MCP SERVER | NO CODE | CLOUD HOSTED

HTML XSS Sanitizer MCP

Stop Malicious Code Before It Hits Your Database

HTML XSS Sanitizer: Stops malicious code from entering your database. This MCP takes raw HTML inputs—like user comments or blog content—and strips out dangerous scripts, event handlers, and unsafe tags before saving them. It provides a critical security layer that standard AI models can't handle.

F Quality Score 3.6/100

xss-protection | sanitization | web-security | input-validation | data-integrity | cybersecurity

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/html-xss-sanitizer

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.

HTML XSS Sanitizer MCP

1 tools available
Cloud-hosted on Vinkius

When you let an agent process public input, you risk data contamination from Cross-Site Scripting (XSS) attacks. Standard language models don't have the native rendering engine required to test for obfuscated or malicious code payloads. This MCP acts as a necessary security shield. You simply feed it any raw HTML payload, and the tool surgically cleans the content, eliminating dangerous tags like `<script>` and unsafe attributes like `onload`. It enforces a strict whitelist of safe elements, ensuring that only clean, harmless markup makes it into your database. By connecting this through Vinkius, you give your AI client the reliable protection it needs to handle any user-submitted content safely.

Core Capabilities

01 — Strip Dangerous Tags

The MCP removes known malicious tags like script blocks and iframes from raw HTML.

02 — Clean Malicious Attributes

It scrubs unsafe attributes, such as 'onload' or 'onerror', that attackers use to execute code in the browser.

03 — Enforce Safe Markup

The tool only allows specific, safe HTML tags you define, blocking everything else by default.

04 — Process Live Payloads

You pass it real-world inputs, such as user comments or forum posts, for immediate sanitization.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/html-xss-sanitizer** — connect your AI agent in three steps.

01 Send the raw HTML content (e.g., a blog comment) to the MCP.

02 The tool analyzes the payload against known XSS vectors and strips all dangerous elements and attributes.

03 Receive clean, safe HTML that is guaranteed not to execute malicious code in a browser.

The bottom line is you get database-safe HTML, removing manual checks for every single input source.

Built For

Developers and backend engineers who handle user inputs are the primary users. If your application accepts any form of rich text or user content—comments, profiles, forum posts—you need this MCP to prevent catastrophic security breaches.

Backend Engineer

You connect this MCP before calling database write functions to validate and strip all incoming HTML from the application layer.

Security Architect

You mandate its use in CI/CD pipelines or agent workflows to ensure that any content ingested by your system meets a zero-trust security standard.

Full Stack Developer

You integrate it into the front end or API layer, running sanitization checks instantly when a user submits a form containing rich text.

What Changes When You Connect

01 You stop Cross-Site Scripting (XSS) attacks dead in their tracks. Instead of worrying about obscure event handlers, you trust the output of the `sanitizeHtml_html` tool to clean every payload.

02 It moves security checks out of your code and into a specialized service. Your agents don't need complex regex or custom parsers; they just call this MCP function.

03 You enforce a strict whitelist approach, meaning only approved HTML tags survive the process. This is much safer than trying to blacklist every possible attack vector.

04 The tool handles both dangerous tags like `<script>` and malicious attributes like `onload` simultaneously. It's an all-in-one security layer for rich text input.

05 It gives you instant, reliable data integrity. You can run this MCP client-side or server-side to ensure no bad data ever enters your system.

Real-World Applications

A user submits a profile bio with embedded scripts

The developer asks their agent to process the raw text. The agent uses `sanitizeHtml_html` and strips out the hidden JavaScript, returning only clean text that can safely be saved to the database.

A content ingestion pipeline receives external blog posts

The system needs to save a third-party article. Before writing it, the agent runs `sanitizeHtml_html` on the whole body copy, guaranteeing that any embedded iframes or malicious scripts are gone.

A forum post contains multiple attempts at obfuscated code

The backend receives a user comment full of weird tags. The agent calls `sanitizeHtml_html`, which correctly identifies and removes the encoded vectors, letting you save usable content without risk.

An agent processes mixed HTML/text data streams

The goal is to extract clean text while preserving benign formatting. The agent uses `sanitizeHtml_html` first to strip the junk and then extracts the pure, safe content for display.

