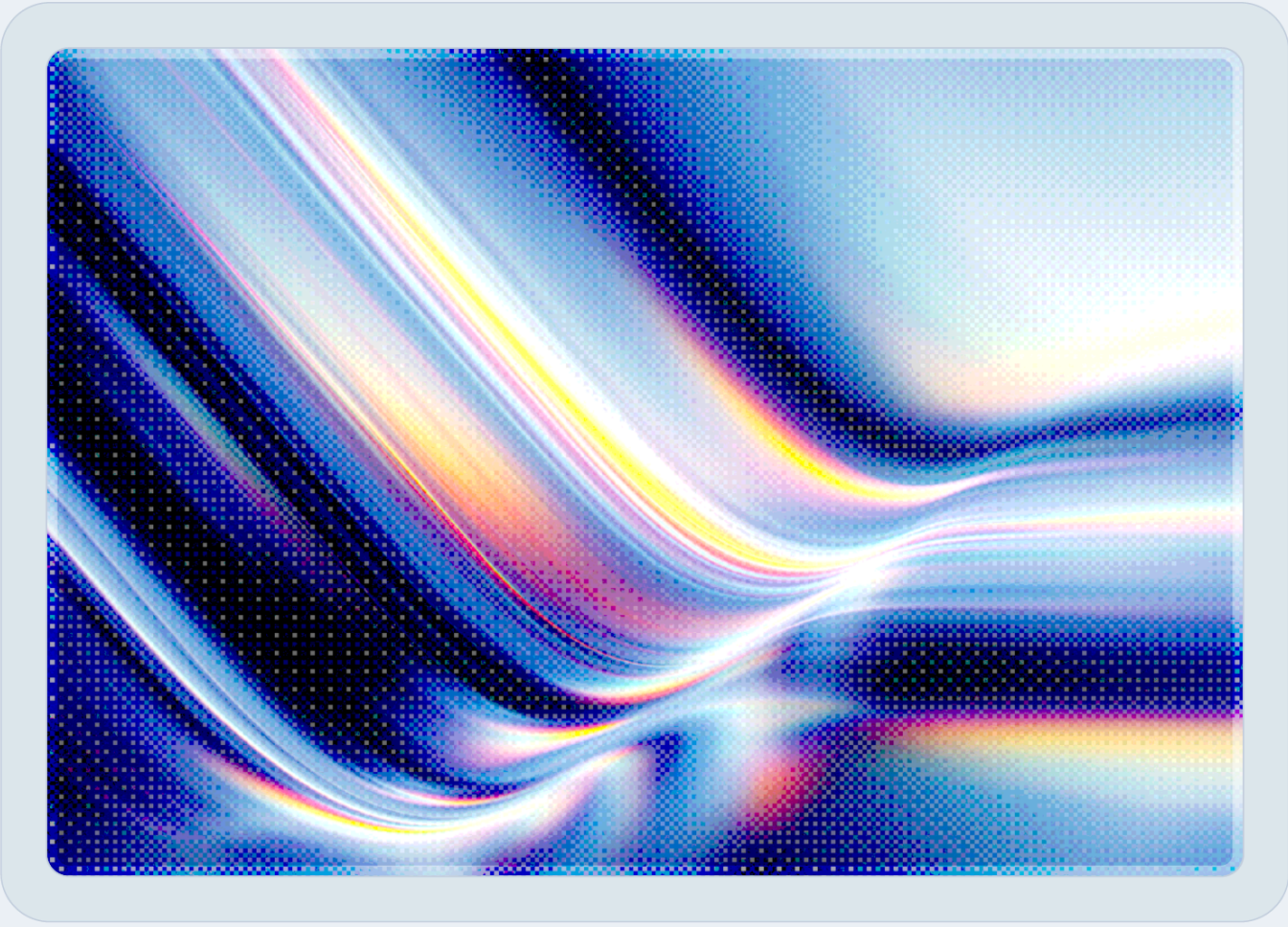
CSAI Foundation | Cloud Security Alliance

