

# Model Context Protocol: A View from the Trenches

What Enterprise MCP Deployments  
Actually Look Like

# Executive Summary

Clutch Security analyzed MCP server deployments across enterprise environments between October 2024 and November 2025, examining both the published ecosystem and real-world enterprise adoption patterns. The findings reveal silent, explosive adoption with total security blindness.

**The bottom line:** Your employees are installing arbitrary code from npm, GitHub, and PyPI - registries with no verification mechanisms - and feeding your most critical non-human identities to that code. You have no visibility into this.

The problem is accelerating.

## What we found

**15.28%**

**of enterprise employees** run at least one local MCP server (1,528 per 10,000 employees)

**86%**

**local vs 14% remote architecture** - developers choose local servers despite remote alternatives being available

**38%**

**are unofficial implementations** - unvetted code from unknown authors with full credential access

**3,056**

**total server installations** in a typical 10,000-person organization (average 2 per user, maximum 10+)

**95%**

**run on employee endpoints** where security tools have no visibility

**5%**

**run in production** CI pipelines, cloud workloads, and automated systems

**3%**

**of published servers contain valid, hardcoded credentials** embedded in their source code

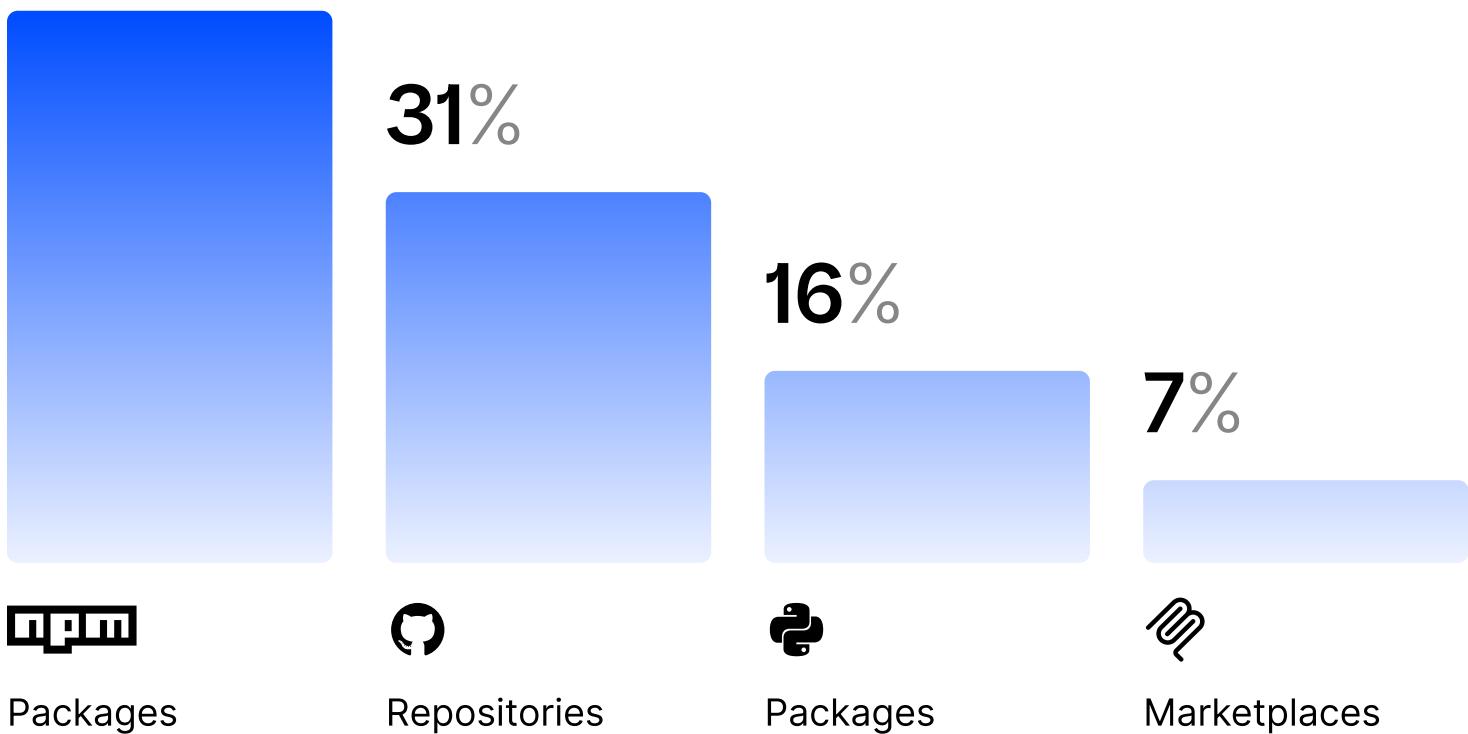
**115**

**distinct enterprise services** exposed:



## Distribution sources

46%



**In a 10,000-person organization:** 15.28% of employees (1,528) run an average of 2 MCP servers each - 3,056 total deployments. Of these, 38% (1,161 servers) are unofficial implementations from unknown authors with direct access to AWS keys, GitHub tokens, database passwords, and service account credentials stored in plaintext. 