INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:35 UTC

Supply Chain Attacks Surge to Top Global Cyber Threat Position in 2026

SECURITY ANALYSIS | HIGH

SCC Item ID	SCC-STY-2026-0030
Type	Security Analysis
Severity	HIGH
Affected Products	Organizations globally with dependencies on third-party software vendors, SaaS platforms, open-source software packages, and hardware suppliers; estimated one-third of companies worldwide
Published	2026-03-28
Discovery Source	Gemini

Executive Summary

Supply chain attacks have emerged as the leading global cyberthreat in 2026, with a Kaspersky study reporting approximately one-third of organizations worldwide affected over the past year. Adversaries are exploiting transitive trust relationships, compromising a single upstream vendor to reach hundreds or thousands of downstream victims simultaneously, a model that renders perimeter defenses insufficient on their own. For boards and CISOs, the signal is clear: third-party risk management is no longer a compliance checkbox; it is a primary attack surface requiring continuous assurance.

Technical Analysis

Supply chain intrusion has matured from an occasional, nation-state-associated technique into a broadly adopted initial access strategy. The attack model exploits a structural vulnerability in how modern organizations consume software: trust delegated upstream is inherited implicitly by every downstream consumer, creating compounding exposure that scales with ecosystem connectivity.

Three primary intrusion vectors dominate the current landscape. First, trusted software vendor compromise, attackers insert malicious code or backdoors into legitimate software builds before distribution, as documented in the SolarWinds and 3CX case studies catalogued by Outshift/Cisco (T1195.002). Second, open-source package poisoning, adversaries publish malicious packages to public repositories (npm, PyPI) or hijack maintainer accounts to inject code into widely consumed dependencies (T1195.001, CWE-829). Third, SaaS and third-party platform integrity attacks, adversaries compromise identity providers, CI/CD pipeline tooling, or managed service provider infrastructure to establish persistent access across multiple client environments simultaneously (T1199, T1072).

The MITRE ATT&CK techniques mapped to this threat cluster, T1195 (Supply Chain Compromise) and its sub-techniques, T1554 (Compromise Client Software Binary), and T1072 (Software Deployment Tools), reflect an adversary playbook that prioritizes scale and deniability. A single upstream compromise can propagate to thousands of organizations before detection, and because the malicious payload arrives via a trusted update or package, traditional signature-based controls are bypassed by design.

Industrial Cyber reporting indicates ransomware groups are actively integrating supply chain intrusion as a preferred initial access vector, replacing more detectable methods such as phishing and direct exploitation. Industrial and critical infrastructure sectors face elevated exposure because operational technology environments frequently run legacy software with long update cycles and limited behavioral monitoring, making them attractive downstream targets once an upstream vendor is compromised.

The Hacker News analysis from April 2025 introduced an additional risk dimension: geopolitical pressures, including U.S. tariff policy, are accelerating vendor diversification decisions that often outpace security vetting, introducing new third-party relationships without corresponding due diligence.

CWE-1357 (Reliance on Insufficiently Trustworthy Component) and CWE-494 (Download of Code Without Integrity Check) identify the core defensive gaps this threat class exploits: organizations are consuming components without verifiable provenance, and software delivery pipelines frequently lack cryptographic integrity validation at the point of installation.

Note: The one-third global prevalence statistic is vendor-reported (Kaspersky) and could not be independently verified against NIST or CISA primary sources. It is cited for directional context, not as a precise measurement.

Action Checklist

1. Step 1: Assess exposure, audit your software bill of materials (SBOM) and third-party dependency inventory; identify vendors delivering code, updates, or platform access into your environment and flag any without published SBOMs or verifiable signing practices
2. Step 2: Review controls, verify cryptographic signing and integrity validation is enforced at software installation and update points (CWE-494 mitigation); confirm CI/CD pipeline access is gated by MFA and least-privilege service accounts; validate that EDR behavioral detection is deployed on build and deployment infrastructure, not only endpoints
3. Step 3: Update threat model, add T1195 (Supply Chain Compromise) sub-techniques and T1199 (Trusted Relationship) to your threat register; document which critical vendors, if compromised upstream, would provide adversaries with direct access to your environment and what the blast radius would be
4. Step 4: Communicate findings, brief leadership on third-party concentration risk: identify your top five software dependencies by blast radius, not by contract value, and present the business impact scenario for each; frame this as an active threat vector, not a theoretical risk
5. Step 5: Monitor developments, track CISA advisories and NIST NVD for disclosures involving your active vendor list; subscribe to security advisories from your top-tier software suppliers; monitor open-source package registries relevant to your stack for dependency confusion and typosquatting alerts

