INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-07-06 06:33 UTC

TeamPCP Threat Actor Conducts Ongoing Supply Chain Attacks Against Developer and Security Tools

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.0

SCC Item ID	SCC-CAM-2026-0621
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.0
Affected Products	Trivy, KICS (Checkmarx), LiteLLM (PyPI), Telnyx Python SDK (PyPI); broad developer and security toolchain ecosystem
Published	2026-07-04
Discovery Source	Gemini

Executive Summary

A threat actor group tracked as TeamPCP is conducting an active, systematic supply chain campaign targeting developer tools and security scanners distributed through PyPI and related package ecosystems. Confirmed compromised packages include LiteLLM and the Telnyx Python SDK, with the attack chain extending to security scanning tools Trivy and KICS, meaning the tools developers use to verify safety were themselves weaponized. The campaign is designed to steal cloud credentials, SSH keys, Kubernetes secrets, and corporate access tokens from developer environments and CI/CD pipelines, creating direct pathways to cloud infrastructure, source code repositories, and production systems.

Technical Analysis

TeamPCP is conducting a chained supply chain attack campaign targeting Python package ecosystems, with confirmed malicious versions published to PyPI for LiteLLM and the Telnyx Python SDK (Telnyx first-party security notice, March 2026; Datadog Security Labs attribution). The attack chain documented by Snyk illustrates a compound vector: a poisoned security scanner (Trivy) was used as a delivery mechanism to backdoor LiteLLM, meaning organizations that ran Trivy scans against their LiteLLM dependencies may have been exposed through the scanning process itself. Endor Labs reporting confirms TeamPCP activity is ongoing, with scope potentially extending beyond the publicly confirmed packages. No CVE ID is assigned to this campaign. Applicable CWEs: CWE-506 (Embedded Malicious Code), CWE-494 (Download of Code Without Integrity Check), CWE-1357 (Reliance on Insufficiently Trustworthy Component). MITRE ATT&CK coverage:

T1195.001 (Supply Chain Compromise: Compromise Software Dependencies and Development Tools), T1528 (Steal Application Access Token), T1608.002 (Stage Capabilities: Upload Tool), T1554 (Compromise Host Software Binary), T1552.004 (Unsecured Credentials: Private Keys), T1552.001 (Unsecured Credentials: Credentials In Files). Primary objective is credential harvesting, specifically cloud provider tokens, SSH private keys, Kubernetes secrets, and corporate access credentials resident in developer workstations and CI/CD pipeline environments. Patch status: Telnyx has issued a first-party security notice; affected package versions should be treated as compromised pending vendor-confirmed clean version verification. Full scope of compromised package versions has not been publicly enumerated across all affected tools.

Action Checklist

1. Step 1: Containment. Immediately audit your Python dependency manifest (requirements.txt, pyproject.toml, Pipfile.lock) and CI/CD pipeline configurations for any version of LiteLLM or the Telnyx Python SDK installed during the TeamPCP active window. Isolate any build environments, developer workstations, or CI/CD runners that installed these packages. Consult the Telnyx first-party security notice and Datadog Security Labs advisory (see Sources) for confirmed malicious version ranges before continuing operations. Per NIST AC-4 (Information Flow Enforcement), restrict outbound network access from affected build environments until investigation is complete.
2. Step 2: Detection. Audit package installation logs and CI/CD pipeline run histories for installs of LiteLLM, Telnyx Python SDK, Trivy, and KICS during the campaign window. Check for anomalous outbound network connections from build systems or developer machines to unexpected external hosts following package installation events (per NIST AU-6, Audit Record Review, Analysis, and Reporting). Review cloud provider access logs (AWS CloudTrail, GCP Audit Logs, Azure Activity Logs) for credential use from unusual source IPs, geolocations, or times. Review SSH authentication logs for private key use from unrecognized hosts. Scan Kubernetes audit logs for secret access events not initiated by known service accounts. Apply D3-LAM (Local Account Monitoring) and D3-SFA (System File Analysis) to identify unauthorized credential file reads on affected developer systems.
3. Step 3: Eradication. Remove and replace any version of LiteLLM, Telnyx Python SDK, Trivy, or KICS installed during the active campaign window. Do not reinstall from PyPI until you have verified the clean version against the vendor-confirmed hash or advisory. Rotate all credentials that were present in any environment where a compromised package was installed: cloud provider API tokens, SSH private keys, Kubernetes secrets, and any stored corporate access credentials. Apply D3-CRO (Credential Rotation) immediately across affected scopes. Per NIST CM controls and CIS 2.3 (Address Unauthorized Software), document and remediate all confirmed unauthorized package versions. Per CIS 7.4 (Perform Automated Application Patch Management), pin dependency versions to vendor-verified hashes going forward.
4. Step 4: Recovery. After credential rotation, validate that no residual unauthorized sessions are active across cloud consoles, Kubernetes clusters, SSH targets, and corporate SSO. Re-run CI/CD pipelines from clean, verified dependency states and confirm build outputs against expected hashes. Monitor cloud provider billing and resource provisioning for anomalous activity that may indicate credentials were used prior to rotation. Per NIST AU-12 (Audit Record Generation), ensure logging is active and capturing post-remediation access events. Apply D3-MFA (Multi-factor Authentication) enforcement review; verify MFA is active on all cloud provider accounts and privileged access paths exposed through compromised environments (CIS 6.3, 6.4, 6.5).
5. Step 5: Post-Incident. Conduct a lessons-learned review focused on dependency integrity verification gaps. Implement hash-pinned dependency locking for all CI/CD pipelines and require integrity verification

