Directed by : Tagzirt Elissa Group:SIQ1 _2023-2024

Lab7b : Attackson web applications

Objectives

The primary objective of this lab is to study two common web application attacks as per the OWASP Top 10 vulnerabilities:

1. Cross-Site Scripting (XSS)
2. Cross-Site Request Forgery (CSRF)

Lab Environment

The following resources were used to execute the lab:

- **Ubuntu Server with Apache**
- **Kali Linux VM**
- **Firefox Web Browser**
- **DVWA (Damn Vulnerable Web Application)**

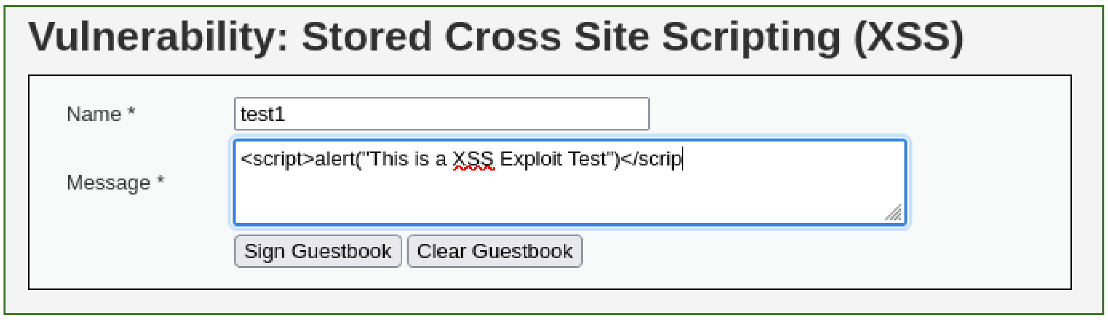
Cross-Site Scripting (XSS)

1. Stored XSS Attack Overview

Stored XSS involves injecting malicious scripts that are permanently stored on the target server, such as in a database, comment field, etc. When other users access the infected page, the script is executed in their browser context.

Steps to Execute Stored XSS Attack:

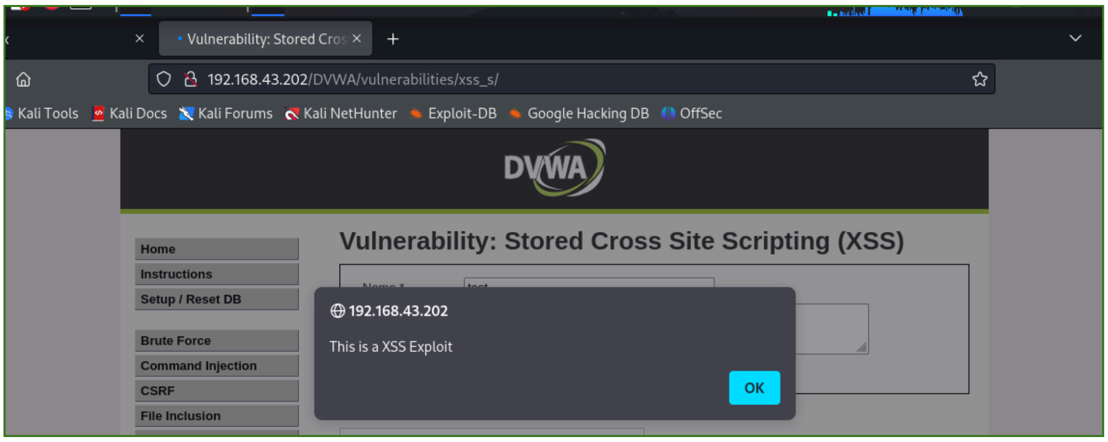
1. **Login to DVWA:**
   - Navigate to http://192.168.43.202/dvwa/login.php
   - Credentials: admin/password
2. **Set Security Level to Low:**
   - Navigate to DVWA Security from the left-hand menu
   - Select Low and click Submit
3. **Stored XSS Basic Exploit:**
   - Navigate to XSS Stored from the left-hand menu
   - Inject the following payload:
     <script>alert("This is a XSS Exploit Test")</script>



