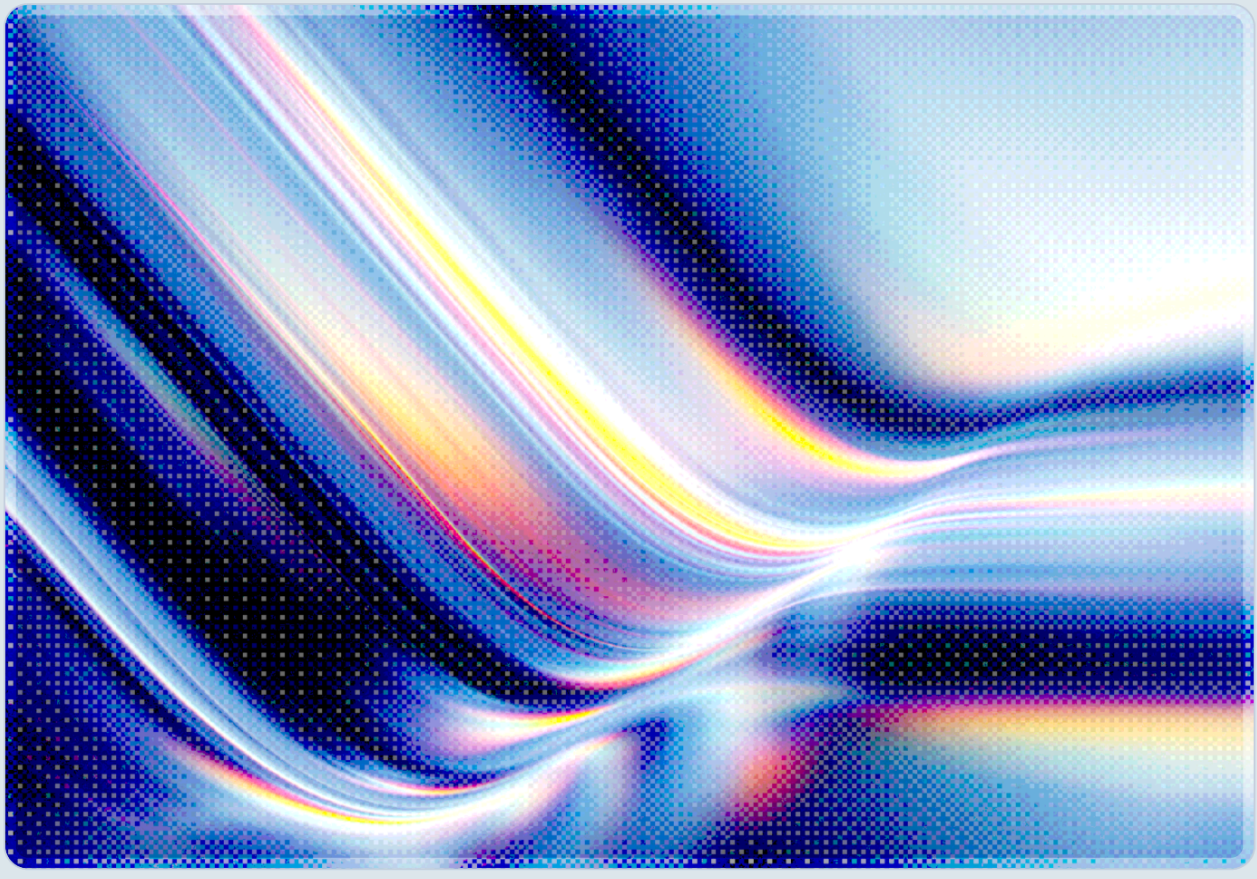
CSAI Foundation | Cloud Security Alliance

LiteLLM's Default Admin Key: A Cloud Credential Vault

Security Implications and Guidance

2026-09-10

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Wiz Research scanned 3,074 internet-facing LiteLLM gateways on Shodan in February 2026 and found that 294 of them, roughly 9.6 percent, accepted `sk-1234`, the example master key printed in LiteLLM's own setup documentation; 191 of those had no key configured at all and would have accepted any credential offered [1][2].
- The LiteLLM master key functions as a full administrative credential: whoever holds it can read every model provider API key stored on the gateway, inspect proxied prompts and responses, reach any MCP-connected tool server, and in Wiz's testing, use unrestricted pass-through routing to query the host's cloud instance metadata service and retrieve IAM credentials [1].
- CVE-2026-59822, a separate authentication bypass in LiteLLM's MCP Streamable HTTP endpoint that let an unauthenticated caller reach MCP tooling using a fabricated bearer token, was added to CISA's Known Exploited Vulnerabilities catalog on September 2, 2026, with a federal remediation deadline of September 16, 2026; industry trackers have described it as the first MCP-specific flaw on the KEV list [3][8].
- A LiteLLM compromise disclosed publicly in August 2026 showed the pattern working end to end: an attacker who obtained code execution on a gateway host read the container's environment variables, recovered the master key and a database connection string, and used the latter to copy records directly out of LiteLLM's backing PostgreSQL database [2].
- Wiz's broader telemetry indicates that roughly one-third of surveyed cloud environments run a LiteLLM deployment, and because the gateway is frequently granted cloud IAM permissions to reach model providers and internal services, a single exposed or default-credentialed instance can function as a pivot point into the surrounding cloud account rather than a contained application flaw [1].

