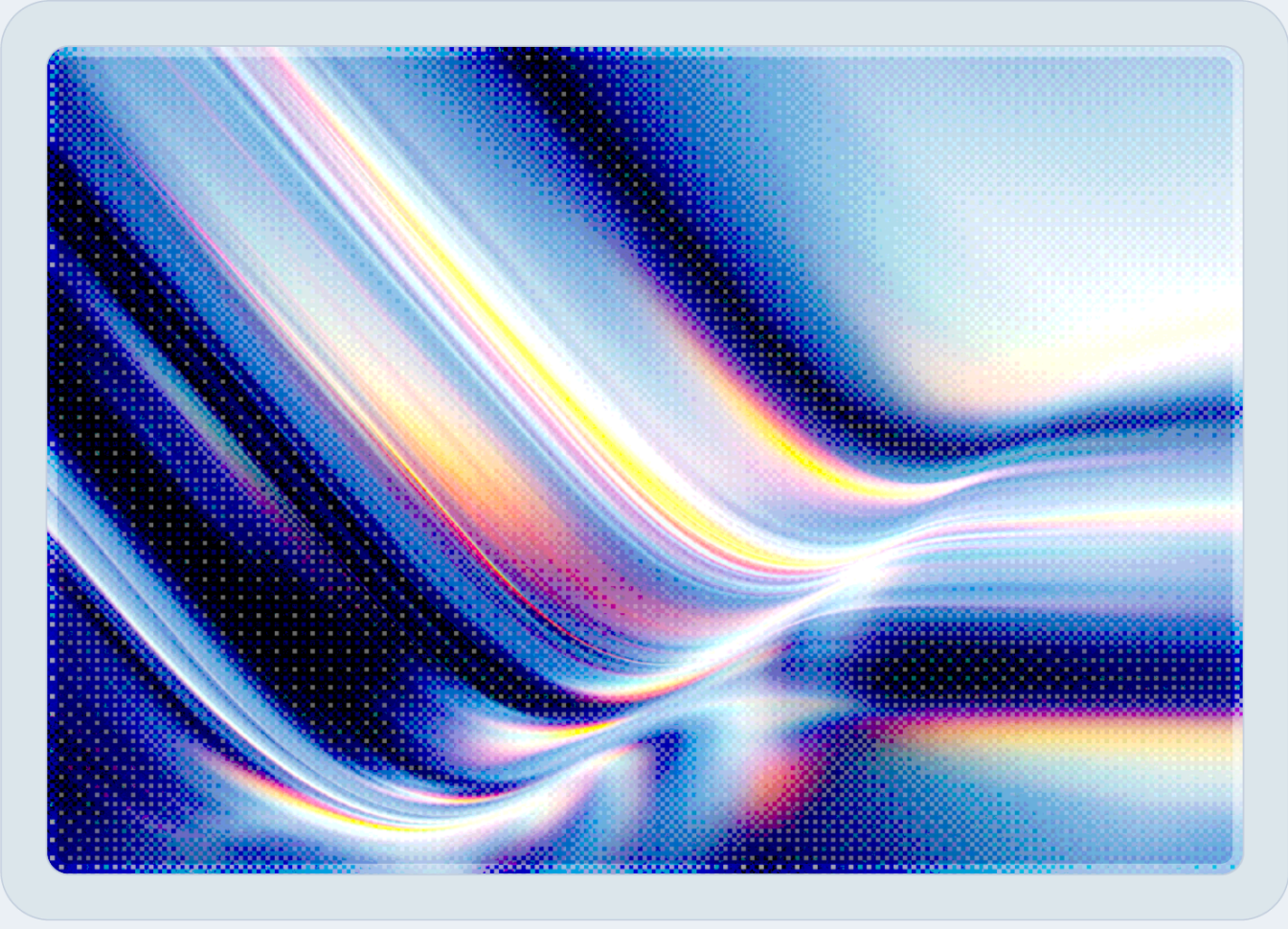
CSAI Foundation | Cloud Security Alliance

After TeamPCP: Patched LiteLLM Gateways Remain Pivot Points

Arrests Disrupt a Crew but Not the Stolen Credentials or Post-Patch Gateway Abuse They Left Behind

