INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-23 13:58 UTC

Concurrent npm Supply Chain Campaigns Deliver Windows RAT, Linux Rootkit, and Developer Credential Stealers, One Cluster Linked to North Korean PolinRider Operation

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0542
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	npm ecosystem, postcss-selector-parser (typosquatting targets; ~127M weekly downloads), Windows hosts, Linux hosts, GitHub repositories (~2,000 compromised), Claude Code, GitHub CLI, Docker, SSH key stores
Published	2026-06-23T04:54:32
Discovery Source	Rss

Executive Summary

At least five separate malicious npm campaign clusters, discovered in June 2026, are actively delivering Windows and Linux RATs, a native C rootkit, and credential stealers targeting developer toolchains including Claude Code, GitHub CLI, and SSH keys. Three packages impersonating the widely used postcss-selector-parser (127M weekly downloads) remain available on npm as of publication, lowering the barrier for accidental installation. One cluster overlaps a confirmed North Korean operation (PolinRider) that has compromised approximately 2,000 GitHub repositories to distribute BeaverTail and InvisibleFerret malware.

Technical Analysis

The PostCSS-themed cluster (aes-decode-runner-pro, postcss-minify-selector, postcss-minify-selector-parser) demonstrates a layered evasion strategy built around legitimate package branding. The attack chain moves from a JavaScript dropper to a PowerShell downloader ('settings.ps1'), then retrieves a ZIP payload from 'nvidiadriver[.]net' via curl.exe. That archive unpacks a VBScript launcher, a Python runtime, and Nuitka-compiled .pyd extension modules, a deliberate fragmentation that distributes malicious logic across six compiled components (config.pyd, api.pyd, audiodriver.pyd, command.pyd, auto.pyd, util.pyd) to complicate

static analysis. The RAT communicates with C2 at 95.216.92[.]207:8080 and specifically targets Chrome's app-bound encryption (ABE) protections for credential theft, indicating the authors anticipated modern browser hardening. Download counts are low (145-615), but the impersonation target, postcss-selector-parser, pulls over 127 million downloads weekly, creating a plausible confusion surface for dependency confusion or typosquatting attacks at scale.

The concurrent campaigns reveal a shared attacker playbook across distinct threat clusters: abuse of high-trust ecosystem names, credential exfiltration as a primary objective, and multi-stage payloads designed to survive endpoint detection. The '@withgoogle/stitch-sdk' package targets eight specific credential sources, Claude Code config, git credentials, SSH public keys, GitHub CLI tokens, npm config, .npmrc, and Docker config, representing a systematic sweep of the full developer identity surface. The exfiltration domain 'stitch-production[.]org/api/v1' mimics legitimate Google infrastructure naming. The 'apintergrationpost' Linux package goes further, compiling a native C rootkit at install time, establishing three independent persistence mechanisms, and masquerading as a systemd service, capabilities that extend well beyond a simple RAT and suggest a prepared operator with post-exploitation goals beyond initial access.

The gonex-AI/Understand-Anything supply chain attack introduces a notable C2 innovation: the second-stage command is encoded in a Tron blockchain transaction, with the active payload hash stored in a BSC (Binance Smart Chain) transaction. This architecture makes the C2 channel effectively immutable and resistant to domain takedown, defenders cannot sinkhole a blockchain address. The same operation uses horizontal whitespace in diffs to hide payloads, a technique that bypasses most automated code review tooling and challenges human reviewers. SafeDep's attribution overlaps this with the North Korean PolinRider operation, which has injected obfuscated JavaScript into approximately 2,000 compromised GitHub repositories to deliver BeaverTail (a known downloader/stealer) and the InvisibleFerret backdoor. This overlap is significant: it suggests nation-state actors are actively seeding the broader open source supply chain, not just isolated packages.

