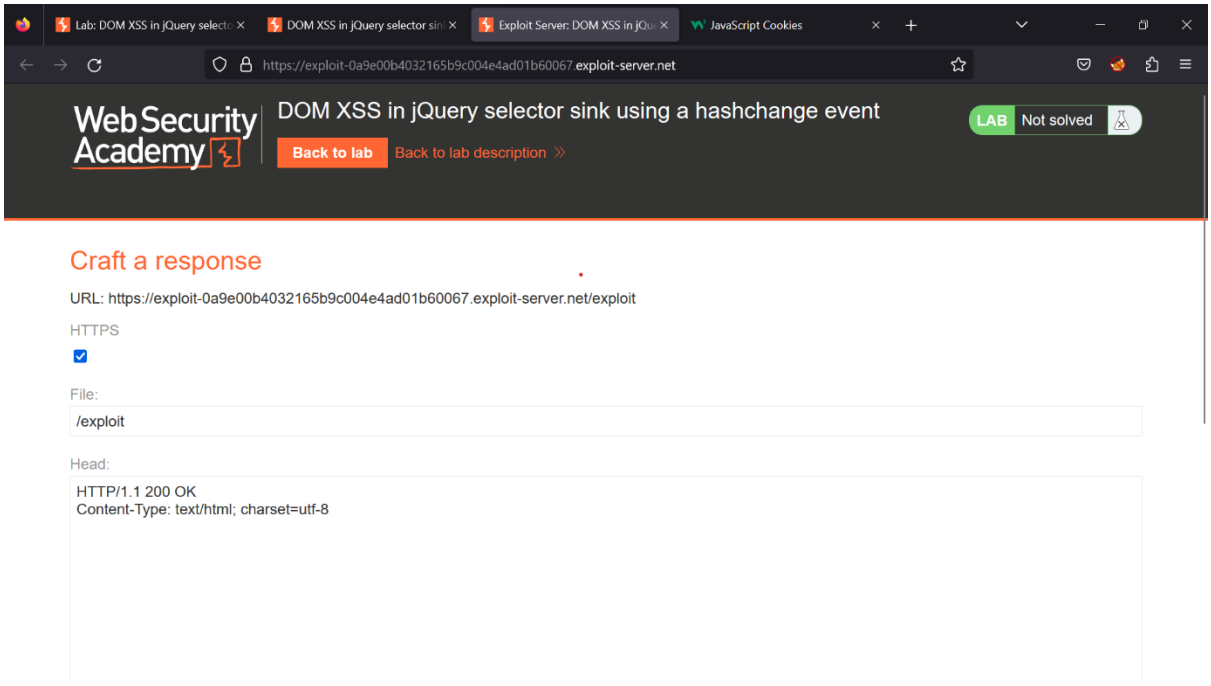
This lab contains a DOM-based cross-site scripting vulnerability on the home page. It uses jQuery's $() selector function to auto-scroll to a given post, whose title is passed via the location.hash property.

To solve the lab, deliver an exploit to the victim that calls the print() function in their browser.

Solution:

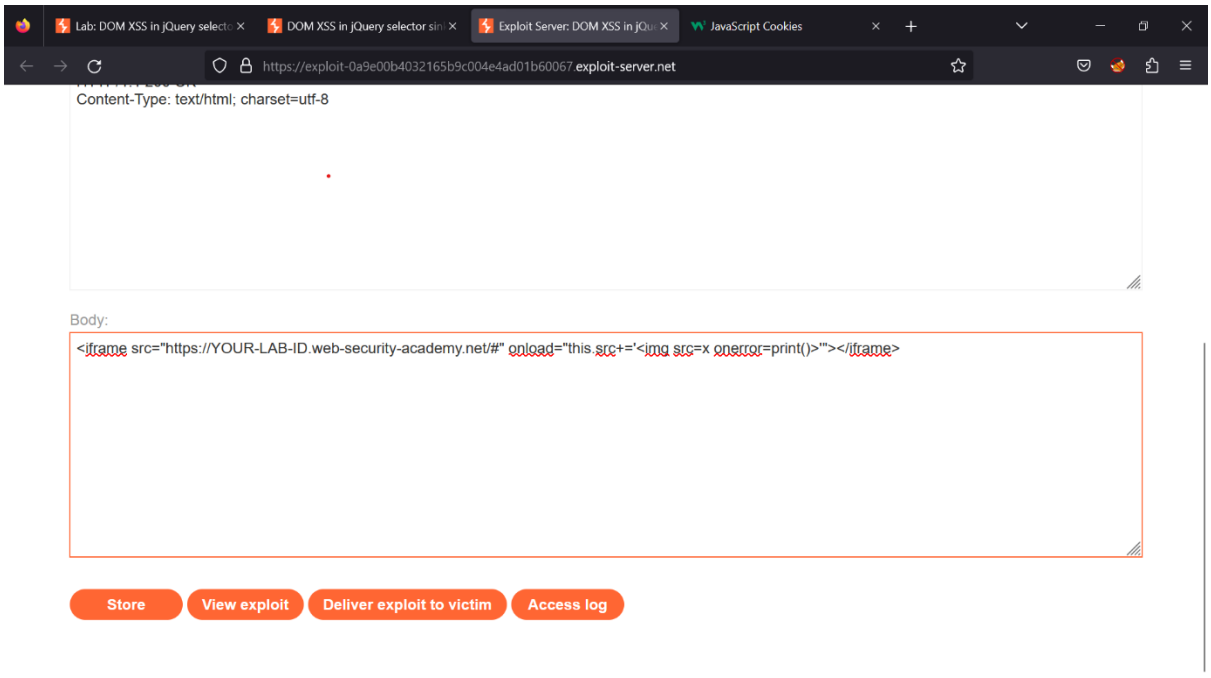
- To solve this lab we need to use a custom exploitation tool that portswigger has built for us.
- To access this tool click on the exploit server option that is provided on the top of the lab site.



