INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-05 08:36 UTC

AI-Assisted Attacks Collapse Exploit Windows and Scale Supply Chain Threats: 2025 Data Informs 2026 Defense Posture

SECURITY ANALYSIS | **HIGH** | CVSS 7.5

SCC Item ID	SCC-STY-2026-0108
Type	Security Analysis
Severity	HIGH
CVSS Base Score	7.5
Affected Products	npm ecosystem, PyPI, Chainguard Libraries, ChatGPT, Claude Code, Rakuten Mobile, Kaikatsu Club, Trust Wallet Chrome extension
Published	2026-05-04T07:58:00
Discovery Source	Rss

Executive Summary

Quantitative intelligence from 2025 confirms a structural shift in the attacker economy: AI-assisted tools have compressed the average exploit window from roughly 700 days in 2020 to 44 days in 2025, with nearly one-third of disclosed vulnerabilities now exploited within 24 hours. Simultaneously, malicious package injection into public software repositories has scaled to an estimated 454,600 uploads per year, enabling non-technical actors to execute supply chain attacks that previously required specialized expertise. For boards and CISOs, the operational implication is direct: speed and volume have outpaced conventional patch and review cycles, and the assumption that complexity filters out unsophisticated attackers no longer holds.

Technical Analysis

The 2025 threat landscape, as synthesized by Chainguard research and consistent with industry threat reporting, describes an inflection point driven by two intersecting forces: AI-accelerated exploitation and industrialized supply chain poisoning.

On the exploitation side, according to 2025 threat intelligence aggregates (confidence: directional, source tier T3), the median time-to-exploitation for newly disclosed CVEs has collapsed from approximately 700 days in 2020 to 44 days in 2025. More critically, based on the same threat intelligence sources, 28.3% of CVEs are now exploited within 24 hours of public disclosure, a figure that effectively nullifies patch-before-exploitation windows for organizations without automated remediation pipelines. AI-assisted coding tools appear to be the primary

accelerant: they lower the skill floor for weaponizing disclosed vulnerabilities, enable rapid variant generation, and produce malware variants that reportedly evade traditional static analysis tooling by altering signatures at scale. The MITRE ATT&CK techniques mapped to this cluster reflect this reality: T1203 (exploitation for client execution), T1059 (command and scripting interpreter abuse), T1562.001 (impair defenses), and T1486 (data encrypted for impact) suggest a full kill chain from initial access through impact.

On the supply chain side, the npm and PyPI ecosystems are the primary battlegrounds. According to 2025 threat intelligence aggregates (confidence: directional, source tier T3), an estimated 454,600 malicious packages were uploaded to public repositories in 2025. Attack techniques observed across named campaigns include layered dependency confusion, where attackers nest malicious payloads inside transitive dependencies to evade shallow review, and backdoored GPT-proxy packages that present as AI utility libraries while silently relaying traffic to attacker infrastructure. According to Aikido's 2026 analysis of the GPT-proxy backdoor campaign, some variants specifically target Chinese LLM infrastructure endpoints. A separate campaign cluster focused on credential harvesting: packages designed to exfiltrate cryptographic keys, CI/CD pipeline secrets, and API tokens from developer environments, mapped to T1552.001 (credentials in files) and T1195.001/T1195.002 (supply chain compromise at the dependency and software levels).

Named affected entities, Rakuten Mobile, Kaikatsu Club, and the Trust Wallet Chrome extension, span telecommunications, consumer loyalty platforms, and cryptocurrency tooling, indicating broad targeting rather than sector-specific campaigns. The Shai-Hulud campaign, specifically attributed to npm ecosystem operations, represents a named threat cluster within this broader trend. CWEs mapped to the threat cluster (CWE-1104, CWE-494, CWE-829, CWE-693) collectively describe the same root failure: organizations consuming third-party code without adequate integrity verification, provenance validation, or dependency auditing.

Confidence note: Quantitative figures cited (454,600 packages, 44-day exploit window, 28.3% 24-hour exploitation rate) derive from secondary aggregation sources rated T3. These figures are directionally credible and consistent with observed trends, but should be validated against NVD, CISA, or the Chainguard primary publication before use in formal risk reporting or board presentations.

