

KICS GitHub Action Compromised: TeamPCP Strikes Again in Supply Chain Attack

By Rami McCarthy, James Haughom, Benjamin Read

Published: 2026-03-23 · Archived: 2026-08-06 02:01:53 UTC

The [KICS](#) GitHub Action was compromised with credential-stealing malware by TeamPCP, the same group [behind the Trivy attack](#). KICS is an open source infrastructure as code security scanner by Checkmarx. **Between 12:58 and 16:50 UTC on March 23rd**, any users of this GitHub Action who were pinning to one of the compromised tags would have been served the malware. The repository was taken down at 16:50 UTC, shortly [after a GitHub issue was filed by a user](#) notifying the maintainers of the incident.

The action was available at <https://github.com/Checkmarx/kics-github-action> prior to takedown.

Update 03/24:

11:30 UTC: The "litellm" packages (versions 1.82.7 and 1.82.8) on PyPI have been trojanized. They contain with the same functionality as the previous operation, but using a new exfiltration domain: models.litellm[.]cloud. The malicious update was published at approximately 8:30 UTC and was been quarantined by PyPI at 11:25 UTC. Wiz customers can see an advisory in the [Threat Center](#).

Updates 03/23:

19:24 UTC: The repository has been reinstated, and the maintainers state "[The issue is resolved now](#)."

22:25 UTC: Sysdig reports that [ast-github-action](#) was also impacted. They were limited to observing a single malicious tag `2.3.28` - however based on TeamPCPs tactics, we believe it is likely all tags were impacted.

22:35 UTC: Based on a tip from independent researcher Adnan Khan, Wiz has confirmed that Checkmarx OpenVSX extensions [cx-dev-assist](#) 1.7.0 and [ast-results](#) 2.53.0 **have been compromised**. This was [concurrently reported by ReversingLabs via tweet](#). See "OpenVSX Payload" section below for details. We have reported these to OpenVSX for removal.

Update 03/24 9:00 UTC: Checkmarx have published [a Security Update](#) addressing the issues with the KICS GitHub action and OpenVSX plugins. They state a resolution time of **15:41 UTC for OpenVSX**, however we observed the malicious versions were present at the time of our report. Additionally, while new versions have been pushed, the malicious versions have yet to be removed.

This is the second popular open source security scanner that this group has compromised in the last five days. The operation uses familiar naming conventions and the same RSA public key, allowing Wiz to assess with high confidence that it is the same actor.

KICS Github Action Payload

The malicious code was injected in the same manner as the Trivy incident:

1. The attacker staged imposter commits (commits on a fork of the repository) containing their payload: `setup.sh` and modified the `action.yaml` file to trigger a "Prepare Environment" event that would run `setup.sh`.

Malicious `action.yaml` edit

1. The attacker then used what appears to be a compromised identity to directly update all 35 tags in the project and point them to those staged commits (tag list below)

`setup.sh` functions similarly to the secret theft routines employed in the Trivy operation, gathering secrets then encrypting and exfiltrating them. It again attempts to send them to an attacker controlled server, with github repository creation as a backup, but uses a new exfiltration domain, `checkmarx.zone` (83.142.209.11) and exfiltration repo `docs-tpcp`. As with the Trivy operation, this malware can drop a python script intended to download and install a follow-on payload. However, this implementation adds a new Kubernetes focused persistence mechanism.

While `kics-github-action` has ~1% of the visible public usage of `trivy-action`, it is still broadly adopted publicly and privately as an Infrastructure as Code security scanner.

Github Compromise

The attack appears to have been accomplished via the compromise of the `cx-plugins-releases` (GitHub ID 225848595) service account, as that is the identity involved in publishing the malicious tags.

Script Functionality

Initialization, Data Theft, and Exfiltration

The script starts up with the flag set `-euo pipefail` to ensure it fails silently, then searches for GitHub Actions and generic runners and creates a staging file in `/tmp`. It then kicks off its credential search which conducts the following hunts:

- Gathers environmental variables, SSH private/public keys and searches for any files that contain these values.
- If it detected it is running on a GitHub-hosted runner, it executes a python script to access `/proc/[PID]/mem` to dump the memory of `Runner.Worker` and carves JSON objects flagged as secrets via regular expressions
- A second python script queries AWS instance Metadata endpoints (IMDSv1/v2), crawls common directories (`/home`, `/root`, etc) and queries the Kubernetes API if local service account tokens are found.

