

# Cyber Security Task 1



## Web Application Security Testing

Intern Name: Dhrumi Sonani

Report Title: VAPT Report on DVWA

Internship provider: Future Intern

Domain: Cyber Security

### Table of Contents

- Objective
- Test Environment Setup
- Methodology & tools
- Vulnerability Assessment Results
- Risk Evaluation and Impact Analysis
- Remediation Recommendations
- References

### Objective

The objective of this project is to identify and assess common web application vulnerabilities in DVWA, such as SQL Injection, XSS (cross-site scripting), and CSRF attacks and Brute Force attacks, using tools like Burp Suite, Nikto. The goal is to understand how attackers exploit these flaws and to recommend effective remediation measures.

### Test Environment Setup

A controlled test environment was created using **Kali Linux** and the **Damn Vulnerable Web Application (DVWA)** to simulate a realistic web application penetration testing scenario. The environment includes intentionally vulnerable software and industry-standard security testing tools.

Access URL : <http://localhost/dvwa>

Security level: Set to low via DVWA Security panel

The screenshot shows the DVWA Security interface. On the left is a sidebar with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, and XSS (DOM). The main content area is titled "DVWA Security" with a padlock icon. Below it, "Security Level" is set to "low". A note states: "You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:". A numbered list explains the levels: 1. Low - Completely vulnerable with no security measures at all. 2. Medium - Example of bad security practices. 3. High - Extension to medium difficulty with harder or alternative bad practices. 4. Impossible - Secure against all vulnerabilities.

## Methodology & Tools

- **Firefox** – Used for testing and interacting with DVWA
- **Burp Suite** – Intercepted and analyzed HTTP requests
- **DVWA** – Target application for vulnerability testing

## Task Overview

- Logged into the DVWA application hosted on the local server



A screenshot of the DVWA login page. It features the DVWA logo at the top. Below it is a form with two input fields: "Username" containing "root" and "Password" containing "root". A blue rectangular box highlights the "Password" field. At the bottom of the form is a "Login" button.

- Switched the DVWA security level between Low and Medium to observe vulnerability behavior under different configurations
- Explored and tested key vulnerabilities including SQL Injection, XSS, and CSRF

## Test for Vulnerabilities

### 1. SQL Injection Testing (DVWA > SQL Injection Testing)

SQL Injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It can allow attackers to:

- Bypass authentication
- Retrieve, modify or delete data
- Execute administrative operations

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (the selected tab), SQL Injection (Blind), Weak Session IDs, and XSS (DOM). The main content area has a title "Vulnerability: SQL Injection". A form contains a "User ID:" field with the value "1'OR'1='1" and a "Submit" button. Below the form, under the heading "More Information", there is a bulleted list of four links related to SQL injection.

