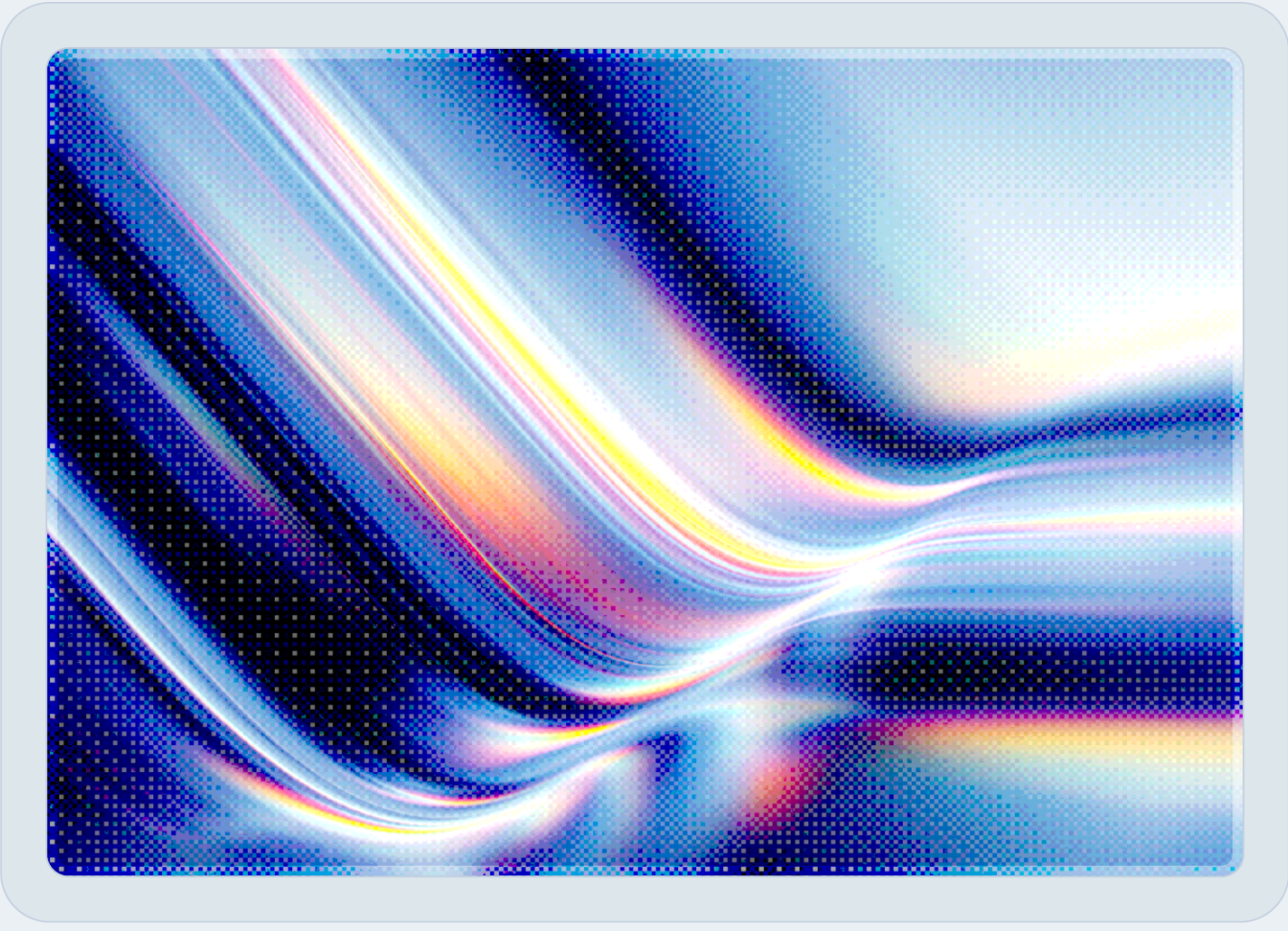
CSAI Foundation | Cloud Security Alliance

Quasar Linux RAT: Developer Credential Theft for AI Supply Chain

How the QLNX implant exploits ML ecosystem credentials to enable software supply chain compromise

