INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-26 18:40 UTC

CVE-2026-12957: Amazon Q Developer MCP Trust Gap Enables Credential Theft via Malicious Repository Config

CVE VULNERABILITY | HIGH | CVSS 7.5

SCC Item ID	SCC-CVE-2026-0356
Type	CVE Vulnerability
CVE ID	CVE-2026-12957, CVE-2026-12958, CVE-2025-59536, CVE-2025-54136, CVE-2026-30615
Severity	HIGH
CVSS Base Score	7.5
EPSS Score	0.0012 (2th percentile)
Affected Products	Amazon Q Developer (VS Code plugin ≥2.20, JetBrains ≥4.3, Eclipse ≥2.7.4, Visual Studio ≥1.94.0.0); Language Servers for AWS (fixed in 1.69.0)
Published	2026-06-26T09:53:00
Discovery Source	Rss

Executive Summary

A high-severity flaw in Amazon Q Developer, an AI coding assistant used across major IDEs, allows a malicious code repository to steal AWS cloud credentials automatically when a developer opens and trusts the workspace. No additional user interaction is required beyond opening the project; a single crafted configuration file can exfiltrate active AWS session tokens to an attacker-controlled server. Organizations with developers using Amazon Q Developer face direct risk of AWS account compromise, including unauthorized access to cloud infrastructure, data stores, and production environments.

Technical Analysis

CVE-2026-12957 (CVSS 8.5) is an improper authorization/trust boundary vulnerability in the Amazon Q Developer plugin's Model Context Protocol (MCP) implementation. Affected versions: VS Code plugin ≥2.20, JetBrains ≥4.3, Eclipse ≥2.7.4, Visual Studio ≥1.94.0.0; Language Servers for AWS (fixed in 1.69.0). Root cause: the MCP implementation fails to adequately restrict or validate process execution contexts when handling locally-sourced tool definitions (CWE-863: Incorrect Authorization; CWE-862: Missing Authorization), permitting OS command injection via MCP stdio channels (CWE-78). A related path/symlink traversal issue (CWE-61) may further expand the attack surface. Wiz Research demonstrated a proof-of-concept in which a single crafted repository config file spawns AWS CLI processes and exfiltrates the active AWS session token

without any additional user prompt after workspace trust is granted. The attack chain maps to MITRE ATT&CK T1195.001 (Compromise Software Supply Chain), T1552/T1552.001 (Unsecured Credentials/Credentials in Files), T1059/T1059.007 (Command and Scripting Interpreter/JavaScript), T1106 (Native API), T1078.004 (Valid Accounts: Cloud Accounts), T1548 (Abuse Elevation Control Mechanism), and T1190 (Exploit Public-Facing Application). Amazon released a patch on May 12, 2026, ahead of coordinated public disclosure on June 26, 2026. A related cluster includes CVE-2026-12958, CVE-2025-59536, CVE-2025-54136, and CVE-2026-30615. A related vulnerability, CVE-2026-39861, documents an analogous RCE flaw in Anthropic Claude Code, confirming a systemic MCP trust architecture problem across AI coding assistants. Not listed on CISA KEV as of configuration date.

Action Checklist

1. Containment: Immediately restrict or disable Amazon Q Developer plugin use in developer workstations until the patched Language Servers for AWS version 1.69.0 (or later) is confirmed deployed. Revoke and rotate any AWS IAM session credentials (temporary tokens, access keys) associated with developer workstations that may have opened untrusted repositories with Amazon Q Developer enabled. Isolate affected developer machines from production AWS environments pending credential rotation.

2. Detection: Query AWS CloudTrail for anomalous AssumeRole, GetSessionToken, or STS API calls originating from developer workstation IP addresses, particularly calls followed by data exfiltration patterns (S3 GetObject, Lambda invocations, or outbound API calls to unknown endpoints). Review IDE plugin logs and process execution logs for unexpected AWS CLI invocations spawned by the Amazon Q Developer plugin process. Search endpoint logs for MCP stdio channel process trees spawning aws-cli or curl/wget with AWS credential environment variables. Check for outbound connections from developer endpoints to non-corporate, non-AWS IP addresses following repository open events.

3. Eradication: Upgrade all Amazon Q Developer plugin instances to the patched versions: VS Code plugin, JetBrains, Eclipse, and Visual Studio IDE plugins released after May 12, 2026, and confirm Language Servers for AWS is at version 1.69.0 or later. Remove or audit all repository-level MCP configuration files (.mcp.json or equivalent) in developer workspaces. Disable MCP tool execution in Amazon Q Developer settings until organizational policy for trusted MCP sources is defined and enforced.