86% run locally on endpoints despite remote alternatives.

**The ecosystem:** 2,200% growth in 13 months. 38% remain unofficial because npm, PyPI, and GitHub don't verify packages. These servers connect to 115 enterprise services: AWS, Atlassian, GitHub, Docker, Snowflake, Postgres, Slack, Terraform, and 107 others.

**What you cannot see:** Which MCP servers exist in your environment, which are official, what credentials they access, what connections they make. The ecosystem grew 2,200% in 13 months. Developers choose local architecture 86% of the time. Registries publish without verification. You're operating 1,161 unofficial servers with access to your infrastructure credentials. Visibility is not optional.





# 2,200%

13 Months

📎  
Server

86%  
Local

14%  
Remote

✓  
**62%**  
Official

✗  
**38%**  
Unofficial

## Distribution breakdown

npm

npm

**46%**

GitHub



**31%**

PyPI



**16%**

Marketplaces



**7%**

**15.28%**

Adoption

100%



**115**

Services



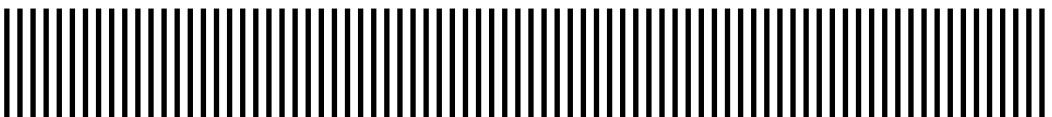
**2**

Avg. per user



**3%**

Hardcoded Secrets



**5%** Production

**95%** Endpoints



Model Context Protocol: A View from the Trenches.

What Enterprise MCP Deployments Actually Look Like. © 2025 Clutch Security

# The MCP Ecosystem: 2,200% Growth in 13 Months

Model Context Protocol launched in late 2024 as a standard for connecting AI assistants to external tools and data sources. The adoption curve wasn't gradual. It was explosive.

**October 2024:** 3 published MCP servers. **November 2025:** 6,878 published MCP servers. **That's 2,200% growth in 13 months.**

The curve accelerated throughout 2025. January saw 253 servers. By April, that number hit 1,776. August crossed 3,863. November 2025 alone added 996 new servers - more than existed in the protocol's first six months combined.

