

Lab Experiment: Finding & Exploiting XSS Vulnerabilities using DVWA on Kali Linux

Aim

To identify and exploit **Cross-Site Scripting (XSS)** vulnerabilities in a vulnerable web application (DVWA) using Kali Linux.

Requirements

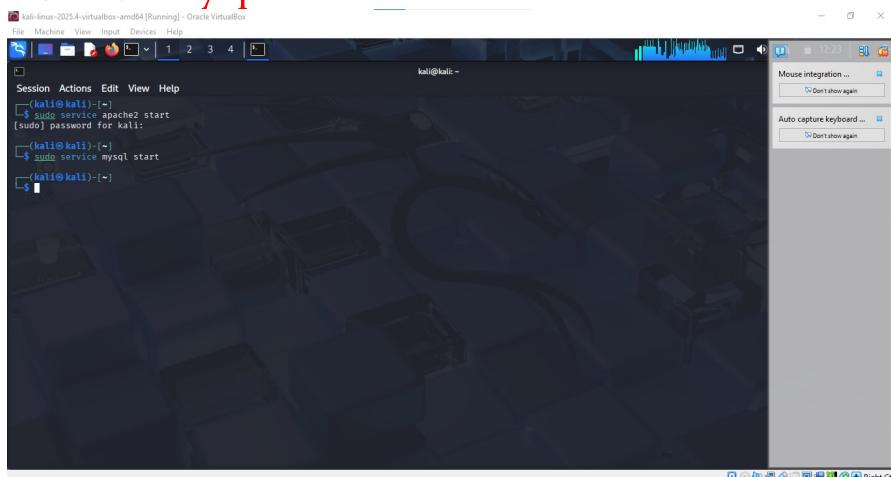
- Kali Linux
- DVWA (Damn Vulnerable Web Application)
- Browser (Firefox/Chromium)
- Apache & MySQL running

Step I: Start DVWA Services

Open terminal:

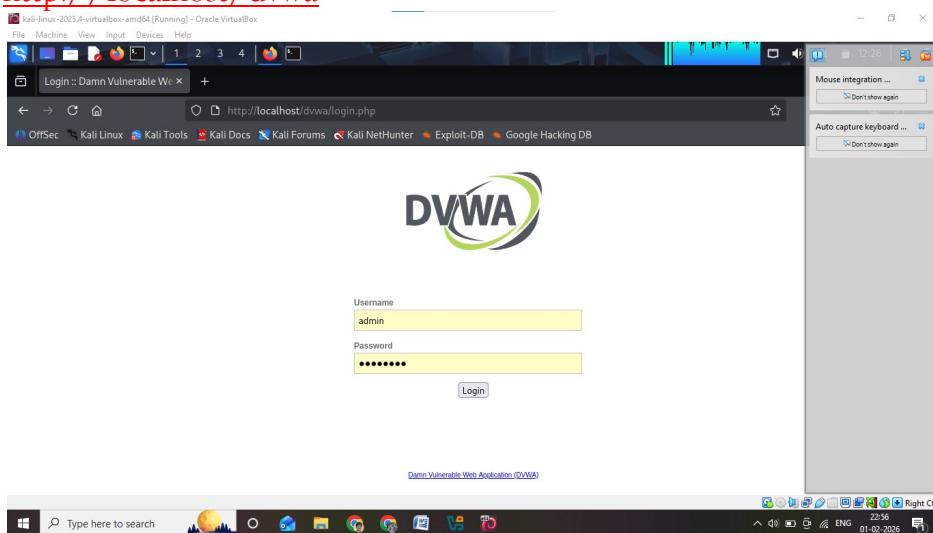
sudo service apache2 start

sudo service mysql start



Open browser and go to:

http://localhost/dvwa



Login:

- **Username:** admin
- **Password:** password

Click DVWA Security → set level to Low → Submit.

A screenshot of a Windows desktop environment showing a Firefox browser window. The title bar says "DVWA Security :: Damn V...". The address bar shows "http://localhost/dvwa/security.php". The main content area displays the DVWA Security page. On the left, there's a sidebar with various attack types: Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, and Cryptography. Below this is a green bar labeled "DVWA Security". Under "DVWA Security", there are links for "PHP Info" and "About". A text box in the center says: "You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:". Below this is a dropdown menu with options: Impossible (selected), Low, Medium, High, and Impossible. To the right of the dropdown is a "Submit" button. The status bar at the bottom shows "Type here to search", system icons, and the date/time "01-02-2026 22:57".

