INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-22 18:51 UTC

Checkmarx KICS and VS Code Extensions Weaponized to Exfiltrate IaC Secrets Across DevSecOps Pipelines

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0203
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Checkmarx KICS Docker Hub tags (v2.1.20, alpine, v2.1.21); Checkmarx VS Code Extension versions 1.17.0 and 1.19.0; Terraform, CloudFormation, and Kubernetes IaC scanning workflows
Published	2026-04-22T13:55:00
Discovery Source	Rss

Executive Summary

Unknown threat actors compromised Checkmarx's official Docker Hub repository and VS Code Extension distribution channel, embedding credential-stealing malware in KICS versions used to scan Infrastructure as Code files. Any organization running KICS in CI/CD pipelines with these affected versions should assume cloud credentials, API keys, and infrastructure secrets captured during scans have been exfiltrated to an attacker-controlled endpoint. The business risk is immediate and severe: exposed credentials provide direct access to cloud environments, enabling data theft, ransomware deployment, or full infrastructure takeover.

Technical Analysis

Threat actors overwrote legitimate Checkmarx KICS Docker Hub image tags (v2.1.20, alpine, v2.1.21) with malicious variants and simultaneously embedded malicious code in VS Code Extension versions 1.17.0 and 1.19.0. The malware encrypts IaC scan output, which routinely contains plaintext secrets embedded in Terraform, CloudFormation, and Kubernetes configurations, and exfiltrates it to an external attacker-controlled endpoint (T1041, T1552.001). The multi-vector delivery across Docker Hub and VS Code Marketplace indicates a supply chain compromise affecting multiple distribution channels (T1195.002, T1608.001). The malware uses obfuscation (T1027) and masquerades within legitimate tool packaging (T1036.005, T1553.002). JavaScript execution is used as part of the payload chain (T1059.007). Relevant weaknesses: CWE-506 (embedded

malicious code), CWE-311 (missing encryption of sensitive data at source), CWE-494 (download of code without integrity check), CWE-829 (inclusion of functionality from untrusted control sphere). No CVE has been assigned. CVSS base score assessed at 9.5 (critical). Checkmarx has published a security advisory at checkmarx.com/blog/checkmarx-security-update/. Immediate remediation: stop using affected image tags and extension versions, pull clean verified images, and rotate all secrets that passed through affected scanning workflows.