Vulnerability: Stored Cross Site Scripting (XSS)

Name *	test1
Message *	<script>alert("This is a XSS Exploit Test")</scrip

4. **View Results:**



192.168.43.202/DVWA/vulnerabilities/xss_s/

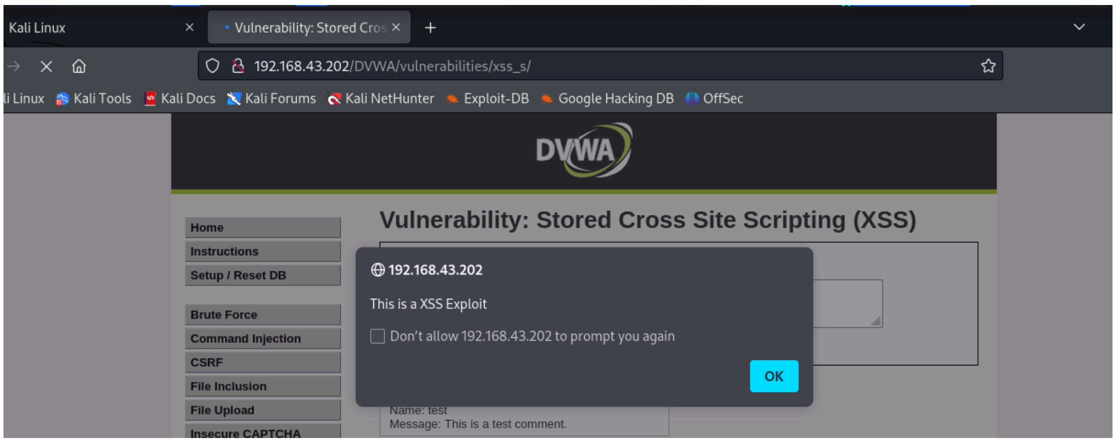
Vulnerability: Stored Cross Site Scripting (XSS)

192.168.43.202

This is a XSS Exploit

- The JavaScript alert displayed, indicating successful execution of the XSS payload.

Sample Payloads and Effects:



192.168.43.202/DVWA/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site Scripting (XSS)

192.168.43.202

This is a XSS Exploit

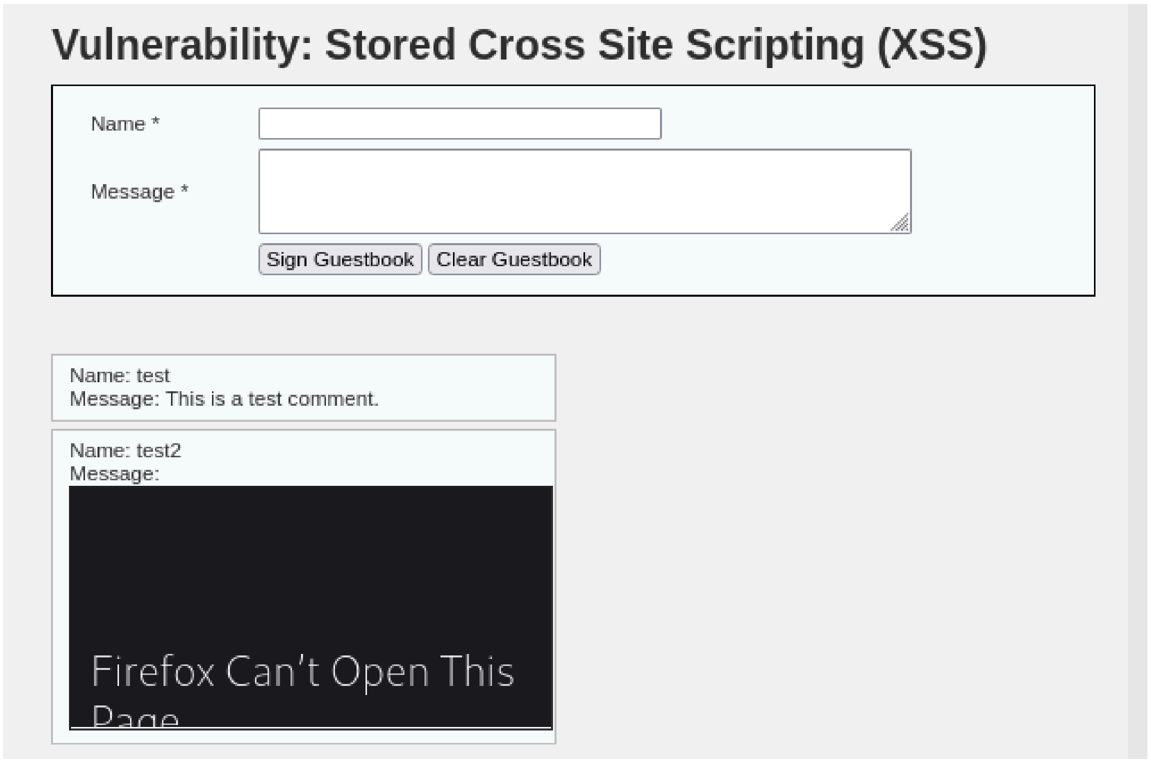
Don't allow 192.168.43.202 to prompt you again