Action Checklist

1. Step 1: Assess exposure, audit your software bill of materials (SBOM) for direct and transitive dependencies on npm and PyPI packages; flag any packages introduced or updated in the past 90 days that lack verified publisher provenance
2. Step 2: Review controls, verify that your CI/CD pipeline enforces dependency integrity checks (lockfile pinning, hash verification, package signing via Sigstore or equivalent); confirm that secrets scanning tools cover CI/CD environment variables, not just source code
3. Step 3: Update threat model, incorporate the Shai-Hulud campaign TTP cluster and AI-assisted exploitation acceleration into your threat register; update mean-time-to-patch targets to reflect a 44-day (or sub-24-hour for critical CVEs) exploitation window rather than legacy assumptions
4. Step 4: Communicate findings, brief development leads and platform engineering on the layered dependency confusion technique; brief leadership on the business risk of an industrialized supply chain attack volume that has outpaced manual review capacity
5. Step 5: Monitor developments, track Chainguard's Unchained blog (https://www.chainguard.dev/unchained) for follow-up package disclosures and supply chain threat trend updates; monitor CISA's Known Exploited Vulnerabilities catalog for CVEs entering the 24-hour exploitation cohort relevant to your asset inventory

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and legal if SBOM audit identifies a confirmed malicious package that was executed in a production build environment, any CI/CD secrets (cloud provider keys, signing certificates, API tokens) were exposed to postinstall scripts from unverified packages, or if the Trust Wallet Chrome extension or any Rakuten/Kaikatsu Club-adjacent dependency is present in environments handling PII, PHI, or payment card data triggering breach notification obligations under GDPR, CCPA, or PCI-DSS.
Recovery Notes	Post-containment, rotate all secrets (API keys, cloud credentials, signing certificates, webhook tokens) that were present as environment variables in any CI/CD pipeline that executed an unverified or flagged package, treating them as fully compromised regardless of whether exfiltration is confirmed — the Shai-Hulud campaign's postinstall script vector means exposure is simultaneous with install. Re-pin all dependencies to verified hashes, rebuild all production artifacts from a clean pipeline with the hardened controls from Step 2 in place, and verify artifact integrity using Sigstore or equivalent before re-deployment. Maintain enhanced monitoring of outbound DNS and HTTPS from build infrastructure and production systems for 30 days post-remediation, specifically watching for beaconing patterns (regular-interval connections to recently-registered domains) consistent with delayed-activation malicious package payloads.
Forensic Artifacts	npm postinstall script execution logs: check npm debug log at ~/.npm/_logs/*.log and CI/CD stdout for any 'postinstall' script execution events tied to packages flagged in the SBOM audit — malicious packages in this campaign category embed credential harvesting code in package.json 'scripts.postinstall' that executes automatically on 'npm install' | PyPI package cache and install receipts: examine ~/.cache/pip/wheels/ and site-packages/.dist-info/RECORD files for packages installed from unexpected indexes; cross-reference install timestamps in RECORD files against your CI/CD pipeline run timestamps to identify packages pulled during suspicious build windows | CI/CD environment variable access logs: in GitHub Actions, retrieve the workflow run logs from the Actions tab for any job that ran 'npm install' or 'pip install' without hash verification — look for unexpected outbound HTTP requests in the runner network logs, which would indicate a postinstall script attempting to exfiltrate secrets to attacker-controlled infrastructure | DNS query logs from build servers and developer workstations: query DNS resolver logs or firewall DNS logs for first-seen domain lookups occurring within seconds of a package install event — the Shai-Hulud campaign and similar supply chain attacks use postinstall beaconing to confirm successful compromise, producing anomalous first-seen FQDN queries from build infrastructure | Chrome extension storage and network activity for Trust Wallet extension: if the Trust Wallet Chrome extension (or any flagged browser extension) is deployed in managed environments, capture the extension's IndexedDB storage at chrome://extensions > developer mode > inspect views, and review Chrome's net-internals logs (chrome://net-internals/#events) for any anomalous WebSocket or HTTPS connections to non-official Trust Wallet infrastructure that would indicate the compromised extension version was present

Per-Action IR Details

Step 1: Assess exposure — audit your software bill of materials (SBOM) for direct and transitive dependencies on npm and PyPI packages; flag any packages introduced or updated in the past 90 days that

lack verified publisher provenance

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: establishing IR capability and asset visibility before an incident occurs

Controls: NIST SI-2 (Flaw Remediation), NIST CM-8 (System Component Inventory), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Run 'npm audit --json > npm_audit.json' and 'pip-audit --output-format json > pip_audit.json' against each project. For SBOM generation without enterprise tooling, use Syft (free, Anchore): 'syft dir:. -o spdx-json > sbom.json'. Cross-reference package names and versions against the OSS Index (Sonatype) free API. Flag any package where the publisher account was created within the last 90 days or where the package name closely resembles a known internal package (dependency confusion indicator) using a simple Python diff script against your internal package registry list.

