STACKLOK

How to Secure your Model Context Protocol Estate

Model Context Protocol (MCP) is exploding; while MCP was introduced in November 2024, it's expected that by the end of 2026, 75% of enterprises will be using MCP servers.

The rapid adoption of MCP comes at the cost of hard-won security best practices. Some of the early use of MCP is regressive from a security perspective and ignores commonly accepted practices and standards. This paper outlines ten practical steps that an enterprise can take to shore up its MCP estate.

Contents

Intro: The “S” in MCP Stands for Security

A common joke at the expense of MCP is that the “S” stands for security. When MCP was first introduced in November 2024, the focus was on interoperability and usability, and early adopters were expected to run MCP in controlled environments.

Increasing interest in and adoption of MCP has created new pressures. In 2025 alone, thousands of MCP servers were published. And that means that developers have the potential to ‘npm install’ random stuff off the internet. Malicious actors rarely miss these kinds of opportunities, and they’re already exploring MCP’s attack surface.

All that said, there’s been important progress toward better securing MCP. There’s a proper governance structure in place now, and in recent releases, that team has adopted an improved authorization specification based on modern OAuth 2.1 standards, clarified security best practices for MCP server authors, and more. But while there’s momentum toward proper governance structures, in practice it’s still the wild west as organizations and MCP vendors roll their own policies and approaches.

And that’s why we’re here. Ultimately, the developer and/or the enterprise is accountable for securing their MCP estate, and we’re going to walk you through the necessary considerations and steps. So dig in, and should you ever have questions, just get in touch via our Discord channel or directly at hello@stacklok.com.

Current Security Concerns

Let’s first set forth the most prominent security concerns we have to overcome. Some of these are tried and true tactics that malicious actors have transposed into the world of MCP; other concerns are specific to MCP and require new solutions.

Prompt injection

An attacker can craft inputs that trick a client into invoking a tool in unauthorized ways, or into revealing sensitive information retrieved via MCP. There could be exposure of customer PII that violates GDPR.

Tool poisoning

A MCP server's description, schema, or behavior can be manipulated so that the client is misled about what it does; instructions could trick a client into exfiltrating data, escalating privileges, or going rogue.

Weak provenance

There's no universal concept of provenance for MCP servers, so it's often difficult (or impossible) to know who is behind a given server. Provenance helps an enterprise assess trustworthiness, and it can suggest the frequency with which a MCP server will be updated. Without this information, enterprises are flying blind.

BYO-server

With thousands of MCP servers available, it's hard to know which ones to trust. It's reasonable to assume that if access is not controlled, some clients will be connected to low-quality or infected servers.

Shadow AI

When employees are using clients and/or accessing context without central controls, it introduces the risk of data leakage, compliance gaps, IP leakage, and more. And when employees are using their own AI agents and MCP servers, they can unknowingly expose sensitive data like passwords and PII with a simple copy-paste

Misconfiguration

When employees are allowed / expected to configure their own MCP servers, the inconsistencies across a team or enterprise could create inconsistent outputs, inaccuracies, and mistrust.

Over-privileged access

If a client is granted broad authorization, it may access more data or perform more actions than intended. This can surface sensitive details to non-permissioned employees, or a client could exfiltrate data.

Exacerbating the above concerns is the lack of observability of MCP. Without good visibility or auditability of usage, server health, and more, a developer or enterprise can find themselves reacting to challenges. Fortunately, there are some proactive measures that can be taken to stay ahead of concerns, both old and new.

Ten Steps to a Secure MCP Estate

Step 1: Curating a Trusted Registry

As noted earlier, thousands of MCP servers have been published in a number of different languages and with wide variance in quality. How do you know which MCP servers to trust and use? An official MCP registry is in preview, but it's likely to include more MCP servers and more tools than you would ever need, and that could confuse or overwhelm a user.

One answer is to curate your own registry of trusted MCP servers. In particular, enterprises can put together a registry of MCP servers they know employees need and pre-assign MCP servers to specific roles. With the right underlying control plane, an enterprise can group multiple MCP servers to tackle specific tasks.

Step 2: Isolating Servers

It's not always clear how an MCP server is communicating with clients and other MCP servers. And unfortunately, an MCP server could be interacting with other MCP servers or spilling sensitive information behind your back; perhaps intentionally when a server has been compromised or unintentionally when a server has been poorly designed or configured.

We've seen this movie before, and it ends with containers. By containerizing each MCP server, you introduce a simple and useful form of isolation, immediately limiting the blast radius of any issue. You can also apply network isolation to verify and enforce which external hosts an MCP server can talk to. With the right solution, you can define permission profiles and restrict traffic to internal domains or specific cloud services. If maximum isolation is needed, you can apply a 'none' designation that disallows outbound communication altogether.

Step 3: Preconfiguring MCP servers

One of the security concerns raised above stems from inconsistent configuration or misconfiguration of hosts, transports, communications, credentials, and more. An enterprise with hundreds or thousands of employees will undoubtedly see differing configurations across users and clients. Delegating configuration choices introduces friction for employees and a nightmare for operations.

