INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-24 06:22 UTC

Cordyceps Campaign Targets CI/CD Pipelines via Malicious Pull Requests Across Major Open Source Projects

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0551
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Microsoft Azure Sentinel, Google AI Agent Development Kit (ADK), Apache Doris, Cloudflare Workers SDK, Python Black formatter
Published	2026-06-23T15:16:42
Discovery Source	Rss

Executive Summary

A threat campaign called 'Cordyceps' is targeting CI/CD pipelines across five high-profile open source projects, including Microsoft Azure Sentinel, Google AI ADK, Apache Doris, the Cloudflare Workers SDK, and the Python Black formatter, by submitting malicious pull requests that execute code inside automated build and test workflows. If successful, attackers can poison software artifacts distributed to thousands of downstream organizations that depend on these projects. The business risk is supply chain contamination: software your teams build or deploy using these projects could unknowingly carry adversary-inserted code. Attribution and campaign scope are based on third-party reporting and have not been independently confirmed by affected vendors or CISA.

Technical Analysis

The Cordyceps campaign exploits trust relationships that CI/CD pipelines extend to external contributor pull requests. Targeted repositories span cloud security tooling (Azure Sentinel), AI infrastructure (Google AI ADK), data warehousing (Apache Doris), edge compute SDKs (Cloudflare Workers SDK), and developer formatting tooling (Python Black). Attack vectors include insufficient PR validation controls (CWE-306: Missing Authentication for Critical Function), inclusion of malicious external code via dependency or workflow file manipulation (CWE-829: Inclusion of Functionality from Untrusted Control Sphere), and absence of pipeline protection mechanisms (CWE-693: Protection Mechanism Failure). Mapped MITRE ATT&CK techniques: T1195.001 (Supply Chain Compromise: Compromise Software Dependencies and Development Tools), T1554

(Compromise Host Software Binary), T1059 (Command and Scripting Interpreter), T1072 (Software Deployment Tools), and T1552.001 (Unsecured Credentials: Credentials in Files). No CVE ID has been assigned.

