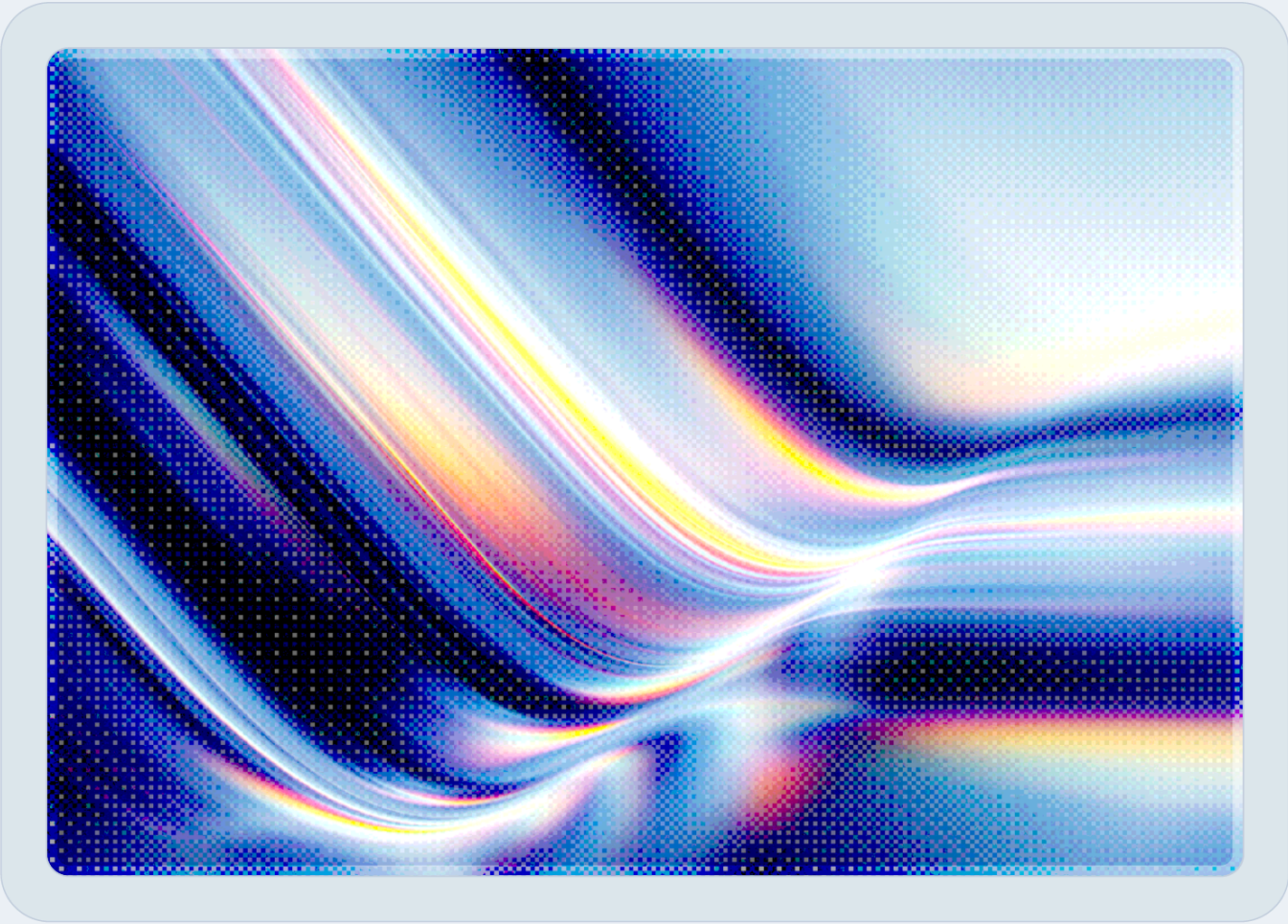
CSAI Foundation | Cloud Security Alliance

