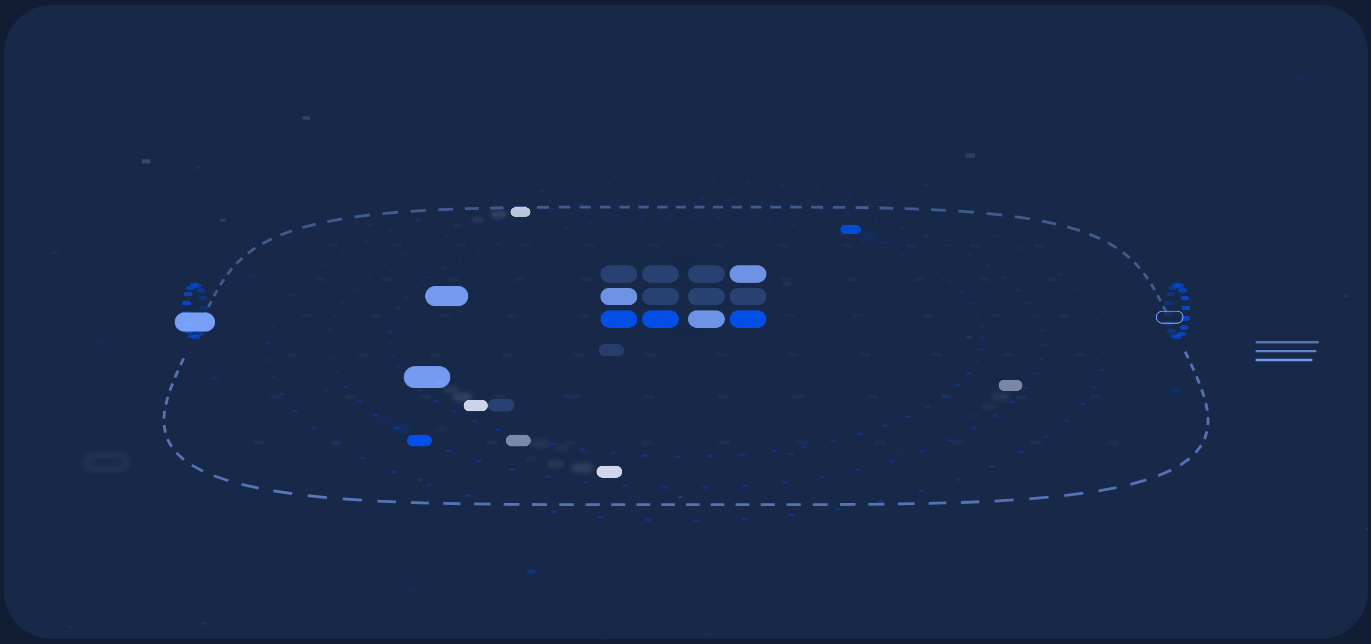
Rillor

EXPLAINER

MCP security checklist: servers, scopes and pinned tool descriptions

MCP security best practices as one checklist for superintelligence (SI) agents: which servers to allow, which scopes to grant and how to pin and re-approve tool descriptions.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

MCP security best practices fall to the people who deploy agents and servers, because the Model Context Protocol (MCP) specification says the protocol "cannot enforce these security principles at the protocol level" [1].

- The MCP specification says tools "represent arbitrary code execution" and that descriptions of tool behavior, such as annotations (hints that mark a tool read-only or destructive), "should be considered untrusted, unless obtained from a trusted server" [1, 15].
- A listing on the official MCP Registry shows which GitHub account or domain published a server, and the registry tells consumers to "assume minimal-to-no moderation" [2, 3]. The MCP project's local server guide asks users to know where each server comes from and to pin its version [4].
- Under revision 2026-07-28 of the MCP specification, an MCP server must accept only access tokens issued for it, and a client must name the server it wants a token for with an OAuth resource indicator [5]. The MCP project's security guide lists wildcard scopes such as `*`, `all` and `full-access` among common mistakes [6].
- Tool names and descriptions are instructions the model reads. A server can hide directives in them, which is called tool poisoning, or change them after a user approved the server, which is called a rug pull [4, 7].
- MCP itself does not require clients to pin tool definitions [4]. OWASP's MCP cheat sheet asks for a cryptographic hash of each complete reviewed definition, including its input schema and its annotations (hints such as read-only), and a new review when it changes [8]. OWASP adds that a hash catches changed metadata and misses changes to the server's code behind it [8].
- In the MCPTox benchmark, poisoned tool descriptions on 45 live servers succeeded against 20 agent settings at an average rate of 36.5%, and no setting refused more than 3% of the attacks [9].
- A 2025 study of 1,899 open-source MCP servers found tool poisoning in 5.5% of them [10]. A survey of 5,205 open-source server repositories found that 53% rely on static API keys or personal access tokens and 8.5% use OAuth [11].
- Tool results are untrusted input, including results from approved servers [8]. Revision 2026-07-28 of the MCP specification deprecates Sampling, the feature through which a server asks the client for a model completion [12, 13].

