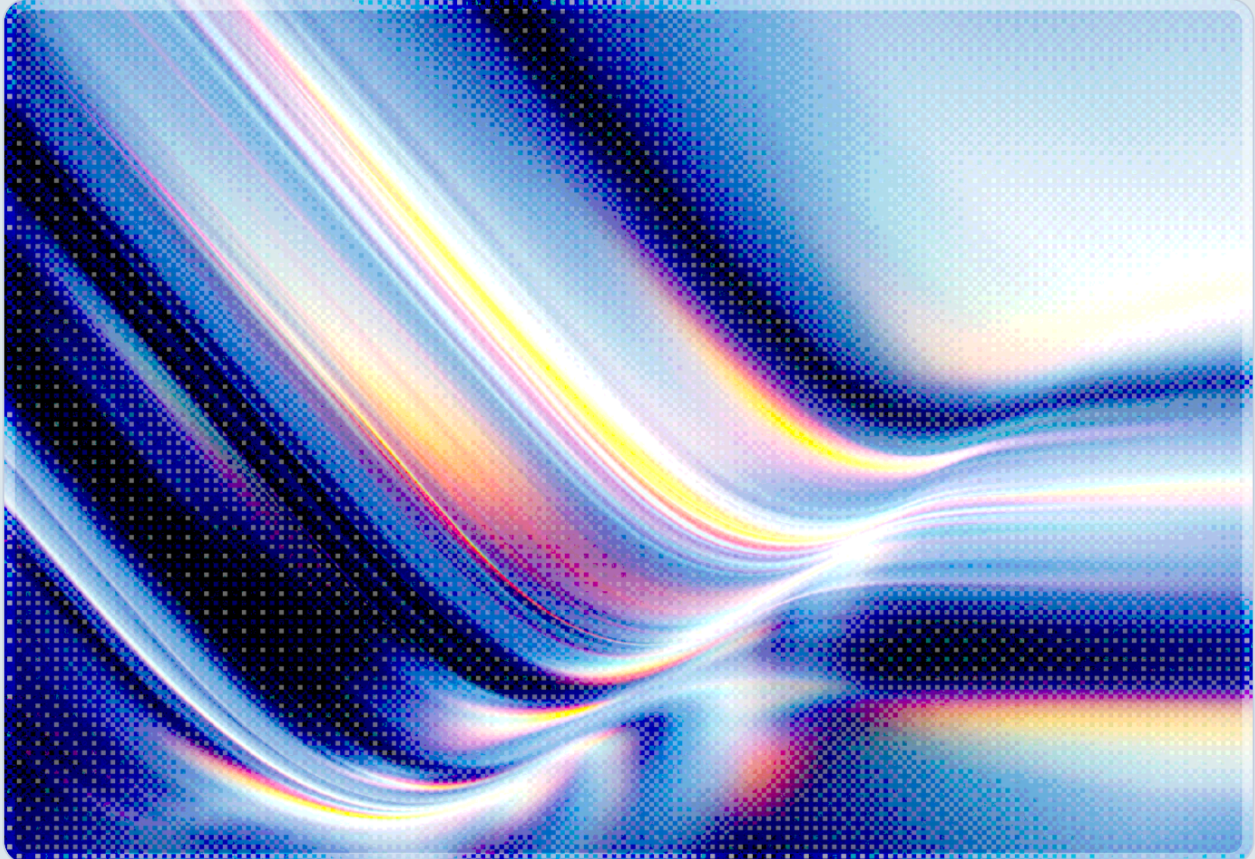


GitLab AI Gateway Sandbox Escape: Template Injection RCE

CVE-2026-90970 and Template-Injection Risk in Agent
Platforms

2026-10-06

 AI-assisted Rapid Research



CSAI

cloud
CSA security
alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

GitLab disclosed CVE-2026-90970 on October 2, 2026, a critical flaw (CVSS 3.1 score 9.9) in the self-hosted GitLab AI Gateway. An authenticated user with Duo Agent Platform access can escape the prompt template sandbox through a crafted flow configuration and run arbitrary commands on the gateway [1][2]. The weakness is classified as CWE-1336, improper neutralization of special elements used in a template engine [2][6]. An earlier gateway flaw, CVE-2026-1868, also involved template expansion of Duo Agent Platform flow definitions and also scored 9.9 [3]. The similarity raises the possibility of a recurring defect class, though the sources reviewed do not establish a shared root cause.

