Experiment No. 2

Aim: To prevent Cross-Site Scripting (XSS) by implementing output escaping in HTML and JavaScript.

Learning Objective: To understand and demonstrate how output escaping mitigates XSS vulnerabilities in web applications.

Theory:

Cross-Site Scripting (XSS) is a security vulnerability where an attacker injects malicious scripts into content from otherwise trusted websites. These scripts execute in the victim's browser and can be used to steal session tokens, passwords, or other sensitive information. The primary defense against XSS is **output escaping**.

What is Output Escaping? Output escaping is the process of converting special characters in user-provided data into their safe entity equivalents. This ensures the browser interprets the data as plain text to be displayed, rather than as executable code.

Key Characters and Their HTML Entities:

- & becomes &
- < becomes <
- > becomes >
- " becomes "
- ' becomes '

Contexts for Escaping:

1. **Escaping in HTML Body:** When placing user input inside an HTML element like a , <p>, or <td>.
2. **Escaping in HTML Attributes:** When placing user input inside an attribute like value, href, or src.
3. **Escaping in JavaScript:** When inserting user data into JavaScript code, especially within strings.

Procedure / Implementation:

This experiment has two parts: first, demonstrating the XSS vulnerability, and second, applying the fix using output escaping.

In this step, we will inject a malicious script using innerHTML, which does not perform any escaping.

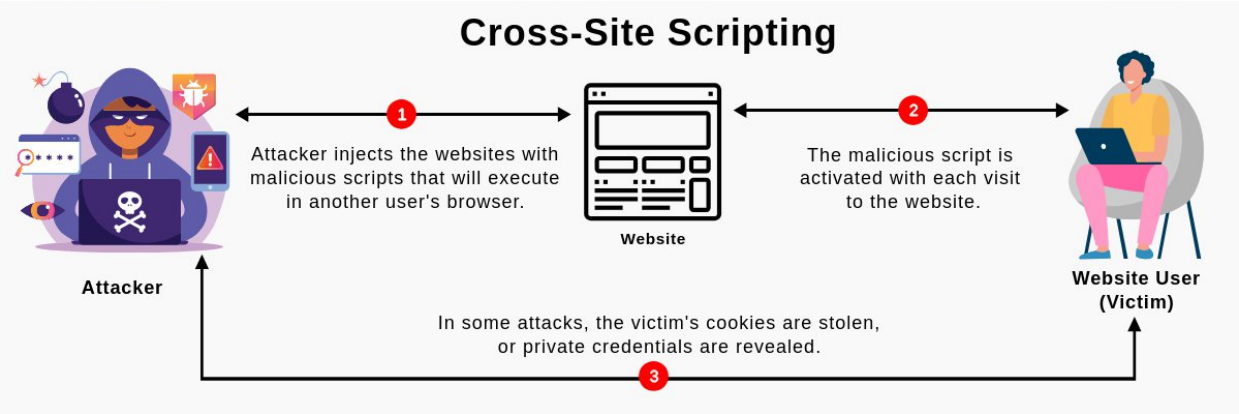
1. Create a new HTML file named vulnerable.html.
2. Add the following code. This code simulates taking user input and displaying it on the page.

```
<!DOCTYPE html>
<html>
<head>
  <title>Vulnerable Page</title>
</head>
<body>
  <h1>User Comment Section (Vulnerable)</h1>
  <div id="comment-output"></div>

  <script>
        // Malicious input simulating an attacker's comment
        const userInput = '<img src=x onerror=alert("XSS Attack!")>';

        // VULNERABLE: Using innerHTML allows the script to execute
        document.getElementById('comment-output').innerHTML = userInput;
  </script>
</body>
</html>
```

3. Open vulnerable.html in a web browser. You will see an alert box pop up, which demonstrates that the JavaScript in the userInput string was executed.



Part 2: Preventing XSS with Output Escaping

Here, we will fix the vulnerability by using a safe property, textContent, which automatically escapes the content.

1. Create a new HTML file named secure.html.
2. Add the following code. The only change is replacing .innerHTML with .textContent.

