

When Your AI Gateway Hands Over the Keys

Investigating the LiteLLM supply chain compromise, its cascading impact on downstream organizations, and what it means for vendor monitoring.

EDITION 01

8 MIN READ

SUPPLY CHAIN / AI INFRASTRUCTURE / TPRM

Exposure Window

~3 HRS

Daily Downloads

3.4M+

Attack Vector

SUPPLY CHAIN

Who is

**LiteLLM** ?

LiteLLM is a Y Combinator-backed open-source Python library that gives developers a standardized way to call over 100 LLM providers through one consistent API format. Think of it as a universal adapter for AI models.

This utility has made it extremely popular. **The package sees over 3.4 million downloads per day. According to Wiz Research, LiteLLM is present in 36% of monitored cloud environments.** Organizations deploy it on everything from local developer laptops to production AI gateways, where it routes requests, manages API keys, and tracks spend. Companies as large as Okta use LiteLLM as the backbone of their internal AI gateway.

WHY THIS MATTERS FOR THIRD-PARTY RISK

LiteLLM sits in a uniquely privileged position. Because it routes traffic between applications and LLM providers, a single deployment typically holds API credentials for multiple cloud services at once. A compromise here does not yield one key. It yields the entire keyring.



What Happened

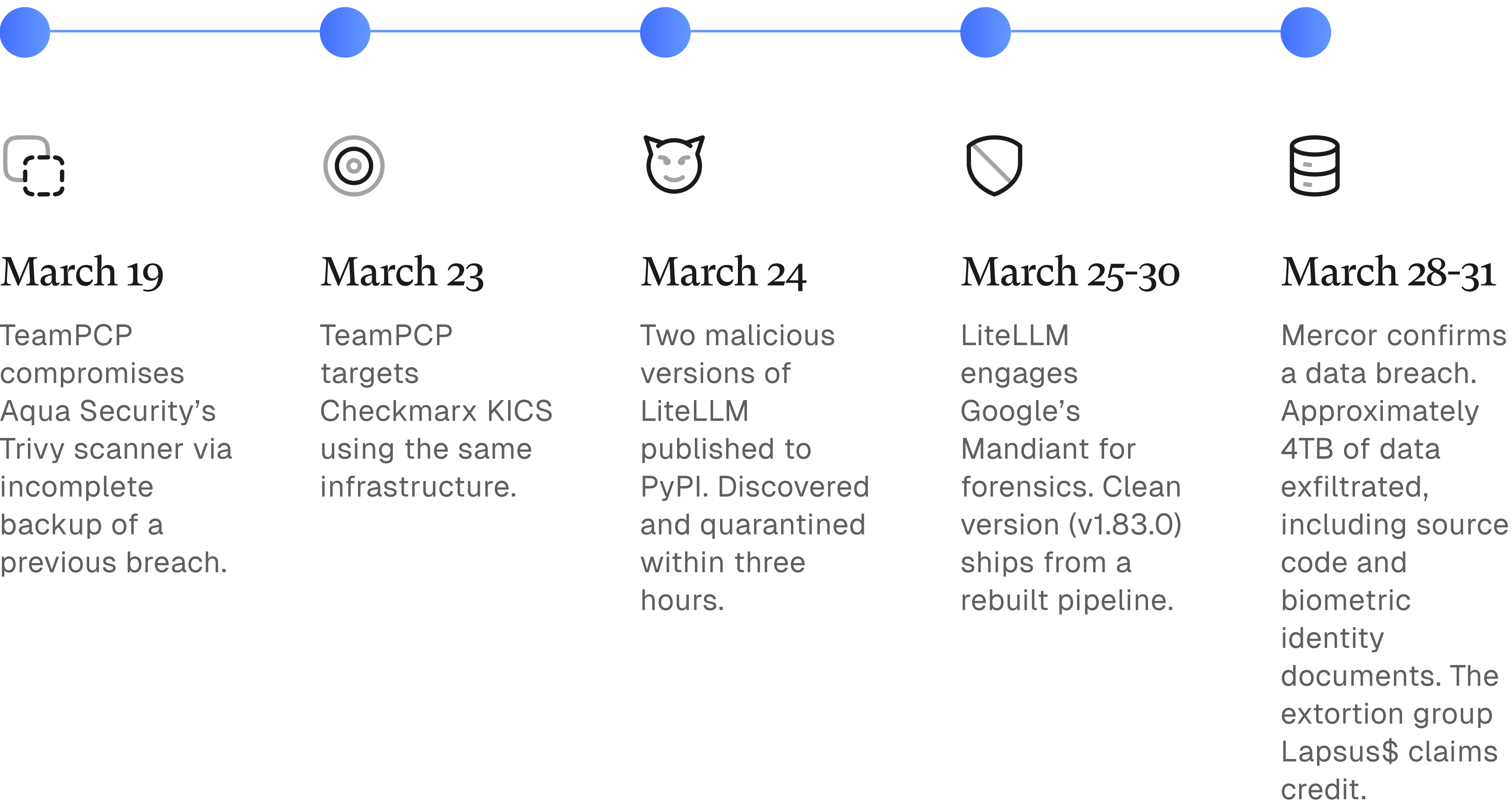
On March 24, 2026, two malicious versions of the LiteLLM package were published to PyPI, the main repository where developers download Python software.

