INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-23 18:50 UTC

TeamPCP Weaponizes KICS Toolchain: Multi-Vector Attack Harvests Cloud Credentials from Developer Pipelines

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0210
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Checkmarx KICS (Docker Hub image, VS Code Marketplace extension, Open VSX extension), Checkmarx ast-github-action, Checkmarx Developer Assist extension, artifacts published or updated during April 22, 2026 exposure window
Published	2026-04-23T12:05:12
Discovery Source	Rss

Executive Summary

On April 22, 2026, threat actor TeamPCP compromised the Checkmarx KICS open-source infrastructure-as-code scanning toolchain across three simultaneous channels: Docker Hub, VS Code Marketplace, and Open VSX, during a 90-minute exposure window. Malicious artifacts were designed to steal cloud credentials (AWS, GCP, Azure), GitHub tokens, SSH keys, and CI/CD secrets from developer pipelines that pulled affected packages. Any organization that consumed KICS artifacts during that window faces confirmed credential compromise risk across cloud and source control environments, with direct exposure to unauthorized cloud access, data exfiltration, and pipeline takeover.

Technical Analysis

TeamPCP executed a coordinated supply chain attack against the Checkmarx KICS toolchain on April 22, 2026, with a confirmed ~90-minute exposure window. Three delivery vectors were simultaneously compromised: the official KICS Docker Hub image, the KICS VS Code Marketplace extension, and the KICS Open VSX extension. The Checkmarx ast-github-action and Developer Assist extension were also identified as affected components. The malware payload targeted credential theft across multiple secret types: AWS, GCP, and Azure cloud provider credentials; GitHub personal access tokens (T1528); SSH private keys (T1552.004); and CI/CD pipeline environment variables (T1552.001). Attack technique coverage includes: T1195.001 and T1195.002 (supply chain compromise of open-source and software components), T1608.001 (stage capabilities via upload

to legitimate infrastructure), T1554 (compromise client software binary), T1036.005 (masquerading as legitimate tooling), T1059/T1059.007 (script execution), T1071.001 (C2 over HTTP/S), and T1567.001 (exfiltration to web service). Relevant CWEs: CWE-494 (download of code without integrity check), CWE-829 (inclusion of functionality from untrusted control sphere), CWE-798 (use of hard-coded credentials as an artifact of the compromise), CWE-312 (cleartext storage of sensitive information). No CVE has been assigned. CVSS scores reflect the impact of credential theft but are not formally assigned to this campaign; severity is based on attack scope and operational impact. The incident is confirmed by Checkmarx in their April 22 security update. TeamPCP is linked by campaign pattern to prior supply chain compromises of the Trivy container vulnerability scanner and the LiteLLM AI gateway library. Checkmarx released clean replacement artifacts identified in their April 22, 2026 security advisory. Verify artifact integrity via Checkmarx's official checksums or provenance attestation before deployment.

