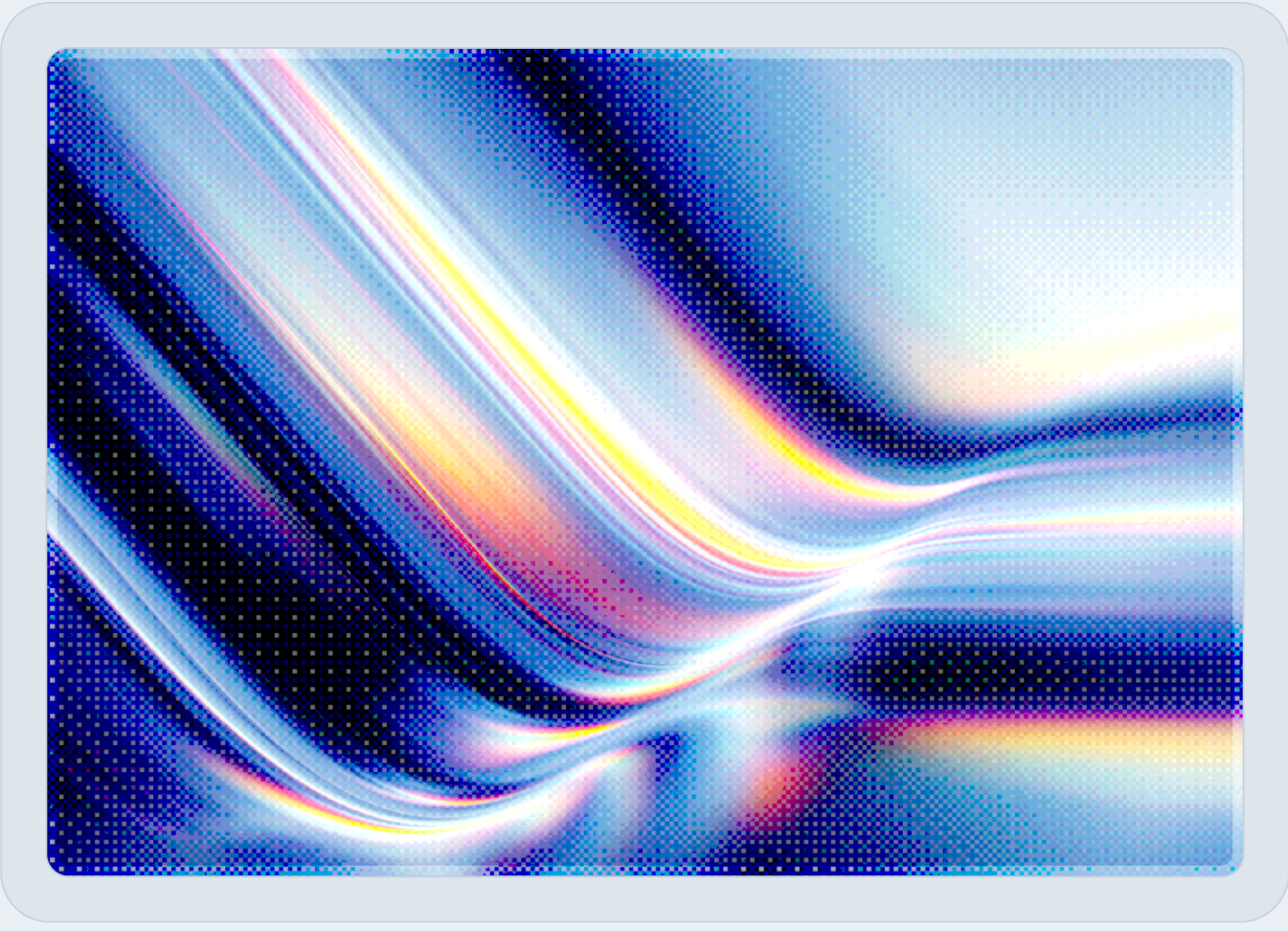
CSAI Foundation | Cloud Security Alliance