The malicious versions contained credential-stealing code designed to harvest API keys, cloud credentials, SSH keys, and database passwords from any machine that installed them. This was not a direct hack of LiteLLM. It was the third step of a cascading supply chain attack.

A threat actor group called TeamPCP first compromised Trivy, a popular security scanning tool made by Aqua Security. LiteLLM's build pipeline used Trivy for security checks, and the compromised version of Trivy stole LiteLLM's publishing credentials. With those credentials, the attackers uploaded poisoned versions of LiteLLM directly to PyPI, bypassing the normal release process entirely. The malicious packages were live for roughly three hours before PyPI quarantined them. A security researcher happened to catch it when the malware crashed his machine due to a bug in the attack code. Had the attackers written cleaner code, it may have gone undetected for much longer.



Key Dates





Scope of Impact

Given LiteLLM's download volume and the fact that it shows up in over a third of monitored cloud environments, even a three-hour exposure window creates a significant blast radius. The malware targeted every credential it could find on an infected machine: cloud platform keys, database passwords, API tokens, SSH keys, and more. For TPRM teams, the more important question is what happens after the initial compromise. In this case, the answer is: a lot.

Nth-Party Ripple Effects

The compromised vendor

Stolen publishing credentials and distribution of two malicious versions force a rebuild of the entire release pipeline from scratch and an audit of every prior release, as well as a compliance provider rip & replace. LiteLLM's remediation was relatively fast (clean version shipped within a week), but the damage to downstream consumers was already done.

3RD PARTY

Your vendors who use LiteLLM

Any vendor in your portfolio that auto-updated to the compromised versions was immediately exposed. Because LiteLLM can hold API keys for every LLM provider the org uses, a single compromised install could expose credentials across their entire AI infrastructure.

The challenge: most TPRM programs have no visibility into whether a vendor runs LiteLLM, since it is an open-source dependency that would not typically appear in a SOC 2 report or a standard questionnaire response.

4TH PARTY

You, as a customer of those vendors

Mercor confirmed a breach affecting Slack data, ticketing systems, source code, and biometric identity documents. They work with clients including OpenAI and Anthropic for AI training contracting, meaning data from those engagements was potentially in scope.

If you are a Mercor customer, you now have an incident to manage even though you have no direct relationship with LiteLLM. Stolen credentials have reportedly been shared with Lapsus\$ and a ransomware group called Vect, meaning organizations whose cloud environments were accessed before rotation may still be exposed weeks later.

THE TRANSITIVE DEPENDENCY PROBLEM

The researcher who first caught the attack was not even using LiteLLM directly. He was testing a plugin that pulled in LiteLLM as an indirect dependency. You do not have to choose a vendor for it to become your risk. Most organizations have no visibility into whether a tool like this exists anywhere in their vendor's stack.



Remediation

LiteLLM paused all releases, engaged Mandiant for forensics, audited every prior release against known indicators of compromise, and rebuilt their CI/CD pipeline from scratch. A clean version shipped on March 30. For affected organizations, the guidance was straightforward: remove the compromised versions, search systems for persistence files the malware left behind, block outbound traffic to the known command-and-control domains, and rotate every credential on any system that may have been exposed.

All of them. Immediately. Wiz's analysis found that organizations that rotated credentials quickly were able to contain the damage. Organizations that waited are likely still exposed.

THE ROOT CAUSE LESSON

The entire chain started because Aqua Security's remediation of a previous breach was incomplete. Credential rotation was not done all at once, and the attackers used a still-valid token to steal newly rotated secrets during the transition. Incomplete incident response does not just fail to contain one breach. It creates the conditions for the next one.



