

wing3e / public\_exp Public[Code](#) [Issues 35](#) [Pull requests](#) [Actions](#) [Projects](#) [Security and quality](#)[New issue](#)

# Command Injection Vulnerability in Vale-MCP #27

[Open](#)

BruceJqs opened 2 weeks ago



## Command Injection Vulnerability in Vale-MCP

### 1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: March 17, 2026

### 2) Reporter Contact (fill before submit)

- Reporter name: BruceJin
- Reporter email: brucejin@zju.edu.cn
- Permission to share contact with vendor: Yes

### 3) Vendor / Product Identification

- Vendor: ChrisChinchilla
- Product: vale-mcp
- Repository: <https://github.com/ChrisChinchilla/Vale-MCP>
- Affected component(s):
  - src/index.ts
  - src/vale-runner.ts

## 4) Vulnerability Type

---

- CWE: CWE-78 (OS Command Injection)
- Short title: OS command injection in MCP/HTTP request handling

## 5) Affected Versions

---

- Confirmed affected: 0.1.0
- Suspected affected range: revisions containing the same request-to-sink flows listed below
- Fixed version: Not available at time of report (March 17, 2026)

## 6) Vulnerability Description

---

A command injection vulnerability (CWE-78) has been identified in vale-mcp, specifically within the vale-runner.ts component. An attacker with network access to the MCP/HTTP interface can supply maliciously crafted input through the config\_path or other tool arguments, which flows unsanitized into OS command execution via exec or execAsync. This allows arbitrary system commands to be executed with the privileges of the server process, leading to full host compromise, including data exposure, integrity loss, and potential service disruption. Versions up to and including 0.1.0 are confirmed affected.

## 7) Technical Root Cause

---

1. `js/command-injection-from-request`
  - Source: `src/index.ts:278 (request)`
  - Sink: `src/vale-runner.ts:309`
  - Sink code: `const result = await execAsync(command, execOptions);`
2. `js/command-injection-from-request`
  - Source: `src/index.ts:278 (request)`
  - Sink: `src/vale-runner.ts:389`
  - Sink code: `const child = exec(command, execOptions, (error, stdout, stderr) => {`

## 8) Attack Prerequisites

---

- Attacker can invoke the MCP/HTTP endpoint or tool handler that reaches the vulnerable sink.
- No effective runtime policy strips or constrains attacker-controlled values before sink usage.
- If SSRF applies: server has network egress to attacker-chosen or internal targets.

## 9) Proof of Concept / Reproduction Guidance

---

This proof of concept provides a concise, CVE-style reproduction example for the reported issue.

1. Reproduction request

```
{"jsonrpc": "2.0", "id": 1, "method": "tools/call", "params": {"name": "check_file", "arguments": ...}}
```

```
{"jsonrpc": "2.0", "id": 1, "method": "tools/call", "params": {"name": "check_text", "arguments": ...}}
```

## 10) Security Impact

- Confidentiality: High (host/system data exposure possible).
- Integrity: High (command execution may alter server state).
- Availability: High (service disruption via command abuse possible).
- Scope: Changed.

## 11) CVSS v3.1 Suggestion

- Suggested vector: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H`
- Suggested base score: 10.0 (Critical)
- Adjust `PR` upward if the vulnerable tools are strictly admin-only and strongly authenticated.

## 12) Workarounds / Mitigations

- Remove direct shell-string execution from request-driven paths.
- Replace free-form commands with fixed allowlists and validated argument schemas.
- Prefer argument-array process execution without shell interpretation.
- Add authentication, authorization, logging, and rate limiting on sensitive MCP/HTTP handlers.

## 13) Recommended Fix

- Eliminate the request-to-sink data flow documented above.
- Add input schema validation at MCP/HTTP boundaries.
- Add regression tests proving attacker-controlled values cannot reach sensitive sinks.
- Publish a maintainer security advisory once a patch is released.

## 14) References

- Repository: <https://github.com/ChrisChinchilla/Vale-MCP>
- Reviewed source file: `src/index.ts`
- Reviewed source file: `src/vale-runner.ts`
- CWE-78: <https://cwe.mitre.org/data/definitions/78.html>

## 15) Credits

- Discoverer: BruceJin
- Discovery method: Static analysis (CodeQL) plus repository source-code audit

## 16) Additional Notes for Form Mapping

- Audit verdict: Likely exploitable: command injection path reaches OS execution sink.
- Dynamic exploit replay status: not completed in this batch run.
- Maintainer should validate release mapping before coordinated disclosure.

