INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-07-10 07:19 UTC

Aged Ghost Accounts Fuel Coordinated GitHub Reconnaissance Campaign Targeting Corporate Organizations

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0643
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	GitHub (GitHub API, GitHub Organizations, GitHub OAuth, Personal Access Tokens)
Published	2026-07-09T14:38:49
Discovery Source	Rss

Executive Summary

Threat actors are conducting a systematic reconnaissance campaign against corporate GitHub organizations using dormant accounts aged two to five years, compromised OAuth tokens, and exposed personal access tokens (PATs). According to Datadog Security Labs, at least one attacker escalated from public repository enumeration to cloning a private repository, signaling intent beyond passive intelligence gathering. Organizations with private source code on GitHub face supply chain risk: unauthorized code access could enable downstream compromise of software build pipelines or dependency injection attacks.

Technical Analysis

Threat actors are abusing three credential vectors against corporate GitHub organizations: dormant accounts aged two to five years (providing API traffic that blends with legitimate developer activity), compromised OAuth tokens, and exposed PATs. The use of pre-aged accounts is a deliberate OPSEC measure to evade anomaly-based detection that would flag newly created accounts. Datadog Security Labs attributed at least one private repository clone to this campaign. Relevant CWEs: CWE-798 (hardcoded/exposed credentials covering PAT exposure), CWE-284 (improper access control), CWE-522 (insufficiently protected credentials), CWE-200 (exposure of sensitive information). Mapped MITRE techniques include T1078 (valid accounts), T1078.004 (cloud accounts), T1528 (steal application access token), T1552 (unsecured credentials), T1552.001 (credentials in files), T1591.002 (gather victim org information), T1087/T1087.003 (account discovery), T1530 (data from cloud storage), T1526 (cloud service discovery), T1590/T1592.002 (gather victim network/host information), T1199 (trusted relationship), T1190 (exploit public-facing application). No CVE is assigned.

Attribution is unconfirmed. The primary source is a Tier 2 news report (The Hacker News); Datadog Security Labs is the named research source for the private repository cloning claim but no independently published Datadog advisory was included in the provided data. Confidence is medium, treat as a credible draft lead pending authoritative corroboration.

