OWASP MCP Top 10

Compliance assessment — 2025

Server: io.github.getmcpm/cli

Slug: io-github-getmcpm-cli
Scan id: 4a7aa6fd-387c-48c6-acd5-1ff2477ec5db

Assessed at: 2026-05-24 07:36:16 UTC
Sentinel version: 0.4.0
Rules version: 2026-04-23

' Non-compliant

DRAFT for review — not legal advice. See attestation block for verification instructions.

DRAFT

Table of contents

1. Executive summary
2. Coverage & transparency
3. Controls summary
4. Control details
5. Multi-step attack chains
6. Cryptographic attestation

1. Executive summary

Assessment of io.github.getmcpm/cli against OWASP MCP Top 10: overall status non compliant. Of 10 controls, 0 met, 3 unmet, 0 partial, 7 not assessed, 0 not applicable. 3 control(s) had complete mapped-assessor execution evidence; 7 lacked sufficient execution evidence and 0 were outside the current assessor mapping. Unmet controls have findings at or above the framework's mandatory severity threshold and should be remediated before relying on this server in a regulated deployment. All claims are traceable to individual finding rows via finding_id and to the governing rule via rule_id; the enclosing signed envelope commits MCP Sentinel to the exact bytes of this report.

2. Coverage & transparency

Coverage band: medium

Coverage ratio: 66%

Rules version: 2026-04-23

Analysis techniques applied:

- ast-taint
- composite
- cross-module
- dependency-audit
- entropy
- similarity
- structural
- stub

3. Controls summary

ControlName	Status	Evidence
MCP01Prompt Injection?	Not	

assessed0
MCP02Tool Poisoning' Unmet1
MCP03 Command Injection? Not assessed0
MCP04Data Exfiltration? Not assessed0
MCP05 Privilege Escalation? Not assessed0
MCP06 Excessive Permissions? Not assessed0
MCP07 Insecure Configuration ? Not assessed0
MCP08 Dependency Vulnerabilities' Unmet1
MCP09 Logging & Monitoring Failures? Not assessed0
MCP10Supply Chain Compromise' Unmet2

4. Control details

MCP01 — Prompt Injection

? Not assessed

0/24 mapped assessor rule(s) are proven to have executed; no adverse findings were observed, but the control is not assessed because execution evidence is incomplete. Unexecuted assessors: A1 (execution-ledger); A5 (execution-ledger); A7 (execution-ledger); A8 (execution-ledger); A9 (execution-ledger); B5 (execution-ledger); F1 (execution-ledger); F6 (execution-ledger); G1 (execution-ledger); G2 (execution-ledger); G3 (execution-ledger); G4 (execution-ledger); G5 (execution-ledger); H2 (execution-ledger); H3 (execution-ledger); I3 (execution-ledger); I6 (execution-ledger); I7 (execution-ledger); J3 (execution-ledger); J5 (execution-ledger); J6 (execution-ledger); N4 (execution-ledger); N9 (execution-ledger); N12 (execution-ledger).

MCP02 — Tool Poisoning

' Unmet

0/15 mapped assessor rule(s) are proven to have executed; 1 finding(s) observed (1 critical); at least one finding is at or above the high threshold (status: unmet). 15 mapped assessor(s) did not execute, so this finding-backed result does not imply complete control coverage.

Evidence:

[Critical] Official Namespace Squatting (finding 7b01e6a9-b4fb-474e-abad-4cd8ed6ceba8, confidence 90%)

SOURCE: external-content at initialize.server_name — The MCP client surfaces the server name verbatim in its approval dialog, and the LLM ingests the server name alongside the tool descriptions. A nam

Required mitigations:

- If you own the server and are NOT affiliated with the vendor whose namespace it contains, rename the server to remove the vendor token. Choose a name that makes your actual publisher identity clear. If you ARE a vendor-approved partner and intentionally use the vendor's namespace, request inclusion in the rule's OFFICIAL_NAMESPACES.verified_github_orgs list by publishing the server under a vendor-sanctioned GitHub organisation. Users deciding whether to approve the server should check the repository owner against the vendor's published list of approved partners before granting trust.

MCP03 — Command Injection

? Not assessed

0/6 mapped assessor rule(s) are proven to have executed; no adverse findings were observed, but the control is not assessed because execution evidence is incomplete. Unexecuted assessors: C1 (execution-ledger); C9 (execution-ledger); C13 (execution-ledger); C16 (execution-ledger); J2 (execution-ledger); J7 (execution-ledger).

MCP04 — Data Exfiltration

? Not assessed

0/15 mapped assessor rule(s) are proven to have executed; no adverse findings were observed, but the control is not assessed because execution evidence is incomplete. Unexecuted assessors: A3 (execution-ledger); F1 (execution-ledger); F3 (execution-ledger); F7 (execution-ledger); G7 (execution-ledger); I9 (execution-ledger); I13 (execution-ledger); K18 (execution-ledger); O4 (execution-ledger); O5 (execution-ledger); O6 (execution-ledger); O8 (execution-ledger); O9 (execution-ledger); O10 (execution-ledger); P3 (execution-ledger).

MCP05 — Privilege Escalation

? Not assessed

0/7 mapped assessor rule(s) are proven to have executed; no adverse findings were observed, but the control is not assessed because execution evidence is incomplete. Unexecuted assessors: C2 (execution-ledger); C8 (execution-ledger); C10 (execution-ledger); C12 (execution-ledger); I4 (execution-ledger); I12 (execution-ledger); J1 (execution-ledger).

