EXPERIMENT 8

Aim -

To study and implement cross site scripting XSS vulnerability.

Theory -

1.What is Cross-Site Scripting?

Cross-site Scripting (XSS) is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application. The actual attack occurs when the victim visits the web page or web application that executes the malicious code.

A web page or web application is vulnerable to XSS if it uses unsanitized user input in the output that it generates. This user input must then be parsed by the victim's browser. XSS attacks are possible in VBScript, ActiveX, Flash, and even CSS. However, they are most common in JavaScript, primarily because JavaScript is fundamental to most browsing experiences.

Cross-site Scripting may also be used to deface a website instead of targeting the user. The attacker can use injected scripts to change the content of the website or even redirect the browser to another web page, for example, one that contains malicious code.

2.What can XSS be used for?

An attacker who exploits a cross-site scripting vulnerability is typically able to:

➔Impersonate or masquerade as the victim user.
➔Carry out any action that the user is able to perform.
➔Read any data that the user is able to access.
➔Capture the user's login credentials.
➔Perform virtual defacement of the web site.
➔Inject trojan functionality into the web site.

3.What are the types of XSS attacks?

There are three main types of XSS attacks. These are:

1.Reflected XSS: Reflected XSS is the simplest variety of cross-site scripting. It arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

2.Stored XSS: Stored XSS (also known as persistent or second-order XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

3.DOM-based XSS: DOM-based XSS (also known as DOM XSS) arises when an application contains some client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM.

4.How to Prevent XSS Attacks?

To keep yourself safe from XSS, you must sanitize your input. Your application code should never output data received as input directly to the browser without checking it for malicious code. Effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

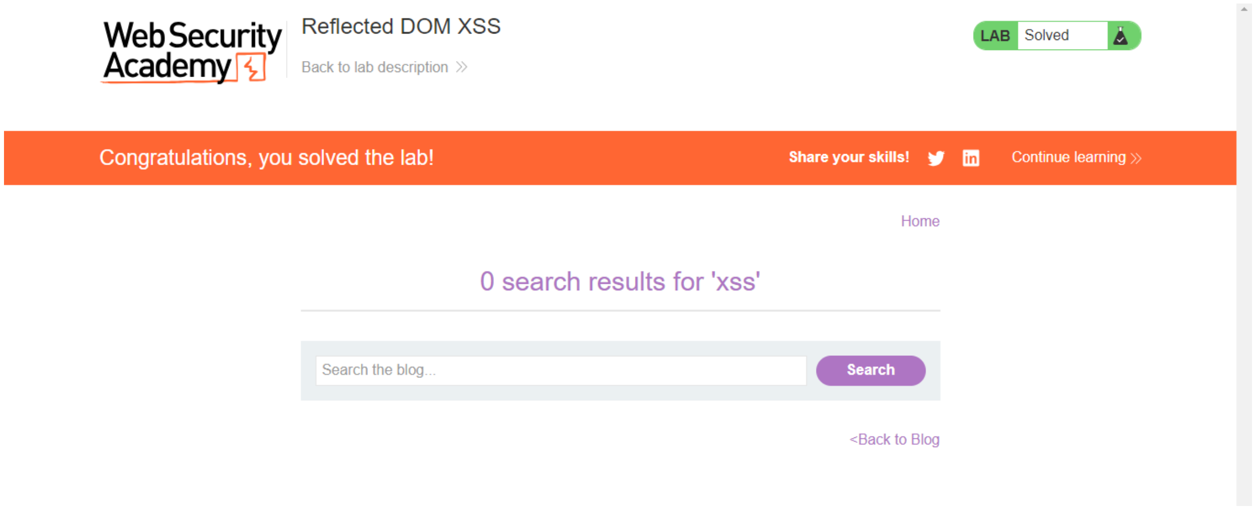
1.Filter input on arrival: At the point where user input is received, filter as strictly as possible based on what is expected or valid input.

