INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-29 06:52 UTC

LAPSUS$ Exploits Trivy Supply Chain Breach to Compromise Checkmarx GitHub, Publish Malicious DevSecOps Artifacts

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0236
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Checkmarx KICS (malicious Docker images, VSCode extensions, Open VSX extensions), Trivy (supply chain compromise attributed to TeamPCP), GitHub private repositories
Published	2026-04-28T10:50:40
Discovery Source	Rss

Executive Summary

LAPSUS$, using credentials stolen in a prior supply chain compromise of the open-source Trivy scanner, gained access to Checkmarx's private GitHub repositories and published malicious Docker images and IDE extensions for the KICS infrastructure-as-code security scanner. Approximately 96GB of data was exfiltrated and published across dark web and clearnet extortion portals. Security and development teams that downloaded Checkmarx KICS artifacts recently, Docker images, VSCode extensions, or Open VSX extensions, may have introduced trojanized tooling directly into their build and development pipelines.

Technical Analysis

Attack chain: Threat actor TeamPCP compromised Trivy's supply chain and harvested credentials (CWE-522: Insufficiently Protected Credentials). LAPSUS$ leveraged those credentials via valid account abuse (T1078) to authenticate to Checkmarx's private GitHub repositories (T1213). Attackers staged and published malicious artifacts, Docker Hub images and VSCode/Open VSX Marketplace extensions associated with Checkmarx KICS, containing trojanized binaries (T1554) assembled outside your organization's artifact provenance controls (CWE-829). Downstream consumers who pulled these artifacts without integrity verification (CWE-494) were exposed to credential, key, token, and config file theft via malicious KICS tooling (T1552). Exfiltration of 96GB of data was routed over web services (T1567.001/T1567.002). No CVE has been assigned; this is a supply chain compromise, not a software vulnerability. Affected artifact classes: Docker Hub images tagged as Checkmarx KICS, VSCode Marketplace extensions for KICS, Open VSX extensions for KICS. No vendor-confirmed safe

version range has been published at the time of this item's creation, treat all recently pulled KICS artifacts as suspect pending Checkmarx's integrity verification guidance. CWEs: CWE-522, CWE-829, CWE-295, CWE-494. MITRE techniques: T1195.001, T1554, T1552, T1078, T1213, T1567.001, T1567.002, T1059.004, T1199, T1608.001, T1588.001.

