MCP SERVER | NO CODE | CLOUD HOSTED

Equixly MCP

Automate Full-Scope API Security Audits

Equixly MCP automates API security testing directly through your AI agent. Manage target services, upload OpenAPI specs, and run autonomous pentests to find critical vulnerabilities like BOLA and IDOR without manual configuration. It delivers detailed reports on exploitable flaws from any compatible client.

A+ Quality Score 98.33/100

api-security | penetration-testing | autonomous-testing | vulnerability-management | cybersecurity | logic-errors

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/equixly

SCAN TO VIEW

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.

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.

Equixly MCP

10 tools available
Cloud-hosted on Vinkius

Connect your Equixly account via Vinkius and give your AI client full control over API security testing and vulnerability management through natural conversation. You can start by registering a new target service, defining the base URL you want to protect. Next, upload comprehensive API specifications—OpenAPI or Postman files work great—to expand what the autonomous hacker knows about your system. When ready, simply trigger an attack session for BOLA, IDOR, and common injection flaws across all defined endpoints. Your agent tracks progress and lets you pull detailed lists of confirmed vulnerabilities, including severity ratings and remediation steps. You don't have to jump between a dashboard and your IDE; the process happens entirely through conversation with your AI client.

Core Capabilities

01 — Registering API Targets

You can establish new API services by defining their base URLs for continuous security monitoring.

02 — Expanding the Attack Surface

Upload OpenAPI, GraphQL, or Postman specifications to ensure the autonomous AI hacker has a complete map of your API endpoints.

03 — Running Automated Penetration Tests

Initiate comprehensive security scans designed to find specific flaws like Broken Object Level Authorization (BOLA) and IDORs.

04 — Analyzing Vulnerability Reports

Retrieve detailed lists of confirmed, exploitable security flaws, complete with OWASP mapping and suggested fixes.

05 — Monitoring Scan Progress

Track the real-time status of a test, seeing metrics like total requests made or endpoints explored.

06 — Retrieving Service Metadata

Fetch configuration details for any API service, including authentication hooks and safety settings.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/equixly** — connect your AI agent in three steps.

01 First, subscribe to this MCP and provide your Equixly API Token in the Vinkius setup.

02 Next, tell your AI client which APIs need protection; you can do this by using `create_service` or uploading specs via `upload_api_spec`.

03 Finally, use `trigger_scan` to start testing, and then ask for findings using `get_scan_findings`.

The bottom line is that you manage the entire lifecycle of security testing—from definition to discovery—without ever leaving your chat window.

Built For

This MCP is for Security Engineers and DevSecOps teams who are tired of manually updating tools, clicking through multiple dashboards, or writing complex scripts just to check if their API endpoints have vulnerabilities. You need a conversational way to audit security posture.

Security Engineer

You use this MCP to monitor the overall security health of production APIs and trigger full pentests without needing manual tool configuration.

DevSecOps Developer

You integrate security scans directly into your deployment pipeline, using the AI agent to run tests immediately after code pushes.

QA Engineer

You verify that new features adhere to strict security boundaries and test for business logic flaws through natural conversation with your agent.

What Changes When You Connect

01 You eliminate manual setup. Instead of configuring tools, you simply use `create_service` to define a new target URL and start monitoring its security posture instantly.

02 Maximized coverage means fewer blind spots. By using `upload_api_spec`, you feed the autonomous AI hacker every piece of documentation—OpenAPI, GraphQL, etc.—so no endpoint is missed.

03 Get actionable results immediately. Rather than just finding a flaw, `get_scan_findings` gives you the OWASP category and direct remediation guidance for fixing it.

04 Manage complexity with one command. Instead of running separate scripts for different types of tests, use `trigger_scan` to launch a comprehensive attack session covering BOLA, IDOR, and more.

05 Maintain audit trails easily. With `list_scans`, you track every test session, seeing the status and total vulnerability count without opening a dashboard.

Real-World Applications

Post-Deployment Security Check

A backend developer just pushed a new API endpoint. They ask their agent to run an audit on the service, triggering `trigger_scan` immediately. The agent confirms the scan is running and notifies them when they can use `get_scan_findings` to review any critical flaws before deployment.

Auditing a Legacy System

A security engineer needs to check an old, undocumented API. They use `list_services` to confirm the base URL and then manually feed documentation using `upload_api_spec`, ensuring the agent knows exactly what surface area to test.

Comparing Test Runs

A QA engineer needs to prove that a patch fixed a vulnerability. They use `list_scans` to find the previous failed scan and then run a new one, comparing the total flaw count using `get_scan`.

Decommissioning an API

The team is retiring an old payment gateway. Instead of manually deleting it from multiple systems, they use `delete_service`, ensuring all scan history and the service itself are cleanly removed.

Patterns to Avoid

Assuming full coverage

✗ AVOID

Only running a basic test without providing documentation means the agent only checks what it knows, leaving critical endpoints completely untouched.

✓ INSTEAD

Always supplement testing by using `upload_api_spec`. This ensures that even if an endpoint isn't called frequently, the autonomous hacker has its definition and can audit it.