Action Checklist

1. Step 1: Containment, Immediately stop all CI/CD pipeline jobs and developer workloads running KICS Docker image tags v2.1.20, alpine, or v2.1.21, or VS Code Extension versions 1.17.0 or 1.19.0. Quarantine or terminate any containers spun from those images. Block outbound traffic from pipeline infrastructure to unrecognized external endpoints pending investigation.
2. Step 2: Detection, Audit Docker pull logs and VS Code Extension install records for use of the affected tags and versions across all pipeline agents, developer workstations, and build servers. Review network egress logs from CI/CD infrastructure for unexpected outbound HTTPS connections, particularly to endpoints not in your approved inventory. Check pipeline job logs for anomalous subprocess execution or encrypted payload activity. Correlate against MITRE T1041 (exfiltration over C2 channel) and T1552.001 (credentials in files) behavioral patterns.
3. Step 3: Eradication, Replace affected KICS Docker images with verified clean versions by pulling explicitly pinned image digests confirmed against Checkmarx's security advisory. Uninstall affected VS Code Extension versions and install a verified clean release confirmed via the Checkmarx security update post. Verify image integrity using digest pinning rather than mutable tags going forward.
4. Step 4: Recovery, Treat all secrets, API keys, cloud credentials, and tokens that were present in any IaC files scanned by affected versions as fully compromised. Rotate them immediately across all target systems: AWS IAM keys, Azure service principals, GCP service accounts, Kubernetes secrets, Terraform state backends, and any other credential stores referenced in scanned configurations. After rotation, validate that attacker-held credentials are revoked and confirm no unauthorized access occurred in cloud audit logs (AWS CloudTrail, Azure Monitor, GCP Audit Logs) during the exposure window.
5. Step 5: Post-Incident, Implement mandatory image digest pinning in all CI/CD pipelines to prevent mutable tag substitution attacks. Add extension and container image integrity verification (sigstore/cosign or equivalent) to your supply chain security controls, aligned with NIST SP 800-161 supply chain risk management guidance. Review your secrets management posture: secrets should not be embedded in IaC files in plaintext; adopt a secrets manager (HashiCorp Vault, AWS Secrets Manager, or equivalent) with dynamic secret injection. Map this incident to CIS Benchmark controls for container security and CI/CD pipeline hardening.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO, legal counsel, and breach notification counsel immediately if CloudTrail, Azure Monitor, or GCP Audit Log review confirms any API calls, resource access, or authentication events using compromised credentials during the exposure window, or if scanned IaC files contain credentials for systems processing PII, PHI, PCI-DSS cardholder data, or regulated workloads — as these conditions likely trigger mandatory breach notification obligations under GDPR, HIPAA, or state privacy laws.
Recovery Notes	After credential rotation is complete, re-enable CI/CD pipelines only after replacing all affected KICS image references with digest-pinned clean versions and confirming cosign or digest verification gates are enforced in the pipeline definition. Monitor cloud provider audit logs (CloudTrail, Azure Monitor, GCP Audit Logs) continuously for 30 days post-rotation using alerting on the rotated credential IDs — attacker-held copies of old credentials may still be attempted if the attacker cached them before your revocation completed. Re-scan all IaC repositories with the verified clean KICS version to establish a new clean baseline, and compare results against pre-incident scan history to identify any infrastructure changes made using compromised credentials that may have introduced backdoors or unauthorized resources.
Forensic Artifacts	Docker image layer filesystem diff between affected KICS tags (v2.1.20, alpine, v2.1.21) and the verified clean release — the malicious payload will appear as an added or modified binary, script, or library in one or more layers, extractable via `docker history --no-trunc checkmarx/kics:v2.1.20` and `docker save | tar -xvf` layer inspection | CI/CD pipeline runner process execution logs showing child processes spawned by the KICS binary that are inconsistent with IaC file scanning — the credential-stealing malware would need to execute a subprocess or make network calls to exfiltrate secrets, visible in GitLab Runner job traces, Jenkins build logs, or GitHub Actions step logs as anomalous process names or curl/wget invocations not part of normal KICS output | Network PCAP or flow records from the CI/CD subnet showing HTTPS connections (TLS SNI or JA3 fingerprint) initiated by the KICS container process to non-Checkmarx external endpoints during the scan job execution window — the exfiltration channel (MITRE T1041) would appear as an outbound connection to an attacker-controlled IP or domain immediately following a scan job that processed IaC files containing secrets | IaC file contents (Terraform .tf/.tfvars, CloudFormation templates, Kubernetes Secret manifests, Helm values.yaml) present in repositories scanned by affected KICS versions — these files define the exact credential types and identifiers that were exposed, enabling precise scoping of which AWS IAM keys, Azure service principals, GCP service account keys, and Kubernetes secrets must be treated as compromised | VS Code Extension activation and telemetry logs at `%APPDATA%\Code\logs\` (Windows) or `~/.config/Code/logs/` (Linux/macOS) for developer workstations with versions 1.17.0 or 1.19.0 installed — the extension's activation events will show when it was triggered against IaC files in the developer's workspace, establishing the local exposure window and which workspace directories (and their embedded secrets) were processed by the malicious extension version

Per-Action IR Details

Step 1: Containment — Immediately stop all CI/CD pipeline jobs and developer workloads running KICS Docker image tags v2.1.20, alpine, or v2.1.21, or VS Code Extension versions 1.17.0 or 1.19.0. Quarantine or terminate any containers spun from those images. Block outbound traffic from pipeline infrastructure to unrecognized external endpoints pending investigation.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

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: Run `docker ps --filter ancestor=checkmarx/kics:v2.1.20 --filter ancestor=checkmarx/kics:alpine --filter ancestor=checkmarx/kics:v2.1.21 -q | xargs docker stop` on all pipeline agents and build servers. On Linux, use `iptables -I OUTPUT -j DROP` with destination filters for unrecognized external IPs until a clean allowlist is rebuilt. On developer workstations, disable the VS Code Extension immediately via `code --uninstall-extension checkmarx.kics` and disconnect the workstation from CI/CD network segments. Use `ss -tunap` or `netstat -anp` to snapshot all active outbound connections before terminating containers — this preserves C2 endpoint evidence.