2026-08-29

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- Australian Federal Police arrested two alleged members of the TeamPCP hacking collective on August 27, 2026, following a campaign that compromised the LiteLLM AI gateway's software supply chain and touched thousands of downstream organizations [1][2].
- The arrests close a chapter on the identified perpetrators but do nothing to invalidate the credentials, API keys, and cloud tokens TeamPCP's campaign already exfiltrated; the FBI has warned that harvested credentials are likely to be weaponized long after the original intrusion [6].
- A technique published independently on August 3, 2026 shows that a fully patched, up-to-date LiteLLM gateway can still be hijacked for traffic interception, key theft, and tool-call injection — provided an attacker holds a valid admin or master-key credential, which is exactly the class of secret TeamPCP's operation was built to harvest [3].
- Organizations that patched LiteLLM's 2026 CVE chain and considered the incident closed should treat that as necessary, not sufficient; credential rotation, configuration-change monitoring, and outbound-traffic controls matter as much as version currency.

Background

On August 27, 2026, the Australian Federal Police arrested two Western Australian men — 21-year-old Ruben Ian Thomson of Cottesloe, who used the handle "Ellis," and 23-year-old Michael Gaebler — on a combined fourteen cybercrime charges, and both were denied bail [1][2]. Investigators and researchers link the pair to TeamPCP, a loosely organized cybercriminal collective that coordinated through a Matrix chat server known as "Cybercats" rather than operating as a hierarchical crime syndicate. Journalist Brian Krebs, who identified Thomson's real identity in June 2026, reported that TeamPCP functioned as a peer network of skilled individuals collaborating on software supply chain intrusions, with Thomson claiming to have led the group until March 2026 [1]. TechCrunch's reporting on the arrests lists a victim roster that spans an unusually wide cross-section of the technology and AI economy: Mercor, OpenAI, GitHub, European Commission cloud infrastructure, and organizations dependent on the Trivy vulnerability scanner and the LiteLLM AI gateway [2].

TeamPCP's signature technique was not a single exploit but a self-propagating supply chain worm known as Shai-Hulud, which harvested developer credentials from GitHub and npm and then used those credentials to inject malicious code into widely trusted open-source packages, creating a cycle in which each newly poisoned package compromised the next set of developer machines that installed it [1]. The most consequential single episode of that campaign targeted LiteLLM directly. In March 2026, TeamPCP obtained a leaked automation token belonging to Trivy, the widely used open-source vulnerability scanner; the token had reportedly been rotated at some point but not fully revoked, leaving it usable for an attacker who had already captured it. That access let TeamPCP inject malicious code into Trivy's build pipeline, which in turn fed into LiteLLM's own continuous integration process, resulting in two poisoned releases — versions 1.82.7 and 1.82.8 — published to PyPI [4]. The malicious payload rode in on a `.pth` file, a mechanism that executes automatically at Python interpreter startup and is consequently able to bypass the `--ignore-scripts` protections many teams rely on to block malicious install-time code [4].

The scale of that single compromise arguably elevates it from an isolated incident to a structural concern. CloudSEK's reconstruction of the exposure, published August 11, 2026, identified more than 2,500 companies and roughly 434,000 CI/CD pipelines as potentially affected, even though the poisoned packages were live on PyPI for only about forty minutes before quarantine [4]. High-confidence victims identified in that dataset include AWS, Samsung, Cisco, FedEx, Deloitte, X Corp, and Vodafone, alongside additional organizations across the automotive, industrial, and pharmaceutical sectors that CloudSEK's dataset names at high confidence, including Volkswagen, Siemens, Robert Bosch, Airbus, John Deere, Roche, ServiceNow, Zscaler, Epic Games, and NGINX [4]. What the compromised build pipeline actually stole was not limited to source code: cloud provider keys for AWS, GCP, and Azure; SSH keys and repository tokens; Kubernetes secrets; AI provider API keys and gateway credentials; environment variables and other CI/CD secrets; and even package-publishing credentials that could be used to continue the propagation cycle [4]. TeamPCP encrypted the stolen material with AES-256 under a hardcoded RSA-4096 key before exfiltrating it, a level of operational tradecraft that sits oddly alongside the personal-security lapses — reusing cybercrime handles in a registered company name, reusing passwords across accounts, posting identifying details on social media — that researcher Charlie Eriksen of Aikido Security said ultimately unraveled the group: "They can be noisy, they can make mistakes. They can leave evidence everywhere. They can take risks that a professional criminal group would consider completely unacceptable" [1].

