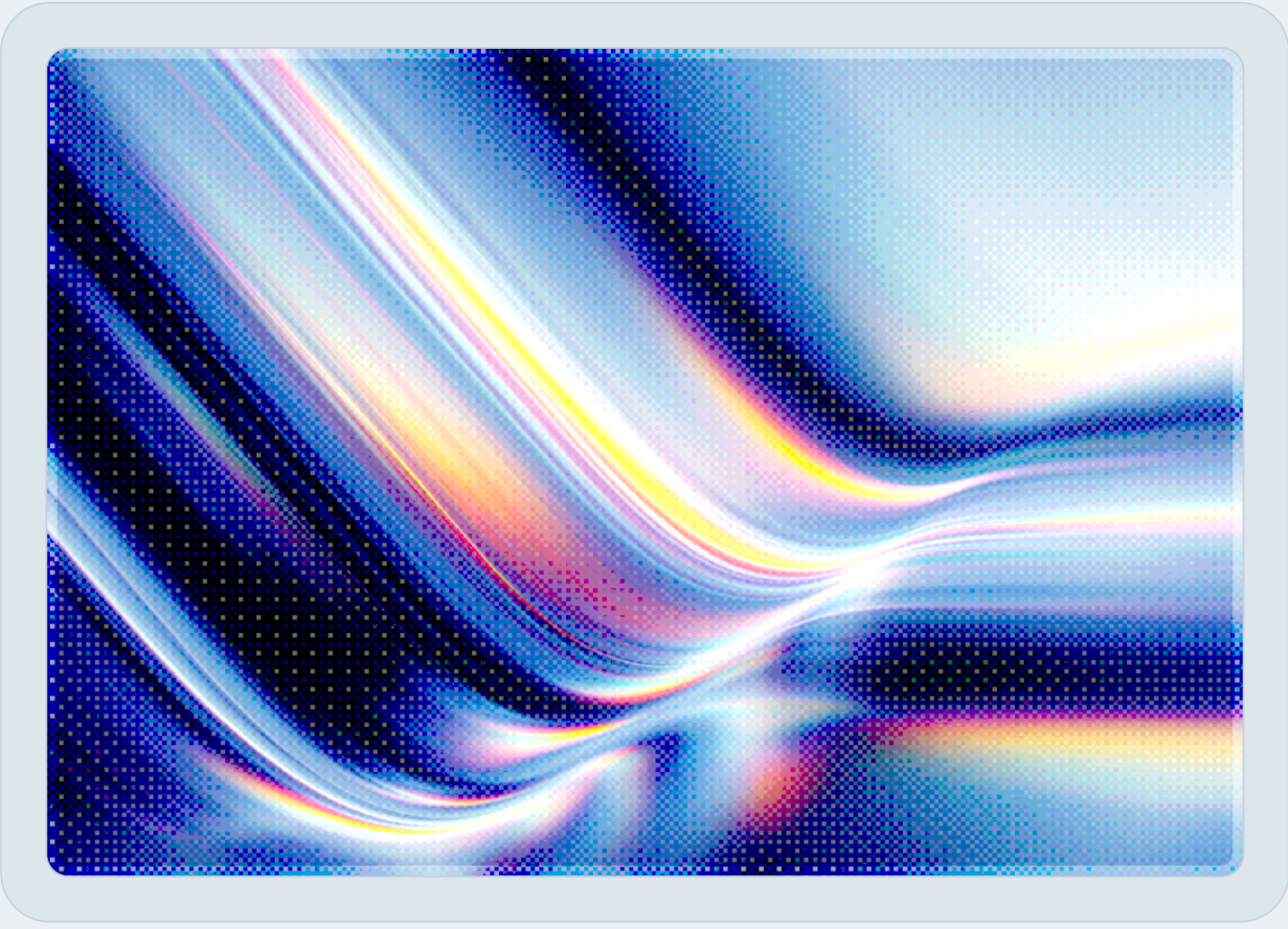
CSAI Foundation | Cloud Security Alliance