Background

What MCP connects

MCP connects language model applications to tools and data through three roles [1]. Hosts are "LLM applications that initiate connections", clients are "Connectors within the host application" and servers are "Services that provide context and capabilities" [1]. A server offers resources, prompts and tools, and tools are functions the model can call [1]. Revision 2026-07-28, the current specification, removed protocol-level sessions, the `Mcp-Session-Id` header and the `initialize` handshake, so every request carries its own protocol version and client capabilities [12]. The project reported "close to half-a-billion downloads a month" across its Tier 1 SDKs when it released the revision [14].

What the specification leaves to deployers

Hosts "must obtain explicit user consent before invoking any tool", and users "should understand what each tool does before authorizing its use" [1]. The protocol itself enforces none of this. Implementers should build consent and authorization flows, apply access controls and data protections, and follow security best practices in their integrations [1]. Authorization is optional. HTTP transports should follow the authorization specification, and stdio servers should "instead retrieve credentials from the environment" [5]. Pinning tool definitions sits outside the protocol too: "The protocol does not require this" [4].

Revisions that changed security

Tool annotations, hints such as `readOnlyHint` and `destructiveHint`, "shipped in the 2025-03-26 spec revision" [15]. Revision 2026-07-28 has clients check a present `iss` against the recorded issuer before redeeming an authorization code, which the project says "closes an authorization-server mix-up hole" [12, 14]. It requires clients to key stored credentials by issuer, adds the request headers `Mcp-Method` and `Mcp-Name` on Streamable HTTP, deprecates Dynamic Client Registration in favor of Client ID Metadata Documents, and deprecates Roots, Sampling and Logging [12]. Deprecated features keep working for "a minimum twelve-month deprecation window" [12]. The project's security guide and OWASP both keep advice for older, session-based revisions [6, 8].

This report is for teams that connect SI agents to MCP servers, teams that write servers and the reviewers who approve them.

Method

What was examined. The checks that apply when an agent connects to MCP servers: which servers to allow, how they run, how they authenticate and authorize, how tool definitions are pinned, how results, sampling and approvals are handled, and what to log. For each check, the source that sets it, its strength and any published case or measurement behind it.

Sources. Six pages of revision 2026-07-28 [1, 5, 12, 16, 17, 18], the project's security guides, registry pages and blog [2, 3, 4, 6, 14, 15], OWASP and CoSAI guidance [8, 19, 20], two OAuth RFCs [21, 22], published attacks and commentary on them [7, 13, 23, 24, 25, 26], vulnerability records, a vendor's write-up and a news report [27, 28, 29, 30, 31], scans and surveys [10, 11, 32, 33], two benchmarks [9, 34] and one server's documentation [35].

How. Requirements were read in each source's own text, with keywords (MUST, SHOULD, MAY) kept as written. Figures keep the source's base and date. The benchmarks name the models they tested, and this report gives only aggregate figures. All pages were viewed on 7 November 2026.

Findings

1. Servers: an inventory, an allow-list and a pinned version

The project's local server guide asks users to "remove the servers you no longer use, and make sure you know where each remaining one comes from and that its version is pinned" [4]. For organizations it asks for a short allowlist of vetted servers "(with pinned versions)" and a continuous inventory [4]. The Coalition for Secure AI (CoSAI) asks for "allow-lists of approved MCP servers with documented security reviews", an inventory with each server's version, owner and purpose, and "automated discovery to detect shadow or unauthorized MCP deployments" [20].

Provenance. A registry name shows its publisher. Publishers who sign in with GitHub use names of the form `io.github.username/*`, and publishers who prove a domain use its reverse-DNS form, such as `com.example.*/*` [2]. The registry "does not make guarantees about moderation", removes malware and spam, and keeps "Servers with security vulnerabilities" [3]. "A listing on a registry is not an endorsement," the local server guide says [4].

Names. OWASP's cheat sheet warns that "typosquatting (e.g., mcp-server-filesystem vs mcp-server-filesytem) is a common attack vector" [8]. OWASP's Top 10 for Agentic Applications lists impersonation and typo squatting among its supply chain examples [19].

