INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-02 06:12 UTC

Aqua Security Trivy & KICS Supply Chain Attack, Persistent Re-Compromise After Initial Containment

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.0

SCC Item ID	SCC-CAM-2026-0138
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.0
Affected Products	Aqua Security Trivy (incl. malicious v0.69.4 release), trivy-action, setup-trivy, KICS GitHub Action; GitHub Actions pipeline consumers of these tools
Published	2026-04-01
Discovery Source	Gemini

Executive Summary

Attackers compromised Aqua Security's GitHub Actions infrastructure for the Trivy vulnerability scanner and KICS static analysis tool, hijacking approximately 75 release tags and injecting malicious code into CI/CD workflows. After Aqua's initial containment, the threat actor re-established access and published a second malicious release (v0.69.4), confirming persistence beyond the initial intrusion. Any organization running Trivy GitHub Actions, trivy-action, setup-trivy, or KICS GitHub Action in their pipelines is at risk of executing attacker-controlled code during their software build and deployment processes.

Technical Analysis

This multi-stage supply chain attack targeted Aqua Security's GitHub Actions ecosystem. The threat actor, attributed at medium confidence to 'TeamPCP' (per Wiz), compromised Aqua's GitHub Actions infrastructure and hijacked ~75 tags across the trivy-action and setup-trivy repositories, injecting malicious code into workflow steps. A parallel compromise of the KICS GitHub Action was also identified. Following initial remediation by Aqua, the attacker published a second malicious release tagged v0.69.4, indicating either a persistence mechanism survived containment or an unresolved access vector was re-exploited. Relevant CWEs: CWE-1357 (Reliance on Insufficiently Trustworthy Component), CWE-693 (Protection Mechanism Failure), CWE-494 (Download of Code Without Integrity Check). MITRE ATT&CK techniques: T1195.001 (Supply Chain Compromise: Compromise Software Dependencies), T1554 (Compromise Client Software Binary), T1586.003 (Compromise Accounts: Cloud Accounts), T1059 (Command and Script Interpreter), T1071.001 (Application

Layer Protocol: Web Protocols). No CVE has been assigned. Malicious release v0.69.4 is confirmed compromised; all pipeline consumers of affected GitHub Actions should treat recent executions as potentially tainted.