Background

LiteLLM is an open-source proxy, maintained by BerriAI, that gives enterprises a single API surface for calling more than a hundred large language model providers, translating requests and normalizing responses so application teams do not have to write separate integration code for OpenAI, Anthropic, Google, and other backends. That consolidation is exactly what makes the software attractive: rather

than scattering provider API keys across dozens of internal services, an organization centralizes them behind one gateway, applies rate limits and spend controls in one place, and standardizes logging and observability for every model call flowing through the enterprise. The tradeoff that comes with that design is architectural rather than accidental. A LiteLLM instance is, by construction, a single system that holds the credentials for every model provider an organization uses, and increasingly it also holds the cloud identity of the host it runs on, because many deployments grant the gateway an IAM role so it can reach cloud-native model endpoints such as Amazon Bedrock or Vertex AI without a separately managed key.

LiteLLM's setup documentation instructs administrators to configure a master key that governs administrative access to the proxy, and the example value shown throughout that documentation is the string `sk-1234`. That value was always intended as a placeholder to be replaced before production use, in the same category as `admin/admin` or a default router password. Wiz Research's February 2026 internet scan established that a meaningful share of operators never made that substitution: of 3,074 LiteLLM gateways discoverable through Shodan, 294 accepted the literal example key, and within that group, 191 had not set any master key at all, meaning the gateway would authenticate a request bearing any string whatsoever [1][2]. The remaining instances had specifically left the documentation's placeholder value in production. Neither failure mode requires an attacker to find a bug in LiteLLM's code; both simply require finding the gateway's address and trying the value printed in the project's own README.

This finding does not stand alone. It surfaced alongside a September 2026 wave of LiteLLM security disclosures that, taken together, describe a pattern CSA's AI Safety Initiative has now tracked across three prior rapid-response advisories: LiteLLM gateways are attractive, frequently internet-facing, and under active attack, and the vulnerabilities affecting them compound each other because the gateway concentrates so much value behind a single authentication boundary [1][2][3]. This note focuses specifically on the credential and identity dimension of that pattern — what happens once the master key, whether guessed, defaulted, or leaked, ends up in an attacker's hands — and on CVE-2026-59822, a distinct authentication bypass in LiteLLM's Model Context Protocol (MCP) support that CISA has now confirmed is being exploited in the wild.

Security Analysis

The master key as a single point of failure

LiteLLM's authorization model treats the master key as a fully trusted administrative credential, and that design decision has direct consequences once the key is exposed. An attacker who obtains a valid master key, whether by trying the documented default, brute-forcing a weak value, or extracting it from a compromised host, gains the same access an intended administrator would have. That access includes reading the provider API keys the gateway stores for every backend it proxies to, which means a single leaked LiteLLM credential could cascade into leaked OpenAI, Anthropic, Google, or other provider keys that were never directly exposed anywhere else. It also includes visibility into the prompts and completions flowing through the proxy, which for many enterprise deployments means exposure to whatever sensitive business data, customer information, or proprietary logic is embedded in those exchanges, and access to any MCP tool servers the gateway is configured to reach, extending the blast radius from the LLM traffic itself into whatever internal systems those tools were built to touch.

Wiz's research pushed the analysis one step further and found that the master key's reach extends beyond the application layer into the underlying cloud identity of the host. LiteLLM exposes pass-through routing features that let an authenticated administrator direct a request at an arbitrary target URL, and Wiz found that this routing does not validate the target against private address ranges, localhost, or cloud metadata service addresses. An administrator, or an attacker holding the master key, can point that routing at the instance metadata service running on the gateway's host and retrieve the IAM credentials the cloud provider makes available there. Wiz further found that switching the underlying cloud metadata protection to IMDSv2, which normally defends against this class of server-side request forgery by requiring a session token obtained through a separate call, did not stop the technique, because LiteLLM's pass-through logic forwards any header an operator sends with an `x-pass-` prefix to the target after stripping the prefix, allowing the required session-token header to be smuggled through [1]. Wiz characterizes this specific pivot as consistent with LiteLLM's stated threat model, which treats administrators as trusted actors, and notes that the behavior carries no CVE identifier and, as of publication, no fix. That characterization is defensible as a statement about intended trust boundaries, but it does not change the operational consequence: once a default or leaked master key erases the assumption that only trusted administrators hold it, the same pass-through behavior becomes a direct path from application-layer compromise to cloud account compromise.

