INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-27 18:49 UTC

LAPSUS$ Claims Checkmarx Data After Supply Chain Attack Cascades Through Developer Toolchain

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0227
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Checkmarx KICS Docker image, Checkmarx GitHub Actions workflows, Open VSX plugins, Bitwarden CLI (npm package), VS Code extensions, Trivy (supply chain vector)
Published	2026-04-27T10:19:00
Discovery Source	Rss

Executive Summary

Beginning March 23, 2026, threat actors attributed to LAPSUS$ (also tracked as TeamPCP by threat intelligence sources) compromised multiple components of the Checkmarx developer security toolchain, including KICS Docker images, GitHub Actions workflows, VS Code extensions distributed via Open VSX, and the Bitwarden CLI npm package. LAPSUS$ claimed responsibility for exfiltrating Checkmarx GitHub repository content, including source code, API keys, and database credentials, and published alleged data on a dark web leak site. Any development team that executed the compromised tooling during the exposure window faces elevated risk of credential theft, code integrity compromise, and downstream supply chain contamination in their own software pipelines.

Technical Analysis

The attack chain attributed to LAPSUS$ targeted Checkmarx infrastructure across multiple vectors simultaneously. Malicious KICS Docker images were published and distributed as legitimate scanning tools; GitHub Actions workflows were tampered to execute attacker-controlled code during CI/CD pipeline runs; VS Code extensions were poisoned and pushed via the Open VSX registry. The Bitwarden CLI npm package was compromised; evidence suggests the compromise resulted from credential-stealing malware propagated through the Checkmarx toolchain, though the exact infection vector has not been publicly confirmed. Relevant CWEs: CWE-494 (Download of Code Without Integrity Check), CWE-312 (Cleartext Storage of Sensitive Information), CWE-798 (Use of Hard-coded Credentials), CWE-506 (Embedded Malicious Code). MITRE

ATT&CK techniques include T1195.001 and T1195.002 (Supply Chain Compromise), T1528 (Steal Application Access Token), T1078 (Valid Accounts), T1552.001 and T1552.004 (Credentials in Files, Private Keys), T1059 (Command and Scripting Interpreter), T1567.001 (Exfiltration to Code Repository), T1213 (Data from Information Repositories). Checkmarx confirmed unauthorized access to its GitHub repository. The company states no customer production data is stored in the affected repository, but the scope of credential exposure from compromised tooling remains unresolved. No CVE has been assigned. Checkmarx issued a security update on April 22, 2026; affected version ranges and patch details are documented in the official advisory at checkmarx.com. Specific IOC hashes and malicious image digests have not yet been published in available reporting; monitor Checkmarx's security advisory page for updated indicators as the investigation progresses. Source confidence: moderate (0.64 on 0.0-1.0 scale), pending official threat intelligence corroboration.