Credential Weaponization in the AI/ML Supply Chain

Quasar Linux RAT and ZiChatBot PyPI Campaigns

2026-05-08

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Two distinct threat campaigns, Quasar Linux (QLNX) and ZiChatBot, demonstrate that attackers are increasingly targeting developer workstations and open-source package registries as a gateway into AI and machine learning infrastructure.
- Quasar Linux (QLNX) is a newly identified Linux remote access trojan with rootkit and PAM backdoor capabilities, designed specifically to harvest developer credential stores including PyPI API tokens, npm tokens, AWS credentials, Kubernetes configuration files, and GitHub CLI tokens [1][2].
- The ZiChatBot campaign, attributed with moderate confidence to the OceanLotus APT group (also known as APT32), used three malicious PyPI packages — `uuid32-utils`, `colorinal`, and `termncolor` — as dropper mechanisms, employing AI chatbot names as social engineering lures and Zulip's legitimate team chat APIs as covert command-and-control infrastructure [3][4].
- Credential compromise at the developer level does not merely expose individual systems; it provides attackers with the signing authority and publishing access needed to inject malicious code into downstream packages, creating a secondary wave of supply chain compromise across any organization that depends on the affected packages.
- These campaigns arrive in the context of a broader escalation: the LiteLLM PyPI breach in March 2026 [5], the PyTorch Lightning compromise in April 2026 [6], and the dYdX npm/PyPI compromise in February 2026 [7] collectively suggest that AI and ML infrastructure has become a priority target for both nation-state and financially motivated actors.

Background

The software supply chain for AI and machine learning systems depends on a complex web of open-source packages hosted on public registries, primarily PyPI for Python-based ML frameworks and tooling. Developer workstations serve as the root of trust for this ecosystem: when a maintainer publishes a package, they do so using credentials stored in local configuration files such as `.pypirc` for PyPI tokens, `.npmrc` for npm tokens, and the AWS credential store for cloud infrastructure access.

If an attacker compromises a developer's machine and extracts these credentials, they inherit that developer's ability to publish new package versions — versions that may be automatically consumed by millions of downstream users before any detection mechanism fires.

This attack surface has long been understood in the abstract, but the concentration of three high-profile AI/ML package compromises in the first four months of 2026 suggests that exploitation may be intensifying. AI and ML packages in particular represent a high-value target: packages like LiteLLM, which recorded approximately 3.4 million downloads per day prior to its March 2026 compromise [5], carry implicit trust from organizations that have integrated them into production inference pipelines. A single compromised package version delivered through a trusted registry can simultaneously exfiltrate cloud provider credentials, API keys, and model serving secrets across thousands of organizations.

The two campaigns examined in this research note represent distinct threat postures. Quasar Linux (QLNX) is a sophisticated implant that targets developer endpoints directly, building persistent access before harvesting every credential store it can locate. ZiChatBot represents the complementary approach: rather than compromising developer machines to gain registry access, it uses the registry itself as the delivery mechanism, relying on developers to install malicious packages as part of their normal workflow. Together, they illustrate that the path from initial access to supply chain compromise now runs in both directions.

Security Analysis

Quasar Linux (QLNX): Endpoint Compromise as Supply Chain Access

Quasar Linux, designated QLNX by initial researchers, was identified in early May 2026 as a Linux-targeted remote access trojan with capabilities that go beyond typical RAT functionality to include a kernel-level rootkit and a PAM backdoor, both of which complicate detection and remediation [1][2]. Its architecture combines remote access, keylogging, clipboard monitoring, file manipulation, and network tunneling with a rootkit module and a PAM (Pluggable Authentication Modules) backdoor that can survive standard detection and remediation efforts. The PAM backdoor is particularly significant for developer environments: it allows attackers to authenticate as any local user through a hardcoded credential, persisting even if the malware's primary process is discovered and terminated.

