

Update: Ongoing Investigation and Continued Remediation

By Aqua Team

Published: 2026-04-01 · Archived: 2026-08-06 02:02:05 UTC

Open Source Security Advisory

Update: Wednesday, April 1, 2026 Boston, MA 10:00 AM ET

Over the past week, we have nearly finalized our investigation and are now in the final stages of documentation and review. There continues to be no indication that Aqua's commercial products have been affected.

As part of this process, we identified additional details about the attack and have incorporated insights from both our internal investigation and industry partners, identified by *. To reflect this, we have updated and expanded the sections below, adding further details, IOCs and context to the attack timeline to make it easier to understand.

Update: Wednesday, March 25, 2026 Boston, MA 12:30 AM ET

Our response has progressed into the remediation and documentation phase. With the core investigation and immediate containment actions largely complete, our focus is now on consolidating findings and communicating them clearly to customers and stakeholders.

Working closely with Sygnia, we are developing formal documentation that includes the confirmed timeline, remediation actions, and supporting materials for customer assurance and attestation. A comprehensive review informs this effort of credentials, access controls, and affected systems.

In parallel, we are engaging directly with customers to provide structured updates and ensure they have the information needed to assess any potential impact on their environments.

At this stage, there are no material changes to our previously communicated findings. Should new information emerge that affects the scope or impact of this incident, we will provide updates here.

Update: Tuesday, March 24, 2026 Boston, MA 1:43 AM ET

Our investigation into the Trivy supply chain attack continues, with our focus now shifting towards active remediation.

Working closely with Sygnia, we have established a clear understanding of the multi-stage nature of this attack, including initial credential compromise, incomplete early containment, and subsequent reuse of access to distribute malicious artifacts. We are actively executing remediation actions across all identified vectors while continuing to validate the full scope of potential credential exposure and downstream impact.

Enterprise Environment Isolation

There is still no indication that Aqua's commercial products are affected.

The commercial platform is architecturally isolated from the compromised open-source environment:

- Built and operated entirely separate from GitHub
- No shared repositories, CI/CD infrastructure, secrets, or signing systems
- Dedicated pipelines and access controls, including SSO, IP allowlisting, and ZTNA
- Controlled integration process where the commercial fork lags open-source releases and undergoes a gated security review

As a result, the malicious Trivy v0.69.4 release was never incorporated into the commercial environment, and the GitHub-based attack path does not apply to the commercial build system.

Ongoing Actions

- Revoke and rotate credentials across all environments
- Transition away from long-lived tokens
- Conduct forensic validation and log review with Sygnia
- Strengthen CI/CD and access control protections

This remains an active investigation. We will continue to share updates as we learn more and will provide another update tomorrow.

Update: Monday, March 23, 2026 Boston, MA 2:00 AM ET

We are providing this update to share new developments identified during our ongoing investigation into the Trivy open source incident described below.

Over the weekend, the Trivy team continued the analysis of the previously reported incident and began implementing additional security measures across repositories and automation systems associated with our open-source projects. To further strengthen the response, the Aqua security team engaged the global incident response company Sygnia to assist with forensic investigation and remediation.

During this process, and while onboarding the response team, the team identified additional suspicious activity on Sunday, March 22nd, involving unauthorized changes and repository tampering. Based on our current understanding, this activity is consistent with the attacker's previously observed behavior.

*This additional suspicious activity included the publication of malicious Docker images to Docker Hub (Trivy tags 0.69.5 and 0.69.6) at approximately 16:00 UTC on March 22, as well as the unauthorized publication of internal Aqua repositories to GitHub as public repositories, observed at approximately 21:40 UTC on March 22. Both actions confirm the attacker had reestablished access to Aqua's infrastructure after the initial containment effort.

*Separately, Wiz Research identified a parallel compromise of the kics-github-action GitHub Action (KICS is Aqua's Infrastructure-as-Code scanner), detailed in a separate advisory at <https://www.wiz.io/blog/teamcp-attack-kics-github-action>. Organizations using KICS GitHub Actions should audit their environments using the same remediation guidance provided below.

This development suggests that the incident is part of an ongoing and evolving attack, with the threat actor reestablishing access. Our investigation is actively focused on validating that all access paths have been identified and fully closed.