Name: test
Message: This is a test comment.

Effect: Every time a user navigates to the page, this alert box is displayed, indicating that the script is stored on the server and executed each time the page is loaded.

2. IFRAME Exploit:

- Name: Test 2
- Message: <iframe src="http://www.cnn.com"></iframe>



Vulnerability: Stored Cross Site Scripting (XSS)

Name: test
Message: This is a test comment.

Name: test2
Message:

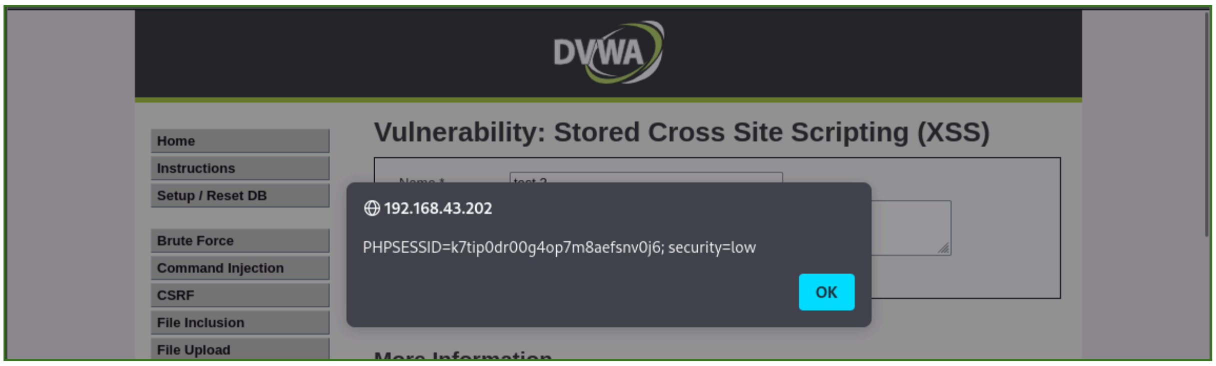
Firefox Can't Open This Page

Effect: An external website (CNN in this case) is loaded within the page, demonstrating the ability to embed arbitrary content.

- Note that modern browsers might block this for security reasons, as shown in the screenshot where the message "Firefox Can't Open This Page" is displayed instead of the embedded content

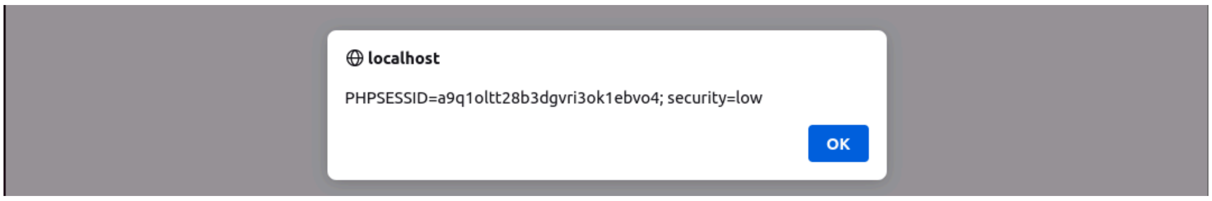
3. Cookie Stealing Script:

