INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-18 19:04 UTC

Malicious AI-Themed Plugins and Extensions Targeting Developer Credentials and AI API Keys

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0513
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	JetBrains IDEs (all versions with plugin marketplace access), Google Chrome (all versions with extension support)
Published	2026-06-17
Discovery Source	Gemini

Executive Summary

Threat actors are distributing malicious plugins and browser extensions disguised as AI assistant tools through plugin and extension distribution channels, targeting developers' AI API keys and session credentials. Organizations with developers using JetBrains IDEs or Chrome-based AI tooling face direct risk of AI platform credential theft, which can lead to unauthorized API usage charges, exposure of proprietary code shared in AI sessions, and a foothold for broader network intrusion. Discovery is sourced from secondary intelligence; specific attribution and full technical indicators remain unverified from Tier 1 sources at time of publication.

Technical Analysis

Security researchers have identified two potential attack vectors through which threat actors could distribute malicious software: (1) JetBrains plugin marketplace, where malicious plugins impersonating AI assistant tools could operate under elevated IDE permissions, enabling access to environment variables, .env files, configuration files, and project source where API keys are commonly stored in plaintext; (2) Chrome extension distribution, where malicious extensions could leverage browser-level permissions to intercept HTTPS traffic to AI chatbot platforms (OpenAI, Anthropic, Google Gemini), capturing conversation content and session tokens. Targeted credentials include AI API keys and active session tokens for major LLM platforms. MITRE ATT&CK techniques mapped: T1176 (Browser Extensions), T1195.001 (Compromise Software Supply Chain, Development Tools), T1555 (Credentials from Password Stores), T1552.001 (Unsecured Credentials, Credentials in Files), T1056.001 (Keylogging), T1567 (Exfiltration Over Web Service). CWEs present: CWE-494 (Download of Code Without Integrity Check), CWE-506 (Embedded Malicious Code), CWE-312 (Cleartext

Storage of Sensitive Information), CWE-522 (Insufficiently Protected Credentials), CWE-200 (Exposure of Sensitive Information). No CVE ID is assigned. CVSS base score 7.5 (High). No CISA KEV listing. No vendor patch available, mitigation is policy and configuration-based. Source quality is T3; specific IOCs and actor attribution are not independently verified from Tier 1 or Tier 2 sources at time of analysis.

Action Checklist

1. Step 1: Containment. Immediately audit all JetBrains plugins installed across developer workstations; remove any AI-assistant plugins not explicitly approved by your security or engineering leadership. Revoke and rotate all AI API keys (OpenAI, Anthropic, Google Gemini, and other LLM providers) for any developer whose environment includes or included unreviewed plugins. Review Chrome extensions installed on developer endpoints and remove unapproved AI-themed extensions. Reference: NIST AC-6 (Least Privilege), restrict plugin installation permissions to approved-only lists.
2. Step 2: Detection. Query endpoint logs and IDE activity logs for plugin installation events from non-allowlisted sources. In Chrome enterprise environments, review chrome://extensions/ or MDM extension inventories for AI-themed extensions with broad host permissions (access to all sites, clipboard, or storage). Search developer workstations and CI/CD environments for .env files and configuration files containing API keys in plaintext (CWE-312, CWE-522). Monitor AI platform billing dashboards (OpenAI, Anthropic, Gemini) for unexpected API usage spikes indicating key abuse. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 8.2 (Collect Audit Logs).
3. Step 3: Eradication. Enforce an approved plugin and extension allowlist through JetBrains IDE settings (Settings > Plugins) and Chrome enterprise policy (ExtensionInstallAllowlist / ExtensionInstallBlocklist). Remove all unapproved AI-themed plugins and extensions and reimage workstations where confirmed-malicious plugins were present. Migrate API keys from plaintext .env files and config files to secrets management solutions (e.g., HashiCorp Vault, AWS Secrets Manager, environment-level secret injection via CI/CD). Reference: NIST CM (Configuration Management), CIS 2.3 (Address Unauthorized Software).
4. Step 4: Recovery. After key rotation, validate new keys function correctly in all affected pipelines and developer workflows. Monitor AI platform account activity for 30 days post-rotation for residual unauthorized usage. Confirm plugin allowlists are enforced and not bypassable by developer-local settings. Verify endpoint security tooling (EDR) covers developer workstations, not only servers. Reference: NIST IR (Incident Response), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), D3-CRO (Credential Rotation), D3-UAP (User Account Permissions).
5. Step 5: Post-Incident. Conduct a lessons-learned review covering: (a) absence of plugin installation controls (CIS 2.3, CIS 4.6, Securely Manage Enterprise Assets and Software); (b) cleartext API key storage in developer environments (CWE-312, CWE-522, implement secrets management as a required engineering standard); (c) lack of developer security awareness training specific to supply chain risks in IDE and browser ecosystems. Formalize a software supply chain security policy that includes extension and plugin vetting as a mandatory pre-approval step. Reference: NIST AC-6 (Least Privilege), CIS 2.1 (Establish and Maintain a Software Inventory), D3-CH (Credential Hardening).

