**SAD Lab Exp-8**

**Shashwat Tripathi**

**D20A 64**

**Aim:** To study and implement cross site scripting, vulnerability lack

**Theory:**

**What is XSS?**

Cross-Site Scripting (XSS) is a security vulnerability found in web applications that allows attackers to inject malicious scripts into content that is viewed by other users. The core issue lies in the way web applications handle and display user input without proper validation or escaping.

**How XSS Works**

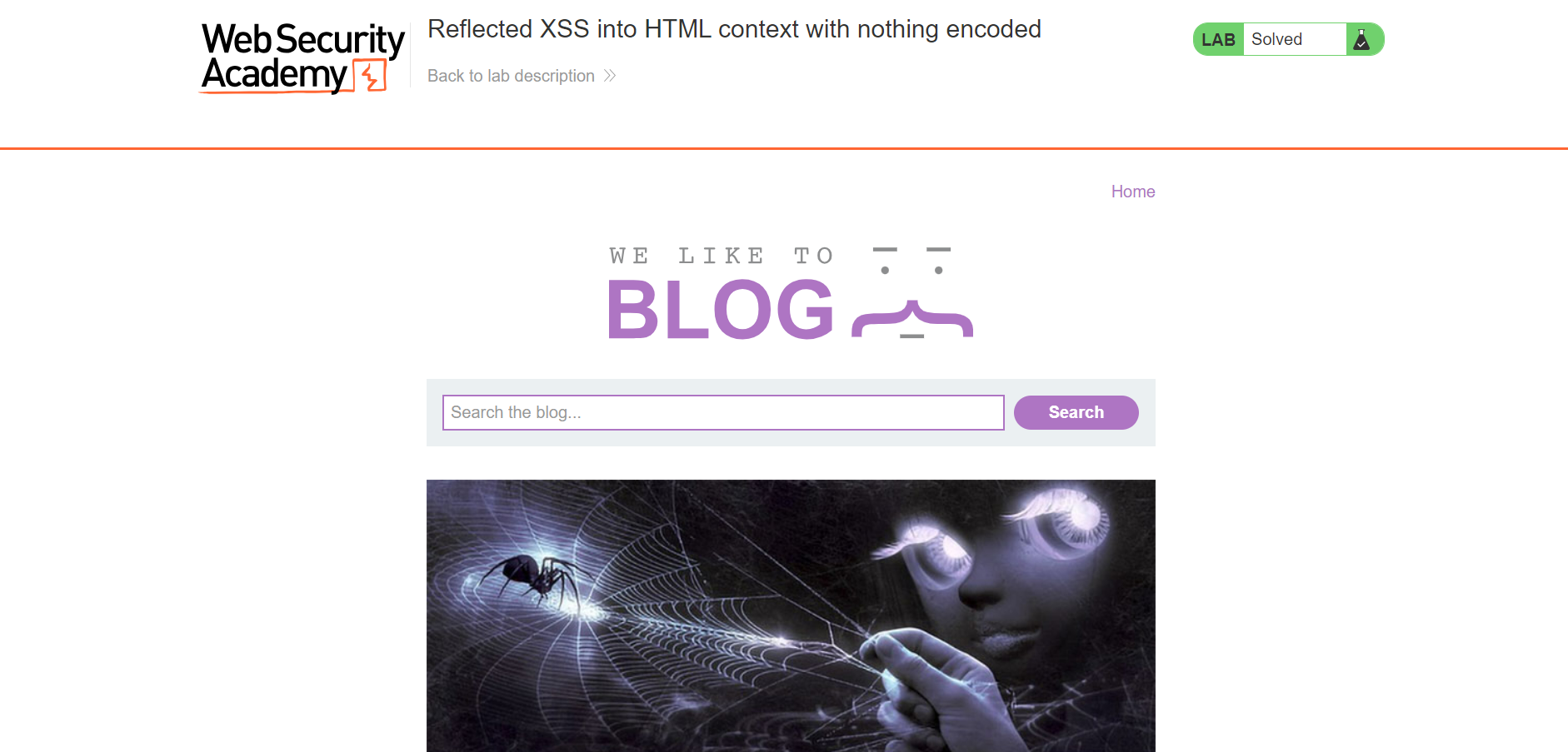
1. **Injection**: An attacker injects a script (often JavaScript) into a web page. This can happen through various input fields, such as comment sections, user profiles, or even URLs.
2. **Execution**: When another user views the affected page, the injected script runs in their browser as if it were part of the legitimate page. This can lead to various harmful actions.

