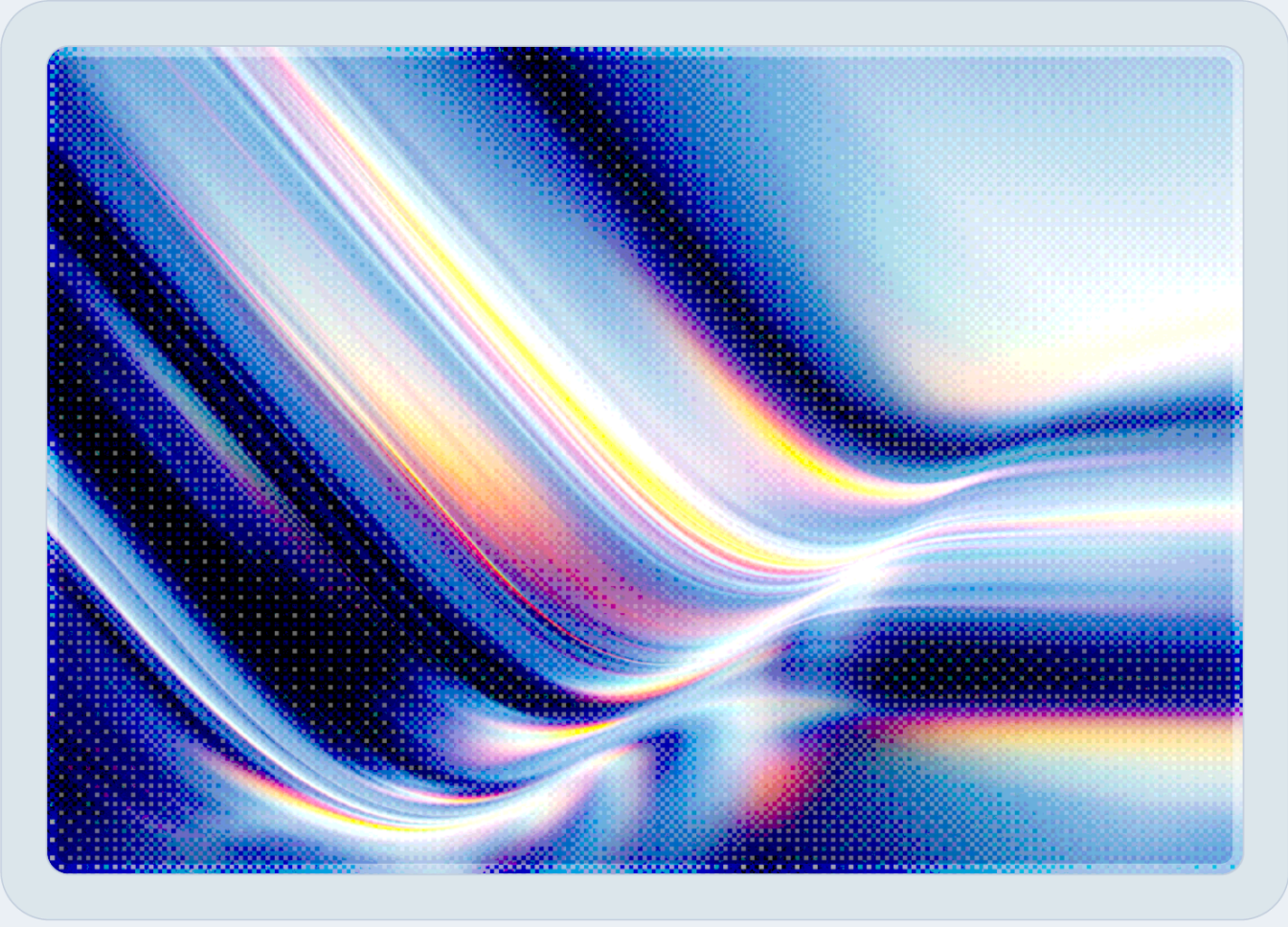
CSAI Foundation | Cloud Security Alliance

GPUStack CVE-2026-102895 KEV Listing: Security Implications and Guidance

A loopback-trust admin bypass turns GPU inference clusters into shell hosts

2026-11-10

AI-assisted Rapid Research



CSAI

cloud 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

On November 9, 2026, the U.S. Cybersecurity and Infrastructure Security Agency (CISA) added CVE-2026-102895 to its Known Exploited Vulnerabilities (KEV) catalog. The flaw is in GPUStack, an open-source GPU cluster manager for serving AI models. CISA set a federal remediation due date of November 12, 2026, which gives agencies three days, and it flagged the entry as requiring forensic triage under Binding Operational Directive 26-04 [1][10]. In GPUStack versions before 2.2.0, any request that appears to come from the loopback address (127.0.0.1) is treated as the built-in administrator. That happens by default, because the `force_auth_localhost` setting is false out of the box. A remote request can look like it came from loopback in two ways: it passes through a reverse proxy on the same host, or it carries a forged `X-GPUStack-Real-IP` header [2][3]. The CVE Numbering Authority (GitHub) scores the flaw CVSS 3.1 9.8 (Critical), as shown on NVD, and NVD records exploitation in the wild against a version 2.1.1 deployment [2].

