

MCP SERVER

NO CODE

CLOUD HOSTED

42Crunch MCP for AI Agents

Automating API Security and OpenAPI Governance

42Crunch automates API security testing directly from your AI agent. Connect this MCP to continuously audit and secure your APIs without leaving your conversation window. You can manage entire collections of API specifications, run static vulnerability scans on OpenAPI definitions, and retrieve detailed compliance reports immediately.

F Quality Score 3.6/100

api-security

vulnerability-scanning

openapi

swagger

security-audit

shift-left



The infrastructure that powers AI agents in the real world.



Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest
managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect 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.

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.

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.

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

Encryption at rest

Ed25519

PKI vault signatures

24h TTL

Ephemeral session keys

V8 Isolate

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.

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.

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.

06 — Honeypot Trap System

Phantom credentials are injected into isolated environments. If a honeypot 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.

FinOps

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

Firewall & DLP

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

Agent Activity

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

Tool Health

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

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.

42Crunch MCP

10 tools available

Cloud-hosted on Vinkius

Building robust APIs means keeping up with changing security standards. This connector lets you talk through complex security governance right from your natural conversation flow. Instead of manually uploading specs to a web dashboard or jumping between tools, you simply tell your agent what needs auditing. Your agent handles the entire process: it can retrieve metadata for specific API collections, import new OpenAPI definitions into those groups, and then trigger deep static security audits on the specifications.

If you need proof that an endpoint behaves correctly in practice, the connector also manages dynamic conformance scans, showing historical execution reports of any implementation flaw. It's a full lifecycle audit tool, letting developers stay focused on coding while keeping security compliance top-of-mind. When you connect this MCP via Vinkius, your agent gets access to all these critical functions, making API security governance conversational.

Core Capabilities

01 — View Collection Overview

List all managed API collections, viewing the aggregated security score for each one.

03 — Perform Static Audits

Initiate comprehensive security scans on an API's definition to score design risks and find flaws like missing authentication details.

02 — Control Definitions

Import new OpenAPI specs into a collection or delete definitions that are no longer needed in production.

04 — Inspect Live Behavior Scans

List past dynamic conformance scans and retrieve detailed execution reports highlighting undocumented endpoints or implementation issues.

One Click on Vinkius — From Prompt to Execution

Available at vinkius.com/mcp/42crunch — connect your AI agent in three steps.

- 01 Subscribe to this MCP in Vinkius and provide your 42Crunch API Token.
- 02 Tell your agent which collection needs attention, or if you need to import a new OpenAPI file.
- 03 Ask the agent to trigger an audit or retrieve a report. It runs the scan against the platform and presents the findings back to you immediately.

The bottom line is that this MCP brings complex API security management directly into your chat window, so you don't have to context switch at all.

Built For

This MCP is essential for DevSecOps Engineers and Platform Teams who are tired of manually jumping between dashboards just to check API compliance. If your job involves governing a microservice ecosystem, this connector makes security audits part of your daily conversation.

DevSecOps Engineer

Needs to quickly trigger static security audits after code pushes and get immediate, actionable steps for fixing OWASP vulnerabilities.

Platform Architect

Manages the overall health of a microservices ecosystem, keeping track of security scores across multiple API collections.

Backend Developer

Needs to import new spec versions and run comparisons against old specs to see how the security grade has changed without leaving their IDE.

What Changes When You Connect

- 01 Instead of manual uploads, you can use `import_api` to add new OpenAPI specs directly from the conversation. This keeps your workflow moving.

-
- 02** You don't have to guess if an API is secure; just ask for a report. The agent uses `get_audit_report` and delivers clear, actionable findings on vulnerabilities like missing rate limits.
-
- 03** Need to track security debt? Use `list_collections` to see the overall risk profile of your entire microservice portfolio at a glance.
-
- 04** Comparing versions is easy. You can use `trigger_audit` whenever an API spec changes and immediately compare the new score against the old one, all in text chat.
-
- 05** Don't forget live testing. If you suspect a behavior issue, running a dynamic scan via `list_scans` provides proof of undocumented endpoints or flaws.
-

Real-World Applications

Security review for a new microservice

A platform architect has just finished defining a service's OpenAPI spec. They ask the agent to run `trigger_audit` on that definition, instantly getting an audit report that scores critical risks like insufficient scope restriction before writing any code.

Checking overall compliance for an application

A DevSecOps Engineer wants a full security picture of their payment services. They ask the agent to run `list_collections` and then request details on any collections scoring below 70/100.

Auditing retirement of old APIs

A developer needs to decommission a legacy API. They first use `get_api` to retrieve its score, then confirm it by running `delete_api`, ensuring the record is cleaned up properly.

Investigating suspicious runtime behavior

A QA engineer notices an endpoint behaving unexpectedly. They use the agent to list historical scans (`list_scans`) and retrieve a detailed `get_scan_report` to confirm if it's an undocumented flaw.

Patterns to Avoid

Treating security audits as manual tasks

✗ AVOID

A user manually downloads the report, opens a spreadsheet, and has to copy-paste findings into JIRA tickets one by one.

✓ INSTEAD

Instead, simply ask your agent to ``get_audit_report``. The agent processes the data and gives you a summarized list of critical issues right in the chat, ready for immediate action.

Forgetting to update specs before auditing

✗ AVOID

Running an audit on old documentation means you get false positives or miss current vulnerabilities because the spec hasn't changed.

✓ INSTEAD

Always use ``import_api`` first to bring in the latest OpenAPI definition, then immediately run ``trigger_audit``. This guarantees you test what you just built.