How Could You Have Prepared?

If you were a customer of an organization running LiteLLM, or a customer of one of their customers, what are you supposed to do to prevent exposure? Great question.

1

Require SBOMs or dependency disclosures from critical vendors

Most TPRM programs have no mechanism to discover whether a vendor depends on a specific open-source package. A software bill of materials (SBOM) for vendors classified as critical or high-risk, especially those processing sensitive data or integrating with your infrastructure, would have let teams identify LiteLLM exposure within hours of the disclosure instead of waiting for a vendor to self-report.

If a full SBOM is not realistic, even a targeted question in your assessment around AI infrastructure dependencies (e.g., "Do you use an LLM gateway or proxy layer? If so, which one?") would have given you the information you need to act quickly.

2

Collect CI/CD and supply chain controls during assessments, not just policies

A SOC 2 Type II from LiteLLM would have confirmed a policy exists around version control and release management. It would not have told you they were pulling Trivy from apt without pinning it to a specific version, which is what created the opening.

During vendor assessments, controls like dependency pinning to cryptographic hashes, MFA enforcement on package registry accounts, isolated build environments, and credential scoping for CI/CD runners are more predictive of supply chain risk than a policy-level attestation. These are questions your infosec team can define once and collect across the vendor portfolio without reinventing them every cycle.

3

Have a vendor incident response playbook that maps to your integration layer

For most affected organizations, preventing this was not realistic. The priority in such a case is containment. That means maintaining a current map of which vendor integrations touch which internal systems, what credentials those integrations hold, and a runbook for rotating them quickly.

The Trivy-to-LiteLLM chain is itself a lesson in what happens when credential rotation is slow or partial: Aqua's incomplete rotation is the reason the LiteLLM compromise was possible at all. Your response plan needs to account for the same risk on your side.

The Bottom Line

The LiteLLM compromise is an example of how modern supply chain attacks cascade. A threat actor compromised a security scanner, which gave them access to an AI library's publishing credentials, which gave them a foothold in every organization that auto-updated, and every customer those organizations served.

No assessment, questionnaire, or external security score would have predicted this. That is not an argument for throwing out those tools. But it does make the case for thinking more carefully about what we monitor, how often, and whether the questions we ask vendors actually tell us anything about the risks that matter.



Could Coverbase Have Helped?

A supply chain attack that propagates through a compromised security scanner and hits a PyPI package in under three hours is at the extreme end of what any TPRM program can detect in advance. No vendor risk platform was going to prevent this. That said, there are specific places where Coverbase would have made a real difference for teams managing the fallout.



Surfacing the right controls during assessments

If your team assessed a vendor running LiteLLM through Coverbase, controls around CI/CD pipeline security, dependency pinning practices, and publishing credential management (MFA on PyPI accounts, scoped tokens, isolated runners) could have been collected and tracked as part of the infosec review without your team manually chasing those answers each cycle.

That does not stop the attack, but it gives your program documented awareness of the risk category before the incident occurs, and gives you something concrete to point to when scoping your response.



Instant alert

The first public signal was a GitHub issue opened by researcher Callum McMahon at FutureSearch at 11:48 UTC on March 24, which spread to Hacker News and Reddit within minutes. Snyk published a formal advisory within hours and was among the first security vendors to issue detailed guidance. Sonatype's automated scanning also detected the malicious packages within seconds of publication (tracked as sonatype-2026-001357). Coverbase's monitoring layer ingests from these sources.

If "LiteLLM" or "BerriAI" appeared in your vendor inventory, or as a known dependency of a vendor in your inventory, an alert was issued tying the public disclosure to your specific exposure. Beyond news sources, a connected third-party tool like Wiz, Snyk, or an EDR platform that flagged outbound connections to the known C2 domains could have fed into Coverbase as a corroborating signal, compressing your time-to-awareness from days to hours.



Coordinating outreach

Once an incident is confirmed against a vendor in your inventory, Coverbase supports structured response workflows. Internally, that means generating a notification to your risk, security, and procurement stakeholders with the affected vendor name, the nature of the incident, which business units or integrations are potentially in scope, and recommended next steps.

Externally, that means drafting and tracking outreach to the affected vendors requesting their incident response status, remediation timeline, and any indicators of compromise relevant to your environment. Both of these can be initiated from within Coverbase without your team context-switching through email and spreadsheets to figure out who to notify and what to ask.