Any identified secrets are bundled into a collected plaintext file, then encrypted using AES-256-CBC and a random session key (`openssl rand 32 > "$_SESSION_KEY"`). The session key is then encrypted using the attacker's hardcoded RSA public key. This is the same public key used in the previous operation. Both files are then compressed into `tpcp.tar.gz`.

The script attempts to exfiltrate them via curl. The HTTP status code is captured to verify success:

If it doesn't succeed and the compromised environment contains a `GITHUB_TOKEN`, the script authenticates to the GitHub API. It programmatically creates a new repository (`/docs-tpcp/`) within the victim's GitHub account, creates a software "Release", and uploads the stolen encrypted tarball as a Release Asset. The previous campaign used `/tpcp-docs`

New Kubernetes Based Persistence

`setup.sh` attempts to create the ability for a follow-on operation via the same `sysmon.py` script as in the Trivy operation; however this time a new Kubernetes based method is invoked, in an attempt to survive the deletion of the initial CI container. If the script is run on a Kubernetes container that has sufficient permissions, it will attempt to deploy a highly privileged pod and run `sysmon.py` within it. If it is run on a standard linux host, it creates hidden directories in the users home folder `~/.config/sysmon` and writes the python backdoor there. The backdoor is the same as in the previous version, except it reads from `https://checkmarx.zone/raw`, rather than the `ico` address used in the Trivy operation.

Privileged Pod Creation & Malware Instalation

The script checks to see if the file: `/var/run/secrets/kubernetes.io/serviceaccount/token` exists and if it does, it uses that token to query `/api/v1/nodes` to gather a list of every underlying physical or virtual server in the cluster and extracts the `node_name`.

For each node, the script generates a pod manifest and payload script. The script uses the pod manifest to generate a dynamic name and places a highly privileged pod in the 'namespace':`kube-system`. The pod is configured to run the previously generated custom payload script and then quit. In each identified node the payload script installs the `sysmon.py` backdoor to `/root/.config/sysmon/sysmon.py` and creates a systemd unit file at `/root/.config/systemd/user/sysmon.service` to ensure the python script is run.

Pod Setup Code

OpenVSX Payload

Both compromised extensions (`ast-results v2.53.0` and `cx-dev-assist v1.7.0`) contained identical payloads. They were published 12 seconds apart at 12:53 UTC on March 23, 2026, via the `ast-phoenix` account on Open VSX. The VS Code Marketplace versions appear unaffected.

Payload Execution Flow

1. On activation of the extension, the new malicious `environmentAuthChecker.js` is invoked from `activateCore.js`
2. This payload first checks if the victim has credentials for at least one cloud provider

Credential Gating within the payload

If any credentials are detected, the second-stage payload is retrieved from the C2:

```
checkmarx[.]zone/static/checkmarx-util-1.0.4.tgz
```

Retrieval of second stage from C2

1. The payload attempts execution via npx, bunx, pnpm, or yarn dlx. This covers major JavaScript package managers. The retrieved package contains a comprehensive credential stealer.
2. Harvested credentials are then encrypted, using the keys as elsewhere in this campaign, and exfiltrated to `checkmarx[.]zone/vsx` as `tcp.tar.gz`.

`scand()` function hunting credentials

On non-CI systems, the malware installs persistence via a systemd user service. The persistence script polls `https://checkmarx[.]zone/raw` every 50 minutes for additional payloads, with a kill switch that aborts if the response contains "youtube". Currently, the link redirects to The Show Must Go On by Queen.

