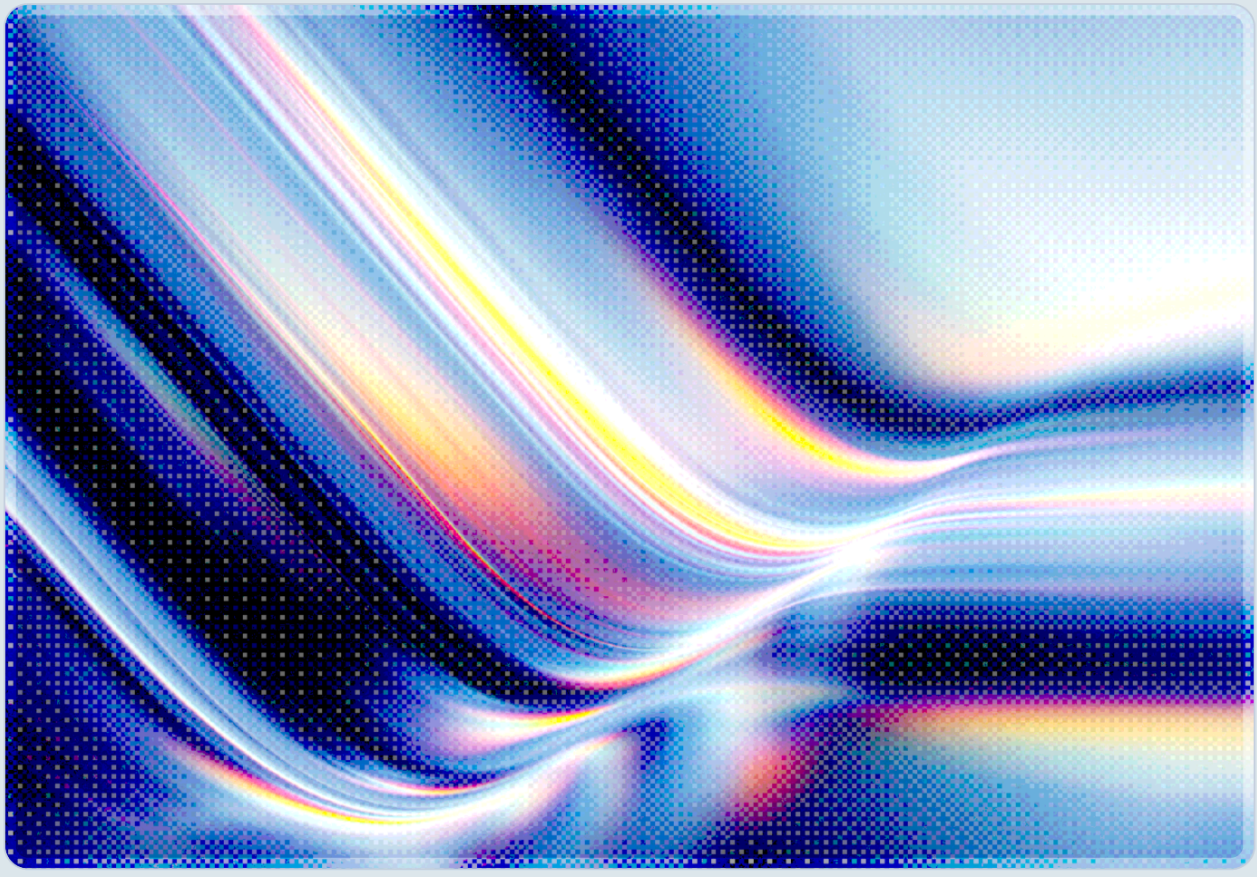
CSAI Foundation | Cloud Security Alliance

