The State of MCP

What the protocol unlocked, how far it spread, and the trust layer it still lacks

A Major Labs field scan of 2,348 Model Context Protocol servers, read against the public security and governance record. May 2026.

Major Labs is editorially independent of Major Matters and operationally independent of Mastercard.

How to read this

Eighteen months ago the Model Context Protocol did not exist. Today it is the way most AI agents reach the tools, data, and systems they act on, and the three other large model labs have adopted it. This is a report on what that protocol made possible, how big it got, and the one layer it still lacks: a way to tell a server you can trust from one you cannot.

Two kinds of data sit in here. The first is our own: a census of 2,348 public MCP server repositories, scanned by the Major Labs bench agent we call Sentinel. Those figures are firsthand, and the method is at the end. The second is the public record, the peer-reviewed papers, vendor scans, CVE filings, and official announcements that frame what we found. Every external quantitative claim carries a source-strength tag so the piece survives review:

[A] primary, peer-reviewed, or official: specs, registries, NVD records, foundation releases, accepted papers.

[B] vendor research with a disclosed method: a named scan with a stated approach, usable with attribution.

[C] secondary aggregation or estimate: trade reports, directory self-counts, round numbers. Directional only, never a headline.

The rule we hold ourselves to: a soft number is attributed in line, never printed as settled fact.

The one-paragraph version

MCP solved a real problem, and it solved it well. Before it, every connection between a model and a tool was bespoke, so the industry was building the same integrations over and over. MCP turned that into write-once, and adoption followed at a pace that has few precedents: from zero to more than 10,000 active public servers and 97 million-plus monthly SDK downloads in about a year and a half [A]. But the layer that would let an enterprise trust a given server never shipped alongside the layer that made servers easy to build. Our scan of the public surface found that about a third of repositories have no documentation, nearly half have no dependency manifest, four in ten advertise a remote network surface, and the honest count of servers an engineer could actually evaluate is close to 1,200, not the tens of thousands the directories list. The security record points the same way: the protocol's worst weaknesses are architectural, not bugs, and the most rigorous study to date got a 72.8 percent attack success rate against a leading model by hiding instructions in a tool's description [A]. The protocol has won. The trust layer underneath it is the next thing someone has to build.

Part 1 — What MCP unlocked

Start with what was hard before, because that is what explains the speed.

Until late 2024, connecting an AI model to an outside system meant writing a custom integration for that exact pairing. A model from one lab, a tool from one vendor, glued together by hand. Add a second model and you wrote it again. Add a second tool and you wrote it again. The industry was solving an M-by-N problem, M models times N tools, and the cost grew with the product of the two.

Anthropic introduced the Model Context Protocol on 25 November 2024 to collapse that product into a sum [A]. One open protocol sits between models and tools, so a tool builder writes a single MCP server and every compatible client can reach it, and a client adds MCP support once and reaches every server. M times N becomes M plus N. Anthropic's own metaphor for it, a universal port, has stuck: MCP is the USB-C of AI integration [A].

That is the whole capability shift, and a series of additions since has widened what the protocol can carry. Each one let an agent do something it could not do cleanly before:

Reach any tool from any client. Write one server, and it works in Claude, ChatGPT, Gemini, Copilot, Cursor, and VS Code rather than one model at a time [A].

Chain tools across applications in a single run. An agent can read an issue tracker, open a pull request, and post to a chat channel through one uniform interface, instead of three separate integrations [A].

Find servers programmatically. The official registry, launched in preview in September 2025, lets clients and agents discover servers rather than hard-code endpoints [A].

Run long jobs and ask mid-task. The November 2025 specification added asynchronous tasks for work that takes minutes or hours, and an elicitation mechanism that can open a secure browser-side prompt for an OAuth login or a payment, so credentials never pass through the chat transcript [A].

Push reasoning to the server. The same revision lets a server run its own model loop using the client's tokens, moving complexity off the client [A].

Return interactive surfaces, not just text. MCP Apps, the first official extension in January 2026, lets a tool send back a dashboard, a form, or a chart rendered in a sandboxed frame inside the chat, with every UI action routed through the same consent path as a tool call. Figma, Canva, Slack, Asana, Box, and others shipped on day one [A].

