CanisterWorm and the Blockchain Dead-Drop

Second Trivy Supply Chain Compromise in 30 Days

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-23

Key Takeaways

On March 19–21, 2026, threat actors identified as TeamPCP executed the second supply chain compromise of Trivy — Aqua Security's widely deployed open-source vulnerability scanner — within a single month, force-pushing malicious payloads to 75 of 76 version tags in the official `aquasecurity/trivy-action` GitHub Action and distributing a credential-stealing binary through the tool's official release channel [1][2]. The incident stands apart from routine supply chain attacks not only because of the recurrence against the same target but because of the novel command-and-control mechanism introduced by the follow-on worm: a smart contract on the Internet Computer Protocol (ICP) blockchain, a public decentralized network with no registrar to contact, no hosting provider to engage, and no conventional takedown mechanism — though an on-chain governance process did eventually disable the canister approximately 26 hours after its role in the attack was publicly disclosed [3].

The malware, dubbed CanisterWorm by Aikido Security researcher Charlie Eriksen — who characterized it as the first publicly documented npm worm and the first malware of any kind to use an ICP blockchain canister as a dead-drop resolver for its command-and-control infrastructure — used a smart contract address that can only be disabled through ICP's own on-chain governance process [3]. Rather than registering a traditional C2 domain subject to registrar suspension, DNS blocking, or hosting provider abuse complaints, CanisterWorm's operators embedded a canister address in every infected package. The governance process that ultimately took it offline took approximately 26 hours to act after public disclosure [3].

The compounded nature of these events — a first exploitation event spanning February 27–28 UTC, incomplete containment that left a retained credential in attacker hands, and a far more destructive second exploitation three weeks later — illustrates a structural failure mode relevant to any organization that relies on open-source security tooling in CI/CD pipelines: the tools that scan pipelines for vulnerabilities are themselves potential vectors for the attacks they are supposed to detect.

Background

Trivy and Its Role in Developer Pipelines

Trivy is Aqua Security's primary open-source vulnerability scanning tool and, by the breadth of its adoption in GitHub Actions workflows, one of the most widely deployed open-source scanners for container and cloud-native environments. Operating under the Apache 2.0 license, Trivy scans container images, Kubernetes configurations, infrastructure-as-code files, software bill of materials (SBOM) artifacts, and source code repositories for vulnerabilities, misconfigurations, and exposed secrets. Its low barrier to adoption — a single binary with no external database dependency — has made it a common choice for CI/CD pipeline security scanning, with the `aquasecurity/trivy-action` alone referenced in more than 10,000 workflow files on GitHub [2].

The `aquasecurity/trivy-action` GitHub Action, which wraps Trivy for use in GitHub Actions workflows, and the `aquasecurity/setup-trivy` action, which installs a specific version of the scanner into a workflow environment, both carry similarly broad adoption. This reach is precisely what made these Actions high-value targets: a single successful compromise of a version tag would silently redirect all downstream consumers of that tag — across thousands of organizations — to execute attacker-controlled code with the privileges of their CI/CD runner at their next workflow run [2].

The Internet Computer Protocol and Blockchain Canisters

The Internet Computer Protocol (ICP) is a public blockchain network built by the DFINITY Foundation, designed to host software and services directly on a decentralized network of independent data centers without reliance on commercial cloud infrastructure. ICP's execution primitives are called "canisters" — tamperproof computational units that bundle both code and persistent state and can serve HTTP traffic directly to external clients [3][4]. A canister is identified by a canonical address such as `tdtqy-oyaaa-aaaae-af2dq-cai` and is reachable at a stable URL in the form `https://[canister-id].raw.icp0.io/`.

The critical security property of ICP canisters — from an attacker's perspective — is their resistance to conventional takedown. Unlike a domain that can be seized by a registrar, an IP address that can be null-routed by a hosting provider, or a repository that can be taken down by a platform trust-and-safety team, external parties cannot unilaterally disable a canister absent the attacker's cooperation: a canister's designated controller (the principal who deployed it) can stop or delete it, but if the canister was deployed without an accessible controller, external intervention must go through the ICP Network Nervous System (NNS), an on-chain governance mechanism that processes proposals through a voting

process involving NNS stakeholders. When Aikido Security identified the CanisterWorm canister on March 20, ICP governance eventually took it offline for policy violation — but not until approximately 26 hours after its role in the attack was publicly disclosed [3].

