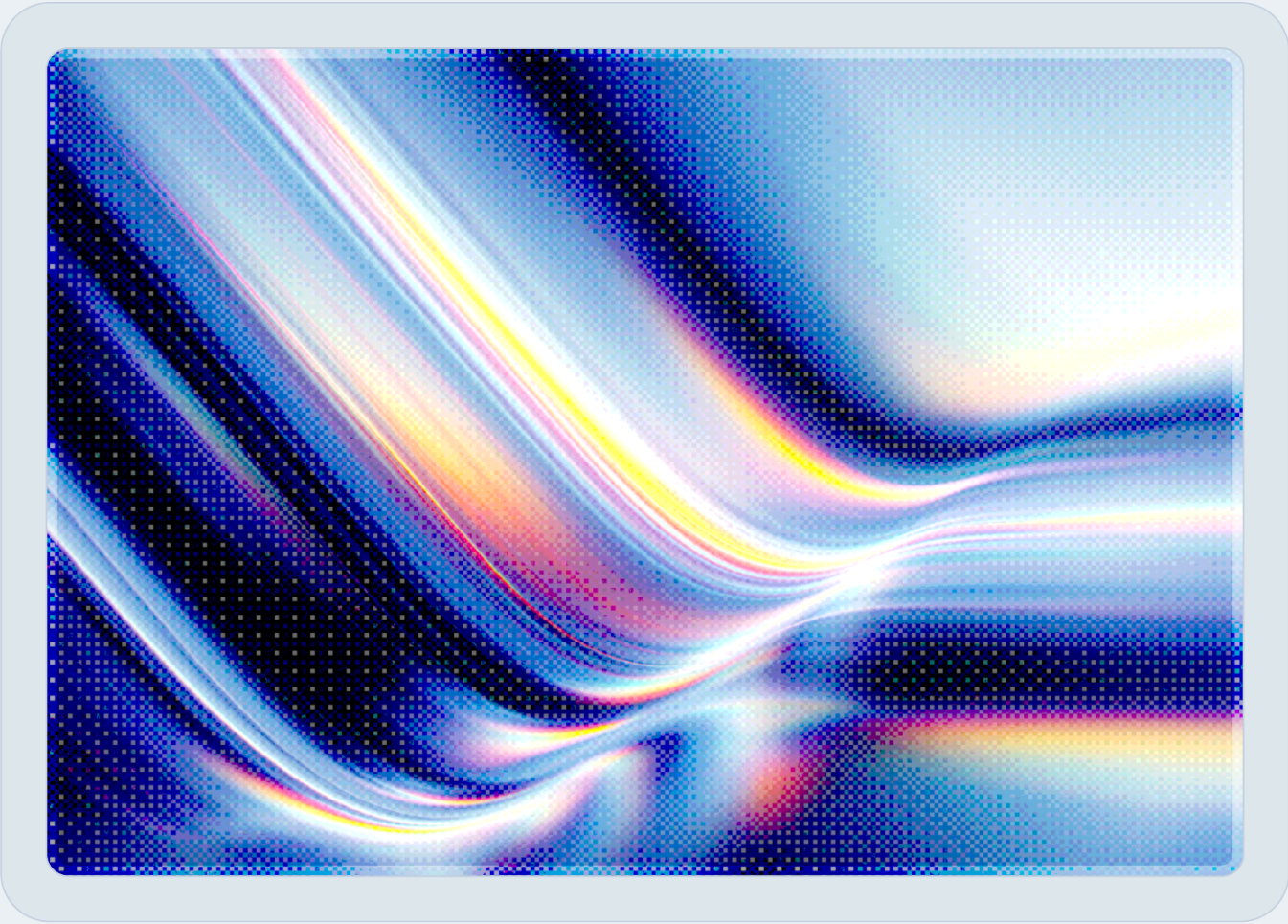
CSAI Foundation | Cloud Security Alliance

MCP's Architectural Flaw: The AI Tool Permission Crisis

Structural Security Deficiencies in a Rapidly Scaling AI Integration Standard

2026-05-03

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Table of Contents

Executive Summary

The Model Context Protocol has achieved what few technical standards accomplish in less than two years: it has become the infrastructure layer for an entire industry. Since Anthropic released the open standard in November 2024 [1], it has been adopted by Microsoft, Google, Cloudflare, Atlassian, Block, and hundreds of independent software vendors. SDK downloads exceeded 150 million by early 2026 [2]. Registries now list more than 5,000 public servers [24], and security researchers using scanning methodology developed by OX Security have estimated that more than 200,000 instances are deployed across public and private infrastructure when accounting for servers running STDIO transport on accessible endpoints [2]. The protocol has, in the span of roughly eighteen months, become the connective tissue of agentic AI deployment.