IR / Forensic Enrichment

Triage Priority	URGENT

Escalation Criteria	Escalate immediately to CISO and legal counsel if SBOM audit or monitoring feeds reveal that a currently-compromised vendor (per CISA advisory or NVD disclosure) has had active code execution, update delivery, or privileged remote access to production systems within the past 90 days, as this meets the threshold for a potential reportable incident under SEC cybersecurity disclosure rules (material incident) and may trigger GDPR/CCPA breach notification obligations if the vendor had access to personal data.
Recovery Notes	Post-containment recovery for a confirmed supply chain compromise requires re-establishing a trusted software baseline: rebuild affected systems from known-good images predating the vendor compromise window (not from backups that may contain compromised binaries), then re-deploy only software with cryptographically verified signatures from sources independent of the compromised vendor. Monitor all systems that received updates or code from the affected vendor for a minimum of 90 days post-recovery using enhanced Sysmon logging (Event IDs 1, 3, 7, 8, 10, 11) watching for persistence mechanisms typical of supply-chain-implanted malware: scheduled tasks, WMI subscriptions, and DLL side-loading from paths written during the compromised update window. Re-validate your entire SBOM and re-run vendor access reviews after recovery to confirm no secondary implants or lateral movement occurred via trusted relationship paths (T1199) before containment was achieved.
Forensic Artifacts	Software update agent logs and package manager audit logs (Windows: C:\ProgramData\Microsoft\Windows\WindowsUpdate\Logs\WindowsUpdate.log; Linux: /var/log/apt/history.log, /var/log/yum.log, /var/log/dnf.log) — these record exactly which vendor-supplied packages were installed and when, enabling timeline reconstruction of when a malicious update was delivered via the supply chain | Code signing verification logs and Authenticode/GPG signature records for all installed binaries — in SolarWinds-style attacks (T1195.002), the malicious DLL (e.g., SolarWinds.Orion.Core.BusinessLayer.dll) carried a valid vendor signature, making signature presence insufficient; capture both signature validity AND certificate chain and thumbprint for comparison against known-good vendor certificate hashes | CI/CD pipeline execution logs from build platforms (GitHub Actions workflow run logs, Jenkins build console output, GitLab CI job logs) — supply chain attacks targeting build infrastructure (T1195.001) inject malicious steps that appear as legitimate build jobs; look for unexpected outbound network connections from build agents (Sysmon Event ID 3) and process creation of interpreters (python.exe, node.exe, bash) spawned by build agent service accounts outside of declared pipeline steps | DNS query logs from internal recursive resolvers and endpoint DNS client logs (Windows: Event ID 3006/3020 in Microsoft-Windows-DNS-Client/Operational log; Linux: /var/log/named/ or query logs from dnsmasq/bind) — compromised supply chain components frequently beacon to adversary C2 infrastructure using domains registered to mimic legitimate vendor update endpoints; baseline comparison enables detection of new domains contacted post-update that were not present in pre-compromise DNS history | Package registry access logs and artifact repository logs (Nexus Repository, JFrog Artifactory, or AWS CodeArtifact access logs) — dependency confusion attacks (T1195.001) succeed when the artifact repository resolves a public registry package over an internal one of the same name; these logs record which registry (internal vs. public) served each package pull, identifying whether a typosquatted or dependency-confusion package was fetched during the exposure window

Per-Action IR Details

Step 1: Assess exposure — audit your software bill of materials (SBOM) and third-party dependency inventory; identify vendors delivering code, updates, or platform access into your environment and flag any without published SBOMs or verifiable signing practices

