Hive Pro

Threat Level

Red

Hiveforce Labs

THREAT ADVISORY

ATTACK REPORT

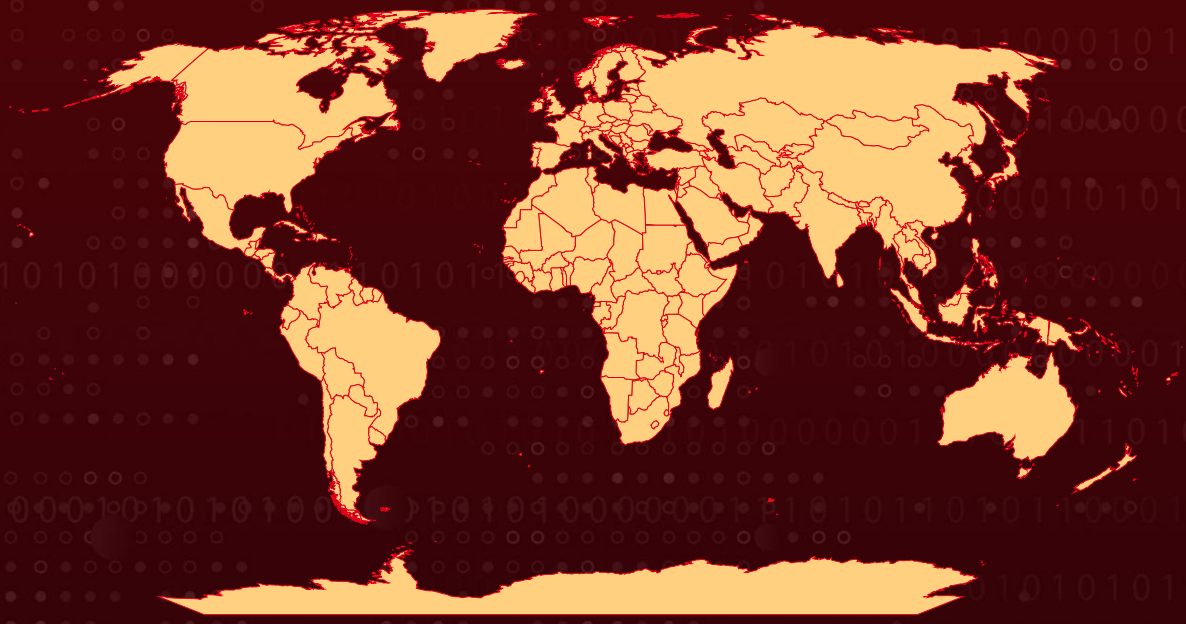
QLNX Unmasked: Advanced Linux Malware Targets Developers and Cloud Infrastructure

Date of Publication	Admiralty Code	TA Number
May 12, 2026	A1	TA2026127

Summary

First Seen: May 2026
Targeted Regions: Global
Targeted Platforms: Linux (x86-64)
Targeted Products: npm, PyPI, GitHub, AWS, Docker, Kubernetes, Git, HashiCorp Vault, Terraform
Targeted Industries: Software Development, DevOps, Cloud Infrastructure
Malware: Quasar Linux (QLNX)
Attack: Quasar Linux (QLNX) is a previously undocumented, full-featured Linux remote access trojan (RAT) designed to establish a fileless, persistent foothold on developer and DevOps workstations. The malware combines rootkit capabilities, a PAM backdoor for plaintext credential interception, and a comprehensive credential harvesting module targeting software supply chain assets such as NPM tokens, PyPI credentials, AWS keys, Kubernetes configs, and Docker credentials. Its ultimate objective is to compromise developer publishing pipelines and enable supply-chain attacks by trojanizing packages or pivoting into cloud production environments.

Attack Regions



Targeted Non-Targeted

Attack Details

#1

Quasar Linux (QLNX) is a sophisticated Linux malware strain designed to target developer workstations and DevOps environments. While researchers have not yet identified the initial infection vector, the malware quickly shifts into a stealth-focused deployment once executed. QLNX uses the Linux memfd_create syscall to load itself directly into memory, re-execute from the in-memory copy, and erase the original binary from disk, effectively eliminating its on-disk footprint. To avoid repeated execution loops, it relies on the MFD_RE environment variable as a safeguard, while older systems that do not support memfd execution are handled through a fallback mechanism using /proc/self/fd. The malware further masks its activity by impersonating legitimate kernel thread names such as [kworker/0:0], [ksoftirqd/0], and [rcu_sched], while also clearing forensic environment variables that could expose how it was launched.