Evidence: Before auditing, snapshot current state to establish a forensic baseline: capture 'pip freeze > pip_freeze_baseline.txt' and 'npm list --all --json > npm_tree_baseline.json' from each build environment. Preserve CI/CD pipeline logs showing when each dependency version was first introduced — specifically npm/PyPI install logs with timestamps from the past 90 days. For PyPI, check ~/.cache/pip/ and for npm check node_modules/.package-lock.json for install timestamps that may reveal when a malicious package update was pulled. If using GitHub Actions, preserve .github/workflows/ YAML files and the Actions run logs from the same 90-day window.

Step 2: Review controls — verify that your CI/CD pipeline enforces dependency integrity checks (lockfile pinning, hash verification, package signing via Sigstore or equivalent); confirm that secrets scanning tools cover CI/CD environment variables, not just source code

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: implementing preventive controls and hardening the environment to reduce IR burden

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST SA-12 (Supply Chain Protection), NIST CM-3 (Configuration Change Control), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Enforce lockfile integrity with 'npm ci' (not 'npm install') in all pipeline stages — 'npm ci' fails if package-lock.json is missing or mismatched, blocking silent dependency substitution. For PyPI, use 'pip install --require-hashes -r requirements.txt' with hashes pre-computed via 'pip-compile --generate-hashes'. Integrate Sigstore cosign (free) for verifying signed package artifacts: 'cosign verify-blob --certificate --signature '. For secrets scanning covering CI/CD environment variables, deploy Gitleaks (free) with a pre-commit hook and add a pipeline stage running 'gitleaks detect --source . --report-format json'. Trufflehog (free) can scan GitHub Actions secrets exposure: 'trufflehog github --repo '.

Evidence: Before modifying pipeline configuration, preserve the current state of all CI/CD pipeline definition files (e.g., .github/workflows/*.yml, .gitlab-ci.yml, Jenkinsfile, .circleci/config.yml) with 'git log --all --full-history -- .github/workflows/ > pipeline_git_history.txt'. Capture the current package-lock.json and requirements.txt with their git commit hashes. For the Shai-Hulud campaign specifically, examine CI/CD environment variable stores for any recently added variables containing base64-encoded strings, webhook URLs, or cloud provider credentials — these are the exfiltration artifacts malicious packages in this campaign inject into the build environment via postinstall scripts.

Step 3: Update threat model — incorporate the Shai-Hulud campaign TTP cluster and AI-assisted exploitation acceleration into your threat register; update mean-time-to-patch targets to reflect a 44-day (or sub-24-hour for critical CVEs) exploitation window rather than legacy assumptions

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: updating IR policies, playbooks, and threat intelligence to reflect current adversary capability

Controls: NIST RA-3 (Risk Assessment), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST IR-8 (Incident Response Plan), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Document the Shai-Hulud TTP cluster against the MITRE ATT&CK Supply Chain Compromise technique (T1195) and its sub-technique Compromise Software Dependencies and Development Tools (T1195.001). Map AI-assisted exploit acceleration to T1588.006 (Obtain Capabilities: Vulnerabilities) and T1190 (Exploit Public-Facing Application) with the updated 44-day window as a threat register parameter. Use the free MITRE ATT&CK Navigator (https://mitre-attack.github.io/attack-navigator/) to create a layer file documenting these TTPs. Update your SLA policy document to define 'critical' patches as requiring deployment within 24 hours for any CVE appearing in CISA KEV, and 14 days for High severity CVEs given the compressed 44-day mean exploitation window.