From a detection standpoint, the campaigns expose three consistent gaps: pre-install scanning does not catch payloads fetched post-install from external servers; build dependency names receive less scrutiny than direct application dependencies; and blockchain-based C2 falls outside conventional domain reputation and DNS filtering controls. The five-package cluster ('procwire', 'routecraft', 'endpointmap', 'bytecraft', 'staticlayer') uses a dependency chain to force installation of multiple malicious packages from a single entry point, while 'staticlayer' operates server-side to serve payloads to clients matching a specific User-Agent, a technique that frustrates sandbox detonation. Security teams relying solely on post-install behavioral monitoring will miss the install-time execution vectors present across most of these clusters. The consistent targeting of developer credential stores, particularly CI/CD tokens, container registry credentials, and AI coding assistant configs, signals that attackers view developer workstations as pivot points into production pipelines, not just endpoints.

Action Checklist

1. Remove immediately: aes-decode-runner-pro, postcss-minify-selector, postcss-minify-selector-parser, apintergrationpost, @withgoogle/stitch-sdk, procwire, routecraft, endpointmap, bytecraft, and staticlayer. Rotate all credentials on any machine where these were installed, including GitHub tokens, SSH keys, npm tokens, Docker credentials, and Claude Code configs.
2. Block IOCs at the network layer: nvidiadriver[.]net (payload delivery), 95.216.92[.]207:8080 (Windows RAT C2), and stitch-production[.]org (credential exfiltration endpoint). Standard domain blocklists will not cover blockchain-based C2 used in the gonex-AI/Understand-Anything campaign.

3. Enforce pre-install scanning with a tool that evaluates package metadata, publish history, and dependency chains, not just post-install behavior. Packages published by new or low-activity accounts (e.g., npm user 'abdrizak') with names closely matching high-download packages warrant automatic review holds.

4. Audit developer workstation credential stores specifically: ~/.git-credentials, ~/.npmrc, ~/.docker/config.json, GitHub CLI token storage, and any AI coding assistant config files. These are now confirmed primary targets across multiple concurrent campaigns, not incidental collection.

5. The PolinRider overlap with the gonex-AI campaign and approximately 2,000 compromised GitHub repositories elevates this beyond opportunistic malware. Treat any recently merged PR from an unfamiliar contributor, particularly those with horizontal whitespace anomalies in diffs, as requiring manual review before merge.

6. Blockchain-based C2 (Tron and BSC addresses used in the gonex-AI campaign) cannot be blocked via DNS sinkholes or domain takedowns. Detection requires behavioral monitoring for unusual outbound connections to blockchain RPC endpoints or Web3 API gateways from developer or build systems.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership, legal counsel, and potentially CISA if forensic evidence confirms credential exfiltration to stitch-production[.]org or active PolinRider/DPRK actor presence in internal repositories, as this triggers potential insider threat protocols, supply chain breach notification obligations, and OFAC nexus considerations for North Korean threat actor involvement.
Recovery Notes	Before restoring any developer workstation or CI/CD runner to production use, reimage from a known-good OS baseline and reinstall only npm packages verified against a pinned, hash-locked package-lock.json using `npm ci` rather than `npm install`. After credential rotation, monitor GitHub audit logs, npm token usage logs, and Docker Hub access logs for a minimum of 30 days for anomalous access patterns (unexpected geographies, off-hours API calls, bulk repository cloning) that would indicate stolen credentials were exfiltrated before rotation completed. For repositories with merged PRs from suspect contributors, conduct a line-by-line diff audit of all commits in the 90 days preceding discovery using `git log --all -p --since='90 days ago'` before re-enabling automated build pipelines.