## Monthly MCP server publications

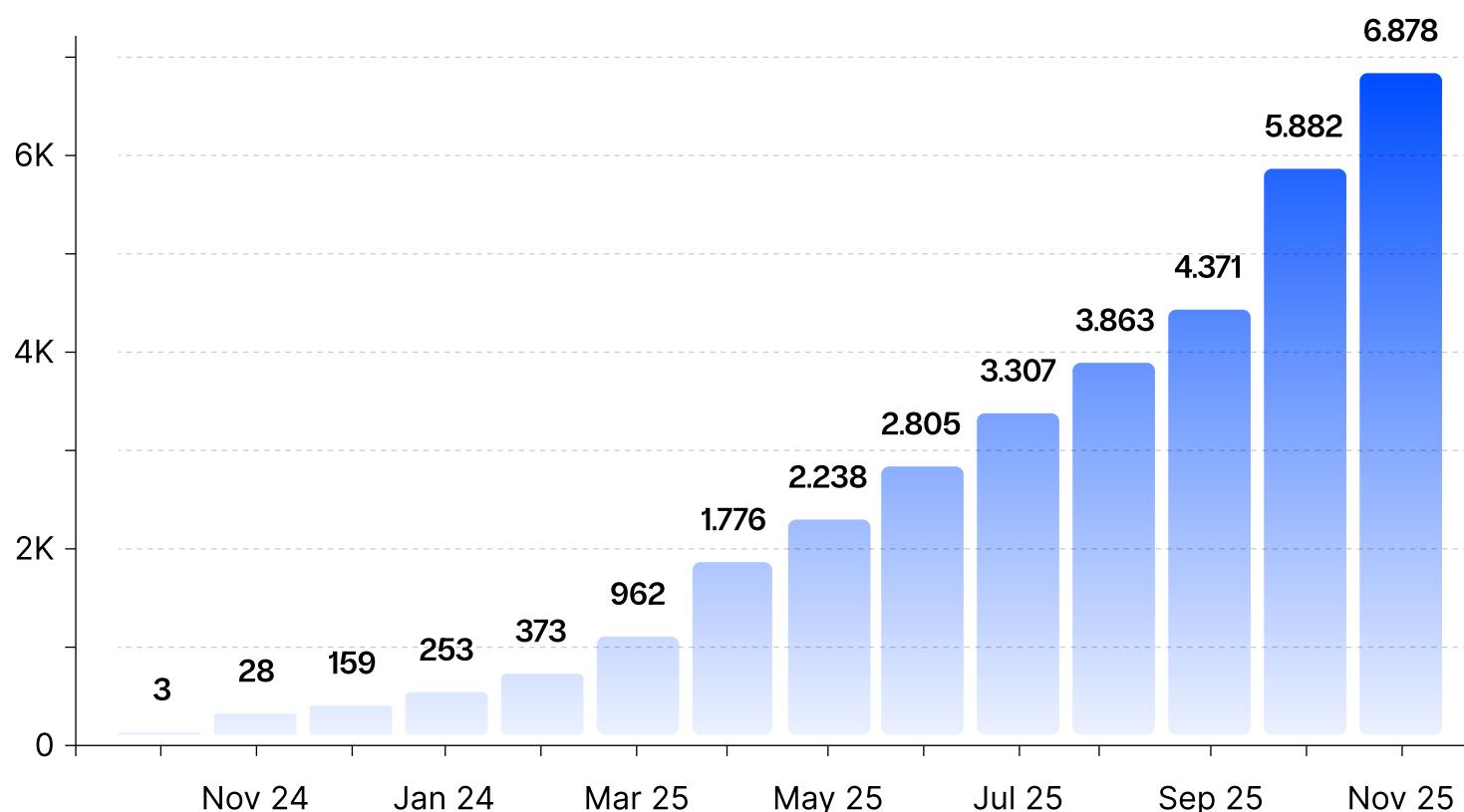
Oct 2024 - Nov 2025

► 2,000%

Growth in one year

⌚ +996 servers

Nov 2025, more than first 6 months combined



Developers discovered they could extend Claude, ChatGPT, and other AI assistants with specialized capabilities through single-command installation. The value proposition was immediate: `npm install @modelcontextprotocol/server-github` and you have GitHub automation inside your AI workflow.

Some developers didn't even bother with installation. They used `npx -y [package-name]` to run MCP servers on-demand without local installation. The `-y` flag auto-accepts all installation prompts. This "I'm feeling lucky" approach means arbitrary code execution from unknown authors with a single command, no vetting, no verification.

This created conditions for widespread adoption without governance. No procurement process, no security review, and no architectural discussion. Developers found MCP servers through search, installed or ran them immediately, configured credentials, and gained productivity. Meanwhile, security teams saw none of it.

Clutch's visibility into enterprise environments positioned us to quantify what no one suspected: MCP adoption was happening at massive scale, completely invisible, with significant security implications. The 2,200% growth rate shows no signs of slowing.



# In The Trenches: What We Found in Enterprise Deployments

Our analysis of enterprise MCP deployments reveals adoption at scale.

**15.28%**

of enterprise employees are running at least one local MCP server

In a 10,000-employee company, that translates to 1,528 employees with MCP servers and 3,056 total installations. Employees average 2 servers each. Some endpoints run 10+ distinct servers.