Treating security as a one-time task

✗ AVOID

Running a scan once and assuming everything is safe. API threats change constantly, meaning yesterday's successful test might fail today.

✓ INSTEAD

Set up continuous monitoring by registering the service with `create_service` and running regular scans to keep your security posture current.

Trying to find a flaw without context

✗ AVOID

Running `trigger_scan` but not knowing if the results were critical, high, or low. You just get a raw list of findings.

✓ INSTEAD

After running a scan, always use `get_scan_findings` first. This filters and organizes the data by severity, making it immediately clear where you need to focus.

The Right Fit

Use this MCP if your primary goal is automated API security auditing—finding specific flaws like BOLA or IDOR across a defined set of endpoints. You should use it when you need to prove that an API meets a security standard before deployment, or when you're tracking changes to existing services using `get_service`. Don't use this if you are simply trying to look up user data, check general system health metrics unrelated to APIs, or manage basic configuration settings. For simple record management or querying non-API related business logic, a generic database connector is better. This MCP is specialized for deep, technical security analysis.

The struggle of manual API security audits

Today, checking if your APIs are secure is a nightmare of tabs and exports. You have to manually update tool configurations, copy-paste URLs into different dashboards, write complex scripts just to cover all the endpoints, and then spend hours correlating raw vulnerability lists with internal documentation.

With this MCP, you talk to your agent like a teammate. You tell it what service needs protecting; it handles the rest—from registering the base URL via `create_service` to running the full attack simulation in one conversational flow. You get actionable reports on exploitable flaws without lifting a finger.

Get full visibility with Equixly's API Security Audits

The manual process involves separate steps: 1. Listing endpoints in one tool, 2. Uploading specs to another, and 3. Manually triggering the test run in a third system. This scattershot approach always leaves gaps.

Now, your agent coordinates it all. It uses `upload_api_spec` to ingest the full scope and then executes `trigger_scan`, consolidating status tracking and vulnerability reporting into one single, conversational output.

Equixly: 10 Tools for API Security Audits

These tools let you manage target services, upload documentation, trigger autonomous pentests, and analyze detailed vulnerability reports using natural conversation.

#	TOOL	DESCRIPTION
01	`create_service`	Registers a brand new API service target by providing its readable name and live base URL for testing.
02	`delete_service`	Permanently removes an existing API service and all associated scan history from the system.
03	`get_scan_findings`	Downloads a detailed report of every exploitable vulnerability found during a pentest, including severity and how to fix it.
04	`get_scan`	Retrieves the overall summary of a specific scan, showing total requests made and a breakdown by flaw severity.
05	`get_service`	Fetches the current configuration details for a specific API service before you adjust its testing behavior.
06	`list_scans`	Lists all recorded pentest sessions for an API, providing status, timestamps, and how many vulnerabilities were found in each run.
07	`list_services`	Gets a list of every API service currently registered with the platform, including their base URL and endpoint count.
08	`list_api_specs`	Shows all API specifications that have been uploaded to a service, helping you track coverage for the autonomous hacker.
09	`trigger_scan`	Launches an immediate, autonomous penetration test against a registered service to check for common security flaws.
10	`upload_api_spec`	Adds full API documentation (like OpenAPI or Postman) to a service, maximizing the scope of potential vulnerability discovery.

See It in Action

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

U Trigger a security scan for API service 'Billing-v1'

Pentest initiated! I've triggered an autonomous attack session for 'Billing-v1' (ID: svc_123). The AI Hacker is now exploring and testing your API. I'll notify you when the results are ready.

U Show me the critical findings from the latest scan of service 'Payments'

Retrieving findings for 'Payments'... I found 2 critical vulnerabilities: 'Unauthenticated IDOR on /users/me' and 'BOLA on /orders/{id}'. Would you like the remediation guidance for these?

U List all registered API target services

I found 3 registered services: 'Billing-v1' (https://api.billing.com), 'Payments' (https://pay.example.com), and 'Auth-Service'. Each is configured for continuous security monitoring.

Frequently Asked Questions

01 **How do I start using Equixly MCP for basic testing?**

You must first register the API service using `create_service` with its base URL. Once that's done, you can use `list_services` to confirm it's ready for initial scans.

02 **Can I test an API without having OpenAPI documentation? Using Equixly MCP?**

Yes, but coverage will be limited. While you should always upload specs using `upload_api_spec`, the agent can still run tests based only on the service URL defined by `create_service`.

03 **What is the difference between getting scan data and finding flaws with Equixly MCP?**

The `get_scan` tool gives you the summary metrics—total requests, endpoints explored. The `get_scan_findings` tool drills down to give you specific details about every confirmed vulnerability.

04 **If I update my API, do I have to re-register it with Equixly MCP?**

No. You can use `get_service` to check the existing configuration and then simply run a new scan using `trigger_scan` against the established service.

05 **Which tool should I use if I want to see what endpoints are available?**

Start by running `list_services`. This will provide you with all registered API services and their corresponding unique IDs, which helps guide your next actions.

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 `"equixly": { "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

Equixly 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 Equixly. 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	Equixly MCP
Server ID	`019d7591-7d60-7026-8cbc-1208447b7e5e`
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/equixly.