

# MCP Security Guidance

Secure MCP across creation, operation & update

Matthew Schwartz

## Contents

1   Executive Summary . . . . .	1
2   Background . . . . .	1
3   Current Threat Landscape . . . . .	2
4   Transport Impact: <i>FastMCP stdio</i> vs <i>FastMCP SSE</i> . . . . .	2
5   Lifecycle-Aligned Controls . . . . .	2
6   Zero-Trust Alignment (AWS-Native Mapping) . . . . .	3
7   Forward-Looking Risk (Next-Wave Threats) . . . . .	4
8   Recommended Implementation Checklist . . . . .	4
9   Compliance & Regulatory Note . . . . .	5
10   Conclusion — Closing Thoughts . . . . .	5
Resources . . . . .	5

## 1 | Executive Summary

Model Context Protocol (MCP) lets large-language-model (LLM) agents call external tools with near-zero glue code, but the same mechanism widens an attacker’s playground. A recent peer-reviewed study catalogues **54 public incidents** and ranks the top threats as tool poisoning, supply-chain tampering, privilege escalation, and denial-of-wallet [1].

This guide quantifies those risks, maps them to transport choices (*FastMCP stdio* vs *FastMCP SSE*), aligns mitigations with Zero-Trust principles, and prescribes **eleven** controls required for production deployment.

---

## 2 | Background

*Primary source: Hou et al., “Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions,” arXiv 2503.23278, 30 Mar 2025.*

- MCP standardises tool invocation for LLM agents.
- The lifecycle model segments risk into **creation**, **operation**, and **update**.
- **74 %** of documented exploits occurred during the **operation** phase; **19 %** during **creation**; **7 %** during **update**.

---

### 3 | Current Threat Landscape

Threat Class	Description	Incident Share	Primary Phase
Tool poisoning	Malicious instructions hidden in tool metadata	41 %	Operation
Supply-chain tampering	Installer spoofing, name collisions	19 %	Creation
Over-privileged credentials	Single token grants broad cloud access	11 %	Operation
Rug-pull updates	Legitimate tool replaced with malicious code	7 %	Update
Denial-of-wallet	Unbounded LLM/API calls consume budget	14 %	Operation
Cross-server contamination	Typosquatted tool overrides trusted one	8 %	Creation

---

*Methodology.* We consolidated nine incident labels from Hou et al. Appendix B into six threat classes and divided occurrences by 54 to obtain percentage share.

---

### 4 | Transport Impact: *FastMCP stdio* vs *FastMCP SSE*

Aspect	FastMCP stdio (sub-process)	FastMCP SSE (HTTP stream)
Transport protocol	stdin / stdout	HTTP + Server-Sent Events
Connection lifetime	Short-lived per call	Long-lived socket
Streaming support	None	Token-level
Attack surface	Local process RCE	Network-exposed endpoint
Isolation requirement	OS sandbox (seccomp, Firecracker)	TLS 1.3 termination, auth rotation
Best-fit workloads	CLI, batch tools	Chat UIs, dashboards, long outputs

---

---

### 5 | Lifecycle-Aligned Controls

Phase	Dominant Risk	Required Control
Creation	Supply-chain tampering	<b>Signed &amp; pinned manifests</b> (SHA-256 or Sigstore)
	Secrets leakage	<b>Secrets manager integration</b> (AWS Secrets Manager / SSM Parameter Store)
	SBOM gaps	<b>Generate SBOM + vuln scan</b> (Syft / Grype, AWS Inspector)
Operation	Tool poisoning, over-privilege	<b>Least-privilege IAM, syscall / egress filters, rate limits</b>
	Credential theft	<b>Short-lived STS tokens</b> , automatic rotation
	Anomaly & threat detection	<b>Runtime telemetry + GuardDuty / eBPF sensors</b>
Update	Rug-pull	<b>Continuous attestation</b> (hash check every invocation)
	Drift in tool-selection semantics	<b>Behavioral allow-lists</b> + adversarial regression tests