Security Analysis

Incident One: The AI-Automated GitHub Actions Compromise (February 27–28, 2026)

The first Trivy compromise was executed by a GitHub account named `hackerbot-claw`, created on February 20, 2026 and reportedly attributed by security researchers to an autonomous AI agent [7][8]. The attacker exploited a well-known misconfiguration in Trivy's own CI/CD workflows: the `apidiff.yaml` workflow used the `pull_request_target` event trigger, which executes with base-repository secrets and full repository write permissions. The workflow additionally checked out code from the pull request's head branch — attacker-controlled fork code — and executed it with those elevated privileges, a combination that the GitHub security team has documented as a critical anti-pattern for public repositories [7].

At 00:18 UTC on February 27, the `hackerbot-claw` account opened and immediately closed pull request #10252 against the Trivy repository. The `pull_request_target` trigger fired, executed the attacker's forked code, and exfiltrated a Personal Access Token (PAT) to an attacker-controlled domain [7][8]. With the stolen PAT, the attacker privatized the Trivy repository, deleted all published GitHub Releases from v0.27.0 through v0.69.1, and pushed a suspicious artifact to the Trivy Visual Studio Code extension on the Open VSX marketplace [7]. Aqua Security rotated credentials and restored the repository, but the remediation was not atomic: at least one credential retained by the attacker survived the rotation, providing the persistent access that enabled the second, substantially more damaging campaign three weeks later.

Incident Two: The TeamPCP Campaign (March 19–21, 2026)

TeamPCP — a cloud-native threat actor also tracked under aliases including DeadCatx3, PCPcat, ShellForce, and CipherForce — initiated the second phase of exploitation on March 19, 2026, using credentials that survived the incomplete containment of the February incident [5]. The actor leveraged a

compromised `aqua-bot` service account to obtain write access to multiple Aqua Security repositories simultaneously, demonstrating that the retained credential had broad organizational scope rather than being scoped to a single repository [5].

The scope of the compromise was extensive. At 17:43 UTC on March 19, the attacker pushed a tag triggering automated release infrastructure to publish a trojaned v0.69.4 binary through Trivy's official release channel; the malicious binary was picked up automatically by Homebrew, propagating it to individual developers' machines without any explicit action on their part [5]. Docker images tagged `aquasec/trivy:0.69.5` and `0.69.6`, published on March 22, also contained hardcoded C2 infrastructure from the same campaign [2]. The attackers then force-pushed malicious commits to 75 of 76 version tags (v0.0.1 through v0.34.2) in the `aquasecurity/trivy-action` repository, and to 7 tags in `aquasecurity/setup-trivy` [2]. The attackers also renamed all 44 repositories in Aqua Security's `aquasec-com` GitHub organization with a `tpcp-docs-` prefix [5].

The malware embedded in these releases operated in three stages. During the collection phase, it ran the legitimate Trivy scanner in parallel with its malicious payload — a technique that allowed it to return plausible scan results while simultaneously harvesting SSH keys, AWS, GCP, and Azure credentials, Kubernetes service account tokens, Docker daemon configurations, Git credentials, cryptocurrency wallet files, and environment variables [2][5]. In GitHub Actions runner environments specifically, the malware read `/proc/*/environ` and accessed the `Runner.Worker` process memory directly through `/proc/<pid>/mem`, extracting secrets that GitHub's log-masking mechanism had marked as `isSecret: true` — bypassing log redaction entirely [2]. During the encryption phase, the collected data was packaged as `tpcp.tar.gz` using AES-256-CBC with RSA-4096 hybrid encryption. Exfiltration proceeded to a typosquatted domain `scan.aquasecurtiy[.]org`, with a fallback path that created a public `tpcp-docs` repository in the victim's own GitHub account and pushed the encrypted bundle there, exploiting the victim's own trusted infrastructure as an exfiltration channel [5].

On non-CI developer machines, the malware additionally dropped a Python backdoor at `~/.config/systemd/user/sysmon.py` that polled an ICP blockchain canister for follow-on instructions — the same canister infrastructure subsequently used by CanisterWorm [3]. StepSecurity's Harden-Runner product detected the anomalous outbound connections to the C2 domain in real time during the exposure window. Analysis of 767 prominent public repositories indicated that 45 had compromised workflow runs during the active window, with 5 repositories having direct exposure to custom secrets [5]. The GitHub advisory assigned to this compound incident is GHSA-69fq-xp46-6x23 [8].