Action Checklist

1. Step 1: Containment. IF your organization uses any of the five affected projects (Azure Sentinel, Google AI ADK, Apache Doris, Cloudflare Workers SDK, Python Black), audit CI/CD pipeline trigger configurations immediately. Disable or restrict external contributor PR workflows from executing in privileged pipeline contexts. Enforce branch protection rules requiring maintainer review before any CI workflow runs on PRs from forks. If you do not use these projects, proceed to Step 2 to verify your supply chain risk posture for similar open source dependencies.
2. Step 2: Detection. Review pipeline execution logs for unexpected script execution, outbound network calls, or artifact modifications during PR-triggered builds. Query CI/CD audit logs (GitHub Actions, GitLab CI, Azure DevOps) for workflow runs initiated by external fork contributors within the past 90 days. Inspect workflow definition files (.github/workflows/*.yml or equivalent) for unauthorized changes. Hunt for T1552.001 indicators: scan build logs and environment variable outputs for credential exposure patterns. Reference NIST 800-53 AU-2 (Event Logging) and AU-6 (Audit Record Review) to ensure pipeline events are captured and reviewed.
3. Step 3: Eradication. Remove or quarantine any pipeline workflow files modified by unrecognized external contributors. Rotate all secrets, tokens, and credentials that were accessible to CI/CD pipeline environments potentially exposed during suspect PR builds (NIST 800-53 IA-4: Identifier Management). Implement required reviewer approvals and 'pull_request_target' scope restrictions in GitHub Actions to prevent fork-sourced PRs from accessing repository secrets. Verify artifact integrity for any builds executed during the exposure window.
4. Step 4: Recovery. Re-run clean builds from verified source commits after eradication. Validate artifact hashes against known-good baselines before redistributing or deploying build outputs. Monitor pipeline environments post-remediation for anomalous execution patterns using SIEM alerting aligned with NIST 800-53 AU-6 (Audit Record Review, Analysis, and Reporting). Confirm that branch protection rules and PR workflow restrictions are enforced across all affected repositories.
5. Step 5: Post-Incident. Conduct a formal review of CI/CD pipeline security posture against NIST 800-53 SI-4 (Information System Monitoring) and CIS 4.6 (Securely Manage Enterprise Assets and Software). Implement mandatory code signing for build artifacts to enable downstream integrity verification (NIST 800-53 SI-7: Software, Firmware, and Information Integrity). Establish a software bill of materials (SBOM) process to track open source dependencies. Document lessons learned and update the software supply chain risk section of your third-party risk management program.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to senior leadership, legal counsel, and potentially affected downstream organizations immediately if forensic analysis confirms any build artifact distributed during the exposure window was modified by Cordyceps-injected pipeline code, as this constitutes a software supply chain compromise with potential breach notification obligations for downstream consumers of Azure Sentinel content packs, Google AI ADK packages, Apache Doris binaries, Cloudflare Workers SDK releases, or Python Black formatter distributions.
Recovery Notes	Recovery must begin only from git commits that predate or post-date the confirmed Cordyceps campaign activity window, with all workflow files reviewed by a trusted maintainer before the first clean build runs. Artifact integrity validation against pre-campaign SHA-256 baselines is mandatory before any package is republished to PyPI, npm, Maven Central, or equivalent registries. Monitor pipeline execution logs and fork contributor activity continuously for a minimum of 30 days post-remediation, as the Cordyceps campaign's pattern of targeting multiple high-profile projects simultaneously suggests the threat actor may attempt re-entry via new fork accounts or variation of the pull_request_target injection technique.
Forensic Artifacts	GitHub Actions workflow run logs (downloadable via `gh run view {run_id} --log`) for all PR-triggered runs over the past 90 days on repos consuming the five affected projects — primary evidence of injected script execution, outbound exfiltration calls, and environment variable dumping by the Cordyceps campaign | Git history of `.github/workflows/*.yml` files (`git log --all --follow -p .github/workflows/`) across affected repositories — captures the exact diff of any workflow step injected by a Cordyceps fork PR, including added `run:` blocks executing curl/wget/base64/env commands | CI runner host network egress logs for the exposure window — outbound connections from the build environment to non-GitHub IP ranges or domains during a PR-triggered workflow run are a direct indicator of credential or artifact exfiltration by the injected pipeline payload | SHA-256 checksums and provenance metadata for all build artifacts (PyPI wheels, npm tarballs, Docker images, GitHub release assets) produced during the exposure window — comparison against pre-campaign release hashes identifies whether downstream-distributed artifacts were modified by poisoned pipeline steps | GitHub organization and repository audit log exports (`GET /orgs/{org}/audit-log?phrase=action:workflows` via GitHub API) — records actor identity, IP address, and exact workflow file path for every workflow creation, modification, and run event, enabling attribution of Cordyceps campaign activity to specific external fork contributor accounts

Per-Action IR Details

Step 1: Containment — Audit CI/CD pipeline trigger configurations across all repositories using the five affected projects. Disable or restrict external contributor PR workflows from executing in privileged pipeline contexts. Enforce branch protection rules requiring maintainer review before any CI workflow runs on PRs from forks. Apply these controls immediately to any pipeline that ingests or builds from Azure Sentinel, Google AI ADK, Apache Doris, Cloudflare Workers SDK, or Python Black source.

NIST Phase: Containment

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

Controls: NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Without an enterprise pipeline governance tool, use the GitHub CLI (`gh api repos/{owner}/{repo}/branches/{branch}/protection`) to enumerate branch protection status across all repos consuming these five packages. Pipe output to `jq` to identify any branch missing `required_pull_request_reviews` or `required_status_checks`. For self-hosted GitHub Actions runners, immediately set `ACTIONS_RUNNER_REQUIRE_JOB_TOKEN=true` and restrict runner labels to trusted workflow files only. Document each repo's current state before making changes.

Evidence: Before disabling or restricting fork PR workflows, capture the current state of all `.github/workflows/*.yml` files via `git log --all --oneline -- .github/workflows/` to establish a baseline of recent changes. Export the full GitHub Actions workflow run history for the past 90 days via the GitHub API (`GET /repos/{owner}/{repo}/actions/runs?created=>YYYY-MM-DD`) before any workflow file is modified or deleted — this log is volatile if runs are purged. Record all currently active workflow run IDs and their triggering fork contributor accounts before disabling triggers.

Step 2: Detection — Review pipeline execution logs for unexpected script execution, outbound network calls, or artifact modifications during PR-triggered builds. Query CI/CD audit logs (GitHub Actions, GitLab CI, Azure DevOps) for workflow runs initiated by external fork contributors within the past 90 days. Inspect workflow definition files (.github/workflows/*.yml or equivalent) for unauthorized changes. Hunt for T1552.001 indicators: scan build logs and environment variable outputs for credential exposure patterns. Reference AU-2 (Event Logging) and AU-6 (Audit Record Review) to ensure pipeline events are captured and reviewed.

