INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-04 19:21 UTC

IronWorm Rust-Based Infostealer Exploits npm Trusted Publishing to Compromise 36 Packages in Supply Chain Attack

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0410
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	npm packages (36 trojanized), GitHub Actions CI/CD pipelines, OpenAI API, Anthropic API, AWS credential stores, Exodus cryptocurrency wallet, SSH key material, Linux systems with eBPF kernel support
Published	2026-06-04T11:25:37
Discovery Source	Rss

Executive Summary

A threat actor designated IronWorm compromised 36 npm packages by abusing stolen Trusted Publishing credentials, injecting a Rust-based infostealer that targets AI API keys, AWS credentials, SSH private keys, and cryptocurrency wallet material across developer and CI/CD environments. Organizations that installed affected packages during the exposure window must assume secrets were exfiltrated, including credentials granting access to cloud infrastructure and AI services. Containment was achieved before high-traffic packages were reached, but any environment that consumed a trojanized package version during the exposure window must be treated as compromised.

Technical Analysis

IronWorm is a Rust-based infostealer delivered via trojanized npm package versions published using stolen npm Trusted Publishing tokens (T1195.001, T1078.001). Once installed, the malware deploys an eBPF kernel rootkit (T1014) for persistence and detection evasion on Linux systems with eBPF support. Credential harvesting targets 86 environment variables and 20 credential file types, including OpenAI and Anthropic API keys, AWS credential stores (~/.aws/credentials), SSH private keys (T1552.004), and Exodus cryptocurrency wallet material (T1552.001, T1552.005). Exfiltration routes over the Tor network (T1090.003, T1567.002). The attack chain propagates into downstream CI/CD pipelines and developer environments consuming the packages (T1195.002). A concurrent JavaScript-based attack leveraging binding.gyp (T1059.007) was

documented in parallel Ox Security research, suggesting coordinated supply-chain activity. Relevant CWEs: CWE-312 (cleartext storage of sensitive information), CWE-494 (download of code without integrity check), CWE-522 (insufficiently protected credentials), CWE-295 (improper certificate validation), CWE-506 (embedded malicious code). No CVE ID assigned. Individual affected packages carry a CVSS base of 7.5 (representing code injection risk); campaign severity is qualitatively rated high due to supply chain scope and credential access potential. Source: Ox Security research (ox.security), corroborated by BleepingComputer reporting.