#2

After gaining a foothold, QLNX performs extensive reconnaissance on the compromised system to assess its capabilities and security posture. It checks for root access, kernel version, SELinux enforcement, containerized environments, GCC compiler availability, X11 display access, and accessibility to /dev/input devices. Based on these findings, the malware selectively activates different modules. Researchers identified 58 separate command handlers tied to its command-and-control (C2) framework, enabling the malware to execute a broad range of operations. Persistence is maintained through seven separate mechanisms spanning both user and system levels, including systemd services, cron jobs, SysVinit scripts, XDG autostart entries, .bashrc modifications, and LD_PRELOAD shared library injection. Every persistence component contains the QLNX_MANAGED marker for internal tracking. Among these methods, the LD_PRELOAD technique is particularly dangerous, as it forces the malware to relaunch whenever dynamically linked applications execute, allowing it to survive even after the main process is terminated.

#3

To remain hidden, QLNX employs a dual-layer rootkit architecture that combines both userland and kernel-level concealment. Its LD_PRELOAD rootkit is compiled directly on the victim system using embedded C code and hooks critical libc functions such as readdir, stat, open, and access to hide malware components, credential logs, PAM backdoors, and running processes. In parallel, an eBPF-based rootkit manipulates kernel-level BPF maps to conceal processes, files, and network ports from monitoring tools like ps, ls, and netstat. This kernel component requires Linux kernel 4.18 or newer, along with root privileges. QLNX also deploys a PAM inline-hook backdoor capable of intercepting plaintext credentials during authentication events. The module includes a hardcoded master password (O$$f$QtYJK) that can bypass authentication controls and logs outbound SSH activity to monitor lateral movement. Stolen credentials are XOR-encrypted and stored in hidden log locations such as /var/log/.ICE-unix and /var/log/.Test-unix, while an additional PAM credential harvester logs usernames, services, and authentication tokens into /tmp/.pam_cache.

#4 When instructed by its operators, QLNX can perform large-scale credential and data harvesting operations. It extracts SSH private keys, browser-stored credentials from Chrome, Chromium, and Firefox, cloud and developer configuration files, Git credentials, Kubernetes and Docker configurations, Vault tokens, Terraform secrets, shell histories, X11 clipboard data, and even /etc/shadow when running with elevated privileges. Communication with its operators occurs through a custom binary protocol transported over TLS, HTTPS, or HTTP, all using the same framing structure identified by the QLNX magic header. Beacon traffic includes system profiling information, geolocation data obtained from ip-api.com, and unique machine fingerprints derived from hardware identifiers.

#5 Beyond credential theft, QLNX incorporates an extensive set of offensive capabilities, including TCP tunneling, SOCKS proxying, port forwarding, packet capture, port scanning, SSH-based lateral movement, in-memory shared object execution, and process injection using ptrace and /proc/pid/mem. The malware also supports Beacon Object File (BOF/COFF) execution, screenshot capture, keylogging, clipboard monitoring with SHA256-based deduplication, real-time filesystem monitoring via inotify, timestomping, and system log wiping. One of its most advanced features is a peer-to-peer mesh networking capability that allows infected systems to operate as resilient relay nodes, creating a distributed infrastructure for command-and-control communications and making disruption significantly more difficult.

Recommendations

Audit LD_PRELOAD and PAM Configurations: Regularly inspect /etc/ld.so.preload for unauthorized entries and verify the integrity of PAM modules under /usr/lib/ and /lib/security/. Any unrecognized shared objects such as libsecurity_utils.so.1, pam_security.so, or .libpam_cache.so should be treated as indicators of compromise.

Monitor for Fileless Execution Patterns: Deploy detection rules for memfd_create and execveat syscalls, which QLNX uses for in-memory execution. Alert on processes executing from /proc/self/fd or memfd: paths, as legitimate applications rarely use this pattern.

Restrict GCC Access on Production and Developer Systems: QLNX requires GCC to compile its rootkit and PAM backdoor modules on the target host. Limit compiler availability to build environments only and remove development toolchains from production servers and general-purpose developer workstations where possible.

Implement File Integrity Monitoring on Critical Paths: Monitor for unauthorized modifications to /etc/ld.so.preload, /etc/pam.d/, systemd service directories, crontab entries, init.d scripts, .bashrc files, and XDG autostart locations. Alert on the creation of files matching QLNX artifact patterns such as /tmp/.X -lock, /tmp/.pam_src_ , /tmp/.hide_src_ , and /var/log/.ICE-unix.