Action Checklist

1. Step 1: Containment, Immediately quarantine any hosts that pulled Checkmarx KICS Docker images or installed KICS VSCode/Open VSX extensions within the past 90 days. Block outbound connections from those hosts pending investigation. Pull the Checkmarx official blog incident update (https://checkmarx.com/blog/supply-chain-security-incident-update/) for the vendor's confirmed artifact inventory and revocation list.
2. Step 2: Detection, Audit Docker pull history and container registries for KICS images. Query IDE extension logs for KICS extension installation events. Review CI/CD pipeline logs for Trivy and KICS invocations. Search SIEM for outbound connections from build agents to unknown external hosts post-KICS execution. Look for credential files, tokens, or SSH keys accessed by KICS scanner processes. Check GitHub audit logs for authentication events using service accounts with Trivy or Checkmarx-adjacent permissions.
3. Step 3: Eradication, Remove all KICS Docker images and IDE extensions installed from Docker Hub or marketplace sources until Checkmarx confirms clean artifact hashes. Rotate all credentials, API keys, tokens, and SSH keys present on systems where malicious artifacts executed. Revoke and reissue any secrets stored in repositories or CI/CD pipelines accessible from compromised hosts. Do not reinstall KICS tooling until Checkmarx publishes verified artifact hashes or a confirmed clean release.
4. Step 4: Recovery, Verify artifact integrity against vendor-published hashes before reinstalling any KICS tooling. Re-scan all code and infrastructure previously analyzed by KICS during the suspect window using a verified clean instance or an alternative scanner. Monitor for anomalous authentication events using rotated credentials for at least 30 days post-remediation. Confirm no unauthorized repository access persists via GitHub audit logs.
5. Step 5: Post-Incident, Conduct a pipeline dependency audit: map every open-source tool integrated into your CI/CD chain and document the trust boundary for each. Implement artifact signing and hash verification (SLSA framework or equivalent) for all third-party tooling pulled into pipelines. Apply least-privilege to service accounts used by scanners, KICS and Trivy should not have write or exfiltration-capable access. Review credential storage practices in repositories against NIST SP 800-53 IA-5 and SC-28 controls.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to executive leadership, legal counsel, and your breach notification team immediately if GitHub audit logs confirm unauthorized access to repositories containing customer data, PII, PHI, PCI-scoped code, or regulated infrastructure configurations, as LAPSUS$'s confirmed 96GB exfiltration and public clearnet/dark web publication triggers breach notification obligations under GDPR Article 33, HIPAA 45 CFR §164.410, and applicable state breach notification laws; also escalate if rotated credentials show any post-rotation authentication activity indicating LAPSUS$ retains persistent access.
Recovery Notes	Before restoring KICS or Trivy to any pipeline, require Checkmarx to publish SHA256 hashes and SLSA provenance attestations for the clean release, then verify each artifact independently using `sha256sum` and `slsa-verifier` before deployment. Re-execute all IaC security scans performed by KICS during the 90-day suspect window using a verified alternative scanner (Checkov, Semgrep, or Terrascan) and treat any previously clean scan results as untrusted until re-validated, since the malicious KICS artifact may have suppressed or falsified findings. Maintain enhanced GitHub audit log monitoring and cloud provider API call alerting for a minimum of 30 days post-credential rotation, given LAPSUS$'s documented pattern of maintaining persistent access through secondary credential stores and OAuth token abuse.
Forensic Artifacts	Docker daemon pull logs and image digests: `/var/log/docker` or `journalctl -u docker` on Linux build agents — cross-reference all `checkmarx/kics` image digests pulled during the 90-day window against Checkmarx's pre-incident verified digest list to identify exactly which hosts executed the malicious image layers | GitHub organization audit log (REST API or exported CSV): filter for `git.clone`, `git.push`, OAuth token creation/use, and repository permission changes attributed to service accounts associated with Trivy or KICS CI/CD integrations — this directly maps LAPSUS$'s lateral movement path from the Trivy supply chain compromise into Checkmarx's private repositories | CI/CD pipeline execution logs with KICS and Trivy invocation records: GitHub Actions runner logs, Jenkins build logs, or GitLab CI job traces capturing process arguments, environment variable exposure, and any unexpected network calls (curl/wget to non-registry endpoints) during scanner execution steps — these establish what credentials KICS had access to and whether exfiltration occurred | File system access audit records on build agents for credential file paths: auditd logs (`ausearch -k credential_access`) or Sysmon Event ID 11 (FileCreate) and Event ID 10 (ProcessAccess) filtered on paths matching `~/.ssh/`, `~/.aws/credentials`, `~/.kube/config`, `.env` files, and GitHub Actions runner `_work/` directories — identifies which secrets the malicious KICS artifact accessed or staged for exfiltration | VSCode extension install records and VSIX package manifests: `~/.vscode/extensions/checkmarx.kics-*/` directory contents including `package.json` (version and publisher hash), any bundled JavaScript with obfuscated network calls, and VSCode extension host logs at `~/.config/Code/logs/` — the malicious Open VSX/VSCode extension would leave behavioral artifacts in the extension host process logs distinct from the legitimate KICS scanner extension

Per-Action IR Details

Step 1: Containment — Immediately quarantine any hosts that pulled Checkmarx KICS Docker images or installed KICS VSCode/Open VSX extensions within the past 90 days. Block outbound connections from those hosts pending investigation. Pull the Checkmarx official blog incident update (checkmarx.com/blog/supply-chain-security-incident-update/) for the vendor's confirmed artifact inventory and revocation list.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: isolate affected systems to prevent further credential exfiltration or lateral movement while preserving forensic state

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Use iptables or Windows Firewall to block all outbound traffic from affected build agents except to approved update servers: `iptables -I OUTPUT -j DROP` on Linux CI runners, then allowlist only internal registries. For Docker hosts, run `docker network disconnect ` to isolate running KICS containers without destroying forensic state. Cross-reference Docker pull timestamps using `docker inspect --format='{{.Created}}' ` against the 90-day window. For VSCode, check `%APPDATA%\Code\User\extensions\` (Windows) or `~/.vscode/extensions/` (Linux/Mac) for KICS extension directories and record install timestamps before removal.

Evidence: BEFORE quarantine, capture: full outbound network connection state from build agents using `ss -tnp` (Linux) or `netstat -anob` (Windows) to document any active C2 or exfiltration connections initiated post-KICS execution; Docker daemon logs at `/var/log/docker` or `journalctl -u docker` showing exact image pull timestamps and image digests for all KICS images; VSCode extension manifest files at `~/.vscode/extensions/checkmarx.kics-*/package.json` recording the installed version hash; container runtime logs showing KICS process tree, environment variable exposure, and any volume mounts that could have exposed secrets.