Only self-hosted gateways are affected. GitLab.com, GitLab Dedicated, and GitLab-hosted gateways are reported as already protected, and the vendor advisory documents no workaround, so patching to 19.2.4, 19.3.2, or 19.4.1 is the remediation [1][4]. Sources report no evidence of active exploitation at the time of writing, and the reporting we reviewed does not describe a method for detecting prior compromise [1][4].

The broader lesson is that user-authored agent definitions, such as flows, workflows, and prompt templates, are executable inputs. Platforms that render them server-side face a risk profile similar to that of server-side template injection.

Background

GitLab's AI Gateway is the service that sits between GitLab and the large language models that power its AI features, including the Duo Agent Platform. Organizations that run GitLab on their own infrastructure can also run the gateway themselves, which gives them control over where gateway traffic is routed. The Duo Agent Platform lets users define custom flows, which are configuration documents that describe multi-step agent behavior and include prompt templates that the platform renders before sending content to a model or tool [1][4].

On October 2, 2026, GitLab published fixed releases for a vulnerability in this component. GitLab states that it remediated an issue that could have allowed an authenticated user with Duo Agent Platform access to "escape the prompt template sandbox via a specially crafted flow configuration, leading to arbitrary command execution on the AI Gateway" [4]. The CVE record scores the issue at CVSS 3.1 9.9

with the vector AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H. That vector describes a network-reachable, low-complexity attack that requires only low privileges, needs no user interaction, and changes scope, meaning the impact extends beyond the vulnerable component itself [2]. The finding was credited to the HackerOne researcher "invisiblemeerkat" [1][4][5].

The affected and fixed versions follow GitLab's three concurrent release lines. Administrators on any release from 18.1.6 up to the first fixed release in their line should treat the gateway as vulnerable [1][2][4].

Release line	Affected versions	Fixed version
18.1.6 through 19.2	18.1.6 through 19.2.3	19.2.4
19.3	19.3.0 through 19.3.1	19.3.2
19.4	19.4.0	19.4.1

The gateway has been here before. CVE-2026-1868, patched earlier in 2026, was described as a template injection flaw in the Duo Workflow Service component of the AI Gateway, in which insecure template expansion of crafted Duo Agent Platform flow definitions allowed code execution. It also carried a CVSS score of 9.9 and was fixed in 18.6.2, 18.7.1, and 18.8.1 [3]. Public reporting on the new CVE describes it as a sandbox escape in the same general area, and it is reasonable to read the two together. GitLab has not, in the sources reviewed, published a root-cause comparison between them, and the source we consulted for the earlier CVE did not show its CWE classification.

Security Analysis

Why template rendering is an RCE surface

A prompt template is a small program. It mixes literal text with expressions that the template engine evaluates against a context, and many template engines can reach into the host language's object model unless they are constrained. Platforms commonly respond by running templates in a sandboxed environment that restricts which attributes, methods, and globals an expression can reach. CWE-1336 describes the failure mode: special elements in attacker-influenced input are not neutralized, so the engine interprets them as template syntax instead of data [6]. A sandbox is only as strong as its allow-list and the engine's handling of edge cases, and the public history of sandboxed template engines includes repeated escapes.

The two terms used in this note relate in a specific way. The template injection is the entry point, because attacker-controlled content is evaluated as template syntax. The sandbox failure is what turns that injection into code execution on the host. The public descriptions of CVE-2026-90970 do not name the template engine or disclose the exploit technique, and we have not independently reproduced the issue. What the sources do establish is the shape of the trust boundary: the attacker supplies a flow configuration, the gateway renders its prompt template, and the sandbox that is meant to contain that rendering fails [1][2][4]. The rest of this note therefore treats the specific technique as unknown and focuses on the structural conditions that made the outcome possible.

