

DVWA Penetration Test Report

Project Title: Web Application Penetration Test — DVWA (Damn Vulnerable Web Application)

Analyst: Ramjith M – Group-6

Date: 22/06/2025

Target: Local DVWA instance

Security Level: Low

1. Summary

This report documents a web application penetration test conducted on DVWA. The assessment simulated common web attacks mapped to the OWASP Top 10 vulnerabilities. Multiple vulnerabilities were identified, some exploited successfully, and proof-of-concept (PoC) evidence was collected. Remediation guidance is provided to mitigate these risks.

2. Scope

- **Target:** DVWA running in a controlled lab environment.
 - **Test Type:** Manual exploitation with Burp Suite and browser-based payloads.
 - **Focus:**
 - SQL Injection
 - Command Injection
 - Cross-Site Scripting (Reflected & Stored)
 - Brute Force
 - CSRF
 - Sensitive Data Exposure
 - Broken Access Control (attempted)
 - Security Misconfiguration
-

3. Methodology

Testing followed industry-standard practice:

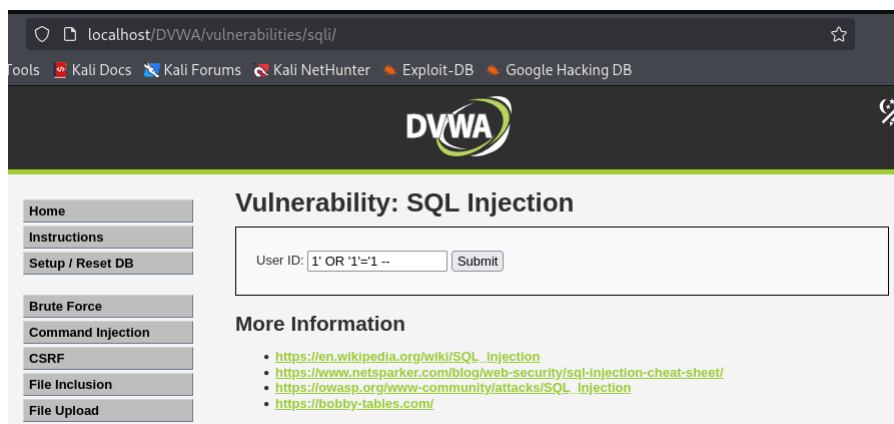
- Reconnaissance using Burp Suite proxy
 - Manual injection and payload testing for SQLi, XSS, Command Injection.
 - Brute force performed using Burp Intruder.
 - CSRF tested using custom HTML forms and direct requests.
 - Traffic inspection for sensitive data exposure.
 - Attempts to bypass access controls directly by URLs.
-

Vulnerability	Typical CVSS v3.1 Score	Risk Level
SQL Injection	9.8 (Critical)	Critical impact, allows DB compromise
Command Injection	9.0 (Critical)	Remote code execution, full system compromise
Reflected XSS	6.1 (Medium)	User session hijack, limited scope
Stored XSS	7.4 (High)	Persistent user compromise, broader impact
Brute Force (Weak Authentication)	6.5 (Medium)	Credential compromise, unauthorized access
CSRF	6.8 (Medium)	Unauthorized state changes
Sensitive Data Exposure (HTTP)	7.5 (High)	User credentials exposed over network
Broken Access Control	<i>(Attempted, no bypass)</i> — Not rated, as not exploitable	
Security Misconfiguration (Low Security Level)	4.3 (Low)	Configuration issue, no direct exploit