QLNX: Linux RAT Targeting AI/ML Developer Supply Chains

Credential Harvesting via Rootkit and PAM Backdoor Threatening AI Package Registries

2026-05-09

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- QLNX (Quasar Linux) is a newly disclosed, sophisticated Linux remote access trojan that targets developers and DevOps teams by harvesting credentials for package registries, cloud platforms, and CI/CD pipelines, with particular relevance to AI/ML supply chains.
- The malware deploys a two-tier rootkit — combining an LD_PRELOAD userland component with an eBPF kernel-level controller — alongside a PAM backdoor that captures plaintext credentials from SSH and system authentication events.
- QLNX's credential harvester explicitly targets assets central to AI/ML supply chains: PyPI API keys (.pypirc), npm tokens (.npmrc), GitHub CLI tokens, AWS credentials, Kubernetes configuration files, and application environment files that frequently hold API keys for commercial AI services.
- At the time of initial disclosure, only four security products detected QLNX, reflecting high risk of undetected persistence on developer workstations [4].
- Organizations that develop or consume AI frameworks distributed through PyPI or npm should treat Linux developer workstations as high-priority threat surfaces requiring immediate credential rotation, package hash pinning, and enhanced endpoint behavioral monitoring.

Background

The software supply chain has become a primary target for sophisticated threat actors, and developer workstations — historically treated as secondary targets — are now recognized as high-value footholds. The reason is straightforward: these machines hold the authentication material that governs who can publish code to package registries, who can access cloud infrastructure, and who can approve changes to critical delivery pipelines. The compromise of a single developer credential can propagate malicious code to millions of downstream users within hours.

The AI and machine learning ecosystem has intensified this risk profile significantly. Python is the dominant language for AI/ML development, and PyPI is the primary public registry through which most Python-based AI frameworks are distributed. PyPI package publication depends on maintainer-controlled API keys, and a stolen key is sufficient for an attacker to publish a poisoned package under a trusted maintainer's identity. The LiteLLM incident of March 2026 illustrated the stakes in concrete terms:

two malicious releases (versions 1.82.7 and 1.82.8) reached PyPI for approximately 40 minutes before quarantine, yet more than 40,000 downloads of the compromised package occurred during that window [1][2][11]. The malicious payload, once installed, harvested environment variables, SSH keys, cloud credentials, Kubernetes configurations, Docker settings, and CI/CD secrets from victim environments [2]. Investigators linked the initial access to credential theft targeting the LiteLLM build pipeline and attributed the broader campaign to the threat actor group TeamPCP [3].

QLNX — disclosed in May 2026 by Trend Micro Research — represents the class of pre-intrusion tooling that enables attacks of this kind at scale [4]. Named "Quasar Linux" and tracked under the identifier QLNX, this previously undocumented Linux remote access trojan is engineered to silently compromise developer and DevOps environments, harvest the credentials that matter most to supply chain attackers, and maintain undetected persistence long enough to enable downstream abuse. The malware's targeting logic — selecting specific credential files that govern access to AI/ML registries and infrastructure — suggests a threat actor with detailed operational knowledge of how modern AI development environments are structured.

Security Analysis

Architecture and Fileless Deployment

QLNX is distributed as a compiled binary that carries all of its secondary components — rootkit source code, PAM backdoor code, and credential harvesting logic — embedded as string literals within the binary itself [4]. On deployment, the malware dynamically compiles its rootkit shared objects and PAM backdoor modules on the target host using the system's locally installed gcc toolchain, then deploys them without leaving conventional artifact files that static scanners would identify as malicious [4][5]. The malware subsequently executes filelessly from memory and spoofs its process name to mimic legitimate Linux kernel threads — for example, kworker or ksoftirqd — making it difficult to isolate in process listings alongside benign kernel activity [4].

Before establishing persistence, QLNX profiles the host environment to detect containerized deployments and adjusts its behavior accordingly [5][6]. This profiling capability indicates that the malware authors understand modern DevOps infrastructure, where developers frequently work inside Docker containers or within CI/CD-proxied environments, and have designed QLNX to behave differently in monitored versus workstation-native contexts.