What changes in agent platforms

Template injection is an old vulnerability class, but agent platforms change who writes templates and where they run. In a conventional web application, templates are written by developers and reviewed before deployment, and user input is only supposed to flow into them as data. In an agent platform, the product's value comes from letting users author the agent, so flows, workflows, system prompts, and tool definitions are all user-controlled documents. When the server renders such documents, the "template author" and the "attacker" can be the same low-privilege account.

This helps explain the low privilege requirement in the CVSS vector [2]. Any user who can create or edit a custom flow is a potential attacker, and in typical self-hosted deployments we would expect that population to be considerably larger than the set of administrators. The inference we draw is that the effective threat model for the gateway includes every developer account with Duo Agent Platform access, not only external adversaries.

Blast radius of a compromised gateway

The scope-changed flag in the CVSS vector indicates that impact extends beyond the vulnerable component [2]. An AI gateway is a concentration point. It typically holds credentials for upstream model providers, handles prompts and completions that may contain source code and secrets, and has network paths to both the GitLab instance and external model endpoints. Code execution on such a host could plausibly expose provider API keys, read or alter model traffic, and offer a foothold for lateral movement. These consequences are our inference from the gateway's architectural role, not reported outcomes of this vulnerability, and exploitation has not been reported [1][4].

The same pattern appeared in CSA's analysis of the LiteLLM AI gateway, where a command injection flaw in MCP test endpoints (CVE-2026-42271) combined with an authentication bypass (CVE-2026-48710) produced a critical unauthenticated RCE chain against a component that brokers access to many

model providers [7]. Different vendors and different bug types still led to the same position in the architecture: the gateway is a high-value, high-privilege node that is often deployed with less scrutiny than the applications it serves.

A related class: untrusted agent inputs treated as trusted

CVE-2026-90970 is a server-side injection into the platform, which is distinct from prompt injection against a model. Prompt injection manipulates what the model decides to do, whereas here the flaw operates before or beside the model, in the rendering layer that prepares model inputs. CSA's work on Agent Data Injection makes a related argument, that defenses focused on the model's reasoning can miss attacks on the metadata and structure that agents and their platforms treat as trusted [8]. Template rendering is one such trusted-structure layer. Defenders who invest only in prompt-level filtering would likely not have been protected against this class of flaw, because the vulnerable step occurs outside model reasoning.

Patterns across the two GitLab disclosures

Taken together, the two disclosures suggest several conclusions, offered as inferences rather than established facts. The sources do not say how GitLab's sandbox is isolated. If it runs in the same process and privilege domain as the gateway, it should be treated as a mitigation that can fail, not as a security boundary. A critical template flaw recurring in the same product within the same year would also suggest that patching individual escapes may not close the class, and that architectural isolation (a separate process, container, or user with minimal privileges) is the more durable control. Finally, because GitLab states that hosted gateways were remediated before disclosure while self-hosted customers must act themselves, the exposure window for self-managed deployments largely depends on operator patch cadence [1][4].

Recommendations

Immediate Actions

Organizations running a self-hosted GitLab AI Gateway should first inventory their deployments and identify the gateway version, since the gateway is versioned alongside GitLab release lines and may be deployed separately from the main application. Any instance in the affected ranges should be upgraded to 19.2.4, 19.3.2, or 19.4.1 or later, as the vendor advisory documents no workaround [1][4]. Teams that

cannot patch immediately should consider restricting network access to the gateway and limiting Duo Agent Platform access to a small set of trusted accounts, as OpenCVE also suggests alongside auditing existing flows [2]. These steps reduce but do not remove exposure.

