INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-20 18:55 UTC

Dual-Vector Threat: TamperedChef Trojanized Productivity Apps and Shai-Hulud 2.0 npm Supply Chain Worm Target Enterprise Endpoints and Dev Pipelines

THREAT CAMPAIGN | CRITICAL | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0346
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	7.5
Affected Products	Windows endpoints; trojanized productivity apps (AppSuite PDF, DocuFlex, Calendaromatic, CrystalPDF, Easy2Convert, PDF-Ezy, JustAskJacky, GoCookMate, RocketPDFPro, ManualReaderPro); npm ecosystem; GitHub repositories; CI/CD pipelines
Published	2026-05-20T10:00:46+00:00
Discovery Source	Rss:T1 Threatintel

Executive Summary

Two active malware campaigns are targeting enterprise Windows endpoints and software development pipelines simultaneously. The TamperedChef operation has distributed 4,000+ malware samples hidden inside trojanized productivity applications that delay malicious activity for weeks to evade detection, while the Shai-Hulud 2.0 npm worm has compromised tens of thousands of GitHub repositories, stealing credentials and destroying developer environments when theft fails. Organizations running the affected productivity apps or consuming open-source npm packages face credential theft, remote access compromise, and potential data destruction, with a secondary risk that compromised developer systems become entry points into broader enterprise infrastructure.

Technical Analysis

Unit 42 has identified three TamperedChef clusters (CL-CRI-1089, CL-UNK-1090, CL-UNK-1110; Unit 42 internal identifiers) distributing trojanized versions of at least 10 productivity applications: AppSuite PDF, DocuFlex, Calendaromatic, CrystalPDF, Easy2Convert, PDF-Ezy, JustAskJacky, GoCookMate, RocketPDFPro, and ManualReaderPro. The malware employs extended dormancy (weeks to months) before deploying infostealers, RATs, or proxy tooling, specifically to defeat behavioral sandboxing and short-window EDR

analysis. MITRE techniques include T1204.002 (malicious file execution), T1553.002 (code signing bypass), T1547 (boot/logon autostart persistence), T1562.001 (defense evasion via security tool impairment), T1027 (obfuscation), T1140 (deobfuscation at runtime), T1083 (file and directory discovery), T1041 (exfiltration over C2 channel), and T1059 (command and scripting interpreter execution). CWE-506 (embedded malicious code), CWE-829 (inclusion of functionality from untrusted control sphere), CWE-494 (download of code without integrity check), CWE-502 (deserialization of untrusted data), and CWE-312 (cleartext storage of sensitive information) apply across both campaigns. The Shai-Hulud 2.0 npm worm (T1195.001, supply chain compromise) shifts infection to the npm pre-install lifecycle hook, bypassing post-install detection tooling. It propagates by compromising GitHub repositories and replicating into dependent projects. When credential theft fails, a destructive fallback (T1485, data destruction) wipes the developer's home directory. Credential theft targets (T1552, T1078) include tokens, SSH keys, and environment variables commonly stored in developer workstations. The convergence risk: a CI/CD pipeline or developer endpoint compromised by either campaign becomes a lateral pivot vector into the other. No CVE identifiers are assigned to these campaigns. No vendor patch is applicable; mitigations are behavioral and architectural. Detection surfaces identified by Unit 42 include Palo Alto Cortex XDR, XSIAM, and Prisma Browser. Corroborated by Truesec analysis of the AppSuite PDF variant.

