INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:33 UTC

TeamPCP PyPI Supply Chain Campaign: Audio Steganography in Telnyx Package Enables Credential Harvesting Across CI/CD Pipelines

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0112
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	telnyx 4.87.1, 4.87.2 (PyPI); litellm (PyPI); Trivy; KICS, Windows, Linux, macOS; CI/CD pipeline environments broadly
Published	2026-03-27
Discovery Source	Rss

Executive Summary

Threat actor TeamPCP has compromised two versions of the telnyx Python package on PyPI (4.87.1, 4.87.2), embedding credential-harvesting malware inside WAV audio files to evade detection, part of a confirmed multi-package supply chain campaign that has also hit litellm, Trivy, and KICS. Any CI/CD pipeline that installed these packages should be treated as fully compromised; all secrets, tokens, and credentials accessible in those environments are at risk of exfiltration. Reporting from Datadog and Aikido links TeamPCP to LAPSUS$ and ransomware group Vect based on TTP overlap and campaign scope; attribution confidence is moderate pending independent corroboration, raising the likelihood of follow-on ransomware or extortion beyond initial credential theft.

Technical Analysis

TeamPCP compromised telnyx versions 4.87.1 and 4.87.2 on PyPI by embedding malicious payloads inside WAV audio files using steganography (T1027.003, Steganography), a technique designed to bypass EDR inspection and network-layer content filtering. The malware targets credentials and secrets accessible within the execution environment (T1552.001, T1552.004, Credentials In Files, Private Keys), performs system reconnaissance (T1082), exfiltrates data over command-and-control channels (T1041), and establishes persistence via registry run keys or startup items (T1547.001). The package masquerades as the legitimate telnyx SDK (T1036.005, Match Legitimate Name or Location) within a supply chain compromise vector (T1195.001). Python is the execution environment (T1059.006). The broader campaign confirms the same actor

compromised litellm, Trivy, and KICS on PyPI, all tools routinely run with elevated privileges in automated pipeline contexts. CWE references: CWE-506 (Embedded Malicious Code), CWE-494 (Download of Code Without Integrity Check), CWE-312 (Cleartext Storage of Sensitive Information), CWE-311 (Missing Encryption of Sensitive Data). No CVE has been assigned to this campaign; individual compromised packages may have CVE identifiers assigned separately. T1486 (Data Encrypted for Impact) is mapped conditionally, consistent with LAPSUS$/Vect attribution suggesting ransomware as a potential follow-on payload; no encryption-for-impact has been confirmed in the telnyx malware payload itself. Sources: Datadog Security Labs, Aikido, BleepingComputer, LiteLLM official security advisory; all URLs are RSS-originated and unverified via active resolution; human validation recommended before operationalizing.

Action Checklist

1. Step 1: Containment. Immediately audit all Python environments, CI/CD pipelines, and container images for installed versions of telnyx==4.87.1 or telnyx==4.87.2. Run: pip show telnyx or scan requirements.txt, Pipfile.lock, and poetry.lock files across all repositories. Isolate any system where either version is confirmed installed. Extend the audit to litellm, Trivy, and KICS packages; confirm installed versions against known-good release hashes from official PyPI. Revoke network egress for affected pipeline runners pending full investigation.
2. Step 2: Detection. Query package manager logs, CI/CD run logs, and container build logs for installation events referencing telnyx 4.87.1 or 4.87.2 within the affected window. Search for WAV file creation or access events in directories associated with Python package installation paths (%APPDATA%, /tmp, site-packages subdirectories). Monitor outbound network connections from pipeline runners to unknown or newly observed IPs/domains, particularly from Python processes. Review audit logs for unusual credential access patterns: cloud provider credential file reads, SSH key access, environment variable enumeration (T1082, T1552.001, T1552.004). Check for new persistence artifacts: Windows registry Run keys, Linux cron additions, or systemd unit modifications created around the time of package installation (T1547.001).
3. Step 3: Eradication. Remove telnyx 4.87.1 and 4.87.2 from all environments. Upgrade to the latest verified clean release per PyPI, confirming the package hash against the official PyPI package page (https://pypi.org/project/telnyx/) or official GitHub release manifest before installation. Do not reinstall without hash verification. Remove and rebuild any container images that incorporated the compromised versions. Update dependency lockfiles and repin to verified clean versions. Apply the same process to litellm, Trivy, and KICS if installed versions are unverified. Block telnyx 4.87.1 and 4.87.2 in your internal artifact proxy or package allowlist to prevent reintroduction.
4. Step 4: Recovery. Rotate all secrets, API tokens, SSH keys, cloud credentials, and service account credentials that were accessible in any compromised pipeline environment; treat the full secrets scope of each affected runner as exposed. Verify rotation is complete before returning pipelines to production operation. After cleanup, re-run a dependency audit across all pipelines. Monitor pipeline environments and downstream services for 30 days for anomalous authentication, lateral movement, or data access patterns consistent with credential reuse (T1078). Validate that WAV-file-based artifacts and any unrecognized files introduced during the compromise window have been removed from build artifacts and registries.
5. Step 5: Post-Incident. This campaign exposed three control gaps: (1) absence of package integrity verification at install time - implement hash pinning and signature verification in all pipelines, enforced via a private artifact proxy; (2) overly broad secret exposure in CI/CD runners - apply least-privilege secret scoping so individual pipeline jobs access only the credentials they require; (3) no behavioral monitoring

