INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-03 06:20 UTC

Vidar Infostealer Distributed via SEO-Poisoned GitHub Repos Exploiting Claude Code Source Leak

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0140
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Developers and security researchers searching for Anthropic Claude Code leaked source; GitHub, npm ecosystem
Published	2026-04-02T16:30:55
Discovery Source	Rss

Executive Summary

Threat actors are exploiting media coverage of an accidental Anthropic Claude Code source leak to distribute Vidar infostealer malware through fake GitHub repositories. Developers and security researchers who downloaded from these repositories may have credential theft malware installed, putting corporate accounts, session tokens, and cryptocurrency wallets at risk. Organizations with developers actively following AI news are the primary exposure surface; a single compromised developer workstation may potentially yield access to source code repositories, cloud environments, and internal systems.

Technical Analysis

Campaign origin: Following a March 31, 2026 npm packaging error that exposed Anthropic Claude Code source code, unattributed threat actor 'idbzoomh' (tracked by Zscaler threat intelligence) created SEO-optimized GitHub repositories impersonating the leaked codebase to attract organic Google Search traffic. Attack chain: Victim clones or downloads from a fake repo; a Rust-based dropper executes (T1059, T1204.002), installs Vidar infostealer and GhostSocks SOCKS5 proxy (T1090, T1090.001). Vidar targets browser-stored credentials (T1555.003), session cookies (T1539), cryptocurrency wallets, and general credential stores (T1555). GhostSocks provides backconnect proxy capability, likely for C2 anonymization or persistent network access (T1071). Attacker pre-staged malware prior to victim interaction (T1608.001, T1608.003). No CVE, attack vector is social engineering and SEO manipulation, not a software vulnerability. Relevant CWEs: CWE-494 (Download of Code Without Integrity Check), CWE-506 (Embedded Malicious Code). MITRE coverage: T1566 (Phishing

via fake repo lure), T1195/T1195.002 (Supply Chain Compromise, developer toolchain targeting). No patch applies; this is a behavioral and detection problem. BleepingComputer and The Hacker News reporting confirmed by Zscaler threat intelligence; human validation of IOCs recommended before actioning.