The practical impact goes well beyond the management console. With administrator access, an attacker can mint persistent API keys, read the credentials stored for workers and cloud providers, and register a malicious inference backend through `POST /v2/inference-backends/from-yaml`. Registering that backend runs code on the GPU worker hosts, and the reported exploitation of a version 2.1.1 deployment used this path to launch reverse shells on workers [2][3][4].

The first action for security teams is to upgrade to at least version 2.2.0, and preferably to 2.2.2. Version 2.2.0 removes the loopback auto-admin path, while 2.2.2 also fixes CVE-2026-102894, a separate CVSS 9.1 SAML authentication bypass [2][5]. Patching alone, however, does not evict an attacker, because upgrading neither revokes attacker-created users or API keys nor cleans compromised worker hosts [4]. Given the confirmed exploitation and CISA's forensic-triage flag, teams should treat any exposed pre-2.2.0 instance as potentially compromised and rotate every worker and cloud-provider credential that GPUStack held [1][4]. Finally, the trust bug reflects a design pattern rather than a one-off coding mistake [3]. Any AI platform that derives identity from network position can fail the same way once it sits behind a same-host proxy.

Background

GPUStack is an Apache-2.0 licensed project that manages GPU clusters for AI model serving and provisions GPU instances [6]. One server process schedules work across worker nodes that may be on-premises, in Kubernetes, or in public clouds. Workers join the cluster with a server URL and a registration token. The server chooses and configures an inference engine, such as vLLM, SGLang, TensorRT-LLM, or a custom engine, and exposes models through an OpenAI-compatible API [6]. The project supports a broad range of accelerators from NVIDIA, AMD, Huawei Ascend, and several other vendors, making it a candidate for organizations building private inference capacity on mixed hardware. On November 5, 2026, the project had roughly 6,000 GitHub stars, and its Docker Hub image had more than 940,000 pulls [4]. Those figures show how widely the software is used, but they are not a count of exposed or vulnerable deployments.

GPUStack's role in the stack is what gives this flaw its severity. A GPUStack server is a control plane: it decides which code runs on expensive accelerator hosts, and it holds the credentials needed to reach those hosts and the clouds behind them. Taking over the control plane therefore hands the attacker every worker it manages. CSA has documented the same dynamic for other AI infrastructure components that reached the KEV catalog this year. Those include the Ray distributed compute framework, which CISA added in August 2026 [12], and the LiteLLM AI gateway [14]. GPUStack's own history also shows that the management surface has had trouble with unauthenticated access. CVE-2026-58658, disclosed in July 2026, let unauthenticated clients stream inference logs containing prompts and completions through unprotected worker endpoints [7].

The disclosure timeline matters for prioritization because a patched release was available long before the KEV listing. According to Ostorlab's reconstruction, the fix commit landed on May 19, 2026, and GPUStack 2.2.0 shipped on June 18, 2026. The GitHub repository security advisory (GHSA-gj9m-f8ph-pmhr) followed on July 31, 2026. The CVE identifier was published to NVD on November 4, 2026, and CISA listed the flaw in KEV five days later [3][4][9]. The advisory credits independent discovery to Nikko Enggaliano, Wei Yunzhou, and Rui Yang and Haoyu Wang of Johns Hopkins University, and credits Javier Sixto of ulex.fr with reporting the in-the-wild exploitation [3]. Organizations that track vulnerabilities only by CVE feed may not have seen this issue until November. That leaves a window of several months in which a fix existed but the vulnerability may not have been visible in CVE-driven tracking. The table below summarizes the sequence.