Evidence: Before finalizing the updated threat model, gather quantitative baseline evidence from your environment: pull the last 12 months of patch deployment timestamps from your patch management system to document your actual current mean-time-to-patch. Query your dependency update history to determine how many npm/PyPI package updates occurred without human review. If you have proxy or DNS logs, search for any historical connections to known Shai-Hulud campaign infrastructure — the campaign has been associated with exfiltration over DNS and HTTPS to attacker-controlled domains registered to mimic legitimate package registry infrastructure. This baseline documents your current exposure gap against the 44-day exploitation window.

Step 4: Communicate findings — brief development leads and platform engineering on the layered dependency confusion technique; brief leadership on the business risk of an industrialized supply chain attack volume that has outpaced manual review capacity

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: communicating incident scope, impact, and analysis findings to appropriate stakeholders

Controls: NIST IR-4 (Incident Handling), NIST IR-6 (Incident Reporting), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Prepare a one-page technical brief for development leads that includes: (1) a concrete example of dependency confusion using the npm/PyPI naming pattern this campaign exploits — an attacker registers 'company-internal-utils' on public PyPI after discovering the name from a leaked requirements.txt, and 'pip install' resolves the public malicious version over the internal one if registry priority is misconfigured; (2) a live demonstration using 'pip install --dry-run' showing which registry wins under current configuration. For leadership, translate the 454,600 malicious package/year figure into a per-week rate (approximately 8,742/week) to contextualize why manual review is structurally insufficient and quantify the business risk in terms of a potential software supply chain incident like the 2020 SolarWinds event.

Evidence: Collect evidence to support the brief: generate a dependency confusion risk report by diffing your internal package registry namespace against public npm and PyPI namespaces using a script querying the public registry APIs. Document any namespace collisions found — these are direct evidence of existing exposure to the layered dependency confusion vector used in the Shai-Hulud campaign. Preserve this report as a timestamped artifact for the communication record and for any subsequent regulatory or audit requirements under NIST IR-6 (Incident Reporting).

Step 5: Monitor developments — track Chainguard's Unchained publication and The Hacker News for follow-up package disclosures; monitor CISA's Known Exploited Vulnerabilities catalog for CVEs entering the 24-hour exploitation cohort relevant to your asset inventory

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: continuous monitoring, threat intelligence integration, and adverse event correlation

Controls: NIST SI-4 (System Monitoring), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Automate CISA KEV monitoring with a free cron job pulling the KEV JSON feed ('curl https://www.cisa.gov/sites/default/files/feeds/known_exploited_vulnerabilities.json') and diffing against yesterday's

snapshot, then alerting via email or Slack webhook when new CVEs appear that match packages in your SBOM. For Chainguard Unchained and new malicious package disclosures, configure an RSS feed aggregator (FreshRSS, free self-hosted) to consolidate Chainguard, CISA, and OSS security advisory feeds. Implement osquery on build servers with a query monitoring package installation events: 'SELECT name, version, install_time FROM deb_packages' (Linux) or equivalent, scheduled every 15 minutes, with results shipped to a central log file for daily diff review by the 2-person team.

Evidence: Establish detection baselines before new package disclosures occur: snapshot current npm and PyPI package versions in all production and CI/CD environments and store with SHA-256 hashes. For the Trust Wallet Chrome extension compromise referenced in this campaign, if your organization uses browser extensions in managed environments, export the current extension inventory from Chrome policy logs or via 'chrome://extensions' export and preserve it. Monitor outbound DNS queries from build servers for any newly observed domains — malicious packages in this campaign category frequently beacon on first install via postinstall scripts, producing DNS queries to attacker-controlled infrastructure that would appear as first-seen domains in DNS logs.

Detection Guidance

Detection for this threat cluster requires coverage across three layers: repository ingestion, runtime behavior, and credential telemetry.

At the repository and build layer: alert on new or updated dependencies not present in your lockfile baseline; flag packages with names closely resembling internal or popular public packages (typosquatting pattern matching); detect packages that make outbound network connections during installation or post-install script execution, this is anomalous behavior for legitimate libraries. Chainguard's 2026 supply chain threat analysis specifically implicates packages that establish persistent relay connections; network flow logs from build agents are the primary detection surface.

At the runtime and process layer: hunt for scripting interpreter invocations (T1059) originating from dependency installation directories or package manager processes; look for file system enumeration (T1083) by processes that should have no legitimate reason to traverse home directories, .ssh folders, or CI/CD credential stores. Monitor for defense impairment signals, specifically, any process disabling or modifying endpoint agent configuration (T1562.001).

