MCP Security Adoption Guide
March 2026

MCP Security Guide

Written by Adib Saikali, Distinguished Engineer
VMware Tanzu

Table of contents

Introduction to Model Context Protocol

Teams that are considering building agentic applications need to consider their approach to data and service integrations up front. Model Context Protocol is emerging as the de facto specification for services and data integrations for agentic applications. MCP is a relatively new specification, and this guide introduces security considerations that are important to explore further as part of this process.

What is Model Context Protocol?

The Model Context Protocol (MCP) is an open standard designed to allow AI models to securely and consistently discover, connect to, and utilize external resources. MCP is a transport layer, and it can connect to multiple types of interfaces — such as APIs, databases, workflows, and even other applications — enabling agents to execute actions beyond simple text generation. MCP consists of three architectural components: hosts, clients, and servers.

Host Application: An MCP host is the application for example a coding agent, or an integrated development environment (IDE) that receives the inquiry from the end user. To complete the user query, the host may determine it does not have the knowledge, skills, or tools that are needed to complete the end user request, at which point the MCP host will reach out to an MCP client to initiate a connection to the necessary target service or data source.

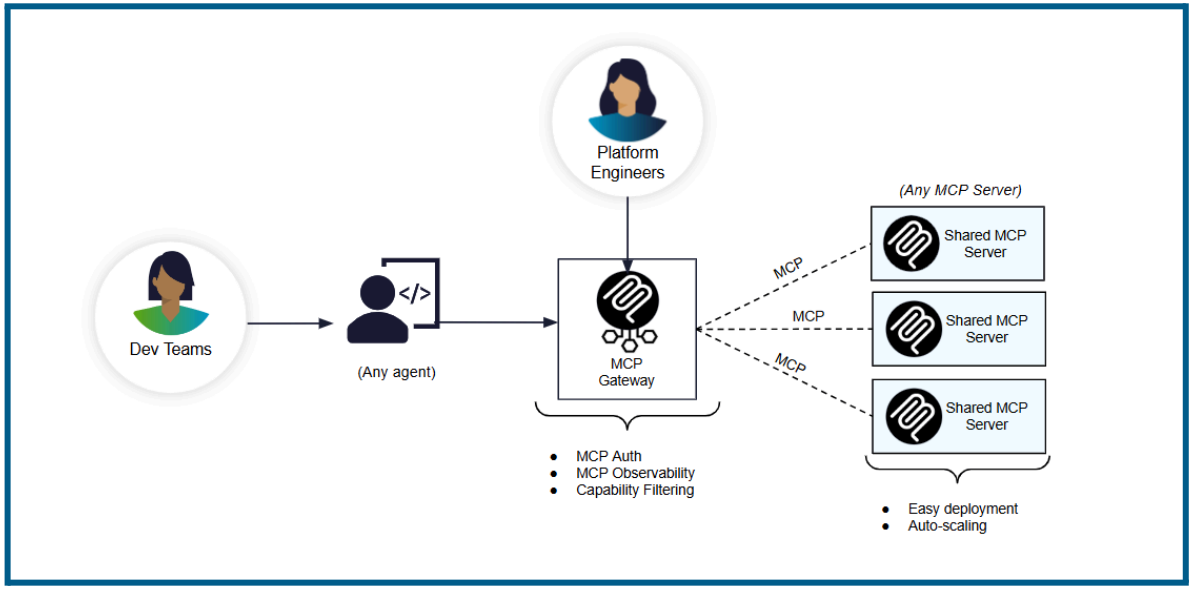
MCP clients: Acting as the session manager for MCP hosts, clients broker the connection to the target service or data source. They have the capability to manage the session lifecycle, including restarting or terminating the connection as needed during the process of obtaining feedback.

MCP servers: MCP servers are positioned in front of the external service or data source that bring context to the original inquiry. Their role is to pass information from the services or data sources through the MCP transport layer to the MCP client. The client then relays this information back to the MCP host, which finally delivers it back to the model where the end user initially posed their question.

Why do I need to consider MCP?