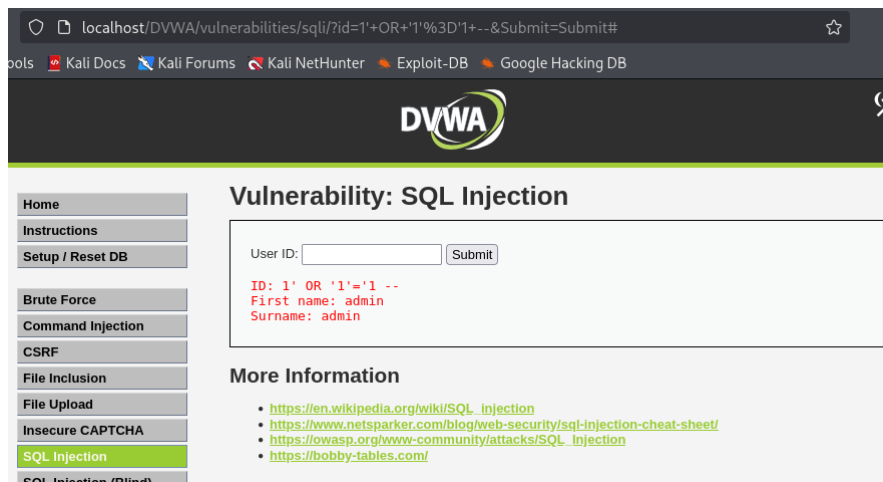
4. Findings

4.1 SQL Injection

- **Status:** Successfully exploited.
- **Technique:** 'OR' 1 '1'='1 -- bypassed query logic, dumping user table.
- **Impact:** Full read access to backend database.
- **Evidence:**
- SQLi input –



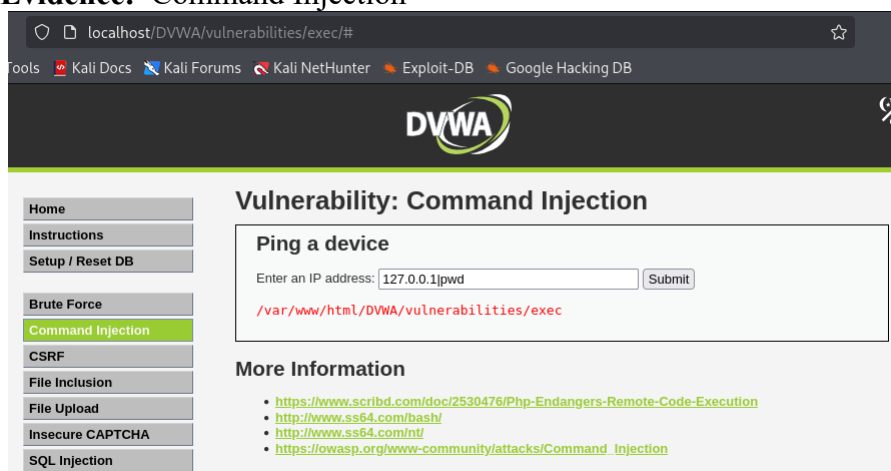
- SQLi output –



- **Remediation:** Use parameterized queries, enforce input validation.

4.2 Command Injection

- **Status:** Successfully exploited.
- **Technique:** `127.0.0.1|pwd` Executed OS command.
- **Impact:** Revealed Print Working Directory.
- **Evidence:** Command Injection

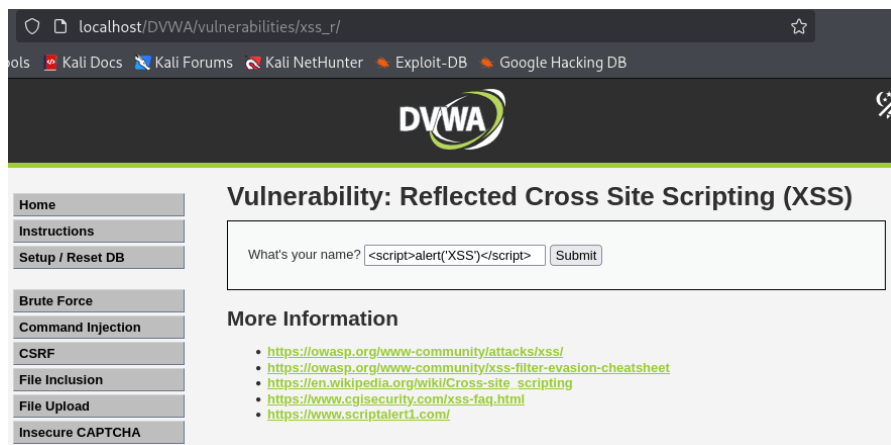


- **Remediation:** Sanitize input, use safe API calls.

