CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Malicious LLM Proxy Routers: Hidden AI Supply Chain Risk

How Compromised API Gateways Expose AI Workloads to Supply Chain Attack

Unofficial AI-assisted Research

2026-04-16

Key Takeaways

- LLM API proxy routers occupy a privileged application-layer man-in-the-middle position, able to read, rewrite, retain, or fabricate every prompt and response flowing between an organization's applications and upstream AI providers—positioning them as high-value targets for supply chain attackers, as demonstrated by the incidents described below.
- A peer-reviewed measurement study published in April 2026 analyzed 428 commodity LLM proxy routers and found 9 actively injecting malicious code and 17 abusing researcher-supplied credentials; researcher-deployed decoys attracted two billion billed tokens across 440 autonomous agent sessions over the measurement period [1].
- The March 2026 LiteLLM PyPI supply chain compromise—affecting approximately 40,000 downloads of backdoored versions 1.82.7 and 1.82.8 during a 40-minute initial quarantine window [2][13]—demonstrated that a single upstream CI/CD dependency can reach a library downloaded approximately 3 million times per day [10][13], turning any brief compromise window into a large-scale credential-harvesting event.
- Attack capabilities observed in production environments include simultaneous exfiltration of LLM provider API keys, cloud credentials (AWS, GCP, Azure), Kubernetes service account tokens, SSH keys, Git credentials, shell history, and cryptocurrency wallet files [2][4][10].
- Organizations should immediately audit their LLM proxy dependencies, pin package versions, rotate all credentials accessible from proxy environments, and formally classify their AI gateway layer as critical security infrastructure subject to the same controls as authentication and payment systems.

Background

Many enterprise AI application stacks insert one or more intermediary services between an application and an upstream provider such as OpenAI, Anthropic, or Azure OpenAI Service: LLM proxy routers and API gateways that handle routing, cost tracking, rate limiting, usage attribution, and multi-provider failover. Tools such as LiteLLM—downloaded approximately 3 million times per day [10][13]—the one-api/new-api family, and comparable open-source projects have seen rapid adoption as de-facto infrastructure components for AI deployments at scale. This intermediary layer delivers genuine

operational value—centralized logging, guardrail enforcement, and provider abstraction—but it also introduces an attack surface that, until recently, had received comparatively little formal security research—a gap the April 2026 measurement study [1] was among the first to address systematically.

By design, every LLM proxy router terminates the client-side TLS connection and opens a new upstream connection to the model provider. This places the router in an application-layer man-in-the-middle position: it can access the full plaintext of every prompt submitted to the model and every completion returned by it. In legitimate deployments, this visibility is intentional and necessary. In a compromised or maliciously operated router, that same architectural position enables silent credential harvesting, response manipulation, and downstream propagation of malicious payloads—often without any indication visible to the end application or its users, particularly in passive credential-harvesting scenarios where response content is not altered.

Two distinct, documented phenomena have emerged in 2026 to validate this threat model. First, a peer-reviewed measurement study demonstrated that commodity LLM routers available on public marketplaces and code-sharing platforms exhibit active malicious behaviors in the wild, with empirically verified instances of code injection, credential theft, and direct financial loss [1]. Second, the high-profile supply chain compromise of LiteLLM—the open-source proxy library downloaded approximately 3 million times per day—delivered credential-harvesting malware through the PyPI package ecosystem using a poisoned upstream CI/CD dependency [2][5][11]. Taken together, these incidents mark the LLM proxy layer as an active target category and a component of the AI supply chain deserving formal security classification.

Security Analysis

The Commodity Router Threat

Researchers published a systematic measurement study in April 2026 analyzing 428 LLM API routers drawn from commercial marketplaces and public repositories including GitHub and Taobao [1]. The study enumerated four primary attack classes. Active manipulation (AC-1) involves rewriting tool-call payloads —for instance, substituting a legitimate package installer URL with an attacker-controlled script, redirecting a downstream AI agent to execute arbitrary commands. Secret exfiltration (AC-2) involves passively harvesting API keys and credentials from plaintext traffic without altering any responses, making detection particularly challenging because downstream application behavior appears entirely normal. Evasion variants of both classes incorporate delay triggers and warm-up periods specifically designed to frustrate automated behavioral detection [1].