IR / Forensic Enrichment

Triage Priority	URGENT

Escalation Criteria	Escalate immediately to CISO and legal counsel if AI provider usage logs confirm unauthorized API calls occurred before key rotation (indicating active key abuse), if exfiltrated keys were scoped to organizational accounts with access to proprietary code or customer data shared in AI sessions, or if any affected developer workstation has access to production CI/CD pipelines — any of these conditions triggers potential data breach notification assessment under applicable privacy regulations.
Recovery Notes	After completing key rotation across all affected AI platforms, validate each new key against its associated pipeline in a non-production environment before re-enabling production workflows to avoid service disruption. Monitor AI provider billing and usage dashboards daily for the first 14 days and weekly through day 30, specifically watching for API calls from source IPs outside your known developer egress ranges, which would indicate a key was exposed through a vector not yet identified. Do not consider recovery complete until plugin allowlist enforcement is confirmed non-bypassable at the OS or MDM policy level — developer-local JetBrains settings alone are insufficient, as the same social engineering that delivered the initial malicious plugin could deliver a policy override.
Forensic Artifacts	JetBrains IDE plugin cache directories (`~/.cache/JetBrains//plugins/` on Linux/macOS, `%LOCALAPPDATA%\JetBrains\\plugins\` on Windows) — contain the malicious plugin JAR or ZIP with embedded credential-harvesting code; file creation timestamps establish when the plugin was installed relative to any observed API key abuse. | Chrome extension unpacked source files at `~/.config/google-chrome/Default/Extensions//` — specifically `background.js`, `content_scripts`, and `manifest.json`; the manifest `permissions` array and `host_permissions` fields reveal the intended exfiltration scope, and background script logic shows the exact credential-harvesting mechanism (clipboard monitoring, localStorage scraping, or network interception). | AI provider API usage logs from OpenAI (`/v1/usage`), Anthropic, and Google Cloud console — contain per-key call timestamps, source IPs, model endpoints, and token volumes; cross-referencing these against developer workstation egress IPs and the plugin installation timestamp establishes whether exfiltrated keys were used and for how long before rotation. | JetBrains `idea.log` (full path: `~/.cache/JetBrains//log/idea.log`) and Chrome `chrome_debug.log` — contain plugin installation events, HTTP requests initiated during plugin initialization, and any JavaScript console errors from malicious extension execution; these provide a timeline anchor for when the malicious component became active on each endpoint. | Developer repository and CI/CD environment variable stores (`.env` files, `docker-compose.yml`, GitHub Actions secrets audit log, GitLab CI variable history) — malicious plugins that exfiltrate via file system scanning rather than network interception leave access timestamps on plaintext credential files; the CI/CD audit logs also reveal whether stolen keys were used to authenticate to pipeline infrastructure beyond direct AI platform abuse.

Per-Action IR Details

Step 1: Containment — Immediately audit all JetBrains plugins installed across developer workstations; remove any AI-assistant plugins not explicitly approved by your security or engineering leadership. Revoke and rotate all AI API keys (OpenAI, Anthropic, Google Gemini, and other LLM providers) for any developer whose environment included unreviewed plugins. Review Chrome extensions installed on developer endpoints and remove unapproved AI-themed extensions. Reference: NIST AC-6 (Least Privilege) — restrict plugin installation permissions to approved-only lists.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: Stop ongoing credential exfiltration by eliminating the malicious plugin/extension vector and invalidating any keys already exposed before broader eradication actions begin.

Controls: NIST AC-6 (Least Privilege), NIST AC-2 (Account Management), CIS 6.2 (Establish an Access Revoking Process)

Compensating: On each developer workstation, run: `ls ~/.config/JetBrains/*/plugins/` (Linux/macOS) or `dir %APPDATA%\JetBrains\*\plugins` (Windows) to enumerate installed plugins and cross-reference against an approved list. For Chrome, use `jq` against the Extensions directory at `~/.config/google-chrome/Default/Extensions/` to extract manifest.json fields (name, permissions) and flag any extension declaring ``, clipboard, or storage permissions. Immediately revoke API keys via each provider's dashboard (platform.openai.com, console.anthropic.com, console.cloud.google.com) — no tooling required.