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)

Compensating: Without a SIEM, use the GitHub CLI to pull raw workflow run logs: `gh run list --repo {owner}/{repo} --json databaseId,actor,event,conclusion --limit 500 | jq '.[] | select(.event=="pull_request")'` to isolate fork-triggered runs. Download individual run logs with `gh run view {run_id} --log` and pipe through `grep -Ei '(curl|wget|nc |ncat|base64|eval|echo.*\$|export.*TOKEN|export.*SECRET|export.*KEY|ACTIONS_RUNTIME_TOKEN)'` to surface credential exfiltration or shell injection patterns specific to the Cordyceps campaign's pipeline poisoning technique. For Azure DevOps, export pipeline run audit events via `az devops invoke --area audit --resource auditLog --query-parameters startTime={date}`.

Evidence: Capture the following before any workflow files are removed or modified: (1) Full stdout/stderr logs for all PR-triggered workflow runs on repos consuming Azure Sentinel, Google AI ADK, Apache Doris, Cloudflare Workers SDK, or Python Black over the past 90 days — these logs may contain exfiltrated `ACTIONS_RUNTIME_TOKEN` or `GITHUB_TOKEN` values printed by injected `env` or `printenv` commands. (2) The git object hashes of all `.github/workflows/*.yml` files at HEAD and in recent commits (`git log --follow -p .github/workflows/`) to detect unauthorized step injection. (3) Network egress logs from the CI runner host for the exposure window — look for outbound DNS or HTTPS to non-GitHub domains initiated from the build process.

Step 3: Eradication — Remove or quarantine any pipeline workflow files modified by unrecognized external contributors. Rotate all secrets, tokens, and credentials that were accessible to CI/CD pipeline environments potentially exposed during suspect PR builds (D3-CRO: Credential Rotation). Implement required reviewer approvals and 'pull_request_target' scope restrictions in GitHub Actions to prevent fork-sourced PRs from accessing repository secrets. Verify artifact integrity for any builds executed during the exposure window.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: For credential rotation without a secrets management platform: enumerate all repository and organization secrets via `gh secret list --repo {owner}/{repo}` and `gh secret list --org {org}`, then rotate each manually through the respective service's API or console (GitHub PATs, Azure service principals, GCP service account keys). For artifact integrity verification, compute SHA-256 hashes of all build outputs from the exposure window (`sha256sum *.whl *.jar *.tar.gz`) and compare against hashes recorded in the original release pipeline logs before any Cordyceps-era PR ran. Flag any mismatch for quarantine.

Evidence: Before rotating credentials or removing workflow files, capture: (1) A complete dump of all GitHub Actions secrets names (not values) currently scoped to affected repos and environments — establishes what was potentially in scope during compromised runs. (2) The full git diff of any modified `.github/workflows/*.yml` file (`git show {commit_hash}`) to preserve the injected payload for threat intelligence purposes. (3) Build artifact checksums (SHA-256) and provenance metadata for every release built during the exposure window — these are forensic

evidence of potential supply chain poisoning and must be preserved before any artifact is pulled from distribution or overwritten.

Step 4: Recovery — Re-run clean builds from verified source commits after eradication. Validate artifact hashes against known-good baselines before redistributing or deploying build outputs. Monitor pipeline environments post-remediation for anomalous execution patterns using SIEM alerting aligned with AU-6 (Audit Record Review, Analysis, and Reporting). Confirm that branch protection rules and PR workflow restrictions are enforced across all affected repositories.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-2 (Event Logging), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without a SIEM, implement post-remediation monitoring using a Sigma rule converted to a grep-based log scanner targeting GitHub Actions run logs for the same credential-exfiltration patterns identified in Step 2. Schedule a daily cron job: `gh run list --repo {owner}/{repo} --json actor,event,conclusion,headBranch --limit 100 | jq '.[] | select(.event=="pull_request" and .actor.type=="User")' >> /var/log/pipeline_audit.log`. Validate clean build artifact hashes by comparing `sha256sum` output against hashes independently recorded from the last known-good release tag before the Cordyceps campaign's earliest identified PR submission date.

Evidence: Recovery builds should be triggered only from commits that predate the earliest identified Cordyceps campaign PR or from commits that have passed post-eradication maintainer review. Before redistributing any artifact produced during recovery, record its SHA-256 hash and the exact git commit SHA used as the build source — this establishes a signed chain of custody. If any downstream consumer (e.g., an organization pulling Python Black or Azure Sentinel content) has already ingested a potentially poisoned artifact, flag that as a separate sub-incident requiring its own containment assessment under NIST 800-61r3 §3.3.