It has done so with a security model that its own specification acknowledges as incomplete. The official MCP specification states that the protocol "explicitly does not enforce security at the protocol level," placing implementation responsibility entirely on individual developers and organizations [3]. In practice, this design choice has produced a large, rapidly growing ecosystem in which authentication is inconsistently applied, tool definitions can be manipulated after approval, authorization grants lack the granularity enterprise environments require, and registries have proven susceptible to poisoning with malicious submissions that pass through without review. Security researchers from OX Security, Trail of Bits, Palo Alto Networks Unit 42, Wiz [21], CyberArk [20], Elastic Security Labs [19], and Invariant Labs have each documented attack classes that exploit these conditions, with working demonstrations including cross-server behavioral hijacking [9], line-jumping prompt injection [12], and sampling-mechanism manipulation [11].

Among the most consequential disclosed vulnerabilities is a critical remote code execution vulnerability in MCP's STDIO transport mechanism, disclosed by OX Security in April 2026. Researchers identified that default configuration values in MCP's official SDKs across Python, TypeScript, Java, and Rust create a path from tool invocation to operating system command execution [2]. Anthropic declined to modify the protocol architecture in response, characterizing the behavior as expected and necessary to maintain MCP's position as "an unopinionated, universal standard" [4]. That decision left unpatched a vulnerability affecting implementations totaling more than 150 million downloads, with a secondary consequence that has received less attention: it suggests, though does not conclusively demonstrate, that architectural changes in MCP may not be achievable through security-only arguments alone. Enterprises should factor this pattern into their governance assumptions while recognizing that a single disclosure response is limited evidence for a general principle.

For enterprise security teams, MCP presents a governance challenge that is partly technical and partly organizational. The technical challenges — validating tool definitions, enforcing least-privilege, auditing AI agent actions — are significant but tractable given adequate controls. The organizational challenge is more

difficult: survey data showing that 78 percent of enterprise AI teams report at least one MCP-backed agent in production [26], combined with the absence of MCP-specific governance controls in most established security frameworks, indicates that adoption has in many environments outpaced security review, driven by individual developer decisions that preceded or bypassed standard procurement governance and created a shadow AI infrastructure populated by tools whose security posture ranges from carefully constructed to deployed without documented security review. CSA's AI Controls Matrix, MAESTRO agentic threat model, and Zero Trust guidance provide a framework for addressing both dimensions. Enterprises that do not apply this framework are not merely accepting risk — they are accepting risk they cannot currently measure.

Introduction: The Integration Standard Without Standardized Security

What MCP Does and Why It Spread

Before the Model Context Protocol existed, connecting an AI assistant to an external tool required custom integration work for each combination of model and service. A development team that wanted Claude to query their database, check their issue tracker, and send Slack messages had to build three separate integrations, each with its own authentication approach, data serialization format, and error handling logic. The same was true for OpenAI, Google's Gemini, and every other model. Integration code was model-specific, service-specific, and non-reusable.

MCP solved this problem by defining a universal communication layer between AI models and external tools. Under the protocol, tool providers implement MCP servers that expose capabilities through standardized interfaces, and AI clients implement MCP hosts that can discover and invoke any compliant server. A developer who builds an MCP server for their database service automatically makes that service available to every AI platform that supports MCP — without any additional integration work. From an engineering productivity perspective, MCP aims to do for AI integration what REST APIs did for web services — providing a common interface convention that reduces bespoke integration work — with similar architectural tradeoffs between flexibility and standardization.

The protocol defines three core primitives. Tools are executable functions the AI can invoke — querying a database, sending a message, executing a command. Resources are data sources the AI can read — documents, code repositories, calendar entries. Prompts are templated interaction patterns that servers can offer to standardize common workflows. Communication flows through transport mechanisms: Stdio (standard input/output) for local processes, and HTTP with Server-Sent Events for remote servers. The simplicity of this design made adoption easy, and the results are reflected in the growth numbers. Within

three months of launch, the MCP ecosystem had expanded from Anthropic's own reference servers to hundreds of community-built integrations; within a year, major enterprise software vendors had shipped MCP support for production use [5].

That speed of adoption, however, bypassed the security review cycles that would normally accompany the introduction of a new integration standard into enterprise environments. Individual developers adopted MCP because it made their AI workflows more capable, not because their organization had assessed and approved it. The result is an integration standard that has become enterprise infrastructure through a path that preceded or bypassed enterprise governance review cycles.

