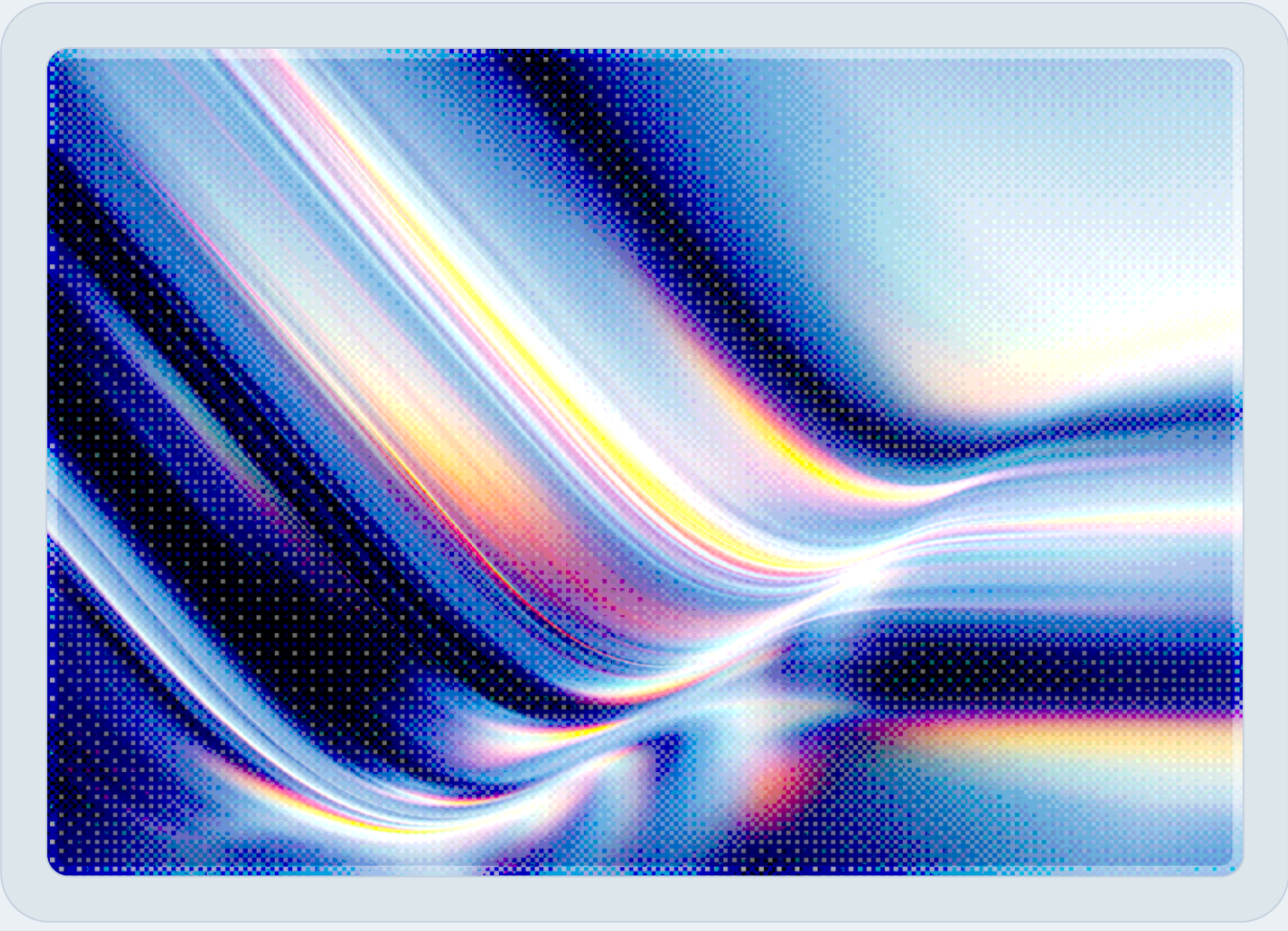
CSAI Foundation | Cloud Security Alliance

