MCP SERVER | NO CODE | CLOUD HOSTED

FOSSA MCP

Audit your software supply chain, conversationally.

FOSSA License Compliance connects your open-source auditing tools directly to your AI client. It automates security vulnerability checks and license compliance reviews across entire software supply chains, letting you query project dependencies in natural language. Stop clicking through dashboards; start asking questions about where every piece of code comes from.

F Quality Score 3.6/100

open-source | license-compliance | vulnerability-scanning | dependency-management | software-audit

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/fossa-license-compliance

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.

FOSSA (License Compliance) MCP

6 tools available
Cloud-hosted on Vinkius

Manually tracking open-source licenses or hunting down a single vulnerable dependency is a massive time sink. This MCP lets your AI client bypass the FOSSA web interface entirely. Instead, you talk to it naturally and get precise audit data for your whole organization's codebase. You can list every project, pinpoint exactly which parent applications rely on a risky package, or check dozens of dependencies for vulnerabilities all in one go. When paired with Vinkius, this MCP becomes the central point for accessing enterprise-grade security intelligence from any compatible client. It takes deep, complex data—like dependency trees and revision metadata—and turns it into actionable answers you can use right away.

Core Capabilities

01 — List all projects

Retrieves a list of every project in your organization, supporting filtering by criteria like department or status.

02 — Analyze specific revisions

Gets detailed metadata for any version locator, allowing you to audit a project at a precise point in time.

03 — Map dependency trees

Deep-dives into the full dependency list of a revision, building an accurate software bill of materials (SBOM).

04 — Identify project impact areas

Determines which parent projects contain specific dependencies that are vulnerable or non-compliant.

05 — Scan for vulnerabilities

Checks multiple dependency locators against the FOSSA database in a single query to find security risks.

06 — View project history

Lists all available revisions for a given project, helping you track changes over time.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/fossa-license-compliance** — connect your AI agent in three steps.

01 Subscribe to this MCP and input your FOSSA API Token.

02 Direct your AI client to use the connected tools when prompted with an audit task.

03 Your agent executes the necessary commands, returning structured data on compliance or vulnerabilities.

The bottom line is you get a conversational interface for complex security audits, eliminating manual API calls and UI navigation.

Built For

This MCP is built for the security engineer who needs to prove compliance instantly or the architect tired of manually mapping dependency risks across microservices. If your job involves knowing exactly what code runs where, this tool is essential.

Security Engineer

Uses `check_vulnerabilities` to quickly identify where critical packages are used across the entire codebase, rather than just checking a single service.

Compliance Officer

Runs audits on specific project revisions and uses `get_parent_projects` to ensure every deployed application meets legal license standards.

DevOps Architect

Uses tools like `list_projects` and `get_revision_dependencies` to verify the complete dependency graph of an entire system before deployment.

What Changes When You Connect

01 Pinpoint risks instantly. Instead of manually checking one package, you can run `check_vulnerabilities` against multiple locators in a single query, giving immediate security coverage.

02 See the full scope. Use `get_parent_projects` to answer questions like, 'Which services are using this deprecated library?' and get an exhaustive list of every consumer project.

03 Audit specific moments in time. By checking revisions using `list_revisions` and `get_revision`, you can audit a project's compliance state exactly as it was last month, not just its current state.

04 Cover the entire codebase. Start by running `list_projects` to get an inventory of all potential targets, ensuring no service is missed during your security sweep.

05 Understand dependency depth. The `get_revision_dependencies` tool doesn't just list what's in a project; it maps out the full tree structure you need for compliance checks.

Real-World Applications

The sudden vulnerability alert

A security engineer gets an alert about a critical CVE affecting `npm+ssh2$0.6.1`. Instead of spending hours checking every service's source code, they ask their agent to run the dependency locators through `check_vulnerabilities` and immediately get confirmation on which projects need patching.

Debugging dependency sprawl

A developer can't figure out why a feature is breaking. They ask their agent to use `get_parent_projects` for the failing library, instantly revealing that three unrelated microservices are relying on the problematic code.

License review before merger

A legal team needs to know if a newly acquired codebase is compliant. They use `list_projects` to get the inventory, then run detailed audits on specific revisions using `get_revision`, ensuring no non-compliant licenses sneak into the merged product.

Patterns to Avoid

Only checking package.json

✗ AVOID

Assuming the top-level `package.json` file contains all necessary dependency data for a complete compliance audit.

✓ INSTEAD

You must use `get_revision_dependencies` to map the full, nested tree structure of the code. This captures every indirect and direct library used by the project.

