# Scanner as Attack Vector: Trivy, CanisterWorm, and CI/CD Risk

How Weaponized Security Tooling Cascades into Ecosystem-Wide Compromise

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-21

 

# Key Takeaways

Within a 72-hour window spanning March 19–21, 2026, the security community witnessed a high-impact CI/CD supply chain attack that compromised one of the ecosystem's most widely deployed security scanning tools and cascaded across multiple registries within 27 hours. The threat actor known as TeamPCP compromised Trivy — one of the most widely adopted open-source container and infrastructure scanners, with over 100 million Docker Hub downloads — by poisoning 75 of 76 GitHub Actions version tags and publishing a backdoored release to every major distribution channel [1][2]. The malware embedded in those artifacts stole CI/CD secrets and developer credentials at pipeline execution time, then used those stolen npm tokens as fuel for CanisterWorm, a novel self-propagating npm backdoor discovered the following day across 47 or more packages [3]. The incident is notable not only for its scale, but for two structural innovations: the use of a previously exploited service account token that was never revoked after a first breach six weeks earlier, and what Aikido Security characterized as the first publicly documented abuse of an Internet Computer Protocol (ICP) blockchain canister as a censorship-resistant command-and-control resolver [3].

The practical guidance for organizations is immediate: pin all GitHub Actions to commit SHAs rather than tags, audit CI/CD environments for indicators of compromise, treat any npm package in the `@EmilGroup` or `@opengov` scopes with heightened scrutiny, and establish out-of-band monitoring for CI/CD network egress. The broader lesson is structural: security tooling occupies a uniquely privileged position in the software delivery pipeline, and that privilege makes it a high-value target requiring the same supply chain controls applied to production code.

# Background

## The Trivy Scanner and Its Position in the Ecosystem

Trivy is an open-source vulnerability and misconfiguration scanner developed by Aqua Security. It is capable of scanning container images, Kubernetes clusters, code repositories, infrastructure-as-code configurations, software bills of materials, secrets, and cloud environments from a single tool. Trivy has achieved broad adoption across DevSecOps pipelines, reflected in its 33,000+ GitHub stars and 100 million Docker Hub downloads, with distribution through Docker Hub, GitHub Container Registry,

Amazon ECR Public, Homebrew, and native package repositories [1]. Its deep integration via the `aquasecurity/trivy-action` and `aquasecurity/setup-trivy` GitHub Actions means that a compromise affecting those action repositories touches any CI/CD workflow that references a poisoned tag — given the widespread adoption of these action repositories, a population that could plausibly span tens to hundreds of thousands of CI/CD workflows.

### TeamPCP and the February 2026 Precursor

The March attack did not occur in isolation. On February 27–28, 2026, a separate actor using the GitHub account `hackerbot-claw` (who self-identified in account metadata as an autonomous AI agent — a claim not independently verified) exploited a `pull_request_target` misconfiguration in Trivy's GitHub Actions workflow — a pattern known in the research community as a "Pwn Request" [4]. The `pull_request_target` trigger is designed to run with access to base repository secrets while receiving contributions from forks; when the workflow also checks out fork code, an attacker can execute arbitrary code with those elevated permissions. The attack exfiltrated a Personal Access Token (PAT) belonging to the `aqua-bot` service account, which was then used to delete all 178 historical releases, temporarily rename the repository, and publish malicious VS Code extension versions to the Open VSX marketplace [4][5].

The critical failure that enabled the March 19 attack was that the compromised PAT was apparently never revoked during incident response. TeamPCP, also tracked under observed account handles including DeadCatx3, PCPcat, PersyPCP, and ShellForce [6], is a documented cloud-native threat group with prior campaigns exploiting misconfigured Docker APIs, Kubernetes dashboards, Ray clusters, and Redis servers. Whether hackerbot-claw is a separate actor that incidentally left the PAT available, or an alias or tool used by TeamPCP in the initial access phase, has not been publicly confirmed as of this writing. Regardless of that attribution question, the retention of that PAT gave TeamPCP a ready-made key to Trivy's entire release infrastructure.