90 Days of Attacks on AI Infrastructure: A Honeypot View

2026-09-01

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Wiz Threat Research operated honeypots mimicking LiteLLM, Flowise, LangChain, Langflow, ChromaDB, Ollama, OpenWebUI, and Node-RED for 90 days and published the resulting telemetry on August 27, 2026, with the same campaign independently corroborated in subsequent security trade coverage [1][7]. The observation period captured three distinct attack patterns rather than opportunistic noise: remote code execution against internet-facing Model Context Protocol (MCP) servers, blind prompt injection against agent frameworks confirmed through out-of-band DNS callbacks, and post-exploitation tooling written specifically to interrogate the internal Python state of LiteLLM rather than search its disk for secrets [1]. The MCP-focused campaign chained an authentication bypass in LiteLLM's MCP Gateway (CVE-2026-59822) with a command-injection flaw in its MCP test endpoints (CVE-2026-42271), the same command-injection vulnerability CSA's rapid-research team analyzed in June alongside its chaining partner, the Starlette "BadHost" host-header bypass (CVE-2026-48710) [2] [3]. Attackers camouflaged cryptominer payloads inside directory names resembling legitimate Claude Code and Node-RED artifacts, and Wiz's baseline data indicates the exposure is broad: across the cloud environments Wiz surveyed for this report, 90 percent run self-hosted AI software, 81 percent run managed AI services, and 63 percent self-host models, meaning most organizations carry some version of this attack surface today [1]. The findings are consistent with a pattern CSA has tracked since June: AI gateways and agent frameworks are being treated by attackers as production-grade targets worth building bespoke tooling for — a gap this note's authors believe most defenders have not yet closed [2] [3][4].

Background

LiteLLM is an open-source proxy and SDK that lets organizations route requests to more than 100 large language model (LLM) providers through a single OpenAI-compatible interface, and it has become a widely used AI gateway component in enterprise AI stacks [3]. Because a gateway sits between application code and every downstream model provider, it typically concentrates provider API keys, routing secrets, and sometimes cloud or Kubernetes credentials in one place — a property CSA's prior analysis of the LiteLLM/Starlette vulnerability chain described as reclassifying the AI gateway tier into critical security infrastructure [3]. The Model Context Protocol (MCP), which LiteLLM and many agent

frameworks implement to let LLMs discover and invoke external tools, adds a second layer of exposure: MCP configuration and test endpoints are frequently designed for developer convenience rather than adversarial input, and several are reachable without authentication by default [2][3].

CSA has already documented two of the vulnerabilities central to the Wiz findings. CVE-2026-42271 is a command-injection flaw in LiteLLM's MCP test endpoints, present in versions 1.74.2 through 1.83.6 and fixed in 1.83.7, that lets an attacker pass a caller-controlled command directly into a subprocess call [2][3]. On its own it is rated CVSS 8.7 and requires an authenticated session; chained with CVE-2026-48710, a Starlette host-header validation bypass ("BadHost") that lets an attacker manipulate which authentication middleware a request is routed through, the combination becomes a fully unauthenticated remote-code-execution path rated CVSS 10.0 [2][3]. CISA added the chain to its Known Exploited Vulnerabilities catalog in June 2026 after confirming in-the-wild exploitation, and federal civilian agencies faced a June 22, 2026 remediation deadline under the applicable Binding Operational Directive, as CSA's prior analysis reported [2][3]. The Wiz honeypot telemetry adds a third component to this picture: CVE-2026-59822, an authentication bypass specific to LiteLLM's MCP Gateway OAuth2 fallback handling, in which a fabricated bearer token — in observed cases, a single character such as "x" — triggered a faulty fallback path and granted full access to MCP tooling without a valid LiteLLM key [1]. Read together, the two disclosures show attackers working through three separate authentication and input-validation gaps in LiteLLM's MCP surface across two disclosures over the same several-month window.