Versions and digests. Name a specific version in the install command [4]. A launcher such as `npx` or `uvx` pins only the top-level package, and its transitive dependencies are typically resolved fresh at install time. "Pinning a container image by digest (`image@sha256:...`) freezes the entire dependency tree" [4]. CoSAI asks for "dependency pinning with hash-based verification rather than version ranges" and for signatures checked against approved sources and signing keys [20].

Why. OSV records that the npm package postmark-mcp "turned malicious in v1.0.16 and exfiltrates email data via BCC" [30]. OWASP says it was "Reported as the first in-the-wild malicious MCP server on npm" and that it impersonated postmark-mcp [19]. The guide's reason for the pin is that it keeps a compromised future release from arriving through an auto-update [4]. OWASP also asks for a kill switch that can "instantly disable specific tools, prompts, or agent connections" [19].

2. Local servers run with the user's privileges

A local server "inherits your environment variables, can read anything your account can read" and can open outbound connections anywhere, and "None of that is governed by the protocol" [4]. "The stdio transport is not a sandbox," and because every configured server shares the same model context, "a malicious server can influence how the agent uses the tools of every other server" [4].

The project's security guide names three routes: "a malicious 'startup' command in a client configuration", "a malicious payload inside the server itself" and "an insecure local server that's left running on localhost via DNS rebinding" [6]. Clients that offer one-click setup must ask for consent and show the exact command,

without truncation, before running it. They should run it "in a sandboxed environment with minimal default privileges" and warn that "MCP servers run with the same privileges as the client" [6]. OWASP adds "Never allow web content or untrusted data to trigger MCP server installation" [8].

Isolation. The guide calls process isolation "the single highest-leverage mitigation in this guide", with a container that mounts only what the server needs and no network where none is needed, one flag, `--network=none` [4]. "Binding to 127.0.0.1 is not an authentication boundary," and allow-lists in a configuration file are "cooperative boundaries" [4]. OWASP says to keep sensitive servers (payment, auth, PII) apart from general-purpose ones [8].

Credentials. Stdio servers take credentials from the environment, outside OAuth [5]. The guide says to pass secrets per server, give each server its own token "with the smallest scope that works", keep long-lived secrets out of plaintext config and, going further, use "short-lived credentials issued at launch time" [4]. When a widely used local server turns out to be vulnerable, it says to "rotate every credential those configs passed to it" [4].

Developer tools. CVE-2025-49596, published in June 2025, let unauthenticated requests launch MCP commands through MCP Inspector versions below 0.14.1, "due to lack of authentication between the Inspector client and proxy" [29]. Its CNA scored it 9.4, Critical, under CVSS v4.0 [29].

3. Remote servers: origin, binding, authentication and state

Servers "MUST validate the `Origin` header on all incoming connections to prevent DNS rebinding attacks" and answer HTTP 403 to an invalid one [17]. Local servers "SHOULD bind only to localhost (127.0.0.1) rather than all network interfaces (0.0.0.0)", and servers "SHOULD implement proper authentication for all connections" [17]. OWASP adds a Host header check [8].

Scans show the gap. In June 2025, Backslash Security covered about half of the public servers then available and found "hundreds of cases" of servers bound to all interfaces, open to anyone on the same network, and "dozens of instances" that allowed arbitrary command execution [32]. In July 2025, Knostic found 1,862 MCP servers exposed to the internet, checked 119 by hand with read-only listing requests, and found that "All 119 servers granted access to internal tool listings without authentication" [33].

Authorization URLs. NVD's CVE-2025-6514 says mcp-remote "is exposed to OS command injection when connecting to untrusted MCP servers due to crafted input from the authorization_endpoint response URL", scored 9.6, Critical, under CVSS v3.1 by its CNA, JFrog [27]. JFrog reports versions 0.0.5 to 0.1.15 affected and 0.1.16 fixed, and advises connecting only to trusted servers over HTTPS [28]. Clients "MUST validate authorization URLs and reject dangerous schemes", allowing only `http://` and `https://`, and `http://` only for loopback [6].

Server-side request forgery. OAuth metadata discovery makes a client fetch URLs a malicious server controls, such as `http://169.254.169.254/`, so clients deployed to a server "MUST consider SSRF risks" [6].

State. Servers that need state across calls use "explicit, server-minted handles" passed as tool arguments [12]. They "MUST NOT treat possession of a state handle as authentication" and should bind handles to the authenticated user, as in `<user_id>:<handle>` [6]. For older session-based revisions, OWASP gives the same rule for session IDs [8].

4. Authorization and scopes

Audience. Clients "MUST implement Resource Indicators for OAuth 2.0 as defined in RFC 8707", naming the server's canonical URI [5, 22]. Servers "MUST validate that access tokens were issued specifically for them as the intended audience" and "MUST NOT accept or transit any other tokens" [5]. Tokens never go in a query string [5]. The security guide calls token passthrough "an anti-pattern", with risks that include security control circumvention and audit trail issues [6]. RFC 9700 says access tokens "SHOULD be audience-restricted to a specific resource server" [21].