Of the 428 routers examined, 9 were actively injecting malicious code—notably including one router obtained from a paid commercial marketplace, not only free or community-sourced deployments. Seventeen routers abused researcher-supplied AWS credentials, and one router directly drained Ethereum from a researcher-controlled test wallet, confirming that financial theft is within the demonstrated capability set, not merely theoretical [1][7]. To measure opportunistic abuse patterns, the researchers deliberately leaked an OpenAI API key on Chinese-language developer forums and deployed 20 intentionally vulnerable proxy instances as honeypots. The resulting telemetry was significant: decoy infrastructure processed two billion billed tokens over the measurement period, recorded 99 leaked credentials across 440 autonomous agent sessions, and observed 401 sessions already operating in autonomous "YOLO" execution mode—agent configurations where tool calls are executed without human approval [1].

The study's technical findings also highlight a structural amplification factor. Open-source router templates—particularly the new-api and one-api projects—underpin a large share of commodity router deployments across both paid and free markets. Both templates have been pulled millions of times from public registries [1], and Chinese open-source models reached nearly 30 percent of total usage on OpenRouter during some measurement windows [1]. This concentration means that a vulnerability or malicious modification to a single upstream template propagates across a wide population of derived deployments, affecting organizations that believe they are running independent infrastructure.

The LiteLLM Supply Chain Compromise

The highest-profile supply chain incident affecting LLM infrastructure in early 2026 involved LiteLLM, the widely adopted open-source Python library for routing requests across LLM providers [2][5]. On March 24, 2026, the threat actor known as "TeamPCP" published two backdoored PyPI releases—versions 1.82.7 and 1.82.8—after compromising LiteLLM's PyPI publishing credentials. The initial access vector was a poisoned Trivy security scanner GitHub Action embedded in the project's CI/CD pipeline, which exfiltrated the PYPI_PUBLISH token from the GitHub Actions runner environment [2][11]. The compromised packages remained available on PyPI for approximately 40 minutes before being quarantined [2][13]; cached copies on mirrors and developer machines extended the effective exposure window further [10][11]. During that window, the packages were downloaded approximately 40,000 times [13].

The attack employed a stealthy persistence mechanism designed to survive library upgrades and uninstallation. Rather than embedding an obvious standalone executable, the attackers inserted a Python path configuration file—`litellm_init.pth`—into the package's `site-packages` directory. Because Python's interpreter automatically processes all `.pth` files at startup, the malicious code executed on every Python invocation in the affected environment, regardless of whether LiteLLM

itself was being actively called [2][10]. This meant that once installed, the malware persisted through LiteLLM upgrades and remained active even if the library was subsequently uninstalled without also clearing site-packages. The payload operated in multiple stages: harvesting environment variables, SSH keys, cloud provider credentials (AWS, GCP, and Azure), Kubernetes service account tokens, Git credentials, API keys, shell history, and cryptocurrency wallet files; encrypting the collected data; and exfiltrating it via HTTPS POST to the attacker-controlled domain `models.litellm.cloud` [2][5][10].

The breadth of data targeted reflects the unique privilege position of an LLM proxy in enterprise environments. Organizations running LiteLLM as a centralized AI gateway aggregate the API credentials of every upstream model provider in a single process environment. A credential stealer operating in that environment obtains simultaneous access to OpenAI, Anthropic, Azure OpenAI, and other provider keys in a single operation. When deployed in Kubernetes—a common production deployment pattern for containerized AI infrastructure—the same compromise yields lateral movement capability through harvested Kubernetes service account tokens, potentially enabling an attacker to pivot from the AI proxy into broader cloud infrastructure [5][6].

Structural Risk Factors

Several structural properties of the LLM proxy ecosystem elevate its risk profile beyond that of conventional API gateway software. LLM proxies are unusual in aggregating high-value credentials for multiple external services in a single process: they must hold valid API keys for every provider they route to, and in cloud-native deployments, they typically also hold cloud platform credentials for logging and storage. This secret concentration makes them high-value targets for credential-harvesting campaigns.

The ecosystem's rapid growth has not been matched by commensurate investment in supply chain security practices: the LiteLLM incident revealed that a widely deployed library's PyPI publishing credentials were accessible through a compromised CI/CD action, a basic security hygiene failure [11] that reflects a broader surge in supply chain attacks targeting developer toolchains in early 2026 [14]. Proxy deployments frequently appear in CI/CD pipelines and developer environments—contexts where production-grade hardening is less commonly applied. Dependency chains can extend through multiple tiers of tooling—as the LiteLLM incident demonstrated, an upstream CI/CD security scanner was itself the attack vector [11][12].