before package installation (CIS 2.1, Establish and Maintain a Software Inventory; CIS 7.1, Establish and Maintain a Vulnerability Management Process). This campaign exposed a specific control gap: security scanning tools themselves are not verified before use, enabling a chained attack. Implement signed artifact verification for all security tooling (D3-FMBV, File Magic Byte Verification; NIST SI controls). Subscribe to Endor Labs, Datadog Security Labs, and Snyk advisory feeds for ongoing TeamPCP campaign updates, as Endor Labs reporting indicates the campaign is active beyond currently confirmed packages.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and breach notification review immediately if AWS CloudTrail, GCP Audit Logs, or Azure Activity Logs confirm that credentials harvested from affected environments were used to access systems storing PII, PHI, PCI-scoped data, or regulated intellectual property, or if Kubernetes secrets accessed by the malware included database credentials or encryption keys for production workloads.
Recovery Notes	After credential rotation, maintain elevated CloudTrail and Kubernetes audit log alerting for a minimum of 30 days, specifically watching for use of any IAM access key IDs or service account tokens that were active during the TeamPCP campaign window — threat actors may have exfiltrated credentials before you detected the compromise and will attempt to use them after rotation is expected to have lapsed. Re-run all security scans performed by Trivy or KICS during the campaign window using verified-clean binary versions and treat prior scan results as untrusted. Monitor cloud provider billing dashboards daily for the first two weeks post-recovery, as TeamPCP credential use has been associated with compute resource abuse (cryptomining, data exfiltration staging) that may have been initiated before containment.
Forensic Artifacts	Malicious PyPI wheel files (.whl) and sdist tarballs (.tar.gz) for LiteLLM and Telnyx Python SDK cached in ~/.cache/pip/ (Linux) or %LOCALAPPDATA%\pip\Cache\ (Windows) — preserve these for SHA-256 hash comparison against advisory IOCs and YARA scanning for the exfiltration code targeting ~/.aws, ~/.kube, and ~/.ssh paths | AWS CloudTrail records for the campaign window filtered on GetSecretValue, AssumeRole, CreateAccessKey, ListBuckets, and RunInstances originating from build-host source IPs — primary evidence of credential theft and post-exfiltration use by TeamPCP infrastructure | Kubernetes audit log entries for secrets/get and secrets/list operations by service accounts or users that map to build environment identities, timestamped to within minutes of malicious package import events — confirms whether kubeconfig or in-cluster tokens were accessed | Process execution logs and environment variable snapshots (/proc//environ on Linux) from Python processes that imported litellm or telnyx during the campaign window — captures in-memory credential values and any subprocess execution (e.g., curl, wget, python -c) initiated by the malicious package post-install hooks or import-time code | DNS query logs from affected build hosts and CI/CD runners for the campaign window — TeamPCP packages beacon to attacker-controlled C2 infrastructure at install or import time, so novel external hostnames resolved by python or pip processes are high-fidelity IOCs that survive even after package removal

Per-Action IR Details

Step 1: Containment — Immediately audit your Python dependency manifest (requirements.txt, pyproject.toml, Pipfile.lock) and CI/CD pipeline configurations for any version of LiteLLM or the Telnyx Python SDK installed during the TeamPCP active window. Isolate any build environments, developer workstations, or CI/CD runners that installed these packages. Consult the Telnyx first-party security notice (https://telnyx.com/resources/telnyx-python-sdk-supply-chain-security-notice-march-2026) and Datadog Security Labs advisory for confirmed malicious version ranges before continuing operations. Per NIST AC-4 (Information Flow Enforcement), restrict outbound network access from affected build environments until investigation is complete.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST AC-4 (Information Flow Enforcement)

