INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-20 18:55 UTC

Grafana IR Token Rotation Failure Enables Post-Compromise Repository Access After TanStack Supply Chain Attack

DATA BREACH | **HIGH** | CVSS 7.5

SCC Item ID	SCC-DBR-2026-0136
Type	Data Breach
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Grafana (private GitHub repositories, source code, business contact data); TanStack npm packages (initial vector); GitHub Actions workflows
Published	2026-05-20T11:46:37
Discovery Source	Rss

Executive Summary

Grafana Labs suffered a secondary breach after its incident response team failed to rotate a GitHub workflow token exposed during the TanStack npm supply chain attack. According to Grafana's official statement, the unrotated credential gave threat actor TeamPCP continued access to Grafana's private GitHub repositories, resulting in source code theft and exfiltration of business contact and operational data. While Grafana confirmed no customer production systems were compromised, the incident demonstrates how procedural gaps in incident response execution can extend an organization's exposure window well beyond the initial attack.

Technical Analysis

Following the TanStack/Shai-Hulud supply chain compromise (MITRE T1195.002), TeamPCP's malicious npm package executed within Grafana's GitHub Actions CI/CD workflows and exposed a GitHub workflow token (T1552.001, CWE-255, CWE-522). During remediation, Grafana's IR team failed to rotate this credential, representing a gap in systematic token inventory and rotation procedures. The unrotated token was subsequently used for valid account abuse (T1078, T1078.004) to re-access private GitHub repositories, enabling source code exfiltration (T1213) and data theft from cloud storage (T1530). No CVE has been assigned; the root cause is procedural rather than a novel vulnerability. CWE-255 (Credentials Management Errors) and CWE-522 (Insufficiently Protected Credentials) are the relevant weakness classifications. The attack chain: compromised upstream npm package → GitHub Actions workflow execution → token harvested → IR

rotation gap exploited → persistent repository access achieved. Grafana has not published specific affected workflow names or token types beyond GitHub Actions context. The timeline and duration of post-remediation token reuse has not been disclosed.

Action Checklist

1. Step 1: Containment, Immediately audit all GitHub Actions workflow tokens, OAuth tokens, personal access tokens (PATs), and repository secrets in your CI/CD pipelines. Revoke any token that was active during a period of suspected compromise or that cannot be verified as unexposed. See GitHub's token audit UI under Settings > Security > Personal access tokens. Apply NIST SP 800-61 Revision 3 (Computer Security Incident Handling Guide), Section 4, requiring explicit credential scope documentation during containment.
2. Step 2: Detection, Query GitHub audit logs (available via API: /orgs/{org}/audit-log) for unexpected repository clone, download, or API access events, especially outside business hours or from unfamiliar IP ranges. Cross-reference with Actions workflow run logs for any external npm install steps that executed packages not in your approved inventory. Map against MITRE T1213 (Data from Information Repositories) and T1552.001 (Credentials In Files) behavioral indicators. Reference CIS 8.2 (Collect Audit Logs) and confirm audit logging is enabled for GitHub org-level events.
3. Step 3: Eradication, Rotate all CI/CD pipeline credentials systematically using a documented token inventory. Apply NIST SP 800-53 IA-5 (Authenticator Management) procedures: enumerate every token, secret, deploy key, and OAuth credential associated with affected workflows; rotate each one with proof of rotation recorded. Enforce short-lived tokens where supported - GitHub Actions supports OIDC-based short-lived credentials as a structural control. Review all npm dependencies in affected workflows against the TanStack/Shai-Hulud package list and remove or pin versions as appropriate per NIST SI-2 (Flaw Remediation).
4. Step 4: Recovery, After rotation, validate repository access logs show no continued activity from pre-rotation tokens. Re-run CI/CD pipelines in a monitored state and confirm expected behavior. Track GitHub service account activity post-remediation for unauthorized access attempts. Audit repository contents for unauthorized commits, forks, or data staging artifacts. Document what was accessed and when, consistent with NIST AU-11 (Audit Record Retention) requirements for post-incident review.
5. Step 5: Post-Incident, Update your incident response playbook to include an explicit 'Credential Inventory and Rotation' step triggered by any supply chain or CI/CD-related incident, referencing NIST SP 800-61. Implement a mandatory token rotation checklist as a non-optional IR gate - no incident may be closed without signed-off completion. Apply CIS 5.1 (Establish and Maintain an Inventory of Accounts) to CI/CD service accounts and tokens, not only user accounts. Conduct a tabletop exercise simulating a supply chain trigger to validate the updated playbook. Consider adopting GitHub Actions OIDC for keyless authentication to eliminate long-lived token exposure structurally.