User ID: 1'OR'1='1

More Information

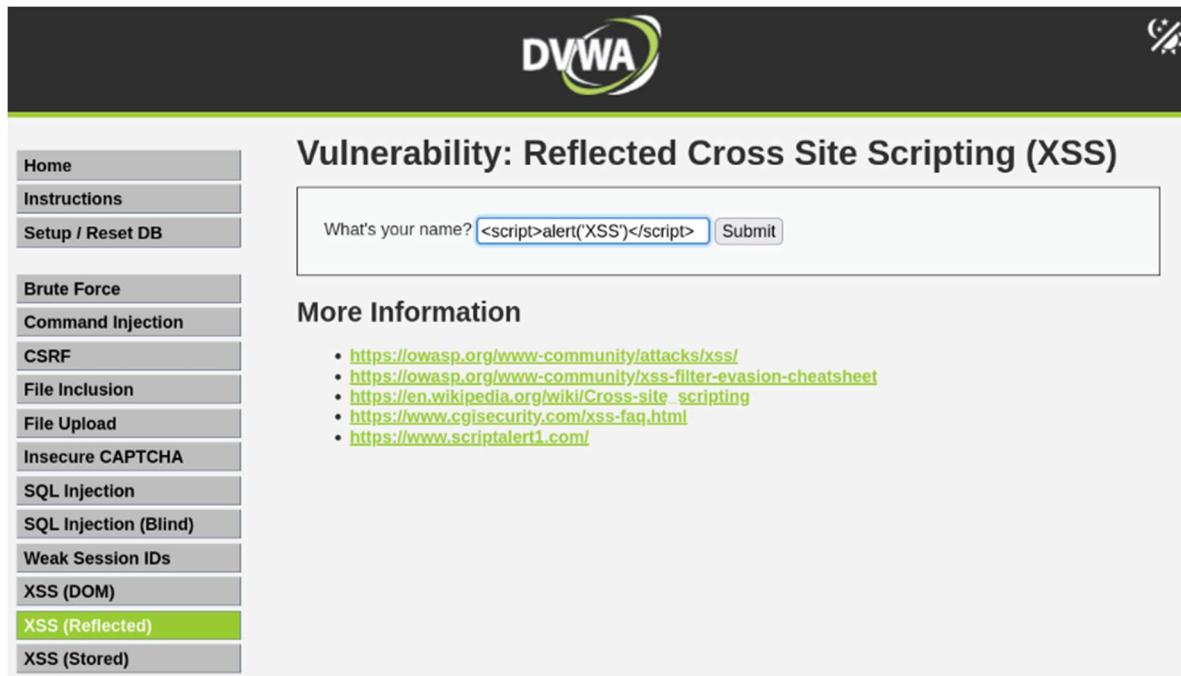
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

The screenshot shows the DVWA SQL Injection page after exploiting the vulnerability. The "User ID:" field now contains "1'OR'1='1". The main content area displays a dump of user data from the database, listing five users with their ID, first name, and surname. The output is color-coded in red for the database query and black for the results.

ID	First name	Surname
1'OR'1='1	admin	admin
1'OR'1='1	Gordon	Brown
1'OR'1='1	Hack	Me
1'OR'1='1	Pablo	Picasso
1'OR'1='1	Bob	Smith

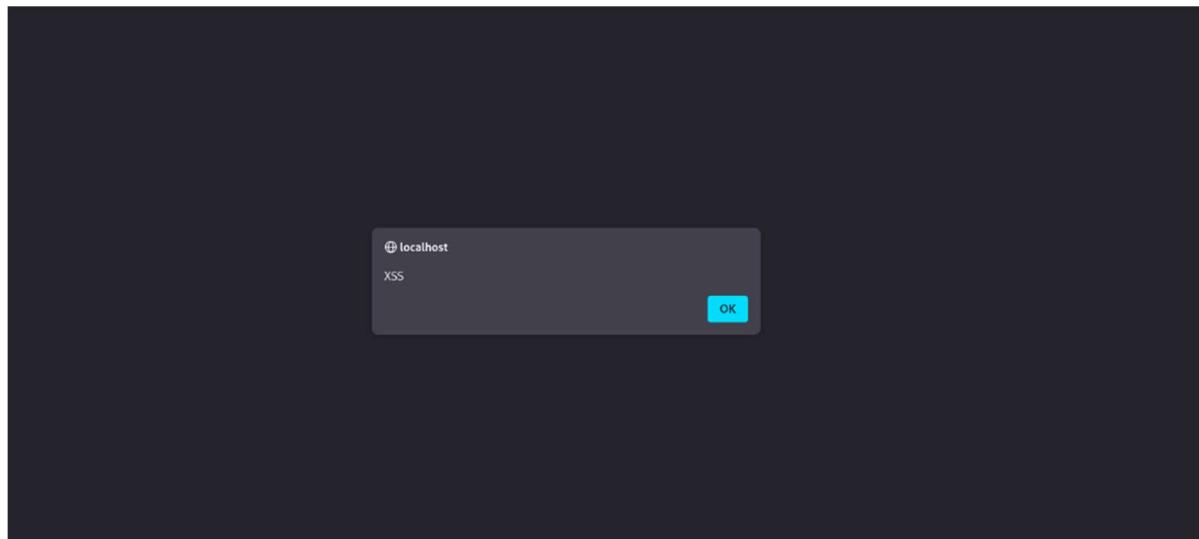
## 2. Vulnerability: Reflected Cross-Site Scripting (XSS)

Reflected XSS is a client-side vulnerability where user input is immediately back by the server without proper validation or encoding. This allows attackers to inject malicious JavaScript, which executes in the victim's browser.