The Security Paradox at the Core of MCP

The MCP specification's security philosophy reflects a deliberate tradeoff. The protocol's designers prioritized broad interoperability and ease of implementation, recognizing that restrictive security requirements built into the standard would raise the cost of adoption and might favor well-resourced implementors over independent developers. They documented this choice explicitly: the protocol does not enforce security, and implementors bear responsibility for applying security controls appropriate to their deployment context [3].

This is not an unusual design philosophy in open standards. HTTP does not enforce encryption; that is HTTPS's role. SMTP does not authenticate senders; SPF, DKIM, and DMARC address that. The assumption underlying such designs is that the ecosystem will develop appropriate security layers as adoption matures, and that mandating security in the base protocol would create barriers that slow adoption without guaranteeing adequate outcomes.

The problem with this assumption in MCP's context is one of timing. HTTP and SMTP each spent years or decades in widespread deployment before security layers became standard practice — SSL/TLS did not achieve broad deployment until the 2000s, and email authentication frameworks took years to gain traction after spam and spoofing became endemic. That maturation was accompanied by significant exploitation throughout the interim. The question for MCP is whether the same slow maturation arc is acceptable given that enterprise AI agents operate with substantially higher access privileges than early web browsers or email clients. MCP reached enterprise production deployment at scale before the security ecosystem around it had formed. Researchers are documenting novel attack classes. Enterprises are deploying MCP servers in production. Security tooling for MCP discovery, monitoring, and control is nascent. The standard deferred security to implementors; the implementors assumed the standard provided a secure foundation; and security research is discovering that neither assumption held.

Section 1: An Architecture That Deferred Security

Transport-Level Vulnerabilities in STDIO

MCP supports two transport mechanisms with meaningfully different security profiles. The HTTP/SSE transport is designed for remote servers and inherits the relatively mature security model of web APIs — authentication headers, TLS encryption, firewall rules, and standard network controls all apply in familiar ways. The STDIO transport is designed for local processes and operates through a fundamentally different model: the host process spawns an MCP server as a child process and communicates through the spawned process's standard input and output streams.

OX Security's April 2026 research examined the security implications of STDIO transport in depth and identified a critical vulnerability in how MCP's official SDKs across Python, TypeScript, Java, and Rust implement this mechanism [2]. The vulnerability arises from default configuration values that allow tool invocations to propagate to operating system command execution. When a malicious actor can influence the tool description or parameters that an AI agent processes — through tool poisoning, described below, or through a compromised server — they can construct inputs that traverse the path from tool invocation to arbitrary command execution on the host system. Researchers identified more than 7,000 servers actively listening on public IP addresses with STDIO transport and, extrapolating from this data, estimated that roughly 200,000 vulnerable instances are deployed across the broader ecosystem [2]. That extrapolation carries uncertainty, and the figure should be understood as a research estimate rather than a census count; the number of publicly catalogued MCP servers in registries is substantially lower. What the research establishes clearly is that the vulnerability class is real, exploitable, and present across all four major official SDKs.

Anthropic's response to OX Security's disclosure acknowledged the finding but declined to modify the protocol architecture. The company characterized the STDIO behavior as intended, explaining that restricting it would compromise MCP's design philosophy as an unopinionated standard [4]. A subsequent update to Anthropic's security guidance recommended using STDIO adapters with caution — a documentation change that does not remediate the vulnerability in the roughly 150 million SDK downloads already in deployment. Researchers noted that a root-level patch to the SDKs would have reduced risk at scale without requiring changes to the protocol specification itself; the decision not to issue such a patch drew criticism from the security community [4][22].

OAuth Implementation Gaps

For remote MCP servers using HTTP transport, authentication is intended to be handled through OAuth 2.0 flows. The specification defines how clients should authorize with servers and how servers should manage access tokens. The implementation guidance, however, contains gaps that create exploitable conditions in practice.

The most significant gap involves what security researchers have termed the "confused deputy" problem, in which an MCP server holding legitimate authorization for one purpose applies that authorization incorrectly, enabling an attacker to exploit the server's credentials without directly stealing them. In the MCP context, this attack class is well-documented through two disclosed vulnerabilities in FastMCP's OAuthProxy component, assigned CVE-2024-53983 [6] and CVE-2026-27124 [27]. The attack proceeds through a sequence that exploits the absence of per-client consent validation. An attacker initiates a legitimate OAuth flow with a malicious client to a benign MCP server. The attacker intercepts the GitHub authorization URL and tricks a victim who has previously authorized the benign server — and thus bypasses GitHub's consent prompt due to prior authorization — into visiting that URL. The victim's browser is redirected to the OAuthProxy callback, which, lacking browser-bound state validation, accepts the authorization exchange and issues a token to the attacker's client. The result is that the attacker obtains an access token tied to the victim's GitHub account without any credential theft in the conventional sense.