Action Checklist

1. Step 1: Containment, Audit all active OAuth tokens and PATs scoped to your GitHub organization immediately. Revoke any token that cannot be attributed to a current, active employee or authorized service account. Use GitHub's organization security settings (Settings > Third-party access > Active tokens) to enumerate and revoke. Disable fine-grained and classic PATs for accounts inactive beyond 45 days, consistent with CIS 5.3 (Disable Dormant Accounts).
2. Step 2: Detection, Query GitHub audit logs (available via the GitHub Audit Log API or the organization's audit log UI) for the following signals over the trailing 90 days: API calls originating from accounts with no prior activity in the last 60+ days; repository clone events (git.clone) against private repositories from unfamiliar IP ranges or user agents; rapid sequential enumeration of organization members or repositories (T1591.002, T1087.003); OAuth application authorizations from unrecognized third-party apps (T1528). Cross-reference source IPs against known threat intelligence feeds. Enable NIST AU-6 (Audit Record Review, Analysis, and Reporting) and ensure GitHub audit log streaming to your SIEM is active per NIST AU-12 (Audit Record Generation) and CIS 8.2 (Collect Audit Logs).
3. Step 3: Eradication, Rotate all PATs and OAuth tokens for accounts flagged in Step 2. Enforce fine-grained PATs with minimum required scopes per GitHub's official fine-grained PAT permission documentation (https://docs.github.com/en/rest/authentication/permissions-required-for-fine-grained-personal-access-tokens). Remove or downscope any token with repository:read or repo scope that is not operationally required. Apply NIST AC-6 (Least Privilege) and AC-2 (Account Management): require justification for private repository access grants. Enable GitHub secret scanning with push protection to prevent future PAT exposure in committed code (D3-CRO: Credential Rotation; D3-CH: Credential Hardening).
4. Step 4: Recovery, Validate that all revoked tokens are no longer functional by attempting a test API call with the rotated credential and confirming 401 responses. Review private repository clone history post-remediation for 30 days to confirm no residual unauthorized access. Monitor organization membership for accounts that may have been added during the campaign window. Confirm audit log streaming continuity. Validate MFA enforcement on all organization member accounts per CIS 6.3 (Require MFA for Externally-Exposed Applications) and CIS 6.5 (Require MFA for Administrative Access) using D3-MFA (Multi-factor Authentication).
5. Step 5: Post-Incident, Conduct a formal review of PAT lifecycle governance: how tokens are issued, scoped, tracked, and expired. Establish maximum token lifetimes (GitHub supports expiration settings). Implement CIS 5.1 (Establish and Maintain an Inventory of Accounts) for all GitHub service accounts and bot accounts. Evaluate whether private repository contents accessed during the campaign could appear in downstream build artifacts or dependency manifests, this is the primary supply chain risk vector. Brief software engineering leadership on the risk of PAT exposure in CI/CD pipeline configuration files (T1552.001). Map findings to NIST AC-2 and AC-17 control gaps for inclusion in the next GRC review cycle.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and legal counsel if GitHub audit logs confirm a `git.clone` event against a private repository containing source code, credentials, or customer data — this constitutes potential data exfiltration triggering breach notification assessment under applicable regulations (GDPR, state privacy laws) and supply chain incident notification obligations to downstream customers or partners.
Recovery Notes	Post-containment, monitor all private repository `git.clone` and API access events daily for a minimum of 30 days, specifically watching for source IPs, user agents, or GitHub account handles that overlap with the pre-remediation campaign window identified in audit logs. Validate that GitHub secret scanning with push protection is active across all private repositories to prevent re-exposure of PATs in committed code, and confirm that all organization members have MFA enforced before restoring any previously revoked access. The supply chain risk window — the period between initial ghost account reactivation and token revocation — must be scoped precisely using audit log timestamps and communicated to software engineering and product leadership for downstream artifact review.
Forensic Artifacts	GitHub Audit Log API export (90-day window): JSON records of `git.clone` events against private repositories, including `actor` (GitHub login), `actor_location.country_code`, `user_agent`, `@timestamp`, and `repository` fields — directly maps ghost account activity to specific private repo exfiltration events in this campaign. | GitHub `/orgs/{ORG}/credential-authorizations` API response: timestamped snapshot of all active OAuth tokens and PATs with associated login, granted scopes, creation date, last-used date, and last-used IP — the only source for correlating specific token IDs to the recon API calls observed in audit logs before revocation destroys the association. | GitHub Audit Log events for `org.add_member`, `personal_access_token.create`, and `oauth_access.create` actions: identifies accounts added to the organization or tokens granted during the 2–5 year aged account dormancy window consistent with this campaign's ghost account infrastructure. | CI/CD pipeline configuration files (`.github/workflows/*.yml`, `Jenkinsfile`, `.circleci/config.yml`, `.env` files in repository history): source of PAT or OAuth secret exposure that may have seeded the compromised token inventory used in this campaign, recoverable via `git log -p --'*.yml'` and `trufflehog` history scan. | Private repository `git log --all --oneline` and file tree snapshots for repositories confirmed cloned by threat actors: establishes what source code, secrets, dependency manifests, or build configurations were accessible during the campaign window, required for supply chain blast radius assessment.

Per-Action IR Details

Step 1: Containment — Audit all active OAuth tokens and PATs scoped to your GitHub organization immediately. Revoke any token that cannot be attributed to a current, active employee or authorized service account. Use GitHub's organization security settings (Settings > Third-party access > Active tokens) to enumerate and revoke. Disable fine-grained and classic PATs for accounts inactive beyond 45 days, consistent with CIS 5.3 (Disable Dormant Accounts).

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: prioritize stopping ongoing access before eradication to prevent further private repository exfiltration by aged ghost accounts holding live tokens.

Controls: NIST AC-2 (Account Management), NIST AC-12 (Session Termination), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Export the token list via GitHub API with: `curl -H 'Authorization: Bearer ' 'https://api.github.com/orgs/{ORG}/credential-authorizations'`. Pipe output through `jq` to filter tokens with `authorized_credential_expires_at` null or `last_used_at` older than 45 days. For orgs without API access tier, manually export from Settings > Third-party access and cross-reference against HR roster in a spreadsheet. Two-person task: one audits, one revokes in real time.