Action Checklist

1. Step 1: Containment, Immediately block execution of and quarantine any instance of the following applications enterprise-wide: AppSuite PDF, DocuFlex, Calendaromatic, CrystalPDF, Easy2Convert, PDF-Ezy, JustAskJacky, GoCookMate, RocketPDFPro, ManualReaderPro. Use application control policies (NIST CM-7 Configuration Change Control, CIS 2.3) to deny execution. For npm environments, freeze all package installs and audit package-lock.json and yarn.lock files for recently added or modified pre-install hooks. Isolate any developer workstation or CI/CD runner with confirmed exposure.
2. Step 2: Detection, Query EDR telemetry for process trees originating from any of the 10 named applications, specifically looking for child processes spawned after dormancy gaps (no activity for days or weeks post-install, then sudden outbound connections or registry writes). Hunt for npm pre-install script execution in CI/CD logs that invokes curl, wget, PowerShell, or base64-encoded payloads (NIST AU-6 Audit Review Analysis Monitoring, CIS 8.2). Search SIEM for T1547 indicators: new autostart registry keys (HKCU\Software\Microsoft\Windows\CurrentVersion\Run) written by productivity app processes. Check for T1562.001 activity: security tool process termination events correlated with app execution timestamps. Audit GitHub repositories for unexpected commits modifying package.json pre-install fields (NIST AU-12 Audit Log Generation).
3. Step 3: Eradication, Remove all identified trojanized applications and purge associated files, registry entries, and scheduled tasks. For Shai-Hulud-affected npm packages, remove compromised packages from registries and pipelines, rotate all secrets, tokens, SSH keys, and environment variables accessible from affected developer environments (NIST IA-5 Authentication Management). Revoke and reissue any code-signing certificates or API tokens that may have been present on compromised systems. For CI/CD runners, rebuild from a clean, verified base image rather than attempting in-place remediation.
4. Step 4: Recovery, Validate remediation by re-running behavioral detection queries post-cleanup and confirming no recurrence of autostart entries or anomalous child processes. Restore developer environments from pre-compromise snapshots or clean images only. Verify npm package integrity using lockfile hash validation and npm audit before re-enabling pipelines (NIST SI-7 Software Firmware and Information Integrity, CIS 7.3). Increase monitoring duration to at least 90 days given TamperedChef's

documented weeks-to-months dormancy window. Confirm outbound C2 channels are severed by monitoring for any resumed connections to previously identified suspicious domains or IPs.

5. Step 5: Post-Incident, This campaign exposed gaps in software acquisition controls, pre-install script visibility, and long-window behavioral detection. Implement verified software inventory and acquisition policies restricting productivity app sources to approved, cryptographically verified vendors (NIST CM-11 Software Version Control, CIS 2.1 and 2.2). Establish mandatory code review for pre-install and post-install npm script hooks in all consumed packages (CIS 4.6). Extend behavioral detection windows beyond standard 24-72 hour sandbox analysis to address dormancy evasion. Implement developer workstation secrets management to eliminate plaintext credential storage in environment variables and dotfiles (CWE-312, NIST IA-5). Review and enforce CI/CD pipeline least-privilege principles to limit blast radius of future supply chain compromises (NIST AC-6 Least Privilege).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership, legal, and executive stakeholders immediately if any of the following are confirmed: (1) Shai-Hulud 2.0 exfiltrated secrets from a CI/CD runner with write access to production repositories or package registries, indicating potential downstream customer supply chain impact requiring breach notification analysis; (2) TamperedChef payloads are identified on endpoints with access to PII, PHI, PCI-scoped systems, or regulated data stores triggering statutory breach notification timelines; (3) code-signing certificates present on compromised developer workstations cannot be confirmed as uncompromised, requiring emergency certificate revocation to prevent signed malware distribution; or (4) the responding team lacks the forensic capability to definitively scope Shai-Hulud credential exposure across all developer environments, requiring engagement of an external DFIR retainer.
Recovery Notes	Restore developer environments exclusively from pre-compromise disk snapshots or clean, hash-verified base images — do not attempt in-place remediation of Shai-Hulud-affected workstations given the worm's documented credential harvesting and environment destruction behaviors. Before re-enabling any CI/CD pipeline, validate every package in the dependency tree against its published lockfile hash using 'npm ci' (never 'npm install') and confirm 'npm audit' returns zero critical findings against the restored package set. Maintain elevated behavioral monitoring for a minimum of 90 days post-recovery across all previously exposed endpoints and pipelines, with weekly review of autostart registry keys, scheduled tasks, and outbound network connections to flag any TamperedChef dormant sample reactivation that survived initial eradication.