Action Checklist

1. Step 1: Containment, Immediately pin all references to Aqua Security GitHub Actions (aquasecurity/trivy-action, aquasecurity/setup-trivy, checkmarx/kics-github-action) to a known-good SHA commit hash rather than a version tag or 'latest'. Do not use any tag reference until Aqua confirms full remediation. Remove or disable any pipeline jobs referencing these actions until you have validated their state. Monitor Aqua's official incident discussion at https://github.com/aquasecurity/trivy/discussions/10425 for authoritative remediation guidance.
2. Step 2: Detection, Review CI/CD pipeline logs for any execution of trivy-action, setup-trivy, or the KICS GitHub Action between the initial compromise date (approximately 2026-03-19) and present. Look for unexpected outbound network connections (T1071.001) originating from runner environments during those job steps. Examine artifact outputs and runner process trees for anomalous script execution (T1059). Check GitHub Actions workflow run logs for unexpected steps, environment variable exfiltration patterns, or calls to external endpoints not present in the action's published source. Query your SIEM for GitHub Actions runner logs if your runners are self-hosted.
3. Step 3: Eradication, Replace all tag-based references to affected actions with verified SHA-pinned references tied to commits predating the compromise window, confirmed clean by Aqua's official advisory. If v0.69.4 of Trivy was installed in any environment directly, remove it and replace with the latest verified clean release per Aqua's advisory. Rotate any secrets, tokens, or credentials that were accessible to compromised workflow steps, including GitHub tokens, cloud provider credentials, and registry credentials scoped to affected runners.
4. Step 4: Recovery, After re-pinning actions to verified commits, re-run affected pipelines in an isolated environment and validate outputs against a known baseline. Confirm no unauthorized artifacts were published to registries or package repositories during the compromise window. Verify that secrets rotated in Step 3 are no longer referenced by compromised workflow versions. Establish monitoring for unexpected outbound connections from CI/CD runner environments on an ongoing basis.
5. Step 5: Post-Incident, This incident exposes a systematic control gap: tag mutability in GitHub Actions allows supply chain substitution without consumer awareness. Mandate SHA-pinned action references across all pipelines as a policy control. Implement a CI/CD dependency inventory to track all third-party actions in use. Evaluate tools such as StepSecurity Harden-Runner or equivalent for runtime monitoring of GitHub Actions. Review your vendor risk posture for open-source security tooling; tools used in security pipelines represent high-value targets and warrant the same scrutiny as production dependencies.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO and legal counsel immediately if evidence from Step 2 or Step 3 confirms that a GitHub Actions secret (GITHUB_TOKEN, cloud provider key, registry credential, or signing key) was exfiltrated and used by the threat actor — this constitutes a confirmed breach of production credential material with potential regulatory notification obligations, and if any published artifacts (container images, packages, binaries) were signed with a compromised key during the window, downstream consumers are themselves at risk and coordinated disclosure is required.
Recovery Notes	Before resuming normal pipeline operations, conduct a controlled re-execution of each affected pipeline with Harden-Runner or equivalent egress monitoring enabled and validate that no network connections are made to endpoints outside the action's documented legitimate infrastructure. Because the threat actor demonstrated persistence by re-establishing access after Aqua's initial containment and publishing a second malicious release (v0.69.4), treat Aqua's remediation confirmation as a necessary but not sufficient gate — independently verify the SHA of any re-pinned commit against the action's git history and the code diff before trusting it. Maintain heightened monitoring of CI/CD runner egress and GitHub Actions workflow run logs for a minimum of 30 days post-recovery, given the confirmed adversary capability to re-compromise after initial remediation.
Forensic Artifacts	GitHub Actions workflow run logs (step-level, including `##[group]` markers) for all executions of aquasecurity/trivy-action, aquasecurity/setup-trivy, and checkmarx/kics-github-action from 2026-03-19 to present — these logs will reveal injected steps, unexpected curl/wget calls, or environment variable dumps not present in the action's published YAML source | Self-hosted runner Sysmon Event ID 1 (Process Creation) and Event ID 3 (Network Connection) records scoped to processes spawned under the runner service account during action execution windows, specifically filtering for child processes of `node` or `runner.Worker` making outbound connections to non-Aqua, non-GitHub infrastructure | Filesystem artifacts in the runner's `_work/` and `$RUNNER_TEMP` directories — malicious code injected into trivy-action's entrypoint.sh or action.yml composite steps may have staged payloads, written exfiltration scripts, or cached stolen credentials in these ephemeral directories before cleanup | SHA-256 hash of any Trivy binary at version 0.69.4 found in runner environments, container images, or CI artifact caches — compare against the known-malicious binary hash from Aqua's advisory to confirm exposure, and retain the binary itself under chain-of-custody if legal proceedings are anticipated | Cloud provider IAM and API access logs (AWS CloudTrail, GCP Cloud Audit Logs, Azure Monitor) for the compromise window, filtering on API calls attributed to credentials that were in scope of affected workflow jobs — unexpected AssumeRole, GetSecretValue, or registry push events from CI/CD service accounts during this window are the primary indicator of credential exfiltration and downstream exploitation

Per-Action IR Details

Step 1: Containment — Immediately pin all references to Aqua Security GitHub Actions (aquasecurity/trivy-action, aquasecurity/setup-trivy, checkmarx/kics-github-action) to a known-good SHA commit hash rather than a version tag or 'latest'. Do not use any tag reference until Aqua confirms full remediation. Remove or disable any pipeline jobs referencing these actions until you have validated their state. Monitor Aqua's official incident discussion at https://github.com/aquasecurity/trivy/discussions/10425 for authoritative remediation guidance.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-3 (Configuration Change Control), NIST SA-12 (Supply Chain Protection), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Run `grep -r 'aquasecurity/trivy-action\|aquasecurity/setup-trivy\|checkmarx/kics-github-action' .github/workflows/` across all repos to enumerate every reference. For each match, replace tag-based refs (e.g., `@v0.19.0` or `@master`) with a full 40-character SHA using `git ls-remote https://github.com/aquasecurity/trivy-action.git` to retrieve commit hashes for known-good releases. Use `gh workflow disable ` via the GitHub CLI to immediately halt pipeline execution without deleting workflow files, preserving them as forensic evidence.