There is no indication that the versions of Trivy used within Aqua's commercial products are impacted at this time. By design, the forked version of Aqua's commercial platform lags behind Trivy open source due to a controlled integration process. This process allows us to evaluate the code and incorporate new Trivy features.

We will continue to share updates as more information becomes available and will provide a further update, including additional findings, as the investigation progresses.

Initial Post, March 22:

What Happened

On March 19, 2026, a threat actor used compromised credentials to publish malicious releases of **Trivy version 0.69.4**, as well as **trivy-action** and **setup-trivy**. While this activity initially appeared to be an isolated event, it was the result of a broader, multi-stage supply chain attack that began weeks earlier.

Attack Timeline

- **Late February 2026:** Attackers exploited a misconfiguration in Trivy's GitHub Actions environment, extracting a privileged access token and establishing a foothold in repository automation and release processes.
- **March 1, 2026:** The Trivy team disclosed the earlier incident and executed credential rotation. Subsequent investigation revealed the rotation was not fully comprehensive, allowing the threat actor to retain residual access via still-valid credentials.
- **March 19, 2026 (~17:43 UTC):** The attacker force-pushed 76 of 77 version tags in the aquasecurity/trivy-action repository and all 7 tags in aquasecurity/setup-trivy, redirecting trusted references to malicious commits. Simultaneously, the compromised aqua-bot service account triggered release automation to publish a malicious Trivy binary designated v0.69.4.
- **March 19, 2026 (~20:38 UTC):** The Trivy team identified and contained the attack, removing malicious artifacts from distribution channels.
- **March 20, 2026:** Safe versions, user guidance, and indicators of compromise were published for defenders.

*The attacker also attempted to use the compromised `aqua-bot` service account to push malicious workflows to the open source `tfsec`, `traceeshark`, and `trivy-action` repositories. These workflows were designed to steal additional Trivy credentials (not yet confirmed), including GPG keys and credentials for Docker Hub, Twitter, and Slack (Aqua corporate uses Teams). The stolen credentials were intended to be exfiltrated to a Cloudflare Tunnel C2 endpoint (`plug-tab-protective-relay.trycloudflare.com`).

*The attacker made imposter commits by spoofing GitHub users rauchg (pushed to actions/checkout) and DmitriyLewen (pushed to aquasecurity/trivy).

Rather than introducing a new, clearly malicious version, the attackers used a more sophisticated approach. By modifying existing version tags associated with **trivy-action**, they injected malicious code into workflows that organizations were **already running**. Because many CI/CD pipelines rely on version tags rather than pinned commits, these pipelines continued to execute without any indication that the underlying code had changed.

Just a reminder as we share more details, the incident appears to be isolated to the Trivy open source project in GitHub. We have no indication that Aqua Security's commercial offerings were impacted by this activity.

Tag Poisoning Technique

******For each of the 75 compromised trivy-action tags, the attacker executed the following technique:

1. Started from the master HEAD tree (57a97c7e), containing the latest code.
2. Swapped only entrypoint.sh with the infostealer payload, leaving all other files intact.
3. Looked up the original commit the tag previously pointed to.
4. Cloned that original commit's metadata — author name, email, committer, both timestamps, and the full commit message including PR number and "Fixes" references.
5. Set the parent to 57a97c7e (master HEAD) instead of the original parent.
6. Force-pushed the tag to this newly crafted commit.

The result was a file tree identical across all 75 malicious commits (master plus the swapped entrypoint.sh), with only the spoofed commit metadata varying per tag to appear legitimate in git log.

Forensic indicators of the forgery:

- Original commits were GPG-signed by GitHub when merged via the web UI; the attacker's commits are unsigned.
- Each commit claims a date from the original release (2021, 2022, etc.) but has a parent dated March 2026, an impossible lineage.
- Each malicious commit modifies only entrypoint.sh; the originals touched multiple files.
- GitHub's release page shows "0 commits to master since this release" for tags from 2020, even though there should be hundreds of commits.

Why tag 0.35.0 was not poisoned: It already pointed to master HEAD (57a97c7e), the base tree the attacker used. Replacing it would have produced a self-referencing commit and risked drawing attention to the latest release.

