INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-20 13:49 UTC

TanStack npm Supply Chain Worm Enables TeamPCP to Breach Grafana, OpenAI, and GitHub via Unrotated Workflow Tokens

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0343
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Grafana Labs, TanStack (npm ecosystem), OpenAI, Mistral AI, GitHub, specific package versions not confirmed in available source data
Published	2026-05-20T01:12:06
Discovery Source	Rss

Executive Summary

Threat actor TeamPCP compromised the TanStack npm package and 160+ additional packages with a self-propagating worm that stole GitHub Actions workflow tokens from CI/CD pipelines. Grafana Labs confirmed on May 19, 2026 that an unrotated workflow token left over from initial remediation allowed attackers to access private source code and internal operational data; victims also include OpenAI, Mistral AI, and GitHub itself. The business risk is significant: stolen source code, exposure of internal infrastructure secrets, and demonstrated attacker willingness to escalate to extortion create compounded financial, operational, and reputational exposure.

Technical Analysis

TeamPCP injected malicious code into the TanStack npm package and reportedly 160+ additional npm and PyPI packages. The payload functions as a worm targeting GitHub Actions workflow environments: it exfiltrates secrets stored in CI/CD pipeline contexts (GITHUB_TOKEN, repository secrets) via CWE-522 (insufficiently protected credentials) and CWE-312 (cleartext storage of sensitive information), then uses those tokens for lateral movement into victim repository infrastructure (CWE-829, inclusion of functionality from untrusted control sphere). No CVE has been assigned. Applicable MITRE ATT&CK techniques include T1195.001 (Supply Chain Compromise: Compromise Software Dependencies), T1552.004 (Private Keys), T1552.001 (Credentials in Files), T1059 (Command and Scripting Interpreter), T1537 (Transfer Data to Cloud Account), T1213 (Data from Information Repositories), T1486 (Data Encrypted for Impact), and T1657 (Financial Theft via Extortion).

Grafana confirmed on May 19, 2026 that a single unrotated token overlooked during initial triage extended attacker access to private repos and operational data. The attack escalated to ransomware extortion on May 16, 2026; Grafana declined to pay. No specific patched package versions have been confirmed in available source data as of this writing. Sources include T1 primary (Grafana official blog) and T3 news/security vendor coverage (TheHackerNews, Orca Security, SafeDep, Reddit). Grafana blog post is the highest-confidence primary source for incident timeline and token remediation failure details.

Action Checklist

1. Containment, Immediately audit all GitHub Actions workflows for use of TanStack packages or any of the 160+ reported compromised npm packages. Revoke and rotate all GitHub Actions GITHUB_TOKEN values, repository secrets, and any workflow-scoped credentials that may have been exposed in affected pipeline runs. Disable workflows referencing suspect packages until clean versions are confirmed. Reference: NIST SI-3 (Malicious Code Protection), CIS 2.3 (Address Unauthorized Software).

2. Detection, Query CI/CD pipeline logs for unexpected outbound network connections or secret access during workflow runs involving TanStack or related packages. Search GitHub Actions audit logs for anomalous token usage, cross-repository access, or secret reads not matching expected pipeline behavior. Look for processes spawned by npm install or postinstall hooks making external HTTP requests. Apply NIST SI-4 (System Monitoring) and AU-6 (Audit Record Review, Analysis, and Reporting). Use D3-SFA (System File Analysis) to inspect workflow YAML files and package-lock.json for unexpected dependency changes.

3. Eradication, Pin all npm dependencies to verified, uncompromised versions using lockfiles and integrity hashes (npm shrinkwrap or package-lock.json with integrity fields). Remove any TanStack or flagged package versions from build environments and clear npm cache. Audit postinstall scripts across all dependencies and block unapproved scripts via npm config set ignore-scripts true where feasible. Reference: NIST SI-2 (Flaw Remediation), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), D3-FMBV (File Magic Byte Verification for package integrity).