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: establishing IR capability and asset visibility before an incident occurs

Controls: NIST SI-2 (Flaw Remediation) — requires identifying and tracking software components subject to known or emerging flaws, including upstream vendor-supplied code, NIST SA-12 (Supply Chain Protection) — directs organizations to protect against supply chain threats including the use of SBOMs and provenance verification for acquired software, NIST SR-3 (Supply Chain Controls and Plans) — requires establishing controls for third-party components including integrity verification and supplier trustworthiness assessment, CIS 2.1 (Establish and Maintain a Software Inventory) — demands a detailed inventory of all licensed software, which must include vendor origin and update delivery mechanism to detect supply chain exposure, CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) — underpins SBOM work by ensuring all assets receiving third-party code or updates are enumerated

Compensating: Generate a dependency graph using free tooling: run `syft` (Anchore open-source) against container images and installed packages to produce CycloneDX or SPDX SBOMs at no cost. For Windows environments, use `winget export` and cross-reference against the CISA Known Exploited Vulnerabilities (KEV) catalog via a simple Python script hitting the CISA KEV JSON feed (https://www.cisa.gov/sites/default/files/feeds/known_exploited_vulnerabilities.json). For open-source package dependencies, run `pip-audit` (Python), `npm audit` (Node.js), or `bundler-audit` (Ruby) against your package manifests and pipe output to a CSV for manual triage by a 2-person team.

Evidence: Before modifying any inventory records, snapshot the current state: export the full installed software list per host (`Get-WmiObject -Class Win32_Product | Export-Csv` on Windows; `dpkg -l > installed_packages.txt` on Debian/Ubuntu; `rpm -qa > installed_packages.txt` on RHEL). Capture all active software update agent configurations (WSUS, Ansible, Chef, Puppet, SaltStack config files) to document what channels are trusted to push code into the environment. These baselines establish the pre-audit state and serve as forensic reference if a supply chain compromise is later discovered.

Step 2: Review controls — verify cryptographic signing and integrity validation is enforced at software installation and update points (CWE-494 mitigation); confirm CI/CD pipeline access is gated by MFA and least-privilege service accounts; validate that EDR behavioral detection is deployed on build and deployment infrastructure, not only endpoints

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: hardening systems and establishing detection capabilities to reduce the probability and impact of supply chain compromise

Controls: NIST SI-7 (Software, Firmware, and Information Integrity) — mandates integrity verification tools to detect unauthorized changes to software and firmware, directly addressing CWE-494 (Download of Code Without Integrity Check) in the update pipeline, NIST SA-10 (Developer Configuration Management) — requires developers and integrators to control and track changes to software under development, including integrity controls on build artifacts, NIST IA-2 (Identification and Authentication — Organizational Users) — requires MFA for privileged access, directly applicable to CI/CD pipeline administrative accounts, NIST AC-6 (Least Privilege) — requires restricting service account permissions in build and deployment systems to the minimum necessary, limiting blast radius if a pipeline account is compromised, NIST SI-4 (System Monitoring) — requires monitoring of build and deployment infrastructure for anomalous behavior, not only production endpoints, CIS 6.3 (Require MFA for Externally-Exposed Applications) — directly requires MFA on CI/CD platforms exposed externally (GitHub Actions, GitLab, Jenkins, CircleCI), CIS 6.5 (Require MFA for Administrative Access) — requires MFA for all administrative accounts including build system service accounts with elevated permissions

Compensating: For teams without enterprise EDR on build infrastructure: deploy Sysmon (using the SwiftOnSecurity or Florian Roth config) on all build servers to capture process creation (Event ID 1), network connections (Event ID 3), and file creation (Event ID 11) — focus rule tuning on spawning of unexpected child processes from build agents (e.g., msbuild.exe, gradle, npm spawning PowerShell or curl). Enforce signing verification by configuring Windows Software Restriction Policies or AppLocker to block unsigned binaries on build nodes. For Linux CI/CD runners, implement `debsig-verify` or GPG signature checks in your pipeline scripts before any package installation step. Use `git-secrets` and `trufflehog` (both free) to scan repositories for leaked credentials that could expose pipeline service accounts.