GitHub "Immutable" badge caveat: GitHub's release UI displayed "Immutable" badges next to each poisoned tag. The attacker may have deliberately published immutable releases after force-pushing, locking in the malicious state. Organizations should not rely solely on the "Immutable" indicator. Pinning to full commit SHAs remains the only truly immutable protection.

Threat Actor Attribution

,*The malware self-identifies as "TeamPCP Cloud stealer" in a Python comment embedded in the filesystem credential harvester payload. TeamPCP (also tracked as DeadCatx3, PCPcat, and ShellForce) is a documented

cloud-native threat actor known for exploiting misconfigured Docker APIs, Kubernetes clusters, Ray dashboards, and Redis servers. The group has been linked to worm-driven ransomware, data exfiltration, and cryptomining campaigns, and was profiled by Flare and reported by The Hacker News in February 2026. Wiz tracks the actor formally at threats.wiz.io/all-actors/teampcp.

The self-labeling could be a false flag, but the technical overlap with prior TeamPCP tooling, including the cloud-native theft-and-monetization profile, the emphasis on cryptocurrency wallet targeting, and the use of ICP-hosted infrastructure, makes genuine attribution plausible.

Technical Analysis of the Malicious Payloads

In affected environments, the payload was designed to collect sensitive information, including API tokens, cloud credentials (AWS, GCP, Azure), SSH keys, Kubernetes tokens, Docker configuration files, Git credentials, and other secrets available within CI/CD systems. Critically, the malware executed prior to legitimate Trivy scanning logic, so compromised workflows appeared to complete normally while silently exfiltrating data to attacker-controlled infrastructure via two pathways (see IOCs below).

Malicious GitHub Action Payload (entrypoint.sh)

******The malicious entrypoint.sh injected into both trivy-action and setup-trivy is 204 lines long. Lines 4–105 contain the infostealer; lines 106–204 contain the legitimate Trivy scanning code. Because the malware executes first and the real scan follows as usual, users see the expected output and may not notice anything is wrong. The payload operates in three distinct stages:

Stage 1 — Collection (lines 4–36):

- The malware locates GitHub Actions runner processes (Runner.Worker, Runner.Listener, runsvc, run.sh) and reads null-delimited environment variables from /proc/<pid>/environ, filtering for keys containing “env” or “ssh”.
- **On GitHub-hosted runners:** A base64-encoded Python script is decoded and executed with sudo (GitHub-hosted Linux runners provide passwordless sudo). It locates the Runner.Worker process, parses /proc/<pid>/maps to identify all readable memory regions, and dumps each region from /proc/<pid>/mem. The shell wrapper then searches for the JSON pattern {“value”:“<secret>”, “isSecret”: true} — the internal format GitHub Actions uses to store secrets in the Runner.Worker .NET process heap.
- **On self-hosted runners:** A comprehensive filesystem credential harvester (self-identified as “TeamPCP Cloud stealer”) targets 50+ sensitive file paths across SSH, Git, AWS, GCP, Azure, Kubernetes, Docker, databases, CI/CD configs, TLS private keys, VPN configs, shell history, and cryptocurrency wallets.

Category	Targets
Reconnaissance	hostname, whoami, uname -a, ip addr, ip route, printenv
SSH	~/.ssh/id_rsa, id_ed25519, id_ecdsa, id_dsa, authorized_keys, known_hosts, config; /etc/ssh/ssh_host_*_key

Git	~/.git-credentials, ~/.gitconfig
AWS	~/.aws/credentials, ~/.aws/config, AWS_* env vars, EC2 IMDS (169.254.169.254), ECS (169.254.170.2)
GCP	~/.config/gcloud/*, application_default_credentials.json, GOOGLE_APPLICATION_CREDENTIALS
Azure	~/.azure/*, AZURE_* env vars
Kubernetes	~/.kube/config, /etc/kubernetes/*.conf, service account tokens, kubectl get secrets
Docker	~/.docker/config.json, /kaniko/.docker/config.json
Database	~/.pgpass, ~/.my.cnf, ~/.mongorc.js, /etc/mysql/my.cnf, /etc/redis/redis.conf
Package mgrs	~/.npmrc, ~/.vault-token, ~/.netrc
CI/CD configs	terraform.tfvars, terraform.tfstate, .gitlab-ci.yml, .travis.yml, Jenkinsfile, .drone.yml
TLS keys	*.key, *.pem, *.p12, *.pfx; /etc/ssl/private/, /etc/letsencrypt/
VPN	/etc/wireguard/*.conf, wg showconf all
Crypto wallets	Bitcoin, Litecoin, Dogecoin, Zcash, Dash, Ripple, Monero; Ethereum keystores; Cardano keys; Solana validator keypairs; Ledger device files; Anchor deploy keys
Shell history	~/.bash_history, ~/.zsh_history, ~/.mysql_history, ~/.psql_history