Action Checklist

1. Step 1: Containment. Immediately audit all CI/CD pipelines, Docker image registries, and VS Code extension installations for any Checkmarx components executed since March 23, 2026. Identify and isolate any KICS Docker images, GitHub Actions workflows, or Open VSX extensions sourced from Checkmarx channels during the exposure window. Revoke and rotate any credentials, API keys, or tokens present in environments where these components executed. Suspend Bitwarden CLI npm package usage pending integrity verification.
2. Step 2: Detection. Review CI/CD pipeline logs for unexpected outbound network connections, anomalous process execution, or unauthorized repository access events originating from KICS, GitHub Actions, or VS Code extension processes. Check npm audit logs for the Bitwarden CLI package version installed. Search secrets management systems and environment variable stores for API keys or database credentials that may have been exposed to compromised pipeline steps. Look for LAPSUS$-associated dark web activity referencing your organization's identifiers.
3. Step 3: Eradication. Replace all KICS Docker images, GitHub Actions workflow files, and Open VSX extensions with versions verified against Checkmarx's official integrity guidance from the April 22 security update. Remove and reinstall the Bitwarden CLI npm package after confirming a clean version is available. Rotate all secrets, API keys, private keys, and database credentials that were accessible in any environment where compromised components executed.
4. Step 4: Recovery. Validate that rebuilt pipeline environments produce expected checksums and exhibit no anomalous runtime behavior. Re-run security scans of your own codebase using verified, clean tooling to confirm no malicious modifications were introduced during the window of compromise. Monitor authentication logs and privileged account activity for 30 days post-remediation for signs of persistent access using stolen credentials.
5. Step 5: Post-Incident. This attack exploited the absence of cryptographic integrity verification for third-party tooling in CI/CD pipelines (CWE-494). Implement mandatory digest pinning for all Docker image pulls and dependency installs. Enforce signed commits and verified actions in GitHub workflows. Establish an approved registry or artifact proxy with integrity validation for VS Code extensions and npm packages. Review supply chain security controls against NIST SP 800-161r1 and CISA's Secure Software Development guidance.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to executive leadership, legal counsel, and potentially regulatory authorities immediately if forensic review of GitHub Actions logs or secrets manager audit trails confirms that production database credentials, customer PII, or regulated data (PCI card data, PHI, or data subject to GDPR/CCPA breach notification thresholds) were accessible to compromised KICS, Bitwarden CLI, or Open VSX extension processes during the March 23–April 22 window, or if LAPSUS$ dark web publications are confirmed to contain your organization's identifiers, source code, or credentials.
Recovery Notes	Before restoring any CI/CD pipeline to production, validate every rebuilt component against cryptographic digests published in Checkmarx's April 22 security advisory — do not rely on image tags or package version numbers alone, as these were the mutable identifiers exploited in this campaign. Re-run full SAST scans of your own codebase using verified clean KICS and alternative tooling (e.g., Semgrep OSS) to detect any malicious code modifications that compromised pipeline steps may have injected into your own repositories during the exposure window. Maintain elevated monitoring of all service account authentications, GitHub repository access events, and cloud provider API calls for a minimum of 30 days post-remediation, with specific alerting on access patterns using credentials that were rotated during eradication, as LAPSUS$ is known to cache and reuse harvested credentials weeks after initial compromise.
Forensic Artifacts	GitHub Actions workflow run logs (March 23–April 22, 2026): downloadable via `gh run list` and `gh run view --log` — primary artifact for identifying injected pipeline steps, outbound exfiltration calls, and secrets variable accesses introduced by compromised Checkmarx GitHub Actions workflows | Docker image layer filesystem (`docker save checkmarx/kics: | tar -xO | tar -tv`): layer-by-layer inspection of pulled KICS images to identify injected binaries, modified entrypoints, or added cron jobs that would constitute the malware delivery mechanism in this supply chain attack | npm package tarball integrity records (`~/.npm/_cacache/` and `package-lock.json` integrity SHA-512 fields): the forensic record of exactly which Bitwarden CLI package version and hash was installed on each developer workstation, enabling comparison against known-compromised version hashes from the advisory | VS Code extension host logs (`~/.config/Code/logs/exthost*.log` on Linux, `%APPDATA%\Code\logs\exthost*.log` on Windows): runtime behavior logs for Open VSX-sourced Checkmarx extensions that would capture credential reads, unexpected file system access, or outbound API calls made by malicious extension code | Secrets manager and environment variable access audit logs (AWS Secrets Manager CloudTrail events for `GetSecretValue`, HashiCorp Vault audit log `secret/data/*` read events, or GitHub Actions `secrets.*` access records): the definitive evidence of which credentials and API keys were read by compromised pipeline steps, defining the full blast radius of the LAPSUS$/TeamPCP credential harvesting operation

Per-Action IR Details

Step 1: Containment — Immediately audit all CI/CD pipelines, Docker image pulls, and VS Code extension installations executed since March 23, 2026. Pull and isolate any KICS Docker images, GitHub Actions workflows, or Open VSX extensions sourced from Checkmarx channels during the exposure window. Revoke and rotate any credentials, API keys, or tokens present in environments where these components executed. Suspend Bitwarden CLI npm package usage pending integrity verification.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy (CSF [RS]: Execute IR plan, categorize, contain, communicate, mitigate)

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management) — revocation of exposed credentials and API keys, NIST CM-2 (Baseline Configuration) — isolating deviations from known-good pipeline configurations, CIS 2.3 (Address Unauthorized Software) — removing unauthorized or suspect KICS Docker images and Open VSX extensions, CIS 5.1 (Establish and Maintain an Inventory of Accounts) — identifying all service accounts and tokens exposed to compromised pipeline steps