Reach inside a private network. In May 2026 both Anthropic and OpenAI shipped MCP tunnels in research preview: an agent can reach a server running inside a corporate network over an outbound-only connection, with no inbound firewall port opened and nothing exposed to the public internet [A].

Read that list as a single arc. In eighteen months MCP went from "a model can call one tool" to "an agent can discover a tool, run it for an hour, ask you for a credential safely, hand you back a live interface, and do all of it against a system locked inside your own network." That is the thing that could not be done before. It is also why the population of servers grew the way it did.

The protocol made it cheap to give an agent reach. It did not make it cheap to know whether the thing on the other end of that reach can be trusted.

Part 2 — How big it got

Here the story gets harder to tell honestly, because nobody can count MCP servers cleanly, and saying so is the first finding.

The public directories do not agree, and they are not measuring the same thing. As of May 2026, **mcp.so** lists close to 20,000 servers, **Glama** more than 21,000, **PulseMCP** around 16,000, and **Smithery** about 7,300 [C]. A meta-index, **MCPfinder**, reported a deduplicated union of 27,432 on 25 May 2026 [C]. A separate tracked-subset estimate put roughly 9,400 distinct servers in mid-April, up from about 6,800 at the end of 2025, a 38 percent rise in four months [C]. Every one of those is a self-report with no audited method, inflated by forks, thin wrappers, and the same server re-published under several names.

The one number with a primary source behind it is Anthropic's own: more than 10,000 active public servers [A]. That is the figure to anchor on. Everything above it is directory accounting.

Adoption is easier to measure through usage than through inventory. The Python and TypeScript SDKs together passed 97 million monthly downloads by December 2025, the figure Anthropic cited when it donated the protocol [A], and they have climbed since: the Python package alone was clearing roughly 250 million downloads a month by late spring 2026 [B]. Downloads overcount, because continuous-integration systems re-pull packages on every build, but the order of magnitude is consistent across counters and the trajectory is not in doubt.

So counting was never going to settle anything. We went past it. Rather than ask how many servers are listed, we asked how many a careful engineer could actually adopt, and to answer that you have to look at each one.

Our census. Sentinel discovered 2,413 distinct MCP-related repositories across five GitHub queries, found 2,348 active after removing archives and forks, and deep-analyzed all 2,348 for documentation, manifests, transport, license, and maintenance. Together they carry 1.8 million GitHub stars. Three cuts of that data describe the shape of the ecosystem better than any directory total:

Velocity. From a handful of repositories a quarter before the launch to nearly 2,400 today. Roughly 57 percent were created in the trailing twelve months, and the trailing quarter ran at about four new servers a day. This is an ecosystem still in its first growth phase, not a settled one.

Concentration. Attention is winner-take-most. The 10 most-starred repositories hold 39 percent of all 1.8 million stars, and the top 50 hold 67 percent. The median server has 74 stars, 59 percent sit under 100, and 160 have none at all.

Ownership. Of 2,000 distinct owners, 1,816, that is nine in ten, own exactly one repository. The official modelcontextprotocol organization accounts for 37 repositories and 174,000 stars; nearly everyone else is a single maintainer with a single project.

Put those together and the population is two ecosystems wearing one name. At the top, a healthy, vendor-backed protocol with real engineering behind it. Below it, a long tail of individual experiments. Nothing in MCP, or in the way agents discover servers, tells the two apart.

Part 3 — What we found scanning it

The deep pass turned the shape into specifics. Four findings, in order of how much they should worry an enterprise.

Finding 1: the protocol is winning, and the win is real. The top of the distribution is genuinely healthy. **GitHub**, **AWS**, **Microsoft**, **Sentry**, **Cloudflare**, **Stripe**, and **Atlassian** all ship maintained servers, and the official organization ships the SDKs, the inspector, and the registry. TypeScript at 761 repositories and Python at 718 are 63 percent of the active set, which tells you the builders are application developers, not systems engineers. Half of all active repositories, 1,171 of them, were pushed in the last 30 days. The center of gravity is busy and alive. The problem is everything underneath it.