The man-in-the-middle position of the proxy layer creates risks beyond credential theft. A malicious or compromised proxy can conduct response injection attacks, silently modifying the AI output that downstream agents act upon. In agentic workflows where models execute tool calls autonomously, proxy-layer response substitution is logically equivalent to infrastructure-level indirect prompt injection: the agent receives what appears to be a trusted model response instructing it to take a harmful action.

Unlike prompt injection delivered through user-controlled data, infrastructure-level injection bypasses application-layer prompt sanitation controls entirely, making it particularly difficult to detect or prevent at the agent level [1][3].

Recommendations

Immediate Actions

Organizations running LiteLLM should immediately verify that their deployed versions are not 1.82.7 or 1.82.8. Any environment where these versions were installed should be treated as potentially compromised. Investigate for the presence of `litellm_init.pth` in Python `site-packages` directories and remove it if found. Audit outbound network connections for communication with `models.litellm.cloud` or `checkmarx.zone` [2], and search Kubernetes clusters for suspicious pod deployments matching the pattern `node-setup-*` in the `kube-system` namespace. As a precautionary measure, rotate all LLM provider API keys, cloud provider credentials, and Kubernetes service account tokens that were accessible from the proxy environment, and rebuild CI/CD runner images from clean baselines [2][6].

For organizations evaluating whether to conduct broader hygiene on commodity router infrastructure, the findings of the measurement study argue for auditing any LLM proxy router obtained from a marketplace, community repository, or third-party managed service, particularly those sourced from less-established vendors. The presence of malicious behavior in paid commercial offerings—not only free options—means that price or commercial status is not a reliable quality signal [1].

Short-Term Mitigations

Dependency pinning is the most immediately actionable structural defense. All LLM proxy libraries and their transitive dependencies should be version-pinned in `requirements.txt` or equivalent package manifests, and pip's `--require-hashes` flag should be used to verify package integrity cryptographically. Organizations should integrate software composition analysis (SCA) tooling and generate a software bill of materials (SBOM) for their AI infrastructure layer, subscribing to security advisories for each proxy library dependency [6]. The LiteLLM incident underscores that this must extend to CI/CD tooling as well: the attack chain began with a compromised Trivy scanner, not with LiteLLM itself.

LLM API keys and cloud credentials should be stored in dedicated secrets management platforms—such as HashiCorp Vault, AWS Secrets Manager, or equivalent—rather than as environment variables in container definitions or CI/CD runner configurations. Credential scope should enforce least-privilege: an LLM proxy requires API keys for upstream providers and little else. It does not require cloud storage write access, Kubernetes cluster administration rights, or access to SSH key material. Minimizing credential scope limits the blast radius substantially if the proxy process is compromised [6].

Network egress filtering for proxy processes deserves explicit attention. A legitimate LLM proxy communicates with a defined set of upstream model provider API endpoints. Strict egress policies that block all other outbound destinations would likely have impeded or blocked the exfiltration in this case, substantially limiting the impact of the compromise even after the malicious package was installed—provided the policy explicitly blocked the attacker's domain rather than permitting broad HTTPS to internet destinations [6].

Strategic Considerations

Organizations should formally reclassify their LLM proxy and gateway layer as critical security infrastructure, applying the same security standards they apply to authentication services, payment processors, and other high-value intermediaries. This means including LLM proxies in formal threat models, conducting dependency audits on the same cadence as application code reviews, and establishing incident response runbooks specific to AI gateway compromise. Given that the LiteLLM compromise achieved a 40,000-download blast radius in approximately 40 minutes, organizations should evaluate their detection and response capabilities specifically for supply chain attacks against AI infrastructure and establish monitoring baselines before incidents occur.

For organizations selecting commercial LLM proxy solutions, vendor security posture should be assessed as a primary criterion alongside capabilities and cost. Relevant evaluation criteria include the vendor's CI/CD signing and provenance attestation practices, whether the vendor publishes a current SBOM, the vendor's historical mean time to patch disclosed vulnerabilities, and their security disclosure process. The academic evidence that even paid commercial routers exhibit malicious behavior indicates that commercial availability alone does not guarantee security quality [1].