Forensic Artifacts	npm package installation records and postinstall script execution logs: `~/.npm/_logs/*.log` (Linux/macOS) and `%AppData%\npm-cache\_logs\*.log` (Windows) — the malicious packages execute credential-harvesting code in their postinstall hooks, leaving execution timestamps and error output in these logs. | Credential file access timestamps (atime) on ~/.git-credentials, ~/.npmrc, ~/.docker/config.json, and ~/.config/gh/hosts.yml — the stealer payloads in aes-decode-runner-pro and @withgoogle/stitch-sdk read these files, and forensic atime comparison against the npm install timestamp confirms exfiltration timing. | Sysmon Event ID 3 (Network Connection) and Event ID 1 (Process Creation) logs on Windows showing node.exe or npm.cmd initiating outbound connections to 95.216.92[.]207:8080 (Windows RAT C2) or stitch-production[.]org — these events tie the malicious package execution directly to C2 beacon establishment. | Linux rootkit persistence indicators: kernel module load events in `/var/log/kern.log` or `dmesg` output showing unsigned or unexpected module insertions, combined with `/proc/modules` and `lsmod` output captured immediately at detection time — the native C rootkit may hook syscalls and will not appear in standard process listings after installation. | GitHub repository audit logs (accessible at github.com/settings/security-log or via GitHub API `GET /users/{username}/events`) showing token usage, repository forks, PR submissions, or secrets access from IP addresses or user agents not matching the legitimate developer's normal access pattern — critical for establishing whether stolen tokens were operationalized by the PolinRider actor against the ~2,000 compromised repository infrastructure.

Per-Action IR Details

Remove immediately: aes-decode-runner-pro, postcss-minify-selector, postcss-minify-selector-parser, apintergrationpost, @withgoogle/stitch-sdk, procwire, routecraft, endpointmap, bytecraft, and staticlayer. Rotate all credentials on any machine where these were installed, including GitHub tokens, SSH keys, npm tokens, Docker credentials, and Claude Code configs.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: Remove malicious artifacts and revoke compromised credentials before restoring affected systems to a known-good state.

Controls: NIST SI-2 (Flaw Remediation), NIST AC-2 (Account Management), CIS 2.3 (Address Unauthorized Software), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Run `npm ls --all 2>/dev/null | grep -E 'aes-decode-runner-pro|postcss-minify-selector|apintergrationpost|withgoogle/stitch-sdk|procwire|routecraft|endpointmap|bytecraft|staticlayer'` across all developer workstations and CI runners. Use `find / -name 'package.json' -not -path '*/\.*' 2>/dev/null | xargs grep -l` for the package names to catch transitive installs. Revoke GitHub tokens via `gh auth logout` and regenerate via GitHub Settings > Developer Settings > Personal Access Tokens; rotate SSH keys with `ssh-keygen -R` and reissue; invalidate npm tokens at registry.npmjs.org/settings/tokens.

Evidence: BEFORE removing packages or rotating credentials, capture: (1) full in-memory process tree on each affected host — on Windows run `Get-Process | Select-Object Name,Id,Path,StartTime | Export-Csv processes.csv` and on Linux run `ps auxf > process_tree.txt`; (2) active network connections with `Get-NetTCPConnection | Export-Csv netconn.csv` (Windows) or `ss -antp > connections.txt` (Linux) to document live C2 beaconing to 95.216.92[.]207:8080 or stitch-production[.]org; (3) contents of ~/.git-credentials, ~/.npmrc, ~/.docker/config.json, and Claude Code config directories before they are overwritten by rotation; (4) timestamps and install metadata from `npm ls -g --json` and local node_modules to establish when the malicious packages first appeared.

Block IOCs at the network layer: nvidiadriver[.]net (payload delivery), 95.216.92[.]207:8080 (Windows RAT C2), and stitch-production[.]org (credential exfiltration endpoint). Standard domain blocklists will not cover blockchain-based C2 used in the gonex-AI/Understand-Anything campaign.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: Isolate affected systems and block attacker-controlled infrastructure to prevent further credential exfiltration and C2 communication while eradication proceeds.