Ignoring collection health

✗ AVOID

Only checking individual APIs leads to blind spots; a single weak link can compromise an entire service group.

✓ INSTEAD

Start by using ``list_collections`` and then ask the agent for details on the overall security score of a whole grouping via ``get_collection``.

The Right Fit

Use this MCP if your primary pain point is context switching during API governance. If you need to audit an API's definition (static analysis) or its live behavior (dynamic analysis), this connector works. It excels at reading existing OpenAPI specs, importing them into collections, and generating reports like `get_audit_report`. Don't use it if your only goal is simple documentation storage; you still need a dedicated repository for that. If your process involves complex, multi-step approvals or human review gates outside of the technical audit itself, you might need an integration with specialized workflow tools instead.

42Crunch MCP: Solving OpenAPI Security Gaps in API Governance

Today, securing an API is a mess. Developers have to write code, then export the spec, then manually upload that spec to a separate security dashboard. They run the audit, download a massive PDF, and then copy-paste the critical findings—the missing rate limits, the weak authentication points—into a ticket system to get fixed.

With this MCP, you keep everything in one place. You tell your agent to check the spec, it handles the entire audit process using `trigger_audit`, and it spits out the results directly into the chat window. It's immediate feedback without ever leaving your development environment.

42Crunch MCP: Governing API Security Scores Across Microservice Collections

Keeping track of security hygiene across dozens of microservices is a nightmare. You have to log in, navigate through multiple tabs, and cross-reference scores for every single collection just to know which service needs the most attention.

This MCP lets you bypass that manual work entirely. Use `list_collections` and then ask for details using `get_collection`. It gives you a clear, aggregated score for your entire portfolio instantly. You stop managing dashboards; you start governing security via conversation.

42Crunch: 10 API Security Tools for Governance

Invoke these tools to perform comprehensive actions, from listing entire API collections to triggering detailed vulnerability scans on specific definitions.

#	TOOL	DESCRIPTION
01	<code>get_audit_report</code>	Downloads a detailed static security audit report for any specified API definition.
02	<code>delete_api</code>	Removes an existing API definition from the platform's managed collection.
03	<code>get_api</code>	Retrieves specific metadata and the current security score for a single, named API.
04	<code>get_collection</code>	Gets metadata and the overall aggregated security score for an entire group of APIs (a collection).
05	<code>import_api</code>	Adds a new OpenAPI definition file into an existing API collection.
06	<code>list_apis</code>	Lists all API definitions within a collection, showing each one's ID and current security score.
07	<code>list_collections</code>	Retrieves a list of all managed API collections available on the platform.
08	<code>list_scans</code>	Shows a record of historical dynamic conformance scans run against a live API endpoint.
09	<code>get_scan_report</code>	Retrieves detailed results from the most recent dynamic conformance scan report for an API.
10	<code>trigger_audit</code>	Runs a fresh static security audit on any specified API definition after its source specification has been updated.

See It in Action

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

U What is the overall security score for all my internal microservices?



Internal Microservices Collection

- **Score:** 62/100 (Needs attention)
- **API Count:** 12
- **Primary Risk:** Authentication flow weaknesses.

We found two endpoints in that collection— `/user/profile` and `/data/export` —that are missing proper authorization checks. Would you like me to get the full metadata for those two APIs?

U Can I check if the 'User Management' API has been updated since its last audit?



Yes, let's trigger a fresh static security audit now.

Running scan on User Management...

✅ Audit Complete. Score: 89/100 (Improved)

Findings: We fixed the OAuth2 scope issue from last week. The only remaining minor risk is that the user schema accepts generic data, which isn't ideal. I recommend adding type constraints to your YAML.

U Show me all my API collections and their current status.



You currently manage 3 collections:

1. **Mobile Services:** Score: **85/100**. (4 APIs)
2. **Internal Microservices:** Score: **62/100**. (12 APIs)
3. **Legacy Billing:** Score: **30/100**. (2 APIs) 🔥

The Legacy Billing collection is highly uncompliant. Should I pull the details for that one and suggest immediate remediation steps?

Frequently Asked Questions

01 How can 42Crunch MCP help me audit my OpenAPI specs?

The MCP lets you talk through the entire auditing process. You can tell it to trigger a static security audit on any API spec, and it will return a detailed report right in your conversation showing exactly what's wrong with the design.

02 Is 42Crunch MCP only for checking vulnerabilities?

No. It manages more than just security. You can use it to import new API definitions into collections, list all existing APIs, and track the overall health score of your entire service portfolio.

03 What if I want to see how an API performs live?

You can run dynamic conformance scans. The MCP will list historical scan results and give you detailed reports that show what was actually observed when the API ran, not just what the spec says it should do.

04 Do I need to manage my APIs in multiple places?

No. This MCP centralizes your governance. You can list all your collections and view their combined security score from one place, eliminating dashboard hopping.

05 Can 42Crunch MCP help me with old or decommissioned APIs?

Yes. If an API is retired, you can use the tools to delete it from the platform and keep your records clean, while also allowing you to audit its final status before removal.

06 How do I start using 42Crunch MCP with my AI agent?

First, subscribe to this MCP in Vinkius and provide the required API token. After that, just ask your agent conversational questions about security audits or collection scores.

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 `"42crunch": { "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

42Crunch 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](https://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 42Crunch. 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	42Crunch MCP
Server ID	019d7545-0444-7142-b3e4-b6d705c15b1c
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/42crunch.