IR / Forensic Enrichment

Triage Priority	URGENT

Escalation Criteria	Escalate immediately to legal and executive leadership if GitHub audit logs confirm TeamPCP accessed repositories containing customer data, PII, regulated data (GDPR/CCPA), or cryptographic signing keys — any of these conditions triggers breach notification obligations or supply chain integrity concerns that extend beyond internal IR scope.
Recovery Notes	After token rotation is complete, maintain elevated monitoring of GitHub org audit logs for a minimum of 30 days, specifically alerting on any `git.clone`, `repo.download_zip`, or `org.add_member` events from IP ranges not associated with Grafana office or CI/CD runner infrastructure — TeamPCP may retain access via a persistence mechanism (forked repo, previously cloned local copy, or a second credential not yet identified) that survives token rotation. Verify repository integrity by diffing all affected private repository commit histories against a pre-compromise baseline to confirm no backdoors or malicious code were introduced to Grafana source during the access window. Conduct a dependency audit of any Grafana products that consume the affected private repositories as build inputs to assess whether compromised source code propagated into any release artifacts distributed to customers.
Forensic Artifacts	GitHub Org Audit Log (JSON export via /orgs/{org}/audit-log API): primary forensic record of all git.clone, git.fetch, repo.download_zip, org.oauth_application_token_revoke, and workflow_run events attributable to the exposed token during the TeamPCP access window — this log is the chain-of-custody anchor for the entire investigation. | GitHub Actions Workflow Run Logs for npm install steps: step-level logs showing which TanStack/Shai-Hulud package versions were fetched from the npm registry, what postinstall scripts executed, and which GitHub Actions secrets were referenced via ${{ secrets.* }} syntax during the malicious run — these establish how the token was exfiltrated from the workflow environment. | npm package-lock.json and npm-shrinkwrap.json at the time of compromise: version-pinned dependency tree showing the exact compromised @tanstack package version installed, its resolved tarball URL from the npm registry, and its integrity hash — compare against the known-good Shai-Hulud package list to confirm which malicious version executed in the Grafana workflow. | GitHub Repository Fork and Clone Records (gh api /repos/{owner}/{repo}/forks and git.clone audit events): enumerates all forks created and external clones performed on Grafana private repositories during the TeamPCP access window — unauthorized forks persist independently of token revocation and represent continued exfiltration risk even after containment. | GitHub Actions Runner Host Network Logs (if self-hosted runners used): outbound DNS queries and HTTP connections made by the runner process during the compromised workflow execution, specifically connections to non-npm external endpoints that may indicate the malicious TanStack package phoning home or the token being transmitted to TeamPCP-controlled infrastructure.

Per-Action IR Details

Step 1: Containment — Immediately audit all GitHub Actions workflow tokens, OAuth tokens, personal access tokens (PATs), and repository secrets in your CI/CD pipelines. Revoke any token that was active during a period of suspected compromise or that cannot be verified as unexposed. See GitHub's token audit UI under Settings > Security > Personal access tokens. Apply NIST IR-4 (Incident Handling) procedures requiring explicit credential scope during containment.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), NIST IA-5 (Authenticator Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: For a 2-person team without an identity governance platform: run `gh api /orgs/{org}/installations` and `gh api /users/{user}/installations` via the GitHub CLI to enumerate all OAuth app authorizations. Export all Actions

secrets per repo with `gh secret list --repo {owner}/{repo}` and cross-reference against your known token inventory. For PATs, use the GitHub API endpoint `GET /orgs/{org}/members` combined with manual review at github.com/settings/tokens — there is no bulk export, so build a spreadsheet during triage. Immediately revoke any PAT or OAuth token whose last-used timestamp falls within the TanStack compromise window (March 2025 disclosure date) before moving to eradication.