- Navigate to XSS Stored from the left-hand menu
- Injecting :<script>alert(document.cookie)</script>



192.168.43.202

PHPSESSID=k7tip0dr00g4op7m8aefsnv0j6; security=low



localhost

PHPSESSID=a9q1oltt28b3dgvri3ok1ebvo4; security=low

Effect: Every time a user navigates to the page, this alert box displays the current session cookie, showing that the script is stored on the server and executed each time the page is loaded.

Cross-Site Request Forgery (CSRF)

CSRF Attack Overview

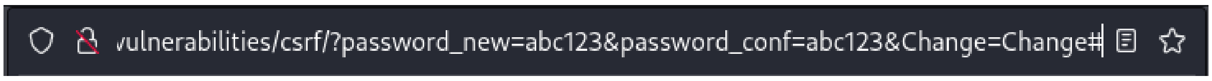
CSRF exploits the trust that a web application has in the user's browser. It forces the user to execute unwanted actions on a web application where they are authenticated.

Steps to Execute CSRF Attack:

1. **Reset the Database:**
    - Navigate to Setup from the left-hand menu
    - Click Create/Reset Database
2. **CSRF Password Change Exploit:**
    - Navigate to CSRF from the left-hand menu
    - Change password using the following parameters:
        - New Password: abc123
        - Confirm New Password: abc123

1.URL Manipulation:

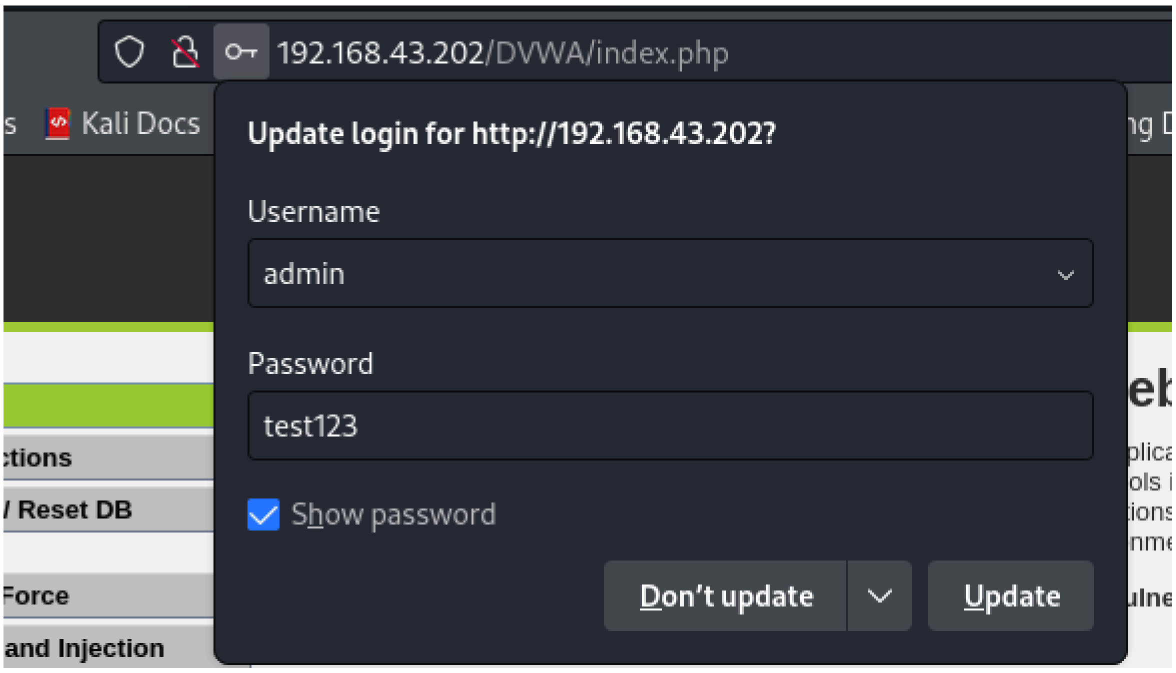
Observe the URL parameters:password_new=abc123&password_conf=abc123



