Command Injection

Software Security

CWEs

- 77 – Failure to Sanitize Data into a Control Plane
- 78 – Improper Neutralization of Special Elements used in an OS Command
- 88 – Improper Neutralization of Argument Delimiters in a Command (‘Argument Injection’)

The Problem

- Untrusted data passed to some compiler or interpreter

Classic Example

- User input passed to a call to the system's command line
  - Or a variant of that
- A short uncomprehensive list…
- In C…
  - `system();`
  - `system("ls -la");`
- In Python…
  - `os.system("ls -la")`
  - `subprocess.call("ls -la", shell=True)`

Building out Commands

- Attackers can use certain characters to chain on additional commands
  - Semicolon – ends a statement on UNIX systems
  - Backtick – data between backticks are executed
  - Vertical bar (pipe) – chains another command with the output of the previous process

Discovery

- Commands or control information are mingled with data
- Presence of the following functions
  - C: system, popen, execlp, execvp
  - C (Windows): ShellExecute
  - Python: exec, eval, os.system/popen, subprocess.Popen, execfile
    - Python2 – input evaluates user input as code

https://wiki.owasp.org/index.php/Testing_for_Command_Injection_(OTG-INPVAL-013)

Defense and Remediation

- Don’t use OS commands directly – especially if using user input within the argument list
  - Most common commands have equivalent functions
- Escape values added to OS commands
  - Ex: escapeshellarg(), escapeshellcmd() PHP shlex.quote() Python
- Whitelist allowed characters and commands
- Always use rule of least privilege
- Use taint defenses if your platform supports them

https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html

References

- https://Wikipedia.org
- 24 Deadly Sins of Software Security
  - ISBN-13: 978-0071626750