The FBI's Internet Crime Complaint Center issued a FLASH advisory on July 2, 2026 (FLASH-20260702-01) describing TeamPCP's campaign as extending beyond Trivy and LiteLLM to include the KICS infrastructure-as-code scanner and the Telnyx Python SDK [6]. Independent reporting on that advisory identified the group's toolkit as comprising four tracked components — CanisterWorm, SANDCLOCK, Mini Shai-Hulud, and Miasma — and put the campaign's total harvest at more than

500,000 credentials pulled from over 10,000 CI/CD pipelines [11][12]. Crucially, the FBI's warning was not backward-looking: it stated that affiliated actors are likely to weaponize the harvested credentials long after the original intrusion, independent of whether the initially compromised software has since been patched [6]. That warning is the hinge on which this research note turns. The August 27 arrests are a genuine law enforcement success, but they attach names to a supply chain campaign whose exhaust — potentially hundreds of thousands of still-valid credentials — is likely circulating well beyond whatever TeamPCP members remain at large.

Security Analysis

LiteLLM's own 2026 vulnerability history compounds the credential problem described above. CISA added CVE-2026-42271, a command-injection flaw in LiteLLM's Model Context Protocol test endpoints, to its Known Exploited Vulnerabilities catalog on June 8, 2026, after confirming active in-the-wild exploitation, and set a remediation deadline of June 22, 2026 for federal civilian agencies [5]. On its own the flaw carries a CVSS score of 8.7 and requires an authenticated proxy API key; researchers demonstrated that chaining it with CVE-2026-48710, a Host-header authentication bypass in the Starlette framework nicknamed "BadHost," removes the authentication requirement entirely and raises the practical severity to a maximum CVSS 10.0 [5]. Organizations that have since upgraded to patched LiteLLM and Starlette releases have correctly closed that specific exploitation path. What they have not necessarily closed is the exposure created by credentials that left their environment before the patch was applied — through the March supply chain compromise, through the SQL injection vulnerability tracked as CVE-2026-42208 [13], through the privilege-escalation chain comprising CVE-2026-47101, CVE-2026-47102, and CVE-2026-40217 [14], or through ordinary secrets-hygiene failures unrelated to any CVE.

A technique published on August 3, 2026 by the security research blog Embrace The Red illustrates exactly how that residual exposure gets exploited, and it does so without touching a single unpatched vulnerability [3]. The prerequisite is possession of a LiteLLM proxy's master key or an equivalent admin-scoped credential — precisely the kind of secret TeamPCP's campaign was designed to harvest at scale, and precisely the kind of secret that source-control leaks, weak authentication, or a prior undetected compromise can also produce. With that credential in hand, an attacker calls LiteLLM's legitimate `/model/update` administrative API to change two configuration values: the `api_base` setting, which they repoint to a server they control, and `use_litellm_proxy`, which they set to true. Because this is a supported configuration change rather than a bug, it triggers no alert in most default deployments. The victim organization's client-side credentials and application code remain completely

unchanged, but every subsequent request is now silently routed through the attacker's infrastructure — a transparent, persistent man-in-the-middle position established through the gateway's own administrative surface [3].

Two further capabilities follow directly from that positioning. First, because LiteLLM resolves the actual upstream provider credentials and attaches them to the `Authorization` header of every rerouted request, the attacker's gateway can capture those resolved API keys using a custom authentication hook, then provision the stolen keys on infrastructure of its own choosing — achieving durable, independent access to the victim's OpenAI, Anthropic, or other model-provider accounts even after the redirect is eventually discovered and reversed [3]. Second, the attacker can install a post-call hook that modifies inference responses after they return from the model but before they reach the client, allowing it to inject forged tool-calls that appear indistinguishable from legitimate model output. Because this injection happens below the layer where most prompt-injection defenses operate, it can reach AI agent clients directly; in deployments where agents are configured to execute tool calls with limited human review, a forged tool call can translate into unauthorized local command execution [3]. The table below summarizes how the distinct risk vectors in LiteLLM's 2026 history relate to one another and to patch status.