At the credential and secrets layer: audit CI/CD secret stores for access patterns inconsistent with pipeline execution schedules; deploy secrets scanning on all repository commits and pull requests; monitor for API token usage from geographic locations or IP ranges inconsistent with your developer base, exfiltrated tokens will be used from attacker infrastructure, not developer endpoints.

For the GPT-proxy backdoor class specifically: inspect outbound DNS and TLS connections from build and development environments for resolutions to attacker-controlled infrastructure endpoints. Legitimate AI API calls from developer tools should resolve to known provider endpoints (OpenAI, Anthropic, etc.); unexpected resolutions warrant immediate investigation.

Hunting hypothesis: developer workstations or CI runners generating outbound connections to unfamiliar LLM API endpoints within 48 hours of a dependency update event.

Indicators of Compromise

Type	Value	Context	Confidence
TOOL	`Pending — refer to Chainguard Unchained (2026-the-year-of-ai-assisted-attacks) for published package names and hashes`	Malicious npm and PyPI package names, SHA hashes, and associated publisher accounts identified in Chainguard research; specific values not present in aggregated source text provided	LOW
TOOL	`Pending — refer to Aikido Security blog post (gpt-proxy-backdoor-npm-pypi-chinese-llm-relay) for published indicators`	GPT-proxy backdoor packages: package names, registry identifiers, and C2/relay endpoint domains published by Aikido Security; specific values not present in aggregated source text provided	LOW
TOOL	`Pending — refer to The Hacker News (2026/02/malicious-npm-packages-harvest-crypto) for published indicators`	Package names and associated payload hashes for npm packages harvesting cryptographic keys, CI/CD secrets, and API tokens; specific values not present in aggregated source text provided	LOW

Framework Mappings

MITRE-ATTACK

- **T1657** — Financial Theft
- **T1552.001** — Credentials In Files
- **T1078** — Valid Accounts
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1554** — Compromise Host Software Binary
- **T1083** — File and Directory Discovery
- **T1195.002** — Compromise Software Supply Chain
- **T1562.001** — Disable or Modify Tools
- **T1566** — Phishing
- **T1059** — Command and Scripting Interpreter
- **T1486** — Data Encrypted for Impact

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes

- **SI-7** — Software, Firmware, and Information Integrity
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **SA-4** — Acquisition Process
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan
- **SC-13** — Cryptographic Protection
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A06:2021** — Vulnerable and Outdated Components
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **16.4** — Establish and Manage an Inventory of Third-Party Software Components
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **15.1** — Establish and Maintain an Inventory of Service Providers

ISO-27001-2022

- **A.5.34** — Privacy and protection of personal information
- **A.5.21** — Managing information security in the ICT supply chain
- **A.8.24** — Use of cryptography

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program
- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

SOC2-TSC

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1657	Financial Theft	Impact

Technique ID	Technique Name	Tactic
T1552.001	Credentials In Files	Credential-Access
T1078	Valid Accounts	Defense-Evasion
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1554	Compromise Host Software Binary	Persistence
T1083	File and Directory Discovery	Discovery
T1195.002	Compromise Software Supply Chain	Initial-Access
T1562.001	Disable or Modify Tools	Defense-Evasion
T1566	Phishing	Initial-Access
T1059	Command and Scripting Interpreter	Execution
T1486	Data Encrypted for Impact	Impact

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/05/2026-year-of-ai-assisted-attacks....	T3
2026: The year of AI-assisted attacks - Chainguard	https://www.chainguard.dev/fr-FR/unchained/2026-the-year-of-ai-assi...	T3
A wave of attacks is using layered npm dependencies to ... - Facebook	https://www.facebook.com/thehackernews/posts/-a-wave-of-attacks-is-...	T3
Malicious npm Packages Harvest Crypto Keys, CI Secrets, and API ...	https://thehackernews.com/2026/02/malicious-npm-packages-harvest-cr...	T3
GPT-Proxy Backdoor in npm and PyPI turns Servers into Chinese ...	https://www.aikido.dev/blog/gpt-proxy-backdoor-npm-pypi-chinese-llm...	T3

Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-05-05 08:36 UTC by TJS Security Command Center