## Security Analysis

### The March 19 Compromise: Tag Poisoning at Scale

At approximately 17:43 UTC on March 19, 2026, TeamPCP used the retained `aqua-bot` PAT to execute a systematic tag-poisoning operation across three repositories [2][7]. In Trivy's main repository, a malicious release was published as v0.69.4 and pushed to every distribution channel simultaneously. In

`aquasecurity/trivy-action`, 75 of 76 version tags (0.0.1 through 0.34.2) were force-pushed to point at malicious commits, with only tag 0.35.0 remaining unaffected — a count reported by Wiz as 75 of 76 [2] and by CrowdStrike as 76 of 77 [5]; the discrepancy between these primary sources has not been formally reconciled. The `aquasecurity/setup-trivy` repository saw 7 of 8 tags similarly overwritten, with v0.2.6 surviving intact. The `tfsec` and `traceeshark` repositories were also affected in the same operation [2].

The mechanism exploited a fundamental characteristic of GitHub's release interface: release names, descriptions, and publication dates are stored in the release object, not derived from the tag target. A force-pushed tag that now points to a malicious commit is visually indistinguishable from the original release in GitHub's UI. To further obscure the substitution, imposter commits spoofed the display names of legitimate Aqua Security contributors [2].

The malicious payload embedded in affected artifacts was a five-stage credential stealer. Upon execution, it enumerated `/proc/*/environ` for credential-related environment variables and used `/proc/<pid>/mem` to read process memory from the GitHub Actions Runner.Worker process, specifically targeting secrets marked `isSecret: true` that would otherwise be masked in logs [7]. It then performed comprehensive credential harvesting across platform-specific paths: SSH private keys, Git credentials, AWS and GCP and Azure tokens, Kubernetes configuration files, Docker registry credentials, Terraform state files, and CI/CD configuration secrets. Exfiltrated data was encrypted using a hybrid scheme — AES-256-CBC for the session key wrapped with RSA-4096 — and transmitted to a typosquatted C2 domain `scan.aquasecurtiy[.]org` (note the transposed letters) resolving to IP `45.148.10.212`, hosted at TECHOFF SRV LIMITED in Amsterdam [2][7]. A fallback exfiltration path created a public GitHub repository named `tpcp-docs` and uploaded stolen data as release assets using any captured `INPUT_GITHUB_PAT` environment variables.

On developer machines where the Trivy binary itself was executed directly (outside CI/CD), the payload wrote a Python dropper to `~/.config/systemd/user/sysmon.py` for persistence. Notably, legitimate Trivy functionality continued operating normally throughout — the scanner produced accurate results while silently exfiltrating credentials in the background, a design decision that significantly delayed detection.

The attack was detected by StepSecurity's Harden-Runner platform, which flagged anomalous outbound connections to the C2 domain against Trivy's historical network baseline [7]. The exposure windows were approximately three hours for the main `trivy` v0.69.4 binary, four hours for `setup-trivy`, and twelve hours for `trivy-action` — with cleanup completing around 05:40 UTC on March 20 [2].

## CanisterWorm: Weaponizing Stolen npm Tokens

Based on the timing of the events and Aikido Security's attribution analysis, the npm tokens used to seed the CanisterWorm campaign are assessed to have originated from credentials harvested during the Trivy CI/CD compromise [3]. TeamPCP appears to have weaponized those tokens within approximately 27 hours of the initial Trivy exfiltration to deploy CanisterWorm across the npm ecosystem — first observed on March 20, 2026 at 20:45 UTC and publicly reported by Charlie Eriksen at Aikido Security [3]. The campaign ultimately affected 47 or more npm packages, including 28 packages in the `@EmilGroup` scope, 16 in the `@opengov` scope, and several additional individually-scoped packages including `@teale.io/eslint-config`, `@airtm/uuid-base32`, and `@pypestream/floating-ui-dom` [3].

CanisterWorm takes its name from its novel command-and-control architecture. Rather than pointing implants directly at an attacker-controlled server, the malware queries a canister — a tamperproof smart contract — running on the Dfinity Internet Computer Protocol (ICP) blockchain network [3]. The specific canister endpoint is `https://tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0.io/`. The canister exposes an `update_link` method the attacker calls to rotate the actual payload URL at will, while returning only a plain-text URL to implants. This architecture makes the C2 infrastructure simultaneously censorship-resistant, immune to conventional domain seizure, and dynamically reconfigurable without touching deployed implants. Aikido Security's analysis characterizes this as the first publicly documented abuse of an ICP canister as a dead-drop C2 resolver in a malware campaign [3]. A kill switch is built in: when the canister returns a YouTube URL, the backdoor skips execution — the attacker arms the campaign by pointing the canister at a real payload binary.