Forensic Artifacts	Windows Installer logs (%SystemRoot%\Logs\CBS\CBS.log and %TEMP%\MSI*.log) recording install timestamps for all 10 named trojanized apps — establishes the TamperedChef dormancy baseline by anchoring the timeline between installation and first observed malicious activity | Sysmon Event ID 1 (ProcessCreate) and Event ID 3 (NetworkConnect) logs filtered to parent process names matching the 10 named app executables — captures the delayed payload execution chain and C2 callback timing specific to TamperedChef's weeks-to-months dormancy mechanism | Registry export of HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\Software\Microsoft\Windows\CurrentVersion\Run captured at time of containment — preserves TamperedChef T1547 autostart persistence keys written by the trojanized productivity app processes before eradication removes them | npm package cache artifacts from ~/.npm/_cacache/ and node_modules directories on Shai-Hulud-affected developer workstations, along with original package-lock.json and yarn.lock files showing pre-install hook fields — provides the malicious package tarballs and hook payloads for reverse engineering and IOC extraction | CI/CD runner stdout/stderr logs from npm install operations capturing Shai-Hulud 2.0 pre-install script execution, specifically log lines containing invocations of curl, wget, powershell, or base64-encoded strings — documents the credential exfiltration stage and identifies which secrets were in scope at time of compromise

Per-Action IR Details

Step 1: Containment — Immediately block execution of and quarantine any instance of the following applications enterprise-wide: AppSuite PDF, DocuFlex, Calendaromatic, CrystalPDF, Easy2Convert, PDF-Ezy, JustAskJacky, GoCookMate, RocketPDFPro, ManualReaderPro. Use application control policies (NIST CM-7, CIS 2.3) to deny execution. For npm environments, freeze all package installs and audit package-lock.json and yarn.lock files for recently added or modified pre-install hooks. Isolate any developer workstation or CI/CD runner with confirmed exposure.

NIST Phase: Containment

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

Controls: NIST CM-7 (Least Functionality) — deny execution of the 10 named trojanized productivity apps via application allowlist, NIST IR-4 (Incident Handling) — execute containment actions per incident response plan, NIST AC-3 (Access Enforcement) — enforce execution deny policies at OS and GPO level, CIS 2.3 (Address Unauthorized Software) — treat all 10 named apps as unauthorized; remove or exception-document immediately, CIS 4.5 (Implement and Manage a Firewall on End-User Devices) — block outbound C2 channels from endpoints where these apps are installed

Compensating: Without enterprise EDR, deploy AppLocker or Windows Defender Application Control (WDAC) via GPO to deny execution of AppSuite PDF, DocuFlex, Calendaromatic, CrystalPDF, Easy2Convert, PDF-Ezy, JustAskJacky, GoCookMate, RocketPDFPro, and ManualReaderPro by publisher hash and path rule. Run the following PowerShell one-liner across endpoints via PSRemoting to identify installations: Get-WmiObject Win32_Product | Where-Object {$_.Name -match 'AppSuite|DocuFlex|Calendaromatic|CrystalPDF|Easy2Convert|PDF-Ezy|JustAskJacky|GoCookMate|RocketPDF|ManualReader'} | Select-Object Name,Version,InstallDate. For npm, run 'cat package-lock.json | python3 -c "import sys,json; [print(k,v.get(\"scripts\",{}).get(\"preinstall\",\"\")) for k,v in json.load(sys.stdin).get(\"packages\",{}).items() if v.get(\"scripts\",{}).get(\"preinstall\")]"' to surface any pre-install hooks before freezing installs.

Evidence: Before blocking, image or preserve: (1) Windows Installer logs at %SystemRoot%\Logs\CBS\CBS.log and %TEMP%\MSI*.log capturing install timestamps for the 10 named apps to establish the dormancy baseline; (2) full directory listings of %ProgramFiles%, %ProgramFiles(x86)%, %LocalAppData%, and %AppData%\Roaming for files deposited by the named apps; (3) registry snapshot of HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM\Software\Microsoft\Windows\CurrentVersion\Run to capture any autostart keys already written before containment removes them; (4) package-lock.json and yarn.lock from all active developer workstations to preserve the pre-freeze state of pre-install hook fields before they can be modified; (5) network connection state via 'netstat -anob'

or Sysinternals TCPView to capture any active outbound connections from the named app processes at the moment of containment.