Descope's analysis of MCP vulnerability classes identified excessive permissions, insufficient permission granularity, and the absence of permission revocation mechanisms as structural problems across the ecosystem [7]. These are not implementation bugs in specific servers — they reflect the absence of authorization primitives in the MCP specification itself. The specification does not define scoping mechanisms for tools, does not require servers to enforce least-privilege by default, and does not provide a standard mechanism for revoking granular permissions when they are no longer needed. Individual implementations must build these controls independently, with predictably inconsistent results.

The Structural Absence of Audit Infrastructure

Enterprise security programs depend on audit trails to detect anomalous behavior, investigate incidents, and demonstrate compliance with regulatory requirements. MCP's specification does not define logging requirements, event formats, or audit interfaces. Individual server implementations may log their activity in idiosyncratic ways, or may not log it at all. AI agents invoking tools across multiple MCP servers generate distributed activity that has no standard mechanism for correlation.

Workato's analysis of security gaps in the MCP ecosystem found that missing audit trails are among the most commonly cited concerns from enterprise security and compliance teams evaluating MCP adoption [8]. The implication is not merely that investigation after an incident is harder — it is that detection during an incident is harder, and that demonstrating compliance to auditors in regulated industries is harder. Financial services

firms and healthcare organizations operating under regulatory frameworks that require evidence of AI system oversight face particular challenges: MCP's activity happens in a governance gap between the AI model's inputs and outputs, where neither the model's built-in logging nor the tool's own application logs necessarily capture the full interaction.

Section 2: Tool Poisoning and the Invisible Attack Surface

How Agents Trust Tool Definitions

MCP's tool invocation model creates a distinctive attack surface that has no precise analogue in traditional software security. When an AI agent encounters an MCP server, it reads that server's tool definitions — names, descriptions, and parameter schemas — to understand what the server offers and how to invoke its capabilities. The agent then uses these definitions to reason about which tools to call in response to user requests. This is, by design, a trust relationship: the agent trusts that tool descriptions accurately represent tool behavior, and the user trusts that the agent's tool invocations reflect the user's intent.

Tool poisoning attacks exploit this trust relationship by embedding malicious instructions in tool descriptions or parameter metadata [9]. Because the AI agent reads tool metadata to guide its reasoning, and because users typically see only a high-level summary of what the agent is doing rather than the raw tool definitions, there is a gap between what the user expects and what the agent is instructed to do. A tool description might appear benign in its name and summary while containing hidden instructions — rendered invisible through Unicode manipulation, ANSI escape sequences, or simply by embedding instructions in fields the user interface does not display — that direct the agent to exfiltrate data, invoke additional tools, or take actions the user has not requested.

Invariant Labs' disclosure of this attack class in early 2025 included a demonstration in which a malicious MCP server hijacked the behavior of a co-connected trusted email server by embedding instructions in tool metadata that directed the AI agent to redirect outgoing messages to an attacker-controlled destination [9]. The attack succeeded because the agent operated across multiple MCP servers in a single session, meaning a malicious server in the session could leverage the agent's access to trusted servers — including the data they held — without directly compromising those servers.

Empirical Attack Success Rates

The MCPTox benchmark, an academic evaluation of tool poisoning attacks against production MCP deployments, assessed 45 live MCP servers and 353 authentic tools across 20 major AI agent platforms [10]. Attack success rates exceeded 60 percent across tested agents, with the highest single figure reaching 72 percent. A counterintuitive finding emerged from the analysis: more capable models were, in some configurations, more susceptible to tool poisoning than less capable ones — a pattern the researchers attributed to those models' higher instruction-following compliance making them more responsive to embedded directives [10]. If this interpretation holds, the implication is that security does not improve automatically as models become more powerful; in this attack class, it may degrade.

Palo Alto Networks' Unit 42 researchers documented additional attack variants through MCP's sampling mechanism, in which servers can request completions from the client's language model [11]. By controlling both the prompt content and how the model's response is processed, a malicious server can inject instructions into the agent's reasoning context, manipulate output to influence subsequent tool calls, and potentially cause the agent to take actions that appear user-directed but originate from the server. This attack class is distinct from tool poisoning in that it does not require access to the tool definition — it operates through the AI's own inference process.

Trail of Bits and the Line-Jumping Attack

Trail of Bits' security research produced a specific attack variant they termed "line jumping," in which malicious servers inject prompts through tool descriptions to manipulate AI behavior before any tool invocation occurs [12]. The attack is notable because it bypasses human-in-the-loop controls without requiring the agent to call a malicious tool at all. Instead, it operates directly on the agent's context window during the tool-discovery phase, when the agent reads server metadata to understand available capabilities. By that point, the malicious instructions are already in the model's context, and the human approval step — which occurs before tool calls, not before metadata reading — does not provide protection.

