## Introduction

Cross-Site Scripting (XSS) is a type of web security vulnerability that allows attackers to inject malicious scripts into websites viewed by other users. These attacks are possible when a web application includes untrusted data in the response it sends to a user’s browser without proper validation or escaping.  
  
I explored:  
  
Reflected XSS  
Cookie theft via XSS  
Stored XSS  
Additional XSS payloads (Redirection & Fake Login Form)

### Task 1: Reflected XSS

Tested URL: http://localhost/DVWA/vulnerabilities/xss\_r/

**Payload Used:**

<script>alert('XSS Test')</script>

Observation: A JavaScript alert popped up displaying “XSS Test”, confirming the vulnerability.

### Task 2: Stealing Cookie via XSS

**Payload Used:**

<script>document.write(document.cookie)</script>

Observation: The browser displayed the current session cookie (e.g., PHPSESSID=...), proving that JavaScript had access to sensitive data.

### Task 3: Stored XSS

Tested URL: http://localhost/DVWA/vulnerabilities/xss\_s/

**Payload Used:**

<script>alert('Stored XSS')</script>

Observation: After submitting the comment and refreshing the page, the alert was triggered every time, affecting all users who visited the page.

### Extra Payload 1: Redirection

Payload Used:

<script>window.location='https://example.com'</script>

Observation: The browser was redirected to the external website. This can be used in phishing attacks.

### Extra Payload 2: Fake Login Form

Payload Used:

<script>  
document.write('<form><input type="text" placeholder="Username"><br><input type="password" placeholder="Password"><br><button>Login</button></form>');  
</script>

Observation: A fake login form appeared on the page, which could potentially be used to collect credentials.

## Lessons Learned

* Web applications are vulnerable to XSS if user input is not properly sanitized.
* Attackers can exploit XSS to:
  + Execute arbitrary JavaScript
  + Steal session cookies
  + Redirect users
  + Inject fake content/forms

Developers must use input validation, output encoding, and security headers to protect against XSS.