Action Checklist

1. Step 1, Containment: Identify any developer or researcher workstations that downloaded content from GitHub repositories impersonating Claude Code or Anthropic source material in the April 2026 timeframe. Isolate those endpoints from the network immediately. Revoke active sessions and rotate credentials for any accounts accessible from those machines, prioritizing cloud provider credentials, GitHub tokens, npm tokens, and VPN/SSO accounts.
2. Step 2, Detection: Query EDR telemetry for Rust-compiled dropper execution on developer workstations (look for newly spawned processes from user-writable directories, especially following git clone or archive extraction activity). Search browser history and DNS logs for access to GitHub repositories associated with 'idbzoomh' or repos using title patterns referencing 'Claude Code leak', 'Claude source', or 'Anthropic npm'. Check for GhostSocks SOCKS5 proxy activity: unexpected outbound connections on non-standard ports from developer hosts. Monitor for Vidar behavioral indicators: bulk reads of browser profile directories (Login Data, Cookies, Local State), access to cryptocurrency wallet files, and staging of files in temp directories prior to exfiltration. Review npm audit logs for packages installed from non-official sources around the March 31-April 2026 window.
3. Step 3, Eradication: For confirmed-infected endpoints: perform full reimaging; do not attempt in-place cleanup of infostealer infections. Rotate all credentials that were stored in browsers or credential managers on affected systems. Revoke and reissue all API keys, tokens, and certificates accessible from compromised machines. Remove any GitHub repos or forks cloned from the malicious source. Audit npm and Git configurations on developer machines for injected registry redirects or malicious pre/post-install hooks.
4. Step 4, Recovery: After reimaging and credential rotation, verify no unauthorized access occurred against GitHub organization audit logs, cloud provider access logs (AWS CloudTrail, Azure Activity Log, GCP Audit Log), and SSO provider sign-in logs. Confirm no new OAuth applications were authorized or SSH keys added during the compromise window. Monitor previously compromised accounts for 30 days for anomalous access patterns. Validate that GhostSocks proxy traffic is no longer present in network flow data.
5. Step 5, Post-Incident: This campaign exploited the gap between developer curiosity and code integrity verification. Assess and enforce controls: require checksum or signature verification for all externally sourced code (addressing CWE-494). Implement GitHub organization policies that restrict cloning from unvetted external repositories on corporate devices. Evaluate whether developer workstations have EDR coverage equivalent to server infrastructure. Brief development teams on SEO poisoning as a delivery mechanism; this is a recurring pattern that will recur with future high-visibility AI or security news events. Consider adding 'fake repo lure following major AI news' to threat hunt playbook triggers.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO, legal, and breach notification counsel immediately if analysis of GitHub audit logs, CloudTrail, or SSO logs confirms the threat actor used stolen developer credentials to access production systems, customer data repositories, secrets managers (AWS Secrets Manager, HashiCorp Vault), or CI/CD pipelines — any of which would trigger breach notification obligations under GDPR, CCPA, or applicable state privacy law given the potential for downstream customer data exposure via compromised developer access.
Recovery Notes	After reimaging and full credential rotation, prioritize re-validating the integrity of any code committed or CI/CD pipelines triggered by compromised developer accounts during the exposure window — Vidar-compromised developer credentials could have been used to inject malicious commits or tamper with build artifacts before detection. Monitor all previously compromised GitHub accounts, AWS IAM users, and SSO identities for 30 days using the audit log queries defined in Step 4, with particular attention to any OAuth application authorizations or new SSH/deploy keys that persist post-rotation. Validate network flow baselines confirm GhostSocks SOCKS5 tunnel absence for at least 72 hours post-reimaging before declaring containment complete, as GhostSocks can re-establish via residual implant components if reimaging was incomplete.
Forensic Artifacts	Vidar credential staging directory in %TEMP% or %APPDATA%\Roaming\{random 8-char alphanumeric}\ — contains zip archive of harvested browser Login Data (Chrome/Edge SQLite DBs with plaintext-decryptable credentials), session cookies (Cookies SQLite DB), autofill data, and cryptocurrency wallet.dat or seed phrase files; this directory is deleted post-exfiltration but recoverable via $MFT or VSS if captured before overwrite | Sysmon Event ID 1 (Process Create) on developer workstations: Rust-compiled dropper PE spawned from %USERPROFILE%\Downloads\ or git clone destination directory, with CommandLine showing no arguments (Vidar droppers typically execute silently) and ParentImage of git.exe, node.exe, or cmd.exe following archive extraction | DNS query logs or Windows DNS cache ('ipconfig /displaydns') for api.telegram.org with corresponding outbound HTTPS POST traffic from a non-browser process — Vidar historically uses Telegram Bot API as C2 channel to receive configuration and exfiltrate harvested data, making this a high-confidence IOC distinct from normal developer Telegram client traffic | npm cache and log files at %APPDATA%\npm-cache_logs*.log and %USERPROFILE%\.npm_logs\ showing package resolution from non-registry.npmjs.org sources, specifically any packages with postinstall or preinstall hooks that executed PowerShell or curl download cradles during the March 31–April 2026 installation window | GitHub organization audit log entries (exportable via API or Settings UI) for the compromise window showing git clone events, OAuth token creation, SSH key additions, or Actions secret modifications attributable to compromised developer accounts — these are immutable server-side logs not affected by endpoint reimaging and represent the authoritative record of what the threat actor accessed using stolen session tokens

Per-Action IR Details

Step 1: Containment — Identify any developer or researcher workstations that downloaded content from GitHub repositories impersonating Claude Code or Anthropic source material in the April 2026 timeframe. Isolate those endpoints from the network immediately. Revoke active sessions and rotate credentials for any accounts accessible from those machines, prioritizing cloud provider credentials, GitHub tokens, npm tokens, and VPN/SSO accounts.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), CIS 6.2 (Establish an Access Revoking Process), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts)

Compensating: Without EDR, pivot to GitHub org audit logs (Settings → Audit log → filter by actor and date) to enumerate which corporate accounts cloned repos from 'idbzoomh' or repos with 'Claude Code leak' in the title between March 31 and April 30, 2026. Cross-reference DNS query logs or proxy logs for github.com/idbzoomh. Use 'git log --all --remotes' on developer workstations to identify remote URLs pointing to malicious repos. Immediately disable GitHub personal access tokens via: 'gh auth token && gh api user/tokens' or force-revoke via GitHub org admin console. For npm token revocation without a secret scanning tool, run 'npm token list' and 'npm token revoke ' for each developer's npm account.