2026-05-10

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- A previously undocumented Linux implant designated QLNX (Quasar Linux RAT) was disclosed by Trend Micro in early May 2026. It executes entirely from memory, deploys dual rootkits and a PAM authentication backdoor, and registers 58 operator commands — making it one of the most technically sophisticated developer-targeted Linux implants publicly analyzed to date, notable for its combination of fileless execution, dual-tier rootkit architecture, and breadth of C2 capability. [1]

- QLNX's primary objective is credential exfiltration from developer workstations. Its harvester explicitly targets PyPI tokens (`.pypirc`), npm credentials (`.npmrc`), Git tokens, AWS credentials, Kubernetes configs, Docker Hub credentials, Vault tokens, Terraform credentials, GitHub CLI tokens, and `.env` files — the exact toolkit used by machine learning engineers to publish packages, access training infrastructure, and manage model registries. [1][2]

- ML developers are a particularly high-value subset of QLNX's target population, because their credentials combine registry publishing rights, cloud compute access, and model artifact storage access into a single workstation credential profile. Python package ecosystem credentials give an attacker the ability to trojanize widely-used ML libraries. Cloud credentials provide direct access to GPU compute, model weights, and training datasets. Environment variable files commonly store API tokens for Hugging Face, OpenAI, and other AI service providers. [2][3]

- The LiteLLM supply chain compromise of March 2026 provides a concrete prior-incident model for the damage QLNX-acquired credentials could enable: stolen CI/CD token access allowed a threat actor to publish backdoored versions of an AI gateway library with roughly 3.4 million daily downloads, exposing over 40,000 downstream installations to a credential-stealing payload and persistent remote access backdoor. [6][7]

- At the time of Trend Micro's disclosure, only four endpoint security solutions detected QLNX's binary as malicious. [1] Organizations relying exclusively on signature-based endpoint detection are unlikely to identify an active QLNX infection. Behavioral controls, credential rotation procedures, and CI/CD pipeline integrity monitoring represent the highest-priority near-term defenses given QLNX's signature-evasion architecture.

Background

Discovery and Context

Trend Micro disclosed QLNX in early May 2026 following what the firm describes as the discovery of a "previously undocumented" Linux implant targeting developer and DevOps environments. [1] The malware carries the internal designator Quasar Linux (QLNX), a name derived from the 4-byte magic value `0x51 4C 4E 58` — the ASCII encoding of "QLNX" — that appears at the start of every C2 session the implant initiates. [4][5] No threat actor attribution has been established. The initial infection vector has not been publicly documented as of this writing; Trend Micro's analysis focused on post-execution behavior. [2]

The timing of QLNX's disclosure matters. It arrives less than two months after the LiteLLM supply chain compromise demonstrated in operational terms exactly the kind of damage that a credential-focused Linux implant deployed on a developer workstation can cause. QLNX does not appear to have been responsible for the LiteLLM incident — that compromise involved a different attack path — but together the two cases illustrate the full kill chain: QLNX provides the credential theft capability; the LiteLLM incident documents the downstream damage that capability can enable. The LiteLLM incident demonstrates that a single stolen package-registry token can be sufficient to compromise tens of thousands of downstream installations, establishing a documented proof-of-concept that other threat actors can study and adapt. [6][7]

The Developer Workstation as a High-Value Target

Many high-profile software supply chain attacks have targeted build systems, registries, and CI/CD pipeline infrastructure directly. QLNX represents a tactical refinement of this pattern. Rather than attacking the pipeline infrastructure directly, it targets the human engineers who possess authenticated access to that infrastructure. A developer's Linux workstation is, in effect, a credential vault: it routinely holds API tokens for package registries, SSH keys for source control, cloud access keys for compute and storage, container registry credentials, and configuration files that collectively authorize every downstream system the developer interacts with professionally.

This attack surface is especially pronounced in machine learning engineering environments. ML developers on Linux typically maintain active credentials for multiple Python package registries, cloud provider consoles used for GPU-backed training jobs, container registries for model deployment, and AI service APIs used for evaluation and integration. They also frequently store sensitive values — including

Hugging Face access tokens and third-party AI API keys — in `.env` files that are treated as development configuration rather than secrets management artifacts. QLNX's harvester is designed to collect all of these in a single pass.