What makes QLNX specifically relevant to AI and ML supply chains is its credential harvesting module, which appears purpose-built for developer workstations. The module systematically enumerates a defined list of credential store locations that are characteristic of software developers rather than

general enterprise users: `.pypirc` for PyPI API tokens, `.npmrc` for npm registry credentials, `.git-credentials` and the Git credential manager store, `.aws/credentials` and `~/.aws/config` for AWS access, `.kube/config` for Kubernetes cluster authentication, `.docker/config.json` for Docker registry credentials, `.vault-token` for HashiCorp Vault, Terraform credential stores, and GitHub CLI authentication tokens [1]. Environment files, which are frequently used in AI development pipelines to store API keys for services like Hugging Face, OpenAI, Anthropic, and cloud AI platforms, are also targeted.

At the time of initial discovery, QLNX was flagged by only four security solutions in common detection databases [2][9]. This low detection rate, combined with its rootkit capability, means that compromised developer systems may not generate alerts through conventional endpoint security tooling. The implication for organizations is that QLNX may be present on developer machines for extended periods before detection — long enough to harvest credentials, observe development workflows, and potentially identify which packages a maintainer has publishing access to. Once PyPI API tokens are in the attacker's possession, they can publish malicious versions of any package associated with those tokens without any further access to the developer's machine.

ZiChatBot: Package Registry as Delivery Mechanism

The ZiChatBot campaign takes a structurally different approach, using PyPI itself as the initial compromise vector rather than a downstream consequence of endpoint compromise [3][4]. Three malicious packages — `uuid32-utils`, `colorinal`, and `termncolor` — were uploaded to PyPI between July 16 and July 22, 2025. The packages remained on the registry for approximately ten months before discovery in May 2026, during which time they were available for installation by any developer searching for utility or terminal color libraries.

Attribution analysis by Kaspersky researchers identified a 64% code similarity to dropper components previously associated with OceanLotus (APT32), a Vietnamese-aligned threat actor with a documented history of targeting software developers, technology companies, and government entities [3]. Attribution at this confidence level should be treated as a strong indicator rather than a definitive conclusion; the overlap may reflect shared tooling, deliberate mimicry, or common ancestry in a shared exploit kit.

The packages' behavior differs between Windows and Linux targets. On Windows, installation extracts a file named `terminate.dll`, which loads automatically when the library is imported into a project and creates an auto-run registry entry for persistence. On Linux, the corresponding payload is `terminate.so`, a shared object deployed to `/tmp/obsHub/obs-check-update` with a crontab entry configured for persistent execution [3]. Both paths ultimately deliver the ZiChatBot

implant, which communicates with its command-and-control infrastructure not through attacker-controlled servers — a pattern that network security tools are designed to detect — but through Zulip's legitimate REST API endpoints. Commands arrive as messages in a Zulip organization controlled by the attackers; the implant confirms successful command execution by responding with a heart emoji [3][4][8].

The use of a legitimate, well-known collaboration platform as a C2 channel is a technique with growing precedent. It allows malicious traffic to blend into the outbound communication patterns typical of developer environments, where Slack, Discord, Microsoft Teams, and similar services generate constant traffic that is rarely inspected for content in most organizational environments, though CASB and DLP solutions may provide partial visibility. Network-level blocking of Zulip would be unusable as a detection mechanism for most organizations. The Zulip organization associated with this campaign was deactivated and the packages removed from PyPI following discovery, but the architectural template — delivery through a plausible-looking utility package, persistence through platform-native mechanisms, C2 through legitimate SaaS APIs — remains available for reuse.

The Compound Threat: Credential Theft Enabling Package Compromise

Both campaigns become significantly more dangerous when considered in combination. QLNX harvests the credentials that give developers publishing authority over PyPI packages. ZiChatBot uses PyPI packages to establish persistent access and remote code execution capability on developer machines. The following describes a plausible multi-stage threat scenario based on the tools and techniques observed in these campaigns: an attacker could use a ZiChatBot-style package to gain initial footholds on developer workstations, deploy QLNX to establish durable persistent access, harvest PyPI credentials from those machines, and then publish QLNX-infused or credential-harvesting versions of the same legitimate packages the developers maintain.