Evidence: BEFORE isolating the endpoint, capture: full memory dump using WinPmem or LiME (Vidar executes in-memory and stages stolen data in %TEMP% before exfiltration — memory may contain live credential buffers); snapshot of running processes via 'tasklist /v /fo csv > processes.csv'; netstat output 'netstat -ano > netstat.csv' to capture any active GhostSocks SOCKS5 outbound connections; browser profile directories in their current state at %LOCALAPPDATA%\Google\Chrome\User Data\Default\{Login Data, Cookies, Local State} and equivalent Firefox profile at %APPDATA%\Mozilla\Firefox\Profiles*.default-release\{logins.json, cookies.sqlite} — Vidar bulk-reads these and copies them to a staging directory; contents of %TEMP% and %APPDATA%\Roaming for any newly created subdirectories containing aggregated credential files.

Step 2: Detection — Query EDR telemetry for Rust-compiled dropper execution on developer workstations (look for newly spawned processes from user-writable directories, especially following git clone or archive extraction activity). Search browser history and DNS logs for access to GitHub repositories associated with 'idbzoomh' or repos using title patterns referencing 'Claude Code leak', 'Claude source', or 'Anthropic npm'. Check for GhostSocks SOCKS5 proxy activity: unexpected outbound connections on non-standard ports from developer hosts. Monitor for Vidar behavioral indicators: bulk reads of browser profile directories (Login Data, Cookies, Local State), access to cryptocurrency wallet files, and staging of files in temp directories prior to exfiltration. Review npm audit logs for packages installed from non-official sources around the March 31–April 2026 window.

NIST Phase: Detection Analysis

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

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy Sysmon with SwiftOnSecurity config (minimum) and add targeted rules: Event ID 1 (Process Create) filtering on Image path containing \AppData\, \Temp\, or \Downloads\ with ParentImage containing git.exe or node.exe — this catches the Rust dropper spawning after 'git clone' of the malicious repo. Event ID 11 (File Create) filtering on TargetFilename matching *\Chrome\User Data\Default\Login Data* or *wallet.dat* to detect Vidar's credential harvesting file access pattern. For GhostSocks detection without SIEM, run this PowerShell on suspected hosts: 'Get-NetTCPConnection | Where-Object {$_.State -eq "Established" -and $_.RemotePort -notin @(80,443,22,53)} | Select-Object LocalAddress,LocalPort,RemoteAddress,RemotePort,OwningProcess | Export-Csv ghost_socks_check.csv'. For npm install audit: 'cat ~/.npm/_logs/*.log | grep -E "(resolved|added)" | grep -v "registry.npmjs.org"' identifies packages pulled from non-official registries. Use Sigma rule 'proc_creation_win_susp_exec_from_temp.yml' from SigmaHQ as a baseline detection.

Evidence: GitHub Actions/API audit log entries for the 'idbzoomh' actor or repo names matching 'claude-code-leak', 'anthropic-source', 'claude-npm' — export via GitHub API: 'gh api orgs/{org}/audit-log?phrase=idbzoomh&include=git'; Windows Security Event Log Event ID 4688 (Process Creation) on developer hosts showing executable spawning from %TEMP% or %APPDATA% with parent process git.exe or npm.exe; Sysmon Event ID 1 for the Rust-compiled dropper — identifiable by PE characteristics (Rust runtime imports: ntdll, kernel32 with no .NET or VB runtime); DNS query logs for resolution of Vidar C2 infrastructure (historically abuses Telegram API for C2 — look for api.telegram.org queries followed by large outbound POST requests from non-browser processes); npm log files at %APPDATA%\npm-cache_logs\ showing package resolution from non-npmjs.org sources during the March 31–April 2026 window.

Step 3: Eradication — For confirmed-infected endpoints: perform full reimaging; do not attempt in-place cleanup of infostealer infections. Rotate all credentials that were stored in browsers or credential managers on affected systems. Revoke and reissue all API keys, tokens, and certificates accessible from compromised machines. Remove any GitHub repos or forks cloned from the malicious source. Audit npm and Git configurations on developer machines for injected registry redirects or malicious pre/post-install hooks.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control) — implied by config audit requirement, CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 2.3 (Address Unauthorized Software)

Compensating: For teams without enterprise imaging infrastructure: boot from a known-good USB (Windows PE or Linux live ISO), mount the drive read-only, and extract forensic evidence before wiping. For Git config auditing on surviving machines, run: 'git config --global --list | grep -E "(url|insteadof|hook)"' to identify injected URL rewrites; examine ~/.gitconfig and per-repo .git/config for 'url.insteadOf' entries that could redirect future clones to attacker-controlled infrastructure. For npm hook audit: 'cat package.json | python3 -m json.tool | grep -A5 -B5 scripts' on all repos the developer worked in — look for 'preinstall', 'postinstall', or 'prepare' scripts containing curl, wget, PowerShell download cradles, or base64-encoded commands. Revoke GitHub tokens via: 'gh api -X DELETE /applications/{client_id}/token' or organization admin forced revocation.