Action Checklist

1. Step 1: Containment. Immediately identify any systems, pipelines, or CI/CD jobs that pulled KICS Docker images, the KICS VS Code extension, or the KICS Open VSX extension on April 22, 2026. Isolate affected build runners and developer workstations from cloud control planes. Revoke all cloud provider credentials (AWS IAM keys, GCP service account keys, Azure service principals), GitHub personal access tokens, and SSH keys that were accessible in any environment where KICS was executed during the exposure window within 4 hours of identification as affected. Suspend CI/CD pipeline jobs that used the ast-github-action or Developer Assist extension on that date pending investigation.
2. Step 2: Detection. Query CI/CD logs, Docker pull logs, and extension installation records for KICS artifact pulls timestamped April 22, 2026. In GitHub Actions, review workflow run logs for ast-github-action usage on that date. Check developer workstation VS Code extension histories and Docker daemon logs. Look for outbound HTTPS connections from KICS processes to external hosts not in your approved allowlist, particularly POST requests with payloads exceeding 100KB, which are consistent with credential exfiltration. Review cloud provider access logs (AWS CloudTrail, GCP Audit Logs, Azure Activity Log) for API calls using credentials that were present in affected environments, specifically calls originating from unexpected IPs or at unexpected times on or after April 22.
3. Step 3: Eradication. Pull and replace all KICS Docker images in local registries and CI/CD job definitions with artifacts confirmed clean by Checkmarx's April 22 security advisory. Remove and reinstall the KICS VS Code and Open VSX extensions only from a source confirmed by Checkmarx as unaffected. Remove the compromised versions of ast-github-action and Developer Assist from all workflows and developer environments. Verify artifact integrity using checksums or provenance attestations provided by Checkmarx before redeploying. Do not redeploy any version of these tools until the specific clean artifact identifier has been confirmed in the vendor advisory.
4. Step 4: Recovery. After rotating all secrets and replacing affected artifacts, validate that new credentials are functioning and the old credentials are fully revoked across all cloud providers and GitHub. Re-run IaC scans using verified clean tooling to confirm pipeline integrity. Monitor cloud provider access logs and GitHub audit logs for 30 days post-rotation for any residual use of revoked credentials, which would indicate the attacker retained copies. Restore CI/CD pipeline operations only after artifact replacement and secret rotation are confirmed complete.
5. Step 5: Post-Incident. Conduct a supply chain risk review of all open-source security tooling in your developer pipelines, focusing on tools that run with access to cloud credentials or secrets. Implement artifact integrity verification (e.g., cosign, SLSA provenance) for Docker images and VS Code extensions

used in CI/CD. Enforce least-privilege credential scoping so IaC scanning tools cannot access production credentials. Establish alerting on Docker Hub pulls and extension installs for security-critical tooling. Document TeamPCP's TTPs against the MITRE ATT&CK framework for threat-informed detection rule development.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and breach notification counsel immediately if AWS CloudTrail, GCP Audit Logs, or Azure Activity Log confirm that revoked credentials were used to access systems containing PII, PHI, PCI-scoped data, or regulated intellectual property after April 22, 2026, or if any residual use of revoked credentials is detected during the 30-day post-rotation monitoring window indicating active attacker retention of harvested secrets.
Recovery Notes	Recovery is not complete until every credential class accessible in affected environments — AWS IAM keys, GCP service account keys, Azure service principals, GitHub PATs, SSH keys, and any secrets injected via CI/CD secret stores (GitHub Actions secrets, HashiCorp Vault, AWS Secrets Manager) into KICS execution contexts — has been rotated and the old values confirmed revoked via API validation. Monitor AWS CloudTrail, GCP Audit Logs, Azure Activity Log, and GitHub audit logs continuously for 30 days post-rotation using CloudWatch metric filters or equivalent free alerting on the specific revoked key IDs, as TeamPCP's exfiltration mechanism had a 90-minute active window suggesting automated credential harvesting with potential delayed use. Re-enable CI/CD pipelines only after pinning all KICS and ast-github-action references to cosign-verified or SLSA-attested artifact digests, not mutable tags, to prevent re-compromise via a repeat supply chain substitution.
Forensic Artifacts	Docker daemon pull records for `checkmarx/kics` image tags pulled on April 22, 2026 — located at `journalctl -u docker` output or `/var/log/syslog` on Linux runners — will show the specific digest pulled, confirming whether the malicious layer was consumed; cross-reference the pulled digest against the clean digest published in the Checkmarx advisory. | GitHub Actions workflow run logs for jobs invoking `checkmarx/ast-github-action` on April 22 — downloadable via `gh run view --log` — will contain stdout/stderr from the action execution including any anomalous subprocess spawning or unexpected HTTP requests made by the malicious action code during the credential harvesting phase. | AWS CloudTrail `GetSecretValue`, `s3:GetObject`, and `sts:GetCallerIdentity` API call records attributed to IAM access key IDs that were present in runner environment variables during KICS execution on April 22 — filter by `sourceIPAddress` to identify calls from non-runner IP ranges that would indicate TeamPCP used harvested keys from external infrastructure. | VS Code extension directory timestamps and contents at `~/.vscode/extensions/checkmarx.kics-/` on developer workstations — the malicious extension would have introduced modified JavaScript files in the extension's `dist/` or `out/` subdirectory; preserve directory with `find ~/.vscode/extensions/checkmarx.kics-* -type f -exec sha256sum {} \;` before removal for comparison against known-good extension manifest hashes. | Network flow records or VPC Flow Logs from the build subnet covering April 22, 2026 during the 90-minute exposure window — specifically outbound TCP/443 connections from runner instance IPs to destinations outside Checkmarx, AWS, GCP, and Azure IP ranges, initiated by processes with parent ancestry tracing to the KICS container entrypoint or VS Code extension host, representing the TeamPCP exfiltration channel for harvested credentials.