Quasar Linux RAT: Credential Harvesting as Supply Chain Attack Vector

How QLNX Exploits Developer Workstations to Compromise Software Ecosystems

2026-05-10

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- A previously undocumented Linux implant tracked as Quasar Linux (QLNX) specifically targets developer workstations, harvesting the credentials that power software build and deployment pipelines—NPM tokens, PyPI keys, Git credentials, cloud provider keys, and Kubernetes configurations.
- QLNX is engineered for persistence and concealment: it executes entirely from memory using `memfd_create`, establishes no fewer than seven redundant persistence mechanisms, and hides itself through a dual-layer rootkit combining userland `LD_PRELOAD` hooks with a kernel-level eBPF component.
- The attack chain is deliberately sequential—arrive, erase from disk, persist, hide, harvest—culminating in credential theft that enables downstream supply chain compromise at ecosystem scale.
- At the time of Trend Micro's disclosure in May 2026, only four security products detected QLNX as malicious [1], underscoring that endpoint protection designed for enterprise servers may fail to catch threats targeting developer laptops.
- The LiteLLM supply chain incident in March 2026 illustrates the real-world consequence of this attack class: stolen pipeline credentials enabled the publication of backdoored Python packages; PyPI quarantined the versions within three hours of upload, but not before approximately 40,000 downloads had occurred from a package with a baseline daily download rate of 3.4 million [2, 3, 8].
- Organizations should treat developer workstations as high-value targets, enforce short-lived credentials in CI/CD pipelines, and deploy endpoint monitoring capable of detecting eBPF rootkit activity and `memfd_create`-based fileless execution.

Background

Software development infrastructure has emerged as a high-value target class for advanced threat actors. Where a compromised enterprise server may yield access to a single organization's data, a compromised developer workstation—particularly one belonging to a maintainer of a widely used open-source package—can yield credentials capable of poisoning software that reaches millions of downstream

users. This threat model is not new—the SolarWinds SUNBURST compromise in 2020, the 3CX supply chain attack in 2023, and the XZ Utils backdoor in 2024 each demonstrated its operational reality—but 2025 and 2026 have seen a specific intensification of attacks targeting developer workstations and the credentials they hold.

Trend Micro researchers publicly disclosed Quasar Linux RAT (internally tracked as QLNX) in May 2026 after analyzing a previously undocumented Linux implant with an unusually coherent focus: every capability within the malware serves the goal of establishing a durable, invisible foothold on a developer's machine from which credentials can be harvested and exfiltrated [1]. Unlike ransomware or cryptojackers, QLNX does not announce its presence. Its operators appear to be playing a longer game, treating compromised developer environments as access points into the software supply chain rather than as endpoints to monetize directly.

The timing of QLNX's emergence is notable. The March 2026 compromise of the LiteLLM PyPI package —in which threat actors used stolen CI/CD credentials to publish backdoored versions of an AI gateway library downloaded at a rate of roughly 3.4 million times per day—demonstrated precisely the downstream damage that stolen developer credentials enable [2, 3]. While researchers have not attributed QLNX to the same actors responsible for the LiteLLM incident, the two events illustrate a converging threat landscape where developer credential theft is the pivot point between a single machine compromise and ecosystem-scale supply chain risk.

Security Analysis

Technical Architecture and Delivery

QLNX arrives as a Linux ELF binary, though it takes immediate steps to erase evidence of its binary presence. Upon execution, the implant copies itself into a RAM-backed anonymous file descriptor created via `memfd_create`, deletes the original on-disk binary, and re-executes itself entirely from memory using `execveat` (or, as a fallback, `/proc/self/fd/<memfd>`) [4, 5]. This fileless execution model means conventional file-based scanning and hash-based detection have no persistent artifact to act on. The implant then spoofs its process name to impersonate legitimate Linux kernel threads such as `kworker` or `ksoftirqd`, blending into the background process table in a way that standard process enumeration tools would not flag.

QLNX also profiles the environment before proceeding. The malware detects containerized runtime environments, suggesting the operators are aware that developer workstations may run Docker or similar environments and have built in logic to adapt behavior accordingly. It wipes relevant system and authentication logs before settling into its operational posture [1, 5].

The exact initial delivery vector has not been confirmed by researchers as of this writing. In comparable developer-targeted campaigns, observed vectors have included malicious or trojanized development tools, typosquatting packages in public registries, social engineering via platform-specific developer communities, and exploitation of vulnerabilities in IDE extensions or build tooling [2, 5].

Dual-Layer Rootkit

QLNX employs a two-tier rootkit architecture designed to make the implant invisible at both the operating system and kernel level. The first tier operates in userland through `LD_PRELOAD`, deploying a shared library to `/etc/ld.so.preload` that intercepts libc function calls system-wide. This hook layer conceals the malware's files, processes, and artifacts from standard system utilities. Critically, QLNX carries the source code for both this LD_PRELOAD module and its PAM backdoor as embedded string literals within its binary, dynamically compiling them on the target host using the system's own `gcc` installation [1]. This approach avoids shipping precompiled artifacts that security tooling might recognize while ensuring the rootkit is built for the exact architecture and kernel of the compromised host.

The second tier operates at the kernel level using the Linux eBPF subsystem. eBPF programs, originally designed for network monitoring and performance tracing, are increasingly abused by advanced malware to achieve kernel-level visibility and manipulation without requiring a traditional loadable kernel module. QLNX's eBPF component manages kernel BPF maps to conceal specific process IDs, file paths, and network ports, creating a stealth layer that persists even when the userland rootkit might otherwise be detected or removed [5, 6]. Unlike rootkits that rely solely on userland hooks, QLNX's eBPF component operates at the kernel level, making detection possible only through kernel-aware tooling—a capability gap that many enterprise security products were not designed to address on developer endpoints.

