MCP SERVER | NO CODE | CLOUD HOSTED

Salt Security MCP

Real-Time API Defense via Conversation

Salt Security gives your AI client real-time defense for your APIs. It lets you check API inventories, find hidden or 'shadow' endpoints, monitor live attacks, and automatically block malicious actors—all through conversation. Use it to audit security posture and manage governance rules without logging into a dashboard.

A+ Quality Score 100/100

api-security | threat-detection | behavioral-analytics | shadow-api | posture-management | real-time-remediation

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/salt-security

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.

Salt Security MCP

10 tools available
Cloud-hosted on Vinkius

Your agent connects directly to Salt Security, giving it eyes on your entire Application Programming Interface environment. You stop guessing about what APIs are running or if they're secure. Instead, you ask questions like, "What endpoints haven't been formally documented?" and get an immediate list of potential vulnerabilities or shadow APIs.

It monitors for active attacks as they happen, listing malicious events and even profiling the attackers involved. When a threat is identified, you can immediately trigger remediation commands to block that attacker at your WAF level. This capability means you don't have to switch between monitoring dashboards and incident response tools; everything flows through your AI client. By connecting this MCP via Vinkius, you give your agent access to an entire catalog of security tools, making API defense as simple as a chat prompt.

Core Capabilities

01 — Map API Inventory

The tool retrieves a complete list of all auto-discovered APIs, including hidden or 'shadow' endpoints in your network.

02 — Review Endpoint Details

You can get specific details about any single API endpoint to check for exposed sensitive data or structural issues.

03 — Analyze Live Attacks and Threat Actors

The system lists current malicious API attacks, helping you understand the attack patterns and profiling known threat actors.

04 — Remediate Threats Instantly

You issue a command to block an attacker immediately, passing instructions directly to your integrated WAFs.

05 — Audit Security Design Flaws

The MCP identifies vulnerabilities and design flaws before they ever hit the live production environment.

06 — Verify Governance Rules

You check which API governance rules are currently active and manage uploaded OpenAPI specifications.

One Click on Vinkius — From Prompt to Execution

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

01 Enable the Salt Security integration in your workspace.

02 Generate an API Token within the Salt Security console and paste it into the configuration fields provided by Vinkius.

03 Ask your AI client a direct question, like asking if there are known threat actors exploiting your APIs right now.

The bottom line is you get real-time visibility and active control over API security without leaving your chat interface.

Built For

This is for the Security Operations Center (SOC) analyst who gets tired of manually switching between monitoring dashboards and incident response tools. It's also for the Application Developer needing to validate secure API designs before deployment, and the Compliance Officer who needs proof that shadow APIs are accounted for.

Security Operations Analyst

You monitor active attacks or profile threat actors by simply asking your agent, getting immediate data on suspicious activity.

DevSecOps Engineer

You audit the API posture and check for design flaws using the MCP before releasing a new version to production.

API Architect / Product Owner

You list all discovered APIs, including any unknown or 'shadow' endpoints, ensuring nothing is exposed without proper governance.

What Changes When You Connect

01 Stop worrying about forgotten endpoints. Use `get_inventory` to automatically discover all APIs, including unknown or 'shadow' resources that could be exposed.

02 React instantly during an attack. Instead of manually creating firewall rules, just ask your agent to block the threat using `block_attacker` and pass the command straight to your WAFs.

03 Go beyond basic monitoring. Use `get_attackers` to profile known malicious actors so you understand their methods, not just the attacks itself.

04 Audit before deployment. Run `get_posture_vulnerabilities` to catch design flaws and weaknesses in APIs that haven't even reached a testing environment yet.

05 Ensure compliance easily. Use `list_oas_specs` or `upload_oas_spec` to manage your API documentation, making sure governance rules are always current.

Real-World Applications

Investigating a Breach

A SOC analyst detects unusual traffic. Instead of checking logs for hours, they ask the agent about recent attacks. The agent runs `get_attacks` and finds 12 malicious attempts targeting authentication modules, immediately directing the team to the point of failure.

Onboarding a New Service

A DevSecOps engineer finishes building an API but isn't sure if it has vulnerabilities. They run `get_posture_vulnerabilities` via the MCP, which flags several structural issues that must be fixed before deployment.

Discovery Audit

A Compliance Officer needs proof of full API coverage. They ask to list all APIs using `get_inventory`, finding four 'zombie' endpoints that were forgotten and require immediate documentation or removal.

Policy Update

The team updates their API structure. Instead of manually updating the security rules, they use `upload_oas_spec` to feed the new OpenAPI spec into Salt Security, instantly updating governance policies.

Patterns to Avoid

Assuming APIs are documented

✗ AVOID

A developer thinks because a team *says* they document everything, no shadow APIs exist and skips the audit step.

✓ INSTEAD

Run `get_inventory` first to see every live API endpoint. If you find unknown endpoints, use `get_endpoint` on those specific paths to check for sensitive data exposure.

