Web Application Vulnerability Assessment Report

Internship Project – Cybersecurity | Future Interns

Project Title: Web Application Security Testing on DVWA (Damn Vulnerable Web Application)
Name: Achyuth C.V
Platform Used: Kali Linux (via VirtualBox)
Target Application: DVWA (Docker Container)
Tools Used: Manual Browser Input, Docker, Kali Linux

- Objective

To conduct a security assessment of a sample web application (DVWA) by identifying at least 3-5 real-world vulnerabilities that map to the OWASP Top 10 threats. The vulnerabilities tested include SQL Injection, Cross-Site Scripting (XSS), Broken Authentication, and CSRF.

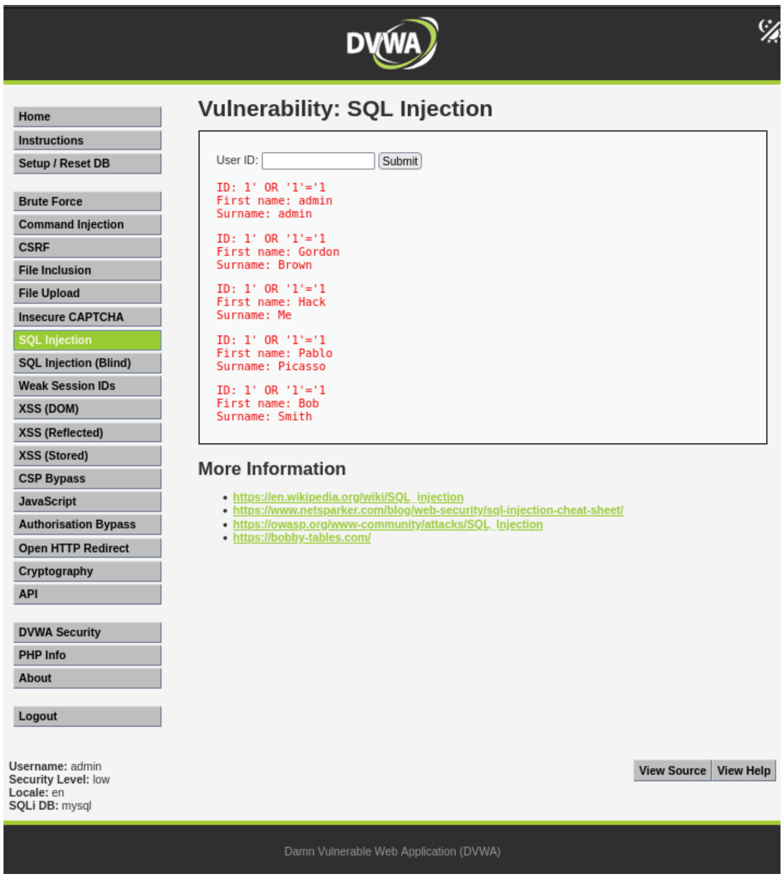
- Setup Summary

- **Operating System:** Kali Linux (running in VirtualBox)
- **DVWA Installation:** Launched via Docker (vulnerables/web-dvwa image)
- **Database:** Auto-configured within DVWA Docker container
- **Browser:** Firefox (Kali default)
- **Security Level in DVWA:** Set to "Low" to allow easy testing of vulnerabilities

- Vulnerability 1: SQL Injection

Tested Page: DVWA → SQL Injection
Payload Used: 1' OR '1'='1
Result: Successfully retrieved admin user data by injecting into the "User ID" field. This proves that input validation is absent and database queries are vulnerable to injection.
OWASP Mapping: A1: Injection

Impact: High
Mitigation: Implement parameterized queries and input sanitization.



