



WHITE PAPER

THE CTO'S ESSENTIAL GUIDE TO SECURING MCP SERVERS: **FOUR AUDITS THAT CAN'T WAIT**

David Rivkin, Ph.D.

Brian Scott Glassman, Ph.D.




Introduction

Model Context Protocol (MCP) servers represent an excellent tool for enterprise AI integration, and when implemented correctly, they can deliver major productivity gains and enable AI systems to orchestrate tasks across multiple services with unprecedented efficiency. CTOs who enable AI systems to access enterprise services through Model Context Protocol (MCP) implementations bear the critical responsibility of conducting comprehensive security audits to ensure these integrations maintain robust security postures and do not introduce vulnerabilities or exploitable attack vectors into their organization's infrastructure. The rapid adoption of MCP, with over 15,000 servers now deployed worldwide, has outpaced security maturity, creating an urgent need for rigorous evaluation before production deployment. This whitepaper examines the four most critical security concerns that demand immediate attention: (1) Widespread Misconfiguration and Exposure, (2) Security Retrofitted as an Afterthought, (3) Schemaless JSON Without Type Safety, and (4) Vast Attack Surface. Each section provides actionable audit methodologies that CTOs can implement to identify and remediate vulnerabilities before they result in catastrophic security breaches.

To understand the scope of this challenge, we must first examine the most immediate threat facing organizations today.

Widespread Misconfiguration and Exposure

The most immediate and critical threat facing MCP deployments stems from widespread misconfiguration and network exposure vulnerabilities. Research analyzing over 7,000 publicly accessible MCP servers reveals that hundreds are vulnerable to the "NeighborJack" exploit, which exposes these servers to any actor on the same local network. Approximately 70 servers exhibit severe compound flaws, combining unchecked input handling with excessive permissions, that enable attackers to achieve complete host machine compromise. With over 15,000 MCP servers now deployed globally since the protocol's late 2024 introduction, the attack surface has become enormous. This isn't a theoretical concern: the Backslash Security analysis discovered these vulnerabilities in live production systems, with many servers left entirely unprotected due to inadequate setup or missing authentication mechanisms. Moreover, the fundamental problem is compounded by MCP's role as a proxy layer that inadvertently obfuscates the client-side actor, making it difficult for security teams to identify and trace malicious activity back to its source.



- • •
- • •
- • •
- • •
- • •
- • •
- • •


Audit Approach: Given these vulnerabilities, CTOs must implement a systematic audit strategy immediately. Organizations should conduct network scans to identify all MCP servers accessible beyond localhost (127.0.0.1), specifically testing for NeighborJack vulnerability on local network interfaces. Review each server's configuration for proper input validation controls, audit the permission levels granted, and verify that authentication mechanisms are not just configured but actually enforced. Examine firewall rules and network segmentation to ensure MCP workloads are properly isolated, and conduct penetration testing by attempting unauthorized connections to validate that access controls function as intended under real-world attack conditions.

While misconfiguration represents immediate tactical risks that can be addressed through proper deployment practices, the protocol's fundamental design reveals deeper strategic vulnerabilities that cannot be easily remediated.

Security Retrofitted as an Afterthought

MCP's fundamental architecture reveals a protocol rushed to market without essential security features baked into its design, a decision that has forced enterprises to retrofit critical protections after deployment. OAuth 2.1 support, a cornerstone authentication mechanism, was only added in the March 2025 revision after organizations had already adopted the protocol in production environments. Even this belated addition only applies to HTTP transports, leaving the stdio transport dependent on environment variables for credential management, a 1970s era approach that lacks the granular access control required for modern enterprise security. Furthermore, the protocol similarly lacks built-in distributed tracing, correlation IDs, capability-based authorization beyond rudimentary tool annotations, standardized audit trails, deadline propagation to prevent cascading failures, and rate limiting or quota management at the protocol level. As a result, this pattern of adding airbags after the crash, as the article aptly describes it, has left enterprises in an impossible position: they adopted MCP based on promises and hype rather than operational reality, and now must cobble together a constellation of third-party libraries to fill gaping security holes that should have been addressed at the protocol's foundation.

