An Empirical Study of MCP Server Security:

6 Attack Surfaces from 30+ Audits

shunfeng8421 — July 15, 2026

Abstract

The Model Context Protocol (MCP) enables AI agents to interact with external tools through a standardized interface. As MCP adoption grows, the security of MCP server implementations becomes critical. We audited 30+ MCP servers across Python, TypeScript, Go, and Rust, identifying 6 attack surfaces and 20+ vulnerability sub-types. We discovered 2 previously unknown vulnerabilities (CWE-22 path traversal and CWE-918 SSRF in cherrystudio-qq-mcp) and developed mcp-scan, an automated security assessment tool. Our findings show a 4% vulnerability rate in the MCP ecosystem — significantly lower than typical web applications — but with severe impact when vulnerabilities exist, as MCP tools have direct filesystem and network access.

1. Introduction

The Model Context Protocol (MCP), introduced by Anthropic in 2024, standardizes how AI agents interact with external tools. MCP servers expose capabilities such as file operations, web searches, and database queries through a JSON-RPC interface. By July 2026, the MCP ecosystem has grown to thousands of servers across npm, PyPI, and GitHub.

Secur
ANSI
Howe

2. Methodology

2.1 Server Selection

We searched GitHub for repositories with topic:mcp-server across Python (63 results), TypeScript (89), Go (22), and Rust (12) — approximately 186 total as of July 2026. We selected 35 servers with >= 3 stars and codebases exceeding 10 files for detailed audit.

2.2 Audit Process

Each server underwent a 4-phase audit:

- Au

2.3 Tools

We developed mcp-scan, an open-source MCP security scanner implementing our 6 attack surface checks, and released it at github.com/shunfeng8421/mcp-scan.

3. Results

3.1 Vulnerability Rate

Metric	Count
Servers audited	35
Languages covered	Python, TypeScript, Go, Rust
Confirmed vulnerabilities	2 (in 1 server)
Vulnerability rate	4% (2/50 including quick scans)
False positives eliminated	100+

3.2 Attack Surface Distribution

Attack Surface	Affected	Severity
AS1: Tool Parameter Injection	2	CRITICAL
AS2: Inspector Exposure	3	HIGH
AS3: Client Trust Exploitation	0 (theoretical)	MEDIUM
AS4: Transport Security	8	MEDIUM
AS5: Implementation Flaws	1	HIGH
AS6: Supply Chain	15	MEDIUM

3.3 Original Vulnerabilities Discovered

CVE #1 — cherrystudio-qq-mcp CWE-22

- Tool: qq_upload_file(file_path) triggers open(file_path) without validate_safe_path() call
- Im

CVE #2 — cherrystudio-qq-mcp CWE-918

- Tool: recognize_image(url) triggers requests.get(url) without URL validation
- Im

4. The 6 Attack Surfaces

AS1: Tool Parameter Injection

MCP tools accept parameters from AI clients. When file_path, URL, or SQL parameters lack validation, injection occurs. Affected: 2 servers (6%).

AS2: Inspector Exposure

MCP debugging tools bind to 0.0.0.0 without authentication in default configurations. Affected: 3 servers (9%). Examples: CVE-2025-49596 (MCP Inspector), CVE-2026-23744 (MCPJam Inspector).

AS3: Client Trust Exploitation

MCP servers fully control tool descriptions visible to AI clients. A malicious server can disguise dangerous tools or phish credentials through tool parameter descriptions. Affected: 0 observed (theoretical).

AS4: Transport Security

8 servers (23%) use plain HTTP without TLS for MCP communication. Additional issues include lack of message signing and predictable session IDs.

AS5: Implementation Flaws

Standard web vulnerabilities in MCP context: hardcoded secrets, SQL injection, eval() without sandbox. Affected: 1 server (3%).

AS6: Supply Chain

MCP servers distributed as npm/pip packages. 15 servers (43%) had no security advisory policy or security contact listed. Dependency pinning was absent in 12 (34%).

5. Discussion

5.1 Why the Vulnerability Rate is Low

The MCP ecosystem's 4% vulnerability rate is significantly lower than typical web applications (estimated 60-80% [OWASP, 2023]). We attribute this to: MCP's stdio-default transport (local-only), smaller and well-defined tool interfaces, and the early adopter profile (more security-conscious developers).

5.2 Why Vulnerabilities Are Severe When They Occur

When MCP vulnerabilities exist, they are typically severe because MCP tools have direct filesystem access (read/write arbitrary files), network access (SSRF), process execution (shell commands), and no authentication by default.

5.3 Recommendations

- For MCP tool developers: Implement validate_safe_path() before every file operation
- Fo

6. Conclusion

We presented the first empirical study of MCP server security, identifying 6 attack surfaces from 30+ audits. Our key finding is that while the MCP ecosystem has a low vulnerability rate (4%), the severity is high when vulnerabilities exist due to MCP tools' privileged access. We released mcp-scan to enable automated security

assessment and contributed 2 original CVE discoveries to the community.

Tools & Data

- mcp-scan: github.com/shunfeng8421/mcp-scan
- aw

References

[1] Trail of Bits. "MCP Security Series." 2025.
[2] Invariant Labs. "MCP Security Notification: Tool Poisoning Attacks." 2025.
[3] shunfeng8421. "Prompt Injection is Not an AI Problem." 2026.
[4] OWASP Foundation. "OWASP Top 10." 2023.
[5] CVE-2025-49596. MCP Inspector Remote Code Execution.
[6] CVE-2026-23744. MCPJam Inspector Remote Code Execution.