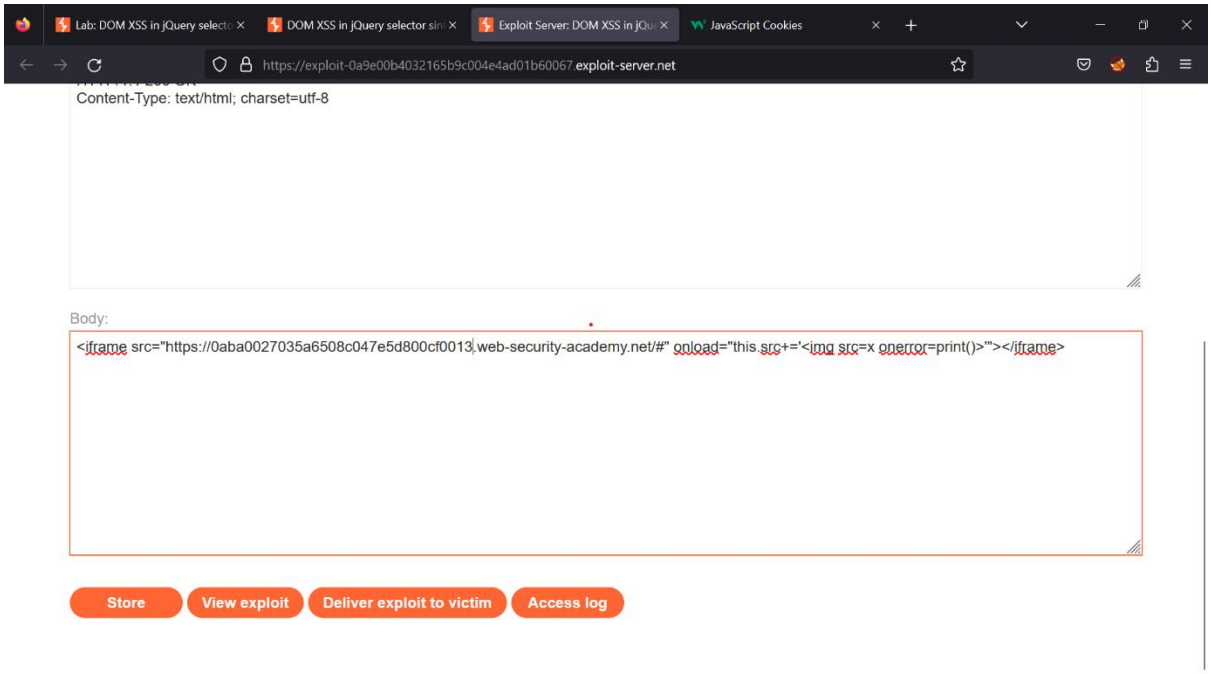


- Type in the following command in the body section of the exploit server.

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



- Edit the command by writing in your lab id into the snippet. The part where to change is highlighted in red above.



- Once the changes are made, click store. This will store the changes so that they will be shown again incase the server shuts down.
- You can click on 'View exploit' to see how the hack would work.
- Click on 'Deliver exploit to victim' to send the data to the lab and it should be solved.

7. 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.

Explanation:

- The following command needs to be input in the search bar:

"onmouseover="alert(1)

- The onmouseover event in JavaScript gets activated when the mouse cursor moves over an HTML element. This event gets activated when the mouse cursor hovers over an element. The onmouseover event does not activate when the user clicks an element, or the cursor leaves an element. For example, an onmouseover event can be used to highlight a hyperlink whenever the user hovers the mouse over it.

8. 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.

Explanation:

- Type the following command in the website field of the feedback form:

javascript:alert(1)

- This function will display text in a dialog box that pops up on the screen. Before this function can work, we must first call the showAlert() function. JavaScript functions are called in response to events.