Step 5: Post-Incident — Conduct a formal review of CI/CD pipeline security posture against NIST SI-4 monitoring controls and CIS 4.6 (Securely Manage Enterprise Assets and Software). Implement mandatory code signing for build artifacts to enable downstream integrity verification (D3-FMBV: File Magic Byte Verification as a partial analog). Establish a software bill of materials (SBOM) process to track open source dependencies. Document lessons learned and update the software supply chain risk section of your third-party risk management program.

NIST Phase: Post Incident

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

Controls: CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), NIST AU-11 (Audit Record Retention)

Compensating: Without an enterprise supply chain security tool, generate SBOMs using the free `syft` CLI (`syft packages {image_or_dir} -o spdx-json`) for build environments that consume Azure Sentinel, Google AI ADK, Apache Doris, Cloudflare Workers SDK, or Python Black. Implement artifact signing using the free `cosign` tool (Sigstore project) to produce verifiable signatures for all release artifacts going forward. Store lessons-learned documentation in a version-controlled internal wiki and cross-reference it with the Cordyceps campaign's specific TTPs — fork-submitted workflow injection and `pull_request_target` privilege escalation — so future PR reviews include explicit checks for these patterns.

Evidence: Retain all forensic artifacts collected during this incident for a minimum period consistent with your audit record retention policy under NIST AU-11 (Audit Record Retention): this includes the captured workflow run logs, git diffs of modified workflow files, build artifact checksums from the exposure window, and the rotated credentials inventory. These records support regulatory notification decisions if poisoned artifacts were distributed to downstream organizations. Document the earliest and latest dates of suspected malicious PR activity to bound the exposure window for any third-party disclosure obligations.

Detection Guidance

Focus detection on CI/CD pipeline execution logs and workflow configuration files. In GitHub Actions, query audit logs for 'workflow_run' and 'pull_request' events originating from forked repositories by non-organization accounts. Flag any workflow run that initiates outbound network connections to unexpected destinations during build or test phases. In Azure DevOps or GitLab CI equivalents, alert on pipeline jobs triggered by unreviewed external contributions that access secret variables or artifact storage. Behavioral indicators aligned with T1059 include unexpected shell interpreter invocations (bash, PowerShell, Python) within build steps not present in the baseline workflow definition. For T1552.001, scan build logs for patterns matching secret formats (API keys, tokens, cloud credentials) written to stdout or log files. For T1195.001, compare current workflow YAML files against the last known-good committed version and alert on diffs introducing new external script fetches or dependency URLs. NIST 800-53 AU-2 and AU-12 require that these pipeline events be logged; confirm your logging pipeline captures workflow trigger source, actor identity, and execution output. CIS 8.2 (Collect Audit Logs) applies: verify CI/CD audit logging is enabled and forwarded to your SIEM.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://www.darkreading.com/application-security/cordyceps-malicious-pull-requests-developer-workflows`	Primary reporting source for Cordyceps campaign — T3, Dark Reading. No authoritative IOCs confirmed from CISA or vendor advisories at time of writing.	LOW

Framework Mappings

MITRE-ATTACK

- **T1554** — Compromise Host Software Binary
- **T1059** — Command and Scripting Interpreter
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1072** — Software Deployment Tools
- **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
- **IA-2** — Identification and Authentication (Organizational Users)

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **6.3** — Require MFA for Externally-Exposed Applications

ISO-27001-2022

- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1554	Compromise Host Software Binary	Persistence
T1059	Command and Scripting Interpreter	Execution
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1072	Software Deployment Tools	Execution
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Security News	https://www.darkreading.com/application-security/cordyceps-maliciou...	T3
Microsoft Sentinel—AI-Ready Platform | Microsoft Security	https://www.microsoft.com/en-us/security/business/siem-and-xdr/micr...	T1
Kickstart AI Security with Microsoft Defender for Cloud	https://www.youtube.com/watch?v=rdN67slPqtY	T3
Microsoft responds to security challenges facing code, AI ...	https://www.helpnetsecurity.com/2026/06/03/microsoft-ai-agent-secur...	T3
Cybersecurity News	https://aiagentsdirectory.com/news/topic/cybersecurity	T3

Generated 2026-06-24 06:22 UTC by TJS Security Command Center