CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Flowise CVSS 10.0 RCE: AI Agent Builders Under Attack

CVE-2025-59528 Exploited in the Wild Across 12,000–15,000 Exposed Instances

Unofficial AI-assisted Research

2026-04-09

Key Takeaways

- CVE-2025-59528 is a maximum-severity (CVSS 10.0) code injection vulnerability in Flowise, an open-source AI agent builder used by Fortune 500 companies including Deloitte, Accenture, and AWS [3], with over 43,000 GitHub stars [7] and between 12,000 and 15,000 internet-exposed instances [2][5][9].
- The vulnerability resides in the CustomMCP node, which processes user-supplied configuration for connecting to external Model Context Protocol (MCP) servers; the platform transmitted this input directly to JavaScript's `Function()` constructor without validation, allowing unauthenticated attackers to execute arbitrary code on the host server via a single POST request [1][4].
- Active exploitation was detected by VulnCheck in April 2026, more than six months after the patch was released in Flowise version 3.0.6 (September 2025), a window that underscores the persistent lag between patch availability and deployment across AI developer tooling [2][5].
- Successful exploitation grants an attacker full Node.js runtime privileges on the Flowise server, including access to the `child_process` module for operating system command execution and the `fs` module for unrestricted file system access, enabling complete host compromise and data exfiltration [1][4].
- CVE-2025-59528 is the third Flowise vulnerability exploited in the wild within a twelve-month period, following CVE-2025-26319 (CVSS 8.9, arbitrary file upload) [6] and CVE-2025-8943 (CVSS 9.8, unauthenticated OS command execution) [12], establishing Flowise as a sustained target of opportunistic exploitation campaigns [2][5].
- Organizations running Flowise in any capacity — production AI workflows, internal developer environments, or sandbox deployments — should treat this as an emergency upgrade requiring immediate action; the remediation is a version upgrade to 3.1.1 or, at minimum, 3.0.6 [1][2].

Background

Flowise is an open-source, low-code platform designed to allow developers and business teams to visually construct AI workflows, LLM orchestration pipelines, and multi-agent systems through a drag-and-drop interface. Built on top of LangChain, the platform abstracts the complexity of chaining language models, retrieval-augmented generation (RAG) pipelines, vector databases, and external tool integrations into a node-based visual editor [3][7]. Its breadth of integrations — over one hundred LLMs, vector databases, and external APIs — combined with a no-code interface has driven rapid adoption: the project has accumulated over 43,000 GitHub stars and is actively deployed in enterprise AI workflows at major professional services firms and technology companies [3].

The platform's popularity in AI development environments means that a compromised Flowise instance is often not an isolated endpoint but a gateway into broader AI infrastructure. Flowise instances commonly hold API keys for commercial LLM providers such as OpenAI, Anthropic, and Google, database credentials for connected vector stores, and configuration for internal data sources feeding RAG pipelines. A full compromise of a Flowise host therefore carries the potential not only for direct data exfiltration from the server itself but for lateral movement into connected AI infrastructure and the cloud services underlying it.

The vulnerability, designated CVE-2025-59528, was discovered by security researcher Kim SooHyun and classified under CWE-94 (Improper Control of Generation of Code). It was patched by the Flowise development team in version 3.0.6, released to GitHub in September 2025 [4][5]. Despite the availability of a fix, exploitation was first detected by VulnCheck in early April 2026, with attack activity attributed to a single Starlink IP address conducting opportunistic reconnaissance and exploitation against exposed instances [2]. Between 12,000 and 15,000 Flowise deployments are visible on public internet scanning infrastructure, representing a substantial pool of potential targets for any actor capable of adapting the publicly documented proof-of-concept technique [2][5][9].

The timing of active exploitation — more than six months after the patch release — is consistent with a pattern observed across AI developer tooling: open-source platforms deployed in internal development environments are often not subject to the same rigorous patch management processes applied to production-facing infrastructure. Teams that provision Flowise as an internal tool for experimentation or rapid prototyping may omit it from vulnerability management programs, resulting in deployments that accumulate unpatched critical vulnerabilities over extended periods.