Audit Approach: To address these architectural deficiencies, CTOs must first inventory which MCP transport types are deployed, with particular scrutiny on stdio transports that rely on environment variables for credentials. Verify that OAuth 2.1 is properly implemented and functioning on HTTP transports, and document all third-party libraries being used to compensate for missing protocol-level security features such as distributed tracing, audit logging, and rate limiting. Audit the maintenance status and security posture of these




- • •
- • •
- • •
- • •
- • •
- • •
- • •

third-party dependencies, assess whether the current patchwork implementation can meet compliance requirements under frameworks like NIST AI RMF or the EU AI Act, and establish incident response procedures specifically designed to address the unique gaps created by retrofitted security architectures. Beyond these architectural security gaps, the protocol's handling of data types introduces another critical layer of risk that can have life-threatening consequences.

Schemaless JSON Without Type Safety

The absence of enforced type safety in MCP's schemaless JSON architecture represents a fundamental security flaw that can lead to catastrophic real-world consequences. Unlike mature protocols that enforce strict type validation, MCP relies on optional, non-enforced hints that allow data type mismatches to pass through silently without failing cleanly. When an AI tool expects an ISO-8601 timestamp but receives a Unix epoch integer, the language model doesn't reject the malformed input; instead, it hallucinates dates and continues processing with corrupted data. This architectural weakness has direct life-and-death implications: in financial services, trades can be executed with incorrect decimal precision; in healthcare, patient data types get coerced incorrectly, potentially causing medication dosing errors; in industrial systems, sensor readings lose precision, leading to quality control failures. The problem is compounded by the fact that these errors don't surface immediately; they propagate through systems as seemingly valid data, making detection and remediation exponentially more difficult. Consequently, without a guaranteed contract between client and server, enterprises cannot ensure data integrity at the protocol level, leaving critical business logic vulnerable to unpredictable AI behavior based on misinterpreted information.

Audit Approach: To mitigate these type safety risks, CTOs must systematically test MCP integrations by deliberately introducing type mismatches: sending Unix epoch timestamps when ISO-8601 is expected, varying floating-point representations, and testing Unicode handling across different language implementations to observe whether systems fail cleanly or allow corrupted data to propagate. Review all MCP tool definitions to identify which rely on optional hints versus strict validation, examine transaction logs in financial systems for decimal precision anomalies, audit healthcare medication dosing outputs for type coercion errors, and implement additional type validation layers outside the MCP protocol itself. Establish monitoring systems capable of detecting when type mismatches occur and review incident logs for unexplained data errors that may stem from type safety failures.



- • •
- • •
- • •
- • •

These type safety failures become even more alarming when examined alongside the comprehensive range of attack vectors targeting MCP deployments.


- • •
- • •
- • •

Vast Attack Surface

- • •
- • •
- • •

MCP deployments face an alarming taxonomy of over 20 distinct attack vectors spanning three major categories, creating what the research describes as an "enormous" attack surface across the 15,000+ servers now deployed worldwide. Classical attack vectors include supply chain compromises through malicious Docker containers or Python packages, authentication bypass exploits, token passthrough without proper scope validation, tool poisoning via malicious metadata, privilege escalation through improper RBAC controls, man-in-the-middle interception, protocol exploits in fallback systems, model inversion attacks to reconstruct training data, and feedback loop exploitation. Building on these traditional threats, advanced attack vectors introduce even more sophisticated dangers: MCP server data corruption that compromises orchestration workflows, cross-agent contamination where malicious payloads spread through shared memory, the confused deputy problem where proxies use their own credentials on attackers' behalf, session hijacking enabling prompt manipulation, and context poisoning that tampers with data sources to manipulate AI outputs. The third category, prompt injection and jailbreak attacks, encompasses role-play manipulation, prompt obfuscation using typos and Unicode tricks, refusal suppression, multilingual exploits, multi-turn contextual attacks, automated attacks like AutoDAN, and output-level manipulation affecting token probabilities. The proxy nature of MCP compounds these risks by obscuring malicious actors, making detection significantly harder, while each new integration exponentially expands the attack surface and enables attack vectors to be chained together for increasingly sophisticated exploits.