The screenshot shows the DVWA application interface. The left sidebar contains a menu with various security test categories. The 'XSS (Reflected)' option is highlighted with a green background. The main content area has a title 'Vulnerability: Reflected Cross Site Scripting (XSS)'. Below the title is a form field containing the value '<script>alert('XSS')</script>'. To the right of the field is a 'Submit' button. Further down, there is a section titled 'More Information' with a bulleted list of links related to XSS attacks.

- <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://en.wikipedia.org/wiki/Cross-site_scripting)
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

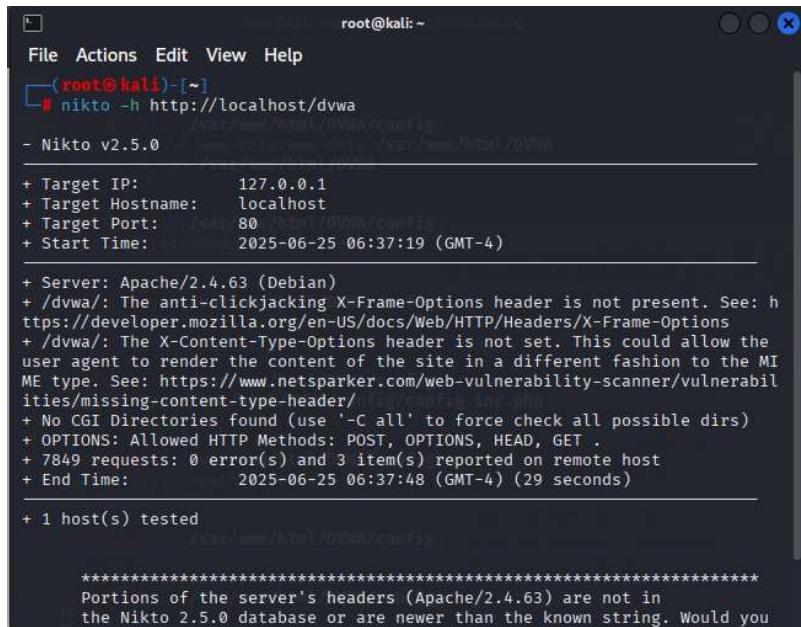


### 3. Vulnerability Scanning using Nikto on DVWA

Vulnerabilities are detectable by scanners like Nikto.

Nikto will start scanning and output results like:

- HTTP headers
- Server banner
- Directory structure
- Security issues



```
root@kali: ~
File Actions Edit View Help
[root@kali] ~]
# nikto -h http://localhost/dvwa
[+] http://127.0.0.1/DVWA/cnfFlag
- Nikto v2.5.0
+ Target IP:      127.0.0.1
+ Target Hostname: localhost
+ Target Port:    80
+ Start Time:    2025-06-25 06:37:19 (GMT-4)

+ Server: Apache/2.4.63 (Debian)
+ /dvwa/: The anti-clickjacking X-Frame-Options header is not present. See: h
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /dvwa/: The X-Content-Type-Options header is not set. This could allow the
user agent to render the content of the site in a different fashion to the MI
ME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabil
ties/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OPTIONS: Allowed HTTP Methods: POST, OPTIONS, HEAD, GET .
+ 7849 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:      2025-06-25 06:37:48 (GMT-4) (29 seconds)

+ 1 host(s) tested
[+] http://127.0.0.1/DVWA/cnfFlag

*****
Portions of the server's headers (Apache/2.4.63) are not in
the Nikto 2.5.0 database or are newer than the known string. Would you
```