Evidence: Before revoking any credential, capture: (1) GitHub org audit log export covering the TanStack compromise window — use `gh api '/orgs/{org}/audit-log?phrase=action:org.oauth_application_token_revoke&per_page=100'` to record the pre-revocation state; (2) screenshot or JSON export of all active PATs from github.com/settings/tokens showing token name, scope, and last-used date for every token active during the exposure window; (3) GitHub Actions workflow run logs for any workflow that performed `npm install` or `npm ci` steps referencing TanStack or @tanstack scoped packages — download via `gh run view {run-id} --log` before logs expire (GitHub retains logs 90 days by default); (4) list of all repository secrets and environment secrets via `gh secret list` across all affected repos — record names and last-updated timestamps before rotation destroys the pre-incident state.

Step 2: Detection — Query GitHub audit logs (available via API: /orgs/{org}/audit-log) for unexpected repository clone, download, or API access events — especially outside business hours or from unfamiliar IP ranges. Cross-reference with Actions workflow run logs for any external npm install steps that executed packages not in your approved inventory. Map against MITRE T1213 (Data from Information Repositories) and T1552.001 (Credentials In Files) behavioral indicators. Reference CIS 8.2 (Collect Audit Logs) — confirm audit logging is enabled for GitHub org-level events.

NIST Phase: Detection Analysis

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

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-3 (Content of Audit Records), NIST IR-4 (Incident Handling), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without a SIEM, use the GitHub CLI to query audit logs directly: `gh api '/orgs/{org}/audit-log?phrase=action:git.clone&per_page=100' | jq '.[] | select(.actor != "expected-bot-name") | {actor, repo, created_at, ip}'` — pipe output to a CSV for timeline analysis in a spreadsheet. For npm supply chain detection specifically, grep workflow YAML files for unversioned or unverified @tanstack package references: `grep -r '@tanstack' .github/workflows/ --include='*.yml'`. Use `npm audit --json` against your lockfiles to identify the specific compromised TanStack package versions documented in the Shai-Hulud disclosure. For IP geolocation of suspicious access, run actor IPs through ipinfo.io CLI or AbuseIPDB free tier to identify TeamPCP-associated infrastructure.

Evidence: Before pivoting to eradication, preserve: (1) full GitHub org audit log in JSON format covering 30 days prior to and following the TanStack npm supply chain disclosure, focusing on `git.clone`, `git.fetch`, `repo.download_zip`, and `org.add_member` action types; (2) network-level evidence — if GitHub Enterprise or a self-hosted runner is in use, capture runner host network logs showing outbound connections to npm registry endpoints and any non-npm external hosts contacted during the compromised workflow run; (3) the specific TanStack/Shai-Hulud package versions installed during affected workflow runs, extractable from `package-lock.json` or `npm-shrinkwrap.json` committed to the repository at the time of the malicious run; (4) GitHub Actions workflow run artifacts and step summaries for any run that executed between the TanStack npm compromise window and the token revocation date, showing which secrets were referenced via `${{ secrets.* }}` syntax in the run context.

Step 3: Eradication — Rotate all CI/CD pipeline credentials systematically using a documented token inventory (CWE-1326 remediation). Apply NIST IA-5 (Authenticator Management) procedures: enumerate every token, secret, deploy key, and OAuth credential associated with affected workflows; rotate each one with proof of rotation recorded. Apply D3-CRO (Credential Rotation) countermeasure. Enforce short-lived tokens where supported — GitHub Actions supports OIDC-based short-lived credentials as a structural control. Review all npm dependencies in affected workflows against the TanStack/Shai-Hulud package list and remove or pin versions as appropriate per NIST SI-2 (Flaw Remediation).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IA-5 (Authenticator Management), NIST SI-2 (Flaw Remediation), NIST CM-2 (Baseline Configuration), NIST AC-6 (Least Privilege), CIS 5.2 (Use Unique Passwords), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For teams without a secrets management platform (Vault, AWS Secrets Manager): create a rotation tracking spreadsheet with columns: token name, repository, scope, rotation date, rotated-by, new token last-four, and sign-off. For each GitHub Actions workflow, replace long-lived PAT-based auth with OIDC by adding the `permissions: id-token: write` block and using `actions/configure-aws-credentials` or `google-github-actions/auth` patterns — this eliminates the credential class that enabled TeamPCP's persistence. For npm dependency pinning without a commercial SCA tool, run `npm ci --ignore-scripts` in affected workflow steps to prevent malicious postinstall execution, and use `npm audit --audit-level=high` as a gate. Pin all @tanstack dependencies to verified-clean versions using exact version specifiers (remove `^` and `~` prefixes) in package.json.