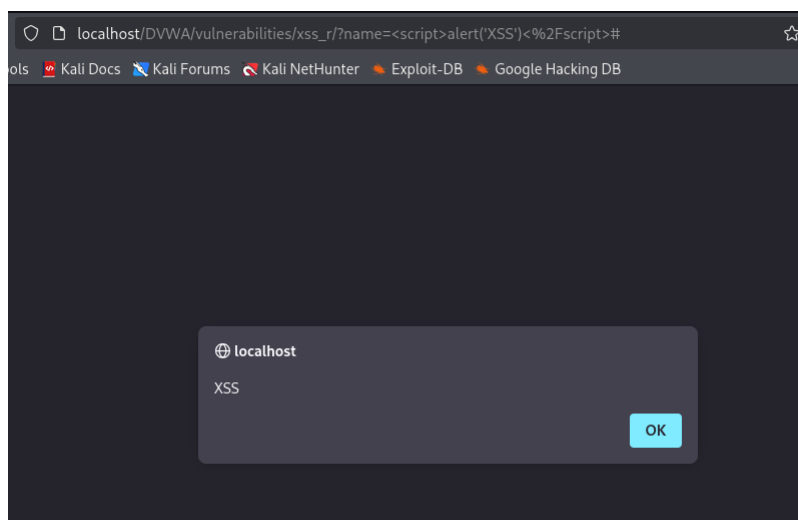
4.3 Cross-Site Scripting (Reflected)

- **Status:** Successfully exploited.
- **Technique:** `<script>alert('XSS')</script>`
- **Impact:** Arbitrary script execution in user's browser.
- **Evidence:**

XSS Reflected input



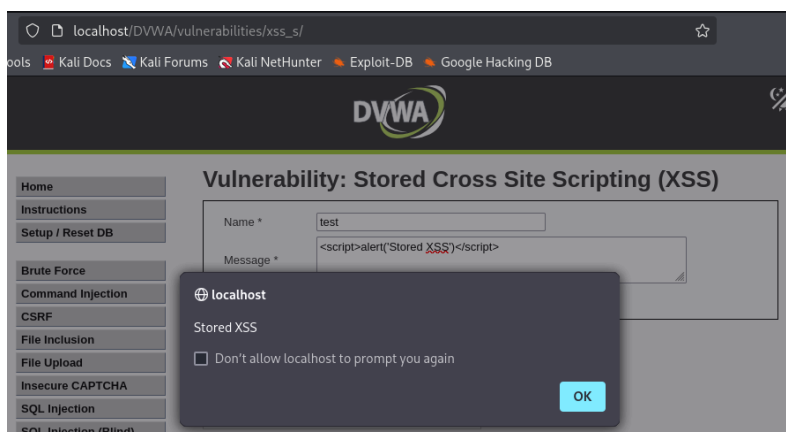
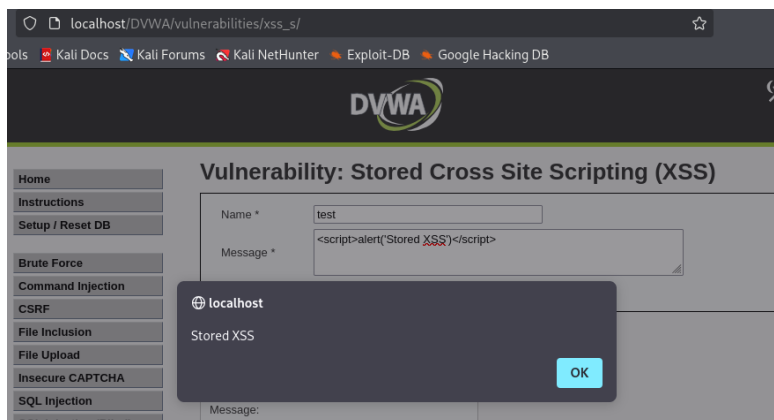
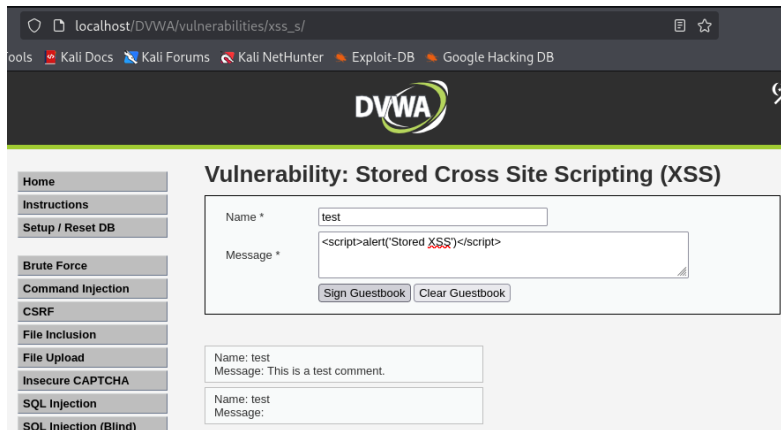
XSS Reflected output.