A solution is to preconfigure all these pieces so that an employee only needs to connect to an MCP server and move on. Again, this requires an MCP control plane, and the preconfiguration step is a natural complement to the curation of a trusted registry.

Step 4: Implementing AuthN to Z

First, let's be clear that authentication (authN) and authorization (authZ) are separate concerns. Authentication verifies who is making a request, and authorization determines what that identity is allowed to do.

MCP addresses authentication with OAuth 2.1, but offers only a recommendation, not a requirement; as a result, most MCP servers do not include authentication. And MCP does not address authorization; it expects servers will validate access tokens.

You can potentially avoid incremental layers of authorization by using something like OpenID Connect (OIDC), which is based on OAuth, to eliminate custom security code and to integrate with your existing identity provider. And centralized controls can put you in a position to apply finer-grained permissions and least privilege access.

There is much more to explore when it comes to authentication and authorization; more mature perspectives and solutions are needed. For example, the current state of the art involves creating a system where authentication is handled in the toolchain. That seems reasonable until you realize that all agentic actions performed by the client are incorrectly viewed as being performed by the user. In summary, we recommend you dig into all things auth with a team (internal or external) that deeply understands both MCP and security best practices.

Step 5: Managing Secrets

The MCP spec doesn't direct server authors how to handle secrets. As a result, it's up to each author to handle secrets, and to date, that has meant that many users end up putting secrets in plain text configuration files or environment variables that are easy to leak.

The clear action is to integrate your MCP estate with your existing secrets management solution (or to adopt such a solution if you haven't already). For example, you can integrate with 1Password, the macOS Keychain, or explore an integration of a control plane like ToolHive with a secrets manager like HashiCorp Vault.

Step 6: Sanitizing Inputs and Outputs

Each MCP server includes one or more tools, and every tool needs to be sanitized. For example, you'll want to validate parameter-type checking, length limits, schema libraries, format validation, SQL injection prevention, and command injection prevention.

The emerging way to address this need is the MCP Gateway. A gateway intercepts a tool call request before it reaches the tool and checks formats, limits scope, and removes suspicious inputs. It works similarly to a web application firewall. The gateway can be one more check on outputs, redacting secrets or PII from tool responses before they are passed back to a client.

Step 7: Server provenance and signing

Returning to an earlier point, there are already thousands of MCP servers, and it's difficult to assess trustworthiness.

Here, we can apply learning from the world of open source software; one of the ways to assess package trustworthiness is by determining provenance. In short, identify who is behind the package (creators, maintainers, and their respective relationships with other — trusted or untrusted — entities). Solutions are emerging that include parallel assessments of MCP servers.

There's also the potential for solutions like Sigstore to be used for a signed attestation of MCP server provenance. Enterprises will increasingly look for these indicators of trustworthiness and quality as they curate registries.

Step 8: Server update and version management

Like any piece of software, MCP servers may contain bugs or vulnerabilities; the more popular servers are regularly updated to shore up these gaps and to add capabilities. Attackers could exploit an outdated server to escalate privileges, bypass isolation, or exfiltrate data.

For these reasons, you'll want to ensure you're using the most up-to-date version of an MCP server. This is where you can use your curated registry to pin versions, recording and verifying the exact MCP server version and its permissions. Set up alerts when new versions are made available and define SLAs for patching MCP servers (just like you do for OS or container updates)

Step 9: Ensuring Observability and Auditability

If you've made it this far, you're thinking hard about the security of your MCP estate, and so undoubtedly, you want observability and auditability. Specifically, you'll want to log all tool calls and responses, monitor suspicious patterns, track authentication failures, and receive alerts on unusual behavior.

A growing handful of MCP solutions tout observability on their websites, but there's no need to add more observability tools when you can simply integrate with your existing solution. There's an opportunity to ensure alignment with your Security Operations Center and to build shared visibility of your MCP estate.

Step 10: Testing your MCP server packaging pipeline

The above steps pertain to discovery and curation of existing MCP servers. Many are also relevant to the inevitable case where you're building your own MCP server. When you get to that point though, you'll need to consider how to secure your MCP server packaging pipeline.

> For example, you'll need to ensure that: (i) builds are reproducible, (ii) network policies limit egress, (iii) versioned artifacts are pushed to private registries, (iv) fast rollback and drift detection are in place and (v) a whole lot more. Designing and tuning that pipeline is a bigger task that demands more energy; this is precisely when you should think about leaning on expert guidance.

Conclusions

There's a ton of energy around the Model Context Protocol. Thousands of MCP servers have been published in less than a year, and vendors are touting new tools that make it easier and faster to create more servers. In short, the excitement is accompanied by some chaos.

Conditions like this tend to attract malicious actors, and we expect to see an increasing volume of tactics and attacks as the surface area grows. And as enterprises rapidly adopt MCP, there will be internal challenges and the potential emergence of shadow AI.

Progressive enterprises and technical leaders will get ahead of this wave; they'll take the steps we've outlined here (and more) to solve the security challenges of today, and set themselves up to tackle new risks that emerge tomorrow. If you're working to put MCP into production, you'll want to find a couple of foundational tools to build with and an expert team you can trust for guidance. If those are conversations we can support, please get in touch with us at hello@stacklok.com.