Finding 2: half of it is not deployable software. Of 2,348 analyzed repositories, 823, that is 35 percent, have no README at all: no description of what the server does, what it connects to, what scopes it asks for, or who maintains it. Separately, 1,129, that is 48 percent, ship no dependency manifest, no package.json, pyproject, requirements, cargo, or go.mod. Two independent signals, missing documentation and missing manifest, converge on the same honest count: roughly 1,200 servers a reasonable engineer could evaluate, not the tens of thousands the directories advertise. An undocumented, un-manifested repository is not a candidate. It is a liability with a star count.

Finding 3: four in ten advertise a remote surface, and some have been abandoned there. Transport is the security-relevant split. A server that runs locally over stdio sits inside a trust boundary you control. A server reachable over HTTP is where remote

attack surface lives: authentication, input handling, server-side request forgery, secrets in transit. From documentation pattern matching, 953 repositories, that is 41 percent, advertise an HTTP-capable surface. Now intersect that with maintenance: 210 of those HTTP-capable servers have not been touched in more than six months. Remotely reachable, advertised, and forgotten. In any other supply chain, an abandoned network-exposed endpoint is the thing you learn about after it is exploited.

Finding 4: the governance black holes. Licensing is where intent meets legal reality. 341 active repositories carry no license and another 181 carry an unrecognized one, so more than one in five cannot be adopted without a legal review regardless of code quality. Worse, 102 have neither a license nor a README at the same time: you cannot legally use them and you cannot tell what they do, yet they appear in topic searches, and an agent told to “find an MCP server for X” can route straight into one. Selection has to be gated, not searched.

Part 4 — The security reality

The supply-chain findings describe a population. The security research describes what happens when an agent trusts the wrong member of it, and the most important point in this section is structural: MCP’s worst weaknesses are not bugs that a patch removes. They are consequences of the protocol’s trust model.

The model treats a tool’s description as authoritative. When a server registers a tool, the text describing that tool enters the agent’s context as trusted content, before any code runs. Hide an instruction in that description and you have an attack with no execution and no runtime trace.

That attack class has a name and now a peer-reviewed measurement. **MCPTox**, accepted to AAAI 2026 by researchers at the University of Science and Technology of China and Beihang University, built 1,312 malicious test cases on 45 live MCP servers and 353 real tools, and ran them against 20 leading agents [A]. The headline results:

The most susceptible model, **o1-mini**, was successfully attacked 72.8 percent of the time, and several others, including DeepSeek-R1, crossed 60 percent [A].

More capable models were often more vulnerable, not less, because stronger instruction-following makes them more compliant with malicious metadata [A].

Refusal was almost absent. The best-defended model tested, Claude-3.7-Sonnet, still refused less than 3 percent of the time, which means standard safety alignment does not address tool poisoning [A].

The attack class is not theoretical, and it predates the paper. **Invariant Labs** demonstrated tool poisoning and a “rug pull,” a server that turns malicious after it is trusted, in April 2025 [B]. In September 2025 the first malicious MCP server found in the wild, a backdoored build of **postmark-mcp**, silently copied every outgoing email to an attacker’s address before it was pulled [B].

A separate static analysis sizes the latent risk. **Endor Labs** scanned 2,614 MCP implementations in January 2026 and found that 82 percent use file operations prone to path traversal, 67 percent use APIs related to code injection, and 34 percent use APIs susceptible to command injection [B]. These are capability findings, the code can do the dangerous thing, not confirmed exploitable holes, and they come from one vendor’s static method, so the honest reading is “the dangerous patterns are everywhere,” not “82 percent are vulnerable.”

The disclosed-vulnerability record fills in the rest. A community tracker, the Vulnerable MCP Project, lists more than 50 known MCP vulnerabilities, 13 of them critical, and notes that figure is a floor because first-party servers get fixed without ever receiving a CVE [B/C]. The confirmed filings include two genuinely critical remote-code-execution flaws, CVE-2025-49596 in the MCP Inspector at CVSS 9.4 and CVE-2025-6514 in mcp-remote at 9.6, the latter affecting a package with more than 437,000 downloads [A]. A chain of three flaws in Anthropic’s own mcp-server-git, disclosed in January 2026, reached remote code execution when combined with a filesystem server [A]. Two other widely covered issues, MCPoison in Cursor and the EscapeRoute pair in Anthropic’s filesystem server, were serious but high-severity rather than critical, at CVSS 7.2 and 8.4 and 7.3 [A].