Date	Event	Source
2026-05-19	Fix commit lands in GPUStack repository	[4]

Date	Event	Source
2026-06-18	GPUStack 2.2.0 released with fix	[4]
2026-07-31	GitHub repository advisory GHSA-gj9m-f8ph-pmhr published	[3]
2026-11-04	CVE-2026-102895 and CVE-2026-102894 published to NVD	[2][5]
2026-11-09	CVE-2026-102895 added to CISA KEV	[1]
2026-11-12	Federal remediation due date	[1]

Security Analysis

Root Cause: Network Position as Identity

The flaw is in `get_current_user` in `gpustack/api/auth.py`. If a request carries no credentials and its apparent source address is loopback, the function loads the built-in administrator account. It does not check the HTTP method, the path, or who the caller actually is [2][4]. The developers seem to have meant this as a convenience for local administration. On a single host with no intermediary, a request from 127.0.0.1 can only come from a process on that machine. That assumption fails as soon as the deployment adds a same-host reverse proxy, which the advisory describes as the canonical self-hosted shape for putting a service behind TLS and a public hostname [3]. The advisory frames the root cause directly: the system trusted a request's network source, whether the socket peer or a header the client controls, as proof of identity [3]. MITRE's weakness taxonomy covers the same failure: improper authentication (CWE-287), authentication bypass by spoofing (CWE-290), and origin validation error (CWE-346) [1][3].

The source address comes from a helper called `client_ip_getter`, and the two exploitation paths differ in what the attacker needs. In the first path, an Nginx, Traefik, or Caddy instance on the same host forwards traffic with a directive such as `proxy_pass http://127.0.0.1:<port>`. The TCP peer GPUStack sees is then always loopback, so every unauthenticated request from the internet arrives as administrator. The attacker needs no header and no credential, only a request [3][4]. In the second path, the client supplies `X-GPUStack-Real-IP: 127.0.0.1`, and `client_ip_getter` accepts the value. The GitHub advisory and NVD describe the header path as generally available [2][3].

Ostorlab's code analysis, however, says the helper honors the header only in embedded gateway mode [4]. Defenders should take the broader reading until they confirm their own configuration, since either way the result is full administrative access.

The fix in 2.2.0 removes the loopback auto-admin branch entirely. It also changes the code so the forwarding header is read only after the gateway authenticates with its token [4]. The vendor's own guidance is to stop deriving authentication from network source and to accept client-IP headers only from an authenticated, trusted gateway [3]. That guidance applies well beyond GPUStack.

From Admin Console to Shell on GPU Workers

Administrative access to GPUStack is a starting point for broader compromise, not the end of it. NVD and the GitHub advisory list what a successful attacker can do. They can enumerate and modify users, models, workers, and configuration. They can create persistent administrator API keys that work from any IP address. They can disclose worker credentials and cloud-provider secrets. And they can achieve code execution on worker hosts through the `from-yaml` inference backend endpoint [2][3]. Ostorlab notes that an API key with the `*` scope stays valid even after the operator removes the vulnerable proxy rule [4]. Persistence therefore survives the obvious first reaction of closing off the proxy.

The observed attack chain follows this progression. The attackers reached the administrator context, registered custom inference backends, and used those backends to launch reverse shells on workers of a version 2.1.1 deployment [2][3]. Public reporting does not include victim counts, dates of first exploitation, or indicators of compromise [4]. Reports also differ on whether a public proof-of-concept exists: one outlet states that one does [8], and the vendor and NVD records do not mention it. In CSA's assessment, the chain is simple enough to reconstruct from the advisory text alone.

Why GPU Inference Clusters Are High-Value Targets

Several features of GPU inference infrastructure raise the consequences of this flaw beyond those of a typical web application bypass. The table below summarizes the assets an attacker gains and how each could plausibly be abused. The abuse scenarios are informed inferences drawn from documented campaigns against similar infrastructure, not confirmed observations from the GPUStack exploitation.

Asset reachable via GPUStack admin	Documented capability	Plausible abuse
Worker hosts with GPUs	Code execution via custom backend [2]	Cryptomining, LLMjacking, staging offensive tooling

Asset reachable via GPUStack admin	Documented capability	Plausible abuse
Cloud-provider secrets	Disclosure of stored secrets [2]	Pivot into AWS, GCP, or Azure accounts; resource creation
Model registry and configuration	Model changes [2]	Swapping or tampering with served models
Inference traffic	Admin control over serving backends [3]	Interception of prompts and completions
Persistent API keys	Creation of admin keys [2]	Long-term re-entry after patching

CSA's analysis of the NadMesh botnet found attackers treating an exposed self-hosted AI service as an entry point to broader cloud and Kubernetes environments, with the specific aim of harvesting cloud keys and model credentials [11]. A GPUStack server holding cloud-provider secrets fits that targeting profile closely. CSA's research on rapid exploitation of AI inference frameworks also observed that these deployments often run on slower patch cycles than externally facing services [13]. The exploited deployment was still running version 2.1.1 months after 2.2.0 shipped, which is consistent with those slower patch cycles. Public data on GPUStack upgrade rates, however, is not available.