QLNX: Linux RAT Targets AI Developer Credentials

How the Quasar Linux Implant Threatens the AI/ML Software Supply Chain

2026-05-09

AI-assisted Rapid Research



CSAI

cloud security alliance®

Key Takeaways

- Trend Micro researchers disclosed Quasar Linux (QLNX) in May 2026 as a previously undocumented, full-featured Linux remote access trojan designed specifically to compromise developer and DevOps workstations and harvest software supply chain credentials [1].
- QLNX employs a two-tier rootkit—a userland LD_PRELOAD hook combined with a kernel-level eBPF component—plus a Pluggable Authentication Module (PAM) backdoor that intercepts plaintext credentials at login time, achieving stealth that only four security solutions could detect at the time of disclosure [1][2][5].
- The malware systematically targets the credential files central to AI/ML development pipelines: npm tokens, PyPI publishing keys, AWS credentials, Kubernetes configurations, Docker Hub logins, Terraform state, GitHub CLI tokens, and `.env` files that commonly store LLM API keys [1].
- The LiteLLM supply chain compromise of March 2026—in which stolen CI/CD credentials were used to backdoor a widely-installed Python package—demonstrates the concrete downstream blast radius that QLNX-class credential theft makes possible [3][4].
- Based on the threat capabilities and incident evidence described in this research note, security teams supporting AI development organizations should treat developer workstation credential hygiene as a first-order supply chain control: the LiteLLM incident demonstrates how a single maintainer's compromised token can enable downstream poisoning of widely-used packages that reach installations across the AI ecosystem.

Background

The cloud and server infrastructure underlying AI and machine learning development runs predominantly on Linux. GPU-accelerated cloud instances, CI/CD pipelines, and production inference clusters are overwhelmingly Linux-based, and many AI developers also work on Linux desktop environments. This concentration creates a large and underappreciated attack surface: the same infrastructure where researchers fine-tune models and developers publish AI frameworks also holds the credentials that control package registries, cloud environments, and container repositories. A threat actor who compromises one of these machines does not merely compromise one developer's work—they gain a potential pivot point into every downstream consumer of every package that developer maintains.

It is against this backdrop that Trend Micro researchers Aliakbar Zahravi and Ahmed Mohamed Ibrahim disclosed Quasar Linux (QLNX) in May 2026 [1]. QLNX is a previously undocumented Linux implant that researchers characterize as a "full-featured" threat: it combines remote access capabilities, advanced persistence mechanisms, a multi-layer rootkit, a backdoored authentication module, and a systematic credential harvesting engine targeting the specific files that underpin software supply chains. The malware is not a repurposed general-purpose tool; its credential-harvesting logic reflects deliberate design choices around the files that matter most to developers who publish open-source packages and manage cloud infrastructure.

The significance of QLNX is amplified by its timing. Throughout early 2026, a series of high-profile supply chain incidents demonstrated that AI infrastructure packages are high-value targets. The LiteLLM compromise in March 2026 showed that stealing a maintainer's PyPI publishing token—one file on one workstation—could enable a threat actor to distribute malicious code to a broad swath of AI deployments within hours [3][4]. QLNX represents a systematic operational capability built precisely to achieve that kind of access, at scale and with extended dwell time.

Security Analysis

Technical Architecture and Capabilities

QLNX implements a 58-command C2 framework communicated over raw TCP, HTTPS, and HTTP, giving operators a broad post-compromise toolkit covering interactive shell access, file and process management, network tunneling, and lateral movement [2]. The breadth of available commands reflects an implant designed for sustained, persistent access rather than rapid hit-and-run credential exfiltration —operators can return repeatedly over an extended engagement and direct further targeting based on what they find.

The RAT executes in a fileless manner: upon launch, QLNX copies itself into memory, re-executes from RAM, and deletes the original binary from disk [2]. It then spoofs its process name to impersonate a legitimate kernel thread—typically `kworker` or `ksoftirqd`—making it difficult to distinguish from normal system activity in a process listing. The implant also wipes system logs and clears forensic environment variables to obstruct post-incident investigation.

For persistence, QLNX implements seven distinct methods—systemd unit files, crontab entries, init.d scripts, XDG autostart entries, `.bashrc` shell injection, LD_PRELOAD injection, and self-respawning if terminated—ensuring that reboots or manual process kills are unlikely to achieve clean removal [2][5].

This redundancy is consistent with operational tradecraft designed to ensure continuity if any single persistence mechanism is discovered and removed.

The rootkit architecture departs from conventional malware practice in a technically significant way: rather than bundling a pre-compiled kernel module, QLNX carries embedded C source code for both its PAM backdoor and its LD_PRELOAD rootkit as string literals within the binary itself. On the target host it invokes `gcc` to compile these components locally, then deploys them dynamically [2]. The LD_PRELOAD component intercepts system calls in userland to hide files, ports, and processes associated with the implant, while the kernel-level eBPF component provides deeper concealment that persists even against tools that bypass userland libraries. This compile-on-target approach sidesteps signature detection of pre-built binaries and adapts the compiled artifacts to the specific kernel and library versions present on the victim machine.