Evidence: Capture current CI/CD access logs before any control changes: export all active service account credentials and OAuth tokens from your pipeline platform (GitHub Actions secrets, GitLab CI variables, Jenkins credentials store — document what exists, not values). Pull the last 90 days of pipeline execution logs to establish a behavioral baseline

of normal build activity — in a supply chain attack such as SolarWinds-style (MITRE T1195.002), the malicious build step appears as a legitimate pipeline job. Review build artifact signing logs (Sigstore, Cosign, or vendor-specific logs) to identify any artifacts delivered to production without a valid signature — unsigned artifacts are the primary forensic indicator of CWE-494 exploitation in the update path.

Step 3: Update threat model — add T1195 (Supply Chain Compromise) sub-techniques and T1199 (Trusted Relationship) to your threat register; document which critical vendors, if compromised upstream, would provide adversaries with direct access to your environment and what the blast radius would be

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: developing scenario-based IR plans and maintaining a current threat model aligned to the organization's specific risk landscape

Controls: NIST RA-3 (Risk Assessment) — requires assessing risk from supply chain threats including likelihood and impact of upstream vendor compromise propagating into the organization, NIST RA-9 (Criticality Analysis) — directs identification of critical system components and their supply chain dependencies to prioritize protective measures by blast radius, NIST IR-8 (Incident Response Plan) — requires the IR plan to include scenarios relevant to the organization's threat landscape; T1195 and T1199 scenarios must be explicitly documented given 2026 threat prevalence, NIST SR-2 (Supply Chain Risk Management Plan) — requires a documented plan that identifies critical suppliers and assesses the impact of their compromise on organizational missions, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — vulnerability management scope must extend to third-party vendor components and supply chain dependencies, not only CVEs in directly-managed software

Compensating: A 2-person team can build a functional threat model using the free MITRE ATT&CK Navigator (web-based, no install required) — create a layer annotating T1195.001 (Compromise Software Dependencies and Development Tools), T1195.002 (Compromise Software Supply Chain), T1195.003 (Compromise Hardware Supply Chain), and T1199 (Trusted Relationship) with organization-specific vendor names in the notes field. Use a simple spreadsheet to map each critical vendor to: access type granted (code execution, network, data), authentication method used, and estimated number of internal systems affected if that vendor is compromised — this constitutes your blast radius register. Cross-reference your top vendors against the CISA Secure by Design advisory list and the ENISA THREAT LANDSCAPE FOR SUPPLY CHAIN ATTACKS report for adversary group associations.

Evidence: Before finalizing the threat model, collect current network flow data showing all outbound connections to vendor update endpoints and SaaS platforms (NetFlow/IPFIX exports, or `ss -tunap` snapshots on Linux, `netstat -b` on Windows) — this documents actual trusted network paths that an adversary exploiting T1199 would leverage. Review firewall and proxy logs for vendor-specific IP ranges and domains to identify undocumented or shadow-IT vendor relationships that would not appear in the official vendor register but represent live supply chain exposure.

Step 4: Communicate findings — brief leadership on third-party concentration risk: identify your top five software dependencies by blast radius, not by contract value, and present the business impact scenario for each; frame this as an active threat vector, not a theoretical risk

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: ensuring management support, defined roles, and organizational readiness to respond; communication of current threat posture to decision-makers is a prerequisite for IR resource authorization

Controls: NIST IR-8 (Incident Response Plan) — requires that the IR plan address stakeholder communication responsibilities, including leadership briefings on threat posture and resource requirements, NIST PM-9 (Risk Management Strategy) — requires communicating supply chain risk to senior leadership as part of the organizational risk management strategy, especially when a threat category reaches the prevalence described in the Kaspersky 2026 findings, NIST IR-6 (Incident Reporting) — while formally triggered post-incident, the reporting infrastructure and communication protocols established here reduce delay when an actual supply chain incident is declared, CIS 7.2 (Establish and Maintain a Remediation Process) — a risk-based remediation strategy requires leadership awareness of blast-radius-prioritized risk to secure approval and budget for remediation activities

