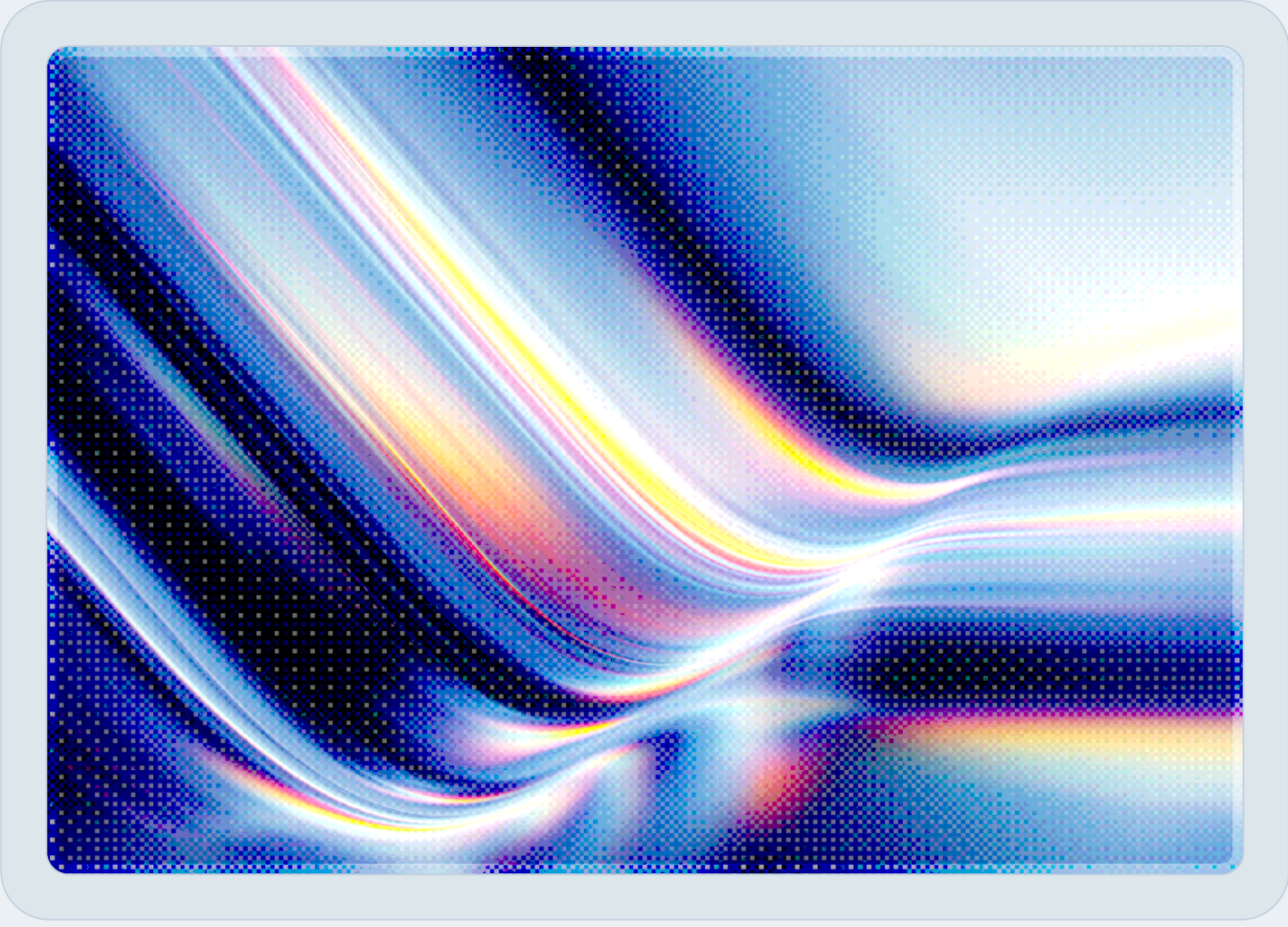
CSAI Foundation | Cloud Security Alliance