Megalodon: Mass CI/CD Pipeline Poisoning via GitHub Actions

How a Six-Hour Automated Campaign Backdoored 5,500+ Repositories

2026-05-24

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- On May 18, 2026, an automated campaign dubbed Megalodon pushed 5,718 malicious commits to 5,561 distinct GitHub repositories in under six hours — a scale that, per StepSecurity's reporting, exceeded previous single-day pipeline poisoning incidents in repository count. [1][2][9][10]
- The attack exploited throwaway or compromised GitHub accounts to inject malicious GitHub Actions workflow files containing base64-encoded payloads capable of harvesting over 30 categories of secrets, including AWS access keys, GCP OAuth tokens, Azure instance role credentials, SSH private keys, OIDC tokens, and Kubernetes configurations. [2]
- Two variants were deployed: a mass "SysDiag" workflow triggered on every push and pull request for broad secret capture, and a targeted "Optimize-Build" backdoor using `workflow_dispatch` triggers that the attacker could activate on demand via the GitHub API. [3]
- At least one npm package — @tiledesk/tiledesk-server versions 2.18.6 through 2.18.12 — was unknowingly published from a poisoned repository, propagating the backdoor to downstream npm consumers without any compromise of the npm account itself. [1][2]
- Megalodon succeeded at scale because thousands of targeted repositories lacked branch protection on `.github/workflows/`, granted broad write permissions to external contributors, and stored production secrets in over-privileged CI runner environments. [1][2][3]
- Organizations can substantially reduce exposure to this attack class through three durable controls: pinning all GitHub Actions references to immutable commit SHAs, enforcing minimum token permissions per workflow job, and replacing long-lived CI secrets with OIDC-based workload identity. Together these controls eliminate the most common exploitation paths and limit the value of any credentials captured from a compromised runner.

Background

The Expanding Attack Surface of CI/CD Pipelines

Modern software development is structurally dependent on CI/CD pipelines. The same pipeline that builds application code also holds keys to cloud infrastructure, npm publishing credentials, container registry tokens, database connection strings, and signing certificates. A successful attacker who can execute arbitrary code inside a CI runner gains access to all of these simultaneously — a single point of leverage with a blast radius that can extend from source repositories into production environments and published package registries.

GitHub Actions has become the most widely adopted hosted CI/CD platform for open source projects. Its deep integration with the GitHub repository model — where workflow files live inside the repository itself under `.github/workflows/` — creates a powerful development convenience but also a fundamental security property that is frequently misunderstood: any actor with write access to a repository, including collaborators, can modify workflow definitions. This is categorically different from an external trigger modifying application code, because workflow modifications instruct the CI system itself what to do. Changing a workflow file is changing a build instruction, not an application file, and the distinction is consequential: workflow modifications execute in the CI system's security context, not the application's, and are therefore not reviewed by the same code-review gates that protect application changes.

The exploitation model that Megalodon used — Direct Poisoned Pipeline Execution (d-PPE) — has been recognized by security researchers for several years. [4] In d-PPE, an adversary with repository write access modifies workflow definition files to cause the CI system to execute attacker-controlled commands the next time the pipeline runs. No vulnerability in GitHub's platform is exploited; the attacker is operating within the intended permission model. The attack is made possible by over-permissioned access controls and the absence of code-review requirements on workflow file changes specifically.

A Pattern of Escalating Incidents

Megalodon did not emerge in isolation. The GitHub Actions ecosystem has faced a series of escalating supply chain incidents over the preceding 14 months that established both the attack technique and the defender response gap. In March 2025, the widely used `tj-actions/changed-files` action was compromised after attackers hijacked its mutable version tags, affecting an estimated 23,000 downstream repositories that had been pointing to tags rather than immutable commit SHAs. [4] In March 2026, attackers hijacked version tags in `aquasecurity/trivy-action`, a popular

container security scanner, redirecting downstream workflows to attacker-controlled code and harvesting cloud credentials from affected pipeline runs. In April 2026, the `prt-scan` campaign exploited the `pull_request_target` trigger — which runs with write access even when triggered by external pull requests — to steal secrets from repositories over a six-week window.