Compensating: For a 2-person team without enterprise tooling: (1) Run `docker images --filter reference='checkmarx/*'` and `docker ps -a` on all build hosts to enumerate pulled KICS images; cross-reference digests against Checkmarx's April 22 advisory hashes. (2) Query GitHub Actions run history via `gh run list --limit 200 --json conclusion,workflowName,createdAt` to surface all workflow executions since 2026-03-23. (3) Scan `.env` files, GitHub Actions secrets, and CI environment variable exports with truffleHog (`trufflehog filesystem --directory=.`) to enumerate secrets accessible during the exposure window. (4) For npm: run `npm list --depth=0 @bitwarden/cli` across all developer workstations using a simple bash loop or Ansible ad-hoc command.

Evidence: BEFORE revoking credentials or removing images, preserve the following: (1) Docker image digests (`docker inspect --format='{{index .RepoDigests 0}}' `) for every pulled Checkmarx KICS image — these are the forensic anchors for integrity comparison. (2) Full GitHub Actions workflow run logs from March 23–present, downloadable via `gh run view --log`; look specifically for `actions/checkout` steps, outbound curl/wget calls in KICS scan steps, and any `env` export commands. (3) npm install logs at `~/.npm/_logs/` and `package-lock.json` snapshots showing the exact `@bitwarden/cli` version hash installed on each affected system. (4) A snapshot of all environment variables accessible to the compromised pipeline steps — export via `printenv` executed in the same execution context before teardown.

Step 2: Detection — Review CI/CD pipeline logs for unexpected outbound network connections, anomalous process execution, or unauthorized repository access events originating from KICS, GitHub Actions, or VS Code extension processes. Check npm audit logs for the Bitwarden CLI package version installed. Search secrets management systems and environment variable stores for API keys or database credentials that may have been exposed to compromised pipeline steps. Look for LAPSUS$-associated dark web activity referencing your organization's identifiers.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis (CSF [DE]: Monitor, detect, analyze, correlate, triage adverse events)

Controls: NIST SI-4 (System Monitoring) — monitoring CI/CD runtime environments for anomalous process and network behavior, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — structured review of GitHub Actions, Docker daemon, and npm logs for indicators of LAPSUS$/TeamPCP activity, NIST IR-5 (Incident Monitoring) — tracking and documenting all pipeline executions and credential exposure events across the March 23–April 22 window, CIS 8.2 (Collect Audit Logs) — ensuring pipeline execution logs, Docker daemon logs, and npm audit logs are collected and retained, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — cross-referencing installed component versions against the compromised version list from Checkmarx's advisory

Compensating: For a 2-person team: (1) Use `falco` (free, open source) with a custom rule targeting container image names matching `checkmarx/kics*` to alert on anomalous syscalls (unexpected `connect()`, `execve()` of shells, file writes outside expected scan output paths). (2) On build hosts, run `ss -tnp` or `netstat -tnp` and cross-reference outbound connections from KICS container PIDs against expected Checkmarx scan endpoints — any connection to non-Checkmarx IPs during scan execution is a high-fidelity IOC. (3) For GitHub Actions log analysis without SIEM, use `gh api` to pull workflow run logs and pipe through `grep -E '(curl|wget|nc |ncat|/bin/sh|/bin/bash|base64 -d)'` to surface common exfiltration or shell-spawn patterns embedded in compromised workflow steps. (4) For Bitwarden CLI npm compromise detection, compare the SHA-512 hash of the installed package tarball (`npm cache verify` or manually via `sha512sum $(npm root -g)/@bitwarden/cli/package.json`) against the known-clean version hash from the npm registry integrity field.

Evidence: Preserve before analysis is complete: (1) GitHub Actions runner audit logs — specifically look for OIDC token requests, `secrets.*` variable references, and outbound HTTPS POSTs in KICS and Bitwarden CLI steps; these are the primary exfiltration vectors LAPSUS$ would have used to harvest API keys and database credentials. (2) Docker daemon logs (`/var/log/docker.log` or `journalctl -u docker`) for the exposure window, filtering on KICS image names — look for unexpected `iptables` changes or network namespace escapes. (3) VS Code extension host logs at

`~/.config/Code/logs/` (Linux) or `%APPDATA%\Code\logs\` (Windows), specifically `exthost*.log` entries for Open VSX-sourced Checkmarx extensions — malicious extensions may log credential reads or outbound API calls here. (4) npm audit output (`npm audit --json > npm_audit_snapshot.json`) and the `package-lock.json` integrity hashes for `@bitwarden/cli` on all affected developer workstations, preserved as timestamped evidence before any remediation.

