Technical Report: Command Injection in DVWA

1. Introduction

This report details the discovery and exploitation of a **Command Injection** vulnerability using the **Damn Vulnerable Web Application (DVWA)**. The purpose of this task is to demonstrate how improperly handled user input can allow attackers to execute arbitrary system-level commands on the server hosting the web application.

2. Environment Setup

- **Operating System**: Windows
- **Web Stack**: XAMPP (Apache + MySQL + PHP)
- **Application**: DVWA (vulnerable PHP web app)
- **Installation Path**: E:/New folder/htdocs/dvwa/
- **Browser**: Chrome
- **DVWA Security Level**: Low
- **Tested Module**: Command Injection

3. Accessing the Vulnerable Module

After logging into DVWA using default credentials (admin / password), the **security level** was set to **"Low"** from the *DVWA Security* page.
From the left-hand menu, the **Command Injection** module was selected. This page presents a form where the user can enter an IP address to ping. Internally, the application runs a system-level ping command with this input.

4. Exploitation Process

Objective:

To determine if the application is vulnerable to command injection by inserting additional system commands into the input.

Payloads Tested:

For Linux/Unix systems:

127.0.0.1; ls
127.0.0.1 && whoami
127.0.0.1 || echo vulnerable

For Windows systems (XAMPP):

127.0.0.1 & dir
127.0.0.1 && whoami
127.0.0.1 | echo success

How It Works:

- The form likely runs something like:
- System ("ping -c 4 " . $_GET['ip']);
- If input is unsanitized, appending && or; allows attackers to inject extra commands.

Results:

- Commands like dir, whoami, and echo success executed successfully.
- Output was displayed in the browser below the input field, indicating that system commands were processed.

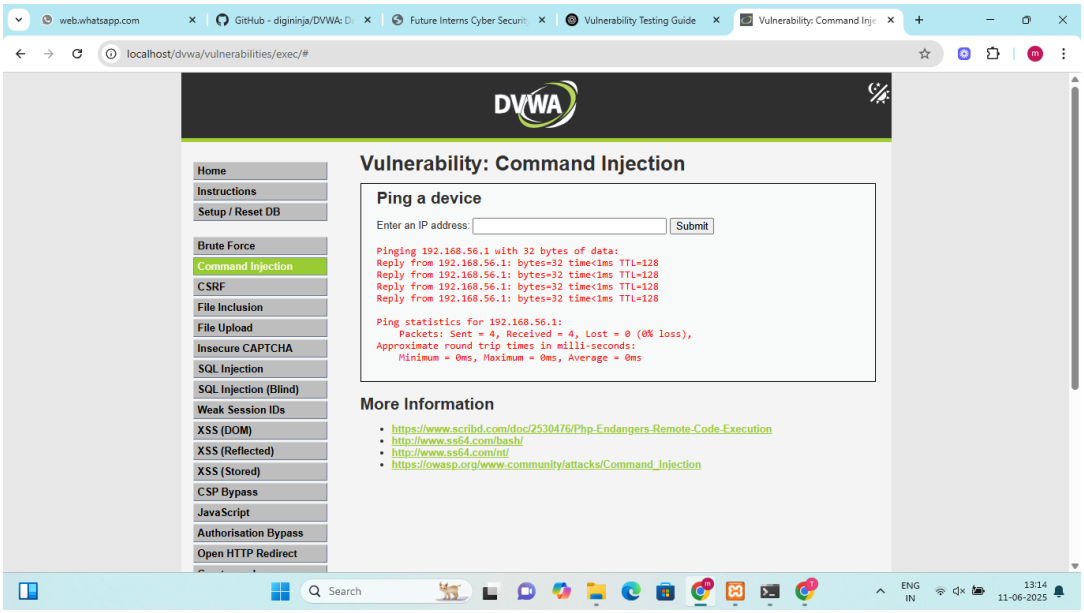
5. Screenshot Summary

Example Output:

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Directory of C:\xampp\htdocs\dvwa

01/06/2025 11:42 AM <DIR>
01/06/2025 11:42 AM <DIR>



6. Security Impact

Command Injection is a **critical vulnerability** that can allow:

- Unauthorized access to sensitive system files
- User enumeration and privilege discovery
- Full system compromise if used with nc, powershell, curl, or backdoors

7. Remediation Recommendations

To mitigate command injection:

- Never pass unsanitized input directly into system functions like system(), exec(), shell_exec(), or backticks (``).
- Use strict input validation (e.g., regex to allow only digits and dots for IP).
- Use PHP built-in validation functions or safely escape shell arguments.
- Apply least privilege principles to the web server user.
- Disable unnecessary shell access and restrict system command execution.

8. Conclusion

This exercise successfully demonstrated exploitation of a **Command Injection** vulnerability in DVWA with low security settings. The test emphasizes the importance of proper input validation and secure coding practices to prevent arbitrary command execution through user-controlled input fields.