LiteLLM AI Gateway: KEV-Listed Attack Chain Enables Full Takeover

Active Exploitation of CVE-2026-42271; Multi-CVE Chain Reaches CVSS 10.0

2026-06-17

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- The U.S. Cybersecurity and Infrastructure Security Agency (CISA) added CVE-2026-42271, a command injection flaw in LiteLLM's Model Context Protocol (MCP) preview endpoints, to its Known Exploited Vulnerabilities (KEV) catalog on June 8, 2026, with a federal remediation deadline of June 22, 2026 [1][11].
- When CVE-2026-42271 is chained with CVE-2026-48710 ("BadHost"), a host header validation bypass in the Starlette web framework, attackers can achieve unauthenticated remote code execution (RCE) requiring no API key, no login, and no prior foothold — a practical severity that security researchers assess as equivalent to CVSS 10.0, though no single CVE in the chain carries that score independently [2].
- A separate three-CVE privilege escalation chain (CVE-2026-47101, CVE-2026-47102, CVE-2026-40217) allows a low-privilege account holder to escalate to administrator and execute arbitrary Python code, rated CVSS 9.9 [3].
- In March 2026, a threat actor tracked as TeamPCP poisoned LiteLLM versions 1.82.7 and 1.82.8 on PyPI, deploying a three-stage payload that harvested cloud credentials and installed persistent backdoors in Kubernetes environments [4].
- Organizations must upgrade LiteLLM to v1.83.14-stable or later, upgrade Starlette to v1.0.1 or later, and immediately rotate all AI provider API keys stored by the proxy [2][3].

Background

LiteLLM is an open-source Python proxy and software development kit that provides a unified API interface for interacting with more than 100 large language model (LLM) providers, including OpenAI, Anthropic, Google Gemini, AWS Bedrock, and Azure OpenAI. Organizations deploy it as an "AI gateway" — a centralized routing, load balancing, rate limiting, and logging layer that sits between internal applications and upstream model providers. Because of this architectural role, a single LiteLLM proxy instance frequently holds authentication credentials for an organization's entire AI provider ecosystem, including master keys, provider API keys, database connection strings, and routing secrets. LiteLLM is among the most widely deployed open-source AI gateway packages, with millions of monthly downloads across the PyPI ecosystem [4].

The gateway's position as a trust broker between users, applications, and model providers creates an attractive target profile. Unlike a typical web application vulnerability, successful exploitation of an AI gateway exposes a broader credential inventory: every model provider key, internal API key, and logged conversation the gateway proxies. The blast radius scales directly with the organization's AI deployment footprint. For organizations that have integrated LiteLLM as the backbone of their AI operations — routing sensitive queries involving source code, personnel data, financial records, or strategic documents — the impact of credential exfiltration extends well beyond the gateway server itself.

Between February and June 2026, security researchers disclosed at least six CVEs affecting LiteLLM across three distinct vulnerability classes: command injection in MCP test endpoints, privilege escalation through missing field-level authorization controls, pre-authentication SQL injection, and a compounding host header bypass in LiteLLM's upstream web framework dependency. A concurrent supply chain attack against the project's PyPI distribution channel underscored the breadth of the attack surface. The six CVEs disclosed during this period, spanning four distinct vulnerability classes and culminating in an active KEV listing within weeks of discovery, represent an unusually concentrated period of high-severity AI infrastructure disclosures.

Security Analysis

KEV-Listed Command Injection: CVE-2026-42271

CVE-2026-42271 is a command injection vulnerability residing in two MCP server preview endpoints introduced in LiteLLM versions 1.74.2 through 1.83.6: `POST /mcp-rest/test/connection` and `POST /mcp-rest/test/tools/list`. These endpoints were designed to allow users to test an MCP server configuration before saving it. However, both accepted a full server configuration object in the request body — including `command`, `args`, and `env` fields used by the stdio transport — and when called with a stdio-mode configuration, LiteLLM spawned the supplied command as a subprocess on the proxy host with the full operating system privileges of the proxy process [2]. No validation, sandboxing, or argument filtering was applied.

Standalone, CVE-2026-42271 carries a CVSS v4.0 score of 8.7 and requires a valid API key to invoke the vulnerable endpoints. That authentication barrier collapses entirely when the vulnerability is combined with CVE-2026-48710 [1][2].