Stage 2 — Encryption (lines 39–68):

- Generates 32 bytes of random session material.
- Encrypts collected data with AES-256-CBC using PBKDF2 key derivation.
- Wraps the session material with the attacker’s embedded RSA-4096 public key using OAEP padding.
- Bundles encrypted payload and wrapped session key into tpcp.tar.gz.

Stage 3 — Exfiltration (lines 70–104):

- **Primary channel:** HTTPS POST to scan.aquasecurity[.]org (typosquat) with X-Filename: tpcp.tar.gz header.
- **Fallback channel:** If primary fails and INPUT_GITHUB_PAT is available, creates a public repository named tpcp-docs on the victim’s own GitHub account, creates a timestamped release, and uploads the encrypted bundle as a release asset. Note: requires a user-scoped PAT; the default GITHUB_TOKEN would fail.
- **Cleanup:** All temporary files are removed.

Malicious Trivy Binary (v0.69.4)

*When the malicious binary is executed, it starts both the legitimate Trivy service and the malicious code in parallel.

Data Theft: The malicious process gathers environment variables, scans the filesystem for stored credentials, and enumerates network interfaces. It then compresses and encrypts the data and attempts to exfiltrate it via a POST request to `scan.aquasecurity[.]org`. If that fails, the malware attempts to use a stolen `GITHUB_TOKEN` to create a `tpcp-docs` repository and post the data there.

Persistence Mechanism: The malicious binary checks whether it is running on a developer machine (not GitHub Actions) by verifying that `GITHUB_ACTIONS` is not “true”. If that check passes, it writes a Python dropper to `~/.config/systemd/user/sysmon.py` and creates a systemd user unit that runs the script persistently. `sysmon.py` initially sleeps for 5 minutes, then polls the ICP-hosted C2 canister (`tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0.io`). If a non-YouTube link is found, the dropper downloads to `/tmp/pglog` and executes it in a new session.

At the time of initial analysis (March 20), the ICP endpoint was set to a RickRoll URL. By March 22, the ICP-hosted fallback C2 was actively serving an iteratively developed payload designated `kamikaze.sh`. As of March 22, 21:31 UTC, the ICP canister was marked “Unavailable Due to Policy Violation.”

Developer Machine Remediation: Any system on which the malicious binary was executed outside of GitHub Actions should be checked for the presence of `~/.config/systemd/user/sysmon.py` and associated systemd unit files, and these should be removed immediately.

What Was Affected

The following open source components were affected:

- **Trivy binary release:** v0.69.4
- **GitHub Action aquasecurity/trivy-action:** 76 of 77 version tags force-pushed to malicious commits (only v0.35.0 was unaffected, protected by GitHub’s immutable releases feature)
- **GitHub Action aquasecurity/setup-trivy:** multiple version tags compromised

Any CI/CD workflow that referenced these actions via a mutable version tag, or that downloaded:

- **trivy v0.69.4**, between approximately **18:22 UTC and 21:42 UTC on March 19, 2026**,
- **trivy-action v0.69.4**, between approximately **17:43UTC UTC on March 19, 2026 and 05:40 UTC on March 20, 2026**,
- **setup-trivy**, between approximately **17:43 UTC and 21:44 UTC on March 19, 2026**,

These should be treated as potentially compromised. All secrets accessible to those runner environments must be considered exposed.

What Was Not Affected

There is no indication that Aqua Security’s commercial products were impacted by this incident, including Trivy as delivered within the Aqua Platform.

