Vulnerability Name:	Command Injection
Affected Vendor:	DVWA
Affected Product Name:	http://dvwa/vulnerabilities/exec/
Product Official Website URL	http://dvwa/login.php
Affected Components:	Affected Parameters: - Enter an IP address

Description: - A vulnerability that allows attackers to execute arbitrary system commands on a target system.

Root Cause: - Inadequate input validation and improper handling of user-controlled input in system commands.

Impact: - Unauthorized access to system resources, data leakage, system compromise, or execution of arbitrary commands.

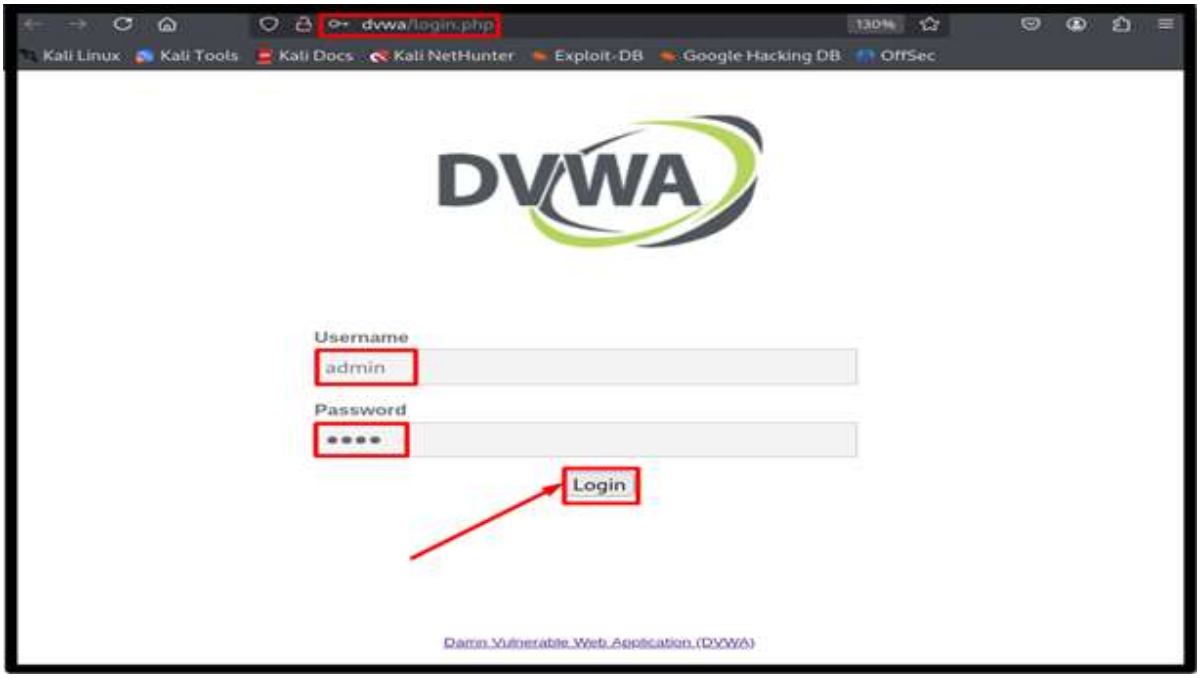
Mitigation: - Implement strict input validation, use parameterized queries or prepared statements, sanitize user input, and avoid executing system commands with user-controlled input to mitigate Command Injection.

Remediation: - To Remediate a Command Injection,

Input Validation & Sanitization – Use allowlists and reject special characters.
Avoid Direct Shell Execution – Use safe functions like subprocess. Run() instead of system().
Use Parameterized APIs – Avoid string concatenation in system commands.
Least Privilege Principle – Run applications with minimal permissions.
Web Application Firewall (WAF) – Deploy a WAF to detect and block attacks.
Secure Coding Practices – Review code, use static/dynamic analysis tools.
Monitor & Audit Logs – Track command executions for anomalies.
Conduct Penetration Testing – Regularly test with security tools (Burp Suite, OWASP ZAP).