Evidence: Before reimaging, preserve: full disk image using dcfldd or FTK Imager (chain of custody required if legal action is anticipated); contents of %APPDATA%\Roaming\ and %LOCALAPPDATA% staging directories where Vidar aggregates stolen data before exfiltration (Vidar typically creates a uniquely named subdirectory containing zip archives of harvested credentials); ~/.npmrc and ~/.gitconfig files showing any malicious registry or URL redirects injected by the dropper; Windows Registry export of HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon for persistence mechanisms added by Vidar; list of all SSH keys in ~/.ssh/ and all GPG keys in the developer's keyring — these may have been exfiltrated and must be revoked regardless of whether reimaging destroys the evidence of access.

Step 4: Recovery — After reimaging and credential rotation, verify no unauthorized access occurred against GitHub organization audit logs, cloud provider access logs (AWS CloudTrail, Azure Activity Log, GCP Audit Log), and SSO provider sign-in logs. Confirm no new OAuth applications were authorized or SSH keys added during the compromise window. Monitor previously compromised accounts for 30 days for anomalous access patterns. Validate that GhostSocks proxy traffic is no longer present in network flow data.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-11 (Audit Record Retention), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Without a SIEM, build a targeted monitoring checklist: (1) Query GitHub org audit log daily for 30 days filtering on compromised account usernames for events: repo.create, repo.fork, org.invite_member, oauth_application.create, deploy_key.create, personal_access_token.create — export via: 'gh api "orgs/{org}/audit-log?phrase=actor:{username}&include=all&per_page=100"'; (2) For AWS, run: 'aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue={compromised_user} --start-time {compromise_date}' focusing on IAM CreateAccessKey, AssumeRole, and S3 GetObject events; (3) For GhostSocks validation, use Wireshark or tcpdump on the network perimeter with filter: 'tcp and not port 80 and not port 443 and not port 22 and src net {developer_subnet}' — any persistent SOCKS5 tunnel would appear as long-lived TCP sessions on non-standard ports from developer IP ranges.

Evidence: GitHub organization audit log exports covering the full compromise window showing OAuth app authorizations (event type: oauth_application.create or org.oauth_app_access_approved), SSH key additions (deploy_key.create or user.add_public_key), and any new repo secrets or Actions secrets created (org.add_actions_secret); AWS CloudTrail logs filtered for IAM actions by compromised developer IAM users/roles —

specifically CreateAccessKey, AttachUserPolicy, PutUserPolicy — which would indicate the threat actor attempted privilege escalation using stolen AWS credentials from browser-stored developer credentials; SSO provider logs (Okta System Log, Azure AD Sign-in Logs) for the compromised accounts showing sign-ins from IP addresses not associated with corporate egress or the developer's known locations during the compromise window.

Step 5: Post-Incident — This campaign exploited the gap between developer curiosity and code integrity verification. Assess and enforce controls: require checksum or signature verification for all externally sourced code (addressing CWE-494). Implement GitHub organization policies that restrict cloning from unvetted external repositories on corporate devices. Evaluate whether developer workstations have EDR coverage equivalent to server infrastructure. Brief development teams on SEO poisoning as a delivery mechanism — this is a recurring pattern that will recur with future high-visibility AI or security news events. Consider adding 'fake repo lure following major AI news' to threat hunt playbook triggers.

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-7 (Software, Firmware, and Information Integrity), NIST SI-2 (Flaw Remediation), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: For code integrity verification without enterprise tooling: enforce a Git pre-clone hook policy that requires developers to verify repo owner identity against a corporate allowlist before cloning — distribute a shell function 'safe-clone()' that checks the GitHub org namespace against an approved list before executing 'git clone'. For SEO poisoning detection, add a bookmark/DNS sinkhole for github.com/idbzoomh and known campaign repo patterns. Implement a YARA rule targeting Rust-compiled Vidar dropper PE characteristics (MZ header + Rust runtime imports + high entropy sections) deployable via ClamAV on developer machines: scan all files extracted from git archives with 'clamscan -r --detect-pua=yes {extract_path}'. Add the threat hunt trigger 'new high-profile AI tool leak/release + GitHub SEO poisoning' as a standing hypothesis in the hunt playbook, activated whenever major AI news breaks, referencing MITRE ATT&CK T1608.001 (Stage Capabilities: Upload Malware) and T1195.001 (Supply Chain Compromise: Compromise Software Dependencies and Development Tools).