MCP servers enhance the efficiency of AI agents by clearly advertising the available tools and information within a service or data source. This targeted approach is more efficient, allowing the agent to select only the relevant server and preventing the agent's context window from being overloaded with unnecessary APIs during exploration. By providing a clear menu of options, MCP servers reduce the need for unstructured, freeform exploration that can quickly consume context capacity.

For enterprises, MCP gateways are a key enabler, offering a governed and auditable method for integrating agents into real systems. This eliminates the need to hard-code and maintain custom connectors, such as individual APIs.



This illustration shows how MCP integrations can be governed by an MCP gateway.

However, MCP has known challenges. While MCP is more agent friendly, it does have known security concerns that need to be addressed.

MCP security challenges

MCP is rapidly becoming a standard for connecting advanced AI systems to enterprise applications and data. Teams are beginning to deploy MCP servers to integrate AI models with internal systems. While this enables powerful new workflows, it also introduces significant security risk.

An MCP server bridges AI models and corporate data, creating a powerful integration point — but also a potential pathway for unintended, autonomous interactions. Without strong guardrails, an AI agent could issue actions or data requests that were never explicitly authorized or anticipated, performing privileged operations, accessing confidential information, or influencing internal workflows in unpredictable ways.

Because these MCP servers directly connect AI systems to sensitive corporate data, even a small mistake can have major consequences. A poorly secured server isn't just a technical flaw — it can behave like a rogue insider, using legitimate access paths to perform unintended or unauthorized actions.

This guide provides information that should be considered a mandatory starting point for any team deploying or integrating a remote MCP server, whether:

- Developing a custom MCP server that connects to internal tools, or
- Onboarding a third-party MCP server provided by an external vendor.

This guide establishes a security policy framework for MCP deployments. It enables teams to analyze risk, confirm adherence to mandatory controls, and maintain a defensible security posture across all MCP integrations — ensuring that every server connection upholds the protection of corporate data and infrastructure.

Determining an MCP server's access tier

Before applying security controls, each team must determine the access tier of its MCP server — that is, the level of identity verification and authorization required for it to perform its intended functions. This assessment defines the server's trust boundary and determines which security policies in this guide apply.

Access Tier 1 – Open Access Servers

Some MCP servers work entirely with data that does not require authentication, whether it is publicly available or broadly accessible within the company.

> A good example is a Meeting Planner MCP Server that helps employees coordinate cross-regional meetings by referencing public holidays and general travel guidance. Employees might ask questions such as “Which countries have overlapping public holidays next month?” or “What’s the best time to schedule a meeting with our teams in Munich and Bangalore?”

Because this type of server handles only non-confidential information, it does not require user authentication and can be safely shared across the enterprise. It must still operate in a secure and monitored environment consistent with your IT operational standards.

Access Tier 2 – Group Access Servers

Some MCP servers are designed to support the work of a specific department or organizational group.

> *For example, a Sales Insights MCP Server might help regional sales teams analyze pipeline data, generate opportunity reports, or access shared materials relevant to their division. Only employees who belong to the authorized group can use the server’s tools or view its results. The MCP server checks the user's identity to establish group membership, before executing the request against the back end.*

These servers operate on data and actions authorized for a defined role or organization, not for specific employees. Access is managed through group membership, ensuring that departmental information remains properly scoped and secure.

Access Tier 3 – User Access Servers

Some MCP servers perform actions or retrieve information that are tied directly to a specific employee's identity. They act on behalf of authenticated users and connect to systems that hold personalized or sensitive data, or that rely on individual user permissions to control access.

> For example, an Employee Assistant MCP Server might let employees review upcoming time off, submit expense reports, or check reimbursement status. Similarly, a GitHub MCP Server could allow engineers to list their assigned issues, update repositories, or review pull requests.

Because these servers handle identity-specific operations, they represent the highest sensitivity level in the MCP model and are subject to your organization's strictest requirements for identity assurance, access control, and data protection.

Authentication

For both Group Access and User Access MCP servers, authentication is required to determine who is making a request and to ensure that only authorized users and systems can interact with the server. Authentication establishes the foundation for all downstream authorization and auditing controls. MCP servers must rely on standardized, enterprise-grade identity protocols rather than implementing custom authentication logic. MCP servers act as OAuth 2.1 resource servers, validating tokens issued by trusted enterprise authorization servers. The server's responsibility is to verify that each request originates from a legitimate, authenticated user or service within the approved identity domain.