Evidence: BEFORE revoking any API keys or removing plugins, capture: (1) full plugin directory listings with file timestamps (`ls -la` or `dir /T:C`) to establish installation timeline; (2) JetBrains IDE log at `~/.cache/JetBrains//log/idea.log` (Linux/macOS) or `%APPDATA%\JetBrains\\log\idea.log` (Windows) for plugin load events and any outbound connection attempts initiated by the plugin; (3) Chrome extension manifest files (`manifest.json`) and any `background.js` or service worker scripts from the extension directory before removal, as these contain the exfiltration logic; (4) network connection state — run `netstat -ano` (Windows) or `ss -tnp` (Linux) to capture any live C2 or exfiltration connections the malicious plugin may have open at the moment of discovery; (5) browser localStorage and IndexedDB snapshots if the malicious Chrome extension stored harvested keys locally before transmission.

Step 2: Detection — Query endpoint logs and IDE activity logs for plugin installation events from non-allowlisted sources. In Chrome enterprise environments, review chrome://extensions/ or MDM extension inventories for AI-themed extensions with broad host permissions (access to all sites, clipboard, or storage). Search developer workstations and CI/CD environments for .env files and configuration files containing API keys in plaintext (CWE-312, CWE-522). Monitor AI platform billing dashboards (OpenAI, Anthropic, Gemini) for unexpected API usage spikes indicating key abuse. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 8.2 (Collect Audit Logs).

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: Correlate IDE plugin installation telemetry, Chrome extension permission profiles, plaintext credential exposure in developer repositories, and AI platform billing anomalies to establish scope of compromise and identify all affected developer identities.

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-2 (Event Logging), CIS 8.2 (Collect Audit Logs)

Compensating: Use `grep -r 'OPENAI_API_KEY\|ANTHROPIC_API_KEY\|GEMINI_API_KEY\|sk-' /home/ /root/ /srv/ --include='*.env' --include='*.json' --include='*.yaml' --include='*.toml' 2>/dev/null` to locate plaintext keys on Linux CI/CD nodes; equivalent PowerShell: `Get-ChildItem -Recurse -Include *.env,*.json,*.yaml | Select-String -Pattern 'sk-|OPENAI|ANTHROPIC|GEMINI'`. Deploy Sysmon with EventID 3 (Network Connection) rules to detect outbound connections from JetBrains IDE processes (idea64.exe, idea.exe, pycharm64.exe) to non-JetBrains infrastructure. Write a Sigma rule matching Sysmon EventID 11 (File Create) for new files written under the JetBrains plugin directory by processes other than the IDE itself, which would indicate a malicious plugin dropping a secondary payload.

Evidence: Before any remediation alters endpoint state, collect: (1) JetBrains `idea.log` entries containing 'Plugin installed', 'Downloaded plugin', or HTTP requests to non-jetbrains.com domains initiated during plugin load; (2) Sysmon EventID 3 records showing outbound TCP connections from IDE processes to external IPs, especially on ports 443 or 80, time-correlated with plugin installation timestamps; (3) Chrome extension network requests captured via browser DevTools HAR export or Wireshark on the developer workstation NIC, specifically looking for POST requests carrying base64-encoded or JSON-serialized credential blobs; (4) CI/CD pipeline environment variable dumps or build logs that may have echoed API keys to stdout, stored in artifact logs on Jenkins, GitHub Actions, or GitLab CI; (5) AI provider usage logs from platform dashboards showing API call timestamps, source IPs, and model endpoints — these establish whether keys were already abused and provide an exfiltration timeline independent of endpoint evidence.

Step 3: Eradication — Enforce an approved plugin and extension allowlist through JetBrains IDE settings (Settings > Plugins) and Chrome enterprise policy (ExtensionInstallAllowlist / ExtensionInstallBlocklist).

Remove all unapproved AI-themed plugins and extensions and reimage workstations where confirmed-malicious plugins were present. Migrate API keys from plaintext .env files and config files to secrets management solutions (e.g., HashiCorp Vault, AWS Secrets Manager, environment-level secret injection via CI/CD). Reference: NIST CM (Configuration Management), CIS 2.3 (Address Unauthorized Software).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: Remove malicious plugin and extension artifacts from all developer endpoints, enforce allowlist controls to prevent re-infection, and eliminate the plaintext credential storage pattern that enabled initial key exposure.