- **Remediation:** Encode output, sanitize input.

4.4 Cross-Site Scripting (Stored)

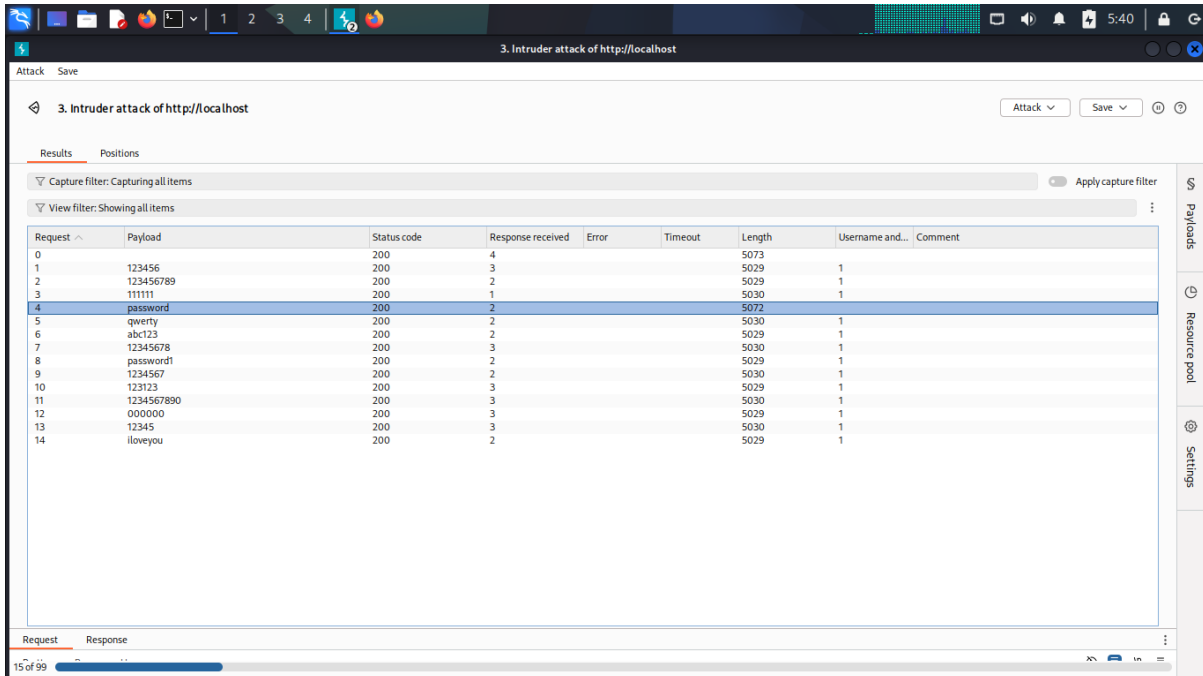
- **Status:** Successfully exploited.
- **Technique:** `<script>alert('Stored XSS')</script>` in guestbook.
- **Impact:** Persistent client-side code execution.
- **Evidence:**



- **Remediation:** Apply input sanitization and output encoding.

4.5 Brute Force

- **Status:** Successfully exploited.
- **Technique:** Burp Intruder attack against login page.
- **Impact:** Weak password policy allows guessing.
- **Evidence:**



3. Intruder attack of http://localhost

Attack Save

Results Positions

Capture filter: Capturing all items Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Username and...	Comment
0		200	4			5073		
1	123456	200	3			5029	1	
2	123456789	200	2			5029	1	
3	111111	200	1			5030	1	
4	password	200	2			5072		
5	qwerty	200	2			5030	1	
6	abc123	200	2			5029	1	
7	12345678	200	3			5030	1	
8	password1	200	2			5029	1	
9	1234567	200	2			5030	1	
10	123123	200	3			5029	1	
11	1234567890	200	3			5030	1	
12	000000	200	3			5029	1	
13	12345	200	3			5030	1	
14	iloveyou	200	2			5029	1	