CanisterWorm: A Self-Propagating npm Worm with Blockchain C2

Within 24 hours of the Trivy GitHub Actions compromise, threat researchers identified a follow-on campaign that used npm authentication tokens stolen during the CI/CD exploitation to publish malicious versions of npm packages under compromised publisher accounts [3]. Aikido Security researcher Charlie Eriksen, who identified and named the worm on approximately March 20 at 20:45 UTC, characterized the ICP canister C2 technique as the attack's defining innovation: because no hosting provider can be asked to take it down through conventional abuse procedures, the C2 infrastructure can substantially outlast both victim incident response and platform security teams' standard procedures [3].

CanisterWorm's payload architecture comprised three interdependent layers. The outermost layer, a Node.js `postinstall` hook, executed immediately upon `npm install`, with no user interaction required beyond the installation itself. This hook installed a persistent Python backdoor configured as a `systemd --user` service named `pgmon` — chosen for its resemblance to PostgreSQL monitoring tooling — which polled the ICP canister at `https://tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0.io/` approximately every 3,000 seconds (50 minutes) for a URL pointing to the current follow-on payload [3]. A 5-minute initial sleep was included, consistent with known sandbox evasion techniques: automated analysis sandboxes that execute postinstall hooks with time limits would typically timeout before the malicious network call was made [3]. The ICP canister itself exposed three methods: `get_latest_link`, which returned the current payload URL; `http_request`, which served the URL to the backdoor; and `update_link`, which allowed the operator to rotate to a new payload at any time without republishing or modifying any infected package [3]. When the canister returned a YouTube URL — which it did after ICP governance began its review — the backdoor's logic interpreted this as a kill-switch signal and skipped execution of the follow-on binary.

The worm's self-propagation mechanism, introduced in a second wave of packages deployed approximately 30 minutes after the initial wave, transformed the attack's blast radius qualitatively. The evolved variant embedded a `findNpmTokens()` function in each package's `index.js` that scanned the host developer's home directory for `.npmrc` files, checked `/etc/npmrc` for system-wide npm credentials, and inspected environment variables for `NPM_TOKEN` or `NPM_TOKENS` [3]. Harvested tokens were passed to a detached `deploy.js` process that enumerated all packages the credential holder could publish to, automatically bumped their patch version numbers, and republished malicious versions under `--tag latest` [3]. Eriksen's characterization of this transition is precise: the attack escalated from a compromised account distributing malware to malware that autonomously compromises additional accounts and re-publishes itself, with every developer or CI pipeline possessing an accessible npm token becoming an unwitting propagation node.

The affected packages spanned multiple organizational namespaces. The initial compromise included 28 packages in the `@EmilGroup` scope and 16 packages in the `@opengov` scope, with individual packages including `@teale.io/eslint-config`, `@airtm/uuid-base32`, and `@pypestream/floating-ui-dom` [1]. By March 21, reported counts had expanded to 135 malicious artifacts across 64 or more unique packages, and subsequently to 141 artifacts across 66 or more packages [3][9]. GitLab assigned advisory identifiers in the GMS-2026-2 through GMS-2026-62 series to individual affected packages [9].

The Blockchain C2 Paradigm: Structural Implications

Researchers have characterized the use of an ICP canister as a dead-drop resolver as a meaningful evolution in C2 resilience rather than merely a novel implementation choice [3]. Traditional C2 infrastructure resilience is bounded by the speed at which defenders can act: domain registrars can suspend registrations in hours, hosting providers can respond to abuse complaints within a day, and BGP routing changes can null-route an IP address in minutes. Fast-flux DNS and domain generation algorithms (DGAs) raise the cost of C2 disruption, but all such techniques ultimately depend on infrastructure controlled by an intermediary entity that can be compelled to act. Blockchain-based dead-drop resolvers substantially raise the cost and delay of disruption: absent the attacker's own cooperation, the only available mechanism for an external party to act is the underlying blockchain's on-chain governance process, with no centralized intermediary capable of being compelled [3][4].

The 26-hour window between public disclosure and ICP governance action in this incident is instructive. The ICP Network Nervous System does not have an emergency disable mechanism for rapid response; governance actions require proposal submission, voting, and execution — a deliberative process that, whatever its design rationale, prioritizes consensus over response speed in practice. This gap between attacker deployment speed and governance response time creates a structural window during which blockchain-hosted C2 infrastructure may be effectively durable from a defender's perspective [3].

