Experiment No. 6

Aim: Detecting and Exploiting Reflected, Stored, and DOM-Based XSS in Web Applications.

Learning Objective:

- Understand the theory behind Reflected, Stored, and DOM-based XSS vulnerabilities in web applications.
- Gain hands-on experience in detecting and exploiting these types of XSS.
- Learn secure coding and validation practices to prevent these common vulnerabilities.

Theory:

Introduction to XSS

Cross-Site Scripting (XSS) is a vulnerability wherein an attacker injects malicious scripts into web pages viewed by other users. The main types are:

- Reflected XSS: Payload is reflected off the web server in an immediate response.
- Stored XSS: Malicious input is permanently stored and served to users later.
- DOM-based XSS: The vulnerability and execution both occur within the client-side JavaScript and DOM.

XSS attacks can lead to session hijacking, defacement, data theft, and account compromise.

Reflected XSS

1. About:
Occurs when user input (e.g., form fields, URL parameters) is immediately echoed back by the server in HTTP responses without proper sanitization.

2. Steps to Detect:

- Find user input fields and URL parameters.
- Craft test payloads, e.g., <script>alert('XSS')</script>, and inject them into parameters.
- Observe if the script executes or is reflected unsanitized on the response page.
- Use automated tools like Burp Suite, OWASP ZAP, or XSStrike to facilitate detection.

3. Steps to Exploit:

- Send a link containing the XSS payload to a victim.

- If the victim clicks, the malicious script executes in their browser context, potentially stealing cookies, data, or performing actions on their behalf.

Stored XSS

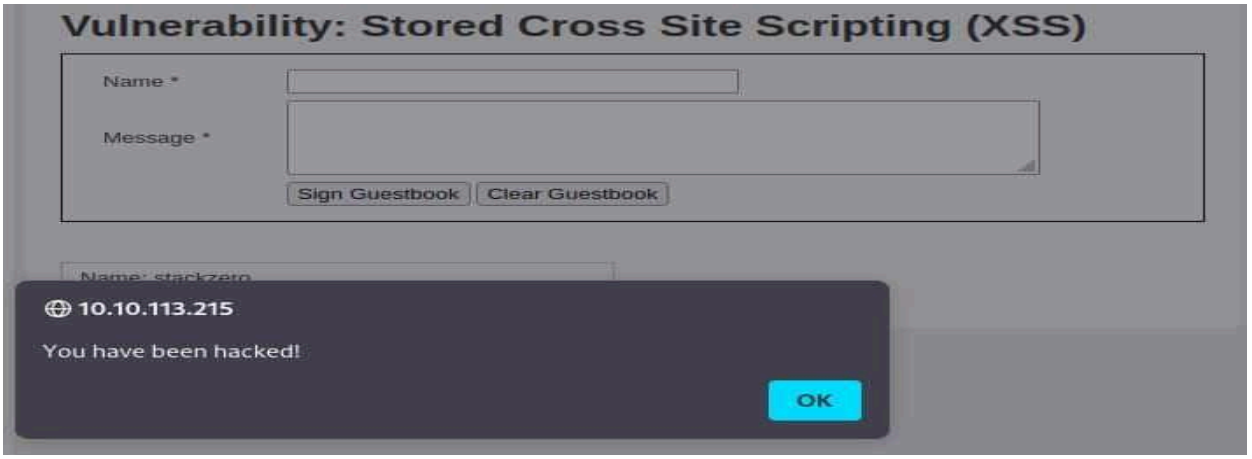
1. About:
Malicious scripts are injected into data that the application stores (e.g., comment sections, forums) and are then displayed to other users later. This is also called persistent or second-order XSS.

2. Steps to Detect:

- Locate input mechanisms that result in content being stored by the application (e.g., posting comments, feedback forms).
- Submit test payloads into these fields.
- Access (or wait for) the output area where the data is displayed. If the payload executes, the site is vulnerable.
- Automated tools like XSS Hunter can help detect stored XSS, especially blind variants.

3. Steps to Exploit:

- Inject persistent script payload through an available input channel.
- Any user who views the affected content triggers the script, compromising their session or exfiltrating sensitive data.



DOM-Based XSS

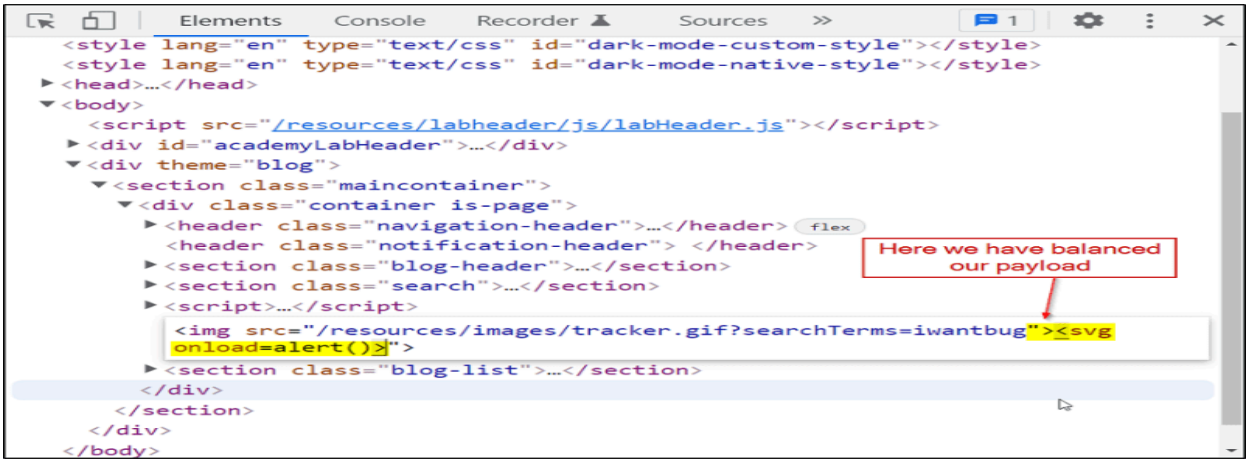
1. About:
Occurs when vulnerabilities exist in client-side scripts that dynamically write user input to the DOM without proper validation or sanitization—even if the server is secure.

2. Steps to Detect:

- Review JavaScript code or dynamic DOM manipulations (e.g., using innerHTML).
- Pass payloads, such as <img src=x onerror=alert(1)>, via URL fragments or parameters that JavaScript processes.
- Observe client-side code execution rather than server response output.
- XSS Hunter and browser-based dev tools can be used for deeper analysis of DOM behaviors.

3. Steps to Exploit:

- Craft a URL or input that manipulates the DOM through client-side scripts to insert the payload.
- The attack is successful if visiting the link executes the injected JavaScript in the victim's browser—with full access to that session's context.



Learning Outcome:

- Identify the theory, mechanisms, and practical differences between reflected, stored, and DOM-based XSS vulnerabilities.
- Develop skills in manual and automated techniques to detect and exploit each type of XSS.
- Recommend secure coding and validation strategies to mitigate and prevent XSS in future development.

Conclusion:

……

……

……

……

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				