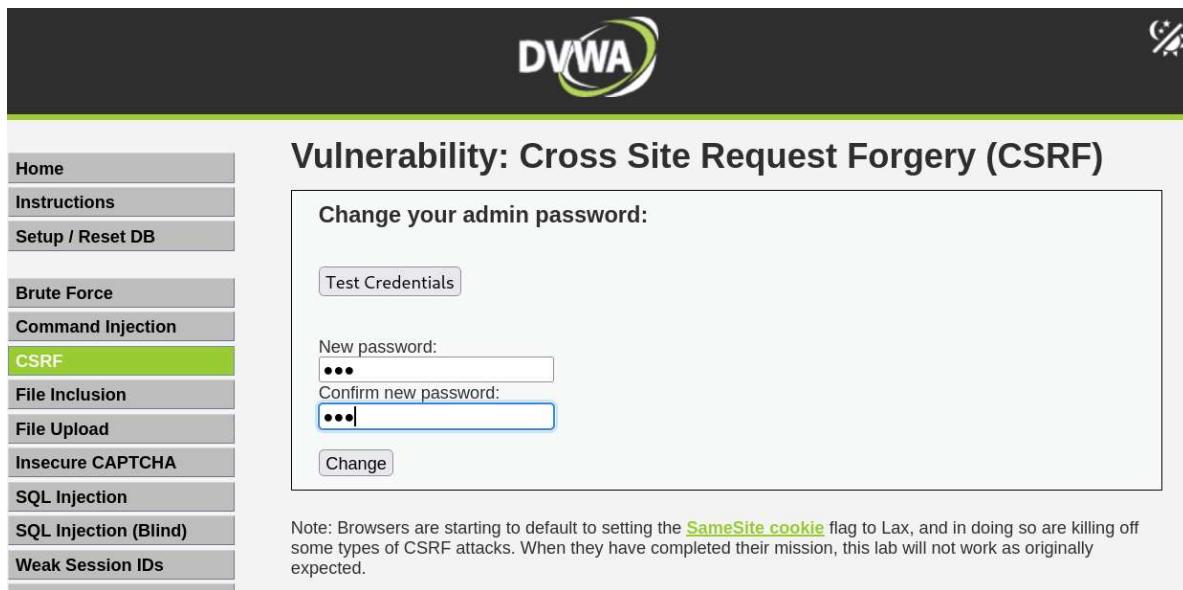
### 4. Vulnerability Assessment: Cross-Site Request Forgery (CSRF)

CSRF occurs when a malicious website or script causes a user's browser to perform unwanted actions on a web application where they are authenticated. Since DVWA lacks CSRF tokens and origin validation, it accepts requests without verifying if they came from a trusted source.

#### Impact:

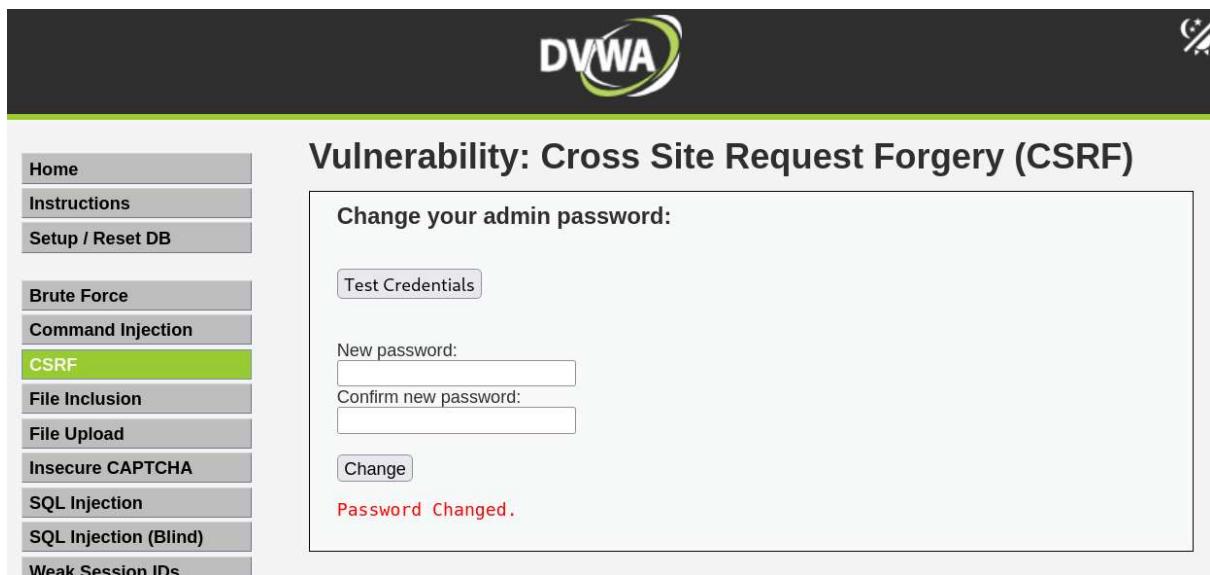
An attacker can:

