MCP SERVER | NO CODE | CLOUD HOSTED

Cerbos MCP for AI Agents

Manage complex access control policies and audit permissions in real time

Cerbos provides an open standard connection for managing access control policies and checking permissions using your AI agent. It lets developers verify exactly if a user can perform a specific action on a resource, keeping authorization logic outside of application code. You'll audit complex security rules, manage policies (RBAC/ABAC), and generate query plans directly through natural language prompts.

A+ Quality Score 98.33/100

authorization | rbac | abac | policy-as-code | access-control

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/cerbos-access-control

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

Cerbos (Access Control) MCP

19 tools available
Cloud-hosted on Vinkius

Managing permissions used to mean writing messy, complex logic into your main codebase—logic that breaks when requirements change or needs auditing. This MCP changes that process entirely. It lets you connect your AI agent to a dedicated policy engine, treating authorization rules like data instead of code. You can now ask questions like, 'Can this user delete this report?' and get an immediate, definitive answer based on your current security policies. Furthermore, if you need to check the health of the entire system or audit past access attempts, your agent handles it without needing manual API calls. By using Vinkius as your central catalog, you connect once from any compatible client and gain control over all these policy functions through conversation.

Core Capabilities

01 — Check resource permissions

Verify if a specific user or group is authorized to perform an action on a given data object.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/cerbos-access-control** — connect your AI agent in three steps.

01 Subscribe to this MCP on Vinkius.

02 Provide your Cerbos Policy Decision Point URL and necessary administrative credentials to your AI client.

03 Ask your agent a natural language question—like, 'Show me all policies related to the expense report resource.'—and it handles the rest.

The bottom line is that you get an immediate, conversation-based interface for managing and testing core security rules without writing any code.

Built For

Security Engineers and Backend Developers need this. If your job involves figuring out why a user suddenly lost access to a feature, or if you spend time manually checking policy definitions in an admin console, this MCP saves you hours of clicking through dashboards.

Security Engineer

Auditing existing policies and verifying permission logic for compliance checks without running manual scripts.

Backend Developer

Testing complex authorization scenarios early in the development cycle, especially generating database filter plans before writing service code.

Compliance Officer

Retrieving detailed access audit logs and verifying that policy definitions meet specific organizational security standards.

What Changes When You Connect

01 You stop manually running API calls. Your agent handles policy definitions, allowing you to quickly use `list_policies` or `update_policy` by simply asking a natural language question.

02 Audit risk instantly. Need to know if an action was allowed? Running the `check_resources` tool lets you test permissions for any user against any resource without touching your code base.

03 Improve developer speed. Instead of guessing how database filters work, use `plan_resources` to generate and test optimized query plans based on policy rules directly within your agent chat.

04 Ensure compliance easily. By using the audit logging tools like `list_audit_logs`, you can retrieve comprehensive records proving that security standards were followed.

05 Gain visibility into system health. You'll get immediate answers about performance or status by running `get_health` or pulling operational metrics with `get_metrics`.

Real-World Applications

Verifying a new feature rollout

A backend developer needs to ensure that only managers can approve expenses. They prompt their agent: 'Check if the manager role can execute the approval action on expense reports.' The agent uses `check_resources` and confirms the exact required permissions before the code is merged.

Investigating a data breach

A security engineer discovers unusual access patterns. They prompt their agent to 'List all audit logs for resource type X in the last week.' The agent uses `list_audit_logs` and provides a chronological list, immediately narrowing down the scope of the investigation.

Refactoring authorization logic

A developer needs to change how department ownership works. They ask their agent to 'Update policy resource.dept.v1 to include owner attribute.' The agent uses `update_policy` and confirms the successful modification.

Optimizing reporting queries

The team needs a report that only shows records owned by the current user's department. Instead of writing complex SQL, they prompt their agent to 'Generate an AST query plan for filtering resources based on department ownership.' The agent uses `plan_resources`.

Patterns to Avoid

Hardcoding permission checks

✗ AVOID

Writing `if user.role == 'admin' and resource.status == 'active': allow()` directly into the application service layer. This means every time a rule changes, a developer must redeploy code.

✓ INSTEAD

Instead of hardcoding logic, use your agent to run `check_resources`. You ask your agent: 'Can this user delete this record?' The policy engine handles the decision without you touching the source code.

Ignoring existing policies

✗ AVOID

Building a new feature and forgetting that an old, undocumented policy might interfere with the new functionality. You'll run into unpredictable bugs in production.

✓ INSTEAD

Always start by running `list_policies` to see everything active. Then use your agent to check for conflicts or gaps before you proceed.

Assuming policy structure

✗ AVOID

Manually trying to write a complex new rule without knowing the exact schema definition, leading to invalid policies that break at runtime.

✓ INSTEAD

First, run `list_schemas` and then use `get_schema` on specific resources. This ensures you have the correct data structure before attempting to add or modify any policy.

The Right Fit

Use this MCP if your core problem is determining *authorization*—that is, answering the question 'Does X have permission Y on Z?' You need a way to test, audit, and manage complex security rules (RBAC/ABAC) outside of your primary application code. Don't use this if you just need simple data retrieval or CRUD operations; those belong in standard database connectors. If all you need is to know the current health status, while `get_health` works, consider a simpler monitoring tool. When in doubt about policy complexity, always check with `list_policies` first.