Evidence: Before revoking any token, capture the full token credential-authorizations API response (`/orgs/{ORG}/credential-authorizations`) as a timestamped JSON snapshot — this is the only machine-readable record of which token ID was associated with which login, scopes granted, and last-used IP. Also export GitHub audit log entries for `oauth_access.create`, `personal_access_token.create`, and `org.add_member` events for the past 90 days before any revocation action destroys the ability to correlate token IDs to specific recon activity. Once tokens are revoked, the association between token ID and observed API calls in the audit log may become orphaned.

Step 2: Detection — Query GitHub audit logs (available via the GitHub Audit Log API or the organization's audit log UI) for the following signals over the trailing 90 days: API calls originating from accounts with no prior activity in the last 60+ days; repository clone events (git.clone) against private repositories from unfamiliar IP ranges or user agents; rapid sequential enumeration of organization members or repositories (T1591.002, T1087.003); OAuth application authorizations from unrecognized third-party apps (T1528). Cross-reference source IPs against known threat intelligence feeds. Enable NIST AU-6 (Audit Record Review, Analysis, and Reporting) and ensure GitHub audit log streaming to your SIEM is active per NIST AU-12 (Audit Record Generation) and CIS 8.2 (Collect Audit Logs).

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: correlate GitHub audit log signals across dormant account reactivation, private repo clone events, and OAuth grant anomalies to distinguish passive reconnaissance from active exfiltration by this campaign.

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, pull GitHub audit logs via API: `curl -H 'Authorization: Bearer ' 'https://api.github.com/orgs/{ORG}/audit-log?phrase=action:git.clone&per_page=100'`. Filter for `git.clone` events against private repos using `jq '[.[] | select(.repository_visibility=="private")]'`. For dormant account reactivation, query `phrase=action:org.add_member OR action:personal_access_token.access` and cross-reference `actor_location.country_code` and `user_agent` fields. Use a free threat intel source such as AbuseIPDB CLI lookup against source IPs extracted from the log. Sigma rule `github_audit_log_recon` (available in SigmaHQ community rules) can be adapted for offline log analysis with `sigma-cli`.

Evidence: This step is read-only analysis and does not alter live state, so order-of-volatility sequencing is not a primary concern here. However, GitHub audit logs for free/Team orgs are retained for only 90 days — export the full 90-day window immediately via the Audit Log API before any org-level policy changes (such as enabling log streaming) could reset or interrupt retention. Preserve: raw JSON audit log export, source IP list with geolocation, user-agent strings from `git.clone` and API enumeration events, and a timeline mapping `actor` (GitHub login) to event type and timestamp. These artifacts directly support attribution to the aged ghost account TTPs described by Datadog Security Labs.

Step 3: Eradication — Rotate all PATs and OAuth tokens for accounts flagged in Step 2. Enforce fine-grained PATs with minimum required scopes per GitHub's official fine-grained PAT permission documentation (github.com/en/rest/authentication/permissions-required-for-fine-grained-personal-access-tokens). Remove or downscope any token with repository:read or repo scope that is not operationally required. Apply NIST AC-6 (Least Privilege) and AC-2 (Account Management): require justification for private repository access grants. Enable GitHub secret scanning with push protection to prevent future PAT exposure in committed code (D3-CRO: Credential Rotation; D3-CH: Credential Hardening).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: remove threat actor's persistent access mechanisms (compromised OAuth tokens, over-scoped PATs) from the GitHub organization and harden token issuance policy to prevent reestablishment using the same aged-account credential abuse vector.

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege)

Compensating: Use GitHub's secret scanning REST API to identify any PATs or OAuth secrets already committed to repositories: `curl -H 'Authorization: Bearer ' 'https://api.github.com/repos/{ORG}/{REPO}/secret-scanning/alerts'`. For organizations without GitHub Advanced Security, use `trufflehog` (open source) to scan repository history: `trufflehog github --org={ORG} --token= --only-verified`. Enforce fine-grained PAT policy via org settings (Settings > Personal access tokens > Allow only fine-grained personal access tokens) — this is a free org-level control requiring no tooling budget.