- Change the victim's password
- Perform sensitive actions on behalf of the victim
- Potentially gain full control of user accounts



The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, **CSRF**, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), and Weak Session IDs. The 'CSRF' item is highlighted with a green background. The main content area has a title 'Vulnerability: Cross Site Request Forgery (CSRF)'. Below it, a form titled 'Change your admin password:' contains fields for 'Test Credentials', 'New password' (containing three asterisks), 'Confirm new password' (also containing three asterisks), and a 'Change' button. A note at the bottom states: 'Note: Browsers are starting to default to setting the [SameSite cookie](#) flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed their mission, this lab will not work as originally expected.'

DVWA showing password changed successfully.



This screenshot shows the DVWA application after a successful password change. The layout is identical to the previous one, with the 'CSRF' item still highlighted in the sidebar. The main content area now displays the message 'Password Changed.' in red text below the password input fields.

## 5. Vulnerability: Insecure Direct Object Reference (IDOR)

IDOR occurs when an application exposes object identifiers (e.g., user IDs) in the URL or request, and fails to properly enforce access control. Attackers can manipulate these references to access unauthorized data.

### Impact:

- Unauthorized access to sensitive data
- Potential for data leakage or privilege escalation

DVWA

## Vulnerability: SQL Injection

User ID:   
1' OR '1='1

Submit

**More Information**

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

Showing data for user 1

DVWA

## Vulnerability: SQL Injection

User ID:   
ID: 1  
First name: admin  
Surname: admin

Submit

**More Information**

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

Showing data for user 2

The screenshot shows the DVWA application interface. On the left is a sidebar with navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (the current module), and SQL Injection (Blind). The main content area has a title 'Vulnerability: SQL Injection'. Below it is a login form with a 'User ID:' input field containing '2' and a 'Submit' button. The response shows red text: 'ID: 2', 'First name: Gordon', and 'Surname: Brown'. Below this is a 'More Information' section with a bulleted list of four external links.

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

## Findings Summary

- The DVWA application was tested under low-security settings to identify common web vulnerabilities. Four major issues were observed: SQL Injection, Reflected Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Insecure Direct Object Reference (IDOR).
- The **SQL Injection** vulnerability was found in the login form, where unsanitized input allowed malicious SQL queries to be executed. This posed a high risk of unauthorized access or data manipulation.
- The **Reflected XSS** vulnerability was observed in the XSS module, where user input was reflected back in the response without proper encoding. This allowed JavaScript execution in the browser, resulting in a medium-level security risk.
- The **CSRF** vulnerability affected the password change functionality. The absence of CSRF tokens allowed unauthorized requests to be submitted on behalf of authenticated users, leading to a high-severity issue.
- A simulated **IDOR** vulnerability demonstrated that by modifying object references (e.g., user ID in the URL), it was possible to access other users' data without proper authorization checks. This was categorized as a high-risk issue.

## Recommendations

- Use input validation and output encoding to prevent SQLi and XSS.
- Implement CSRF tokens and verify request origins.
- Apply access control checks to protect against unauthorized access (IDOR).
- Use secure identifiers (e.g., UUIDs) instead of direct object references.
- Keep all systems and applications updated with security patches.