Security Analysis

Vulnerability Mechanics

The vulnerable code path originates at the REST API endpoint `/api/v1/node-load-method/customMCP`, which accepts unauthenticated POST requests containing an `mcpServerConfig` parameter [1][4]. This endpoint is part of Flowise's CustomMCP node, a component that allows users to configure connections to external Model Context Protocol servers by specifying server connection parameters in a JavaScript object format. The MCP protocol, designed to standardize tool integration for LLM agents, requires flexible server configuration that the Flowise team elected to accept as evaluated JavaScript rather than as a strictly typed data structure.

The flaw lies in how this input traverses the application's processing chain. The request passes through the route handler, controller, and service layers without undergoing validation or sanitization at any stage. At the terminal point of this chain, Flowise invoked JavaScript's `Function()` constructor to evaluate the user-supplied `mcpServerConfig` string as executable code. The `Function()` constructor is semantically equivalent to `eval()` in its willingness to execute arbitrary JavaScript; unlike `eval()`, it does not share the caller's lexical scope, but it runs within the same Node.js runtime environment and inherits access to all modules available to that runtime [4]. An attacker can exploit this to load the `child_process` module and invoke shell commands, or to load the `fs` module and read, write, or delete files on the host system. Because the endpoint required no authentication, exploitation required only network reachability to the Flowise instance.

The fix implemented in version 3.0.6 replaced the `Function()` constructor call with `JSON5.parse()`, a strict data-format parser that processes JSON-like structures without executing code. This single targeted change eliminated the documented injection vector by enforcing that `mcpServerConfig` input is treated as data rather than as an instruction [4][10].

The MCP Protocol as an Emerging Attack Surface

CVE-2025-59528 is technically rooted in unsafe JavaScript evaluation, but its context within the MCP protocol integration deserves independent scrutiny. The Model Context Protocol has seen significant adoption across AI agent platforms as a common integration layer for connecting LLM agents to external tools and data sources. Flowise's CustomMCP node represents a direct implementation of this

protocol's server-side configuration model, and the vulnerability demonstrates that MCP's inherent flexibility — designed to accommodate a wide variety of external tool configurations — can create pressure on developers to accept overly permissive input formats to support valid use cases.

This dynamic is not unique to Flowise. MCP implementations across the AI agent ecosystem invite similar risks wherever server configuration or tool definitions are accepted in executable or semi-executable formats rather than in strictly typed schemas. CSA research has examined related MCP protocol risks, including supply chain vulnerabilities in Git-hosted MCP servers, establishing a broader pattern: as MCP becomes infrastructure for AI agent ecosystems, the security properties of every MCP node and server configuration path become a meaningful part of an organization's AI attack surface.

Exploitation Pattern and Scale

VulnCheck's detection of active exploitation in April 2026 documented a pattern consistent with opportunistic scanning: a single source IP conducting structured reconnaissance against publicly exposed Flowise instances, likely probing for unpatched deployments at scale before transitioning to targeted exploitation [2][5]. The six-month gap between the patch and detected exploitation is consistent with opportunistic actors who monitor vulnerability disclosures and systematically work through populations of unpatched targets over extended campaigns, though the specific actor motivation has not been independently confirmed.

The exposed instance population presents a compounding risk. Between 12,000 and 15,000 Flowise instances appear in public internet scans [2][5][9]. Even a conservative estimate of successful exploitation across a small fraction of this population implies hundreds of compromised AI workflow environments, each potentially holding API keys, database credentials, and access to connected cloud and data infrastructure. The breadth of the Flowise deployment base — spanning startups building AI products through large enterprises running internal knowledge retrieval systems — means that the downstream impact of a successful exploitation campaign is difficult to bound without visibility into individual deployment configurations.

The vulnerability carries particular weight in the context of Flowise's vulnerability history. CVE-2025-26319, an arbitrary file upload vulnerability (CVSS 8.9) in the `/api/v1/attachments` endpoint introduced in Flowise v2.2.6, saw in-the-wild exploitation following its public disclosure [6]. CVE-2025-8943, a missing-authentication flaw enabling remote OS command execution (CVSS 9.8), similarly attracted active exploitation [12]. Three exploited critical vulnerabilities within twelve months is a pattern that warrants organizational reassessment of Flowise's suitability for environments that handle sensitive data or maintain credentials for critical infrastructure, regardless of whether the most recent patch has been applied.