4. Recovery: After credential rotation and patching, validate that no new unauthorized IAM users, roles, or access keys were created during the exposure window using AWS IAM Access Analyzer and CloudTrail. Re-enable Amazon Q Developer only on workstations confirmed to be running patched plugin versions. Establish a process for developers to report suspicious repository config files. Monitor AWS environments for 30 days post-remediation for residual unauthorized activity using AWS GuardDuty and CloudTrail anomaly detection.

5. Post-Incident: This vulnerability exposes a control gap in how developer tooling with AI/MCP capabilities is governed. Implement explicit organizational policy governing approved MCP server sources and workspace trust decisions (aligns with NIST AC-3: Access Enforcement and AC-6: Least Privilege). Require sandboxed execution environments for AI coding assistants handling untrusted repositories. Add Amazon Q Developer and similar MCP-enabled tools to the software inventory and vulnerability management process (CIS 2.1: Establish and Maintain a Software Inventory; CIS 7.1: Establish and Maintain a Vulnerability Management Process). Treat MCP-enabled AI developer tools as a distinct attack surface category in future risk assessments.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal counsel immediately if CloudTrail analysis confirms any exfiltrated developer STS token was used to access S3 buckets, RDS instances, or other data stores containing customer PII, PHI, or regulated data, as this triggers breach notification obligations under applicable privacy regulations; also escalate if evidence shows attacker-created IAM users or roles persist in the AWS account, indicating active account compromise beyond credential theft.
Recovery Notes	After credential rotation and patch verification, conduct a full IAM entitlement review using AWS IAM Access Analyzer to enumerate any roles, policies, or access keys created during the exposure window that may provide attacker re-entry — do not restore developer AWS access until this review is complete and clean. Monitor AWS CloudTrail and GuardDuty findings continuously for 30 days post-remediation with specific attention to AssumeRole calls from unfamiliar geographic regions or IP ranges, which would indicate the attacker retained a copy of the stolen STS token with remaining validity or established a persistent backdoor. Validate that the Language Servers for AWS upgrade to 1.69.0 resolved the MCP trust gap by reviewing the AWS security advisory and confirming the specific patch changelog addresses the `.mcp.json` workspace trust enforcement mechanism before treating the environment as clean.
Forensic Artifacts	AWS CloudTrail STS API event log: AssumeRole, GetSessionToken, and GetFederationToken events from developer workstation egress IPs during the exposure window, including the sourceIPAddress, userAgent, and requestParameters fields that identify whether the call was made by the Amazon Q Developer Language Server process or a subsequent attacker-controlled session | Amazon Q Developer Language Server log files at ~/.aws/amazonq/cache/ (Linux/macOS) or %APPDATA%\Amazon Q\logs\ (Windows): these logs record MCP server connection events, stdio channel activity, and tool execution requests, and will show the moment the Language Server connected to the attacker-specified MCP server URI embedded in the malicious .mcp.json | Malicious .mcp.json configuration file from the opened repository: contains the attacker-controlled MCP server URI and any credential exfiltration parameters, serving as the primary IOC for attribution, network-level blocking, and threat intelligence sharing | IDE extension host process memory dump (ProcDump/winpmem of VS Code, JetBrains, or Eclipse process): may contain plaintext AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and AWS_SESSION_TOKEN values that were loaded into the Language Server process environment and transmitted to the attacker MCP server via the stdio trust gap | Endpoint network capture (Wireshark/tcpdump) showing outbound HTTP/S connections from the IDE or Language Server process to non-AWS IP addresses immediately following repository workspace open event — the POST body or TLS session metadata (via JA3 fingerprint or SNI) to attacker MCP server will corroborate the credential exfiltration path and timing

Per-Action IR Details

Containment — Immediately restrict or disable Amazon Q Developer plugin use in developer workstations until the patched Language Servers for AWS version 1.69.0 (or later) is confirmed deployed. Revoke and rotate any AWS IAM session credentials (temporary tokens, access keys) associated with developer workstations that may have opened untrusted repositories with Amazon Q Developer enabled. Isolate affected developer machines from production AWS environments pending credential rotation.