4. Recovery, After rotating all workflow tokens and secrets, re-run affected pipelines in an isolated environment and verify expected behavior before restoring production builds. Confirm no unauthorized repository access, branch modifications, or exfiltrated artifacts remain accessible. Validate audit log continuity per NIST AU-11 (Audit Record Retention) and AU-9 (Protection of Audit Information). Monitor for follow-on extortion contact or new unauthorized access attempts. Apply D3-CRO (Credential Rotation) across all affected service accounts and pipeline identities.

5. Post-Incident, Implement automated secret scanning in CI/CD pipelines (e.g., GitHub Advanced Security secret scanning, truffleHog) to catch unrotated or exposed credentials before attackers do. Establish a formal process for complete credential rotation as part of every incident response cycle, addressing the specific gap Grafana identified: a single overlooked token extended the breach. Formalize dependency review gates using software composition analysis (SCA) tooling. Map gaps to NIST IR-4 (Incident Handling), IR-8 (Incident Response Plan), SI-7 (Software, Firmware, and Information Integrity), CIS 8.2 (Collect Audit Logs), and D3-UAP (User Account Permissions) for least-privilege enforcement on workflow tokens.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to executive leadership, legal counsel, and applicable regulatory bodies immediately if forensic analysis confirms that stolen workflow tokens were used to access repositories containing customer PII, PHI, payment card data, or proprietary source code subject to trade secret protections, or if any of the 160+ compromised packages were published to public npm registries used by downstream customers — both conditions trigger breach notification obligations under GDPR, CCPA, and sector-specific regulations.
Recovery Notes	After completing full credential rotation, monitor GitHub audit logs daily for a minimum of 30 days for any resumed access attempts using previously compromised token patterns or from IP ranges associated with TeamPCP infrastructure identified during the investigation. Verify that all downstream consumers of any internally published npm packages built during the compromised window have been notified and have re-pinned to verified clean versions, since the worm's self-propagating mechanism means any package published using a stolen token may itself be a propagation vector. Do not restore production CI/CD pipelines until SCA tooling and truffleHog secret scanning are confirmed operational as required pipeline gates, directly addressing the credential-rotation gap that extended the Grafana Labs breach.
Forensic Artifacts	GitHub Actions audit log JSON export (via Audit Log API) for `action:secrets.access`, `action:git.clone`, `action:repo.download_zip`, and `action:git.push` events — these entries will show exactly which repositories TeamPCP accessed using stolen GITHUB_TOKEN values and what data was exfiltrated during the breach window. | Malicious postinstall script files within `node_modules/@tanstack/` and the 160+ flagged package directories on self-hosted runners and build nodes — the worm's payload is embedded here and will contain the outbound exfiltration logic and token-theft mechanism specific to the TeamPCP campaign. | `package-lock.json` integrity field discrepancies — comparison of SHA-512 `integrity` values in committed lockfiles against npm registry checksums (`npm view @ dist.integrity`) will identify which specific installed package versions were trojanized and confirm propagation scope across internal projects. | npm debug logs at `~/.npm/_logs/*.log` on affected build nodes — these capture postinstall script execution output, including any error messages or network call attempts made by the worm during `npm install`, providing a timestamp-anchored execution trail tied to specific workflow run IDs. | GitHub repository git history diff (`git log --all --format='%H %ae %s' --since=`) for all repositories whose workflows had access to compromised tokens — unauthorized commits, branch modifications, or new deploy keys added by TeamPCP using stolen GITHUB_TOKEN credentials will appear as anomalous author email addresses or bot-attributed commits outside normal pipeline service account patterns.

Per-Action IR Details

Containment — Immediately audit all GitHub Actions workflows for use of TanStack packages or any of the 160+ reported compromised npm packages. Revoke and rotate all GitHub Actions GITHUB_TOKEN values, repository secrets, and any workflow-scoped credentials that may have been exposed in affected pipeline runs. Disable workflows referencing suspect packages until clean versions are confirmed. Reference: NIST SI-3 (Malicious Code Protection), CIS 2.3 (Address Unauthorized Software).

