New issue

OS Command Injection Vulnerability in @burtthecoder/mcp-dnstwist #13

Open

BruceJqs opened 2 weeks ago

OS Command Injection Vulnerability in @burtthecoder/mcp-dnstwist

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: Apr 14, 2026

2) Reporter Contact

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

3) Vendor / Product Identification

- Vendor: BurtTheCoder
- Product: @burtthecoder/mcp-dnstwist
- Repository: https://github.com/BurtTheCoder/mcp-dnstwist
- Affected component(s):
- `src/index.ts`

4) Vulnerability Type

- CWE: CWE-78 (Improper Neutralization of Special Elements used in an OS Command)
- Short title: OS command injection in fuzz_domain command construction

5) Affected Versions

- Confirmed affected: 1.0.4
- Suspected affected range: revisions containing the same request-to-shell-command flow listed below
- Fixed version: Not available at time of report

6) Vulnerability Description

An OS command injection vulnerability (CWE-78) has been identified in @burtthecoder/mcp-dnstwist version 1.0.4, specifically within the fuzz_domain MCP tool in src/index.ts. The tool accepts user-controlled parameters such as nameservers, joins them into a shell command string using args.join(' '), and executes the resulting string via child_process.exec without shell escaping or argument-vector separation. An attacker with network access to the MCP interface can inject shell metacharacters into parameters like nameservers to execute arbitrary operating system commands with the privileges of the server process, leading to full host compromise, including data exposure, integrity loss, and service disruption. No fixed version is available at the time of reporting.

7) Technical Root Cause

1. `js/command-injection-from-request`
   - Source: `src/index.ts:176` (`request`)
   - Source argument extraction: `src/index.ts:194` to `src/index.ts:203`
   - Propagation: `src/index.ts:212` (`args.push('--nameservers', nameservers);`)
   - Propagation: `src/index.ts:220` (`args.join(' ')`)
   - Sink: `src/index.ts:82`
   - Sink code: `const result = await execAsync(command, { maxBuffer: 10 * 1024 * 1024 });`

8) Attack Prerequisites

- Attacker can invoke the `fuzz_domain` MCP tool through the MCP interface, for example through mcp-inspector or an exposed MCP client/server integration.
- Docker is installed and available to the MCP server process, because the vulnerable tool runs `docker run`.
- The attacker can supply a crafted string to `nameservers`.
- No effective shell escaping, allowlist validation, or argument-vector execution prevents attacker-controlled values from reaching `child_process.exec`.

9) Proof of Concept / Reproduction Guidance

This proof of concept uses `id` and base64 encoding to make command execution visible while keeping the tool output parseable as JSON.

1. Start the MCP server in mcp-inspector from the repository root after installing dependencies and building the project.

```
npm install
npm run build
npx @modelcontextprotocol/inspector node build/index.js
```

2. Invoke the `fuzz_domain` tool with the following arguments.

```
{
  "domain": "example.com",
  "nameservers": "1.1.1.1 >/dev/null 2>&1; printf '[{\"domain\":\"'; id | base64 | tr -d '[:s
  "threads": 1,
  "format": "json",
  "registered_only": false,
  "mxcheck": false,
  "banners": false
}
```

3. Validation

- The injected shell command runs on the MCP server host.
- The Inspector response contains a JSON object whose `domain` value is base64-encoded output from `id`.
- Decode the returned value locally with `base64 -d` to confirm it contains the UID/GID of the MCP server process.

10) Security Impact

- Confidentiality: High (attacker-controlled commands can read files and environment data accessible to the MCP server process).
- Integrity: High (attacker-controlled commands can modify files or execute further payloads with the MCP server process privileges).
- Availability: High (attacker-controlled commands can terminate processes, remove writable data, or exhaust resources).
- Scope: Unchanged.

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`
- Suggested base score: 9.8 (Critical)
- Adjust `PR` upward if the vulnerable MCP tool is strictly authenticated and only available to lower-privileged users.

12) Workarounds / Mitigations

- Do not expose this MCP server to untrusted clients until the command construction is fixed.
- Restrict access to the `fuzz_domain` tool to trusted users only.
- Apply strict allowlist validation to `domain`, `nameservers`, and numeric option values before command execution.
- Avoid passing attacker-controlled strings to shell-interpreted command lines.

13) Recommended Fix

- Replace `child_process.exec(command)` with an argument-vector API such as `child_process.spawn` or `child_process.execFile` with `shell: false`.
- Build Docker arguments as an array, for example `['run', '--rm', DOCKER_IMAGE, domain, '--format', format, '--nameservers', nameservers, '-t', threads.toString()]`, rather than joining user-controlled values into a shell string.
- Validate `nameservers` with a strict allowlist, such as comma-separated IP addresses or trusted DNS server hostnames, and reject shell metacharacters.
- Add regression tests proving injected values such as `; id; #` are passed as literal arguments or rejected before execution.
- Publish a maintainer security advisory once a patch is released.

14) References

- Repository: https://github.com/BurtTheCoder/mcp-dnstwist
- Reviewed source file: `src/index.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 and dynamic reproduction

16) Additional Notes for Form Mapping

- Audit verdict: Exploitable: attacker-controlled MCP tool arguments reach a shell command sink.
- Dynamic exploit replay status: reproduced successfully with mcp-inspector using the `id` base64 proof of concept.
- Maintainer should validate release mapping before coordinated disclosure.

For furthermore information, please refer to BruceJqs/public_exp#22

Sign up for free **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

No branches or pull requests

Participants