Audit Approach: Given the breadth of potential threats, CTOs must implement comprehensive security testing across all attack categories, including supply chain verification through cosign-signed artifacts and SBOM scanning, container scanning with Prisma and Trivy, authentication bypass testing, token scope validation audits, RBAC privilege escalation assessments, and man-in-the-middle attack simulations to verify TLS 1.3 enforcement. Deploy automated penetration testing using OWASP ZAP and Burp Suite configured for AI-specific attack vectors, conduct extensive prompt injection testing covering role-play manipulation, obfuscation techniques, multilingual exploits, and automated attacks, and implement ML-based anomaly detection for unusual query patterns. Test for session hijacking vulnerabilities, context poisoning attempts, cross-agent contamination in multi-agent environments, confused deputy exploits, and model inversion attempts,



- • •
- • • while integrating real-time threat intelligence feeds (MISP, MITRE ATLAS) to
- • • maintain awareness of emerging attack patterns specific to AI and MCP
- • • deployments.

- • •
- • • While integrating real-time threat intelligence feeds (MISP, MITRE ATLAS) to
- • • maintain awareness of emerging attack patterns specific to AI and MCP
- • • deployments.

Additional Operational and Security Risks

Beyond the critical vulnerabilities of misconfiguration, retrofitted security, type safety failures, and expansive attack surfaces, MCP deployments face a constellation of additional risks that compound security challenges and operational complexity. The protocol's missing observability and debugging capabilities, lacking distributed tracing, correlation IDs, and proper error handling, mean that when security incidents occur, enterprises face nightmare scenarios where identifying a problem that would take 30 minutes with mature protocols like gRPC instead requires three days of manual log analysis. This operational blindness is exacerbated by the third-party library dependency problem, where critical security features must be cobbled together from a fragmented ecosystem of semi-compatible extensions of varying quality and uncertain maintenance commitments, forcing enterprise architects into impossible decisions about which authentication, tracing, or audit libraries to standardize across hundreds of developers.

Additionally, the absence of cost attribution or quota management at the protocol level creates financial exposure where organizations cannot determine which departments or specific tool calls drove tens of thousands of dollars in API usage, while simultaneously lacking mechanisms to prevent runaway costs. Language implementation fragmentation, where Python's JSON encoder handles Unicode differently from JavaScript's and float representations vary across implementations, introduces subtle incompatibilities that only surface under edge cases, creating security gaps that attackers can exploit. Finally, the lack of independent schema versioning means tool interfaces can change without warning, breaking all clients and potentially introducing security vulnerabilities when updates deploy inconsistently across distributed systems, leaving organizations unable to safely evolve their MCP integrations without risking system-wide failures.



• • • • • • **Conclusion**

• • • The integration of AI systems with enterprise services through the Model Context
• • • Protocol represents both tremendous opportunity and significant risk. As this
• • • whitepaper has demonstrated, CTOs must address four critical security concerns
• • • before deploying MCP servers in production environments: widespread
• • • misconfiguration and exposure that leaves systems vulnerable to immediate
exploitation, security features retrofitted as an afterthought rather than built into
the protocol's foundation, the absence of type safety that enables silent data
corruption with catastrophic consequences, and a vast attack surface
encompassing over 20 distinct threat vectors. The audit methodologies outlined
for each area provide a roadmap for identifying and remediating vulnerabilities
before they result in breaches, data loss, or operational failures. However,
security is not a one-time achievement; teams must conduct comprehensive
audits prior to initial deployment and establish processes for re-evaluation after
every configuration change, software update, or integration modification.
Ultimately, only through rigorous, continuous security assessment can
organizations harness the power of agentic AI while protecting their
infrastructure, data, and stakeholders from the inherent risks of this rapidly
evolving protocol.