**PortSwigger Cross-Site Scripting (XSS) Vulnerability**

**Intern id:** 195

**Lab :** Reflected XSS into HTML Context (Nothing Encoded)

**Environment :** Linux, browser(Android)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective**

Demonstrate that unsanitized user input is reflected directly into the HTML response, allowing the execution of malicious JavaScript.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Target**

Application: PortSwigger XSS Lab – Reflected XSS (HTML context)

Vector: Search query parameter

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Vulnerability Description**

The application takes user input from the search query (?search=...) and reflects it unencoded into the HTML body. An attacker can inject arbitrary HTML/JavaScript, leading to reflected Cross-Site Scripting (XSS).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Steps to Reproduce**

1. Navigate to the search function:

https://<lab-url>/?search=test

2. Observe that the value test is reflected directly in the page:

<p>Search results for 'test'</p>

3. Inject a malicious payload:

https://<lab-url>/?search=<script>alert(1)</script>

4. The server responds with:

<p>Search results for '<script>alert(1)</script>'</p>

5. When the page loads, the injected script executes, triggering:

[ALERT BOX POPS UP with "1"]

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Technical Explanation**

Since nothing is encoded, the browser interprets the injected string as raw HTML/JS.

Vulnerable code (hypothetical):

<?php

$search = $\_GET['search'];

echo "<p>Search results for '$search'</p>";

?>

The developer concatenates untrusted input into the HTML, which allows the <script> tag to run.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Impact**

Attacker can execute arbitrary JavaScript in the victim’s browser.

Possible attacks include:

Stealing session cookies.

Redirecting users to malicious sites.

Defacing the page.

Phishing.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mitigation**

HTML encode user input before inserting into the page (< → &lt;, > → &gt;).

Use secure templating engines that auto-escape content.

Apply a strict Content Security Policy (CSP).

Sanitize input with a library (e.g., DOMPurify for JS).

**\_\_\_\_\_\_\_\_\_\_\_ THANK YOU \_\_\_\_\_\_\_\_\_\_\_**