Step 2: Understanding XSS

XSS allows attackers to inject **JavaScript code** into a webpage that runs in another user's browser.

Types in DVWA:

- Reflected XSS
- Stored XSS
- DOM Based XSS

Step 3: Reflected XSS Test

Go to:

DVWA → XSS (Reflected)

A screenshot of a Windows desktop environment showing a Firefox browser window. The title bar says "Vulnerability: Reflected C...". The address bar shows "http://localhost/dvwa/vulnerabilities/xss_r/". The main content area displays the DVWA XSS (Reflected) test page. On the left, there's a sidebar with various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected) (selected), and XSS (Stored). Below this is a green bar labeled "DVWA". A section titled "Vulnerability: Reflected Cross Site Scripting (XSS)" contains a form with a text input field labeled "What's your name?" and a "Submit" button. Below the form is a "More Information" section with a list of links:

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

The status bar at the bottom shows "Type here to search", system icons, and the date/time "01-02-2026 22:59".

In the input box, type:

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

Click **Submit**.

A screenshot of a Windows desktop environment showing a Firefox browser window. The browser title bar says "Vulnerability: Reflected C x". The address bar shows the URL [http://localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert\('XSS'\)<%2Fscript>#](http://localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert('XSS')<%2Fscript>#). The main content area displays the DVWA logo and the title "Vulnerability: Reflected Cross Site Scripting (XSS)". Below the title is a form field with the placeholder "What's your name?". Inside the field, the user has typed "<script>alert('XSS')</script>". To the right of the input field is a "Submit" button. Below the input field, the word "Hello" is displayed in red, followed by "<script>alert('XSS')</sc...>". To the left of the main content is a sidebar menu with various vulnerability categories, and "XSS (Reflected)" is highlighted. On the right side of the screen, there is a "Mouse integration ..." and "Auto capture keyboard ..." tool window.

Output:

You will see a popup alert → **XSS vulnerability confirmed.**

A screenshot of a Windows desktop environment showing a Firefox browser window. The browser title bar says "Vulnerability: Reflected C x". The address bar shows the URL [http://localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert\('XSS'\)<%2Fscript>#](http://localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert('XSS')<%2Fscript>#). The main content area is mostly blacked out. A modal dialog box from "localhost" is centered on the screen with the text "XSS" and an "OK" button. At the bottom of the screen, a taskbar is visible with icons for various applications like File Explorer, Google Chrome, and Microsoft Edge. The system tray shows the date and time as "01-02-2026 23:01".

Step 4: Stored XSS Test

Go to:

DVWA → XSS (Stored)

Fill the form:

Name:

<h1>Hacked</h1>

Message:

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

Click Sign Guestbook.

A screenshot of a Windows desktop environment showing a Firefox browser window running on a Kali Linux VM via Oracle VirtualBox. The browser is displaying the DVWA 'Stored XSS' page. A modal dialog box from 'localhost' titled 'Stored XSS' shows the input 'Name: <h1>Hacked'. Below the dialog, the guestbook entries are visible, including one with 'Name: test' and 'Message: This is a test comment.' Another entry shows 'Name: s' and 'Message: <script>alert('Stored XSS')</script>'. The status bar at the bottom of the screen shows the date and time as 01-02-2026 23:04.