The BadHost Authentication Bypass: CVE-2026-48710

CVE-2026-48710, publicly named "BadHost," is a host header validation vulnerability in Starlette, the ASGI web framework underlying LiteLLM's proxy server. In Starlette versions 1.0.0 and earlier, the HTTP `Host` request header was not validated before being used to reconstruct `request.url`. Because Starlette's routing algorithm operates on the raw HTTP path while access control logic in middleware and endpoints may depend on `request.url`, a malformed `Host` header causes these two values to diverge. Security restrictions applied against `request.url` — including LiteLLM's authentication middleware — can be bypassed with a single injected character in the header [5].

The compound effect eliminates the authentication barrier entirely: an attacker who sends a crafted `Host` header to a LiteLLM instance running on Starlette ≤ 1.0.0 can invoke the MCP test endpoints of CVE-2026-42271 without any API key or session credential. Security researchers assess the practical severity of this chain as equivalent to CVSS 10.0 — the maximum possible rating — because the combination removes all authentication preconditions from an otherwise-authenticated command injection vulnerability [2]. CISA's addition of CVE-2026-42271 to the KEV catalog on June 8, 2026 constitutes official U.S. government confirmation that this chain is actively being weaponized [1].

Privilege Escalation to RCE: CVE-2026-47101, CVE-2026-47102, CVE-2026-40217

Security researchers at Obsidian Security disclosed a second attack chain of comparable severity — rated CVSS 9.9 — beginning with responsible disclosure to BerriAI on February 19, 2026 [3]. Unlike the first chain, this path requires at minimum a low-privilege user account; it does not require Starlette to be unpatched, instead proceeding through three flaws entirely within LiteLLM's own authorization model.

The first vulnerability, CVE-2026-47101, is an authorization bypass in LiteLLM's virtual API key creation flow. When generating a new key, users can supply an `allowed_routes` field specifying which API paths the key is authorized to access. LiteLLM stored this caller-supplied value without validation, meaning any `internal_user` could create a key with `allowed_routes: ["/*"]` — a wildcard granting access to all routes, including admin-only endpoints [3].

The second flaw, CVE-2026-47102 (CVSS 8.8), resides in the `/user/update` and `/user/bulk_update` endpoints. These endpoints applied no field-level authorization checks, allowing any authenticated user to POST `user_role: "proxy_admin"` against their own account record and grant themselves full administrator privileges [3]. Together, CVE-2026-47101 and CVE-2026-47102 form a two-request escalation path from unprivileged account to gateway administrator.

CVE-2026-40217 converts that administrative access into arbitrary code execution. LiteLLM's Custom Code Guardrail feature allowed administrators to supply Python code for filtering model inputs and outputs; the feature compiled and ran this code using Python's built-in `exec()` function while leaving `__builtins__` accessible in the execution namespace. An attacker with admin rights could therefore submit a guardrail payload that spawns a reverse shell, reads secrets from environment variables, or exfiltrates the proxy's configuration [3]. A secondary bypass — runtime bytecode rewriting to circumvent the guardrail endpoint's regex deny-list — was also documented. Partial fixes were released across multiple pull requests between February and April 2026, with complete remediation reaching v1.83.14-stable on April 25, 2026 [3].

Pre-Authentication SQL Injection: CVE-2026-42208

Researchers at Miggo Security documented a separate pre-authentication attack path tracked as CVE-2026-42208, chained with CVE-2026-42203 [6]. LiteLLM's proxy concatenated the `Authorization: Bearer` header value directly into SQL queries without parameterized binding. An unauthenticated attacker could inject SQL through the bearer token to extract virtual API keys and provider credentials from the LiteLLM database in a single request, then replay those credentials to access authenticated endpoints. When combined with CVE-2026-42203, this path also achieved RCE inside the LiteLLM container. Miggo's researchers reported that a functional exploit was available 36 hours and 7 minutes after their initial responsible disclosure [6].

PyPI Supply Chain Compromise: TeamPCP Campaign

On March 24, 2026, a threat actor self-identifying as TeamPCP published two malicious versions of LiteLLM to PyPI — v1.82.7 and v1.82.8 — after stealing the project's PyPI publishing credentials through a compromised Trivy security-scanning GitHub Action in LiteLLM's CI/CD pipeline [4][7]. Each poisoned package contained a malicious `.pth` file (`litellm_init.pth`) that Python's import machinery executes automatically at startup for every process in the environment where the package is installed.