2.Encode data on output: At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

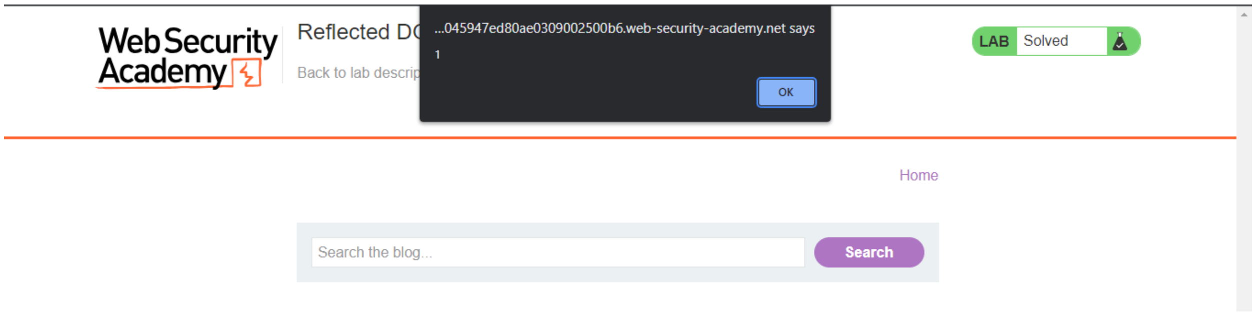
3.Use appropriate response headers: To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.

4.Content Security Policy: As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

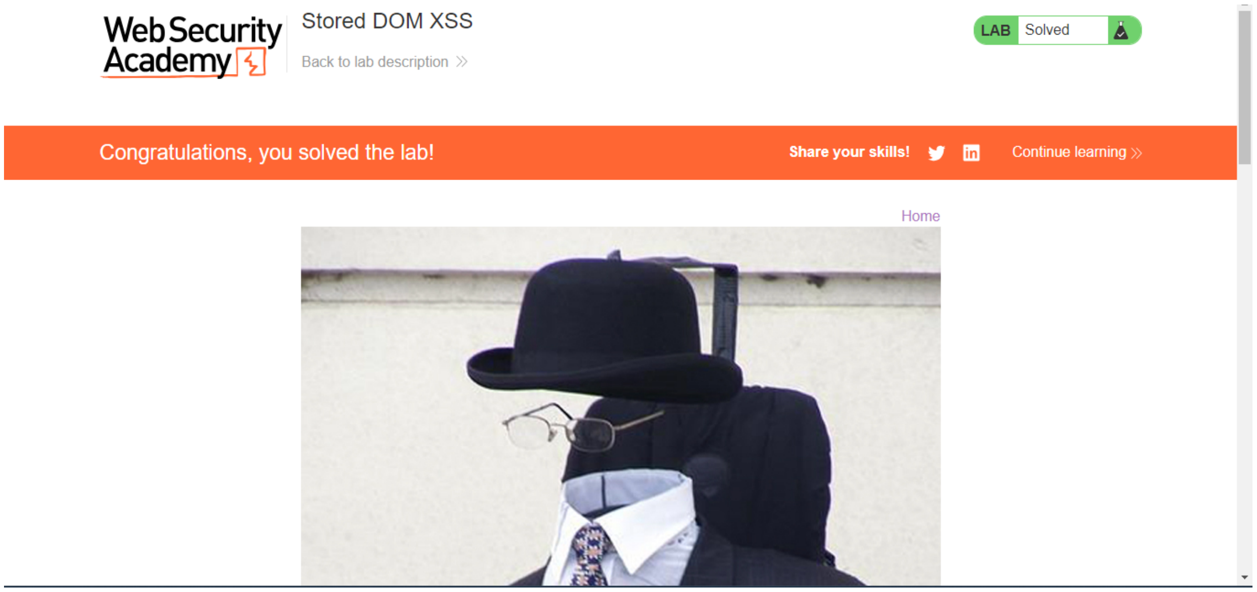
1. Reflected DOM XSS
   Go to the target website and use the search bar to search for a random test string, such as "XSS".