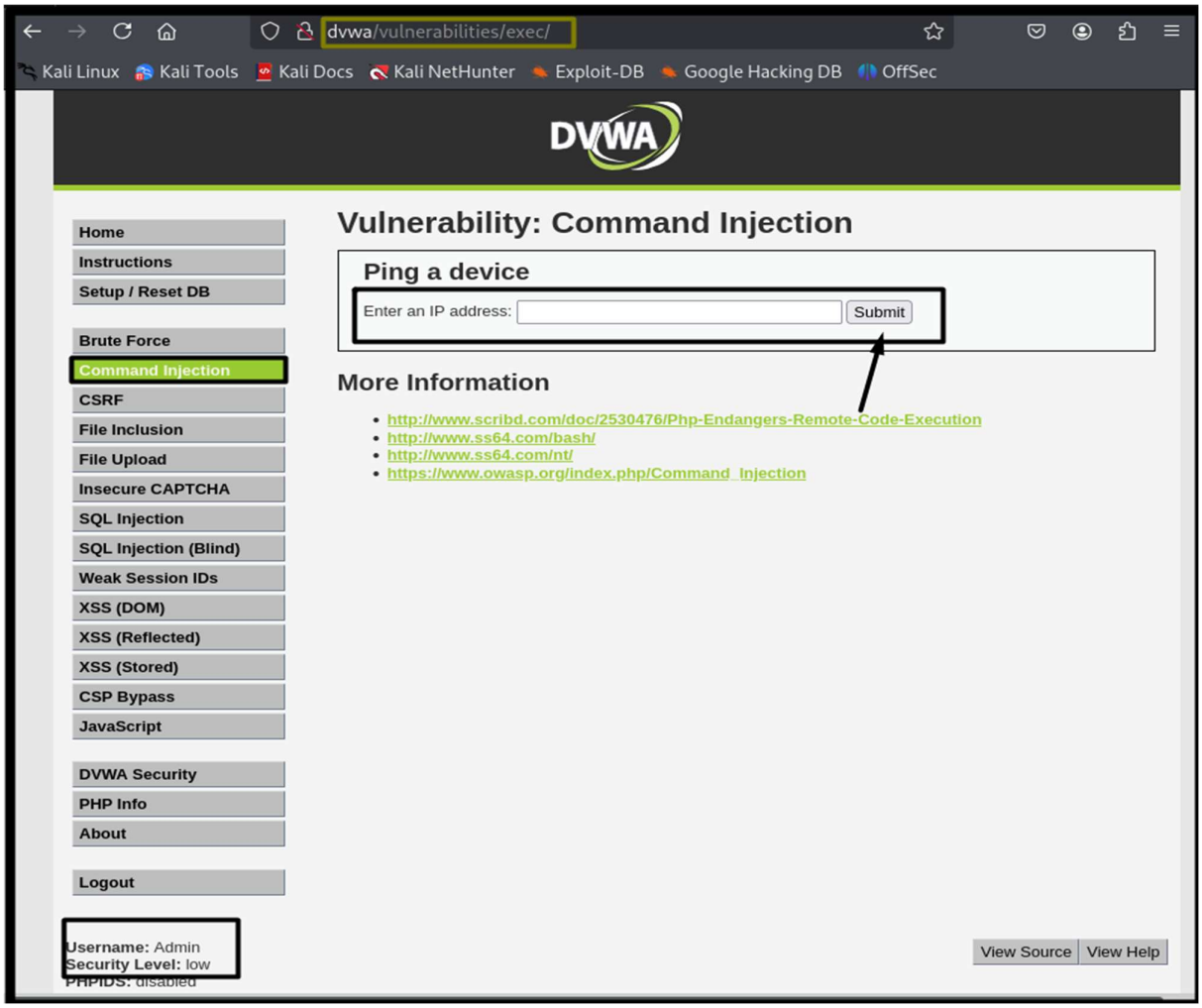
Proof of Concept

Step: -1 First navigate to http://dvwa/login.php and login with username and Password.



Step: -2 log in the home page of DVWA then click to the Command Injection Section.