Per-Action IR Details

Step 1: Containment — Immediately identify any systems, pipelines, or CI/CD jobs that pulled KICS Docker images, the KICS VS Code extension, or the KICS Open VSX extension on April 22, 2026. Isolate affected build runners and developer workstations from cloud control planes. Revoke all cloud provider credentials (AWS IAM keys, GCP service account keys, Azure service principals), GitHub personal access tokens, and SSH keys that were accessible in any environment where KICS was executed during the exposure window. Suspend CI/CD pipeline jobs that used the ast-github-action or Developer Assist extension on that date pending investigation.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), NIST AC-17 (Remote Access), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Use Docker daemon logs on each runner host to enumerate pulls: run `docker system events --since '2026-04-22T00:00:00' --until '2026-04-22T23:59:59' --filter 'type=image' --filter 'event=pull' | grep -i kics` on Linux runners. For GitHub Actions, query the REST API: `gh api /repos/{org}/{repo}/actions/runs --jq '.workflow_runs[] | select(.created_at | startswith("2026-04-22")) | {id, name, conclusion}'` then retrieve individual run logs for ast-github-action references. Block outbound traffic from isolated runners using iptables: `iptables -I OUTPUT -j DROP` before credentials are rotated. Use AWS CLI `aws iam delete-access-key --access-key-id ` and `aws iam create-access-key` per affected IAM user to rotate; for GCP, `gcloud iam service-accounts keys delete --iam-account=@.iam.gserviceaccount.com`.

Evidence: Before isolating runners, snapshot the following: full Docker daemon log (`/var/lib/docker/containers//-json.log` and `journalctl -u docker --since '2026-04-22' --until '2026-04-23'`); GitHub Actions workflow run logs for any job referencing `checkmarx/ast-github-action` on April 22 (downloadable via `gh run view --log`); VS Code extension installation log at `~/.vscode/extensions/` directory listing with timestamps (specifically `checkmarx.kics-*` or `checkmarx.ast-results-*` folder mtimes); environment variable dumps from runner process memory if runner is still live (capture via `/proc//environ` on Linux before kill); and any `.env`, `terraform.tfvars`, or CI secret injection points accessible to the KICS process at execution time.

Step 2: Detection — Query CI/CD logs, Docker pull logs, and extension installation records for KICS artifact pulls timestamped April 22, 2026. In GitHub Actions, review workflow run logs for ast-github-action usage on that date. Check developer workstation VS Code extension histories and Docker daemon logs. Look for anomalous outbound HTTP/S connections from build environments to external hosts during or after pipeline execution — particularly from processes spawned by the IaC scanner. Review cloud provider access logs (AWS CloudTrail, GCP Audit Logs, Azure Activity Log) for API calls using credentials that were present in affected environments, specifically calls originating from unexpected IPs or at unexpected times on or after April 22.

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 AU-12 (Audit Record Generation), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For Docker pull detection without SIEM: on each Linux runner run `grep -i 'kics' /var/log/syslog` and `journalctl -u docker | grep -E '(pull|2026-04-22)' | grep -i kics`. For outbound connection detection from KICS processes, use Sysmon for Linux or `ss -tnp` captured during a live runner and correlate parent PID to the KICS container. For Windows developer workstations, deploy Sysmon with the SwiftOnSecurity config and query Event ID 3 (Network Connection) where Image path contains the KICS extension host process. For AWS CloudTrail without a SIEM, use Athena ad-hoc query: `SELECT eventTime, userIdentity.arn, sourceIPAddress, eventName FROM cloudtrail_logs WHERE eventTime >= '2026-04-22T00:00:00Z' AND userIdentity.accessKeyId IN (') ORDER BY eventTime;`. For GCP, use `gcloud logging read 'timestamp>="2026-04-22T00:00:00Z" AND protoPayload.authenticationInfo.serviceAccountKeyName=~"" --format=json`.

