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

**Intern id:** 195

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

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

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

**Objective**

Show that user input is stored on the server and reflected into the HTML page without encoding, allowing persistent malicious JavaScript execution.

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

**Target**

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

Vector: Comment / Feedback / Blog post input field

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

**Vulnerability Description**

The application allows users to submit input (e.g., product reviews, blog comments). The submitted input is stored in the backend database and later displayed in the HTML page. Since no output encoding is applied, attackers can inject arbitrary HTML/JS.

This makes the payload persistent — every visitor to the page will execute the injected script.

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

**Steps to Reproduce**

1. Navigate to the comment form (example: product review submission).

2. Submit the following payload in the comment field:

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

3. The application accepts the comment and saves it.

4. Reload the page or navigate to the product/review section.

5. Observe that the page now includes the malicious input directly in the HTML response:

<div class="comment">

<p><script>alert('1')</script></p>

</div>

6. As soon as the page loads, the script executes, showing:

[ALERT BOX POPS UP with "1"]

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

**Technical Explanation**

The backend does not sanitize or HTML-encode the input before storing or displaying it.

Example vulnerable code (PHP style):

// Store user input in database

$comment = $\_POST['comment'];

$db->query("INSERT INTO comments (text) VALUES ('$comment')");

// Later, render comments

echo "<p>" . $row['text'] . "</p>";

Because the value is inserted raw into the HTML, the <script> tag runs in every visitor’s browser.

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

**Impact**

Stored (Persistent) XSS is more severe than reflected:

Every user visiting the page executes the payload.

Attacker can steal cookies, tokens, or sensitive data.

Malware delivery, phishing, defacement.

Full account takeover possible.

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

**Mitigation**

Encode user input before rendering (< → &lt;, > → &gt;).

Use secure frameworks or templating engines with auto-escaping.

Validate input on submission.

Sanitize stored HTML with trusted libraries (e.g., DOMPurify).

Apply Content Security Policy (CSP).

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