Trail of Bits documented additional vectors including ANSI escape sequence injection to conceal instructions from log display systems and embedding directives in metadata fields that standard user interfaces do not render [12]. These techniques share a common property: they exploit the gap between what the user sees and what the AI model processes. As long as there is no cryptographic or structural guarantee that tool metadata presented to the model is identical to what was reviewed and approved by a human, this gap remains exploitable.

Section 3: Rug-Pull Attacks and Dynamic Trust Degradation

The Approval That No Longer Applies

MCP's authorization model is transactional: a user approves a server's tools at a particular moment, and the agent proceeds to use them. What the current specification does not address is what happens when the tools change after approval. An MCP server can modify tool descriptions, parameters, and behaviors between the user's approval event and the agent's tool invocations, and under normal protocol operation, nothing alerts the user or the agent that the approved definition is no longer the active one.

This vulnerability class, termed a "rug-pull attack," was analyzed by ReversingLabs and Akto, among others [13][14]. The attack scenario proceeds as follows: a server operator deploys a legitimate, useful MCP server that earns user trust and widespread adoption. After achieving deployment across a significant number of AI agent environments, the operator modifies the server's tool definitions — changing descriptions to include malicious instructions, altering parameter handling to redirect sensitive data, or redefining behavior entirely. The agents connected to the server continue to operate on the assumption that the definitions they approved are still accurate. There is no built-in mechanism in the MCP specification for detecting, alerting on, or preventing this class of change.

The absence of this mechanism is a structural property of MCP's design, not an oversight in a particular implementation. The specification does not define version identifiers for tool definitions, does not require servers to emit change notifications, and does not provide clients with a standard way to pin a particular definition and detect deviation from it. Individual security tools — MCP gateways and proxy layers — have begun to offer definition pinning as a feature, but these controls are not present by default in any standard MCP client [13]. Academic analysis has identified rug-pull attacks and context poisoning as among the most structurally problematic attack classes in the MCP ecosystem, arising directly from the protocol's lack of tool definition versioning and change notification mechanisms [18].

Malicious Updates and the Trust Lifecycle

The rug-pull attack class is related to, but distinct from, a broader concern about how MCP server trust evolves over time. A server that is deployed as genuinely useful and benign can be updated to become malicious through several mechanisms: voluntary change by the operator, compromise of the operator's infrastructure, or acquisition of the server by a new operator with different intentions. Because MCP clients

do not have a standard mechanism for associating a server's current behavior with its behavior at the time of initial trust establishment, all of these scenarios produce the same result: an agent operating with an outdated approval decision.

SC Media's analysis of MCP supply chain risks noted that real-world exploit activity against MCP servers accelerated significantly in early 2026, with the first confirmed cases of MCP servers being used as initial access vectors in enterprise intrusions [15]. The supply chain dynamic mirrors patterns seen in npm and PyPI package compromises, where attackers target widely-used packages not for the value of compromising any single dependency but for the leverage that a single compromise provides over the many downstream environments that consume it. The mcp-remote package, a widely-used dependency for connecting to remote MCP servers, had accumulated more than 437,000 downloads as of early 2026 — a scale at which a single supply chain compromise becomes a significant enterprise security event [15].

Section 4: The Enterprise Risk Landscape

What Enterprises Are Actually Deploying

Understanding MCP's risk profile in enterprise environments requires understanding how MCP adoption is actually proceeding. Survey data on MCP adoption found that 78 percent of enterprise AI teams report at least one MCP-backed agent in production as of April 2026 [26]. An analysis of 1,400 MCP server implementations by Bloomberry found that 38.7 percent of servers require no authentication, 22.9 percent configure open CORS settings, and only 2.4 percent implement rate limiting — and that 81 percent of organizations running MCP servers have fewer than 200 employees, suggesting that the majority of MCP server operators are not large enterprises with mature security programs [16].

The access that MCP servers hold in practice is significant. An AI agent connected to a database MCP server, a filesystem server, an email server, and a code repository server — a configuration representative of productivity-focused AI deployments — has read and potentially write access to an organization's most sensitive data categories simultaneously. Credential stores, customer records, source code, and internal communications all become available to the agent in a single session. Because MCP is designed for utility, servers are commonly configured to request the broadest access necessary for their anticipated function rather than the narrowest access consistent with current task requirements, a pattern consistent with the authentication and access control gaps identified in security assessments of MCP implementations [16]. The principle of least privilege, foundational to enterprise access control, has no standard enforcement mechanism in the MCP architecture.

Witness AI's analysis identified seven risk categories that enterprise security teams should assess in MCP deployments: sensitive data exfiltration through compromised servers, unauthorized agent actions beyond user intent, overprivileged access granted by design, supply chain exposure through third-party servers, missing audit trails preventing detection and investigation, privilege escalation through lateral movement across connected tools, and shadow AI sprawl as MCP deployments accumulate outside IT governance [17]. Each of these categories represents a known risk class from traditional security contexts; MCP concentrates them in a single integration layer that is currently deployed without systematic enterprise oversight.

Financial Sector Signaling

Regulated industries face particular exposure from MCP's governance gaps. Financial institutions operating enterprise AI deployments face the MCP governance gap compounded by specific regulatory obligations that require demonstrable oversight of material technology dependencies. The EU's DORA regulation, which entered full applicability in January 2025, and the U.S. SEC's cybersecurity disclosure rules require evidence

that organizations have assessed and manage the risks of their technology infrastructure. An enterprise AI deployment that relies on MCP servers without documented security assessment, access logging, or change management creates compliance exposure alongside security risk — and MCP's native architecture does not produce the documentation these frameworks require.

The structural challenge for financial institutions is that compensating controls can address individual MCP implementations but cannot retroactively enforce security properties that are absent from the integration standard itself. A financial organization can log at the application layer; it cannot obtain logs that the protocol does not produce. It can validate inputs to servers it controls; it cannot prevent tool poisoning attacks on servers it does not control. Industry analysis has characterized this gap between what enterprise security programs require and what MCP provides natively as one of the sector's most significant unaddressed technology risks [4]. The concern is structural: the gap between what enterprise security requires and what MCP provides natively is not closeable through internal controls alone.

Registry Poisoning and Ecosystem Trust

MCP servers are distributed through registries — public catalogs where developers list their servers and users discover them. The security of these registries is a prerequisite for the trustworthiness of the broader ecosystem, and the research findings on registry security are concerning. A security analysis testing eleven major MCP registries found that nine of them could be successfully poisoned with malicious submissions that passed through without automated or human review [15][23]. The implication is that a user discovering an MCP server through a registry has limited assurance that the server has been reviewed for malicious behavior, and that attackers can establish a presence in discovery channels that enterprise users rely on.

The supply chain parallel to npm and PyPI is structural, not incidental. Both of those ecosystems have experienced sustained compromise campaigns because they combine high value to attackers — access to many downstream installations — with low barriers to establishing a presence. MCP registries have not yet invested in the security review infrastructure that npm and PyPI have developed, in some cases imperfectly, over years. They are at an early and vulnerable stage in the ecosystem maturity cycle, and they are operating at enterprise scale.

Section 5: Recommendations

Immediate Actions

Enterprises that have deployed MCP agents in production environments should treat the current period as a security remediation window. The highest-priority action is an inventory of deployed MCP servers: what servers are present in agent configurations, what access those servers hold, and whether the servers originate from the organization's own development, from trusted vendors, or from public registries. Without this inventory, the risk surface cannot be assessed or managed.

Alongside inventory, enterprises should evaluate whether tool definitions for deployed servers have changed since initial deployment. Definition pinning — recording approved tool schemas cryptographically and alerting on deviation — is available through several emerging MCP gateway products, and organizations should implement it for any servers holding access to sensitive systems. The absence of a built-in MCP mechanism for this control does not eliminate the requirement; it means the control must be applied at the gateway or proxy layer.

For any MCP server that holds credentials to production systems — database access, API keys, authentication tokens — organizations should review whether those credentials are stored in plaintext on local filesystems, a vulnerability class identified in security assessments of MCP implementations [12]. The MCP architecture does not define secure credential storage requirements; individual implementations vary widely, and ad hoc credential storage in developer environments is a common path for credential exposure.

Short-Term Mitigations

Over the next quarter, enterprise security programs should establish MCP-specific governance controls as a recognized component of their AI security practice. This includes incorporating MCP server security into vendor assessments for AI tools that use MCP integrations, applying the same third-party risk management scrutiny to MCP server operators as to other software vendors with access to enterprise data. Vendors who cannot provide audit logging, change management documentation, or security assessment evidence for their MCP servers should be treated as high-risk integrations.

Network segmentation for MCP traffic can limit the blast radius of a compromised server. AI agents should operate in environments where their network access is constrained to the specific systems they need to reach, so that a server that achieves tool-poisoning-based code execution does not have unrestricted access to the internal network. Where STDIO transport is in use, the processes involved should run in

restricted execution environments with minimal filesystem and network permissions. Configuration guidance from enterprise security vendors including Red Hat [25] addresses specific hardening patterns for MCP deployments within containerized and otherwise restricted execution environments.