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

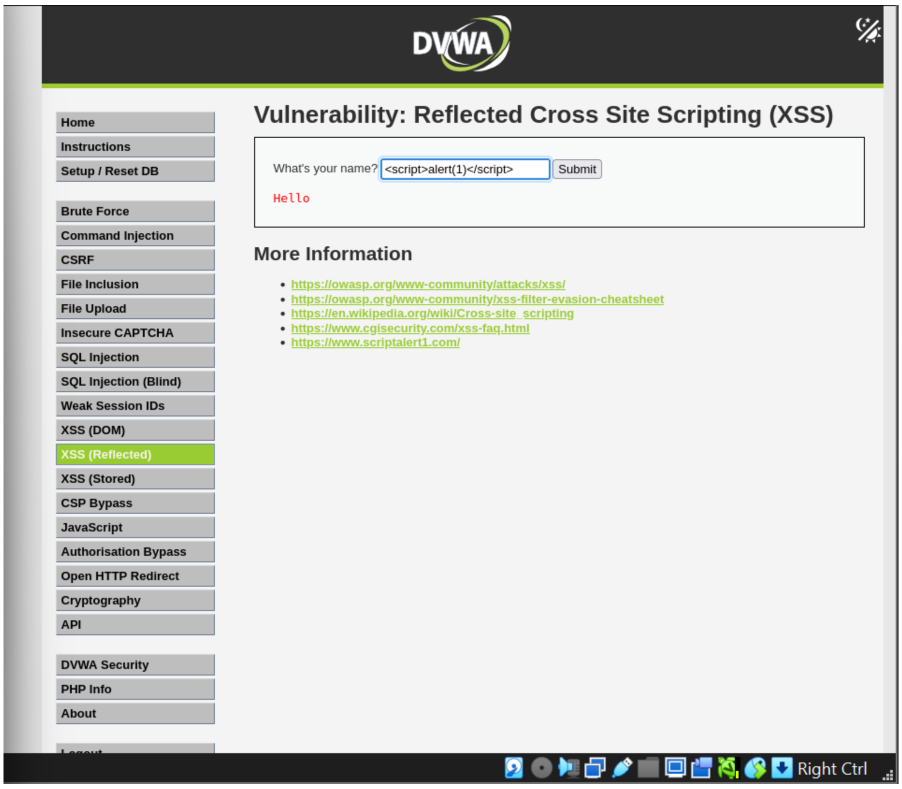
- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

Username: admin
Security Level: low
Locale: en
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

- Vulnerability 2: Reflected XSS

Tested Page: DVWA → Reflected XSS
Payload Used: <script>alert('Hello')</script>
Result: JavaScript executed in browser alert, proving XSS vulnerability via user input. Reflected payload was not properly encoded.
OWASP Mapping: A7: Cross-Site Scripting
Impact: Medium to High
Mitigation: Escape all user inputs on output; implement Content Security Policy (CSP).



Vulnerability: Reflected Cross Site Scripting (XSS)