on pipeline runners for file system and network anomalies - deploy runtime monitoring on build infrastructure equivalent to what is applied to production systems. Review your PyPI package allowlist policy and evaluate whether automatic dependency updates without hash verification should be disabled.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal/privacy counsel immediately if forensic analysis confirms exfiltration of secrets that provide access to production systems, customer data stores, or regulated data (PII/PHI/PCI) — this triggers breach notification obligations under GDPR Article 33 (72-hour window), CCPA, and HIPAA Breach Notification Rule, and exceeds the response capability of a standard IR team given the multi-package blast radius of the TeamPCP campaign.
Recovery Notes	Do not return any pipeline to production operation until all credentials accessible during the compromise window are confirmed rotated and revoked — treat this as a full pipeline credential compromise, not a partial exposure, given TeamPCP's confirmed credential-harvesting payload. Monitor all cloud provider audit logs (AWS CloudTrail, GCP Audit Logs, Azure Activity Log) and authentication systems for 30 days post-rotation, specifically filtering for authentication events using the now-rotated key IDs or any IP addresses identified in the TeamPCP C2 infrastructure from threat intelligence reporting. Verify that no WAV files or unrecognized binary blobs were promoted into production artifact registries or deployed container images during the compromise window before restoring automated deployment pipelines.
Forensic Artifacts	WAV files within Python site-packages telnyx package subdirectories (e.g., /usr/lib/python3.x/site-packages/telnyx/**/*.wav or %LOCALAPPDATA%\Python\PythonXX\site-packages\telnyx***.wav) — these are the steganographic payload carriers specific to the TeamPCP audio steganography technique and are the highest-value forensic artifacts for malware analysis and IOC extraction | CI/CD build logs showing `pip install telnyx==4.87.1` or `pip install telnyx==4.87.2` with exact timestamps — from GitHub Actions workflow run logs (_diag/ directory on self-hosted runners), Jenkins build console output, or GitLab CI job trace artifacts — establish the infection window and identify all affected pipeline runs | Outbound network connection records from Python processes on pipeline runners to non-RFC1918 IP addresses during the telnyx installation and post-install execution window — specifically Sysmon Event ID 3 (NetworkConnect, Image: python.exe/python3) on Windows or auditd SYSCALL records for connect() calls from python3 PIDs on Linux — identify TeamPCP C2 exfiltration destinations | Cloud provider credential access logs: AWS CloudTrail `GetSecretValue`, `AssumeRole`, and S3 `GetObject` events; GCP Audit Log `storage.objects.get` and `iam.serviceAccounts.actAs`; Azure Activity Log `Microsoft.KeyVault/vaults/secrets/read` — correlated against timestamps of telnyx 4.87.x execution to identify what credentials the malware enumerated and potentially exfiltrated via T1552.001 | Linux auditd records or Windows Sysmon Event ID 13 (RegistryValueSet) / Event ID 11 (FileCreate) for persistence artifacts created at telnyx package import time — specifically cron entries in /etc/cron.d/ or /var/spool/cron/, systemd unit files in /etc/systemd/system/, and Windows HKCU\Software\Microsoft\Windows\CurrentVersion\Run keys — timestamp correlation to pip install events identifies TeamPCP persistence mechanisms deployed via T1547.001

