INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-19 18:35 UTC

DPRK BlueNoroff Cluster Backdoors Axios npm Package via Stolen Credentials, Deploys Cross-Platform ZshBucket Malware

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0188
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Axios npm package (trojanized versions, March 31 2026); Node.js ecosystem; Linux, macOS, Windows; fintech and cryptocurrency sector organizations
Discovery Source	Rss:T1 Threatintel

Executive Summary

On March 31, 2026, North Korean threat actors compromised the widely used Axios npm package by stealing a maintainer's credentials and publishing two backdoored versions that deploy the ZshBucket malware across Windows, macOS, and Linux. Any organization whose software build pipeline installed the trojanized versions may have introduced a persistent, cross-platform backdoor with remote access and data exfiltration capabilities into production systems. Fintech and cryptocurrency organizations face elevated risk given BlueNoroff's established targeting pattern, but any Node.js-dependent environment is potentially affected.

Technical Analysis

Threat actor STARDUST CHOLLIMA (attributed with moderate confidence by CrowdStrike; overlapping with Google's UNC1069 and Microsoft's Sapphire Sleet, all mapping to the DPRK-nexus BlueNoroff cluster) seized a maintainer's npm credentials (CWE-522) and published two trojanized versions of the Axios npm package on March 31, 2026. Axios receives over 100,000 downloads per week, making the blast radius of a short-lived compromise significant. The malicious versions delivered ZshBucket, a cross-platform backdoor with OS-specific execution chains: PowerShell on Windows (T1059.001), Python on Linux (T1059.006), and AppleScript on macOS (T1059.002). ZshBucket supports JSON-based C2 over HTTP (T1071.001), file and directory discovery (T1083), process injection via a peinject binary (T1055), and exfiltration of system profiling data over the C2 channel (T1041). A SILKBELL component performs forensic self-cleanup to remove indicators after execution (T1070). Initial access relied on stolen credentials rather than a vulnerability in Axios itself

(T1078). The attack exploits CWE-494 (Download of Code Without Integrity Check) and CWE-829 (Inclusion of Functionality from Untrusted Control Sphere) at the package registry level. No CVE has been assigned. Affected versions are the two trojanized releases published on March 31, 2026; specific version numbers should be confirmed against the CrowdStrike and Snyk advisories listed in sources. Clean versions remain available; organizations should verify installed version integrity via npm audit and lockfile review.