The malware operates in three stages. A Node.js postinstall loader executes via npm's `postinstall` hook during routine package installation, establishing a systemd user service named `pgmon` (styled to resemble PostgreSQL monitoring) at `~/.config/systemd/user/pgmon.service` with `Restart=always` for persistence. That service launches a base64-encoded Python backdoor deployed to `~/.local/share/pgmon/service.py`, which sleeps for five minutes on initialization (a common sandbox-evasion technique) and then polls the ICP canister at approximately fifty-minute intervals. Later iterations of the malware added a `findNpmTokens()` function that scrapes `~/.npmrc`, project `.npmrc`, `/etc/npmrc`, and `NPM_TOKEN` environment variables before auto-spawning a detached `deploy.js` process that enumerates all packages owned by each compromised token, bumps patch version numbers, preserves original READMEs for cover, and republishes with `--tag latest` to ensure default installation [3]. This self-propagation mechanism allowed TeamPCP to publish 28 packages in under 60 seconds in the initial wave.

### Structural Context: Why Security Tools Are Disproportionately Valuable Targets

The Trivy/CanisterWorm incident is a sharp illustration of a structural dynamic that has not yet been systematically addressed in security tooling supply chain guidance. Security scanning tools operate with elevated privileges almost by design: they must read secrets to scan for exposed credentials, access container registries to pull images, and communicate with cloud control planes to audit configurations. In CI/CD pipelines, they frequently execute with access to deployment tokens that touch production infrastructure. A compromise of the scanner itself is therefore self-concealing in a particularly dangerous way — the tool continues producing accurate security findings while simultaneously exfiltrating the secrets it was trusted to protect.

ReversingLabs reported in its 2026 Software Supply Chain Security Report that open-source malware on public registries increased 73% in 2025 compared to the prior year, and characterized 2025 as the year isolated incidents became integrated campaigns [8]. The Trivy/CanisterWorm incident fits that pattern precisely: a first breach establishes a foothold, incident response fails to fully remediate it, a second breach exploits the retained access to compromise distribution infrastructure, and the resulting credential harvest funds a third campaign that self-propagates across an entirely separate ecosystem (npm). The cascade crossed organizational and registry boundaries within hours — a pace that challenges the response capacity of any organization relying on manual review alone.

Parallel campaigns operating in the same period reinforce that this is a systemic problem, not an isolated event. The GlassWorm campaign compromised 72 or more VS Code extensions with an estimated 9 million installs, using transitive dependency trust and Solana blockchain dead-drop C2 [9]. CVE-2025-58178 in the SonarQube Scanner GitHub Action introduced a command injection path (CWE-77, CVSS 7.8 High) exploitable in CI/CD contexts, demonstrating that even well-maintained commercial security tooling can introduce pipeline risk [10]. TeamPCP's own prior campaigns exploited CVE-2025-55182 (CVSS 10.0) in React Server Components to build distributed illicit compute infrastructure, according to Rescana's threat intelligence analysis [6], showing the group's capacity to pivot between attack surfaces as opportunities arise.

## Recommendations

### Immediate Actions

Organizations should immediately audit their GitHub Actions workflows to identify any that reference `aquasecurity/trivy-action`, `aquasecurity/setup-trivy`, or direct Trivy binary downloads. Safe versions are confirmed as `trivy` v0.69.3 or earlier, `trivy-action` at tag

`0.35.0` (commit `57a97c7`), and `setup-trivy` at v0.2.6 (commit `3fb12ec`) [2]. Any workflow that ran between 17:43 UTC March 19 and 05:40 UTC March 20, 2026, using a poisoned version should be treated as potentially compromised: all secrets, tokens, and credentials accessible to those workflows should be rotated immediately, and cloud provider activity logs for that window should be reviewed for anomalous access.