Per-Action IR Details

Step 1: Containment — Immediately audit all Python environments, CI/CD pipelines, and container images for installed versions of telnyx==4.87.1 or telnyx==4.87.2. Run: pip show telnyx or scan requirements.txt, Pipfile.lock, and poetry.lock files across all repositories. Isolate any system where either version is confirmed installed. Extend the audit to litellm, Trivy, and KICS packages; confirm installed versions against known-good release hashes from official PyPI. Revoke network egress for affected pipeline runners pending full investigation.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality), NIST SC-7 (Boundary Protection), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Run the following across all pipeline hosts and container build nodes: `pip show telnyx 2>/dev/null | grep -E 'Version: 4.87.[12]'` and `grep -r 'telnyx==4.87.[12]' /path/to/repos --include='*.txt' --include='*.lock' --include='*.toml'`. For containers, run `docker inspect | jq '.[].Config.Env'` to surface environment variables, then `docker run --rm pip show telnyx` to check installed version. Block egress with `iptables -I OUTPUT -m owner --uid-owner -j DROP` on Linux runners or Windows Firewall outbound rules scoped to the runner service account. Use osquery with `SELECT name, version FROM python_packages WHERE name='telnyx';` if osquery is deployed on runners.

Evidence: Before isolating, capture a full pip freeze output (`pip freeze > freeze_$(hostname)_$(date +%s).txt`) from every affected environment to establish the exact package state at time of discovery. Collect and preserve the telnyx package directory from site-packages (e.g., `/usr/lib/python3.x/site-packages/telnyx/` or `%LOCALAPPDATA%\Python\PythonXX\site-packages\telnyx\`) including all WAV files embedded within it — these are the primary steganographic payload carriers and must be preserved as forensic evidence before any eradication. Image affected pipeline runner disks or capture container layer snapshots before removal.

Step 2: Detection — Query package manager logs, CI/CD run logs, and container build logs for installation events referencing telnyx 4.87.1 or 4.87.2 within the affected window. Search for WAV file creation or access events in directories associated with Python package installation paths (%APPDATA%, /tmp, site-packages subdirectories). Monitor outbound network connections from pipeline runners to unknown or newly observed IPs/domains, particularly from Python processes. Review audit logs for unusual credential access patterns: cloud provider credential file reads, SSH key access, environment variable enumeration (T1082, T1552.001, T1552.004). Check for new persistence artifacts: Windows registry Run keys, Linux cron additions, or systemd unit modifications created around the time of package installation (T1547.001).

NIST Phase: Detection Analysis

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

Controls: NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: On Linux runners: `ausearch -k access -f ~/.aws/credentials` and `ausearch -k access -f ~/.ssh/id_rsa` to detect credential file reads (requires auditd rules pre-placed; if absent, check `/var/log/auth.log` and `/var/log/syslog` for python3 process activity). For WAV file creation: `find /tmp /var/tmp $(python3 -c 'import site; print(site.getsitepackages()[0])') -name '*.wav' -newer /tmp/telnyx_install_marker -ls 2>/dev/null`. On Windows: query Sysmon Event ID 11 (FileCreate) filtering on `.wav` extensions in `%APPDATA%` and site-packages paths, and Sysmon Event ID 3 (NetworkConnect) where Image contains `python.exe`. For CI/CD log scraping: `grep -E 'telnyx.*(4\.87\.1|4\.87\.2)' ~/.jenkins/jobs/*/builds/*/log` (Jenkins) or parse GitHub Actions runner logs in `_diag/` for pip install events. Check cron with `crontab -l` and `ls -la /etc/cron.*` for entries added within the installation window. MITRE ATT&CK: T1082 (System Information Discovery), T1552.001 (Credentials In Files), T1552.004 (Private Keys), T1547.001 (Registry Run Keys / Startup Folder).