NIST Phase: Containment

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

Controls: NIST AC-2 (Account Management), NIST AC-12 (Session Termination), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Disable Amazon Q Developer plugin immediately via IDE CLI: for VS Code run `code --uninstall-extension amazonwebservices.amazon-q-vscode`; for JetBrains use the plugin manager CLI or delete the plugin JAR from the plugins directory. Revoke all developer STS tokens using `aws sts get-caller-identity` to enumerate active sessions, then `aws iam delete-access-key` and `aws sts` invalidation via role policy attach. Use AWS CLI `aws iam list-access-keys --user-name ` across all developer IAM users to identify keys requiring rotation.

Evidence: BEFORE revoking credentials or isolating the workstation, capture: (1) full memory dump of the IDE process (VS Code / JetBrains / Eclipse) using ProcDump (`procdump -ma `) or `winpmem` to recover in-memory AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and AWS_SESSION_TOKEN values that may have been loaded into the plugin process environment; (2) live network connections from the IDE process using `Get-NetTCPConnection -OwningProcess ` (Windows) or `ss -tp | grep ` (Linux) to identify active exfiltration connections to attacker-controlled MCP servers; (3) running process tree showing parent-child relationship between the Amazon Q Developer Language Server process and any spawned aws-cli, curl, or wget child processes (`Get-WmiObject Win32_Process | Select ProcessId,ParentProcessId,CommandLine`); (4) environment variable snapshot of the IDE/Language Server process (`cat /proc//environ` on Linux or Sysinternals Process Explorer on Windows) to confirm which AWS credentials were live in the process at time of containment.

Detection — Query AWS CloudTrail for anomalous AssumeRole, GetSessionToken, or STS API calls originating from developer workstation IP addresses, particularly calls followed by data exfiltration patterns (S3 GetObject, Lambda invocations, or outbound API calls to unknown endpoints). Review IDE plugin logs and process execution logs for unexpected AWS CLI invocations spawned by the Amazon Q Developer plugin process. Search endpoint logs for MCP stdio channel process trees spawning aws-cli or curl/wget with AWS credential environment variables. Check for outbound connections from developer endpoints to non-corporate, non-AWS IP addresses following repository open events.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

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

Compensating: Query CloudTrail without a SIEM using AWS CLI: `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=AssumeRole --start-time --end-time ` filtered by developer workstation IPs. On Windows endpoints without EDR, deploy Sysmon with a config that captures Event ID 1 (Process Create) with ParentImage matching `amazonq-language-server.exe` or `node.exe` (the Language Server runtime) and CommandLine matching `aws` or `curl`; export with `wevtutil qe Microsoft-Windows-Sysmon/Operational`. On Linux developer workstations, use `auditd` with a rule watching execve calls by the Language Server process UID. For network detection without a commercial tool, run Wireshark capture filtered on `http.request.method == POST` or TLS SNI not matching `*.amazonaws.com` during a controlled repository-open test on a quarantined workstation.

Evidence: This is an analysis step that does not alter live state; however, capture before any parallel containment actions invalidate log integrity: (1) AWS CloudTrail event history for the 72-hour window preceding detection, specifically EventName in [AssumeRole, GetSessionToken, GetFederationToken, AssumeRoleWithWebIdentity] with sourceIPAddress matching developer NAT egress ranges — export as JSON via `aws cloudtrail lookup-events`; (2) Amazon Q Developer Language Server log files located at `~/.aws/amazonq/cache/` and `%APPDATA%\Amazon Q\logs\` (Windows) or `~/.local/share/amazon-q/logs/` (Linux/macOS), which may record MCP server connection attempts and stdio channel activity; (3) IDE extension host log for VS Code at `~/.config/Code/logs//exthost.log` filtering on 'amazonq' or 'mcp' entries; (4) `.mcp.json` or workspace-level MCP configuration files within the opened repository directory that contain the attacker-controlled server URI — preserve as-is with hash before any modification.

Eradication — Upgrade all Amazon Q Developer plugin instances to the patched versions: VS Code plugin, JetBrains, Eclipse, and Visual Studio IDE plugins released after May 12, 2026, and confirm Language Servers for AWS is at version 1.69.0 or later. Remove or audit all repository-level MCP configuration files (.mcp.json or

equivalent) in developer workspaces. Disable MCP tool execution in Amazon Q Developer settings until organizational policy for trusted MCP sources is defined and enforced.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: For a 2-person team without endpoint management tooling: enumerate all installed Amazon Q Developer plugin versions across workstations using a PowerShell one-liner `Get-Content "$env:USERPROFILE\.vscode\extensions\amazonwebservices.amazon-q-vscode*\package.json" | Select-String '"version"'` or equivalent shell script for Linux (`find ~/.local/share/code/extensions -name package.json -path '*amazon-q*' -exec grep '"version"' {} \;`). Recursively search all developer workspace directories for malicious MCP config files: `Get-ChildItem -Recurse -Filter '.mcp.json' | Select-Object FullName` (Windows) or `find /home -name '.mcp.json' 2>/dev/null` (Linux), then review each for non-approved MCP server URIs before deletion. Verify Language Server version by checking the installed npm package: `npm list -g @aws/language-servers-aws` or inspecting the binary version string.