Evidence: Before stopping containers, capture: (1) `docker inspect ` output for all running KICS containers to record environment variables, mounted volumes, and network settings — the malware likely accessed secrets injected as env vars or mounted credential files; (2) active network connection state via `ss -tunap` or `netstat -anp` to identify live C2 connections from the KICS process; (3) container filesystem snapshot using `docker export > kics_forensic_.tar` to preserve any dropped payloads, modified binaries, or exfil staging files within the container layer before termination; (4) `/proc//net/tcp` and `/proc//environ` from the KICS process PID on the host to capture in-memory network state and runtime environment variables containing secrets.

Step 2: Detection — Audit Docker pull logs and VS Code Extension install records for use of the affected tags and versions across all pipeline agents, developer workstations, and build servers. Review network egress logs from CI/CD infrastructure for unexpected outbound HTTPS connections, particularly to endpoints not in your approved inventory. Check pipeline job logs for anomalous subprocess execution or encrypted payload activity. Correlate against MITRE T1041 (exfiltration over C2 channel) and T1552.001 (credentials in files) behavioral patterns.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Query Docker Hub pull history via the Docker Hub API: `curl -s 'https://hub.docker.com/v2/repositories/checkmarx/kics/tags/' | jq '.results[] | select(.name=="v2.1.20" or .name=="alpine" or .name=="v2.1.21")'`. On pipeline agents, grep CI/CD job logs for the affected image references: `grep -rE 'kics:(v2\.1\.20|alpine|v2\.1\.21)' /var/log/gitlab-runner/ /home/jenkins/` (adjust path for your runner). For VS Code Extension installs, check `~/.vscode/extensions/` for `checkmarx.kics-1.17.0` or `checkmarx.kics-1.19.0` directories. For network egress, use Zeek or `tcpdump -w kics_egress.pcap -i eth0 'tcp port 443'` filtered to the CI/CD subnet during the exposure window, then parse with `zeek-cut` filtering on non-allowlisted destinations. Deploy the Sigma rule for T1041/T1552.001 behavioral detection using `sigma convert` targeting your log format.

Evidence: Capture before analysis concludes: (1) Docker daemon logs at `/var/log/docker` or `journalctl -u docker` filtered for pull events referencing `checkmarx/kics` with tags `v2.1.20`, `alpine`, `v2.1.21` — timestamps establish the exposure window start; (2) VS Code Extension install logs at `%APPDATA%\Code\logs\` (Windows) or `~/.config/Code/logs/` (Linux/macOS) for extension activation events for versions 1.17.0 and 1.19.0; (3) CI/CD pipeline runner logs (GitLab Runner: `/var/log/gitlab-runner/`, Jenkins: `$JENKINS_HOME/jobs//builds//log`) for subprocess spawning by the KICS process — the malware would launch child processes not consistent with normal IaC scanning; (4) DNS query logs from pipeline infrastructure resolving attacker-controlled domains not in your approved third-party inventory; (5) TLS SNI fields from network captures showing HTTPS connections initiated by the KICS container process to non-Checkmarx external endpoints.

Step 3: Eradication — Replace affected KICS Docker images with verified clean versions by pulling explicitly pinned image digests confirmed against Checkmarx's security advisory. Uninstall affected VS Code Extension versions and install a verified clean release confirmed via the Checkmarx security update post. Verify image integrity using digest pinning rather than mutable tags going forward.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-2 (Baseline Configuration), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software)

Compensating: Remove all local cached copies of affected images: `docker rmi checkmarx/kics:v2.1.20 checkmarx/kics:alpine checkmarx/kics:v2.1.21 --force` and prune build cache with `docker buildx prune --all`. Pull the clean replacement using an explicit digest pin from the Checkmarx advisory (format: `docker pull checkmarx/kics@sha256:`). Verify the pulled image digest matches the advisory: `docker inspect --format='{{index .RepoDigests 0}}' checkmarx/kics@sha256:`. For VS Code: `code --uninstall-extension checkmarx.kics` then reinstall only from the verified VSIX package linked in the Checkmarx advisory using `code --install-extension checkmarx.kics-.vsix`. Run `cosign verify` against the image if Checkmarx provides a signed manifest; if not, treat digest pinning as the minimum integrity gate.