Action Checklist

1. Step 1: Containment, Immediately audit all package-lock.json, yarn.lock, and pnpm-lock.yaml files across build pipelines and production environments. Identify any Axios versions published on or around March 31, 2026. Isolate systems that installed affected versions from the network pending investigation. Consult the Snyk and CrowdStrike advisories to confirm the exact trojanized version strings.
2. Step 2: Detection, Query SIEM and EDR telemetry for ZshBucket indicators: outbound HTTP C2 connections using JSON-formatted payloads from Node.js processes, unexpected PowerShell spawned from npm scripts (Windows), Python processes spawned from Node.js (Linux), and AppleScript invocations from npm lifecycle hooks (macOS). Look for peinject binary artifacts on disk. Search for SILKBELL self-cleanup behavior: file deletions correlated with npm install or postinstall script execution. Review npm install logs for the March 31, 2026 timeframe.
3. Step 3: Eradication, Remove the trojanized Axios version and replace with a verified clean release. Run 'npm install axios@latest' and pin the version in package.json. Rotate all credentials and tokens present on systems where the trojanized package executed, including npm tokens, cloud provider credentials, and any secrets accessible to the Node.js process. Reimage systems confirmed to have executed ZshBucket payloads rather than attempting in-place cleanup, given SILKBELL's forensic evasion capability.
4. Step 4: Recovery, Validate that all environments reference a clean, integrity-verified Axios version. Confirm via 'npm audit' and cross-reference package hashes against the official npm registry. Implement npm package integrity verification (package-lock.json with integrity hashes) and enforce it in CI/CD pipelines. Monitor outbound network traffic from rebuilt systems for 30 days for anomalous C2 patterns. Review secrets vaults and rotate any credentials that were in scope during the window of potential compromise.
5. Step 5: Post-Incident, Conduct a dependency risk review: enumerate all third-party npm packages with broad write access in your supply chain. Implement controls requiring MFA on npm maintainer accounts for any packages you publish or directly depend on. Introduce automated Software Composition Analysis (SCA) tooling into CI/CD pipelines to flag newly published versions before they are automatically installed. Map this incident to NIST SP 800-161 (Supply Chain Risk Management) and assess gaps against CIS Benchmark controls for build pipeline security. Consider requiring signed npm provenance for packages critical to your build process.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO, legal counsel, and external IR retainer immediately if any evidence confirms ZshBucket payload execution (child process spawning, C2 beacon, peinject artifact, or SILKBELL cleanup activity) on systems with access to cryptocurrency wallets, customer PII, payment card data, or cloud credentials with financial account access — given BlueNoroff's established pattern of targeting fintech and crypto organizations for financial theft and the CRITICAL CVSS 9.5 severity with confirmed active exploitation.
Recovery Notes	Before returning any rebuilt system to production, verify the clean Axios version integrity hash against the npm registry using 'npm view axios@ dist.integrity' and document the result. Maintain enhanced outbound network monitoring on all previously affected build and production hosts for a minimum of 30 days post-recovery, specifically watching for JSON-body HTTP POST traffic from Node.js processes to non-CDN destinations consistent with ZshBucket's C2 beacon format. For fintech and cryptocurrency organizations, additionally audit all blockchain wallet signing keys, exchange API keys, and trading platform credentials that were accessible to any compromised Node.js process during the exposure window, as BlueNoroff campaigns prioritize financial asset theft as a primary objective.
Forensic Artifacts	node_modules/axios/ directory: full filesystem forensic copy with preserved timestamps — the trojanized dispatchRequest.js or package distribution files will show modification timestamps of March 31, 2026 and a SHA-512 integrity hash mismatch against the official npm registry value for the same version string | npm install debug logs at ~/.npm/_logs/ (Linux/macOS) or %APPDATA%/npm-cache/_logs/ (Windows): contain the exact resolved Axios version, integrity hash accepted during install, and postinstall lifecycle hook invocation timestamp — primary evidence for establishing when ZshBucket was first introduced | Process execution logs (Sysmon Event ID 1 on Windows, auditd execve records on Linux, macOS Unified Log for process creation): parent-child chains showing node.exe or npm.cmd spawning PowerShell (Windows), Python (Linux), or osascript (macOS) during the npm postinstall lifecycle hook — the definitive indicator of ZshBucket cross-platform payload execution | Network flow logs and pcap captures: outbound HTTP connections from node.exe processes carrying JSON-formatted POST bodies to non-npm-registry IP addresses, consistent with ZshBucket's C2 beacon — preserve full packet capture with timestamps to establish C2 communication timeline and identify exfiltrated data scope | Memory dump from confirmed-compromised hosts captured before reimage (avml on Linux, WinPmem on Windows): SILKBELL's self-cleanup deletes on-disk artifacts post-execution, making volatile memory the only remaining source for recovering decrypted ZshBucket payload components, injected shellcode from the peinject binary, or in-memory C2 configuration containing BlueNoroff infrastructure details

Per-Action IR Details

Step 1: Containment — Immediately audit all package-lock.json, yarn.lock, and pnpm-lock.yaml files across build pipelines and production environments. Identify any Axios versions published on or around March 31, 2026. Isolate systems that installed affected versions from the network pending investigation. Consult the Snyk and CrowdStrike advisories to confirm the exact trojanized version strings.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-8 (System Component Inventory), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Run 'grep -r "axios" $(find . -name "package-lock.json" -o -name "yarn.lock" -o -name "pnpm-lock.yaml")' across all CI runner workspaces and developer laptops to identify installed Axios versions without a SIEM. Cross-reference resolved version strings against the trojanized version hashes published in the Snyk advisory.

Use 'npm ls axios' in each project root to confirm the installed version tree. Block outbound npm registry traffic at the perimeter firewall for affected build subnets immediately while investigation proceeds.

Evidence: Before isolating systems, capture: (1) full copies of all lock files (package-lock.json, yarn.lock, pnpm-lock.yaml) with timestamps preserved using 'cp --preserve=timestamps'; (2) npm install logs from CI/CD runners located at ~/.npm/_logs/ or the build system's artifact store for the March 31, 2026 window; (3) filesystem metadata on the node_modules/axios directory — 'stat node_modules/axios/lib/core/dispatchRequest.js' to confirm modification timestamps consistent with trojanized publish date; (4) running process list and open network connections at time of isolation using 'ss -tulnp' (Linux) or 'netstat -ano' (Windows) to capture any active ZshBucket C2 sessions before network cut.