Each incident was documented, analyzed, and followed by security guidance. Yet Megalodon's scale on May 18, 2026, provided stark evidence that adoption of recommended controls across the open source ecosystem had not kept pace with the threat. The conditions that enabled the earlier incidents — mutable action version references, weak branch protections, and over-privileged CI tokens — remained pervasive.

Security Analysis

How Megalodon Operated

The campaign was active from approximately 11:36 UTC to 17:48 UTC on May 18, 2026, a window of roughly six hours and twelve minutes. [1] SafeDep's Malysis supply chain security engine first detected the campaign after identifying a base64-encoded payload inside a bundled workflow file in the npm package `@tiledesk/tiledesk-server@2.18.12`, triggering a broader retrospective analysis that surfaced the full scope of the operation. [2]

The attackers used throwaway GitHub accounts with randomized eight-character usernames and configured their git clients to forge author identities. Commit author fields were set to plausible automation personas — `build-bot`, `auto-ci`, `ci-bot`, and `pipeline-bot` — with email addresses `build-system@noreply.dev` and `ci-bot@automated.dev`, deliberately chosen to resemble routine CI tooling activity. [3] Commit messages were similarly designed to avoid human scrutiny: `ci: add build optimization step` and `chore: optimize pipeline runtime` follow widely used conventional commit conventions and would blend seamlessly into a project's commit history for any maintainer performing a casual review. [3]

Access to the targeted repositories appears to have been obtained through two routes: compromised personal access tokens (PATs) or deploy keys, most likely acquired by infostealer malware from developer machines prior to the campaign. [5] Security researchers noted that the email address `build-system@noreply.dev` was associated with a recently compromised host in infostealer

telemetry, consistent with a workflow in which credential harvesting by a commodity infostealer provided the initial access that enabled the mass injection campaign. [5] This infostealer-to-pipeline-poisoning kill chain is becoming a recognized escalation pattern in software supply chain attacks.

The Two Payload Variants

StepSecurity and SafeDep's analyses identified two distinct workflow variants deployed by the campaign. [1][2] The first, named `SysDiag` in the injected workflow's `name:` field, created a new workflow file with triggers on every push event and every pull request. This variant was designed for breadth: any repository activity would cause the runner to execute the malicious payload, maximizing the probability that the backdoor would fire and exfiltrate secrets before the injection was detected and reverted. The SysDiag variant was used against the majority of the 5,561 targeted repositories. [1][9][10]

The second variant, named `Optimize-Build`, was more selectively targeted. Rather than adding a new workflow, it replaced or modified existing workflow definitions to use the `workflow_dispatch` trigger only. [3] This trigger does not fire automatically on repository events; it fires only when explicitly invoked via the GitHub API or through the GitHub Actions UI. The implication is that Optimize-Build installed dormant backdoors that the attacker could activate selectively on demand — potentially against high-value targets where an automated immediate trigger would be too noisy — while leaving the repository's normal CI behavior unchanged and thus delaying detection. This variant was found embedded in the `@tiledesk/tiledesk-server` npm package.

Payload Capabilities and the Exfiltration Scope

The base64-encoded bash payloads in both variants performed comprehensive secrets sweeps across the CI runner environment. [2][3] The malware queried the AWS Instance Metadata Service v2 endpoint to retrieve EC2 instance role credentials, queried the Google Cloud metadata server for OAuth access tokens, and queried the Azure Instance Metadata Service for managed identity credentials — a multi-cloud sweep designed to succeed regardless of which cloud platform hosted the affected CI runner. Beyond cloud credentials, the payload collected environment variables, SSH private keys, Docker configuration files (`~/.docker/config.json`), Kubernetes configuration files (`~/.kube/config`), HashiCorp Vault tokens, Terraform state credentials, and GitHub Actions OIDC tokens. A pattern-matching scan of the source code checkout covered more than 30 credential regex formats, targeting API keys for major SaaS providers, database connection strings, PEM-encoded private keys, and JSON Web Tokens. [3]