NIST Phase: Containment

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

Controls: NIST SI-3 (Malicious Code Protection), NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), CIS 2.3 (Address Unauthorized Software), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Run: `grep -r 'tanstack\|@tanstack' .github/workflows/ --include='*.yml' --include='*.yaml' -l` across all repositories to enumerate affected workflow files. For GITHUB_TOKEN revocation without enterprise tooling, use the GitHub CLI: `gh secret list --repo ` for each affected repo, then immediately disable workflows via `gh workflow disable `. For self-hosted runners, kill and re-image runner VMs to ensure the worm's postinstall payload has not persisted on the runner filesystem.

Evidence: Before revoking any tokens, capture the full GitHub Actions audit log for the 30 days preceding May 19, 2026 via the GitHub Audit Log API (`GET /orgs/{org}/audit-log?phrase=action:workflows`), exporting to JSON. Preserve the raw runner working directory at `/home/runner/work//` and `/tmp/` on any self-hosted runner, as TeamPCP's postinstall hook would have written staging or exfiltration artifacts there. Snapshot the `package-lock.json` integrity hash values for all affected workflow runs — the worm modifies these during `npm install` to propagate, so hash drift against the committed lockfile is direct forensic evidence of infection.

Detection — Query CI/CD pipeline logs for unexpected outbound network connections or secret access during workflow runs involving TanStack or related packages. Search GitHub Actions audit logs for anomalous token usage, cross-repository access, or secret reads not matching expected pipeline behavior. Look for processes spawned by npm install or postinstall hooks making external HTTP requests. Apply NIST SI-4 (System Monitoring) and AU-6 (Audit Record Review, Analysis, and Reporting). Use D3-SFA (System File Analysis) to inspect workflow YAML files and package-lock.json for unexpected dependency changes.

NIST Phase: Detection Analysis

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

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-2 (Event Logging), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: On GitHub-hosted runners, use the Actions debug log (`ACTIONS_STEP_DEBUG=true`) and parse workflow run logs via the GitHub API: `gh run list --limit 100 --json databaseId,status,workflowName | jq` then `gh run view --log` for each flagged run. For outbound HTTP detection from postinstall hooks on self-hosted Linux runners, query `auditd` logs: `ausearch -sc connect -ts ` filtering on PIDs spawned by `npm` or `node`. On Windows runners, query Sysmon Event ID 3 (Network Connection) filtering on parent process `node.exe` or `npm.cmd`. For secret access anomalies, use the GitHub Audit Log API filtering on `action:secrets.access` and correlate timestamps against workflow run IDs that pulled TanStack packages.

Evidence: Capture GitHub Actions audit log entries for `action:secrets.access`, `action:repo.download_zip`, and `action:git.push` events tied to workflow runs that executed `npm install` with TanStack in scope — TeamPCP's worm exfiltrates `GITHUB_TOKEN` via outbound HTTP from within the postinstall script, so network telemetry showing `node` or `npm` processes initiating connections to non-registry, non-CDN IPs during install phases is a primary indicator. Preserve `~/.npm/_logs/` on self-hosted runners for npm debug output showing postinstall script execution. Collect workflow YAML git diff history to detect if the worm self-modified workflow files to ensure re-infection on subsequent runs.

Eradication — Pin all npm dependencies to verified, uncompromised versions using lockfiles and integrity hashes (npm shrinkwrap or package-lock.json with integrity fields). Remove any TanStack or flagged package versions from build environments and clear npm cache. Audit postinstall scripts across all dependencies and block unapproved scripts via npm config set ignore-scripts true where feasible. Reference: NIST SI-2 (Flaw Remediation), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), D3-FMBV (File Magic Byte Verification for package integrity).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-2 (Baseline Configuration), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Run `npm cache clean --force` on all build nodes and self-hosted runners, then delete `node_modules/` directories in all affected repositories before reinstalling from a verified lockfile. Use `npm audit

--audit-level=critical` against a known-clean registry mirror, or configure `.npmrc` to point to a private registry (Verdaccio, self-hosted) stocked only with verified package versions. To enumerate all postinstall scripts across the dependency tree: `cat package-lock.json | jq -r '.packages | to_entries[] | select(.value.scripts.postinstall != null) | .key'` — manually review each output entry against expected behavior. Apply YARA rules targeting the worm's known outbound HTTP pattern in `node_modules//` JS files before reinstall.