CVE-2026-59822 and the MCP authentication bypass

Separately from the master key exposure, LiteLLM's MCP integration carried its own authentication flaw, tracked as CVE-2026-59822. Several outlets report this vulnerability at a CVSS 4.0 score of 8.8, though GitLab's advisory lists a base score of 8.2 for the same flaw, and this note does not attempt to adjudicate between the two scoring methodologies beyond noting that both place it in the high-to-critical range [4][8]. The vulnerability affected LiteLLM's MCP Streamable HTTP endpoint, the interface through which the gateway exposes MCP tool sessions to clients. According to the public advisory, an unauthenticated caller could present a fabricated Authorization header that triggered an OAuth2 passthrough fallback path in LiteLLM's authentication logic; when that path failed to validate the supplied token, it substituted an empty `UserAPIKeyAuth()` object rather than rejecting the request, which allowed the request through to MCP tooling without ever presenting a valid LiteLLM key [4]. In practical terms, an attacker did not need the master key, a valid API key, or any credential at all to list and invoke whatever MCP tools a given LiteLLM deployment exposed; a crafted request against an unpatched endpoint was sufficient. The issue is fixed in LiteLLM 1.84.0 and later [3][4].

CISA added CVE-2026-59822 to its Known Exploited Vulnerabilities catalog on September 2, 2026, based on confirmed active exploitation, setting a remediation deadline of September 16, 2026 for federal civilian agencies [3]. Security researchers have noted that this makes it the first MCP-specific vulnerability to appear on the KEV list, a milestone that raises questions about the runtime hardening of MCP integrations more broadly, though this note speaks only to LiteLLM's implementation [3][8]. The KEV listing sat alongside a related, separately tracked flaw, CVE-2026-42271, an authenticated command-execution vulnerability in LiteLLM versions 1.74.2 through versions before 1.83.7, which Wiz's honeypot telemetry observed being actively exploited through MCP test endpoints to spawn attacker-controlled subprocesses, in at least some cases to install cryptocurrency-mining software on the compromised host [1][2].

From gateway compromise to cloud account compromise: a documented case

The theoretical chain connecting a compromised LiteLLM gateway to broader cloud impact was demonstrated in a real incident Microsoft disclosed in August 2026. In that case, an attacker who achieved command execution inside a LiteLLM gateway process read the container's environment variables directly, recovering the configured master key, the API keys for the model providers the gateway proxied to, and a connection string for the underlying PostgreSQL database that LiteLLM uses to persist configuration, virtual keys, and usage records. The attacker then used that connection string to connect directly to the database and copy records from the model and virtual-key tables [2]. This one documented case suggests that the risk described in this note is not confined to internet-exposed

gateways with default keys; it also applies to internally deployed gateways whose environment variables or database credentials become reachable through any other foothold, because the gateway's own storage of secrets makes it a productive target once an attacker is anywhere near it.

The table below summarizes the distinct but related issues covered here, since they are frequently conflated in vendor and press coverage but require different remediation steps.

Issue	Type	Requires prior credential?	Fixed by
Default/unset master key (`sk-1234`)	Configuration weakness	No – key is guessable or absent	Operator sets a strong, unique master key
Metadata service pass-through (SSRF to IMDS)	Design behavior within admin trust model	Yes – requires master key	No fix released; requires network-layer mitigation
CVE-2026-59822 (MCP auth bypass)	Software vulnerability	No – unauthenticated	Upgrade to LiteLLM 1.84.0+
CVE-2026-42271 (MCP command execution)	Software vulnerability	Yes – authenticated	Upgrade to version 1.83.7+

Recommendations

Immediate Actions

Organizations running LiteLLM in any environment, whether internet-facing or internal, should treat master key hygiene and patch currency as the two most urgent items on this list. Rotate the LiteLLM master key to a long, randomly generated value if it has never been explicitly set or if it still matches the documentation's example, and confirm that no fallback path allows an unset or empty key to authenticate successfully. Upgrade LiteLLM to version 1.84.0 or later, which addresses CVE-2026-59822 and CVE-2026-49468 – an earlier authentication bypass in which LiteLLM's auth layer derived the effective request route from a Host header that Starlette reconstructs, so a crafted Host header could make the gateway evaluate its authentication decision against a public route while still dispatching

the request to a protected management endpoint [9] — and confirm the deployed version also postdates 1.83.7 to close the MCP command-execution path tracked as CVE-2026-42271. Audit environment variables and secrets storage on every host or container running LiteLLM to confirm that the master key, provider API keys, and database connection strings are not readable by any process or user beyond the gateway itself, since the Microsoft-disclosed incident depended entirely on environment-variable exposure rather than a software flaw.