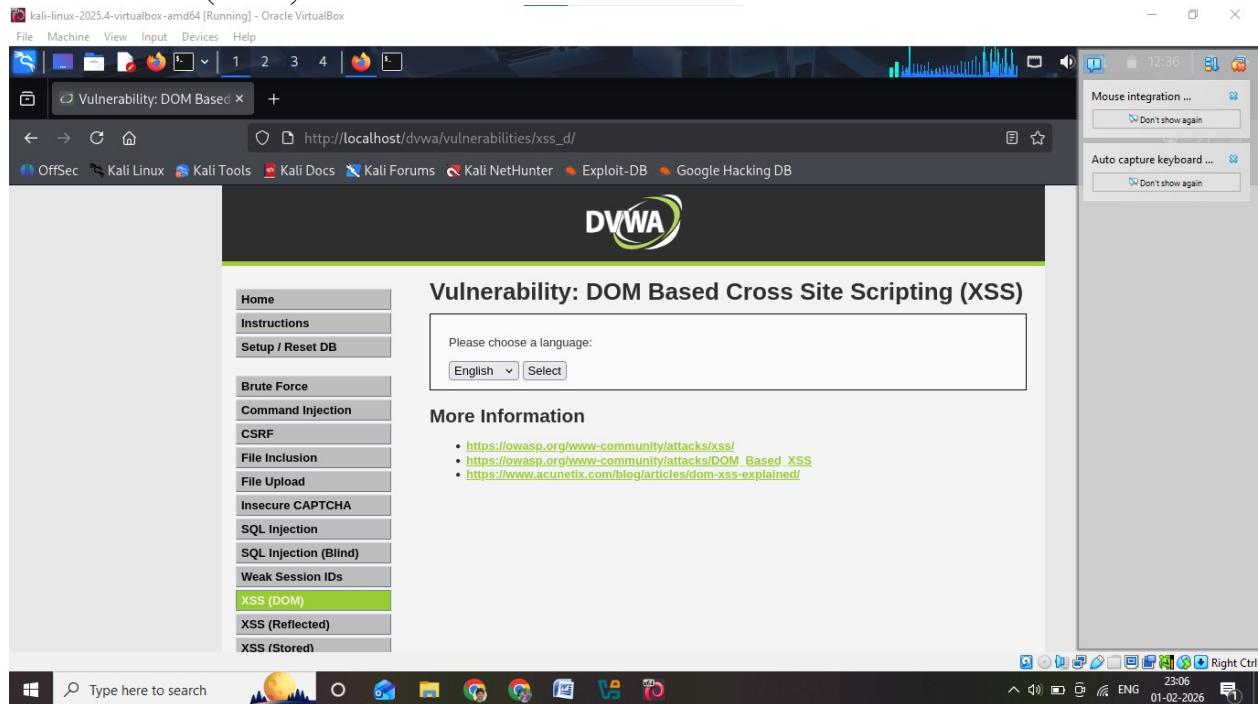
Refresh page → popup appears every time → Stored XSS successful.

A second screenshot of the same DVWA setup, showing the result of the exploit. The guestbook now displays two entries with 'Hacked' messages. The first entry has 'Name: test' and 'Message: Hacked'. The second entry has 'Name: s' and 'Message: Hacked'. The status bar at the bottom of the screen shows the date and time as 01-02-2026 23:05.

Step 5: DOM Based XSS

Go to:

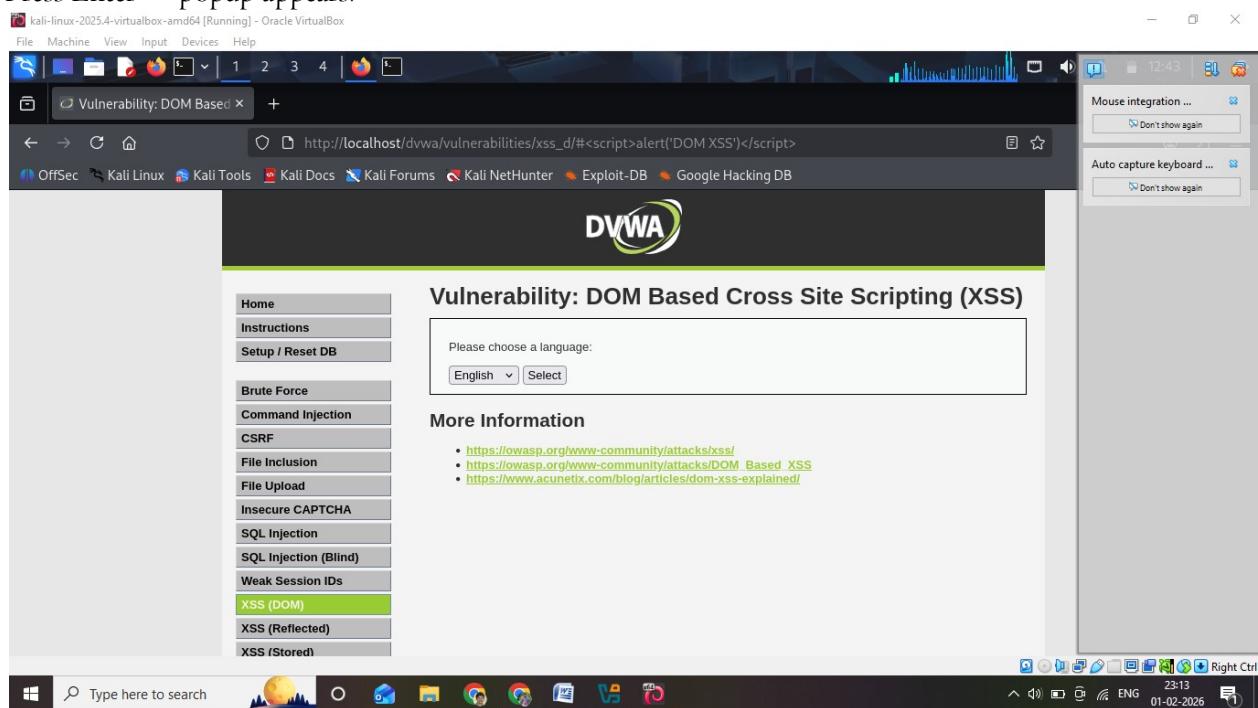
DVWA → XSS (DOM)