Step 2: Detection — Query EDR telemetry for process trees originating from any of the 10 named applications, specifically looking for child processes spawned after dormancy gaps (no activity for days or weeks post-install, then sudden outbound connections or registry writes). Hunt for npm pre-install script execution in CI/CD logs that invokes curl, wget, PowerShell, or base64-encoded payloads (NIST AU-6, CIS 8.2). Search SIEM for T1547 indicators: new autostart registry keys (HKCU\Software\Microsoft\Windows\CurrentVersion\Run) written by productivity app processes. Check for T1562.001 activity: security tool process termination events correlated with app execution timestamps. Audit GitHub repositories for unexpected commits modifying package.json pre-install fields (AU-12).

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting) — review process creation and registry audit logs for dormant-then-active behavior patterns from the 10 named apps, NIST AU-12 (Audit Record Generation) — ensure process creation auditing (Event ID 4688) and registry modification auditing are enabled on all endpoints running the named apps, NIST SI-4 (System Monitoring) — monitor for T1547 autostart persistence and T1562.001 defense evasion correlated with TamperedChef app execution timestamps, CIS 8.2 (Collect Audit Logs) — validate that CI/CD runner logs capturing npm script execution are being collected and retained, MITRE ATT&CK T1547 (Boot or Logon Autostart Execution) — hunt for registry Run key writes by the named productivity app processes, MITRE ATT&CK T1562.001 (Impair Defenses: Disable or Modify Tools) — correlate security tool termination events against TamperedChef app execution timestamps, MITRE ATT&CK T1195.002 (Supply Chain Compromise: Compromise Software Supply Chain) — detect Shai-Hulud 2.0 pre-install hook execution in CI/CD pipeline logs

Compensating: Without SIEM/EDR, deploy Sysmon with a configuration that enables ProcessCreate (Event ID 1), NetworkConnect (Event ID 3), RegistryEvent (Event IDs 12/13/14), and ProcessTerminate (Event ID 5). Use the following PowerShell query against Windows Security Event Log to find process creation chains from the named apps: Get-WinEvent -LogName Security | Where-Object {$_.Id -eq 4688 -and $_.Message -match 'AppSuitePDF|DocuFlex|Calendaromatic|CrystalPDF|Easy2Convert|PDFEzy|JustAskJacky|GoCookMate|RocketPDF|ManualReader'} | Select-Object TimeCreated,Message. For dormancy gap detection without EDR, diff Windows Security Event Log timestamps between first recorded app execution and first outbound network event using Sysmon Event ID 3 filtered to the app's process name — any gap exceeding 48 hours post-install is a TamperedChef behavioral indicator. For GitHub pre-install hook auditing, run 'git log --all --diff-filter=M -- package.json' on all developer repos to surface commits modifying pre-install fields. Deploy this Sigma rule concept: detect cmd.exe or powershell.exe spawned as child of any of the 10 named app executables with parent image path matching %ProgramFiles% or %LocalAppData% install directories.

Evidence: Collect before analysis is complete: (1) Windows Security Event Log Event ID 4688 (Process Creation) with full command line logging enabled, filtered to parent process names matching the 10 named app executables — this captures TamperedChef's delayed payload execution chain; (2) Sysmon Event ID 3 (NetworkConnect) from the same process names to identify C2 callback timing relative to install date, confirming the weeks-to-months dormancy window; (3) Windows Security Event ID 4657 (Registry Value Modified) and 4698 (Scheduled Task Created) events written by the named app processes — TamperedChef persistence artifacts; (4) CI/CD runner stdout/stderr logs from npm install operations showing pre-install script execution, specifically lines containing curl, wget, powershell, or base64 — Shai-Hulud 2.0 exfiltration stage evidence; (5) GitHub repository audit logs (Settings > Security > Audit Log) for unexpected commits to package.json or .npmrc files in the days preceding confirmed compromise, and git diff output for pre-install hook field changes.