Confused deputy. A proxy server that uses one static client ID with a third-party authorization server lets an attacker obtain authorization codes "without proper user consent", so "MCP proxy servers MUST implement per-client consent" [6].

Scopes. The guide's example failure is a token "carrying broad scopes (`files:*`, `db:*`, `admin:*`)" granted up front because the server listed every scope in `scopes_supported` and the client requested them all [6]. The fix is a minimal initial set, such as `mcp:tools-basic`, and step-up through `WWW-Authenticate` challenges when a privileged operation is first attempted, each elevation logged with a correlation ID [6]. Its common mistakes include wildcard scopes (`*`, `all`, `full-access`), bundled privileges, silent changes to what a scope means, and "Treating claimed scopes in token as sufficient without server-side authorization logic" [6]. A challenge with no scope sends clients back to "requesting all scopes listed in `scopes_supported`", so the published list should be short [6]. RFC 9700 limits token privileges to "the minimum required for the particular application or use case" [21]. OWASP adds "never share tokens across servers", narrow scopes such as `mail.readonly`, and "ephemeral, short-lived tokens over long-lived PATs" [8].

Practice. Astrix Security's October 2025 survey of 5,205 open-source server repositories found that about 88% require credentials, 53% rely on static API keys or personal access tokens, "which are long-lived and rarely rotated", 8.5% use OAuth, and 79% of API keys pass through environment variables [11]. The figures come from a model's reading of README files, and 26.4% of servers that mentioned credentials could not be assigned a credential type [11].

Tools the task needs. A server's tool list "MAY vary by the authorization presented on the request", for example returning only the tools a caller's scopes permit [16]. CoSAI asks for tools with "a single, clearly defined purpose" and says implementations "should not rely on the LLM to perform security-critical operations" [20]. GitHub's MCP server enables chosen tools through `--toolsets` and `--tools`, and "Read-only mode takes priority" [35].

Server logic. In June 2025, a work-management software vendor said a logic flaw in its MCP server had exposed customer data to other organizations, affecting roughly 1,000 customers [31]. The flaw was found on 4 June 2025, and the server was taken offline and returned on 17 June [31].

Elicitation. Servers "MUST NOT use form mode elicitation to request sensitive information such as passwords, API keys, access tokens, or payment credentials", and URL mode keeps such credentials out of "the LLM context, MCP client or any intermediate MCP servers" [18].

5. Tool descriptions: poisoning, line jumping, shadowing and rug pulls

Invariant Labs defined the attack in April 2025: "A Tool Poisoning Attack occurs when malicious instructions are embedded within MCP tool descriptions that are invisible to users but visible to AI models" [7]. Trail of Bits called it line jumping, because malicious servers "execute attacks before any tool is even invoked": the client adds the descriptions from `tools/list` to the model's context, and "tool descriptions themselves are prime vectors for prompt injection" [24]. The local server guide puts it plainly: "Tool names and descriptions are instructions your model reads" [4].

The surface is the whole definition. OWASP places poisoning in "tool descriptions, parameter schemas, or return values" [8]. A definition carries a name, title, description, icons, input and output schemas and annotations [16]. OWASP's agentic list names tool-descriptor injection in "a tool's metadata or MCP/agent-card" [19].

Shadowing. In Invariant's second experiment, a malicious server's bogus tool for adding numbers carried a description that redirected a trusted email server's messages [7]. OWASP calls this tool shadowing or cross-origin escalation [8].

Rug pulls. In Invariant's words, "a malicious server can change the tool description after the client has already approved it" [7]. The local server guide says a server can make that change quietly, so "a one-time consent dialog and package pinning do not cover this surface" [4]. CoSAI's threat MCP-T6 describes the missing control: without integrity verification, "malicious actors can modify tool definitions post-deployment, alter configuration files after approval, and inject malicious updates without detection" [20].

Stakes. Simon Willison's lethal trifecta is access to private data, exposure to untrusted content and a way to communicate externally [26]. He writes that MCP "encourages users to mix and match tools from different sources that can do different things" [26]. A poisoned description is untrusted content inside the agent's own tool list [4, 24].

6. Pinning tool definitions and asking again on change

"Prefer clients that pin tool definitions," the local server guide says. "Some clients record tool definitions when you first approve a server and ask again when they change. The protocol does not require this, so it is a property worth choosing a client for" [4]. Willison asked in April 2025 for clients to "show users initial tool descriptions and alert them if those descriptions ever change" [25].