Pluggable Authentication Module hooks sit at the authentication layer of Linux systems, intercepting credentials at the moment of login and SSH session initiation—before they are hashed or processed by any higher-level security control. QLNX's PAM module captures plaintext passwords and transmits them to the attacker's C2 infrastructure, creating a live credential feed that accumulates new secrets every time the compromised developer authenticates to any service [1][2].

Credential Harvesting and AI/ML Targeting

QLNX's credential harvesting engine systematically seeks out configuration and token files that grant access to software publishing pipelines and cloud infrastructure. The targeting is specific enough to indicate that the implant was engineered for this class of victim. The table below maps the harvested files to the access they represent and to their particular significance for AI/ML organizations.

Credential File	Access Granted	AI/ML Relevance
`.npmrc`	npm registry publishing tokens	AI frameworks and tooling distributed via npm
`.pypirc`	PyPI API keys for package publishing	AI/ML Python ecosystem (LLMs, agents, data science tools)
`.git-credentials`	Git repository authentication	Source code and model weights hosted on GitHub/GitLab
`~/.aws/credentials`	AWS IAM access keys	SageMaker, Bedrock, S3 training data, GPU instances

Credential File	Access Granted	AI/ML Relevance
`~/.kube/config`	Kubernetes cluster admin tokens	Model serving infrastructure, inference clusters
`~/.docker/config.json`	Docker Hub registry credentials	AI model container images
`.vault-token`	HashiCorp Vault root or service tokens	Secrets management for AI application backends
Terraform credentials	Cloud infrastructure provisioning	GPU cluster provisioning, AI platform deployments
GitHub CLI tokens	Repository and Actions API access	CI/CD pipelines, automated model publishing
`.env` files	Application-level secrets	**LLM API keys** (OpenAI, Anthropic, Cohere, etc.)

The `.env` file targeting is the entry that most directly threatens AI/ML organizations beyond the package publishing dimension. Among AI developers, it is common practice to store LLM API keys—including OpenAI, Anthropic, and cloud AI service keys—in `.env` files at the root of project directories, a convention actively encouraged by most framework documentation and development tutorials. These keys carry both financial exposure (API usage billed to the account) and intelligence value (conversation history and fine-tuning data accessible via the API). QLNX's harvesting of `.env` files means that an AI developer's API credentials are treated as a first-class target alongside their package publishing tokens.

The AWS credential theft is equally consequential. AI and machine learning workloads make disproportionate use of cloud GPU resources, S3 buckets containing training datasets, and managed AI services such as SageMaker and Bedrock. AWS access keys with broad IAM permissions—frequently present on developer workstations where convenience can take precedence over least-privilege—enable an attacker to exfiltrate proprietary training data, abuse cloud GPU capacity for their own inference workloads, or modify model serving infrastructure.

Evasion and Detection Challenges

At the time of Trend Micro's disclosure, only four security solutions flagged QLNX's binary as malicious [1][2][5]. This extremely low detection rate reflects several compounding factors. The fileless execution model eliminates the on-disk artifact that endpoint detection and response (EDR) tools most readily identify. The compile-on-target rootkit approach means the loaded shared objects and kernel modules are built fresh for each victim and lack the signatures of known malware. Process name spoofing as a kernel thread places the implant in a category of processes that many security tools treat as inherently trusted. Finally, the eBPF-based concealment operates at a layer below most user-space security tools, making the implant invisible to process enumeration and network monitoring tools that do not explicitly account for eBPF-based hiding.

This detection gap has direct operational significance: at the time of disclosure, an organization's existing endpoint security stack was unlikely to catch an active QLNX infection without behavioral detection capabilities and specific hunting rules for the indicators Trend Micro has published. Security teams should verify whether their EDR vendors have issued updated detection coverage since the disclosure before concluding that their current tooling provides adequate protection.

