Technical Report: Web Application Vulnerability Assessment Using DVWA and Wapiti

1. Introduction

This technical report presents the findings from a web application vulnerability assessment performed using Damn Vulnerable Web Application (DVWA) and the Wapiti vulnerability scanner. The objective was to simulate a Local File Inclusion (LFI) attack on DVWA and assess the overall web application security posture using an automated scan. This assessment was conducted as part of a hands-on cybersecurity internship task.

2. Test Environment

- Operating System: Windows
- Local Server Stack: XAMPP
- Vulnerable Application: DVWA (installed in E:/New folder/htdocs/dvwa/)
- Browser: Chrome
- Scanner Tool: Wapiti (installed via pip)
- LFI file tested: C:/Windows/win.ini

3. Local File Inclusion (LFI) Vulnerability Testing

DVWA was configured with its security level set to 'Low' to allow vulnerable behaviors. The File Inclusion module in DVWA was accessed and tested using crafted URL parameters. Various directory traversal patterns were attempted, and success was achieved with the absolute path to the Windows system file:

http://localhost/dvwa/vulnerabilities/fi/?page=C:/Windows/win.ini

Upon visiting the above URL, the contents of the 'win.ini' file were successfully displayed in the browser, confirming the presence of a Local File Inclusion vulnerability. Apache error logs and DVWA source code were inspected to ensure the inclusion mechanism used PHP's 'include ($file)' without sanitization.

4. Wapiti Scan Summary

Wapiti was executed against the DVWA instance using the following command:
wapiti -u http://localhost/dvwa/ -f html -o report.html

The tool successfully scanned the web application and reported several issues, including:
- CSP headers not set
- Secure flag missing on cookies
- Missing HTTP security headers like X-Content-Type-Options and X-Frame-Options

A full HTML report was saved and reviewed.

5. Remediation Recommendations

Based on the findings, the following actions are recommended:

- Sanitize all user inputs before including files
- Use allowlists for acceptable file paths
- Configure PHP to disable 'allow_url_include' and restrict 'open_basedir'
- Set security-related HTTP headers such as Content-Security-Policy (CSP), X-Frame-Options, and X-Content-Type-Options
- Ensure cookies are marked with the 'Secure' and 'HttpOnly' flags

6. Conclusion

This assessment demonstrated the practical exploitation of a Local File Inclusion vulnerability and provided an overview of web security risks highlighted by the Wapiti automated scanner. The findings reinforce the importance of secure coding practices, proper input validation, and comprehensive testing before deploying web applications to production environments.