Compensating: Run `pip show litellm telnyx` on each developer workstation and CI/CD runner to confirm installed versions; compare against the Telnyx and Datadog Security Labs advisory version ranges. Block outbound traffic from build hosts using host-based firewall rules: `sudo iptables -I OUTPUT -j DROP` (Linux) or Windows Defender Firewall with Advanced Security → Outbound Rules → Block All. For CI/CD runners (GitHub Actions, GitLab CI), disable or pause affected runner pools from the platform UI immediately — no EDR required.

Evidence: BEFORE isolating any build host or CI/CD runner, capture: (1) active outbound network connections via `ss -tunap` or `netstat -ano` to identify any live C2 or exfiltration channels opened by the malicious package at install or import time; (2) the running process tree via `ps auxf` (Linux) or `Get-Process` / `Get-CimInstance Win32_Process` (Windows) to identify child processes spawned by pip or the Python interpreter during package installation; (3) a full copy of pip's install log (`~/.local/pip/log/` or `%APPDATA%\pip\pip.log`) and the current dependency lock files (Pipfile.lock, poetry.lock, requirements.txt) before any modification. These artifacts document which malicious package version executed and what network activity it generated.

Step 2: Detection — Audit package installation logs and CI/CD pipeline run histories for installs of LiteLLM, Telnyx Python SDK, Trivy, and KICS during the campaign window. Check for anomalous outbound network connections from build systems or developer machines to unexpected external hosts following package installation events (per NIST AU-6, Audit Record Review, Analysis, and Reporting). Review cloud provider access logs (AWS CloudTrail, GCP Audit Logs, Azure Activity Logs) for credential use from unusual source IPs, geolocations, or times. Review SSH authentication logs for private key use from unrecognized hosts. Scan Kubernetes audit logs for secret access events not initiated by known service accounts. Apply D3-LAM (Local Account Monitoring) and D3-SFA (System File Analysis) to identify unauthorized credential file reads on affected developer systems.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation)

Compensating: Use `grep -r 'litellm\|telnyx\|trivy\|kics' ~/.pip/pip.log /var/log/ /home/*/.local/pip/` to locate installation events. For cloud credential abuse, query AWS CloudTrail with: `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=AssumeRole --start-time ` and filter for source IPs outside known corporate ranges. For SSH key abuse, audit `/var/log/auth.log` or `journalctl -u sshd` for authentication events using keys stored in developer `~/.ssh/` directories on affected hosts. For Kubernetes, run `kubectl get events --all-namespaces` and `kubectl logs -n kube-system` filtered for secret access. Use osquery: `SELECT * FROM process_events WHERE cmdline LIKE '%litellm%' OR cmdline LIKE '%telnyx%'` to surface post-install execution.

Evidence: This step is analytical and does not alter live state; however, preserve evidence before any follow-on containment or eradication action. Capture: (1) AWS CloudTrail JSON logs for the campaign window filtered on `GetSecretValue`, `AssumeRole`, `CreateAccessKey`, and `ListBuckets` — TeamPCP targets cloud credentials so unusual API calls from build-host source IPs are primary indicators; (2) Kubernetes audit log entries for `secrets/get` and `secrets/list` by non-standard service accounts — the malware specifically targets kubeconfig and in-cluster service account tokens; (3) SSH `authorized_keys` and `known_hosts` files from affected developer workstations

alongside `/var/log/auth.log` entries timestamped after the malicious package install; (4) DNS query logs from build hosts for the campaign window — malicious PyPI packages in this campaign beacon to attacker-controlled infrastructure, so novel external hostnames resolved by Python processes are strong IOCs.