The Developer Workstation as Supply Chain Control Point

The LiteLLM supply chain compromise of March 2026 provides a concrete illustration of what QLNX-class credential theft enables at scale. In that incident, the threat actor group TeamPCP compromised the Trivy container scanning tool by force-pushing malicious commits to 76 of 77 `trivy-action` GitHub Action tags between March 19 and March 23, 2026 [3]. LiteLLM's CI/CD pipeline used Trivy and inadvertently exfiltrated its PyPI publishing credentials to the attacker's infrastructure. TeamPCP then published malicious versions `1.82.7` and `1.82.8` of the LiteLLM package to PyPI, embedding a multi-stage credential stealer in the proxy server code that fired on every import. The malicious versions remained available for approximately 40 minutes before PyPI quarantined them—long enough to be installed by a meaningful share of the package's user base across the AI ecosystem [3][4][5].

The LiteLLM incident used a CI/CD pipeline compromise as its credential-theft vector. QLNX is designed to achieve the same outcome from the opposite direction: rather than attacking the CI/CD infrastructure, it implants itself on the developer's workstation and harvests credentials directly. A developer whose `.pypirc` is exfiltrated by QLNX is functionally indistinguishable, from the attacker's perspective, from the CI/CD token compromise that enabled the LiteLLM attack.

The mechanism is functionally identical: a valid PyPI API key, however obtained, is sufficient to publish malicious code under the compromised maintainer's package name. The downstream blast radius scales with the package's install base—in LiteLLM's case, its position as one of the most widely-installed Python

LLM proxy frameworks meant that even a brief window of availability produced significant exposure; for maintainers of smaller packages, proportionally less, but still consequential for any downstream dependency. This equivalence underscores a structural vulnerability in the AI/ML open-source ecosystem. Package registries authenticate publishers by credential, not by identity or behavior. A valid PyPI API key obtained from a compromised workstation is as good as a key obtained by any other means. The same is true for npm, Docker Hub, and GitHub Actions secrets. Developer credential hygiene is therefore not merely a personal security matter; it is a supply chain control that directly protects the integrity of every package the developer maintains.

Recommendations

Immediate Actions

AI development organizations should begin credential rotation and hardening immediately, without waiting for confirmation of active QLNX compromise. Given the implant's low detection rate at the time of disclosure, the absence of an alert from current endpoint security tooling provides limited assurance.

Rotating package publishing credentials is the highest-priority step. PyPI API tokens and npm authentication tokens should be regenerated, and the potentially harvested credentials revoked. Publishing pipelines should be migrated to scoped, project-specific tokens rather than account-level keys where possible, limiting the blast radius of any future compromise. AWS, GitHub, and Kubernetes credentials present on developer workstations should similarly be audited and rotated on a regular schedule, with workstation-resident credentials scoped to the minimum permissions required.

Organizations should deploy specific threat hunting for QLNX indicators of compromise. Trend Micro's disclosure includes behavioral and network indicators; security teams should implement hunting rules to search for processes named `kworker` or `ksoftirqd` that exhibit unexpected network activity, LD_PRELOAD environment variable modifications in login shells, unexpected `gcc` compilation events on developer workstations, and outbound connections to infrastructure not associated with normal development workflows [1].

Endpoint detection should be evaluated for eBPF-aware behavioral monitoring. Conventional EDR tools that rely on userland hooks may be blind to QLNX's eBPF concealment layer. Security teams should confirm whether their deployed endpoint agents have eBPF-aware detection capabilities and engage vendors about detection coverage for this specific threat class.

Short-Term Mitigations

Moving PyPI and npm publishing credentials out of developer workstations entirely is the most structurally effective mitigation. Automated publishing should be handled exclusively by CI/CD systems with ephemeral, scoped credentials generated per-pipeline-run and not persisted to disk. This architecture ensures that even a fully compromised developer workstation cannot yield credentials sufficient to publish malicious packages. PyPI's Trusted Publishing and npm's Provenance feature both use OpenID Connect to authenticate publishing workflows from CI/CD environments rather than long-lived API tokens, eliminating the static credential file that QLNX targets.

For AI/ML API keys stored in `.env` files, organizations should implement secrets management practices that eliminate workstation-resident plaintext secrets. Tools such as HashiCorp Vault, AWS Secrets Manager, or environment-variable injection from a secrets vault at container startup provide access to credentials at runtime without persisting them to developer-accessible files. Where `.env` files cannot be eliminated in the short term, they should be excluded from version control, encrypted at rest, and audited for scope—many developers accumulate broad API keys that should instead be scoped to specific projects or environments.