Recommendations

Immediate Actions

Organizations that use `aquasecurity/trivy-action` or `aquasecurity/setup-trivy` in GitHub Actions workflows should audit their pipeline configurations for affected version tags and rotate any secrets that were accessible to workflow runs during the exposure windows of March 19–21, 2026 [2]

[5][8]. Any GitHub Actions workflow that uses a floating version tag (for example, `@v0.18.0`) rather than a commit SHA is vulnerable to the same tag-replacement technique TeamPCP used; pinning Action references to full commit SHA values closes the tag-replacement attack vector used in this incident [2]. Organizations should also audit their npm token scopes: tokens with publish access to all packages a user can publish to are disproportionately dangerous in this threat model, and scoping npm automation tokens to the minimum required package set would limit the blast radius of future stolen-token exploitation [3].

Developer environments on machines where the malicious Trivy binary or affected npm packages were installed should be treated as fully compromised pending credential rotation. The malware's collection phase targeted SSH keys, cloud provider credentials, Kubernetes service account tokens, and Docker daemon configurations — a scope broad enough that incomplete rotation may leave active attacker footholds in cloud infrastructure, container registries, or version control systems [5].

Short-Term Mitigations

Pipeline security tooling such as StepSecurity's Harden-Runner, which detected the TeamPCP campaign's anomalous outbound connections in real time, provides a detection layer that complements static policy enforcement [5]. Deploying network egress monitoring for GitHub Actions runners — either through purpose-built runner hardening tools or by routing runner traffic through an enterprise proxy with anomaly detection — would have reduced the exposure window in this incident and would surface comparable future attacks. Security teams should also configure npm token monitoring: platforms such as npm, GitHub, and cloud providers offer token usage logs that can surface anomalous publishing activity from legitimate credentials.

Organizations that depend on Homebrew, container registry mirror caches, or package manager proxies that automatically promote new releases should review their auto-update policies. The malicious Trivy v0.69.4 release propagated to developer machines via Homebrew's automatic update mechanism without explicit user action [5]. Mirroring policies that require human or automated review before promoting new upstream versions would reduce the blast radius of future incidents of this type.

Strategic Considerations

The two Trivy compromises within a single month expose a systemic risk that extends well beyond Aqua Security's specific tooling: security scanning software deployed in CI/CD pipelines inherits the full scope of secrets available to the runner, and its elevated trust position in that environment makes it a high-value supply chain target. Organizations should apply the same security rigor to their security tooling dependencies — vulnerability scanners, SAST tools, secret detectors, policy engines — that they apply to

their application dependencies. This includes verifying artifact integrity through sigstore attestation or similar supply chain provenance mechanisms, not solely relying on the implicit trust conveyed by a tool's GitHub tag or official release channel [2].

The blockchain C2 paradigm demonstrated by CanisterWorm requires a corresponding evolution in network security controls. Because ICP canisters serve traffic over standard HTTPS through well-known stable URL patterns at `icp0.io`, organizations cannot rely on domain-reputation feeds or IP reputation lists to block this C2 communication path without also blocking legitimate use of the ICP network. Behavioral egress analysis — flagging unusual network endpoints reached by background processes, particularly those exhibiting periodic polling behavior at fixed intervals — is better suited to detecting this class of C2 infrastructure than domain-reputation or IP-reputation feeds alone [3].

CSA Resource Alignment

The CanisterWorm and TeamPCP incidents map directly to multiple CSA frameworks and should be read in conjunction with existing CSA guidance across the supply chain and cloud-native security domains.

CSA AI Cloud Matrix (AICM) and Cloud Controls Matrix (CCM): The TeamPCP credential exfiltration campaign — specifically its collection of cloud provider access keys, Kubernetes service account tokens, and Docker daemon credentials — directly implicates AICM Pillar 3 (Data, Integrity, and Privacy) and CCM's Supply Chain Management (STA) domain. The CI/CD pipeline environment represents a boundary where cloud infrastructure credentials are concentrated and frequently accessible to automation with broad permissions, but where the same identity and access management rigor applied to production infrastructure is often absent [10][11].