Action Checklist

1. Step 1: Containment, Immediately audit your npm dependency tree for any of the 36 identified trojanized packages. Freeze CI/CD pipeline executions that consume unverified npm packages until all dependencies are confirmed clean. Reference the Ox Security advisory (ox.security/blog/npm-2-0-hack-40-npm-packages-hit-in-major-supply-chain-attack/) for the current confirmed package list. Revoke and rotate all secrets accessible from any environment where affected packages were installed, prioritizing AWS credentials, OpenAI/Anthropic API keys, and SSH private keys.
2. Step 2: Detection, Audit environment variables and credential files across developer workstations and CI/CD runners for the 86 targeted variable names and 20 credential file types documented in the Ox Security report. Check for unexpected outbound connections to Tor exit nodes or .onion-adjacent infrastructure in network logs. On Linux hosts, inspect for unexpected eBPF programs loaded via bpftool or /sys/fs/bpf; correlate with NIST 800-53 AU-2 (Audit Events) and CIS Controls 8.2 (Collect Audit Logs). Review npm publish logs for unexpected version releases from your own packages, which may indicate lateral compromise of your own Trusted Publishing tokens.
3. Step 3: Eradication, Pin all npm dependencies to known-good versions predating the compromise window identified in the Ox Security advisory. Remove the trojanized package versions from local caches, node_modules directories, and CI/CD build caches. On any Linux host where affected packages executed, inspect and purge unexpected eBPF programs not matched against a baseline of known, organization-approved programs. Revoke all npm Trusted Publishing tokens associated with affected packages and reissue under verified ownership. Enforce lockfile integrity checking (npm ci with verified lockfiles) per CIS Controls 2.5 (Inventory and Control of Software Assets).
4. Step 4: Recovery, After rotating credentials, validate that no unauthorized API calls or cloud resource changes occurred using AWS CloudTrail, OpenAI usage logs, and Anthropic API logs during the exposure window. Confirm SSH key rotation across all systems where private keys were accessible. Reintroduce CI/CD pipelines only after verifying clean dependency resolution against a pinned, integrity-checked lockfile. Monitor for re-emergence of eBPF rootkit indicators per NIST 800-53 AU-6 (Audit Record Review, Analysis, and Reporting).
5. Step 5: Post-Incident, This attack exposed gaps in dependency integrity verification, secrets management in CI/CD environments, and eBPF visibility on Linux hosts. Implement software composition analysis (SCA) tooling integrated into CI/CD pipelines. Migrate secrets from environment variables to a secrets manager with short-lived credential issuance (addresses CWE-522). Enforce NIST 800-53 SI-7 (Software, Firmware, and Information Integrity) to include third-party package integrity checks. Evaluate eBPF monitoring controls for Linux production and build hosts. Review npm Trusted Publishing token lifecycle and scope per NIST 800-53 AC-2 (Account Management) and AC-2(7) (Automatic Removal or Disabling of Accounts).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior leadership, legal counsel, and potentially CISA if AWS CloudTrail, OpenAI, or Anthropic API logs confirm unauthorized API calls or cloud resource provisioning using exfiltrated credentials during the exposure window, or if Exodus wallet key material was present in the environment, indicating potential financial loss and triggering breach notification assessment obligations.
Recovery Notes	After credential rotation and pipeline restoration, maintain heightened monitoring of AWS CloudTrail for 'AssumeRole' and 'CreateAccessKey' events and SSH authentication logs for at least 30 days, as IronWorm-exfiltrated credentials may be stockpiled and used days or weeks after initial theft. Verify that all 36 trojanized package versions are delisted from the npm registry and that your own packages have not had unauthorized versions published under compromised Trusted Publishing tokens by checking npm package version history weekly for 60 days. Re-run 'bpftool prog list' on all Linux build and production hosts weekly for 30 days post-eradication to confirm the eBPF rootkit component has not been re-dropped via a residual persistence mechanism not identified during initial eradication.
Forensic Artifacts	node_modules//: compiled Rust infostealer binary embedded in the npm package, recoverable from any host where 'npm install' executed during the compromise window — SHA-256 hash each file for chain-of-custody before deletion | ~/.npm/_logs/ and GitHub Actions runner logs: npm install stdout/stderr capturing the exact trojanized package version resolved and installed, establishing the precise time and scope of payload execution on each affected host | /sys/fs/bpf/ and 'bpftool prog list --json' output: pinned eBPF program objects loaded by the IronWorm Rust payload for kernel-level credential interception — bytecode dump via 'bpftool prog dump xlated' captures the hooking logic before unload | AWS CloudTrail management events filtered on access key IDs issued to CI/CD environments during the exposure window: specifically 'AssumeRole', 'GetCallerIdentity', 'PutObject', and 'RunInstances' events from non-organizational source IPs indicating active use of exfiltrated credentials | Firewall/proxy outbound connection logs filtered for Tor exit node IP ranges and tor2web gateway domains originating from build runner or developer workstation IPs: IronWorm routes exfiltrated credential material (SSH keys, AWS keys, AI API keys, Exodus wallet data) over Tor, making this the definitive network-layer indicator of successful exfiltration

Per-Action IR Details

Step 1: Containment — Immediately audit your npm dependency tree for any of the 36 identified trojanized packages. Freeze CI/CD pipeline executions that consume unverified npm packages until all dependencies are confirmed clean. Reference the Ox Security advisory (ox.security/blog/npm-2-0-hack) for the current confirmed package list. Revoke and rotate all secrets accessible from any environment where affected packages were installed, prioritizing AWS credentials, OpenAI/Anthropic API keys, and SSH private keys.

NIST Phase: Containment

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

Controls: NIST AC-2 (Account Management), NIST AC-3 (Access Enforcement), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Run 'npm ls --all 2>/dev/null | grep -Ff trojanized_packages.txt' against each repo using a plaintext list of the 36 IronWorm package names. For CI/CD runners (GitHub Actions), disable all active workflow runs via 'gh run cancel --repo $(gh run list --status in_progress --json databaseId -q .[].databaseId)'. Immediately issue 'aws iam

delete-access-key' and 'aws iam create-access-key' for every IAM identity whose key material was accessible in the build environment; use 'grep -rn AWS_ACCESS_KEY_ID ~/.bashrc ~/.bash_profile ~/.zshrc .env* /etc/environment' to enumerate exposed variable locations on developer workstations.