Evidence: Before eradicating, preserve forensic copies: (1) Export all affected image layers via `docker save checkmarx/kics:v2.1.20 | gzip > kics_v2.1.20_forensic.tar.gz` for malware analysis — the embedded credential-stealing payload will be identifiable in the image layer diff vs. the clean version; (2) Hash all affected VS Code Extension files at `~/.vscode/extensions/checkmarx.kics-1.17.0/` and `~/.vscode/extensions/checkmarx.kics-1.19.0/` using `find . -type f -exec sha256sum {} \;` before uninstallation to support vendor and law enforcement evidence requirements; (3) Document all pipeline `Dockerfile` and `docker-compose.yml` references to affected mutable tags — these files are evidence of the attack surface and scope.

Step 4: Recovery — Treat all secrets, API keys, cloud credentials, and tokens that were present in any IaC files scanned by affected versions as fully compromised. Rotate them immediately across all target systems: AWS IAM keys, Azure service principals, GCP service accounts, Kubernetes secrets, Terraform state backends, and any other credential stores referenced in scanned configurations. After rotation, validate that attacker-held credentials are revoked and confirm no unauthorized access occurred in cloud audit logs (AWS CloudTrail, Azure Monitor, GCP Audit Logs) during the exposure window.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), NIST IA-5 (Authenticator Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: For AWS: `aws iam list-access-keys --user-name ` to enumerate all keys, then `aws iam delete-access-key` for every key associated with roles or users whose credentials appeared in IaC files scanned by affected KICS versions. Query CloudTrail for unauthorized use during the exposure window: `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=AssumeRole --start-time --end-time `. For GCP: `gcloud iam service-accounts keys list` then revoke with `gcloud iam service-accounts keys delete`. For Kubernetes: `kubectl get secrets -A -o json | jq '.items[] | select(.metadata.creationTimestamp '')'` to identify pre-rotation secrets still in use. For Terraform state backends, rotate backend access credentials and re-encrypt state files if using remote backends (S3, GCS, Azure Blob) — state files themselves may contain plaintext secret values read during scanning.

Evidence: Before and during rotation, capture: (1) AWS CloudTrail `GetCallerIdentity`, `AssumeRole`, `CreateSession`, and any service API calls made with the compromised IAM key IDs during the exposure window — filter CloudTrail by the specific access key IDs found in scanned IaC files; (2) Azure Monitor activity logs filtered on service principal object IDs present in scanned Terraform or ARM templates for the exposure window; (3) GCP Audit Logs (`cloudaudit.googleapis.com/activity`) filtered on the service account emails referenced in scanned Kubernetes manifests or Terraform configs; (4) Kubernetes API server audit logs (`/var/log/kubernetes/audit.log`) for requests authenticated with secrets that existed in scanned manifests — look for `get`, `list`, `create` verbs against sensitive resource types from unexpected source IPs during the exposure window.

Step 5: Post-Incident — Implement mandatory image digest pinning in all CI/CD pipelines to prevent mutable tag substitution attacks. Add extension and container image integrity verification (sigstore/cosign or equivalent) to your supply chain security controls, aligned with NIST SP 800-161 supply chain risk management guidance. Review your secrets management posture: secrets should not be embedded in IaC

files in plaintext; adopt a secrets manager (HashiCorp Vault, AWS Secrets Manager, or equivalent) with dynamic secret injection. Map this incident to CIS Benchmark controls for container security and CI/CD pipeline hardening.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-7 (Software, Firmware, and Information Integrity), NIST SA-12 (Supply Chain Protection), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Enforce digest pinning immediately by linting all `Dockerfile` and pipeline YAML files: `grep -rE 'FROM checkmarx/kics:|image: checkmarx/kics:' . | grep -v sha256` will surface every mutable tag reference. Add a pre-commit hook using `opa eval` or a shell script that rejects any pipeline definition pulling a container image without a `@sha256:` digest suffix. Install `cosign` (free, CNCF project) and add `cosign verify ` as a mandatory pipeline gate for all third-party scanner images. For secrets management without budget: deploy HashiCorp Vault Community Edition on-premise and migrate IaC secrets to dynamic secrets with short TTLs — this directly eliminates the plaintext-credential-in-IaC-file attack surface that KICS malware exploited. Write a YARA rule targeting the specific malware strings or network indicators identified from the forensic image analysis in Step 3 and scan all pipeline agent filesystems monthly.