Evidence: Before rotating each credential, record: (1) the token's full scope list (read:org, repo, workflow, etc.) and creation date — over-privileged scopes may indicate the token was purpose-built for exfiltration or was misconfigured at creation; (2) for each GitHub deploy key, capture the key fingerprint via `gh api /repos/{owner}/{repo}/keys` before deletion — this establishes forensic proof of what access existed; (3) a diff of `package-lock.json` between the commit immediately before the TanStack supply chain event and the current HEAD, to identify exactly which dependency versions were swapped in by the malicious package and what transitive dependencies they introduced; (4) GitHub Actions OIDC token claims from any workflow run during the compromise window if available in runner debug logs — these establish what cloud resources the compromised workflow may have authenticated to beyond GitHub itself.

Step 4: Recovery — After rotation, validate repository access logs show no continued activity from pre-rotation tokens. Re-run CI/CD pipelines in a monitored state and confirm expected behavior. Apply D3-LAM (Local Account Monitoring) to track GitHub service account activity post-remediation. Audit repository contents for unauthorized commits, forks, or data staging artifacts. Document what was accessed and when, consistent with NIST AU-11 (Audit Record Retention) requirements for post-incident review.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-11 (Audit Record Retention), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AC-2 (Account Management), NIST SI-2 (Flaw Remediation), CIS 8.2 (Collect Audit Logs), CIS 5.3 (Disable Dormant Accounts)

Compensating: For post-rotation monitoring without EDR on runners: enable GitHub org audit log streaming to an S3 bucket or Azure Blob via the GitHub audit log streaming API (available on GitHub Enterprise or free for public org events via webhook) — this gives you persistent, tamper-resistant log storage outside GitHub's 90-day retention. Write a daily cron job using the GitHub CLI to query `gh api '/orgs/{org}/audit-log?phrase=action:git.clone&per_page=100'` and diff against a known-clean baseline, alerting on any actor that is not in your approved service account list. For repository integrity verification, use `git log --all --oneline --since='2025-03-01'` on a clean clone of each affected private repository and compare commit hashes against your pre-incident state to detect unauthorized commits TeamPCP may have made during access.

Evidence: After rotation, capture as recovery verification artifacts: (1) GitHub audit log query results showing zero activity for each revoked token ID after its rotation timestamp — this is your proof of successful containment; (2) git commit history audit for all Grafana private repositories accessible during the TeamPCP access window, using `git log --format='%H %an %ae %ai %s' --all` to surface any commits authored by non-Grafana identities or committed outside normal business hours; (3) repository fork audit via `gh api /repos/{owner}/{repo}/forks` for all affected repos — unauthorized forks are a primary exfiltration method for source code theft and are not removed by token rotation; (4) GitHub Actions workflow run results for the first post-rotation pipeline execution, confirming OIDC token issuance succeeded and no legacy PAT references remain in the workflow logs.

Step 5: Post-Incident — Update your IR playbook to include an explicit 'Credential Inventory and Rotation' step triggered by any supply chain or CI/CD-related incident, referencing NIST IR-8 (Incident Response Plan) and mapping to NIST IA-5. Implement a mandatory token rotation checklist as a non-optional IR gate — no incident may be closed without signed-off completion. Apply CIS 5.1 (Establish and Maintain an Inventory of

Accounts) to CI/CD service accounts and tokens, not only user accounts. Conduct a tabletop exercise simulating a supply chain trigger to validate the updated playbook. Consider adopting GitHub Actions OIDC for keyless authentication to eliminate long-lived token exposure structurally (D3-CH — Credential Hardening).