Controls: NIST CM-7 (Least Functionality), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: For JetBrains allowlist enforcement without enterprise MDM: use the JetBrains Plugin Management API or push a `plugin_restrictions.xml` file to `~/.config/JetBrains//` specifying `` entries only. For Chrome without Google Workspace admin: deploy a `managed_preferences` JSON policy file under `/etc/chromium/policies/managed/` (Linux) or via registry at `HKLM\SOFTWARE\Policies\Google\Chrome\ExtensionInstallAllowlist` (Windows) — both are free and require no commercial tooling. To migrate plaintext keys from .env files, use Doppler (free tier), HashiCorp Vault OSS, or GitHub Actions encrypted secrets — all provide secret injection at runtime without persisting keys to disk.

Evidence: BEFORE reimaging any confirmed-malicious workstation, capture a full forensic disk image using `dc3dd` or `dd` with hash verification, plus a live memory acquisition using Volatility-compatible tools (LiME for Linux, WinPmem for Windows) to preserve any in-memory API key material the malicious plugin may have staged. Specifically preserve: (1) the malicious plugin's JAR or ZIP archive from the JetBrains plugin cache directory (`~/.cache/JetBrains//plugins/` or `%LOCALAPPDATA%\JetBrains\\plugins\`) — this is the primary malware artifact and must be hashed and retained for analysis before the reimage destroys it; (2) Chrome extension unpacked directory contents including `background.js`, `content_script.js`, and any `wasm` or obfuscated JS files; (3) browser credential store (`Login Data` SQLite file at `~/.config/google-chrome/Default/Login Data`) which may show what additional credentials the extension accessed beyond AI API keys.

Step 4: Recovery — After key rotation, validate new keys function correctly in all affected pipelines and developer workflows. Monitor AI platform account activity for 30 days post-rotation for residual unauthorized usage. Confirm plugin allowlists are enforced and not bypassable by developer-local settings. Verify endpoint security tooling (EDR) covers developer workstations, not only servers. Reference: NIST IR (Incident Response), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), D3-CRO (Credential Rotation), D3-UAP (User Account Permissions).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: Restore developer productivity using verified-clean environments and rotated credentials, with active monitoring of AI platform usage telemetry to detect any residual unauthorized access from keys exposed prior to rotation.

Controls: NIST AC-2 (Account Management), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 6.3 (Require MFA for Externally-Exposed Applications)

Compensating: For teams without EDR, deploy Sysmon on all developer workstations using a configuration that logs EventID 3 (network connections from IDE processes), EventID 7 (image/DLL loads into IDE processes), and EventID 11 (file creation in plugin directories). Forward Sysmon logs to a centralized syslog server using Winlogbeat (free). For AI platform monitoring without a SIEM, write a daily cron job that queries the OpenAI usage API (`GET https://api.openai.com/v1/usage`) and Anthropic's equivalent, diffs against a baseline, and emails an alert on anomalies exceeding a defined threshold.

Evidence: During the recovery monitoring window, preserve: (1) AI platform API usage logs exported daily from provider dashboards — retain source IP, timestamp, model, and token counts for each API call to detect usage from IPs outside your known developer egress ranges, which would indicate a key rotated too late or a second exfiltration path not yet identified; (2) JetBrains plugin installation audit trail from IDE logs and any MDM telemetry to confirm no

re-installation of blocklisted plugins occurs post-recovery; (3) developer workstation DNS query logs (from local resolver or network DNS) for domains contacted by JetBrains or Chrome processes, as malicious plugins often use DNS-based C2 or exfiltration that survives credential rotation if the plugin itself was not fully removed.

Step 5: Post-Incident — Conduct a lessons-learned review covering: (a) absence of plugin installation controls (CIS 2.3, CIS 4.6 — Securely Manage Enterprise Assets and Software); (b) cleartext API key storage in developer environments (CWE-312, CWE-522 — implement secrets management as a required engineering standard); (c) lack of developer security awareness training specific to supply chain risks in IDE and browser ecosystems. Formalize a software supply chain security policy that includes extension and plugin vetting as a mandatory pre-approval step. Reference: NIST AC-6 (Least Privilege), CIS 2.1 (Establish and Maintain a Software Inventory), D3-CH (Credential Hardening).

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Document the gap between JetBrains Marketplace and Chrome Web Store trust assumptions versus actual vetting rigor, update the software supply chain policy to treat IDE plugins and browser extensions as third-party software requiring the same approval gate as production dependencies, and feed IOCs from the malicious plugin artifacts into detection engineering.