What to hash. Invariant proposed "a hash or checksum to verify the integrity of the tool description before executing it" [7]. OWASP widens it: "Pin reviewed tool definitions using cryptographic hashes and require review when they change", and "Hash the complete tool definition (including `inputSchema`, `outputSchema`, and `annotations`), not only the description" [8]. The reason: "a flipped `readOnlyHint` or `destructiveHint` can change client approval behavior without any visible description change" [8]. OWASP's agentic list asks to "Pin prompts, tools, and configs by content hash and commit ID", with staged rollout and "auto-rollback on hash drift or behavioral change" [19].

When to compare. A tool list "MAY change over time", and servers that declared `listChanged` "SHOULD send a notification" when it does [16]. A server can also change definitions quietly [4], so OWASP asks clients to "Monitor for changes to tool descriptions post-installation" and to "Re-prompt for consent when tool definitions change" [8]. A client that compares every `tools/list` response with the approved hashes catches changes the server did not announce. The list may also vary with the authorization on the request [16], so a list fetched with other credentials can differ with no change on the server.

Annotations. The project's March 2026 post says "An untrusted server can lie. A server can claim readOnlyHint: true and delete your files anyway", and it lists "Skip the confirmation dialog" as the client decision that `readOnlyHint: true` drives [15]. Clients "MUST consider tool annotations to be untrusted unless they come from trusted servers" [16].

Collisions. Tool name uniqueness "is scoped to a single server", and clients that combine servers "SHOULD implement a disambiguation strategy such as prefixing tool names with a server identifier", without relying on the server name in `serverInfo` [16]. MCP Security Bench counts name collision among its attack types [34].

Limits of a pin. Hashing "detects metadata changes, not changes to server code or behavior behind an unchanged definition" [8]. The local server guide says no check can prove in advance how a server behaves, and asks users to treat a third-party local server "as untrusted code with exactly the privileges you hand it" [4]. Pinning sits beside isolation, narrow credentials and a small roster of servers [4, 8].

7. Tool results and sampling

Results. OWASP says to "Treat every tool response as untrusted data, including responses from approved servers", and to validate later tool calls "in trusted application code, independently of the model's interpretation" [8]. The specification asks clients to "Validate tool results before passing to LLM" [16]. In Invariant's May 2025 case, a malicious GitHub issue could "hijack a user's agent" through the GitHub MCP server and coerce it "into leaking data from private repositories" [23]. The researchers called it "a fundamental architectural issue that must be addressed at the agent system level" and showed a policy that limits an agent "to working with only one repository per session" [23].

Sampling. "With sampling, MCP servers can proactively request LLM completions by sending sampling requests back to the client" [13]. Unit 42's December 2025 proofs of concept used it to drain a client's compute quota, plant persistent instructions in a conversation and make a hidden file write [13]. Its defenses include explicit approval for tool execution, limits on what servers can request, "context isolation that

prevents access to conversation history" and rate limits [13]. Revision 2026-07-28 deprecates Sampling: new implementations should not add it, and the suggested migration is to "integrate directly with LLM provider APIs instead of Sampling" [12].

8. Approvals and logs

Approval. The specification says "there SHOULD always be a human in the loop with the ability to deny tool invocations" [16]. Clients should confirm sensitive operations and "Show tool inputs to the user before calling the server, to avoid malicious or accidental data exfiltration" [16]. OWASP adds "Display full tool call parameters to the user" and "Never auto-approve tool calls, especially in multi-server setups" [8]. In Invariant's first experiment, the client's confirmation dialog hid the full tool input [7]. Servers must validate inputs, apply access controls, rate limit calls and sanitize outputs [16].

Logs. Clients should "Log tool usage for audit purposes" [16]. OWASP asks to "Log all MCP tool invocations with full parameters, user context, and timestamps", to "Redact secrets and PII from logs" and to watch for credentials from one server appearing in calls to another [8]. The security guide logs each scope elevation with a correlation ID [6]. Revision 2026-07-28 adds OpenTelemetry trace context conventions [12].

Testing. The authors of the 1,899-server study advocate "enabling automated security scanning within MCP registries" [10]. A test of a server before an agent connects looks for the behaviors in this report: poisoned or changed definitions, instructions in results and access beyond the task [4, 8].

9. Measured rates

MCPTox, first posted in August 2025 and revised in September 2026, built 1,348 malicious test cases on "45 live, real-world MCP servers and 353 authentic tools" and ran them against 20 agent settings [9]. The average attack success rate was 36.5%, the most affected setting reached 72.8% and two more exceeded 70%, and the highest refusal rate was under 3% [9]. The authors found that "more capable models are often more susceptible" [9].