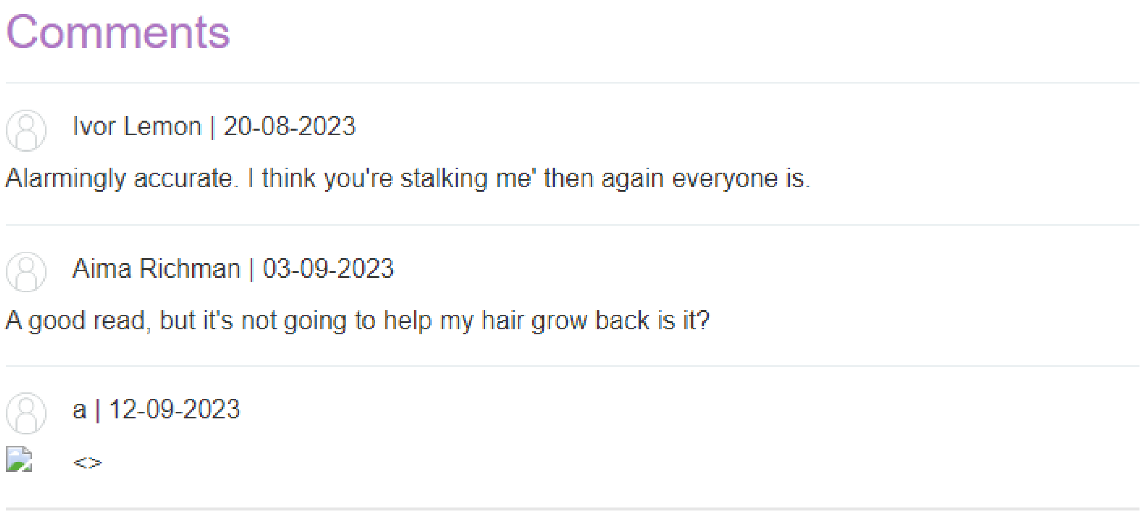
Enter the following search term: \"-alert(1)}//



2. Stored DOM XSS
   Open a blog.



Post a comment with an extra pair of angular brackets as the replace() function ius used to replace the first pair of brackets.
<><img src=1 onerror=alert(1)>



Comments

Ivor Lemon | 20-08-2023

Alarmingly accurate. I think you're stalking me' then again everyone is.

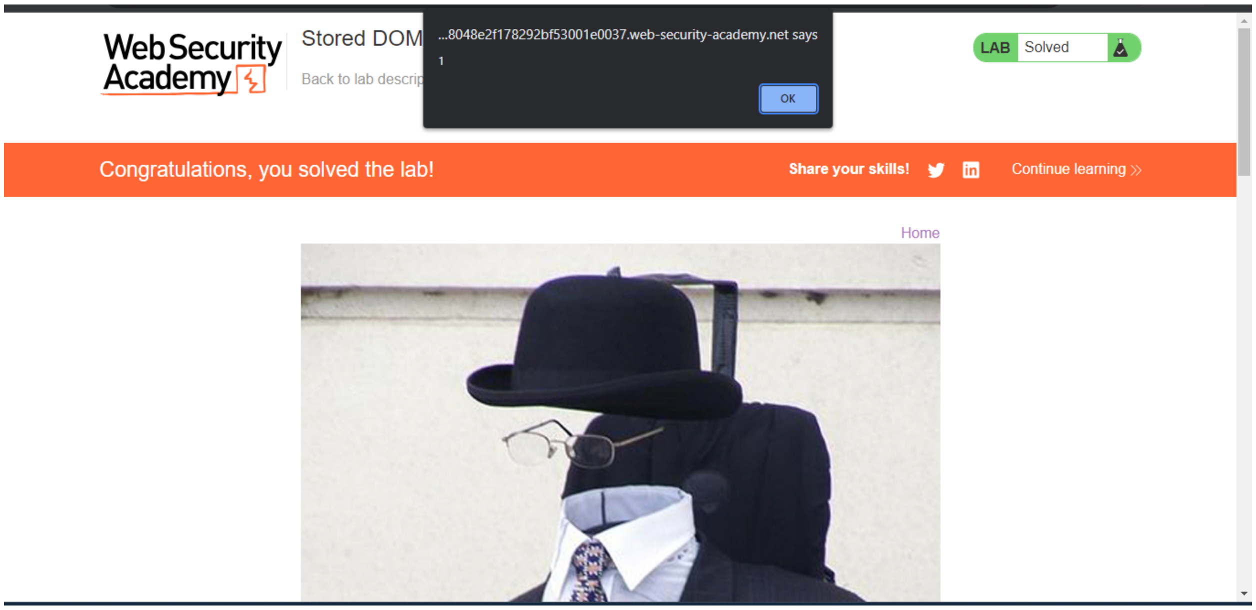
Aima Richman | 03-09-2023

A good read, but it's not going to help my hair grow back is it?

a | 12-09-2023

<>

Leave a comment



Stored DOM

Back to lab descrip

...8048e2f178292bf53001e0037.web-security-academy.net says

1

Congratulations, you solved the lab!

Conclusion-

Thus we have studied how to perform different types of Cross-Site Scripting (XSS) vulnerability.