Standard MCP server security checklist	
	Use OAuth 2.1 standards for authentication, implemented through a trusted framework, such as Spring Security OAuth2 Resource Server (maintained by IT organization) or another enterprise-grade OAuth 2.1 library.
	Accept access tokens only from approved and trusted enterprise authorization servers.
	Verify that the OAuth library is securely configured to perform full token validation and reject any invalid or expired credentials.
	Require TLS for all communication channels.

Identity propagation and delegation

Once an MCP server authenticates a request, it must decide how to represent that identity when performing operations on connected back-end systems. This ensures that actions are executed under the correct authorization context — either as the end user or on behalf of a group or service account — and prevents privilege escalation or impersonation errors.

Propagation models

Service Account Delegation (Group Access Servers)
For Group Access MCP servers (Tier 2), actions are typically authorized at the team or department level.

The MCP server authenticates each user to verify their identity, determine their group membership and role, and make authorization decisions accordingly. It then performs the resulting operations on back-end systems using a service account credential rather than the individual user's credentials. This approach simplifies integration and avoids complex token-exchange flows, but it limits per-user traceability since downstream systems see only the service account.

User Identity Propagation (User Access Servers)
For User Access MCP servers (Tier 3), actions must reflect the specific permissions of the authenticated user. The MCP server not only authenticates the user but also propagates that identity to back-end systems so that authorization decisions continue to apply at the user level. This is typically implemented by presenting a user-scoped access token or performing a token exchange in compliance with RFC 8693 (OAuth 2.0 Token Exchange). This model enforces fine-grained, user-level access control but introduces additional complexity and requires strict validation of token audiences and scopes to prevent token passthrough vulnerabilities.

Back-End-Issued and Per-Request Verified Token (Stateless Single-System Integrations)
Some MCP servers are designed to work exclusively with a single back-end system that already integrates with the enterprise identity provider — such as GitHub Enterprise, Jira, or another SaaS product federated with the corporate IDP.

In these cases, the back end itself issues the access token after authenticating the user through corporate SSO. The MCP server acts only as a transport and policy layer for that back end and does not introduce a separate trust boundary.

Because the back end is already protected by enterprise authentication, the MCP server may safely forward the back-end-issued token without performing token exchange, provided that:

- The token is issued by the back-end system's own authorization server, not directly by the enterprise IDP
- The MCP clients are known trusted clients, where the user already has a cli or other tools that have a token to access the back end anyway
- The MCP server interacts only with that back-end system and remains completely stateless. It must not retain session data, cached responses, or user-specific context between requests, as doing so could result in incorrect or cross-user responses.
- On every request to the back end, the back end validates the token and applies authorization decisions

Because these tokens are often opaque, the MCP server cannot validate them directly. Instead, it must rely on the back end's response to determine token validity, treating any authentication failure from the back end as a signal to return a 401 Unauthorized status that follows the rules of OAuth 2.0 Protected Resource Metadata (RFC 9728) specification.

This pattern is only safe when the back-end system itself enforces proper token validation, expiration, and audience scoping. If the back end does not perform these checks, or if the MCP server interacts with more than one back end, a full token-exchange model (per RFC 8693) must be used instead.