Controls: NIST AC-4 (Information Flow Enforcement), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Immediately push host-based firewall rules: on Windows `netsh advfirewall firewall add rule name='Block_PolinRider_C2' dir=out action=block remoteip=95.216.92.207`; on Linux `iptables -A OUTPUT -d 95.216.92.207 -j DROP`. Block nvidiadriver[.]net and stitch-production[.]org via local hosts file (`echo '0.0.0.0 nvidiadriver.net' >> /etc/hosts`) or DNS RPZ on an internal resolver. For blockchain C2 interception, use Wireshark or tcpdump with a capture filter for outbound connections to TCP/443 or TCP/80 destined for public blockchain RPC endpoints (e.g., api.trongrid.io, bsc-dataseed.binance.org) from build systems: `tcpdump -i any 'host api.trongrid.io or host bsc-dataseed.binance.org' -w blockchain_c2.pcap`.

Evidence: BEFORE implementing blocks, capture a full packet trace of any existing outbound sessions to the three named IOCs using Wireshark or tcpdump to preserve payload content and timing for forensic reconstruction. On Windows, document DNS query history from `Get-DNSClientCache | Export-Csv dns_cache.csv` and browser/process-level DNS from Sysmon Event ID 22 (DNS Query) filtering for nvidiadriver.net and stitch-production.org. On Linux, review `/var/log/syslog` and `/var/log/auth.log` for outbound connection attempts. Save full netflow or proxy logs covering the 30 days prior to discovery to establish initial compromise timing.

Enforce pre-install scanning with a tool that evaluates package metadata, publish history, and dependency chains — not just post-install behavior. Packages published by new or low-activity accounts (e.g., npm user 'abdrizak') with names closely matching high-download packages warrant automatic review holds.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Establish preventive controls and tooling that reduce the likelihood of the incident class recurring, specifically targeting the typosquatting vector exploited against postcss-selector-parser's 127M weekly download surface.

Controls: NIST SI-2 (Flaw Remediation), 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)

Compensating: Integrate Socket.dev CLI (`socket npm install` as a drop-in wrapper) or `npm audit --audit-level=moderate` combined with a custom pre-install hook in `.npmrc` (`before-install=/usr/local/bin/check-publisher.sh`) that calls the npm registry API to reject packages from accounts with fewer than 5 published packages or less than 90 days of account age: `curl -s https://registry.npmjs.org/-/user/ | jq '.created'`. Maintain an allow-list of approved package names using a `.npmrc` `allow-scripts=false` policy plus a shell alias that diffs requested package names against a Levenshtein-distance check against the top-1000 npm packages list.

Evidence: This is a preparation step that does not alter live system state; no volatile evidence capture is required before implementation. However, before deploying the scanning policy, inventory all currently installed npm packages across CI/CD runners and developer workstations using `npm ls -g --json > global_packages_$(hostname)_$(date +%Y%m%d).json` to establish a baseline and identify any additional typosquatted packages beyond the ten named in Step 1.

Audit developer workstation credential stores specifically: ~/.git-credentials, ~/.npmrc, ~/.docker/config.json, GitHub CLI token storage, and any AI coding assistant config files. These are now confirmed primary targets across multiple concurrent campaigns, not incidental collection.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: Determine the scope of credential exposure by examining the exact file paths targeted by the stealers embedded in aes-decode-runner-pro, @withgoogle/stitch-sdk, and the PolinRider-linked gonex-AI packages.

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AC-3 (Access Enforcement), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 3.2 (Establish and Maintain a Data Inventory)

Compensating: Run the following audit script on each developer workstation: `for f in ~/.git-credentials ~/.npmrc ~/.docker/config.json; do [ -f "$f" ] && echo "=== $f ==" && stat "$f" && sha256sum "$f"; done`. Check GitHub CLI token

store at `~/.config/gh/hosts.yml` and Claude Code at `~/.claude/` or equivalent config directory. On Windows, check `%APPDATA%\GitHub CLI\hosts.yml` and `%USERPROFILE%\.npmrc`. Flag any file with an `atime` (last access time) within the window of package installation — use `stat -c '%x %n' ~/.git-credentials` on Linux — as evidence of stealer access. Cross-reference GitHub audit logs at github.com/settings/security-log for any token usage from unexpected IPs or geographies following the install timestamp.