Evidence: Lessons-learned documentation should capture: timeline of how many developer machines accessed the malicious repo before detection (sourced from GitHub audit log and DNS logs); dwell time from initial Vidar execution to network isolation — calculated from earliest Sysmon Event ID 1 for the dropper versus isolation timestamp; list of all credentials confirmed exfiltrated (browser-stored passwords, session cookies, API tokens) based on Vidar's known harvest scope correlated against what was stored in affected browsers; network flow records showing GhostSocks C2 beacon timing and data volume to quantify exfiltration scope; post-incident comparison of developer workstation EDR coverage rate versus server EDR coverage rate to document the detection gap this campaign exploited.

Detection Guidance

Behavioral indicators: (1) Process spawned from a directory created by git clone or archive extraction on a developer workstation, especially if compiled in Rust (check PE/ELF headers or file metadata). (2) Bulk filesystem reads targeting browser profile paths: '%APPDATA%\Local\Google\Chrome\User Data\Default\Login Data', 'Cookies', 'Local State'; Firefox 'logins.json'; equivalent paths for Edge and Brave. (3) File staging in %TEMP% or %APPDATA% subdirectories followed by outbound HTTPS or SOCKS5 connections to non-corporate destinations. (4) GhostSocks proxy activity: SOCKS5 negotiation on atypical ports from workstation-class hosts; look for persistent outbound TCP sessions with irregular keep-alive patterns. Network indicators: Unexpected outbound connections from developer workstations to residential or cloud-hosted IPs on ports other than 80/443. IOC Status: Specific hashes, domains, and IPs are not available in this structured report. Source primary reporting (Zscaler threat intelligence, BleepingComputer) for current IOC lists before ingesting into detection tooling. Log sources to query: EDR process/file telemetry, DNS resolver logs, proxy

logs, GitHub audit log (organization level), npm registry access logs, cloud provider authentication logs.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`github.com/idbzoomh`	GitHub account identified by Zscaler as hosting malicious repositories impersonating Claude Code leaked source	MEDIUM
URL	`https://www.bleepingcomputer.com/news/security/claude-code-leak-used-to-push-infostealer-malware-on-github/`	Primary reporting source — consult for additional IOCs including hashes and C2 infrastructure not included in structured item data	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1555** — Credentials from Password Stores
- **T1204.002** — Malicious File
- **T1566** — Phishing
- **T1195** — Supply Chain Compromise
- **T1555.003** — Credentials from Web Browsers
- **T1608.003** — Install Digital Certificate
- **T1608.001** — Upload Malware
- **T1539** — Steal Web Session Cookie
- **T1090.001** — Internal Proxy
- **T1071** — Application Layer Protocol
- **T1059** — Command and Scripting Interpreter
- **T1195.002** — Compromise Software Supply Chain
- **T1090** — Proxy

NIST-800-53R5

- **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
- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan

- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-7** — Least Functionality
- **CM-3** — Configuration Change Control

OWASP-TOP10-2021

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

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks

HIPAA-SECURITY

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

SOC2-TSC

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

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1555	Credentials from Password Stores	Credential-Access
T1204.002	Malicious File	Execution
T1566	Phishing	Initial-Access
T1195	Supply Chain Compromise	Initial-Access
T1555.003	Credentials from Web Browsers	Credential-Access
T1608.003	Install Digital Certificate	Resource-Development
T1608.001	Upload Malware	Resource-Development
T1539	Steal Web Session Cookie	Credential-Access
T1090.001	Internal Proxy	Command-And-Control
T1071	Application Layer Protocol	Command-And-Control
T1059	Command and Scripting Interpreter	Execution

Technique ID	Technique Name	Tactic
T1195.002	Compromise Software Supply Chain	Initial-Access
T1090	Proxy	Command-And-Control

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/claude-code-leak-use...	T3
Claude Code Source Leaked via npm Packaging Error, Anthropic ...	https://thehackernews.com/2026/04/claude-code-tleaked-via-npm-packa...	T3
Claude Code source code accidentally leaked in NPM package	https://www.bleepingcomputer.com/news/artificial-intelligence/claud...	T3
Claude code just got leaked in npm - r/Anthropic on Reddit	https://www.reddit.com/r/Anthropic/comments/1s8n865/claude_code_jus...	T3
anthropics/claude-code-security-review: An AI-powered ... - GitHub	https://github.com/anthropics/claude-code-security-review	T3

Generated 2026-04-03 06:20 UTC by TJS Security Command Center