Choosing a propagation model

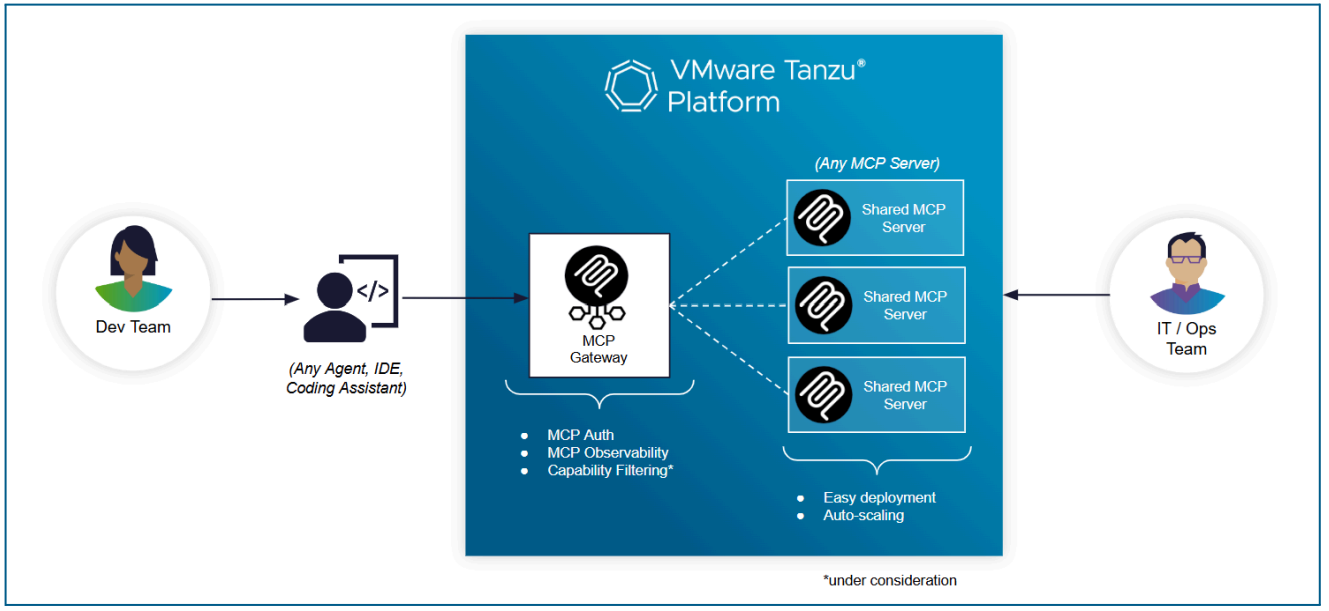
Selecting the right identity propagation model depends on the MCP server's access tier and the trust boundaries involved:

- Service Account Delegation – For Group Access servers (Tier 2) that act on behalf of a team or department. The MCP server authenticates the user, determines group membership, and executes back-end operations using a shared service account with minimal scoped, auditable privileges.
- User Identity Propagation – For User Access servers (Tier 3) that perform actions tied to individual users. The MCP server propagates each user's identity — typically through a secure token exchange — to ensure that downstream authorization decisions remain user-specific.
- Stateless Back-End-Issued Token – For single-system integrations where the back end itself issues tokens and enforces enterprise authentication. The MCP server forwards the token transparently, remaining stateless and deferring all token validation to the back end.

Teams should select the simplest model that meets functional requirements while maintaining clear accountability and least-privilege access.

Governed and auditable MCP with VMware Tanzu Platform

Tanzu Platform is the optimal environment for running MCP servers because it is an AI-ready, pre-engineered, platform-as-a-service (PaaS). As the creators and maintainers of the Java SDK for MCP, the Tanzu team has integrated features that streamline agentic application delivery while mitigating AI risks. The platform simplifies security, monitoring, and compliance by including middleware components like an MCP gateway for streamlined AuthN/Z, observability, and governance that can support organizations looking to comply with NIST 600-1 AI security and risk management standards. Furthermore, Tanzu Platform boosts reliability through MCP server auto-scaling, allowing for faster and more dependable delivery of agentic applications. While framework agnostic, Tanzu Platform is best suited for use with the agent-compatible Spring framework.



If you would like to learn more about how VMware Tanzu Platform can help with your MCP security, contact your VMware sales representative today.

References

- Model Context Protocol Authorization Specification (version 2025-11-25): Authorization - Model Context Protocol
- Model Context Protocol Security Best Practices: Security Best Practices - Model Context Protocol

Adib Saikali is a Distinguished Engineer at VMware Tanzu, focused on helping large enterprises design and build AI-enabled platforms and applications using Spring. His work centers on AI agent architectures, agentic control loops, API design for autonomous systems, and enterprise governance for AI in production. Adib is the author of Software Security for Developers and brings over 25 years of experience across startups and global enterprises in roles spanning developer, architect, and CTO.