Short-Term Mitigations

Beyond the immediate patch and rotation work, security teams should restrict network exposure and validate that pass-through routing cannot reach sensitive internal targets. Place LiteLLM gateways behind network controls that prevent direct internet access to the administrative and MCP endpoints, reserving any public exposure for the minimum set of routes application traffic actually requires. Where the deployment platform supports it, block outbound requests from the LiteLLM host to link-local metadata addresses at the network layer, rather than relying solely on IMDSv2 token requirements, since LiteLLM's header pass-through behavior has been shown to defeat that protection from within an authenticated admin session. Review MCP tool server configurations attached to each gateway and confirm that the tools it can reach are limited to what the gateway's legitimate use case requires, since MCP connectivity extends the practical impact of any authentication bypass into whatever systems those tools touch.

Strategic Considerations

Over the longer term, organizations should treat AI gateways as tier-one identity infrastructure rather than as application middleware, because that is what their credential-holding role has made them. Bring LiteLLM and comparable AI gateway deployments into existing secrets management and privileged access review processes, with the same rotation cadence, access logging, and least-privilege IAM role scoping applied to database servers or credential vaults. Where a gateway is granted a cloud IAM role to reach cloud-native model endpoints, scope that role as narrowly as the specific model APIs required, so that a metadata-service credential leak yields limited value rather than broad account access. Finally, incorporate AI gateway configuration checks, including master key strength and version currency, into routine vulnerability management and continuous security monitoring rather than treating them as a one-time deployment step, given how quickly new LiteLLM CVEs have continued to surface through 2026.

CSA Resource Alignment

This note extends a pattern CSA's AI Safety Initiative has tracked closely through 2026. LiteLLM AI Gateway: Critical Vulnerability Chain Exposes API Keys [5], published in June 2026, documented an earlier cascade of supply chain, injection, and remote code execution flaws in the same platform that similarly converged on the gateway's role as a centralized credential store; the default admin key issue described here is best understood as a configuration-layer instance of the same structural weakness that report identified at the code level. LiteLLM AI Gateway: KEV-Listed Attack Chain Enables Full Takeover [6] covers the KEV-listing dynamics and credential-exfiltration outcomes of a related exploit chain, and its remediation guidance on patch currency and credential rotation applies directly to CVE-2026-59822. A third piece, LiteLLM AI Gateway: Active Exploitation via MCP Injection [10], published June 13, 2026, examined CVE-2026-42271's exploitation through MCP test endpoints in more detail than this note's comparison table provides, and remains the more complete reference for organizations investigating that specific command-execution path.

The AI Controls Matrix (AICM) v1.1 [7] provides the governing control framework for the identity and secrets-management failures this note describes. AICM's identity and access management domain calls for strong authentication and credential lifecycle controls for AI infrastructure components, and its treatment of AI system supply chain and configuration management directly addresses the default-credential pattern Wiz documented: a gateway shipped with a documented example key is only as secure as the operator's discipline in replacing it, which is precisely the kind of control AICM's IAM domain is designed to enforce and audit. Organizations conducting an AICM-aligned assessment of their AI infrastructure should treat LiteLLM master key configuration, MCP endpoint exposure, and pass-through routing scope as concrete control points to verify rather than abstract policy statements.

References

[1] Wiz. "Off Guard: Breaking LiteLLM from Authentication Bypass to Cloud Compromise." Wiz Blog, September 2026.

[2] The Hacker News. "Nearly 1 in 10 Exposed LiteLLM Gateways Accepted the Example "sk-1234" Admin Key." The Hacker News, September 10, 2026.

[3] CISA. "CISA Adds Seven Known Exploited Vulnerabilities to Catalog." Cybersecurity and Infrastructure Security Agency, September 2, 2026.

[4] GitLab Advisory Database. "CVE-2026-59822: LiteLLM: MCP Authentication Bypass via OAuth2 Passthrough Fallback." GitLab, 2026.

[5] Cloud Security Alliance. "LiteLLM AI Gateway: Critical Vulnerability Chain Exposes API Keys." Cloud Security Alliance, June 2026.

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

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

[8] Tech Insider. "CISA Flags First MCP Flaw: LiteLLM CVE-2026-59822." Tech Insider, September 2026.

[9] GitLab Advisory Database. "CVE-2026-49468: LiteLLM: Authentication Bypass via Host Header Injection." GitLab, 2026.

[10] Cloud Security Alliance. "LiteLLM AI Gateway: Active Exploitation via MCP Injection." Cloud Security Alliance, June 13, 2026.