Rotate Developer Credentials and Tokens Immediately: If QLNX compromise is suspected, revoke and rotate all NPM tokens (.npmrc), PyPI credentials (.pypirc), Git credentials, AWS access keys, Kubernetes service account tokens, Docker Hub credentials, GitHub CLI tokens, HashiCorp Vault tokens, and Terraform credentials stored on affected workstations.

Enforce Multi-Factor Authentication on Package Registries: Require MFA for all publishing operations on NPM, PyPI, and other package registries to prevent attackers from using stolen tokens to push trojanized packages through compromised developer accounts.

Potential MITRE ATT&CK TTPs

Tactic	Technique	Sub-technique
Execution	T1059: Command and Scripting Interpreter	T1059.004: Unix Shell
	T1106: Native API	
Persistence	T1543: Create or Modify System Process	T1543.002: Systemd Service
	T1053: Scheduled Task/Job	T1053.003: Cron

Tactic	Technique	Sub-technique
Persistence	T1546: Event Triggered Execution	T1546.004: Unix Shell Configuration Modification
	T1574: Hijack Execution Flow	T1574.006: Dynamic Linker Hijacking
Defense Evasion	T1014: Rootkit	
	T1070: Indicator Removal	T1070.002: Clear Linux or Mac System Logs
		T1070.004: File Deletion
		T1070.006: Timestomp
	T1036: Masquerading	T1036.004: Masquerade Task or Service
	T1620: Reflective Code Loading	
	T1027: Obfuscated Files or Information	
Credential Access	T1556: Modify Authentication Process	T1556.003: Pluggable Authentication Modules
	T1555: Credentials from Password Stores	T1555.003: Credentials from Web Browsers
	T1552: Unsecured Credentials	T1552.001: Credentials In Files
		T1552.004: Private Keys
Discovery	T1082: System Information Discovery	
	T1057: Process Discovery	
	T1049: System Network Connections Discovery	
Lateral Movement	T1021: Remote Services	T1021.004: SSH

Tactic	Technique	Sub-technique
Collection	T1056: Input Capture	T1056.001: Keylogging
	T1113: Screen Capture	
	T1115: Clipboard Data	
Command and Control	T1071: Application Layer Protocol	T1071.001: Web Protocols
	T1095: Non-Application Layer Protocol	
	T1573: Encrypted Channel	T1573.002: Asymmetric Cryptography
	T1090: Proxy	T1090.001: Internal Proxy
	T1104: Multi-Stage Channels	
Exfiltration	T1041: Exfiltration Over C2 Channel	

Indicators of Compromise (IOCs)

TYPE	VALUE
SHA256	ea1d34b21b739a6bbf89b3f7e67978005cf7f3eda612cefc7eac1c8ead7c5545, 82DAA93219BA40A6E41CDF3174BA57EB5D3383D1CD805584E9954EB0200182A1, 42D0C420EB5FE181388F2E4F0B7D7C0D302971E7A06FDC1BEC481B68C8CCAE1F, C99CF0DC1EF1057D713CB082ACAF42E4DF4656809C91741752BDDCAB39BBFACA, EA89CAAB82181881D971BE312412795051F6322B105C8B9D29CFB5729FAB8D33, 417430b2d4ae8d005224a9ff5dcb4007d452338acbcbcbb62c4e8ed1a70552dd, d55549d5655e2f202e215676f4bdb0994ea08a93d15ec4ded413f64cfa7facc8

TYPE	VALUE
SHA1	b0f2c668cbdd63a871c90592b6c93e931115872e
MD5	70f70743f287a837d17c56933152a8a6
File Name	quasar-implant, libsecurity_utils.so.1, pam_security.so, libpam_cache.so, hide_src_39ZzHo.c, pam_src_51YyC3.c, pcs_a3kf9x.c
File Path	/usr/lib/libsecurity_utils.so.1, /usr/lib/.libpam_cache.so, /var/log/.ICE-unix, /var/log/.Test-unix, /tmp/.pam_cache, /tmp/.X752e2ca1-lock

References

https://www.trendmicro.com/en_us/research/26/e/quasar-linux-qlnx-a-silent-foothold-in-the-software-supply-chain.html

What Next?

At **Hive Pro**, it is our mission to detect the most likely threats to your organization and to help you prevent them from happening.

Book a Demo of **HivePro**.

REPORT GENERATED ON

May 12, 2026 • 7:50 AM

More at www.hivepro.com