Evidence: BEFORE any credential rotation from this step (which would overwrite the files), use `stat` to record exact access times (atime) on all credential files listed — the stealer reads these files, leaving an atime footprint that proves exfiltration occurred. On Linux, run `find ~ -name '.git-credentials' -o -name '.npmrc' -o -name 'config.json' -path '*docker*' | xargs stat 2>/dev/null > cred_file_timestamps.txt`. Capture GitHub token usage logs via the GitHub API: `curl -H 'Authorization: token ' https://api.github.com/users//events` before the token is revoked, to determine if the stolen token was used for repository access, PR creation, or secrets enumeration consistent with the ~2,000 compromised repository pattern.

The PolinRider overlap with the gonex-AI campaign and approximately 2,000 compromised GitHub repositories elevates this beyond opportunistic malware. Treat any recently merged PR from an unfamiliar contributor — particularly those with horizontal whitespace anomalies in diffs — as requiring manual review before merge.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: Expand scope assessment to include repository integrity, as the PolinRider/North Korean actor TTPs include code injection via compromised accounts with subtle diff-level obfuscation that evades automated merge checks.

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST CM-3 (Configuration Change Control), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Use the GitHub CLI to enumerate recently merged PRs from first-time contributors: `gh pr list --state merged --json author,mergedAt,files | jq '.[] | select(.author.login | IN(known_contributors[]) | not)'`. For whitespace anomaly detection, run `git log --all --diff-filter=M -p | grep -P '[\t ]{4,}$'` to surface trailing whitespace insertions or Unicode homoglyph substitutions in recent commits. Install a pre-receive hook or use `git diff --check` in CI to flag whitespace warnings. Cross-reference PR author accounts against npm publish history — if a GitHub contributor also owns one of the malicious npm accounts, treat all their merged code as potentially compromised.

Evidence: This step does not alter live system state directly, so no volatile capture prerequisite applies. However, before reverting any suspicious commits (which would alter repository history), preserve the full git log: `git log --all --format='%H %ae %ai %s' > full_commit_log_$(date +%Y%m%d).txt` and clone all potentially affected repositories with `git clone --mirror` to preserve the exact state for forensic analysis. Document the specific diff content of flagged PRs, including raw byte-level inspection with `git show | xxd | grep -A2 -B2 'e2 80'` to detect Unicode homoglyphs (U+200B zero-width space, U+00A0 non-breaking space) used in PolinRider-style whitespace anomaly injection.

Blockchain-based C2 (Tron and BSC addresses used in the gonex-AI campaign) cannot be blocked via DNS sinkholes or domain takedowns. Detection requires behavioral monitoring for unusual outbound connections to blockchain RPC endpoints or Web3 API gateways from developer or build systems.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Establish detection capability for a C2 channel that bypasses conventional domain-based blocking, requiring behavior-based network monitoring tuned to Tron (api.trongrid.io) and BSC (bsc-dataseed.binance.org) RPC traffic patterns from non-Web3 developer environments.

Controls: NIST AU-2 (Event Logging), NIST AU-12 (Audit Record Generation), NIST SC-7 (Boundary Protection), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy a Sigma rule on developer and build system hosts to alert on outbound HTTPS connections from Node.js processes (node, npm, npx) to known blockchain RPC domains: target hosts include api.trongrid.io (Tron), bsc-dataseed.binance.org, bsc-dataseed1.binance.org, bsc-dataseed2.defibit.io (BSC). On Linux, use auditd: `auditctl -a always,exit -F arch=b64 -S connect -F exe=/usr/bin/node -k blockchain_c2`. On Windows, enable Sysmon Event ID 3 (Network Connection) and filter for `node.exe` connecting to port 443 with destination hostnames matching the RPC endpoint list. Use tcpdump on CI runners: `tcpdump -i any -w blockchain_monitor.pcap 'host api.trongrid.io or

host bsc-dataseed.binance.org'` and rotate captures daily for retrospective analysis.