The clearest illustration of the architectural point arrived in April 2026. **Ox Security** reported that the standard input-output transport in the official MCP SDKs allows command execution by design, across the Python, TypeScript, Java, and Rust implementations, in a supply chain with more than 150 million downloads [C]. Anthropic confirmed the behavior is intended and declined to change the protocol. The often-quoted figure of “up to 200,000 vulnerable instances” from that report is an explicit upper-bound estimate with an unclear denominator, so treat it as the softest number in this brief and quote it as an estimate or not at all [C]. The durable point is not the count. It is that a root-cause design property, not a coding mistake, sits under a large share of the disclosed issues.

For a shared vocabulary, the field is converging on the **OWASP MCP Top 10**, currently in beta [A]. Its categories run from token mismanagement and scope-creep privilege escalation through tool poisoning, supply-chain tampering, command injection, intent-flow subversion, weak authentication, missing audit, shadow servers, and context over-sharing. Several of those are not implementation bugs. They are the trust model showing through.

Part 5 — The governance response

The encouraging half of the story is that the people who built MCP know the trust layer is missing, and the last twelve months are mostly the record of them building it.

A discovery layer. The official MCP Registry launched in preview on 8 September 2025 as a canonical, machine-readable source of truth [A]. It is deliberately the smallest of the directories, around 2,000 curated entries against the directories' tens of thousands, because it is meant to be a verified layer rather than an aggregator [C].

A harder specification. The 25 November 2025 revision, landing on the protocol's first anniversary, was its largest, and much of it is security work: OAuth 2.1 enforcement, explicit client security requirements, and structured outputs alongside the new task and elicitation features [A].

Neutral governance. On 9 December 2025 Anthropic donated MCP to the **Agentic AI Foundation**, a fund under the **Linux Foundation**, co-founded with **Block** and **OpenAI**, with **Google**, **Microsoft**, **AWS**, **Cloudflare**, and **Bloomberg** as members [A]. The protocol that began as one company's project is now vendor-neutral and community-governed, while keeping technical autonomy.

Client-side hardening. The model and tooling vendors are tightening the boundary where tool poisoning lands. Anthropic's Claude Code, in the Opus 4.8 release of 28 May 2026, now marks unapproved servers as pending rather than auto-connecting them, and improved its detection of bulk data exfiltration [A]. This is exactly the trust boundary MCPTox attacks.

Public-sector standards. The US National Institute of Standards and Technology launched an AI Agent Standards Initiative on 17 February 2026, naming MCP among the protocols it will profile, with an interoperability profile expected in the fourth quarter of 2026 [A].

The common thread across every vendor’s guidance is defense in depth: approval gating, tool allowlisting, least-privilege scope, runtime inspection of tool arguments, identity binding, and human checkpoints. No single control is treated as sufficient, which is the correct posture for a problem that is partly architectural.

Part 6 — The forward read

Every prior package ecosystem ran the same sequence: growth first, incident second, registry third. npm, PyPI, and the container registries all learned governance after an incident taught them they needed it. MCP is mid-stride between the first stage and the second. The growth is undeniable, the incidents have started, and the registry-with-a-quality-bar is still being assembled.

What that means for the next six to twelve months, read off the official roadmap and the standards work rather than the hype:

Enterprise readiness becomes the gating issue. The March 2026 roadmap names it the top priority, with four concrete gaps: audit trails, identity and single sign-on to replace static secrets, gateway and proxy patterns, and configuration portability [A]. Most of this is planned to ship as extensions rather than core-spec growth.

The registry has to earn trust, not just list servers. A curated, verified, security-audited directory with usage signals is the intended answer to today’s fragmentation, targeted around the fourth quarter of 2026 [B]. Whether it arrives on time is the single most important variable in this whole report.

Identity is the next battleground. OAuth 2.1 is in the spec; brokered, scoped, short-lived tokens are where it goes next, and tunnels are becoming the standard pattern for private access.

The category-defining security finding is still ahead. Our scan stopped at the public surface and did not probe live servers. The next pass, active testing of the 953 HTTP-capable endpoints, is where the first systematic, measured vulnerability rate for deployed MCP servers will come from. We expect it to be uncomfortable.