Each server represents code downloaded from npm, GitHub, PyPI, or arbitrary unofficial MCP Marketplaces. Each server is configured with credentials for enterprise services. Each runs with developer privileges. Multiply 1,528 users by 2 servers by credentials per server and the exposure compounds rapidly.

**86%**

of MCP users adopt local servers versus 14% using remote alternatives

This architecture choice matters. Local servers run on endpoints with full credential access. Remote servers run in vendor-controlled environments where credentials never leave secure infrastructure. Developers overwhelmingly choose local despite remote options being available from major vendors. The local model provides flexibility, immediate access, and no intermediary infrastructure. The 86/14 split demonstrates market preference, not architectural limitation.

Of those local server deployments, 95% run on employee endpoints where security tools have no visibility. The remaining 5% run in production environments - CI pipelines, cloud workloads, Kubernetes pods, and other automated systems. These are more dangerous: persistent, privileged, automated, often running with credentials that have broader access than individual developers.



**38%**

of deployed servers are unofficial implementations

Not vendor-published. Not verified. Not reviewed. Community packages from unknown authors, GitHub repositories with no organizational affiliation, npm modules that appeared in search results. In that same 10,000-person organization, that's 1,161 instances of unvetted code with credential access.

The deployment model makes this inevitable. MCP servers install in seconds with `npm install -g salesforce-mcp-enhanced` or run immediately with `npx -y salesforce-mcp-server`. No approval is required, and no alert is generated. No audit trails beyond package manager logs that security teams don't monitor for this purpose.

**95%**

of servers run on employee endpoints

The remaining 5% run in production environments. CI pipelines, cloud workloads, Kubernetes pods, and other automated systems. These are more dangerous: persistent, privileged, automated, often running with credentials that have broader access than individual developers.

A compromised MCP server on a developer laptop exposes that developer's credentials. A compromised MCP server in a CI runner exposes deployment keys, cloud infrastructure access, and production secrets. The 5% in production represents concentrated risk.



01	 Employees	10,000
02	 Adoption	15.28%
03	 Users	1,528
04	 Servers (Avg.)	×2
05	 Total	3,056
06	 Unofficial	38%
07	 Unvetted servers with credential access	1,161



2,200%

Growth



# The Target Surface: 115 Services Exposed

MCP servers don't connect to theoretical systems. They connect to the infrastructure that runs your business using the non-human identities that provide programmatic access to those systems.

Each service in our analysis represents a category of non-human identities at risk. AWS deployments expose access keys and secret keys. GitHub deployments expose personal access tokens and deploy keys. Atlassian deployments expose API tokens for Jira and Confluence. Snowflake deployments expose account credentials and OAuth tokens. These aren't abstract vulnerabilities - they're the API keys, service account credentials, and access tokens that your infrastructure depends on.

Our analysis of enterprise deployments **identified 115** distinct services with active MCP server connections. The distribution reveals what's actually at risk:

-  **Generic utility servers** represent 20.5% of deployments. These are tools like memory management, filesystem access, and sequential thinking. They don't require credentials but they normalize casual MCP server installation. When developers routinely run 2-3 generic servers, adding one more for AWS or GitHub requires no mental gear shift.
-  **Browser automation** accounts for 18.6% of deployments through Playwright, with additional concentration in Puppeteer (1.3%). Developers use these for automated testing, web scraping, and workflow automation. All require credentials to access internal applications and data.





**Atlassian dominates collaboration tools** at 12.3% of deployments. Developers wanted Jira and Confluence integration. They searched, found implementations, installed them. Whether that's one viral unofficial package or dozens of competing implementations, the pattern is clear: demand drives installation without vetting. Every unofficial Atlassian server represents potential access to your project management systems, documentation repositories, and team workflows.



**AWS leads cloud infrastructure** at 5.0% of deployments. Docker follows at 4.7%. These servers have access to cloud credentials that control your infrastructure, storage, compute resources, and network configuration. Microsoft (0.8%) and Google Cloud (0.5%) add to the cloud exposure, along with Cloudflare (0.3%).



**DevOps and source control** shows heavy concentration: GitHub (3.6%), Terraform (3.3%), Notion (3.3%), Git (3.3%), and GitLab (2.5%). These servers access source code repositories, container registries, infrastructure-as-code definitions, and CI/CD configurations. CircleCI (0.3%) and Azure DevOps (0.1%) represent additional CI/CD exposure.