Compensating: A 2-person team can produce a credible leadership brief using only open-source data: use the CISA KEV catalog to show how many of the organization's active vendors have previously appeared in known exploitation

events. Use public breach databases (e.g., Have I Been Pwned domain search for vendor domains, free tier) and OSINT on vendor security posture (SecurityScorecard free tier, or Shodan free tier for exposed vendor infrastructure) to substantiate blast radius claims with verifiable external evidence rather than internal estimates alone. Structure the brief around three scenarios per vendor: (1) vendor code execution path into your environment, (2) data the vendor can access, (3) detection time based on your current monitoring gaps.

Evidence: Assemble supporting evidence for the leadership brief from existing logs and records: pull the historical vendor access log — specifically privileged or administrative sessions originating from vendor IP ranges — from your VPN or remote access gateway (Cisco ASA, Palo Alto GlobalProtect, or Fortinet logs) for the past 12 months. This quantifies actual vendor access frequency and scope, converting the blast radius from a theoretical model to a data-backed risk statement. Document any existing vendor-related security incidents or near-misses from your ticketing system (ServiceNow, Jira, or even email archives) to demonstrate that this is an active, not hypothetical, exposure.

Step 5: Monitor developments — track CISA advisories and NIST NVD for disclosures involving your active vendor list; subscribe to security advisories from your top-tier software suppliers; monitor open-source package registries relevant to your stack for dependency confusion and typosquatting alerts

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: establishing continuous monitoring and intelligence feeds to detect adverse events involving supply chain dependencies; aligns to DE.AE-07 (CTI integration into adverse event analysis) and DE.CM-09 (monitoring common attack vectors)

Controls: NIST SI-5 (Security Alerts, Advisories, and Directives) — explicitly requires receiving and acting on security advisories from external organizations including CISA and software vendors, directly supporting this monitoring step, NIST SI-4 (System Monitoring) — requires monitoring systems and networks for supply-chain-relevant indicators; for this threat, monitoring must extend to package registry anomalies and upstream vendor advisory feeds, NIST IR-5 (Incident Monitoring) — requires tracking and documenting incidents; the advisory monitoring described here is the detection precursor that triggers incident tracking when a vendor in the active list is disclosed as compromised, NIST RA-10 (Threat Intelligence) — requires organizations to establish a threat intelligence capability that includes supply chain threat feeds, which at minimum encompasses CISA advisories and NVD disclosures for active vendors, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — vulnerability management must include monitoring of third-party advisory channels for vendor-specific disclosures affecting the organization's active dependency inventory, CIS 7.4 (Perform Automated Application Patch Management) — effective automated patching requires the advisory monitoring described here as its intelligence input to trigger patch cycles when upstream vendor components are affected

Compensating: A 2-person team can operationalize this monitoring without a SIEM: use the CISA KEV catalog JSON feed with a daily cron job running a Python script that cross-references vendor names against your SBOM inventory and emails alerts on matches (free, <50 lines of Python). Subscribe to osv.dev (Google's open-source vulnerability database) API to monitor CVEs affecting specific package names in your stack — filter by ecosystem (PyPI, npm, Maven, RubyGems, Go). For dependency confusion and typosquatting, configure `pip-audit --fix` in a weekly cron job and monitor PyPI/npm RSS feeds for newly published packages with names similar to your internal packages using `confused` (free CLI tool by Visma). For vendor advisory subscriptions, use a free RSS aggregator (FreshRSS self-hosted) to centralize all vendor security bulletin RSS feeds into a single daily digest reviewable by one analyst.

Evidence: To establish a monitoring baseline before this step is operationalized, capture a point-in-time snapshot of all currently installed package versions across your environment (`pip list --format=json`, `npm list --json`, `gem list`) and commit it to a private git repository — this creates a signed, timestamped record that enables diff-based detection of unauthorized package version changes, which is the primary forensic indicator of a dependency confusion attack (MITRE T1195.001) where a malicious package silently replaces a legitimate internal one. Retain DNS query logs (from your recursive resolver or endpoint DNS client logs) for the 30 days prior to this step — supply chain attacks exploiting trusted relationships (T1199) frequently use legitimate vendor domains as C2 staging points, and baseline DNS data enables retrospective detection if a vendor is later disclosed as compromised.