The screenshot shows a web interface for configuring and running a tool named 'check\_file'. On the left, there is a 'Tools' sidebar with a search bar and a 'Clear' button. Below it, several tool descriptions are listed, including 'vale\_status', 'vale\_sync', 'check\_file', and 'check\_text'. The 'check\_file' tool description states: 'Lint a file at a specific path against Vale style rules. Returns issues found with their locations and severity. If Vale is not installed, returns error with installation guidance.'

The main area shows the configuration for 'check\_file'. It has a 'path' field containing 'LICENSE' and a 'config\_path' field containing '.vale.ini', touch poc.txt;#. Below the configuration, there are buttons for 'Run Tool' and 'Copy Input'. The 'Tool Result' is 'Success'. The 'Meta' section shows a JSON output:

```
{
  structured_data: {
    file: ".LICENSE"
    issues: []
    summary: {
      total: 0
      errors: 0
      warnings: 0
      suggestions: 0
    }
  }
}
```

At the bottom, two terminal screenshots are shown. The first terminal shows the command 'ls poc.txt' resulting in the error 'ls: poc.txt: No such file or directory'. The second terminal shows the command 'ls poc.txt' resulting in the output '-rw-r--r-- 1 bruce staff 0B Mar 22 21:44 poc.txt', indicating the file has been successfully created.

After executing command via the command injection vulnerability, the file "poc.txt" has been successfully created.

### Tools

List Tools

Clear

- vale\_status**  
Check if Vale (vale.sh) is installed and accessible. Use this first if other Vale tools fail. Returns installation status, version if available, and installation instructions for the current platform.
- vale\_sync**  
Download Vale styles and packages by running 'vale sync'. Use this when you see errors about missing styles directories (E100 errors like "The path does not exist"). This command reads the .vale.ini configuration and downloads the required...
- check\_file**  
Lint a file at a specific path against Vale style rules. Returns issues found with their locations and severity. If Vale is not installed, returns error with installation guidance.
- check\_text**  
Lint text content directly against Vale style rules without requiring a file. Useful for checking text snippets, clipboard content, or dynamically generated content. Returns issues found with their locations and severity.

### check\_text

Lint text content directly against Vale style rules without requiring a file. Useful for checking text snippets, clipboard content, or dynamically generated content. Returns issues found with their locations and severity.

Read-only  Destructive  Idempotent  Open-world

text \*

abc

text\_file\_ext

txt

config\_path

.vale.ini,touch poc.txt,#

**Tool-specific Metadata:** Add Pair

No metadata pairs.

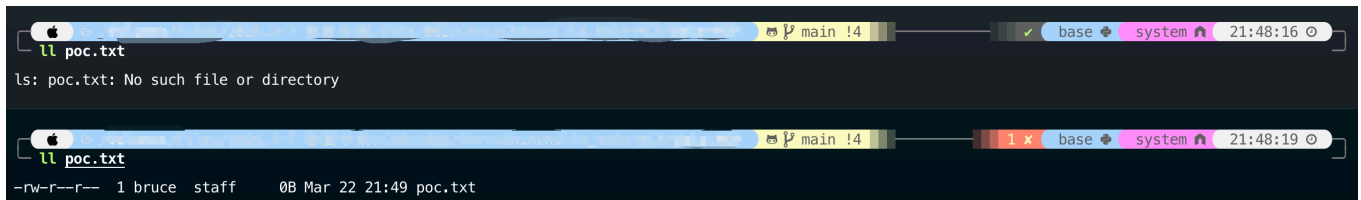
**Tool Result: Success**

Meta:

```
{
  structured_data: {
    file: "stdin"
    issues: []
    summary: {
      total: 0
      errors: 0
      warnings: 0
      suggestions: 0
    }
  }
}
```

**✔️ \*\*No style issues found!❗️**

The text looks good according to Vale style rules."



After executing command via the command injection vulnerability, the file "poc.txt" has been successfully created.

to join this conversation on [GitHub](#). Already have an account? [Sign in to comment](#)

## Metadata

**Assignees**

No one assigned

---

**Labels**

No labels

---

**Projects**

No projects

---

**Milestone**

No milestone

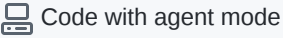

---

**Relationships**

None yet

---

**Development**

 Code with agent mode 

No branches or pull requests

---

**Participants**