Step 3: Eradication — Remove and replace any version of LiteLLM, Telnyx Python SDK, Trivy, or KICS installed during the active campaign window. Do not reinstall from PyPI until you have verified the clean version against the vendor-confirmed hash or advisory. Rotate all credentials that were present in any environment where a compromised package was installed: cloud provider API tokens, SSH private keys, Kubernetes secrets, and any stored corporate access credentials. Apply D3-CRO (Credential Rotation) immediately across affected scopes. Per NIST CM controls and CIS 2.3 (Address Unauthorized Software), document and remediate all confirmed unauthorized package versions. Per CIS 7.4 (Perform Automated Application Patch Management), pin dependency versions to vendor-verified hashes going forward.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 2.3 (Address Unauthorized Software), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Uninstall compromised packages with `pip uninstall litellm telnyx trivy kics -y` and verify removal with `pip list | grep -E 'litellm|telnyx'`. Before reinstalling, verify the clean package hash: `pip download litellm== --no-deps -d /tmp/verify && pip hash /tmp/verify/*.whl` and compare against the vendor advisory SHA-256. For credential rotation without a PAM tool: AWS — `aws iam delete-access-key` + `aws iam create-access-key`; Kubernetes — `kubectl delete secret -n ` and regenerate via your provisioning process; SSH — generate new keypair with `ssh-keygen -t ed25519`, remove the compromised public key from all `~/.ssh/authorized_keys` and `~/.ssh/known_hosts`, and revoke old keys from any jump hosts or bastion servers.

Evidence: BEFORE rotating credentials or removing packages, capture: (1) a memory dump of any Python processes currently running that imported litellm or telnyx — use `procdump` (Windows Sysinternals) or `gcore ` (Linux) to preserve in-memory secrets and decoded payload strings that will be lost on process termination; (2) a snapshot of all environment variables from affected process PIDs via `/proc//environ` (Linux) or Sysinternals Process Explorer → Environment tab — TeamPCP malware harvests env vars containing AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, KUBECONFIG, and similar tokens; (3) a copy of `~/.aws/credentials`, `~/.kube/config`, and `~/.ssh/id_*` (public keys only for documentation) from each affected host before overwriting; (4) the full pip install cache directory (`~/.cache/pip/` or `%LOCALAPPDATA%\pip\Cache`) preserving the malicious wheel file for hash verification and YARA scanning.

Step 4: Recovery — After credential rotation, validate that no residual unauthorized sessions are active across cloud consoles, Kubernetes clusters, SSH targets, and corporate SSO. Re-run CI/CD pipelines from clean, verified dependency states and confirm build outputs against expected hashes. Monitor cloud provider billing and resource provisioning for anomalous activity that may indicate credentials were used prior to rotation. Per NIST AU-12 (Audit Record Generation), ensure logging is active and capturing post-remediation access events. Apply D3-MFA (Multi-factor Authentication) enforcement review — verify MFA is active on all cloud provider accounts and privileged access paths exposed through compromised environments (CIS 6.3, 6.4, 6.5).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-12 (Audit Record Generation), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.4 (Require MFA for Remote Network Access), CIS 6.5 (Require MFA for Administrative Access)

Compensating: Enumerate all active AWS sessions using `aws iam list-access-keys --user-name ` and `aws sts get-caller-identity` from known-clean hosts; terminate any sessions issued under rotated keys with `aws iam update-access-key --status Inactive`. For Kubernetes, run `kubectl auth can-i --list --as=` to validate least-privilege posture post-rotation and `kubectl get secrets --all-namespaces` to confirm no orphaned secrets remain. Validate CI/CD pipeline integrity by re-running a build from a clean ephemeral runner with pinned dependency hashes and comparing build artifact checksums. For cloud billing anomaly detection without a paid service, configure AWS Budget

Alerts via the console with a 20% threshold above baseline — TeamPCP credential use often triggers EC2 instance launches or S3 data transfers.

Evidence: Recovery does not typically destroy new forensic evidence, but confirm logging continuity before declaring recovery complete: verify AWS CloudTrail is actively logging to an S3 bucket that was not accessible from the compromised build environment (i.e., the exfiltration path could not have modified trail settings); confirm Kubernetes audit log retention is active post-rotation; and validate that SSH authentication logs on reinstated hosts are shipping to a centralized log store, not just local disk — a host-local log on a previously compromised system cannot be fully trusted for post-incident monitoring.

Step 5: Post-Incident — Conduct a lessons-learned review focused on dependency integrity verification gaps. Implement hash-pinned dependency locking for all CI/CD pipelines and require integrity verification before package installation (CIS 2.1, Establish and Maintain a Software Inventory; CIS 7.1, Establish and Maintain a Vulnerability Management Process). This campaign exposed a specific control gap: security scanning tools themselves are not verified before use, enabling a chained attack. Implement signed artifact verification for all security tooling (D3-FMBV, File Magic Byte Verification; NIST SI controls). Subscribe to Endor Labs, Datadog Security Labs, and Snyk advisory feeds for ongoing TeamPCP campaign updates, as Endor Labs reporting indicates the campaign is active beyond currently confirmed packages.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Enforce hash-pinned dependencies at zero cost by switching to `pip install --require-hashes -r requirements.txt` — generate the initial hashed requirements file with `pip-compile --generate-hashes` (pip-tools). For Trivy and KICS binary verification, add a pre-run step in CI/CD scripts: `sha256sum -c trivy__checksums.txt` against the GitHub release checksum file before executing any scan. Create a YARA rule targeting the known-malicious package patterns (exfiltration of `~/.aws`, `~/.kube`, `~/.ssh` paths in Python bytecode) and run it against the pip cache directory on each build host using `yara -r ~/.cache/pip/`. Subscribe to the PyPI RSS feed for the four confirmed packages to receive immediate notification of new releases for manual review.