Security Analysis

Malware Architecture and Evasion

QLNX achieves persistence through a combination of technical mechanisms that, as reflected in its four-solution detection rate at disclosure, defeats the signature-based scanning present in most enterprise endpoint stacks. Upon initial execution, the malware copies itself into a RAM-backed anonymous file using `memfd_create`, relaunches from that in-memory copy using `execveat` or the `/proc/self/fd` fallback path, and deletes the original on-disk binary. [5] The result is a process that leaves no file-system artifact for disk-based scanners to identify.

The implant then deploys a two-tier rootkit architecture. The first tier operates in userspace via an `LD_PRELOAD` shared library injected into every dynamically linked process on the system, hooking libc functions to conceal QLNX's files, processes, and network sockets from standard administrative tools. The second tier operates at the kernel level through an eBPF controller that manages BPF maps, extending the concealment into kernel data structures and process accounting. [1][2] Both rootkit components are embedded in the QLNX binary as C source code string literals and compiled on the target host using the system's own `gcc` installation, which means no compiled rootkit artifact needs to be transferred to the victim system.

Process concealment is supplemented by name masquerading: QLNX disguises its process as a standard kernel worker thread, choosing names such as `[kworker/0:0]` or `[ksoftirqd]` that blend into `ps` and `top` output. It also rewrites its command-line metadata in `/proc`, removes forensic environment variables, and wipes logs stored in its own hidden file paths. [5]

Persistence is achieved through seven distinct mechanisms deployed at the operator's discretion, including `LD_PRELOAD` configuration, systemd service units, crontab entries, init.d scripts, XDG autostart desktop entries, and `.bashrc` injection. [1][2] Multiple methods can be deployed simultaneously on a single host. The PAM backdoor adds an authentication-layer component: QLNX carries two separate PAM module implementations, the first of which intercepts plaintext credentials during authentication events, logs outbound SSH session data for exfiltration, and contains a hardcoded

master password (`O$$f$QtYJK`) that grants access to any service using PAM authentication regardless of the legitimate user's password. [5] The second PAM module loads into every dynamically linked process to extract service names, usernames, and authentication tokens at the point of use.

C2 communication operates over three channels in priority order: raw TCP with TLS encryption, HTTPS using Base64-encoded POST requests for outbound data and five-second GET polls for command receipt, and plaintext HTTP as a fallback. [5] The implant's 58-command framework covers file system manipulation, interactive shell access, process management, keylogging, screenshot capture, clipboard monitoring, port scanning, SSH-based lateral movement, network tunneling, and rootkit deployment and management. [1][2]

Credential Harvesting: The ML Ecosystem Attack Surface

QLNX's credential harvester is explicitly structured around developer authentication artifacts rather than consumer credentials. Documented target paths include `.pypirc` for PyPI authentication tokens, `.npmrc` for npm registry tokens, `.git-credentials` and `~/.gitconfig` for Git token authentication, `.aws/credentials` for AWS access keys, `.kube/config` for Kubernetes service account tokens, `.docker/config.json` for container registry credentials, `.vault-token` for HashiCorp Vault access, Terraform credential stores, and GitHub CLI token files. [1][3] The harvester also sweeps `.env` files system-wide and collects SSH private keys and Firefox browser credential databases.

This credential profile maps directly onto the authentication artifacts present on a machine learning engineer's development workstation. PyPI credentials grant package publishing access; for ML teams that maintain open-source libraries or private model distribution packages, this is a direct path to supply chain injection. AWS credentials authorize SageMaker training job execution, S3 model weight storage access, and ECR container registry operations. Kubernetes configurations provide access to inference clusters and model serving infrastructure. The `.env` file sweep, while general-purpose, is the mechanism most likely to collect AI-service API tokens: Hugging Face tokens stored at `~/.huggingface/token` or in project-level `.env` files, along with API keys for OpenAI, Anthropic, Cohere, and other AI providers commonly present in `.env` files, would fall within the harvester's scope, even though these credentials are not explicitly named in published reporting.