Evidence: Before rotating secrets, capture: (1) a full snapshot of ~/.aws/credentials, ~/.ssh/id_rsa* and ~/.ssh/id_ed25519* on each developer workstation to confirm key material was present; (2) CI/CD runner environment variable dumps from GitHub Actions job logs (Settings > Actions > Workflow runs) showing which secrets were injected into the build context during the IronWorm exposure window; (3) AWS CloudTrail 'GetSecretValue', 'AssumeRole', and 'CreateAccessKey' events from the compromised window to establish whether exfiltrated keys were used before rotation; (4) OpenAI and Anthropic API usage logs for requests originating from non-organizational IP ranges during the exposure window.

Step 2: Detection — Audit environment variables and credential files across developer workstations and CI/CD runners for the 86 targeted variable names and 20 credential file types documented in the Ox Security report. Check for unexpected outbound connections to Tor exit nodes or .onion-adjacent infrastructure in network logs. On Linux hosts, inspect for unsigned or unexpected eBPF programs loaded via bpftool or /sys/fs/bpf; correlate with AU-2 (Event Logging) and CIS 8.2 (Collect Audit Logs). Review npm publish logs for unexpected version releases from your own packages, which may indicate lateral compromise of your own Trusted Publishing tokens.

NIST Phase: Detection Analysis

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

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: On each Linux build host run 'sudo bpftool prog list' and 'ls -la /sys/fs/bpf/' to enumerate loaded eBPF programs; flag any with names not matching kernel subsystems or your own tooling. Use 'ss -tnp | grep ESTABLISHED' and 'netstat -an' combined with a Tor exit node IP list (bulk download from https://check.torproject.org/torbulkexitlist — validate the URL before use) to detect active C2 connections. For the 86 targeted env variable names, run 'printenv | grep -iE "OPENAI|ANTHROPIC|AWS_SECRET|SSH_KEY|EXODUS"' across workstations. Use osquery with 'SELECT * FROM process_envs WHERE key IN (...)' on Linux hosts to enumerate live process environment variables matching IronWorm's target list without requiring a SIEM.

Evidence: Before concluding this phase, preserve: (1) output of 'bpftool prog list --json' and 'bpftool map list --json' to document eBPF program state at time of discovery, as the IronWorm Rust payload uses eBPF for kernel-level credential interception; (2) firewall or proxy logs filtered for connections to known Tor exit node IP ranges and any DNS queries for .onion resolver proxies (e.g., tor2web gateways) originating from build runner IP addresses; (3) npm audit log at '~/.npm/_logs/' and the npm registry publish history for your own package namespaces, checking for IronWorm's technique of lateral token abuse to publish trojanized versions of maintainer-owned packages; (4) '/proc/[pid]/maps' and '/proc/[pid]/environ' dumps for any Node.js or npm processes that executed during the suspect window, to capture in-memory credential access before process termination.

Step 3: Eradication — Pin all npm dependencies to known-good versions predating the compromise window identified in the Ox Security advisory. Remove the trojanized package versions from local caches, node_modules directories, and CI/CD build caches. On any Linux host where affected packages executed, inspect and purge unexpected eBPF programs. Revoke all npm Trusted Publishing tokens associated with affected packages and reissue under verified ownership. Enforce lockfile integrity checking (npm ci with verified lockfiles) per CIS 4.6 (Securely Manage Enterprise Assets and Software).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST CM-3 (Configuration Change Control), NIST SI-2 (Flaw Remediation), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Remove trojanized package versions with 'npm cache clean --force' followed by 'rm -rf node_modules package-lock.json' and reinstall from a pinned, integrity-verified lockfile using 'npm ci'. For GitHub Actions, purge runner tool caches via the Actions cache API: 'gh cache delete --repo --all'. To unload a malicious eBPF program on Linux, use 'sudo bpftool prog detach pinned /sys/fs/bpf/' followed by 'sudo rm /sys/fs/bpf/'; verify removal with a second 'bpftool prog list'. Revoke compromised npm Trusted Publishing OIDC tokens by navigating to npmjs.com > package settings > Publishing > remove the affected GitHub Actions environment grant; verify no residual tokens remain with 'npm token list'.

Evidence: Before purging, collect: (1) a copy of the trojanized package's installed files from 'node_modules//' including the compiled Rust binary payload, preserving SHA-256 hashes with 'sha256sum node_modules//**/*' for chain-of-custody documentation; (2) 'bpftool prog dump xlated' output for any suspicious eBPF programs to capture the translated bytecode before unloading, establishing whether kernel-level credential hooking was active; (3) npm package-lock.json and yarn.lock snapshots at time of incident, showing the resolved trojanized version with its integrity hash, as this establishes the exact compromise vector and window for affected package versions.