Evidence: For the lessons-learned record, preserve: (1) The complete list of IaC repositories scanned by affected KICS versions during the exposure window — this is the definitive blast radius document and must be retained for any regulatory breach notification analysis; (2) Network flow records showing all outbound connections from KICS containers to attacker-controlled endpoints — these are needed for threat intelligence sharing with CISA and sector ISACs; (3) The forensic image exports from Step 3 with chain-of-custody documentation, as Checkmarx and law enforcement may require them; (4) A timestamped audit trail of every credential rotation performed in Step 4, cross-referenced against the CloudTrail/Azure Monitor/GCP Audit Log review, to demonstrate to auditors that no post-compromise unauthorized access was confirmed or that confirmed access is being reported.

Detection Guidance

Query Docker pull logs for image tags checkmarx/kics:v2.1.20, checkmarx/kics:alpine, and checkmarx/kics:v2.1.21 across all pipeline agents and build infrastructure. Query VS Code Extension install and update logs for Checkmarx KICS extension versions 1.17.0 and 1.19.0 on developer workstations. Review outbound network logs from CI/CD build nodes for HTTPS POST requests to endpoints outside your known-good allowlist, particularly those receiving large or encrypted payloads. Look for subprocess spawning within KICS container execution that is inconsistent with normal scan behavior (T1059, T1059.007). Check cloud provider audit logs (CloudTrail, Azure Activity Log, GCP Audit Log) for API calls using credentials that were present in scanned IaC files, focusing on activity from unexpected source IPs or service principals during and after the exposure window. No public IOC hashes or C2 IP addresses have been confirmed in available sources at time of this report, monitor Checkmarx's security advisory page for updated indicators.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://hub.docker.com/r/checkmarx/kics (tags: v2.1.20, alpine, v2.1.21)`	Malicious KICS Docker image tags confirmed compromised — do not pull	HIGH

Type	Value	Context	Confidence
URL	`VS Code Marketplace — Checkmarx KICS Extension versions 1.17.0 and 1.19.0`	Malicious VS Code Extension versions confirmed compromised — uninstall immediately	HIGH

Framework Mappings

MITRE-ATTACK

- **T1552.001** — Credentials In Files
- **T1059.007** — JavaScript
- **T1195.002** — Compromise Software Supply Chain
- **T1608.001** — Upload Malware
- **T1553.002** — Code Signing
- **T1059** — Command and Scripting Interpreter
- **T1027** — Obfuscated Files or Information
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1041** — Exfiltration Over C2 Channel

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan
- **SC-13** — Cryptographic Protection

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **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

- **164.312(d)** — Person or Entity Authentication
- **164.312(e)(1)** — Transmission Security

SOC2-TSC

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

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
- **A.8.24** — Use of cryptography
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1552.001	Credentials In Files	Credential-Access
T1059.007	JavaScript	Execution
T1195.002	Compromise Software Supply Chain	Initial-Access
T1608.001	Upload Malware	Resource-Development
T1553.002	Code Signing	Defense-Evasion
T1059	Command and Scripting Interpreter	Execution
T1027	Obfuscated Files or Information	Defense-Evasion
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1041	Exfiltration Over C2 Channel	Exfiltration

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/04/malicious-kics-docker-images-and-...	T3
checkmarx/kics - Docker Image	https://hub.docker.com/r/checkmarx/kics	T3
Checkmarx Security Update	https://checkmarx.com/blog/checkmarx-security-update/	T3

Source	URL	Tier
KICs - Open-Source IaC by Checkmarx	https://checkmarx.com/product/kics/	T3
GitHub - Checkmarx/kics: Find security vulnerabilities, compliance ...	https://github.com/checkmarx/kics	T3

Generated 2026-04-22 18:51 UTC by TJS Security Command Center