SECURITY LEVEL:- Low

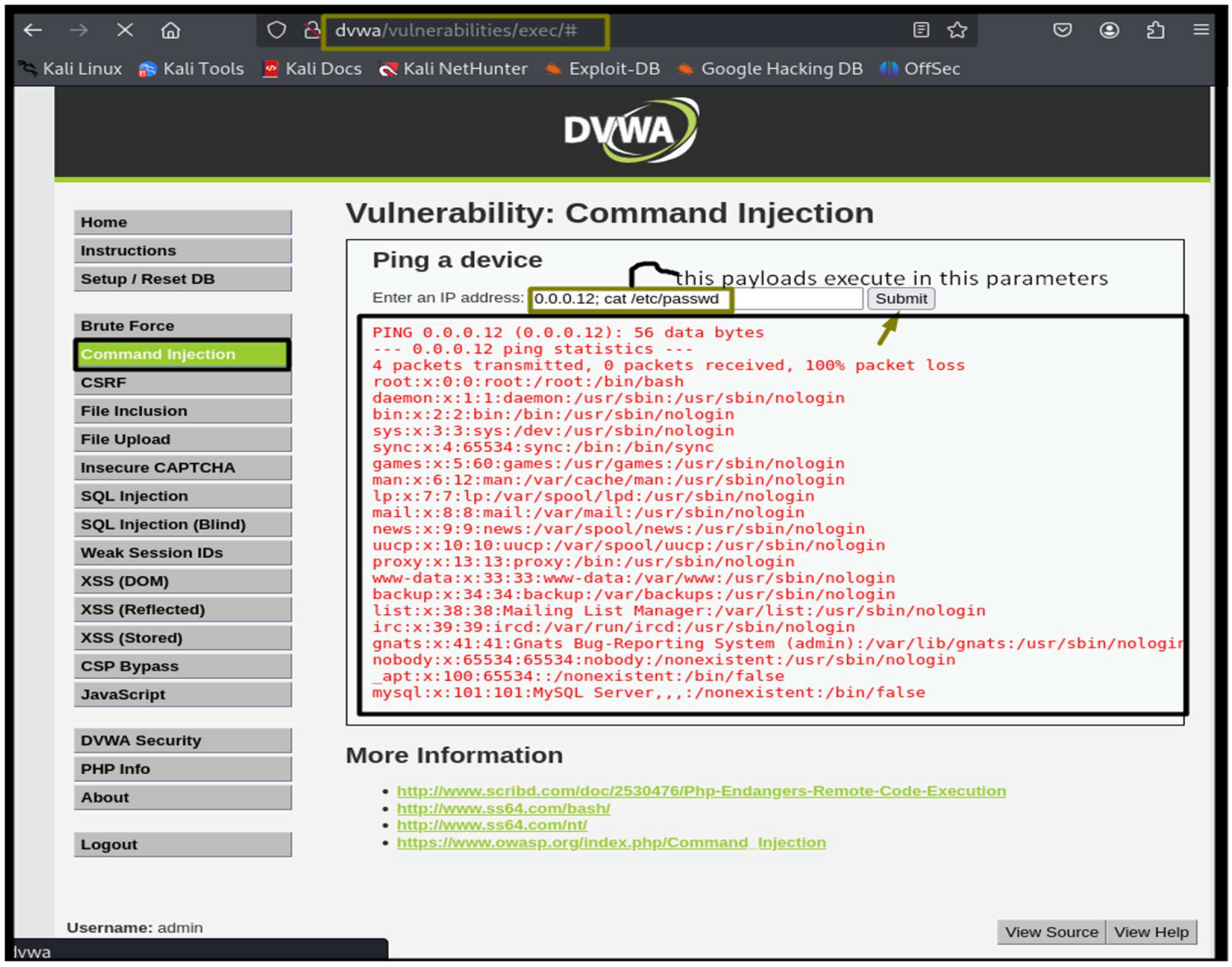


Step:-3 Check the "View Page Source" in Command Injection, and you will notice that all code lines end with a semicolon (;).