Evidence: Before clearing the npm cache, forensically copy `~/.npm/` and the full `node_modules/` tree from each affected build environment — the malicious postinstall script embedded by TeamPCP in TanStack and the 160+ propagated packages will be recoverable here and constitutes primary malware evidence. Preserve the `package-lock.json` integrity SHA-512 field values and compare against npm registry checksums (`npm view @ dist.integrity`) to confirm which specific installed versions were trojanized. Document the npm shrinkwrap or lockfile state at time of compromise to establish the scope of propagation across internal projects.

Recovery — After rotating all workflow tokens and secrets, re-run affected pipelines in an isolated environment and verify expected behavior before restoring production builds. Confirm no unauthorized repository access, branch modifications, or exfiltrated artifacts remain accessible. Validate audit log continuity per NIST AU-11 (Audit Record Retention) and AU-9 (Protection of Audit Information). Monitor for follow-on extortion contact or new unauthorized access attempts. Apply D3-CRO (Credential Rotation) across all affected service accounts and pipeline identities.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-11 (Audit Record Retention), NIST AU-9 (Protection of Audit Information), NIST IR-4 (Incident Handling), NIST CP-10 (System Recovery and Reconstitution), CIS 6.2 (Establish an Access Revoking Process), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Stand up an isolated GitHub Actions runner (ephemeral Docker container) using `act` (https://github.com/nektos/act) to replay affected workflows against clean package versions before re-enabling production pipelines. Use `git log --all --oneline` and `git diff ..HEAD` on all repositories that had compromised workflow tokens to detect any unauthorized branch pushes or tag modifications made by TeamPCP using the stolen `GITHUB_TOKEN`. For extortion monitoring, set up a free RSS or email alert on your organization name via Google Alerts as a minimum indicator of data leak site publication — this directly mirrors the Grafana Labs breach timeline where stolen source code exposure was the business risk.

Evidence: Before re-enabling production pipelines, collect a complete export of GitHub repository access logs via the Audit Log API filtering on `action:git.clone`, `action:git.fetch`, and `action:repo.create` events tied to the compromised token identities — this establishes what source code and internal data TeamPCP accessed using the unrotated workflow tokens, replicating the specific gap Grafana identified on May 19, 2026. Preserve deploy key and OAuth token access logs from any third-party integrations (e.g., npm publish tokens, cloud provider OIDC tokens) that were accessible from affected workflow environments, as the worm's token theft scope may extend beyond `GITHUB_TOKEN` to any secret mounted in the pipeline.

Post-Incident — Implement automated secret scanning in CI/CD pipelines (e.g., GitHub Advanced Security secret scanning, truffleHog) to catch unrotated or exposed credentials before attackers do. Establish a formal process for complete credential rotation as part of every incident response cycle, addressing the specific gap Grafana identified — a single overlooked token extended the breach. Formalize dependency review gates using software composition analysis (SCA) tooling. Map gaps to NIST IR-4 (Incident Handling), IR-8 (Incident Response Plan), SI-7 (Software, Firmware, and Information Integrity), CIS 8.2 (Collect Audit Logs), and D3-UAP (User Account Permissions) for least-privilege enforcement on workflow tokens.