Evidence: BEFORE patching or removing MCP configuration files, capture: (1) cryptographic hash (SHA-256) of each `.mcp.json` file found in developer workspaces to preserve as forensic evidence of the malicious configuration — `Get-FileHash .mcp.json -Algorithm SHA256` or `sha256sum .mcp.json`; (2) full content copy of each `.mcp.json` including the attacker-controlled MCP server URI/endpoint, which will be used for threat intelligence and IOC development; (3) currently installed plugin version strings for all four IDE plugin variants before upgrade, to establish the exposure window and confirm vulnerability; (4) Language Server for AWS process memory if still running, as the Language Server process (node.js-based) may retain in-memory copies of AWS credentials loaded via the MCP trust gap — use `procdump -ma $(pgrep -f 'aws-language-server')` before killing or upgrading the process.

Recovery — After credential rotation and patching, validate that no new unauthorized IAM users, roles, or access keys were created during the exposure window using AWS IAM Access Analyzer and CloudTrail. Re-enable Amazon Q Developer only on workstations confirmed to be running patched plugin versions. Establish a process for developers to report suspicious repository config files. Monitor AWS environments for 30 days post-remediation for residual unauthorized activity using AWS GuardDuty and CloudTrail anomaly detection.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-2 (Account Management), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.1 (Establish an Access Granting Process)

Compensating: Without GuardDuty, implement manual CloudTrail monitoring using AWS CLI scheduled queries: `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=CreateUser` and equivalent for CreateRole, CreateAccessKey, and AttachUserPolicy — run daily via cron/Task Scheduler and diff against a known-good IAM state snapshot taken at containment time. Use AWS IAM Access Analyzer via CLI (`aws accessanalyzer list-findings`) at no additional cost to detect externally accessible roles or policies created during the exposure window. For developer workstation re-enablement validation, require each developer to run `node --version` on the Language Server binary and confirm the npm package version matches 1.69.0 before plugin re-activation.

Evidence: Before re-enabling Amazon Q Developer on any workstation, verify and document: (1) AWS IAM Access Analyzer findings exported for the full exposure window to confirm no persistent backdoor roles with trust policies to attacker-controlled AWS accounts were created — `aws accessanalyzer list-findings --analyzer-arn --filter '{"status":{"eq":["ACTIVE"]}}'`; (2) CloudTrail CreateUser, CreateRole, CreateAccessKey, and PutRolePolicy events during the exposure window, which would indicate the attacker used the exfiltrated developer STS token to establish persistence; (3) patched plugin version confirmation screenshot or CLI output per workstation as a reactivation gate record; (4) final network traffic baseline from each recovered workstation for the first 48 hours post-re-enablement, capturing any residual beaconing to the MCP server URI identified in the malicious `.mcp.json`.

Post-Incident — This vulnerability exposes a control gap in how developer tooling with AI/MCP capabilities is governed. Implement explicit organizational policy governing approved MCP server sources and workspace trust decisions (aligns with NIST AC-3: Access Enforcement and AC-6: Least Privilege). Require sandboxed execution environments for AI coding assistants handling untrusted repositories. Add Amazon Q Developer and similar MCP-enabled tools to the software inventory and vulnerability management process (CIS 2.1: Establish and Maintain a Software Inventory; CIS 7.1: Establish and Maintain a Vulnerability Management Process). Treat MCP-enabled AI developer tools as a distinct attack surface category in future risk assessments.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For a 2-person team: create a developer workstation baseline policy document enumerating approved MCP server URIs (initially an empty allowlist requiring explicit approval), distribute as a checked-in `.mcp.json.template` in your internal repo scaffolding tool so all new projects start with a safe config. Implement a pre-commit hook using a shell script that rejects any `.mcp.json` containing URIs outside an approved domain list — deploy via `git config core.hooksPath` at the org level. Add Amazon Q Developer Language Servers for AWS to a watched-package list in a free OSS vulnerability scanner such as `pip-audit` equivalent for npm (`npm audit`) or `osv-scanner` against the installed Language Server package to catch future CVEs in the same component.