MCP Security Bench (MSB), first posted in October 2025 and revised in March 2026, ran 2,000 attacks of 12 types against ten agents over 405 tools [34]. The overall average attack success rate was 40.35% [34]. Out-of-scope parameter requests averaged 76.5%, user-impersonating responses 45.69% and prompt injection 20.21% [34]. The authors report that "Models with stronger performance are more vulnerable" [34].

Study	Date	Measured	Result
MCPTox [9]	Aug 2025, rev. Sep 2026	1,348 poisoning cases, 20 settings	36.5% average success, refusals under 3%
MSB [34]	Oct 2025, rev. Mar 2026	2,000 attacks, ten agents	40.35% average success

Study	Date	Measured	Result
Hasan and others [10]	Jun 2025, rev. Apr 2026	1,899 open-source servers	5.5% tool poisoning, 7.2% general vulnerabilities
Astrix [11]	Oct 2025	5,205 server repositories	53% static keys or tokens, 8.5% OAuth
Knostic [33]	Jul 2025	1,862 exposed servers	119 of 119 checked listed tools without authentication

10. The checklist with its sources

"Spec" is revision 2026-07-28 and "Guide" one of the project's security guides, each with its MUST, SHOULD or MAY, or a deprecation. "Guidance" is advice without such a keyword, from the project's guides, OWASP, CoSAI, an RFC or a study.

Area	Check	Strength	Sources
Servers	Inventory every server, remove unused ones	Guidance	[4, 20]
Servers	Allow only reviewed servers with an owner	Guidance	[4, 20]
Servers	Check namespace and exact package name	Guidance	[2, 3, 8]
Servers	Pin versions, images by digest	Guidance	[4, 20]
Servers	Keep a kill switch per tool and connection	Guidance	[19]
Local	Show the exact command before one-click install	Guide MUST	[6]
Local	Sandbox with minimal privileges, no network unless needed	Guide SHOULD	[4, 6]
Local	One short-lived, narrow credential per server	Guidance	[4, 8]
Remote	Validate `Origin`, bind to 127.0.0.1, authenticate	Spec MUST, SHOULD	[17]
Remote	Reject dangerous authorization URL schemes	Guide MUST	[6]

Area	Check	Strength	Sources
Remote	Guard metadata fetches against SSRF	Guide MUST	[6]
Remote	Never authenticate by state handle or session ID	Guide MUST	[6, 8]
Scopes	Resource indicators, audience-checked tokens	Spec MUST	[5]
Scopes	No token passthrough	Spec MUST	[5, 6]
Scopes	Per-client consent in proxy servers	Guide MUST	[6]
Scopes	Minimal scopes, step-up, no wildcards	Guidance	[6, 21]
Scopes	Expose only the tools the task needs	Spec MAY, guidance	[16, 35]
Scopes	Sensitive credentials only by URL mode	Spec MUST	[18]
Tools	Hash the full definition, compare every list	Guidance	[8, 19]
Tools	Re-prompt on any change	Guidance	[8, 25]
Tools	Treat annotations as untrusted	Spec MUST	[16]
Tools	Prefix names by server	Spec SHOULD	[16]
Results	Treat results as untrusted, validate next calls	Spec SHOULD, guidance	[8, 16]
Approvals	Person can deny, full inputs, no auto-approve	Spec SHOULD, guidance	[8, 16]
Sampling	None in new implementations	Spec deprecation	[12]
Logs	Every call, parameters, user, time, redacted	Spec SHOULD, guidance	[8, 16]
Testing	Scan and test before an agent connects	Guidance	[10]

Implications

For teams that connect agents. Start with the inventory and the allow-list, because every later check runs per server [4, 20]. Choose clients that pin definitions and ask again on change, since the protocol does not require it [4]. Run local servers in containers without network unless needed [4]. Check each agent's tools for the lethal trifecta [26].

For teams that write servers. Validate `Origin`, accept only tokens issued for the server and never pass them through [5, 17]. Publish a short `scopes_supported` list and elevate through challenges [6]. Keep authorization in the server's own code [6, 20]. Add no Sampling to new implementations [12].

For reviewers. Treat a registry listing as identity [2, 3]. Read the full definition, and treat a matching hash as proof of unchanged metadata only [8]. In MCPTox and MSB, agents followed poisoned or manipulated tools in more than a third of cases on average [9, 34].

Testing. Rillor's agentic security work tests the MCP servers an agent connects to for tool poisoning, descriptions changed after approval, instructions in tool results and access beyond what the task needs. It tests only systems the client owns or is authorized to test, within a written scope.