Step 2: Detection — Audit Docker pull history and container registries for KICS images. Query IDE extension logs for KICS extension installation events. Review CI/CD pipeline logs for Trivy and KICS invocations. Search SIEM for outbound connections from build agents to unknown external hosts post-KICS execution. Look for credential files, tokens, or SSH keys accessed by KICS scanner processes. Check GitHub audit logs for authentication events using service accounts with Trivy or Checkmarx-adjacent permissions.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: correlate artifact pull events, process execution chains, and GitHub authentication anomalies to establish the full scope of LAPSUS$ access and malicious artifact execution

Controls: NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: For teams without SIEM: use `docker history ` and `docker inspect ` to enumerate layers and environment variables baked into pulled KICS images; compare image digest SHA256 against any vendor-published clean hashes. Query GitHub audit log via API (`GET /orgs/{org}/audit-log?phrase=actor:kics+action:git.clone`) filtering for service account activity. On CI runners (Jenkins, GitLab, GitHub Actions), grep pipeline logs for Trivy and KICS execution: `grep -rE '(trivy|kics)' /var/log/jenkins/` or equivalent. Use osquery to hunt credential file access: `SELECT * FROM file_events WHERE path LIKE '%/.ssh/%' OR path LIKE '%/.aws/credentials' AND time > ;`. Write a Sigma rule targeting process creation where parent process matches KICS or Trivy executables spawning curl, wget, or python with outbound IP arguments — reference MITRE ATT&CK T1552.001 (Credentials in Files) and T1041 (Exfiltration Over C2 Channel).

Evidence: GitHub organization audit log entries for the 90-day window showing: OAuth token authentications, repository clone/fork events from Trivy or KICS service accounts, and any `git.push` or package publish events from non-human actors consistent with LAPSUS$ TTPs (MITRE T1586.002 — Compromise Accounts); CI/CD pipeline execution logs from GitHub Actions (`.github/workflows/` runner logs), Jenkins build history, or GitLab CI job traces showing KICS and Trivy invocation arguments, environment variable injection, and any unexpected network calls during scan steps; Docker registry pull logs (Docker Hub access logs or internal registry logs) with image digests for `checkmarx/kics` pulls, specifically flagging any digest that does not match Checkmarx's pre-incident verified hash list; process execution telemetry (Sysmon Event ID 1 or auditd `execve`) on build agents showing child processes spawned by KICS or Trivy binaries, particularly shell spawns, curl/wget calls, or SSH invocations inconsistent with normal scanner behavior.

Step 3: Eradication — Remove all KICS Docker images and IDE extensions installed from Docker Hub or marketplace sources until Checkmarx confirms clean artifact hashes. Rotate all credentials, API keys, tokens, and SSH keys present on systems where malicious artifacts executed. Revoke and reissue any secrets stored in repositories or CI/CD pipelines accessible from compromised hosts. Do not reinstall KICS tooling until Checkmarx publishes verified artifact hashes or a confirmed clean release.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: remove malicious KICS and Trivy artifacts from all execution environments and eliminate all credentials that were accessible to those artifacts, given LAPSUS$'s established pattern of credential harvesting and repository exfiltration

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST IA-5 (Authenticator Management), CIS 5.2 (Use Unique Passwords), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Remove KICS Docker images from all hosts: `docker rmi -f $(docker images | grep kics | awk '{print $3}')` and purge from local cache with `docker system prune -a`. For VSCode, run `code --uninstall-extension checkmarx.kics` or manually delete from `~/.vscode/extensions/`. Enumerate all secrets accessible to CI runners during the suspect window using GitHub's secret scanning API (`GET /repos/{owner}/{repo}/secret-scanning/alerts`) and revoke any flagged tokens immediately. For SSH key rotation, audit `~/.ssh/authorized_keys` and `known_hosts` on all build agents, revoke any key pairs generated before or during the suspect window, and reissue using `ssh-keygen -t ed25519`. For GitHub Actions, rotate all repository and organization secrets via Settings > Secrets and re-scope service account PATs to minimum required permissions. Use `truffleHog` or `git-secrets` CLI to scan repository history for any secrets inadvertently committed during the period KICS had repository access.

Evidence: Before credential rotation, preserve forensic copies of: the full list of GitHub repository secrets, Actions environment variables, and CI/CD pipeline variable stores accessible from compromised runners (document names only, not values, for chain-of-custody); SSH `authorized_keys` and `known_hosts` files from all build agents to establish which keys were present during the exposure window; any `.env` files, `kubeconfig` files, cloud provider credential files (`~/.aws/credentials`, `~/.azure/`, `~/.config/gcloud/`) found on compromised hosts; Docker image layer contents extracted via `docker save | tar xv` to identify what malicious payload was embedded in the KICS image layers, specifically looking for added scripts in `/usr/local/bin/`, modified entrypoints, or injected environment variable exfiltration routines.