Controls: NIST AC-6 (Least Privilege), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Create a YARA rule targeting the malicious plugin's JAR or JS artifacts (pattern-match on obfuscated credential-harvesting strings or C2 domain literals extracted during forensic analysis) and run it weekly via `yara -r rule.yar /home/` and against CI/CD artifact stores. Publish an internal developer advisory that specifically names the AI-themed plugin/extension category as a social-engineering lure, includes screenshots of the observed malicious plugin UI, and provides a checklist for evaluating any future AI assistant tool before installation. Add plugin and extension inventory to the quarterly asset review cycle alongside software inventory under CIS 2.1.

Evidence: Retain and archive for the lessons-learned record: (1) the complete malicious plugin JAR/ZIP and Chrome extension directory with cryptographic hashes (SHA-256) — these are the primary evidence artifacts for policy justification and potential law enforcement referral; (2) a timeline reconstructed from IDE logs, Sysmon events, and AI provider usage logs showing the interval between plugin installation and first observed unauthorized API call — this gap measurement directly informs detection SLA targets for the updated playbook; (3) a full list of all developer accounts, API key identifiers (not values), and AI platform organizations that had exposure, documented for breach notification assessment and to scope any required customer or partner notification obligations.

Detection Guidance

Detection focuses on three surfaces: plugin/extension inventory, credential exposure, and API key abuse. (1) Plugin inventory: Query MDM or endpoint management tools for JetBrains plugin installation logs; flag any AI-assistant plugin not on an approved list. JetBrains stores plugin data in the IDE configuration directory (e.g., ~/.config/JetBrains/ on Linux, %APPDATA%\JetBrains\ on Windows), enumerate installed plugins across the fleet and compare against an approved baseline. (2) Chrome extensions: Use Chrome enterprise reporting or endpoint MDM to inventory extensions by extension ID; flag extensions with host_permissions set to broad patterns (e.g., '' or specific AI platform domains) that were not centrally deployed. (3) Credential exposure: Search developer workstations and repositories for hardcoded API key patterns, regex patterns for OpenAI (sk-[A-Za-z0-9]{48}), Anthropic (sk-ant-[A-Za-z0-9-_]{95}), and Google Gemini (AIza[A-Za-z0-9_\-]{35}) keys. Scan .env, .yaml, .json, and .config files. (4) API abuse: Enable usage monitoring and alerting on all AI platform accounts; anomalous spikes in token consumption or requests from unfamiliar IP addresses are behavioral indicators of key theft and active use. D3-LAM (Local Account Monitoring) and D3-SFA (System File Analysis) are applicable countermeasures. No confirmed IOC hashes, domains, or IPs are available from verified sources at time of analysis. Defenders should focus on behavioral indicators (plugin inventory, credential patterns, API

usage anomalies) rather than file-based signatures.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`not available`	No confirmed malware hashes for malicious JetBrains plugins identified in verified sources at time of analysis	LOW
DOMAIN	`not available`	No confirmed exfiltration domains identified in verified Tier 1 or Tier 2 sources at time of analysis	LOW

Framework Mappings

MITRE-ATTACK

- **T1567** — Exfiltration Over Web Service
- **T1176** — Software Extensions
- **T1555** — Credentials from Password Stores
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1552.001** — Credentials In Files
- **T1056.001** — Keylogging

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures
- **A01:2021** — Broken Access Control

NIST-800-53R5

- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **IA-5** — Authenticator Management
- **AC-3** — Access Enforcement
- **SC-28** — Protection of Information at Rest
- **SR-2** — Supply Chain Risk Management Plan

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(a)(1)** — Access Control
- **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

ISO-27001-2022

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1567	Exfiltration Over Web Service	Exfiltration
T1176	Software Extensions	Persistence
T1555	Credentials from Password Stores	Credential-Access
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1552.001	Credentials In Files	Credential-Access
T1056.001	Keylogging	Collection

Sources

Source	URL	Tier
Security Vulnerabilities Study in Software Extensions and Plugins	https://eunomia.dev/blog/2025/02/10/security-vulnerabilities-study-...	T3
Fixed security issues - JetBrains	https://www.jetbrains.com/privacy-security/issues-fixed/	T3
How safe are intellij plugins : r/IntelliJIDEA - Reddit	https://www.reddit.com/r/IntelliJIDEA/comments/1e7f4bv/how_safe_are...	T3

Source	URL	Tier
IDE Extensions Pose Hidden Risks to Software Supply Chain	https://www.darkreading.com/application-security/ide-extensions-ris...	T3
IDE extensions threaten the software supply chain - Techzine Global	https://www.techzine.eu/news/security/132750/ide-extensions-threate...	T3

Generated 2026-06-18 19:04 UTC by TJS Security Command Center