Request Response

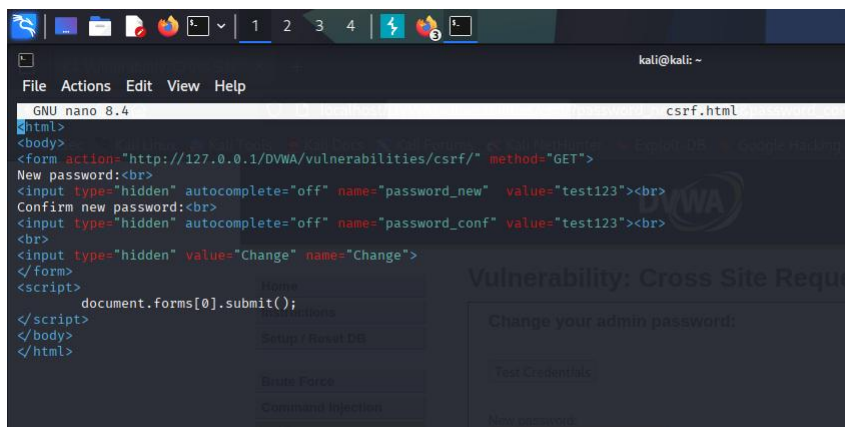
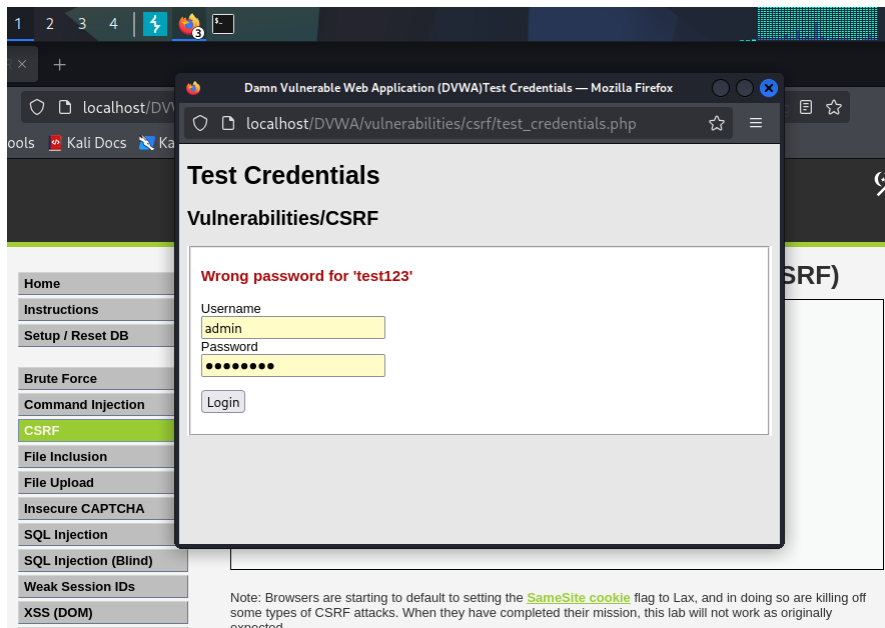
15 of 99

Payloads Resource pool Settings

- **Remediation:** Enforce strong password policy, implement lockouts.

4.6 CSRF

- **Status:** Attempted. Limited due to browser security; tested via crafted HTML and Burp.
- **Evidence:**



- **Remediation:** Implement anti-CSRF tokens for state-changing requests.

4.7 Sensitive Data Exposure

- **Status:** Confirmed.
- **Technique:** Plaintext credentials intercepted.
- **Impact:** Usernames and passwords visible over HTTP.

- **Evidence:**

The screenshot displays the Burp Suite interface. The top menu bar includes options like Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The main workspace is divided into three panes: a list of intercepted requests, a detailed view of the selected request, and a detailed view of the selected response.