Step 3: Eradication — Replace all KICS Docker images, GitHub Actions workflow files, and Open VSX extensions with versions verified against Checkmarx's official integrity guidance from the April 22 security update. Remove and reinstall the Bitwarden CLI npm package after confirming a clean version is available. Rotate all secrets, API keys, private keys, and database credentials that were accessible in any environment where compromised components executed.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication (CSF [RS]: Remove threat from environment, verify eradication)

Controls: NIST SI-2 (Flaw Remediation) — replacing compromised KICS Docker images, GitHub Actions workflows, and Bitwarden CLI npm package with integrity-verified clean versions per Checkmarx's April 22 advisory, NIST SI-7 (Software, Firmware, and Information Integrity) — verifying Docker image digests and npm package integrity hashes against Checkmarx-published values before reintroduction to pipelines, NIST AC-2 (Account Management) — rotating all service account credentials, API keys, and database credentials exposed during the March 23–April 22 window, CIS 7.2 (Establish and Maintain a Remediation Process) — executing credential rotation and component replacement on a risk-prioritized basis, starting with database credentials and production API keys, CIS 4.6 (Securely Manage Enterprise Assets and Software) — managing replacement pipeline components through version-controlled configuration to prevent re-introduction of compromised versions

Compensating: For a 2-person team: (1) Pull the clean KICS Docker image with explicit digest pinning: `docker pull checkmarx/kics@sha256:` — do not pull by tag alone, as tags are mutable and were the likely injection point. (2) Use `cosign verify` (free, Sigstore project) to validate image signatures if Checkmarx has published signed manifests in their April 22 update. (3) For Bitwarden CLI npm replacement: `npm uninstall -g @bitwarden/cli && npm cache clean --force && npm install -g @bitwarden/cli@` — verify the installed package hash with `npm ls --json @bitwarden/cli`. (4) For GitHub Actions workflow replacement, use `git diff HEAD~30 -- .github/workflows/` to compare current workflow files against pre-March 23 state and verify no unauthorized steps were introduced before replacing with Checkmarx-verified workflow content.

Evidence: Before eradicating components, preserve: (1) Full filesystem snapshot or container layer export (`docker save checkmarx/kics: -o kics_compromised_evidence.tar`) of each affected KICS image for post-incident forensic analysis — these images are the primary malware delivery vehicle and must not be deleted without preservation. (2) A copy of all GitHub Actions `.yml` workflow files in their current compromised-window state, committed to an evidence branch with `git stash` or archived — diff against Checkmarx's official workflow templates to identify injected steps. (3) npm package tarball of the compromised Bitwarden CLI version, preserved from npm cache (`~/.npm/_cacache/`) before cache flush. (4) HashiCorp Vault, AWS Secrets Manager, or equivalent audit logs showing which secrets were accessed or read by pipeline service accounts during the exposure window — this defines the full credential rotation scope.

Step 4: Recovery — Validate that rebuilt pipeline environments produce expected checksums and exhibit no anomalous runtime behavior. Re-run security scans of your own codebase using verified, clean tooling to confirm no malicious modifications were introduced during the window of compromise. Monitor authentication logs and privileged account activity for 30 days post-remediation for signs of persistent access using stolen credentials.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery (CSF [RC]: Execute recovery plan, restore systems, verify integrity, communicate)

Controls: NIST IR-4 (Incident Handling) — executing recovery consistent with the incident response plan, verifying system integrity before returning pipelines to production, NIST SI-6 (Security and Privacy Function Verification) — verifying correct operation of rebuilt CI/CD pipeline components and security scan tooling before production use, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — structured 30-day monitoring of authentication logs and

privileged account activity for credential-based persistent access by LAPSUS$, CIS 7.3 (Perform Automated Operating System Patch Management) — ensuring build host operating systems are patched as part of pipeline environment rebuild, CIS 6.2 (Establish an Access Revoking Process) — verifying that all rotated credentials from Step 3 are confirmed revoked across all downstream systems (databases, cloud providers, third-party SaaS) before pipeline restoration