Assessing Exposure

An instance is exposed to the full-impact scenario when three conditions hold together. The server is earlier than version 2.2.0; `force_auth_localhost` is false or unset; and the API sits behind a same-host reverse proxy that forwards from loopback, or is directly reachable by clients able to set the `X-GPUStack-Real-IP` header [4]. Ostorlab notes that the `FORCE_AUTH_LOCALHOST` environment variable prints nothing at its default setting. An operator inspecting the environment may therefore read a blank value as "not configured" rather than "insecure default" [4]. Deployments reachable only over a VPN or through an authenticating edge are at lower risk, but they are not safe. An attacker with any foothold on the internal network, or a server-side request forgery in an adjacent service, could still reach the vulnerable branch.

Recommendations

Immediate Actions

The first priority is to inventory every GPUStack server, including instances run by research teams and lab clusters outside central IT. Confirm their versions with `gpustack version` or the container equivalent [4]. Upgrade any instance below 2.2.0 now, and recreate containers rather than restarting them so that the new image actually takes effect [4]. Organizations should move directly to 2.2.2 or later. That release also fixes CVE-2026-102894, a separate pre-authentication flaw in the `/auth/saml/callback` endpoint. In affected versions, that endpoint minted session tokens without verifying SAML signatures, even on deployments where SAML was never configured [5]. CVE-2026-102894 is not currently listed in KEV [1], but it is also unauthenticated and has the same outcome of administrator impersonation.

Where an upgrade cannot happen within hours, apply the interim controls published in the advisory. Set `force_auth_localhost=true` so loopback requests must authenticate. Configure the edge proxy to strip any client-supplied `X-GPUStack-Real-IP` header. Make sure the proxy cannot make external traffic appear to come from loopback. Remove the API from untrusted networks behind a VPN or an authenticating gateway [3]. These steps reduce risk but do not replace patching. Stripping the header alone, for example, leaves the same-host proxy path open [4][8].

Because CISA flagged this entry for forensic triage and exploitation is confirmed, teams should hunt for compromise on any instance that was reachable while vulnerable, rather than assume patching closed the matter [1][4]. The hunt should follow the attack chain described above, moving from the GPUStack control plane out to the worker hosts and cloud accounts it could reach. The table below sets out the areas to examine and the action to take in each.

Area	What to check	Action
Users and API keys	All administrator accounts and API keys, with particular attention to keys carrying the `*` scope [4]	Revoke any you did not create
Inference backends	Registered backends compared against change records [4]	Examine any backend created through `from-yaml`

Area	What to check	Action
Proxy and server logs	Successful API calls without an authentication header, and POST requests to `/v2/inference-backends/from-yaml` [4][8]	Treat matches as indicators of compromise and scope the affected window
Worker hosts	Process trees and outbound connections for shells spawned by inference backend processes [4]	Isolate and rebuild hosts showing unexpected shells
Credentials and cloud accounts	All worker tokens and cloud-provider credentials GPUStack held [4]	Rotate them, then audit the cloud accounts for resources you did not create

Completing these checks is what distinguishes a patched instance from a remediated one, since the persistence mechanisms an attacker would rely on survive the upgrade.

Short-Term Mitigations

Over the following weeks, organizations should review the trust boundary between their reverse proxies and every AI service behind them, not just GPUStack. The pattern that made this flaw exploitable, a co-located proxy forwarding from 127.0.0.1 to a service that grants privilege to loopback, appears in many quick-start deployments of model servers, notebooks, and agent platforms. For each such service, confirm that authentication is enforced regardless of source address. Confirm also that forwarding headers such as `X-Forwarded-For`, `X-Real-IP`, and vendor-specific variants are overwritten at the edge, not passed through from clients.

Network segmentation for GPU workers deserves the same scrutiny. Workers that can open arbitrary outbound connections make reverse shells and credential exfiltration easy. Egress filtering that allows only model registries, package mirrors, and the control plane would likely have raised the cost of the observed reverse-shell technique. Cloud credentials held by cluster managers should also be narrowly scoped and short-lived wherever possible. That way, disclosing them grants a bounded and expiring capability rather than broad account access.

Strategic Considerations

Together with the Ray and LiteLLM KEV listings earlier this year [12][14], the GPUStack case adds to the evidence that AI-serving control planes are being actively exploited, and it supports governing them like any other privileged management infrastructure. In practice, that means including them in the asset inventory, vulnerability scanning, and patch service-level agreements that apply to hypervisor managers and CI/CD systems. Vulnerability intake should watch GitHub security advisories directly for open-source AI components, because this case shows that a fix and advisory can predate the CVE record by months [3][4].