NIST Phase: Post Incident

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

Controls: NIST IR-8 (Incident Response Plan), NIST IA-5 (Authenticator Management), NIST AU-11 (Audit Record Retention), NIST AC-2 (Account Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For teams without a GRC platform to manage playbook versioning: maintain the IR playbook in a protected branch of a dedicated private GitHub repository with required reviewers — this gives you version control, change history, and approval workflow at no cost. For the tabletop exercise, use the Grafana/TanStack scenario directly: inject a fictional malicious npm package into a test workflow, task the team with identifying the exposed token, and time the rotation process end-to-end. Document the gap between theoretical rotation time and actual — Grafana's failure was a procedural gap, not a technical one. For continuous token inventory without a CMDB, use a GitHub Actions scheduled workflow (cron) that runs `gh secret list` and `gh api /orgs/{org}/installations` weekly and commits the output diff to a secured audit repository, creating an automated evidence trail for the CIS 5.1 account inventory requirement.

Evidence: For the post-incident report and lessons-learned documentation, preserve: (1) the complete token rotation log showing before/after state for every credential rotated, with timestamps and responsible party — this is the primary evidence that the procedural gap identified in the Grafana incident has been closed; (2) a timeline reconstruction document mapping the TanStack npm package compromise date, the GitHub workflow execution that consumed the malicious package, the window during which the exposed token was valid and unrotated, and the date of confirmed TeamPCP access to Grafana repositories — this timeline is required for any regulatory breach notification assessment; (3) the updated IR playbook diff showing the new mandatory 'Credential Inventory and Rotation' gate, with its trigger conditions explicitly listing supply chain and CI/CD incident types, signed off by the IR lead; (4) results of the post-incident OIDC migration audit confirming which workflows have been converted from PAT-based to keyless authentication, reducing the long-lived token attack surface that made this incident possible.

Detection Guidance

Primary detection surface is GitHub's organization audit log. Query for repository access events (git.clone, repo.download, org.create_actions_secret changes) occurring after your initial IR containment timestamp; any such event using a token that predates rotation is a confirmed indicator of the gap. GitHub audit logs are accessible via REST API at /orgs/{org}/audit-log with event filters; export and ingest into your SIEM. Behavioral indicators aligned to MITRE T1213 and T1078.004: service account or automation token accessing repositories outside scheduled pipeline windows; bulk repository clones or archive downloads; Actions workflow runs triggered by external or forked repositories. For the supply chain vector (T1195.002), inspect package-lock.json and workflow YAML files for TanStack packages active during the compromise window (March-April 2025 per disclosed timelines); specifically, look for packages that introduced postinstall scripts. Apply system file analysis to workflow definition files for unauthorized modifications. Cross-reference with npm audit logs for the Shai-Hulud package family. As of reviewed sources, no public IOC hashes for Shai-Hulud payloads have been published in CISA or threat intelligence feeds. Treat any unrecognized npm postinstall execution in GitHub Actions as suspicious pending investigation.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`Not confirmed in reviewed sources`	No specific C2 domains, IPs, or file hashes have been publicly confirmed for TeamPCP's tooling in this incident as of reviewed sources	LOW

Framework Mappings

MITRE-ATTACK

- **T1195.002** — Compromise Software Supply Chain
- **T1213** — Data from Information Repositories
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1552.001** — Credentials In Files
- **T1078.004** — Cloud Accounts
- **T1078** — Valid Accounts
- **T1530** — Data from Cloud Storage

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

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

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
T1195.002	Compromise Software Supply Chain	Initial-Access
T1213	Data from Information Repositories	Collection
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1552.001	Credentials In Files	Credential-Access
T1078.004	Cloud Accounts	Defense-Evasion
T1078	Valid Accounts	Defense-Evasion
T1530	Data from Cloud Storage	Collection

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/grafana-breach-cause...	T3
Latest on TanStack npm supply chain ransomware incident	https://grafana.com/blog/grafana-labs-security-update-latest-on-tan...	T3
Grafana's Breach Update — Attackers accessed #GitHub source ...	https://www.facebook.com/thehackernews/posts/-grafanas-breach-updat...	T3
Grafana Labs has confirmed a security breach linked to the recent ...	https://www.instagram.com/p/DYjTrCMFOpE/	T3

Source	URL	Tier
Grafana GitHub Actions Security Incident - StepSecurity	https://www.stepsecurity.io/blog/grafana-github-actions-security-in...	T3

Generated 2026-05-20 18:55 UTC by TJS Security Command Center