vulnerabilities/csrf/?password_new=abc123&password_conf=abc123&Change=Change#

Manipulate the URL to change the password:
http://192.168.43.202/dvwa/vulnerabilities/csrf/?password_new=test123&password_conf=test123&Change=Change

Reload the page to apply changes.


Update login for http://192.168.43.202?

Username: admin

Password: test123

Effect: Changes the user's password to test123.

Countermeasures

Preventing XSS:

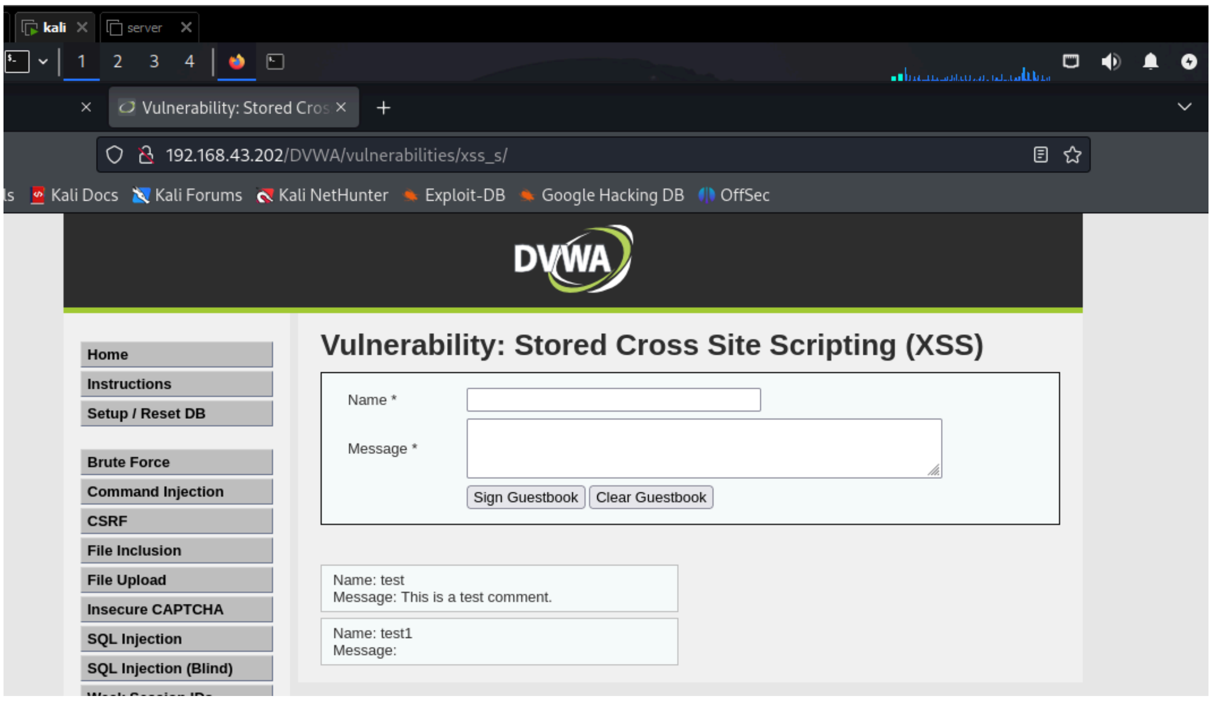
- **Input Validation:** Implement strict input validation to sanitize user inputs.
- **Output Encoding:** Encode output data to neutralize malicious scripts.
- **Content Security Policy (CSP):** Deploy CSP headers to restrict the execution of unauthorized scripts.