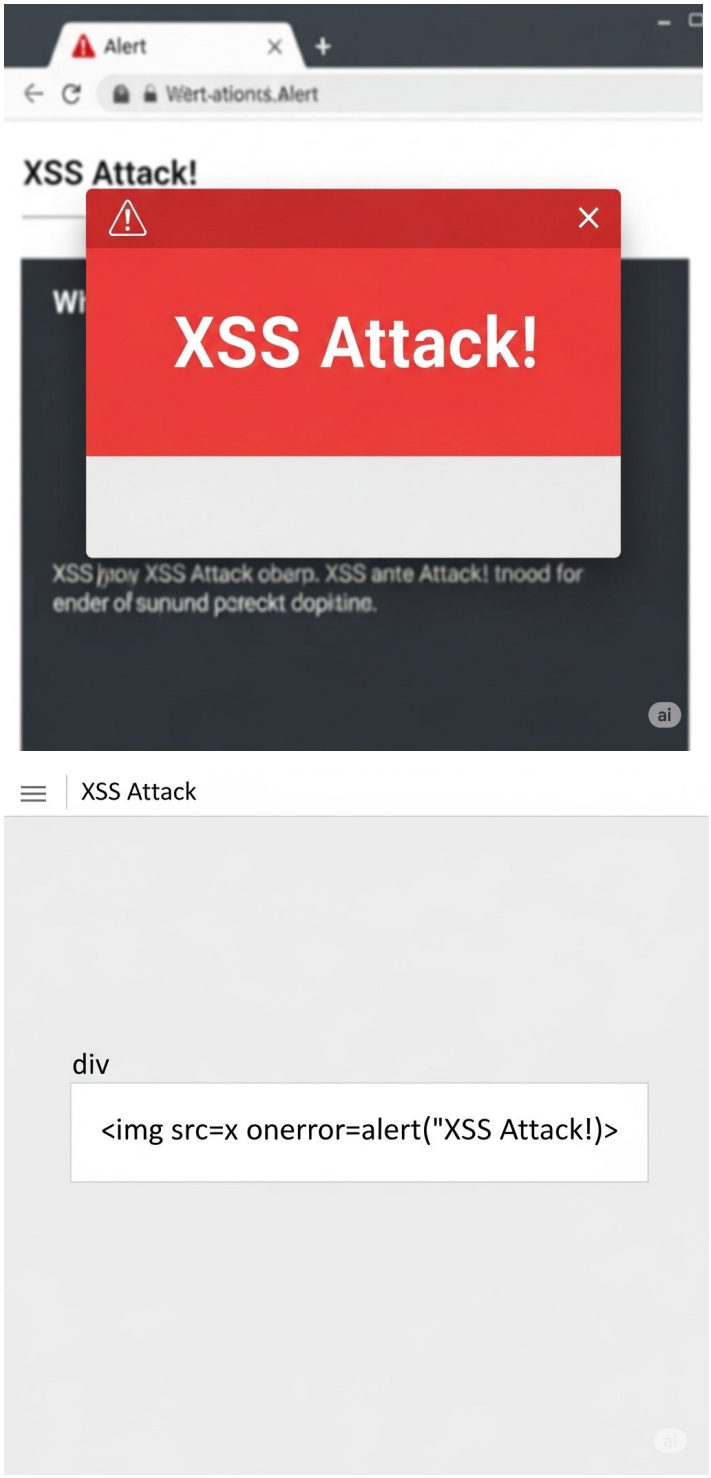
```
<!DOCTYPE html>
<html>
<head>
  <title>Secure Page</title>
</head>
<body>
  <h1>User Comment Section (Secure)</h1>
  <div id="comment-output"></div>

  <script>
        // Malicious input simulating an attacker's comment
        const userInput = '<img src=x onerror=alert("XSS Attack!")>';

        // SECURE: Using textContent treats the input as plain text
        document.getElementById('comment-output').textContent = userInput;
  </script>
</body>
</html>
```

3. Open secure.html in a web browser. This time, no alert box will appear. Instead, the malicious string <img src=x onerror=alert("XSS Attack!")> will be safely displayed as plain text on the page.

Screenshot:



Learning Outcome: Performed an experiment to demonstrate how untrusted user input can cause

an XSS vulnerability when rendered with innerHTML. Successfully prevented the XSS attack by implementing output escaping using the textContent property, which correctly displays user input as harmless text.

Conclusion: This experiment confirms that output escaping is a fundamental and critical security practice for developing secure web applications. By treating all user input as untrusted data and encoding it appropriately for the output context (HTML body, attributes, JavaScript), we can effectively mitigate the risk of Cross-Site Scripting attacks. Never trust user input and always escape output are the key principles for preventing XSS.

Name:
Class: BE-CSE
Roll No.:

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				