**Data platforms** include Postgres (1.2%), Snowflake (0.4%), and smaller deployments across Supabase, Databricks, MongoDB, SQLite, DuckDB, and MySQL. These servers can query, extract, or modify production data.



**Other tools** like Context7 (3.5%), Figma (2.0%), Altimate AI (2.2%), and Slack (1.4%) round out the collaboration and productivity tools. Development tools include Postman (0.6%), Auth0 (0.4%), and Semgrep (0.4%). Security and monitoring tools like Wiz (0.6%), Sentry (0.3%), and Datadog (0.1%) also appear in the deployment distribution.



20.5%

18.6%

12.3%

5.0%

4.7%

3.6%

3.3%

3.3%

3.3%



Think about your technology stack. The services listed above aren't exotic outliers - they're standard enterprise infrastructure. The probability that unofficial MCP servers exist for your core services is high. The probability that they're installed in your environment without your knowledge is higher.

The popularity correlation matters. Atlassian's 12.3% deployment share didn't happen by accident. High adoption makes these services prime targets. Attackers building a malicious "atlassian-mcp-pro" server know they'll find users.

# The Architecture Problem: Plaintext Credentials by Design

MCP servers operate in two modes with fundamentally different security models.

Remote servers run in vendor-controlled environments. Anthropic hosts an MCP server for GitHub integration. Your AI assistant sends requests to Anthropic's infrastructure, the server validates and executes actions, credentials never leave the hosting environment. The trust boundary holds.

Local servers run on developer endpoints as arbitrary processes. Installed via package managers, configured with local credentials, executing with developer privileges. No sandboxing, full filesystem access, standard network egress. This is where 86% of adoption concentrates.

The preference for local architecture is clear in the data. Our analysis reveals that 86% of MCP adoption uses local servers despite remote alternatives being available from major vendors. Developers choose local architecture because it provides flexibility, immediate access, and no intermediary infrastructure. The 14% remote vs 86% local split demonstrates market preference, not architectural limitation. Security teams hoping for a shift to remote-only deployments are fighting against overwhelming developer behavior.

The local deployment model creates direct credential access. The MCP server runs as a program with developer privileges. It reads configuration files where credentials are stored in plaintext.



Here's the actual [.env.example](#) file from an unofficial [Salesforce MCP server](#) published on GitHub (42 stars, actively used):

Code    Blame    4 lines (4 loc) · 143 Bytes

```
1 SF_LOGIN_URL=https://login.salesforce.com
2 SF_USERNAME=your-username@example.com
3 SF_PASSWORD=your-password
4 SF_SECURITY_TOKEN=your-security-token
```

That file sits in the MCP server's directory. The server reads it when it starts. Developers replace the placeholder values with real credentials. A malicious server doesn't need to exploit anything. It reads the config file it was given and sends those credentials wherever it wants.

Same pattern for environment variables, JSON config files, and interactive prompts that store values locally. No encryption, no secure enclaves, no hardware-backed keystores. Plaintext files that any process with user-level permissions can read.



**38%**

## unofficial rate inverts the security model

The 38% unofficial rate inverts the security model. Official servers from AWS, GitHub, Anthropic, Slack have vendor reputation, source code visibility, and some level of review. Unofficial servers from unknown npm publishers or anonymous GitHub accounts have none of that. The code could be legitimate productivity tools or credential harvesters. The installation experience is identical.

Package managers provide no security guarantees. npm doesn't require code signing, and GitHub repositories can be created anonymously. Package names aren't protected - publishing `@mcp/server-salesforce` doesn't require Salesforce's permission. Download counts and stars are easily manipulated. There's no verification chain.

A developer searching for "mcp server aws" receives results mixing official AWS implementations with community packages and potentially malicious code designed to look legitimate. The decision comes down to search ranking and documentation quality. Neither correlates with security.

With 2,200% growth in published servers over 13 months, the volume of unofficial implementations is accelerating faster than any vetting process could handle. Developers using `npx -y` to run servers on-demand don't even see the code before it executes.