Low Command Injection Source

```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

Step:-4 In this step, I am creating payloads such as 0.0.0.12; cat /etc/passwd and 3; cat /etc/passwd. After entering the payloads, I click "Submit," which reveals all hidden password entries.



SECURITY LEVEL:- MEDIUM

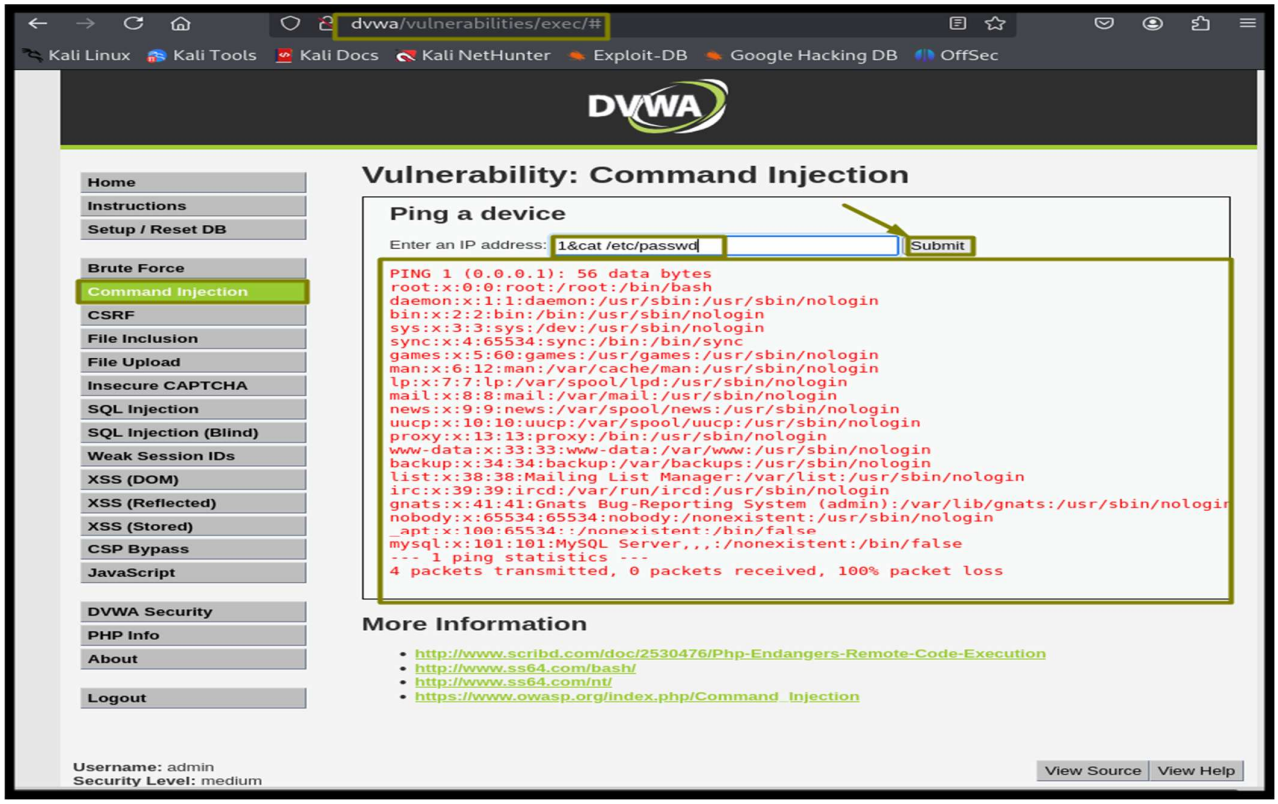
Step:- 1 Check the "View Page Source" in Command Injection, and you will notice that all code lines end with a semicolon and , (;).

Command Injection Source

vulnerabilities/exec/source/medium.php

```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '',
        ';'  => '',
    );

    // Remove any of the charactars in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

Step:-2 In this step, I am creating payloads such as 1&cat /etc/passwd and 3|cat /etc/passwd. After entering the payloads, I click "Submit," which reveals all hidden password entries.



SECURITY LEVEL:- HIGH

Step:-1 Check the "View Page Source" in Command Injection, and you will notice that all code lines end with a semicolon and ,| (;).

Command Injection Source

vulnerabilities/exec/source/high.php

```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = trim($_REQUEST[ 'ip' ]);

    // Set blacklist
    $substitutions = array(
        '&'  => '',
        ';'  => '',
        '| ' => '',
        '-'  => '',
        '$'  => '',
        '('  => '',
        ')'  => '',
        '`'  => '',
        '||' => '',
    );

    // Remove any of the charactars in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

Step:-2 In this step, I am creating payloads such as 1|cat /etc/passwd. After entering the payloads, I click "Submit," which reveals all hidden password entries.