Risk Profile: AI Infrastructure Targeting

The targeting of AI agent builders represents a logical extension of the software supply chain attack model. Flowise instances occupy a privileged position in AI development pipelines: they hold credentials for upstream AI services, they process organizational knowledge through RAG pipelines, and they serve as the orchestration layer for agent workflows that may themselves have elevated permissions in connected systems. An attacker who compromises a Flowise deployment does not merely gain access to a development tool; they gain a foothold at the intersection of an organization's AI capabilities and the data and cloud infrastructure those capabilities depend on.

This attack vector provides access to a distinctive class of artifacts — LLM API keys, vector database credentials, retrieval corpus contents, and agent system prompts — that are specifically concentrated in AI agent builder environments and are not typically present in conventional application deployments. An organization's Flowise instance may hold credentials that enable an attacker to query proprietary documents through a RAG pipeline, run expensive inference at organizational expense, or inject data into training and retrieval pipelines to influence AI outputs in future sessions.

Recommendations

Immediate Actions

Organizations running Flowise in any environment should treat CVE-2025-59528 as a P0 incident requiring emergency response. The remediation path is unambiguous: upgrade to Flowise version 3.1.1 (or at minimum 3.0.6) immediately [1][2]. Versions up to and including 3.0.5 are confirmed vulnerable [10]. If an immediate upgrade is not possible for operational reasons, take the following containment actions without delay: remove public network exposure of the Flowise instance, enforce network-level access controls restricting the API to trusted internal IP ranges, and audit access logs for POST requests to `/api/v1/node-load-method/customMCP` dating back to September 2025 to identify potential prior compromise.

Any Flowise instance that was internet-accessible and running a vulnerable version should be treated as potentially compromised regardless of observed exploitation indicators. Rotate all credentials held within the Flowise configuration — LLM API keys, vector database credentials, connected cloud service credentials — and revoke any OAuth tokens or service account permissions granted to the Flowise deployment. Because a successful exploitation delivers full Node.js runtime access, forensic analysis

should extend beyond the Flowise application to the underlying host, reviewing process execution history, network connection logs, and file system modifications for indicators of post-compromise activity.

Short-Term Mitigations

Following emergency remediation, organizations should establish a structured baseline for Flowise and similar AI developer tooling within their vulnerability management programs. AI agent builders are production infrastructure in many organizations, but in practice they are often managed as development tools exempt from production-grade security controls. Flowise instances should be enrolled in asset inventory systems, subject to authenticated scanning, and included in the organization's patch management process with SLA targets aligned to CVSS severity. Given Flowise's demonstrated vulnerability cadence, critical patches should be applied within 24 to 72 hours of release.

Network architecture for Flowise deployments should enforce the principle of least-access exposure. Flowise instances should not be directly internet-accessible unless that exposure is operationally required and protected by authentication controls, web application firewall rules, and rate limiting on the API surface. The `/api/v1/node-load-method/` route family, which has been the entry point for this vulnerability, should be treated as particularly sensitive and monitored with anomaly detection rules for unexpected payloads or unusual source IPs. Where Flowise is used exclusively for internal AI workflows, deployment behind a VPN or private network boundary eliminates the opportunistic exploitation vector entirely.

Credential hygiene for AI infrastructure credentials deserves particular attention. API keys, database connection strings, and cloud service credentials stored in Flowise configuration should be scoped to the minimum permissions necessary for the platform's function, rotated on a regular schedule, and monitored for anomalous usage through the audit logging mechanisms of the upstream services they access. This practice limits the blast radius of a Flowise compromise even when patch management has lapsed.

Strategic Considerations

CVE-2025-59528 illustrates a structural challenge in the AI tooling ecosystem: the rapid adoption of open-source AI platforms has outpaced the security maturity of those platforms and the organizations deploying them. The three-CVE exploitation history of Flowise within twelve months reflects not an exceptional failure but a characteristic pattern of open-source AI tooling developed by small teams

under significant growth pressure, where security engineering resources compete with feature development demands and where the speed of capability expansion creates a persistent surface area that vulnerability management programs struggle to keep pace with.

Organizations procuring or building on open-source AI agent platforms should evaluate the security track record and security investment posture of those platforms as a first-class procurement criterion. Platforms that have demonstrated repeated critical vulnerability patterns, that lack dedicated security engineering resources, or that have not adopted software composition analysis and regular security auditing in their development pipeline should be evaluated against enterprise-grade alternatives or isolated to environments where the blast radius of a compromise is bounded by compensating controls.