Evidence: This phase does not alter live host state on production systems; however, preserve the complete incident timeline and all collected artifacts (malicious wheel files, CloudTrail logs, memory dumps, network captures) in tamper-evident cold storage for a minimum of 90 days to support any regulatory breach notification review. The specific control gap documented — that Trivy and KICS were weaponized, meaning security scan results from the campaign window cannot be trusted — must be recorded in the lessons-learned report as a finding that invalidates any clean scan results produced by these tools during the TeamPCP active window.

Detection Guidance

Focus detection on three artifact classes: package installation events, post-installation network behavior, and subsequent credential access patterns. (1) Package audit: Query artifact manager logs, pip install history (~/.local/lib, site-packages), and CI/CD runner logs for installs of litellm, telnyx, trivy, or kics during the campaign window. Cross-reference installed versions against vendor-confirmed clean versions from Telnyx and Datadog Security Labs advisories. (2) Network behavior: Look for outbound DNS queries and TCP connections to unrecognized external hosts initiated by Python processes, build agents, or scanner processes shortly after package installation. Anomalous beaconing or data exfiltration patterns from CI/CD runners are a strong indicator. (3) Credential access: In cloud provider logs, hunt for API token use from source IPs or user-agents inconsistent with your build infrastructure. In Kubernetes, audit secret read events not matching expected service account patterns. In SSH logs, review private key authentication from hosts not in your known asset inventory. (4) File system: On developer workstations and build servers, apply D3-SFA (System File Analysis) to

detect unauthorized reads of credential files including ~/.aws/credentials, ~/.ssh/id_rsa, ~/.kube/config, and environment variable stores. NIST AU-6 (Audit Record Review, Analysis, and Reporting) and AU-2 (Event Logging) should be verified as active across build and developer environments. No public IOC list (hashes, IPs, domains) has been confirmed in the provided source material for this campaign; monitor Datadog Security Labs and Endor Labs advisories for updated IOC releases.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://securitylabs.datadoghq.com/articles/litellm-compromised-pypi-teampcp-supply-chain-campaign/`	Datadog Security Labs advisory confirming LiteLLM and Telnyx PyPI compromise attributed to TeamPCP — check for updated IOC tables in this advisory	HIGH
URL	`https://www.endorlabs.com/learn/teampcp-isnt-done`	Endor Labs reporting on ongoing TeamPCP activity beyond initially disclosed packages — monitor for updated affected package lists and IOC releases	HIGH
URL	`https://telnyx.com/resources/telnyx-python-sdk-supply-chain-security-notice-march-2026`	Telnyx first-party security notice confirming compromise of their Python SDK — authoritative source for affected version ranges	HIGH

Framework Mappings

MITRE-ATTACK

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1528** — Steal Application Access Token
- **T1608.002** — Upload Tool
- **T1554** — Compromise Host Software Binary
- **T1552.004** — Private Keys
- **T1552.001** — Credentials In Files

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

NIST-800-53R5

- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1528	Steal Application Access Token	Credential-Access
T1608.002	Upload Tool	Resource-Development
T1554	Compromise Host Software Binary	Persistence
T1552.004	Private Keys	Credential-Access
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
gemini	https://threat-modeling.com/vulnerability-intelligence-report-july-...	T3
LiteLLM and Telnyx compromised on PyPI - Datadog Security Labs	https://securitylabs.datadoghq.com/articles/litellm-compromised-pyp...	T3

Source	URL	Tier
Telnyx Python SDK: Supply Chain Security Notice	https://telnyx.com/resources/telnyx-python-sdk-supply-chain-securit...	T3
TeamPCP Isn't Done: Threat Actor Behind Trivy and KICS ...	https://www.endorlabs.com/learn/teampcp-isnt-done	T3
How a Poisoned Security Scanner Became the Key to Backdooring ...	https://snyk.io/blog/poisoned-security-scanner-backdooring-litellm/	T1

Generated 2026-07-06 06:33 UTC by TJS Security Command Center