PAM hardening can limit QLNX's ability to harvest authentication credentials even after implant deployment. Requiring hardware-backed authentication (YubiKey or similar FIDO2 hardware tokens) for SSH access prevents PAM hooks from capturing reusable plaintext passwords, since the authentication exchange does not involve a password that can be intercepted.

Strategic Considerations

AI organizations that maintain high-download-count packages on PyPI, npm, or similar registries should consider their package maintainers as a population of high-value targets who merit enterprise endpoint security management, regardless of whether they are full-time employees or contracted contributors. Open-source ecosystem security has treated maintainer workstations as personal devices, but QLNX and the LiteLLM incident together demonstrate that this assumption creates an asymmetric vulnerability. Threat actors need only compromise one maintainer, while defenders must protect an entire population—which underscores why registry-level controls (signature verification, anomaly detection on publishing events) and rapid incident response are essential complements to workstation hardening.

Software bill of materials (SBOM) and package integrity controls—including Sigstore code signing for Python packages—provide a verification layer that can detect unauthorized packages even when publishing credentials are compromised. If every QLNX-targeted package were signed with a hardware-backed key, a credential theft that granted PyPI token access would still be insufficient to publish a

package that passes downstream signature verification. Sigstore adoption in the AI/ML ecosystem has been gradual, and its continued maturation represents a meaningful structural improvement to supply chain integrity.

Organizations should also evaluate the security posture of their CI/CD pipelines against the Trivy action compromise pattern that enabled the LiteLLM incident. Any CI/CD workflow that executes third-party GitHub Actions, particularly those from security tooling vendors, and that has access to secrets including PyPI tokens, should be audited. Actions should be pinned to commit SHAs rather than mutable tags to prevent tag force-push attacks of the type TeamPCP employed.

CSA Resource Alignment

The threat profile described in this research note maps to several dimensions of CSA's published security frameworks and guidance.

CSA's MAESTRO framework for agentic AI threat modeling explicitly addresses the software supply chain as an attack surface for AI systems [6]. QLNX represents a concrete instantiation of MAESTRO's supply chain compromise threat category: an attacker who harvests a developer's package publishing credentials can inject malicious behavior into an AI agent's tool dependencies or model serving infrastructure without exploiting any weakness in the AI system itself. MAESTRO guidance recommends that AI system architects treat third-party package dependencies as untrusted until independently verified, with integrity controls applied at each dependency ingestion point.

The AI Controls Matrix (AICM), CSA's superset control framework derived from the Cloud Controls Matrix (CCM), includes supply chain integrity and developer credential management domains that are directly applicable here [7]. AICM controls in the Identity and Access Management domain specify that credentials used in automated publishing pipelines should be scoped, rotated, and managed through dedicated secrets infrastructure—precisely the control gap that QLNX exploits. Organizations seeking a structured framework for addressing the developer credential hygiene recommendations above can map their remediation work to the relevant AICM control identifiers.

CSA's Zero Trust guidance is also relevant: QLNX's effectiveness depends in part on the implicit trust that Linux systems extend to authenticated sessions and to processes bearing known-good names. Zero Trust principles applied to developer workstations—continuous behavioral verification, micro-segmentation of development environments, and least-privilege scoping of credentials—reduce the value of any single compromised workstation by limiting lateral movement and credential scope [8].

Organizations implementing Zero Trust architectures for their AI development environments are materially better positioned against QLNX-class threats than those relying on perimeter security and endpoint agent detection alone.

Finally, CSA's STAR program provides a framework for assessing the security posture of cloud providers and AI platform vendors that supply infrastructure credentials to developer workstations. Organizations can leverage STAR assessments to evaluate whether their cloud AI service providers implement credential hygiene and anomaly detection controls consistent with the threat landscape described here.

References

[1] Zahravi, Aliakbar, and Ahmed Mohamed Ibrahim. "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.

[2] Kovacs, Eduard. "Sophisticated Quasar Linux RAT Targets Software Developers." SecurityWeek, May 2026.

[3] Evenden, Ian, and Ruben Circelli. "LiteLLM and Telnyx compromised on PyPI: Tracing the TeamPCP supply chain campaign." Datadog Security Labs, March 2026.

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

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

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

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

[8] Cloud Security Alliance. "Zero Trust Guidance for Critical Infrastructure." CSA, 2024.