MAESTRO Threat Modeling Framework: The first Trivy compromise, attributed to the `hackerbot-claw` AI agent that exploited a `pull_request_target` misconfiguration, represents a MAESTRO Layer 1 (Model Layer) threat: an autonomous AI agent used as an attack tool to execute a specific exploitation technique — an open pull request followed by immediate closure — that triggered a vulnerable workflow. This incident may represent an early example of a broader threat class in which AI agents are used for offensive supply chain exploitation. As AI agent capabilities and autonomy increase, security teams should anticipate that techniques like the `pull_request_target` exploit demonstrated here could be applied at greater scale and speed than human-paced manual campaigns [12].

CSA Guidance on Zero Trust Architecture: The lateral movement path from the retained credential to broad organizational access across 44 repositories is precisely the blast radius that Zero Trust architecture and just-in-time credential issuance are designed to constrain. The Aqua Security incident is a concrete case study for why persistent service account tokens with organizational-scope write access represent structural risk that Zero Trust controls address at the credential-management layer [13].

STAR and Supply Chain Assurance: CSA's Security Trust Assurance and Risk (STAR) registry and related guidance on third-party risk management should incorporate criteria for evaluating the security posture of open-source security tooling specifically — a category where the implicit trust model (widely deployed, security-branded software used in trusted pipeline positions) does not match the actual security investment and incident response capacity of most open-source projects.

References

[1] The Hacker News, "Trivy Supply Chain Attack Triggers Self-Spreading CanisterWorm Across 47 npm Packages," March 21, 2026. https://thehackernews.com/2026/03/trivy-supply-chain-attack-triggers-self.html

[2] The Hacker News, "Trivy Security Scanner GitHub Actions Breached, 75 Tags Hijacked to Steal CI/CD Secrets," March 20, 2026. https://thehackernews.com/2026/03/trivy-security-scanner-github-actions.html

[3] Aikido Security / Charlie Eriksen, "TeamPCP Deploys CanisterWorm on NPM Following Trivy Compromise," March 20–21, 2026. https://www.aikido.dev/blog/teampcp-deploys-worm-npm-trivy-compromise

[4] DFINITY Foundation, "A Closer Look at Software Canisters: An Evolution of Smart Contracts on the Internet Computer," Medium, 2021. https://medium.com/dfinity/software-canisters-an-evolution-of-smart-contracts-internet-computer-f1f92f1bfffb

[5] Wiz Research / Rami McCarthy, "Trivy Compromised: Everything You Need to Know About the Latest Supply Chain Attack," March 20, 2026. https://www.wiz.io/blog/trivy-compromised-teampcp-supply-chain-attack

[6] BleepingComputer / Lawrence Abrams, "Trivy Vulnerability Scanner Breach Pushed Infostealer via GitHub Actions," March 21, 2026. https://www.bleepingcomputer.com/news/security/trivy-vulnerability-scanner-breach-pushed-infostealer-via-github-actions/

[7] Awesome Agents, "An AI Agent Just Pwned Trivy's 32K-Star Repo via GitHub Actions," March 2026. https://awesomeagents.ai/news/hackerbot-claw-trivy-github-actions-compromise/

[8] GitHub Security Advisory, "Trivy Ecosystem Supply Chain Temporarily Compromised," GHSA-69fq-xp46-6x23, March 2026. https://github.com/aquasecurity/trivy/security/advisories/GHSA-69fq-xp46-6x23

[9] GitLab Security Advisories, GMS-2026-2 through GMS-2026-62 series (CanisterWorm-affected npm packages), March 2026. https://advisories.gitlab.com/pkg/npm/@emilgroup/account-sdk/GMS-2026-2/

[10] Cloud Security Alliance, "AI Cloud Matrix (AICM)," Cloud Security Alliance Publications, 2025–2026. https://cloudsecurityalliance.org/research/aicm (URL as of publication; see CSA website for current location)

[11] Cloud Security Alliance, "Cloud Controls Matrix v4.0," Cloud Security Alliance Publications, 2021. https://cloudsecurityalliance.org/research/cloud-controls-matrix

[12] Cloud Security Alliance AI Safety Initiative, "MAESTRO: A Threat Modeling Framework for AI Agentic Systems," Cloud Security Alliance Publications, 2025. https://cloudsecurityalliance.org/research/maestro (URL as of publication; see CSA website for current location)

[13] Cloud Security Alliance, "Software Defined Perimeter and Zero Trust," Cloud Security Alliance Publications, 2022. https://cloudsecurityalliance.org/research/sdp-zero-trust (URL as of publication; see CSA website for current location)