Limits of this analysis

- This explainer reads published requirements and guidance as of 7 November 2026, against revision 2026-07-28. Older revisions differ on sessions and sampling [6, 12].
- The project's guides, registry pages and the OWASP cheat sheet carry no page date [2, 3, 4, 6, 8]. CoSAI's paper gives its status as "Draft" [20], and the registry calls itself "currently in preview" [2].
- The protocol does not require pinning [4]. No source reviewed names the clients that pin definitions, so this report names none.
- Benchmark rates come from their authors' test settings and are not production rates [9, 34].
- Scan figures date from 2025 and use different methods: README reading by a model [11], about half of public servers [32], and 119 of 1,862 exposed servers [33].
- CoSAI describes "nearly forty threats", and its three tier headings list 7, 8 and 19, which sum to 34 [20].

Sources

1. Model Context Protocol specification, revision 2026-07-28: Overview. Model Context Protocol project, 28 July 2026, viewed 7 November 2026. https://modelcontextprotocol.io/specification/2026-07-28

2. How to Authenticate When Publishing to the Official MCP Registry. Model Context Protocol project, undated, viewed 7 November 2026. https://modelcontextprotocol.io/registry/authentication

3. The MCP Registry Moderation Policy. Model Context Protocol project, undated, viewed 7 November 2026. https://modelcontextprotocol.io/registry/moderation-policy

4. Local Server Security (documentation for revision 2026-07-28). Model Context Protocol project, undated, viewed 7 November 2026. https://modelcontextprotocol.io/docs/2026-07-28/tutorials/security/local-server-security

5. Model Context Protocol specification, revision 2026-07-28: Authorization. Model Context Protocol project, 28 July 2026, viewed 7 November 2026. https://modelcontextprotocol.io/specification/2026-07-28/basic/authorization

6. Security Best Practices (documentation for revision 2026-07-28). Model Context Protocol project, undated, viewed 7 November 2026. https://modelcontextprotocol.io/docs/2026-07-28/tutorials/security/security_best_practices

7. MCP Security Notification: Tool Poisoning Attacks. Invariant Labs, 1 April 2025, viewed 7 November 2026. https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks

8. MCP (Model Context Protocol) Security Cheat Sheet. OWASP Cheat Sheet Series, undated, viewed 7 November 2026. https://cheatsheetseries.owasp.org/cheatsheets/MCP_Security_Cheat_Sheet.html

9. MCPTox: A Benchmark for Tool Poisoning Attack on Real-World MCP Servers. Zhiqiang Wang, Yichao Gao, Yanting Wang and others, AAAI 2026, arXiv 2508.14925, version 2, 29 September 2026 (first version 19 August 2025), viewed 7 November 2026. https://arxiv.org/abs/2508.14925

10. Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers. Mohammed Mehedi Hasan, Hao Li, Emad Fallahzadeh and others, arXiv 2506.13538, version 5, 13 April 2026 (first version 16 June 2025), viewed 7 November 2026. https://arxiv.org/abs/2506.13538

11. State of MCP Server Security 2025: 5,200 Servers, Credential Risks, and an Open-Source Fix. Tal Skverer, Astrix Security, 15 October 2025, viewed 7 November 2026. https://astrix.security/learn/blog/state-of-mcp-server-security-2025/

12. Model Context Protocol specification, revision 2026-07-28: Key Changes. Model Context Protocol project, 28 July 2026, viewed 7 November 2026. https://modelcontextprotocol.io/specification/2026-07-28/changelog

13. New Prompt Injection Attack Vectors Through MCP Sampling. Yongzhe Huang, Akshata Rao, Changjiang Li, Yang Ji and Wenjun Hu, Unit 42, Palo Alto Networks, 5 December 2025, viewed 7 November 2026. https://unit42.paloaltonetworks.com/model-context-protocol-attack-vectors/

14. The 2026-07-28 Specification. David Soria Parra and Den Delimarsky, Model Context Protocol Blog, 28 July 2026, viewed 7 November 2026. https://blog.modelcontextprotocol.io/posts/2026-07-28/

15. Tool Annotations as Risk Vocabulary: What Hints Can and Can't Do. Ola Hungerford, Sam Morrow and Luca Chang, Model Context Protocol Blog, 16 March 2026, viewed 7 November 2026. https://blog.modelcontextprotocol.io/posts/2026-03-16-tool-annotations/

16. Model Context Protocol specification, revision 2026-07-28: Tools. Model Context Protocol project, 28 July 2026, viewed 7 November 2026. https://modelcontextprotocol.io/specification/2026-07-28/server/tools

17. Model Context Protocol specification, revision 2026-07-28: Transports, Streamable HTTP. Model Context Protocol project, 28 July 2026, viewed 7 November 2026. https://modelcontextprotocol.io/specification/2026-07-28/basic/transports/streamable-http