Step 4: Recovery — Verify artifact integrity against vendor-published hashes before reinstalling any KICS tooling. Re-scan all code and infrastructure previously analyzed by KICS during the suspect window using a verified clean instance or an alternative scanner. Monitor for anomalous authentication events using rotated credentials for at least 30 days post-remediation. Confirm no unauthorized repository access persists via GitHub audit logs.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: restore pipeline integrity only after verified clean artifacts are available, and implement 30-day anomalous authentication monitoring to detect any LAPSUS$-held credentials being used post-rotation

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST IR-4 (Incident Handling), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Verify KICS artifact integrity using `sha256sum` against Checkmarx-published hashes before any reinstallation: `echo ' checkmarx-kics-.tar.gz' | sha256sum -c`. For Docker images, verify digest: `docker pull checkmarx/kics:@sha256:`. Re-scan IaC previously processed by KICS using Semgrep (free tier), Checkov, or Terrascan as alternative scanners — specifically re-analyze any Terraform, CloudFormation, Kubernetes, or Dockerfile content that KICS scanned during the 90-day window, as LAPSUS$ may have exfiltrated those files. Monitor GitHub audit logs daily for 30 days using: `GET /orgs/{org}/audit-log?phrase=action:git&after=` and alert on any authentication from IP addresses not matching known CI/CD egress ranges. Use `fail2ban` or equivalent on CI runners to alert on repeated authentication failures with new credentials, which could indicate LAPSUS$ testing rotated credential validity.

Evidence: During recovery validation, collect and retain: GitHub audit log exports covering the 30-day post-rotation monitoring window, specifically filtering for `git.clone`, `git.push`, and OAuth token usage events; re-scan outputs from the alternative IaC scanner (Checkov, Semgrep) for the same codebase scope previously covered by KICS, preserving diff reports that could indicate LAPSUS$ had access to sensitive infrastructure configurations; network flow logs or firewall logs from CI/CD egress points during the monitoring window to confirm no outbound connections to the same external IPs or domains observed during the initial KICS execution period; authentication logs from cloud providers (AWS CloudTrail, Azure Activity Log, GCP Audit Log) for API calls using credentials that were present on compromised hosts, covering the full 90-day suspect window plus 30-day post-rotation monitoring period.

Step 5: Post-Incident — Conduct a pipeline dependency audit: map every open-source tool integrated into your CI/CD chain and document the trust boundary for each. Implement artifact signing and hash verification (SLSA framework or equivalent) for all third-party tooling pulled into pipelines. Apply least-privilege to service accounts used by scanners — KICS and Trivy should not have write or exfiltration-capable access. Review credential storage practices in repositories against NIST SP 800-53 IA-5 and SC-28 controls.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: lessons learned from LAPSUS$'s exploitation of Trivy's supply chain trust to pivot into Checkmarx must drive systematic hardening of all open-source scanner integrations and CI/CD trust boundaries

Controls: NIST IA-5 (Authenticator Management), NIST SC-28 (Protection of Information at Rest), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-7 (Least Functionality), NIST IR-8 (Incident Response Plan), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Implement SLSA provenance verification using the free `slsa-verifier` CLI tool (`slsa-verifier verify-artifact`) for any open-source scanner binary pulled into CI/CD pipelines, starting with Trivy and KICS as the two tools directly implicated in this campaign. Pin all Docker image references in pipeline configs to immutable SHA256 digests rather than mutable tags (e.g., replace `checkmarx/kics:latest` with `checkmarx/kics@sha256:`). Enforce least-privilege for scanner service accounts: GitHub Actions KICS and Trivy steps should run with a scoped token granting `contents:read` only — add `permissions: contents: read` to the workflow YAML block. Use `cosign` (free, Sigstore project) to verify image signatures: `cosign verify ` before any pipeline pull. Document each open-source tool in your CI/CD chain in a software bill of materials (SBOM) using `syft` CLI: `syft -o spdx-json > sbom.json` — this gives you a baseline to detect future supply chain tampering. Schedule a quarterly review of all CI/CD-integrated scanners against their upstream project health indicators (release cadence, maintainer activity, recent security advisories).