Evidence: Before disabling pipelines, capture full GitHub Actions workflow run logs for all executions of trivy-action, setup-trivy, and kics-github-action since 2026-03-19 via the GitHub API: `gh run list --workflow= --json databaseId,conclusion,createdAt,headSha | gh run view --log`. Archive the raw YAML of all affected workflow files at their current committed state using `git show HEAD:.github/workflows/.yml > evidence__$(date +%s).yml` before any remediation commits overwrite them.

Step 2: Detection — Review CI/CD pipeline logs for any execution of trivy-action, setup-trivy, or the KICS GitHub Action between the initial compromise date (approximately 2026-03-19) and present. Look for unexpected outbound network connections (T1071.001) originating from runner environments during those job steps. Examine artifact outputs and runner process trees for anomalous script execution (T1059). Check GitHub Actions workflow run logs for unexpected steps, environment variable exfiltration patterns, or calls to external endpoints not present in the action's published source. Query your SIEM for GitHub Actions runner logs if your runners are self-hosted.

NIST Phase: Detection Analysis

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

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

Compensating: On GitHub-hosted runners, retrieve all workflow run logs for the compromise window via: `gh api /repos/{owner}/{repo}/actions/runs --paginate | jq '.workflow_runs[] | select(.created_at >= "2026-03-19") | {id, name, conclusion, head_sha}'`. On self-hosted runners, enable Sysmon with SwiftOnSecurity's config and query for network connections (Event ID 3) and process creation (Event ID 1) where ParentImage contains `runner` or `node` and CommandLine contains `curl`, `wget`, `Invoke-WebRequest`, or base64-encoded strings. Apply the Sigma rule `proc_creation_win_susp_base64_decode.yml` against runner process logs to detect T1059 script execution. For outbound exfiltration (T1071.001), run `tcpdump -i any -w runner_capture.pcap` on self-hosted runner hosts during a controlled re-execution of a sanitized test pipeline and diff DNS queries against the action's known legitimate endpoints.

Evidence: Collect GitHub Actions runner diagnostic logs from `_diag/` directory on self-hosted runners (default path: `/home/runner/_diag/` on Linux, `C:\actions-runner\_diag\` on Windows) covering the full window from 2026-03-19 to present. For each affected workflow run, capture the full step-level log including the `##[group]` markers that delineate action execution boundaries — these logs will reveal injected steps not present in the workflow YAML. On self-hosted runners, extract Sysmon Event ID 3 (Network Connection) records filtering on processes spawned during action execution, and Event ID 11 (FileCreate) for any files written to TEMP or runner work directories by the trivy or kics action steps. Check for environment variable dumps by searching runner logs for patterns matching `GITHUB_TOKEN`, `AWS_SECRET_ACCESS_KEY`, `DOCKER_PASSWORD`, or any credential-shaped strings using: `grep -E '(ghp_|ghs_|AKIA|eyJ)[A-Za-z0-9+/]{20,}' runner_log_*.txt`.

Step 3: Eradication — Replace all tag-based references to affected actions with verified SHA-pinned references tied to commits predating the compromise window, confirmed clean by Aqua's official advisory. If v0.69.4 of Trivy was installed in any environment directly, remove it and replace with the latest verified clean release per Aqua's advisory. Rotate any secrets, tokens, or credentials that were accessible to compromised workflow steps, including GitHub tokens, cloud provider credentials, and registry credentials scoped to affected runners.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST IA-5 (Authenticator Management), NIST CM-2 (Baseline Configuration), CIS 5.2 (Use Unique Passwords), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: To verify Trivy v0.69.4 installation across environments, run: `find / -name 'trivy' -exec {} --version \; 2>/dev/null | grep '0.69.4'` on Linux hosts; on Windows: `Get-ChildItem -Path C:\ -Recurse -Filter trivy.exe -ErrorAction SilentlyContinue | ForEach-Object { & $_.FullName --version }`. For credential rotation scope: use `gh secret list --repo /` to enumerate all secrets accessible to affected workflows, cross-reference with your cloud provider's IAM audit logs to determine which credentials were actually loaded into runner environment variables during compromised runs. Use `gh api /repos/{owner}/{repo}/actions/secrets` to confirm rotation timestamps post-incident. For OIDC-based cloud credentials, revoke the associated IAM role trust policy and re-issue rather than rotating a static secret.