Compensating: For a 2-person team: (1) Validate rebuilt pipeline integrity by running KICS scans against a known-static test repository and comparing output hashes — any deviation from expected scan results (false negatives, missing rule sets) indicates the replacement image may still be tampered. (2) Deploy a free osquery scheduled query (`SELECT * FROM process_open_sockets WHERE pid IN (SELECT pid FROM processes WHERE name='node')`) on build hosts to detect Node.js processes (VS Code extensions, Bitwarden CLI) making unexpected outbound connections post-recovery. (3) For the 30-day credential abuse monitoring window, use free SIEM alternatives: ship GitHub audit logs, AWS CloudTrail, and SSH auth logs to a self-hosted Grafana + Loki stack and create alerts on logins with rotated-credential usernames from unexpected source IPs — these patterns are characteristic of LAPSUS$ re-entry attempts using harvested credentials.

Evidence: During the recovery monitoring window, collect and retain: (1) GitHub audit log entries (`gh api /orgs/{org}/audit-log --paginate`) covering all repository access, secret reads, and workflow dispatches for 30 days post-remediation — LAPSUS$ commonly re-enters via stolen OAuth tokens or PATs rather than rotated passwords, so token-based access from unexpected IPs is the primary re-entry indicator. (2) Authentication logs for all database systems and cloud provider accounts whose credentials were accessible in compromised pipeline environments — filter on source IPs and user agents inconsistent with your normal CI/CD runner pool. (3) Checksums (`sha256sum`) of all rebuilt pipeline configuration files (workflow YAML, Dockerfile, npm lock files) at recovery completion, stored as a signed baseline for future integrity comparison.

Step 5: Post-Incident — This attack exploited the absence of cryptographic integrity verification for third-party tooling in CI/CD pipelines (CWE-494). Implement mandatory digest pinning for all Docker image pulls and dependency installs. Enforce signed commits and verified actions in GitHub workflows. Establish an approved registry or artifact proxy with integrity validation for VS Code extensions and npm packages. Review supply chain security controls against NIST SP 800-161r1 and CISA's Secure Software Development guidance.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity (CSF [GV, ID]: Lessons learned, update policies, improve detection, share intelligence)

Controls: NIST SI-7 (Software, Firmware, and Information Integrity) — implementing digest pinning, cosign signature verification, and artifact proxy enforcement to prevent recurrence of the CWE-494 supply chain injection that enabled this LAPSUS$/TeamPCP campaign, NIST SI-2 (Flaw Remediation) — establishing a documented process to rapidly identify and replace compromised third-party tooling (KICS, Bitwarden CLI, Open VSX extensions) when future supply chain advisories are issued, NIST IR-8 (Incident Response Plan) — updating the IR plan to include CI/CD supply chain compromise as a named scenario with specific detection indicators and response procedures based on lessons from this incident, NIST AU-12 (Audit Record Generation) — mandating audit logging of Docker image digest pulls, npm install integrity validation results, and GitHub Actions workflow hash verification in all pipeline environments, CIS 2.1 (Establish and Maintain a Software Inventory) — extending the software inventory to include all CI/CD pipeline components (Docker images by digest, npm packages by integrity hash, VS Code extensions by version and publisher), CIS 2.2 (Ensure Authorized Software is Currently Supported) — establishing a formal approved-tooling list for CI/CD components that requires integrity verification before authorization, preventing future use of unverified Checkmarx or similar third-party tooling, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — incorporating supply chain compromise advisories (such as Checkmarx's April 22 update) into the vulnerability management intake process with defined SLAs for CI/CD component replacement

Compensating: For a 2-person team without enterprise tooling: (1) Implement digest pinning immediately in all Dockerfiles and GitHub Actions workflow files — replace `uses: checkmarx/kics-github-action@v1` with `uses: checkmarx/kics-github-action@sha256:` and `FROM checkmarx/kics:latest` with `FROM checkmarx/kics@sha256:`. (2) Deploy a free Verdaccio npm proxy (self-hosted, open source) as an artifact mirror — configure it to validate npm package integrity hashes against the public registry before serving to CI/CD pipelines, blocking unsigned or

hash-mismatched packages like the compromised Bitwarden CLI. (3) Use the free `in-toto` framework to generate and verify software supply chain attestations for your own pipeline outputs, creating a tamper-evident record of what tooling produced each build artifact. (4) Write a Sigma rule targeting GitHub Actions runner logs for patterns consistent with this attack: process spawns from `node` or `docker` processes making outbound connections to non-allowlisted IPs during scan steps — share with the community via the Sigma HQ repository as a threat intelligence contribution.