For CanisterWorm exposure, any Linux systems where npm packages from the `@EmilGroup` or `@opengov` scopes were recently installed should be inspected for the indicators of compromise: systemd user services named `pgmon` or `pglog`, state files named `.pg_state`, Python processes running from `~/.local/share/`, and network connections to `tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0.io`.

## Short-Term Mitigations

The most impactful structural change organizations can make immediately is replacing all mutable GitHub Actions version tags with immutable commit SHA pins. A reference like `uses: aquasecurity/trivy-action@0.35.0` is vulnerable to any tag rewrite by the upstream repository; `uses: aquasecurity/trivy-action@57a97c7` is not. This single practice would have neutralized the March 19 tag-poisoning technique for any pipeline that had adopted it — the force-pushed tags would have pointed to malicious commits, but the pinned SHAs would have continued to reference the original, unaffected commits. Tools such as StepSecurity's Harden-Runner and GitHub's own Dependency Review Action can automate SHA pinning and detect unexpected network egress from CI/CD workflows.

Organizations should also implement allowlist-based network egress controls for CI/CD runners. The typosquatted C2 domain used in this attack — `scan.aquasecurtiy[.]org` — would have been blocked immediately in any environment with an outbound allowlist, and in fact the detection by StepSecurity relied on exactly this kind of baseline deviation analysis [7]. Where network allowlisting is impractical, DNS-level monitoring for newly registered or lookalike domains in CI/CD network traffic provides a meaningful compensating control.

Service account credential hygiene deserves specific attention in light of how this attack unfolded. The root cause of the March 19 breach was a PAT that survived a February 28 incident without being rotated. Organizations should establish explicit incident response procedures requiring the immediate rotation of all credentials that could have been exposed, not merely those confirmed to have been accessed. Automated credential rotation triggered by security alerts, combined with short token lifetimes enforced at the identity provider level, would significantly reduce the window of opportunity for follow-on attacks.

### Strategic Considerations

The CanisterWorm campaign's use of an ICP blockchain canister as a C2 resolver points toward an emerging class of infrastructure abuse that conventional security controls are poorly equipped to handle. Domain-based blocking and threat intelligence feeds cannot take down a canister on a decentralized, censorship-resistant blockchain network. Given the historical pattern of novel C2 techniques being rapidly adopted once publicly demonstrated — and the documented difficulty of disrupting decentralized blockchain-hosted infrastructure — security teams should anticipate that ICP canister dead-drop C2 will appear in campaigns beyond TeamPCP's operations, and should add ICP canister endpoints, Solana transaction-based dead drops, and similar decentralized infrastructure patterns to their threat model for CI/CD and npm dependency monitoring.

More broadly, the Trivy/CanisterWorm incident argues for treating security tooling with the same level of supply chain scrutiny applied to production dependencies. Software bills of materials (SBOMs) and SLSA provenance attestations for the security tools themselves — not just the software they scan — should become a standard procurement and operational requirement. Where a security tool cannot provide a verifiable provenance attestation for each release, organizations should treat updates as untrusted until independently verified.

## CSA Resource Alignment

This incident maps directly to several areas of established CSA guidance and framework coverage.

While MAESTRO was designed specifically for agentic AI systems, its treatment of supply chain integrity at the Agent Orchestration Layer (Tier 3) offers an instructive analogy: tools operating with elevated pipeline trust — whether AI agents or conventional scanners — can propagate compromise downstream through every workflow that depends on them [11]. The Trivy incident illustrates this dynamic precisely, with the scanner's privileged position enabling credential exfiltration that cascaded far beyond the immediate compromise.

The CSA AI Controls Matrix (AICM) addresses supply chain risk under its Software Development and Acquisition domain, with controls requiring provenance verification and integrity validation for third-party components used in AI and automation pipelines [12]. The pin-to-SHA guidance recommended above aligns directly with AICM control objectives for reducing unmanaged dependency risk.

CSA's Cloud Controls Matrix (CCM) v4.0 addresses these concerns under the Supply Chain Management and Transparency (STA) domain, particularly STA-09 (Third-Party Assessment Capability) and STA-10 (Supply Chain Governance Review), which call for continuous assessment of third-party

software components and automated validation of artifact integrity [13]. The use of network egress monitoring to detect the attack also aligns with CCM's Infrastructure and Virtualization Security (IVS) domain controls on network traffic monitoring.