This statement does not apply to the independent use of open-source Trivy components outside the Aqua Platform. Users who consume open-source Trivy directly should follow the remediation guidance below.

Enterprise Environment Isolation

The commercial platform is architecturally isolated from the compromised open-source environment:

- Built and operated entirely separate from GitHub
- No shared repositories, CI/CD infrastructure, secrets, or signing systems
- Dedicated pipelines and access controls, including SSO, IP allowlisting, and ZTNA
- Controlled integration process where the commercial fork lags open-source releases and undergoes a gated security review

As a result, the malicious Trivy v0.69.4 release was never incorporated into the commercial environment, and the GitHub-based attack path does not apply to the commercial build system.

What Actions We Are Taking

Our corporate security and engineering teams are actively working in close coordination with the Trivy maintainers and the broader security community to investigate the incident and ensure full containment. Steps taken include:

- **Artifact removal:** All malicious releases, including v0.69.4 binaries across GitHub Releases, Docker Hub, GHCR, and ECR, have been deleted. **[UPDATE — Source: Wiz/Socket]** This includes the subsequently published Docker Hub images tagged 0.69.5 and 0.69.6.
- **Tag restoration:** All compromised version tags have been deleted or repointed to known-safe, verified commits.
- **Credential revocation:** A comprehensive lockdown of all automated actions, service accounts, and tokens across the Aqua Security open-source organization has been implemented.
- **Access control hardening:** Stricter safeguards have been implemented around automation and token usage, including tightened permissions and reduced reliance on long-lived credentials.
- **Immutable release enforcement:** We are implementing immutable release verification and provenance attestations for all future deployments.
- **Ongoing monitoring:** We continue to analyze the full scope of the incident and monitor for signs of downstream impact.

GitHub Security Advisory: [GHSA-cxm3-wv7p-598c](https://github.com/advisories/GHSA-cxm3-wv7p-598c)

Ongoing updates: github.com/aquasecurity/trivy/discussions/10425

Required Actions for the Community

For users of open source Trivy, immediate action is required.

Step 1: Update to Known-Safe Versions

Component	Safe Version	Reference
Trivy binary	v0.69.2–v0.69.3	GitHub Releases
trivy-action	v0.35.0	GitHub Action
setup-trivy	v0.2.6	GitHub Action

Step 2: Rotate All Potentially Exposed Secrets

If there is any possibility that a compromised version ran in your environment, all secrets accessible to affected pipelines must be treated as exposed and rotated immediately:

- Credentials for cloud providers (AWS, GCP, Azure)
- Source control and Git credentials
- Container registry credentials
- SSH keys and Kubernetes tokens
- Environment variables and other automation secrets
- **NPM publish tokens:** treat as actively compromised; stolen tokens are being weaponized to propagate malware across the NPM ecosystem.
- ****Cryptocurrency wallets and validator keys:** Bitcoin, Litecoin, Dogecoin, Zcash, Dash, Ripple, Monero configs; Ethereum keystores; Cardano signing/verification keys; Solana keypairs (validator-keypair.json, vote-account-keypair.json, identity.json); Ledger device files; Anchor deploy keys. Rotate or transfer to new wallets immediately.
- ****Database credentials:** ~/.pgpass, ~/.my.cnf, ~/.mongorc.js, Redis config files, and environment variables matching DATABASE, DB_, MYSQL, POSTGRES, MONGO, REDIS, VAULT.
- ****TLS private keys:** *.key, *.pem, *.p12, *.pfx, including /etc/ssl/private/ and /etc/letsencrypt/.
- ****VPN configurations:** WireGuard configuration files in /etc/wireguard/.

Step 3: Audit Trivy Versions

Check whether your organization pulled or executed Trivy v0.69.4 from any source. Remove any affected artifacts immediately.

Also check for Docker images tagged 0.69.5 and 0.69.6, published to Docker Hub on March 22.

Step 4: Audit GitHub Action References

Review all workflows using aquasecurity/trivy-action or aquasecurity/setup-trivy. Check workflow run logs from March 19–20, 2026, for signs of compromise.

Also review any workflows referencing aquasecurity/kics-github-action, which was subject to a parallel compromise identified on March 23.