The embedded payload operated in three stages. The first stage harvested environment credentials: cloud provider API keys, database connection strings, and any Kubernetes service account tokens present in the host environment. The second stage attempted lateral movement within Kubernetes clusters — reading secrets from all namespaces and attempting to schedule privileged pods on every node in `kube-system`, with each pod mounting the host filesystem. The third stage installed a persistent backdoor at `/root/.config/sysmon/sysmon.py` backed by a systemd user service

that polled for additional remote payloads [4]. PyPI quarantined the packages approximately 40 minutes after publication [10], representing a window during which the malicious packages were available for download and installation across the Python ecosystem [4].

Datadog Security Labs attributed the campaign to TeamPCP and identified LiteLLM as one of multiple AI and developer tooling packages targeted in the same operation [4]. Snyk's analysis [7] independently traced the initial pivot point to a poisoned version of the Trivy container scanner GitHub Action used in the LiteLLM repository's CI pipeline. BerriAI published a vendor security advisory on March 24, 2026 confirming the supply chain incident and advising affected users to treat all secrets present in the environment as compromised [10].

Compound Risk Profile

The clustering of these vulnerabilities reflects a structural pattern rather than independent misfortune: centralized credential broker architecture concentrates risk, and defenders should expect it to attract sustained researcher and adversary attention. LiteLLM instances that are internet-accessible, running with broad outbound network permissions, and holding AI provider keys with no usage limits represent a high concentration of AI operational risk in a single process — any single exploitation event could expose credentials spanning every model provider the organization uses. Successful exploitation of any of the documented attack chains may expose: OpenAI, Anthropic, Gemini, AWS, and Azure API keys; LiteLLM master keys and salt keys; database connection strings; and all logged prompts and responses, which may contain source code, personally identifiable information, internal documents, and operational secrets [2][3].

Recommendations

Immediate Actions

Organizations running LiteLLM should treat this as a critical-priority patching event aligned with CISA's June 22 remediation deadline. The first priority is upgrading LiteLLM to v1.83.14-stable or later and upgrading Starlette to v1.0.1 or later; the combined CVE-2026-42271 and CVE-2026-48710 chain is exploitable by any network-accessible attacker against unpatched instances. Organizations that installed LiteLLM versions 1.82.7 or 1.82.8 should treat all credentials and secrets present in the affected environment as fully compromised: these packages contained a persistent backdoor and must be quarantined immediately, with all credentials rotated and Kubernetes clusters inspected for unauthorized pods and service account modifications.

If immediate patching is not achievable, blocking `POST /mcp-rest/test/connection` and `POST /mcp-rest/test/tools/list` at the edge reverse proxy removes the exploitable surface for CVE-2026-42271. Network access to LiteLLM proxy ports should be restricted to explicitly trusted IP ranges, and no LiteLLM instance should be internet-accessible without authenticated, TLS-terminated reverse proxy protection in front of it. API keys for all model providers configured in the proxy – including OpenAI, Anthropic, Gemini, AWS Bedrock, and Azure OpenAI – should be rotated regardless of exploitation confidence, given the breadth of active threat activity.

Short-Term Mitigations

Organizations should audit LiteLLM deployments for internet exposure using asset discovery tools capable of identifying HTTP services on non-standard ports; runZero and similar platforms have published fingerprints for LiteLLM proxy detection [9]. LiteLLM instances should be deployed with the principle of least privilege: the proxy process should not run as root, should not have direct access to host network interfaces beyond what its routing function requires, and should not hold credentials beyond those strictly necessary for its configured routing table.

Monitoring should be enhanced for anomalous Host header values in proxy request logs, unexpected subprocess execution events originating from the LiteLLM process, and unusual API key usage patterns against model provider accounts – particularly keys making requests from IP addresses or at times inconsistent with normal application behavior. Organizations should also audit their Python CI/CD pipelines for use of third-party GitHub Actions that have access to PyPI publishing secrets, given the TeamPCP supply chain attack vector.

Strategic Considerations

The LiteLLM vulnerability series illustrates a structural concern that security teams should factor into AI architecture decisions: concentrating authentication credentials for all AI providers in a single proxy process creates a high-value target that combines the blast radius of a secrets management compromise with the operational criticality of a core application dependency. Organizations should evaluate whether their LiteLLM deployments benefit from secrets management integration (such as HashiCorp Vault or AWS Secrets Manager) rather than storing provider keys directly in environment variables or configuration files. Usage-limited, scoped API keys per downstream application – rather than shared master credentials – reduce the harm from any single credential exfiltration event.