The structural insight underneath all of it: for eighteen months, capability outran governance. Every spec revision added power, asynchronous tasks, server-side loops, interactive apps, private tunnels, faster than the trust layer added safety. Enterprise readiness did not become the headline priority until sixteen months in. That gap,

between what an agent can now reach and what anyone can vouch for, is the opportunity and the risk in one sentence.

Major Labs is building toward the third stage. Sentinel, the scanner behind this report, is the instrument for an ongoing MCP quality and provenance index: a way to tell, for any server, whether it is maintained, scoped, documented, and attributable, before an agent is allowed to call it. The point of publishing the scan is not to raise an alarm. It is to put a number on a gap that is otherwise invisible, and to be early and specific about it while there is still time to build the control rather than the incident report.

What this means for any regulated enterprise

Strip the findings to four decisions for anyone about to let agents act on its behalf:

Agent-reachable tools are a new third-party-risk category. A server in an agent's path is a dependency with the same supply-chain exposure as any library, and almost none of the registry, scanning, or provenance tooling that npm and PyPI eventually grew. Treat it like one.

"Find an MCP server for X" is an unsafe default. With about a third undocumented, nearly half un-manifested, and one in five unlicensed, automated or semi-automated tool selection can route an agent into an abandoned or unattributable server. Selection has to be gated, not searched.

Govern remote transport first. The 953 HTTP-capable servers, and especially the 210 abandoned ones, are where the risk concentrates. Apply the controls you already apply to any externally reachable endpoint: identity, scope, logging, and a maintenance check before anything enters the path.

The missing control is a registry with a quality bar. Not a marketplace, a gate: maintained or not, scoped or not, documented or not, attributable or not. Whoever runs that gate for the enterprise owns the trust layer of the agentic supply chain.

Method and limits

Source. Public GitHub, five queries, deduplicated by canonical URL. Discovery completed 29 May 2026.

Sample. 2,413 discovered, 2,348 active after removing archives and forks, 2,348 deep-analyzed.

Transport signal is inferred from README and manifest pattern matching, not from running the servers. A repository with no README cannot be assigned a transport, which is itself a finding. “HTTP-capable” means a server advertises a network surface; it is not a confirmed vulnerability.

Functional count. The roughly 1,200 figure is an inference: analyzed repositories minus those with no documentation and no dependency manifest, two independent signals that agree.

Not in scope for this pass. Active probing, authentication testing, or server-side request forgery detection against live servers. That is the next pass, and where a measured vulnerability rate will come from.

External claims carry source-strength tags; the load-bearing security and adoption figures are confirmed against primary sources, and the soft estimates are marked as such.

Bias note. GitHub is the public surface. Private and vendor-hosted servers are not counted here and may be better governed. The public tail is the part an agent can wander into.

Sources

[A] Primary, peer-reviewed, official. Anthropic, Introducing the Model Context Protocol (25 Nov 2024) and the MCP tunnels, Opus 4.8, and Agentic AI Foundation announcements; MCP specification 2025-11-25 and the MCP blog posts for the registry preview (2025-09-08), MCP Apps (2026-01-26), and the 2026 roadmap; MCPTox, arXiv:2508.14925, AAAI 2026; NVD records for CVE-2025-49596, CVE-2025-6514, CVE-2025-54136, CVE-2025-53109, CVE-2025-53110, and CVE-2025-68143/68144/68145; OWASP MCP Top 10 (beta); NIST AI Agent Standards Initiative (17 Feb 2026).

[B] Vendor research, method disclosed. Endor Labs, static analysis of 2,614 MCP implementations (23 Jan 2026); Koi Security, postmark-mcp backdoor; Invariant Labs, tool-poisoning and rug-pull disclosures; npm and PyPI download counters; WorkOS spec and roadmap analysis; the Vulnerable MCP Project tracker.

[C] Secondary aggregation or estimate, directional only. Ox Security STDIO estimate (April 2026); directory self-reports from mcp.so, Glama, PulseMCP, Smithery, and MCPfinder; the tracked-subset server-count estimate.

The Major Labs scan figures are firsthand and carry no external tag; the method above is the disclosure. All [B] and [C] figures are attributed in line and should never be restated as settled fact.