Step 5: Search for Exfiltration Artifacts

Look for repositories named `tpcp-docs` in your GitHub organization. The presence of such a repository may indicate successful exfiltration via the fallback mechanism.

*On developer machines where the malicious Trivy binary may have been executed, check for the persistence dropper at `~/.config/systemd/user/sysmon.py` and associated systemd user unit files. Remove immediately if found.

Step 6: Long-Term Hardening — Pin to Full SHA Hashes

Pin GitHub Actions to full, immutable commit SHA hashes — not mutable version tags. Version tags can be moved to point at malicious commits.

Example:

UNSAFE: `uses: aquasecurity/trivy-action@0.34.2`

SAFE: `uses: aquasecurity/trivy-action@57a97c7e7821a5776cebc9bb87c984fa69cba8f1`

****Note:** GitHub's "Immutable" release badge does not prevent tag force-pushing. Pinning to full commit SHA remains the only reliable protection.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

UNSAFE — mutable tag, can be silently redirected to malicious code

`uses: aquasecurity/trivy-action@v0.34.2`

SAFE — pinned to an immutable commit SHA

`uses: aquasecurity/trivy-action@57a97c7e7821a5776cebc9bb87c984fa69cba8f1`

UNSAFE — mutable tag, can be silently redirected to malicious code `uses: aquasecurity/trivy-action@v0.34.2`

SAFE — pinned to an immutable commit SHA `uses: aquasecurity/trivy-action@57a97c7e7821a5776cebc9bb87c984fa69cba8f1`

```
# UNSAFE - mutable tag, can be silently redirected to malicious code
uses: aquasecurity/trivy-action@v0.34.2
# SAFE - pinned to an immutable commit SHA
uses: aquasecurity/trivy-action@57a97c7e7821a5776cebc9bb87c984fa69cba8f1
```

Indicators of Compromise (IOCs)

Network and Infrastructure IOCs

Indicator Type	IOC Value	Recommended Action
Network C2	scan.aquasecurtiy[.]org	Block at network perimeter; hunt DNS logs
Network IP	45.148.10.212	Block at firewall; search outbound connections
Secondary C2	plug-tab-protective-relay.trycloudflare.com	Search DNS logs; flag for exfiltration
GitHub Exfil	Repository: tcp-docs	Search GitHub org for unauthorized repo creation
Compromised Binary	trivy v0.69.4	Search container registries and CI caches
ICP Blockchain C2	tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0.io	Block egress to icp0.io

Malicious Binary Hashes

*SHA-256 hashes of the malicious Trivy v0.69.4 binaries:

Hash (SHA-256)	Platform
887e1f5b5b50162a60bd03b66269e0ae545d0aef0583c1c5b00972152ad7e073	FreeBSD-64bit
f7084b0229dce605ccc5506b14acd4d954a496da4b6134a294844ca8d601970d	Linux-32bit
822dd269ec10459572dfaafe163dae693c344249a0161953f0d5cdd110bd2a0	Linux-64bit
bef7e2c5a92c4fa4af17791efc1e46311c0f304796f1172fce192f5efc40f5d7	Linux-ARM
e64e152afe2c722d750f10259626f357cdea40420c5eedae37969fbf13abbecf	Linux-ARM64 (unconfirmed)
ecce7ae5ffc9f57bb70efd3ea136a2923f701334a8cd47d4fbf01a97fd22859c	Linux-PPC64LE
d5edd791021b966fb6af0ace09319ace7b97d6642363ef27b3d5056ca654a94c	Linux-s390x
e6310d8a003d7ac101a6b1cd39ff6c6a88ee454b767c1bdce143e04bc1113243	macOS-64bit
6328a34b26a63423b555a61f89a6a0525a534e9c88584c815d937910f1ddd538	macOS-ARM64
0880819ef821cff918960a39c1c1aada55a5593c61c608ea9215da858a86e349	Windows-64bit

****SHA-256 hash of the malicious GitHub Action payload:**

18a24f83e807479438dcab7a1804c51a00dafc1d526698a66e0640d1e5dd671a — entrypt.sh (injected into trivy-action and setup-trivy)