The PyPI supply chain attack warrants a broader review of AI dependency governance. LiteLLM's CI/CD pipeline was compromised through a third-party GitHub Action, a vector that is prevalent across the open-source ecosystem and that organizations using AI libraries as build dependencies cannot audit at

scale. Pinning dependency versions to verified hashes, deploying private PyPI mirrors with pre-ingested, vetted packages, and monitoring for unexpected package version updates in automated AI infrastructure pipelines are practices that meaningfully reduce the supply chain attack surface available to campaigns like TeamPCP.

CSA Resource Alignment

The vulnerability pattern documented here maps directly to multiple domains in the CSA AI Controls Matrix (AICM). The MCP endpoint command injection (CVE-2026-42271) and the SQL injection chain (CVE-2026-42208) fall under AICM's AI Application Security domain, which addresses input validation, API exposure controls, and secure integration patterns for AI services. The privilege escalation chain (CVE-2026-47101, CVE-2026-47102) reflects failures in Identity and Access Management controls defined in both AICM and the CSA Cloud Controls Matrix (CCM), particularly controls governing least-privilege enforcement, field-level authorization, and separation of roles in multi-tenant API environments.

CSA's MAESTRO threat modeling framework for agentic AI systems identifies the AI gateway layer — including LiteLLM-style proxies — as a critical trust boundary between applications and model providers. MAESTRO's threat categories for orchestration layer compromise and credential harvesting at the gateway level are directly realized by this vulnerability series. Organizations applying MAESTRO to their AI architecture should treat any internet-accessible gateway process holding multi-provider credentials as a Tier 1 threat surface requiring the same hardening standards applied to identity providers and secrets management systems.

The PyPI supply chain attack aligns with CSA's guidance on AI supply chain security and the CCM's Supply Chain Management domain. CSA's published Zero Trust guidance is also relevant: a Zero Trust posture that treats internal AI infrastructure components — including LiteLLM proxy instances — as untrusted network participants requiring continuous authentication and authorization would have meaningfully constrained lateral movement in environments affected by the TeamPCP backdoor. The STAR (Security Trust Assurance and Risk) program provides a mechanism for organizations to assess and document their LiteLLM deployment posture against these controls for third-party assurance purposes.

References

[1] CISA. "Known Exploited Vulnerabilities Catalog: CVE-2026-42271." cisa.gov, June 8, 2026.

[2] The Hacker News. "LiteLLM Flaw CVE-2026-42271 Exploited in the Wild, Chains to Unauthenticated RCE." The Hacker News, June 2026.

[3] Obsidian Security. "Breaking LiteLLM: From Low-Privilege User to Admin and RCE (CVE-2026-47101, CVE-2026-47102, CVE-2026-40217)." Obsidian Security Blog, April 2026.

[4] Datadog Security Labs. "LiteLLM and Telnyx Compromised on PyPI: Tracing the TeamPCP Supply Chain Campaign." Datadog Security Labs, March 2026.

[5] BadHost Project. "BadHost — CVE-2026-48710: Starlette Host-Header Auth Bypass." badhost.org, 2026.

[6] Miggo Security. "RCE in LiteLLM (CVE-2026-42208): How Two Vulnerabilities and 36 Hours Turn an AI Gateway into a Backdoor." Miggo Security Blog, 2026.

[7] Snyk. "How a Poisoned Security Scanner Became the Key to Backdooring LiteLLM." Snyk Security Blog, March 2026.

[8] Rescana. "Critical LiteLLM Vulnerability Chain Enables Remote Code Execution and Full AI Gateway Server Takeover." Rescana, June 2026.

[9] runZero. "LiteLLM Proxy Vulnerabilities: How to Find Impacted Assets." runZero Research, June 2026.

[10] BerriAI / LiteLLM. "Security Update: Suspected Supply Chain Incident." LiteLLM Documentation Blog, March 2026.

[11] SOCRadar. "CISA KEV Highlights LiteLLM RCE (CVE-2026-42271) & Check Point VPN Auth Bypass (CVE-2026-50751)." SOCRadar Blog, June 2026.