Risk Vector	Prerequisite	Status as of August 2026	Residual Risk After Patching
CVE-2026-42271 + CVE-2026-48710 chain (unauthenticated RCE)	Network reachability to an unpatched proxy	Patched in LiteLLM ≥1.83.7 and Starlette ≥1.0.1; KEV-listed and federally mandated for remediation [5]. A separate privilege-escalation chain (CVE-2026-47101/47102/40217) was not closed until the later ≥1.83.14-stable release [14].	Eliminated once both components are upgraded to ≥1.83.14-stable
March 2026 PyPI supply chain compromise (poisoned 1.82.7/1.82.8)	Installation of the two malicious package versions during	Malicious versions quarantined; TeamPCP's broader Shai-Hulud campaign disrupted by the August 2026 arrests [1][4]	Any credentials exposed during the compromised install window remain valid until explicitly rotated

Risk Vector	Prerequisite	Status as of August 2026	Residual Risk After Patching
	a ~40-minute window		
Admin-credential gateway hijack (traffic interception, key theft, tool-call injection)	Possession of a valid master key or admin-scoped credential, obtained by any means	No CVE exists because the technique abuses intended API functionality [3]	Persists indefinitely for any organization that has not rotated credentials and does not monitor `api_base`/routing changes

The pattern across all three rows is the same: LiteLLM's value as an AI gateway comes from centralizing credentials for dozens of model providers behind a single administrative surface, and that centralization is precisely what makes any successful compromise of that surface — whether through an unpatched CVE, a poisoned dependency, or a leaked master key — disproportionately damaging. Patching addresses the first row. Nothing short of comprehensive credential rotation and configuration-change monitoring addresses the second and third.

Recommendations

Immediate Actions

Organizations running LiteLLM should confirm they are on a version at or above 1.83.14-stable with Starlette at or above 1.0.1. The MCP command-injection chain (CVE-2026-42271/CVE-2026-48710) was fixed in 1.83.7, but a separate privilege-escalation chain (CVE-2026-47101, CVE-2026-47102, and CVE-2026-40217) was not fully closed until 1.83.14-stable, so 1.83.7 alone is not a sufficient upgrade target [14]. Any instance that was ever running 1.82.7 or 1.82.8 — the two versions poisoned during the March 2026 supply chain window — should be treated as compromised pending forensic review, rather than simply upgraded past. Every provider API key that a LiteLLM instance has ever held should be rotated, not just keys associated with recently discovered anomalies, because the March compromise's exfiltration scope is not fully knowable from the victim side. Teams should also audit their proxy's current `api_base` and routing configuration for any value that does not match an expected, allow-listed upstream endpoint — the hijack technique described above leaves exactly that kind of artifact behind.

Short-Term Mitigations

Configuration-change monitoring deserves the same priority as vulnerability patching going forward: alerts should fire on any modification to `api_base`, `use_litellm_proxy`, or other routing-affecting settings made through the `/model/update` endpoint or equivalent admin APIs, and those alerts should route to the same on-call rotation that handles CVE-driven incidents. Master keys and other admin-scoped credentials should be stored in a dedicated secrets manager with short lifetimes and automatic rotation rather than long-lived environment variables, and outbound network policy should restrict a LiteLLM proxy's ability to establish new upstream connections to a defined allow-list of provider endpoints, which would have blocked the redirect technique regardless of credential compromise. Dependency pinning to verified hashes and use of private package mirrors reduce exposure to a repeat of the Trivy-to-LiteLLM compromise pathway.

Strategic Considerations

The recurring theme across LiteLLM's 2026 incident history — and across the TeamPCP campaign more broadly — is that AI gateways function as credential brokers with a blast radius comparable to identity infrastructure, and security programs should govern them accordingly rather than treating them as ordinary developer tooling. That means including AI gateways explicitly in incident response playbooks, vulnerability management SLAs, and vendor risk assessments, and it means recognizing that a law enforcement action against the operators of a supply chain campaign, however welcome, does not retire the credentials that campaign already stole. Organizations should assume that some fraction of the roughly 500,000 credentials attributed to TeamPCP's toolkit remain usable and will surface in unrelated intrusions for months or years to come [2][12].