Over a longer horizon, the industry would benefit from cryptographic response envelope verification: a mechanism through which model providers sign their completions and tool-call outputs, enabling downstream agents to verify that a response genuinely originated from the claimed upstream provider and was not modified in transit by an intermediary. Researchers have proposed this as a structural mitigation for infrastructure-level injection attacks [1]. While this capability is not yet available in mainstream provider APIs, organizations building proprietary AI infrastructure should evaluate whether analogous integrity verification can be implemented at the proxy layer within their own control boundary.

CSA Resource Alignment

The threat landscape described in this note aligns closely with CSA's MAESTRO framework for agentic AI threat modeling [8][9]. MAESTRO's seven-layer architecture spans attack surfaces from Foundation Models through Ecosystem Integration, and the LLM proxy router sits at the Infrastructure layer—a component that, when compromised, enables attacks that cascade upward through the Agent Trust Boundaries and Tool and Resource Access layers simultaneously. The compound scenario described here—a compromised proxy silently rewriting tool-call responses received by an autonomous agent operating in YOLO mode—is precisely the class of cross-layer agentic threat that MAESTRO was designed to surface. Organizations performing MAESTRO-based threat modeling should explicitly enumerate LLM proxy and gateway components as discrete threat modeling targets within the Infrastructure and Ecosystem Integration layers.

The AI Controls Matrix (AICM) provides structured control guidance directly applicable to these findings [15]. AICM's supply chain security domain addresses third-party dependency risk, software provenance verification, and software composition analysis as required controls for AI application providers and orchestrated service providers. Version pinning, SBOM generation, secrets management, and egress filtering—the mitigations recommended above—map to AICM controls in the supply chain and data security domains. Organizations mapping their AI deployments to the AICM should ensure their proxy and gateway layer is explicitly in scope for supply chain security controls, not treated as background plumbing outside the control boundary.

CSA's Zero Trust guidance is also directly applicable here [16]. Zero Trust architecture holds that all components—including internal infrastructure services—should be treated as potentially compromised. Applied to AI infrastructure, this principle has concrete implications: LLM proxy responses should not be unconditionally trusted by downstream agents, credentials available to proxy processes should be scoped to the minimum necessary for their routing function, and network controls should verify that proxy processes communicate only with explicitly authorized upstream endpoints. The LiteLLM incident demonstrates concretely what happens when an infrastructure component is trusted implicitly rather than verified: a compromised dependency in the build pipeline translated directly into credential theft at scale.

References

[1] Arxiv. "Your Agent Is Mine: Measuring Malicious Intermediary Attacks on the LLM Supply Chain." arXiv, April 2026.

[2] LiteLLM Team. "Security Update: Suspected Supply Chain Incident." LiteLLM Blog, March 2026.

[3] Trend Micro Research. "Your AI Gateway Was a Backdoor: Inside the LiteLLM Supply Chain Compromise." Trend Micro, March 2026.

[4] Risky Business. "Malicious LLM proxy routers found in the wild." Risky Bulletin, April 2026.

[5] BleepingComputer. "Popular LiteLLM PyPI package backdoored to steal credentials, auth tokens." BleepingComputer, March 2026.

[6] Cycode. "Shedding The Lite: Unfolding The Dramatic Turn of Events with the LiteLLM Compromise." Cycode Blog, March 2026.

[7] OECD AI Observatory. "Malicious AI Routers Enable Cryptocurrency and Credential Theft." OECD.AI Incident Monitor, April 2026.

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

[9] Cloud Security Alliance. "MAESTRO for Real-World Agentic AI Threats." CSA Blog, February 2026.

[10] Sonatype. "Compromised litellm PyPI Package Delivers Multi-Stage Credential Stealer." Sonatype, March 2026.

[11] Kaspersky. "Trojanization of Trivy, Checkmarx, and LiteLLM solutions." Kaspersky Blog, March 2026.

[12] Security Boulevard. "The AI Supply Chain is Actually an API Supply Chain: Lessons from the LiteLLM Breach." Security Boulevard, April 2026.

[13] InfoQ. "PyPI Supply Chain Attack Compromises LiteLLM, Enabling the Exfiltration of Sensitive Information." InfoQ, March 2026.

[14] Zscaler ThreatLabz. "Supply Chain Attacks Surge in March 2026." Zscaler, March 2026.

[15] Cloud Security Alliance. "AI Controls Matrix (AICM)." CSA, 2025.

[16] Cloud Security Alliance. "Zero Trust Advancement Center." CSA, 2022.