Exfiltrated data was transmitted to a command-and-control server at `216.126.225.129:8443`, with the string `megalodon` included as a query parameter — a detail that allowed StepSecurity's analysis to correlate traffic across unrelated repositories and establish the campaign's full scope. [1]

From Repository to Registry: The Tiledesk Propagation Path

The `@tiledesk/tiledesk-server` case illustrates why CI/CD poisoning attacks carry consequences that extend beyond the immediate repository. Between May 19 and May 21, 2026, the Tiledesk maintainer published seven package releases — apparently without detecting that the repository had been backdoored — producing versions 2.18.6 through 2.18.12 from the compromised codebase. [1][2] Because the backdoor lived in a CI workflow file, not in application source code that a diligent maintainer might spot in a code review, the malicious payload propagated to the npm registry as bundled CI artifacts inside the published package tarball. Downstream consumers who installed any of these versions received a package that, when used in their own CI/CD pipelines, would execute the malicious workflow and exfiltrate their secrets. [11]

This propagation path — where the attacker never directly touches the package registry account, instead relying on the legitimate maintainer to unknowingly publish from poisoned source — has significant implications for traditional supply chain security controls. Code signing and two-factor authentication on registry accounts, while valuable, would not have prevented this attack. The integrity guarantee that matters here is over the source-to-publication pipeline, not the publication event itself.

Attribution Context

SafeDep and multiple security researchers linked Megalodon to the broader TeamPCP threat actor cluster, a group responsible for a series of escalating GitHub supply chain campaigns in 2026. [5][6] However, researchers cautioned that the attribution is tentative. The assessment from SafeDep's threat hunters was that Megalodon "is most likely a different threat actor copying TeamPCP's behavior and style, but not much of the code itself" — suggesting convergent adoption of an established technique by a distinct operator rather than a direct TeamPCP operation. [5] TeamPCP's confirmed 2026 activity includes a Wave Four campaign involving a compromised Nx Console VS Code extension and a self-propagating worm targeting the `durabletask` PyPI package that reportedly exfiltrated data from approximately 3,800 internal GitHub repositories. [6] Regardless of attribution, the Megalodon campaign suggests that d-PPE techniques developed and refined by one threat actor group are being independently adopted by others, which — if confirmed — would meaningfully broaden the threat landscape for CI/CD pipeline defenders.

Recommendations

Immediate Actions

A high-priority action for any organization using GitHub Actions is to audit every workflow file for mutable action references — version tags such as `@v3` or `@main` — and replace them with fully qualified, immutable commit SHA references. A reference like `uses: actions/checkout@11bd71901bbe5b1630ceea73d27597364c9af683 # v4.2.2` ties the workflow to a specific, unalterable point in the action's history. [7][8] An attacker who compromises the version tag cannot redirect a pinned workflow. GitHub's administrative policy now supports enforcing SHA pinning at the organization level, blocking any workflow run that references an action by tag rather than by commit hash. [8] Organizations should enable this policy and use automated tooling — such as StepSecurity's Harden-Runner or Dependabot's GitHub Actions support — to maintain SHA currency as upstream actions release new versions.

Second, every workflow job should declare explicit, minimum-scope token permissions using the `permissions:` key. [7] The default `GITHUB_TOKEN` in many older workflows carries read/write access across the repository scope, which is far broader than most individual jobs require. A job that only reads code for testing has no legitimate need for `packages: write` or `contents: write`. Scoping permissions to the minimum required eliminates the value of any secrets captured from that job's runner environment.

Third, organizations should immediately audit the branch protection rules on their default and release branches, with particular attention to the `.github/workflows/` path. In repositories where workflow file changes do not require code review — the configuration that allowed Megalodon to succeed at scale — any contributor with push access can inject malicious CI behavior without oversight. Enabling required pull request reviews for workflow file modifications, combined with the CODEOWNERS mechanism to route those reviews to designated security-aware maintainers, closes this vector. [7]

For any repository that uses secrets to authenticate CI runners to cloud platforms, organizations should assess whether those secrets can be replaced with OIDC-based workload identity. GitHub Actions natively emits OIDC tokens that all three major cloud platforms accept for credential exchange without storing any static secret in the repository or CI environment. [7] For workloads already running on GitHub Actions, the migration to OIDC-based authentication is typically a configuration exercise requiring hours rather than weeks of engineering effort. Repositories that have completed this migration eliminate the risk of long-lived static secrets being stolen from repository settings or CI environment variables — a

significant reduction in persistent exposure. However, a compromised workflow that executes attacker-controlled code can still exfiltrate the short-lived credentials exchanged during that job run, so OIDC migration should be paired with runtime behavior monitoring and minimum-scope cloud IAM policies.