CSA's Zero Trust guidance is relevant to the credential architecture exposed in this incident. Zero Trust principles call for eliminating implicit trust in any tool or component, including those operated by the security function itself. Applying least-privilege identity to CI/CD service accounts — ensuring the `aqua-bot` equivalent has only the narrowest permissions required and that those permissions expire automatically — would have limited the blast radius of the initial PAT theft [14].

The CSA Agentic AI Red Teaming Guide, which addresses adversarial testing of automated systems, provides a methodology applicable to CI/CD pipeline security that organizations should consult when evaluating their own pipeline trust models [15].

# References


[1] Aqua Security, "Trivy: Comprehensive Security Scanner," GitHub, https://github.com/aquasecurity/trivy, accessed March 21, 2026.

[2] Wiz Security Research, "Trivy Compromised by 'TeamPCP': Inside the Supply Chain Attack," Wiz Blog, March 20, 2026, https://www.wiz.io/blog/trivy-compromised-teampcp-supply-chain-attack.

[3] Charlie Eriksen, Aikido Security, "TeamPCP Deploys CanisterWorm on NPM Following Trivy Compromise," Aikido Security Blog, March 20, 2026, https://www.aikido.dev/blog/teampcp-deploys-worm-npm-trivy-compromise.

[4] Awesome Agents, "AI Agent Pwned Trivy via GitHub Actions Pwn Request," Awesome Agents News, February 28, 2026, https://awesomeagents.ai/news/hackerbot-claw-trivy-github-actions-compromise/.

[5] CrowdStrike, "From Scanner to Stealer: Inside the trivy-action Supply Chain Compromise," CrowdStrike Blog, March 2026, https://www.crowdstrike.com/en-us/blog/from-scanner-to-stealer-inside-the-trivy-action-supply-chain-compromise/.

[6] Rescana, "TeamPCP Worm Targets Docker, Kubernetes, Ray and Redis via React2Shell CVE-2025-55182," Rescana Threat Intelligence, 2026, https://www.rescana.com/post/teampcp-worm-targets-docker-kubernetes-ray-and-redis-via-react2shell-cve-2025-55182-to-build-crim.

[7] StepSecurity, "Trivy Compromised a Second Time — Malicious v0.69.4 Release," StepSecurity Blog, March 20, 2026, https://www.stepsecurity.io/blog/trivy-compromised-a-second-time---malicious-v0-69-4-release.

[8] ReversingLabs, "2026 Software Supply Chain Security Report," ReversingLabs, 2026, https://www.reversinglabs.com/sscs-report.

[9] The Hacker News, "GlassWorm Supply-Chain Attack Abuses 72 Open VSX Extensions," The Hacker News, March 2026, https://thehackernews.com/2026/03/glassworm-supply-chain-attack-abuses-72.html. [Note: URL was returning 404 at time of publication; statistics confirmed via SecurityWeek and Dark Reading coverage of the same event.]

[10] NIST National Vulnerability Database, "CVE-2025-58178: SonarQube Scanner GitHub Action Command Injection," September 2025, https://nvd.nist.gov/vuln/detail/CVE-2025-58178. [Command injection vulnerability (CWE-77/CWE-88), CVSS 7.8 High; exploitable in CI/CD contexts with potential for pipeline secret exposure.]


 

[11] Cloud Security Alliance, "MAESTRO: A Framework for AI Agent Threat Modeling," CSA AI Safety Initiative, 2025, https://cloudsecurityalliance.org/ai-safety-initiative.

[12] Cloud Security Alliance, "AI Controls Matrix (AICM)," CSA, 2025, https://cloudsecurityalliance.org/research/working-groups/ai-controls.

[13] Cloud Security Alliance, "Cloud Controls Matrix v4.0," CSA, 2021 (updated 2024), https://cloudsecurityalliance.org/research/cloud-controls-matrix.

[14] Cloud Security Alliance, "Zero Trust Advancement Center," CSA, https://cloudsecurityalliance.org/education/cczt.

[15] Cloud Security Alliance, "Agentic AI Red Teaming Guide," CSA AI Safety Initiative, 2025, https://cloudsecurityalliance.org/ai-safety-initiative.

 