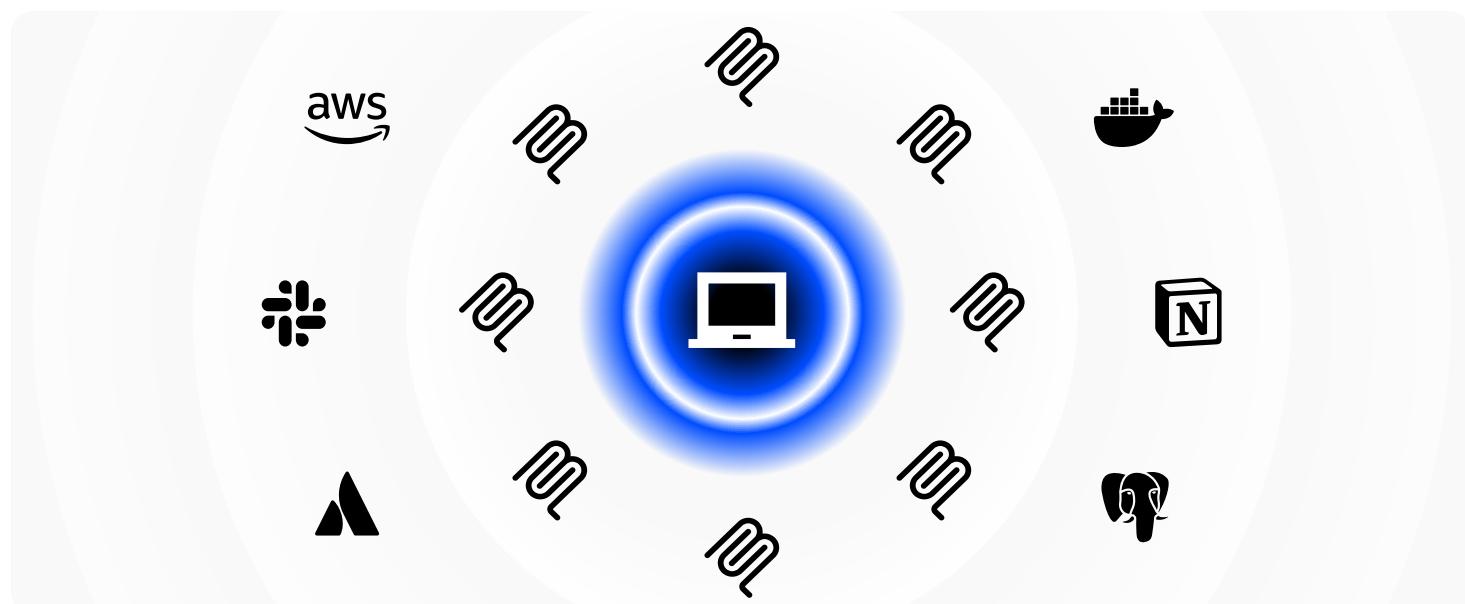
# Speed, Scale & The Detection Gap

The architecture creates exposure. The adoption numbers multiply it. The 2,200% growth rate accelerates it. The attack execution is fast. Detection capabilities are nonexistent.

## The Multiplication Effect

Employees average 2 MCP servers. Some run 10+. Each server installation requires non-human identities for configuration. A developer with 5 MCP servers configured for AWS, GitHub, Snowflake, Slack, and Postgres has exposed at least 5 distinct non-human identities on a single endpoint.

Scale that across 1,528 employees in a 10,000-person org: 3,056 servers, thousands of configured non-human identities, all of them in plaintext configuration files on developer endpoints.



## Attack Timeline: 60 Seconds

Attacker publishes an MCP server to npm with a compelling name: github-automation-pro, aws-dev-tools-enhanced, atlassian-productivity-pack. The server provides actual functionality while simultaneously exfiltrating credentials.

The developer searches for the tool, finds it, runs `npm install -g github-automation-pro` or `npx -y github-automation-pro`. Installation or execution completes in seconds.

The server prompts for GitHub credentials or directs the developer to create a config file. Developer complies - these are their credentials for their productivity tools. The config is saved locally.

The AI assistant starts the server. Server loads the config (credentials included), initializes functionality, makes a background HTTPS POST to an attacker-controlled endpoint disguised as analytics. Credentials exfiltrated.