Evidence: Capture outbound netflow or connection logs from pipeline runners for the full period since the earliest possible telnyx 4.87.1/4.87.2 install date — specifically python3/python.exe process connections to non-RFC1918

addresses (TeamPCP C2 infrastructure). Preserve CI/CD build logs (GitHub Actions workflow run logs, Jenkins build console output, GitLab CI job traces) showing the pip install invocation with exact timestamps. On Linux, capture `ss -tunap` and `/proc/net/tcp` snapshots. Collect `/proc//environ` for any python3 processes running at time of discovery to capture environment variable state (cloud tokens, API keys) that the malware would have enumerated via T1082/T1552.001.

Step 3: Eradication — Remove telnyx 4.87.1 and 4.87.2 from all environments. Upgrade to the latest verified clean release per PyPI, confirming the package hash against the official PyPI release page before installation. Do not reinstall without hash verification. Remove and rebuild any container images that incorporated the compromised versions. Update dependency lockfiles and repin to verified clean versions. Apply the same process to litellm, Trivy, and KICS if installed versions are unverified. Block telnyx 4.87.1 and 4.87.2 in your internal artifact proxy or package allowlist to prevent reintroduction.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-7 (Least Functionality), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Verify clean package hash before reinstallation: `pip download telnyx== -d /tmp/telnyx_verify && pip hash /tmp/telnyx_verify/telnyx-*.whl` then compare against the SHA-256 listed on `https://pypi.org/project/telnyx/#files`. Remove malicious versions: `pip uninstall telnyx -y && find $(python3 -c 'import site; print(site.getsitepackages()[0])') -name '*.wav' -delete` — note that WAV payload files may persist in site-packages subdirectories after pip uninstall and must be manually purged. For container images: `docker images --filter 'since=' -q | xargs docker rmi -f` to remove all images built after the first known telnyx 4.87.x install. If using Nexus or Artifactory as a proxy, add telnyx 4.87.1 and 4.87.2 to the blocklist by SHA-256 hash. Write a YARA rule targeting WAV file headers within Python package directories to scan for residual steganographic artifacts.

Evidence: Before removing the compromised telnyx package, extract and hash all WAV files found in the package directory: `find $(python3 -c 'import site; print(site.getsitepackages()[0])')/telnyx* -name '*.wav' -exec sha256sum {} \;`. Preserve the full telnyx egg-info or dist-info directory for version provenance. Capture the INSTALLER and RECORD files from the telnyx dist-info directory (`pip show -f telnyx`) to document exactly which files were deployed by the malicious package. These WAV files and their SHA-256 hashes constitute primary forensic evidence of the steganographic payload and should be submitted to threat intelligence sharing platforms (e.g., VirusTotal, MISP) for community benefit.

Step 4: Recovery — Rotate all secrets, API tokens, SSH keys, cloud credentials, and service account credentials that were accessible in any compromised pipeline environment — treat the full secrets scope of each affected runner as exposed. Verify rotation is complete before returning pipelines to production operation. After cleanup, re-run a dependency audit across all pipelines. Monitor pipeline environments and downstream services for 30 days for anomalous authentication, lateral movement, or data access patterns consistent with credential reuse (T1078). Validate that WAV-file-based artifacts and any unrecognized files introduced during the compromise window have been removed from build artifacts and registries.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), NIST IA-5 (Authenticator Management), NIST CP-10 (System Recovery and Reconstitution), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 5.2 (Use Unique Passwords), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Enumerate every secret accessible to affected runners by reviewing: CI/CD platform secret stores (GitHub Actions encrypted secrets, GitLab CI/CD variables, Jenkins credentials store), mounted Kubernetes secrets, `~/.aws/credentials`, `~/.ssh/`, and environment variables injected at pipeline runtime — document each before rotation to ensure completeness. For AWS: `aws iam list-access-keys --user-name ` then rotate via `aws iam

create-access-key` and `aws iam delete-access-key`. For SSH keys: `ssh-keygen -R ` and remove public keys from `~/.ssh/authorized_keys` on target systems. After rotation, scan artifact registries (Docker Hub, ECR, GCR, GHCR) for any images built during the compromise window using `docker manifest inspect` and cross-reference build timestamps. Monitor for T1078 (Valid Accounts) abuse using AWS CloudTrail `LookupEvents` filtering on the rotated key IDs for 30 days post-rotation.