MCP06 — Excessive Permissions

? Not assessed

0/7 mapped assessor rule(s) are proven to have executed; no adverse findings were observed, but the control is not assessed because execution evidence is incomplete. Unexecuted assessors: A2 (execution-ledger); B3 (execution-ledger); B7 (execution-ledger); E4 (execution-ledger); F2 (execution-ledger); I11 (execution-ledger); I16 (execution-ledger).

MCP07 — Insecure Configuration

? Not assessed

0/11 mapped assessor rule(s) are proven to have executed; no adverse findings were observed, but the control is not assessed because execution evidence is incomplete. Unexecuted assessors: B6 (execution-ledger); C7 (execution-ledger); C8 (execution-ledger); C11 (execution-ledger); C14 (execution-ledger); C15 (execution-ledger); D6 (execution-ledger); E1 (execution-ledger); E2 (execution-ledger); I15 (execution-ledger); J4 (execution-ledger).

MCP08 — Dependency Vulnerabilities

' Unmet

0/7 mapped assessor rule(s) are proven to have executed; 1 finding(s) observed (1 high); at least one finding is at or above the high threshold (status: unmet). 7 mapped assessor(s) did not execute, so this finding-backed result does not imply complete control coverage.

Evidence:

[High]
Typosquatting
Risk in
Dependencies
(finding b440567
4-69f5-4f4d-991d
-3479401b2d19,
confidence 88%)

SOURCE: external-content at pypi:yaml@2.9.0 — Dependency names are external content resolved from public package registries. A near-miss to a popular canonical name is a supply-chain anomaly under ISO

Required mitigations:

- Verify that the flagged dependency is the package you intended to install. Open the registry page for the candidate and compare publisher, publish date, download count, and postinstall scripts against the target. If it is not the intended package, replace it with the legitimate target, regenerate the lockfile, and audit the install environment (CI and developer machines) for any artifacts the malicious package may have written. Adopt a typosquat-aware package firewall (Socket.dev, Snyk Advisor, GitHub Dependabot) that rejects near-miss names at install time, in line with ISO 27001 A.5.21 supply-chain controls.

MCP09 — Logging & Monitoring Failures

? Not assessed

0/6 mapped assessor rule(s) are proven to have executed; no adverse findings were observed, but the control is not assessed because execution evidence is incomplete. Unexecuted assessors: C6 (execution-ledger); E3 (execution-ledger); K1 (execution-ledger); K2 (execution-ledger); K3 (execution-ledger); K20 (execution-ledger).

MCP10 — Supply Chain Compromise

' Unmet

0/27 mapped assessor rule(s) are proven to have executed; 2 finding(s) observed (1 high, 1 critical); at least one finding is at or above the high threshold (status: unmet). 27 mapped assessor(s) did not execute, so this finding-backed result does not imply complete control coverage.

Evidence:

[High]
Typosquatting
Risk in
Dependencies
(finding b440567
4-69f5-4f4d-991d
-3479401b2d19,
confidence 88%)

SOURCE: external-content at pypi:yaml@2.9.0 — Dependency names are external content resolved from public package registries. A near-miss to a popular canonical name is a supply-chain anomaly under ISO

[Critical]
Official
Namespace
Squatting

(finding 7b01e6a9-b4fb-474e-abad-4cd8ed6ceba8, confidence 90%)

SOURCE: external-content at initialize.server_name — The MCP client surfaces the server name verbatim in its approval dialog, and the LLM ingests the server name alongside the tool descriptions. A nam

Required mitigations:

- Verify that the flagged dependency is the package you intended to install. Open the registry page for the candidate and compare publisher, publish date, download count, and postinstall scripts against the target. If it is not the intended package, replace it with the legitimate target, regenerate the lockfile, and audit the install environment (CI and developer machines) for any artifacts the malicious package may have written. Adopt a typosquat-aware package firewall (Socket.dev, Snyk Advisor, GitHub Dependabot) that rejects near-miss names at install time, in line with ISO 27001 A.5.21 supply-chain controls.
- If you own the server and are NOT affiliated with the vendor whose namespace it contains, rename the server to remove the vendor token. Choose a name that makes your actual publisher identity clear. If you ARE a vendor-approved partner and intentionally use the vendor's namespace, request inclusion in the rule's OFFICIAL_NAMESPACES.verified_github_orgs list by publishing the server under a vendor-sanctioned GitHub organisation. Users deciding whether to approve the server should check the repository owner against the vendor's published list of approved partners before granting trust.

5. Multi-step attack chains

No multi-step attack chains were synthesized for this server.

6. Cryptographic attestation

Algorithm: `HMAC-SHA256`
Key ID: `mcps-hkdf-e8990a4e1526`
Signer: `mcp-sentinel/v1`
Signed at: `2026-05-24T07:36:16.554Z`
Canonicalization: `RFC8785`

HMAC-SHA256 signature (base64, wrapped at 64 chars):


```
4EhmXHlTzx/0d2pxlDpEuanewRFWc4BqKgqCwDgsRf4=
```


Verification instructions:

```
To verify this report:
  1. Extract the report body (everything except the .attestation field).
  2. Canonicalize the body via RFC 8785 (JCS).
  3. Compute HMAC-SHA256 with the signing key for key_id "mcps-hkdf-e8990a4e1526".
  4. Base64-encode the result and compare with the signature above.
```