Evidence: Before rotating any PAT or OAuth token flagged in Step 2, record the full scope list, associated GitHub login, creation date, last-used date, and last-used IP for each token — this data is only available via the `/orgs/{ORG}/credential-authorizations` endpoint while the token is still active. Once rotated, this metadata is no longer retrievable. Also capture a `git log --all --oneline` snapshot of any private repositories confirmed cloned by threat actors (identified in Step 2 `git.clone` audit events) to establish a baseline of what code was accessible at time of exfiltration — this is critical for downstream supply chain impact assessment.

Step 4: Recovery — Validate that all revoked tokens are no longer functional by attempting a test API call with the rotated credential and confirming 401 responses. Review private repository clone history post-remediation for 30 days to confirm no residual unauthorized access. Monitor organization membership for accounts that may have been added during the campaign window. Confirm audit log streaming continuity. Validate MFA enforcement on all organization member accounts per CIS 6.3 (Require MFA for Externally-Exposed Applications) and CIS 6.5 (Require MFA for Administrative Access) using D3-MFA (Multi-factor Authentication).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: confirm revocation efficacy and restore verified-clean operational posture for GitHub org access, with active monitoring of private repository clone events for 30 days to detect any residual or re-established threat actor access.

Controls: NIST AC-2 (Account Management), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: Validate revocation by scripting a test API call for each previously active token: `curl -H 'Authorization: token ' https://api.github.com/user` — a 401 response confirms revocation. For 30-day clone monitoring without a SIEM, schedule a daily cron job: `curl -H 'Authorization: Bearer ' 'https://api.github.com/orgs/{ORG}/audit-log?phrase=action:git.clone&after=$(date -d yesterday +%Y-%m-%dT%H:%M:%SZ)' | jq '[.[] | select(.repository_visibility=="private")]' >> /var/log/github_clone_monitor.json`. Review output each morning for unfamiliar actors or IPs.

Evidence: This step involves validating revocation (non-destructive) and monitoring; it does not alter live state, so volatile evidence capture is not a prerequisite. However, maintain the 90-day audit log export captured in Step 2 as the forensic baseline for comparison — any `git.clone` events against private repositories appearing post-remediation that share source IPs, user agents, or actor patterns with the pre-remediation campaign window should be treated as indicators of threat actor reentry using a previously unidentified token or newly compromised account. Preserve MFA compliance reports (exported from GitHub org security settings) as evidence of remediation completeness for GRC documentation.

Step 5: Post-Incident — Conduct a formal review of PAT lifecycle governance: how tokens are issued, scoped, tracked, and expired. Establish maximum token lifetimes (GitHub supports expiration settings). Implement CIS 5.1 (Establish and Maintain an Inventory of Accounts) for all GitHub service accounts and bot accounts. Evaluate whether private repository contents accessed during the campaign could appear in downstream build artifacts or dependency manifests — this is the primary supply chain risk vector. Brief software engineering leadership on the risk of PAT exposure in CI/CD pipeline configuration files (T1552.001). Map findings to NIST AC-2 and AC-17 control gaps for inclusion in the next GRC review cycle.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: document lessons learned from the ghost account and token governance failures, update PAT lifecycle controls, and evaluate downstream supply chain exposure from any private repository contents confirmed accessed during the campaign.

Controls: NIST AC-2 (Account Management), NIST AC-17 (Remote Access), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Build a GitHub service account and bot account inventory using the org members API: `curl -H 'Authorization: Bearer ' 'https://api.github.com/orgs/{ORG}/members?role=all&per_page=100'` and cross-reference against HR/contractor roster in a spreadsheet updated monthly. For CI/CD PAT exposure scanning, run `trufflehog filesystem --directory=.` against pipeline configuration directories (`.github/workflows/`, `Jenkinsfile`, `.circleci/`) in all repositories to detect hardcoded PATs. Document token governance gaps in a one-page control gap memo referencing NIST AC-2 and AC-17 for the next quarterly GRC review.

Evidence: No live state is altered in this phase; the forensic record compiled across Steps 1–4 is the primary input. Specifically required for the supply chain impact assessment: the list of private repositories confirmed as cloned (from `git.clone` audit events in Step 2), the commit history and file tree of those repositories at the time of the campaign (establish via `git log` snapshots and repository content exports), and any CI/CD pipeline configuration files that reference PATs or secrets — these must be reviewed to determine whether exposed source code could enable a downstream dependency confusion, secrets extraction, or build pipeline compromise scenario consistent with the supply chain risk described in the Datadog Security Labs reporting.