Security Analysis

The honeypot data organizes into three attack patterns, each reflecting a different level of attacker sophistication and a different point in the AI infrastructure stack. The first, and most directly tied to the LiteLLM vulnerability chain CSA has already flagged, involved attackers submitting fabricated MCP stdio configurations containing Python download-and-execute scripts [1]. These scripts fetched cryptomining binaries and staged them in directories designed to survive a cursory review: one payload landed in `/tmp/.dbus-cache/gmon`, mimicking a system service cache; another was placed at `/app/data/.claude/` and renamed "unicorn" so it would resemble a legitimate Claude Code artifact; a third was deployed to `/usr/src/node-red/xmrig`, embedding an XMRig Monero miner inside what looks like a normal Node-RED process tree [1]. In each case the malicious MCP configuration still returned a valid protocol handshake, so the compromised endpoint would likely appear to be functioning normally to automated monitoring and a human operator glancing at logs. This is consistent with CSA's earlier assessment that MCP's stdio transport model — which spawns local

subprocesses based on caller-supplied command, argument, and environment fields — creates a broad and largely unaudited execution surface whenever an endpoint accepts that configuration without an allowlist or sandbox [2][3].

The second pattern, blind prompt injection against LangChain, Flowise, OpenWebUI, and Node-RED deployments, targeted agent frameworks rather than the gateway layer directly [1]. Attackers injected prompts intended to trigger an agent's shell-execution or code-interpreter tool, then confirmed success not by reading command output — which they typically could not see — but by watching for an out-of-band DNS lookup to an attacker-controlled domain, with the requesting host's IP address encoded into the subdomain to correlate the callback with a specific target [1]. Payload delivery relied on fetching base64-encoded commands from Pastebin, a technique that defeats naive keyword filtering on the inbound prompt while keeping the actual payload off the wire until execution time [1]. This pattern does not depend on any single CVE; it exploits the general willingness of agent frameworks to execute instructions embedded in untrusted input, a risk CSA's MAESTRO threat-modeling framework for agentic systems treats as a foundational rather than incidental concern for any framework layer that grants an agent tool-calling privileges [5].

The third pattern stands out as the most distinctive from a defender's perspective: post-exploitation reconnaissance written with specific knowledge of LiteLLM's internals. Rather than searching the filesystem for configuration files or environment variables, attackers issued a Python one-liner — `import litellm; print('litellm.api_key:', getattr(litellm, 'api_key', None))` — to read the proxy's master key directly out of the running process's memory space [1]. Because LiteLLM's proxy consolidates credentials for every configured model provider, a single successful query of this kind can yield the equivalent of an entire organization's AI provider credential portfolio in one step, a concentration-of-credentials risk CSA's analysis of the LiteLLM/Starlette chain identified as the primary reason the gateway tier warrants IAM-grade scrutiny [3]. The technique signals that at least some fraction of the observed activity is not commodity malware repurposed against a new target, but tooling purpose-built against the AI stack's specific architecture.

The infrastructure supporting these campaigns showed some reuse and some specialization across the honeypot fleet. Wiz identified a malware download server at 185.62.1[.]8, a cryptominer command-and-control host at 185.84.98[.]85, and a Monero mining pool and proxy at pool.hashvault[.]pro and crazyeltonproxy[.]top, alongside separate staging infrastructure for the Langflow-targeting payloads (94.26.106[.]29, with drop paths at `/tmp/x86_64` and `/tmp/amd64`) and a compromised WordPress site at 1710.rwlp[.]be used to host binaries [1]. The presence of distinct toolchains for LiteLLM/MCP, Langflow, and Node-RED targets suggests either a single actor rotating techniques by target type or multiple actors converging on the same class of exposed infrastructure independently; the honeypot data alone cannot distinguish between the two.

Recommendations

Immediate Actions

Organizations running LiteLLM should confirm they are on version 1.83.7 or later with Starlette upgraded to 1.0.1 or later, and should separately verify their MCP Gateway configuration rejects malformed or single-character bearer tokens rather than falling back to an unauthenticated path — the specific failure mode behind CVE-2026-59822 [1][2]. Any MCP test or configuration endpoint that accepts stdio command definitions from a caller should be restricted to an administrative role and removed from public network reachability immediately; this is the same remediation CSA prescribed for the CVE-2026-42271/CVE-2026-48710 chain and it applies equally to the newly reported authentication-bypass path [2][3]. Because attackers demonstrated the ability to read LiteLLM's master key from process memory, organizations that cannot rule out exposure during this window should treat all proxy-issued provider API keys as potentially disclosed and rotate them rather than relying on log review alone [1][3].