Evidence: For lessons-learned documentation and future detection baselining, preserve: the complete CI/CD dependency map produced by the pipeline audit, including all Docker image references, their pull sources, and pinned vs. floating tag usage — this becomes the baseline for detecting future supply chain drift; SBOM outputs generated by `syft` or equivalent for all scanner images currently in use, establishing a known-good artifact inventory against which future image digests can be compared; GitHub Actions workflow YAML files from the incident period documenting the permission scopes granted to KICS and Trivy steps, to evidence the over-privileged access that enabled LAPSUS$ to reach repository secrets; a documented trust boundary map identifying which CI/CD service accounts had access to which secret stores, repository scopes, and cloud provider credentials — this directly addresses the attack path LAPSUS$ used to pivot from Trivy's supply chain compromise into Checkmarx's GitHub repositories and ultimately to the 96GB exfiltration.

Detection Guidance

Docker: Query registry pull logs for KICS image tags pulled from Docker Hub within the past 90 days. Compare image digests against vendor-confirmed clean hashes when published. IDE extensions: Search endpoint management logs (e.g., Intune, Jamf, or local extension manifests) for KICS VSCode or Open VSX extension installation events. CI/CD pipelines: Review pipeline execution logs for Trivy and KICS invocations; flag any pipeline steps that contact external hosts not in your approved egress list. Network: Alert on outbound traffic from build agents or developer workstations to cloud storage endpoints (S3, GCS, Azure Blob) not previously observed, consistent with T1567.002 exfiltration behavior. Credential access: Look for processes spawned by KICS scanner binaries accessing .env files, ~/.ssh/, token stores, or CI/CD secret variables. GitHub: Pull audit logs for repository access events authenticated with service accounts or tokens associated with Checkmarx or Trivy integrations; flag any access from unexpected IPs or user agents. Behavioral indicator: Scanner tooling that spawns unexpected child processes or makes DNS queries to unfamiliar domains during a scan run is a high-confidence indicator of trojanization.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://hub.docker.com/ (KICS-tagged images — specific malicious image digests to be confirmed via Checkmarx advisory)`	Malicious Docker images published under Checkmarx KICS branding on Docker Hub	MEDIUM
URL	`VSCode Marketplace — KICS extension (specific extension ID and version to be confirmed via Checkmarx advisory)`	Trojanized VSCode extension for Checkmarx KICS distributed via official marketplace	MEDIUM
URL	`Open VSX Registry — KICS extension (specific extension ID and version to be confirmed via Checkmarx advisory)`	Trojanized Open VSX extension for Checkmarx KICS	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1486** — Data Encrypted for Impact
- **T1213** — Data from Information Repositories
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1608.001** — Upload Malware
- **T1588.001** — Malware
- **T1552** — Unsecured Credentials
- **T1059.004** — Unix Shell
- **T1078** — Valid Accounts
- **T1567.002** — Exfiltration to Cloud Storage
- **T1199** — Trusted Relationship
- **T1567.001** — Exfiltration to Code Repository
- **T1554** — Compromise Host Software Binary

NIST-800-53R5

- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AC-2** — Account Management

- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SC-8** — Transmission Confidentiality and Integrity
- **SC-17** — Public Key Infrastructure Certificates
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures
- **A02:2021** — Cryptographic Failures
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **5.2** — Use Unique Passwords
- **3.10** — Encrypt Sensitive Data in Transit
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners
- **CC6.3** — Authorizes, modifies, or removes access

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1486	Data Encrypted for Impact	Impact

Technique ID	Technique Name	Tactic
T1213	Data from Information Repositories	Collection
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1608.001	Upload Malware	Resource-Development
T1588.001	Malware	Resource-Development
T1552	Unsecured Credentials	Credential-Access
T1059.004	Unix Shell	Execution
T1078	Valid Accounts	Defense-Evasion
T1567.002	Exfiltration to Cloud Storage	Exfiltration
T1199	Trusted Relationship	Initial-Access
T1567.001	Exfiltration to Code Repository	Exfiltration
T1554	Compromise Host Software Binary	Persistence

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/checkmarx-confirms-l...	T3
	https://www.bleepingcomputer.com/news/security/french-govt-agency-c...	T3
Malicious KICS Docker Images and VS Code Extensions Hit ...	https://thehackernews.com/2026/04/malicious-kics-docker-images-and-...	T3
Supply Chain Security Incident Update - Checkmarx	https://checkmarx.com/blog/supply-chain-security-incident-update/	T3
Checkmarx confirms LAPSUS$ hackers leaked its stolen GitHub data	https://www.bleepingcomputer.com/news/security/checkmarx-confirms-l...	T3

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-04-29 06:52 UTC by TJS Security Command Center