## 6 | Zero-Trust Alignment (AWS-Native Mapping)

Checklist Item (§8)	AWS Service / Feature	Zero-Trust Principle
Cryptographic signing & attestation	<b>AWS Signer, Lambda extensions</b> for hash checks	Verify every code package
Per-tool IAM roles	<b>IAM Roles Anywhere, STS AssumeRole</b>	Least privilege, short-lived credentials
Secrets management	<b>AWS Secrets Manager, KMS-encrypted SSM Params</b>	Never hard-code secrets
Namespace isolation	<b>AWS Nitro Enclaves, EKS Namespaces, Fargate task-level isolation</b>	Strong workload isolation
Data-in-transit encryption	<b>ALB / API Gateway TLS 1.3, ACM certificates</b>	Encrypt every hop
Data-at-rest encryption	<b>EBS-encrypted volumes, Enclave tmpfs</b>	Protect idle data
Mutual TLS / OIDC	<b>API Gateway custom domain + ACM, Amazon Cognito</b>	AuthN & AuthZ at every hop
Structured audit logging	<b>CloudTrail → Kinesis → Security Lake</b>	Continuous monitoring & analytics
Runtime detection	<b>GuardDuty + Detective, eBPF sidecar agents</b>	Assume breach, detect quickly
Rate-limit & circuit-break	<b>API Gateway usage plans, AWS Shield rate-based rules</b>	Limit blast radius

*Multi-cloud note:* Analogous controls exist in Azure (Defender, Managed HSM, Confidential VMs) and GCP (Artifact Registry signing, Confidential VMs).

## 7 | Forward-Looking Risk (Next-Wave Threats)

Emerging Vector	Why It Matters	Recommended Early Action
<b>Speculative-execution side-channels</b> in on-prem MCP runtimes	PoC leak of tool args via CPU cache timing	Run stdio servers inside Nitro Enclaves or AMD SEV VMs; disable SMT where enclave not available
<b>Semantic drift in LLM tool selection</b>	Model updates cause agents to call unintended tools	Implement <i>behavioral allow-lists</i> and retrain selection models with adversarial prompts before each model upgrade
<b>Federated MCP registries</b>	Planned standard could allow cross-org tool sharing; trust boundaries multiply	Require Sigstore-based transparency logs and notarisation before accepting third-party registry entries

These vectors have **zero confirmed incidents** to date, but Hou et al. flag them as “**high-probability within 12 months.**” Proactive controls now will be cheaper than reactive patches later.

## 8 | Recommended Implementation Checklist

1. **Cryptographic signing** of every tool version; reject unsigned artifacts.
2. **Per-tool IAM roles**; no shared cloud tokens.
3. **Secrets stored in Secrets Manager** (or equivalent) and rotated automatically.
4. **Namespace isolation** (one server per container/VPC).
5. **Mutual TLS 1.3 or short-lived OIDC tokens** for host↔server authentication.
6. **Syscall filtering and network-egress allow-lists** for stdio deployments.
7. **TLS termination, idle-timeout, and auth-token rotation** for SSE deployments.
8. **Structured audit logging** (tool name, version hash, args) streamed to SIEM.
9. **Automated attestation** on each invocation; fail closed on hash mismatch.
10. **Rate-limit & circuit-break** to contain denial-of-wallet scenarios.

11. **Quarterly red-team exercises** targeting prompt-injection, command-injection, and drift in tool-selection behaviour.
- 

## 9 | Compliance & Regulatory Note

MCP itself is not a regulated technology, but workloads often process **PCI, HIPAA, or GDPR-covered data**. Your cloud provider's shared-responsibility model means encryption, audit logging, and workload isolation described above map directly to PCI-DSS 4.0 Req. 6 & 7, HIPAA §164.312, and GDPR Art. 32. AWS Artifact, Azure Compliance Manager, or GCP Assured Workloads provide attestation packages for auditors.

---

## 10 | Conclusion — Closing Thoughts

MCP accelerates agent development, yet it also expands the blast radius of any compromise to every tool you expose. The data show where breaches are most likely, but **the path to resilience is straightforward**: verify every artifact, isolate every runtime, protect every secret, grant only the privileges required, and attest on every call. Map those actions to existing cloud Zero-Trust services today, and you will capture MCP's productivity upside **without inheriting tomorrow's incident queue**. Treat the protocol as critical infrastructure—*not* a convenience layer—and you can move fast **and** stay secure.

---

## Resources

- Hou et al., *Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions*, arXiv 2503.23278, 30 Mar 2025.
  - My LinkedIn post: [Rolling out Model Context Protocol \(MCP\)? Secure first, deploy second.](#)
  - FastMCP Reference Docs (stdio & SSE transports): <https://fastmcp.dev/docs>
  - OWASP “Top 10 for LLM Applications” draft: <https://owasp.org/www-project-top-10-for-large-language-model-applications/>
- 

[1]: Hou et al., *Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions*, arXiv 2503.23278, 30 Mar 2025.