18. Model Context Protocol specification, revision 2026-07-28: Elicitation. Model Context Protocol project, 28 July 2026, viewed 7 November 2026. https://modelcontextprotocol.io/specification/2026-07-28/client/elicitation

19. OWASP Top 10 for Agentic Applications for 2026. OWASP Gen AI Security Project, 9 December 2025, viewed 7 November 2026. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

20. Model Context Protocol (MCP) Security. Coalition for Secure AI (CoSAI), an OASIS Open Project, Workstream 4, 8 January 2026, viewed 7 November 2026. https://www.coalitionforsecureai.org/wp-content/uploads/2026/03/model-context-protocol-security-1.pdf

21. RFC 9700: Best Current Practice for OAuth 2.0 Security. T. Lodderstedt, J. Bradley, A. Labunets and D. Fett, IETF, January 2025, viewed 7 November 2026. https://www.rfc-editor.org/rfc/rfc9700

22. RFC 8707: Resource Indicators for OAuth 2.0. B. Campbell, J. Bradley and H. Tschofenig, IETF, February 2020, viewed 7 November 2026. https://www.rfc-editor.org/rfc/rfc8707

23. GitHub MCP Exploited: Accessing private repositories via MCP. Invariant Labs, 26 May 2025, viewed 7 November 2026. https://invariantlabs.ai/blog/mcp-github-vulnerability

24. Jumping the line: How MCP servers can attack you before you ever use them. The Trail of Bits Blog, 21 April 2025, viewed 7 November 2026. https://blog.trailofbits.com/2025/04/21/jumping-the-line-how-mcp-servers-can-attack-you-before-you-ever-use-them/

25. Model Context Protocol has prompt injection security problems. Simon Willison, simonwillison.net, 9 April 2025, viewed 7 November 2026. https://simonwillison.net/2025/Apr/9/mcp-prompt-injection/

26. The lethal trifecta for AI agents: private data, untrusted content, and external communication. Simon Willison, simonwillison.net, 16 June 2025, viewed 7 November 2026. https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/

27. CVE-2025-6514 Detail. National Vulnerability Database, National Institute of Standards and Technology, published 9 July 2025, viewed 7 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-6514

28. Critical RCE Vulnerability in mcp-remote: CVE-2025-6514 Threatens LLM Clients. Or Peles, JFrog, 9 July 2025, viewed 7 November 2026. https://jfrog.com/blog/2025-6514-critical-mcp-remote-rce-vulnerability/

29. CVE-2025-49596 Detail. National Vulnerability Database, National Institute of Standards and Technology, published 13 June 2025, viewed 7 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-49596

30. MAL-2025-47604: Malicious code in postmark-mcp (npm). OSV (Open Source Vulnerabilities), 26 September 2025, viewed 7 November 2026. https://osv.dev/vulnerability/MAL-2025-47604

31. Asana warns MCP AI feature exposed customer data to other orgs. Bill Toulas, BleepingComputer, 18 June 2025, viewed 7 November 2026. https://www.bleepingcomputer.com/news/security/asana-warns-mcp-ai-feature-exposed-customer-data-to-other-orgs/

32. Threat Research: Hundreds of MCP Servers Vulnerable to Abuse. Backslash Security, 25 June 2025, viewed 7 November 2026. https://www.backslash.security/blog/hundreds-of-mcp-servers-vulnerable-to-abuse

33. Exposing the Unseen: Mapping MCP Servers Across the Internet. Imri Goldberg, Knostic, 17 July 2025, viewed 7 November 2026. https://www.knostic.ai/blog/mapping-mcp-servers-study

34. MCP Security Bench (MSB): Benchmarking Attacks Against Model Context Protocol in LLM Agents. Dongsen Zhang, Zekun Li, Xu Luo, Xuannan Liu, Peipei Li and Wenjun Xu, ICLR 2026, arXiv 2510.15994, version 2, 24 March 2026 (first version 14 October 2025), viewed 7 November 2026. https://arxiv.org/abs/2510.15994

35. GitHub MCP Server (README). GitHub, undated, viewed 7 November 2026. https://github.com/github/github-mcp-server

About this report

Published by Rillor on 8 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. MCP security checklist: servers, scopes and pinned tool descriptions. 8 November 2026. https://rillor.com/insights/mcp-security-checklist

Notices

Superintelligence (SI) outputs can be wrong. Important decisions should include human review.

Work with Rillor

Send us the agents, the tools and data they reach, where they run and what you want tested. Contact us at rillor.com/contact?type=security or contact@rillor.com. More on agentic security: rillor.com/si/agentic-security