Step 3: Eradication — Remove all identified trojanized applications and purge associated files, registry entries, and scheduled tasks. For Shai-Hulud-affected npm packages, remove compromised packages from registries and pipelines, rotate all secrets, tokens, SSH keys, and environment variables accessible from affected developer environments (NIST IA-5, D3-CRO). Revoke and reissue any code-signing certificates or API tokens that may have been present on compromised systems. For CI/CD runners, rebuild from a clean,

verified base image rather than attempting in-place remediation.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IA-5 (Authenticator Management) — rotate all credentials, SSH keys, API tokens, and environment variable secrets exposed on Shai-Hulud-compromised developer workstations and CI/CD runners, NIST CM-7 (Least Functionality) — remove the 10 named trojanized apps and verify no residual executables or DLLs remain in install directories or temp paths, NIST SI-2 (Flaw Remediation) — treat trojanized app removal and npm package purge as a remediation action requiring post-removal validation, NIST AC-2 (Account Management) — revoke all service accounts, PATs, and OAuth tokens accessible from compromised CI/CD runners before rebuilding, CIS 5.2 (Use Unique Passwords) — enforce credential uniqueness during rotation to prevent reuse of potentially harvested secrets, CIS 6.2 (Establish an Access Revoking Process) — immediately revoke GitHub, npm registry, and cloud provider tokens exposed on Shai-Hulud-affected systems

Compensating: For teams without enterprise endpoint management, execute trojanized app removal via: (1) wmic product where 'name like "%AppSuite%" or name like "%DocuFlex%" or name like "%Calendaromatic%" or name like "%CrystalPDF%" or name like "%Easy2Convert%" or name like "%PDF-Ezy%" or name like "%JustAskJacky%" or name like "%GoCookMate%" or name like "%RocketPDF%" or name like "%ManualReader%"' call uninstall; (2) follow with registry cleanup: reg delete 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run' /v [identified key name] /f and schtasks /delete /tn [identified task name] /f; (3) for Shai-Hulud credential rotation on developer workstations, audit dotfiles (.env, .bashrc, .zshrc, ~/.npmrc, ~/.gitconfig) and CI/CD environment variable stores (GitHub Actions Secrets, GitLab CI variables) for harvested plaintext tokens — rotate every secret found, not just those confirmed accessed; (4) rebuild CI/CD runners using a pinned, hash-verified base Docker image rather than latest tags, and validate with 'docker inspect --format={{.Id}} ' against a known-good digest.

Evidence: Before eradication begins, preserve: (1) full recursive directory listing and SHA-256 hashes of all files in the named app install directories (%ProgramFiles%\[AppName]\, %LocalAppData%\[AppName]\) to document TamperedChef's dropped payload file artifacts for threat intelligence sharing; (2) export of all scheduled tasks via 'schtasks /query /fo XML /v > tasks_pre_eradication.xml' to capture any persistence tasks created by the trojanized apps; (3) registry export of HKCU\Software\Microsoft\Windows\CurrentVersion\Run, HKLM\Software\Microsoft\Windows\CurrentVersion\Run, and HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce in full before removal; (4) npm package cache contents from ~/.npm/_cacache/ and node_modules directories on affected developer workstations, preserving Shai-Hulud-compromised package tarballs for malware analysis; (5) CI/CD runner environment variable dumps (redacted for storage) documenting which secrets were accessible at time of Shai-Hulud execution, to scope the blast radius for downstream credential rotation.