Evidence: AWS CloudTrail events from April 22 onward for the specific IAM key IDs exposed in affected runner environments — filter for `AssumeRole`, `GetCallerIdentity`, `ListBuckets`, `DescribeInstances`, and any data-plane calls (`s3:GetObject`, `secretsmanager:GetSecretValue`) from source IPs not matching your runner egress ranges; GCP Audit Log entries under `data_access` and `activity` log buckets for the compromised service account keys; GitHub audit log entries (`https://api.github.com/orgs/{org}/audit-log?phrase=action:workflows&after=2026-04-22`) for workflow runs and any OAuth token usage anomalies post-April 22; Sysmon Event ID 3 or Linux `/proc/net/tcp` snapshots showing outbound connections from the KICS container or extension host process to non-Checkmarx, non-cloud-provider IP ranges during the 90-minute window; and network flow records (NetFlow/IPFIX or VPC Flow Logs) from build subnet egress showing DNS lookups or TLS connections to TeamPCP C2 infrastructure initiated by runner processes.

Step 3: Eradication — Pull and replace all KICS Docker images in local registries and CI/CD job definitions with artifacts confirmed clean by Checkmarx's April 22 security advisory. Remove and reinstall the KICS VS Code and Open VSX extensions only from a source confirmed by Checkmarx as unaffected. Remove the compromised versions of ast-github-action and Developer Assist from all workflows and developer environments. Verify artifact integrity using checksums or provenance attestations provided by Checkmarx before redeploying. Do not redeploy any version of these tools until the specific clean artifact identifier has been confirmed in the vendor advisory.

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-3 (Configuration Change Control), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: To verify Docker image integrity before redeployment: `docker pull checkmarx/kics: && docker inspect checkmarx/kics: --format='{{index .RepoDigests 0}}'` and compare the digest against the sha256 value published in the Checkmarx April 22 advisory. For cosign verification (free, CNCF): `cosign verify --key checkmarx/kics:`. For GitHub Actions ast-github-action, pin to the specific clean commit SHA in all workflow YAML files: `uses: checkmarx/ast-github-action@` rather than a mutable tag. For VS Code extensions, verify the VSIX file hash before install: `sha256sum checkmarx.kics-.vsix` against the value in the Checkmarx advisory. Remove poisoned Docker images from local cache: `docker rmi $(docker images checkmarx/kics --format '{{.ID}}')` and purge from any internal Harbor or ECR mirror registry.

Evidence: Before removing compromised artifacts, preserve: a bit-for-bit copy of the malicious KICS Docker image layers (`docker save checkmarx/kics: -o kics-compromised-evidence.tar`) stored to write-once or hashed storage for forensic analysis; the full contents of `~/.vscode/extensions/checkmarx.kics-/` directory archived with preserved timestamps (`tar --preserve-permissions -czf kics-ext-evidence.tar.gz ~/.vscode/extensions/checkmarx.kics-*/`); the `.github/workflows/*.yml` files from all repositories that referenced `checkmarx/ast-github-action` at the time of the April 22 runs, preserving the exact action ref used; any runner ephemeral disk image or container overlay filesystem snapshot showing files written during KICS execution; and a memory dump of any long-running KICS or Developer Assist process if still resident (`gcore ` on Linux) before termination, as TeamPCP's credential harvester may have staged exfiltrated secrets in process memory.