Evidence: Before rotating credentials, document the full scope of what was accessible: capture a snapshot of CI/CD secret variable names (not values) from the platform's admin UI, list all IAM roles assumed by runner instances (`aws sts get-caller-identity`), and record all Kubernetes service account tokens mounted in runner pods (`kubectl get serviceaccount -n -o yaml`). This establishes the blast radius for breach notification assessment. After rotation, preserve the rotation timestamps and old key IDs — these are needed to scope CloudTrail, GCP Audit Log, or Azure Activity Log queries when hunting for credential reuse by TeamPCP during the post-recovery monitoring window.

Step 5: Post-Incident — This campaign exposed three control gaps: (1) absence of package integrity verification at install time — implement hash pinning and signature verification in all pipelines, enforced via a private artifact proxy; (2) overly broad secret exposure in CI/CD runners — apply least-privilege secret scoping so individual pipeline jobs access only the credentials they require; (3) no behavioral monitoring on pipeline runners for file system and network anomalies — deploy runtime monitoring on build infrastructure equivalent to what is applied to production systems. Review your PyPI package allowlist policy and evaluate whether automatic dependency updates without hash verification should be disabled.

NIST Phase: Post Incident

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

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-7 (Least Functionality), NIST AC-6 (Least Privilege), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 6.3 (Require MFA for Externally-Exposed Applications)

Compensating: Hash pinning: enforce `--require-hashes` in all pip invocations within pipeline definitions (`pip install --require-hashes -r requirements.txt`) and generate hashes with `pip hash .whl`. For a private proxy without budget: deploy Devpi (`pip install devpi-server`) or use Sonatype Nexus OSS (free tier) to mirror and allowlist approved packages by SHA-256. Runtime monitoring on pipeline runners: deploy Sysmon on Windows runners using the SwiftOnSecurity Sysmon config (github.com/SwiftOnSecurity/sysmon-config) and auditd on Linux runners with rules targeting python3 reads of credential file paths (`-w /root/.aws -p r -k aws_cred_access`). Write a Sigma rule to detect WAV file creation in Python site-packages paths — this is a highly specific TeamPCP behavioral indicator. For secret scoping: restructure pipeline YAML to pass only the required secret per job stage using environment injection (`env:` block scoped to a single step in GitHub Actions) rather than org-wide secrets.

Evidence: Conduct a lessons-learned session documenting: (a) the earliest confirmed telnyx 4.87.x install timestamp across all affected systems, (b) the full list of credentials in scope for rotation, (c) which detection controls would have caught the WAV-file-based steganographic payload had they been in place (e.g., Sysmon FileCreate on .wav in site-packages), and (d) whether the TeamPCP campaign was disclosed via any threat intelligence feed the organization subscribes to prior to the incident. Preserve this record per NIST AU-11 (Audit Record Retention) to support future audit evidence and to measure control improvement effectiveness at the next review cycle.

Detection Guidance

Primary detection targets: (1) Package presence - scan all environments for telnyx==4.87.1 or telnyx==4.87.2 using pip list, pip show, or lockfile inspection. (2) WAV file artifacts - search Python package installation directories (site-packages and subdirectories) for unexpected WAV files; flag any WAV file created or accessed by a Python process outside application code paths. (3) Network - alert on outbound connections initiated by Python interpreter processes to IPs or domains not in an established allowlist, particularly from CI/CD runner