Evidence: Before rotating credentials, capture IAM and cloud provider access logs for the compromise window to establish whether exfiltrated tokens were actually used: for AWS, query CloudTrail for API calls from unusual source IPs or user-agents during the window using `aws cloudtrail lookup-events --start-time 2026-03-19 --lookup-attributes AttributeKey=EventName,AttributeValue=AssumeRoleWithWebIdentity`; for GCP, query Cloud Audit Logs for service account key usage. On affected runner hosts, collect the contents of `$RUNNER_TEMP` and the runner's work directory (`_work/`) before cleanup — injected malicious action code from trivy-action or kics-github-action may have staged payloads or written artifacts there. Also capture a filesystem hash manifest of any Trivy binary found at v0.69.4 using `sha256sum $(which trivy)` and compare against the known-malicious hash published in Aqua's advisory.

Step 4: Recovery — After re-pinning actions to verified commits, re-run affected pipelines in an isolated environment and validate outputs against a known baseline. Confirm no unauthorized artifacts were published to registries or package repositories during the compromise window. Verify that secrets rotated in Step 3 are no longer referenced by compromised workflow versions. Establish monitoring for unexpected outbound connections from CI/CD runner environments on an ongoing basis.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CP-10 (System Recovery and Reconstitution), NIST AU-2 (Event Logging), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Re-run pipelines against an ephemeral isolated environment (GitHub Actions with `runs-on: ubuntu-latest` and no access to production secrets) and compare scan output artifacts using `diff <(jq -S . baseline_scan.json) <(jq -S . new_scan.json)` — deviations in trivy scan results unrelated to actual code changes may indicate prior result tampering. To verify no unauthorized container image pushes occurred, query your registry's push event log: for DockerHub, use `GET https://hub.docker.com/v2/repositories/{namespace}/{repo}/tags/?page_size=100` and cross-reference push timestamps against the compromise window. Use StepSecurity's free Harden-Runner GitHub App (no-cost for public repos) to generate a network egress allowlist baseline — install it on a clean pipeline run and flag any deviations in subsequent runs.

Evidence: Before declaring recovery complete, pull the full artifact and release publication history for your affected repositories during the compromise window: `gh api /repos/{owner}/{repo}/releases | jq '.[] | select(.created_at >= "2026-03-19") | {tag_name, created_at, author}'`. Audit container registry push logs, npm/PyPI publication logs, and any binary artifact stores for uploads occurring during compromised pipeline runs. Retain GitHub Actions run logs (which expire after 90 days by default) by exporting them: `gh run view --log > run__$(date +%s).log` for all runs in the compromise window, as these constitute your primary forensic record of what the malicious action code executed.

Step 5: Post-Incident — This incident exposes a systematic control gap: tag mutability in GitHub Actions allows supply chain substitution without consumer awareness. Mandate SHA-pinned action references across all pipelines as a policy control. Implement a CI/CD dependency inventory to track all third-party actions in use. Evaluate tools such as StepSecurity Harden-Runner or equivalent for runtime monitoring of GitHub Actions. Review your vendor risk posture for open-source security tooling — tools used in security pipelines represent high-value targets and warrant the same scrutiny as production dependencies.

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 SA-12 (Supply Chain Protection), NIST CM-3 (Configuration Change Control), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software)

Compensating: Enforce SHA pinning at scale using `grep -r 'uses:' .github/workflows/ | grep -v '@[0-9a-f]\{40\}'` in a pre-commit hook or CI gate to fail any workflow referencing a non-SHA action. For a CI/CD dependency inventory on a budget, use `find .github/workflows -name '*.yml' -exec grep -h 'uses:' {} \; | sort -u > cicd_action_inventory.txt` and schedule this as a weekly cron job with output diffed against the previous week. For Harden-Runner equivalent on self-hosted runners without budget, deploy Falco (open source) with the rule `rule: Unexpected outbound connection from CI runner` filtering on process ancestry from `runner.Listener`. Map this incident to MITRE ATT&CK T1195.001 (Supply Chain Compromise: Compromise Software Dependencies and Development Tools) and T1553.002 (Subvert Trust Controls: Code Signing) in your lessons-learned documentation to ensure detection engineering work targets the correct technique family.