The PAM backdoor creates a parallel credential collection channel that is independent of the file-based harvester. Any subsequent SSH authentication to the compromised host — including automated CI/CD pipeline connections — will have its credentials captured and exfiltrated. For ML development

environments where the developer's workstation serves as a jump host or SSH target for GPU-backed training runs, this represents an ongoing credential collection mechanism that persists as long as the PAM module remains installed.

The Supply Chain Pivot Path

The operational consequence of a successful QLNX compromise on a developer workstation is not limited to the workstation itself. Stolen PyPI tokens allow an attacker to publish new or modified versions of any package the victim maintainer has publish rights on. This is the exact mechanism that enabled the LiteLLM compromise in March 2026, where a stolen CI/CD token was used to publish backdoored versions of an AI gateway library before the compromise was detected and the affected versions quarantined. [6][7] The window between publishing and quarantine in that incident was approximately 40 minutes, during which over 40,000 installations downloaded a payload that deployed a credential harvester, a Kubernetes lateral movement toolkit, and a persistent RCE backdoor. [6][8]

A threat actor operating QLNX has a broader credential portfolio to work with than the single token stolen in the LiteLLM incident. The combination of PyPI publish rights, AWS credentials, and Kubernetes configurations provides multiple independent supply chain entry points, any of which can be exploited to reach downstream consumers of packages, container images, or deployed model artifacts. For ML-specific packages with large install bases — frameworks, adapters, evaluation libraries, dataset utilities — the downstream exposure surface is substantial. The GitHub CI/CD token captures enabled by QLNX's PAM backdoor extend this attack surface to the automated pipelines that publish packages on behalf of developers, eliminating even the brief manual authentication step that might alert a human operator.

Detection Gap

The four-solution detection rate at disclosure is consistent with QLNX's evasion architecture and likely reflects both the novelty of the sample and the effectiveness of its fileless and rootkit-based concealment. [1] The combination of fileless execution, dual rootkit concealment, process name masquerading, and on-host compilation of evasion components creates a posture that is specifically designed to defeat the scanning mechanisms present in most enterprise endpoint detection stacks. Behavioral detection — identifying the unusual pattern of a process that copies itself to an anonymous memory file and then deletes its own binary — is more likely to surface QLNX activity than signature-based scanning, but behavioral rules for this specific pattern must be explicitly implemented.

Recommendations

Immediate Actions

Organizations with Linux developer workstations should audit for QLNX indicators of compromise as defined in Trend Micro's published research. [1] Indicators include the presence of unsigned `LD_PRELOAD` entries in `/etc/ld.so.preload`, unexpected PAM modules in `/etc/pam.d` directories, anonymous memory-backed file descriptors visible in `/proc/<pid>/fd`, and network connections using the QLNX protocol magic `0x514C4E58`. The SOC Prime platform has published detection content mapped to QLNX TTPs for SIEM integration. [11]

ML engineering teams should treat all long-lived PyPI, npm, and cloud credentials currently stored on developer workstations as potentially compromised and rotate them. New credentials should be issued through secrets management infrastructure — HashiCorp Vault, AWS Secrets Manager, or equivalent — rather than stored in plaintext configuration files. This rotation is warranted even absent confirmed QLNX infection, as the `.pypirc`-and-`.env` credential storage pattern is independently high-risk.

Short-Term Mitigations

CI/CD pipelines should be audited to eliminate the presence of long-lived publish tokens in runner environments. The LiteLLM compromise specifically exploited a PyPI publish token stored as a GitHub Actions secret that was exfiltrated from the runner environment by a compromised build dependency. [9] OIDC-based short-lived credentials, scoped to individual pipeline runs, remove the persistent token that made the LiteLLM theft operationally durable — an attacker exfiltrating a run-scoped credential has a narrow window rather than indefinite access. Third-party GitHub Actions and build tools pulled from external repositories should be pinned to verified commit SHAs rather than mutable version tags or branch references.

Endpoint detection rules for Linux developer workstations should be extended to cover the behavioral patterns associated with QLNX: processes using `memfd_create` followed by `execveat` for in-memory self-relaunch, unexpected writes to `/etc/ld.so.preload`, gcc invocations producing shared libraries in non-standard paths, and PAM configuration modifications outside of authenticated change-management events. Network monitoring for egress on non-standard TLS ports from developer systems should alert on unexpected outbound C2 patterns.