Step 4: Recovery — After rotating all secrets and replacing affected artifacts, validate that new credentials are functioning and the old credentials are fully revoked across all cloud providers and GitHub. Re-run IaC scans using verified clean tooling to confirm pipeline integrity. Monitor cloud provider access logs and GitHub audit logs for 30 days post-rotation for any residual use of revoked credentials, which would indicate the attacker retained copies. Restore CI/CD pipeline operations only after artifact replacement and secret rotation are confirmed complete.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST CP-10 (System Recovery and Reconstitution), NIST IA-5 (Authenticator Management), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 5.2 (Use Unique Passwords), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access)

Compensating: Validate AWS key revocation by attempting an API call with the old key: `AWS_ACCESS_KEY_ID= AWS_SECRET_ACCESS_KEY= aws sts get-caller-identity` — a successful response means revocation failed and must be retried. For GCP, verify key deletion: `gcloud iam service-accounts keys list --iam-account=` and confirm the compromised key ID no longer appears. For GitHub PAT revocation, verify via `curl -H 'Authorization: token ' https://api.github.com/user` — a 401 response confirms revocation. Set up a 30-day CloudTrail alert for the revoked key IDs using a free CloudWatch metric filter: `{ $.userIdentity.accessKeyId = "" }` with SNS notification to your team. For residual SSH key use, monitor `~/.ssh/authorized_keys` changes using auditd rule: `-w /root/.ssh/authorized_keys -p wa -k ssh_key_tamper`.

Evidence: Before restoring pipeline operations, document and retain: AWS CloudTrail `LookupEvents` output filtered to all API calls made by revoked key IDs from April 22 through rotation date, exported to JSON (`aws cloudtrail lookup-events --lookup-attributes AttributeKey=AccessKeyId,AttributeValue= --start-time 2026-04-22 --output json`); GCP Audit Log export for the compromised service account covering the same window; GitHub audit log export for the affected organization covering PAT usage from April 22 onward; a timestamped screenshot or CLI output confirming each cloud provider's revocation (AWS IAM console key status, GCP key list, Azure portal credential status); and the pipeline job execution history showing the last run of the clean artifact, as a baseline for the 30-day monitoring comparison window.

Step 5: Post-Incident — Conduct a supply chain risk review of all open-source security tooling in your developer pipelines, focusing on tools that run with access to cloud credentials or secrets. Implement artifact integrity verification (e.g., cosign, SLSA provenance) for Docker images and VS Code extensions used in CI/CD. Enforce least-privilege credential scoping so IaC scanning tools cannot access production credentials. Establish alerting on Docker Hub pulls and extension installs for security-critical tooling. Document TeamPCP's TTPs against the MITRE ATT&CK framework for threat-informed detection rule development.

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 RA-3 (Risk Assessment), NIST SA-12 (Supply Chain Protection), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Map TeamPCP's observed TTPs to MITRE ATT&CK using the free ATT&CK Navigator (https://mitre-attack.github.io/attack-navigator/): specifically layer T1195.001 (Supply Chain Compromise: Compromise Software Dependencies and Development Tools), T1552.001 (Unsecured Credentials: Credentials in Files), T1528 (Steal Application Access Token), and T1567 (Exfiltration Over Web Service). Write a Sigma rule for future KICS-style supply chain pulls: detect Docker pulls of security tooling images followed within 60 seconds by outbound HTTPS to non-expected destinations using Sysmon Event IDs 1 (Process Create) and 3 (Network Connection) correlated by ProcessId. Implement a free SLSA provenance check script in CI using `slsa-verifier verify-image` (github.com/slsa-framework/slsa-verifier) for any Docker-based security tool before first use. Restrict IaC scanner credentials to a dedicated read-only IAM role with an explicit deny on `secretsmanager:*`, `iam:*`, and `sts:AssumeRole` using an AWS SCP or inline deny policy.

Evidence: For the post-incident review, compile: the full timeline of TeamPCP's April 22 campaign reconstructed from Docker Hub pull timestamps, GitHub Actions run logs, and cloud provider audit logs, cross-referenced to establish dwell time per environment; a complete inventory of all repositories, runners, and developer workstations that had KICS or ast-github-action in scope during the exposure window (sourced from Step 2 detection queries), which defines the full blast radius for the lessons-learned report; attacker-controlled infrastructure indicators (C2 hostnames/IPs) extracted from network flow records and KICS malicious artifact behavioral analysis, formatted as STIX 2.1 for sharing via ISAC or internal threat intel platform; a gap analysis comparing pre-incident software inventory (CIS 2.1) against the actual set of open-source security tools running with credential access in pipelines, to quantify the supply chain exposure; and the before/after comparison of IAM policies for scanning tool roles, documenting least-privilege gaps that allowed KICS to access secrets it should not have reached.