The MCP protocol integration vector highlighted by this vulnerability warrants specific attention in security architecture reviews for organizations deploying agentic AI systems. Any system that accepts externally supplied MCP server configurations — whether Flowise or another platform — should be evaluated for the security properties of its configuration parsing logic, with particular scrutiny of any path where user-supplied content reaches a code execution context.

CSA Resource Alignment

CVE-2025-59528 maps directly to multiple threat categories identified in the CSA MAESTRO framework for agentic AI threat modeling [8][11]. Within MAESTRO's seven-layer architecture, the Flowise vulnerability operates at the integration and orchestration layer, where external tool configurations are ingested and processed by the AI agent platform. MAESTRO explicitly catalogs supply chain risks in AI agent ecosystems — including vulnerabilities in the ML libraries and integration components on which agents depend — and this CVE is a concrete realization of that threat category. MAESTRO's emphasis on provenance tracking for components that an AI agent interacts with applies directly: the CustomMCP node, as an integration point for external MCP servers, represents a trust boundary that should have been guarded by strict input validation.

The CSA AI Organizational Responsibilities framework is relevant to the remediation context. Organizations that deploy AI agent builders such as Flowise inherit operational responsibilities for the security of those deployments, including the obligation to maintain patch currency and to include AI infrastructure in vulnerability management programs. The six-month exploitation lag in CVE-2025-59528 reflects a failure of this organizational responsibility model across at least a portion of the Flowise deployment base — particularly for the 12,000–15,000 instances that remained internet-exposed after the patch was available [2][9].

The CSA Cloud Controls Matrix (CCM) provides applicable controls through its Vulnerability and Patch Management domain, which requires organizations to establish processes for the timely identification and remediation of software vulnerabilities in deployed components. The AICM extends these requirements into AI-specific tooling, recognizing that AI development platforms and agent builders constitute critical cloud-hosted or self-hosted infrastructure for which the same controls apply. Organizations should map their Flowise and broader AI tooling patch management against CCM's vulnerability management control family and identify gaps where AI tooling has been exempted from controls applied to conventional software.

The CSA STAR program provides a framework for AI service providers and AI infrastructure operators to demonstrate security assurance to customers and downstream organizations. As AI agent builders such as Flowise are increasingly deployed as shared internal platforms serving multiple teams or as embedded components in SaaS AI products, the security posture of those platforms becomes a supply chain risk for all downstream users. STAR certification or equivalent third-party security assessment for AI agent infrastructure components would provide the visibility necessary for organizations to make informed risk-based decisions about which platforms to trust with sensitive credentials and data.

References

[1] SonicWall. "FlowiseAI Flowise RCE via CustomMCP Node CVE-2025-59528." SonicWall Blog, April 2026.

[2] Bleeping Computer. "Max severity Flowise RCE vulnerability now exploited in attacks." Bleeping Computer, April 7, 2026.

[3] FlowiseAI. "Flowise - Build AI Agents, Visually." FlowiseAI, accessed April 2026.

[4] Vulert. "Flowise AI CVE-2025-59528 RCE Exploitation." Vulert Blog, April 2026.

[5] The Hacker News. "Flowise AI Agent Builder Under Active CVSS 10.0 RCE Exploitation; 12,000+ Instances Exposed." The Hacker News, April 7, 2026.

[6] GitHub Advisory Database. "FlowiseAI Flowise arbitrary file upload vulnerability · CVE-2025-26319." GitHub, 2025.

[7] GitHub. "FlowiseAI/Flowise: Build AI Agents, Visually." GitHub, accessed April 2026.

[8] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 6, 2025.

[9] GBHackers. "Attackers Exploit Flowise Injection Vulnerability as 15,000+ Instances Remain Exposed." GBHackers, April 2026.

[10] GitLab Advisory. "CVE-2025-59528: Flowise has Remote Code Execution vulnerability." GitLab, 2025.

[11] Cloud Security Alliance. "Applying MAESTRO to Real-World Agentic AI Threat Models." CSA Blog, February 11, 2026.

[12] GitHub Advisory Database. "Flowise OS Command Remote Code Execution via Custom MCPs · CVE-2025-8943." GitHub, 2025.