Preventing CSRF:

- **Anti-CSRF Tokens:** Implement tokens that must be included in all forms and verified server-side.
- **SameSite Cookies:** Set the SameSite attribute for cookies to prevent cross-origin requests.
- **User Interaction:** Require user interactions such as CAPTCHA or re-authentication for sensitive actions.

Conclusion

This lab exercise demonstrated the practical execution of XSS and CSRF attacks on a vulnerable web application using DVWA. The results highlighted the importance of securing web applications against these common vulnerabilities. Implementing the recommended countermeasures can significantly reduce the risk of exploitation.

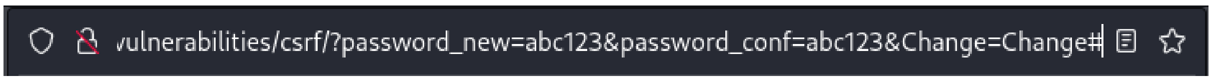


Vulnerability: Stored Cross Site Scripting (XSS)

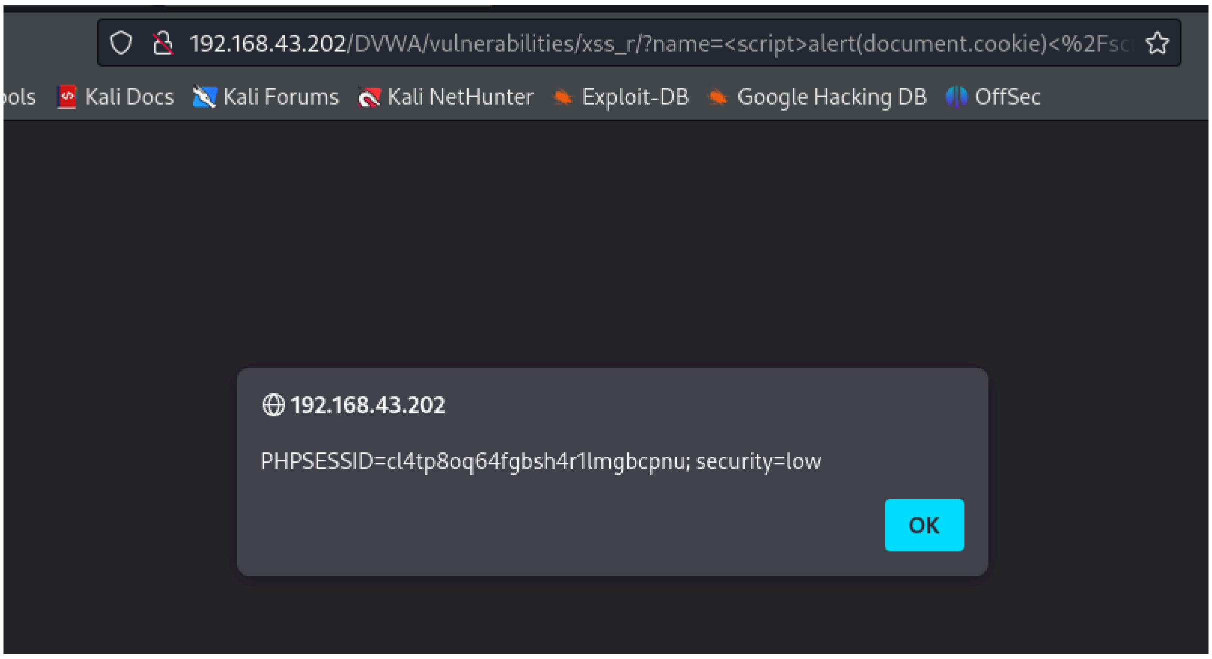
Name *	
Message *	

Name: test
Message: This is a test comment.

Name: test1
Message:



```
vulnerabilities/csrf/?password_new=abc123&password_conf=abc123&Change=Change#
```



```
192.168.43.202/DVWA/vulnerabilities/xss_r/?name=<script>alert(document.cookie)<%2Fsc
```

192.168.43.202

```
PHPSESSID=cl4tp8oq64fgbsh4r1lmgbcpnu; security=low
```