Evidence: For the post-incident lessons learned and future detection baseline: (1) Preserve the complete incident timeline — all GitHub Actions run logs, Docker pull records, and npm install logs from March 23–April 22 — as a reference dataset for tuning future supply chain compromise detection rules. (2) Document the specific LAPSUS$/TeamPCP TTPs observed: initial access via compromised Open VSX publisher account or Checkmarx GitHub repository (MITRE ATT&CK T1195.001 — Compromise Software Dependencies and Development Tools), credential harvesting from CI/CD environment variables (T1552.001 — Credentials in Files), and data exfiltration to dark web leak site (T1567 — Exfiltration Over Web Service). (3) Archive the dark web leak site claim details (screenshots, hashes of published data if safely obtainable via threat intel feeds) to support future breach notification assessments and legal proceedings.

Detection Guidance

Focus detection on three areas. First, CI/CD pipeline telemetry: search for outbound connections from pipeline worker nodes to non-Checkmarx, non-approved endpoints during KICS or GitHub Actions execution between March 23 and April 22, 2026. Look for unexpected process spawning (T1059) within container or runner contexts. Second, credential exposure indicators: audit secrets scanning logs for any findings in repositories or pipelines that interacted with affected Checkmarx components. Check for API key or database credential reuse or authentication from unexpected IP addresses following the exposure window (T1078). Third, npm and package integrity: query npm audit logs for the specific Bitwarden CLI package version installed; compare installed package hashes against known-good hashes published by Bitwarden. For VS Code environments, review extension installation logs in the Open VSX-sourced extension directories for unexpected modifications. Specific IOC hashes, malicious image digests, and tampered workflow file signatures have not been confirmed in currently available reporting; monitor Checkmarx's official security update page and threat intelligence feeds for published indicators as the investigation matures.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://thehackernews.com/2026/04/checkmarx-confirms-github-repository.html`	The Hacker News reporting on Checkmarx GitHub repository compromise and LAPSUS$ claim — source URL, not a malicious indicator	LOW
URL	`https://checkmarx.com/blog/checkmarx-security-update-april-22/`	Official Checkmarx security update — consult for authoritative IOCs and affected version details as investigation matures	LOW

Framework Mappings

MITRE-ATTACK

- **T1528** — Steal Application Access Token
- **T1195.002** — Compromise Software Supply Chain
- **T1078** — Valid Accounts
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1567.001** — Exfiltration to Code Repository
- **T1213** — Data from Information Repositories
- **T1552.001** — Credentials In Files
- **T1552.004** — Private Keys
- **T1059** — Command and Scripting Interpreter

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
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

CIS-V8

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

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.5.21** — Managing information security in the ICT supply chain

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
- **CC6.3** — Authorizes, modifies, or removes access

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1528	Steal Application Access Token	Credential-Access
T1195.002	Compromise Software Supply Chain	Initial-Access
T1078	Valid Accounts	Defense-Evasion
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1567.001	Exfiltration to Code Repository	Exfiltration
T1213	Data from Information Repositories	Collection
T1552.001	Credentials In Files	Credential-Access
T1552.004	Private Keys	Credential-Access
T1059	Command and Scripting Interpreter	Execution

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/04/checkmarx-confirms-github-reposit...	T3
Malicious KICS Docker Images and VS Code Extensions Hit ...	https://thehackernews.com/2026/04/malicious-kics-docker-images-and-...	T3
Checkmarx Security Update: April 22	https://checkmarx.com/blog/checkmarx-security-update-april-22/	T3
Bitwarden CLI Compromised in Ongoing Checkmarx Supply Chain ...	https://www.reddit.com/r/sysadmin/comments/1stqsjm/bitwarden_cli_co...	T3

Source	URL	Tier
New Checkmarx supply-chain breach affects KICS analysis tool	https://www.bleepingcomputer.com/news/security/new-checkmarx-supply...	T3

Generated 2026-04-27 18:49 UTC by TJS Security Command Center