Evidence: Produce a post-incident timeline reconstructed from: GitHub repository audit log exports (`gh api /orgs/{org}/audit-log?phrase=action:git.push&per_page=100`), GitHub Actions run history for the full compromise window, and any WAF or egress proxy logs from self-hosted runner network segments. Document the specific commits where malicious code was injected into trivy-action, setup-trivy, and kics-github-action by diffing the tagged releases against adjacent clean commits (`git diff `), and retain these diffs as permanent incident artifacts to inform future detection rules targeting similar injection patterns in GitHub Actions `action.yml` or `entrypoint.sh` files.

Detection Guidance

Primary detection focus is CI/CD pipeline activity during the compromise window (approximately 2026-03-19 onward). Query GitHub Actions workflow run logs for jobs invoking aquasecurity/trivy-action, aquasecurity/setup-trivy, or checkmarx/kics-github-action, any version or tag. For self-hosted runners, review host-level process execution logs for unexpected child processes spawned by the runner agent during those job steps, particularly interpreters (bash, python, node) executing scripts not present in the committed workflow YAML. Look for outbound DNS or HTTP requests to domains not in your expected action dependency list originating from runner hosts. For cloud-hosted runners, review any cloud provider audit logs for API calls made using credentials scoped to those runners during the affected window. As of publication, no confirmed IOCs (hashes, IPs, domains) were publicly available; monitor Wiz (https://www.wiz.io/blog/trivy-compromised-teampcp-supply-chain-attack) and StepSecurity (https://www.stepsecurity.io/blog/trivy-compromised-a-second-time---malicious-v0-69-4-release) for published IOCs as the investigation progresses. Treat any artifacts produced by pipeline runs during the compromise window as potentially tainted until cleared.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://github.com/aquasecurity/trivy/releases/tag/v0.69.4`	Confirmed malicious Trivy release published by threat actor after initial containment. Do not use or reference.	HIGH

Type	Value	Context	Confidence
URL	`aquasecurity/trivy-action (any tag reference during compromise window ~2026-03-19 onward)`	GitHub Action with hijacked tags; specific malicious tag values not publicly confirmed in available sources at time of writing. Pin to verified SHA.	MEDIUM
URL	`aquasecurity/setup-trivy (any tag reference during compromise window ~2026-03-19 onward)`	GitHub Action with hijacked tags; specific malicious tag values not publicly confirmed in available sources at time of writing. Pin to verified SHA.	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1071.001** — Web Protocols
- **T1586.003** — Cloud Accounts
- **T1059** — Command and Scripting Interpreter
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1554** — Compromise Host Software Binary

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **15.1** — Establish and Maintain an Inventory of Service Providers

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

SOC2-TSC

- **CC9.2** — Manages risks associated with vendors and business partners

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1071.001	Web Protocols	Command-And-Control
T1586.003	Cloud Accounts	Resource-Development
T1059	Command and Scripting Interpreter	Execution
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1554	Compromise Host Software Binary	Persistence

Sources

Source	URL	Tier
Update: Ongoing Investigation and Continued Remediation	https://www.aquasec.com/blog/trivy-supply-chain-attack-what-you-nee...	T3
Trivy Compromised a Second Time - Malicious v0.69.4 Release ...	https://www.stepsecurity.io/blog/trivy-compromised-a-second-time---...	T3
Trivy Compromised by "TeamPCP" | Wiz Blog	https://www.wiz.io/blog/trivy-compromised-teampcp-supply-chain-attack	T3
Trivy Security incident 2026-03-19 #10425 - GitHub	https://github.com/aquasecurity/trivy/discussions/10425	T3
Trivy Security Scanner GitHub Actions Breached, 75 Tags Hijacked ...	https://thehackernews.com/2026/03/trivy-security-scanner-github-act...	T3

Generated 2026-04-02 06:12 UTC by TJS Security Command Center