**Command Injection**

**Descripción**

La inyección de comandos (Command Injection) es una vulnerabilidad que ocurre cuando una aplicación pasa datos no confiables suministrados por el usuario (entrada) a un intérprete del sistema operativo. Esto permite a un atacante ejecutar comandos arbitrarios en el sistema operativo del servidor que ejecuta la aplicación vulnerable.

**Ejemplo**

Considere una aplicación web que toma un nombre de dominio del usuario y ejecuta el comando ping en el servidor para verificar la conectividad:

php

Copy code

<?php

$domain = $\_GET['domain'];

$output = shell\_exec("ping -c 4 " . $domain);

echo "<pre>$output</pre>";

?>

Si un atacante proporciona una entrada como example.com; ls, el comando ejecutado sería:

bash

Copy code

ping -c 4 example.com; ls

Esto ejecutará el comando ping, seguido por el comando ls, que listará los archivos en el directorio actual del servidor.

**Impacto**

* Ejecución de comandos arbitrarios en el servidor.
* Acceso no autorizado a datos sensibles.
* Compromiso total del sistema.

**Mitigación**

* **Validar y sanitizar entradas:** Asegúrese de que las entradas del usuario se validen y saniticen adecuadamente para evitar la ejecución de comandos.
* **Usar APIs seguras:** Use funciones del sistema operativo que no invoquen el intérprete de comandos directamente, como exec y shell\_exec en PHP. En su lugar, use funciones más seguras como escapeshellarg y escapeshellcmd para sanitizar entradas.
* **Principio de privilegios mínimos:** Asegúrese de que la aplicación web y el usuario del sistema operativo bajo el cual se ejecuta la aplicación tengan los privilegios mínimos necesarios.

**Ejemplo de Mitigación en PHP**

En lugar de usar shell\_exec, use una función que escape los argumentos del shell:

php

Copy code

<?php

$domain = escapeshellarg($\_GET['domain']);

$output = shell\_exec("ping -c 4 " . $domain);

echo "<pre>$output</pre>";

?>

En este ejemplo, escapeshellarg asegura que el dominio proporcionado por el usuario se maneje de manera segura, escapando cualquier carácter que podría ser usado para ejecutar comandos adicionales.

**Ejemplo de Mitigación en Python**

En Python, use el módulo subprocess con subprocess.run y pase los argumentos como una lista:

python

Copy code

import subprocess

domain = request.args.get('domain')

# Sanitizar entrada

sanitized\_domain = domain.replace(';', '').replace('&', '')

result = subprocess.run(['ping', '-c', '4', sanitized\_domain], capture\_output=True, text=True)

print(result.stdout)

**Resumen**

Command Injection es una vulnerabilidad crítica que puede comprometer completamente un sistema. La validación y sanitización adecuadas de las entradas del usuario, junto con el uso de funciones seguras del sistema operativo, son esenciales para prevenir estos ataques.