hosts; look for low-volume, periodic beaconing patterns consistent with C2 (T1041). (4) Credential access - correlate file access events against known credential file paths: ~/.aws/credentials, ~/.ssh/, environment variable reads in pipeline contexts, cloud metadata API calls from runner hosts. (5) Persistence - on Windows, monitor HKCU\Software\Microsoft\Windows\CurrentVersion\Run and equivalent HKLM keys for new entries created around package install time; on Linux, inspect /etc/cron.d/, user crontabs, and ~/.bashrc or ~/.profile for additions. Behavioral indicators: Python processes reading binary audio files, Python processes making DNS lookups or HTTP requests to domains first seen within the last 30 days, runner processes accessing credential stores outside normal job execution flow. Specific IOCs (C2 infrastructure, malware hashes, exfiltration domains) are reported in active investigations by Datadog Security Labs and Aikido; their articles and threat intelligence feeds should be queried directly for current IOC feeds, as C2 infrastructure is rapidly rotated.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://pypi.org/project/telnyx/4.87.1/`	Compromised telnyx package version 4.87.1 on PyPI — do not install; remove if present	HIGH
URL	`https://pypi.org/project/telnyx/4.87.2/`	Compromised telnyx package version 4.87.2 on PyPI — do not install; remove if present	HIGH
URL	`https://securitylabs.datadoghq.com/articles/litellm-compromised-pypi-teampcp-supply-chain-campaign/`	Datadog Security Labs primary technical analysis — RSS-originated URL, unverified via active resolution; human validation recommended	MEDIUM
URL	`https://aikido.dev/blog/telnyx-pypi-compromised-teampcp-canisterworm`	Aikido telnyx-specific campaign analysis — RSS-originated URL, unverified via active resolution; human validation recommended	MEDIUM
URL	`https://docs.litellm.ai/blog/security-update-march-2026`	LiteLLM official security advisory — RSS-originated URL, unverified via active resolution; human validation recommended	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1041** — Exfiltration Over C2 Channel
- **T1140** — Deobfuscate/Decode Files or Information
- **T1082** — System Information Discovery
- **T1552.004** — Private Keys
- **T1005** — Data from Local System
- **T1027.003** — Steganography

- **T1552.001** — Credentials In Files
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1547.001** — Registry Run Keys / Startup Folder
- **T1078** — Valid Accounts
- **T1059.006** — Python
- **T1486** — Data Encrypted for Impact

NIST-800-53R5

- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-4** — System Monitoring
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **IR-4** — Incident Handling
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

CIS-V8

- **2.5**
- **2.6**
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers
- **8.2** — Collect Audit Logs

NIST-CSF-2

- **RS.MI-01** — Incidents are contained
- **GV.SC-01** — Cybersecurity supply chain risk management program
- **DE.CM-01** — Networks and network services are monitored

HIPAA-SECURITY

- **164.308(a)(7)(ii)(A)** — Data Backup Plan
- **164.312(d)** — Person or Entity Authentication

ISO-27001-2022

- **A.5.29** — Information security during disruption
- **A.5.34** — Privacy and protection of personal information
- **A.5.21** — Managing information security in the ICT supply chain

SOC2-TSC

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1041	Exfiltration Over C2 Channel	Exfiltration
T1140	Deobfuscate/Decode Files or Information	Defense-Evasion
T1082	System Information Discovery	Discovery
T1552.004	Private Keys	Credential-Access
T1005	Data from Local System	Collection
T1027.003	Steganography	Defense-Evasion
T1552.001	Credentials In Files	Credential-Access
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1547.001	Registry Run Keys / Startup Folder	Persistence
T1078	Valid Accounts	Defense-Evasion
T1059.006	Python	Execution
T1486	Data Encrypted for Impact	Impact

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/03/teampcp-pushes-malicious-telnyx.html	T3
Popular telnyx package compromised on PyPI by TeamPCP	https://www.aikido.dev/blog/telnyx-pypi-compromised-teampcp-caniste...	T3

Source	URL	Tier
LiteLLM compromised on PyPI: Tracing the March 2026 TeamPCP ...	https://securitylabs.datadoghq.com/articles/litellm-compromised-pyp...	T3
Popular LiteLLM PyPI package backdoored to steal credentials, auth ...	https://www.bleepingcomputer.com/news/security/popular-litellm-pypi...	T3
Security Update: Suspected Supply Chain Incident - LiteLLM Docs	https://docs.litellm.ai/blog/security-update-march-2026	T3

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-03-29 18:33 UTC by TJS Security Command Center