Cerbos (Access Control) MCP: Policy Management and Auditing

Today, managing access control means developers spend cycles writing boilerplate code that checks user roles against resource types. When a policy needs to change—say, moving from role-based access to attribute-based access—the team has to manually hunt down every instance of the old logic across dozens of files and redeploy everything just for a rule tweak.

With this MCP, you simply describe the required changes in plain English. Your agent interacts with the system using tools like `add_policy` or `update_policy`. You get immediate confirmation that the policy was correctly registered and activated at the engine level, drastically reducing deployment risk.

Cerbos (Access Control) MCP: Testing Authorization in Development

The manual development process involves setting up mock users, writing unit tests for every single access path, and then running those tests to ensure the correct policies are hit. This is time-consuming, brittle, and often misses edge cases involving overlapping permissions.

Now, you can ask your agent to generate a query plan using `plan_resources` based on complex policy logic. You get an optimized, data-filtering blueprint before writing any database interaction code. It's testing the rule itself, not just the endpoint.

20 Tools in the Cerbos (Access Control) MCP for Policy Management

Use these tools to manage every aspect of your application's security layer—from creating policies to auditing historical access logs.

#	TOOL	DESCRIPTION
01	`add_policy`	Creates a completely new access control rule for the system.
02	`add_schema`	Adds or updates the structural definition (schema) of a resource type.
03	`authzen_evaluation`	Performs a single, immediate test to determine if access is allowed using AuthZEN rules.
04	`authzen_evaluations`	Runs multiple, batch checks to evaluate permissions across several scenarios at once.
05	`check_resources`	Checks if a set of users have the required rights for specific resources.
06	`delete_policy`	Removes an existing policy entirely using its unique identifier.
07	`disable_policy`	Temporarily turns off a policy without deleting it, useful for testing or maintenance.
08	`enable_policy`	Restores a previously disabled access control policy to active status.
09	`get_authzen_config`	Retrieves metadata about the AuthZEN configuration setup.
10	`get_health`	Returns the current operational status of the Cerbos policy engine.
11	`get_metrics`	Pulls performance metrics data, useful for monitoring system load and usage patterns.
12	`get_policy`	Retrieves the full text of a specific policy by its ID.
13	`get_schema`	Fetches the detailed structure definition for a schema type.
14	`get_server_info`	Gets version and build information about the Cerbos server instance.
15	`list_audit_logs`	Generates a list of all recorded access control audit logs for review.
16	`list_policies`	Displays every single policy currently active in the system.
17	`list_schemas`	Lists all defined resource schemas available to the application.

#	TOOL	DESCRIPTION
18	plan_resources	Creates a database query plan (AST) that filters resources according to complex authorization logic.
19	update_policy	Modifies the rules and parameters of an existing policy.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U I need to confirm if a user in the 'read-only' group can access financial records for Q3.

Policy Check Results

- **Principal:** `user:jdoe` (Group: read-only)
- **Action:** Read
- **Resource:** Financial Record (Q3)
- **Result:** ✅ **ALLOWED.** The policy 'finance.read.q3' grants read access to all users tagged with the 'read-only' group.

No further actions needed; the system is compliant.

U List all policies that affect resource types related to employee payroll.

Payroll Policy Audit

I found 5 policies affecting payroll records:

- `resource.payroll.view`: Allows read access for HR staff.
- `resource.payroll.update`: Restricted to Payroll Managers only.
- `resource.salary.delete`: Only permitted if the user is a system admin.
- `resource.tax.read`: Available globally.
- `resource.bonus.create`: Requires multi-factor approval.

U What are the necessary database filters to ensure I only see records owned by my department?

Query Plan Generated

The following Abstract Syntax Tree (AST) plan will filter your results correctly:

```json
{
"field": "department_id",
"operator": "equals",
"value": "{{current_user.dept_id}}",
"join_condition": "is_active"
}
```

This plan ensures that only records matching your department ID are returned.

Frequently Asked Questions

01 **How does the Cerbos MCP help manage my company's access control policies?**

You can use this MCP to centralize policy management. Instead of coding rules into your application, you define them using tools like `list_policies` and `add_policy`. Your agent then handles all checking and enforcement logic for you.

02 **Can I audit past security events with the Cerbos MCP?**

Yes. You can use this MCP to retrieve comprehensive records of every access decision using tools like `list_audit_logs`. This lets compliance teams prove exactly who accessed what and when.

03 **Does this MCP help developers write more secure code?**

Absolutely. You can use it during development to test authorization scenarios by generating query plans with `plan_resources`, ensuring your database queries are always correctly scoped based on security rules before they ever reach production.

04 **What if I need to update a policy, but I'm not sure which one is correct?**

You can list all available policies using `list_policies` and then use the agent to retrieve specific details about any rule with `get_policy`. This gives you the full context needed for safe updates.

05 **How do I check if a user has permission without writing boilerplate code?**

You simply ask your AI agent to check permissions using tools like `check_resources`. The MCP handles all the complex policy evaluation, returning a simple ALLOWED or DENIED status for you.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"cerbos-access-control": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

Cerbos (Access Control) is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Cerbos (Access Control). All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	Cerbos (Access Control) MCP
Server ID	`019e3875-7d51-7276-a490-038c89aa5d52`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/cerbos-access-control.