Two-Tier Rootkit

QLNX deploys two complementary rootkit mechanisms designed to conceal its presence at different levels of the operating system. The first is a userland rootkit implemented via LD_PRELOAD injection: by inserting a shared library into the dynamic linker path, QLNX intercepts standard library calls made by system utilities such as ls, ps, and netstat, filtering their output to hide the malware's processes, files, and listening network ports from administrators investigating the host [4][5]. The second component is an eBPF-based kernel rootkit controller that manages BPF maps — kernel data structures — to extend hiding capability directly into kernel space, beyond the reach of user-level investigation tools [4].

The combination of these two layers is meaningful from a detection standpoint. User-level host-based detection tools that operate within userland are subject to the same library interception that hides QLNX from administrators. Kernel-level detection approaches face the eBPF controller. Security teams investigating a potentially compromised host should assume that standard diagnostic tools may return incomplete results and should employ memory forensics techniques capable of operating outside the compromised process space.

PAM Backdoor and Credential Interception

Pluggable Authentication Modules (PAM) govern authentication across Linux systems, including SSH login, sudo privilege escalation, and graphical session login. QLNX installs two distinct PAM backdoor implementations that exploit this centrality to the authentication stack [4][5]. The first implementation harvests plaintext credentials from authentication events — capturing passwords at the moment of entry — and includes a hardcoded master password that permits the threat actor to authenticate to any compromised host regardless of the legitimate user's credentials. It also logs outbound SSH session data, capturing credentials used when the compromised developer connects to remote systems such as CI/CD servers, cloud instances, or shared development infrastructure [4]. The second PAM module loads into dynamically linked processes to extract the service name, username, and authentication token at each authentication invocation [4].

Together, these components give the attacker a persistent, low-visibility mechanism for credential capture that operates continuously across the malware's lifetime on the host. PAM-level interception is particularly damaging because it captures credentials at the point of use — meaning even credentials rotated after the initial compromise may be harvested again upon next use.

Credential Harvesting and the AI/ML Supply Chain

The most directly consequential capability for AI/ML supply chain security is QLNX's targeted credential harvesting routine. The malware conducts a systematic sweep of well-known credential file locations, extracting authentication material across a comprehensive set of targets: npm configuration files (.npmrc) containing npm registry tokens; Python package index credentials (.pypirc) containing PyPI API keys used to publish packages; git credential stores (.git-credentials) and GitHub CLI token files; AWS credential files (.aws/credentials); Kubernetes configuration files (.kube/config); Docker registry credentials (.docker/config.json); HashiCorp Vault tokens (.vault-token); Terraform credentials; and application environment files (.env) that commonly contain API keys for commercial AI services, database connection strings, and infrastructure integration secrets [4][5][6].

For organizations developing or deploying AI and machine learning systems, this credential set covers substantially all of the authentication material required for supply chain compromise. PyPI API keys allow the holder to publish packages under a trusted maintainer's identity, enabling trojanization of AI frameworks with large install bases. AWS credentials provide access to S3 buckets that may contain proprietary model weights, training datasets, and AI artifacts. Kubernetes configuration files may govern GPU cluster access for model training workloads. Environment files in AI development projects commonly contain API keys for commercial LLM services and model serving infrastructure. The breadth and specificity of QLNX's targeting strongly suggests deliberate focus on individuals with package publishing permissions and AI infrastructure access — not opportunistic credential sweeping.

Rankiteo researchers monitoring QLNX infrastructure have reported active credential-driven supply chain attacks against PyPI and npm [6], though this observation has not yet been independently corroborated by major threat intelligence vendors.

Persistence Mechanisms

QLNX establishes persistence through at least seven distinct mechanisms, ensuring that the removal of any single foothold does not eliminate the threat [4][5]. Documented persistence methods include systemd service installation, crontab entries, and .bashrc shell injection, alongside additional redundant techniques. This multi-layered approach is operationally significant: standard incident response steps such as disabling a suspicious service may leave other persistence channels intact. The eBPF rootkit's ability to hide specific processes, files, and network connections further complicates forensic investigation, since the investigators may not observe the full scope of QLNX's installed components even during an active response.