Patterns to Avoid

Using basic string replacement

✗ AVOID

You try to clean HTML by just searching for and replacing all instances of '<script>' with an empty string. This fails immediately when the attacker uses encoding or event handlers.

✓ INSTEAD

Use the `sanitizeHtml_html` tool. It is designed to understand complex, nested malicious payloads and strips them regardless of how they are encoded.

Relying solely on frontend validation

✗ AVOID

You assume that because JavaScript blocked the input in the browser, it's safe for the database. This is a critical failure; bad actors bypass client-side checks easily.

✓ INSTEAD

Always run this MCP as a mandatory server-side step before writing any user content to your database.

Using generic regex cleaning

✗ AVOID

You write complex regular expressions hoping to catch all XSS vectors. This is an endless, losing battle because attackers constantly find new ways to obfuscate code.

✓ INSTEAD

Let the specialized `sanitizeHtml_html` tool handle it. It uses established security libraries designed specifically for this vulnerability.

The Right Fit

Use this MCP if your application accepts any user-generated rich text—think comments, profiles, or forum posts. If you need to strip out malicious code from HTML before saving it anywhere, this is non-negotiable. Don't use it if your input is purely plain text; in that case, simple trimming will suffice. Crucially, don't use it for *validating* structure (e.g., ensuring a user only used H1 tags); use specialized schema validation tools for that. This MCP's job is strictly security: cleaning out the dangerous stuff so your database stays clean.

The Problem of User-Generated Content

Every day, developers face the same headache: someone posts a comment or updates their profile with HTML they shouldn't have. These payloads aren't just text; they can contain hidden `<script>` tags or malicious `onload` attributes that execute code in a visitor's browser when viewed. The manual process involves writing complex checks and constantly updating regex patterns to catch the latest obfuscation techniques.

With this MCP, you simply run the content through the `sanitizeHtml_html` tool before storage. You get back guaranteed safe markup—the bad stuff is gone. This moves security from a never-ending coding chore into a single, reliable function call.

Get Clean HTML with sanitizeHtml_html

Manual sanitization means checking for every known tag, remembering to strip event handlers like `onerror`, and figuring out how to decode Base64 payloads. You're building a security system that requires constant maintenance.

Now, you pass the raw content through this MCP tool. It handles all those complex checks automatically in one step. Your focus shifts from fighting XSS vectors to building features.

HTML XSS Sanitizer: 1 Tool

This single tool allows you to take raw, potentially unsafe HTML input and return clean markup that is safe for display in a browser or storage in a database.

#	TOOL	DESCRIPTION
01	`sanitizeHtml_html`	Pass raw HTML content to strip dangerous scripts and attributes, returning clean markup safe for database storage.

See It in Action

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

U Sanitize this HTML input: `<p>Hello</p><script>alert('hack');</script>`

✅ **Sanitized HTML:** `<p>Hello</p>`

U Clean this blog post content to ensure no malicious iframes are present.

✅ **Cleaned:** Stripped 2 `iframe` tags and 1 `onload` event successfully.

U Check if this user comment contains any XSS vectors before we save it.

✅ **Sanitized Payload:** Returned safe string.

Frequently Asked Questions

01 **Does HTML XSS Sanitizer handle plain text inputs?**

No, this MCP is designed specifically for cleaning existing HTML payloads. If your input is purely plain text, you don't need to use the `sanitizeHtml_html` tool.

02 **Is using sanitizeHtml_html fast enough for high-traffic sites?**

Yes. It's built as a dedicated security service designed for speed and reliability, making it suitable for high-volume data ingestion pipelines without performance bottlenecks.

03 **What if I want to allow *some* HTML tags but block others?**

The tool uses whitelisting logic. While the core function strips dangerous items by default, its underlying mechanisms enforce strict rules that let you define what is safe and what gets stripped.

04 **Is sanitizeHtml_html effective against modern XSS vectors?**

Yes. It uses enterprise-grade sanitization techniques designed to defeat obfuscated payloads, including Base64 encoding and obscure event handlers that basic filters miss.

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 `"html-xss-sanitizer": { "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

HTML XSS Sanitizer 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 HTML XSS Sanitizer. 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	HTML XSS Sanitizer MCP
Server ID	`019e38a9-951e-712b-8dbc-6de345e7f8ad`
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/html-xss-sanitizer.