Step 2: Detection — Query SIEM and EDR telemetry for ZshBucket indicators: outbound HTTP C2 connections using JSON-formatted payloads from Node.js processes, unexpected PowerShell spawned from npm scripts (Windows), Python processes spawned from Node.js (Linux), and AppleScript invocations from npm lifecycle hooks (macOS). Look for peinject binary artifacts on disk. Search for SILKBELL self-cleanup behavior: file deletions correlated with npm install or postinstall script execution. Review npm install logs for the March 31, 2026 timeframe.

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)

Compensating: On Windows: deploy Sysmon with SwiftOnSecurity config and query Event ID 1 (Process Create) for parent process 'node.exe' or 'npm.cmd' spawning 'powershell.exe' or 'cmd.exe'; query Event ID 3 (Network Connect) for 'node.exe' making outbound HTTP(S) connections to non-npm-registry destinations. On Linux/macOS: run 'auditd' with a rule '-a always,exit -F arch=b64 -S execve -F ppid=$(pgrep node)' to catch child process spawning from Node.js. Use osquery query 'SELECT pid, parent, name, cmdline FROM processes WHERE name IN ("python", "python3", "osascript", "powershell") AND parent IN (SELECT pid FROM processes WHERE name = "node")' to detect ZshBucket's cross-platform child-process behavior. Scan disk for 'peinject' binary using 'find / -name "peinject" -o -name "peinject.exe" 2>/dev/null'. For SILKBELL cleanup detection, correlate 'inotifywait' (Linux) or Sysmon Event ID 23 (File Delete) events timestamped within 60 seconds of any npm postinstall script execution.

Evidence: Capture before concluding detection phase: (1) Sysmon or auditd process tree logs showing node.exe/npm process ancestry at the time of the March 31 install — specifically parent-child chains where a postinstall hook spawned secondary processes; (2) network flow logs or pcap from the build host showing outbound JSON-body HTTP POST connections from node.exe to non-registry IPs, consistent with ZshBucket C2 beacon format; (3) disk image or forensic copy of the node_modules/axios directory before any eradication, preserving the trojanized dispatchRequest.js or equivalent payload-carrying file; (4) any remnant SILKBELL artifacts — check $TMPDIR (macOS/Linux) and %TEMP% (Windows) for staged binaries that survived the self-cleanup, and recover deleted file metadata via 'extundelete' (Linux) or Windows shadow copies; (5) npm debug logs at ~/.npm/_logs/ capturing the exact install invocation, resolved package hash, and postinstall hook execution for the affected version.

Step 3: Eradication — Remove the trojanized Axios version and replace with a verified clean release. Run 'npm install axios@latest' and pin the version in package.json. Rotate all credentials and tokens present on systems where the trojanized package executed, including npm tokens, cloud provider credentials, and any secrets accessible to the Node.js process. Reimage systems confirmed to have executed ZshBucket payloads rather than attempting in-place cleanup, given SILKBELL's forensic evasion capability.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST IA-5 (Authenticator Management), NIST CM-3 (Configuration Change Control), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 5.2 (Use Unique Passwords)

Compensating: For credential rotation without a secrets management platform: enumerate all secrets accessible to the Node.js process at install time by reviewing environment variables ('printenv' at build time captured in CI logs), .env

files, ~/.npmrc (which contains npm publish tokens), ~/.aws/credentials, and any mounted Kubernetes secrets or Docker environment injections. Revoke and reissue each: npm tokens via 'npm token revoke ' on npmjs.com, AWS keys via IAM console, GitHub tokens via Settings > Developer Settings > Personal Access Tokens. For systems where ZshBucket execution is confirmed, perform bare-metal reimage from a known-good snapshot predating March 31, 2026 rather than attempting artifact removal — SILKBELL's self-deletion makes in-place cleanup forensically unreliable.

Evidence: Before reimaging confirmed-compromised systems: (1) capture full disk image using 'dd if=/dev/sda of=/mnt/evidence/hostname.img bs=4M status=progress' or equivalent for forensic preservation; (2) dump running memory using 'avml' (Linux) or WinPmem (Windows) to recover any in-memory ZshBucket payload components or decrypted C2 config that SILKBELL may have deleted from disk; (3) export all currently valid credentials and tokens from the system's environment context to document the full blast radius before rotation; (4) pull cloud provider API call logs (AWS CloudTrail, GCP Audit Logs, Azure Monitor) for the window from March 31 onward to determine whether credentials accessible to the compromised Node.js process were used for unauthorized API calls by BlueNoroff infrastructure.