Step 4: Recovery — After rotating credentials, validate that no unauthorized API calls or cloud resource changes occurred using AWS CloudTrail, OpenAI usage logs, and Anthropic API logs during the exposure window. Confirm SSH key rotation across all systems where private keys were accessible. Reintroduce CI/CD pipelines only after verifying clean dependency resolution against a pinned, integrity-checked lockfile. Monitor for re-emergence of eBPF rootkit indicators per AU-6 (Audit Record Review, Analysis, and Reporting).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-11 (Audit Record Retention), NIST AC-17 (Remote Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Query AWS CloudTrail for anomalous activity during the exposure window using: 'aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=AssumeRole --start-time --end-time ' and separately for 'CreateAccessKey', 'PutObject', 'RunInstances', and 'InvokeModel' (for AWS Bedrock). For SSH key validation, run 'for host in $(cat hosts.txt); do ssh-keyscan $host; done' and compare returned public keys against your pre-incident inventory to confirm old keys are no longer accepted. For eBPF re-emergence monitoring, add a daily cron job: 'bpftool prog list > /var/log/bpf_audit_$(date +%Y%m%d).txt' and diff against the post-eradication clean baseline.

Evidence: Before closing recovery, preserve: (1) AWS CloudTrail event history exported for the full exposure window showing all API calls made with potentially compromised IAM access keys, saved in JSON format as the authoritative record of blast radius; (2) 'authorized_keys' file contents from all SSH-accessible systems before and after key rotation to confirm old IronWorm-exfiltrated public keys cannot re-authenticate; (3) GitHub Actions workflow run logs for all pipelines that executed during the exposure window, capturing which secrets were injected and which steps ran the trojanized npm packages, to bound the scope of potential secret exposure across your CI/CD fleet.

Step 5: Post-Incident — This attack exposed gaps in dependency integrity verification, secrets management in CI/CD environments, and eBPF visibility on Linux hosts. Implement software composition analysis (SCA) tooling integrated into CI/CD pipelines. Migrate secrets from environment variables to a secrets manager with short-lived credential issuance (addresses CWE-522). Enforce CIS 7.1 (Establish and Maintain a Vulnerability Management Process) to include third-party package integrity checks. Evaluate eBPF monitoring controls for Linux production and build hosts. Review npm Trusted Publishing token lifecycle and scope per CIS 5.1 (Establish and Maintain an Inventory of Accounts) and CIS 6.2 (Establish an Access Revoking Process).

NIST Phase: Post Incident

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

Controls: NIST AU-2 (Event Logging), NIST AU-9 (Protection Of Audit Information), NIST AC-6 (Least Privilege), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Integrate free SCA into CI/CD pipelines using 'npm audit --audit-level=high' as a required pipeline gate, supplemented by Syft (free SBOM generator: 'syft dir:. -o cyclonedx-json') to generate a software bill of materials for each build artifact. Replace plaintext environment variable secrets with HashiCorp Vault (open-source) using short-lived dynamic AWS credentials via the Vault AWS secrets engine, eliminating static keys that IronWorm targeted. For eBPF visibility on Linux build hosts without commercial EDR, deploy Falco (open-source) with a rule targeting unexpected 'bpf()' syscalls: 'condition: syscall.type = bpf and not proc.name in (known_bpf_tools)'. Enumerate all npm Trusted Publishing OIDC configurations quarterly using 'npm access list packages ' and audit GitHub Actions environment protections to enforce least-privilege token scope per CIS 5.4.

Evidence: Preserve for lessons-learned and threat intelligence sharing: (1) the complete list of the 36 trojanized npm package names and versions with their malicious integrity hashes, to contribute to community blocklists and seed future SCA policy; (2) network flow data showing IronWorm's exfiltration destinations (Tor exit nodes, any clearnet C2 IPs identified) for submission to CISA and relevant ISACs; (3) the Rust binary payload extracted from node_modules (if recovered) for submission to VirusTotal and static analysis to extract the full list of 86 targeted environment variable names, enabling more complete detection rule development; (4) a timeline mapping npm package install events to first observed eBPF program load and first observed outbound Tor connection, establishing IronWorm's post-execution dwell-time pattern for future threat hunting hypotheses.