Step 4: Recovery — Validate remediation by re-running behavioral detection queries post-cleanup and confirming no recurrence of autostart entries or anomalous child processes. Restore developer environments from pre-compromise snapshots or clean images only. Verify npm package integrity using lockfile hash validation and npm audit before re-enabling pipelines (NIST SI-7, CIS 7.3). Increase monitoring duration to at least 90 days given TamperedChef's documented weeks-to-months dormancy window. Confirm outbound C2 channels are severed by monitoring for any resumed connections to previously identified suspicious domains or IPs.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-7 (Software, Firmware, and Information Integrity) — validate npm package integrity via lockfile hash comparison and npm audit before restoring pipeline operations, NIST CP-10 (System Recovery and Reconstitution) — restore developer environments from verified pre-compromise snapshots or clean base images only, NIST SI-4 (System Monitoring) — extend behavioral monitoring window to minimum 90 days post-remediation specifically to account for TamperedChef's documented weeks-to-months delayed execution pattern, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — re-run post-cleanup detection queries for autostart registry keys and anomalous child process spawning from previously infected endpoints, CIS 7.3 (Perform Automated Operating System Patch Management) — ensure restored developer environment base images are fully patched before pipeline re-enablement,

CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) — re-validate asset inventory to confirm all affected endpoints are accounted for in recovery tracking

Compensating: For teams without EDR, implement post-recovery validation using: (1) a weekly scheduled PowerShell task querying Windows Security Event Log for Event ID 4688 filtering on the 10 named app process names — any hit post-eradication indicates reinfection or missed instance; (2) 'npm audit --audit-level=critical' and 'npm ci' (which enforces lockfile integrity) rather than 'npm install' for all pipeline restores, rejecting installs if lockfile hashes do not match; (3) deploy a simple Sysmon-based C2 detection rule monitoring NetworkConnect (Event ID 3) for connections to IPs and domains identified during the investigation, persisted as a custom Windows Event Log alert via Task Scheduler triggering on Sysmon Event ID 3 with matching DestinationIp or DestinationHostname; (4) use osquery with a scheduled query against the autostart_entries table ('SELECT * FROM startup_items') run daily for 90 days post-recovery, diffing output against a known-clean baseline captured immediately after eradication.

Evidence: During recovery validation, collect and retain: (1) post-eradication Sysmon Event ID 3 logs for 90 days filtered to previously identified C2 IP/domain indicators — TamperedChef's dormancy mechanism means payload reactivation from a missed sample is a realistic threat within this window; (2) npm audit JSON output ('npm audit --json > audit_post_recovery.json') from all restored pipelines at the moment of re-enablement, timestamped as the recovery baseline; (3) Windows Security Event ID 4688 process creation logs from restored endpoints for 30 days post-recovery to confirm no child processes are spawning from residual named app artifacts; (4) network flow logs (NetFlow or Windows Firewall logs) for outbound connections on ports 443, 80, and uncommon high ports from developer workstations for 90 days, specifically monitoring for beaconing patterns consistent with delayed C2 check-in behavior documented in TamperedChef samples.

Step 5: Post-Incident — This campaign exposed gaps in software acquisition controls, pre-install script visibility, and long-window behavioral detection. Implement verified software inventory and acquisition policies restricting productivity app sources to approved, cryptographically verified vendors (NIST CM-11, CIS 2.1, CIS 2.2). Establish mandatory code review for pre-install and post-install npm script hooks in all consumed packages (CIS 4.6). Extend behavioral detection windows beyond standard 24-72 hour sandbox analysis to address dormancy evasion. Implement developer workstation secrets management to eliminate plaintext credential storage in environment variables and dotfiles (CWE-312, D3-CH, NIST IA-5). Review and enforce CI/CD pipeline least-privilege principles to limit blast radius of future supply chain compromises (NIST AC-6, D3-UAP).