Short-Term Mitigations

Defenders should inventory every AI framework in their environment — not only LiteLLM, but Flowise, LangChain, Langflow, ChromaDB, Ollama, OpenWebUI, and Node-RED — and confirm each requires authentication by default, since Wiz's honeypots specifically targeted the common case where these tools ship open [1]. Egress filtering and narrow IAM scoping around AI serving infrastructure would blunt both the cryptomining payloads and the DNS-callback confirmation technique used in the blind prompt injection campaign, since both rely on the compromised host reaching attacker-controlled infrastructure outbound [1]. Runtime monitoring should specifically watch for anomalous process ancestry under AI framework directories — a Python process spawning a downloader, or a miner process nested under a Node.js or Claude Code-styled path — since the campaign's camouflage strategy depended on blending into expected process trees rather than hiding entirely [1].

Strategic Considerations

The honeypot findings support treating AI infrastructure — gateways, agent orchestration frameworks, and vector databases alike — as production systems subject to the same patch cadence, network segmentation, and identity controls as any other credential-holding service, rather than as developer tooling exempted from standard hardening [1][3]. Given that CISA added the LiteLLM/Starlette chain to its Known Exploited Vulnerabilities catalog only after confirming active exploitation, and that Wiz's own telemetry shows continued attacker interest in the same product months later, organizations maintaining

open-source AI infrastructure should assume that any publicly disclosed vulnerability in a widely deployed component will be weaponized quickly enough that patching should proceed on the assumption exploitation is already underway [1][2].

CSA Resource Alignment

This research connects most directly to two CSA rapid-research publications analyzing the same underlying LiteLLM vulnerability chain the Wiz honeypots observed being exploited in the wild. **LiteLLM AI Gateway: Active Exploitation via MCP Injection** documents CVE-2026-42271 and its chaining with the Starlette BadHost bypass (CVE-2026-48710), the same command-injection and authentication-bypass mechanics underlying the MCP RCE pattern in the honeypot data, and provides the patch versions (LiteLLM 1.83.7, Starlette 1.0.1) referenced in this note's recommendations [2]. **LiteLLM AI Gateway: KEV-Listed Attack Chain Enables Full Takeover** generalizes that chain into an architectural pattern across the Python ASGI/AI middleware ecosystem and argues for reclassifying the AI gateway tier as critical security infrastructure — the framing this note applies to the newly reported CVE-2026-59822 authentication bypass in LiteLLM's MCP Gateway [3].

The credential-theft dimension of the honeypot findings — attackers querying LiteLLM's Python module state to extract provider API keys directly from memory — extends the concentration-of-credentials risk described in **LLMjacking Evolved: Stolen AI Compute as Offensive Infrastructure**, which documented the first observed use of hijacked AI compute as the reasoning engine for an autonomous offensive pipeline and recommended the same API-layer authentication, key rotation, and least-privilege scoping controls this note applies to the honeypot's MCP and blind-injection findings [4]. The blind prompt injection pattern observed against LangChain, Flowise, OpenWebUI, and Node-RED falls within the threat space mapped by CSA's **MAESTRO Agentic AI Threat Modeling Framework**, whose layered model treats an agent's tool-calling capability as a threat surface independent of any single implementation's vulnerabilities, consistent with the honeypot's finding that this campaign required no CVE at all [5]. Finally, the broad set of unauthenticated-by-default AI services identified across the honeypot fleet aligns with the identity and access management domain of the **AI Controls Matrix (AICM) v1.1**, CSA's control framework for AI systems, which this note recommends organizations use as an audit baseline when inventorying self-hosted AI infrastructure for default-open configurations [6].

References

[1] Wiz Threat Research. "Attacks on AI Infrastructure: 90-Day Honeypot Telemetry." Wiz Blog, August 27, 2026.

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

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

[4] Cloud Security Alliance. "LLMjacking Evolved: Stolen AI Compute as Offensive Infrastructure." CSA Labs, June 20, 2026.

[5] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 6, 2025.

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

[7] GBHackers. "Attackers Exploit MCP RCE, Blind Prompt Injection and Memory Credential Theft Against AI Infrastructure." GBHackers, August 2026.