**How to prevent XSS attacks?**

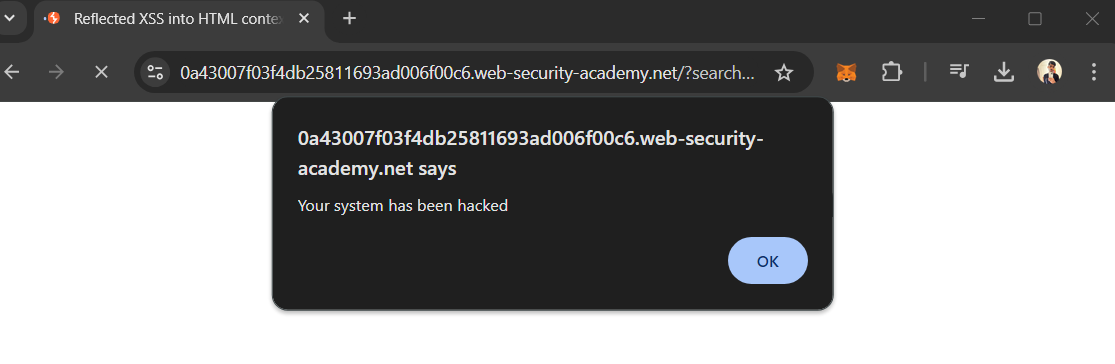
1. **Input Validation**: Always validate and sanitize user input to ensure it doesn't contain harmful scripts.
2. **Output Encoding**: Encode output to ensure that any injected scripts are rendered as plain text instead of executable code.
3. **Content Security Policy (CSP)**: Implement CSP headers to restrict the sources from which scripts can be loaded.
4. **Use Security Libraries**: Employ libraries and frameworks that provide built-in protection against XSS.
5. **Regular Security Audits**: Conduct regular testing and audits to identify and remediate vulnerabilities.

**Implementation of XSS:**

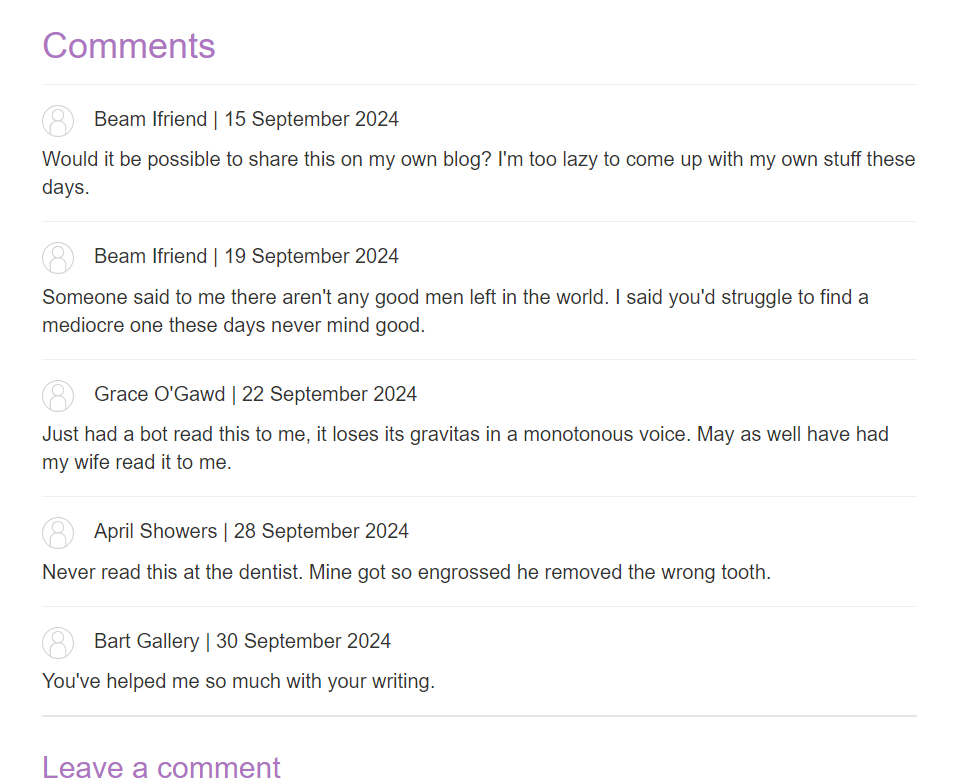
1. **Reflected XSS**: The script is reflected off a web server, often via a URL parameter. It executes immediately upon the user clicking a specially crafted link.