Organizations should begin building towards MCP-specific monitoring capabilities. Even where the protocol does not produce audit-ready logs natively, it is possible to instrument at the transport layer — logging the content of tool invocations and responses — to create a record suitable for anomaly detection and incident investigation. This is particularly important for agents with access to sensitive data categories, where the absence of audit infrastructure creates both security and compliance risk.

Strategic Considerations

The longer-term strategic challenge is that MCP is becoming infrastructure — embedded in major AI platforms and enterprise software products — faster than the security ecosystem around it is maturing. Organizations that wait for the ecosystem to produce mature, standardized security tooling before deploying MCP risk being displaced by competitors who accept higher risk for faster deployment. Organizations that deploy without governance risk accumulating significant security debt in their AI infrastructure.

The appropriate strategic response is structured adoption: deploying MCP capabilities under governance controls that are less mature than ideal but better than none, while tracking the evolution of the ecosystem's security tooling and upgrading controls as better options become available. This requires establishing an MCP security roadmap as a recognized part of the organization's AI governance program, rather than treating MCP security as a one-time implementation decision.

Industry-level responses are also warranted. Enterprises that are significant customers of Anthropic and other AI vendors that depend on MCP should use their commercial relationships to advocate for protocol-level security improvements. Anthropic's decision to decline a root-level patch for the STDIO RCE vulnerability reflects a particular reading of the tradeoff between security and interoperability; sustained pressure from enterprise customers can shift that calculus. Industry groups, including CSA, can provide forums for coordinating that pressure and for developing consensus on minimum security requirements for MCP implementations.

Section 6: CSA Resource Alignment

MAESTRO Threat Model

CSA's MAESTRO framework for agentic AI threat modeling provides the most directly applicable analytical structure for MCP security risks [28]. MAESTRO's seven-layer model — spanning Foundation Models (Layer 1) through the Agent Ecosystem (Layer 7) — maps directly onto the MCP attack surface. Tool poisoning attacks are introduced through Layer 7 (Agent Ecosystem), where external MCP tool servers are sourced and consumed, and exploit the trust relationships in Layer 3 (Agent Frameworks), where agent reasoning and tool invocation logic operates. Rug-pull attacks represent supply chain risk concentrated in Layer 7 (Agent Ecosystem), arising when trusted server definitions are modified after the initial trust decision. OAuth confused deputy vulnerabilities span Layer 7, where external authorization services are accessed, and Layer 3, where agent authorization handling operates. The MAESTRO framework provides security teams with a consistent vocabulary for describing and categorizing MCP threats, and its threat catalog can be used to structure MCP-specific threat modeling exercises.

AI Controls Matrix

CSA's AI Controls Matrix (AICM) maps security controls to AI system components and provides an enterprise compliance framework for AI deployments. Several AICM control domains are directly relevant to MCP governance. The AI Supply Chain Security domain addresses the third-party risk controls that enterprises need for MCP server vetting. The AI System Access Control domain provides the framework for least-privilege enforcement in tool access. The AI Audit and Accountability domain covers the logging and monitoring requirements that MCP's native architecture does not fulfill. Organizations using AICM for AI governance can map MCP-specific controls to these domains to integrate MCP security into their existing AI compliance programs rather than treating it as a separate initiative.

Cloud Controls Matrix and STAR

CSA's Cloud Controls Matrix (CCM) and the STAR (Security Trust Assurance and Risk) certification program provide mechanisms for enterprise organizations to assess the security posture of third-party cloud services. MCP server operators — particularly those offering hosted, remote MCP services — should be assessed against relevant CCM domains including Application and Interface Security, Identity and Access Management, and Security Incident Management. STAR certification provides a standardized evidence

framework for these assessments. Organizations that require STAR certification or equivalent from major SaaS vendors should extend that requirement to MCP server operators holding access to sensitive enterprise systems.

Zero Trust Application

Zero Trust architecture principles apply directly to MCP deployments and provide a governance philosophy consistent with MCP's threat model. The Zero Trust principle of "never trust, always verify" maps to MCP tool invocation: an agent should not treat tool definitions as inherently trustworthy simply because they arrive from a configured server, and should not assume that a server's current behavior matches its behavior at the time of trust establishment. Zero Trust's emphasis on continuous verification and minimal privilege maps to the controls discussed in the recommendations section: definition pinning, scope-limited credentials, and session-level authorization rather than persistent broad access. CSA's Zero Trust guidance, combined with the MAESTRO and AICM frameworks, provides a comprehensive governance foundation for enterprise MCP deployments.