PAM Backdoor and Credential Interception

QLNX includes two distinct implementations of a Pluggable Authentication Module (PAM) backdoor that intercept credentials at the authentication layer—before they are hashed or otherwise protected. The first implementation harvests plaintext passwords from any authentication event on the system, contains a master password mechanism allowing the operator to authenticate as any user, and records outbound

SSH session data including remote hostnames and the credentials used [1, 4]. The second PAM module loads into dynamically linked processes and extracts the service name, username, and authentication token at the moment of each authentication call.

This approach is particularly effective against developers because their workstations serve as authentication hubs: they SSH to remote servers and CI/CD runners, authenticate to cloud provider CLIs, interact with container registries, and issue signed commits to version control systems. Every one of these authentication events passes through PAM on a standard Linux workstation, meaning QLNX's backdoor can passively accumulate a comprehensive map of the developer's access without requiring any active probing or lateral movement.

Credential Harvesting Scope

Beyond intercepting live authentication events, QLNX deploys a dedicated credential harvester that systematically scans the filesystem for files whose contents provide direct access to publishing pipelines and cloud infrastructure [1, 5]. The harvester targets `.npmrc` files containing NPM registry tokens, `.pypirc` files containing PyPI upload credentials, `.git-credentials` and GitHub CLI token files, `.aws/credentials`, `.kube/config` files containing Kubernetes cluster access, `.docker/config.json`, HashiCorp Vault token files, Terraform state credentials, and `.env` files that commonly contain application secrets and API keys.

This targeting scope is almost certainly not accidental. Each file type maps to a specific downstream capability available to an attacker who obtains its contents. An NPM token enables publishing to the npm registry; a PyPI key enables pushing packages to the Python Package Index; a `.kube/config` grants cluster-level access that can span development, staging, and production Kubernetes environments. The harvested credentials can be used to push malicious packages to public registries at scale, inject backdoors into build artifacts, access cloud infrastructure where source code and deployment pipelines reside, or pivot into additional environments connected to the same CI/CD ecosystem.

Persistence Architecture

QLNX establishes no fewer than seven distinct persistence mechanisms, designed so that neither a system reboot, a user logout, a deleted scheduled job, nor the removal of a single configuration entry will terminate the implant [1, 5]. Documented mechanisms include systemd service and timer units, crontab entries at both user and system level, shell initialization file injection (`.bashrc` and related files), desktop environment autostart entries, and traditional init script registration. The combination creates a

redundant persistence web that mirrors the defensive assumption that any single persistence vector will eventually be discovered and removed, while the others remain intact and can re-establish the full foothold.

Command and Control

Once established, QLNX initializes 58 distinct command handlers and connects to its command and control infrastructure over a custom TLS-based protocol, with HTTP and HTTPS as fallback channels [6]. The initial beacon transmits a detailed system profile to the operator, including machine fingerprint, hostname, current privilege level, network configuration data, and geolocation—giving the operator sufficient information to prioritize which compromised hosts represent the highest-value credential targets.

Supply Chain Consequences: The LiteLLM Precedent

The LiteLLM incident provides a concrete illustration of what QLNX-class credential theft enables at scale. On March 24, 2026, threat actors published backdoored versions of the `litellm` Python package (versions 1.82.7 and 1.82.8) after obtaining the project maintainer's PyPI publishing credentials through a compromised Trivy GitHub Action in LiteLLM's CI/CD pipeline [2, 3, 7]. The malicious packages contained a `.pth` file that executed automatically on every Python process startup, deploying a three-stage payload: a credential harvester reportedly targeting more than 50 categories of secrets, a Kubernetes lateral movement toolkit, and a persistent backdoor. PyPI quarantined the packages approximately three hours after upload, but not before over 40,000 downloads had occurred from a package with a baseline daily download rate of approximately 3.4 million [2, 8].

QLNX's capability set represents a plausible upstream enabler for attacks of this pattern. Where the LiteLLM incident leveraged a compromised CI/CD tool, a developer workstation infected with QLNX would expose the same PyPI credentials—plus the SSH keys, cloud credentials, and container registry tokens needed to extend the attack further into infrastructure. The attack surface is not limited to any single package ecosystem; a single well-placed developer credential harvest can provide the access needed to poison packages across multiple registries simultaneously.

Recommendations

Immediate Actions

Developer workstations must be treated with the same security rigor applied to production servers. Security teams should conduct an immediate audit of long-lived credentials stored in common locations (`.npmrc`, `.pypirc`, `.aws/credentials`, `.kube/config`, `.docker/config.json`, `.env` files, and similar paths) on developer machines and rotate any credentials that cannot be confirmed clean. Where those credentials include NPM or PyPI publishing tokens, package registries should be checked for unauthorized publications during the period when the credentials may have been exposed.