Short-Term Mitigations

Within the next 30 days, security teams should implement supply chain scanning as a required gate in their CI pipelines. Tools such as StepSecurity's Harden-Runner analyze runtime behavior of GitHub Actions workflows and can detect anomalous outbound network connections — including exfiltration to novel IP addresses — and unexpected secret access patterns. [1] SafeDep's Malysis engine identified the Megalodon campaign through behavioral analysis of bundled workflow artifacts in npm packages, demonstrating that scanning at the package registry level can catch attacks that bypass source code review. [2] Organizations that publish packages to npm, PyPI, or other registries should incorporate workflow artifact scanning as part of their release pipeline.

Repository administrators should review their collaborator lists and rotate any personal access tokens or deploy keys associated with accounts that cannot be verified as currently under the control of their legitimate owner. Given that Megalodon's initial access appears to have involved infostealer-compromised developer credentials, treating any PAT or deploy key as potentially compromised and rotating it on a defined schedule — with an accelerated rotation following any indication of developer endpoint compromise — reduces the exposure window for subsequent campaigns.

Supply chain scanning of the npm, PyPI, and other package registries for the Megalodon C2 indicator (`216.126.225.129`) and the known workflow file names (`SysDiag`, `Optimize-Build`) can identify packages that may have been published from poisoned repositories during the campaign window. Organizations that consume open source packages published between May 18–22, 2026 from repositories without branch protection on workflow files should assess whether any of those packages were among the 5,561 affected repositories. [1][11]

Strategic Considerations

Megalodon's scale was enabled by two systemic conditions: the widespread absence of workflow file protection controls across public repositories, and the use of long-lived CI secrets that make every runner a high-value exfiltration target. Both conditions are addressable through organizational policy rather than new technology. Enterprises that have adopted a secure-by-default CI/CD posture — with SHA-pinned actions, workflow-scoped minimum permissions, OIDC-based cloud authentication, and mandatory review for workflow file changes — were not meaningfully exposed to this campaign.

Security teams should evaluate GitHub Actions in the same threat model they apply to production workload infrastructure. CI runners execute arbitrary code with access to the organization's most sensitive credentials, yet many organizations apply substantially weaker access controls to their pipeline configuration than they apply to their production systems. Treating pipeline-as-code with the same rigor as application code — version control, mandatory code review, least-privilege access, and continuous behavioral monitoring — is the long-term structural response to this class of attack.

For organizations building or operating AI systems, the CI/CD pipeline merits special attention as an attack surface. Training pipelines, model serving deployments, and AI inference services frequently require access to cloud storage containing model weights, GPU cluster credentials, ML platform API keys, and data lake configurations — a concentration of sensitive access that makes AI-related pipelines high-value targets for actors seeking either monetizable cloud compute access or intellectual property. The controls described here apply with equal or greater force to AI and ML pipeline infrastructure.

CSA Resource Alignment

MAESTRO Framework

Megalodon maps directly to Layer 0 (Infrastructure and Deployment) of CSA's MAESTRO agentic AI threat model. The CI/CD pipeline occupies the infrastructure layer of AI development environments: it provisions, builds, and deploys AI workloads, and its compromise can propagate upward through every layer above it. MAESTRO's threat modeling for agentic infrastructure addresses scenarios in which an attacker who compromises the execution environment of an agent — rather than the agent itself — gains equivalent or superior access to the target systems that agent is authorized to reach. Megalodon instantiates this model precisely: by compromising the CI runner environment, the attacker bypassed all application-level security controls to access the secrets those controls were designed to protect.