In the URL bar add:

#<script>alert('DOM XSS')</script>

Press Enter → popup appears.

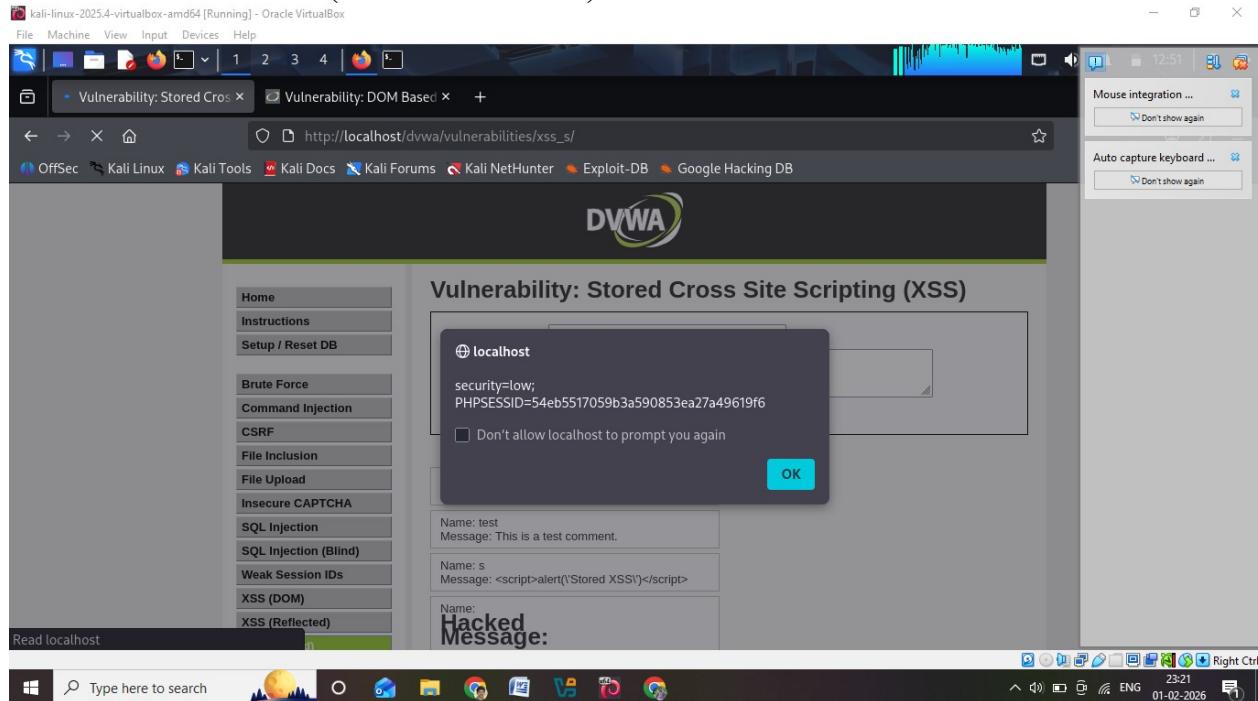


Step 6: Capture Cookie (Lab Demo)

In Stored XSS Message box:

```
<script>alert(document.cookie)</script>
```

This shows session cookies (demo of session theft).



Step 7: Change Security Level

Go to DVWA Security → set:

- Medium
- High

Repeat the same payloads → see how filtering blocks them.

Result

XSS vulnerabilities were successfully identified and exploited in DVWA.