Relationship to the Broader AI/ML Threat Ecosystem

QLNX does not represent an isolated tool. It fits within a broader ecosystem of threats targeting AI/ML infrastructure from multiple angles. Malicious AI models distributed through repositories such as Hugging Face have been documented carrying payloads that execute arbitrary code when a user loads a PyTorch model file via Python's pickle deserializer [7][8]. Separately, Jinja2 metadata templates embedded in GGUF model files can carry instructions that execute at inference time, bypassing static file scanners [12]. These vectors target different points in the AI/ML workflow: QLNX compromises the developer who builds and publishes AI packages; malicious model files target the data scientist or engineer who downloads and loads those packages; and compromised packages reach both categories of victim at scale through trusted registries. The convergence of these techniques across the stack suggests that threat actors are building layered approaches to AI supply chain compromise rather than relying on any single method.

Recommendations

Immediate Actions

Security and engineering teams should treat the following as urgent priorities in the 72 hours following this advisory. All package registry credentials — PyPI API keys, npm tokens, and GitHub personal access tokens — should be rotated immediately, with priority given to any maintainer of packages with significant install counts or with access to AI/ML framework repositories. Cloud credentials — AWS IAM access keys, GCP service account keys, and Azure service principal secrets — that are stored on developer workstations or in local environment files should be rotated and reviewed in cloud access logs for signs of unauthorized use.

Behavioral endpoint detection should be deployed or tuned to alert on anomalous gcc invocations, unexpected modifications to /etc/ld.so.preload, BPF program loading events by non-root processes, and process name patterns inconsistent with the host's kernel version — all indicators associated with QLNX deployment [4][5]. Security teams should assume that standard process-listing and file-listing tools may return manipulated output on any host suspected of compromise, and should conduct forensic analysis using memory forensics tools capable of operating outside the compromised userland context.

CI/CD pipeline configurations should be audited to identify any secrets stored in developer workstation environment variables rather than dedicated secrets management infrastructure. Any such secrets should be migrated to a managed secrets vault and removed from workstation-local storage.

Short-Term Mitigations

Over a 30-day horizon, organizations should enforce hardware security key (FIDO2) authentication for all package registry publishing operations. PyPI, npm, and GitHub all support mandatory hardware key requirements for publishing actions; enforcing this control removes stolen API key credentials as a sufficient condition for supply chain compromise, requiring physical possession of the hardware token in addition to the credential.

Kernel integrity controls on developer Linux workstations should be reviewed and strengthened. Secure Boot enforcement, kernel module signing requirements, and restrictions on unprivileged eBPF program loading via Linux Security Module policies (AppArmor or SELinux) limit QLNX's ability to deploy its kernel-level rootkit component. PAM configuration directories (/etc/pam.d/ and the PAM module path) should be placed under file integrity monitoring with alerts generated on any modification.

AI/ML package consumers should implement dependency hash pinning across all requirements files and CI/CD pipeline dependency specifications, ensuring that package updates require explicit maintainer action rather than automatic resolution to the latest version. The LiteLLM incident demonstrated that even a 40-minute exposure window is sufficient to cause significant downstream impact when hash verification is absent [1][2].

Strategic Considerations

The QLNX threat model exposes a structural assumption that warrants reconsideration across AI/ML engineering organizations: developer workstations are often treated as trusted endpoints, while package registries and cloud infrastructure are treated as the primary control surfaces. QLNX inverts this architecture. The workstation is the initial access point, and the credentials it holds are the mechanism for registry and infrastructure compromise. Organizations that have invested significantly in registry-side security controls — such as PyPI two-factor requirements and npm publish controls — may have a false sense of security if the developer workstations that hold publishing credentials are not receiving commensurate protection.

