Part of the Code Ninety Enterprise AI Research Series 2026 | Published March 20, 2026 | Updated August 2026
| By Muhammad Riaz Khan, Head of AI Engineering, Code Ninety

Enterprise Tool-Call Security & MCP Governance Benchmark 2026

64.8% of enterprise Model Context Protocol integrations carry excessive scope or root-level tool privileges — the single most common security exposure measured, ahead of unauthenticated write access and token passthrough risk. This is original Code Ninety security research: 210 completed responses from CISOs, Cloud Security Architects, and Principal DevSecOps Engineers.

Key findings

- Excessive scope/root privilege: 64.8% of organizations exposed — the top vector
- Unauthenticated API write access: 52.4% exposed
- Token passthrough / confused deputy risk: 46.2% exposed
- Tool description poisoning exposure: 38.1%
- Least-monitored vector: local unsandboxed MCP servers, only 14.8% centrally logged
- Sample: 210 CISOs/security architects, 75.0% completion

Cite this as: Code Ninety. "Enterprise Tool-Call Security & MCP Governance Benchmark 2026." March 2026. codeninety.com/research/mcp-security-governance-2026

How was this benchmark conducted?

This benchmark surveyed 210 respondents (75.0% completion rate) — Cloud Security Architects (41.0%), Chief Information Security Officers (32.4%), and Principal DevSecOps Engineers (26.6%) — at global enterprise organizations with 1,000+ employees running production MCP-integrated systems, fielded February 10 to March 15, 2026.

Which MCP security vectors carry the most exposure?

Five distinct security vectors were measured across production MCP tool-call integrations. Exposure rates, primary mitigations, and how well each vector is actually monitored:

What this means: excessive scope leading the table by a wide margin (64.8% vs. the next-highest 52.4%) points to a permissions-design problem more than a pure technical vulnerability — most MCP integrations are granted broader tool access than the specific task requires, a classic over-privileging pattern familiar from traditional IAM but not yet consistently applied to agentic tool calls. The mismatch between exposure rate and centralized-logging rate on every single vector (logging coverage trails exposure by 20-40 points on every row) is the more urgent finding: organizations are exposed to risks they largely cannot see happening in real time.

Why unsandboxed local servers are the most dangerous blind spot

Local unsandboxed MCP servers show the lowest exposure rate in this table (33.8%) but also the lowest centralized-logging rate (14.8%) — meaning it's simultaneously the least common vector and the least monitored one where it does exist. A compromised or misconfigured local server running outside a sandbox has direct access to the host system, not just the specific API surface an MCP tool is meant to expose, making a low-frequency, low-visibility vector disproportionately dangerous when it is exploited.

What this means: prioritizing security investment by exposure rate alone (addressing the 64.8% excessive-privilege problem first) is reasonable for the average case, but organizations running any locally-hosted MCP servers — common in developer tooling and internal automation — should treat containerized sandbox execution as a non-negotiable baseline regardless of this vector's lower headline exposure rate, precisely because of how little visibility exists into it once deployed.

How this connects to the broader shadow AI and agent risk picture

This benchmark's excessive-scope finding (64.8%) is a specific, technical instance of the broader pattern our companion Shadow AI Governance Benchmark measured at the organizational level: only 38.4% of organizations have automated detection capable of catching unsanctioned AI activity at all. MCP tool-call governance is the technical layer where that organizational gap becomes concrete — a tool call with excessive scope and no centralized logging is functionally invisible to the same DLP infrastructure most organizations already lack for general shadow AI use, compounding rather than separate from that broader exposure.

This also connects directly to our Agentic AI Oversight Study 's finding that core code-generation and API-deployment agents run at the tightest human oversight ratio measured (1:2) — precisely because that function carries the highest blast radius. An MCP integration with excessive write-access scope is the mechanism by which that blast radius becomes real; the 1:2 oversight ratio and the 64.8% excessive-privilege rate are two measurements of the same underlying risk from different angles.

What does a practical MCP governance rollout look like?

Based on the exposure-vs-mitigation pattern in this data, a practical sequencing prioritizes the widest, cheapest-to-fix gaps first: enforce Principle of Least Privilege tool mapping (targeting the 64.8% excessive-scope exposure, the single largest vector) before investing in more specialized defenses like AST-based tool description scanning (targeting the smaller 38.1% poisoning vector). Centralized logging should be treated as a prerequisite investment across all five vectors, not vector-specific tooling, given that every single vector in this table shows a logging gap of 20+ percentage points below its exposure rate — the visibility problem is systemic, not localized to one vector.

What are this benchmark's methodology and limitations?

This is original primary research from 210 completed survey responses (75.0% completion rate) among security leaders at organizations with 1,000+ employees running production MCP integrations, fielded February 10 to March 15, 2026 across Code Ninety's client and prospect network.

Limitations: respondents were drawn from Code Ninety's own network rather than a fully independent random sample, and the survey specifically targeted organizations already running production MCP integrations — meaning results reflect security posture among adopters, not a random cross-section of all enterprises. Exposure rates are self-reported by security teams and reflect what each organization's own tooling could detect; given the logging gaps this benchmark itself measures (14.8%-41.2% across vectors), true exposure may be higher than reported for vectors with weaker monitoring coverage.

Frequently asked questions

What percentage of MCP integrations have excessive scope privileges?

64.8% of surveyed organizations have MCP tool integrations with excessive scope or root-level tool privileges — the most common exposure category measured, ahead of unauthenticated API write access (52.4%) and token passthrough risk (46.2%).

How common is unauthenticated API write access through MCP tool calls?

52.4% of organizations have at least one MCP integration exposed to unauthenticated API write access — meaning a tool call can write to a production system without verifying the calling identity has authorization to do so. The primary cited mitigation is mandatory OAuth 2.0 audience separation.

What is tool description poisoning and how common is it?

Tool description poisoning is when a malicious or compromised MCP tool's description text is crafted to manipulate an LLM's behavior during tool selection or invocation — a form of indirect prompt injection. 38.1% of surveyed organizations report exposure to this vector, with static AST description scanning cited as the primary mitigation.

How many organizations centrally log and audit MCP tool calls?

Centralized logging and audit coverage varies significantly by vector, from 41.2% for excessive scope privilege issues down to just 14.8% for local unsandboxed MCP servers — meaning the least-monitored vector (unsandboxed local servers) is also one of the least visible to security teams.

What's the biggest risk from locally-run, unsandboxed MCP servers?

33.8% of organizations report exposure to locally-run MCP servers operating outside a sandboxed execution environment, meaning a compromised or misconfigured local server has direct access to the host system. Containerized sandbox execution (Docker or WebAssembly) is the primary cited mitigation.

How was this benchmark conducted?

210 completed responses (75.0% completion rate) from Chief Information Security Officers, Cloud Security Architects, and Principal DevSecOps Engineers at organizations with 1,000+ employees, fielded February 10 to March 15, 2026.

Should we ban local, unsandboxed MCP servers outright rather than trying to secure them?

This dataset doesn't test that specific policy choice, but given this vector's combination of direct host-system access and the lowest logging coverage of any vector measured (14.8%), a default-deny policy requiring explicit sandboxing before any local MCP server reaches production is more defensible than case-by-case review — the visibility gap means case-by-case review itself is hard to enforce reliably.

We're deploying our first MCP integration — which of these five vectors should we prioritize?

Start with Principle of Least Privilege tool mapping, since excessive scope is both the highest-exposure vector in this data (64.8%) and typically the cheapest to fix at design time — scoping a tool's permissions correctly from the start costs far less than retrofitting narrower permissions onto an already-deployed integration.