Teams should deploy endpoint detection and response (EDR) tooling capable of detecting `memfd_create`-based execution and eBPF program loading on developer Linux workstations. At the time of Trend Micro's disclosure, only four security products detected QLNX as malicious [1], suggesting that solutions not specifically instrumented for `memfd_create` and eBPF activity on developer endpoints may miss this class of threat. Additionally, audit logging for PAM authentication events should be enabled and forwarded to a centralized SIEM platform, creating an independent record of authentication activity that an `LD_PRELOAD`-based rootkit cannot intercept and delete.

Short-Term Mitigations

CI/CD pipelines should be migrated from long-lived static credentials to short-lived, scoped tokens wherever registry and cloud provider APIs support them. NPM Granular Access Tokens and PyPI Trusted Publishers (OIDC-based publishing from GitHub Actions, GitLab CI, or similar) eliminate the need for persistent publishing tokens stored on developer machines entirely [10, 11]. For cloud credentials, IAM roles with assume-role mechanisms and instance metadata credentials should replace static access keys in all pipeline contexts.

Package registries that support them should have multi-factor authentication enforced on all publishing accounts, and CI/CD service accounts should be separated from individual developer accounts so that a single workstation compromise does not grant publishing rights. Software Bill of Materials (SBOM) generation should be integrated into build pipelines, creating a verifiable artifact of what went into each published package and providing a baseline against which suspicious modifications can be detected.

System integrity monitoring on developer workstations—checking for modifications to `/etc/ld.so.preload`, unexpected entries in systemd unit directories, and eBPF program loading by non-root processes—provides early warning for the rootkit deployment phase of QLNX. Indicators of compromise published by Trend Micro in their QLNX disclosure should be ingested into threat intelligence platforms and used to hunt for retrospective activity across the developer fleet [1].

Strategic Considerations

The QLNX threat model exposes a structural gap in most organizations' security architecture: developer workstations occupy a privileged position in the software delivery chain but are typically governed by more permissive security policies than production systems. The credentials accessible on a senior developer's laptop may provide more effective access to production infrastructure than a direct attack on that infrastructure would. Security programs should formally classify developer workstations—particularly those belonging to maintainers of widely used packages or owners of CI/CD service accounts—as high-value targets and apply commensurate controls.

At the ecosystem level, package registry operators and organizations that depend on open-source packages should expand adoption of sigstore-based artifact signing. When each published package version is cryptographically signed by a verifiable build pipeline identity, registry consumers can detect packages published outside the expected pipeline even if the credentials used were legitimately obtained. Dependency management tooling should alert on unexpected version bumps in high-criticality packages, creating a tripwire that may catch supply chain injections before they propagate into production builds.

CSA Resource Alignment

The threat posed by QLNX maps directly to guidance across several CSA frameworks and working group publications.

The CSA Cloud Controls Matrix (CCM) addresses the controls most directly relevant to this threat under the Identity and Access Management (IAM) domain, which covers the management, rotation, and least-privilege scoping of credentials—the primary target category for QLNX's harvester. The Threat and Vulnerability Management (TVM) domain covers the endpoint detection, patching, and IOC monitoring disciplines necessary to detect and respond to an active QLNX deployment. The Infrastructure and Virtualization Security (IVS) domain is relevant to organizations whose compromised developer credentials expose Kubernetes cluster access or cloud provider infrastructure.

CSA's Software Transparency: Securing the Digital Supply Chain [12] establishes the case for SBOM generation, package signing, and CI/CD pipeline security controls that represent the strategic mitigations most effective against the downstream consequences of developer credential theft. Organizations building or consuming AI infrastructure—such as the LiteLLM AI gateway targeted in March 2026—should additionally consult CSA's MAESTRO threat modeling framework [13], which addresses the agentic AI attack surface including the pipeline security of AI tools, model serving infrastructure, and the credential chains that connect AI applications to cloud backends.

CSA's Zero Trust guidance [14] is applicable to both the workstation-level controls and the CI/CD pipeline architecture recommended above. Zero Trust principles—verify explicitly, use least-privilege access, assume breach—translate directly into the short-lived token, MFA enforcement, and continuous authentication monitoring practices that reduce the blast radius of a successful QLNX credential harvest.

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

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

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

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

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

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

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

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

[10] npm Docs. "Creating and Viewing Access Tokens." npm, 2026.

[11] PyPI Docs. "Publishing to PyPI with a Trusted Publisher." Python Packaging Authority, 2026.

[12] Cloud Security Alliance. "Software Transparency: Securing the Digital Supply Chain." CSA, 2022.

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

[14] Cloud Security Alliance. "Zero Trust Guiding Principles." CSA.