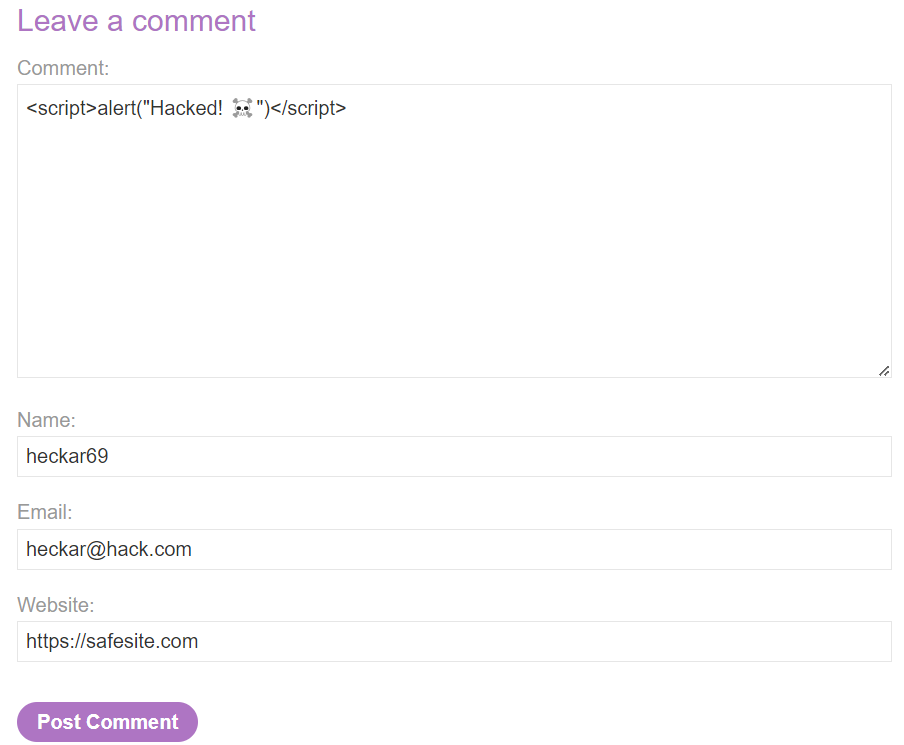
****

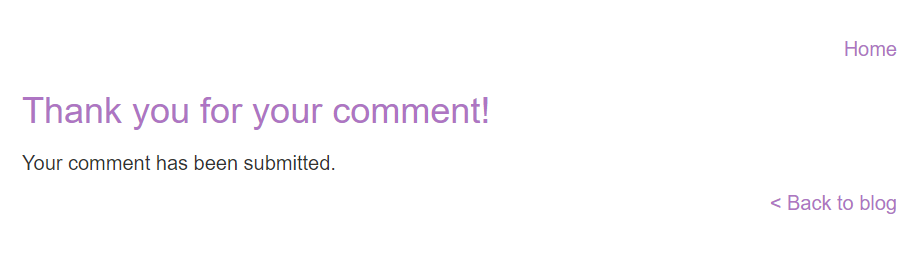
****

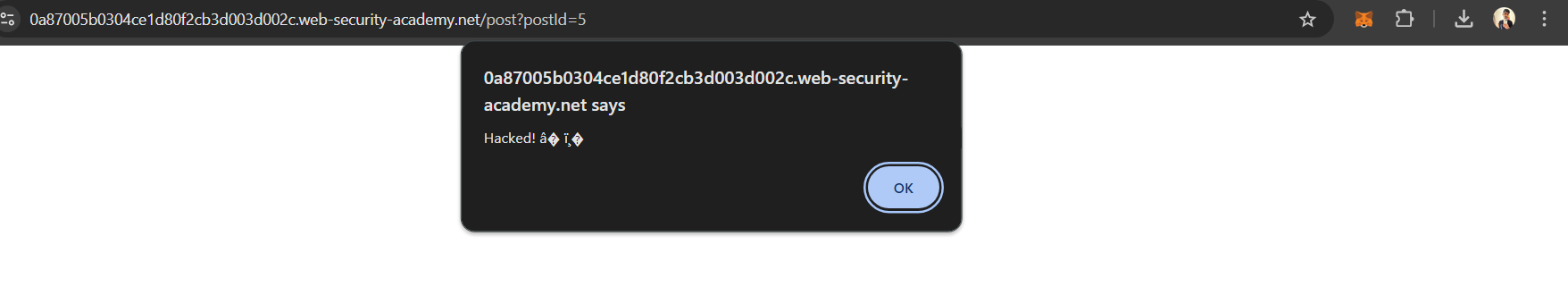
****

1. **Stored XSS**: The malicious script is stored on the server (e.g., in a database) and served to users when they visit the affected page.

****

****

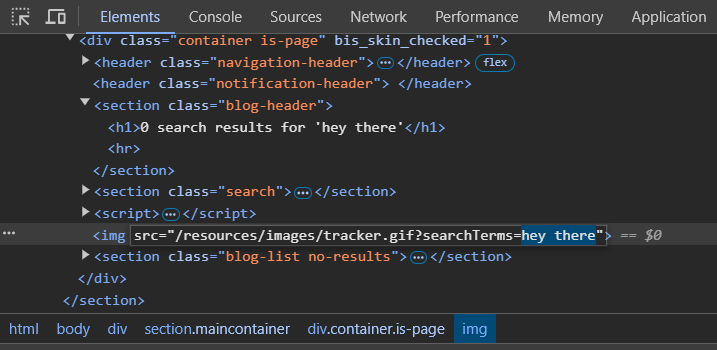
****

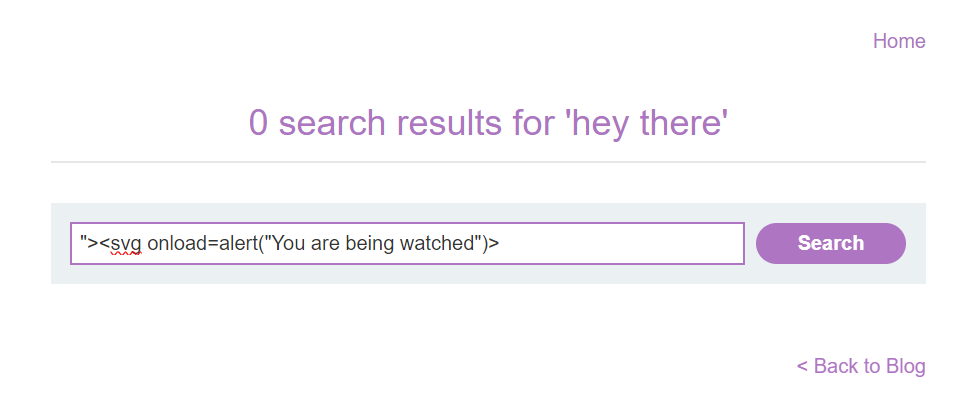
****

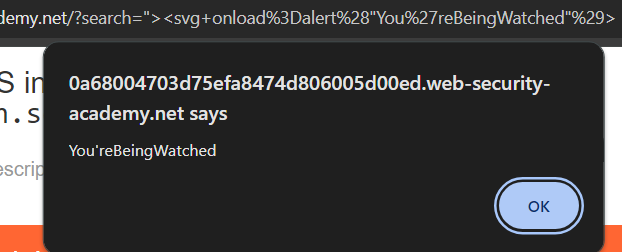
****

1. **DOM-based XSS**: The vulnerability exists in the client-side code (JavaScript) rather than on the server. The attack modifies the DOM (Document Object Model) in the user’s browser.

****

****

****

****

### 

### 

### **Conclusion:** In this experiment, we demonstrated how cross-site scripting (XSS) vulnerabilities can be exploited due to inadequate input validation and output encoding, leading to potential security risks for users.