Evidence: For the lessons-learned record, preserve and attach to the post-incident report: (1) the original malicious `.mcp.json` (hashed and quarantined) with annotated analysis of the MCP server URI and credential exfiltration mechanism, to serve as a training artifact; (2) the CloudTrail timeline reconstruction showing the sequence from repository open event to first anomalous STS API call, quantifying the credential theft dwell time; (3) IAM Access Analyzer report confirming no persistent attacker-controlled resources survived credential rotation; (4) the before/after plugin version inventory across all developer workstations as evidence of eradication completeness; (5) a gap analysis document identifying which other AI coding assistant plugins (GitHub Copilot, Cursor, Codeium) in the developer environment also support MCP or similar workspace-trust execution models and should be assessed under the same threat model.

Detection Guidance

Primary detection surface is AWS CloudTrail and developer endpoint process telemetry. In CloudTrail, alert on STS GetSessionToken or AssumeRole calls from developer workstation IPs followed within minutes by API calls to S3, Lambda, or external endpoints not matching corporate patterns, this matches the credential exfiltration chain demonstrated in the Wiz Research proof-of-concept. On developer endpoints, monitor for process trees where the Amazon Q Developer plugin process (or Language Server for AWS) spawns aws, aws-cli, curl, wget, or PowerShell with AWS_ACCESS_KEY_ID or AWS_SESSION_TOKEN in the environment. Search IDE extension host logs for unexpected MCP stdio channel activations triggered by repository open events. Behavioral indicator: AWS CLI invocations that occur immediately after a developer opens a new or cloned repository, without explicit developer-initiated commands. IOC pattern: outbound HTTP/S POST requests from developer workstations to non-AWS, non-corporate domains carrying JSON payloads containing the string 'SessionToken' or 'Credentials'. MITRE mapping: T1552.001 (Credentials in Files) and T1059.007 (JavaScript execution via MCP stdio) are the highest-signal techniques to tune detection rules against. NIST AU-2 (Event Logging) and AU-6 (Audit Record Review, Analysis, and Reporting) provide the control framework for ensuring these log sources are collected and reviewed. CIS 8.2 (Collect Audit Logs) supports baseline log coverage requirements.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`Attacker-controlled HTTP/S endpoint receiving JSON POST with AWS SessionToken values`	Wiz Research proof-of-concept demonstrated exfiltration of AWS session tokens to an external server via crafted MCP config file; specific C2 domains not publicly disclosed	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1548** — Abuse Elevation Control Mechanism
- **T1552** — Unsecured Credentials
- **T1552.001** — Credentials In Files
- **T1190** — Exploit Public-Facing Application
- **T1106** — Native API
- **T1059** — Command and Scripting Interpreter
- **T1059.007** — JavaScript
- **T1078.004** — Cloud Accounts

NIST-800-53R5

- **AC-6** — Least Privilege
- **CM-6** — Configuration Settings
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AC-3** — Access Enforcement
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control
- **A03:2021** — Injection

CIS-V8

- **6.1** — Establish an Access Granting Process
- **6.8** — Define and Maintain Role-Based Access Control
- **2.5** — Allowlist Authorized Software
- **16.10** — Apply Secure Design Principles in Application Architectures
- **6.3** — Require MFA for Externally-Exposed Applications
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1548	Abuse Elevation Control Mechanism	Privilege-Escalation
T1552	Unsecured Credentials	Credential-Access
T1552.001	Credentials In Files	Credential-Access
T1190	Exploit Public-Facing Application	Initial-Access
T1106	Native API	Execution
T1059	Command and Scripting Interpreter	Execution
T1059.007	JavaScript	Execution
T1078.004	Cloud Accounts	Defense-Evasion

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/06/amazon-q-developer-flaw-could-let...	T3

Source	URL	Tier
MCP STDIO Command Injection: Full Vulnerability Advisory	https://www.ox.security/blog/mcp-supply-chain-advisory-rce-vulnerab...	T3
MCP by Design: RCE Across the AI Agent Ecosystem - Lab Space	https://labs.cloudsecurityalliance.org/research/csa-research-note-m...	T3
CVE-2026-39861: Anthropic Claude Code RCE Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2026-39861/	T3
Known Exploited Vulnerabilities Catalog | CISA	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-12957, CVE-2026-12958, CV...	T1

Generated 2026-06-26 18:40 UTC by TJS Security Command Center