Detection Guidance

Primary detection focuses on artifact pull timing and anomalous outbound traffic from build environments. Query Docker daemon logs and container registry pull records for the KICS image pulled on April 22, 2026; any pull during the ~90-minute exposure window indicates high-confidence exposure. In GitHub Actions, search workflow run logs for checkmarx/ast-github-action executions on that date. For VS Code environments, review extension installation timestamps via the extensions log or marketplace pull history. Network-layer detection: look for outbound HTTPS connections from CI/CD runners or developer machines to external hosts not in your approved allowlist, particularly POST requests with payloads exceeding 100KB consistent with credential exfiltration (T1567.001, T1071.001). In cloud provider logs, AWS CloudTrail, GCP Cloud Audit Logs, Azure Monitor Activity Log, flag API calls using credentials that were present in affected pipelines but originating from IPs outside your known CI/CD infrastructure, or calls at times inconsistent with pipeline schedules. GitHub audit logs should be reviewed for PAT usage from unexpected source IPs after April 22. IOC details (C2 infrastructure, artifact hashes) are available from the Checkmarx April 22 advisory and should be used for detection rule development. Monitor threat intelligence feeds for confirmed infrastructure indicators as additional details are published.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://hub.docker.com/r/checkmarx/kics (affected image pulled April 22, 2026)`	Official KICS Docker Hub image — malicious version distributed during ~90-minute window on April 22, 2026. Specific malicious digest not yet publicly confirmed.	HIGH
URL	`VS Code Marketplace — Checkmarx KICS extension (affected version distributed April 22, 2026)`	Malicious version of the KICS VS Code extension distributed via official marketplace during exposure window. Specific version number not yet publicly confirmed.	HIGH
URL	`Open VSX — Checkmarx KICS extension (affected version distributed April 22, 2026)`	Malicious version of the KICS Open VSX extension distributed during exposure window. Specific version number not yet publicly confirmed.	HIGH

Framework Mappings

MITRE-ATTACK

- **T1552.001** — Credentials In Files
- **T1608.001** — Upload Malware
- **T1567.001** — Exfiltration to Code Repository
- **T1554** — Compromise Host Software Binary

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1059.007** — JavaScript
- **T1078.004** — Cloud Accounts
- **T1071.001** — Web Protocols
- **T1195.002** — Compromise Software Supply Chain
- **T1552.004** — Private Keys
- **T1059** — Command and Scripting Interpreter
- **T1528** — Steal Application Access Token

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
- **IA-5** — Authenticator Management
- **CM-3** — Configuration Change Control
- **SI-2** — Flaw Remediation
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **15.1** — Establish and Maintain an Inventory of Service Providers

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

HIPAA-SECURITY

- **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

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1552.001	Credentials In Files	Credential-Access
T1608.001	Upload Malware	Resource-Development
T1567.001	Exfiltration to Code Repository	Exfiltration
T1554	Compromise Host Software Binary	Persistence
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1059.007	JavaScript	Execution
T1078.004	Cloud Accounts	Defense-Evasion
T1071.001	Web Protocols	Command-And-Control
T1195.002	Compromise Software Supply Chain	Initial-Access
T1552.004	Private Keys	Credential-Access
T1059	Command and Scripting Interpreter	Execution
T1528	Steal Application Access Token	Credential-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/new-checkmarx-supply...	T3
Malicious KICS Docker Images and VS Code Extensions ...	https://thehackernews.com/2026/04/malicious-kics-docker-images-and-...	T3

Source	URL	Tier
Checkmarx Security Update: April 22	https://checkmarx.com/blog/checkmarx-security-update-april-22/	T3
New Checkmarx supply-chain breach affects KICS analysis ...	https://www.instagram.com/p/DXevieMFeII/	T3
Checkmarx suffers second supply chain attack	https://cybernews.com/security/checkmarx-popular-tools-spread-crede...	T3

Generated 2026-04-23 18:50 UTC by TJS Security Command Center