Conclusion

The Model Context Protocol has achieved rapid, broad adoption by solving a genuine problem — the fragmentation of AI integration work — and by making adoption as frictionless as possible. The same design philosophy that produced those outcomes produced the security gaps this paper has examined. Protocols that defer security to implementors create ecosystems in which security is inconsistent at best; implementors who assume the protocol provides a secure foundation deploy with inadequate controls; and the resulting surface area is large, heterogeneous, and difficult to assess.

The enterprise security community's appropriate response is neither to reject MCP — the interoperability benefits are significant, and MCP is becoming infrastructure in ways that make avoidance increasingly impractical — nor to accept the current risk profile uncritically. It is to govern MCP adoption with the same rigor applied to any infrastructure component: inventorying deployments, assessing third-party server operators, applying compensating controls where protocol-level protections are absent, and advocating through commercial and industry channels for protocol improvements that address the structural gaps.

The period ahead is particularly consequential. MCP is at the stage in its lifecycle where the ecosystem's security norms are still being established. The practices that become standard now — whether organizations demand audit logging, whether registries implement review processes, whether server operators ship with secure defaults — will shape the security profile of AI integration infrastructure for years. CSA's frameworks provide the analytical tools and governance vocabulary to make that case within enterprise security programs. The urgency is not hypothetical: research has demonstrated working exploits, disclosed critical

vulnerabilities affecting hundreds of millions of downloads, and documented real-world supply chain compromises. SC Media's reporting documents accelerating real-world exploit activity against MCP servers in early 2026 [15], and the absence of systematic enterprise governance controls means that incident response, when required, will occur without the audit trails and access inventories that investigation requires.

References

[1] Anthropic. "Introducing the Model Context Protocol." Anthropic, November 2024.

[2] OX Security. "The Mother of All AI Supply Chains: Critical Systemic Vulnerability at the Core of the MCP." OX Security, April 2026.

[3] Model Context Protocol. "Security Best Practices." modelcontextprotocol.io, 2025.

[4] VentureBeat. "200,000 MCP servers expose a command execution flaw that Anthropic calls a feature." VentureBeat, April 2026.

[5] Zuplo. "The State of MCP: Adoption, Security & Production Readiness." Zuplo, 2026.

[6] Daily CVE. "FastMCP OAuthProxy Confused Deputy Vulnerability (CVE-2024-53983)." Daily CVE, 2024.

[7] Descope. "Top 6 MCP Vulnerabilities." Descope, 2025.

[8] Workato. "Top 10 MCP Security Gaps CISOs Need to Know." Workato, 2026.

[9] Invariant Labs. "MCP Security Notification: Tool Poisoning Attacks." Invariant Labs, 2025.

[10] Arxiv. "MCPTox: Benchmarking Tool Poisoning Attacks in the MCP Ecosystem." arxiv.org, 2025.

[11] Palo Alto Networks Unit 42. "New Prompt Injection Attack Vectors Through MCP Sampling." Palo Alto Networks, 2025.

[12] Trail of Bits. "We Built the Security Layer MCP Always Needed." Trail of Bits, July 2025.

[13] ReversingLabs. "MCP Client Rug-Pull Attack Worries." ReversingLabs, 2025.

[14] Akto. "MCP Attack Matrix: Rug Pull Attacks." Akto, 2025.

[15] SC Media. "MCP Servers Emerge as New Supply Chain Risk as Real Attacks Accelerate." SC Media, 2026.

[16] Bloomberry. "We Analyzed 1,400 MCP Servers – Here's What We Learned." Bloomberry, 2025.

[17] Witness AI. "7 MCP Server Security Risks Enterprises Must Address." Witness AI, 2026.

[18] Arxiv. "Securing the Model Context Protocol." arxiv.org, 2025.

[19] Elastic Security Labs. "MCP Tools: Attack and Defense Recommendations." Elastic, 2025.

[20] CyberArk. "Poison Everywhere: No Output from Your MCP Server is Safe." CyberArk, 2025.

[21] Wiz. "MCP and LLM Security Research Briefing." Wiz, 2025.

[22] The Hacker News. "Anthropic's MCP Protocol Has a Design Vulnerability That Could Affect Millions." The Hacker News, April 2026.

[23] The Register. "Anthropic's AI Agent Connector Standard Has a Critical Design Flaw." The Register, April 2026.

[24] MCP Manager. "MCP Adoption Statistics 2026." MCP Manager, 2026.

[25] Red Hat. "Model Context Protocol (MCP): Understanding Security Risks and Controls." Red Hat, 2025.

[26] Digital Applied. "MCP Adoption Statistics 2026: Model Context Protocol." Digital Applied, April 2026.

[27] National Vulnerability Database. "CVE-2026-27124." NIST, 2026.

[28] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, February 2025.