NIST Phase: Post Incident

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

Controls: NIST CM-11 (User-Installed Software) — restrict productivity application installation to cryptographically verified, allowlisted vendors to prevent future TamperedChef-style trojanized app distribution, NIST IA-5 (Authenticator Management) — implement secrets management tooling (HashiCorp Vault free tier, AWS Secrets Manager, or git-crypt) to eliminate plaintext credential storage in .env files and dotfiles exposed by Shai-Hulud 2.0, NIST AC-6 (Least Privilege) — restrict CI/CD pipeline runner permissions to minimum required scopes; Shai-Hulud's blast radius was amplified by runners with broad repository and registry access, NIST SI-2 (Flaw Remediation) — update vulnerability management process to include npm pre-install script review as a mandatory check in dependency onboarding procedures, NIST AU-2 (Event Logging) — update logging policy to capture npm script execution events in CI/CD pipeline logs as a mandatory audit event category, CIS 2.1 (Establish and Maintain a Software Inventory) — add all 10 named trojanized apps as permanently blocklisted entries in the software inventory, CIS 2.2 (Ensure Authorized Software is Currently Supported) — require cryptographic publisher verification for all productivity apps added to the authorized software list going forward, CIS 4.6 (Securely Manage Enterprise Assets and Software) — implement mandatory pre-install and post-install npm hook review as part of the software onboarding process, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — extend behavioral sandbox analysis windows beyond 72 hours to detect TamperedChef-style delayed activation; consider 7-14 day minimum for new productivity app evaluation, CIS 7.2 (Establish and Maintain a Remediation Process) — add supply chain compromise scenarios (trojanized apps, malicious npm packages) as named threat categories in the remediation strategy

Compensating: For teams without budget for commercial secrets management: (1) deploy git-secret (free, GPG-based) or sops (free, Cloud KMS or PGP-backed) to eliminate plaintext secrets in developer dotfiles and repo

configs — directly addresses Shai-Hulud's credential harvesting from .env and .npmrc files; (2) create a mandatory pre-merge GitHub Actions workflow (free for public repos, included in GitHub Free tier) that runs 'node -e "const p=require(\"./package.json\"); const hooks=[\"preinstall\",\"install\",\"postinstall\"]; hooks.forEach(h=>{if(p.scripts&&p.scripts[h]) process.exit(1)});"' to reject any package.json modifications introducing new pre/post-install hooks without explicit review; (3) implement YARA rules scanning npm package tarballs for base64-encoded payloads, curl/wget invocations in script fields, and obfuscated JS consistent with Shai-Hulud 2.0 patterns, run as a pre-install gate via a simple bash wrapper around 'npm pack'; (4) extend behavioral monitoring for new software to 14 days minimum using a free Sysmon + Windows Event Log collection pipeline, specifically targeting the dormancy-then-activation pattern that allowed TamperedChef to evade standard 24-72 hour sandbox windows.

Evidence: Post-incident evidence to compile for lessons learned and threat intelligence sharing: (1) full timeline reconstruction correlating app install timestamps (from Windows Installer logs) against first malicious activity timestamps (from Sysmon Event ID 1/3) for each affected endpoint — this documents TamperedChef's exact dormancy durations across your environment for detection tuning; (2) inventory of all plaintext secrets discovered in developer dotfiles and CI/CD environment variables during Shai-Hulud eradication, categorized by secret type (SSH key, API token, npm auth token, cloud credential) without values — for blast radius documentation and secrets hygiene baseline; (3) list of all GitHub repository commits touching package.json pre-install fields in the 90 days preceding detection, extracted from git audit logs — establishes the Shai-Hulud propagation timeline across your codebase; (4) DNS query logs and proxy logs for domains contacted by the named app processes during the dormancy-to-active transition — C2 infrastructure indicators for threat intelligence sharing with sector ISACs and CISA.

Detection Guidance

TamperedChef Detection:

• Hunt EDR telemetry for any of the 10 named productivity apps spawning child processes, especially cmd.exe, powershell.exe, wscript.exe, or mshta.exe, days or weeks after initial installation. This dormancy gap is the primary behavioral signature.

• Search Windows Event Log for Event ID 4688 (new process creation) with parent image paths matching the trojanized app directories.

• Hunt Registry for autostart persistence writes (T1547): new entries under HKCU\Software\Microsoft\Windows\CurrentVersion\Run or HKLM\Software\Microsoft\Windows\CurrentVersion\Run created by those app processes.

• Alert on T1562.001: security product process termination (e.g., antivirus or EDR service stops) within the execution context of any named application.

• Monitor outbound network connections from productivity app processes to non-CDN external IPs, particularly over non-standard ports.

Shai-Hulud Detection:

• In CI/CD logs, parse all npm install operations for pre-install script execution. Flag any pre-install hook invoking shell commands with base64-decoded payloads, remote downloads, or environment variable enumeration.