This recursive dynamic — using supply chain access to deepen supply chain compromise — has been observed in fragments across recent campaigns. The PyTorch Lightning compromise in April 2026 used stolen GitHub tokens to propagate malicious code into repository branches, demonstrating how compromised publishing credentials can enable lateral spread across a package's version history [6]. The dYdX npm and PyPI compromise in February 2026 was attributed to a developer account compromise that enabled the publishing of wallet-stealing and RAT-bearing package versions [7]. While this compound attack has not been directly attributed to a single end-to-end operation in the public record, the components needed to execute it are independently established: two campaigns in May 2026 have demonstrated each stage in isolation.

Recommendations

Immediate Actions

Organizations should audit credential stores on developer workstations for signs of unauthorized access or exfiltration. PyPI API tokens should be rotated immediately if there is any uncertainty about the security of the workstations used to publish packages. Organizations that publish packages to PyPI, npm, or other registries should review their publishing history for any releases not explicitly authorized by the responsible maintainer.

Teams that have installed terminal utility or color library packages from PyPI in the 2025–2026 timeframe should audit their dependency trees for the presence of `uuid32-utils`, `colorinal`, or `termncolor`. If any of these packages are present in project dependencies, they should be removed immediately, all credentials accessible from the affected environment should be rotated, and the system should be examined for the Linux payload at `/tmp/obsHub/obs-check-update` and for the Windows registry auto-run entries associated with `terminate.dll`.

Short-Term Mitigations

Developer workstation security deserves dedicated attention as a supply chain risk control point, not merely an endpoint security problem. Organizations should implement scoped, single-purpose PyPI API tokens rather than tokens with broad publishing scope across multiple packages. PyPI supports project-scoped tokens that restrict publishing authority to a specific package; using them limits the blast radius of any credential compromise to a single package rather than an entire maintainer account. Two-factor authentication should be enforced on all package registry accounts; PyPI has made 2FA mandatory for critical package maintainers [12], but organizations should verify that their developers have enrolled.

Monitoring for unusual publishing activity — version releases outside normal working hours for single-maintainer packages or teams operating in a defined time zone, releases from unfamiliar IP addresses, or releases that substantially increase package size or introduce binary components — should be implemented through CI/CD pipeline controls or registry monitoring tooling. Anomaly detection thresholds should be calibrated to each package's historical publishing patterns to reduce false positives. Dependency pinning and the use of lock files, combined with automated scanning of transitive dependencies against known-malicious package databases, reduce exposure to packages that have been compromised after initial installation.

On the network side, outbound communication from developer workstations to collaboration APIs such as Zulip, Discord, or Slack should be subject to inspection where technically feasible. While blocking legitimate platforms is impractical, anomalous patterns — communication to unfamiliar Zulip organizations, high-frequency API calls to collaboration endpoints during non-working hours, or outbound calls from background processes rather than user-initiated sessions — may indicate C2 activity.

Strategic Considerations

The credential ecosystem that AI and ML developers carry on their workstations has expanded substantially alongside the tooling landscape — in ways that have outpaced most organizations' secrets management practices. An ML researcher working with production AI systems may hold credentials for cloud GPU providers, model hosting platforms, experiment tracking services, vector databases, private model repositories, and multiple inference API services, in addition to the traditional developer credential stores targeted by QLNX. Organizations building AI systems should treat this credential surface as a first-class risk and apply secrets management practices — vault-based secret injection, workload identity rather than static credentials, and short-lived tokens where the service supports them — rather than relying on local credential store files.