Compromised GitHub Action Workflow Hashes

****The full catalog of all 75 compromised trivy-action tags and 7 setup-trivy tags is maintained by Socket at:**
socket.dev/supply-chain-attacks/trivy-github-actions-compromise

setup-trivy (7 tags):

Action	Malicious Commit SHA
setup-trivy	8afa9b9f9183b4e00c46e2b82d34047e3c177bd0
setup-trivy	386c0f18ac3d7f2ed33e2d884761119f4024ff8a
setup-trivy	384add36b52014a0f99c0ab3a3d58bd47e53d00f
setup-trivy	7a4b6f31edb8db48cc22a1d41e298b38c4a6417e
setup-trivy	6d8d730153d6151e03549f276faca0275ed9c7b2
setup-trivy	99b93c070aac11b52dfc3e41a55cbb24a331ae75
setup-trivy	f4436225d8a5fd1715d3c2290d8a50643e726031

trivy-action (representative sample of 75 tags):

Tag	Malicious Commit SHA
0.0.1	f77738448eec70113cf711656914b61905b3bd47
0.16.0	f4f1785be270ae13f36f6a8cfbf6faaae50e660a
0.18.0	85cb72f1e8ee5e6e44488cd6cbdbca94722f96ed
0.25.0	ddb94181dcbc723d96ffc07fddd14d97e4849016
0.30.0	ad623e14ebdfe82b9627811d57b9a39e283d6128
0.33.0	19851bef764b57ff95b35e66589f31949eeb229d
0.34.0	ab6606b76e5a054be08cab3d07da323e90e751e8
0.34.2	ddb9da4475c1cef7d5389062bdfdfdbbd1394648

A Note to the Broader Ecosystem

We would also like to recognize and thank our industry partners and the broader security community for their role in helping contain this situation. As an open source project, Trivy does not maintain a comprehensive record of its user base. While it is widely adopted across organizations of all sizes, there is no centralized way to notify every user directly. In this context, the rapid response from researchers, partners, and community members has been invaluable.

By identifying suspicious behavior, publishing analyses, and sharing guidance across channels, the community has helped ensure that critical information reached users quickly. We particularly thank the research teams at Aikido Security and CrowdStrike for their rapid technical publications, which materially accelerated response and community awareness.

We additionally recognize the contributions of Wiz Research(*) and Socket Security(**), whose independent analyses provided critical technical depth to the community response. Wiz Research published a comprehensive advisory that includes the SITF threat model diagram, a full binary hash IOC set, and formal threat actor tracking for TeamPCP (wiz.io/blog/trivy-compromised-teampcp-supply-chain-attack). Socket Security published detailed payload reverse-engineering, the complete catalog of all 75 compromised trivy-action workflow hashes, and an ongoing campaign tracking dashboard (socket.dev/blog/trivy-under-attack-again).

What's Next

This remains an active investigation, and we are committed to continuing to share updates as more information becomes available. As confirmed by community researchers at Aikido Security and CrowdStrike, the threat actor has pivoted beyond the initial CI/CD compromise and is actively weaponizing stolen credentials across the broader ecosystem. Organizations should treat this as an ongoing campaign, not a contained incident.

The threat actor has expanded operations to the npm ecosystem via a self-propagating worm dubbed “CanisterWorm,” leveraging stolen NPM publish tokens exfiltrated from compromised CI/CD pipelines. Aikido Security documented this worm at aikido.dev/blog/teampcp-deploys-worm-npm-trivy-compromise. The use of stolen publish tokens to propagate malware across the NPM package registry represents a significant escalation of the campaign’s downstream impact.

Incidents like this underscore a broader reality in today’s threat landscape. Even widely trusted security tools can become targets. As attackers increasingly focus on software supply chains, transparency, rapid response, and community collaboration are essential to minimizing impact.

We will continue to keep the community informed as the situation evolves. All updates will be published to: github.com/aquasecurity/trivy/discussions/10425

**Indicates this was updated with material sourced from Wiz Research.*

***Indicates this was updated with material sourced from Socket Security.*

This advisory is provided for informational purposes. For the latest updates, refer to the linked GitHub discussion.

Source: <https://www.aquasec.com/blog/trivy-supply-chain-attack-what-you-need-to-know>