Step 4: Recovery — Validate that all environments reference a clean, integrity-verified Axios version. Confirm via 'npm audit' and cross-reference package hashes against the official npm registry. Implement npm package integrity verification (package-lock.json with integrity hashes) and enforce it in CI/CD pipelines. Monitor outbound network traffic from rebuilt systems for 30 days for anomalous C2 patterns. Review secrets vaults and rotate any credentials that were in scope during the window of potential compromise.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST SI-2 (Flaw Remediation), NIST AU-12 (Audit Record Generation), NIST CP-10 (System Recovery and Reconstitution), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Verify Axios package integrity without enterprise tooling: run 'npm view axios@ dist.integrity' to retrieve the official SHA-512 integrity hash from the npm registry and compare against the value recorded in your local package-lock.json — a mismatch confirms a tampered package. Enforce integrity checking in CI/CD by adding 'npm ci' (which enforces lock file integrity) rather than 'npm install' in all pipeline build steps. For 30-day C2 monitoring without a SIEM, configure pfSense or iptables to log all outbound connections from build and production hosts and pipe to a local syslog server; write a daily cron job using 'zeek' or Wireshark's 'tshark -r -Y "http.request.method == POST and json"' to flag JSON-body HTTP POST traffic from node.exe to non-CDN, non-registry destinations consistent with ZshBucket beacon behavior.

Evidence: During recovery validation: (1) re-run 'npm ls axios' and record the resolved version and integrity hash in a recovery verification log with timestamp and analyst signature; (2) capture a baseline network traffic profile from rebuilt systems during the first 72 hours using tcpdump or Wireshark to establish a clean outbound connection baseline for comparison against subsequent anomaly detection; (3) retrieve updated cloud provider audit logs confirming no further unauthorized API calls using previously rotated credentials, establishing a clean cutoff point for the incident timeline; (4) document the exact clean Axios version hash accepted into production as an artifact for post-incident reporting and future supply chain audits.

Step 5: Post-Incident — Conduct a dependency risk review: enumerate all third-party npm packages with broad write access in your supply chain. Implement controls requiring MFA on npm maintainer accounts for any packages you publish or directly depend on. Introduce automated Software Composition Analysis (SCA) tooling into CI/CD pipelines to flag newly published versions before they are automatically installed. Map this incident to NIST SP 800-161 (Supply Chain Risk Management) and assess gaps against CIS Benchmark controls for build pipeline security. Consider requiring signed npm provenance for packages critical to your build process.

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 SA-12 (Supply Chain Protection), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: For SCA without a commercial license: integrate 'npm audit' and OSV-Scanner (free, Google) into CI/CD pre-build steps via a GitHub Actions workflow or Jenkins shell step that fails the build on any newly published dependency version not previously approved. For npm provenance enforcement without enterprise tooling, add a pre-install script using 'npm pack --dry-run' combined with manual SHA hash verification against registry-published integrity values for the 20-30 highest-criticality packages in your dependency tree. Enforce npm MFA for maintainer accounts by navigating npmjs.com > Account Settings > Two-Factor Authentication — this is free and directly addresses the BlueNoroff credential-theft vector used in this attack. For a low-budget lessons-learned exercise, use the CISA Tabletop Exercise Package (CTEP) format to run a 2-hour supply chain scenario with the dev and security teams using this incident as the scenario basis.

Evidence: Post-incident documentation artifacts to preserve: (1) a complete dependency graph showing all npm packages installed across affected pipelines during the March 31, 2026 window, exported as 'npm ls --all --json > dependency-snapshot-20260331.json', to support root cause analysis and any regulatory notification requirements; (2) full incident timeline log correlating the BlueNoroff credential theft of the npm maintainer account, the trojanized publish event, the first internal install, and detection/containment timestamps — essential for NIST 800-61r3 §4 lessons-learned and any breach notification obligations in fintech/crypto regulatory contexts; (3) the credential rotation audit trail documenting every rotated token and the timestamp of rotation, confirming no gap exists between ZshBucket execution and credential invalidation.