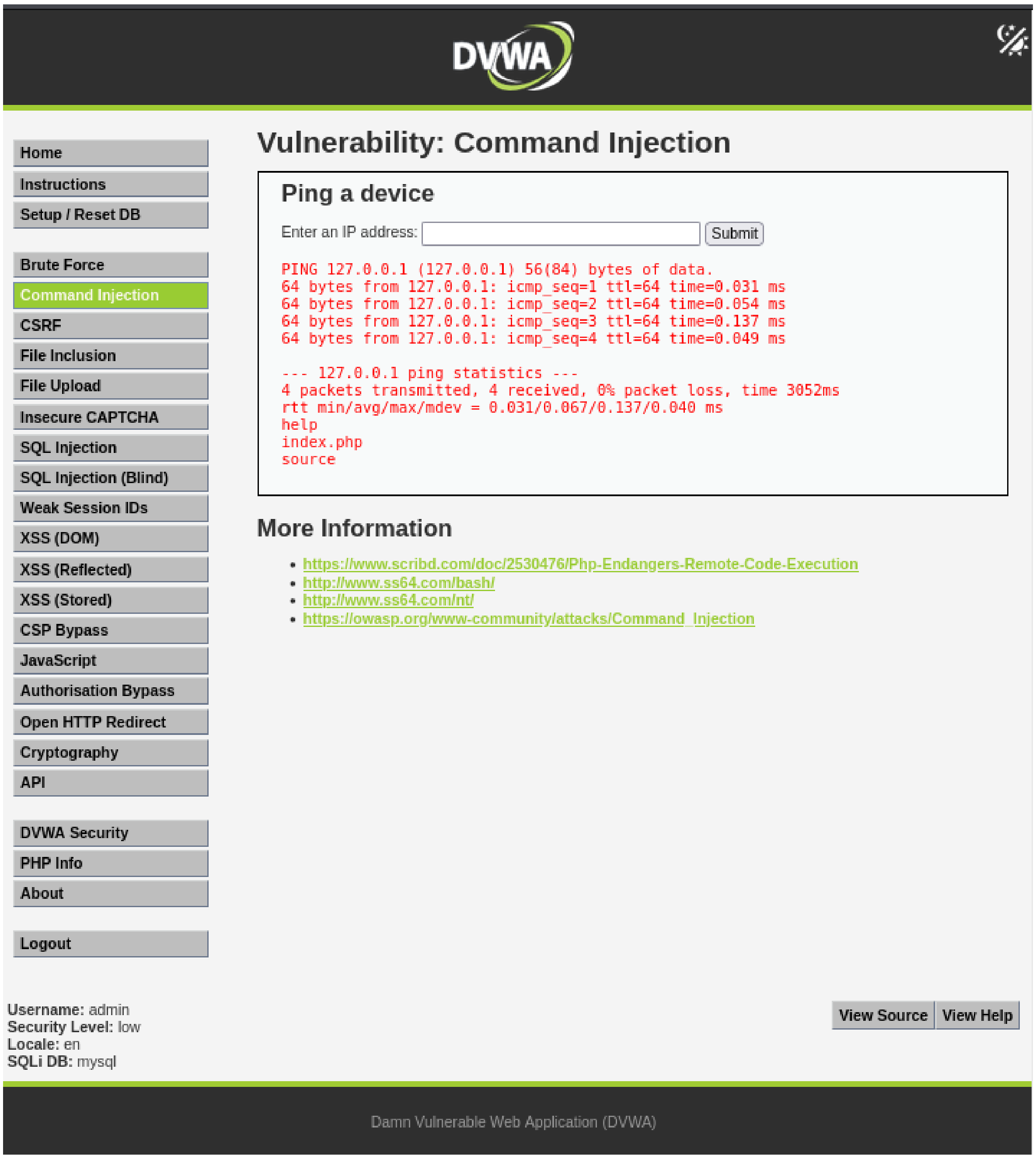
What's your name? <script>alert(1)</script>

```
Hello
```

More Information

- 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/

Vulnerability 3: Command Injection
Tested Page: DVWA → Command Injection
Test Attempt: Submitted input containing shell metacharacters via the vulnerable parameter/form field (user-supplied data was passed to a system call).
Result: The application executed system commands and returned their output, confirming unsafe handling of user input and successful command injection.
OWASP Mapping: A1: Injection (Command Injection)
Impact: High — allows execution of arbitrary system commands, data disclosure, and potential full system compromise.
Mitigation: Strictly validate and sanitize all user input (use allowlists), never pass raw user data to shell/system calls, use safe APIs instead of invoking a shell, apply least-privilege for any system processes, and deploy runtime protections such as a WAF and OS-level restrictions.



Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.031 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.137 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.031/0.067/0.137/0.040 ms
help
index.php
source
```

More Information

- https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://owasp.org/www-community/attacks/Command_Injection

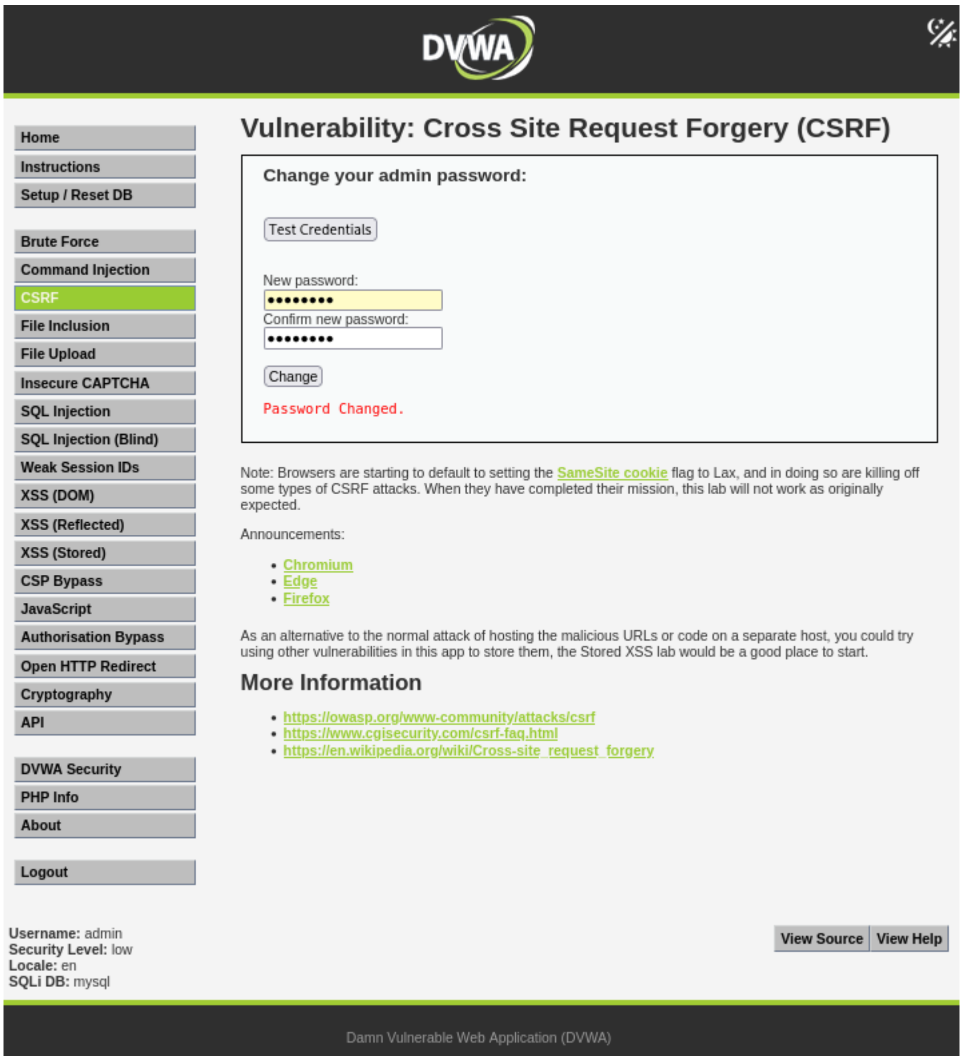
Username: admin
Security Level: low
Locale: en
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

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

Tested Page: DVWA → CSRF
Test: Changed admin password without confirming old password or session validation.
Result: Password changed silently; no CSRF token or authentication checks.

OWASP Mapping: A5: Broken Access Control / Insecure Design
Impact: High
Mitigation: Implement CSRF tokens and validate session identity for critical requests.



Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password: ••••••••
Confirm new password: ••••••••

```
Password Changed.
```

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.

Announcements:

- Chromium
- Edge
- Firefox

As an alternative to the normal attack of hosting the malicious URLs or code on a separate host, you could try using other vulnerabilities in this app to store them, the Stored XSS lab would be a good place to start.

More Information

- https://owasp.org/www-community/attacks/csrf
- https://www.cgisecurity.com/csrf-faq.html
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

Username: admin
Security Level: low
Locale: en
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

- Conclusion

The above assessment successfully demonstrated 4 core web application vulnerabilities using DVWA, mapped directly to the OWASP Top 10. This hands-on experience helped reinforce understanding of web application attack vectors and defense strategies. Each issue found includes screenshots, technical impact, and recommended mitigations.

Report Prepared by: Achyuth C.V
Cybersecurity Intern – Future Interns