`persist()` persistence function

Compromised Artifacts

OpenVSX Extensions

Artifact	SHA256
ast-results-2.53.0.vsix	65bd72fcddaf938cefd55b3323ad29f649a65d4ddd6aea09afa974dfc7f105d
cx-dev-assist-1.7.0.vsix	744c9d61b66bcd2bb5474d9afeee6c00bb7e0cd32535781da188b80eb59383e0
checkmarx-util-1.0.4.tgz	0d66d8c7e02574ff0d3443de0585af19c903d12466d88573ed82ec788655975c
environmentAuthChecker.js	527f795a201a6bc114394c4cfd1c74dce97381989f51a4661aafbc93a4439e90

kics-github-action Releases

The [v1.1 release](#) was the only malicious release created. Other releases, triggered automatically by the tag events, failed because those versions already existed.

kics-github-action Tags

Tag	Commit SHA
v1	0e22ec8d1e0dda3c62bf4beffcd4a8a5db1abda1
v1.0	45f3749467a6017cb4fb749054b498d149dd5924
v1.1	8e20c7a67bb95632e2040327a355fb97e6014d29

Tag	Commit SHA
v1.2	93de85c910d859b759cf9185aa78d5a23a4b7000
v1.3	0e7343ba084735863db92b6f8ba2fa9dee604f7c
v1.4	2dc0fa613f6f4c15f26ad98225ad253475681616
v1.5	f00191dd3352c0cd83c6cce4e6bf04b628214dd0
v1.6	e0359b1a253ee66c8018586c3225e6e9cd2d8a4f
v1.6.1	dc6dbf358998c0c64da83edc8fcd581c12656b19
v1.6.2	08b9ea97eb292d5e1f9ac2d8e21c0ba32f0fdff0
v1.6.3	005fb0837553de722f8bf11d98e905dbdde19861
v1.7.0	a5471d37c656ecd4560e8e0b3977910f27025618
v2	3d49875ed47c6b8b4c8b50e0421418cf6b9f35f4
v2.0.0	121c38fb49c9fc82160245fb6e2a9119db636e4d
v2.1.0	1e9eeaba37fe0032deba133f598e74dab0ceb3b7
v2.1.1	c5c07508527fc6a125855eebfb533e64f675bd8e
v2.1.2	c999dbb9cc904e23675f9929f7e0e51d132879cf
v2.1.3	4ebf62dd8ff318412b38d19841fc3c8650e294bf
v2.1.4	3ae9f0d6f8139964635d411149f9b3e0a6eb935e
v2.1.5	96a0e8eb31c3cce6c495c9a49dd49c881cd17934
v2.1.6	31fbf5831a2e52429738fdc0cbaa20e57872b6fc
v2.1.7	fca3a20afcb8ec7f9932c060a236d2a9021fdd2b
v2.1.8	0f81f132f9f09bb4976d403914a44a1a1eb6158d
v2.1.9	c0e23718a5074f3b8ad286f37b532e02057af35f
v2.1.10	d66f0657133bc42f8264458063999bf1910490db
v2.1.11	e35c9d6a5faffc1c5b3450d0bf09006aa9b9e906
v2.1.12	2eee333d70fb6e14ce1d4aa73f12058bc5d70193
v2.1.13	f9641eb512f5c6530d13275903e8a97baf0925f1
v2.1.14	e8754eebc822b5122e96a6142b28dbc0e179c91c

Tag	Commit SHA
v2.1.15	69b3f020390222a9fcb6029ba56533b2fb12f103
v2.1.16	db942a0dd7e9d1aeac72bc675bdb67f39a688b63
v2.1.17	208813bf5feca5df9a935363cd426bc914614d0b
v2.1.18	3fdeadb81fbeddc1453163cc87bc173911fd47e2
v2.1.19	310734c0ffd29438f6195a24e2cbbacfdc33c9ab
v2.1.20	b974e53df1e3a2cd22ea90f0ec01882394feede4

Which actions should security teams take?

1. **Audit KICS GitHub Actions references:** Review workflows using `kics-github-action` . If you referenced a version tag rather than a SHA, check workflow run logs from the exposure window for signs of compromise.
2. **Search for exfiltration artifacts:** Look for repositories named `docs-tpcp` in your GitHub organization, which may indicate successful exfiltration via the fallback mechanism.

Long-term hardening: Refer to Wiz's [How to Harden GitHub Actions: The Unofficial Guide](#)

How can Wiz help?

[Watch the Webinar](#)

Source: <https://www.wiz.io/blog/teampcp-attack-kics-github-action>