Request List:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener
54	http://localhost	GET	/DWA/vulnerabilities/brute/?user...		✓	200	5072	HTML		Vulnerability: Brute Fo...		127.0.0.1	127.0.0.1		05:27:49.24...	8081
53	http://localhost	GET	/DWA/vulnerabilities/brute/			200	4977	HTML		Vulnerability: Brute Fo...		127.0.0.1	127.0.0.1		05:27:45.24...	8081
52	http://localhost	GET	/DWA/vulnerabilities/brute/			200	4916	HTML		Vulnerability: Brute Fo...		127.0.0.1	127.0.0.1		05:27:41.24...	8081
50	http://localhost	GET	/DWA/vulnerabilities/brute/?user...		✓	200	5073	HTML		Vulnerability: Brute Fo...		127.0.0.1	127.0.0.1		05:27:08.24...	8081
49	http://localhost	GET	/DWA/vulnerabilities/brute/?user...		✓	200	5073	HTML		Vulnerability: Brute Fo...		127.0.0.1	127.0.0.1		05:27:03.24...	8081
48	http://localhost	GET	/DWA/vulnerabilities/brute/?user...		✓	200	5073	HTML		Vulnerability: Brute Fo...		127.0.0.1	127.0.0.1		05:26:14.24...	8081
47	http://localhost	GET	/DWA/vulnerabilities/brute/			200	4978	HTML		Vulnerability: Brute Fo...		127.0.0.1	127.0.0.1		05:26:07.24...	8081
46	http://localhost	GET	/DWA/			200	6730	HTML		Welcome :: Damn Vul...		127.0.0.1	127.0.0.1		05:25:36.24...	8081
45	http://localhost	GET	/DWA/security.php			200	5367	HTML	php	DWA Security :: Dam...		127.0.0.1	127.0.0.1		05:25:36.24...	8081
44	http://localhost	POST	/DWA/security.php		✓	302	497	HTML	php			127.0.0.1	127.0.0.1	security=low;PH...	05:25:36.24...	8081
43	http://localhost	GET	/DWA/security.php			200	5299	HTML	php	DWA Security :: Dam...		127.0.0.1	127.0.0.1		05:25:33.24...	8081
42	http://localhost	POST	/DWA/security.php		✓	200	5299	HTML	php			127.0.0.1	127.0.0.1		05:25:25.24...	8081

Request Details (Selected):

```

1 GET /DWA/vulnerabilities/brute/?username=admin&password=password&login=Login
2 HTTP/1.1
3 Host: localhost
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
5 Firefox/128.0
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
7 Accept-Language: en-US,en;q=0.5
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10 Referer: http://localhost/DWA/vulnerabilities/brute/
11 Cookie: PHPSESSID=014a5214e2671ca3eac28d39bbf6ff6; security=low
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
  
```

Response Details (Selected):

```

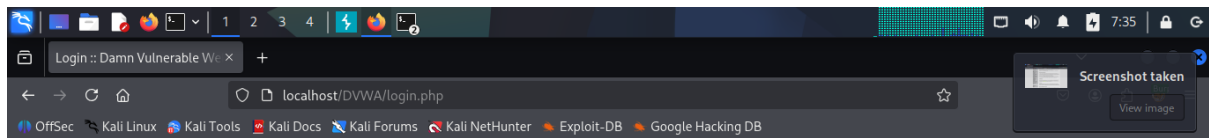
1 HTTP/1.1 200 OK
2 Date: Tue, 24 Jun 2025 09:27:50 GMT
3 Server: Apache/2.4.63 (Debian)
4 Expires: Tue, 23 Jun 2009 12:00:00 GMT
5 Cache-Control: no-cache, must-revalidate
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 4745
9 Keep-Alive: timeout=5, max=98
10 Connection: Keep-Alive
11 Content-Type: text/html; charset=utf-8
12
13 <!DOCTYPE html>
14
15 <html lang="en-GB">
16
17 <head>
18 <meta http-equiv="Content-Type" content="text/html" />
  
```

The Inspector pane on the right shows the selected text from the response body: `GET /DWA/vulnerabilities/brute/?username=admin&password=password&login=Login HTTP/1.1`.

- **Remediation:** Use HTTPS/TLS for all authentication endpoints.