At the architectural level, the lesson echoes long-standing Zero Trust principles: network location is not identity. Security reviews for AI platforms, whether built in-house or adopted from open source, should explicitly test for "trusted localhost," "trusted header," and "trusted subnet" shortcuts. They should also confirm that every administrative route requires a cryptographically verifiable credential. Procurement and open-source intake reviews can add a simple check: does the software ship with authentication enforced for all callers by default? A default of `force_auth_localhost=false` would not survive that check.

CSA Resource Alignment

CSA's research note on Ray's KEV listing is the closest precedent [12]. Ray is another distributed AI compute framework, and its compromise likewise gave attackers code execution across a cluster of accelerator hosts. GPUStack follows Ray into the catalog with a similar blast radius. Both cases make the same point: cluster schedulers and serving managers for AI workloads need to be treated as privileged infrastructure. The guidance in that note on isolating cluster control APIs and hunting for unauthorized job submissions maps directly to GPUStack's inference backends.

CSA's analysis of the LiteLLM AI gateway KEV-listed attack chain [14] looked at a closely related component class, AI middleware that holds provider credentials and mediates model access. That note's emphasis on rotating credentials after compromise and revoking unrecognized keys applies directly here, given GPUStack's persistent API keys and stored cloud secrets. CSA's research note on NadMesh, a botnet built to hunt exposed AI infrastructure for cloud keys [11], gives the threat-actor context. It shows that opportunistic actors already scan for self-hosted AI services specifically to pivot into cloud accounts. CSA's note on sub-24-hour exploitation of AI inference frameworks [13] supports the urgency of the recommendations above, documenting compressed disclosure-to-exploitation timelines and slow patch cycles across the inference ecosystem.

For control mapping, the CSA AI Controls Matrix (AICM) v1.1 [15] applies across several domains. Identity and Access Management controls address the core defect of granting privilege without verified authentication. Threat and Vulnerability Management controls cover timely remediation of KEV-listed components and monitoring of upstream advisories. Infrastructure and Virtualization Security controls support segmentation and egress restriction for GPU worker hosts. Logging and Monitoring controls underpin the compromise-assessment steps. Organizations already using AICM can treat GPUStack and comparable cluster managers as in-scope AI infrastructure assets under those control objectives.

References

[1] CISA. "Known Exploited Vulnerabilities Catalog — CVE-2026-102895." Cybersecurity and Infrastructure Security Agency, November 9, 2026.

[2] NIST. "CVE-2026-102895 Detail." National Vulnerability Database, November 4, 2026.

[3] GPUStack. "GHSA-gj9m-f8ph-pmhr: Authentication bypass via trusted loopback source (reverse-proxy TCP peer or X-GPUStack-Real-IP header) grants unauthenticated admin." GitHub Security Advisory, July 31, 2026 (updated September 29, 2026; added to GitHub Advisory Database November 9, 2026).

[4] Ostorlab. "GPUStack Auth Bypass RCE, Exploited in the Wild (CVE-2026-102895)." Ostorlab Blog, November 5, 2026 (updated November 10, 2026).

[5] NIST. "CVE-2026-102894 Detail." National Vulnerability Database, November 4, 2026.

[6] GPUStack. "gpustack/gpustack." GitHub, accessed November 10, 2026.

[7] NIST. "CVE-2026-58658 Detail." National Vulnerability Database, July 15, 2026.

[8] byteiota. "GPUStack CVE-2026-102895: Auth Bypass Exploited — Patch Now." byteiota, November 2026 (accessed November 10, 2026).

[9] GitLab. "GPUStack: Authentication bypass via trusted loopback source (CVE-2026-102895)." GitLab Advisory Database, November 9, 2026.

[10] CISA. "BOD 26-04: Prioritizing Security Updates Based on Risk." Cybersecurity and Infrastructure Security Agency, June 10, 2026.

[11] Cloud Security Alliance. "NadMesh: A Botnet Built to Hunt Exposed AI Infrastructure for Cloud Keys." CSA AI Safety Initiative, July 20, 2026.

[12] Cloud Security Alliance. "Ray Framework RCE Under Active Exploitation Joins CISA KEV." CSA AI Safety Initiative, August 18, 2026.

[13] Cloud Security Alliance. "Sub-24-Hour Exploitation of AI Inference Frameworks." CSA AI Safety Initiative, May 19, 2026.

[14] Cloud Security Alliance. "LiteLLM AI Gateway: KEV-Listed Attack Chain Enables Full Takeover." CSA AI Safety Initiative, June 17, 2026.

[15] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, June 22, 2026.