New issue

simple-openstack-mcp OS Command Injection via shell metacharacter bypass #3

Open

wing3e opened 3 weeks ago

simple-openstack-mcp OS Command Injection via shell metacharacter bypass

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: April 2, 2026

2) Reporter Contact (fill before submit)

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

3) Vendor / Product Identification

- Vendor: choieastsea
- Product: simple-openstack-mcp
- Repository: https://github.com/choieastsea/simple-openstack-mcp
- Reviewed local source path: `datasets_set/003/choieastsea-simple-openstack-mcp`
- Affected component(s):
- `server.py`

- `commander.py`

4) Vulnerability Type

- CWE: CWE-78 (OS Command Injection)
- Short title: simple-openstack-mcp OS Command Injection via shell metacharacter bypass

5) Affected Versions

- Confirmed affected: `767b2f4`
- Suspected affected range: versions containing the same input-to-sink flow documented below
- Fixed version: Not available at time of report (April 2, 2026)

6) Vulnerability Description

The `exec_openstack` MCP tool only checks that input starts with `openstack`, then passes the full string to `subprocess.run(..., shell=True)`. An attacker can append shell metacharacters (for example `;`) to execute arbitrary OS commands under the service account.

7) Technical Root Cause

1. Input validation in `server.py` is prefix-only (`startswith('openstack')`) and does not neutralize shell metacharacters.
2. `OpenStackCommander.execute` in `commander.py` invokes `subprocess.run(command, shell=True, ...)` with attacker-controlled `command`.
3. No allowlist parsing/tokenization is applied before execution.

```
# server.py
@mcp.tool()
def exec_openstack(cmd: str) -> str:
    if not cmd.strip().startswith("openstack"):
        raise ValueError("Command must start with 'openstack'")
    return OpenStackCommander.execute(cmd)

# commander.py
result = subprocess.run(
    command, shell=True, capture_output=True, text=True, timeout=timeout
)
```

8) Attack Prerequisites

- Attacker can send requests to the vulnerable endpoint or MCP tool interface.
- Deployment does not enforce compensating controls that block the demonstrated payload.

- Vulnerable code path remains enabled in runtime configuration.

9) Proof of Concept / Reproduction Guidance

1. Start the vulnerable server.
2. Invoke MCP tool `exec_openstack` with a metacharacter payload.
3. Verify OS command side effect.

PoC request (JSON-RPC):

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/call",
  "params": {
    "name": "exec_openstack",
    "arguments": {
      "cmd": "openstack --help; touch /tmp/choieastsea_cmdinj_poc"
    }
  }
}
```

Verification:

```
ls -l /tmp/choieastsea_cmdinj_poc
```

10) Security Impact

- Confidentiality: High (read local secrets/configuration via arbitrary command execution).
- Integrity: High (modify files or runtime state).
- Availability: High (service disruption or destructive command execution).
- Scope: Unchanged to Changed depending on surrounding deployment topology.

11) CVSS v3.1 Suggestion

- Suggested vector: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (9.8, if exposed to untrusted callers)`
- Final score should be adjusted by CNA/vendor based on real deployment exposure.

12) Workarounds / Mitigations

- Restrict access to vulnerable interfaces (network ACL, authn/authz, mTLS).
- Reject untrusted metacharacters/URLs and enforce strict server-side allowlists.

- Add regression tests for all PoC payload patterns included in this report.

13) Recommended Fix

- Remove direct execution/fetch of attacker-controlled values.
- Use safe APIs (`subprocess.run` with argument arrays and `shell=False`; strict URL policy checks for outbound requests).
- Enforce endpoint-level authentication and authorization before reaching sensitive sinks.

14) References

- Repository: https://github.com/choieastsea/simple-openstack-mcp
- Reviewed source location: `datasets_set/003/choieastsea-simple-openstack-mcp`
- SARIF evidence references:
- `py/mcp-command-line-injection` at `commander.py:9`

15) Credits

- Discoverer: `Winegee`
- Discovery method: Static analysis (CodeQL/SARIF) plus source-code audit

16) Additional Notes for Form Mapping

- Issue status at report time: source-code confirmed in local dataset and exploitation path documented with explicit PoC.
- Dynamic verification was represented through deterministic command/URL payloads suitable for lab reproduction.
- Version-range precision should be finalized by maintainer release history before disclosure.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants