INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-29 18:48 UTC

DPRK Famous Chollima Escalates Contagious Interview Campaign with AI-Assisted Supply Chain Attacks and LLC Cover Infrastructure

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0239
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	npm registry (axios, @solana-launchpad/sdk, @validate-sdk/v2, express-session-js, csec-crypto-utils, graph-dynamic, graphbase-js, graphlib-js), PyPI (solana-sdk), Solana ecosystem, GitHub, Vercel, Windows, Linux, macOS
Published	2026-04-29T10:43:00
Discovery Source	Rss

Executive Summary

North Korea's Famous Chollima group has significantly expanded the Contagious Interview campaign, embedding malicious code into widely-used npm and PyPI packages, including the popular axios library, to steal cryptocurrency wallets and source code from software developers. The operation now uses AI-generated commits to evade detection and a legitimate registered U.S. LLC as cover infrastructure, representing an increase in operational maturity. Organizations with developers working in Web3, Solana, or any project consuming the affected packages face direct risk of credential theft, source code exfiltration, and full system compromise across Windows, Linux, and macOS.

Technical Analysis

Famous Chollima (DPRK-affiliated, overlapping with the broader Lazarus Group cluster) is running three named sub-operations, PromptMink, Contagious Trader, and graphalgo, under the Contagious Interview campaign umbrella. The attack chain targets software developers through fake job interview lures (T1566, T1566.003, T1204.002), trojanized open-source packages (T1195.001, CWE-829), and dependency confusion/injection (CWE-494) across npm and PyPI. Confirmed or suspected malicious packages include: axios (npm, trojanized version), @solana-launchpad/sdk, @validate-sdk/v2, express-session-js, csec-crypto-utils, graph-dynamic, graphbase-js, graphlib-js (npm), and solana-sdk (PyPI). Payloads are cross-platform RATs capable of operating on Windows, Linux, and macOS, with capabilities including keylogging (T1056.001), clipboard capture (T1115),

screenshot capture (T1113), credential harvesting from files (T1552, T1552.001, CWE-312), file enumeration (T1083), SSH lateral movement (T1021.004), and exfiltration over C2 (T1041). AI-generated commit histories (attributed to large language models in threat intelligence reports) reduce detection probability (T1027, T1036, T1036.005, CWE-506). The threat actor registered a Florida LLC as operational cover for C2 and infrastructure legitimacy (T1588.006). JavaScript is the primary execution vehicle (T1059.007) alongside broader scripting (T1059). No CVE IDs are assigned. Affected packages have been removed from npm and PyPI by registry administrators. No patch is available; mitigation is immediate dependency removal and environment verification. CWE coverage: CWE-312 (cleartext credential storage), CWE-494 (download without integrity check), CWE-506 (embedded malicious code), CWE-829 (inclusion of functionality from untrusted control sphere).

Action Checklist

1. Step 1: Containment, Immediately audit all package.json and requirements.txt files across every developer workstation, CI/CD pipeline, and build server. Identify and remove any of the following packages: axios (verify against official axios maintainer releases at npmjs.com/package/axios), @solana-launchpad/sdk, @validate-sdk/v2, express-session-js, csec-crypto-utils, graph-dynamic, graphbase-js, graphlib-js (npm), and solana-sdk (PyPI). Isolate any system that installed these packages since the earliest known campaign activity. Block outbound connections to any C2 infrastructure identified in threat intelligence feeds associated with Famous Chollima/Lazarus Group.
2. Step 2: Detection, Search endpoint and SIEM logs for: npm install or pip install events involving the named packages; outbound connections to unfamiliar or newly registered domains from developer workstations and CI runners; clipboard access, screenshot, or keylogging process behavior on developer systems; SSH connections (T1021.004) originating from developer machines to internal systems post-package install; file access patterns consistent with source code enumeration (T1083), large recursive reads of .git directories, .env files, and wallet key files. Check GitHub Actions and Vercel build logs for unexpected dependency additions. Query EDR for process trees spawned by node or python processes that initiate network connections to external IPs.
3. Step 3: Eradication, Remove all identified malicious packages from every environment (dev, staging, production, CI/CD). Clear npm and pip caches on affected systems and regenerate clean package-lock.json and requirements.txt files from verified sources. Rotate all credentials, API keys, wallet private keys, SSH keys, and secrets that may have been accessible on any system where malicious packages executed. Revoke and regenerate any GitHub tokens, cloud provider credentials, and deployment keys stored in .env files or shell history on affected machines. Re-image or perform a validated clean restore of any developer workstation confirmed to have executed malicious package code.
4. Step 4: Recovery, Verify package integrity by re-resolving all dependencies against package-lock.json or requirements.txt pinned to known-good hashes. Enable npm audit and pip-audit in CI pipelines before restoring builds to production. Confirm no unauthorized commits or dependency changes were introduced to internal repositories during the exposure window. Monitor new outbound connections and process behavior on restored systems for 30 days. Validate that rotated credentials have propagated correctly and that old credentials no longer authenticate.
5. Step 5: Post-Incident, This campaign exploited three control gaps: absence of software composition analysis (SCA) tooling in CI/CD pipelines, lack of sub-resource integrity or dependency pinning enforcement, and insufficient developer security awareness about supply chain lures. Implement mandatory SCA scanning (e.g., Socket.dev, Snyk, or Dependabot with alerting) in all pipelines. Enforce dependency pinning with hash verification. Establish a vetting policy for new package adoption. Brief

development teams on interview-based social engineering lures. Map findings to MITRE ATT&CK T1195.001 and review supply chain controls against NIST SP 800-161r1.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to executive leadership, legal counsel, and external IR retainer immediately if forensic evidence confirms execution of malicious package payload on any system with access to Solana wallet private keys, cloud provider credentials, GitHub tokens with repo write access, or source code repositories containing proprietary IP, as this constitutes a likely data breach triggering regulatory notification obligations and potential financial loss from cryptocurrency theft by a DPRK-attributed threat actor subject to OFAC sanctions considerations.
Recovery Notes	Re-admit developer workstations to production CI/CD pipelines only after completing full re-image from a pre-incident gold image, re-resolving all dependencies via `npm ci` against a cryptographically verified lockfile, and confirming all credentials rotated during eradication have been invalidated at their issuing platforms. Monitor restored systems for 30 days using Sysmon Event ID 3 and auditd `connect` syscall rules watching for `node` or `python3` outbound connections to non-CDN external IPs, as BeaverTail/InvisibleFerret payloads associated with this campaign have exhibited delayed callback behavior. Any new outbound connection from a restored developer host to a domain registered within the past 90 days during the monitoring window should be treated as a potential re-infection indicator requiring immediate re-isolation.
Forensic Artifacts	npm debug logs at ~/.npm/_logs/ (timestamped, contain exact package names, versions, install times, and post-install script execution records for the malicious axios variant and companion packages) | node_modules directory contents under all project roots — specifically secondary loader files injected into legitimate package paths (e.g., axios/lib/core/ or axios/lib/adapters/) by the malicious package that shadow clean axios internals and contain the BeaverTail/InvisibleFerret stager code | macOS Unified Log entries for clipboard access and screenshot APIs (`log show --predicate 'subsystem == "com.apple.screencapture" OR subsystem == "com.apple.pasteboard"' --last 7d`) and equivalent Sysmon Event ID 24 (Clipboard Change) on Windows, capturing credential and wallet key harvesting activity from the Contagious Interview payload | Shell history files (~/.bash_history, ~/.zsh_history, ~/.node_repl_history) and npm install logs correlated with network flow data to establish the exact install-to-callback timeline, confirming whether post-install hooks executed C2 beacon traffic to Famous Chollima infrastructure | Git reflog and `git log --all --full-history` output from all internal source code repositories accessible from compromised developer workstations during the exposure window, to detect T1083 enumeration exfiltration and any unauthorized commits introduced via compromised GitHub tokens

Per-Action IR Details

Step 1: Containment — Immediately audit all package.json and requirements.txt files across every developer workstation, CI/CD pipeline, and build server. Identify and remove any of the following packages: axios (verify against the known-clean version pinned to official axios maintainer releases at npmjs.com/package/axios), @solana-launchpad/sdk, @validate-sdk/v2, express-session-js, csec-crypto-utils, graph-dynamic, graphbase-js, graphlib-js (npm), and solana-sdk (PyPI). Isolate any system that installed these packages since the earliest known campaign activity. Block outbound connections to any C2 infrastructure identified in threat

intelligence feeds associated with Famous Chollima/Lazarus Group.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality), NIST SI-3 (Malicious Code Protection), CIS 2.3 (Address Unauthorized Software), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: On each developer workstation and CI runner, execute: `find / -name 'package.json' -o -name 'requirements.txt' 2>/dev/null | xargs grep -l 'axios\|solana-launchpad\|validate-sdk\|express-session-js\|csec-crypto-util s\|graph-dynamic\|graphbase-js\|graphlib-js\|solana-sdk'` to enumerate affected manifests. Use `npm ls --all 2>/dev/null | grep -E 'axios|solana-launchpad|csec-crypto'` to surface transitive installs. For network isolation on Linux hosts without a managed firewall, run `iptables -I OUTPUT -d -j DROP` using IOCs from CISA or community Famous Chollima TI reports; on Windows use `netsh advfirewall firewall add rule name='Block_ChollimaCnC' dir=out action=block remoteip=`. A 2-person team can script this across hosts via Ansible or a simple SSH for-loop.

Evidence: Before removing any package, snapshot: (1) `npm ls --json > npm_tree_$(hostname)_$(date +%F).json` and `pip list --format=json > pip_list_$(hostname)_$(date +%F).json` to preserve installed dependency state; (2) node_modules directory listings under project roots — the malicious axios variant and companion packages write a secondary loader file (commonly `lib/core/settle.js` or a similarly named file shadowing legitimate axios internals) that should be preserved as forensic evidence before removal; (3) ~/.npm/_cacache and ~/.cache/pip/wheels directories which retain cached tarballs of the malicious packages and their embedded payloads; (4) outbound NetFlow or firewall deny logs showing connections from node or python processes to newly registered or low-reputation domains from developer workstation IPs at package install time (correlate with npm/pip install timestamps from shell history).

Step 2: Detection — Search endpoint and SIEM logs for: npm install or pip install events involving the named packages; outbound connections to unfamiliar or newly registered domains from developer workstations and CI runners; clipboard access, screenshot, or keylogging process behavior on developer systems; SSH connections (T1021.004) originating from developer machines to internal systems post-package install; file access patterns consistent with source code enumeration (T1083) — large recursive reads of .git directories, .env files, and wallet key files. Check GitHub Actions and Vercel build logs for unexpected dependency additions. Query EDR for process trees spawned by node or python processes that initiate network connections to external IPs.

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 AU-12 (Audit Record Generation), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy Sysmon (config: SwiftOnSecurity or Olaf Hartong baseline) on all developer Windows workstations — Event ID 1 (Process Create) will capture `node.exe` and `python.exe` spawning child processes such as `cmd.exe`, `powershell.exe`, or `sh`; Event ID 3 (Network Connection) will log outbound TCP from `node.exe`/`python.exe` to external IPs. On Linux/macOS developer machines, enable auditd with rules: `-a always,exit -F arch=b64 -S execve -F exe=/usr/bin/node -k node_exec` and `-a always,exit -F arch=b64 -S connect -F exe=/usr/bin/python3 -k python_net`. Use osquery with this query to surface suspicious file reads: `SELECT p.name, p.pid, f.path FROM process_open_files f JOIN processes p ON f.pid = p.pid WHERE f.path LIKE '%/.env' OR f.path LIKE '%/.git/config' OR f.path LIKE '%wallet%' OR f.path LIKE '%.pem';` Apply the Sigma rule `proc_creation_win_node_network_connection.yml` (community repo) against Windows Event Logs using `sigmac` converting to grep-compatible format if no SIEM is available. For GitHub Actions, run: `gh api /repos/{owner}/{repo}/actions/runs --paginate | jq '.workflow_runs[] | select(.created_at > "")'` and diff `package-lock.json` commits against known-good baselines using `git log --diff-filter=M -- package-lock.json`.

Evidence: Collect before analysis: (1) Shell history files (`~/.bash_history`, `~/.zsh_history`) on developer workstations for `npm install` or `pip install` invocations referencing the named packages, with timestamps; (2) npm install logs at `~/.npm/_logs/` (timestamped debug log files) which record exact package names, versions, and install times for correlation with network events; (3) Sysmon Event ID 3 or auditd `connect` syscall records showing `node` or `python3` processes initiating TCP connections to external IPs within seconds to minutes of package installation — this is the

Famous Chollima post-install hook execution window; (4) GitHub Actions workflow run logs and `package-lock.json` git diff history to identify AI-generated dependency injection commits, which may show unusually clean commit messages or generic contributor personas consistent with the LLC cover infrastructure used by this campaign; (5) Clipboard API access events on macOS (unified log: `log show --predicate 'subsystem == "com.apple.pasteboard""`) and Windows (Sysmon Event ID 24 — Clipboard Change) capturing potential keylogger or clipboard stealer activity from the malicious package payload.

Step 3: Eradication — Remove all identified malicious packages from every environment (dev, staging, production, CI/CD). Clear npm and pip caches on affected systems. Rotate all credentials, API keys, wallet private keys, SSH keys, and secrets that may have been accessible on any system where malicious packages executed. Revoke and regenerate any GitHub tokens, cloud provider credentials, and deployment keys stored in .env files or shell history on affected machines. Re-image or perform a validated clean restore of any developer workstation confirmed to have executed malicious package code.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication and Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST IA-5 (Authenticator Management), NIST CM-7 (Least Functionality), CIS 5.2 (Use Unique Passwords), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For cache clearing: `npm cache clean --force` and `pip cache purge` on each affected host; additionally delete `~/.npm/_cacache` manually to eliminate cached tarballs of malicious packages. For credential rotation without a secrets manager: enumerate all `.env` files with `find / -name '.env' -not -path '*/node_modules/*' 2>/dev/null`, extract all key/value pairs referencing tokens, keys, or passwords, and immediately revoke each via its issuing platform API (GitHub: `gh auth token` revoke; AWS: `aws iam delete-access-key`). For SSH key rotation, enumerate `~/.ssh/authorized_keys` on all internal servers that developer workstations authenticated to post-install, remove compromised public keys, and issue new keypairs. Re-imaging should use a validated gold image with cryptographic hash verification — do not restore from backup taken after earliest possible package install date. Use ClamAV with the `npm-malware` community signature set to scan any system where re-imaging is deferred.

Evidence: Preserve before eradication: (1) Full forensic disk image of any confirmed-compromised developer workstation using `dc3dd` or `ddrescue` prior to re-imaging — Famous Chollima payloads associated with Contagious Interview (BeaverTail/InvisibleFerret malware family) write persistent artifacts to `~/.npm/` subdirectories and macOS `~/Library/Application Support/` paths that will be lost on re-image; (2) Memory dump using `LiME` (Linux) or `winpmem` (Windows) to capture in-memory credential material or running payload stages from `node`/`python` processes before killing them; (3) Complete list of outbound connections made by affected processes during the exposure window from firewall or endpoint logs — this establishes the data exfiltration scope for Solana wallet keys and source code; (4) Git reflog (`git reflog --all`) from all internal repos accessible from compromised workstations to detect unauthorized pushes or branch creation during the exposure window.

Step 4: Recovery — Verify package integrity by re-resolving all dependencies against package-lock.json or requirements.txt pinned to known-good hashes. Enable npm audit and pip-audit in CI pipelines before restoring builds to production. Confirm no unauthorized commits or dependency changes were introduced to internal repositories during the exposure window. Monitor new outbound connections and process behavior on restored systems for 30 days. Validate that rotated credentials have propagated correctly and that old credentials no longer authenticate.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST IR-4 (Incident Handling), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For integrity verification without a commercial SCA tool: run `npm ci` (not `npm install`) against a known-good `package-lock.json` committed before the campaign window — `npm ci` enforces exact lockfile resolution and will fail on hash mismatches. Verify lockfile integrity with `cat package-lock.json | python3 -c "import sys,json,hashlib; data=sys.stdin.read(); print(hashlib.sha256(data.encode()).hexdigest())"` and compare to a hash

captured from a trusted pre-incident git commit. For pip: use `pip-audit` (free, pip-installable) to scan resolved dependencies against OSV and PyPI advisory databases — `pip-audit -r requirements.txt --require-hashes`. For 30-day post-recovery monitoring on Linux without EDR, configure auditd to watch for new outbound connections from `node` and `python3` and alert via a daily cron job that diffs `ss -tnp` output against a baseline. For unauthorized commit detection: `git log --since='' --all --pretty=format:'%H %an %ae %s' | grep -v '` across all internal repos.

Evidence: During recovery validation: (1) Capture `npm audit --json` output post-re-resolution and retain as a signed artifact proving clean dependency state at recovery time; (2) Pull `git log --stat` for all internal repositories covering the exposure window and diff against the pre-incident baseline to identify any source code exfiltration or unauthorized modification — Famous Chollima's T1083 file enumeration of `.git` directories means internal repo contents should be treated as potentially exfiltrated; (3) Verify old rotated credentials are inactive by attempting authentication with revoked tokens against GitHub, AWS, and any Solana wallet interfaces — log the 401/403 responses as evidence of successful rotation; (4) Capture a baseline network connection profile for restored developer workstations using `netstat -tnp` or `ss -tnp` immediately post-recovery to anchor the 30-day monitoring period.

Step 5: Post-Incident — This campaign exploited three control gaps: absence of software composition analysis (SCA) tooling in CI/CD pipelines, lack of sub-resource integrity or dependency pinning enforcement, and insufficient developer security awareness about supply chain lures. Implement mandatory SCA scanning (e.g., Socket.dev, Snyk, or Dependabot with alerting) in all pipelines. Enforce dependency pinning with hash verification. Establish a vetting policy for new package adoption. Brief development teams on interview-based social engineering lures. Map findings to MITRE ATT&CK T1195.001 and review supply chain controls against NIST SP 800-161r1.

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 SI-2 (Flaw Remediation), NIST SA-12 (Supply Chain Protection), NIST RA-3 (Risk Assessment), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: For SCA without budget: integrate `npm audit` and `pip-audit` as mandatory CI gates using a free GitHub Actions workflow — a failing audit score blocks merge. For dependency pinning enforcement, configure `.npmrc` with `package-lock=true` and add a pre-commit hook using `husky` that runs `npm ci --dry-run` to detect lockfile drift before commit. For package vetting, create a simple markdown-based Package Adoption Request policy requiring maintainer identity verification (check npmjs.com maintainer account age and activity), download velocity review, and a manual diff of the package tarball against its published GitHub source using `npm pack` followed by `diff -r`. For developer awareness specific to this campaign: run a tabletop exercise where developers receive a simulated 'interview task' zip file from a recruiter persona and must identify the embedded `npm install` social engineering lure — this directly mirrors the Famous Chollima Contagious Interview delivery mechanism. Maintain a free MITRE ATT&CK Navigator layer file mapping T1195.001 (Supply Chain Compromise: Compromise Software Dependencies and Development Tools), T1083 (File and Directory Discovery), T1021.004 (Remote Services: SSH), and T1555 (Credentials from Password Stores) as a living reference for future detections.

Evidence: Lessons-learned artifacts to produce: (1) Timeline reconstruction document mapping first possible package install date to detection date, quantifying the dwell time during which Famous Chollima payloads had access to developer credentials, wallet keys, and source code — this establishes breach notification scope if PII or regulated data was accessible; (2) Inventory of all internal repositories, .env files, and Solana wallet files accessible from compromised workstations during the exposure window, retained as evidence for potential regulatory disclosure; (3) Root cause analysis documenting the absence of SCA controls in the CI/CD pipeline as the primary control failure enabling T1195.001 exploitation, with before/after pipeline configuration screenshots; (4) ATT&CK technique mapping document for T1195.001, T1083, T1021.004, T1555 tied to specific observed artifacts from this incident, formatted for sharing with ISACs or CISA per IR-6 (Incident Reporting) obligations.

Detection Guidance

Priority detection targets for this campaign: (1) Package installation events, query package manager logs, CI/CD pipeline logs (GitHub Actions, Vercel, Jenkins), and endpoint EDR for installs of the named packages. (2) Outbound C2 beaconing, look for periodic outbound HTTP/S or raw TCP connections from node.js or python processes to newly registered or low-reputation domains; Famous Chollima C2 infrastructure has been associated with LLC-registered U.S.-hosted domains in prior campaigns. (3) Credential file access, alert on any process reading .env files, ~/.ssh/id_rsa, wallet keystore files (*.keystore, *.json wallet files), or browser credential stores outside of expected application context. (4) Clipboard and keylogging behavior, EDR telemetry showing unexpected clipboard API calls or input capture from scripting runtimes. (5) Screenshot activity, process-level screenshot API calls from node or python outside of test frameworks. (6) Lateral SSH movement, SSH sessions initiated by developer workstations to internal servers not in normal baseline. Behavioral IOC pattern: a node or python process that installs, then immediately enumerates the filesystem, accesses secrets, and initiates an outbound connection is the core kill chain signature. YARA and Sigma rules targeting the named package names and known-bad commit hashes should be sourced from Socket.dev's published research and community threat intel feeds. Refer to Socket.dev's blog post (listed in sources) for current IOC lists, package metadata, and advisory links.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`@solana-launchpad/sdk (npm package)`	Malicious npm package identified in Contagious Trader sub-operation; targets Solana developers for wallet exfiltration	HIGH
DOMAIN	`@validate-sdk/v2 (npm package)`	Malicious npm package linked to Famous Chollima Contagious Interview campaign	HIGH
DOMAIN	`express-session-js (npm package)`	Trojanized npm package used for credential and session theft	HIGH
DOMAIN	`csec-crypto-utils (npm package)`	Malicious npm package in Contagious Interview campaign; crypto utility masquerade	HIGH
DOMAIN	`graph-dynamic (npm package)`	Malicious npm package linked to graphalgo sub-operation	HIGH
DOMAIN	`graphbase-js (npm package)`	Malicious npm package linked to graphalgo sub-operation	HIGH
DOMAIN	`graphlib-js (npm package)`	Malicious npm package linked to graphalgo sub-operation	HIGH
DOMAIN	`solana-sdk (PyPI package)`	Malicious PyPI package targeting Solana Python developers; wallet and credential exfiltration	HIGH

Type	Value	Context	Confidence
DOMAIN	`axios (npm package — trojanized version)`	Widely-used axios npm package confirmed trojanized or compromised in a specific version; verify against official maintainer releases before trusting	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1027** — Obfuscated Files or Information
- **T1083** — File and Directory Discovery
- **T1021.004** — SSH
- **T1588.006** — Vulnerabilities
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1552** — Unsecured Credentials
- **T1036** — Masquerading
- **T1059.007** — JavaScript
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1566** — Phishing
- **T1056.001** — Keylogging
- **T1204.002** — Malicious File
- **T1566.003** — Spearphishing via Service
- **T1113** — Screen Capture
- **T1059** — Command and Scripting Interpreter
- **T1041** — Exfiltration Over C2 Channel
- **T1115** — Clipboard Data
- **T1552.001** — Credentials In Files

NIST-800-53R5

- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **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
- **15.1** — Establish and Maintain an Inventory of Service Providers
- **8.2** — Collect Audit Logs

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program
- **DE.CM-01** — Networks and network services are monitored

ISO-27001-2022

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

SOC2-TSC

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1027	Obfuscated Files or Information	Defense-Evasion
T1083	File and Directory Discovery	Discovery
T1021.004	SSH	Lateral-Movement
T1588.006	Vulnerabilities	Resource-Development
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1552	Unsecured Credentials	Credential-Access
T1036	Masquerading	Defense-Evasion
T1059.007	JavaScript	Execution
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1566	Phishing	Initial-Access
T1056.001	Keylogging	Collection
T1204.002	Malicious File	Execution
T1566.003	Spearphishing via Service	Initial-Access

Technique ID	Technique Name	Tactic
T1113	Screen Capture	Collection
T1059	Command and Scripting Interpreter	Execution
T1041	Exfiltration Over C2 Channel	Exfiltration
T1115	Clipboard Data	Collection
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/04/new-wave-of-dprk-attacks-uses-ai....	T3
solana-launchpad/sdk - NPM	https://npmjs.com/package/@solana-launchpad/sdk	T3
michaelhly/solana-py: Solana Python SDK - GitHub	https://github.com/michaelhly/solana-py	T3
solana-sdk - PyPI	https://pypi.org/project/solana-sdk/	T3
Malicious npm Package Targets Solana Developers and Hijacks ...	https://socket.dev/blog/malicious-npm-package-targets-solana-develo...	T3

Generated 2026-04-29 18:48 UTC by TJS Security Command Center