Manually listing projects

✗ AVOID

Trying to keep a spreadsheet list of all 50+ services in your organization that need auditing.

✓ INSTEAD

Start with `list_projects`. This tool provides an up-to-date, filtered inventory of every project available for audit, ensuring you don't miss anything.

Using outdated version data

✗ AVOID

Running vulnerability checks against a local copy of code that hasn't been updated in months.

✓ INSTEAD

Always reference specific historical states using `list_revisions` and then fetching the metadata with `get_revision`. This ensures you are auditing the correct, verified version.

The Right Fit

Use this MCP if your core problem is determining the *scope* of a risk. You need to answer questions like: 'What does Project X depend on?' or 'Which 15 projects use this vulnerable library?'. It's for deep, systemic auditing where you must trace dependencies across multiple applications and time periods.

Don't use it if your only goal is to check the license of a single file or run a quick, isolated vulnerability scan. For that, simpler code analysis tools might suffice. But when the risk spans an entire organization—the 'blast radius' problem—you need the comprehensive view this MCP provides.

The manual process of checking compliance is always chasing ghosts.

Right now, to audit your supply chain, you open the FOSSA web interface. You have to manually navigate from project list to specific revisions, then click through dependency trees, and finally run a separate vulnerability scan for every single library of concern. This process takes hours just to gather the data.

With this MCP connected via Vinkius, you skip all that clicking. You simply ask your agent: 'Show me all projects using version 1.2 of Library Z.' The system compiles the full answer—project list, revision status, and dependency map—and presents it back to you in plain language.

Get a complete project picture with FOSSA License Compliance.

You no longer need to copy-paste package names into separate forms or tabs. You use `get_revision_dependencies` and `list_projects` together, telling your agent exactly what you need for the whole system at once.

The difference is that data moves from siloed dashboards into a cohesive conversation. Your audit results are immediate and actionable, allowing engineers to patch vulnerabilities in minutes instead of days.

FOSSA (License Compliance) with 6 Tools

These tools let you list projects, inspect historical revisions, map complex dependencies, and run vulnerability scans across your entire organization's codebase.

#	TOOL	DESCRIPTION
01	`check_vulnerabilities`	Checks a list of package locators to see if any are known security vulnerabilities.
02	`get_revision`	Fetches detailed metadata for a specific, named project revision.
03	`get_parent_projects`	Finds all parent projects that incorporate a given dependency.
04	`list_projects`	Lists every project in your organization, with filtering options to narrow the search.
05	`get_revision_dependencies`	Retrieves the full dependency list for a specific version of a project.
06	`list_revisions`	Lists all available revisions (versions) that exist for a given project.

See It in Action

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

U List all projects in my FOSSA organization named 'DocsExample'.

I've found the project: 'custom+27932/DocsExample'. Would you like to see its recent revisions or check for vulnerabilities?

U Check for vulnerabilities in npm+ssh2$0.6.1 and npm+coa$2.0.2.

I've analyzed those locators. 'npm+ssh2$0.6.1' has 2 critical vulnerabilities related to command injection. 'npm+coa$2.0.2' is currently clear of known vulnerabilities in the FOSSA database.

U Which projects are using the dependency npm+coa$2.0.2?

The dependency 'npm+coa$2.0.2' is found in 3 parent projects: 'Core-API', 'Frontend-Dashboard', and 'Legacy-Auth-Service'.

Frequently Asked Questions

01 **How does FOSSA License Compliance MCP help with dependency auditing?**

It allows your agent to analyze the full dependency tree using `get_revision_dependencies`, which builds a complete Software Bill of Materials (SBOM) without you having to manually map out every layer.

02 **Can I check vulnerabilities across multiple projects with FOSSA License Compliance MCP?**

Yes. You can first use `list_projects` to gather the targets, and then use `check_vulnerabilities` in a single query against those combined locators.

03 **What if I need data from an old version of a project?**

You can't rely on current data. Use `list_revisions` to find the historical versions, and then use `get_revision` to pull metadata for that specific point in time.

04 **Does FOSSA License Compliance MCP tell me which project owns a risky dependency?**

Absolutely. The `get_parent_projects` tool will search your entire organization and list every single parent application using the problematic package, solving the 'blast radius' problem.

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 `"fossa-license-compliance": { "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

FOSSA (License Compliance) 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 FOSSA (License Compliance). 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	FOSSA (License Compliance) MCP
Server ID	`019e389b-01eb-735d-a35b-c5192044387b`
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/fossa-license-compliance.