The pattern of APT-attributed campaigns targeting the Python ecosystem through plausible utility packages represents a continued escalation of nation-state and advanced persistent threat tradecraft that has been active since at least the early 2020s. The ZiChatBot campaign's ten-month dwell time in the PyPI registry before discovery illustrates that detection coverage for malicious packages can be insufficient to catch patient, low-noise campaigns — even with improved automated scanning. Organizations should not assume that packages removed from PyPI were caught promptly. Organizations that publish or consume Python packages should engage with the broader ecosystem's supply chain security initiatives: participating in package signing verification through Sigstore [13], integrating SBOM generation into release pipelines, and monitoring dependency advisories through both the GitHub Advisory Database and dedicated package security feeds.

CSA Resource Alignment

The threat pattern described in this research note maps directly to several domains within the CSA AI Controls Matrix (AICM) v1.0.3 [10]. The Supply Chain Management, Transparency and Accountability (STA) domain addresses third-party model and package risk, vendor assessment requirements, and AI bill of materials obligations — all directly applicable to the open-source package dependency risk

surfaced by ZiChatBot and QLNX. The Identity and Access Management (IAM) domain's controls govern access controls for training pipelines, model serving infrastructure, and registry publishing credentials, with particular relevance to the scoped token and 2FA recommendations outlined above.

The Threat and Vulnerability Management (TVM) domain covers adversarial testing and model vulnerability management practices that extend to the pipeline and tooling layer, including dependency scanning and supply chain integrity verification. The Logging and Monitoring (LOG) domain's controls for AI behavior telemetry and anomaly detection apply to the monitoring for anomalous registry publishing activity and unusual outbound API communication described in the mitigation guidance.

CSA's MAESTRO framework for agentic AI threat modeling [11] provides threat categorization relevant to these campaigns under the "Insecure Supply Chain" threat category, which encompasses compromised package dependencies, malicious model artifacts, and credential theft enabling downstream artifact manipulation. Organizations using MAESTRO for threat modeling of AI development pipelines should explicitly include developer workstations and package registry accounts within the trust boundary definition, as these campaigns demonstrate that the path from workstation compromise to production AI system compromise is direct and well-understood by attackers.

More broadly, these campaigns reinforce the case for treating AI/ML development pipelines under the same Zero Trust principles that apply to production infrastructure. The assumption that developer workstations operating within a corporate network are inherently trusted is inconsistent with the threat model illustrated here. CSA's Zero Trust guidance recommends continuous verification of device posture, workload identity for service-to-service communication, and least-privilege access patterns — all of which apply directly to the credential stores and publishing authorities that QLNX and ZiChatBot are designed to exploit.

References

[1] Trend Micro Research. "Quasar Linux (QLNX): A Silent Foothold in the Software Supply Chain." Trend Micro, May 2026.

[2] BleepingComputer. "New Stealthy Quasar Linux Malware Targets Software Developers." BleepingComputer, May 2026.

[3] Kaspersky Securelist. "OceanLotus Suspected PyPI ZiChatBot Campaign." Kaspersky, May 2026.

[4] The Hacker News. "PyPI Packages Deliver ZiChatBot Malware via Zulip C2." The Hacker News, May 2026.

[5] Trend Micro Research. "Your AI Stack Just Handed Over Your Root Keys: Inside the LiteLLM PyPI Breach." Trend Micro, March 2026.

[6] The Hacker News. "PyTorch Lightning Compromised in PyPI Supply Chain Attack." The Hacker News, April 2026.

[7] The Hacker News. "Compromised dYdX npm and PyPI Packages Deliver RAT Malware." The Hacker News, February 2026.

[8] GB Hackers. "ZiChatBot Malware Abuses Zulip APIs for Covert Command and Control." GB Hackers, May 2026.

[9] SecurityWeek. "Sophisticated Quasar Linux RAT Targets Software Developers." SecurityWeek, May 2026.

[10] CSA. "AI Controls Matrix (AICM) v1.0.3." Cloud Security Alliance, 2025.

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

[12] PyPI. "PyPI Two-Factor Authentication Enforcement." Python Software Foundation.

[13] Sigstore. "Sigstore: A New Era of Software Signing." Sigstore Project.