Detection Guidance

Primary indicators to hunt: (1) Node.js or npm postinstall scripts spawning PowerShell (Windows), Python (Linux), or osascript/AppleScript (macOS) processes, this is abnormal behavior for a legitimate HTTP library. (2) Outbound HTTP connections carrying JSON-formatted C2 traffic originating from npm-related processes; look for structured JSON payloads containing system profiling data (hostname, OS version, username, directory listings). (3) Presence of a peinject binary on disk, particularly in temp directories or npm cache paths. (4) File deletion events correlated temporally with npm install or postinstall lifecycle hook execution (SILKBELL cleanup signature). (5) In EDR telemetry, flag process injection activity (T1055) originating from Node.js worker processes. For log sources: Windows Security Event Log (process creation Event ID 4688), Sysmon Event IDs 1 (process create), 3 (network connection), 11 (file creation), and 23 (file deletion); Linux auditd syscall logs for execve and connect; macOS Unified Log for osascript invocations. Network-level: inspect proxy and firewall logs for HTTP POST requests with JSON bodies from developer workstations or CI/CD runners around March 31 through April 2026. IOC enrichment should reference the CrowdStrike advisory for confirmed C2 infrastructure indicators, as specific IP addresses and domains were not included in the structured data provided for this item.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`Not available in source data — refer to CrowdStrike advisory for confirmed ZshBucket binary hashes`	ZshBucket malware variants across Windows, macOS, and Linux payloads	LOW

Type	Value	Context	Confidence
DOMAIN	`Not available in source data — refer to CrowdStrike advisory for confirmed C2 infrastructure domains`	STARDUST CHOLLIMA C2 infrastructure used for JSON-based command and control	LOW
URL	`https://www.crowdstrike.com/en-us/blog/stardust-chollima-likely-compromises-axios-npm-package/`	CrowdStrike primary advisory — source for confirmed IOCs including hashes, domains, and version identifiers	HIGH
URL	`https://snyk.io/blog/axios-npm-package-compromised-supply-chain-attack-delivers-cross-platform/`	Snyk advisory — source for affected version identifiers and npm package integrity guidance	HIGH

Framework Mappings

MITRE-ATTACK

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1083** — File and Directory Discovery
- **T1055** — Process Injection
- **T1078** — Valid Accounts
- **T1027** — Obfuscated Files or Information
- **T1071.001** — Web Protocols
- **T1059** — Command and Scripting Interpreter
- **T1070** — Indicator Removal
- **T1571** — Non-Standard Port
- **T1059.001** — PowerShell
- **T1543** — Create or Modify System Process
- **T1059.006** — Python
- **T1059.002** — AppleScript
- **T1041** — Exfiltration Over C2 Channel
- **T1195.002** — Compromise Software Supply Chain

NIST-800-53R5

- **AC-6** — Least Privilege
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AC-2** — Account Management
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management

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

OWASP-TOP10-2021

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

CIS-V8

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

HIPAA-SECURITY

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

SOC2-TSC

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

ISO-27001-2022

- **A.5.34** — Privacy and protection of personal information
- **A.5.21** — Managing information security in the ICT supply chain

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1083	File and Directory Discovery	Discovery

Technique ID	Technique Name	Tactic
T1055	Process Injection	Defense-Evasion
T1078	Valid Accounts	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1071.001	Web Protocols	Command-And-Control
T1059	Command and Scripting Interpreter	Execution
T1070	Indicator Removal	Defense-Evasion
T1571	Non-Standard Port	Command-And-Control
T1059.001	PowerShell	Execution
T1543	Create or Modify System Process	Persistence
T1059.006	Python	Execution
T1059.002	AppleScript	Execution
T1041	Exfiltration Over C2 Channel	Exfiltration
T1195.002	Compromise Software Supply Chain	Initial-Access

Sources

Source	URL	Tier
Blog	https://www.crowdstrike.com/en-us/blog/stardust-chollima-likely-com...	T3
	https://thehackernews.com/2026/04/google-attributes-axios-npm-suppl...	T3
	https://cybersecuritynews.com/north-korea-linked-hackers-compromise...	T3
	https://backendnews.net/crowdstrike-stolen-credentials-used-in-axio...	T3
Axios npm Package Compromised: Supply Chain Attack ... - Snyk	https://snyk.io/blog/axios-npm-package-compromised-supply-chain-att...	T3

Generated 2026-04-19 18:35 UTC by TJS Security Command Center