4.8 Broken Access Control

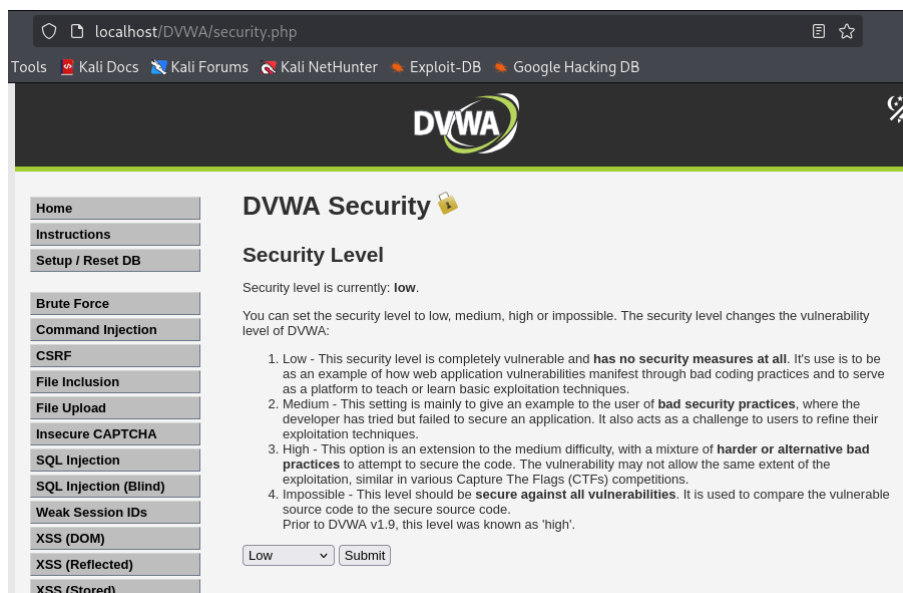
- **Status:** Attempted. Direct URL access redirects to login; no bypass found.
- **Evidence:** broken access control not working.png

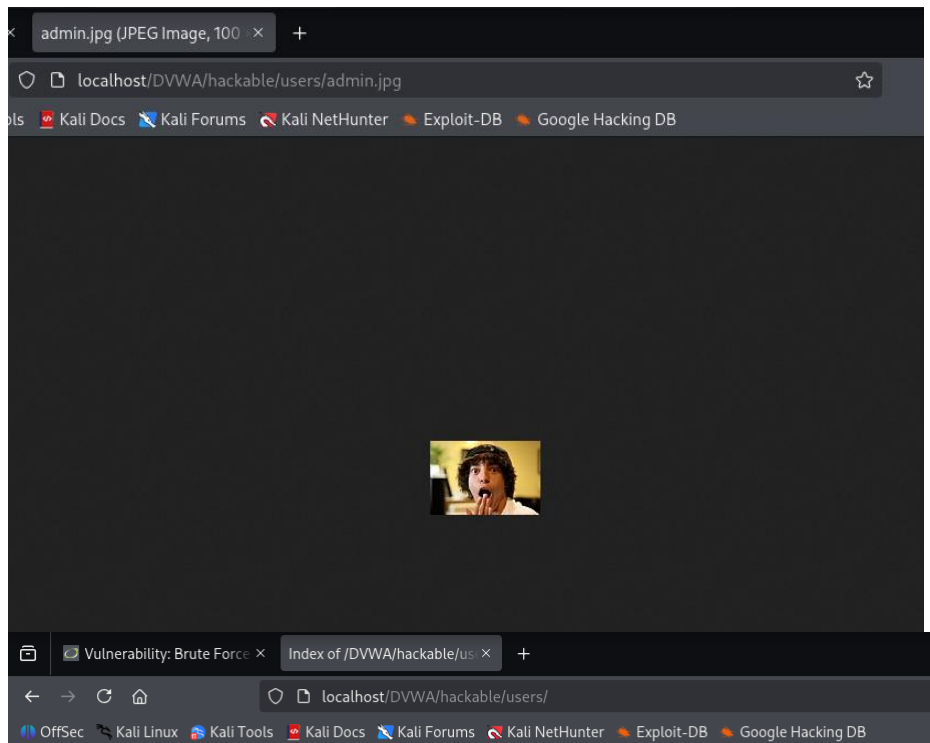


- **Remediation:** Keep session checks enforced server-side.

4.9 Security Misconfiguration

- **Status:** Confirmed.
- **Finding:** DVWA security level set to “Low”.
- **Evidence:**





Index of /DVWA/hackable/users

Name	Last modified	Size	Description
Parent Directory	-	-	-
1337.jpg	2025-06-16 19:36	3.6K	
admin.jpg	2025-06-16 19:36	3.5K	
gordonb.jpg	2025-06-16 19:36	3.0K	
pablo.jpg	2025-06-16 19:36	2.9K	
smithy.jpg	2025-06-16 19:36	4.3K	

Apache/2.4.63 (Debian) Server at localhost Port 80

- **Remediation:** Use secure configurations in production; remove debug/low settings.

Conclusion

The DVWA instance demonstrates common web vulnerabilities when misconfigured. Critical issues like SQLi, XSS, and Command Injection were exploited successfully. Remediation requires secure coding, strict input validation, proper authentication controls, and encryption of sensitive data in transit.