Reacting only after an attack hits

✗ AVOID

The security team waits until a customer reports that their account was compromised before investigating the source of the breach.

✓ INSTEAD

Use `get_attacks` proactively to monitor for signs of business logic abuse. If attacks are found, use `block_attacker` immediately to contain the threat.

Treating security as a manual checklist

✗ AVOID

The architect manually checks compliance against dozens of governance policies one by one and gets bored halfway through.

✓ INSTEAD

Ask your agent to use `get_governance_policies` to list the active rules, then ask it to check if any specific endpoints fail those rules.

The Right Fit

Use this MCP if your primary problem is API visibility or real-time threat response. You need a single pane of glass that lets you audit inventory (`get_inventory`), spot weaknesses before launch (`get_posture_vulnerabilities`), and act immediately when something goes wrong (`block_attacker`). Don't use it if your issue is simply general network monitoring (use an existing SIEM tool) or managing internal user credentials (use a dedicated IAM tool). If you only need to read static API documentation, `list_oas_specs` works. But if you need to *enforce* policies and actively block threats in real-time conversationally, this is the right tool.

The headache of finding your own APIs.

Today, discovering every single API endpoint in a large microservice architecture feels like detective work. You have to log into multiple consoles—the network dashboard for traffic flow, the documentation portal for official specs, and sometimes you just don't know where to look. It’s tedious copying data between tabs just to build a picture of your true attack surface.

With this MCP, you ask your agent directly: "What APIs are running?" It runs `get_inventory` and gives you the full list. You instantly see endpoints that were never documented or approved—the 'shadow' resources that pose the biggest risk.

Salt Security MCP: Real-Time API Threat Defense

Manual incident response means switching between a monitoring system, an attacker profile database, and then finally logging into your WAF to manually type the block rule. This process takes minutes of high stress and potential human error.

Now you just ask the agent: "Block this threat."

The MCP runs `block_attacker` and passes the command directly to the integrated gateways. You move from detection to remediation in seconds, right inside your chat.

Salt Security with 10 Tools

These ten tools allow you to audit every aspect of your API environment, from mapping unknown endpoints to automatically blocking malicious threat actors.

#	TOOL	DESCRIPTION
01	`block_attacker`	Issues a command to block a specific attacker profile from accessing your APIs.
02	`get_attackers`	Lists profiles of known threat actors identified by Salt Security.
03	`get_attacks`	Retrieves a list of detected malicious API attack events.
04	`get_endpoint`	Gets detailed information for a specific, named API endpoint.
05	`get_governance_policies`	Lists all currently active governance rules governing your APIs.
06	`get_inventory`	Retrieves the complete, auto-discovered list of every API endpoint in your environment.
07	`get_posture_vulnerabilities`	Identifies and lists design flaws or vulnerabilities found during pre-production testing.
08	`get_system_health`	Checks the operational status of your traffic mirror ingestion service.
09	`list_oas_specs`	Lists all OpenAPI (OAS) specifications that you have uploaded for governance.
10	`upload_oas_spec`	Uploads a new OAS/Swagger specification to register it with your security policies.

See It in Action

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

U List all auto-discovered APIs including shadow and zombie APIs in our infrastructure.

I successfully queried your API inventory. There are currently 32 distinct APIs. Among these, exactly 4 are classified structurally as 'zombie' missing formal documentations.

U Are we facing any recent attacks aimed at business logic?

Reviewing your active threat ledger, you have exactly 12 malicious attack events attempting account takeovers targeting primary user authentication web modules currently.

U Block attacker 'ATT-992' immediately.

The remediation command was actively passed to your Salt Security console. The threat actor profile 'ATT-992' has been assigned an internal blockade rule via integrated gateways.

Frequently Asked Questions

01 **How do I find unapproved API endpoints using Salt Security?**

Use `get_inventory` to pull the entire list of discovered APIs. This tool automatically flags any endpoint that isn't formally documented or governed, helping you identify shadow resources.

02 **Does Salt Security MCP help with compliance reporting?**

Yes. You can use `get_governance_policies` to list active rules and then verify specific APIs against those policies using `get_endpoint`, ensuring your system meets compliance standards.

03 **What if I need to block an attacker right now? How do I use the Salt Security MCP?**

You simply prompt your agent with a command like "Block threat 'XYZ'", and it executes the `block_attacker` tool, passing the rule directly to your WAFs for immediate enforcement.

04 **Can I use Salt Security MCP to see what attacks are happening right now?**

Absolutely. Use the `get_attacks` tool to list all detected malicious API attack events, giving you a clear record of current threats and how they attempt account takeovers.

05 **Does this MCP cover pre-production vulnerabilities?**

Yes, before your code hits live, use `get_posture_vulnerabilities`. This tool retrieves identified design flaws that need fixing in development, preventing issues later on.

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 `"salt-security": { "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

Salt Security 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 Salt Security. 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	Salt Security MCP
Server ID	`019d7602-edb8-70c1-b172-ed11861985f0`
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/salt-security.