Detection Guidance

Query GitHub organization audit logs for the following behavioral indicators. All queries should cover at minimum the trailing 90 days given the aged-account OPSEC strategy in use.

1. Private repository clone events: Filter audit log for action:git.clone on repositories with visibility:private. Flag any clone from an account with fewer than 5 prior audit events in the preceding 60 days.

2. Dormant account API activity: Identify organization member accounts with zero audit log entries for 60 or more days that suddenly generate API calls. Cross-reference against HR/identity provider records to confirm active employment.

3. Rapid repository or member enumeration: Look for sequential calls to /orgs/{org}/repos and /orgs/{org}/members within short time windows from a single account or IP, consistent with T1591.002 and T1087.003.

4. OAuth token authorization from unrecognized apps: Review /orgs/{org}/oauth_authorizations for third-party application grants not in your approved application registry (T1528).

5. PAT usage from anomalous IPs or user agents: If audit log streaming is configured, correlate PAT-authenticated API calls against known developer IP ranges and standard tooling user agents (e.g., git/2.x, GitHub CLI). Flag Tor exit nodes, VPS hosting ranges, or headless browser user agents.

CIS 8.2 (Collect Audit Logs) requires audit log streaming to be active. NIST AU-6 (Audit Record Review, Analysis, and Reporting) requires periodic review cadence. D3-LAM (Local Account Monitoring) applies to the dormant account detection use case.

Framework Mappings

MITRE-ATTACK

- **T1078** — Valid Accounts
- **T1552** — Unsecured Credentials
- **T1591.002** — Business Relationships
- **T1087.003** — Email Account
- **T1528** — Steal Application Access Token
- **T1078.004** — Cloud Accounts
- **T1552.001** — Credentials In Files
- **T1590** — Gather Victim Network Information
- **T1530** — Data from Cloud Storage
- **T1526** — Cloud Service Discovery
- **T1190** — Exploit Public-Facing Application
- **T1087** — Account Discovery
- **T1199** — Trusted Relationship
- **T1592.002** — Software

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-3** — Access Enforcement
- **SC-28** — Protection of Information at Rest
- **SR-2** — Supply Chain Risk Management Plan
- **SI-4** — System Monitoring

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures
- **A01:2021** — Broken Access Control
- **A04:2021** — Insecure Design

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process
- **5.2** — Use Unique Passwords

- **15.1** — Establish and Maintain an Inventory of Service Providers
- **8.2** — Collect Audit Logs

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.5.21** — Managing information security in the ICT supply chain

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC9.2** — Manages risks associated with vendors and business partners

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.308(a)(5)(ii)(D)** — Password Management

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1078	Valid Accounts	Defense-Evasion
T1552	Unsecured Credentials	Credential-Access
T1591.002	Business Relationships	Reconnaissance
T1087.003	Email Account	Discovery
T1528	Steal Application Access Token	Credential-Access
T1078.004	Cloud Accounts	Defense-Evasion
T1552.001	Credentials In Files	Credential-Access
T1590	Gather Victim Network Information	Reconnaissance
T1530	Data from Cloud Storage	Collection
T1526	Cloud Service Discovery	Discovery
T1190	Exploit Public-Facing Application	Initial-Access
T1087	Account Discovery	Discovery
T1199	Trusted Relationship	Initial-Access

Technique ID	Technique Name	Tactic
T1592.002	Software	Reconnaissance

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/07/dormant-github-accounts-help-atta...	T2
Permissions required for fine-grained personal access tokens	https://docs.github.com/en/rest/authentication/permissions-required...	T3
Managing your personal access tokens	https://docs.github.com/en/authentication/keeping-your-account-and-...	T3
OAuth Personal Token to access organization's private repo	https://stackoverflow.com/questions/69729227/oauth-personal-token-t...	T3
Secret scanning: on-demand revocation for ...	https://github.com/orgs/community/discussions/139967	T3

Generated 2026-07-10 07:19 UTC by TJS Security Command Center