Detection Guidance

Primary indicators: (1) Outbound connections to Tor network infrastructure from developer workstations, CI/CD runners, or build servers; check firewall and proxy logs for connections to known Tor entry/guard nodes or unusual SOCKS5 traffic patterns. (2) Unexpected eBPF programs on Linux hosts; run 'bpftool prog list' and compare against a known-good baseline, flagging any eBPF programs not matched against organization-approved kernel patches or security utilities. (3) Environment variable access patterns; on hosts with auditd enabled, query for mass reads of credential-related environment variables (search for OpenAI_API_KEY, ANTHROPIC_API_KEY, AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and SSH_AUTH_SOCK in audit logs). (4) Unexpected npm publish events; review npm audit logs for version releases not initiated by a human operator during the exposure window. (5) Filesystem access to ~/.aws/credentials, ~/.ssh/id_rsa, and Exodus wallet files from node or npm processes; flag via endpoint telemetry or auditd file access rules (NIST 800-53 AU-3, Content of Audit Records). (6) binding.gyp-triggered native build steps in packages that do not typically require native compilation; flag in CI/CD build logs as a secondary indicator of the concurrent JavaScript-based attack variant. Confidence on Tor exfil pattern: high (per Ox Security). Confidence on eBPF rootkit indicator: high (per campaign description). Specific package list: retrieve from Ox Security advisory at ox.security/blog/npm-2-0-hack-40-npm-packages-hit-in-major-supply-chain-attack/ or contact Ox Security directly for current enumeration.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`Tor network exfiltration infrastructure (specific nodes not publicly disclosed)`	IronWorm routes harvested credentials over the Tor network per Ox Security reporting; monitor for outbound Tor connections from build and developer hosts	HIGH

Type	Value	Context	Confidence
HASH	`[not available — specific malware hashes not published in available sources]`	Rust-based IronWorm binary; hashes not disclosed in T3-tier sources reviewed; obtain from Ox Security advisory directly	LOW
URL	`https://www.ox.security/blog/npm-2-0-hack-40-npm-packages-hit-in-major-supply-chain-attack/`	Ox Security primary campaign advisory — authoritative source for confirmed package list and IOCs; human validation recommended as URL is T3-tier search-retrieved	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1608.003** — Install Digital Certificate
- **T1136** — Create Account
- **T1078.001** — Default Accounts
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1090.003** — Multi-hop Proxy
- **T1053** — Scheduled Task/Job
- **T1059.007** — JavaScript
- **T1078** — Valid Accounts
- **T1195.002** — Compromise Software Supply Chain
- **T1014** — Rootkit
- **T1567** — Exfiltration Over Web Service
- **T1552.001** — Credentials In Files
- **T1567.002** — Exfiltration to Cloud Storage
- **T1552.004** — Private Keys
- **T1552.005** — Cloud Instance Metadata API

NIST-800-53R5

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

- **CM-3** — Configuration Change Control
- **SC-8** — Transmission Confidentiality and Integrity
- **SC-17** — Public Key Infrastructure Certificates
- **SR-2** — Supply Chain Risk Management Plan
- **SI-4** — System Monitoring

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures
- **A02:2021** — Cryptographic Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **5.2** — Use Unique Passwords
- **3.10** — Encrypt Sensitive Data in Transit
- **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

- **164.308(a)(5)(ii)(D)** — Password Management
- **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
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1608.003	Install Digital Certificate	Resource-Development

Technique ID	Technique Name	Tactic
T1136	Create Account	Persistence
T1078.001	Default Accounts	Defense-Evasion
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1090.003	Multi-hop Proxy	Command-And-Control
T1053	Scheduled Task/Job	Execution
T1059.007	JavaScript	Execution
T1078	Valid Accounts	Defense-Evasion
T1195.002	Compromise Software Supply Chain	Initial-Access
T1014	Rootkit	Defense-Evasion
T1567	Exfiltration Over Web Service	Exfiltration
T1552.001	Credentials In Files	Credential-Access
T1567.002	Exfiltration to Cloud Storage	Exfiltration
T1552.004	Private Keys	Credential-Access
T1552.005	Cloud Instance Metadata API	Credential-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/new-ironworm-malware...	T3
180+ NPM Packages Hit in Major Supply Chain Attack - OX Security	https://www.ox.security/blog/npm-2-0-hack-40-npm-packages-hit-in-ma...	T3
npm package manager compromised by malware - Facebook	https://www.facebook.com/groups/linux.fans.group/posts/347247545971...	T3
Your AI Gateway Was a Backdoor: Inside the LiteLLM Supply Chain ...	https://www.trendmicro.com/en_us/research/26/c/inside-litellm-suppl...	T3

Source	URL	Tier
A new supply chain attack targeting the Node Package ... - Instagram	https://www.instagram.com/p/DXca2aEEl2x/	T3

Generated 2026-06-04 19:21 UTC by TJS Security Command Center