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-7 (Software, Firmware, and Information Integrity), NIST SI-5 (Security Alerts, Advisories, and Directives), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 8.2 (Collect Audit

Logs)

Compensating: Deploy truffleHog v3 (free, open source) as a pre-commit hook and GitHub Actions step: `trufflehog git file://. --since-commit HEAD~1 --only-verified` — this directly addresses Grafana's overlooked-token gap by scanning for live, verified credentials before each pipeline run. For SCA without enterprise tooling, integrate `npm audit` and `osv-scanner` (Google's free OSS Vulnerability Scanner) as required CI steps that fail the build on any package matching the TeamPCP-compromised package list. Enforce least-privilege workflow tokens by adding `permissions: read-all` as a top-level default in all workflow YAML files and explicitly granting only required scopes per job — this limits future `GITHUB_TOKEN` blast radius to read-only operations in the event of another supply chain compromise.

Evidence: Produce a post-incident artifact inventory documenting: (1) all GitHub repositories whose Actions workflows imported TanStack or the 160+ flagged packages, (2) the full list of secrets and tokens confirmed rotated with rotation timestamps for audit trail continuity per NIST AU-11 (Audit Record Retention), and (3) a git history diff report showing any unauthorized commits, branch creations, or tag modifications made during the period the stolen tokens were active — this constitutes the breach scope documentation required for any regulatory notification obligations and directly supports the lessons-learned process mandated by NIST 800-61r3 §4.

Detection Guidance

Primary detection focus: GitHub Actions audit logs and CI/CD pipeline runtime behavior. Specific indicators to hunt: (1) Outbound HTTP/HTTPS connections initiated from npm postinstall or lifecycle script execution, flag any process spawned by node/npm making external requests to non-registry endpoints during package installation. (2) GitHub Actions audit log entries showing GITHUB_TOKEN or repository secrets accessed by jobs that installed TanStack or flagged packages, cross-reference job run timestamps against known compromise window (pre-May 19, 2026). (3) Unexpected cross-repository access or secret reads from workflow identities, particularly reads on repositories not normally touched by the affected workflow. (4) New or modified workflow YAML files with altered dependency pins or added steps not matching change history. (5) Package-lock.json or yarn.lock modifications introducing unexpected resolved URLs or integrity hash changes for TanStack packages. Behavioral indicator: worm behavior means compromise may have propagated, treat any repository that ran a compromised workflow as potentially exposed, not just the initial entry point. Apply NIST AU-6 (Audit Record Review) and AU-12 (Audit Record Generation) to ensure pipeline logs are retained and queryable. NIST SI-4 (System Monitoring) should cover CI/CD environments explicitly. D3-LAM (Local Account Monitoring) applies to service account and token usage patterns within GitHub.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`Not confirmed in available source data`	Exfiltration endpoint(s) used by worm payload — specific domains not disclosed in T3 sources reviewed; monitor Orca Security and Grafana blog for updates	LOW

Framework Mappings

MITRE-ATTACK

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1059** — Command and Scripting Interpreter
- **T1486** — Data Encrypted for Impact
- **T1537** — Transfer Data to Cloud Account
- **T1213** — Data from Information Repositories
- **T1657** — Financial Theft
- **T1552.004** — Private Keys
- **T1552.001** — Credentials In Files

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **IA-5** — Authenticator Management
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1059	Command and Scripting Interpreter	Execution
T1486	Data Encrypted for Impact	Impact
T1537	Transfer Data to Cloud Account	Exfiltration
T1213	Data from Information Repositories	Collection
T1657	Financial Theft	Impact
T1552.004	Private Keys	Credential-Access
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/05/grafana-github-breach-exposes-sou...	T3
Mass Supply Chain Attack Hits TanStack, Mistral AI npm and PyPI ...	https://safedep.io/mass-npm-supply-chain-attack-tanstack-mistral	T3
Latest on TanStack npm supply chain ransomware incident	https://grafana.com/blog/grafana-labs-security-update-latest-on-tan...	T3
Mass npm Supply Chain Attack Hits TanStack, Mistral AI, and 170+ ...	https://www.reddit.com/r/programming/comments/1tapmvi/mass_npm_supp...	T3
TanStack & 160+ npm Packages Compromised - Orca Security	https://orca.security/resources/blog/tanstack-npm-supply-chain-worm/	T3

Generated 2026-05-20 13:49 UTC by TJS Security Command Center