 Timeline from installation to compromise → **60 sec**



## Publisher Credential Exposure

Our analysis of the published MCP ecosystem found that 3% of servers contain valid, hardcoded credentials in their source code. Active Anthropic Claude keys, AWS access keys, Google API keys, Stripe keys, GitHub tokens, npm tokens, OAuth secrets, GCP service account private keys - embedded in code published to npm and GitHub.

```
2903 VERSION_DOWNLOAD_PATH = path2.join(  
2904   getCacheHome(),  
2905   "mcp_use",  
2906   "download_version"  
2907 );  
2908 PROJECT_API_KEY = "phc_lyTtbY" * * * * * * * ;  
2909 HOST = "https://eu.i.posthog.com";  
2910 SCARF_GATEWAY_URL = "https://mcpulse.gateway.scarf.sh/events-ts";  
2911 UNKNOWN_USER_ID = "UNKNOWN_USER_ID";  
2912 _currUserId = null;  
2913 _posthogClient = null;  
2914 _scarfClient = null;  
2915 _source = "typescript";  
2916 constructor () {  
2917   const isNodeJS = isNodeJSEnvironment2();  
2918   const telemetryDisabled = typeof process !== "undefined" && process.env?.MCP_USE_ANONYMIZE  
2919   this._source = typeof process !== "undefined" && process.env?. MCP_USE_TELEMETRY_SOURCE ||
```

This represents risk primarily to the publishers who accidentally committed credentials. Some providers auto-revoke (GitHub Secret Scanning, AWS Secret Detection). Others remain active. The 3% rate reveals casual credential management across the MCP ecosystem.



# The Visibility Imperative

These numbers represent non-human identity exposure at scale. The 3,056 servers in a typical 10,000-person enterprise translate to thousands of API keys, access tokens, service account credentials, and OAuth tokens stored in plaintext configuration files. Each unofficial server (38% of deployments) represents unvetted code with direct access to the non-human identities that control cloud infrastructure, data platforms, and enterprise applications.

MCP servers provide real productivity value. Developers adopt them because they work. The single-command installation, immediate functionality, and seamless AI integration drive adoption regardless of security concerns. Prohibition will fail the same way it failed for cloud services and SaaS applications.

The architecture won't change. Local MCP servers with plaintext credential access are fundamental to the developer experience that makes the protocol valuable. The 86% local adoption rate proves this - even when remote alternatives exist, developers choose local servers. Remote-only servers would eliminate the flexibility that drives adoption. The market has spoken: 86% local, 14% remote. The 2,200% growth rate shows the model is working exactly as designed. Security must adapt to this reality, not hope it reverses.

What needs to change is security visibility and governance.

Security teams need answers to specific questions: What MCP servers exist in our environment? Which are official vendor implementations versus unofficial community packages? What credentials are those servers configured to access? What network connections are they making? What actions are they performing? Which servers are running in production environments versus developer endpoints?



Without this visibility, the alternative is blind trust. Trust in 3,056 servers. Trust that 38% unofficial implementations from unknown authors won't exfiltrate credentials to AWS, Snowflake, GitHub, Atlassian, Slack, and 110 other enterprise services. Trust that the 2,200% growth rate won't continue accelerating. Trust that the 86% local preference will somehow shift to remote. Trust that developers will vet code before running arbitrary MCP servers with `npx -y`.

The MCP ecosystem will continue growing. November 2025 added 996 servers - more than the first six months of the protocol's existence. December will add more. January will add more. AI assistants will become more capable. Developers will continue finding productivity tools. The unofficial implementation rate will remain near 40% because package managers provide no meaningful verification. The local vs remote split will remain 86/14 because the market has chosen.

This research provides the baseline. The adoption patterns, unofficial rates, service exposure, and detection gaps documented here represent the current state across enterprise environments. The visibility requirement is clear. The question is whether security teams will build that visibility before or after the first major MCP-based credential theft incident.



# About Clutch Security

Clutch Security provides the industry's most comprehensive platform for non-human identity security, including the MCP servers and AI agents that use those identities. The platform delivers complete visibility across cloud environments, SaaS applications, on-premises infrastructure, code repositories, CI/CD pipelines, and vaults, enabling enterprises to discover, manage, and protect the API keys, service accounts, tokens, and credentials that power modern infrastructure.