• Audit package.json files in your dependency tree for pre-install fields added or modified in recent commits.

• In GitHub audit logs, look for unexpected force-pushes or commits to package.json or .npmrc files, especially from accounts with recently changed credentials.

• Monitor developer endpoints for mass file deletion events in home directory paths (T1485 indicator).

Behavioral IOCs:

• Outbound connections to newly registered or low-reputation domains from npm install processes.

• SSH key reads or environment variable dumps (env, printenv, set) executed by package install scripts.

• Sudden disappearance of files under /home/* or C:\Users* correlated with npm activity.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://unit42.paloaltonetworks.com/tracking-tampered-chef-clusters/`	Unit 42 primary research on TamperedChef clusters CL-CRI-1089, CL-UNK-1090, CL-UNK-1110 — consult for full IOC list including hashes and C2 infrastructure	HIGH
URL	`https://unit42.paloaltonetworks.com/npm-supply-chain-attack/`	Unit 42 primary research on Shai-Hulud 2.0 npm worm — consult for package names, repository indicators, and pre-install script signatures	HIGH
URL	`https://www.truesec.com/hub/blog/tamperedchef-the-bad-pdf-editor`	Truesec corroborating analysis of AppSuite PDF variant of TamperedChef — includes additional behavioral indicators	HIGH

Framework Mappings

MITRE-ATTACK

- **T1204.002** — Malicious File
- **T1553.002** — Code Signing
- **T1072** — Software Deployment Tools
- **T1176** — Software Extensions
- **T1078** — Valid Accounts
- **T1027** — Obfuscated Files or Information
- **T1083** — File and Directory Discovery
- **T1547** — Boot or Logon Autostart Execution
- **T1562.001** — Disable or Modify Tools
- **T1041** — Exfiltration Over C2 Channel
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1485** — Data Destruction
- **T1140** — Deobfuscate/Decode Files or Information
- **T1059** — Command and Scripting Interpreter
- **T1566** — Phishing
- **T1553** — Subvert Trust Controls
- **T1552** — Unsecured Credentials

- **T1071** — Application Layer Protocol

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-7** — Software, Firmware, and Information Integrity
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection
- **CM-3** — Configuration Change Control
- **SI-10** — Information Input Validation
- **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
- **16.10** — Apply Secure Design Principles in Application Architectures
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers
- **8.2** — Collect Audit Logs

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
- **DE.CM-01** — Networks and network services are monitored

ISO-27001-2022

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1204.002	Malicious File	Execution
T1553.002	Code Signing	Defense-Evasion
T1072	Software Deployment Tools	Execution
T1176	Software Extensions	Persistence
T1078	Valid Accounts	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1083	File and Directory Discovery	Discovery
T1547	Boot or Logon Autostart Execution	Persistence
T1562.001	Disable or Modify Tools	Defense-Evasion
T1041	Exfiltration Over C2 Channel	Exfiltration
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1485	Data Destruction	Impact
T1140	Deobfuscate/Decode Files or Information	Defense-Evasion
T1059	Command and Scripting Interpreter	Execution
T1566	Phishing	Initial-Access
T1553	Subvert Trust Controls	Defense-Evasion
T1552	Unsecured Credentials	Credential-Access
T1071	Application Layer Protocol	Command-And-Control

Sources

Source	URL	Tier
Unit 42	https://unit42.paloaltonetworks.com/tracking-tampered-chef-clusters/	T3
	https://unit42.paloaltonetworks.com/tracking-tampered-chef-clusters/	T3
	https://unit42.paloaltonetworks.com/npm-supply-chain-attack/	T3

Source	URL	Tier
Malicious Appsuite PDF Editor Spreads Tamperedchef Malware	https://www.truesec.com/hub/blog/tamperedchef-the-bad-pdf-editor	T3
Secure Browser | Prisma Browser - Palo Alto Networks	https://www.paloaltonetworks.com/sase/prisma-browser	T3

Generated 2026-05-20 18:55 UTC by TJS Security Command Center