Strategic Considerations

QLNX's capability profile is best understood as a symptom of a broader structural risk: developer workstations are treated as trusted endpoints in most enterprise security architectures, yet they hold credentials that authorize actions across the entire software delivery pipeline. Zero Trust principles that restrict lateral movement and apply just-in-time access to production-adjacent credentials are directly applicable here. An ML engineer whose workstation credentials can publish to PyPI, access production training infrastructure, and modify container images in a shared registry is, from an access-control standpoint, an extremely high-value target. Access scoping commensurate with that risk — short-lived credentials, hardware-bound authentication where available, mandatory MFA for registry and cloud console access, and separation between development and production credentials — meaningfully reduces the damage radius of a workstation compromise.

Organizations that distribute or consume open-source ML libraries should strengthen their supply chain integrity verification practices. Package signature verification, SBOM generation for internally consumed packages, and automated monitoring for unexpected version publishes from established maintainer accounts are controls that provide early-warning capability for supply chain injection events of the type QLNX credentials would enable.

CSA Resource Alignment

MAESTRO: Agentic AI Threat Modeling

The CSA MAESTRO framework provides a seven-layer threat model for agentic AI systems that is directly relevant to the QLNX threat vector. [10] Layer 4 (Deployment Infrastructure) addresses the security of CI/CD pipelines, container registries, and model deployment systems — the infrastructure that stolen ML developer credentials could be used to compromise. Layer 7 (Agent Ecosystem) covers external dependencies, tool integrations, and package registries, which constitute the downstream supply chain that a QLNX-enabled attacker would target. CSA's application of MAESTRO to CI/CD pipeline threat modeling [12] provides a practical framework for identifying where credential theft introduces risk into AI system deployment flows.

AI Controls Matrix (AICM)

CSA's AI Controls Matrix explicitly addresses AI supply chain security as a control domain. [13] The AICM's shared security responsibility model defines the obligations of model providers, application providers, orchestrated service providers, and cloud service providers in securing the provenance, integrity, and delivery of AI system components. QLNX-facilitated supply chain compromise touches multiple AICM domains: AI Supply Chain Security controls address package registry integrity; Data Operations controls apply to training data accessible via stolen cloud credentials; and Infrastructure Security controls apply to the GPU compute and model serving systems accessible through stolen Kubernetes and AWS credentials.

DevSecOps: Automation

CSA's Six Pillars of DevSecOps series addresses pipeline security and automation directly relevant to the QLNX threat model. The Automation pillar covers credential management practices in CI/CD contexts, secret scanning, and the security implications of automated software delivery processes. [14] The guidance on eliminating long-lived secrets from pipeline runner environments aligns directly with the mitigations most effective against QLNX-style credential theft.

Software Transparency and Supply Chain

CSA's work on software transparency and supply chain security, including SBOM adoption guidance and the Software Transparency: Securing the Digital Supply Chain briefing, provides the governance and process framework for the supply chain integrity verification practices recommended in response to QLNX. Establishing verified software provenance for ML dependencies reduces both the probability of supply chain injection and the detection timeline when injection does occur.

References

[1] Trend Micro Research. "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, May 2026.

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

[3] The Hacker News. "Quasar Linux RAT Steals Developer Credentials for Software Supply Chain Compromise." The Hacker News, May 2026.

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

[5] Security Affairs. "Quasar Linux RAT (QLNX): A Fileless Linux Implant Built for Stealth and Persistence." Security Affairs, May 2026.

[6] LiteLLM. "Security Update: Suspected Supply Chain Incident." LiteLLM Documentation, March 2026.

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

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

[9] Snyk. "How a Poisoned Security Scanner Became the Key to Backdooring LiteLLM." Snyk Security Research, March 2026.

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

[11] SOC Prime. "Quasar Linux (QLNX): A Supply Chain Foothold with Full RAT Capabilities." SOC Prime Threat Intelligence, May 2026.

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

[13] Cloud Security Alliance. "AI Controls Matrix (AICM)." CSA Research Artifacts, 2024.

[14] Cloud Security Alliance. "The Six Pillars of DevSecOps: Automation." CSA Research Artifacts, 2023.