CSA Resource Alignment

This incident sits squarely within a body of CSA research the AI Safety Initiative has already published on LiteLLM specifically, including two prior notes that examined the same post-patch, admin-credential hijack technique before this one was written. "LLM Heist: Abusing LiteLLM Callback Hooks Post-Compromise," published August 5, 2026, was the AI Safety Initiative's first treatment of the Embrace The Red disclosure, framing the `/model/update` and callback-hook technique as "a control-plane risk rather than a code-execution bug" [15]. "LLM Heist: Post-Compromise Abuse of LiteLLM Gateways," published August 20, 2026 — one week before the TeamPCP arrests — extended that analysis across the full traffic-rerouting, credential-harvesting, and tool-call-injection sequence this note also describes [16]. What this note adds to that existing coverage is the arrest timeline: it ties the hijack

technique's credential prerequisite directly to the large-scale credential-harvesting campaign — TeamPCP's Shai-Hulud worm and its LiteLLM supply chain compromise — that the August 27 arrests only partially disrupt, and argues that CSA's vulnerability-chain research and its post-compromise hijack research describe the same underlying risk from opposite ends.

Earlier CSA research on LiteLLM's CVE history remains directly relevant to that argument. "LiteLLM AI Gateway: KEV-Listed Attack Chain Enables Full Takeover" consolidates the CVE-2026-42271/CVE-2026-48710 chain and the March 2026 PyPI supply chain compromise into a single advisory, and its description of a LiteLLM proxy as routinely holding "authentication credentials for an organization's entire AI provider ecosystem" is the same structural argument this note extends to the post-patch hijack scenario [7]. "LiteLLM AI Gateway: Critical Vulnerability Chain Exposes API Keys" documents how LiteLLM's proxy-mode architecture aggregates high-value provider credentials in a way that makes any single compromise disproportionately damaging [8]. "LiteLLM AI Gateway: Active Exploitation via MCP Injection" provides the technical detail on the CVE-2026-42271/CVE-2026-48710 chain and maps it to Layers 1 and 2 of CSA's MAESTRO framework for agentic AI threat modeling, which is directly relevant given that the tool-call injection capability in the hijack technique targets agentic clients specifically [9]. Read together, these five prior CSA publications and this note converge on the same conclusion: LiteLLM's credential-aggregation architecture is the persistent risk, and CVE remediation addresses only part of it.

More broadly, the credential-rotation, secrets-management, and configuration-monitoring controls recommended here map to the Identity and Access Management and Application and Interface Security domains of CSA's AI Controls Matrix (AICM v1.1). AICM builds on and extends CSA's Cloud Controls Matrix with AI-specific control guidance, and organizations assessing LiteLLM or an equivalent AI gateway deployment should treat AICM as the more current baseline for that evaluation [10].

References

[1] Brian Krebs. "Two Alleged 'TeamPCP' Hackers Arrested in Australia." Krebs on Security, August 27, 2026.

[2] TechCrunch. "Australian police arrest two over TeamPCP hacks targeting Mercor, OpenAI, and others." TechCrunch, August 27, 2026.

[3] Embrace The Red. "LLM Heist: Hijacking LiteLLM for Traffic Interception, Key Theft, and Tool-Call Injection." Embrace The Red, August 3, 2026.

[4] CloudSEK. "LiteLLM Supply Chain Attack: 2,500+ Companies Exposed in the Largest AI Supply Chain Breach of 2026." CloudSEK, August 11, 2026.

[5] CISA. "CISA Adds Two Known Exploited Vulnerabilities to Catalog." Cybersecurity and Infrastructure Security Agency, June 8, 2026.

[6] CloudSEK. "FBI FLASH-20260702-01 Explained: The AI Supply Chain Advisory and Who It Applies To." CloudSEK, July 2026.

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

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

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

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

[11] Pierluigi Paganini. "FBI: TeamPCP Compromised Dev Tools to Steal Cloud Credentials." Security Affairs, July 4, 2026.

[12] Cybernews. "Over 1,000 firms hit by TeamPCP's supply-chain attacks – FBI took notice." Cybernews, July 3, 2026.

[13] The Hacker News. "LiteLLM CVE-2026-42208 SQL Injection Exploited within 36 Hours of Disclosure." The Hacker News, April 29, 2026.

[14] Obsidian Security. "Breaking LiteLLM: From Low-Privilege User to Admin and RCE." Obsidian Security, June 11, 2026.

[15] Cloud Security Alliance. "LLM Heist: Abusing LiteLLM Callback Hooks Post-Compromise." Cloud Security Alliance, August 5, 2026.

[16] Cloud Security Alliance. "LLM Heist: Post-Compromise Abuse of LiteLLM Gateways." Cloud Security Alliance, August 20, 2026.