Adopting Zero Trust principles for developer workstations is the appropriate strategic response. Under Zero Trust, workstations are treated as potentially compromised endpoints rather than implicitly trusted nodes. This means that registry publishing operations should require multi-party approval and hardware authentication; cloud access from developer machines should use short-lived credentials with minimum necessary scope rather than persistent access keys; and CI/CD pipelines that build and publish AI packages should operate in isolated, ephemeral environments that do not inherit credential material from developer workstation sessions.

The converging threat landscape — Linux RAT tooling targeting developer workstations, malicious model files targeting data scientists, and trojanized packages reaching end consumers — indicates that AI/ML supply chain security requires defense-in-depth spanning the developer environment, the package registry, the model repository, and the inference infrastructure simultaneously. Organizations that approach any one of these layers in isolation should expect adversaries to route around that single control.

CSA Resource Alignment

CSA's MAESTRO framework (Multi-Agent Environment, Security, Threat, Risk, and Outcome) provides threat modeling guidance designed for agentic AI systems and their dependency chains [9][10]. QLNX attack scenarios map directly to MAESTRO's treatment of compromised supply chain artifacts reaching AI agent execution environments: a developer whose credentials are stolen may inadvertently publish a trojanized version of a model library that an agentic system loads and executes with elevated trust. MAESTRO's emphasis on validating tool and dependency provenance at each layer of an AI system architecture is directly applicable to the QLNX threat model, particularly for organizations operating agentic pipelines that depend on PyPI or npm packages for tool execution [10].

CSA's AI Controls Matrix (AICM) v1.0.3 addresses AI supply chain security across provider roles — Application Providers, Orchestrated Service Providers, and Cloud Service Providers — with control domains covering AI model security, supply chain governance, and shared security responsibility [13]. The AICM's supply chain controls are relevant here, though the developer workstation credential exposure that QLNX exploits may sit outside the scope of controls oriented toward registry and infrastructure-level governance. Organizations implementing AICM should explicitly extend supply chain control requirements to cover developer endpoint security as a pre-condition for registry access.

CSA's Cloud Controls Matrix (CCM) Supply Chain Management domain addresses third-party risk management practices applicable to AI package dependencies [14]. The STA control family, particularly STA-09 (Supply Chain Inventory), provides a framework for tracking the provenance of AI package dependencies and establishing requirements for dependency verification that QLNX-class threats now make urgently applicable.

CSA's Zero Trust guidance is directly applicable to the workstation-as-initial-access-vector pattern QLNX represents. Developer workstations should be treated as untrusted endpoints under Zero Trust architecture, with continuous verification required for registry publishing, cloud access, and CI/CD pipeline participation — rather than persistent session credentials that can be silently exfiltrated and reused.

References

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

[2] Datadog Security Labs. "LiteLLM and Telnyx compromised on PyPI: Tracing the TeamPCP supply chain campaign." Datadog Security Labs, March 2026.

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

[4] Trend Micro. "Quasar Linux (QLNX) – A Silent Foothold in the Supply Chain: Inside a Full-Featured Linux RAT With Rootkit, PAM Backdoor, Credential Harvesting Capabilities." Trend Micro Research, May 2026.

[5] BleepingComputer. "New stealthy Quasar Linux malware targets software developers." BleepingComputer, May 2026.

[6] Rankiteo. "PyPI and npm: QLNX Threat Actors Steal Developer Credentials For Supply Chain Attacks." Rankiteo Blog, May 2026.

[7] BleepingComputer. "Malicious AI models on Hugging Face backdoor users' machines." BleepingComputer, February 2024.

[8] JFrog. "Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor." JFrog Blog, February 2024.

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

[10] CSA. "Applying MAESTRO to Real-World Agentic AI Threat Models: From Framework to CI/CD Pipeline." Cloud Security Alliance, February 2026.

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

[12] JFrog Security Research. "GGUF-SSTI: Jinja2 Template Injection in GGUF Model Metadata." JFrog Security Research, May 2024.

[13] CSA. "AI Controls Matrix." Cloud Security Alliance, 2025.

[14] CSA. "Cloud Controls Matrix and CAIQ v4.1." Cloud Security Alliance, 2026.