Because the reporting reviewed does not describe a way to detect prior exploitation [1], teams should also review gateway host and container logs for unexpected child processes, outbound connections, and file changes dating from before the patch. Organizations that find indications of compromise, or that cannot rule it out for a long-exposed instance, should rotate model-provider API keys and any other secrets accessible to the gateway process. Existing custom flow definitions should be audited for unfamiliar authors, recent edits, and unusual template expressions.

Short-Term Mitigations

Over the following weeks, run the gateway as an unprivileged user in a hardened container with a read-only root filesystem, dropped Linux capabilities, and no more secrets than it needs. Apply egress filtering so that the host can reach only the GitLab instance and approved model endpoints, which limits the usefulness of code execution to an attacker. Narrow who may create or modify custom flows, and require review for changes to flow configurations in the same way organizations review CI pipeline definitions. Forward gateway logs to central monitoring and alert on process execution that does not match the service's normal behavior.

Strategic Considerations

Longer term, organizations should add agent-platform authoring surfaces to their threat models and application security reviews. Any feature that lets users define prompts, templates, workflows, or tool descriptions that the platform later interprets should be assessed as an interpreter, with the same scrutiny given to expression languages and plugin systems. When selecting or building agent platforms, ask vendors how templates are rendered, whether rendering occurs in an isolated process, how sandbox escapes have been handled historically, and how quickly self-hosted customers are notified of fixes. Where feasible, prefer designs that treat templates as logic-less data substitution and do not evaluate arbitrary expressions. Finally, include AI gateways in patch-management SLAs comparable to those for internet-facing infrastructure, because compromise of the gateway can expose provider credentials and model traffic, as described under blast radius above.

CSA Resource Alignment

CSA's research note on the LiteLLM AI gateway is the closest prior work on this subject [7]. It documents an actively exploited chain against a different gateway and reinforces the point made here, that gateways concentrate credentials and trust. The mitigation guidance on isolation, secret scoping, and timely patching in that note applies directly to GitLab self-hosted deployments.

CSA's Agent Data Injection research note describes attacks that corrupt the trusted metadata agents rely on, bypassing prompt-injection defenses [8]. The GitLab flaw lies in a different layer, but the defensive lesson is shared: platform-level structures that agents and gateways trust, including templates and flow configurations, need validation and isolation independent of model-level safeguards.

For threat modeling, CSA's MAESTRO framework provides a layered approach to agentic systems, and the gateway and orchestration layers it identifies are where this class of flaw sits [9]. For control mapping, the AI Controls Matrix v1.1 includes Application and Interface Security (AIS) and Threat and Vulnerability Management (TVM) domains that are relevant to secure input handling, sandboxing, and patch timeliness for AI platform components [10].

References

- [1] The Hacker News. "[GitLab Patches Critical Self-Hosted AI Gateway Flaw](#)." The Hacker News, October 2026.
- [2] OpenCVE. "[CVE-2026-90970](#)." OpenCVE, October 2, 2026.
- [3] Tenable. "[CVE-2026-1868](#)." Tenable CVE Database, accessed October 6, 2026.
- [4] GitLab. "[GitLab AI Gateway 19.4.1 Released](#)." GitLab Documentation, October 2, 2026.
- [5] Security Affairs. "[CVE-2026-90970: Critical GitLab AI Gateway Flaw Fixed](#)." Security Affairs, October 3, 2026.
- [6] MITRE. "[CWE-1336: Improper Neutralization of Special Elements Used in a Template Engine](#)." MITRE, accessed October 2026.
- [7] Cloud Security Alliance. "[LiteLLM AI Gateway: Active Exploitation via MCP Injection](#)." CSA AI Safety Initiative, 2026.
- [8] Cloud Security Alliance. "[Agent Data Injection: A New Attack Class Beyond Prompt Injection](#)." CSA AI Safety Initiative, July 16, 2026.
- [9] Cloud Security Alliance. "[Agentic AI Threat Modeling Framework: MAESTRO](#)." CSA Blog, February 6, 2025.
- [10] Cloud Security Alliance. "[AI Controls Matrix v1.1](#)." CSA, June 22, 2026.