Evidence: This is a preparation step that does not alter live state; no volatile evidence prerequisite applies before implementation. For retrospective detection on systems where the gonex-AI/Understand-Anything packages may have already been installed, review historical outbound connection logs: on Linux check `/var/log/ufw.log`, `/var/log/syslog` for DNS queries or TCP sessions to Tron/BSC RPC hostnames; on Windows review Windows Firewall logs at `%SystemRoot%\System32\LogFiles\Firewall\pfirewall.log` filtering for the RPC endpoint IP ranges. If Sysmon was pre-installed, query Event ID 3 logs for `node.exe` or `npm.cmd` network initiations within the 30-day window preceding discovery.

Detection Guidance

Monitor for outbound connections to 95.216.92.207:8080 and blockchain RPC endpoints (Tron, BSC) from developer workstations and build systems. Alert on install-time execution of PowerShell, VBScript, or Python from npm package directories. Flag horizontal whitespace anomalies in GitHub diffs using regex or AST-based code review tools. Detect post-install network calls to external domains (nvidiadriver[.]net, stitch-production[.]org) from npm package install contexts. Monitor for credential file access patterns (git-credentials, .npmrc, .docker/config.json) during or immediately after npm package installation.

Framework Mappings

MITRE-ATTACK

- **T1041** — Exfiltration Over C2 Channel
- **T1082** — System Information Discovery
- **T1059.007** — JavaScript
- **T1071.001** — Web Protocols
- **T1056** — Input Capture
- **T1564.001** — Hidden Files and Directories
- **T1059.001** — PowerShell
- **T1543.003** — Windows Service
- **T1102** — Web Service
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1105** — Ingress Tool Transfer
- **T1552.004** — Private Keys
- **T1554** — Compromise Host Software Binary
- **T1027.002** — Software Packing
- **T1555.003** — Credentials from Web Browsers
- **T1497** — Virtualization/Sandbox Evasion
- **T1027** — Obfuscated Files or Information
- **T1552.001** — Credentials In Files
- **T1543** — Create or Modify System Process
- **T1059.005** — Visual Basic

- **T1057** — Process Discovery
- **T1547.001** — Registry Run Keys / Startup Folder
- **T1059.006** — Python

NIST-800-53R5

- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

CIS-V8

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

HIPAA-SECURITY

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

SOC2-TSC

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

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
T1041	Exfiltration Over C2 Channel	Exfiltration
T1082	System Information Discovery	Discovery
T1059.007	JavaScript	Execution

Technique ID	Technique Name	Tactic
T1071.001	Web Protocols	Command-And-Control
T1056	Input Capture	Collection
T1564.001	Hidden Files and Directories	Defense-Evasion
T1059.001	PowerShell	Execution
T1543.003	Windows Service	Persistence
T1102	Web Service	Command-And-Control
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1105	Ingress Tool Transfer	Command-And-Control
T1552.004	Private Keys	Credential-Access
T1554	Compromise Host Software Binary	Persistence
T1027.002	Software Packing	Defense-Evasion
T1555.003	Credentials from Web Browsers	Credential-Access
T1497	Virtualization/Sandbox Evasion	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1552.001	Credentials In Files	Credential-Access
T1543	Create or Modify System Process	Persistence
T1059.005	Visual Basic	Execution
T1057	Process Discovery	Discovery
T1547.001	Registry Run Keys / Startup Folder	Persistence
T1059.006	Python	Execution

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/06/malicious-npm-packages-pose-as-po...	T3
nikvdp/cco: A thin protective layer for Claude Code	https://github.com/nikvdp/cco	T3

Source	URL	Tier
postcss-selector-parser	https://security.snyk.io/package/npm/postcss-selector-parser	T3
Blog - Real-time Open Source Software Supply Chain ...	https://safedep.io/blog	T3
Newsroom	https://www.stepsecurity.io/newsroom	T3

Generated 2026-06-23 13:58 UTC by TJS Security Command Center