AI Controls Matrix (AICM)

CSA's AI Controls Matrix (AICM), a superset of the Cloud Controls Matrix (CCM), addresses supply chain integrity requirements that are directly implicated by Megalodon. [12][13] The AICM's controls on software supply chain risk management call for cryptographic verification of build artifacts, integrity monitoring of CI/CD pipeline definitions, and access control governance for pipeline modification rights — the three primary control categories whose absence was exploited by this campaign. Organizations using the AICM as a control framework should prioritize the pipeline integrity controls for CI environments that build or deploy AI workloads.

CCM and STAR

The Cloud Controls Matrix addresses the foundational controls relevant to this attack across several domains. [13] The Application and Interface Security (AIS) domain covers secure development lifecycle practices including code review requirements and build pipeline integrity. The Identity and Access Management (IAM) domain's controls on least-privilege access and credential lifecycle management apply directly to CI runner token permissions and the use of long-lived secrets in pipeline environments. The Supply Chain Management, Transparency, and Accountability (STA) domain addresses third-party code dependency risk, including the use of external actions in CI pipelines.

Organizations seeking to document their CI/CD pipeline security posture against these controls can publish their assessment through CSA STAR, providing a structured mechanism for communicating supply chain security practices to customers and partners.

Zero Trust Guidance

CSA's Zero Trust guidance is directly applicable to CI/CD pipeline architecture. [14] Zero Trust's principle of "never trust, always verify" translates to the CI/CD context as: never grant standing access to production secrets based on repository membership alone; always verify the workload's identity at the moment of credential issuance through OIDC or equivalent cryptographic attestation; and always scope access to the minimum required for the specific operation being performed. A CI/CD architecture designed around these principles would not have provided the conditions Megalodon exploited — a runner environment with standing access to production credentials accessible to any workflow file modification by any collaborator.

References

[1] StepSecurity. "Megalodon: Mass GitHub Actions Secret Exfiltration Across 5,500+ Public Repositories." StepSecurity Blog, May 2026.

[2] SafeDep. "Megalodon: Mass GitHub Repo Backdooring via CI Workflows." SafeDep Real-time Open Source Software Supply Chain Security, May 2026.

[3] Phoenix Security. "MEGALODON_CI: Automated GitHub Actions Workflow Poisoning and CI/CD Credential Harvesting at Scale." Phoenix Security, May 2026.

[4] Palo Alto Unit 42. "GitHub Actions Supply Chain Attack: A Targeted Attack on Coinbase Expanded to the Widespread tj-actions/changed-files Incident." Palo Alto Networks Unit 42 Threat Assessment, April 2025.

[5] InfoStealers. "Infostealers Just Spawned a 5,000+ Repo GitHub Supply Chain Attack." InfoStealers.com, May 2026.

[6] Phoenix Security. "TeamPCP Wave Four: GitHub Breach via Poisoned VS Code Extension, durabletask PyPI Worm, and ~4,000 Internal Repositories Exfiltrated." Phoenix Security, 2026.

[7] GitHub. "Security hardening for GitHub Actions." GitHub Docs, 2026.

[8] GitHub. "GitHub Actions policy now supports blocking and SHA pinning actions." GitHub Changelog, August 2025.

[9] The Hacker News. "Megalodon GitHub Attack Targets 5,561 Repos with Malicious CI/CD Workflows." The Hacker News, May 2026.

[10] The Register. "Megalodon chums the waters in 5.5K+ GitHub repo poisonings." The Register, May 22, 2026.

[11] OX Security. "Megalodon: New CI/CD Malware Spreads Across GitHub, Infecting ~5,000+ Repositories." OX Security Blog, May 2026.

[12] Cloud Security Alliance. "AI Controls Matrix (AICM)." Cloud Security Alliance, 2026.

[13] Cloud Security Alliance. "Cloud Controls Matrix v4.1 (CCM)." Cloud Security Alliance, 2024.

[14] Cloud Security Alliance. "Zero Trust Working Group." Cloud Security Alliance, 2026.