Detection Guidance

Detection for supply chain attacks requires shifting focus from perimeter indicators to behavioral anomalies within trusted processes.

Software integrity: Monitor for unsigned or unexpectedly signed binaries delivered via automated update channels. Alert on hash mismatches between vendor-published checksums and installed binaries. Where SBOMs are available, compare installed component hashes against known-good manifests.

CI/CD and build pipeline: Log and alert on unexpected changes to build scripts, pipeline configuration files, or deployment automation. Flag new outbound network connections originating from build servers or package management tools. Treat any privilege escalation within pipeline service accounts as high-priority.

Software deployment tools (T1072): Hunt for software deployment tools (endpoint management agents, remote monitoring tools, patch management systems) executing commands or scripts not originating from your authorized management plane. Correlate deployment tool activity with time-of-day baselines and authorized change windows.

Third-party and SaaS access: Review OAuth grants and API token usage for third-party integrations; revoke any token with excessive scope or no recent legitimate activity. Monitor for authentication events from third-party platforms outside expected business hours or from unexpected geographies.

Open-source dependencies: Implement software composition analysis (SCA) tooling in your CI pipeline to flag newly introduced packages, version changes, and packages with no prior installation history in your environment. Alert on packages that request unusual permissions at install time.

Log sources to prioritize: Endpoint process creation logs (Sysmon Event ID 1 or equivalent), software installer and update service logs, CI/CD platform audit logs, identity provider sign-in logs for service accounts, and outbound network logs from build and deployment infrastructure.

Framework Mappings

MITRE-ATTACK

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1195.002** — Compromise Software Supply Chain
- **T1199** — Trusted Relationship
- **T1195** — Supply Chain Compromise
- **T1554** — Compromise Host Software Binary
- **T1072** — Software Deployment Tools

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
- **SR-2** — Supply Chain Risk Management Plan
- **CM-3** — Configuration Change Control
- **CP-9** — System Backup

- **IR-4** — Incident Handling
- **AT-2** — Literacy Training and Awareness

OWASP-TOP10-2021

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

CIS-V8

- **2.5**
- **2.6**
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **15.1** — Establish and Maintain an Inventory of Service Providers

NIST-CSF-2

- **RS.MI-01** — Incidents are contained
- **GV.SC-01** — Cybersecurity supply chain risk management program

HIPAA-SECURITY

- **164.308(a)(7)(ii)(A)** — Data Backup Plan
- **164.312(e)(1)** — Transmission Security

ISO-27001-2022

- **A.5.29** — Information security during disruption
- **A.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

SOC2-TSC

- **CC9.2** — Manages risks associated with vendors and business partners

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1195.002	Compromise Software Supply Chain	Initial-Access
T1199	Trusted Relationship	Initial-Access
T1195	Supply Chain Compromise	Initial-Access
T1554	Compromise Host Software Binary	Persistence
T1072	Software Deployment Tools	Execution

Sources

Source	URL	Tier
The Risks of Over-Relying on Large Software Companies - Rebasoft	https://www.rebasoft.net/the-risks-of-over-relying-on-large-softwar...	T3
Top 15 software supply chain attacks: Case studies - Outshift | Cisco	https://outshift.cisco.com/blog/insights/top-10-supply-chain-attacks	T3
From Third-Party Vendors to U.S. Tariffs: The New Cyber Risks ...	https://thehackernews.com/2025/04/from-third-party-vendors-to-us-ta...	T3
Hidden threats and critical third-party vendor risks - TrustCloud	https://www.trustcloud.ai/tpra/hidden-threats-and-critical-third-pa...	T3
Software supply chain attacks surge, as ransomware groups ...	https://industrialcyber.co/reports/software-supply-chain-attacks-su...	T3

Generated 2026-03-29 18:35 UTC by TJS Security Command Center